Findings

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021

Findings of ACL: ACL-IJCNLP 2021

August 1 - 6, 2021

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
`acl@aclweb.org`

ISBN 978-1-954085-54-1

Message from the Program Chairs

Welcome to the *Findings* of ACL: ACL-IJCNLP 2021! To continue the success of *Findings* of ACL: EMNLP 2020, we decided to follow this initiative to produce this accompanying volume, consisting of papers that are not accepted for publication in the main conference, but nonetheless have been assessed by the Program Committee as solid work with sufficient substance, quality and novelty. Out of the $3,350$ full submissions to ACL-IJCNLP 2021, 493 papers were invited to be included in the *Findings*. Thirty-six papers declined the offer, leading to 457 papers (118 short and 339 long) to be published in the *Findings* of ACL: ACL-IJCNLP 2021.

Papers published in *Findings* of ACL count as full publications. They are not assigned a presentation slot in the main conference, but rather are published online in a separate volume in the ACL Anthology. There are a number of motivations for this new publication, from allowing timely work to be published quickly, to being more accepting of solid work, and helping to manage the increasing reviewing burden on the community. To increase the visibility of the *Findings* papers, this year the authors of *Findings* papers can choose to make a 3-minute video to be included in the virtual conference. Our workshop chairs also helped to pair *Findings* papers with ACL-IJCNLP 2021 workshops, and as a result, more than 100 *Findings* papers will be presented at those workshops.

The reviewing process for *Findings* is largely the same as for the main conference and accordingly we wish to thank all involved in ACL-IJCNLP 2021 for their efforts, as detailed in the Preface to the Proceedings of ACL-IJCNLP 2021. We would like to specifically thank:

- The whole **Program Committee** for reviewing the submissions, and in particular, the **Senior Area Chairs** for making paper recommendation decisions for *Findings*.
- The **Ethics Advisory Committee**, chaired by Min-Yen Kan, Malvina Nissim, and Xanda Schofield, for their hard work to ensure that all the accepted *Findings* papers have addressed the ethical issues appropriately.
- The **Publication Co-Chairs**, Jing-Shin Chang, Yuki Arase, and Yvette Graham, for their tremendous effort in making the volume of *Findings* of ACL: ACL-IJCNLP 2021.
- The **Workshop Chairs**, Kentaro Inui and Michael Strube, for connecting *Findings* paper authors with individual workshops for possible presentations.
- The **Program Co-Chairs of EMNLP 2020**, Trevor Cohn, Yulan He and Yang Liu, for sharing their experience with *Findings* papers.

We hope that *Findings* will continue to serve as a companion to future conferences, and become an important venue for excellent, widely-read, and highly cited work in NLP.

Fei Xia, University of Washington
Wenjie Li, The Hong Kong Polytechnic University
Roberto Navigli, Sapienza University of Rome

ACL-IJCNLP 2021 Program Committee Co-Chairs

Table of Contents

Explainable Inference Over Grounding-Abstract Chains for Science Questions

Mokanarangan Thayaparan[†], **Marco Valentino**[†], **André Freitas**[†‡]
Department of Computer Science, University of Manchester, United Kingdom[†]
Idiap Research Institute, Switzerland[‡]
{firstname.lastname}@manchester.ac.uk

Abstract

We propose an explainable inference approach for science questions by reasoning on grounding and abstract inference chains. This paper frames question answering as a *natural language* abductive reasoning problem, constructing *plausible* explanations for each candidate answer and then selecting the candidate with the *best* explanation as the final answer. Our method, *ExplanationLP*, elicits explanations by constructing a weighted graph of relevant facts for each candidate answer and employs a linear programming formalism designed to select the optimal subgraph of explanatory facts. The graphs' weighting function is composed of a set of parameters targeting relevance, cohesion and diversity, which we fine-tune for answer selection via Bayesian Optimisation. We carry out our experiments on the WorldTree and ARC-Challenge datasets to empirically demonstrate the following contributions: (1) ExplanationLP obtains strong performance when compared to transformer-based and multi-hop approaches despite having a significantly lower number of parameters; (2) We show that our model is able to generate plausible explanations for answer prediction; (3) Our model demonstrates better robustness towards semantic drift when compared to transformer-based and multi-hop approaches.

1 Introduction

Answering science questions remain a fundamental challenge in Natural Language Processing and AI as it requires complex forms of inference, including causal, model-based and example-based reasoning (Jansen, 2018; Clark et al., 2018; Jansen et al., 2016; Clark et al., 2013). Current state-of-the-art (SOTA) approaches for answering questions in the science domain are dominated by transformer-based models (Devlin et al., 2019; Sun et al., 2019). Despite remarkable performance on answer prediction, these approaches are black-box by nature, lacking the capability of providing *explanations* for their predictions (Thayaparan et al., 2020; Miller, 2019; Biran and Cotton, 2017; Jansen et al., 2016).

Explainable Science Question Answering (XSQA) is often framed as a *natural language abductive reasoning* problem (Khashabi et al., 2018; Jansen et al., 2017). Abductive reasoning represents a distinct inference process, known as *inference to the best explanation* (Peirce, 1960; Lipton, 2017), which starts from a set of complete or incomplete observations to find the hypothesis, from a set of plausible alternatives, that *best* explains the observations. Several approaches (Khashabi et al., 2018; Jansen et al., 2017; Khot et al., 2017a; Khashabi et al., 2016) employ this form of reasoning for multiple-choice science questions to build a set of plausible explanations for each candidate answer and select the one with the best explanation as the final answer.

XSQA solvers typically treat explanation generation as a multi-hop graph traversal problem. Here, the solver attempts to compose multiple facts that connect the question to a candidate answer. These *multi-hop* approaches have shown diminishing returns with an increasing number of hops (Jansen et al., 2018; Jansen, 2018). Fried et al. (2015) conclude that this phenomenon is due to *semantic drift* – i.e., as the number of aggregated facts increases, so does the probability of drifting out of context. Khashabi et al. (2019) propose a theoretical framework, empirically supported by Jansen et al. (2018); Fried et al. (2015), attesting that ongoing efforts with *very long* multi-hop reasoning chains are unlikely to succeed, emphasising the need for a *richer* representation with fewer hops and higher importance to abstraction and grounding mechanisms.

Consider the example in Figure 1A where the central concept the question examines is the understanding of *friction*. Here, an inference solver's

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1–12
August 1–6, 2021. ©2021 Association for Computational Linguistics

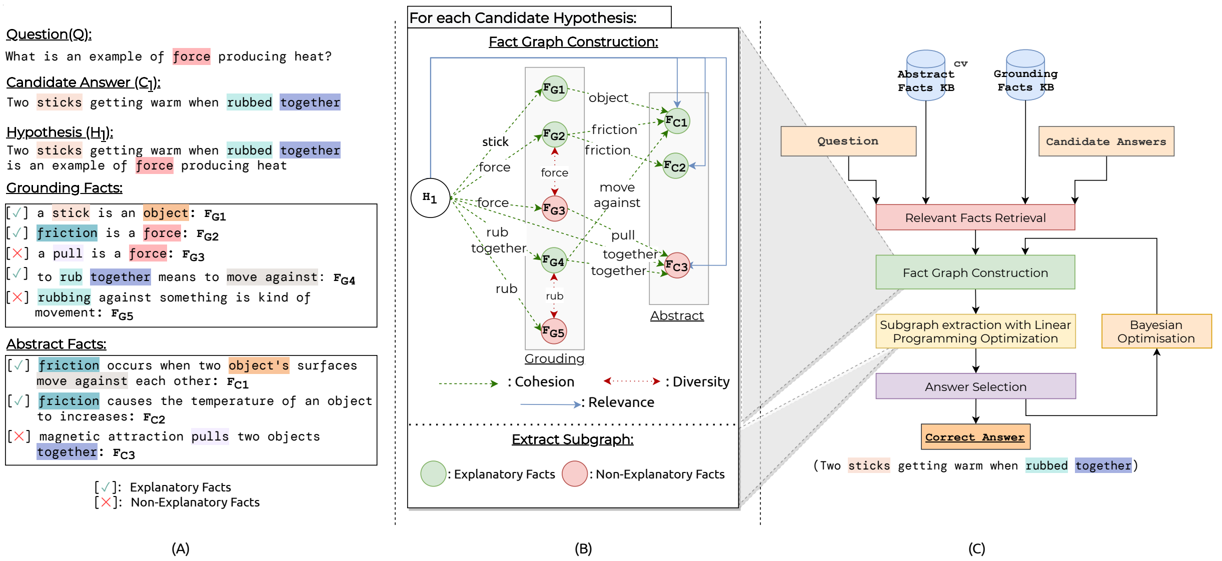

Figure 1: Overview of our approach: (A) Depicts a question, answer and formulated hypothesis along with the set of facts retrieved from a fact retrieval approach (B) Illustrates the optimisation process behind extracting explanatory facts for the provided hypothesis and facts. (C) Details the end-to-end architecture diagram.

challenge is to identify the core scientific facts (**Abstract Facts**) that best explain the answer. To achieve this goal, a QA solver should be able first to go from *force* to *friction*, *stick* to *object* and *rubbing together* to *move against*. These are the **Grounding Facts** that link generic or abstract concepts in a core scientific statement to specific terms occurring in question and candidate answer (Jansen et al., 2018). The grounding process is followed by the identification of the abstract facts about *friction*. A complete explanation for this question would require the composition of five facts to derive the correct answer successfully. However, it is possible to reduce the global reasoning in two hops, modelling it with grounding and abstract facts.

In line with these observations, this work presents a novel approach that explicitly models abstract and grounding mechanisms. The contributions of the paper are:

1. We present a novel approach that performs natural language abductive reasoning via grounding-abstract chains combining Linear Programming with Bayesian optimisation for science question answering (Section 2).

2. We obtain comparable performance when compared to transformers, multi-hop approaches and previous Linear Programming models despite having a significantly lower number of parameters (Section 3.1).

3. We demonstrate that our model can generate plausible explanations for answer prediction (Section 3.2) and validate the importance of grounding-abstract chains via ablation analysis (Section 3.3).

2 ExplanationLP: Abductive Reasoning with Linear Programming

ExplanationLP answers and explains multiple-choice science questions via abductive natural language reasoning. Specifically, the task of answering multiple-choice science questions is reformulated as the problem of finding the candidate answer that is supported by the best explanation. For each Question Q and candidate answer $c_i \in C$, ExplanationLP converts to a hypothesis h_i and attempts to construct a plausible explanation.

Figure 1C illustrates the end-to-end framework. From an initial set of facts selected using a retrieval model, ExplanationLP constructs a fact graph where each node is a fact, and the nodes and edges have a score according to three properties: *relevance*, *cohesion* and *diversity*. Subsequently, an optimal subgraph is extracted using Linear Programming, whose role is to select the best sub-set of facts while preserving structural constraints imposed via grounding-abstract chains. The subgraphs' global scores computed by summing up the nodes and edges scores are adopted to select the final answer. Since the subgraph scores depend on the sum of nodes and edge scores, each property is multiplied by a learnable weight which is optimised via Bayesian Optimisation to obtain the best possible combination with the highest accuracy for answer selection. To the best of our knowledge, we are the first to combine a parameter optimisation method with Linear Programming for inference. The rest of this section describes the model in detail.

2.1 Relevant facts retrival

Given a question (Q) and candidate answers $C = \{c_1,\ c_2,\ c_3, ...,\ c_n\}$ we convert them to hypotheses $\{h_1,\ h_2,\ h_3, ...,\ h_n\}$ using the approach proposed by Demszky et al. (2018). For each hypothesis h_i we adopt fact retrieval approaches (e.g: BM25, Unification-retrieval (Valentino et al., 2021)) to select the top m relevant *abstract* facts $F_A^{h_i} = \{f_1^{h_i}, f_2^{h_i}, f_3^{h_i}, ..., f_m^{h_i}\}$ from a knowledge base containing abstract facts (*Abstract Facts KB*) and top l relevant *grounding* facts $F_G^{h_i} = \{f_1^{h_i}, f_2^{h_i}, f_3^{h_i}, ..., f_l^{h_i}\}$ from a knowledge base containing grounding facts (*Grounding Facts KB*) that *at least* connects one abstract fact with the hypothesis, such that $F^{h_i} = F_A^{h_i} \cup F_G^{h_i}$ and $l+m = k$.

2.2 Fact graph construction

For each hypothesis h_i we build a weighted undirected graph $G^{h_i} = (V^{h_i},\ E^{h_i},\ \omega_v,\ \omega_e)$ with vertices $V^{h_i} \in \{\{h_i\} \cup F^{h_i}\}$, edges E^{h_i}, edge-weight function $\omega_e(e_i; \theta_1)$ and node-weight function $\omega_v(v_i; \theta_2)$ where $e_i \in E^{h_i}$, $v_i \in V^{h_i}$ and $\theta_1, \theta_2 \in [0, 1]$ is a learnable parameter which is optimised via Bayesian optimisation.

The model scores the nodes and edges based on the following *three* properties (See Figure 1B):

(1) Relevance: We promote the inclusion of highly relevant facts in the explanations by encouraging the selection of sentences with higher lexical relevance and semantic similarity with the hypothesis. We use the following scores to measure the relevance and the semantic similarity of the facts:

Lexical Relevance score (L): Obtained from the upstream facts retrieval model (e.g: BM25 score/ Unification score (Valentino et al., 2021)).

Semantic Similarity score (S): Cosine similarity obtained from neural sentence representation models. For our experiments, we adopt Sentence-BERT (Reimers et al., 2019) since it shows state-of-the-art performance in semantic textual similarity tasks.

(2) Cohesion: Explanations should be cohesive, implying that grounding-abstract chains should remain within the same context. To achieve cohesion, we encourage a high degree of overlaps between different hops (e.g. hypothesis-grounding, grounding-abstract, hypothesis-abstract) to prevent the inference chains from drifting away from the original context. The overlap across two hops is quantified using the following scoring function:

Cohesion score (C): We denote the set of unique terms of a given fact $f_i^{h_i}$ as $t(f_i^{h_i})$ after being lemmatized and stripped of stopwords. The overlap score of two facts $f_j^{h_i}$ and $f_j^{h_i}$ is given by:

$$C(f_j^{h_i}, f_k^{h_i}) = \frac{|t(f_j^{h_i}) \cap t(f_k^{h_i})|}{max(|t(f_j^{h_i})|, |t(f_k^{h_i})|)}$$

Therefore, the higher the number of term overlaps, the higher the cohesion score.

(3) Diversity: While maximizing relevance and cohesion between different hops, we encourage diversity between facts of the same type (e.g. abstract-abstract, grounding-grounding) to address different parts of the hypothesis and promote completeness in the explanations. We measure diversity via the following function:

Diversity score (D): We denote the overlaps between hypothesis h_i and the fact $f_i^{h_i}$ as $t_{h_i}(f_i^{h_i}) = t(f_i^{h_i}) \cap t(h_i)$. The diversity score of two facts $f_j^{h_i}$ and $f_j^{h_i}$ is given by:

$$D(f_j^{h_i}, f_k^{h_i}) = -1 \frac{|t_{h_i}(f_j^{h_i}) \cap t_{h_i}(f_k^{h_i})|}{max(|t_{h_i}(f_j^{h_i})|, |t_{h_i}(f_k^{h_i})|)}$$

The goal is to maximise diversity and avoid redundant facts in the explanations. Therefore, if two facts overlap with different parts of the hypothesis, they will have a higher diversity score compared to two facts that overlap with the same part.

Given these premises, the weight functions of the graph is designed as follows:

$$\omega_e(v_j, v_k; \theta_1) = \begin{cases} \theta_{gg} D(v_j, v_k) & v_j, v_k \in F_G^{h_i} \\ \theta_{aa} D(v_j, v_k) & v_j, v_k \in F_A^{h_i} \\ \theta_{ga} C(v_j, v_k) & v_j \in F_G^{h_i}, v_k \in F_A^{h_i} \\ \theta_{qg} C(v_j, v_k) & v_j \in F_G^{h_i}, v_k = h_i \\ \theta_{qa} C(v_j, v_k) & v_j \in F_A^{h_i}, v_k = h_i \end{cases}$$

$$\omega_v(v_i^{h_i}; \theta_2) = \begin{cases} \theta_{lr} L(v_j, h_i) + \theta_{ss} S(v_j, h_i) & v_j \in F_A^{h_i} \\ 0 & v_i \in F_G^{h_i} \\ 0 & v_i = h_i \end{cases}$$

where $\theta_{gg},\ \theta_{aa},\ \theta_{ga},\ \theta_{gq},\ \theta_{qa} \in \theta_1$ and $\theta_{lr},\ \theta_{ss} \in \theta_2$.

2.3 Subgraph extraction with Linear Programming (LP) optimisation

The construction of the explanation graph has to be optimised for the downstream answer selection task. Specifically, from the whole set of facts retrieved by the upstream retrieval models, we need to select the optimal subgraph that maximises the performance of answer prediction. To achieve this goal, we adopt a Linear Programming approach.

The selection of the explanation graph is framed as a rooted maximum-weight connected subgraph problem with a maximum number of K vertices (R-MWCS$_K$). This formalism is derived from the generalized maximum-weight connected subgraph problem (Loboda et al., 2016). R-MWCS$_K$ has two parts: objective function to be maximized and constraints to build a connected subgraph of explanatory facts. The formal definition of the objective function is as follows:

Definition 1. Given a connected undirected graph $G = (V, E)$ with edge-weight function $\omega_e : E \rightarrow \mathbb{R}$, node-weight function $\omega_v : V \rightarrow \mathbb{R}$, root vertex $r \in V$ and expected number of vertices K, the rooted maximum-weight connected subgraph problem with K number of vertices (R-MWCS$_K$) problem is finding the connected subgraph $\hat{G} = (\hat{V}, \hat{E})$ such that $r \in \hat{V}$, $|V| \leq K$ and

$$\Omega(\hat{G};\theta_3) = \theta_{vw} \sum_{v\in\hat{V}} \omega_v(v;\theta_1) + \theta_{ew} \sum_{e\in\hat{E}} \omega_e(e;\theta_2) \rightarrow max$$

where $\theta_{vw},\ \theta_{ew} \in \theta_3$, $\theta_3 \in [0,1]$ and θ_3 is a learnable parameter optimized via Bayesian optimisation. The LP solver will seek to extract the optimal subgraph with the highest possible sum of node and edge weights. Since the solver seeks to obtain the highest possible score, it will avoid negative edges and will prioritise high-value positive edges resulting in higher diversity, cohesion and relevance. We adopt the following binary variables to represent the presence of nodes and edges in the subgraph:

1. Binary variable y_v takes the value of 1 iff $v \in V^{h_i}$ belongs to the subgraph.
2. Binary variable z_e takes the value of 1 iff $e \in E^{h_i}$ belongs to the subgraph.

In order to emulate the grounding-abstract inference chains and obtain a valid subgraph, we impose the set constraints described in Table 1 for the LP solver.

2.4 Bayesian Optimisation for Answer Selection

Given Question Q and choices $C = \{c_1,\ c_2,\ c_3, ...,\ c_n\}$ we extract the optimal explanation graphs $\hat{G}^Q = \{\hat{G}^{c_1},\ \hat{G}^{c_2},\ \hat{G}^{c_3}, ...,\ \hat{G}^{c_n}\}$ for each choice. We consider the hypothesis with the highest relevance, cohesion and diversity to be the correct the answer. Based on this premise we define the correct answer as $c_{ans} = \arg\max_{h_i}\ (\Omega(\hat{G}^{h_i}))$.

In order to automatically optimize the Linear Programming model (i.e, $\theta_1,\ \theta_2,\ \theta_3$) we use Bayesian optimisation. The algorithm is defined as below (Here $\mathcal{GP}$ is Gaussian Process and LP is the Linear Programming module).

Algorithm 1: Bayesian Optimisation

```
$\theta_1, \theta_2, \theta_3$ = initRandom(seed)
$G^Q$ = fact-graph-construction($\omega_e(\theta_1'),\ \omega_v(\theta_2')$)
$\hat{G}^Q$ = LP($G^Q,\ \Omega(\theta_3)$)
$X$ = evaluate-accuracy($G^Q$)
model = $\mathcal{GP}(X, \{\theta_1, \theta_2, \theta_3\})$
iteration = 0
while iteration $\leq N$ do
    $\theta_1', \theta_2', \theta_3'$ = get-next-exploration-point()
    $G^{Q'}$ = fact-graph-construction($\omega_e(\theta_1'),\ \omega_v(\theta_2')$)
    $\hat{G}^{Q'}$ = LP($G^{Q'},\ \Omega(\theta_3')$)
    $X'$ = evaluate-accuracy($G^{Q'}$)
    model.update($X', \{\theta_1', \theta_2', \theta_3'\}$)
    iteration = iteration + 1
end
Result: Best accuracy for model and respective
        parameters $\theta_1, \theta_2, \theta_3$
```

3 Empirical Evaluation

Background Knowledge: We construct the required knowledge bases using the following sources.

(1) Abstract KB: Our Abstract knowledge base is constructed from the WorldTree Tablestore corpus (Xie et al., 2020; Jansen et al., 2018). The Tablestore corpus contains a set of common sense and scientific facts adopted to create explanations for multiple-choice science questions. The corpus is built for answering elementary science questions encouraging possible knowledge reuse to elicit explanatory patterns. We extract the core scientific facts to build the Abstract KB. Core scientific facts

$y_{v_i} = 1 \quad \text{if } v_i = h_i \quad (1)$ $y_{v_i} \leq \sum_j y_{v_j} \quad \forall v_j \in N_{G^{h_i}}(v_i) \quad (2)$ $z_{v_i,v_j} \leq y_{v_i} \quad \forall e_{(v_i,v_j)} \in E \quad (3)$ $z_{v_i,v_j} \leq y_{v_j} \quad \forall e_{(v_i,v_j)} \in E \quad (4)$ $z_{v_i,v_j} \geq y_{v_i} + y_{v_j} - 1 \quad \forall e_{(v_i,v_j)} \in E \quad (5)$	**Chaining constraint**: Equation 1 states that the subgraph should always contain the hypothesis node. Inequality 2 states that if a vertex is to be part of the subgraph, then at least one of its neighbors with a lexical overlap should also be part of the subgraph. Equation 1 and Inequality 2 restrict the LP method to construct explanations that originate from the hypothesis and perform multi-hop aggregation based on the existence of lexical overlap. Inequalities 3, 4 and 5 state that if two vertices are in the subgraph then the edges connecting the vertices should be also in the subgraph. These inequality constraints will force the LP method to avoid grounding nodes with high overlap regardless of their relevance.
$\sum_i y_{v_i} \leq K \quad \forall v_i \in F_A^{h_i} \quad (6)$	**Abstract fact limit constraint**: Equation 6 limits the total number of abstract facts to K. Instead of limiting of total selected number of nodes to K, by limiting the abstract facts we dictate the need for grounding facts based on the number of terms present in the hypothesis and in the abstract facts.
$\sum_{v_j} y_{v_i} - 2 \geq -2(1 - y_{v_j}) \quad \forall v_i \in N_{G^{h_i}}(v_j),\ v_i \in \{F_A^{h_i} \cup h_i\},\ v_j \in F_G^{h_i} \quad (7)$	**Grounding neighbor constraint**: Inequality 7 states that if a grounding fact is selected, then at least two of its neighbors should be either both abstract facts or a hypothesis and an abstract fact. This constraint ensures that grounding facts play the linking role connecting hypothesis-abstract facts.

Table 1: Linear programming constraints employed by ExplanationLP to emulate grounding-abstract inference chains and extract the optimal subgraph

are independent from the specific questions and represent general scientific and commonsense knowledge, such as `Actions` (*friction occurs when two object's surfaces move against each other*) or `Affordances` (*friction causes the temperature of an object to increase*).

(2) Grounding KB: The grounding knowledge base consists of definitional knowledge (e.g., synonymy and taxonomy) that can take into account lexical variability of questions and help it link it to abstract facts. To achieve this goal, we select the *is-a* and *synonymy* facts from ConceptNet (Speer et al., 2017) as our grounding facts. ConceptNet has high coverage and precision, enabling us to answer a wide variety of questions.

Question Sets: We use the following question sets to evaluate ExplanationLP's performance and compare it against other explainable approaches:

(1) WorldTree Corpus: The 2,290 questions in the WorldTree corpus are split into three different subsets: *train-set* (987), *dev-set* (226) and *test-set* (1,077). We use the *dev-set* to assess the explainability performance and robustness analysis since the explanations for *test-set* are not publicly available.

(2) ARC-Challenge Corpus: ARC-Challenge is a multiple-choice question dataset which consists of question from science exams from grade 3 to grade 9 (Clark et al., 2018). We only consider the Challenge set of questions. These questions have proven to be challenging to answer for other LP-based question answering and neural approaches. ExplanationLP rely only on the *train-set* (1,119) and test on the *test-set* (1,172). ExplanationLP does not require *dev-set*, since the possibility of over-fitting is non-existent with only ten parameters.

Relevant Facts Retrieval (FR): We experiment with two different fact retrieval scores. The first model – i.e. *BM25 Retrieval*, adopts a BM25 vector representation for hypothesis and explanation facts. We apply this retrieval for both Grounding and Abstract retrieval. We use the IDF score from BM25 as our downstream model's relevance score. The second approach – i.e. *Unification Retrieval (UR)*, represents the BM25 implementation of the Unification-based Reconstruction framework described in Valentino et al. (2021). The unification score for a given fact depends on how often the same fact appears in explanations for similar questions.

Baselines: The following baselines are replicated on the WorldTree corpus to compare against ExplanationLP:

(1) Bert-Based models: We compare the ExplanationLP model's performance against a set of BERT baselines. The first baseline – i.e. $BERT_{Base}$/$BERT_{Large}$, is represented by a standard BERT language model (Devlin et al., 2019) fine-tuned for multiple-choice question answering. Specifically, the model is trained for binary classification on each question-candidate answer pair to maximize the correct choice (i.e., predict 1) and minimize the wrong choices (i.e., predict 0). During inference, we select the choice with the highest prediction score as the correct answer. BERT baselines are further enhanced with explanatory facts retrieved by the retrieval models. *BERT + BM25* and *BERT + UR*, is fine-tuned for binary classification by complementing the question-answer pair with grounding and abstract facts selected by BM25 and Unification retrieval, respectively.

Similarly, the second model *BERT + UR* complements the question-answer pair with grounding and abstract facts selected using BM25 and Unification retrieval, respectively.

(2) PathNet (Kundu et al., 2019): PathNet is a neural approach that constructs a single linear path composed of two facts connected via entity pairs for reasoning. PathNet also can explain its reasoning via explicit reasoning paths. They have exhibited strong performance for multiple-choice science questions by composing two facts. Similar to Bert-based models, we employ PathNET with the top k facts retrieved utilizing Unification (*PathNet + UR*) and BM25 (*PathNet + BM25*) retrieval. We concatenate the facts retrieved for each candidate answer and provide as supporting facts.

Further details regarding the hyperparameters and code used for each model, along with information concerning the knowledge base construction and dataset information, can be found in the Supplementary Materials.

#	Model	Accuracy Easy	 Challenge
1	BERT$_{Base}$	51.04	28.75
2	BERT$_{Large}$	54.58	29.39
3	BERT$_{Base}$ + BM25 (k=10)	53.92	42.72
4	BERT$_{Large}$ + BM25 (k=10)	54.05	43.45
5	BERT$_{Base}$ + UR (k=10)	52.87	42.17
6	BERT$_{Large}$ + UR (k=10)	58.50	43.72
7	PathNet + BM25 (k=20)	43.32	36.42
8	PathNet + UR (k=15)	47.64	33.55
9	Ours + BM25 (k=30)	63.82	48.24
10	Ours + UR (k=30)	**66.23**	**50.15**

Table 2: Accuracy on Easy (764) and Challenge split (313) of WorldTree *test-set* corpus from the best performing k of each model

#	Model	Explainable	Accuracy
1	BERT$_{Large}$	No	35.11
2	IR Solver (Clark et al., 2016)	Yes	20.26
3	TupleInf (Khot et al., 2017b)	Yes	23.83
4	TableILP (Khashabi et al., 2016)	Yes	26.97
5	DGEM (Clark et al., 2016)	Partial	27.11
6	KG^2 (Zhang et al., 2018)	Partial	31.70
7	ET-RR (Ni et al., 2019)	Partial	36.61
8	Unsupervised AHE (Yadav et al., 2019a)	Partial	33.87
9	Supervised AHE (Yadav et al., 2019a)	Partial	34.47
10	AutoRocc (Yadav et al., 2019b)	Partial	**41.24**
11	Ours + BM25 (k=40)	Yes	**40.21**
12	Ours + UR (k=40)	Yes	39.84

Table 3: ARC challenge scores compared with other Fully or Partially explainable approaches trained *only* on the ARC dataset.

3.1 Answer Selection

WorldTree Corpus: We retrieve the top l relevant grounding facts from Grounding KB and the top m relevant abstract facts from Abstract KB such that $l + m = k$ and $l = m$. To ensure fairness across the approaches, the same amount of facts are presented to each model. We experimented with $k = \{10, 20, 30, 40, 50\}$ and report the accuracy across Easy and Challenge split of the best performing setting in Table 2. We draw the following conclusions:

(1) Despite having a smaller number of parameters to train (BERT$_{Base}$: 110M parameters, BERT$_{Large}$: 340M parameters, ExplanationLP: 9 parameters), the best performing ExplanationLP (#10) overall outperforms all the BERT$_{Base}$ and BERT$_{Large}$ models on both Challenge and Easy split. We outperform the best performing BERT model with facts (BERT$_{Large}$ (#6)) by 7.74% in Easy and 6.43% in Challenge. We also outperform best performing BERT without facts (BERT$_{Large}$ (#2)) by 11.66% in Easy and 20.76% in Challenge.

(2) BERT is inherently a black-box model, not being entirely possible to explain its prediction. By contrast, ExplanationLP is fully explainable and produces a complete explanatory graph.

(3) Similar to ExplanationLP, PathNet is also explainable and demonstrates robustness to noise.

CASE I: All the selected facts are in the gold explanation (**Frequency**: 33%)

Question: A company wants to make a game that uses a magnet that sticks to a board. Which material should it use for the board? **Answer:** steel
Explanations: (1) steel is a metal (*Grounding*), (2) if a magnet is attracted to a metal then that magnet will stick to that metal (*Abstract*), (3) a magnet attracts magnetic metals through magnetism (*Abstract*),

CASE II: At least one selected facts are in the gold explanation (**Frequency**: 58%)

Question: A large piece of ice is placed on the sidewalk on a warm day. What will happen to the ice? **Answer**: It will melt to form liquid water.
Explanations: (1) drop is liquid small amount (*Grounding*), (2) forming something is change (*Grounding*), (3) ice wedging is mechanical weathering (*Grounding*), (4) melting means changing from a solid into a liquid by adding heat energy (*Abstract*), (5) weathering means breaking down surface materials from larger whole into smaller pieces by weather (*Abstract*),

CASE III: No retrieved facts is in the gold explanation (**Frequency**: 9%)

Question:Wind is a natural resource that benefits the southeastern shore of the Chesapeake Bay. How could these winds best benefit humans? **Answer:** The winds could be converted to electrical energy
Explanations: (1) renewable resource is natural resource (*Grounding*), (2) wind is a renewable resource (*Abstract*), (3) electrical devices convert electricity into other forms of energy (*Abstract*)

Table 4: Case study of explanation extracted by ExplanationLP

ExplanationLP also outperforms PathNet's best performance setting (#8) by 18.59% in Easy and 16.60% in Challenge.

(4) ExplanationLP consistently exhibits better scores on both BM25 and UR than BERT and PathNet, demonstrating independence of the upstream retrieval model for performance.

ARC-Challenge : We also evaluated our model on the ARC-Challenge corpus (Clark et al., 2018) to evaluate ExplanationLP on a more extensive general question set and compare against contemporary approaches that provide explanations for an inference that has *only* been trained on ARC corpus. Table 3 reports the results on the *test-set*. We compare ExplanationLP against published approaches that are fully/partly explainable. Here explainability indicates if the model produces an explanation/evidence for the predicted answer. A subset of the approaches produces evidence for the answer but remains intrinsically black-box. These models have been marked as *Partial*.

As depicted in the Table 3, we outperform the best performing fully explainable (#4 TableILP) model by 13.28%. We also outperform specific neural approaches with larger parameter sets (#5 - #9) that provide explanations for their inference and BERT (#1). Despite having a smaller number of training parameters, we also exhibit competitive performance with a state-of-the-art Bert-based approach (#10) that do not use external resources to train the QA system.

3.2 Explainability

Approach	**Precision**	**Recall**	**F1**
PathNet + UR (k=20)	21.56	36.55	29.06
Ours + UR (k=30)	**57.96**	**49.92**	**48.13**

Table 5: Explanation retrieval performance on the WorldTree Corpus *dev-set*.

Table 5 shows the Precision, Recall and F1$_{Macro}$ score for explanation retrieval for PathNet and ExplanationLP. These scores are computed using gold abstract explanations from WorldTree corpus. We outperform PathNet across all spectrum by a significant margin.

Table 4 reports three representative cases that show how explanation generation relates to correct answer prediction. The first example (Case I) represents the situation in which all the selected sentences are annotated as gold explanations in the WorldTree corpus (dev-set). The second example (Case II) shows the case in which at least one sentence in the explanation is labelled as gold. Finally, the third example (Case III) represents the case in which the explanation generated by the method does not contain any gold fact. We observe Case I and Case II occur over 91% of the questions, demonstrating that the correct answers are mostly derived from plausible explanations.

3.3 Ablation Study

In order to understand the contribution lent by different components, we choose the best setting

#	Approach	Accuracy WT	 ARC
1	ExplanationLP **(Best)**	**61.37**	**40.21**
Structure			
2	Grounding-Abstract Categories	58.33	35.13
3	Edge weights	43.78	29.45
4	Node weights	42.80	27.87
Cohesion			
5	Hypothesis-Abstract cohesion	38.71	30.37
6	Hypothesis-Grounding cohesion	59.33	38.73
7	Grounding-Abstract cohesion	59.12	38.14
Diversity			
8	Abstract-Abstract diversity	60.16	37.62
9	Grounding-Grounding diversity	60.44	37.71
Relevance			
10	Hypothesis-Abstract semantic similarity	55.38	35.49
11	Hypothesis-Abstract lexical relevance	54.68	36.01

Table 6: Ablation study, removing different components of ExplanationLP. The scores reported here are accuracy for answer selection on the WorldTree (WT) and ARC-Challenge (ARC) test-set.

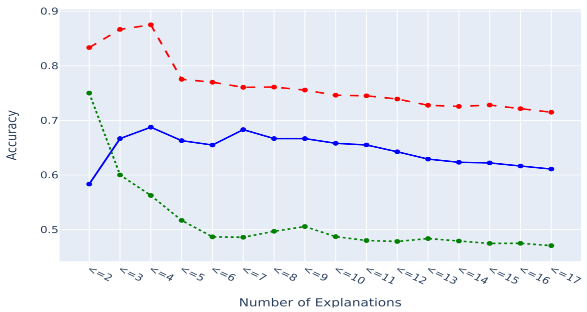

Figure 2: Change in accuracy of answer prediction the development set varying across different models with increasing explanation length for WorldTree *dev-set*. Red dashed line represents ExplanationLP + UR (k=30), blue line represents BERT$_{Large}$ + UR (k=10) and green dotted line represents PathNet + UR (k=20)

(*WorldTree*: ExplanationLP + UR (k=30) and *ARC*: ExplanationLP + BM25 (k=40)) and drop different components to perform an ablation analysis. We retain the ensemble after removing each component. The results are summarized in Table 6.

(1) The grounding-abstract chains (#2) play a significant role, particularly in the reasoning mechanism on a challenging question set like ARC-Challenge.

(2) As observed in #3, #4 removing node weights and edge weights lead to a dramatic drop in performance. This drop indicates that both are fundamental for the final prediction, highlighting the role of graph structure in explainable inference.

(3) The importance of cohesion varies across different types of facts. We observe that Hypothesis-Abstract cohesion (#5) is significantly more important than the others. We attribute this to the fact that without Hypothesis-Abstract cohesion, multi-hop inference can quickly go out of context.

(4) From the ablation analysis, we can see how lexical relevance and semantic similarity (#10, 11) complements each other towards the final prediction. For WorldTree corpus, the relevance score has a higher parameter score translating into a higher impact and vice-versa for ARC.

(5) Diversity plays a smaller role when compared to cohesion and relevance. The impact of diversity in ARC is higher than that of WorldTree.

Semantic Drift To validate the performance across an increasing number of hops, we plot the accuracy against explanation length as illustrated in Figure 2. As demonstrated in explanation regeneration (Valentino et al., 2021; Jansen and Ustalov, 2019), the complexity of a science question is directly correlated with the explanation length – i.e. the number of facts required in the gold explanation. Unlike BERT, PathNet and ExplanationLP use external background knowledge, addressing the multi-hop process in two main reasoning steps. However, in contrast to ExplanationLP, PathNet combines only two explanatory facts to answer a given question. This assumption has a negative impact on answering complex questions requiring long explanations. This is evident in the graph, where we observe a sharp decrease in accuracy with increasing explanation length. Comparatively, ExplanationLP achieves more stable performance, showing a lower degradation with an increasing number of explanation sentences. These results crucially demonstrate the positive impact of grounding-abstract mechanisms on semantic drift. We also exhibit consistently better performance when compared with BERT as well.

4 Related Work

Our approach broadly falls into Linear Programming based approaches for science question answering. LP-based approaches perform inference over either semi-structured tables (Khashabi et al., 2016) or structural representations extracted from the text (Khashabi et al., 2018; Khot et al., 2017a). These approaches treat all facts homogeneously and attempt to connect the question with the correct answer through long hops. While they have exhibited good performance with no supervision, the performance tends to be lower when answering complex questions requiring long explanatory chains. In contrast, our approach performs inference over unstructured text by imposing structural constraints via grounding-abstract chains, lowering the hops, and also combine parametric optimisation to extract the best performing model.

The other class of approaches that provide explanations are graph-based approaches. Graph-based approaches have been successfully applied for open-domain question answering (Fang et al., 2020; Qiu et al., 2019; Thayaparan et al., 2019) where the question only requires only two hops. PathNet (Kundu et al., 2019) operates within the same design principles and has been applied on OpenbookQA science dataset. As indicated in the empirical evaluation, it struggles with long-chain explanations since it relies only on two facts. Graph-based approaches have also been employed for mathematical reasoning (Ferreira and Freitas, 2020a,b) and textual entailment (Silva et al., 2019, 2018).

The third category of partially explainable approaches employs black-box neural models in combination with a retrieval approach. The SOTA model for Science Question (Khashabi et al., 2020) answering is pretrained across multiple datasets and is not explainable. The current partially explainable SOTA approach that does not rely on external resource (Yadav et al., 2019b) employs a large parameter BERT model for question answering resulting. In contrast, with a low number of parameters, we have introduced a model that demonstrates competitive performance and leaves a smaller carbon footprint in terms of energy consumption (Henderson et al., 2020). Other methods construct explanation chains by leveraging explanatory patterns emerging in a corpus of scientific explanations (Valentino et al., 2020, 2021).

5 Conclusion

This paper presented a robust, explainable and efficient science question answering model that performs abductive natural language inference. We also presented an in-depth systematic evaluation demonstrating the impact on the various set of design principles via an in-depth ablation analysis. Despite having a significantly lower number of parameters, we demonstrated competitive performance compared with contemporary explainable approaches while also showcasing its robustness, explainability and interpretability.

Acknowledgements

The authors would like to thank the anonymous reviewers for the constructive feedback. A special thanks to Deborah Ferreira for the helpful discussions and comments. Additionally, we would like to thank the Computational Shared Facility of the University of Manchester for providing the infrastructure to run our experiments.

References

Or Biran and Courtenay Cotton. 2017. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, pages 8–13.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*, pages 2580–2586. Citeseer.

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 37–42.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8823–8838.

Deborah Ferreira and André Freitas. 2020a. Natural language premise selection: Finding supporting statements for mathematical text. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2175–2182.

Deborah Ferreira and André Freitas. 2020b. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374.

Daniel Fried, Peter Jansen, Gustave Hahn-Powell, Mihai Surdeanu, and Peter Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? characterizing knowledge and inference requirements for elementary science exams. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2956–2965.

Peter Jansen, Rebecca Sharp, Mihai Surdeanu, and Peter Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics*, 43(2):407–449.

Peter Jansen and Dmitry Ustalov. 2019. Textgraphs 2019 shared task on multi-hop inference for explanation regeneration. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 63–77.

Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. 2018. Worldtree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Peter A Jansen. 2018. Multi-hop inference for sentence-level textgraphs: How challenging is meaningfully combining information for science question answering? *NAACL HLT 2018*, page 12.

Daniel Khashabi, Erfan Sadeqi Azer, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2019. On the capabilities and limitations of reasoning for natural language understanding. *arXiv preprint arXiv:1901.02522*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2016, pages 1145–1152.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2018. Question answering as global reasoning over semantic abstractions. *Conference of Association for the Advancement of Artificial Intelligence*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1896–1907.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017a. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–316.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017b. Answering complex questions using open information extraction. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Souvik Kundu, Tushar Khot, Ashish Sabharwal, and Peter Clark. 2019. Exploiting explicit paths for multi-hop reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2737–2747.

Peter Lipton. 2017. Inference to the best explanation. *A Companion to the Philosophy of Science*, pages 184–193.

Alexander A Loboda, Maxim N Artyomov, and Alexey A Sergushichev. 2016. Solving generalized maximum-weight connected subgraph problem for network enrichment analysis. In *International Workshop on Algorithms in Bioinformatics*, pages 210–221. Springer.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2019. Learning to attend on essential terms: An enhanced retriever-reader model for open-domain question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 335–344.

Charles Sanders Peirce. 1960. *Collected papers of charles sanders peirce*, volume 2. Harvard University Press.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150.

Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Vivian Dos Santos Silva, Siegfried Handschuh, and André Freitas. 2018. Recognizing and justifying text entailment through distributional navigation on definition graphs. In *AAAI*, pages 4913–4920.

Vivian S Silva, André Freitas, and Siegfried Handschuh. 2019. Exploring knowledge graphs in an interpretable composite approach for text entailment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7023–7030.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. *Proceedings of the 2019 Conference of the North*.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2020. A survey on explainability in machine reading comprehension. *arXiv preprint arXiv:2010.00389*.

Mokanarangan Thayaparan, Marco Valentino, Viktor Schlegel, and André Freitas. 2019. Identifying supporting facts for multi-hop question answering with document graph networks. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 42–51.

Marco Valentino, Mokanarangan Thayaparan, and André Freitas. 2020. Explainable natural language reasoning via conceptual unification. *arXiv preprint arXiv:2009.14539*.

Marco Valentino, Mokanarangan Thayaparan, and Andre Freitas. 2021. Unification-based reconstruction of multi-hop explanations for science questions. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 200–211. Association for Computational Linguistics.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. WorldTree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5456–5473, Marseille, France. European Language Resources Association.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019a. Alignment over heterogeneous embeddings for question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2681–2691.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019b. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2578–2589, Hong Kong, China. Association for Computational Linguistics.

Yuyu Zhang, Hanjun Dai, Kamil Toraman, and Le Song. 2018. Kgˆ 2: Learning to reason science exam questions with contextual knowledge graph embeddings. *arXiv preprint arXiv:1805.12393*.

A Supplementary Material

This section consists of all the hyperparameters, code and libaries used in our approach. We present this in the hope it fosters reproducibility.

A.1 Linear Programming Optimization

The components of the linear programming system is as follows:

- Solver: CPLEX optimization studio V12.9.0 `https://www.ibm.com/products/ilog-cplex-optimization-studio`

The hyperparatemers used in the LP constraints:

- Maximum number of abstract facts (K): 2
- Average time per epoch: 6 minutes for train-set
- Number of Epochs: 200

Infrastructures used:

- CPU Cores: 32
- CPU Model: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
- Memory: 128GB
- OS: Ubuntu 18.04 LTS

A.2 Parameter tuning

Our work employed Bayesian optmization with Gaussian process for hyperparamter tuning. We used the `https://github.com/fmfn/BayesianOptimization`: Bayesian-Optimization python library to implement the code. These parameters are as follows:

- Gaussian Kernels:
  - RationalQuadratic Kernel with default parameters
  - WhiteKernel with noise level of 1e-5, noise level bounds (1e-10, 1e1) and rest of the default parameters
- Number of iterations: 200
- alpha (α): 1e-8
- random state: 1

A.3 Sentence-BERT for Semantic Similarity Scores

We use: *roberta-large nli-stsb mean-tokens* model to calculate the semantic similarity scores.

A.4 BERT model

The BERT model was taken from the Hugginface Transformers (`https://github.com/huggingface/transformers`) library and fine-tuned using 4 Tesla V100 GPUs for 10 epochs in total with batch size 16 for BERT$_{Large}$ and 32 for for BERT$_{Base}$. The hyperparameters adopted for BERT are as follows:

- gradient accumulation steps: 1
- learning rate: 1e-5
- weight decay: 0.0
- adam epsilon: 1e-8
- warmup steps: 0
- max grad norm: 1.0
- seed: 42

A.5 PathNet

We use the code and dependencies provided by the PathNet github repository (`https://github.com/allenai/PathNet`).
We used the training config provided for OpenBookQA as a baseline: `https://github.com/allenai/PathNet`, file name: *blob/master/training_configs/config_obqa.json*.

A.6 Relevant facts retrieval

The code for BM25 and Unification retrieval approaches were adopted from the Unification Explanation Retrieval GitHub repository (`https://github.com/ai-systems/unification_reconstruction_explanations`).

A.7 Code

The code for reproducing the ExplanationLP and the experiments described in this paper are attached with the code appendix and will be available at the following GitHub repository (with a Dockerized container): `https://github.com/ai-systems/explanationlp`.

A.8 Data

WorldTree Dataset : The 2,290 questions in the WorldTree corpus are split into three different subsets: *train-set* (987), *dev-set* (226), and *test-set* (1,077). We only considered questions with explanations for our evaluation. The reasoning behind omitting questions without explanations was to ensure fact coverage for all questions. For AbstractKB building we excluded facts from 'KINDOF' and 'SYNONYMY' table, as these are the one primarily composed of grounding facts.

ARC-Challenge Dataset : Only used the Challenge split: `https://allenai.org/data/arc`.

LV-BERT: Exploiting Layer Variety for BERT

Weihao Yu
National University
of Singapore
weihaoyu6@gmail.com

Zihang Jiang
National University
of Singapore
jzihang@u.nus.edu

Fei Chen
Huawei Noah's
Ark Lab
chen.f@huawei.com

Qibin Hou
National University
of Singapore
andrewhoux@gmail.com

Jiashi Feng
National University
of Singapore
elefjia@nus.edu.sg

Abstract

Modern pre-trained language models are mostly built upon backbones stacking self-attention and feed-forward layers in an interleaved order. In this paper, beyond this stereotyped layer pattern, we aim to improve pre-trained models by exploiting *layer variety* from two aspects: the layer type set and the layer order. Specifically, besides the original self-attention and feed-forward layers, we introduce convolution into the layer type set, which is experimentally found beneficial to pre-trained models. Furthermore, beyond the original interleaved order, we explore more layer orders to discover more powerful architectures. However, the introduced layer variety leads to a large architecture space of more than billions of candidates, while training a single candidate model from scratch already requires huge computation cost, making it not affordable to search such a space by directly training large amounts of candidate models. To solve this problem, we first pre-train a supernet from which the weights of all candidate models can be inherited, and then adopt an evolutionary algorithm guided by pre-training accuracy to find the optimal architecture. Extensive experiments show that LV-BERT model obtained by our method outperforms BERT and its variants on various downstream tasks. For example, LV-BERT-small achieves 78.8 on the GLUE testing set, 1.8 higher than the strong baseline ELECTRA-small. [1]

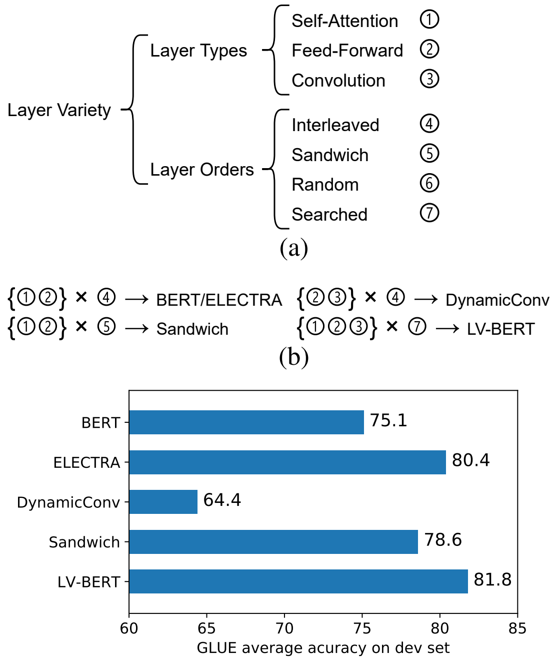

(a)

(b)

(c)

Figure 1: (a) Illustration of layer variety. This concept consists of two aspects: layer type and layer order. (b) Different models represented by layer variety. (c) Performance of different models with hidden size of 256 on GLUE (Wang et al., 2018) development set. Except BERT pre-trained with the Masked Language Modeling objective (Devlin et al., 2019), the other models are pre-trained with Replaced Token Detection objective (Clark et al., 2020) to save computation cost.

1 Introduction

In recent years, pre-trained language models, such as the representative BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020), have gained great success in natural language processing tasks (Peters et al., 2018a; Radford et al., 2018; Yang et al., 2019; Clark et al., 2020). The backbone architectures of these models mostly adopt a stereotyped layer pattern, in which the *self-attention* and *feed-forward* layers are arrayed in an interleaved order (Vaswani et al., 2017). However, there is no evidence supporting that this layer pattern is optimal (Press et al., 2020). We then consider a straightforward and interesting question: Could we change the layer pattern to improve pre-trained models? We attempt to answer this question by exploiting more *layer variety* from two aspects, as shown in Figure 1(a): the layer type set and the layer order.

---

[1] `https://github.com/yuweihao/LV-BERT`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 13–27
August 1–6, 2021. ©2021 Association for Computational Linguistics

We first consider the layer types. In previous pre-trained language models, the most widely-used layer set contains the self-attention layer for capturing global information and the feed-forward layer for non-linear transformation. However, some recent works have unveiled that some self-attention heads in pre-trained models tend to learn local dependencies due to the inherent property of natural language (Kovaleva et al., 2019; Brunner et al., 2020; Jiang et al., 2020), incurring computation redundancy for capturing local information. In contrast, convolution is a local operator (LeCun et al., 1998; Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016) and has shown effectiveness on extracting local information for language models (Zeng et al., 2014; Kim, 2014; Kalchbrenner et al., 2014; Wu et al., 2018, 2019b; Jiang et al., 2020). Thus, we propose to augment the layer set by including convolution for local information extraction.

For layer orders, most of the existing pre-trained models adopt an interleaved order to arrange the different types of layers. Differently, Press et al. (2020) presented the sandwich order, *i.e.*, stacking consecutive self-attention and feed-forward layers at the bottom and top, respectively, while keeping the interleaved order in the middle. It has been shown that the sandwich order can bring improvement on language modeling task, indicating the layer order contributes to model performance. However, Press et al. (2020) did not show the generalization capability of this order to other tasks. There is still a large room for exploring more effective orders for pre-trained models. We show the different layer variety designs of existing models in Figure 1(b), including BERT (Devlin et al., 2019)/ELECTRA (Clark et al., 2020), DynamicConv (Wu et al., 2018) and Sandwich (Press et al., 2020). Their performance is summarized in Figure 1(c). It can be seen that layer variety significantly influences model performance. We thus claim it is necessary to investigate layer variety for promoting pre-trained models. However, to perform such investigation for a common model backbone, *e.g.*, with 24 layers, we need to evaluate performance of every candidate within an architecture space of $3^{24} \approx 2.8 \times 10^{11}$ candidates. Pre-training a single language model already needs to consume a large amount of computation, *e.g.*, 2400 P100 GPU days for pre-training BERT (Lin et al., 2020). It is barely affordable to pre-train such a large amount of model candidates from scratch. To reduce the computation cost, inspired by recent works on Neural Architecture Search (NAS) (Guo et al., 2020; Cai et al., 2019), we construct a supernet according to the layer variety discussed above and pre-train it with Masked Language Modeling (MLM) (Devlin et al., 2019) objective. After obtaining the pre-trained supernet, we develop an evolutionary algorithm guided by MLM evaluation accuracy to search an effective architecture with specific layer variety. We call the resulted model LV-BERT. Extensive experiments show that LV-BERT outperforms BERT and its variants. The contributions of our paper are two-fold. Firstly, to the best of our knowledge, this work is the first to exploit layer variety w.r.t. both layer types and orders for pre-trained language models. We found convolutions and layer orders both benefit pre-trained model performance. We hope our observations would facilitate the development of pre-trained lauguage models. Secondly, our obtained LV-BERT shows superiority over BERT and its variants. For example, LV-BERT-small achieves 79.8 on GLUE testing set, 1.8 higher than the baseline ELECTRA-small (Clark et al., 2020).

2 Related Work

Pre-trained Language Models Pre-trained language models have achieved great success and promoted the development of NLP techniques. Instead of separate word representation (Mikolov et al., 2013a,b), McCann et al. (2017) and Peters et al. (2018b) propose CoVe and ELMo respectively which both utilize LSTM (Hochreiter and Schmidhuber, 1997) to generate contextualized word representations. Later, Radford et al. (2018) introduce GPT that changes the backbone to transformers where self-attention and feed-forward layers are arrayed interleavedly. They also propose generative pre-training objectives. BERT (Devlin et al., 2019) continues to use the same layer set and order for backbone but employs different pre-training objectives, *i.e.*, Masked Language Modeling and Next Sentence Prediction. Then more works introduce new effective pre-training objectives, like Generalized Autoregressive Pretraining (Yang et al., 2019), Span Boundary Objective (Joshi et al., 2020) and Replaced Token Detection (Clark et al., 2020). Besides designing pre-training objectives, some other works try to extend BERT by incorporating knowledge (Zhang et al., 2019; Peters et al., 2019; Liu et al., 2020; Xiong et al., 2020) or with multiple languages (Huang et al., 2019; Conneau and Lample, 2019; Chi et al., 2019). These works utilize the stereotyped layer pattern, which is unnecessarily optimal (Press et al., 2020), inspiring us to further investigate more layer variety to improve pre-trained models. To the best of our knowledge, we are the first to exploit layer variety from both the layer type set and the layer order for pre-trained language models.

Neural Architecture Search Manually designing neural architecture is a time-consuming and error-prone process (Elsken et al., 2019). To solve this, many neural architecture search algorithms are proposed. Pioneering works utilize reinforcement learning (Zoph and Le, 2017; Baker et al., 2017) or evolutionary algorithm (Real et al., 2017) to sample architecture candidates and train them from scratch, which demand huge computation that ordinary researchers can not afford. To reduce computation cost, recent methods (Pham et al., 2018; Liu et al., 2018; Xie et al., 2018; Brock et al., 2018; Cai et al., 2018; Bender et al., 2018; Wu et al., 2019a; Guo et al., 2020) adopt a *weight sharing* strategy that a supernet subsuming all architectures is trained only once and all architecture candidates can inherit their weights from the supernet. Despite the boom of NAS research, most works focus on computer vision tasks (Chen et al., 2019; Ghiasi et al., 2019; Liu et al., 2019a), while NAS on NLP is not fully investigated. Recently, So et al. (2019) and Wang et al. (2020) search architectures of transformers for translation tasks. Chen et al. (2020) leverage differentiable neural architecture to automatically compress BERT with task-oriented knowledge distillation for specific tasks. Zhu et al. (2020) utilize architecture search to improve models based on pre-trained BERT for the relation classification task. However, these methods only focus on specific tasks or the fine-tuning phase. Besides, Khetan and Karnin (2020) employ pre-training loss to help prune BERT, but their method can not find new architectures. Different from them, our work is the first to use NAS to help explore new architectures in a pre-training scenario for general language understanding.

3 Method

An overview of our approach is shown in Figure 2. We first define the layer variety to introduce a large architecture search space, and then pre-train a supernet subsuming all candidate architectures, followed by an evolutionary algorithm guided by pre-training MLM (Devlin et al., 2019) accuracy to search an effective model. In what follows, we will give detailed descriptions.

3.1 Layer Variety

As shown in Figure 1(a), the proposed layer variety contains two aspects: layer type and layer order, both of which are important for the performance of pre-trained models but not exploited before.

Layer Type The layer type set of current BERT-like models consists of self-attention for information communication and feed-forward for non-linear transformation. However, as a global operator, self-attention needs to take as input all tokens to compute attention weights for each token, which is inefficient in capturing local information (Wu et al., 2019b; Jiang et al., 2020). We notice that convolution (LeCun et al., 1998; Krizhevsky et al., 2012), as a local operator, has been successfully applied in language models (Zeng et al., 2014; Kim, 2014; Kalchbrenner et al., 2014; Wu et al., 2018, 2019b; Jiang et al., 2020). A typical example is the dynamic convolution (Wu et al., 2018) for machine translation, language modeling and summarization. Therefore, we augment the layer type set by introducing dynamic convolution as a new layer type. The layer set considered in this work thus contains three types of layers,

$$\mathcal{L}^{\text{type}} = \{L^{\text{SA}}, L^{\text{FF}}, L^{\text{DC}}\}, \quad (1)$$

where the set elements denote self-attention, feed-forward and dynamic convolution layers respectively. See Appendix for more detailed formulation description on them.

Layer Order The other variety aspect is layer order. The most widely-used order for pre-trained models is the interleaved order (Vaswani et al., 2017; Devlin et al., 2019). For a model with 24 layers, the interleaved order can be expressed by the following list,

$$[L_1^{\text{SA}}, L_2^{\text{FF}}, L_3^{\text{SA}}, L_4^{\text{FF}}, ..., L_{23}^{\text{SA}}, L_{24}^{\text{FF}}]. \quad (2)$$

Similarly, the sandwich order (Press et al., 2020) can be expressed as

$$\begin{aligned}&[L_1^{\text{SA}}, L_2^{\text{SA}}, ..., L_5^{\text{SA}},\\ &L_6^{\text{SA}}, L_7^{\text{FF}}, L_8^{\text{SA}}, L_9^{\text{FF}}, ..., L_{18}^{\text{SA}}, L_{19}^{\text{FF}},\\ &L_{20}^{FF}, L_{21}^{FF}, ..., L_{24}^{\text{FF}}].\end{aligned} \quad (3)$$

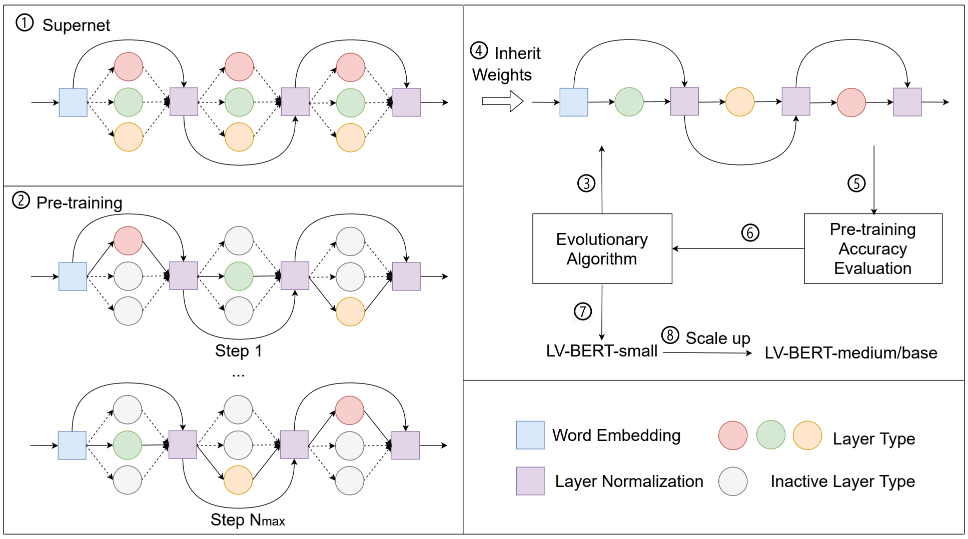

Figure 2: Overview on how to search LV-BERT. ① Construct a supernet with small hidden size by including all types of layers at each layer. ② Pre-train the supernet with Masked Language Modeling (MLM) objective (Devlin et al., 2019) by only uniformly sampling one type of layer into training at each layer. ③ Apply evolutionary algorithm to produce candidate models. ④ The candidate models inherit their weights from the supernet. ⑤ The candidate models with inherited weights are directly evaluated with pre-training MLM accuracy on validation set. ⑥ The accuracy is used to guide the evolutionary algorithm for generating new candidate models. ⑦ After T iterations, the candidate with best pre-training accuracy is output as LV-BERT-small. ⑧ LV-BERT-small can be scaled up to LV-BERT-medium/base with larger hidden size.

Beyond the above manually designed orders, we take advantage of neural architecture search to identify more effective layer orders for pre-trained models. The order to be discovered can be expressed as

$$[L_1, L_2, ..., L_i, ..., L_N], \tag{4}$$

where $L_i \in \mathcal{L}^{\text{type}}$ and N is the number of layers. Here, N is set to 24, following common practice.

3.2 Supernet

The layer variety introduced above leads to a huge architecture space of $3^{24} \approx 2.8 \times 10^{11}$ candidate models to be explored. Thus, it is not affordable to pre-train every candidate model in the space from scratch to evaluate their performance since the pre-training procedure requires huge computations. To reduce the search computations, recent NAS works (Pham et al., 2018; Guo et al., 2020; Cai et al., 2019) exploit a weight sharing strategy. It first trains a supernet subsuming all candidate architectures, and then each candidate architecture can inherit its weights from the trained supernet to avoid training from scratch. Inspired by this strategy, we construct a supernet where each layer contains all types of layers, *i.e.*, self-attention, feed-forward, and dynamic convolution. The supernet architecture can be expressed as

$$\mathcal{A} = [\{L_1^{\text{SA}}, L_1^{\text{FF}}, L_1^{\text{DC}}\}, \{L_2^{\text{SA}}, L_2^{\text{FF}}, L_2^{\text{DC}}\}, ..., \{L_N^{\text{SA}}, L_N^{\text{FF}}, L_N^{\text{DC}}\}]. \tag{5}$$

Masked Language Modeling (MLM) (Devlin et al., 2019) is utilized as the pre-training objective to pre-train the supernet since MLM accuracy can reflect the model performance on downstream tasks (Lan et al., 2020). Most weight sharing approaches on NAS (Wu et al., 2019a; Liu et al., 2018) train and optimize the full supernet: the output of each layer is the weighted sum of all types of candidate layers. However, it cannot guarantee the sampled single type of layer also works (Guo et al., 2020).

To handle this issue, we propose to randomly sample a submodel from the supernet to participate in forward and backward propagation per training step (Cai et al., 2018; Guo et al., 2020). The sampled submodel architecture can be expressed as

$$a = [L_1, L_2, ..., L_i, ..., L_N], \tag{6}$$

where $L_i \in \mathcal{L}^{\text{type}} \sim U$ with uniform probability distribution $Pr = 1/3$. In this pre-training method, the optimized supernet weights can be expressed

Algorithm 1: Evolutionary Search Guided by Pre-training MLM Accuracy

```
Input: $W_{\mathcal{A}}$: supernet weights; $P$: population size; $D_{\text{val}}$: pre-training validation set; $T$: # iteration; $N^{\text{cro}}$: # crossover; $N^{\text{mut}}$: # mutation; $p$: mutation probability; $k$: # top candidates for crossover and mutation
Output: $a^*$: the architecture with the best pre-trianing MLM validation accuracy
$S_0 := \text{Init}(P)$; // Randomly generate $P$ architecture candidates
$S^{\text{topk}} := \emptyset$; // The set of top $k$ candidates
for $i = 1 : T$ do
    $S_{i-1}^{\text{MLM}} := \emptyset$;
    for $a$ in $S_{i-1}$ do
        $\text{MLM}_{\text{val}}^{a} := \text{Inference}(\mathcal{N}(a, W_{\mathcal{A}}(a)), D_{\text{val}})$;
        $S_{i-1}^{\text{MLM}} := S_{i-1}^{\text{MLM}} \cup \text{MLM}_{\text{val}}^{a}$;
    $S^{\text{topk}} := \text{Update}(S^{\text{topk}}, \text{S}_{i-1}, S_{i-1}^{\text{MLM}})$;
    $S^{\text{cro}} := \text{Crossover}(S^{\text{topk}}, N^{\text{cro}})$;
    $S^{\text{mut}} := \text{Mutation}(S^{\text{topk}}, N^{\text{mut}}, p)$;
    $S_i := \text{S}^{\text{cro}} \cup \text{S}^{\text{mut}}$;
return $a^* = \text{argmax}_{a \in S^{\text{topk}}} \text{MLM}_{\text{val}}^{a}$;
```

as

$$W_{\mathcal{A}} = \underset{W}{\text{argmin}}\, \mathbb{E}_{a \sim U(A)}[\mathcal{L}_{\text{pre-train}}(\mathcal{N}(a, W(a)))], \tag{7}$$

where $W(a)$ denotes the submodel weights inherited from the supernet, $\mathcal{N}$ means the submodel with specific architecture and weights, $\mathcal{L}_{\text{pre-train}}$ denotes the pre-training MLM loss and $a \sim U(\mathcal{A})$ means a is uniformly sampled from $\mathcal{A}$.

3.3 Evolutionary Search

Inspired by the recent NAS works (Elsken et al., 2019; Ren et al., 2020; Guo et al., 2020; Wang et al., 2020), we adopt an evolutionary algorithm (EA) to search the model. Previously Real et al. (2017) utilized an evolutionary method in NAS but they trained each candidate model from scratch which is costly and inefficient. Instead, thanks to the supernet mentioned above, we do not need to train the candidate models from scratch since their weights can be inherited from the supernet. Next problem is how to select indicator of the candidate models to guide the EA. Note that our goal is to search a general pre-trained model to benefit a variety of downstream tasks instead of a specific task. Traditional NAS methods (Chen et al., 2020; Zhu et al., 2020) use downstream task performance as the objective to search for task-specific models. Instead, similar to the work by Khetan and Karnin (2020) that utilize pre-training loss to prune BERT, our method uses pre-training MLM accuracy to search for a unified architecture that can generalize well to different downstream tasks. Besides, using this accuracy, candidate models can be directly evaluated on pre-training validation set without any fine-tuning on specific tasks, which can help save computations.

The detailed algorithm description is shown in Algorithm 1. $\text{Crossover}(S^{\text{topk}}, N^{\text{cro}})$ means the procedure to generate N^{cro} new candidate architectures that two candidate architectures randomly selected from top k candidate set S^{topk} are crossed to produce a new one. Similarly, $\text{Mutation}(S^{\text{topk}}, N^{\text{mut}}, p)$ denotes the procedure to generate N^{mut} new candidates that a random candidate from S^{topk} mutates its every layer choice with probability p to generate a new one. Finally, the candidate architecture with highest pre-training validation accuracy in S^{topk} is returned as LV-BERT. The algorithm is set with population size P of 50, search iteration number T of 20, crossover number N^{cro} of 25, mutation number M^{mut} of 25, mutation probability p of 0.1, top candidate number k of 10 for crossover and mutation.

4 Experiments

4.1 Datasets

Pre-training Datasets Devlin et al. (2019) propose WikiBooks corpus for training BERT including English Wikipedia and BooksCorpus (Zhu et al., 2015). However, BooksCorpus is no longer publicly available. To ease reproduction, we train models on OpenWebText (Gokaslan and Cohen, 2019) that is open-sourced and of similar size with the corpus used by BERT. When pre-training the supernet, we leave 2% data as our validation set for evolutionary search.

Fine-tuning Datasets To compare our model with other pre-trained models, we fine-tune LV-BERT on GLUE (Wang et al., 2018), including various tasks for general language understanding, and SQuAD 1.1/2.0 (Rajpurkar et al., 2016, 2018) for question answering. See Appendix for more details of all tasks.

Model	**Layer Variety**				**Params**		**GLUE**
	DC	SA	FF	Order	Word Emb	Backbone	
BERT-small (Devlin et al., 2019)		✓	✓	Interleaved	3.9M	9.5M	75.1
ELECTRA-small (Clark et al., 2020)		✓	✓	Interleaved		9.5M	80.4
DynamicConv-small* (Wu et al., 2018)	✓		✓	Interleaved		9.6M	64.4
Sandwich-small* (Press et al., 2020)	✓		✓	Sandwich		9.5M	78.6
LV-BERT-small variants		✓	✓	Random	3.9M	9.5M	80.8
		✓	✓	Randomly searched		9.8M	81.1
		✓	✓	EA searched		10.3M	81.2
	✓		✓	Random		9.6M	64.9
	✓		✓	Randomly searched		9.6M	65.4
	✓		✓	EA searched		9.6M	65.7
	✓	✓		Random		6.4M	79.7
	✓	✓		Randomly searched		6.4M	79.9
	✓	✓		EA searched		6.4M	79.8
	✓	✓	✓	Random		7.7M	80.6
	✓	✓	✓	Randomly searched		8.8M	80.9
LV-BERT-small	✓	✓	✓	EA searched	3.9M	8.5M	**81.8**

Table 1: Performance of the models with different layer types and orders on the GLUE development set. DC, SA and FF denote dynamic convolution, self-attention and feed-forward layers respectively. For each design of layer type set, "Random" means the best order among five randomly generated ones that are estimated by training model from scratch. "Randomly searched" or "EA searched" are both based on the supernet. "Randomly searched" denotes the orders searched at random while "EA searched" denotes ones searched by evolutionary algorithm. * denotes the methods implemented by us for language pre-training. All models are pre-trained on OpenWebText by 1M steps with sequence length 128 using ELECTRA (Clark et al., 2020) pre-training objective except BERT-small using MLM objective.

Model	**Size**	**Params**		**CoLA**	**MPRC**	**MNLI**	**SST**	**RTE**	**QNLI**	**QQP**	**STS**	**Avg.**
		Word Emb	Backbone									
ELECTRA (Clark et al., 2020)	Small	3.9M	9.5M	56.8	87.4	78.9	88.3	68.5	87.9	88.3	86.8	80.4
	Medium*	3.9M	21.3M	61.2	89.5	82.1	89.1	65.7	88.9	90.5	89.3	82.0
	Base*	23.4M	85.0M	64.8	88.5	85.7	92.6	76.5	91.7	**91.1**	89.9	85.1
DynamicConv† (Wu et al., 2018)	Small	3.9M	9.6M	60.2	69.2	56.6	85.6	49.5	68.0	82.1	44.1	64.4
	Medium	3.9M	21.4M	61.5	67.9	55.7	85.9	49.1	68.3	83.3	51.6	65.4
	Base	23.4M	85.2M	62.1	70.6	61.0	88.5	51.3	72.0	85.6	64.7	69.5
Sandwich† (Press et al., 2020)	Small	3.9M	9.5M	53.2	87.1	77.5	88.1	63.9	86.4	88.3	84.6	78.6
	Medium	3.9M	21.3M	55.6	86.2	81.5	90.3	63.0	88.9	89.6	86.6	80.2
	Base	23.4M	85.0M	58.8	89.7	83.8	91.9	72.6	90.2	90.1	88.5	83.2
LV-BERT	Small	3.9M	8.5M	62.3	86.9	81.1	89.9	69.0	88.9	89.3	87.4	81.8
	Medium	3.9M	19.0M	64.4	88.0	82.4	90.5	68.6	89.4	90.1	89.7	82.9
	Base	23.4M	75.7M	**66.8**	**90.3**	**86.3**	**93.2**	**76.9**	**92.3**	90.9	**90.8**	**85.9**

Table 2: Performance of different models in different sizes on GLUE development set. * denotes results obtained by running official code. † denotes the methods implemented by us for language pre-training. All models are pre-trained on OpenWebText by 1M steps with sequence length 128 using ELECTRA (Clark et al., 2020) pre-training objective.

4.2 Implementation Details

Model Size Similar to Devlin et al. (2019), Clark et al. (2020) and Jiang et al. (2020), we define different model sizes, *i.e.*, "small", "medium" and "base", with the same layer number of 24 but different hidden sizes of 256, 384, and 768, respectively. The detailed hyperparameters are shown in Appendix.

Pre-training Supernet To reduce training cost, we construct the supernet only in small size. Since the layer number of models in medium and base sizes are the same as that of the small-sized one, the obtained architecture of LV-BERT-small can be easily scaled up to the ones of medium and base sizes. We use Adam (Kingma and Ba, 2015) to pre-train the supernet with MLM loss (Devlin et al., 2019) , learning rate of 2e-4, batch size of 128, max sequence length of 128 and pre-training step number of 2 million. See Appendix for more details.

Evaluation Setup To compare with other pre-trained models, we pre-train the searched LV-BERT architecture for 1M steps from scratch on the OpenWebText (Gokaslan and Cohen, 2019) using Replaced Token Detection (Clark et al., 2020) since it can save computation cost. We fine-tune LV-BERT on GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016, 2018) downstream tasks with most hyperparameters the same as those of ELECTRA (Clark et al., 2020) for fair comparison. For GLUE tasks, the evaluation metrics are Matthews correlation for CoLA, Spearman correlation for STS, and accuracy for other tasks, which are averaged to get GLUE score. We utilize evaluation metrics of Exact-Match (EM) and F1 for SQuAD 1.1/2.0. Some of the fine-tuning datasets are small, and consequently, the results may vary substantially for different random seeds. Similar to ELECTRA (Clark et al., 2020), we report the median of 10 fine-tuning runs from the same pre-trained model for each result. See Appendix for more evaluation details.

4.3 Ablation Study

Layer Variety Various models are constructed with different layer variety designs, and their results on GLUE development set are shown in Table 1. For the layer types, if only two layer types are provided, selecting self-attention and feed-forward yields the best result, which can always achieve performance higher than 80 under different search methods. With only dynamic convolution and feed-forward, the performance drops dramatically to around 65. Surprisingly, without feed-forward, the layer set of dynamic convolution and self-attention can still achieve relatively good score, near 80. When using all the three layer types, we can obtain the best 81.8 score, 1.4 higher than the strong baseline ELECTRA (80.4) and 0.6 higher than the model searched with only self-attention and feed-forward (81.2). This indicates that it is effective to augment the layer type set by including convolution to extract local information for pre-trained models.

For layer orders, with the same layer types, the models with either EA or randomly searched orders perform better than those with randomly sampled orders, reflecting the importance of investigating layer orders. For example, with the same layer types of self-attention and feed-forward, the EA searched model obtains 81.2 score, improving BERT/ELECTRA by 6.1/0.8 as well as Sandwich by 2.6.

Search Method Table 1 shows the results with different search methods. "Random" means for each design of layer type set, the order is the best one among 5 randomly generated orders that are estimated by training models from scratch. "Randomly searched" and "EA searched" are both supernet-based methods, in which the weights of candidate models are inherited from the supernet. "Randomly searched" produces candidate models at random for estimation while "EA searched" generates candidate models with evolutionary algorithm guided by the pre-training MLM accuracy. With the same layer types, EA searched orders are generally better than randomly searched ones while the randomly searched ones are generally better than random ones. Figure 3 plots the pre-trianing MLM evaluation accuracy over search iterations with both random and evolutionary search methods. It shows that the accuracy of evolutionary search is obviously higher than that of random search, demonstrating the effectiveness of evolutionary search.

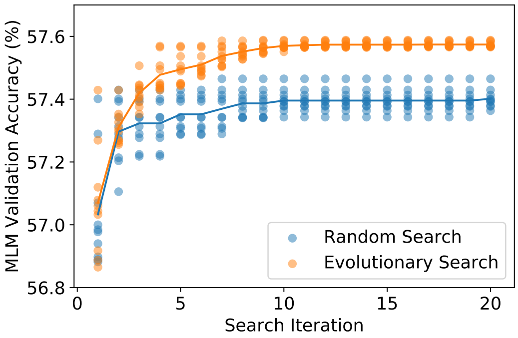

Figure 3: The pre-training MLM validation accuracy comparison between random search and evolutionary search with the layer set of all three types of layers. Blue and yellow dots denote the accuracy of top 10 candidates for each method respectively, while the plots mean their averages.

4.4 LV-BERT Architecture

As shown in Table 1, LV-BERT achieves the best performance. Its architecture is

$$\begin{aligned}[&L_1^{\mathrm{DC}}, L_2^{\mathrm{DC}}, L_3^{\mathrm{SA}}, L_4^{\mathrm{FF}}, L_5^{\mathrm{FF}}, L_6^{\mathrm{SA}},\\ &L_7^{\mathrm{DC}}, L_8^{\mathrm{FF}}, L_9^{\mathrm{FF}}, L_{10}^{\mathrm{SA}}, L_{11}^{\mathrm{DC}}, L_{12}^{\mathrm{DC}},\\ &L_{13}^{\mathrm{SA}}, L_{14}^{\mathrm{FF}}, L_{15}^{\mathrm{DC}}, L_{16}^{\mathrm{FF}}, L_{17}^{\mathrm{SA}}, L_{18}^{\mathrm{DC}},\\ &L_{19}^{\mathrm{FF}}, L_{20}^{\mathrm{DC}}, L_{21}^{\mathrm{SA}}, L_{22}^{\mathrm{SA}}, L_{23}^{\mathrm{FF}}, L_{24}^{\mathrm{SA}}].\end{aligned} \tag{8}$$

Pre-trained with MLM from scratch by 1M steps (sequence length 128) on OpenWebText, LV-BERT-small can achieve 61.2% MLM accuracy while BERT-small is 60.4%. More specific architectures of the models in Table 1 are listed in Appendix.

Model	Train FLOPs	Params	CoLA	MPRC	MNLI	SST	RTE	QNLI	QQP	STS	Avg.
TinyBERT* (Jiao et al., 2020)	6.4e19+ (54x+)	15M	51.1	82.6	84.6	93.1	70.0	90.4	89.1	83.7	80.6
MobileBERT* (Sun et al., 2020)	6.4e19+ (54x+)	25M	51.1	84.5	84.3	92.6	70.4	91.6	88.3	84.8	81.0
ELECTRA-small (Clark et al., 2020)	1.4e18 (1.2x)	14M	54.6	83.7	79.7	89.1	60.8	87.7	88.0	80.2	78.0
GPT (Radford et al., 2018)	4.0e19 (33x)	117M	45.4	75.7	82.1	91.3	56.0	88.1	88.5	80.0	75.9
BERT-base (Devlin et al., 2019)	6.4e19 (54x)	110M	52.1	84.8	84.6	93.5	66.4	90.5	89.2	85.8	80.9
ELECTRA-base (Clark et al., 2020)	6.4e19 (54x)	110M	59.7	86.7	85.8	93.4	73.1	**92.7**	89.1	87.7	83.5
LV-BERT-small	1.2e18 (1x)[†]	13M	57.2	84.1	81.0	90.4	64.6	88.9	88.2	83.8	79.8
LV-BERT-medium	3.1e18 (2.6x)[†]	23M	60.1	85.0	82.0	91.4	67.6	89.7	88.9	85.9	81.3
LV-BERT-base	1.8e19 (15x)[†]	100M	**64.0**	**87.9**	**86.4**	**94.7**	**77.0**	92.6	**89.5**	**88.8**	**85.1**

Table 3: Performance of models with similar size on GLUE testing set. * denotes knowledge distillation methods that rely on large pre-trained teacher models and are orthogonal to other methods. [†] We set the sequence length as 128 for pre-training to save computation although it hurts the performance.

Model	Train FLOPs	Params	SQuAD 1.1		SQuAD 2.0	
			EM	F1	EM	F1
DistillBERT* (Sanh et al., 2019)	6.4e19+ (54x+)	52M	71.8	81.2	60.6	64.1
TinyBERT* (Jiao et al., 2020)	6.4e19+ (54x+)	15M	72.7	82.1	65.3	68.8
MobileBERT* (Sun et al., 2020)	6.4e19+ (54x+)	25M	83.4	90.3	77.6	80.2
ELECTRA-small[†] (Clark et al., 2020)	1.4e18 (1.2x)	14M	74.3	81.8	66.8	69.4
BERT-base (Devlin et al., 2019)	6.4e19 (54x)	110M	80.7	88.4	74.2	77.1
ELECTRA-base (Clark et al., 2020)	6.4e19 (54x)	110M	84.5	**90.8**	80.5	83.3
LV-BERT-small	1.2e18 (1x)[‡]	13M	77.1	84.1	71.0	73.7
LV-BERT-medium	3.1e18 (2.6x)[‡]	23M	79.6	86.4	74.9	77.5
LV-BERT-base	1.8e19 (15x)[‡]	100M	**84.8**	**90.8**	**80.9**	**83.7**

Table 4: Performance of models with similar model size on SQuAD 1.1/2.0 development set. * denotes knowledge distillation methods that rely on large pre-trained teacher models and are orthogonal to other methods. [†] denotes results obtained by running official code. [‡] We set the sequence length as 128 for pre-training to save computation although it hurts the performance.

When running the evolutionary method with different seeds, we see that the resulting models prefer stacking dynamic convolutions at the bottom two layers for extracting local information and self-attention at the top layer to fuse the global information. According to these observation, for ELECTRA-small, if we replace the bottom two layers with dynamic convolutions or the top layer with self-attention, the performance can be improved by 0.3 or 0.5 respectively on GLUE development set. If we replace the bottom 8 layers with manually designed 'ccsfccsf' ('c', 's' and 'f' denote dynamic convolution, self-attention and feed-forward layers, respectively) and replace the top 8 layers with manually designed 'ssfsssfs' together, we observe 0.7 performance improvement. These results show that it is helpful to stack dynamic convolution at the bottom and self-attention at the top.

4.5 Generalization to Larger Models

We only investigate layer variety and search models in a small-sized setting to save computation cost. It is interesting to know whether the searched models can be generalized to larger models with large hidden size. The results are shown in Table 2. For larger model size "medium" and "base", LV-BERTs still outperform other baseline models, demonstrating the good generalization in terms of model size.

4.6 Comparison with State-of-the-arts

We compare LV-BERT with state-of-the-art pre-trained models (Radford et al., 2018; Devlin et al., 2019; Clark et al., 2020; Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020) on GLUE testing set and SQuAD 1.1/2.0 to show its advantages. Although more pre-training data/steps and lager model size can significantly help improve performance (Yang et al., 2019; Liu et al., 2019b; Lan et al., 2020), due to the computation resource limit, we only pre-train our models in small/medium/base sizes for 1M steps with OpenWebText (Gokaslan and Cohen, 2019). We leave evaluating models with more pre-training data/steps and larger model size for future work. We also list some knowledge distillation methods for comparison. However, note that these methods rely on a pre-trained large teacher network and thus are orthogonal to LV-BERT and other methods.

Table 3 presents the performance of LV-BERT and other pre-trained models on GLUE testing set. It shows that LV-BERT outperforms other pre-trained models with similar model size. Remarkably, LV-BERT-small/base achieve 79.8/85.1, 1.8/1.6 higher than strong baselines ELECTRA-small/base. Even compared with knowledge distillation based model MobileBERT (Sun et al., 2020), LV-BERT-medium still outperforms it by 0.3.

Since there is nearly no single model submission on SQuAD leaderboard[2], we only compare LV-BERT with other pre-trained models on the development sets. The results are shown in Table 4. We find that LV-BERT-small outperforms ELECTRA-small significantly, like F1 score 73.7 versus 69.4 on SQuAD 2.0. However, when we generalize LV-BERT-small to base size, the gap between LV-BERT and ELECTRA with base size is narrower than that with small size. One reason may be LV-BERT-small is searched by our method while LV-BERT-base is only generalized from LV-BERT-small with larger hidden size.

5 Conclusion

We are the first to exploit layer variety for improving pre-trained language models, from two aspects, *i.e.*, layer types and layer orders. For layer types, we augment the layer type set by including convolution for local information extraction. For layer orders, beyond the stereotyped interleaved one, we explore more effective orders by using an evolutionary based search algorithm. Experiment results show our obtained model LV-BERT outperforms BERT and its variants on various downstream tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-100E/-2019-035). Jiashi Feng was partially supported by MOE2017-T2-2-151, NUS_ECRA_FY17_P08 and CRP20-2017-0006. The authors also thank Quanhong Fu and Jian Liang for the help to improve the technical writing aspect of this paper. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg). Weihao Yu would like to thank TPU Research Cloud (TRC) program and Google Cloud Research Credits Program for the support of computational resources.

References

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Andrew Brock, Theodore Lim, James Millar Ritchie, and Nicholas J Weston. 2018. Smash: One-shot model architecture search through hypernetworks. In *6th International Conference on Learning Representations*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers. In *International Conference on Learning Representations*.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2019. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*.

Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and

---

[2] rajpurkar.github.io/SQuAD-explorer/

crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Daoyuan Chen, Yaliang Li, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, and Jingren Zhou. 2020. Adabert: Task-adaptive BERT compression with differentiable neural architecture search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2463–2469. ijcai.org.

Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. 2019. Detnas: Neural architecture search on object detection. *arXiv preprint arXiv:1903.10979*, 1(2):4–1.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs.

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2019. Cross-lingual natural language generation via pre-training. *arXiv preprint arXiv:1909.10481*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7057–7067.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7036–7045.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2485–2494.

Zi-Hang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. *Advances in Neural Information Processing Systems*, 33.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online*

Event, 16-20 November 2020, pages 4163–4174. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

Ashish Khetan and Zohar Karnin. 2020. schubert: Optimizing elements of bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2807–2818.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Jiahuang Lin, Xin Li, and Gennady Pekhimenko. 2020. Multi-node bert-pretraining: Cost-efficient approach. *arXiv preprint arXiv:2008.00177*.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. 2019a. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 82–92.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. In *International Conference on Learning Representations*.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proc. of NAACL*.

Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104.

Ofir Press, Noah A. Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2996–3005. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2020. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2158–2170. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7675–7688, Online. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019a. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2018. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2019b. Lite transformer with long-short range attention. In *International Conference on Learning Representations*.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. Snas: stochastic neural architecture search. In *International Conference on Learning Representations*.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Wei Zhu, Xiaoling Wang, Xipeng Qiu, Yuan Ni, and Guotong Xie. 2020. Autorc: Improving bert based relation classification models via architecture search. *arXiv preprint arXiv:2009.10680*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

A Details about Layer Types

For a layer, assume its input is $I \in \mathbb{R}^{s\times c}$ and output is $O \in \mathbb{R}^{s\times c}$, where s is the sequence length and c is the hidden size (channel dimension). For simplicity, c takes the same value for the input and output.

Self-Attention The self-Attention layer, also known as multi-head self-attention (Vaswani et al., 2017), transforms the input by three linear transformations into the key K, query Q and value V vectors respectively,

$$\begin{aligned} K &= \mathrm{Reshape}(IW^{\mathrm{K}} + b^{\mathrm{K}}) \\ Q &= \mathrm{Reshape}(IW^{\mathrm{Q}} + b^{\mathrm{Q}}) \\ V &= \mathrm{Reshape}(IW^{\mathrm{V}} + b^{\mathrm{V}}), \end{aligned} \tag{9}$$

where $K, Q, V \in \mathbb{R}^{h\times s\times d}$, $W^{\mathrm{K}}, W^{\mathrm{Q}}, W^{\mathrm{V}} \in \mathbb{R}^{c\times c}$, and $b^{\mathrm{K}}, b^{\mathrm{Q}}, b^{\mathrm{V}} \in \mathbb{R}^{c}$. Notice that $h \times d = c$ where h is the number of heads and d is the head dimension.

The above K and Q are used to compute their similarity matrix M which is then used to generate new value V':

$$\begin{aligned} M &= \mathrm{Softmax}(KQ^{\top}/\sqrt{d}) \\ V' &= \mathrm{Reshape}(MV), \end{aligned} \tag{10}$$

where $M \in \mathbb{R}^{h\times s\times s}$ and $V' \in \mathbb{R}^{s\times c}$. Finally, a linear transformation is used to exchange information between different heads, followed by shortcut connection and layer normalization,

$$O = \mathrm{Norm}(V'W_O + b_O + I), \tag{11}$$

where $W_O \in \mathbb{R}^{c\times c}$ and $b_O \in \mathbb{R}^{c}$.

Feed-Forward The feed-forward layer (Vaswani et al., 2017) includes two linear transformations with a non-linear activation, followed by a shortcut connection and layer normalization,

$$\begin{aligned} N &= \mathrm{GELU}(IW_1 + b_1) \\ O &= \mathrm{Norm}(NW_2 + b_2 + I), \end{aligned} \tag{12}$$

where $W_1 \in \mathbb{R}^{c\times rc}$ and $W_2 \in \mathbb{R}^{rc\times c}$ with a ratio r. $\mathrm{GELU}(\cdot)$ denotes the Gaussian Error Linear Unit (Hendrycks and Gimpel, 2016).

Dynamic Convolution Dynamic convolution is introduced by Wu et al. (2018) to replace self-attention, which shows strong competitiveness in the tasks of machine translation, language modeling and summarization. The dynamic convolution first uses gated linear unit (GLU) (Dauphin et al., 2017) to generate new representation,

$$V = \mathrm{GLU}(I). \tag{13}$$

Different from the vanilla dynamic convolution that directly generates dynamic kernel from $V \in \mathbb{R}^{s\times c}$, in this work, we supplement a separate convolution (Howard et al., 2017) with depthwise weights $W^{\mathrm{Dep}} \in \mathbb{R}^{k\times c}$ (k is the convolution kernel size, set as 9 in this paper) and pointwise weights $W^{\mathrm{Poi}} \in \mathbb{R}^{c\times c}$ to extract local information to help the following kernel generation. Denoting the output as $S \in \mathbb{R}^{s\times c}$, the separate convolution can be formulated as

$$S_{i,:} = \left(\sum_{j=1}^{k} W^{\mathrm{Dep}}_{j,:} \cdot V_{i+j-\frac{k+1}{2},:} \right) W^{\mathrm{Poi}}. \tag{14}$$

Then the output of separate convolution is used to generate dynamic kernels,

$$D = \mathrm{Softmax}(\mathrm{Reshape}(SW^{\mathrm{Dyn}})), \tag{15}$$

where $W^{Dyn} \in \mathbb{R}^{c \times hk}$ and $D \in \mathbb{R}^{h \times s \times k}$. Then lightweight convolution is applied to the reshaped $V' = \mathrm{Reshape}(V) \in \mathbb{R}^{h \times s \times d}$. The output $C \in \mathbb{R}^{h \times s \times d}$ can be expressed as

$$C_{p,i,:} = \sum_{j=1}^{k} D_{p,i,j} \cdot V'_{p,i+j-\frac{k+1}{2},:}. \tag{16}$$

Finally, C is reshaped to $C' = \mathrm{Reshape}(C) \in \mathbb{R}^{s \times c}$ and a linear transformer is applied to fuse the information among multiple heads, followed by a short connection and layer normalization,

$$O = \mathrm{Norm}(C' W^{\mathrm{Out}} + b^{\mathrm{Out}} + I), \tag{17}$$

where $W^{\mathrm{Out}} \in \mathbb{R}^{c \times c}$ and $b^{\mathrm{Out}} \in \mathbb{R}^{c}$.

B Details about Datasets

B.1 GLUE Dataset

Introduced by Wang et al. (2018), General Language Understanding Evaluation (GLUE) benchmark is a collection of nine tasks for natural language understanding, where testing set labels are hidden and predictions need to be submitted to the evaluation server[3]. We provide details about the GLUE tasks below.

CoLA The Corpus of Linguistic Acceptability (Warstadt et al., 2019) is a binary single-sentence classification dataset for predicting whether an sentence is grammatical or not. The samples are from books and journal articles on linguistic theory.

MRPC The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a dataset for the task to predict whether two sentences are semantically equivalent or not. It is extracted from online news sources with human annotations.

MNLI The Multi-Genre Natural Language Inference Corpus (Williams et al., 2018) is a dataset of sentence pairs. Each pair has a premise sentence and a hypothesis sentence, requiring models to predict its relationships containing *ententailment*, *contradiction* or *neutral*. It is from ten distinct genres of spoken and written English.

SST The Stanford Sentiment Treebank (Socher et al., 2013) is a dataset for the task to predict whether a sentence is positive or negative in sentiment. The dataset is from movie reviews with human annotations.

Hyperparameter	Supernet	Small	Medium	Base
Layer number	24	24	24	24
Word emb. size	128	128	128	768
Hidden size	256	256	384	768
FF inner hidden size	1024	1024	1536	3072
Generator size	N/A	1/4	1/3	1/3
Head number	4	4	6	12
Head size	64	64	64	64
Learning rate	2e-4	5e-4	5e-4	2e-4
Learning rate decay	Linear	Linear	Linear	Linear
Warmup steps	10000	10000	10000	10000
Adam ϵ	1e-6	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999	0.999
Dropout	0.1	0.1	0.1	0.1
Batch size	128	128	128	256
Input sequence length	128	128	128	128

Table 5: Pre-training hyperparameters. Generator size means the multiplier for hidden size, feed-forward inner hidden size and head number to construct generator for Replaced Token Detection pre-trianing objective (Clark et al., 2020).

Hyperparameter	Value
Learning rate	3e-4 for small/medium size 1e-4 (except 2e-4 for SQuAD) for base size
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Layerwise LR decay	0.8 for every two layers
Learning rate decay	Linear
Warmup fraction	0.1
Attention Dropout	0.1
Dropout	0.1
Weight eecay	0.01
Batch size	32
Train epochs	10 for RTE and STS, 2 for SQuAD, and 3 for other tasks

Table 6: Fine-tuning hyperparameters.

RTE The Recognizing Textual Entailment (RTE) dataset is for the task to determine whether the relationship of a pair of premise and hypothesis sentences is entailment. The dataset is from several annual textual entailment challenges including RTE1 (Dagan et al., 2005), RTE2 (Haim et al., 2006), RTE3 (Giampiccolo et al., 2007), and RTE5 (Bentivogli et al., 2009).

QNLI Question Natural Language Inference is a dataset converted from The Stanford Question Answering Dataset (Rajpurkar et al., 2016). An example is a pair of a context sentence and a question, requiring to predict whether the context sentence contains the answer to the given question.

QQP The Quora Question Pairs dataset (Chen et al., 2018) is the dataset from Quora, requiring to

---

[3] `https://gluebenchmark.com`

Model	Layer Variety				Architecture	GLUE
	DC	SA	FF	Order		
BERT-small		✓	✓	Interleaved	[1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]	75.1
ELECTRA-small		✓	✓	Interleaved	[1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]	80.4
DynamicConv-small*	✓		✓	Interleaved	[0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2]	64.4
Sandwich-small*	✓		✓	Sandwich	[1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2]	78.6
LV-BERT-small variants		✓	✓	Random	[1, 1, 1, 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1]	80.8
		✓	✓	Randomly searched	[1, 2, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1]	81.1
		✓	✓	EA searched	[1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2]	81.2
	✓		✓	Random	[2, 0, 0, 2, 2, 0, 0, 2, 2, 2, 0, 2, 2, 0, 0, 0, 2, 2, 0, 2, 2, 0, 0, 0]	64.9
	✓		✓	Randomly searched	[2, 2, 0, 2, 2, 0, 2, 2, 2, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 2]	65.4
	✓		✓	EA searched	[0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2]	65.7
	✓	✓		Random	[0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1]	79.7
	✓	✓		Randomly searched	[0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0]	79.9
	✓	✓		EA searched	[0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1]	79.8
	✓	✓	✓	Random	[1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 2, 2, 2, 1, 0, 1, 0, 1, 0, 2, 2, 1]	80.6
	✓	✓	✓	Randomly searched	[1, 1, 0, 2, 0, 1, 2, 0, 2, 2, 1, 2, 0, 1, 2, 0, 2, 2, 0, 0, 1, 1, 2, 1]	80.9
LV-BERT-small	✓	✓	✓	EA searched	[0, 0, 1, 2, 2, 1, 0, 2, 2, 1, 0, 0, 1, 2, 0, 2, 1, 0, 2, 0, 1, 1, 2, 1]	**81.8**

Table 7: Architectures of different models and their performance on GLUE development set. In Architecture column, 0, 1, and 2 denote dynamic convolution, self-attention, and feed-forward layers respectively * denotes methods implemented by us for language pre-training.

determine whether a pair of questions are semantically equivalent or not.

STS The Semantic Textual Similarity Benchmark (Cer et al., 2017) is a collection of sentence pairs with human-annotated similarity score on a 1-5 scare.

WNLI Winograd NLI (Levesque et al., 2012) is a small dataset for natural language inference. However, there are issues with the construction of this dataset[4]. Therefore, this dataset is exclude in this paper for comparison as BERT (Devlin et al., 2019) *etc*.

B.1.1 SQuAD dataset

The Stanford Question Answering Dataset (**SQuAD 1.1**) (Rajpurkar et al., 2016) is a dataset of more than 100K questions which all can be answered by locating a span of text from the corresponding context passage. Besides this data, the upgraded version **SQuAD 2.0** (Rajpurkar et al., 2018) supplements it with over 50K unanswerable questions.

C Pre-training Details

For supernet, We pre-train it for 2M steps with hyperparameters listed in Table 5, using Masked Language Modeling (MLM) pre-training objective (Devlin et al., 2019). This objective masks 15% input tokens that require the model to predict. The reason to use this objective is that the MLM validation accuracy can reflect the performance of models on downstream tasks (Lan et al., 2020).

For pre-training LV-BERTs and other compared baselines like DynamicConv (Wu et al., 2018) and Sandwich (Press et al., 2020) from scratch, we utilize Replaced Token Detection (RTE) pre-training objective (Clark et al., 2020). This objective employs a small generator to predict masked tokens and utilize a larger discriminator to determine predicted tokens from the generator are the same as original ones or not. RTE can help save computation cost but achieve good performance (Clark et al., 2020). We pre-train the models for 1M steps, mostly using the same hyperparameters as ELECTRA (Clark et al., 2020). We set the pre-training sequence length 128 that can help us save computation cost. For downstream task SQuAD 1.1/2.0 that needs longer input sequence length, we pre-train more 10% steps with the sequence length of 512 to learn the position embedding before fine-tuning. The hyperparameters are listed in Table 5.

D Fine-tuning Details

For fine-tuning on downstream tasks, most of the hyperparameters are the same as ELECTRA (Clark et al., 2020). See Table 6.

E Searched Architectures

The different searched architectures are listed in Table 7.

---

[4] `https://gluebenchmark.com/faq`

Few-Shot Event Detection with Prototypical Amortized Conditional Random Field

Xin Cong[1,2] **Shiyao Cui**[1,2] **Bowen Yu**[1,2]
Tingwen Liu[1,2*] **Yubin Wang**[1,2] and **Bin Wang**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences. Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences. Beijing, China
[3]Xiaomi AI Lab, Xiaomi Inc., Beijing, China
`{congxin, cuishiyao, yubowen}@iie.ac.cn`
`{liutingwen, wangyubin}@iie.ac.cn`
`wangbin11@xiaomi.com`

Abstract

Event detection tends to struggle when it needs to recognize novel event types with a few samples. The previous work attempts to solve this problem in the identify-then-classify manner but ignores the trigger discrepancy between event types, thus suffering from the error propagation. In this paper, we present a novel unified model which converts the task to a few-shot tagging problem with a double-part tagging scheme. To this end, we first propose the **P**rototypical **A**mortized **C**onditional **R**andom **F**ield (PA-CRF) to model the label dependency in the few-shot scenario, which approximates the transition scores between labels based on the label prototypes. Then Gaussian distribution is introduced for modeling of the transition scores to alleviate the uncertain estimation resulting from insufficient data. Experimental results show that the unified models work better than existing identify-then-classify models and our PA-CRF further achieves the best results on the benchmark dataset FewEvent. Our code and data are available at `http://github.com/congxin95/PA-CRF`.

1 Introduction

Event detection (ED) systems extract events of specific types from the given text. Traditionally, researchers use pipeline approaches (Ahn, 2006) where a trigger identification (TI) system is used to identify event triggers in a sentence and then a trigger classifier (TC) is used to find the event types of extracted triggers. Such a framework makes the task easy to conduct but ignores the interaction and correlation between the two subtasks, being susceptible to cascading errors. In the last few years, several neural network-based models were proposed to jointly identify triggers and classify event types from a sentence (Chen et al., 2015; Nguyen and Grishman, 2015, 2018; Liu et al., 2018; Yan et al., 2019; Cui et al., 2020b,a). These models have achieved promising performance and proved the effectiveness of solving ED in the joint framework. But they almost followed the supervised learning paradigm and depended on the large-scale human-annotated dataset, while new event types emerge every day and most of them suffer from the lack of sufficient annotated data. In the case of insufficient resources, existing joint models cannot recognize the novel event types with only few samples, i.e., Few-Shot Event Detection (FSED).

Marry	It served as the location of Bogart's [wedding] to Bacall.
E-Mail	If you have a better idea, please [e-mail] me.

Figure 1: An example from FewEvent dataset revealing the trigger discrepancy. "[·]" marks the event trigger.

One intuitive way to solve this problem is to first identify event triggers in the conventional way and then classify the event types based on the few-shot learning (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018), these two subtasks can be trained jointly by parameter sharing. Such identify-then-classify paradigm (Deng et al., 2020) seems to be convincing because TI aims to recognize triggers and does not need to adapt to novel classes, so we just need to solve the TC in the few-shot manner. Unfortunately, our preliminary experiments reveal that TI tends to struggle when recognizing triggers of novel event types because novel events usually contain completely different triggers with the semantic distinction from the known events, i.e., **Trigger discrepancy** problem. Figure 1 gives an example that the trigger "e-mail" would only occur in event *E-Mail* but not in *Marry* and triggers of two events have disparate context. And experiments on FewEvent (a benchmark dataset for FSED) show that 59.21% triggers in the test set do not trigger

---

[*] Corresponding Author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 28–40
August 1–6, 2021. ©2021 Association for Computational Linguistics

any events in the training set and the F1 score of TI with the SOTA TI model BERT-tagger (Yang et al., 2019) is only 31.06%. Thus, the performance of the identify-then-classify paradigm will be limited by the TI part due to the cascading errors.

In this paper, we present a new unified method to solve FSED. Specifically, we convert this task to a sequence labeling problem and design a double-part tagging scheme using trigger and event parts to describe the features of each word in a sentence. The key to the sequence labeling framework is to model the dependency between labels. Conditional Random Field (CRF) is a popular choice to capture such label dependency by learning transition scores of fixed label space in the training dataset. Nevertheless, in FSED, CRF cannot be applied directly due to the **label discrepancy** problem, that is the label space of the test set is non-overlapping with the training set since FSED aims to recognize novel event types. Therefore, the learned transition scores of CRF from the training set do not model the dependency of the novel labels in the test set.

To address the label discrepancy problem, we propose **P**rototypical **A**mortized **C**onditional **R**andom **F**ield (PA-CRF), which approximates the transition scores based on the label prototypes (Snell et al., 2017) instead of learning by optimization. Specifically, we first apply the self-attention mechanism to capture the dependency information between labels and then map the label prototype pairs to the corresponding transition scores. In this way, PA-CRF can produce label-specific transition scores based on the few supportive samples, which can adapt to arbitrary novel event types. However, predicting the transition score as a single fixed value actually acts as the point estimation, which usually acquires a large amount of annotated data to achieve accurate estimation. Estimated from the handful of samples, the transition scores may suffer from the statistical uncertainty due to the random fluctuation of scant data. To release this issue, inspired by variational inference (Kingma and Welling, 2014; Yoon et al., 2018; Gordon et al., 2019), we treat the transition score as the random variable and utilize the Gaussian distribution to approximate its distribution to model the uncertainty. Thus, our PA-CRF is to estimate the parameters of the Gaussian distribution rather than the transition scores directly, i.e., in the amortized manner (Kingma and Welling, 2014; Gordon et al., 2019). The Probabilistic Inference (Gordon et al., 2019) is employed based on the Gaussian distribution to make the inference robust by taking the possible perturbation of transition scores into account since the perturbation is also learned in a way that coherently explains the uncertainty of the samples.

To summarize, our contributions are as follows:

- We devise a tagging-based unified model for FSED. To the best of our knowledge, we are the first to solve this task in a unified manner, free from the cascading errors.
- We propose a novel model, PA-CRF, which estimates the distributions of transition scores for modeling the specific label dependency in the few-shot sequence labeling setting.
- Experimental results show that our proposed PA-CRF outperforms other competitive baselines on the FewEvent dataset. Further analyses show the effectiveness of our unified model and the limitation of the identify-then-classify models.

2 Related Work

Few-shot Event Detection Event Detection (ED) aims to recognize the specific type of events in a sentence. In recent years, various neural-based models have been proposed and achieved promising performance in ED (Chen et al., 2015; Nguyen and Grishman, 2015, 2018; Liu et al., 2018; Yan et al., 2019; Cui et al., 2020b). Chen et al. (2015) and Nguyen and Grishman (2015) proposed the convolution architecture to capture the semantic information in the sentence. Nguyen et al. (2016) introduced the recurrent neural network to model the sequence contextual information of words. Recently, GCN-based models (Nguyen and Grishman, 2018; Liu et al., 2018; Yan et al., 2019; Cui et al., 2020b) have been proposed to exploit the syntactic dependency information and achieved state-of-the-art performance. However, all these models are data-hungry, limiting dramatically their usability and deployability in real-world scenarios.

Recently, there has been an increasing research interest in solving event detection in the few-shot scenarios (Deng et al., 2020; Lai et al., 2020a,b), by exploiting the Few-Shot Learning (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Sung et al., 2018; Cong et al., 2020). Lai et al. (2020a) proposed LoLoss which splits the part of the support set to act as the auxiliary query set to train the model. Lai et al. (2020b) introduced two regularization matching losses to improve the performance of models. These works only focus on the few-shot trigger classification which classifies the event type of the annotated trigger according to the context based on few samples. This is unrealistic as triggers of novel events are predicted by some existing toolkits in advance. Deng et al. (2020) first proposed the benchmark dataset, *FewEvent*, for FSED and designed the DMBPN based on the dynamic memory networks. They train a conventional trigger identifier and a few-shot trigger classifier jointly and evaluated the model performance in the identify-then-classify paradigm. Moreover, our preliminary experiments reveal that the conventional trigger identification model tends to struggle when recognizing triggers of novel event types because of the trigger discrepancy between different event types. Thus, errors of the trigger identifier might be propagated to the event classification. Different from the previous identify-then-classify framework, for the first time, we solve Few-Shot Event Detection with two subtasks in a unified manner.

Few-shot Sequence Labeling In recent years, several works (Fritzler et al., 2019; Hou et al., 2020; Yang and Katiyar, 2020) have been proposed to solve the few-shot named entity recognition using sequence labeling methods. Fritzler et al. (2019) applied the vanilla CRF in the few-shot scenario directly. Hou et al. (2020) proposed a collapsed dependency transfer mechanism (CDT) into CRF, which learns label dependency patterns of a set of task-agnostic abstract labels and utilizes these patterns as transition scores for novel labels. Yang and Katiyar (2020) trains their model on the training data in a standard supervised learning manner and then uses the prototypical networks and the CDT for prediction in the inference phase. Different from these methods learning the transition scores by optimization, we build a network to generate the transition scores based on the label prototypes instead. In this way, we can generate exact label-specific transition scores of arbitrary novel event types to achieve adaptation ability. And we further introduce the Gaussian distribution to estimate the data uncertainty. Experiments prove the effectiveness of our method over the previous methods.

3 Problem Formulation

We convert event detection to a sequence labeling task. Each word is assigned a label that contributes to detecting the events. Labels consist of two parts: the word position in the trigger and the event type. We use the "BI" (Begin, Inside) signs to represent the position information of a word in the event trigger. The event type information is obtained from a predefined set of events. Label "O" (Other) means that the corresponding word is independent of the target events. Thus, the total number of labels is $2N + 1$ (N for *B-EventType*, N for *I-EventType*, and an additional *O* label), where N is the number of predefined event types.

Furthermore, we formulate the Few-Shot Event Detection in the typical N-way-K-shot paradigm. Let $\boldsymbol{x} = \{w_1, w_2, \ldots, w_n\}$ denote an n-word sequence, and $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ denote the label sequence of the $\boldsymbol{x}$. Given a *support set* $\mathcal{S} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N \times K}$ which contains N event types and each event type has only K instances, FSED aims to predict the labels of a unlabeled *query set* $\mathcal{Q}$ based on the *support set* $\mathcal{S}$. Formally, a $\{\mathcal{S}, \mathcal{Q}\}$ pair is called a N-way-K-shot task $\mathcal{T}$. There exist two datasets consisting of a set of tasks : $\mathcal{D}_{train} = \{\mathcal{T}^{(i)}\}_{i=1}^{M_{train}}$ and $\mathcal{D}_{test} = \{\mathcal{T}^{(i)}\}_{i=1}^{M_{test}}$ where M_{train} and M_{test} denote the number of the task in two datasets respectively. As the name suggests, $\mathcal{D}_{train}$ is used to train models in the training phase while $\mathcal{D}_{test}$ is for evaluation. It is noted that these two datasets have their own event types, which means that the label space of two datasets is disjoint with each other.

4 Methodology

4.1 Overview

As described above, we formulate FSED as the few-shot sequence labeling task with interdependent labels. Following the widely used CRF framework, we propose a novel PA-CRF model to model such label dependency in the few-shot setting, and decode the best-predicted label sequence. Our PA-CRF contains three modules: 1) Emission Module: It first computes the prototype of each label based on the support set, and then calculates the similarity between prototypes and each token in the query set as the emission scores. 2) Transition Module: It exploits the prototypes to generate the parameters of Gaussian distribution of the transition scores for decoding. 3) Decoding Module: Based on the emission scores and Gaussian distributed transition scores, the Decoding Module calculates the probabilities of possible label sequences for the given query set and decodes the predicted label sequence.

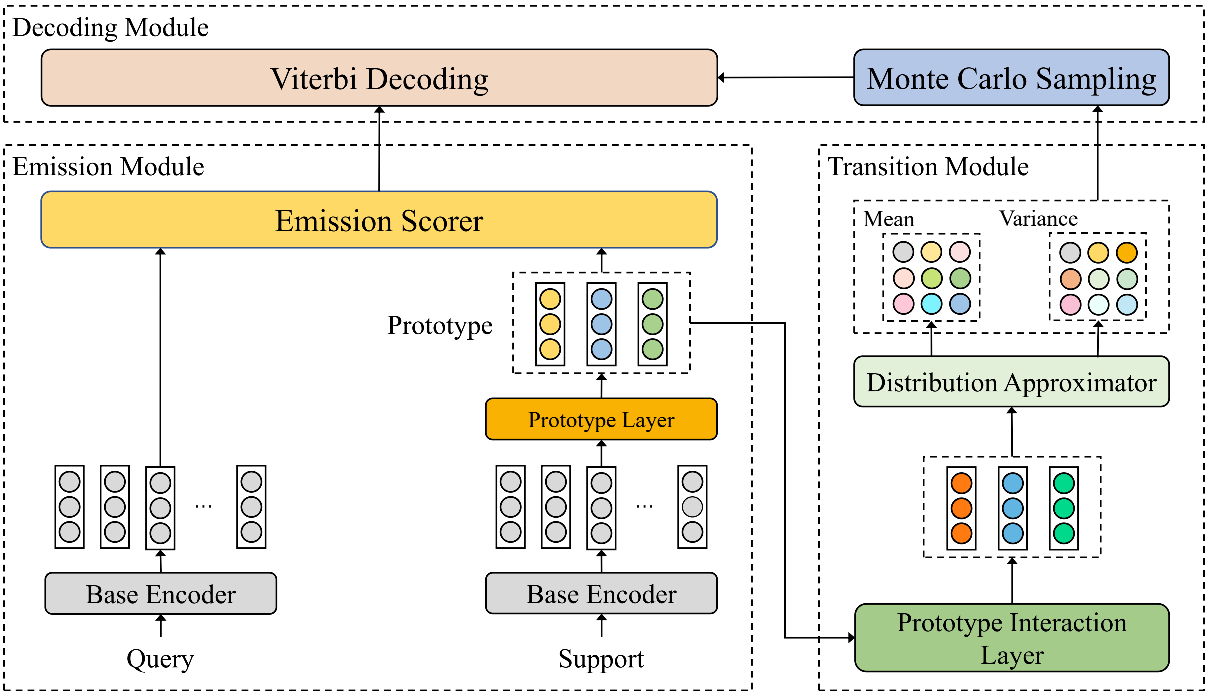

Figure 2: Architecture of our proposed PA-CRF. It consists of three modules: a) Emission Module calculates the emission scores for the query instance based on the prototypes derived from the support set. b) Transition Module generates the Gaussian distributed transition scores with respect to prototypes. c) Decoding Module exploits the emission scores and approximated Gaussian distributed transition scores to decode the predicted label sequence with the Monte Carlo Sampling.

Figure 2 gives an illustration of PA-CRF. We detail each component from the bottom to the top.

4.2 Emission Module

The Emission Module assigns the emission scores to each token of sentences in the query set $\mathcal{Q}$ with regard to each label based on the support set $\mathcal{S}$.

4.2.1 Base Encoder

Base Encoder aims to embed tokens in both support set $\mathcal{S}$ and query set $\mathcal{Q}$ into real-value embedding vectors to capture the semantic information.

Since BERT (Devlin et al., 2019) shows its advanced ability to capture the sequence information and has been widely used in NLP tasks recently, we use it as the backbone. Given an input word sequence $\boldsymbol{x}$, BERT first maps all tokens into hidden embedding representations. We denote this operation as:

$$\{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n\} = \mathrm{BERT}(\boldsymbol{x}) \tag{1}$$

where $\mathbf{h}_i \in \mathbb{R}^{d_h}$ refers to the hidden representation of token w_i, d_h is the dimension of the hidden representation.

4.2.2 Prototype Layer

Prototype Layer is to derive the prototypes of each label from the support set $\mathcal{S}$. As described in the problem formulation, we use the BIO schema to annotate the event trigger and N event types could contain $2N+1$ labels. Thus, indeed, we could get $2N+1$ prototypes. Following the previous work (Snell et al., 2017), we calculate the prototype of each label by averaging all the word representations with that label in the support set $\mathcal{S}$:

$$\mathbf{c}_i = \frac{1}{|\mathcal{S}(y_i)|} \sum_{w \in \mathcal{S}(y_i)} \mathbf{h}, \quad i = 1, 2, \ldots, 2N+1, \tag{2}$$

where $\mathbf{c}_i$ denotes the prototype for label y_i, $\mathcal{S}(y_i)$ refers to the token set containing all words in the support set $\mathcal{S}$ with label y_i, $\mathbf{h}$ represents the corresponding hidden representation of token w, and $|\cdot|$ is the number of set elements.

4.2.3 Emission Scorer

Emission Scorer aims to calculate the emission score for each token in the query set $\mathcal{Q}$. The emission scores are calculated according to the similarities between tokens and prototypes. The computation of the emission score of the label y_i for the word w_j is defined as:

$$f_E(y_i, w_j, \mathcal{S}) = d(\mathbf{c}_i, \mathbf{h}_j), \tag{3}$$

where $d(\cdot,\cdot)$ is the similarity function. In practice, we choose the dot product operation to measure the similarity.

Finally, given a word sequence $\boldsymbol{x}$, the emission score of the whole sentence with its corresponding ground-truth label sequence $\boldsymbol{y}$ is computed as:

$$\mathrm{EMIT}(\boldsymbol{y}, \boldsymbol{x}, \mathcal{S}) = \sum_{i=1}^{n} f_E(y_i, w_i, \mathcal{S}). \tag{4}$$

4.3 Transition Module

In vanilla CRF, the transition scores are learnable parameters and optimized from large-scale data to model the label dependency. However, in the few-shot scenarios, the learned transition scores cannot adapt to the novel label set due to the disjoint label space. To overcome this problem, we use neural networks to generate the transition scores based on the label prototypes instead of learning transition scores by optimization to achieve adaptation ability. In this case, a problem needing to be solved is that using few support instances with random data fluctuation to generate transition scores would cause uncertain estimation and result in wrong inference. To model the uncertainty, we treat the transition score as a random variable and use the Gaussian distribution to approximate its distribution. Specifically, the Transition Module is to generate the distributional parameters (mean and variance) of transition scores based on the label prototypes. It consists of two layers: 1) Prototypical Interaction Layer and 2) Distribution Approximator. Details of each layer are listed in the following parts.

4.3.1 Prototype Interaction Layer

Since the transition score is to model the dependency between labels, the individual prototype for each event type with rare dependency information is hard to generate their transition scores. Thus, we propose a Prototype Interaction Layer which exploits the self-attention mechanism to capture the dependency between labels.

We first calculate the attention scores of each prototype $\mathbf{c}_i$ with others:

$$\alpha_{ij} = \frac{\exp(\mathbf{c}_i^{(q)} \cdot \mathbf{c}_j^{(k)})}{\sum_{m=1}^{2N+1} \exp(\mathbf{c}_i^{(q)} \cdot \mathbf{c}_m^{(k)})}, \tag{5}$$

where $\mathbf{c}_i^{(q)}$ and $\mathbf{c}_i^{(k)}$ are transformed from $\mathbf{c}_i$ by two linear layers respectively:

$$\begin{aligned} \mathbf{c}_i^{(q)} &= W^{(q)}\mathbf{c}_i + b^{(q)} \\ \mathbf{c}_i^{(k)} &= W^{(k)}\mathbf{c}_i + b^{(k)} \end{aligned} \tag{6}$$

Getting the attention scores, the prototype $\tilde{\mathbf{c}}_i$ with dependency information is calculated as follows:

$$\tilde{\mathbf{c}}_i = \sum_{j=1}^{2N+1} \alpha_{ij}\mathbf{c}_j^{(v)}, \tag{7}$$

where $\mathbf{c}_i^{(v)}$ is also transformed linearly from $\mathbf{c}_i$:

$$\mathbf{c}_i^{(v)} = W^{(v)}\mathbf{c}_i + b^{(v)} \tag{8}$$

4.3.2 Distribution Approximator

This module aims to generate the mean and variance of Gaussian distributions based on the prototypes with dependency information.

Given the label set $\mathcal{Y}$ with total $2N+1$ labels, we first denote the transition score matrix as $T_r \in \mathbb{R}^{(2N+1)\times(2N+1)}$ for all label pairs, and denote the the i-th row j-th column element of T_r as $[T_r]_{ij}$ which refers to the transition score for i-th label transiting to j-th label in the label set $\mathcal{Y}$. As treating $[T_r]_{ij}$ as random variable, we use the Gaussian distribution $[\tilde{T_r}]_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$ to approximate $[T_r]_{ij}$, where $\mathcal{N}(\cdot,\cdot)$ refers to the Gaussian distribution. To estimate the mean μ_{ij} and variance σ_{ij} of $[\tilde{T_r}]_{ij}$, we concatenate the corresponding prototypes $\tilde{\mathbf{c}}_i$ and $\tilde{\mathbf{c}}_j$ and feed into two feed-forward neural networks respectively:

$$\mu_{ij} = W^{(\mu)}\left[\tilde{\mathbf{c}}_i \| \tilde{\mathbf{c}}_j\right] + b^{(\mu)} \tag{9}$$

$$\sigma_{ij}^2 = \exp\left(W^{(\sigma^2)}\left[\tilde{\mathbf{c}}_i \| \tilde{\mathbf{c}}_j\right] + b^{(\sigma^2)}\right) \tag{10}$$

where $[\cdot\|\cdot]$ means the concatenation operation. Given a label sequence $\boldsymbol{y}$, the transition score of the whole label sequence is approximated by:

$$\mathrm{TRANS}(\boldsymbol{y}, \tilde{T_r}) = \sum_{i=1}^{n-1} [\tilde{T_r}]_{\mathbb{I}(y_i)\mathbb{I}(y_{i+1})} \tag{11}$$

where $\mathbb{I}(y_i)$ refers to the label index in $\mathcal{Y}$ of y_i.

4.4 Decoding Module

Decoding Module derives the probabilities for a specific label sequence of the query set according to the emission scores and approximated Gaussian distributions of transition scores.

Since the approximated transition score is Gaussian distributional and not a single value, we denote the probability density function of the approximated transition score matrix as $q(\tilde{T_r}|\mathcal{S})$. According to the Probabilistic Inference (Gordon et al., 2019), the probability of label sequence $\boldsymbol{y}$ of a word sequence $\boldsymbol{x}$ based on the support set $\mathcal{S}$ is calculated as:

$$P(\boldsymbol{y}|\boldsymbol{x},\mathcal{S}) = \int P(\boldsymbol{y}|\boldsymbol{x},\mathcal{S},\tilde{T}_r)q(\tilde{T}_r|\mathcal{S})\mathrm{d}\tilde{T}_r \tag{12}$$

Following the CRF algorithm, the probability can be calculated based on the Equation 4 and Equation 11:

$$P(\boldsymbol{y}|\boldsymbol{x},\mathcal{S}) = \int \frac{1}{Z}\exp\left(\mathrm{EMIT}(\boldsymbol{y},\boldsymbol{x},\mathcal{S}) + \mathrm{TRANS}(\boldsymbol{y},\tilde{T}_r)\right) q(\tilde{T}_r|\mathcal{S})\mathrm{d}\tilde{T}_r \tag{13}$$

where

$$Z = \sum_{\boldsymbol{y}'\in Y} \exp\left(\mathrm{EMIT}(\boldsymbol{y}',\boldsymbol{x},\mathcal{S}) + \mathrm{TRANS}(\boldsymbol{y}',\tilde{T}_r)\right) \tag{14}$$

and Y refers to all possible label sequences.

In the training phase, we use negative log-likelihood loss as our objective function:

$$\mathcal{L} = -\mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{Q}}[\log P(\boldsymbol{y}|\boldsymbol{x},\mathcal{S})] \tag{15}$$

Due to the hardness to compute the integral of Equation 13, in practice, we use the Monte Carlo sampling technique (Gordon et al., 2019) to approximate the integral. To make the sampling process differentiable for optimization, we employ the reparameterization trick (Kingma and Welling, 2014) for each transition score $[\tilde{T}_r]_{ij}$:

$$[\tilde{T}_r]_{ij} = \mu_{ij} + \epsilon\sigma_{ij}, \text{where } \epsilon \sim \mathcal{N}(0,1) \tag{16}$$

In the inference phase, the Viterbi algorithm (Forney, 1973) is employed to decode the best-predicted label sequence for the query set.

5 Experiment

5.1 Dataset

We conduct experiments on the benchmark *FewEvent* dataset introduced in the previous work (Deng et al., 2020), which is the currently largest few-shot dataset for event detection. It contains 70,852 instances for 100 event types and each event type owns about 700 instances on average. Since Deng et al. (2020) do not share their split train/dev/test set, we re-split the FewEvent in the same ratio as Deng et al. (2020). We use 80 event types as the training set, 10 event types as the dev set, and the rest 10 event types as the test set. More statistics of FewEvent dataset are listed in Appendix A.

5.2 Evaluation

We follow the evaluation metrics in previous event detection works (Chen et al., 2015; Liu et al., 2018; Cui et al., 2020b), an event trigger is marked correct if and only if its event type and its offsets in the sentence are both correct. We adopt the standard micro F1 score to evaluate the results and report the averages and standard deviations over 5 randomly initialized runs.

6 Implementation Details

We employ *BERT-BASE-UNCASED* (Devlin et al., 2019) as the base encoder. The maximum sentence length is set as 128. Our model is trained using *AdamW* optimizer with the learning rate of 1e-5. All the hyper-parameters are tuned on the dev set manually. In the training phase, we follow the widely used episodic training (Vinyals et al., 2016) in few-shot learning. Episodic training aims to mimic N-way-K-shot scenario in the training phase. In each epoch, we randomly sample N event types from the training set and each event type randomly sample K instances as support set and other M instances as the query set. We train our model with 20,000 iterations on the training set and evaluate its performance with 3,000 iterations on the test set following the episodic paradigm. We run all experiments using PyTorch 1.5.1 on the Nvidia Tesla T4 GPU, Intel(R) Xeon(R) Silver 4110 CPU with 256GB memory on Red Hat 4.8.3 OS.

6.1 Baselines

To investigate the effectiveness of our proposed method, we compare it with a range of baselines and state-of-the-art models, which can be categorized into three classes: fine-tuning paradigm, identify-then-classify paradigm and unified paradigm.

Fine-tuning paradigm solves the FSED in the standard supervised learning, i.e., pre-training on the large scale dataset and fine-tuning on the handful target data. We adopt the state-of-the-art model, **PLMEE** (Yang et al., 2019), of the standard ED into the FSED directly.

Identify-then-classify models first perform trigger identification (named as TI) and then classify the event types based on the few-shot learning methods (named as FSTC). We investigate two typed of identify-then-classify paradigms: separate and multi-task. For the separate models, the trigger identifier and few-shot trigger classifier are

Paradigm	Model	5-Way-5-Shot	5-Way-10-Shot	10-Way-5-Shot	10-Way-10-Shot
Fine-tuning	PLMEE	4.43 ± 0.19	4.69 ± 0.85	2.52 ± 0.28	2.76 ± 0.55
Separate	LoLoss	30.14 ± 0.30	30.91 ± 0.29	29.33 ± 0.40	30.08 ± 0.39
	MatchLoss	29.78 ± 0.14	30.75 ± 0.15	28.75 ± 0.23	29.59 ± 0.21
Multi-task	LoLoss	31.51 ± 1.56	31.70 ± 1.21	30.46 ± 1.38	30.32 ± 0.89
	MatchLoss	30.44 ± 0.99	30.68 ± 0.78	28.97 ± 0.61	30.05 ± 0.93
	DMBPN	37.51 ± 2.60	38.14 ± 2.32	34.21 ± 1.45	35.31 ± 1.69
Unified	Match	39.93 ± 1.67	46.02 ± 1.20	30.88 ± 1.08	35.91 ± 1.19
	Proto	50.11 ± 0.77	52.97 ± 0.95	43.51 ± 1.16	42.70 ± 0.98
	Proto-Dot	58.82 ± 0.88	61.01 ± 0.23	55.04 ± 1.62	58.78 ± 0.88
	Relation	28.91 ± 1.13	29.83 ± 0.78	18.49 ± 1.25	21.47 ± 1.40
	Vanilla CRF	59.01 ± 0.81	62.21 ± 1.94	56.00 ± 1.51	59.35 ± 1.09
	CDT	59.30 ± 0.23	62.77 ± 0.12	56.41 ± 1.09	59.44 ± 1.83
	StructShot	57.69 ± 0.91	61.54 ± 1.23	54.54 ± 0.95	57.14 ± 0.79
	PA-CRF	**62.25*** ± 1.42	**64.45*** ± 0.49	**58.48*** ± 0.68	**61.64*** ± 0.81

Table 1: F1 scores (10^{-2}) of different models on the FewEvent test set. Bold marks the highest number among all models, underline marks the second-highest number, and ± marks the standard deviation. * marks statistically significant improvements over the best baseline with $p < 0.01$ under a boostrap test.

trained separately without parameter sharing. We first exploit the state-of-the-art BERT-tagger for the TI task. It uses BERT (Devlin et al., 2019) and a linear layer to tag the trigger in the sentence as a sequence labeling task. Since TI just aims to recognize the occurrence of the trigger, the label set only contains three labels: *O*, *B-Trigger*, *I-Trigger*. For the FSTC task, we reimplement two FSTC models: **LoLoss** (Lai et al., 2020a), **MatchLoss** (Lai et al., 2020b). In the multi-task models, we reimplement **DMBPN** (Deng et al., 2020) and replace its encoder with BERT for the fair comparison. DMBPN combines a conventional trigger identification module and a few-shot trigger classification module by parameter sharing. But in the inference phase, it detects the event trigger still in the identify-then-classify paradigm. Additionally, we also provide the multi-task version of the LoLoss and MatchLoss which are trained jointly with BERT-tagger with shared BERT parameters.

Unified models perform few-shot event detection with a single model without task decomposition. Because we are the first to solve this task in a unified way, there is no previous unified model that can be compared. But for the comprehensive evaluation of our proposed PA-CRF model, we also construct two groups of variants of PA-CRF: non-CRF models and CRF-based models. Non-CRF models use emission scores to predict via softmax and do not take the label dependency into consideration. We implement four typical few-shot classifiers: 1) **Match** (Vinyals et al., 2016) uses cosine function to measure the similarity, 2) **Proto** (Snell et al., 2017) uses Euclidean Distance as the similarity metric, 3) **Proto-Dot** uses dot product to compute the similarity, 4) **Relation** (Sung et al., 2018) builds a two-layer neural networks to measure the similarity. Since CRF with the capacity of modeling label dependency is widely used in sequence labeling task, we implement three kinds of CRF-based models as our baselines: 1) **Vanilla CRF** (Fritzler et al., 2019): We adopt the vanilla CRF in the FSED task without considering the adaptation problem. 2) **CDT** (Hou et al., 2020): As the SOTA of the few-shot NER task, we reimplement it according to the official code and adapt it in the FSED task to replace our Transition Module. 3) **StructShot** (Yang and Katiyar, 2020): It is also a few-shot NER model. It first pre-trains on the training set and utilizes the prototypical networks and the CDT for prediction based on the support set in the inference phase. For the fair comparison, the emission module of these CRF-based baseline models is the same as our PA-CRF.

6.2 Main Results

Table 1 summarizes the results of our PA-CRF against other baseline models on the FewEvent test set.

Comparison with fine-tuning model It is obvious that PLMEE performs poorly in all four few-shot settings and all few-shot-based methods outperform it with an absolute gap, which powerfully proves that the conventional supervised methods is incapable of solving FSED.

Comparison with identify-then-classify models (1) Most of unified models (except Relation) perform higher than all identify-then-classify models, especially for PA-CRF with huge gaps about 30%, proving the effectiveness of the unified framework. (2) Comparing with the separate paradigm, the multi-task paradigm is able to improve performance but it still cannot catch up with the unified paradigm. (3) DMBPN works better than other two models but still works poorly to handle the FSED due to the limitation of the TI. We will discuss the bottleneck of the identify-then-classify paradigm in Section 6.3.

Comparison with unified models (1) Over the best non-CRF baseline model Proto-Dot, PA-CRF achieves substantial improvements of 3.43%, 3.44%, 3.44% and 2.86% on four few-shot scenarios respectively, which confirms the effectiveness and rationality of PA-CRF to model the label dependency. (2) Vanilla CRF performs better than other non-CRF baseline methods, which demonstrates that CRF is able to improve the performance by modeling the label dependency, even if the learned transition scores do not match the label space of the test set. (3) Compared to Vanilla CRF, both CDT and StructShot achieve slightly higher F1 scores, indicating the transition scores of abstract BIO labels can improve the model adaptation ability to some extent. (4) CDT exceeds the StructShot since CDT is trained based on the episodic training, which makes it learns the class-agnostic token representations. (5) PA-CRF outperforms CDT (2.95%, 1.68%, 2.07% and 2.20% in four few-shot settings respectively) with absolute gaps. We consider that it is because CDT learning the transition scores of the abstract labels cannot model the exact dependency of specific label set, so its adaptation ability is limited. In contrast, PA-CRF generates the label-specific transition scores based on the label prototype, which can capture the dependency for specific novel event types. (6) Comparing four few-shot scenarios, we can find that the F1 score increases as the K-shot increases, which shows that more support samples can provide more information of the event type. The F1 score decreases as the N-way increases when the shot number is fixed, which reveals that the larger way number causes more event types to predict which increases the difficulty of the correct detection.

Model	TI	FSTC	FSED
LoLoss	31.06	95.27	30.14
DMBPN	40.64	95.44	37.51
DMBPN(CDT-TI)	54.69	95.49	53.93
PA-CRF	63.68	96.76	62.25

Table 2: Comparison of PA-CRF and baselines on two subtasks. F1 scores are reported on the FewEvent test set in the 5-way-5-shot setting.

To summarize, we can draw the conclusion that (1) The identify-then-classify paradigm is incapable of solving the FSED task. (2) Compared to the identify-then-classify paradigm, the unified paradigm works more effectively for the FSED task. (3) Approximating transition scores based on the label prototypes not by optimization, our PA-CRF achieves better adaptation on novel event types.

6.3 Bottleneck Analysis

To investigate the bottleneck of the identify-then-classify paradigm, we evaluate LoLoss (separate model), DMBPN (multi-task model) and PA-CRF (unified model) on two subtasks: TI and FSTC separately in the 5-way-5-shot setting on the FewEvent test set. To reduce the influence of the cascading errors, we use the ground truth trigger span for evaluation in the FSTC. The experimental results are reported in Table 2. From Table 2, we find that: (1) All models achieve more than 95% F1 score on the FSTC task, indicating that both identify-then-classify and unified models is capable enough of solving the FSTC problem. (2) For the TI task, two identify-then-classify baselines perform 31.06% and 40.64% F1 score respectively, which demonstrates that the conventional TI module has difficulty in adapting to novel event triggers. Hence, due to the cascading errors, the poorly-performed TI module limits the performance of the identify-then-classify models. (3) PA-CRF achieves 63.68% F1 score on TI task, which exceeds the two kinds of identify-then-classify models significantly. Unlike identify-then-classify models recognizing triggers based on seen triggers, PA-CRF utilizes the trigger representations from the support set of the novel event types to identify novel triggers so our unified

Model	5-Shot	10-Shot
PA-CRF	44.39	51.06
- Distribution Estimation	43.47	49.41
- Interaction Layer	41.62	45.74
- Transition Score	39.83	45.07

Table 3: Ablation study of PA-CRF in 5-Way settings. F1 scores are reported on the FewEvent dev set.

model works better in the TI task of FSED. In conclusion, the conventional trigger identifier cannot identify novel triggers in FSED, and exploiting the support set of novel event types is necessary.

6.4 Effectiveness Analysis

To verify the effectiveness of the unified framework, we adapt our best baseline model, CDT, to replace TI module of DMBPN to solve trigger identification in the few-shot manner. It identifies triggers based on the emission scores between tokens and label prototypes calculating from the support set and learned abstract transition scores. In this case, we rename it as DMBPN(CDT-TI) and evaluate it in TI and FSTC subtasks. Results are also reported based on the 5-way-5-shot setting in Table 2 and we observe that: The CDT-TI-based DMBPN achieve 54.69% in TI task, exceeding the conventional TI based models, which shows that solving TI in the few-shot manner by utilizing the support set can reduce the trigger discrepancy to some extent. Although the performance of FSTC is similar to the original DMBPN, owing to the improvements of TI task, the final performance of FSED exceeds the original DMBPN by 16.42% but they are still inferior to PA-CRF with a huge gap (8.99% on TI task). Therefore, we draw the conclusion that solving FSED in the unified manner can utilize the correlation between two subtasks to improve the model performance significantly.

6.5 Ablation Study

To study the contribution of each component in our PA-CRF model, we run the ablation study on the FewEvent dev set. From these ablations (see Table 3), we find that: (1) - Distribution Estimation: To study whether distributional estimation is helpful to improve the performance, we remove it and make the Distribution Approximator generate a single value as the transition score directly as the point estimation. And the inference is based on the generated transition scores without Probabilistic Inference. As a result, the F1 score drops 1.02% and 1.65% in two scenarios, respectively. We attribute these gaps to our proposed Gaussian-based distributional estimation which can model the data fluctuation to relieve the influence of data uncertainty. (2) - Interaction Layer: To certify that the Prototype Interaction Layer contributes to capturing the information between prototypes, we remove it and evaluate in two scenarios. We read from Table 3 that F1 scores decrease significantly by 2.77% and 5.32% respectively, which indicates that the Prototype Interaction Layer is able to capture the dependency among prototypes. (3) - Transition Score: To prove the contribution of the label dependency, we remove the Transition Module and only use the emission score for prediction. Results show that without transition scores, the performance of the model drops dramatically by 4.56% and 5.99% respectively, which powerfully proves that the transition score can improve the performance of the few-shot sequence labeling task.

Furthermore, we have conducted case study and error analysis to validate the strength of our PA-CRF and explore its weakness. Details are listed in Appendix B and Appendix C.

7 Conclusion

In this paper, we explore a new viewpoint of solving few-shot event detection in a unified manner. Specifically, we propose a prototypical amortized conditional random field to generate the transition scores to achieve adaptation ability for novel event types based on the label prototypes. Furthermore, we present the Gaussian-based distributional estimation to approximate transition scores to relieve the statistical uncertainty of data fluctuation. Finally, experimental results on the benchmark FewEvent dataset prove the effectiveness of our proposed method. In the future, we plan to adapt our method to other few-shot sequence labeling tasks such as named entity recognition.

Acknowledgements

We would like to thank all reviewers for their insightful comments and suggestions. This work is supported in part by the Strategic Priority Research Program of Chinese Academy of Sciences (grant No. XDC02040400) and the Youth Innovation Promotion Association of CAS (Grant No. 2021153).

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Xin Cong, Bowen Yu, Tingwen Liu, Shiyao Cui, Hengzhu Tang, and Bin Wang. 2020. Inductive unsupervised domain adaptation for few-shot classification via clustering.

Shiyao Cui, Bowen Yu, Xin Cong, Tingwen Liu, Quangang Li, and Jinqiao Shi. 2020a. Label enhanced event detection with heterogeneous graph attention networks. *CoRR*, abs/2012.01878.

Shiyao Cui, Bowen Yu, Tingwen Liu, Zhenyu Zhang, Xuebin Wang, and Jinqiao Shi. 2020b. Edge-enhanced graph convolution networks for event detection with syntactic relation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP*.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*.

G. D. Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. 2019. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393, Online. Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR*.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020a. Exploiting the matching information in the support set for few shot event classification. In *Advances in Knowledge Discovery and Data Mining - 24th Pacific-Asia Conference, PAKDD*.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020b. Extensively matching for few-shot learning event detection. *CoRR*, abs/2006.10093.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*.

Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. 2019. Event detection with multi-order graph convolution and aggregated attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5766–5770, Hong Kong, China. Association for Computational Linguistics.

Fan Yang, Xiaochang Peng, Gargi Ghosh, Reshef Shilon, Hao Ma, Eider Moore, and Goran Predovic. 2019. Exploring deep multimodal fusion of text and photo for hate speech classification. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 11–18, Florence, Italy. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. 2018. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*.

A Dataset Statistics

	Training Set	Dev Set	Test Set
# Cls.	80	10	10
# Sent.	67982	2173	697
# Tok./sent	36.5	38.6	30.8

Table 4: Statistics of FewEvent Dataset.

Table 4 lists the statistics of FewEvent dataset containing the number of event type (#Cls.), the number of sentence(# Sent.), the number of token per sentence (# Tok./sent) for the train/dev/test set.

Figure 3 demonstrates the data imbalance problem of FewEvent dataset. Event “Marry” has the most instance (26135 instances) while event “E-Mail” only has 30 instances. 69% event types have less than 100 instances while 7% event types have more than 1000 instances. However, since we use episodic training (Vinyals et al., 2016) to train our model, the data imbalance problem can be relieved to some extent.

B Case Study

We compare our method with the best identify-then-classify baseline, DMBPN and the best unified baseline, CDT in some cases, as shown in Table 5.

As demonstrated by the first example of a *Sponsorship* event, DMBPN, in the identify-then-classify paradigm, fails to identify the trigger *sponsorship*. According to our statistics about the FewEvent dataset, 95.16% triggers of *Sponsorship* event do not occur in the training set. Since DMBPN uses the conventional TI module which is trained on the training set to identify the event trigger, it is incapable of identifying the *Sponsorship* event trigger. Although the classification module of DMBPN succeeds to distinguish the event type as *Sponsorship*, due to the cascading errors, the final prediction of event trigger (containing the span and type) is incorrect. As a result, the performance of DMBPN as an identify-then-classify model on the FSED task is limited. In contrast, our unified PA-CRF is successful to detect the event trigger *sponsorship* of this case since PA-CRF utilizes the information of the support set of *Sponsorship* event in which word *sponsorship* appears and acts the trigger.

In the second example, the best unified baseline, CDT, tags the first trigger word *locked* with *I-Jail* label wrongly. That is because CDT learns the abstract transition scores among a set of abstract labels which cannot model the label dependency for this specific event type accurately. Thanks to the PA-CRF which models the label dependency based on the label prototypes from the support set of *Jail* event, our model is capable of tagging the word *locked* with *B-Jail* label correctly.

C Error Study

Although our method outperforms all baseline models, we still observe some failure cases. Table 6 gives a typical example of the wrong prediction of event *Transport* (*Trans* for short). For the query instance, the ground truth event trigger is “pouring out”. The word “pouring” should be labeled as *B-Trans* and the *out* should be labeled as *I-Trans*. However, our model only detects “pouring” with *B-Trans* while missing “out”. From the support set, we find that all support instances of this event type only contain the one-word trigger without *I-Trans* label tokens, resulting in that the prototype of *I-Trans* is zero vector. As a result, the emission score for the label *I-Trans* of each query token is calculated as zero and the transition scores based on the prototypes are also affected. Therefore, our model is not able to detect the *I-Trans* label correctly in this case. In the future, we will further study to

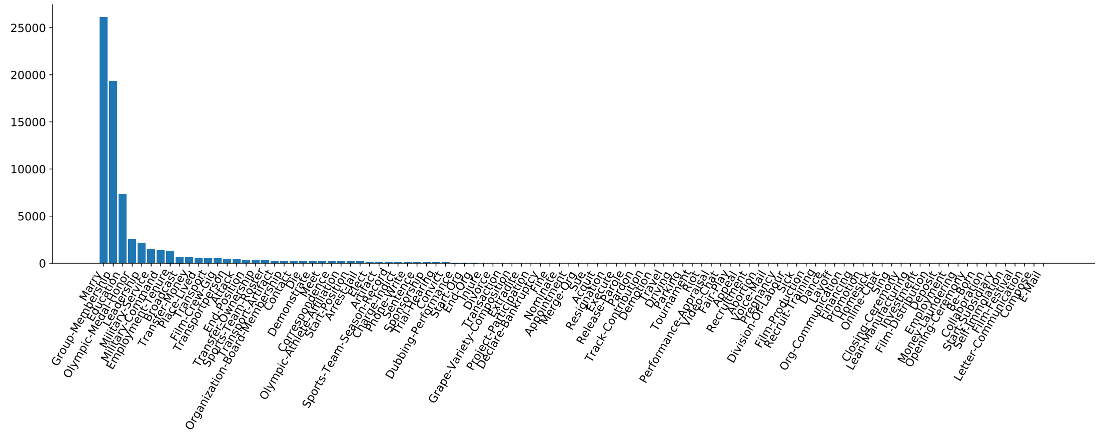

Figure 3: The data imbalance of FewEvent dataset.

Model	Prediction
DBMPN PA-CRF	Candlestick Park was dropped when the $[sponsorship]_O$ agreement expired. Candlestick Park was dropped when the $[sponsorship]_{B-Sponsorship}$ agreement expired.
CDT PA-CRF	Willmore will tell everyone for wanting to keep the poor man $[locked]_{I-Jail}$ $[up]_{I-Jail}$. Willmore will tell everyone for wanting to keep the poor man $[locked]_{B-Jail}$ $[up]_{I-Jail}$.

Table 5: Output of PA-CRF, DMBPN and CDT on samples from the FewEvent test set. The subscripts denote the labels tagged by the models.

Support #1 Support #2	Cult members $[visited]_{B-Trans}$ and built a laser weapon mounted on a truck Israel $[leave]_{B-Trans}$ the West Bank and Gaza and dismantle Jewish settlements.
Truth Prediction	Refugees have been $[pouring]_{B-Trans}$ $[out]_{I-Trans}$ of Fallujah over the last few days. Refugees have been $[pouring]_{B-Trans}$ $[out]_O$ of Fallujah over the last few days.

Table 6: A case of the wrong prediction from the FewEvent test set. The subscripts denote the triggers and their event types. We only list two support instances to reduce space.

solve the missing *I* label problem.

D Analyses about Various Dataset Split

Model	R1	R2	R3	R4	R5
PA-CRF	59.0	33.4	53.1	42.4	48.0
DMBPN	44.9	31.1	40.8	32.6	27.9

Table 7: Performance of our PA-CRF and DMBPN in various split FewEvent dataset in the 5-Way-5-Shot scenario. F1 scores (10^{-2}) are reported.

Since Deng et al. (2020) do not public their split train/dev/test set of FewEvent dataset, to compare our PA-CRF with DMBPN (Deng et al., 2020), we re-split the FewEvent randomly in the same split ratio as the Deng et al. (2020) (80 event types for training set, 10 event types for dev set and the rest 10 event types for test set) and evaluate the DBMPN performance on our split test set. However, in our experiments, the performance of DMBPN is lower than the original paper. We assume that the different data split may influence the model performance badly. To validate our assumption, we re-split the FewEvent dataset for five random seeds and conduct more experiments on these various split train/dev/test set. The results are reported in Table 7. From Table 7, it can be observed that: (1) Data split does influence the model performance significantly indeed. In these five different split train/dev/test set, the performance of PA-CRF varies from 59.0% to 33.4% with a huge range. Similarly, DMBPN also varies from 44.9% to 27.9%, owning a huge gap about 20%. It demonstrates that for the FewEvent dataset, different split could cause huge fluctuation of the model performance. Therefore, our PA-CRF including baselines performs lower than those Deng et al. (2020) reported due to the different data split. (2) PA-CRF outperforms DMBPN in all five random split settings, which powerfully proves the robustness of PA-CRF over the identify-then-classify paradigm.

LUX (Linguistic aspects Under eXamination): Discourse Analysis for Automatic Fake News Classification

Lucas Azevedo[1], Mathieu d'Aquin[1], Brian Davis[2], and Manel Zarrouk[3]

[1]Data Science Institute, Insight SFI Centre for Data Analytics
National University of Ireland, Galway
[2]ADAPT, DCU, School Of Computing
[3]Galilée Institute, University Sorbonne Paris North
{lucas.azevedo, mathieu.daquin}@nuigalway.ie
brian.davis@adaptcentre.ie
manel.zarrouk@sorbonne-paris-nord.fr

Abstract

The democratization/decentralization of both the production and consumption of information has resulted in a subjective and often misleading depiction of facts known as Fake News - a phenomenon that is effectively shaping the perception of reality for many individuals. Manual fact-checking is time-consuming and cannot scale and although automatic fact-checking, vis a vis machine learning holds promise, it is significantly hindered by a deficit of suitable training data. We present both a novel dataset, VERITAS(VERIfying Textual Aspects), a collection of fact-checked claims, containing their original documents and LUX(Language Under eXamination), a text classifier that makes use of an extensive linguistic analysis to infer the likelihood of the input being a piece of fake-news.

1 Introduction

Often defined as the intentional or unintentional spread of false information (K et al., 2019), Fake News has found fertile ground in the actual scenario of ever-growing data consumption and generation, where factors like news source decentralization, citizen journalism, democratization of media and astroturfing[1] (Lee, 2010) make the task of manually checking and correcting disinformation across the internet impractical if not infeasible, (Shao et al., 2016) despite the significant efforts of Fact-Checking Agencies - organised groups of journalists that manually identify and investigate rumours conveyed by Fake-news articles.

Consequently, it is imperative that we develop an efficient and reliable way to account for the veracity of what is produced and spread as information; this process is known as automatic fact-checking. (Hassan et al., 2015)

Although the there has been significant research effort to tackle the task of automatic fact-checking (Azevedo, 2018), the deficit of datasets containing organic news articles - in their entirety - which have been manually labeled with respect to their veracity is a common obstacle for the development of supervised classification models. The absence of such datasets makes researchers rely on other approaches, e.g., stance determination (Popat et al., 2017), knowledge base matching (Wu et al., 2014), trust assessment of sources (Balakrishnan and Kambhampati, 2011), data structuring (Conroy et al., 2015), network pattern analysis (Shao et al., 2016), etc.

In this work we present the challenges faced in the process of developing a language model enriched by discourse features for fake-news detection, along with experimental results. The contributions of this work are mainly two: the dataset creation process, described in Section 2 and the introduction of the text classification model, - LUX(Language Under eXamination), in Section 3. Section 4 brings a comprehensive evaluation of both VERITAS and LUX, while also featuring an ablation analysis of the latter.

2 Datasets for Fake News Classification

2.1 Available Corpora on Fake News

The deficit of suitable corpora for the intended approach is the main influence behind the creation of the VERITAS Dataset, and by consequence, the VERITAS Annotator. Below we present a list of datasets commonly used in related tasks. Note that although the following are considered valuable resources for many related tasks, none of them include all of the three most important characteristics required for a content based supervised classifier which are i) a significant volume of entries, ii) gold standard labels and iii) the entire fake news articles

---

[1] Astroturfing is the practice of masking the sponsors of a message or organization to make it appear as though it originates from and is supported by grassroots participants.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 41–56
August 1–6, 2021.

(i.e., the origin).

Emergent16 a collection of 300 rumours and 2,595 associated news articles - a counterpart to 'origin' in the VERITAS Dataset. Each claim's veracity is estimated by journalists after they have judged that enough evidence has been collected (Ferreira and Vlachos, 2016). Besides the claim labeling, each associated article is summarized into a headline and also labelled according to its stance towards the claim. Given the fixed structured of the website we were able to obtain valid labeled examples using a scraper.[2] Unfortunately they sum up to less than 100 usable claim-origin pairs (discussed in subsection 2.3).

LIAR17 includes around 13K human-labeled short statements which are rated by the fact-checking website PolitiFact into: "pants on fire", "false", "barely true", "half true", "mostly true", or "true" (Wang, 2017). The domain-restricted data as well as the reduced length of text that can be retrieved from this corpus makes it unsuitable for generic domain linguistic fake news detection.

FakeNewsNet18 is a data repository containing a collection of around 22K real and fake news obtained from Politifact and GossipCop[3] fact-checking websites. Each row contains an ID, URL, title, and a list of tweets that shared the URL. It also includes linguistic, visual, social, and spatiotemporal context regarding the articles. This repository could still be used for supervised learning models if it were not for the fact that it does not provide sufficiently long texts to be used by a classifier based on linguistic aspects. For the same reason, **CREDBANK** (Mitra and Gilbert, 2015) and **PHEME** (Derczynski and Bontcheva, 2014) are also unsuitable for the authors' use case. Those three datasets focus on network indicators (e.g. number of retweets, sharing patterns, etc.) of fake news, instead of its contents. CREDBANK is a crowd sourced corpus of "more than 60 million tweets grouped into 1,049 real-world events, each annotated by 30 human annotators", while PHEME includes 4,842 tweets, in the form of 330 threads, related to 9 events.

FEVER18 (Thorne et al., 2018) created FEVER, a set of more than 185K claims by modifying sentences from a collection of 50K Wikipedia articles. Annotators were tasked with annotating other sentences from the same article in respect to their stance towards the modified sentence. The corpus is the largest to our knowledge, but since it is synthetically created and focused on a sentence-level stance classification approach, it is unlikely to perform efficiently on heterogeneous web documents as a fake news classifier.

Snopes19 (Hanselowski et al., 2019) provides a large collection of more than 16 thousand manually annotated text snippets extracted from 6,422 `snopes.com` articles. Unfortunately, less than half of those snippets present a stance (agreeing or disagreeing) towards the fact-checked claim. Also, the annotated snippets are, by definition, only a portion of the original article. Nevertheless, an origin identification process could generate a significant amount of valid examples from this data.

Due to space restrictions, we cannot provide a detailed description of the following list of datasets, although is important to include them: **BuzzFeed16** (Potthast et al., 2018), **Kaggle**[4] and **NELA17** (Horne et al., 2018).

2.2 The VERITAS Dataset

The VERITAS Dataset is, to our knowledge, the most complete data collection of manually annotated claims in regards to their veracity. It is the only dataset to contain not only the mentioned veracity labels but also the document (in its entirety) from which the checked claim originated. VERITAS has been developed in a two step process: **1)** Fact-Checking articles scraping and **2)** Claim Origin Identification.

Step 1: Scraping FCAs As the cost for manually checking a large number of disputed claims is extensive, both in time and money, we have started the dataset creation process by scraping articles from fact-checking agencies and consequently trusting the work made by their journalists that undertake the processes of: 1)selecting controversial

---

[2] While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler.

[3] `https://www.gossipcop.com`

[4] `https://www.kaggle.com/mrisdal/fake-news/data`

claims, 2)leveraging web documents that either support or deny those statements to 3) finally come to a veracity verdict. In simple terms, a Fact-Checking Article (FCA) is a narrative of this investigative process.

For each scraped FCA, we create an entry in the dataset and extract a number of attributes, most importantly: the claim, the veracity label, and the list of hyperlinks to the mentioned web documents, which we call *Origin Candidates*, since they will be the subject of the Origin Identification process. The code used to scrap the pages is openly available[5].

Step 2: Claim Origin Identification One of the most important steps of the dataset creation pipeline was a task we defined as "origin identification". In short, after three automatic ways of identifying the article in which a fact-checked claim originated were carried out and yielded non-satisfactory results, it was decided that a manual annotation process would be used to select the correct entries from the totality of the dataset. An annotation tool[6] was developed in order to make the task easier and faster. This annotation process not only provided a large and complete version of the dataset, but also leaves a possibility for an automation of the origin identification process as a future improvement of the project.

The final structure of each entry contains the following fields: Fact-Checking Article URL, Checked Claim, Claim Label, Tags, FCA date, Origin URL, Origin Domain, Origin Body, Origin Title, Origin Summary, Origin Keywords, Origin Date and Origin Author. Given the limited space, a more in-depth description of each field is not provided but can be found within the supplementary material (appendix 1) and also along an extensive description of the origin annotation process in (Azevedo and Moustafa, 2019). The past versions of the dataset are also openly available[7]

2.3 Consolidation of VERITAS Dataset

A consolidation of the VERITAS dataset followed the large annotation process over the scraped FCA pages that augmented both the quantity of annotated origins (1032 consolidated origins from more than 10k annotations) and the quality of the annotations, measured by Krippendorff's Alpha[8], reaching a substantial score of 0.6014. This consolidation generated the fourth version of the dataset, here addressed as V4.

Given the constant structure of `Emergent.info` articles, we have also incorporated its few valid claims, i.e., the ones with "true" or "false" verdict, and their respective sources.

Although the majority of origins obtained from Emergent were linked to "true" claims, when aggregated to the consolidated origins from VERITAS v4.0, the data collection showed a false/true class imbalance ratio of ≈ 1.44. Therefore, in order to quickly obtain "true" labeled news articles to balance the scraped Dataset, reporting articles were scraped and automatically labeled as "true" and composed a **separated** dataset where their headlines are used for the claim field. The sources of those articles were selected according to studies determining the least biased[9] and/or most trusted[10] news outlets in the U.S..

We are aware that the label assumption of those articles is far from ideal. Notwithstanding, it offers another **option** of palliative solution for the label unbalance issue and yielded positive results in similar works (Horne and Adali, 2017; Ireland, 2018). It should, however, be tested with caution and compared with other - also sub-optimal - methods, i.e., discarding "false" entries and/or implementing class weights on the model training. Both the collection of reporting articles and the emergent articles are provided **separately** so they can be optionally disregarded and eventually substituted by gold-standard data. Table 1 provides additional details about each subset.

Since the improvement of incorporating the entries from emergent was still to be evaluated by the proposed classifier, two different sample sets from the trusted sources were created, to balance both the v4.0 dataset by itself (V4+T1), as well as the concatenation of VERITAS and emergent (V4+EM+T2). The evaluation results will be presented at Section 4, as they are also the evaluation for the linguistic model. By comparing both balanced sets we can gain a better understanding of the

---

[5] `https://github.com/lucas0/VeritasCrawler`

[6] `https://github.com/lucas0/Annotator`

[7] `https://github.com/lucas0/VeritasCorpus`

[8] `https://en.wikipedia.org/wiki/Krippendorff%27s_alpha`

[9] `https://www.businessinsider.com/most-biased-news-outlets-in-america-cnn-fox-nytimes-2018-8`

[10] `businessinsider.com/most-and-least-trusted-news-outlets-in-america-cnn-fox-news-new-york-times-2019-4`

Table 1: VERITAS Subsets

	#E	#T	#F	#U
VERITAS v4.0 (V4)	1032	276	664	92
Emergent (EM)	865	308	179	378
Trusted1 (T1)	388	388	-	-
Trusted2 (T2)	259	259	-	-
V4+T1	1420	664	664	92
V4+EM+T2	2156	843	843	470

*Columns represent **#E**: total entries, **#T**:true entries, **#F**: false entries, **#U**:unverified entries*

quality of the data obtained from emergent, keeping in mind that the difference in volume of entries would still affect the performance.

3 LUX - Language Under eXamination

The core contribution of this work is the investigation of the usage of linguistic aspects as discriminative features in a text classification model that should determine whether the given article is fake or not. We call this classifier LUX, short for Language Under eXamination.

Previous work investigated the use of such linguistic aspects as features for similar tasks such as deception detection (Reichel and Lendvai, 2016; Zhou et al., 2004), document clustering (Yu and Hatzivassiloglou, 2003a), text classification (Louis and Nenkova, 2011; Biyani et al., 2016) among others. Related works make use of few (mainly one) of those aspects and the majority of them report an improvement of their results by doing so.

Here we present a set of linguistic aspects that were shown to be correlated to deception. For each of these aspects, we present their contextual definition, along with a short literature review and a description of the methods we use to evaluate its presence or absence in a given piece of text. The objective is to build LUX (Language Under eXamination), a Fake News Classifier, effectively using these linguistic aspects to estimate the likelihood of an article containing fake news. Here, we present the results obtained with two baseline language models (BERT[11] (Devlin et al., 2018) and Word2Vec (W2V) (Mikolov et al., 2013)) towards building this classifier.

We are aware of an imbued redundancy that our features might present, since the aspects analyzed by the different approaches, in some cases, overlap with each other, but expect that the eventual bias this redundancy might add to the model can be overcome with the implementations of techniques such as LDA (Linear Discriminant Analysis) or PCA (principal component analysis).

3.1 Linguistic Aspects

Subjectivity Louis and Nenkova (Louis and Nenkova, 2011) observed that general sentences tend to be more subjective. Some of the shallow features that are correlated to the subjectivity level of a sentence are also used in their model, for example, punctuation marks, average number of characters and average number of words.

Pattern[12], a python library for text analysis, states in its section about subjectivity: “Written texts can be broadly categorized into two types: facts and opinions.” Based on a lexicon of adjectives produced for product review analysis, pattern.en provides a function that maps the subjectivity score of a sentence to a range between 0 and 1 depending on the number of adjectives it contains. It also provides implementations of measuring functions for mood and polarity.

Riloff et Wiebe (Riloff et al., 2003) presents a methodology for the creation of the MPQA Subjectivity Lexicon. In summary, the authors: 1) use an automatic subjectivity classifier to label data while also 2) identifying patterns present in the sentences labeled as subjective and 3) use the learned patterns to improve the classification model(1) and iterate between the three steps, making bootstrapping possible. The MPQA Lexicon is also used for us to measure the subjectivity of a given text. Based on the lexicon, (Wilson et al., 2005) also created OpinionFinder, a Subjectivity Classifier.

Another interesting method was presented by (Yu and Hatzivassiloglou, 2003a), where a Naive Bayes classifier is trained over a Wall Street Journal dataset containing two classes: Subjective (every article with type Editorial or Letter to Editor) and Objective (Business or

---

[11] Bidirectional Encoder Representations from Transformers

[12] https://pypi.org/project/Pattern/

News). By analysing low level features on the texts, the NB classifier achieved a 0.91 recall and 0.86 precision on the binary classification task.

In order to measure the subjectivity of a text, two values are calculated. Both are a sum of each word's subjectivity score normalized by the length of the document (in words) but use as reference different lexicons: the TextBlob[13] (a python library based on Pattern[12]) lexicon and the MPQA lexicon, described above.

Specificity Zhou et al. (Yu and Hatzivassiloglou, 2003b) uses specificity and measures it by words depicting the following aspects: perceptual information (sounds, smells, physical sensations and visual details) and spatio-temporal. (Fuller et al., 2009) measure bi-logarithmic type-token ratio (LogTTR) for evaluating specificity.

(Li and Nenkova, 2015) introduced Speciteller, a python framework for fast and accurate prediction of sentence specificity, which was enhanced and presented by (Ko et al., 2019). It introduces a new algorithm that adjust its weights to the training set, making it applicable to any domain, out-of-the-box. Speciteller is a machine learning classifier that uses as input a combination of:

Shallow features extracted from the text Number of words, number of symbols, average number of characters per word, number of stop-words, explicit discourse connectives (Prasad et al., 2008). From lexicons (General Inquirer (Stone et al., 1962), MRC (Wilson, 1988) and MPQA) other features like sentiment, subjectivity, polarity, familiarity, concreteness, imageability and meaningfulness are also evaluated.

Non-sparse features Brown clusters (Brown et al., 1992) are used to classify words into 100 groups and a vector of corresponding cardinality is used to keep track of the frequency of each class in the input text. Speciteller also uses averaged Word embeddings to represent a sentence embedding. These also are 100-dimensional vectors provided by (Turian et al., 2010).

The ablation results show that Speciteller contributes significantly to the LUX classifier and suggest that the framework could be even more impactful if contemporary word embedding generation techniques were to be used.

Complexity (Biyani et al., 2016) focused on the detection of click-baits (that can be seen as a subcategory of fake news) and reported that features used to measure the formality of a text were the most correlated to click bait articles. Using a slang lexicon and a list of bad words, as well as several readability scores, they obtained a reasonable F-1 score of 74,9.

A 1999 paper by (Heylighen and Dewaele, 1999) presents a famous metric for Formality evaluation, named the F-measure (not to be confused with the F1 score). (Pavlick and Tetreault, 2016) present a statistical model for predicting formality, but do not provide access to the model's code.

Another famous work on the formality area is Coh-Metrix (Graesser et al., 2014), but the only access to its implementation is through a simple HTML portal, so we have discarded this option.

Fortunately, a python library[14] provides several readability measuring tools, including known metrics as the Flesch-Kinkaid (Kincaid et al., 1975) and Coleman-Liau (Coleman and Liau, 1975), LIX (Björnsson, 1968) and RIX, which were also used by (Biyani et al., 2016). Those last two metrics are simple but effective, being the first one (LIX) calculated as $W/S + C/W * 100$ where W is the number of words in a text, S is the number of sentences and C is the number of *complex words* (words with more than 6 letters). The RIX metric is a simpler and graded version of LIX and is calculated as C/S.

Another python library[15], initially developed for the AFEL project (d'Aquin et al., 2018), provides more measuring tools for semantic complexity analyzer. The library starts by identifying the entities present in the input text and the relations between them in order to represent it as a knowledge graph which

---

[13] `https://textblob.readthedocs.io/en/dev/`

[14] `https://pypi.org/project/readability/`

[15] `https://github.com/afel-project/pySemanticComplexity/blob/master/pysemcom.py`

is then used to extract metrics as number of nodes, radius, assortativity[16] and other graph properties.

Both *readability* and *pySemCom* libraries are used by us to implement the highest amount of unique metrics for Complexity, Formality and Readability.

Uncertainty According to (Szarvas et al., 2012), "Uncertainty can be interpreted as lack of information: The receiver of the information cannot be certain about some pieces of information".

Rubin (Rubin et al., 2006) provides a solid survey on Certainty Identification. Building on that, (Vincze, 2015) elaborates on the same subject and achieves great results (Vincze et al., 2008) on the CoNLL Shared Task 10, that aimed for the classification of uncertain texts from the BioScope corpus. The approach was implemented very conveniently as a python library for Uncertainty detection, that is used by us for uncertainty measurement. The classifier is a simple model trained on a corpus of words that were assigned a binary label regarding their certainty. The model only requires the input text to be P.O.S-tagged in order to resolve syntactic ambiguity.

(Reichel and Lendvai, 2016) tried to identify hoax-resolving tweets by using the ratio between four data augmented lexicons (knowledge, report, belief, and doubt) as features, along with low-level syntactic features, not achieving good results.

Loughran and McDonald Sentiment Word Lists (Loughran and McDonald, 2011) and MPQA (Deng and Wiebe, 2015) are Uncertainty Lexicons that are leveraged by us for the evaluation of this aspect. A simple average of uncertain words over the number of words of the input text is used in our model.

Affect (Pang and Lee, 2008) is an extensive review of the literature on sentiment analysis and opinion mining that encompasses the field of linguistic aspect evaluation, which this work is focused on.

(Whissell, 2004) provides the Dictionary of Affect in Language, which includes people's mean ratings for the Pleasantness, Activation, and Imagery of close to 9,000 words. The dictionary is a lexicon with ratings representing the two main dimensions of emotional space, valence and arousal, along with another rating for people's assessment of imageability, i.e., how easily it is to form a mental picture of a word.

A better definition of Affect in the context of deception detection is necessary in order to decide which resource is more appropriate for the aspect evaluation, for now we are going to let the experiments evaluations indicate what is the most appropriate way of measuring affect for our task.

(Li and Nenkova, 2015) mention the MRC Psycholinguistic Database (Wilson, 1988) has words annotated w.r.t imageability among other aspects, while VADER (Valence Aware Dictionary and sEntiment Reasoner) (Hutto and Gilbert, 2014) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to social media. Thus, it seems to be quite appropriate for us.

For this aspect we make use of two different sentiment classifiers: VADER and Pattern/TextBlob[13], already mentioned on the Subjectivity section. From each one of the two classifiers we obtain three metrics: the sum of all the positive scores, the sum of all negative scores and the total sum of scores, all averaged respectively by the number of words with positive score, words with negative score and total number of words in the input text. By using these metrics we ensure that statistics as variance and range of emotion within the text is passed to the LUX classifier.

Verbal Immediacy (Mehrabian and Wiener, 1966) first defined Immediacy as a linguistic property that refers to the degree to which a source associates himself/herself with the topics of a message; that is, "immediacy is the degree to which a source approaches or avoids a topic". Based on that definition, (Zhou et al., 2004) measured it by analysing spatial and temporal terms, passive voice ratio, self reference manner and group reference manner, among others. Different works relate the non-immediacy to the presence of deception in text since these try to disassociate oneself

---

[16] `https://en.wikipedia.org/wiki/Assortativity`

from one's communication.

Negative affect and passive voice are some indicators of non-immediacy. Since the first is already addressed by us, we will be using a ratio between passive sentences over the total number of sentences to determine how passive is the text. In this context, a sentence is deemed passive, if it contains a "BE" verb followed by some other, non-BE verb, except for a gerund.

Diversity / Quantity / Pausality Those are syntactic features and some of the previous defined ones already make use of one or more ways of measuring them. For example, the diversity measurement is used to evaluate a sentence's Complexity. Still, there are many different ways to measure diversity and since we intend to remove the redundancy of the features anyways, we will measure it with many different formulas.

In a 2013 article, (Jarvis, 2013) proposed that the six properties of lexical diversity should be measured by Variability, Volume, Evenness, Rarity, Dispersion and Disparity. Using a python library[17], we measure some of those metrics, namely different types of type-token ratio (TTR), vocd (McCarthy and Jarvis, 2007) and measure of textual lexical diversity (MTLD) (McCarthy, 2005).

Other simple aspects are also taken into account, as the overall quantity of words in absolute number and by P.O.S.-tag as well as the pausality, measured by the ratio between punctuation marks and number of sentences.

4 Evaluation and results

In simple terms LUX is a binary model for classifying general text into fake news / real news and it was originally proposed as a way to evaluate the efficiency of the above mentioned linguistic features. Aiming for generality, this model takes a text document (that could be a long article or a simple headline) as sole input and outputs the probabilities of it being fake or not, based on its psycho-linguistic profile and contextual representation. For the latter, different types of text encodings were tested and it became clear that the usage of fixed-size BERT document embeddings outperformed Word2Vec, which was tested on RNN, LSTM (Hochreiter and Schmidhuber, 1997) and Bi-LSTM (Schuster and Paliwal, 1997), with the latter having the best results, but still inferior to BERT.

After performing a grid search with different optimizers, activation functions, learning rates, training epochs and fully connected layer(FLC) dimensions, the initial model was decided to be composed of a simple ReLu[18] activated 64-dimensional FLC with a dropout of 30% attached to the final layer, of dimensionality 2 where a softmax filter would represent the false and true labels probabilities. Adam (Kingma and Ba, 2014) was the best performing optimizer and a combination of $\alpha = 0.001$ over 100 training epochs generally yielded the best results. Figure 1 brings an outline of the model.

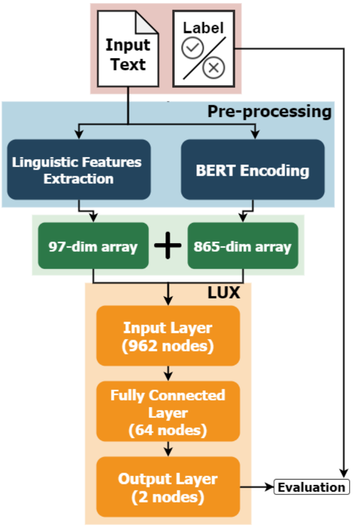

Figure 1: Outline of LUX classifier

All the reported values in Table 2 for Accuracy and F1 score come from a 9-fold training over the data. The results for the two best baseline models are also included, namely the same model using only the BERT document embeddings and only the w2v embeddings over a simple Bi-LSTM with 128 dimensions on the recurrent layer.

Since the data from FEVER18 (Thorne et al., 2018) and Snopes19 (Hanselowski et al., 2019) is composed of short statements a comparative analysis is also presented alongside a V4+EM+T2 run using only the claims as input text, instead of the larger body texts.

The final input for each article is a an ensemble of a document embedding generated by BERT

---

[17] `https://github.com/kristopherkyle/lexical_diversity`

[18] `https://deepai.org/machine-learning-glossary-and-terms/relu`

Table 2: First Evaluation

Model	Dataset	Avg. Acc	Avg. F1
BERT*	V4	0.7365	0.734
W2V*	V4	0.6000	0.598
LUX	V4	0.7896	0.768
LUX	V4+T1	0.7603	0.757
LUX	EM	0.7911	0.778
LUX	V4+EM	0.7928	0.767
LUX	**V4+EM+T2**	**0.8050**	**0.804**
LUX	FEVER18	0.6942	0.691
LUX	Snopes19	0.7405	0.517
LUX	V4+EM+T2#	0.7723	0.708

**:Only the embeddings were used as input, these results serve as baselines to analyse the improvement added by LUX's linguistic features*
#:A version of V4+EM+T2 using the claim (and not the origin body) as input for comparison with other datasets focused on small texts.

trained on the BERT-Large uncased corpus[19] and the 97 linguistic features described in the previous section. A version of the code repository is available at `https://github.com/lucas0/Lux`.

Given the initial results, the robustness added from the a different source, i.e. emergent, with the benefit from balancing classes using the trusted news (T2) yielded the best results. Consequently, it was decided this was the selected subset for the linguistic features ablation analysis.

4.1 Ablation

Table 3 presents the three most impactful positive and negative features, i.e. features that, when removed, most decrease or most increase the accuracy of the model, respectively. Those are all results using as base the best model run, i.e., LUX model over the V4+EM+T2 data, depicted in Table 2. A longer table containing the results for the full ablation analysis can be found within the supplementary material (appendix 2).

Positive Features(PF): When individually removed, each of the 97 features of the model, 50 have report a decreased accuracy of the model by an average of 0.056%, where 21 'contribute' with more than the average of all the positive features and only 10 features decrease more than 1% accuracy when absent. All three top PF fall into the Quantity group, as P.O.S.-tag counts, while most of the most sophisticated, i.e. higher semantic level, make to the top 10. Besides the ones featured(pun intended) in the table, the top 10 also comprises, unordered: Pausability, Coleman-Liau informality score, specificity, measure of lexical textual diversity(MTLD), and three features from the semantic complexity evaluator (Venant and d'Aquin, 2019): assortativity, average number of in-links, and the density. In short, those features are metrics from a graph generated from entities identified in the text, when matched against DBpedia knowledge graph. They refer to, respectively, the similarity of connections with respect to the vertice the number of edges a vertex has to other vertices; the number of links that go from entities of the global DBPedia to the identified entities; and the density of a graph stresses how much nodes are connected to each other.

Negative Features(NF): As expected, the negative features account for the other 47 features. On average, each negative feature increases the accuracy of the model by 0.6% when removed individually. From those, 17 have a better-than-average impact. Avoiding the risk of removing important features from the model and given the high number of negative features, we mention the 9 features that, when **not** considered, improved LUX's accuracy by more than 1%, but focus the discussion on the top 3. Our results point to the number of VBD (verbs, in the past tense form) in the input text as being the third least important feature of the model, while the top two NF are metrics from the same complexity evaluation approach mentioned above. They are nbTypesStd and diameter of nodes, meaning respectively: the standard deviation on the number of different link types per node and the "spreadness"(sic.) of concepts, i.e., the more unrelated and specific concepts we have, the higher the diameter will be. The other six NF improved the accuracy of the model in more than 1% when removed are: the number of words P.O.S.-tagged as PDT (predeterminer), two readability metrics (Dale–Chall and Flesch Reading Ease) and three other features from PySemCom: number of entities, entities density in the text, standard deviation over the number of in-links.

---

[19] `github.com/google-research/bert/blob/master/README.md`

Table 3: Ablation Results

Feat.Idx	Feature	Avg. Acc	≈Removal Impact
	Most Positive Features		
55	‘CD’	0.7864	-1.8%
74	‘RBR’	0.7871	-1.7%
71	‘PRP’	0.7904	-1.4%
	Most Negative Features		
17	diameter	0.8215	+1.6%
23	nbTypesStd	0.8208	+1.5%
81	‘VBD’	0.8201	+1.5%

5 Conclusion and Future Work

This work has done the following two significant contributions: i)the consolidation of the VERITAS Dataset, which is unique due its provision of organic origins for each given claim in the collection, which has, in turn, been manually verified by FCAs. Given the completeness of the released data, it can be an useful resource for a number of related tasks, namely: Document Retrieval, Stance Detection and Claim Validation. As a second contribution, we have confirmed the hypothesis that the inclusion of linguistic metrics as model features allows for a better text classification performance, at least in the target task of identifying fake-news.

After having set up an initial version of the classifier, named LUX, we could demonstrate an improvement from its first evaluation by increasing the quality and quantity of the training data, as well removing the most negative features from the model. The final LUX version performs better than both tested baselines. When used to evaluate the quality of datasets, LUX yields better scores when trained with VERITAS, than when compared with two other fake-news datasets, FEVER18 and Snopes19.

Future work would involve the development of an automatic origin identification step for the VERITAS dataset would allow for a much larger version of it, which in turn could further enhance the classification model (LUX). If this step is achieved, a bootstrapping loop for claim veracity checking with origin identification would be complete, and both the inclusion of new entries to the data collection as well as the further training of classification model could be fully automated, having as their only bottleneck, the permanent scraping of manually fact-checked claims, which is already an automatic process.

Another enhancement being added to this work is the output and analysis of BERT attention weights (Vaswani et al., 2017) for both explainability and interpretability of the model. (Yin et al., 2016; Rush et al., 2015)

Increasing the size of the VERITAS dataset could also be achieved by leveraging the work done by (Hanselowski et al., 2019) and identifying as the origins of a claim, the website containing the snippets annotated as ’supportive’ of the claim. This task is currently ongoing.

Acknowledgments

This work was supported by the Insight SFI Research Centre for Data Analytics (grant No 12/RC/2289_P2).

References

Lucas Azevedo. 2018. Truth or lie: Automatically fact checking news. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 807–811. International World Wide Web Conferences Steering Committee.

Lucas Azevedo and Mohamed Moustafa. 2019. Veritas annotator: Discovering the origin of a rumour. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics (ACL).

Raju Balakrishnan and Subbarao Kambhampati. 2011. Sourcerank: Relevance and trust assessment for deep web sources based on inter-source agreement. In *Proceedings of the 20th International Conference on World Wide Web*, WWW ’11, pages 227–236, New York, NY, USA. ACM.

Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. 2016. ” 8 amazing secrets for getting more clicks”: Detecting clickbaits in news streams using article informality. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Carl-Hugo Björnsson. 1968. *Läsbarhet: Lesbarkeit durch Lix.(Aus dem Schwedischen)*. Liber.

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Mathieu d'Aquin, Dominik Kowald, Angela Fessl, Elisabeth Lex, and Stefan Thalmann. 2018. Afel-analytics for everyday learning. In *Companion Proceedings of the The Web Conference 2018*, pages 439–440.

Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1323–1328.

Leon Derczynski and Kalina Bontcheva. 2014. Pheme: Veracity in digital social networks. In *UMAP workshops*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Christie M Fuller, David P Biros, and Rick L Wilson. 2009. Decision support for determining veracity via linguistic-based cues. *Decision Support Systems*, 46(3):695–703.

Arthur C Graesser, Danielle S McNamara, Zhiqang Cai, Mark Conley, Haiying Li, and James Pennebaker. 2014. Coh-metrix measures text characteristics at multiple levels of language and discourse. *The Elementary School Journal*, 115(2):210–229.

Andreas Hanselowski, Christian Stab, Claudia Schulz, Zile Li, and Iryna Gurevych. 2019. A richly annotated corpus for different tasks in automated fact-checking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL2019)*.

Naeemul Hassan, Bill Adair, James T Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The quest to automate fact-checking. *world*.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center "Leo Apostel", Vrije Universiteit Brüssel*, 4.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Benjamin D Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media*.

Benjamin D Horne, Sara Khedr, and Sibel Adali. 2018. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Twelfth International AAAI Conference on Web and Social Media*.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Sonnet Ireland. 2018. Fake news alerts: Teaching news literacy skills in a meme world. *The Reference Librarian*, 59(3):122–128.

Scott Jarvis. 2013. Capturing the diversity in lexical diversity. *Language Learning*, 63:87–106.

Anoop K, Manjary Gangan, Deepak P, and Lajish V L. 2019. *Leveraging Heterogeneous Data for Fake News Detection*, pages 229–264.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019. Domain agnostic real-valued specificity prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6610–6617.

Caroline W Lee. 2010. The roots of astroturfing. *Contexts*, 9(1):73–75.

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of finance*, 66(1):35–65.

Annie Louis and Ani Nenkova. 2011. Automatic identification of general and specific sentences by leveraging discourse annotations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 605–613, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Philip M McCarthy. 2005. *An assessment of the range and usefulness of lexical diversity measures and the potential of the measure of textual, lexical diversity (MTLD)*. Ph.D. thesis, The University of Memphis.

Philip M McCarthy and Scott Jarvis. 2007. vocd: A theoretical and empirical evaluation. *Language Testing*, 24(4):459–488.

Albert Mehrabian and Morton Wiener. 1966. Non-immediacy between communicator and object of communication in a verbal message: application to the inference of attitudes. *Journal of Consulting Psychology*, 30(5):420.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Tanushree Mitra and Eric Gilbert. 2015. Credbank: A large-scale social media corpus with associated credibility annotations. In *Ninth International AAAI Conference on Web and Social Media*.

Ndapandula Nakashole and Tom M Mitchell. 2014. Language-aware truth assessment of fact candidates. In *ACL (1)*, pages 1009–1019.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1003–1012. International World Wide Web Conferences Steering Committee.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 231–240, Melbourne, Australia. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Uwe D Reichel and Piroska Lendvai. 2016. Veracity computing from lexical cues and perceived certainty trends. *arXiv preprint arXiv:1611.02590*.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 25–32.

Victoria L Rubin, Elizabeth D Liddy, and Noriko Kando. 2006. Certainty identification in texts: Categorization model and manual tagging results. In *Computing attitude and affect in text: Theory and applications*, pages 61–76. Springer.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A platform for tracking online misinformation. In *Proceedings of the 25th international conference companion on world wide web*, pages 745–750.

Philip J Stone, Robert F Bales, J Zvi Namenwirth, and Daniel M Ogilvie. 1962. The general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7(4):484.

György Szarvas, Veronika Vincze, Richárd Farkas, György Móra, and Iryna Gurevych. 2012. Cross-genre and cross-domain detection of semantic uncertainty. *Computational Linguistics*, 38(2):335–367.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Rémi Venant and Mathieu d'Aquin. 2019. Towards the prediction of semantic complexity based on concept graphs. In *12th International Conference on Educational Data Mining (EDM 2019)*, pages 188–197.

Veronika Vincze. 2015. *Uncertainty detection in natural language texts*. Ph.D. thesis, szte.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC bioinformatics*, 9(11):1–9.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Cynthia Whissell. 2004. Using computer-scored measures of emotion and style to discriminate among disputed and undisputed pauline and non-pauline epistles. *Perceptual and motor skills*, 98(3_suppl):1117–1125.

Michael Wilson. 1988. Mrc psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior research methods, instruments, & computers*, 20(1):6–10.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 34–35.

You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward computational fact-checking. *Proceedings of the VLDB Endowment*, 7(7):589–600.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Hong Yu and Vasileios Hatzivassiloglou. 2003a. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136. Association for Computational Linguistics.

Hong Yu and Vasileios Hatzivassiloglou. 2003b. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 129–136.

Lina Zhou, Judee K Burgoon, Jay F Nunamaker, and Doug Twitchell. 2004. Automating linguistics-based cues for detecting deception in text-based asynchronous computer-mediated communications. *Group decision and negotiation*, 13(1):81–106.

6 Appendix

6.1 Description of VERITAS fields

Fact-Checking Article URL The article where the fact-checker journalist analyses the claim, its source(s), characteristics, possible counter arguments, etc.

Checked Claim The main affirmation being verified in the article.

Claim Label The verdict, along with the source document of a fact candidate compose the input/outcome pairs of the dataset to be used in our classification model. In other applications or tasks it might not even be necessary.

We assign the gold-standard status to this annotation, given that each one of those checked documents was manually investigated by one or more fact-checking journalists, before coming to a verdict regarding its veracity, and thus, are as trustworthy as the journalists and correspondent fact-checking agencies themselves.

Different FCAs use different labels, e.g.'mostly-true', 'mixture', 'unproven', etc. consequently there is a need for normalization or removal of the ones that cannot be directly mapped into "true" nor "false".

Tags The set of tags used by the journalist that wrote the fact-checking article. These are mainly used for navigation within the website but could be used for clustering of the dataset and retrieval of other claims regarding the same topic.

FCA Date The date the claim was checked by one of the fact-checking agencies.

Origin URL The URL of the web document that originated the claim, i.e. its origin. Here, origin is defined as a source that directly supports the claim.

Note that an origin does not have to be the very first article that stated the claim and that there could be multiple origins for a single claim.

Origin Domain The origin URL domain. This can have great impacts in results of a neural network classifier's accuracy, or even in the weighting of a simpler classifier method. Examples of using the URL domain as a feature for it's content veracity are not new. (Nakashole and Mitchell, 2014; Balakrishnan and Kambhampati, 2011)

Origin Body The whole text extracted from the origin URL. Which method is used to obtain the Origin Body is the main difference across the versions, as discussed above.

Origin Title The title of the origin page. This is another possibly useful feature for related tasks or extra features for our classifier. (Popat et al., 2017)

Since the title and the checked claim have similar lengths, using this attribute instead of the whole origin text would have probably yielded better results on the stance classification ranking.

Origin Summary Besides being faster than the previously used crawling methods, the current version of the crawler[5] also generates a summary of the origin. This could be a valuable piece of information but would demand checking whether it is a valid depiction of the content of the origin.

Origin Keywords Similar to the Tags of the fact-checking article with the difference that these are obtained by the great article curator newspaper3k[20]. This could also be used as a feature for the Origin Identification Classifier (see on Future Work section).

Origin Date The date at which the origin article was published.

Origin Author The author of the origin article.

---

[20] `https://newspaper.readthedocs.io/en/latest/`

6.2 LUX's full ablation table

Table 4: Ablation Results (Ordered from most Positive Features to most Negative Features)

Feat.Idx	Feature	Avg. Acc	≈Removal Impact
55	'CD'	0.8237	-0.01862
74	'RBR'	0.82304	-0.01796
71	'PRP'	0.81973	-0.01465
52	'.'	0.81775	-0.01267
2	Coleman-Liau	0.81709	-0.01201
18	assortativity	0.81576	-0.01068
24	nbLinkInMean	0.81576	-0.01068
41	Measure of lexical textual diversity (MTLD)	0.81576	-0.01068
94	Speciteller scores	0.81576	-0.01068
29	densityDBPedia	0.8151	-0.01002
16	radius	0.81444	-0.00936
43	MTLD (moving average, bi-directional)	0.81444	-0.00936
34	Simple TTR	0.81378	-0.0087
40	Hypergeometric distribution D (HDD)	0.81378	-0.0087
76	'RP'	0.81378	-0.0087
15	nbNodes	0.81246	-0.00738
26	nbLinkOutMean	0.81246	-0.00738
6	SMOGIndex	0.8118	-0.00672
50	')'	0.8118	-0.00672
70	'POS'	0.8118	-0.00672
82	'VBG'	0.81113	-0.00605
7	RIX	0.81047	-0.00539
64	'MD'	0.81047	-0.00539
77	'SYM'	0.81047	-0.00539
68	'NNS'	0.80981	-0.00473
85	'VBZ'	0.80981	-0.00473
27	nbLinkOutStd	0.80915	-0.00407
32	nbNodes-yago	0.80915	-0.00407
46	'#'	0.80849	-0.00341
59	'IN'	0.80849	-0.00341
62	'JJS'	0.80849	-0.00341

Continued on next page

Table 4 – *Continued from previous page*

Feat.Idx	**Feature**	**Avg. Acc**	**≈Removal Impact**
93	Sum TextBlob's Subjectivity score over sentences	0.80849	-0.00341
87	'WP'	0.80783	-0.00275
1	ARI	0.80717	-0.00209
28	nbNodes-DBPedia	0.80717	-0.00209
51	','	0.80717	-0.00209
35	Root TTR	0.8065	-0.00143
39	Moving average TTR (MATTR)	0.8065	-0.00143
53	':'	0.8065	-0.00143
86	'WDT'	0.8065	-0.00143
12	nbUniqueConcepts	0.80584	-0.00076
13	conceptsWordsRatio	0.80584	-0.00076
38	Mean segmental TTR (MSTTR)	0.80584	-0.00076
45	#tokens	0.80584	-0.00076
63	'LS'	0.80584	-0.00076
65	'NN'	0.80584	-0.00076
66	'NNP'	0.80584	-0.00076
84	'VBP'	0.80584	-0.00076
36	Log TTR	0.80518	-0.0001
72	'PRP$'	0.80452	0.00056
0	Kincaid	0.80386	0.00122
57	'EX'	0.80386	0.00122
79	'UH'	0.80386	0.00122
80	'VB'	0.80386	0.00122
91	Sum TextBlob's Polarity score over sentences	0.80386	0.00122
58	'FW'	0.8032	0.00188
95	Count of '.'-tag tokens	0.8032	0.00188
22	nbTypesMean	0.80254	0.00254
31	density-Schema	0.80254	0.00254
54	'CC'	0.80254	0.00254
78	'TO'	0.80254	0.00254
19	density	0.80187	0.0032
47	'$'	0.80187	0.0032
60	'JJ'	0.80187	0.0032
21	textDensityStd	0.80121	0.00387

Continued on next page

Table 4 – *Continued from previous page*

Feat.Idx	**Feature**	**Avg. Acc**	**≈Removal Impact**
30	nbNodes-Schema	0.80121	0.00387
37	Mass TTR	0.80121	0.00387
44	#terms	0.80121	0.00387
61	‘JJR’	0.80121	0.00387
67	‘NNPS’	0.80121	0.00387
90	‘‘‘’	0.80121	0.00387
9	nbNodesKB	0.80055	0.00453
10	nbWord	0.80055	0.00453
48	’’’’’’	0.80055	0.00453
49	‘(’	0.80055	0.00453
56	‘DT’	0.80055	0.00453
75	‘RBS’	0.79989	0.00519
92	Sum VADER’s Polarity score over sentences	0.79989	0.00519
4	GunningFogIndex	0.79923	0.00585
5	LIX	0.79791	0.00717
33	density-yago	0.79725	0.00783
42	Measure of lexical textual diversity (moving average, wrap)	0.79725	0.00783
88	‘WP$’	0.79725	0.00783
73	‘RB’	0.79658	0.0085
14	uniqueConceptsWordsRatio	0.79592	0.00916
83	‘VBN’	0.79592	0.00916
25	nbLinkInStd	0.79526	0.00982
3	FleschReadingEase	0.7946	0.01048
89	‘WRB’	0.7946	0.01048
8	DaleChallIndex	0.79328	0.0118
11	nbConcepts	0.79328	0.0118
69	‘PDT’	0.79129	0.01379
20	textDensityMean	0.78997	0.01511
81	‘VBD’	0.78997	0.01511
23	nbTypesStd	0.78931	0.01577
17	diameter	0.78865	0.01643

Diagnosing Transformers in Task-Oriented Semantic Parsing

Shrey Desai **Ahmed Aly**
Facebook
{shreyd, ahhegazy}@fb.com

Abstract

Modern task-oriented semantic parsing approaches typically use seq2seq transformers to map textual utterances to semantic frames comprised of intents and slots. While these models are empirically strong, their specific strengths and weaknesses have largely remained unexplored. In this work, we study BART (Lewis et al., 2020) and XLM-R (Conneau et al., 2020), two state-of-the-art parsers, across both monolingual and multilingual settings. Our experiments yield several key results: transformer-based parsers struggle not only with disambiguating intents/slots, but surprisingly also with producing syntactically-valid frames. Though pre-training imbues transformers with syntactic inductive biases, we find the ambiguity of copying utterance spans into frames often leads to tree invalidity, indicating span extraction is a major bottleneck for current parsers. However, as a silver lining, we show transformer-based parsers give sufficient indicators for whether a frame is likely to be correct or incorrect, making them easier to deploy in production settings.

1 Introduction

Task-oriented semantic parsing—mapping textual utterances to semantic frames—is a critical component of modern conversational AI systems (Gupta et al., 2018; Aghajanyan et al., 2020). Recent methodology casts parsing as transduction, using seq2seq pre-trained transformers to produce linearized parse trees (Aghajanyan et al., 2020; Chen et al., 2020; Li et al., 2021); here, each frame token is either *copied* from the utterance or *generated* from an ontology. Compared to explicit grammar-based approaches (Gupta et al., 2018), this plug-and-play of transformers simplifies the learning objective and scales to multilingual settings, but the lack of provenance makes it challenging to understand model behavior "under the hood."

In this work, we investigate the strengths and weaknesses of transformer-based semantic parsers and provide modeling directions based on data-driven insights. Specifically, we study BART (Lewis et al., 2020) and XLM-R (Conneau et al., 2020), two state-of-the-art conversational semantic parsers, on both monolingual (TOP/TOPv2; (Gupta et al., 2018; Chen et al., 2020)) and multilingual (MTOP; (Li et al., 2021)) datasets. The compositionality of utterances in these datasets provide a strong testbed for resolving both complex syntactic structure and semantic ambiguity, mirroring the types of challenges our parsers are likely to encounter in practice.

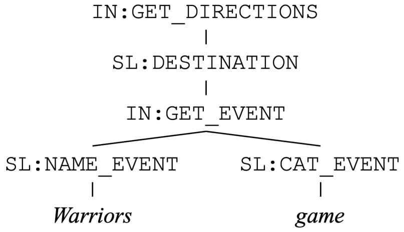

Figure 1: Example decoupled semantic frame representation (Aghajanyan et al., 2020) for the utterance *Directions to the Warriors game*.

We design our experiments around three main questions. First, broadly speaking, what types of errors do transformer-based parsers make? We begin by annotating 500+ predicted frames across 6 languages and categorize them with fine-grained types. We find transformer-based parsers struggle not only with *classification* (i.e., disambiguating intents/slots) but also *planning* (i.e., switching between copying/generating). Planning errors are more egregious: misplacing close brackets, for example, can violate tree constraints, rendering the entire frame unusable.

Next, we investigate transformer-based parsers' abilities to generate syntactically-valid trees. Specifically, are planning mistakes caused by general uncertainty, or worse, a pathology of seq2seq learning? To address this, we devise an oracle set-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 57–62
August 1–6, 2021.

split	TOP	TOPv2	MTOP
train	31,279	124,579	73,956
dev	4,462	17,160	10,852
test	9,042	38,785	30,541

Table 1: Dataset splits for TOP, TOPv2, and MTOP.

N	d_{model}	d_{ff}	h	d_k	d_v
6	1024	4096	16	64	64

Table 2: Dimensions of transformer decoder added to XLM-R for MTOP fine-tuning. Notation is borrowed from Vaswani et al. (2017).

ting where a model conditions on partially gold information (either utterance spans or syntactic structure) and predicts the remaining parts of the frame. Surprisingly, we find conditioning on gold spans—not gold structures—results in near-perfect trees at most depths, pointing towards span extraction as a major bottleneck for current parsers.

Finally, though transformer-based parsers are susceptible to error, ideally, we should be able to proactively diagnose mistakes. Using features from model generations (e.g., confidence), can we intrinsically judge if a sequence is correct or incorrect? Encouragingly, we show that a confidence estimation system combining a transformer-based parser and feature-based classifier can detect *correct* frames with 90%+ F1, indicating usability in production settings.

2 Experimental Setup

We conduct experiments on the following task-oriented semantic parsing datasets: (1) **TOP:** parallel corpus consisting of English utterances and corresponding semantic frames (Gupta et al., 2018); (2) **TOPv2:** monolingual extension of TOP to 6 domains (Chen et al., 2020); (3) **MTOP:** multilingual extension of TOP spanning English, Spanish, French, German, Hindi, and Thai (Li et al., 2021). Table 1 shows train, dev, and test splits for the datasets.

Each dataset sample consists of a textual utterance x and (linearized) semantic frame y. Here, frames are in decoupled form (Aghajanyan et al., 2020), as each token is derived either from *copying* from the utterance or *generating* from the ontology (see Figure 1). Following prior work, we fine-tune seq2seq transformers to maximize the log likelihood of the gold frame token at each timestep: $\sum_{(x,y)} \sum_t \log P(y_t|y_{<t}, x; \theta)$.

split	TOP	TOPv2	MTOP
dev	85.41	87.53	76.00
test	85.74	87.52	77.20

Table 3: Exact match (EM) of BART and XLM-R on TOP/TOPv2 and MTOP, respectively.

setting	TOP	TOPv2	MTOP
model	BART	BART	XLM-R
dropout	8.68e-2	1.82e-1	0
batch size	16	16	16
epochs	50	50	50
optimizer	Lamb	Lamb	Lamb
lr	3.72e-4	4.88e-4	6.91e-4
weight decay	6.25e-7	6.26e-7	6.25e-7
swa lr	2.08e-4	1.86e-4	3.96e-4
swa start	8945	18876	19450
swa freq	219	233	185
scheduler	exp	exp	exp
warmup	5000	5000	5000
gamma	0.95	0.95	0.95

Table 4: Hyperparameters for fine-tuning models on TOP, TOPv2, and MTOP.

On TOP/TOPv2, we fine-tune BART (Lewis et al., 2020), a seq2seq transformer pre-trained with a denoising autoencoder objective on monolingual corpora, and on MTOP, we fine-tune XLM-R (Conneau et al., 2020) (equipped with a randomly-initialized decoder), a transformer encoder pre-trained with a masked language modeling objective on multilingual corpora. For XLM-R, specifically, we attach a randomly-initialized decoder (see Table 2). Table 3 shows model performance as judged by exact match. Hyperparameters for all models are listed in Table 4.

3 Error Analysis

In this section, we seek to better understand the types of errors transformer-based parsers make across both monolingual and multilingual settings.

3.1 Error Types

To standardize our analysis, we categorize model errors under the following types: **intent** (incorrect intent prediction), **slot** (incorrect slot prediction), **out-of-domain** (incorrect out-of-domain intent prediction), **mode** (confusion between copying an utterance token or generating an ontology token), and **leaf** (incorrect span in a frame leaf slot). In addition, we report the syntactic **validity** of parse trees separately, though we note mode errors typically result in invalid constructions.

	Exact Match			Tree Validity		
d	TOP	TOPv2	MTOP	TOP	TOPv2	MTOP
1	78.03	86.58	84.75	98.65	94.57	91.23
2	92.30	90.67	85.73	96.97	96.82	93.80
3	90.94	88.50	74.56	97.10	96.35	90.85
4	88.24	86.32	64.53	95.93	95.47	85.73
5	83.39	83.63	44.29	94.29	94.85	69.55
6	83.06	84.54	44.44	94.00	94.45	62.50

Table 5: Benchmarks of BART and XLM-R on TOP/TOPv2 and MTOP, respectively, according to exact match and tree validity at increasing tree depths (d).

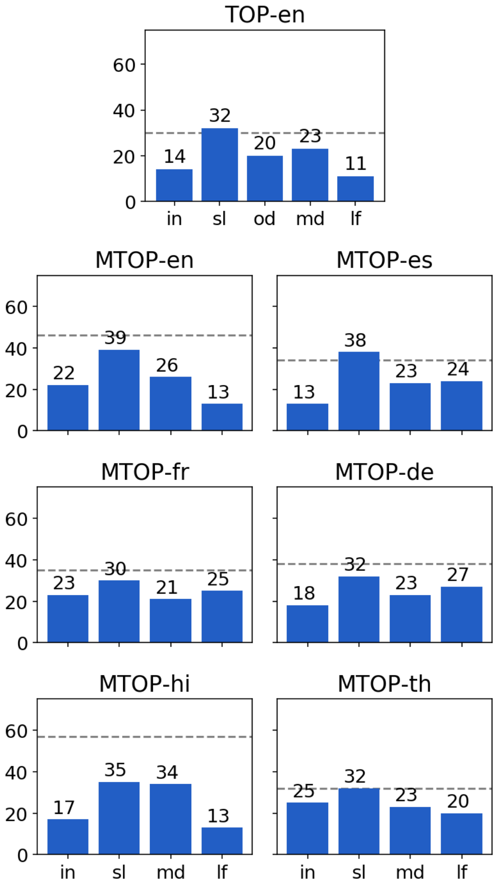

Figure 2: Distribution of errors across TOP and MTOP categorized by intent (in), slot (sl), out-of-domain (od), mode (md), and leaf (lf). Dashed lines indicate the percentage of trees which are syntactically valid.

One complicating factor is that a predicted sequence may potentially contain several errors, and because decoding is conducted autoregressively, a given error may be influenced by earlier errors (if any such exist). Therefore, to reduce the number of confounding variables, we only consider settings where an incorrect prediction has gold history $\mathrm{argmax}_{y_i} P(y_i|y^*_{<t}, x) \neq y^*_i$; put another way, we only count the *first* error in a sequence.

Using the framework discussed above, we annotate 700 errors across BART and XLM-R on TOP and MTOP, respectively; 100 errors are from TOP and 6×100 errors are from MTOP (100 per language).

3.2 Results

Table 5 benchmarks overall model performance and Figure 2 categorizes errors with fine-grained types; from these results, we draw the following conclusions:

Transformer-based parsers typically struggle with both classification and planning. In the seq2seq formulation, models must jointly *classify* (i.e., provide intent and slot labels) and *plan* (i.e., switch between copying and generating) when producing a semantic frame. Our results show intent/slot and mode errors, which generally fall under the theme of classification and planning, respectively, account for nearly 70-80% of errors. A key observation, however, is that classification and planning error statistics are relatively consistent across languages, suggesting our models may not need language-specific fine-tuning to address these particular errors.

Nearly 40% of incorrectly predicted frames are syntactically invalid. Surprisingly, a large percentage of incorrectly predicted frames violate tree constraints; for linearized frames, this implies the number of open brackets (`[in` or `[sl`) do not match the number of close brackets (`]`). Though well-formedness *is* correlated with depth, we see tree validity (1) is not substantially improved by increasing the number of monolingual samples (TOP → TOPv2) and (2) drops off quite rapidly for multilingual samples (TOP/TOPv2 → MTOP).

Span extraction is more challenging in multilingual settings. Leaf errors in English `(TOP|MTOP)-en` are typically twice as lower compared to those in non-English languages `MTOP-(es|fr|de|hi|th)`. Upon closer inspection, we find most leaf errors in English are relatively benign; the model may drop a preposition when copying a span (e.g., *Monday* as opposed to *on Monday*). However, for languages beyond English, extracted spans in leaf slots typically consist of hallucinated or duplicated subwords, which are much more serious in nature. Finally, though languages with non-projective structures (e.g., German) can populate leaf slots with non-contiguous spans, we noticed errors on these types of samples were infrequent.

Out-of-domain detection is also a significant source of error. TOP, in particular, mixes the canonical semantic parsing task with out-of-domain detection by assigning such utterances the frame `[in:unsupported ]`.[1] Though well-motivated, roughly 20% of errors are related to incorrect out-of-domain predictions, suggesting our models have not precisely learned the boundary between in-domain and out-of-domain utterances. If high detection accuracy is preferred, multi-tasking parsers in this fashion may not be an effective use of parameters (assuming more data is not available); instead, out-of-domain detection can be conducted independently with alternate methodology (Gangal et al., 2019).

4 Syntactic Structure

Our case study above demonstrates transformer-based parsers can produce syntactically-invalid frames at a high rate. These structural errors are more serious than disambiguation errors since they render the frame unusable, potentially causing cascading failures in a task-oriented dialog system. Therefore, in this section, we dive deeper into why tree constraints are not satisfied and question the possibility of achieving perfect tree validity.

While transduction models do not explicitly impose tree constraints, there is precedent that strong neural representations do implicitly model tree structures; recent studies demonstrate large-scale pre-training, in particular, imbues strong notions of syntax (Goldberg, 2019; Jawahar et al., 2019; Tenney et al., 2019). Taking these results together, we hypothesize that transformer representations may be "good enough", but instead there exist ambiguous aspects of our task-oriented semantic parsing task which cause tree invalidity.

Previously, we saw transformer-based semantic parsers largely struggled with classification- and planning-related errors. Therefore, the question we pose is: if we resolve these ambiguities by creating oracle models, can we achieve perfect tree validity? This setup also enables us to gain a deeper understand of the upper-bound performance of transformer-based semantic parsers, even as their representations get stronger.

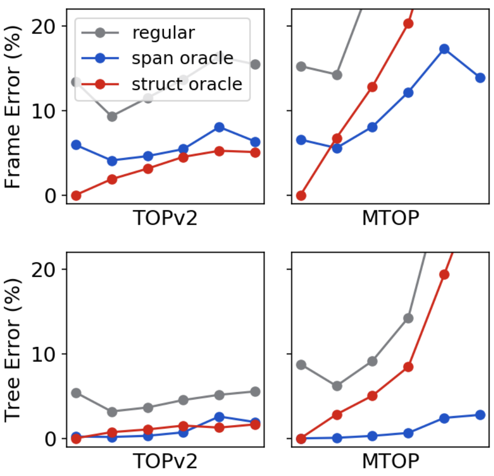

Figure 3: Exact match (EM) and tree validity (TV) error (%) of the regular, span oracle, and structure oracle models on TOPv2 and MTOP. Dots from left → right indicate increasing frame compositionality (the graph depths of 1 → 6).

Oracle Models. Because classification and planning target inherently different phenomena, creating an oracle that simultaneously makes both less ambiguous is challenging. Instead, we experiment with two separate oracles—**span oracle** and **structure oracle** models for classification and planning, respectively—which map an utterance x along with a "partially gold" snippet z to generate the frame y, inducing the objective $\sum_{(x,y,z)} \sum_t \log P(y_t|y_{<t}, x, z; \theta)$.

For example, given an utterance x *Where can I see fireworks tonight?* and frame y `[in [sl` *fireworks* `[sl` *tonight* `] ]`, the span oracle model defines z as `[span1]` *fireworks* `[span2]` *tonight* and the structure oracle model defines z as `[in [sl [span1] [sl [span2] ] ]`.[2] Here, providing z as input helps the model learn $y \setminus z$; span oracle models optimize for correct structure and structure oracle models optimize for correct spans. Table 6 shows example source and target pairs for the regular, span oracle, and structure oracle models.

Results. Figure 3 shows the oracle model results; we measure both exact match and tree validity error. **A key phenomenon we observe is that conditioning on gold spans results in near-zero tree validity error at most depths.** Surprisingly, we see conditioning on gold structures (to stress, the *exact* syntactic structure) never consistently results in well-formed trees, especially as the depth in-

---

[1] There also exist more fine-grained out-of-domain categories, such as `[in:unsupported-event ]`.

[2] Fine-grained intent/slot labels are omitted for visual clarity, but are included during model training.

model type	utterance x (+ snippet z)	frame y
regular	Where can I see fireworks tonight?	`[in [sl` fireworks `[sl` tonight `] ]`
span oracle	+ `[span1]` fireworks `[span2]` tonight	
struct oracle	+ `[in [sl [span1] [sl [span2] ] ]`	

Table 6: Example source and target pairs for oracle experiments. The span oracle specifies the gold spans while the struct oracle specifies the gold structure. Note that `[in` and `[sl` are used for brevity.

creases. Structure oracle models still suffer from mode errors during generation: augmenting a leaf span with an extra word instead of placing a close bracket, for example, is a typical mistake. Furthermore, we see this problem is magnified in MTOP, which connects to the notion that span extraction tends to be difficult in multilingual settings.

Our experiments suggest seq2seq transformer-based parsers *can* achieve near-perfect tree validity—even at large depths—provided that span extraction is precise. Currently, however, this is a major source of ambiguity our parsers are not well-equipped to handle, especially when scaling to languages beyond English.

5 Confidence Estimation

Despite the criticism we have presented of state-of-the-art, transformer-based conversational semantic parsers, these models do demonstrate strong performance over prior baselines, and correctly parse a vast majority of samples. A property that can make these models easier to deploy in practice is if they "know what they don't know" (Desai and Durrett, 2020); besides interpretability, this is particularly useful for identifying and correcting errors in tail scenarios via active learning (Dredze and Crammer, 2008; Duong et al., 2018; Sen and Yilmaz, 2020). We frame this problem as confidence estimation (Blatz et al., 2004): given an utterance x, predicted frame y', and gold frame y, we seek to learn a binary classifier which uses target-side features $f(y')$ to estimate $P(y' = y) = \text{sigmoid}(w^\top f(y'))$.

To make our approach as generalizable as possible, we constrain $f(y')$ to be as model-agnostic and recall-oriented as possible. We select the following features: (1) **length:** $|y'|$; (2) **validity:** $\max(0, \sum_i \mathbb{1}[y'_i \in V^+] - \mathbb{1}[y'_i \in V^+])$ where V^+ and V^- are the set of open and close brackets, respectively; and (3) **confidence:** $\frac{1}{|y'|}\sum_i P(y'_t|y'_{<t}, x)$. Using our best transformer-based parsers, we obtain predictions on a held-out set D_{dev} and test set D_{test}. Then, we train and test a SVM on D_{dev} and D_{test}, respectively, using the features defined above.

	TOPv2			MTOP		
	P	R	F1	P	R	F1
SVM	97.2	85.7	91.2	95.0	85.2	89.8
–length	**97.7**	84.8	90.8	**95.1**	84.7	89.6
–validity	97.0	82.6	89.2	94.9	80.5	87.1
–confidence	91.6	**98.8**	**95.1**	85.3	**95.8**	**90.2**

Table 7: Precision (P), recall (R), and F1 of the SVM-based confidence estimator. –x indicates an ablation of feature x (i.e., it is omitted during learning).

In addition to the standard hinge loss, we also add a class imbalance penalty as positive examples are typically 5-8× as prevalent depending on the dataset. We chiefly evaluate the binary classifier's ability to identify semantic frames which are correct (i.e., the positive class). From an active learning standpoint, getting positive samples wrong is more serious than getting negative samples wrong; annotation resources are best directed towards boundary or incorrect predictions.

Table 7 shows the performance and ablations of our confidence estimator. **In both monolingual and multilingual settings, using transformer-based features, we can detect *correct* semantic frames with 90%+ F1.** In particular, we see length and validity largely capture the space of correct frames (recall) and confidence effectively distinguishes between correct and incorrect frames (precision). Practitioners may select an SVM variant depending on whether precision or recall is preferred.

6 Conclusion

In this work, we assess the strengths and weaknesses of seq2seq transformers for task-oriented semantic parsing. These models "know what they don't know", making them easier to depoy in practice, but cannot perfectly model compositional utterances, as indicated by the challenges of span extraction. We believe that modeling efforts in this direction—as opposed to simply annotating more data—can improve parsers substantially.

References

Armen Aghajanyan, Jean Maillard, Akshat Shrivastava, Keith Diedrick, Michael Haeger, Haoran Li, Yashar Mehdad, Veselin Stoyanov, Anuj Kumar, Mike Lewis, and Sonal Gupta. 2020. Conversational Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence Estimation for Machine Translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Xilun Chen, Ashish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-Resource Domain Adaptation for Compositional Task-Oriented Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shrey Desai and Greg Durrett. 2020. Calibration of Pre-trained Transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mark Dredze and Kovy Crammer. 2008. Active Learning with Confidence. In *Proceedings of the Annual Meeting of the Conference on Computational Linguistics (ACL)*.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip Cohen, and Mark Johnson. 2018. Active Learning for Deep Semantic Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. 2019. Likelihood Ratios and Generative Classifiers for Unsupervised Out-of-Domain Detection In Task Oriented Dialog. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Yoav Goldberg. 2019. Assessing BERT's Syntactic Abilities. *arXiv preprint arXiv:1901.05287*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic Parsing for Task Oriented Dialog using Hierarchical Representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pretraining for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A Comprehensive Multilingual Task-Oriented Semantic Parsing Benchmark. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Priyanka Sen and Emine Yilmaz. 2020. Uncertainty and Traffic-Aware Active Learning for Semantic Parsing. In *Proceedings of the Workshop on Interactive and Executable Semantic Parsing (INTEXSEMPAR)*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Semantic Relation-aware Difference Representation Learning for Change Captioning

Yunbin Tu[1]*, Tingting Yao[3], Liang Li[2], Jiedong Lou[3],
Shengxiang Gao[1]†, Zhengtao Yu[1], Chenggang Yan[3]
[1]Yunnan Key Laboratory of Artificial Intelligence,
Kunming University of Science and Technology
[2] Key Lab of Intell. Info. Process., Inst. of Comput. Tech., Chinese Academy of Sciences
[3] Intelligent Information Processing Laboratory, Hangzhou Dianzi University
`{tuyunbin1995,gaoshengxiang.yn}@foxmail.com`
`liang.li@ict.ac.cn, ztyu@hotmail.com`

Abstract

Change captioning is to describe the difference in a pair of images with a natural language sentence. In this task, the distractors, such as the illumination or viewpoint change, bring the huge challenges about learning the difference representation. In this paper, we propose a semantic relation-aware difference representation learning network to explicitly learn the difference representation in the existence of distractors. Specifically, we introduce a self-semantic relation embedding block to explore the underlying changed objects and design a cross-semantic relation measuring block to localize the real change and learn the discriminative difference representation. Besides, relying on the POS of words, we devise an attention-based visual switch to dynamically use visual information for caption generation. Extensive experiments show that our method achieves the state-of-the-art performances on CLEVR-Change and Spot-the-Diff datasets [1].

1 Introduction

Change Captioning aims to describe a semantic change between a pair of "before" and "after" images, which has many practical applications such as facility monitoring (Sakurada and Okatani, 2015), medical imaging (Patriarche and Erickson, 2004), and aerial photography (Gueguen and Hamid, 2015).

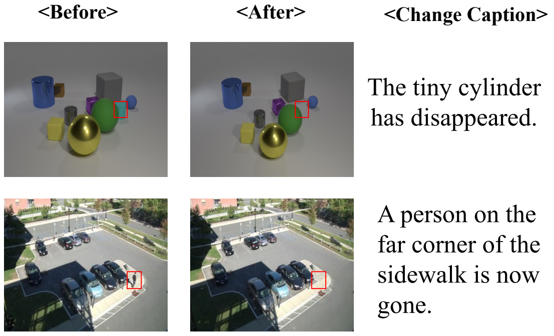

Figure 1: Two examples of change captioning with and without a viewpoint change.

The previous work (Jhamtani and Berg-Kirkpatrick, 2018) introduced this task with an ideal assumption that there is a semantic change between a completely-aligned image pair. However, there is always illumination change in a dynamic world, and same or similar scenes are prone to shoot under different viewpoints. Compared to semantic changes, both illumination and viewpoint changes are irrelevant distractors, so realistic change captioning requires a model: 1) distinguishing semantic changes (e.g., an object has moved) from distractors (e.g., a viewpoint change) and 2) conveying the detected change in a logically and grammatically accurate sentence. To this end, recent works (Park et al., 2019; Shi et al., 2020) focused on addressing change captioning in the presence of distractors.

Despite the progress, there are still two limitations for their approaches. First, the semantic difference was modeled only relying on the semantic features of objects, while ignoring their self-semantic relations. Hence, the feature difference is hard to capture the tiny change. As shown in Figure 1, compared with many unchanged objects, the dropped object is tiny and easy to ignore. Differently, if one of the objects has changed, especially number or position change (*e.g.*, "add", "drop", or "move"), the semantic relations surrounding it would change as well, which would be beneficial to explore the underlying objects that have changed. Second, due to the existing of irrelevant distractors, the model would capture the semantic difference

---

* This work was done when Yunbin Tu visited VIPL research group, CAS and was supervised by Prof. Liang Li.

† Corresponding author

[1] The code of this paper has been made publicly available at `https://github.com/tuyunbin/SRDRL`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 63–73
August 1–6, 2021.

with noises and thus learn a wrong difference representation. However, both distractors are irrelevant to the semantics of image contents. Therefore, the cross-semantic relation between the captured semantic difference and the image pair is beneficial to judge whether the semantic change has actually happened, and further learn the difference representation in the "before" and "after" images.

Besides, during caption generation, previous works exploited visual information to generate each word, which is unnecessary or even misleading (Lu et al., 2017; Song et al., 2017). As words with different part-of-speech (POS) information not only play different grammatical roles in a sentence (Wang et al., 2019), but also have different relationships with the visual information in an image. As shown in the first example of Figure 1, some words (e.g., "tiny", "cylinder" and "disappeared") belong to adjective, noun and verb words, which denote the size, category, and state of the visual object, while the word (i.e.,"the") is a determiner word which does not have corresponding canonical visual signals. Thus, it is useful to introduce the POS of words for switching visual information during change caption generation.

In this paper, we propose a Semantic Relation-aware Difference Representation Learning (SRDRL) network to localize the semantic change in the presence of distractors, and introduce an Attention-based Visual Switch (AVS) to dynamically decide when to use visual information during change caption generation. Specifically, first, a Self-Semantic Relation Embedding block (SSRE) builds semantic relations of objects for each image in the "before"/"after" pair via the self-attention mechanism. The built relations are embedded into image features for computing a relation-embedded feature difference. Second, a Cross-Semantic Relation Measuring block (CSRM) leverages the obtained difference to query the underlying "candidate change" in the each image. Further, CSRM uses the difference to generate an attention gate measuring its cross-semantic relations with respect to each image. Subsequently, the attention gate is applied to the candidate change to distinguish semantic change from the viewpoint/illumination change. Third, the change localizer is introduced to learn the accurate difference representation in the image pair under the guidance of a prior knowledge (the above distinguished information).

Finally, according to POS information of words, an Attention-based Visual Switch (AVS) is devised and incorporated into the caption generator to dynamically control visual information when predicting the next word. Extensive experiments show that our approach outperforms the state-of-the-art change captioning models with a large margin.

In summary, the contributions of this work have threefold: (1) We propose SRDRL that explicitly learns the semantic difference representation in the image pair by embedding self-semantic relations into object features of each image and further measuring the cross-semantic relations between the image pair and their difference. (2) Both SSRE and CSRM blocks are designed to help the change localizer to accurately focus on the changed objects. (3) An AVS is customized to dynamically utilize visual information for caption generation based on the POS information of words.

2 Related Work

Different from conventional image (Liu et al., 2020, 2019; Li et al., 2020; Yan et al., 2019, 2020a, 2021) or video captioning (Deng et al., 2021; Zhang et al., 2017; Tu et al., 2017, 2020; Yan et al., 2020b), change captioning addresses two-image captioning, especially to describe their difference. Jhamtani *et al.* (Jhamtani and Berg-Kirkpatrick, 2018) is the first work for change captioning. However, it is built upon an ideal situation by assuming there are no distractors (illumination/viewpoint change) between a pair of images. To make this task more close to our dynamic world, Park *et al.* and Shi *et al.* (Park et al., 2019; Shi et al., 2020) both aimed to address change captioning in the existence of distractors. On one hand, Park *et al.* directly concatenated the coarse feature difference with the image pair to operate spatial attention to localize the change. However, due to the existing of distractors, when the captured feature difference is not what the model really expects, the spatial attention module could be misled to give fallacious results. On the other hand, Shi *et al.* first exploited a cross-attention mechanism to search the most similar patches between the image pair and they are regarded as the unchanged representation. Then, they subtracted them from the original image to get the difference representation. However, as our aforementioned, the changed object is tiny and easy to ignore, so it is insufficient to capture the difference representation only at feature level.

Different from the above state-of-the-art meth-

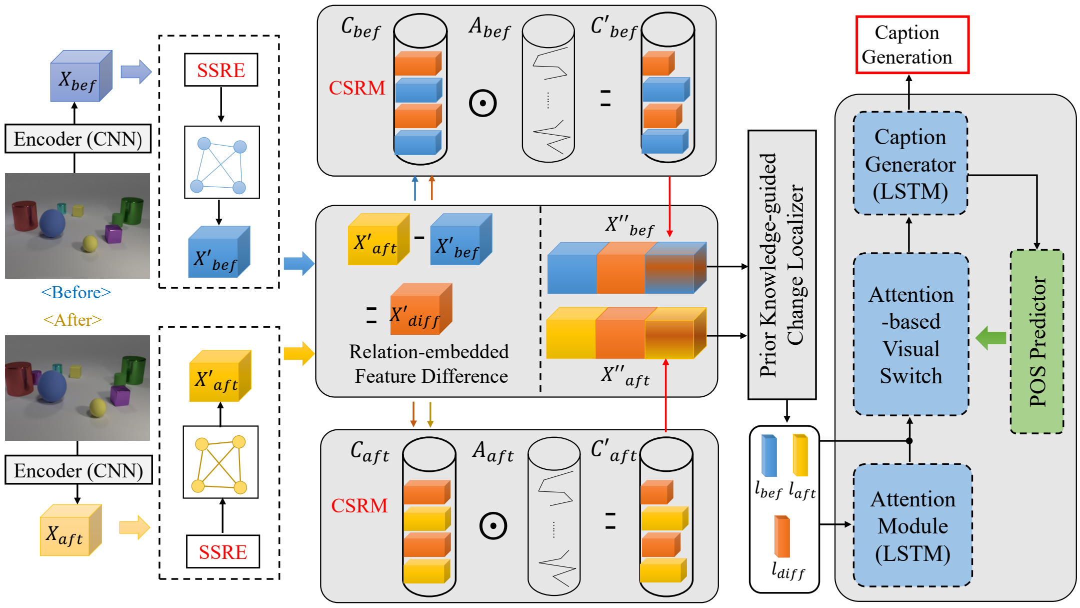

Figure 2: The architecture of the proposed semantic relation-aware difference representation learning (SRDRL) network and an attention-based visual switch (AVS). The SRDRL consists of a self-semantic relation embedding block (SSRE), a cross-semantic relation measuring block (CSRM) and a prior knowledge-guided change localizer. The AVS is incorporated into the caption generator and guided by a POS predictor.

ods, we first use SSRE to improve the fine-grained representation ability of object features by embedding the self-semantic relations among them. Then, we exploit CSRM to distinguish the actual semantic change from irrelevant distractors via measuring cross-semantic relations between the captured candidate difference and the original images. Finally, we use POS information to devise an attention-based visual switch that dynamically determines not only when to use visual information, but also which to use (*e.g.*, "before" and "after"). Compared to the aforementioned methods, our method not only can learn discriminative difference representation, but also can describe it using an accurate natural language sentence.

3 Methodology

We present a semantic relation-aware difference representation learning (SRDRL) network for change localization and devise an attention-based Visual Switch (AVS) under the guidance of POS information for caption generation. When a pair of "before" and "after" images are given (denoted as I_{bef} and I_{aft}), our SRDRL first detects *what* (position, number, attribute, or nothing) has changed in a scene and further decides *where* to localize on both I_{bef} and I_{aft}. Then, during caption generation, the AVS is able to dynamically decide *when* to use visual information and *which* to use (*e.g.*, "before" and "after").

3.1 Semantic Relation-aware Difference Representation Learning Network

3.1.1 Self-Semantic Relation Embedding

Formally, given a pair of I_{bef} and I_{aft}, we first use pre-trained CNN model to extract object-level features and denote them as X_{bef} and X_{aft}, where $X_i \in \mathbb{R}^{C\times H\times W}$; C, H, W indicate the number of channels, height, and width. However, These original object features are independent, and there exist semantic relations among them (Huang et al., 2020; Wu et al., 2019; Yin et al., 2020). Inspired by the self-attention (Vaswani et al., 2017) using in machine translation, the self-semantic relation embedding block (SSRE) relies on it to implicitly model the semantic relations among objects in each image. Specifically, we first reshape X_i to $X_i \in \mathbb{R}^{N\times C}$ $(N = HW)$, where $i \in (bef, aft)$. Then, given (*key*, *value*), SSRE exploits the scaled dot-product attention on queries Q by:

$$SSRE(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1)$$

In our case, the queries, keys and values are all projections of the object features of X_i:

$$(Q, K, V) = \left(X_i W^Q, X_i W^K, X_i W^V\right). \quad (2)$$

Though the SSRE, the semantic relations are embedded in the original object features; both X_{bef} and X_{aft} can be updated to X'_{bef} and X'_{aft}. Finally, we subtract X'_{bef} from X'_{aft} to capture the semantic difference X'_{diff} in the both object feature and relation aspects.

3.1.2 Cross-Semantic Relation Measuring

Due to the existing of distractors, the resulting X'_{diff} would include some irrelevant information, which would be noises for the accurate difference representation learning on both X'_{bef} and X'_{aft}. Thus, we propose a cross-semantic relation measuring block (CSRM) to distinguish the semantic change from the irrelevant illumination or viewpoint change by measuring the cross-semantic relation between the X'_{diff} and X'_{bef} (X'_{aft}). Concretely, the CSRM utilizes the X'_{diff} to first query the possible “candidate change” C_{bef} on the X'_{bef}, and then generates an “attention gate” A_{bef} measuring its semantic relations with respect to X'_{bef}. These are defined by using two separate non-linear transformations:

$$\begin{aligned} C_{bef} &= \phi\left(X'_{diff}W_q^i + X'_{bef}W_v^i + b^i\right), \\ A_{bef} &= \sigma\left(X'_{diff}W_q^g + X'_{bef}W_v^g + b^g\right), \end{aligned} \tag{3}$$

where $W_q^i, W_v^i, W_q^g, W_v^g \in \mathbb{R}^{C\times C}, b^i, b^g \in \mathbb{R}^C$, and C is the dimension of X'_{diff} and X'_{bef} ; σ and ϕ denote the sigmoid and tanh function. The value in the “attention gate” indicates the semantic relevance between the “candidate change” and the “before”. Thus, the more information in the “candidate change” passes through the “attention gate”, the more X'_{diff} is relevant to X'_{bef}.

Next, the CSRM applies the A_{bef} to the C_{bef} to filter all the underlying change information and focus on only the information about semantic change via element-wise multiplication:

$$C'_{bef} = A_{bef} \odot C_{bef}. \tag{4}$$

Besides, the information about semantic change C'_{aft} is computed via the similar operation between the X'_{diff} and X'_{aft} :

$$\begin{aligned} C_{aft} &= \phi\left(X'_{diff}U_q^i + X'_{aft}U_v^i + z^i\right), \\ A_{aft} &= \sigma\left(X'_{diff}U_q^g + X'_{aft}U_v^g + z^g\right), \\ C'_{aft} &= A_{aft} \odot C_{aft}. \end{aligned} \tag{5}$$

3.1.3 Prior Knowledge-guided Change Localizer

After obtaining the C'_{bef} and C'_{aft}, we use them as the prior knowledge to guide the change localizer to learn the difference representation. Specifically, the change localizer first predicts two separate attention maps under the guidance of C'_{bef} and C'_{aft}, respectively:

$$\begin{aligned} X''_{\text{bef}} &= [X'_{\text{bef}}\,; X'_{\text{diff}}\,; C'_{\text{bef}}\,], \\ a_{\text{bef}} &= \sigma\left(\text{conv}_2\left(\text{ReLU}\left(\text{conv}_1\left(X''_{\text{bef}}\right)\right)\right)\right), \\ X''_{\text{aft}} &= [X'_{\text{aft}}\,; X'_{\text{diff}}\,; C'_{\text{aft}}\,], \\ a_{\text{aft}} &= \sigma\left(\text{conv}_2\left(\text{ReLU}\left(\text{conv}_1\left(X''_{\text{aft}}\right)\right)\right)\right), \end{aligned} \tag{6}$$

where $[;]$, conv, and σ indicate concatenation, convolutional layer, and element-wise sigmoid, respectively. After that, the difference representation features l_{bef} and l_{aft} are attended to by applying a_{bef} and a_{aft} to the input image features X'_{bef} and X'_{aft} :

$$\begin{aligned} l_{\text{bef}} &= \textstyle\sum_{H,W} a_{\text{bef}} \odot X'_{\text{bef}}, l_{\text{bef}} \in \mathbb{R}^C, \\ l_{\text{aft}} &= \textstyle\sum_{H,W} a_{\text{aft}} \odot X'_{\text{aft}}, l_{\text{aft}} \in \mathbb{R}^C. \end{aligned} \tag{7}$$

3.2 Change Caption Generation

3.2.1 POS Predictor

Inspired by POS used in machine translation (Yin et al., 2019), we dynamically predict POS tags [1] of target words based on the previous hidden states $h_c^{(t-1)}$ of the caption generator. The predicted tags help the captioning model use visual information in a dynamic way.

Specifically, at time t, $h_c^{(t-1)}$ is first fed into a single hidden layer with the ReLU activation function:

$$d_t^p = \text{ReLU}\left(W_p^{(1)}h_c^{(t-1)} + b_p^{(1)}\right), \tag{8}$$

where $W_p^{(1)} \in \mathbb{R}^{M\times M}$ and $b_p^{(1)} \in \mathbb{R}^M$, and M is the dimension of the hidden state in caption generator. Then, a POS tag probability is predicted by a linear transformation with a softmax function:

$$w_t^p = \text{softmax}\left(W_p^{(2)}d_t^p + b_p^{(2)}\right), \tag{9}$$

where $W_p^{(2)} \in \mathbb{R}^{M\times n}$ and $b_p^{(2)} \in \mathbb{R}^n$, and n is the number of POS tag. After obtaining w_t^p, we represent the POS tag of the target word w_t using a semantic representation p_t:

$$p_t = E_p w_t^p, \tag{10}$$

where $E_p \in \mathbb{R}^{n\times N}$ is a POS embedding matrix and N is the dimension of the POS representation.

---

[1] The POS tags of words in ground truth are processed by Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003).

3.2.2 Attention-based Visual Switch

Visual Attention. We first use a visual attention module to select a candidate feature from l_{bef}, l_{aft}, or l_{diff} (l_{aft} - l_{bef}), which could be relevant to the target word:

$$l_{\text{dyn}}^{(t)} = \sum_i \alpha_i^{(t)} l_i, \tag{11}$$

where $i \in$ (bef, diff, aft). $\alpha_i^{(t)}$ are current visual attention weights and they are computed by an attention LSTM$_a$:

$$\begin{aligned} v &= \operatorname{ReLU}\left(W_{d_1}\left[l_{\text{bef}} ; l_{\text{diff}} ; l_{\text{aft}}\right]+b_{d_1}\right) \\ u^{(t)} &= \left[v; h_c^{(t-1)}\right] \\ h_a^{(t)} &= \operatorname{LSTM}_a\left(h_a^{(t)} \mid u^{(t)}, h_a^{(0:t-1)}\right) \\ \alpha_i^{(t)} &\sim \operatorname{Softmax}\left(W_{d_2} h_a^{(t)}+b_{d_2}\right) \end{aligned} \tag{12}$$

where W_{d_1}, b_{d_1}, W_{d_2}, and b_{d_2} are learnable parameters. $h_a^{(*)}$ and $h_c^{(*)}$ are hidden states of the attention module LSTM$_a$ and the caption generator LSTM$_c$, respectively.

Visual Switch. Then, we exploit a visual switch to decide whether to rely on visual information to predict the next word based on the predicted POS information p_t. At time step t, the visual switch β_t is defined as:

$$\begin{aligned} m_t &= \left[p_t; h_c^{t-1}; l_{dyn}^{(t)}\right], \\ \beta_t &= \sigma(W_{s2}(\operatorname{ReLU}(W_{s1} m_t))), \end{aligned} \tag{13}$$

where σ is the sigmoid function and W_{s*} are the learnable parameters. The range of β_t is [0,1] and the value of it indicates how much visual information to use when predicting the target word. Then, we apply this switch to attended visual feature $l_{\text{dyn}}^{(t)}$ to control the use of visual information:

$$L_{dyn}^{(t)} = \beta_t \odot l_{dyn}^{(t)}. \tag{14}$$

3.2.3 Caption generator

After the proper visual information is obtained, we use it and the previous word w_{t-1} (ground-truth word during training, predicted word during inference) to the caption generator LSTM$_c$ to predict a series of distributions over the next word:

$$\begin{aligned} c^{(t)} &= \left[E\left[w_{t-1}\right]; L_{\text{dyn}}^{(t)}\right], \\ h_c^{(t)} &= \operatorname{LSTM}_c\left(h_c^{(t)} \mid c^{(t)}, h_c^{(0:t-1)}\right), \\ w_t &\sim \operatorname{Softmax}\left(W_c h_c^{(t)}+b_c\right), \end{aligned} \tag{15}$$

where E is a word embedding matrix; W_c and b_c are learnable parameters.

3.3 Joint Training

We jointly train the POS predictor and the caption generator end-to-end by maximizing the likelihood of the observed POS and word sequence. For the POS predictor, given the target ground-truth POS tags $(w_1^p, \ldots, w_m^p)$, we minimize its negative log-likelihood loss:

$$L_{pos}(\theta_p) = -\sum_{t=1}^{m} \log p\left(w_t^p \mid w_{<t}^p; \theta_p\right), \tag{16}$$

where θ_p are the parameters of the POS predictor and m is the length of the POS tag.

For the caption generator, given the target ground-truth caption words $(w_1^c, \ldots, w_m^c)$, we minimize its negative log-likelihood loss:

$$L_{cap}(\theta_c) = -\sum_{t=1}^{m} \log p\left(w_t^c \mid w_{<t}^c; \theta_c\right), \tag{17}$$

where θ_c are the parameters of the caption generator and m is the length of the caption. Thus, the final loss function is optimized as follows:

$$L(\theta) = L_{pos} + L_{cap} \tag{18}$$

4 Experiments

4.1 Datasets

CLEVR-Change. This dataset (Park et al., 2019) is a large scale dataset with a set of basic geometry objects, which consists of 79,606 image pairs and 493,735 captions. The change types consist of five cases, *i.e.*, "Color", "Texture", "Add", "Drop", and ''Move". We use the official split with image pairs of 67,660 for training, 3, 976 for validation and 7,970 for testing.

Spot-the-Diff. This dataset (Jhamtani and Berg-Kirkpatrick, 2018) contains 13,192 real image pairs which are well aligned image pairs, with one or more changes between the images (but no distractors). Similar to (Park et al., 2019), we only evaluate our model in a single change setting and split it into training, validation, and test sets with a ratio of 8:1:1.

4.2 Evaluation Metrics

We use five standard metrics to evaluate the quality of generated sentences, *i.e.*, BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016). We get all the results in this paper according to the Microsoft COCO evaluation server (Chen et al., 2015).

Table 1: Ablation studies on CLEVR-Change in terms of total performance, where B-4, M, R, C, and S are short for BLEU-4, METEOR, ROUGE-L, CIDEr, and SPICE, respectively.

	Total				
Method	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
Baseline	53.1	37.3	70.6	115.6	31.2
SSRE	54.2	39.2	72.2	120.1	32.0
CSRM	53.7	38.5	71.6	118.0	32.0
SRDRL	54.8	40.1	73.2	121.0	32.6
AVS	53.2	38.5	71.3	115.7	31.6
SRDRL+AVS	**54.9**	**40.2**	**73.3**	**122.2**	**32.9**

Table 2: Ablation studies on CLEVR-Change in terms of different settings.

	Scene Change					None-scene Change				
Method	B-4	M	R	C	S	B-4	M	R	C	S
Baseline	50.9	33.0	65.3	100.9	27.7	62.0	50.0	75.9	116.1	34.7
SSRE	51.7	35.0	67.7	111.2	29.3	62.0	51.2	76.8	115.6	34.8
CSRM	51.8	34.6	67.3	106.5	29.4	61.4	49.9	75.9	115.5	34.7
SRDRL	52.0	35.8	68.9	112.3	30.3	62.1	**52.0**	**77.5**	116.3	**34.9**
AVS	50.9	34.2	66.5	103.6	28.8	60.3	50.5	76.1	113.5	34.4
SRDRL+AVS	**52.7**	**36.4**	**69.7**	**114.2**	**30.8**	**62.2**	51.3	76.9	**117.0**	**34.9**

4.3 Implementation Details

To extract image features, we use ResNet-101 (He et al., 2016) pre-trained on the Imagenet dataset (Russakovsky et al., 2015). We use features from the convolutional layer with dimensionality of 1024 × 14 × 14. The hidden size is set to 512 and the number of attention heads in SSRE is set to 4. The words are represented by trainable 300D word embedding features. POS tags are divided into 16 categories. In the training phase, on CLEVR-Change and Spot-the-Diff, we respectively set the mini-batch size as 128 and 96. We use Adam optimizer (Kingma and Ba, 2014) with the learning rate of 1×10^{-3} and 5×10^{-4}, respectively. At inference, greedy decoding strategy is used to generate target captions. Both training and inference are implemented with PyTorch (Paszke et al., 2019) on a TITAN Xp GPU.

4.4 Ablation studies

In order to figure out the contribution of each module, we carry out the following ablation studies on CLEVR-Change: (1) Baseline which is based on DDUA (Park et al., 2019); (2) SSRE which only embeds the self-semantic relations of objects into their representations; (3) CSRM which only measures the cross-semantic relations between the captured candidate difference and the original images, and the learned discriminative difference representation is used as a prior knowledge to guide the change localizer; (4) SRDRL which is the combination of (2) and (3); (5) AVS which only relies on the POS information to determine when to use visual information and which of them should be used; (6) SRDRL+AVS which is the combination of (4) and (5).

The Evaluation on Total Performance. We frist study the total performance of each block of the proposed method under the whole dataset, including scene change and none-scene change. Experimental results are shown in Table 1. We can observe that each module of the proposed method improves the total performance of the baseline. Moreover, the best performance is achieved when putting them together, which indicates each block not only plays its unique role, but also can be a supplementary role for the others. This global statistical performance validates the generalization ability of the proposed method, that is, it not only can explicitly judge whether there is a semantic change between a pair of unaligned images, but also can describe the change using an accurate sentence.

The Evaluation on Scene Change and None-scene Change. The experimental results are shown

Table 3: Comparing with state-of-the-art methods on CLEVR-Change in Total Performance. RL is short for reinforcement learning training strategies.

		Total				
Method	RL	B-4	M	R	C	S
Capt-Dual (Park et al., 2019)	×	43.5	32.7	-	108.5	23.4
DDUA (Park et al., 2019)	×	47.3	33.9	-	112.3	24.5
M-VAM (Shi et al., 2020)	×	50.3	37.0	69.7	114.9	30.5
M-VAM+RAF (Shi et al., 2020)	✓	51.3	37.8	70.4	115.8	30.7
SRDRL+AVS	×	**54.9**	**40.2**	**73.3**	**122.2**	**32.9**

Table 4: Comparing with state-of-the-art methods on CLEVR-Change in terms of two settings.

		Scene Change				None-scene Change			
Method	RL	B-4	M	C	S	B-4	M	C	S
Capt-Dual (Park et al., 2019)	×	38.5	28.5	89.8	18.2	56.3	44.0	108.9	28.7
DDUA (Park et al., 2019)	×	42.9	29.7	94.6	19.9	59.8	45.2	110.8	29.1
M-VAM+RAF (Shi et al., 2020)	✓	-	-	-	-	-	**66.4**	**122.6**	33.4
SRDRL+AVS	×	**52.7**	**36.4**	**114.2**	**30.8**	**62.2**	51.3	117.0	**34.9**

in Table 2, in terms of scene change, we can observe that 1) SSRE, CSRM and AVS all achieve improvements over the baseline; 2) compared with SSRE, the improvement is relatively small when respectively using CSRM and AVS; 3) better performances are achieved when using two kinds of combinations (SRDRL and SRDRL+AVS). These indicate 1) the effectiveness of our proposed SRDRL and its single block, as well as the AVS; 2) the priority of this task is to capture the semantic difference in the image pair. The reason is that only if the semantic difference is captured sufficiently, can the following specific change localization and caption generation do well on itself part.

Besides, we can observe that although each single block can improve the baseline in the case of scene change, but they are worse than the baseline in one or more metrics in the case of none-scene change. Our conjecture is that the robustness of single block is relatively weak, so it would sometimes misidentify the illumination or viewpoint change as the actual semantic change. When observing the performance of two kinds of combinations (SRDRL and SRDRL+AVS), both of them improve the baseline in all metrics, which indicates the robustness of our overall model is strong.

4.5 Performance Comparison

4.5.1 Results on CLEVR-Change

In this dataset, we compare with four state-of-the-art methods, Capt-Dual (Park et al., 2019), DUDA (Park et al., 2019), M-VAM (Shi et al., 2020) and M-VAM+RAF (Shi et al., 2020), in four dimensions: 1) the total performance of scene change and none-scene change; 2) only scene change; 3) only none-scene change; 4) specific type of scene change. The comparison results are shown in Table 3, Table 4, and Table 5, respectively.

From Table 3, in terms of total performance, we can clearly observe that our method achieves significant improvements over them in all evaluation metrics, in particular with an increase of 34.3% and 7.2% in SPICE, respectively. From Table 4, under two kinds of settings, we can observe that our method outperforms DDUA with a large margin. Furthermore, since the M-VAM+RAF did not report the results on scene change, we only compare with them in the setting of none-change. We can observe that it outperforms us in METEOR and CIDEr. This superiority could derive from the reinforcement learning strategy. However, this strategy will remarkably increase training time and computation complexity. Moreover, as reported in Table 3, our total performance is much better than them, which is evaluated under the both scene change and none-scene change. Hence, compared to them, our method is more robust due to the discriminative difference representation learning.

Table 5 is the detailed breakdown of the evaluation based on five change types: "Color" (C), "Texture" (T), "Add" (A), "Drop" (D), and "Move" (M). Specifically, compare to all SOTA methods,

Table 5: A Detailed breakdown of Change Captioning evaluation on CLEVR-Change by different change types: "Color" (C), "Texture" (T), "Add" (A), "Drop" (D), and "Move" (M).

Method	RL	Metrics	C	T	A	D	M
Capt-Dual (Park et al., 2019)	×	CIDEr	115.8	82.7	85.7	103.0	52.6
DDUA (Park et al., 2019)	×	CIDEr	120.4	86.7	108.3	103.4	56.4
M-VAM+RAF (Shi et al., 2020)	✓	CIDEr	122.1	98.7	**126.3**	115.8	82.0
SRDRL+AVS	×	CIDEr	**136.1**	**122.7**	121.0	**126.0**	78.9
Capt-Dual (Park et al., 2019)	×	METEOR	32.1	26.7	29.5	31.7	22.4
DDUA (Park et al., 2019)	×	METEOR	32.8	27.3	33.4	31.4	23.5
M-VAM+RAF (Shi et al., 2020)	✓	METEOR	35.8	32.3	37.8	36.2	27.9
SRDRL+AVS	×	METEOR	**39.0**	**35.6**	**38.9**	**38.0**	**30.1**
Capt-Dual (Park et al., 2019)	×	SPICE	19.8	17.6	16.9	21.9	14.7
DDUA (Park et al., 2019)	×	SPICE	21.2	18.3	22.4	22.2	15.4
M-VAM+RAF (Shi et al., 2020)	✓	SPICE	28.0	26.7	30.8	**32.3**	22.5
SRDRL+AVS	×	SPICE	**32.4**	**30.9**	**33.0**	**32.4**	**25.4**

Table 6: Comparing with state-of-the-art methods on Spot-the-Diff.

Method	RL	M	R	C	S
DDLA	×	12.0	28.6	32.8	-
DDUA	×	11.8	29.1	32.5	-
SDCM	×	12.7	29.7	36.3	-
FCC	×	12.9	29.9	36.8	-
static rel-att	×	13.0	28.3	34.0	-
dynamic rel-att	×	12.2	31.4	35.3	-
M-VAM	×	12.4	31.3	38.1	14.0
M-VAM+RAF	✓	12.9	**33.2**	**42.5**	17.1
SRDRL+AVS	×	**13.0**	31.0	35.3	**18.0**

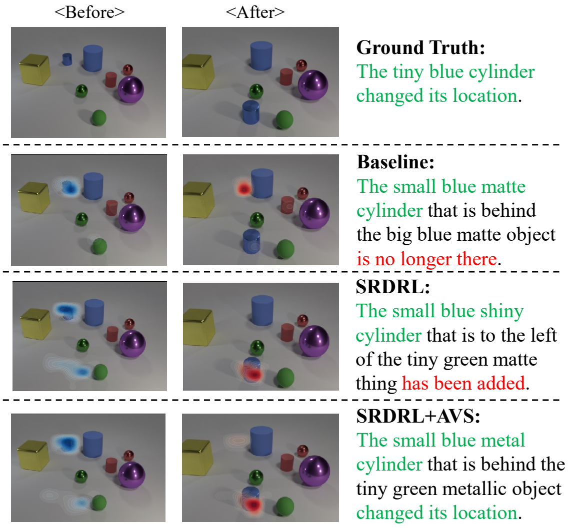

Figure 3: A comparative example about "Move" case from the test set of CLEVR-Change, which involves the caption generated by the baseline, SRDRL, and SRDRL+AVS. We visualize the localization results on "before" (blue) and "after" (red).

our method significantly raises the CIDEr scores in "Color" and "Texture" types, which indicates our method can better distinguish the attribute change of objects from an illumination change. Besides, for the number or position change of objects ("Add", "Drop", and "Move"), our method all outperforms them in most of metrics. Especially for SPICE, compared to them, our method has 64.9% and 12.9% improvements for "Move" case, respectively, which also shows our method can better localize the object movement from the viewpoint change. In particular, the most challenging change types are "Texture" and "Move" in this dataset, because they are most often confused with the illumination or viewpoint changes (Park et al., 2019). The relative experiments show that our method is more robust than SOTAs, and this benefits from the fact that the CSRM block helps attend to the actually semantic change by measuring the cross-semantic relations of the image pair and their difference.

4.5.2 Results on Spot-the-Diff

To validate the generalization ability of the proposed method, we conduct the experiments on a recent published Spot-the-Diff dataset, where the image pairs are mostly well aligned and their is no viewpoint change. We compare with eight SOTA methods and most of them cannot consider handling viewpoint changes: DDLA (Jhamtani and Berg-Kirkpatrick, 2018), DDUA (Park et al., 2019), SDCM (Oluwasanmi et al., 2019a), FCC (Oluwasanmi et al., 2019b), static rel-att / dynamic rel-att (Tan et al., 2019), and M-VAM / M-VAM+RAF (Shi et al., 2020).

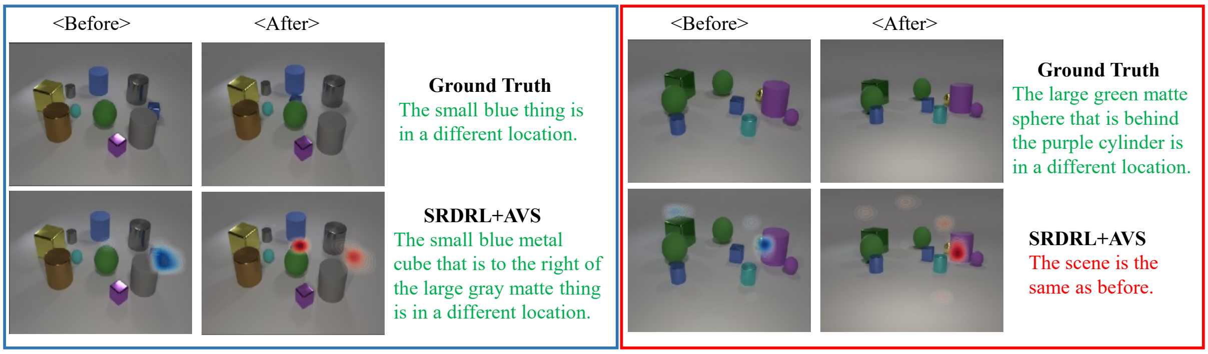

Figure 4: Qualitative examples of SRDRL+AVS. The left is a successful case that SRDRL+AVS localizes the accurate changed object and generates a correct sentence to describe the change. The right is a failure case that a slight movement of the object is not detected.

The results are reported in Table 6. We can observe that our method achieves the best performance in terms of METEOR and SPICE. Especially for SPICE which is recently designed for evaluating the image captioning task, our method achives 28.6% and 5.3% improvements over the current SOTA method M-VAM and M-VAM+RAF. Hence, compared to the above methods, the generated captions by our method are more in line with standards of human caption evaluation. This superiority results from that the SSRE block can capture the relation-embedded feature difference so as to better explore those tiny changed objects.

4.6 Qualitative Analysis

Figure 3 shows a comparative example about "Move" from the CLEVR-Change dataset, which includes the change captions generated by humans, baseline, SRDRL, and SRDRL+AVS. We also visualize the results of change detection. The baseline is implemented based on DDUA (Park et al., 2019). We can clearly observe that it localizes a wrong region on the "after" and thus misidentifies "Move" as "Drop". By contrast, both proposed methods (SRDRL and SRDRL+AVS) can accurately localize the moved object on both "before" and "after" images, which validates the effectiveness of the proposed SRDRL. Moreover, it is interesting to note that, for the proposed methods, although the results of change localization are accurate, only using SRDRL generates a wrong caption, which indicates the POS tags of target words indeed guide and regularize the change caption generation.

Figure 4 illustrates two examples with viewpoint changes on CLEVR-Change dataset. The left example is a success in which SRDRL+AVA can distinguish the small blue changed cube from the irrelevant viewpoint change. This benefits from that SRDRL can learn discriminative difference representation and overcome viewpoint changes. The right example shows a failure, where SRDRL+AVA judges there is no difference. Our conjecture is that the movement of this sphere is very slight and thus confused with the viewpoint change. Hence, we will improve our method to learn more fine-grained difference representation in the future work.

5 Conclusion

In this paper, we propose a semantic relation-aware difference representation learning network (SRDRL) and attention-based visual switch (AVS) to address change captioning in the presence of distractors, where SRDRL can explicitly learn the difference representation in the image pair and AVS can aid the caption generator to convey the localized change in a logically and grammatically accurate sentence. Extensive experiments conducted on both CLEVR-Change and Spot-the-Diff datasets show that the proposed method achieves state-of-the-art results.

Acknowledgements

The work was supported by National Natural Science Foundation of China (Grant Nos. 61761026, 61972186, 61732005 and 61762056), National Key Research and Development Plan (Grant Nos. 2019QY1802, 2019QY1801, and 2019QY1800), Yunnan high-tech industry development project (Grant No. 201606), Yunnan provincial major science and technology special plan projects (Grant No. 202002AD080001-5), and Yunnan Basic Research Project (Grant Nos. 202001AS070014, 2018FB104).

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *ECCV*, pages 382–398.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL*, pages 65–72.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Jincan Deng, Liang Li, Beichen Zhang, Shuhui Wang, Zhengjun Zha, and Qingming Huang. 2021. Syntax-guided hierarchical attention network for video captioning. *IEEE Transactions on Circuits and Systems for Video Technology*.

L. Gueguen and R. Hamid. 2015. Large-scale damage detection using satellite imagery. In *CVPR*, pages 1321–1328.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Qingbao Huang, Jielong Wei, Yi Cai, Changmeng Zheng, Junying Chen, Ho-fung Leung, and Qing Li. 2020. Aligned dual channel graph convolutional network for visual question answering. In *ACL*, pages 7166–7176.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2018. Learning to describe differences between pairs of similar images. In *EMNLP*, pages 4024–4034.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liang Li, Shijie Yang, Li Su, Shuhui Wang, Chenggang Yan, Zheng-jun Zha, and Qingming Huang. 2020. Diverter-guider recurrent network for diverse poems generation from image. In *ACM MM*, pages 3875–3883.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xuejing Liu, Liang Li, Shuhui Wang, Zheng-Jun Zha, Dechao Meng, and Qingming Huang. 2019. Adaptive reconstruction network for weakly supervised referring expression grounding. In *ICCV*, pages 2611–2620.

Zhenhuan Liu, Jincan Deng, Liang Li, Shaofei Cai, Qianqian Xu, Shuhui Wang, and Qingming Huang. 2020. Ir-gan: Image manipulation with linguistic instruction by increment reasoning. In *ACM MM*, pages 322–330.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, pages 3242–3250.

Ariyo Oluwasanmi, Muhammad Umar Aftab, Eatedal Alabdulkreem, Bulbula Kumeda, Edward Y Baagyere, and Zhiquang Qin. 2019a. Captionnet: Automatic end-to-end siamese difference captioning model with attention. *IEEE Access*, 7:106773–106783.

Ariyo Oluwasanmi, Enoch Frimpong, Muhammad Umar Aftab, Edward Y Baagyere, Zhiguang Qin, and Kifayat Ullah. 2019b. Fully convolutional captionnet: Siamese difference captioning attention model. *IEEE Access*, 7:175929–175939.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Dong Huk Park, Trevor Darrell, and Anna Rohrbach. 2019. Robust change captioning. In *ICCV*, pages 4623–4632.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035.

J. Patriarche and B. Erickson. 2004. A review of the automated detection of change in serial imaging studies of the brain. *Journal of Digital Imaging*, 17:158–174.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *In IJCV,*, pages 211–252.

Ken Sakurada and Takayuki Okatani. 2015. Change detection from a street image pair using cnn features and superpixel segmentation. In *BMVC*.

Xiangxi Shi, Xu Yang, Jiuxiang Gu, Shafiq R. Joty, and Jianfei Cai. 2020. Finding it at another side: A viewpoint-adapted matching encode for change captioning. In *ECCV*, pages 574–590.

Jingkuan Song, Lianli Gao, Zhao Guo, Wu Liu, Dongxiang Zhang, and Heng Tao Shen. 2017. Hierarchical lstm with adjusted temporal attention for video captioning. In *IJCAI*, pages 2737–2743.

Hao Tan, Franck Dernoncourt, Zhe Lin, Trung Bui, and Mohit Bansal. 2019. Expressing visual relationships via language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1873–1883, Florence, Italy. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.

Yunbin Tu, Xishan Zhang, Bingtao Liu, and Chenggang Yan. 2017. Video description with spatial-temporal attention. In *ACM MM*, pages 1014–1022.

Yunbin Tu, Chang Zhou, Junjun Guo, Shengxiang Gao, and Zhengtao Yu. 2020. Enhancing the alignment between target words and corresponding frames for video captioning. *Pattern Recognition*, page 107702.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Bairui Wang, Lin Ma, Wei Zhang, Wenhao Jiang, Jingwen Wang, and Wei Liu. 2019. Controllable video captioning with pos sequence guidance based on gated fusion network. In *ICCV*, pages 2641–2650.

Aming Wu, Linchao Zhu, Yahong Han, and Yi Yang. 2019. Connective cognition network for directional visual commonsense reasoning. In *NeurIPS*, pages 5669–5679.

Chenggang Yan, Biao Gong, Yuxuan Wei, and Yue Gao. 2020a. Deep multi-view enhancement hashing for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chenggang Yan, Yiming Hao, Liang Li, Jian Yin, Anan Liu, Zhendong Mao, Zhenyu Chen, and Xingyu Gao. 2021. Task-adaptive attention for image captioning. *IEEE Transactions on Circuits and Systems for Video Technology*.

Chenggang Yan, Liang Li, Chunjie Zhang, Bingtao Liu, Yongdong Zhang, and Qionghai Dai. 2019. Cross-modality bridging and knowledge transferring for image understanding. *IEEE Transactions on Multimedia*, 21(10):2675–2685.

Chenggang Yan, Yunbin Tu, Xingzheng Wang, Yongbing Zhang, Xinhong Hao, Yongdong Zhang, and Qionghai Dai. 2020b. STAT: spatial-temporal attention mechanism for video captioning. *IEEE Trans. Multimedia*, 22(1):229–241.

Yongjing Yin, Fandong Meng, Jinsong Su, Chulun Zhou, Zhengyuan Yang, Jie Zhou, and Jiebo Luo. 2020. A novel graph-based multi-modal fusion encoder for neural machine translation. In *ACL*, pages 3025–3035.

Yongjing Yin, Jinsong Su, Huating Wen, Jiali Zeng, Yang Liu, and Yidong Chen. 2019. Pos tag-enhanced coarse-to-fine attention for neural machine translation. *ACM transactions on Asian language information processing*, 18(4):46.1–46.14.

Xishan Zhang, Ke Gao, Yongdong Zhang, Dongming Zhang, Jintao Li, and Qi Tian. 2017. Task-driven dynamic fusion: Reducing ambiguity in video description. In *CVPR*, pages 3713–3721.

The Authors Matter:
Understanding and Mitigating Implicit Bias in Deep Text Classification

Haochen Liu[1], **Wei Jin**[1], **Hamid Karimi**[1], **Zitao Liu**[2]*, **Jiliang Tang**[1]
[1] *Michigan State University, East Lansing, MI, USA*
[2] *TAL Education Group, Beijing, China*
{liuhaoc1,jinwei2,karimiha}@msu.edu; liuzitao@100tal.com; tangjili@msu.edu

Abstract

It is evident that deep text classification models trained on human data could be biased. In particular, they produce biased outcomes for texts that explicitly include identity terms of certain demographic groups. We refer to this type of bias as explicit bias, which has been extensively studied. However, deep text classification models can also produce biased outcomes for texts written by authors of certain demographic groups. We refer to such bias as implicit bias, of which we still have a rather limited understanding. In this paper, we first demonstrate that implicit bias exists in different text classification tasks for different demographic groups. Then, we build a learning-based interpretation method to deepen our knowledge of implicit bias. Specifically, we verify that classifiers learn to make predictions based on language features that are related to the demographic attributes of the authors. Next, we propose a framework **Debiased-TC** to train deep text classifiers to make predictions on the right features and consequently mitigate implicit bias. We conduct extensive experiments on three real-world datasets. The results show that the text classification models trained under our proposed framework outperform traditional models significantly in terms of fairness, and also slightly in terms of classification performance.

1 Introduction

Many recent studies have suggested that machine learning algorithms can learn social prejudices from data produced by humans, and thereby show systemic bias in performance towards specific demographic groups or individuals (Mehrabi et al., 2019; Blodgett et al., 2020; Shah et al., 2020). As one machine learning application, text classification has been proven to be discriminatory towards certain groups of people (Dixon et al., 2018; Borkan et al., 2019). Text classification applications such as sentiment analysis and hate speech detection are common and widely used in our daily lives. If a biased hate speech detection model is deployed by a social media service provider to filter users' comments, the comments related to different demographic groups can have uneven chances to be recognized and removed. Such a case will cause unfairness and bring in negative experiences to users. Thus, it is highly desired to mitigate the bias in text classification.

The majority of existing studies on bias and fairness in text classification have mainly focused on the bias towards the individuals mentioned in the text content. For example, in (Dixon et al., 2018; Park et al., 2018; Zhang et al., 2020), it is investigated how text classification models perform unfairly on texts containing demographic identity terms such as "gay" and "muslim". In such scenarios, the demographic attributes of the individuals subject to bias explicitly exist in the text. In this work, we refer to this kind of bias as **explicit bias**. Bias in texts, however, can be reflected more subtly and insidiously. While a text may not contain any reference to a specific group or individual, the content can somehow be revealing of the demographic information of the author. As shown in (Coulmas, 2013; Preoţiuc-Pietro and Ungar, 2018), the language style (e.g., wordings and tone) of a text can be highly correlated with its author's demographic attributes (e.g., age, gender, and race). We find that a text classifier can learn to associate the content with demographic information and consequently make unfair decisions towards certain groups. We refer to such bias as **implicit bias**. Table 1 demonstrates an example of implicit bias. There are two short texts where the first text is written by a white American and the second one is written by an African American. The task is to predict the sentiment of a text by a convolutional

---

[*] The corresponding author: Zitao Liu

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 74–85
August 1–6, 2021.

Table 1: An illustrative example on the implicit bias of a CNN text classification model.

Author	Text	Label	Prediction
White American	Can't wait to visit your new home. Yes, I going to be a great guest!	positive	positive
African American	Can't wait to visit your new home. Yup , I goin to be a great guest!	positive	negative

neural network (CNN) model. Words with a red background indicate those with the salient predictive capability by the model where the darker the color, the more salient the words. The words "yup" and "goin" in the second text are commonly used by African Americans (Liu et al., 2020a) and are irrelevant to the sentiment. However, the CNN model has hinted at them and consequently has predicted a positive text to be negative.

In this work, we aim to understand and mitigate implicit bias in deep text classification models. One key source of bias is the imbalance of training data (Dixon et al., 2018; Park et al., 2018). Thus, existing debiasing methods mainly focus on balancing the training data, such as adding new training data (Dixon et al., 2018) and augmenting data based on identity-term swap (Park et al., 2018). However, these methods cannot be directly applied to mitigate implicit bias. Obtaining new texts from authors of various demographic groups is very expensive. It requires heavy human labor. Meanwhile, given that there is no explicit demographic information in texts, identity-term swap data augmentation is not applicable. Thus, we propose to enhance deep text classification models to mitigate implicit bias in the training process. To achieve this goal, we face tremendous challenges. First, to mitigate the implicit bias, we have to understand how deep models behave. For example, how they correlate implicit features in text with demographic attributes and how the models make biased predictions. Second, we need to design new mechanisms to take advantage of our understandings to mitigate the implicit bias in deep text classifiers.

To address the above challenges, in this paper, we first propose an interpretation method, which sheds light on the formation mechanism of implicit bias in deep text classification models. We show that the implicit bias is caused by the fact that the models make predictions based on incorrect language features in texts. Second, based on this finding, we propose a novel framework **Debiased-TC** (Debiased Text Classification) to mitigate the implicit bias of deep text classifiers. More specifically, we equip the deep classifiers with an additional saliency selection layer that first determines the correct language features which the model should base on to make predictions. We also propose an optimization method to train the classifiers with the saliency selection layer. Note that both our proposed interpretation method and the learning framework are model-agnostic, which means that they can be applied to any deep text classifier. We evaluate the framework with two popular deep text classification models across various text classification tasks on three public datasets. The experimental results demonstrate that our method significantly mitigates the implicit bias in the classification models while maintaining or even improving their prediction performance.

2 Preliminary Study

In this section, we perform a preliminary study to validate the existence of implicit bias in deep text classification models. We first introduce the data and text classification tasks, and then present the empirical results.

2.1 Data and Tasks

In the preliminary study, we investigate different text classification tasks and various demographic groups to validate the implicit bias. We use three datasets, including the DIAL and PAN16 datasets processed by (Elazar and Goldberg, 2018) and the Multilingual Twitter Corpus (MTC) introduced in (Huang et al., 2020).

The DIAL dataset contains dialectal texts collected from Twitter. Each tweet's text is associated with the *race* of the author as the demographic attribute, denoted as "white" or "black", respectively. This dataset is annotated for two classification tasks: sentiment analysis and mention detection. The sentiment analysis task aims to categorize a text as "happy" or "sad". The mention detection task tries to determine whether a tweet mentions another user, which can also be viewed as distinguishing conversational tweets from non-conversational ones.

The PAN16 dataset consists of tweets. For each tweet, *age* and *gender* of its author have been manually labelled. The demographic attribute age has two categories of "18-34" and "≥ 35", and gender has “male” and “female”. Also, this dataset is annotated for the mention detection task as described above.

The MTC dataset contains multilingual tweets for the hate speech detection task. Each tweet is annotated as “hate speech” or “non hate speech” and associated with four author’s demographic attributes: race, gender, age, and country. We only use the English corpus with the attribute *race*. In this dataset, the attribute race has two categories, i.e., “white” and “nonwhite”.

More information about these three datasets, including their statistical information, annotation process, and the links to downloadable versions of the data can be found in Appendix A.

2.2 Empirical study

In this subsection, we aim to empirically study if text classification models make the predictions dependent on the demographic attributes of the authors of the texts. The explicit bias in text classification tasks stems from the imbalance of training data (Dixon et al., 2018; Park et al., 2018). For example, when there are more negative examples from one group in the training data, the model learns to correlate that group with the negative label, which results in bias. Inspired by this observation, to validate the existence of implicit bias, we investigate if the imbalance of training data in terms of demographic attributes of the authors can lead to biased predictions. To answer this question, we consider the following setting: (1) the training data has an equal number of positive and negative examples; and (2) positive and negative examples in the training data are imbalanced among different groups of the authors according to their demographic attributes. Intuitively, if the predictions are independent of the demographic attributes of authors, the model should still perform similarly for different groups.

For each task and demographic attribute of authors, we consider two labels (i.e., positive and negative) and two demographic groups (i.e., Group I and Group II). For each dataset, we follow the aforementioned setting to build a training set. We make the training set overall balanced in terms of the labels and demographic groups. That is, we set the overall ratio of positive and negative examples as 1:1, and the overall ratio of examples from Group I and Group II as 1:1 as well. Meanwhile, we make the data in each group imbalanced. In particular, for Group I, we set the ratio of its positive and negative examples to 4:1, while the ratio is automatically set to 1:4 for Group II. We name the proportion of positive and negative samples in Group I as the “balance rate”. We train a CNN text classifier as a representative model on the training set and evaluate it on the test set. We use the false positive/negative rates (Dixon et al., 2018) and the demographic parity rate (a.k.a., positive outcome rate, the probability of the model predicting a positive outcome for one group) (Dwork et al., 2012; Kusner et al., 2017) to evaluate the fairness of the classification models.

The results are shown in Table 2. For the demographic attribute race, Group I/Group II stands for white/black in the DIAL dataset, and white/nonwhite in the MTC dataset. For gender and age, Group I/Group II stands for male/female and age ranges (18-34)/($\geq$35), respectively. From the table, we observe that in terms of different tasks and demographic attributes of authors, the model shows significant bias with the same pattern. For all cases, the demographic group with more positive examples (Group I) always gets a higher false positive rate, a lower false negative rate, and a higher demographic parity rate than the other group. This demonstrates that imbalanced data can cause implicit bias, and the predictions are not independent of the demographic attributes of authors. Since the text itself doesn’t explicitly contain any demographic information, the model could learn to recognize the demographic attributes of authors based on implicit features such as language styles and associate them with a biased outcome. Next, we will understand one formation of implicit bias and then propose Debiased-TC to mitigate it.

3 Understanding Implicit Bias

In this section, we aim to understand the possible underlying formation mechanism of implicit bias. Our intuition is – when a training set for sentiment analysis has more positive examples from white authors and more negative examples from black authors, a classification model trained on such a dataset may learn a “shortcut” (Mahabadi et al., 2020) to indiscriminately associates the language style features of white people with the positive sentiment and those of black people with the negative sentiment. In other words, the model does not use the correct language features (e.g., emotional words) to make the prediction. Thus, we attempt

Table 2: Preliminary study.

Dataset	Task	Demo	False Positive (%)		False Negative (%)		Demographic Parity (%)	
			Group I	Group II	Group I	Group II	Group I	Group II
DIAL	Sentiment	Race	46.97	23.38	21.29	62.75	62.84	30.32
	Mention	Race	48.72	15.99	17.32	34.90	65.70	40.55
PAN16	Mention	Gender	23.90	12.30	13.06	23.01	55.42	44.64
	Mention	Age	24.91	9.88	16.48	26.43	54.22	41.72
MTC	Hate Speech	Race	80.33	1.77	12.13	49.35	84.10	26.21

to examine the following hypothesis: *A deep text classification model presents implicit bias since it makes predictions based on language features that should be irrelevant to the classification task but are correlated with a certain demographic group of authors.* To verify this hypothesis, we first propose an interpretation method to detect the salient words a text classification model relies on to make the prediction. The interpretation model enables us to check the overlapping between the salient words and the words related to the authors' demographic attributes. Consequently, it allows us to understand the relationship between such overlapping and the model's implicit bias.

3.1 An Interpretation Method

We follow the idea of the learning-based interpretation method L2X (Chen et al., 2018) to train an explainer to interpret a given model. The reasons for choosing L2X are – 1) as a learning-based explainer, it learns to globally explain the behavior of a model, instead of explaining a single instance at one time; and 2) the explainer has the potential to be integrated into our debiasing framework to mitigate implicit bias in an end-to-end manner, which will be introduced in Section 4.

A binary text classification model $\mathcal{M} : X \rightarrow Y$ maps an input text $X = (x_1, x_2, \ldots, x_n)$ to a label $Y \in \{0, 1\}$. For a certain model $\mathcal{M}$, we seek to specify the contribution of each word in X for $\mathcal{M}$ to make the prediction Y. The contributions can be denoted as a saliency distribution $S = (s_1, s_2, \ldots, s_n)$, where s_i is the saliency score of the word x_i, and $\sum_{i=1}^{n} s_i = 1$. Given a model $\mathcal{M}$, we train an explainer $\mathcal{E}^{\mathcal{M}} : X \rightarrow S$ to estimate the saliency distribution S of an input text X.

The explainer is trained by maximizing $I(X_S, Y)$, the mutual information (Cover, 1999) between the response variable Y and the selected feature X_S of X under saliency distribution S. The selected feature $X_S = X \odot S = (s_1 \cdot x_1, s_2 \cdot x_2, \ldots, s_n \cdot x_n)$ [1] is calculated as the element-wise product between X and S. In our implementation, we parametrize the explainer by a bi-directional recurrent neural network (RNN) followed by a linear layer and a Softmax layer. More details about the optimization of the explainer can be found in Appendix B.

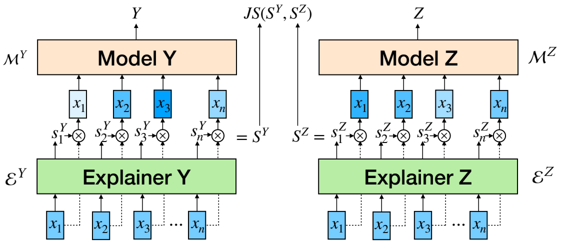

Figure 1: An illustration of the bias interpretation model.

3.2 Saliency Correlation Measurement

In this work, we assume that the text classification task is totally independent of the demographic attribute of the author of the text. In other words, language features that reflect the author's demographic information should not be taken as evidence for the main task. Thus, we propose to understand the implicit bias of a deep text classification model by examining the overlapping between salient words for the main task and the words correlated with the demographic attribute.

With the interpretation model, we can estimate the saliency distributions of the input words for the classification task and the demographic attribute prediction task, respectively, and then check their overlapping. As shown in Figure 1, we train two models $\mathcal{M}^Y$ and $\mathcal{M}^Z$ with the same architecture for the former and the latter tasks, respectively. Then, two corresponding explainers $\mathcal{E}^Y$ and $\mathcal{E}^Z$ are trained for them. Thus, given an input text X, two explainers can estimate the saliency distributions S^Y and S^Z on two tasks, respectively. We use the Jensen-Shannon (JS) divergence $JS(S^Y || S^Z)$ to measure the overlap between language features that

---

[1] Without confusion, we use x_i to denote both a word and its word embedding vector.

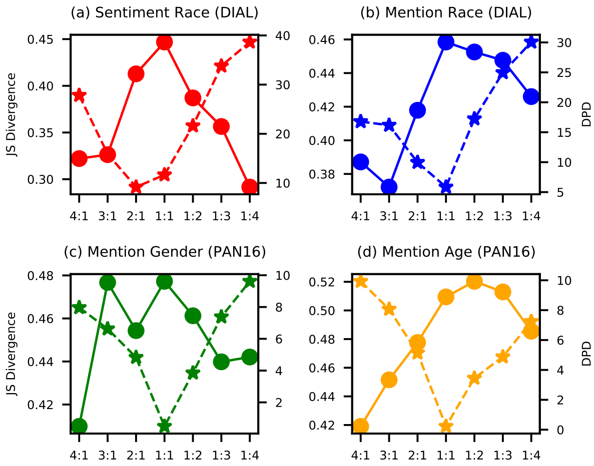

Figure 2: The average JS divergence (solid lines) and DPD (dash lines) vs. the balance rate. The x-axis indicates the balance rate of the training set. The y-axis on the left hand indicates the average JS divergence, and the y-axis on the right hand is the DPD.

these two tasks relying on to make the predictions on Y and Z.

3.3 Empirical Analysis

In this subsection, we present the experiments to verify our hypothesis on the formulation of implicit bias. Following the experimental settings in Section 2.2, we vary the "balance rate" of the training data and then observe how the saliency correlation changes. We use CNN text classifiers (see Appendix C for details) for both $\mathcal{M}^Y$ and $\mathcal{M}^Z$. In Figure 2, we show how the average JS divergence and the demographic parity difference (DPD) vary with the changes of the balance rate. DPD is the absolute value of the difference between the demographic parity rates for the two groups. We only report the results for DIAL and PAN16 datasets and DPD as the fairness metric since we achieved similar results for other settings. For each task and each demographic attribute, the DPD is small when the training data are balanced and becomes large when the data are imbalanced. However, the JS divergence is large for balanced data while small for imbalanced data. A larger DPD indicates stronger implicit bias and a smaller JS divergence stands for a stronger overlap between the saliency distributions for the two tasks. Thus, these observations suggest that when the training data are imbalanced, the text classifiers tend to use language features related to the demographic attribute of authors to make the prediction.

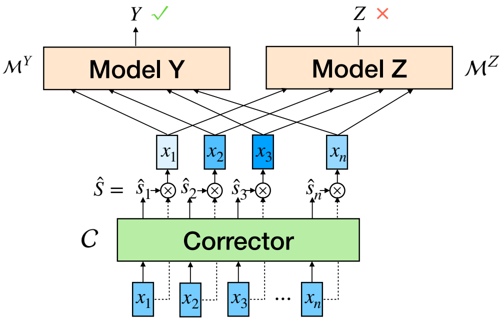

Figure 3: An illustration of the bias mitigation model.

4 The Bias Mitigation Framework

In the previous section, we showed that a model with implicit bias tends to utilize features related to the demographic attribute of authors to make the prediction, especially when training data is imbalanced in terms of the demographic attribute of authors. One potential solution is to balance the training data by augmenting more examples from underrepresented groups. However, collecting new data from authors of different demographics is expensive. Thus, to mitigate the implicit bias, we propose a novel framework **Debiased-TC**. Our proposed approach can mitigate implicit bias by automatically correcting their selection of input features. In this section, we will introduce the proposed framework with the corresponding optimization method.

4.1 Debiased Text Classification Model

An illustration of Debiased-TC is shown in Figure 3. Similar to the explainer in the interpretation model, we equip the base model $\mathcal{M}^Y$ with a corrector layer $\mathcal{C}$ after the input layer. The corrector $\mathcal{C} : X \rightarrow S$ learns to correct the model's feature selection. It first maps an input text $X = (x_1, x_2, \ldots, x_n)$ to a saliency distribution $S = (s_1, s_2, \ldots, s_n)$, which is expected to give high scores to words related to the main tasks and low scores to words related to demographic attributes of authors. Then, it assigns weights to the input features with the saliency scores by calculating $X_S = X \odot S$, which is fed into the classification model $\mathcal{M}^Y$ for prediction.

To train a corrector to achieve the expected goal, we adopt the idea of adversarial training. More specifically, in addition to the main classifier $\mathcal{M}^Y$, we introduce an adversarial classifier $\mathcal{M}^Z$, which takes X_S as the input and predicts the demographic attribute Z. During the adversarial training, the corrector attempts to help $\mathcal{M}^Y$ make correct predictions while preventing $\mathcal{M}^Z$ from predicting demographic attributes. To make this feasible, we use the gradient reversal technique (Ganin and Lempitsky, 2015), where we add a gradient-reversal layer between the weighted inputs X_S and the adversarial classifier $\mathcal{M}^Z$. The gradient-reversal layer has no effect on its downstream components (i.e., the adversarial classifier $\mathcal{M}^Z$). However, during backpropagation, the gradients that pass down through this layer to its upstream components (i.e., the corrector $\mathcal{C}$) are getting reversed. As a result, the corrector $\mathcal{C}$ receives opposite gradients from $\mathcal{M}^Z$. The outputs of the $\mathcal{M}^Y$ and $\mathcal{M}^Z$ are used as signals to train the corrector such that it can upweight the words correlated with the main task label Y and downweight the words correlated with the demographic attribute Z. We set the adversarial classifier $\mathcal{M}^Z$ with the same architecture as the main classifier $\mathcal{M}^Y$. The corrector $\mathcal{C}$ has the same architecture as the explainer introduced in Section 3.

4.2 An Optimization Method for Debiased-TC

In this subsection, we discuss the optimization method for the proposed framework. We denote the parameters of $\mathcal{M}^Y$, $\mathcal{M}^Z$ and $\mathcal{C}$ as $\mathbf{W}^Y$, $\mathbf{W}^Z$ and $\mathbf{\Theta}$, respectively. The optimization task is to jointly optimize the parameters of the classifiers, i.e., $\mathbf{W}^Y$ and $\mathbf{W}^Z$, and the parameters of the corrector, i.e., $\mathbf{\Theta}$. We can view the optimization as an architecture search problem. Since our debiasing framework is end-to-end and differentiable, we develop an optimization method for our framework based on the differentiable architecture search (DARTS) technique (Liu et al., 2018). We update $\mathcal{M}^Y$, $\mathcal{M}^Z$ by optimizing the training losses L^Y_{train} and L^Z_{train} on the training set and update $\mathbf{\Theta}$ by optimizing the validation loss L_{val} on the validation set through gradient descent. We denote the cross-entropy losses for $\mathcal{M}^Y$ and $\mathcal{M}^Z$ as L^Y and L^Z, respectively. L^Y_{train} and L^Z_{train} indicate the cross-entropy losses L^Y and L^Z on the training set. L_{val} denotes the combined loss of the two cross-entropy losses $L = L^Y + L^Z$ on the validation set.

The goal of optimizing the corrector is to find optimal parameters $\mathbf{\Theta}^*$ that minimizes the validation loss $L_{val}(\mathbf{W}^{Y*}, \mathbf{W}^{Z*}, \mathbf{\Theta})$, where the optimal parameters $\mathbf{W}^{Y*}$ and $\mathbf{W}^{Z*}$ are obtained by minimizing the training losses as follows.

$$\mathbf{W}^{Y*} = \arg\min_{\mathbf{W}^Y} L^Y_{train}(\mathbf{W}^Y, \mathbf{\Theta}^*)$$
$$\mathbf{W}^{Z*} = \arg\min_{\mathbf{W}^Z} L^Z_{train}(\mathbf{W}^Z, \mathbf{\Theta}^*)$$

The above goal forms a bi-level optimization problem (Maclaurin et al., 2015; Pham et al., 2018), where $\mathbf{\Theta}$ is the upper-level variable and $\mathbf{W}^Y$ and $\mathbf{W}^Z$ are the lower-level variables:

$$\begin{aligned}&\min_{\mathbf{\Theta}} L_{val}\big(\mathbf{W}^{Y*}(\mathbf{\Theta}), \mathbf{W}^{Z*}(\mathbf{\Theta}), \mathbf{\Theta}\big)\\ &s.t.\ \mathbf{W}^{Y*}(\mathbf{\Theta}) = \arg\min_{\mathbf{W}^Y} L^Y_{train}(\mathbf{W}^Y, \mathbf{\Theta}^*)\\ &\qquad \mathbf{W}^{Z*}(\mathbf{\Theta}) = \arg\min_{\mathbf{W}^Z} L^Z_{train}(\mathbf{W}^Z, \mathbf{\Theta}^*)\end{aligned}$$

Optimizing $\mathbf{\Theta}$ is time-consuming due to the expensive inner optimization of $\mathbf{W}^Y$ and $\mathbf{W}^Z$. Therefore, we leverage the approximation scheme as DARTS:

$$\begin{aligned}&\nabla_{\mathbf{\Theta}} L_{val}\big(\mathbf{W}^{Y*}(\mathbf{\Theta}), \mathbf{W}^{Z*}(\mathbf{\Theta}), \mathbf{\Theta}\big)\\ \approx\ &\nabla_{\mathbf{\Theta}} L_{val}\big(\mathbf{W}^Y - \xi\nabla_{\mathbf{W}^Y} L^Y_{train}(\mathbf{W}^Y, \mathbf{\Theta}),\\ &\qquad \mathbf{W}^Z - \xi\nabla_{\mathbf{W}^Z} L^Z_{train}(\mathbf{W}^Z, \mathbf{\Theta}), \mathbf{\Theta}\big)\end{aligned}$$

where ξ is the learning rate for updating $\mathbf{W}^Y$ and $\mathbf{W}^Z$. The approximation scheme estimates $\mathbf{W}^{Y*}(\mathbf{\Theta})$ and $\mathbf{W}^{Z*}(\mathbf{\Theta})$ by updating $\mathbf{W}^Y$ and $\mathbf{W}^Z$ for a single training step, which avoids the total optimization $\mathbf{W}^*(\mathbf{\Theta}) = \arg\min_{\mathbf{W}} L_{train}(\mathbf{W}, \mathbf{\Theta}^*)$ to the convergence. In our implementation, we apply first-order approximation with $\xi = 0$, which can even lead to more speed-up. Also, in our specific experiments, since the amount of validation data is limited, we build an augmented validation dataset $\mathcal{V}' = \mathcal{V} \cup \mathcal{T}$ combining the original validation set $\mathcal{V}$ with the training set $\mathcal{T}$ for optimizing Θ.

We present our DARTS-based optimization algorithm in Algorithm 1. In each iteration, we first update the corrector's parameters based on the augmented validation set $\mathcal{V}'$ (lines 2-3). Then, we collect a new mini-batch of training data (line 4). We generate the saliency scores $S = (s_1, s_2, \ldots, s_n)$ for the training examples via the corrector with its current parameters (line 5). Next, we make predictions via the classifiers with their current parameters and X_S (line 6). Eventually, we update the parameters of the classifiers (line 7).

5 Experiment

In this section, we conduct experiments to evaluate our proposed debiasing framework. Through the

Algorithm 1: The DARTS-based optimization method for Debiased-TC.

```
Input: Training data $\mathcal{T} = \{X_i, Y_i, Z_i\}_{i=1}^{|\mathcal{T}|}$ and
  Validation data $\mathcal{V} = \{X_i, Y_i, Z_i\}_{i=1}^{|\mathcal{V}|}$
Output: classifier parameters $\mathbf{W}^{Y*}$ and $\mathbf{W}^{Z*}$; and
  corrector parameters $\mathbf{\Theta}^*$
Initialize $\mathbf{W}^Y$, $\mathbf{W}^Z$ and $\mathbf{\Theta}$
while not converged do
    Sample a mini-batch of validation data from
      $\mathcal{V}' = \mathcal{V} \cup \mathcal{T}$
    Update $\mathbf{\Theta}$ by descending
      $\nabla_{\mathbf{\Theta}}\, L_{val}(\mathbf{W}^Y - \xi \nabla_{\mathbf{W}^Y} L^Y_{train}(\mathbf{W}^Y, \mathbf{\Theta}),$
      $\mathbf{W}^Z - \xi \nabla_{\mathbf{W}^Z} L^Z_{train}(\mathbf{W}^Z, \mathbf{\Theta}), \mathbf{\Theta})$
      ($\xi = 0$ for first-order approximation)
    Collect a mini-batch of training data from $\mathcal{T}$
    Generate $S$ via the corrector with current parameters
      $\mathbf{\Theta}$
    Generate predictions via the classifiers with current
      parameters $\mathbf{W}^Y$, $\mathbf{W}^Z$ and $X_S$
    Update $\mathbf{W}^Y$ and $\mathbf{W}^Z$ by descending
      $\nabla_{\mathbf{W}^Y} L^Y_{train}(\mathbf{W}^Y, \mathbf{\Theta})$ and $\nabla_{\mathbf{W}^Z} L^Z_{train}(\mathbf{W}^Z, \mathbf{\Theta})$
end while
```

experiments, we try to answer two questions: 1) Does our framework effectively mitigate the implicit bias in various deep text classification models? and 2) Does our framework maintain the performance of the original models (without debasing) while reducing the bias?

5.1 Baselines

In our experiments, we compare our proposed debiasing framework with two baselines. Since there is no established method for mitigating implicit bias, we adopt two debiasing methods designed for traditional explicit bias and adapt them for implicit bias.

Data Augmentation* (Data Aug) (Dixon et al., 2018). We manually balance the training data of two demographic groups by adding sufficient negative examples for Group I and positive examples for Group II. As a result, the ratio of positive and negative training examples for both groups is 1:1. As discussed in the introduction, obtaining additional labeled data from specific authors is very expensive. In this work, we seek to develop a bias mitigation methodology without extra data. *Since Data Aug introduces more training data, it's not fair to directly compare it with other debiasing methods that only utilize original training data (including our method). We include Data Aug as a special baseline for reference.*

Instance Weighting (Ins Weigh) (Zhang et al., 2020). We re-weight each training instance with a numerical weight $\frac{P(Y)}{P(Y|Z)}$ based on the label distribution for each demographic group to mitigate explicit bias. In this method, a random forest classifier is built to estimate the conditional distribution $P(Y|Z)$ and the marginal distribution $P(Y)$ is manually calculated.

5.2 Experimental Settings

We conduct our experiments for implicit bias mitigation on two representative base models: **CNN** (Kim, 2014) and **RNN** (Chung et al., 2014). We use the same datasets with manually designed proportions, as described in Section 2.2. The details of the base models, as well as the implementation details of the experiments, can be found in Appendix C.

5.3 Performance Comparison

We train the base models with our proposed debiasing framework as well as the baseline debiasing methods. We report the performance on the test set in terms of fairness and classification performance.

Fairness Evaluation. Table 3 shows the results for fairness evaluation metrics: false positive equality difference (FPED), false negative equality difference (FNED), and DPD. FPED/FNED indicates the absolute value of the difference between the false positive/negative rates of the two groups. We make the following observations. First, the base models attain high FPED, FNED, and DPD, which indicates the existence of significant implicit bias towards the authors of the texts. Ins Weigh seems ineffective in mitigating implicit bias since it only achieved comparable fairness scores with the base models. Note that not every example that belongs to a certain group necessarily results in bias towards that group. Thus, assigning a uniform weight for all examples with the same label Y and demographic attribute Z is not a proper way to reduce implicit bias. Third, both Data Aug and Debiased-TC can mitigate the implicit bias by achieving lower equality and demographic parity differences. However, compared to Data Aug, Debiased-TC has two advantages. First, Data Aug needs to add more training data while Debiased-TC does not. Debiased-TC can locate the main source of implicit bias by analyzing how it forms in a deep text classification model. Due to the proposed corrector model, it can make a classification model focus on the relevant features for predictions and discard the features that may lead to implicit bias. Second, Debiased-TC is more stable than Data Aug. For the sentiment classification task with race as the demographic

Table 3: Fairness Performance Comparison. Note that Data Aug is a special baseline for reference.

Task	Methods	CNN			RNN		
		FPED (%)	FNED (%)	DPD (%)	FPED (%)	FNED (%)	DPD (%)
Sentiment Race (DIAL)	**Base Model**	23.59	41.45	32.52	26.86	42.36	34.61
	Data Aug*	21.00*	3.88*	12.44*	19.84*	0.59*	10.22*
	Ins Weigh	25.47	41.43	33.45	26.86	42.36	34.61
	Debiased-TC	6.08	4.63	0.73	6.67	5.68	0.50
Mention Race (DIAL)	**Base Model**	32.73	17.58	25.16	30.44	17.55	24.00
	Data Aug*	1.31*	7.31*	3.00*	0.77*	7.91*	4.34*
	Ins Weigh	24.66	19.46	22.06	28.83	17.26	23.05
	Debiased-TC	3.61	2.40	0.61	4.97	1.07	1.95
Mention Gender (PAN16)	**Base Model**	11.60	9.95	10.78	10.62	8.33	9.47
	Data Aug*	0.84*	0.19*	0.32*	2.42*	0.72*	1.57*
	Ins Weigh	12.73	10.22	11.47	11.20	9.35	10.28
	Debiased-TC	3.95	3.04	3.49	5.41	3.73	4.57
Mention Age (PAN16)	**Base Model**	15.03	9.96	12.49	13.07	7.34	10.20
	Data Aug*	3.71*	1.59*	1.06*	0.17*	2.69*	1.26*
	Ins Weigh	16.53	8.71	12.62	13.24	7.94	10.59
	Debiased-TC	7.29	2.91	5.10	7.64	2.69	5.16
Hate Speech Race (MTC)	**Base Model**	78.56	37.22	57.89	81.51	28.50	55.01
	Data Aug*	88.81*	26.15*	57.48*	83.51*	22.73*	53.12*
	Ins Weigh	87.51	31.92	59.72	84.45	27.44	55.95
	Debiased-TC	75.97	17.08	46.53	74.56	18.85	46.70

attribute, the CNN and RNN classifiers trained on augmented data still result in high FPED and DPD scores. This suggests that balancing the training data cannot always mitigate implicit bias. In fact, only training examples with demographic language features can contribute to the implicit bias. Since some texts in the training set do not contain any language features belonging to a demographic group, they do not help balance the data.

Text Classification Performance Evaluation. The prediction performance of the text classification models trained under various debiasing methods is shown in Table 4, where we report the accuracy and F1 scores. First, it is not surprising to see that Data Aug achieves the best performances, since the data augmentation technique introduces more training data. It's not fair to directly compare it with other debiasing methods that only utilize original training data. Second, in most cases, our method achieves comparable or even better performance than the original base models. As we verified before, the implicit bias of a text classification model is caused by the fact that it learns a wrong correlation between labels and demographic language features. Debiased-TC corrects the model's selection of language features for predictions and thereby improves its performance on the classification task.

In conclusion, our proposed debiasing framework significantly mitigates the implicit bias, while maintaining or even slightly improving the classification performance.

6 Related Work

Fairness in Machine Learning. With the wide spread of the machine learning (ML) applications in our daily lives, bias and fairness issues in them are drawing increasing attention from the community. Researches are conducted to detect and mitigate the bias in ML models on various tasks. Specifically, studies investigate how algorithms can be biased in classification (Kamiran and Calders, 2009; Chouldechova, 2017), regression (Berk et al., 2017; Agarwal et al., 2019), and clustering tasks (Backurs et al., 2019; Chen et al., 2019). In the domain of computer vision, researchers show that ML-based face recognition (Buolamwini and Gebru, 2018) and object detection (Ryu et al., 2017) models perform unfairly for different demographic groups. Besides, a lot of works examine the bias in language related tasks, including word embedding (Bolukbasi et al., 2016), coreference resolution (Zhao et al., 2018), machine translation (Prates et al., 2019) and dialogue generation (Liu et al., 2020a,b), etc. Moreover, some recent studies also explore the relationship between the fairness of an ML model and its other properties, such as robustness (Xu et al., 2020; Nanda et al., 2021) and privacy (Cummings et al., 2019).

Fairness in Text Classification. In this work, we focus on the fairness issues in the text classification task. In this task, Dixon et al. (2018)

Table 4: Text Classification Performance Comparison (%). Note that Data Aug is a special baseline for reference.

Methods	Sentiment/Race (DIAL)		Mention/Race (DIAL)		Mention/Gender (PAN16)		Mention/Age (PAN16)		Hate Speech/Race (MTC)	
	Acc.	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
					CNN					
Base Model	61.40	60.03	70.77	71.65	81.93	81.94	80.57	80.17	64.10	65.86
Data Aug*	67.58*	71.53*	76.42*	76.03*	84.11*	84.31*	84.08*	84.36*	66.96*	71.10*
Ins Weigh	61.06	60.36	71.62	69.66	81.86	81.85	80.70	81.05	65.25	68.73
Debiased-TC	63.60	66.58	73.15	71.84	81.67	82.01	80.41	79.68	69.14	72.69
					RNN					
Base Model	61.23	61.53	72.97	73.68	83.46	83.40	82.78	82.43	66.31	69.57
Data Aug*	67.82*	69.35*	78.42*	77.26*	86.25*	86.05*	86.12*	85.68*	68.55*	72.37*
Ins Weigh	61.23	61.53	73.37	73.79	83.46	83.32	82.80	82.58	67.26	70.94
Debiased-TC	63.68	66.70	74.05	73.41	81.81	81.51	80.21	79.17	66.76	70.76

demonstrate that the source of unintended bias in models is the imbalance of training data, and they provide a debiasing method, which introduces new data to balance the training data. In (Park et al., 2018), gender bias is measured on abusive language detection models, and the effects of different pretrained word embeddings and model architectures are analyzed. By considering the various ways that a classifier's score distribution can vary across designated groups, a suite of threshold-agnostic metrics is introduced in (Borkan et al., 2019), which provides a nuanced view of unintended bias. Furthermore, the work (Zhang et al., 2020) proposes to debias text classification models using instance weighting, i.e., different weights are assigned to the training samples involving different demographic groups. The works discussed above focus on explicit bias, where the demographic attributes are explicitly expressed in the text. However, works studying implicit bias are rather limited. Huang et al. (2020) introduce the first multilingual hate speech dataset with inferred author demographic attributes. Through experiments on this dataset, they show that popular text classifiers can learn the bias towards the demographic attribute of the author. But this work doesn't discuss how the bias is produced, and no debiasing method is provided.

7 Conclusion

In this paper, we demonstrate that a text classifier with implicit bias makes predictions based on language features correlated with demographic groups of authors, and propose a novel learning framework Debiased-TC to mitigate such implicit bias. The experimental results show that Debiased-TC significantly mitigates implicit bias, and maintains or even improves the text classification performance of the original models. In the future, we will investigate implicit bias in other NLP applications.

Acknowledgments

Haochen Liu, Wei Jin, Hamid Karimi and Jiliang Tang are supported by the National Science Foundation (NSF) under grant number CNS1815636, IIS1928278, IIS1714741, IIS1845081, IIS1907704, and IIS1955285. Zitao Liu is supported by Beijing Nova Program (Z201100006820068) from Beijing Municipal Science & Technology Commission.

References

Alekh Agarwal, Miroslav Dudik, and Zhiwei Steven Wu. 2019. Fair regression: Quantitative definitions and reduction-based algorithms. In International Conference on Machine Learning, pages 120–129. PMLR.

Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. 2019. Scalable fair clustering. In International Conference on Machine Learning, pages 405–413. PMLR.

Richard Berk, Hoda Heidari, Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. 2017. A convex framework for fair regression. arXiv preprint arXiv:1706.02409.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in nlp. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5454–5476.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of african-american english. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1119–1130.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advances in neural information processing systems, pages 4349–4357.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In Companion Proceedings of The 2019 World Wide Web Conference, pages 491–500.

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on fairness, accountability and transparency, pages 77–91. PMLR.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. 2018. Learning to explain: An information-theoretic perspective on model interpretation. In International Conference on Machine Learning, pages 883–892. PMLR.

Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. 2019. Proportionally fair clustering. In International Conference on Machine Learning, pages 1032–1041. PMLR.

Alexandra Chouldechova. 2017. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big data, 5(2):153–163.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In NIPS 2014 Workshop on Deep Learning, December 2014.

Florian Coulmas. 2013. Sociolinguistics: The study of speakers' choices. Cambridge University Press.

Thomas M Cover. 1999. Elements of information theory. John Wiley & Sons.

Rachel Cummings, Varun Gupta, Dhamma Kimpara, and Jamie Morgenstern. 2019. On the compatibility of privacy and fairness. In Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, pages 309–315.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, pages 67–73.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In Proceedings of the 3rd innovations in theoretical computer science conference, pages 214–226.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 11–21.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In International conference on machine learning, pages 1180–1189. PMLR.

Xiaolei Huang, Linzi Xing, Franck Dernoncourt, and Michael Paul. 2020. Multilingual twitter corpus and baselines for evaluating demographic bias in hate speech recognition. In Proceedings of The 12th Language Resources and Evaluation Conference, pages 1440–1448.

Faisal Kamiran and Toon Calders. 2009. Classifying without discriminating. In 2009 2nd International Conference on Computer, Control and Communication, pages 1–6. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In ICLR (Poster).

Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In Advances in neural information processing systems, pages 4066–4076.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. In International Conference on Learning Representations.

Haochen Liu, Jamell Dacon, Wenqi Fan, Hui Liu, Zitao Liu, and Jiliang Tang. 2020a. Does gender matter? towards fairness in dialogue systems. In Proceedings of the 28th International Conference on Computational Linguistics, pages 4403–4416.

Haochen Liu, Wentao Wang, Yiqi Wang, Hui Liu, Zitao Liu, and Jiliang Tang. 2020b. Mitigating gender bias for neural dialogue generation with adversarial learning. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 893–903.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In International Conference on Machine Learning, pages 2113–2122.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-end bias mitigation by modelling biases in corpora. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8706–8716.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. arXiv preprint arXiv:1908.09635.

Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P Dickerson. 2021. Fairness through robustness: Investigating robustness disparity in deep learning. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pages 466–477.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2799–2804.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. In ICML.

Marcelo OR Prates, Pedro H Avelar, and Luis C Lamb. 2019. Assessing gender bias in machine translation: a case study with google translate. Neural Computing and Applications, pages 1–19.

Daniel Preoţiuc-Pietro and Lyle Ungar. 2018. User-level race and ethnicity predictors from twitter text. In Proceedings of the 27th International Conference on Computational Linguistics, pages 1534–1545.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. Working Notes Papers of the CLEF, 2016:750–784.

Hee Jung Ryu, Margaret Mitchell, and Hartwig Adam. 2017. Improving smiling detection with race and gender diversity. arXiv preprint arXiv:1712.00193, 1(2):7.

Deven Santosh Shah, H Andrew Schwartz, and Dirk Hovy. 2020. Predictive biases in natural language processing models: A conceptual framework and overview. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5248–5264.

Han Xu, Xiaorui Liu, Yaxin Li, and Jiliang Tang. 2020. To be robust or to be fair: Towards fairness in adversarial training. arXiv preprint arXiv:2010.06121.

Guanhua Zhang, Bing Bai, Junqi Zhang, Kun Bai, Conghui Zhu, and Tiejun Zhao. 2020. Demographics should not be the reason of toxicity: Mitigating discrimination in text classifications with instance weighting. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4134–4145.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 15–20.

A Details of the Datasets

In this section, we describe the statistical information and the annotation process of the three datasets DIAL, PAN16, and MTC used in our experiments. The datasets DIAL and PAN16 can be downloaded from the link [2]. The dataset MTC can be downloaded from the link [3].

A.1 Data Statistics

The statistics of the datasets DIAL, PAN16, and MTC are shown in Table 5. In the table, the "task" section shows the text classification tasks included in a dataset. "Sentiment" is short for sentiment analysis. "Mention" is short for mention detection. "Hate Speech" is short for hate speech detection. "Demog." indicates the demographic attribute of the tweet authors collected in a dataset. The "Size" section shows the total number of instances in a dataset. Each instance is a tweet text. The "Avg.Len." section shows the average number of words in one instance in a dataset.

Table 5: Statistics of the datasets.

Dataset	Task	Demog.	Size	Avg.Len.
DIAL	Sentiment	Race	317,151	11.20
	Mention	Race	400,000	10.56
PAN16	Mention	Gender	175,871	14.64
	Mention	Age	175,471	14.55
MTC	Hate Speech	Race	47,627	19.60

A.2 Data Annotation

The DIAL dataset is annotated based on the dialectal tweet corpus (Blodgett et al., 2016), which contains 59.2 million tweets from 2.8 million users. The race attribute is annotated by an automated probabilistic inference method based on the geolocation information of the user and the tweet text. Given that geolocation information (residence) is highly associated with the race of a user, the model can make accurate predictions. To further ensure the accuracy, DIAL only keeps annotations with confidence above 80%.

---

[2] `https://github.com/yanaiela/demog-text-removal`

[3] `https://github.com/xiaoleihuang/Multilingual_Fairness_LREC`

The PAN16 dataset (Rangel et al., 2016) contain 436 Twitter users, each of which has up to 1,000 tweets. The age and gender of the users are manually annotated by referring to their LinkedIn profiles. Specifically, annotators judge the gender based on the user's name and profile photo. The age is inferred based on the user's birth date or degree starting date.

The MTC dataset (Huang et al., 2020) is annotated based on 7 published Twitter hate speech datasets in five languages. The dataset contains user demographic information such as race, gender, age, and country. We only focus on the English corpus and the attribute race in our experiments. The race of a user is inferred by the computer vision API, Face++[4], based on the profile photo.

B Optimization of the Explainer

We train the explainer $\mathcal{E}$ by maximizing the mutual information between the response variable Y and the selected features X_S. The optimization problem can be formulated as:

$$\begin{aligned} \max_{\mathcal{E}} \quad & I(X_S; Y) \\ \text{s.t.} \quad & S \sim P_{\mathcal{E}}(S|X) \end{aligned} \tag{1}$$

where

$$\begin{aligned} I(X_S, Y) &= \mathbb{E}\Big[\log \frac{P(X_S, Y)}{P(X_S)P(Y)}\Big] \\ &= \mathbb{E}\Big[\log \frac{P_{\mathcal{M}}(Y|X_S)}{P(Y)}\Big] \\ &\propto \mathbb{E}\Big[\log P_{\mathcal{M}}(Y|X_S)\Big] \\ &= \mathbb{E}_X \mathbb{E}_{S|X} \mathbb{E}_{Y|X_S}\Big[\log P_{\mathcal{M}}(Y|X_S)\Big] \end{aligned}$$

Solving the optimization problem in Eq. (1) is equivalent to finding an explainer $\mathcal{E}$ satisfying the following:

$$\max_{\mathcal{E}} P_{\mathcal{M}}(Y|X_S) \quad \text{s.t.} \qquad S \sim P_{\mathcal{E}}(S|X).$$

Hence, we train the explainer $\mathcal{E}$ by optimizing $P_{\mathcal{M}}(Y|X_S)$ with the parameters of the classification model $\mathcal{M}$ fixed. In our implementation, we adopt the cross-entropy loss for training, as we do when we train the classification model $\mathcal{M}$.

C Implementation Details

C.1 Details of Base Models

In the base model CNN, we use 100 filters with three different kernel sizes (3, 4, and 5) in the convolution layer, where we use a Rectified Linear Unit (ReLU) as the non-linear activation function. Each obtained feature map is processed by a max-pooling layer. Then, the features are concatenated and fed into a linear prediction layer to get the final predictions. A dropout with a rate of 0.3 is applied before the linear prediction layer.

For the base model RNN, we use a one-layer unidirectional RNN with Gated Recurrent Units (GRU). The hidden size is set to 300. The last hidden state of the RNN is fed into a linear prediction layer to get the final predictions. We apply a dropout with a rate of 0.2 before the linear prediction layer.

C.2 Details of Experimental Settings

For the text classifiers, we use randomly initialized word embeddings with a size of 300. All the models are trained by an Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.001. We apply gradient clipping with a clip-value of 0.25 to prevent the exploding gradient problem. The batch size is set to 64. For the base model and the baseline methods, when the prediction accuracy of the validation data doesn't improve for 5 consecutive epochs, the training is terminated, and we pick the model with the best performance on the validation set. Our model utilizes the validation data for training. To avoid it overfitting the validation data, we don't select the model based on its performance on the validation set. Instead, we train the model for a fixed number of epochs (5 epochs, the same for all the three datasets) and evaluate the obtained model.

---

[4] `https://www.faceplusplus.com/`

From What to Why:
Improving Relation Extraction with Rationale Graph

Zhenyu Zhang, Bowen Yu, Xiaobo Shu, Mengge Xue, Tingwen Liu[*] and **Li Guo**
Institute of Information Engineering, Chinese Academy of Sciences. Beijing, China
School of Cyber Security, University of Chinese Academy of Sciences. Beijing, China
`{zhangzhenyu1996, yubowen, shuxiaobo}@iie.ac.cn`
`{xuemengge, liutingwen, guoli}@iie.ac.cn`

Abstract

Which type of information affects the existing neural relation extraction (RE) models to make correct decisions is an important question. In this paper, we observe that entity type and trigger are the most indicative information for RE in each instance. Moreover, these indicative clues are always constrained to co-occur with specific relations at the corpus level. Motivated by this, we propose a novel RAtionale Graph (RAG) to organize such co-occurrence constraints among entity types, triggers and relations in a holistic graph view. By introducing two subtasks of entity type prediction and trigger labeling, we build the connection between each instance and RAG, and then leverage relevant global co-occurrence knowledge stored in the graph to improve the performance of neural RE models. Extensive experimental results indicate that our method outperforms strong baselines significantly and achieves state-of-the-art performance on the document-level and sentence-level RE benchmarks.

1 Introduction

Relation extraction (RE), which aims to identify the semantic relation between two entities in plain text, is one of the fundamental tasks in information extraction (IE). In the deep learning era, many approaches are proposed including models based on attention mechanism (Lin et al., 2016; Zhang et al., 2017), graph neural networks (Zhang et al., 2018; Guo et al., 2019), and pre-trained language models (Joshi et al., 2020; Yu et al., 2020).

While these neural RE models have achieved the latest state-of-the-art results, little is known about which type of information affects the models to make decisions. Recently, an empirical study shows that the understanding of two main information sources, entity type, and textual context, is necessary and effective for training a RE model (Peng et al., 2020). Entity type, is always an important side information for RE (Liu et al., 2014; Vashishth et al., 2018). In the textual context, some words play an indicative role in relation expression. Yu et al. (2020) initially annotated the minimal contiguous indicative word span and named them *trigger*. For example, in Figure 1, when we notice that both the subject and object entities are `person`, as well as the trigger `children` appears in the context, our immediate reaction is that they probably hold a parent-child relation, then we make a further judgment by reading the complete text.

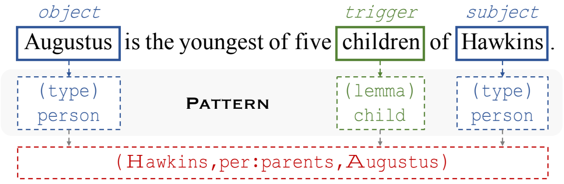

Figure 1: Illustration of the decision-making process in RE, where patterns are the most indicative information.

What is the support behind such rapid and accurate decision-making of human beings? In RE, if we look at the entire corpus from a global view, we can find a common phenomenon that one certain entity type or trigger is constrained to co-occur with specific relations. Taking entity type as an example, two entities of type `person` can only participate in person-related relations (e.g., `per:parents`, `per:siblings`). Such global co-occurrence induced by multiple seen instances serves as the crucial prior knowledge in the process of human cognition (Chater et al., 2006), and can naturally form a bipartite graph, in which the nodes on two sides are entity types and relations respectively. Similarly, the same logic can also go for triggers.

Inspired by the above observation, in this paper, we propose a RAtionale Graph (RAG) to organize the global co-occurrence statistics aggregated from

---

[*] Corresponding Author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 86–95
August 1–6, 2021. ©2021 Association for Computational Linguistics

the corpus. Specifically, nodes in the graph are constructed based on the relations and patterns[1]. There are totally four types of directed edges that exist between different types of nodes. For example, the edge between a trigger node and a relation node depicts the co-occurrence probability of a text expressing the relation when the trigger appears in the text. This probabilistic knowledge, together with the involved nodes, is collectively referred to as *rationale*. In the end, RAG is excepted to present a holistic view of all patterns and relations, and then facilitate the relation prediction.

Now we incorporate RAG with neural networks to improve the RE performance. Given an instance with a text and two entities, we first predict the entity type and label the trigger, then establish the link between the input instance with the known patterns in RAG, and finally enhance the instance representation with the attended relation node features in the graph. Meanwhile, we introduce the gate mechanism and graph neural networks (GNNs) to perform the information propagation from the input instance to relation nodes. Hence, this workflow makes full use of all aforementioned rationale knowledge to guide the processing of new instances by linking them to each seen pattern stored in the graph, like humans recognizing new things by intuitively associating with the knowledge they have memorized. In the training phase, the model learns simultaneously (1) the relation along with (2) the entity type and trigger for each instance. This means that we care about not only the final relation label (*what*), but also the intermediate results, i.e., whether the entity type and trigger are correctly predicted (*why*). By doing so, we can retrieve the relevant global pattern knowledge from the graph with the predicted trigger and entity types, during testing.

To evaluate our approach, we first conduct experiments on the document-level RE task DialogRE (Yu et al., 2020). Experimental results show the benefits of the proposed method, leading to state-of-the-art performance. An exciting discovery is that our method is very effective in small-scale annotation scenes, using only half (with 2,584 positive instances) of the pattern-annotated instances results in a comparable performance as using all conventional annotated instances. To further validate this advantage, we manually annotate 20% (with 2,585 positive instances) patterns of the sentence-level RE benchmark TACRED (Zhang et al., 2017), and empirically demonstrate similar experimental conclusions with DialogRE.

2 Related Work

Extracting relational facts between entities from text is an essential and classical problem in natural language processing. The popular research methods have gone through the iteration from pattern-based methods (Mooney, 1999; Chang and Lui, 2001) to feature-based methods (Kambhatla, 2004; Zhou et al., 2005), and then to neural-based methods (Zeng et al., 2014; Zhang et al., 2017). Nowadays, most state-of-the-art work develops powerful neural models based on pre-trained language models or graph neural networks (Soares et al., 2019; Zhang et al., 2019; Guo et al., 2019). All the time, there are two main consensuses in the community: when extracting a relation, entity types are important side indicators, which are often used to enhance the input or output layer (Vashishth et al., 2018; Kuang et al., 2020). On the other hand, not all the words in the text are beneficial to RE. Thus there are also efforts focusing on the heuristic or implicit selection of the key clues related to relation expression (Zhang et al., 2018; Yu et al., 2019), and Yu et al. (2020) is the first work to annotate such clue words in texts and name them trigger.

However, most previous studies are only based on local features, in other words, models are trained on individual instance, limiting the ability to capture the connection between textual indicative information and relations globally. Conversely, Su et al. (2018) emphasized the importance of the global view, and embed the textual relations with global statistics to combat the wrong labeling problem of distant supervision. Wang et al. (2020) proposed an interpretable network embedding model based on a corpus-level entity graph to rationalize medical relation prediction. Unfortunately, their methods are not suitable for the supervised RE task in the general domain. The most related work, (Zhang et al., 2020), collected a global type-relation mapping as prior knowledge to guide the optimization with knowledge distillation. One major difference is that we systematically consider both entity type and textual trigger to collect all indicative knowledge in a holistic view. Another unique aspect of this work is that we perform the prediction of entity type and trigger as two subtasks, while previous studies only focus on the final relation labels.

---

[1] For the sake of generality, we refer to the *entity type* and *trigger* as *pattern* in the remaining of this paper.

3 Rationale Graph (RAG)

Different from existing work only using raw text for RE, we assume the global co-occurrence statistics among relations, triggers, and entity types is given, which are pre-construed based on the whole corpus, and denoted as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ refers the *relation*, *trigger*, or *entity type pair* extracted from the corpus and each edge $e \in \mathcal{E}$ is associated with the global co-occurrence count for the connected nodes. Inspired by Zhang et al. (2020), we organize the global co-occurrence count between two kinds of nodes as bipartite rationale mapping and pack all bipartite mappings together to obtain a rationale graph (RAG). Figure 2 shows the schematic diagram for clarity.

3.1 Bipartite Rationale Mapping

Here we take *type* (short for *entity type pair*) and *relation* as an example to describe the construction process of bipartite rationale mapping. Specifically, for instance with a text x and two entities (s, o), we combine two entity types to achieve a *pattern* p. From this step, we obtain the pattern set $\mathcal{T} = \{t_i\}$ and formulate a support set $\mathcal{S}(t_i)$ for each t_i, in which the support set $\mathcal{S}(t_i)$ contains all instances with pattern t_i. Besides, we also collect a set of relations $\mathcal{R} = \{r_j\}$, and the support set $\mathcal{S}(r_j)$ denoting the set of instances holding relation r_j. The co-occurrence number of pattern t_i and relation r_j is defined as $w_{ij} = |\mathcal{S}(t_i) \cap \mathcal{S}(r_j)|$. In other word, every instance (x, s, o) with pattern t_i and relation r_j is counted as a co-occurrence of t_i and r_j.

However, it is inappropriate to take the raw co-occurrence count as mapping weight directly. The relation distribution in reality typically has a power-law tail (Zhang et al., 2017), meaning that the count spans several orders of magnitude in different relations. To meet this challenge, for each pattern, we normalize its co-occurrence count to form a valid probability distribution over relations. In the end, the bipartite mapping $\mathcal{M}_{tp2re}$ is constructed, with one node set being the types, the other being the relations, and the weighted edges $\bar{w}_{ij} = p(r_j|t_i) = w_{ij}/\sum_{j'} w_{ij'}$ representing the normalized global co-occurrence probability.

3.2 Graph Construction

Considering that trigger and type are two kinds of information sources for RE (Peng et al., 2020), we first introduce the bipartite rationale mapping from type to relation $\mathcal{M}_{tp2re}$ and the mapping from trigger to relation $\mathcal{M}_{tg2re}$ in RAG. In this way, we assume that the graph reflects the prior probability of relation when some indicative information appears in the text. Furthermore, triggers are actually relations in the form of natural language (Hu et al., 2020) and entity types are tightly bound to certain trigger words within the context (Lin et al., 2020). In other words, type and trigger are mutually related and restricted. Therefore, we introduce a set of bidirectional mapping, that is, from type to trigger $\mathcal{M}_{tp2tg}$ and from trigger to type $\mathcal{M}_{tg2tp}$. Finally, we place four kinds of edges in the graph: $\mathcal{E} \leftarrow \{\mathcal{M}_{tp2re}, \mathcal{M}_{tg2re}, \mathcal{M}_{tp2tg}, \mathcal{M}_{tg2tp}\}^2$.

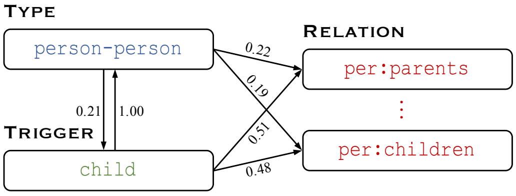

Figure 2: Schematic diagram of RAG, in which edges are weighted by normalized co-occurrence statistics.

4 Relation Extraction with RAG

In this section, we exemplify how to incorporate existing RE models with RAG. Given a text, a subject entity, and an object entity, the model aims to identify the semantic relationship between these two entities with the aid of RAG. Moreover, we also require the model to predict entity type pair and label trigger (if possible) as two auxiliary subtasks. For the example in Figure 3, we build a unified model that not only accurately predicts the relation `per:parents`, but also provides meaningful rationales on how the prediction is made: the subject and object entities are both `person`, and the key clue `children` appears in the context.

4.1 Encoding Module

We utilize BERT (Devlin et al., 2019) as the feature encoder to extract token representations due to its effectiveness in representation learning. Theoretically, the encoding module can be easily replaced by other advanced models. The encoder receives a BERT-style packed sequence and outputs a context representation matrix $\mathbf{H} \in \mathbb{R}^{n \times d}$ with an overall vector $\mathbf{h}_{cls} \in \mathbb{R}^d$ (the representation of the [CLS] token in BERT), where d is the vector dimension

---

[2] In view of the diversity of natural language, we use spaCy (`https://spacy.io/`) to perform lemmatization on triggers, before putting them into RAG as vertexes.

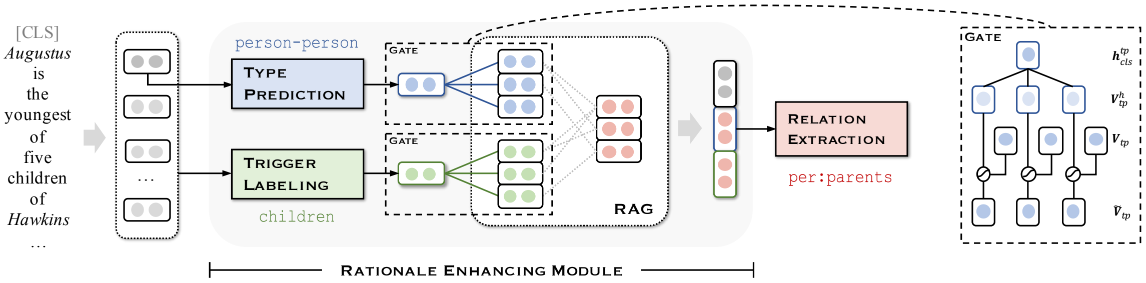

Figure 3: The overall architecture of the proposed model. Rationale enhancing module is the core component in our approach, which enhances the instance representation by retrieving pertinent rationales stored in RAG.

of the last layer of BERT. Typically, existing BERT-based RE solutions first concatenate target entities with the text or mark them in the input sequence with special tokens, and then directly take $\mathbf{h}_{cls}$ as the input of final classification module (Joshi et al., 2020; Yu et al., 2020).

4.2 Rationale Enhancing Module

The rationale enhancing module consists of two enhancing branches and one rationale integration unit. In each branch, we first predict pattern (type or trigger) for the input instance and then calculate the *pattern probability* that the instance belongs to each pattern in RAG. The integration unit aims to collect rationale enhancing features for final relation extraction based on the pattern probability and the rationale in the graph.

4.2.1 Type Enhancing Branch

In this branch, we predict the types of subject and object entities at the same time. Similar to RE, type prediction is regarded as a closed-world classification problem, and the class space is all seen entity type pairs, that is, all type nodes in RAG. Following the classification paradigm of BERT (Devlin et al., 2019), we project the overall vector $\mathbf{h}_{cls}$ into a new space for type prediction:

$$\begin{aligned}\mathbf{h}_{cls}^{tp} &= \texttt{tanh}(\texttt{MLP}_{\{d,d\}}(\mathbf{h}_{cls})),\\ \mathbf{p}_{tp} &= \texttt{SoftMax}\left(\texttt{MLP}_{\{d,n_{tp}\}}(\mathbf{h}_{cls}^{tp})\right).\end{aligned} \tag{1}$$

Here $\texttt{MLP}_{d,n_{tp}}(\cdot)$ denotes a multi-layer perceptron module with input dimension d and output dimension n_{tp}, $\mathbf{p}_{tp} \in \mathbb{R}^{n_{tp}}$ is the type probability that the given instance belongs to each type pair, where n_{tp} is the number of all known type pairs.

4.2.2 Trigger Enhancing Branch

Different from the prediction of entity type, triggers are flexible and can be any word or phrase in the text. We formulate the trigger recognition task as a labeling problem with two label sequences.

Given the representation matrix $\mathbf{H}$ output from BERT, the model predicts two probabilities of each token being the start index and end index of a trigger, respectively. To handle the instances without clear trigger (about half of them), we concatenate $\mathbf{H}$ with $\mathbf{h}_{cls}$ to form $\bar{\mathbf{H}} = [\mathbf{H}; \mathbf{h}_{cls}]$, and set the boundary index pointing to the [CLS] token. These two probability distributions over the entire sequence $\mathbf{p}_{sta}, \mathbf{p}_{end} \in \mathbb{R}^{(n+1)}$ can be obtained by

$$\begin{aligned}\mathbf{p}_{sta} &= \texttt{SoftMax}\left(\texttt{MLP}_{\{d,1\}}(\bar{\mathbf{H}})\right),\\ \mathbf{p}_{end} &= \texttt{SoftMax}\left(\texttt{MLP}_{\{d,1\}}(\bar{\mathbf{H}})\right).\end{aligned} \tag{2}$$

To align the labeling result with the triggers in RAG, we first weight each token in $\bar{\mathbf{H}}$ based on the two index probabilities and get the representation of predicted trigger $\mathbf{h}_{pre}^{tg} \in \mathbb{R}^{d}$, then calculate and normalize the similarity between $\mathbf{h}_{pre}^{tg}$ and all known triggers $\mathbf{V}_{tg} \in \mathbb{R}^{n_{tg} \times d}$:

$$\begin{aligned}\mathbf{h}_{pre}^{tg} &= \frac{1}{2}\left(\mathbf{p}_{sta} + \mathbf{p}_{end}\right)\bar{\mathbf{H}},\\ \mathbf{p}_{tg} &= \texttt{SoftMax}\left(\texttt{sim}(\mathbf{h}_{pre}^{tg}, \mathbf{V}_{tg})\right),\end{aligned} \tag{3}$$

where $\mathbf{p}_{tg} \in \mathbb{R}^{n_{tg}}$ is the probability of the given instance corresponding to each known trigger, n_{tg} is the number of all triggers, and $\texttt{sim}(\cdot)$ is a similarity function as follows:

$$\begin{aligned}\texttt{sim}(\mathbf{h}_{pre}^{tg}, \mathbf{v}_{tg}^{i}) = &\texttt{MLP}_{\{4d,1\}}([\mathbf{h}_{pre}^{tg}; \mathbf{v}_{tg}^{i};\\ &\mathbf{h}_{pre}^{tg} - \mathbf{v}_{tg}^{i}; \mathbf{h}_{pre}^{tg} \circ \mathbf{v}_{tg}^{i}]),\end{aligned} \tag{4}$$

where $\mathbf{v}_{tg}^{i} \in \mathbb{R}^{d}$ is the i-th trigger in $\mathbf{V}_{tg}$ and $\circ$ denotes element-wise product. In that case, even if we run into a new trigger that we have never seen before, we can also estimate the correlation between the new trigger and the known triggers via semantic similarity, and then absorb more global statistics from similar triggers. It provides the possibility for the rationale enhancing on trigger branch.

4.2.3 Rationale Integration

For each type node in RAG, we update its embedding with the instance type feature $\mathbf{h}_{cls}^{tp}$. It is intuitive that the higher the probability of an instance to a type, the more its contribution to the updating process of that type. Specifically, we first compute the update representation for each type node based on the pattern probability $\mathbf{p}_{tp}$, and then aggregate information on the text side $\mathbf{V}_{tp}^{h} \in \mathbb{R}^{n_{tp} \times d}$ and graph side $\mathbf{V}_{tp} \in \mathbb{R}^{n_{tp} \times d}$ via a gate mechanism:

$$\begin{aligned} \mathbf{V}_{tp}^{h} &= \mathbf{p}_{tp}^{\top} \mathbf{h}_{cls}^{tp}, \\ \delta_{tp} &= \texttt{Sigmoid}\left(\texttt{MLP}_{\{2d,1\}}([\mathbf{V}_{tp}; \mathbf{V}_{tp}^{h}])\right), \\ \tilde{\mathbf{V}}_{tp} &= (1-\delta_{tp}) \circ \mathbf{V}_{tp} + \delta_{tp} \circ \mathbf{V}_{tp}^{h}. \end{aligned} \tag{5}$$

Similarly, we perform the same computation on the trigger branch to reconstruct the trigger node embeddings in RAG and result in $\tilde{\mathbf{V}}_{tg} \in \mathbb{R}^{n_{tg} \times d}$.

Next, we execute GNNs-based algorithm on the RAG to update the representation of relation nodes. R-GCN (Schlichtkrull et al., 2018) is chosen as the message propagation strategy here because RAG is naturally a heterogeneous graph:

$$\bar{\mathbf{V}}_{tp}, \bar{\mathbf{V}}_{tg}, \bar{\mathbf{V}}_{re} = \texttt{R-GCN}\left(\tilde{\mathbf{V}}_{tp}, \tilde{\mathbf{V}}_{tg}, \mathbf{V}_{re}\right). \tag{6}$$

After that, for the type enhancing branch, we first calculate the *mapping probability* of an instance to each relation based on the type probability $\mathbf{p}_{tp}$ and corresponding bipartite rationale mapping $\mathbf{M}_{tp2re} \in \mathbb{R}^{n_{tp} \times n_{re}}$ (i.e., the edge weight $\mathcal{M}_{tp2re}$), and then weight the updated relation embeddings based on the mapping probability to obtain type enhancing vector $\mathbf{h}_{tp} \in \mathbb{R}^{d}$. Meanwhile, similar operations are performed in the trigger branch:

$$\begin{aligned} \mathbf{h}_{tp} &= \mathbf{p}_{tp} \mathbf{M}_{tp2re} \bar{\mathbf{V}}_{re}, \\ \mathbf{h}_{tg} &= \mathbf{p}_{tg} \mathbf{M}_{tg2re} \bar{\mathbf{V}}_{re}. \end{aligned} \tag{7}$$

4.3 Classification Module

The output module combines the overall vector and two enhancing features to get final representation, which is fed into a multi-layer perceptron followed by a softmax function for relation classification:

$$\begin{aligned} \mathbf{h}_{re} &= [\mathbf{h}_{cls}; \mathbf{h}_{tp}; \mathbf{h}_{tg}], \\ \mathbf{p}_{re} &= \texttt{SoftMax}\left(\texttt{MLP}_{\{3d,n_{re}\}}(\mathbf{h}_{re})\right). \end{aligned} \tag{8}$$

4.4 Training Objectives

Recall that there are totally three tasks in our model, including relation extraction, type prediction, and trigger (start and end indexes) labeling, which are all reduced to the classification problem. In optimization, we train the model end-to-end in a multi-task manner here, and adopt cross-entropy as the loss function for each task:

$$\mathcal{L}_{task} = \texttt{CrossEntropy}(\mathbf{y}_{task}, \mathbf{p}_{task}), \tag{9}$$

where $\mathbf{y}_{task}$ denotes the ground truth, represented by one-hot vector, $\mathbf{p}_{task \in \{re,tp,sta,end\}}$ is the estimated probability for each class.

Towards learning to perceive the strong signal that a known trigger exactly in the text, we utilize contrastive loss (Hadsell et al., 2006). The intuition is that the trigger in text h_{pre}^{tg} and the matched trigger in RAG v_{tg}^{mat} should have similar representations (i.e., have a small distance in vector space, d). For the mismatched trigger, we expect a margin m between their embeddings. The contrastive loss of trigger matching is as follows, where $\mathbb{1}_{mat}$ is 1 if a trigger is originally in the text and 0 if it is not:

$$\begin{aligned} d &= ||\mathbf{h}_{pre}^{tg} - \mathbf{v}_{tg}^{mat}||_2, \\ \mathcal{L}_{mat} &= (1-\mathbb{1}_{mat})(\max\{0, m-d\})^2 \\ &\quad + (\mathbb{1}_{mat})(d)^2 . \end{aligned} \tag{10}$$

The joint loss of trigger labeling is thus

$$\mathcal{L}_{tg} = \mathcal{L}_{sta} + \mathcal{L}_{end} + \mathcal{L}_{mat}. \tag{11}$$

Finally, the losses from the main RE task and two subtasks are aggregated to form the training objective, with two weight factors λ_{tp} and λ_{tg}:

$$\mathcal{L} = \mathcal{L}_{re} + \lambda_{tp}\mathcal{L}_{tp} + \lambda_{tg}\mathcal{L}_{tg}. \tag{12}$$

Extension. Here, we introduce a simple extension to simultaneously make full use of all data with relation label and any number of data with pattern annotation. Specifically, when there are intact pattern annotations for an instance, we set $\mathbb{1}_{ext}$ to 1 and calculate the losses of type prediction and trigger labeling. Otherwise, we do not calculate them and set $\mathbb{1}_{ext}$ to 0. In this way, the training objective (Equation 12) is modified as follow,

$$\mathcal{L}^{'} = \mathcal{L}_{re} + \mathbb{1}_{ext}(\lambda_{tp}\mathcal{L}_{tp} + \lambda_{tg}\mathcal{L}_{tg}). \tag{13}$$

5 Experiments

We name our proposed model RARE[3], which can be adopted to both document-level and sentence-level RE tasks. Due to the differences in data formats, applicable baseline models, and the custom

---

[3] abbreviation of RAtionale enhanced Relation Extraction

Model	Dev		Test	
	F1 $\pm \sigma$	F1c $\pm \sigma$	F1 $\pm \sigma$	F1c $\pm \sigma$
Majority (Yu et al., 2020)	38.9 $\pm$ 0.0	38.7 $\pm$ 0.0	35.8 $\pm$ 0.0	35.8 $\pm$ 0.0
CNN (Yu et al., 2020)	46.1 $\pm$ 0.7	43.7 $\pm$ 0.5	48.0 $\pm$ 1.5	45.0 $\pm$ 1.4
LSTM (Yu et al., 2020)	46.7 $\pm$ 1.1	44.2 $\pm$ 0.8	47.4 $\pm$ 0.6	44.9 $\pm$ 0.7
BiLSTM (Yu et al., 2020)	48.1 $\pm$ 1.0	44.3 $\pm$ 1.3	48.6 $\pm$ 1.0	45.0 $\pm$ 1.3
BERT (Devlin et al., 2019)	60.6 $\pm$ 1.2	55.4 $\pm$ 0.9	58.5 $\pm$ 2.0	53.2 $\pm$ 1.6
TypeKD$_{BERT}$ (Zhang et al., 2020)†	62.4 $\pm$ 1.1	57.7 $\pm$ 1.0	60.8 $\pm$ 1.5	55.6 $\pm$ 1.4
RARE$_{BERT}$ (ours)	**64.6** $\pm$ 0.7	**60.1** $\pm$ 0.8	**64.2** $\pm$ 1.2	**58.7** $\pm$ 1.1
BERTs (Yu et al., 2020)	63.0 $\pm$ 1.5	57.3 $\pm$ 1.2	61.2 $\pm$ 0.9	55.4 $\pm$ 0.9
TypeKD$_{BERTs}$ (Zhang et al., 2020)†	65.1 $\pm$ 1.2	59.4 $\pm$ 0.9	63.5 $\pm$ 1.3	57.8 $\pm$ 1.2
RARE$_{BERTs}$ (ours)	**67.5** $\pm$ 0.8	**62.6** $\pm$ 1.0	**66.4** $\pm$ 0.8	**61.0** $\pm$ 1.0

Table 1: Main results on the document-level RE (DialogRE) task, σ denotes the standard deviation computed from five independent runs of each model. † marks the results we reproduce based on the official released code.

in handling entities, we conduct two sets of experiments, comparing RARE to their respective state-of-the-art models on the two tasks. In the experiment, we take *bert-base-uncased* as backbone encoder to verify the effectiveness of RARE and perform further analysis. Besides, we reproduce TypeKD (Zhang et al., 2020) as an extra baseline, which is a recent work using global statistics between entity types and relations in RE.

Implementation Details. We follow the same input format and hyper-parameter settings as in baselines for fair composition. Besides, the layer number of RAG is set to 2 (chosen from $\{1, 2, 3\}$), the match margin in $\mathcal{L}_{mat}$ is set to 0.1 (chosen from $\{1, 0.1, 0.01\}$) for the two sets of experiments. We tune the loss weights λ_{tp} and λ_{tg} with grid search (chosen from $[0.01, 0.05]$ in steps of 0.01) and set λ_{tp} to 0.01 and λ_{tg} to 0.03. For the nodes in RAG, we regard entity types, triggers, and relations as plain text, then employ the encoding module to achieve their initial embeddings. All the hyper-parameters are tuned based on dev set.

Evaluation Metrics. Following popular choices and previous work, we use F1/F1c scores as evaluation metrics in the document-level RE task (i.e., DialogRE), where F1c is computed by only taking in the early part of a dialogue as input, instead of the entire dialogue. In the sentence-level RE task (i.e., TACRED/V), we report micro-averaged Precision, Recall, and F1 scores.

5.1 Document-Level Relation Extraction

DialogRE (Yu et al., 2020) is a human-annotated document-level RE dataset constructed from the transcripts of an American television situation comedy *Friends*. It is also the first RE dataset with both entity type and trigger annotation.

	P	R	F1
Type Prediction	79.3	77.4	78.3
Trigger Labeling	51.5	54.2	52.7

Table 2: Performance of two subtasks on DialogRE.

5.1.1 Experimental Setup

We employ BERT and BERTs (Yu et al., 2020) as the encoding module of RARE in this task. BERTs is a speaker-aware version of BERT, achieving the best performance on the dataset. For the completeness of experiments, we include all official baselines: Majority strategy and CNN/LSTM/BiLSTM-based models (Yu et al., 2020).

5.1.2 Results and Analysis

Main Results. Comparing the performance of different models in Table 1, the first conclusion we draw is that RARE$_{BERTs}$ outperforms all baseline models in all evaluation matrices, which demonstrates the effectiveness of our rationale enhanced approach, as well as the motivation of using global pattern co-occurrence statistics to boost the performance of RE models. Secondly, RARE$_{BERTs}$ improves by a relative margin against RARE$_{BERT}$. It is strong evidence that RARE is flexible enough to adapt to various encoders. Thus, we have reason to believe that a more powerful encoding module could bring further performance gain for RARE. Lastly, TypeKD-based models have a similar trend, but their performance is relatively worse than models based on RARE, which shows that trigger and type are two non-overlapping information sources, and only considering one of them is not enough to capture complete indicative knowledge.

We report the performance of RARE$_{BERTs}$ on the two subtasks in Table 2. From the results, we find

	Dev F1
RARE$_{BERT}$	64.6
w/o Rationale graph	62.3
w/o Type enhancing branch	62.8
w/o Trigger enhancing branch	63.3
w/o Trigger matching loss	64.0
w/o Probabilistic edge weights	63.7
w/o Gate mechanism & GNNs	63.5

Table 3: Ablation study on DialogRE dev set.

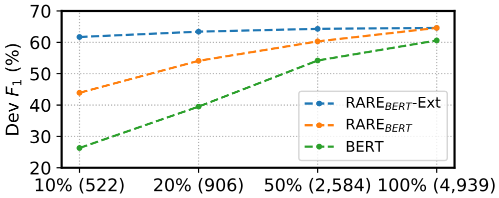

Figure 4: Performance of models on DialogRE dev set with partial training data. The positive instance number with pattern annotation is shown in brackets.

that type prediction is relatively simpler than trigger labeling. We explain that the entity type is a kind of shallow linguistic feature, while the labeling trigger requires a full understanding of context semantics. We also notice that trigger labeling performance is even worse than that of RE, since about half of the positive instances have no explicit trigger (Yu et al., 2020), meaning that the recognition of trigger faces a more serious data imbalance problem than RE. Overall, there is still a long way to improve the performance of these two subtasks, which can be left as a possible future direction.

Ablation Study. To investigate the effectiveness of each module in RARE, we conduct an ablation study on the DialogRE dev set. From the ablations in Table 3, we observe that: (1) Rationale graph is a necessary component that contributes 2.3% F1. The performance superiority of this ablation over BERT also shows that the two auxiliary subtasks of type prediction and trigger labeling are beneficial to RE. (2) Without the type or trigger enhancing branch, the performance degradation suggests that both type and trigger are necessary for our RARE. (3) The ablation of removing the trigger matching loss hurts the final result by 0.6% F1, which justifies the design philosophy of entrusting the model with the ability to perceive whether the trigger is exactly in text. (4) We also try to remove the probabilistic edge weights in RAG to make it degenerate into a standard heterogeneous graph. In that case, the performance drops by 0.9% F1. We think that such probabilistic weights are capable of carrying more global information than one-hot constraints. (5) The information propagation (i.e., gate mechanism and GNNs) brings the improvement of 1.1% F1, which provides a channel to integrate the features of input instance in the output layer.

Labor-Efficiency Study. Considering that most RE datasets have no trigger annotation, we seek to study the cost-effectiveness of adding patterns as additional annotation in this experiment. Accordingly, we explore the performance of RARE$_{BERT}$ and BERT for various fractions of training data. From Figure 4, we can see that RARE$_{BERT}$ with pattern annotations delivers competitive or even better performance as BERT with twice the traditional training data. The drastic performance gain justifies the slightly additional cost incurred in annotating patterns. Furthermore, we also introduce RARE-Ext, the extension of RARE, to fully use the partial data with pattern annotations and the remaining data with only relation labels in training, which provides a plug-and-play manner to utilize pattern annotations. The results show that with the increase of annotations, the performance improvement becomes less significant. When using 50% (with 2,584 positive instances) pattern annotations, the performance of the model is comparable to that of 100% annotations.

5.2 Sentence-Level Relation Extraction

In this section, we evaluate RARE on the sentence-level RE task with two datasets TACRED (Zhang et al., 2017) and TACREV (Alt et al., 2020). TACRED is the most widely used sentence-level RE dataset that constructed from *New York Times*. The recent TACREV (a.k.a TACRED-Revised) dataset has the same training set as TACRED, which corrects the wrong labels in the dev and test sets.

5.2.1 Experimental Setup

To our knowledge, SpanBERT (Joshi et al., 2020) is the best performance model without external knowledge in TACRED. We employ it as another encoder (besides BERT) for RARE. For completeness, we also include two official baselines, LSTM and PA-LSTM (Zhang et al., 2017), as well as two recent graph-based models, AG-GCN (Guo et al., 2019) and LST-AGCN (Sun et al., 2020), here.

Different from DialogRE, TACRED/V annotates only entity types. Inspired by the results of the label-efficiency study on DialogRE, we annotate

Model	TACRED			TACREV		
	P	R	F1	P	R	F1
LSTM (Zhang et al., 2017)	65.7	59.9	62.7	71.5	69.7	70.6
PA-LSTM (Zhang et al., 2017)	65.7	64.5	65.1	74.5	74.1	74.3
AG-GCN (Guo et al., 2019)	73.1	60.9	68.2	77.7	73.4	75.5
LST-AGCN (Sun et al., 2020)	-	-	68.8	-	-	-
BERT (Devlin et al., 2019)‡	67.2	69.3	68.2	76.0	75.6	75.1
TypeKD$_{BERT}$ (Zhang et al., 2020)†	70.6	68.7	69.6	77.9	76.1	77.0
RARE$_{BERT}$-Ext (ours)*	71.4	68.1	**69.8**	78.6	76.2	**77.4**
SpanBERT (Joshi et al., 2020)	70.8	70.9	70.8	75.7	80.7	78.0
TypeKD$_{SpanBERT}$ (Zhang et al., 2020)†	71.7	70.4	**71.0**	79.8	78.3	78.8
RARE$_{SpanBERT}$-Ext (ours)*	72.5	69.3	70.8	80.1	78.0	**79.0**

Table 4: Main results on the sentence-level RE (TACRED/V) task. ‡ marks the results we reproduce based on the repository released by (Joshi et al., 2020). We implement RARE-Ext with 20% extra annotations ($\star$).

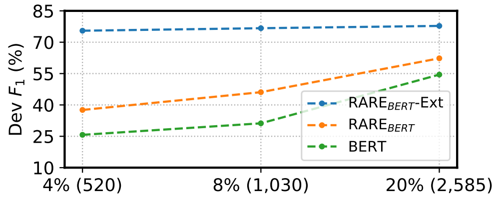

Figure 5: Performance of models on TACREV dev set with partial training data.

triggers for 2,585 positive instances, which accounts for about 20% of all positive instances in the training set of TACRED/V, to verify whether RARE could maintain such excellent lab efficient performance on sentence-level RE task. We repeat our experiments for five random seed initializations, and the results are statistically significant with a p-value of less than 0.05.

5.2.2 Results and Analysis

Main Results. With 20% pattern annotations, we compare RARE-Ext against several representative baselines and summarize the results in Table 4. Similar observations hold that RARE is capable of achieving superior performances with advanced encoding modules. Moreover, RARE-Ext achieves or even surpasses the performance of TypeKD that using 100% type annotations. Although sometimes RARE does not make significant improvements on TACRED, it outperforms the baselines in TACREV and leads to state-of-the-art performances, which is a more accurate evaluation set. Overall, the performance gain of RARE on this task is not as amazing as the document-level task. We analyze that because the sentence is much shorter than the document, and involves fewer relations, BERT-based models are sufficient in capturing the key semantic clue for decision-making, thus the benefits of global knowledge are slightly limited.

Labor-Efficiency Study. Following the approximate number of positive instances in DialogRE, we split the pattern-annotated data to perform the labor-efficiency study on TACREV (see Figure 5). The results indicate that when both using partial data, RARE$_{BERT}$ consistently outperforms BERT. It enlightens us to fully exploit the potential knowledge of the dataset, including local annotation and global statistics, to improve the performance of RE, especially under a low-resource scenario. The considerable progress of RARE$_{BERT}$-Ext demonstrates that RARE is able to improve RE by annotating patterns on any part of an existing dataset. Considering the differences between DialogRE and TACREV (e.g., relation number, domain and style, the ratio of positive and negative instances), it is under investigation whether further improvements could be made by increasing annotations on TACREV, and we leave it as future work.

5.3 Case Study

In Figure 6, we select two representative cases to demonstrate the working principle of RARE. The first case is a short snippet from a DialogRE document, in which two entities are scattered in different sentences, and the context semantics is complex and changeable, BERT fails to capture the relation between them. Conversely, RARE predicts the trigger `engaged` and aligns it with the known trigger `engagement`, and then highlights the strong signal to identify the relation correctly. In the second case, which is from TACREV, BERT mistakenly regards `Jackson Hewitt` as a person, leading to a wrong answer of person-related relation. With the help of type prediction and the global type-relation

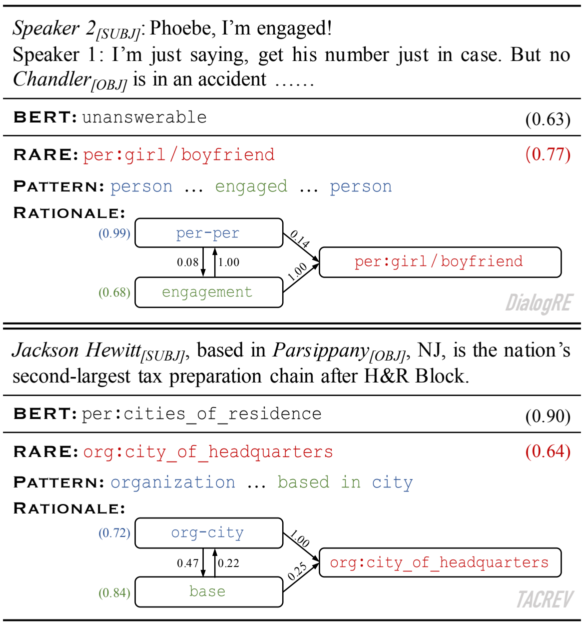

Speaker 2$_{[SUBJ]}$: Phoebe, I'm engaged!
Speaker 1: I'm just saying, get his number just in case. But no *Chandler*$_{[OBJ]}$ is in an accident ……

BERT: `unanswerable` (0.63)

RARE: `per:girl/boyfriend` (0.77)

PATTERN: `person … engaged … person`

RATIONALE:

Jackson Hewitt$_{[SUBJ]}$, based in *Parsippany*$_{[OBJ]}$, NJ, is the nation's second-largest tax preparation chain after H&R Block.

BERT: `per:cities_of_residence` (0.90)

RARE: `org:city_of_headquarters` (0.64)

PATTERN: `organization … based in city`

RATIONALE:

Figure 6: Internal principles of RARE. The number in bracket refers the probabilities predicted by model.

constraints in RAG, RARE could avoid this error and make the right decision.

6 Conclusion

In this paper, we propose a novel rationale graph to organize the global co-occurrence statistics among entity types, triggers, and relations. By introducing the two subtasks of entity type prediction and trigger labeling, we build the connection between input instance and the known patterns in rationale graph, which provides the model with the possibility to benefit from the global co-occurrence knowledge stored in the graph, so as to improve the performance of RE. Experimental results on two public datasets prove the effectiveness of our method. We also highlight two directions for future work: the first is to improve the performance of two subtasks, especially trigger labeling, the other is to adopt the proposed approach in more RE scenarios.

Acknowledgments

We would like to thank all reviewers for their insightful comments and suggestions. We would also like to thank Ning Cong, Minghuan Yuan, Gehang Zhang, and Haoran Xing for their help. This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences (grant No. XDC02040400), and the Youth Innovation Promotion Association of Chinese Academy of Sciences (grant No.2021153).

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. Tacred revisited: A thorough evaluation of the tacred relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1558–1569.

Chia-Hui Chang and Shao-Chen Lui. 2001. Iepad: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*, pages 681–688.

Nick Chater, Joshua B Tenenbaum, and Alan Yuille. 2006. Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences*, 10(7):287–291.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie (NAACL-HLT)*, pages 4171–4186.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 241–251.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742.

Xuming Hu, Lijie Wen, Yusong Xu, Chenwei Zhang, and Philip S Yu. 2020. Selfore: Self-supervised relational feature learning for open relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3673–3682.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics (TACL)*, 8:64–77.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181.

Jun Kuang, Yixin Cao, Jianbing Zheng, Xiangnan He, Ming Gao, and Aoying Zhou. 2020. Improving neural relation extraction with implicit mutual relations. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1021–1032.

Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang

Ren. 2020. Triggerner: Learning with entity triggers as explanations for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8503–8511.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2124–2133.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 2107–2116.

R Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI)*, pages 328–334.

Hao Peng, Tianyu Gao, Xu Han, Yankai Lin, Peng Li, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2020. Learning from context or names? an empirical study on neural relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3661–3672.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the 2018 European Semantic Web Conference (ESWC)*, pages 593–607.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2895–2905.

Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gür, Huan Sun, and Xifeng Yan. 2018. Global relation embedding for relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 820–830.

Kai Sun, Richong Zhang, Yongyi Mao, Samuel Mensah, and Xudong Liu. 2020. Relation extraction with convolutional network over learnable syntax-transport graph. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 8928–8935.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1257–1266.

Zhen Wang, Jennifer Lee, Simon Lin, and Huan Sun. 2020. Rationalizing medical relation prediction from corpus-level statistics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8078–8092.

Bowen Yu, Zhenyu Zhang, Tingwen Liu, Bin Wang, Sujian Li, and Quangang Li. 2019. Beyond word attention: using segment attention in neural relation extraction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5401–5407.

Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. 2020. Dialogue-based relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4927–4940.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 2335–2344.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 35–45.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1441–1451.

Zhenyu Zhang, Xiaobo Shu, Bowen Yu, Tingwen Liu, Jiapeng Zhao, Quangang Li, and Li Guo. 2020. Distilling knowledge from well-informed soft labels for neural relation extraction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 9620–9627.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 427–434.

More Parameters? No Thanks!

Zeeshan Khan[1], **Kartheek Akella**[1], **Vinay P. Namboodiri**[2], **C V Jawahar**[1]
[1]CVIT, IIIT-H
[2]University of Bath
{zeeshank606,sukruthkartheek}@gmail.com
vpn22@bath.ac.uk, jawahar@iiit.ac.in

Abstract

This work studies the long-standing problems of model capacity and negative interference in multilingual neural machine translation (MNMT). We use network pruning techniques and observe that pruning 50-70% of the parameters from a trained MNMT model results only in a 0.29-1.98 drop in the BLEU score. Suggesting that there exist large redundancies in MNMT models. These observations motivate us to use the redundant parameters and counter the interference problem efficiently. We propose a novel adaptation strategy, where we iteratively prune and retrain the redundant parameters of an MNMT to improve bilingual representations while retaining the multilinguality. Negative interference severely affects high resource languages, and our method alleviates it without any additional adapter modules. Hence, we call it parameter-free adaptation strategy, paving way for the efficient adaptation of MNMT. We demonstrate the effectiveness of our method on a 9 language MNMT trained on TED talks, and report an average improvement of +1.36 on high resource pairs. Code will be released here.

1 Introduction

Multilingual neural machine translation(MNMT) has seen various advances in recent years (Dong et al., 2015; Firat et al., 2016; Zoph et al., 2016; Tan et al., 2019; Aharoni et al., 2019; Arivazhagan et al., 2019). However, the core principle behind the effectiveness in terms of modelling multiple languages remains the same, i.e., sharing all the model parameters between all the languages (Johnson et al., 2017). Although highly scalable and effective, the performance on high resource languages decreases as more low resource languages are added in the model; this is called negative interference. To overcome this, recent works (Bapna and Firat, 2019; Philip et al., 2020; Zhang et al., 2020) proposed language-specific adapter modules, which provide extra parameters to learn language specific representations, and overcomes the effect of negative interference caused by a high degree of parameter sharing.

In this paper, we propose an alternative to adapter modules. Instead of adding more parameters, we show that the Transformer (Vaswani et al., 2017) has enough capacity to model multiple languages and overcome negative interference effectively. Inspired by the work of Mallya and Lazebnik (2018), we apply iterative pruning to free up the redundant parameters from an MNMT, and retrain them to learn language specific representations. We start with a trained MNMT model, and prune a fraction of the model parameters, we freeze the surviving parameters and retrain the free ones on a bilingual dataset. This process is iteratively applied for each bilingual pair to get bilingual masks over all the model parameters, as illustrated in figure 1. We show that using only a fraction of redundant parameters, significantly improves the performance on high resource languages. Also, we retain the multilinguality and the zero-shot translation ability after adaptation. By demonstrating the effectiveness of this approach, we open a potential research direction towards parameter-free adaptation in MNMT.

2 Related Work

Adding multiple tasks to a single network: Due to the over-parameterized nature of deep neural networks, prior works (Kirkpatrick et al., 2017; Lee et al., 2017; Li and Hoiem, 2017; Triki et al., 2017) aimed at developing methods to learn multiple tasks while avoiding catastrophic forgetting. Mallya and Lazebnik (2018) proposed an iterative pruning approach to free up parameters for adding new tasks and retain the previously trained parameters at the same time. Inspired by the concept,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 96–102
August 1–6, 2021.

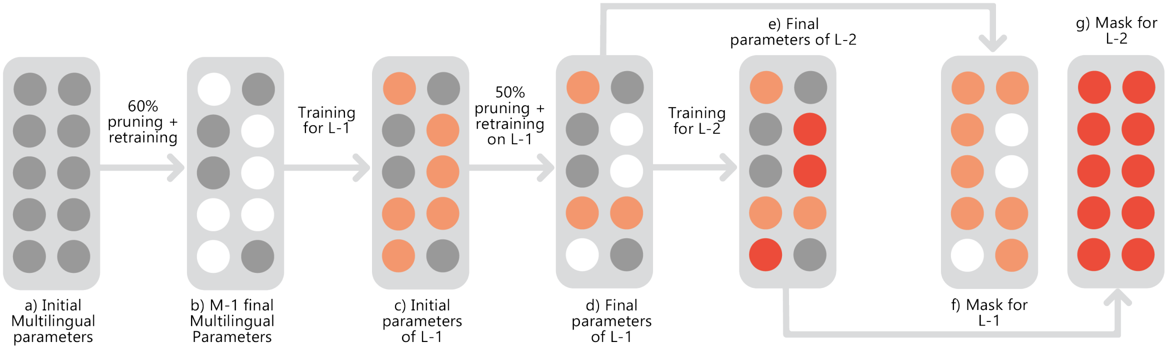

Figure 1: (Better seen in colour.) Illustration of the evolution of model parameters. (a) shows the multilingual parameters in grey. Through 60% pruning and retraining, we arrive at (b), here white represents the free weights with value=0. The surviving weights in grey will be fixed for the rest of the method. Now, we train the free parameters on the first bilingual pair (L-1) and arrive at (c), which represents the initial parameters of L-1 in orange, and share weights with the previously trained multilingual parameters in grey. Again, with 50% pruning and retraining on the current L-1 specific weights in orange, we get the final parameters for L-1 shown in (d) and extract the final mask for L-1 in (f). We repeat the same procedure for all the bilingual pairs and extract the masks for each pair.

we show that an MNMT Transformer model can be heavily pruned and the freed up parameters can be retrained to improve bilingual performance, while retaining the multilinguality.

Adapting multilingual model to a new language pair and domain adaptation: Prior works on adaptation (Neubig and Hu, 2018; Variš and Bojar, 2019; Stickland et al., 2020; Escolano et al., 2020; Akella et al., 2020; Bapna and Firat, 2019; Philip et al., 2020; Zhang et al., 2020) aims at improving language specific performance by either fine-tuning the same MNMT model or adding language specific modules. While being effective, these methods either lose their multilinguality or introduce additional parameters. Sharing the same objective, we propose a method to adapt an MNMT, without adding language-specific modules, while retaining the multilinguality at the same time. Another line of work (Thompson et al., 2018; Wuebker et al., 2018), proposed training of subnetworks and freezing the rest for domain adaptation.

3 Method

The central idea of our method is to use magnitude pruning to free up parameters in the model and learn bilingual specific representations. Figure 1 depicts the evolution of model weights during the training procedure, with (a) representing the initial multilingual weights in grey. We prune away a fraction of parameters using the one-shot magnitude pruning technique (Han et al., 2015), which results in a compressed multilingual representation. We further train the survived multilingual weights for a few more epochs on the multilingual dataset to compensate for extreme pruning, now the multilingual parameters will remain fixed. Then, we use the free parameters to learn the first language-specific representations. We select the first bilingual dataset and train the free parameters. Next we again prune a fraction of weights from the current bilingual parameters only, to accommodate more bilingual representations. We repeat the same procedure for all the existing bilingual pairs. A point to note is that during a forward pass data flows through all the shared and specific weights, while during the backward pass only the current bilingual-specific parameters get updated. Hence, the accuracy is retained for all the previously trained bilingual pairs and it enables a high degree of sharing and specificity at the same time.

Pruning Approach: We perform magnitude pruning (Han et al., 2015) over the weights of all layers. For simplicity, we do not use the more sophisticated pruning methods (Frankle and Carbin, 2019; Michel et al., 2019; Voita et al., 2019). We do not perform pruning over biases and layer normalization parameters, since they correspond to less than 1% of the total parameters, which is insignificant. Also, we do not prune the embeddings, as they are data specific parameters. All are kept fixed after training the multilingual model.

Inference: After finishing the training for each bilingual pair, we get the final mask over all the parameters of the model. Values of the mask range from $1 \rightarrow N$, where N is the total number of bilingual pairs. Each model parameter is masked according to the bilingual pair of interest. To predict a translation for the t^{th} pair, all the parameters learned for languages $1 \rightarrow t$ will be used, as shown in figure 1(f) and (g).

4 Experiments

4.1 Datasets

We use the TED talks (Qi et al., 2018) in all our experiments, and all the numbers are BLEU (Papineni et al., 2002) scores over the test set[1]. Here we have chosen to train on 8 English centring language pairs[2] en-xx covering a spectrum of sizes from high resource *Ar (Arabic)*, 214K to low resource *Be (Belarusian)*, 4.5K.

4.2 Training

Architecture: We use Transformer architecture (Vaswani et al., 2017), implemented in `fairseq` (Ott et al., 2019), which was modified to include the pruning and masking modules. We train a joint BPE model (Sennrich et al., 2016) on all languages to the vocabulary size of 40K. The Transformer (Vaswani et al., 2017) architecture used in this work[3] has 8 attention heads, 6 encoder and decoder layers, an embedding size of 512, and a feed-forward dimension of 2048. We set the dropout to 0.3.

MNMT Training: We train a standard MNMT model following similar settings as Johnson et al. (2017). A single many-to-many model is trained on all the English-centric data, using a source-side control token to indicate the target language. We use Adam (Kingma and Ba, 2015) with an inverse square root schedule, with 4500 warm-up updates and a maximum learning rate of 0.0003. We set the maximum batch size per GPU to 3050 tokens and train on 4 GPUs. Like Arivazhagan et al. (2019), to avoid the size imbalance, we use the temperature-based sampling strategy with $T = 5$. The MNMT is trained for 40 epochs over 8 English-centric language pairs, i.e., 16 directions. As shown in table 1, we train a strong parent MNMT baseline.

Pruning MNMT: We prune 50% of parameters from a fully converged MNMT model, and retrain the surviving parameters on the same multilingual dataset for ten more epochs, to compensate for the lost parameters.

Adapting MNMT to bilingual specific representations: After pruning the MNMT model, we select each bi-direction datasets (en-xx and xx-en) in the descending order of dataset sizes. We use the original source side control token, reset the learning rate scheduler and train all the free parameters for 20 epochs. Then, we prune 75% of parameters from the current bilingual specific parameters and retrain for ten more epochs to compensate for heavy pruning.

Pruning ratios are decided based on the trade off between the accuracy lost and the space left to adapt all the languages. We prune 50-70% of parameters from the parent MNMT and observe that it leads to a drop of 0.29-1.98 Bleu score. Therefore, we select 50% to be the first pruning ratio, and is kept constant in all the experiments. The second pruning ratio is kept 75% such that the last language pairs get at least 2-5% of parameters. More variations in the second pruning ratio is demonstrated in section 5.4.

5 Results and Discussions

5.1 Overcoming interference for high resource pairs:

In table 1, we present a comparative study of a high resource language scenario, severely affected by negative interference. Adapted MNMT outperforms the parent MNMT on all the 8 directions, with an average improvement of +1.40 on xx-en, and +1.32 on en-xx directions, and closes the gap with high performing bilingual baselines.

Analysing model capacity and negative interference: Now, we expound on the problems of model capacity and interference. As shown in table 1, pruning 50% of parameters from the parent MNMT model leads to an average loss of just 0.29 BLEU points. This observation confirms, that there exists large redundancies even in a 9-language MNMT model. The drop in the performance of an MNMT over its counterpart bilingual models is loosely associated with the lack of capacity. As can be seen in figure 2, by using only a fraction of parameters for each bilingual pair, we can significantly improve the performance over the parent MNMT. Our results demonstrate the ability of parameter-free adaptation to fight negative interference, and improve the performance of severely affected high resource language pairs.

---

[1] Scores reported are SacreBLEU (Post, 2018)

[2] `ar, az, be, de, gl, he, it, sk`

[3] `transformer in fairseq`

		xx→ en				*en→ xx*			
		Ar	*De*	*He*	*It*	*Ar*	*De*	*He*	*It*
(1)	Aharoni et al. (2019)	27.84	30.50	34.37	33.64	12.95	23.31	23.66	30.33
	Philip et al. (2020)	32.99	37.36	39.00	39.73	**17.22**	**29.94**	**27.47**	35.42
	Our Bilingual	**33.11**	**39.01**	**39.11**	**41.40**	16.79	29.73	26.80	**36.23**
(2)	Aharoni et al. (2019)	28.32	32.97	33.18	35.14	14.25	27.95	24.16	33.26
	Philip et al. (2020)	30.68	36.53	36.00	38.77	15.40	28.60	24.53	34.02
	Parent MNMT	31.33	37.13	36.86	39.54	15.71	26.32	24.60	33.91
	50% Pruned MNMT	30.84	37.10	36.29	39.44	15.41	26.20	24.06	33.70
	Adapted MNMT	32.68	38.41	38.31	41.04	16.72	27.63	25.76	35.76

Table 1: BLEU scores of our models on the TED test sets compared to the literature, (1) - Bilingual baselines. (2) - Multilingual models scores. Here Aharoni et al. (2019) and Philip et al. (2020) are trained on 59 and 20 languages respectively. Parent MNMT is our multilingual model trained till convergence on 9 languages. 50% pruned MNMT is the compressed parent MNMT. Adapted MNMT is the proposed model.

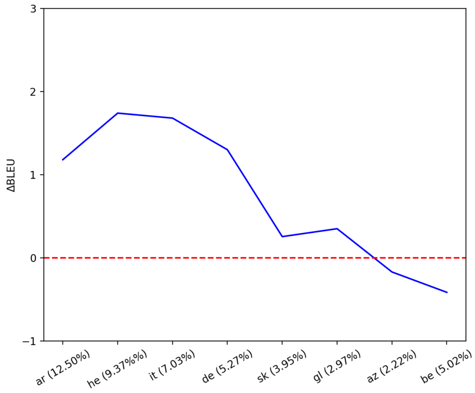

Figure 2: Absolute difference in the BLEU scores, with the parent MNMT, for 8 bilingual pairs. Each bilingual pair is the average over both the English-centric directions. The languages are arranged in the exact order of the training sequence. Numbers on the x-axis are percentages of the bilingual specific parameters used.

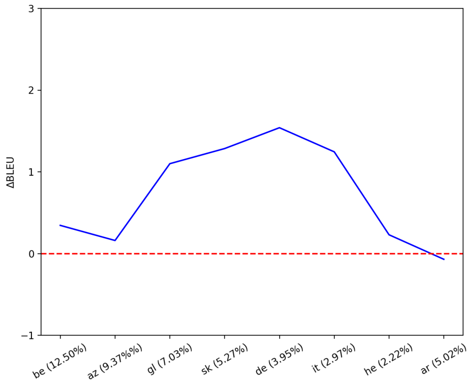

Figure 3: Same as figure 2, trained in the reverse order

5.2 Analysing differences in the adaptation of high and low resource pairs:

To understand the impact of parameter-free adaptation on both the high and low resource language pairs in an unbiased setting. We train two models in opposite orders of adding bilingual pairs. First, we train in the order of high to low resource languages (Ar to Be). Second, we train in the order of low to high resource languages (Be to Ar). Now, we assign the same proportion of parameters, to the high and low resource languages (Ar, He) in case 1, and (Be, Az) in case 2 respectively. As evident from figure 2 and 3, the improvements in Ar and He in case 1 is significantly more, than the improvement in Be and Az in case 2. This observation agrees with the fact that negative interference severely affects the high resource languages in an MNMT, and it needs adaptation to be improved. But, the performance of low resource languages in an MNMT, is already near saturation due to the positive transfer from high resource languages. Hence, to extract the most out of parameter-free adaptation, it is better to prune and retrain the network in the order of high to low resource languages. This assigns high proportion of parameters to high resource pairs, to effectively overcome negative interference.

5.3 Zero-shot Translation:

Zero-shot translation in the context of MNMT, refers to inference between pairs that are not seen directly during the training phase xx-xx. We show that we retain this important ability in our adapted MNMT. Adapted MNMT consists of 50% pruned MNMT weights and 50% language specific weights. The pruned MNMT weights are used to evaluate on zero-shot pairs, just like a traditional MNMT by appending the source side language control token

		xx→ *en*				*en*→ *xx*			
		Ar	*He*	*It*	*De*	*Ar*	*He*	*It*	*De*
(Bilingual)	Full-FT	33.89	39.66	41.64	40.00	17.38	27.50	36.72	29.87
(Multilingual)	Ar only	33.01	-	-	-	16.80	-	-	-
	Ar-He	33.21	38.26	-	-	16.72	25.52	-	-
	Ar-He-It	32.99	38.45	41.03	-	16.61	26.00	35.56 -	
	Ar-He-It-De	32.68	38.43	41.14	38.39	16.72	25.89	36.08	27.25

Table 2: Full-FT represents the bilingual models derived from finetuning the full parent MNMT. Rest are the adapted MNMTs adapted over 50% free parameters of the pruned MNMT. 1) Ar only with 50% parameters, 2) Ar, He with 25% each, 3) Ar, He, It with 16.6% each, and 4) Ar, He, It, De with 12.5% each.

Johnson et al. (2017). As shown in figure 4, adapted MNMT performs as good as the parent MNMT on all the 56, xx-xx directions even with only 50% of the total parameters.

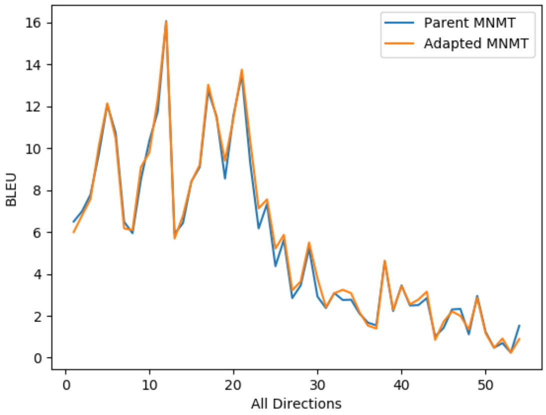

Figure 4: Absolute BLEU scores for the parent and the adapted MNMT on all the 56 zero-shot xx-xx pairs arranged from high to low resource.

5.4 Adapting to a subset of languages and retaining the multilinguality:

Due to limited and fixed number of parameters, we cannot adapt to arbitrary number of languages. However, this framework allows high flexibility in adapting the parent MNMT to only the languages of interest, while retaining the multilinguality simultaneously. We adapt the parent MNMT to four models: 1) Ar, 2) Ar, He, 3) Ar, He, It and, 4) Ar, He, It, De. This way, we can assign all the free parameters to only the languages of interest and increase their capacities. The first pruning ratio is set to 50% for all four models. The second pruning ratio is set such that each language receives equal proportion of parameters. From the results in table 2, we observe that assigning more parameters improve the performance marginally. The four adapted MNMTs have similar performances, even with a significant difference in the proportion of parameters assigned for each language. The 4^{th} model, with only 12.5% parameters reserved for Ar, performs competitively with the 1^{st} model with 50% parameters for Ar. This implies, that a small fraction of parameters can effectively overcome negative interference, hence allowing space to adapt to multiple languages. To infer on the remaining languages which are not adapted, we can use 50% pruned MNMT weights, as done for zero-shot translation in the previous section, hence retaining the multilinguality.

In table 2, we also compare the results of the four adapted MNMTs, with naive finetuning of the full parent MNMT to bilingual pairs (Full-FT). The difference between naive finetuning and the proposed adaptation approach is that the former uses all the 100% of model parameters and the embeddings to adapt to a single bilingual pair, thus the multilinguality is lost. While in our approach, the pruned MNMT weights and the embeddings are fixed, and we only retrain the free parameters very efficiently, allowing to adapt to multiple languages. As can be seen in table 2, adapted MNMTs perform competitively with Full-FT while retaining the multilinguality.

6 Conclusion

We investigate the problems of model capacity and negative interference in multilingual neural machine translation. We show that even a 9 language MNMT has a large proportion of redundant parameters, which are efficiently retrained to overcome interference. We propose a parameter-free adaptation strategy. Where, we use iterative pruning and retraining to improve bilingual representations, without any additional parameters. We hope that our work will attract more attention to practical and efficient ways of adapting an MNMT.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Kartheek Akella, Sai Himal Allu, S. Ragupathi, Aman Singhal, Z. Khan, Vinay P. Namboodiri, and C. Jawahar. 2020. Exploring pair-wise nmt for indian languages. In *arXiv preprint arXiv:2012.05786*. ArXiv.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. In *arXiv preprint arXiv:1907.05019*. ArXiv.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Carlos Escolano, Marta R. Costa-jussà, José A. R. Fonollosa, and M. Artetxe. 2020. Multilingual machine translation: Closing the gap between shared and language-specific encoder-decoders. In *arXiv preprint arXiv:2004.06575*. ArXiv.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4652–4662.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. In *arXiv preprint arXiv:1606.09282*. ArXiv.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7765–7773. IEEE Computer Society.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2020. Recipes for adapting pre-trained monolingual and multilingual models to machine translation. In *arXiv preprint arXiv:2004.14911*. ArXiv.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*.

Brian Thompson, Huda Khayrallah, Antonios Anastasopoulos, Arya D. McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 124–132, Brussels, Belgium. Association for Computational Linguistics.

Amal Rannen Triki, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. 2017. Encoder based lifelong learning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1329–1337. IEEE Computer Society.

Dušan Variš and Ondřej Bojar. 2019. Unsupervised pretraining for neural machine translation using elastic weight consolidation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 130–135, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Kaiser Lukasz, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 881–886, Brussels, Belgium. Association for Computational Linguistics.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

SyGNS: A Systematic Generalization Testbed Based on Natural Language Semantics

Hitomi Yanaka[1,2], **Koji Mineshima**[3], **and Kentaro Inui**[4,2]
[1]The University of Tokyo, [2]RIKEN, [3]Keio University, [4]Tohoku University
`hyanaka@is.s.u-tokyo.ac.jp`, `minesima@abelard.flet.keio.ac.jp`,
`inui@ecei.tohoku.ac.jp`

Abstract

Recently, deep neural networks (DNNs) have achieved great success in semantically challenging NLP tasks, yet it remains unclear whether DNN models can capture *compositional* meanings, those aspects of meaning that have been long studied in formal semantics. To investigate this issue, we propose a Systematic Generalization testbed based on Natural language Semantics (SyGNS), whose challenge is to map natural language sentences to multiple forms of scoped meaning representations, designed to account for various semantic phenomena. Using SyGNS, we test whether neural networks can systematically parse sentences involving novel combinations of logical expressions such as quantifiers and negation. Experiments show that Transformer and GRU models can generalize to unseen combinations of quantifiers, negations, and modifiers that are similar to given training instances in form, but not to the others. We also find that the generalization performance to unseen combinations is better when the form of meaning representations is simpler. The data and code for SyGNS are publicly available at `https://github.com/verypluming/SyGNS`.

1 Introduction

Deep neural networks (DNNs) have shown impressive performance in various language understanding tasks (Wang et al., 2019a,b, i.a.), including semantically challenging tasks such as Natural Language Inference (NLI; Dagan et al., 2013; Bowman et al., 2015). However, a number of studies to probe DNN models with various NLI datasets (Naik et al., 2018; Dasgupta et al., 2018; Yanaka et al., 2019; Kim et al., 2019; Richardson et al., 2020; Saha et al., 2020; Geiger et al., 2020) have reported that current DNN models have some limitations to generalize to diverse semantic phenomena, and it is still not clear whether DNN models obtain the ability to capture *compositional* aspects of meaning in natural language.

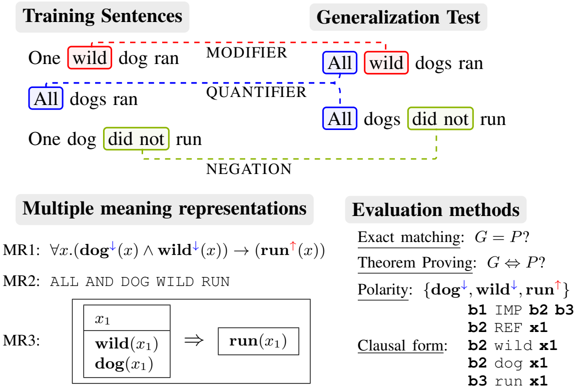

Figure 1: Illustration of our evaluation protocol using SyGNS. The goal is to map English sentences to meaning representations. The generalization test evaluates novel combinations of operations (modifier, quantifier, negation) in the training set. We use multiple meaning representations and evaluation methods.

There are two issues to consider here. First, recent analyses (Talmor and Berant, 2019; Liu et al., 2019; McCoy et al., 2019) have pointed out that the standard paradigm for evaluation, where a test set is drawn from the same distribution as the training set, does not always indicate that the model has obtained the intended generalization ability for language understanding. Second, the NLI task of predicting the relationship between a premise sentence and an associated hypothesis without asking their semantic interpretation tends to be *black-box*, in that it is often difficult to isolate the reasons why models make incorrect predictions (Bos, 2008).

To address these issues, we propose SyGNS (pronounced as *signs*), a Systematic Generalization testbed based on Natural language Semantics. The goal is to map English sentences to various meaning representations, so it can be taken as a sequence-to-sequence semantic parsing task.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 103–119
August 1–6, 2021. ©2021 Association for Computational Linguistics

Figure 1 illustrates our evaluation protocol using SyGNS. To address the first issue above, we probe the generalization capability of DNN models on two out-of-distribution tests: *systematicity* (Section 3.1) and *productivity* (Section 3.2), two concepts treated as hallmarks of human cognitive capacities in cognitive sciences (Fodor and Pylyshyn, 1988; Calvo and Symons, 2014). We use a train-test split controlled by each target concept and train models with a minimally sized training set (Basic set) involving primitive patterns composed of semantic phenomena such as quantifiers, modifiers, and negation. If a model learns different properties of each semantic phenomenon from the Basic set, it should be able to parse a sentence with novel combination patterns. Otherwise, a model has to memorize an exponential number of combinations of linguistic expressions.

To address the second issue, we use multiple forms of meaning representations developed in formal semantics (Montague, 1973; Heim and Kratzer, 1998; Jacobson, 2014) and their respective evaluation methods. We use three *scoped* meaning representation forms, each of which preserves the same semantic information (Section 3.3). In formal semantics, it is generally assumed that scoped meaning representations are standard forms for handling diverse semantic phenomena such as quantification and negation. Scoped meaning representations also enable us to evaluate the compositional generalization ability of the models to capture semantic phenomena in a more fine-grained way. By decomposing an output meaning representation into constituents (e.g., words) in accordance with its structure, we can compute the matching ratio between the output representation and the gold standard representation. Evaluating the models on multiple meaning representation forms also allows us to explore whether the performance depends on the complexity of the representation forms.

This paper provides three main contributions. First, we develop the SyGNS testbed to test model ability to systematically transform sentences involving linguistic phenomena into multiple forms of scoped meaning representations. The data and code for SyGNS are publicly available at `https://github.com/verypluming/SyGNS`. Second, we use SyGNS to analyze the systematic generalization capacity of two standard DNN models: Gated Recurrent Unit (GRU) and Transformer. Experiments show that these models can generalize to unseen combinations of quantifiers, negations, and modifiers to some extent. However, the generalization ability is limited to the combinations whose forms are similar to those of the training instances. In addition, the models struggle with parsing sentences involving nested clauses. We also show that the extent of generalization depends on the choice of primitive patterns and representation forms.

2 Related Work

The question of whether neural networks obtain the systematic generalization capacity has long been discussed (Fodor and Pylyshyn, 1988; Marcus, 2003; Baroni, 2020). Recently, empirical studies using NLI tasks have revisited this question, showing that current models learn undesired biases (Glockner et al., 2018; Poliak et al., 2018; Tsuchiya, 2018; Geva et al., 2019; Liu et al., 2019) and heuristics (McCoy et al., 2019), and fail to consistently learn various inference types (Rozen et al., 2019; Nie et al., 2019; Yanaka et al., 2019; Richardson et al., 2020; Joshi et al., 2020). In particular, previous works (Goodwin et al., 2020; Yanaka et al., 2020; Geiger et al., 2020; Yanaka et al., 2021) have examined whether models learn the systematicity of NLI on monotonicity and veridicality. While this line of work has shown certain limitations of model generalization capacity, it is often difficult to figure out why the NLI model fails and how to improve it, partly because NLI tasks depend on multiple factors, including semantic interpretation of target phenomena and acquisition of background knowledge. By focusing on semantic parsing rather than NLI, one can probe to what extent models systematically interpret the meaning of sentences according to their structures and the meanings of their constituents.

Meanwhile, datasets for analysing the compositional generalization ability of DNN models in semantic parsing have been proposed, including SCAN (Lake and Baroni, 2017; Baroni, 2020), CLUTRR (Sinha et al., 2019), and CFQ (Keysers et al., 2020). For example, the SCAN task is to investigate whether models trained with a set of primitive instructions (e.g., *jump* → `JUMP`) and modifiers (e.g., *walk twice* → `WALK WALK`) generalize to new combinations of primitives (e.g., *jump twice* → `JUMP JUMP`). However, these datasets deal with artificial languages, where the variation of linguistic expressions is limited, so it is not clear to what extent the models systematically interpret various semantic phenomena in natural language, such as quantification and negation.

Regarding the generalization capacity of DNN models in natural language, previous studies have focused on syntactic and morphological generalization capacities such as subject-verb agreement tasks (Linzen et al., 2016; Gulordava et al., 2018; Marvin and Linzen, 2018, i.a.). Perhaps closest to our work is the COGS task (Kim and Linzen, 2020) for probing the generalization capacity of semantic parsing in a synthetic natural language fragment. For instance, the task is to see whether models trained to parse sentences where some lexical items only appear in subject position (e.g., *John ate the meat*) can generalize to structurally related sentences where these items appear in object position (e.g., *The kid liked John*). In contrast to this work, our focus is more on semantic parsing of sentences with logical and semantic phenomena that require scoped meaning representations. Our study also improves previous work on the compositional generalization capacity in semantic parsing in that we compare three types of meaning representations and evaluate them at multiple levels, including logical entailment, polarity assignment, and partial clause matching (Section 3.3).

3 Overview of SyGNS

We use two evaluation concepts to assess the systematic capability of models: systematicity (Section 3.1) and productivity (Section 3.2). In evaluating these two concepts, we use synthesized pairs of sentences and their meaning representations to control a train-test split (Section 3.4). The main idea is to analyze models trained with a minimum size of a training set (Basic set) involving primitive patterns composed of various semantic phenomena; if a model systematically learns primitive combination patterns in the Basic set, it should parse a new sentence with different combination patterns. We target three types of scoped meaning representations and use their respective evaluation methods, according to the function and structure of each representation form (Section 3.3).

3.1 Systematicity

Pattern		Sentence
		Train
Primitive quantifier		**One** tiger ran
Basic 1	EXI	**A** tiger ran
	NUM	**Two** tigers ran
	UNI	**Every** tiger ran
Basic 2	ADJ	**One** *small* tiger ran
	ADV	**One** tiger ran *quickly*
	CON	**One** tiger ran *or came*
		Test
EXI+ADJ		**A** *small* tiger ran
NUM+ADV		**Two** tigers ran *quickly*
UNI+CON		**Every** tiger ran *or came*

Table 1: Training and test instances for systematicity.

Table 1 illustrates how we test systematicity, i.e., the capacity to interpret novel combinations of primitive semantic phenomena. We generate Basic set 1 by combining various quantifiers with sentences without modifiers. We also generate Basic set 2 by setting an arbitrary quantifier (e.g., *one*) to a primitive quantifier and combining it with various types of modifiers. We then evaluate whether models trained with Basic sets 1 and 2 can parse sentences involving unseen combinations of quantifiers and modifiers. We also test the combination of quantifiers and negation in the same manner; the detail is given in Appendix D.

To provide a controlled setup, we use three quantifier types: existential quantifiers (EXI), numerals (NUM), and universal quantifiers (UNI). Each type has two patterns: *one* and *a* for EXI, *two* and *three* for NUM, and *all* and *every* for UNI. We consider three settings where the primitive quantifier is set to the type EXI, NUM, or UNI.

For modifiers, we distinguish three types — adjectives (ADJ), adverbs (ADV), and logical connectives (CON; conjunction *and*, disjunction *or*) — and ten patterns for each. Note that each modifier type differs in its position; an adjective appears inside a noun phrase (e.g., one *small* tiger), while an adverb (e.g., *quickly*) and a coordinated phrase with a logical connective (e.g., *or came*) appears at the end of a sentence. Although Table 1 only shows the pattern generated by the primitive quantifier *one* and the noun *tiger*, the noun can be replaced with ten other nouns (e.g., *dog*, *cat*, etc.) in each setting. See Appendix A for more details on the fragment of English considered here.

3.2 Productivity

Productivity refers to the capacity to interpret an indefinite number of sentences with recursive operations. To test productivity, we use embedded relative clauses, which interact with quantifiers to gen-

Pattern		Sentence
Train (Basic 1: depth 0, Basic 2: depth 1)		
Basic 1	NON	Two dogs loved Ann
Basic 2	PER	Bob liked a bear [that chased all polite cats]
	CEN	Two dogs [that all cats kicked] loved Ann
Test (examples: depth 2)		
PER+PER		Bob liked a bear [that chased all polite cats [that loves Ann]]
PER+CEN		Two dogs [that a bear [that chased all polite cats] kicked] loved Ann

Table 2: Training and test instances for productivity.

erate logically complex sentences. Table 2 shows examples. We provide two Basic sets; Basic set 1 consists of sentences without embedded clauses (NON) and Basic set 2 consists of sentences with a single embedded clause, which we call sentences with depth one. We then test whether models trained with Basic sets 1 and 2 can parse a sentence involving deeper embedded clauses, i.e., sentences whose depth is two or more. As Table 2 shows, we consider both peripheral-embedding (PER) and center-embedding (CEN) clauses.

3.3 Meaning representation and evaluation

Overview To evaluate generalization capacity in semantic parsing at multiple levels, we use three types of scoped meaning representations: (i) First-Order Logic (FOL) formulas, (ii) Discourse Representation Structures (DRSs; Kamp and Reyle, 1993), and (iii) Variable-Free (VF) formulas (Baader et al., 2003; Prat-Hartmann and Moss, 2009). DRSs can be converted to *clausal forms* (van Noord et al., 2018a) for evaluation. For instance, the sentence (1) is mapped to the FOL formula in (2), the DRS in (3a), its clausal form in (3b), and the VF formula in (4).

(1) One white dog did not run.

(2) $\exists x_1.(\mathbf{white}(x_1) \wedge \mathbf{dog}(x_1) \wedge \neg\mathbf{run}(x_1))$

(3) a. $\begin{array}{|l|} \hline x_1 \\ \hline \mathbf{white}(x_1) \\ \mathbf{dog}(x_1) \\ \neg\ \boxed{\mathbf{run}(x_1)} \\ \hline \end{array}$

b. **b1** REF **x1**
b1 white **x1**
b1 dog **x1**
b1 NOT **b2**
b2 run **x1**

(4) EXIST AND WHITE DOG NOT RUN

Using these multiple forms enables us to analyze whether the difficulty in semantic generalization depends on the format of meaning representations.

Previous studies for probing generalization capacity in semantic parsing (e.g., Lake and Baroni, 2017; Sinha et al., 2019; Keysers et al., 2020; Kim and Linzen, 2020) use a fixed type of meaning representation, with its evaluation method limited to the exact-match percentage, where an output is considered correct only if it exactly matches the gold standard. However, this does not properly assess whether models capture the structure and function of meaning representation. First, exact matching does not directly take into account whether two meanings are logically equivalent (Blackburn and Bos, 2005): for instance, schematically two formulas $A \wedge B$ and $B \wedge A$ are different in form but have the same meaning. Relatedly, scoped meaning representations for natural languages can be made complex by including parentheses and variable renaming mechanism (the so-called α-conversion in λ-calculus). For instance, we want to identify two formulas which only differ in variable naming, e.g., $\exists x_1.F(x_1)$ and $\exists x_2.F(x_2)$. It is desirable to compare exact matching with alternative evaluation methods, and to consider alternative meaning representations that avoid these problems. Having this background in mind, below we will describe each type of meaning representation in detail.

FOL formula FOL formulas are standard forms in formal and computational semantics (Blackburn and Bos, 2005; Jurafsky and Martin, 2009), where content words such as nouns and verbs are represented as predicates, and function words such as quantifiers, negation, and connectives are represented as logical operators with scope relations (cf. the example in (2)). To address the issue on evaluation, we consider two ways of evaluating FOL formulas in addition to exact matching: (i) automated theorem proving (ATP) and (ii) monotonicity-based polarity assignment.

First, FOL formulas can be evaluated by checking the logical entailment relationships that directly consider whether two formulas are logically equivalent. Thus we evaluate predicted FOL formulas by using ATP. We check whether a gold formula G entails prediction P and vice versa, using an off-the-shelf FOL theorem prover[1]. To see the logical relationship between G and P, we measure the accuracy for unidirectional and bidirectional

---

[1] We use a state-of-the-art FOL prover Vampire available at `https://github.com/vprover/vampire`

One dog$^{\uparrow}$ ran$^{\uparrow}$:	$\exists x.(\mathbf{dog}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x))$
All dogs$^{\downarrow}$ ran$^{\uparrow}$:	$\forall x.(\mathbf{dog}^{\downarrow}(x) \rightarrow \mathbf{run}^{\uparrow}(x))$
All dogs$^{\downarrow}$ did not run$^{\downarrow}$:	$\forall x.(\mathbf{dog}^{\downarrow}(x) \rightarrow \neg\mathbf{run}^{\downarrow}(x))$

Table 3: Examples of monotonicity-based polarity assignments for FOL formulas.

entailment: $G \Rightarrow P$, $G \Leftarrow P$, and $G \Leftrightarrow P$.

Second, the polarity of each content word appearing in a sentence can be extracted from the FOL formula using its monotonicity property (van Benthem, 1986; MacCartney and Manning, 2007). This enables us to analyze whether models can correctly capture entailment relations triggered by quantifier and negation scopes. Table 3 shows some examples of monotonicity-based polarity assignments. For example, existential quantifiers such as *one* are upward monotone (shown as ↑) with respect to the subject NP and the VP, because they can be substituted with their hypernyms (e.g., *One dog ran* ⇒ *One animal moved*). These polarities can be extracted from the FOL formula because $\exists$ and $\wedge$ are upward monotone operators in FOL. Universal quantifiers such as *all* are downward monotone (shown as ↓) with respect to the subject NP and upward monotone with respect to the VP. Expressions in downward monotone position can be substituted with their hyponymous expressions (e.g., *All dogs ran* ⇒ *All white dogs ran*). The polarity can be reversed by embedding another downward entailing context like negation, so the polarity of *run* in the third case in Table 3 is flipped to downward monotone.[2] For evaluation based on monotonicity, we extract a polarity for each content word in a gold formula and a prediction and calculate the F-score for each monotonicity direction (upward and downward).

DRS A DRS is a form of scoped meaning representations proposed in Discourse Representation Theory, a well-studied formalism in formal semantics (Kamp and Reyle, 1993; Asher, 1993; Muskens, 1996). By translating a box notation as in (3a) to the clausal form as in (3b), one can evaluate DRSs by COUNTER[3], which is a standard tool for evaluating neural DRS parsers (Liu et al., 2018; van Noord et al., 2018b). COUNTER searches for the best variable mapping between predicted DRS clauses and gold DRS clauses and calculates an F-score over matching clause, which is similar to SMATCH (Cai and Knight, 2013), an evaluation metric designed for Abstract Meaning Representation (AMR; Banarescu et al., 2013). COUNTER alleviates the process of variable renaming and correctly evaluates the cases where the order of clauses is different from that of gold answers.

VF formula FOL formulas in our fragment have logically equivalent forms in a variable-free format, which does not contain parentheses nor variables as in the example (4). Our format is similar to a variable-free form in Description Logic (Baader et al., 2003) and Natural Logic (Prat-Hartmann and Moss, 2009). VF formulas alleviate the problem of parentheses and variable renaming, while preserving semantic information (cf. Wang et al., 2017). Due to the equivalence with FOL formulas, it is possible to extract polarities from VF formulas. See Appendix A for more examples of VF formulas.

3.4 Data generation

To provide synthesized data, we generate sentences using a context-free grammar (CFG) associated with semantic composition rules in the standard λ-calculus (see Appendix A for details). Each sentence is mapped to an FOL formula and VF formula by using the semantic composition rules specified in the CFG. DRSs are converted from the generated FOL formulas using the standard mapping (Kamp and Reyle, 1993). To generate controlled fragments for each train-test split, the CFG rules automatically annotate the types of semantic phenomena involved in sentences generated. We annotate seven types: the positions of quantifiers (subject or object), negation, adjectives, adverbs, conjunction, disjunction, and embedded clause types (peripheral or center embedding).

To test systematicity, we generate sentences using the CFG, randomly select 50,000 examples, and then split them into 12,000 training examples and 38,000 test examples according to a primitive quantifier. To test productivity, we apply up to four recursive rules and randomly select 20,000 examples for each depth.

4 Experiments and Analysis

Using SyGNS, we test the performance of Gated Recurrent Unit (GRU; Cho et al., 2014) and Transformer (Vaswani et al., 2017) in an encoder-decoder setup. These are widely used models

---

[2] We follow the surface order of NPs and take it that the subject NP always take scope over the VP.

[3] `https://github.com/RikVN/DRS_parsing`

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: existential quantifier *one*								
Exi	96.1	99.5	99.9	99.7	99.9	99.8	99.9	100.0
Num	7.6	12.7	86.0	37.0	18.1	96.9	99.7	20.7
Uni	3.1	4.4	56.8	39.5	8.3	2.2	74.2	17.7
Valid	98.2	99.7	100.0	99.6	100.0	100.0	100.0	100.0
primitive quantifier: numeral *two*								
Exi	11.6	42.1	91.4	45.3	34.0	84.5	98.3	10.5
Num	59.5	83.6	98.7	42.8	99.9	97.4	99.8	80.9
Uni	2.5	1.8	68.6	39.2	0.0	0.1	72.3	90.9
Valid	84.3	99.7	100.0	98.9	100.0	100.0	100.0	100.0
primitive quantifier: universal quantifier *every*								
Exi	1.6	0.3	43.8	61.3	2.1	0.2	70.8	20.8
Num	1.4	0.3	75.9	69.3	0.1	0.1	76.8	99.7
Uni	33.8	96.5	99.4	100.0	100.0	100.0	100.0	99.9
Valid	93.4	100.0	100.0	99.3	100.0	100.0	100.0	99.9
primitive quantifiers: *one*, *two*, *every*								
Exi	99.7	99.0	100.0	100.0	100.0	100.0	100.0	100.0
Num	91.2	96.4	99.2	99.3	100.0	100.0	100.0	100.0
Uni	95.7	97.6	99.4	100.0	99.9	100.0	100.0	100.0
Valid	98.4	100.0	100.0	99.3	100.0	100.0	100.0	100.0

Table 4: Accuracy by quantifier type. “DRS (cnt)” columns show the accuracy of predicted DRSs by COUNTER, and “Valid” rows show the validation accuracy. Each accuracy is measured by exact matching, except for “DRS (cnt)” columns.

to perform well on hierarchical generalization tasks (McCoy et al., 2018; Russin et al., 2020).

4.1 Experimental setup

In all experiments, we trained each model for 25 epochs with early stopping (patience = 3). We performed five runs and reported their average accuracies. The input sentence is represented as a sequence of words, using spaces as a separator. The maximum input and output sequence length is set to the length of a sequence with maximum depths of embedded clauses. We set the dropout probability to 0.1 on the output and used a batch size of 128 and an embedding size of 256. Since incorporating pre-training would make it hard to distinguish whether the models’ ability to perform semantic parsing comes from training data or from pre-training data, we did not use any pre-training.

For the GRU, we used a single-layer encoder-decoder with global attention and a dot-product score function. Since a previous work (Kim and Linzen, 2020) reported that unidirectional models are more robust regarding sentence structures than bi-directional models, we selected a unidirectional GRU encoder. For the Transformer, we used a three-layer encoder-decoder, a model size of 512, and a hidden size of 256. The number of model parameters was 10M, respectively. See Appendix B for additional training details.

4.2 Results on systematicity

Generalization on quantifiers Table 4 shows the accuracy by quantifier type. When the existential quantifier *one* was the primitive quantifier, the accuracy on the problems involving existential quantifiers, which have the same type as the primitive quantifier, was nearly perfect. Similarly, when the universal quantifier *every* was the primitive quantifier, the accuracy on the problems involving universal quantifiers was much better than that on the problems involving other quantifier types. These results indicate that models can easily generalize to problems involving quantifiers of the same type as the primitive quantifier, while the models struggle with generalizing to the others. We also experimented with larger models and observed the same trend (see Appendix C). The extent of generalization varies according to the primitive quantifier type and meaning representation forms. For example, when the primitive quantifier is the numeral expression *two*, models generalize to problems of VF formulas involving universal quantifiers. This can be explained by the fact that VF formulas involving universal quantifiers like (5b) have a similar form to those involving numerals as in (6b), whereas FOL formulas involving universal quantifiers have a different form from those involving numerals as in (5c) and (6c).

(5) a. All small cats chased Bob
    b. `ALL AND CAT SMALL EXIST BOB CHASE`
    c. $\forall x_1.(\mathbf{cat}(x_1) \wedge \mathbf{small}(x_1) \rightarrow \exists x_2.(\mathbf{bob}(x_2) \wedge \mathbf{chase}(x_1, x_2)))$

(6) a. Two small cats chased Bob
    b. `TWO AND CAT SMALL EXIST BOB CHASE`
    c. $\exists x_1.(\mathbf{two}(x_1) \wedge \mathbf{cat}(x_1) \wedge \mathbf{small}(x_1) \wedge \exists x_2.(\mathbf{bob}(x_2) \wedge \mathbf{chase}(x_1, x_2)))$

We also check the performance when three quantifiers *one*, *two*, and *every* are set as primitive quantifiers. This setting is easier than that for the systematicity in Table 1, since the models are exposed to combination patterns of all the quantifier types and all the modifier types. In this setting, the models achieved almost perfect performance on the test set involving non-primitive quantifiers (*a*, *three*, *all*).

Generalization on modifiers Table 5 shows the accuracy by modifier type where *one* is set to the primitive quantifier (see Appendix C for the results where other quantifier types are set to the primitive quantifier). No matter which quantifier is set as the primitive quantifier, the accuracy for problems involving logical connectives or adverbs is better than those involving adjectives. As in (8), an adjective is placed between a quantifier and a noun, so the position of the noun *dog* with respect to the quantifier *every* in the test set changes from the example in the training (Basic) set in (7). In contrast, adverbs and logical connectives are placed at the end of a sentence, so the position of the noun does not change from the training set, as in (9). This suggests that models can more easily generalize in problems involving unseen combinations of quantifiers and modifiers where the position of the noun is the same between the training and test sets.

(7) Every dog ran   Train (Basic set)
(8) Every large dog ran   Test (ADJ)
(9) Every dog ran and cried   Test (CON)

Table 5 also shows that the accuracy is nearly the same regardless of the existence of negation. Basic set contains examples involving negation, and this indicates that the existence of complex phenomena like negation does not affect generalization performance of models on modifiers so long as such phenomena are included in the training set.

Meaning representation comparison Comparing forms of meaning representations, accuracy by exact matching is highest in the order of VF formulas, DRS clausal forms, and FOL formulas. This indicates that models can more easily generalize to unseen combinations where the form of meaning representation is simpler; VF formulas do not contain parentheses nor variables, DRS clausal forms contain variables but not parentheses, and FOL formulas contain both parentheses and variables.

4.3 Model comparison

Regarding the generalization capacity of models for decoding meaning representations, the left two figures in Figure 2 show learning curves on FOL prediction tasks by quantifier type. While GRU achieved perfect performance on the same quantifier type as the primitive quantifier, where the number of training data is 2,500, Transformer achieved the same performance when the number of training data is 8,000. The right two figures in Figure 2 show learning curves by modifier type. The GRU accuracy is unstable even when the number of training examples is maximal. In contrast, the Transformer accuracy is stable when the number of training data exceeds 8,000. These results indicate that GRU generalizes to unseen combinations of quantifiers and modifiers with a smaller training set than can Transformer, while the Transformer performance is more stable than that of GRU.

ATP-based evaluation Table 6 shows the ATP-based evaluation results on FOL formulas. For combinations involving numerals, both GRU and Transformer achieve high accuracies for $G \Rightarrow P$ entailments but low accuracies for $G \Leftarrow P$ entailments. Since both models fail to output the formulas corresponding to modifiers, they fail to prove $G \Leftarrow P$ entailments. Regarding combinations involving universal quantifiers, the GRU accuracy for both $G \Rightarrow P$ and $G \Leftarrow P$ entailments is low, and the Transformer accuracy for $G \Leftarrow P$ entailments is much higher than that for $G \Rightarrow P$ entailments. As indicated by examples shown in Table 7,

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
ADJ	18.9	20.1	78.1	42.3	26.8	59.1	91.3	27.6
ADJ+NEG	18.8	20.2	80.5	39.7	23.1	59.5	93.6	27.5
ADV	20.1	47.7	87.5	58.4	36.2	67.3	97.6	50.7
ADV+NEG	26.9	62.7	92.7	67.2	50.7	69.4	97.3	62.1
CON	28.9	58.3	84.7	72.9	54.3	66.8	88.3	65.9
CON+NEG	33.6	62.8	86.6	74.9	60.1	65.1	89.9	69.1
Valid	98.2	99.7	100.0	99.6	100.0	100.0	100.0	100.0

Table 5: Accuracy by modifier type (primitive quantifier: existential quantifier *one*). +NEG indicates problems involving negation. Each accuracy is measured by exact matching, except for "DRS (cnt)" columns.

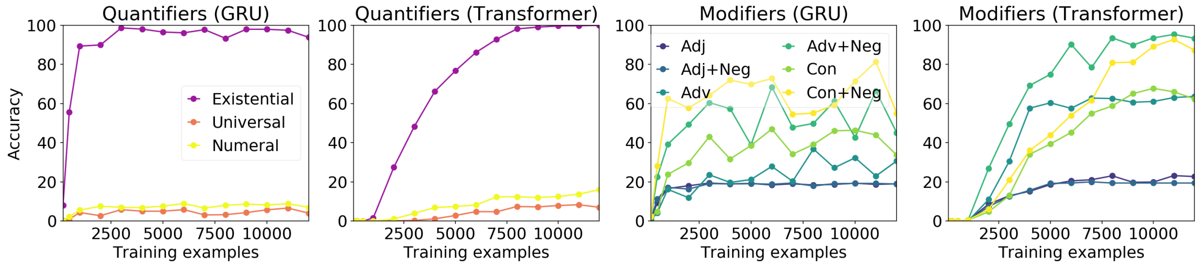

Figure 2: Learning curves on FOL formula generalization tasks (primitive quantifier: *one*).

Test	GRU			Transformer		
	$G \Rightarrow P$	$G \Leftarrow P$	$G \Leftrightarrow P$	$G \Rightarrow P$	$G \Leftarrow P$	$G \Leftrightarrow P$
EXI	99.8	100.0	99.8	100.0	100.0	100.0
NUM	77.1	19.0	10.4	91.3	21.1	12.4
UNI	7.1	18.7	2.7	21.1	83.4	12.3

Table 6: ATP-based evaluation results on FOL formulas (primitive quantifier: *one*).

Input	Every wild cat escaped and ran
Gold	$\forall x.((\mathbf{cat}^{\downarrow}(x) \wedge \mathbf{wild}^{\downarrow}(x)) \rightarrow (\mathbf{escape}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x)))$
GRU	$\forall x.(\mathbf{cat}^{\downarrow}(x) \rightarrow (\mathbf{escape}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x)))$
Trans	$\forall x.(\mathbf{cat}^{\downarrow}(x) \rightarrow \mathbf{wild}^{\uparrow}(x) \wedge (\mathbf{escape}^{\uparrow}(x) \wedge \mathbf{run}^{\uparrow}(x)))$

Table 7: Examples of typical errors.

Test	GRU		Transformer	
	Up	Down	Up	Down
EXI	99.9	100.0	100.0	100.0
NUM	84.8	96.8	88.1	97.5
UNI	90.9	40.7	94.9	73.4

Table 8: Monotonicity-based evaluation results on FOL (primitive quantifier: *one*). "Up" and "Down" columns show upward and downward accuracy, respectively.

GRU tends to fail to output the formula for a modifier, e.g., $\mathbf{wild}(x)$ in this case, while Transformer fails to correctly output the position of the implication ($\rightarrow$). The ATP-based evaluation results reflect such differences between error trends of models in problems involving different forms of quantifiers.

Monotonicity-based evaluation Table 8 shows accuracies for the monotonicity-based polarity assignment evaluation on FOL formulas. The accuracies were higher than those using exact matching (cf. Table 4). Monotonicity-based evaluation captures the polarities assigned to content words even for the problems that exact-matching judges as incorrect because of the differences in form. Table 7 shows examples of predicted polarity assignments. Here both models predicted correct polarities for three content words, $\mathbf{cat}^{\downarrow}$, $\mathbf{escape}^{\uparrow}$, $\mathbf{run}^{\uparrow}$. Exact-matching cannot take into account such partial matching. The downward monotone accuracies for problems involving universal quantifiers are low (40.7 and 73.4 in Table 8). In Table 7, both models failed to predict the downward monotonicity of $\mathbf{wild}^{\downarrow}$. The results indicate that both models struggle with capturing the scope of universal quantifiers. Appendix C shows the evaluation results on the polarities of VF formulas.

4.4 Results on productivity

Table 9 shows very low generalization accuracy for both GRU and Transformer at unseen depths. Although the evaluation results using COUNTER on DRS prediction tasks is much higher than those by exact matching, this is due to the fact that COUNTER uses partial matching; both models tended to correctly predict the clauses in the subject NP that are positioned at the beginning of the

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
Dep2	0.36	0.41	55.5	0.32	0.61	0.61	64.6	0.58
Dep3	0.04	0.07	45.6	0.04	0.11	0.12	46.6	0.12
Dep4	0.00	0.01	38.0	0.00	0.02	0.02	37.6	0.02
Valid	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 9: Accuracy for productivity. “Dep” rows show embedding depths, “DRS (cnt)” columns show accuracy of predicted DRSs by COUNTER, and “Valid” row shows the validation accuracy. Each accuracy is measured by exact matching, except for “DRS (cnt)” columns.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
Dep1	22.1	21.9	81.9	48.9	96.6	77.1	96.5	97.5
Dep2	3.52	3.89	59.1	12.3	76.3	54.6	90.5	85.4
Dep3	0.15	0.12	43.3	0.31	24.9	4.5	70.4	37.0
Dep4	0.08	0.15	37.7	0.46	4.4	1.6	60.3	5.57
Valid	94.3	94.9	100.0	96.0	97.6	98.1	100.0	97.8

Table 10: Evaluation results for systematicity involving embedding quantifiers. “Dep” rows show embedding depths.

sentence (see Appendix E for details).

We checked whether models can generalize to unseen combinations involving embedded clauses when the models are exposed to a part of training instances at each depth. We provide Basic set 1 involving non-embedding patterns like (10), where **Q** can be replaced with any quantifier. This Basic set 1 exposes models to all quantifier patterns. We also expose models to Basic set 2 involving three primitive quantifiers (*one*, *two*, and *every*) at each embedding depth, like (11) and (12). We provide around 2,000 training instances at each depth. We then test models on a test set involving the other quantifiers (*a*, *three*, and *all*) at each embedding depth, like (13) and (14). If models can distinguish quantifier types during training, they can correctly compose meaning representations involving different combinations of multiple quantifiers. Note that this setting is easier than that for productivity in Table 2, in that models are exposed to some instances at each depth.

(10) **Q** dog(s) liked Bob

(11) **One** dog liked Bob [that loved **two** rats]

(12) **One** dog liked Bob [that loved **two** rats [that knew **every** pig]]

(13) **A** dog liked Bob [that loved **three** rats]

(14) **A** dog liked Bob [that loved **three** rats [that knew **all** pigs]]

Table 10 shows that both models partially generalize to the cases where the depth is 1 or 2. However, both models fail to generalize to the cases where the depth is 3 or more. This suggests that even if models are trained with some instances at each depth, the models fail to learn distinctions between different quantifier types and struggle with parsing sentences whose embedding depth is 3 or more.

5 Conclusion

We have introduced an analysis method using SyGNS, a testbed for diagnosing the systematic generalization capability of DNN models in semantic parsing. We found that GRU and Transformer generalized to unseen combinations of semantic phenomena whose meaning representations are similar in forms to those in a training set, while the models struggle with generalizing to the others. In addition, these models failed to generalize to cases involving nested clauses. Our analyses using multiple meaning representation and evaluation methods also revealed detailed behaviors of models. We believe that SyGNS serves as an effective testbed for investigating the ability to capture compositional meanings in natural language.

Acknowledgement

We thank the three anonymous reviewers for their helpful comments and suggestions. This work was partially supported by JSPS KAKENHI Grant Number JP20K19868.

References

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Springer.

Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Marco Baroni. 2020. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B*, 375(1791):20190307.

Johan van Benthem. 1986. *Essays in Logical Semantics*. Springer.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Center for the Study of Language and Information.

Johan Bos. 2008. Let's not argue about semantics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Paco Calvo and John Symons. 2014. *The Architecture of Cognition: Rethinking Fodor and Pylyshyn's Systematicity Challenge*. MIT Press.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. 2018. Evaluating compositionality in sentence embeddings. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 1596–1601.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 163–173, Online. Association for Computational Linguistics.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Emily Goodwin, Koustuv Sinha, and Timothy J. O'Donnell. 2020. Probing linguistic systematicity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1958–1969, Online. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell.

Pauline Jacobson. 2014. *Compositional Semantics: An Introduction to the Syntax/Semantics Interface*. Oxford University Press.

Pratik Joshi, Somak Aditya, Aalok Sathe, and Monojit Choudhury. 2020. TaxiNLI: Taking a ride up the NLU hill. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 41–55, Online. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Brenden M. Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of International Conference on Machine Learning (ICML)*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia. Association for Computational Linguistics.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2171–2179, Minneapolis, Minnesota. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Gary Marcus. 2003. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society, CogSci 2018, Madison, WI, USA, July 25-28, 2018*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius M. E. Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language*, pages 189–224. Reidel, Dordrecht. Reprinted in Richmond H. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, 247–270, 1974, New Haven: Yale University Press.

Reinhard Muskens. 1996. Combining Montague semantics and discourse representation. *Linguistics and philosophy*, 19(2):143–186.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yixin Nie, Yicheng Wang, and Mohit Bansal. 2019. Analyzing compositionality-sensitivity of NLI models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6867–6874.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Ian Prat-Hartmann and Lawrence S. Moss. 2009. Logics for the relational syllogistic. *The Review of Symbolic Logic*, 2(4):647–683.

Kyle Richardson, Hai Hu, Lawrence S. Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ohad Rozen, Vered Shwartz, Roee Aharoni, and Ido Dagan. 2019. Diversify your datasets: Analyzing generalization via controlled variance in adversarial datasets. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 196–205, Hong Kong, China. Association for Computational Linguistics.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.

Swarnadeep Saha, Yixin Nie, and Mohit Bansal. 2020. ConjNLI: Natural language inference over conjunctive sentences. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8240–8252, Online. Association for Computational Linguistics.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jan Van Eijck. 2005. Natural logic for natural language. In *International Tbilisi Symposium on Logic, Language, and Computation*, pages 216–230. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. 2017. Premise selection for theorem proving by deep graph embedding. In *Proceedings of Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2786–2796.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. 2020. Do neural models learn systematicity of monotonicity inference in natural language? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6105–6117, Online. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, and Kentaro Inui. 2021. Exploring transitivity in neural NLI models through veridicality. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 920–934.

A Data generation details

Table 12 shows a set of context-free grammar rules and semantic composition rules we use to generate a fragment of English annotated with meaning representations in the SyGNS dataset. Each grammar rule is associated with two kinds of semantic composition rules formulated in λ-calculus. One is for deriving first-order logic (FOL) formulas, and the other is for deriving variable-free (VF) formulas. For FOL, semantic composition runs in the standard Montagovian fashion where all NPs (including proper nouns) are treated as generalized quantifiers (Heim and Kratzer, 1998; Jacobson, 2014). From FOL formulas, we can extract the *polarity* of each content word using the monotonicity calculus (Van Eijck, 2005). Table 13 shows some examples of polarized FOL formulas. The derivation of VF formulas runs in two steps. To begin with, a sentence is mapped to a variable-free form by semantic composition rules. For instance, the sentence *a small dog did not swim* is mapped to a variable-free formula `EXIST(AND(SMALL,DOG),NOT(SWIM))` by the rules in Table 12. Second, since this form is in prefix notation, all brackets can be eliminated without causing ambiguity. This produces the resulting VF formula `EXIST AND SMALL DOG NOT SWIM`. Some other examples are shown in Table 13. DRSs are converted from FOL formulas in the standard way (Kamp and Reyle, 1993).

B Training details

We implemented the GRU model and the Transformer model using PyTorch. Both models were optimized using Adam (Kingma and Ba, 2015) at an initial learning rate of 0.0005. The hyperparameters (batch size, learning rate, number of epochs, hidden units, and dropout probability) were tuned by random search. In all experiments, we trained models on eight NVIDIA DGX-1 Tesla V100 GPUs. The runtime for training each model was about 1-4 hours, depending on the size of the training set. The order of training instances was shuffled for each model. We used 10% of the training set for a validation set.

C Detailed evaluation results

Effect of Model Size The results we report are from a model with 10M parameters. How does the number of parameters affect the systematic generalization performance of models? Table 11 shows the performance of three models of varying size (large: 27M, medium: 10M, small: 4M). The number of parameters did not have a large impact on the generalization performance; all runs of the models achieved higher than 90% accuracy on the validation set and the test set involving quantifiers of the same type as the primitive quantifier, while they did not work well on the test set involving the other types of quantifiers.

Test	GRU			Transformer		
	4M	10M	27M	4M	10M	27M
Exi	96.8	99.9	97.1	99.9	99.8	99.3
Num	7.1	11.5	10.4	12.3	12.2	12.4
Uni	6.0	4.9	2.9	7.8	5.9	7.9
Valid	97.2	99.9	97.6	100.0	99.8	97.2

Table 11: The effect of model size on generalization performance (primitive quantifier: existential quantifier *one*, representation form: FOL).

Modifier type Table 14 shows all evaluation results by modifier types where *two* or *every* is set to the primitive quantifier. Regardless of primitive quantifier type, accuracies for problems involving logical connectives or adverbs were better than those for problems involving adjectives.

Monotonicity Table 15 shows all evaluation results of predicted FOL formulas and VF formulas based on monotonicity. We evaluate the precision, recall, and F-score for each monotonicity direction (upward and downward). Regardless of meaning representation forms, downward monotone accuracy on problems involving universal quantifiers is low. This indicates that both models struggle with learning the scope of universal quantifiers.

D Evaluation on systematicity of quantifiers and negation

We also analyze whether models can generalize to unseen combinations of quantifiers and negation. Here, we generate Basic set 1 by setting an arbitrary quantifier to a primitive quantifier and combining it with negation. As in (15b), we fix the primitive quantifier to the existential quantifier *one* and generate the negated sentence *One tiger did not run*. Next, as in (16a) and (16b), we generate Basic set 2 by combining a primitive term (e.g., *tiger*) with various quantifiers. If a model has the ability to systematically understand primitive combinations in Basic set, it can represent a new mean-

Grammar rules			**Semantic composition rules: FOL**	**Semantic composition rules: VF**
S	→	NP VP	$[\![\text{S}]\!] = [\![\text{NP}]\!]([\![\text{VP}]\!])$	$[\![\text{S}]\!] = [\![\text{NP}]\!]([\![\text{VP}]\!])$
S	→	NP *did not* VP	$[\![\text{S}]\!] = [\![\text{NP}]\!](\lambda x.\neg[\![\text{VP}]\!](x))$	$[\![\text{S}]\!] = [\![\text{NP}]\!](\texttt{NOT}([\![\text{VP}]\!]))$
NP	→	PN	$[\![\text{NP}]\!] = [\![\text{PN}]\!]$	$[\![\text{NP}]\!] = [\![\text{PN}]\!]$
NP	→	Q N	$[\![\text{NP}]\!] = [\![\text{Q}]\!]([\![\text{N}]\!])$	$[\![\text{NP}]\!] = [\![\text{Q}]\!]([\![\text{N}]\!])$
NP	→	Q Adj N	$[\![\text{NP}]\!] = [\![\text{Q}]\!](\lambda x.([\![\text{N}]\!](x) \wedge [\![\text{Adj}]\!](x)))$	$[\![\text{NP}]\!] = [\![\text{Q}]\!](\texttt{AND}([\![\text{N}]\!], [\![\text{Adj}]\!]))$
NP	→	Q N $\overline{\text{S}}$	$[\![\text{NP}]\!] = [\![\text{Q}]\!](\lambda x.([\![\text{N}]\!](x) \wedge [\![\overline{\text{S}}]\!](x)))$	$[\![\text{NP}]\!] = [\![\text{Q}]\!](\texttt{AND}([\![\text{N}]\!], [\![\overline{\text{S}}]\!]))$
VP	→	IV	$[\![\text{VP}]\!] = [\![\text{IV}]\!]$	$[\![\text{VP}]\!] = [\![\text{IV}]\!]$
VP	→	IV Adv	$[\![\text{VP}]\!] = \lambda x.([\![\text{IV}]\!](x) \wedge [\![\text{Adv}]\!](x))$	$[\![\text{VP}]\!] = \texttt{AND}([\![\text{IV}]\!], [\![\text{Adv}]\!]))$
VP	→	IV or IV′	$[\![\text{VP}]\!] = \lambda x.([\![\text{IV}]\!](x) \vee [\![\text{IV}']\!](x))$	$[\![\text{VP}]\!] = \texttt{OR}([\![\text{IV}]\!], [\![\text{IV}']\!]))$
VP	→	IV and IV′	$[\![\text{VP}]\!] = \lambda x.([\![\text{IV}]\!](x) \wedge [\![\text{IV}']\!](x))$	$[\![\text{VP}]\!] = \texttt{AND}([\![\text{IV}]\!], [\![\text{IV}']\!]))$
VP	→	TV NP	$[\![\text{VP}]\!] = \lambda x.[\![\text{NP}]\!](\lambda y.[\![\text{TV}]\!](x, y))$	$[\![\text{VP}]\!] = [\![\text{NP}]\!]([\![\text{TV}]\!])$
$\overline{\text{S}}$	→	*that* VP	$[\![\overline{\text{S}}]\!] = [\![\text{VP}]\!]$	$[\![\overline{\text{S}}]\!] = [\![\text{VP}]\!]$
$\overline{\text{S}}$	→	*that did not* VP	$[\![\overline{\text{S}}]\!] = \lambda x.\neg[\![\text{VP}]\!](x)$	$[\![\overline{\text{S}}]\!] = \texttt{NOT}([\![\text{VP}]\!])$
$\overline{\text{S}}$	→	*that* NP TV	$[\![\overline{\text{S}}]\!] = \lambda y.[\![\text{NP}]\!](\lambda x.[\![\text{TV}]\!](x, y))$	$[\![\overline{\text{S}}]\!] = [\![\text{NP}]\!](\texttt{INV}([\![\text{TV}]\!]))$
$\overline{\text{S}}$	→	*that* NP *did not* TV	$[\![\overline{\text{S}}]\!] = \lambda y.[\![\text{NP}]\!](\lambda x.\neg[\![\text{TV}]\!](x, y))$	$[\![\overline{\text{S}}]\!] = [\![\text{NP}]\!](\texttt{NOT}(\texttt{INV}([\![\text{TV}]\!])))$
Q	→	{*every, all, a, one, two, three*}	$[\![\textit{every}]\!] = [\![\textit{all}]\!] = \lambda F \lambda G.\forall x(F(x) \rightarrow G(x))$	$[\![\textit{every}]\!] = [\![\textit{all}]\!] = \lambda F \lambda G.\texttt{ALL}(F, G)$
			$[\![\textit{a}]\!] = [\![\textit{one}]\!] = \lambda F \lambda G.\exists x.(F(x) \wedge G(x))$	$[\![\textit{a}]\!] = [\![\textit{one}]\!] = \lambda F \lambda G.\texttt{EXIST}(F, G)$
			$[\![\textit{two}]\!] = \lambda F \lambda G.\exists x.(\mathbf{two}(x) \wedge F(x) \wedge G(x))$	$[\![\textit{two}]\!] = \lambda F \lambda G.\texttt{TWO}(F, G)$
			$[\![\textit{three}]\!] = \lambda F \lambda G.\exists x.(\mathbf{three}(x) \wedge F(x) \wedge G(x))$	$[\![\textit{three}]\!] = \lambda F \lambda G.\texttt{THREE}(F, G)$
N	→	{*dog, rabbit, cat, bear, tiger,...*}	$[\![\textit{dog}]\!] = \lambda x.\mathbf{dog}(x)$	$[\![\textit{dog}]\!] = \texttt{DOG}$
PN	→	{*ann, bob, fred, chris, eliott,...*}	$[\![\textit{ann}]\!] = \lambda F.F(\mathbf{ann})$	$[\![\textit{ann}]\!] = \lambda F.\texttt{EXIST}(\texttt{ANN}, F)$
IV	→	{*ran, walked, swam, danced, dawdled,...*}	$[\![\textit{ran}]\!] = \lambda x.\mathbf{run}(x)$	$[\![\textit{ran}]\!] = \texttt{RUN}$
IV′	→	{*laughed, groaned, roared, screamed,...*}	$[\![\textit{laugh}]\!] = \lambda x.\mathbf{laugh}(x)$	$[\![\textit{laugh}]\!] = \texttt{LAUGH}$
TV	→	{*kissed, kicked, cleaned, touched,...*}	$[\![\textit{kissed}]\!] = \lambda y \lambda x.\mathbf{kiss}(x, y)$	$[\![\textit{kissed}]\!] = \texttt{KISS}$
Adj	→	{*small, large, crazy, polite, wild,...*}	$[\![\textit{small}]\!] = \lambda x.\mathbf{small}(x)$	$[\![\textit{small}]\!] = \texttt{SMALL}$
Adv	→	{*slowly, quickly, seriously, suddenly,...*}	$[\![\textit{slowly}]\!] = \lambda x.\mathbf{slowly}(x)$	$[\![\textit{slowly}]\!] = \texttt{SLOWLY}$

Table 12: A set of context-free grammar rules and semantic composition rules used to generate the SyGNS dataset.

Sentence	**FOL**	**VF**
a small dog did not swim	$\exists x.(\mathbf{small}^{\uparrow}(x) \wedge \mathbf{dog}^{\uparrow}(x) \wedge \neg\mathbf{swim}^{\downarrow}(x))$	$\texttt{EXIST AND SMALL}^{\uparrow}\ \texttt{DOG}^{\uparrow}\ \texttt{NOT SWIM}^{\downarrow}$
all tigers ran or swam	$\forall x.(\mathbf{tiger}^{\downarrow}(x) \rightarrow \mathbf{run}^{\uparrow}(x) \vee \mathbf{swim}^{\uparrow}(x))$	$\texttt{ALL TIGER}^{\downarrow}\ \texttt{OR RUN}^{\uparrow}\ \texttt{SWIM}^{\uparrow}$
ann did not chase two dogs	$\neg\exists x.(\mathbf{two}(x) \wedge \mathbf{dog}^{\downarrow}(x) \wedge \mathbf{chase}^{\downarrow}(\mathbf{ann}, x))$	$\texttt{EXIST ANN NOT TWO DOG}^{\downarrow}\ \texttt{CHASE}^{\downarrow}$

Table 13: Example of (polarized) FOL formulas and VF formulas.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: numeral *two*								
Adj	10.7	16.8	74.9	22.8	34.4	58.2	91.6	52.0
Adj+Neg	10.1	22.3	79.7	24.3	33.4	58.7	94.4	51.0
Adv	12.8	29.3	79.8	46.9	40.3	56.1	89.1	60.5
Adv+Neg	14.1	33.9	83.8	58.6	34.4	56.4	93.3	65.8
Con	18.3	37.9	80.2	64.8	34.3	52.4	83.4	67.2
Con+Neg	24.4	40.6	82.6	68.7	31.3	50.8	88.0	68.5
primitive quantifier: universal quantifier *every*								
Adj	7.7	19.5	70.6	58.5	20.7	20.5	89.9	62.2
Adj+Neg	6.9	19.2	75.5	56.8	20.3	20.2	92.2	63.6
Adv	9.2	18.2	82.3	70.1	19.7	19.7	85.1	70.4
Adv+Neg	14.8	18.1	79.4	76.1	22.7	19.8	89.3	75.5
Con	14.7	18.0	70.3	79.6	21.5	19.2	68.8	75.8
Con+Neg	18.3	18.2	80.1	81.2	22.7	19.1	80.5	76.8

Table 14: Accuracy by modifier type where *two* or *every* is the primitive quantifier. “DRS (cnt)” columns show accuracies of predicted DRSs by Counter.

Test	GRU							Transformer						
	Exact	Upward			Downward			Exact	Upward			Downward		
	Match	Prec	Rec	F	Prec	Rec	F	Match	Prec	Rec	F	Prec	Rec	F
FOL formula														
Exi	96.1	100.0	99.9	99.9	100.0	100.0	100.0	99.9	100.0	100.0	100.0	100.0	100.0	100.0
Num	7.6	99.2	75.5	84.8	99.6	94.7	96.8	18.1	100.0	79.5	88.1	100.0	95.6	97.5
Uni	3.1	92.6	89.9	90.9	42.9	39.4	40.7	8.3	97.3	93.4	94.9	79.4	70.0	73.4
VF formula														
Exi	99.7	100.0	99.9	99.9	99.9	99.9	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Num	37.0	70.2	54.4	59.8	68.6	58.5	62.0	20.7	99.2	77.0	85.4	99.3	95.4	97.0
Uni	39.5	91.1	80.7	84.4	49.0	35.2	39.7	17.7	99.9	97.4	98.4	98.6	72.7	82.3

Table 15: Evaluation results on monotonicity. “Prec”, “Rec”, “F” indicate precision, recall, and F-score.

ing representation with different combinations of quantifiers and negations, like (17a) and (17b).

(15) a. **One** tiger ran
b. **One** tiger *did not* run

(16) a. **Every** tiger ran
b. **Two** tigers ran

(17) a. **Every** tiger *did not* run
b. **Two** tigers *did not* run

Table 16 shows the accuracy on combinations of quantifiers and negations by quantifier type. Similar to the results with unseen combinations of quantifiers and modifiers, models can easily generalize to problems involving quantifiers of the same type as the primitive quantifier. Table 17 shows the accuracy on combinations of quantifiers and negations by modifier types. Similar to the results in Table 14, the accuracies on problems involving logical connectives or adverbs were slightly better than those on problems involving adjectives.

E Error analysis of predicted DRSs

In the productivity experiments, the evaluation results using COUNTER on DRS prediction tasks are much higher than those by exact matching. Table 18 shows an example of predicted DRSs for the sentence *all lions that did not follow two bears that chased three monkeys did not cry*. This sentence contains embedded clauses with depth two, having the following gold DRS:

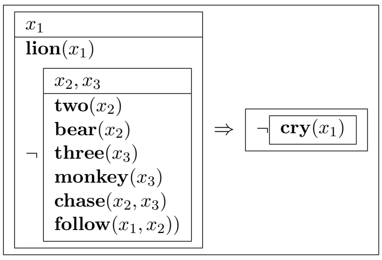

Both GRU and Transformer tend to correctly predict some of the clauses for content words, implication, and negation that appear at the beginning of the input sentence, while they fail to capture long-distance dependencies between subject nouns and verbs (e.g., *all lions ... did not cry*). Also, COUNTER correctly evaluates the cases where the order of clauses is different from that of gold answers.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: existential quantifier *one*								
Exi	65.9	88.0	98.1	94.5	99.9	96.6	97.6	45.0
Num	48.4	71.0	96.7	54.5	65.8	86.6	98.8	26.3
Uni	22.3	0.0	52.0	53.1	0.0	0.0	70.2	26.6
Valid	96.6	98.1	100.0	99.8	100.0	100.0	100.0	100.0
primitive quantifier: numeral *two*								
Exi	36.4	63.6	89.8	13.9	14.4	49.6	86.9	11.5
Num	40.0	66.7	94.1	21.5	33.0	59.1	87.6	14.3
Uni	15.4	0.0	50.7	0.0	0.0	0.0	71.0	12.4
Valid	96.4	97.8	100.0	99.3	100.0	100.0	100.0	100.0
primitive quantifier: universal quantifier *every*								
Exi	12.8	0.0	73.7	78.6	0.8	0.5	52.2	59.4
Num	17.1	0.0	75.3	78.1	0.0	0.6	59.2	65.1
Uni	91.1	88.3	97.4	94.9	86.6	70.1	92.3	76.6
Valid	98.8	98.1	100.0	98.8	100.0	100.0	100.0	100.0

Table 16: Accuracy on combinations of quantifiers and negation by quantifier type. “DRS (cnt)” columns show accuracies of predicted DRSs by Counter. “Valid” row shows the validation accuracy.

Test	GRU				Transformer			
	FOL	DRS	DRS (cnt)	VF	FOL	DRS	DRS (cnt)	VF
primitive quantifier: existential quantifier *one*								
Adj+Neg	34.6	33.2	68.3	45.8	19.4	45.1	84.1	25.3
Adv+Neg	38.0	36.3	77.9	53.0	43.2	54.8	88.1	37.5
Con+Neg	36.8	33.2	73.2	54.7	47.4	52.4	85.4	37.0
primitive quantifier: numeral *two*								
Adj+Neg	21.2	18.2	69.3	8.0	11.6	29.3	75.9	1.2
Adv+Neg	26.5	28.5	71.4	10.3	19.3	36.5	81.8	17.8
Con+Neg	21.9	28.1	68.5	9.1	11.7	34.6	78.7	16.3
primitive quantifier: universal quantifier *every*								
Adj+Neg	26.7	12.4	63.9	60.5	13.9	14.6	58.0	50.7
Adv+Neg	25.9	13.2	69.2	66.1	21.9	15.2	60.9	63.3
Con+Neg	28.5	18.8	71.4	65.9	20.2	14.6	59.7	62.8

Table 17: Accuracy on combinations of quantifiers and negation by modifier type.

(a) Gold answer	(b) GRU (F: 0.45)	(c) Transformer (F: 0.42)
b1 IMP **b2** **b4**	**b1** IMP **b2** **b3**	**b1** IMP **b2** **b3**
b2 REF **x1**	**b2** REF **x1**	**b2** REF **x1**
b2 lion **x1**	**b2** lion **x1**	**b2** lion **x1**
b2 NOT **b3**	**b2** NOT **b3**	**b2** NOT **b3**
b3 REF **x2**	**b3** REF **x2**	**b3** REF **x2**
b3 REF **x3**	**b3** two **x2**	**b3** two **x2**
b3 two **x2**	**b3** bear **x2**	**b3** monkey **x2**
b3 bear **x2**	**b3** follow **x2** **x2**	**b3** follow **x2** **x1**
b3 three **x3**	**b3** REF **x3**	**b3** REF **x3**
b3 monkey **x3**	**b3** three **x3**	**b3** john **x3**
b3 chase **x3** **x2**	**b4** monkey **x3**	**b3** chase **x1** **x3**
b3 follow **x2** **x1**	**b4** like **x3** **x2**	
b4 NOT **b5**	**b4** like **x1** **x2**	
b5 cry **x1**		

Table 18: Error analysis of DRSs for the sentence “all lions that did not follow two bears that chased three monkeys did not cry”. Clauses in green are correct and those in red are incorrect. “F” shows F-score over matching clause.

Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade

Jiatao Gu[*]
Facebook AI Research
jgu@fb.com

Xiang Kong[*]
Language Technologies Institute
Carnegie Mellon University
xiangk@cs.cmu.edu

Abstract

Fully non-autoregressive neural machine translation (NAT) simultaneously predicts tokens with single forward of neural networks, which significantly reduces the inference latency at the expense of quality drop compared to the Transformer baseline. In this work, we target on closing the performance gap while maintaining the latency advantage. We first inspect the fundamental issues of fully NAT models, and adopt *dependency reduction* in the learning space of output tokens as the primary guidance. Then, we revisit methods in four different aspects that have been proven effective for improving NAT models, and carefully combine these techniques with necessary modifications. Our extensive experiments on three translation benchmarks show that the proposed system achieves the state-of-the-art results for fully NAT models, and obtains comparable performance with the autoregressive and iterative NAT systems. For instance, one of the proposed models achieves **27.49** BLEU points on WMT14 En-De with **16.5**× speed-up compared to similar sized autoregressive baseline under the same inference condition. The implementation of our model is available here[1].

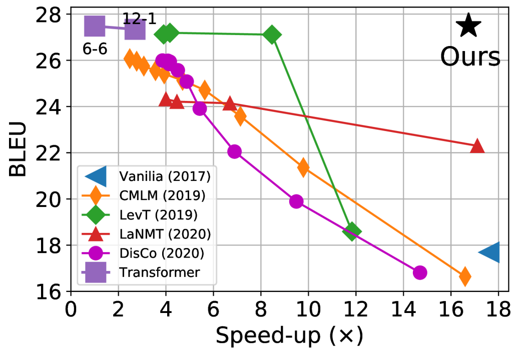

Figure 1: The translation quality v.s. inference speed-up on WMT'14 En→De test set. The upper right corner achieves the best trade-off.

1 Introduction

State-of-the-art neural machine translation (NMT) systems are based on autoregressive models (Bahdanau et al., 2015; Vaswani et al., 2017) where each generation step depends on the previously generated tokens. This sequential nature inevitably leads to inherent latency at inference time. On the other hand, non-autoregressive neural machine translation models (NAT, Gu et al., 2018a) attempt to generate output sequences in parallel to speed-up the decoding process. The incorrect independence assumption nevertheless prevents NAT models to properly learn the dependency between target tokens in real data distribution, resulting in poorer performance compared to autoregressive (AT) models. One popular solution to improve the NAT translation accuracy is to sacrifice the speed-up by incorporating an iterative refinement process, through which the model explicitly learns the conditional distribution over partially observed reference tokens (Ghazvininejad et al., 2019; Gu et al., 2019). However, recent studies (Kasai et al., 2020b) indicated that iterative NAT models seem to lose the speed advantage compared to AT models with careful tuning of the layer allocation. For instance, an AT model with *deep encoder and shallow decoder* obtains similar latency as iterative NAT models without hurting the translation accuracy.

Therefore, how to build a competitive fully NAT model without iterative refinements calls for more exploration. Several works (Ghazvininejad et al., 2020a; Saharia et al., 2020; Qian et al., 2020) have recently been proposed to improve the training of NAT, though the performance gap compared to the iterative ones remains. In this work, we first ar-

---

[*] Equal contribution.

[1] `https://github.com/pytorch/fairseq/tree/master/examples/nonautoregressive_translation`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 120–133
August 1–6, 2021.

gue that the key to successfully training a fully NAT model is to perform *dependency reduction* in the learning space of output tokens (§ 2) from all aspects. With this guidance, we revisit various methods which are able to reduce the dependencies among target tokens as much as possible including four different perspectives, i.e., training corpus (§ 3.1), model architecture (§ 3.2), training objective (§ 3.3) and learning strategy (§ 3.4). The performance gap can not be near closed unless we combine these techniques' advantages.

We validate the proposed fully NAT model on standard translation benchmarks including 5 translation directions where our system achieves new state-of-the-art results for fully NAT models on all directions. We also demonstrate the quality-speed trade-off comparing with AT and recent iterative NAT models in Figure 1. Moreover, compared to the Transformer baseline, our model achieves **16.5**× inference speed-up under the same software/hardware conditions while maintaining comparable translation quality.

2 Motivation

Given an input sequence $\boldsymbol{x} = x_1 \dots x_{T'}$, an autoregressive model (Bahdanau et al., 2015; Vaswani et al., 2017) predicts the target $\boldsymbol{y} = y_1 \dots y_T$ sequentially based on the conditional distribution $p(y_t|y_{<t}, x_{1:T'}; \theta)$, which tends to suffer from high latency in generation especially for long sequences. In contrast, non-autoregressive machine translation (NAT, Gu et al., 2018a), proposed for speeding-up the inference by generating all the tokens in parallel, has recently been on trend due to its nature of parallelizable on devices such as GPUs and TPUs. A typical NAT system assumes a conditional independence in the output token space, that is

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{t=1}^{T} \log p_\theta(y_t|x_{1:T'}) \qquad (1)$$

where θ is the parameters of the model. Typically, NAT models are modeled with Transformer without causal attention map in the decoder side. As noted in Gu et al. (2018a), the independence assumption, however, generally does not hold in real data distribution for sequence generation tasks such as machine translation (Ren et al., 2020), where the failure of capturing such dependency between target tokens leads to a serious performance degradation in NAT. This is a fairly understandable but fundamental issue of NAT modeling which can be easily shown with a toy example in Figure 2. Given a simple corpus with only two examples: *AB* and *BA*, each of which has 50% chances to appear. It is designed to represent the dependency that symbol *A* and *B* should co-occur. Although such simple distribution can be instantly captured by any autoregressive model, learning the vanilla NAT model with maximum likelihood estimation (MLE, Eq. (1)) assigns probability mess to incorrect outputs (*AA*, *BB*) even these samples never appear during training. In practice, the dependency in real translation corpus is much more complicated. As shown in Figure 1, despite the inference speed-up, the vanilla NAT leads to a quality drop over **10** BLEU points.

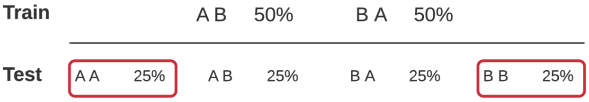

Figure 2: Toy example shows that NAT fails to learn when dependency exists in output space.

To ease the modeling difficulty, recent state-of-the-art NAT systems (Lee et al., 2018; Stern et al., 2019; Ghazvininejad et al., 2019; Gu et al., 2019; Kasai et al., 2020a; Shu et al., 2020; Saharia et al., 2020) trade accuracy with latency by incorporating iterative refinement in non-autoregressive prediction. For instance, Gu et al. (2019) learns to translate by editing (deletion, insertion) on previously generated sequence iteratively. Although iterative NAT models have already achieved comparable or even better performance than the autoregressive counterpart, Kasai et al. (2020b) showed AT models with a deep encoder and a shallow decoder can readily outperform strong iterative models with similar latency, indicating that the latency advantage of iterative NAT has been overestimated.

By contrast, while maintaining a clear speed advantage, fully NAT system – model makes parallel predictions with single neural network forward – still lags behind in translation quality and has not been fully explored in literature (Libovický and Helcl, 2018; Li et al., 2018; Sun et al., 2019; Ma et al., 2019; Ghazvininejad et al., 2020a). This motivates us in this work to investigate various approaches to push the limits of learning a fully NAT model towards autoregressive models regardless of the architecture choices (Kasai et al., 2020b).

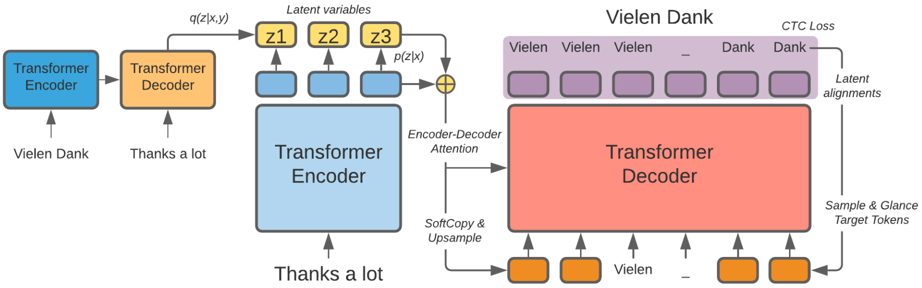

Figure 3: The overall framework of our fully NAT model.

3 Methods

In this section, we discuss several essential ingredients to train a fully NAT model. As discussed in § 2, we argue that the guiding principle of designing any NAT models is to perform *dependency reduction* as much as possible in the output space so that it can be captured by the NAT model. For example, iterative-based models (Ghazvininejad et al., 2019) explicitly reduce the dependencies between output tokens by learning the conditional distribution over the observed reference tokens. The overall framework of training our fully NAT system is presented in Figure 3. We also summarize the pros/cons for each proposed method in Table 1 for reference.

3.1 Data: Knowledge Distillation

The most effective *dependency reduction* technique is knowledge distillation (KD) (Hinton et al., 2015; Kim and Rush, 2016) which is firstly proposed to improve NAT in Gu et al. (2018a) and has been widely employed for all subsequent NAT models. The original target samples are replaced with sentences generated from a pre-trained autoregressive model. As analyzed in Zhou et al. (2020), KD is able to simplify the training data where the generated targets have less noise and are aligned to the inputs more deterministically. Also, it showed that the capacity of the teacher model should be constrained to match the desired NAT model to avoid further degradation, especially for weak NAT students without iterative refinement.

3.2 Model: Latent Variables

Different from iterative NAT, *dependency reduction* can be done with (nearly) zero additional cost at inference by adding latent variables to the model. In such case, output tokens $y_{1:T}$ are modeled conditionally independent over the latent variables z which are predicted from the prior distribution:

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \log \int_{\boldsymbol{z}} p_\theta(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x}) d\boldsymbol{z} \quad (2)$$

z can be either extracted by a fixed external library (e.g. fertility in Gu et al. (2018a)), or jointly optimized with the NAT model using variational autoencoders (VAEs) (Kaiser et al., 2018; Shu et al., 2020) or normalizing flow (Ma et al., 2019).

In this work, we followed the formulation proposed in Shu et al. (2020) where continuous latent variables $\boldsymbol{z} \in \mathbb{R}^{T' \times D}$ are modeled as spherical Gaussian at the encoder output of each position. Like typical VAEs (Kingma and Welling, 2013), the model is trained by maximizing the evidence lower-bound (ELBO) with a posterior network q_ϕ:

$$\underbrace{\mathop{\mathbb{E}}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})\right]}_{\text{likelihood}} - \mathcal{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) \| p_\theta(\boldsymbol{z}|\boldsymbol{x})) \quad (3)$$

where $\mathcal{D}_{\mathrm{KL}}$ is the Kullback–Leibler divergence between the prior and posterior. In this work, we use a Transformer to encode $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$. Only the embedding layers are shared between θ and ϕ

3.3 Loss Function: Latent Alignments

Standard NMT models are trained with the cross entropy (CE) loss which compares the model's output with target tokens at each corresponded position. However, as NAT ignores the dependency in the output space, it is almost impossible for such models to model token offset accurately. For instance, while with little effect to the meaning, simply changing "*Vielen Dank !*" to "*, Vielen Dank*" causes a huge penalty for fully NAT models.

To ease such limitation, recent works proposed to consider the latent alignments between the target positions, and optimize (Ghazvininejad et al.,

Methods	Distillation	Latent Variables	Latent Alignments	Glancing Targets
What it can do?	simplifying the training data	model any types of dependency in theory	handling token shifts in the output space	ease the difficulty of learning hard examples
What it cannot?	uncertainty exists in the teacher model	constrained by the modeling power of the used latent variables	unable to model non-monotonic dependency, e.g. reordering	training / testing phase mismatch
Potential issues	sub-optimal due to the teacher's capacity	difficult to train; posterior collapse	decoder inputs must be longer than targets	difficult to find the optimal masking ratio

Table 1: Comparison between the proposed techniques for improving fully NAT models.

2020a), or marginalize all alignments (Libovický and Helcl, 2018; Saharia et al., 2020). As a special form of latent variables in loss computation, latent alignments can be easily computed through dynamic programming. The dependency is reduced because the NAT model is able to freely choose the best prediction regardless of the target offsets. In this work, we put our primary focus on Connectionist Temporal Classification (CTC) (Graves et al., 2006) as the latent alignments, considering its superior performance and the flexibility of variable length prediction. Formally, CTC is capable of efficiently finding all valid aligned sequences $\boldsymbol{a}$ which the target $\boldsymbol{y}$ can be recovered from, and marginalize log-likelihood:

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \log \sum_{\boldsymbol{a}\in\Gamma(\boldsymbol{y})} p_\theta(\boldsymbol{a}|\boldsymbol{x}) \tag{4}$$

where $\Gamma^{-1}(\boldsymbol{a})$ is the collapse function that recovers the target sequence by collapsing consecutive repeated tokens, and then removing all blank tokens. Also, it is straightforward to apply the same CTC loss into the VAE models (§ 3.2) by replacing the likelihood term in Eq (3) with the CTC loss. Because of the strong assumptions of monotonic alignment, it is impossible to reduce all dependencies between target tokens in real distribution.

3.4 Learning: Glancing Targets

Ghazvininejad et al. (2019) showed that it improved test time performance by glancing the reference tokens when training NAT models. That is, instead of $\log p_\theta(\boldsymbol{y}|\boldsymbol{x})$, we optimize $\log p_\theta(\boldsymbol{y}|\boldsymbol{m}\odot\boldsymbol{y},\boldsymbol{x}), \boldsymbol{m} \sim \gamma(l,\boldsymbol{y}), l \sim \mathcal{U}_{|\boldsymbol{y}|}$, where $\boldsymbol{m}$ is the mask, and γ is the sampling function given the number of masked tokens l. As mentioned earlier, we suspect such explicit modeling of the distribution conditional to unmasked tokens assists the *dependency reduction* in the output space.

Naively applying random masks for every training example may cause severe mismatch between training and testing. To migrate this, Qian et al. (2020) proposed GLAT – a curriculum learning strategy, in which the ratio of glanced target tokens is proportional to the translation error of the fully NAT model. More precisely, instead of sampling uniformly, we sample l by:

$$l \sim g(f_{\text{ratio}} \cdot \mathcal{D}(\hat{\boldsymbol{y}},\boldsymbol{y})) \tag{5}$$

where $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \log p_\theta(\boldsymbol{y}|\boldsymbol{x})$, $\mathcal{D}$ is the discrepancy between the model prediction and the target sequence, e.g. Levenshtein distance (Levenshtein, 1966), and f_{ratio} is a hyperparameter to adjust the mask ratio. The original formulation (Qian et al., 2020) utilized a deterministic mapping (g), while we use a Poisson distribution to sample a wider range of lengths including "no glancing".

The original GLAT (Qian et al., 2020) assumes to work with golden length so that it can glance at the target by placing the target word embedding to the corresponded inputs, which is incompatible with CTC as we always require the inputs longer than the targets. To enable GLAT training, we glance at target tokens from the viterbi aligned tokens $\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}\in\Gamma(\boldsymbol{y})} \log p_\theta(\boldsymbol{a}|\boldsymbol{x})$ which has the same length as the decoder inputs. Intuitively, a poorly trained model will glance at many target tokens. When the model becomes better and generates higher quality sequences, the number of masked words will be larger, which helps the model gradually learn generating the whole sentence.

4 Experiments

We perform extensive experiments on three challenging translation datasets by combining all mentioned techniques to check (1) whether the proposed aspects for *dependency reduction* are complementary; (2) how much we can minimize the gap between a fully non-autoregressive model with the autoregressive counterpart.

4.1 Experimental Setup

Dataset and Preprocessing We validate our proposed models on three standard translation benchmarks with variant sizes, i.e., WMT14 English (EN) ↔ German (DE) (4.0M pairs), WMT16 English (EN) ↔ Romanian (RO) (610k pairs) and WMT20 Japanese (JA) → English (EN) (13M pairs after filtering). For EN↔DE and EN↔RO, we apply the same prepossessing steps and learn subwords as mentioned in prior work (EN↔DE: Zhou et al., 2020, EN↔RO: Lee et al., 2018). For JA→EN, the original data (16M pairs) is first filtered with Bicleaner (Sánchez-Cartagena et al.)[2] and we apply SentencePiece (Kudo and Richardson, 2018) to generate 32,000 subwords.

Knowledge Distillation Following previous efforts, we also train the NAT models on distilled data generated from pre-trained transformer models (*base* for WMT14 EN↔DE and WMT16 EN↔RO and *big* for WMT20 JA→EN) using beam search with a beam size 5 and length penalty 1.0.

Decoding At inference time, the most straightforward way is to generate the sequence with the highest probability at each position. The outputs from the CTC-based NAT models require an additional collapse process Γ^{-1} which can be done instantly. A relatively more accurate method is to decode multiple sequences, and rescore them to obtain the best candidate in parallel, i.e. *noisy parallel decoding* (NPD, Gu et al., 2018a). Furthermore, CTC-based models are also capable of decoding sequences using beam-search (Libovický and Helcl, 2018), and optionally combined with n-gram language models (Heafield, 2011; Kasner et al., 2020). More precisely, we search in a beam to approximately find the optimal $\boldsymbol{y}^*$ that maximizes:

$$\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) + \alpha \cdot \log p_{\text{LM}}(\boldsymbol{y}) + \beta \log |\boldsymbol{y}| \quad (6)$$

where α and β are hyperparameters for language model scores and word insertion bonus. In principle, it is no longer non-autoregressive as beam-search is a sequential process by nature. However, it does not contain any neural network computations and can be implemented efficiently in C++[3].

Baselines We adopt Transformer (AT) and existing NAT approaches (see Table 2) for comparison. For AT, except for the standard *base* and *big* architectures (Vaswani et al., 2017), we also compare with a *deep encoder, shallow decoder* Transformer suggested in Kasai et al. (2020b) that follows the model dimensions of *base* with 12 encoder layers and 1 decoder layer (i.e. *base* (12-1) for short).

Evaluation BLEU (Papineni et al., 2002) is used to evaluate the translation performance for all models. Following prior works, we compute tokenized BLEUs for EN↔DE and EN↔RO, while using SacreBLEU (Post, 2018) for JA→EN. In this work, we use three measures to fully investigate the translation latency of all the models:

- $\mathcal{L}_1^{\text{GPU}}$: translation latency by running the model with one sentence/batch on single GPU, aligning applications like instantaneous translation.
- $\mathcal{L}_1^{\text{CPU}}$: the same as $\mathcal{L}_1^{\text{GPU}}$ while running the model without GPU speed-up. Compared to $\mathcal{L}_1^{\text{GPU}}$, it is less friendly to NAT models that make use of parallelism, however, closer to real scenarios.
- $\mathcal{L}_{\text{max}}^{\text{GPU}}$: the same as $\mathcal{L}_1^{\text{GPU}}$ on GPU while running the model in a batch with as many sentences as possible. In this case, the hardware memory bandwidth are taken into account.

We measure the wall-clock time for translating the whole test set, and report the averaged time over sentences as the latency measure. For more implementation details, please refer to Appendix A.

4.2 Results

WMT'14 EN↔DE & WMT'16 EN↔RO We report the performance of our fully NAT model comparing with AT and existing NAT approaches (including both iterative and fully NAT models) in Table 2. Iterative NAT models with enough number of iterations generally outperform fully NAT baselines by a certain margin as they are able to recover the generation errors by explicitly modeling dependencies between (partially) generated tokens. However, the speed advantage is relatively small compared to AT *base* (12-1) which also achieves 2.5 times faster than the AT baseline.

Conversely, our fully NAT models are able to readily achieve over 16 times speed-up on EN→DE by restricting translation within a single iteration. Surprisingly, merely training NAT with KD and CTC loss already beats the state-of-the-art for single iteration NAT models across all four directions. Moreover, combining with either latent variables (VAE) or glancing targets (GLAT) further closes the performance gap or even outperforms the AT results on both language pairs. For example, our best

---

[2] `https://github.com/bitextor/bicleaner`

[3] `https://github.com/parlance/ctcdecode`

Models		Iter.	Speed	WMT'14		WMT'16	
				EN-DE	DE-EN	EN-RO	RO-EN
AT	Transformer *base* (teacher)	N	1.0×	**27.48**	**31.39**	**33.70**	**34.05**
	Transformer *base* (12-1)	N	2.4×	26.21	30.80	33.17	33.21
	+ KD	N	2.5×	27.34	30.95	33.52	34.01
Iterative NAT	iNAT (Lee et al., 2018)	10	1.5×	21.61	25.48	29.32	30.19
	Blockwise (Stern et al., 2018)	$\approx N/5$	3.0×	27.40	-	-	-
	InsT (Stern et al., 2019)	$\approx \log N$	4.8×	27.41	-	-	
	CMLM (Ghazvininejad et al., 2019)*	10	1.7×	27.03	30.53	33.08	33.31
	LevT (Gu et al., 2019)	Adv.	4.0×	27.27	-	-	33.26
	KERMIT (Chan et al., 2019)	$\approx \log N$	-	27.80	30.70	-	-
	LaNMT (Shu et al., 2020)	4	5.7×	26.30	-	-	29.10
	SMART (Ghazvininejad et al., 2020b)*	10	1.7×	27.65	31.27	-	-
	DisCO (Kasai et al., 2020a)*	Adv.	3.5×	27.34	31.31	33.22	33.25
	Imputer (Saharia et al., 2020)*	8	3.9×	**28.20**	**31.80**	**34.40**	**34.10**
Fully NAT	Vanilla-NAT (Gu et al., 2018a)	1	15.6×	17.69	21.47	27.29	29.06
	LT (Kaiser et al., 2018)	1	3.4×	19.80	-	-	-
	CTC (Libovický and Helcl, 2018)	1	-	16.56	18.64	19.54	24.67
	NAT-REG (Wang et al., 2019)	1	-	20.65	24.77	-	-
	Bag-of-ngrams (Shao et al., 2020)	1	10.0×	20.90	24.60	28.30	29.30
	Hint-NAT (Li et al., 2018)	1	-	21.11	25.24	-	-
	DCRF (Sun et al., 2019)	1	10.4×	23.44	27.22	-	-
	Flowseq (Ma et al., 2019)	1	1.1 ×	23.72	28.39	29.73	30.72
	ReorderNAT (Ran et al., 2019)	1	16.1×	22.79	27.28	29.30	29.50
	AXE (Ghazvininejad et al., 2020a)*	1	15.3×	23.53	27.90	30.75	31.54
	ENGINE (Tu et al., 2020)	1	15.3×	22.15	-	-	33.16
	EM+ODD (Sun and Yang, 2020)	1	16.4×	24.54	27.93	-	-
	GLAT (Qian et al., 2020)	1	15.3×	25.21	29.84	31.19	32.04
	Imputer (Saharia et al., 2020)*	1	18.6×	25.80	28.40	32.30	31.70
	Ours (Fully NAT)	1	17.6×	11.40	16.47	24.52	24.79
	+ KD	1	17.6×	19.50	24.95	29.91	30.25
	+ KD + CTC	1	16.8×	26.51	30.46	33.41	34.07
	+ KD + CTC + VAE	1	16.5×	**27.49**	31.10	**33.79**	33.87
	+ KD + CTC + GLAT	1	16.8×	27.20	**31.39**	33.71	**34.16**

Table 2: Comparison between our models and existing methods. The speed-up is measured on WMT'14 En→De test set. All results reported standalone are without re-scoring. **Iter.** denotes the number of iterations at inference time, **Adv.** means adaptive, * denotes models trained with distillation from a *big* Transformer.

model achieves **27.49** BLEU on WMT14 EN-DE – almost identical to the AT performance (27.48) while **16.5** times faster in the inference time.

Table 2 also indicates the difficulties of learning NAT on each dataset. For instance, EN↔RO is relatively easier as "KD + CTC" is enough to close the performance gap. By contrast, applying VAE or GLAT helps to capture non-monotonic dependencies and improve by $0.5 \sim 1$ BLEU points on EN↔ED. For both datasets, we ONLY need a single greedy generation to achieve similar translation quality as AT beam-search results.

WMT'20 JA→EN In Table 3, we also present results for training the fully NAT model on a more challenging benchmark – WMT'20 JA→EN which is much larger (13M pairs) and noisier. In addition, JA is linguistically distinct from EN which makes it harder to learn mappings between them. Consequently, both AT (12-1) and our fully NAT models become less confident and tend to generate shorter translations (BP < 0.9), which in turn underperform the AT teacher even trained with KD.

Beam search & NPD Previous works (Gu et al., 2018a; Libovický and Helcl, 2018) find that NAT performance can be effectively improved by allowing advanced decoding methods, such as beam-search and re-ranking (NPD). To fully examine our proposed fully NAT model and demonstrate its extensibility with advanced decoding approaches, we further conduct experiments on WMT'20 JA→EN.

For CTC beam search, we use a fixed beam-size 20 while grid-search α, β (Eq.(6)) based on the performance on the validation set. The language model [4] is trained directly on the distilled target sentences to avoid introducing additional information. We explored both 3-gram and 4-gram LMs in our initial experiments, and found 4-gram worked slightly better with no effect on the infer-

---

[4] `https://github.com/kpu/kenlm`

	Configuration	BLEU (Δ)	BP	$\mathcal{L}_1^{\text{GPU}}$ (Speed-up)		$\mathcal{L}_1^{\text{CPU}}$ (Speed-up)	
AT	*big* (teacher)	21.07	0.920	345 ms	1.0 ×	923 ms	1.0 ×
	base	18.91	0.908	342 ms	1.0 ×	653 ms	1.4 ×
	base (12-1)	15.47	0.806	152 ms	2.3 ×	226 ms	4.0 ×
	base (12-1) + KD	18.76	0.887	145 ms	2.4 ×	254 ms	3.6 ×
NAT	KD + CTC	16.93 (+0.00)	0.828	17.3 ms	19.9 ×	84 ms	11.0 ×
	KD + CTC + VAE	18.73 (+1.80)	0.862	16.4 ms	21.0 ×	83 ms	11.1 ×
	w. *BeamSearch20*	19.80 (+2.87)	0.958	28.5 ms	12.1 ×	99 ms	9.3 ×
	w. *BeamSearch20* + *4-gram LM*	**21.41 (+4.48)**	0.954	31.5 ms	11.0 ×	106 ms	8.7 ×
	w. *NPD5*	18.88 (+1.95)	0.866	34.9 ms	9.9 ×	313 ms	2.9 ×
	w. *NPD5* + *BeamSearch20* + *4-gram LM*	**21.84 (+4.91)**	0.962	57.6 ms	6.0 ×	284 ms	3.2 ×

Table 3: Performance comparison between fully NAT and AT models on WMT'20 JA→EN. Translation latency on both the GPU and CPUs are reported over the test set. The brevity penalty (BP) is also shown for reference.

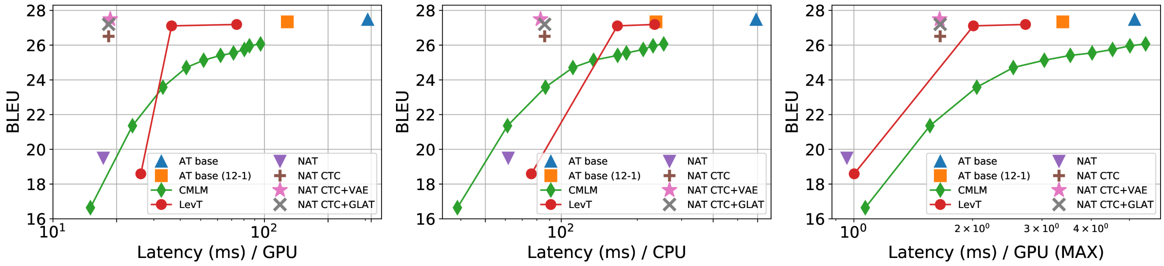

Figure 4: Quality v.s. Latency (the upper left corner achieves the best trade-off) for fully NAT models with other translation models (AT *base* and *base* 12-1 (Kasai et al., 2020b), CMLM (Ghazvininejad et al., 2019) and LevT (Gu et al., 2019)) on WMT'14 EN→DE. We evaluate latency in three setups (from left to right: $\mathcal{L}_1^{\text{GPU}}$, $\mathcal{L}_1^{\text{CPU}}$, $\mathcal{L}_{\text{max}}^{\text{GPU}}$) and show them in Logarithmic scale for better visualization.

ence speed. For noisy parallel decoding (NPD), we draw multiple z from the learned prior distribution with temperature 0.1, and use the teacher model to rerank the best z with the corresponded translation.

As shown in Table 3, with similar GPU latency ($\mathcal{L}_1^{\text{GPU}}$), beam search is much more effective than NPD with re-ranking, especially combined with a 4-gram LM where we achieve a BLEU score of 21.41, beating the teacher model with 11× speed-up. More importantly, by contributing the insertion bonus (3rd term in Eq (6)) with β in beam search, we have the explicit control to improve BP and output longer translations. Also, we gain another half point by combining NPD and beam search. To have a fair comparison, we also report latency on CPUs where it is limited to leverage parallelism of the device. The speed advantage drops rapidly for NAT models, especially for NAT with NPD, however, we still maintain around 100 ms latency via beam search – over 2× faster than the lightweight AT (12-1) systems with higher translation quality.

Quality v.s. Latency We perform a full investigation for the trade-off between translation quality and latency under three measures defined in § 4.1. The results are plotted in Figure 4. For fully NAT models, no beam search or NPD is considered. The latency is measured by $\mathcal{L}_1^{\text{GPU}}$, $\mathcal{L}_1^{\text{CPU}}$ and $\mathcal{L}_{\text{max}}^{\text{GPU}}$ so as to understand this trade-off in various scenarios. In all three setups, our fully NAT models obtain superior trade-off compared with AT and iterative NAT models. Iterative NAT models (LevT and CMLM) require multiple iterations to achieve reliable performance with the sacrifice of latency, especially under $\mathcal{L}_1^{\text{CPU}}$ and $\mathcal{L}_{\text{max}}^{\text{GPU}}$ where iterative NAT performs similarly or even worse than AT *base* (12-1), leaving fully NAT models a more suitable position in quality-latency trade-off.

Figure 4 also shows the speed advantage of fully NAT models shrinks in the setup of $\mathcal{L}_1^{\text{CPU}}$ and $\mathcal{L}_{\text{max}}^{\text{GPU}}$ where parallelism is constrained. Moreover, NAT models particularly those with CTC consume more computation and memory compared to AT models with a shallow decoder. For instance when calculating $\mathcal{L}_{\text{max}}^{\text{GPU}}$, we notice that the maximum allowed batch is 120K tokens for AT base (12-1), while we can only compute 15K tokens at a time for NAT with CTC due to the up-sampling step, even though the NAT models still win the wall-clock time. We

KD	AXE	CTC	VAE	RND	GLAT	BLEU
						11.40
✓						19.50
	✓					16.59
✓	✓					21.66
		✓				18.18
✓		✓				**26.51**
		✓	✓			23.58
✓	✓		✓			22.19
✓		✓	✓			**27.49**
✓	✓			✓		22.74
✓	✓				✓	24.67
✓		✓		✓		26.16
		✓			✓	21.81
✓		✓			✓	**27.20**
✓		✓	✓		✓	**27.21**

Table 4: Ablation on WMT'14 EN→DE test set with different combinations of techniques. The default setup shows a plain NAT model (Gu et al., 2018a) directly trained on raw targets with the cross entropy (CE) loss.

mark it as one limitation for future research.

4.3 Ablation Study

Impact of various techniques Our fully NAT models benefit from dependency reduction techniques in four aspects (data, model, loss function and learning), and we analyze their effects on translation accuracy through various combinations in Table 4. First of all, the combinations without KD have clear performance drop compared to those with KD, showing its vital importance in NAT training. For the loss function, although both AXE (Ghazvininejad et al., 2019) and CTC consider the latent alignments, the CTC-based model obtains much better accuracy due to its flexibility of output length. In all cases, incorporating latent variables also effectively improves the accuracy, especially for CTC without KD ($\sim$ 5 BLEU improvement). Because of the capability to reduce the mismatch between training and inference time, the model with GLAT is superior to those with randomly (RND) sampled masks. To conclude, we find that KD and CTC are necessary components for a robust fully NAT model. Adding either VAE or GLAT to them achieve similar improvements.

Distillation corpus We report the performance of models trained on real data and distilled data generated from AT *base* and *big* models in Table 5. For *base* models, both AT (12-1) and NAT achieve better accuracy with distillation, while AT benefits more by moving from *base* to *big* distilled data. On the contrary, the NAT model improves marginally indicating that in terms of the modeling capacity, our fully NAT model is still worse than AT model even with 1 decoder layer. It is not possible to further boost the NAT performance by simply switching the target to a better distillation corpus, which aligns the finding in Zhou et al. (2020). Nonetheless, we can increase the NAT capacity by learning in *big* size. As shown in Table 5, we can achieve superior accuracy compared to AT (12-1) with little effect on the translation latency ($\mathcal{L}_1^{\text{GPU}}$).

Models		Distillation *base*	Distillation *big*	BLEU	Speed-up
AT	*base*			27.43	1.0×
	big			28.14	0.9×
	base (12-1)			26.12	2.4×
		✓		27.34	2.5×
			✓	27.83	2.4×
NAT	*base*			23.58	16.5×
		✓		27.49	16.5×
			✓	27.56	16.5×
	big		✓	**27.89**	15.8×

Table 5: Performance comparison between AT and NAT models on the test set of WMT'14 EN→DE. The latency is measured one sentence per batch and compared with the Transformer *base*. For NAT model, we adopt CTC+VAE as the basic configuration.

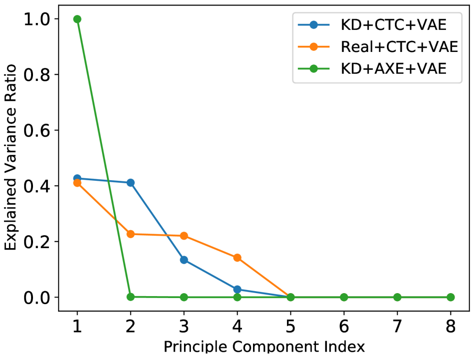

Figure 5: Principle component explained variance ratios of latent variables on WMT'14 EN→DE test set.

Effective Latent Dimensionality of Latent Variables To confirm the necessity of combining VAEs with CTC, We apply principal component analysis (PCA) (Wold et al., 1987) on the learned latent variables. More precisely, we extract the latent variables from the posterior of various models (see Table 4) on WMT'14 EN→DE test set. These main components' explained variance ratios, the percentage of variance that is attributed by each of the component, are shown in Figure 5.

First, we find that the number of effective latent dimensionality (capturing at least 95% of the total variance) is much lower than the number of latent dimensions (8 in our experiments), which indicates simply increasing the number of latent dimensions does not lead to better representations, and the ability to capture dependencies is limited. Therefore, VAEs need to be combined with other techniques e.g. KD, CTC to take effect. Also, compared to the AXE, the effective dimensionality of latent variables in CTC loss-based models is higher.

We include more analysis with qualitative examples in Appendix B.

5 Discussion and Future work

In this section, we go through the proposed four techniques again for fully NAT models. In spite of the success to close the gap with autoregressive models on certain benchmarks, we still see limitations when using non-autoregressive systems as mentioned in Table 1.

We and most of the prior research have repeatedly found that knowledge distillation (KD) is the indispensable *dependency reduction* components, especially for training fully NAT models. Nevertheless, we argue that due to the model agnostic property, KD may lose key information that is useful for the model to translate. Moreover, Anonymous (2021) pointed out KD does cause negative effects on lexical choice errors for low-frequency words in NAT models. Therefore, an alternative method that improves the training of NAT models over raw targets using such as GANs (Bińkowski et al., 2019) or domain specific discriminators (Donahue et al., 2020) might be the future direction.

Apart from KD, we also notice that the usage of CTC loss is another key component to boost the performance of fully NAT models across all datasets. As discussed in § 4.2, however, the need of up-sampling constrains the usage of our model on very long sequences or mobile devices with limited memory. In future work, it is possible to explore models to hierarchically up-sample the length with a dynamic ratio to optimize the memory usage.

Lastly, both experiments with VAE and GLAT prove that it is helpful but not enough to train NAT models with loss based on monotonic alignments (e.g. CTC) only. To work on difficult pairs such as JA-EN, it may be a better option to adopt stronger models to capture richer dependency information, such as normalizing flows (van den Oord et al., 2018; Ma et al., 2019) or non-parametric approaches (Gu et al., 2018b).

6 Related Work

Besides iterative NAT and fully NAT models, there are other works trying to improve the decoding speed of translation models from other aspects. One research line is to hybrid AT and NAT models. Wang et al. (2018) proposed a semi-autoregressive model which adopted non-autoregressive decoding locally but kept the autoregressive property in global. On the contrary, Kong et al. (2020); Huang et al. (2017) and Ran et al. (2020) introduced a local autoregressive NAT models which retained the non-autoregressive property in global.

Alternatively, there are also efforts improving the decoding speed of AT models directly. Model quantization and pruning have been widely studied as a way to improve the decoding speed (See et al., 2016; Junczys-Dowmunt et al., 2018; Aji and Heafield, 2020). Also, specialized light-weight AT model (e.g. replacing self-attention with SSRU) together with improved teacher-student training (Kim et al., 2019) are explored.

7 Conclusion

In this work, we aim to minimize the performance gap between fully NAT and AT models. We investigate *dependency reduction* methods from four perspectives and carefully unite them with necessary revisions. Experiments on three translation benchmarks demonstrate that the proposed fully NAT models achieve the SoTA performance. For future work, it is worth exploring simpler but more effective diagrams for learning NAT models. For instance, with the combination of CTC and more powerful latent variable models, it is possible to remove the necessity of knowledge distillation.

Acknowledgements

We would like to thank Jason Lee, Xuezhe Ma and Chunting Zhou for thoughtful discussion. We would also like to thank the anonymous reviewers for their time and providing helpful suggestions.

References

Alham Fikri Aji and Kenneth Heafield. 2020. Compressing neural machine translation models with 4-bit precision. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 35–42, Online. Association for Computational Linguistics.

Anonymous. 2021. Understanding and improving lexical choice in non-autoregressive translation. In *Submitted to International Conference on Learning Representations*. Under review.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C Cobo, and Karen Simonyan. 2019. High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*.

Jeff Donahue, Sander Dieleman, Mikołaj Bińkowski, Erich Elsen, and Karen Simonyan. 2020. End-to-end adversarial text-to-speech. *arXiv preprint arXiv:2006.03575*.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3515–3523. PMLR.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018a. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, Canada, April 30-May 3, 2018, Conference Track Proceedings*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pages 11181–11191. Curran Associates, Inc.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018b. Search engine guided neural machine translation.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2017. Towards neural phrase-based machine translation. *CoRR*, abs/1706.05565.

Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. 2018. Marian: Cost-effective high-quality neural machine translation in C++. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 129–135, Melbourne, Australia. Association for Computational Linguistics.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2395–2404.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020a. Non-autoregressive machine translation with disentangled context transformer. In *International Conference on Machine Learning*, pages 5144–5155. PMLR.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. 2020b. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. *arXiv preprint arXiv:2006.10369*.

Zdeněk Kasner, Jindřich Libovickỳ, and Jindřich Helcl. 2020. Improving fluency of non-autoregressive machine translation. *arXiv preprint arXiv:2004.03227*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. 2020. Incorporating a local translation mechanism into non-autoregressive translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1067–1073, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1173–1182.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Hint-based training for non-autoregressive translation.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. 2018. Parallel WaveNet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926, Stockholmsmässan, Stockholm Sweden. PMLR.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2020. Glancing transformer for non-autoregressive neural machine translation. *arXiv preprint arXiv:2008.07905*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2020. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3059–3069, Online. Association for Computational Linguistics.

Yi Ren, Jinglin Liu, Xu Tan, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. *arXiv preprint arXiv:2004.10454*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online. Association for Computational Linguistics.

Víctor M. Sánchez-Cartagena, Marta Bañón, Sergio Ortiz-Rojas, and Gema Ramírez-Sánchez. Prompsit's submission to wmt 2018 parallel corpus filtering shared task. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared*

Task Papers, Brussels, Belgium. Association for Computational Linguistics.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 198–205.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985, Long Beach, California, USA. PMLR.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, pages 10107–10116.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, pages 3016–3026.

Zhiqing Sun and Yiming Yang. 2020. An em approach to non-autoregressive conditional sequence generation. In *International Conference on Machine Learning*, pages 9249–9258. PMLR.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2826, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium. Association for Computational Linguistics.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5377–5384.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, Jun Xie, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1906.02041*.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, Conference Track Proceedings*.

Appendix

A Implementation Details

Architecture We design our fully NAT model with the hyperparameters of the *base* Transformer: 8-512-2048 (Vaswani et al., 2017). For EN→DE experiments, we also implement the NAT model in *big* size: 8-1024-4096 for comparison.

VAEs For experiments using variational autoencoders (VAE), we use the last layer encoder hidden states to predict the mean and variance of the prior distribution. The latent dimension D is set to 8, and the predicted $\boldsymbol{z}$ are linearly projected and added on the encoder outputs. Following Shu et al. (2020), we use a 3 layer encoder-decoder as the posterior network, and apply freebits annealing (Chen et al., 2016) to avoid posterior collapse.

CTC By default, we upsample the length of decoder inputs $3\times$ as long as the source for CTC, while using the golden length for other objectives (CE and AXE). We also train an additional length predictor when CTC is not used. For both cases, we use *SoftCopy* (Wei et al., 2019) which interpolated the encoder outputs as the decoder inputs based on the relative distance of source and target positions.

GLAT The mask ratio, f_{ratio}, is 0.5 for GLAT training. The original GLAT (Qian et al., 2020) assumes to work with the golden length so that it can glance at the target by placing the target word embedding to a clear corresponded inputs. It is incompatible with CTC loss where we always need longer inputs than the targets. To enable GLAT learning, we glance at target tokens from the viterbi aligned tokens ($\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}\in\beta(\boldsymbol{y})} p(\boldsymbol{\alpha}|\boldsymbol{x})$) which has the same length as the decoder inputs.

Training For both AT and NAT models, we set the dropout rate as 0.3 for EN↔DE and EN↔RO, and 0.1 for JA→EN. We apply weight decay 0.01 as well as label smoothing $\epsilon = 0.01$. All models are trained for 300K updates using Nvidia V100 GPUs with a batch size of approximately 128K tokens. We measure the validation BLEU scores for every 1000 updates, and average the last 5 checkpoints to obtain the final model.

Inference We measure the GPU latency by running the model on a single Nvidia V100 GPU, and CPU latency on Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz with 80 cores. All models are implemented on `fairseq` (Ott et al., 2019).

λ	BLEU	$\mathcal{L}_1^{\text{GPU}}$	$\mathcal{L}_{\max}^{\text{GPU}}$	$\mathcal{L}_1^{\text{CPU}}$
1.5	26.16	17.9 ms	**0.95 ms**	**66.6 ms**
2.0	26.39	17.5 ms	1.03 ms	71.6 ms
2.5	**26.54**	17.6 ms	1.16 ms	76.9 ms
3.0	26.51	**17.0 ms**	1.32 ms	81.8 ms

Table 6: Performance comparison of different upsample ratios (λ) for CTC-based models on WMT'14 EN→DE test set. All models are trained on distilled data.

B More ablation study

Upsampling Ratio (λ) for CTC Loss To meet the length requirements in CTC loss, we upsample the encoder output by a factor of 3 in our experiments. We also explore other possible values and report the performance in Table 6. The higher upsampling ratio provides a larger alignment space, leading to better accuracy. Nevertheless, with a large enough sampling ratio, a further increase will not lead to the performance increase. Because of the high degree of parallelism, $\mathcal{L}_1^{\text{GPU}}$ speed is similar among these ratios. However, the model with a larger ratio has a clear latency drop on CPU or GPU with large batches.

Representation reordering in the latent space In our main experiments, VAEs has been proven to effectively improve the performance of NAT models. Here, we perform a qualitative study to show how VAEs helps NAT models.

Ott et al. (2018) collected additional reference translations for each source sentence in the WMT'14 En→De test set. We first choose three source sentences and show the alignments between them and two of their different translations in Figure 6. In each sample, it is clear to find that the word order of the first pair is more similar to the second one (e.g., in the second sample, the verb 'light' in the source sentence is translated to the end of the second reference sentence). However, given the monotonic alignment assumption, CTC is difficult to align sentence pairs with different word orders. Then, for each sample, we extract latent variables of both sentence pairs and align them by first computing the Euclidean distance between every position and then employing the linear sum assignment algorithm (LAP).

Regarding the first pair as the baseline, we find that the latent variable is able to adjust the word order according to the input sentence pair. For example, the alignment between latent variables of

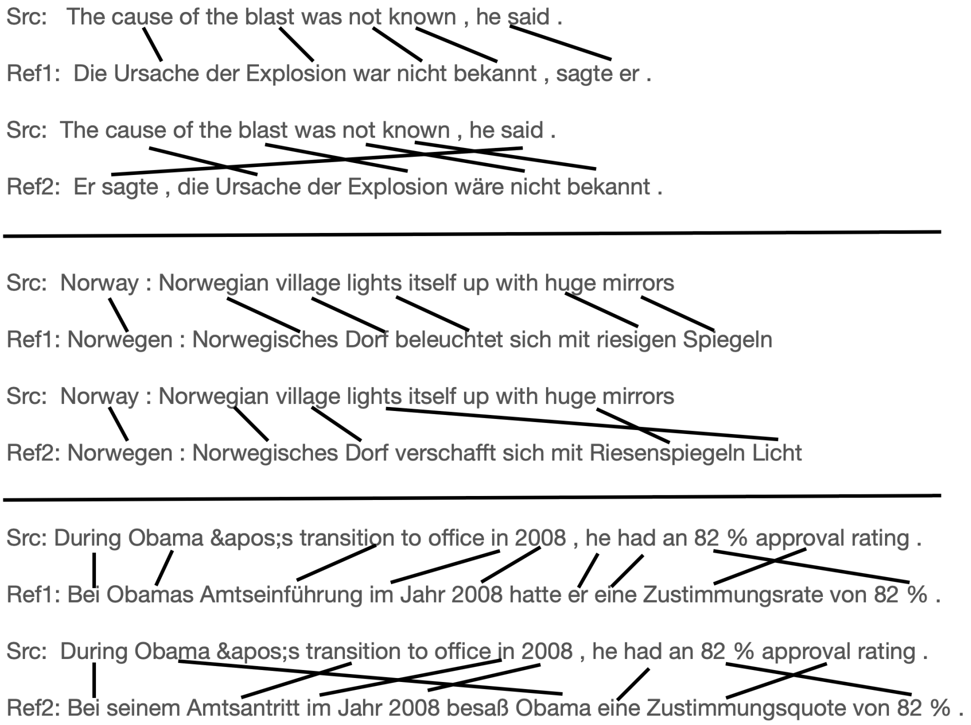
Src: The cause of the blast was not known , he said .

Ref1: Die Ursache der Explosion war nicht bekannt , sagte er .

Src: The cause of the blast was not known , he said .

Ref2: Er sagte , die Ursache der Explosion wäre nicht bekannt .

Src: Norway : Norwegian village lights itself up with huge mirrors

Ref1: Norwegen : Norwegisches Dorf beleuchtet sich mit riesigen Spiegeln

Src: Norway : Norwegian village lights itself up with huge mirrors

Ref2: Norwegen : Norwegisches Dorf verschafft sich mit Riesenspiegeln Licht

Src: During Obama 's transition to office in 2008 , he had an 82 % approval rating .

Ref1: Bei Obamas Amtseinführung im Jahr 2008 hatte er eine Zustimmungsrate von 82 % .

Src: During Obama 's transition to office in 2008 , he had an 82 % approval rating .

Ref2: Bei seinem Amtsantritt im Jahr 2008 besaß Obama eine Zustimmungsquote von 82 % .

Figure 6: Alignments between source sentences and their different translations.

the second sample is shown as: 0-0, 1-1, 2-2, 3-3, 4-9, 5-5, 6-6,7-7, 8-8, 9-4, which shows that the latent representation of the 9th position in the second pair is aligned to the 5th position of the second pair. In another word, the latent representation of the word 'lights' is reordered to the last position in the second pair's latent variable, which corresponds to the word order difference in the second pair. Therefore, given various reference information, the latent variable makes the alignment between the source and target representation more monotonic. CTC can consequently benefit from it to learn a better NAT model.

Generate, Prune, Select: A Pipeline for Counterspeech Generation against Online Hate Speech

Wanzheng Zhu and **Suma Bhat**
University of Illinois at Urbana-Champaign, USA
`wz6@illinois.edu`, `spbhat2@illinois.edu`

Abstract

Warning: *this paper contains content that may be offensive or upsetting.*

Countermeasures to effectively fight the ever increasing hate speech online without blocking freedom of speech is of great social interest. Natural Language Generation (NLG), is uniquely capable of developing scalable solutions. However, off-the-shelf NLG methods are primarily sequence-to-sequence neural models and they are limited in that they generate commonplace, repetitive and safe responses regardless of the hate speech (e.g., "Please refrain from using such language.") or irrelevant responses, making them ineffective for de-escalating hateful conversations. In this paper, we design a three-module pipeline approach to effectively improve the *diversity* and *relevance*. Our proposed pipeline first generates various counterspeech candidates by a generative model to promote *diversity*, then filters the ungrammatical ones using a BERT model, and finally selects the most *relevant* counterspeech response using a novel retrieval-based method. Extensive Experiments on three representative datasets demonstrate the efficacy of our approach in generating diverse and relevant counterspeech.

1 Introduction

Hate speech is any form of expression through which speakers intend to vilify, humiliate, or incite hatred against a group or a class of persons on the basis of some characteristics, including race, religion, skin color, sexual identity, gender identity, ethnicity, disability, or national origin (Ward, 1997; Nockleby, 2000). Its ever-growing increase on the Internet makes it a problem of significant societal concern (Williams, 2019); effective countermeasures call for not blocking freedom of speech by means of censorship or active moderation (Gagliardone et al., 2015; Strossen, 2018). A very promising countermeasure is *counterspeech*—a response that provides non-negative feedback through fact-bound arguments and broader perspectives to mitigate hate speech and fostering a more harmonious conversation in social platforms (Schieb and Preuss, 2016; Munger, 2017; Mathew et al., 2018; Shin and Kim, 2018). Counterspeech as a measure to combat abusive language online is also promoted in active campaigns such as "Get The Trolls Out".[1]

Hate Speech:	I am done with Islam and isis. All Muslims should be sent to their homeland. Britain will be better without their violence and ideology.
Expert:	I agree that ISIS is an evil aberration, but to extend this to include up to 3 million people just in the UK is just plain silly.
Commonplace:	Hate speech is not tolerated. Please review our user policies. Thank you for your cooperation.
Not relevant:	Use of the r-word is unacceptable as it demeans and insults people with disabilities.

Table 1: An illustrative example of hate speech and counterspeech.

What makes an effective counterspeech? Informed by psychosocial and linguistic studies on counterspeech (Mathew et al., 2019b) and the large number of effective counterspeech examples created by crowdsourcing (Qian et al., 2019) and by experts (Chung et al., 2019), we identify that effective counterspeech should be **diverse** and **relevant** to the hate speech instance. *Diversity* is the requirement that a collection of counterspeech should not be largely commonplace, repetitive and safe responses without regard to the target or type of hate speech (e.g., "Please refrain from using such language."). *Relevance* refers to the property that counterspeech should directly address and target the central aspects of the hate speech, enabling

---

[1] `https://getthetrollsout.org/stoppinghate`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 134–149
August 1–6, 2021. ©2021 Association for Computational Linguistics

coherent conversations rather than irrelevant or off-topic ones (e.g., the hate speech instance targets an ethnic group, while the counterspeech talks about people with disabilities). Comparative examples are shown in Table 1 where we list some counterspeech that lack diversity or relevance.

While NLG systems (in particular, sequence-to-sequence models) offer much promise for generating text at scale (Sutskever et al., 2014; Zhu et al., 2018; Lewis et al., 2020), the quality of the outputs is modest in the context of the requirements identified above. Indeed, Qian et al. (2019), the only existing quality work on counterspeech generation, has highlighted their limitations: the responses are largely commonplace and sometimes irrelevant. These limitations apply more broadly to general conversational language generation tasks, arising primarily due to the intrinsic end-to-end training nature of a single sequence-to-sequence architecture (Sordoni et al., 2015; Li et al., 2016; Serban et al., 2017; Jiang and de Rijke, 2018). Model refinements to account for these limitations have been addressed individually: improved diversity (Li et al., 2016; Xu et al., 2018) or improved relevance (Gao et al., 2019; Li et al., 2020). However, combining these improvements into a single model is not straightforward. Such is the goal of this paper.

We tackle the problem from an entirely novel angle by proposing a three-module pipeline approach, *Generate*, *Prune*, *Select* (denoted as "GPS") to ensure the generated sentences adhere to the required properties of diversity and relevance. First, the *Candidate Generation* module generates a large number of diverse response candidates using a generative model. As such, a large candidate pool is made available for selection, which accounts for improved diversity. Second, the *Candidate Pruning* module prunes the ungrammatical candidates from the candidate pool. Last, from the pruned counterspeech candidate pool, the *Response Selection* module selects the most relevant counterspeech for a given hate speech instance by a novel retrieval-based response selection method.

We demonstrate the efficacy of GPS, the first pipeline approach for counterspeech generation, by a systematic comparison with other competitive NLG approaches in generating *diverse* and *relevant* counterspeech. We derive new state-of-the-art results on three benchmark datasets by showing improved diversity and relevance using both automatic and human evaluations.

2 Proposed Model

We assume access to a corpus of labeled pairs of conversations $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where x_i is a hate speech and y_i is the appropriate counterspeech as decided by experts or by crowdsourcing. The goal is to learn a model that takes as input a hate speech x and outputs a counterspeech y. A motivating example is shown in Table 1. Most importantly, we aim at generating *diverse* and *relevant* counterspeech. We present an overview of the model in Figure 1 and describe each module in detail below.

2.1 Candidate Generation

The main goal of this module is to create a diverse candidate pool for counterspeech selection. We extract all available counterspeech instances $Y = [y_1, y_2, ..., y_n]$ from the training dataset and enlarge the counterspeech pool by a generative model.

Specifically, we utilize an RNN-based variational autoencoder (Bowman et al., 2016), that incorporates the global distributed latent representations of all sentences to generate candidates. Both the encoder and the decoder have two layers with 512 nodes each, and we use two highway network layers (Srivastava et al., 2015) to facilitate robust training. Like all other generative models, it aims to maximize the lower bound of the likelihood $\mathcal{L}$ of generating the training data Y,

$$\mathcal{L} = -KL(q_\theta(z|y) \,||\, p(z)) + \mathbb{E}_{q_\theta(z|y)}[\log p_\theta(y|z)]$$

where θ denotes all parameters of the generative model, z is a latent variable having a Gaussian distribution with a diagonal covariance matrix, p denotes the prior distribution, q denotes the posterior distribution, and KL denotes the KL-divergence (Kullback and Leibler, 1951). In the training process, we apply the KL annealing technique (Bowman et al., 2016) to prevent the undesirable stable equilibrium problem (i.e., the first term of the likelihood function $KL(q_\theta(z|y) || p(z))$ becomes zero). Upon the completion of the training, we generate candidates by simply decoding from noise ϵ sampled from a standard Gaussian distribution (i.e., $\epsilon \sim \mathcal{N}(0, 1)$).

As demonstrated by Bowman et al. (2016) (and as inferred from our own experiments described in Section 3), the generative model not only captures

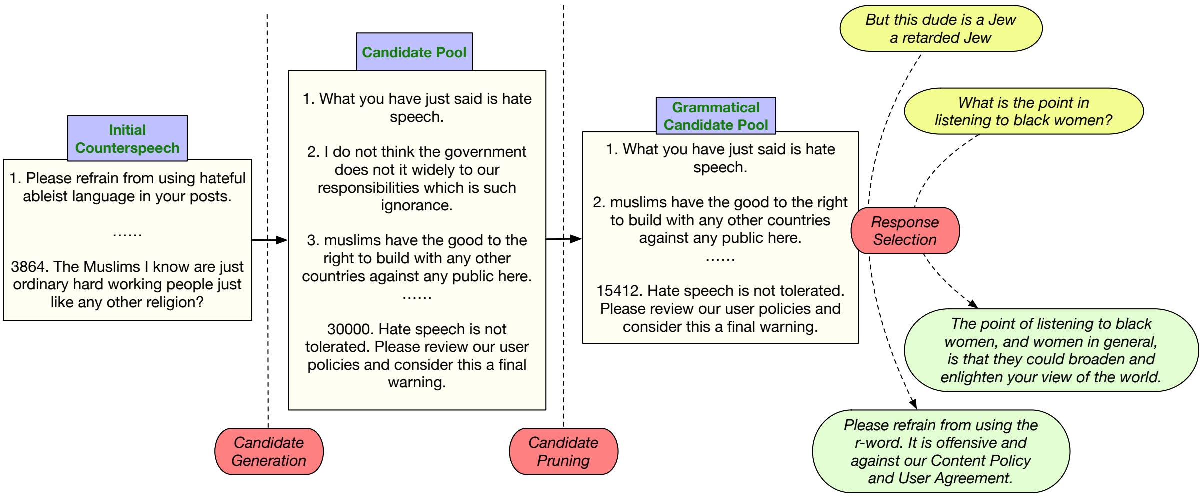

Figure 1: Overview of GPS. The red ovals correspond to the individual modules.

holistic properties of sentences such as style, topic, and high-level syntactic features, but also produces *diverse* candidates.

2.2 Candidate Pruning

Though candidates generated by such an RNN-based variational autoencoder are diverse, they are not always grammatical as pointed out by Bowman et al. (2016). Therefore, in this module, we prune the candidate list and retain only the grammatical ones. Toward this, we train a grammaticality classifier on the corpus of linguistic acceptability (CoLA) (Warstadt et al., 2018), a dataset with 10,657 English sentences labeled as grammatical or ungrammatical from linguistics publications. We select BERT (Devlin et al., 2019) as the classification model, and fine-tune it on the CoLA dataset. The choice of BERT is to best capture both the syntactic and the contextual information, and we select the 'bert-base-cased' model for its better computational efficiency.

2.3 Response Selection

We now have a collection of diverse and grammatical counterspeech responses. Finally, we aim to select the most *relevant* response to a given hate speech instance.

Taking into consideration the limited training instances that are realistically available (Chung et al., 2019; Qian et al., 2019), and inspired by the recent success of pretrained models (Devlin et al., 2019), we innovate on a pretrained response selection model for task-oriented dialogue systems (Henderson et al., 2019) and perform fine-tuning on our dataset. Henderson et al. (2019) proposed two response selection methods, but we find that neither of them is well-suited for our task.

1. *Train a response selection classifier with the negative sampling technique*: It relies on randomly drawing other candidates from the candidate pool as negative examples. However, in our task, one hate speech instance usually has multiple appropriate counterspeech instances. For example, given the hate speech in Table 1, there are many other instances that can work as quality counterspeech, such as "You cannot blame all people for the actions of a few. Banning something altogether will not solve anything." or "Does prohibition of anything ever work? I thought religious tolerance was one of our 'British values'?". Therefore, many wrongly chosen negative examples may negatively impact the inductive bias of the response selection classifier.

2. *Select by cosine similarity*: we point out that the embeddings of the input (hate speech) and the responses (counterspeech candidates) do not share the same latent vector space and therefore, the learned embeddings and their cosine similarities may not fully serve the purpose of relating the response to the input.

Therefore, instead of adopting the two available methods directly, we improve on the second one by fusing the latent spaces of the input and the responses, inspired by Gao et al. (2019). Specifically, we propose to learn a linear embedding mapping from the latent space of the responses to the latent space of the input, and then select the best response by cosine similarity. Mathematically, we use e_x to denote the input embedding and e_y to denote the response embedding. We aim to learn a linear mapping from e_y to e'_y, where $e'_y = (W + BI) \cdot e_y$, W and B are learnable parameters, and I is an identity matrix. We learn the mapping such that the sum of the cosine similarities between e_x and e'_y for the training data is maximized. By way of this transformation, e'_y now maps the vector space of the responses to that of the input, and thus allows the pretrained model to effectively utilize the discriminative power of the sentence embeddings. We empirically observe that the linear mapping works well and leave other advanced mapping techniques for future work.

3 Empirical Evaluation

In this section, we empirically evaluate the performance of our proposed approach and a set of baseline models.

3.1 Experimental Setup

Datasets: We use the benchmark datasets collected by Qian et al. (2019), which are fully-labeled hate speech intervention datasets collected from Reddit and Gab, comprising 5,257 and 14,614 hate speech instances respectively. We use the filtered conversation setting in Qian et al. (2019), which includes the posts labeled as hate speech only and discards other non-hateful conversations. Besides, we use the English language portion of the CONAN dataset (Chung et al., 2019), which contains counterspeech for 408 hate speech instances, written by experts trained on countering hatred. The Reddit, Gab and CONAN datasets have on average 2.66, 2.86 and 9.47 ground truth counterspeech for each hate speech respectively.

Training Data: Since each hate speech can have multiple ground truth counterspeech, we follow Qian et al. (2019) to dis-aggregate the counterspeech and construct a pair (hate speech, counterspeech) for each of the ground truth counterspeech in each dataset. Given a counterspeech dataset, we randomly choose 70% (hate speech, counterspeech) pairs for model training, 15% for cross validation and the rest 15% for testing.

Baselines: We compare our proposed approach with the following competitive baseline models:

1. Seq2Seq (Sutskever et al., 2014; Cho et al., 2014) is a widely used neural model for language generation. We use 2 bidirectional Gated Recurrent Unit (GRU) layers for the encoder and 2 GRU layers followed by a 3-layer neural network as the decoder.
2. Maximum Mutual Information (MMI) (Li et al., 2016) is a diversity-promoting approach for neural conversation models. We implement the MMI-bidi model (Li et al., 2016) and adopt incremental learning (Ranzato et al., 2016) to facilitate robust training.
3. SpaceFusion (Gao et al., 2019) optimizes both diversity and relevance by introducing a fused latent space, where the direction and distance from the predicted response vector roughly match the relevance and diversity, respectively. We align the direction parameter with the ground truth counterspeech. To better exercise the diversity power, we randomly choose the distance parameter at each time of generation.
4. BART (Lewis et al., 2020) is the state-of-the-art pre-trained sequence-to-sequence model for language generation. It has a standard Transformers-based neural machine translation architecture which can be seen as generalizing BERT (Devlin et al., 2019), GPT (Radford et al., 2018) and many pretraining schemes. We fine-tune the BART model on our training data.

We compare with Seq2Seq since they are initially proposed and used by Qian et al. (2019).[2] We select MMI, SpaceFusion and BART as baselines because they are the state-of-the-art models in promoting diversity, optimizing both diversity and relevance, and generating quality language respectively.

3.2 Evaluation

We evaluate all model outputs along three dimensions: diversity, relevance and language quality. *Diversity* refers to vocabulary richness, variety in expression and the extent to which the response is dissimilar from the rest in a generated collection of responses. *Relevance* captures the extent to which the counterspeech addresses the central aspect of the hateful message and makes a coherent conversation towards mitigating the hate speech. A low relevance score means that the counterspeech is irrelevant to the hate speech or off-topic (e.g., the hate speech talks about LGBTQ whereas the counterspeech is related to religious beliefs). *Language*

---

[2] We do not include the results of the variational auto-encoder model and the reinforcement learning model in Qian et al. (2019) for comparison as they has very similar performance as Seq2Seq. Readers are referred to Qian et al. (2019) for detailed performance.

		Diversity						Relevance					LQ.
		Dist-1	**Dist-2**	**Ent-1**	**Ent-2**	**SB1***	**SB2***	**B2**	**R2**	**MS**	**BS**	**BM25**	**GR**
CONAN	**Seq2Seq**	**0.06**	0.23	5.12	6.63	0.54	0.30	3.4	3.0	4.4	0.83	2.66	0.38
	MMI	**0.06**	0.23	4.88	6.41	0.57	0.35	2.9	2.3	3.9	0.82	1.63	0.33
	SpaceFusion	0.00	0.00	1.06	1.86	0.98	0.98	0.0	0.0	-14.2	0.76	0.12	0.38
	BART	0.04	0.23	**5.98**	**7.80**	0.52	0.26	3.9	3.6	7.1	0.84	1.86	**0.71**
	GPS	**0.06**	**0.27**	5.77	7.41	**0.43**	**0.19**	**7.1**	**6.5**	**10.9**	**0.85**	**5.43**	**0.71**
Reddit	**Seq2Seq**	0.04	0.24	5.07	6.61	0.58	0.31	6.5	4.0	6.8	0.85	0.14	0.64
	MMI	0.05	0.32	5.11	6.76	0.56	0.29	6.4	4.0	6.9	0.85	0.14	0.56
	SpaceFusion	0.00	0.02	2.73	4.16	0.87	0.76	0.9	0.0	-2.5	0.79	0.16	0.26
	BART	0.03	0.19	5.08	6.63	0.69	0.55	7.8	6.9	**7.8**	0.86	0.83	0.72
	GPS	**0.09**	**0.53**	**5.74**	**7.61**	**0.41**	**0.15**	**8.1**	**7.1**	**7.8**	**0.87**	**2.58**	**0.75**
Gab	**Seq2Seq**	0.02	0.17	5.14	6.71	0.56	0.30	7.5	5.0	6.7	0.86	0.14	0.67
	MMI	0.02	0.17	5.28	6.82	0.55	0.30	5.8	3.6	6.2	0.85	0.18	0.65
	SpaceFusion	0.00	0.01	3.72	4.84	0.81	0.73	1.8	0.1	0.0	0.82	0.17	0.21
	BART	0.03	0.17	5.42	7.25	0.60	0.38	6.9	**6.4**	**6.8**	0.86	0.81	0.72
	GPS	**0.06**	**0.40**	**5.82**	**7.83**	**0.39**	**0.15**	**7.6**	**6.4**	**6.8**	**0.87**	**1.94**	**0.76**

Table 2: Automatic evaluation results. An asterisk * by the metric name indicates that the metric favors smaller values. Best results are in bold. LQ.: Language Quality; SB1: Self-BLEU-1; SB2: Self-BLEU-2; B2: BLEU-2; R2: ROUGE-2; MS: MoverScore; BS: BERTScore; GR: GRUEN.

		Div.	**Rel.**	**LQ.**
CONAN	**Seq2Seq**	0.50	0.22	0.06
	MMI	0.55	0.08	0.02
	BART	0.40	0.73	0.65
	GPS	**0.80**	**0.83**	**0.66**
Reddit	**Seq2Seq**	0.25	0.23	0.38
	MMI	0.35	0.23	0.35
	BART	0.00	0.47	**0.51**
	GPS	**1.00**	**0.58**	0.48
Gab	**Seq2Seq**	0.35	0.36	0.31
	MMI	0.55	0.34	0.27
	BART	0.10	0.42	0.35
	GPS	**0.80**	**0.47**	**0.36**

Table 3: Human evaluation results. Div.: Diversity; Rel.: Relevance; LQ.: Language Quality.

quality measures whether the generated responses are grammatical, fluent and readable.

3.2.1 Automatic Evaluation

We evaluate *diversity* by distinct n-grams (**Dist-n**) (Li et al., 2016), Entropy (**Ent-n**) (Zhang et al., 2018) and **Self-BLEU** (Zhu et al., 2018). For *relevance*, we compare 1) the generated response with the ground truth counterspeech by **BLEU** (Papineni et al., 2002) and **ROUGE** (Lin and Hovy, 2003; Lin, 2004) for syntactic similarity, and by **MoverScore** (Zhao et al., 2019) and **BERTScore** (Zhang et al., 2020a) for semantic similarity; 2) the generated response with the hate speech by **BM25** (Manning et al., 2008), a relevance estimation function widely used in information retrieval. We adopt GRUEN (Zhu and Bhat, 2020) to evaluate the *language quality*. Note that larger scores indicate better quality, except for Self-BLEU.

3.2.2 Human Evaluation

Following Qian et al. (2019), we ask human annotators to perform "winner selection" instead of numeric ratings. We randomly sampled 100 hate speech instances from each dataset's test collection, and corresponding model-generated counterspeech outputs from Seq2Seq, MMI, BART, and our model.[3] Two human annotators (native English speakers and sensitive about hate speech online) were presented with one hate speech and the four generated outputs, and were asked to evaluate the outputs for relevance and language quality separately. As for diversity, the annotators were presented with 10 responses for each of the four models each time, and were asked to select the model with the best diversity. For each of the three dimensions, the *best* output received a 1 (ties were allowed, i.e., multiple outputs could receive a 1) and the rest received a 0.

3.3 Results

We obtain the results of our model (denoted as "GPS") by generating 30k, 30k, 40k candidates for the CONAN, Reddit and Gab datasets respectively.[4] After the Candidate Pruning module, we

---

[3] To save human annotation effort, we asked human raters to annotate only the best 4 models w.r.t. automatic metrics, and therefore leave SpaceFusion out of annotation.

[4] The number of generated candidates are selected based on the performance on the cross validation datasets.

Hate Speech	Generated Counterspeech by GPS
(a): All the Muslims should have been punished for yesterday, not only a few. Citizen will take action if the government does not do anything.	You cannot blame all people for actions of a few. Banning something altogether will not solve anything.
(b): The world would be a better place without Muslims. They are only killing and raping our children.	The world would actually be a very dark place without Muslims who contribute a lot to our society. What about our Muslim doctors, scientists, philanthropists, actors, job-creators?
(c): What is the point in listening to black women? They are all retarded.	The point of listening to black women, and women in general, is that they could broaden and enlighten your view of the world.

Table 4: Examples of the generated counterspeech

have 15.4k, 17.9k and 25.4k grammatical candidates for each dataset respectively.

The results by automatic metrics and human evaluation metrics are presented in Table 2 and Table 3 respectively. Overall, GPS has the best diversity with significant margins than the baselines. For relevance, GPS has slightly better performance for BLEU, ROUGE, MoverScore and BERTScore, while has much better performance on BM25. This implies the counterspeech generated by GPS are more related to the hate speech and therefore, make more coherent conversations. Examples of counterspeech generated by GPS are presented in Table 4. We find that GPS is able to generate diverse and relevant rather than merely commonplace responses, such as “Please refrain from using such language”. Comparative case studies for different baseline models are shown in Appendix A.4. Therefore, we conclude that GPS has the best diversity and relevance, compared to the baselines. Besides, GPS has comparable language quality with the best baseline model—BART.

Among these baselines, BART is the strongest one with much better relevance and language quality. Yet, BART still suffers from the diversity issue, as discussed in Section 4.3. SpaceFusion has very poor results overall, though a manual inspection of the latent space fusion visualization suggests otherwise. One explanation is that SpaceFusion, with substantially more parameters compared with the Seq2Seq model may not have had sufficient training instances for its optimal performance. In their own experiments, Gao et al. (2019), demonstrate that SpaceFusion worked well on two datasets with 0.2M and 7.3M conversations, which is at least one to two orders of magnitude larger than our dataset. If provided with more training data, SpaceFusion could possibly be a strong candidate too. In comparison, though BART is an even more complicated model with 139M parameters, it was pre-trained on the BooksCorpus dataset (Zhu et al., 2015) with over 7,000 unique unpublished books and has the fine-tunable property.

3.4 Ablation Study

We compare with the following ablations of GPS and show the results in Figure 2.

1. G-BART: instead of generating the candidates by the RNN-based variational autoencoder (Bowman et al., 2016), we generate the candidates by BART (Lewis et al., 2020).
2. P-no: we exclude the pruning module and make all generated candidates available for selection.
3. S-tfidf: we select the most relevant response by tf-idf on raw texts.
4. S-cos: we exclude the latent space fusion step and select the best response by the cosine similarity of the response embeddings and the hate speech embeddings (Henderson et al., 2019).
5. S-neg: we use the negative sampling technique to train a response selection classifier (Henderson et al., 2019).

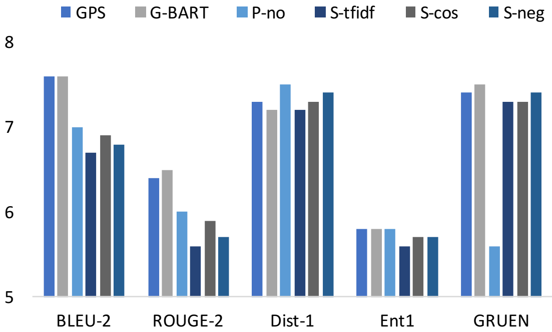

Figure 2: Ablation study. Plots show average results across all three datasets. We scale Dist-1 by 100 times and GRUEN by 10 times for better visualization.

G-BART has almost the same performance as GPS. Therefore, we select the RNN-based variational autoencoder for candidate generation for its better computational efficiency. Compared with the full model, though P-no has slightly better performance on diversity, it performs poorly on both relevance and language quality. Three ablation methods for response selection have similar performance. They have comparable performance to GPS on diversity and language quality, but worse results on relevance.

The ablation study demonstrates the significance of the Candidate Pruning module and our proposed Response Selection method. It also implies that diversity, language quality and relevance are improved by the Candidate Generation module, the Candidate Pruning module, and the Response Selection module respectively.

3.5 Generation *vs.* Selection

This section studies the relationship between the Candidate Generation module and the Response Selection module. The more candidates we generate, the more diversity the model gains potentially. However, one might think that the selection model may suffer from a very large candidate pool and result in poor relevance. Empirically as shown in Figure 3, we find that once the number of candidates generated has passed a threshold, the diversity (i.e., the blue line) almost converges. Besides, we also find the relevance is not compromised and relatively stable even with more candidates generated beyond the threshold. Therefore, we select the number of candidates at the "elbow" point based on the performance on the validation dataset, for efficient computations.

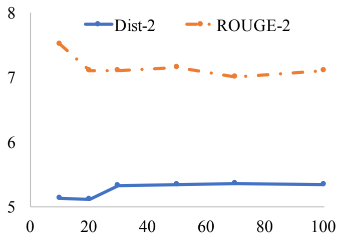

Figure 3: Dist-2 and ROUGE-2 *vs.* Number of candidates (in thousands) generated on the Reddit dataset. We scale Dist-2 by 10 times for better visualization.

3.6 Explicit Relevance (BM25) *vs.* Diversity

Based on the reasoning that models with better BM25 scores should specifically address the central aspect of hate speech and thus produce dissimilar responses for different hate speech, we hypothesize that models with better BM25 should generate more diverse responses. Therefore, we present scatter plots of BM25 and diversity scores for all five models (in Section 3.1) on all three datasets altogether in Figure 4, resulting 15 data points per subfigure. We find that BM25 and diversity have a reasonably strong correlation (Pearson's Correlation scores are 0.47 and -0.60 for Ent-1 and Self-BLEU-1 respectively).

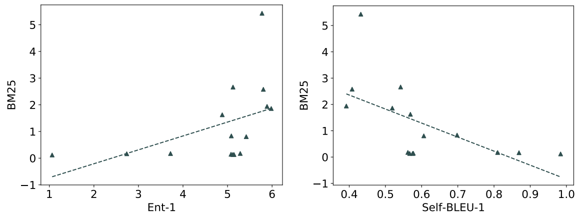

Figure 4: BM25 *vs.* Diversity. Each data point denotes a (diversity, BM25) pair for one model on one dataset, and the dotted lines indicate regression lines.

4 Related Work

We focus on three areas to the problem of hate speech and its countermeasures, i.e. (i) psychosocial analysis, (ii) automatic counterspeech generation, and more broadly, (iii) conversational language generation.

4.1 Psychosocial Analysis of Counterspeech

Effectiveness of Counterspeech: There is a significant research interest in understanding the effectiveness of counterspeech to fight hatred and de-escalate the conversation as evidenced by a growing number of recent studies (Schieb and Preuss, 2016; Munger, 2017; Mathew et al., 2018). Munger (2017) found that subjects who were educated by high-follower white males, significantly reduced their use of racist slurs on Twitter. Schieb and Preuss (2016) studied counterspeech on Facebook via a simulation, and concluded that counterspeech could have a considerable impact on a given audience, and the impact was a function of the proportion of hate speakers in the audience. In a subsequent study, Mathew et al. (2018) recorded the case of a user who, after seeing the counterspeech posted to her hateful messages on Twitter, openly apologized for her actions. Besides academia, some organizations are also set to promote countermeasures via campaigns such as the no hate speech movement[5] and the Facebook counterspeech campaign[6]. Therefore, Benesch (2014); Mathew et al. (2019b) suggest that counterspeech can be regarded as one of the most promising and "constitutionally preferred" approaches to hate speech. In addition, counterspeech could be likened to the effect of prosocial active bystanders in face-to-face bullying scenarios, where bystander intervention (speaking on behalf of the victim) has been found to successfully abate victimization most of the time (Oconnell et al., 1999; Craig et al., 2000).

Psychosocial and Linguistic Aspects: Besides the effectiveness of counterspeech, psychosocial and linguistic aspects of both counterspeech and hate speech have been actively studied by Mathew et al. (2019a); Siegel (2019); Schieb and Preuss (2016); Weingartner and Stahel (2019); Mathew et al. (2018). For instance, Mathew et al. (2019b) performed detailed psycholinguistic analysis on counterspeech, compared the effectiveness of different counterspeech strategies, and revealed some important insights, such as counterspeech comments receive much more "likes" on YouTube compared to the non-counterspeech comments, suggesting a communal empathy for the target of hate speech. Besides, Mathew et al. (2019b); Chung et al. (2019) studied different strategies (e.g., call for influential users) to produce effective counterspeech. Mathew et al. (2018) found that the hate tweets by verified accounts were much more viral as compared to tweets by non-verified accounts, by analyzing the hate speech and counterspeech accounts on Twitter. Mathew et al. (2019a) study how hate speech spreads in online social media. More recently, Sap et al. (2020) studied pragmatic formalisms to capture ways in which people express social biases and power differentials in language, permitting a broader computational framework for processing hate speech.

4.2 Counterspeech Generation

Though the effectiveness of counterspeech is well-motivated from both psychosocial and linguistic perspectives, limits to manual counterspeech generation at scale have prompted automatic generation of counterspeech, an area that has received little attention to date. The first key challenge in this direction is the creation of reliable counterspeech datasets of high quality. Mathew et al. (2019b) collected counterspeech from YouTube comments, but omit the hate speech associated with each counterspeech. Such a dataset may be good for psychosocial and linguistic analysis, but is not sufficient for training an NLG model. To enable model training, Qian et al. (2019) released two fully-labeled datasets collected from Reddit and Gab. Besides, Chung et al. (2019) collected a quality dataset where the counterspeech instances are written by trained experts and are meant to fight each hate speech and de-escalate a hateful situation. Recently, Tekiroglu et al. (2020) proposed an approach to collect counterspeech responses in a more effective manner, but have not yet released a quality dataset. In our work, we conduct the experiments on all the publicly available datasets (i.e., (Chung et al., 2019; Qian et al., 2019)) to date, to the best of our knowledge.

Research on NLG algorithms for counterspeech generation is still in its infancy. Qian et al. (2019) made the only initial attempt and proposed the use of three neural models to generate counterspeech. However, they only experimented with the most basic model architectures (e.g., Seq2Seq) to prove the feasibility of the task, and leave the performance improvement for future work. In our work, we extend their results by studying more advanced architectures, identifying principal dimensions of effective counterspeech, and proposing a novel pipeline to better solve the problem. To the best of our knowledge, this paper represents the first successful pipeline model for counterspeech generation.

From the technical perspective, our work shares some high-level similarities with Tekiroglu et al. (2020) since we both use generative models to generated candidates. However, we would like to highlight that our essential goals are different. Tekiroglu et al. (2020) aim to *collect quality data* by enabling language models and studying human annotation strategies, while we aim to *generate counterspeech* to a given hate speech.

4.3 Conversational Language Generation

Counterspeech generation is broadly related to conversational language generation, where most of the best performing approaches are based on neural models trained in a sequence-to-sequence manner (See et al., 2019a). Despite the good performance of these models, one of their widely acknowledged intrinsic drawbacks is the generation of safe and commonplace responses (Sordoni et al., 2015) due

---

[5] `https://www.nohatespeechmovement.org`

[6] `https://counterspeech.fb.com`

to improper objective function (Li et al., 2016), lack of model variability (Serban et al., 2017; Zhao et al., 2017), weak conditional signal (Tao et al., 2018), and model over-confidence (Jiang and de Rijke, 2018). Such tendency has prompted the study of methods that improve diversity and has resulted in a wide variety of solutions, such as optimizing a different loss function (Li et al., 2016; Zhang et al., 2018), varying the latent space (Shao et al., 2019; Gao et al., 2019), utilizing adversarial learning (Xu et al., 2018; Shetty et al., 2017; Shi et al., 2018), and leveraging non-conversational information (Wu et al., 2020; Su et al., 2020; Tu et al., 2019). Our work is different from all above in that we adopt a *pipeline* model which promotes diversity by generating a variety of candidates. As such, it does not have the aforementioned intrinsic drawback of a sequence-to-sequence model.

5 Conclusion and Future Work

We proposed a three-module pipeline — *Generate*, *Prune*, *Select* for counterspeech generation against online hate speech. Empirical evaluation on three datasets demonstrates that our model is effective in producing diverse and relevant counterspeech.

Future works could include the following two directions: 1) stylistic counterspeech generation: Mathew et al. (2019b) find that different counterspeech styles/strategies may be needed for different hate speech topics and therefore, it would be interesting to develop new techniques to generate the most effective style of counterspeech for each hate topic. We think this could be a natural extension to our proposed model, since we can utilize a style classifier in the Candidate Pruning module. 2) system deployment: studying the real social impacts of automatic counterspeech generation in reducing online hate speech via system deployment and the actual activity monitoring can directly inform research in this area.

Reproducibility: Our code is available at `https://github.com/WanzhengZhu/GPS`.

Acknowledgments

We thank the anonymous reviewers for comments on earlier drafts that significantly helped improve this manuscript. We thank Pooja Bhagchandani and Neha Prabhu for annotating the model outputs. This research was supported by the National Science Foundation award CNS-1720268.

Ethical Considerations

We recognize that studying counterspeech generation necessarily requires us to confront online content that may be offensive or disturbing. However, deliberate avoidance does not eliminate such problems (Sap et al., 2020). Since the effectiveness of counterspeech has already been widely studied in Section 4.1, our work makes a positive step towards automating the process, which could potentially educate hate speakers and mitigate hate speech online. Besides, the automation process could help reduce the amount of human work and therefore, potential harm to human moderators (Barrett, 2020; Zhu et al., 2021). In addition, the collective analysis over large corpora and counterspeech can also be insightful for educating people on reducing the usage of hate speech consciously or unconsciously in their language.

Risks in deployment: The deployment of counterspeech generation (e.g., (de los Riscos and DHaro, 2020)) should be done after paying attention to several ethical aspects some of which we list below.

- Social and racial bias (Sap et al., 2020): Does the model have any pragmatic implications which project unwanted social or racial biases and stereotypes onto online users?
- Fairness (Mitchell et al., 2019; Corbett-Davies et al., 2017): can the model ensure fairness for different demographic groups or speakers of different forms/dialects/vernaculars of English?
- Failure cases: are there any failure cases, which could further incite more aggressive hate speech? It is crucial to ensure that counterspeech deployment does not escalate a given hateful situation.
- Evaluation metrics (Corbett-Davies et al., 2017): the present study improves upon prior works by more comprehensive evaluations on diversity, relevance and language quality. However, there is a chance that the three criteria are sufficient for deployment in a realistic setting and there may be additional criteria associated with their effectiveness.
- Potential nefarious side effects and misuse potential (Lau et al., 2020): how to ensure that our model is not misused for other unwanted purposes?

Given the limited scope of the present study, we call for attention to these aspects by way of well-designed experiments before deploying counterspeech generation bots.

Regulatory standpoint on the present study: Institutional Review Board (IRB) gave us clear feedback on what is considered human research and thus subject to IRB review. Analyses relying on user-generated content do not constitute human-subject research, and are thus not the purview of the IRB, as long as 1) the data analyzed are posted on public fora and were not the result of direct interaction from the researchers with the people posting, 2) there are no private identifiers or personally identifiable information associated with the data, and 3) the research is not correlating different public sources of data to infer private data.[7] All of these conditions apply to the present study. Additionally, the hate speech and counterspeech instances were secondary data, previously collected by Qian et al. (2019); Chung et al. (2019) and the annotators in our study were evaluating the quality of the generated sentences only.

Risks in annotation: The data we use in this paper were posted on publicly accessible websites, and do not contain any personally identifiable information (i.e., no real names, email addresses, IP addresses, etc.). The annotators were undergraduate assistants in the lab receiving research credit for their annotation and were blind to the systems they were annotating. They were warned about the offensive content before they read the data, and were informed that they could quit the task at any time if they were uncomfortable with the content.

References

Paul M. Barrett. 2020. Who moderates the social media giants? *Center for Business*.

Susan Benesch. 2014. Countering dangerous speech: New ideas for genocide prevention. *Washington, DC: United States Holocaust Memorial Museum*.

Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*.

Yi-Ling Chung, Elizaveta Kuzmenko, Serra Sinem Tekiroglu, and Marco Guerini. 2019. Conan-counter narratives through nichesourcing: a multilingual dataset of responses to fight online hate speech. In *Proceedings of ACL*.

Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Wendy M Craig, Debra Pepler, and Rona Atlas. 2000. Observations of bullying in the playground and in the classroom. *School Psychology International*, 21(1):22–36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Iginio Gagliardone, Danit Gal, Thiago Alves, and Gabriela Martinez. 2015. *Countering online hate speech*. Unesco Publishing.

Xiang Gao, Sungjin Lee, Yizhe Zhang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Jointly optimizing diversity and relevance in neural response generation. In *Proceedings of NAACL-HLT*.

Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. 2019. Training neural response selection for task-oriented dialogue systems. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Shaojie Jiang and Maarten de Rijke. 2018. Why are sequence-to-sequence models so dull? *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Jey Han Lau, Timothy Baldwin, et al. 2020. Give me convenience and give her death: Who should decide what uses of nlp are appropriate, and on what basis? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Association for Computational Linguistics (ACL))*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*.

---

[7] This position is in line with Title 45 of the Code of Federal Regulations, Part 46 (45 CFR 46), which defines human research.

Xin Li, Piji Li, Wei Bi, Xiaojiang Liu, and Wai Lam. 2020. Relevance-promoting language model for short-text conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *North American Association for Computational Linguistics (NAACL)*.

Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge University Press.

Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. 2019a. Spread of hate speech in online social media. In *Proceedings of the 10th ACM Conference on Web Science*.

Binny Mathew, Navish Kumar, Pawan Goyal, Animesh Mukherjee, et al. 2018. Analyzing the hate and counter speech accounts on twitter. *arXiv preprint arXiv:1812.02712*.

Binny Mathew, Punyajoy Saha, Hardik Tharad, Subham Rajgaria, Prajwal Singhania, Suman Kalyan Maity, Pawan Goyal, and Animesh Mukherjee. 2019b. Thou shalt not hate: Countering online hate speech. In *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

Kevin Munger. 2017. Tweetment effects on the tweeted: Experimentally reducing racist harassment. *Political Behavior*, 39(3):629–649.

John T Nockleby. 2000. Hate speech. *Encyclopedia of the American Constitution*, 3(2):1277–1279.

Paul Oconnell, Debra Pepler, and Wendy Craig. 1999. Peer involvement in bullying: Insights and challenges for intervention. *Journal of Adolescence*, 22(4):437–452.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*.

Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding, and William Yang Wang. 2019. A benchmark dataset for learning to intervene in online hate speech. In *Proceedings of EMNLP-IJCNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Agustín Manuel de los Riscos and Luis Fernando DHaro. 2020. Toxicbot: A conversational agent to fight online hate speech. *Conversational Dialogue Systems for the Next Decade*, 704:15.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Carla Schieb and Mike Preuss. 2016. Governing hate speech by means of counterspeech on facebook. In *66th ICA Annual Conference*.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. 2019a. Do massively pretrained language models make better storytellers? In *Proceedings of Computational Natural Language Learning (CoNLL)*.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019b. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, et al. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of EMNLP-IJCNLP*.

Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Toward diverse text generation with inverse reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Youngsoo Shin and Jinwoo Kim. 2018. Data-centered persuasion: Nudging user's prosocial behavior and designing social innovation. *Computers in Human Behavior*, 80:168–178.

Alexandra A Siegel. 2019. Online hate speech. *Social Media and Democracy*, page 56.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Nadine Strossen. 2018. *Hate: Why we should resist it with free speech, not censorship*. Oxford University Press.

Hui Su, Xiaoyu Shen, Sanqiang Zhao, Xiao Zhou, Pengwei Hu, Randy Zhong, Cheng Niu, and Jie Zhou. 2020. Diversifying dialogue generation with non-conversational text. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao, and Rui Yan. 2018. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Serra Sinem Tekiroglu, Yi-Ling Chung, and Marco Guerini. 2020. Generating counter narratives against online hate speech: Data and strategies. In *Proceedings of ACL*.

Lifu Tu, Xiaoan Ding, Dong Yu, and Kevin Gimpel. 2019. Generating diverse story continuations with controllable semantics. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*.

Kenneth D Ward. 1997. Free speech and the development of liberal virtues: An examination of the controversies involving flag-burning and hate speech. *U. Miami L. Rev.*, 52:733.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Sebastian Weingartner and Lea Stahel. 2019. Online aggression from a sociological perspective: An integrative view on determinants and possible countermeasures. In *Proceedings of the Third Workshop on Abusive Language Online*.

Matthew Williams. 2019. Hatred behind the screens: A report on the rise of online hate speech.

Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2020. Diverse and informative dialogue generation with context-specific commonsense knowledge awareness. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. BERTScore: Evaluating text generation with bert. In *International Conference on Learning Representations (ICLR)*.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *Advances in Neural Information Processing Systems (NIPS)*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020b. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Wanzheng Zhu and Suma Bhat. 2020. GRUEN for evaluating linguistic quality of generated text. In *Empirical Methods in Natural Language Processing: Findings (Findings of EMNLP)*.

Wanzheng Zhu, Hongyu Gong, Rohan Bansal, Zachary Weinberg, Nicolas Christin, Giulia Fanti, and Suma Bhat. 2021. Self-supervised euphemism detection and identification for content moderation. In *42nd IEEE Symposium on Security and Privacy*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

A Appendix

A.1 Selection of Automatic Metrics

A.1.1 Diversity

We measure distinct n-grams (Dist-n) (Li et al., 2016), Entropy (Ent-n) (Zhang et al., 2018) and Self-BLEU (Zhu et al., 2018) for diversity.

Dist-n reflects the vocabulary diversity by simply dividing the number of unique n-grams by the total number of n-grams of model output. One limitation of Dist-n is that it fails to accommodate the frequency difference of n-grams. To accommodate the frequency difference of n-grams, we also use the Entropy metric (Zhang et al., 2018), which reflects how evenly the empirical n-gram distribution is.

Though Dist-n and Ent-n evaluate the vocabulary diversity well, they fail to evaluate the inter-response diversity. For instance, they favor responses with diverse n-grams even when they are highly similar with the rest of the responses. Therefore, to accommodate such inter-response diversity, we resort to use Self-BLEU (Zhu et al., 2018) to evaluate how one response resembles the rest in a generated collection of responses. Self-BLEU regards one generated sentence as the hypothesis and the other generated sentences as the reference, and calculates the BLEU score for every generated sentence. Therefore, the smaller the Self-BLEU, the better the diversity.

A.1.2 Relevance

Most existing works measure relevance *implicitly* by BLEU and ROUGE, a set of metrics evaluating syntactic similarity between the ground truth and the generated output. They assume that the ground truth is highly relevant to the conversational input (i.e., it refers to the hate speech in our task) and therefore, the "closer" the generated output is to the ground truth, the more relevant the output is to the hate speech instance.

Explicit relevance evaluation (i.e., relatedness between the conversational input and the generated output) has been studied in only a few existing works. For instance, See et al. (2019b) and Zhang et al. (2020b) ask human annotators to evaluate relevance explicitly. Li et al. (2020) propose to use HIT-Q and HIT-R, two hit rate based metrics which require hand-crafted rules. For automatic metrics, Gao et al. (2019) propose to use "Precision" to measure relevance. However, we consider "Precision" inappropriate in our problem setting, because it only measures the relationship between the generated output and the ground truth, but not the relationship between the generated output and the conversational input.

Since there is no consensus on which automatic metric best serves the purpose of explicit relevance, we select BM25 (Manning et al., 2008) — a relevance estimation function widely used in information retrieval. Besides, we follow existing works to evaluate implicit relevance by measuring BLEU and ROUGE for syntactic similarity, and MoverScore and BERTScore for semantic similarity.

A.1.3 Language Quality

GRUEN (Zhu and Bhat, 2020) is the only existing open-source unsupervised metric that measures the language quality of generated text. It requires no reference to compare with and has been shown to correlate well with human annotations on a variety of language generation tasks.

A.2 Relevance and Diversity *vs.* Number of Epochs

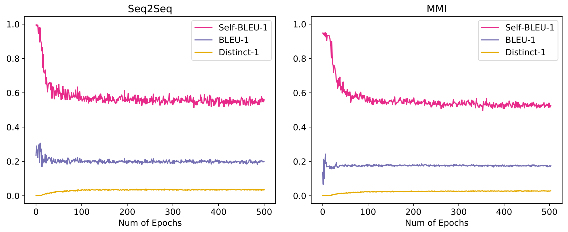

Figure 5: Effects of number of epochs for the Seq2Seq and MMI models on the Gab dataset.

In order to see how the robustness of baseline neural models changes with the number of epochs, we plot relevance and diversity measured by automatic metrics against the number of epochs for Seq2Seq and MMI in Figure 5. For each sub-figure, the middle line indicates relevance while the other two lines indicate diversity.

We note that the diversity increases with the number of epochs, until converges at about 100 epochs. Surprisingly, the relevance has a spike in the initial few training epochs and then converges to a lower score at about 50 epochs. We inspected the results where the spike occurs and observed that the model learns to produce only a few general repetitive counterspeech (e.g., "Hi there, please refrain from using derogatory comments in the thread. They are hurtful and unwanted. If you continue, Admin will be alerted.") to all hate speech. Such general responses, though result in high relevance scores (e.g., BLEU and ROUGE), are not yet effective due to the lack of diversity. With more training epochs, the models learn to produce more diverse responses at the cost of reduced BLEU and ROUGE.

Note that all results in this paper (e.g., Table 2) are reported when both relevance and diversity stabilize (i.e., 100 epochs of training). Qian et al. (2019) report higher BLEU and ROUGE scores than our results in Table 2 for the Seq2Seq model and we suspect that their reported results were obtained with only a few epochs of training.

A.3 Efficiency Comparison

We implemented all models in Python 3.7 and conducted all the experiments on a computer with twenty 2.9 GHz Intel Core i7 CPUs and one GeForce GTX 1080 Ti GPU. We report the average training time on three datasets. Seq2seq: 4.2 hours; MMI: 7.8 hours; BART: 7.1 hours; SpaceFusion: 16.2 hours (running on the CPUs only); GPS: 4.0 hours. We observe that our model requires lower or similar training cost, compared to the baselines.

A.4 Case Study

Table 5 presents case studies on the generated response for different models. In cases (a) and (b), both BART and our model make reasonable responses, whereas Seq2Seq and MMI produce only nonsense. In cases (c)-(e), Seq2Seq, MMI and BART generate general and safe responses while our model directly targets the bad words (e.g., "twat", "fairy gay faggot") in the hate speech, and even shows understanding and kind warnings to the hate speaker in case (c). Therefore, our model may make the hate speaker feel their voices have been heard and is considered closer to human-like moderators. Moreover, we find BART sometimes identifies wrong hate words (In case (d), the hate word is "twat" while BART refers to "troll" in the response. In case (e), the hate word is "fairy gay faggot" while BART refers to "kike".). The incorrect referral could potentially make the hate speakers irritated and become even more offensive.

A.5 Examples of Annotations

Table 6 presents human annotation guidelines and examples on the three dimensions. The inter-annotator reliability scores are 0.50, 0.46, 0.36 for diversity, relevance and language quality respectively.

Hate Speech	What is the point in listening to Muslim women? They are all retarded.
Seq2Seq	“So is thinly veiled Muslim Muslim women , the , population , perhaps we go to”
MMI	“That is influential figures of Baroness Sayeeda Warsi , Malala Yousafzai- I Harvard Fellow Hayfa Al-Kaylani ? Should I list some more brilliantly smart Muslim women ?”
BART	And when CVs submitted under a non Muslim name are 3 times more likely to get an interview than those with a Muslim name attached, the problem very clearly lies with those employers who discriminate against Muslim women who do not want to work with them and instead discriminate against those who do.
GPS	The point of listening to Muslim women, and women in general, is that they could broaden and enlighten your view of the world.

(a) Islamophobia

Hate Speech	All the Muslims should have been punished for yesterday, not only a few. Citizen will take action if the government does not do anything.
Seq2Seq	Are you suggesting that the government deports the good in the countries.
MMI	I am not think the government ’s Witness , I is not think that they are not.
BART	What happened to the good old British values of stuff like ‘tolerance’ and ‘respect for the rule of law’ - what you are suggesting sounds like mob rule, vigilante action and people running around with burning torches? If you really care about stopping crime, I think you should focus your energy on campaigning against it, not using it to support your anti-Muslim rhetoric.
GPS	You cannot blame all people for actions of a few. Banning something altogether will not solve anything.

(b) Islamophobia

Hate Speech	I WILL NOT support this one. Not interested in a SJW nigger loving virtue signaler. Will oppose her to all my friends and elected officials.
Seq2Seq	It does n’t make sense to hate an entire race or group of people based on the actions of a few people .
MMI	We does n’t make sense to hate an entire race or religion based on the actions of a few people .
BART	Whoa there, Racist views are hurtful and derogatory. Please be considerate of others when posting. Please refrain from using this type of language. Thank you and have a great day, thank you. (bloody lot of you) (n-word)
GPS	I understand you don’t care for her, but using words like that will only turn people against you, not her.

(c) Derogatory Language

Hate Speech	delusional twat
Seq2Seq	Use of the r-word is unacceptable in discourse as it demeans and insults people with mental disabilities .
MMI	We can make the same point more effectively without the use of hateful terminology .
BART	Use of the slur here is unacceptable in our discourse as it demeans and insults women, and is hurtful and offensive to the mentally disabled and their families.“Troll” is an offensive term. is an ethnic slur, please use better word choices.
GPS	Please refrain using the word twat.

(d) Derogatory Language

Hate Speech	A political homo? I am not listening to a fairy gay faggot for anyone. Thank you very much.
Seq2Seq	Stop using hateful language regarding sexual orientation.
MMI	Please refrain from using hateful words in your posts . It ! It adds nothing of value to the discussion .
BART	Use of the slurs and rank hatred here is unacceptable in our discourse as it demeans and insults and alienates others. “faggot” and “kike” are hate speech and are offensive and should be avoided. the word faggot is wrong.
GPS	Attacking someone’s sexuality (“fairy gay faggot”) to express your disagreement with someone’s politics is not acceptable in this forum.

(e) Derogatory Language

Table 5: Examples of the generated counterspeech.

Hate Speech	What is the point in listening to Muslim women? They are all retarded.
System A	“So is thinly veiled Muslim Muslim women , the , population , perhaps we go to”
System B	“That is influential figures of Baroness Sayeeda Warsi , Malala Yousafzai- I Harvard Fellow Hayfa Al-Kaylani ? Should I list some more brilliantly smart Muslim women ?”
System C	And when CVs submitted under a non Muslim name are 3 times more likely to get an interview than those with a Muslim name attached, the problem very clearly lies with those employers who discriminate against Muslim women who do not want to work with them and instead discriminate against those who do.
System D	The point of listening to Muslim women, and women in general, is that they could broaden and enlighten your view of the world.

Question 1:	Which system-generated counterspeech is most relevant to the hate speech? A relevant counterspeech should address the central aspect of the hateful message and make a coherent conversation.
Question 2:	Which system-generated counterspeech is most grammaticality correct, readable and fluent?

(a) Evaluation on relevance and language quality

System A	1. ”Using “c–ts” to refer to women is offensive, unnecessary and should be avoided.
	2. Using the foul language will make people more appropriate words.
	......
	10. Using the word “retards” is offensive to the mentally disabled. its a direct attack to their disability.
System B	1. “Ret–ds” as used here may offend the mentally disabled and their families; it should be avoided as it adds nothing of substance.
	2. Please don’t use the r-word in your posts. It doesn’t help to the discussion in this thread.
	......
	10. Please do not use derogatory language for women.
System C	1. Please refrain from using hateful and ableist language in your posts. It adds nothing to your argument or the discussion in this thread. Please refrain in the future if you would like to keep your account active. Thank you, and have a nice day!
	2. Please refrain from using hateful ableist language in your posts. It adds nothing productive to the conversation or the sub. Please refrain from it in the future if you would like to keep your account active. “Retard” is a hateful word that is used to demean people who struggle with intellectual disability.
	......
	10. Using the word “cunts” is a direct attack against a person based on their gender. Its offensive, unnecessary and should be avoided. “B–ch” and “c–t” are hateful terms used to demeans women in a hateful manner.
System D	1. Feminists are just human beings fighting for their human rights. Please refrain from using the term in a negative context.
	2. I don’t think that you should be spending so much energy defending your right to violence.
	......
	10. Right. I cannot stand this either. As a woman I’m annoyed when female characters are forced into the story.

Question 3:	Which system has the most diversified counterspeech in terms of vocabulary richness, variety in expression and inter-response diversity?

(b) Evaluation on diversity. The hate speech are not shown to the annotators.

Table 6: Examples of Annotation. We randomize the system outputs to avoid annotators’ selection preferences.

REPT: Bridging Language Models and Machine Reading Comprehension via Retrieval-Based Pre-training

Fangkai Jiao[1]*, **Yangyang Guo**[1], **Yilin Niu**[2], **Feng Ji**[3], **Feng-Lin Li**[3], **Liqiang Nie**[1]†
[1] School of Computer Science and Technology, Shandong University, Qingdao, China
[2] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3] Damo Academy, Alibaba Group, Hangzhou, China
jiaofangkai@hotmail.com guoyang.eric@gmail.com niuyl14j@gmail.com
{zhongxiu.jf, fenglin.lfl}@alibaba-inc.com nieliqiang@gmail.com

Abstract

Pre-trained Language Models (PLMs) have achieved great success on Machine Reading Comprehension (MRC) over the past few years. Although the general language representation learned from large-scale corpora does benefit MRC, the poor support in evidence extraction which requires reasoning across multiple sentences hinders PLMs from further advancing MRC. To bridge the gap between general PLMs and MRC, we present REPT, a REtrieval-based Pre-Training approach. In particular, we introduce two self-supervised tasks to strengthen evidence extraction during pre-training, which is further inherited by downstream MRC tasks through the consistent retrieval operation and model architecture. To evaluate our proposed method, we conduct extensive experiments on five MRC datasets that require collecting evidence from and reasoning across multiple sentences. Experimental results demonstrate the effectiveness of our pre-training approach. Moreover, further analysis shows that our approach is able to enhance the capacity of evidence extraction without explicit supervision.

1 Introduction

Machine Reading Comprehension (MRC) is an important task to evaluate the machine understanding of natural language. Given a set of documents and a question (with possible options), an MRC system is required to provide the correct answer by either retrieving a meaningful span (Rajpurkar et al., 2018a) or selecting the correct option from a few candidates (Lai et al., 2017; Sun et al., 2019; Guo et al., 2019, 2021). Recently, with the development of self-supervised learning, the pre-trained language models (Devlin et al., 2019; Yang et al., 2019b) fine-tuned on several machine reading comprehension benchmarks (Reddy et al., 2019; Kwiatkowski et al., 2019) have achieved superior performance. The dominant reason lies in the strong and general contextual representation learned from large-scale natural language corpora. Nevertheless, PLMs focus more on the general language representation and semantics to benefit various downstream tasks, while MRC demands the capability of extracting evidence across one or multiple documents and performing reasoning over the collected clues (Fang et al., 2020; Yang et al., 2018). Put it differently, there exists an obvious gap, indicating an insufficient exploitation of PLMs over MRC.

Some efforts have been made to bridge the gap between PLMs and downstream tasks, which can be roughly divided into two categories: knowledge enhancement and task-oriented pre-training (Qiu et al., 2020). The former introduces commonsense or world knowledge into the pre-training (Zhang et al., 2019; Sun et al., 2020; Varkel and Globerson, 2020; Ye et al., 2020) or fine-tuning (Yang et al., 2019a) for better performance over knowledge-driven tasks. And the latter includes some delicately designed pre-training tasks, e.g., the contrastive approach of learning discourse knowledge towards textual entailment task (Iter et al., 2020). Although these approaches have achieved some improvements on certain tasks, few of them are specifically designed for evidence extraction, which is indeed indispensable to MRC.

In fact, equipping PLMs with the capability of evidence extraction in MRC is challenging due to the following two factors. 1) The process of collecting clues from a document is difficult to be integrated into PLMs without designing specific model architectures or pre-training tasks (Qiu et al., 2020; Zhao et al., 2020). And 2) large-scale pre-training process would make PLMs overfit to pre-

---

[*] Work is done during internship at Alibaba Group.

[†] Corresponding author: Liqiang Nie.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 150–163
August 1–6, 2021.

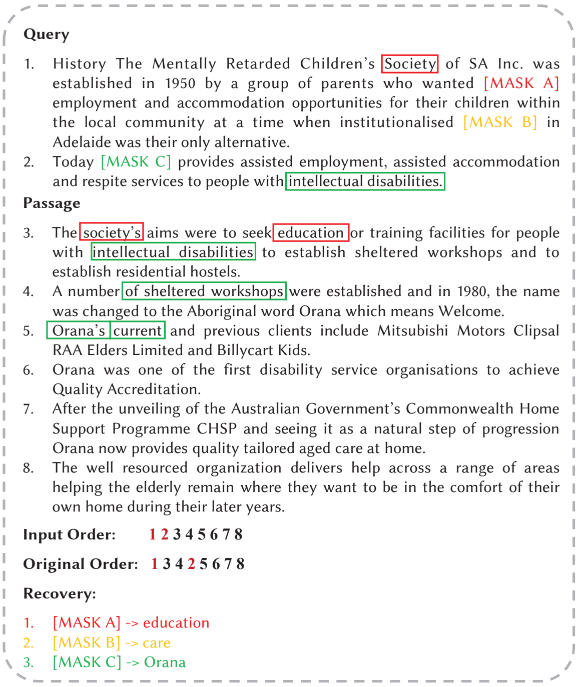

Query

1. History The Mentally Retarded Children's Society of SA Inc. was established in 1950 by a group of parents who wanted [MASK A] employment and accommodation opportunities for their children within the local community at a time when institutionalised [MASK B] in Adelaide was their only alternative.
2. Today [MASK C] provides assisted employment, assisted accommodation and respite services to people with intellectual disabilities.

Passage

3. The society's aims were to seek education or training facilities for people with intellectual disabilities to establish sheltered workshops and to establish residential hostels.
4. A number of sheltered workshops were established and in 1980, the name was changed to the Aboriginal word Orana which means Welcome.
5. Orana's current and previous clients include Mitsubishi Motors Clipsal RAA Elders Limited and Billycart Kids.
6. Orana was one of the first disability service organisations to achieve Quality Accreditation.
7. After the unveiling of the Australian Government's Commonwealth Home Support Programme CHSP and seeing it as a natural step of progression Orana now provides quality tailored aged care at home.
8. The well resourced organization delivers help across a range of areas helping the elderly remain where they want to be in the comfort of their own home during their later years.

Input Order: 1 2 3 4 5 6 7 8

Original Order: 1 3 4 2 5 6 7 8

Recovery:

1. [MASK A] -> education
2. [MASK B] -> care
3. [MASK C] -> Orana

Figure 1: A running example obtained from our method. The query sentences are extracted from the original document with some crucial information being randomly masked, i.e., the sentence 1 and 2. The model is required to predict the preceding and following sentence for each query in the original document and recover the masked clues, i.e., infer the original order from input order and fill the [MASK] with the initial token. The phrases in boxes are the possible clues for recovering the masked tokens and the correct order.

training tasks (Chung et al., 2021; Tamkin et al., 2020). In other words, it is difficult to take full advantage of the pre-training merits if the training objectives of pre-training and downstream MRC are greatly separated.

To deal with the aforementioned challenges, we propose a novel retrieval-based pre-training approach, REPT, to bridge the gap between PLMs and MRC. Firstly, to unify the training objective, we design a novel pre-training task, namely Surrounding Sentences Prediction (SSP), as illustrated in Figure 1. Given a document, several sentences will be firstly selected as queries, and the others are jointly treated as a passage[1]. Thereafter, for each query, the model should predict its preceding and following sentences in the original document by collecting clues from each sentence, which is compatible with evidence extraction in MRC tasks. It is worth emphasizing that, the repeated occurrence of entities or nouns across different sentences often lead to information short-cut (Lee et al., 2020), from which the order of sentences can be easily recovered. In view of this, we propose to mask such explicit clues. As a result, the model is enforced to infer the correct positions of queries by gathering evidence with the incomplete information. Secondly, to preserve the effectiveness of contextual representation, the masked clues are also required to be recovered through retrieving relevant information from other parts of the document, which is implemented via our Retrieval based Masked Language Modeling (RMLM) task.

In this way, the pre-training stage can be properly aligned with MRC: 1) the training objectives are connected through the introduction of the two pre-training tasks, which will be inherited by downstream MRC tasks through consistent retrieval operation. And 2) the capability of evidence extraction from documents or sentences is enhanced during pre-training, and will be smoothly transferred to MRC. Our contributions in this paper are summarized as follows:

1. We present REPT, a novel pre-training approach, to bridge the gap between PLMs and MRC through retrieval-based pre-training.

2. We design two self-supervised pre-training tasks, i.e., SSP and RMLM, to augment PLMs with the ability of evidence extraction with the help of retrieval operation and eliminating information short-cut, which can be smoothly transferred to downstream MRC tasks.

3. We evaluate our method over five reading comprehension benchmarks of two different task forms: Multiple Choice QA (MCQA) and Span Extraction (SE). The substantial improvements over strong baselines demonstrate the effectiveness of our pre-training approach. We conduct an empirical study to verify that our method are able to enhance evidence extraction as expected.

2 Related Work

MRC has received increasing attention in recent years. Many challenging benchmarks have been established to examine various forms of reasoning abilities, e.g., multi-hop (Yang et al., 2018), discrete (Dua et al., 2019), and logic reasoning (Yu et al., 2020). To solve the problem, a typical design is to gather possible clues through entity linking

---

[1] We use *passage* here to keep consistent with MRC tasks. And *document* refers to the combination of queries and *passage*.

(Zhao et al., 2020) or self-constructed graph (Fang et al., 2020; Ran et al., 2019), and then perform multi-step reasoning. It is worth noting that, gathering clues is vital but challenging, especially for long document understanding. Some efforts have been dedicated to improving evidence extraction via direct (Wang et al., 2018) or distant supervision (Niu et al., 2020).

Generally, the fine-tuned PLMs (Devlin et al., 2019; Yang et al., 2019b) can obtain superior performance in MRC due to their strong and general language representation. However, there still exist some gaps between PLMs and various downstream tasks, since certain abilities required by the downstream tasks cannot be learned through the existing pre-training tasks (Qiu et al., 2020). In order to take full advantage of PLMs, a few studies attempt to align the pre-training and fine-tuning stages. For example, Tamborrino et al. (2020) reformulated the commonsense question answering task as scoring via leveraging the predicted probabilities for Masked Language Modeling (MLM) in RoBERTa (Liu et al., 2019). With the help of the commonsense learned through MLM, the method achieves comparable results with supervised approaches in zero-shot setting, indicating that bridging the gap between these two stages yields considerable improvement. Chung et al. (2021) tried to address the overfitting problem during pre-training through decoupling input and output embedding weights and enlarging the embedding size during decoding. The resultant model is therefore more transferable across tasks and languages.

In addition, some task-oriented pre-training methods have also been developed. For instance, Wang et al. (2020) proposed a novel pre-training method for sentence representation learning, where the masked tokens in a sentence are forced to be recovered from other sentences through sentence-level attention. Based on this, the attention weights can be directly fine-tuned to rank the candidates in answer selection or information retrieval. Lee et al. (2019) tried to learn the dense document representation for information retrieval by minimizing the distance between the representation of an query sentence and its context. Guu et al. (2020) designed an augmented MLM tasks to jointly train a neural retriever and a language model for Open-domain QA. Different from these methods ranking the documents for open-domain QA, our approach focuses on enhancing the ability of evidence extraction in MRC, where the MLM based task by it alone is insufficient.

3 Method

In this section, we present the details of the proposed method, REPT. We firstly describe the data pre-processing part (§3.1), and then illustrate the two pre-training tasks, i.e., SSP and RMLM (§3.3) and the training objectives (§3.4). Finally, we detail how to fine-tune our pre-trained model for downstream tasks through retrieval-based evidence extraction (§3.5).

3.1 Data Pre-processing

For pre-training, we use the English Wikipedia[2] as our training data. We divide each Wikipedia article into segments, each containing up to 500 tokens[3] without overlapping. We treat each segment as a document and split it into several sentences[4].

In order to increase the difficulty and efficiency of pre-training, for each document, we select 30% of the most important sentences as queries and the rest in their original order as a passage. Specifically, the importance of each sentence in a document is measured through the summation of the importance of entities and nouns it contains, which is further defined as the number of sentences an entity/noun occurs. Hereafter, masking is introduced to entities and nouns in queries according to pre-defined ratios to eliminate information short-cut. More details about the masking strategy are described in Appendix A and an example after pre-processing can be found in Figure 1.

3.2 Task Definition

We treat a document as a sequence of n sequential sentences with m tokens. Supposing that there are t sentences selected as queries following §3.1, the rearranged sequence is defined as $\mathcal{S} = [\mathbf{s}^1, \mathbf{s}^2, \cdots, \mathbf{s}^t, \cdots, \mathbf{s}^n]$, and the index of queries is $\mathcal{Q} = \{1, 2, \cdots, t\}$. Besides, we define a mapping function r to map the rearranged sentences to their original position. Taking Figure 1 as an example, the mapping $r(\mathbf{s}^1) = 1,\ r(\mathbf{s}^2) = 4,\ r(\mathbf{s}^3) = 2$ and $(\mathbf{s}^4) = 3$ indicates that the original order is $\{\mathbf{s}^1, \mathbf{s}^3, \mathbf{s}^4, \mathbf{s}^2, \cdots\}$.

Taking $\mathcal{S}$ as input, the Surrounding Sentences Prediction task should predict the correct sentence

---

[2] We use the 2020/05/01 dump.

[3] The tokenized sub-words following BERT and RoBERTa.

[4] Any sentence with less than five tokens is concatenated to its previous one.

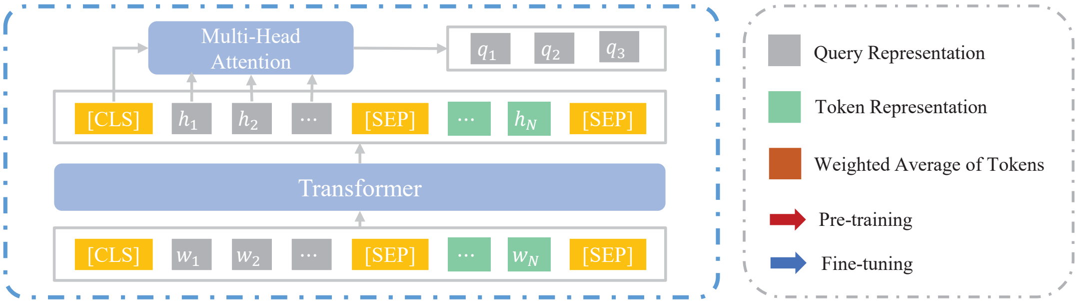

a) Encoder

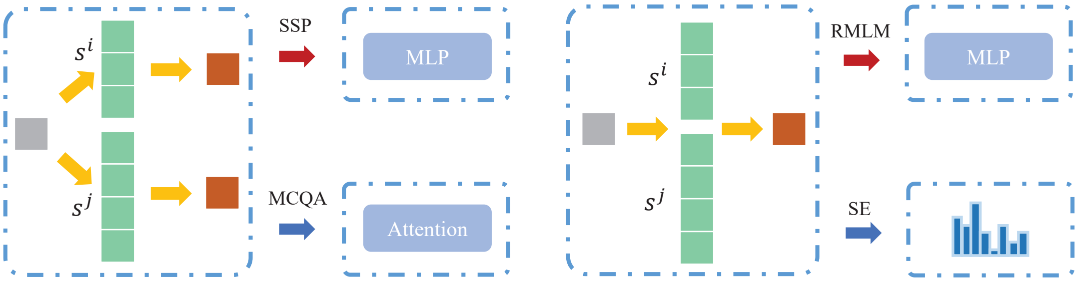

b) Sentence-Level Retrieval for SSP and MCQA

c) Document-Level Retrieval for RMLM, ODQA and Span Extraction (SE)

Figure 2: Framework of our model. a) Encoder composed of a pre-trained Transformer encoder and a query generator based on multi-head attention. b) The attention-based sentence-level retrieval for evidence extraction for each sentence, which will be further adopted by SSP during pre-training and MCQA during fine-tuning. c) The attention-based document-level retrieval for evidence extraction among the input sequence, which is employed for RMLM. For SE, the similarity function is directly fine-tuned.

index a and b for each query $\mathbf{s}^q$ with $q \in \mathcal{Q}$[5]:

$$\begin{cases} r(\mathbf{s}^a) = r(\mathbf{s}^q) - 1, \\ r(\mathbf{s}^b) = r(\mathbf{s}^q) + 1. \end{cases} \tag{1}$$

As for the Retrieval based Masked Language Modeling (RMLM) task, the model should recover all the masked tokens in each query $\mathbf{s}^q$.

3.3 Model

First of all, we leverage a pre-trained Transformer (Vaswani et al., 2017), such as BERT, as our encoder to obtain the contextual representation of sentences. The output of Transformer is formulated as:

$$\mathbf{H} = [\mathbf{h}_{\text{cls}}, \cdots, \mathbf{h}_m, \mathbf{h}_{\text{sep}}] = \text{Encoder}(\tilde{\mathcal{S}}), \tag{2}$$

where $\mathbf{H} \in \mathbb{R}^{d \times (m+3)}$, and d is the hidden size. For a better illustration, we will use $\mathbf{H}^i$ to represent the hidden state matrix of tokens that belong to sentence $\mathbf{s}^i$, such that:

$$\mathbf{H} = [\mathbf{H}^1, \mathbf{H}^2, \cdots, \mathbf{H}^n], \ \mathbf{H}^i \in \mathbb{R}^{d \times l_i},$$

where l_i is the length of sentence $\mathbf{s}^i$ and $m = \sum_i l_i$. Since the process for each query is exactly the same, we use $q \in \mathcal{Q}$ as a representative to introduce the calculation with respect to each query below.

3.3.1 Query Representation

In order to gather potential clues from a document or sentences, we adopt the multi-head attention mechanism proposed by (Vaswani et al., 2017) to obtain the sentence-level representation for each query. Formally, the attention mechanism is defined as $\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are query, key and value matrices, respectively. To consider the global information, we leverage $\mathbf{h}_{\text{cls}}$ as the query vector, and $\mathbf{H}^q$ as $\mathbf{K}$ and $\mathbf{V}$:

$$\mathbf{v}_0^{q\top} = \text{MHA}(\mathbf{h}_{\text{cls}}^\top, \mathbf{H}^q, \mathbf{H}^q). \tag{3}$$

During pre-training, we reuse the layer defined by Equation 3 with $\mathbf{Q} = \mathbf{v}_0^q$ and $\mathbf{K} = \mathbf{V} = \mathbf{H}^q$, to generate the task-specific query representation $\mathbf{v}^q$, which is designed to alleviate the overfitting problem (He et al., 2021).

3.3.2 Surrounding Sentence Prediction

To enhance the capability of pre-trained models for evidence extraction, we have carefully designed the SSP task, where the model should predict the preceding and following sentences for a given query by extracting the relevant evidence from each sentence. Consequently, we introduce a retrieval operation, which is implemented via a single-head attention

---

[5] Specifically, for $r(\mathbf{s}^q) = 1$ or $r(\mathbf{s}^q) = n$, the corresponding prediction task is removed since its preceding or following sentence does not exist.

mechanism[6]:

$$\mathbf{u}_q^{i\top} = \mathrm{Att}(\mathbf{v}^{q\top}, \mathbf{H}^i, \mathbf{H}^i), \tag{4}$$

where $\mathbf{u}_q^i$ is the representation of sentence s^i, highlighting the evidence information pertaining to query s^q. Finally, the score of each sentence in the document with regard to s^q is obtained through:

$$\mathbf{o}_q^i = \mathbf{W}_2(\tanh(\mathbf{W}_1\mathbf{u}_q^i + \mathbf{b}_1)) + \mathbf{b}_2. \tag{5}$$

3.3.3 Retrieval based MLM

Since the masking noise introduced when constructing queries could also bring inconsistency between pre-training and fine-tuning, we further designed a retrieval based MLM task to alleviate this problem. In the RMLM task, the model should predict the masked entities or nouns through retrieving relevant information from a document. More specifically, the query-aware evidence representation of the input sequence is obtained via:

$$\mathbf{g}^{q\top} = \mathrm{Att}(\mathbf{v}^{q\top}, \mathbf{H}, \mathbf{H}). \tag{6}$$

Denoting the index of a masked token in query s^q as z, the representation of the masked token s_z^q used for recovering is:

$$\tilde{\mathbf{h}}_z^q = f(\mathbf{h}_z, \mathbf{g}^q), \tag{7}$$

where the function $f(\cdot,\cdot)$ is implemented as a normalized 2-layer feed-forward network, and the details are illustrated in Appendix B.2.

3.4 Optimization

As the definition in Equation 1, given a and b as the index of the original preceding and following sentences of the query s^q in $\mathcal{S}$, the corresponding probabilities for surrounding sentences are formulated as:

$$\begin{cases} p_{\mathrm{ssp}}(a|q,\mathcal{S}) = \dfrac{\exp(\mathbf{o}_q^a)}{\sum_{j=1, j\notin\{b,q\}}^{n} \exp(\mathbf{o}_q^j)}, \\ p_{\mathrm{ssp}}(b|q,\mathcal{S}) = \dfrac{\exp(\mathbf{o}_q^b)}{\sum_{j=1, j\notin\{a,q\}}^{n} \exp(\mathbf{o}_q^j)}. \end{cases} \tag{8}$$

The objective of SSP is subsequently defined as:

$$\mathcal{L}_{\mathrm{ssp}} = \mathbb{E}(-\frac{1}{|\mathcal{Q}|}\sum_q \big(\log p_{\mathrm{ssp}}(a|q,\mathcal{S}) + \log p_{\mathrm{ssp}}(b|q,\mathcal{S})\big)). \tag{9}$$

As for RMLM, supposing the index set of masked tokens in query $\mathbf{s}^q$ is $\mathcal{Z}^q$, and the set of corresponding original tokens is $\mathcal{X}^q$, the probability for recovering a masked token is:

$$p_{\mathrm{rmlm}}(x_z|z,q,\mathcal{S}) = \frac{\exp(\mathrm{e}(x_z)^\top \tilde{\mathbf{h}}_z^q)}{\sum_{x'} \exp(\mathrm{e}(x')^\top \tilde{\mathbf{h}}_z^q)}, \tag{10}$$

where $z \in \mathcal{Z}^q, x_z \in \mathcal{X}^q$, x' is a token in vocabulary, and $\mathrm{e}(x)$ denotes the word embedding of x. Then the objective of RMLM is:

$$\mathcal{L}_{\mathrm{rmlm}} = \mathbb{E}(-\frac{\sum_q \sum_z \log p_{\mathrm{pmlm}}(x_z|z,q,\mathcal{S})}{\sum_q |\mathcal{Z}^q|}). \tag{11}$$

During pre-training, the model tries to optimize the two objectives jointly:

$$\mathcal{L} = \mathcal{L}_{\mathrm{ssp}} + \mathcal{L}_{\mathrm{rmlm}}. \tag{12}$$

3.5 Fine-tuning

During fine-tuning, the input contains a query sentence and a passage. For multiple choice QA tasks, we concatenate a question with an option to form a question-option pair and use it as a whole query. In this section, we use $q = 0$ to represent the index of the query and the sentences of passage are kept in their original order. The input sequence can be thus denoted as:

$$\mathcal{S} = [s^q, s^1, s^2, \cdots, s^n].$$

To inherit the evidence extraction ability augmented during pre-training, we incorporate the same retrieval operation into fine-tuning to collect clues from the passage. Firstly, we reuse the attention mechanism defined in Equation 3 to obtain the query representation $\mathbf{v}^q$. As for the evidence extraction process, we formulate it differently for Multiple Choice QA and Span Extraction.

3.5.1 Multiple Choice QA

Similar to Equation 4, we adopt an attention mechanism, whereby the query-aware sentence representation $\mathbf{u}_q^i$ is obtained via gathering evidence from each sentence:

$$\mathbf{u}_q^{i\top} = \mathrm{Att}(\mathbf{v}^{q\top}, \mathbf{H}^i, \mathbf{H}^i),\ i \neq q. \tag{13}$$

And the final passage representation highlighting the evidence can be obtained via the sentence-level evidence extraction:

$$\mathbf{v}^p = \mathrm{Att}(\mathbf{v}^{q\top}, \mathbf{U}, \mathbf{U}), \tag{14}$$

---

[6] The details are illustrated in Appendix B.1.

where $\mathbf{U} = [\mathbf{u}_q^1, \cdots, \mathbf{u}_q^n]$ and $\mathbf{U} \in \mathbb{R}^{d \times n}$. Finally, we represent the probability of each option c using both the query $\mathbf{v}^q$ and the passage $\mathbf{v}^p$:

$$p_c^{\text{mc}} \propto \exp(\mathbf{W}_6(\tanh(\mathbf{W}_5[\mathbf{v}^q; \mathbf{v}^p] + \mathbf{b}_5)) + \mathbf{b}_6). \tag{15}$$

Specifically, for Multi-RC, since the number of correct answer options for each question is uncertain, the task is often treated as a binary classification problem for each option. As a result, we adopt a MLP to get the probability of whether an option c is correct:

$$p_c^{\text{mc}} = \sigma(\mathbf{W}_8(\tanh(\mathbf{W}_7[\mathbf{v}^q; \mathbf{v}^p] + \mathbf{b}_7)) + \mathbf{b}_8), \tag{16}$$

where σ is the sigmoid function.

3.5.2 Span Extraction

Since answer spans are often consistent with corresponding evidences, we directly leverage the query to extract relevant spans. The probability of selecting start position s and end position e of an answer span is given by:

$$\begin{cases} p_s^{\text{span}} \propto \exp(\mathbf{v}^{q\top} \mathbf{W}_9 \mathbf{h}_s), \\ p_e^{\text{span}} \propto \exp(\mathbf{v}^{q\top} \mathbf{W}_{10} \mathbf{h}_e). \end{cases} \tag{17}$$

4 Experiment

4.1 Dataset

4.1.1 Multiple Choice Question Answering

DREAM (Sun et al., 2019) contains 10,197 multiple choice questions for 6,444 dialogues collected from English Examinations designed by human experts, in which 85% of the questions require reasoning across multiple sentences, and 34% of the questions also involve commonsense knowledge.

RACE (Lai et al., 2017) is a large-scale reading comprehension dataset collected from English Examinations and created by domain experts to test students' reading comprehension skills. It has a wide variety of question types, e.g., summarization, inference, deduction and context matching, and requires complex reasoning techniques.

Multi-RC (Khashabi et al., 2018) is a dataset of short paragraphs and multi-sentence questions. The number of correct answer options for each question is not pre-specified and the correct answer(s) is not required to be a span in the text. Moreover, the dataset provides annotated evidence sentence.

ReClor (Yu et al., 2020) is extracted from logical reasoning questions of standardized graduate admission examinations. Existing studies show that the state-of-the-art models perform poorly on ReClor, indicating the deficiency of logical reasoning ability of current PLMs.

4.1.2 Span Extraction

Hotpot QA (Yang et al., 2018) is a question answering dataset involving natural and multi-hop questions. The challenge contains two settings, the distractor setting and the full-wiki setting. In this paper, we focused on the full-wiki setting, where the system should retrieve the relevant paragraphs from Wikipedia and then predict the answer.

SQuAD2.0 (Rajpurkar et al., 2018b) is reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.

4.2 Implementation Detail

We leave the details about the implementation and pre-training corpora in Appendix A due to the limitation of space.

4.3 Baseline

Since our method is used for further pre-training, we mainly compared our model with BERT/RoBERTa and their variants. For Hotpot QA, we integrated our models into an open-sourced and well-accepted system (Asai et al., 2020) and evaluated the performance. The details of baselines are summarized as follows:

4.3.1 Multiple Choice QA

BERT is the BERT-base model with 2-layer MLP as the task-specific module.

BERT-Q & **RoBERTa-Q** refer to the designed but not further trained models, which include an extra multi-head attention for generating query representation via Equation 3, and our retrieval operation for evidence extraction as in §3.5.1 and §3.5.2.

BERT-Q w. R/S & **RoBERTa-Q w. R/S** refer to the designed models further trained with our proposed SSP and RMLM tasks (denoted as **S** and **R**, respectively).

BERT-Q w. R & **BERT-Q w. S** refer to the models further trained with only one pre-training task, RMLM or SSP.

BERT-Q w. M & **BERT w. M** refer to the models further trained with MLM. For fair comparison, we

Model / Dataset	RACE Dev Acc.	 Test Acc.	DREAM Dev Acc.	 Test Acc.	ReClor Dev Acc.	 Test Acc.	Multi-RC Dev EM	 $\mathrm{F1}_a$	 $\mathrm{F1}_m$
BERT-base†	–	65.0	63.4	63.2	**54.6**	47.3	–	–	–
BERT w. M	67.7	66.3	62.9	63.2	51.6	45.1	26.6	71.8	74.2
BERT-Q	67.2	65.2	62.9	62.3	48.4	45.0	22.8	69.6	72.0
BERT-Q w. M	67.7	66.9	61.8	62.2	48.8	48.3	23.8	70.1	72.6
BERT-Q w. R	65.5	64.7	59.0	58.6	46.8	45.1	26.4	71.5	74.0
BERT-Q w. S	69.5	66.5	**64.8**	62.2	52.0	46.5	30.0	73.0	75.8
BERT-Q w. R/S	**70.1**	**68.1**	64.4	**64.0**	50.6	**49.2**	**31.9**	**73.8**	**76.3**
RoBERTa-base	76.0	75.5	**71.2**	69.8	54.8	**50.8**	38.7	77.1	79.2
RoBERTa-Q	76.8	**75.7**	70.9	69.5	**56.0**	49.7	34.6	75.4	77.4
RoBERTa-Q w. R/S	**77.1**	74.9	70.9	**70.8**	54.8	50.3	**40.4**	**77.6**	**80.0**

Table 1: Results on multiple choice question answering tasks. ($\mathrm{F1}_a$: F1 score on all answer-options; $\mathrm{F1}_m$: macro-average F1 score of all questions.) We ran all experiments using **four** different random seeds with the same hyper-parameters, and report the average performance, except for ReClor and Multi-RC. For ReClor, we submitted the best model on the development set to the leaderboard to get the results on the test set. For MultiRC, we merely reported the performance on development set since the test set is unavailable. †: The results are reported by the leaderboard.

further train BERT with the same Wikipedia corpus for equivalent steps.

4.3.2 Hotpot QA

For hotpot QA, we constructed the system based on Graph-based Recurrent Retriever (Asai et al., 2020), which includes a retriever and a reader based on BERT. We simply replaced the reader with our models and evaluated their performance in comparison with several published strong baselines on the leaderboard[7].

5 Results and Analyses

5.1 Results for Multiple Choice QA

Table 1 shows the results of the baselines and our method on multiple choice question answering.

From Table 1, we can observe that: 1) Compared with BERT-Q and BERT, our method significantly improves the performance over all the datasets, which validates the effectiveness of our proposed pre-training method. 2) As for the model structure, BERT-Q obtains similar or worse results compared with BERT, which suggests that the retrieval operation can hardly improve the performance without specialised pre-training. 3) Taking the rows of BERT, BERT-Q, BERT w. M, BERT-Q w. M for comparison, the models with further pre-training using MLM achieve similar or slightly higher performance. The results show that further training BERT using MLM and the same corpus can only achieve very limited improvements. 4) Regarding the two pre-training tasks, BERT-Q w. R/S leads to similar performance on the development sets compared with BERT-Q w. S, but a much higher accuracy on the test sets, which suggests RMLM can help to maintain the effectiveness of contextual language representation. However, there is a significant degradation over all datasets for BERT-Q w. R. The main reason is possibly because the model cannot tolerate the sentence shuffling noise, which may lead to the discrepancy between pre-training and MRC, and thus need to be alleviated through SSP. And 5) considering the experiments over RoBERTa-based models, RoBERTa-Q w. R/S outperforms RoBERTa-Q and RoBERTa-base with considerable improvements over Multi-RC and the test set of DREAM, which also indicates that our method can benefit stronger PLMs.

5.2 Performance on Span Extraction QA

The results of span extraction on Hotpot QA are shown in Table 2. We constructed the system using the Graph Recurrent Retriever (GRR) proposed by Asai et al. (2020) and different readers. As shown in the table, GRR + BERT-Q w. R/S outpeforms GRR + BERT-base by more than 2.5% absolute points on both EM and F1. And GRR + RoBERTa-Q w. R/S also achieves a significant improvement over GRR + RoBERTa-base. During the test stage, our best system, GRR + RoBERTa-Q w. R/S performs better than the strong baselines and get closer to GRR + BERT-wwm-large. The above results strongly demonstrate the effectiveness of our pre-

---

[7] https://hotpotqa.github.io/.

Model / Dataset	Dev		Test	
	EM	F1	EM	F1
Transformer-XH (Zhao et al., 2020)	54.0	66.2	51.6	64.7
HGN (Fang et al., 2020)	–	–	56.7	69.2
GRR + BERT-wwm-Large*	**60.5**	**73.3**	**60.0**	**73.0**
GRR + BERT-base*	52.7	65.8	–	–
GRR + BERT-Q w. R/S	**55.2**	**68.4**	–	–
GRR + RoBERTa-base	56.8	69.6	–	–
GRR + RoBERTa-Q w. R/S	**58.4**	**71.3**	58.1	71.0

Table 2: Results of our method and other strong baselines on Hotpot QA. *GRR* means the Graph Recurrent Retriever proposed by Asai et al. (2020), *GRR + BERT-base* means the system whose retriever is GRR and reader is built on BERT-base. *: The results are reported by Asai et al. (2020).

Model / Dataset	EM	F1
BERT-Q	71.7	74.9
BERT-Q w. R/S	**77.2**	**80.4**
RoBERTa-Q	80.3	83.7
RoBERTa-Q w. R/S	**81.7**	**85.0**

Table 3: Results of our method and other baselines on the dev set of SQuAD2.0.

training method on the task requiring multi-hop evidence extraction and reasoning.

Besides, we also conducted experiments on the most common benchmark, SQuAD2.0. The results on development set shown in Table 3 have also verified the effectiveness of our proposed pre-training method.

5.3 Evaluation of Evidence Extraction

To evaluate the performance of our method for evidence extraction in the setting of implicit supervision (with only answers), we ranked sentences in a passage using their attention weights obtained in Equation 4 and chose those sentences with higher weights as the evidences.

As shown in Table 4, the models with our proposed pre-training tasks obtain considerable improvements on the precision and recall of evidence extraction, which verifies that our pre-training method is able to effectively equip PLMs with the capability for gathering evidence without explicit supervision. For a better illustration, we further provided two examples in Appendix C.

5.4 Effect of Different Masking Ratio During Pre-training

Table 5 shows the results of our model pre-trained with different masking ratios. Due to the small amount of entities contained in the document, we only consisdered the masking ratio of nouns as the variable. Formally, we considered three ratios: 30%, 60%, 90%, and an extra setting, where the entities and nouns are all kept and the RMLM task is also removed during pre-training.

Model	**P@1**	**R@1**	**P@2**	**R@2**
BERT-Q	21.83	9.66	20.24	17.73
BERT-Q w. R/S	**45.30**	**20.38**	**38.51**	**34.55**
RoBERTa-Q	28.25	12.45	26.93	23.74
RoBERTa-Q w. R/S	**35.34**	**15.76**	**30.33**	**26.85**

Table 4: Results of evidence extraction on the development set of Multi-RC.

Model/Dataset	RACE		Multi-RC		
	Dev	Test	Dev		
	Acc.	Acc.	EM	$\mathrm{F1}_a$	$\mathrm{F1}_m$
B.Q w.R/S (30%)	70.1	68.1	31.9	**73.8**	**76.3**
B.Q w.R/S (60%)	70.2	67.3	**32.0**	**73.8**	**76.3**
B.Q w.R/S (90%)	**70.4**	**68.2**	31.0	73.5	76.2
B.Q w.S (No Mask)	69.0	67.2	29.0	72.7	75.4

Table 5: Results on RACE and Multi-RC using models pre-trained with different mask ratios. *B.Q* means *BERT-Q*.

As shown in the table, with more possible clues being masked, the model tend to obtain better results on the downstream tasks. For example, BERT-Q w. R/S (90%) achieves the best accuracy on RACE, and BERT-Q w. R/S (60%) obtains the highest performance over Multi-RC. And all models that employ masking outperform BERT-Q w. S (no masking). The main reason can be that with more explicit information short-cut being eliminated, it is more difficult for models to collect potential clues, and PLMs are enhanced with stronger reasoning ability of evidence extraction. However, there also exists a trade-off: as higher masking ratio leads to more noise, it could worsen the mismatch between pre-training and fine-tuning, and cause performance degradation, e.g., BERT-Q w. R/S (90%) performs the worst on Multi-RC.

5.5 Performance in Low Resource Scenario

Figure 3 depicts the performance of BERT-Q w. R/S on the development and test set of RACE with limited training set. For each specific relative ratio, four reduced training sets are automatically generated using different random seeds and the corresponding accuracies are plotted on the figure. It is observed that with 70% training data, our model outperforms the baseline, BERT-Q, which was initialized using BERT and has not been further pre-

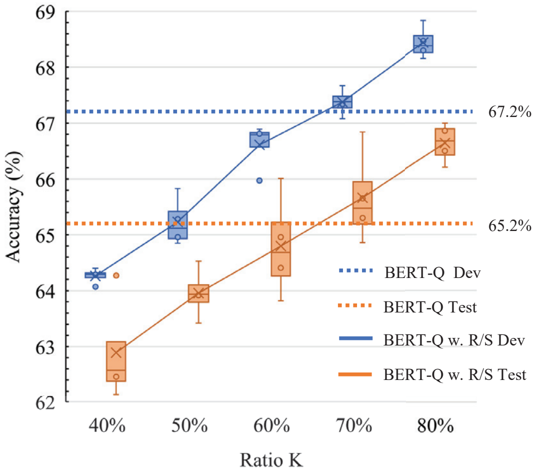

Figure 3: The accuracy of BERT-Q w. R/S on the development and test of RACE. The horizontal axis refers to the ratio K of training data compared to the original training set.

trained. The results indicate that our method can help to reduce the amount of annotated training data for downstream MRC tasks, which is especially useful in low resource scenarios.

6 Conclusion and Future Work

In this paper, we present a novel pre-training approach, REPT, to bridge the gap between pre-trained language models and machine reading comprehension through retrieval-based pre-training. Specifically, we design two retrieval-based pre-training tasks equipped with self-supervised learning, namely Surrounding Sentences Prediction (SSP) and Retreval based Masked Language Modeling (RMLM), to enhance PLMs with the capability of evidence extraction for MRC. The experiments over five different datasets validate the effectiveness of our proposed method. In the future, we plan to extend the proposed pre-training approach to the more challenging open-domain settings.

7 Acknowledgements

This work is supported by the National Key Research and Development Project of New Generation Artificial Intelligence, No.:2018AAA0102502, and the Alibaba Research Intern Program of Alibaba Group.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking embedding coupling in pre-trained language models. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186. ACL.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*, pages 2368–2378. ACL.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *EMNLP*, pages 8823–8838. ACL.

Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yibing Liu, Yinglong Wang, and Mohan Kankanhalli. 2019. Quantifying and alleviating the language prior problem in visual question answering. In *SIGIR*, pages 75–84.

Yangyang Guo, Liqiang Nie, Zhiyong Cheng, Feng Ji, Ji Zhang, and Alberto Del Bimbo. 2021. Adavqa: Overcoming language priors with adapted margin cosine loss. In *IJCAI*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: Decoding-enhanced bert with disentangled attention. In *ICLR*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. 2020. Pretraining with contrastive sentence objectives improves discourse performance of language models. In *ACL*, pages 4859–4870.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *TACL*, 8:64–77.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking Beyond the Surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*, pages 252–262. ACL.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a benchmark for question answering research. *TACL*, 7:452–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *EMNLP*, pages 785–794. ACL.

Haejun Lee, Drew A. Hudson, Kangwook Lee, and Christopher D. Manning. 2020. SLM: learning a discourse language representation with sentence unshuffling. In *EMNLP*, pages 1551–1562. ACL.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*, pages 6086–6096.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Yilin Niu, Fangkai Jiao, Mantong Zhou, Ting Yao, Jingfang Xu, and Minlie Huang. 2020. A self-training method for machine reading comprehension with soft evidence extraction. In *ACL*, pages 3916–3927.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018a. Know what you don't know: Unanswerable questions for squad. In *ACL*, pages 784–789.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018b. Know what you don't know: Unanswerable questions for squad. In *ACL*, pages 784–789.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. Numnet: Machine reading comprehension with numerical reasoning. In *EMNLP-IJCNLP*, pages 2474–2484. ACL.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *TACL*, 7:249–266.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *TACL*, 7:217–231.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Alexandre Tamborrino, Nicola Pellicanò, Baptiste Pannier, Pascal Voitot, and Louise Naudin. 2020. Pre-training is (almost) all you need: An application to commonsense reasoning. In *ACL*, pages 3878–3887. ACL.

Alex Tamkin, Trisha Singh, Davide Giovanardi, and Noah D. Goodman. 2020. Investigating transferability in pretrained language models. In *EMNLP: Findings*, pages 1393–1401. ACL.

Yuval Varkel and Amir Globerson. 2020. Pre-training mention representations in coreference models. In *EMNLP*, pages 8534–8540. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Shuohang Wang, Yuwei Fang, Siqi Sun, Zhe Gan, Yu Cheng, Jingjing Liu, and Jing Jiang. 2020. Cross-thought for sentence encoder pre-training. In *EMNLP*, pages 412–421. ACL.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced ranker-reader for open-domain question answering. In *AAAI*, pages 5981–5988.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019a. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*, pages 2346–2357. ACL.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380. ACL.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential reasoning learning for language representation. In *EMNLP*, pages 7170–7186. ACL.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. ReClor: A reading comprehension dataset requiring logical reasoning. In *ICLR*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451. ACL.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul N. Bennett, and Saurabh Tiwary. 2020. Transformer-XH: Multi-evidence reasoning with extra hop attention. In *ICLR*.

A Implementation Detail

We built our model on Huggingface's Pytorch transformer repository (Wolf et al., 2019), and used AdamW (Loshchilov and Hutter, 2019) as the optimizer. We used the pre-trained BERT-base-uncased and RoBERTa-base checkpoint to initialize our encoder, and performed pre-training using 16 P100 GPUs simultaneously. The pre-training processes last around 16 hours for BERT and 4 days for RoBERTa, which takes 20,000 steps and 80,000 steps with the batch size as 512, respectively. All hyper-parameters can be found in Table 6 for pre-training and Table 7 for fine-tuning.

During constructing the training sample for pre-training, we controlled the masking ratio for entity and noun in query. For BERT, we masked 90% entities and 30% nouns. For RoBERTa, we constructed two datasets, where the masking ratios for entity and noun are set to 90%, 30% and 90%, 90%, respectively. And we mixed the two for jointly training. We also explored the effect of different masking ratios and the analysis is detailed in §5.

As for the fine-tuning stage, for multiple choice QA, we ran all experiments using for different random seeds (i.e., 33, 42, 57 and 67) and reported the average performance, except for ReClor, in which we only submitted the results obtained from the model which performs the best on development set to the leaderboard because the limitation of submission times. For Hotpot QA, we mainly followed the hyper-parameters of Asai et al. (2020) and thus did not repeat the experiments using different random seeds. Due to the submission limitation, we only submitted our best model on the development set to the leaderboard and reported its performance on test set.

B The Details About Modeling

B.1 Single-head Attention

To reduce the extra parameters introduced, we define a single-head attention mechanism compared to the multi-head one. Given the query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$ and value matrix $\mathbf{V}$, the simple attention mechanism is formualted as:

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}((\mathbf{Q}\mathbf{W} + \mathbf{b})^\top \mathbf{K})\mathbf{V},$$

where $\mathbf{W}$ and $\mathbf{b}$ is the learnable parameters.

B.2 Normalized Feed-forward Network

We adopt a 2-layer feed-forward network with GeLU activation (Hendrycks and Gimpel, 2016) and layer normalization (Ba et al., 2016) to predict the masked entities and nouns. Following SpanBERT (Joshi et al., 2020), the Equation 7 is decomposed as:

$$\begin{cases} \mathbf{h}_0 = [\mathbf{h}_z; \mathbf{g}^q], \\ \mathbf{h}_1 = \mathrm{LayerNorm}(\mathrm{GeLU}(\mathbf{W}_3\mathbf{h}_0 + \mathbf{b}_3)), \\ \tilde{\mathbf{h}}_z^q = \mathrm{LayerNorm}(\mathrm{GeLU}(\mathbf{W}_4\mathbf{h}_1 + \mathbf{b}_4)). \end{cases}$$

C Case Study About Evidence Extraction

In §5.3, the results show that our pre-training method can augment the ability to extract the correct evidence. To give an intuitive clarification over this, we select two cases shown in Figure 4. As we can see, BERT-Q w. R/S and RoBERTa-Q w. R/S can select the correct evidence sentences, while the baselines models attend to the wrong sentences. Besides, Figure 5 shows the attention maps of the two groups of comparison. It can be observed that our pre-training approach can help the model learn a uniform attention distribution over the possible evidence sentences.

D Analysis of Extra Parameters Introduced

For fair comparison, we try to introduce as few additional parameters as possible. Since the output layer is highly task-specific and the single head-attention defined in Appendix B.1 is simple, we main analyze the extra parameters introduced for query representation learning defined in §3.3.1. A single layer of Transformer comprises of a multi-head attention module and a feed-forward network. As a result, the multi-head attention module generating the query representation has introduced 2.8% extra parameters compared with a 12-layer Transformer without consideration to the parameters in embedding layer and layer normalization.

HyperParam	**BERT-base**	**RoBERTa-base**
Peak Learning Rate	2e-4	5e-5
Learning Rate Decay	Linear	Linear
Batch Size	512	512
Max Steps	20,000	80,000
Warmup Steps	2,000	4,000
Weight Decay	0.01	0.01
Gradient Clipping	1.0	0.0
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.999	0.98
Max Sequence Length	512	512
Query Generator Dropout	0.1	0.1
SSP Dropout	0.1	0.1
RMLM Dropout	0.1	0.1
FP16 option level	O2	O2

Table 6: Hyper-parameters for pre-training.

HyperParam	**RACE**	**DREAM**	**ReClor**	**MultiRC**	**Hotpot QA**
Peak Learning Rate	4e-5$^{\clubsuit}$/2e-5$^{\spadesuit}$	3e-5$^{\clubsuit}$/2e-5$^{\spadesuit}$	2e-5$^{\clubsuit}$/1e-5$^{\spadesuit}$	3e-5	5e-5$^{\clubsuit}$/3e-5$^{\spadesuit}$
Learning Rate Decay	Linear	Linear	Linear	Linear	Linear
Batch Size	32$^{\clubsuit}$/16$^{\spadesuit}$	24	24	32	32$^{\clubsuit}$/48$^{\spadesuit}$
Epoch	4	8	10	8.0	3$^{\clubsuit}$/4$^{\spadesuit}$
Warmup Proportion	0.1$^{\clubsuit}$/0.06$^{\spadesuit}$	0.1	0.1	0.1	0.1
Weight Decay	0.01	0.01	0.01	0.01	0.01
Adam ϵ	1e-6	1e-6	1e-6	1e-6	1e-6$^{\clubsuit}$/1e-8$^{\spadesuit}$
Adam β_1	0.9	0.9	0.9	0.9	0.9
Adam β_2	0.999$^{\clubsuit}$/0.98$^{\spadesuit}$	0.999$^{\clubsuit}$/0.98$^{\spadesuit}$	0.999$^{\clubsuit}$/0.98$^{\spadesuit}$	0.999	0.999
Gradient Clipping	1.0$^{\clubsuit}$/0.0$^{\spadesuit}$	0.0$^{\clubsuit}$/5.0$^{\spadesuit}$	0.0	1.0	0.0
Max Sequence Length	512	512	256	512	384$^{\clubsuit}$/386$^{\spadesuit}$
Max Query Length	128	512	256	512	64
Dropout	0.1	0.1	0.1	0.1	0.1

Table 7: Hyper-parameters for fine-tuning. ♣: Hyper-parameters for BERT-based models. ♠: Hyper-parameters for RoBERTa-based models.

Case 1

Passage:

(0)A group of researchers at a remote jungle island outpost discover the natives are practicing voodoo and black magic. … (4)She returns years later as an adult with a group of mercenaries to attempt to uncover what happened to her parents. (5)Shortly after arriving at the island their boat's engine dies, stranding them. (6)Meanwhile elsewhere on the island a trio of hikers discover a cave, the same cave leading to the underground temple where the original curse was created. (7)After accidentally reviving the curse, the dead once again return to kill any who trespass on their island. (8)The mercenaries encounter their first zombie, who injures a member of the team. (9)Taking shelter in the remains of the old research facilities medical quarters they are soon joined by Chuck, the only surviving hiker. (10)Arming themselves with weapons left behind by the long dead research team, they make their stand as the dead once again rise. (11)One by one they are injured or killed, one of whom sacrifices himself to blow up the medical facility and his newly undead team members. (12)Jenny and Chuck flee, the only survivors remaining. (13)They stumble upon the cave once again, where the zombies appear and attack.

Question: Where did Chuck find weapons?

Option: From the previous research team.

Sentences Used: 9, 10.

BERT-Q: Answer: False Evidence: 0

BERT-Q w. R/S: Answer: True Evidence: 10, 9

Case 2

Passage:

(0)The film opens with Sunita, a medical student , and her friends working on a project about the human brain. (1)She wants to investigate the curious case of Sanjay Singhania, a notable city businessman, who is reported to have anterograde amnesia. (2)Her professor denies access to Sanjay's records as it is currently under criminal investigation. (3)Sunita, nonetheless, decides to investigate the matter herself. (4)Sanjay is introduced as he brutally murders a man. (5)He takes a Polaroid picture of the man, and writes on it ``done''. (6)It is revealed that Sanjay has anterograde amnesia where he loses his memory every 15 minutes. (7)Sanjay uses a system of photographs, notes, and tattoos on his body to recover his memory after each cycle. (8)It is revealed that Sanjay is ultimately out to avenge the death of his sweetheart Kalpana , and that he is systematically killing the people who were responsible for it. (9)His main target is ``Ghajini'', a notable social personality in the city. (10)Police Inspector Arjun Yadav, on the case of the serial murders, tracks Sanjay down to his flat and attacks and disables him. (11)Yadav finds two diaries where Sanjay has chronicled the events of 2005 and 2006. …

Question: Who denies Sunita access to Sanjay's records, who is reported to have anterograde amnesia, because they are under criminal investigation?

Option: Sunita's professor&Arjun Yadav.

Sentences Used: 1, 2.

RoBERTa-Q: Answer: False Evidence: 0

RoBERTa-Q w. R/S: Answer: True Evidence: 2, 1

Figure 4: Two cases from the development set of Multi-RC.

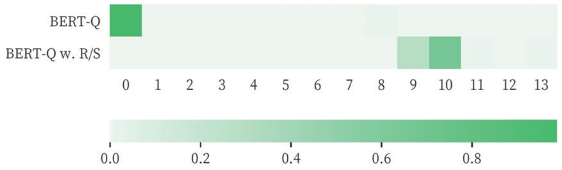

(a) Normalized attention weights for Case 1 in Figure 4.

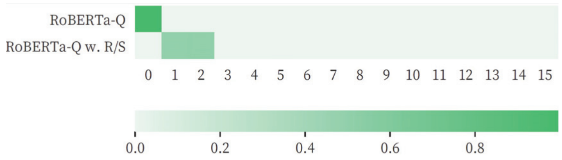

(b) Normalized attention weights for Case 2 in Figure 4.

Figure 5: Two cases of the normalized attention weights of evidence extraction.

CasEE: A Joint Learning Framework with Cascade Decoding for Overlapping Event Extraction

Jiawei Sheng[1,2], **Shu Guo**[3], **Bowen Yu**[1,2], **Qian Li**[4], **Yiming Hei**[5],
Lihong Wang[3], **Tingwen Liu**[1,2] and **Hongbo Xu**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]National Computer Network Emergency Response Technical Team/Coordination Center of China
[4]School of Computer Science and Engineering, Beihang University, Beijing, China
[5]School of Cyber Science and Technology, Beihang University, Beijing, China
`shengjiawei@iie.ac.cn, guoshu@cert.org.cn, wlh@isc.org.cn`

Abstract

Event extraction (EE) is a crucial information extraction task that aims to extract event information in texts. Most existing methods assume that events appear in sentences without overlaps, which are not applicable to the complicated overlapping event extraction. This work systematically studies the realistic event overlapping problem, where a word may serve as triggers with several types or arguments with different roles. To tackle the above problem, we propose a novel joint learning framework with cascade decoding for overlapping event extraction, termed as CasEE. Particularly, CasEE sequentially performs type detection, trigger extraction and argument extraction, where the overlapped targets are extracted separately conditioned on the specific former prediction. All the subtasks are jointly learned in a framework to capture dependencies among the subtasks. The evaluation on a public event extraction benchmark FewFC demonstrates that CasEE[1] achieves significant improvements on overlapping event extraction over previous competitive methods.

1 Introduction

Event Extraction (EE) is an important yet challenging task in natural language understanding. Given a sentence, an event extraction system ought to identify event types, triggers and arguments appearing in the sentence. As an example, Figure 1(b) presents an event mention of type `Share Reduction`, triggered by "reduced". There are several arguments, such as "Fuda Industry" playing the `subject` role in the event.

However, events often appear in sentences complicatedly, where the triggers and arguments may have overlaps in a sentence. This paper focuses on a challenging and realistic problem in EE: *overlapping event extraction*. Generally, we categorize all the overlapping cases into three patterns: 1) A word may serve as triggers with different event types across several events. Figure 1(a) shows the token "acquired" triggers an `Investment` event and a `Share Transfer` event at the same time. 2) A word may serve as arguments with different roles across several events. Figure 1(a) shows "Shengyue Network" plays an `object` role in the `Investment` event and a `subject` role in the `Share Transfer` event. 3) A word may serve as arguments playing different roles in one event. Figure 1(b) shows that "Fuda Industry" plays a `subject` role and a `target` role in an event. For simplicity, we call pattern 1) as *overlapped trigger problem*, and both pattern 2) and 3) as *overlapped argument problem* in the following sections. There are about 13.5% / 21.7% sentences having overlapped trigger/argument problems in the Chinese financial event extraction dataset, FewFC (Zhou et al., 2021).

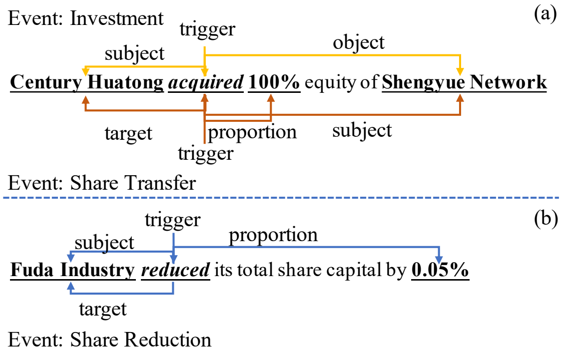

Figure 1: Examples of event overlapping problem: (a) Events with overlapped triggers and arguments; (b) An event with an overlapped argument in several roles.

Most existing EE studies assume that events appear in sentences without overlaps, which are not applicable to the complicated overlapping scenarios. Typically, current EE studies can be roughly

---

[1] The source code is available at `https://github.com/JiaweiSheng/CasEE`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 164–174
August 1–6, 2021.

categorized into two groups: 1) Traditional joint methods (Nguyen et al., 2016; Liu et al., 2018; Nguyen and Nguyen, 2019), which simultaneously extract triggers and arguments by a unified decoder labeling the sentence only once. However, they fail in extracting overlapped targets due to label conflicts, where a token may have several typed labels but only one label can be assigned. 2) Pipeline methods (Chen et al., 2015; Yang et al., 2019; Du and Cardie, 2020b), which sequentially extract triggers and arguments in separate stages. Yang et al. (2019) attempts to tackle the overlapped argument problem in the pipeline manner, but overlooks the overlapped trigger problem. Nevertheless, the pipeline methods neglect the feature-level dependencies between the trigger and arguments, and suffer from error propagation. In our knowledge, existing researches in EE neglect overlapping problems or only focus on one overlapping problem. Few researches simultaneously solve all the three mentioned overlapping patterns.

To address the above issues, we propose CasEE, a joint learning framework with Cascade decoding for overlapping Event Extraction. Specifically, CasEE realizes event extraction with a shared textual encoder and three decoders for type detection, trigger extraction and argument extraction. To extract overlapped targets across events, CasEE sequentially decodes the three subtasks, conducting trigger extraction and argument extraction according to the former predictions. Such a cascade decoding strategy extracts event elements according to the different conditions, so that the overlapped targets can be extracted in separate phases. A condition fusion function is designed to explicitly model the dependencies between adjacent subtasks. All the subtask decoders are jointly learned to further build connections among subtasks, which refines the shared textual encoder with feature-level interactions among downstream subtasks.

The contributions of this paper are three-fold:

(1) We systematically investigate the overlapping problems in EE, and categorize them into three patterns. To the best of our knowledge, this paper is among the first to simultaneously tackle all the three overlapping patterns.

(2) We propose CasEE, a novel joint learning framework with cascade decoding, to simultaneously solve all the three overlapping patterns.

(3) We conduct experiments on a public Chinese financial event extraction benchmark, FewFC. Experimental results reveal that CasEE achieves significant improvements on overlapping event extraction over existing competitive methods.

2 Related Work

Current EE research can be roughly categorized into two groups: 1) Traditional joint methods (Li et al., 2013; Nguyen et al., 2016; Nguyen and Nguyen, 2019; Liu et al., 2018; Sha et al., 2018) that perform trigger extraction and argument extraction simultaneously. They solve the task in a sequence labeling manner, and extract triggers and arguments by tagging the sentence only once. However, these methods fail in solving overlapping event extraction since the overlapping tokens would cause label conflicts when forced to have more than one label. 2) Pipeline methods (Chen et al., 2015; Yang et al., 2019; Wadden et al., 2019; Li et al., 2020; Du and Cardie, 2020b; Liu et al., 2020; Chen et al., 2020) that perform trigger extraction and argument extraction in separate stages. Though pipeline methods have the potential capacity to solve overlapping EE, they usually lack explicit dependencies between triggers and arguments, and suffer from error propagation. Among the researches, Yang et al. (2019) and Xu et al. (2020) solve the overlapped argument problem, but overlook the overlapped trigger problem, thus can not discern correct triggers for argument extraction. All the above methods can not simultaneously solve all the overlapping patterns in event extraction.

The overlapping problem has also been explored in other information extraction tasks outside event extraction. Luo and Zhao (2020) tackles nested named entity recognization with bipartite flat-graph networks. Zeng et al. (2018) tackles overlapped relational triple extraction by applying a sequence-to-sequence paradigm with a copy mechanism. Wei et al. (2020) and Yu et al. (2020) extract overlapped relational triples with a novel cascade tagging strategy, which inspire us to solve overlapping event extraction in the cascade decoding paradigm. Wang et al. (2020) further discusses the propagation error in cascade decoding. All the above researches are proposed for other tasks, which can not be directly transferred for overlapping event extraction due to the complicated event extraction definition.

3 Our Approach

Given an input sentence, the goal of EE is to identify triggers with their event types and arguments

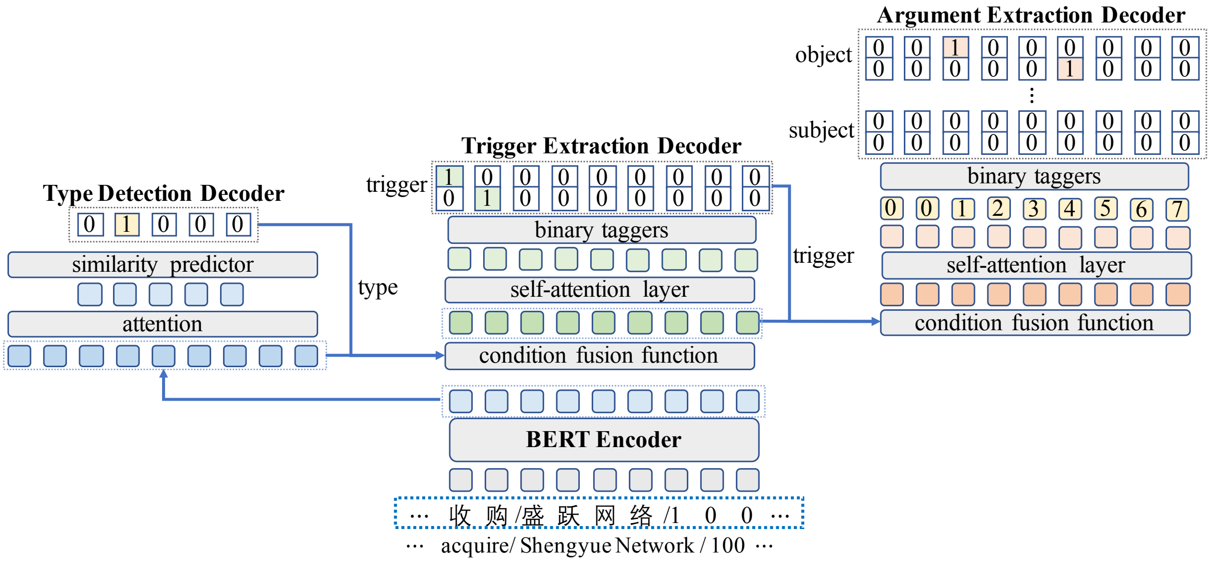

Figure 2: The overview of our proposed approach, CasEE, which contains a shared BERT encoder, a type detection decoder, a trigger extraction decoder and an argument extraction decoder.

with their corresponding roles, where triggers and arguments may overlap on some tokens. To tackle this problem, we propose a training objective at the event level. Formally, according to the pre-defined event schema, we have an event type set $\mathcal{C}$ and an argument role set $\mathcal{R}$. The overall goal is to predict all events in gold set $\mathcal{E}_x$ of the sentence x. We aim to maximize the joint likelihood of training data $\mathcal{D}$:

$$\prod_{x\in\mathcal{D}}\Big[\prod_{(c,t,a_r)\in\mathcal{E}_x} p((c,t,a_r))|x)\Big]$$

$$=\prod_{x\in\mathcal{D}}\Big[\prod_{c\in\mathcal{C}_x} p(c|x)\prod_{t\in\mathcal{T}_{x,c}} p(t|x,c)\prod_{a_r\in\mathcal{A}_{x,c,t}} p(a_r|x,c,t)\Big] \tag{1}$$

where $\mathcal{C}_x$ denotes the set of types occurring in x, $\mathcal{T}_{x,c}$ denotes the trigger set of type c, and $\mathcal{A}_{x,c,t}$ denotes the argument set of type c and trigger t. Note that each c is a type in $\mathcal{C}$, each t is a trigger word, and each $a_r \in \mathcal{A}_x$ is an argument word corresponding to its own role $r \in \mathcal{R}$. Eq. (1) exploits the fact of dependencies among the type, trigger and argument. Actually, it motivates us to learn a type detection decoder $p(c|x)$ to detect event types occurring in the sentence, a trigger extraction decoder $p(t|x,c)$ to extract triggers of type c, and an argument extraction decoder $p(a_r|x,c,t)$ to extract role-specific arguments with type c and trigger t.

Such a task decomposition solves all the event overlapping patterns claimed in the Introduction. Specifically, we first detect event types occurring in the sentence. When extracting triggers, we only predict the triggers with a specific type, thus the triggers overlapped across several events will be predicted in separate phases. Similarly, when extracting arguments, we predict the arguments with a specific type and trigger, thus the arguments overlapped across several events will also be predicted in separate phases. Since we adopt role-specific taggers in argument extraction, the overlapped arguments having several roles in an event can be predicted separately with specific taggers. All the predictions in type detection, trigger extraction and argument extraction form the final prediction.

Figure 2 demonstrates the details of CasEE. CasEE adopts a shared BERT encoder to capture textual features, and three decoders for type detection, trigger extraction and argument extraction. Since all subtasks are jointly learned in contrast to previous pipeline methods (Yang et al., 2019; Li et al., 2020), CasEE could capture feature-level dependencies among subtasks. For prediction, CasEE sequentially predicts event types, triggers and arguments in the cascade decoding process.

3.1 BERT Encoder

To capture the feature-level dependencies among subtasks, we share the textual representations of each sentence. As BERT has shown performance improvements across multiple NLP tasks, we adopt BERT (Devlin et al., 2019) as our textual encoder. BERT is a bi-directional language representation model based on transformer architecture (Vaswani et al., 2017), which generates textual representations conditioned on token context and remains rich textual information. Formally, the sentence with N tokens is denoted as $x = w_1, w_2, ..., w_N$. We input the tokens into BERT, and then obtain the hidden states $\mathbf{H} = \mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N$ as the token representations for the following downstream subtasks.

3.2 Type Detection Decoder

Since we tackle the overlapped trigger problem by extracting triggers conditioned on the type predictions, we devise a type detection decoder to predict event types. Inspired by event detection without triggers (Liu et al., 2019), we adopt attention mechanism to detect event types, capturing the most relative context for each possible type. Specifically, we randomly initialize embedding matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{C}|\times d}$ as the type embeddings. We define a similarity function δ to measure the relevance between the candidate type $\mathbf{c} \in \mathbf{C}$ and each token representation $\mathbf{h}_i$. To fully capture the similarity information in different aspects, we achieve δ with an expressive learnable function. According to the relevance scores, we obtain the sentence representation $\mathbf{s}_c$ adaptive to the type. The details are as follows:

$$\delta(\mathbf{c}, \mathbf{h}_i) = \mathbf{v}^\intercal \tanh(\mathbf{W}[\mathbf{c}; \mathbf{h}_i; |\mathbf{c} - \mathbf{h}_i|; \mathbf{c} \odot \mathbf{h}_i])$$
$$\mathbf{s}_c = \sum_{i=1}^{N} \frac{\exp(\delta(\mathbf{c}, \mathbf{h}_i))}{\sum_{j=1}^{N} \exp(\delta(\mathbf{c}, \mathbf{h}_j))} \mathbf{h}_i \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{4d\times 4d}$ and $\mathbf{v} \in \mathbb{R}^{4d\times 1}$ are learnable parameters, $|\cdot|$ is an absolute value operator, $\odot$ is the element-wise production, and $[\cdot;\cdot]$ denotes the concatenation of representations.

Finally, we predict event types by measuring the similarity of the adaptive sentence representation $\mathbf{s}_c$ and the type embedding $\mathbf{c}$ with the same similarity function δ. Then, the predicted probability of each event type c occurring in the sentence is:

$$\hat{c} = p(c|x) = \sigma(\delta(\mathbf{c}, \mathbf{s}_c)) \tag{3}$$

where σ denotes sigmoid function. We select the event type with $\hat{c} > \xi_1$ as results, where $\xi_1 \in [0, 1]$ is a scalar threshold. All predicted types in sentence x form event type set $\mathcal{C}_x$. The decoder learnable parameter $\theta_{td} \triangleq \{\mathbf{W}, \mathbf{v}, \mathbf{C}\}$.

3.3 Trigger Extraction Decoder

To discern overlapped triggers with several types, we extract triggers conditioned on a specific type $c \in \mathcal{C}_x$. This decoder contains a condition fusion function, a self-attention layer, and a pair of binary taggers for triggers.

To model the conditional dependency between type detection and trigger extraction, we devise a condition fusion function ϕ to integrate condition information into textual representation. Specifically, we obtain the conditional token representation $\mathbf{g}_i^c$ by integrating the type embedding $\mathbf{c}$ into the token representation $\mathbf{h}_i$ as:

$$\mathbf{g}_i^c = \phi(\mathbf{c}, \mathbf{h}_i) \tag{4}$$

Actually, ϕ can be achieved by concatenation, addition operator or gate mechanism. To fully generate conditional representations in the statistical aspect, we introduce an effective and general mechanism, conditional layer normalization (CLN) (Su, 2019; Yu et al., 2021), to achieve ϕ. CLN is mostly based on the well-known layer normalization (Ba et al., 2016), but can dynamically generate gain γ and bias β based on the condition information. Given a condition embedding $\mathbf{c}$ and a token representation $\mathbf{h}_i$, CLN is formulated as:

$$\begin{aligned} \mathrm{CLN}(\mathbf{c}, \mathbf{h}_i) &= \gamma_c \odot (\frac{\mathbf{h}_i - \mu}{\sigma}) + \beta_c, \\ \gamma_c &= \mathbf{W}_\gamma \mathbf{c} + \mathbf{b}_\gamma, \beta_c = \mathbf{W}_\beta \mathbf{c} + \mathbf{b}_\beta \end{aligned} \tag{5}$$

where $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}$ are the mean and standard variance taken across the elements of $\mathbf{h}_i$, and $\gamma_c \in \mathbb{R}^d$ and $\beta_c \in \mathbb{R}^d$ are the conditional gain and bias, respectively. In this way, the given condition representation is encoded into the gain and bias, and then integrated into contextual representations.

To further refine representations for trigger extraction, we adopt a self-attention layer over the conditional token representations. Formally, the refined token representations are derived as:

$$\mathbf{Z}^c = \mathrm{SelfAttention}(\mathbf{G}^c) \tag{6}$$

where $\mathbf{G}^c$ is the representation matrix composed of $\mathbf{g}_i^c$. For details of the self-attention layer, please refer to Vaswani et al. (2017).

To predict triggers, we devise a pair of binary taggers. For each token w_i, we predict whether it corresponds to a start or end position of a trigger as:

$$\begin{aligned} \hat{t}_i^{sc} &= p(t_s|w_i, c) = \sigma(\mathbf{w}_{t_s}^\intercal \mathbf{z}_i^c + b_{t_s}) \\ \hat{t}_i^{ec} &= p(t_e|w_i, c) = \sigma(\mathbf{w}_{t_e}^\intercal \mathbf{z}_i^c + b_{t_e}) \end{aligned} \tag{7}$$

where σ denotes sigmoid function, and $\mathbf{z}_i^c$ denotes the i-th token representation in $\mathbf{Z}^c$. We select tokens with $\hat{t}_i^{sc} > \xi_2$ as the start positions, and those with $\hat{t}_i^{ec} > \xi_3$ as end positions, where $\xi_2, \xi_3 \in [0, 1]$ are scalar thresholds. To obtain the trigger word t, we enumerate all the start positions and search the nearest following end position in the sentence, and the tokens between the start and end position form an entire trigger. In this way, the overlapped triggers can be extracted separately according to the type in separate phases. All the predicted trigger t of type c in sentence s forms the set $\mathcal{T}_{c,s}$. The decoder parameter θ_{te} includes all the parameters in the condition fusion function, the self-attention layer and the trigger taggers.

3.4 Argument Extraction Decoder

To tackle the overlapped argument problem, we extract role-specific arguments conditioned on both the specific event type $c \in \mathcal{C}_s$ and event trigger $t \in \mathcal{T}_{c,s}$. This decoder also contains a condition fusion function, a self-attention layer, and a group of role-specific binary tagger pairs for arguments.

We further integrate the trigger information into the typed textual representation $\mathbf{g}_i^c$ in Eq. (4) with function ϕ achieved by CLN. Here we take the average of the start and end position token representations of t as the trigger embedding. We also adopt a self-attention layer to derive the refined textual representations $\mathbf{Z}^{ct'}$. To be aware of the trigger position, we adopt the relative position embedding as used in Chen et al. (2015), which indicates the relative distance from current token to the trigger boundary token. Finally, the token representations $\mathbf{Z}^{ct}$ for argument extraction are derived as:

$$\mathbf{Z}^{ct} = [\mathbf{Z}^{ct'}; \mathbf{P}] \tag{8}$$

where $\mathbf{P} \in \mathbb{R}^{N \times d_p}$ is the relative position embeddings, d_p is the dimension, and $[\cdot;\cdot]$ denotes the concatenation of representations.

To predict arguments in roles, we devise a group of role-specific tagger pairs. For each token w_i, we predict whether it corresponds to a start or end position of an argument of the role $r \in \mathcal{R}$ as:

$$\begin{aligned} \hat{r}_i^{sct} &= p(a_r^s|w_i, c, t) = \mathrm{I}(r, c)\sigma(\mathbf{w}_{r_s}^{\intercal}\mathbf{z}_i^{ct} + b_{r_s}) \\ \hat{r}_i^{ect} &= p(a_r^e|w_i, c, t) = \mathrm{I}(r, c)\sigma(\mathbf{w}_{r_e}^{\intercal}\mathbf{z}_i^{ct} + b_{r_e}) \end{aligned} \tag{9}$$

where σ denotes sigmoid function, and $\mathbf{z}_i^{ct}$ denotes the i-th token representation in $\mathbf{Z}^{ct}$. Since not all roles belonging to the specific type c, we adopt an indicator function $\mathrm{I}(r, c)$ to indicate whether the role r belongs to the type c according to the pre-defined event scheme. To make the indicator function derivable, we parameterize $\mathrm{I}(r, c)$ to learn with the model parameters. Specifically, given the type embedding $\mathbf{c} \in \mathcal{C}$, we build the connection between the type and roles as:

$$\mathrm{I}(r, c) = \sigma(\mathbf{w}_r^{\intercal}\mathbf{c} + b_r) \tag{10}$$

where σ denotes sigmoid function, $\mathbf{w}_r, b_r$ are parameters associated with the role r. For each role r, we select tokens with $\hat{r}_i^{sct} > \xi_4$ as the start positions, and those with $\hat{r}_i^{ect} > \xi_5$ as end positions, where $\xi_4, \xi_5 \in [0, 1]$ are scalar thresholds. To obtain the argument word a with role r, we enumerate all the start positions and search the nearest following end position in the sentence, and the tokens between the start and end position form an entire argument. In this way, the overlapped arguments can be extracted separately according to the different types and triggers with role-specific taggers. All the predicted argument a_r with type c and trigger t in sentence x forms the set $\mathcal{A}_{t,c,x}$. The decoder parameter θ_{ae} includes the type embedding matrix $\mathbf{C}$, and all parameters in the condition fusion function, the self-attention layer and the argument taggers.

3.5 Model Training

To train the model, we take log of Eq (1), and the overall objective $\mathcal{J}(\Theta)$ is deployed as:

$$\begin{aligned} \sum_{x \in \mathcal{D}} \Big[\sum_{c \in \mathcal{C}_x} & \log p_{\theta_1}(c|x) + \\ & \sum_{t \in \mathcal{T}_{x,c}} \log p_{\theta_2}(t|x, c) + \sum_{a_r \in \mathcal{A}_{x,c,t}} \log p_{\theta_3}(a_r|x, c, t) \Big] \end{aligned} \tag{11}$$

where $\Theta \triangleq \{\theta_1, \theta_2, \theta_3\}$; $p_{\theta_1}(c|x)$, $p_{\theta_2}(t|x, c)$, and $p_{\theta_3}(a_r|x, c, t)$ for the subtasks are defined as:

$$\begin{aligned} p_{\theta_1}(c|x) &= (\hat{c})^{\bar{c}}(1 - \hat{c})^{(1-\bar{c})} \\ p_{\theta_2}(t|x, c) &= \prod_{z \in \{s,e\}} \prod_{i=1}^{N} (\hat{t}_i^{zc})^{\bar{t}_i^{zc}}(1 - \hat{t}_i^{zc})^{(1-\bar{t}_i^{zc})} \\ p_{\theta_3}(a_r|x, c, t) &= \prod_{r \in \mathcal{R}} \prod_{z \in \{s,e\}} \prod_{i=1}^{N} (\hat{r}_i^{zct})^{\bar{r}_i^{zct}}(1 - \hat{r}_i^{zct})^{(1-\bar{r}_i^{zct})} \end{aligned} \tag{12}$$

where $\hat{c}$, $\hat{t}_i^{sc}$, $\hat{t}_i^{ec}$, $\hat{r}_i^{sct}$, $\hat{r}_i^{ect}$ are the predicted probabilities in Eq (3), Eq (7), Eq (9), and $\bar{c}$, $\bar{t}_i^{sc}$, $\bar{t}_i^{ec}$, $\bar{r}_i^{sct}$, $\bar{r}_i^{ect}$ are the true $0/1$ labels of the training data, respectively. $\theta_1 \triangleq \{\theta_{bert}, \theta_{td}\}$, $\theta_2 \triangleq \{\theta_{bert}, \theta_{te}\}$, $\theta_3 \triangleq \{\theta_{bert}, \theta_{ae}\}$, where $\theta_{bert}, \theta_{td}, \theta_{te}, \theta_{ae}$ denote parameters in BERT, type detection, trigger extraction and argument extraction, respectively. We train the model by maximizing $\mathcal{J}(\Theta)$ through Adam stochastic gradient descent (Kingma and Ba, 2015) over the shuffled mini-batches.

	#Overlap	#Normal	#Sentence	#Event
Training	1,560	5,625	7,185	10,277
Validation	205	694	899	1,281
Testing	210	688	898	1,332
All	1,975	7,007	8,982	12,890

Table 1: Statistics of the dataset. Each column denotes the number of the sentences with overlapped elements, the sentences without overlapped elements, all the sentences and all the events.

4 Experiments

In this section, we conduct experiments to evaluate the performance of CasEE.

4.1 Dataset and Evaluation Metric

We conduct experiments[2] on a Chinese financial event extraction benchmark FewFC (Zhou et al., 2021). We split data with 8:1:1 for training/validation/testing. Table 1 shows more details.

For evaluation, we follow the traditional evaluation metrics (Chen et al., 2015; Du and Cardie, 2020b): 1) Trigger Identification (TI): A trigger is correctly identified if the predicted trigger span matches with a golden span; 2) Trigger Classification (TC): A trigger is correctly classified if it is correctly identified and assigned to the correct type; 3) Argument Identification (AI): An argument is correctly identified if its event type is correctly recognized and the predicted argument span matches with a golden span; 4) Argument Classification (AC): an argument is correctly classified if it is correctly identified and the predicted role matches with a golden role. We report Precision (P), Recall (R) and F measure (F1) for each of the four metrics.

4.2 Comparision Methods

Though various models have recently been developed for EE, few researches are investigated to solve overlapping event extraction. We attempt to develop the following baselines based on current solutions. For the realistic consideration, no candidate entities are previously known for EE.

Joint sequence labeling methods. This kind of method formulates event extraction into a sequence labeling task. **BERT-softmax** (Devlin et al., 2019) adopts BERT to learn textual representations and uses hidden states for classifying event triggers and arguments. **BERT-CRF** adopts conditional random field (CRF) to capture label dependencies, which is adopted in (Du and Cardie, 2020a) for document-level event extraction. **BERT-CRF-joint** borrows idea from joint extraction of entity and relation (Zheng et al., 2017), which adopts joint labels of the type and role as `B/I/O-type-role`. All the above methods can not solve the overlapping problem due to label conflicts.

Pipelined event extraction methods. This kind of method solves event extraction with a pipeline manner. **PLMEE** (Yang et al., 2019) solves overlapped argument problem by extracting role-specific arguments according to the trigger. Motivated by current Machine Reading Comprehension (MRC) based EE studies (Li et al., 2020; Du and Cardie, 2020b; Liu et al., 2020; Chen et al., 2020), we train multiple MRC BERTs for overlapping event extraction. We extend MQAEE (Li et al., 2020) for multi-span extraction and re-assemble the following methods[3] to consider conditions in EE: 1) The method first predicts types, and then predicts overlapped triggers/arguments according to the type, termed as **MQAEE-1**. 2) The method first predicts overlapped triggers with types, and then predicts overlapped arguments according to the typed triggers, termed as **MQAEE-2**. 3) The method sequentially predicts types, predicts overlapped triggers according to the type, and predicts overlapped arguments according to the type and trigger, termed as **MQAEE-3**. All the above pipeline methods could solve (or partly solve) overlapping event extraction.

4.3 Implementation Details

We adopt source code for PLMEE with its best hyper-parameters reported in the original literature. To achieve other baselines, we implement the code based on the Transformers library (Wolf et al., 2020). For all the methods, we adopt Chinese BERT-Base model[4] as the textual encoder, which has 12 layers, 768 hidden units and 12 attention heads. We use the same value for the common hyper-parameters among the methods, including the optimizer, learning rate, batch size and epoch. For all the hyper-parameters, we adopt grid search

---

[2] Though ACE 2005 dataset is usually used to evaluate traditional EE models, we observe that it contains a low proportion of sentences with overlapped argument problem (nearly 10% reported in Yang et al. (2019)), and doesn't exist sentences with overlapped trigger problem.

[3] For more details, please refer to the Appendix B.

[4] `https://huggingface.co/bert-base-chinese`

	TI(%)			TC(%)			AI(%)			AC(%)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BERT-softmax	89.8	79.0	84.0	80.2	61.8	69.8	74.6	62.8	68.2	72.5	60.2	65.8
BERT-CRF	**90.8**	80.8	85.5	**81.7**	63.6	71.5	75.1	64.3	69.3	72.9	61.8	66.9
BERT-CRF-joint	89.5	79.8	84.4	80.7	63.0	70.8	**76.1**	63.5	69.2	**74.2**	61.2	67.1
PLMEE	83.7	85.8	84.7	75.6	74.5	75.1	74.3	67.3	70.6	72.5	65.5	68.8
MQAEE-1	90.1	85.5	87.7	77.3	76.0	76.6	62.9	71.5	66.9	51.7	70.4	59.6
MQAEE-2	89.1	85.5	87.4	79.7	76.1	77.8	70.3	68.3	69.3	68.2	66.5	67.3
MQAEE-3	88.3	86.1	87.2	75.8	76.5	76.2	69.0	67.9	68.5	67.2	65.9	66.5
CasEE	89.4	**87.7**	**88.6**	77.9	**78.5**	**78.2**	72.8	**73.1**	**72.9**	71.3	**71.5**	**71.4**

Table 2: Results of event extraction on FewFC dataset, where TI, TC, AI, AC denote trigger identification, trigger classification, argument identification and argument classification, respectively.

strategy. We train all the methods with an Adam weight decay optimizer. The initial learning rate is tuned in $[1e^{-5}, 5e^{-5}]$ for BERT parameters and $[1e^{-4}, 3e^{-4}]$ for other parameters. The warming up proportion for learning rate is 10%, and the max training epoch is set to 20. The batch size is set to 8. For CasEE, the dimension d_p of the relative position embedding is tuned in $\{16, 32, 64\}$. To avoid overfitting, we apply dropout to BERT hidden states with the rate tuned in $[0, 1]$. Besides, the thresholds $\xi_1, \xi_2, \xi_3, \xi_4, \xi_5$ for prediction are tuned in $[0, 1]$. We select the best model leading to the highest performance on the validation data. The optimal hyper-parameter settings are tuned by grid search, listed in the Appendix A.

4.4 Main Results

The performance of all methods on the FewFC dataset is shown in 2. The table reveals that:

(1) Compared to the joint sequence labeling methods, CasEE achieves better performance on the F1 score. Specifically, CasEE achieves improvements of 4.5% over BERT-CRF and 4.3% over BERT-CRF-joint on F1 score of AC, respectively. Besides, CasEE produces higher results on the recall of the evaluation metrics, since the sequence labeling methods have label conflicts that only one label can be predicted for those multi-label tokens. The results demonstrate the effectiveness of CasEE on overlapping event extraction.

(2) Compared to the pipeline methods, our method also outperforms them on the F1 score. The results show that CasEE achieves 3.1% and 2.6% improvements on F1 score of TC and AC over PLMEE, indicating the importance of solving the overlapped trigger problem in EE. Though the MRC based baselines can extract the overlapped triggers and arguments, CasEE still achieves better performance. Specifically, CasEE improves by a relative margin of 4.1% against the strong baseline MQAEE-2. The reason may be that CasEE jointly learns textual representations for subtasks, building helpful interactions and connections among the subtasks. The results demonstrate the superiority of CasEE over the above pipeline baselines.

Variants	TI (%)	TC (%)	AI (%)	AC (%)
BERT-softmax	76.5	49.0	56.1	53.5
BERT-CRF	77.9	52.4	61.0	58.4
BERT-CRF-joint	77.8	52.0	58.8	56.8
PLMEE	80.7	66.6	63.2	61.4
MQAEE-1	87.0	73.4	69.4	62.3
MQAEE-2	83.6	70.4	62.1	60.1
MQAEE-3	87.5	73.7	64.3	62.2
Ours	**89.0**	**74.9**	**71.5**	**70.3**

Table 3: Results of overlap sentences in testing. F1 scores are reported for each evaluation metric.

Variants	TI (%)	TC (%)	AI (%)	AC (%)
BERT-softmax	86.9	79.9	**76.2**	**74.1**
BERT-CRF	88.4	80.8	74.9	72.8
BERT-CRF-joint	86.9	79.9	76.1	74.0
PLMEE	86.4	79.7	75.7	74.0
MQAEE-1	88.0	78.5	65.1	57.7
MQAEE-2	**89.0**	**82.0**	74.2	72.3
MQAEE-3	87.1	77.6	71.3	69.6
Ours	88.4	80.2	74.0	72.3

Table 4: Results of normal sentences in testing. F1 scores are reported for each evaluation metric.

4.5 Analysis on Overlap/Normal Data

To further understand the performance in testing, we divide the original test data into two groups: the sentences with overlapped elements and the sentences without overlapped elements.

Variants	P (%)	R (%)	F1 (%)
MaxP	88.1	89.2	88.5
MeanP	88.3	89.8	88.6
CLS	**88.9**	88.7	88.5
CasEE	87.5	**91.9**	**89.2**

Table 5: Results of type detection decoder variants. The evaluation metric is precision, recall and F1 score in macro average of type classification.

Variants	P (%)	R (%)	F1 (%)
w/o self-attention	89.2	88.1	88.6
w/o condition	86.5	87.6	87.0
repl. concatenation	89.3	87.7	88.5
repl. addition	**90.4**	88.8	89.6
repl. gate mechanism	90.2	88.2	89.1
CasEE	90.1	**90.2**	**90.1**

Table 6: Results of trigger extraction decoder variants. The evaluation metric is precision, recall and F1 score on TC metric with oracle results of type detection.

Performance on Overlap Sentences. As shown in table 3, our method significantly outperforms previous methods on the overlap sentences. The improvements may come from the property that our method avoids label conflicts compared to the sequence labeling methods, and builds more effective feature-level connections among subtasks compared to the pipeline methods.

Performance on Normal Sentences. As shown in table 4, our method still performs acceptable results on the normal sentences without overlapped event elements. The sequence labeling methods reveal similiar results on trigger extraction but relatively better results on argument extraction, where they avoid potential propagation errors of the cascade decoding. Besides, PLMEE performs similar results on trigger extraction but relatively better results on argument extraction, where the reason may be that it adopts elaborate re-weighting losses for different argument roles as in its original literature. In addition, MQAEE-2 predicts more accurate triggers since it jointly predicts triggers with types, but it unfortunately ignores feature-level connections among the subtasks, making the argument extraction results similar to CasEE. Even so, the vanilla CasEE still conducts acceptable performance on the normal sentences compared to the baselines. We would further tackle the potential propagation errors and improve the performance for the general event extraction in the future work.

Variants	P (%)	R (%)	F1 (%)
w/o self-attention	82.8	81.7	82.2
w/o indicator function	84.1	81.4	82.7
w/o position embedding	83.2	81.5	82.3
w/o condition	**84.7**	78.2	81.3
repl. concatenation	84.0	79.3	81.6
repl. addition	84.2	78.2	81.1
repl. gate mechanism	84.6	80.2	82.4
CasEE	84.1	**83.7**	**83.9**

Table 7: Results of argument extraction decoder variants. The evaluation metric is precision, recall and F1 score on AC metric with oracle results of type detection and trigger extraction.

4.6 Discussion for Model Variants

To investigate the effectiveness of each module, we conduct variant experiments for CasEE.

Detection Module Variants. Table 5 shows performance of type detection variants. Specifically, MaxP/MeanP aggregates textual representations by applying max/mean pooling over BERT hidden states; CLS utilizes the hidden state of the special token `<CLS>` as the sentence representation. The results show that our method outperforms all the above variants on F1 score, indicating that learning sentence representation adaptive to the event type produces better representation for type detection.

Extraction Module Variants. Table 6 and Table 7 show performance of decoder variants for trigger extraction and argument extraction, respectively. We remove the self-attention layer in the both extraction decoders, and remove the relative position embeddings and the indicator function in the argument extraction decoder. The results demonstrate the effectiveness of each module.

Furthermore, we conduct experiments to explore the impact of condition fusion function ϕ. The experiments include: 1) we simply remove condition integrate function; 2) we achieve ϕ by concatenating the condition and token representations; 3) we achieve ϕ by simply adding the condition embedding to token representations; 4) we achieve ϕ by the gate mechanism, which adds the condition embedding to token representations according to a learnable trade-off factor. The results show that the performance without condition fusion function decline significantly on the F1 score in the two decoders, since the model can not discern different targets to extract in the sentence. Besides, empirical results also show that CLN performs better performance than other fusion functions on F1 scores in the two decoders, indicating that CLN can generate better conditional token representations for downstream subtasks.

5 Conclusion

This paper proposes a joint learning framework with cascade decoding for overlapping event extraction, termed as CasEE. Previous studies usually assume that events appear in sentences without overlaps, which are not applicable to the complicated overlapping scenarios. CasEE sequentially performs type detection, trigger extraction and argument extraction, where the overlapped targets are separately extracted conditioned on former predictions. All subtasks are jointly learned to capture dependencies among subtasks. Experiments on the public dataset demonstrate that our model outperforms previous competitive methods on overlapping event extraction. Our future work may further tackle the potential error propagation problem in the cascade decoding paradigm, and improve the performance for the general event extraction.

Acknowledgments

Corresponding authors: Lihong Wang and Shu Guo. We thank all the anonymous reviewers for their insightful comments and constructive suggestions. This work is supported by the National Natural Science Foundation of China (No.61772151), and the Youth Innovation Promotion Association of CAS (Grant No. 2021153).

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, pages 167–176*.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP@EMNLP 2020, pages 74–83*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186.

Xinya Du and Claire Cardie. 2020a. Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8010–8020*.

Xinya Du and Claire Cardie. 2020b. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 671–683*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 829–838*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 73–82*.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1641–1651*.

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019. Event detection without triggers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 735–744*.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1247–1256*.

Ying Luo and Hai Zhao. 2020. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6408–6418*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 300–309*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *The Thirty-Third AAAI Conference on Artificial Intelligence, pages 6851–6858*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, pages 5916–5923*.

Jianlin Su. 2019. Conditional text generation based on conditional layer normalization.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, pages 5998–6008*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5783–5788*.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. 2020. Tplinker: Single-stage joint extraction of entities and relations through token pair linking. In *Proceedings of the 28th International Conference on Computational Linguistics, pages 1572–1582*.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1476–1488*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Nuo Xu, Haihua Xie, and Dongyan Zhao. 2020. A novel joint framework for multiple chinese events extraction. In *Chinese Computational Linguistics - 19th China National Conference, pages 174–183*.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, pages 5284–5294*.

Bowen Yu, Zhenyu Zhang, Jiawei Sheng, Tingwen Liu, Yubin Wang, Yucheng Wang, and Bin Wang. 2021. Semi-open information extraction. In *Proceedings of the Web Conference*.

Bowen Yu, Zhenyu Zhang, Xiaobo Shu, Tingwen Liu, Yubin Wang, Bin Wang, and Sujian Li. 2020. Joint extraction of entities and relations based on a novel decomposition strategy. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, pages 2282–2289*.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 506–514*.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1227–1236*.

Yang Zhou, Yubo Chen, Jun Zhao, Yin Wu, Jiexin Xu, and Jinlong Li. 2021. What the role is vs. what plays the role: Semi-supervised event argument extraction via dual question answering. In *Proceedings of AAAI-21*.

A Hyper-parameter Settings

Our implementation is based on PyTorch[5]. We trained our models with a NVIDIA TESLA T4 GPU. For re-implementation, we report our hyper-parameter settings on the dataset in Table 8. Note that the hyper-parameter settings are tuned on the validation data by grid search with 3 trials.

Hyper-parameter	CasEE
type embedding dimension d	768
position embedding dimension d_p	64
dropout rate of decoders	0.3
batch size	8
training epoch	20
initial learning rate of BERT	$2e^{-5}$
learning rate of decoders	$1e^{-4}$
threshold ξ_1	0.5
threshold ξ_2	0.5
threshold ξ_3	0.5
threshold ξ_4	0.5
threshold ξ_5	0.5

Table 8: Hyper-parameter settings of CasEE.

B Details of MRC Based Baselines

Here we describe the details of the extended MRC baselines. Since the MRC paradigm could place condition information in the questions, we extend it to solve the overlapping event extraction.

MQAEE-1 contains two models: 1)A BERT classifier to detect event types; 2) A MRC BERT to extract triggers and arguments. The question template is like `<type>` to predict triggers with type `type`, and `<role>` to predict arguments with role `role`. Though this method neglects associations between the trigger and argument, it tackles overlapped trigger problem and overlapped argument problem since the overlapped targets are extracted separately according to different questions.

MQAEE-2 contains two models: 1) A MRC BERT to extract all triggers with types. The question template is a single word `trigger` to predict all typed triggers. 2) A MRC BERT to extract arguments in different roles. The question template is like `<type>and<trigger>` to predict all arguments associated with the type `type` and the trigger `trigger`. This method tackles overlapped trigger problem with multiple taggers, and tackles overlapped argument problem by extracting argument separately according to both the type and trigger.

MQAEE-3 contains three models: 1)A BERT classifier to detect event types; 2) A MRC BERT to extract triggers with different types. The question template is like `<type>` to predict triggers with type `type`. 3) A MRC BERT to extract arguments in different roles. The question template is like `<type>and<trigger>` to predict all arguments associated with the type `type` and the trigger `trigger`. This method tackles overlapped trigger problem by extracting triggers according to the type, and tackles overlapped argument problem according to both the type and trigger.

---

[5] https://pytorch.org/

Discovering Topics in Long-tailed Corpora with Causal Intervention

Xiaobao Wu[1] **Chunping Li**[1] **Yishu Miao**[2]
[1]Tsinghua University [2]Imperial College London
`wxb18@mails.tsinghua.edu.cn`, `cli@mail.tsinghua.edu.cn`,
`y.miao20@imperial.ac.uk`

Abstract

Topic models are effective in capturing the latent semantics of large-scale textual data while existing methods are normally designed and evaluated on balanced corpora. However, it contradicts the fact that general corpora in our world are naturally long-tailed, and the long-tailed bias can highly impair the topic modeling performance. Therefore, in this paper, we propose a causal inference framework to explain and overcome the issues of topic modeling on long-tailed corpora. In a neat and elegant way, causal intervention is applied in training to take out the influence brought by the long-tailed bias. Extensive experiments on manually constructed and naturally collected datasets demonstrate that our model can mitigate the bias effect, greatly improve topic quality and better discover the hidden semantics on the tail.

1 Introduction

Topic models are proposed to discover the underlying topics and semantic structures from unlabelled text collections. Due to the effectiveness and interpretability, topic models have been applied in various downstream tasks like information retrieval (Wang et al., 2007), content summarization (Ma et al., 2012) and recommendation systems (McAuley and Leskovec, 2013). One of the most widely used topic models is Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a probabilistic graphical model using the conjugate of Dirichlet and Multinomial distribution and inferring the parameters with approximation methods(Griffiths and Steyvers, 2004; Blei et al., 2017). Recently, some popular neural topic models based on Variational AutoEncoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) have been introduced, such as Neural Variational Document Model (NVDM) (Miao et al., 2016) and Product of Experts LDA (ProdLDA) (Srivastava and Sutton, 2017). Compared to probabilistic ones, they can easily carry out the inference by gradient backpropagation.

However, these topic models are generally designed and evaluated on balanced corpora, such as the commonly used 20News (Lang, 1995) with evenly distributed labels through which we can infer that the latent topics are also evenly distributed. It hence conflicts with the fact that natural text collections are regularly long-tail distributed following Zipf's law (Reed, 2001), especially the textual data on social network platforms(Zhang and Luo, 2019). More precisely, Figure 1 illustrates that in a collected corpus, documents about some hot topics are numerous (*head topics*), while the documents about most topics are few (*tail topics*). Due to this bias, similar to long-tailed classification tasks where a classifier favors to predict an image as the head classes (Kang et al., 2020; Zhou et al., 2020), topic models on long-tailed corpora tend to reveal the semantics of documents about head topics and ignore the documents about tail topics to a great extent. Namely, the discovered topics are mostly about the latent head ones in the corpus. As a result, their diversities are much impaired and incomplete to represent the whole semantics of a corpus. Thus, it is crucial to explore effective ways for long-tailed topic modeling.

Different from other long-tailed tasks like image classification or relation extraction, the key challenge of this problem lies in that topic modeling is originally designed for unlabelled datasets, so we have no access to classification labels to infer the latent global topic distributions while designing solutions[1]. Owing to this factor, we intend not to introduce complicated modules conditioning

---

[1] Admittedly, there are supervised topic models (Mcauliffe and Blei, 2007; Card et al., 2018), but the necessity of labels will hugely narrow their application scopes. So, we concentrate on solving the issue without additional labels.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 175–185
August 1–6, 2021.

on accessible labels, e.g., re-weighting (Mahajan et al., 2018) or re-sampling (Khan et al., 2017; Lin et al., 2017; Cui et al., 2019) approaches for other long-tail problems.

To overcome this challenge, in this paper, we present a Structural Causal Model (SCM) (Pearl et al., 2016; Pearl and Mackenzie, 2018) to precisely explain how the long-tailed bias undermines the topic modeling performance. Then, to remove the bias effect, we propose an approach via the causal intervention (Pearl et al., 2016) on topic distributions and adopt the backdoor adjustment (Pearl, 1995) to calculate the causality in the condition of no auxiliary information. Furthermore, we introduce a novel neural model named as **Deconfounded Topic Model (DecTM)** in the framework of VAE with deconfounded training through an approximation manner. Through comprehensive experiments, we manifest that our new model can mitigate the influence of the long-tailed bias and produce high-quality topics that are more diverse and better disclose the semantics of documents about tail topics. The main contributions of this paper can be concluded as follows:

1. We present a structural causal model to clarify how the problems of topic modeling are incurred by the long-tailed bias in detail;
2. We further propose a neat method to approximate the causal intervention for reducing bias influence, depending on which a novel neural topic is also introduced with deconfounded training;
3. We validate our model on both manually-constructed and extreme multi-label text classification datasets and demonstrate our model is effective to alleviate the impact of bias and greatly improve the topic quality compared to both probabilistic and neural baseline models.

2 Related Work

Topic Modeling Probabilistic topic models can date back to Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999) and LDA (Blei et al., 2003), derving numerous variants(Blei and Lafferty, 2006; Yan et al., 2013; Wu and Li, 2019). Previously, Wang et al. (2015) adapted LDA to discover long-tail semantics from large-scale corpora. Those models usually adopt Gibbs Sampling (Griffiths and Steyvers, 2004) or Variational Inference (Blei et al., 2017) for parameter estimations. Based on VAE (Kingma and Welling, 2014; Rezende et al., 2014), neural topic models (Miao et al., 2016; Srivastava and Sutton, 2017; Wu et al., 2020a) are introduced. They are derivation-free and can apply gradient backpropagation directly. Nevertheless, these former works including probabilistic and neural methods are normally evaluated on balanced datasets. Since long-tail distributed data are common in our natural world (Reed, 2001), this inspires us to find out how these topic models perform on long-tailed corpora and propose useful ways to alleviate the long-tailed bias impact.

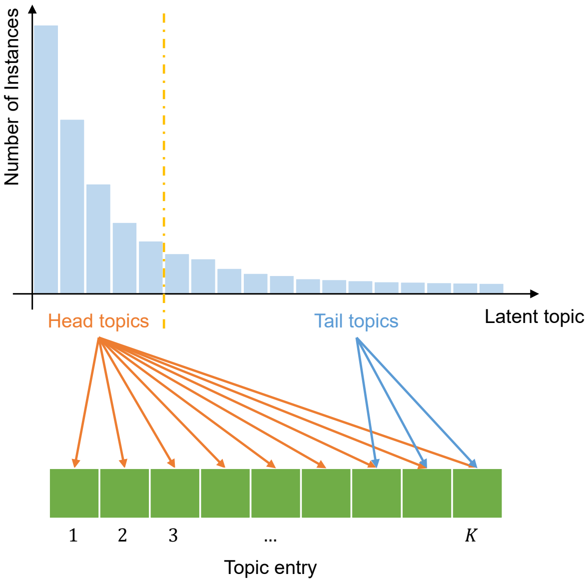

Figure 1: Illustration of topic entries assigned to documents in a long-tailed corpus.

Causal Inference Causal inference (Pearl et al., 2016) has been widely adopted in various fields for years, like psychology, epidemiology, and medicine (MacKinnon et al., 2007; Richiardi et al., 2013), providing solutions to investigate the causation between research objects. Recently, the causal inference has also increasingly attracted attention in computer vision and NLP society for removing the biases in datasets (Tang et al., 2020; Wu et al., 2020c) or providing counterfactual examples (Zeng et al., 2020) in domain-specific applications. In this paper, we propose to employ the causal inference mechanism to investigate wherefores for the issues of long-tailed topic modeling and propose a solution with deconfounded training via the intervention to alleviate the bias effect.

3 Method

In this section, we first explain how the long-tailed bias affects topic modeling from the perspective of causal inference, and then propose a novel model to overcome this issue with deconfounded training by the causal intervention.

3.1 SCM for Topic Modeling

First of all, we investigate the causal relationship between the latent variables in a topic model with a Structural Causal Model (SCM). SCMs are expressed visually by using directed acyclic graphs. In the graph, vertices are random variables, and directed edges represent direct causation from one variable to another (Pearl et al., 2016). There is a special vertex in the graph: *confounder*, a variable that influences both correlated and independent variables, creating a spurious statistical correlation. For example, considering an interesting study that chocolate consumption is statistically related to the number of Nobel prizes of a country (Dablander, 2020). Is it justified to argue that people can get Nobel prizes if they eat more chocolate? Common sense intuitively tells us this assertion is inaccurate. We can draw a causal graph to detail it: **chocolate consumption** ← **economy** → **number of Nobel prizes**, the chocolate consumption is usually higher in a developed country with good economy, and the number of Nobel prizes is also larger since the citizens' education level is higher in this country. Therefore, the economy acts as a confounder that creates a spurious correlation between chocolate consumption and the number of Nobel prizes.

Similar to the above example, we build a SCM shown in Figure 2a to describe how a biased corpus influences the text generation process of topic modeling. In the graph, $\boldsymbol{C}$ means the unobserved confounding bias in a long-tailed corpus. We note the vocabulary size is V and set K topic entries (the topic number is K) which means the model needs to discover K latent topics. In the setting of topic modeling, a topic entry k is interpreted as the related words and represented with a word distribution $\boldsymbol{\beta}_k \in \mathbb{R}^V$. Then, the word distributions of all topic entries (topic-word distribution matrix) is $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_k, ..., \boldsymbol{\beta}_K) \in \mathbb{R}^{V \times K}$. A document $\boldsymbol{x}$ is assigned with various topic entries with each probability as θ_k, so the distribution over all topic entries (topic distribution of $\boldsymbol{x}$) is $\boldsymbol{\theta} = (\theta_1, ..., \theta_k, ..., \theta_K)^{\mathrm{T}} \in \mathbb{R}^K$. Then, $\boldsymbol{x}$ is generated with its topic distribution $\boldsymbol{\theta}$ and the topic-word distribution matrix $\boldsymbol{\beta}$ of the whole corpus. The paths in Figure 2a can be specifically interpreted as follows:

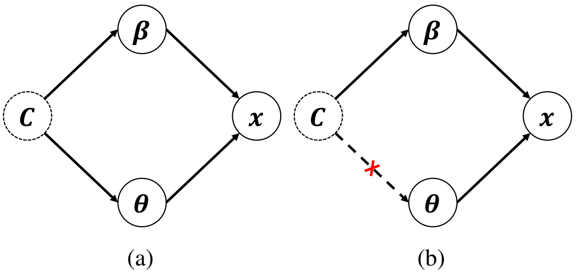

(a) (b)

Figure 2: The structural causal model of topic modeling. (a) Complete SCM without interventions. (b) Do intervention on the topic distribution $\boldsymbol{\theta}$.

- $\boldsymbol{C} \rightarrow \boldsymbol{\theta}$: This path says that the topic distributions are trained under bias. If there is no bias, different topic entries are ideally assigned to documents about various topics, and the inferred topic distributions of these documents are also different. However, in a long-tailed corpus with bias, the topic distributions of documents about different topics could be similar. As shown in Figure 1, since documents about the head topics are the absolute majority, most of the topic entries are assigned to them[2]. In this case, for a document about tail topics, its assigned topic entries probably are also assigned to the documents about head topics, as a result of which, its inferred topic distribution becomes similar to the topic distributions of some documents about head topics.

- $\boldsymbol{C} \rightarrow \boldsymbol{\beta} \rightarrow \boldsymbol{x}$: This link denotes the topic-word distribution matrix $\boldsymbol{\beta}$ is trained under the bias and is used to generate the document $\boldsymbol{x}$. Due to the long-tailed bias, the generated $\boldsymbol{x}$ tends to contain words in the documents about head topics.

- $\boldsymbol{\theta} \leftarrow \boldsymbol{C} \rightarrow \boldsymbol{\beta} \rightarrow \boldsymbol{x}$: Because of the confounder $\boldsymbol{C}$, the inferred topic distribution of a document about tail topics could be similar to the topic distributions of some documents about head topics, and the generated documents of these similar topic distributions tend to include words in the documents about head topics instead of tail topics. Therefore, this

---

[2] If the documents about tail topics are few enough to be ignorable, all the topic entries will be assigned to the documents about head topics.

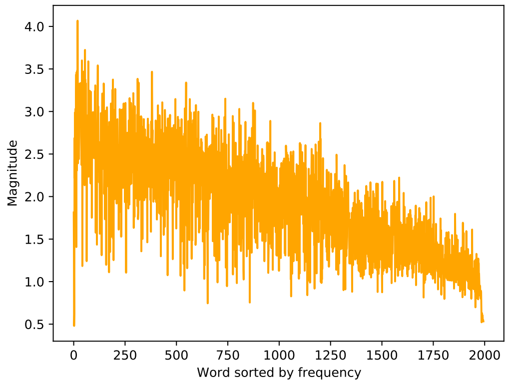

Figure 3: The magnitude of $\boldsymbol{\beta}_{i*}$ obtained from ProdLDA sorted by word frequency.

> backdoor path via $\boldsymbol{C}$ causes the spurious correlation between the topic distribution of a document about tail topics and the words in the documents about head topics.

In consequence, this spurious correlation through the confounder incurs that the discovered topics from documents about tail topics are mixed by the words of latent head topics. When the bias in the corpus gets severer, the discovered topics are even totally occupied by these words. Namely, topic models tend to ignore the semantics of the documents about tail topics and cannot discover the latent tail topics of a corpus.

In the above discussion, we clarify how the bias leads to the problems of long-tailed topic modeling with the presented SCM. In the next section, we propose a neat method to solve this issue without any auxiliary information.

3.2 Intervention on Topic Distribution

To remove the spurious correlation (deconfound), we propose to do causal intervention via *do*-operator (Pearl et al., 2016). Taking the chocolate and Nobel prizes for example again, intervening on the chocolate consumption means we fix its value through which we curtail the natural tendency of it to vary in response to the economy in nature. This amounts to remove the edges directed into the chocolate consumption. For example, if we were to close all chocolate factories, denoted as $do(\textbf{chocolate consumption} = 0)$, we will find the causality between the chocolate consumption and the number of Nobel prizes.

Similarly, we do intervention on the topic distribution $\boldsymbol{\theta}$ to compute the causality of $\boldsymbol{\theta}$ on $\boldsymbol{x}$, i.e., $p(\boldsymbol{x}|do(\boldsymbol{\theta}))$. As shown in Figure 2b, doing intervention on $\boldsymbol{\theta}$ means cutting off the edge $\boldsymbol{C} \rightarrow \boldsymbol{\theta}$ so that $\boldsymbol{C}$ cannot affect $\boldsymbol{\theta}$. But it is difficult to actually intervene variables (like closing all chocolate factories), so we utilize the backdoor adjustment (Pearl, 1995). The variable $\boldsymbol{\beta}$ meets the backdoor criterion and blocks the backdoor path $\boldsymbol{\theta} \leftarrow \boldsymbol{C} \rightarrow \boldsymbol{\beta} \rightarrow \boldsymbol{x}$. Following the backdoor adjustment, we use Inverse probability Weighting (Pearl et al., 2016) as

$$\begin{aligned} &p(\boldsymbol{x}|do(\boldsymbol{\theta})) \\ &\quad= \sum_{\boldsymbol{\beta}} p(\boldsymbol{x}|do(\boldsymbol{\theta}), \boldsymbol{\beta}) p(\boldsymbol{\beta}|do(\boldsymbol{\theta})) \quad (1) \\ &\quad= \sum_{\boldsymbol{\beta}} p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\beta}) p(\boldsymbol{\beta}) \quad (2) \\ &\quad= \sum_{\boldsymbol{\beta}} \frac{p(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\beta})}{p(\boldsymbol{\theta}|\boldsymbol{\beta})} \quad (3) \end{aligned}$$

In Figure 2b, all of $\boldsymbol{\theta}$ and $\boldsymbol{x}$ association flows along the directed path from $\boldsymbol{\theta}$ to $\boldsymbol{x}$ since there cannot be any backdoor paths because $\boldsymbol{\theta}$ has no incoming edges, so $p(\boldsymbol{x}|do(\boldsymbol{\theta}), \boldsymbol{\beta}) = p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\beta})$. Also, $p(\boldsymbol{\beta}|do(\boldsymbol{\theta})) = p(\boldsymbol{\beta}|\boldsymbol{\theta})$ since there's no other edges from $\boldsymbol{\theta}$ to $\boldsymbol{\beta}$ except through the collider $\boldsymbol{x}$.

But this equation is intractable, we need to approximate it. To find a proper way, we bury in mind that topics are interpreted as word distributions, so long-tail distributed topics can also be seen as long-tail distributed words. If we treat these words as "labels", then the generative process of a document is roughly predicting the probability under each "label". This inspires us to discover the relation between long-tailed topics and long-tailed classification tasks (Kang et al., 2020; Tang et al., 2020). Similar to these tasks, as shown in Figure 3, we observe that the magnitudes of topic distributions of words, i.e., $\boldsymbol{\beta}_{i*}$ for word i, gradually decrease along with the word frequency. Intuitively, the magnitude of $\boldsymbol{\beta}_{i*}$ means the "correlation score" between word i and all topics; therefore, this phenomenon may be because most inferred topics tend to relate to the words in documents about the head topics as mentioned before. Due to this finding, we propose an approximation method following the propensity score modeling (Rosenbaum and Rubin, 1983; Austin, 2011):

$$p(\boldsymbol{x}|do(\boldsymbol{\theta})) \approx \prod_{i} \frac{\boldsymbol{\beta}_{i*}\boldsymbol{\theta}}{\|\boldsymbol{\beta}_{i*}\|_2 \|\boldsymbol{\theta}\|_2} \quad (4)$$

where i refers to a word in $\boldsymbol{x}$ and we also empirically add the magnitude of $\boldsymbol{\theta}$. Here, the denominator works as a normalizer that balances the magnitude of the variables: $\boldsymbol{\beta}_{i*}$ and $\boldsymbol{\theta}$ for approximating the intervention probability.

3.3 Proposed Model

In this section, we propose a neural topic model for long-tailed corpora with deconfounded training based on the aforementioned intervention method, named as **Deconfounded Topic Model** (**DecTM**). Our network architecture is under the basic framework of VAE (Kingma and Welling, 2014; Rezende et al., 2014) with an encoder and a deconfounded decoder.

3.3.1 Encoder

The encoder transforms a text $\boldsymbol{x}$ into its topic distribution $\boldsymbol{\theta}$. Following the setting of Miao et al. (2016), we take the bag-of-words (BoW) assumption that ignores the word orders since topic models normally leverage word co-occurrences for topic inference. Inputted the BoW representation of $\boldsymbol{x}$, we first obtain its intermediate representation $\boldsymbol{\pi}$ with a Multi-Layer Perceptron (MLP). Based on $\boldsymbol{\pi}$, we then compute $q(\boldsymbol{r}|\boldsymbol{x})$, the variational distribution of the latent representation $\boldsymbol{r}$. Since the prior distribution $p(\boldsymbol{r})$ is assumed to be a Logistic Normal distribution for approximating the Dirichlet distribution (Srivastava and Sutton, 2017), we model the $q(\boldsymbol{r}|\boldsymbol{x})$ as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ are calculated as

$$\boldsymbol{\mu} = \boldsymbol{W}_{\mu}\boldsymbol{\pi} + \boldsymbol{b}_{\mu} \tag{5}$$

$$\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{W}_{\Sigma}\boldsymbol{\pi} + \boldsymbol{b}_{\Sigma}) \tag{6}$$

where $\boldsymbol{W}_{\mu}$, $\boldsymbol{W}_{\Sigma}$, $\boldsymbol{b}_{\mu}$ and $\boldsymbol{b}_{\Sigma}$ are weight matrices and biases respectively , and $\mathrm{diag}(\cdot)$ means converting a vector to a diagonal matrix. Later, to reduce the gradient variance, we adopt the reparameterization trick (Kingma and Welling, 2014) to sample $\boldsymbol{r}$ as

$$\boldsymbol{r} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{7}$$

Next, $\boldsymbol{r}$ is normalized with a softmax function to get the topic distribution $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta} = \mathrm{softmax}(\boldsymbol{r}). \tag{8}$$

3.3.2 Deconfounded Decoder

After getting the topic distribution of the input text, we then feed it to the proposed deconfounded decoder for reconstruction. According to the method in Equation (4), the objective function of DecTM can be written as

$$\mathcal{L}(\boldsymbol{x}) = \mathrm{KL}\left(q(\boldsymbol{r}|\boldsymbol{x}) \| p(\boldsymbol{r})\right) - \mathbb{E}_{\epsilon \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\sum_{i=1}^{N} \log \frac{\boldsymbol{\beta}_{i*}\boldsymbol{\theta}}{\|\boldsymbol{\beta}_{i*}\|_2 \|\boldsymbol{\theta}\|_2}\right] \tag{9}$$

where the first term is the Kullback-Leibler (KL) divergence between the posterior and prior distribution. It can be computed with the analytical solution for two Normal distributions. The second term is the reconstruction error between the input and output text. Different from normal neural topic models (Miao et al., 2016; Srivastava and Sutton, 2017), the deconfounded decoder in our model employs the approximated probability for causal intervention on $\boldsymbol{\theta}$ to weaken the long-tailed bias. Note that our model can be directly applied to naturally collected corpora since no additional auxiliary information is necessary for our model.

4 Experiment Setup

4.1 Datasets

Unfortunately, common datasets for topic modeling are almost all balanced, so we manually construct the long-tailed variants of them by repeating and deleting documents according to the given labels, making them follow a long-tailed distribution. Through the distribution of labels, we can roughly assume the latent topics are long-tailed distributed. In this way, we form the long-tailed versions (-LT) of 20News (Lang, 1995)[3] and Yahoo Answer[4], called 20News-LT and Yahoo Answer-LT respectively. Moreover, to better evaluate the performance of long-tailed topic modeling, we adopt the datasets for eXtreme Multi-label Text Classification (XMTC) (You et al., 2019), a task to predict the most relevant multiple labels for texts from an extremely large-scale label set. The label set includes hundreds and thousands, even millions of labels, and most are tail labels with very few positive samples. These plentiful labels can be naturally interpreted as the latent topics of documents; thus, we can evaluate the proposed model on these long-tailed distributed datasets. We conducted experiments on the the subsets of standard benchmark XMTC datasets Amazon-670K, Wiki-500K, AmazonCat-13K and Amazon-3M (Bhatia

---

[3] `http://qwone.com/~jason/20Newsgroups`

[4] `https://answers.yahoo.com`

Datasets	#docs	Average length	#labels	Vocabulary size	Average #labels per doc	Average #docs per label
20News-LT	11,314	73.5	20	1,984	-	-
Yahoo Answer-LT	99,806	31.0	10	4,738	-	-
Wiki-500K	100,000	504.0	175,206	5,000	6.0	3.4
Amazon-670K	100,000	90.0	333,863	5,000	2.0	1.6
AmazonCat-13K	200,000	78.8	11,096	5,000	5.0	91.0
Amazon-3M	300,000	40.7	1,917,999	5,000	13.0	5.6

Table 1: Statistics of datasets.

et al., 2016)[5].

For all datasets, we conduct the following steps for preprocessing: (1) tokenize texts and lowercase words; (2) remove stop words and illegal characters; (3) remove low-frequency words. The statistics of preprocessed datasets are reported in Table 1. It is worth noting that although labels are provided in these datasets, they are not used by our model.

4.2 Baseline Models

We take both probabilistic and neural topic models as baselines. For probabilistic models, we consider the widely used LDA (Blei et al., 2003) with python-lda[6] package for topic inference. For neural topic models, we use NVDM (Miao et al., 2016)[7], ProdLDA (Srivastava and Sutton, 2017)[8] and Scholar (Card et al., 2018)[9]. Scholar is an extension of ProdLDA via optionally incorporating metadata of documents like sentiments.

5 Experiment Results

5.1 Topic Quality Analysis

5.1.1 Evaluation Metrics

Following Nan et al. (2019) and Wu et al. (2020b), we evaluate the topic quality concerning two aspects, topic coherence and diversity. Topic coherence means that the words in the discovered topics are supposed to be as coherent as possible instead of irrelevant ones, and topic diversity means that topics should differ from each other instead of being similar ones.

Topic Coherence For topic coherence, we employ C_V (Röder et al., 2015), an improved variant of the Normalized Pointwise Mutual Information (NPMI) (Bouma, 2009; Chang et al., 2009; Newman et al., 2010). Its detailed calculation can be found in Wu et al. (2020b). We need to mention that given a topic z and its top T probable words $(x_1, x_2, ..., x_T)$, the NPMI of (x_i, x_j) used in the C_V computation is defined as

$$\text{NPMI}(x_i, x_j) = \frac{\log \frac{p(x_i,x_j)+\epsilon}{p(x_i)p(x_j)}}{-\log(p(x_i, x_j) + \epsilon)} \qquad (10)$$

where $p(x_i)$ is the occurrence probability of word x_i and $p(x_i, x_j)$ the co-occurrence probability of (x_i, x_j). These probabilities are estimated in a reference corpus. To exhaustively assess the topic coherence performance of long-tailed topic modeling, we use three kinds of C_V scores with the probabilities estimated in different reference corpora. First, we adopt the public tool[10] which uses Wikipedia documents as the external reference corpus (-E), so it is named as C_V-E. Then, we directly use the internal training documents (-I) as the reference corpus, named as C_V-I. However, since documents about head topics occupy the main portion of a long-tailed corpus, previous C_V-E and C_V-I probably are insufficient to appraise the performance on the documents about tail topics. To this end, we heuristically introduce C_V-T that employs the documents including the tail labels (-T) provided by the datasets instead of all the training documents as the reference corpus, so it can assess whether the discovered topics can reveal the hidden semantics of documents about tail topics, i.e., discover the tail topics.

Topic Diversity For topic diversity evaluation, we employ the Topic Unique (TU) (Nan et al.,

---

[5] `http://manikvarma.org/downloads/XC/XMLRepository.html`

[6] `https://github.com/lda-project/lda`

[7] `https://github.com/ysmiao/nvdm`

[8] `https://github.com/akashgit/autoencoding_vi_for_topic_models`

[9] `https://github.com/dallascard/scholar`

[10] `https://github.com/dice-group/Palmetto`

Models	20News-LT					Yahoo Answer-LT					Wiki-500K				
	TU	C_V-E	C_V-I	C_V-T	TQ	TU	C_V-E	C_V-I	C_V-T	TQ	TU	C_V-E	C_V-I	C_V-T	TQ
LDA	0.469	0.318	**0.712**	0.378	0.220	0.531	0.335	0.408	0.384	0.199	0.759	0.426	0.680	0.669	0.449
NVDM	0.849	0.315	0.367	0.492	0.332	0.934	0.352	0.440	0.549	0.417	0.836	0.397	0.429	0.372	0.334
ProdLDA	0.743	0.321	0.521	0.543	0.343	0.830	**0.383**	0.434	0.516	0.369	0.889	**0.438**	0.698	0.653	0.530
Scholar	0.787	0.321	0.537	0.528	0.364	0.876	0.380	0.445	0.519	0.392	0.874	0.437	0.682	0.633	0.510
DecTM	**0.937**	**0.324**	0.543	**0.554**	**0.443**	**0.948**	0.381	**0.516**	**0.573**	**0.464**	**0.979**	0.437	**0.754**	**0.716**	**0.622**
Models	Amazon-670K					AmazonCat-13K					Amazon-3M				
	TU	C_V-E	C_V-I	C_V-T	TQ	TU	C_V-E	C_V-I	C_V-T	TQ	TU	C_V-E	C_V-I	C_V-T	TQ
LDA	0.752	0.387	0.601	0.566	0.390	0.722	0.385	0.616	0.585	0.382	0.746	0.368	0.692	**0.624**	0.418
NVDM	0.873	0.391	0.338	0.366	0.319	0.881	0.403	0.392	0.405	0.352	0.889	0.433	0.425	0.487	0.399
ProdLDA	0.869	0.424	0.606	0.501	0.443	0.893	**0.430**	0.641	0.531	0.477	0.925	0.442	0.636	0.498	0.486
Scholar	0.874	**0.427**	0.607	0.502	0.449	0.886	0.428	0.644	0.540	0.476	0.908	**0.445**	0.614	0.508	0.474
DecTM	**0.987**	0.404	**0.672**	**0.571**	**0.542**	**0.991**	0.406	**0.702**	**0.590**	**0.561**	**0.991**	0.405	**0.701**	0.530	**0.541**

Table 2: Topic quality results concerning topic coherence and diversity. The best in each column is in bold.

2019) defined as

$$TU(z) = \frac{1}{T}\sum_{i=1}^{T}\frac{1}{\text{cnt}(x_i)} \tag{11}$$

where $\text{cnt}(x_i)$ is the total number of times that word x_i appears in the top T words of all topics. Accordingly, TU ranges from $1/K$ to 1, and a higher TU score means topics are more diverse since fewer words are repeated across all.

Comprehensive Evaluation It is necessary to mention that if the topic coherence performance of a model remains about the same and the diversity gets higher, it means the overall topic quality is also improved since it can unearth more various semantics of documents. To provide a forthright and comprehensive evaluation of both coherence and diversity performance, following Dieng et al. (2019), we propose Topic Quality (TQ) that combines C_V and TU as

$$TQ = TU \times \frac{1}{3}(C_V\text{-E} + C_V\text{-I} + C_V\text{-T}) \tag{12}$$

which is the product of TU and the average of three different C_V scores. Thus, TQ can provide a direct comparison of the overall topic quality performance.

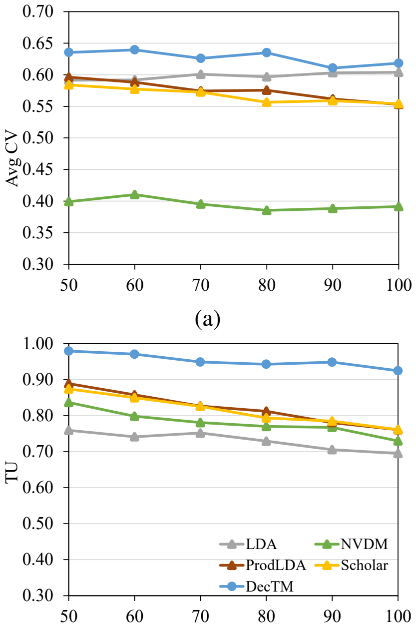

(a)

(b)

Figure 4: TU and average C_V scores on Wiki-500K under different topic numbers.

5.1.2 Results Analysis

Table 2 reports the topic quality results concerning different metrics of the top 15 words with the topic number $K = 50$. At first, we notice that C_V-E scores of DecTM are the highest on 20News-LT and are very close to the best on Yahoo Answer-LT and Wiki-500K. Although C_V-E scores of some baseline models are higher on other datasets, DecTM stably outperforms them in terms of TU by a large margin, and the C_V-I scores of DecTM are also mostly better. This implies that baseline models are disposed to generate repetitive topics because of the bias of these long-tailed corpora, while the topics of our DecTM are more diverse. Therefore, despite that C_V-E of some baselines are higher, their lower diversity performance indicates that their yielded topics are redundant. More significantly, DecTM commonly surpasses baseline models in terms of C_V-T, which shows the

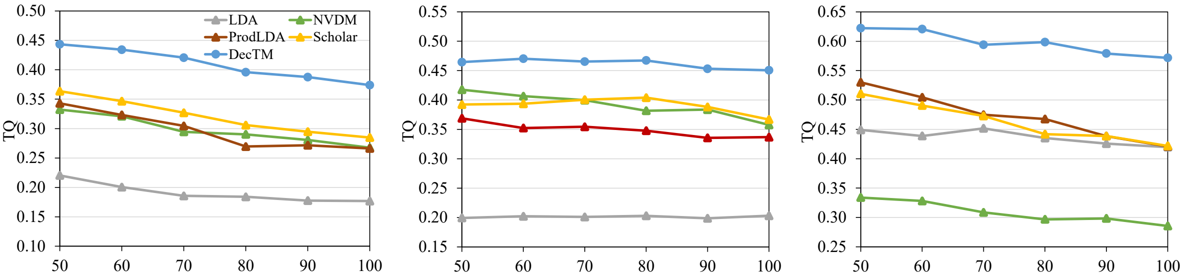

(a) 20News-LT

(b) Yahoo Answer-LT

(c) Wiki-500K

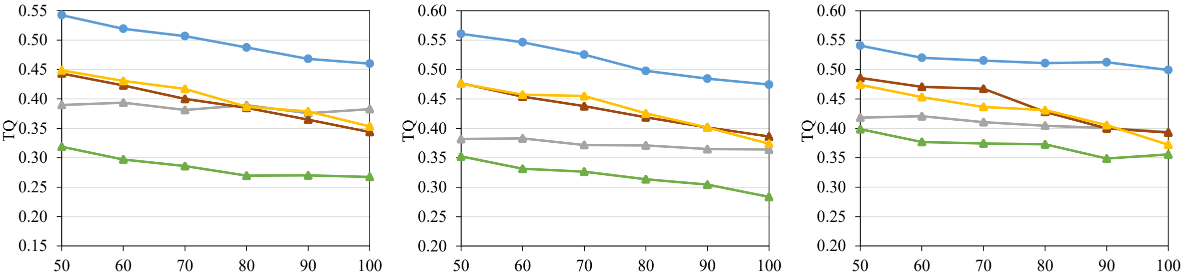

(d) Amazon-670K

(e) AmazonCat-13K

(f) Amazon-3M

Figure 5: Topic quality performance (TQ) under different topic numbers.

discovered topics can preferably reflect the semantics of documents about tail topics; thus, the produced topics of DecTM are more complete. These arguments are further illustrated with topic examples in Section 5.3. At last, we find that DecTM achieves the highest TQ scores on all datasets, showing the overall performance of our model is fairly better.

From the above experimental results, we observe the problems of the long-tailed topic modeling formerly mentioned in Section 1 and Section 3.1, that the performance of existing topic models, especially topic diversity, is deteriorated on account of the bias. But with the help of the deconfounded decoder, our proposed method can alleviate the effect of the bias and hugely improve the topic diversity while remain good coherence performance with a better ability to expose the semantics of documents about tail topics.

5.2 Impact of Topic Number

To investigate how performance varies concerning the topic number, we report the TU and the average C_V (Avg C_V) scores defined in Equation (12) under topic number K ranging from 50 to 100 on Wiki-500K in Figure 4. We can see that the Avg C_V of DecTM is relatively better in Figure 4a. Besides, as shown in Figure 4b, TU scores of all models gradually decline when the topic number gets bigger, but the performance of DecTM is constantly higher and decreases slower. The reason is that those baseline models tend to focus on documents about head topics which are inadequate to support larger topic numbers, while DecTM can also discover semantics of documents about tail topics. Furthermore, Figure 5 presents the TQ with different topic numbers of all datasets. We can observe that whether on manually constructed or XMTC datasets, our model DecTM outperforms baseline models under different topic numbers. These experiments demonstrate that the performance of our model is relatively stable.

5.3 Discovered Topic Examples

To further illustrate the topic quality performance of different models, Table 3 reports some discovered topic examples. As shown by the comparison of topic diversity in Section 5.1.2, we can see baseline models produce some topics including repeated words. To be more specific, LDA generates topics with repetitive words like “subject”, “organization” from 20News-LT and “book”, “author” from Amazon-3M. Similar topics about “newcastle”, “orchestra” and “hockey” are yielded by NVDM, ProdLDA, and Scholar respectively. We also notice that NVDM, ProdLDA and Scholar all yield several same topics about “census” from Wiki-500K. These topics are coherent indeed, but they can trickily improve the C_V scores and are redundant in the downstream applications. On the contrary, we find only one coherent topic generated by DecTM corresponding to the aforementioned ones.

Models	Topic examples
LDA	hp nasa organization subject article new re
	subject organization access good pc support
	subject problem organization file world get
	scsi organization subject mark university ide
	thanks organization subject pt scott imagine
	life book god church author christian spiritual
	book students guide text chapter reading
	book story love read author stories characters
	books author lives writing new years book
	guide book design new using use techniques
NVDM	galaxy texas newcastle sky austin theta madrid
	edinburgh harbour newcastle fortress tunnel
	edwards leeds birmingham newcastle
	townships cdp islander nonfamilies
	couples females males husband nonfamilies
ProdLDA	orchestra hits songs symphony unreleased song
	concert orchestra opera biography symphony
	translation symphony subtitles orchestra mozart
	median capita nonfamilies residing household
	nonfamilies households householder residing
	residing township householder nonfamilies
	quot bmw yamaha honda macbook laptop
Scholar	guitarist pianist composer hockey player
	montreal nhl provincial provinces hockey
	championships finals medal olympics hockey
	householder nonfamilies households residing
	nonfamilies residing households householder
	township norway nonfamilies residing
	demographics median census hamlet town
DecTM	median residing nonfamilies household
	tires tire steering truck motorcycle honda
	wales welsh yorkshire scotland glasgow
	violin orchestra symphony concerto piano
	nonfiction copies manga novels bestseller
	championships mens olympic competed
	ink inkjet paper printer printers cartridges
	episodes episode season vol inspector series
	europe russian paris germany german spain

Table 3: Topic examples. Repeated words are underlined.

What is more, DecTM also discovers some latent topics like “printer”, “series” and “europe” while these cannot be found by baseline models, which could verify the superior C_V-T performance of our model. These topic examples qualitatively show the overall topic quality performance of DecTM is adequately preferable.

6 Conclusion

In this paper, for discovering the topics in long-tailed corpora, we present a causal inference model to describe how the bias influences topic modeling, and to reduce the impact of the bias, we then propose a causal intervention method for deconfounding, relying on which we introduce the Deconfounded Topic Model (DecTM) with a deconfounded decoder. Comprehensive experiments demonstrate that our model can produce topics with better quality and mitigate the effect of long-tailed bias.

Acknowledgement

We want to thank all anonymous reviewers for their helpful comments. This work is supported by China NSFC under Grant 61672309 and MOST Fundamental Research Project under Grant 2017FY201407.

References

Peter C Austin. 2011. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Dallas Card, Chenhao Tan, and Noah A Smith. 2018. Neural Models for Documents with Metadata. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2031–2040.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277.

Fabian Dablander. 2020. An introduction to causal inference. *PsyArXiv*.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. 2019. The dynamic embedded topic model. *arXiv preprint arXiv:1907.05545*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. 2020. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*.

Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. 2012. Topic-driven reader comments summarization. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 265–274. ACM.

David P MacKinnon, Amanda J Fairchild, and Matthew S Fritz. 2007. Mediation analysis. *Annu. Rev. Psychol.*, 58:593–614.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Jon Mcauliffe and David Blei. 2007. Supervised topic models. *Advances in neural information processing systems*, 20:121–128.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736.

Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. 2019. Topic modeling with Wasserstein autoencoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6345–6381, Florence, Italy. Association for Computational Linguistics.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics.

Judea Pearl. 1995. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. 2016. *Causal inference in statistics: A primer*. John Wiley & Sons.

Judea Pearl and Dana Mackenzie. 2018. *The book of why: the new science of cause and effect*. Basic books.

William J Reed. 2001. The pareto, zipf and other power laws. *Economics letters*, 74(1):15–19.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *In Proceedings ofthe 31th International Conference on Machine Learning*.

Lorenzo Richiardi, Rino Bellocco, and Daniela Zugna. 2013. Mediation analysis in epidemiology: methods, interpretation and bias. *International journal of epidemiology*, 42(5):1511–1519.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408. ACM.

Paul R Rosenbaum and Donald B Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *ICLR*.

Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. 2020. Long-tailed classification by keeping the good and removing the bad momentum causal effect. In *NeurIPS*.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 697–702. IEEE.

Yi Wang, Xuemin Zhao, Zhenlong Sun, Hao Yan, Lifeng Wang, Zhihui Jin, Liubin Wang, Yang Gao, Ching Law, and Jia Zeng. 2015. Peacock: Learning long-tail topic features for industrial applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):1–23.

Xiaobao Wu and Chunping Li. 2019. Short Text Topic Modeling with Flexible Word Patterns. In *International Joint Conference on Neural Networks*.

Xiaobao Wu, Chunping Li, Yan Zhu, and Yishu Miao. 2020a. Learning Multilingual Topics with Neural Variational Inference. In *International Conference on Natural Language Processing and Chinese Computing*.

Xiaobao Wu, Chunping Li, Yan Zhu, and Yishu Miao. 2020b. Short text topic modeling with topic distribution quantization and negative sampling decoder. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1772–1782, Online.

Yiquan Wu, Kun Kuang, Yating Zhang, Xiaozhong Liu, Changlong Sun, Jun Xiao, Yueting Zhuang, Luo Si, and Fei Wu. 2020c. De-biased court's view generation with causality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 763–780.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456. ACM.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *Advances in Neural Information Processing Systems*, pages 5820–5830.

Xiangji Zeng, Yunliang Li, Yuchen Zhai, and Yin Zhang. 2020. Counterfactual generator: A weakly-supervised method for named entity recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7270–7280.

Ziqi Zhang and Lei Luo. 2019. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, 10(5):925–945.

Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. 2020. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9719–9728.

More than just Frequency?
Demasking Unsupervised Hypernymy Prediction Methods

Thomas Bott, Dominik Schlechtweg and Sabine Schulte im Walde
Institute for Natural Language Processing, University of Stuttgart, Germany
`{bottts,schlecdk,schulte}@ims.uni-stuttgart.de`

Abstract

This paper presents a comparison of unsupervised methods of hypernymy prediction (i.e., to predict which word in a pair of words such as *fish–cod* is the hypernym and which the hyponym). Most importantly, we demonstrate across datasets for English and for German that the predictions of three methods (WeedsPrec, invCL, SLQS Row) strongly overlap and are highly correlated with frequency-based predictions. In contrast, the second-order method SLQS shows an overall lower accuracy but makes correct predictions where the others go wrong. Our study once more confirms the general need to check the frequency bias of a computational method in order to identify frequency-(un)related effects.

1 Introduction

Hypernymy represents a major paradigmatic semantic relation between two concepts, a hypernym (superordinate) and a hyponym (subordinate), as in *tree–oak* and *fish–cod*, where the hyponym implies the hypernym, but not vice versa. From a cognitive perspective hypernymy is central to the organisation of the mental lexicon (Deese, 1965; Miller and Fellbaum, 1991; Murphy, 2003), next to further semantic relations such as synonymy, antonymy, etc. From a computational perspective hypernymy is central to solving a number of Natural Language Processing (NLP) tasks such as taxonomy creation (Hearst, 1998; Cimiano et al., 2004; Snow et al., 2006; Navigli and Ponzetto, 2012), textual entailment (Dagan et al., 2006; Clark et al., 2007) and text generation (Biran and McKeown, 2013).

Accordingly, the field has witnessed active research on two subtasks involved in computational models of hypernymy (see Shwartz et al. (2017) for an extensive overview): *hypernymy detection* (i.e., distinguishing hypernymy from other semantic relations) and *hypernymy prediction* (i.e., determining which word in a pair of words is the hypernym and which is the hyponym). The target subtask of the current study is hypernymy prediction: we perform a comparative analysis of a class of approaches commonly refered to as *unsupervised hypernymy methods* (Weeds et al., 2004; Kotlerman et al., 2010; Clarke, 2012; Lenci and Benotto, 2012; Santus et al., 2014). These methods all rely on the distributional hypothesis (Harris, 1954; Firth, 1957) that words which are similar in meaning also occur in similar linguistic distributions. In this vein, they exploit asymmetries in distributional vector space representations, in order to contrast hypernym and hyponym vectors.

While these unsupervised hypernymy prediction methods have been explored and compared extensively on a number of benchmark datasets (Shwartz et al., 2017), this study takes a novel perspective and performs a detailed analysis of whether and where the methods make similar or different decisions. Our prediction experiments on simplex and complex nouns in English and German WordNets and evaluation benchmarks show that most of the methods we investigate overlap in their specific predictions to a surprisingly high degree, and that the predictions strongly correlate with those based on raw frequencies. Our study therefore emphasises the general need to check the frequency bias of a computational method and to distinguish between frequency-related and frequency-unrelated effects.

2 Data and Methods

In the following we describe our gold standard datasets (Section 2.1), our corpora and vector spaces (Section 2.2) and our hypernymy prediction methods (Section 2.3). The code and links to the gold standards are available from `https://github.com/Thommy96/hyp-freq-comp`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 186–192
August 1–6, 2021.

2.1 Gold Standard Datasets

Our study focuses on hypernymy between nouns and uses two types of gold standard resources for hypernymy relations. On the one hand, we rely on WordNets as classical large-scale taxonomies where hypernymy represents one of the core semantic relations for organisation: the English *WordNet*[1] (Miller et al., 1990; Fellbaum, 1998), version 3, and the German *GermaNet*[2] (Hamp and Feldweg, 1997; Kunze and Wagner, 1999; Lemnitzer and Kunze, 2007), version 11. From both WordNets, we extracted all noun–noun pairs with a hypernymy relation and removed duplicates, autohyponyms and space-separated multiword expressions. We also distinguish between compounds (which frequently represent hyponyms of their constituent heads, as in *dog–lapdog*) and non-compounds by applying a simple heuristic, i.e., categorising all hypernym–hyponym pairs as compounds if one is a substring of the other. We expected this subset to exhibit idiosyncratic behaviour in our prediction experiments.

On the other hand, we rely on a number of benchmark datasets for hypernymy evaluation: *BLESS* (Baroni and Lenci, 2011) provides related concepts for 200 English concrete nouns connected through a semantic relation (hypernymy, co-hyponymy, meronymy, attribute, event) or a null-relation. The dataset by Lenci and Benotto (2012) contains a subset of BLESS relation pairs, as created for previous comparisons of hypernymy detection methods. A dataset similar to BLESS, *EVALution*, was induced from ConceptNet and WordNet (Santus et al., 2015). Its semantic relations include hypernymy, synonymy, antonymy and meronymy. For quality reasons, the pairs were filtered by automatic methods and crowd-sourcing to improve consistency and to determine prototypical pairs. Finally, we use the Weeds dataset (Weeds et al., 2004; Weeds and Weir, 2005) which contains word pairs related by hypernymy and co-hyponymy across word classes. From all four benchmark datasets we extracted all noun–noun pairs related by hypernymy.

The first row in Table 1 shows the numbers of hypernymy pairs in the WordNets and in the benchmark datasets.

2.2 Corpora and Vector Spaces

We created our distributional vector spaces based on the *WaCky*[3] corpora (Baroni et al., 2009) for English and for German. The English *PukWaC* corpus is the syntax-annotated version of *ukWaC* (Ferraresi et al., 2008) and contains ≈1.9 billion words; the German *SdeWaC* corpus (Faaß and Eckart, 2013) is a cleaned version of the *WaCky* corpus *deWaC* and contains ≈880 million words; both corpora are pos-tagged with the *TreeTagger* (Schmid, 1994).

For each corpus we created a traditional count vector space[4] based on a co-occurrence window of ± 10 words within sentences (because sentences in the SdeWaC are shuffled, so going beyond sentence border is meaningless). We used a bag-of-words approach only taking into account lemmatised nouns, verbs and adjectives.

2.3 Hypernymy Methods and Baselines

We selected four unsupervised hypernymy methods and defined two baselines. The methods were chosen from different families with regard to how they exploit the distributional hypothesis for hypernymy detection: *WeedsPrec* and *InvCL* rely on the *Distributional Inclusion Hypothesis*, according to which a significant number of distributional features of a word x is included in the distributional features of a word y, if x is semantically more specific than y. *SLQS Row* and *SLQS Sec*[5] rely on the *Distributional Informativeness Hypothesis* using first- and second-order variants of word entropy, respectively. The methods are defined as follows regarding the distributional features f in the two word vectors $\vec{x}$ and $\vec{y}$ for a word pair $\langle x, y \rangle$.

WeedsPrec: An asymmetric precision method suggested by Weeds et al. (2004) that quantifies the weighted inclusion of the features of word x in the features of word y. If $WeedsPrec(x, y) > WeedsPrec(y, x)$, then x is predicted as the hyponym and y as the hypernym, and vice versa.

$$WeedsPrec(x, y) = \frac{\sum_{f \in (\vec{x} \cap \vec{y})} x_f}{\sum_{f \in \vec{x}} x_f}$$

---

[1] https://wordnet.princeton.edu

[2] https://uni-tuebingen.de/en/142806

[3] http://wacky.sslmit.unibo.it/

[4] Note that not all of the selected methods are applicable to embeddings, and it also not our goal to identify the optimal vector spaces, rather than analysing their predictions; this is why our analyses rely on standard count dimensions.

[5] Originally, this method is called SLQS, but to distinguish it from SLQS Row we refer to it as SLQS Sec.

	WordNet		GermaNet		BLESS	EVALution	LB	Weeds
	¬comp	comp	¬comp	comp				
sizes:	106,397	3,366	102,714	35,963	1,337	606	1,747	1,117
Word Length	47.26	**94.65**	56.14	**99.41**	23.19	34.86	52.42	44.76
Word Frequency	**73.19**	92.81	73.66	98.78	62.30	**68.96**	76.48	**76.63**
WeedsPrec	72.22	92.93	74.01	98.87	57.52	64.22	**77.02**	74.22
InvCL	72.97	92.84	73.92	98.78	63.05	68.86	76.48	76.45
SLQS Row	71.82	93.02	**74.40**	98.79	58.56	55.91	71.27	72.43
SLQS Sec	65.05	74.66	70.38	90.15	**71.80**	59.63	62.66	65.71

Table 1: Sizes of datasets and overall prediction results across datasets.

InvCL: An asymmetric precision method suggested by Lenci and Benotto (2012) that takes both feature inclusion as well as feature non-inclusion (originally suggested as *ClarkDE (cde)* by Clarke (2012)) into account. If $invCL(x, y) > invCL(y, x)$, then x is predicted as the hyponym and y as the hypernym, and vice versa.

$$cde(x, y) = \frac{\sum_{f \in (\vec{x} \cap \vec{y})} min(x_f, y_f)}{\sum_{f \in \vec{x}} x_f}$$

$$invCL(x, y) = \sqrt{cde(x, y) \cdot (1 - cde(y, x))}$$

SLQS Row: An asymmetric method suggested by Shwartz et al. (2016) which relies on the word entropy $H(w)$ for a word w, taking all context words as features into account: w_f. If $SLQS_{Row}(x, y) > 0$, then x is predicted as the hyponym and y as the hypernym, and vice versa.

$$SLQS_{Row}(x, y) = 1 - \frac{H(x)}{H(y)}$$

$$H(w) = -\sum_{w_f} p(w_f|w) \cdot log_2(p(w_f|w))$$

SLQS Sec: An asymmetric method suggested by Santus et al. (2014) which relies on second-order word entropy $E(w)$ and is calculated as the median entropy Med of a word's most strongly associated context words w_f. We use the 50 strongest contexts in our vector spaces, as determined by weighted co-occurrence scores using *positive local mutual information* (Evert, 2005). If $SLQS_{Sec}(x, y) > 0$, then x is predicted as the hyponym and y as the hypernym, and vice versa.

$$SLQS_{Sec}(x, y) = 1 - \frac{E(x)}{E(y)}$$

$$E(w) = Med_{w_f}\ H(w_f)$$

Baselines: In comparison to the hypernymy methods we applied two baselines, cf. Zipf's principles of least effort (Zipf, 1949):

- ***Word Length***: Given that hyponyms refer to more specific concepts than their hypernyms, and assuming that more specific concepts tend to have a longer word length, this baseline predicts the longer word in a word pair (as measured by the number of characters) as the hyponym.
- ***Word Frequency***: Given that hyponyms refer to more specific concepts than their hypernyms and assuming that more specific concepts appear less often in a corpus, this baseline predicts the less frequent word in a word pair (as measured by corpus frequency) as the hyponym.

3 Prediction Results and Comparisons

3.1 Overall results

Table 1 shows the overall accuracy results of the predictions across methods and datasets (best results in bold fonts). Accuracy is defined by the proportion of correct predictions given that we know which word in a word pair is the hypernym and which is the hyponym.

For each WordNet we list two results, one for the non-compound pairs (in blue, as the benchmark results) and one for the compound pairs (in grey). For compound pairs word length is an almost perfect predictor,[6] as expected, and all unsupervised methods are also above 90%, with SLQS Sec as an exception. In all other columns we can see that word length is generally a poor baseline. Word frequency, however, is a very powerful baseline; across datasets it keeps up or even outperforms the respective best methods, which are SLQS Sec

---

[6] The prediction does not reach 100% because our heuristic included non-compound pairs, such as *selection–election*.

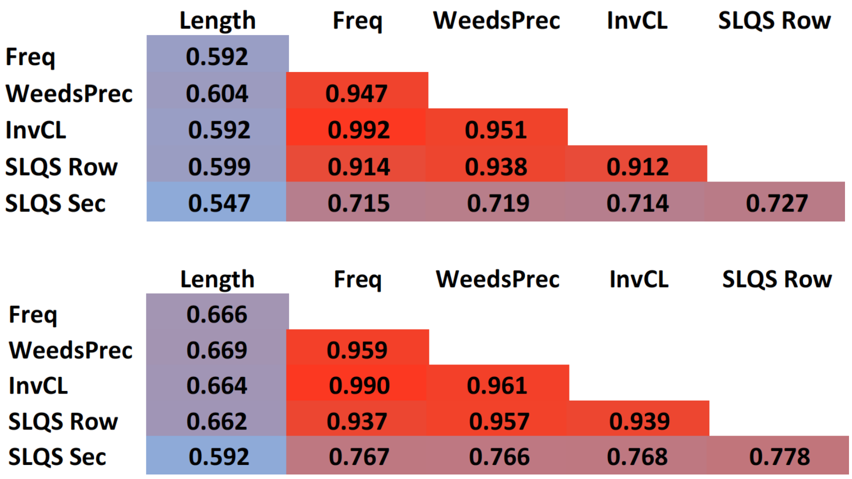

	Length	Freq	WeedsPrec	InvCL	SLQS Row
Freq	0.592				
WeedsPrec	0.604	0.947			
InvCL	0.592	0.992	0.951		
SLQS Row	0.599	0.914	0.938	0.912	
SLQS Sec	0.547	0.715	0.719	0.714	0.727

	Length	Freq	WeedsPrec	InvCL	SLQS Row
Freq	0.666				
WeedsPrec	0.669	0.959			
InvCL	0.664	0.990	0.961		
SLQS Row	0.662	0.937	0.957	0.939	
SLQS Sec	0.592	0.767	0.766	0.768	0.778

Figure 1: SMC correlations between methods for WordNet (above) and GermaNet (below) non-compound pairs.

on BLESS; InvCL on EVALution and Weeds; and WeedsPrec on LB. Across datasets, the best results vary between 68.96% and 77.02% for non-compounds; compounds obviously represent “easy” cases of hypernymy.

3.2 Correlations between predictions

To explore similarities in predictions across methods, we applied the *Simple Matching Coefficient (SMC)* (Sokal, 1958) to determine for each two methods to which degree their decisions overlap, by comparing the number of matching decisions (i.e., where both methods predicted the same noun in a noun pair as the hypernym) against the number of decisions (i.e., the total number of noun pairs). The heatmaps in Figure 1 show the results for the non-compounds in the English WordNet (left) and in GermaNet (right). They clearly demonstrate that word length makes very different decisions to word frequency and the unsupervised methods, and that word frequency and all unsupervised methods but SLQS Sec highly correlate in their predictions.

3.3 Role of frequency

We go one step further to explore the role of frequency. Figure 2 presents the prediction results on 10 equally-sized subsets of the non-compound pairs in the WordNets after the target pairs were sorted by decreasing difference in hypernym corpus frequency minus hyponym corpus frequency. I.e., in the left-most subset on the x-axis we see the results on the subset with largest differences in hypernym–hyponym frequencies.

We can clearly see that up to subset 7 (up to which the hyponym frequencies are all below the hypernym frequencies), decisions based on word frequency, WeedsPrec, invCL and SLQS Row predict the hypernym almost perfectly; for subset 8 (where the hyponym frequencies start to become larger than the hypernym frequencies) their predictions are becoming worse; and for subsets 9–10 the predictions are mostly wrong. Results by relying on word length and SLQS Sec are clearly worse for the first seven subsets but also better for the last two subsets, thus confirming that they make different predictions.

3.4 Correctness of predictions

While SMC in Section 3.2 informed us about overlap in decisions, it did not tell us whether one of the methods is qualitatively superior, so we analysed whether some methods are simply worse than others, according to their lower accuracy in prediction, or whether the methods all have their own strengths. We calculated for each pair of methods which proportion of wrongly predicted pairs of one method was predicted correctly by the other method. Figure 3 illustrates for the English WordNet how many pairs wrongly predicted by word frequency are predicted correctly by another method (see x-axis).

As we can see, while word length and SLQS Sec are often worse in performance than frequency, they still manage to make correct predictions when frequency fails, which is much less the case for the frequency-alike methods WeedsPrec, invCL and SLQS Row. In particular, invCL seems to make almost identical predictions as frequency, which was already indicated by their almost perfectly overlapping lines in Figure 2.

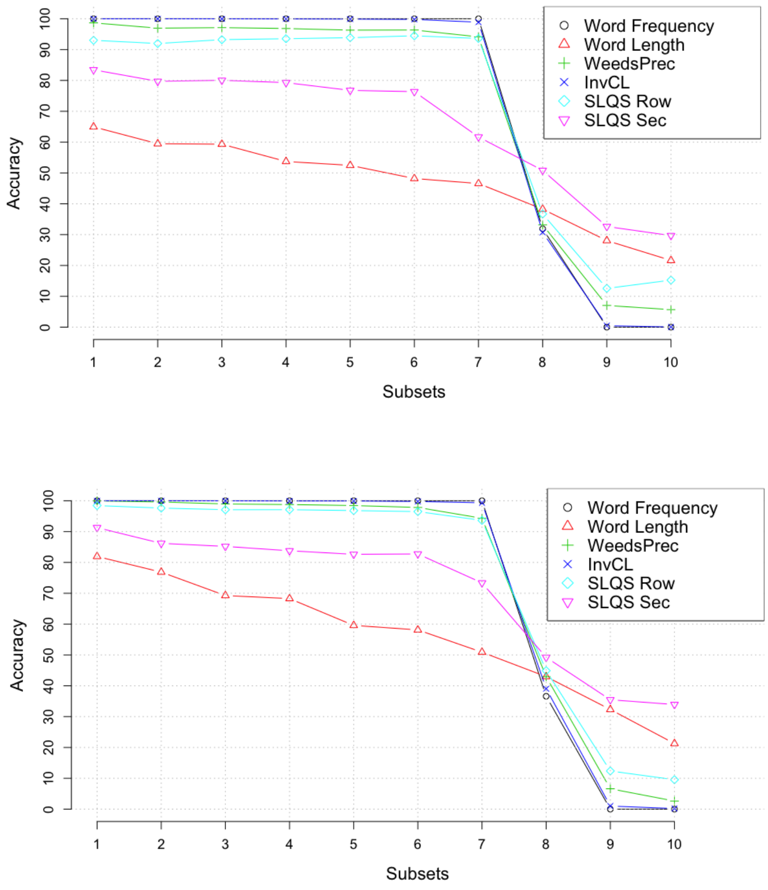

Figure 2: Prediction results for WordNet (above) and GermaNet (below) non-compound pairs across equally-sized subsets of target pairs sorted by difference in hypernym frequency and hyponym frequency.

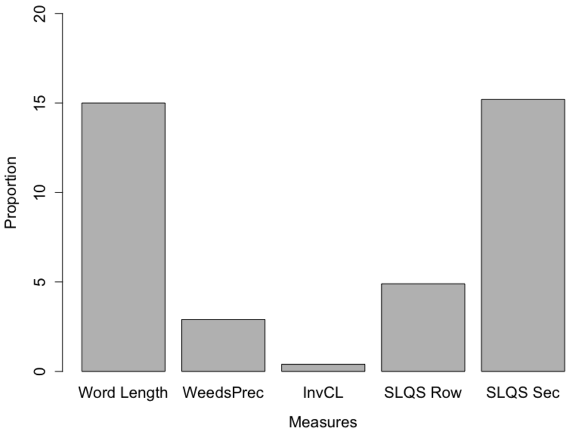

Figure 3: Proportions of pairs predicted wrongly by word frequency but correctly by the given method.

4 Conclusion

This study performed a series of hypernymy predictions by unsupervised methods. We demonstrated that across datasets for English and for German the predictions of three methods (WeedsPrec, invCL and SLQS Row) are highly correlated and also mostly identical with frequency-based predictions. In contrast, word length and SLQS Sec show an overall lower accuracy but at the same time make correct predictions where the others go wrong. Our study once more confirms the general need to check the frequency bias of a computational method in order to identify frequency-(un)related effects.

Acknowledgements

Dominik Schlechtweg was supported by the Konrad Adenauer Foundation and the CRETA center funded by the German Ministry for Education and Research during the conduct of this research.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed Distributional Semantic Evaluation. In *Proceedings of the EMNLP Workshop on Geometrical Models for Natural Language Semantics*, pages 1–10, Edinburgh, UK.

Or Biran and Kathleen McKeown. 2013. Classifying Taxonomic Relations between Pairs of Wikipedia Articles. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 788–794, Nagoya, Japan.

Philipp Cimiano, Lars Schmidt-Thieme, Aleksander Pivk, and Steffen Staab. 2004. Learning Taxonomic Relations from Heterogeneous Evidence. In *Proceedings of the ECAI Workshop on Ontology Learning and Population*.

Peter Clark, William R. Murray, John Thompson, Phil Harrison, Jerry Hobbs, and Christiane Fellbaum. 2007. On the Role of Lexical and World Knowledge in RTE3. In *Proceedings of the Workshop on Textual Entailment and Paraphrasing*, pages 54–59, Prague, Czech Republic.

Daoud Clarke. 2012. A Comparison of Vector-based Representations for Semantic Composition. *Computational Linguistics*, 38(1):41–71.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science*, 3944:177–190.

James Deese. 1965. *The Structure of Associations in Language and Thought*. The John Hopkins Press, Baltimore, MD, USA.

Stefan Evert. 2005. *The Statistics of Word Co-Occurrences: Word Pairs and Collocations*. Ph.D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC – A Corpus of Parsable Sentences from the Web. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology*, pages 61–68, Darmstadt, Germany.

Christiane Fellbaum, editor. 1998. *WordNet – An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA, USA.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and Evaluating *ukWaC*, a Very Large Web-Derived Corpus of English. In *Proceedings of the 4th Web as Corpus Workshop*, pages 47–54, Marrakech, Morocco.

John R. Firth. 1957. *Papers in Linguistics 1934-51*. Longmans, London, UK.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet – A Lexical-Semantic Net for German. In *Proceedings of the ACL Workshop on Automatic Information Extraction and Building Lexical Semantic Resources for NLP Applications*, pages 9–15, Madrid, Spain.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Marti Hearst. 1998. Automated Discovery of WordNet Relations. In (Fellbaum, 1998).

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional Distributional Similarity for Lexical Inference. *Natural Language Engineering*, 16(4):359–389.

Claudia Kunze and Andreas Wagner. 1999. Integrating GermaNet into EuroWordNet, a Multilingual Lexical-Semantic Database. *Sprache und Datenverarbeitung*, 23(2):5–19.

Lothar Lemnitzer and Claudia Kunze. 2007. *Computerlexikographie*. Gunter Narr Verlag, Tübingen, Germany.

Alessandro Lenci and Giulia Benotto. 2012. Identifying Hypernyms in Distributional Semantic Spaces. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, pages 75–79, Montréal, Canada.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–244.

George A. Miller and Christiane Fellbaum. 1991. Semantic Networks of English. *Cognition*, 41:197–229.

M. Lynne Murphy. 2003. *Semantic Relations and the Lexicon*. Cambridge University Press.

Roberto Navigli and Simone Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing Hypernyms in Vector Spaces with Entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 38–42, Gothenburg, Sweden.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: an Evolving Semantic Dataset for Training and Evaluation of Distributional Semantic Models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics*, pages 64–69, Beijing, China.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging using Decision Trees. In *Proceedings of the 1st International Conference on New Methods in Language Processing*.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2389–2398, Berlin, Germany.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under Siege: Linguistically-motivated Artillery for Hypernymy Detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–75, Valencia, Spain.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic Taxonomy Induction from Heterogenous Evidence. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Sydney, Australia.

Robert R. Sokal. 1958. A Statistical Method for Evaluating Systematic Relationships. *University of Kansas Science Bulletin*, 38:1409–1438.

Julie Weeds and David Weir. 2005. A Flexible Framework for Lexical Distributional Similarity. *Computational Linguistics*, 31(4):439–476.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising Measures of Lexical Distributional Similarity. In *Proceedings of the 20th International Conference of Computational Linguistics*, pages 1015–1021, Geneva, Switzerland.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge.

WIKITABLET: A Large-Scale Data-to-Text Dataset for Generating Wikipedia Article Sections

Mingda Chen **Sam Wiseman** **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{mchen,swiseman,kgimpel}@ttic.edu

Abstract

Datasets for data-to-text generation typically focus either on multi-domain, single-sentence generation or on single-domain, long-form generation. In this work, we cast generating Wikipedia sections as a data-to-text generation task and create a large-scale dataset, WIKITABLET, that pairs Wikipedia sections with their corresponding tabular data and various metadata. WIKITABLET contains millions of instances, covering a broad range of topics, as well as a variety of flavors of generation tasks with different levels of flexibility. We benchmark several training and decoding strategies on WIKITABLET. Our qualitative analysis shows that the best approaches can generate fluent and high quality texts but they struggle with coherence and factuality, showing the potential for our dataset to inspire future work on long-form generation.[1]

1 Introduction

Data-to-text generation (Kukich, 1983; McKeown, 1992) is the task of generating text based on structured data. Most existing data-to-text datasets focus on single-sentence generation, such as WIKIBIO (Lebret et al., 2016), LogicNLG (Chen et al., 2020), and ToTTo (Parikh et al., 2020). Other datasets are relatively small-scale and focus on long-form text generation, such as ROTOWIRE (Wiseman et al., 2017) and MLB (Puduppully et al., 2019). In this work, we cast generating Wikipedia sections as a data-to-text generation task and build a large-scale dataset targeting multi-sentence data-to-text generation with a variety of domains and data sources.

To this end, we create a dataset that we call WIKITABLET ("Wikipedia Tables to Text") that pairs Wikipedia sections with their corresponding tabular data and various metadata. The data resources we consider are relevant either to entire Wikipedia articles, such as Wikipedia infoboxes and Wikidata tables, or to particular sections. Data from the latter category is built automatically from either naturally-occurring hyperlinks or from named entity recognizers. This data construction approach allows us to collect large quantities of instances while still ensuring the coverage of the information in the table. We also perform various types of filtering to ensure dataset quality.

WIKITABLET contains millions of instances covering a broad range of topics and a variety of flavors of generation with different levels of flexibility. Figure 1 shows two examples from WIKITABLET. The first instance has more flexibility as it involves generating a fictional character biography in a comic book, whereas the second is more similar to standard data-to-text generation tasks, where the input tables contain all of the necessary information for generating the text. While the open-ended instances in WIKITABLET are to some extent similar to story generation (Propp, 1968; McIntyre and Lapata, 2009; Fan et al., 2018), the fact that these instances are still constrained by the input tables enables different evaluation approaches and brings new challenges (i.e., being coherent and faithful to the input tables at the same time).

Because of the range of knowledge-backed generation instances in WIKITABLET, models trained on our dataset can be used in assistive writing technologies for a broad range of topics and types of knowledge. For example, technologies can aid students in essay writing by drawing from multiple kinds of factual sources. Moreover, WIKITABLET can be used as a pretraining dataset for other relatively small-scale data-to-text datasets (e.g., ROTOWIRE). A similar idea that uses data-to-text generation to create corpora for pretraining language models has shown promising results (Agarwal et al.,

---

[1] Code, data, and pretrained models are available at `https://github.com/mingdachen/WikiTableT`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 193–209
August 1–6, 2021. ©2021 Association for Computational Linguistics

2021).

In experiments, we train several baseline models on WIKITABLET and empirically compare training and decoding strategies. We find that the best training strategies still rely on enforcing hard constraints to avoid overly repetitive texts. Human evaluations reveal that (1) humans are unable to differentiate the human written texts from the generations from our neural models; (2) while the annotations show that grammatical errors in the reference texts and the generations may prevent humans from fully understanding the texts, the best decoding strategy (i.e., beam search with n-gram blocking (Paulus et al., 2018)) does not have such a problem and shows the best performance on several aspects; (3) the degree of topical similarity between the generations and the reference texts depends on the open-endedness of the instances.

Our analysis shows that the generations are fluent and generally have high quality, but the models sometimes struggle to generate coherent texts for all the involved entities, suggesting future research directions. For example, when the instance has a high degree of flexibility, we find the models making mistakes about what a particular entity type is capable of. We also find errors in terms of the factuality of the generated text, both in terms of contradictions relative to the tables and commonsense violations.

2 Related Work

There have been efforts in creating data-to-text datasets from various resources, including sports summaries (Wiseman et al., 2017; Puduppully et al., 2019), weather forecasts (Liang et al., 2009), and commentaries (Chen and Mooney, 2008). Most of the recent datasets focus on generating single sentences given tables, such as WIKIBIO, ToTTo, LogicNLG, and WikiTableText (Bao et al., 2018), or other types of data formats, such as data triples (Vougiouklis et al., 2017; Gardent et al., 2017; Nan et al., 2021), abstract meaning representations (Flanigan et al., 2016), minimal recursion semantics (Hajdik et al., 2019), or a set of concepts (Lin et al., 2020). Other than single sentences, there have been efforts in generating groups of sentences describing humans and animals (Wang et al., 2018), and generating a post-modifier phrase for a target sentence given a sentence context (Kang et al., 2019). In this work, our focus is long-form text generation and we are interested in automatically creating a large-scale dataset containing multiple types of data-to-text instances. As shown in Table 1, WIKITABLET differs from these datasets in that it is larger in scale and contains multi-sentence texts. More details are in the next section.

Wikipedia has also been used to construct datasets for other text generation tasks, such as generating Wikipedia movie plots (Orbach and Goldberg, 2020; Rashkin et al., 2020) and short Wikipedia event summaries (Gholipour Ghalandari et al., 2020), and summarizing Wikipedia documents (Zopf, 2018; Liu* et al., 2018) or summaries of aspects of interests (Hayashi et al., 2020) from relevant documents.

As part of this work involves finding aligned tables and text, it is related to prior work on aligning Wikipedia texts to knowledge bases (Elsahar et al., 2018; Logan et al., 2019).

3 The WIKITABLET Dataset

The WIKITABLET dataset pairs Wikipedia sections[2] with their corresponding tabular data and various metadata; some of this data is relevant to entire Wikipedia articles ("article data") or article structure ("title data"), while some is section-specific ("section data"). Each data table consists of a set of **records**, each of which is a tuple containing an **attribute** and a **value**.

The instances in WIKITABLET cover a range of flavors of language generation. Some have more flexibility, requiring models to generate coherent stories based on the entities and knowledge given in the tables. The first instance in Figure 1 is such an example. The text is from the Wikipedia article entitled "Wolfsbane (comics)" and resides within two nested sections: the higher-level section "Fictional character biography" and the lower-level section "Messiah Complex". The task is challenging as models need to generate a coherent passage that can connect all the entities in the section data, and the story also needs to fit the background knowledge provided in the article data.

Other instances are more similar to standard data-to-text generation tasks, where the input tables contain all the necessary information for generating

---

[2] We define a Wikipedia section to be all text starting after a (sub)section heading and proceeding until the next (sub)section heading. We include Wikipedia sections at various nesting levels. For example, a top level section may start with a few paragraphs describing general information followed by two subsections with more specific information, in which case the example will be converted into three instances in our dataset.

During the 2007–2008 "Messiah Complex" storyline, Rahne helps Rictor infiltrate the Purifiers; she fakes being shot by Rictor. She is also a member of the new X-Force. During a battle against Lady Deathstrike and the Reavers, Rahne learns that Father Craig was in league with the Purifiers, supposedly divulging enough information about her that the Purifiers can claim to "know her well." She travels with X-Force to her former home Muir Island, now the base of the Marauders. During the climactic battle, Rahne is injured by Riptide, but her wounds, according to Professor X, are superficial and she will recover.

Section Data

Attribute	Value
PERSON	Reavers
GPE	Muir Island
group of fictional characters	Purifiers (Marvel Comics)
DATE	the 2007-2008
film character	Riptide (comics)
film character	Lady Deathstrike
PER	Father Craig

Article Data

Attribute	Value
birth name	Rahne Sinclair
instance of	superhero
member of	X-Men
from narrative universe	Marvel universe

Title Data

Document title	Wolfsbane (comics)
Section title$_1$	Fictional character biography
Section title$_2$	"Messiah Complex"

Journey to the Center of the Earth (also called Jules Verne's Journey to the Center of the Earth) is a 1959 American science fiction adventure film in color by De Luxe, distributed by 20th Century Fox. The film, produced by Charles Brackett and directed by Henry Levin, stars James Mason, Pat Boone, and Arlene Dahl. Bernard Herrmann wrote the film score, and the film's storyline was adapted by Charles Brackett from the 1864 novel of the same name by Jules Verne.

Section Data

Attribute	Value
musical composition	20th Century Fox
PERSON	Jules Verne
dependence syndrome	alcoholic
film genre	adventure film
business	Deluxe Entertainment Services Group, Inc.
based on	A Journey to the Center of the Earth

Article Data

Attribute	Value
instance of	film
director	Henry Levin
composer	Bernard Herrmann
released	1959, 12, 16
genre	science fiction film
genre	fantasy film
starring	James Mason, Pat Boone, Arlene Dahl

Title Data

Document title	Journey to the Center of the Earth (1959 film)	Section title	Introduction

Figure 1: Two examples from WIKITABLET. Only parts of the tables are shown due to space constraints. Underlined texts are hyperlinks. Records with the attributes "DATE", "PER", "PERSON", or "GPE" are from NER. The subscripts for section titles indicate the ordering of nesting, where smaller numbers are for higher level sections.

the text. The second instance in Figure 1 is an example of this sort of task. However, these tasks are still challenging due to the wide variety of topics contained in WIKITABLET.

3.1 Dataset Construction

We begin by describing the steps we take to construct WIKITABLET. More details are in the supplementary material. In general, the steps can be split into two parts: collecting data tables and filtering out texts. When collecting data, we consider five resources: Wikidata tables, infoboxes in Wikipedia pages,[3] hyperlinks in the passage, named entities in the passage obtained from named entity recognition (NER), and Wikipedia article structure. For a given Wikipedia article, we use the same infobox and Wikidata table for all sections. These tables can serve as background knowledge for the article. For each section in the article, we create a second table corresponding to section-specific data, i.e., section data. The section data contains records constructed from hyperlinks and entities identified by a named entity recognizer.[4]

We form records for named entities by using the type of the entity as the attribute and the identified entity as the value. We form records for hyperlinks as follows. For the attribute, for a hyperlink with surface text t and hyperlinked article ℓ, we use the value of the "instance of" or "subclass of" tuple in the Wikidata table for ℓ. For example, the first instance in Figure 1 will be turned into a record with attribute "superhero" and value "Wolfsbane (comics)". If ℓ does not have a Wikidata table or no appropriate tuple, we consider the parent categories of ℓ. For the value of the tuple, we use the document title of ℓ rather than the actual surface text t to avoid giving away too much information in the reference text.

Complementary to the article data, we create a title table that provides information about the position in which the section is situated, which includes the article title and the section titles for the target section. As the initial sections in Wikipedia articles do not have section titles, we use the section title "Introduction" for these.

We also perform various filtering to ensure the quality of the data records, the coverage of the input data, and the length of the reference text. The final dataset contains approximately 1.5 million instances. We randomly sample 4533 instances as the development set and 4351 as the test set. We

---

[3] Wikidata is a consistently-structured knowledge base (e.g., has a fixed set of attributes), whereas infoboxes are not consistently-structured and this flexibility sometimes allows the infobox to contain extra information. Therefore, we consider using infoboxes as extra resources.

[4] We use the NER tagger from spaCy (Honnibal and Montani, 2017) and a BERT model (Devlin et al., 2019) finetuned on CoNLL03 data (Tjong Kim Sang and De Meulder, 2003).

also ensure that there are no overlapping Wikipedia articles among splits.

3.2 Dataset Characteristics

Table 1 shows statistics for WikiTableT and related datasets. While the average length of a WikiTableT instance is not longer than some of the existing datasets, WikiTableT offers more diverse topics than the sports-related datasets RotoWire and MLB, or the biography-related dataset WikiBio. Compared to the prior work that also uses Wikipedia for constructing datasets, WikiBio, LogicNLG, ToTTo, and DART (Nan et al., 2021) all focus on sentence generation, whereas WikiTableT requires generating Wikipedia article sections, which are typically multiple sentences and therefore more challenging. WikiTableT is also much larger than all existing datasets.

To demonstrate the diversity of topics covered in WikiTableT, we use either the "instance of" or "subclass of" relation from Wikidata as the category of the article.[5] We show the top 10 most frequent document categories in Table 2. Due to the criteria we use for filtering, only 1.05% of articles in WikiTableT do not have these relations or Wikidata entries, and we omit these articles in the table. As the table demonstrates, more than 50% of the articles in WikiTableT are not about people (i.e., the topic of WikiBio), within which the most frequent category covers only 4.61%.

3.3 Dataset Challenges

In this subsection, we highlight two challenges of WikiTableT.

1. In contrast to work on evaluating commonsense knowledge in generation where reference texts are single sentences describing everyday scenes (Lin et al., 2020), WikiTableT can serve as a testbed for evaluating models' abilities to use world knowledge for generating coherent long-form text.
2. Compared to other long-form data-to-text datasets such as RotoWire where the input tables are box scores, the input tables in WikiTableT are more diverse, including both numbers (e.g., economy and population data of an area throughout years), and short phrases. This makes WikiTableT more challenging and applicable to various scenarios.

4 Methods

In this section, we describe details of models that we will benchmark on WikiTableT.

Our base model is based on the transformer (Vaswani et al., 2017). To encode tables, we linearize the tables by using special tokens to separate cells and using feature embeddings to represent records in tables. For the title table in the first instance in Figure 1 the linearized table will be

$$\begin{aligned}&\langle\text{boc}\rangle_1\text{Doc.}_1\ \text{title}_1\langle\text{bov}\rangle_1\ \text{Wolfsbane}_1\ \text{(comics)}_1\\&\langle\text{boc}\rangle_2\text{Sec.}_2\ \text{title}_2\langle\text{bov}\rangle_2\ \text{Fictional}_2\ \text{character}_2\\&\text{biography}_2\langle\text{boc}\rangle_3\cdots\langle\text{eoc}\rangle\end{aligned} \tag{1}$$

As shown in Eq. 1, we employ several techniques when encoding tables: (1) we use special tokens $\langle\text{boc}\rangle$ and $\langle\text{bov}\rangle$ to separate attributes and values, and $\langle\text{eoc}\rangle$ to indicate the end of a sequence; (2) we use subscript indices to indicate unique ID embeddings that are added to the embeddings for each record, which helps models align attributes with values; and (3) we restart the positional embeddings at each $\langle\text{boc}\rangle$, such that models will not use the ordering of the input records. In addition, we add a special embedding to each record to indicate if it is from the section table or the article/title table. In Wikidata, there could be multiple qualifiers attached to a record, in which case we replicate the record for each qualifier separately.

Similar linearization approaches have been used in prior work (Dhingra et al., 2019; Hwang et al., 2019; Herzig et al., 2020; Yin et al., 2020). With linearized tables, training and inference become similar to other sequence-to-sequence settings. We train our models with teacher-forcing and standard cross entropy loss unless otherwise specified.

4.1 Training Strategies

We experiment with three types of modifications to standard sequence-to-sequence training:

α-entmax. α-entmax (Peters et al., 2019) is a mapping from scores to a distribution that permits varying the level of sparsity in the distribution. This mapping function has been used in machine translation (Peters et al., 2019) and text generation (Martins et al., 2020). When using α-entmax in the decoder, we also replace the cross entropy loss with the α-entmax loss (Peters et al., 2019). Both

---

[5] When there are multiple values in these two relations, we pick the one that has the smallest number of words, as it often is the most generic phrase, suitable for representing the topic.

	Vocab.	Tokens	Examples	Avg. Len.	Record Types	Avg. Records	Domain
WikiTableText	-	185.0k	13.3k	13.9	3.0k	4.1	Wikipedia
WIKIBIO	400.0k	19.0M	728.0k	26.1	1.7k	19.7	Biography
ROTOWIRE	11.3k	1.6M	4.9k	337.1	39.0	628.0	Sports
MLB	38.9k	14.3M	26.3k	542.1	53.0	565.0	Sports
LogicNLG	122.0k	52.7k	37.0k	14.2	11.7k	13.5	Wikipedia
ToTTo	136.0k	1.3M	136.0k	17.4	41.8k	32.7	Wikipedia
DART	33.2k	717.1k	82.2k	21.6	-	-	Wikipedia+Restaurant
WIKITABLET	1.9M	169.0M	1.5M*	115.9	147.4k†	51.9	Wikipedia

Table 1: Statistics for several data-to-text datasets. WIKITABLET combines a large number of examples, moderate generation length (typically more than one sentence), and a large variety of record types. We omit record types and avg. records for DART as its input units are triple sets instead of table records. *887.7k unique Wikipedia articles. †Number of record types for each resource: 31.8k (Infobox), 1.7k (Wikidata), 115.6k (Hyperlinks), 17 (NER).

Category	Fraction (%)
human	45.62
film	4.61
single (music)	1.74
human settlement	1.53
album	1.41
sports season	1.26
television series	1.17
village	1.12
taxon	0.89

Table 2: Top 10 most frequent article categories and their corresponding proportions in WIKITABLET.

α-entmax and the α-entmax loss have a hyperparameter α. We follow Martins et al. (2020) and use $\alpha = 1.2$ as they found it to be the best value for reducing repetition in generation.

Copy Mechanism. Similar to prior work on data-to-text generation (Wiseman et al., 2017; Puduppully et al., 2019), we use pointer-generator network style copy attention (See et al., 2017) in the decoder.

Cyclic Loss. Cyclic losses have been shown to be effective in textual style transfer (Shetty et al., 2018; Pang and Gimpel, 2019) and neural machine translation (Cheng et al., 2016; He et al., 2016; Tu et al., 2017). Wiseman et al. (2017) also used this for data-to-text and found it helpful for generating long sequences. In this work, we experiment with adding the cyclic loss to our transformer models, where the backward model can be seen as an information extraction system. We expect that adding the cyclic loss should enable a data-to-text model to generate sentences that are more faithful to the conditioned tables. The cyclic loss is used during training only and does not affect the models during inference. More details are in the appendix.

4.2 Decoding Strategies

Massarelli et al. (2020) showed that the choice of decoding strategy can affect the faithfulness or repetitiveness of text generated by language models. We are also interested in these effects in the context of data-to-text generation, and therefore benchmark several decoding strategies on WIKITABLET. Our models use byte-pair encoding (BPE; Sennrich et al., 2016) and for all of the following strategies, we always set the minimum number of decoding steps to 100 as it improves most of the evaluation metrics, and the maximum number of decoding steps to 300.

Specifically, we benchmark (1) greedy decoding; (2) nucleus sampling (Holtzman et al., 2020) with threshold 0.9 as suggested by Holtzman et al. (2020); (3) beam search; and (4) beam search with n-gram blocking (Paulus et al., 2018) where we set the probabilities of repeated trigrams to be 0 during beam search. We set the beam size to be 5 by default. The appendix has more details about the decoding strategies.

5 Experiments

5.1 Setup

We experiment with two sizes of transformer models. One is "Base", where we use a 1-layer encoder and a 6-layer decoder, each of which has 512 hidden size and 4 attention heads. The other one is "Large", where we use a 1-layer encoder and a 12-layer decoder, each of which has 1024 hidden size and 8 attention heads. Models similar to the base configuration have shown strong performance on ROTOWIRE (Gong et al., 2019).[6] Due to limited

---

[6] When training the base model with entmax on WIKIBIO, it achieves BLEU-4 45.75 and ROUGE-4 39.39 on the test set using greedy decoding, which are comparable to the current state-of-the-art results of Liu et al. (2018).

	REP	BLEU	RL	MET	PAR-P	PAR-R	PAR-F1
References	1.2	100.0	100.0	100.0	100.0	59.2	72.9
Linearized article tables	8.0	2.2	14.7	9.3	100.0	16.3	25.6
Linearized section tables	1.0	1.9	27.9	15.5	100.0	20.9	33.4
Linearized tables	7.9	6.4	22.0	18.3	100.0	48.3	63.0
Linearized tables + references	7.6	36.5	61.3	56.5	99.9	100.0	100.0
Base models trained on the 500k training set (beam search)							
Base	33.0	15.6	36.9	20.3	66.3	28.8	37.7
Base + entmax	25.9	15.4	36.2	20.3	64.6	29.0	37.7
Base + copy	30.1	15.9	37.5	20.7	67.1	29.4	38.5
Base + copy + cyclic loss	28.0	15.7	37.5	20.8	67.5	29.7	38.9
Large models trained on the full training set (different decoding strategies)							
Large + greedy	26.8	18.9	38.5	23.5	60.4	33.1	40.4
Large + nucleus sampling	2.3	18.3	36.1	23.7	54.2	32.5	38.7
Large + beam search	18.8	19.5	39.9	23.9	65.8	34.3	42.8
Large + beam search + n-gram blocking	1.9	19.3	39.3	24.4	62.2	35.3	43.0

Table 3: Test set results for our models. When training the large models, we use the "copy + cyclic loss" setting as it gives the best performance for the base models for most of the metrics.

computational power, we parameterize our backward model as a transformer model with a 2-layer encoder and a 2-layer decoder.[7]

We use BPE with 30k merging operations. We randomly sample 500k instances from the training set and train base models on them when exploring different training strategies. We train a large model with the best setting (using the copy mechanism and cyclic loss) on the full training set. We train both models for 5 epochs. During training we perform early stopping on the development set using greedy decoding.

We report BLEU (Papineni et al., 2002), ROUGE-L (RL) (Lin, 2004), METEOR (MET) (Banerjee and Lavie, 2005), and PARENT (Dhingra et al., 2019), including precision (PAR-P), recall (PAR-R), and F1 (PAR-F1) scores. The first three metrics consider the similarities between generated texts and references, whereas PARENT also considers the similarity between the generation and the table. When using PARENT, we use all three tables, i.e., the section, article, and title tables.

As we are also interested in the repetitiveness of generated texts, we define a metric based on n-gram repetitions which we call "REP". REP computes the ratio of the number of repeated n-grams to the total number of n-grams within a text, so when REP has higher value, it indicates that the text has more repetitions. Here we consider n-grams that appear 3 or more times as repetitions and the n-grams we consider are from bigrams to 4-grams. When reporting REP scores for a dataset, we average the REP scores for each instance in the dataset. Similar metrics have been used in prior work (Holtzman et al., 2020; Welleck et al., 2020).

5.2 Results

In Table 3, we report the test results for both our base models and large models. We also report a set of baselines that are based on simply returning the linearized tables and their concatenations with the references. The linearized table baselines show how much information is already contained in the table, while the reference baselines show the upper bound performance for each metric.

In comparing training strategies, we find that using α-entmax improves REP significantly but not other metrics. Adding the cyclic loss or the copy mechanism helps improve performance for the PAR scores and REP, and combining both further improves these metrics.

When comparing decoding strategies, we find that both nucleus sampling and n-gram blocking are effective in reducing repetition. Nucleus sampling harms the PAR scores, especially PAR-P, but has less impact on the other metrics, indicating that it makes the model more likely to generate texts that are less relevant to the tables. Using beam search improves all metrics significantly when compared to greedy decoding, especially the PAR-P and REP scores. Adding n-gram blocking further reduces the REP score, pushing it to be even lower than that from nucleus sampling, but still retains the improvements in PAR scores from beam search. The best overall decoding strategy appears to be beam search with n-gram blocking.

---

[7] We did not experiment with pretrained models because they typically use the entirety of Wikipedia, which would presumably overlap with our test set.

	Grammar	Coherence	Faithfulness
Reference	4.0 (1.0)	4.1 (0.9)	3.8 (0.8)
Beam search	4.0 (1.0)	4.0 (1.0)	3.9 (1.0)
Nucleus sampling	4.0 (0.8)	4.1 (0.9)	3.9 (0.8)
n-gram blocking	4.2 (0.9)	4.2 (0.9)	3.9 (1.0)

Table 4: Average human ratings (standard deviations in parentheses) for grammaticality, coherence, and faithfulness to the input article table.

	Relevance	Support
Beam search	3.8 (1.1)	3.6 (1.2)
Nucleus sampling	3.7 (1.2)	3.8 (1.1)
n-gram blocking	3.9 (1.0)	3.8 (1.0)

Table 5: Average human ratings (standard deviations in parentheses) of relevance and support when comparing to the reference text.

6 Analysis

We now describe a manual evaluation and analyze some generated examples. All results in this section use the development set. We also conduct experiments on analyzing the effect of using the section data and the article data during training, finding that the benefits that they bring to the model performance are complementary. See the appendix for more details.

6.1 Human Evaluation

We conduct a human evaluation using generations from the large model on the development set. We choose texts shorter than 100 tokens and that cover particular topics as we found during pilot studies that annotators struggled with texts that were very long or about unfamiliar topics.[8]

We design two sets of questions. The first focuses on the text itself (i.e., grammaticality and coherence) and its faithfulness to the input article table. Since this set does not involve the reference, we can ask these questions about both generated texts and the reference texts themselves. The second set of questions evaluates the differences between the generations and the reference texts (i.e., relevance and support), allowing us to see if the generated text matches the human written section text. Specifically, relevance evaluates topical similarity between generations and references, and support evaluates whether the facts expressed in the generations are supported by or contradictory to those in the references. The full questions and numerical answer descriptions are in the appendix.

We report results in Tables 4 and 5. The scores are on a 1-5 scale with 5 being the best. For the first set, we collect 480 annotations from 38 annotators. For the second set, we collect 360 annotations from 28 annotators. We also ensure that each system has the same number of annotations.[9]

It is interesting to note from Table 4 that human annotators are unable to differentiate the human written texts from the generations from our neural models. Since the Wikipedia section texts are parts of Wikipedia articles, showing the section texts in isolation can make them difficult to understand, potentially resulting in noisy annotations. As shown by the first instance in Table 6, the text uses the pronoun “he” without clarifying what the pronoun refers to. The paragraph is rated 3 for coherence, presumably due to this ambiguity. Also, Wikipedia texts are sometimes grammatically complex and annotators can mistake them for being ungrammatical, e.g., the second instance in Table 6.

On the other hand, the coherence errors in the generated texts are not always easy to spot. See, for example, the last two instances in Table 6, where the incoherence lies in the facts that (1) it is impossible to marry a person before the person is born, and (2) senior year takes place after junior year. These details are embedded in long contexts, which may be overlooked by annotators and lead to results favorable to these neural models.

To study the relationship between coherence and grammaticality, we compute Spearman’s correlations between the human annotations for coherence and grammaticality after removing the ones with perfect scores for coherence. Table 7 shows the results. The correlations are much higher for references, beam search, and nucleus sampling than for n-gram blocking. This trend suggests that the imperfect coherence scores for the reference texts are likely because annotators find the texts to contain grammatical errors (or to possess grammatical complexity) which may prevent them from fully understanding the texts. However, n-gram blocking does not have this problem and thus achieves the best results for both coherence and grammaticality. We hypothesize that n-gram blocking is able to avoid the types of grammatical errors that

---

[8] We did not find the filtering to change the observed trends for the automatic metrics and provide the list of selected topics in the appendix.

[9] We used Amazon Mechanical Turk. To ensure annotation quality, we only recruited annotators with master qualification. We collected one annotation for each instance (so that we can cover more instances) and paid 30 cents per annotation. The amount of wage per annotation is decided by (1) the amount of time each annotator spent on the task during our pilot study and (2) a target hourly wage of approximately \$11.

Method	Text	G	C
Reference	He contested the parliamentary seat of Meriden at the 1987 general election, where he was defeated by the sitting Conservative MP Iain Mills by a margin of 16,820. He was then selected to fight the Conservative-held marginal seat of Birmingham Northfield ...	3	3
Reference	Boscawen married on 23 April 1700 in Henry VII's Chapel, Westminster Abbey, Charlotte Godfrey elder daughter and coheir of Colonel Charles Godfrey, master of the jewel office and his wife Arabella Churchill ...	3	4
Sampling	7th Marquess of Exeter married, firstly, Edith Csanady de Telegd (born **1 September 1935** in England; died 16 June 1956 in London), on **17 January 1934** ...	4	5
Blocking	... He averaged 10.9 rebounds and 3.0 assists per game **as a senior in 1987-88**. He was selected to the Sweet 16 of the NCAA Tournament **as a junior in 1988-89** ...	5	5

Table 6: Human annotation examples for grammaticality (G) and coherence (C). Due to space constraints, only parts of the texts are shown. We highlight texts that are incoherent.

	Ref.	Beam	Samp.	Block.
Spearman corr.	39.6	39.7	40.8	16.4
# annotations	67	80	76	67

Table 7: Spearman correlations between the human evaluation results for grammaticality and coherence. We omit annotations with perfect scores for coherence.

	1	2	3	4	5
Relevance	24.2	19.2	13.6	12.0	8.9
# annotations	10	48	65	124	113
Support	17.0	11.0	17.5	12.5	9.4
# annotations	13	47	68	135	97

Table 8: Averaged perplexities and the corresponding numbers of annotations for each option for the relevance and support questions (5 is the best option). We aggregate annotations for different decoding algorithms. We note that the perplexities are computed based on the reference texts using the large model.

prevent understanding because (1) unlike nucleus sampling, n-gram blocking does not rely on randomness to avoid repetition; (2) n-gram blocking does not suffer from repetitions like beam search.

We report results for the second set of questions in Table 5. The three evaluated systems show similar performance. To investigate the relationship between the degree of open-endedness of a WIKITABLET instance and its corresponding evaluation scores, we compute the averaged perplexities (based on our large models) for each option in Table 8. The most relevant generations are typically from more closed-ended or constrained instances.[10] Similarly for the support scores, more open-ended instances are distributed at score 3, which means that there is no fact supported by or contradictory to the shown tables. While the open-endedness of an instance usually depends on its topics (e.g., movie plots are open-ended), there are many cases where the models can benefit from better entity modeling, such as understanding what a particular entity type is capable of (e.g., see the last example in Sec. 6.3).

percentile	train perp.	dev perp.
10	2.3	2.5
20	3.1	3.6
30	4.0	4.7
40	4.9	6.1
50	6.1	7.6
60	7.6	9.8
70	9.8	12.9
80	13.3	18.2
90	20.6	29.3
100	278.8	129.4

Table 9: Percentiles of perplexities for training and development splits of our dataset. We use the large model.

Recent work has also found conducting human evaluation for long-form generation to be challenging, for example in the context of question answering (Krishna et al., 2021) and story generation (Akoury et al., 2020). Our observations for data-to-text generation complement theirs and we hope that our dataset can inspire future research on human evaluation for long-form text generation.

6.2 Distribution of Perplexity

To determine the fraction of WIKITABLET that can be seen as constrained, we report the percentiles of perplexities for training and development splits in Table 9. From Table 8, it can be observed that instances with perplexities around 9.0 generally lead to model generations that are closely relevant to the reference texts and mostly supported by the input tables, and therefore are likely to be the constrained instances. From Table 9, we see that at least half of our dataset has perplexities lower than 9.0, so we conjecture that half of our dataset consists of constrained instances.

6.3 Generation Examples

Table 10 shows generation examples for nucleus sampling and beam search with n-gram blocking.

---

[10] Li and Hovy (2015) use entropy as a proxy to quantify complexity of tasks. In this work, we use perplexity to measure how open-ended the instances are.

Method	Generated Text
Nucleus sampling (threshold 0.9)	In a two-issue mini-series titled ”Messiah Complex”, portrayed by Maisie Williams, Father Craig is ordered to rescue the world from Reavers and run out of the X-Force causing the X-Force to permanently withdraw. The original humans were replaced by slots and Muir Island being pulled away. They also possessed the entire X-Force after issue #9 in which the X-Force appeared. Claremont and Sinclair later reprised their parts in ”The Donderful World of Wolverine” with Wolfsbane as one of the new X-Men and her group taken back to their reservation for unknown reasons.
n-gram blocking	In the ”Messiah Complex” storyline, Wolfsbane is a member of the X-Men, a group of mutants recruited by Father Craig to fight the Reavers in Muir Island. Father Craig is killed in the battle with Reavers, but Father Craig manages to rescue the mutant from the Rain of Gesham and retrieve the mutant. She is later revealed to have been killed by Rain in a battle with Father Craig. In the aftermath, Father Craig tells Father Craig that she is the only mutant to be killed by a mutant.
Nucleus sampling (threshold 0.9)	In May 2013, McCoy allegedly assault a Spoiler business official. The incident motivated him to take a business trip to Harrisburg, Pennsylvania and sign a letter from the Buffalo Bills paying him over $50,000 as well as verbally prohibiting legal kickoff work. In September 2014, McCoy shared the issue with some former players of the first two film franchises. McCoy played a leading role in the 2012 Marvel Cinematic Universe film ”Erab of the Press”, and was named one of the film's biggest stars on Twitter. On September 9, 2014, McCoy filed a lawsuit against PYT to change its Twitter name to LeSean McCoy Productions.
n-gram blocking	On September 9, 2014, McCoy was arrested in Harrisburg, Pennsylvania on suspicion of assault. He was charged with assault and battery. In May 2013, he was fined over $50,000 by the Buffalo Bills. In September 2014, he was suspended for two games by the PYT for violating the Marvel Cinematic Universe. He was released by the Bills in October of the same year. He was cleared of all charges on Twitter, and was banned from playing in the 2014 Pro Bowl due to his Twitter account.

Table 10: Generation examples from the large model. The first example corresponds to the first instance in Figure 1. The complete set of generations is in the appendix.

We observe very different trends between the two instances in Figure 1. For the first instance about the X-Men, although both generations look fluent, their stories differ dramatically. The generated text for nucleus sampling describes a story that starts by saying Father Craig rescues the world from Reavers and ends with Wolfsbane joining as one of the new X-Men. On the other hand, n-gram blocking generates a story where Wolfsbane already is a member of X-Men, and the story says Father Craig fought and was killed by the Reavers, but manages to rescue the mutant. For the less open-ended instances (e.g., the second instance in Figure 1), different decoding strategies mostly generate similar details (see the appendix for generations).

Despite having different details, these generations appear to try to fit in as many entities from the tables as possible, in contrast to beam search (shown in the appendix) which mostly degenerates into repetition for more open-ended instances. This explains our previous observation that n-gram blocking helps with the PAR-R score.

Even though the generations are of good quality for most instances, their implausibility becomes more apparent when readers have enough background knowledge to understand the involved entities. For example, the second instance in Table 10 comes from the Wikipedia page “LeSean McCoy” (a football player) under the sections “Personal life” and “Controversies” (details in the appendix). The generation from nucleus sampling is implausible/nonsensical in some places (“assault a Spoiler business official”) and factually incorrect elsewhere (McCoy did not play a leading role in any film, and “Erab of the Press” is not an actual film). The fourth generation is implausible because a player is unlikely to be suspended for “violating the Marvel Cinematic Universe”, and it is unlikely for a person to be cleared of all charges on Twitter. Our models have limited access to knowledge about entities, e.g., the capabilities of a social media company like Twitter. Future research may incorporate extra resources, make use of pretrained models, or incorporate factuality modules to solve these problems.

7 Conclusion

We created WIKITABLET, a dataset that contains Wikipedia article sections and their corresponding tabular data and various metadata. WIKITABLET contains millions of instances covering a broad range of topics and kinds of generation tasks. Our manual evaluation showed that humans are unable to differentiate the references and model generations, and n-gram blocking performs the best on grammaticality and coherence. However, qualitative analysis showed that our models sometimes struggle with coherence and factuality, suggesting several directions for future work.

Acknowledgments

This work was supported in part by a Google Fellowship to M. Chen.

Impact Statement

We highlight a few limitations as follows: (1) Wikipedia texts are generally written in objective tones, but some of the texts may contain controversial content that even the community contributors do not agree upon; (2) models trained on our dataset may generate deceitful texts that are unfaithful to what actually happened to particular entities; (3) though the instances in WIKITABLET cover various topics, the writing style is almost always the same. Future work may explore more diverse writing styles.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484, Online. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *AAAI Conference on Artificial Intelligence*.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 128–135, New York, NY, USA. Association for Computing Machinery.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany. Association for Computational Linguistics.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. 2020. A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online. Association for Computational Linguistics.

Li Gong, Josep Crego, and Jean Senellart. 2019. Enhanced transformer model for data-to-text generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 148–156, Hong Kong. Association for Computational Linguistics.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M. Bender. 2019. Neural text generation from rich semantic representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2259–2266, Minneapolis, Minnesota. Association for Computational Linguistics.

Hiroaki Hayashi, Prashant Budania, Peng Wang, Chris Ackerson, Raj Neervannan, and Graham Neubig. 2020. WikiAsp: A dataset for multi-domain aspect-based summarization. *ArXiv*, abs/2011.07832.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 820–828. Curran Associates, Inc.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Jun Seok Kang, Robert Logan, Zewei Chu, Yang Chen, Dheeru Dua, Kevin Gimpel, Sameer Singh, and Niranjan Balasubramanian. 2019. PoMo: Generating entity-specific post-modifiers in context. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 826–838, Minneapolis, Minnesota. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, Online. Association for Computational Linguistics.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Jiwei Li and Eduard Hovy. 2015. The NLP engine: A universal turing machine for NLP.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4881–4888. AAAI Press.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy. Association for Computational Linguistics.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2020. Sparse text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4252–4273, Online. Association for Computational Linguistics.

Luca Massarelli, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, Fabrizio Silvestri, and Sebastian Riedel. 2020. How decoding strategies affect the verifiability of generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 223–235, Online. Association for Computational Linguistics.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 217–225, Suntec, Singapore. Association for Computational Linguistics.

Kathleen McKeown. 1992. *Text generation*. Cambridge University Press.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Eyal Orbach and Yoav Goldberg. 2020. Facts2Story: Controlling text generation by key facts. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2329–2345, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Richard Yuanzhe Pang and Kevin Gimpel. 2019. Unsupervised evaluation metrics and learning criteria for non-parallel textual transfer. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 138–147, Hong Kong. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, Florence, Italy. Association for Computational Linguistics.

Vladimir Propp. 1968. *Morphology of the Folktale*, volume 9. University of Texas Press.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A4nt: author attribute anonymity by adversarial training of neural machine translation. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1633–1650.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Pavlos Vougiouklis, Hady ElSahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon S. Hare, and Elena Simperl. 2017. Neural Wikipedian: Generating textual summaries from knowledge base triples. *CoRR*, abs/1711.00155.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21, Tilburg University, The Netherlands. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Markus Zopf. 2018. Auto-hMDS: Automatic construction of a large heterogeneous multilingual multi-document summarization corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

A Dataset Construction

When collecting data, we consider five resources: Wikidata tables, infoboxes in Wikipedia pages, hyperlinks in the passage, named entities in the passage obtained from named entity recognition (NER), and Wikipedia article structure. For each article in Wikipedia, we use the same infobox and Wikidata table for all sections. These tables can serve as background knowledge for the article. For each section in the article, we create a second table corresponding to section-specific data, i.e., section data. The section data contains records constructed from hyperlinks and entities identified by a named entity recognizer. Section data contributes around 25% of the records in WIKITABLET.

We filter out several entity types related to numbers[11] as the specific meanings of these numbers in the section of interest are difficult to recover from the information in the tables. After filtering, we use the identified entities as the values and the entity types as the attributes. This contributes roughly 12% of the records in our final dataset.

We also create records from hyperlinks in the section of interest. We first expand the hyperlinks available for each section with hyperlinks available in the parent categories. We first group hyperlinks across all Wikipedia articles with those same categories, and then we perform string matching between these hyperlinks and the text in the section. If there are exact matches, we will include those hyperlinks as part of the hyperlinks in this section.

Details for constructing a record with attribute a and value v for a hyperlink with surface text t and hyperlinked article ℓ are as follows. To set a, we use the value of the "instance of" or "subclass of" tuple in the Wikidata table for ℓ. If ℓ does not have a Wikidata table or no appropriate tuple, we consider the parent categories of ℓ as candidates for a. If there are multiple candidates for a, we first embed these candidates and a using GloVe (Pennington et al., 2014) embeddings and then choose the one that maximizes cosine similarity between the document titles or section titles and the candidates for a. For the value v of the tuple, we use the document title of ℓ rather than the actual surface text t to avoid giving away too much information in the reference text. The records formed by hyperlinks contribute approximately 13% of the records in WIKITABLET.

We shuffle the ordering of the records from NER and the hyperlinks to prevent models from relying on the ordering of records in the reference text.

The records from the section data can be seen as section-specific information that can make the task more solvable. Complementary to the article data, we create a title table that provides information about the position in which the section is situated, which includes the article title and the section titles for the target section. As the initial sections in Wikipedia articles do not have section titles, we use the section title "Introduction" for these.[12]

As the records in our data tables come from different resources, we perform extra filtering to remove duplicates in the records. In particular, we give Wikidata the highest priority as it is a human-annotated well-structured data resource (infoboxes are human-annotated but not well-structured due to the way they are stored on Wikipedia) and the entities from NER the lowest priority as they are automatically constructed. That is, when we identify duplicates across different resources, we will keep the records from the higher priority resource and drop those from the lower one. More specifically, the duplicates between Wikidata records and infoboxes are determined by whether there are duplicate values or duplicate attributes: for hyperlinks and infoboxes or Wikidata, they are judged by duplicate values; for NER and hyperlinks, they are based on whether there is any token overlapping between values.

After table collection, we have the following criteria for filtering out the texts: (1) we limit the text length to be between 50 and 1000 word tokens; (2) to ensure that there is sufficient information in the table, we only keep data-text pairs that contain more than 2 records per sentence and more than 15 records per 100 tokens from Wikidata and infoboxes; (3) to avoid texts such as lists of hyperlinks, we filter out texts where more than 50% of their word tokens are from hyperlink texts.

B Human Evaluation

The selected topics for human evaluations are: human (excluding the introduction and biography section), film, single (song), song, album, television series. When evaluating grammaticality and coherence, only the generated text is shown to annotators.

---

[11] List of filtered entity types: PERCENT, TIME, QUANTITY, ORDINAL, CARDINAL.

[12] Among millions of section titles in Wikipedia, there are only 4672 sections, including nested sections, that are called "Introduction". Therefore, we believe this process will not introduce much noise into the dataset.

1 = it is completely ungrammatical, as it is impossible to understand the text.
2 = it has many grammatical errors, and these errors make the text very difficult to understand.
3 = it has grammatical errors, and some of them make part of the text difficult to understand.
4 = it has some grammatical errors, but they are minor errors that do not affect reading.
5 = it is completely grammatical, as it does not have any grammatical errors.

Table 11: Rating explanations for grammaticality.

1 = it is completely incoherent, as it is impossible to piece together information in the text.
2 = it is incoherent in most places. You can only understand part of the story.
3 = it is incoherent in many places, but if you spend time reading it, you still can understand the whole story.
4 = it is mostly coherent. Although the text is incoherent in some places, it does not affect reading.
5 = it is completely coherent.

Table 12: Rating explanations for coherence.

1 = it is completely contradictory to what is described in the table.
2 = it has some facts contradictory to what is described in the table.
3 = it is not supported by the table, and it does not contradict the table.
4 = some of the text is supported by the facts in the table, and the rest of it does not contradict the facts in the table.
5 = it is completely supported by the table.

Table 13: Rating explanations for faithfulness.

1 = the text is completely irrelevant to the reference.
2 = most of the text is irrelevant to the reference.
3 = some of the text is relevant to the reference.
4 = most of the text is relevant to the reference.
5 = the text is talking about the same thing as the reference.

Table 14: Rating explanations for relevance.

The question for grammaticality is "On a scale of 1-5, how much do you think the text is grammatical? (Note: repetitions are grammatical errors.)" (option explanations are shown in Table 11), and the question for coherence is "On a scale of 1-5, how much do you think the text is coherent? (Coherence: Does the text make sense internally, avoid self-contradiction, and use a logical ordering of information?)" (rating explanations are in Table 12).

When evaluating faithfulness, we show annotators the article data and the generation. The question is "On a scale of 1-5, how much do you think the text is supported by the facts in the following table?" (rating explanations are in Table 13).

When evaluating coherence and relevance, annotators were shown the reference text and the generation, as well as the Wikipedia article title and section titles for ease of understanding the texts. Annotators were asked two questions, with one being "On a scale of 1-5, how much do you think the text is relevant to the reference" (Table 14), and the other being "On a scale of 1-5, how much do you think the text is supported by the facts in the reference?" (Table 15).

C Effect of α-entmax

In this section, we disentangle the effect of α-entmax and that of α-entmax loss. We note that (1) when not using the α-entmax loss, we use standard cross entropy loss (e.g., in the case of "base+ent." we maximize the log probabilities generated by α-entmax); (2) when combining α-entmax and copy mechanism, we aggregate the probabilities generated by α-entmax and those from softmax. This is because we use the first attention head in the transformer decoder as the copy attention, following the implementation in OpenNMT (Klein et al., 2017). While it is feasible to combine the α-entmax and α-entmax loss with the copy mechanism if we use the sparse transformer (Correia et al., 2019), we leave this for future study. We report the results in Table 16. It is interesting to see that when using greedy decoding, "ent. + ent. loss" outperforms the baseline model by a significant margin on all the metrics, however the improvement disappears (except for repetition) after we switch to use beam search as the decoding strategy. This is likely because α-entmax promotes sparsity in the generated probabilities, making beam search decoding unnecessary. Removing the α-entmax loss hurts the performance, but its gains become larger in switching to beam search decoding. Adding copy mechanism improves the performance, leading to comparable performance to the baseline model. Although "base+ent.+copy" still underperforms "base+copy" when using beam search, we believe that combining α-entmax and α-entmax loss with the copy mechanism is promising as (1) α-entmax is not used in our large models and the initial results have shown that α-entmax and the copy mechanism are complementary, so it may further improve our current best performance; (2) α-entmax already shows the best performance when using greedy decoding, which has speed and optimization advantages compared to the beam search based decoding strategies especially considering the long-form characteristic

1 = it has quite a few facts contradictory to what is described in the reference.
2 = it has some facts contradictory to what is described in the reference.
3 = it is not supported by the reference, and it does not contradict the reference.
4 = some of the text is supported by the facts in the reference, and the rest of it does not contradict the reference.
5 = it is completely supported by the reference.

Table 15: Rating explanations for supportedness.

	REP	BLEU	PAR-P	PAR-R	PAR-F1
Greedy decoding					
base	38.1	14.7	61.6	27.7	35.8
+ ent. + ent. loss	36.0	16.2	62.2	28.9	37.0
+ ent.	44.5	13.9	63.5	25.5	33.9
+ ent. + copy	43.7	14.8	64.2	26.6	35.2
+ copy	37.8	15.8	61.3	28.3	36.3
Beam search (beam size 5)					
base	33.0	15.6	66.3	28.8	37.7
+ ent. + ent. loss	25.9	15.4	64.6	29.0	37.7
+ ent.	34.7	13.8	67.2	26.6	35.8
+ ent. + copy	34.1	15.0	69.4	28.1	37.6
+ copy	30.1	15.9	67.1	29.4	38.5

Table 16: Effect of using α-entmax and α-entmax loss. When not using the α-entmax loss, we use standard cross entropy loss.

of WIKITABLET.

D Details of Cyclic Loss

In this section, we will denote the linearized table where the values are replaced with a special $\langle$mask$\rangle$ token by $u_1, \cdots, u_n$, and denote the reference text by $x_1, \cdots, x_m$. Formally, the training loss is

$$\sum_{w \in S} -\log p(w|u_1, \cdots, u_n, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_m) \quad (2)$$

where S represents the set of masked tokens, and $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$ is the sequence of token-level probabilities predicted by the forward model (in our experiments, these could either come from the softmax function, or the α-entmax function). Specifically, we multiply the backward transformer's input embedding matrix by the $\boldsymbol{v}$ probability vectors to obtain the input representations to the first encoder layer. We find that it is helpful to add a "reference loss" while training with the cyclic loss, defined as

$$\sum_{w \in S} -\log p(w|u_1, \cdots, u_n, x_1, \cdots, x_m) \quad (3)$$

This loss does not contain the generation model in it explicitly, but it does lead to an improved backward model by training it with clean inputs. Improving the backward model then increases the benefits of the cyclic loss.[13]

	REP	BLEU	PAR-P	PAR-R	PAR-F1
Both	38.1	14.7	61.6	27.7	35.8
Art. only	60.9	8.4	55.2	14.7	20.8
Sec. only	39.0	13.4	56.1	24.3	31.7

Table 17: Effect of dropping section or article data from the input (using the "base" setting).

	REP	BLEU	PAR-P	PAR-R	PAR-F1
None	37.8	15.8	61.3	28.3	36.3
Both	35.9	15.8	62.0	28.5	36.7
Art. only	37.2	15.8	61.7	28.1	36.2
Sec. only	34.8	15.9	61.9	28.2	36.2

Table 18: Effect of dropping section or article data when using cyclic training. The results are based on the "base + copy" and "base + copy + cyclic loss" settings.

E Effect of Article Data and Section Data

We report results in Table 17 for the models that are trained with partial data input, where art. only and sec. only indicate that we use only article data or section data, respectively. We always use title data. Section data contributes the most to the BLEU and PAR scores, but using section data and article data together is the best setting.

We also investigate the effect of partial data input for the cyclic loss in Table 18, where "None" is the model that is not trained with the cyclic loss. We note that in this setting, we still use both data resources as the input to the forward model, but vary the input data and the gold standard for the backward model. Although using only section data gives the best REP score and improves the PAR-P score, it does not help the model in other metrics. Combining the article data with the section data gives significant improvements to the PAR-F1 score compared to section data alone.

Both experiments show that there are interactions between these two data resources that can help models to learn better from both kinds.

F Generation Examples

We show the full set of generations in Table 19. The part of input data and reference text for Table 19 is shown in Figure 2.

---

[13] We experimented with initializing the backward model with pretrained checkpoints, but did not find it helpful.

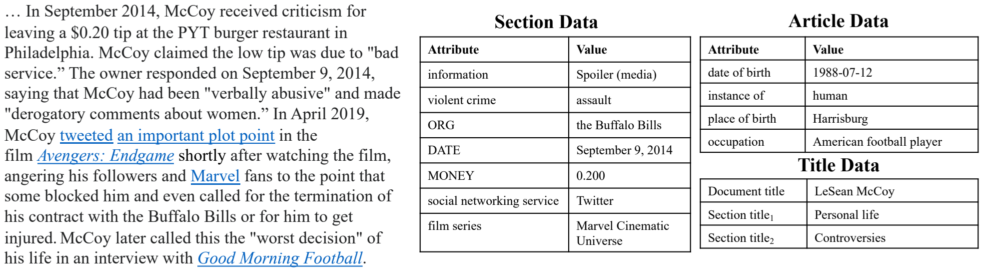

… In September 2014, McCoy received criticism for leaving a $0.20 tip at the PYT burger restaurant in Philadelphia. McCoy claimed the low tip was due to "bad service." The owner responded on September 9, 2014, saying that McCoy had been "verbally abusive" and made "derogatory comments about women." In April 2019, McCoy tweeted an important plot point in the film *Avengers: Endgame* shortly after watching the film, angering his followers and Marvel fans to the point that some blocked him and even called for the termination of his contract with the Buffalo Bills or for him to get injured. McCoy later called this the "worst decision" of his life in an interview with *Good Morning Football*.

Section Data

Attribute	**Value**
information	Spoiler (media)
violent crime	assault
ORG	the Buffalo Bills
DATE	September 9, 2014
MONEY	0.200
social networking service	Twitter
film series	Marvel Cinematic Universe

Article Data

Attribute	**Value**
date of birth	1988-07-12
instance of	human
place of birth	Harrisburg
occupation	American football player

Title Data

Document title	LeSean McCoy
Section title$_1$	Personal life
Section title$_2$	Controversies

Figure 2: An example from WIKITABLET. Only parts of the tables and reference are shown due to space constraints. Underlined texts are hyperlinks. Records with the attributes "DATE", "ORG', "MONEY" are from NER. Subscripts for section titles indicate the ordering of nesting, where smaller numbers indicate higher level sections.

Method	Generated Text
sampling 0.9	In a two-issue mini-series titled "Messiah Complex", portrayed by Maisie Williams, Father Craig is ordered to rescue the world from Reavers and run out of the X-Force causing the X-Force to permanently withdraw. The original humans were replaced by slots and Muir Island being pulled away. They also possessed the entire X-Force after issue #9 in which the X-Force appeared. Claremont and Sinclair later reprised their parts in "The Donderful World of Wolverine" with Wolfsbane as one of the new X-Men and her group taken back to their reservation for unknown reasons.
beam search (size=5)	In the "Messiah Complex" storyline, Wolfsbane is a member of the X-Men. She is a member of the X-Factor Investigations team and a member of the X-Force/
blocking (size=5)	In the "Messiah Complex" storyline, Wolfsbane is a member of the X-Men, a group of mutants recruited by Father Craig to fight the Reavers in Muir Island. Father Craig is killed in the battle with Reavers, but Father Craig manages to rescue the mutant from the Rain of Gesham and retrieve the mutant. She is later revealed to have been killed by Rain in a battle with Father Craig. In the aftermath, Father Craig tells Father Craig that she is the only mutant to be killed by a mutant.
sampling 0.9	Journey to the Center of the Earth is a 1959 American science fiction film based on Jules Verne's 1959 novel of the same name. The film, directed by Henry Levin, stars James Mason as Professor Otto Lidenbrock, a scientist who gets entangled in the power of the archenemy as he tries to impress his closest allies and compromises them with the future. The film marks the first screen appearance of future CinemaScope of 20th Century Fox lead singer Peter Ronson. "The 20th Century Fox" earned critical acclaim and was a box-office success, grossing $10 million worldwide against a production budget of $3.44 million .
beam search (size=5)	Journey to the Center of the Earth is a 1959 American science fiction film directed by Henry Levin and starring James Mason, Pat Boone, and Arlene Dahl. The screenplay by Charles Brackett and Walter Reisch is based on Jules Verne's novel "A Journey to the Center of the Earth" (1959). The film was produced by Charles Brackett and Walter Reisch and distributed by Twentieth Century-Fox Film Corporation. The film was nominated for an Academy Award for Best Art Direction. It was also nominated for an Academy Award for Best Art Direction and a Golden Globe Award for Best Visual Effects. It was also nominated for a Golden Globe Award for Best Motion Picture - Musical or Comedy, but lost to 20th Century Fox's "CinemaScope".
blocking (size=5)	Journey to the Center of the Earth is a 1959 American science fiction film directed by Henry Levin and starring James Mason, Pat Boone, and Arlene Dahl. The screenplay by Charles Brackett and Walter Reisch is based on Jules Verne's novel of the same name. The film was produced by CinemaScope and 20th Century Fox and distributed by Twentieth Century-Fox Film Corporation and Joseph M. Schenck Enterprises, Inc. It was nominated for an Academy Award for Best Art Direction.
sampling 0.9	In May 2013, McCoy allegedly assault a Spoiler business official. The incident motivated him to take a business trip to Harrisburg, Pennsylvania and sign a letter from the Buffalo Bills paying him over $50,000 as well as verbally prohibiting legal kickoff work. In September 2014, McCoy shared the issue with some former players of the first two film franchises. McCoy played a leading role in the 2012 Marvel Cinematic Universe film "Erab of the Press", and was named one of the film's biggest stars on Twitter. On September 9, 2014, McCoy filed a lawsuit against PYT to change its Twitter name to LeSean McCoy Productions.
beam search (size=5)	On September 9, 2014, McCoy was arrested in Harrisburg, Pennsylvania on suspicion of assault. He was charged with assault and battery. In May 2013, McCoy was fined over $50,000 by the Buffalo Bills. In September 2014, McCoy was arrested in Harrisburg, Pennsylvania on suspicion of assaulting a woman who had been raped by McCoy. McCoy was charged with assault and possession of marijuana. McCoy was suspended from the PYT for the first two games of the Marvel Cinematic Universe.
blocking (size=5)	On September 9, 2014, McCoy was arrested in Harrisburg, Pennsylvania on suspicion of assault. He was charged with assault and battery. In May 2013, he was fined over $50,000 by the Buffalo Bills. In September 2014, he was suspended for two games by the PYT for violating the Marvel Cinematic Universe. He was released by the Bills in October of the same year. He was cleared of all charges on Twitter, and was banned from playing in the 2014 Pro Bowl due to his Twitter account.

Table 19: Top: generation examples for the first instance in Figure 1. Middle: generation examples for the second instance in Figure 1. Bottom: generation examples that correspond to the instance in Figure 2.

G Details of Decoding Strategies

Nucleus Sampling. Generating long sequences usually suffers from repetitions. Nucleus sampling (Holtzman et al., 2020) aims to reduce the repetitions in generations by sampling from truncated probability distributions. The truncation is based on whether the cumulative probability is above a threshold. We set the threshold to be 0.9 as suggested in Holtzman et al. (2020).

Beam Search with n-gram Blocking. Paulus et al. (2018) found it effective to reduce the repetitions during beam search by "blocking" n-grams that have been generated in previous decoding steps. We follow their approach by using trigram blocking and setting the probability of repeated trigrams to be 0 during beam search.

CoDesc: A Large Code–Description Parallel Dataset

Masum Hasan[1]*, **Tanveer Muttaqueen**[1]*, **Abdullah Al Ishtiaq**[1]
Kazi Sajeed Mehrab[1], **Md. Mahim Anjum Haque**[1], **Tahmid Hasan**[1]
Wasi Uddin Ahmad[2], **Anindya Iqbal**[1], **Rifat Shahriyar**[1]
[1]Bangladesh University of Engineering and Technology (BUET)
[2]University of California, Los Angeles
`masum@ra.cse.buet.ac.bd`, `1505002.tm@ugrad.cse.buet.ac.bd`
`rifat@cse.buet.ac.bd`

Abstract

Translation between natural language and source code can help software development by enabling developers to comprehend, ideate, search, and write computer programs in natural language. Despite growing interest from the industry and the research community, this task is often difficult due to the lack of large standard datasets suitable for training deep neural models, standard noise removal methods, and evaluation benchmarks. This leaves researchers to collect new small-scale datasets, resulting in inconsistencies across published works. In this study, we present CoDesc - a large parallel dataset composed of 4.2 million Java methods and natural language descriptions. With extensive analysis, we identify and remove prevailing noise patterns from the dataset. We demonstrate the proficiency of CoDesc in two complementary tasks for code–description pairs: code summarization and code search. We show that the dataset helps improve code search by up to 22% and achieves the new state-of-the-art in code summarization. Furthermore, we show CoDesc's effectiveness in pre-training–fine-tuning setup, opening possibilities in building pretrained language models for Java. To facilitate future research, we release the dataset, a data processing tool, and a benchmark at https://github.com/csebuetnlp/CoDesc.

1 Introduction

Neural models for natural language processing have benefited from large datasets and standard evaluation benchmarks (Wang et al., 2019b,a; Rajpurkar et al., 2016; Hermann et al., 2015; CommonCrawl). However, the programming language counterpart is lagging behind due to a lack in such large datasets and benchmarks. To put this into perspective, the original Transformer network (Vaswani et al., 2017) was trained on WMT'14 English–German and English–French datasets (Bojar et al., 2014) containing 4.5 million and 36 million parallel sentences, respectively, whereas a similar network that achieved state-of-the-art results in source code summarization has been trained on only 69 thousand code-description pairs (Ahmad et al., 2020). We argue that the existing models used for programming language tasks in the literature have a significant scope of improvement given a large, good-quality dataset, and such a dataset is the missing link for effectively applying deep learning methods on programming languages.

In this work, we collect and release a large (4.2 million) Java source code - natural language (NL) parallel dataset along with denoising methods and baseline results. We apply our dataset to established works in both training from scratch and pre-training–fine-tuning setting and we demonstrate a notable performance gain in both settings. We gain 10% to 22% improvement over baseline code search models using CoDesc, and attain performances comparable to models having 8× more parameters. We achieve a new state-of-the-art BLEU score of 45.89 in code summarization by pretraining a Transformer network with our dataset for two epochs. With extensive empirical analysis, we propose a set of noise removal techniques for the source code and the NL descriptions in our dataset.

Our work brings together several datasets and multiple tasks on the intersection of Natural Language Processing (NLP) and Software Engineering (SE), such as code summarization, code search and code synthesis, and allows researchers to compare their methods on the same benchmark. It also opens the door for building large pretrained models to jointly learn code and NL representations that can be leveraged in downstream tasks that do not have adequate data, such as, code refactoring, clone detection, etc. as done by Feng et al. (2020).

---

[*] Equal contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 210–218
August 1–6, 2021.

Source	#Projects	#Raw data	#Clean data	Code			Description		
				#Unique tokens	**Avg len**	**≤200**	**#Unique tokens**	**Avg len**	**≤50**
CSN-Java	N/A	542,991	490,169	284,214	140.41	83.42	168,507	25.14	89.42
DeepCom	9,714	588,108	424,028	306,422	128.35	84.04	91,933	17.80	94.76
FunCom	28,000	2,149,121	2,130,247	469,354	51.30	99.83	399,338	15.52	95.87
CONCODE	33,000	2,184,310	733,040	131,852	33.75	99.99	166,239	14.87	96.27
CSN-Py2Java	N/A	456,000	434,032	414,018	163.78	72.32	223,277	57.11	68.69
CoDesc (All)	N/A	5,920,530	4,211,516	1,128,909	77.97	93.53	813,078	21.04	92.28
Balanced *train-valid-test* split for CoDesc data									
train	-	-	3,369,218	991,395	78.01	93.53	718,204	21.05	92.28
valid	-	-	421,149	269,435	77.73	93.51	188,145	21.08	92.26
test	-	-	421,149	269,318	77.88	93.55	187,230	20.97	92.33

Table 1: Statistics of CoDesc datasets and a balanced train-valid-test split. ≤200 and ≤50 indicates the percentage (%) of data where source code and description are smaller than 200 and 50 tokens, respectively.

2 Related Works

Code-Description Parallel Datasets With the advent of data-driven code search and code summarization methods, several datasets are proposed to facilitate research in code-NL parallel tasks. Husain et al. (2019) introduced CodeSearchNet (CSN), a benchmark for code search techniques with 2.1 million code-NL parallel data in 6 programming languages, 6.4 million monolingual code data, a leader-board, and baseline results with 5 code search techniques. CONCODE (Iyer et al., 2018), DEEPCOM (Hu et al., 2018a), FunCom (LeClair and McMillan, 2019) are some notable dataset papers that released 2.18 million, 2.15 million, and 0.59 million parallel data respectively. Clement et al. (2020) released a parallel corpus of 26 million monolingual Python methods and 7.7 million method-docstring pairs. CoNaLa (Yin et al., 2018) is a Python line by line natural language description dataset containing nearly 3k parallel data.

Code Search and Summarization CODE-NN (Iyer et al., 2016) is a pioneering work in data-driven code summarization. The CodeSearchNet dataset paved the way for CodeBERT (Feng et al., 2020), a pretrained BERT (Devlin et al., 2019) model trained on CSN data with Masked Language Modeling (MLM) (Devlin et al., 2019) and Replaced Token Detection (RTD) (Clark et al., 2020) objective that achieved a high performance in the CSN benchmark. Wei et al. (2019) proposed a dual learning method that simultaneously trained code summarization and code generation and improved both of them using 60k parallel data. In the same dataset, Ahmad et al. (2020) achieved state-of-the-art results in source code summarization using a Transformer network (Vaswani et al., 2017). Along with the mentioned dataset, Clement et al. (2020) presented PyMT5, a text to text transformer that notably improved method generation and code summarization. Ahmad et al. (2021) collected more than 300 GB monolingual code and NL data, and trained PLBART, a pretrained seq2seq model for both program understanding and comprehension.

3 CoDesc Dataset

3.1 Data Sources

We collect our data from several sources and formulate rules for data cleaning. 5 of the authors spent 45-50 man-hours manually going over the dataset to identify patterns of noises in different data sources. Upon group discussion, common patterns were identified and a noise removal method was established. Details about these noise patterns are provided in Appendix A.

One of the datasets used in CoDesc is CodeSearchNet (CSN)[1] (Husain et al., 2019) - a parallel method-description dataset for code search. Furthermore, other datasets used are DeepCom[2] (Hu et al., 2018a), CONCODE[3] (Iyer et al., 2018), FunCom[4] (LeClair and McMillan, 2019) - datasets created for code summarization. The CodeSearchNet dataset originally contained 6 programming languages, from which the Java methods are directly used in CoDesc, however, the Python methods are used after being automatically translated to Java. We combine all aforementioned datasets to create CoDesc. Appendix B shows a

---

[1] https://github.com/github/CodeSearchNet

[2] https://github.com/xing-hu/DeepCom

[3] https://github.com/sriniiyer/concode

[4] http://leclair.tech/data/funcom/

sample code-description parallel data from each of these datasets. Table 1 describes our data sources and their characteristics in detail.

CSN Python to Java Translation To utilize maximum possible data from the CSN CORPUS, we translate the Python methods to Java using TransCoder (Lachaux et al., 2020), a state-of-the-art, neural source-to-source compiler. We modified and re-released the open-source implementation of TransCoder[5], enabling it to translate data in batches instead of one at a time, and resulting in a 16X faster translation. Upon empirical inspection, we found that the converted Java codes are human-readable and bear a strong resemblance to the original Python code intent. The converted codes seem correct to the human eye and their syntax matches with Java syntax. However, a few cases the transcompiler suffers are – converting to Java library methods, and converting from Python coding conventions that does not have a Java equivalent (e.g. use of SELF). These conversion errors, however, were not severe enough to affect our model to learn the NL-source code mapping.

3.2 Data Cleaning and Noise Removal

We created an easy-to-use, parameterized data processing tool for removing the different types of noise that we observed in our dataset. From the natural language descriptions, we remove symbols and characters that do not carry a meaning in a natural language description, such as, comment tags (e.g., `//, /*, */`), stray code characters (e.g., `@, #, {, }`, etc.), HTML and XML tags, non-ASCII and escape characters, and some patterns of autogenerated tags (e.g., `@param, @return, @throws`, etc.). From source code, we remove comments and the non-ASCII and escape characters. In previous studies, many meaningful data are discarded due to having some noisy patterns/symbols either in the code or description (Husain et al., 2019; Iyer et al., 2018; LeClair and McMillan, 2019). We identify and remove the noisy part of the data points without excluding them from the dataset to reduce data loss during preprocessing.

For both source code and NL description, we split CamelCase and snake_case code tokens into subtokens (e.g., Camel Case, snake case) and separate linked alphabets and numbers (e.g., var0 to var 0) (Ahmad et al., 2020; LeClair and McMillan, 2019). After the aforementioned processing, we remove the data points where the source code is less than 3 tokens, or the description contains less than 2 alphabets (Husain et al., 2019). We lowercase the natural language as the case is not necessary for describing codes. We release our data processing tool along with the CoDesc dataset for applying the dataset to diverse tasks.

3.3 Dataset Characteristics

After the previous steps, we are left with nearly 4.2 million Java method and description parallel data. Table 1 presents the statistics characteristics of our dataset. The combined CoDesc dataset consists of more than one million unique tokens, which is significantly larger than natural language vocabulary (Chen et al., 2019). This can be partially attributed to inseparable multi-words (e.g. 'updateproductvariationlocalizeddeltaprice') in our dataset. Hence, we perform BPE (Sennrich et al., 2016) tokenization in our preprocessing pipeline. We also see that although the average token length of Java source codes vary in the different dataset sources, the natural language descriptions have a relatively uniform length. We create a balanced, deduplicated, and representative *train–valid–test* dataset by splitting individual source-dataset in 8:1:1 ratio (Table 1).

4 Experiments

We evaluate our code-description corpus in two well-known complementary tasks: source code summarization and natural language code search. In this section, we demonstrate that models trained on CoDesc bring about a noticeable improvement over two established baselines in code search and code summarization. Each benchmarking follows a standard cleaning, preprocessing, and train-test de-duplication process.

4.1 Natural Language Code Search

We use the code search models used by Husain et al. (2019) that jointly trains a source code and an NL encoder networks to minimize their encoded vector distance (Figure. 1). We apply our dataset on the CODESEARCHNET (CSN) (Husain et al., 2019) – a well-studied benchmark in the semantic code search literature. We train 5 different encoder networks (Table. 2) with the CSN Java dataset, and CoDesc respectively. We compare our results with CodeBERT and RoBERTa (code) (Feng et al., 2020), two pretrained models achieving state-of-

---

[5] https://github.com/csebuetnlp/TransCoder

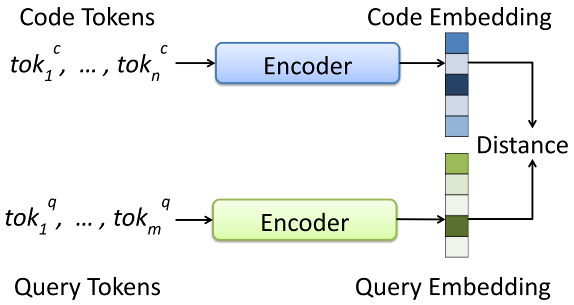

Figure 1: Code search model architecture; code and NL (query) encoders jointly train to reduce their embedded distance. During search, we select the code that is closest to the query in their shared embedding space.

Model	#Param	CSN test MRR	
		CSN-Java	CoDesc
NBOW	11.6 M	0.589	0.683
RNN	12.6 M	0.582	0.679
Sel-attn	13.6 M	0.583	0.723
1D Conv	16.4 M	0.520	0.686
Conv self-attn	16.0 M	0.509	**0.729**
State-of-the-art models			
RoBERTa (code)	125 M	0.721	
CodeBERT	125 M	**0.748**	

Table 2: Baseline models trained with default dataset and CoDesc, along with, comparison with SoTA pretrained models in CSN Java test set. Training on CoDesc outperforms training on CSN-Java only, and it is comparable to SOTA with 8x fewer parameters.

the-art results in CSN Benchmark. They are trained with a Masked Language Modeling (MLM) (Devlin et al., 2019) objective on 2.1 million bimodal code-NL data, and 6.4 million unimodal data released with CODESEARCHNET.

Results We use Mean Reciprocal Rank (MRR) – the commonly used evaluation metric for code search (Husain et al., 2019; Sachdev et al., 2018; Cambronero et al., 2019) as the evaluation criteria for code search. Table 2 shows our results, along with state-of-the-art models (Liu et al., 2019; Feng et al., 2020) that have nearly 8-10 times more parameters than the baseline networks and a more complex training objective. We achieve remarkably close performance with the state-of-the-art models with much simpler and smaller networks.

4.2 Source Code Summarization

For this task, we follow the methodology proposed by Ahmad et al. (2020). They used a seq2seq Transformer (Vaswani et al., 2017) network with 77M parameters with relative positional encoding (Shaw et al., 2018) and copying mechanism (See et al., 2017) and achieved state-of-the-art results.

Methods	BLEU	METEOR	ROUGE-L
Transformer	44.58	26.43	54.76
CoDesc pretrained	**45.89**	**28.01**	**56.59**

Table 3: Code summarization with Ahmad et al. (2020) proposed Transformer network with and without pretraining with CoDesc.

Data preparation Ahmad et al. (2020) used a Java dataset released by Hu et al. (2018b) and preprocessed by Wei et al. (2019) consisting of *training*, *validation*, and *test* datasets of size 69,708, 8,714, and 8,714 respectively. We refer to this training data as *train-small*. We create a new dataset *CoDesc-train* by combining *train-small* with CoDesc. We replace all literals as Wei et al. (2019) and tokenize the dataset using Character BPE Tokenization (Sennrich et al., 2016) to create the same size vocabulary as the previous works.

Training We train a Transformer model proposed by Ahmad et al. (2020) with *CoDesc-train* dataset. We use Adam optimizer with an initial learning rate of 10^{-4}, mini-batch size of 32, and dropout rate 0.2, vocabulary size 50k for code and 30k for NL. However, we use maximum input length of 200 token instead of 150 based on our observation of CoDesc dataset from Table 1. Each epoch of the model took nearly 8 hours in an NVIDIA V100 16GB GPU. In comparison, the *train-small* dataset took 8.5 minutes only. For limitation of computational resource, we saved the network weights after training it with the large dataset for two epochs, and to be consistent with the original implementation, trained them further with the *train-small* dataset for a maximum of 198 more epochs. We perform an early stop if the validation performance does not improve for consecutive 20 epoch. The pretraining provides the network parameters a more favorable initialization than random, helping the network find better local minima.

Results Table 3 shows that our two epoch pretraining with CoDesc significantly improves the state-of-the-art code summarization methods in all three evaluation metrics – BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and ROUGE-L (Lin, 2004). We observe that the pretrained model often generates more descriptive summary even when it achieves lower BLEU score (Fig. 2). We believe the model has more room for

```
public void makeImmutable(){
   if(mutable){
      if(results ! = null){
         int length = results.size();
         for(int i = NUM; i < length; i + +){
            Result result = (Result)results.get(i); result.makeImmutable();
         } results = Collections.unmodifiableList(results);
      } mutable = BOOL; } }
```

Human written: makes the object immutable

Transformer prediction (BLEU: 1.0): makes the object immutable

CoDesc pretrained model prediction (BLEU: 0.12): if there are any object in the list then the object is not immutable

Figure 2: CoDesc pretrained model generates more descriptive summary, even in cases it achieves lower score.

Dataset	**Raw data**	**Clean data**	**Inc. (%)**
CSN (Java)	0.5870	0.6427	5.57
DeepCom	0.4677	0.6069	13.92
FunCom	0.5379	0.6366	9.87
CONCODE	0.5444	0.6234	7.90
CSN (Python2Java)	0.5081	0.5546	4.65
CoDesc (All)	0.5852	0.6826	9.74

Table 4: MRR of individual datasets (Section 3.1) before and after noise removal.

improvement with further pretraining and we wish to validate this in future work.

4.3 Ablation & Analysis

To quantify the effect of individual data sources and our noise removal methodology, we train each dataset before and after applying our data cleaning method using an NBOW model and test them in the CSN benchmark using their released test set.

Although our collected data was already cleaned by the respective authors, Table 4 shows that the performance of every dataset improves drastically after our noise removal. Interestingly, without our extra layer of data cleaning, CoDesc dataset performs worse than training with only CSN data although being significantly larger. This shows the importance of a standard cleaning and processing method. Moreover, CSN (Java) have the highest accuracy, which can be attributed to the fact that it came from the same distribution of data as the evaluation and test sets, and hence contains similar tokens and patterns (Husain et al., 2019). We can see from Table 4 that the model trained with CSN (Python2Java) achieves an MRR score of 0.5548. Although this score is lower than other datasets, it is still a good indication that the translated data is helping the model is to learn NL-code association.

New Benchmark Results in Code Search We provide a new set of benchmark results for CoDesc dataset in natural language code search. We train, validate, and test an NBOW, an RNN, and a Self-attn code search network with the balanced *train*, *validation*, and *test* data shown in Table 1. The three models achieve MRR score of **0.812**, **0.766**, and **0.839** respectively.

5 Discussion and Conclusion

In this work, we have accumulated CoDesc – a large code-description parallel dataset and established baseline results. CoDesc brings a noteworthy improvement in two tasks: code search and code summarization. We believe CoDesc will serve as a base for future studies on code-description joint tasks. We also show that automatically translated source code from a source-to-source compiler can be applied in a code-NL parallel task, suggesting that, translating our Java dataset to other programming languages can also be helpful.

The most striking finding of our study is that, by training with 2X larger parallel data, we achieve equivalent performance to models having 8X parameters (Feng et al., 2020) in code search. This raises an interesting question: are we fully utilizing the model capacities in code–description studies? From our pretraining results in code summarization, it can be reasonably assumed that pretraining with our large dataset the larger models will also improve further. In future works, we wish to apply new techniques for code search, code summarization, along with exploring our dataset for general-purpose code synthesis, where the best models are still struggling in accuracy (Wei et al., 2019; Yin and Neubig, 2017).

Acknowledgement

We thank the ICT Division, Bangladesh for funding the project and Intelligent Machines Limited for providing the cloud support.

References

Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A transformer-based approach for source code summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4998–5007, Online. Association for Computational Linguistics.

Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Unified pre-training for program understanding and generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2668, Online. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jose Cambronero, Hongyu Li, Seohyun Kim, Koushik Sen, and Satish Chandra. 2019. When deep learning met code search. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, pages 964–974.

Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Yang Wang. 2019. How large a vocabulary does text classification need? a variational approach to vocabulary selection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3487–3497, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Colin Clement, Dawn Drain, Jonathan Timcheck, Alexey Svyatkovskiy, and Neel Sundaresan. 2020. PyMT5: multi-mode translation of natural language and python code with transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9052–9065, Online. Association for Computational Linguistics.

CommonCrawl. Common crawl. `https://commoncrawl.org/`. Accessed: 2021-01-31.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS 2015)*, pages 1693–1701, Montreal, Canada.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018a. Deep code comment generation. In *Proceedings of the 26th Conference on Program Comprehension*, ICPC '18, page 200–210.

Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018b. Summarizing source code with transferred api knowledge. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2269–2275. International Joint Conferences on Artificial Intelligence Organization.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652, Brussels, Belgium. Association for Computational Linguistics.

Marie-Anne Lachaux, Baptiste Roziere, Lowik Chanussot, and Guillaume Lample. 2020. Unsupervised translation of programming languages. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*.

Alexander LeClair and Collin McMillan. 2019. Recommendations for datasets for source code summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3931–3937, Minneapolis, Minnesota. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

C. Lopes, S. Bajracharya, J. Ossher, and P. Baldi. 2010. UCI source code data sets.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Saksham Sachdev, Hongyu Li, Sifei Luan, Seohyun Kim, Koushik Sen, and Satish Chandra. 2018. Retrieval on source code: a neural code search. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2018, page 31–41.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, page 6000–6010, Long Beach, California, USA.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 3266–3280, Vancouver, Canada.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code generation as a dual task of code summarization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 6563–6573, Vancouver, Canada.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *International Conference on Mining Software Repositories*, MSR, pages 476–486. ACM.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Supplementary Material: Appendices

A Dataset Details

CodeSearchNet (CSN) Corpus is a code search dataset for 6 programming languages (Husain et al., 2019).[6] Despite the authors' effort for data cleaning, in our observation, CSN CORPUS is one of the noisiest. The dataset contains duplicate descriptions, inseparable multi-words (e.g., `updateproductvariationlocalizeddeltaprice`, `updatelocationinventory`), XML tags (e.g., `<tt>`, `<soup>`, `<sub>`), non-English documentation, non-ASCII and escape characters, unwanted symbols (e.g., `@`, `#`, `{`, `}`), deprecated methods and descriptions, comments inside code, annotations (e.g., `@link`, `@code`, `@inheritdoc`) in description, etc. Datapoints truncated by TransCoder during Python to Java translation (total 27,471) are marked with a special flag in our released corpus.

DeepCom Hu et al. (2018a) released a dataset of 588,108 Java method and documentation pairs collected from 9,714 GitHub projects for code summarization[7]. Similar to CODESEARCHNET (Husain et al., 2019), they considered the first sentence of a documentation as the summary of the method as it typically describes the functionalities of Java methods. They filter out empty and single world descriptions and the setter, getter, constructors, and test methods, since they are easy for a model to summarize. In our manual analysis, we found HTML tags (e.g. `<tt>`, `<p>`, `<p class = ...>` , `<li>`, `<ul>`), comment tags, annotations, escape characters inside descriptions, empty parentheses as descriptions, repetitive and non-meaningful descriptions, comments inside source code, etc. Despite the authors' claim, we found numerous test methods in the dataset, which were mostly meaningful data.

CONCODE Iyer et al. (2018) released a dataset named CONCODE, collected by mining nearly 33,00 GitHub repositories[8]. In their preprocessed dataset, they replaced the names of the identifier and method names with generic terms, (e.g., `arg0`, `loc0`, `function`, etc.) and replaced all string literals with constants. This created a discrepancy with the other datasets, hence, we opted for their unprocessed dataset rather than the processed version. The unprocessed dataset released with CONCODE contained approximately 2.1 million Java methods and lowercased Javadoc-style document pairs. Upon duplicate removal, we were left with 733,878 datapoints.

Although some noises were present in this dataset, we found this data to be least noisy in manual observation. We find that because of lower casing the documentations, some CamelCase tokens became inseparable. The dataset also contained non-English comments with English alphabets (mostly Italian). We found these documents hard to identify and remove.

FunCom LeClair and McMillan (2019) released a dataset of over 2.1 million pairs of Java methods and one-sentence method descriptions from over 28k Java projects[9]. They collected this dataset by filtering over 51 million Java methods from UCI Source Code datasets (Lopes et al., 2010). In their preprocessing step, LeClair and McMillan (2019) removed all datapoints where the method is more than 100 tokens long, or the method description is over 13 tokens or below 3 tokens.

In our observation of this dataset, we found method descriptions containing HTML tokens (e.g. `<tt>`, annotations (e.g., `@link`, `@param`), comment tokens, unwanted symbols, solely non-alphabetic characters, etc. It also contained comments inside methods, and a large portion of the data were `getter`, `setter`, `tester`, and `toString` methods.

B Sample Data

```
protected void hideTabs(){
  if (getPageCount() <= 1) {
    setPageText(0,"");
    if (getContainer() instanceof
        CTabFolder) {
      ((CTabFolder)getContainer())
      .setTabHeight(1);
      Point point =
          getContainer().getSize();
      getContainer().setSize(point.x,
        point.y + 6);
    }
  }
}
```

Description: if there is just one page in the multi - page editor part , this hides the single tab at the bottom. (DeepCom)

---

[6] https://github.com/github/CodeSearchNet

[7] https://github.com/xing-hu/DeepCom

[8] https://github.com/sriniiyer/concode

[9] http://leclair.tech/data/funcom/

```
@Exported
   public boolean isIdle() {
      lock.readLock().lock();
      try {
         return workUnit == null &&
             executable == null;
      } finally {
         lock.readLock().unlock();
      }
   }
```

Description: returns true if this executor is ready for action. (CodeSearchNet)

```
public static void dbCommand(ParserArgs
    args){
  final Synergy synergy
     =(Synergy)args.get("synergy");
  if(synergy.reset){
    synergy.resetDb();
    synergy.update = true;
  }
  if(synergy.update){
    synergy.updateDb();
  }
}
```

Description: manages synergy db state (CodeSearchNet-Python to Java)

```
Object getBean(String beanName){
   if(null == beanName){
      return null;
   }
   return
       applicationContext.getBean(beanName);
}
```

Description: this method is used to retrieve a bean by its name. note that this may result in new bean creation if the scope is set to “prototype” in the bean configuration file. (CONCODE)

```
public void sort (boolean
    transformChanged) {
   if (list Size > 1){
      if (tlist == null || tlist.length
          != list.length){
         tlist = list.clone();
      } else {
         System.arraycopy(list, 0,
             tlist, 0, list.length);
      }
      if (transform Changed) {
         for(int i = 0; i < listSize;
             i++) {
            list[i]
            .computeLastDistance(owner);
         }
      }
      SortUtil.msort(tlist, list, 0,
          list Size - 1, c);
   }
}
```

Description: sorts the elements in the list acording to their comparator. there are two reasons why lights should be resorted. first, if the lights have moved, that means their distance to the spatial changed. second, if the spatial itself moved, it means the distance from it to the individual lights might have changed. (FunCom)

Deep Cognitive Reasoning Network for Multi-hop Question Answering over Knowledge Graphs

Jianyu Cai **Zhanqiu Zhang** **Feng Wu** **Jie Wang**[*]
University of Science and Technology of China
Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
`{jycai,zqzhang}@mail.ustc.edu.cn`
`{fengwu, jiewangx}@ustc.edu.cn`

Abstract

Knowledge Graphs (KGs) provide human knowledge with nodes and edges being entities and relations among them, respectively. Multi-hop question answering over KGs—which aims to find answer entities of given questions through reasoning paths in KGs—has attracted great attention from both academia and industry recently. However, this task remains challenging, as it requires to accurately identify answers in a large candidate entity set, of which the size grows exponentially with the number of reasoning hops. To tackle this problem, we propose a novel **D**eep **C**ognitive **R**easoning **N**etwork (DCRN), which is inspired by the *dual process theory* in cognitive science. Specifically, DCRN consists of two phases—the *unconscious* phase and the *conscious* phase. The unconscious phase first retrieves informative evidence from candidate entities by leveraging their semantic information. Then, the conscious phase accurately identifies answers by performing sequential reasoning according to the graph structure on the retrieved evidence. Experiments demonstrate that DCRN significantly outperforms state-of-the-art methods on benchmark datasets.

1 Introduction

Knowledge Graphs (KGs) store structured human knowledge, in which nodes represent entities and edges represent relations between pairs of entities. Multi-hop Question Answering over KGs (KGQA) aims to find answer entities by reasoning over paths in KGs. We illustrate this task with an example in Figure 1. Recently, multi-hop question answering over KGs has attracted great attention from both academia and industry (Li et al., 2017; Fu et al., 2020; Saxena et al., 2020). However, this task remains challenging, because the number of candidate entities grows exponentially with the number of reasoning hops (Sun et al., 2018, 2019a), making it difficult to accurately identify answers.

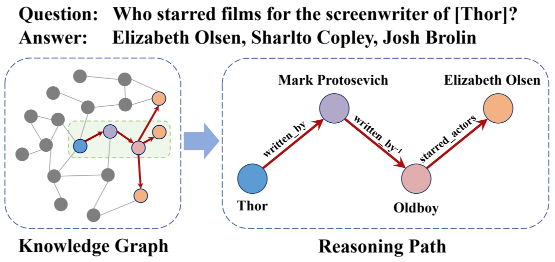

Figure 1: Illustration of multi-hop question answering over KGs. Given a natural language question, we start from the topic entity in it, and reason along paths in KGs to find answers.

Previous works mitigate this problem by reducing the size of the candidate entity sets, but they often sacrifice the recall of answers. These methods including GRAFT-Net (Sun et al., 2018) and PullNet (Sun et al., 2019a) first extract question-specific subgraphs, and then perform multi-hop reasoning on the extracted subgraph via Graph Neural Networks (GNNs) to find answers. However, these approaches often sacrifice the recall of answers in exchange for small candidate entity sets. That is, the extracted subgraph may contain no answer at all. This trade-off between the recall of answer entities and the size of candidate entity sets limits their practical usage. Therefore, it is still desirable to find an approach that is capable of accurately identifying answers without sacrificing their recalls.

To tackle this problem, we take inspiration from the *dual process theory* (Evans, 1984, 2003, 2008) in cognitive science and propose a novel **D**eep **C**ognitive **R**easoning **N**etwork (DCRN). In cognitive science, researchers found that humans can reason over a large-capacity memory to find answers (Wang et al., 2003). Specifically, the *dual*

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 219–229
August 1–6, 2021.

process theory (Evans, 1984, 2003, 2008) suggests that humans accomplish cognitive tasks by first exploiting *fast intuition* to retrieve task-relevant evidence via an *unconscious process*, and then performing *sequential reasoning* based on the aforementioned evidence to derive answers via a *conscious process*. Similarly, the proposed DCRN consists of two phases. The first one is the unconscious phase, which can retrieve informative evidence by *softly* selecting candidate entities that are most likely to be correct answers. The second one is the conscious phase, which can accurately identify answers by performing sequential reasoning with Bayesian networks based the retrieved evidence from the first phase. Experiments demonstrate that DCRN significantly outperforms state-of-the-art methods on benchmark datasets.

2 Preliminaries

In this section, we first review the background of this paper and then introduce the notations used throughout this paper.

2.1 Background

In this part, we review the background of knowledge graph and milti-hop KGQA.

Knowledge Graph Given a set of entities $\mathcal{E}$, a set of relations $\mathcal{R}$, and a set of triplets $\mathcal{T} = \{(e_i, r_j, e_k)\} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, we define a knowledge graph $\mathcal{G}$ by $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$.

Multi-hop KGQA Given a knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$ and a natural language question q with its topic entity $e_{topic} \in \mathcal{E}$, the task of KGQA is to predict the answer e^* to question q by

$$e^* = \arg\max_{e_i \in \mathcal{E}} f(e_i),$$

where $f(e_i)$ is the score function that measures the plausibility of e_i being the correct answer. In multi-hop KGQA, the answers are not guaranteed to be direct neighbours of the topic entity in the given question. Therefore, it often requires multi-hop reasoning over KGs to find answers.

Bayesian Network A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). In a Bayesian network, the nodes represents random variables and the directed edges represent the conditional dependencies between random variables.

2.2 Notations

In this paper, we use lower-case letters e and r to represent an entity and a relation, respectively. The corresponding boldface letters $\mathbf{e}$ and $\mathbf{r}$ denotes the embeddings of e and r, respectively.

3 Related Work

In this section, we review related work for multi-hop KGQA and knowledge graph embeddings.

3.1 Multi-hop KBQA

Recent work in multi-hop KBQA can be divided into two categories: semantic parsing methods and information retrieval methods. Semantic parsing methods first parse the given question into an executable query, and then execute the query to locate answers. Information retrieval methods embeds questions and the knowledge graph into low-dimensional spaces, and then find answers based on question-answer semantic similarity. Our proposed DCRN belongs to information retrieval methods.

Key-Value Memory Network (KV-Mem) KV-Mem (Miller et al., 2016) is a variant of Memory Network (Weston et al., 2015), which performs reasoning based on a memory component, i.e., an array storing triplets in KGs. KV-Mem iteratively reads from the memory to update the question embedding, which is used to match correct answers.

Variational Reasoning Network (VRN) VRN (Zhang et al., 2018) proposes a variational framework for multi-hop KGQA. To identify answers, it computes the compatibility scores between the question type and the reasoning graph of each candidate. However, its performance is limited on the question that requires long reasoning paths to answer, due to the exponentially grown candidates.

GRAFT-Net GRAFT-Net (Sun et al., 2018) first extracts question-specific subgraph based on Personalized Page Rank (PPR), and then encode the subgraph with Graph Neural Networks (GNN) to identify answers. However, as described in Sun et al. (2019a), the extracted subgraphs are often too large and have a low recall for answer entities.

PullNet PullNet (Sun et al., 2019a) mitigates the problem of GraftNet with a trainable subgraph expansion strategy. It constructs question-specific subgraph starting from the list of entities mentioned in the question, and then iteratively "pulls" the relevant entities to expand the subgraph. However, it inevitably sacrifices the recall of answer entities

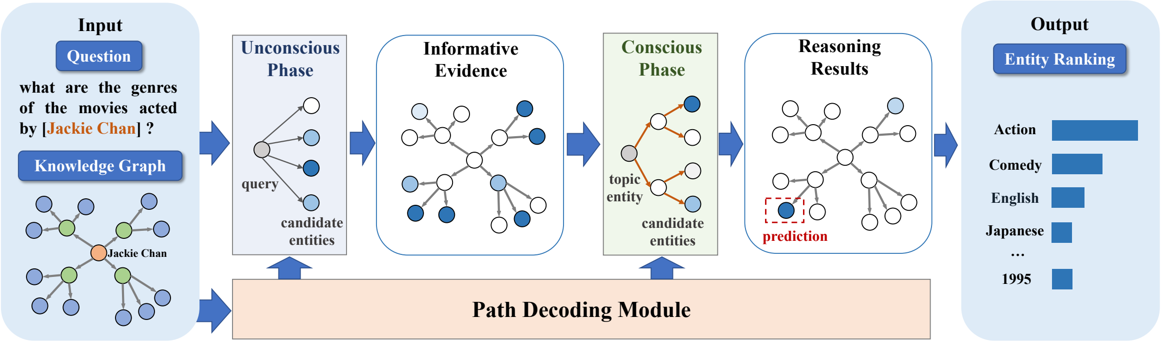

Figure 2: Overview of the proposed Deep Cognitive Reasoning Network (DCRN). DCRN consists of a Path Decoding Module and two phases—the unconscious phase and the conscious phase. Deeper color in coarse/fine-grained results denotes higher prediction score as the correct answer.

in exchange for small candidate entity sets, which limits its performance in practical usage.

EmbedKGQA EmbedKGQA (Saxena et al., 2020) models multi-hop KBQA as a link prediction task. It first embeds the given question into a latent relation embedding, and then exploits knowledge graph embedding techniques to identify answers.

3.2 Knowledge Graph Embedding in KGQA

Knowledge Graph Embedding (KGE) methods (Hitchcock, 1927; Trouillon et al., 2016; Sun et al., 2019b; Zhang et al., 2020a,b) aim to map entities and relations within KGs into distributed representations (vectors, matrices, etc.). These embeddings are often trained by the link prediction task, where the model is required to predict the missing head or tail entity of a triplet.

EmbedKGQA (Saxena et al., 2020) use ComplEx (Trouillon et al., 2016) to train knowledge graph embeddings, which represents entity and relation embeddings as vectors in complex spaces. For fair comparison with previous work including GRAFT-Net (Sun et al., 2018) and PullNet (Sun et al., 2019a), we use Canonical Polyadic (CP) decomposition (Hitchcock, 1927) to train knowledge graph embeddings in our proposed DCRN, which represents entity and relation embeddings as vectors in real spaces.

3.3 The Dual Process Theory

The *dual process theory* (Evans, 1984, 2003, 2008) is originally proposed in cognitive science. Inspired by this theory, researchers propose to mimic human cognition in various cognitive tasks. For example, Du et al. (2019) applies the theory to one-shot KG reasoning, and Ding et al. (2019) proposes a cognitive framework for multi-hop reasoning over documents. Different from these work, we focus on the task of multi-hop question answering over knowledge graphs.

4 Method

In this section, we introduce our proposed Deep Cognitive Reasoning Network (DCRN) for multi-hop KGQA. In section 4.1, we introduce the motivation and the overall architecture of DCRN. In Section 4.2, 4.3, and 4.4, we introduce the components of the proposed DCRN.

4.1 Motivation

For multi-hop questions, it is challenging to accurately identify answers from a large candidate set, of which the size grows exponentially with the number of reasoning steps. Existing approaches (Sun et al., 2018, 2019a) aim to reduce the size of candidate entity set by extracting question-specific subgraphs. However, these approaches often sacrifice the recall of answers in exchange for small candidate sets, which limits their performances in practical usage.

We take inspiration from the *dual process theory* (Evans, 1984, 2003, 2008) in cognitive science. Specifically, the theory suggests that humans accomplish cognitive tasks by first exploiting *fast intuition* to retrieve task-relevant evidence via an *unconscious process* (System 1), and then performing *sequential reasoning* based on the aforementioned evidence to derive answers via a *conscious process* (System 2).

Inspired by the *dual process theory* in cognitive science, we propose Deep Cognition Reasoning Network (DCRN) for multi-hop KGQA. The proposed DCRN consists of two phases. The first one is the *unconscious phase*, which can retrieve informative evidence from candidate entities by leveraging their semantic information. The second one is the *conscious phase*, which can accurately identify answers by performing sequential reasoning according to the graph structure on the retrieved evidence from the first phase.

The overall architecture of DCRN is shown in Figure 2. In DCRN, the basic module is the Path Decoding Module, based on which are the two phases—unconscious phase and conscious phase.

4.2 Path Decoding Module

, The Path Decoding Module is the basic component of DCRN. As multi-hop KGQA requires multi-hop reasoning to arrive at answer entities, we decode the reasoning path information from the question in this module.

Specifically, we adopt an RNN-based encoder-decoder structure, which first encodes the question into a hidden representation, and then decodes this representation to obtain the reasoning path information, i.e., the scores of each relation at each reasoning step. These scores will be used in the unconscious and conscious phase.

First, we encode the given question q with an RNN to obtain its latent representation $\mathbf{q} \in \mathbb{R}^d$.

$$\mathbf{q} = \text{RNN-Encoder}(q).$$

Then, we decode this representation $\mathbf{q}$ to obtain reasoning path information. We illustrate this process in Figure 3. At each step of decoding, the decoder predicts the scores of each relation. The predictions at step t is the input of the decoder at step $t + 1$.

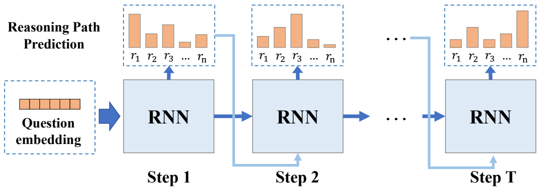

Figure 3: Illustration of the Path Decoding Module.

At step t, given the hidden state $\mathbf{h}^{(t-1)}$ of the previous iteration and the input $\mathbf{i}^{(t)}$, the RNN decoder outputs the updated hidden state $\mathbf{h}^{(t)}$ by

$$\mathbf{h}^{(t)} = \text{RNN-Decoder}(\mathbf{h}^{(t-1)}, \mathbf{i}^{(t)}),$$

where the initial hidden state $\mathbf{h}^{(0)}$ is initialized as the question embedding $\mathbf{q}$, and the initial input $i^{(0)}$ is a zero vector. Then, we compute the output of step t by

$$\mathbf{o}^{(t)} = \sum_i \alpha_i^{(t)} \mathbf{r}_i,$$

where the weights $\alpha_i^{(t)}$ is computed as

$$\alpha_i^{(t)} = \frac{\exp\left(f_{rel}^{(t)}(r_i)\right)}{\sum_j \exp\left(f_{rel}^{(t)}(r_j)\right)}.$$

Note that $f_{rel}^{(t)}(r_i)$ denotes the scores of each relation at step t, which is computed by

$$f_{rel}^{(t)}(r_i) = \mathbf{h}^{(t)} \mathbf{r}_i^\top.$$

Then, the output of step t will be the input of step $t + 1$. That is, $\mathbf{i}^{(t+1)} = \mathbf{o}^{(t)}$.

4.3 The Unconscious Phase

The unconscious phase corresponds to the *unconscious process* (System 1) in the *dual process theory* from cognitive science. In this phase, we retrieve informative evidence from candidate entities by leveraging their semantic information.

The *evidence* refers to sketched results that predicts which candidates are most likely to be correct answers. We expect the retrieved evidence to effectively filter out those candidate entities that are irrelevant to the given question.

To achieve this, we perform semantic matching between the given question and each candidate entity. The semantic matching scores $f_s(e)$ of candidate entity e is

$$f_s(e) = \overline{\mathbf{q}} \mathbf{e}^\top,$$

where $\overline{\mathbf{q}} \in \mathbb{R}^{1\times d}$ is the query embedding obtained based on the given question, and $\mathbf{e} \in \mathbb{R}^{1\times d}$ is the embedding of entity e.

In our model, the entity embedding $\mathbf{e} \in \mathbb{R}^d$ is pretrained by the CP (Hitchcock, 1927) model. Therefore, the key of the unconscious phase is to design informative query representation $\overline{\mathbf{q}} \in \mathbb{R}^d$.

To design informative query representations, we take inspiration from PTransE (Lin et al., 2015), which extends knowledge graph embedding to relation paths. In PTransE, if a relation path $e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \dots \xrightarrow{r_{n-1}} e_n$ holds, then PTransE optimize the following objective:

$$\mathbf{e}_1 \circ \mathbf{r}_1 \circ \dots \circ \mathbf{r}_{n-1} = \mathbf{e}_n,$$

where $\circ$ is an composition operation, and it can be addition, element-wise multiplication, RNN, etc. This objective can be considered as the semantic matching between the query $\mathbf{e}_1 \circ \mathbf{r}_1 \circ ... \circ \mathbf{r}_{n-1}$ and its target $\mathbf{e}_n$.

Similarly, we encode the query representation $\overline{\mathbf{q}}$ as follows. First, the start entity of the reasoning path is the topic e_{topic} in the given question. Second, recall that we decode reasoning path information in the Path Decoding Module, in which the output at step t (i.e., $\mathbf{o}^{(t)}$) represents the weighted sum of relation embeddings. Therefore, we represent the query embedding as

$$\overline{\mathbf{q}} = \mathbf{e}_{topic} \circ \mathbf{o}^{(1)} \circ ... \circ \mathbf{o}^{(T)},$$

where $\circ$ denotes element-wise multiplication in this formula, and T denotes the number of steps in the Path Decoding Module (i.e., the number of reasoning steps).

4.4 The Conscious Phase

The conscious phase corresponds to the *conscious process* (System 2) in the *dual process theory* from cognitive science. In this phase, we accurately identify answers by performing sequential reasoning according to the graph structure on the retrieved evidence from the unconscious phase.

To model the sequential reasoning, we take inspiration from the *consciousness prior* (Bengio, 2017). It suggests that the conscious process only refers to a few variables at a time, which can be modeled as *factor graphs*, a form of knowledge representation which is factored into pieces involving a few variables at a time.

In this work, we perform sequential reasoning based on Bayesian networks, which can be seen as a type of factor graphs. First, we build question-specific Bayesian networks from the given KG, in which we view the predictions of entities as random variables and relations as the relational dependencies between them. Second, we perform marginal inference on the Bayesian networks to predict the probability of each candidate entity as a correct answer.

4.4.1 Building Bayesian Networks

We build question-specific Bayesian networks from the given KG with the following two steps. First, we perform graph pruning on the KG to obtain a directed acyclic graph (DAG). Second, we transform the DAG into a Bayesian network.

Given a knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$ and a question q with a topic entity $e_{topic} \in \mathcal{E}$, we prune G to obtain a directed acyclic graph (DAG) by applying the breadth-first search (BFS) algorithm starting from e_{topic}. Specifically, we only keep the visited edges during searching, and remove the unvisited edges. We illustrate this process in Figure 4, in which we perform two-step BFS starting from the topic entity, and prunes the unvisited edges. Note that we add inverse relations r^{-1} for each relation r in KGs following previous work (Sun et al., 2018, 2019a). That is, if (e_i, r_j, e_k) is a valid triplet, then (e_k, r_j^{-1}, e_i) is also valid.

The reasons to perform the graph pruning is two-fold. First, the number of potential reasoning paths from e_{topic} to an arbitrary candidate entity e in the KG can be extremely large. Therefore, we apply graph pruning to reduce the search space, and only keep the shortest paths. Second, Bayesian Network is required to be a directed acyclic graph (DAG). Therefore, the graph pruning procedure only removes redundant edges, and the answer entities are guaranteed to be within the candidate set.

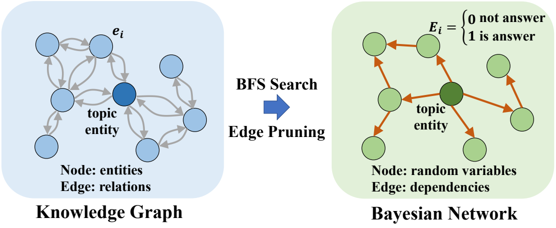

Figure 4: Illustration of question-specific Bayesian networks built from KGs. In KGs, we add an inverse relation r^{-1} for each relation r. That is, if a triplet (e_i, r, e_j) exists, then (e_j, r^{-1}, e_i) also exists.

We use $\hat{\mathcal{G}}(e_{topic})$ to denote the pruned graph given the topic entity e_{topic}. Then, we have the following proposition.

Proposition 1. *The pruned graph $\hat{\mathcal{G}}(e_{topic})$ is a directed acyclic graph (DAG).*

For detailed proof, please refer to the appendix. According to the properties of BFS, if $\mathcal{G}$ is connected, then there exists a path in the pruned graph $\hat{\mathcal{G}}(e_{topic})$ that starts from e_{topic} and ends with e. Furthermore, this path must be the shortest one among all paths in $\mathcal{G}$ that connects e_{topic} and e.

We then introduce how this DAG corresponds $\hat{\mathcal{G}}(e_{topic})$ to a Bayesian network. The transformed Bayesian Network $\mathcal{B}(_{topic})$ shares the same graph structure with $\hat{\mathcal{G}}(_{topic})$, but the definitions on nodes and edges are different. In Table 1, we illustrate the

Table 1: The relationship between the DAG and the corresponding Bayesian Network.

	DAG $\hat{\mathcal{G}}(e_{topic})$	**Bayesian Network** $\mathcal{B}(e_{topic})$
Nodes	entity e_i	random variable $X_{e_i} = \{0, 1\}$
Edges	relation r_j between e_i and e_k	conditional dependencies between X_{e_i} and X_{e_k}

relationship between the DAG and the corresponding Bayesian network.

Each entity e in $\hat{\mathcal{G}}(e_{topic})$ corresponds to a random variable $X_e = \{0, 1\}$ in $\mathcal{B}(e_{topic})$, where $X_e = \{0, 1\}$ represents the prediction of a candidate entity e. Given a question q, $X_e = 0$ denotes that e is an incorrect answer and $X_e = 1$ denotes that e is a correct answer. In $\hat{\mathcal{G}}(_{topic})$, each relation r connecting entity e_i and e_j corresponds to an directed edge connecting X_{e_i} and X_{e_j} in $\mathcal{B}(_{topic})$, which denotes the dependencies between them.

4.4.2 Bayesian Reasoning

Based on the Bayesian network $\mathcal{B}(_{topic})$, we can make marginal inferences to predict whether an entity e is a correct answer, which can be represented in a probabilistic way:

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}),$$

where $\mathcal{G}$ is the KG, q is the given question, and e_{topic} is the topic entity. To calculate this marginal probability, we have the following proposition.

Proposition 2. *The marginal probability* $\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic})$ *that predicts a candidate entity e can be calculated via variable elimination:*

$$\begin{aligned}&\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic})\\&= \overline{\mathbb{P}}(X_e = 1) \prod_{e' \in pa(e)} \mathbb{P}(X_{e'} = 0|\mathcal{G}, q, e_{topic}),\end{aligned}$$

where $pa(e)$ *denotes the set of parent nodes of e, i.e., the nodes that have edges directing at e. The first component* $\overline{\mathbb{P}}(X_e = 1)$ *is the abbreviation of*

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}, X_{pa(e)} = 0).$$

For detailed proofs, please refer to the appendix. The marginal probability $\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic})$ is the product of two components. The first component denotes the probability that entity e is an answer given that all e's parent entities $\text{pa}(e)$ are incorrect. The second component $\prod_{e' \in \text{pa}(X_e)} \mathbb{P}(X_{e'} = 0|\mathcal{G}, q, e_{topic})$ denotes the product of the predictions of X_e's parent nodes. Note that we assume $\mathbb{P}(X_{e'} = 0|\mathcal{G}, q, e_{topic}) = 1$ when computing $\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}$ in our implementation for convenience of computation.

We model the first component as follows.

$$\begin{aligned}&\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}, X_{\text{pa}(e)} = 0)\\&= \text{sigmoid}(g(f_s(e), f_b(e))),\end{aligned}$$

where $f_s(e)$ is the evidence provided by the unconscious phase, $f_b(e)$ represents the score computed in the Bayesian network. $g(\cdot, \cdot)$ is a function for combining the two scores, and we choose $g(x, y) = x + y$ in this work. The score $f_b(e)$ is defined as follows:

$$f_b(e) = \sum_{\substack{e' \in \text{pa}(e),\\ (e', r, e) \in \mathcal{T}}} \alpha_r^{(t)} f_b(e') + f_{rel}^{(t)}(r),$$

where $f_{rel}^{(t)}(r)$ represents the prediction score of relation r at reasoning step t in the Path Decoding Module, $\alpha_r^{(t)}$ is defined in section **??**. Note that t also denotes the topological distance between e_{topic} and e, i.e., the required reasoning steps from e_{topic} to e. We initialize the score for the topic entity to zero. That is, $f_b(e_{topic}) = 0$.

The conscious phase is different from previous multi-hop reasoning approaches (Zhang et al., 2018; Sun et al., 2018, 2019a) in the following two aspects. First, we model the reasoning process in a probabilistic perspective with Bayesian networks, while previous works often apply GNN for reasoning. Second, the conscious phase propagates scalar scores along the paths for multi-hop reasoning, while previous works often propagates embeddings with GNNs.

4.5 Loss Function

We use the Binary Cross Entropy Loss for training. Specifically, given a question q, the loss $\mathcal{L}$ is computed as

$$\mathcal{L} = \frac{1}{|\mathcal{E}|}(\sum_{e \in \mathcal{A}} \log p(e) + \sum_{e' \in \mathcal{E}/\mathcal{A}} \log(1 - p(e'))),$$

where $\mathcal{E}$ is the set of entities, $\mathcal{A}$ is the set of correct answers, and $p(e) = \mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic})$.

5 Experiments

This section is organized as follows. In Section 5.1, we introduce experimental settings in detail. In Section 5.2, we show the effectiveness of our model on benchmark datasets. In Section 5.3, we conduct ablation studies and analysis.

5.1 Experimental Settings

In this part, we introduce the benchmark datasets and the protocols for training and evaluation.

5.1.1 Datasets

We conduct experiments on two public datasets—WebQuestionSP (Yih et al., 2015) and MetaQA (Zhang et al., 2018), which have been divided into training, validation, and testing set by previous works. The statistics are shown in Table 2.

Table 2: The statistics of benchmark datasets. The second to fourth columns show the number of entities, relations, and triplets, respectively. The fifth to seventh columns show the size of training, validation, and testing dataset, respectively.

Dataset	Entity	Relation	Triplet	Train	Valid	Test
WebQuestionSP	601,445	567	1,261,849	2,848	250	1,639
MetaQA 1-hop	43,233	9	134,741	96,106	9,992	9,947
MetaQA 2-hop	43,233	9	134,741	118,980	14,872	14,872
MetaQA 3-hop	43,233	9	134,741	114,196	14,274	14,274

WebQuestionSP WebQuestionSP is a small dataset containing 4,737 questions. Those questions are 1-hop or 2-hop questions that can be answered with the Freebase (Bollacker et al., 2008) knowledge graph. Note that WebQuestionSP mainly consists of 1-hop questions, and only 0.5% of the questions are 2-hop.

MetaQA MetaQA is a large dataset containing over 400k questions in the movie domain. It is split into 1-hop, 2-hop, and 3-hop questions. Following previous work (Sun et al., 2018, 2019a; Saxena et al., 2020), we use the “vanilla” version of the dataset. On MetaQA, we evaluate our model under two settings: “full” setting and “half” setting. In the “full” setting, we use the vanilla knowledge graph for training. In the “half” setting, we follow previous work (Saxena et al., 2020) to randomly drop 50% of triplets in the knowledge graph.

5.1.2 Evaluation Protocols

Following previous work (Sun et al., 2018, 2019a), we use Hits at N (H@N) to evaluate model performance. For each given question, we rank the candidates in descending order according to their scores, and compute the percentage of correct answers that ranks at top N.

5.1.3 Training Protocols

We choose Adam (Kingma and Ba, 2015) as the optimizer, and use grid search to find the best hyperparameters based on the model performance on the validation datasets. For the details of hyperparameter selection, please refer to the appendix.

Following previous work (Sun et al., 2018, 2019a), we use GloVe (Pennington et al., 2014) as word embeddings, and use bidirectional LSTM as the encoder. We also use CP (Hitchcock, 1927) to train entity and relation embeddings.

5.1.4 Candidate Set Generation Protocol

In the ”full” setting, the candidate set of a n-hop question q consists of all entities that are within the n-hop of the topic entity of q. In the ’half’ setting, the candidate set of any question consists of all entities in the KG. Therefore, the answers are guaranteed to be included in the candidate sets, and the recall of answers is 1.0.

5.1.5 Hyperparameters

We use grid search to find the best hyperparameters. Specifically, we search the learning rate in $\{0.1, 0.01, 0.001\}$, and dropout rate in $\{0.1, 0.2, 0.3\}$. The optimal configurations of DCRN is that learning rate $= 0.01$ and dropout rate $= 0.2$. For fair comparison with previous work (Sun et al., 2018, 2019a; Saxena et al., 2020), we set the embedding size to 300. When training the knowledge graph embeddings with CP (Hitchcock, 1927), we search the learning rate in $\{0.1, 0.01, 0.001\}$, and the optimal configuration is learning rate $= 0.1$. We choose the values hyperparameter T as follows. On WebQuestionSP, we set $T = 2$. On MetaQA with ”full” setting, we set $T = t$ for t-hop questions. On MetaQA with ”half” setting, we set $T = 4$.

5.2 Main Results

In Table 3, we show the results of our proposed DCRN on WebQuestionSP and MetaQA datasets. Overall, our model significantly outperforms state-of-the-art models on benchmark datasets.

WebQuestionSP is a small dataset but it uses a large-scale KG, which is a subset of Freebase. This dataset follows an inductive setting—some entities in the test set have not appeared in the training set. Experiments demonstrate that our DCRN achieves 67.8 on H@1, which outperforms GraftNet and KV-Mem, and performs comparatively against previous state-of-the-art PullNet.

MetaQA is a large dataset consisting of 1 to 3-hop questions. Overall, our DCRN achieves state-of-the-art on all the three subdatasets. On MetaQA 1-hop and 2-hop, although some previous methods exhibits satisfying performance, they fail to achieve

Table 3: Evaluation results (H@1) of the proposed DCRN and previous state-of-the art methods on WebQuestionSP and MetaQA datasets.

Methods	WebQSP	MetaQA 1-hop	MetaQA 2-hop	MetaQA 3-hop
KV-Mem (Miller et al., 2016)	46.7	96.2	82.7	48.9
VRN (Zhang et al., 2018)	-	**97.5**	89.9	62.5
GraftNet (Sun et al., 2018)	66.4	97.0	94.8	77.7
PullNet (Sun et al., 2019a)	**68.1**	97.0	**99.9**	91.4
EmbedKGQA (Saxena et al., 2020)	66.6	**97.5**	98.8	94.8
DCRN (Ours)	67.8	**97.5**	**99.9**	**99.3**

consistent performances on both datasets. For example, PullNet only achieves 97.0 on MetaQA 1-hop, and EmbedKGQA only achieves 98.8 on MetaQA 2-hop. Different from previous methods, our DCRN achieves state-of-the-art on both MetaQA 1-hop and 2-hop.

The questions in the MetaQA 3-hop dataset are more difficult to answer compared to those in MetaQA 1-hop and 2-hop, as they require longer reasoning paths to find answers. However, experiments demonstrate that our model achieves 99.3 on H@1, which significantly outperforms previous state-of-the-arts. Specifically, it gains 7.9 against PullNet and 4.5 against EmbedKGQA. The results on MetaQA 3-hop illustrates the effectiveness of our model on answering questions that require long reasoning paths.

We also conduct experiments in "half" setting. In this setting, 50% of triplets are dropped, making it more challenging to accurately identify answers. The results are shown in Table 4. Experiments demonstrate our model achieves state-of-the art on all subsets of MetaQA.

Table 4: The evaluation results (H@1) of DCRN and previsou state-of-the art methods on MetaQA datasets under the "**half**" setting. In this setting, 50% triplets are randomly dropped.

Methods	MetaQA 1-hop	MetaQA 2-hop	MetaQA 3-hop
KV-Mem	63.6	41.8	37.6
GraftNet	64.0	52.6	59.2
PullNet	65.1	52.1	59.7
EmbedKGQA	83.9	91.8	70.3
DCRN (Ours)	**88.5**	**91.9**	**72.5**

5.3 Analysis

In this part, we conduct analysis on our model. In Section 5.3.1, we conduct ablation studies on the two phases in DCRN. In Section 5.3.2, we conduct a case study to illustrate the two-phase strategy of the proposed DCRN.

5.3.1 Ablation Studies on the Two Phases

In Table 5, we conduct ablations studies to show the performances of the two phases in DCRN.

Table 5: Ablation results (H@1) of the two modules on WebQuestionSP and MetaQA datasets. "Unconscious" and "Conscious" denote the unconscious phase and the conscious phase, respectively.

Methods	WebQSP	MetaQA 1-hop	MetaQA 2-hop	MetaQA 3-hop
DCRN	**67.8**	**97.5**	**99.9**	**99.3**
Unconscious Phase	60.8	96.9	92.1	68.4
Conscious Phase	47.2	97.4	93.8	37.2

Overall, the experiments show that both two phases are indispensable in our model. The reason is that the unconscious and the conscious phase are designed to better exploit node-level and path-level features, respectively. Both levels of features are critical to the accurate answer identification. Therefore, the cooperation of the two phases brings significant improvements to the performance, as shown in Table 5. On MetaQA 1-hop and 2-hop, both two phases achieves satisfying performances, as the number of candidate entities is relatively small. Furthermore, the conscious phase outperforms the unconscious phase. This is because the unconscious phase exploits the coarse-grained semantic of entities, while the conscious phase considers the fine-grained relational dependencies between entities. Therefore, on small candidate entity sets, the conscious phase could make more accurate predictions.

On MetaQA 3-hop, the unconscious phase outperforms conscious phase. This is because 3-hop questions usually have large candidate entity set, and the errors can propagate along reasoning paths. Therefore, to make accurate predictions, DCRN requires the unconscious phase to softly filter out irrelevant candidates. Experiments demonstrate that, by considering both phases, DCRN achieves 99.3 on H@1.

We further compare between the unconscious phase of DCRN and EmbedKGQA (Saxena et al., 2020). EmbedKGQA consists of two parts—knowledge graph embedding and relation matching. The former part use the question representation as latent relation embedding. Different from EmbedKGQA, the unconscious phase in DCRN decode a question into relation paths. To illustrate the effectiveness of the unconscious phase, we compare it with EmbedKGQA (w/o relation matching), and the results are shown in Table 6.

Table 6: Comparisons between EmbedKGQA (w/o relation matching) and the unconscious phase in the proposed DCRN on the MetaQA datasets.

Methods	**MetaQA 1-hop**	**MetaQA 2-hop**	**MetaQA 3-hop**
EmbedKGQA (w/o relation matching)	94.7	86.5	67.2
Unconscious Phase	**96.9**	**92.1**	**68.4**

Note that EmbedKGQA (Saxena et al., 2020) use RoBERTa (Liu et al., 2019) for word embeddings and ComplEx (Trouillon et al., 2016) for entity embeddings. For fair comparison with previous work including GRAFT-Net (Sun et al., 2018) and PullNet (Sun et al., 2019a), we use GloVe (Pennington et al., 2014) for word embeddings and CP (Hitchcock, 1927) for entity embeddings, and we reimplement EmbedKGQA (w/o relation matching) under our settings.

Experiments demonstrate that the unconscious phase outperforms EmbedKGQA (w/o relation matching) on all the three datasets of MetaQA, illustrating the effectiveness of our design on the query representation in the unconscious phase.

5.3.2 Case Study

In this part, we conduct a case study to illustrate the effectiveness of the two-phase strategy in DCRN. In Figure 5, we show the predictions made by DCRN on a 2-hop question *who is listed as screenwriter of John Derek acted films?*. This question is taken from the test set of MetaQA 2-hop.

The figure on the left shows the predictions of the unconscious phase. It shows that the unconscious phase successfully filters out the candidates that are unlikely to be correct answers. The predictions made by the unconscious phase provide informative evidence for the subsequent conscious phase. The figure on the right shows the predictions of the conscious phase. Based on the retrieved evidence, the conscious phase successfully ranks the correct answers in the first place.

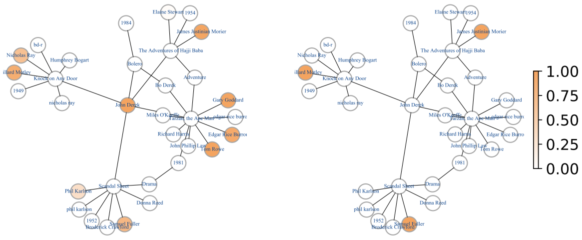

Figure 5: Illustration of the predictions made by the unconscious phase (**left**) and the conscious phase (**right**) to the question *who is listed as screenwriter of John Derek acted films?*. We exhibit the 2-hop subgraph of the topic entity *John Derek*. Deeper color for an entity indicates higher prediction score as a correct answer.

6 Conclusion

Multi-hop question answering over knowledge graphs aims to answer questions by multi-hop reasoning over knowledge graphs to find answers. In this work, we propose a novel Deep Cognitive Reasoning Network (DCRN), which is inspired by the dual process theory in cognitive science. DCRN can accurately identify answers with two phases—unconscious phase and conscious phase. Experiments demonstrate that our model outperforms state-of-the-art methods on benchmark datasets.

Acknowledgments

We would like to thank all the anonymous reviewers for their insightful comments. This work was supported in part by NSFC (61822604, 61836006, U19B2026), Fundamental Research Funds for the Central Universities No. WK3490000004, and National Key Research and Development Program of China under Grant 2017YFB1002203.

References

Yoshua Bengio. 2017. The consciousness prior. *arXiv e-prints*, 1709.08568.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. SIGMOD '08, page 1247–1250.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703.

Zhengxiao Du, Chang Zhou, Ming Ding, Hongxia Yang, and Jie Tang. 2019. Cognitive knowledge graph reasoning for one-shot relational learning. *arXiv preprint arXiv:1906.05489*.

Jonathan Evans. 2003. In two minds: Dual-process accounts of reasoning. *Trends in cognitive sciences*, 7:454–9.

Jonathan Evans. 2008. Dual-processing accounts of reasoning, judgment, and social cognition. *Annual review of psychology*, 59:255–78.

Jonathan St B. T. Evans. 1984. Heuristic and analytic processes in reasoning. *British Journal of Psychology*, 75(4):451–468.

Bin Fu, Yunqi Qiu, Chengguang Tang, Y. Li, H. Yu, and J. Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. *ArXiv*, abs/2007.13069.

F. L. Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys*, 6(1):164–189.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, Guwei Jin, and Wei Chu. 2017. Alime assist: An intelligent assistant for creating an innovative e-commerce experience. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2495–2498.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019a. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Yingxu Wang, Dong Liu, and Ying Wang. 2003. Discovering the capacity of human memory. *Brain and Mind*, 4(2):189–198.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1321–1331.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhanqiu Zhang, Jianyu Cai, and Jie Wang. 2020a. Duality-induced regularizer for tensor factorization based knowledge graph completion. In *Advances in Neural Information Processing Systems*, volume 33, pages 21604–21615.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020b. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3065–3072. AAAI Press.

Appendix

A Proofs of the Propositions

Proposition 1. *The pruned graph $\hat{\mathcal{G}}(e_{topic})$ is a directed acyclic graph (DAG).*

Proof. *The pruned graph $\hat{\mathcal{G}}(e_{topic})$ is obtained by employing the Breadth-First Search (BFS) algorithm starting from e_{topic}. During the search process, we only keep the visited edges.*

First, it is clear that $\hat{\mathcal{G}}(e_{topic})$ is a directed graph. Second, we prove that $\hat{\mathcal{G}}(e_{topic})$ is acyclic. During the BFS search, the nodes are classified into several categories according to their topological distance to e_{topic}. After the pruning, each edge in $\hat{\mathcal{G}}(e_{topic})$ must connect a node with topological distance n to e_{topic} and a node with distance $(n+1)$ to e_{topic}. Therefore, there is no loop in $\hat{\mathcal{G}}(e_{topic})$.

Proposition 2. *The marginal probability $\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic})$ that predicts a candidate entity e can be calculated via variable elimination:*

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}) \\ = \overline{\mathbb{P}}(X_e = 1) \prod_{e' \in pa(e)} \mathbb{P}(X_{e'} = 0|\mathcal{G}, q, e_{topic}),$$

where $pa(e)$ denotes the set of parent nodes of e, i.e., the nodes that have edges directing at e. The first component $\overline{\mathbb{P}}(X_e = 1)$ is the abbreviation of

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}, X_{pa(e)} = 0).$$

Proof. *First, the marginal probability is defined as*

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}) = \\ \sum_{e_1, e_2, \ldots \in \mathcal{E}/e} \mathbb{P}(X_e = 1, X_{e_1}, X_{e_2}, \ldots|\mathcal{G}, q, e_{topic}),$$

where $\mathbb{P}(X_e = 1, X_{e_1}, X_{e_2}, \ldots|\mathcal{G}, q, e_{topic})$ is the joint probability of variables $X_{e_1}, \ldots X_{e_{|\mathcal{E}|}}$.

By the definition of Bayesian networks, the joint probability is factorized into several components:

$$\mathbb{P}(X_e = 1, X_{e_1}, X_{e_2}, \ldots|\mathcal{G}, q, e_{topic}) = \\ \prod_{v \in \mathcal{E}} \mathbb{P}(X_v|X_{pa(v)}, \mathcal{G}, q, e_{topic}),$$

Therefore, the marginal probability is represented as follows:

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}) = \\ \sum_{v \in \mathcal{E}/e} \prod_{v \in \mathcal{E}} \mathbb{P}(X_v|X_{pa(v)}, \mathcal{G}, q, e_{topic}).$$

We then perform variable elimination, which eliminates the variables other than X_e and $X_{pa(e)}$. The results are as follows:

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}) = \\ \sum_{e' \in pa(e)} \hat{\mathbb{P}}(X_e = 1)\mathbb{P}(X_{e'}|\mathcal{G}, q, e_{topic}),$$

where the notation $\hat{\mathbb{P}}(X_e = 1)$ denotes

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}, X_{pa(e)}).$$

We then make the following assumption: if any parent of e is an answer, then e is not an answer. This assumption represents the fact that each question corresponds to a unique reasoning path.

Following this assumption, we have that

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}, X_{pa(e)}) = 0,$$

if there exists $e' \in pa(e)$ such that $X_{e'} = 1$. Therefore, we have the following conclusion:

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}) = \\ \overline{\mathbb{P}}(X_e = 1) \prod_{e' \in pa(e)} \mathbb{P}(X_{e'} = 0|\mathcal{G}, q, e_{topic}),$$

where $pa(e)$ denotes the set of parent nodes of e, i.e., the nodes that have edges directing at e. The first component $\overline{\mathbb{P}}(X_e = 1)$ is the abbreviation of

$$\mathbb{P}(X_e = 1|\mathcal{G}, q, e_{topic}, X_{pa(e)} = 0).$$

GoG: Relation-aware Graph-over-Graph Network for Visual Dialog

Feilong Chen, Xiuyi Chen, Fandong Meng, Peng Li, Jie Zhou
Pattern Recognition Center, WeChat AI, Tencent Inc, Beijing, China
{ivess.chan,hugheren.chan}@gmail.com
{fandongmeng,patrickpli,withtomzhou}@tencent.com

Abstract

Visual dialog, which aims to hold a meaningful conversation with humans about a given image, is a challenging task that requires models to reason the complex dependencies among visual content, dialog history, and current questions. Graph neural networks are recently applied to model the implicit relations between objects in an image or dialog. However, they neglect the importance of 1) coreference relations among dialog history and dependency relations between words for the question representation; and 2) the representation of the image based on the fully represented question. Therefore, we propose a novel relation-aware graph-over-graph network (GoG) for visual dialog. Specifically, GoG consists of three sequential graphs: 1) H-Graph, which aims to capture coreference relations among dialog history; 2) History-aware Q-Graph, which aims to fully understand the question through capturing dependency relations between words based on coreference resolution on the dialog history; and 3) Question-aware I-Graph, which aims to capture the relations between objects in an image based on fully question representation. As an additional feature representation module, we add GoG to the existing visual dialogue model. Experimental results show that our model outperforms the strong baseline in both generative and discriminative settings by a significant margin.

1 Introduction

Vision-language tasks have drawn more attention with the development of multi-modal natural language processing (Baltrušaitis et al., 2018; Chen et al., 2020b, 2019), such as image captioning (Xu et al., 2015; Anderson et al., 2016, 2018; Cornia et al., 2020; Ghanimifard and Dobnik, 2019), visual question answering (Ren et al., 2015a; Gao et al., 2015; Lu et al., 2016; Anderson et al., 2018; Li et al., 2019; Huang et al., 2020) and visual dialog (Das et al., 2017; Kottur et al., 2018; Agarwal et al., 2020; Wang et al., 2020; Qi et al., 2020). Relations in these tasks are significant for reasoning and understanding the textual and visual information. Specifically, visual dialog, which aims to hold a meaningful conversation with a human about a given image, is a challenging task that requires models to reason complex relations among visual content, dialog history, and current questions.

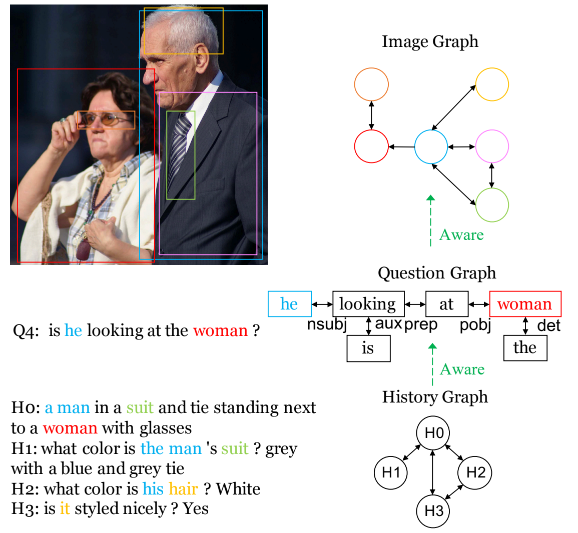

Figure 1: An example of complex relationships in visual dialog. The color in the text corresponds to the same color box in the image, which indicates the same entity. In visual dialog, we construct three graphs. History graph (H-Graph): arrows indicate the coreference relations between QA pairs in dialog history. Question graph (Q-Graph): arrows indicate dependency relations of the question. Image graph (I-Graph): arrows indicate spatial relations between objects in an image. The dark green dotted line indicates the bottom graph affects the upper graph.

Kinds of attention mechanisms are served as the backbone of previous mainstream approaches (Lu et al., 2017; Wu et al., 2018; Kottur et al., 2018;

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 230–243
August 1–6, 2021. ©2021 Association for Computational Linguistics

Gan et al., 2019; Guo et al., 2019b), following Das et al. 2017. HCAIE (Lu et al., 2017) provides a history-conditioned image attentive encoder to represent the question, the question-attended history, and the attended image. CoAtt (Wu et al., 2018) provides a sequential co-attention encoder to realize that each input feature is co-attended by the other two features in a sequential fashion. ReDAN (Gan et al., 2019) and DMAM (Chen et al., 2020a) use multi-step reasoning based on dual attention to answer a series of questions about an image. DAN (Guo et al., 2019b), MCAN (Agarwal et al., 2020) and LTMI (Nguyen et al., 2020) utilize multi-head attention mechanisms to manage multi-modal intersection. However, these approaches tend to catch only the most discriminative information, ignoring other rich complementary clues, such as relations between objects in an image.

Recent visual dialog studies (Zheng et al., 2019; Schwartz et al., 2019; Jiang et al., 2020b; Guo et al., 2020; Jiang et al., 2020a) explore the higher-level semantic representation of images or dialog history, notably with graph-based structures for modeling the image or dialog history. Although graph-based structures have been considered, these graph-based models lack explicitly capturing complex relations within visual content or textual contexts, and relations between them. As shown in Figure 1, there are complex relations such as coreference relations among dialog history, dependency relations between words in the question, spatial relations between objects in the image. For example, to answer the question Q4 *"is he looking at the woman ?"*, we firstly need to reason in dialog history to know who *"he"* is, then further understand the intention of the question with the understood of history and syntax of questions, and finally know clearly the spatial location and relation about *"the man"* and *"the woman"* in the image based on fully question understanding. How to 1) understand the coreference among history, 2) understand the intention of the question with its syntax and history, 3) understand the image with fully question understanding are worth exploring.

Therefore, in this paper, we propose a novel relation-aware graph-over-graph network (GoG) for visual dialog. Specifically, GoG consists of three sequential graphs: 1) H-Graph, which aims to capture coreference relations among dialog history; 2) History-aware Q-Graph, which aims to fully represent the question through capturing dependency relations between words based on coreference resolution on the dialog history; and 3) Question-aware I-Graph, which aims to capture the relations between objects in an image on the basis of fully question representation. As an additional feature representation module, we add GoG to the strong visual dialogue model LTMI (Nguyen et al., 2020). We test the effectiveness of our proposed model on two large-scale datasets: VisDial v0.9 and v1.0 (Das et al., 2017). Both automatic and manual evaluations show that our approach can be used to improve the prior strong models. The contributions of this work are summarized as follows:

- We explore how to construct complex explicit relations in visual dialog, i.e., coreference relations among dialog history, dependency relations between words in the question, spatial relations between objects in the image.
- We propose a novel relation-aware graph-over-graph network to reason relations within and among different graphs to obtain a high-level representation of multi-modal information and use it to generate a visually and contextually coherent response.
- We conduct extensive experiments and ablation studies on two large-scale datasets VisDial v0.9 and v1.0. Experimental results show that our GoG model can be used to improve the previous strong visual dialog model in both generative and discriminative settings.

2 Relation-aware Graph-over-Graph Network

2.1 Preliminary

Following Das et al. (Das et al., 2017), a visual dialog agent is given three inputs, i.e., an image I, dialog history (the caption and question-answer pairs) till round $t-1$: $H = (\underbrace{Cap}_{H_0}, \underbrace{(Q_1, A_1)}_{H_1}, \cdots, \underbrace{(Q_{t-1}, A_{t-1})}_{H_{t-1}})$ and the current question Q_t at round t, where Cap is the caption describing the image taken as H_0, and $H_1, \ldots, H_{t-1}$ are concatenations of question-answer pairs. The goal of the visual dialog agent is to generate a response A_t to the question Q_t.

As shown in Figure 2, our relation-aware graph-over-graph network (GoG) firstly takes the image, the dialog history, and the question as inputs and represent them using Faster RCNN (Ren et al.,

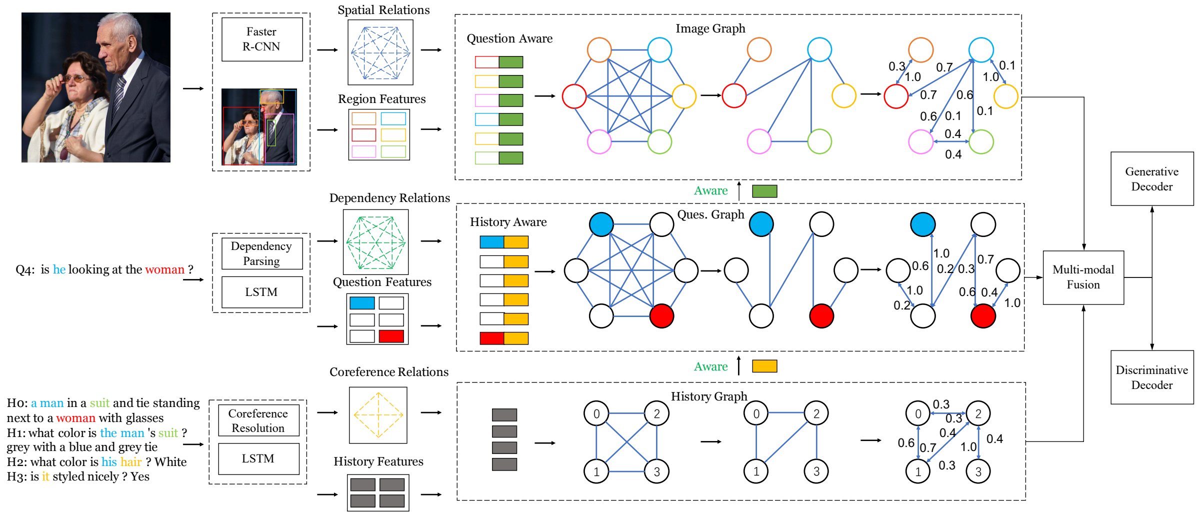

Figure 2: Framework of our Relation-aware Graph-over-Graph Network.

2015b) and LSTM (Hochreiter and Schmidhuber, 1997). Secondly, GoG constructs the history graph, the history-aware question graph, and the question-aware image graph. Thirdly, GoG utilizes the attention alignment module to fuse the three graphs. Finally, GoG uses the fused multi-modal information to give corresponding answers.

Firstly, we simply describe the feature representation of three inputs. Secondly, we introduce our graph attention. Then we describe how we apply our graph attention to the history graph, question graph and image graph to construct our graph-over-graph network. Finally, we describe how we apply our graph-over-graph network to the strong visual dialog models.

2.2 Feature Representation

Similar to (Anderson et al., 2018), we extract the image features by using a pretrained Faster RCNN (Ren et al., 2015b). We select μ object proposals for each image, where each object proposal is represented by a 2048-dimension feature vector. The obtained visual region features are denoted as $v = v_{i=0}^{\mu} \in R^{\mu \times d_v}$.

To extract the question features, each word is embedded into a 300-dimensional vector initialed with the Glove vector (Pennington et al., 2014). The word embeddings are taken as inputs by an LSTM encoder (Hochreiter and Schmidhuber, 1997), which produces the initial question representation $q \in R^{\lambda \times d_q}$. Each history sentence features are obtained as same as the question features. We concatenate the last state $h^{last} \in R^{d_h}$ of each turn history features to get the initial history representation $h = h_0^{t-1} = [h_0^{last}, \ldots, h_{t-1}^{last}] \in R^{t \times d_h}$. λ denotes the length of the question, t denotes the turn of dialog history, d_q denotes the dimension of each word in questions, d_h denotes the dimension of each word in history and $[\cdot, \cdot]$ denotes the concatenation operation.

2.3 Graph Attention

Given a target node i and a neighboring node $j \in \mathcal{N}(i)$ with a $k \times k$ adjacency matrix R, where $\mathcal{N}(i)$ is the set of k nodes neighboring with node i, and the representations of node i and node j are u_i and u_j, respectively. To obtain the correlation score s_{ij} between node i and j, self-attention (Vaswani et al., 2017) is then performed on the vertices, which generates a relation score s_{ij} between node features u_i and u_j:

$$s_{ij} = \frac{(U_i u_i)^T \cdot V_j u_j}{\sqrt{d_u}}, \tag{1}$$

where U_i and V_j are trainable parameters. We apply a softmax function over the correlation score s_{ij} to obtain weight α_{ij} :

$$\alpha_{ij} = \frac{\exp(s_{ij} + c_{u,lab(i,j)})}{\sum_{j \in \mathcal{N}(i)} \exp(s_{ij} + c_{u,lab(i,j)})}, \tag{2}$$

where $c_{\{\cdot\}} = W_{lab} A_{ij}$ is a bias term, $lab(i,j)$ represents the label of each edge, and W_{lab} is a learned parameter. The representations of neighboring nodes u_j are first transformed via a learned linear transformation with W_u. Those transformed representations are then gathered with weight α_{ij} , followed by a non-linear function σ. This propaga-

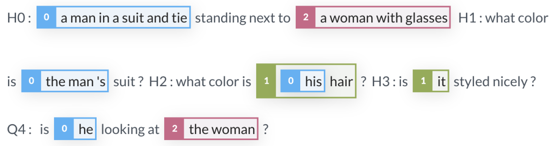

Figure 3: The history is performed by coreference resolution. The same color box and the same number indicate the coreference relation between different expressions of the same entity.

tion can be denoted as:

$$u_i^* = \sigma\Big(u_i + \sum_{j \in \mathcal{N}(i)} R_{ij}\alpha_{ij}W_u u_j\Big). \tag{3}$$

We utilize $\mathrm{GraphAtt}(\cdot)$ to denote equations from Eq. (1) to Eq. (3)

2.4 History Graph Construction

In practice, we observe that coreference relations exist in dialog history. To fully understand the coreference among dialog history, we utilize the coreference resolution tool (Lee et al., 2017) to identify coreference relations. We use the caption and questions to identify the relations instead of QA pairs because there is no ground truth answer in the test split. As shown in Figure 3, we provide a four-turn dialog to show coreference resolution. The same color boxes with the same numbers indicate the coreference relations. For example, the blue box with number 0 indicates they are related to the word “a man” with its attribute “in a suit and tie”.

Pruned History Graph with Coreference Relations. We treat each turn history as a node. By analyzing the coreference relations of the history, we obtain the relations between history as shown in Figure 3. According to coreference relations, we construct a sparse graph, as shown in the history graph of Figure 2.

History Graph Attention. Given a graph with t nodes, i.e. a t-turn dialog, each turn representation in history is a node. We represent the graph structure with a $t \times t$ adjacency matrix A, where $A_{ij} = 1$ if there is a coreference relation between node i and node j; else $A_{ij} = 0$.

The relation-aware graph based history representation h_i^* is as follows:

$$h_i^* = \mathrm{GraphAtt}(h_i, A) \tag{4}$$

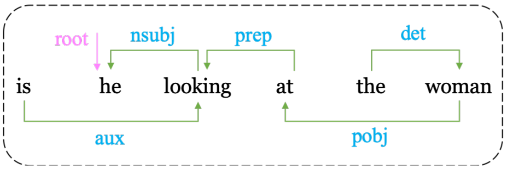

Figure 4: The question is performed by dependency parsing. The word in pink is the root node. The direction of green arrows indicates the dependency relation between two words, and the blue words (e.g., det, dobj) are relation types.

2.5 Question Graph Construction

In practice, we observe that two words in a sentence usually hold a certain relation. Such relations can be identified by the Neural Dependency Parsing (Dozat and Manning, 2017).

Pruned Question Graph with Dependency Relations. We treat each word in a question as a vertex. By parsing the dependency relations of a question, we obtain the relations between words as shown in Figure 4. According to dependency relations, we obtain our sparse question graph, as shown in the question graph of Figure 2.

History-aware Question Graph Attention. Given a graph with λ nodes, each word in a question is a node. We represent the graph structure with a $\lambda \times \lambda$ adjacency matrix B, where $B_{ij} = 1$ if there is a dependency relation between node i and node j; else $B_{ij} = 0$.

In order to utilize the history to help understand questions, we use a history-aware attention mechanism to inject semantic information from the history into the question graph. The aware history representation is calculated as follows:

$$\hat{h} = \mathrm{softmax}\big(W_{h_1}\sigma(W_{h_2}h^*)\big)h^*, \tag{5}$$

where $\hat{h} \in R^{d_h}$, h^* is the final representation of dialog history. W_{h_1} and W_{h_2} are learned parameters. The history-aware question features are achieved by concatenating the adaptive history representation $\hat{h}$ with each of question features q_i, denoted as:

$$q_i' = [q_i, \hat{h}], \quad \text{for} \quad i = 1, \dots, \lambda. \tag{6}$$

The history-aware and relation-aware graph based question representation q_i^* is as follows:

$$q_i^* = \mathrm{GraphAtt}(q_i', B) \tag{7}$$

2.6 Image Graph Construction

Pruned Image Graph with Spatial Relations. By treating each object region in an image as a vertex, we can construct a fully-connected undirected graph, as shown in the image graph of Figure 2. Each edge represents a relation between two object regions. Spatial relations represent an object position in an image, which correspond to a 4-dimension spatial coordinate $[x_1, y_1, x_2, y_2]$. Note that (x_1, y_1) is the coordinate of the top-left point of the bounding box and (x_2, y_2) is the coordinate of the bottom-right point of the bounding box. Following Yao et al. (Yao et al., 2018), we classify different spatial relations into 11 different categories, such as *inside*, *cover* and *overlap*. We utilize the overlapping region between two object regions to judge whether there is an edge between two regions. If two object regions have overlapping parts, it means that there is a strong correlation between these two objects. If two object regions are too far away from each other, it means that there is no relation between these two objects. According to the spatial relations, we prune some irrelevant relations between objects and obtain a sparse graph, as shown in the image graph of Figure 2.

Question-aware Image Graph Attention. Given a graph with μ nodes, each object in an image is a node. We represent the graph structure with a $\mu \times \mu$ adjacency matrix D, where $D_{ij} = 1$ if there is a spatial relation between node i and node j; else $D_{ij} = 0$.

Based on the fully question understanding, we use a question-aware attention mechanism to inject semantic information from the question into the image graph. The adaptive question representation is calculated as follows:

$$\hat{q} = \mathrm{softmax}\big(W_{q_1}\sigma(W_{q_2}q^*)\big)q^*, \qquad (8)$$

where W_{q_1} and W_{q_2} are learned parameters. The question-aware image features are achieved by concatenating the aware question representation $\hat{q}$ with each of the μ image features v_i, denoted as:

$$v_i' = [v_i, \hat{q}], \quad \text{for} \quad i = 1, \ldots, \mu. \qquad (9)$$

The question-aware and relation-aware graph based image representation v_i^* is as follows:

$$v_i^* = \mathrm{GraphAtt}(v_i', D) \qquad (10)$$

2.7 Multi-modal Fusion

After obtaining the relation-aware representation, we fuse the question representation q^*, history representation h^*, visual representation v^* through a multi-modal fusion strategy. We can use any existing visual dialog models to learn a joint representation $\mathcal{J}$:

$$\mathcal{J} = \mathcal{F}(q^*, h^*, v^*; \Theta), \qquad (11)$$

where $\mathcal{J}$ is a visual dialog model and Θ are trainable parameters of the fusion module. The design of generative and discriminative decoders (Das et al., 2017), and multi-task learning strategy (Nguyen et al., 2020) can be referred to Appendix A.

3 Experiments

3.1 Experiment Setup

Datasets and Implementation Details. We conduct experiments on the VisDial v0.9 and v1.0 datasets (Das et al., 2017) to verify our approach. VisDial v0.9 contains 83k dialogs on COCO-train (Lu et al., 2017) and 40k dialogs on COCO-val images as the test set, for a total of 1.23M dialog question-answer pairs. VisDial v1.0 dataset is an extension of VisDial v0.9 dataset with additional 10k COCO-like images. VisDial v1.0 dataset contains 123k, 2k, and 8k images as train, validation, and test splits, respectively.

To represent image regions, we use Faster R-CNN (Ren et al., 2015b) with ResNet-101 (He et al., 2016) finetuned on the Visual Genome dataset (Krishna et al., 2017), thus obtaining a 2048-dimension feature vector for each region. Following (Nguyen et al., 2020), we detect K = 100 objects from each image. For the question and history features, we first build the vocabulary composed of 11,322 words that appear at least five times in the training split. The captions, questions, and answers are truncated or padded to 40, 20, and 20 words, respectively. We employ multi-head attention with 4 heads for all three graph attention networks. The dimension of hidden features is set to 512.

Our model is implemented based on PyTorch (Paszke et al., 2017). In experiments, we use Adam (Kingma and Ba, 2014) optimizer for training, with the mini-batch size as 32. For the choice of the learning rate, we employ the warm-up strategy (Goyal et al., 2017). Specifically, we begin

Model	VisDial v0.9 (val)					VisDial v1.0 (val)					
	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓	NDCG ↑	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓
Attention-based Model											
RvA (Niu et al., 2019)	55.43	45.37	65.27	$\underline{72.97}$	$\underline{10.71}$	-	-	-	-	-	-
DVAN (Guo et al., 2019b)	55.94	$\underline{46.58}$	65.50	71.25	14.79	-	-	-	-	-	-
DMRM (Chen et al., 2020a)	$\underline{55.96}$	46.20	$\underline{66.02}$	72.43	13.15	-	50.16	40.15	60.02	67.21	15.19
DAM (Jiang et al., 2020c)	-	-	-	-	-	60.93	$\underline{50.51}$	$\underline{40.53}$	$\underline{60.84}$	67.94	16.65
Pretraining-based Model											
VDBERT (Wang et al., 2020)$^\diamond$	55.95	46.83	65.43	72.05	13.18	-	-	-	-	-	-
Graph-based Model											
KBGN (Jiang et al., 2020a)	-	-	-	-	-	60.42	50.05	40.40	60.11	66.82	17.54
LTMI (Nguyen et al., 2020)†	55.85	46.07	65.97	72.44	14.17	$\underline{61.61}$	50.38	40.30	60.72	$\underline{68.44}$	$\underline{15.73}$
LTMI-GoG (Ours)	**56.32**	**46.65**	**66.41**	72.69	13.78	**62.63**	**51.32**	**41.25**	**61.83**	**69.44**	**15.32**
LTMI-GoG-Multi (Ours)	56.89	47.04	66.92	72.87	13.45	63.35	51.80	41.78	62.23	69.79	15.16
LTMI-GoG-Multi* (Ours)	59.38	48.58	71.33	78.78	9.94	65.20	55.38	43.93	68.22	76.75	9.98

Table 1: Main comparisons on both VisDial v0.9 and v1.0 datasets using the generative decoder. † denotes that we re-implemented the model. ◇ denotes that the model utilizes large extra datasets for training. Underline indicates the highest performance among previous approaches except pretraining-base models. (t-test, p-value<0.01)

with a learning rate of 0.0001, linearly increasing it at each epoch till it reaches 0.0002 at epoch 4. After 15 epochs, the learning rate is decreased by 1/4 for every 2 epochs up to 20 epochs. We use 4 Titan-XP GPU for training. We spend about 4 hour / 1 epoch for the discriminative setting and 1 hour / 1 epoch for the generative setting. The total parameter of our GoG model is 46.94M, while the total parameter of LTMI (Das et al., 2017) is 42.20M. GoG only has an increase of 4.74M than LTMI.

Automatic Evaluation. We use a retrieval setting to evaluate individual responses at each round of a dialog, following (Das et al., 2017). Specifically, at test time, a list of 100-candidate answers is also given. The model is evaluated on retrieval metrics: (1) Rank of human response, (2) existence of the human response in $top-k$ ranked responses, i.e., R@k (3) Mean reciprocal rank (MRR) of the human response and (4) Normalized discounted cumulative gain (NDCG) for VisDial v1.0.

Human Evaluation. We randomly extract 100 samples for human evaluation (Wu et al., 2018) and then ask 3 human subjects to guess whether the last response in the dialog is human-generated or machine-generated. If at least 2 of them agree it is generated by a human, we think it passes the Truing Test (M1). We record the percentage of responses that are evaluated better than or equal to human responses (M2), according to the human subjects' evaluation.

3.2 Main Results

Baseline methods. In our experiment, compared methods can be grouped into four types: (1) Fusion-based models. (2) Attention-based models: ReDAN, CorefNMN, RvA, DVAN, DMRM, DAM. (3) The pretraining model: VDBERT and VisualBERT. (4) Graph-based models: GNN-EM, DualVD, FGA, KBGN. Please refer to Appendix B for more compared methods.

GoG denotes our relation-aware graph-over-graph network. We use the strong model LTMI (Nguyen et al., 2020)[1] as our multi-modal fusion module. LTMI is a very strong model which achieves some the-state-of-the-art results. "Multi" indicates the model uses multi-task learning at training but utilizes the generative or discriminative decoder at inference, respectively. "Multi*" indicates the model uses multi-task learning and utilizes the discriminative decoder to improve the generative decoder. In general, our model outperforms the strong baseline by a significant margin. We use t-test to analyze our model and LTMI (Nguyen et al., 2020). The p-values is less than 0.01, indicating that the results are significantly different.

Generative Results As shown in the right half of Table 1, we compare generative performance on the val v1.0 split. Our method improves significantly (about 1% on all metrics) on the strong baseline LTMI (Nguyen et al., 2020) and outperforms all the compared methods on all metrics with large margins, which proves that GoG can

---

[1] We reproduce results of LTMI by their official GitHub repo (https://github.com/davidnvq/visdial). We apply the default hyper-parameters as them.

Model	VisDial v0.9 (val)					VisDial v1.0 (test-std)					
	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓	NDCG ↑	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓
Attention-based Model											
ReDAN (Gan et al., 2019)	-	-	-	-	-	57.63	64.75	51.10	81.73	90.90	3.89
MCA (Agarwal et al., 2020)	-	-	-	-	-	72.73	37.68	20.67	56.67	72.12	8.89
Pretraining-based Model											
VisualBERT (Murahari et al., 2020)$^{\diamond}$	-	-	-	-	-	74.47	50.74	37.95	64.13	80.00	6.28
VDBERT (Wang et al., 2020)$^{\diamond}$	70.04	57.79	85.34	92.68	4.04	75.35	51.17	38.90	62.82	77.98	6.69
Graph-based Model											
GNN-EM (Zheng et al., 2019)	62.85	48.95	79.65	88.36	4.57	52.82	61.37	47.33	77.98	87.83	4.57
DualVD (Jiang et al., 2020b)	62.94	48.64	80.89	89.94	4.17	56.32	63.23	49.25	80.23	89.70	4.11
FGA (Schwartz et al., 2019)	65.25	51.43	82.08	89.56	4.35	56.90	66.20	52.75	82.92	91.07	3.80
CAG (Guo et al., 2020)	67.56	54.64	83.72	91.48	3.75	56.64	63.49	49.85	80.63	90.15	4.11
KBGN (Jiang et al., 2020a)	-	-	-	-	-	57.60	64.13	50.47	80.70	90.16	4.08
LTMI (Nguyen et al., 2020)†	66.41	53.36	82.53	90.54	4.03	60.74	61.20	47.08	77.78	87.60	4.88
LTMI-GoG (Ours)	66.76	53.84	82.89	90.90	3.91	60.38	63.13	49.88	79.65	89.05	4.39
LTMI-GoG-Multi (Ours)	66.97	54.03	83.10	91.22	3.83	61.04	63.52	50.01	80.13	89.28	4.31

Table 2: Main comparisons on both VisDial v0.9 and v1.0 datasets using the discriminative decoder. $\diamond$ denotes that the model utilizes large extra datasets for training. Underline indicates the highest performance among previous approaches except the pretraining-based models. (t-test, p-value<0.01)

Model	NDCG	MRR	R@1	R@5	R@10	Mean
ReDAN (Gan et al., 2019)	-	64.29	50.65	81.29	90.17	4.10
KBGN	59.08	64.86	51.37	81.71	90.54	4.00
VDBERT (Wang et al., 2020)‡	56.20	62.25	48.16	79.57	89.01	4.31
VDBERT (Wang et al., 2020)$^{\diamond}$	63.22	67.44	54.02	83.96	92.33	3.53
LTMI (Nguyen et al., 2020)†	61.52	62.31	48.92	78.55	87.77	4.86
LTMI-GoG	**62.24**	63.81	50.33	80.48	89.24	4.35
LTMI-GoG-Multi	63.15	62.68	49.46	78.77	87.87	4.81

Table 3: Main comparisons on VisDial v1.0 val datasets using the discriminative decoder.

Row	Model	NDCG
	LTMI	61.61
0	LTMI-GoG	62.63
1	w/o I-Graph	61.96
2	w/o Q-Graph	62.15
3	w/o H-Graph	62.03
4	w/o Q-Aware	62.41
5	w/o H-Aware	62.31
6	w/o Spatial Relation	62.15
7	w/o Dependency Relation	62.24
8	w/o Coreference Relation	62.31

Table 4: Ablation study on VisDial v1.0 val datasets using the generative decoder.

improve the performance of visual dialog models by introducing explicit relation reasoning. Compared with the graph-based model KBGN (Jiang et al., 2020a), our GoG-gen improves NDCG from 60.42 to 62.63 (+2.21%), MMR from 50.05 to 51.32 (+1.27%), which illustrates that our explicit relation reasoning is more effective because our approach reduce the noise of implicit relation modeling. LTMI-GoG-Multi and LTMI-GoG-Multi* obtain higher performance with large margins comparing with LTMI (Nguyen et al., 2020), which shows that our approach is effective on multi-task setting. As shown in the left half of Table 1, we come to a similar conclusion on the val v0.9 split. Our method improves a big margin (about 0.5% on all metrics) on LTMI (Nguyen et al., 2020) and outperforms all the none pre-trained methods on MRR, R@1, and R@5.

Discriminative Results As shown in the right half of Table 2, our method improves a lot (near 1.5% on all metrics except NDCG) based on LTMI (Nguyen et al., 2020) on the test-std v1.0 split. We also compare the performance on the val v1.0 split as shown in Table 3. As shown in the left half of Table 2, we compare discriminative performance on the val v0.9 split. Our method improves a lot based on the LTMI (Nguyen et al., 2020). As shown in Table 3, our approach outperforms VDBERT (Wang et al., 2020)‡ which trains from scratch without extra datasets. All the comparison show that our approach is valid due to explicit relation modeling.

3.3 Ablation Study

As shown in Table 4, we firstly remove the I-Graph, Q-Graph, H-Graph to validate the effect of each graph, respectively. Secondly, we validate the importance of concatenating operation. Finally, we use full connections to replace the relation in the graph to validate the importance of each relation. Firstly, the comparison between line 0 and line 1/2/3 shows all three graphs are crucial for visual dialog, leading to higher performance, and the I-Graph is most important. Secondly, the comparison between line 0 and line 4/5 shows that adding adaptive features gives a gain of approximately

	LTMI (Nguyen et al., 2020)	GoG
Method 1 (M1)	53	64
Method 2 (M2)	60	67

Table 5: Human evaluation on 100 sampled responses on VisDial val v1.0. M1: percentage of responses pass the Turing Test. M2: percentage of responses evaluated better than or equal to human responses.

about +0.2. Thirdly, the comparison between line 1 and line 6/7/8 shows that doing graphs with relations gives better gain than simple fully-connected graphs. Spatial relation is the pick of the bunch because the full connection of 100 objects in an image will bring lots of noise.

3.4 Human Study

As shown in Table 5, we conduct human study to further prove the effectiveness of our model. Our model achieves the highest scores both on the metric M1 and M2 compared with the previous model, LTMI (Nguyen et al., 2020). These results show that our model can generate a better contextually and visually coherent response.

3.5 Qualitative Results

As shown in Figure 5, we visualize the learned attention maps. For the image, the colorful region means higher attention weights. We draw the bounding boxes of the first three highest scores. For the question, the word which has the darker color has higher attention weights. For dialog history, the darker QA pairs have a higher coreference score with the question. Figure 6 provides some dialog examples, as a comparison of the results between GoG and the state-of-the-art LTMI model in the supplementary material. We have two observations by analyzing a set of randomly selected examples. Firstly, GoG generally provides more accurate answers. Secondly, GoG tends to provide longer and more natural human-like answers. More examples can be referred to Appendix B.

4 Related Work

4.1 Visual dialog

For the visual dialog task (Das et al., 2017), GNN-EM (Zheng et al., 2019) utilizes an EM-style GNN to conduct the textual coreference, which regards the caption and the previous question-answer (QA) pairs as observed nodes, and the current answer is deemed as an unobserved node inferred using EM algorithms (Moon, 1996) on the textual contexts. FGA (Schwartz et al., 2019) realizes a factor graph attention mechanism, which constructs the graph over all the multi-modal features and estimates their interactions. DualVD (Jiang et al., 2020b) constructs a scene graph to represent the image while embedding both relationships provided by (Zhang et al., 2019b) and original object detection features (Anderson et al., 2018). CAG (Guo et al., 2020) focuses on an iterative question-conditioned context-aware graph, including both fine-grained visual-objects and textual-history semantics. In this paper, we model explicit complex relations within and among visual content, dialog history and the current question and design a graph-over-graph structure which are different from graph-based models mentioned above.

4.2 Graph Neural Network

Graph neural networks (Kipf and Welling, 2016; Veličković et al., 2017; Xinyi and Chen, 2018; Zhang et al., 2019a) have attracted attention in various tasks (Wang et al., 2019; Liu et al., 2018; Gu et al., 2019). The core idea is to combine the graphical structural representation with neural networks, which is suitable for reasoning-style tasks. For visual question answering, Liu et al. (Teney et al., 2017) propose the first GNN-based approach, which builds a scene graph of the image and parses the sentence structure of the question, and calculates their similarity weights. Li et al. (Li et al., 2019) propose to encode each image into a graph and model multi-type inter-object relations via a graph attention mechanism, such as spatial relations and semantic, and implicit relations (Li et al., 2019). Huang et al. (Huang et al., 2020) propose a novel dual-channel graph convolutional network for better combining visual and textual advantages. These approaches are limited to built independent graphs. There is no exploration of the coreference among dialog history and relations between graphs in the approach mentioned above.

5 Conclusion

In this paper, we present a relation-aware graph-over-graph network (GoG), a novel framework for visual dialog, which models and reasons the explicit complex relations among visual content, dialog history, and the current question. GoG exploits the graph-over-graph structure to obtain three relation-aware multi-modal representation which

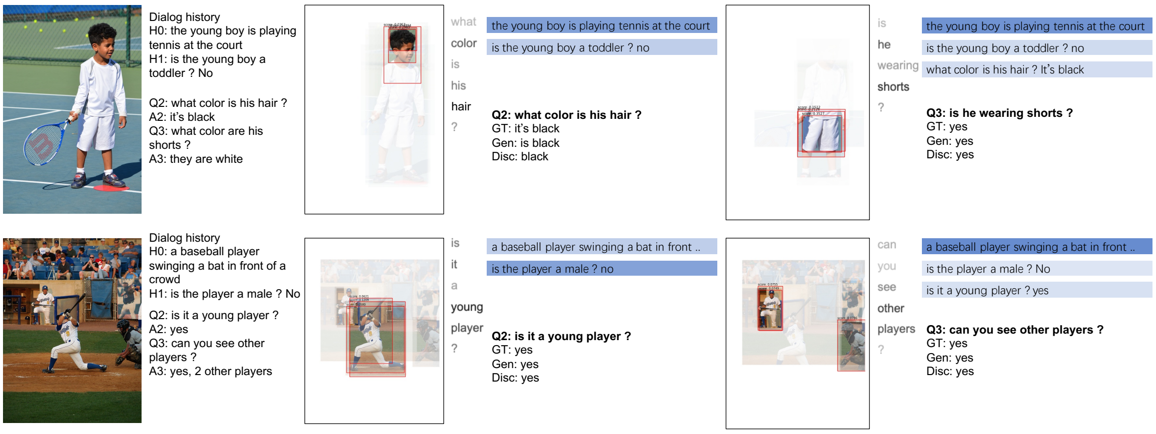

Figure 5: Visualization of attention maps generated in our model at two Q&A rounds on two images.

traffic light in front a tree a statue of a lion is behind the tree a large building is in back

Question	Ground Truth	Generation by GoG	Generation by LTMI	Retrieval by GoG	Retrieval by LTMI
Q1: are there a lot of tree ?	i see 1 tree and some bushes	i see one tree	just see 1 tree	i see 1 tree and some bushes	no, just 1
Q2: what color is the traffic light ?	it 's red	it 's red	red 's a	the light is red	the light is red
Q3: is this in black and white ?	it 's in color	no, it 's in color	no 's a color	it 's in color	no
Q4: is there more than 1 building in the background ?	i can only see one	i only only see 1	no see only see the building	i can only see one	i can only see one
Q5: are there animals ?	no live animals	no	no	no live animals	no live animals

Figure 6: Examples of dialogs generated and retrieved by our model and the LTMI baseline. Our model provides answers that are more accurate than LTMI (green denotes correct answers, and red denotes wrong answers). Results from our model are also more natural and comprehensive (highlighted in blue).

can be added to prior visual dialog models. Experimental results on two large-scale datasets show that our approach improves the previous models by a significant margin.

References

Shubham Agarwal, Trung Bui, Joon-Young Lee, Ioannis Konstas, and Verena Rieser. 2020. History for visual dialog: Do we really need it? *arXiv preprint arXiv:2005.07493*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic propositional image caption evaluation. *Adaptive Behavior*, 11(4):382–398.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443.

Feilong Chen, Fandong Meng, Jiaming Xu, Peng Li, Bo Xu, and Jie Zhou. 2020a. DMRM: A dual-channel multi-hop reasoning model for visual dialog. *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Xiuyi Chen, Fandong Meng, Peng Li, Feilong Chen, Shuang Xu, Bo Xu, and Jie Zhou. 2020b. Bridging the gap between prior and posterior knowledge selection for knowledge-grounded dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3426–3437.

Xiuyi Chen, Jiaming Xu, and Bo Xu. 2019. A working memory model for task-oriented dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2687–2693.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, and Jianfeng Gao. 2019. Multi-step reasoning via recurrent dual attention for visual dialog. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6463–6474.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, pages 2296–2304.

Mehdi Ghanimifard and Simon Dobnik. 2019. What goes into a word: generating image descriptions with top-down spatial knowledge. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 540–551.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. 2019. Scene graph generation with external knowledge and image reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1969–1978.

Dalu Guo, Chang Xu, and Dacheng Tao. 2019a. Image-question-answer synergistic network for visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10434–10443.

Dan Guo, Hui Wang, and Meng Wang. 2019b. Dual visual attention network for visual dialog. pages 4989–4995.

Dan Guo, Hui Wang, Hanwang Zhang, Zheng-Jun Zha, and Meng Wang. 2020. Iterative context-aware graph inference for visual dialog. *arXiv preprint arXiv:2004.02194*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Qingbao Huang, Jielong Wei, Yi Cai, Changmeng Zheng, Junying Chen, Ho-fung Leung, and Qing Li. 2020. Aligned dual channel graph convolutional network for visual question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7166–7176.

Xiaoze Jiang, Siyi Du, Zengchang Qin, Yajing Sun, and Jing Yu. 2020a. KBGN: Knowledge-bridge graph network for adaptive vision-text reasoning in visual dialogue. *Proceedings of the 28th ACM International Conference on Multimedia*.

Xiaoze Jiang, Jing Yu, Zengchang Qin, Yingying Zhuang, Xingxing Zhang, Yue Hu, and Qi Wu. 2020b. DualVD: An adaptive dual encoding model for deep visual understanding in visual dialogue. In *AAAI*, volume 1, page 5.

Xiaoze Jiang, Jing Yu, Yajing Sun, Zengchang Qin, Zihao Zhu, Yue Hu, and Qi Wu. 2020c. DAM: Deliberation, abandon and memory networks for generating detailed and non-repetitive responses in visual dialogue. *arXiv preprint arXiv:2007.03310*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. *ArXiv*, abs/1809.01816.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Relation-aware graph attention network for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10313–10322.

Yong Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2018. Structure inference net: Object detection using scene-level context and instance-level relationships. In *Proceedings of the IEEE conference* *on computer vision and pattern recognition*, pages 6985–6994.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems*, pages 314–324.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Todd K Moon. 1996. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60.

Vishvak Murahari, Dhruv Batra, Devi Parikh, and Abhishek Das. 2020. Large-scale pretraining for visual dialog: A simple state-of-the-art baseline. *Proceedings of the European Conference on Computer Vision*.

Van-Quang Nguyen, Masanori Suganuma, and Takayuki Okatani. 2020. Efficient attention mechanism for visual dialog that can handle all the interactions between multiple inputs. *Proceedings of the European Conference on Computer Vision*.

Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. 2019. Recursive visual attention in visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6679–6688.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Jiaxin Qi, Yulei Niu, Jianqiang Huang, and Hanwang Zhang. 2020. Two causal principles for improving visual dialog. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015a. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015b. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G Schwing. 2019. Factor graph attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2039–2048.

Damien Teney, Lingqiao Liu, and Anton van Den Hengel. 2017. Graph-structured representations for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Peng Wang, Qi Wu, Jiewei Cao, Chunhua Shen, Lianli Gao, and Anton van den Hengel. 2019. Neighbourhood watch: Referring expression comprehension via language-guided graph attention networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1960–1968.

Yue Wang, Shafiq Joty, Michael R Lyu, Irwin King, Caiming Xiong, and Steven CH Hoi. 2020. VD-BERT: A unified vision and dialog transformer with bert. *arXiv preprint arXiv:2004.13278*.

Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton van den Hengel. 2018. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6106–6115.

Zhang Xinyi and Lihui Chen. 2018. Capsule graph neural network. In *International conference on learning representations*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of International Conference on Machine Learning*, pages 2048–2057.

Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2018. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 684–699.

Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019a. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803.

Ji Zhang, Yannis Kalantidis, Marcus Rohrbach, Manohar Paluri, Ahmed Elgammal, and Mohamed Elhoseiny. 2019b. Large-scale visual relationship understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9185–9194.

Zilong Zheng, Wenguan Wang, Siyuan Qi, and Song-Chun Zhu. 2019. Reasoning visual dialogs with structural and partial observations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6669–6678.

A Relation-aware Graph-over-Graph Network

A.1 Question Graph Construction

In practice, we observe that two words in a sentence usually hold certain relations. Such relations can be identified by the Neural Dependency Parsing (Dozat and Manning, 2017). As shown in Table 6, we list a part of commonly-used dependency relations.

A.2 Attention Alignment Module

After obtaining relation-aware features, we fuse the question representation q^*, history representation h^*, visual representation v^* through a multi-modal fusion strategy. We can use any existing multi-modal fusion method to learn a joint representation $\mathcal{J}$:

$$\mathcal{J} = \mathcal{F}(q^*, h^*, v^*; \Theta), \tag{12}$$

where $\mathcal{J}$ is a multi-modal fusion method and Θ are trainable parameters of the fusion module. Here we utilize an efficient attention mechanism method (Nguyen et al., 2020) to fuse the multi-modal information, which is the state-of-the-art model in visual dialog.

Let $A_X(Y)$ denotes the efficient attention mechanism (Nguyen et al., 2020) from the information X to the information Y. For example, $A_{v^*}(v^*)$ denotes the efficient self-attention. The fused visual representation is obtained as follows:

$$\begin{aligned} v_{contact} &= [A_{v^*}(v^*), A_{q^*}(v^*), A_{h^*}(v^*)], && (13) \\ \overline{v}' &= \mathrm{LayerNorm}(\sigma(v_{concat} W_{v^*}) + v^*), && (14) \\ a_V &= \mathrm{softmax}(W_{V_1} \sigma(W_{V_2} \overline{v}')), && (15) \\ \overline{v} &= \sum_{i=1}^{\mu} a_{V,i} \overline{v}'_i, && (16) \end{aligned}$$

where W_{v^*}, W_{V_1}, W_{V_2} are learned parameters. $\overline{q}$ and $\overline{h}$ can be obtained similarly. Thus, the joint representation $\mathcal{J}$ is obtained:

$$\mathcal{J} = W_J[\overline{q}, \overline{h}, \overline{v}], \tag{17}$$

where W_J is a learned parameter.

Relations	Relation Description	Proportion
nsubj	nominal subject	16.1%
root	root node	16.0%
dep	dependent	15.7%
punct	punctuation	14.3%
det	determiner	9.0%
cop	copula	9.0%
prep	prepositional modifier	4.6%
aux	auxiliary	4.0%
pobj	object of a preposition	3.6%
amod	adjective modifier	3.2%
advmod	adverbial modifier	2.5%
dobj	direct object	1.5%

Table 6: The main categories of relations classified by the dependency parsing tool (Dozat and Manning, 2017) in VisDial v1.0 training split (Das et al., 2017).

A.3 Generative and Discriminative Decoders

Following Das et al. (Das et al., 2017), we consider both generative and discriminative decoders to score the candidate answers using the likelihood scores and the log-likelihood scores, respectively.

Generative Decoder Following Das et al. (Das et al., 2017), we design the generative decoder to score the candidate answers using the log-likelihood scores. Specifically, the generative decoder utilizes a two-layer LSTM (Hochreiter and Schmidhuber, 1997) to generate an answer using the context vector $\mathcal{J}$ as the initial hidden state. In the training phase, the generative decoder generates the next token based on the current token from the ground truth answer. In detail, we first append the special token "SOS" at the beginning of the ground truth answer and "EOS" at the end. We use Glove (Pennington et al., 2014) to initialize the embedding and obtain the embedding vectors $a_{gt} = [w_0, w_1, \ldots, w_N]$ where w_0 is the embedding of "SOS" and w_N is the embedding of "EOS". The hidden state h_n at timestep n is computed as follows:

$$h_n = \mathrm{LSTM}(w_{n-1}, h_{n-1}), \tag{18}$$

where h_0 is intializaed by $\mathcal{J}$. Then we obtain the log-likelihood of n-th word as follows:

$$p = \mathrm{logsoftmax}(Wh + b), \tag{19}$$

where W and b are learned parameters. In the training phase, we minimize the summation of the

Model	VisDial v0.9 (val)					VisDial v1.0 (val)					
	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓	NDCG ↑	MRR ↑	R@1 ↑	R@5 ↑	R@10 ↑	Mean ↓
Fusion-based Models											
LF (Das et al., 2017)	51.99	41.83	61.78	67.59	17.07	-	-	-	-	-	-
HRE (Das et al., 2017)	52.37	42.23	62.28	68.11	16.97	-	-	-	-	-	-
Attention-based Model											
MN (Das et al., 2017)	52.59	42.29	62.85	68.88	17.06	51.86	47.99	38.18	57.54	64.32	18.60
HCIAE (Lu et al., 2017)	53.86	44.06	63.55	69.24	16.01	59.70	49.07	39.72	58.23	64.73	18.43
CorefNMN (Kottur et al., 2018)	53.50	43.66	63.54	69.93	15.69	-	-	-	-	-	-
CoAtt (Wu et al., 2018)	54.11	44.32	63.82	69.75	16.47	59.24	49.64	40.09	59.37	65.92	17.86
RvA (Niu et al., 2019)	55.43	45.37	65.27	72.97	10.71	-	-	-	-	-	-
DVAN (Guo et al., 2019b)	55.94	46.58	65.50	71.25	14.79	-	-	-	-	-	-
Primary (Guo et al., 2019a)	-	-	-	-	-	-	49.01	38.54	59.82	66.94	16.60
ReDAN (Gan et al., 2019)	-	-	-	-	-	60.47	50.02	40.27	59.93	66.78	17.40
DMRM (Chen et al., 2020a)	55.96	46.20	66.02	72.43	13.15	-	50.16	40.15	60.02	67.21	15.19
DAM (Jiang et al., 2020c)	-	-	-	-	-	60.93	50.51	40.53	60.84	67.94	16.65
Pretraining-based Model											
VDBERT (Wang et al., 2020)	55.95	46.83	65.43	72.05	13.18	-	-	-	-	-	-
Graph-based Model											
KBGN (Jiang et al., 2020a)	-	-	-	-	-	60.42	50.05	40.40	60.11	66.82	17.54
LTMI (Nguyen et al., 2020)†	55.85	46.07	65.97	72.44	14.17	61.61	50.38	40.30	60.72	68.44	15.73
GoG-Gen (Ours)	**56.32**	**46.65**	**66.41**	72.69	13.78	**62.63**	**51.32**	**41.25**	**61.83**	**69.44**	**15.32**
GoG-Multi-Gen (Ours)	56.89	47.04	66.92	72.87	13.45	63.35	51.80	41.78	62.23	69.79	15.16
GoG-Multi (Ours)	59.38	48.58	71.33	78.78	9.94	65.20	55.38	43.93	68.22	76.75	9.98

Table 7: Main comparisons on both VisDial v0.9 and v1.0 datasets using the generative decoder. † denotes that we re-implemented the model. Underline indicates the highest performance among previous approaches except pretraining-base models.

negative log-likelihood $\mathcal{L}_\mathcal{G}$ defined by:

$$\mathcal{L}_\mathcal{G} = -\sum_{n=1}^{N} p_n. \tag{20}$$

In the validation and test phase, we compute the summation s_i of the log-likelihood for each candidate answer $\hat{a}_i$:

$$s_i = \sum_{n=1}^{N} p_n^{\hat{a}_i}. \tag{21}$$

Then, the rankings of the candidate answers are derived as $\mathrm{softmax}(s_1, \ldots, s_{100})$.

Discriminative Decoder A discriminative decoder outputs the likelihood score for each of 100 candidate answers for the current question. Similar to the generative decoder, we use LSTM to obtain the hidden state h_n for b-th word but we do not use context vector J to initialize the h_0. The representation of each candidate answer is $a_i = h_N$. The score p_i for i-th candidate answer is computed by:

$$p = \mathrm{logsoftmax}(a_1^T \mathcal{J}, \ldots, a_{100}^T \mathcal{J}) \tag{22}$$

In the test phase, we sort the candidate answers using these scores. In the training phase, the cross-entropy loss $\mathcal{L}_\mathcal{D}$.

A.4 Multi-Task Learning

According to (Nguyen et al., 2020), we apply our GoG to the state-of-the-art model (Nguyen et al., 2020) in the multi-task learning setting that accuracy is improved by training the entire network using the two decoders simultaneously. This is simply done by minimizing the sum of the losses, $\mathcal{L}_\mathcal{D}$ for the discriminative one and $\mathcal{L}_\mathcal{G}$ for the generative one:

$$\mathcal{L} = \mathcal{L}_\mathcal{D} + \mathcal{L}_\mathcal{G} \tag{23}$$

The increase in performance may be attributable to the synergy of learning two tasks while sharing the same encoder.

B Experiments

B.1 Main Results

Comparison with previous approaches using generative decoders. As shown in Table 7, we provide the full comparison with all the previous generative approaches.

B.2 Qualitative Results

More examples generated and retrieved by our GoG are provided in Figure 7. Due to the limited number of pages, we only provide an additional example of Figure 7.

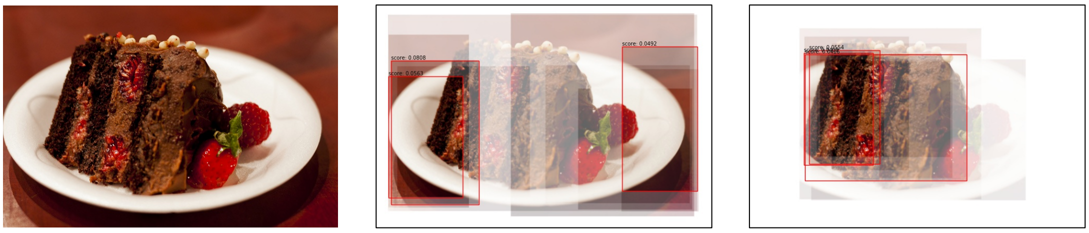

chocolate cake with fresh strawberries and small nuts

Q: is it on a table ?
GT: i think it
Gen: yes think so
Disc yes

Q: is there any silverware ?
GT: no
Gen: no
Disc no

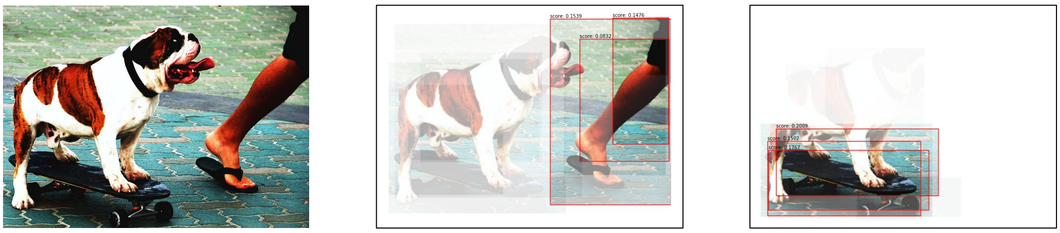

a brown and white dog riding a skateboard

Q: can you see any people ?
GT: part of a person
Gen: just of 1 person
Disc part of a person

Q: what color is the skateboard ?
GT: black
Gen: black
Disc black

a man holding a disk smiles as he leans against a sign at a disc golf course

Q: how many discs are shown ?
GT: 1
Gen: 1
Disc 3

Q: what color disk ?
GT: yellow
Gen: yellow
Disc yellow

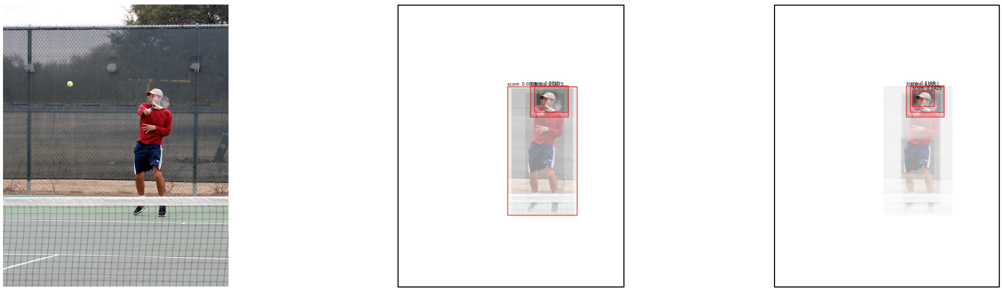

a man playing tennis is attempting to hit the ball with his racket

Q: how old is the man ?
GT: i don ’ t know probably in late 30s
Gen: maybe would ’ t know
Disc looks middle aged

Q: does he wear sunglasses ?
GT: yes
Gen: yes
Disc yes

Figure 7: Examples of dialogs generated and retrieved by our model. blue denotes has the same meaning with the ground truth and red denotes wrong answers.

Joint Optimization of Tokenization and Downstream Model

Tatsuya Hiraoka[†], **Sho Takase**[†], **Kei Uchiumi**[‡], **Atsushi Keyaki**[‡], **Naoaki Okazaki**[†]

[†] Tokyo Institute of Technology
[‡] Denso IT Laboratory, Inc.
`{tatsuya.hiraoka, sho.takase}@nlp.c.titech.ac.jp`
`{kuchiumi, akeyaki}@d-itlab.co.jp`
`okazaki@c.titech.ac.jp`

Abstract

Since traditional tokenizers are isolated from a downstream task and model, they cannot output an appropriate tokenization depending on the task and model, although recent studies imply that the appropriate tokenization improves the performance. In this paper, we propose a novel method to find an appropriate tokenization to a given downstream model by jointly optimizing a tokenizer and the model. The proposed method has no restriction except for using loss values computed by the downstream model to train the tokenizer, and thus, we can apply the proposed method to any NLP task. Moreover, the proposed method can be used to explore the appropriate tokenization for an already trained model as post-processing. Therefore, the proposed method is applicable to various situations. We evaluated whether our method contributes to improving performance on text classification in three languages and machine translation in eight language pairs. Experimental results show that our proposed method improves the performance by determining appropriate tokenizations.

1 Introduction

Tokenization, which converts a raw sentence into a sequence of tokens, is a crucial process that affects the performance of NLP tasks. Existing studies have proposed various tokenization methods including rule-based tokenization (Koehn et al., 2007), dictionary-based tokenization (Kudo, 2006; Morita et al., 2015; Tolmachev et al., 2018; Takaoka et al., 2018), supervised tokenization with neural networks (Yang et al., 2017; Cai et al., 2017; Yang et al., 2019), and unsupervised tokenization (Goldwater et al., 2006, 2009; Mochihashi et al., 2009; Sennrich et al., 2016; Kudo and Richardson, 2018). Much of prior research has reported that an appropriate tokenization depends on each downstream task (Xu et al., 2008; Chang et al., 2008; Nguyen et al., 2010; Domingo et al., 2018; Hiraoka et al., 2019; Gowda and May, 2020). Moreover, Hiraoka et al. (2020) implies that we have to consider a downstream model to determine an appropriate tokenization. In other words, we can improve the performance of a downstream model by determining an appropriate tokenization for the downstream model. However, since traditional tokenizers are isolated from a downstream model, we need to train a given downstream model with each possible tokenization and evaluate its performance to determine the appropriate tokenization. Performing such an exploration whenever we construct a new downstream model is impractical.

Several studies have addressed the optimization of a tokenizer based on a downstream task or/and model (He et al., 2020; Hiraoka et al., 2020), but existing methods are restricted to specific tasks. He et al. (2020) proposed DPE as a tokenization method for a sequence-to-sequence problem such as machine translation. Their method trains a tokenizer with a given training corpus, but it is isolated from a downstream model such as a neural encoder–decoder for machine translation. Hiraoka et al. (2020) proposed OpTok, which jointly trains a tokenizer and a downstream model. However, its architecture is specific to classification problems based on sentence representations, and thus, it cannot be applied for various tasks such as sequence-to-sequence problems. Therefore, there is no method to optimize a tokenizer depending on any downstream task and model.

In this paper, we propose a novel method to jointly optimize a tokenizer and downstream model without any restriction on a task[1]. The proposed method can determine an appropriate tokenization for a downstream model because it explores different tokenizations based on loss values of the down-

---

[1] Code: `https://github.com/tatHi/optok4at`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 244–255
August 1–6, 2021.

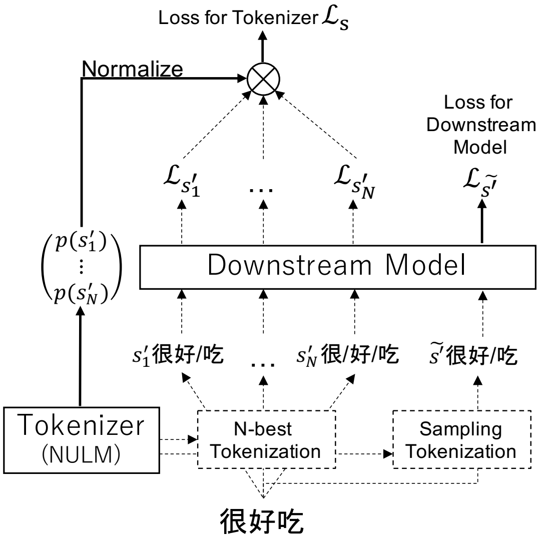

Figure 1: Overview of the proposed method calculating a loss for a tokenizer $\mathcal{L}_s$ and that for a downstream model $\mathcal{L}_{\tilde{s}'}$ using N-best tokenizations (Section 2.2) and a sampled tokenization (Section 2.3), respectively. The arrows with the continuous lines indicate the differentiable path for back-propagation.

stream model. Since the proposed method requires only loss values of the downstream model, we can apply it for any task and model. Moreover, even if a given downstream model is already trained, our proposed method can be applied to improve the performance by refining tokenization. We call this refinement of tokenization *post-processing*. Thus, we can easily use the proposed method in various situations including the case where we have a sufficiently trained downstream model.

We conducted experiments on text classification and machine translation tasks in various languages. Experimental results indicate that the proposed method outperformed existing tokenization methods in both the tasks. We also showed that our method can enhance the performance by refining tokenization as post-processing for downstream models trained with subword regularization (Kudo, 2018; Provilkov et al., 2020).

2 Proposed Method

The proposed method comprises a tokenizer and a downstream model. We optimize the two modules simultaneously. First, we present the training outline of the case where we use one sentence as an input in Section 2.1. Second, we introduce the training of the tokenizer (Section 2.2) and the downstream model (Section 2.3). Finally, we explain the training strategy for a task that requires multiple inputs such as machine translation (Section 2.4).

2.1 Optimizing Tokenization with Loss

The proposed method tokenizes a sentence s into a sequence of words w in vocabulary V, $s' = w_1, ..., w_I$, where I is the sequence length. In this tokenization process, our purpose is to minimize the following loss value:

$$\mathcal{L}_{s'} = q(f(s'), z), \tag{1}$$

where $f(s')$ is a downstream model that outputs a prediction of the downstream task from a tokenized sentence s', and $q(f(\cdot), z)$ is a task-specific loss function between a model prediction and supervisory signal z.

Figure 1 presents an outline of the proposed method. To determine the tokenization satisfying $\mathrm{argmin}_{s'}(q(f(s'), z))$, we update the tokenizer to assign a higher probability to a useful tokenization for the downstream model. Concretely, we construct N tokenizations $s'_1, ..., s'_n, ...s'_N$ for a training instance and then compute loss values for each tokenization. We weight each loss based on probability $p(s'_n)$ computed by the tokenizer and use the weighted sum to train the tokenizer, as follows:

$$a_n = \frac{p(s'_n)}{\sum_{m=1}^{N} p(s'_m)}, \tag{2}$$

$$\mathcal{L}_s = \sum_{n=1}^{N} a_n \mathcal{L}_{s'_n}. \tag{3}$$

In this study, we used N-best tokenizations. In these equations, we weight losses $\mathcal{L}_{s'_1}, ..., \mathcal{L}_{s'_N}$ corresponding to N-best tokenizations with their sentence probabilities normalized such that the sum is 1. By optimizing the tokenizer based on the weighted sum, $\mathcal{L}_s$, the tokenizer assigns high probability to the appropriate tokenization for the downstream model.

We can use any function for $f(\cdot)$ and $q(f(\cdot), \cdot)$ in Eq.(1). Therefore, the proposed method has no restrictions on the downstream task and model. For instance, in the case where text classification is the downstream task, $f(\cdot)$ is a neural network predicting a label of a given tokenized sentence, and $q(f(\cdot), \cdot)$ is the cross-entropy loss between the model prediction and the true label.

2.2 Tokenizer: NULM

We employ a neural unigram language model (NULM) as our tokenizer. It calculates the unigram probability of a word $p(w)$ with a word embedding

$\boldsymbol{v}_w$ as follows:

$$d_w = \mathrm{MLP}(\boldsymbol{v}_w), \tag{4}$$

$$p(w) = \frac{\exp(d_w)}{\sum_{\hat{w} \in V} \exp(d_{\hat{w}})}, \tag{5}$$

where $\mathrm{MLP}(\cdot)$ is a multilayer perceptron. We initialize vocabulary V with a reasonable size of words. For example, we can use a tokenization by SentencePiece (Kudo and Richardson, 2018) or BPE (Sennrich et al., 2016) for the initialization. We also calculate the probability of a tokenization $p(s')$ as follows:

$$p(s') = \prod_{w \in s'} p(w). \tag{6}$$

For the training with Eq.(3), we obtain N-best tokenizations by applying Forward-DP Backward-A* algorithm (Nagata, 1994) for possible tokens against sentence s. In the inference phase, we can also obtain the 1-best appropriate tokenization for the downstream task using Viterbi algorithm (Viterbi, 1967) for the trained NULM.

2.3 Downstream Model Training

We can train the downstream model with loss $\mathcal{L}_s$ in Eq.(3), but we use subword regularization (Kudo, 2018) to obtain a better model. Thus, we compute $\mathcal{L}_{\tilde{s}'} = q(f(\tilde{s}'), z)$ for a sampled tokenization $\tilde{s}'$ and use $\mathcal{L}_{\tilde{s}'}$ to train the downstream model.

We sample tokenization $\tilde{s}'$ from $p(\tilde{s}')^\alpha / \sum_{k=1}^{K} p(s'_k)^\alpha$ computed by the NULM in Eq.(5) (Kudo, 2018). Here, $\alpha \in \mathbb{R}^+$ is a hyperparameter that controls the diversity of the sampled tokenization. If we set α as a lower value, the distribution is similar to the uniform distribution; otherwise, the distribution strongly depends on each tokenization probability $p(\tilde{s}')$. K is also a hyperparameter denoting the number of candidates for sampling, and we use Forward Filtering Backward Sampling (Scott, 2002; Mochihashi et al., 2009) if $K = \infty$.

Subwod regularization not only sophisticates the downstream model but also provides various tokenizations to the downstream model during training. Therefore, subwod regularization helps in exploring the appropriate tokenization.

2.4 Training in Multiple Sentences as Inputs

Previous sections discussed the case where we use one sentence as an input, but we have to input multiple sentences to the downstream model in some tasks. This section describes our training strategy in such cases.

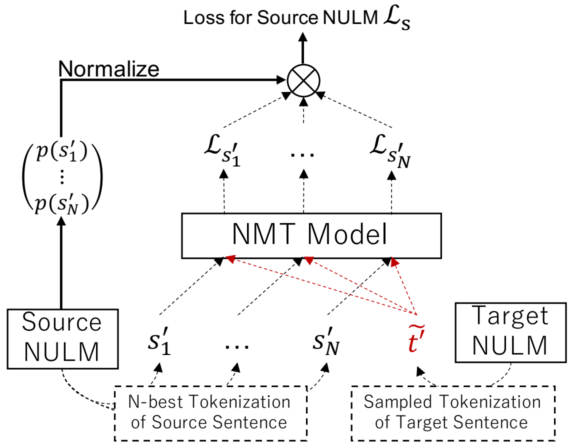

Figure 2: Overview of the calculation of a tokenization loss $\mathcal{L}_s$ for the source-side NULM in NMT requiring two inputs, source and target sentences s and t, respectively. The arrows with the continuous line indicate the differentiable path for back-propagation.

To compute the loss value for training the tokenizer, we consider multiple tokenizations for one sentence and use the sampled tokenization for the others. For example, in machine translation, we input the source and target sentences to train the downstream model. The source sentence is the input of the downstream model, and the target sentence is the supervisory signal. Let s and t be the source sentence and target sentences, respectively, and s' and t' be the corresponding tokenizations. We update the NULM of the source side using $\mathcal{L}_{s'_n} = q(f(s'_n), \tilde{t}')$, where $\tilde{t}'$ is a sampled tokenization for the target sentence. We also compute the loss for the NULM of the target side with $\mathcal{L}_{t'_n} = q(f(\tilde{s}'), t'_n)$, where $\tilde{s}'$ is a sampled tokenization. For training the downstream model, we use sampled tokenizations for all the input sentences. Thus, we compute $\mathcal{L}_{\tilde{s}', \tilde{t}'} = q(f(\tilde{s}'), \tilde{t}')$ for the downstream model. We outline this training process for the NULM of the source side in Figure 2, and the training for the target side is explained in the same manner.

3 Experiment

To validate the applicability of the proposed method to various downstream tasks, we conducted experiments on text classification and machine translation tasks from existing literature. To compare our method with the existing methods that determine the appropriate tokenization for a specific downstream task, we employ OpTok (Hiraoka

	SP	SP+R	OpTok	Ours
Weibo(Zh)*	92.70	92.79	92.82	**93.06**
Twitter(Ja)*	85.89	86.51	**86.97**	86.92
Twitter(En)*	75.98	77.31	78.52	**78.88**
Genre(Zh)	44.19	47.95	48.18	**48.41**
Rating(Zh)	48.96	49.41	49.63	**49.76**
Genre(Ja)	46.82	49.84	50.15	**50.79**
Rating(Ja)	52.88	53.43	53.55	**53.69**
Genre(En)*	69.93	71.68	**71.88**	71.83
Rating(En)*	66.42	67.53	67.68	**67.90**
SNLI*	75.62	76.75	77.04	**77.05**

Table 1: Experimental results on text classification tasks (F1-score). SP and R indicate SentencePieec and subword regularization, respectively. The highest scores are highlighted in bold. Scores of SP+R and OpTok on datasets with * are quoted from Hiraoka et al. (2020).

et al., 2020) for text classification and DPE (He et al., 2020) for machine translation.

3.1 Text Classification

Settings We utilized ten datasets of text classification tasks in three languages. Weibo(Zh)[2], Twitter(Ja)[3], and Twitter(En)[4] are sentiment analyses on SNS corpora in Chinese, Japanese, and English, respectively. Genre and Rating are genre prediction and rating predictions from reviews posted on E-commerce corpora, respectively, in Chinese (Zhang et al., 2015)[5], Japanese (Rakuten, Inc., 2014), and English (He and McAuley, 2016)[6]. In addition, we employed the SNLI corpus (Bowman et al., 2015) to evaluate our method on the setting requiring two sentences as the input.

We focus on SentencePiece (SP) (Kudo and Richardson, 2018) and OpTok (Hiraoka et al., 2020) as other tokenizers for comparison with the proposed method. OpTok is a method to optimize tokenization for text classification by weighting a sentence vector with N-best tokenization. In addition, we trained each model with subword regularization (SP+R) (Kudo, 2018) for fair comparisons. In subword regularization, we used a sampled tokenization for the training phase and a 1-best tokenization for the inference phase.

For the downstream model, we used a BiLSTM encoder since we followed the experimental configurations of OpTok[7]. We used SentencePiece to construct vocabulary V[8] and initialized unigram probabilities of our NULM. The initial vocabulary sizes are 16K for Twitter(Ja) and Twitter(En) and 32K for the others. The number of tokenizations is $N = 3$, and the hyperparameters for subword regularization are $\alpha = 0.2$ and $k = \infty$. For the SNLI corpus, the system shares the same NULM for the premise and hypothesis and optimizes the NULM in the manner explained in Section 2.4.

Results Table 1 presents the experimental results on text classification. This table indicates that our proposed method surpasses OpTok in eight datasets. For the other two datasets, the performance of our method is comparable to OpTok. These results indicate that the proposed method is a better alternative to the existing tokenizers for text classification tasks. We consider that the difference in the performance is caused by the difference in strategy between the training the downstream models. OpTok trains the downstream model with a weighted sum of sentence vectors corresponding to N-best tokenization with their tokenization probabilities, but it uses 1-best tokenization in the inference. This gap might harm the downstream model. In contrast, since our method trains the downstream model with only one sampled tokenization, the downstream model receives one tokenization in both training and inference consistently. We consider that this consistency improves the performance.

3.2 Machine Translation

Settings For experiments on the machine translation task, we employ IWSLT and WMT corpora on eight language pairs. We pre-tokenized all the datasets except for the Chinese corpus with Moses Tokenizer[9], and we used jieba[10] for the Chinese corpus. We evaluate the performance of each method with detokenized BLEU with SacreBLEU (Post, 2018) after detokenization.

As a recent tokenizer for machine translation, we compare the proposed method with DPE (He et al., 2020), which tokenizes a target sentence, considering the source tokenization, in addition to SentencePiece. We employed the official implementation of DPE[11] and train the DPE model using SentencePiece tokenization. In the same as text

---

[2] `https://github.com/wansho/senti-weibo`

[3] `http://www.db.info.gifu-u.ac.jp/data/Data_5d832973308d57446583ed9f`

[4] `https://www.kaggle.com/c/twitter-sentiment-analysis2`

[5] `http://yongfeng.me/dataset/`

[6] `http://jmcauley.ucsd.edu/data/amazon/`

[7] Our implementation is based on the existing code: `https://github.com/tatHi/optok`

[8] We limit the size of the vocabulary to the half size of the initial vocabulary as well as the training of OpTok.

[9] `https://github.com/moses-smt/mosesdecoder`

[10] `https://github.com/fxsjy/jieba`

[11] `https://github.com/xlhex/dpe`

	Encoder	SP	SP+R	SP+R	Ours	SP+R	Ours
	Decoder	SP	SP+R	DPE	SP+R	Ours	Ours
IWSLT14	De-En	33.79	35.03	35.02	34.90	**35.78**	35.13
	En-De	28.09	29.13	29.39	29.56	**29.57**	29.30
IWSLT15	Vi-En	28.70	28.78	28.85	29.34	**29.69**	29.44
	En-Vi	30.87	31.60	31.63	31.41	**31.74**	31.70
	Zh-En	20.44	21.17	21.38	21.63	21.65	**21.89**
	En-Zh	14.40	15.25	15.21	15.45	**15.59**	15.31
IWSLT17	Ar-En	29.23	29.39	29.37	29.48	**30.04**	29.78
	En-Ar	15.45	17.75	17.83	**18.49**	18.18	18.21
	Fr-En	37.87	38.43	38.52	**38.82**	38.68	38.58
	En-Fr	37.95	39.83	39.90	40.01	**40.08**	39.68
WMT09	Hu-En	14.84	15.51	**15.75**	15.73	15.74	15.60
	En-Hu	11.02	12.14	12.30	12.30	**12.37**	12.33
WMT14	De-En	31.46	31.89	31.97	**32.19**	31.98	31.90
	En-De	27.10	27.41	27.49	**27.62**	27.52	27.44
WMT16	Ro-En	29.10	31.79	31.80	31.80	**31.83**	31.72
	En-Ro	21.78	24.05	24.29	24.36	**24.53**	24.03

Table 2: Results of experiments on machine translation task using IWSLT and WMT corpus (BLEU). We show the tokenization method for Encoder and Decoder. SP and R mean SentencePiece and subword regularization, respectively. The highest scores are highlighted in bold.

classification, we used subword regularization as a strong baseline.

For the downstream model, we used Transformer (Vaswani et al., 2017) implemented in Fairseq (Ott et al., 2019). For the IWSLT dataset, we used the small Transformer, and we created the initial vocabulary using SentencePiece with a 16K size of the vocabulary for each language. For the WMT dataset, we employed Transformer (base), and the size of the vocabulary is 32K. Similar to the case of text classification tasks, we initialized our NULM with the result of SentencePiece. The hyperparameters for subword regularization are $\alpha = 0.2$ for IWSLT, $\alpha = 0.5$ for WMT, and $k = \infty$ for both datasets. The number of tokenizations for the training of the proposed method is $N = 8$ for ISWLT and $N = 3$ for WMT.

In the training of NMT with DPE, we applied subword regularization for the source side language, similar to He et al. (2020). For the proposed method, we prepared three configurations: used our method only for a source side language, only for a target side language, and for both side languages.

Results Table 2 details the performance of each configuration. This table indicates that the system employing our approach achieves the best performance in most datasets. The setting where the proposed method is used only for the decoder side succeeds on many datasets. In contrast, when we use our method for both sides, the performance degrades. These results imply that it is challenging to optimize the tokenization of source and target languages simultaneously, and it can degrade the performance. We discuss the simultaneous optimization of source and target languages on NMT in Section 5.1.

4 Tokenization as Post-processing

Settings As described in Section 1, our proposed method can be applied as post-processing to an already trained model. In this section, we evaluate the effectiveness of optimizing our tokenizer for the trained model. Concretely, we trained the NULM with $\mathcal{L}_s$ in Eq.(3) without updating the parameters of the downstream model.

We conducted experiments on text classification (Sentiment) and machine translation (IWSLT15) tasks. We trained the downstream models used in Section 3 with subword regularization (Kudo, 2018). We trained the models with 30 epochs for text classification and 100 epochs for machine translation. After the training, we trained only our tokenizer with five epochs using the loss values computed by the trained models. Moreover, for text classification, we trained OpTok in the same manner as our proposed method as a baseline.

Results Table 3 details the performance of each method. This table indicates that the proposed method also improves the performance from the base model trained with subword regularization. The proposed method outperforms OpTok on two datasets of text classification. Moreover, the proposed method increases the BLEU scores consistently in machine translation. These results show that the proposed method is useful to improve the performance of the downstream model even if we

	SP+R	OpTok	Ours
Sentiment (F1)			
Weibo(Zh)	92.69	**93.08**	92.99
Twitter(Ja)	85.88	86.23	**86.28**
Twitter(En)	77.21	77.41	**77.77**
IWSLT15 (BLEU)			
Vi-En	28.82	-	**28.91**
En-Vi	30.48	-	**30.60**
Zh-En	21.55	-	**21.82**
En-Zh	14.57	-	**14.83**

Table 3: Improvement in the performance by optimizing tokenization as post-processing by OpTok and our method. SP+R indicates SentencePiece with subword regularization. The highest scores are highlighted in bold.

use a sufficiently trained model as the downstream model. In other words, since we do not necessarily require training for the proposed method with the downstream model from scratch, our proposed method can be applied to various situations such as the combination with a pre-trained model.

5 Discussion

5.1 Learning Both Encoder and Decoder

The results of the machine translation task (Section 3.2) reveal that the performance decreases when we incorporate our method into both the encoder and the decoder sides. We consider that the cause of this decrease to be the gap in the tokenization strategy between the source and target languages. In this section, we attempt to make it stable to train our method on both the encoder and decoder sides simultaneously with three possible strategies.

Enc→Dec: We train only the encoder-side NULM in the first 50 epochs, with the decoder-side NULM being frozen; then, we train the decoder-side NULM in the last 50 epochs, with the encoder NULM being frozen.

Dec→Enc: We train our method with the reversed version of the strategy Enc→Dec strategy.

Random: We randomly update either of the NULM on the encoder or that on the decoder sides with at a 0.5 ratio in each mini-batch training.

Table 4 presents the results of the experiments. These results indicate that the Enc→Dec strategy contributes to improving the performance of the simultaneous learning of tokenization on both sides. In particular, the scores of Vi-En, En-Vi, and Zh-En surpass the best scores reported in Table 2, indicating that the Enc→Dec strategy is effective for the training of our method. In contrast, the Dec→Enc strategy decreases the performance in many language pairs. The performance obtained using the Random strategy is slightly lower than that obtained using the original method (Both). From these results, we can conclude that it is effective for the machine translation task to learn the tokenization of each side step-by-step, specifically, from the encoder-side to the decoder-side, instead of optimizing both sides simultaneously.

	Both	Enc→Dec	Dec→Enc	Random
Vi-En	29.44	**30.22**	29.47	29.37
En-Vi	31.70	**31.78**	31.33	31.70
Zh-En	21.89	**21.99**	21.82	21.66
En-Zh	15.31	**15.54**	14.88	15.14

Table 4: Performance of NMT on the IWSLT15 datasets with three strategies for the simultaneous training of our method. The scores of Both are taken from the Ours-Ours column in Table 2. The highest scores are highlighted in bold.

5.2 Analysis of Tokenization

Optimized Tokenization In this section, we analyze the tokenization obtained using the proposed method on a machine translation task. Table 5 presents the comparison of tokenization among SentencePiece, DPE, and our method. We utilized the IWSLT15 Zh-En corpus for this comparison and tokenized English side sentences using each method. For our method, we only optimized the English side tokenization.

Table 5a presents a comparison of the tokenization on the source side between SentencePiece and the proposed method. Our method splits words into smaller segments than SentencePiece, which is the initial tokenization of our method. For example, our method cuts off the suffix from a stem word, such as splitting "don" into "do-n," "have" into "hav-e," and "hours" into "hour-s."

Table 5b presentas a comparison of the tokenization on the target side between SentencePiece, DPE, and the proposed method. Compared to the tokenization on the source side, our method does not split words into tiny units on the target side. The proposed method exhibits the same tendency of tokenization as DPE, such as splitting the past-suffix "-ed." However, the DPE tokenization contains smaller units than our tokenization; an example of this is the difference in the tokenization for "away."

Tokenization Granularity To compare the granularities of each tokenizer, we confirm the number of tokens in the corpus tokenized by each method. Table 6 presents the ratio of the number of tokens in the training corpus between the initial tokenization (SentencePiece) and the optimized tokeniza-

SP	Student s **don ’ t have** long **hours** of learning .
Ours	Student s **do n ’ t hav e** long **hour s** of learning .
TGT	学生 在 校 学习 时间 不 长 。

(a) Tokenziation difference for the source language (En-Zh)

SRC	引力 与 其它 力 分 隔 开来
SP	Gra vity **separate d away** from the other force s .
DPE	Gra vity **separat ed a way** from the other force s .
Ours	Gra vity **separat ed away** from the other force s .

(b) Tokenization difference for the target language (Zh-En)

Table 5: Comparison of English tokenization on Zh-En pairs using SentencePiece (SP), DPE, and our method. SRC and TGT indicate the tokenization of source and target side, respectively. The different tokenization is highlighted in bold.

tion (DPE and the proposed method). In the table, a value greater than 1 indicates an increase in the number of tokens compared to SentencePiece.

The results reveal that the number of tokens in the proposed method increases for the source side tokenization, which means that our method tokenizes a source corpus into small units by splitting morphemes, as shown in Table 5a.

For the tokenization of the target side, the ratio of the number of tokens for the proposed method is slightly smaller than the initial tokenization, other than for the En-Zh pair. We consider that our method seeks appropriate tokenization to aid in the decoding process while maintaining the granularity of the initial tokenization. With respect to the translation of the En-Zh pair, our method splits a Chinese sentence into smaller tokens. Chinese characters contain much more information than English characters, and the number of Chinese tokens in a sentence is smaller than that of English. We consider that this difference causes the increased tokens on the target side to use the same granularity as the source English corpus.

Compared with the tokenization by the proposed method, the number of tokens for the DPE varies for each language pair. DPE tokenization is more flexible than our method because DPE employs the Transformer and a special decoding algorithm for tokenization, whereas we simply use a unigaram language model and the Viterbi algorithm. In addition, DPE tokenizes the target sentence by directly considering the source tokenization by inputting a source sentence to the Transformer. In contrast, we use the target side NULM trained with the both side information to find the target side tokenization. Although our tokenization flexibility is limited, our method improves the performance on NMT tasks, as demonstrated in the experimental results.

Encoder Decoder	Ours SP+R	SP+R Ours	SP+R DPE
IWSLT14			
De-En	2.5353	0.9992	1.0439
En-De	1.3809	0.9996	0.9923
IWSLT15			
Vi-En	1.5320	0.9993	1.0428
En-Vi	1.4650	0.9999	0.9923
Zh-En	1.5175	0.9994	0.9907
En-Zh	1.3516	1.4713	1.0346
IWSLT17			
Ar-En	2.5350	0.9997	0.9952
En-Ar	1.4765	0.9994	0.9945
Fr-En	1.7194	0.9996	1.0001
En-Fr	1.5996	0.9997	0.9935

Table 6: Ratio of the number of tokens between initial tokenization (SentencePiece) and optimized tokenization (DPE and Ours) on the IWSLT corpora. SP+R denotes SentencePiece with subword regularization.

5.3 Effect of Hyperparameter N

The proposed method updates the NULM using N-best tokenized candidates. In this section, we confirm the effect of the hyperparameter N on the performance for the downstream tasks.

We conducted experiments on text classification and machine translation with different N, as mentioned in Section 3. Figures 3 and 4 show the results respectively. In these figures, we illustrate the difference from the performance of the model with the settings used in Section 3, i.e., $N = 3$ for text classification and $N = 8$ for machine translation.

For the text classification task, we confirm the effect of N on the sentiment analysis datasets. In Figure 3, we can observe that the number of N does not have a strong effect on the performance of the proposed method. In addition, the larger N leads to slightly better performance for the Japanese and English datasets. In contrast, the performance for the Chinese dataset decreases with a large N. We consider that this occurs because a Chinese sentence has more tokenization candidates than the others, and the optimization of tokenization becomes unstable with larger N.

Compared to the existing OpTok method (Hiraoka et al., 2020) the proposed method is robust to large N. As described in Section 3, our method avoids the gap between training and inference in terms of the weighting strategy. Because the proposed method uses one sampled tokenization to train the downstream model, the number of N does not affect the text classification performance. The experiment with various N verifies that our method is superior to Hiraoka et al. (2020).

For the machine translation task, we conduct ex-

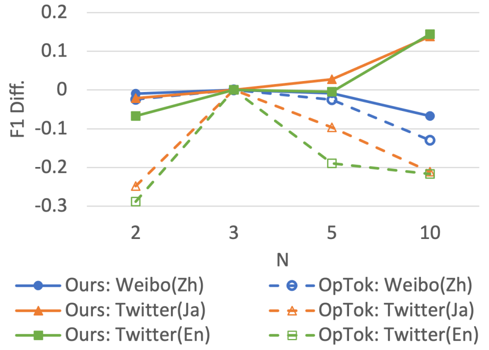

Figure 3: Difference in performance on text classification tasks against N.

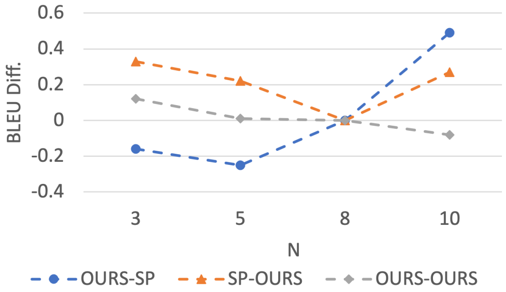

Figure 4: Difference in performance on machine translation tasks (Vi-En) against N.

periments on the Vi-En pair of IWSLT15. Figure 4 illustrates that the number of N does not have a strong effect on the performance when we use the proposed method solely for the target side (SP-OURS). When we incorporate our method to the source side (OURS-SP), the performance increases with a large N. We consider that the proposed method is able to seek appropriate tokenization from the large search space when we set a large number as N because the neural encoder of NMT allows various tokenizations for its input. When we use our method for both the encoder and the decoder (OURS-OURS), the performance decreases slightly with higher N. We consider that optimization of the tokenization of both sides with a large N becomes unstable because tokenization on the source side varies vastly during training.

6 Related Work

Many researchers have tackled the problem of optimizing tokenization, especially in the machine translation field. For statistical machine translation, Nießen and Ney (2004) and Goldwater and McClosky (2005) attempted to obtain good tokenization using hand-crafted linguistic information. Some studies explored appropriate tokenization using alignment information between the source and target languages (Xu et al., 2008; Chung and Gildea, 2009; Nguyen et al., 2010).

Recent studies have attempted to obtain appropriate tokenization for the downstream task using neural networks. Gowda and May (2020) analysed the optimal granularity of tokenization on NMT. Salesky et al. (2020) proposed Incremental-BPE, which automatically explores the appropriate granularity of BPE tokenization. They stopped the merge operation of the BPE depending on the loss on a validation split. He et al. (2020) proposesd DPE, which obtains the appropriate tokenization of a target language depending on the tokenization of the source side on the NMT. Our method is different from DPE in that our method can optimize tokenization considering the parameters of the downstream model. Moreover, our method can be applied to both the source and target languages of the machine translation task. Hiraoka et al. (2020) proposed OpTok, which optimizes the tokenizer and the downstream model on text classification simultaneously. We extend their idea to be applicable to any downstream task, including machine translation. Moreover, our method uses a different training strategy to that used in OpTok. We split the training loss for the downstream loss and tokenization loss, as mentioned in Section 2, and the experimental results demonstrate that our strategy is superior to OpTok.

We employ subword regularization to train the downstream model. Kudo (2018) proposed training a model by sampling tokenization with a unigram language model, and Provilkov et al. (2020) modified this idea to use the BPE (Sennrich et al., 2016) process to yield various tokenizations. Hiraoka et al. (2019) applied subword reguralization for text classification tasks.

7 Conclusion

We propose a novel method for optimizing tokenization by considering downstream tasks, such as a training corpus and a downstream model. Our method is the first approach to explore appropriate tokenization for any downstream task. Experimental results demonstrate that the proposed method achieves higher performance than existing systems with respect to text classification and machine translation tasks. Because the proposed method is applicable to any architecture using loss for its optimization, we expect our method to improve the performance for other NLP tasks.

Acknowledgments

These research results were obtained from the commissioned research by National Institute of Information and Communications Technology (NICT), Japan.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada. Association for Computational Linguistics.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232, Columbus, Ohio. Association for Computational Linguistics.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 718–726, Singapore. Association for Computational Linguistics.

Miguel Domingo, Mercedes García-Martínez, Alexandre Helle, Francisco Casacuberta, and Manuel Herranz. 2018. How much does tokenization affect neural machine translation? *CoRR*, abs/1812.08621.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia. Association for Computational Linguistics.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 676–683, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 507–517, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Tatsuya Hiraoka, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Stochastic tokenization with a language model for neural text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1620–1629, Florence, Italy. Association for Computational Linguistics.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2020. Optimizing word segmentation for downstream task. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1341–1351, Online. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. http://taku910.github.io/mecab/.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore. Association for Computational Linguistics.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*.

ThuyLinh Nguyen, Stephan Vogel, and Noah A. Smith. 2010. Nonparametric word segmentation for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 815–823, Beijing, China. Coling 2010 Organizing Committee.

Sonja Nießen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Rakuten, Inc. 2014. Rakuten dataset. Informatics Research Data Repository, National Institute of informatics. (dataset).

Elizabeth Salesky, Andrew Runge, Alex Coda, Jan Niehues, and Graham Neubig. 2020. Optimizing segmentation granularity for neural machine translation. *Machine Translation*, pages 1–19.

Steven L Scott. 2002. Bayesian methods for hidden markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Kazuma Takaoka, Sorami Hisamoto, Noriko Kawahara, Miho Sakamoto, Yoshitaka Uchida, and Yuji Matsumoto. 2018. Sudachi: a Japanese tokenizer for business. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A morphological analysis toolkit for scriptio continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised Chinese word segmentation for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1017–1024, Manchester, UK. Coling 2008 Organizing Committee.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Shuailong Liang. 2019. Subword encoding in lattice LSTM for Chinese word segmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2720–2725, Minneapolis, Minnesota. Association for Computational Linguistics.

Yongfeng Zhang, Min Zhang, Yi Zhang, Guokun Lai, Yiqun Liu, Honghui Zhang, and Shaoping Ma. 2015. Daily-aware personalized recommendation based on feature-level time series analysis. In *Proceedings of the 24th international conference on world wide web*, pages 1373–1383.

Method	Tokenization
SentencePiece	This didn ' t have full **seasons** . I expected **more** than 6 **episodes** on this because I **liked** this series .
Genre (Gold: Movies_and_TV)	
OpTok	This didn ' t have full **seasons** . I expected **more** than 6 **episode s** on this because I **liked** this series .
Ours	This didn ' t have full **season s** . I expected **more** than 6 **episode s** on this because I **liked** this series .
Rating (Gold: 1)	
OpTok	This didn ' t have full **seasons** . I expected **mor e** than 6 **episodes** on this because I **like d** this series .
Ours	This didn ' t have full **seasons** . I expected **more** than 6 **episodes** on this because I **liked** this series .

Table 7: Tokenization examples on the text classification task. Bold highlights the difference of tokenization among methods.

A Detailed Experimental Settings

For all the experiments, we implement the proposed method using PyTorch, and we run all the experiments on NVIDIA Tesla V100 (16 GiB). For the initialization of the NULM, we terminate the pretraining when the loss is less than 1×10^{-7} or the training epoch achieves the maximum number (100,000).

A.1 Text Classification

In Section 3.1, we use two new datasets in addition to the datasets used in the existing research (Hiraoka et al., 2020). We prepare both datasets in the same manner as that used for Genre(En) and Rating(En).

For Genre(Zh) and Rating(Zh), we use 13 genres that contain a sufficient number of reviews and sample 30,000 reviews from each genre, balancing the number of ratings and limiting the number of characters in the review to 100. The dataset contains 390,000 reviews, and we split it at a ratio of 8:1:1 for training, validation, and testing. Each review has ratings from 1 to 5 attached by reviewers, and we use the sampled dataset for a rating prediction task and a genre prediction task.

For Genre(Ja) and Rating(Ja), we use 21 genres that contain a sufficient number of reviews and sample 5,000 reviews from each genre and rate, limiting the number of characters in the review to 100. The dataset contains 525,000 reviews, and we split it at a ratio of 8:1:1 for training, validation, and testing. Each review has ratings from 1 to 5 attached by reviewers, and we use them for rating and genre prediction tasks.

We conduct experiments on text classification tasks under the same settings as those used in the existing literature (Hiraoka et al., 2020). Thus, we conduct a text classification with BiLSTM encoders whose hidden size is 256. The size of the word embedding is 64, and we set the batch size to 256 and the maximum training epoch to 20. The pretrained word embeddings are frozen in the training of text classification, and both NULM and the downstream model share word embeddings.

A.2 Machine Translation

For experiments on machine translation, we do not freeze the word embeddings. We empirically find that the training becomes unstable when word embeddings are shared between NULM and the downstream model without freezing. Therefore, we prepare different word embeddings for NULM and the downstream model, called the Transformer. We set the word embedding size to 64 for the NULM. We make the mini-batch by specifying the number of maximum tokens, and we set it to 1,000. The maximum number of training epochs is 100 for all the experiments, and we average the parameters of the last 10 epochs for evaluation.

B Tokenization for Text Classification

We present examples for the tokenization on text classification tasks, Genre/Rating(En), in Table 7. As both Genre and Rating datasets are created from the same review corpus, we can confirm whether each method can tokenize a sentence depending on the task, which might be a genre prediction or a rating prediction. We compare the tokenization by SentencePiece, OpTok, and our method.

The tendency for tokenization by our method is similar to that by OpTok because our method is based on OpTok. In the example, both OpTok and our method split a suffix “s” from “episodes” only on the genre prediction task. This example implies that both methods yield tokenization that includes task-specific word such as “episode” for the “Movies_and_TV” genre. In addition, our method tokenizes “seasons” into “season-s,” which is also related to the movie genre.

With respect to the rating prediction, our method does not split “liked” into “like,” which might be helpful for predicting ratings, whereas OpTok does. We consider that our method uses the original word to distinguish the verb from the adjective/adverb “like.”

How does Attention Affect the Model?

Cheng Zhang[1],**Qiuchi Li**[2], **Lingyu Hua**[3], **Dawei Song** [3*]
[1]College of Intelligence and Computing, Tianjin University, China
[2]Department of Information Engineering, University of Padua, Italy
[3]School of Computer Science, Beijing Institute of Technology, China
`cheng.zhang@tju.edu.cn`
`qiuchili@dei.unipd.it`
`hualingyu@hotmail.com`
`dwsong@bit.edu.cn`

Abstract

The attention layer has become a prevalent component in improving the effectiveness of neural network models for NLP tasks. Figuring out why attention is effective and its interpretability has attracted a widespread deliberation. Current studies mostly investigate the effect of attention mechanism based on the attention distribution it generates with one single neural network structure. However they do not consider the changes in semantic capability of different components in the model due to the attention mechanism, which can vary across different network structures. In this paper, we propose a comprehensive analytical framework that exploits a convex hull representation of sequence semantics in an n-dimensional Semantic Euclidean Space and defines a series of indicators to capture the impact of attention on sequence semantics. Through a series of experiments on various NLP tasks and three representative recurrent units, we analyze why and how attention benefits the semantic capacity of different types of recurrent neural networks based on the indicators defined in the proposed framework.

1 Introduction and Motivation

The first appearance of the attention mechanism in natural language processing (NLP) can be traced back to its successful application in Neural Machine Translation (NMT). Bahdanau et al. (2014a) proposed an attention mechanism in an Encoder-Decoder model, which achieved great success, and showed that attention weight produced in this mechanism improved the interpretability of the model by providing a way of aligning the source and target languages through a simple quantitative analysis. Subsequently, the assumption that attention could improve the interpretability and transparency of a model was acquiesced by many later works, such as AEN (Song et al., 2019) (applied to targeted sentiment classification), ATAE-LSTM (Wang et al., 2016) (applied to aspect-level sentiment classification), and CMLA (Wang et al., 2017) (applied to semantic sentiment analysis).

More recently, this hypothetical premise has aroused controversies. For example, Serrano and Smith (2019) and Jain and Wallace (2019) used an erasure-based approach and advocated the attention weight does not necessarily correspond to importance. Wiegreffe and Pinter (2019) and Vashishth et al. (2019) considered attention to be interpretable, using a more model-driven approach and manual verification. These investigations focus on whether the attention distribution is unique and the correlation between attention weight and model prediction results, based on a similar neural network setting that consists of an embedding layer, a specific Recurrent Neural Network (RNN) and an attention component. Complex components such as the encoder-decoder structure were removed from the network as they may bias the analysis on the effect of attention weights. However, these works fail to explain two critical issues as follows:

(1) Neglecting the changes in the rest of the model before and after introducing attention, especially the word embedding layer and the RNNs' hidden layer. Figure 1 shows the transition before and after the introduction of attention. When a model does not introduce attention, the model generates an embedding sequence $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ from the original one-hot word representation. Subsequently, the sequence $\mathcal{E}$ is processed by a specific RNN and converted into a hidden layer sequence $\mathcal{H} = \{h_1, h_2, ..., h_n\}$, which is used to produce the output. The introduction of attention will cause the model to change the gradient during the back-propagation in the training phase, which will lead to the embedding and hidden sequences to move away from the pre-

---

[*] Corresponding Author: Dawei Song

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 256–268
August 1–6, 2021.

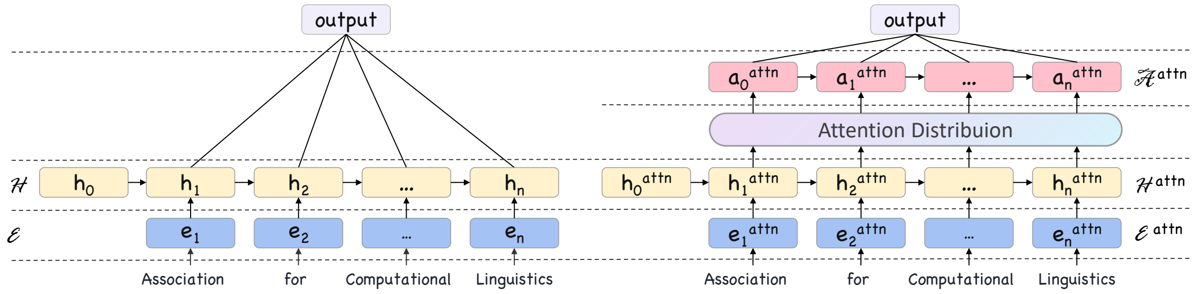

Figure 1: The transformation before and after the introduction of the attention in the RNNs.

vious values $\mathcal{E}$ and $\mathcal{H}$ when the model training step is done. The new embedding sequence is represented by $\mathcal{E}^{attn} = \{e_1^{attn}, e_2^{attn}, ..., e_n^{attn}\}$, and the hidden layer sequence is represented by $\mathcal{H}^{attn} = \{h_1^{attn}, h_2^{attn}, ..., h_n^{attn}\}$. After the attention layer, $\mathcal{H}^{attn}$ is adjusted by the attention distribution to produce a new sequence $\mathcal{A}^{attn} = \{a_1^{attn}, a_2^{attn}, ..., a_n^{attn}\}$. At this point, $\mathcal{A}^{attn}$ is used to generate the output of the model.

The existing works have focused on whether the attention distribution is unique or reasonable (if it is not unique). However, they ignore the extent of semantic changes in the sequences caused by the attention mechanism, including the changes in the embedding sequence ($\mathcal{E} \rightarrow \mathcal{E}^{attn}$), in the hidden layer sequence ($\mathcal{H} \rightarrow \mathcal{H}^{attn}$), and even in the emerging attention sequence ($\mathcal{A}^{attn}$). We posit that such ignorance would lead to an unfair and biased analysis of the attention.

(2) Lacking a systematic study of the attention effect on different types of RNNs. The attention layer is compatible with various types of RNNs, regardless of which recurrent unit out of Vanilla-RNN (Mikolov et al., 2010), LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014), is used, or whether it has a uni-directional or bi-directional structure. Although the existing works have experimented on many datasets, they solely focus on a single type of RNN preceding the attention layer at a time. We argue that a comprehensive comparison of the changes before and after introducing attention to different types of RNNs mentioned above will better reveal the intrinsic interpretability of attention.

To address these two issues, we propose to explore the effect of attention from a new perspective by conducting a systematic investigation on the semantic changes across different sequences of a RNN model with or without attention, and comparing the differences in the changes across mainstream recurrent units. Based on the analysis results, we expect to better understand what happens before and after the introduction of attention into the model.

The proposed analysis requires a comprehensive framework with reasonable metrics to evaluate the quality of sequence semantics based on their vector representations. For this purpose, we adopt the concept of Convex Hull in n-dimensional Semantic Euclidean Space ($\mathbb{SR}^n$) (Zhang et al., 2020) to represent the semantics of a sequence. Since an attention mechanism always produces a point in the convex hull of its preceding hidden units, we can establish suitable metrics based on the convex hull formed by a sequence of vectors in $\mathbb{SR}^n$, to facilitate the analysis of attention effect. Section 2 will briefly introduce the Semantic Euclidean Space and the convex hull representation of the sequence semantics. In Section 3, we analyze the attention mechanism and establish the relationship between the attention weight and the sequence meaning (as convex hull in $\mathbb{SR}^n$). Section 4 formulates a set of indicators to analyze the semantic changes before and after attention. With the proposed framework, we conduct comparative experiments on various datasets concerning text classification and sentiment analysis tasks in Section 5. Based on the experimental results, we conduct in-depth analysis from the perspective of why and how the attention mechanism benefits the semantic capacity of different recurrent units of RNNs.

2 Background

Zhang et al. (2020) proposed an n-dimensional Semantic Euclidean Space ($\mathbb{SR}^n$), which defines the mapping relationship between points in an Euclidean Space ($\mathbb{R}^n$) and their semantics. As a semantic extension of $\mathbb{R}^n$, $\mathbb{SR}^n$ is defined as:

$$\mathbb{SR}^n = \{\forall x = (x_1, ..., x_n) \in \mathbb{R}^n | x \to \text{semantics}\} \tag{1}$$

In $\mathbb{SR}^n$, the points are divided into *specific semantic points* and *abstract semantic points*. A specific semantic point has a word corresponding to it, which can also be regarded as a word embedding. An abstract semantic point does not have a specific word corresponding to it, such as a point generated by the hidden layer of RNNs.

Zhang et al. (2020) then proposed to use the convex hull and centroid of points in $\mathbb{SR}^n$ to measure the **meaning** and **central idea** of a sequence of words. It provides a theoretical basis for exploring the semantic changes that occur before and after introducing attention into a model.

2.1 Meaning of a Sequence

Definition. *The meaning of a sequence composed of semantic points is represented by the convex hull composed of these points.*

Given a sequence $\mathcal{X}$ composed of semantic points, its meaning, denote as **ME**$(\mathcal{X})$, is formulated as:

$$\text{ME}(\mathcal{X}) = \text{Conv}(\mathcal{X}) \tag{2}$$

$\text{Conv}(\mathcal{X})$ denotes the convex hull of a finite point set $\mathcal{X}$, as the set of all convex combinations of the points (Faux and Pratt, 1979). In a convex combination, each point x_i in $\mathcal{X}$ is assigned with a weight or coefficient α_i in such a way that the coefficients are all non-negative and sum to one. These weights are used to produce a weighted average of points. It is formulated as:

$$\text{Conv}(\mathcal{X}) = \left\{ \sum_{i=1}^{|\mathcal{X}|} \alpha_i x_i \middle| \alpha_i \geq 0 \wedge \sum_{i=1}^{|\mathcal{X}|} \alpha_i = 1 \right\} \tag{3}$$

The mapping between the definition of the convex hull and the meaning of a sequence is intuitive. A sentence (sequence) consists of words (semantic points). In addition to the semantics expressed by the individual words, a sentence should also include the implicit semantics (abstract semantic points) produced by all possible combinations of words.

2.2 Central Idea of a Sequence

Definition. *The central idea of a sequence composed of semantic points is represented by the centroid of the sequence's meaning.*

The central idea of a sequence $\mathcal{X}$ of semantic points is denoted as $\text{Centroid}(\mathcal{X})$, formulated as:

$$\text{CI}(\mathcal{X}) = \text{Centroid}(\text{ME}(\mathcal{X})) \tag{4}$$

The centroid a subset $\mathcal{X}$ of $\mathbb{R}^n$ is the mean position of all the points in all coordinate directions. It is computed as:

$$\text{Centroid}(\mathcal{X}) = \frac{\int x g(x) dx}{\int g(x) dx} \tag{5}$$

where the integrals are taken over the whole space $\mathbb{R}^n$, and g is the characteristic function, which is 1 if a point is inside $\mathcal{X}$ and 0 otherwise (Protter and Morrey, 1977).

However, the central idea of a sequence needs to be calculated as $\text{Centroid}(\text{Conv}(\mathcal{X}))$, instead of $\text{Centroid}(\mathcal{X})$ directly, to guarantee that the central idea of the sequence lies within the convex hull (meaning) of the sequence. In contrast, even though the geometric centroid of a convex object always lies within the area representing its meaning, a non-convex object might also have a centroid that is outside the area, which is undesirable. As introduced above, ME scopes the meaning of a sequence as an area in $\mathbb{SR}^n$, while the central idea of the sequence should be at the centre of the ME area.

The central idea of a sequence can be considered as a "summary" of the sentence's meaning. Operationally, it is the centroid of the convex hull representation of the sentence meaning, within a $\mathbb{SR}^n$. Take the phrase "The Association for Computational Linguistics" as an example, the central idea of this phrase can be summarized as a semantic point, which corresponds to the abbreviation "ACL". Considering another phrase, "good enough but not excellent", the central idea of this phrase also can be summarized as a semantic point, but for the time being, there is no word that corresponds to this semantic point. Perhaps with the development of natural language, people will soon create a word to describe this semantic point. This is actually the specific semantic point and abstract semantic point defined in $\mathbb{SR}^n$. More explanations about $\mathbb{SR}^n$ can be found in Zhang et al..

3 Attention With Convex Hull

Motivated by the ability of $\mathbb{SR}^n$ to measure the meaning and central idea of sequences, we will theoretically analyze the role of attention from the perspective of semantic change.

Take the well-known Scaled Dot-Product attention (Vaswani et al., 2017) as an example (this will be abbreviated as **dot-attn** later). When a sequence $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ passes through a dot-attn layer, the specific calculation process is shown as follows:

$$\alpha_i = softmax(\frac{x_i X}{\sqrt{m}}) \tag{6}$$

$$y_i = \alpha_i X \tag{7}$$

$X \in \mathbb{R}^{n \times m}$ denotes the matrix of word vectors corresponding to the input sequence $\mathcal{X}$, where m is the dimensionality of word vectors. Essentially this process can be described as the following two steps:

1. Construct an attention distribution $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ through the input sequence $\mathcal{X}$ and the softmax function,
2. Use the attention weight and $\mathcal{X}$ to generate a new sequence $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$, which is called an attention sequence.

α_i is a probability distribution generated from the softmax function, which ensures that each component in it will be in the interval $(0, 1)$ and the components will add up to 1. Focusing on a specific vector y_i in $\mathcal{Y}$, it can be expressed as follows:

$$y_i = \sum_{j=1}^{n} \alpha_i^j x_j \left| \sum_{j=1}^{n} \alpha_i^j = 1, \alpha_i^j \geq 0 \right. \tag{8}$$

Comparing Formula 8 with Formula 3, we can find that under the action of dot-attn, the process of converting $\mathcal{X}$ to $\mathcal{Y}$ is indeed a process of continuously selecting new semantic points from the convex hull of $\mathcal{X}$ (i.e., the meaning of $\mathcal{X}$, ME($\mathcal{X}$)) to form a new semantic sequence $\mathcal{Y}$. Therefore, the new sequence $\mathcal{Y}$ is a semantic transformation of the original sequence $\mathcal{X}$ to some extent. An example of $\mathcal{X} \xrightarrow{\text{dot-attn}} \mathcal{Y}$ is shown in Figure 2.

Furthermore, although the convex hull of $\mathcal{X}$ can be used to express the meaning of a sentence, the model usually uses $\mathcal{X}$ to construct a vector $c = \frac{1}{n}\sum_{i=1}^{n} x_i$ as representation of a sentence, followed by a dense layer and activation function to generate prediction result. The form of sentence representation c is consistent with the definition of the central idea in $\mathbb{SR}^n$. Therefore, from the sentence representation's perspective, attention is to transform the central idea expressed by the original sequence (CI($\mathcal{X}$)) to a new semantic point CI($\mathcal{Y}$) (the central idea of $\mathcal{Y}$). The offset from the yellow diamond to the red triangle in Figure 2 represents the conversion from CI($\mathcal{X}$) to CI($\mathcal{Y}$).

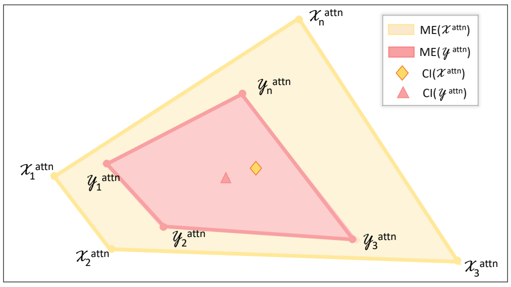

Figure 2: Use attention to convert sequence $\mathcal{X}$ to sequence $\mathcal{Y}$. The meaning of $\mathcal{X}$ is represented by the yellow shaded part, and the meaning of $\mathcal{Y}$ is represented by the red part.

In summary, the model has undergone the following changes after the introduction of attention:

1. Attention adjusts the meaning expressed by the original sequence by adjusting each semantic point in the sequence.
2. Attention changes the central idea (an instance representation) of the original sequence.

Through the above analysis, we have a deeper understanding of how attention transforms the original input from the perspective of the convex hull at the theoretical level. It is important to note that our proposed attention analysis framework above is applicable to other forms of attention, such as $\tanh$ attention (Zhou et al., 2016). No matter a popular dot-attn or a traditional $\tanh$ attention is used, they can be regarded as firstly adjusting the sequence $\mathcal{X}$ to sequence $\mathcal{Y}$, and then further averaging them to make predictions. The subtle difference between them lies in the dimensions of attention distributions. For an input sequence, the dot-attn generates a 2-dimensional distribution, while the $\tanh$ attention generates a 1-dimensional distribution.

However, only the above analysis framework is not enough. During the training process, due to the introduction of attention, the gradient change in the back-propagation process will cause the original sequence $\mathcal{X}$ to be converted into a new sequence $\mathcal{X}^{attn}$, and this has also been ignored in previous studies. To this end, we will first construct relevant evaluation indicators, and then give our complete analysis framework based on the theoretical analysis of attention in this section.

4 Assessing the Effect of Attention

Following the typical settings in this area, we use an RNN model as the basis to systematically analyze the changes between different sequences in the process of introducing attention to the model. We specify multiple indicators to measure these changes and accordingly present our analysis framework.

4.1 Various Sequences Before and After the Introduction of Attention

As shown in Figure 3, the model takes the one-hot representation of the word sequence as the initial input. When attention is not used by the model, the model contains an embedding sequence $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ inferred from a dense layer and a hidden sequence $\mathcal{H} = \{h_1, h_2, ..., h_n\}$ produced by an RNN. When an attention mechanism is introduced, the model weight changes due to the gradient changes during the training process. Hence, $\mathcal{E}$ and $\mathcal{H}$ will be converted to new sequences $\mathcal{E}^{attn} = \{e_1^{attn}, e_2^{attn}, ..., e_n^{attn}\}$ and $\mathcal{H}^{attn} = \{h_1^{attn}, h_2^{attn}, ..., h_n^{attn}\}$. The introduction of attention will be further transformed $\mathcal{H}^{attn}$ into an attention sequence $\mathcal{A}^{attn} = \{a_1^{attn}, a_2^{attn}, ..., a_n^{attn}\}$. For the above five sequences, there is the following progressive relationship:

$$\mathcal{E} \rightarrow \mathcal{H} \tag{9}$$

$$\mathcal{E}^{attn} \rightarrow \mathcal{H}^{attn} \rightarrow \mathcal{A}^{attn} \tag{10}$$

The differences between the above two links of sequences should be carefully examined to explore the impact of attention on the model. In this work, we first define a series of indicators to measure the semantic expression ability of an independent sequence and the semantic relationship between two sequences that belong to the same link. A framework is proposed to systematically compare the differences between the two links to assess the effect of attention.

4.2 Degree of Semantic Unsaturation

In $\mathbb{SR}^n$, the meaning of a sequence $\mathcal{X}$ of length $|\mathcal{X}|$ is calculated by ME. We define the degree of semantic unsaturation of a sequence as follow:

$$\text{DSU}(\mathcal{X}) = \frac{\text{ME}(\mathcal{X})}{|\mathcal{X}|} \tag{11}$$

$\text{DSU}(\mathcal{X})$ reflects the degree of semantic unsaturation regarding $\mathcal{X}$. Normally, the smaller the semantic space contained in the meaning of a sequence, the more precise the semantics expressed. Specifically, for sequences $\mathcal{X}$ and $\mathcal{Y}$ have same sequence length, if the meaning expressed by $\mathcal{X}$ is more precise than the meaning expressed by $\mathcal{Y}$, i.e. $\text{ME}(\mathcal{X}) < \text{ME}(\mathcal{Y})$, then $\text{DSU}(\mathcal{X})$ is less than $\text{DSU}(\mathcal{Y})$, which means that the degree of unsaturation of $\mathcal{X}$ is lower than $\mathcal{Y}$. For this reason, the smaller $\text{DSU}(\mathcal{X})$, the better.

4.3 Semantic Coverage

For two sequences $\mathcal{X}, \mathcal{Y}$, the sequence $\mathcal{Y}$ is a semantic transformation of the previous sequence $\mathcal{X}$ (this transformation may be synonymous transformation or even semantic extraction), we use semantic coverage (SC) (Zhang et al., 2020) to indicate the overlap between two sequences:

$$\text{SC}(\mathcal{X}, \mathcal{Y}) = \text{ME}(\mathcal{X}) \cap \text{ME}(\mathcal{Y})$$

Since $\mathcal{X}$ is the original sequence and $\mathcal{Y}$ is the converted sequence, then three indicators Semantic Coverage Precision (SCP), Semantic Coverage Recall (SCR), and Semantic Coverage F-Measure (SCF) can be naturally defined to observe the changes between the two sequences:

$$\text{SCP}(\mathcal{X}, \mathcal{Y}) = \frac{\text{SC}(\mathcal{X}, \mathcal{Y})}{\text{ME}(\mathcal{Y})} \tag{12}$$

$$\text{SCR}(\mathcal{X}, \mathcal{Y}) = \frac{\text{SC}(\mathcal{X}, \mathcal{Y})}{\text{ME}(\mathcal{X})} \tag{13}$$

$$\text{SCF}(\mathcal{X}, \mathcal{Y}) = \frac{2 \times \text{SCP} \times \text{SCR}}{\text{SCP} + \text{SCR}} \tag{14}$$

4.4 Central Idea Offset

In addition to the difference in meaning between two sequences, it is crucial to check the deviation of the central idea between the two sequences $\mathcal{X}, \mathcal{Y}$. The offset distance between the central idea of $\mathcal{Y}$ and that of the original sequence $\mathcal{X}$ is called Central Idea Offset (CIO), formulated as follows:

$$\text{CIO}(\mathcal{X}, \mathcal{Y}) = \|\text{CI}(\mathcal{X}), \text{CI}(\mathcal{Y})\| \tag{15}$$

4.5 Analysis Framework

Base on the definition of the above five indicators, we propose a framework to analyze the impact of

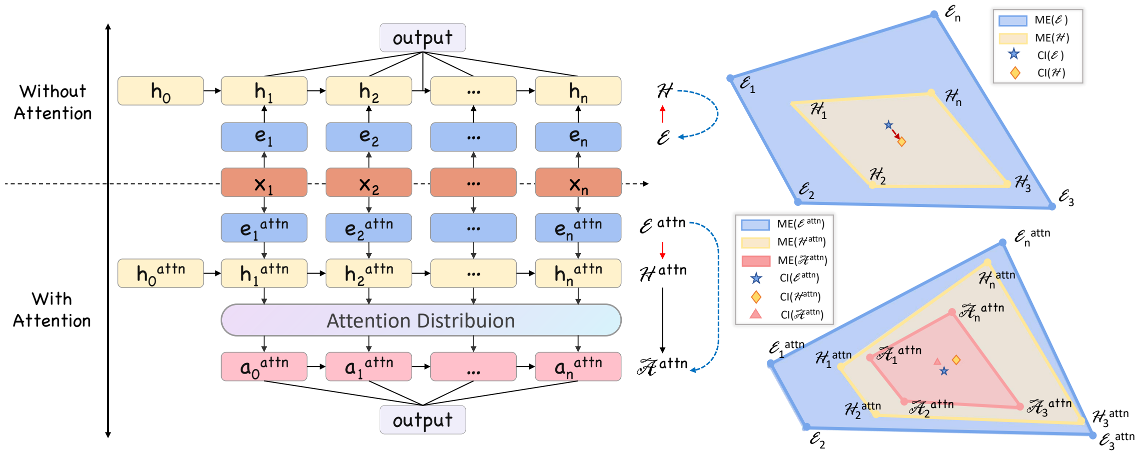

Figure 3: Splice the two models before and after the attention is introduced in a symmetrical manner. The word vector sequence and hidden layer sequence linked by the red arrow in the figure represent the observation from the corresponding perspective. The sequence linked at both ends of the blue dashed arrow represents observation from a shift perspective.

attention on a certain model. Recalling the two links in Eq. 9, since the number of sequences contained in each link is different, we propose two perspectives for comparison: the corresponding perspective and the shift perspective.

4.5.1 The Corresponding Perspective

The introduction of attention to a model has caused changes in its embedding sequence ($\mathcal{E} \rightarrow \mathcal{E}^{attn}$) and its hidden layer sequence ($\mathcal{H} \rightarrow \mathcal{H}^{attn}$). This observation on the changes in the corresponding layers of the model is called the corresponding perspective. Using Δ to represent the difference, $\Delta(\rho(\mathcal{E}), \rho(\mathcal{E}^{attn}))$ reflects the influence of the introduction of attention on the embedding layer from the corresponding perspective, and similarly for $\mathcal{H}$ and $\mathcal{H}^{attn}$.

In addition to the changes on a single sequence, the difference between the links between adjacent sequences ($\mathcal{E} \rightarrow \mathcal{H}$, $\mathcal{E}^{attn} \rightarrow \mathcal{H}^{attn}$) can also be used to observe the impact of attention. For example, $\Delta(\text{SCP}(\mathcal{E}, \mathcal{H}), \text{SCP}(\mathcal{E}^{attn}, \mathcal{H}^{attn}))$ is employed to compare the changes of semantic coverage precision between embedding layer and hidden layer before and after introduction of attention. This difference can also be computed for SCP, SCF and CIO.

4.5.2 The Shift Perspective

According to the analysis in Section 3, before attention is added, the generated attention sequence $\mathcal{H}$ is actually a conversion of the embedding layer. In the presence of attention, the embedding sequence $\mathcal{E}^{attn}$ is converted to the sequence $\mathcal{H}^{attn}$, which is further transformed to $\mathcal{A}^{attn}$ by the attention mechanism. Therefore, the difference between $\text{CIO}(\mathcal{E}, \mathcal{H})$ and $\text{CIO}(\mathcal{E}^{attn}, \mathcal{A}^{attn})$ can be used to reflect the influence of attention on the overall semantic shift along the link.

Dataset	Train/Valid/Test Size	Vocab Size	Label Size
AG_News	96000/24000/7600	95812	4
SST	8544/1101/2210	16583	4

Table 1: Dataset statistics.

In the mean time, an input sentence is finally converted into $\mathcal{H}$ or $\mathcal{A}^{attn}$ to express the meaning of the sentence, so we can alternatively express this change by $\Delta(\text{DSU}(\mathcal{H}), \text{DSU}(\mathcal{A}^{attn}))$.

5 Exploring the Attention

We first introduce the dataset and models used in the experiments and then explore the impact of attention using the analysis framework above.

5.1 Experimental Setup

In order to make our analysis concise, our experiments focused on both text classification task (Stanford Sentiment Treebank (SST) (Socher et al., 2013)) and sentiment analysis task (AGNews [1]). In the future, we will extend our work to more data

---

[1] `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

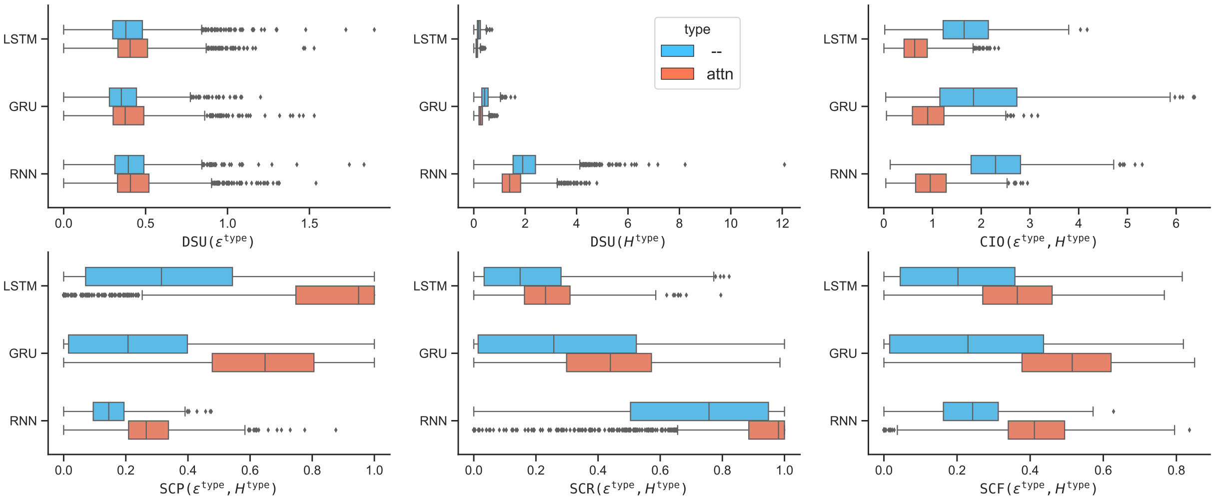

Figure 4: From the corresponding perspective observe the impact of the introduction of attention on the model. In each bin, top (blue) is the model without attention, bottom (red) is the model with attention.

	AG_News		SST	
	-	dot	-	dot
LSTM	0.969	0.976	0.918	0.946
Bi-LSTM	0.974	0.977	0.941	0.951
GRU	0.966	0.973	0.938	0.946
Bi-GRU	0.968	0.970	0.945	0.947
RNN	0.953	0.967	0.848	0.948
Bi-RNN	0.971	0.972	0.892	0.948

Table 2: The accuracy of different types of recurrent units before and after the introduction of attention.

sets, especially the machine translation dataset in the encoder-decoder model.

Since the AGNews dataset does not have a predefined validation set, the training set is split into a training set and validation set at a ratio of 8:2. The statistics of datasets are shown in Table 1. For each dataset, the base model we used for training is shown in Figure 1. It has an embedding layer for convert one-hot representation to distribution representation, a specific RNN-layer (recurrent units can be Vanilla-RNN, LSTM or GRU. The overall structure can be uni-directional or bi-directional, resulting in 6 different combinations.), without or with a dot-attn layer, followed by an additive layer and softmax prediction. The accuracy results of these models on the validation set are shown in Table 2, we calculate and analyze our indicators on the test set (The distribution of sentence length in the test set is shown in Appendix A). It should be pointed out that for the problem that the convex hull of high-dimensional vectors cannot be calculated temporarily, we use t-SNE to reduce the collected vectors to 2-dimensional at first (like the work of Zhang et al.), and further continue calculate the convex hull of the sequence, and use the area to represent the semantic size covered by the convex hull (The reproducibility is shown in Appendix B).

Both in the datasets, regardless of the uni-directional or bi-directional RNNs structure is used, the experimental results' trends are similar. Therefore, we only show the results generate by the use of bi-directional RNNs to compare different recurrent units before and after the introduction of attention on the dataset SST. The more experimental results can found in Appendix C, such as uni-directional RNNs in SST dataset, uni/bi-directional RNNs on AGNews.

5.2 Analysis from the Corresponding Perspective

As shown in Figure 4, by observing the difference in semantic density from the corresponding perspective. We can find that the embedding sequences ($\mathcal{E}, \mathcal{E}^{\text{attn}}$) learned by the model is basically the same for different recurrent types with or without attention. However, from the hidden layer sequence, $\mathcal{H}, \mathcal{H}^{attn}$, we can observe the change of this difference, no matter what type of recurrent unit, $\mathcal{H}^{\text{attn}}$ are less than $\mathcal{H}$, this shows that the hidden layer sequence of RNNs using the attention escapes or abstracts the original text with a smaller semantic range, and the semantics expressed are more accurate. After using the attention, the hidden layer sequence improves the accuracy of semantic expression (decrease the degree of semantic unsaturation) also brings about the improvement of the model effect, as shown in Table 2. The attention mechanism's introduction also led to the shortening of the central idea offset between embedding sequences and hidden sequences.

From the semantic coverage perspective, the introduction of attention makes the semantic conversion between the embedding layer and the hidden layer generally improve in all of the three indicators, semantic coverage recall, semantic coverage accuracy, and semantic coverage F-Measure. This improvement shows that the semantic closeness between embedding and hidden layer and the degree of unsaturation of the hidden layer greatly influence the model results. The introduction of attention improve the accuracy of hidden layer sequence expression semantics and makes the semantic conversion between the embedding sequence and the hidden layer sequence more natural. If we compare different types of recurrent units, it is not difficult to find that RNNs is significantly worse than LSTM and GRU on most indicators, which shows that the prediction accuracy of RNNs is lower than LSTM and GRU is truthfully reflected in our indicators.

5.3 Analysis from the Shift Perspective

The result of the shift perspective is shown in Figure 5, the picture on the top reflects $\Delta(\mathrm{CIO}(\mathcal{E}, \mathcal{H}), \mathrm{CIO}(\mathcal{E}^{attn}, \mathcal{A}^{attn}))$, the bottom picture reflects $\Delta(\mathrm{DSU}(\mathcal{H}), \mathrm{DSU}(\mathcal{A}^{attn}))$. After the introduction of attention, the sequence $\mathcal{A}^{attn}$ used for original semantic expression has a smaller degree of semantic unsaturation than $\mathcal{H}$ used for original semantic expression before the introduction of attention. At the same time, the distance between the central idea of the embedding sequence ($\mathrm{CI}(\mathcal{E})$, $\mathrm{CI}(\mathcal{E}^{attn})$) and the final vector used as an instance representation of the embedding sequence ($\mathrm{CI}(\mathcal{H}), \mathrm{CI}(\mathcal{A}^{attn})$) is also shortened. The improvement of these indicators is also reflected in the accuracy of the model.

It is worth mentioning that if we observe the changes from the perspective of different recurrent types, it is not difficult to find that the number of gate structures in the recurrent type is positively correlated with the degree of semantic unsaturation of embedding sequences and attention sequences. (There are three gate structures in LSTM, 2 in GRU and 0 in Vanilla-RNN.)

5.4 Analysis from the Holistic Perspective

Table 1 and Figure A in Appendix show that in terms of dataset size, vocabulary size and sentence length, the AGNews dataset is larger than SST. By observing the experimental results, it can be found that the changes in the many indicators of RNNs after adding attention on SST are the most obvious. On the other hand, the experimental results for LSTM and GRU without using attention are significantly better than RNN in term of semantic expression ability. However, this superiority is largely compromised by the introduction of attention, which can well recognize the central idea and effectively condense the semantics of a sequence. Therefore, for LSTM and GRU, the changes in performance caused by adding attention are relatively less than that for RNN. This also explains why the accuracy of any RNN variant can be greatly improved after the introduction of attention.

Through Figure 4 and Figure 7 in Appendix, we can see that the introduction of attention on the SST dataset has led to substantial improvements for all indicators, and the improvements on the AGNews dataset are significantly lower. Furthermore, for CIO indicators, RNN results are similar to LSTM and GRU after the introduction of attention. The CIO indicator measures the offset between central ideas and is directly related to the vector that the model finally uses to make predictions (see Formula 8 and Formula 15). Therefore, the high performance of RNN with dot-attn on SST validation set is explainable, especially a single-directional RNN model with dot-attn on the SST validation set reached 0.948.

All of these results verify that the analysis framework in our paper can objectively reflect the attention mechanism's effectiveness on the semantic expression ability.

6 Related Work

Guidotti et al. (2018) divided the problem of blackbox models in detail, and the interpretability problem of attention discussed in this paper belongs to the model explanation problem.

RNNs can be said to be the basic ancestor model that introduced the attention mechanism in NLP tasks. Karpathy et al. (2015) established a mapping between the neurons of hidden layers and the content represented to explore the RNNs, Du et al. (2019) proposed a quantitative analysis framework to pave the way for effective quality analysis of

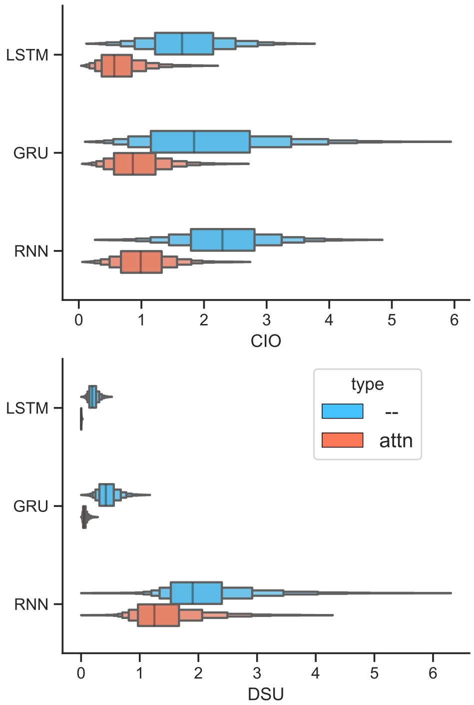

Figure 5: From the shift perspective observe the impact of the introduction of attention on the model. In each bin, top (blue) is the model without attention, bottom (red) is the model with attention.

RNNs, Zhang et al. (2020) assessing the memory ability of RNNs.

Bahdanau et al. (2014b) explained the attention from the perspective of translation alignment, Lee et al. (2017) presented an interactive interface for visualizing and intervening behavior of attention. Recently, a large amount of quantitative analysis work on the interpretability of attention has emerged, such as Vashishth et al. (2019); Jain and Wallace (2019); Serrano and Smith (2019); Wiegreffe and Pinter (2019); Jain et al. (2019), which explained the attention only focused on the attention distribution itself, and used an *erase* method.

7 Conclusions

In this paper, we have proposed a novel framework, based on a convex hull representation of sequence semantics over a Semantic Euclidean Space, to analyze the effect of attention on the semantic capacity of a RNN model and how the effect differs on different network structures. Extensive experiments on two NLP tasks provide in-depth insights on how and why attention impacts the model. From the corresponding perspective, the introduction of attention directly leads to (1) a reduction of semantic unsaturation in the hidden layer of RNNs, that is, an increase in accuracy of the original semantic expression, (2) narrowing the central idea distance between the hidden layer sequence and the embedding layer sequence, (3) an improved performance of semantic coverage between embedding layer sequence and hidden layer sequence. These are critical impacts of attention on the model and improve the capabilities of different types of RNNs. From the shift perspective, the attention layer sequence further reduces the degree of semantic unsaturation, and gets a closer proximity to the embedding layer sequence in the central idea. This is a critical factor in improving the model's accuracy. Our method illustrates how attention affects the model from the perspective of semantic transformation and makes up the limitations of the previous studies in which they only uses a single model to analyze attention.

We believe that the method proposed in this paper will help carry out more in-depth analysis of the role of attention and provide a brand-new perspective for semantic visualization in NLP tasks.

Acknowledgments

This work is funded in part by the National Key Research and Development Program of China (grant No. 2018YFC0831704) and Natural Science Foundation of China (grant No. U1636203, U1736103).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014a. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014b. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. A quantitative analysis framework for recurrent neural network. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1062–1065.

Ivor D Faux and Michael J Pratt. 1979. *Computational geometry for design and manufacture*. Horwood Chichester.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sarthak Jain, Ramin Mohammadi, and Byron C. Wallace. 2019. An analysis of attention over clinical notes for predictive tasks. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 15–21, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. 2017. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126, Copenhagen, Denmark. Association for Computational Linguistics.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Murray H Protter and Charles Bradfield Morrey. 1977. *College calculus with analytic geometry*. Addison-Wesley.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention interpretability across nlp tasks. *arXiv preprint arXiv:1909.11218*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Cheng Zhang, Qiuchi Li, Lingyu Hua, and Dawei Song. 2020. Assessing the memory ability of recurrent neural networks. In *Proceedings of the 24th European Conference on Artificial Intelligence*, pages 1658–1665. IOS Press.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

A Datasets Detail

The statistics of sentence length in the test dataset of SST and AGNews are shown in Figure 6. The distribution of sentence length in the figure shown the sentence length in the AGNews' test set is mainly concentrated in (0, 80), in the SST's test dataset is mainly concentrated in (0, 40).

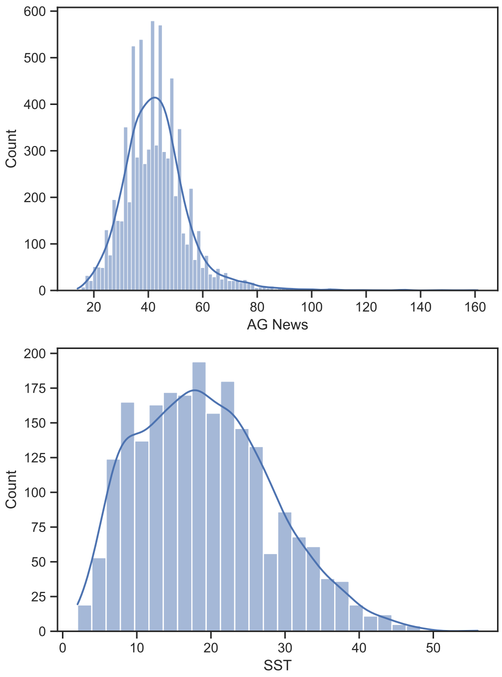

Figure 6: Sentence length statistics. The abscissa indicates the length of the sentence, and the ordinate indicates the count number.

B Reproducibility

Our experiment uses public dataset SST and AGNews. At the same time, in order to reproduce the experimental results more conveniently, we store the scores of each set of sequences in the dataset on defined indicators in a pickle binary file [2], which is convenient for you load it in and use Pandas [3] to view it. We uploaded all the pickle files saved under different models and different datasets to the code part and provided our drawing part of the code to view the experimental results disclosed in our paper, and the model code and training code will be released after some sorting.

C Experiment

Figure 7 from the corresponding perspective observe the impact of the introduction of attention. Each set of pictures shows the experimental results under different experimental settings. Contains the dataset used in the experiment (SST or AGNews) and the directionality of RNNs (uni-directional or bi-directional), different types of recurrent units are compared on the ordinate of each picture.

Figure 8 from the shift perspective observe the impact of the introduction of attention. Each set of pictures shows the experimental results under different experimental settings. Contains the dataset used in the experiment (SST or AGNews) and the directionality of RNNs (uni-directional or bi-directional), different types of recurrent units are compared on the ordinate of each picture.

---

[2] `https://docs.python.org/3/library/pickle.html`

[3] `https://pandas.pydata.org/`

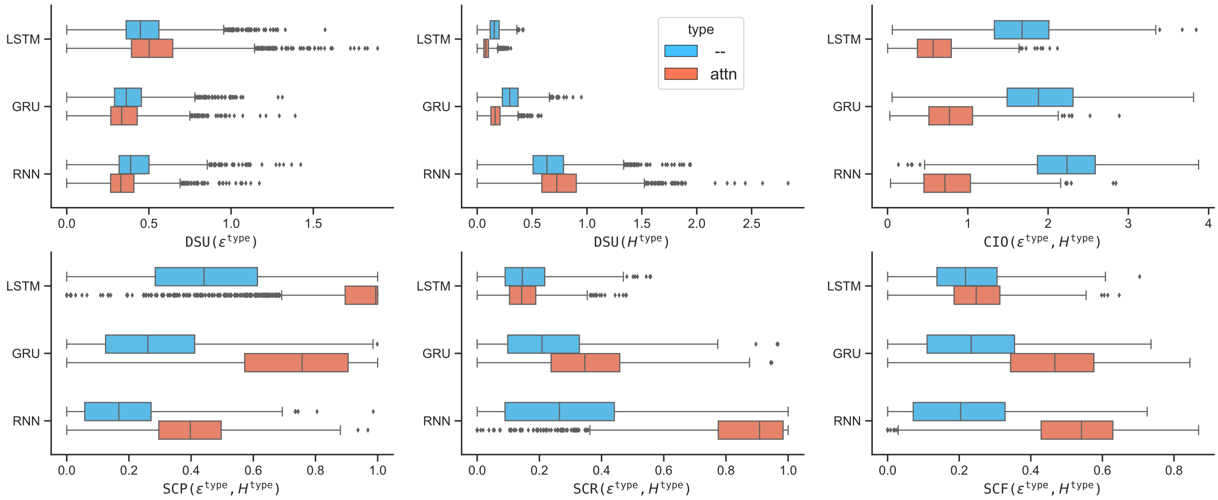

(a) SST, uni-directional

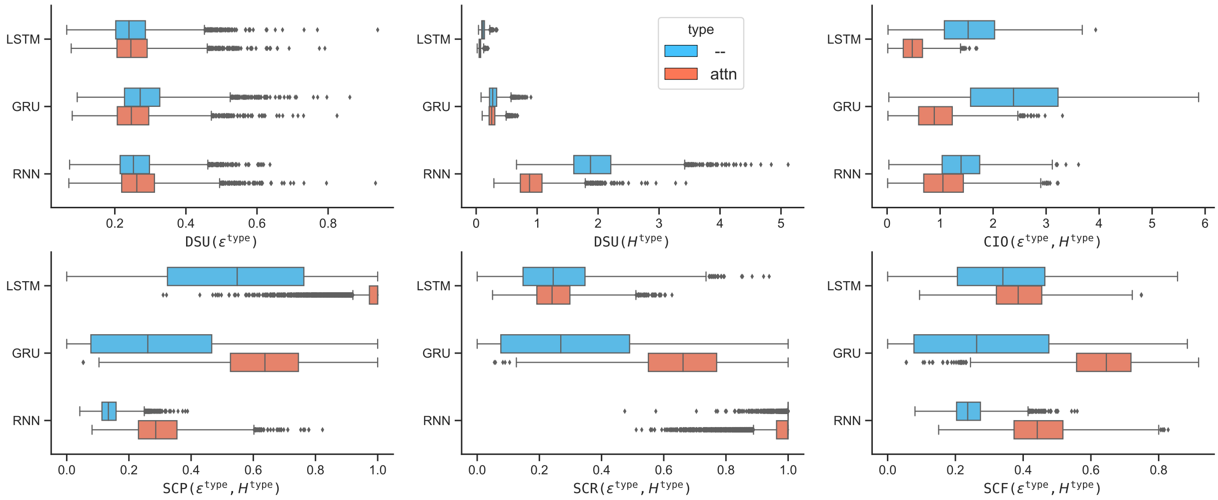

(b) AGNews, bi-directional

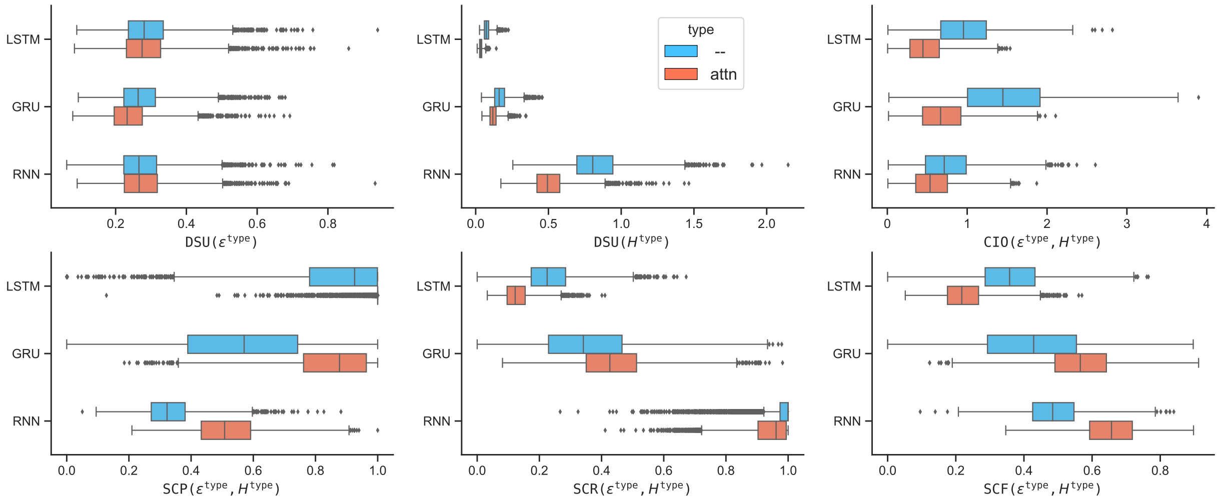

(c) AGNews, uni-directional

Figure 7: From the corresponding perspective observe the impact of the introduction of attention.

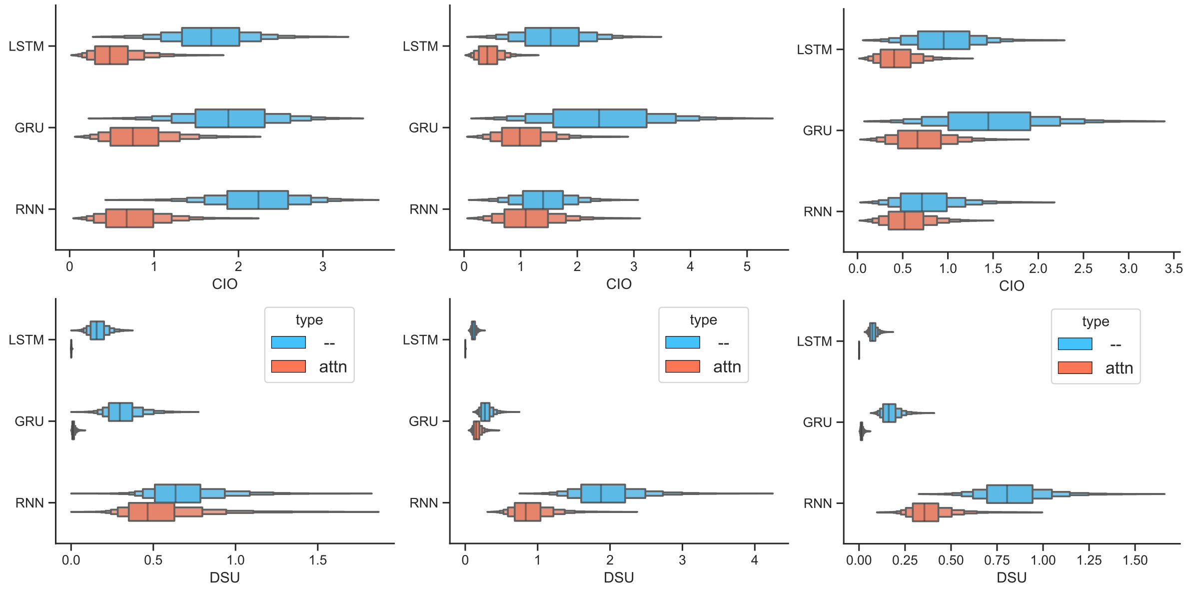

(a) SST, uni-directional (b) AGNews, bi-directional (c) AGNews, uni-directional

Figure 8: From the shift perspective observe the impact of the introduction of attention.

Contrastive Attention for Automatic Chest X-ray Report Generation

Fenglin Liu[1], **Changchang Yin**[3], **Xian Wu**[4], **Shen Ge**[4], **Ping Zhang**[3], **Xu Sun**[2]
[1]School of ECE, Peking University
[2]MOE Key Laboratory of Computational Linguistics, School of EECS, Peking University
[3]The Ohio State University [4]Tencent Medical AI Lab, Beijing, China
`fenglinliu98@pku.edu.cn; {yin.731, zhang.10631}@osu.edu`
`{kevinxwu, shenge}@tencent.com; xusun@pku.edu.cn`

Abstract

Recently, chest X-ray report generation, which aims to automatically generate descriptions of given chest X-ray images, has received growing research interests. The key challenge of chest X-ray report generation is to accurately capture and describe the abnormal regions. In most cases, the normal regions dominate the entire chest X-ray image, and the corresponding descriptions of these normal regions dominate the final report. Due to such data bias, learning-based models may fail to attend to abnormal regions. In this work, to effectively capture and describe abnormal regions, we propose the Contrastive Attention (CA) model. Instead of solely focusing on the current input image, the CA model compares the current input image with normal images to distill the contrastive information. The acquired contrastive information can better represent the visual features of abnormal regions. According to the experiments on the public IU-X-ray and MIMIC-CXR datasets, incorporating our CA into several existing models can boost their performance across most metrics. In addition, according to the analysis, the CA model can help existing models better attend to the abnormal regions and provide more accurate descriptions which are crucial for an interpretable diagnosis. Specifically, we achieve the state-of-the-art results on the two public datasets.

1 Introduction

A medical report is a paragraph containing multiple sentences that describe both the normal and abnormal regions in the chest X-ray image. Chest X-ray images and their corresponding reports are widely used in clinical diagnosis (Delrue et al., 2011). However, writing medical reports requires particular domain knowledge (Goergen et al., 2013), and only experienced radiologists can accurately interpret chest X-ray images and note down corresponding findings in a coherent manner. An automatic chest X-ray report generation system (Jing et al., 2018, 2019; Liu et al., 2021b,a, 2019c) can reduce the workload of radiologists and are in urgent need (Brady et al., 2012; Delrue et al., 2011).

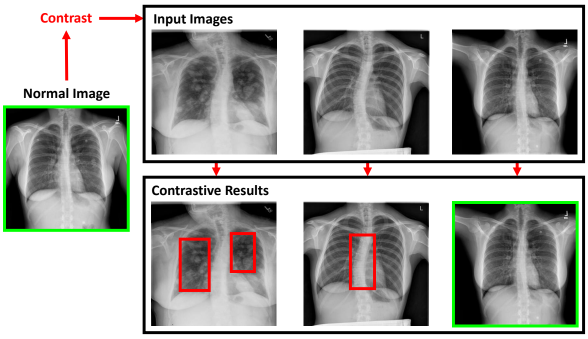

Figure 1: By contrasting current input images and known normal images, it could be easier to capture the suspicious abnormal regions (Red bounding boxes). The images with Green boxes are normal.

In recent years, several deep learning-based methods have been proposed (Jing et al., 2018, 2019; Li et al., 2018, 2019; Chen et al., 2020c; Liu et al., 2021b,a) for automatic chest X-ray report generation; however, there are serious data deviation problems in the medical report corpus. For example, 1) the normal images dominate the dataset over the abnormal ones (Shin et al., 2016); 2) given an input image, the normal regions usually dominate the image and their descriptions dominate the medical report (Jing et al., 2019; Liu et al., 2021b,a). Such data deviation may prevent learning-based methods from capturing the rare but important abnormal regions (e.g., lesion regions). As a result, the learning-based model tends to generate plausible general reports with no prominent abnormal narratives (Jing et al., 2019; Li et al., 2018; Yuan et al., 2019). In clinical practice, accurate detection and depiction of abnormalities are more helpful in disease diagnosing and treat-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 269–280
August 1–6, 2021.

ment planning. Therefore, existing learning-based methods may fail to assist radiologists in clinical decision-making (Goergen et al., 2013).

To capture the abnormal regions from a chest X-ray image, a natural intuition is to compare it with normal images and identify the differences. As Figure 1 shows, given known normal images, it cloud be easier for models to learn and identify the suspicious abnormal regions (Red bounding boxes). Therefore, we propose the Contrastive Attention (CA) model (see Figure 2), which is based on the attention mechanism (Bahdanau et al., 2015a; Vaswani et al., 2017). The CA model can be easily integrated into existing learning-based methods and enable them to better capture and describe the abnormalities. We build the CA model in the following three steps: 1) we first build a set of normal images which are all extracted from the training dataset; 2) we introduce the Aggregate Attention to prioritize normal images that are closer to the current input image, and filter out normal images that appear differently; 3) and we further introduce the Differentiate Attention to distill the common features between the input image and the refined normal images. Then, the acquired common features are subtracted from the visual features of the input image. In this manner, the residual visual features of the input image are treated as the contrastive information that captures the differentiating properties between input image and normal images.

We evaluate our approach on two datasets, including a widely-used benchmark IU-X-ray dataset (Demner-Fushman et al., 2016) and a recently released large-scale MIMIC-CXR dataset (Johnson et al., 2019). On both automatic metrics and human evaluations, existing methods equipped with the proposed Contrastive Attention model outperform baselines, which proves our argument and demonstrates the effectiveness of our approach.

Overall, the main contributions of our work are:

- We propose the Contrastive Attention model to capture and depict the abnormalities by comparing the input image with known normal images. The proposed approach can be easily incorporated into existing models to improve their performance.

- We evaluate our approach on two public datasets. After equipping our Contrastive Attention model, the baselines achieve up to 14% gain and 17% gain in BLEU-4 on the MIMIC-CXR and IU-X-ray datasets, respectively.

- More encouragingly, we achieve the state-of-the-art performance on the two public datasets, i.e., IU-X-ray and MIMIC-CXR. Moreover, we invite professional clinicians to conduct human evaluation to measure the effectiveness in terms of its usefulness for clinical practice.

2 Related Works

In this section, we will describe the related works in three categories: 1) Image Captioning; 2) Chest X-ray Report Generation and 3) Contrastive Learning.

Image Captioning Image captioning aims to understand the given images and generate corresponding descriptive sentences (Chen et al., 2015). The task combines image understanding and language generation. In recent years, a large number of encoder-decoder based neural systems have been proposed for image captioning (Cornia et al., 2020; Pan et al., 2020; Pei et al., 2019; Venugopalan et al., 2015; Vinyals et al., 2015; Xu et al., 2015; Rennie et al., 2017; Lu et al., 2017; Anderson et al., 2018; Liu et al., 2018, 2019b, 2020, 2019a). However, the sentence generated by image captioning is usually short and describes the most prominent visual contents, which cannot fully represent the rich feature information of the image. Recently, visual paragraph generation (Krause et al., 2017), which aims to generate long and coherent reports or stories to describe visual contents, has recently attracted increasing research interests. However, due to the data bias in the medical domain, the widely-used hierarchical LSTM in the visual paragraph generation does not perform very well in automatic chest X-ray report generation and is tend to produce normal reports (Xue et al., 2018; Li et al., 2018; Jing et al., 2019; Yin et al., 2019).

Chest X-ray Report Generation Inspired by the success of deep learning models on image captioning, a lot of encoder-decoder based frameworks have been proposed (Jing et al., 2018, 2019; Liu et al., 2021b,a, 2019c; Yuan et al., 2019; Xue et al., 2018; Li et al., 2018, 2019, 2020; Zhang et al., 2020a; Kurisinkel et al., 2021; Ni et al., 2020; Nishino et al., 2020; Chen et al., 2020c; Wang et al., 2021; Boag et al., 2019; Syeda-Mahmood et al., 2020; Yang et al., 2020; Lovelace and Mortazavi, 2020; Zhang et al., 2020b; Miura et al., 2021). Specifically, Jing et al. (2018) proposed a hierarchical LSTM with the attention mechanism (Bahdanau et al., 2015b; Xu et al., 2015; You et al., 2016). Yuan et al. (2019) further incorporated the medical concept to enrich the decoder with descriptive semantics. Xue et al. (2018) proposed a multimodal recurrent model containing an iterative decoder with visual attention to improve the coherence between sentences. Miura et al. (2021) proposed an Exact Entity Match Reward and an Entailing Entity Match Reward to improve the factual completeness and consistency of the generated reports, resulting in significant improvements on clinical accuracy. Jing et al. (2019); Li et al. (2018); Liu et al. (2019c) and Zhang et al. (2020a); Li et al. (2019); Liu et al. (2021b) introduced the reinforcement learning and medical knowledge graph for chest X-ray report generation, respectively. However, some errors occur in the generated reports of the existing methods, like duplicate reports, inexact descriptions, etc (Xue et al., 2018; Yuan et al., 2019; Yin et al., 2019).

Contrastive Learning The most related work to our contrastive attention mechanism is in the field of contrastive learning (Chen et al., 2020a; He et al., 2020; Hénaff et al., 2020; Grill et al., 2020; Chen et al., 2020b; Radford et al., 2021; Jia et al., 2021), which learns similar/dissimilar image representations from data that are organized into similar/dissimilar image pairs. In image captioning, Dai and Lin (2017) introduced the contrastive learning to extract the contrastive information from additional images into the captioning models to improve the distinctiveness of the generated captions. Moreover, Song et al. (2018) and Duan et al. (2019) proposed the contrastive attention mechanism for person re-identification and summarization, respectively. Song et al. (2018) utilized a pre-provided person and background segmentation to learn features contrastively from the body and background regions, resulting they can be easily discriminated. Duan et al. (2019) contrastively attended to relevant parts and irrelevant parts of source sentence for abstractive sentence summarization. In this work, we leverage the contrastive information between the input image and the normal images to help models efficiently capture and describe the abnormalities for automatic chest X-ray report generation.

3 Approach

In Section 3.1, we formulate the automatic chest X-ray report generation problems; In Section 3.2, we introduce the Contrastive Attention in detail.

3.1 Problem Formulation

Given a chest X-ray image I, the goal of automatic chest X-ray report generation is to generate a coherent report R that addresses findings of I. Most existing methods (Jing et al., 2018) adopt the encoder-decoder frameworks, which normally include an image encoder (He et al., 2016) and a report decoder (Krause et al., 2017). The encoder-decoder framework can be formulated as:

$$\text{Image Encoder}: I \rightarrow V; \quad \text{Report Decoder}: V \rightarrow R. \quad (1)$$

The image encoder, e.g., ResNet (He et al., 2016), aims to generate the visual features $V \in \mathbb{R}^{N_\mathrm{I} \times d}$. The report decoder, e.g., Hierarchical LSTM (Krause et al., 2017), is used to generate the report R from V. Specifically, in the Hierarchical LSTM, a paragraph-level LSTM first generates topic vectors to represent the sentences, then a sentence-level LSTM takes each topic vector as input to generate the corresponding sentence. As a result, the Hierarchical LSTM can better model a paragraph of multiple sentences (report) than a single LSTM (Jing et al., 2018; Krause et al., 2017). Given the ground truth medical report provided by the radiologists for the input chest X-ray image, existing methods train the encoder-decoder frameworks by minimizing training loss, e.g., cross-entropy loss. Due to limited space, please refer to Huang et al. (2019); Jing et al. (2018) for detailed introduction.

In this paper, we adopt the ResNet-50 (He et al., 2016) to extract the visual features, i.e., the output of the last convolutional layer is used as the visual information:

$$V = \text{ResNet}(I)\mathrm{W}^I, \quad (2)$$

where I denotes the input image, $\text{ResNet}(I) \in \mathbb{R}^{49 \times 2048}$ and $\mathrm{W}^I \in \mathbb{R}^{2048 \times d}$ which reduces the dimension from 2,048 to d[1]. Specifically, the d is set to 512, resulting $V = \{v_1, v_2, \ldots, v_{N_\mathrm{I}}\} \in \mathbb{R}^{N_\mathrm{I} \times d}$, where $N_\mathrm{I} = 49$ and $d = 512$. Moreover, we apply the average pooling to obtain the global visual feature:

$$\hat{v} = \text{AveragePooling}(V) = \frac{1}{N_\mathrm{I}} \sum_{i=1}^{N_\mathrm{I}} v_i. \quad (3)$$

After the above calculations, we obtain the visual feature vectors $V = \{v_1, v_2, \ldots, v_{N_\mathrm{I}}\} \in \mathbb{R}^{N_\mathrm{I} \times d}$ and the global visual feature vector $\hat{v} \in \mathbb{R}^{1 \times d}$.

---

[1] For conciseness, all the bias terms of linear transformations in this paper are omitted.

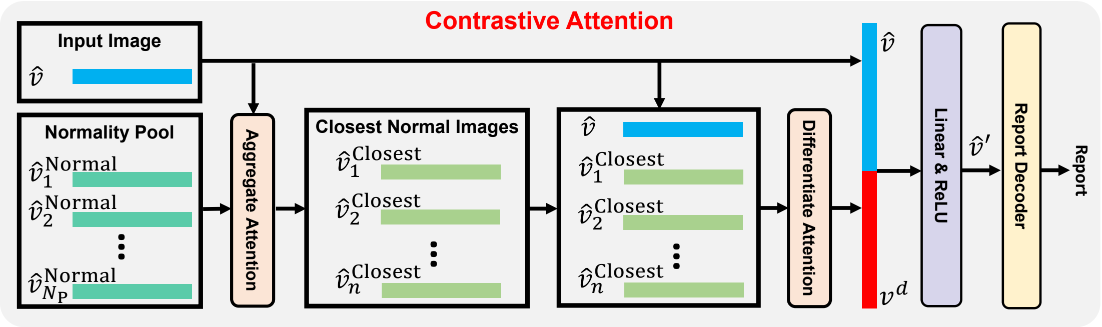

Figure 2: Illustration of our proposed Contrastive Attention, which consists of the Aggregate Attention and Differentiate Attention. In particular, the Aggregate Attention devotes to finding the normal images that are closest to the current input image in the normality pool. The Differentiate Attention devotes to summarizing the common information between the input image and the closest normal images and subtract it from the input image to capture the differentiating properties between the input image and the normal images.

3.2 Contrastive Attention

In this paper, we propose the Contrastive Attention to enable the models to capture the differentiating properties between the input image and normal images. To this end, we first collect a normality pool $P = \{\hat{v}_1^{\text{Normal}}, \hat{v}_2^{\text{Normal}}, \dots, \hat{v}_{N_\text{P}}^{\text{Normal}}\} \in \mathbb{R}^{N_\text{P} \times d}$ which consists of $N_\text{P} = 1,000$ normal images randomly extracted from the training dataset, where $\hat{v}_i^{\text{Normal}} \in \mathbb{R}^{1\times d}$ denotes the global visual feature of i^{th} extracted normal image. Then, as shown in Figure 2, the proposed Contrastive Attention introduces the Aggregate Attention and Differentiate Attention which aims to obtain the contrastive information between the input image $\hat{v}$ and the normality pool P.

Aggregate Attention Since the images in the normality pool P are all normal, there is no ranking order among these images. Therefore it's natural to treat all normal images equally in capturing the contrastive information. However, as shown in Figure 3, we note that there are many noisy images in the normality pool (see the Purple boxes in Figure 3). For example, some normal images have different orientation information or rotation angles from the input image, we cannot direct compare these images with the input image. Therefore, these noisy images will bring noise information, preventing the Contrastive Attention from capturing accurate abnormal regions efficiently. Motivated by the above observations, we introduce Aggregate Attention to increase the weights of normal images that are close to the current input image and lower the weights of images that are not close to the current input image. As a result, the contrasting process could be improved. To implement the Aggregate Attention, we utilize the dot-product attention[2] (Vaswani et al., 2017), which is defined as:

$$\begin{aligned} \text{Att}(x, y) &= \text{softmax}(M)y \\ \text{where } M &= \frac{x\text{W}^x(y\text{W}^y)^T}{\sqrt{d}}, \end{aligned} \tag{4}$$

where $\text{W}^x, \text{W}^y \in \mathbb{R}^{d\times d}$ are learnable parameters. Given $x \in \mathbb{R}^{N_x \times d}$ and $y \in \mathbb{R}^{N_y \times d}$, the acquired M is then of the shape of $N_x \times N_y$, the function $softmax$ is conducted on each row of M, resulting in $\text{Att}(x, y) \in \mathbb{R}^{N_x \times d}$. Then we apply Eq. (4) to $\hat{v} \in \mathbb{R}^{1\times d}$ and $P \in \mathbb{R}^{N_\text{P}\times d}$:

$$\hat{v}^{\text{Closest}} = \text{Att}(\hat{v}, P). \tag{5}$$

Since the attention mechanism is a function that computes the similarity of $\hat{v}$ and each $\hat{v}_i^{\text{Normal}}$ in P, and $\text{Att}(\hat{v}, P) \in \mathbb{R}^{1\times d}$ is the attended vector for the $\hat{v}$. In this way, we can increase the weights of normal images that are similar to current image and lower the weights of the ones that are dissimilar.

Moreover, following Lin et al. (2017), we repeat the Aggregate Attention n times with different learnable attention weights to further promote the performance of our approach, defined as follows:

$$\begin{aligned} P' &= \text{Aggregate-Attention}(\hat{v}, P) \\ &= [\text{Att}_1(\hat{v}, P); \text{Att}_2(\hat{v}, P); \dots; \text{Att}_n(\hat{v}, P)] \\ &= \{\hat{v}_1^{\text{Closest}}, \hat{v}_2^{\text{Closest}}, \dots, \hat{v}_n^{\text{Closest}}\} \in \mathbb{R}^{n\times d}, \end{aligned} \tag{6}$$

where $[\cdot;\cdot]$ stands for concatenation operation. In particular, for chest X-ray images, the result $\text{Att}(\hat{v}, P)$ in Eq. (5) turns out to be the found normal images that are closest to the current entire

---

[2] Our preliminary experiments show that using the dot-product attention (Vaswani et al., 2017) could achieve better performance than the additive attention (Bahdanau et al., 2015a; Xu et al., 2015; Lu et al., 2017) and bilinear attention (Kim et al., 2018; You et al., 2016).

input image. However, $\text{Att}(\hat{v}, P)$ cannot capture those normal images that are closest to the input image only in a specific part rather than the entire image. Fortunately, through repeating the function $\text{Att}(\hat{v}, P)$ in Eq. (5) n times, our Aggregate Attention can efficiently find the closest normal images $P' = \{\hat{v}_1^{\text{Closest}}, \hat{v}_2^{\text{Closest}}, \ldots, \hat{v}_n^{\text{Closest}}\}$ from n parts.

Differentiate Attention To learn the contrastive information between the input image and the closest normal images, we first attempt to capture their similarity, i.e., the common information, then we remove this similarity portion from the input image to obtain the contrastive information. To this end, we introduce the Differentiate Attention.

The first step is learning to summarize the common information $v^c \in \mathbb{R}^{1\times d}$ between the information of current input image $\hat{v} \in \mathbb{R}^{1\times d}$ and the information of closest normal images $P' \in \mathbb{R}^{n\times d}$. In implementation, we employ the same dot-product self-attention mechanism in Eq. (5) and average pooling operation to obtain the $v^c \in \mathbb{R}^{1\times d}$:

$$v^c = \text{AveragePooling}\left(\text{Att}\left([\hat{v}; P'], [\hat{v}; P']\right)\right), \quad (7)$$

where $[\cdot; \cdot]$ denotes row-wise concatenation operation, then the $[\hat{v}; P']$ is in the shape of $(n+1) \times d$ and the $\text{Att}\left([\hat{v}; P'], [\hat{v}; P']\right)$ function outputs a matrix in the shape of $(n+1) \times d$.

In this way, via such self-attention mechanism, we exploit the similarity between P' and $\hat{v}$ to capture the significant common information. Next, to obtain the contrastive information $v^d \in \mathbb{R}^{1\times d}$, we remove (i.e., subtract) the common information $v^c \in \mathbb{R}^{1\times d}$ from the input image $\hat{v} \in \mathbb{R}^{1\times d}$:

$$v^d = \hat{v} - v^c. \quad (8)$$

At last, we update the original image features, i.e., $\hat{v} \in \mathbb{R}^{1\times d}$ and $V = \{v_1, v_2, \ldots, v_{N_\text{I}}\} \in \mathbb{R}^{N_\text{I}\times d}$, with the contrastive information $v^d \in \mathbb{R}^{1\times d}$:

$$\hat{v}' = \text{ReLU}([\hat{v}; v^d]\text{W}') \quad (9)$$

$$v_i' = \text{ReLU}([v_i; v^d]\text{W}'), \quad (10)$$

where $\text{ReLU}(\cdot)$ represents the ReLU activation function and $\text{W}' \in \mathbb{R}^{2d\times d}$ is the matrix for linear transformation. The resulting $\hat{v}'$ and $V' = \{v_1', v_2', \ldots, v_{N_\text{I}}'\}$ are used to replace the original image features V in Eq. (1) and are then fed into existing models to generate coherent reports.

In our subsequent analysis, we show that the contrastive features indeed focus on the abnormal regions and provide a better starting point for downstream models.

4 Experiments

4.1 Datasets, Baselines and Settings

Datasets We use the widely-used benchmark IU-X-ray (Demner-Fushman et al., 2016) dataset and the recently released large-scale MIMIC-CXR (Johnson et al., 2019) datasets for evaluation.

- **IU-X-ray**[3] contains 3,955 radiology reports and 7,470 X-rays images. We use the widely-used splits in Chen et al. (2020c); Jing et al. (2019); Li et al. (2019, 2018); Liu et al. (2021b,a) for evaluation. There are 70%, 10% and 20% instances in training set, validation set and test set, respectively.
- **MIMIC-CXR**[4] is the recently released largest dataset to date and consists of 377,110 chest X-ray images and 227,835 reports from 64,588 patients. Following Chen et al. (2020c); Liu et al. (2021b,a), we use the official splits to report our results. Thus, there are 368,960 in the training set, 2,991 in the validation set and 5,159 in the test set. We convert all tokens of reports to lower-cases and remove the tokens whose frequency of occurrence in the training set is less than 10, resulting in 4k words.

Baselines We experiment with two lines of baselines that are originally designed for image captioning and chest X-ray report generation.

- **Image Captioning Baselines** We experiment on six representative models: NIC (Vinyals et al., 2015), Visual-Attention (Xu et al., 2015), Spatial-Attention (Lu et al., 2017), Att2in (Rennie et al., 2017), Adaptive-Attention (Lu et al., 2017) and Up-Down (Anderson et al., 2018).
- **Chest X-ray Report Generation Baselines** We conduct the experiment on four baselines consisting of HLSTM (Krause et al., 2017; Jing et al., 2018), HLSTM+att+Dual (Harzig et al., 2019), Co-Attention (Jing et al., 2018) and Multi-Attention (Huang et al., 2019).

Settings For our Contrastive Attention model, the model size d is set to 512. Based on the average performance on the validation set, the n in the Aggregate Attention is set to 6. For the normality pool, we randomly extract 1,000 normal images, i.e., $N_\text{P} = 1,000$, for both two datasets from their training set. To re-implement the baselines, following Jing et al. (2019); Li et al. (2019, 2018); Liu et al. (2021b,a), we adopt the ResNet-50 (He et al., 2016) pre-trained on ImageNet (Deng et al.,

---

[3] `https://openi.nlm.nih.gov/`

[4] `https://physionet.org/content/mimic-cxr/2.0.0/`

Settings	Methods	Dataset: MIMIC-CXR (Johnson et al., 2019)						Dataset: IU-X-ray (Demner-Fushman et al., 2016)					
		B-1	B-2	B-3	B-4	M	R-L	B-1	B-2	B-3	B-4	M	R-L
(a)	NIC (Vinyals et al., 2015)†	0.290	0.182	0.119	0.081	0.112	0.249	0.352	0.227	0.154	0.109	0.133	0.313
	w/ Contrastive Attention	**0.317**	**0.200**	**0.127**	**0.089**	**0.120**	**0.262**	**0.368**	**0.232**	**0.166**	**0.118**	**0.144**	**0.323**
(b)	Visual-Attention (Xu et al., 2015)†	**0.318**	0.186	0.122	0.085	0.119	**0.267**	0.371	0.233	0.159	0.118	**0.147**	**0.320**
	w/ Contrastive Attention	0.309	**0.202**	**0.129**	**0.093**	**0.122**	0.265	**0.384**	**0.245**	**0.172**	**0.125**	0.141	0.315
(c)	Spatial-Attention (Lu et al., 2017)†	0.302	0.189	0.122	0.082	0.120	0.259	0.374	0.235	0.158	**0.120**	**0.146**	0.322
	w/ Contrastive Attention	**0.320**	**0.204**	**0.129**	**0.091**	**0.122**	**0.266**	**0.378**	**0.236**	**0.161**	0.116	**0.146**	**0.335**
(d)	Att2in (Rennie et al., 2017)†	0.314	0.199	0.126	0.087	**0.125**	0.265	0.410	0.257	0.173	0.131	0.149	0.325
	w/ Contrastive Attention	**0.327**	**0.205**	**0.132**	**0.095**	0.124	**0.271**	**0.442**	**0.281**	**0.200**	**0.150**	**0.171**	**0.344**
(e)	Adaptive-Attention (Lu et al., 2017)†	0.307	0.192	0.124	0.084	0.119	0.262	**0.433**	**0.285**	0.194	0.137	0.166	**0.349**
	w/ Contrastive Attention	**0.330**	**0.208**	**0.134**	**0.095**	**0.126**	**0.270**	0.425	0.279	**0.198**	**0.142**	**0.167**	0.347
(f)	Up-Down (Anderson et al., 2018)†	0.318	0.203	0.128	0.089	0.123	0.266	**0.389**	**0.251**	**0.170**	0.126	**0.154**	0.317
	w/ Contrastive Attention	**0.336**	**0.209**	**0.134**	**0.097**	**0.128**	**0.273**	0.378	0.246	0.169	**0.129**	0.152	**0.330**
(g)	HLSTM (Krause et al., 2017)†	0.321	0.203	0.129	0.092	0.125	0.270	0.435	0.280	0.187	0.131	0.173	0.346
	w/ Contrastive Attention	**0.352**	**0.216**	**0.145**	**0.105**	**0.139**	**0.276**	**0.453**	**0.290**	**0.203**	**0.153**	**0.178**	**0.361**
(h)	HLSTM+att+Dual (Harzig et al., 2019)†	**0.328**	**0.204**	0.127	0.090	0.122	0.267	0.447	0.289	0.192	0.144	0.175	0.358
	w/ Contrastive Attention	0.323	0.202	**0.130**	**0.102**	**0.138**	**0.277**	**0.464**	**0.292**	**0.205**	**0.149**	**0.176**	**0.364**
(i)	Co-Attention (Jing et al., 2018)†	0.329	0.206	0.133	0.095	0.129	**0.273**	0.463	0.293	0.207	0.155	0.178	0.365
	w/ Contrastive Attention	**0.351**	**0.213**	**0.148**	**0.106**	**0.147**	0.270	**0.486**	**0.311**	**0.223**	**0.178**	**0.187**	**0.372**
(j)	Multi-Attention (Huang et al., 2019)†	0.337	0.211	0.136	0.097	0.130	0.274	0.468	0.299	0.211	0.155	0.180	0.366
	w/ Contrastive Attention	**0.350**	**0.219**	**0.152**	**0.109**	**0.151**	**0.283**	**0.492**	**0.314**	**0.222**	**0.169**	**0.193**	**0.381**

Table 1: Performance of automatic evaluations on the test set of the MIMIC-CXR dataset and the IU-X-ray dataset. † denotes our own implementation. B-n, M and R-L are short for BLEU-n, METEOR and ROUGE-L, respectively. Higher is better in all columns. In this paper, the Red colored numbers denote the best results across all approaches in Table. As we can see, most baseline models enjoy a comfortable improvement with our approach.

Methods	Dataset: MIMIC-CXR (Johnson et al., 2019)						Dataset: IU-X-ray (Demner-Fushman et al., 2016)					
	B-1	B-2	B-3	B-4	M	R-L	B-1	B-2	B-3	B-4	M	R-L
HRGR-Agent (Li et al., 2018)	-	-	-	-	-	-	0.438	0.298	0.208	0.151	-	0.322
CMAS-RL (Jing et al., 2019)	-	-	-	-	-	-	0.464	0.301	0.210	0.154	-	0.362
SentSAT + KG (Zhang et al., 2020a)	-	-	-	-	-	-	0.441	0.291	0.203	0.147	-	0.367
Transformer (Chen et al., 2020c)	0.314	0.192	0.127	0.090	0.125	0.265	0.396	0.254	0.179	0.135	0.164	0.342
R2Gen (Chen et al., 2020c)	0.353	0.218	0.145	0.103	0.142	0.277	0.470	0.304	0.219	0.165	0.187	0.371
Contrastive Attention (Ours)	0.350	0.219	0.152	0.109	0.151	0.283	0.492	0.314	0.222	0.169	0.193	0.381

Table 2: Comparison with existing state-of-the-art methods on the test set of the MIMIC-CXR dataset and the IU-X-ray dataset. As we can see, we achieve the state-of-the-art performance on major metrics on the two datasets.

2009) and fine-tuned on CheXpert dataset (Irvin et al., 2019) to extract the patch visual features with the dimension of each feature is 2,048, which will be projected to 512. Besides, we utilize paired images of a patient as the input for IU-X-ray and utilize single image as the input for MIMIC-CXR to ensure consistency with the experiment settings of previous works (Chen et al., 2020c). For all baselines, since our focus is to provide explicit abnormal region features, which tends to improve existing baselines, we keep the inner structure of the baselines untouched and preserve the original parameter setting and training strategy.

4.2 Main Evaluation

Metrics We first perform the automatic evaluation to conduct a fair comparison. To measure performance, we adopt the widely-used evaluation toolkit (Chen et al., 2015) to calculate the standard metrics: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE-L (Lin, 2004). Specifically, BLEU and METEOR are originally designed for machine translation evaluation, while ROUGE is originally proposed for automatic evaluation of the extracted text summarization.

Results The results on the test set of MIMIC-CXR and IU-X-ray datasets are reported in Table 1. As we can see, our Contrastive Attention can successfully boost baselines with improvement up to 14% and 17% for MIMIC-CXR and IU-X-ray in terms of BLEU-4 score, respectively, where the Setting (g) achieves the greatest improvements. The results prove the effectiveness and generalization capabilities of our approach to a wide range of models.

Moreover, in Table 2, we choose five competitive models including the current state-of-the-art models, i.e., SentSAT + KG (Zhang et al., 2020a) and R2Gen (Chen et al., 2020c), for comparison. For these competitive models, we directly report the results from the original paper. Table 2 shows that when our approach is applied to the Multi-Attention, we outperform these existing state-of-

Metrics	vs. Models	Dataset: MIMIC-CXR (Johnson et al., 2019)			Dataset: IU-X-ray (Demner-Fushman et al., 2016)		
		Baseline wins	Tie	'w/ CA' wins	Baseline wins	Tie	'w/ CA' wins
Fluency	HLSTM (Krause et al., 2017)†	21	45	**34**	18	43	**39**
	Multi-Attention (Huang et al., 2019)†	30	34	**36**	27	41	**32**
Comprehensiveness	HLSTM (Krause et al., 2017)†	13	33	**54**	10	19	**71**
	Multi-Attention (Huang et al., 2019)†	26	35	**39**	22	31	**47**
Faithfulness	HLSTM (Krause et al., 2017)†	20	28	**52**	13	25	**62**
	Multi-Attention (Huang et al., 2019)†	31	32	**37**	24	30	**46**

Table 3: Results of human evaluation on MIMIC-CXR and IU-X-ray datasets for comparing our method with baselines in terms of the **fluency** of generated reports, the **comprehensiveness** of the generated true abnormalities and the **faithfulness** to the ground truth reports. All values are reported in percentage (%).

the-art models on major metrics on the IU-X-ray and MIMIC-CXR datasets, which further proves the effectiveness of our Contrastive Attention.

4.3 Clinical Efficacy

Metrics The metrics used in the Table 1 measure the match between the generated reports and ground truth reports, but are not specialized for the abnormalities in the reports. Therefore, to measure the accuracy of descriptions for clinical abnormalities, we follow Chen et al. (2020c) to adopt the CheXpert labeler (Irvin et al., 2019), which will label the given reports in 14 different categories related to thoracic diseases and support devices, to further report the clinical efficacy metrics. As a result, we can calculate the clinical efficacy scores by comparing the generated reports with ground truth reports in 14 different categories, producing the Precision, Recall and F1 scores.

Results The results are shown in Table 4. As we can see, our approach can boost the performance of baselines under all clinical efficacy metrics. The results prove our arguments and the effectiveness of our approach in boosting the baselines to correctly capture and depict the abnormalities. Specifically, our Multi-Attention w/ CA outperforms the R2Gen (Chen et al., 2020c) with relatively 6%, 9% and 10% margins in terms of Precision, Recall and F1 scores, respectively The superior clinical efficacy scores, which measure the accuracy of descriptions for clinical abnormalities, demonstrate that our approach can help existing models produce higher quality descriptions for clinical abnormalities.

Methods	Clinical Efficacy Metrics		
	Precision	Recall	F1
NIC (Vinyals et al., 2015)	0.249	0.203	0.204
AdaAtt (Lu et al., 2017)	0.268	0.186	0.181
Att2in (Rennie et al., 2017)	0.322	0.239	0.249
Up-Down (Anderson et al., 2018)	0.320	0.231	0.238
Transformer (Chen et al., 2020c)	0.331	0.224	0.228
R2Gen (Chen et al., 2020c)	0.333	0.273	0.276
HLSTM (Krause et al., 2017)†	0.307	0.225	0.228
w/ Contrastive Attention	**0.340**	**0.269**	**0.274**
Multi-Attention (Huang et al., 2019)†	0.336	0.257	0.265
w/ Contrastive Attention	**0.352**	**0.298**	**0.303**

Table 4: Performance of automatic evaluations in terms of clinical efficacy metrics, which measure the accuracy of descriptions for clinical abnormalities, on the MIMIC-CXR dataset. Higher is better in all columns.

4.4 Human Evaluation

Metrics For medical-related task, it is important to know (1) on what fraction of images with abnormalities did the system not mention the abnormality and (2) on what fraction of images the system described abnormality that does not exist according to doctors. To this end, we randomly select 200 samples from the IU-X-ray and MIMIC-CXR test sets, which are 100 samples from each dataset. Specifically, it is important to generate accurate reports (*faithfulness*) with comprehensive true abnormalities (*comprehensiveness*) and it is unacceptable to generate repeated sentences (*fluency*). Therefore, we invite several professional clinicians to compare our approach and baselines independently and evaluate the perceptual quality, including the *fluency* of generated reports, the *comprehensiveness* of generated true abnormalities and the *faithfulness* to the ground truth reports. The clinicians are unaware of which model generates these reports.

Results To conduct the human evaluation, we select a representative chest X-ray report generation baseline: HLSTM (Krause et al., 2017) and a competitive chest X-ray report generation baseline: Multi-Attention (Huang et al., 2019). The results in Table 3 show that our approach enjoys the obvious advantage in terms of the three aspects, i.e., *fluency*, *comprehensiveness* and *faithfulness*, meaning that the reports generated by the "Baseline w/ Contrastive Attention" are of higher clinical quality, which also proves the advantage of our approach in

Settings	Methods	Attention Function	n	Dataset: IU-X-ray (Demner-Fushman et al., 2016)											
				Baseline: HLSTM (Krause et al., 2017)						Baseline: Multi-Attention (Huang et al., 2019)					
				B-1	B-2	B-3	B-4	M	R-L	B-1	B-2	B-3	B-4	M	R-L
(a)	Baseline	-	-	0.435	0.280	0.187	0.131	0.173	0.346	0.468	0.299	0.211	0.155	0.180	0.366
(b)	w/ DA	dot-product attention	-	0.447	0.284	0.195	0.138	0.176	0.357	0.479	0.306	0.215	0.162	0.184	0.373
(c)	w/ DA + AA		1	0.451	0.287	0.198	0.144	0.176	0.359	0.486	0.310	0.218	0.164	0.187	0.375
(f)	w/ DA + AA	dot-product attention	2	0.451	0.285	0.199	0.146	0.175	0.359	0.488	0.312	0.219	0.165	0.189	0.377
(g)	w/ DA + AA		4	0.453	0.287	0.202	0.149	0.177	0.361	0.493	0.314	0.220	0.165	0.191	0.378
(h)	w/ DA + AA		6	0.453	0.290	0.203	0.153	0.178	0.361	0.492	0.314	0.222	0.169	0.193	0.381
(i)	w/ DA + AA		8	0.451	0.288	0.200	0.150	0.180	0.362	0.488	0.311	0.221	0.164	0.194	0.379
(j)	w/ DA + AA		10	0.448	0.284	0.197	0.142	0.175	0.355	0.490	0.310	0.219	0.162	0.188	0.375

Table 5: Quantitative analysis of our Contrastive Attention, which includes the Differentiate Attention (DA) and Aggregate Attention (AA). We conduct the analysis on a widely-used baseline model HLSTM (Krause et al., 2017) and a competitive baseline model Multi-Attention (Huang et al., 2019).

clinical practice. Especially, by using our proposed Contrastive Attention, the winning chances of models increased by maximum of $54 - 13 = 41$ points and $71 - 10 = 61$ points in terms of the *comprehensiveness* metric on the MIMIC-CXR and IU-X-ray datasets, respectively. It demonstrates the effectiveness of our approach in helping existing baselines generate more accurate abnormality descriptions, and thus improve the usefulness of models in better assisting radiologists in clinical decision-makings and reducing their workload.

Overall From the results of automatic and human evaluations, we can see that our proposed Contrastive Attention can provide a solid basis for describing chest X-ray images, especially for the abnormalities. As a result, our approach can successfully boost baselines and achieves new state-of-the-art results on the MIMIC-CXR and IU-X-ray datasets, which verifies the effectiveness of the proposed approach and indicates that our approach is less prone to the variations of model structures, hyper-parameters (e.g., learning rate and batch size), and learning paradigms.

5 Analysis

We conduct analysis on the benchmark IU-X-ray dataset to better understand our proposed approach.

5.1 Quantitative Analysis

We conduct the quantitative analysis on two representative models, i.e., HLSTM and Multi-Attention, to evaluate the contribution of each component.

Effect of Contrastive Attention Our Contrastive Attention consists of the Differentiate Attention (DA) and Aggregate Attention (AA). As shown in Table 5(b), the DA can promote the performance over all metrics. Especially for the HLSTM (Krause et al., 2017), which does not incorporate the attention mechanism to allow more efficient use of the image features. We can see that an up to 5% gain in BLEU-4 score makes the "HLSTM w/ DA" an equally competitive model as the "HLSTM+att+Dual" model in Table 1(h). This indicates that the contrastive information extracted by the DA contains sufficient accurate abnormal information, which is vital in improving the performance of chest X-ray report generation. In other words, our approach can ease the design of the neural models for the task.

For the AA, it devotes to identifying the closest normal images and filtering out the noisy images, which can improve the contrasting process in the DA. As expected, Table 5(c) shows that the AA can consistently boost the performance of baselines under all metrics, which further demonstrates the effectiveness of our approach.

Effect of n In this section, we analyze the effect of n in our Aggregate Attention (see Eq. (6)). Table 5(f-j) shows that when n is smaller than 6, the performance will increase with as n increases. The reason may be that repeating the Aggregate Attention n times with different learnable attention weights can encourage the model to identify the closest normal images from n aspects, e.g., organs or tissues, i.e., if an aspect of the normal image is similar to the input image, it will be identified as the closest normal image. In this way, the Aggregate Attention can capture accurate and robust closest normal images from n aspects. To verify this, we randomly visualize two attention weights of Aggregate Attention in Figure 3. As we can see, the Aggregate Attention can indeed identify the closest normal images from multiple aspects, e.g., 'Bone/Clavicle' (Blue boxes) and 'Right Lung' (Green boxes), which proves our arguments. Moreover, larger n will bring noise, i.e., normal images that are not similar to the current input image, to the model and thus impair the performance.

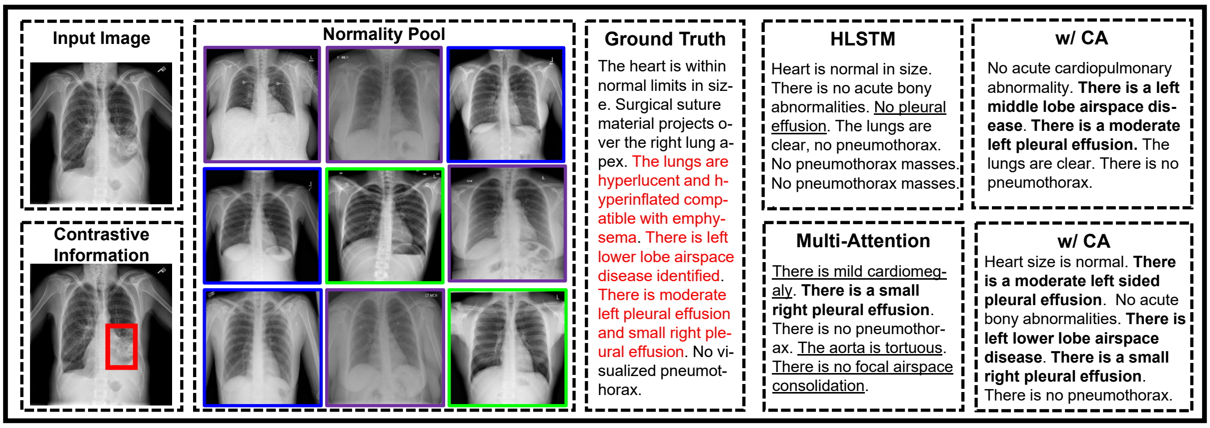

Ground Truth

The heart is within normal limits in size. Surgical suture material projects over the right lung apex. The lungs are hyperlucent and hyperinflated compatible with emphysema. There is left lower lobe airspace disease identified. There is moderate left pleural effusion and small right pleural effusion. No visualized pneumothorax.

HLSTM

Heart is normal in size. There is no acute bony abnormalities. No pleural effusion. The lungs are clear, no pneumothorax. No pneumothorax masses. No pneumothorax masses.

Multi-Attention

There is mild cardiomegaly. **There is a small right pleural effusion**. There is no pneumothorax. The aorta is tortuous. There is no focal airspace consolidation.

w/ CA

No acute cardiopulmonary abnormality. **There is a left middle lobe airspace disease**. **There is a moderate left pleural effusion.** The lungs are clear. There is no pneumothorax.

w/ CA

Heart size is normal. **There is a moderate left sided pleural effusion**. No acute bony abnormalities. **There is left lower lobe airspace disease**. **There is a small right pleural effusion**. There is no pneumothorax.

Figure 3: Examples of the generated reports and the visualization of the Contrastive Attention (CA). Please view in color. CA model can capture the abnormal region (Red bounding box) by contrasting the input image with normal images. Besides, our Aggregate Attention in the CA model can find the closest normal images (Blue and Green boxes visualized from two different attention weights (see Eq. (6))) and filter out the noisy images (Purple boxes). The Red colored text denotes the abnormal descriptions in the ground truth report. Underlined text denotes the generated wrong sentences. **Bold** text denotes the generated true abnormalities. As we can see, the abnormal region and the abnormal descriptions generated by our method show significant alignment with ground truth reports.

5.2 Qualitative Analysis

In this section, we show the reports generated by the Baseline models, i.e., HLSTM and Multi-Attention, and the "Baseline w/ CA" models, and the visualization in Figure 3 to analyze the strength of our Contrastive Attention model intuitively. As we can see, for the HLSTM, it tends to produce repeated findings and normal findings, which results from the overwhelming normal findings in the dataset, i.e., data deviation (Shin et al., 2016). For the Multi-Attention, with the help of attention mechanism, it can describe abnormalities, but some abnormalities are incorrect (Underlined text). The reason is that it is difficult for the Multi-Attention model to efficiently learn the medical expertise from the dataset with data deviation to correctly detect the abnormal regions. Since our Contrastive Attention model can efficiently capture the suspicious abnormal regions by contrasting the input images and normal images and transfer such power to the downstream models and datasets, we can thus help multiple baseline models to detect and describe comprehensive and accurate abnormalities. As a result, the "Baseline w/ CA" models can generate fluent and accurate reports supported with accurate abnormal descriptions, showing significant alignment with ground truth reports.

6 Conclusion

In this paper, we propose the Contrastive Attention model to capture abnormal regions by contrasting the input image and normal images for chest X-ray report generation. The experiments on two public datasets demonstrate the effectiveness of our approach, which can be easily incorporated into existing models to boost their performance under most metrics. The clinical efficacy scores and human evaluation further prove our arguments and the effectiveness of our approach in helping existing models capture and depict the abnormalities. Specifically, we achieve the state-of-the-art results on the two datasets with the best human preference, which could better assist radiologists in clinical decision-making and reduce their workload.

In the future, there are two potential ways to improve the contrastive attention. First, it may be better to perform the contrastive attention in path feature rather than global feature. Second, it may be better to utilize multiple feature maps from different convolutional layers rather than just the feature maps of last convolutional layer.

Acknowledgments

This work is partly supported by Beijing Academy of Artificial Intelligence (BAAI). We would like to sincerely thank the clinicians (Xiaoxia Xie, Jing Zhang, Minghui Shao, etc.) of the Harbin Chest Hospital in China for providing the human evaluation. We also sincerely thank all the anonymous reviewers and chairs for their constructive comments and suggestions. Xu Sun and Ping Zhang are the corresponding authors of this paper.

Ethical Considerations

In this work, we focus on helping several existing chest X-ray report generation systems better capture and describe the abnormalities. To this end, we provide a detailed human evaluation in Table 3 (Section 4.4) and an automatic evaluation in terms of clinical efficacy metrics in Table 4 (Section 4.3) to know (1) on what fraction of images with abnormalities did the system not mention the abnormality and (2) on what fraction of images the system described abnormality that does not exist according to doctors. The results show that our work can help existing systems generate more accurate descriptions for clinical abnormalities, improving the usefulness of existing systems in better **assisting** radiologists in clinical decision-makings and reducing their workload. In particular, for radiologists, given a large amount of medical images, the systems can automatically generate medical reports, the radiologists only need to make revisions rather than write a new report from scratch. This study uses the public MIMIC-CXR and IU-X-ray datasets. All protected health information was de-identified. De-identification was performed in compliance with Health Insurance Portability and Accountability Act (HIPAA) standards in order to facilitate public access to the datasets. Deletion of protected health information (PHI) from structured data sources (e.g., database fields that provide patient name or date of birth) was straightforward. All necessary patient/participant consent has been obtained and the appropriate institutional forms have been archived.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and VQA. In *CVPR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015a. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015b. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*.

William Boag, Tzu-Ming Harry Hsu, Matthew B. A. McDermott, Gabriela Berner, Emily Alsentzer, and Peter Szolovits. 2019. Baselines for chest x-ray report generation. In *ML4H@NeurIPS*.

A. Brady, R. Ó. Laoide, Peter Mccarthy, and R. Mcdermott. 2012. Discrepancy and error in radiology: Concepts, causes and consequences. *The Ulster Medical Journal*, 81:3 – 9.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020a. A simple framework for contrastive learning of visual representations. In *ICML*.

Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. 2020b. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Zhihong Chen, Yan Song, Tsung-Hui Chang, and Xiang Wan. 2020c. Generating radiology reports via memory-driven transformer. In *EMNLP*.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *CVPR*.

Bo Dai and Dahua Lin. 2017. Contrastive learning for image captioning. In *NIPS*.

Louke Delrue, Robert Gosselin, Bart Ilsen, An Van Landeghem, Johan de Mey, and Philippe Duyck. 2011. Difficulties in the interpretation of chest radiography. *Comparative Interpretation of CT and Standard Radiography of the Chest*, pages 27–49.

Dina Demner-Fushman, Marc D. Kohli, Marc B. Rosenman, Sonya E. Shooshan, Laritza Rodriguez, Sameer K. Antani, George R. Thoma, and Clement J. McDonald. 2016. Preparing a collection of radiology examinations for distribution and retrieval. *J. Am. Medical Informatics Assoc.*, 23(2):304–310.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Xiangyu Duan, Hongfei Yu, Mingming Yin, Min Zhang, Weihua Luo, and Yue Zhang. 2019. Contrastive attention mechanism for abstractive sentence summarization. In *EMNLP/IJCNLP*.

Stacy K Goergen, Felicity J Pool, Tari J Turner, Jane E Grimm, Mark N Appleyard, Carmel Crock, Michael C Fahey, Michael F Fay, Nicholas J Ferris, Susan M Liew, et al. 2013. Evidence-based guideline for the written radiology report: Methods, recommendations and implementation challenges. *Journal of medical imaging and radiation oncology*, 57(1):1–7.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

Philipp Harzig, Yan-Ying Chen, Francine Chen, and Rainer Lienhart. 2019. Addressing data bias problems for chest x-ray image report generation. In *BMVC*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. 2020. Data-efficient image recognition with contrastive predictive coding. In *ICML*.

Xin Huang, Fengqi Yan, Wei Xu, and Maozhen Li. 2019. Multi-attention and incorporating background information model for chest x-ray image report generation. *IEEE Access*.

Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn L. Ball, Katie S. Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. 2019. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *AAAI*.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*.

Baoyu Jing, Zeya Wang, and Eric P. Xing. 2019. Show, describe and conclude: On exploiting the structure information of chest x-ray reports. In *ACL*.

Baoyu Jing, Pengtao Xie, and Eric P. Xing. 2018. On the automatic generation of medical imaging reports. In *ACL*.

Alistair E. W. Johnson, Tom J. Pollard, Seth J. Berkowitz, Nathaniel R. Greenbaum, Matthew P. Lungren, Chih-ying Deng, Roger G. Mark, and Steven Horng. 2019. MIMIC-CXR: A large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *NeurIPS*.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *CVPR*.

Litton J Kurisinkel, Ai Ti Aw, and Nancy F. Chen. 2021. Coherent and concise radiology report generation via context specific image representations and orthogonal sentence states. In *NAACL-HLT*.

Christy Y. Li, Xiaodan Liang, Zhiting Hu, and Eric P. Xing. 2019. Knowledge-driven encode, retrieve, paraphrase for medical image report generation. In *AAAI*.

Mingjie Li, Fuyu Wang, Xiaojun Chang, and Xiaodan Liang. 2020. Auxiliary signal-guided knowledge encoder-decoder for medical report generation. *arXiv preprint arXiv:2006.03744*.

Yuan Li, Xiaodan Liang, Zhiting Hu, and Eric P. Xing. 2018. Hybrid retrieval-generation reinforced agent for medical image report generation. In *NeurIPS*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *ACL*.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Fenglin Liu, Shen Ge, and Xian Wu. 2021a. Competence-based multimodal curriculum learning for medical report generation. In *ACL*.

Fenglin Liu, Yuanxin Liu, Xuancheng Ren, Xiaodong He, and Xu Sun. 2019a. Aligning visual regions and textual concepts for semantic-grounded image representations. In *NeurIPS*.

Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Kai Lei, and Xu Sun. 2019b. Exploring and distilling cross-modal information for image captioning. In *IJCAI*.

Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Houfeng Wang, and Xu Sun. 2018. simnet: Stepwise image-topic merging network for generating detailed and comprehensive image captions. In *EMNLP*.

Fenglin Liu, Xuancheng Ren, Xian Wu, Shen Ge, Wei Fan, Yuexian Zou, and Xu Sun. 2020. Prophet attention: Predicting attention with future attention. In *NeurIPS*.

Fenglin Liu, Xian Wu, Shen Ge, Wei Fan, and Yuexian Zou. 2021b. Exploring and distilling posterior and prior knowledge for radiology report generation. In *CVPR*.

Guanxiong Liu, Tzu-Ming Harry Hsu, Matthew B. A. McDermott, Willie Boag, Wei-Hung Weng, Peter Szolovits, and Marzyeh Ghassemi. 2019c. Clinically accurate chest x-ray report generation. In *MLHC*.

Justin R. Lovelace and Bobak Mortazavi. 2020. Learning to generate clinically coherent chest x-ray reports. In *EMNLP (Findings)*.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*.

Yasuhide Miura, Yuhao Zhang, Emily Bao Tsai, Curtis P. Langlotz, and Dan Jurafsky. 2021. Improving factual completeness and consistency of image-to-text radiology report generation. In *NAACL-HLT*.

Jianmo Ni, Chun-Nan Hsu, Amilcare Gentili, and Julian J. McAuley. 2020. Learning visual-semantic embeddings for reporting abnormal findings on chest x-rays. In *EMNLP (Findings)*.

Toru Nishino, Ryota Ozaki, Yohei Momoki, Tomoki Taniguchi, Ryuji Kano, Norihisa Nakano, Yuki Tagawa, Motoki Taniguchi, Tomoko Ohkuma, and Keigo Nakamura. 2020. Reinforcement learning with imbalanced dataset for data-to-text medical report generation. In *EMNLP (Findings)*.

Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. 2020. X-linear attention networks for image captioning. In *CVPR*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for automatic evaluation of machine translation. In *ACL*.

Wenjie Pei, Jiyuan Zhang, Xiangrong Wang, Lei Ke, Xiaoyong Shen, and Yu-Wing Tai. 2019. Memory-attended recurrent network for video captioning. In *CVPR*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*.

Hoo-Chang Shin, Kirk Roberts, Le Lu, Dina Demner-Fushman, Jianhua Yao, and Ronald M. Summers. 2016. Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation. In *CVPR*.

Chunfeng Song, Yan Huang, Wanli Ouyang, and Liang Wang. 2018. Mask-guided contrastive attention model for person re-identification. In *CVPR*.

Tanveer F. Syeda-Mahmood, Ken C. L. Wong, Yaniv Gur, Joy T. Wu, Ashutosh Jadhav, Satyananda Kashyap, Alexandros Karargyris, Anup Pillai, Arjun Sharma, Ali Bin Syed, Orest B. Boyko, and Mehdi Moradi. 2020. Chest x-ray report generation through fine-grained label learning. In *MICCAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *HLT-NAACL*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Fuyu Wang, Xiaodan Liang, Lin Xu, and Liang Lin. 2021. Unifying relational sentence generation and retrieval for medical image report composition. *IEEE Transactions on Cybernetics*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Yuan Xue, Tao Xu, L. Rodney Long, Zhiyun Xue, Sameer K. Antani, George R. Thoma, and Xiaolei Huang. 2018. Multimodal recurrent model with attention for automated radiology report generation. In *MICCAI*.

Shaokang Yang, Jianwei Niu, Jiyan Wu, and Xuefeng Liu. 2020. Automatic medical image report generation with multi-view and multi-modal attention mechanism. In *ICA3PP*.

Changchang Yin, Buyue Qian, Jishang Wei, Xiaoyu Li, Xianli Zhang, Yang Li, and Qinghua Zheng. 2019. Automatic generation of medical imaging diagnostic report with hierarchical recurrent neural network. In *ICDM*.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *CVPR*.

Jianbo Yuan, Haofu Liao, Rui Luo, and Jiebo Luo. 2019. Automatic radiology report generation based on multi-view image fusion and medical concept enrichment. In *MICCAI*.

Yixiao Zhang, Xiaosong Wang, Ziyue Xu, Qihang Yu, Alan L. Yuille, and Daguang Xu. 2020a. When radiology report generation meets knowledge graph. In *AAAI*.

Yuhao Zhang, Derek Merck, Emily Bao Tsai, Christopher D. Manning, and Curtis Langlotz. 2020b. Optimizing the factual correctness of a summary: A study of summarizing radiology reports. In *ACL*.

O2NA: An Object-Oriented Non-Autoregressive Approach for Controllable Video Captioning

Fenglin Liu[1]*, Xuancheng Ren[2]*, Xian Wu[4], Bang Yang[1], Shen Ge[4], Xu Sun[2,3]
[1]ADSPLAB, School of ECE, Peking University
[2]MOE Key Laboratory of Computational Linguistics, School of EECS, Peking University
[3]Center for Data Science, Peking University
[4]Tencent, Beijing, China
`{fenglinliu98, renxc, yb.ece, xusun}@pku.edu.cn`
`{kevinxwu, shenge}@tencent.com`

Abstract

Video captioning combines video understanding and language generation. Different from image captioning that describes a static image with details of almost *every* object, video captioning usually considers a sequence of frames and biases towards *focused* objects, e.g., the objects that stay in focus regardless of the changing background. Therefore, detecting and properly accommodating focused objects is critical in video captioning. To enforce the description of focused objects and achieve controllable video captioning, we propose an Object-Oriented Non-Autoregressive approach (O2NA), which performs caption generation in three steps: 1) identify the focused objects and predict their locations in the target caption; 2) generate the related attribute words and relation words of these focused objects to form a draft caption; and 3) combine video information to refine the draft caption to a fluent final caption. Since the focused objects are generated and located ahead of other words, it is difficult to apply the word-by-word autoregressive generation process; instead, we adopt a non-autoregressive approach. The experiments on two benchmark datasets, i.e., MSR-VTT and MSVD, demonstrate the effectiveness of O2NA, which achieves results competitive with the state-of-the-arts but with both higher diversity and higher inference speed.

1 Introduction

The task of video captioning, which aims to generate a descriptive sentence based on the input video, has a wide range of applications. In recent years, deep neural models, particularly the models based on the encoder-decoder framework (Venugopalan et al., 2015; Pan et al., 2016b; Xu et al., 2017; Aafaq et al., 2019), have achieved great success in advancing the state-of-the-art (Pan et al., 2020; Zheng et al., 2020; Perez-Martin et al., 2021; Yang et al., 2021). These models usually entail the autoregressive property, i.e., conditioning each word on the previously generated words.

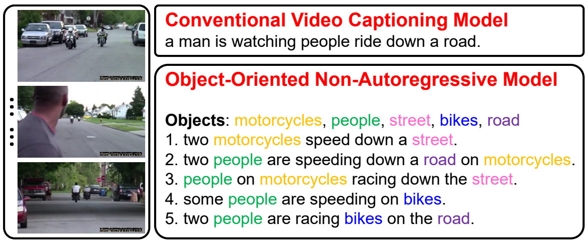
Conventional Video Captioning Model
a man is watching people ride down a road.

Object-Oriented Non-Autoregressive Model

Objects: motorcycles, people, street, bikes, road
1. two motorcycles speed down a street.
2. two people are speeding down a road on motorcycles.
3. people on motorcycles racing down the street.
4. some people are speeding on bikes.
5. two people are racing bikes on the road.

Figure 1: Examples of the captions generated by a state-of-the-art conventional video captioning model (Zheng et al., 2020) and our model. Compared to the conventional model, whose generation process is hardly controllable, our model can be guided to mention the desired objects (i.e., the colored objects) and generate diverse, object-oriented captions for a video.

In video captioning, one critical step is to detect and include focused objects. As exemplified in Figure 1, when a dangerous situation occurs, a captioning-based blind-aid system should focus on the dangerous objects on the road to alert the visually-impaired people, rather than over-describe the presence of pedestrians or shops nearby. It means that in the above example, *speeding vehicles* should be considered as focused objects and should be mentioned in the generated caption. While people could identify focused objects in video easily (Shinn-Cunningham, 2008; Corbetta and Shulman, 2002; Posner and Petersen, 1990), existing captioning systems can hardly be controlled to generate focused objects because of their word-to-word generation practice. Motivated by those observations, we introduce the problem of controllable video captioning in the sense of controlling contents.

As shown in Figure 2, to solve the controllable video captioning problem, we propose the Object-

---

* Equal Contributions.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 281–292
August 1–6, 2021.

Oriented Non-Autoregressive approach (O2NA). Different from conventional models that adopt a left-to-right or word-by-word decoding process, O2NA applies a non-autoregressive manner to control the caption generation. O2NA first detects all objects that appear in the video and then selects the focused objects for the final caption. For example, in the aforementioned blind-aid system, the system would select the dangerous objects *speeding vehicles* in case of an emergency. Next, the caption generation process consists of three main steps: 1) locate all focused objects in the proper locations of the target caption; 2) generate the related attribute words and relation words to form a draft caption; and 3) adopt the iterative refinement approach (Ghazvininejad et al., 2019; Lee et al., 2018) to proofread and improve the draft caption.

For each step, as there is no dependency among generated words, the words can be generated in parallel, indicating a fixed computing time regardless of caption length, while computing time of the conventional autoregressive approach is linear with the caption length. For long captions, conventional methods embody high inference latency, which limits their adoption in real-time applications, e.g., blind-aid system (Voykinska et al., 2016) and human-robot interaction (Das et al., 2017). According to our experiments and analyses on two benchmark datasets, i.e., MSR-VTT (Xu et al., 2016) and MSVD (a.k.a. Youtube2Text) (Guadarrama et al., 2013), our O2NA is able to produce a descriptive and fluent caption which outperforms several existing methods in terms of both accuracy and efficiency.

Overall, the main contributions of this paper are:

- We introduce the problem of controllable video captioning in the sense of controlled contents, which has more practical values than the existing studies on syntactic variations.
- Specifically, we propose the Object-Oriented Non-Autoregressive approach (O2NA) to tackle the controllable video captioning problem by injecting strong control signals conditioned on selected objects, with the benefits of fast and fixed inference time, which are critical for real-time applications.
- We evaluate our approach on two datasets. In particular, our O2NA achieves competitive results with the state-of-the-art methods with higher diversity and higher inference speed.

The rest of this paper is organized as follows: Section 2 reviews the related work; Section 3 introduces the proposed Object-Oriented Non-Autoregressive approach (O2NA) in detail; Section 4 and Section 5 present the experimental results and analyses, respectively; and finally, Section 6 concludes the paper.

2 Related Work

In this section, we describe the related work from 1) Video Captioning, 2) Controllable Image Captioning and 3) Non-Autoregressive Decoding.

2.1 Video Captioning

Recently, a large number of encoder-decoder based neural models have been proposed for video captioning (Venugopalan et al., 2015; Yao et al., 2015; Pan et al., 2016b,a; Xu et al., 2017; Aafaq et al., 2019; Yang et al., 2019; Aafaq et al., 2020; Zheng et al., 2020; Yang et al., 2021; Perez-Martin et al., 2021). These methods mainly introduce a convolutional neural network (CNN) (Krizhevsky et al., 2012) to encode the video and employ a LSTM (Hochreiter and Schmidhuber, 1997) or a Transformer (Zhou et al., 2018) to generate the coherent captions with the attention mechanism (Bahdanau et al., 2015; Pan et al., 2016b). However, these methods lack controllability, i.e., their behaviors can hardly be influenced. Our model allows an easy way to control the contents of video captions rather than merely syntactic variations in existing studies.

2.2 Controllable Image Captioning

Different from image captioning (Xu et al., 2015; Vinyals et al., 2015; Lu et al., 2017; Anderson et al., 2018; Liu et al., 2018, 2019a,b, 2020) that processes a static image with details of almost every appeared object, video captioning considers a sequence of frames which biases towards focused objects. It is still worth noting that the controllable image captioning has been explored most recently (Cornia et al., 2019; Chen et al., 2020; Zheng et al., 2019). However, all of them are based on autoregressive decoding, i.e., conditioning each word on the previously generated outputs. Therefore, to control the generation of image captions, a major challenge is to decide the timing to attend to the region-of-interest (i.e., the object we care about). Zheng et al. (2019) first fixes the cared object and generates the rest captions to its left and right which can only apply to the case with a single cared object. To scale to multiple cared objects, Cornia et al. (2019) implements a region pointer mechanism to predict, at each timestep, whether this pointer should be incremented or not; Chen et al. (2020) introduces the abstract scene graph, to control the generation of captions, they proposed graph-based attention and graph updating mechanisms to adaptively select relevant nodes, which contain the concerned objects to generate next word.

In this work, we focus on controllable video captioning, which is a more challenging problem than controllable image captioning. It is hard for controllable video captioning to construct the same region-of-interests (RoIs) as in Cornia et al. (2019) and scene graphs as in Chen et al. (2020). To this end, based on the non-autoregressive decoding methods in neural machine translation (Gu et al., 2018; Lee et al., 2018; Ghazvininejad et al., 2019; Wang et al., 2019b; Shao et al., 2019), we propose Object-Oriented Non-Autoregressive model, which does not need the RoIs in Cornia et al. (2019) or scene graphs in Chen et al. (2020) to generate controllable video captions. Moreover, our approach can generate all the objects we care about in parallel, leading to fast generation speed.

It is worth noting that Wang et al. (2019a); Yuan et al. (2020) also introduced the controllable video captioning. However, they devoted to employing Part-of-Speech (POS) information to guide caption generation, which mainly focuses on improving diversity and adjusting the syntactic structure of the captions, instead of constraining the model to generate captions containing the focused objects.

2.3 Non-Autoregressive Decoding

Most recently, non-autoregressive decoding has received growing attention in the community of neural machine translation (NMT) (Gu et al., 2018; Ghazvininejad et al., 2019; Lee et al., 2018; Guo et al., 2019; Shao et al., 2019; Ghazvininejad et al., 2020; Kasai et al., 2020; Ren et al., 2020; Haviv et al., 2021; Hao et al., 2021). Such models remove the sequential dependency and can generate all words of a sequence in one step, resulting in high inference efficiency. Inspired by the success of non-autoregressive decoding, we propose the Object-Oriented Non-Autoregressive model. As for the network structure, these current non-autoregressive models usually employ a completely empty sequence as the input of decoder to generate the whole sentence in the early stages, which gives a high risk of producing translation errors. Different from these works, we consider exploiting the objects in the video and propose to first generate an object-oriented coarse-grained caption, and then refine each object word with rich contextual information to generate the whole caption to alleviate the description ambiguity problem.

3 Approach

We first briefly introduce the backgrounds of our approach and then describe the approach in detail.

3.1 Backgrounds

The backgrounds are introduced from the used Video Representations and Basic Module.

Video Representations For video captioning, image and motion features have been widely used. Image features are good at illustrating the shapes, the colors and the relationships of the items in the image; Motion features are important for capturing the actions and temporal interactions. Following Pei et al. (2019), given a video, $N = 8$ key frames are uniformly sampled to extract image features I. Considering both the past and the future contexts, we take each key frame as the center to generate corresponding motion features M. Specifically, for the image features, we adopt the ResNet-101 (He et al., 2016) pre-trained on ImageNet (Deng et al., 2009) to extract the 2048-D image features $I \in \mathbb{R}^{N \times d_i}$ ($d_i = 2048$), which are the output of the last convolutional layer. The motion features are usually given by the 3D CNN (Tran et al., 2015), we adopt the ResNeXt-101 (Hara et al., 2018) pre-trained on the Kinetics dataset (Kay et al., 2017) to extract the 2048-D motion features $M \in \mathbb{R}^{N \times d_m}$ ($d_m = 2048$). In this paper, both features are projected to $d_h = 512$. Then, we use the concatenation of the two projected features as the video representations $V \in \mathbb{R}^{2N \times d_h}$ to our model.

Basic Module Our approach is adapted from the non-autoregressive decoding models (Lee et al., 2018; Ghazvininejad et al., 2019), which is based on the Transformer decoder (TFM) (Vaswani et al., 2017). Specifically, the TFM consists of a self-attention, a source-attention and a feed-forward network (FF). The multi-head attention (MHA) is the basic of self-attention and source-attention. Overall, the TFM is defined as follows:

$$\text{TFM}(Q, K, V) = \text{FF}(\text{MHA}(\text{MHA}(Q, Q, Q), K, V)). \quad (1)$$

Please refer to Vaswani et al. (2017) for the detailed introduction of the Transformer decoder (TFM).

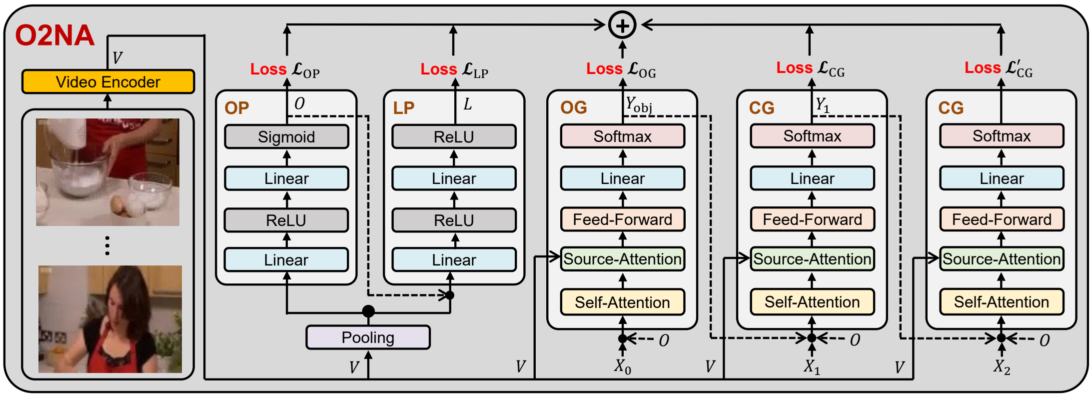

Figure 2: Illustration of our proposed O2NA, which consists of an object predictor (OP), a length predictor (LP), an object generator (OG) and a caption generator (CG). The object predictor and length predictor extract the objects appearing to the input video and estimate the length of target caption, respectively; The object generator locates all the focused objects we care about in the target caption; The caption generator generates the rest words to link focused objects to form a fluent caption. It is worth noting that the focused objects could be the objects predicted by the object predictor, the preferred objects given by the user or the pre-defined concerned objects, e.g., the dangerous objects in the captioning-based blind-aid system.

3.2 Object-Oriented Non-Autoregressive Approach (O2NA)

As stated above, we adopt the Transformer decoder (Vaswani et al., 2017) to implement our Object-Oriented Non-Autoregressive approach (O2NA). Specifically, as shown in Figure 2, O2NA consists of an object predictor, a length predictor and two Transformer decoders, where the first decoder focuses on generating all the objects we care about in parallel (i.e., object generator), and the second decoder pays attention to linking these objects to form a fluent caption (i.e., caption generator).

Object Predictor (OP) The OP is expected to predict the objects that appear in the given video. We first build an object vocabulary based on the training captions. Given this object vocabulary, we can associate each video with a set of objects according to its human-annotated captions. Specifically, we denote the ground truth objects as $O^* = \{o_1^*, o_2^*, \dots, o_M^*\}$, where M represents the size of object vocabulary; $o_i^* = 1$ if the video is annotated with object i, and $o_i^* = 0$ otherwise. During the training phase, we directly use the ground truth objects O^*. At the inference stage, we adopt a two-layer non-linear layer to predict the objects $O \in \mathbb{R}^M$, defined as:

$$\begin{aligned} O &= \text{Object-Predictor}(V) \\ &= \sigma\left(\text{ReLU}\left(\text{MP}\left(V\right) W_{\text{O}_1}\right) W_{\text{O}_2}\right) \\ \text{where } \ \text{MP}\left(V\right) &= \frac{1}{2N}\sum_{i=1}^{2N} v_i, \end{aligned} \tag{2}$$

where MP denotes the Mean Pooling, σ is the sigmoid function; $W_{\text{O}_1} \in \mathbb{R}^{d_h \times d_h}$ and $W_{\text{O}_2} \in \mathbb{R}^{d_h \times M}$ are the parameters to be learned. Next, following Wu et al. (2016), we minimize the element-wise logistic loss function $\mathcal{L}_{\text{OP}}$ to train our OP:

$$\mathcal{L}_{\text{OP}} = \sum_{i=1}^{M} \log\left(1 + \exp\left(-o_i^* o_i\right)\right). \tag{3}$$

During the inference procedure, to select the final predicted objects, we set a threshold γ, which means that if the $o_i > \gamma$, we reset $o_i = 1$, and reset $o_i = 0$ otherwise. In particular, if we care about some specific objects, for example, the user preferred objects or the pre-defined dangerous objects in the captioning-based blind-aid system, we could just set the value of these concerned objects equal to 1, and set the value of other objects equal to 0.

Length Predictor (LP) In the generation process, the non-autoregressive decoding model needs to know the length of target captions (Ghazvininejad et al., 2019). To this end, at training time, we use the sequence length l^* of ground truth caption. At inference stage, given the video information $V \in \mathbb{R}^{2N \times d_h}$ and the focused objects $O \in \mathbb{R}^M$, we adopt a LP to predict the length l. In detail, we apply a two-layer network to achieve the effect:

$$\begin{aligned} l \sim p_l &= \text{Length-Predictor}(V, O) \\ &= \text{softmax}\left(\text{ReLU}\left([\text{MP}(V) W_{\text{L}_V}; O W_{\text{L}_O}]\right) W_{\text{L}}\right), \end{aligned} \tag{4}$$

where $[\cdot;\cdot]$ represents the concatenation operation; $W_{\text{L}_V} \in \mathbb{R}^{d_h \times d_h}$, $W_{\text{L}_O} \in \mathbb{R}^{M \times d_h}$ and $W_{\text{L}} \in \mathbb{R}^{2d_h \times l_{max}}$ are learnable parameters; $l_{max} = 30$ denotes the pre-defined maximum sequence length. Thus, $p_l \in \mathbb{R}^{l_{max}}$ is a probability. We adopt the cross entropy loss $\mathcal{L}_{\text{LP}}$ to train the LP, which can be defined as follows:

$$\mathcal{L}_{\text{LP}} = -\log(p_l(l^*|V, O^*)). \tag{5}$$

Object Generator (OG) The object generator is based on the non-autoregressive decoder and is dedicated to generating all the objects we care about at once. To achieve such effect, we adopt a single-layer Transformer decoder[1], followed by a linear layer and a softmax function. In implementation, the object generator takes the fully masked sequence $X_0 = (x_{\text{m}_1}, x_{\text{m}_2}, \dots, x_{\text{m}_L})\,, x_{\text{m}_i} \in \mathbb{R}^{d_h}$ with predicted length l by length predictor as input. The $x_{\text{m}_i} = w_{\text{[MASK]}} + e_i$, where $w_{\text{[MASK]}}$ and e_i denotes the word embedding of [MASK] token and position embedding, respectively. Then the object information O is added to X_0, i.e., $x'_{\text{m}_i} = x_{\text{m}_i} + OW_O$, where $W_O \in \mathbb{R}^{M \times d_h}$. At last, the transformer decoder in the object generator takes the $X_0 \oplus OW_O$ as input ($\oplus$ denotes the matrix-vector addition), and generates all objects at the position in the final caption, i.e., an object-oriented coarse-grained caption, which can be defined as follows:

$$\begin{aligned} Y_{\text{obj}} \sim p_0 &= \text{Object-Generator}(X_0, V, O) \\ &= \text{softmax}(\text{TFM}(X_0 \oplus OW_O, V, V)W_{\text{OG}}), \end{aligned} \tag{6}$$

where $X_0 \in \mathbb{R}^{l \times d_h}$, $V \in \mathbb{R}^{2N \times d_h}$, $O \in \mathbb{R}^{M}$ represent the input sequence, the video representations and the predicted objects, respectively; $W_O \in \mathbb{R}^{M \times d_h}$ and $W_{\text{OG}} \in \mathbb{R}^{d_h \times |D|}$ are the matrices for linear transformation; $|D|$ is the size of vocabulary D. Each value of $p_0 \in \mathbb{R}^{l \times |D|}$ is a probability indicating how likely each word in D should be the current output word.

At training time, for each human-annotated caption, we mask all the non-object words based on the object vocabulary to acquire the ground truth object sequence $Y^*_{\text{obj}} = (\dots, \text{[MASK]}, \dots, \text{object}_i, \dots)$. Our goal is to minimize the following standard cross entropy loss:

$$\mathcal{L}_{\text{OG}} = -\sum\nolimits_{i=1}^{l^*} \log(p_0(y^*_{\text{obj}_i}|X_0, V, O^*)). \tag{7}$$

Caption Generator (CG) In implementation, the caption generator shares the same structure with object generator. The main differences between the two generators are the different generating objective and the input sequence. Specifically, the caption generator takes the object sequence X_1 as input, where X_1 equals to Y^*_{obj} and Y_{obj} at the training stage and inference stage, respectively, and generates the related attribute words and relation words to form a draft caption, which is defined as:

$$\begin{aligned} Y_1 \sim p_1 &= \text{Caption-Generator}(X_1, V, O) \\ &= \text{softmax}(\text{TFM}(X_1 \oplus OW'_O, V, V)W_{\text{CG}}), \end{aligned} \tag{8}$$

where $p_1 \in \mathbb{R}^{l \times |D|}$. Given the ground truth caption $Y^*_{\text{cap}} = (y^*_{\text{cap}_1}, y^*_{\text{cap}_2}, \dots, y^*_{\text{cap}_l})$, we adopt standard cross entropy loss as the loss function to train the CG, which can be defined as follows:

$$\mathcal{L}_{\text{CG}} = -\sum\nolimits_{i=1}^{l^*} \log(p_1(y^*_{\text{cap}_i}|X_1, V, O^*)). \tag{9}$$

Since the non-autoregressive approach removes the sequential dependency, we may have introduced the "multi-modality problem" (Gu et al., 2018) (i.e., a word could appear in multiple position to form different captions). So we further adopt the iterative refinement approach (Lee et al., 2018) to proofread Y_1. In implementation, to acquire the input sequence X_2, we randomly mask $n = \lfloor l * r \rfloor$ words in Y^*_{cap} and mask out top n words with the lowest confidence in Y_1 at the training time and inference time, respectively, where l and r represent the caption length and masking ratio, respectively, and the confidence is taken to be the output probability. To obtain the final caption, we employ the following equation, which is defined as:

$$Y_2 \sim p_2 = \text{Caption-Generator}(X_2, V, O). \tag{10}$$

Finally, the cross entropy loss is defined similar as Eq. (9):

$$\mathcal{L}'_{\text{CG}} = -\sum\nolimits_{i=1}^{l^*} \log(p_2(y^*_{\text{cap}_i}|X_2, V, O^*)). \tag{11}$$

Overall, by combining the $\mathcal{L}_{\text{OP}}$ in Eq. (3), $\mathcal{L}_{\text{LP}}$ in Eq. (5), $\mathcal{L}_{\text{OG}}$ in Eq. (7), $\mathcal{L}_{\text{CG}}$ in Eq. (9) and $\mathcal{L}'_{\text{CG}}$ in Eq. (11), the full training objective is:

$$\mathcal{L}_{\text{full}} = \lambda_1\mathcal{L}_{\text{LP}} + \lambda_2\mathcal{L}_{\text{OP}} + \lambda_3\mathcal{L}_{\text{OG}} + \lambda_4\mathcal{L}_{\text{CG}} + \lambda_5\mathcal{L}'_{\text{CG}}, \tag{12}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and λ_5 are the hyperparameters that control the regularization. For simplicity, we set $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 1$, since we find that our approach can achieve competitive results with the state-of-the-art models in major metrics under this setting (see Section 4.2), thus we do not attempt to explore other settings.

Overall, through Eq. (12), we are able to realize our Object-Oriented Non-Autoregressive approach (O2NA). The trained model is encouraged to describe the focused objects that a user cares about.

---

[1] Our experiments showed that using a single-layer Transformer decoder can achieve the best performance in major metrics with fastest inference speed (Please refer to Section 5.1.3).

4 Experiments

In this section, we first describe the datasets, metrics and settings used for evaluation, then followed by the experimental results of our approach.

4.1 Datasets, Metrics and Settings

4.1.1 Datasets

Our results are evaluated on the benchmark Microsoft Video Description (MSR-VTT) (Xu et al., 2016) and Microsoft Video Description (MSVD) (Guadarrama et al., 2013) datasets. For MSR-VTT, the dataset contains 10,000 video clips, and each video is paired with 20 annotated sentences. Following common practice (Pei et al., 2019; Yang et al., 2021; Pan et al., 2020), we use the official splits to report our results. Thus, there are 6513, 497 and 2990 video clips in the training set, validation set and test set, respectively. For MSVD, it contains 1,970 video clips and roughly 80,000 English sentences. We follow the split settings in Pei et al. (2019), resulting in 1,200, 100 and 670 videos for the training set, validation set and test set, respectively. Following previous works, we replace caption words that occur less than 3 times in the training set with the [UNK] token, plus with a [MASK] token, resulting in a vocabulary of 10,546 words for MSR-VTT and 9,467 words for MSVD.

Metrics We test the model performance with a standard captioning evaluation toolkit (Chen et al., 2015). It reports the widely-used automatic evaluation metrics CIDEr (Vedantam et al., 2015), ROUGE-L (Lin, 2004), METEOR (Lin and Hovy, 2003; Banerjee and Lavie, 2005) and BLEU (Papineni et al., 2002). Among them, CIDEr, which incorporates the consensus of a reference set for an example, is based on n-gram matching, is specifically designed for evaluating captioning systems. BLEU and METEOR are originally designed for machine translation evaluation, while ROUGE-L is proposed for automatic evaluation of the extracted text summarization. Besides, we further adopt the evaluation metrics Novel, Unique and Vocab Usage, provided by Dai et al. (2018), to evaluate the diversity of the generated captions. Novel is calculated by the percentage of generated captions that have not been seen in the training data; Unique is calculated by the percentage of generated unique words among the other all generated captions; Vocab Usage denotes the percentage of words that are used to generate captions in the vocabulary.

4.1.2 Settings

As stated in Section 3.1, we set $N = 8$, $d_i = d_m = 2048$ and $d_h = 512$ for the video representations. All category tags (Xu et al., 2016) included in MSR-VTT. For the object predictor, to compare with existing methods, we set the threshold $\gamma = 0.8$ and directly select all the predicted objects to generate captions. For the length predictor, the maximum sequence length l_{max} is set to 30. For the object generator and caption generator, following the original setting as in Transformer (Vaswani et al., 2017), the model size $d_h = 512$. The number of heads in multi-head attention is set to 8 and the feed-forward network dimension is set to 2048. The masking ratio $r = 0.5$. To build the object vocabulary, we use the spaCy library[2] for noun tagging from the training dataset, resulting in 5,647 and 4,681 noun words for MSR-VTT and MSVD, respectively. The tagged noun words are taken as the object words, building up the object vocabulary with sizes of 5,647 and 4,681 for MSR-VTT and MSVD, respectively. Therefore, we do not use external data to build the object vocabulary. Specifically, the object predictor labels will match the words used to name objects in the captions. We use Adam optimizer (Kingma and Ba, 2014) with a batch size of 64 and a learning rate of 5e-4 within maximum 50 epochs for parameter optimization.

As each video is annotated with multiple sentences, i.e., Video – $\{\text{Caption}_i\}$, where each sentence Caption_i includes a set of objects $\{\text{Object}_i\}$, we use all objects appearing in these sentences as the ground truth objects for each video to train the object predictor. However, we treat the different sentences as independent training samples, i.e., Video – Caption_i – $\{\text{Object}_i\}$, to train length predictor, object generator and caption generator. In this manner, we can ensure that the focused objects $\{\text{Object}_i\}$ appears in the target sentence Caption_i during training and inference, which allows an easy way to control the contents of video captions.

Following the non-autoregressive decoding models of neural machine translation, we incorporate the knowledge distillation (Kim and Rush, 2016; Gu et al., 2018) and de-duplication (Wang et al., 2019b) techniques to improve the performance of our non-autoregressive model on MSR-VTT. Furthermore, following Gu et al. (2018); Wang et al. (2019b); Yang et al. (2021), to generate the captions, we also adopt the teacher re-scoring tech-

---

[2] `https://spacy.io/`

Methods	Dataset: MSVD (Guadarrama et al., 2013)				Dataset: MSR-VTT (Xu et al., 2016)							
	BLEU-4	METEOR	ROUGE-L	CIDEr	BLEU-4	METEOR	ROUGE-L	CIDEr	Novel	Unique	Vocab	VPS
RecNet (Wang et al., 2018)	52.3	34.1	69.8	80.3	39.1	26.6	59.3	42.7	-	-	-	-
PickNet (Chen et al., 2018)	52.3	33.3	69.6	76.5	41.3	27.7	59.8	44.1	-	-	-	-
OA-BTG (Zhang and Peng, 2019)	56.9	36.2	-	90.6	41.4	28.2	-	46.9	-	-	-	-
MARN (Pei et al., 2019)	48.6	35.1	71.9	92.2	40.4	28.1	60.7	47.1	-	-	-	-
GRU-EVE (Aafaq et al., 2019)	47.9	35.0	71.5	78.1	38.3	28.4	60.7	48.1	-	-	-	-
POS-Control (Wang et al., 2019a)	52.5	34.1	71.3	88.7	42.0	28.2	61.6	48.7	-	-	-	-
STAT (Yan et al., 2020)	52.0	33.3	-	73.8	39.3	27.1	-	43.8	-	-	-	-
STGN-OAKD (Pan et al., 2020)	52.2	36.9	73.9	93.0	40.5	28.3	60.9	47.1	-	-	-	-
ORG-TRL (Zhang et al., 2020)	54.3	36.4	73.9	95.2	43.6	28.8	62.1	50.9	-	-	-	-
SAAT (Zheng et al., 2020)	46.5	33.5	69.4	81.0	39.9	27.7	61.2	51.0	26.8†	35.7†	3.9†	17.6†
SGN (Ryu et al., 2021)	52.8	35.5	72.9	94.3	40.8	28.3	60.8	49.5	-	-	-	-
SemSynAN (Perez-Martin et al., 2021)	64.4	41.9	79.5	111.5	46.4	30.4	64.7	51.9	-	-	-	-
O2NA (Ours)	55.4	37.4	74.5	96.4	41.6	28.5	62.4	51.1	37.2	46.7	4.6	70.8

Table 1: Performance of automatic evaluation on the test sets of MSVD and MSR-VTT. Higher is better in all columns. † denotes our own implementation. VPS stands for videos per second at the inference stage, which is measured on a single NVIDIA GeForce GTX 1080 Ti. In this paper, the Red- and the Blue- colored numbers denote the best and the second best results across all approaches, respectively. All existing video captioning systems follow the autoregressive approach to generate the captions and cannot control the video captioning process to ensure the inclusion of the focused objects. In comparison, O2NA can not only describe the focused objects, but also achieve competitive performances with the state-of-the-arts in major metrics with both higher diversity and faster inference.

nique and noisy parallel decoding (Gu et al., 2018; Yang et al., 2021) techniques, which could generate a set of candidate sentences in parallel, then, we select the candidate sentence with the highest output probability as the final generated caption. For the detailed introduction of these techniques, please refer to original papers (Kim and Rush, 2016; Gu et al., 2018; Wang et al., 2019b; Yang et al., 2021).

4.2 Evaluation Results

In comparable settings, twelve representative methods, including five most recently published state-of-the-art approaches, namely STAT (Yan et al., 2020), STGN-OAKD (Pan et al., 2020), ORG-TRL (Zhang et al., 2020), SAAT (Zheng et al., 2020), SGN (Ryu et al., 2021) and SemSynAN (Perez-Martin et al., 2021), are selected for comparison. Unless specifically stated, we directly report the results from the original papers. The results on the test of MSVD and MSR-VTT datasets are shown in Table 1. As we can see, our O2NA achieves the results competitive with the state-of-the-art models on the two datasets in major metrics. The competitive performances verify the validity of our O2NA for standard video captioning. More encouragingly, in terms of the metrics that evaluate the diversity of the generated captions, O2NA surpasses the previous state-of-the-art models with relatively 39%, 31% and 18% margins in terms of Novel, Unique and Vocab scores, which proves our arguments and corroborates the effectiveness of our approach. Moreover, since our O2NA generate the entire captions in three steps with a fixed generation time, we achieve the fastest inference speed (highest VPS in Table 1) among existing methods.

Overall, our O2NA achieves performances competitive with state-of-the-arts in major metrics but with higher diversity scores and faster inference speed. The experimental results show that our approach is able to generate fluent and diverse video captions with fast inference speed. More importantly, our O2NA allows an easy way to control the contents of video captions rather than merely syntactic variations in existing studies. These advantages of our approach could have the potential to promote the application of video captioning for real-time industrial applications, e.g., helping visually impaired people see (Voykinska et al., 2016) and human-robot interaction (Das et al., 2017).

5 Analysis

In this section, we conduct analysis on the benchmark MSR-VTT dataset from different perspectives to better understand our approach.

5.1 Quantitative Analysis

We first conduct the quantitative analysis to investigate the contribution of each component in our proposed O2NA.

5.1.1 Ablation Study

Compared to conventional non-autoregressive decoding models (Baseline) from neural machine translation (Lee et al., 2018; Gu et al., 2018; Ghazvininejad et al., 2019), our O2NA further introduces the *object predictor* and *object generator*

Sections	Settings	Methods	Iteration Times	Number of Layers	Dataset: MSR-VTT							
					BLEU-4	METEOR	ROUGE-L	CIDEr	Novel	Unique	Vocab	VPS
5.1.1	(a)	Baseline	1	1	40.0	26.9	60.2	44.6	6.6	27.1	2.6	113.7
	(b)	w/ OP	1	1	40.7	27.4	60.6	47.9	18.0	26.9	3.1	99.5
	O2NA	w/ OP + OG	1	1	41.6	28.5	62.4	51.1	37.2	46.7	4.6	70.8
5.1.2	(c)	w/ OP + OG	2	1	42.1	28.7	62.5	51.6	31.9	42.3	4.0	61.0
	(d)	w/ OP + OG	3	1	42.4	28.8	62.5	51.8	25.1	33.0	3.5	54.9
	(e)	w/ OP + OG	4	1	42.5	28.8	62.6	51.9	21.1	29.3	3.0	49.3
5.1.3	(f)	w/ OP + OG	1	2	41.8	28.5	62.1	50.8	36.0	43.7	4.5	48.5
	(g)	w/ OP + OG	1	3	41.1	28.4	61.5	50.3	30.4	38.6	3.9	36.9
	(h)	w/ OP + OG	1	4	40.5	27.6	61.0	48.7	22.3	30.6	3.4	30.2

Table 2: Quantitative analysis of O2NA. Baseline denotes the conventional non-autoregressive decoding model in neural machine translation (Lee et al., 2018; Gu et al., 2018; Ghazvininejad et al., 2019). OP and OG denote the object predictor and object generator, respectively.

for controllable video captioning. Therefore, we investigate the contribution of the two components and the results are shown in Table 2.

Effect of the Object Predictor (OP) As expected, since the OP can provide explicit visual clues (i.e., objects) of the input video, the model achieves improved results (c.f. Table 2(b)), especially in Novel and Unique scores, indicating that the OP helps to generate diverse captions. The improved results prove the effectiveness of our OP.

Effect of the Object Generator (OG) As shown in Table 2(O2NA), when further equipping with the OG, the model significantly outperforms the Baseline, which employs a completely empty sequence as the input to generate the whole sentence. Intuitively, such practice in Baseline may give high risk of producing errors. Fortunately, the object-oriented coarse-grained captions generated by our OG could provide rich contextual information for the following non-autoregressive decoding model to generated accurate revised captions. It proves our arguments and verifies the effectiveness of generating captions in a coarse-grained to fine-grained manner.

Overall, the proposed OP and OG can boost the performance from different perspectives, making our O2NA generate diverse and accurate captions.

5.1.2 Effect of the Iteration Times

In O2NA, we adopt the iterative refinement technique (Lee et al., 2018) to proofread and improve the generated captions (see Eq. (10)). However, in conventional non-autoregressive decoding methods for neural machine translation (Gu et al., 2018; Lee et al., 2018; Ghazvininejad et al., 2019; Guo et al., 2019; Shao et al., 2019), they usually adopt more iterations to obtain better results. As to O2NA, Table 2(c-e) shows that performances stabilize with the increasing number of iterations but do not show a significant increase as in Lee et al. (2018); Ghazvininejad et al. (2019). The reason is that our generated object-oriented coarse-grained captions have provided a solid guidance (i.e., rich contextual information) for non-autoregressive video captioning model, which further proves the effectiveness of our approach. The decreased performance of diversity may be due to the over-fitting problem brought by more iterations, making the model prone to generating frequent captions in the training data. Thus, considering the trade-off between "the performance of caption generation" and "the performance of diversity and inference speed", we only proofread the generated captions once.

5.1.3 Effect of the Number of Layers

When increasing the number of layers to 2 (c.f. Table 2(f)), the model can only achieve a slightly improved result on BLEU-4 (i.e., 41.6 → 41.8), but loses 31.5% inference speed. At the same time, if the number of layers is further increased, the performance decreases. We hypothesize that when training on video captioning datasets that are relatively small compared to those for neural machine translation, larger depths add to the difficulty of training, which is the same case with deep RNNs. In brief, considering the trade-off between the performance and inference speed, we adopt a single-layer Transformer decoder.

5.2 Case Study and Error Analysis

In this section, we list some correct and incorrect examples to show the controllability of our proposed O2NA intuitively. In the analysis, we manually select the predicted objects to encourage the model to generate a set of diverse captions. Fig-

(person, electronics) a person is showing a electronics.
(hand, electronics) a hand pointing to a electronics.
(electronics, device) an electronics device is shown.

(person, suitcase) a person is describing a suitcase.
(device, suitcase) a device sitting on a suitcase.
(suitcase) a suitcase is opening.

(woman, kitchen) a woman is in a kitchen .
(woman, food, pan) a woman is cooking some food with a pan.

(woman, shirt) a woman wearing a black shirt.
(woman, food, shirt) a woman with a black shirt is preparing food.

Figure 3: Examples of captions generated by our proposed O2NA. For each example, the left plot shows the input video. The upper, middle and lower parts in the right plot show the predicted objects, correct examples and error cases, respectively. The designated objects are listed in brackets. The color Red denotes unfavorable objects.

ure 3 shows that our approach is controllable and explainable. Specifically, it can generate multiple diverse captions for the same video, and can accurately follow the selected objects we care about. Besides, we find that the error mainly takes place when there are incorrectly predicted objects, e.g., “*suitcase*” and “*shirt*”. O2NA mistakes the incorrect object for an appropriate one during its object sequence generation. A more powerful object predictor may be helpful in solving these problems, but it is unlikely to be completely avoided.

6 Conclusions

In this work, we introduce the problem of controllable video captioning in the sense of controlled contents. In contrast to the existing studies considering syntactic variations, controlling contents is of more practical value. To tackle the problem, we propose the Object-Oriented Non-Autoregressive approach (O2NA), which encourages the model to describe the focused objects that a user cares about by generating captions conditioned on the focused objects non-autoregressively. The experiments and analyses verify the flexibility and demonstrate the effectiveness of O2NA, which achieves competitive results with existing state-of-the-art models on two benchmark datasets in major metrics with higher diversity and faster inference. These advantages could promote the application of video captioning adapting to real-world scenarios.

In the future, it may be interesting to implement a more fine-grained control of the objects, e.g., the order of the objects in the caption according to their priority in utterance.

Acknowledgments

This work is supported in part by Beijing Academy of Artificial Intelligence (BAAI). We sincerely thank all the anonymous reviewers and chairs for their constructive comments and suggestions that substantially improved this paper. We also sincerely thank Bang Yang for providing the implementation code of non-autoregressive framework for video captioning.[3] Xu Sun is the corresponding author of this paper.

Impact Statement

This paper introduces the problem of controllable video captioning in the sense of controlled contents to efficiently understand the visual content of a given video and generate corresponding descriptive sentences. As a result, our work can control the video captioning process and include focused objects, i.e., the video captions generated by our model are more likely to contain preferred objects given by a user or pre-defined objects that should be prioritized in generation. It improves the practicality of video captioning in real-world applications, such as visual retrieval, human-robot interaction and aiding visually-impaired people. However, the training of our proposed model relies on a large volume of video-caption pairs, which may not be easily obtained in the real world but could be alleviated using techniques such as distillation from publicly-available pre-trained models. Hence, it requires specific and appropriate treatment by experienced practitioners.

References

Nayyer Aafaq, Naveed Akhtar, Wei Liu, Syed Zulqarnain Gilani, and Ajmal Mian. 2019. Spatio-temporal dynamics and semantic attribute enriched visual encoding for video captioning. In *CVPR*.

Nayyer Aafaq, Ajmal Mian, Wei Liu, Syed Zulqarnain Gilani, and Mubarak Shah. 2020. Video description: A survey of methods, datasets, and evaluation metrics. *ACM Comput. Surv.*

---

[3] `https://github.com/yangbang18/Non-Autoregressive-Video-Captioning`

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and VQA. In *CVPR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL*.

Shizhe Chen, Qin Jin, Peng Wang, and Qi Wu. 2020. Say as you wish: Fine-grained control of image caption generation with abstract scene graphs. In *CVPR*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Yangyu Chen, Shuhui Wang, Weigang Zhang, and Qingming Huang. 2018. Less is more: Picking informative frames for video captioning. In *ECCV*.

Maurizio Corbetta and Gordon L Shulman. 2002. Control of goal-directed and stimulus-driven attention in the brain. *Nature reviews neuroscience*, 3(3):201–215.

Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2019. Show, control and tell: A framework for generating controllable and grounded captions. In *CVPR*.

Bo Dai, Sanja Fidler, and Dahua Lin. 2018. A neural compositional paradigm for image captioning. In *NeurIPS*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *CVPR*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE Computer Society.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. In *ICML*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *AAAI*.

Yongchang Hao, Shilin He, Wenxiang Jiao, Zhaopeng Tu, Michael R. Lyu, and Xing Wang. 2021. Multi-task learning with shared encoder for non-autoregressive machine translation. In *NAACL*.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. 2018. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*.

Adi Haviv, Lior Vassertail, and Omer Levy. 2021. Can latent alignments improve autoregressive machine translation? In *NAACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *ICML*.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *ACL*.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL-HLT*.

Fenglin Liu, Yuanxin Liu, Xuancheng Ren, Xiaodong He, Kai Lei, and Xu Sun. 2019a. Aligning visual regions and textual concepts for semantic-grounded image representations. In *NeurIPS*.

Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Kai Lei, and Xu Sun. 2019b. Exploring and distilling cross-modal information for image captioning. In *IJCAI*.

Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Houfeng Wang, and Xu Sun. 2018. simnet: Stepwise image-topic merging network for generating detailed and comprehensive image captions. In *EMNLP*.

Fenglin Liu, Xuancheng Ren, Xian Wu, Shen Ge, Wei Fan, Yuexian Zou, and Xu Sun. 2020. Prophet attention: Predicting attention with future attention. In *NeurIPS*.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*.

Boxiao Pan, Haoye Cai, De-An Huang, Kuan-Hui Lee, Adrien Gaidon, Ehsan Adeli, and Juan Carlos Niebles. 2020. Spatio-temporal graph for video captioning with knowledge distillation. In *CVPR*.

Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016a. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*.

Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016b. Jointly modeling embedding and translation to bridge video and language. In *CVPR*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for automatic evaluation of machine translation. In *ACL*.

Wenjie Pei, Jiyuan Zhang, Xiangrong Wang, Lei Ke, Xiaoyong Shen, and Yu-Wing Tai. 2019. Memory-attended recurrent network for video captioning. In *CVPR*.

Jesus Perez-Martin, Benjamin Bustos, and Jorge Perez. 2021. Improving video captioning with temporal composition of a visual-syntactic embedding. In *WACV*.

Michael I Posner and Steven E Petersen. 1990. The attention system of the human brain. *Annual review of neuroscience*, 13(1):25–42.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *ACL*.

Hobin Ryu, Sunghun Kang, Haeyong Kang, and Chang D. Yoo. 2021. Semantic grouping network for video captioning. In *AAAI*.

Chenze Shao, Yang Feng, Jinchao Zhang, Fandong Meng, Xilin Chen, and Jie Zhou. 2019. Retrieving sequential information for non-autoregressive neural machine translation. In *ACL*.

Barbara G Shinn-Cunningham. 2008. Object-based auditory and visual attention. *Trends in cognitive sciences*, 12(5):182–186.

Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *NAACL-HLT*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Violeta Voykinska, Shiri Azenkot, Shaomei Wu, and Gilly Leshed. 2016. How blind people interact with visual content on social networking services. In *CSCW*.

Bairui Wang, Lin Ma, Wei Zhang, Wenhao Jiang, Jingwen Wang, and Wei Liu. 2019a. Controllable video captioning with POS sequence guidance based on gated fusion network. In *ICCV*.

Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. 2018. Reconstruction network for video captioning. In *CVPR*.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019b. Non-autoregressive machine translation with auxiliary regularization. In *AAAI*.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony R. Dick, and Anton van den Hengel. 2016. What value do explicit high level concepts have in vision to language problems? In *CVPR*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*.

Jun Xu, Ting Yao, Yongdong Zhang, and Tao Mei. 2017. Learning multimodal attention LSTM networks for video captioning. In *ACM MM*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Chenggang Yan, Yunbin Tu, Xingzheng Wang, Yongbing Zhang, Xinhong Hao, Yongdong Zhang, and Qionghai Dai. 2020. STAT: spatial-temporal attention mechanism for video captioning. *IEEE Trans. Multim.*

Bang Yang, Fenglin Liu, and Yuexian Zou. 2019. Non-autoregressive video captioning with iterative refinement. *arXiv preprint arXiv:1911.12018*.

Bang Yang, Yuexian Zou, Fenglin Liu, and Can Zhang. 2021. Non-autoregressive coarse-to-fine video captioning. In *AAAI*.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher J. Pal, Hugo Larochelle, and Aaron C. Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV*.

Yitian Yuan, Lin Ma, Jingwen Wang, and Wenwu Zhu. 2020. Controllable video captioning with an exemplar sentence. In *ACM Multimedia*.

Junchao Zhang and Yuxin Peng. 2019. Object-aware aggregation with bidirectional temporal graph for video captioning. In *CVPR*.

Ziqi Zhang, Yaya Shi, Chunfeng Yuan, Bing Li, Peijin Wang, Weiming Hu, and Zheng-Jun Zha. 2020. Object relational graph with teacher-recommended learning for video captioning. In *CVPR*.

Qi Zheng, Chaoyue Wang, and Dacheng Tao. 2020. Syntax-aware action targeting for video captioning. In *CVPR*.

Yue Zheng, Yali Li, and Shengjin Wang. 2019. Intention oriented image captions with guiding objects. In *CVPR*.

Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong. 2018. End-to-end dense video captioning with masked transformer. In *CVPR*.

Better Chinese Sentence Segmentation with Reinforcement Learning

Srivatsan Srinivasan
DeepMind
`vatsan@deepmind.com`

Chris Dyer
DeepMind
`cdyer@deepmind.com`

Abstract

A long-standing challenge in Chinese–English machine translation is that sentence boundaries are ambiguous in Chinese orthography, but inferring good splits is necessary for obtaining high quality translations. To solve this, we use reinforcement learning to train a segmentation policy that splits Chinese texts into segments that can be independently translated so as to maximise the overall translation quality. We compare to a variety of segmentation strategies and find that our approach improves the baseline BLEU score on the WMT2020 Chinese–English news translation task by +0.3 BLEU overall and improves the score on input segments that contain more than 60 words by +3 BLEU.

1 Introduction

Machine translation systems typically operate on sentence-like units, where sentences are translated independently of each other (Vaswani et al., 2017; Bahdanau et al., 2015; Koehn et al., 2003; Brown et al., 1993), in some cases with additional conditioning on a representation of adjacent sentences to improve coherence (Miculicich et al., 2018; Zhang et al., 2018). While many pairs of languages use similar orthographic conventions to designate sentence boundaries, English and Chinese diverge considerably: complete sentences in Chinese may be terminated either unambiguously with a full stop (。) or ambiguously with a comma. See Figure 1 for an example.

This divergence poses a challenge for Chinese–English translation systems since they must either be able to cope with potentially long, multi-sentence inputs (i.e., translating any text that falls between unambiguous sentence-ending punctuation) or, alternatively, they must be able to determine which comma occurrences terminate complete sentences that can be translated independently and which do not.

习近平主席2015年9月宣布成立的“中国-联合国和平与发展基金”，将支持非洲维和、反恐、实现可持续发展作为重点，迄今已开展34个涉非项目，成为中国、非洲、联合国三方合作的新平台。

The “China-United Nations Peace and Development Fund” announced by President Xi Jinping in September 2015，focuses on supporting African peacekeeping, counter-terrorism, and achieving sustainable development， So far, 34 projects involving Africa have been carried out， becoming a tripartite cooperation between China, Africa and the United Nations.

Figure 1: An example taken from the WMT2020 test set that shows a single source Chinese segment is translated into two separate English sentences. The highlighted comma separates the two corresponding complete sentences in the Chinese text, whereas the other two commas are sentence-internal boundaries.

Being able to directly accommodate long, multi-sentence inputs has clear appeal. However, in practice, the training data available for translation models is dominated by the relatively short (sub)sentence pairs that are preferentially recovered by standard approaches to sentence alignment (Tiedemann, 2011; Gale and Church, 1993), and as a result of the natural distribution of sentence lengths. Unfortunately, generalisation from training on short sequences to testing on long sequences continues to be an unsolved problem even in otherwise well-performing translation models (Lake and Baroni, 2018; Koehn and Knowles, 2017). Rather than addressing the length generalisation problem directly, in this paper we side-step it by learning to make decisions about segmentation so as to maximise the performance of an unreliable machine translation system that operates optimally only on shorter segments of input Chinese text.

While numerous text segmentation techniques designed to improve machine translation have been proposed over the years (§5), these have typically

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 293–302
August 1–6, 2021.

been based on heuristics that capture linguistic or statistical notions about what "minimal translatable units" consist of. In Chinese, robustly identifying such units is particularly challenging, on account of the lack of overt tense and frequent argument dropping which deprive an annotator of important clues (Huang, 1984). In contrast, we formalise the segmentation problem as a series of classification decisions about whether or not to split at candidate segmentation boundaries and use reinforcement learning to train the segmentation policy to optimise the aggregate BLEU score that results from translating the resulting segments with a *particular* translation system. Our approach is therefore robust to the idiosyncrasies of the underlying translation system, it is capable of discovering a policy that deviates from perhaps unreliable intuitions about minimal translatable units, and it can easily be retrained as the translation system improves.

Experiments indicate that the proposed approach outperforms a baseline that carries out no sentence splitting other than at unambiguous points, a classification approach based on linguistic criteria, and a heuristic system used in prior work. Overall, we improve the BLEU score on the WMT2020 Chinese–English news translation task by 0.3 BLEU, but for segments consisting of more than 60 words, the BLEU score increases by 3 BLEU.

2 Problem setup

We setup segmentation problem as a Markov decision problem (MDP) whose state, actions and rewards are characterized as follows. Every example in the training dataset is treated as a new episode and the objective is to maximise the sentence level BLEU score of the translation.

- The action set $\mathcal{A} = \{\textsc{split}, \textsc{continue}\}$.
- The state representation at time t, $\mathbf{s}_t = [\boldsymbol{\phi}(x)_{1,\dots,n}]; [\mathbf{p}^{(t)}_{1,\dots,n}]$ where n is the length of the input sequence $\boldsymbol{x}$ in words, $\boldsymbol{\phi}(x_i)$ is a vector encoding of ith token in context, and $\mathbf{p}^{(t)}_i$ is a record of previous decisions taken by the classifier about split decisions. The decision state for each token at timestep t can take on 4 possible discrete values: **no punctuation** (no actions are taken on these), **undecided punctuation** (punctuation on which an action still needs to be taken), **un-split** punctuation and **split** punctuation. At each timestep t, an action is taken at the next immediate undecided punctuation from the left and the state update involves appropriately updating the punctuation marker (to unsplit or split) corresponding to that position. The episode is considered terminal when there are no unattended punctuation markers in the sentence.
- For our reward, we use $r_t = \text{BLEU}(\tau(\mathbf{s}_{t+1}), \boldsymbol{y}^*) - \text{BLEU}(\tau(\mathbf{s}_t), \boldsymbol{y}^*)$, the marginal difference added to BLEU score based on the current action decision similar to Wu et al. (2018). τ represents the translation of the source inputs constructed from the state definition, where we split the sentence according to the punctuation markers in $\mathbf{s}_t$, translate each segment independently and recombine the best translation of each segment. An action of no split yields 0 rewards as segments remain identical and splitting would produce positive or negative rewards depending on the improvement/degradation to the quality of the overall sentence translation. Marginal BLEU sums to the sentence level BLEU obtained on the full sequence of translations, but provides a denser reward signal which makes policy learning more efficient.

Network and Learning Algorithm. We learn optimal segmentation policies using a distributed deep RL algorithm, IMPALA (Espeholt et al., 2018) which provides data-efficient stable learning at high throughput by combining decoupled acting and learning. At each time-step, our policy receives a state $\mathbf{s}_t$ and defines a distribution over the discrete action set $\mathcal{A}$ – i.e. $\pi(s_t) : \mathbb{R}^n \rightarrow \mathcal{A}\{\textsc{split}, \textsc{continue}\}$. Our algorithm also employs a value network that learns the value of a state $V(s_t) : \mathbb{R}^n \rightarrow \mathbb{R}$, which is used to regularize the policy network. In this work, we use a transformer encoder with self-attention layers as an observation encoder and then apply a feed-forward classifier on this encoded observation to learn the policy's action distribution. More details on the network architecture can be found in Appendix A.

3 Experiments

Our experiments are carried out on the WMT2020 Chinese–English news translation task, which is subjected to the same pre-processing steps as described in (Yu et al., 2020). Figure 2 shows the distribution of the maximum number of possible segments that we could split for each given example in our test dataset if a model were to split on every available punctuation (comma and full-stop

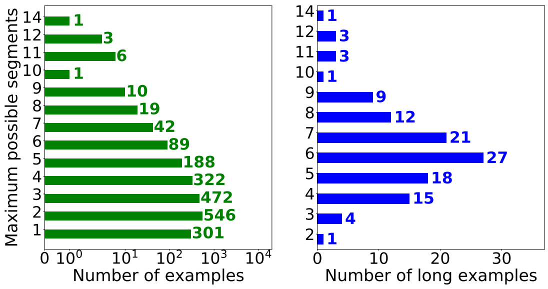

Figure 2: Distribution of maximum possible number of segments (if a model were to split on every available punctuation) for all the examples in the entire WMT Chinese–English test dataset and on long examples (source length ≥60 words) only. Note that the test dataset contains 2000 examples in total and 115 long examples.

in our case). We report case-sensitive BLEU, as computed with `sacrebleu` (Post, 2018). All the model/training details such as datasets, model architectures and hyperparameters pertaining to the baseline models are listed in Appendix B. We compare RLSEGMENT, our proposed solution's performance to six other baselines which help highlight different trade-offs involved in segmentation decisions for machine translation.

- NOSPLIT – Our key baseline that we wish to improve – a strategy of not doing any splits on the source inputs beyond unambiguous full stops.
- ALLSPLIT – An aggressive segmentation policy where we segment the source on every possible comma and full-stop.
- ORACLE – To compute the oracle score, we translate possible splits of a given source sentence and select the one that maximizes the example-level BLEU score. This benchmark is the upper limit of any segmentation policy, given a translation model. It is quite expensive to compute since it requires decoding all possible segmentations of every sequence.
- ORACLESUP – Oracle segmentation decisions from the training corpus are used to setup a supervised binary classification task with an architecture similar to RLSEGMENT's policy network.
- COMMACLASS – Using the syntactic patterns from the Penn Chinese Treebank 6.0, this system builds a comma classifier to disambiguate terminal and non-terminal commas similar to (Xue and Yang, 2011). This uses a transformer encoder followed by a positional feed-forward network to classify every comma.

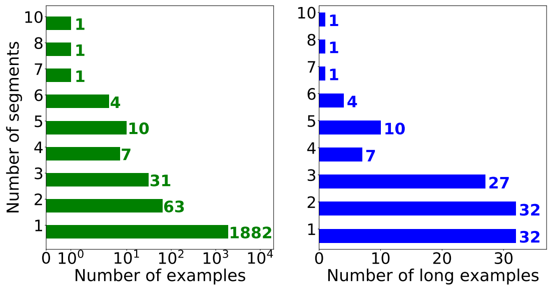

Figure 3: Distribution of number of segments split by our model RLSEGMENT on the entire WMT20 Chinese–English test dataset and on long examples (source length ≥60 words) only. Out of the 115 long examples, the model leaves 32 (28%) of them unsegmented and segments the rest into two or more segments.

Model	full BLEU	full BP	≥ 60 BLEU	≥ 60 BP
NOSPLIT (Baseline) [2000]	31.89	89.42	25.73	78.17
ALLSPLIT [6233]	29.45	93.18	27.88	94.43
ORACLESUP [3107]	31.57	90.89	27.66	88.02
COMMACLASS [2569]	31.82	89.98	26.93	81.16
HEURISTIC [2071]	32.02	89.59	27.05	80.80
RLSEGMENT (Ours) [2228]	**32.21**	**91.34**	**29.03**	**88.09**
ORACLE [3513]	*36.36*	*96.19*	*34.71*	*93.45*

Table 1: BLEU and brevity penalty scores, both on the corpus and long sentences only (source length ≥ 60 words) on the test dataset of WMT20 Chinese–English. Reported in square brackets are the number of segments independently translated under each policy.

- HEURISTIC – The uses a combination of predictions from COMMACLASS together with heuristic length constraints that only split long inputs (> 60 words) on terminal commas suggested by the model and terminal punctuations, and only if the resulting segments are not too short (> 10 words).

As discussed above, there is no standard segmentation of Chinese texts into sentences, and therefore all the "supervised" approaches—including our baselines ORACLESUP and HEURISTIC, construct their own training data to train a classifier. RLSEGMENT on the other hand, requires only a translation system and a corpus of standard parallel data for training and learns without any hand-engineered constraints on the model itself, thus presenting a generic solution that can scale across languages and system variants.

Source (showing segmentation candidate split points and RLSEGMENT policy decision in grey)

中新网深圳10月1日电(郑小红徐殿伟刘姝梦)10月1日为内地国庆“黄金周”首日,也是香港国庆公众假期的最后一天,许多内地居民利用假期出国境游玩、购物,也有许多来深的香港居民当日返港,深圳各口岸出现客流“爆棚”的情况,罗湖、深圳湾口岸新启用22条自助通道,提升口岸通关效率,便利旅客高效快捷通关。

NOSPLIT *Shenzhen, Oct. 1 (Zheng Xiaohong Xu Dianwei Liu Jiameng) Oct. 1 is the first day of the Mainland National Day “Golden Week” and the last day of the Hong Kong National Day Public Holiday. Many mainland residents used the holiday to visit and shop abroad. Many Hong Kong residents who came to Shenzen returned to Hong Kong on the same day. There are passenger flow “burst shed” at all ports.*

RLSEGMENT *Shenzhen, Oct. 1 (Zheng Xiaohong Xu Dianwei Liu Jiameng) Oct. 1 is the first day of the Mainland National Day “Golden Week” and the last day of the Hong Kong National Day Public Holiday. Many mainland residents used their holidays to visit and shop abroad. Many Hong Kong residents who came to Shenzhen returned to Hong Kong on the same day. There are passenger flow “burst shed” at all ports in Shenzhen. 22 new self-service channels opened at Lo Wu and Shenzhen Bay ports to improve the customs clearance efficiency at ports, facilitate efficient and rapid customs clearance of passengers.*

Figure 4: A example translation where segmentation with RLSEGMENT mitigates premature truncation in our system; material dropped by the baseline system is highlighted in grey.

4 Results

Figure 3 shows the distribution of segments proposed by our model RLSEGMENT. We see that a lot of the long sentences (source length ≥ 60 words) are split into two or more independent segments to mitigate premature truncation seen in transformer models with these long sentences. Table 1 compares different segmentation policies when translating the WMT20 Chinese–English news test dataset. While the BLEU scores indicate the quality of translations on the entire corpus, we also report BLEU scores on sources longer than 60 words as a metric to show the performance of these models on longer sentences where standard transformers tend to produce translations that are too short. In both the cases, we also report the brevity penalty (BP), a component of BLEU to show the impact on the overall length of the translation.

We see that our proposed segmentation policy, RLSEGMENT improves both the BLEU scores and brevity penalties as compared to the baseline translation case NOSPLIT. Specifically, the RL model improves BLEU scores on long sentences by 3+ BLEU points and BP on those sentences by about 9+ points. This shows that our model, via smart segmentation, suffers less because of premature truncation of long translations as compared to the baseline—a common problem (Meister et al., 2020; Koehn and Knowles, 2017). While segmentation of long sentences at appropriate punctuations helps performance, segmentation at all punctuations is expected to hurt performance as it is highly likely to produce extremely small segments which lose a lot of necessary source context when individually translated. This is demonstrated by poorer BLEU score of the ALLSPLIT baseline, even though it achieves good BP scores both on the corpus and long translations. Compared to supervised baselines trained on syntactic data such as COMMACLASS and HEURISTIC, our model performs competitively on both BLEU and BP without any supervised data for segmentation or hand-engineered length constraints. In Figure 4, we see an example where RLSEGMENT mitigates premature truncation of the resultant translation. In this example, although input (and resulting English translation) consists of three sentences separated by commas, the segmentation policy has only chosen to split at only *one* position, having learned that the underlying translation system is capable of translating some two-sentence inputs. This example thus illustrates the practicality of learning a segmentation policy based on the abilities of the underlying translation system, not just on the basis of normative notions of translatable units. (More examples can be found in Appendix C.1, C.2)

While our model does better than the baselines, there is a sufficient performance gap to the oracle BLEU scores (because of the different data/length characteristics between training and test time) that could be achieved via “perfect” segmentation, demonstrating the value for further research into better segmentation strategies. However, we also note that the RLSEGMENT outperforms ORACLESUP—especially on long sentences. We suspect this has to do with the relative scarcity of such examples in the training data—while a supervised learner can happily ignore those rare cases at little cost in terms of cross-entropy, they have an out-sized impact on BLEU, and therefore the RL learner is sensitive to them.

Finally, it is important to note that while RLSEG-

MENT improves BLEU at a corpus level, there exist cases where individual translation examples (Appendix C.3) are worse because of inappropriate segmentation.

5 Related Work

The segmentation of long texts and sentences into segments suitable for translation has been a recurring topic in machine translation research (Tien and Minh, 2019; Pouget-Abadie et al., 2014; Goh and Sumita, 2011; Doi and Sumita, 2003); however, we are the first to apply reinforcement learning to solve the problem. A related problem to the segmentation problem occurs in automated simultaneous interpretation, where the system must produce translations as quickly as possible, but it is necessary to wait until sufficient context has been received before an accurate translation can be produced. Grissom II et al. (2014) used an RL approach, targeting a reward that balances translation quality with translation latency.

Chinese comma disambiguation has likewise been studied. However, without exception these have sought to predict normative notions of what constitutes a complete clause or elementary discourse unit (Xu and Li, 2013; Xue and Yang, 2011; Jin et al., 2004), on the basis of syntactic annotations in the Chinese Treebank (Xue et al., 2005). In contrast, our solution is directly targeted at developing a segmentation strategy that results in a good downstream translation, rather than conforming to any single normative notion of what constitutes a complete sentence.

6 Conclusion

In this work, we have addressed a key challenge in Chinese-English machine translation : the ambiguity of English-like sentence boundaries in Chinese, resulting in long Chinese sentence data for machine translation tasks. Our solution casts the Chinese sentence segmentation problem into a sequential decision making problem and then uses Reinforcement Learning to learn an optimal segmentation policy to maximize the BLEU scores of the eventual translation from the independent segment translations. Our solution does not require any paired training data for segmentation and is able to learn an optimal strategy purely from paired machine translation data. Our model is able to outperform a baseline translation strategy that segments on only unambiguous full-stops by 0.3 BLEU at a corpus level and by 3 BLEU on a sub-corpus comprising only source sentences longer than 60 words.

Acknowledgements

We would like to thank Laurent Sartran, Wojciech Stokowiec, Lei Yu and Wang Ling for helpful discussions and guidance during different stages of this work in terms of experimentation, presentation and software development.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Takao Doi and Eiichiro Sumita. 2003. Input sentence splitting and translating. In *Proceedings of the HLT/-NAACL Workshop on Building and Using Parallel Texts*.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proc. ICML*.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1).

Chooi-Ling Goh and Eiichiro Sumita. 2011. Splitting long input sentences for phrase-based statistical machine translation. In *Proceedings of the Association of Natural Language Processing*.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proc. EMNLP*.

C. T. James Huang. 1984. On the distribution and reference of empty pronouns. *Linguistic Inquiry*, 15:531–574.

Mei xun Jin, Mi-Young Kim, Dongil Kim, and Jong-Hyeok Lee. 2004. Segmentation of Chinese long sentences using commas. In *Proceedings of the Third SIGHAN Workshop on Chinese Language Processing*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL*.

Brendan Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proc. ICML*.

Clara Meister, Ryan Cotterell, and Tim Vieira. 2020. If beam search is the answer, what was the question? In *Proc. EMNLP*.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proc. EMNLP*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Jean Pouget-Abadie, Dzmitry Bahdanau, Bart van Merriënboer, Kyunghyun Cho, and Yoshua Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150.

Jörg Tiedemann. 2011. *Bitext Alignment*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Ha Nguyen Tien and Huyen Nguyen Thi Minh. 2019. Long sentence preprocessing in neural machine translation. In *IEEE-RIVF International Conference on Computing and Communication Technologies*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

Shengqin Xu and Peifeng Li. 2013. Recognizing Chinese elementary discourse unit on comma. In *Proceedings of the 2013 International Conference on Asian Language Processing*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2).

Nianwen Xue and Yaqin Yang. 2011. Chinese sentence segmentation as comma classification. In *Proc. ACL*.

Lei Yu, Laurent Sartran, Po-Sen Huang, Wojciech Stokowiec, Domenic Donato, Srivatsan Srinivasan, Alek Andreev, Wang Ling, Sona Mokra, Agustin Dal Lago, Yotam Doron, Susannah Young, Phil Blunsom, and Chris Dyer. 2020. The DeepMind Chinese–English document translation system at WMT2020. In *Proceedings of the Fifth Conference on Machine Translation*, pages 326–337, Online. Association for Computational Linguistics.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proc. EMNLP*.

A Data and Model details for RLSEGMENT

A.1 Dataset

We use the training, validation and test datasets from the WMT2020 Chinese-English constrained data provided to shared task participants. All the examples are aligned paired translations with sentence-like units provided as part of the dataset. Pre-processing of text is done in exactly the same methodology as described in Section 3 of (Yu et al., 2020). We also use the same `sentencepiece` tokenizer described in Section 3 of that work to convert the text into integer tokens. In total, we operate on a dataset with 18 million training examples, 1619 validation examples and 2000 test examples.

A.2 Translation Model

For all experiments and baselines, we use the same transformer model to translate any source input (Vaswani et al., 2017). The model used has 6 encoder layers and 2 decoder layers with 8 encoder and decoder attention heads each. The model uses a feed-forward layer of size 2048 and employs Multi-Query Attention layers (Shazeer, 2019) for faster inference. It uses different dropout values for different components: 0.1 for the attention, 0.05 in the feed-forward network, and finally 0.3 after the sub-layer. The other hyperparameters and learning rate schedules of the model are similar to those described in Section 4.1 of (Yu et al., 2020).

A.3 RL Models

The policy and value networks share common encoding layers, with different feed-forward networks on top of the shared encoder layer. The shared encoder consists of 2 stacked self-attention layers with 8 attention heads each and feed-forwards of size 512, with 0.1 as the rate for all attention, feed-forward and sub-layer dropouts. Sequence lengths for training were restricted to 280 tokens (maximum sentence length in the validation set is 277). The policy network applies a feed-forward network of sizes 256-2 on the outputs of the shared encoder and value network applies a feed-forward network of sizes 256-1. Adam optimizer with learning rate=0.0002, b1=0.0 and b2=0.99 was used for training. For the IMPALA style loss function (Espeholt et al., 2018), the weights of policy, baseline and entropy loss are set as 1.0, 0.5 and 0.0005 respectively. Key hyperparameters such as the weights of losses, learning rates and model sizes were tuned (a single trial for each hyperparameter configuration) using BLEU scores on the WMT20 Chinese–English validation dataset.

Bounds for hyperparameters - Number of attention layers [2,8], Number of attention heads [2,8], Size of feed-forward network [128, 2048], Learning rates [0.01, 0.00001], Weights of losses [0.0001, 1].

Compute and other details - For both inference and learning, we use 2x2 slices (4 TPUs) of Google TPU v2 with 2 cores per device (8 cores were split into 2 cores used for inference and 6 for learning). 512 actors on CPU were run in parallel to generate transition data for the distributed learner to learn from. Learning was done on 70 million episodes with a batch size of 256 per core and the entire experiment had an average run-time of approximately 6 hours.

B Data and Model details for BASELINES

All baseline models use the same inference model as described in Appendix A.2. All baselines are evaluated on the WMT20 Chinese–English test data as described in Appendix A.1. The baselines NOSPLIT, ALLSPLIT and ORACLE do not require any training data and are directly employed during test time.

B.1 COMMACLASS and HEURISTIC

Dataset - The comma classifier used in these baselines is trained on Chinese Treebank data prepared in the same format as described in (Xue and Yang, 2011).

Model - Both these baselines rely on a comma classifier model. In our experiments, this model uses a standard Transformer style encoder with a feed-forward classifier for Chinese comma disambiguation. It uses the encoder layers from a pre-trained transformer (which was trained on WMT20 Chinese–English translation task) since the Chinese Treebank dataset alone is small for training large models. The encoder of this model has 16 heads, 9 layers and a feed-forward size of 4096. On top of this encoder, a feed-forward classifier of size 512-128-2 is used for classification. The pre-trained encoders are further fine-tuned during classification. We use Adam Optimizer with a custom learning rate schedule similar to (Vaswani et al., 2017).

Bounds for hyperparameters - Number of attention layers [2,16], Number of attention heads [2,16], Size of feed-forward network [128, 4096], Learning rates [0.01, 0.00001].

B.2 ORACLESUP

Dataset - The ORACLESUP baseline uses WMT2020 Chinese-English data described in Appendix A.1. From this dataset, supervised labels were generated by splitting an example into its all possible splits on punctuations and then choosing the split set which optimized the BLEU score at an individual example level.

Model - This model uses the same network as the policy network described in Appendix A.3, but with 6 attention layers and 8 attention heads each. The model uses a learning rate schedule and Adam optimizer, similar to (Vaswani et al., 2017). The classes (No split:split - 1:2.2) were assigned different weights in the loss function to account for class imbalance in the data.

Bounds for hyperparameters - Number of attention layers [2,8], Number of attention heads [2,8], Size of feed-forward network [128, 2048], Learning rates [0.01, 0.00001], Weight for the split class [1,5].

C Example Model Outputs

The splits in the source sentence have been highlighted.

C.1 RLSEGMENT addresses truncation in long source sentences.

Source

中国人民银行授权中国外汇交易中心公布, 2018年1月18日银行间外汇市场人民币汇率中间价为: 1美元对人民币6.4401元, 1欧元对人民币7.8482元, 100日元对人民币5.7854元, 1港元对人民币0.82399元, 1英镑对人民币8.9017元, 1澳大利亚元对人民币5.1302元, 1新西兰元对人民币4.6779元, 1新加坡元对人民币4.8546元, 1瑞士法郎对人民币6.6731元, 1加拿大元对人民币5.1747元, 人民币1元对0.61399马来西亚林吉特, 人民币1元对8.8352俄罗斯卢布, 人民币1元对1.9128南非兰特, 人民币1元对165.95韩元, 人民币1元对0.57030阿联酋迪拉姆, 人民币1元对0.58233沙特里亚尔, 人民币1元对39.3106匈牙利福林, 人民币1元对0.53093波兰兹罗提, 人民币1元对0.9487丹麦克朗, 人民币1元对1.2501瑞典克朗, 人民币1元对1.2240挪威克朗,

Ground Truth

the People 's Bank of China has authorized the China Foreign Exchange Trade System to release the interbank foreign exchange market central parity rates of the RMB : USD 1 to RMB 6.4401 , EUR1 to RMB 7.8482 , JPY 100 to RMB 5.7854 , HKD1 to RMB 0.82399 , GBP 1 to RMB 8.9017 , AUD1 to RMB 5.1302 , NZD1 to RMB 4.6779 , SGD 1 to RMB 4.8546 , CHF1 to RMB 6.6731 , CAD1 to RMB 5.1747 , RMB 1 to MYR 0.61399 , RMB 1 to RUB8.8352 , RMB 1 to ZAR1.9128 , RMB 1 to KRW 165.95 , RMB 1 to AED0.57030 , RMB 1 to SAR0.58233 , RMB 1 to HUF39.3106 , RMB 1 to PLN0.53093 , RMB 1 to DKK0.9487 , RMB 1 to SEK1.2501 , RMB 1 to NOK1.2240 , RMB 1 to TRY0.59303 , RMB 1 to MXN2.9071 .

NOSPLIT Translation

the central parity rate of RMB in the interbank foreign exchange market was : US$ 1 to RMB 6.4401 , 1 euro to RMB 7.8482 , 100 yen to RMB 5.7854 , 1 Hong Kong dollars to RMB 0.82399 , 1 British pound to RMB 8.9017 , 1 Australian dollar to RMB 5.1302 , 1 NZ dollar versus RMB 4.6779 ,

RLSEGMENT Translation

the People 's Bank of China authorized the China Foreign Exchange Trading Center to announce that the median exchange rate of RMB in the interbank foreign exchange market on January 18 , 2018 was : US$ 1 to RMB 6.4401 , 1 euro to RMB 7.8482 , 100 yen to RMB 5.7854 , 1 Hong Kong dollar to RMB 0.82399 , 1 pound to RMB 8.9017 , 1 Australian dollar to RMB 5.1302 , 1 NZ dollar versus RMB 4.6779 , 1 Singapore dollar to RMB 4.8546 and 1 Swiss franc to RMB 6.6731 , 1 Canadian dollar to RMB 5.1747 , 1 yuan to 0.61399 Malaysian ringgit , 1 yuan to 8.8352 Russian rubles , RMB 1 to 1.9128 South African rand and RMB 1 to 165.95 won , RMB 1 to 0.57030 UAE dirham , RMB 1 to 0.58233 Saudi SAR , RMB 1 to 39.3106 forint , RMB 1 to 0.53093 Polish zloty , RMB 1 to DKr 0.9487, RMB 1 to 0.59303 Turkish lire , RMB 1 to 2.9071 Mexican pesos.

Source

根据广东省委省政府部署, 广东广电网络积极实施新数字家庭4K 行动计划, 2017 年12 月发布了全国首个支持IPv6 、全国广电首款支持4 路真4K 并发的新数字家庭网关终端产品- - - “ U 点家庭服务器” (搭配AVS2 标准的4K 盒子、讯飞语音遥控器) , 加快建设以4K 超高清视频、下一代互联网、有线无线融合网络应用为标志的新数字家庭示范区, 积极参与广东省4K 电视网络应用示范区建设。

Ground Truth

according to the deployment of the Guangdong Provincial Party Committee and the provincial government , Guangdong Cable Network actively implemented the new digital home 4K action plan . in December 2017 , it released the first new digital home gateway terminal product in China that supports IPv6 and the national radio and television industry supporting the 4 - way real 4K concurrency . – “ U - Point Home Server ” (with AVS2 standard 4K box , iFlytek voice remote control) to accelerate the construction of a new digital home demonstration area marked by 4K Ultra HD video , next - generation Internet , wired and wireless integrated network applications , and actively participate in the construction of the 4K TV networks application demonstration area in Guangdong Province .

NoSplit Translation

according to the deployment of the provincial government of Guangdong Province , Guangdong Radio and Television Network actively implemented the new digital family 4K action plan . in December 2017 , it released the first new digital family gateway terminal product in the country to support IPv6 and the first national radio and television support 4zhen 4K concurrence - “ U - point family server ” (with AVS2 standard 4K box and message flying voice remote control) to accelerate the construction of

RLSegment Translation

according to the deployment of the provincial government of Guangdong Province , the Guangdong Radio and Television Network actively implemented the new digital family 4K action plan . in December 2017 , it released the first new digital family gateway terminal product in the country to support IPv6 and the first national radio and television to support 4 Zhen 4K concurrently - “ U - point family server ” (with AVS2 standard 4K box and message flying voice remote control) . it accelerate the construction of a new digital family demonstration area marked by 4K ultra - high - clearing video , next - generation Internet, wired and wireless fusion network applications, actively participate in the construction of 4K TV network application demonstration zone in Guangdong Province .

C.2 RLSegment does not split on all punctuations.

Source

此前, 为防范台风“ 潭美” , 截至28 日11 时统计, 福建已撤离268 艘渔船, 转移渔

排上老弱妇幼人员2282 人。

RLSegment Translation

previously , in order to prevent Typhoon “ Tanmei ” , as of 11:00 on the 28th , Fujian had evacuated 268 fishing boats and transferred 2,282 elderly , weak women and children

Source

除上述便民措施外, 卫生局辖下各卫生中心及卫生站自推出计划后, 已延长周一至五的流感疫苗接种服务时间至晚上8 时, 截至9 月30 日下午4 时, 已为4,158 人接种, 与去年同期的3,714 人多。

RLSegment Translation

in addition to the above convenience measures , health centers and health stations under the Health Bureau have extended the influenza vaccination service from Monday to Friday to 8 pm . as of 4 pm on September 30 , 4,158 people had been vaccinated , up from 3,714 in the same period last year .

C.3 Bad segmentation decisions produced by RLSegment

Source

虽然海风剧烈, 但法国ABM 队的舵手皮埃尔· 伊维斯· 杜兰德在赛前见面会上诉说了自己与比赛举办地青岛的温情往事: 5 年前的国际极限帆船系列赛自己意外受伤, 肌肉断裂, 多亏当地赛事医疗团队的处理才让自己还能在5 年后的今天站在比赛场地。

Ground Truth

though the sea wind blew violently , the helmsman of French ABM Team Pierre - Yves Durand told his warm memories with Qingdao at the pre - game meeting that he was accidentally injured in the Extreme Sailing Series 5 years ago , resulting in the muscular rupture , and fortunately , the local race medical team helped him deal with it so that he could stand here to take part in the race after 5 years .

RLSegment Translation

despite the intense sea breeze , Pierre Ives Durand , the helmsman of the French ABM team , told his warm past with Qingdao , the venue of the competition, that he was injured accidentally in the International Extreme Sailing Series five years ago . caused muscle fracture , thanks to the handling of the local event medical team , he can still stand on the venue today five years later .

Source

“ 喀山” 号、“ 新西伯利亚” 号、“ 克拉斯诺亚尔斯克” 号、“ 阿尔汉格尔斯克” 号多用途潜艇均为“ 白蜡树- M ” 改进型, 排水量为13800 吨, 下潜深度520 米, 人员编制64 人, 水下航速31 节, 将携带水雷、鱼雷和“ 口径” 及“ 缟玛瑙” 巡航导弹。

Ground Truth

the multipurpose submarines of “ Kazan ” , “ Novosibirsk ” , “ Krasnoyarsk ” and “ Arkhangelsk ” are all of “ Fraxinus - M ” improved type , with displacement of 13,800 tons , depth of 520 m , staffing of 64 , submerged speed of 31 knots . all the submarines will carry mines , torpedoes and “ Kalibr ” and “ Onyx ” cruise missiles .

RLSegment Translation

Kazan , Nova Scotia , Krasnoyarsk and Arkhangelsk are all improved models of White wax tree - M , with a displacement of 13,800 tons and a submersible depth of 520 meters . staffing 64 , underwater speed of 31 knots. They will carry mines , torpedoes and “ caliber ” and “ agate ” cruise missiles .

Enhancing Transformers with Gradient Boosted Decision Trees for NLI Fine-Tuning

Benjamin Minixhofer[1]*, **Milan Gritta**[2] and **Ignacio Iacobacci**[2]
[1]Johannes Kepler University, Linz, Austria
[2]Huawei Noah's Ark Lab, London, UK
`bminixhofer@gmail.com`
`{milan.gritta, ignacio.iacobacci}@huawei.com`

Abstract

Transfer learning has become the dominant paradigm for many natural language processing tasks. In addition to models being pretrained on large datasets, they can be further trained on intermediate (supervised) tasks that are similar to the target task. For small Natural Language Inference (NLI) datasets, language modelling is typically followed by pretraining on a large (labelled) NLI dataset before fine-tuning with each NLI subtask. In this work, we explore Gradient Boosted Decision Trees (GBDTs) as an alternative to the commonly used Multi-Layer Perceptron (MLP) classification head. GBDTs have desirable properties such as good performance on dense, numerical features and are effective where the ratio of the number of samples w.r.t the number of features is low. We then introduce FreeGBDT, a method of fitting a GBDT head on the features computed during fine-tuning to increase performance without additional computation by the neural network. We demonstrate the effectiveness of our method on several NLI datasets using a strong baseline model (RoBERTa-large with MNLI pretraining). The FreeGBDT shows a consistent improvement over the MLP classification head.

1 Introduction

Recent breakthroughs in transfer learning ranging from semi-supervised sequence learning (Dai and Le, 2015) to ULMFiT (Howard and Ruder, 2018), ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have brought significant improvements to many natural language processing (NLP) tasks. Transfer learning involves pretraining neural networks, often based on the Transformer (Vaswani et al., 2017), on large amounts of text in a self-supervised manner in order to learn transferable language features useful for many NLP tasks. Pretraining is followed by fine-tuning the model on the target task. Pretrained models can also be further trained on intermediate labelled datasets which are similar to the target task before the final fine-tuning stage (Pruksachatkun et al., 2020). We refer to this as *intermediate supervised pretraining*. In this manner, the network learns more meaningful internal representations of the input text that are better aligned with the target task. In order to fine-tune the pretrained network, some latent representation of the text (e.g. the hidden state corresponding to the `[CLS]` token of BERT-like models) is used as the input to a *classification head*, usually a randomly initialised Multi-Layer Perceptron (MLP) (Wolf et al., 2019). The output of the classification head can then be interpreted as probability distribution over classes. The input to the classification head is referred to as *features* throughout the paper. It is a high-dimensional vector that serves as a rich, distributed representation of the input text.

We investigate whether replacing the commonly used MLP classification head with a GBDT (Friedman, 2001) can provide a consistent improvement, using NLI tasks as our use case. GBDTs are known for strong performance on dense, numerical features (Ke et al., 2019), which includes the hidden states in a neural network. The number of input features p, i.e. the dimension of the hidden state corresponding to the `[CLS]` token is not necessarily much larger than the number of samples n (it may even be smaller). GBDTs have proven effective for tasks where $n < p$ (Kong and Yu, 2018) and can be more effective compared to logistic regression if $n \ngtr p$ (Couronné et al., 2018). Therefore, for a language model that was trained on an intermediate supervised task before fine-tuning, we hypothesise that a GBDT may be able to outperform an MLP

---

[*] Work conducted as Research Intern at Huawei Noah's Ark Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 303–313
August 1–6, 2021. ©2021 Association for Computational Linguistics

classification head as the hidden states already encode information relevant to the target task at the start of fine-tuning. The head must learn to exploit this information exclusively during the fine-tuning stage in which the training data may consist of only a few samples. Our contributions are as follows:

- We integrate the GBDT into a near state-of-the-art (SOTA) language model as an alternative to an MLP classification head and train on the features extracted from the model *after* fine-tuning. We refer to it as standard GBDT.
- We introduce a method to train a GBDT on the features computed *during* fine-tuning, at no extra computational cost by the neural network, showing a consistent improvement over the baseline. We refer to it as **FreeGBDT**.

In the following, we recap different approaches to integrating tree-based methods with neural networks (Section 2). We introduce our FreeGBDT method in Section 3. We present our experimental setup in Section 4 and results on standard NLI benchmarks in Section 5. To conclude, we discuss improvements and limitations of our method in Section 6.

We release our code[1], implemented with LightGBM (Ke et al., 2017) and Huggingface's Transformers (Wolf et al., 2019), to the NLP community.

2 Related Work

Recent work on transfer learning in NLP has often been based on pretrained transformers, e.g. BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), T5 (Raffel et al., 2020) and RoBERTa (Liu et al., 2019). These models are pretrained on large datasets using self-supervised learning, typically a variation of language modelling such as Masked Language Modelling (MLM). MLM consists of masking some tokens as in the Cloze task (Taylor, 1953). The objective of the model is to predict the masked tokens. Recently, approaches using alternatives to MLM such as Electra (Clark et al., 2020) and Marge (Lewis et al., 2020) have also been proposed. Pretraining transformers on large datasets aims to acquire the semantic and syntactic properties of language, which can then be used in downstream tasks. The models can additionally be trained in a supervised manner on larger datasets before being fine-tuned on the target task.

Natural Language Inference is one of the most canonical tasks in Natural Language Understanding (NLU) (Nie et al., 2020; Bowman et al., 2015). NLI focuses on measuring *commonsense reasoning* ability (Davis and Marcus, 2015) and can be seen as a proxy task that estimates the amount of transferred knowledge from the self-supervised phase of training. The task involves providing a *premise* (also called *context*) and a *hypothesis* that a model has to classify as:

- *Entailment*. Given the context, the hypothesis is correct.
- *Contradiction*. Given the context, the hypothesis is incorrect.
- *Neutral*. The context neither confirms nor disconfirms the hypothesis.

The task can also be formulated as binary classification between *entailment* and *not_entailment* (*contradiction* or *neutral*). We focus on NLI as a challenging and broadly applicable NLP task, with multiple smaller evaluation datasets being available as well as the large Multi-Genre Natural Language Inference corpus (Williams et al., 2018, MNLI), which is often used for effective intermediate pretraining (Liu et al., 2019). As such, it provides a testing ground for the GBDT classification head with intermediate supervised pretraining.

Tree-based methods Models based on decision trees have a long history of applications to various machine learning problems (Breiman et al., 1984). Ensembling multiple decision trees via bagging (Breiman, 1996) or boosting (Freund et al., 1999) further improves their effectiveness and remains a popular method for modelling dense numerical data (Feng et al., 2018). Ensemble methods such as Random Forests (Breiman, 2001) and GBDTs combine predictions from many weak learners, which can result in a more expressive model compared to an MLP. There have been several approaches to combining neural networks with tree-based models, approximately divided into two groups.

1. **Heterogeneous ensembling**: The tree-based model and the neural network are trained independently, then combined via ensembling techniques. *Ensembling* refers to any method to combine the predictions of multiple models such as *stacking* (Wolpert, 1992) or an arithmetic mean of the base models' predictions.

---

[1] Code will be available at `https://github.com/huawei-noah/free-gbdt`.

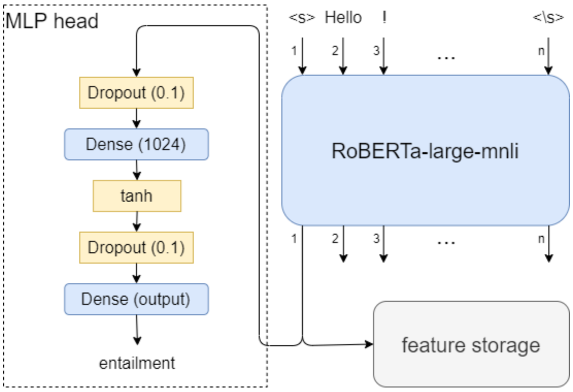

Figure 1: The baseline model architecture. Feature storage is populated *during* fine-tuning for the FreeGBDT but *after* fine-tuning for the standard GBDT method.

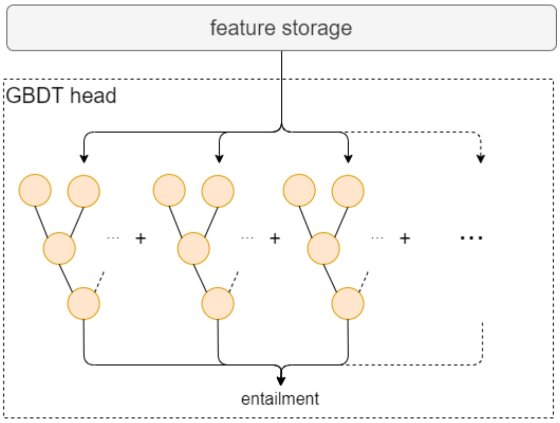

Figure 2: The GBDT classification head.

2. **Direct integration**: The tree-based model is jointly optimised with the neural network.

Heterogeneous ensembling (Li et al., 2019) has proven effective for many applications such as Online Prediction (Ke et al., 2019), Learning-to-Rank for Personal Search (Lazri and Ameur, 2018) and Credit Scoring (Xia et al., 2018). It is also suitable for multimodal inputs, e.g. text, images and/or sparse categorical features as some input types are better exploited by a neural network while others are amenable to tree-based models (Ke et al., 2019).

Direct integration makes the tree-based model compatible with back-propagation thus trainable with the neural network in an end-to-end manner. Examples include the Tree Ensemble Layer (Hazimeh et al., 2020), Deep Neural Decision Forests (Kontschieder et al., 2015) and Deep Neural Decision Trees (Yang et al., 2018). Deep Forests (Zhou and Feng, 2017) are also related although they aim to create deep non-differentiable models instead. Other examples include driving neural network fine-tuning through input perturbation (Bruch et al., 2020), which focuses specifically on using a tree ensemble to fine-tune the neural network representations.

As pretrained transformer-based models have recently achieved strong performance on various NLP tasks (Devlin et al., 2019), we see an opportunity to take advantage of their distributed representations by the means of using a tree-based model as the classification head. Our methods differ from direct integration in that they are not end-to-end differentiable. The training procedure is a sequence, i.e. the transformer-based model is fine-tuned first, then a GBDT is trained with features extracted from the model. We do not interfere with the model updates during training. Finally, the GDBT replaces the MLP classification head. Our approach is invariant to the method with which the neural network is fine-tuned as long as there exists a forward pass in which the features are computed. Recent methods for neural network fine-tuning include FreeLB (Zhu et al., 2020) and SMART (Jiang et al., 2020). FreeLB is an adversarial method, which perturbs the input during training via gradient ascent steps to improve robustness. SMART constrains the model updates during fine-tuning with smoothness-inducing regularisation in order to reduce overfitting. These approaches could theoretically be combined with both the standard GBDT and the FreeGBDT.

3 Methodology

We introduce the standard GBDT (Algorithm 1) and the FreeGBDT (Algorithm 2), our new method of using features generated during fine-tuning. *Features* refers to the hidden state corresponding to the `[CLS]` token of BERT-like pretrained models. We use these features as training data for the GBDT and FreeGBDT.

3.1 The standard GBDT classification head

In order to train the standard GBDT, we apply the feature extraction procedure shown in Algorithm 1. Using the fine-tuned neural network, we perform one additional forward pass over each sample in the training data. We store the features as training data for the GBDT, denoted 'feature storage' in Figures 1 and 2. The GBDT can be then used as a substitute for the MLP classification head.

Algorithm 1 Standard GBDT training procedure. Features are extracted after fine-tuning.

```
Require: training data $X$, pretrained network $f_p$ parametrised by $\theta_p$, classification head $f_h$ parametrised by $\theta_h$.
  for $epoch = 1..N_{epochs}$ do
    for minibatch $(X_b, y_b) \subset X$ do
      $cls \leftarrow f_p(X_b, \theta_p)$
      $y_{pred} \leftarrow f_h(cls, \theta_h)$
      $loss \leftarrow LossFn(y_{pred}, y_b)$
      update $\theta_h$ and $\theta_p$ via backpropagation of $loss$
    end for
  end for

  $features \leftarrow$ empty list
  $labels \leftarrow$ empty list

  for minibatch $(X_b, y_b) \subset X$ do
    $cls \leftarrow f_p(X_b, \theta_p)$
    extend $features$ with $cls$
    extend $labels$ with $y_b$
  end for

  $gbdt \leftarrow train_{gbdt}(features, labels)$
```

3.2 The FreeGBDT classification head

Instead of extracting features once after fine-tuning, the training data for the proposed FreeGBDT is obtained *during* fine-tuning. The features computed in every forward pass of the neural network are stored as training data, shown in Algorithm 2. As no additional computation by the neural network is required, this new classification head is called *FreeGBDT*. Accumulating features in this manner allows the FreeGBDT to be trained on $N \times E$ samples while the standard GBDT is trained on N samples where N is the size of the dataset and E is the number of fine-tuning epochs.

Corpus	Train	Dev	Test	Classes
ANLI	162k	3.2k	3.2k	3
CNLI	6.6k	800	1.6k	3
RTE	2.5k	278	3k	2
CB	250	57	250	3
QNLI	104k	5.4k	5.4k	2

Table 1: NLI evaluation datasets. The tasks with 3 classes contain labels: *entailment*, *neutral* and *contradiction*, tasks with 2: *entailment* and *not_entailment*.

Algorithm 2 FreeGBDT training procedure. Features are accumulated throughout fine-tuning.

```
Require: training data $X$, pretrained network $f_p$ parametrised by $\theta_p$, classification head $f_h$ parametrised by $\theta_h$.
  $features \leftarrow$ empty list
  $labels \leftarrow$ empty list

  for $epoch = 1..N_{epochs}$ do
    for minibatch $(X_b, y_b) \subset X$ do
      $cls \leftarrow f_p(X_b, \theta_p)$
      extend $features$ with $cls$
      extend $labels$ with $y_b$
      $y_{pred} \leftarrow f_h(cls, \theta_h)$
      $loss \leftarrow LossFn(y_{pred}, y_b)$
      update $\theta_h$ and $\theta_p$ via backpropagation of $loss$
    end for
  end for

  $gbdt \leftarrow train_{gbdt}(features, labels)$
```

4 Experimental Setup

We now describe the featured models, details of training procedures and evaluation datasets.

4.1 Datasets

We evaluate our methods on the following NLI datasets, summarised in Table 1.

- Adversarial NLI (ANLI) (Nie et al., 2020). This corpus consists of three rounds of data collection. In each round, annotators try to break a model trained on data from previous rounds. We use the concatenation of R1, R2 and R3.
- Counterfactual NLI (CNLI) (Kaushik et al., 2020). The CNLI corpus consists of counterfactually-revised samples of SNLI (Bowman et al., 2015). We use the full dataset i.e. samples with the revised premise and with the revised hypothesis.
- Recognising Textual Entailment (RTE) (Wang et al., 2019b). We use the data and format as used in the GLUE benchmark: a concatenation of RTE1 (Dagan et al., 2006), RTE2 (Bar Haim et al., 2006), RTE3 (Giampiccolo et al., 2007) and RTE5 (Bentivogli et al., 2009), recast as a binary classification task between *entailment* and *not_entailment*.

Method	CB	RTE	CNLI	ANLI	QNLI
MLP head	93.57 (2.2)	89.51 (0.8)	82.49 (0.8)	57.56 (0.5)	**94.32 (0.1)**
standard GBDT	94.11 (1.7)	89.33 (0.8)	80.84 (0.9)	57.22 (0.5)	94.29 (0.1)
FreeGBDT	**94.20 (1.7)**	**89.69 (0.8)**	**82.53 (0.7)**	**57.63 (0.5)**	94.30 (0.1)

Table 2: Mean Accuracy (Standard Deviation) on the development sets from 20 runs with different random seeds. A Wilcoxon signed-rank test conducted across all five datasets confirms significance with $p \approx 0.01$ c.f. Section 5.

- CommitmentBank (CB) (de Marneffe et al., 2019). We use the subset of the data as used in SuperGLUE (Wang et al., 2019a).
- Question-answering NLI (QNLI) (Demszky et al., 2018). This is a converted version of the Stanford Q&A Dataset (Rajpurkar et al., 2016), aiming to determine whether a given context contains the answer to a question.

4.2 Model and Training

We start all experiments from the RoBERTa-large model (Liu et al., 2019) with intermediate pretraining on the Multi-Genre Natural Language Inference (MNLI) corpus (Williams et al., 2018). The MNLI checkpoint is provided by the fairseq[2] library (Ott et al., 2019). Note that no task-specific tuning of hyperparameters was performed. Instead, we use one learning rate cycle (Smith, 2017) with a maximum learning rate of 1×10^{-5} for each task to fine-tune RoBERTa for 10 epochs with a batch size of 32. We use the Adam optimiser (Kingma and Ba, 2015) to optimise the network. In order to compare the FreeGBDTs with standard GBDTs, we apply Algorithm 1 and Algorithm 2 during the same fine-tuning session to eliminate randomness from different model initialisations.

We use LightGBM[3] to train the GBDT. We do not manually shuffle the data before training. The individual trees of a GBDT are learned in a sequence where each tree is fit on the residuals of the previous trees. One important parameter of the GBDT is thus the number of trees to fit. This is commonly referred to as *boosting rounds*.

We observe that the optimal number of boosting rounds varies significantly across tasks, with a tendency towards more boosting rounds for larger datasets. Thus, we select the number of boosting rounds from the set $\{1, 10, 20, 30, 40\}$ for each task. This is the only task-specific hyperparameter in our experiments. The hyperparameters are identical for the standard GBDT and the FreeGBDT, shown in Table 3, the model shown in Figure 2. The time it takes to train the standard GBDT and the FreeGBDT is negligible compared to the time it takes to fine-tune the RoBERTa model.

Parameter	CB	RTE	CNLI	ANLI	QNLI
learning rate	0.1	0.1	0.1	0.1	0.1
max. leaves	256	256	256	256	256
boosting rounds	10	10	10	40	30

Table 3: Hyperparameters of the standard GBDT and the FreeGBDT. All other hyperparameters are set to the default value as per LightGBM version 2.3.1.

4.3 Evaluation

Development set We evaluate our methods using accuracy. Each experiment is repeated 20 times with different random seeds. We report the mean and standard deviation.

Test set For each task, we train the GBDT with the following boosting rounds: $\{1, 10, 20, 30, 40\}$. We select the GBDT with the best score on the development set. Test scores are obtained with a submission to the SuperGLUE benchmark[4] for CB and the GLUE benchmark[5] for RTE and QNLI. We calculate the test scores on ANLI and CNLI ourselves as the test labels are publicly available. We report accuracy on the test set for each task except for CB, where we report the mean of F1 Score and Accuracy, same as the SuperGLUE leaderboard.

5 Results and Analysis

We summarise the results on the development sets in Table 2. The FreeGBDT is compared with a standard GBDT and the MLP classification head. The standard GBDT achieves a higher score than the MLP head on 1 out of 5 tasks. The FreeGBDT outperforms the standard GBDT on 5 out of 5 tasks and the MLP head on 4 out of 5

---

[2] `https://github.com/pytorch/fairseq`

[3] `https://lightgbm.readthedocs.io`

[4] `https://super.gluebenchmark.com/`

[5] `https://gluebenchmark.com/`

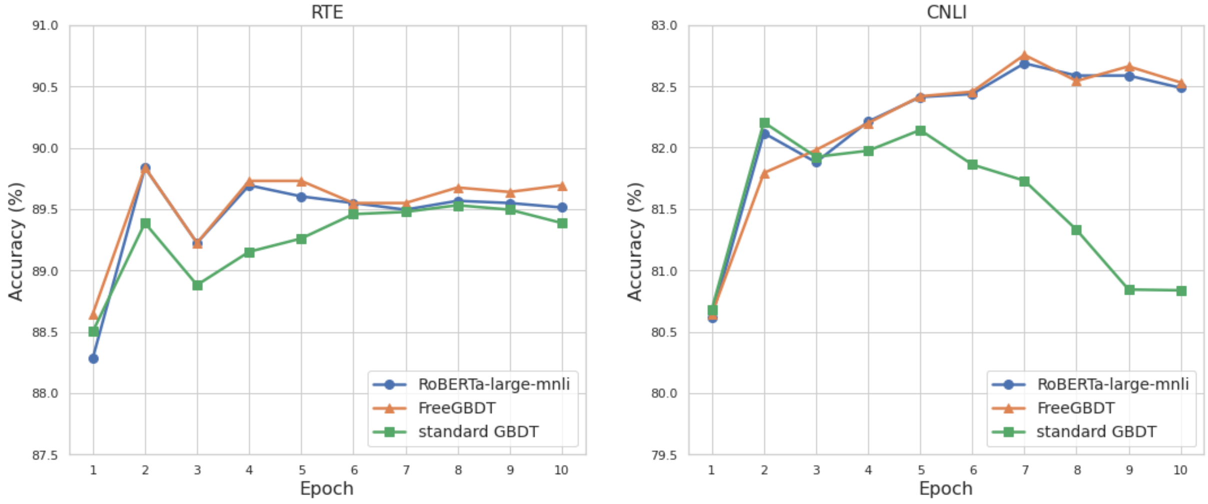

Figure 3: Accuracy on RTE and CNLI development sets. Training is paused after each epoch to compare the GBDT, FreeGBDT and MLP heads. We plot the mean from 20 runs (same hyperparameters but different seeds).

tasks. As recommended for statistical comparison of classifiers across multiple datasets (Demšar, 2006) we conduct a Wilcoxon signed-rank test (Wilcoxon, 1992) with the accuracy differences between the FreeGBDT and the MLP across the 20 seeds and 5 datasets. The test confirms that the improvement from our FreeGBDT method is significant with $p \approx 0.01$.

Method	**CB**	**RTE**	**CNLI**	**ANLI**	**QNLI**
MLP head	92.9	87.5	83.6	57.4	**94.3**
GBDT	91.3	87.5	82.1	57.2	**94.3**
FreeGBDT	**93.3**	**87.8**	**83.7**	**57.6**	**94.3**

Table 4: Results on the test sets.

Results on the test sets are shown in Table 4. The FreeGBDT achieves a small but consistent improvement over the MLP head on each task except QNLI. This task is not a conventional NLI task but a question-answering task converted to an NLI format (Demszky et al., 2018). It has been shown that QNLI does not benefit from MNLI pretraining (Liu et al., 2019) hence this result is not unexpected. Out of the four datasets which do benefit from MNLI pretraining, the FreeGBDT improves over the MLP head on each one with an average score difference of +0.23%. As our experiments start from a competitive baseline, RoBERTa-large with MNLI pretraining, we consider the results important because (a) to the best of our knowledge, this is the first tree-based method that achieves near state-of-the-art performance on benchmark NLI tasks and (b) our method is 'free' as it requires no additional computations by the model. We were able to demonstrate that a FreeGBDT head can be successfully integrated with modern transformers and is a good alternative to the commonly used MLP classification head.

For the Adversarial NLI (ANLI) dataset, we report results which are competitive with the state-of-the-art shown in Table 5, surpassing both SMART (Jiang et al., 2020) and ALUM (Liu et al., 2020). The RoBERTa-large model pretrained with SNLI, MNLI, FEVER (Thorne et al., 2018) and ANLI reported 53.7% accuracy on the ANLI test set (Nie et al., 2020). The state-of-the-art result of 58.3% accuracy on the ANLI dataset was achieved by InfoBERT (Wang et al., 2021). The FreeGBDT achieves a new state-of-the-art on the A2 subset of ANLI with 52.7%. Interestingly, it does not yield an improvement on the easier A1 subset but compares favourably to other recent approaches on the more difficult A2 and A3 subsets of ANLI.

To better understand how performance of the GBDTs evolves during fine-tuning, we carry out an additional experiment. We pause training after each epoch to extract features and train a standard GBDT. We compare it with the MLP classification head and a FreeGBDT trained on the features accumulated up to the current epoch. The result on the RTE and CNLI datasets is shown in Figure 3. Notably, the FreeGBDT does not improve on the standard GBDT after the first epoch where the number of instances the GBDT is trained on is

Method	**A1**	**A2**	**A3**	**All**
RoBERTa-large-mnli *(ours)*	72.0	52.5	49.4	57.4
RoBERTa-large-mnli + GBDT *(ours)*	72.1	52.2	49.0	57.2
SMART (Jiang et al., 2020)	72.4	49.8	**50.3**	57.1
ALUM (Liu et al., 2020)	72.3	52.1	48.4	57.0
InfoBERT (Wang et al., 2021)	**75.0**	50.5	49.8	**58.3**
RoBERTa-large-mnli + FreeGBDT *(ours)*	71.9	**52.7**	49.7	57.6

Table 5: Accuracy across different rounds of the ANLI test set. *All* denotes a sample-weighted average. Our FreeGBDT achieves SOTA on the A2 subset. InfoBERT (Wang et al., 2021) is the SOTA on the full test set.

equal to the size of the training dataset for both. As the FreeGBDT starts accumulating more training data, however, it consistently outperforms the standard GBDT and eventually, the MLP head.

The state-of-the-art in NLI provides some context for our method of combining tree-based models with modern neural networks. RoBERTa (large with MNLI pretraining) reports 89.5% accuracy on the development set of RTE (Liu et al., 2019) and 94.7% accuracy on the development set of QNLI. The same model obtains an F1 Score / Accuracy of $90.5/95.2$ on the CB test set and an accuracy of 88.2% on the RTE test set. Note that ensembles of 5 to 7 models (Liu et al., 2019) were used while our test figures achieve similar scores of $91.3/95.2$ for CB and 87.8% for RTE with a *single* model. These are not direct comparisons, however, the figures demonstrate that FreeGBDT can operate at SOTA levels while matching and exceeding the 'default' MLP head classifier accuracy. Across all datasets, the FreeGBDT improves by an average of 0.2% and 0.5% over the MLP head and the standard GBDT head, respectively.

6 Discussion

The FreeGBDT improves over the MLP head on each task where intermediate supervised pretraining on MNLI is effective. The improvement is significant but not large. This is expected since the input features of the classification head are already a highly abstract representation of the input. Thus, there is limited potential for improvement. However, our results show that a tree-based method is a viable alternative to the commonly used MLP head and can improve over a baseline chosen to be as competitive as possible. Notably, the FreeGBDT improves the MLP baseline on the CB dataset by $> 0.6\%$ solely by switching to our tree-based classification head.

Furthermore, the FreeGBDT outperforms a standard GBDT by a large margin in some cases. For instance, we observe a +1.5% improvement on the CNLI dataset. Figure 3 shows the gap forming towards the end of training. We think this may be due to overfitting to the training data. Recall that the standard GBDT is trained only on features extracted *after* fine-tuning. At this point, the features may exhibit a higher degree of memorisation of the training data. The FreeGBDT is able to mitigate this problem as it was trained with features collected throughout training. Let $f(x, \theta_t)$ denote a mapping from the input text x to the output space parameterised by θ_t where $t \in \{0..T\}$ and T is the total amount of steps the model is fine-tuned for. Then, the standard GBDT is trained on features from $f(x, \theta_T)$, while the FreeGBDT is trained on features from every t in $\{0..T\}$. As such, it may help to think of the FreeGBDT as a type of regularisation through data augmentation (from the FreeGBDT's point of view), having trained on several perturbed views of each training instance.

Figure 4 helps illustrate the regularisation effect by showing the differences between the FreeGBDT and standard GBDT training data beyond just size. The figure shows the temporal changes in a typical feature collected during fine-tuning and the same feature extracted after fine-tuning. We can see that the distribution gradually changes from earlier epochs but remains similar to the distribution of the feature at the end of fine-tuning. FreeGBDT is able to exploit the information at the start of fine-tuning as the features at $t = 0$ already encode information highly relevant to the target task hence all training data is useful. The FreeGBDT head compares favourably to an MLP head, which is randomly initialized at the start of the fine-tuning

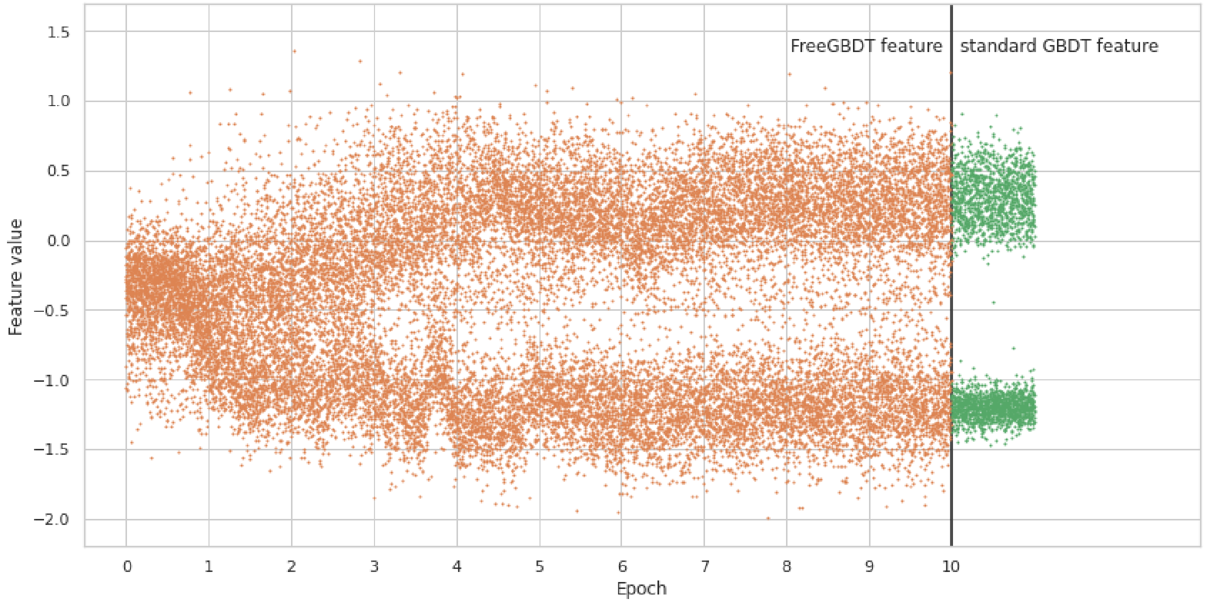

Figure 4: Values of a typical dimension from the 1,024 dimensional vector stored *during* fine-tuning to train a FreeGBDT (left). The same dimension extracted *after* fine-tuning to train a standard GBDT (right).

stage and must thus learn to exploit the latent information from a potentially small amount of training examples. Therefore, we believe intermediate supervised pretraining is essential for the effectiveness of the FreeGBDT, supported by the results from preliminary experiments on the BoolQ dataset (Clark et al., 2019) and QNLI, which does not benefit from pretraining on MNLI (Liu et al., 2019) where FreeGBDT matches the accuracy of the MLP head but does not exceed it. Our experiments also suggest that the potential for improvement from FreeGBDT depends on the size of the training dataset. The gap between FreeGBDT and the MLP head in Table 4 is larger for the smaller datasets CB and RTE and smaller for the larger datasets (ANLI, CNLI, QNLI). This is consistent with prior work showing that GBDTs are especially effective compared to other methods if $n \ggg p$ (Couronné et al., 2018) and hints that the FreeGBDT method might be especially useful for smaller datasets.

7 Future Work

One possible avenue for future work is exploring different features to train the GBDT, e.g. the hidden states from different layers of the pretrained model. This includes new combinations of top layer representations of the Transformer to generate richer input features for the classification head. This could lead to potential improvement by leveraging a less abstract representation of the input. Given that our method operates on distributed representations from a pretrained encoder, applications in other domains such as Computer Vision may be possible, e.g. using features extracted from a ResNet (He et al., 2016) encoder. Furthermore, a GBDT might not be the best choice for each task hence the use of Random Forests (Breiman, 2001) or Support Vector Machines (Cortes and Vapnik, 1995) may also be evaluated to investigate the effectiveness of combining Transformer neural networks with traditional supervised learning methods.

8 Conclusion

State-of-the-art transfer learning methods in NLP are typically based on pretrained transformers and commonly use an MLP classification head to fine-tune the model on the target task. We have explored GBDTs as an alternative classification head due to their strong performance on dense, numerical data and their effectiveness when the ratio of the number of samples w.r.t the number of features is low. We have shown that tree-based models can be successfully integrated with transformer-based neural networks and that the free training data generated during fine-tuning can be leveraged to improve model performance with our proposed FreeGBDT classification head. Obtaining consistent improvements over the MLP head on several NLI tasks confirms that tree-based learners are relevant to state-of-the-art NLP.

References

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees*. CRC press.

Sebastian Bruch, Jan Pfeifer, and Mathieu Guillame-Bert. 2020. Learning representations for axis-aligned decision forests through input perturbation. *CoRR*, abs/2007.14761.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Raphael Couronné, Philipp Probst, and Anne-Laure Boulesteix. 2018. Random forest versus logistic regression: a large-scale benchmark experiment. *BMC bioinformatics*, 19(1):270.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58(9):92–103.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ji Feng, Yang Yu, and Zhi-Hua Zhou. 2018. Multi-layered gradient boosting decision trees. In *Advances in neural information processing systems*, pages 3551–3561.

Yoav Freund, Robert Schapire, and Naoki Abe. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. 2020. The tree ensemble layer: Differentiability meets conditional computation.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with counterfactually augmented data. *International Conference on Learning Representations (ICLR)*.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154.

Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. 2019. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 384–394.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yunchuan Kong and Tianwei Yu. 2018. A deep neural network model using random forest to extract feature representation for gene expression data classification. *Scientific reports*, 8(1):1–9.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. 2015. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475.

Mourad Lazri and Soltane Ameur. 2018. Combination of support vector machine, artificial neural network and random forest for improving the classification of convective and stratiform rain using spectral features of seviri data. *Atmospheric research*, 203:118–129.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. In *Advances in Neural Information Processing Systems*, volume 33, pages 18470–18481. Curran Associates, Inc.

Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining decision trees and neural networks for learning-to-rank in personal search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2032–2040.

Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE.

Wilson L Taylor. 1953. “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Boxin Wang, Shuohang Wang, Yu Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. 2021. Infobert: Improving robustness of language models from an information theoretic perspective. In *International Conference on Learning Representations*.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface’s transformers: State-of-the-art natural language processing.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Yufei Xia, Chuanzhe Liu, Bowen Da, and Fangming Xie. 2018. A novel heterogeneous ensemble credit scoring model based on bstacking approach. *Expert Systems with Applications*, 93:182–199.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. 2018. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Zhi-Hua Zhou and Ji Feng. 2017. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3553–3559.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

Empirical Error Modeling Improves Robustness of Noisy Neural Sequence Labeling

Marcin Namysl[1,2], **Sven Behnke**[1,2], and **Joachim Köhler**[1]

[1] Fraunhofer IAIS, Sankt Augustin, Germany
[2] Autonomous Intelligent Systems, University of Bonn, Germany
`{Marcin.Namysl,Sven.Behnke,Joachim.Koehler}@iais.fraunhofer.de`

Abstract

Despite recent advances, standard sequence labeling systems often fail when processing noisy user-generated text or consuming the output of an Optical Character Recognition (OCR) process. In this paper, we improve the noise-aware training method by proposing an empirical error generation approach that employs a sequence-to-sequence model trained to perform translation from error-free to erroneous text. Using an OCR engine, we generated a large parallel text corpus for training and produced several real-world noisy sequence labeling benchmarks for evaluation. Moreover, to overcome the data sparsity problem that exacerbates in the case of imperfect textual input, we learned noisy language model-based embeddings. Our approach outperformed the baseline noise generation and error correction techniques on the erroneous sequence labeling data sets. To facilitate future research on robustness, we make our code, embeddings, and data conversion scripts publicly available.

1 Introduction

Deep learning models have already surpassed human-level performance in many Natural Language Processing (NLP) tasks[1]. Sequence labeling systems have also reached extremely high accuracy (Akbik et al., 2019; Heinzerling and Strube, 2019). Still, NLP models often fail in scenarios, where non-standard text is given as input (Heigold et al., 2018; Belinkov and Bisk, 2018).

NLP algorithms are predominantly trained on error-free textual data but are also employed to process user-generated text (Baldwin et al., 2013; Derczynski et al., 2013) or consume the output of prior Optical Character Recognition (OCR) or Automatic Speech Recognition (ASR) processes (Miller et al., 2000). Errors that occur in any upstream component of an NLP system deteriorate the accuracy of the target downstream task (Alex and Burns, 2014).

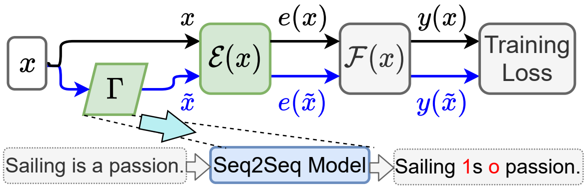

Figure 1: Our modification of the NAT approach (green boxes). We propose a learnable seq2seq-based error generator and re-train FLAIR embeddings using noisy text to improve the accuracy of noisy neural sequence labeling. Γ is a process that induces noise to the input x producing erroneous $\tilde{x}$. $\mathcal{E}(x)$ is an embedding matrix. $\mathcal{F}(x)$ is a sequence labeling model. $e(x)$ and $e(\tilde{x})$ are the embeddings of x and $\tilde{x}$, respectively. $y(x)$ and $y(\tilde{x})$ are the outputs of the model for x and $\tilde{x}$, respectively.

In this paper, we focus on the problem of performing sequence labeling on the text produced by an OCR engine. Moreover, we study the transferability of the methods learned to model OCR noise to the distribution of the human-generated errors. Both misrecognized and mistyped text pose a challenge for the standard models trained using error-free data (Namysl et al., 2020).

We make the following contributions (Figure 1):

- We propose a noise generation method for OCR that employs a sequence-to-sequence (seq2seq) model trained to translate from error-free to erroneous text (§4.1). Our approach improves the accuracy of noisy neural sequence labeling compared to prior work (§6.1).

- We present an unsupervised parallel training data generation method that utilizes an OCR engine (§4.2). Similarly, realistic noisy versions of popular sequence labeling data sets can be synthesized for evaluation (§5.5).

---

[1] GLUE benchmark (Wang et al., 2018a): `https://gluebenchmark.com/leaderboard`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 314–329
August 1–6, 2021. ©2021 Association for Computational Linguistics

- We exploit erroneous text to perform Noisy Language Modeling (NLM; §4.5). Our NLM embeddings further improve the accuracy of noisy neural sequence labeling (§6.3), also in the case of the human-generated errors (§6.4).
- To facilitate future research on robustness, we integrate our methods into the Noise-Aware Training (NAT) framework (Namysl et al., 2020) and make our code, embeddings, and data conversion scripts publicly available.[2]

2 Related Work

Errors of OCR, ASR, and other text generators always pose a challenge to the downstream NLP systems (Lopresti, 2009; Packer et al., 2010; Ruiz et al., 2017). Hence, methods for improving robustness are becoming increasingly popular.

Data Augmentation A widely adopted method of providing robustness to non-standard input is to augment the training data with examples perturbed using a model that mimics the error distribution to be encountered at test time (Cubuk et al., 2019).

Apparently, the exact modeling of noise might be impractical or even impossible—thus, methods that employ randomized error patterns for training recently gained increasing popularity (Heigold et al., 2018; Lakshmi Narayan et al., 2019). Although trained using synthetic errors, these methods are often able to achieve moderate improvements on data from natural sources of noise (Belinkov and Bisk, 2018; Karpukhin et al., 2019).

Spelling- and OCR Post-correction The most widely used method of handling noisy text is to apply error correction on the input produced by human writers (*spelling correction*) or the output of an upstream OCR component (*OCR post-correction*).

A popular approach applies *monotone seq2seq* modeling for the correction task (Schnober et al., 2016). For instance, Hämäläinen and Hengchen (2019) proposed *Natas*—an OCR post-correction method that uses character-level Neural Machine Translation (NMT). They extracted parallel training data using embeddings learned from the erroneous text and used it as input to their translation model.

Grammatical Error Correction Grammatical Error Correction (GEC; Ng et al., 2013, 2014; Bryant et al., 2019) aims to automatically correct ungrammatical text. GEC can be approached as a translation from an ungrammatical to a grammatical language, which enabled NMT seq2seq models to be applied to this task (Yuan and Briscoe, 2016). Due to the limited size of human-annotated GEC corpora, NMT models could not be trained effectively (Lichtarge et al., 2019), though.

Several studies investigated generating realistic erroneous sentences from grammatically correct text to boost training data (Kasewa et al., 2018; Grundkiewicz et al., 2019; Choe et al., 2019; Qiu and Park, 2019). Inspired by *back-translation* (Sennrich et al., 2016; Edunov et al., 2018), Artificial Error Generation (AEG) approaches (Rei et al., 2017; Xie et al., 2018) train an intermediate model in reverse order—to translate correct sentences to erroneous ones. Following AEG, we generate a large corpus of clean and noisy sentences and train a seq2seq model to produce rich and diverse errors resembling the natural noise distribution (§3.3, 4.2).

Noise-Invariant Latent Representations Robustness can also be improved by encouraging the models to learn a similar latent representation for both the error-free and the erroneous input.

Zheng et al. (2016) introduced *stability training*—a general method used to stabilize predictions against small input perturbations. Piktus et al. (2019) proposed *Misspelling Oblivious Embeddings* that embed the misspelled words close to their error-free counterparts. Jones et al. (2020) developed *robust encodings* that balance stability (consistent predictions across various perturbations) and fidelity (accuracy on unperturbed input) by mapping sentences to a smaller discrete space of encodings. Although their model improved robustness against small perturbations, it decreased accuracy on the error-free input.

Recently, Namysl et al. (2020) proposed the Noise-Aware Training method that employs stability training and data augmentation objectives. They exploited both the error-free and the noisy samples for training and used a *confusion matrix-based* error model to imitate the errors. In contrast to their approach, we employ a more realistic empirical error distribution during training (§3.3) and observe improved accuracy at test time (§6.1).

3 Problem Definition

3.1 Noisy Neural Sequence Labeling

Namysl et al. (2020) pointed out that the standard NLP systems are generally trained using error-free

---

[2] `https://github.com/mnamysl/nat-acl2021`

textual input, which causes a discrepancy between the training and the test conditions. These systems are thus more susceptible to non-standard, corrupted, or adversarial input.

To model this phenomenon, they formulated the *noisy neural sequence labeling* problem, assuming that every input sentence might be subjected to some unknown token-level noising process $\Gamma = P(\tilde{x}_i|x_i)$, where x_i is the original i-th token, and $\tilde{x}_i$ is its distorted equivalent. As a solution, they proposed the NAT framework, which trains the sequence labeling model using auxiliary objectives that exploit both the original sentences and their copies corrupted using a noising process that imitates the naturally occurring errors (Figure 1).

3.2 Confusion Matrix-Based Error Model

Namysl et al. (2020) used a confusion matrix-based method to model insertions, deletions, and substitutions of characters. Given a corpus of paired noisy and manually corrected sentences $\mathcal{P}$, they estimated the natural error distribution by calculating the alignments between the pairs $(\tilde{x}, x) \in \mathcal{P}$ of noisy and clean sentences using the *Levenshtein distance* metric (Levenshtein, 1966).

Moreover, as $\mathcal{P}$ is usually laborious to obtain, they proposed a *vanilla error model*, which assumes that all types of edit operations are equally likely:

$$\sum_{\tilde{c}\in\Sigma\setminus\{\varepsilon\}} P_{ins}(\tilde{c}|\varepsilon) = P_{del}(\varepsilon|c) = \sum_{\tilde{c}\in\Sigma\setminus\{c,\varepsilon\}} P_{subst}(\tilde{c}|c),$$

where c and $\tilde{c}$ are the original and the perturbed characters, respectively, Σ is an alphabet, and ε is a symbol introduced to model insertion and deletions.

3.3 Realistic Empirical Error Modeling

Namysl et al. (2020) compared the NAT models that used the vanilla- and the empirically-estimated confusion matrix-based error model and observed no advantages of exploiting the test-time error distribution during training. *Would we make the same observation given a more realistic error model?*

Even though the methods that used randomized error patterns were often successful, we argue that leveraging the empirical noise distribution for training would be beneficial, providing additional accuracy improvements. The data produced by the naïve noise generation methods may not resemble naturally occurring errors, which could lead the downstream models to learn misleading patterns.

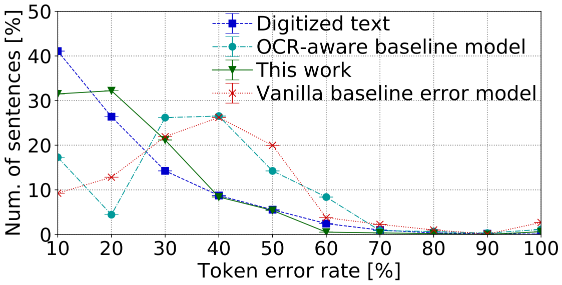

Figure 2: Distributions of the token error rates of sentences produced by the proposed and the baseline error models. For comparison, we plot the distribution of error rates in the text that contains naturally occurring errors. Each value n is the percentage of sentences with a token error rate in $[n-10, n)$.

In Figure 2, we compare the distributions of error rates of sentences produced by the proposed and the prior noise models with the distribution of errors in the digitized text. We can observe that the distribution of naturally occurring errors follows Zipf's law, while the baseline noise models produce Bell-shaped curves. Interestingly, both the vanilla and the empirical models exhibit similar characteristics, which could explain the observations from the prior work. In practice, the error rate is not uniform throughout the text. Some passages are recognized perfectly, while others can barely be deciphered. Our objective is thus to develop a noise model that produces a smoother distribution, imitating the errors encountered at test time more precisely (cf. *This work* in Figure 2).

Moreover, although the exact noise distribution in the test data cannot always be known beforehand, the noising process, e.g., an OCR engine, used to provide the input, can often be identified. We would thus take advantage of such prior knowledge to improve the efficiency of the downstream task.

3.4 Data Sparsity of Natural Language

Embeddings pre-trained on a large corpus of monolingual text are ubiquitous in NLP (Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2019). They capture syntactic and semantic textual features that can be exploited to solve higher-level NLP tasks.

Embeddings are generally trained using corpora that contain error-free text. Due to the data sparsity problem that arises from the large vocabulary sizes and the exponential number of feasible contexts, the majority of possible word sequences do not appear in the input data. Even though increasing the size of the training corpora was shown to improve the performance of language processing tasks (Brown et al., 2020), most of the misrecognized or mistyped tokens would still be unobserved and therefore poorly modeled when using the error-free text only. *Would it be beneficial to pre-train the embeddings on data that includes realistic erroneous sentences?*

3.5 The Flaws of Error Correction

Furthermore, we believe that the correction methods, although widely adopted, can only reliably manage moderately perturbed text (Flor et al., 2019). OCR post-correction has been reported to be challenging in the case of historical books that exhibit high OCR error rates (Rigaud et al., 2019).

We note that correction methods have no information about the downstream task to be performed. Moreover, in the automatic correction setting, they only provide the best guess for each token. Comparing their performance with the NAT approach in the context of sequence labeling would be informative.

4 Empirical Error Modeling

Figure 1 presents our modifications of the NAT framework. Firstly, we propose to replace the confusion matrix-based noising process (§3.2) with a noise induction method that generates a more realistic error distribution (§4.1-4.4). Secondly, to overcome the data sparsity problem (§3.4), we train language model-based embeddings using digitized text and use them as a substitution of the pre-trained model used in prior work (§4.5).

4.1 Sequence-to-Sequence Error Generator

Motivated by the AEG approaches (Rei et al., 2017; Xie et al., 2018), we propose a learnable error generation method that employs a character-level seq2seq model to perform monotone string translation (Schnober et al., 2016). It directly models the conditional probability $p(\tilde{x}|x)$ of mapping error-free text x into erroneous text $\tilde{x}$ using an attention-based encoder-decoder framework (Bahdanau et al., 2015). The encoder computes the representation $h=\{h_1, \ldots, h_n\}$ of x, where n is the length of x. The decoder generates $\tilde{x}$ one token at a time:

$$p(\tilde{x}|x) = \prod_{i=1}^{n} p(\tilde{x}_i|\tilde{x}_{<i}, x, c),$$

where $c=f_{attn}(\{h_1, \ldots, h_n\})$ is a vector generated from h, and f_{attn} is an attention function.

Our models are trained to maximize the likelihood of the training data. At inference time, we randomly sample the subsequent tokens from the learned conditional language model.

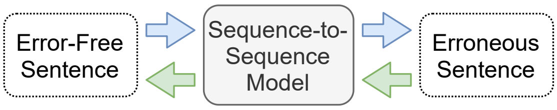

Figure 3: Schematic visualization of the error generation (blue arrows) and the error correction (green arrows) methods. The parallel data can be utilized to train seq2seq models for both tasks.

Note that our approach reverses the standard seq2seq error correction pipeline, which uses the erroneous text as input and trains the model to produce the corresponding error-free string (Figure 3). By interchanging the input and the output data, we can also readily train sentence correction models. One difference is that at inference time we would prefer to perform beam search and select the best decoding result rather than sampling subsequent characters from the learned distribution.

4.2 Unsupervised Parallel Data Generation

To train our error generation model (§4.1), we need a large parallel corpus $\mathcal{P}$ of error-free and erroneous sentences. AEG approaches use seed GEC corpora to learn the inverse models directly. Unfortunately, we are not aware of any comparably large resources for digitized text that could be used for this task.

To address this issue, we propose an unsupervised sentence-level parallel data generation approach for OCR (Figure 4). First, we collect a large seed corpus $\mathcal{T}$ that contains the error-free text. We then render each sentence and subsequently run text recognition on the rendered images using an OCR engine. Moreover, to increase the variation in training data, we sample different fonts for rendering. Furthermore, to simulate the distortions and degradation of the printed material, we induce pixel-level noise to the images before recognition.

Note that our approach is universal and could be used to generate parallel data sets for other tasks, e.g., an ASR system could be trained on samples from a Text-to-Speech engine (Wang et al., 2018b).

4.3 Sentence- and Word-Level Modeling

We note that the sequence labeling problem is formulated at the word-level, i.e., each word has a class label assigned to it. To employ our method in

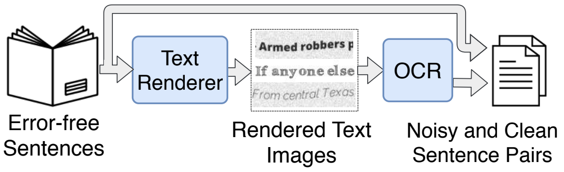

Figure 4: Our parallel data generation method for OCR. We render sentences extracted from a text corpus. Subsequently, an OCR engine recognizes the text depicted in the rendered images. Finally, the pairs of original and recognized sentences are gathered together to form a parallel corpus used to train translation models.

this scenario, we develop (i) a *sentence-level* and (ii) a *token-level* variant of our error generator.

Our sentence-level error generator uses a seq2seq model trained to translate from error-free to erroneous sentences. It can potentially utilize contextual information from surrounding words, which may improve the quality of the results. During the training of a NAT model, a learned seq2seq model translates the original input x to generate $\tilde{x}$. Subsequently, we use an alignment algorithm (§4.4) to transfer the word-level annotations from x to $\tilde{x}$.

Our token-level error generator uses a seq2seq model trained to translate from error-free to erroneous words. It relies exclusively on the input and the output words. We use the alignment algorithm to build a training set for this task, i.e., extract word-level parallel data from the corpus of parallel sentences (§4.2). During the training of a NAT model, a learned generator translates each word x_i from x to produce the erroneous sentence $\tilde{x}$.

4.4 Word-Level Sentence Alignment

Figure 5 illustrates the alignment procedure, which we developed to extract word-level parallel training data for our token-level generator and to transfer the labels to the erroneous sentences for the sentence-level generator in the sequence labeling scenario.

To this end, we align each pair of error-free and noisy sentences at the word-level using the Levenshtein Distance algorithm. Our alignment procedure produces pairs of aligned words. The annotations for words are transferred accordingly.

4.5 Noisy Language Modeling

Recently, Xie et al. (2017) drew a connection between input noising in neural network language models and smoothing in n-gram models. We believe that data noising could be an effective technique for regularizing neural language models that could help to overcome the data sparsity problem of imperfect natural language text and enable learning meaningful representation of erroneous tokens.

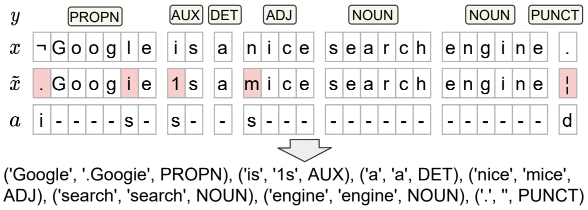

Figure 5: Our sentence alignment procedure. We align the original and the recognized sentences (x and $\tilde{x}$, respectively) using the sequence of edit operations a, which include insertions ”i”, deletions ”d”, and substitutions ”s” of characters. We use ”¬” and ”¦” as placeholders for the insertion and the deletion operation, respectively. Matched characters are marked with ”-”. The alignment procedure produces a list of paired error-free and possibly erroneous words with class labels (optional).

To this end, we propose to include the data from noisy sources in the corpora used to train LM-based embeddings. Specifically, in this work, we learn a noisy language model using the output of an OCR engine (§4.2) that captures the characteristics of OCR errors. Any other noisy source could be readily used to model related domains, e.g., ASR-transcripts or ungrammatical text.

5 Experimental Setup

5.1 Sequence-to-Sequence Error Generator

To learn our error generators (§4.1), we utilize the OpenNMT[3] toolkit (Klein et al., 2017).[4] We encode the input sentence at the character-level before feeding it to the seq2seq model. Subsequently, the output produced by the seq2seq model is decoded back to the original form (Figure 6).

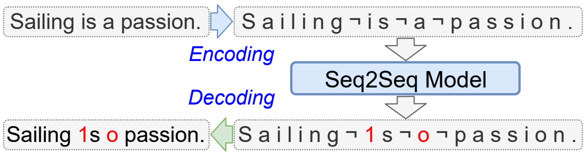

Figure 6: Sentence encoding-decoding schema. The whitespace characters are first replaced with a placeholder symbol ”¬”. The sentences are tokenized at the character-level by adding whitespace between every pair of characters. Decoding reverses this process.

---

[3] `https://github.com/OpenNMT/OpenNMT-py`

[4] We list all non-default hyper-parameters in Table 8.

5.2 Unsupervised Parallel Data Generation

Following the approach from §4.2, we generated a large parallel corpus $\mathcal{P}$ to train our error generation and correction models. We sampled 10 million sentences[5] from the English part of the *1 Billion Word Language Model Benchmark*[6] and used them as the source of error-free text, i.e., the seed corpus $\mathcal{T}$. We rendered each sentence as an image using the *Text Recognition Data Generator* package[7]. We used 90 different fonts for rendering and applied random distortions to the rendered images. Subsequently, we performed OCR on each image of text using a Python wrapper[8] for Tesseract-OCR[9] (Smith, 2007). We present the distribution of error rates in our noisy corpus in Figure 2 (cf. the *digitized text* plot).

5.3 Sequence Labeling

Training Setup We employed the NAT framework[10] (Figure 1) to study the robustness of sequence labeling systems. Following Akbik et al. (2018), we used a combination of *FLAIR* and *GloVe* embeddings in all experiments.[11] We employed the data augmentation ($\mathcal{L}_{\text{AUGM}}$) and the stability training ($\mathcal{L}_{\text{STAB}}$) objectives with default weights ($\alpha = 1.0$), as proposed by Namysl et al. (2020). Consistent with prior work, erroneous sentences $\tilde{x}$ were generated dynamically in every epoch.

Tasks We experimented with the Named Entity Recognition (NER) and Part-of-Speech Tagging (POST) tasks. NER aims to locate all named entity mentions in text and classify them into predefined classes, e.g., person names, locations, and organizations. POST is the process of tagging each word in the text with the corresponding part of speech.

Evaluation Setup The evaluation pipeline is shown in Figure 7. Following Akbik et al. (2018), we report the entity-level micro-average F1 score for NER and the accuracy for POST.

5.4 Baselines

Error Generation We compared our error generator with the OCR-aware noise model from Namysl et al. (2020). We used the noisy part of the parallel corpus $\mathcal{P}$ to estimate the confusion matrix employed by this baseline. Moreover, in the NLM experiment (§6.3), we also evaluated the vanilla error model proposed by Namysl et al. (2020).

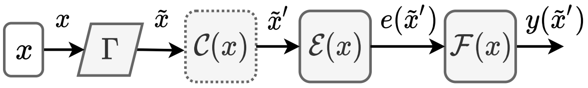

Figure 7: Evaluation pipeline. Γ is a noising process that transforms x into $\tilde{x}$. $\mathcal{C}(x)$ is an optional text correction module that returns $\tilde{x}'$ ($\tilde{x}'=\tilde{x}$, if $\mathcal{C}(x)$ is absent). $\mathcal{E}(x)$ is an embedding matrix. $\mathcal{F}(x)$ is a sequence labeling model. $e(\tilde{x}')$ and $y(\tilde{x}')$ are the embeddings and the output of the model for $\tilde{x}'$, respectively.

Error Correction To evaluate error correction, we trained the sequence labeling models using the standard objective ($\mathcal{L}_0$) and employed the text correction method on the erroneous input before feeding it to the network (Figure 7).

We examined *Natas*[12], the seq2seq OCR post-correction method proposed by Hämäläinen and Hengchen (2019). We trained context-free error correction models compatible with Natas using our parallel corpus (§5.2). Moreover, we also employed the widely adopted spell checker *Hunspell*[13].

5.5 Data Sets

Original Benchmarks For NER, we employed the CoNLL 2003 data set (Tjong Kim Sang and De Meulder, 2003). To evaluate POST, we utilized the Universal Dependency Treebank (UD English EWT; Silveira et al., 2014). We present the detailed statistics of both data sets in Table 5.

Data Set	Geom. Distort.	Pixel-level Noise	CoNLL 2003	UD English EWT
Tesseract 3♣	✗	✗	22.72%	23.31%
Tesseract 4◇	✓	✓	16.35%	22.12%
Tesseract 4♡	✓	✗	14.89%	20.38%
Tesseract 4♠	✗	✓	3.53%	5.83%
Typos	n/a	n/a	15.53%	15.22%

Table 1: The noisy sequence labeling data sets that we generated either by applying OCR on rendered sentences from an original benchmark (first four rows) or by inducing misspellings (last row). We generated multiple variants of the former data sets by combining geometrical distortions and pixel-level noise induction. The last two columns present the token error rates (the column headers indicate the names of the original benchmarks).

---

[5] Which accounts for about 253 million words.

[6] https://www.statmt.org/lm-benchmark

[7] https://pypi.org/project/trdg

[8] https://github.com/sirfz/tesserocr

[9] We used Tesseract v4.0 to generate the parallel data set.

[10] https://github.com/mnamysl/nat-acl2020

[11] Other hyper-parameters also follow Akbik et al. (2018).

[12] https://github.com/mikahama/natas

[13] https://hunspell.github.io

Training Loss	Noise Model	Correction Method	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
$\mathcal{L}_0$	n/a	—	**92.54±0.08**	80.48±0.09	84.71±0.19	85.62±0.08	91.50±0.08
	n/a	Hunspell	**92.54±0.08**	**82.17±0.11**	**85.80±0.11**	**86.70±0.07**	**91.73±0.11**
	n/a	Natas	**92.54±0.08**	77.80±0.19	84.50±0.11	85.24±0.10	91.33±0.13
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	—	92.56±0.06	**85.29±0.16**	88.62±0.08	89.19±0.12	92.04±0.07
	seq2seq (token-level, §4.3)	—	**92.76±0.07**	**85.38±0.16**	**89.39±0.17**	**89.99±0.22**	**92.37±0.10**
	seq2seq (sentence-level, §4.3)	—	**92.81±0.11**	84.38±0.15	88.96±0.18	**89.67±0.26**	**92.44±0.17**
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	—	92.23±0.12	**84.49±0.10**	87.58±0.13	88.40±0.20	91.65±0.14
	seq2seq (token-level, §4.3)	—	**92.24±0.18**	**84.25±0.23**	**88.24±0.25**	**88.91±0.21**	**91.86±0.16**
	seq2seq (sentence-level, §4.3)	—	**92.45±0.12**	83.89±0.30	**88.14±0.23**	**88.88±0.11**	**91.99±0.11**

(a) English CoNLL 2003

$\mathcal{L}_0$	n/a	—	**96.96±0.04**	86.75±0.16	86.97±0.14	88.30±0.16	94.34±0.07
	n/a	Hunspell	**96.96±0.04**	87.53±0.14	86.74±0.14	88.12±0.16	94.49±0.08
	n/a	Natas	**96.96±0.04**	**88.98±0.10**	**88.94±0.14**	**89.68±0.16**	**95.11±0.08**
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	—	**96.90±0.06**	**91.35±0.13**	92.12±0.14	92.99±0.21	96.17±0.07
	seq2seq (token-level, §4.3)	—	96.76±0.04	**91.44±0.11**	**93.65±0.13**	**94.19±0.10**	**96.26±0.07**
	seq2seq (sentence-level, §4.3)	—	96.78±0.06	90.92±0.08	93.37±0.08	**94.10±0.03**	**96.27±0.03**
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	—	**96.80±0.04**	91.16±0.07	91.93±0.11	92.77±0.10	96.06±0.02
	seq2seq (token-level, §4.3)	—	96.65±0.07	**91.36±0.12**	**93.34±0.09**	**93.97±0.05**	**96.14±0.07**
	seq2seq (sentence-level, §4.3)	—	96.67±0.05	90.70±0.14	93.05±0.17	93.71±0.13	**96.15±0.05**

(b) UD English EWT

Table 2: Comparison of error generation (§6.1) and error correction (§6.2) approaches on the original and noisy English CoNLL 2003 and the UD English EWT test sets (§5.5). We report mean and standard deviation F1 scores (CoNLL 2003) and accuracies (UD English EWT) over five runs with different random initialization. $\mathcal{L}_0$, $\mathcal{L}_{\text{AUGM}}$, $\mathcal{L}_{\text{STAB}}$ is the standard, the data augmentation, and the stability objective, respectively (Namysl et al., 2020). **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

Noisy Benchmarks Unfortunately, we did not find any publicly available noisy sequence labeling data set that could be used to benchmark different methods for improving robustness. To this end, we generated several noisy versions of the original sequence labeling data sets (Table 1). We extracted the sentences from each original benchmark and applied the procedure described in §4.2.[14] We transferred the word-level annotations as described in §4.4. Finally, we produced the data in the CoNLL format (Table 7).

Moreover, to evaluate the transferability of error generators, we followed Namysl et al. (2020) and synthetically induced misspellings to the error-free data sets. To this end, we used the lookup tables of possible lexical replacements released by Belinkov and Bisk (2018) and Piktus et al. (2019).[15]

6 Experimental Results

6.1 Empirical Noise Generation Approaches

In this experiment, we compared the NAT models that employed either our seq2seq noise generators or the baseline error models (Table 2). In this evaluation scenario, we do not employ $\mathcal{C}(x)$ (Figure 7).

Our error generators outperformed the OCR-aware confusion matrix-based model on the noisy benchmarks generated using the Tesseract 4 engine. The advantage of our method was less emphasized in the case of the *Tesseract 3*$^{\clubsuit}$ data sets. The token-level translation method performed better than the sentence-level variant, while the latter was more efficient when the error rate of the input was lower (cf. the *original data* and the *Tesseract 4*$^{\spadesuit}$ columns), although it often struggled with translating long sentences. Moreover, data augmentation generally outperformed stability training, which is consistent with the observation from Namysl et al. (2020).

Furthermore, we observe a slight decrease in accuracy on the original UD English EWT with both auxiliary objectives. We believe that this was caused by the different proportions of the tokens that were perturbed during training by our seq2seq error generators (e.g., 18% and 19.5% in the case of our token-level model for CoNLL2003 and UD English EWT, respectively). The trade-off between accuracy for clean and noisy data has thus been shifted towards the latter. We also notice a greater advantage of the seq2seq method over the baseline

---

[14] We directly applied both Tesseract v3.04 and v4.0. We used different sets of distortions and image backgrounds than those employed to generate parallel training data.

[15] We merged both sets of misspellings for evaluation.

Training Loss	Noise Model	NLM	Original Data	Tesseract 3♣	Tesseract 4◇	Tesseract 4♡	Tesseract 4♠
$\mathcal{L}_0$	n/a	✗	**92.54±0.08**	80.48±0.09	84.71±0.19	85.62±0.08	91.50±0.08
	n/a	✓	92.09±0.07	**83.83±0.21**	**88.17±0.12**	**88.71±0.17**	**91.68±0.09**
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	✗	**92.56±0.06**	85.29±0.16	88.62±0.08	89.19±0.12	92.04±0.07
	vanilla (§3.2)	✗	92.39±0.11	85.59±0.23	88.01±0.17	88.65±0.20	91.93±0.13
	vanilla (§3.2)	✓	92.45±0.05	**87.28±0.19**	**90.12±0.19**	**90.43±0.19**	**92.17±0.05**
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	✗	**92.23±0.12**	84.49±0.10	87.58±0.13	88.40±0.20	**91.65±0.14**
	vanilla (§3.2)	✗	92.04±0.06	84.63±0.17	87.24±0.24	88.02±0.10	**91.52±0.12**
	vanilla (§3.2)	✓	91.85±0.07	**86.79±0.11**	**89.32±0.12**	**89.77±0.05**	**91.51±0.07**

Table 3: Comparison of the NAT approach with and without our NLM embeddings (§6.3) on the English CoNLL 2003 test set (§5.5). We report mean and standard deviation F1 scores over five runs with different random initialization. $\mathcal{L}_0$, $\mathcal{L}_{\text{AUGM}}$, $\mathcal{L}_{\text{STAB}}$ is the standard, the data augmentation, and the stability objective, respectively (Namysl et al., 2020). The *NLM* column indicates whether the model employed our NLM embeddings. **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

on the noisy UD English EWT data sets.

Additionally, in §A, we analyze the relationship between the size of the parallel corpus used for training and the F1 score of the NER task.

6.2 Error Generation vs. Error Correction

We compared the NAT approach with the baseline correction methods (§5.4). Preliminary experiments revealed that these baselines underperformed due to the *overcorrection* problem. To make them more competitive, we extended their default dictionaries by adding all tokens from the corresponding test sets for evaluation. Although the vocabulary of a test set could rarely be entirely determined, this setting would simulate a scenario where accurate in-domain vocabularies could be exploited.

Table 2 includes the results of this experiment. As expected, although more general, error correction techniques were outperformed by the NAT approach regardless of the noising method used. Surprisingly, Hunspell performed better than Natas on CoNLL 2003. We carried out a thorough inspection of the results of both methods and found out that Natas, although generally more accurate, had problems with recognizing tokens that were a part of entities. This behavior could be a flaw of data-driven error correction methods, as the entities are relatively rare in written text and are often out-of-vocabulary tokens (Alex and Burns, 2014).

6.3 Noisy Language Modeling

FLAIR (Akbik et al., 2018) learns a bidirectional LM to represent sequences of characters. We used the target side of our parallel data corpus (§5.2) to re-train FLAIR embeddings on the noisy digitized text.[16] Subsequently, we compared the accuracy of the vanilla NAT models (§3.2) that employed either the pre-trained or our NLM embeddings. Moreover, we do not use $\mathcal{C}(x)$ in this scenario (Figure 7).

Note that the noise model and the embeddings are two distinct components of the NAT architecture (Γ and $\mathcal{E}(x)$ in Figure 1, respectively) and therefore they could be easily combined. However, in this work, we do not mix our NLM with empirically estimated error models to avoid the twofold empirical error modeling effect. We leave the evaluation of this combination to future work.

Table 3 summarizes the results of this experiment. Our method significantly improved the accuracy across all training objectives, even when we employed exclusively the standard training objective for the sequence labeling task ($\mathcal{L}_0$). Surprisingly, we also achieved evident improvements for the noisy data set generated using the Tesseract 3 engine, which confirms that NLM embeddings can model the features of erroneous tokens even in the out-of-domain scenarios. On the other hand, the NLM slightly decreased the accuracy on the original data for the standard training objective. We plan to investigate this effect in future work by eliminating possible differences in the pre-training procedure and comparing our NLM against a model trained on the original error-free text corpus instead of using the embeddings from Akbik et al. (2018).

6.4 Human-Generated Errors

In this experiment, we evaluated the utility of our seq2seq error generators learned to model OCR noise (§6.1) and our NLM embeddings (§6.3) in a scenario where the input contains human-generated

---

[16] The hyper-parameters were consistent with prior work.

Training Loss	Noise Model	NLM	English CoNLL 2003
$\mathcal{L}_0$	n/a	✗	88.79±0.07
	n/a	✓	**89.60±0.24**
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	✗	90.82±0.12
	vanilla (§3.2)	✗	90.77±0.14
	vanilla (§3.2)	✓	**91.10±0.05**
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	✗	90.30±0.13
	vanilla (§3.2)	✗	90.34±0.06
	vanilla (§3.2)	✓	**90.53±0.07**

(a) NLM Embeddings

Training Loss	Noise Model	English CoNLL 2003	UD English EWT
$\mathcal{L}_0$	n/a	88.79±0.07	90.54±0.11
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	**90.82±0.12**	**93.63±0.11**
	seq2seq (token-level, §4.3)	**90.92±0.13**	92.87±0.08
	seq2seq (sentence-level, §4.3)	**90.77±0.19**	92.68±0.09
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	**90.30±0.13**	**93.37±0.05**
	seq2seq (token-level, §4.3)	**90.19±0.12**	92.79±0.08
	seq2seq (sentence-level, §4.3)	**90.15±0.16**	92.42±0.11

(b) Empirical Error Generation Methods

Table 4: Transferability of the methods learned to model OCR noise to the distribution of the human-generated errors (§6.4): **(a)** Comparison of the NAT approach with and without our NLM embeddings on the English CoNLL 2003 test set with human-generated errors. **(b)** Comparison of empirical error generation approaches on the English CoNLL 2003 and the UD English EWT test sets with human-generated errors. We report mean and standard deviation F1 scores (CoNLL 2003) and accuracies (UD English EWT) over five runs with different random initialization. $\mathcal{L}_0$, $\mathcal{L}_{\text{AUGM}}$, $\mathcal{L}_{\text{STAB}}$ is the standard, the data augmentation, and the stability objective, respectively (Namysl et al., 2020). The *NLM* column indicates whether the model employed our NLM embeddings. **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

errors. For evaluation, we used the noisy data sets with synthetically induced misspellings (§5.5). We do not employ $\mathcal{C}(x)$ in this scenario (Figure 7).

Table 4 summarizes the results of this experiment. The models with our NLM embeddings outperformed the baselines for all training objectives (Table 4a). The seq2seq error generation approach performed on par with the confusion matrix-based models on the CoNLL 2003 data set, while the latter achieved better accuracy on the UD English EWT data set (Table 4b).

We believe that this difference was caused by the discrepancy between the data distributions. Note that although the data used in this experiment reflects the patterns of human-generated errors, the distribution of these errors does not necessarily follow the natural distribution of human-generated errors, as it was synthetically generated using a fixed replacement probability that was uniform across all candidates.[17] Nevertheless, our methods proved to be beneficial in this scenario, which would suggest that the errors made by human writers and by the text recognition engines have common characteristics that were exploited by our method.

7 Conclusions

In this work, we studied the task of performing sequence labeling on noisy digitized and human-generated text. We extended the NAT approach and proposed the empirical error generator that performs the translation from error-free to erroneous text (§4.1). To train our generator, we developed an unsupervised parallel data synthesis method (§4.2). Analogously, we produced several realistic noisy evaluation benchmarks (§5.5). Moreover, we introduced the NLM embeddings (§4.5) that overcome the data sparsity problem of natural language.

Our approach outperformed the baseline noise induction and error correction methods, improving the accuracy of the noisy neural sequence labeling task (§6.1-6.3). Furthermore, we demonstrated that our methods are transferable to the out-of-domain scenarios - human-generated errors (§6.4) and the noise induced by a different OCR engine (§6.1, 6.3). We incorporated our approach into the NAT framework and make the code, embeddings, and scripts from our experiments publicly available.

Grundkiewicz and Junczys-Dowmunt (2019) showed that that unsupervised systems benefit from domain adaptation on authentic labeled data. For future work, we plan to fine-tune NAT models pretrained on synthetic samples using the labeled data generated directly by the noising process.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

---

[17] For comparison, we visualized the error distributions of our noisy benchmarks in Figure 9.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Beatrice Alex and John Burns. 2014. Estimating and rating the quality of optically character recognised text. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 97–102, New York, NY, USA. ACM.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364, Nagoya, Japan. Asian Federation of Natural Language Processing.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–227, Florence, Italy. Association for Computational Linguistics.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 198–206, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Michael Flor, Michael Fried, and Alla Rozovskaya. 2019. A benchmark corpus of English misspellings and a minimally-supervised model for spelling correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 76–86, Florence, Italy. Association for Computational Linguistics.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2019. Minimally-augmented grammatical error correction. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 357–363, Hong Kong, China. Association for Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Mika Hämäläinen and Simon Hengchen. 2019. From the paft to the fiiture: A fully automatic NMT and word embeddings method for OCR post-correction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 431–436, Varna, Bulgaria. INCOMA Ltd.

Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef van Genabith. 2018. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 68–80, Boston, MA. Association for Machine Translation in the Americas.

Benjamin Heinzerling and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, Florence, Italy. Association for Computational Linguistics.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, Hong Kong, China. Association for Computational Linguistics.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Pooja Lakshmi Narayan, Ajay Nagesh, and Mihai Surdeanu. 2019. Exploration of noise strategies in semi-supervised named entity classification. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 186–191, Minneapolis, Minnesota. Association for Computational Linguistics.

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8).

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Lopresti. 2009. Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):141–151.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

David Miller, Sean Boisen, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 2000. Named entity extraction from noisy input: Speech and OCR. In *Sixth Applied Natural Language Processing Conference*, pages 316–324, Seattle, Washington, USA. Association for Computational Linguistics.

Marcin Namysl, Sven Behnke, and Joachim Köhler. 2020. NAT: Noise-aware training for robust neural sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1501–1517, Online. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Thomas L. Packer, Joshua F. Lutes, Aaron P. Stewart, David W. Embley, Eric K. Ringger, Kevin D. Seppi, and Lee S. Jensen. 2010. Extracting person names from diverse and noisy ocr text. In *Proceedings of the Fourth Workshop on Analytics for Noisy Unstructured Text Data*, AND '10, page 19–26, New York, NY, USA. Association for Computing Machinery.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Associ-*

ation for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Edouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3226–3234, Minneapolis, Minnesota. Association for Computational Linguistics.

Mengyang Qiu and Jungyeul Park. 2019. Artificial error generation with fluency filtering. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 87–91, Florence, Italy. Association for Computational Linguistics.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292, Copenhagen, Denmark. Association for Computational Linguistics.

C. Rigaud, A. Doucet, M. Coustaty, and J. Moreux. 2019. ICDAR 2019 competition on post-OCR text correction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1588–1593.

Nicholas Ruiz, Mattia Antonino Di Gangi, Nicola Bertoldi, and Marcello Federico. 2017. Assessing the tolerance of neural machine translation systems against speech recognition errors. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2635–2639.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan. The COLING 2016 Organizing Committee.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018b. A hybrid approach to automatic corpus generation for Chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527, Brussels, Belgium. Association for Computational Linguistics.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana. Association for Computational Linguistics.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *International Conference on Learning Representations (ICLR)*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Stephan Zheng, Yang Song, Thomas Leung, and Ian J. Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4480–4488.

A Relationship with the Corpus Size

Empirical error generators are especially beneficial when we can approximate the noise distribution to be encountered at test time. In this experiment, we aimed to answer the question, how much parallel training data is required to train a solid seq2seq error generation model.

Figure 8 shows that the NAT models that used our seq2seq error generator performed better than those employing the baseline vanilla error model proposed by Namysl et al. (2020) for all noisy benchmarks that were generated using the *Tesseract 4* OCR engine. The improvements were observed even when we used as few as 1000 parallel training sentences. Our method also outperformed the baseline on the original CoNLL 2003 benchmark. On the contrary, the accuracy of models trained using our generator fell slightly behind the baseline on the *Tesseract 3*♣ and *Typos* data sets.

B Sequence Labeling Data Sets

Original Benchmarks Table 5 presents the detailed statistics of the original sequence labeling benchmarks used in our experiments. For NER, we employed CoNLL 2003[18] (Tjong Kim Sang and De Meulder, 2003). To evaluate POST, we utilized Universal Dependency Treebank (UD English EWT[19]; Silveira et al., 2014).

Noisy Benchmarks Table 6 presents the error rates and the correction accuracies of the Natas and Hunspell methods calculated on the test sets of the noisy sequence labeling benchmarks. Moreover, Table 7 shows an excerpt from a noisy sequence labeling data set generated for evaluation.

Furthermore, Figure 9 presents the distribution of token error rates in relation to the percentage number of sentences in our noisy data sets. For comparison, we also included the distributions obtained by applying different noise generation methods - the vanilla- and the OCR-aware confusion matrix-based models by Namysl et al. (2020), and our token-level seq2seq error generator.

We note that the error distribution of our noisy data sets is closer to the Zipf distribution in contrast to the results of prior methods that exhibit a Bell-Curve pattern. Note that the *Typos* data set was generated by randomly sampling possible lexical replacement candidates from the lookup tables, hence its distribution exhibits slightly different characteristics than the noisy data sets generated by directly applying the OCR engine to the rendered text images. Based on the above results,

	Train	Dev	Test	Total
Sentences	14,041	3,250	3,453	20744
Tokens	203,621	51,362	46,435	301418
PER	6,600	1,842	1,617	10059
LOC	7,140	1,837	1,668	10645
ORG	6,321	1,341	1,661	9323
MISC	3,438	922	702	5062

(a) English CoNLL 2003.

	Train	Dev	Test	Total
Sentences	12543	2002	2077	16622
Tokens	204585	25148	25096	254829
ADJ	12458	1784	1689	15931
ADP	17625	2021	2020	21666
ADV	10553	1264	1226	13043
AUX	12396	1512	1504	15412
CCONJ	6703	780	738	8221
DET	16284	1895	1896	20075
INTJ	688	115	120	923
NOUN	34765	4196	4129	43090
NUM	3996	378	536	4910
PART	5567	630	630	6827
PRON	18584	2219	2158	22961
PROPN	12945	1879	2075	16899
PUNCT	23676	3083	3106	29865
SCONJ	3850	403	386	4639
SYM	643	75	100	818
VERB	23005	2759	2644	28408
X	847	155	139	1141

(b) UD English EWT.

Table 5: Statistics of the English CoNLL 2003 and the UD English EWT data sets. We present statistics of the training (Train) development (Dev) and test (Test) sets, including the number of sentences and tokens. CoNLL 2003 contains the annotations for the following entity types: person names (PER), locations (LOC), organizations (ORG), and miscellaneous (MISC). For UD English EWT, the following universal POS tags were included: ADJ (adjective), ADP (adposition), ADV (adverb), AUX (auxiliary), CCONJ (coordinating conjunction), DET (determiner), INTJ (interjection), NOUN (noun), NUM (numeral), PART (particle), PRON (pronoun), PROPN (proper noun), PUNCT (punctuation), SCONJ (subordinating conjunction), SYM (symbol), VERB (verb), X (other).

---

[18] `https://www.clips.uantwerpen.be/conll2003/ner`

[19] `https://universaldependencies.org/treebanks/en_ewt` (version 2.6)

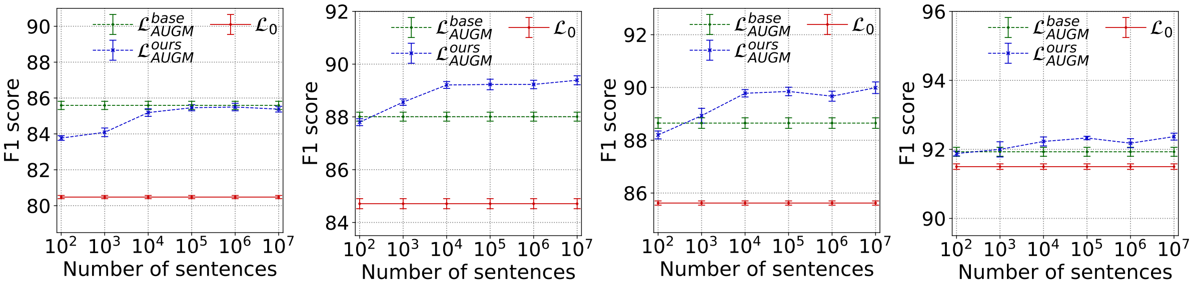

(a) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $3^{\clubsuit}$). (b) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $4^{\diamondsuit}$). (c) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $4^{\heartsuit}$). (d) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $4^{\spadesuit}$).

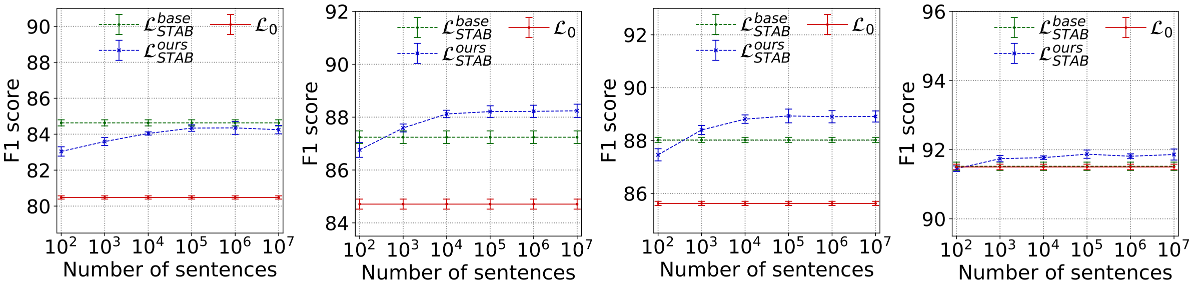

(e) $\mathcal{L}_{\text{STAB}}$ (Tesseract $3^{\clubsuit}$). (f) $\mathcal{L}_{\text{STAB}}$ (Tesseract $4^{\diamondsuit}$). (g) $\mathcal{L}_{\text{STAB}}$ (Tesseract $4^{\heartsuit}$). (h) $\mathcal{L}_{\text{STAB}}$ (Tesseract $4^{\spadesuit}$).

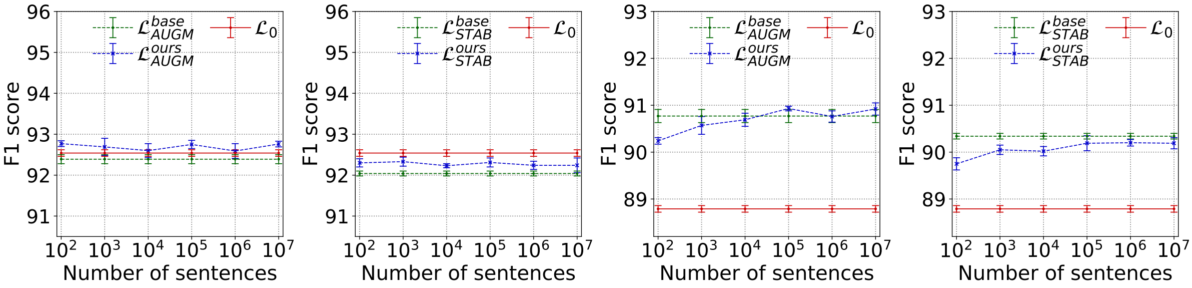

(i) $\mathcal{L}_{\text{AUGM}}$ (original data set). (j) $\mathcal{L}_{\text{STAB}}$ (original data set). (k) $\mathcal{L}_{\text{AUGM}}$ (Typos). (l) $\mathcal{L}_{\text{STAB}}$ (Typos).

Figure 8: F1 score in relation to the number of parallel sentences. The experiments were conducted on the original English CoNLL 2003 benchmark and its noisy variants: Tesseract $3^{\clubsuit}$, Tesseract $4^{\diamondsuit}$, Tesseract $4^{\heartsuit}$, Tesseract $4^{\spadesuit}$, and Typos. We compare the accuracy of our token-level seq2seq approach with the vanilla error model (Namysl et al., 2020), and the standard objective ($\mathcal{L}_0$). We present the results for both auxiliary objectives: the data augmentation ($\mathcal{L}^{\text{ours}}_{\text{AUGM}}$, $\mathcal{L}^{\text{base}}_{\text{AUGM}}$) and the stability training ($\mathcal{L}^{\text{ours}}_{\text{STAB}}$, $\mathcal{L}^{\text{base}}_{\text{STAB}}$).

we believe that our noisy data sets are better suited for the evaluation of the robustness of sequence labeling models than the data generated by the prior approaches.

Data Conversion Scripts Because of licensing and copyright reasons, we did not submit the noisy data sets directly. Our code includes the scripts for the conversion of the original benchmarks into their noisy variants. For reference, we added excerpts of the noisy UD English EWT data set in the supplementary materials.

C Reproducibility

In this section, we present additional information that could facilitate reproducibility.

Hyper-parameters To train our seq2seq translation models, we generally used the default hyper-parameters of the OpenNMT toolkit. We list all non-default values in Table 8. Moreover, we decayed the learning rate eight times during the training for all models. Furthermore, we utilized *copy attention* (See et al., 2017) for our error generation models and *global attention* (Luong et al., 2015) for the error correction model.

Validation Accuracy Table 9 summarizes the validation accuracy of our seq2seq models for error generation. We trained the sentence-level models for 1.6×10^4 and the token-level models for 4×10^5 iterations or at least one epoch of training. Moreover, the token-level error correction model employed by Natas was trained for one epoch (about 4×10^5 iterations) on one million parallel sentences

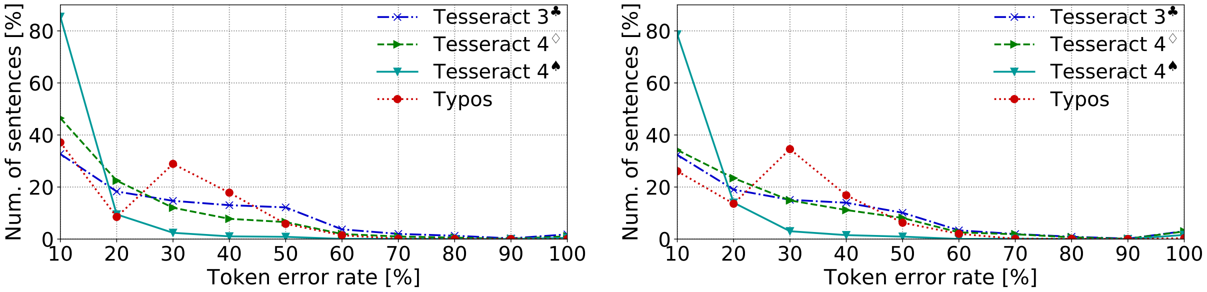

(a) Noisy sentence labeling data sets (English CoNLL 2003). (b) Noisy sentence labeling data sets (UD English EWT).

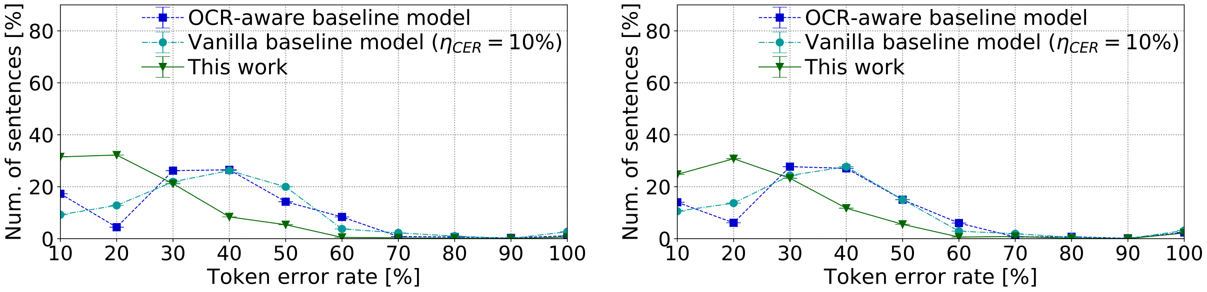

(c) Noise induction methods (English CoNLL 2003). (d) Noise induction methods (UD English EWT).

Figure 9: Distributions of the token error rates of sentences in our noisy sequence labeling data sets (Tesseract 3$^{\clubsuit}$, Tesseract 4$^{\diamondsuit}$, Tesseract 4$^{\spadesuit}$, and Typos). For comparison, we include error distributions obtained by applying our seq2seq token-level error generator and the baseline confusion matrix-based error models (Namysl et al., 2020) to the sentences extracted from the original benchmark. η_{CER} is the character-level noising factor used by the vanilla error model. Each point is the percentage of sentences with a token error rate that falls into a specific token error range, i.e., the value of 50 corresponds to the sentences with a token error rate greater than 40 and lower than or equal to 50.

Measure	Method	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$	Typos
TER	Original	22.72	16.35	14.89	3.53	15.53
	Natas	**17.24**	**12.20**	**11.13**	**2.34**	11.53
	Hunspell	17.44	13.54	12.24	2.43	**10.69**
TER (entities)	Original	29.66	16.70	15.00	3.61	8.20
	Natas	27.81	14.97	13.36	2.93	7.62
	Hunspell	**16.63**	**9.95**	**8.76**	**1.89**	**4.07**
ACC	Natas	**24.13**	**25.40**	**25.24**	**33.70**	25.75
	Hunspell	23.26	17.19	17.76	31.20	**31.17**
ACC (entities)	Natas	6.23	10.41	10.93	18.77	7.07
	Hunspell	**43.93**	**40.44**	**41.58**	**47.78**	**50.38**

(a) English CoNLL 2003.

Measure	Method	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$	Typos
TER	Original	23.31	22.12	20.38	5.83	15.22
	Natas	**17.76**	**17.46**	**16.23**	**4.21**	11.68
	Hunspell	19.14	19.74	18.09	4.75	**11.22**
ACC	Natas	**23.82**	**21.05**	**20.36**	**27.75**	23.27
	Hunspell	17.90	10.74	11.20	18.59	**26.49**

(b) UD English EWT.

Table 6: Token Error Rates (TER) and the correction accuracies (ACC) of Natas and Hunspell on the test sets of our noisy sequence labeling data sets. All values are percentages. **Bold** values represent the lowest TER and the highest ACC.

Noisy Token	Error-Free Token	Class Label
`No`	`No`	`O`
`nzw`	`new`	`O`
`fixtuvzs`	`fixtures`	`O`
`reported`	`reported`	`O`
`from`	`from`	`O`
`New`	`New`	`B-LOC`
`Vork`	`York`	`I-LOC`
`.`	`.`	`O`

Table 7: Example of a sentence from the noisy CoNLL 2003 data set. The first and the second column contains the noisy and the error-free tokens, respectively. The third column denotes the class label in BIO format.

and achieved 96.9% accuracy on the validation set of 5000 sentences.

Learnable Parameters The number of parameters in our sequence labeling models was constant among different models, as we used the same architecture in all experiments. The number of all model parameters was 60.3 million (including embeddings that were fixed during the training), and the number of all trainable parameters was 25.5 million. Moreover, all our seq2seq error generation and correction models had about 7.7 million parameters.

Parameter	Description	Value
`-share_vocab`	Share source and target vocabulary	True
`-share_embeddings`	Share the embeddings between encoder and decoder	True
`-word_vec_size`	Word embedding size	25
`-src_seq_length`	Max. source sequence length	1000
`-tgt_seq_length`	Max. target sequence length	1000
`-encoder_type`	Type of encoder layer	brnn
`-learning_rate`	Starting learning rate	1.0

Table 8: The hyper-parameters of the OpenNMT toolkit used to train our seq2seq error generation models.

Training set size	Validation set size	Validation accuracy	
		token-level	sentence-level
10^7	5000	98.3%	95.7%
10^6	5000	95.4%	94.9%
10^5	5000	95.1%	95.3%
10^4	1000	94.6%	90.1%
10^3	100	93.3%	91.6%

Table 9: Validation accuracy of the seq2seq models for error generation. We trained both the token-level and the sentence-level variants. The first and the second column show the number of parallel sentences used for training and validation, respectively.

Average Runtime The evaluation of the complete test set took 7 and 10 seconds on average in the case of UD English EWT and English CoNLL 2003, respectively. The runtime did not depend on the training method that was used. Nevertheless, when we employed the correction method, the runtime was significantly lengthened, e.g., it took almost 3 minutes to evaluate a model that employed the Natas correction method on English CoNLL 2003.

Computing Architecture The evaluation was performed on a workstation equipped with an Intel Xeon CPU with 10 cores and an Nvidia Quadro RTX 6000 graphics card with 24GB of memory.

Spatial Dependency Parsing for Semi-Structured Document Information Extraction

Wonseok Hwang **Jinyeong Yim** **Seunghyun Park** **Sohee Yang** **Minjoon Seo**
Clova AI, NAVER Corp.
`{wonseok.hwang,jinyeong.yim,seung.park,sh.yang,minjoon.seo}@navercorp.com`

Abstract

Information Extraction (IE) for semi-structured document images is often approached as a sequence tagging problem by classifying each recognized input token into one of the IOB (Inside, Outside, and Beginning) categories. However, such problem setup has two inherent limitations that (1) it cannot easily handle complex spatial relationships and (2) it is not suitable for highly structured information, which are nevertheless frequently observed in real-world document images. To tackle these issues, we first formulate the IE task as *spatial dependency parsing* problem that focuses on the relationship among text tokens in the documents. Under this setup, we then propose SPADE♤ (SPAtial DEpendency parser) that models highly complex spatial relationships and an arbitrary number of information layers in the documents in an end-to-end manner. We evaluate it on various kinds of documents such as receipts, name cards, forms, and invoices, and show that it achieves a similar or better performance compared to strong baselines including BERT-based IOB taggger.

1 Introduction

Document information extraction (IE) is the task of mapping each document to a structured form that is consistent with the target ontology (e.g., database schema), which has become an increasingly important task in both research community and industry. In this paper, we are particularly interested in information extraction from real-world, semi-structured document images, such as invoices, receipts, and name cards, where we assume Optical Character Recognition (OCR, i.e. detecting the locations of the text tokens if the input is an image) has been already applied. Previous approaches for semi-structured document IE often assume as if the input is a one-dimensional sequence and formulate the task as an IOB (Inside Outside Beginning) tagging problem. In this setup, the tokens in the document (either obtained through an OCR engine or trivially parsed from a web page or pdf) are first serialized, and then an independent tagging model classifies each of the flattened lists into one of the pre-defined IOB categories (Ramshaw and Marcus, 1995; Palm et al., 2017). While effective for relatively simple documents, their broader application in the real world is still challenging because (1) semi-structured documents often exhibit a complex layout where the serialization algorithm is non-trivial, and (2) sequence tagging is inherently not effective for encoding multi-layer hierarchical information such as the menu tree in receipts (Fig. 1c).

To overcome these limitations, we propose SPADE♤ (SPAtial DEpendency parser), an end-to-end, serializer-free model that is capable of extracting hierarchical information from complex documents. Rather than explicitly dividing the original problem into two independent subtasks of serialization and tagging, our model tackles the problem in an end-to-end manner by creating a directed relation graph of the tokens in the document (Fig. 1). In contrast to traditional dependency parsing, which parses the dependency structure in purely (one-dimensional) linguistic space, our approach leverages both linguistic and (two-dimensional) *spatial* information to parse the dependency.

We evaluate SPADE♤ on eight document IE datasets created from real-world document images, including invoices, name cards, forms, and receipts, with the varying complexity of information structure. In all of the datasets, our model shows a similar or better accuracy than strong baselines including BERT-based IOB taggers, and particularly outstands in documents with complex layouts (Table 3). These results demonstrate the effectiveness

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 330–343
August 1–6, 2021.

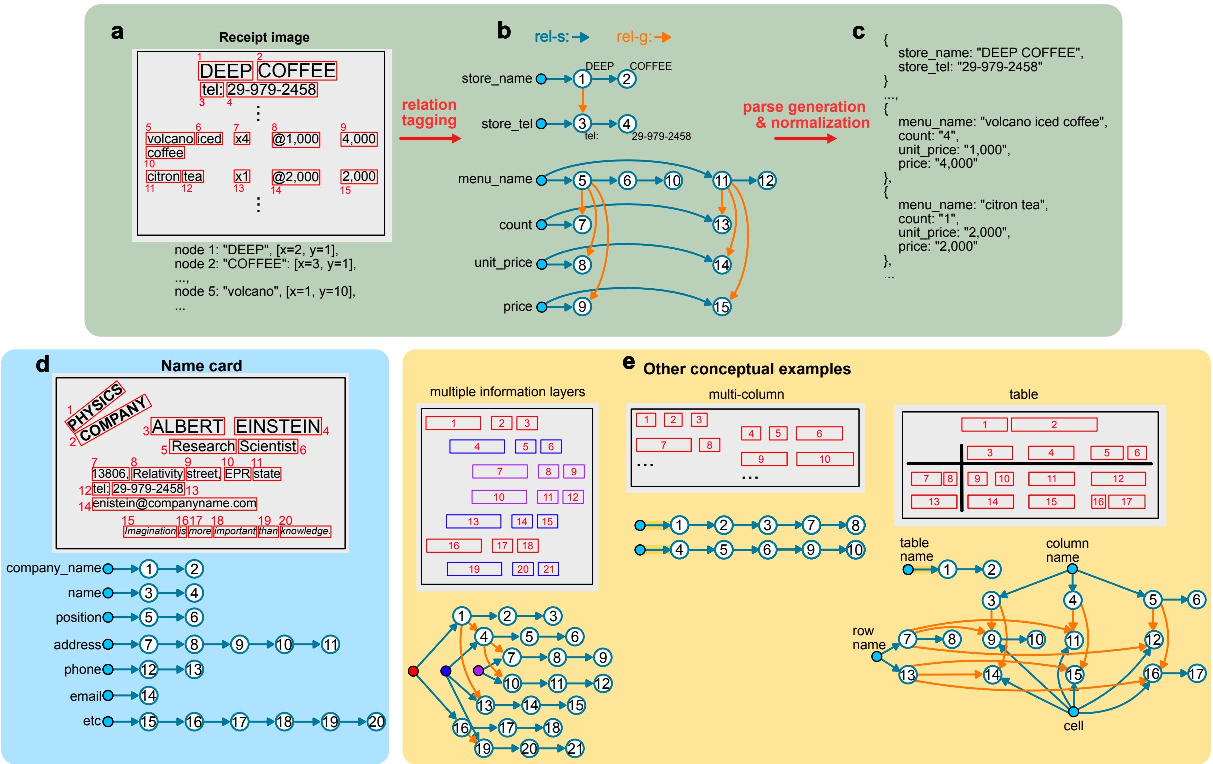

Figure 1: The illustration of spatial dependency parsing problem. Receipt parsing is explained in detail with three subfigures: (a) first, text tokens and their coordinates are extracted from OCR; (b) next, the relations between tokens are classified into two types: `rel-s` for serialization and information type (field) classification, and `rel-g` for inter-grouping between fields (the numbers inside of circles in (b) indicates the box numbers in (a)); (c) the final parse is generated by decoding the graph. (d) A sample name card and its spatial dependency parse. (e) Other conceptual examples showing the versatility of the spatial dependency parsing approach for document IE.

of our end-to-end, graph-based paradigm over the existing sequential tagging approaches.

In short, our contributions are threefold. (1) We present a novel view that information extraction for semi-structured documents can be formulated as a dependency parsing problem in two-dimensional space. (2) We propose SPADE♤ for spatial dependency parsing, which is capable of efficiently constructing a directed semantic graph of text tokens in semi-structured documents.[1] (3) SPADE♤ achieves a similar or better accuracy than the previous state of the art or strong BERT-based baselines in eight document IE datasets.

2 Related Work

The recent surge of interest in automatic information extraction from semi-structued documents are well reflected in their increased number of publication record from both research community and industry (Katti et al., 2018; Qian et al., 2019; Liu et al., 2019; Zhao et al., 2019; Denk and Reisswig, 2019; Hwang et al., 2019; Park et al., 2019; Xu et al., 2019; Jaume et al., 2019; Zhong et al., 2019; Rausch et al., 2019; Yu et al., 2020; Wei et al., 2020; Majumder et al., 2020; Lockard et al., 2020; Garncarek et al., 2020; Lin et al., 2020; Xu et al., 2020; Powalski et al., 2021; Wang et al., 2021; Hong et al., 2021; Hwang et al., 2021). Below, we summarize some of closely related works published before the major development of SPADE♤.

Serialized IE Previous semi-structured document information extraction (IE) methods often require the input text boxes (obtained from OCR) to be serialized into a single flat sequence. Hwang et al. (2019) and Denk and Reisswig (2019) combine a manually engineered text serializer that turn the OCR text boxes into a sequence and a Transformer-based encoder, BERT (Devlin et al., 2018), that performs IOB tagging on the sequence or semantic segmentation from images. In contrast to SPADE♤, these models rely on the serialization of the tokens and thus it is difficult to flexibly apply them to documents with complex layouts such as multi-column or distorted documents. Xu et al. (2019) propose LayoutLM that jointly embeds the

---

[1] `https://github.com/clovaai/spade`

image segments, text tokens, and positions of the tokens in an image to make a pretrained model for document understanding. However, LayoutLM still requires a careful serialization of the tokens as it relies on the position embeddings of BERT. Also, it is only evaluated on classification for the downstream task.

Serializer-free IE Existing serializer-free methods mostly extract flat key-value pairs, as they still formulate the task as tagging the text tokens. They fundamentally differ from SPADE♤ which generates a structured output that captures full information hierarchy represented in the document. Chargrid (Katti et al., 2018) performs semantic segmentation on invoice images to extract target key-value pairs. Although Chargrid uses additional "bounding boxes" for inter-grouping of certain fields, the application to the documents that have more than two information hierarchy levels is non-trivial. Also, when fields that belong to the same group are remotely located, the bounding boxes may need to be modified to have a more complex geometrical shape to avoid overlap between the boxes.

Graph-based IE Liu et al. (2019); Qian et al. (2019); Wei et al. (2020); Yu et al. (2020) utilize a graph convolution network to contextualize the tokens in a document and a bidirectional LSTM with CRF to predict the IOB tags. However, the range of possible parse generations is limited as IOB tagging can be performed only within each OCR bounding box, ignoring inter-box relationship. On the contrary, SPADE♤ predicts both the intra-box relationship and the inter-box relationship by constructing a dependency graph among the tokens.

Lockard et al. (2019, 2020) also utilize a graph to extract semantic relation from semi-structrued web-page. The graph is constructed based on rules from "structured html DOM" and mainly used for information encoding. On the other hand SPADE♤ accepts "unstructured text distributed in 2D" and generates graphs as the result of decoding (in a data-driven way).

Dependency parsing Dependency parsing is the task of obtaining the syntactic or semantic structure of a sentence by defining the relationships between the words in the sentence (Zettlemoyer and Collins, 2012; Peng et al., 2017; Dozat and Manning, 2018). The relations are often expressed as directed, labeled arcs. In our work, we view the problem of information extraction for semi-structured documents as a *spatial* dependency parsing task such that two-dimensional spatial information is mainly considered. This setup enables SPADE♤ to flexibly handle documents with complicated layouts while representing the full information hierarchy.

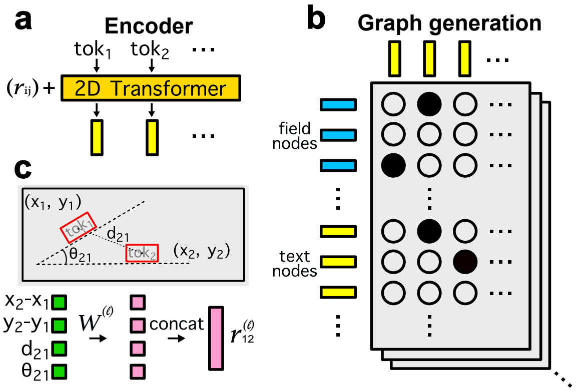

Figure 2: The illustration of SPADE♤. (a) Spatial text encoder contextualizes tokens using their relative spatial vectors $\{r_{ij}\}$ (Eq. 1). (b) The dependency graph is inferred by mapping the vector representations of each pair of tokens into a scalar value. The field embeddings are blue-colored. (c) At each encoder layer ℓ, r_{ij}^{ℓ} is prepared by concatenating four embedding vectors: relative coordinates, distance, and angles embeddings. W^{ℓ} stands for a linear projection.

3 Problem definition

In this section, we first describe the task of information extraction for semi-structured documents, and we briefly discuss how the task was approached in the past as a sequence tagging problem. Then we formulate it as a spatial dependency parsing problem. In Section 4, we show how we design our model for the newly formulated problem.

3.1 Semi-structured document IE

Document IE is often defined as the extraction of structured information (e.g. key-value pairs) in documents. For semi-structured documents, the task becomes more challenging, mainly due to two factors: (1) complex spatial layout and (2) hierarchical information structure. In the simplest case, both of the two factors are minimally present, where the text is strictly a linear sequence, and the desired output is simply a list of fields, similar to Named Entity Recognition (NER) task. However, the problem becomes more difficult when at least one of the factors is significant. In name cards, spatial relationship can be tricky; Fig. 1d shows an example where a naïve left-to-right serialization would fail because the `company_name` ("Physics Company") is tilted. In receipts, their hierarchical information structure complicates the problem. For example, in Fig. 1a), words "volcano" (box 5), "iced" (box 6), and "coffee" (box 10) together form a single field `menu_name`, and the field constitutes another group in the second hierarchical layer with the `count` field (box7), `unit_price` field (box 8), and `price` field (box 9). Other conceptual examples are shown in Fig. 1e); documents that have triple information layers (left), multiple columns (middle), and a table (right).

3.2 Previous formulation: Sequence tagging

As mentioned, IOB sequence tagging is appropriate for document IE when the layout and the information structure are simple (Ramshaw and Marcus, 1995; Lample et al., 2016; Chiu and Nichols, 2016; Ma and Hovy, 2020). When one of the factors is present, however, one has to adopt an ad-hoc solution to detour the inherent limitation of IOB.

In the case of complex spatial relationship (e.g., name card), an advanced, dedicated serialization method can be considered. However, it may require layout-specific manual engineering, which becomes more difficult for documents such as name cards that exhibit diverse layouts.

In the case of complex information structure (e.g., receipt), one can consider augmenting each IOB tag with higher-layer information. For instance, in a typical IOB setting, the `menu_name` field will require two tags, namely `menu_name_B` and `menu_name_I`. To model the second layer information (inter-grouping of fields), `menu_name_B` can be augmented into two, namely `B2_menu_name_B`, `I2_menu_name_B`, where `B2` and `I2` indicate the beginning and the inside of the hierarchy's second layer. While effective for some applications, this method would not generalize well to an arbitrary depth as it requires more tags for each additional layer.

3.3 Our formulation: Spatial dependency parsing

To better model spatial relationship and hierarchical information structure in semi-structured documents, we formulate the IE problem as "spatial dependency parsing" task by constructing a dependency graph with tokens and fields as the graph nodes (node per token and field type). This is demonstrated in Fig. 1, where empty blue circles are text nodes, and filled blue circles are field nodes.

Although the spatial layout of semi-structured documents is diverse, it can be considered as the realization of mainly two abstract properties between each pair of nodes, (1) `rel-s` for the ordering and grouping of tokens belonging to the same information category (blue arrows in Fig. 1b), and (2) `rel-g` for the inter-group relation between grouped tokens or groups (orange arrows in the same figure). Connecting a field node to a text node indicates that the text is classified into the field. For example, "volcano iced coffee" in Fig. 1a) is classified as a menu name by being attached to the `menu_name` field node with blue arrows, and it is connected with "x4", "@1,000", and "4,000" with orange arrows to indicate the hierarchical information among the groups. The dependency graphs of name cards and other conceptual examples are also shown in Fig. 1d and e.

4 Model

To perform the spatial dependency parsing task introduced in the previous section in an end-to-end fashion, we propose SPADE♤ that consists of (1) spatial text encoder, (2) graph generator, and (3) graph decoder. Spatial text encoder and graph generator are trained jointly. Graph decoder is a deterministic function (without trainable parameters) that maps the graph to a valid parse of the output structure.

4.1 Spatial text encoder

Spatial text encoder is based on 2D Transformer architecture. Unlike the original Transformer (Vaswani et al., 2017), there is no order among the input tokens, making the model invariant under the permutation of the input tokens. Inspired by Transformer XL (Dai et al., 2019), the attention weights (between each key and query vector) is computed by

$$q_i^T k_j + q_i^T r_{ij} + (b_i^{key})^T k_j + (b_i^{rel})^T r_{ij} \qquad (1)$$

where q_i is the query vector of the i-th input token, k_j is the key vector of the j-th input token, r_{ij} is the relative spatial vector of the j-th token with respect to the i-th token, and $b_i^{key|rel}$ is a bias vector. In (original) Transformer, only the first term of Equation 1 is used.

The relative spatial vector r_{ij} is constructed as follows (Fig. 2c). First, the relative coordinates between each pair of tokens are computed.[2] Next, the coordinates are quantized into integers and embedded using `sin` and `cos` functions (Vaswani et al., 2017). The physical distance and the relative angle between each pair of the tokens are also embedded in a similar way. Finally, the four embedding vectors are linearly projected (with a trainable projection matrix) and concatenated at each encoder layer.

4.2 Graph generator

As discussed in Section 3.3 and shown in Fig. 1, every token corresponds to a node and each pair of the nodes forms one of the two relations (or no relation): (1) `rel-s` for serializing tokens within the same field, and (2) `rel-g` for inter-grouping between fields. The dependency graph can be represented by using a binary matrix $M^{(r)}$ for each relation type r (Fig. 2b) where $M_{ij}^{(r)} = 1$ if their exists a directed edge from the i-th token to the j-th token and 0 otherwise. Each $M^{(r)}$ consists of $n_{\text{field}} + n_{\text{text}}$ number of rows and n_{text} number of columns where n_{field} and n_{text} represent the number of field types and the number of tokens, respectively. The graph generation task now becomes predicting the binary matrix.

We obtain $M^{(r)}$ as follows. The probability that there exists a directed edge $i \xrightarrow{r} j$ is computed by

$$
\begin{aligned}
h_i^{(r)} &= \begin{cases} u_i^{(\text{field})}, & \text{for } i \leq n_{\text{field}} \\ \mathcal{W}_h^{(r)} v_i & \text{otherwise} \end{cases} \\
d_j^{(r)} &= \mathcal{W}_d^{(r)} v_j \\
s_{0,ij}^{(r)} &= (h_i^{(r)})^T \mathcal{W}_0^{(r)} d_j^{(r)} \\
s_{1,ij}^{(r)} &= (h_i^{(r)})^T \mathcal{W}_1^{(r)} d_j^{(r)} \\
p_{ij}^{(r)} &= \frac{\exp(s_{1,ij}^{(r)})}{\exp(s_{0,ij}^{(r)}) + \exp(s_{1,ij}^{(r)})},
\end{aligned}
\qquad (2)
$$

where $u_i^{(\text{field})}$ represents the trainable embedding vector of the i-th field type node (filled blue circles in Fig. 1), $\{v_i\}$ is a set of vectors of contextualized tokens from the enoder, $\mathcal{W}$ stands for affine transformation, h is the embedding vector of the head token, and d is that of the dependent token.

$M_{ij}^{(r)}$ is obtained by binarizing $p_{ij}^{(r)}$ as follows.

$$
M_{ij}^{(r)}(p_{ij}^{(r)}) = \begin{cases} 1 \text{ for } (r{=}\texttt{s}, i \text{ is field type node}, p_{ij}^{(r)} \geq p_{th}) \\ 1 \text{ for } (r{=}\texttt{s}, i \text{ is text node}, p_{ij}^{(r)} \geq p_{th}, \\ \qquad j = \arg\max_k p_{ik}^{(r)}) \\ 1, \text{ for } (r{=}\texttt{g}, i \text{ is text node}, p_{ij}^{(r)} \geq p_{th}) \\ 0, \text{ otherwise.} \end{cases} \qquad (3)
$$

The recall rate of edges can be controlled by varying the threshold value p_{th}. Here, we set $p_{th} = 0.5$.

Tail collision avoidance algorithm Each node in spatial dependency graphs has a single incoming edge per relation except some special documents such as table (Fig. 1e). Based on this property, we apply the following simple yet powerful tail collision avoidance algorithm: (1) at each tail node having multiple incoming edges, all edges are trimmed except the one with the highest linking probability; (2) at each head node of the trimmed edges, the new tail node is found by drawing the next probable edge whose probability is larger than p_{th} and belongs to the top three; (3) go back to Step 1 and repeat the routine until the process becomes self-consistent or the max iteration limit is reached (set to 20 in this paper). The algorithm prevents loops and token redundancy in parses.

4.3 Graph decoder

We decode the generated graph into the final parse through the following three stages: (1) `SEEDING`, (2) `SERIALIZATION`, and (3) `GROUPING` (Table 1). In `SEEDING`, field type nodes (filled circles in Fig. 1) are linked to multiple text nodes (seeds) by `rel-s`. In `SERIALIZATION`, each seed node found in the previous stage generates a directed edge (`rel-s`) to the next text node (i.e. serialization) recursively until there is no further node to be linked. Finally, in `GROUPING`, the serialized texts are grouped iteratively, constructing information layers from the top to the bottom. The total number of iterations is equal to "the number of information layers-1". To group texts using directed edges, we define a special representative field for each information layer. Then, the first token of the representative field generates directed edges to the first token of other fields that belong to the same group using `rel-g` (for example, `menu_name` ("volcano iced coffee") in Fig. 1a) generates directed edges to other member fields (`count` ("x4"), `unit_price` ("@1,000") and `price` ("4,000")).

---

[2] For example, if "token1" is at ($x_1 = 1, y_1 = 10$) and "token2" is at ($x_2 = 3, y_2 = 4$), the relative coordinate of "token2" with respect to "token1" is $(x_2', y_2') = (3-1, 4-10) = (2, -6)$.

The process generates an *arborescence*[3] for each field (`rel-s`) and group (`rel-g`). The resulting set of graphs has a one-to-one correspondence with the parse through detokenization. The use of beam search in `SERIALIZATION` does not introduce noticeable difference in `rel-s` probably due to the short decoding length of the graph (mostly less than 30). The development of a more advanced decoding algorithm that generates globally optimal multiple arborescences remains as future work.

Although undirected edges can be employed for the inter-grouping of fields, the use of directed edges has the following merits: (1) an arbitrary depth of information hierarchy can be described without increasing the number of relation types (Fig. 1e) under a unified framework and (2) a parse can be generated in a straightforward manner by iteratively selecting dependent nodes.

Table 1: A formal description of the parse decoding process. `s` and `g` stand for `rel-s` and `rel-g` respectively.

Action	Input node	Graph at time $t+1$
`INITIALIZATION`		$G_{t=0}$ = empty set
`SEEDING`(μ)	$\mu \in$ field nodes	$G^{(\text{seed})} = \{\mu \xrightarrow{\text{s}} j \mid M^{(\text{s})}_{\mu j} = 1\}$
`SERIALIZATION`(i)	$i \in G^{(\text{seed})} \cup G_t$	$G_{t+1} = G_t \cup \{i \xrightarrow{\text{s}} j \mid M^{(\text{s})}_{ij} = 1\}$
`GROUPING`(i)	$i \in G^{(\text{seed})} \cup G_t$ i linked to representer fields	$G_{t+1} = G_t \cup \{i \xrightarrow{\text{g}} j \mid M^{(\text{g})}_{ij} = 1\}$
`MERGE`		$G_t = G_t \cup G^{(\text{seed})}$

5 Experimental Setup

5.1 Optical character recognition

To extract the visually embedded texts from an image, we use our in-house OCR system that consists of CRAFT text detector (Baek et al., 2019b) and Comb.best text recognizer (Baek et al., 2019a). The OCR models are finetuned on each of the document IE datasets. The output tokens and their spatial information on the image are used as the inputs to SPADE♤.

5.2 Training

We use 12 layers of 2D Transformer encoder (Section 4.1). The parameters are initialized from `bert-multilingual` (Devlin et al., 2018) [4]. ADAM optimizer (Kingma and Ba, 2015) is used with the following learning rates: 1e-5 for the encoder, 1e-4 for the graph generator, and 2e-5 for S+BERT+IOB$_2$ and S$_{\text{ADV}}$+BERT+IOB$_2$. The decay rates are set to $\beta_1 = 0.9, \beta_2 = 0.999$. The batch size is chosen between 4 and 12. SPADE♤ is trained by using one to eight NVIDIA V100 or P40 GPUs for two to seven days, depending on the tasks. The dev sets are used to pick the best model except FUNSD task in which the model is trained in two steps. First, the 25 examples from training set are sampled and used for a model validation. Next, the model is further trained using entire training set and stopped after 1000 epochs. The training dataset is augmented by randomly rotating the text coordinates by a degree of -10° to +10°, (2) by distorting the whole coordinates randomly using a trigonometric function, and (3) by randomly deleting or inserting a single token with 3.3% probability each. Also, 1–2 random tokens from training is attached at the end of the text segments from OCR bounding box with 1.7% probability each. In NAMECARD task, the tokens are not augmented. The identical augmentation algorithm are applied to S+BERT+IOB$_2$, S$_{\text{ADV}}$+BERT+IOB$_2$ and SPADE♤.

5.3 Evaluation metric

To evaluate the predicted parses that consist of hierarchically organized key-value pairs (e.g. Fig. 3, Fig. 4, 5, 6 in Appendix) we use F_1 score based on exact match. First the group of key-value pairs between predictions and ground truth (gt) are matched based on their string edit distance. Each key-value pairs in the predicted parse is counted as true positive if same key-value pair exists within the corresponding group in gt. Otherwise it is counted as false positive. The unmatched key-value pairs in gt are counted as false negative. The accuracy of dependency parsing is evaluated by computing F_1 of predicted edges. For FUNSD dataset, entity labeling and entity linking scores are computed following the original paper (Jaume et al., 2019). See Appendix A.2 for more details.

5.4 Data statistics

We summarize the data statistics in Table 2, 6. The property of each dataset and their collection process is described in Appendix A.1.

6 Experimental Results

The main focus of SPADE♤ is to handle the two challenging factors of semi-structured document information extraction—complex spatial relationships and highly structured information—in a gen-

---

[3] A directed graph in which, for a vertex u called the root and any other vertex v, there is exactly one directed path from u to v (Excerpted from Wikipedia)

[4] https://github.com/huggingface/transformers

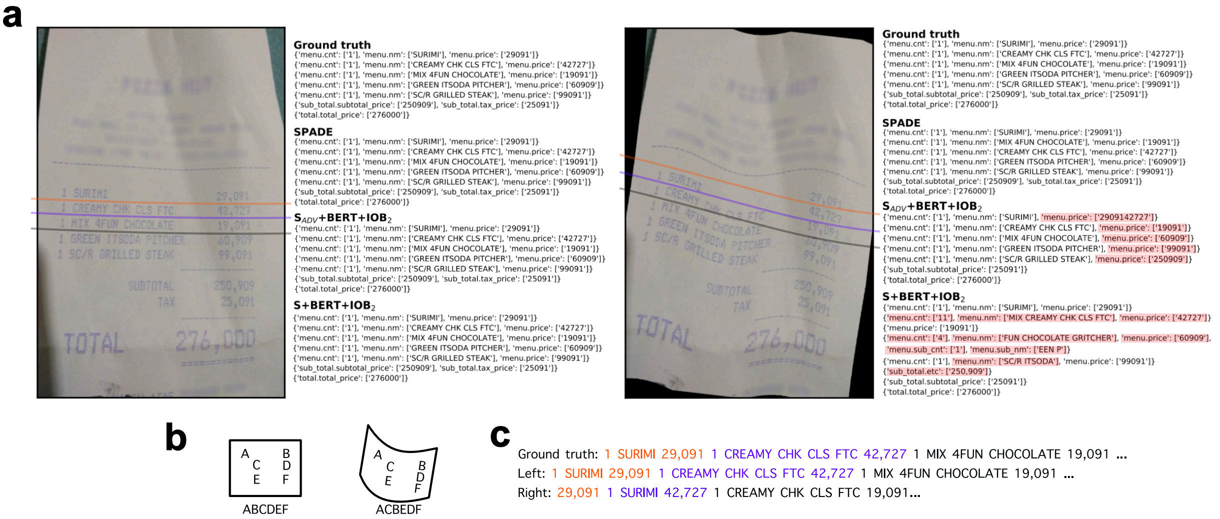

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['99091']}
{'sub_total.subtotal_price': ['250909'], 'sub_total.tax_price': ['25091']}
{'total.total_price': ['276000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['99091']}
{'sub_total.subtotal_price': ['250909'], 'sub_total.tax_price': ['25091']}
{'total.total_price': ['276000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['99091']}
{'sub_total.subtotal_price': ['250909'], 'sub_total.tax_price': ['25091']}
{'total.total_price': ['276000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['99091']}
{'sub_total.subtotal_price': ['250909'], 'sub_total.tax_price': ['25091']}
{'total.total_price': ['276000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['99091']}
{'sub_total.subtotal_price': ['250909'], 'sub_total.tax_price': ['25091']}
{'total.total_price': ['276000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['99091']}
{'sub_total.subtotal_price': ['250909'], 'sub_total.tax_price': ['25091']}
{'total.total_price': ['276000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['2909142727']}
{'menu.cnt': ['1'], 'menu.nm': ['CREAMY CHK CLS FTC'], 'menu.price': ['19091']}
{'menu.cnt': ['1'], 'menu.nm': ['MIX 4FUN CHOCOLATE'], 'menu.price': ['60909']}
{'menu.cnt': ['1'], 'menu.nm': ['GREEN ITSODA PITCHER'], 'menu.price': ['99091']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R GRILLED STEAK'], 'menu.price': ['250909']}
{'sub_total.subtotal_price': ['25091']}
{'total.total_price': ['276000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['SURIMI'], 'menu.price': ['29091']}
{'menu.cnt': ['11'], 'menu.nm': ['MIX CREAMY CHK CLS FTC'], 'menu.price': ['42727']}
{'menu.price': ['19091']}
{'menu.cnt': ['4'], 'menu.nm': ['FUN CHOCOLATE GRITCHER'], 'menu.price': ['60909'],
 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['EEN P']}
{'menu.cnt': ['1'], 'menu.nm': ['SC/R ITSODA'], 'menu.price': ['99091']}
{'sub_total.etc': ['250,909']}
{'sub_total.subtotal_price': ['25091']}
{'total.total_price': ['276000']}
```

```
Ground truth: 1 SURIMI 29,091 1 CREAMY CHK CLS FTC 42,727 1 MIX 4FUN CHOCOLATE 19,091 ...
Left: 1 SURIMI 29,091 1 CREAMY CHK CLS FTC 42,727 1 MIX 4FUN CHOCOLATE 19,091 ...
Right: 29,091 1 SURIMI 42,727 1 CREAMY CHK CLS FTC 19,091...
```

Figure 3: Examples from CORD (left) and CORD++(right) dev sets. (a) Parses are shown in grouped key-value format with the errors in red. (b) The illustration of serialization error. (c) The input tokens serialized by S_{adv}.

Table 2: The dataset properties.

Dataset	Lang.	Abbr.	# of field types	# of examples (train:dev:test)	# of fields	Mean # of text nodes	Depth	Layout complexity
CORD	IDN	co	30	800:100:100	13030	62.3	2	low
CORD+	IDN	co+	"	"	"	62.3	2	high
CORD++	IDN	co++	"	"	"	62.3	2	high
CORD-M	IDN	co-m	"	400:50:50	"	124.6	3	low
Receipt-idn	IDN	ri	50	9508:458:450	209728	209	2	low
namecard	JPN	nc	12	22076:256:100	231528	19.4	1	high
Invoice	JPN	inv	62	896:79:83	37115	412	2	high
FUNSD[a]	ENG	fu	4	149:50	9743	179	3	high

[a] The statistics are from Jaume et al. (2019).

Table 3: Parse prediction accuracy. The datasets are referred by their abbreviations in Table. 2. ΔF_1 indicates the difference between SPADE♤ (2nd row) and s_{adv}+bert+iob$_2$ (4th row).

	test (+oracle†)				test					
Model	co	ri	nc	inv	co	co+	co++	ri	nc	inv
SPADE♤ w/o tca	91.5	92.7	94.0	87.4	87.4	86.1	82.6	88.5	91.1	84.5
SPADE♤	92.5	93.3	94.3	88.1	88.2	87.4	83.1	89.1	91.6	85.0
s+bert+iob$_2$	92.4*	93.3*	-	-	90.1	74.0	52.0	88.1	-	-
s_{adv}+bert+iob$_2$	92.5*	93.4*	94.4*	84.9*	90.1	85.4	64.8	89.3	90.5	83.1
ΔF_1	0	-0.1	-0.1	+3.2	-1.9	+2.0	+18.3	-0.2	+1.1	+1.9
UB-flat	58.1	65.4	100	83.2	-	-	-	-	-	-

† The input tokens are recognized by human annotators.
* The input tokens are line-grouped by human annotators.

eralizable way. We first show that our model can handle hierarchical structure in documents by evaluating the model on two datasets CORD (Park et al., 2019) and Receipt-idn that consist of (Indonesian) receipt images. We then show SPADE♤ can perform well on tasks that require modeling the complex spatial relationship in documents by reporting the performance on name card IE where the spatial layout is more complex than receipts. Then the evaluation on the invoice dataset shows the advantage of SPADE♤ when both of the two challenging factors are simultaneously present. Finally, we show that SPADE♤ can handle even more types of documents by evaluating the model on a form understanding dataset, FUNSD (Jaume et al., 2019). Table 3 summarizes the performance of several baseline models and SPADE♤ in various semi-structured document information extraction tasks.

Handling hierarchical structure in documents CORD consists of receipt images without creases or warping. SPADE♤ initially achieves 91.5% and 87.4% in F_1 with and without the oracle (ground truth OCR results), respectively (Table 3, 1st row, co). Their dependency parsing score is also shown in Table 7 in Appendix (1st panel, co). To push the performance further, we notice that individual text nodes have a single incoming edge for each relation except in special documents like table (Fig. 1). Using this property, we integrate Tail Collision Avoidance algorithm (tca) that iteratively trims the tail-sharing-edges and generate new edges until the process becomes self-consistent (Section 4.2). F_1 increases by +1.0% and +0.8% with and without the oracle upon the integration (2nd row, co).

Importance of generating hierarchical structure in receipt IE In receipt IE task, the intergrouping of fields is critical due to multiple appearance of same field types such as `menu_name` and `price` (Fig. 3a). Without the field grouping, the maximum achievable score is 58.1 F_1 (Table 3, 6th row, UB-flat). Generating hierarchical parses from the semi-structured documents is relatively new and thus the direct comparison to previous state-of-the-art methods are not feasible without considerable modification. General confidential issue related to industrial documents and multi-lingual properties of our task also hinder the comparison. In this regard, we build our own baselines consisting of the manually engineered serializer and BERT-based double IOB taggers (S+BERT+IOB$_2$[5]).

BERT-tagger The serializer generates pseudo-1D-text from the input tokens distributed in 2D and groups them line-by-line based on their height differences. BERT+IOB$_2$ predicts the boundary between the fields and between the groups of the fields (see Section 3.2 for the detail). In CORD, S+BERT+IOB$_2$ shows comparable performance with SPADE♤ with the oracle (-0.1 F_1) but shows +1.9 F_1 on the test set (2nd and 3rd rows, CO). The relatively lower score of SPADE♤ on the test set may originate from the small size of the training set (800, Table 2) as SPADE♤ needs to handle the text serialization in a data-driven way. Indeed, when both models are trained using RECEIPT-IDN that consists of 9508 training examples, SPADE♤ outperforms by +1.0 F_1 on the test set (2nd and 3rd rows, RECEIPT-IDN).

Inflexibility of tagging model in handling complex spatial relationships Next, we prepare CORD+ and CORD++, which are more challenging setups where the images are warped or tilted as often seen in real-world applications (Fig. 3). SPADE♤ significantly outperforms S+BERT+IOB$_2$ (+13.4% F_1 in CORD+, +31.1% F_1.b in CORD++). This is due to the failure in the serialization in S+BERT+IOB$_2$ resulting in line-mixing (Fig. 3b, c and Fig. 5, 6 in Appendix). To understand how much improvement can be achieved through further manual engineering, we prepare S$_{ADV}$+BERT+IOB$_2$ which is equipped with the advanced serializer where polynomial fitting is employed to group tokens placed on curvy line. The result shows although there is a large improvement in CORD+ and CORD++ task compared to S+BERT+IOB$_2$, SPADE♤ still shows the better performance (+2.0% in CORD+, +18.3% in CORD++, 1st and 4th rows). This shows the limitation of a serializer-based method that it cannot be easily generalized to handle document images in wild and the performance can be bottlenecked by the serialization step regardless of how advanced tagging models are. The competent performance of SPADE♤ on CORD-M, a dataset generated by concatenating two receipt images from CORD into a single image (Fig. 4 in Appendix), further highlights the flexibility of SPADE♤.

Handling documents having complex layout We further evaluate SPADE♤ on name card IE task. Unlike receipts, no inter-grouping between fields is necessary for name card IE. However, name cards often have a complex layout such as non-horizontal alignment of text or multi column even without tilting and warping (Fig. 1d). Our model achieves +1.1% F_1 compared to S$_{ADV}$+BERT+IOB$_2$ on the test set (Table 3, NC).

Handling documents having both hierarchical structure and complex layout To fully explore the capability of SPADE♤, we further evaluate the model on invoice IE task. Typical invoices have a hierarchical structure where some fields need to be grouped together, such as `item_name`, `count`, and `price` that correspond to one same item. In addition, invoices also have a relatively complex layout, having multiple tables or columns. SPADE♤ achieves +1.9 F_1 compared to S$_{ADV}$+BERT+IOB$_2$ (Table. 3, INV).

Handling general documents In order to see if SPADE♤ can handle more general kinds of documents, we use the FUNSD form understanding dataset (Jaume et al., 2019) where document IE is performed under a more abstract setting by finding general key-value pairs and their inter-grouping (Section A.1.6). The performance is measured on two OCR-independent subtasks (Jaume et al., 2019): (1) "entity-labeling (ELB)" which predicts the information category of the serialized words, and (2) "entity-linking (ELK)" which measures the score for key-value pair link prediction. The evaluation reveals that SPADE♤ achieves the state of the art on ELK, outperforming the previous baseline by 37.3% F_1 (Table 4, rightmost column). In ELB, SPADE♤ achieves +11.5% F_1 absolute improvement with respect to BERT-Base Tagger. Both models use BERT-Base as a backbone. Although the F_1 scores of LayoutLM are higher than our model, their contributions are orthogonal to ours since they focus on making a better pretrained model. Also, it cannot perform ELK. We emphasize that SPADE♤ solves the three subtasks–ELB, ELK, and word serialization–simultaneously, while other tagger models need to use the perfectly serialized input text and solve only entity labeling. The stable performance of SPADE♤ over randomly rotated documents (ELB-R) or shuffled tokens (ELB-S) supports this highlighting the merit of the serializer-

---

[5] S stands for the serializer.

free architecture.

Table 4: F_1 scores for two FUNSD subtasks: entity labeling (ELB, ELB-R, and ELB-S) and entity linking (ELK). "Need `S`" means the input tokens should be serialized. "# of `D`" indicates the number of documents used for layout pretraining.

Model	Need `S`	# of `D`	ELB	ELB-R	ELB-S	ELK
Baseline[a]	○	0	57	-	-	4
BERT-Base Tagger*	○	0	60.1	43.9 (-16.2)	42.5 (-17.6)	-
BERT-Large Tagger*	○	0	64.6	47.6 (-17.0)	42.7 (-21.9)	-
LayoutLM-Base Tagger[b]	○	500K	69.9	-	-	-
LayoutLM-Base Tagger*	○	11M	**78.9**	**72.5** (-6.4)	70.2 (-8.7)	-
SPADE♤†	×	0	71.6	70.5 (-1.1)	**72.0** (+0.4)$	41.3

[a] Jaume et al. (2019). [b] From Xu et al. (2019).

* The source code from https://github.com/microsoft/unilm/tree/master/layoutlm.

$ The separation of long input text (> 512) into multiple independent inputs introduces small difference in F_1.

†Five encoder layers are used for computational efficiency.

Ablation study We probe the role of each component of SPADE♤ via ablation study (Table 5). The performance drops dramatically upon the removal of the relative coordinate information of tokens in the self-attention layer, highlighting its importance in the serializer-free encoder (2nd row). When the absolute coordinates are used in the input instead of the relative coordinates, F_1 drops by 6.9% (3rd row). Finally, 2.6% drop in F_1 is observed upon the removal of the data augmentation during training (4th row).

Table 5: Ablation study on CORD dataset.

Model	F_1
SPADE♤†	84.5
(-) relative coordinate	10.5 (-74.0)
(-) relative coordinate (+) absolute coordinate	78.6 (-6.9)
(-) data augmentation	81.9 (-2.6)

† Five encoder layers are used for computational efficiency.

7 Conclusion

We present SPADE♤, a spatial dependency parser that can extract highly structured information from documents that have complex layouts. By formulating document IE as a spatial dependency graph construction problem, we provide a powerful unified framework that can extract hierarchical information without feature engineering. We empirically demonstrate the effectiveness of our model over various real-world documents—receipts, name cards, and invoices—and in a popular form understanding task.

Acknowledgments

We thank Geewook Kim for the critical comments on the manuscript and Teakgyu Hong and Sungrae Park for the helpful discussions on FUNSD experiments.

References

Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. 2019a. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *ICCV*.

Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019b. Character region awareness for text detection. In *ICCV*, pages 9365–9374.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*.

Timo I. Denk and Christian Reisswig. 2019. BERTgrid: Contextualized Embedding for 2D Document Representation and Understanding. *arXiv e-prints*, page arXiv:1909.04948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *ACL*.

Lukasz Garncarek, Rafal Powalski, Tomasz Stanislawek, Bartosz Topolski, Piotr Halama, and Filip Gralinski. 2020. LAMBERT: layout-aware language modeling using BERT for information extraction. *arXiv e-prints*.

Teakgyu Hong, DongHyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2021. BROS: A pre-trained language model for understanding texts in document.

Wonseok Hwang, Seonghyeon Kim, Jinyeong Yim, Minjoon Seo, Seunghyun Park, Sungrae Park, Junyeop Lee, Bado Lee, and Hwalsuk Lee. 2019. Post-ocr parsing: building simple and robust parser via bio tagging. In *Workshop on Document Intelligence at NeurIPS 2019*.

Wonseok Hwang, Hyunji Lee, Jinyeong Yim, Geewook Kim, and Minjoon Seo. 2021. Cost-effective end-to-end information extraction for semi-structured document images. *arXiv e-prints*, abs/2104.08041.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *ICDAR-OST*.

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2D documents. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Bill Yuchen Lin, Ying Sheng, Nguyen Vo, and Sandeep Tata. 2020. Freedom: A transferable neural architecture for structured information extraction on web documents. In *KDD*.

Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 2019. Graph convolution for multimodal information extraction from visually rich documents. In *NAACL*.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. OpenCeres: When open information extraction meets the semi-structured web. In *ACL*.

Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. In *ACL*.

Xuezhe Ma and Eduard Hovy. 2020. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Bodhisattwa Prasad Majumder, Navneet Potti, Sandeep Tata, James B. Wendt, Qi Zhao, and Marc Najork. 2020. Representation learning for information extraction from form-like documents. *ACL*.

Rasmus Berg Palm, Ole Winther, and Florian Laws. 2017. Cloudscan - A configuration-free invoice analysis system using recurrent neural networks. *CoRR*.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. 2019. Cord: A consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL*, 5:101–115.

Rafal Powalski, Lukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michal Pietruszka, and Gabriela Palka. 2021. Going full-tilt boogie on document understanding with text-image-layout transformer. *CoRR*, abs/2102.09550.

Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. 2019. GraphIE: A graph-based framework for information extraction. In *NAACL*.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Johannes Rausch, Octavio Martinez, Fabian Bissig, Ce Zhang, and Stefan Feuerriegel. 2019. Docparser: Hierarchical structure parsing of document renderings. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*.

Jiapeng Wang, Chongyu Liu, Lianwen Jin, Guozhi Tang, Jiaxin Zhang, Shuaitao Zhang, Qianying Wang, Yaqiang Wu, and Mingxiang Cai. 2021. Towards robust visual information extraction in real world: New dataset and novel solution. In *AAAI*.

Mengxi Wei, Yifan He, and Qiong Zhang. 2020. Robust Layout-aware IE for Visually Rich Documents with Pre-trained Language Models. In *SIGIR*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei A. F. Florêncio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2020. LayoutLMv2: Multi-modal pretraining for visually-rich document understanding. *arXiv e-prints*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2019. Layoutlm: Pretraining of text and layout for document image understanding. In *KDD*.

Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. 2020. PICK: Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks. In *ICPR*.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars.

Xiaohui Zhao, Zhuo Wu, and Xiaoguang Wang. 2019. CUTIE: learning to understand documents with convolutional universal text information extractor. *arXiv e-prints*.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno-Yepes. 2019. Image-based table recognition: data, model, and evaluation. *arXiv e-prints*.

A Appendices

A.1 Dataset

A.1.1 Dataset collection

The internal datasets Receipt-idn, namecard and Invoice are annotated by the crowd through an in-house web application following (Park et al., 2019; Hwang et al., 2019). First, each text segment is labeled (bounding box and the characters inside) for the OCR task. The text segments are further grouped according to their field types by the crowds. For Receipt-idn and Invoice, additional group-ids are annotated to each field for inter-grouping of them. The text segments placed on the same line are also annotated through row-ids. For quality assurance, the labeled documents are cross-inspected by the crowds.

A.1.2 CORD, CORD+, CORD++, and CORD-M for receipt IE

CORD and their variant consist of 30 information categories such as `menu_name`, `count`, `unit_price`, `price`, and `total_price` (Table 6). The fields are further grouped and forms the information layer at a higher level.

A.1.3 Receipt-idn for receipt IE

Receipt-idn is similar to CORD but includes more diverse information categories (50) such as `store_name`, `store_address`, and `payment_time` (Table 6).

A.1.4 namecard for name card IE

namecard consists of 12 field types, including `name`, `company_name`, `position`, and `address` (Table 6). The task requires grouping and ordering of tokens for each field. Although there is only a single information layer (field), the careful handling of complex spatial relations is required due to the large degree of freedom in the layout.

A.1.5 Invoice for invoice IE

Invoice consists of 62 information categories such as `item_name`, `count`, `price_with_tax`, `item price_without_tax`, `total_price`, `invoice_number`, `invoice_date`, `vendor_name`, and `vendor_address` (Table 6). Similar to receipts, their hierarchical information is represented via inter-field grouping.

A.1.6 FUNSD for general form understanding

FUNSD form understanding task consists of two sub tasks: entity labeling (ELB) and entity linking (ELK). In ELB, tokens are classifed into one of four fields–header, question, answer, and other–while doing serialization of tokens within each field. Both subtasks assume that the input tokens are perfectly serialized with no OCR error. To emphasize the importance of correct serialization in the real-world, we prepare two variant of ELB tasks: ELB-R and ELB-S. In ELB-R, the whole documents are randomly rotated by a degree of -20°–20° and the input tokens are serialized using rotated y-coordinates. In ELB-S task, the input tokens are randomly shuffled. In both tasks, the relative order of the input tokens within each field remain unchanged. In ELK task, tokens are linked based on their key-value relations (inter-grouping between fields). For example, each "header" is linked to the corresponding "question", and "question" is paired with the corresponding "answer".

Table 6: The representative fields of the datasets.

Dataset	representative fields and their numbers
CORD,CORD+, CORD++,CORD-M	`menu_name` (2572), `count` (2357), `unit_price` (737), `price` (2559), `total_price` (974)
Receipt-idn	`menu_name` (28832), `munu_count` (27132), `menu_unitprice` (11530), `menu_price` (28028), `total_price` (10284), `store_name` (9413), `payment_time` (9817)
namecard	`name` (25917), `company_name` (24386), `position` (22848), `address` (26018)
Invoice	`item_name` (2761), `count` (1950), `price_with_tax`(781), `price_without_tax` (2230), `total_price` (844), `invoice_number` (803), `invoice_date` (987), `vendor_name` (993), `vendor_address` (993),
FUNSD^a	`header` (563), `question` (4343), `answer` (3623), `other` (1214)

[a] From (Jaume et al., 2019).

A.2 Evaluation metric

During calculation of F_1 for parses, the difference between prediction and ground truth is not counted in `store_name`, `menu_name`, and `item_name` fields in receipt and invoice when the edit distance (ED) is less then 2 or when the ED/gt-string-length ≤ 0.4. Also, in Japanese documents, white spaces are ignored.

In the FUNSD form understanding task, we measure entity labeling (ELB) and entity linking (ELK) scores following (Jaume et al., 2019). ELB measures the field classification accuracy of already "perfectly" serialized tokens of each field (words group), whereas ELK measures the inter-grouping accuracy between word groups. As SPADE♤ does both the serialization of the fields and grouping between fields simultaneously, we do not feed the serialized tokens into SPADE♤ but only use the oracle information to indicate the first text node of each field from the predicted graph. These text nodes effectively represent entire fields and are used for the evaluation.

A.3 The score for the dependency relation prediction

Table 7: The score for the dependency relation prediction. `s` and `g` stand for `rel-s` and `rel-g`.

		Precision					Recall					F_1				
Model	`rel`	CO	RI	NC	INV	FU	CO	RI	NC	INV	FU	CO	RI	NC	INV	FU
♤ - TCA	s	96.4	97.7	90.7	97.4	60.6†	97.1	98.8	**92.0**	**98.3**	**63.7**†	96.8	98.3	91.3	97.8	62.2†
♤ - TCA	g	87.8	91.1	-	86.7	41.1†	**90.1**	**93.8**	-	**88.0**	**34.4**†	88.9	92.4	-	87.3	37.4†
♤	s	**96.8**	**97.8**	**91.9**	**97.6**	**70.4**†	97.1	98.8	91.3	98.2	59.8†	**96.9**	98.3	**91.6**	**97.9**	**64.6**†
♤	g	**89.9**	**92.2**	-	**88.6**	**49.7**†	89.2	93.1	-	86.3	30.5†	**89.6**	**92.7**	-	**87.4**	**37.8**†
UB-NO-SER	s	100	100	100	100	-	32.7	31.3	57.7	18.8	-	49.3	47.7	73.1	31.7	-
UB-NO-SER	g	0	0	-	0	-	0	0	-	0	-	0	0	-	0	-

†Five encoder layers are used instead of twelve for computational efficiency.

a

c

Table 8: Accuracy table for CORD-M task.

			test + GT-OCR	test
Model	task	language	F_1	F_1
SPADE†	receipt	IDN	87.7	82.9

† Five encoder layers are used for computational efficiency.

b

```
{
  {'menu_name': ['Lemon Tea (L)'], 'count': ['1'], 'price': ['25000']}
  {'total_price': ['25000'], 'cash_price': ['30000'], 'change_price': ['5000']}
}
{
  {'menu_name': ['PKT TELOR/PERK'], 'price': ['26000']}
  {'menu_name': ['TERONG'], 'price': ['12000']}
  {'menu_name': ['PARU'], 'price': ['23000']}
  {'menu_name': ['SBL GR'], 'price': ['20000']}
  {'menu_name': ['NESTLE 330 ML'], 'price': ['8000']}
  {'subtotal_price': ['89000'], 'tax_price': ['8900']}
  {'total_price': ['97900'], 'menuqty_cnt': ['5.00xITEMS'], 'cash_price': ['100000'], 'change_price': ['2100']}
}
```

Figure 4: The example of a receipt image from CORD-M (a), the predicted parse (b), and the accuracy table (c).

CORD dev set id: 0

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['16500']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['13000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['16000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['1650013000']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['16000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['PIZZA TOAST']}
{'total.cashprice': ['50000'], 'total.changeprice': ['4500'], 'total.total_price': ['45500']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['REAL GANACHE'], 'menu.price': ['1650013000']}
{'menu.cnt': ['1'], 'menu.nm': ['EGG TART'], 'menu.price': ['16000']}
{'menu.cnt': ['1'], 'menu.nm': ['PIZZA TOAST'], 'menu.price': ['45500']}
{'total.changeprice': ['4500'], 'total.total_price': ['50000']}
```

CORD dev set id: 1

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['404 - Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['404 - Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel 1'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['-'], 'menu.nm': ['DOA'], 'menu.price': ['23000']}
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.changeprice': ['27000'], 'total.total_price': ['50000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['-'], 'menu.nm': ['DOA'], 'menu.price': ['23000']}
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.changeprice': ['27000'], 'total.total_price': ['50000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['ELD Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['27000'], 'total.total_price': ['23000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['ELD'], 'menu.price': ['23000']}
{'menu.nm': ['Kopi Susu Kolonel'], 'menu.price': ['23000']}
{'total.changeprice': ['27000'], 'total.total_price': ['50000']}
```

CORD dev set id: 2

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine 50%'], 'menu.price': ['20000'], 'menu.unitprice': ['20000'], 'menu.vatyn': ['10% Tax Included']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000'], 'menu.sub_nm': ['50%'], 'menu.unitprice': ['20000']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000'], 'menu.sub_nm': ['50% 10'], 'menu.unitprice': ['20000']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000'], 'menu.sub_nm': ['50% 10'], 'menu.unitprice': ['20000']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine 50%'], 'menu.price': ['20000'], 'menu.unitprice': ['20000'], 'menu.vatyn': ['10% Tax Included']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000'], 'menu.sub_nm': ['50%'], 'menu.unitprice': ['20000']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000']}
{'menu.price': ['20000'], 'menu.sub_nm': ['50%']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000'], 'menu.sub_nm': ['50%'], 'menu.unitprice': ['20000']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine 50%'], 'menu.price': ['20000'], 'menu.unitprice': ['20000'], 'menu.vatyn': ['10% Tax Included']}
{'sub_total.subtotal_price': ['18181'], 'sub_total.tax_price': ['1818']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['S-Ovaltine'], 'menu.price': ['20000'], 'menu.unitprice': ['20000']}
{'sub_total.subtotal_price': ['18181']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.nm': ['S-Ovaltine']}
{'sub_total.subtotal_price': ['200001818']}
{'total.total_price': ['18181']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': [''], 'menu.nm': ['-Ovaltine'], 'menu.price': ['20000']}
{'sub_total.subtotal_price': ['1818']}
{'total.total_price': ['18181']}
{'total.cashprice': ['100000'], 'total.changeprice': ['80000'], 'total.total_price': ['20000']}
```

CORD dev set id: 3

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.subtotal_price': ['28000'], 'sub_total.tax_price': ['0']}
{'total.cashprice': ['28000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['70% Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.tax_price': ['0']}
{'total.cashprice': ['2800028000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['170'], 'menu.nm': ['M-Caramel Black Tea'], 'menu.price': ['28000'], 'menu.sub_nm': ['Less Ice'], 'menu.unitprice': ['28000']}
{'sub_total.tax_price': ['0']}
{'total.cashprice': ['2800028000'], 'total.changeprice': ['0'], 'total.total_price': ['28000']}
```

Figure 5: The example from CORD, CORD+, and CORD++ dev sets (ids 0–3).

CORD dev set id: 4

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['41000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menutype_cnt': ['1'], 'total.total_price': ['41000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['41000']}
{'sub_total.subtotal_price': ['41000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'total.cashprice': ['50000'], 'total.changeprice': ['410009000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['000']}
{'sub_total.subtotal_price': ['41']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['41000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': [': Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menutype_cnt': ['1'], 'total.total_price': ['41000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': [': Sedang'], 'menu.sub_price': ['0']}
{'total.cashprice': ['50000'], 'total.changeprice': ['410009000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['000']}
{'sub_total.subtotal_price': ['41']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': [': Sedang'], 'menu.sub_price': ['0']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['41000']}
{'sub_total.subtotal_price': ['41000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['9000'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['41000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['900'], 'total.menutype_cnt': ['1'], 'total.total_price': ['41000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['BBQ Chicken'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'total.cashprice': ['50000'], 'total.changeprice': ['41000', '900'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['000']}
{'sub_total.subtotal_price': ['41']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': [''], 'menu.nm': ['BBQ'], 'menu.price': ['41000'], 'menu.sub_cnt': ['1'], 'menu.sub_nm': ['Sedang'], 'menu.sub_price': ['0']}
{'sub_total.subtotal_price': ['41000']}
{'total.cashprice': ['50000'], 'total.changeprice': ['900'], 'total.menuqty_cnt': ['1'], 'total.total_price': ['41000']}
```

CORD dev set id: 5

Ground truth
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

SPADE
```
{'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.menuqty_cnt': ['1.00'], 'total.total_price': ['8000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

Ground truth
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

SPADE
```
{'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.menuqty_cnt': ['1.00'], 'total.total_price': ['8000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

Ground truth
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.total_price': ['8000']}
```

SPADE
```
{'menu.nm': ['LE MINERAL'], 'menu.price': ['8000']}
{'sub_total.subtotal_price': ['7273'], 'sub_total.tax_price': ['727']}
{'total.cashprice': ['8000'], 'total.menuqty_cnt': ['1.00'], 'total.total_price': ['8000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['80007273']}
{'sub_total.subtotal_price': ['727'], 'sub_total.tax_price': ['8000']}
{'total.total_price': ['8000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1.00'], 'menu.nm': ['LE MINERAL'], 'menu.price': ['80007273']}
{'sub_total.subtotal_price': ['727'], 'sub_total.tax_price': ['8000']}
{'total.total_price': ['8000']}
```

CORD dev set id: 6

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['152000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD 1'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD 1'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1']}
{'menu.cnt': ['1']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO WHITE POTATO GREEN CHOCO SAUSAGE TEA']}
{'menu.nm': ['BANANA BREAD SPREAD SPREAD']}
{'menu.price': ['52000']}
{'menu.price': ['52000']}
{'menu.price': ['19000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

Ground truth
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': ['1'], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123000'], 'total.total_price': ['123000']}
```

SPADE
```
{'menu.cnt': ['1'], 'menu.nm': ['POTATO SAUSAGE BREAD'], 'menu.price': ['19000']}
{'menu.cnt': ['1'], 'menu.nm': ['OREO GREEN TEA SPREAD'], 'menu.price': ['52000']}
{'menu.cnt': [''], 'menu.nm': ['WHITE CHOCO BANANA SPREAD'], 'menu.price': ['52000']}
{'total.creditcardprice': ['123500'], 'total.total_price': ['123000']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['11'], 'menu.nm': ['GREEN SAUSAGE TEA CHOCO']}
{'total.cashprice': ['52000'], 'total.changeprice': ['123500'], 'total.creditcardprice': ['52000123000'], 'total.total_price': ['19000']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['1'], 'menu.nm': ['SAUSAGE TEA'], 'menu.price': ['19000'], 'menu.sub_nm': ['GREEN']}
{'menu.cnt': ['1'], 'menu.nm': ['CHOCO'], 'menu.price': ['52000']}
{'sub_total.subtotal_price': ['52000']}
{'total.creditcardprice': ['500'], 'total.total_price': ['123123000']}
```

CORD dev set id: 7

Ground truth
```
{'menu.cnt': ['4'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

SPADE
```
{'menu.cnt': ['4'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'menu.nm': ['CP 360 Club Card']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['4'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'menu.cnt': [''], 'menu.discountprice': ['-9545']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['4'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'menu.cnt': ['360'], 'menu.discountprice': ['-9545']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

Ground truth
```
{'menu.cnt': ['4'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

SPADE
```
{'menu.cnt': ['4'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'menu.nm': ['CP 360 Club Card']}
{'sub_total.discount_price': ['', '9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.price': ['']}
{'menu.cnt': ['4'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['4'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'menu.cnt': [''], 'menu.price': ['-9545'], 'menu.sub_nm': ['360 Club Card']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

Ground truth
```
{'menu.cnt': ['4'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

SPADE
```
{'menu.cnt': ['2'], 'menu.discountprice': ['-9545'], 'menu.nm': ['Choco Devil'], 'menu.price': ['63636']}
{'menu.nm': ['CP 368 Club Card']}
{'sub_total.discount_price': ['-9545'], 'sub_total.subtotal_price': ['63636'], 'sub_total.tax_price': ['5409']}
{'total.cashprice': ['100'], 'total.changeprice': ['40500000'], 'total.total_price': ['59500']}
```

S_{ADV}+BERT+IOB$_2$
```
{'menu.cnt': ['-'], 'menu.price': ['636369545']}
{'menu.cnt': ['-'], 'menu.price': ['6363695455']}
{'sub_total.tax_price': ['409']}
{'sub_total.subtotal_price': ['59500']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500']}
```

S+BERT+IOB$_2$
```
{'menu.cnt': ['-'], 'menu.price': ['6363695456369545']}
{'sub_total.subtotal_price': ['63']}
{'sub_total.subtotal_price': ['5409']}
{'total.cashprice': ['100000'], 'total.changeprice': ['40500'], 'total.total_price': ['59500']}
```

Figure 6: The example from CORD, CORD+, and CORD++ dev sets (ids 4–7).

Reader-Guided Passage Reranking for Open-Domain Question Answering

Yuning Mao[1]*, **Pengcheng He**[2], **Xiaodong Liu**[3], **Yelong Shen**[2],
Jianfeng Gao[3], **Jiawei Han**[1], **Weizhu Chen**[2]
[1]University of Illinois, Urbana-Champaign [2]Microsoft Azure AI [3]Microsoft Research
[1]{yuningm2, hanj}@illinois.edu
[2,3]{penhe, xiaodl, yeshe, jfgao,wzchen}@microsoft.com

Abstract

Current open-domain question answering systems often follow a Retriever-Reader architecture, where the retriever first retrieves relevant passages and the reader then reads the retrieved passages to form an answer. In this paper, we propose a simple and effective passage reranking method, named Reader-guIDEd Reranker (RIDER), which does not involve training and reranks the retrieved passages solely based on the top predictions of the reader before reranking. We show that RIDER, despite its simplicity, achieves 10 to 20 absolute gains in top-1 retrieval accuracy and 1 to 4 Exact Match (EM) gains without refining the retriever or reader. In addition, RIDER, without any training, outperforms state-of-the-art transformer-based supervised rerankers. Remarkably, RIDER achieves 48.3 EM on the Natural Questions dataset and 66.4 EM on the TriviaQA dataset when only 1,024 tokens (7.8 passages on average) are used as the reader input after passage reranking.[1]

1 Introduction

Current open-domain question answering (OpenQA) systems often follow a Retriever-Reader (R2) architecture, where the retriever first retrieves relevant passages and the reader then reads the retrieved passages to form an answer. Since the retriever retrieves passages from a large candidate pool (*e.g.*, millions of Wikipedia passages), it often fails to rank the most relevant passages at the very top. One line of work (Mao et al., 2020; Karpukhin et al., 2020) aims to improve the retriever and shows that significantly better QA performance can be achieved when the retrieval results are improved.

An alternative solution is to rerank the initial retrieval results via a reranker, which is widely used in information retrieval (Nogueira and Cho, 2019; Qiao et al., 2019) and explored in early OpenQA systems (Wang et al., 2018a; Lee et al., 2018). However, current state-of-the-art OpenQA systems (Karpukhin et al., 2020; Izacard and Grave, 2020b) do not distinguish the order of the retrieved passages and instead equally consider a large number of retrieved passages (*e.g.*, 100), which could be computationally prohibitive as the model size of the readers becomes larger (Izacard and Grave, 2020b).

We argue that a Retriever-Reranker-Reader (R3) architecture is beneficial in terms of both model effectiveness and efficiency: passage reranking improves the retrieval accuracy of the retriever at top positions and allows the reader to achieve comparable performance with fewer passages as the input. However, one bottleneck of R3 is that the reranker, previously based on BiLSTM (Wang et al., 2018a; Lee et al., 2018) and nowadays typically BERT-based cross-encoder (Nogueira and Cho, 2019; Qiao et al., 2019), is often costly to train and its slow inference delays the whole pipeline as well.

Can we achieve better performance without the bother of training an expensive reranker or refining the retriever (reader)? In this paper, we propose a simple and effective passage reranking method, named **R**eader-gu**IDE**d **R**eranker (RIDER), which does not require any training and reranks the retrieved passages solely based on their lexical overlap with the top predicted answers of the reader before reranking. Intuitively, the top predictions of the reader are closely related to the ground-truth answer and even if the predicted answers are partially correct or incorrect, they may still provide useful signals suggesting which passages may contain the correct answer (Mao et al., 2020).

We conduct experiments on the Natural Questions (NQ) (Kwiatkowski et al., 2019) and Triv-

---

[*] Work was done during internship at Microsoft Azure AI.

[1] Our code is available at `https://github.com/morningmoni/GAR`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 344–350
August 1–6, 2021.

iaQA (Trivia) (Joshi et al., 2017) datasets. We demonstrate that R3 with RIDER, without any additional training, achieves 10 to 20 absolute gains in top-1 retrieval accuracy, and 1 to 4 gains in Exact Match (EM) compared to the R2 architecture. RIDER also outperforms two state-of-the-art transformer-based supervised reranking models that require expensive training and inference. Notably, using only 1,024 tokens (7.8 passages on average) as the input of a generative reader, RIDER achieves EM=47.5/63.5 on NQ/Trivia when the predictions of the same generative reader (EM=45.3/62.2 in R2) are used for reranking, and EM=48.3/66.4 on NQ/Trivia when the predictions of an extractive reader (EM=43.8/62.7 in R2) are used for reranking.

Contributions. (1) We propose Reader-guIDEd Reranker (RIDER), a simple and effective passage reranking method for OpenQA, which reranks the retriever results by the reader predictions without additional training and can be easily applied to existing R2 systems for performance improvements. (2) We demonstrate that the passages reranked by RIDER achieve significantly better retrieval accuracy and consequently lead to better QA performance without refining the retriever or reader. (3) Notably, RIDER achieves comparable or better performance than state-of-the-art methods on two benchmark datasets when only 1,024 tokens are used as the reader input after passage reranking.

2 Method

2.1 Task Formulation

We assume that an OpenQA system with an R2 architecture is available. We denote the initially retrieved passages of the retriever as $\mathbf{R}$. We denote the top-N predictions of the reader on the top-k passages of $\mathbf{R}$ (denoted as $\mathbf{R}^{[:k]}$) as $\mathbf{A}^{[:N]}$. The goal of RIDER is to rerank $\mathbf{R}$ to $\mathbf{R}'$ using $\mathbf{A}^{[:N]}$ such that the retrieval accuracy is improved and better end-to-end QA results are achieved when $\mathbf{R}'^{[:k]}$ is used as the reader input instead of $\mathbf{R}^{[:k]}$.

2.2 Passage Reranking

Given an initially retrieved passage list $\mathbf{R}$ and top-N predictions of the reader $\mathbf{A}^{[:N]}$, RIDER forms a reranked passage list $\mathbf{R}'$ as follows. RIDER scans $\mathbf{R}$ from the beginning of the list and appends to $\mathbf{R}'$ every passage $p \in \mathbf{R}$ if p contains any reader prediction $a \in \mathbf{A}^{[:N]}$ after string normalization (removing articles and punctuation) and tokenization. Then, the remaining passages are appended to $\mathbf{R}'$ according to their original order.

Intuitively, if the reader prediction is perfect, the retrieval accuracy after reranking is guaranteed to be optimal. Specifically, if the reader prediction is correct, it is guaranteed that the retrieval accuracy after reranking is better, since RIDER moves all passages containing the correct answer to the top (or at least the same if those passages are *all* at the top before reranking). If the reader prediction is wrong, RIDER could still be better if the predicted answer co-occurs with the correct answer, the same, or worse if the predicted answer is misleading. In practice, if the reader performs reasonably well, RIDER is also likely to rerank passages well. Overall, we observe quantitatively that RIDER leads to consistent gains in terms of both retrieval accuracy and QA performance without refining the retriever (reader) or even any training itself despite the noise in reader predictions.

2.3 Passage Reading

We consider a scenario where the number of passages that can be used for QA is limited (sometimes deliberately) due to reasons such as insufficient computational resources, the limit of model input length, or requirement for faster responses. We use a generative reader initialized by BART-large (Lewis et al., 2019), which concatenates the question and top-10 retrieved passages, trims them to 1,024 tokens (7.8 passages are left on average) as the input, and learns to generate the answer in a seq2seq manner (Mao et al., 2020; Min et al., 2020). We further add a simple shuffle strategy during reader training, which randomly shuffles the top retrieved passages before concatenation. In this way, the reader appears to be more robust to the reranked passages during inference and achieves better performance after reranking.

3 Experiment Setup

Datasets. We conduct experiments on the open-domain version of two widely used QA benchmarks – Natural Questions (NQ) (Kwiatkowski et al., 2019) and TriviaQA (Trivia) (Joshi et al., 2017), whose statistics are listed in Table 2.

Evaluation Metrics. Following prior studies (Mao et al., 2020; Karpukhin et al., 2020), we use top-k retrieval accuracy to evaluate the retriever and Exact Match (EM) to evaluate the reader. *Top-k retrieval accuracy* is the proportion of questions for

Data Input	NQ					Trivia				
	Top-1	Top-5	Top-10	Top-20	Top-100	Top-1	Top-5	Top-10	Top-20	Top-100
$\mathbf{R}$	46.8	70.7	77.0	81.5	88.9	53.2	73.1	77.0	80.4	85.7
$\mathbf{R}'$ by $\mathbf{G}$ (N=1)	**58.6**	71.4	76.9	81.6	88.9	**68.8**	74.8	77.5	80.4	85.7
$\mathbf{R}'$ by $\mathbf{G}$ (N=10)	56.4	**72.2**	**77.3**	81.6	88.9	66.9	**75.3**	**77.9**	**80.8**	85.7
$\mathbf{R}'$ by $\mathbf{E}$ (N=1)	**60.4**	72.1	77.3	81.7	88.9	**71.9**	77.5	79.8	81.8	85.7
$\mathbf{R}'$ by $\mathbf{E}$ (N=5)	53.5	**75.2**	**80.0**	83.2	88.9	63.2	**77.9**	**80.7**	82.8	85.7
$\mathbf{R}'$ by $\mathbf{E}$ (N=10)	50.3	74.3	80.0	**84.2**	88.9	59.8	77.1	80.5	**82.9**	85.7

Table 1: **Top-k retrieval accuracy on the test sets before and after reranking**. $\mathbf{G}$ and $\mathbf{E}$ denote generative and extractive readers, respectively, whose top predictions are used for reranking.

Dataset	Train / Val / Test	Q-len	A-len	#-A
NQ	79,168 / 8,757 / 3,610	12.5	5.2	1.2
Trivia	78,785 / 8,837 / 11,313	20.2	5.5	13.7

Table 2: **Dataset statistics** that show the number of samples, the average question (answer) length, and the average number of answers for each question.

which the top-k retrieved passages contain at least one answer span. It is an upper bound of how many questions are answerable by an extractive reader. *Exact Match (EM)* is the proportion of the predicted answer spans being exactly the same as one of the ground-truth answers, after string normalization such as article and punctuation removal.

Source of R. Following Mao et al. (2020), we take the top 100 retrieved passages of GAR (Mao et al., 2020) on Trivia and its combination with DPR (Karpukhin et al., 2020) on NQ as the initial retrieval results $\mathbf{R}$ for reranking.

Source of $\mathbf{A}^{[:N]}$. To obtain the top-N predicted answers, we first take the predictions of the generative reader ($\mathbf{G}$) in Sec. 2.3, which is trained on the passages without reranking and used for final passage reading in R3. It represents an apple-to-apple comparison to R2 without any additional information but higher-quality input. We also experiment with an extractive reader ($\mathbf{E}$) that has access to all retrieved passages, where the goal is to study whether we can rerank passages via other signals and further improve $\mathbf{G}$ such that it outperforms both $\mathbf{G}$ and $\mathbf{E}$ when they are in R2. We use the extractive reader in Mao et al. (2020) with BERT-base (Devlin et al., 2019) representation and span voting.

For the generative reader, we either take its top-1 prediction with greedy decoding or sample 10 answers with decoding parameters as follows. We set sampling temperature to 5/2 and the top probability in nucleus sampling to 0.5/0.5 on NQ/Trivia, respectively. Note that there are duplicate samples and on average $\bar{N} = 6$. We set the max input length to 1,024 and max output length to 10. For the extractive reader, the top predictions are the text spans with the highest scores and we set $N = 1, 5, 10$.

4 Experiment Results

4.1 Quality of Reranking Signals

We first analyze the EM of the top-N reader predictions $\mathbf{A}^{[:N]}$. We consider a question correctly answered as long as one of the top-N predictions matches the ground-truth answer. The standard EM is a special case with $N = 1$. As listed in Table 3, the reader EM can be improved by up to 24.0 on NQ and 15.8 on Trivia if we consider the top-10 predictions instead of only the first prediction, suggesting that there is significant potential if we use multiple predicted answers for reranking. That said, using more reader predictions also introduces more noise, *i.e.*, incorrect answers, which could be misleading at times.

Dataset	Top-1	Top-3	Top-5	Top-10
NQ	43.8 (45.3)	57.4	62.6	67.8 (54.2)
Trivia	62.7 (62.2)	72.6	75.5	78.5 (67.7)

Table 3: **EM of top-N predictions of the reader.** Results are mostly on reader $\mathbf{E}$. Only top-1 and top-10 EM are shown (in the brackets) for reader $\mathbf{G}$, as its 10 predictions are sampled without particular order.

4.2 RIDER for Passage Retrieval

We list the top-k retrieval accuracy before and after passage reranking in Table 1. RIDER significantly improves the retrieval accuracy at top positions (especially top-1) without refining the retriever. In particular, we observe that when taking more reader predictions (*i.e.*, larger N), the top-k retriever accuracy tends to improve more at a larger k and less at a smaller k. For example, an improvement of about 3 points is achieved for top-5 and top-10 accuracy when increasing N from 1 to 5 on NQ for reader $\mathbf{E}$, but the top-1 retrieval accuracy also drops significantly (although still better than without reranking), which again suggests that there is a trade-off between answer coverage and noise. Note that the top-100 retrieval accuracy is unchanged after reranking since we rerank the top-100 passages.

4.3 RIDER for Passage Reading

Comparison w. the state-of-the-art. We show the QA performance comparison between RIDER and state-of-the-art methods in Table 4. We observe that RIDER improves GAR (or GAR+DPR) on both datasets by a large margin, despite that they use the same generative reader and no further model training is conducted. Such results indicate that RIDER provides higher-quality input for the reader and better performance can be achieved with the same input length. Moreover, the results of RIDER are better than most of the existing methods that take more passages as input, except for FID-large (Izacard and Grave, 2020b) that reads 100 passages and also has more model parameters.

	Method	**NQ**	**Trivia**
Extractive	Hard EM (Min et al., 2019a)	28.1	50.9
	Path Retriever (Asai et al., 2019)	32.6	-
	ORQA (Lee et al., 2019)	33.3	45.0
	Graph Retriever (Min et al., 2019b)	34.5	56.0
	REALM (Guu et al., 2020)	40.4	-
	DPR (Karpukhin et al., 2020)	41.5	57.9
	BM25 (Mao et al., 2020)	37.7	60.1
	GAR	**41.8**	**62.7**
	GAR+DPR	**43.8**	-
Generative	GPT-3 (Brown et al., 2020)	29.9	-
	T5 (Roberts et al., 2020)	36.6	60.5
	SpanSeqGen (Min et al., 2020)	42.2	-
	RAG (Lewis et al., 2020)	44.5	56.1
	FID-base (Izacard and Grave, 2020b)	48.2	65.0
	FID-large (Izacard and Grave, 2020b)	**51.4**	**67.6**
	BM25 (Mao et al., 2020)	35.3	58.6
	GAR (Mao et al., 2020)	38.1	62.2
	RIDER (GAR)	-	**66.4**
	GAR+DPR (Mao et al., 2020)	45.3	-
	RIDER (GAR+DPR)	**48.3**	-

Table 4: **End-to-end QA comparison of state-of-the-art methods**. RIDER results in up to 4.2 EM gains.

Data Input	**NQ**	**Trivia**
$\mathbf{R}$	45.3	62.2
$\mathbf{R}'_1$ by $\mathbf{G}$ (N=1)	(45.3+1.1) 46.4	(62.2+0.7) 62.9
$\mathbf{R}'_2$ by $\mathbf{G}$ (N=10)	(45.3+2.1) **47.4**	(62.2+0.9) **63.1**
$\mathbf{R}''$ by $\mathbf{R}'_1$	(46.4+1.1) **47.5**	(62.9+0.6) **63.5**
$\mathbf{R}'$ by $\mathbf{E}$ (N=1)	(43.8+3.2) 47.0	(62.7+3.4) **66.1**
$\mathbf{R}'$ by $\mathbf{E}$ (N=5)	(43.8+4.5) **48.3**	(62.7+2.5) 65.2
$\mathbf{R}''$ by best $\mathbf{R}'$	(48.3+0) **48.3**	(66.1+0.3) **66.4**

Table 5: **Comparison of RIDER in EM when different reranking signals are used**. The numbers in the brackets represent the performance of the reader used for reranking and relative gains.

Method	**Top-1**	**Top-5**	**Top-10**	**Top-20**
$\mathbf{R}$	46.8	70.7	77.0	81.5
$\mathbf{R}'$ by BERT reranker	51.4	67.6	75.7	82.4
$\mathbf{R}'$ by BART reranker	55.2	73.5	78.5	82.2
$\mathbf{R}'$ by $\mathbf{G}$ (N=10)	**56.4**	72.2	77.3	81.6
$\mathbf{R}'$ by $\mathbf{E}$ (N=5)	53.5	**75.2**	**80.0**	**83.2**

Table 6: **Comparison with supervised rerankers in top-k retrieval accuracy on NQ**. RIDER outperforms expensive transformer-based models without training.

Ablation Study. A detailed analysis of RIDER with different reranking signals is shown in Table 5. By reranking based on the prediction of the generative reader $\mathbf{G}$ (with input $\mathbf{R}^{[:k]}$), RIDER generally achieves 1 to 2 gains in EM, which shows that RIDER can improve end-to-end QA performance without any additional information. By iterative reranking ($\mathbf{R}''$) using the reader predictions after first reranking, the performance of RIDER is further improved. Conducting more than two iterations of reranking does not appear to bring additional gains.

RIDER achieves even better performance when using the predictions of the extractive reader $\mathbf{E}$ (with input $\mathbf{R}$) for reranking, which is consistent with the results on retrieval. It is also encouraging to see that RIDER significantly outperforms $\mathbf{E}$, which is more computationally expensive and has access to much more passages.

4.4 Comparison w. Supervised Reranking

Finally, we compare RIDER with two state-of-the-art supervised reranking models. The first reranker is a BERT-base cross-encoder (Nogueira and Cho, 2019), which is popularly used for passage reranking in information retrieval. The cross-encoder concatenates the query and passage, and makes a binary relevance decision for each query-passage pair. The second one generates relevance labels as target tokens in a seq2seq manner (Nogueira et al., 2020). We use BART-large as the base model and "YES/NO" as the target tokens.

As listed in Table 6, RIDER, without any training, outperforms the two transformer-based supervised rerankers on retrieval accuracy. Also, for QA performance, the best EM we obtain using the supervised rerankers is merely 46.3 on NQ. Such results further demonstrate the effectiveness of RIDER, which has the advantage of utilizing information from multiple passages (when the reader makes predictions), while the other rerankers consider query-passage pairs independently.

4.5 Runtime Efficiency

The reranking step of RIDER only involves string processing, which can be easily paralleled and reduced to within seconds. We use Nvidia V100 GPUs for reader training and inference. The training of the generative reader takes 8 to 10 hours with 1 GPU, while it takes 12 to 16 hours with 8 GPUs for the DPR reader (Karpukhin et al., 2020). Due to fewer input passages, the inference of the generative reader is also very efficient – it takes around 3.5/11 min to generate answers on the NQ/Trivia test set with 1 GPU. In comparison, the DPR reader takes about 14/40 min with 8 GPUs.

5 Related Work

Reranking for OpenQA. Reranking has been widely used in information retrieval to refine the initial retrieval results. Early effort on passage reranking for OpenQA uses supervised (Lee et al., 2018) or reinforcement learning (Wang et al., 2018a) based on BiLSTM. More recently, BERT-based rerankers that treat the query and passage as a sentence pair (*i.e.*, cross-encoders) achieve superior performance (Nogueira and Cho, 2019; Qiao et al., 2019). However, the training of cross-encoders is rather costly. Moreover, the representations of cross-encoders cannot be pre-computed and matched via Maximum Inner Product Search (MIPS) as in bi-encoders (Karpukhin et al., 2020) but measured online between the query and each passage, which results in slower inference as well.

Another line of work (Das et al., 2018; Qi et al., 2020) reranks the passages by updating the query and often involves a complicated learning process such as R2 interactions. Alternatively, some prior studies (Wang et al., 2018b; Iyer et al., 2020) directly rerank the top predicted answers instead of the passages using either simple heuristics or additional training. In contrast, RIDER utilizes downstream signals (*i.e.*, the predictions of a reader) to rerank the passages without any training.

Reader Distillation. Recent studies (Izacard and Grave, 2020a; Yang and Seo, 2020) show that distillation from the preference of the reader can improve the retriever performance, where the reader preference is measured by the attention scores of the reader over different passages and the retriever is refined by learning to approximate the scores. RIDER, to some extent, can also be seen as one way to distill the reader. However, RIDER is much simpler in that no further training is involved for either the retriever or reader, and explicit reader predictions instead of latent attention scores are leveraged to improve the retriever results directly.

6 Conclusion

In this work, we propose RIDER, a simple and effective passage reranking method for OpenQA, which does not involve additional training or computationally expensive inference, and outperforms state-of-the-art supervised rerankers that involve both. RIDER can be easily integrated into existing R2 systems for performance improvements. Without fine-tuning the retriever or reader, RIDER improves the retrieval accuracy and the QA results on two benchmark datasets significantly. Notably, RIDER achieves comparable or better performance than state-of-the-art methods with less reader input and allows for more efficient OpenQA systems. For future work, we will explore other simple and effective reranking strategies with no (minimal) training or external supervision.

Acknowledgments

We thank Hao Cheng, Sewon Min, and Wenda Qiu for helpful discussions. We thank the anonymous reviewers for valuable comments.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2019. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2018. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Srinivasan Iyer, Sewon Min, Yashar Mehdad, and Wen-tau Yih. 2020. Reconsider: Re-ranking using span-focused cross-attention for open domain question answering. *arXiv preprint arXiv:2010.10757*.

Gautier Izacard and Edouard Grave. 2020a. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*.

Gautier Izacard and Edouard Grave. 2020b. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 565–569, Brussels, Belgium. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020. Generation-augmented retrieval for open-domain question answering. *arXiv preprint arXiv:2009.08553*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2851–2864, Hong Kong, China. Association for Computational Linguistics.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Peng Qi, Haejun Lee, Oghenetegiri Sido, Christopher D Manning, et al. 2020. Retrieve, rerank, read, then iterate: Answering open-domain questions of arbitrary complexity from text. *arXiv preprint arXiv:2010.12527*.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of bert in ranking. *arXiv preprint arXiv:1904.07531*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R^3: Reinforced reader-ranker for open-domain question answering. *The AAAI Conference on Artificial Intelligence (AAAI)*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *International Conference on Learning Representations*.

Sohee Yang and Minjoon Seo. 2020. Is retriever merely an approximator of reader? *arXiv preprint arXiv:2010.10999*.

Entity-Aware Abstractive Multi-Document Summarization

Hao Zhou[1], **Weidong Ren**[1], **Gongshen Liu**[1]*, **Bo Su**[1], **Wei Lu**[2]
[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University
[2] StatNLP Research Group, Singapore University of Technology and Design
{zhou1998,renweidong1997,lgshen,subo}@sjtu.edu.cn
luwei@sutd.edu.sg

Abstract

Entities and their mentions convey significant semantic information in documents. In multi-document summarization, the same entity may appear across different documents. Capturing such cross-document entity information can be beneficial – intuitively, it allows the system to aggregate diverse useful information around the same entity for better summarization. In this paper, we present EMSum, an entity-aware model for abstractive multi-document summarization. Our model augments the classical Transformer-based encoder-decoder framework with a heterogeneous graph consisting of text units and entities as nodes, which allows rich cross-document information to be captured. In the decoding process, we design a novel two-level attention mechanism, allowing the model to deal with saliency and redundancy issues explicitly. Our model can also be used together with pre-trained language models, arriving at improved performance. We conduct comprehensive experiments on the standard datasets and the results show the effectiveness of our approach.

1 Introduction

Multi-document summarization aims at generating a short and informative summary across a set of topic-related documents. It is a task that can be more challenging than single-document summarization due to the presence of diverse and potentially conflicting information (Ma et al., 2020).

While significant progress has been made in single-document summarization, the mainstream sequence-to-sequence models, which can perform well on single-document summarization, often struggle with extracting salient information and handling redundancy in the presence of multiple, long documents. Thus, simply adopting models that were shown effective for single-document summarization to the multi-document setup may not lead to ideal results (Lebanoff et al., 2018; Zhang et al., 2018; Baumel et al., 2018).

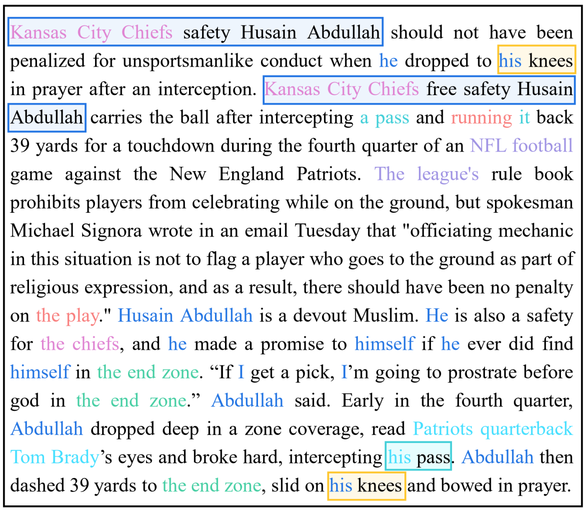

Kansas City Chiefs safety Husain Abdullah should not have been penalized for unsportsmanlike conduct when he dropped to his knees in prayer after an interception. Kansas City Chiefs free safety Husain Abdullah carries the ball after intercepting a pass and running it back 39 yards for a touchdown during the fourth quarter of an NFL football game against the New England Patriots. The league's rule book prohibits players from celebrating while on the ground, but spokesman Michael Signora wrote in an email Tuesday that "officiating mechanic in this situation is not to flag a player who goes to the ground as part of religious expression, and as a result, there should have been no penalty on the play." Husain Abdullah is a devout Muslim. He is also a safety for the chiefs, and he made a promise to himself if he ever did find himself in the end zone. "If I get a pick, I'm going to prostrate before god in the end zone." Abdullah said. Early in the fourth quarter, Abdullah dropped deep in a zone coverage, read Patriots quarterback Tom Brady's eyes and broke hard, intercepting his pass. Abdullah then dashed 39 yards to the end zone, slid on his knees and bowed in prayer.

Figure 1: A sample article from MultiNews. We show the results of co-reference resolution. Mentions of the same entity are highlighted with the same color.

Several previous research efforts have shown that modeling cross-document relations is essential in multi-document summarization (Liu and Lapata, 2019a; Li et al., 2020). Such relations were shown useful in identifying the salient and redundant information from long documents, and can thus guide the summary generation process. However, while effective empirically, such approaches do not focus on explicitly modeling the underlying semantic information across documents.

Entities and their mentions convey rich semantic information, and can be significant in summarization, especially when a specific entity is the topic under discussion for a set of documents. As shown in Figure 1, entity mentions frequently appear in the input article, and are playing unique roles that contribute towards the coherence and conciseness of the text. We believe that entities can be regarded

---

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 351–362
August 1–6, 2021.

as the indicator of saliency and can be used to reduce redundancy. This motivates us to propose an entity-aware abstractive multi-document summarization model that effectively encodes relations across documents with the help of entities, and explicitly solve the issues of saliency and redundancy.

Inspired by Wang et al. (2020a), we build a heterogeneous graph that consists of nodes that represent documents and entities. The entity nodes can serve as bridges that connect different documents – we can model the relations across documents through entity clusters. We apply the graph attention network (GAT) (Veličković et al., 2017) to enable information flow between nodes and iteratively update the node representations. In the decoding process, we design a novel two-level attention mechanism. The decoder first attends to the entities. Next, the attention weights of entities are incorporated with graph edge weights to guide the attention to the documents. Intuitively, the first stage indentifies the salient content in each decoding step. By considering the global interactions between entities and documents in the graph, the second stage is able to handle the redundancy issue. Experiments show that our model significantly improves the performance on several multi-document datasets. Further improvements can be made when our model is used together with the pre-trained language models.

Our contributions are as follows:

- We construct a heterogeneous graph network for multi-document summarization. The graph consists of document-level and entity-level nodes. To the best of our knowledge, we are the first to model the relations between documents and entities in one heterogeneous graph. Experiments show that exploiting entity nodes as the intermediary between documents can be more effective than exploiting other semantic units (e.g., words).
- We propose a novel two-level attention mechanism during the decoding process, solving the issues of saliency and redundancy explicitly. The mechanism can also reduce the computational cost, making it easier to process long inputs.
- Our model achieves state-of-the-art results on WikiSum and MultiNews. Extensive analysis including ablation studies show the effectiveness of our model.[1]

2 Related Work

2.1 Abstractive Document Summarization

Abstractive summarization is often regarded as the ultimate goal of document summarization research. Extractive summarization methods produce summaries that are semantically similar to the original documents. Thus, they may be able to achieve relatively high ROUGE scores (Lin, 2004). However, sentence-level extraction lacks flexibility and tends to produce redundant information. By contrast, the process of abstractive summarization is more similar to the human summarization process and requires more sophisticated natural language understanding and generation techniques. Traditional approaches to abstractive summarization can be divided into sentence fusion-based (Barzilay and McKeown, 2005; Filippova and Strube, 2008; Banerjee et al., 2015), paraphrasing-based (Bing et al., 2015; Cohn and Lapata, 2009) and information extraction-based (Li, 2015; Wang and Cardie, 2013; Pighin et al., 2014).

With the development of neural-based methods, abstractive methods achieved promising results on single document summarization (See et al., 2017; Paulus et al., 2018; Gehrmann et al., 2018; Li et al., 2018). More recently, due to the excellent performance on various text generation tasks, transformer-based methods become the mainstream approach for abstractive multi-document summarization, as well as pre-trained language models. Liu and Lapata (2019b) propose BertSUM for both extractive and abstractive summarization. Zhang et al. (2019) build low-level and high-level Berts for sentence and document understanding, respectively. Moreover, several general purpose sequence-to-sequence pre-trained models are proposed, such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2019). They are further fine-tuned for the summarization task. Zhang et al. (2020) propose PEGASUS, in which they design a pre-training objective tailored for abstractive text summarization. Zou et al. (2020) present three sequence-to-sequence pre-training objectives by reinstating source text for abstractive summarization.

2.2 Graph-based Document Summarization

Graph-based methods have long been utilized for extractive summarization. Text units on graphs are ranked and selected as the most salient ones to be included in the summary. LexRank (Erkan and Radev, 2004) computes sentence salience based on

---

[1] Our code is at https://github.com/Oceandam/EMSum

the eigenvector centrality in the connectivity graph of inter-sentence cosine similarity. Wan (2008) further incorporate the document-level information and the sentence-to-document relationship into the graph-based ranking process. Christensen et al. (2013) build multi-document graphs to approximate the discourse relations across sentences based on indicators including discourse cues, deverbal nouns, co-reference and more.

For recent methods based on graph neural networks, Tan et al. (2017) propose a graph-based attention mechanism to identify salient sentences. Yasunaga et al. (2017) construct an approximate discourse graph based on discourse markers and entity links, then apply graph convolutional networks over the relation graph. Fan et al. (2019) construct a local knowledge graph, which is then linearized into a structured input sequence so that models can encode within the sequence-to-sequence setting. Huang et al. (2020) further design a graph encoder, which improves upon graph attention networks, to maintain the global context and local entities complementing each other. Li et al. (2020) utilize homogeneous graphs to capture cross-document relations and guide the summary generation process. However, Wang et al. (2020a) are the first to introduce different granularity levels of text nodes to construct heterogeneous graphs for extractive summarization. Our work is partly similar to theirs, but we construct heterogeneous graphs composed of text unit nodes and entity nodes for abstractive multi-document summarization.

2.3 Summarization with Additional Features

In addition to the direct application of the general sequence-to-sequence framework, researchers attempted to incorporate various features into summarization. Cao et al. (2018) extract actual fact descriptions from the source text and propose a dual-attention mechanism to force the generation conditioned on both the source text and the extracted fact descriptions. Sharma et al. (2019) take a pipeline method for single-document summarization which is composed of an entity-aware content selection module and a summary generation module. By contrast, our EMSum model is an end-to-end method for multi-document summarization. Gunel et al. (2020) inject structural world knowledge from Wikidata to a transformer-based model, enabling the model to be more fact-aware. Zhu et al. (2020) extract factual relations from the source texts to build a local knowledge graph and integrated it into the transformer-based model.

Apart from entity or fact information, there are several works that incorporate topic information into summarization model. Narayan et al. (2018) recommend an encoder associating each word with a topic vector capturing whether it is representative of the document's content, and a decoder where each word prediction is conditioned on a document topic vector. Zheng et al. (2019) propose to mine cross-document subtopics. In their work, sentence salience is estimated in a hierarchical way with subtopic salience and relative sentence salience. Perez-Beltrachini et al. (2019) explicitly model the topic structure of summaries, and utilize it to guide a structured convolutional decoder. Wang et al. (2020b) rearrange and further explore the semantics of the topic model and develope a friendly topic assistant for transfomer-based abstractive summarization models.

3 Model

Our model is illustrated in Figure 2, which follows the transformer-based encoder-decoder architecture (Vaswani et al., 2017). We modify the encoder with graph neural networks, so we can incorporate entity information and graph representations at the same time. We design a novel two-level decoding process to explicitly deal with the problem of saliency and redundancy.

3.1 Entity Cluster Extraction

Wang et al. (2020a) use words as semantic units in addition to sentence nodes, acting as the intermediary to enrich the relationships between sentences. However, we argue that word-level semantic units are too fine and will bring huge computational costs. For multi-document summarization, models are usually required to process tens of documents. The total number of words will be vast, which further causes a hindrance for the graph construction and message passing process. Therefore, we use entity clusters as more advanced semantic units. We utilize the co-reference resolution tool (Lee et al., 2017) from AllenNLP (Gardner et al., 2018) to extract entity clusters. Note that we perform extraction globally, which means we concatenate all the documents into one long document. We denote the extracted entity clusters as $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, where $C_i = \{mention_1, mention_2, \ldots, mention_l\}$, and l is

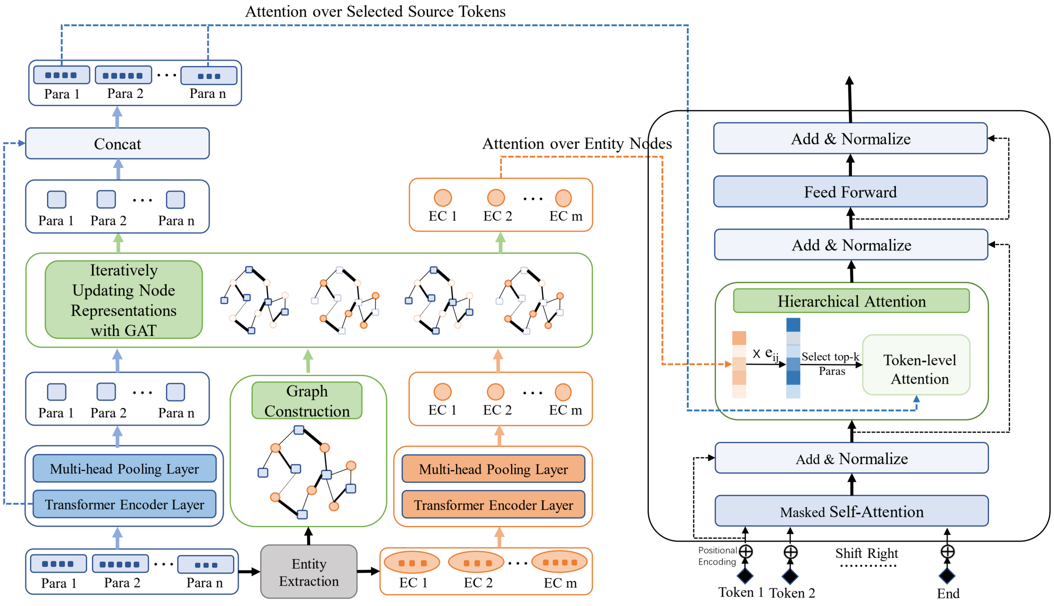

Figure 2: Overall architecture of our model.

the number of entity mentions in cluster C_i.

3.2 Graph Construction

Given a source document cluster $\mathcal{D}$, we firstly divide them into smaller semantic units $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, such as paragraphs and sentences, depending on the characteristics of datasets. We then construct a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ includes paragraph nodes $\mathcal{V}_p$ and entity cluster nodes $\mathcal{V}_c$. $\mathcal{E}$ represents undirected edges between nodes. There exists no edge inside paragraph nodes or entity cluster nodes, but only between them. An edge which connects P_i and C_j means paragraph P_i contains an entity mention in C_j.

We would like to include more information in the graph. We get an occurrence matrix $\mathbf{E} \in \mathbb{R}^{m \times n}$ after extraction, where $e_{ij} \neq 0$ indicates P_i contains entity mentions in C_j for e_{ij} times. Based on $\mathbf{E}$, we further calculate the TF-IDF value matrix $\tilde{\mathbf{E}} \in \mathbb{R}^{m \times n}$ to model the importance of relationships between entity clusters and paragraphs.

3.3 Document Encoder

Paragraph Encoder Several token-level transformer encoding layers are stacked to encode contextual information within each paragraph. The transformer layer is the same as the vanilla transformer layer (Vaswani et al., 2017). Let x_w^0 be the input token vector. For the l-th transformer layer, the input is x_w^{l-1}, the hidden state is h_w^l, and the output is x_w^l.

$$h_w^l = \text{LayerNorm}(x_w^{l-1} + \text{MHAttn}(x_w^{l-1})) \quad (1)$$

$$x_w^l = \text{LayerNorm}(h_w^l + \text{FFN}(h_w^l)) \quad (2)$$

LayerNorm is the layer normalization operation (Ba et al., 2016). MHAttn is multi-head attention from Vaswani et al. (2017). FFN is a feed-forward network with ReLU as activation function. We take the output of last layer as token-level features. We use $\mathbf{H_{p_w}} \in \mathbb{R}^{n_w \times d_w}$ to denote the token-level feature matrix, where n_w is the total number of tokens in all paragraphs and d_w is the dimension of token embedding.

Multi-Head Pooling To obtain fixed length paragraph representations, we follow Liu and Lapata (2019a) to apply a weighted-pooling operation. The multi-head pooling mechanism calculates the weight distributions over tokens, allowing the model to flexibly encode paragraphs in different representational subspace by different head.

$$\mathbf{h_p} = \text{MHPool}(h_{w1}, h_{w2}, \ldots) \quad (3)$$

We use $\mathbf{H_p} \in \mathbb{R}^{n \times d_h}$ to denote the paragraph level feature matrix, where n is the number of paragraphs, d_h is the hidden size.

Entity Cluster Encoder We perform the same encoding process as the paragraph encoder to get entity clusters' representation, but without sharing parameters between the two encoders. We choose this method rather than additional entity embedding methods because we seek to model the relationship between paragraphs and entities in a unified semantic space. Note that we firstly remove pronouns and stopwords in entity mention clusters, which are common in co-reference resolution results but render little benefit for our semantic modeling. We use $\mathbf{H_{c_w}} \in \mathbb{R}^{m_w \times d_w}$ and $\mathbf{H_c} \in \mathbb{R}^{m \times d_h}$ to denote the token level feature matrix and cluster level feature matrix, respectively.

3.4 Graph Encoder

We use graph attention networks (GAT) (Veličković et al., 2017) to update the representations of semantic nodes. We use $i, j \in \{1, ..., (m+n)\}$ to denote an arbitrary node in graph, use $\mathbf{h}_i, \mathbf{h}_j \in \mathbb{R}^{d_h}$ to denote the node representations, and use $\mathcal{N}_i$ to denote the set of neighboring nodes of node i. The GAT layer is designed as follows:

$$z_{ij} = \text{LeakyReLU}(\mathbf{W}_a[\mathbf{W}_q\mathbf{h}_i; \mathbf{W}_k\mathbf{h}_j]) \quad (4)$$

$$\tilde{z}_{ij} = \tilde{e}_{ij} \times z_{ij} \quad (5)$$

$$\alpha_{ij} = \frac{\exp(\tilde{z}_{ij})}{\sum_{l \in \mathcal{N}_i} \exp(\tilde{z}_{il})} \quad (6)$$

$$\mathbf{u_i} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{W}_v\mathbf{h}_j\right) \quad (7)$$

where $\mathbf{W}_a$, $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ are trainable weights, σ is the sigmoid function, $\tilde{e}_{ij}$ is the edge weight derived from TF-IDF value matrix $\tilde{\mathbf{E}}$.

We basically follow Wang et al. (2020a) to iteratively update node representations. They infuse the scalar edge weight $\tilde{e}_{ij}$ by simply discretizing the real values into integers, and then learn embeddings for such integers. That is how they map the weights to the multi-dimensional embedding space $\mathbf{e_{ij}} \in \mathbb{R}^{d_e}$. In this way, the information contained in the values needs to be learned by an additional embedding matrix. However, we argue that TF-IDF values themselves indicate the closeness between an entity cluster and a paragraph. Therefore, we directly incorporate the raw TF-IDF information into the GAT mechanism by modifying the attention weights using Equation 5.

We combine GAT with multi-head operation. We also add a residual connection to avoid gradient vanishing after several iterations:

$$\tilde{\mathbf{h}}_i = \mathbf{h}_i + \mathbf{u}_i \quad (8)$$

We use the above GAT layer and position-wise feed-forward layer to iteratively update the node representations. Each iteration contains a paragraph-to-entity and a entity-to-paragraph updating process. After iterating for t times, we concatenate $\tilde{\mathbf{H}}_\mathbf{p}$ to each corresponding input token vector, arriving at $\tilde{\mathbf{H}}_{\mathbf{p_w}} \in \mathbb{R}^{n_w \times (d_w + d_h)}$.

3.5 Entity-Aware Decoder with Two-level Attention

Under the setting of multi-document summarization, the input source documents may involve an extremely large number of word tokens. If the decoder needs to compute attention weights over all tokens, the cost would be very high and the attention could be dispersed. Our two-level decoding process firstly focuses on several centering entity cluster nodes, which can be regarded as indicators of saliency. The indicator restricts the token-level attention only to some of the paragraphs, which can further reduce redundancy than naively attending to all tokens. Different from Section 3.4, we use i and j to denote the entity node and paragraph node, respectively.

Attending the Entity Cluster Nodes At each decoding step, the state of decoder is $\mathbf{s}$, we compute attention scores over entity cluster nodes $\mathbf{c}_i$.

$$z_i = \mathbf{u}_0^T \text{LeakyReLU}([\mathbf{W}_{z_1}\mathbf{s}; \mathbf{W}_e\mathbf{c}_i]) \quad (9)$$

The entity nodes act as the intermediary between paragraph nodes. We incorporate z_i with edge weights $\tilde{e}_{ij}$ to enable the information flow between entity nodes and paragraph nodes by:

$$\tilde{z}_j = \sum_{i=1}^{m} z_i \times \tilde{e}_{ij} \quad (10)$$

$$\beta_j = \frac{\exp(\tilde{z}_j)}{\sum_{l=1}^{m} \exp(\tilde{z}_l)} \quad (11)$$

Attending the Paragraph Tokens We select the top-k paragraph nodes with the highest attention score β_j. Then we apply the attention mechanism over the T_w tokens in the selected paragraphs.

$$z_{w_i} = \mathbf{u}_1^T \text{LeakyReLU}([\mathbf{W}_{z_2}\mathbf{s}_t; \mathbf{W}_w\tilde{h}_{w_i}]) \quad (12)$$

$$\gamma_{w_i} = \frac{\exp(z_{w_i})}{\sum_{l=1}^{T_w} \exp(z_{w_l})} \quad (13)$$

For token w_i in paragraph P_j, we further modify γ_{w_i} by

$$\hat{\gamma}_{w_i} = \beta_j \times \gamma_{w_i} \tag{14}$$

Then the context vector $\mathbf{v}_t$ can be computed by:

$$\mathbf{v}_t = \sum_i \hat{\gamma}_{w_i} \tilde{h}_{w_i} \tag{15}$$

Token Prediction Context vectors, treated as salient contents summarized from sources, are concatenated with the decoder hidden state $\mathbf{s}_t$ to produce the vocabulary distribution:

$$P_{vocab} = \mathrm{Softmax}(\mathbf{W}_o[\mathbf{s}_t; \mathbf{v}_t]) \tag{16}$$

We use the weight-sharing strategy between the input embedding matrix and the matrix $\mathbf{W}_o$ to reuse linguistic knowledge (Paulus et al., 2018). We further add a copy mechanism as proposed by See et al. (2017).

3.6 Training

Our training process follows that of the traditional sequence-to-sequence modeling, with maximum likelihood estimation that minimizes:

$$\mathcal{L}_{seq} = -\frac{1}{|D|} \sum_{(\mathbf{y},\mathbf{x}) \in D} \log p(\mathbf{y}|\mathbf{x}; \theta) \tag{17}$$

where $\mathbf{x}$ and $\mathbf{y}$ are document-summary pairs from training set D, and θ are parameters to be learned.

3.7 Pre-trained LMs as Document Encoder

Our document encoder illustrated in section 3.3 can be replaced by a pre-trained language model such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). Pre-trained language models can be more effective on short inputs than training stacked transformer layers from scratch. We feed input tokens to a pre-trained language model and take the last layer output as token embeddings. Then a single-layer bidirectional LSTM is employed over token embeddings, producing token features. Finally, we perform the same multi-head pooling strategy to obtain paragraph representations.

4 Experiments

We use ROUGE scores to evaluate summarization quality automatically (Lin, 2004). We report different versions of the metric, based on overlaps of unigrams (ROUGE-1, R-1), bigrams (ROUGE-2, R-2) and the longest common subsequences (ROUGE-L, R-L).

4.1 Experimental Setup

We conduct experiments on two major datasets used in the literature of multi-document summarization, namely WikiSum (Liu et al., 2018) and MultiNews (Fabbri et al., 2019).

WikiSum Dataset Liu et al. (2018) treat the generation of Wikipedia section titles as a supervised multi-document summarization task. Liu and Lapata (2019a) crawled Wikipedia articles and source reference documents through the provided urls. They further split the long and messy source documents into multiple paragraphs by line-breaks and select the top-40 paragraphs as input for summarization systems. However, the top-40 dataset is quite heavy for entity extraction and co-reference resolution. Experiment shows that the ROUGE-L recall of top-20 paragraphs against the gold target text is 53.84, and top-40 is 60.42. So we choose to use the top-20 version of WikiSum dataset in order to find a balance between computational cost and the coverage of input content. We get 300,000 instances for training, 38,144 for validation and 38,205 for test. On average, each paragraph has 70.1 tokens, and target sumamry has 139.4 tokens. We then perform entity cluster extraction on the top-20 WikiSum dataset. For each instance, we get 23.7 clusters on average and each cluster has 10.2 tokens on average.

MultiNews Dataset Introduced by Fabbri et al. (2019), MultiNews consists of news articles and hand-written summaries. The source articles come from a diverse set of news sources, over 1,500 sites. Following their experimental settings, we get 44,972 instances for training, 5,622 for validation and 5,622 for test. Different from the WikiSum dataset, each source article only contains 2.8 paragraphs and 21.6 sentences on average, thus we choose to build graph on sentence level rather than paragraph level for this dataset. For each instance, we get 13.3 clusters on average and each cluster has 9.9 tokens on average.

Hyperparameters We set the number of our vanilla Transformer encoding layers as 6, the hidden size as 256 and the number of heads as 8, while the hidden size of feed-forward layers is 1,024. We truncate the length of input paragraphs and entity clusters to 100 and 50 tokens, respectively. In the multi-head pooling layer, the number of heads is 8. In the graph encoding process, each layer has 8 heads and the hidden size is 256. We select the

	Model	R-1	R-2	R-L
Ext	Lead†	38.22	16.85	26.89
	LexRank†	36.12	11.67	22.52
Abs	FT†	40.56	25.35	34.73
	FT+R†	42.05	27.00	36.56
	T-DMCA†	40.77	25.60	34.90
	HT†	41.53	26.52	35.76
	GraphSum†	42.63	27.70	36.97
	GraphSum+R†	**42.99**	27.83	37.36
	FT+R-20*	39.79	24.39	33.87
	T-DMCA-20*	38.64	21.25	29.77
	HT-20*	37.46	24.71	33.36
	EMSum	42.40	27.97	37.28
	EMSum+R	42.93	**28.11**	**38.19**

Table 1: Evaluation Results on WikiSum using ROUGE scores. The results of models with '†' are taken from Li et al. (2020). '$*$' indicates the results are obtained by running the released code. Model name with suffix '+R' means RoBERTa is used. 'Ext' means extractive methods, 'Abs' means abstractive methods.

number of iterations $t = 2$ based on the performance. We use dropout with probability 0.1 before all linear layers and label smoothing (Szegedy et al., 2016) with smoothing factor 0.1. We train our model for 200,000 steps with gradient accumulation every four steps. During decoding we apply beam search with beam size 5 and length penalty (Wu et al., 2016) with factor 0.4.

For models with pre-trained LMs, we choose the base version of RoBERTa. We follow Liu and Lapata (2019b), employing two Adam optimizers (Kingma and Ba, 2015) for the pre-trained part and other parts, with $\beta_1 = 0.9$, $\beta_2 = 0.998$. For the pre-trained part, the learning rate and warmup steps are set as 0.002 and 20,000, while for other parts are 0.2 and 8,000, respectively.

4.2 Baseline Models

We choose a series of Transformer-based models for comparison due to their excellent performance. Flat Transformer (**FT**) is a 6-layer encoder-decoder model. The title and ranked paragraphs were concatenated and truncated to 800 tokens. Transformer Decoder with Memory Compressed Attention model (**T-DMCA**) is proposed by Liu et al. (2018) with the WikiSum dataset. They use a Transformer decoder but apply a convolutional layer to compress the key and value in self-attention. Moreover, we choose Hierarchical Transformer (**HT**) proposed by Liu and Lapata (2019a), **GraphSum** proposed by Li et al. (2020), and **HeterSumGraph** proposed by Wang et al. (2020a) for comparisons. We have introduced them in Section 2.

	Model	R-1	R-2	R-L
Ext	LexRank†	41.77	s13.81	37.87
	HeterSumGraph†	46.05	16.35	**42.08**
Abs	FT*	43.28	14.59	20.39
	FT+R*	43.10	15.32	21.66
	HT*	42.03	15.18	22.79
	GraphSum†	45.02	16.69	22.50
	GraphSum+R†	46.07	17.42	23.21
	EMSum	45.57	17.71	26.43
	EMSum+R	**46.89**	**18.26**	27.55

Table 2: Evaluation Results on MultiNews using ROUGE scores. The results of models with '†' are taken from Li et al. (2020) or Wang et al. (2020a). '*' indicates the results are obtained by running the released code.

4.3 Results

Results on WikiSum Table 1 summarizes the evaluation results on the WikiSum dataset. The first block shows the baseline model Lead and LexRank (Erkan and Radev, 2004), which are extractive methods. The second block shows the results of abstractive models introduced in Section 4.2. We report their results following Li et al. (2020). The last block shows the results of some abstractive models and our model, but such models are fed with 20 top-ranked paragraphs as input.

The results show that if we limit the number of input paragraphs to 20, ROUGE score of all models will drop by about 2 points. We believe this is because the lower-ranked paragraphs can still provide information anyway.

Our model EMSum performs the best under the top-20 setting. Compared to the reported results of GraphSum (which used top 40 documents), EMSum achieves improvements on ROUGE-2 and ROUGE-L, even though EMSum takes shorter source documents as input. The gap between EMSum and GraphSum on ROUGE-1 score is 0.23 (42.40 vs 42.63). Considering all these three metrics together, the results show the effectiveness of our model.

For models combined with pre-trained LMs, the results show that EMSum+RoBERTa further improves the summarization performance on all metrics over EMSum. The improvements over GraphSum+RoBERTa are 0.28 on ROUGE-2 and 0.83 on ROUGE-L, also showing the effectiveness of our model even in the presence of pre-trained LMs.

Results on MultiNews Table 2 summarizes the evaluation results on the MultiNews dataset. Similarly, the first block shows two extractive baselines LexRank, and HeterSumGraph. The second block shows the abstractive methods. We report the results of FT, HT and GraphSum following Li et al. (2020). The last block shows the results of our models. We can see that EMSum outperforms GraphSum and EMSum+RoBERTa outperforms GraphSum+RoBERTa. HeterSumGraph is a extractive method so it achieves better ROUGE-L score. However, our model still achieves higher ROUGE-1 and ROUGE-2 scores than HeterSumGraph. Overall, the results demonstrate the effectiveness of our model on different types of corpora.

4.4 Analysis

We further conduct experiments to analyze the effects of the number of iterations and the number of paragraphs selected for attention. We also conduct ablation studies to validate the effectiveness of different components of our model.

The Number of Iterations We investigate how the number of iterations t influences the performance of our model. To this end, we conduct experiments on WikiSum dataset when $t = 1, 2, 3, 4$. The first block in Table 3 shows the results. Intuitively, the more iterations the graph is updated, the more information is flowed across the nodes. However, the results show us that $t = 3, 4$ outperforms $t = 2$ on ROUGE-L and the overall performance $\tilde{R}$ fluctuates very little. We argue the performance is limited by the number of introduced parameters. Therefore we choose $t = 2$ finally.

The Number of Paragraphs Selected for Attention At each decoding step, our two-level attention mechanism firstly computes weights over entity nodes to identify the most salient parts of source documents. The attention weights over the entire long token sequence may be sparse. So we need to figure out how much salient information is enough for our model, namely the proper value of k. We conduct experiments on WikiSum dataset when $k = 5, 10, 15, 20$. As the results in the second block of Table 3 show, when $k = 5$, the number of attended paragraphs is relatively small thereby degrading performance heavily. When $k = 20$, that means we do not perform any cut-off but only modify the paragraph attention weights with the entity attention weights, so the performance is also reduced. When $k = 10, 15$, the cut-off strategy works and boosts the performance. Finally, we choose $k = 10$ because it performs the best.

k	t	R-1	R-2	R-L	R̃
10	1	41.61	27.74	37.59	35.65
	2	42.93	28.11	38.19	**36.20**
	3	42.58	27.29	38.50	36.12
	4	42.51	27.18	38.76	36.15
5	2	37.75	21.87	31.44	30.35
10		42.93	28.11	38.19	**36.20**
15		42.26	27.36	38.28	35.97
20		40.33	26.17	36.40	34.30

Table 3: Results on different number of iterations t and different number of paragraphs k for attention on WikiSum dataset. R̃ is the mean of R-1, R-2 and R-L.

Model	R-1	R-2	R-L
EMSum	42.40	27.97	37.28
w/o graph enc	39.47	25.18	29.93
w/o two-level attn	40.51	25.31	31.21

Table 4: Ablation study of our model on WikiSum.

Ablation Study To validate the effectiveness of individual components such as graph encoder module and two-level attention module, we conduct experiments of ablation studies. For experiments without graph encoder module, we simply fix the entity cluster representation and paragraph representation after the multi-head pooling layer. For experiments without two-level attention, we apply token-level attention directly, but attend to the entity cluster representation additionally, which is a naive way to incorporate entity information. Table 4 shows the results. The results show the effectiveness of our new introduced module. Incorporating entity information to construct a heterogeneous graph network enables better information flowing between text nodes, and our design of the novel two-level attention mechanism in this task is indeed playing an important role towards the overall effectiveness of our approach.

4.5 Human Evaluation

We further employ human evaluation to assess model performance. We randomly sampled 20 documents-summary pairs from the WikiSum test set and 20 from the MultiNews test set, and invited 3 participants to assess the outputs of different models independently. Following criteria used by previous work (Liu and Lapata, 2019a), the evaluation score takes three aspects into account: (1) Informativeness: does the summary include salient

Dataset	Model	Rating
WikiSum	FT	-0.517
	T-DMCA	-0.117
	HT	0.250
	EMSum	0.383
MultiNews	FT	-0.650
	T-DMCA	-0.033
	HS	0.317
	EMSum	0.367

Table 5: Human evaluation results on summary quality rating. FT, T-DMCA, HT, HS are baseline models explained in Section 4.2.

parts of the input? (2) Fluency: Is the summary fluent and grammatical? (3) Succinctness: does redundancy occur in the summary? We used Best-Worst Scaling (Louviere et al., 2015) because it has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2017). Annotators are presented with the gold summary and summaries generated from 3 out of 4 systems and decide which summary is the best and which is the worst based on the criteria mentioned above. The rating of each system was computed as the percentage of times it was chosen as best minus the times it was selected as worst. Ratings range from -1 (worst) to 1 (best).

On the WikiSum dataset, we choose FT, T-DMCA, HT, EMSum and conduct human evaluation to compare their performance. On the MultiNews dataset, we choose FT, T-DMCA, HS, together with EMSum. The results are shown in Table 5. These results show that our EMSum model is able to generate summaries of higher quality than other models and further show the effectiveness of our proposed approach.

5 Conclusion

In this paper, we propose an entity-aware multi-document summarization model. We introduce entity nodes in addition to text unit nodes to construct a heterogeneous graph, helping our model capture complicated relations between text units. We also introduce a decoder with a two-level attention mechanism, which firstly attends to the entity nodes, where the attention weights are then subsequently utilized to guide the attention to the text units. With such a novel design, our model is able to deal with the problems of saliency and redundancy explicitly. Moreover, like other Transformer-based models, our model can be easily integrated with pre-trained language models for improved results. Experiments on standard datasets show the effectiveness of our model.

In the future, we would like to explore other approaches such as reinforcement learning based methods (Sharma et al., 2019) to further improve the summary quality in the context of multi-document summarization. We would also like to apply our method to other tasks such as multi-document question answering (Joshi et al., 2017).

Acknowledgments

We thank all the anonymous reviewers for their valuable suggestions. This research work was supported by the National Natural Science Foundation of China (Grant No.61772337, U1736207). Part of this work was done when Wei Lu was a visiting Professor at SJTU.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450.*

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1208–1214.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704.*

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1587–1597, Beijing, China. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of* *the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1173, Atlanta, Georgia. Association for Computational Linguistics.

Trevor Anthony Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale Seq2Seq models to multi-document inputs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4186–4196, Hong Kong, China. Association for Computational Linguistics.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 177–185.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Beliz Gunel, Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2020. Mind the facts: Knowledge-boosted coherent abstractive text summarization. *arXiv preprint arXiv:2006.15435*.

Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5094–5107, Online. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 465–470, Vancouver, Canada. Association for Computational Linguistics.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141, Brussels, Belgium. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Wei Li. 2015. Abstractive multi-document summarization with semantic information extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913, Lisbon, Portugal. Association for Computational Linguistics.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243, Online. Association for Computational Linguistics.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with structural regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4078–4087, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019b. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Congbo Ma, Wei Emma Zhang, Mingyu Guo, Hu Wang, and Quan Z Sheng. 2020. Multi-document summarization via deep learning techniques: A survey. *arXiv preprint arXiv:2011.04843*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Laura Perez-Beltrachini, Yang Liu, and Mirella Lapata. 2019. Generating summaries with topic templates and structured convolutional decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5107–5116, Florence, Italy. Association for Computational Linguistics.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-IE patterns for abstractive summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 892–901, Baltimore, Maryland. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Eva Sharma, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3280–3291, Hong Kong, China. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Xiaojun Wan. 2008. An exploration of document impact on graph-based multi-document summarization. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Honolulu, Hawaii. Association for Computational Linguistics.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020a. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405, Sofia, Bulgaria. Association for Computational Linguistics.

Zhengjue Wang, Zhibin Duan, Hao Zhang, Chaojie Wang, Long Tian, Bo Chen, and Mingyuan Zhou. 2020b. Friendly topic assistant for transformer based abstractive summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 485–497, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada. Association for Computational Linguistics.

Jianmin Zhang, Jiwei Tan, and Xiaojun Wan. 2018. Towards a neural network approach to abstractive multi-document summarization. *arXiv preprint arXiv:1804.09010*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Xin Zheng, Aixin Sun, Jing Li, and Karthik Muthuswamy. 2019. Subtopic-driven multi-document summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3153–3162, Hong Kong, China. Association for Computational Linguistics.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020. Boosting factual correctness of abstractive summarization with knowledge graph. *arXiv preprint arXiv:2003.08612*.

Yanyan Zou, Xingxing Zhang, Wei Lu, Furu Wei, and Ming Zhou. 2020. Pre-training for abstractive document summarization by reinstating source text. In *Proc. of EMNLP*.

LenAtten: An Effective Length Controlling Unit For Text Summarization

Zhongyi Yu[*,1], **Zhenghao Wu**[*,1], **Hao Zheng**[1],
Zhe XuanYuan[2,3], **Jefferson Fong**[1], **Weifeng Su**[†,1,3]
[1]Computer Science and Technology Programme, [2]Data Science Programme
Division of Science and Technology
[3]Guangdong Key Lab of AI and Multi-Modal Data Processing
BNU-HKBU United International College, Zhuhai, China
`{zhongyicst,zhecwu,zhslzwd97}@gmail.com`
`{zhexuanyuan,jeffersonfong,wfsu}@uic.edu.cn`

Abstract

Fixed length summarization aims at generating summaries with a preset number of words or characters. Most recent researches incorporate length information with word embeddings as the input to the recurrent decoding unit, causing a compromise between length controllability and summary quality. In this work, we present an effective length controlling unit Length Attention (LenAtten) to break this trade-off. Experimental results show that LenAtten not only brings improvements in length controllability and ROGUE scores but also has great generalization ability. In the task of generating a summary with the target length, our model is 732 times better than the best-performing length controllable summarizer in length controllability on the CNN/Daily Mail dataset. [1]

1 Introduction

Automatic text summarization aims at generating a short and coherent summary from one or multiple documents while preserving the main ideas of the original documents. Building upon the conventional summarization task, fixed length text summarization (FLS) demands extra focus on controlling the length of output summaries. Specifically, it requires generating summaries with a preset number of characters or words.

FLS is a rising research topic required in many scenarios. For example, in order to get universal user experiences on multiple platforms and devices, titles and abstracts for news articles are expected to have different numbers of characters. Instead of manually rewriting summaries, FLS can automatically generate required summaries by simply inputting the desired output length. Besides, FLS can help news editors to reduce post-editing time (Makino et al., 2019) and further improve summary quality (Liu et al., 2018; Makino et al., 2019). Last but not least, as shown in Table 1, with FLS, users can get customizable summaries by setting different desired lengths.

Source document	
egyptian president hosni mubarak arrived here friday morning to discuss the latest developments of iraqi crisis with his turkish counterpart suleyman demirel .	
Reference summary	
egyptian president to discuss iraqi crisis with turkish counterpart	
Model	***Summary***
PAULUS	egyptian president arrives in ankara
PAULUS+LA2 (GT)	mubarak arrives in ankara for talks on iraqi crisis with turkish pm
PAULUS+LA2 (30)	egyptian president arrives in ankara
PAULUS+LA2 (50)	egyptian president arrives in ankara for talks on iraq crisis
PAULUS+LA2 (70)	mubarak arrives in ankara for talks on iraqi crisis with turkish president demirel

Table 1: Output examples from the proposed method Length Attention (LA) on the Annotated English Gigaword dataset. Numbers in the parentheses represent different desired lengths. (GT) means the desired length is equal to the number of characters in the reference summary. PAULUS (Paulus et al., 2018).

Despite the benefits that could be brought, previous studies on FLS are very limited. Recent researches in FLS apply length information to either (i) the decoder (Kikuchi et al., 2016; Liu et al., 2018; Takase and Okazaki, 2019) or (ii) the optimization objective function (Makino et al., 2019). Though these systems are promising, they have to make a compromise between length controllability and summary quality. Kikuchi et al. (2016); Makino et al. (2019) can generate high-quality summaries, but perform inadequately at controlling length. Liu et al. (2018); Takase and Okazaki (2019) control the output length accurately, but these models suffer from producing summaries with low ROUGE scores.

---

* Equal Contribution

† Corresponding author

[1] Code are publicly available at: `https://github.com/X-AISIG/LenAtten`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 363–370
August 1–6, 2021.

In this paper, we present an effective length controlling unit, Length Attention (LenAtten). With LenAtten, summarizers can generate high-quality summaries with a preset number of characters, successfully breaks the trade-off between length controllability and summary quality.

Our contributions in this work are as follows: (1) A novel length controlling unit with great generalization capability is proposed to make summarizers generate high-quality summaries with a preset number of characters. (2) Experimental results show that LenAtten can break the trade-off between length controllability and summary quality. The length controllability of the proposed method is the new state-of-the-art on the examined datasets, to our knowledge.

2 Related Work

There are two types of approaches for text summarization: the extractive approach and the abstractive approach. Extractive approaches generate summaries by extracting words or sentences from the original text (Dorr et al., 2003; Nallapati et al., 2017; Liu and Lapata, 2019; Zhong et al., 2020), while abstractive approaches produce novel words or phrases (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Gu et al., 2016; See et al., 2017; Fan et al., 2018; Liu and Lapata, 2019).

Derived from the works in general text summarization, two approaches have been developed for FLS: (1) Incorporating length information into the decoder. LenInit proposed in Kikuchi et al. (2016) introduced length information into the initialization stage of a LSTM decoder. Liu et al. (2018) follows a similar approach as LenInit, but it's based on a CNN sequence-to-sequence architecture. Other studies exploit length information in each decoding step. LenEmb introduced in Kikuchi et al. (2016) generates a learnable embedding for each target length, and uses it as an additional input to its decoder. Takase and Okazaki (2019) extended Transformer's sinusoidal positional encoding (Vaswani et al., 2017) to make summarizers take account of stepwise remaining length during prediction. (2) Leveraging length information in global optimization methods. Makino et al. (2019) proposed a global optimization method named GOLC. GOLC incorporates length information with the minimum risk training (MRT) optimization method.

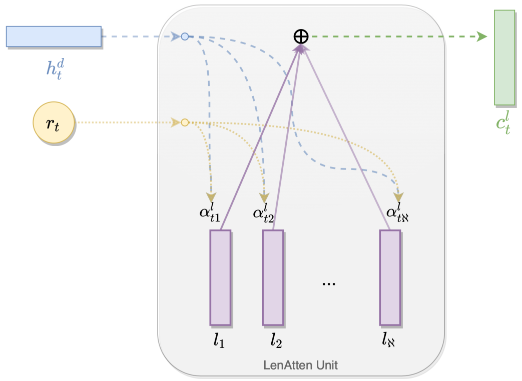

Figure 1: Illustration of the Length Attention Unit. Firstly, decoder hidden state (blue) and remaining length (yellow) are employed to compute the attention weights a^l. Then, the length context vector c_t^l (green) is produced by calculating the weighted sum between attention weights and pre-defined length embeddings (purple). Better viewed in color.

3 Our Approach: Length Attention

The motivation of LenAtten is to separate length information from the input of the recurrent decoding unit and to exploit proper length information based on the stepwise remaining length. As shown in Figure 1, at each decoding step, a length context vector is generated by calculating the weighted sum of a set of pre-defined embedding vectors l_*. Then, the length context vector is concatenated with the decoder hidden state and other attention vectors and fed to the input of the word prediction layer (details are shown in §4.2), so that summarizers can take the remaining length into account. The length context vector c_t^l at t-th decoding step is defined as follows:

$$c_t^l = \sum_{j=1}^{\aleph} \alpha_{tj}^l \, l_j \tag{1}$$

$$\alpha_{tj}^l = \frac{exp(e_{tj}^l)}{\sum_{k=1}^{\aleph} exp(e_{tk}^l)} \tag{2}$$

$$e_t^l = V_l^T \, tanh(W_l \, h_t^d + w_r \, r_t + b_l), \tag{3}$$

where $e_t^l \in \mathbb{R}^{\aleph \times 1}$, α_{tj}^l is the length attention score on the j-th length embedding at the t-th decoding step, h_t^d is the decoder hidden state, and V_l, W_l, w_r, b_l are learnable parameters. r_t is a scalar representing the remaining length at the current decoding step and $\aleph$ is a hyperparameter indicating the number of pre-defined length embeddings.

For length embeddings, we adopt the positional encoding proposed in Vaswani et al. (2017). We keep the embeddings fixed to remove the bias brought by the length distribution of data. The j-th length embedding l_j is defined as follows:

$$l_j = \begin{cases} 0 & j = 1 \\ PE(j-1) & otherwise, \end{cases} \quad (4)$$

where $PE(\,\cdot\,)$ is the positional encoding.

At the t-th decoding step, the remaining length r_t is updated by subtracting the length of the previously generated token. For the first decoding step, r_1 is initialized with desired output length. Following equations are used when $t > 1$:

$$r_t = \begin{cases} 0 & r_{t-1} - L(y_{t-1}) \leq 0 \\ r_{t-1} - L(y_{t-1}) & otherwise \end{cases} \quad (5)$$

where $L(y_{t-1})$ returns the number of characters in the output word y_{t-1}.

4 Experiment

4.1 Experimental Settings

We evaluate LenAtten on the CNN/Daily Mail dataset (See et al., 2017) to compare it with previous studies. In addition, we test LenAtten with short articles and summaries on the Annotated English Gigaword dataset (Rush et al., 2015). By default, all models are trained with maximum likelihood estimation (MLE) on a NVIDIA TITAN RTX GPU.[2]

For evaluation metrics, we adopt the standard F1 score of ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) to evaluate summary quality. For evaluating models' ability to control the output sequence length, we follow (Makino et al., 2019) to compute (1) character-level length variance Var between reference summaries and generated summaries and (2) over-length ratio $\%over$, which measures how many of the generated summaries are longer than their reference summaries. The length variance Var is computed as follows:

$$Var = 0.001 * \frac{1}{n}\sum_{i=1}^{n} |len(y_i) - len(y_i')|^2 \quad (6)$$

where y_i is the reference summary, y_i' is the generated summary, and $len(\cdot)$ returns the number of characters in the given summary. For the FLS task, the length variance Var is expected to be zero as it indicates the lengths of output summaries are exactly the desired summary lengths.

4.2 Methods to be compared

We compare the proposed Length Attention unit with following methods:

LEAD-3 extracts the first three sentences of source articles as the summary.

PG is the standard pointer-generator network proposed in See et al. (2017).

MASS (Song et al., 2019) is a sequence to sequence pre-trained model based on Transformer.

LenAtten is also compared with length controllable summarization methods. For a fair comparison, we choose methods that also aim at generating summaries with a preset number of characters in a word-by-word manner.

LE is the LenEmb method proposed in Kikuchi et al. (2016).

GOLC is a global optimization method introduced in Makino et al. (2019).

We apply LenAtten to three summarization models:

S2S (RNN-based Seq2Seq Model) is a vanilla encoder-decoder summarizer. Specifically, we adopt a Bi-LSTM as the encoder and a unidirectional LSTM as the decoder. To integrate LenAtten, the length context vector c_t^l is added to the input of the word prediction layer to produce the vocabulary distribution P_{vocab}:

$$P_{vocab} = softmax\Big(W[h_t^d||c_t^l||y_{t-1}||C] + b\Big) \quad (7)$$

where W, b are learnable parameters, h_t^d is the decoder hidden state, y_{t-1} is the word embedding of the last generated token, "||" is the vector concatenation operator. C is the last encoder hidden state, which is known as the fixed context vector.

PAULUS (Copying Mechanism) follows the design of Paulus et al. (2018), which incorporates two attention modules and the copying mechanism into a Seq2seq summarizer. To integrate LenAtten, the vocabulary distribution P_{vocab} is calculated using:

$$P_{vocab} = softmax\Big(W[h_t^d||c_t^l||c_t^e||c_t^d] + b\Big) \quad (8)$$

where c_t^e and c_t^d are the context vectors generated from the encoder and decoder attention units.

ATTENTION (Attention-based model) is implemented by removing copying mechanism from **PAULUS**. For the above-mentioned three models, we remove the length context vector c_t^l in the ablation study and keep other components unchanged.

---

[2] Detailed model configurations are provided in the Appendix.

CNN/DM					
Models	R-1 F	R-2 F	R-L F	Var (↓)	$\%over$
Baseline					
*LEAD-3	40.34	17.70	36.57	-	-
†MASS	41.38	19.11	38.42	-	-
‡PG	37.74	15.78	34.35	19.35	58.35
‡PG + LE(MLE)	37.45	15.31	34.28	4.5	19.11
‡PG + LE(GOLC)	38.27	16.22	34.99	5.13	6.70
S2S	19.38	3.58	14.35	89.74	7.00
ATTENTION	34.32	13.76	28.92	48.99	18.11
PAULUS	38.10	16.42	33.17	19.91	37.14
with **Length Attention** ($\aleph = 2$)					
S2S	21.09	3.93	16.79	**0.0069**	30.67
ATTENTION	36.53	14.21	32.63	0.0075	38.72
PAULUS	**39.82**	**17.31**	**36.20**	0.0070	57.75

AEG					
Models	R-1 F	R-2 F	R-L F	Var (↓)	$\%over$
Baseline					
S2S	36.99	16.03	33.01	0.3902	30.12
ATTENTION	42.55	21.54	38.72	0.3285	25.89
PAULUS	43.84	**22.80**	40.12	0.3058	16.01
PAULUS+LE	40.02	17.31	36.99	0.0500	0.4
with **Length Attention** ($\aleph = 2$)					
S2S	38.26	16.24	35.11	0.0043	35.45
ATTENTION	43.15	21.51	40.32	0.0044	50.15
PAULUS	**43.92**	**22.80**	**41.16**	**0.0042**	37.75

Table 2: Results on CNN/DM and AEG dataset. If not specified in the parentheses, the training objective function is MLE by default. Results retrieved from: * See et al. (2017); † Xu et al. (2020); ‡ Makino et al. (2019).

Models	CNN/DM	AEG
S2S	5.450	3.624
S2S+LA2	**5.360**	**3.393**
ATTENTION	4.317	3.279
ATTENTION+LA2	**4.278**	**3.074**
PAULUS	3.478	3.085
PAULUS+LA2	**3.391**	**2.899**

Table 3: Test perplexity of models on the CNN/DM dataset and the AEG dataset.

4.3 Experimental Results [3]

Reference Summary Lengths In this experiment, we evaluate our model by comparing it with previous works. The desired length is set as the number of characters in corresponding reference summaries. Table 2 shows that LenAtten has superior length controllability and higher ROUGE scores on both datasets. Specifically, the length variance (Var) of LenAtten is 732 times better than the best-performing length controllable method PG+LE(GOLC) in the CNN/DM dataset. Besides, adding LenAtten can boost ROUGE scores by 1-3 points. We believe the improvement in the ROUGE scores comes from the introduction of length information (i.e. the desired length information). The desired length information can be viewed as an inductive bias, which helps summarizers prefer some of the outputs over others. Under the same context, by conditioning on the desired output length, summarizers may prefer candidate summaries with output lengths similar to the desired length. Thus, summarizers can learn a better alignment with the reference summaries during training and outputs summaries with higher ROUGE scores in inference.

In addition, previous length controllable methods control the output lengths at the cost of damaging the ROUGE scores. The ROUGE scores of PG and PAULUS drop after adding LenEmb (i.e. PG + LE(MLE) and PAULUS + LE). In comparison, LenAtten not only performs better at reducing the length variance Var but also significantly improves ROUGE scores. This suggests that LenAtten can break the trade-off between summary quality and length controllability.

After integrating with LenAtten, the $\%over$ ratio of summarizers rises. This suggests that more of the generated summaries ended up being longer than the references. We believe this is because when the remaining length is small (e.g. 4 characters) but not 0, instead of stopping the generation process, summarizers with LA tend to generate more tokens to meet the length requirement. Since summarizers output a word at each inference step, they may select a word that's longer than the remaining length. Thus, the generated summaries may end up being longer than the references.

Perplexity To figure out how LenAtten affects the performance of the language model, we examine the log-perplexity of models on the test sets. Perplexity is a commonly-used metric for evaluating language models. A lower perplexity score indicates better language model performance. In this experiment, the desired length is set to the reference summary length. As shown in Table 3, after adding LenAtten, log-perplexity drops consistently on both datasets. This suggests that the adding of LenAtten can boost language model performance.

Various Preset Lengths In this experiment, we test the generalization ability of LenAtten under various desired lengths. For the AEG dataset, most reference summaries contain 30-75 characters, and

---

[3] For all the experiments, the number following LA (the LenAtten unit) represents the number of length embeddings (i.e. the value of $\aleph$).

AEG	30	50	75	100	120
R-1 F	37.89	42.67	41.13	39.03	37.45
R-2 F	18.87	21.01	18.99	17.16	16.00
R-L F	33.64	38.76	35.27	31.95	29.84
Var	0.0027	0.0024	0.0026	0.0030	0.0040

CNN/DM	100	200	400	800	1600
R-1 F	30.88	37.90	39.52	37.25	34.23
R-2 F	13.31	16.26	16.58	15.08	13.50
R-L F	23.17	32.95	33.99	29.34	24.54
Var	0.0067	0.0063	0.0058	0.0054	0.0051

Table 4: ROUGE scores and Length Variance Var of PAULUS+LA2 under different desired lengths.

few of them are more than 100 characters. For the CNN/DM dataset, most reference summaries are 100-750 characters. Thus, the desired length is set as 30, 50, 75, 100, and 120 for the AEG dataset and 100, 200, 400, 800, and 1600 for the CNN/DM dataset. We add the LenAtten unit to PAULUS and exploit full reference summaries to get ROUGE scores.

As shown in Table 4, on the AEG dataset, for frequently appeared lengths (30, 50, 75), and lengths that are exceptionally long (100, 120), LenAtten demonstrates great length controllability along with good ROUGE scores. Same conclusions can be drawn on the CNN/DM dataset. This shows that LenAtten has great generalization ability under various desired lengths.

Exploring Hyperparameter $\aleph$ In this experiment, we analyze how different $\aleph$ (the number of pre-defined length embeddings) affect the performance of LenAtten on the AEG dataset. Desired lengths are set to the lengths of reference summaries. Figure 2 shows the length controllability becomes better as the increase of $\aleph$, with no harm to the ROUGE-L scores.

5 Conclusions

In this paper, we present a novel length controlling unit, LenAtten, to help summarization models generate quality summaries with a preset number of characters. On the examined datasets, LenAtten outperforms length controllable summarization baselines steadily in terms of length controllability and demonstrates great generalization ability. LenAtten also breaks the trade-off between length controllability and summary quality. To our knowledge, in the task of generating summaries with target lengths, LenAtten is the new state-of-the-art on the CNN Daily Mail dataset.

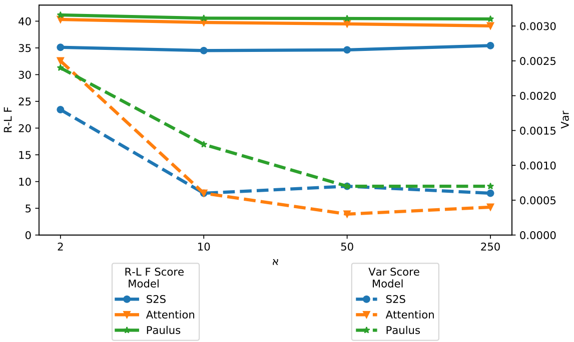

Figure 2: Examining hyperparameter $\aleph$ on the AEG dataset. ROUGE-L F1 scores and Length Variance Var of different models under different $\aleph$ are shown ($\aleph = 2, 10, 50, 250$).

Acknowledgment

We thank friends and colleagues at the UIC-AISIG as well as Minyong Li for their assistance with the study. We would like to thank anonymous reviewers for their helpful comments. This work is supported by the BNU-HKBU United International College research grant.

References

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 1–8.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1328–1338, Austin, Texas. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4110–4119, Brussels, Belgium. Association for Computational Linguistics.

Takuya Makino, Tomoya Iwakura, Hiroya Takamura, and Manabu Okumura. 2019. Global optimization under length constraint for neural text summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1039–1048, Florence, Italy. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI Conference on Artificial Intelligence, 31(1)*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3999–4004, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

A Appendix

A.1 Experimental settings

Model Configuration In our experiments, the dimension of LSTM hidden state is set as 256, and the vocabulary size is 100,000. Word embeddings are fixed 300-dimensional GloVe vectors[4]. If a word is not covered in the GloVe, a random 300-dimensional vector is used. During training, Adam optimizer is applied with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and learning rate $\alpha = 0.001$. Besides, we set a 25% probability of choosing the previously generated token instead of the ground-truth token as y_{t-1} to reduce exposure bias. At test time, summaries are produced using beam search with beam size 4. We use a fully python implemented library[5] to obtain the ROUGE score.

Dataset Distribution We plot the length distribution of reference summaries in the AEG (Figure 3) and the CNN/DM (Figure 4) dataset.

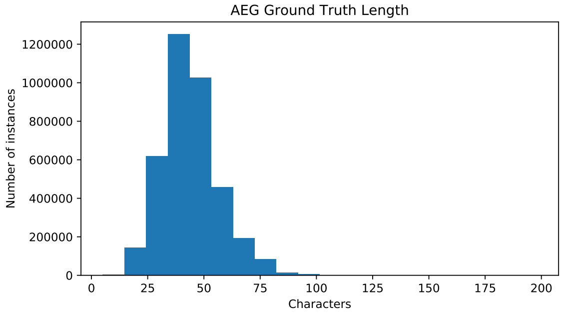

Figure 3: Length distribution of reference summaries on the Annotated English Gigaword dataset. Summaries with 30 to 75 characters cover the majority cases.

A.2 Additional Experiments

Semantic Similarity Another automatic evaluation metric *BertScore* (Zhang et al., 2019) recall score is used to measure the semantic similarity between system outputs and reference summaries. As shown in Figure 5, models with Length Attention module (LA2) outperform baselines (FREE) on both datasets.

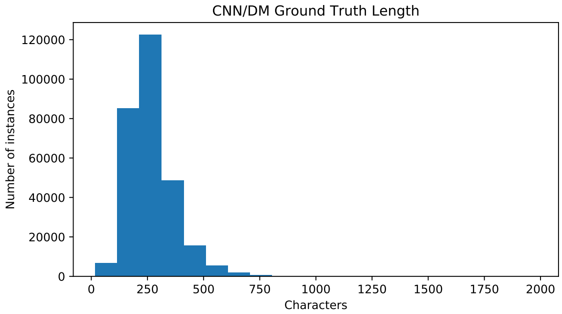

Figure 4: Length distribution of reference summaries on the CNN/Daily Mail dataset. Summaries exceed 2000 characters are ignored, since they only cover 0.009% of the dataset.

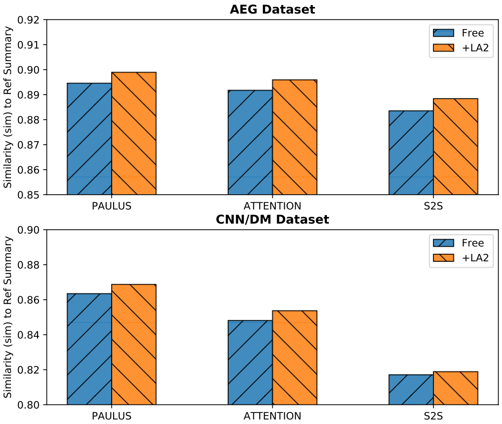

Figure 5: Semantic similarity between model outputs and reference summaries. Desired length is set as reference summary length.

Human Evaluation Correctness (CORR), completeness (COMP), and fluency (FLUE) of system outputs are assessed through 2 human evaluations. We randomly select 10 samples from each dataset. 30 skilled English speakers are presented with the original article and two summaries. One of the summaries is from the model without LenAtten, and the other one is from the same model plus LenAtten (e.g., S2S and S2S+LA2). The evaluation process is well-designed to prevent participants from knowing the source of the presented summaries. Models without LenAtten generate summaries without length restriction, and models with LenAtten are required to output summaries with desired lengths. There are 467 feedbacks collected for the first experiment and 160 for the second.

In the first experiment (Table 5), participants are asked to choose a better one from two given summaries. The desired length is set as the length of the reference summary.

In the second experiment (Table 6), we only examine PAULUS and PAULUS+LA2. The desired

---

[4] http://nlp.stanford.edu/data/glove.6B.zip

[5] https://github.com/pltrdy/rouge

	COMP	CORR	FLUE
AEG			
S2S (Free)	27.1%	44.4%	42.9%
S2S+LA2 (GT)	**72.9%**	**55.6%**	**57.1%**
Attention (Free)	33.3%	44.2%	**56.9%**
Attention+LA2 (GT)	**66.7%**	**55.8%**	43.1%
Paulus (Free)	47.2%	**53.3%**	**61.4%**
Paulus+LA2 (GT)	**52.8%**	46.7%	38.6%
CNN/DM			
S2S (Free)	22.5%	22.9%	**53.8%**
S2S+LA2 (GT)	**77.5%**	**77.1%**	46.2%
Attention (Free)	21.9%	27.5%	36.7%
Attention+LA2 (GT)	**78.1%**	**72.5%**	**63.3%**
Paulus (Free)	**53%**	47.4%	48.1%
Paulus+LA2 (GT)	47%	**52.6%**	**51.9%**

Table 5: Results of the first Human Evaluation. "(Free)": model generates summaries freely. "(GT)": model generates summaries with the desired length set as the length of the reference summary.

	COMP	CORR	FLUE
AEG			
Paulus (Free, avg_len=57.46)	3.328	3.407	3.605
Paulus+LA2 (30)	3.000	3.250	3.392
Paulus+LA2 (50)	3.750	3.500	**3.687**
Paulus+LA2 (70)	**3.875**	**3.750**	3.562
CNN/DM			
Paulus (Free, avg_len=285.5)	3.250	3.345	3.273
Paulus+LA2 (150)	3.125	3.375	3.166
Paulus+LA2 (250)	3.451	**3.483**	**3.419**
Paulus+LA2 (350)	**3.827**	3.482	3.172

Table 6: Results of the second human evaluation. "(Free, avg_len)": model generates summaries freely. The average length of the generated summaries is also listed.

length is set as (30, 50, 70) on the AEG dataset and (150, 250, 350) on the CNN/DM dataset. Participants need to rate each summary from 0 to 5. In order to guarantee the accuracy and credibility of results, each article is presented once to each participant.

As shown in Table 5, models with LenAtten have better completeness and correctness scores on both datasets, along with a few improvements on the fluency. In the second experiment, Table 6 shows that (1) the completeness and correctness scores increase as the desired length increases. This trend is reasonable, since more information should be included in the final summary, as the summary length gets longer. This also suggests that, as the desired length gets longer, models with LenAtten can generate meaningful words instead of simply repeating one or two words. (2) When comparing the results of Paulus (Free, 57.46; 285.5) and Paulus+LA2 (50; 250), Paulus+LA2 outperforms the Paulus (Free) on all metrics. In other words, when the desired length gets smaller (but not too small), LenAtten can help summarization models to use concise words and phrases while maintaining summary quality.

Source document - A			
the indian union government thursday decided to increase customs duty on sugar to ## percent to curb cheap imports of the commodity , said a senior finance ministry official here .			
Reference summary - A			
india increases sugar import duty			
	Summary	R-1 R	Diff
Model	india to increase customs duty on sugar	60.00	4
Model + LA2	india to increase tariffs on sugar	40.00	0
Source document - B			
defending champion **albert costa** of spain reached the last eight in the french open here on monday , beating local favorite ##nd seed arnaud clement of france in straight sets .			
Reference summary - B			
costa enters last eight in french open			
	Summary	R-1 R	Diff
Model	**costa** into last eight in french open	85.71	-2
Model + LA2	**costa** into french open quarter-finals	42.85	-1

Table 7: Synonym substitution is colored in red. **R-1 R**: ROUGE-1 Recall score. **Diff**: len(output) - len(reference).

A.3 Output Examples

Synonym substitution When examining generated summaries, we find adding LenAtten can make summarizers replace long/short words with synonyms to meet the length requirement. Examples are showcased in Table 7.

XeroAlign: Zero-Shot Cross-lingual Transformer Alignment

Milan Gritta and **Ignacio Iacobacci**
Huawei Noah's Ark Lab, London, UK
`{milan.gritta,ignacio.iacobacci}@huawei.com`

Abstract

The introduction of transformer-based cross-lingual language models brought decisive improvements to multilingual NLP tasks. However, the lack of labelled data has necessitated a variety of methods that aim to close the gap to high-resource languages. Zero-shot methods in particular, often use translated task data as a training signal to bridge the performance gap between the source and target language(s). We introduce **XeroAlign**, a simple method for task-specific alignment of cross-lingual pretrained transformers such as XLM-R. XeroAlign uses translated task data to encourage the model to generate similar sentence embeddings for different languages. The XeroAligned XLM-R, called XLM-RA, shows strong improvements over the baseline models to achieve state-of-the-art zero-shot results on three multilingual natural language understanding tasks. XLM-RA performs on par with state-of-the-art models on a cross-lingual adversarial paraphrasing task and its text classification accuracy exceeds that of XLM-R trained with labelled data.

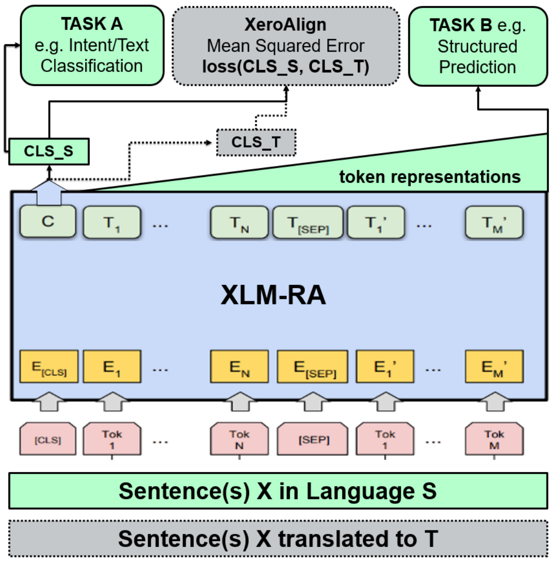

Figure 1: The XeroAligned XLM-R model (XLM-RA) for cross-lingual NLU. XeroAlign is added to the training loop to encourage embeddings in different languages to be similar, enabling a zero-shot classifier transfer.

1 Introduction

Recently, large transformer-based (Vaswani et al., 2017) pretrained language models have achieved state-of-the-art (SOTA) performance on several monolingual and multilingual NLP tasks (Wang et al., 2019a; Hu et al., 2020b). Transfer learning enabled the self-supervised pretraining on unlabelled datasets to learn linguistic features such as syntax and semantics in order to improve tasks with limited training data (Wang et al., 2019b). Pretrained cross-lingual language models (PXLMs) have soon followed to learn general linguistic features and properties of dozens of languages (Lample and Conneau, 2019; Xue et al., 2020). For multilingual tasks in particular, adequate labelled data is usually only available for a few well-resourced languages such as English. Zero-shot approaches were introduced to transfer the knowledge to languages without the requisite training data. To this end, we introduce **XeroAlign**, a conceptually simple and efficient method for task-specific alignment of sentence embeddings generated by PXLMs, aimed at effective zero-shot cross-lingual transfer. XeroAlign is an auxiliary training objective, which uses translated data to bring the performance in the target language closer to the source (labelled) language, as illustrated in Figure 1. We apply our proposed method to the publicly available XLM-R transformer (Conneau et al., 2020) but instead of pursuing large-scale model alignment with general parallel corpora such as Europarl (Koehn, 2005), we show that a simplified, task-specific model alignment is an effective and efficient approach to zero-shot transfer for cross-lingual natural language understanding (XNLU).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 371–381
August 1–6, 2021.

We evaluate XeroAlign on four datasets (two different tasks) that cover 11 unique languages. The XeroAligned XLM-R model (XLM-RA) achieves SOTA scores on three XNLU datasets, exceeds the text classification performance of XLM-R trained with labelled data and performs on par with SOTA models on an adversarial paraphrasing task.

2 Background

We first outline the necessary background on Contrastive Learning (Becker and Hinton, 1992) and transformer-based PXLMs. Contrastive learning (CL) is a framework designed for self-supervised representation learning. Recent examples include Momentum Contrast (MoCo) (He et al., 2020) and SimCLR (Chen et al., 2020), both of which achieved strong improvements on image classification. The essence of CL is to generate representations that are similar for positive examples and dissimilar for negative examples. CL-based methods in cross-lingual NLP replace negative samples (formerly augmented images) with random sentences in the target language, typically thousands of them. Positive examples comprise sentences translated into the target language. While CL is applicable to task-agnostic model alignment, large batches of random negative samples are infeasible for small datasets. Negative samples drawn randomly from a small dataset can result in duplicates (false negatives), which is why our alignment uses only positive samples. The CL-based alignments in section 2.1 aim to improve generic cross-lingual representations with large (parallel) datasets. In contrast, we align the PXLM only with translated task data, making our approach simpler and more efficient, showing a strong zero-shot transfer.

Transformer-based PXLMs For transformer-based PXLMs, two basic types of representations are commonly used: 1) A sentence embedding for tasks such as text classification (Conneau et al., 2018) or sentence retrieval (Zweigenbaum et al., 2018), which use the `[CLS]` representation of the full input sequence, and 2) Token embeddings, which are used for structured prediction (Pan et al., 2017) or Q&A (Lewis et al., 2020), requiring each token's contextualised representation for a per-token inference. While our method uses the `[CLS]` embedding, other approaches based on Contrastive Learning have used both types of representations to obtain a sentence embedding.

2.1 Related Work

In order to cluster prior work, we formulate a taxonomy (Table 1) for the purposes of positioning our method in the appropriate context. The relevant zero-shot transfer approaches can be grouped by a) whether the alignment is targeted at each task, i.e. is task-specific [TS] or is task-agnostic [TA] and b) whether the alignment is applied to the model [MA] or data [DA]. Our paper falls mostly into the [MA,TS] category although methodological similarities are also found in the [MA,TA] group.

Groups	*Task-Specific*	*Task-Agnostic*
Data Align	[DA,TS]	No relevant work
Model Align	[MA,TS]	[MA,TA]

Table 1: An approximate taxonomy of prior work.

[MA,TA] Hu et al. (2020a) have proposed two objectives for cross-lingual zero-shot transfer a) sentence alignment and b) word alignment. While CL is not mentioned, the proposed sentence alignment closely resembles contrastive learning with one encoder (e.g. SimCLR). Taking the average of the contextualised token representations as the input representation (as an alternative to the `[CLS]` token), the model predicts the correct translation of the sentence within a batch of negative samples. An improvement is observed for text classification tasks and sentence retrieval but not structured prediction. The alignment was applied to a 12-layer multilingual BERT and the scores are comparable to the translate-train baseline (translate data and train normally). Instead, we use one of the best publicly available models, XLM-R from Huggingface, as our starting point since an improvement in a weaker baseline is not guaranteed to work in a stronger model that may have already subsumed those upgrades. Contrastive alignment based on MoCo with two PXLM encoders was proposed by Pan et al. (2020). Using an L2 normalised `[CLS]` token with a non-linear projection as the input representation, the model was aligned on 250K to 2M parallel sentences with added Translation Language Modelling (TLM) and a code-switching augmentation. No ablation for MoCo was provided to estimate its effect although the combination of all methods did provide improvements with multilingual BERT as the base learner. Another model inspired by CL is InfoXLM (Chi et al., 2020). InfoXLM is pretrained with TLM, multilingual

Masked Language Modelling and Cross-lingual Contrastive Learning called XLCo. Like MoCo, they employ two encoders that use the `[CLS]` token (or the uppermost layer average) as the sentence representation, taken from layers 8 (base model) and 12 (large model). Ablation showed a 0.2-0.3 improvement in accuracy for XNLI and MLQA (Lewis et al., 2020). Reminiscent of earlier work on cross-lingual alignment (Hermann and Blunsom, 2014), the task-agnostic sentence embedding model (Feng et al., 2020) called LaBSe (Language-agnostic BERT sentence embeddings) uses the `[CLS]` representations of two BERT encoders (compared to our single encoder) with a margin loss and 6 billion parallel sentences to generate multilingual representations. While similarities exist, our multi-task alignment is an independently devised, more efficient, task-specific and a simplified version of the aforementioned approaches.

[DA,TS] Zero-shot cross-lingual models often make use of machine translation to provide a training signal. This is an uncomplicated transformation for text classification tasks given that adequate machine translation models exist for many language pairs. However, for structured prediction tasks such as Slot Filling or Named Entity Recognition, the non-trivial task of *aligning token labels* can lead to an improved cross-lingual transfer. One of the most commonly used word alignment methods is fastalign (Dyer et al., 2013). Frequently used as a baseline, it involves aligning the word indices in parallel sentences in an unsupervised manner, prior to regular supervised learning. In some scenarios, fastalign can approach SOTA scores for slot filling (Schuster et al., 2019), however, the quality of alignment varies between languages and can even degrade performance (Li et al., 2021) below baseline. An alternative data alignment approach called CoSDA (Qin et al., 2020) uses code-switching as data augmentation. Random words in the input are translated and replaced to make model training highly multilingual, leading to improved cross-lingual transfer. Attempts were also made to automatically learn how to code-switch (Liu et al., 2020). While improvements were reported, it's uncertain how much SOTA models would benefit.

[MA,TS] Continuing with label alignment for slot filling, Xu et al. (2020) tried to predict and align slot labels jointly during training instead of modifying data labels explicitly before fine-tuning. While soft-align improves on fastalign, the difficulty of label alignment makes it challenging to improve on the SOTA. For text classification tasks such as Cross-lingual Natural Language Inference (Conneau et al., 2018), an adversarial cross-lingual alignment was proposed by Qi and Du (2020). Adding a self-attention layer on top of multilingual BERT (Devlin et al., 2019) or XLM (Lample and Conneau, 2019), the model learns the XNLI task while trying to fool the language discriminator in order to produce language-agnostic input representations. While improvements over baselines were reported, the best scores were around 2-3 points behind the standard XLM-R model.

3 Methodology

We introduce **XeroAlign**, a conceptually simple and efficient method for task-specific alignment of sentence embeddings generated by PXLMs, aimed at effective zero-shot cross-lingual transfer. XeroAlign is an auxiliary training objective that is jointly optimised with the primary task, e.g. text classification and/or slot filling, as shown in Figure 1. We use standard architecture for each task and only add the minimum required number of new parameters. For text classification tasks, we use the `[CLS]` token of the PXLM as our pooled sentence representation. A linear classifier (hidden size x number of classes) is learnt on top of the `[CLS]` embedding using cross-entropy as the loss function (TASK A in Figure 1). For slot filling, we use the contextualised representations of each token in the input sequence. Once again, a linear classifier (hidden size x number of slots) is learnt with a cross-entropy loss (TASK B in Figure 1).

Algorithm 1 shows a standard training routine augmented with XeroAlign. Let $PXLM$ be a pre-trained cross-lingual transformer language model, X be the standard English training data and U be the machine translated utterances (from X). The English utterances were translated into each target language using our internal machine translation service. A public online translator e.g. Google Translate can also be used. For the PAWS-X task, we use the public version of the translated data[1]. We then obtain the CLS_S and CLS_T embeddings by taking the first token of the $PXLM$ output sequence for the source x_s and target x_t sentences respectively. Using a Mean Squared Error loss

---

[1] https://github.com/google-research-datasets/paws

Algorithm 1 The XeroAlign algorithm.

```
$PXLM \leftarrow$ Pretrained Cross-lingual LM
$X \leftarrow$ Training data tuples in English
$U \leftarrow$ Utterances translated into Target Lang.
$sim \leftarrow$ similarity function e.g. MSE

# training loop
for $(x_s, y), x_t \in X, U$ do
    $task\_loss \leftarrow task\_loss\_fn(x_s, y)$
    $CLS_S \leftarrow PXLM(x_s)$
    $CLS_T \leftarrow PXLM(x_t)$
    $align\_loss \leftarrow sim(CLS_S, CLS_T)$
    $total\_loss \leftarrow task\_loss + align\_loss$
    # update model parameters
end for
```

function as our similarity function sim, we compute the distance/loss between CLS_S and CLS_T. The sum of the losses ($total\_loss$) is then backpropagated normally. We have conducted all XeroAlign training as multi-task learning for the following reason. When the $PXLM$ is aligned first, followed by primary task training, the $PXLM$ exhibits poor zero-shot performance. Similarly, learning the primary task first, followed by XeroAlign fails as the primary task is partially unlearned during alignment. This is most likely due to the catastrophic forgetting problem (Goodfellow et al., 2013).

3.1 Experimental Setup

In order to make our method easily accessible and reproducible[2], we use the publicly available XLM-R transformer from Huggingface (Wolf et al., 2019) built on top of PyTorch (Paszke et al., 2019). We set a single seed for all experiments and a similar learning rate for each dataset. No hyperparameter sweep was conducted to ensure a robust, low-resource, real-world deployment and to make a fair comparison with SOTA models. XLM-R was XeroAligned over 10 epochs and optimised using Adam (Kingma and Ba, 2015) and a OneCycleLR (Smith and Topin, 2019) scheduler.

3.2 Datasets

We evaluate XeroAlign with four datasets covering 11 unique languages (en, de, es, fr, th, hi, ja, ko, zh, tr, pt) across two tasks (intent classification + slot filling, paraphrase detection).

PAWS-X (Yang et al., 2019) is a multilingual version of PAWS (Zhang et al., 2019), a binary classification task for identifying paraphrases. Examples were sourced from Quora Question Pairs[3] and Wikipedia, chosen to mislead simple 'word overlap' models. PAWS-X contains random examples drawn from PAWS (just under 4K for the development and test set) covering seven languages (en, de, es, fr, ja, ko, zh) and totalling almost 28,000 human translated paraphrases. Each train set contains approximately 49.5K machine translated examples.

MTOD is a Multilingual Task-Oriented Dataset provided by Schuster et al. (2019). It covers three domains (alarm, weather, reminder) and three languages of different sizes: English (43K), human-translated Spanish (8.3K) and Thai (5K). MTOD comprises two correlated NLU tasks, intent classification and slot filling. The SOTA scores are reported by Li et al. (2021) and Schuster et al. (2019).

MTOP is a Multilingual Task-Oriented Parsing dataset provided by Li et al. (2021) that covers interactions with a personal assistant (intent recognition and slot filling tasks). We use the standard flat version, which has the SOTA reported by Li et al. (2021). A compositional version of the data designed for nested queries is also provided. MTOP contains 100K+ human translated examples in 6 languages (en, de, es, fr, th, hi) and 11 domains.

MultiATIS++ by Xu et al. (2020) is an extension of the multilingual version of ATIS (Upadhyay et al., 2018), initially translated into Hindi and Turkish. Six new human-translated[4] languages (de, es, fr, zh, ja, pt) were added with ∼4 times as many examples each (around 6K per language) for 9 languages. These datasets are based on the English-only ATIS (Price, 1990) featuring users interacting with an information service. Once again, intent recognition and slot filling are evaluated.

3.3 Metrics

We use standard evaluation metrics, that is, accuracy for paraphrase detection and intent classification, F-Score[5] for slot filling.

---

[2] Resources are available by email (Milan Gritta) or download from `https://www.aclweb.org/anthology/` or `https://github.com/huawei-noah/xnlu`.

[3] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`

[4] We have encountered some minor issues with slot annotations. Around 60-70 entities across 5 languages (fr, zh, hi, ja, pt) had to be corrected as the number of slot tags did not agree with the number of tokens in the sentence. However, this only concerns a tiny fraction of the ∼400k+ tokens covered.

[5] `https://pypi.org/project/seqeval/`

Model	Spanish	French	German	Hindi	Thai	Average
XLM-R Target	95.9 / 91.2	95.5 / 89.6	96.6 / 88.3	95.1 / 89.1	94.8 / 87.7	95.6 / 89.2
XLM-R 0-shot	91.9 / 84.3	93.0 / **83.7**	87.5 / 80.7	91.4 / 76.5	87.6 / 55.6	90.3 / 76.2
XLM-RA	**96.6** / 84.4	**96.5** / 83.3	**95.7 / 84.5**	**95.2 / 80.1**	**94.1 / 69.1**	**95.6 / 80.3**
Li et al. (2021)	96.3 / **84.8**	95.1 / 82.5	94.8 / 83.1	94.2 / 76.5	92.1 / 65.6	94.5 / 77.9

Table 2: MTOP results as Intent Classification Accuracy / Slot Filling F-Score. Best English score: 97.3 / 93.9.

4 Results and Analysis

We use 'XLM-R Target' to refer to models trained on each labelled target language. We also provide zero-shot scores denoted 'XLM-R 0-shot', the 'XLM-RA' results and the reported SOTA figures. For PAWS-X, we provide a second baseline called 'Translate-Train', which comprises the union of Target and English train data. Scores are given for the large[6] model unless specified otherwise.

The XeroAligned XLM-R attains SOTA scores on 3 XNLU datasets. The intent classification (+1.1) and slot filling (+2.4) scores, averaged over 5 MTOP languages and shown in Table 2, improved on an XLM-R model with translated utterances, slot label projection and distant supervision (Li et al., 2021). For the MultiATIS++ dataset (Table 5), XLM-RA shows an improved intent accuracy (+1.1) and F-Score (+3.2) over 8 languages, compared to a large multilingual BERT with translated utterances and slot label softalign (Xu et al., 2020). The MTOD classification accuracy (+1.3) and slot tagging F-Score (+5.0) have improved over an XLM-R with translated utterances, slot label projection and distant supervision (Li et al., 2021). MTOD is the only dataset where the base model outperforms (albeit marginally) the large transformer, as seen in Table 6. Finally, we compare our intent classification accuracy (+8.1) and slot filling F-Score (+8.7) for MTOD to a BiLSTM with translated utterances and slot label projection (Schuster et al., 2019), which had the SOTA F-Score for Thai.

On the adversarial paraphrase task (PAWS-X, Table 3), averaged over 7 languages, XLM-RA scores marginally higher (+0.1 accuracy) than VECO (Luo et al., 2020), a variable cross-lingual encoder-decoder and marginally lower (-0.2 accuracy) than FILTER (Fang et al., 2020), an enhanced cross-lingual fusion model, which was the SOTA until 01/2021. We now turn our attention to the improvements over the XLM-R 0-shot model.

4.1 Text Classification

The intent classification accuracy of XLM-RA exceeds that of an XLM-R trained with labelled data, averaged across three task-oriented XNLU datasets (Tables 2, 5, 6). Starting from a competitive baseline model, XeroAlign improves intent accuracy by ∼5-10 points (similar for the base model, see Table 7 in Section 4.4). The benefits of cross-lingual alignment are particularly evident in low-resource languages (tr, hi, th), which is encouraging for applications with limited resources. Paraphrase detection is another instance of text classification. We report average XLM-RA accuracy in Table 3, which exceeds both Target and the Translate-Train scores by over 1 point and by almost 3 points over XLM-R 0-shot (even larger improvements for the base model).

Note that the amount of labelled data is comparable for XeroAlign and Target (except Thai, Spanish in MTOD and Turkish, Hindi in MultiATIS++) hence there is no more than a negligible advantage from the additional data. As the English scores show the highest performance (∼1.5 points higher on average) compared to the Target languages, we hypothesise that transferring this advantage from a high-resource language via XeroAlign is the primary reason behind its effectiveness compared to using target data directly. Given that Target performance has recently been exceeded with MoCo (He et al., 2020) and the similarities between contrastive learning and XeroAlign, our finding seems in line with recent work, which is subject to ongoing research (Zhao et al., 2020).

4.2 Structured Prediction

While XLM-RA is able to exceed Target accuracy for text classification tasks, even our best F-Scores

---

[6] Large=24 layers, 550M par, Base=12 layers, 270M par.

Model	EN	DE	ES	FR	JA	KO	ZH	Average
XLM-R Target	95.6	90.9	92.5	92.4	85.1	86.4	87.2	90.0
XLM-R Translate-Train	95.7	91.6	92.3	92.5	85.2	85.8	87.7	90.1
XLM-R 0-shot	95.6	91.0	91.1	91.9	81.7	81.6	85.4	88.3
Luo et al. (2020)	**96.4**	**93.0**	**93.0**	93.5	87.2	86.8	87.9	91.1
XLM-RA	95.8	92.9	**93.0**	**93.9**	87.1	87.1	88.9	91.2
Fang et al. (2020)	95.9	92.8	**93.0**	93.7	**87.4**	87.6	**89.6**	**91.4**
Section 4.5 experiment below: aligning with development/test set utterances but no task labels.								
XLM-RA (Exp)	95.8	94.2	94.4	94.8	91.6	92.6	92.1	93.6

Table 3: PAWS-X results as Paraphrase Classification Accuracy.

for slot filling are between 8 and 19 points behind the Target model. This is despite a strong average improvement of +4.1 on MTOP, +5.7 on MultiATIS++ and +5.2 on MTOD (a greater benefit for the base model). We think that the gap is primarily down to the difficulty of the sequence labelling task. That is, zero-shot text classification is ‘easier’ than zero-shot slot filling, which is manifested by a ∼10-20 point gap between scores. Sentences in various languages have markedly different input lengths and token order thus word-level inference in cross-lingual settings becomes significantly more challenging than sentence-level prediction because syntax plays a less critical role in sequence classification.

Another reason, related to XeroAlign’s architecture and training, is our choice to align the PXLM on the `[CLS]` embedding, which is subsequently used ‘as is’ for text classification tasks. Aligning individual token representations through the `[CLS]` embedding improves structured prediction as well, however, as the token embeddings are not directly used, the Multi-Head Attention parameters in the uppermost transformer layer never receive any gradient updates from XeroAlign. Closing this gap is a challenging opportunity, which we reserve for future work. Once again, the languages with lower NLP resources (th, hi, tr) tend to benefit the most from cross-lingual alignment.

4.3 XeroAlign Generalisation

In this subsection, we briefly investigate the generalisation of XeroAlign to unseen languages, taking the PAWS-X task as our use case. We are interested in finding out whether aligning on *just one language* has any zero-shot benefits for other languages. Table 4 shows the XLM-RA results when aligned on a single language (rows) and tested on other languages (columns).

-	EN	DE	ES	FR	JA	KO	ZH	AVE
DE	**96.0**	**92.9**	92.3	92.6	84.0	84.5	86.5	89.8
ES	95.9	92.6	**93.0**	93.1	83.9	84.1	86.4	89.9
FR	95.9	92.5	92.9	**93.9**	83.9	84.1	86.9	90.0
JA	**96.0**	92.6	91.8	93.1	**87.1**	**87.4**	87.9	**90.8**
KO	95.7	92.6	92.0	92.7	80.6	87.1	87.3	90.5
ZH	95.5	92.0	92.6	92.7	86.3	86.2	**88.9**	90.6
EU	96.2	92.5	93.0	94.1	84.9	85.2	87.1	90.4
AS	96.0	93.0	92.1	92.7	85.9	87.6	88.4	90.8

Table 4: XLM-RA aligned on one PAWS-X language (rows), evaluated on others (columns). AVE = average. EU = European languages, AS = Asian languages.

We can see that aligning on Asian languages (Japanese in particular) attains the best average improvement compared to aligning with European languages. This seems to reflect the known performance bias of XLM-R towards (high-resource) European languages, all of which show a strong improvement, regardless of language. Aligning only on European languages (de, es, fr) improves the average to 90.4 but aligning on Asian languages (zh, ko, ja) does not improve over Japanese (90.8). In any case, it is notable that the XLM-R model XeroAligned on *just a single language* is able to carry this advantage well beyond a single language thus improve average accuracy by 1.5-2.5 points over baseline (88.3) from Table 3. This effect is even stronger for MTOP (+4 accuracy, +3 F-Score).

4.4 Smaller Language Models

The XeroAligned XLM-R base model shows an even greater relative improvement than its larger counterpart with 24 layers and 550M parameters. To this end, we report the results for the XLM-

Model	**DE**	**ES**	**FR**	**TR**	**HI**	**ZH**	**PT**	**JA**	**AVE**
XLM-R Target	97.0/95.3	97.3/87.9	97.8/93.8	80.6/74.0	89.7/84.1	95.5/95.9	97.2/94.1	95.5/92.6	93.8/89.7
XLM-R 0-shot	96.4/84.8	97.0/85.5	95.3/81.8	76.2/41.2	91.9/68.2	94.3/82.5	90.9/81.9	89.8/77.6	91.5/75.5
XLM-RA	**97.6**/84.9	**97.8/85.9**	95.4/**81.4**	93.4/**70.6**	**94.0/79.7**	**96.4**/83.3	**97.6**/79.9	**96.1/83.5**	**96.0/81.2**
Jain et al. (2019)	96.0/87.5	97.0/84.0	97.0/79.8	**93.7**/44.8	92.4/77.2	95.2/**85.1**	96.5/**81.7**	88.5/82.6	94.5/77.8
Xu et al. (2020)	96.7/**89.0**	97.2/76.4	97.5/79.6	**93.7**/61.7	92.8/78.6	96.0/83.3	96.8/76.3	88.3/79.1	94.9/78.0

Table 5: MultiATIS++ Intent Classification Accuracy / Slot Filling F1. English model: 97.9/97. AVE = average.

Model	**Spanish**	**Thai**	**AVE**
§ Target (B)	98.7/89.1	96.8/93.1	97.8/91.1
§ Target (L)	98.8/89.8	97.8/94.4	98.3/92.1
§ 0-shot (B)	90.7/70.1	71.9/53.1	81.3/61.6
§ 0-shot (L)	97.1/85.7	82.8/47.7	90.0/66.7
XLM-RA (B)	98.9/86.9	97.9/**60.2**	98.4/**73.6**
XLM-RA (L)	**99.2/88.4**	**98.4**/57.3	**98.8**/72.9
Schuster et al.	85.4/72.9	95.9/55.4	90.7/64.2
Li et al.	98.0/83.0	96.9/52.8	97.5/67.9

Table 6: MTOD results as Intent Classification Accuracy / Slot Filling F-Score. Our best English score: 99.3/96.6, (B) = Base, (L) = Large, § = XLM-R model.

RA base model (12 layers, 270M parameters) in Table 7 as the average scores over all languages for MTOP, PAWS-X, MTOD and MultiATIS++. We use a relative % improvement over XLM-R 0-shot to compare the models fairly. We observe that the paraphrase detection accuracy improves by 3.3% for the large PXLM versus 6.5% for the base model. Across three XNLU datasets, the large XeroAligned XLM-R improves over the XLM-R 0-shot by 9.5% versus 14.2% for the base model on slot filling and by 7.1% (large) versus 19.8% (base) on text/intent classification.

Model	**MTOP**	**PAWS-X**	**M-ATIS**	**MTOD**
§ Target	94.0/88.1	85.2	89.0/86.3	97.6/92.2
§ 0-shot	80.8/68.9	81.7	76.9/65.0	80.1/64.8
XLM-RA	93.3/78.9	87.0	93.0/73.4	98.5/74.7

Table 7: The XLM-R(A) base model averages as intent classification accuracy / slot filling F-Score (or paraphrase accuracy for PAWS-X). § = XLM-R model.

This lets applications with lower computational budgets achieve competitive performance with our efficient cross-lingual alignment method for transformed-based PXLMs. In fact, the XLM-RA base model is able to reach up to 90-95% of the performance of its larger sibling using significantly lower computational resources.

4.5 Discussion

XLM-RA's intent classification accuracy is on average within ∼1.5 points of English accuracy across three XNLU datasets. However, the PAWS-X paraphrase detection accuracy is almost 5 points below English models, which is the case for other state-of-the-art PXLMs in Table 3. Why does the XLM-R struggle to generalise more on this task for languages other than English? A combination of several factors is likely at play. Minor translation errors may account for some of this deficit since the training is done using the publicly available machine-translated data. XeroAlign also leverages this data to align the multilingual representations of the PXLM. It is the nature of the dataset makes translation errors more likely for PAWS-X than for the task-oriented XNLU as the differences between paraphrased sentences are very subtle and challenging to discern (and to translate accurately). With that said, we think that the greater than expected deficit may be primarily caused by 1) a domain/topic shift within the dataset and 2) a possible data leakage for English. The original PAWS data (Zhang et al., 2019) was sourced from Quora Question Pairs and Wikipedia with neither being limited to any particular domain. As the English Wikipedia provides a large portion (relative to other languages) of the training data for XLM-R, it is possible that some of the English PAWS sentences may have been seen in training, which could explain the smaller generalisation gap for English.

In order to find out whether the accuracy gap will diminish, we artificially remove the hypothetical domain shift by using the parallel utterances (but not task labels) from the development and test sets. We thus XeroAlign the XLM-R on an extended vocabulary that may not be present in the train set. We observe that the (Exp) model in Table 3 shows an average improvement of over 2 points compared to the best XLM-RA and other SOTA models. This suggests that the increased generalisation gap may be caused by a domain shift for non-English languages on this task. When that topic shift gets (perhaps artificially) removed, the model is able to bring accuracy back within ∼2 points of the English model (in line with XNLU tasks). Note that this gap can be masked for English due to the language biases in data used for pretraining.

In section 2, we outlined the most conceptually similar methods that conducted large-scale model pretraining with task-agnostic sentence alignment as part of the training routine (Hu et al., 2020a; Feng et al., 2020; Pan et al., 2020; Chi et al., 2020). Where ablation studies were provided, the average improvement attributed to contrastive alignment was ∼0.2-0.3 points (though the tasks were slightly different). While we do not directly compare XeroAlign to contrastive alignment, it seems that a task-specific alignment may be a more effective and efficient technique to improve zero-shot transfer, given the magnitude of our results. This leads us to conclude that the effectiveness of our method comes primarily from cross-lingual alignment of the task-specific vocabulary. Language is inherently ambiguous, the semantics of words and phrases shift somewhat from topic to topic, therefore, a cross-lingual alignment of sentence embeddings *within the context of the target task* should lead to better results. Our simplified, lightweight method only uses translated task utterances, a single encoder model and positive samples, the alignment of which is challenging enough without arbitrary negative samples. In fact, this is the main barrier for applying contrastive alignment in task-specific NLP scenarios, i.e. the lack of carefully constructed negative samples. For smaller datasets, random negative samples would mean that the task is either too easy to solve, resulting in no meaningful learning or the model would receive conflicting signals by training on false positive examples, leading to degenerate learning.

4.6 Future Work

Our recommendations for avenues of promising follow-up research involve any of the following: i) aligning additional tasks such as Q&A, Natural Language Inference, Sentence Retrieval, etc. ii) including additional languages, especially low-resource ones (Joshi et al., 2020) and iii) evaluating large-scale, task-agnostic alignment of PXLMs followed by task-specific alignment, which is reminiscent of the common transfer learning paradigm of pretraining with Masked Language Modelling before fine-tuning on the target task. To that end, there is already some emergent work on monolingual fine-tuning with an additional contrastive loss (Gunel et al., 2020). For the purposes of multilingual benchmarks (Hu et al., 2020b; Liang et al., 2020) or other pure empirical pursuits, an architecture or a language-specific hyperparameter search should optimise XLM-RA for significantly higher performance as the large transformer does not always outperform its smaller counterpart and because our hyperparameters remained fixed for all languages. Most importantly, the follow-up work needs to improve zero-shot transfer for cross-lingual *structured prediction* such as Named Entity Recognition (Pan et al., 2017), POS Tagging (Nivre et al., 2016) or Slot Filling (Schuster et al., 2019), which is still lagging behind Target scores.

5 Conclusions

We have introduced **XeroAlign**, a conceptually simple and efficient method for task-specific alignment of multilingual sentence embeddings generated by PXLMs, aimed at effective zero-shot cross-lingual transfer. XeroAlign is an auxiliary training objective that is easily integrated into the unaltered primary task/model. XeroAlign leverages translated training data to bring the sentence embeddings in different languages closer together. We evaluated the XeroAligned XLM-R models (named XLM-RA) on zero-shot cross-lingual text classification, adversarial paraphrase detection and slot filling tasks, achieving SOTA (or near-SOTA) scores across four datasets covering 11 unique languages. Our ultimate vision is a level of zero-shot performance at or near that of Target models. The XeroAligned XLM-R partially achieved that goal by exceeding the intent classification and paraphrase detection accuracies of XLM-R trained with labelled data. We hope our work will inspire further progress in cross-lingual transfer learning.

Acknowledgements

We would like to thank the reviewers for their thoughtful comments and constructive feedback. We also want to thank the MindSpore[7][8] team for the technical support.

---

[7] `https://github.com/mindspore-ai`

[8] `https://mindspore.cn/`

References

Suzanna Becker and Geoffrey E Hinton. 1992. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, and Jingjing Liu. 2020. Filter: An enhanced fusion method for cross-lingual language understanding. *arXiv preprint arXiv:2009.05166*.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68.

Junjie Hu, Melvin Johnson, Orhan Firat, Aditya Siddhant, and Graham Neubig. 2020a. Explicit alignment objectives for multilingual bidirectional encoders. *arXiv preprint arXiv:2010.07972*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020b. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Alankar Jain, Bhargavi Paranjape, and Zachary C. Lipton. 2019. Entity projection via machine translation for cross-lingual NER. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1083–1092, Hong Kong, China. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. Mlqa: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, et al. 2020. Xglue: A new benchmark datasetfor cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018.

Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8433–8440.

Fuli Luo, Wei Wang, Jiahao Liu, Yijia Liu, Bin Bi, Songfang Huang, Fei Huang, and Luo Si. 2020. Veco: Variable encoder-decoder pre-training for cross-lingual understanding and generation. *arXiv preprint arXiv:2010.16046*.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Lin Pan, Chung-Wei Hang, Haode Qi, Abhishek Shah, Mo Yu, and Saloni Potdar. 2020. Multilingual bert post-pretraining alignment. *arXiv preprint arXiv:2010.12547*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Patti Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Kunxun Qi and Jianfeng Du. 2020. Translation-based matching adversarial network for cross-lingual natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8632–8639.

Libo Qin, Minheng Ni, Yue Zhang, and Wanxiang Che. 2020. Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. *arXiv preprint arXiv:2006.06402*.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3795–3805.

Leslie N Smith and Nicholay Topin. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612. International Society for Optics and Photonics.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual nlu. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5052–5063.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. Paws-x: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3678–3683.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308.

Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. 2020. What makes instance discrimination good for transfer learning? *arXiv preprint arXiv:2006.06606*.

Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. 2018. Overview of the third bucc shared task: Spotting parallel sentences in comparable corpora. In *Proceedings of 11th Workshop on Building and Using Comparable Corpora*, pages 39–42.

Using Word Embeddings to Analyze Teacher Evaluations: An Application to a Filipino Education Non-Profit Organization

Francesca Vera
Stanford University
Stanford, CA 94305
`fvera@alumni.stanford.edu`

Abstract

Analysis of teacher evaluations is crucial to the development of robust educational programs, particularly through the validation of desirable qualities being reflected on in the text. This research applies Natural Language Processing techniques on a real-world dataset from a Filipino education non-profit to explore insights from analyzing evaluations written by Teacher Fellows who assess their own progress. Prior to this research, only qualitative assessment had been conducted on the text. Inspired by the use of word embedding similarities to capture semantic alignment, we utilize GloVe embeddings to determine to what extent these evaluations reflect concepts critical to measuring the competency of Teacher Fellows and upholding the organization's Vision and Mission. As Fellows' quantitative ratings improved, so too did their demonstration of competency in the text. Further, Teacher Fellow language was consistent with the organization's Vision and Mission. This research therefore showcases the possibilities of NLP in education, improving our understanding of Teacher Fellow evaluations, which can lead to advances in program operations and education efforts.

1 Introduction

Applying Natural Language Processing (NLP) techniques to improve the quality of education programs is a crucial step in ensuring the NLP community's contributions to Social Good. Utilizing NLP unlocks the potential of computationally examining texts that were once only qualitatively analyzed or overlooked because of the difficulty in assessing the text. Textual data is plentiful in an educational setting, ranging from comments about student experiences to teacher evaluations and reflections. We look at applying NLP processes to teacher evaluations, which are a basis for documenting teacher growth and performance – components that will impact the quality of the education students receive.

The main contribution of this paper is that it introduces a computational framework through which teacher evaluations can be analyzed, so that insights gained can enhance educational programs. Applied on a real-world dataset from a Filipino education non-profit organization, word embedding similarities reveal which desirable traits within teachers are contained in these Teacher Fellow (a term used by the organization to describe its teachers in training) evaluations and their alignment with the organization's Vision and Mission. The motivation behind this work was to determine what qualities Teacher Fellows embody at different stages, as well as determining if their self-reflection was calibrated with manager (those training Teacher Fellows) evaluations. The organization was also curious if certain competencies emerged more than others.

With this application, we hope to improve the organization's teacher development efforts. Program changes that this piece of work inspired included refinement of the organization's evaluation tools and prompts, reflection on their competency definitions, greater discussions between Teacher Fellows and managers, alignment of interventions provided to Teacher Fellows, and increased support in the overall journey. We envision that this research can be scaled and flexibly applied to other bodies of textual data to serve future educational and Social Good purposes.

2 Related Work

2.1 NLP in Education

The uses of NLP in the education space include: to better understand natural language learning; to improve teaching materials; to develop learning applications; and to enhance student output, as out-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 382–389
August 1–6, 2021.

lined by Dr. Alhawiti (2014) in a survey of the state of NLP applications in education. Relevant to teacher evaluations, research by Rajput et al. (2016) implements a lexicon-based sentiment analysis tool on these evaluations to gain more insight on student feedback. Not only did the tool prove highly correlated with quantitative ratings, but also it provided clearer understanding of teacher performance through its sentiment score. Further, Tzacheva et al. (2019) present a model that detects multiple emotions within student comments on teacher performance in order to increase positivity in student emotions and improve student experience in the classroom. Our research differs from past NLP explorations in that it is mostly concerned with teacher self-evaluation and employs the use of word embeddings to derive a set of pre-defined concepts reflected in the text.

2.2 Word Embedding Similarities

Word embeddings are vector representations of words that capture their semantic features. These vectors can be assessed in relationship to one another through Euclidean distance or cosine similarity, used interchangeably to measure the semantic similarity (Pennington et al., 2014) between the words whose corresponding vectors are being compared. Embedding similarity to affirm the semantic alignment in language is a common technique in NLP. In exploring the semantic similarity between two texts, Kenter and de Rijke (2015) propose the proximity between embeddings as related to semantic proximity. To quantify gender and ethnic stereotypes using embeddings, Garg et al. (2017) computed the Euclidean distance between group words and neutral words to measure the strength of association between the two sets. Similarity can also be calculated on a sentence level, even when the structure of the sentence is ignored, but the associated embeddings of the words within the sentence are considered by averaging. (Faruqui et al., 2014; Yu et al., 2014) We adopt a combination of these past uses of word embeddings by calculating the similarity of sets of words to determine semantic proximity between them, where one of the sets contains sentences for which an associated embedding must be assigned.

3 Dataset

This paper applies NLP techniques on a real-world dataset from a Filipino education non-profit organization since the data consisted of many textual entries. The organization continues to conduct qualitative assessment on this particular dataset, but this paper explores the first application of quantitative analysis on it.

The dataset was made up of Summer Institute (SI) reflections, mid-year Competency Based Evaluations (CBE's), and end-year Competency Based Evaluations. CBE's allow a Teacher Fellow the opportunity to reflect on their progress, as well as provide a formal space for an instructional coach's assessment. Split into four domains: Personal Leadership, Servant Leadership, Change Management, and Critical Learning, CBE's prompt the Fellow to write on "Critical Incidents, Strengths, and Areas of Growth" and give themselves a rating out of 4.0 per domain. A coach repeats the process based on their assessment of the Fellow. Overall, each CBE consists of 8 text entries and 8 ratings. SI reflections differ from CBE's in that the prompts are about high-level, philosophical ideas in education and there are no quantitative ratings.

For the sake of consistency, only Fellows who had SI reflections, mid-, and end-year CBE's were chosen. There were 14 such Fellows. These evaluations were collected in 2019, making it still relevant to the present-day programs of the organization. In total, there were 126 unique text entries: 14 SI reflections, 56 CBE entries by Teacher Fellows, and 56 CBE entries by instructional coaches. Alongside these text entries were 56 ratings by Teacher Fellows and 56 ratings by instructional coaches.

Although the dataset only consisted of 14 Fellows, there were 126 text entries of average length 204 words, which was significant enough to move forward with exploring this application.

4 Method

4.1 Keyword List Development

The organization developed keyword lists to encapsulate the ideas that it was interested in analyzing. A Data and Impact Assessment Manager created these lists based on three important aspects: Competencies, Core Values, and Mission. These Competencies are the listed "indicators" that Fellows should develop throughout the duration of the program and after completion. They were defined by the organization's Fellowship Program team, who have years of experience and pivotal knowledge of the Fellows' impact on students and communities. The "indicators" were condensed into lists of Competency keywords by careful selection of words that carried the most meaning. A similar process was done to create Core Values and Mission lists, which represent the organization's Vision and Mission. Core Values were also embedded in the "indicators," so keywords were extracted from there. The Mission keyword list came directly from the organization's Mission statement. Because the Data and Impact Assessment Manager was not on the Program team that wrote the "indicators," keyword selection was relatively objective, ensuring that certain concepts were not favored nor dismissed.

Although there may be subjectivity within keyword choices, common NLP tasks exist where keywords are grouped by theme into sets (e.g. determining which occupation words are neutral in gender bias embedding explorations (Garg et al., 2017)). The method of having an internal expert choose keywords that represented competencies based on pre-defined qualities that Teacher Fellows should achieve most closely mirrors methods for past NLP datasets where experts from other fields have hand-labeled entries (e.g. cross-validating entries in hate speech datasets (Jha and Mamidi, 2017)) – despite the subjectivity in this exercise. Most importantly, the organization fealt confident that the keywords chosen accurately reflected its desired competencies, core values, and mission.

4.2 Embedding Similarity Calculation

Inspired by the concept that cosine similarity between embeddings measures semantic similarity between words, we calculate the similarity between the keyword lists and teacher evaluations.

For each keyword, we find the keyword embedding. If the keyword was present in the predefined embeddings, we used that embedding. In some cases, however, there were multiple words that made up a keyphrase; for these items, we took the average of embeddings for every word in the keyphrase and assigned that as the embedding. Next, we find the embedding for the teacher evaluations, which we will refer to as the evaluation embedding. Each evaluation was treated as a single document of text with only one assigned embedding. In the pre-processing of this text, punctuation was stripped, spelling was checked through Microsoft Word, and documents were tokenized. Similar to what was done with keyphrases, we averaged the embeddings of every token, and this average was taken as the evaluation embedding.

Competency Domain	Mid-Year Similarity	End-Year Similarity
Personal Leadership	0.498	0.523
Servant Leadership	0.603	0.608
Change Management	0.531	0.532
Critical Learning	0.599	0.600

Table 1: Fellow Competency Similarity Scores

Competency Domain	Mid-Year Similarity	End-Year Similarity
Personal Leadership	0.519	0.523
Servant Leadership	0.632	0.634
Change Management	0.557	0.560
Critical Learning	0.631	0.629

Table 2: Coach Competency Similarity Scores

After finding the keyword embedding and the evaluation embedding, we calculate their cosine similarity, which becomes their similarity score. This process was repeated for every evaluation and keyword. To find the overall score for Competencies, Core Values, or Mission, we average across all Fellow-keyword similarity scores per list.

4.3 Experiments

The following experiments were conducted on the dataset with GloVe 50-dimensional embeddings. A similarity score was calculated for each Fellow across the organization's Competency domains: Personal Leadership, Servant Leadership, Change Management, and Critical Learning, which had unique keyword lists each. This process was repeated for Fellows' and coaches' mid-year and end-year CBE entries, resulting in 16 similarity scores (tables 1 and 2). We also include the Fellows' and coaches' quantitative ratings out of 4.0 from the CBE's (tables 3 and 4) for later comparison. The Competency keywords with the highest similarities are listed out (tables 5 and 6). We calculated the similarity between Fellow evaluations and the Core Values and Mission lists to measure alignment with Vision and Mission. This was done on the Fellows' SI reflections, mid-year CBE's, and end-year CBE's (table 7). We also list the top keywords for Core Values and Mission (table 8).

Competency Domain	Mid-Year Score	End-Year Score
Personal Leadership	2.619	2.833
Servant Leadership	2.381	2.524
Change Management	2.214	2.393
Critical Learning	2.250	2.536

Table 3: Fellow CBE Ratings

Competency Domain	Mid-Year Score	End-Year Score
Personal Leadership	2.500	2.786
Servant Leadership	2.191	2.548
Change Management	2.143	2.446
Critical Learning	2.321	2.750

Table 4: Coach CBE Ratings

5 Discussion of Results

Cosine similarity has maximum score 1.0, which occurs when perfect similarity is achieved; higher cosine similarity indicates higher semantic similarity. Thresholds differ according to vector space and can be calculated through a variety of methods. We employ a cutoff of > 0.37 (Orkphol and Yang, 2019) to indicate similarity based on a predictive model validated by human perception of relevance, as human intuition of similarity is fundamental to how our application would be used in practice. However, we acknowledge that to achieve a truly appropriate cutoff, we would have to replicate the empirical study with our own relevant prompts to generalize for this task.

Fellow and coach mid-year and end-year CBE's display Competency similarity scores above our chosen threshold, indicating that their essence was present in the evaluations. We also notice an improvement in all Competency domains from mid-year to end-year evaluations for Fellows and coaches. This upward trend mirrors the increase in quantitative ratings, suggesting that as Fellows improve in practice, their reflections become more aligned with the domain concepts. The domain with highest similarity was Servant Leadership, while the lowest was Personal Leadership, although the differences between similarities are minimal. It is important distinguish what is reflected on in the evaluations from how the Fellows are rated in practice: for example, Personal Leadership ratings were among the highest out of 4.0, but the similarity scores were the lowest, meaning there may be inconsistencies between what is written and how Fellows are rated. Coaches produced higher similarity scores than Fellows did, which is expected due to coaches who speak more explicitly in terms of the Fellow's competency.

Clear themes emerge from looking at the top keywords for Competency domains. Personal Leadership focuses on life skills that Fellows may improve upon. For Servant Leadership, there is emphasis on the relationship aspects of teaching. Because Change Management is concerned with executing plans for the larger community, the top keywords express big picture thinking. In Critical Learning, the topics cover knowledge and lesson formation.

Personal Leadership	Servant Leadership
Manages Time	Relationship Building
Personal Development	Strong Relationships
Work Habits	Common Goals
Improve	Shared Goals
Describes Drives	Positive Relationships

Table 5: Top 5 Competency Keywords pt. 1

Change Management	Critical Learning
Higher Order Thinking	Working Knowledge
Effective Plans	Lesson Plan
Systems Thinking	Subject Matter
Big Picture	Deliver Lessons
Considering	Subject Content

Table 6: Top 5 Competency Keywords pt. 2

Since the Core Values and Mission keyword lists captured the Vision and Mission of the organization, the significant similarity scores across SI reflections, mid-year, and end-year CBE's indicate that the Vision and Mission was expressed in the Fellows' writing. The top 3 keywords for Core Values are especially salient, as "get the job done" was consistent with the Critical Learning Competency domain, and "working with others" and "working as a team" directly related to Servant Leadership.

5.1 Debrief with the Organization

We set up a formal discussion with the organization's Fellowship Program team, whose expertise in the program, evaluations, and Fellow performance led us to further insights. They endorsed the outcome that Servant Leadership produced the highest similarity scores. They theorized that the interpersonal and community aspects of Servant Leadership were already strong within this cohort of Fellows. When looking at the top 5 keywords for Servant Leadership, the themes of fostering relationships and shared goals were unsurprising to them. The second highest scoring domain, Criti-

Keyword List	SI	Mid-Year CBE	End-Year CBE
Core Values	0.686	0.677	0.679
Mission	0.637	0.620	0.623
Competency	0.565	0.577	0.566

Table 7: Core Values and Mission Similarity Scores

Core Values	Mission
Get the Job Done	Programs
Working with Others	Life Skills
Working as a Team	Developing Students

Table 8: Top 3 Core Values and Mission Keywords

cal Learning, also lined up with their expectations; because this domain is specifically concerned with pedagogical knowledge, Critical Learning tends to be discussed at length, as would be typical in standard teacher evaluations. Further, they said it made sense that coaches produced higher similarity scores than Fellows did since coaches are not only more familiar with concepts, but are also given templates for being explicit in their writing.

The Program team appreciated the conceptual alignment between the evaluations and the Core Values and Mission, claiming that throughout the training process, coaches encourage Fellows to incorporate these components in their practice. After looking at the top 3 Mission keywords, the Program team emphasized that Fellows are coached on “life skills” and “developing students” – two of the most important components of the program.

They noted that one area of improvement that would be easily implemented in future applications of this process would be to update keyword lists as the “indicators” themselves get updated. Because this project inspired reflection within the organization regarding what qualities it is evaluating teachers on, any further analysis should consider appropriate changes in competencies.

It may also be important to note that the Fellowship Program team’s enthusiasm for this project meant that we received dynamic feedback during the entire process.

6 Conclusion

We present an application of word embedding similarities to evaluate how Teacher Fellow evaluations align with requirements for Competency and a non-profit’s Vision and Mission. This analysis adds a new element to the interpretation of textual data that would have otherwise been only qualitatively examined. Moving forward, the organization aims to apply this method to other batches of Fellows. Other members of the organization, such as Admissions and Alumni Program teams, also conduct evaluations on which the method could be applied. A comparison may be conducted to see what certain cohorts were more willing to reflect on in the text and how their focuses differ. More sophisticated models, including Tf-idf weighting and a higher level of pre-processing, are possible, and further validation through correlation with quantitative ratings can help clarify the value of this model. Having demonstrated that Fellows’ evaluations captured ideas contained within Competency, Core Values, and Mission keywords, we are confident that the organization can incorporate this analysis to achieve its goal of providing children access to high quality education across the Philippines.

Acknowledgments

I would like to acknowledge the Filipino education non-profit organization that provided the project scope and dataset that shaped this paper. In particular, Gaby, Ed, and Jo – who offered valuable collaboration, insights, and guidance.

Ethics and Positive Impact Considerations

NLP for Positive Impact aims to promote innovative ways NLP research will positively impact society, and this paper presents a case study of an NLP application for the social good cause of education. We consider the impact of this paper as positive because the improvements to the non-profit's programs (due to this research) will advance the organization's mission to provide Filipino children with the highest standard of education possible. This paper may also inspire positive impact in other situations where similar applications of NLP could be used for social good.

There is also an opportunity to incorporate the UN theme for International Women's Day, "Women in Leadership," because of the relationship between education and empowering women. It is worth noting that in the Philippines, teachers are mostly women, so this research could inform their professional leadership journey, including how to better train and support them. We envision many possibilities for this application in a profession dominated by women in the Philippines.

For context, the Filipino education non-profit focuses on improving teacher quality and addressing system-level educational challenges. This organization that provided the data allowed the submission of this paper. We focus on its Fellowship Program that trains Teacher Fellows to significantly improve student learning outcomes. Because the Fellowship Program is concerned with the growth of Teacher Fellows in their ability to deliver lessons of high quality and create positive change in educational communities, analyzing their evaluations is a worthwhile task in continuing the efficacy of this program. The Fellowship Program reaches over 10,000 public school students in the Philippines annually. Since the organization would be the main beneficiary of this technology, this technology has the potential to improve the Fellowship Program, which in turn will impact the Teacher Fellows and their students. A secondary beneficiary would be other similar non-profit organizations or operations teams that run educational programs.

The dataset used in this application was directly provided to us by the organization's Data and Impact Assessment team. The handover of data was formally documented and a non-disclosure agreement was signed. Further, we met with the Instructional Coaching Team to confirm the consent of the original authors of the texts. With regards to dataset privacy considerations, per the request of the organization and to protect the privacy of the Teacher Fellows and instructional coaches, the evaluations were completely anonymized. For the same reasons, the data cannot be submitted for review or replication. We chose not to include direct quotations from the evaluations (which could have been used to strengthen discussion of results) to ensure the authors' privacy.

An overall evaluation of using word embeddings for similarity tasks was considered. (Faruqui et al., 2016) Further, word embeddings have been shown to carry biases (e.g. gender and ethnic stereotypes) (Garg et al., 2017), so these biases may manifest in the outcomes. Because the similarity threshold was defined by a model based on human perception, the biases of the human participants may also factor into the resulting value. We acknowledge that the research was conducted in English, which may exclude other languages where word embeddings in those languages cannot be applied in a similar way.

References

Khaled Alhawiti. 2014. Natural language processing and its use in education. *International Journal of Advanced Computer Science and Applications*, 5:72–76.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 30–35, Berlin, Germany. Association for Computational Linguistics.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2017. Word embeddings quantify 100 years of gender and ethnic stereotypes. *CoRR*, abs/1711.08412.

Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In *Proceedings of the Second Workshop on NLP and Computational Social Science*, pages 7–16, Vancouver, Canada. Association for Computational Linguistics.

Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 1411–1420, New York, NY, USA. Association for Computing Machinery.

Korawit Orkphol and Wu Yang. 2019. Word sense disambiguation using cosine similarity collaborates with word2vec and wordnet. *Future Internet*, 11(5).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Quratulain Rajput, Sajjad Haider, and Sayeed Ghani. 2016. Lexicon-based sentiment analysis of teachers' evaluation. *Appl. Comp. Intell. Soft Comput.*, 2016:1.

Angelina Tzacheva, Jaishree Ranganathan, and Rajendra Jadi. 2019. Multi-label emotion mining from student comments. In *Proceedings of the 2019 4th International Conference on Information and Education Innovations*, ICIEI 2019, page 120–124, New York, NY, USA. Association for Computing Machinery.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

A Appendix

A.1 Keyword Lists

A.1.1 Personal Leadership

Personal Leadership, Self-Management, Self-Awareness, Self-Regulation, Motivation, Responsibility, Personal Development, Excellence, Describes Emotions, Describes Drives, Describes Values, Personality, Manages Resources, Manages Energy, Manages Emotions, Manages Time, Deliverables, Work Habits, Quality Output, Personal Motivations, Goals, Persevering, Rallying, Challenges, Setbacks, Constructive Feedback, Improve

A.1.2 Servant Leadership

Servant Leadership, Relationship Building, Strong Relationships, Healthy Relationships, Rewarding Relationships, Common Goals, Positive Relationships, Respect, Humility, Empathy, Expresses Clearly, Influences, Shared Goal, Expresses Effectively

A.1.3 Change Management

Change Management, Planning, Strategic, Effective, Ambitious, Realistic, Short-Term Goals, Mid-Term Goals, Long-Term Goals, Community, Effective Plans, Resources, Vision, Education Reform, Collaboration, Stakeholders, Scale, Innovation, Creative, Valuable, Sustainable, Enhancement, Community, Capacity Building, Collaborating, Respecting, Considering, Existing Practices, Existing Procedures, Resources, Efforts, Higher Order Thinking, Creative, Divergent, Convergent, Critical, Analytical, Excellence, Systems Thinking, Big Picture, Social, Political, Cultural, Evidence, In-Depth, Effective Decisions, Time-Critical, Communities, Factors, Proposes, Contextualized, Logical, Perspectives, Execution

A.1.4 Critical Learning

Critical Learning, Content Knowledge, Subject Matter, Contextualized, Applied, Learning, Teaching, Measurable Changes, Broadened Opportunities, Working Knowledge, Subject Content, Lesson Plan, Mastery, Delivery Lessons, Community Engagement Activities, Captivate, Care, Classroom Management, Confer, Teaching Framework, Pedagogical Knowledge

A.1.5 Core Values

Excellent Education, Inclusive Education, Relevant Education, Working as a Team, Excellent Results, Listens, Learns, Get the Job Done, Acts with Respect, Acts with Kindness, In the Face of Ambiguity, Builds Strong Partnerships, Values Strong Partnerships, Collaboration, Humility, Success of Whole, Integrity, Values Consensus, Shared Outcomes, Accepting, Sets Aside Personal Ego, Respect, Positive Relationships, Responsibility to Learn, Extend Helping Hand, Kindness, Inside and Outside Organization, Individual Role, Maximize Learning, Effectively Performs, Takes Action, Different Stakeholders, Collective Success, Working with Others, Takes on Tasks Outside Role, Analyzes Problem, Community Engagement, Relate Positively, Connect, Short-Term Impact, Mid-Term Impact, Long-Term Impact, Desired Outcomes, Seeks Perspectives, Sustainable Solution, Invites, Receptive to Feedback, Acts Decisively, Contribute, Translate Feedback, Commits to High Standards, Assistance, Improve Performance, Respectfully Asserts, Shared Goals, Makes Decisions

A.1.6 Mission

Positively Impact, Academics, Life Skills, Functional Literacy, Education Reform, Partners, Network, Shared Goal, Programs, Developing Students, Long-Term

Relation Classification with Entity Type Restriction

Shengfei Lyu, **Huanhuan Chen**[*]
School of Computer Science and Technology
University of Science and Technology of China, Hefei, China
`saintfe@mail.ustc.edu.cn, hchen@ustc.edu.cn`

Abstract

Relation classification aims to predict a relation between two entities in a sentence. The existing methods regard all relations as the candidate relations for the two entities. These methods neglect the restrictions on candidate relations by entity types, which leads to some inappropriate relations being candidate relations. In this paper, we propose a novel paradigm, RElation Classification with ENtity Type restriction (RECENT), which exploits entity types to restrict candidate relations. Specially, the mutual restrictions of relations and entity types are formalized and introduced into relation classification. Besides, the proposed paradigm, RECENT, is model-agnostic. Based on two representative models GCN and SpanBERT respectively, $\text{RECENT}_{\text{GCN}}$ and $\text{RECENT}_{\text{SpanBERT}}$ are trained in RECENT[1]. Experimental results on a standard dataset indicate that RECENT improves the performance of GCN and SpanBERT by 6.9 and 4.4 F1 points, respectively. Especially, $\text{RECENT}_{\text{SpanBERT}}$ achieves a new state-of-the-art on TACRED.

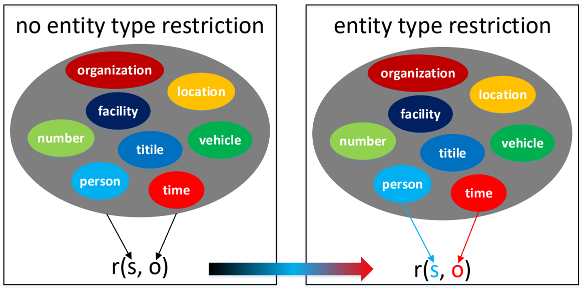

Figure 1: A relation restricts entities with appropriate types. In the figure, r is *who-is-born-when*. Different colored ellipses represent entities with different types.

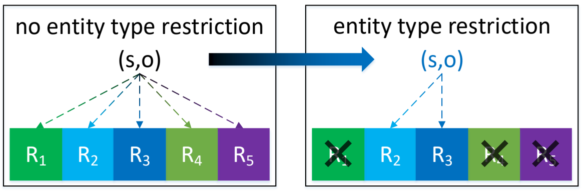

Figure 2: Entity type restriction for relation classification. According to entity type restriction, the number of candidate relations reduces from 5 (left) to 2 (right).

1 Introduction

Relation classification, a supervised version of relation extraction, aims to predict a relation between two entities in a sentence. Relation classification is an important step to construct knowledge bases from a large number of unstructured texts (Trisedya et al., 2019), which benefits many natural language processing applications, such as natural language generation (Kang and Hashimoto, 2020) and question answering (Zhao et al., 2020).

Recently, the majority of methods make use of various neural network architectures to learn a fixed-size representation for a sentence and its entities with various language features, such as part of speech (POS), entity types, and dependency trees. Dependency trees that are parsed from sentences are exploited by GCN (Kipf and Welling, 2017) to model sentences (Zhang et al., 2018; Guo et al., 2019). As a sequence of words, a sentence is modeled by LSTM (Hochreiter and Schmidhuber, 1997) and its entity positions are involved with the attention mechanism (Zhang et al., 2017). More recently, pretrained language models (Devlin et al., 2019; Baldini Soares et al., 2019; Joshi et al., 2020) achieve good performance in relation classification since they are pretrained on massive corpora.

To recap, these methods utilize an encoder architecture (Badrinarayanan et al., 2017) to obtain a representation for a sentence. In other words, they only focus on the modeling of sentences and

---

[*] Corresponding author.

[1] Our code is available at `https://github.com//Saintfe/RECENT`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 390–395
August 1–6, 2021.

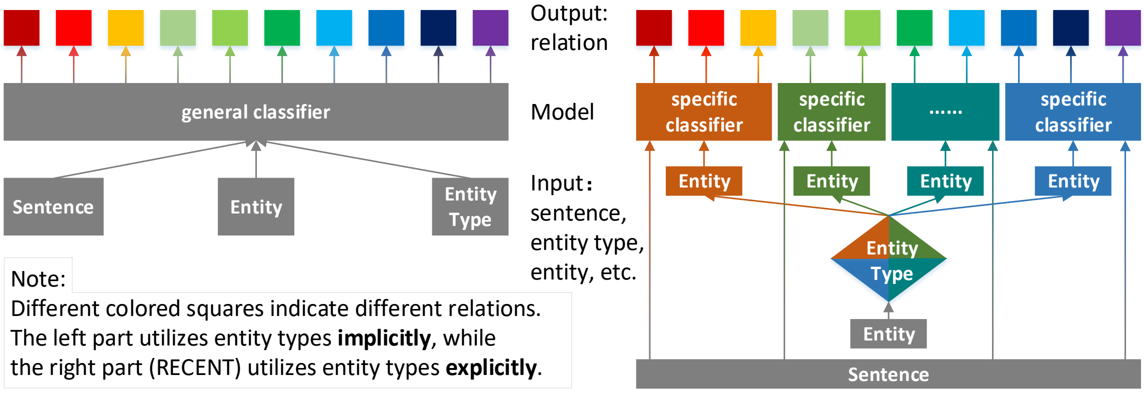

Figure 3: Relation classification with entity type restriction. The left part does not consider the restriction of entity types on relations and only feeds entity types as features into a general classifier. The right part explicitly utilizes entity types to restrict candidate relations and learns a specific classifier for each pair of entity types.

treat relations as labels[2] to be classified. However, in this process, these methods inevitably lose the semantics of relations. Take the mutual restrictions between a relation and entity types as an example. In Figure 1, the relation *who-is-born-when* restricts its first entity to be a person and the second one to be a time. Conversely, entity types can also restrict candidate relations in relation classification. As illustrated in Figure 2, some inappropriate relations can be discarded from candidate relations by entity type restriction. However, the current methods neglect the restriction of entity types on relations so that some inappropriate relations are regarded as candidate relations, which further hurts their performance.

To solve the above problem, a novel paradigm, RElation Classification with ENtity Type restriction (RECENT), is proposed to exploit entity types to restrict candidate relations. As the basis of the paradigm, the mutual restrictions of relations and entity types are formalized. With the entity type restriction, some inappropriate relations are discarded from the candidate relations of a specific pair of entity types, as illustrated in Figure 2. A specific classifier with a specific set of candidate relations is individually learned for each pair of entity types (Figure 3). Therefore, the proposed paradigm, RECENT, can eliminate the interference from inappropriate candidate relations.

The contributions are summarized as follows:

- The mutual restrictions of relations and entity types are formalized.
- A novel paradigm, RECENT, is proposed to exploit entity types to restrict candidate relations in relation classification.
- A new state-of-the-art is achieved on TACRED.

2 Proposed Paradigm

Before introducing the proposed paradigm RECENT, the mutual restrictions between a relation and a pair of entities are formalized as the basis of RECENT.

2.1 Relation Function

When a binary relation is considered as a function, this relation has two entities as its two arguments. Formally, this relation is formalized as $r(s, o)$, where r denotes the <u>r</u>elation and s, o denote the first (<u>s</u>ubject) entity and the second (<u>o</u>bject) entity, respectively. The range of this relation contains two discrete values $\{0, 1\}$:

$$r(s,o) = \begin{cases} 1 & r \text{ holds between } s \text{ and } o, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In a broad sense, the domain of this relation can be any pair of entities. However, when a pair of entities with inappropriate types is fed into a specific relation, the relation can directly return 0, no need to consider the compositional semantics of the relation and the pair of entities. For example, a specific relation *who-is-born-when* expects the first argument to be a person and the second one to be a time. Therefore, (*apple*, *Steven Jobs*) is a pair of inappropriate entities for this relation so that *who-is-born-when(apple, Steven Jobs)* returns 0 without considering the compositional semantics,

---

[2] Specifically, these meaningful relations are treated as meaningless numbers, such as 0, 1, 2.

since *apple* may refer to either a kind of fruit or a company (not a person) and *Steven Jobs* may refer to a famous person (not a time).

Only when a relation receives a pair of appropriate entities whose types match it, the combination of the relation and the entities might make sense (i.e., the function defined in Eq. 1 may return 1). In this case, it is meaningful to further verify the correctness of the compositional semantics. From this perspective, in a narrow sense, the domain (denoted by D_r) of a relation (r) is defined as follows:

$$D_r = \{(s, o) | ts \in S(r) \text{ and } to \in O(r)\}, \tag{2}$$

where ts and to denote the **t**ypes of the subject entity (s) and the object entity (o), respectively. $S(r)$ and $O(r)$ are the appropriate types of r on the subject entity (s) and the object entity (o), respectively.

2.2 Entity Type Restriction

In the previous subsection, the narrow domain of a relation restricts entities whose types need to match the relation. Conversely, given a pair of entities whose types are known, the candidate relations of the entities are also restricted, since the match between relations and entity types is mutual.

Formally, given a pair of entities (s, o) and their types (ts, to), its candidate relations (denoted by $R_{(ts,to)}$) are restricted into a limited set:

$$\begin{aligned} R_{(ts,to)} &= \{r \in R | (s, o) \in D_r\} \\ &= \{r \in R | ts \in S(r) \text{ and } to \in O(r)\}, \end{aligned} \tag{3}$$

where R denotes all possible relations. When the types (ts, to) of a pair of entities (s, o) are explicitly utilized to restrict its candidate relations, the candidate relations reduce from all possible relations R into a rather smaller set $R_{(ts,to)}$.

2.3 Relation Classification

Unlike traditional methods that classify a sentence and its entities on all candidate relations R (the left part of Figure 3), the proposed paradigm, RECENT learns a specific classifier with smaller and more precise candidate relations for each pair of entity types (the right part of Figure 3), based on entity type restriction in the previous subsection.

The procedure of RECENT is summarized in Algorithm 1. In the learning phase, all sentences are first grouped by types of their entities (line 1). For each group (marked as g) with a specific pair of entity types (ts,to), the candidate relations $R_{(ts,to)}$ for the group g are obtained by aggregating the relations in the group g (line 3). Then, a specific classifier (marked by f_g) that maps sentences and their entities in g to $R_{(ts,to)}$, is learned for the group g (line 4). In the prediction phase, given a new sample (se, s, o, ts, to), a group (marked as $g^{'}$) is matched by the entity types (ts, to) (line 6). Then, the classifier $f_{g^{'}}$ learned on the group $g^{'}$ is utilized to predict a relation according to the input (se, s, o) (line 7).

From the 4th line of Algorithm 1, the proposed paradigm RECENT is model-agnostic, which means that RECENT is theoretically compatible with many relation classification models.

Algorithm 1 RECENT

```
Learning Phase:
    Input: $\mathcal{D} = \{(se_i, s_i, o_i, ts_i, to_i, r_i) | i = 1, 2, ..., N\}$ where the subscript $i$ indicates the
    $i$th sample, $se$ is sentence, $s$ is subject entity, $o$ is object entity, $ts$ is type of subject entity, $to$
    is type of object entity, $r$ is relation.
    Output: Multiple classifiers.
 1: Group sentences by entity types.
 2: for each group $g$ (enity types ($ts$, $to$) ) do
 3:     aggregate relations in the group as candidate relations $R_{(ts,to)}$ defined in Eq. 3.
 4:     learn a classifier (marked as $f_g$) on the group that maps $\{(se_i, s_i, o_i) \in g\}$ to $R_{(ts,to)}$.
 5: end for
Prediction Phase:
    Input: A new sample $\{se, s, o, ts, to\}$, each specific classifier for each pair of entity types.
    Output: A relation.
 6: match the sample to a group (marked as $g^{'}$) according to the entity types ($ts$, $to$).
 7: Use the classifier ($f_{g^{'}}$) learned on the group to map ($se$, $s$, $o$) to a relation.
 8: return the relation.
```

3 Experiments

3.1 Dataset

The proposed paradigm RECENT is evaluated on TACRED[3] (Zhang et al., 2017). TACRED contains 41 semantic relations and a special *no_relation* over 106,264 sentences. The subject entities in TACRED are classified into two types: *PERSON* and

---

[3] https://catalog.ldc.upenn.edu/LDC2018T24

ORGANIZATION while the object entities are categorized into 16 fine-grained types, such as *LOCATION* and *TIME*. Namely, entity types are known. By convention, the micro-averaged F1 score (abbreviated as F1) is reported on TACRED.

3.2 Experimental Setup

Since *no_relation* is a candidate relation of each pair of entity types in TACRED, a binary classifier is first learned to distinguish between 41 semantic relations and *no_relation*. In this way, each pair of entity types reduces one candidate relation (i.e. *no_relation*) in RECENT. If the binary classifier predicts *no_relation* for a pair of entities, then the final relation for them is *no_relation*. Otherwise, their specific semantic relation is further predicted in RECENT.

Base Models The proposed paradigm RECENT is model-agnostic. Two representative models that are GCN (Zhang et al., 2018) and SpanBERT (Joshi et al., 2020) are selected as base models (line 4 in Algorithm 1). For a fair comparison with a base model, all classifiers (including the binary classifier) in RECENT are trained by the base model. The corresponding models in the paper are denoted as $\text{RECENT}_{\text{GCN}}$ and $\text{RECENT}_{\text{SpanBERT}}$.

Hyperparameters For $\text{RECENT}_{\text{GCN}}$, the path-centric pruning K is set to 1 as GCN (Zhang et al., 2018). The learning rates for all classifiers in $\text{RECENT}_{\text{GCN}}$ are set to 0.3. For $\text{RECENT}_{\text{SpanBERT}}$, the learning rates for all classifiers are chosen from {5e-6, 1e-5, 2e-5, 3e-5, 5e-5} as SpanBERT.

Compared Models Extensive models in relation classification are regarded as comparison models. They include PA-LSTM (Zhang et al., 2017), C-GCN (Zhang et al., 2018), AGGCN (Guo et al., 2019), C-AGGCN (Guo et al., 2019), MTB (Baldini Soares et al., 2019), KnowBert (Peters et al., 2019), SpanBERT-ALT (Lyu et al., 2020), KEPLER (Wang et al., 2020b), K-Adapter (Wang et al., 2020a), and LUKE (Yamada et al., 2020). To save space, please refer to the original papers of these models for details.

3.3 Experimental Results

The experimental results are presented in Table 1. $\text{RECENT}_{\text{GCN}}$ achieves a significant performance increase on the F1 score above its base model GCN. The absolute increase reaches 6.9 from 64.0 to 70.9. The main contribution for the F1 increase is the improved precision that greatly increases from 69.8 to 88.3. The great increase in precision, which might result from the restriction on candidate relations by entity types in RECENT, indicates the effectiveness of the proposed paradigm RECENT. Besides, $\text{RECENT}_{\text{GCN}}$ suppresses the compared models that do not include pretrained language models.

Model	P	R	F1
PA-LSTM † (Zhang et al., 2017)	65.7	64.5	65.1
C-GCN † (Zhang et al., 2018)	69.9	63.3	66.4
AGGCN † (Guo et al., 2019)	69.9	60.9	65.1
C-AGGCN † (Guo et al., 2019)	71.8	66.4	69.0
GCN † (Zhang et al., 2018)	69.8	59.0	64.0
$\text{RECENT}_{\text{GCN}}$ (ours)	88.3	59.3	70.9
SpanBERT-ALT † (Lyu et al., 2020)	69.0	73.0	70.9
MTB † (Baldini Soares et al., 2019)	-	-	71.5
KnowBert † (Peters et al., 2019)	71.6	71.4	71.5
KEPLER †* (Wang et al., 2020b)	71.5	72.5	72.0
K-Adapter †* (Wang et al., 2020a)	70.14	74.04	72.04
LUKE † (Yamada et al., 2020)	70.4	**75.1**	72.7
SpanBERT † (Joshi et al., 2020)	70.8	70.9	70.8
$\text{RECENT}_{\text{SpanBERT}}$ (ours)	**90.9**	64.2	**75.2**

Table 1: Results on the TACRED dataset. P and R indicate precision and recall, respectively. Bold marks the highest values among models. † marks results reported in the original papers. * marks results from preprint papers.

Similarly, $\text{RECENT}_{\text{SpanBERT}}$ overtakes its base model SpanBERT by absolute 4.4 points on F1. The great soar (absolute 20.1 points) on precision contributes the superior F1 of $\text{RECENT}_{\text{SpanBERT}}$. Unfortunately, the decline in recall limits the further improvement of F1. This might be due to sample imbalance of candidate relations, which will be further studied in future work. On the whole (i.e. F1), $\text{RECENT}_{\text{SpanBERT}}$ outperforms all the compared models. Especially, $\text{RECENT}_{\text{SpanBERT}}$ exceeds the state-of-the-art LUKE model[4] by 2.5 F1 points and achieves a new state-of-the-art.

3.4 Error Analysis of GCN

This subsection analyzes the influence of a baseline model (i.e. GCN) that neglects the restriction of entity types on relations. We retrain a GCN model and the model achieves 68.4 precision, 60.2 recall, and 64.1 F1 (Table 2), which are similar to the

---

[4] LUKE achieves the state-of-the-art (72.7) on the published papers. Cohen et al. (2020) report a new state-of-the-art (74.8) in the preprint way. Anyway, $\text{RECENT}_{\text{SpanBERT}}$ achieves a new state-of-the-art (75.2).

Model	P	R	F1	FP	FP(ET)
GCN	68.4	60.2	64.1	1,323	144

Table 2: Results of our trained GCN on the TACRED dataset. P and R indicate precision and recall, respectively. FP indicates the number of false positives and FP(ET) indicates the number of false positives that break the entity type restriction.

results in its reported paper (Zhang et al., 2018). Observing the prediction results of the model, we find that 1) 1,323 examples are false positives in the test set of TACRED, 2) 144 (about 11%) false positives among them break the entity type restriction. Namely, GCN can make about 89% of false positives meet the entity type restriction, by implicitly using entity types. However, about 11% of false positives still break the restriction. The false positives broken down by relations are counted in Appendix A. In details, false positives broken down by relations are weakly negatively correlated with the amount of training data of relations, where the correlation coefficient is -0.39. This infers that fewer training examples of relations may lead to more false positives of relations.

4 Conclusion

In the paper, a novel paradigm, RECENT, is proposed by entity type restriction. RECENT reduces candidate relations for each pair of entity types by the mutual restrictions between relations and entity types. RECENT is model-agnostic. $\text{RECENT}_{\text{GCN}}$ and $\text{RECENT}_{\text{SpanBERT}}$ that are based on two representative models GCN and SpanBERT respectively, outperform their counterparts on the standard dataset TACRED, which empirically indicates the effectiveness of the proposed paradigm RECENT. Especially, $\text{RECENT}_{\text{SpanBERT}}$ achieves a new state-of-the-art on TACRED.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable and constructive comments on this research. This research is supported in part by the National Key Research and Development Program of China under Grant No. 2016YFB1000905, the National Natural Science Foundation of China under Grant No. 91746209 and the Fundamental Research Funds for the Central Universities.

References

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Amir D. N. Cohen, Shachar Rosenman, and Yoav Goldberg. 2020. Relation extraction as two-way span-prediction. *CoRR*, abs/2010.04829.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Daniel Kang and Tatsunori Hashimoto. 2020. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 718–731, Online. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, Toulon, France*.

S. Lyu, J. Cheng, X. Wu, L. Cui, H. Chen, and C. Miao. 2020. Auxiliary learning for relation extraction. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–10.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A.

Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240, Florence, Italy. Association for Computational Linguistics.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020a. K-Adapter: Infusing knowledge into pre-trained models with adapters. *CoRR*, abs/2002.01808v5.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2020b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *CoRR*, abs/1911.06136v3.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Xinyan Zhao, Feng Xiao, Haoming Zhong, Jun Yao, and Huanhuan Chen. 2020. Condition aware and revise transformer for question answering. In *Proceedings of The Web Conference*, page 2377–2387.

A The Statistics of False Positives

Table 3 presents false positives broken down by relations of our trained GCN on the TACRED dataset. In details, false positives broken down by relations are weakly negatively correlated with the amount of training data of relations, where the correlation coefficient is -0.39.

Relation	FP
org:alternate_names	53
org:city_of_headquarters	30
org:country_of_headquarters	70
org:dissolved	2
org:founded	9
org:founded_by	46
org:member_of	18
org:members	31
org:number_of_employees/members	9
org:parents	62
org:political/religious_affiliation	4
org:shareholders	10
org:stateorprovince_of_headquarters	16
org:subsidiaries	37
org:top_members/employees	61
org:website	1
per:age	13
per:alternate_names	11
per:cause_of_death	37
per:charges	35
per:children	31
per:cities_of_residence	90
per:city_of_birth	3
per:city_of_death	19
per:countries_of_residence	93
per:country_of_birth	5
per:country_of_death	9
per:date_of_birth	3
per:date_of_death	36
per:employee_of	98
per:origin	46
per:other_family	60
per:parents	49
per:religion	16
per:schools_attended	13
per:siblings	25
per:spouse	24
per:stateorprovince_of_birth	4
per:stateorprovince_of_death	10
per:stateorprovinces_of_residence	40
per:title	94

Table 3: False positives (FP) broken down by relations of our trained GCN on the TACRED dataset.

Link Prediction on N-ary Relational Facts: A Graph-based Approach

Quan Wang, Haifeng Wang, Yajuan Lyu, Yong Zhu
Baidu Inc., Beijing, China
`{wangquan05,wanghaifeng,lvyajuan,zhuyong}@baidu.com`

Abstract

Link prediction on knowledge graphs (KGs) is a key research topic. Previous work mainly focused on binary relations, paying less attention to higher-arity relations although they are ubiquitous in real-world KGs. This paper considers link prediction upon n-ary relational facts and proposes a graph-based approach to this task. The key to our approach is to represent the n-ary structure of a fact as a small heterogeneous graph, and model this graph with edge-biased fully-connected attention. The fully-connected attention captures universal inter-vertex interactions, while with edge-aware attentive biases to particularly encode the graph structure and its heterogeneity. In this fashion, our approach fully models global and local dependencies in each n-ary fact, and hence can more effectively capture associations therein. Extensive evaluation verifies the effectiveness and superiority of our approach. It performs substantially and consistently better than current state-of-the-art across a variety of n-ary relational benchmarks. Our code is publicly available.[1]

1 Introduction

Web-scale knowledge graphs (KGs), such as Freebase (Bollacker et al., 2008), Wikidata (Vrandečić and Krötzsch, 2014), and Google Knowledge Vault (Dong et al., 2014), are useful resources for many real-world applications, ranging from Web search and question answering to recommender systems. Though impressively large, these modern KGs are still known to be greatly incomplete and missing crucial facts (West et al., 2014). Link prediction which predicts missing links in KGs has therefore become an important research topic.

Previous studies mainly consider link prediction upon binary relational facts, which encode binary relations between pairs of entities and are usually represented as $(subject, relation, object)$ triples. Nevertheless, besides binary relational facts, n-ary relational facts that involve more than two entities are also ubiquitous in reality, e.g., *Marie Curie received Nobel Prize in Physics in 1903 together with Pierre Curie and Antoine Henri Becquerel* is a typical 5-ary fact. As pointed out by Wen et al. (2016), more than $1/3$ of the entities in Freebase actually participate in n-ary relational facts.

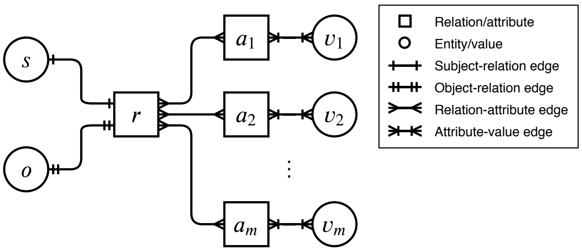

Figure 1: An n-ary fact as a heterogenous graph, with relations/attributes and entities/values as vertices, and four types of edges designed between the vertices.

Despite the ubiquitousness, only a few studies have examined link prediction on n-ary relational facts. In these studies, an n-ary fact is typically represented as a set of peer attributes (relations) along with their values (entities), e.g., {*person: Marie Curie*, *award: Nobel Prize in Physics*, *point-in-time: 1903*, *together-with: Pierre Curie*, *together-with: Antoine Henri Becquerel*}. Link prediction then is achieved by learning the relatedness either between the values (Zhang et al., 2018; Liu et al., 2020; Fatemi et al., 2020) or between the attribute-value pairs (Guan et al., 2019; Liu et al., 2021). This representation inherently assumes that attributes of a same n-ary fact are equally important, which is usually not the case. To further discriminate importance of different attributes, Rosso et al. (2020) and Guan et al. (2020) later proposed to represent an n-ary fact as a primary triple coupled with auxiliary attribute-value descriptions, e.g., in the above 5-ary

---

[1] `https://github.com/PaddlePaddle/Research/tree/master/KG/ACL2021_GRAN`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 396–407
August 1–6, 2021. ©2021 Association for Computational Linguistics

fact, (*Marie Curie*, *award-received*, *Nobel Prize in Physics*) is the primary triple and *point-in-time: 1903*, *together-with: Pierre Curie*, *together-with: Antoine Henri Becquerel* are auxiliary descriptions. Link prediction then is achieved by measuring the validity of the primary triple and its compatibility with each attribute-value pair. These attribute-value pairs, however, are modeled independently before a final aggregation, thus ignoring intrinsic semantic relatedness in between.

This work in general follows Rosso et al. (2020) and Guan et al. (2020)'s expressive representation form of n-ary facts, but takes a novel graph learning perspective for modeling and reasoning with such facts. Given an n-ary fact represented as a primary subject-relation-object triple (s, r, o) with auxiliary attribute-value pairs $\{(a_i : v_i)\}$, we first formalize the fact as a heterogenous graph. This graph, as we illustrate in Figure 1, takes relations and entities (attributes and values) as vertices, and introduces four types of edges, i.e., subject-relation, object-relation, relation-attribute, and attribute-value, to denote distinct connectivity patterns between these vertices. In this fashion, the full semantics of the given fact will be retained in the graph. Then, based on this graph representation, we employ a fully-connected attention module to characterize inter-vertex interactions, while further introducing edge-aware attentive biases to particularly handle the graph structure and heterogeneity. This enables us to capture not only local but also global dependencies within the fact. Our approach directly encodes each n-ary fact as a whole graph so as to better capture rich associations therein. In this sense, we call it GRAph-based N-ary relational learning (GRAN).

The most similar prior art to this work is STARE (Galkin et al., 2020), which uses a message passing based graph encoder to obtain relation (attribute) and entity (value) embeddings, and feeds these embeddings into a Transformer (Vaswani et al., 2017) decoder to score n-ary facts. Our approach is more neatly designed by (1) excluding the computational-heavy graph encoder which, according to a contemporaneous study (Yu and Yang, 2021), may not be necessary given an expressive enough decoder, and (2) modeling the full n-ary structure of a fact during decoding which enables to capture not only global but also local dependencies therein.

We evaluate our approach on a variety of n-ary link prediction benchmarks. Experimental results reveal that GRAN works particularly well in learning and reasoning with n-ary relational facts, consistently and substantially outperforming current state-of-the-art across all the benchmarks. Our main contributions are summarized as follows:

- We present a novel graph-based approach to learning and reasoning with n-ary facts, capable of capturing rich associations therein.
- We demonstrate the effectiveness and superiority of our approach, establishing new state-of-the-art across a variety of benchmarks.

2 Problem statement

This section formally defines n-ary relational facts and the link prediction task on this kind of data.

Definition 1 (N-ary relational fact) An n-ary relational fact $\mathcal{F}$ is a primary subject-relation-object triple (s, r, o) coupled with m auxiliary attribute-value pairs $\{(a_i : v_i)\}_{i=1}^m$, where $r, a_1, \cdots, a_m \in \mathcal{R}$ and $s, o, v_1, \cdots, v_m \in \mathcal{E}$, with $\mathcal{R}$ and $\mathcal{E}$ being the sets of relations and entities, respectively. We slightly abuse terminology here by referring to the primary relation and all attributes as relations, and referring to the subject, object, and values as entities unless otherwise specified. The arity of the fact is $(m+2)$, i.e., the number of entities in the fact.

Definition 2 (N-ary link prediction) N-ary link prediction aims to predict a missing element from an n-ary fact. The missing element can be either an entity $\in \{s, o, v_1, \cdots, v_m\}$ or a relation $\in \{r, a_1, \cdots, a_m\}$, e.g., to predict the primary subject of the incomplete n-ary fact $\big((?, r, o), \{(a_i : v_i)\}_{i=1}^m\big)$.

3 Graph-based n-ary relational learning

This section presents GRAN, our graph-based approach to n-ary link prediction. There are two key factors of our approach: graph representation and graph learning. The former represents n-ary facts as graphs, and the latter learns with these graphs to perform inference on n-ary facts.

3.1 Graph representation

We elaborate the first key factor: graph representation of n-ary facts. Given an n-ary fact defined as $\mathcal{F} = \big((s, r, o), \{(a_i : v_i)\}_{i=1}^m\big)$, we reformulate it equivalently as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$. The vertex set $\mathcal{V}$ consists of all entities and relations in the fact, i.e., $\mathcal{V} = \{r, s, o, a_1, \cdots, a_m, v_1, \cdots, v_m\}$. The link set $\mathcal{L}$ consists of $(2m+2)$ undirected edges of four types between the vertices, i.e.,

- 1 subject-relation edge (s, r),
- 1 object-relation edge (o, r),
- m relation-attribute edges $\{(r, a_i)\}_{i=1}^{m}$,
- m attribute-value edges $\{(a_i, v_i)\}_{i=1}^{m}$.

The graph heterogeneity is reflected in that the vertices and links are both typed, with type mapping functions $\phi\colon \mathcal{V} \rightarrow \{\textit{entity}, \textit{relation}\}$ and $\psi\colon \mathcal{L} \rightarrow \{$ *subject-relation*, *object-relation*, *relation-attribute*, *attribute-value*$\}$, respectively. Figure 1 provides a visual illustration of this heterogenous graph.

As we can see, the graph representation retains the full semantics of a given fact. It also enables us to model the fact as a whole and capture all possible interactions therein, which, as we will show later in our experiments, is crucial for learning with n-ary relational facts.

3.2 Graph learning

The second key factor is learning with heterogeneous graphs to perform inference on n-ary facts. Given an incomplete n-ary fact with a missing element, say $\big((?, r, o), \{(a_i : v_i)\}_{i=1}^{m}\big)$, which is represented as a heterogeneous graph, we feed the graph into an embedding layer, a stack of L successive graph attention layers, and a final prediction layer to predict the missing element, say s. This whole process is sketched in Figure 2 (left).

The input embedding layer maps the elements of the input n-ary fact or, equivalently, the vertices of the input graph, to their continuous vector representations (the missing element is denoted by a special token [MASK]). The L graph attention layers then repeatedly encode the graph and update its vertex representations. Our graph attention generally inherits from Transformer (Vaswani et al., 2017) and its fully-connected attention which captures universal inter-vertex associations, but further introduces edge-aware attentive biases to particularly handle graph structure and heterogeneity. As such, we call it edge-biased fully-connected attention. After the graph encoding process, we use the representation of the special token [MASK] to predict the missing element. In the rest of this section, we emphasize the edge-biased fully-connected attention, and refer readers to (Vaswani et al., 2017) and Appendix A for other modules of our graph attention layer.

Edge-biased fully-connected attention We are given an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, with vertex type mapping function ϕ and link type mapping function ψ. Vertices are associated with hidden states $(\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{V}|}) \in \mathbb{R}^{d_x}$ generated by previous layers. The aim of this attention is to aggregate information from different vertices and update vertex representations, by taking into account the graph structure and its heterogeneity. We employ multi-head attention with H heads, each applied independently to the input $(\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{V}|}) \in \mathbb{R}^{d_x}$ to generate updated vertex representations $(\mathbf{z}_1^h, \cdots, \mathbf{z}_{|\mathcal{V}|}^h) \in \mathbb{R}^{d_z}$ for $h = 1, \cdots, H$. These updated vertex representations are concatenated and linearly transformed to generate final attention output. We set $d_x = d$ and $d_z = \frac{d}{H}$ for all layers and heads. Below we describe the specific design of each head, and we drop the head index h for notational brevity.

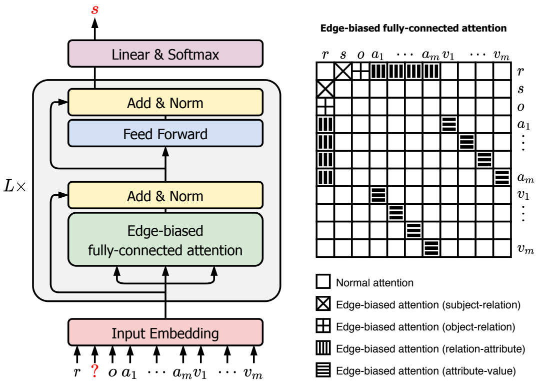

Figure 2: Overview of the graph learning process, with edge-biased fully-connected attention illustrated.

Our attention follows the traditional query-key-value attention (Vaswani et al., 2017). Specifically, for each input $\mathbf{x}_i$, we project it into a triple of query, key, and value as $(\mathbf{W}^Q\mathbf{x}_i, \mathbf{W}^K\mathbf{x}_i, \mathbf{W}^V\mathbf{x}_i) \in \mathbb{R}^{d_z}$, using parameters $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d_z \times d_x}$, respectively. Then we measure the similarity between each pair of vertices, say i and j, as a scaled dot product of i's query and j's edge-biased key:

$$\alpha_{ij} = \frac{(\mathbf{W}^Q\mathbf{x}_i)^\top(\mathbf{W}^K\mathbf{x}_j + \mathbf{e}_{ij}^K)}{\sqrt{d_z}}. \tag{1}$$

After we obtain the similarity scores α_{ij}, a softmax operation is applied, and the edge-biased values are aggregated accordingly to generate the updated representation for each vertex i:

$$\mathbf{z}_i = \sum_{j=1}^{|\mathcal{V}|} \frac{\exp(\alpha_{ij})}{\sum_{k=1}^{|\mathcal{V}|}\exp(\alpha_{ik})}(\mathbf{W}^V\mathbf{x}_j + \mathbf{e}_{ij}^V). \tag{2}$$

We call this attention fully-connected as it takes into account similarity between any two vertices i and j. We call it edge-biased as it further introduces attentive biases $\mathbf{e}_{ij}^K, \mathbf{e}_{ij}^V \in \mathbb{R}^{d_z}$ to encode the typed edge between i and j, one to generate edge-biased key (cf. Eq. (1)) and the other edge-biased value (cf. Eq. (2)). Introducing $\mathbf{e}_{ij}^K$ enables our attention to encode not only global dependencies that universally exist between any pair of vertices, but also local dependencies that are particularly indicated by typed edges. Introducing $\mathbf{e}_{ij}^V$ further propagates edge information to the attention output. If there is no edge linking i and j we set $\mathbf{e}_{ij}^K = \mathbf{e}_{ij}^V = \mathbf{0}$, which, at this time, degenerates to the conventional fully-connected attention used in Transformer (Vaswani et al., 2017). As the attentive biases $\mathbf{e}_{ij}^K, \mathbf{e}_{ij}^V$ can be designed freely to meet any desired specifications, this attention is in essence quite flexible, capable of modeling arbitrary relationships between the input elements. This idea has actually been applied, e.g., to model relative positions between words within sentences (Shaw et al., 2018; Wang et al., 2019a), or to model various kinds of mention dependencies for relation extraction (Xu et al., 2021).

Edge-aware attentive biases We now elaborate how $\mathbf{e}_{ij}^K$ and $\mathbf{e}_{ij}^V$ are specifically designed for n-ary facts. Recall that given an n-ary fact represented as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, there are 4 distinct types of edges in the graph: *subject-relation*, *object-relation*, *relation-attribute*, *attribute-value*. To each we assign a pair of key and value biases. The attentive biases between vertices i and j are then defined as the biases associated with the type of the edge linking i and j:

$$(\mathbf{e}_{ij}^K, \mathbf{e}_{ij}^V) = \begin{cases} (\mathbf{0}, \mathbf{0}), & \text{if } (i,j) \notin \mathcal{L}, \\ (\mathbf{e}_1^K, \mathbf{e}_1^V), & \text{if } \psi(i,j) = \textit{subject-relation}, \\ (\mathbf{e}_2^K, \mathbf{e}_2^V), & \text{if } \psi(i,j) = \textit{object-relation}, \\ (\mathbf{e}_3^K, \mathbf{e}_3^V), & \text{if } \psi(i,j) = \textit{relation-attribute}, \\ (\mathbf{e}_4^K, \mathbf{e}_4^V), & \text{if } \psi(i,j) = \textit{attribute-value}. \end{cases} \quad (3)$$

Here $\mathbf{e}_k^K, \mathbf{e}_k^V \in \mathbb{R}^{d_z}$ for $k = 1, 2, 3, 4$ are the key and value biases corresponding to the 4 edge types, shared across all layers and heads. In this way, the graph structure (whether there is an edge between two vertices) and its heterogeneity (which type the edge is between two vertices) can be well encoded into the attentive biases, and then propagated to the final attention output. Figure 2 (right) visualizes the edge-biased attention between pairs of vertices in an n-ary fact.

3.3 Model training

We directly use n-ary link prediction as our training task. Specifically, given an n-ary fact $\mathcal{F} = \big((s, r, o), \{(a_i : v_i)\}_{i=1}^m\big)$ in the training set we create $(2m + 3)$ training instances for it, each to predict a missing element (either an entity or a relation) given other elements in the fact, e.g., $\big((?, r, o), \{(a_i : v_i)\}_{i=1}^m\big)$ is to predict the primary subject and the answer to which is s. Here and in what follows we denote a training instance as $\widetilde{\mathcal{F}}$, with the missing element indicated by a special token [MASK]. This training instance is reformulated as a heterogeneous graph $\widetilde{\mathcal{G}}$ with vertices $(x_1, \cdots, x_k)$, where $k = 2m + 3$ is the total number of vertices therein. The label is denoted as y. We have $y \in \mathcal{E}$ for entity prediction and $y \in \mathcal{R}$ for relation prediction.

Each training instance $\widetilde{\mathcal{F}}$ or, equivalently, the corresponding graph $\widetilde{\mathcal{G}}$ is fed into the embedding, graph attention, and final prediction layers to predict the missing element, as we introduced above. Suppose after the successive graph attention layers we obtain for the vertices $(x_1, \cdots, x_k)$ their hidden states $(\mathbf{h}_1, \cdots, \mathbf{h}_k) \in \mathbb{R}^d$. The hidden state corresponding to [MASK], denoted as $\mathbf{h}$ for brevity, is used for the final prediction. The prediction layer is constructed by two linear transformations followed by a standard softmax operation:

$$\mathbf{p} = \textsc{Softmax}\big(\mathbf{W}_2^\top (\mathbf{W}_1 \mathbf{h} + \mathbf{b}_1) + \mathbf{b}_2\big). \quad (4)$$

Here, we share $\mathbf{W}_2$ with the weight matrix of the input embedding layer, and $\mathbf{W}_1, \mathbf{b}_1, \mathbf{b}_2$ are freely learnable. The final output $\mathbf{p}$ is a probability distribution over entities in $\mathcal{E}$ or relations in $\mathcal{R}$, depending on the type of the missing element.

We use the cross-entropy between the prediction and the label as our training loss:

$$L = \sum\nolimits_t y_t \log p_t, \quad (5)$$

where p_t is the t-th entry of the prediction $\mathbf{p}$, and y_t the t-th entry of the label $\mathbf{y}$. As a one-hot label restricts each prediction task to a single answer, which might not be the case in practice, we employ label smoothing to lessen this restriction. Specifically, for entity prediction, we set $y_t = 1 - \epsilon^{(e)}$ for the target entity and $y_t = \frac{\epsilon^{(e)}}{|\mathcal{E}|-1}$ for each of the other entities, where $\epsilon^{(e)}$ is a small entity label smoothing rate. For relation prediction y_t is set in a similar way, with relation label smoothing rate $\epsilon^{(r)}$. The loss is minimized using Adam optimizer (Kingma and Ba, 2015). We use learning rate warmup over the first 10% training steps and linear decay of the learning rate. We also use batch normalization and dropout after each layer and sub-layer to regularize, stabilize, and speed up training.

Unlike previous methods which score individual facts and learn from positive-negative pairs (Rosso et al., 2020; Guan et al., 2020), our training scheme bears two advantages: (1) Directly using n-ary link prediction as the training task can effectively avoid training-test discrepancy. (2) Introducing a special token [MASK] enables us to score a target element against all candidates simultaneously, which accelerates convergence during training and speeds up evaluation drastically (Dettmers et al., 2018).

4 Experiments and results

We evaluate GRAN in the link prediction task on n-ary facts. This section presents our experiments and results.

4.1 Datasets

We consider standard n-ary link prediction benchmarks including:

JF17K (Zhang et al., 2018)[2] is collected from Freebase. On this dataset, an n-ary relation is predefined by a set of attributes, and facts of this relation should have all corresponding values completely given. Take *music.group_membership* as an example. All facts of this relation should get three values w.r.t. the predefined attributes, e.g., (*Guitar*, *Dean Fertita*, *Queens of the Stone Age*). The maximum arity of the relations there is 6.

WikiPeople (Guan et al., 2019)[3] is derived from Wikidata concerning entities of type *human*. On this dataset, n-ary facts are already represented as primary triples with auxiliary attribute-value pairs, which is more tolerant to data incompleteness. The maximum arity there is 9. As the original dataset also contains literals, we follow (Rosso et al., 2020; Galkin et al., 2020) and consider another version that filters out statements containing literals. This filtered version is referred to as **WikiPeople**$^-$, and the maximum arity there is 7.

JF17K-3, **JF17K-4**, **WikiPeople-3**, and **WikiPeople-4** (Liu et al., 2020)[4] are subsets of JF17K and WikiPeople, consisting solely of 3-ary and 4-ary relational facts therein, respectively.

For JF17K and its subsets, we transform the representation of an n-ary fact to a primary triple coupled with auxiliary attribute-value pairs. We follow (Rosso et al., 2020; Galkin et al., 2020) and directly take the values corresponding to the first and second attributes as the primary subject and object, respectively. Other attributes and values are taken as auxiliary descriptions. Facts on each dataset are split into train/dev/test sets, and we use the original split. On JF17K which provides no dev set, we split 20% of the train set for development. The statistics of these datasets are summarized in Table 1.

4.2 Baseline methods

We compare against the following state-of-the-art n-ary link prediction techniques:

RAE (Zhang et al., 2018) represents an n-ary fact as an $(n+1)$-tuple consisting of the predefined relation and its n values. It generalizes a binary link prediction method TransH (Wang et al., 2014) to the higher-arity case, which measures the validity of a fact as the compatibility between its n values.

NaLP (Guan et al., 2019) and **RAM** (Liu et al., 2021) represent an n-ary fact as a set of attribute-value pairs. Then, NaLP employs a convolutional neural network followed by fully connected neural nets to model the relatedness of such attribute-value pairs and accordingly measure the validity of a fact. RAM further encourages to model the relatedness between different attributes and also the relatedness between an attribute and all involved values.

HINGE(Rosso et al., 2020) and **NeuInfer**(Guan et al., 2020) regard an n-ary fact as a primary triple with auxiliary attribute-value pairs. Then they deploy neural modules to measure the validity of the primary triple and its compatibility with each auxiliary description, and combine these modules to obtain the overall score of a fact. As different auxiliary descriptions are modeled independently before aggregation, these two methods show limited ability to model full associations within n-ary facts.

STARE (Galkin et al., 2020) is a recently proposed method generalizing graph convolutional networks (Kipf and Welling, 2017) to n-ary relational KGs. It employs a message passing based graph encoder to obtain entity/relation embeddings, and feeds these embeddings to Transformer decoder to score n-ary facts. **Hy-Transformer** (Yu and Yang, 2021) replaces the graph encoder with light-weight embedding processing modules, achieving higher efficiency without sacrificing effectiveness. These two methods employ vanilla Transformer decoders, ignoring specific n-ary structures during decoding.

n-CP, **n-TuckER**, and **GETD** (Liu et al., 2020) are tensor factorization approaches to n-ary link prediction. They all follow RAE and represent each n-ary fact as an $(n+1)$-tuple. A whole KG can thus be represented as a binary valued $(n+1)$-way ten-

---

[2] `https://github.com/lijp12/SIR/`
[3] `https://github.com/gsp2014/NaLP`
[4] `https://github.com/liuyuaa/GETD`

	All facts	Higher-arity facts (%)	Entities	Relations	Train	Dev	Test	Arity
JF17K	100,947	46,320 (45.9%)	28,645	501	76,379	–	24,568	2-6
WikiPeople	382,229	44,315 (11.6%)	47,765	193	305,725	38,223	38,281	2-9
WikiPeople$^-$	369,866	9,482 (2.6%)	34,825	178	294,439	37,715	37,712	2-7
JF17K-3	34,544	34,544 (100%)	11,541	208	27,635	3,454	3,455	3
JF17K-4	9,509	9,509 (100%)	6,536	69	7,607	951	951	4
WikiPeople-3	25,820	25,820 (100%)	12,270	112	20,656	2,582	2,582	3
WikiPeople-4	15,188	15,188 (100%)	9,528	95	12,150	1,519	1,519	4

Table 1: Dataset statistics, where the columns respectively indicate the number of all facts, n-ary facts with $n > 2$, entities, relations, facts in train/dev/test sets, and all possible arities.

sor $\mathcal{X} \in \{0,1\}^{|\mathcal{R}|\times|\mathcal{E}|\times\cdots\times|\mathcal{E}|}$, where $x = 1$ means the corresponding fact is true and $x = 0$ otherwise. $\mathcal{X}$ is then decomposed and approximated by a low-rank tensor $\hat{\mathcal{X}}$ that estimates the validity of all facts. Different tensor decomposition strategies can be applied, e.g., n-CP generalizes CP decomposition (Kruskal, 1977) and n-TuckER is built on TuckER (Balazevic et al., 2019). As the tensor representation inherently requires all facts to have the same arity, these methods are not applicable to datasets of mixed arities, e.g., JF17K and WikiPeople.

4.3 GRAN variants

We evaluate three variants of GRAN to investigate the impact of modeling graph structure and heterogeneity, including:

GRAN-hete is the full model introduced above. It uses edge representations defined in Eq. (3), which encode both graph structure (whether there is an edge) and heterogeneity (which type the edge is).

GRAN-homo retains graph structure but ignores heterogeneity. There are only two groups of edge attentive biases: $(\mathbf{e}_{ij}^K, \mathbf{e}_{ij}^V) = (\mathbf{0}, \mathbf{0})$ or $(\mathbf{e}_{ij}^K, \mathbf{e}_{ij}^V) = (\mathbf{e}^K, \mathbf{e}^V)$. The former is used if there is no edge between vertices i and j, while the latter is employed whenever the two vertices are linked, irrespective of the type of the edge between them. This in essence views an n-ary fact as a homogeneous graph where all edges are of the same type.

GRAN-complete considers neither graph structure nor heterogeneity. It simply sets $(\mathbf{e}_{ij}^K, \mathbf{e}_{ij}^V) = (\mathbf{0}, \mathbf{0})$ for all vertex pairs. The edge-biased attention thus degenerates to the conventional one used in Transformer, which captures only global dependencies between vertices. This in essence regards an n-ary fact as a complete graph in which any two vertices are connected by an (untyped) edge. STARE and Hy-Transformer are most similar to this variant.

We use the following configurations for all variants of GRAN: $L = 12$ graph attention layers, $H = 4$ attention heads, hidden size $d = 256$, batch size $b = 1024$, and learning rate $\eta = 5e{-}4$, fixed across all the datasets. Besides, on each dataset, we tune entity/relation label smoothing rate $\epsilon^{(e)}/\epsilon^{(r)}$, dropout rate ρ, and training epochs τ in their respective ranges. The optimal configuration is determined by dev MRR. We leave the tuning ranges and optimal values of these hyperparameters to Appendix B. After determining the optimal configuration on each dataset, we train with a combination of the train and dev splits and evaluate on the test split, as practiced in (Galkin et al., 2020).

4.4 Evaluation protocol and metrics

During evaluation, we distinguish between entity prediction and relation prediction. Take entity prediction as an example. For each test n-ary fact, we replace one of its entities (i.e., subject, object, or an auxiliary value) with the special token [MASK], feed the masked graph into GRAN, and obtain a predicted distribution of the answer over all entities $\in \mathcal{E}$. Then we sort the distribution probabilities in descending order and get the rank of the correct answer. During ranking, we ignore facts that already exist in the train, dev, or test split. We repeat this whole procedure for all specified entities in the test fact, and report MRR and Hits@k for $k = 1, 10$ aggregated on the test split. MRR is the average of reciprocal rankings, and Hits@k is the proportion of top k rankings (abbreviated as H@k). The same evaluation protocol and metrics also apply to relation prediction, where a relation can be either the primary relation or an auxiliary attribute.

4.5 Results on datasets of mixed arities

Table 2 presents entity prediction results on JF17K and the two versions of WikiPeople, which consist of facts with mixed arities.[5] We consider two set-

---

[5] Tensor factorization based approaches which require all facts to have the same arity are not applicable here.

	JF17K All Entities			JF17K Subject/Object			WikiPeople All Entities			WikiPeople$^-$ Subject/Object		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
RAE	.310	.219	.504	.215	.215	.467	.172	.102	.320	.059	.059	.306
NaLP	.366	.290	.516	.221	.165	.331	.338	.272	.466	.408	.331	.546
HINGE	–	–	–	.449	.361	.624	–	–	–	.476	.415	.585
NeuInfer	.517	.436	.675	–	–	–	.350	.282	.467	–	–	–
RAM	.539	.463	.690	–	–	–	.380	.279	.539	–	–	–
STARE	–	–	–	.574	.496	.725	–	–	–	.491	.398	**.648**
Hy-Transformer	–	–	–	.582	.501	.742	–	–	–	.501	.426	.634
GRAN-hete	**.656**	**.582**	**.799**	**.617**	**.539**	**.770**	**.479**	**.410**	**.604**	**.503**	**.438**	.620
GRAN-homo	.650	.576	.795	.611	.533	.767	.465	.386	.602	.487	.410	.618
GRAN-complete	.622	.546	.774	.591	.510	.753	.460	.381	.601	.489	.413	.617

Table 2: Entity prediction results on JF17K and WikiPeople. RAE and NaLP results for predicting all entities are collected from (Guan et al., 2020), and those for predicting the primary subject/object are collected from (Rosso et al., 2020). Other baseline results are collected from their original literatures. Best scores are highlighted in bold, and "–" denotes missing scores.

	JF17K All Relations			JF17K Primary Relation			WikiPeople All Relations			WikiPeople$^-$ Primary Relation		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
NaLP	.825	.762	.927	.639	.547	.822	.735	.595	.938	.482	.320	.852
HINGE	–	–	–	.937	.901	.989	–	–	–	.950	.916	**.998**
NeuInfer	.861	.832	.904	–	–	–	.765	.686	.897	–	–	–
GRAN-hete	**.996**	**.993**	**.999**	**.992**	**.988**	**.998**	**.960**	**.946**	**.977**	**.957**	**.942**	.976
GRAN-homo	.980	.965	.998	.964	.939	.997	.940	.910	.975	.932	.899	.971
GRAN-complete	.979	.963	.998	.963	.935	.997	.940	.910	.976	.935	.902	.974

Table 3: Relation prediction results on JF17K and WikiPeople. Baseline results for predicting all relations taken from (Guan et al., 2019, 2020), and those for predicting the primary relation taken from (Rosso et al., 2020). Best scores are highlighted in bold, and "–" denotes missing scores.

tings: (1) predicting all entities $s, o, v_1, \cdots, v_m$ in an n-ary fact and (2) predicting only the subject s and object o. This enables us to make a direct comparison to previous literatures (Guan et al., 2020; Rosso et al., 2020; Galkin et al., 2020). From the results, we can see that (1) The optimal setting of our approach offers consistent and substantial improvements over all the baselines across all the datasets in almost all metrics, showing its significant effectiveness and superiority in entity prediction within n-ary facts. (2) All the variants, including the less expressive GRAN-homo and GRAN-complete, perform quite well, greatly surpassing the competitive baselines in almost all cases except for the WikiPeople$^-$ dataset. This verifies the superior effectiveness of modeling n-ary facts as whole graphs so as to capture global dependencies between all relations and entities therein. (3) Among the variants, GRAN-hete offers the best performance. This demonstrates the necessity and superiority of further modeling specific graph structures and graph heterogeneity, so as to capture local dependencies reflected by typed edges linking relations and entities.

Table 3 further shows relation prediction results on these datasets. Again, to make direct comparison with previous literatures, we consider two settings: (1) predicting all relations including the primary relation r and auxiliary attributes $a_1, \cdots, a_m$ and (2) predicting only the primary relation r. Here, on each dataset, GRAN models are fixed to their respective optimal configurations (see Appendix B) determined in the entity prediction task.[6] The results show that GRAN variants perform particularly well in relation prediction. Among these variants, GRAN-hete performs the best, consistently outperforming the baselines and achieving extremely high performance across all the datasets. This is because relation prediction is, by nature, a relatively easy task due to a small number of candidate answers.

---

[6] Relation prediction typically requires much less training epochs than entity prediction according to our initial experiments. But we did not conduct hyperparameter searching for relation prediction, as the configurations we used, though not optimal, perform well enough in this task.

	JF17K-3 All Entities			JF17K-4 All Entities			WikiPeople-3 All Entities			WikiPeople-4 All Entities		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
RAE	.505	.430	.644	.707	.636	.835	.239	.168	.379	.150	.080	.273
NaLP	.515	.431	.679	.719	.673	.805	.301	.226	.445	.342	.237	.540
n-CP	.700	.635	.827	.787	.733	.890	.330	.250	.496	.265	.169	.445
n-TuckER	.727	.664	.852	.804	.748	.902	.365	.274	.548	.362	.246	.570
GETD	.732	.669	.856	.810	.755	.913	.373	.284	.558	.386	.265	.596
GRAN-hete	**.806**	**.759**	**.896**	**.852**	**.801**	**.939**	**.416**	**.325**	**.608**	**.431**	**.309**	**.642**
GRAN-homo	.803	.755	**.896**	.848	.795	.937	.410	.315	.606	.426	.305	.631
GRAN-complete	.730	.670	.862	.844	.794	.930	.408	.314	.602	.365	.248	.604

Table 4: Entity prediction results on the four JF17K and WikiPeople subsets. Baseline results are taken from (Liu et al., 2020). Best scores are highlighted in bold.

	JF17K			WikiPeople			WikiPeople$^-$		
	Subject/Object		Values	Subject/Object		Values	Subject/Object		Values
	$n=2$	$n>2$	$n>2$	$n=2$	$n>2$	$n>2$	$n=2$	$n>2$	$n>2$
GRAN-hete	.413	.768	.758	.495	.361	.471	.503	.505	.713
GRAN-homo	.409	.759	.753	.479	.354	.467	.487	.486	.690
GRAN-complete	.409	.725	.705	.478	.353	.415	.489	.480	.665

Table 5: Breakdown performance of the GRAN variants in entity prediction task on JF17K and WikiPeople. Only MRR scores are reported.

4.6 Results on datasets of a single arity

Table 4 presents entity prediction results on the four subsets of JF17K and WikiPeople, which consist solely of 3-ary or 4-ary facts. Here, an entity means either the subject, the object, or an attribute value. On these four single-arity subsets, tensor factorization based approaches like n-CP, n-TuckER, and GETD apply quite well and have reported promising performance (Liu et al., 2020). From the results, we can observe similar phenomena as from Table 2. The GRAN variants perform particularly well, all surpassing or at least performing on par with the baselines across the datasets. And GRAN-hete, again, offers the best performance in general among the three variants.

4.7 Further analysis

We further look into the breakdown entity prediction performance of the GRAN variants on different arities. More specifically, we group the test split of each dataset into binary ($n = 2$) and n-ary ($n > 2$) categories. Entity prediction means predicting the subject/object for the binary category, or predicting an attribute value in addition for the n-ary category. Table 5 presents the breakdown MRR scores in all these different cases on JF17K, WikiPeople, and WikiPeople$^-$, with the GRAN variants set to their respective optimal configurations on each dataset (see Appendix B). Among the variants GRAN-hete performs best in all cases, which again verifies the necessity and superiority of modeling n-ary facts as heterogeneous graphs. Ignoring the graph heterogeneity (GRAN-homo) or further graph structures (GRAN-complete) always leads to worse performance, particularly when predicting auxiliary attribute values in higher-arity facts.

5 Related work

Link prediction on binary relational data Most previous work of learning with knowledge graphs (KGs) focused on binary relations. Among different binary relational learning techniques, embedding based models have received increasing attention in recent years due to their effectiveness and simplicity. The idea there is to represent symbolic entities and relations in a continuous vector space and measure the validity of a fact in that space. This kind of models can be roughly grouped into three categories: translation distance based (Bordes et al., 2013; Wang et al., 2014; Sun et al., 2019), semantic matching based (Trouillon et al., 2016; Balazevic et al., 2019), and neural network based (Dettmers et al., 2018; Schlichtkrull et al., 2018), according to the design of validity scoring functions. We refer readers to (Nickel et al., 2016; Wang et al., 2017; Ji et al., 2021) for thorough reviews of the literature.

Link prediction on n-ary relational data Since binary relations oversimplify the complex nature of the data stored in KGs, a few recent studies have started to explore learning and reasoning with n-ary relational data ($n > 2$), in particular via embedding based approaches. Most of these studies represent n-ary facts as tuples of pre-defined relations with corresponding attribute values, and generalize binary relational learning methods to the n-ary case, e.g., m-TransH (Wen et al., 2016) and RAE (Zhang et al., 2018) generalize TransH (Wang et al., 2014), a translation distance based embedding model for binary relations, while n-CP, n-TuckER, and GETD (Liu et al., 2020) generalize 3-way tensor decomposition techniques to the higher-arity case. NaLP (Guan et al., 2019) and RAM (Liu et al., 2021) are slightly different approaches which represent n-ary facts directly as groups of attribute-value pairs and then model relatedness between such attributes and values. In these approaches, however, attributes of an n-ary fact are assumed to be equally important, which is often not the case in reality. Rosso et al. (2020) and Guan et al. (2020) therefore proposed to represent n-ary facts as primary triples coupled with auxiliary attribute-value pairs, which naturally discriminates the importance of different attributes. The overall validity of a fact is then measured by the validity of the primary triple and its compatibility with each attribute-value pair. STARE (Galkin et al., 2020) follows the same representation form of n-ary facts, and generalizes graph convolutional networks to n-ary relational KGs to learn entity and relation embeddings. These embeddings are then fed into a Transformer decoder to score n-ary facts. Nevertheless, during the decoding process STARE takes into account solely global dependencies and ignores the specific n-ary structure of a given fact.

Transformer and its extensions Transformer (Vaswani et al., 2017) was initially devised as an encoder-decoder architecture for machine translation, and quickly received broad attention across all areas of natural language processing (Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019). Transformer uses neither convolution nor recurrence, but instead is built entirely with (self-) attention layers. Recently, there has been a lot of interest in modifying this attention to further meet various desired specifications, e.g., to encode syntax trees (Strubell et al., 2018; Wang et al., 2019c), character-word lattice structures (Li et al., 2020), as well as relative positions between words (Shaw et al., 2018; Wang et al., 2019a). There are also a few recent attempts that apply vanilla Transformer (Wang et al., 2019b) or hierarchical Transformer (Chen et al., 2020) to KGs, but mainly restricted to binary relations and deployed with conventional attention. This work, in contrast, deals with higher-arity relational data represented as heterogeneous graphs, and employs modified attention to encode graph structure and heterogeneity.

6 Conclusion

This paper studies link prediction on higher-arity relational facts and presents a graph-based approach to this task. For each given n-ary fact, our approach (1) represents the fact as a heterogeneous graph in which the semantics of the fact are fully retained; (2) models the graph using fully-connected attention with edge-aware attentive biases so as to capture both local and global dependencies within the given fact. By modeling an n-ary fact as a whole graph, our approach can more effectively capture entity relation associations therein, which is crucial for inference on such facts. Link prediction results on a variety of n-ary relational benchmarks demonstrate the significant effectiveness and superiority of our approach.

As future work, we would like to (1) verify the effectiveness of GRAN on newly introduced benchmarks such as WD50K (Galkin et al., 2020) and FB-AUTO (Fatemi et al., 2020); (2) investigate the usefulness of specific modules, e.g., positional embeddings and various forms of attentive biases in GRAN; and (3) integrate other types of data in a KG, e.g., entities's textual descriptions, for better n-ary link prediction.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful suggestions. This work is supported by the National Key Research and Development Program of China (No.2020AAA0109400) and the Natural Science Foundation of China (No. 61876223).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5188–5197.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2020. HittER: Hierarchical Transformers for knowledge graph embeddings. *arXiv preprint arXiv:2008.12813*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2020. Knowledge hypergraphs: Prediction beyond binary relations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 2191–2197.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7346–7359.

Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. NeuInfer: Knowledge inference on n-ary facts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6141–6151.

Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on n-ary relational data. In *Proceedings of the 2019 World Wide Web Conference*, pages 583–593.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference for Learning Representations*.

Joseph B Kruskal. 1977. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and Its Applications*, 18(2):95–138.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. FLAT: Chinese NER using flat-lattice Transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6836–6842.

Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing tensor decomposition for n-ary relational knowledge bases. In *Proceedings of The Web Conference 2020*, pages 1104–1114.

Yu Liu, Quanming Yao, and Yong Li. 2021. Role-aware modeling for n-ary relational knowledge bases. *arXiv preprint arXiv:2104.09780*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020*, pages 1885–1896.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 464–468.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference for Learning Representations*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57(10):78–85.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019a. Extracting multiple-relations in one-pass with pre-trained Transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1371–1377.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019b. CoKE: Contextualized knowledge graph embedding. *arXiv preprint arXiv:1911.02168*.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019c. Tree Transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1061–1070.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1300–1307.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 515–526.

Benfeng Xu, Quan Wang, Yajuan Lyu, Yong Zhu, and Zhendong Mao. 2021. Entity structure within and throughout: Modeling mention dependencies for document-level relation extraction. *arXiv preprint arXiv:2102.10249*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5753–5763.

Donghan Yu and Yiming Yang. 2021. Improving hyper-relational knowledge graph completion. *arXiv preprint arXiv:2104.08167*.

Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*, pages 1185–1194.

A Graph attention layers

After the input embedding layer, we employ a stack of L identical graph attention layers to encode the input graph before making final predictions. These graph attention layers generally follow the design of Transformer encoder (Vaswani et al., 2017), each of which consists of two sub-layers, i.e., an edge-biased fully-connected attention sub-layer followed by an element-wise feed-forward sub-layer. The attention sub-layer, as illustrated in Section 3.2, relates different vertices of the input graph to update its vertex representations. It computes attention in an edge-biased fully-connected fashion, which thus is able to capture both global and local dependencies within the graph. The feed-forward sub-layer is composed of two linear transformations with a GELU activation (Hendrycks and Gimpel, 2016) in between, applied to each element/vertex separately and identically. We further introduce residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) around each graph attention layer and its sub-layers. To facilitate these residual connections, all the layers and their sub-layers produce outputs of the same dimension d.

B Hyperparameter settings

We use the following hyperparameter settings for GRAN: $L=12$ graph attention layers, $H=4$ attention heads, hidden size $d=256$, batch size b=1024, and learning rate $\eta=5e-4$. These configurations are fixed across all the datasets. Besides, on each dataset we tune the following hyperparameters in their respective ranges:

- entity label smoothing rate $\epsilon^{(e)} \in \{.0, .1, .2, .3, .4, .5, .6, .7, .8, .9\}$;
- relation label smoothing rate $\epsilon^{(r)} \in \{.0, .1, .2, .3, .4, .5\}$;
- dropout rate $\rho \in \{.1, .2, .3, .4, .5\}$;
- training epochs τ from 20 to 200 in steps of 20 on all the datasets.

We determine the optimal configuration for GRAN-hete by dev MRR of entity prediction on each dataset. And then we directly set GRAN-homo and GRAN-complete to the same configuration. Table 6 presents the optimal configuration on each dataset.

	$\epsilon^{(e)}$	$\epsilon^{(r)}$	ρ	τ
JF17K	.9	.0	.2	160
WikiPeople	.2	.2	.1	200
WikiPeople$^-$	.2	.1	.1	160
JF17K-3	.8	.2	.2	180
JF17K-4	.8	.0	.3	160
WikiPeople-3	.8	.4	.3	100
WikiPeople-4	.8	.4	.3	100

Table 6: Optimal configuration of the GRAN variants on each dataset.

C Infrastructure and runtime

We train all the GRAN variants on one 16G V100 GPU. With the hyperparameter settings specified in Appendix B, it takes about 3 hours to finish training and evaluation on JF17K, 17 hours on Wikipeople, 10 hours on Wikipeople$^-$, 1 hour on JF17K-3, 0.5 hour on JF17K-4, Wikipeople-3, and WikiPeople-4. This runtime covers the whole training and evaluation process. Compared to previous methods like HINGE (Rosso et al., 2020) and NeuInfer (Guan et al., 2020) which score individual facts and learn from positive-negative pairs, GRAN directly scores each target answer against all candidates in a single pass and drastically speeds up evaluation. GRAN is also much more efficient than STARE (Galkin et al., 2020), which is a graph encoder plus Transformer decoder architecture. By eliminating the computational heavy graph encoder, GRAN requires significantly less running time but still achieves better performance than STARE, e.g., GRAN-hete achieves .617 MRR within 3 hours while STARE takes about 10 hours to achieve .574 MRR on JF17K; GRAN-hete achieves .503 MRR within 10 hours but STARE takes about 4 days to achieve a similar MRR on Wikipeople$^-$ (which is 9-10 times slower).

GLGE: A New General Language Generation Evaluation Benchmark

Dayiheng Liu[♠,*] **Yu Yan**[†,♡] **Yeyun Gong**[†,♣] **Weizhen Qi**[♣] **Hang Zhang**[♣]
Jian Jiao[♡] **Weizhu Chen**[‡] **Jie Fu**[◇] **Linjun Shou**[◆] **Ming Gong**[◆]
Pengcheng Wang[♡] **Jiusheng Chen**[♡] **Daxin Jiang**[◆] **Jiancheng Lv**[♠]
Ruofei Zhang[♡] **Winnie Wu**[♡] **Ming Zhou**[♣] **Nan Duan**[♣]
[♠]College of Computer Science, Sichuan University [♣]Microsoft Research Asia [◇] Mila
[‡]Microsoft Azure AI [◆]Microsoft Search Technology Center Asia [♡]Microsoft
`glge@microsoft.com` [†]Equal contribution

Abstract

Multi-task benchmarks such as GLUE and SuperGLUE have driven great progress of pretraining and transfer learning in Natural Language Processing (NLP). These benchmarks mostly focus on a range of Natural Language Understanding (NLU) tasks, without considering the Natural Language Generation (NLG) models. In this paper, we present the **G**eneral **L**anguage **G**eneration **E**valuation (**GLGE**), a new multi-task benchmark for evaluating the generalization capabilities of NLG models across eight language generation tasks. For each task, we continue to design three subtasks in terms of task difficulty (**GLGE-Easy**, **GLGE-Medium**, and **GLGE-Hard**). This introduces 24 subtasks to comprehensively compare model performance. To encourage research on pretraining and transfer learning on NLG models, we make GLGE publicly available and build a leaderboard with strong baselines including MASS, BART, and ProphetNet[1].

1 Introduction

Pretrained language models, such as BERT (Devlin et al., 2019) and other advanced pretrained models (Raffel et al., 2020; Yang et al., 2019; Liu et al., 2019; Alberti et al., 2019; Brown et al., 2020; Clark et al., 2020) have made great progress in a host of Natural Language Understanding (NLU) tasks. Meanwhile, the development of general evaluation benchmarks has also helped drive the progress of these models. These benchmarks usually use an overall score to evaluate the performance of models across a wide range of NLU tasks. In addition to GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) which are general language understanding evaluation benchmarks for English, several general language understanding evaluation benchmarks for other languages are proposed, such as CLUE (Xu et al., 2020) for Chinese, FLUE (Le et al., 2020) for French, and IndoNLU (Wilie et al., 2020) for Indonesian. Furthermore, the multilingual multi-task benchmarks such as XTREME (Hu et al., 2020) and XGLUE (Liang et al., 2020) are proposed for cross-lingual evaluation.

In addition to NLU tasks, an increasing number of pretrained language models designed for Natural Language Generation (NLG) tasks have recently been proposed, such as MASS (Song et al., 2019), BERT-share (Rothe et al., 2020), BART (Lewis et al., 2020), ProphetNet (Qi et al., 2020), and ERINE-GEN (Xiao et al., 2020). However, the generalization capabilities of the language generation of these models are usually evaluated with different tasks, datasets, and metrics, which cannot provide a coherent and comprehensive evaluation. Although there are several general evaluation benchmarks as we mentioned above, none of them are particularly designed for general language generation evaluation.

To fill the gap of the NLG evaluation benchmark, we introduce the **G**eneral **L**anguage **G**eneration **E**valuation (**GLGE**) benchmark, a new multi-task benchmark for evaluating the generalization capabilities of NLG in English language. It contains eight English language generation tasks, covering text summarization, question generation, generative question answering, and dialogue. We select six pre-existing popular datasets and introduce two new datasets selected from real-world scenarios. Moreover, in order to provide more diversified difficulty challenges, we employ two simple but effective strategies to build three NLG evaluation benchmarks (called **GLGE-Easy**, **GLGE-Medium**, and **GLGE-Hard**) in terms of task difficulty.

---

[*] Work is done during internship at Microsoft Research Asia.

[1] The source code and dataset are publicly available at `https://github.com/microsoft/glge`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 408–420
August 1–6, 2021. ©2021 Association for Computational Linguistics

To better understand the challenges posed by GLGE, we conduct experiments with existing widely used non-pretrained models (e.g., vanilla LSTM Seq2Seq (Bahdanau et al., 2015), vanilla Transformer (Vaswani et al., 2017)), and pretrained models (e.g., MASS (Song et al., 2019), BART (Lewis et al., 2020), and ProphetNet (Qi et al., 2020)). We further analyze the n-gram diversity of the output samples. The experimental results show that there is a large performance gap between the pretrained models and the non-pretrained models. However, on the GLGE-hard task, the performance of the pretrained models still has great room for improvement.

In summary, the contributions of this work are five-fold: (1) a new multi-task NLG evaluation benchmark consisting of eight distinct datasets across four kinds of typical NLG tasks, (2) three NLG evaluation benchmarks of different difficulty levels, (3) standardized evaluation metrics and scripts for model evaluation and comparison, (4) open-sourced baselines and a public leaderboard[2] for the benchmark, (5) a thorough comparative study on existing widely used non-pretrained models and pretrained models with a detailed analysis of the results.

2 GLGE Benchmark

2.1 Design Principles

For the GLGE benchmark, we design and select the NLG tasks based on the following principles:

2.1.1 Task Diversity

The tasks in GLGE focus on evaluating the generalization capabilities of a NLG model, varying the task, the length of the input text, the length of the output text, the type of generated text, and the size of the dataset.

2.1.2 Task Difficulty

The tasks in GLGE should be challenging but solvable, which can encourage researchers to design better NLG models. Furthermore, we aim to provide benchmarks of different difficulty levels like GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a), which allows researchers to comprehensively evaluate the models. Researchers can also select the benchmark with moderate difficulty according to the size of the model and the scale of the used pretraining corpus for comparison.

2.1.3 Ease of Evaluation

The tasks in GLGE should be easily evaluated automatically. For some unconditional, open-ended, and weak conditional language generation tasks (e.g., answer-agnostic question generation, single-turn chit-chat response generation, and story generation), reasonable generation results are diverse. Due to the limited number of references in the automatic evaluation of text generation tasks, it is more difficult for automatic evaluation of those tasks. Therefore, instead of selecting unconditional and weak conditional language generation tasks, we tend to select language generation tasks with stronger conditions (e.g., answer-aware question generation), which makes the automatic evaluation more convincing.

2.1.4 Task Popularity

Most tasks in GLGE should use widely-used NLG datasets, which have been implicitly agreed upon by the NLG community as challenging and meaningful. Since GLGE is mainly designed for the generalization capabilities evaluation of the English NLG pretrained model, the choice of task also refers to several related works of NLG pretraining model, such as MASS (Song et al., 2019), BART (Lewis et al., 2020), ProphetNet (Qi et al., 2020), and ERNIE-GEN (Xiao et al., 2020).

Based on the above principles, we invite 10 NLG experts[3] to discuss and vote on existing widely-used NLG datasets. Note that since the GLGE is designed for evaluating the generalization capabilities of NLG in English language, we do not include the cross-lingual NLG tasks, such as machine translation and cross-lingual text summarization (Liang et al., 2020). Finally, we select 6 existing popular NLG datasets. Besides, we also introduce two new datasets selected from real-world scenarios for the GLGE benchmark, which makes GLGE have more practical values. Unlike the existing datasets, the test sets of these new two datasets are hidden, which further ensures the fairness of the evaluation results. The input sequence and output sequence of the selected tasks are all well-defined. We preprocess them and provide the input and output sequence pairs directly, which benefits researchers to focus on model improvements.

---

[2] `https://microsoft.github.io/glge/`.

[3] Each expert has experience in publishing multiple papers at top conferences in the NLG field.

Corpus	|Train|	|Dev|	|Test|	|Src.|	|Tgt.|	Input	Output	Metric
Abstractive Text Summarization								
CNN/DailyMail	287,113	13,368	11,490	822.3	57.9	article	summary	R-1/R-2/R-L
Gigaword	3,803,957	189,651	1,951	33.7	8.7	passage	headline	R-1/R-2/R-L
XSUM	204,017	11,327	11,333	358.5	21.1	article	summary	R-1/R-2/R-L
MSNews	136,082	7,496	7,562	310.7	9.7	article	headline	R-1/R-2/R-L
Answer-aware Question Generation								
SQuAD 1.1	75,722	10,570	11,877	149.4	11.5	answer/passage	question	R-L/B-4/MTR
MSQG	198,058	11,008	11,022	45.9	5.9	highlight/passage	question	R-L/B-4/MTR
Conversational Question Answering								
CoQA	108,647	3,935	4,048	354.4	2.6	history/passage	answer	F1-Score
Personalizing Dialogue								
PersonaChat	122,499	14,602	14,056	120.8	11.8	persona/history	response	B-1/B-2/D-1/D-2

Table 1: GLGE task descriptions and statistics. |Train|: the number of examples in train set. |Src.|: the average number of words in source inputs. R-L: ROUGE-L. B-4: BLUE-4. MTR: METEOR. D-2: Distinct-2.

2.2 Tasks and Datasets

GLGE contains eight English NLG tasks, covering text summarization, question generation, generative question answering, and dialogue. Descriptions and statistics of these tasks are shown in Table 1, with concrete examples shown in Appendix.

2.2.1 Abstractive Text Summarization

As a typical NLG task, abstractive text summarization aims to generate a short and fluent summary of a long text document. GLGE contains four abstractive text summarization tasks. As discussed in Bhandari et al. (2020), we use ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) as the metrics for these tasks.

CNN/DailyMail (Hermann et al., 2015) dataset contains 220K articles from the Daily Mail newspapers and 93K articles from the CNN. Each article contains a bullet point summary. GLGE uses the non-anonymized variant See et al. (2017). After the pre-processing, there are 311,971 ⟨article, summary⟩ pairs, where the source input is the article, and the target output is the summary which consists of multiple sentences.

Gigaword (Rush et al., 2015) contains 4M examples extracted from the news articles of the Gigaword corpus (Graff et al., 2003). After the pre-processing, there are 3,995,559 ⟨passage, summary⟩ data pairs, where the source input is the first sentence of the article, and the target output is the headline that usually contains a single sentence.

XSum (Narayan et al., 2018) consists of 227K online articles from the British Broadcasting Corporation (BBC), which contains professionally written single-sentence summaries. After the pre-processing, there are 226,677 ⟨article, summary⟩ data pairs, where the source input is the news article, and the target output is a single-sentence summary.

MSNews **M**icro**S**oft **News** headline generation (**MSNews**) is a new News headline generation dataset we collected for GLGE. We random select 151K online news articles from 2012-01-01 to 2020-09-01 from a real-world news search engine. Each article contains a professionally written single-sentence headline. After the pre-processing, there are 151,140 ⟨article, headline⟩ data pairs, where the source input is the news article, and the target output is a news headline.

2.2.2 Answer-aware Question Generation

The question generation task is another typical NLG task, which aims to generate a question based on a given text passage or document. Compared with answer-agnostic question generation tasks that can generate lots of reasonable questions, answer-aware question generation (Zhou et al., 2017) is asked to generate a question asks towards the given answer span based on a given text passage or document. In order to facilitate automatic evaluation, GLGE selects two answer-aware question generation tasks:

SQuAD 1.1 (Rajpurkar et al., 2016) dataset contains over 100K crowd-worker created questions with the corresponding answer spans in 536 Wikipedia articles. Since the original hidden test set of the SQuAD 1.1 is hidden, we follow (Du et al., 2017; Zhao et al., 2018) to re-split the dataset with the examples from the original training set and development set. After the pre-processing, there are 98,169 ⟨answer, passage, question⟩ data triples, in which the source input is a Wikipedia passage along with an answer span, and the target output is a question. ROUGE-L, BLEU-4 (Papineni et al., 2002), and METEOR (Banerjee and Lavie, 2005) are used as the metrics.

MSQG **M**icro**S**oft **Q**uestion **G**eneration (**MSQG**) is another dataset we collected, which is a new challenge dataset, the questions in this dataset are freely edited by daily users. For MSQG, we collect 220K passages from a real world search engine. Each passage contains a highlight span and a related query, we regard the queries as questions in this dataset. After the pre-processing, there are 220,088 ⟨highlight span, passage, question⟩ data triples, where the source input is a news passage along with highlight span, and the target output is a user question. ROUGE-L, BLEU-4, and METEOR are used as the metrics.

2.2.3 Conversational Question Answering

Conversational question answering is a classic and popular generative question answering task. Compared with the extractive question answering, such as SQuAD (Rajpurkar et al., 2016), conversational question answering requires the model to answer the question based on a running conversation history and the given passage.

CoQA (Reddy et al., 2019) dataset contains 127K questions with answers, obtained from 8K conversations about text passages from seven diverse domains. After the pre-processing, there are 116,630 ⟨conversation history, passage, question, answer⟩ data 4-tuples, where the source input is a sequence of conversation history along with a given question and a given passage, and the target output is a free-form answer text. F1-Score (Rajpurkar et al., 2016) is used as the metric.

2.2.4 Personalized Dialogue

Conversational AI is an important topic in NLG. Compared with text summarization, the responses of single-turn conversations are diverse and might lack of specification, and thus it is hard to use the single ground-truth for automatic evaluation. We select the personalizing dialogue task, which is a challenging multi-turn conversation task. In addition to the conversation history, this task gives the profile information as an additional condition to facilitate specific response generation.

PersonaChat (Zhang et al., 2018) dataset consists of about 160K utterances, which requires the model to generate responses according to given multi-turn conversations and persona profile. After pre-processing, there are 151,157 ⟨persona profile description text, conversation history, response⟩ data triples, where the source input is a sequence of conversation history along with several sentences of persona profile description information, and the target output is a response. BLEU-1, BLEU-2, Distinct-1, and Distinct-2 (Li et al., 2016) are used as the evaluation metrics.

2.3 Overall Score

Similar to GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a), we seek to give an overall system performance over all GLGE tasks by aggregating the scores of all tasks. We follow GLUE to adopt a simple approach that weighs each task equally. For the tasks with multiple metrics, we firstly average those metrics to get a task score. Besides, because the values of the original Distinct-1 (D-1) and Distinct-2 (D-2) (Li et al., 2016) scores which are used as the metrics for dialogue task are usually quite small (less than 0.01), we re-scale them by 100.0 so that these score values are in the same order of magnitude as other scores.

2.4 Challenges of Three Difficulty Levels

As discussed in § 2.1.2, GLGE provides three levels of difficulty for each task, called **GLGE-Easy**, **GLGE-Medium**, and **GLGE-Hard**. The original 8 task datasets as described in § 2.2 constitute the GLGE-Easy. Based on GLGE-Easy, we employ two strategies to further increase the task difficulty.

Low-resource. We increase the difficulty of GLGE by simulating low-resource scenarios. For each task, we keep the test and development sets of GLGE-Easy and randomly reduce the scale of the training data to 50% of the original train set. The dataset of 8 tasks under this setting is regarded as GLGE-Medium.

Low-frequency. In order to further evaluate the generalization capability of the NLG model, we increase the difficulty of GLGE by reducing the word overlap rate between the output of the training set and the output of the test set. The motivation is that a good NLG model should be able to generate a fluent target output based on the input information, even if the target output may contain some low-frequency words. For the test set and development sets of GLGE-Hard, we still use those in GLGE-

Easy. For the training set of GLGE-Hard, we first count the frequency of each token in the target sentence of the test set. Then we remove the stop words of the target sentence of each training sample in GLGE-Easy, ranking them by calculating their word frequency score of the test set. This can be formulated as

$$\text{Score}(y) = \frac{\sum_{w_y \in y} \text{TF}(w_y)}{|y|}, \tag{1}$$

where y is a target sentence without stop words, w_y is a token in y, $\text{TF}(w_y)$ denotes the word frequency of the token w_y in the target sentences of the whole test set, and $|y|$ denotes the token length of y. Instead of reducing the training data scale randomly as in GLGE-Medium, we select the top 25% training data with minimum word frequency score of the test set from the original training set as the training set of each dataset. The dataset of 8 tasks under this setting is regarded as GLGE-Hard.

3 Experiments

3.1 Baselines

For the baselines, we first evaluate two widely-used non-pretrained models: vanilla LSTM based Seq2Seq (Bahdanau et al., 2015) and vanilla Transformer (Vaswani et al., 2017). Besides, we evaluate several widely used pretrained NLG models, including MASS (Song et al., 2019), BART (Lewis et al., 2020), and ProphetNet (Qi et al., 2020). To further evaluate the performance of the pretrained NLG models of different model sizes and the different scales of the pretraining corpus, we compare the $\text{MASS}_{\text{base}}$, $\text{ProphetNet}_{\text{base}}$, $\text{MASS}_{\text{middle}}$, $\text{BART}_{\text{large}}$, and $\text{ProphetNet}_{\text{large}}$ on GLGE.

3.2 Implementation Details

Vanilla LSTM (Bahdanau et al., 2015). The hyper-parameters and implementation of LSTM-Seq2Seq are based on the *LSTM* register model of Fairseq[4], where the word embedding dimension, the hidden size, the number of the encoder layer, and the number of the decoder layer are 512, 512, 1, and 1, respectively. For each task in GLGE, we use Adam (Kingma and Ba, 2015) with an initial learning rate of between 0.0001 and 0.0003, and train the LSTM-Seq2Seq for a maximum of 100 epochs.

Vanilla Transformer (Vaswani et al., 2017). The hyper-parameters and implementation of Transformer are based on the *transformer_vaswani_wmt_en_de_big* register model of Fairseq[5], which contains a 6-layer encoder and a 6-layer decoder with 1024 embedding/hidden size and 4096 feed-forward filter size. For each task in GLGE, we use Adam with the initial learning rate of between 0.0003 and 0.001, and train the Transformer for a maximum of 20 epochs.

$\text{MASS}_{\text{base}}$ (Song et al., 2019). The hyper-parameters and implementation of MASS are based on their source code[6]. $\text{MASS}_{\text{base}}$ contains a 6-layer encoder and a 6-layer decoder with 768 embedding/hidden size and 3072 feed-forward filter size. The $\text{MASS}_{\text{base}}$ is pretrained on BookCorpus (Zhu et al., 2015) and English Wikipedia (16GB in total). For each task in GLGE, we fine-tune $\text{MASS}_{\text{base}}$ with the same hyper-parameters used in their source code[7] for a maximum of 25 epochs.

$\text{MASS}_{\text{middle}}$ (Song et al., 2019) which contains a 6-layer encoder and a 6-layer decoder with 1024 embedding/hidden size and 4096 feed-forward filter size. The $\text{MASS}_{\text{middle}}$ is also pretrained on BookCorpus and English Wikipedia (16GB in total). For each task in GLGE, we use the same hyper-parameters as used in $\text{MASS}_{\text{base}}$.

$\text{ProphetNet}_{\text{base}}$ (Qi et al., 2020). The hyper-parameters and implementation of ProphetNet are based on their source code[8]. $\text{ProphetNet}_{\text{base}}$ contains a 6-layer encoder and a 6-layer decoder with 768 embedding/hidden size and 3072 feed-forward filter size. Similar to MASS, the $\text{ProphetNet}_{\text{base}}$ is pretrained on BookCorpus and English Wikipedia (16GB in total) with 125K steps. For each task in GLGE, we fine-tune $\text{ProphetNet}_{\text{base}}$ with the same hyper-parameters used in their source code for a maximum of 10 epochs.

$\text{ProphetNet}_{\text{large}}$ (Qi et al., 2020). It contains a 12-layer encoder and 12-layer decoder with 1024 embedding/hidden size and 4096 feed-forward filter size. The $\text{ProphetNet}_{\text{large}}$ is pretrained on the 160GB English language corpora of news, books, stories, and web text for 14 epochs. For each task in

---

[4] `https://github.com/pytorch/fairseq/blob/master/fairseq/models/lstm.py.`

[5] `https://github.com/pytorch/fairseq/blob/master/fairseq/models/transformer.py.`

[6] `https://github.com/microsoft/MASS`

[7] `https://github.com/microsoft/MASS/tree/master/MASS-summarization`

[8] `https://github.com/microsoft/ProphetNet`

Models	**Avg.**	Text Summarization				Question Generation		QA	Dialogue
		CNN/DM	**Gigaword**	**XSUM**	**MSNews**	**SQuAD 1.1**	**MSQG**	**CoQA**	**PesonaChat**
Metrics		R-1/R-2/R-L				R-L/B-4/MTR		F1	B-1/B-2/D-1/D-2
GLGE-Easy									
LSTM	20.0	37.3/15.7/34.4	34.2/16.0/31.8	25.1/6.9/19.9	30.0/14.6/27.7	27.2/3.8/8.9	25.3/3.5/14.1	15.1	42.2/35.9/0.2/0.7
Transformer	21.9	39.5/16.7/36.7	37.1/18.4/34.5	30.5/10.4/24.2	33.0/15.4/30.0	30.7/4.8/10.9	29.3/5.1/16.6	15.7	38.3/33.6/0.2/0.7
MASS$_{\text{base}}$	33.6	42.1/19.5/39.0	38.7/19.7/35.9	39.7/17.2/31.9	39.4/21.0/36.1	49.4/20.1/24.4	38.9/10.2/23.3	65.4	41.0/35.7/1.4/6.9
ProphetNet$_{\text{base}}$	33.8	42.5/19.7/39.5	38.9/19.9/36.0	39.8/17.1/32.0	40.6/21.6/37.0	48.0/19.5/23.9	37.1/9.3/22.7	65.3	46.0/38.4/1.3/7.3
MASS$_{\text{middle}}$	34.3	42.9/19.8/39.8	38.9/20.2/36.2	39.1/16.5/31.4	40.4/21.5/36.8	49.9/21.3/25.2	38.9/9.5/23.5	67.6	46.0/38.2/1.2/6.2
BART$_{\text{large}}$	35.8	44.1/21.2/40.9	38.1/18.4/34.9	45.1/22.2/37.2	43.8/24.0/39.2	50.3/22.0/26.4	38.8/9.2/24.3	68.6	49.9/40.0/1.3/8.0
ProphetNet$_{\text{large}}$	36.5	44.2/21.1/41.3	39.5/20.4/36.6	44.4/21.3/36.4	44.1/24.4/40.2	51.5/22.5/26.0	38.3/9.6/23.3	73.0	46.7/39.0/1.3/7.5
GLGE-Medium									
LSTM	18.1	35.3/14.1/32.8	33.3/15.2/31.1	21.5/4.6/17.1	27.0/12.1/24.9	26.6/3.5/8.2	18.6/1.7/9.5	12.9	41.3/35.3/0.1/0.5
Transformer	19.5	35.0/11.0/32.4	36.7/18.1/34.1	27.5/8.3/21.8	26.8/9.7/24.3	28.3/4.1/9.8	27.0/4.2/15.0	14.2	37.7/29.6/0.2/0.7
MASS$_{\text{base}}$	33.0	41.2/18.8/38.2	37.9/19.1/35.2	37.4/14.9/29.8	38.9/20.5/35.6	48.9/20.0/24.3	38.2/9.5/22.8	65.0	42.8/36.7/1.3/6.2
ProphetNet$_{\text{base}}$	32.6	41.6/19.2/38.7	38.6/19.6/35.7	37.8/15.3/30.4	39.0/20.4/35.7	46.4/17.9/22.5	37.0/8.7/22.3	62.5	45.4/37.7/1.4/7.3
MASS$_{\text{middle}}$	33.6	41.5/19.0/38.5	38.3/19.1/35.4	38.4/15.8/30.7	39.6/20.9/36.0	49.3/20.4/24.4	38.3/9.9/22.7	67.2	44.0/37.3/1.3/6.1
BART$_{\text{large}}$	35.3	42.8/20.1/39.1	38.0/18.3/34.7	43.1/19.5/34.1	43.4/23.6/38.9	49.7/21.6/25.9	38.4/9.5/24.0	69.4	50.4/39.1/1.2/7.4
ProphetNet$_{\text{large}}$	35.5	43.1/20.3/40.1	39.1/19.8/36.1	41.8/18.7/33.8	43.3/23.5/39.4	50.4/21.9/25.8	39.3/10.0/23.7	72.3	42.0/36.4/1.4/7.8
GLGE-Hard									
LSTM	12.6	26.2/6.8/24.2	26.3/9.2/24.6	17.8/2.4/14.3	8.2/0.9/7.6	27.3/1.0/6.7	12.5/0.4/5.0	10.3	36.8/28.7/0.1/0.4
Transformer	14.4	28.3/6.2/25.8	28.6/10.8/26.5	23.0/5.3/18.3	18.0/3.5/16.2	25.9/1.1/7.0	17.0/1.3/8.2	9.9	30.0/29.7/0.1/0.2
MASS$_{\text{base}}$	28.2	40.4/18.0/37.3	32.2/13.6/29.5	33.7/11.6/26.7	35.4/17.0/32.4	42.8/13.4/19.0	34.1/7.5/18.6	50.2	40.1/34.9/1.6/7.8
ProphetNet$_{\text{base}}$	28.0	40.9/18.4/37.7	32.0/13.5/29.5	34.2/11.6/26.8	35.2/17.0/32.1	41.6/13.4/18.9	32.3/7.2/18.0	48.5	41.6/35.5/1.6/8.3
MASS$_{\text{middle}}$	29.1	41.1/18.5/38.0	32.2/13.5/29.9	34.9/12.5/27.6	36.6/18.0/33.4	45.1/16.0/21.3	34.3/8.0/19.0	51.2	41.4/35.4/1.5/7.6
BART$_{\text{large}}$	31.0	41.7/19.1/37.9	33.0/13.6/30.0	39.7/16.1/30.9	40.8/20.8/36.4	45.9/18.1/23.7	35.1/8.5/20.7	53.5	48.3/37.3/1.3/7.2
ProphetNet$_{\text{large}}$	30.5	41.2/18.7/38.0	32.4/13.7/29.9	39.4/16.1/31.6	40.3/20.5/36.4	46.4/17.0/22.1	34.0/8.2/19.0	54.1	40.5/35.2/1.8/9.2

Table 2: Overall results of baselines on all GLGE tasks. We use the color to highlight the overall score. R-1: ROUGE-1. R-2: ROUGE-2. R-L: ROUGE-L. B-4: BLUE-4. MTR: METEOR. D-1: Distinct-1. D-2: Distinct-2. Note that as discussed in § 2.3, the values of Distinct-1 and Distinct-2 are multiplied by 100.

GLGE, we fine-tune ProphetNet$_{\text{large}}$ with the same hyper-parameters used in their source code for a maximum of 10 epochs.

BART$_{\text{large}}$ (Lewis et al., 2020). The hyper-parameters and implementation of BART$_{\text{large}}$ are based on the source code[9]. BART$_{\text{large}}$ contains a 12-layer encoder and 12-layer decoder with 1024 embedding/hidden size and 4096 feed-forward filter size. The pretraining of BART$_{\text{large}}$ uses the same pretraining data as Liu et al. (2019), consisting of 160GB of news, books, stories, and web text. For each task in GLGE, we fine-tune BART$_{\text{large}}$ with the same hyper-parameters used in their source code[10] for a maximum of 20000 iterations.

Except BART, all the baselines adopt BERT-uncased tokenizer. We fine-tune all baselines on each individual task with 4×16GB NVIDIA V100 GPUs. We evaluate the best model checkpoint based on the loss on the development set. During inference, we use beam search (Och and Ney, 2004) with beam size 4 or 5 and remove the duplicated trigrams in beam search (Fan et al., 2018) to obtain the generated results.

3.3 Results and Analysis

Overall results. The main results are presented in Table 2. From the overall scores (highlighted in color), we can observe the fairly consistent gains moving from LSTM to Transformer, and then to pretrained base models and pretrained large models, such as ProphetNet$_{\text{base}}$ and ProphetNet$_{\text{large}}$. The performance gap between the pretrained model and non-pretrained model is obvious. The difference in terms of overall score is about absolute 15% on the three levels GLGE benchmarks (GLGE-Easy, GLGE-Medium, and GLGE-Hard). As expected, the pretrained large models (ProphetNet$_{\text{large}}$ and BART$_{\text{large}}$) achieve the best overall scores.

From the results of each model on the three levels GLGE benchmarks, we can see that each model has a significant drop in performance from GLGE-Easy to GLGE-Medium and GLGE-Hard benchmarks. For both the non-pretrained model and pretrained models, there is a nearly 2% drop in terms of overall score from GLGE-Easy to GLGE-Medium, and about 4%-8% drop from GLGE-Easy to GLGE-Hard. These results illustrate the diversified difficulty of GLGE. We recommend that researchers choose a GLGE benchmark with moder-

---

[9] `https://github.com/pytorch/fairseq/tree/master/examples/bart`

[10] `https://github.com/pytorch/fairseq/blob/master/examples/bart/README.summarization.md`

		Text Summarization				Question Generation		QA	Dialogue
Models	**Avg.**	**CNN/DM**	**Gigaword**	**XSUM**	**MSNews**	**SQuAD 1.1**	**MSQG**	**CoQA**	**PesonaChat**
Metrics		R-1/R-2/R-L				R-L/B-4/MTR		F1	B-1/B-2/D-1/D-2
GLGE-Medium + Low-frequency Strategy									
LSTM	16.4	35.4/13.9/32.7	31.3/13.2/29.3	20.8/4.0/16.5	25.3/10.4/23.3	24.8/1.2/6.8	15.3/1.1/7.1	10.5	34.8/31.6/0.2/0.6
Transformer	18.1	36.8/13.7/34.0	33.2/14.6/30.9	25.9/7.3/20.7	25.2/7.6/22.7	26.4/1.9/7.8	22.8/2.7/11.9	10.8	39.6/34.3/0.1/0.2
MASS$_{\text{base}}$	29.9	41.3/18.8/38.3	34.4/15.3/31.7	36.1/13.6/28.5	36.4/17.9/33.1	45.3/16.0/21.4	34.6/8.5/19.9	54.6	40.3/34.6/1.5/7.5
ProphetNet$_{\text{base}}$	29.9	41.7/19.1/38.7	35.1/16.1/32.4	36.5/14.0/28.9	37.4/18.6/34.0	43.7/15.1/20.5	33.8/7.6/20.0	53.1	42.3/35.9/1.4/7.6
MASS$_{\text{middle}}$	30.8	41.5/18.9/38.4	35.1/16.0/32.5	37.0/14.4/29.4	37.9/19.1/34.6	46.3/17.0/22.1	35.5/8.9/20.5	56.6	41.0/35.5/1.6/8.0
BART$_{\text{large}}$	32.0	42.2/19.6/38.4	35.7/16.1/32.6	41.5/17.9/32.3	41.5/21.4/37.0	46.5/18.1/24.5	35.9/8.4/21.7	54.6	49.1/38.1/1.3/8.4
ProphetNet$_{\text{large}}$	32.3	42.7/20.0/39.5	35.3/16.2/32.6	40.6/17.4/32.6	40.9/21.4/37.1	47.8/19.2/23.9	36.4/9.4/20.9	57.6	44.4/36.9/1.4/8.3

Table 3: Overall results of baselines across the tasks of GLGE-Medium + Low-frequency strategy. We use the color to highlight the overall score. R-1: ROUGE-1. R-2: ROUGE-2. R-L: ROUGE-L. B-4: BLUE-4. MTR: METEOR. D-1: Distinct-1. D-2: Distinct-2.

ate difficulty based on the model size of the pretrained model and the scale size of the pretrained corpus. At the same time, researchers can also perform more comprehensive evaluation of model performance on GLGE benchmarks at all difficulty levels.

Low-frequency strategy analysis. To further verify the effectiveness of the low-frequency strategy as described in § 2.4. We build the GLGE-Medium + low-frequency benchmark and evaluate the models on it. Both GLGE-Medium and GLGE-Medium + low-frequency retain 50% of the training samples in the GLGE-Easy training set. The only difference between them is that GLGE-Medium uses random sampling to retain 50% training samples, while GLGE-Medium + low-frequency uses the low-frequency strategy as used in GLGE-Hard to select 50% training samples. We use the same baselines with the same settings in § 3.2 to compare the model performance on GLGE-Medium and GLGE-Medium + low-frequency. The results are shown in Table 3. We can see that after the introduction of the low-frequency strategy, the performance of the models has dropped significantly. These results demonstrate that the low-frequency strategy can effectively improve the difficulty of the benchmark.

Output diversity analysis. We further compare the output diversity of each model on all the tasks of GLGE-Easy. We report the mean of the Distinct bigram (Distinct-2) (Li et al., 2016) ratios of the generated samples to the golden references. Note that if the bigram diversity of the generated samples is close to that of the real samples, the distinct bigram ratio is close to 1. The results are shown in Figure 1. In general, output bigram diversity of the pretrained model is higher than non-pretrained models. For the tasks of CNN/DailyMail (CNN/DM), Gigaword, and MSNews, the bigram diversity of the generated samples are close to that of the real samples. However, for the tasks of XSUM, SQuAD 1.1, MSQG, CoQA, and PersonaChat, the bigram diversity of the non-pretrained model is significantly lower than that of the pretrained model. For these tasks, the non-pretrained model tends to generate universal responses (Li et al., 2016) or outputs. Moreover, there is still a huge gap between the bigram diversity of pretrained models and real samples (golden) on the task of XSUM, MSQG, and PersonaChat. Obviously, there exists great room for future improvement of the pretrained models in terms of output diversity.

4 Related Works

Benchmarks Recently, the development of general natural language understanding (NLU) evaluation benchmarks has helped drive the progress of pretraining and transfer learning in NLP. Conneau and Kiela (2018) propose a toolkit, SentEval, for evaluating the quality of universal sentence representations. DecaNLP (McCann et al., 2018) casts ten diversified NLP tasks as a general question-answering format for evaluation. Wang et al. (2019b) propose a widely-used multi-task benchmark, GLUE, for NLU in the English language. There are nine NLU tasks in GLUE, including two single-sentence tasks, three similarity and paraphrase tasks, and four natural language inference tasks. After that, SuperGLUE (Wang et al., 2019a) is proposed as a harder counterpart of GLUE. Besides sentence- and sentence-pair classification tasks used in GLUE, SuperGLUE extends the task formats by introducing coreference resolution and question answering tasks. More recently, A new NLU benchmark called DialoGLUE (Mehri et al., 2020) for task-oriented dialogue is proposed. It consists of seven task-oriented dialogue datasets

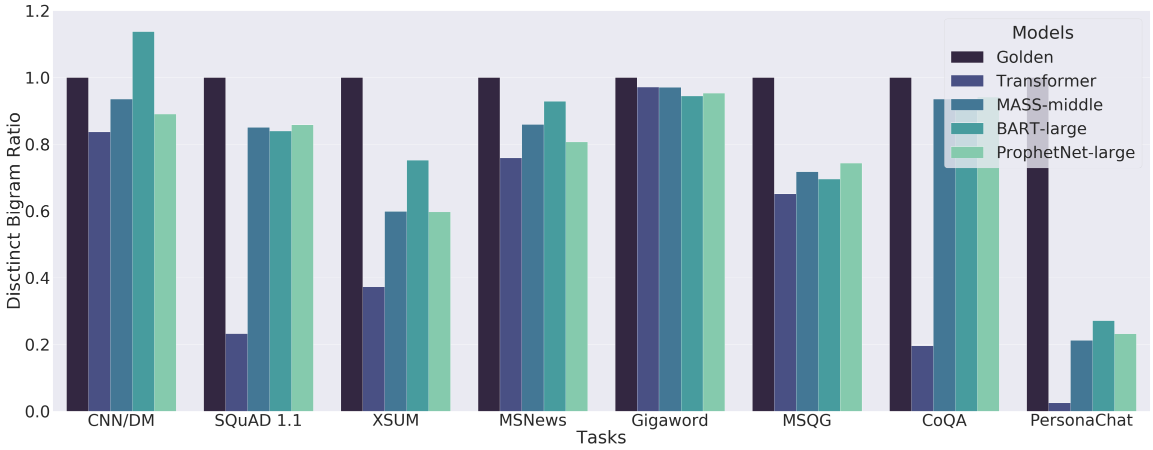

Figure 1: Distinct bigram ratios of the generated samples to the golden references on GLGE-Easy.

covering four kinds of NLU tasks: intent prediction, slot tagging, semantic parsing, and dialogue state tracking.

In addition to the English NLU evaluation benchmark, there has been an increasing amount of new benchmarks in other languages. For example, CLUE (Xu et al., 2020) is a Chinese NLU benchmark that consists of eight diverse Chinese NLU tasks, including single-sentence, sentence-pair, and machine reading comprehension tasks. FLUE (Le et al., 2020) is proposed for the French language, which is a French NLU benchmark that includes several NLU tasks, such as text classification, paraphrasing, language inference, parsing, POS tagging, and word sense disambiguation. IndoNLU (Wilie et al., 2020) is a new benchmark for evaluating Indonesian language understanding, which introduces twelve tasks, ranging from single sentence classification to pair-sentences sequence labeling. Furthermore, the multilingual multi-task benchmarks are proposed for cross-lingual evaluating. Hu et al. (2020) introduce XTREME benchmark which is a multi-task benchmark for evaluating the cross-lingual generalization capabilities of multilingual representations across forty languages and nine tasks. Almost at the same time, XGLUE (Liang et al., 2020) is proposed which is a new multilingual multi-task benchmark for cross-lingual pretraining, understanding, and generation. There are eleven cross-lingual tasks including nine NLU tasks and two NLG tasks in XGLUE, each task provides labeled data in multiple languages.

The above benchmarks mostly focus on NLU which provides a range of language understanding tasks. However, to our best knowledge, there is no benchmark designed specifically for general NLG evaluation. To fill this gap, we introduce GLGE, a new multi-task benchmark for evaluating the generalization capabilities of NLG across eight language generation tasks.

Pretrained NLG Models In recent years, pretrained language models (Devlin et al., 2019; Raffel et al., 2020; Yang et al., 2019; Liu et al., 2019; Alberti et al., 2019; Brown et al., 2020; Clark et al., 2020) have achieved state-of-the-art results in several NLU benchmarks. Besides, more and more pretraining based models which are designed for NLG tasks are proposed. Rothe et al. (2020) adopt the Transformer-based sequence-to-sequence (seq2seq) model and leverage the checkpoints of the pretrained NLU models for sequence generation tasks. MASS (Song et al., 2019) pretrains the seq2seq model by dropping a continuous token span to corrupt the text and learns to reconstruct it. Raffel et al. (2020) investigate several model structures and pretraining tasks, and further propose a unified text-to-text transformer called T5. Similarly, BART (Lewis et al., 2020) adopts the encoder-decoder structure and is pretrained with randomly sentence order reconstruction and text in-filling tasks.

More recently, Qi et al. (2020) propose, ProphetNet, which introduces the future n-gram prediction mechanism for language generation. ENRINE-GEN (Xiao et al., 2020) introduces the infilling generation mechanism, noise-aware generation, and span-by-span generation task for NLG model pretraining. In addition to general NLG tasks, some task-specific pretrained NLG models are proposed. For dialogue and conversation, a dialogue generative pretrained transformer called DialoGPT (Zhang et al., 2020c) is proposed for conversational response generation, which is pretrained on a large-scale conversation-like exchange corpus. Furthermore, PLATO (Bao et al., 2020) is a dialogue generation pretraining framework for chit-chat, knowledge grounded dialogues, and conversational question answering. PLATO introduces discrete latent variables to tackle the one-to-many mapping problem in response generation. For text summarization, Zhang et al. (2020a) propose PEGASUS, which design the pretraining objectives called gap sentence generation tailored for abstractive text summarization.

5 Conclusion

To facilitate the development, evaluation, and comparison of new NLG models, we introduce GLGE, a multi-task evaluation benchmark for NLG with three difficulty levels. To the best of our knowledge, GLGE is the first comprehensive NLG evaluation benchmark. We evaluate several baselines on GLGE and analyze their results. The GLGE benchmark is hosted publicly and we invite the research community to submit to the leaderboard.

In future work, we will try to introduce other automatic evaluation metrics, such as BERTscore (Zhang et al., 2020b) and BLEURT (Sellam et al., 2020). Besides, we will compare the correlation between these metrics and human judgment.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1406702, in part by NFSC under Grant 61625204, 61836006, and the Science and Technology Major Project of Sichuan province under Grant 2020YFG0478. We thank STC-A NLP, Bing Answers, Bing Ads, Bing Relevance and Microsoft News for providing the datasets.

References

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96, Online. Association for Computational Linguistics.

Manik Bhandari, Pranav Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. 2020. Re-evaluating evaluation in text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9347–9359.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pretraining text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. 2020. FlauBERT: Unsupervised language model pre-training for French. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France. European Language Resources Association.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. XGLUE: A new benchmark datasetfor cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming

Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. BLEURT: learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7881–7892. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Bryan Wilie, Karissa Vincentio, Genta Indra Winata, Samuel Cahyawijaya, Xiaohong Li, Zhi Yuan Lim, Sidik Soleman, Rahmad Mahendra, Pascale Fung, Syafri Bahar, and Ayu Purwarianti. 2020. IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 843–857, Suzhou, China. Association for Computational Linguistics.

Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gen: An enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3997–4003.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020c. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

CNN/DailyMail	**Article:** *The only thing crazier than a guy in snowbound Massachusetts boxing up the powdery white stuff and offering it for sale online? People are actually buying it. For $ 89, self-styled entrepreneur Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. [...], a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born [...]* **Target:** A man in suburban Boston is selling snow online to customers in warmer states. For $ 89, he will ship 6 pounds of snow in an insulated Styrofoam box.
Gigaword	**Passage:** *U.S. business leaders lashed out Wednesday at legislation that would penalize companies for employing illegal immigrants.* **Target:** U.S. business attacks tough immigration law.
XSUM	**Article:** *Burberry reported pre-tax profits of £166m for the year to March. [...], Sales rose 7% to £1.28bn, with the company recording double-digit sales growth in Europe and Asia Pacific. Adjusted profit rose 23% to £215m, taking into account one-off items and a favourable exchange rate. Stores in London in particular benefited from favourable currency movements and increased tourism. [...], Burberry shares were up 7.6% at 659 pence in afternoon trading.* **Target:** Luxury fashion designer Burberry has returned to profit after opening new stores and spending more on online marketing.
MSNews	**Article:** *Los Angeles : Actor Chadwick Boseman, who played Black icons Jackie Robinson and James Brown before finding fame as the regal Black Panther in the Marvel cinematic universe, died Friday of cancer, his representative said. He was 43. Boseman died at his home in the Los Angeles area with his wife and family by his side, his publicist Nicki Fioravante told The Associated Press. [...]* **Target:** Black Panther actor Chadwick Boseman dies of cancer at 43.
SQuAD 1.1	**Passage:** *Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, [...].* **Answer:** *Santa Clara, California* **Target:** Where did Super Bowl 50 take place?
MSQG	**Passage:** *On March 28, 1830, Congress passed the Indian Removal Act, beginning the forced relocation of thousands of Native Americans in what became known as the Trail of Tears. Not all members of Congress supported the Indian Removal Act.* **Highlight Span:** *Indian Removal Act* **Target:** What act was passed to relocate the native Americans?
CoQA	**Passage:** *John was in the third grade, and nine years old. Every day he had to walk home from school. There were some kids in his class who were mean to him, and during the winter they would throw snowballs at him. [...], John thought it was a good deal, and ended up being much better at math.* **Conversation History:** *Q1: Who is in third grade? A1: John. Q2: How old is he? A2: Nine.* **Question:** *What did kids do to him?* **Target:** Throw snowballs at him.
PersonaChat	**Persona Profile description text:** *My wife left me and took my children. I don't believe in god. I'm overweight and unhappy. I work at a nursing home. I spend most of my time on Facebook when I'm not working.* **Conversation History:** *Q1: I got a big house with 7 rooms. R1: Nice, majority of my time I am on Facebook one time. Q2: I saw a man fly to the moon.* **Target:** Like on TV? I'm obese and unhappy.

Table 4: Development-set examples from the tasks in GLGE. **Bold** text denotes part of the example format for each task. Text in *italics* is part of the source input text. Underlined text is the target output text. *[...]* denotes the omitted texts.

AMBERT: A Pre-trained Language Model with Multi-Grained Tokenization

Xinsong Zhang, Pengshuai Li, Hang Li
ByteDance AI Lab
{zhangxinsong.0320, lipengshuai.1204, lihang.lh}@bytedance.com

Abstract

Pre-trained language models such as BERT have exhibited remarkable performances in many tasks in natural language understanding (NLU). The tokens in the models are usually fine-grained in the sense that for languages like English they are words or sub-words and for languages like Chinese they are characters. In English, for example, there are multi-word expressions which form natural lexical units and thus the use of coarse-grained tokenization also appears to be reasonable. In fact, both fine-grained and coarse-grained tokenizations have advantages and disadvantages for learning of pre-trained language models. In this paper, we propose a novel pre-trained language model, referred to as AMBERT (A Multi-grained BERT), on the basis of both fine-grained and coarse-grained tokenizations. For English, AMBERT takes both the sequence of words (fine-grained tokens) and the sequence of phrases (coarse-grained tokens) as input after tokenization, employs one encoder for processing the sequence of words and the other encoder for processing the sequence of the phrases, utilizes shared parameters between the two encoders, and finally creates a sequence of contextualized representations of the words and a sequence of contextualized representations of the phrases. Experiments have been conducted on benchmark datasets for Chinese and English, including CLUE, GLUE, SQuAD and RACE. The results show that AMBERT can outperform BERT in all cases, particularly the improvements are significant for Chinese. We also develop a method to improve the efficiency of AMBERT in inference, which still performs better than BERT with the same computational cost as BERT.

1 Introduction

Pre-trained models such as BERT, RoBERTa, and ALBERT (Devlin et al., 2018; Liu et al., 2019; Lan et al., 2019) have shown great power in natural language understanding (NLU). The Transformer-based language models are first learned from a large corpus in pre-training, and then learned from labeled data of a downstream task in fine-tuning. With Transformer (Vaswani et al., 2017), pre-training technique, and big data, the models can effectively capture the lexical, syntactic, and semantic relations between the tokens in the input text and achieve state-of-the-art performance in many NLU tasks, such as sentiment analysis, text entailment, and machine reading comprehension.

In BERT, for example, pre-training is mainly conducted based on masked language modeling (MLM) in which about 15% of the tokens in the input text are masked with a special token [MASK], and the goal is to reconstruct the original text from the masked tokens. Fine-tuning is separately performed for individual tasks as text classification, text matching, text span detection, etc. Usually, the tokens in the input text are fine-grained; for example, they are words or sub-words in English and characters in Chinese. In principle, the tokens can also be coarse-grained, that is, for example, phrases in English and words in Chinese. There are many multi-word expressions in English such as 'New York' and 'ice cream' and the use of phrases also appears to be reasonable. It is more sensible to use words (including single character words) in Chinese, because they are basic lexical units. In fact, all existing pre-trained language models employ single-grained (usually fine-grained) tokenization.

Previous work indicates that the fine-grained approach and the coarse-grained approach have both pros and cons. The tokens in the fine-grained approach are less complete as lexical units but their representations are easier to learn (because there are less token types and more tokens in training data), while the tokens in the coarse-grained approach are more complete as lexical units but their

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 421–435
August 1–6, 2021.

representations are more difficult to learn (because there are more token types and less tokens in training data). Moreover, for the coarse-grained approach there is no guarantee that tokenization (segmentation) is completely correct. Sometimes ambiguity exists and it would be better to retain all possibilities of tokenization. In contrast, for the fine-grained approach tokenization is carried out at the primitive level and there is no risk of 'incorrect' tokenization.

For example, (Li et al., 2019) observe that fine-grained models consistently outperform coarse-grained models in deep learning for Chinese language processing. They point out that the reason is that low frequency words (coarse-grained tokens) tend to have insufficient training data and tend to be out of vocabulary, and as a result the learned representations are not sufficiently reliable. On the other hand, previous work also demonstrates that masking of coarse-grained tokens in pre-training of language models is helpful (Cui et al., 2019; Joshi et al., 2020). That is, although the model itself is fine-grained, masking on consecutive tokens (phrases in English and words in Chinese) can lead to learning of a more accurate model. In Appendix A, we give examples of attention maps in BERT to further support the assertion.

In this paper, we propose A Multi-grained BERT model (AMBERT), which employs both fine-grained and coarse-grained tokenizations. For English, AMBERT extends BERT by simultaneously constructing representations for both words and phrases in the input text using two encoders. Specifically, AMBERT first conducts tokenization at both word and phrase levels. It then takes the embeddings of words and phrases as input to the two encoders with the shared parameters. Finally it obtains a contextualized representation for the word and a contextualized representation for the phrase at each position. Note that the number of parameters in AMBERT is comparable to that of BERT, because the parameters in the two encoders are shared. There are only additional parameters from multi-grained embeddings. AMBERT can represent the input text at both word-level and phrase-level, to leverage the advantages of the two approaches of tokenization, and create richer representations for the input text at multiple granularity.

AMBERT consists of two encoders and thus its computational cost is roughly doubled compared with BERT. We also develop a method for improving the efficiency of AMBERT in inference, which only uses one of the two encoders. One can choose either the fine-grained encoder or the coarse-grained encoder for a specific task using a development dataset.

We conduct extensive experiments to make a comparison between AMBERT and the baselines as well as alternatives to AMBERT, using the benchmark datasets in English and Chinese. The results show that AMBERT significantly outperforms single-grained BERT models with a large margin in both Chinese and English. In English, compared to Google BERT, AMBERT achieves 2.0% higher GLUE score, 2.5% higher RACE score, and 5.1% more SQuAD score. In Chinese, AMBERT improves average score by over 2.7% in CLUE. Furthermore, AMBERT with only one encoder can preform much better than the single-grained BERT models with a similar amount of inference time.

We make the following contributions.

- Study of multi-grained pre-trained language models,
- Proposal of a new pre-trained language model called AMBERT as an extension of BERT,
- Empirical verification of AMBERT on the English and Chinese benchmark datasets GLUE, SQuAD, RACE, and CLUE,
- Proposal of an efficient inference method for AMBERT.

2 Related work

There has been a large amount of work on pre-trained language models. ELMo (Peters et al., 2018) is one of the first pre-trained language models for learning contextualized representations of words in the input text. Leveraging the power of Transformer (Vaswani et al., 2017), GPTs (Radford et al., 2018, 2019) are developed as unidirectional models to make predictions on the input text in an auto-regressive manner, and BERT (Devlin et al., 2018) is developed as a bidirectional model to make predictions on the whole or part of the input text. Masked language modeling (MLM) and next sentence prediction (NSP) are the two tasks in pre-training of BERT. Since the inception of BERT, a number of new models have been proposed to further enhance the performance of it. XLNet (Yang et al., 2019) is a permutation language model which can improve the accuracy of MLM. RoBERTa (Liu et al., 2019) represents a new way of training more reliable BERT with a very large amount of data. ALBERT (Lan et al., 2019) is a light-weight version of BERT, which shares parameters across layers. StructBERT (Wang et al., 2019) incorporates word and sentence structures into BERT to learn better representations of tokens and sentences. ERNIE2.0 (Sun et al., 2020) is a variant of BERT pre-trained on multiple tasks with coarse-grained tokens masked. ELECTRA (Clark et al., 2020) has a GAN-style architecture for efficiently utilizing all tokens in pre-training.

It has been found that the use of coarse-grained tokens is beneficial for pre-trained language models. (Devlin et al., 2018) point out that 'whole word masking' is effective for training of BERT. It is also observed that whole word masking is useful for building a Chinese BERT (Cui et al., 2019). In ERNIE (Sun et al., 2019b), entity level masking is employed as a strategy for pre-training and proved to be effective for language understanding tasks (see also (Zhang et al., 2019)). In SpanBERT (Joshi et al., 2020), text spans are masked in pre-training and the learned model can substantially enhance the accuracies of span selection tasks. It is indicated that word segmentation is especially important for Chinese and a BERT-based Chinese text encoder is proposed with n-gram representations (Diao et al., 2019). All existing work focuses on the use of single-grained tokens in learning and utilization of pre-trained language models. In this work, we propose a general technique of exploiting multi-grained tokens for pre-trained language models and apply it to BERT.

3 Our Method: AMBERT

In this section, we present the model, pre-training, and fine-tuning of AMBERT. We also present a discussion on alternatives to AMBERT.

3.1 Model

Figure 1 gives an overview of AMBERT. AMBERT takes a text as input. Tokenization is conducted on the input text to obtain a sequence of fine-grained tokens and a sequence of coarse-grained tokens. AMBERT has two encoders, one for processing the fine-grained token sequence and the other for processing the coarse-grained token sequence. Each of the encoders has exactly the same architecture as that of BERT (Devlin et al., 2018). The two encoders share the same parameters at each corresponding layer, except that each has its own token embedding parameters. The fine-grained encoder generates contextualized representations from the sequence of fine-grained tokens through its layers. In parallel, the coarse-grained encoder generates contextualized representations from the sequence of coarse-grained tokens through its layers. AMBERT outputs a sequence of contextualized representations for the fine-grained tokens and a sequence of contextualized representations for the coarse-grained tokens.

AMBERT is expressive in that it learns and utilizes contextualized representations of the input text at both fine-grained and coarse-grained levels. The model retains all possibilities of tokenizations and learns the attention weights (importance) of representations of multi-grained tokens. AMBERT is also efficient through sharing of parameters between the two encoders. The parameters represent the same ways of combining representations, no matter whether representations are those of fine-grained tokens or coarse-grained tokens.

3.2 Pre-Training

Pre-training of AMBERT is mainly conducted on the basis of masked language modeling (MLM), at both fine-grained and coarse-grained levels. Next sentence prediction (NSP) is not essential as indicated in many studies after BERT (Lan et al., 2019; Liu et al., 2019). We only use NSP in our experiments for comparison purposes. Let $\hat{\mathbf{x}}$ denote the sequence of fine-grained tokens with some of them being masked, and $\bar{\mathbf{x}}$ denote the masked fine-grained tokens. Let $\hat{\mathbf{z}}$ denote the sequence of coarse-grained tokens with some of them being masked, and $\bar{\mathbf{z}}$ denote the masked coarse-grained tokens. Pre-training is defined as optimization of the following function,

$$\min_{\theta} \; -\log p_{\theta}(\bar{\mathbf{x}}, \bar{\mathbf{z}}|\hat{\mathbf{x}}, \hat{\mathbf{z}}) \approx$$

$$\min_{\theta} -\sum_{i=1}^{M} m_i \log p_{\theta}(x_i|\hat{\mathbf{x}}) - \sum_{j=1}^{n} n_j \log p_{\theta}(z_j|\hat{\mathbf{z}}),$$

where m_i takes 1 or 0 as values and $m_i = 1$ indicates that fine-grained token x_i is masked, m denotes the total number of fine-grained tokens; n_j takes 1 or 0 as values and $n_j = 1$ indicates that coarse-grained token z_j is masked, n denotes the total number of coarse-grained tokens; and θ denotes parameters.

3.3 Fine-Tuning

In fine-tuning of AMBERT for classification, the fine-grained encoder and coarse-grained encoder

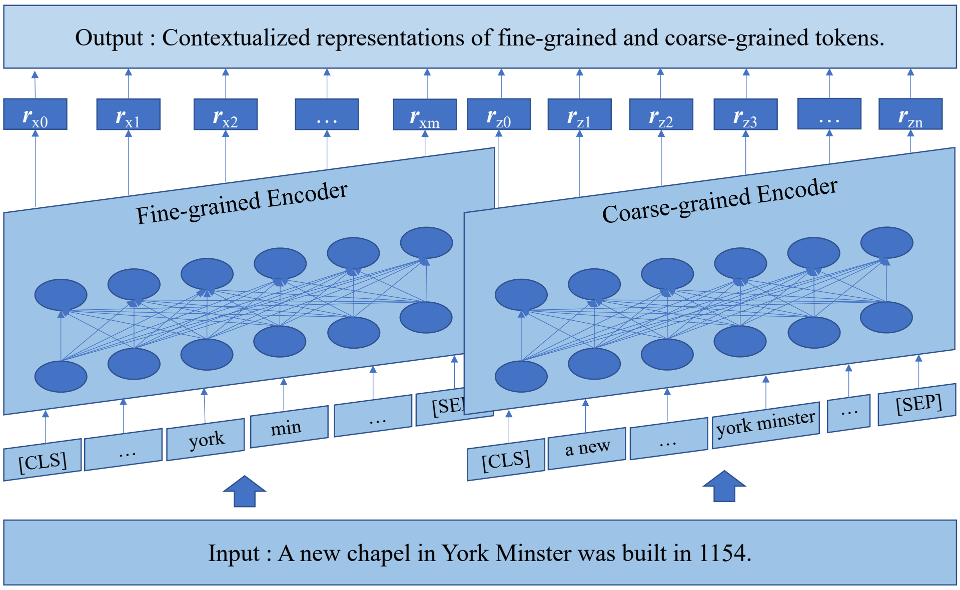

Figure 1: An overview of AMBERT, showing the process of creating multi-grained representations. The input is a sentence in English and output is the overall representation of the sentence. There are two encoders for processing the sequence of fine-grained tokens and the sequence of coarse-grained tokens respectively. The final contextualized representations of fine-grained tokens and coarse-grained tokens are denoted as $\boldsymbol{r}_{x0}, \boldsymbol{r}_{x1}, \cdots, \boldsymbol{r}_{xm}$ and $\boldsymbol{r}_{z0}, \boldsymbol{r}_{z1}, \cdots, \boldsymbol{r}_{zn}$ respectively.

create special [CLS] representations, and both representations are used for classification. Fine-tuning is defined as optimization of the following function, which is a regularized loss of multi-task learning, starting from the pre-trained model,

$$\begin{aligned} &\min_{\theta} \; -\log p_{\theta}(\boldsymbol{y}|\mathbf{x}) \\ &= \min_{\theta} -\log p_{\theta}(\boldsymbol{y}|\boldsymbol{r}_{x0}) - \log p_{\theta}(\boldsymbol{y}|\boldsymbol{r}_{z0}) \\ &- \log p_{\theta}(\boldsymbol{y}|[\boldsymbol{r}_{x0}, \boldsymbol{r}_{z0}]) + \lambda \|\tilde{\boldsymbol{y}}_x - \tilde{\boldsymbol{y}}_z\|_2, \end{aligned}$$

where $\mathbf{x}$ is the input text, $\boldsymbol{y}$ is the classification label, $\boldsymbol{r}_{x0}$ and $\boldsymbol{r}_{z0}$ are the [CLS] representations of fine-grained encoder and coarse-grained encoder, $[\boldsymbol{a}, \boldsymbol{b}]$ denotes concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, λ is a regularization coefficient, and $\|\|_2$ denotes L2 norm. The last term is based on agreement regularization (Brantley et al., 2019), which forces agreement between the predictions ($\tilde{\boldsymbol{y}}_x$ and $\tilde{\boldsymbol{y}}_z$).

Similarly, fine-tuning of AMBERT for span detection can be carried out, in which the representations of fine-grained tokens are concatenated with the representations of corresponding coarse-grained tokens. The concatenated representations are then utilized in the task.

3.4 Inference

We propose two ways of using AMBERT in inference. One is to utilize the AMBERT itself and the other to utilize only one encoder of AMBERT. The former performs better but needs more computation and the latter performs slightly worse but only needs computation comparable to BERT. One can choose either drop the fine-grained encoder or the coarse-grained encoder in AMBERT through evaluation using a development dataset, which makes the computational cost close to that of BERT.

3.5 Alternatives

We can consider two alternatives to AMBERT, which also rely on multi-grained tokenization. We refer to them as AMBERT-Combo and AMBERT-Hybrid and make comparisons of them with AMBERT in our experiments.

AMBERT-Combo has two individual encoders, an encoder (BERT) working on the fine-grained token sequence and the other encoder (BERT) working on the coarse-grained token sequence, without parameter sharing between them. In learning and inference AMBERT-Combo simply combines the output layers of the two encoders. Its fine-tuning is similar to that of AMBERT.

AMBERT-Hybrid has only one encoder (BERT) working on both the fine-grained token sequence and the coarse-grained token sequence. It creates representations on the concatenation of two sequences and lets the representations of the two

Table 1: Performance on classification tasks in CLUE in terms of accuracy (%). The numbers in boldface denote the best results of tasks. Average accuracies of models are also given. Numbers of parameters (param) and time complexities (cmplx) of models are also shown, where l, n, and d denote layer number, sequence length, and hidden representation size respectively. The tasks with mark † are those with data augmentation.

Model	Param.	Cmplx.	Avg.	TNEWS†	IFLYTEK	CLUEWSC2020†	AFQMC	CSL†	CMNLI
Google BERT	108M	$O(ln^2d)$	72.53	66.99	**60.29**	71.03	73.70	83.50	79.69
Our BERT (char)	108M	$O(ln^2d)$	71.90	67.48	57.50	70.69	71.80	83.83	80.08
Our BERT (word)	165M	$O(ln^2d)$	73.72	68.20	59.96	75.52	73.48	85.17	79.97
AMBERT-Combo	273M	$O(2ln^2d)$	73.61	**69.60**	58.73	71.03	**75.63**	85.07	81.58
AMBERT-Hybrid	176M	$O(4ln^2d)$	73.80	69.04	56.42	76.21	74.41	85.60	81.10
AMBERT	176M	$O(2ln^2d)$	**74.67**	68.58	59.73	**78.28**	73.87	**85.70**	**81.87**

Table 2: Performances on MRC tasks in CLUE in terms of F1, EM (Exact Match) and accuracy. The numbers in boldface denote the best results of tasks. Average scores of models are also given.

Model	Avg.	CMRC2018			ChID		C^3	
		DEV(F1,EM)		TEST(EM)	DEV(Acc.)	TEST(Acc.)	DEV(Acc.)	TEST(Acc.)
Google BERT	73.76	85.48	64.77	71.60	82.20	82.04	65.70	64.50
Our BERT (char)	74.46	85.64	65.45	71.50	83.44	83.12	66.43	65.67
Our BERT (word)	65.77	81.87	41.69	41.30	80.89	80.93	66.72	66.96
AMBERT-Combo	75.26	86.12	65.11	72.00	84.53	84.64	67.74	66.70
AMBERT-Hybrid	75.53	86.71	68.16	72.45	83.37	82.85	67.45	67.75
AMBERT	**77.47**	**87.29**	**68.78**	**73.25**	**87.20**	**86.62**	**69.52**	**69.63**

sequences interact with each other at each layer. Its pre-training is formalized in the following function,

$$
\begin{aligned}
&\min_{\theta} \; -\log p_\theta(\bar{\mathbf{x}}, \bar{\mathbf{z}} | \hat{\mathbf{x}}, \hat{\mathbf{z}}) \\
&\approx \min_{\theta} - \sum_{i=1}^{m} m_i \log p_\theta(x_i | \hat{\mathbf{x}}, \hat{\mathbf{z}}) \\
&- \sum_{j=1}^{n} n_j \log p_\theta(z_j | \hat{\mathbf{x}}, \hat{\mathbf{z}}),
\end{aligned}
$$

where the notations are the same as in (1). Its fine-tuning is the same as that of BERT.

4 Experiments

We make comparisons between AMBERT and the baselines including fine-grained BERT and coarse-grained BERT, as well as the alternatives including AMBERT-Combo and AMBERT-Hybrid, using benchmark datasets in both Chinese and English. The experiments on the alternatives can also be seen as ablation study on AMBERT. The ablation studies for the regularization term λ are given in the Appendix E.

4.1 Data for Pre-Training

For Chinese, we use a corpus consisting of 25 million documents (57G uncompressed text) from Jinri Toutiao[1]. Note that there is no common corpus for training of Chinese BERT. For English, we use a corpus of 13.9 million documents (47G uncompressed text) from Wikipedia and OpenWebText (Gokaslan and Cohen, 2019)[2].

The characters in the Chinese texts are naturally taken as fine-grained tokens. We conduct word segmentation on the texts and treat the words as coarse-grained tokens. We employ a word segmentation tool based on a n-gram model. Both tokenizations exploit WordPiece embeddings (Wu et al., 2016). There are 21,128 characters and 72,635 words in the vocabulary of Chinese.

The words in the English texts are naturally taken as fine-grained tokens. We perform coarse-grained tokenization on the English texts in the following way. First, we calculate the n-grams in the Wikipedia documents using KenLM (Heafield, 2011). We next build a phrase-level dictionary consisting of phrases whose frequencies are sufficiently high and whose last words highly depend on their previous words. We then employ a left-to-right search algorithm to perform phrase-level tokenization on the texts. There are 30,522 words and 77,645 phrases in the vocabulary of English.

4.2 Experimental setup

We make use of the same parameter settings for the AMBERT and BERT models. All models in this paper are 'base-models' having 12 layers of encoder. It is too computationally expensive for

---

[1] Jinri Toutiao is a popular news app. in China.

[2] Unfortunately, BookCorpus, one of the two corpora in the original paper for English BERT, is no longer publicly available.

Table 3: State-of-the-art results of Chinese base models in CLUE.

Model	Params	Avg.	TNEWS†	IFLYTEK	WSC.†	AFQMC	CSL†	CMNLI	CMRC.	ChID	C^3
Google BERT	108M	72.59	66.99	60.29	71.03	73.70	83.50	79.69	71.60	82.04	64.50
XLNet-mid	200M	73.00	66.28	57.85	78.28	70.50	84.70	81.25	66.95	83.47	67.68
ALBERT-xlarge	60M	73.05	66.00	59.50	69.31	69.96	84.40	81.13	**76.30**	80.57	**70.32**
ERNIE	108M	74.20	68.15	58.96	**80.00**	73.83	85.50	80.29	74.70	82.28	64.10
RoBERTa	108M	74.38	67.63	**60.31**	76.90	**74.04**	84.70	80.51	75.20	83.62	66.50
AMBERT	176M	**75.28**	**68.58**	59.73	78.28	73.87	**85.70**	**81.87**	73.25	**86.62**	69.63

Table 4: Performances on the tasks in GLUE. Average score over all the tasks is slightly different from the official GLUE score, since we exclude WNLI. CoLA uses Matthew's Corr. MRPC and QQP use both F1 and accuracy scores. STS-B computes Pearson-Spearman Corr. Accuracy scores are reported for the other tasks. Results of MNLI include MNLI-m and MNLI-mm. The other settings are the same as Table 1.

Model	Param	Cmplx	Avg.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE
Google BERT	110M	$O(ln^2d)$	80.7	52.1	93.5	88.9/81.9	81.5/85.8	71.2/88.5	84.6/83.4	90.5	66.4
Our BERT (word)	110M	$O(ln^2d)$	81.6	53.7	93.8	88.8/84.8	84.3/86.0	71.6/89.0	85.0/84.5	91.2	66.8
Our BERT (phrase)	170M	$O(ln^2d)$	80.7	54.8	93.8	87.4/82.5	82.9/84.9	70.1/88.8	84.1/83.8	90.6	65.1
AMBERT-Combo	280M	$O(2ln^2d)$	81.8	**57.1**	**94.5**	89.2/84.8	84.4/85.8	71.8/88.6	84.7/84.2	90.4	66.2
AMBERT-Hybrid	194M	$O(4ln^2d)$	81.7	50.9	93.4	89.0/85.2	**84.7/87.6**	71.0/89.2	84.6/84.7	91.2	68.5
AMBERT	194M	$O(2ln^2d)$	**82.7**	54.3	**94.5**	**89.7/86.1**	**84.7**/87.1	**72.5/89.4**	**86.3/85.3**	**91.5**	**70.5**

us to train the models as 'large models' having 24 layers. To retain consistency, the masked spans in the coarse-grained encoder are also masked in the fine-grained encoder. The details of pre-training and fine-tuning are the same as those in the original BERT paper (Devlin et al., 2018), which are given in Appendix C.

4.3 Chinese Tasks

4.3.1 Benchmarks

We use the benchmark datasets, Chinese Language Understanding Evaluation (CLUE) (Xu et al., 2020) for experiments in Chinese. CLUE contains six classification tasks, that are TNEWS, IFLYTEK and CLUEWSC2020, AFQMC, CSL and CMNLI[3], and three Machine Reading Comprehension (MRC) tasks which are CMRC2018, ChID and C^3. The details of all the benchmarks are shown in Appendix B. Data augmentation is also performed for all models in the tasks of TNEWS, CSL and CLUEWSC2020 to achieve better performance (see Appendix D for detailed explanation).

4.3.2 Experimental Results

We compare AMBERT with the BERT baselines, including the BERT model released from Google, referred to as Google BERT, and the BERT model trained by us, referred to as Our BERT, including fine-grained (character) and coarse-grained (word) models. Case study is given in Appendix F.

Table 1 shows the results of the classification tasks. AMBERT improves average scores of the BERT baselines by about 1.0% and also works better than AMBERT-Combo and AMBERT-Hybrid. The results of MRC tasks are shown in Table 2. AMBERT improves average scores of the BERT baselines by over 3.0%. Our BERT (word) performs poorly in CMRC2018. This is probably because the results of word segmentation are not accurate enough for the task. AMBERT-Combo and AMBERT-Hybrid are on average better than single-grained BERT models. AMBERT further outperforms both of them.

We also compare AMBERT with the state-of-the-art models such as RoBERTa and ALBERT in CLUE benchmark. The base models are trained with different datasets and procedures, and thus the comparisons should only be taken as references. Note that the settings of the base models are the same as that of Xu et al. (2020). Table 3 shows the results. The average score of AMBERT is higher than all the other models. We conclude that multi-grained tokenization is very helpful for pre-trained language models and the design of AMBERT is reasonable.

4.4 English Tasks

4.4.1 Benchmarks

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of nine NLU tasks. Following BERT (Devlin et al., 2018), we exclude the task WNLI for the reason that results of different models on this task are undifferentiated. In addition, three MRC tasks are also included, i.e., SQuAD v1.1, SQuAD v2.0,

---

[3] The task is introduced at the CLUE website.

Table 5: Performances on three English MRC tasks. We use EM and F1 to evaluate the performance of text detection, and report accuracies for RACE, on both development set and test set.

Model	Avg.	SQuAD 1.1		SQuAD 2.0				RACE	
		DEV(EM, F1)		DEV(EM, F1)		TEST(EM, F1)		DEV	TEST
Google BERT	74.0	80.8	88.5	70.1	73.5	73.7	76.3	64.5	64.3
Our BERT (word)	76.7	83.8	90.6	76.6	79.6	77.3	80.3	62.4	62.6
Our BERT (phrase)	-	67.4	82.3	55.4	62.6	-	-	66.9	66.1
AMBERT-Combo	77.2	84.0	**90.9**	76.4	79.6	76.6	79.8	66.6	63.7
AMBERT-Hybrid	77.3	83.6	90.3	76.4	79.4	76.7	79.7	67.1	65.1
AMBERT	**78.6**	**84.2**	90.8	**77.6**	**80.6**	**78.6**	**81.4**	**68.9**	**66.8**

Table 6: State-of-the-art results of English base models in GLUE. Each task only reports one score following Clark et al. (2020), and we report the average EM of SQuAD1.1 and SQuAD2.0 on development set. AMBERT‡ is pre-trained with a corpora with size comparable to that of RoBERTa (160G uncompressed text). Scores with $\star$ are reported from the published papers.

Model	Params	Avg.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	SQuAD	RACE
Google BERT	110M	78.7	$52.1^\star$	$93.5^\star$	$84.8^\star$	$85.8^\star$	$89.2^\star$	$84.6^\star$	$90.5^\star$	$66.4^\star$	75.5	$64.3^\star$
XLNet	110M	78.6	47.9	94.3	83.3	84.1	89.2	86.8	91.7	61.9	$79.9^\star$	$66.7^\star$
SpanBERT	110M	79.1	51.2	93.5	87.0	82.9	89.2	85.1	92.7	69.7	81.8	57.4
ELECTRA	110M	81.3	$59.7^\star$	$93.4^\star$	$86.7^\star$	$87.7^\star$	$89.1^\star$	$85.8^\star$	$92.7^\star$	$73.1^\star$	74.8	69.9
ALBERT	12M	80.1	53.2	93.2	87.5	87.2	87.8	85.0	91.2	71.1	78.7	65.8
RoBERTa	135M	82.7	**61.5**	**95.8**	88.7	**88.9**	89.4	**87.4**	**93.1**	**74.0**	78.6	69.9
AMBERT‡	194M	**82.8**	60.0	95.2	**88.9**	88.2	**89.5**	87.2	92.6	72.6	**82.5**	**71.2**

and RACE. The details of English benchmarks can be found in Appendix B.

4.4.2 Experimental Results

We compare AMBERT with the BERT models on the tasks in GLUE. The results of Google BERT are from the original paper (Devlin et al., 2018), and the results of Our BERT are obtained by us. From Table 4 we can see the following trends, 1) Multi-grained models, particularly AMBERT, can achieve better results than single-grained models. 2) Among the multi-grained models, AMBERT performs best with fewer parameters and less computation. Case study is given in Appendix F.

We also make comparison on the MRC tasks. The results of Google BERT are either from the papers (Devlin et al., 2018; Yang et al., 2019) or from our runs with the official code. From Table 5 we make the following conclusions. 1) in SQuAD, AMBERT outperforms Google BERT with a large margin. Our BERT (word) generally performs well and Our BERT (phrase) performs poorly in the span detection tasks. 2) In RACE, AMBERT performs best among all the baselines for both development set and test set. 3) AMBERT is the best multi-grained model.

We compare AMBERT with the state-of-the-art models in both GLUE and MRC benchmarks. The results of baselines, in Table 6, are either reported in published papers or re-implemented by us with HuggingFace's Transformer (Wolf et al., 2019). We use the provided implementation in HuggingFace's Transformer, without additional data augmentation, question-answering module[4] and other tricks. Note that AMBERT outperforms all the models on average without using training techniques such as bigger batches and dynamic masking.

4.5 Enhancement of Inference Speed

We also conduct experiments on the efficient inference method of AMBERT on CLUE/GLUE/SQuAD/RACE. We choose the fine-grained encoder for the span detection tasks (CMRC2018 and SQuAD) because it performs much better in the tasks. We choose the coarse-grained encoder for the other Chinese tasks and the fine-grained encoder for the other English tasks because they perform better on average. All the decisions are made based on the results from the Dev datasets. The detailed results are shown in Table 7. We conclude that, a) for the English tasks, AMBERT with one chosen encoder achieves similar results as AMBERT with two encoders and outperforms the single-grained "Our BERT" models with a large margin; b) for the Chinese tasks, AMBERT with one chosen encoder performs slightly worse than AMBERT but performs much better than the single-grained "Our BERT" models. Therefore, in practice, one can train an AMBERT with two encoders and use only one of the encoders in inference.

---

[4] For that reason, we cannot use the results for SQuAD 2.0 in Clark et al. (2020).

Table 7: Performances on the development sets of CLUE, GLUE, SQuAD and RACE with different ways for inference. CN-Models and EN-Models denote Chinese and English pre-trained models respectively. CoLA uses Matthew's Corr. We report EM of CMRC2018 and average EM of SQuAD1.1 and SQuAD2.0. The other metrics are all accuracies. We report the better results among single-grained models as "Our BERT".

CN-Models	Speedup	Avg.	TNEWS	IFLYTEK	CLUEWSC2020	AFQMC	CSL	CMNLI	CMRC2018	ChID	C^3	-
AMBERT	1.0	75.3	68.1	60.1	81.6	74.7	85.6	82.3	68.8	87.2	69.5	-
AMBERT (one encoder)	2.0x	74.8	68.0	59.5	81.3	74.2	85.5	82.1	67.4	86.6	68.5	-
Our BERT	2.0x	73.4	67.8	58.7	79.0	74.1	84.5	80.8	65.5	83.4	66.7	-
EN-Models	Speedup	Avg.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	SQuAD	RACE
AMBERT	1.0	79.2	61.7	94.3	92.3	55.0	91.2	86.2	91.3	70.2	80.9	68.9
AMBERT (one encoder)	2.0x	79.1	62.2	93.2	92.5	55.0	91.2	86.1	91.4	70.6	80.3	68.0
Our BERT	2.0x	77.5	56.6	92.4	89.7	54.2	90.4	85.1	90.6	69.1	80.2	66.9

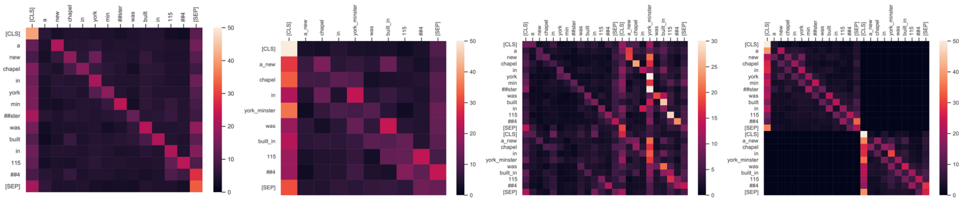

Our BERT (word) Our BERT (phrase) AMBERT-Hybrid AMBERT

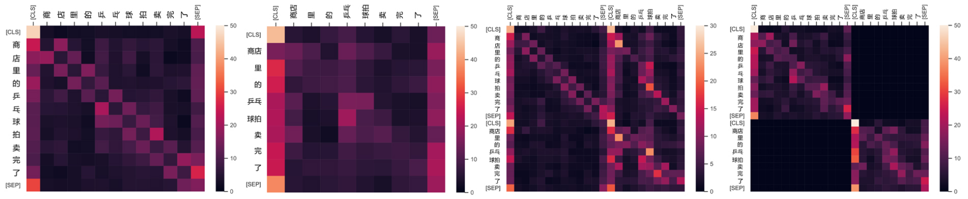

Our BERT (char) Our BERT (word) AMBERT-Hybrid AMBERT

Figure 2: Attention weights of first layers of Our BERT (word/phrase), AMBERT-Hybrid and AMBERT, for English and Chinese sentences.

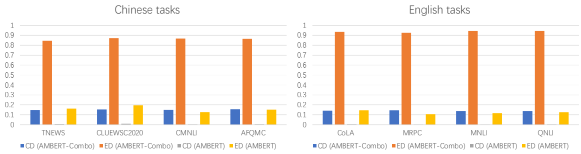

Figure 3: Distances between representations of fine-grained and coarse-grained encoders (representations of [CLS]) in AMBERT-Combo and AMBERT. CD and ED stand for cosine distance (one minus cosine similarity) and normalized Euclidean distance respectively.

4.6 Discussions

We further investigate the reason that AMBERT is superior to AMBERT-Combo. Figure 3 shows the distances between the [CLS] representations of the fine-grained encoder and coarse-grained encoder in AMBERT-Combo and AMBERT after pre-training, in terms of cosine distance (one minus cosine similarity) and normalized Euclidean distance. One can see that the distances in AMBERT-Combo are larger than the distances in AMBERT. We perform the assessment using the data in different tasks and find similar trends. The results indicate that the representations of fine-grained encoder and coarse-grained encoder are closer in AMBERT than in AMBERT-Combo. These are natural consequences of using AMBERT and AMBERT-Combo, whose parameters are respectively shared and unshared across encoders. It implies that the higher performances by AMBERT is due to its parameter sharing, which can learn and represent similar ways of combining tokens no matter whether they are fine-grained or coarse-grained. An intuitive explanation is that the ways of combining representations of fine-grained tokens and the ways of combining representations of coarse-grained tokens "in the same contexts" are exactly the same.

We also examine the reasons that AMBERT works better than AMBERT-Hybrid, while both of them exploit multi-grained tokenization. Figure 2 shows the attention weights of first layers in AMBERT and AMBERT-Hybrid, as well as the single-grained BERT models, after pre-training. In AMBERT-Hybrid, the fine-grained tokens attend more to the corresponding coarse-grained tokens and as a result the attention weights among fine-grained tokens are weakened. In contrast, in AMBERT the attention weights among fine-grained tokens and those among coarse-grained tokens are intact. It appears that attentions among single-grained tokens (fine-grained ones or coarse-grained ones) play important roles in downstream tasks.

To answer the question why the improvements by AMBERT on Chinese are larger than on English in the same pre-training settings, we further make an analysis. We respectively tokenize 10,000 randomly selected Chinese sentences from five tasks in CLUE with our Chinese word tokenizer. The average proportion of words is 51.5%, which indicates that about half of the tokens are fine-grained and half are coarse-grained in Chinese. Similarly, we tokenize 10,000 randomly selected English sentences from five different tasks in GLUE with our English phrase tokenizer. The average proportion of phrases is only 13.1%, which means that there are much less coarse-grained tokens than fine-grained tokens in English. (Please refer to Table 10 in the Appendix for more details of the experiments.) Therefore, we postulate that for Chinese it is necessary for a model to process the language at both fine-grained and coarse-grained levels. AMBERT indeed has the capability.

5 Conclusion

In this paper, we have proposed a novel pre-trained language model called AMBERT, as an extension of BERT. AMBERT employs multi-grained tokenization, that is, it uses both words and phrases in English and both characters and words in Chinese. With multi-grained tokenization, AMBERT learns in parallel the representations of the fine-grained tokens and the coarse-grained tokens using two encoders with shared parameters. We also develop an alternative way of using AMBERT in inference to save computation cost. Experimental results have demonstrated that AMBERT significantly outperforms BERT and other models in NLU tasks in both English and Chinese. AMBERT increases average score of Google BERT by about 2.7% in Chinese benchmark CLUE. AMBERT improves Google BERT by over 3.0% on a variety of tasks in English benchmarks GLUE, SQuAD (1.1 and 2.0), and RACE.

As future work, we plan to study the following issues: 1) to investigate model acceleration methods in learning of AMBERT, such as sparse attention (Child et al., 2019; Kitaev et al., 2020; Zaheer et al., 2020) and synthetic attention (Tay et al., 2020); 2) to apply the technique of AMBERT into other pre-trained language models such as XLNet; 3) to employ AMBERT in other NLU tasks.

Acknowledgments

We thank the teams at ByteDance for providing the Chinese corpus and the Chinese word segmentation tool.

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Kianté Brantley, Wen Sun, and Mikael Henaff. 2019. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2018. A span-extraction dataset for chinese machine reading comprehension. *arXiv preprint arXiv:1810.07366*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shizhe Diao, Jiaxin Bai, Yan Song, Tong Zhang, and Yonggang Wang. 2019. Zen: pre-training chinese text encoder enhanced by n-gram representations. *arXiv preprint arXiv:1911.00720*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is word segmentation necessary for deep learning of chinese representations? *arXiv preprint arXiv:1905.05526*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019a. Probing prior knowledge needed in challenging chinese machine reading comprehension. *arXiv preprint arXiv:1904.09679*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2020. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Liang Xu, Xuanwei Zhang, Lu Li, Hai Hu, Chenjie Cao, Weitang Liu, Junyi Li, Yudong Li, Kai Sun, Yechen Xu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. Chid: A large-scale chinese idiom dataset for cloze test. *arXiv preprint arXiv:1906.01265*.

A Attention maps for single-grained models

We construct fine-grained and coarse-grained BERT models for English and Chinese, and examine the attention maps of the models using the BertViz tool (Vig, 2019). Figure 4 shows the attention maps of the first layer of fine-grained models for several sentences in English and Chinese. One can see that there are tokens that improperly attend to other tokens in the sentences. For example, in the English sentences, the words “drawing”, “new”, and “dog” have high attention weights to “portrait”, “york”, and “food”, respectively, which are not appropriate. For example, in the Chinese sentences, the chars “拍”, “北”, “长” have high attention weights to “卖”, “京”, “市”, respectively, which are also not reasonable. (It is verified that the bottom layers at BERT mainly represent lexical information, the middle layers mainly represent syntactic information, and the top layers mainly represent semantic information (Jawahar et al., 2019).) Ideally a token should only attend to the tokens with which they form a lexical unit at the first layer. This cannot be guaranteed in the fine-grained BERT model, however, because usually a fine-grained token may belong to multiple lexical units (i.e., there is ambiguity).

Figure 5 shows the attention maps of the first layer of coarse-grained models for the same sentences in English and Chinese. In the English sentences, the words are combined into the phrases of “drawing room”, “york minister”, and “dog food”. The attentions are appropriate in the first two sentences, but it is not in the last sentence because of the incorrect tokenization. Similarly, in the Chinese sentences, the high attention weights of words “球拍(bat)” and “京城(capital)” are reasonable, but that of word “市长(mayor)” is not. Note that incorrect tokenization is inevitable.

B Detailed descriptions for the benchmarks

B.1 Chinese Tasks

TNEWS is a text classification task in which titles of news articles in TouTiao are to be classified into 15 classes. IFLYTEK is a task of assigning app descriptions into 119 categories. CLUEWSC2020, standing for the Chinese Winograd Schema Challenge, is a co-reference resolution task. AFQMC is a binary classification task that aims to predict whether two sentences are semantically similar. CSL uses the Chinese Scientific Literature dataset containing abstracts and their keywords of papers and the goal is to identify whether given keywords are the original keywords of a paper. CMNLI is based on translation from MNLI (Williams et al., 2017), which is a large-scale, crowd-sourced entailment classification task. CMRC2018 (Cui et al., 2018) makes use of a span-based dataset for Chinese machine reading comprehension. ChID (Zheng et al., 2019) is a large-scale Chinese IDiom cloze test. C^3 (Sun et al., 2019a) is a free-form multiple-choice machine reading comprehension for Chinese.

B.2 English Tasks

CoLA (Warstadt et al., 2019) contains English acceptability judgments drawn from books and journal articles on linguistic theory. SST-2 (Socher et al., 2013) consists of sentences from movie reviews and human annotations of their sentiment. MRPC (Dolan and Brockett, 2005) is a corpus of sentence pairs automatically extracted from online news sources, and the target is to identify whether a sentence pair is semantically equivalent. STS-B (Cer et al., 2017) is a collection of sentence pairs and the task is to predict similarity scores. QQP is a collection of question pairs and requires models to recognize semantically equivalent ones. MNLI (Williams et al., 2017) is a crowd-sourced collection of sentence pairs with textual entailment annotations. QNLI (Wang et al., 2018) is a question-answering dataset consisting of question-paragraph pairs, where one of the sentences in the paragraph contains the answer to the corresponding question. RTE (Bentivogli et al., 2009) comes from a series of annual textual entailment challenges.

C Hyper-parameters

C.1 Hyper-parameters in pre-training

In pre-training of the AMBERT models, in total 15% of the coarse-grained tokens are masked, which is the same proportion for the BERT models. We adopt the standard hyper-parameters of BERT in pre-training of the models except batch sizes which are tuned to make our fine-grained BERT models comparable to the Google BERT models. Table 8 shows the hyper-parameters in our Chinese AMBERT and English AMBERT. Our BERT models and alternatives of AMBERT (AMBERT-Combo and AMBERT-Hybrid) all use the same

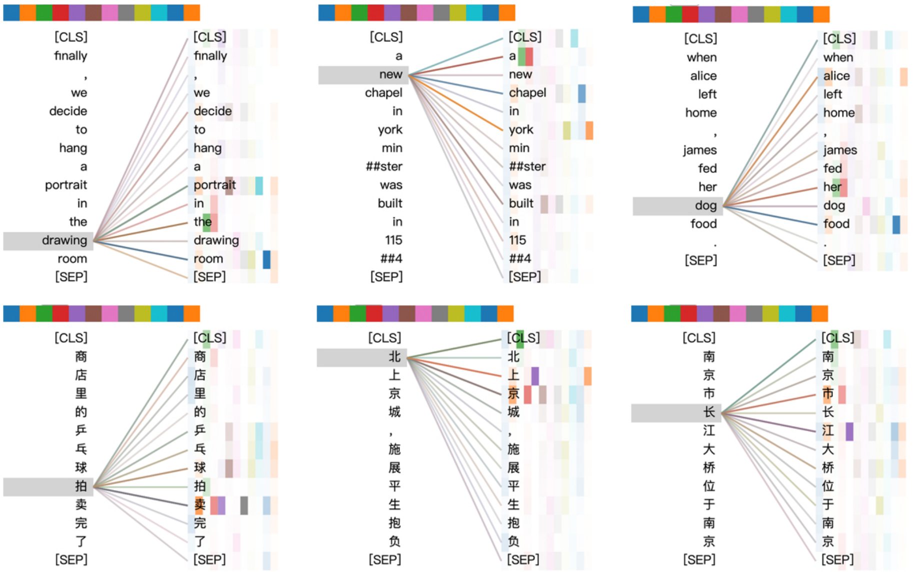

Figure 4: Attention maps of first layers of fine-grained BERT models for English and Chinese sentences. The Chinese sentences are "商店里的乒乓球拍卖完了 (Table tennis bats are sold out in the shop)", "北上京城施展平生报复 (Go north to Beijing to fulfill the dream)", "南京市长江大桥位于南京 (The Nanjing Yantze River bridge is located in Nanjing)". Different colors represent attention weights in different heads and darkness represents weight.

hyper-parameters in pre-training. The optimizer is Adam (Kingma and Ba, 2014). To enhance efficiency, we use mixed precision for all the models. Training is carried out on Nvidia V-100. The numbers of GPUs used for training are from 32 to 64, depending on the model sizes.

Table 8: Hyper-parameters for pre-trained AMBERT.

Hyperparam	Chinese AMBERT	English AMBERT
Number of Layers l	12	12
Hidden Size d	768	768
Sequence Lengh n	512	512
FFN Inner Hidden Size	3072	3072
Attention Heads	12	12
Attention Head Size	64	64
Dropout	0.1	0.1
Attention Dropout	0.1	0.1
Warmup Steps	10,000	10,000
Peak Learning Rate	1e-4	1e-4
Batch Size	512	1024
Weight Decay	0.01	0.01
Max Steps	1m	500k
Learning Rate Decay	Linear	Linear
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999

C.2 Hyper-parameters in Fine-tuning

For the Chinese tasks, since all the original papers do not report detailed hyper-parameters in fine-tuning of the baseline models, we uniformly use the same hyper-parameters as shown in Table 11 except training epoch, because AMBERT and AMBERT-Combo have more parameters and need more training to get converged. We choose the training epochs for all models when the performances on development sets stop to improve. Table 11 also shows all the hyper-parameters in fine-tuning of the English models. We adopt the best hyper-parameters in the original papers for the baselines. Moreover, for AMBERT‡, we also tune learning rate ([1e-5, 2e-5, 3e-5]) and batch size ([16, 32]) for GLUE with the same method in RoBERTa (Liu et al., 2019).

D Data Augmentation

To enhance the performance, we conduct data augmentation for the three Chinese classification tasks of TNEWS, CSL, and CLUEWSC2020. In TNEWS, we use both keywords and titles. In CSL, we concatenate keywords with a special token "_". In CLUEWSC2020, we duplicate a few instances having pronouns in the training data such as "她 (she)".

E Regularization in Fine-tuning

Table 9 shows the results of using different values as regularization coefficients in fine-tuning on the development sets of CLUE, GLUE and RACE. It appears that for most tasks the use of regularization

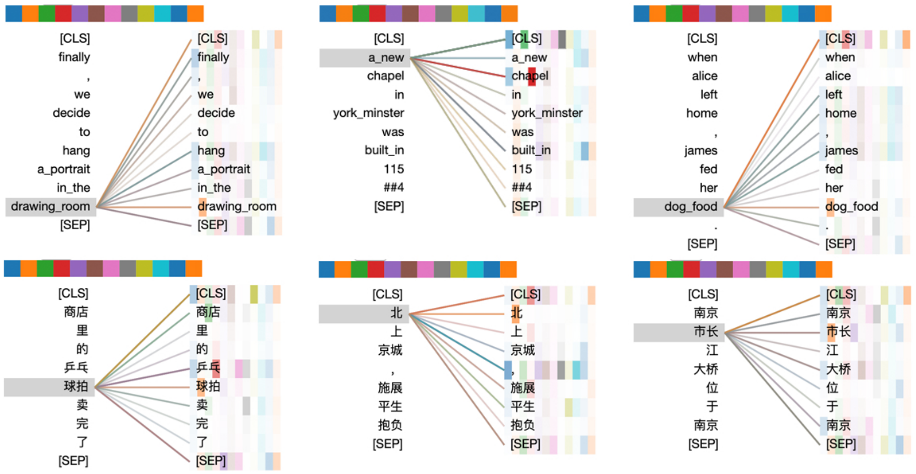

Figure 5: Attention maps of first layers of coarse-grained BERT models for English and Chinese sentences. Note that tokenizations may have errors.

Table 9: Performances on the development sets of CLUE, GLUE and RACE with different regularization coefficients in fine-tuning. CN-Models and EN-Models stand for Chinese and English pre-trained models respectively. CoLA uses Matthew's Corr. The other metrics are accuracies.

CN-Models	λ	TNEWS	IFLYTEK	CLUEWSC2020	AFQMC	CSL	CMNLI	ChID	C^3	-
AMBERT	1.0	**68.1**	60.1	**81.6**	74.7	**85.6**	**82.3**	**87.1**	**69.2**	-
AMBERT	0.0	67.9	**60.3**	80.9	**75.4**	85.0	81.1	86.5	**69.2**	-
EN-Models	λ	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	RACE
AMBERT	1.0	**61.7**	**94.3**	**92.3**	**55.0**	**91.2**	**86.2**	**91.3**	**70.2**	66.6
AMBERT	0.0	61.5	93.4	90.1	54.5	91.1	85.5	91.2	**70.2**	**66.8**

Table 10: The rate of coarse-grained tokens (not included in fine-grained vocabulary) in coarse-grained tokenization.

Datasets	Chinese words	Chinese total tokens	word rate (%)
CMNLI	157,511	335,187	47.0
TNEWS	71,636	137,965	51.9
TNEWS	94,439	165,847	56.9
CSL	836,976	1,739,954	48.1
CHID	958,893	1,763,507	53.4
Avg.	-	-	51.5
Datasets	English phrases	English total tokens	phrase rate (%)
MNLI	43,661	318,985	13.7
QNLI	59,506	395,681	15.0
QNLI	35,256	237,731	14.8
SST-2	9,757	103,048	9.47
CoLA	10,491	82,353	12.7
Avg.	-	-	13.1

is necessary. For simplicity, we did not use the best value of coefficient for each task and instead we adopt 0.0 for RACE and 1.0 for the other tasks.

F Case study

We also qualitatively study the results of BERT and AMBERT, and find that they support our claims (cf., Section 1) very well. Here, we give some random examples from the entailment tasks (QNLI and CMNLI) in Table 12. One can have the following observations. 1) The fine-grained models (e.g., Our BERT word) cannot effectively use complete lexical units such as “Doctor Who” and “打死” (sentence pairs 1 and 5), which may result in incorrect predictions. 2) The coarse-grained models (e.g., Our BERT phrase), on the other hand, cannot effectively deal with incorrect tokenizations, for example, “the blind” and “格式” (sentence pairs 2 and 6). 3) AMBERT is able to make effective use of complete lexical units such as “sister station” in sentence pair 4 and “员工/ 工人” in sentence pair 7, and robust to incorrect tokenizations, such as “used to” in sentence pair 3. 4) AMBERT can in general make more accurate decisions on difficult sentence pairs with both fine-grained and coarse-grained tokenization results.

Table 11: Hyper-parameters for fine-tuning of both Chinese and English tasks.

Dataset	Modes	Batch Size	Max Length	Epoch	Learning Rate	λ
TNEWS/IFLYTEK/AFQMC/CSL/CMNLI	Our BERT, AMBERT-Hybrid	32	128	5	2e-5	-
	AMBERT, AMBERT-Combo	32	128	8	2e-5	1.0
CLUEWSC2020	Our BERT, AMBERT-Hybrid	8	128	50	2e-5	-
	AMBERT, AMBERT-Combo	8	128	80	2e-5	1.0
CMRC2018	All the models	32	512	2	2e-5	-
ChID	Our BERT, AMBERT-Hybrid	24	64	3	2e-5	-
	AMBERT, AMBERT-Combo	24	64	3	2e-5	1.0
C^3	Our BERT, AMBERT-Hybrid	24	512	8	2e-5	-
	AMBERT, AMBERT-Combo	24	512	8	2e-5	1.0
SST-2/MRPC/QQP/MNLI/QNLI	Our BERT, AMBERT-Hybrid	32	512	4	2e-5	-
	AMBERT, AMBERT-Combo	32	512	6	2e-5	1.0
CoLA/STS-B	Our BERT, AMBERT-Hybrid	32	512	10	2e-5	-
	AMBERT, AMBERT-Combo	32	512	20	2e-5	1.0
RTE	Our BERT, AMBERT-Hybrid	32	512	20	2e-5	-
	AMBERT, AMBERT-Combo	32	512	50	2e-5	1.0
SQuAD (1.1 and 2.0)	All the models	32	512	3	2e-5	-
RACE	All except the following two	16	512	4	1e-5	-
	AMBERT, AMBERT-Combo	32	512	6	1e-5	0.0

Table 12: Case study for sentence matching tasks in both English and Chinese (QNLI and CMNLI). The value “0” denotes entailment relation, while the value “1” denotes no entailment relation. WORD/PHRASE represents Our BERT word/phrase. In English the tokens in the same phrase are concatenated with “_”, and in Chinese phrases are split with “/”.

Sentence1	Sentence2	Label	WORD	PHRASE	AMBERT
What Star Trek episode has a nod to Doctor Who? (What Star_Trek episode has_a nod to Doctor_Who?)	There have also been many references to Doctor Who in popular culture and other science fiction, including Star Trek: The Next Generation (”The Neutral Zone”) and Leverage. (There have also been many references_to Doctor_Who in popular_culture and_other science_fiction, including Star_Trek: the_next_generation (”the neutral_zone”) and leverage.)	0	1	0	0
What was the name of the blind date concept program debuted by ABC in 1966? (What was_the name_of the_blind date concept program debuted by ABC in_1966?)	In December of that year, the ABC television network premiered The Dating Game, a pioneer series in its genre, which was a reworking of the blind date concept in which a suitor selected one of three contestants sight unseen based on the answers to selected questions. (In_December of that_year, the_ABC television_network premiered the dating game, a pioneer series in_its genre, which was_a reworking of_the blind_date concept in_which a suitor selected one_of_three contestants sight unseen based_on_the answers to selected questions.)	0	0	1	0
What are two basic primary resources used to guage complexity? (What are two basic primary resources used_to guage complexity?)	The theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying the amount of resources needed to solve them, such as time and storage. (The_theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying the_amount_of resources needed to_solve them, such_as time and storage.)	0	1	1	0
What is the frequency of the radio station WBT in North Carolina? (What_is the_frequency_of the_radio_station WBT in_north_carolina?)	WBT will also simulcast the game on its sister station WBTFM (99.3 FM), which is based in Chester, South Carolina. (WBT will also simulcast the_game on_its sister_station WBTFM (99.3 FM), which_is based_in Chester, South_Carolina.)	1	0	0	1
只打那些面对我们的人，乔恩告诉阿德林。 (只/打/那些/面对/我们/的/人/，/乔恩/告诉/阿/德/林/。)	“打死那些面对我们的人，” 阿德林对乔恩说。 (“/打死/那些/面对/我们/的/人/，/” /阿/德/林/对/乔恩/说/。。)	1	0	1	1
教堂有一个更精致的巴洛克讲坛。 (教堂/有/一个/更/精致/的/巴洛克/讲坛/。)	教堂有一个巴罗格式的讲坛。 (教堂/有/一个/巴/罗/格式/的/讲坛/。)	0	0	1	0
我们已经采取了一系列措施来增强我们员工的能力，并对他们进行投资。 (我们/已经/采取/了/一/系列/措施/来/增强/我们/员工/的/能力/，/并/对/他们/进行/投资/。)	我们一定会投资在我们的工人身上。 (我们/一定/会/投资/在/我们/的/工人/身上/。)	0	1	1	0
科技行业的故事之所以活跃起来，是因为现实太平淡了。 (科技/行业/的/故事/之所以/活跃/起来/，/是/因为/现实/太平/淡/了/。)	现实是如此平淡，以致于虚拟现实技术业务得到了刺激。 (现实/是/如此/平淡/，/以致/于/虚拟/现实/技术/业务/得到/了/刺激/。)	1	0	0	1

Multimodal Incremental Transformer with Visual Grounding for Visual Dialogue Generation

Feilong Chen, Fandong Meng, Xiuyi Chen, Peng Li, Jie Zhou
Pattern Recognition Center, WeChat AI, Tencent Inc, Beijing, China
{ivess.chan,hugheren.chan}@gmail.com
{fandongmeng,patrickpli,withtomzhou}@tencent.com

Abstract

Visual dialogue is a challenging task since it needs to answer a series of coherent questions on the basis of understanding the visual environment. Previous studies focus on the implicit exploration of multimodal co-reference by implicitly attending to spatial image features or object-level image features but neglect the importance of locating the objects explicitly in the visual content, which is associated with entities in the textual content. Therefore, in this paper we propose a **M**ultimodal **I**ncremental **T**ransformer with **V**isual **G**rounding, named MITVG, which consists of two key parts: visual grounding and multimodal incremental transformer. Visual grounding aims to explicitly locate related objects in the image guided by textual entities, which helps the model exclude the visual content that does not need attention. On the basis of visual grounding, the multimodal incremental transformer encodes the multi-turn dialogue history combined with visual scene step by step according to the order of the dialogue and then generates a contextually and visually coherent response. Experimental results on the VisDial v0.9 and v1.0 datasets demonstrate the superiority of the proposed model, which achieves comparable performance.

1 Introduction

Recently, there is increasing interest in vision-language tasks, such as image caption (Xu et al., 2015; Anderson et al., 2016, 2018; Cornia et al., 2020) and visual question answering (Ren et al., 2015a; Gao et al., 2015; Lu et al., 2016; Anderson et al., 2018). In the real world, our conversations (Chen et al., 2020b, 2019) usually have multiple turns. As an extension of conventional single-turn visual question answering, Das et al. (2017) introduce a multi-turn visual question answering task named visual dialogue, which aims to explore the ability of an AI agent to hold a meaningful multi-turn dialogue with humans in natural language about visual content.

Caption: there is a frisbee team with their coach taking a team photo

Q1: how many people ?	A1: 7 people
Q2: is anyone holding a frisbee ?	A2: yes
Q3: is the coach on the right ?	A3: yes, on the far right
Q4: are they wearing matching uniforms ?	A4: all except the coach

Figure 1: An example of visual dialogue. The color in text background corresponds to the same color box in the image, which indicates the same entity. Our model firstly associates textual entities with objects explicitly and then gives contextually and visually coherent answers to contextual questions.

Visual dialogue (Agarwal et al., 2020; Wang et al., 2020; Qi et al., 2020; Murahari et al., 2020) requires agents to give a response on the basis of understanding both visual and textual content. One of the key challenges in visual dialogue is how to solve multimodal co-reference (Das et al., 2017; Kottur et al., 2018). Therefore, some fusion-based models (Das et al., 2017) are proposed to fuse spatial image features and textual features in order to obtain a joint representation. Then attention-based models (Lu et al., 2017; Wu et al., 2018; Kottur et al., 2018) are proposed to dynamically attend to spatial image features in order to find related visual content. Furthermore, models based on object-level image features (Niu et al., 2019; Gan et al., 2019; Chen et al., 2020a; Jiang et al., 2020a; Nguyen

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 436–446
August 1–6, 2021.

et al., 2020; Jiang et al., 2020b) are proposed to effectively leverage the visual content for multimodal co-reference. However, as implicit exploration of multimodal co-reference, these methods implicitly attend to spatial or object-level image features, which is trained with the whole model and is inevitably distracted by unnecessary visual content. Intuitively, specific mapping of objects and textual entities can reduce the noise of attention. As shown in Figure 1, the related objects can help the agent to understand the entities (e.g., Q1: "*people*", Q2: "*frisbee*", Q3: "*coach*") for the generation of correct answers. Then when it answers the question Q4 "*are they wearing matching uniforms ?*", the agent has already comprehended "*people*" and "*coach*" from the previous conversation. On this basis, it can learn the entity "*uniforms*" with the corresponding object in the image, and generate the answer "*all except the coach*". To this end, we need to 1) explicitly locate related objects guided by textual entities to exclude undesired visual content, and 2) incrementally model the multi-turn structure of the dialogue to develop a unified representation combining multi-turn utterances with the corresponding related objects. However, previous work overlooks these two important aspects.

In this paper, we thus propose a novel and effective **M**ultimodal **I**ncremental **T**ransformer with **V**isual **G**rounding, named MITVG, which contains two key parts: visual grounding and multimodal incremental transformer. Visual grounding aims to establish specific mapping of objects and textual entities by explicitly locating related objects in the image with the textual entities. By doing so, our model can exclude undesired visual content and reduce attention noise. On the basis of visual grounding, the multimodal incremental transformer is used to model the multi-turn dialogue history combined with the specific visual content to generate visually and contextually coherent responses. As an encoder-decoder framework, MITVG contains a Multimodal Incremental Transformer Encoder (MITE) and a Gated Cross-Attention Decoder (GCAD).

We test the effectiveness of our proposed model on large-scale datasets: VisDial v0.9 and v1.0 (Das et al., 2017). Both automatic and manual evaluations show that our model substantially outperforms the competitive baselines and achieves the new state-of-the-art results on substantial metrics. Our main contributions are as follows:

- To the best of our knowledge, we are the first to leverage visual grounding to explicitly locate related objects in the image guided by textual entities for visual dialogue.
- We propose a novel multimodal incremental transformer to encode the multi-turn dialogue history step by step combined with the visual content and then generate a contextually and visually coherent response.
- We achieve comparable performance on VisDial v0.9 and v1.0 datasets.

2 Approach

2.1 Overview

In this section, we formally describe the visual dialogue task and then proceed to our proposed **M**ultimodal **I**ncremental **T**ransformer with **V**isual **G**rounding (MITVG).

Following Das et al.(2017), a visual dialogue agent is given three inputs, i.e., an image I, a dialogue history (the caption and question-answer pairs) till round $t-1$: $H = (\underbrace{Cap}_{H_0}, \underbrace{(Q_1, A_1)}_{H_1}, \cdots, \underbrace{(Q_{t-1}, A_{t-1})}_{H_{t-1}})$ and the current question Q_t at round t, where Cap is the caption describing the image taken as H_0 and $H_1, \ldots, H_{t-1}$ are concatenations of question-answer pairs. The goal of the visual dialogue agent is to generate a response (or answer) A_t to the question Q_t. Cap, Q_* and A_* are sentences.

Figure 2 shows the framework of MITVG, which aims to explicitly model multi-turn dialogue history step by step based on the explicit modeling relationship between multiple modalities. MITVG firstly locates related objects in the image explicitly guided by the textual entities via visual grounding, then encodes multi-turn dialogue history in the order of the dialogue utterance based on visual grounding via Multimodal Incremental Encoder (MITE), and finally utilizes the outputs of both encoder and visual grounding to generate the response word by word via Gated Cross-Attention Decoder (GCAD).

2.2 Input Representation

Before describing our method, we introduce the input representation.

Image Features. We use a pre-trained Faster R-CNN model (Ren et al., 2015b) to extract object-

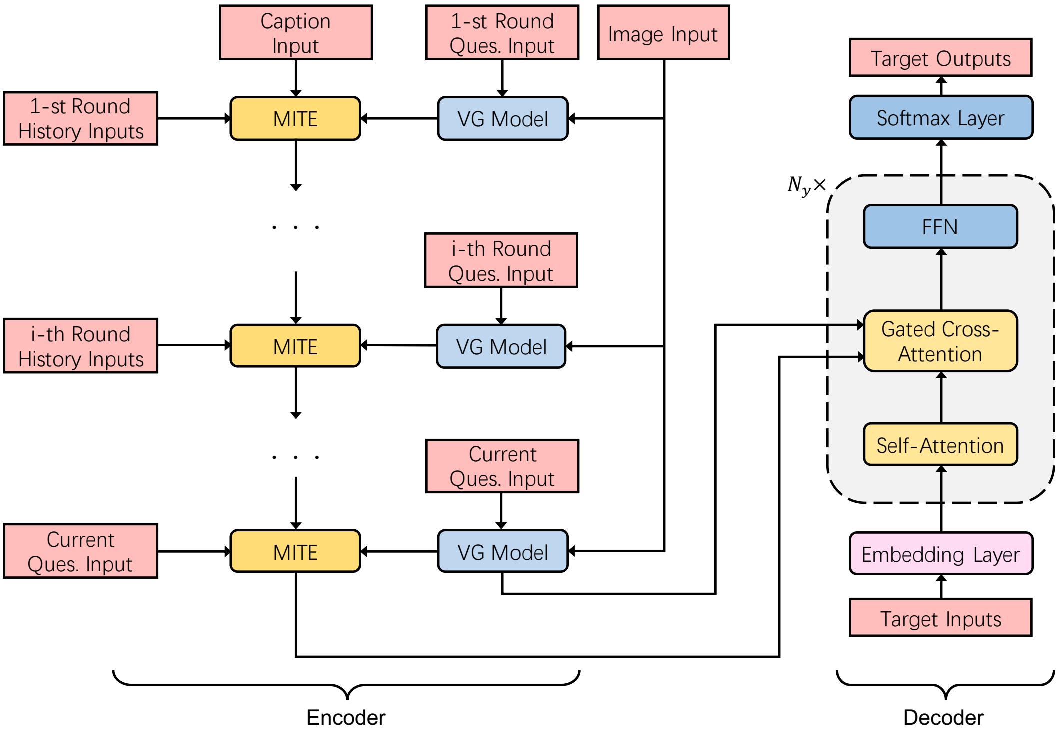

Figure 2: The framework of **M**ultimodal **I**ncremental **T**ransformer with **V**isual **G**rounding (MITVG). "VG Model" indicates visual grounding model (Yang et al., 2019b) (Details are described in Sec. 2.3). "MITE" denotes the multimodal incremental transformer encoder (Details are described in Sec. 2.4.1). MITVG firstly uses the VG model to explicitly model the relationship between the textual content and the visual content, and encodes multi-turn dialogue history in the order of the dialogue based on visual grounding, and finally utilizes the outputs of both encoder and visual grounding to generate the response word by word in the decoding process.

level image features. Specifically, the image features v for an image I are represented by:

$$v = \mathrm{Faster\ R-CNN}(I) \in \mathbb{R}^{K\times V}, \quad (1)$$

where K denotes the total number of the detected objects per image and V denotes the dimension of features for each object.

Language Features. The current (at the t-th round) L-word question features are a sequence of M-dimension word embedding with positional encoding added (Vaswani et al., 2017), as follows:

$$q_t = [s_{t,1}, s_{t,2}, \ldots, s_{t,L}] \in \mathbb{R}^{L\times M}, \quad (2)$$

$$s_{t,j} = w_j + PE(j), \quad (3)$$

where w_j is the word embedding of the j-th word in the question Q_t, and $PE(\cdot)$ denotes positional encoding function (Vaswani et al., 2017). For the dialogue history $H = \{H_0, H_1, \ldots, H_{t-1}\}$ and the answer A_t, the dialogue history features $u = \{u_0, u_1, \ldots, u_{t-1}\}$ and the answer features a_t are obtained in the same way as the question Q_t.

2.3 Visual Grounding

To exclude the needless visual content, we introduce visual grounding, which is defined to ground a natural language query (phrase or sentence) about an image onto a correct region of the image. First of all, we use NeuralCoref[1] for reference resolution. For example, when it processes the question Q4 "*are they wearing matching uniforms ?*" shown in Figure 1, NeuralCoref takes the question Q4 and its history as inputs, and then generates a new question "*are the people wearing matching uniforms ?*" as a new Q4. As shown in Figure 3 (a), visual grounding model (Yang et al., 2019b) takes the i-th question Q_i and the image I as inputs and generates initial visual grounding features, as follows:

$$v_{g_i}^{(0)} = \mathrm{VGM}(Q_i, I), \quad (4)$$

where $\mathrm{VGM}(\cdot)$ denotes visual grounding model[2]. Then $v_{g_i}^{(0)}$ is sent to the multi-head self-attention

---

[1] Introduction and code of NeuralCoref are available at https://github.com/huggingface/neuralcoref. NeuralCoref is only used for visual grounding.

[2] Introduction and code are available at https://github.com/zyang-ur/onestage_grounding.

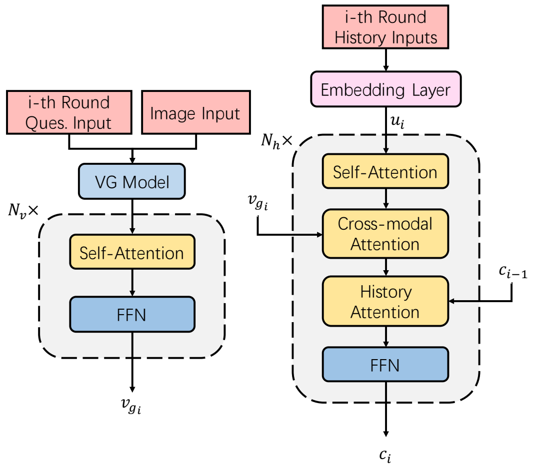

(a) (b)

Figure 3: Framework of (a) Visual Grounding and (b) Multimodal Incremental Transformer Encoder (MITE).

layer followed by a position wise feed-forward network (FFN) layer (stacked N_v times) to generate the i-th visual grounding features as follows[3]:

$$\hat{v}_{g_i}^{n} = \text{MultiHead}\left(v_{g_i}^{(n-1)}, v_{g_i}^{(n-1)}, v_{g_i}^{(n-1)}\right), \quad (5)$$

where $n = 1, \ldots, N_v$ and $\text{MultiHead}(\cdot)$ denotes the multi-head self-attention layer (Vaswani et al., 2017), then

$$v_{g_i}^{(n)} = \text{FFN}\left(\hat{v}_{g_i}^{n}\right), \quad (6)$$

where $n = 1, \ldots, N_v$ and $\text{FFN}(\cdot)$ denotes the position wise feed-forward networks (Vaswani et al., 2017). After N_v layers computation, we obtain the final visual grounding features v_{g_i} by:

$$v_{g_i} = v_{g_i}^{(N_v)}, \quad (7)$$

Actually, there are some questions that do not contain any entities in the visual dialogue, such as "*anything else ?*". For such questions, we use the features of the whole image instead, i.e. $v_{g_i} = v$.

2.4 Multimodal Incremental Transformer

Inspired by the idea of incremental transformer (Li et al., 2019) which is originally designed for the single-modal dialogue task, we make an extension and propose a multimodal incremental transformer, which is composed of a Multimodal Incremental Transformer Encoder (MITE) and a Gated Cross-Attention Decoder (GCAD). The MITE uses an incremental encoding scheme to encode multi-turn dialogue history with an understanding of the image. The GCAD leverages the outputs from both the encoder and visual grounding via the gated cross-attention layer to fuse the two modal information in order to generate a contextually and visually coherent response word by word.

2.4.1 MITE

To effectively encode multi-turn utterances grounded in visual content, we design the Multimodal Incremental Transformer Encoder (MITE). As shown in Figure 3 (b), at the i-th round, where $i = 1, 2, ..., t-1$, the MITE takes the visual grounding features v_{g_i}, the dialogue history features u_i and the context state c_{i-1} as inputs, and utilizes attention mechanism to incrementally build up the representation of the relevant dialogue history and the associated image regions, and then outputs the new context state c_i. This process can be stated recursively as follows:

$$c_i = \text{MITE}\left(v_{g_i}, u_i, c_{i-1}\right), \quad (8)$$

where $\text{MITE}(\cdot)$ denotes the encoding function, c_i denotes the context state after the dialogue history features u_i and the visual grounding features v_{g_i} being encoded, and c_0 is the dialogue history features u_0.

As shown in Figure 3 (b), we use a stack of N_h identical layers to encode v_{g_i}, u_i and c_{i-1}, and to generate c_i. Each layer consists of four sub-layers. **The first sub-layer** is a multi-head self-attention for the dialogue history:

$$\mathrm{A}^{(\mathrm{n})} = \text{MultiHead}\left(\mathrm{C}^{(n-1)}, \mathrm{C}^{(n-1)}, \mathrm{C}^{(n-1)}\right), \quad (9)$$

where $n = 1, \ldots, N_h$, $\mathrm{C}^{(n-1)}$ is the output of the last layer N_{n-1}, and $\mathrm{C}^{(0)}$ is the dialog history features u_i. **The second sub-layer** is a multi-head cross-modal attention:

$$\mathrm{B}^{(\mathrm{n})} = \text{MultiHead}\left(\mathrm{A}^{n}, v_{g_i}, v_{g_i}\right), \quad (10)$$

where v_{g_i} is the visual grounding features. **The third sub-layer** is a multi-head history attention:

$$\mathrm{F}^{(\mathrm{n})} = \text{MultiHead}\left(\mathrm{B}^{(n)}, \mathrm{c}_{i-1}, \mathrm{c}_{i-1}\right), \quad (11)$$

where c_{i-1} is the context state after the previous dialogue history features u_{i-1} being encoded. That's why we call this encoder "Multimodal Incremental Transformer". **The fourth sub-layer** is a position wise feed-forward network (FFN):

$$\mathrm{C}^{(\mathrm{n})} = \text{FFN}\left(\mathrm{F}^{(n)}\right). \quad (12)$$

---

[3] For simplicity, we omit the descriptions of layer normalization and residual connection.

We use c_i to denote the final representation at N_h-th layer:

$$c_i = \mathrm{C}^{(N_h)}. \tag{13}$$

The mulitmodal incremental transformer encoder at the current turn t, i.e., the bottom one in Figure 2, has the same structure as all the other MITEs but takes the visual grounding features v_{g_t}, the current question features q_t and the context state c_{t-1} as inputs and generates the final context state c_t.

2.4.2 GCAD

Motivated by the real-world human cognitive process, we design a Gated Cross-Attention Decoder (GCAD) shown in Figure 2, which takes the masked answer features $a_{<z}$ (where $z = 1, 2, ..., Z$ and Z is the length of the answer), encoder outputs c_t and visual grounding features v_{g_t} as inputs, and generates contextually and visually coherent responses grounded in an image. GCAD is composed of a stack of N_y identical layers, each of which has three sub-layers.

The first sub-layer is a multi-head self-attention as follows:

$$\mathrm{J}^{(n)} = \mathrm{MultiHead}\left(\mathrm{R}^{(n-1)}, \mathrm{R}^{(n-1)}, \mathrm{R}^{(n-1)}\right), \tag{14}$$

where $n = 1, \ldots, N_y$, $\mathrm{R}^{(n-1)}$ is the output of the previous layer, and $\mathrm{R}^{(0)}$ is the masked answer features $a_{<z}$.

The second sub-layer is a multi-head gated cross-modal attention layer (GCA) as shown in Figure 4, calculated as:

$$\mathrm{P}^{(n)} = \alpha^{(n)} \circ \mathrm{E}^{(n)} + \beta^{(n)} \circ \mathrm{G}^{(n)}, \tag{15}$$

where $n = 1, \ldots, N_y$, $\circ$ denotes Hadamard product, $\mathrm{E}^{(n)}$ and $\mathrm{G}^{(n)}$ denote the outputs of two cross-attention functions, computed as follows:

$$\begin{aligned} \mathrm{E}^{(n)} &= \mathrm{MultiHead}\left(\mathrm{J}^{(n)}, c_t, c_t\right), & (16) \\ \mathrm{G}^{(n)} &= \mathrm{MultiHead}\left(\mathrm{J}^{(n)}, v_{g_t}, v_{g_t}\right), & (17) \end{aligned}$$

where $\alpha^{(n)}$, $\beta^{(n)}$ are two gates[4]:

$$\begin{aligned} \alpha^{(n)} &= \sigma\left(W_E[\mathrm{J}^{(n)}, E^{(n)}] + b_E\right), & (18) \\ \beta^{(n)} &= \sigma\left(W_G[\mathrm{J}^{(n)}, G^{(n)}] + b_G\right), & (19) \end{aligned}$$

where σ denotes sigmoid function, W_E, W_G, b_E, b_G are learnable parameters, and $[\cdot, \cdot]$ indicates concatenation.

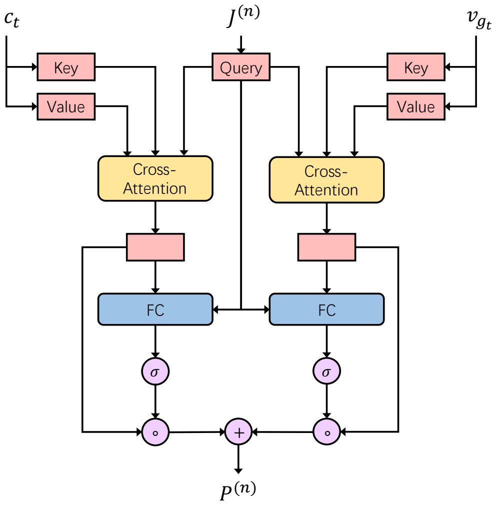

Figure 4: Framework of Gated Cross-Attention (GCA) in the Deocer.

The third sub-layer is a position wise feed-forward network (FFN):

$$\mathrm{R}^{(n)} = \mathrm{FFN}\left(\mathrm{P}^{(n)}\right). \tag{20}$$

We use r_z to denote the final representation at N_y-th layer:

$$r_z = \mathrm{R}^{(N_y)}. \tag{21}$$

Finally, we use softmax to get the word probabilities $\hat{a}_z$:

$$\hat{a}_z = \mathrm{softmax}(r_z). \tag{22}$$

3 Experiments

3.1 Datasets

We conduct experiments on the VisDial v0.9 and v1.0 datasets (Das et al., 2017) to verify our approach. VisDial v0.9 contains 83k dialogs on COCO-train (Lu et al., 2017) and 40k dialogs on COCO-val images as test set, for a total of 1.23M dialog question-answer pairs. VisDial v1.0 dateset is an extension of VisDial v0.9 dateset with additional 10k COCO-like images from Flickr. VisDial v1.0 dateset contains 123k, 2k and 8k images as train, validation and test splits, respectively.

3.2 Implementation and Evaluation

Implementation Details. Following previous work (Das et al., 2017), in order to represent words we firstly lowercase all the texts and convert digits to words, and then remove contractions before tokenization. The captions, questions and answers are further truncated to ensure that they are not longer than 40, 20 and 20 tokens, respectively. We construct the vocabulary of tokens that appear at least

---

[4] Our inspiration comes from Cornia et al. (2020).

Model	Object	Vis-G	MRR ↑	R@1 ↑	R@5↑	R@10 ↑	Mean ↓
AP (Das et al., 2017)	×	×	37.35	23.55	48.52	53.23	26.50
NN (Das et al., 2017)	×	×	42.74	33.13	50.83	58.69	19.62
LF (Das et al., 2017)	×	×	51.99	41.83	61.78	67.59	17.07
HREA (Das et al., 2017)	×	×	52.42	42.28	62.33	68.71	16.79
MN (Das et al., 2017)	×	×	52.59	42.29	62.85	68.88	17.06
HCIAE (Lu et al., 2017)	×	×	53.86	44.06	63.55	69.24	16.01
CorefNMN (Kottur et al., 2018)	×	×	53.50	43.66	63.54	69.93	15.69
CoAtt (Wu et al., 2018)	×	×	55.78	46.10	65.69	71.74	14.43
RvA (Niu et al., 2019)	✓	×	55.43	45.37	65.27	72.97	**10.71**
DVAN (Guo et al., 2019b)	✓	×	55.94	46.58	65.50	71.25	14.79
VDBERT (Wang et al., 2020)	✓	×	55.95	46.83	65.43	72.05	13.18
LTMI (Nguyen et al., 2020)†	✓	×	55.85	46.07	65.97	72.44	14.17
DMRM (Chen et al., 2020a)	✓	×	55.96	46.20	66.02	72.43	13.15
MITVG	✓	✓	**56.83**	**47.14**	**67.19**	**73.72**	11.95

Table 1: Performance on VisDial val v0.9 (Das et al., 2017). † indicates that we re-implement the model. "Object" and "Vis-G" denote if the model uses object-level image features and visual grounding, respectively. Underline denotes the highest score among baselines. Our MITVG exceeds previous work on most of the metrics and achieves comparable performance.

Model	Object	Vis-G	MRR ↑	R@1 ↑	R@5↑	R@10 ↑	Mean ↓	NDCG ↑
MN (Das et al., 2017)‡	✓	×	47.99	38.18	57.54	64.32	18.60	51.86
HCIAE (Lu et al., 2017)‡	✓	×	49.07	39.72	58.23	64.73	18.43	59.70
CoAtt (Wu et al., 2018)‡	✓	×	49.64	40.09	59.37	65.92	17.86	59.24
Primary (Guo et al., 2019a)	✓	×	49.01	38.54	59.82	66.94	16.60	-
ReDAN (Gan et al., 2019)	✓	×	50.02	40.27	59.93	66.78	17.40	60.47
DMRM (Chen et al., 2020a)	✓	×	50.16	40.15	60.02	67.21	15.19	-
LTMI (Nguyen et al., 2020)†	✓	×	50.38	40.30	60.72	68.44	15.73	61.61
DAM (Jiang et al., 2020b)	✓	×	50.51	40.53	60.84	67.94	16.65	60.93
KBGN (Jiang et al., 2020a)	✓	×	50.05	40.40	60.11	66.82	17.54	60.42
MITVG	✓	✓	**51.14**	**41.03**	**61.25**	**68.49**	**14.37**	61.47

Table 2: Performance on VisDial val v1.0 (Das et al., 2017). ‡ denotes that all the models are re-implemented by Gan et al. (2019). Our MITVG outperforms previous work and achieves comparable performance.

5 times in the training split. To represent image regions, we use Faster R-CNN (Ren et al., 2015b) with ResNet-101 (He et al., 2016) finetuned on the Visual Genome dataset (Krishna et al., 2017), thus obtaining a 2048-dimensional feature vector for each region. The layers of our encoder, decoder and visual grounding module are all set to 3. The number of attention heads in multi-head attention is 8 and the filter size is 2048. The word embedding is shared by the history, questions and responses. The dimension of word embedding is set to 512 empirically. We use Adam (Kingma and Ba, 2014) for optimization, following the learning rate scheduling strategy of Vaswani et al. (2017). Our model is implemented using PyTorch v1.0, Python v3.6, and provides out of the box support with CUDA 9 and CuDNN 7. We train our model on TITAN XP with 8 GPUs. For each epoch, we spend about 9,000 seconds on training the model. The total parameters are about 56.79M.

Before we train our model, we use three external tools for image features extracting, reference resolution and visual grounding.

Image Features Extracting We extract image features of VisDial images, using a Faster-RCNN (Ren et al., 2015b) with ResNet-101 (He et al., 2016) pre-trained on Visual Genome (Krishna et al., 2017), introduction and code from https://github.com/peteanderson80/bottom-up-attention.

Reference Resolution we use NeuralCoref v4.0 for reference resolution, which is developed by huggingface. Introduction and code are available at https://github.com/huggingface/neuralcoref.

Visual Grounding We use One-Stage Visual Grounding Model (Yang et al., 2019b) to obtain the visual grounding features. Introduction and code are available at https://github.com/zyang-ur/onestage_grounding.

Automatic Evaluation. We use a retrieval setting to evaluate individual responses at each round of a dialogue, following Das et al. (2017). Specifically, at test time, apart from the image, ground truth dialogue history and the question, a list of 100-candidate answers is also given. The model is evaluated on retrieval metrics: (1) rank of human response (Mean, the lower the better), (2) existence of the human response in $top-k$ ranked responses, i.e., R@k (3) mean reciprocal rank (MRR) of the human response (the higher the better) and (4) normalized discounted cumulative gain (NDCG) for VisDial v1.0 (the higher the better). During evaluation, we use the log-likelihood scores to rank candidate answers.

Human Evaluation. We randomly extract 100 samples for human evaluation according to Wu et al. (2018), and then ask 3 human subjects to guess whether the last response in the dialogue is human-generated or machine-generated. If at least 2 of them agree it is generated by a human, we think it passes the Truing Test (M1). In addition, we record the percentage of responses that are evaluated better than or equal to human responses (M2), according to the human subjects' evaluation.

3.3 Main Results

We compare our proposed model to the state-of-the-art *generative models* developed in previous work. Current encoder-decoder based generative models can be divided into tree facets. (1) Fusion-based models: LF (Das et al., 2017) and HREA (Das et al., 2017) directly encode the multimodal inputs and decode the answer. (2) Attention-based models: HCIAE (Lu et al., 2017), CoAtt (Wu et al., 2018), Primary (Guo et al., 2019a), ReDAN (Gan et al., 2019), DVAN (Guo et al., 2019b) and DMRM (Chen et al., 2020a), DAM, LTMI, KBGN. (3) Visual co-reference resolution models: CorefNMN (Kottur et al., 2018), RvA (Niu et al., 2019). (4) The pretraining model: VDBERT (Wang et al., 2020).

As shown in Table 1 and Table 2, our MITVG, which explicitly locates related objects guided by the textual entities and implements a multimodal incremental transformer to incrementally build the representation of the dialogue history and the image, achieves comparable performance on the VisDial v0.9 and v1.0 datasets. Specifically, our model outperforms previous work by a significant margin both on the VisDial v0.9 dataset (0.87 on MRR, 0.31 on R@1, 1.17 on R@5, 0.75 on R10) and the VisDial v1.0 dataset (0.98 on MRR, 0.76 on R@1, 1.23 on R@5, 1.28 on R10, 0.82 on Mean, and 1.00 on NDCG). The improvement of R@10 is the largest and our method also gains a large increase on MRR and R@1 due to the explicit modeling of multiple modalities (Seeing Sec 3.5 for further quantitative analysis).

	DMRM	MITVG
Method 1 (M1)	0.62	**0.76**
Method 2 (M2)	0.59	**0.70**

Table 3: Human evaluation on 100 sampled responses on VisDial val v1.0. M1: percentage of responses pass the Turing Test. M2: percentage of responses evaluated better than or equal to human responses.

As shown in Table 3, we conduct human study to further prove the effectiveness of our model. Our model achieves the highest scores both on the metric M1 (0.76) and M2 (0.70) compared with the previous model, DMRM (Chen et al., 2020a). These results show that our model can generate a better contextually and visually coherent response.

3.4 Ablation Study

We also conduct an ablation study to illustrate the validity of our proposed Multimodal Incremental Transformer with Visual Grounding. The results are shown in Table 4.

We implement Multimodal Incremental Transformer without Visual Grounding ('MITVG w/o VG') to verify the validity of visual grounding. As shown in Table 4, comparing 'MITVG w/o VG' with MITVG, we find the metrics decrease obviously (0.46 on MRR, 0.60 on R@1, 0.68 on R@5, 0.46 on R@10 and 0.59 on Mean) if visual grounding is deleted from MITVG. This observation demonstrates the validity of visual grounding.

To verify the effectiveness of the incremental transformer architecture, we implement a Multimodal Incremental LSTM without Visual Grounding ('MI-LSTM w/o VG'). A 3-layer bidirectional LSTM (Schuster and Paliwal, 1997) with multi-head attention and a 1-layer LSTM with GCA are applied for encoder and decoder, respectively. All the LSTM hidden state size is 512. Results in Table 4 demonstrate the effectiveness of our incremental transformer architecture (compare 'MITVG w/o VG' with 'MI-LSTM w/o VG'). Results from the comparison between 'MITVG w/o VG' and DMRM (Chen et al., 2020a) also show the validity of our incremental transformer to some extent.

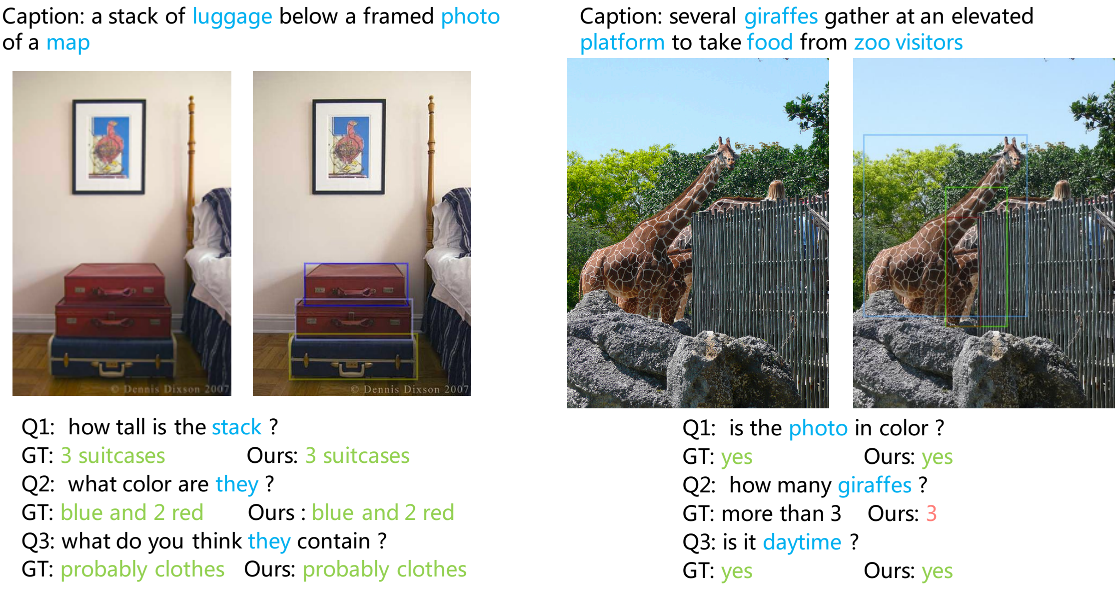

(a) (b)

Figure 5: Case study. The text marked in blue indicates the dialogue topic. The answers marked in green and red indicate the right and wrong answers, respectively. Our MITVG often generates right responses (marked in green) in keeping with human answers.

Model	MRR	R@1	R@5	R@10	Mean
DMRM	50.16	40.15	60.02	67.21	15.19
MITVG	**51.14**	**41.03**	**61.25**	**68.49**	**14.37**
MITVG w/o VG	50.68	40.43	60.57	68.03	14.96
MI-LSTM w/o VG	50.02	39.85	59.86	67.16	15.78

Table 4: Ablation study of our proposed model on VisDial val v1.0. "MI-LISM" indicates Multimodal Incremental LSTM. "VG" indicates visual grounding.

	Train	Validation	Test
VisDial v0.9	2.04	1.95	-
VisDial v1.0	2.05	1.93	1.93

Table 5: Average number of the grounded objects in each question.

3.5 Case Study

As shown in Table 5, we calculate the average number of the objects associated with entities in each question for assistant analysis. As shown in Figure 5 (a), owing to the explicit understanding of visual content via visual grounding and the multimodal incremental transformer architecture, our MITVG generates responses in keeping with human answers. For example, while answering the question Q1 '*'how tall is the stack ?*" and Q2 "*what color are they ?*", our model grounds the three suitcases accurately via visual grounding, thus giving the accurate responses "*3 suitcases*" and "*blue and 2 red*". However, as shown in Figure 5 (b), for questions Q2, MITVG gives a wrong answer because it focuses on wrong number of objects in the question by visual grounding.

4 Related Work

Visual Dialogue. Our work touches two branches of the research in visual dialogue. One is how to leverage image features. Niu et al. (2019) utilize object-level image features as visual attention and refine it by recursively reviewing the dialog history. Gan et al. (2019) and Chen et al. (2020a) regard the object-level image features as visual memory to infer answers progressively through multiple steps. The other is how to model dialogue history. Yang et al. (2019a) propose a new training paradigm inspired by actor-critic policy gradient (Sutton et al., 1999) for history-advantage training. Guo et al. (2020) represent each turn dialogue history with visual content as a node in a context-aware graph neural network. Park et al. (2020) refine history information from both topic aggregation and context matching. Different from these approaches, we explicitly establish specific mapping of objects and textual entities to exclude undesired visual content via visual grounding, and model multi-turn structure of the dialogue based on visual grounding to develop a unified representation combining multi-turn utterances along with the relevant objects.

Incremental Structures. There are some successes on introducing the incremental structure into tasks related to dialog systems (Zilka and Jurcicek, 2015; Coman et al., 2019; Li et al., 2019; Das et al., 2017). In particular, Coman et al. (2019) propose an incremental dialog state tracker which is updated on a token basis from incremental transcriptions. Li et al. (2019) devise an incremental transformer to encode multi-turn utterances along with knowledge in related documents for document grounded conversations. Das et al. (2017) propose a dialog-RNN to produce an encoding for this round and a state for next round. Our model is different from these approaches mainly in two aspects: 1) we explicitly model the relationship between modalities, i.e., textual utterance and image objects, in visual dialogue through visual grounding; 2) based on the explicit association between modalities, our model incrementally encodes the dialogue history and the image with well-designed incremental multimodal architecture to sufficiently understand the dialogue content, thus generating better responses.

5 Conclusion

We propose a novel Multimodal Incremental Transformer with Visual Grounding for visual dialogue, named MITVG, which consists of two key parts: visual grounding and multimodal incremental transformer. Visual grounding aims to explicitly model the relationship between multiple modalities. Based on visual grounding, multimodal incremental transformer aims to explicitly model multi-turn dialogue history in the order of the dialogue. Experiments on the VisDial v0.9 and v1.0 datasets show that our model achieves comparable performance.

References

Shubham Agarwal, Trung Bui, Joon-Young Lee, Ioannis Konstas, and Verena Rieser. 2020. History for visual dialog: Do we really need it? *arXiv preprint arXiv:2005.07493*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic propositional image caption evaluation. *Adaptive Behavior*, 11(4):382–398.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Feilong Chen, Fandong Meng, Jiaming Xu, Peng Li, Bo Xu, and Jie Zhou. 2020a. DMRM: A dual-channel multi-hop reasoning model for visual dialog. *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Xiuyi Chen, Fandong Meng, Peng Li, Feilong Chen, Shuang Xu, Bo Xu, and Jie Zhou. 2020b. Bridging the gap between prior and posterior knowledge selection for knowledge-grounded dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3426–3437.

Xiuyi Chen, Jiaming Xu, and Bo Xu. 2019. A working memory model for task-oriented dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2687–2693.

Andrei C Coman, Koichiro Yoshino, Yukitoshi Murase, Satoshi Nakamura, and Giuseppe Riccardi. 2019. An incremental turn-taking model for task-oriented dialog systems. *arXiv preprint arXiv:1905.11806*.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335.

Zhe Gan, Yu Cheng, Ahmed EI Kholy, Linjie Li, Jingjing Liu, and Jianfeng Gao. 2019. Multi-step reasoning via recurrent dual attention for visual dialog. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6463–6474.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, pages 2296–2304.

Dalu Guo, Chang Xu, and Dacheng Tao. 2019a. Image-question-answer synergistic network for visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10434–10443.

Dan Guo, Hui Wang, and Meng Wang. 2019b. Dual visual attention network for visual dialog. pages 4989–4995.

Dan Guo, Hui Wang, Hanwang Zhang, Zheng-Jun Zha, and Meng Wang. 2020. Iterative context-aware graph inference for visual dialog. *arXiv preprint arXiv:2004.02194*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Xiaoze Jiang, Siyi Du, Zengchang Qin, Yajing Sun, and Jing Yu. 2020a. KBGN: Knowledge-bridge graph network for adaptive vision-text reasoning in visual dialogue. *Proceedings of the 28th ACM International Conference on Multimedia.*

Xiaoze Jiang, Jing Yu, Yajing Sun, Zengchang Qin, Zihao Zhu, Yue Hu, and Qi Wu. 2020b. DAM: Deliberation, abandon and memory networks for generating detailed and non-repetitive responses in visual dialogue. *arXiv preprint arXiv:2007.03310.*

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. *ArXiv*, abs/1809.01816.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Zekang Li, Cheng Niu, Fandong Meng, Yang Feng, Qian Li, and Jie Zhou. 2019. Incremental transformer with deliberation decoder for document grounded conversations. *arXiv preprint arXiv:1907.08854.*

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems*, pages 314–324.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Vishvak Murahari, Dhruv Batra, Devi Parikh, and Abhishek Das. 2020. Large-scale pretraining for visual dialog: A simple state-of-the-art baseline. *Proceedings of the European Conference on Computer Vision.*

Van-Quang Nguyen, Masanori Suganuma, and Takayuki Okatani. 2020. Efficient attention mechanism for visual dialog that can handle all the interactions between multiple inputs. *Proceedings of the European Conference on Computer Vision.*

Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. 2019. Recursive visual attention in visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6679–6688.

Sungjin Park, Taesun Whang, Yeochan Yoon, and Hueiseok Lim. 2020. Multi-view attention networks for visual dialog. *arXiv preprint arXiv:2004.14025.*

Jiaxin Qi, Yulei Niu, Jianqiang Huang, and Hanwang Zhang. 2020. Two causal principles for improving visual dialog. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015a. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015b. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yue Wang, Shafiq Joty, Michael R Lyu, Irwin King, Caiming Xiong, and Steven CH Hoi. 2020. VD-BERT: A unified vision and dialog transformer with bert. *arXiv preprint arXiv:2004.13278.*

Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton van den Hengel. 2018. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6106–6115.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of International Conference on Machine Learning*, pages 2048–2057.

Tianhao Yang, Zheng-Jun Zha, and Hanwang Zhang. 2019a. Making history matter: History-advantage sequence training for visual dialog. In *The IEEE International Conference on Computer Vision (ICCV).*

Zhengyuan Yang, Boqing Gong, Liwei Wang, Wenbing Huang, Dong Yu, and Jiebo Luo. 2019b. A fast and accurate one-stage approach to visual grounding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4683–4693.

Lukas Zilka and Filip Jurcicek. 2015. Incremental lstm-based dialog state tracker. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (Asru)*, pages 757–762. IEEE.

Retrieve & Memorize: Dialog Policy Learning with Multi-Action Memory

Yunhao Li[1], **Yunyi Yang**[1], **Xiaojun Quan**[1]*, **Jianxing Yu**[2]
[1]School of Computer Science and Engineering, Sun Yat-sen University, China
[2]School of Artificial Intelligence, Sun Yat-sen University, China
`{liyh355,yangyy37}@mail2.sysu.edu.cn`
`{quanxj3,yujx26}@mail.sysu.edu.cn`

Abstract

Dialogue policy learning, a subtask that determines the content of system response generation and then the degree of task completion, is essential for task-oriented dialogue systems. However, the unbalanced distribution of system actions in dialogue datasets often causes difficulty in learning to generate desired actions and responses. In this paper, we propose a retrieve-and-memorize framework to enhance the learning of system actions. Specially, we first design a neural context-aware retrieval module to retrieve multiple candidate system actions from the training set given a dialogue context. Then, we propose a memory-augmented multi-decoder network to generate the system actions conditioned on the candidate actions, which allows the network to adaptively select key information in the candidate actions and ignore noises. We conduct experiments on the large-scale multi-domain task-oriented dialogue dataset MultiWOZ 2.0 and MultiWOZ 2.1. Experimental results show that our method achieves competitive performance among several state-of-the-art models in the context-to-response generation task.

1 Introduction

Task-oriented dialogue systems communicate with users through natural language conversations to accomplish a wide range of tasks such as restaurant and flight bookings. Recent years have seen a rapid growth of interest in building task-oriented dialogue systems (Budzianowski et al., 2018). Such systems are usually decomposed into several subtasks, including natural language understanding (Gupta et al., 2018), dialogue state tracking (Zhong et al., 2018), system actions (dialogue policy) prediction, and response generation (Wen et al., 2015; Chen et al., 2019; Zhao et al., 2019), where system actions can be viewed as a semantic plan of response generation. One of the main challenges for context-to-response generation in task-oriented dialogue systems comes from the intrinsic one-to-many property in conversations. As shown in Figure 1, there can be multiple valid system actions for the same dialogue context, which means that multiple satisfactory system responses can be generated correspondingly. However, in most collected dialogue datasets, each dialogue context has only one reference, which leads to an unbalanced distribution of system actions and responses in multi-domain dialogue datasets (Zhang et al., 2020). Models trained on such unbalanced datasets tend to overfit high-frequency system actions and underfit low-frequency ones.

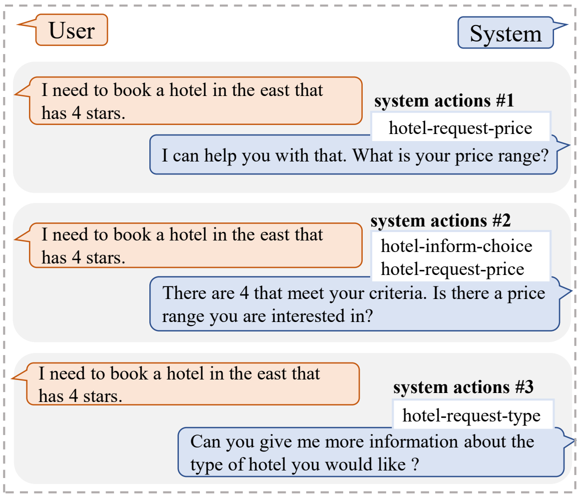

I need to book a hotel in the east that has 4 stars.

I can help you with that. What is your price range?

I need to book a hotel in the east that has 4 stars.

There are 4 that meet your criteria. Is there a price range you are interested in?

I need to book a hotel in the east that has 4 stars.

Can you give me more information about the type of hotel you would like ?

Figure 1: An example of the one-to-many property,where there are multiple appropriate system actions and responses given the same dialogue context.

One line of work focuses on the representation of system actions, which alleviates the unbalanced problem to a certain extent. Chen et al. (2019) reconstruct system actions into a compact graph representation. Zhao et al. (2019) treat system actions as latent variables and use reinforcement learning

---

* Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 447–459
August 1–6, 2021.

to optimize them. Wang et al. (2020b) model system actions prediction as a sequence generation problem by treating system actions as a sequence of tokens. On the other hand, Zhang et al. (2020) explicitly modeling the one-to-many property to enrich system action diversity through a rule-based multi-action data augmentation. Specifically, they treat system actions that follow the same dialogue state as alternative valid actions and train them together with the reference system action. However, their data augmentation framework has two shortcomings. First, it enforces a rigid mapping between dialogue state and system actions. Dialogue state, which consists of information such as belief state and user actions, is not flexible enough to represent the whole dialogue context and thus limits the diversity of the mapped system actions. Second, they treat the mapped system actions as gold references during training which may force the model to fit noise in the mapped system actions and ultimately hinder the quality of the generated system actions.

To address the above limitations, we propose to model the one-to-many property more effectively by retrieving multiple candidate system actions and selectively taking the candidates into consideration when generating system action. We design a retrieve-and-memorize framework that consists of a context-aware neural retrieval module (CARM) and a memory-augmented multi-decoder network (MAMD). Specifically, the context-aware retrieval module uses a pre-trained language model to convert the dialogue history as well as belief state into a context representation of each sample. Multiple candidate system actions are retrieved based on the distances between the context vector and the representations of other samples in the latent space. These retrieved candidate actions are more diverse and consistent with the dialogue context since they are obtained based on a more holistic representation. Instead of treating the candidates impartially with the gold references, we encode them into a memory bank and the memory-augmented multi-decoder network can dynamically attend to the memory bank during system actions generation. Additionally, we employ a random sampling mechanism where during training, the memory bank is filled with randomly sampled system actions with a probability, which allows the model to learn to distinguish the quality of the candidates and adaptively adjust its dependence on the candidate actions.

We evaluate our model on MultiWOZ (Budzianowski et al., 2018), a large-scale multi-domain dataset for task-oriented dialogue systems. Extensive experiments and analyses are conducted to demonstrate the effectiveness of our model, and the results show that it significantly outperforms the baseline model. Our main contributions are summarized as follows:

- We propose a context-aware retrieval module that can retrieve multiple appropriate system actions given a dialogue context.
- We propose a memory-augmented multi-decoder network that can generate system actions based on multiple candidate actions.
- Our model outperforms several state-of-the-art baselines on a large-scale multi-domain dataset for task-oriented dialogue systems.

2 Related Work

One line of research focuses on the representation of system actions. A typical approach to encoding system actions is by concatenating the one-hot representation at each level of actions into a flat vector (Wen et al., 2015; Budzianowski et al., 2018). Such sparse representations make the learning of system actions difficult. To overcome the sparsity issue, Chen et al. (2019) compact the one-hot vector representation based on the intrinsic hierarchical structures of system actions, and apply hierarchical disentangled self-attention to generate system response. Zhao et al. (2019) treat system actions as latent variables and use reinforced learning (He et al., 2016) to optimize them. Recently, Wang et al. (2020b) propose a co-generation framework to generate system actions and response sequentially, which achieves a new state of the art in the context-to-response task. Our proposed framework adopts the idea of modeling belief state and system actions (Wang et al., 2020b; Liang et al., 2020) as sequences and generates the belief state, system action, and response sequentially to make better use of the intermediate supervision.

Another line of research uses data augmentation to expand the training data. Gao et al. (2020) use the paraphrase technique (Li et al., 2019; Wang et al., 2019) to generate user utterances and then expand the training set with the augmented user utterances. Zhang et al. (2020) augment system actions with a mapped dialogue state, which consists of belief state, user action, turn domain, and database search result. Such mapping is rule-based and requires user actions for the construction of dialogue state, which takes extra annotations. Both of the above approaches treat the augmented samples as equivalent to the gold ones, which may force the model to fit noises in the augmented data. In this paper, we focus on a better neural retrieval method for the alternative system actions, and instead of directly training on the augmented actions, we encode them in a memory bank as auxiliary information.

3 Methodology

To frame the problem of dialogue policy learning, we use $X_t = \{U_1, .., U_{t-1}, R_{t-1}, U_t\}$ to denote the dialogue history at turn t of a multi-turn conversation, where $U_i = u_1u_2, ...u_{m_i}$ and $R_i = r_1r_2...r_{n_i}$ are the i-th user utterance and system response, respectively. Following previous works (Zhang et al., 2020; Liang et al., 2020), we convert the belief state and system actions from a list of triples to sequences. For example, the belief state "*restaurant-food-Chinese,restaurant-price-expansive*" is converted to "*restaurant [food] Chinese [price] expansive*", and the system actions "*restaurant-inform-price,restaurant-inform-phone*" are converted to "*restaurant [inform] price phone*". We use $B_t = b_1b_2...b_p$ and $A_t = a_1a_2...a_q$ to represent the current belief state and system action, respectively. Our goal is to generate system actions A_t and system response R_t of turn t based on the dialogue context X_t and belief state B_t.

We employ a retrieve-and-memorize framework to generate the system response. First, we use a context-aware retrieve module to retrieve multiple proper candidate system actions from the training set. Then, we encode the candidate actions into a memory bank and propose a memory-augmented module to enhance the action generation.

3.1 Context-Aware Retrieval Module

In order to retrieve alternative system actions that are more comprehensive and context-aware, we utilize the powerful pre-trained language model BERT (Devlin et al., 2019) to obtain distributed representations of the dialogue context. We search in the training corpus for system actions with similar distributed representations and retrieve them as alternative candidate actions. Concretely, we combine the dialogue history $X_t = \{U_1, .., U_{t-1}, R_{t-1}, U_t\}$ and belief state B_t as dialogue context and feed the concatenated dialogue context into a pre-trained BERT encoder:

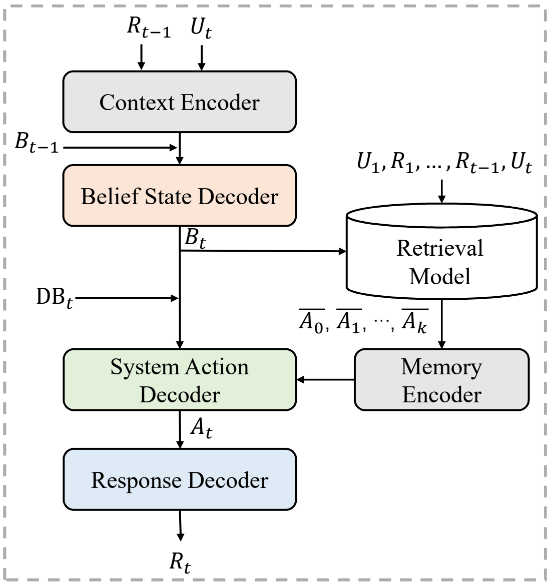

Figure 2: An overview of the proposed model.

$$H = \text{BERT}([CLS] \oplus B_t \oplus [SEP] \oplus X_t), \quad (1)$$

where $\oplus$ is the concatenation operator, $[CLS]$ is a special token that precedes every input sequence of BERT, and $[SEP]$ is a special token used to separate different parts of the input sequence. The BERT model encodes the input dialogue context into a sequence of hidden states $H = \{h^{CLS}, h^1, ..., h^L\}$. We use h^{CLS} to represent the distributed representation of dialogue context, since h^{CLS} is expected to capture the information of the whole sequence. Then we use L_2 distance to measure the similarity between the distributed representations of different dialog contexts:

$$L_2(h_i^{CLS}, h_j^{CLS}) = ||h_i^{CLS} - h_j^{CLS}||_2. \quad (2)$$

Based on the L_2 distance, k most similar dialogue contexts are selected from the training set, and the corresponding system actions constitute a candidate actions set $\{\bar{A}_1, \bar{A}_2, ..., \bar{A}_k\}$.

Pre-training Task Directly applying h^{CLS} from BERT without fine-tuning or further pre-training may not result in desired dialogue context representations that correlate well with system actions. A good dialogue contextual representation should satisfy the property that dialogue contexts with similar semantics are close to each other in the representation space. Therefore, we further pre-train the

BERT model with an actions prediction task:

$$p(y|B_t, X_t) = \text{classifier}(h^{CLS}), \tag{3}$$

where $y \in \mathbb{R}^D$ is a one-hot label of system actions (Chen et al., 2019), D is the dimension of the label space, and *classifier* is a simple linear classifier.

3.2 Memory-Augmented Multi-Decoder Network

We propose a memory-augmented multi-decoder network that jointly generates belief state, system actions, and system response while having access to a memory bank when generating the system action. Given the retrieved candidate system actions, we encode these candidates into the memory bank and enhance the generation of system actions by querying the memory bank during decoding.

Encoding Module We use Bidirectional GRUs (Chung et al., 2014) as our encoders. First, we encode the current user utterance, the previous system response and the previous belief state separately into hidden states:

$$\begin{aligned} H_u &= \text{Encoder}(U_t), \\ H_{pre\_r} &= \text{Encoder}(R_{t-1}), \\ H_{pre\_b} &= \text{Encoder}(B_{t-1}), \end{aligned} \tag{4}$$

Then, another encoder is used to encode the candidate system actions into memory bank:

$$M_t = \text{Encoder}_M(\bar{A}_1 \oplus \bar{A}_2 \oplus ... \oplus \bar{A}_k), \tag{5}$$

where $M_t = \{m_1, ..., m_k\}$.

Belief State Generation The belief state B_t of turn t is generated based on the current user utterance U_t, previous system response R_{t-1} and previous belief state B_{t-1}. The generation of B_t at each time step τ can be formulated as follows:

$$\begin{aligned} s_\tau &= \text{Attn}(h_{\tau-1}, H_u, H_{pre\_r}, H_{pre\_b}), \\ c_\tau &= [s_\tau \oplus e(b_{\tau-1})], \\ p(b_\tau|b_{1:\tau-1}), h_\tau &= \text{Dec}_b(c_\tau, h_{\tau-1}, H_{pre\_b}), \end{aligned} \tag{6}$$

where $Attn$[1] is an attention function, $e(b_{\tau-1})$ is the embedding of the previous token, $h_{\tau-1}$ is the hidden state from the last decoding step, and $h_0 = \mathbf{0}$. Dec_b[1] is the belief state decoder augmented with copy mechanism (Gu et al., 2016), which can copy tokens from the previous belief state. $p(b_\tau|b_{1:\tau-1})$ is a distribution over vocabulary. We use cross entropy between ground truth and the output distribution $\mathcal{L}_b(\theta)$ as the loss of belief state generation. We collect the hidden states $H_b = \{h_0, h_1, ..., h_p\}$ of each step to feed them into the action decoder.

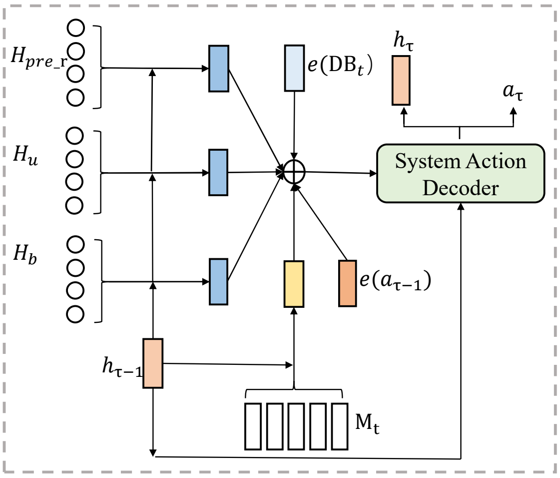

Figure 3: Generation of system actions at time step τ.

Memory-Augmented Action Generation As shown in Figure 3, the system action A_t of turn t is generated based on not only the dialog history and the current belief state, but also the memory bank which encodes the retrieved candidate system actions. For the generation of A_t, at each time step, we first compute the state s_τ:

$$s_\tau = \text{Attn}(h_{\tau-1}, H_u, H_{pre\_r}, H_b). \tag{7}$$

Then, we use the hidden state $h_{\tau-1}$ to query the encoded candidate system actions memory M_t:

$$\begin{aligned} a_\tau^i &= \tanh(W[h_{\tau-1} \oplus m_i]), \\ \alpha_\tau &= \text{Softmax}(a_\tau), \\ v_\tau &= \sum\nolimits_{i=1}^{k} \alpha_\tau^i m_i, \end{aligned} \tag{8}$$

where W are learnable parameters and v_τ contains information from the memory. Now we incorporate v_τ into the generation process:

$$\begin{aligned} c_\tau &= [s_\tau \oplus e(a_{\tau-1}) \oplus e(DB_t) \oplus v_\tau], \\ p(a_\tau|a_{1:\tau-1}), h_\tau &= \text{Dec}_a(c_\tau, h_{\tau-1}, H_b), \end{aligned} \tag{9}$$

where $e(a_{\tau-1})$ is the embedding of the previous token, $e(DB_t)$ is the embedding of the database search result which indicates the number of matched entities. Dec_a is the action decoder augmented with copy mechanism. The cross entropy $\mathcal{L}_a(\theta)$ between the output distribution and ground truth is the loss of actions generation. We

---

[1] Please refer to the appendix for more details.

collect the hidden states $H_a = \{h_0, h_1, ..., h_q\}$ as well to feed it into the system response decoder.

Random Sampling Though the retrieved candidate system actions are considered to be of high quality and suitable given the dialogue context, we would still like our model to avoid taking those candidates for granted and developing excessive dependence on them. To this end, during training, the memory bank is filled with randomly sampled system actions with a probability p, and retrieved candidates with a probability $(1 - p)$. This allows the model to learn to distinguish good candidates from bad candidates.

Response Generation Lastly, we generate the system response conditioned on the hidden states of user utterance H_u, belief state H_b and system actions H_a with the response decoder Dec_r:

$$\begin{aligned} s_\tau &= \text{Attn}(h_{\tau-1}, H_u, H_b, H_a), \\ c_\tau &= [s_\tau \oplus e(r_{\tau-1})], \\ p(r_\tau | r_{1:\tau-1}), h_\tau &= \text{Dec}_r(c_\tau, h_{\tau-1}, H_b), \end{aligned} \tag{10}$$

The response generation loss $\mathcal{L}_r(\theta)$ is the cross entropy between the output and ground truth.

Objective Function The final objective function is the sum of belief state loss, actions generation loss and response generation loss:

$$\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + \mathcal{L}_a(\theta) + \mathcal{L}_r(\theta) \tag{11}$$

4 Experiments

4.1 Dataset and Metrics

We conduct our experiments primarily on MultiWOZ 2.0 (Budzianowski et al., 2018). It consists of 8438 dialogues spanning several domains and topics. Each of the test and validation sets contains 1000 dialogues. As for automatic evaluation, we use *Inform Rate* and *Success Rate* to evaluate dialogue task completion. The former measures whether the system has provided a proper entity and the latter measures whether it has answered all the requested attributes (Budzianowski et al., 2018). Besides, *BLEU* (Papineni et al., 2002) is used to measure the fluency of generated responses. To measure the overall quality, we compute a combined score by $(\textit{Inform}+\textit{Success})\times 0.5 + \textit{BLEU}$ (Mehri et al., 2019).

4.2 Implementation Details

Our model is trained on a 12 GB Nvidia GeForce RTX 2080 Ti with a batch size of 80. Our implementation[2] is based on PyTorch (Paszke et al., 2019). We pre-trained the BERT model based on the open-source library Transformers (Wolf et al., 2020). The dimension of word embeddings is 50 and the hidden size is 100. We use one-layer Bidirectional GRUs (Chung et al., 2014) as context encoders and three GRUs augmented with copy mechanism as decoders. The candidate actions are encoded by another Bidirectional GRU. We use Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.007. We use greedy search to decode system actions and beam search with a beam size of 5 to decode system responses. We use the ground truth belief states for a fair comparison with other baselines. We train our model for 60 epochs and select the best model on the validation set, and then evaluate it on the test set to get the final results.

4.3 Baselines

We compare our full model MAMD with several baselines on MultiWOZ 2.0: SC-LSTM (Wen et al., 2015), LaRL (Zhao et al., 2019), HDSA (Chen et al., 2019) , DAMD (Zhang et al., 2020), PARG (Gao et al., 2020), SimpleTOD (Hosseini-Asl et al., 2020), MarCo (Wang et al., 2020b), UBAR (Yang et al., 2020), HDNO (Wang et al., 2020a), LAVA (Lubis et al., 2020). Especially, SC-LSTM and HDSA treat system actions as one-hot vectors, and LaRL, HDNO, LAVA treats them as latent variables. Besides, HDSA uses BERT to predict system actions. DAMD, PARG, SimpleTOD, MarCo, and UBAR treat belief state, system actions as sequences and generate them along with system response. Besides, DAMD (aug) means DAMD using rule-based multi-action data augmentation to augment the system actions. Similar to HDSA, MarCo also uses BERT to predict system actions.

4.4 Overall Results

As shown in Table 1, our model significantly outperforms the baseline model DAMD in *Inform Rate*, *Success Rate* and especially *Combined Score*. Besides, our model achieves the best performance in *Combined Score* among all the baseline models. We also observe that models that generate system actions as a sequence generally have superior performance, implying that sequence is a better representation to model the inter-relationships among dialogue actions than one-hot vectors. Besides, our

---

[2] https://github.com/yunhaoli1995/MAMD-TOD

Model	DA	LM	Inform	Success	BLEU	Combined Score
SC-LSTM (Wen et al., 2015)	✗	✗	74.50	62.50	20.50	89.00
LaRL (Zhao et al., 2019)	✗	✗	82.80	79.20	12.80	94.10
SimpleTOD (Hosseini-Asl et al., 2020)	✗	✓	88.90	67.10	16.90	94.90
HDSA (Chen et al., 2019)	✗	✓	82.90	68.90	**23.60**	99.50
DAMD (Zhang et al., 2020)	✗	✗	89.50	75.80	18.30	100.90
DAMD (aug) (Zhang et al., 2020)	✓	✗	89.20	77.90	18.60	102.15
PARG (Gao et al., 2020)	✓	✗	91.10	78.90	18.80	103.80
MarCo (Wang et al., 2020b)	✗	✓	92.30	78.60	20.02	105.47
UBAR (Yang et al., 2020)	✗	✓	94.00	83.60	17.20	106.00
LAVA (Lubis et al., 2020)	✗	✗	**97.50**	**94.80**	12.10	108.25
HDNO (Wang et al., 2020a)	✗	✗	96.40	84.70	18.85	109.37
MAMD	✓	✗	95.70	88.90	18.90	**111.20**

Table 1: Overall results on the MultiWOZ 2.0 dataset. *DA* indicates whether to use data augmentation, and *LM* indicates whether to use pre-trained language models to predict system action.

Model	Inform	Success	BLEU	Score
SimpleTOD	85.10	73.50	16.22	95.52
HDSA	86.30	70.60	**22.36**	100.81
MarCo	92.50	77.80	19.54	104.69
UBAR	92.70	81.00	16.70	103.55
LAVA	**96.39**	83.57	14.02	104.00
HDNO	92.80	83.00	18.97	106.87
MAMD	94.20	**86.20**	18.80	**109.00**

Table 2: Overall results on the MultiWOZ 2.1 dataset.

model outperforms all the methods with data augmentation, which shows the effectiveness of our proposed retrieve-and-memorize framework.

We also evaluate our model on MultiWOZ 2.1 (Eric et al., 2020), an updated version of MultiWOZ 2.0. As shown in Table 11, the results are consistent with that on MultiWOZ 2.0 in Table 1.

4.5 Performance Across Different Domains

We report the performance of our model on different domains of MultiWOZ 2.0 and compare it with DAMD and DAMD (aug). The results are shown in Figure 4. From the bar chart, we can find that our model achieves the best performance across all domains. Besides, our model achieves significant performance improvements in *taxi* and *attraction* domains, which appear less frequently in the training data than other domains. Our MAMD narrows the performance gaps among different domains.

4.6 Ablation Study

In this section, we conduct experiments to study the contributions of the proposed context-aware retrieval module and memory-augmented module.

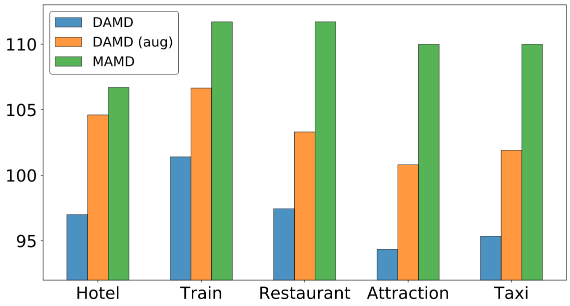

Figure 4: Results of our MAMD and DAMD in combined scores across different domains. If a dialogue involves more than one domain, it is counted into each.

As shown in Table 3, the first group is the baseline directly trained on four types of augmented data, where it treats the augmented actions as equivalent to the golden ones. We observe that the performance drops significantly if the augmented actions are randomly selected, suggesting that the benefit of such data augmentation is strongly subject to the quality of the augmented data. Additionally, the model trained with CARM outperforms the Rule, which indicates the higher quality of our context-aware retrieved candidates and the effectiveness of the proposed CARM. What's more, removing the system actions prediction pre-training task in CARM causes a performance drop, which demonstrates the necessity to adjust the pre-trained model and obtain more task-related representations.

The second group in Table 3 shows the results of the model with the memory-augmented (MA) module trained as well as evaluated with various augmented data. First, with MA, our MAMD is much more robust to random noise, only slightly under-

Method	**Score**	**Δ**
Baseline	98.95	0
+ Random	91.65	-7.30
+ Rule	102.15	+3.20
+ CARM	106.65	+7.70
+ CARM w/o Pt	102.25	+3.30
+ Random + MA	98.10	-0.85
+ Rule + MA	106.75	+7.80
+ CARM + MA	108.70	+9.75
+ CARM + MA + RS	111.20	+12.25

Table 3: Results of ablation study. *Baseline* is MAMD without the memory-augmentation component. *Random* means randomly selected actions, *Rule* is the rule-based augmentation proposed by DAMD, *CARM* is the proposed context-aware retrieval module, and *w/o Pt* means without pre-training before retrieval. *MA* is the proposed memory-augmentation module, and *RS* is the proposed random sampling technique.

performing the baseline. This is because, during training, a model with MA can learn to ignore the noises in the memory and pay less attention to the memory during evaluation. Second, we see more performance gains with MA from both rule-based and context-aware retrieved candidates, which suggests a model with MA can utilize the candidate system actions more effectively. Last but not least, with the random sampling mechanism, the performance of our full model further improves.

4.7 Effect of Random Sampling

To further analyze the effect of random sampling, we adjust the random sampling probability during training from 0 (no random sampling and all candidates are from CARM) to 1 (all candidates are randomly sampled), and evaluate MAMD with retrieved candidates and randomly sampled candidates in the memory bank. As shown in Figure 5, the first thing to notice is that without random sampling, i.e., the random sampling probability p is set to 0, the performance of MAMD with random candidate system actions drops drastically to 66.40. This indicates MAMD trained with all decent-quality candidates has developed excessive dependence on the candidates and in a way treats them as ground truth actions, which is what we try to avoid by introducing random sampling. Once we introduced random sampling, the performance gap between MAMD evaluated with retrieved actions and random actions is significantly narrowed, which suggests MAMD is capable of telling the quality of the candidates in the memory bank.

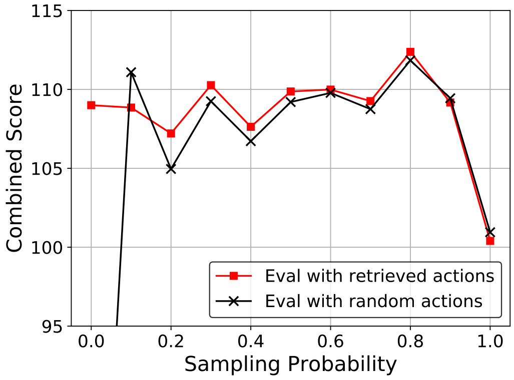

Figure 5: Results of our model trained with different random sampling probabilities and evaluated with different type of candidate actions on the development set.

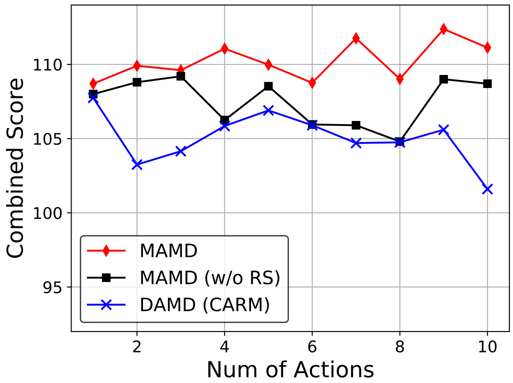

Figure 6: Combined score of three models trained with different numbers of candidate actions retrieved by CARM on the development set, where *MAMD (w/o RS)* means our model without random sampling and *DAMD (CARM)* means DAMD trained with augmented system actions retrieved by CARM.

4.8 Effect of the Number of Candidate Actions

To analyze the effect of the number of candidate actions on our proposed modules, we train three model variations with different numbers of candidate actions retrieved by CARM. As shown in Figure 6, we can see that both MAMD and MAMD (w/o RS) achieve their best performances with 9 candidate actions. Additionally, both our models consistently outperform DAMD, which suggests the effectiveness of the memory-augmented module. What's more, the performance of our full model increases more steadily as the number of candidate actions goes up, while without random sampling, the performance of our model is much more unstable across different numbers of candidate actions, which indicates that random sampling can bring in some desirable regularization.

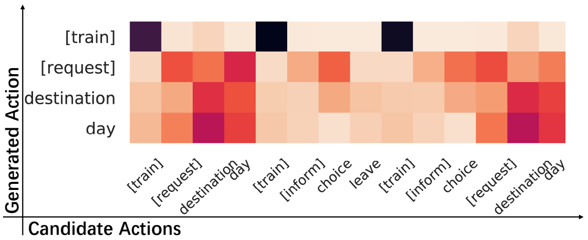

Figure 7: Visualization of the attention from generated system actions to candidate actions. The y-axis is generated system actions and the x-axis is candidate system actions. At each decoding step, the generated system actions selectively attend to the candidate actions. (Dialogue ID:MUL0473)

Context:
Sys: I would recommend the cambridge museum of technology, would you like any information about that?
User: Yes. What is the postcode and phone number?

DAMD:
[attraction] [recommend] postcode phone name type
[attraction] [inform] phone postcode
[attraction] [nooffer] type
......

CARM:
[attraction] [inform] phone postcode
[attraction] [inform] postcode phone [general] [reqmore]

Reference:
[attraction] [inform] postcode phone [general] [reqmore]

Table 4: Comparison of retrieved candidate system actions of DAMD and our CARM.

Context: ... User: Please book tickets and provide me with the total cost of tickets and confirmation number.

DAMD:
The [value_id] is **[value_price]**. The train id is [value_id]. Is there anything else I can help you with?

MAMD:
Booking was successful, the total fee is **[value_price]** payable at the station. Reference number is: **[value_reference]**. Is there anything else I can help you with?

Reference:
It has been booked! Your reference number is **[value_reference]**. The cost is **[value_price]**. Do you need anything else?

Table 5: An example of response generation of DAMD and MAMD.

MAMD vs. DAMD	**Win%**	**Tie%**	**Lose%**
Completion	19.25%	66.54%	14.21%
Readability	3.13%	93.08%	3.79%

MAMD vs. Reference	**Win%**	**Tie%**	**Lose%**
Completion	14.51%	56.11%	29.38%
Readability	2.85%	92.03%	5.11%

Table 6: Results of human evaluation on response quality. *Reference* means ground truth response. *Win*, *Tie* and *Lose* respectively indicate the proportions that our model wins over, ties with or loses to its counterpart.

5 Visualization and Case Study

An illustrative example is shown in Figure 7, the current user utterance is “*I need a train departing cambridge arriving by 20:30*”. The action decoder successfully attends to appropriate actions and ignores the noisy ones like “*[train] [inform] leave*”, as the leaving time has not provided by the user.

Table 4 shows an example of candidate system actions that CARM appropriately retrieved but DAMD failed. The user asks the system to provide the postcode and phone number of the attraction, while DAMD returns “*[attraction][nooffer][type]*”.

We also present an example of response generation in Table 5, where the user asks for the price and reference number. DAMD manages to provide the postcode but fails to provide the reference number, while our MAMD model successfully provides both the postcode and the reference number.

6 Human Evaluation

Finally, we conduct a human study to evaluate our model from the human perspective. We randomly select 30 dialogue sessions (211 dialog turns in total) from the test dataset and have 5 postgraduates as judges to compare two groups of systems: MAMD vs. DAMD and MAMD vs. Reference, in terms of *Readability* and *Completion* (Wang et al., 2020b). *Completion* measures whether a response has correctly answered a user query, including relevance and informativeness. *Readability* measures the fluency and consistency of the response.

We report the human evaluation results in Table 6, from which we can observe that our model outperforms DAMD and beats or ties with Reference nearly 70% of the time in terms of *Completion*. In *Readability*, our model ties more than 92% with DAMD as well as Reference. This may suggest the language of responses lacks diversity and is easy to learn. Overall, our model is superior to DAMD in human evaluation, which demonstrates its competence in a more holistic evaluation other than automatic metrics.

7 Conclusion

In this paper, we proposed a retrieve-and-memorize framework to deal with the unbalanced distribution of system actions in task-oriented dialogue systems. Our framework includes a neural retrieval module that can retrieve multiple candidate system actions given a dialogue context, and a memory-augmented multi-decoder network that can generate system actions conditioned on multiple candidate system actions. Extensive experiments were conducted on a large-scale multi-domain task dialogue dataset and the results demonstrate the effectiveness of our framework. In essence, the whole framework, including its random sampling strategy, can be viewed as an attempt to prevent the systems from overfitting skewed dialogue datasets with an unbalanced distribution of system actions.

Acknowledgments

The paper was supported by the National Natural Science Foundation of China (No.61906217) and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No.2017ZT07X355).

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. MultiWOZ-A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Paraphrase augmented task-oriented dialog generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 639–649, Online. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020. MOSS: End-to-end dialog system framework with modular supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8327–8335.

Nurul Lubis, Christian Geishauser, Michael Heck, Hsien-chin Lin, Marco Moresi, Carel van Niekerk, and Milica Gasic. 2020. LAVA: Latent action spaces via variational auto-encoding for dialogue policy optimization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 465–479, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. Structured fusion networks for dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 165–177.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020a. Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system. *arXiv preprint arXiv:2006.06814*.

Kai Wang, Junfeng Tian, Rui Wang, Xiaojun Quan, and Jianxing Yu. 2020b. Multi-domain dialogue acts and response co-generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7125–7134, Online. Association for Computational Linguistics.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7176–7183.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2020. UBAR: Towards fully end-to-end task-oriented dialog systems with GPT-2. *arXiv preprint arXiv:2012.03539*.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, pages 1208–1218.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1458–1467, Melbourne, Australia. Association for Computational Linguistics.

A More Details of MAMD

A.1 Attention Function

In MAMD, we use an attention function *Attn* to attend to three groups of hidden states. In this study, $Attn(h, H_a, H_b, H_c)$ is defined as:

$$\begin{aligned} h_a &= \text{CatAttn}(h, H_a), \\ h_b &= \text{CatAttn}(h, H_b), \\ h_c &= \text{CatAttn}(h, H_c), \\ \text{Attn}(h, H_a, H_b, H_c) &= [h_a \oplus h_b \oplus h_c], \end{aligned} \tag{12}$$

where $\oplus$ is the concatenation operator and CatAttn is a simple concat-attention defined as:

$$\begin{aligned} a_i &= \tanh(W[h \oplus H_i]), \\ \alpha_i &= \text{Softmax}(a), \\ \text{CatAttn}(h, H) &= \sum\nolimits_{i=1}^{n} \alpha_i H_i, \end{aligned} \tag{13}$$

where W represents learnable parameters, H is the sequence of encoded hidden states,[3] and n is the number of hidden states in H.

A.2 Decoder with Copy Mechanism

The decoder used to generate the belief state, system action and response is a one-layer GRU augmented with copy mechanism. Each step of the

---

[3] For example, the encoded hidden states of user utterance.

generation in $\text{Dec}(c_t, h_{t-1}, H)$ is defined as follows:

$$\begin{aligned}
h_t &= \text{GRU}(c_t, h_{t-1}), \\
p_{vocab} &= \text{Softmax}(W_v h_t), \\
s_i &= h_t^\top \tanh(W_c H i), \\
p_{copy} &= \text{Softmax}(s), \\
p_{final}(w) &= p_{vocab}(w) + \sum_{i:X(i)=w} p^i_{copy}, \\
\text{Dec}(c_t, h_{t-1}, H) &= p_{final}, h_t,
\end{aligned} \tag{14}$$

where W_v and W_c are learnable weights, and X is the corresponding context of H.

B More implementation Details

B.1 Hyperparameters

In this section, we report the hyperparameter setting in our model. For MAMD, we adopt the default hyperparameters in DAMD, as shown in Table 7. As for the learning rate, the number of candidate actions, and the random sampling probability, we apply grid search to find the best combination on the development set. It takes about 10 hours to train our model on a single 12 GB Nvidia GeForce RTX 2080 Ti. As for CARM's pre-training task, the hyperparameter setting is shown in Table 8.

Parameter	**Values**
batch size	80
learning rate	7e-3
embedding size	50
hidden size	100
dimension of db search result	6
encoder layers	1
decoder layers	1
epoch	60
candidate actions	9
random sampling probability	0.8
beam size	5
random seed	777

Table 7: Hyperparameter setting of MAMD.

B.2 Delexicalization Strategy

For delexicalization, we follow DAMD's domain-adaptive delexicalization strategy. Specially, we use tokens such as [value_name] to represent the same slot name. In this case, the placeholders [hotel_name] and [restaurant_name] will be converted to [value_name]. During the evaluation, we induce the domain of a placeholder from the transition between two adjacent belief states and the generated system actions of the current dialog turn.[4]

Parameter	**Values**
batch size	6
learning rate	5e-5
epoch	20
random seed	42
max sequence length	400
warmup proportion	0.1

Table 8: Hyperparameter setting of CARM.

B.3 Post-Processing of Candidate Action Retrieval

As for candidate action retrieval, we retrieve 50 candidate actions for each sample. Then, we apply post-processing to clean the candidate actions:

- Duplicated actions are merged. For example, the system actions "[attraction] [inform] postcode phone [general] [reqmore]" and "[attraction] [inform] postcode phone [general] [reqmore]" will be combined into "[attraction] [inform] postcode phone [general] [reqmore]".
- Null system actions are removed.
- System actions with different database query results are filtered out.
- System actions that conflict with current belief are filtered, e.g., requesting a slot that is already included in belief states.

C Dataset Details

We provide more information about the MultiWOZ 2.0 dataset. The training set contains 8438 dialogs, 115,424 turns, and 1,520,970 tokens. The average number of turns per dialog is 13.68, and the average number of tokens per turn is 13.18. The number of slots and values are 25 and 4510, respectively. The ontology is shown in Table 9. We also count the numbers of system actions across different domains. As shown in Figure 8, the numbers of system actions in *attraction* and *taxi* are smaller than the other domains, showing the unbalanced distribution of system actions at the domain level.

---

[4] For more details, please refer to the source code.

act type	inform* / request* / nooffer^{1234} / recommend^{123} / select^{1234} / offerbook^{124} / offerbooked^{124} / nobook^{12} / bye* / greet* / reqmore* / welcome*
slot	car^5 / address^{12367} / postcode^{12367} / phone^{123567} / internet^2 / parking^2 / type^{23} / pricerange^{12} / food^1 / stars^2 / area^{123} / reference^{1234} / time^{14} / leave^{45} / price^{45} / arrive^{45} / id^4 / stay^2 / day^{124} / leave^{45} / people^{123} / name^{123} / destination^{45} / departure^{45} / department^6

Table 9: Ontology for all domains. The upper script indicates which domains it belongs to ($*$: universal, 1: restaurant, 2: hotel, 3: attraction, 4: train, 5: taxi, 6: hospital, 7: policy).

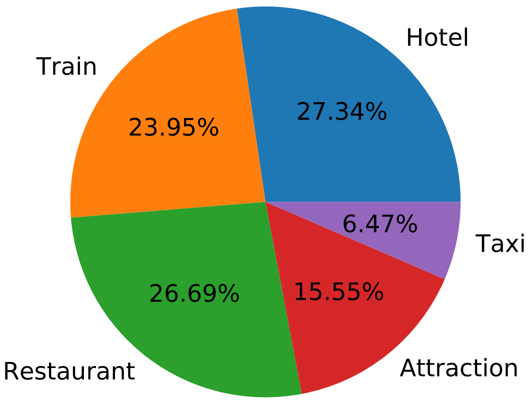

Figure 8: Statistics of system actions across different domains of MultiWOZ 2.0.

Dateset	Inform	Success	BLEU	Score
Development	96.60	90.70	18.70	112.35
Test	95.70	88.90	18.90	111.20

Table 10: Overall results on the MultiWOZ 2.0 dataset.

Dateset	Inform	Success	BLEU	Score
Development	94.90	87.70	18.60	109.90
Test	94.20	86.20	18.80	109.00

Table 11: Overall results on the MultiWOZ 2.1 dataset.

D More Analyses and Discussions

D.1 Results on Development and Test Sets

We report the results of MAMD on the development and test sets of MultiWOZ 2.0 and MultiWOZ 2.1. As shown in Table 10 and Table 11, the results on the development set are generally consistent with that on the test set on both benchmarks.

D.2 Distribution of Generated System Actions

To further analyze the influence of our model on the generation of system actions, we count the appearance of generated actions. Recall that each dimension of the actions stands for either *domain*, *function* or *slot*, where *domain* defines the domain involved in the conversation, and *function* defines the behavior of system such as informing the user or request certain information. Here we only count the first two dimensions of the actions because the third dimension appears to be less important.

As shown in Figure 9, the distribution of system actions generated by DAMD is proportional to the original distribution in the dataset, and DAMD tends to generate fewer actions than the original distribution. After applying their rule-based multi-action data augmentation, DAMD (aug) can generate more diverse system actions compared with DAMD. Compared with DAMD (aug), MAMD generates more actions. More importantly, MAMD generates more important actions such as “attraction-inform” and “taxi-inform” which are more relevant to task completion, while DAMD (aug) tends to generate less useful actions such as “general-require” and “general-greet”. This phenomenon indicates that the memory-augmented mechanism provides some guidance to our model during system action learning. To sum up, our proposed model can generate more diverse and valuable actions, which demonstrates the effectiveness of our proposed memory-augmented mechanism.

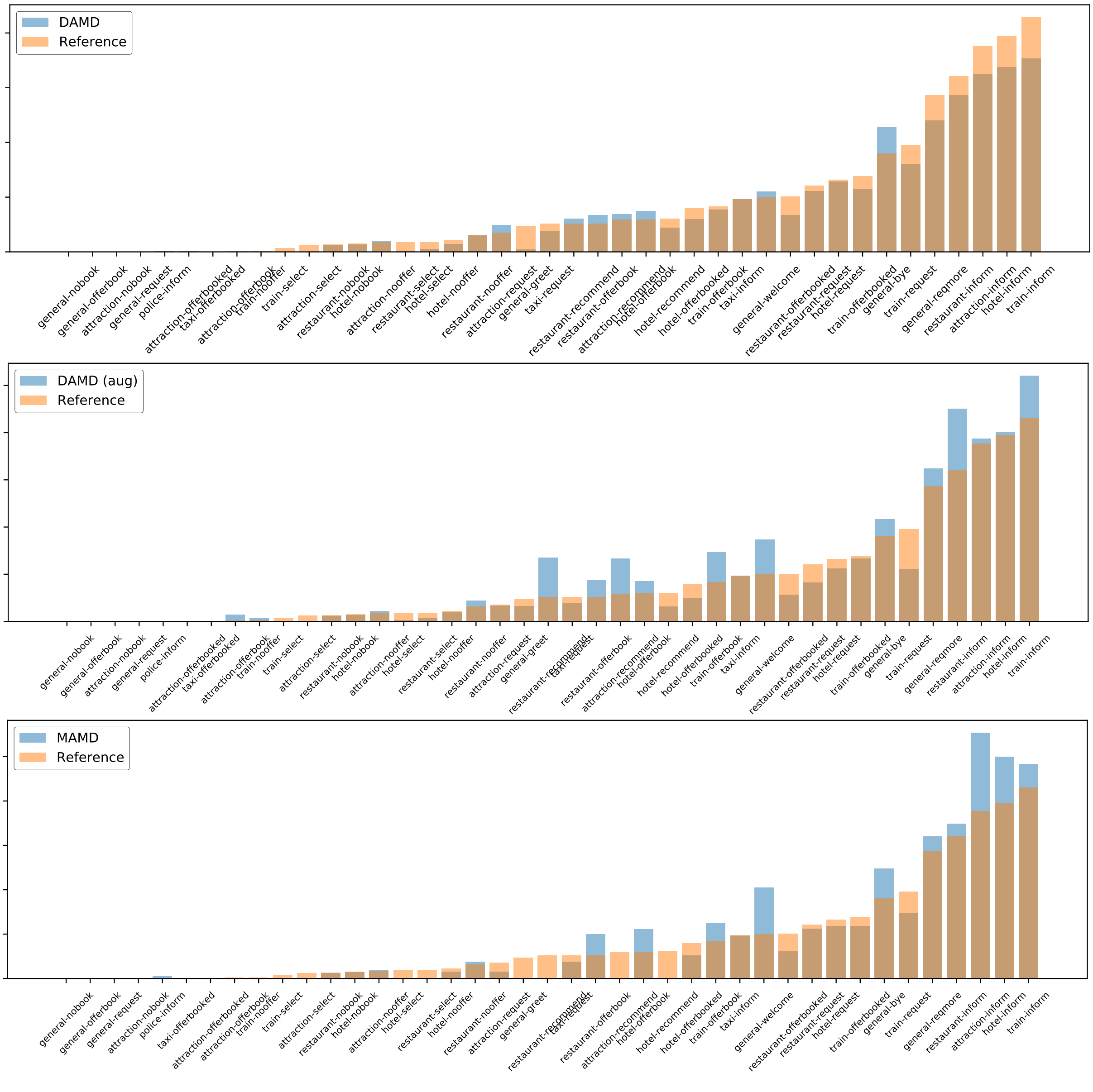

Figure 9: Statistics of generated system actions by DAMD, DAMD (aug) and MAMD, and comparison with reference actions.

Adapt-and-Distill: Developing Small, Fast and Effective Pretrained Language Models for Domains

Yunzhi Yao[†][*], **Shaohan Huang**[‡], **Wenhui Wang**[‡], **Li Dong**[‡], **Furu Wei**[‡]
[†]Shandong University, Jinan, China
[‡]Microsoft Research, Beijing, China
`yyz@mail.sdu.edu.cn`
`{shaohanh,wenhui.wang,lidong1,fuwei}@microsoft.com`

Abstract

Large pretrained models have achieved great success in many natural language processing tasks. However, when they are applied in specific domains, these models suffer from domain shift and bring challenges in fine-tuning and online serving for latency and capacity constraints. In this paper, we present a general approach to developing small, fast and effective pretrained models for specific domains. This is achieved by adapting the off-the-shelf general pretrained models and performing task-agnostic knowledge distillation in target domains. Specifically, we propose domain-specific vocabulary expansion in the adaptation stage and employ corpus level occurrence probability to choose the size of incremental vocabulary automatically. Then we systematically explore different strategies to compress the large pretrained models for specific domains. We conduct our experiments in the biomedical and computer science domain. The experimental results demonstrate that our approach achieves better performance over the $\text{BERT}_{\text{BASE}}$ model in domain-specific tasks while 3.3× smaller and 5.1× faster than $\text{BERT}_{\text{BASE}}$. The code and pretrained models are available at `https://aka.ms/adalm`.

1 Introduction

Pre-trained language models, such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and UniLM (Dong et al., 2019) have achieved impressive success in many natural language processing tasks. These models usually have hundreds of millions of parameters. They are pre-trained on a large corpus of general domain and fine-tuned on target domain tasks. However, it is not optimal to deploy these models directly to edge devices in specific domains. First, heavy model size and high latency makes it difficult to deploy on resource-limited edge devices such as mobile phone. Second, directly fine-tuning a general pre-trained model on a domain-specific task may not be optimal when the target domain varies substantially from the general domain. Thirdly, many specialized domains contain their own specific terms, which are not included in pre-trained language model vocabulary.

(a) From scratch (b) Distill-then-Adapt

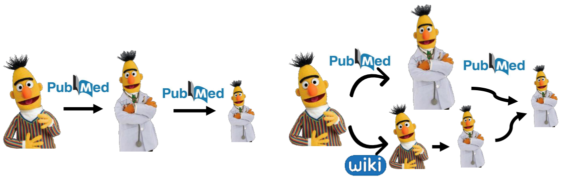

(c) Adapt-then-Distill (d) Adapt-and-Distill

Figure 1: The four alternatives when distilling BERT into specific domains. All strategies are task-agnostic.

In this paper, we introduce AdaLM, a framework that aims to develop small, fast and effective pretrained language models for specific domains. To address domain shift problem, recent studies (Lee et al., 2020; Gururangan et al., 2020) conduct continual pre-training to adapt a general domain pretrained model to specific domains. However, specific domains contain many common in-domain terms, which may be divided into bite-sized pieces (e.g., *lymphoma* is tokenized into [*l, ##ym, ##ph, ##oma*]). Gu et al.(2020) mentions that domain-specific vocabularies play a vital role in domain adaptation of pre-trained models. Specifically, we propose a domain-specific vocabulary expansion in the adaptation stage, which augments in-domain terms or subword units automatically given in-domain text. Also, it is critical to decide the size of incremental vocabulary. Motivated by subword reg-

---

[*] Contribution during internship at Microsoft Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 460–470
August 1–6, 2021.

ularization (Kudo, 2018), AdaLM introduces a corpus occurrence probability as a metric to optimize the size of incremental vocabulary automatically.

We systematically explore different strategies to compress general BERT models to specific domains (Figure 1): (a) From scratch: pre-training domain-specific small model from scratch with domain corpus; (b) Distill-then-Adapt: first distilling large model into small model, then adapting it into a specific domain; (c) Adapt-then-Distill: first adapting BERT into a specific domain, then distilling model into small size; (d) Adapt-and-Distill: adapting both the large and small models, then distilling with these two models initializing the teacher and student models respectively.

We conduct experiments in both biomedical and computer science domain and fine-tune the domain-specific small models on different downstream tasks. Experiments demonstrate that Adapt-and-Distill achieves state-of-the-art results for domain-specific tasks. Specifically, the 6-layer model of 384 hidden dimensions outperforms the $\text{BERT}_{\text{BASE}}$ model while 3.3× smaller and 5.1× faster than $\text{BERT}_{\text{BASE}}$.

2 Related Work

Domain adaptation of pre-trained model Most previous work on the domain-adaptation of pre-trained models targets large models. Lee et al. (2020) conduct continual pre-training to adapt the BERT model to the biomedical domain using the PubMed abstracts and the PMC full text. Gururangan et al. (2020) also employ continual pre-training to adapt pre-trained models into different domains including biomedical, computer science and news. However, many specialized domains contain their own specific words that are not included in pre-trained language model vocabulary. Gu et al.(2020) propose a biomedical pre-trained model PubMedBERT, where the vocabulary was created from scratch and the model is pre-trained from scratch. Furthermore, in many specialized domains, large enough corpora may not be available to support pre-training from scratch. Zhang et al. (2020) and Tai et al. (2020) extend the open-domain vocabulary with top frequent in-domain words to resolve this out-of-vocabulary issue. This approach ignores domain-specific sub-word units (e.g., *blasto-*, *germin-* in biomedical domain). These subword units help generalize domain knowledge and avoid unseen words.

Task-agnostic knowledge distillation In recent years, tremendous progress has been made in model compression (Cheng et al., 2017). Knowledge distillation has proven to be a promising way to compress large models while maintaining accuracy (Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020; Wang et al., 2020). In this paper, we focus on task-agnostic knowledge distillation approaches, where a distilled small pre-trained model can be directly fine-tuned on downstream tasks. DistilBERT (Sanh et al., 2019) employs the soft label and embedding outputs to supervise the student. TinyBERT (Jiao et al., 2020) and MobileBERT (Sun et al., 2020) introduce self-attention distributions and hidden states to train the student model. MiniLM (Wang et al., 2020) avoids restrictions on the number of student layers and employs the self-attention distributions and value relation of the teacher's last transformer layer to supervise the student model. Because this method is more flexible, we implement MiniLM to compress large models in this work. No previous work systematically explores different strategies to achieve an effective and efficient smaller model in specific domains.

3 Methods

3.1 Overview

We systematically explore different strategies to achieve an effective and efficient small model in specific domains. We summarize them into four strategies: from scratch, distill-then-adapt, adapt-then-distill and adapt-and-distill.

Pretrain-from-scratch Domain-specific pretraining from scratch employs a random initialization of a pretrained model and pretrains a small model directly on domain-specific corpus. In this work, we conduct pretraining from scratch on different vocabularies including BERT original vocabulary, from scratch vocabulary, and expanded vocabulary.

Distill-then-adapt These approaches first distill the large general pretrained model which pretrained on Wikipedia and BookCorpus. Then it continues the pretraining process using a domain-specific corpus. In this work, we first distill the BERT model into a small model using task-agnostic knowledge distillation in MiniLM (Wang et al., 2020). Then we initialize the small model with it and conduct continual training with both the BERT original vocabulary and the expanded vocabulary.

Adapt-then-distill In this work, we select different large models as teacher models such as BERT and large models with different vocabularies. We first adapt these models into domain-specific models and then implement MiniLM to compress them to small models.

Adapt-and-distill In the previous part, when doing knowledge distill, we initialized the student model randomly. In order to get a better domain-specific small model, we try to explore the impact of the initialization of the student model. In this part, we adapt large and small models into specific domains separately, then use these two models to initialize the teacher and student model respectively.

3.2 Domain Adaptation

AdaLM contains a simple yet effective domain adaptation framework for a pretrained language model. As shown in Figure 2, it takes a general pretrained language model, original vocabulary and a domain specific corpus as input. Through vocabulary expansion and continual pretraining, AdaLM adapts general models into specific domains.

The core pipeline of domain adaptation consists of the three steps described below:

1. Given original vocabulary and a domain-specific corpus, the vocabulary expansion module aims to augment original vocabulary with domain-specific subword units or terms. We augment domain-specific vocabulary from the target domain, while keeping the original BERT vocabulary unchanged. We describe them in more detail in Section 3.3.
2. Due to the size of the vocabulary having changed, we cannot initialize our model with BERT directly. As illustrated in Figure 3, we initialize the original embedding and Transformer encoder with weights from BERT (the green part in Figure 3). For incremental vocabulary, we first tokenize them into sub-words with the original vocabulary and then use an average pooling of their own sub-words embedding to initialize. As shown in Figure 3, the word '*lymphoma*' is not included in BERT vocabulary. We tokenize it into three sub-words (*lym*, *##pho*, *##ma*). The embedding vector of 'lymphoma' is initialized by the average embedding vector of '*lym*', '*##pho*'and '*##ma*'.
3. After model initialization and data preprocessing, we continually pretrain our model with domain-specific corpus using masked language model loss. Following BERT, we randomly replace 15% of tokens by a special token (e.g., [MASK]) and ask the language model to predict them in continual pretraining.

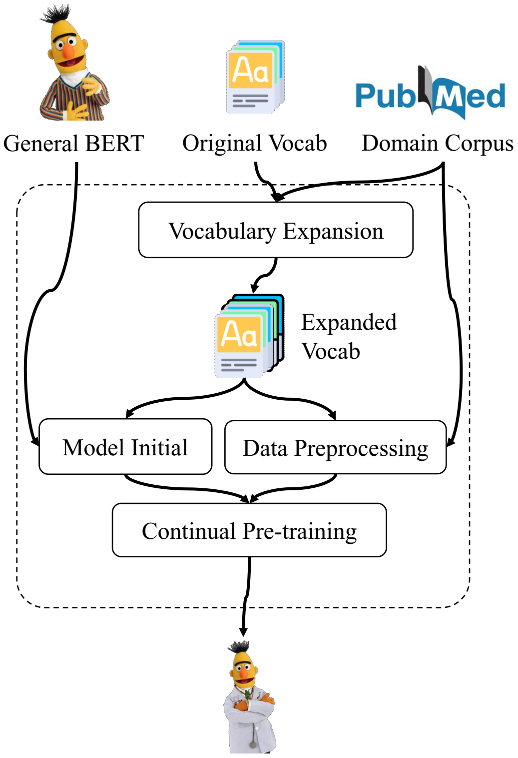

Figure 2: The pipeline of domain adaptation. Here we adapt the BERT model into the biomedical domain with PubMed dataset.

3.3 Vocabulary Expansion

Vocabulary expansion is the core module of AdaLM. It augments domain-specific terms or subword units to leverage domain knowledge. The size of the incremental vocabulary is a vital parameter for vocabulary expansion. Considering that unigram language modeling (Kudo, 2018) aligns more closely with morphology and avoids problems stemming from BPE's greedy construction procedure, as proposed in (Bostrom and Durrett, 2020), we followed Kudo (2018) and introduced a corpus occurrence probability as a metric to optimize the size of incremental vocabulary automati-

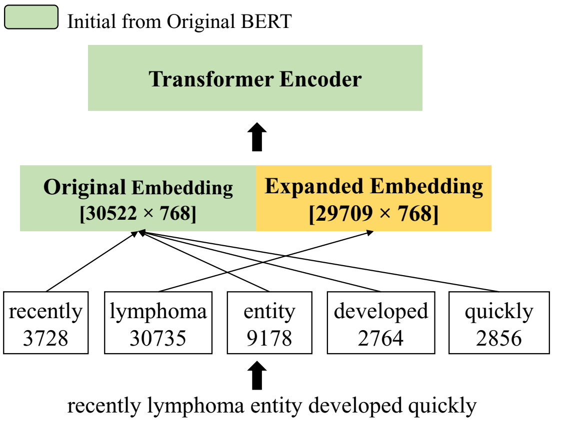

Figure 3: Concatenate original embedding with expanded embedding.

cally. We assume that each subword occurs independently and we assign to each subword in the corpus a probability equal to its frequency in the corpus.

$$\forall i \; x_i \in \mathcal{V}, \quad \sum_{x_i \in \mathcal{V}} p(x_i) = 1, \tag{1}$$

where $\mathcal{V}$ is a pre-determined vocabulary. The probability of a subword sequence $\mathbf{x} = (x_1, \dots, x_M)$ can be computed by the product of the subword appearance probabilities $p(x_i)$. We convert it to logarithmic form:

$$P(\mathbf{x}) = \sum_{i=1}^{M} log(p(x_i)), \tag{2}$$

Given a domain-specific corpus D, the occurrence probability of corpus D is formulated as:

$$P(D) = \sum_{\mathbf{x}}^{|D|} log(P(\mathbf{x})), \tag{3}$$

where $\mathbf{x}$ represents tokenized sentence in corpus D.

We sample 550k sentences from the PubMed corpus and compute the occurrence probability $P(D)$ with different vocabulary sizes. The results are shown in Figure 4. We compare the occurrence probability with BERT and PubMedBERT vocabularies. We observe that $P(D)$ reveals a logarithmic trend with substantial increases at the beginning and little influence after vocabulary size of 70k in the biomedical domain. The PubMedBERT vocabulary performs similarly to the 40k size vocabulary. We present the occurrence probability of different vocabulary sizes in Appendix A.

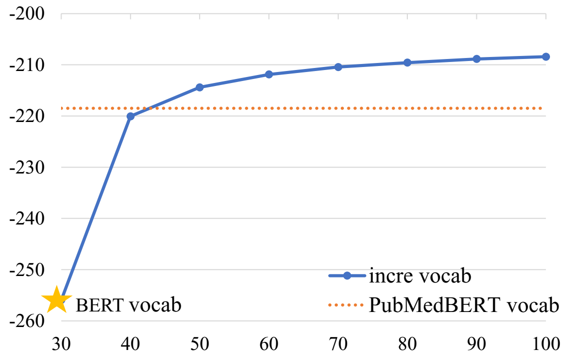

Figure 4: The $P(D)$ of different vocab sizes under biomedical domain. We use the BERT's vocabulary as the 30k vocabulary without vocabulary expanding. The PubMedBERT vocabulary is also 30k.

We propose a simple method to decide the size of the incremental vocabulary. Assume the probability at the time step $i-1$ is $P_{i-1}(D)$ and at the time step i is $P_i(D)$. If the rise $\frac{P_i(D)-P_{i-1}(D)}{P_{i-1}(D)}$ is lower than a threshold δ, we regard the vocabulary size at the time step i as the final size.

```
Algorithm 1: Vocabulary Expansion
  Input: Original vocabulary $raw\_vocab$,
         domain corpora $D$, threshold $\delta$ and
         vocabulary size step $V_\triangle$
  Output: $vocab_{final}$
  $token\_count \leftarrow$ whitespace split from $D$;
  $P_0 \leftarrow$ computed from $raw\_vocab$;
  $V_0 \leftarrow |raw\_vocab|$;
  do
      vocabulary size $V_i \leftarrow V_{i-1} + V_\triangle$;
      $sub\_count \leftarrow$ split token to subwords;
      Sort $sub\_count$ by frequency;
      $incr\_vocab \leftarrow$keep $(V_i - V_0)$ subwords;
      $vocab_i \leftarrow raw\_vocab + incr\_vocab$;
      $P_i \leftarrow$ computed from $vocab_i$
  while $\frac{P_i - P_{i-1}}{P_{i-1}} > \delta$;
  return $vocab_{final} \leftarrow vocab_i$ ;
```

We expand the domain-specific vocabulary with the process shown in Algorithm 1. We implement our vocabulary expansion algorithm referring to SubwordTextBuilder in tensor2tensor[1]. In experiments, we set the threshold δ as 1% and vocabulary size step $V_\triangle$ as 10k. Finally, we obtain the expanded vocabulary size of biomedical as 60k and computer science domain as 50k.

---

[1] https://github.com/tensorflow/tensor2tensor

4 Experiment Details

We conduct our experiments in two domains: biomedical and computer science.

4.1 Datasets

Domain corpus: For the biomedical domain, we collect a 16GB corpus from PubMed[2] abstracts to adapt our model. We use the latest collection and pre-process the corpora with the same process as PubMedBERT (we omit any abstracts with less than 128 words to reduce noise.).

For the computer science domain, we use the abstracts text from the arXiv[3] Dataset. We select abstracts in computer science categories, collecting 300M entries for the corpus.

Fine-tuning tasks: For the biomedical domain, we choose three tasks: named entity recognition (NER), evidence-based medical information extraction (PICO), and relation extraction (RE). We perform entity-level F1 in NER task and word-level macro-F1 in the PICO task. The RE task uses the micro-F1 of positive classes evaluation. JNLPBA (Collier and Kim, 2004) NER dataset contains 6,892 disease mentions, which are mapped to 790 unique disease concepts with BIO tagging (Ramshaw and Marcus, 1995). EBM PICO (Nye et al., 2018) datasets annotates text spans with four tags: Participants, Intervention, Comparator and Outcome. ChemProt (Krallinger et al., 2017) dataset consists of five interactions between chemical and protein entities. We list the statistics of those tasks in Table 1.

We fine-tune two downstream tasks in the computer science domain. They are both classification tasks. The ACL-ARC (Jurgens et al., 2018) dataset mainly focuses on analyzing how scientific works frame their contributions through different types of citations. SCIERC (Luan et al., 2018) dataset includes annotations for scientific entities, their relations, and coreference clusters. The statistics are available in Table 2.

4.2 Implementation

We use the uncased version of $\text{BERT}_{\text{BASE}}$ (12 layers, 768 hidden size) as the large model and the MiniLM (6 layers, 384 hidden size) as the small model.

Dataset	Train	Dev	Test
JNLPBA	46,750	4551	8,662
EBM PICO	339,167	85321	16,364
ChemProt	18,035	11268	15,745

Table 1: Biomedical dataset used in our experiment. All selected from BLURB[4]

Dataset	Train	Task	Test	Classes
ACL-ARC	1,688	114	139	6
SCIERC	3,219	455	974	7

Table 2: Computer science dataset used in our experiment. We use the same train, development, and test splits as Gururangan et al. (2020)

To adapt the large model, we set the batch size at 8192 and the training step at 30,000. The peak learning rate was set to 6e-4. To adapt the small model, we set the batch size as 256 and the training step as 200,000. The learning rate is set to 1e-4. The maximum length of the input sequence was 512 and the token masking probability was 15% for both the large model and the small model.

We implement MiniLM to compress large models and follow the setting of MiniLM, where the batch size was set to 256 and peak learning rate as 4e-4. We set the training step as 200,000.

For biomedical tasks, we follow the setting of PubMedBERT (Gu et al., 2020) to fine-tune these three tasks. For computer science tasks, we use the same setting as Gururangan et al. (2020). The concrete parameters are shown in Appendix B.

5 Results

The results of the tasks are shown in the Table 3 and 4. We structure our evaluation by stepping through each of our three findings:

(1) Domain-specific vocabulary plays a significant role in domain-specific tasks and expanding vocabulary with the general vocabulary is better than just using domain-specific vocabulary.

We observe improved results via the expanded vocabulary with both the large and small models. For large model, AdaLM achieves the best results under each domain, where 77.74 on biomedical domain tasks, beating BioBERT and PubMedBERT and 77.76 on the computer science domain tasks.

For small models, in the biomedical domain, whether we train from scratch or distill-then-adapt with small models, incremental vocabulary mod-

---

[2] https://pubmed.ncbi.nlm.nih.gov/

[3] https://www.kaggle.com/Cornell-University/arxiv

[4] https://microsoft.github.io/BLURB/

Config	Type	Model	Teacher	JNLPBA	PICO	Chemprot	Average
L=12; d=786	Large model	BERT†	-	78.63	72.34	71.86	74.28
		BioBERT†	-	79.35	73.18	76.14	76.22
		PubMedBERT†	-	**80.06**	73.38	77.24	76.89
		AdaLM$^\diamondsuit$	-	79.46	**75.47**	**78.41**	**77.74**
L=6; d=384	Small model	MiniLM	-	77.44	71.69	68.08	72.40
	From scratch	BERT vocab (a)	-	77.89	72.97	70.21	73.69
		PubMed vocab (b)	-	77.82	73.82	70.32	73.99
		AdaLM vocab (c)	-	77.80	73.39	70.86	74.02
	Distill-then-Adapt	BERT vocab (d)	-	78.63	74.00	72.28	74.97
		PubMed vocab (e)	-	78.36	73.91	71.33	74.53
		AdaLM vocab (f)	-	78.77	74.23	**72.29**	75.10
	Adapt-then-Distill	Random initial (g)	BERT	77.98	72.38	68.86	73.07
		Random initial (h)	PubMedBERT	78.78	74.20	70.89	74.62
		Random initial (i)	AdaLM$^\diamondsuit$	78.98	74.78	71.51	75.09
	Adapt-and-Distill	Model (f) initial (j)	AdaLM$^\diamondsuit$	**79.04**	**74.91**	72.06	**75.34**

Table 3: Comparison between different strategies on biomedical tasks. The AdaLM$^\diamondsuit$ means we just adapt the large model without distillation. Scores of the methods marked with † are taken from (Gu et al., 2020). Underlined data marks the small models whose performances surpass the BERT model's performance. L and d indicate the number of layers and the hidden dimension of the model.

els always perform better than the general vocabulary or just the domain-specific vocabulary. (When distill-then-adapt with the PubMed vocabulary, we initialize the word embedding in the same way as mentioned in Section 3.2). In addition, with distill-then-adapt, the model (f) (75.10) can surpass the BERT model (74.28).

In the computer science domain, distill-then-adapt models with incremental vocabulary also show great performance. Model (d) achieves a comparable result of 72.91 as BERT and outperforms BERT in the ACL-ARC datasets with 65.93 (+1.01 F1). We also observe that when training from scratch, the results of Model (b) with incremental vocabulary are lower (1.45 lower) than that of model (a). This may be because after vocabulary expansion, a from-scratch model needs to be pretrained with more unlabeled data.

(2) Continual pretraining on domain-specific texts from general language models is better than pretraining from scratch.

Gu et al. (2020) finds that for domains with abundant unlabeled texts, pretraining language models from scratch outperforms continual pretraining of general-domain language models. However, in our experiments, we find that general-domains model can help our model to learn the target domain better. In the biomedical domain, we use MiniLM model to initialize the model (d), (e) and (f) in distill-then-adapt setting. No matter which vocabulary is used, continual pretraining on domain-specific texts from general language models is better than pretraining from scratch. For AdaLM vocabulary, the model (f) gets 75.10, outperforming the model (c) trained from scratch with the same vocabulary by 1.08. On the other hand, for domains that do not have enormous unlabeled texts such as the computer science domain in our experiments, continual pretraining also showed better results. With continual pretraining, model (d) achieves higher results exceeding both model (b) (+5.66 F1) and model (c) (+0.47 F1).

(3) Adapt-and-Distill is the best strategy to develop a task-agnostic domain-specific small pretrained model.

In the Adapt-then-Distill part, our findings supports evidence from previous observations (Wang et al., 2020) that a better teacher model leads to a better student model. Using AdaLM which performs best among large models as the teacher model can yield good results: 75.09 in the biomedical domain and 71.62 in the computer science, better than other domain-specific large models. Furthermore, we find that a better student model for initialization can also help to get a better small model. In the Adapt-and-Distill part, we adapt large and small models into specific domains separately and then compress the adapted large model as the teacher with the adapted small model as initialization. In the biomedical domain, the model (j), initialized from model (i), achieves the best result of 75.34 among the small models. It also

Config	Type	Model	Teacher	ACL-ARC	SCIERC	Average
L=12; d=786	Large model	BERT	-	64.92	81.14	73.03
		AdaLM$^\diamondsuit$	-	**73.61**	**81.91**	**77.76**
L=6; d=384	Small model	MiniLM	-	61.5	72.88	67.19
	From scratch	BERT vocab (a)	-	62.48	74.93	68.70
		AdaLM vocab (b)	-	59.57	74.93	67.25
	Distill-then-Adapt	BERT vocab (c)	-	65.75	79.13	72.44
		AdaLM vocab (d)	-	65.93	**79.88**	72.91
	Adapt-then-Distill	Random initial (e)	BERT	63.12	77.89	70.50
		Random initial (f)	AdaLM$^\diamondsuit$	66.21	77.04	71.62
	Adapt-and-Distill	Model (d) initial (g)	AdaLM$^\diamondsuit$	**68.74**	78.88	<u>**73.81**</u>

Table 4: Comparison between different strategies on computer science tasks. The AdaLM$^\diamondsuit$ is the adapted large model without compressing. We report averages across five random seeds. Data marked with underlines are the results of small models which outperform the BERT model's. L and d indicate the number of layers and the hidden dimension of the model.

outperforms the BERT model (+1.06 F1). In the computer science domain, model (g), initialized by model (d), is the only small model that outperforms BERT (+0.78 F1).

6 Analysis

6.1 Inference Speed

We compare AdaLM's parameters' size and inference speed with the BERT model in the biomedical domain in Table 5.

Type	Model	#Params	Speedup
Large	BERT	109M	×1.0
	PubMedBERT	109M	×1.0
	AdaLM vocab	132M	×1.07
Small	BERT vocab	22M	×5.0
	AdaLM	34M	×5.1

Table 5: Comparison of model's parameter size and the inference speed. The inference speedup is computed by the classification task ChemProt and evaluated on a single NVIDIA P100 GPU.

First we can find that the vocabulary expansion yields marginal improvements on the model's inference speed. We added about 20M parameters in the embedding weights in the large model using AdaLM vocabulary, but its inference speed is slightly faster than BERT and PubMedBERT. Since most domain-specific terms are shattered into fragmented subwords, the length of the token sequence we get by using the incremental vocabulary is shorter than the length of the sequence got by the original vocabulary, which reduces the computation load. We list the change of the sequence length of the downstream tasks in Appendix C. Meanwhile, in the embedding layers, the model just needs to map the sub-words' id to their dense representations, which is little affected by the parameters' size. The small model shows the same trend.

In addition, the small model AdaLM shows great potential. Compared with the 12-layer model of 768 hidden dimensions, the 6-layer model of 384 hidden dimensions is 3.3x smaller and 5.1x faster in the model efficiency, while performing similarly to or even better than BERT$_{\text{BASE}}$.

6.2 Impact of Training Time

Pre-training often demands lots of time. In this section, we examine the adapted model's performance as a function of training time. Here we use the biomedical domain since its unlabelled texts are abundant and compare the large domain-specific adapted model with BioBERT. For every 24 hrs of continual pre-training, we fine-tuned the adapted model on the downstream tasks. For comparison, we convert the training time of BioBERT to the time it may take with the same computing resource of this work (16 V100 GPUs).

We list the results in Table 6, we denote the large adapted model as AdaLM in the table. AdaLM at 0 hrs means that we fine-tune the initialized model directly without any continual pre-training. We find that BERT is slightly better than 0hr AdaLM and after 24 hrs, AdaLM outperforms BioBERT, which demonstrates that domain-specific vocabulary is very critical for domain adaption of pre-trained model. Our experiments demonstrate promising re-

Model	Training Time	Average
AdaLM	0 hrs	74.25
	24 hrs	76.80
	48 hrs	77.36
	72 hrs	77.74
BERT	0 hrs	74.28
BioBERT	120 hrs	76.22

Table 6: Results with different pre-training time. In the table, AdaLM is the adapted large model without compressing.

sults in the biomedical domain. Under constrained computation, AdaLM achieves better performance compared to BioBERT. More details can be found in Appendix D

6.3 Impact of Vocabulary Size

To understand the impact of the vocabulary size, we conduct some experiments with different vocabulary sizes in the biomedical domain. We select the biomedical large AdaLM model and to reduce the computation load, we set the batch size as 256 and step as 250K in our ablation studies. We show performance of the model with different sizes in Table 7.

	40k	50k	60k	70k	80k
JNLPBA	78.84	**79.02**	78.91	78.94	79.01
PICO	**75.09**	74.81	74.99	74.58	75.00
ChemProt	76.10	76.80	**77.21**	76.40	76.85
Average	76.67	76.87	**77.03**	76.64	76.95

Table 7: The performance of different vocabulary sizes

We observe that the model of 60k achieves the best results in our ablation studies. The result is a bit surprising. Despite having a larger vocabulary, the 70k and 80k model does not show a stronger performance. A possible explanation for these results may be that a larger vocabulary set may contain some more complicated but less frequent words, which cannot be learnt well through continual pre-training. For example, the word *ferrocytochrome* exists in 70k and 80k vocabularies but is split into (*'ferrocy', '##tochrom', '##e'*) in the 60k vocabulary. In our sampled data (about 550k sentences), '*ferrocytochrome*' appears less than 100 times, while the subword '*##tochrom*' appears more than 10k times and '*ferrocy*' appears more than 200 times. The representation of those rare words cannot be learnt well due to the sparsity problem.

6.4 Vocabulary Visualization

The main motivation for using an the expanded vocabulary set is to leverage domain knowledge better. Compared to PubMedBERT which just uses the domain-specific vocabulary and initializes the model randomly, the keep of the general vocabulary and the general language model's weights may help us make good use of the existing knowledge and word embedding.

To assess the importance of the expanded vocabulary, we compute the $L2$-distance of the embedding weights before and after pre-training in our AdaLM model in the biomedical domain in Figure 5.

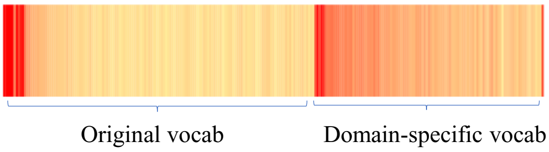

Figure 5: The $L2$-distance of the embedding layer. The deeper the color, the farther the distance.

We observe that the domain-specific vocabulary part changes a lot during the pre-training time, which indicates that our model learns much information about these domain-specific terms. We also observe that there is little change in many original sub-words' embedding weights, which indicates that many general vocabularies can be used directly in continual training.

7 Conclusion

In this paper, we investigate several variations to compress general BERT models to specific domains. Our experiments reveal that the best strategy to obtain a task-agnostic domain-specific pretrained model is to adapt large and small models into specific domains separately and then compress the adapted large model with the adapted small model as initialization. We show that the adapted 6-layer model of 384 hidden dimensions outperforms the BERT$_{\text{BASE}}$ model while 3.3× smaller and 5.0× faster than BERT$_{\text{BASE}}$. Our findings suggest that domain-specific vocabulary and general-domain language model play vital roles in domain adaptation of a pretrained model. In the future, we will investigate more directions in domain adaptation, such as data selection and efficient adaptation.

References

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *arXiv preprint arXiv:2007.15779*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Martin Krallinger, Obdulia Rabal, Saber A Akhondi, Martın Pérez Pérez, Jesús Santamaría, GP Rodríguez, et al. 2017. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, volume 1, pages 141–146.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain Marshall, Ani Nenkova, and Byron Wallace. 2018. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 197–207, Melbourne, Australia. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Wen Tai, HT Kung, Xin Luna Dong, Marcus Comiter, and Chang-Fu Kuo. 2020. exbert: Extending pre-trained models with domain-specific vocabulary under constrained training resources. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1433–1439.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Rong Zhang, Revanth Gangi Reddy, Md Arafat Sultan, Vittorio Castelli, Anthony Ferritto, Radu Florian, Efsun Sarioglu Kayi, Salim Roukos, Avi Sil, and Todd Ward. 2020. Multi-stage pre-training for low-resource domain adaptation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5461–5468, Online. Association for Computational Linguistics.

A Occurrence probability of different vocabulary sizes

Vocabulary	P(D)
BERT	-255.92
PubMed	-218.49
40k vocab	-220.06
50k vocab	-214.40
60k vocab	-211.88
70k vocab	-210.44
80k vocab	-209.57
90k vocab	-208.86
100k vocab	-208.42

Table 8: The $P(D)$ of different vocabulary under biomedical domain.

Vocabulary	P(D)
BERT	-211.14
40k vocab	-194.08
50k vocab	-192.56
60k vocab	-191.87
70k vocab	-191.45
80k vocab	-191.09
90k vocab	-190.76
100k vocab	-190.53

Table 9: The $P(D)$ of different vocabulary under computer science domain.

B Fine-tuning hyperparameters for downstream tasks

Hyperameter	Assignment		
	NER	PICO	RE
Batch size	32	{16,32}	32
Learning rate	{1e-5,3e-5,5e-5}		
Epoch	{30-40}	{10,15}	{40-50}
Dropout	0.1		

Table 10: Hyparameters we used to finetune on biomedical tasks.

C Sequence Length

After the vocabulary expansion, the length of the token sequence may get shorter. We compute the average sentence length of the downstream tasks. We list the results in Table 12

Hyperameter	Assignment	
	ACL-ARC	SCIERC
Batch size	16	
Learning rate	2e-5	
Epoch	20	
Dropout	0.1	

Table 11: Hyparameters we used to finetune on computer science tasks.

Dataset	Original Vocab	Incr. Vocab
ChemProt	66	53
EBM PICO	36	31
JNLPBA	41	32
ACL-ARC	53	50
SCIERC	45	42

Table 12: The sequence length tokenized by the original vocabulary and expanded vocabulary.

D Results of different training time

We list the biomedical tasks' results of each pretraining time in the following table.

	0h	24h	48h	72h
JNLPBA	77.56	79.14	79.11	79.46
PICO	73.29	74.22	75.28	75.36
ChemProt	71.91	77.06	77.69	78.42
Average	74.25	76.80	77.36	77.74

Table 13: The performance of models with different pretraining time

Decoupling Adversarial Training for Fair NLP

Xudong Han **Timothy Baldwin** **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia
xudonghl@student.unimelb.edu.au {tbaldwin,tcohn}@unimelb.edu.au

Abstract

Adversarial debiasing can help to learn fairer models. Previous work has assumed that both main task labels and protected attributes are available in the dataset. However, protected labels are often unavailable, or only available in limited numbers. In this paper, we propose a training strategy which needs only a small volume of protected labels in adversarial training, incorporating an estimation method to transfer private-labelled instances from one dataset to another. We demonstrate the in- and cross-domain effectiveness of our method through a range of experiments.

1 Introduction

Protected attributes such as user gender can act as confounding variables in models, and spurious correlations with task response variables can lead to unfair predictions, as seen in tasks such as part-of-speech tagging (Hovy and Søgaard, 2015), hate speech detection (Huang et al., 2020), and sentiment analysis (Kiritchenko and Mohammad, 2018).

Adversarial methods are a popular method for mitigating bias associated with protected attributes, wherein the encoder attempts to prevent a discriminator from identifying protected attributes (Zhang et al., 2018; Li et al., 2018; Han et al., 2021). An adversarial network consists of a discriminator A and an encoder E. Each input x_i is required to be annotated with both a main task label y_i and protected attribute label g_i, and the discriminator identifies protected information in the representation generated by the encoder ($\hat{g}_i = A(h_i)$). The objective of the encoder training incorporates two parts: (1) predicting the main task label ($\hat{y}_i = C(E(x_i))$); and (2) preventing protected attributes from being identified by the discriminator.

An important limitation of previous adversarial debiasing work is that training instances must be annotated with both main task and protected labels (Li et al., 2018; Wadsworth et al., 2018; Zhang et al., 2018; Wang et al., 2019; Han et al., 2021). However, sourcing protected labels can be difficult, for reasons ranging from privacy regulations/ethical concerns, to only a small subset of users explicitly publicly disclosing protected attributes.

Our contributions are as follows: (1) we present a novel way of training the main task model and the discriminator separately, removing the restriction that every training instance needs to be annotated with protected labels; (2) we conduct in-domain experiments with diminishingly small amounts of protected-labelled data for sentiment analysis and hate speech detection, and show that our method can be successfully applied with remarkably little protected data; and (3) we present preliminary results for cross-domain transfer of protected attributes for sentiment analysis and POS tagging. The source code and data associated with this paper are available at: `https://github.com/HanXudong/Decoupling_Adversarial_Training_for_Fair_NLP`.

2 Methodology

Adversarial Separation Training Intuitively, adversarial supervision can be decoupled from the main task training, i.e., the inputs used for training the main model do not have to be annotated with protected labels. In doing so, we attain flexibility in being able to train the discriminator over only those instances where we have access to the protected attribute, as well as being able to transfer private attributes between datasets. Following the setup of Li et al. (2018), the optimisation objective is:

$$\min_{E,C} \max_{A} \sum_{i \in \mathcal{D}_{\text{main}}} \mathcal{X}(y_i, \hat{y}_i(E, C)) - \lambda_{\text{adv}} \sum_{j \in \mathcal{D}_{\text{adv}}} \mathcal{X}(g_j, \hat{g}_j(E, A)) \,,$$

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 471–477
August 1–6, 2021.

Algorithm 1: Predictability Estimation

```
Input: Out of domain dataset $D_O = (X_O, G_O)$,
       pretrained main task model $M_I$ in the target
       domain, $\mathcal{M}$, number of folds $k$, number of
       test folds at each step $t$
Output: protected label predictability scores, $\mathcal{P}$
Calculate hidden representations $H_O$ of $X_O$ from $M_I$
Partition $D_O$ into $k$ equi-sized folds as $\{F_1, \ldots, F_k\}$
Create $\binom{k}{t}$ test fold combinations as $T$
for $i \leftarrow$ to $\binom{k}{t}$ do
    Use $T_i$ as $F_{\text{test}}$, remaining folds as $F_{\text{train}}$
    Train $\mathcal{M}$ on $F_{\text{train}}$
    forall $(h_{O,j}, g_{O,j}) \in F_{test}$ do
        Make prediction $\hat{g}_{O,j} = \mathcal{M}(h_{O,j})$
        Accumulate correct predictions
          $\mathcal{P}(j) \mathrel{+}= \delta(\hat{g}_{O,j}, g_{O,j})$
    end
end
return $\mathcal{P}$
```

where $\mathcal{X}$ is cross entropy loss, and λ_{adv} is a tunable hyperparameter. The critical observation of this work is that the two sources of data $\mathcal{D}_{\text{main}}$ and $\mathcal{D}_{\text{adv}}$ need not be the same, but may overlap or be entirely disjoint, as we explore in Section 3.

Filtering Cross-domain Data The inputs used for discriminator training do not have to be annotated with the main task label. Inspired by the domain robustness results of Li et al. (2018) with adversarial training, we examine cross-domain adversarial learning where protected labels are unknown for the target task in two settings: (1) sentiment analysis classification, and (2) part of speech (POS) tagging. In both cases we use external race labels from a hate speech dataset, and ignore any protected attributes in the original dataset.

According to our experiments, one problem associated with using cross-domain protected-labelled data is that some protected labels may not be relevant to the target domain. To address this problem, inspired by adversarial filtering (Le Bras et al., 2020), we conduct preliminary exploration on filtering cross-domain data in adversarial separation training. This method finds out-of-domain instances with the most confident predictions of the protected label, and selects these instances to use as a silver standard in training in-domain adversaries.

To estimate the protected label predictability of a cross-domain instance $(x_{O,i}, g_{O,i})$ in the target domain given a trained main task model in the target domain (M_I) and an estimator $\mathcal{M}$ (a logistic regression classifier), the protected label predictability of each instance is estimated as shown in Algorithm 1. Specifically, for folds k and test folds t, the predictability of each instance is estimated n times (i.e., by n different models) over different training sets. n can be derived from k and t as follows: $n = \frac{(k-1)!}{(k-t)!(t-1)!}$. Note that when $t = 1$, our method equates to k-fold cross-validation, and the predictability of each instance is estimated once. We demonstrate how the estimated predictability $\mathcal{P}$ can be used in Section 3.5.

3 Experiments and Analysis

In this section, we report on experiments under two scenarios: (1) in-domain, where protected labelled data and main labelled data are from the same domain; and (2) cross-domain, where protected labelled data are from a different domain to the main task data.

3.1 GAP

A common way of measuring fairness is GAP: the absolute difference for a metric between data subsets selected by different settings of the protected attribute. For instance, in the binary setting, we can compare the true positive rate (TPR) for male- vs. female-authored documents in the test set; this difference is the TPR-GAP, and is zero for a fair model.

3.2 In-domain: Sentiment Analysis

Data We experiment with the dataset of Blodgett et al. (2016), which contains tweets that are either African American English (AAE)-like or Standard American English (SAE)-like (following Han et al. (2021)). Each tweet is annotated with a binary "race" label (based on AAE or SAE), and a binary sentiment score determined by the (redacted) emoji within it.

We use the train/dev/test splits from Han et al. (2021) of 100k/8k/8k instances, respectively. The full dataset is artificially balanced across the four race–sentiment combinations. To (re)introduce bias into the dataset, previous work has skewed the training data to generate race–sentiment combinations (AAE–happy, SAE–happy, AAE–sad, and SAE–sad) of 40%, 10%, 10%, and 40%, respectively, leaving the dev and test data unbiased.

To examine how much private labelled data is needed, we randomly mask the protected attribute from up to 99% of the training data.

Models We use the same model architecture as Han et al. (2021), in the form of the fixed-parameter DeepMoji encoder (Felbo et al., 2017)

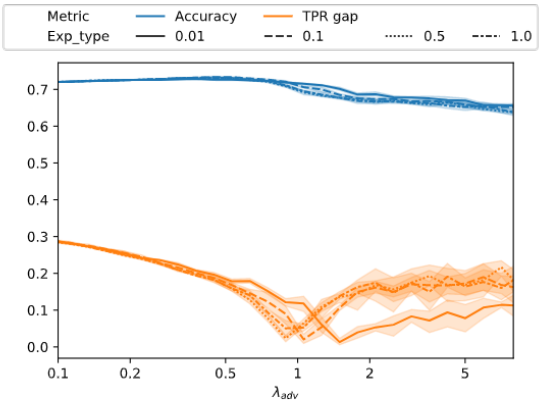

Figure 1: In-domain Sentiment Analysis: main task accuracy and GAP with respect to the trade-off hyperparameter λ_{adv}; shaded areas = 95% CI estimated over 5 runs.

followed by a trainable 3-layer MLP. DeeMoji contains 22.4 million parameters and is pretrained over 1246 million tweets to predict one of 64 common emojis. The discriminator for adversarial training (for all experiments in this paper) is trained to predict the protected attribute from the hidden states of the last hidden layer of the MLP classifier. Full training details are provided in the Appendix.

Results We explore 4 dataset settings where 100%, 50%, 10%, and 1% of the training data is labelled with its private attribute. We tune λ_{adv} log-uniformly under each data setting, using the same case-control training strategies for all experiments in this paper.

As shown in Figure 1, tuning λ_{adv} results in a series of candidate models, and there is a clear inflection point for the TPR GAP under each data setting, at different values of λ_{adv}.

To compare the adversarial training performance across different numbers of private labels, we show the trade-off plot in Figure 2a. Each point reflects the Accuracy and TPR GAP of a candidate model with a given λ_{adv}. The points for the three data settings of 100%, 50%, and 10% are hard to separate, indicating that adversarial training with only 10% of protected labels can achieve similar results to using protected labels for 100% of the data. Even with 1% of private labels, debiasing is evident, but this comes at a lower accuracy for a given TPR GAP level.

3.3 In-domain: Hate Speech Detection

Data Our second in-domain dataset is the English Twitter hate speech detection dataset of Huang et al. (2020), where each tweet is labelled with a binary hate speech label and also contains (binary) demographic indicators for the tweet author: binary gender (female or male), location (U.S. or other), age (older or younger than the median), and race (white or other). We focus on *age*, which has been shown to result in the greatest model unfairness (Huang et al., 2020), and use the train/dev/test splits of Huang et al. (2020). Since age information is not available for all authors, we downsample to get a subset of tweets which are annotated with age, with approximately 31k/6.7k/6.7k in training/dev/test.

Model Huang et al. (2020) compare 4 different model architectures for the hate speech detection task — TF-IDF-weighted feature-based logistic regression, convolutional neural network, an RNN (in the form of a biGRU), and BERT (Devlin et al., 2019) — and found the RNN model to consistently perform best. Based on this, we use the same RNN model, and perform debiasing on top of it.

Results Figure 2b shows the trade-off plot with respect to hate speech detection models under similar data conditions as our first experiment (100%, 50%, 10%, and 0.1%). Consistent with our previous observations, there is little distinguishing 100%, 50%, and 10%. In fact, when we further decrease the proportion of protected labels, we observe that even with 0.1% of protected attributes, the trade-off is close to the 100% model.

3.4 Cross-domain: Sentiment Analysis

Next, we turn to the cross-domain setting, in taking protected labelled data from one domain and using it to adversarially debias a sentiment analyser over a different but similar domain.

Data We use the same model architecture and sentiment analysis dataset as in Section 3.2, but source private attributes (in the form of race) from the hate speech dataset from Section 3.3 to train the discriminator. Note that the race labels for the hate speech dataset (white or other) diverge slightly from those in the target domain (SAE or AAE).

Results Figure 3 shows the trade-off plot for the model. To compare the cross- and in-domain settings, we include in-domain adversarial debiasing results (1% data setting). Compared with in-domain results, the trade-off for cross-domain debiasing is worse than the 1% in-domain setting, but substantially better than random.

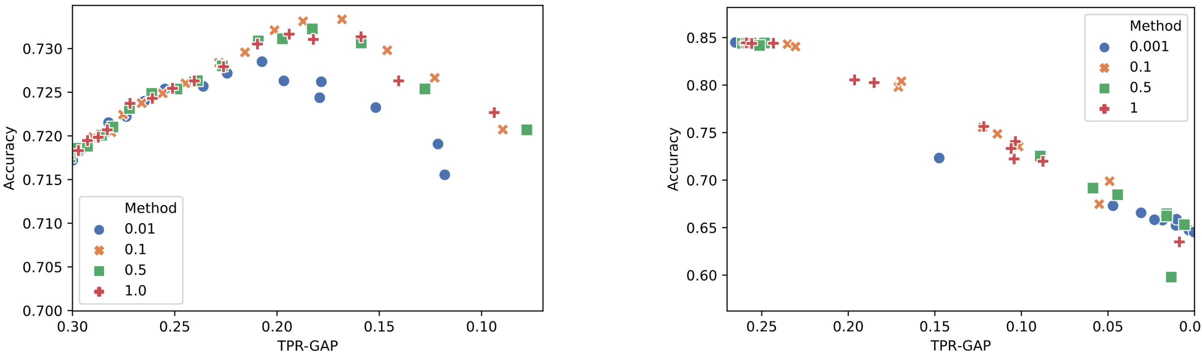

(a) Sentiment Analysis

(b) Hate Speech Detection

Figure 2: In-domain evaluation showing the accuracy–TPR-GAP trade-off for different fractions of protected-labels in the training dataset. For each data setting, we evaluate predictions for λ_{adv} settings near to that where the model achieves its smallest development TPR GAP. Since there is strong correlation between TPR GAP and TNR GAP, we only include TPR GAP.

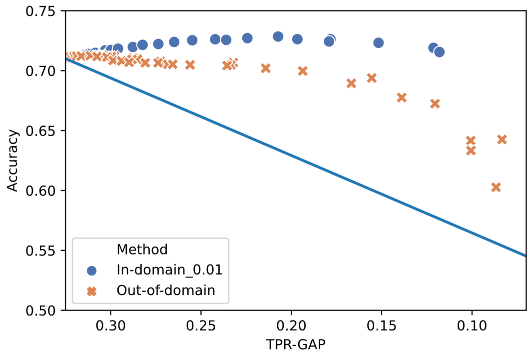

Figure 3: Cross-domain Sentiment Analysis: trade-off plot. The blue solid line denotes baseline debiasing results, based on randomly replacing main task predictions with a Bernoulli r.v. sampled from $p = 0.5$.

In terms of the drop in model accuracy during debiasing, each point in Figure 3 corresponds to a candidate model, and in this cross-domain setting, some models (e.g., for those models with TPR-GAP around 0.2) are able to reduce the bias by about 50% while maintaining performance that is close to the vanilla model. Managing the trade-off relates to model selection and the requirements of a given application scenario, for example, choosing a model that is able to achieve at least a certain fairness level. Overall, at a given bias level, our method doesn't make use of any in-domain protected labels, and consistently outperforms the random baseline.

3.5 Cross-domain: POS tagging

As second cross-domain task, we follow Li et al. (2018) in performing POS tagging.

Data We use three datasets for different purposes: main task training, adversarial training, and out-of-domain evaluation.

Following Li et al. (2018), we train a biLSTM POS tagging model on the English Web Treebank (Bies et al., 2012), comprising 13.5k POS-tagged sentences without protected labels. To evaluate model performance and fairness, we use the TrustPilot English POS-tagged dataset (Hovy and Søgaard, 2015), consisting of 600 sentences with both POS labels and binary author-age labels (over-45-year-old and under-35-year-old).

To train the discriminator, we use unlabelled TrustPilot data (Hovy, 2015), which consists of 156.5k English reviews with author-age labels. Based on protected-label predictability estimation (Algorithm 1), we examine 4 subsampling strategies: (1) "random", based on random-sampling; (2) "largest leakage", select instances with the highest predictability (intuitively the most biased instances); (3) "smallest leakage", select instances where the predictability is below a majority-class baseline; and (4) "absolute leakage", a combination of largest and smallest leakage where equal number of instances from the largest leakage sampling and the smallest leakage sampling are concatenated together.

Results We follow Li et al. (2018), in evaluating fairness via the difference in tagging accuracy between age groups.

Figure 4 shows the trade-off plot with respect to the 4 filtering strategies. Note that the test set only includes 600 instances, so we explore a wider range of the λ_{adv}, and train 5 random initialized

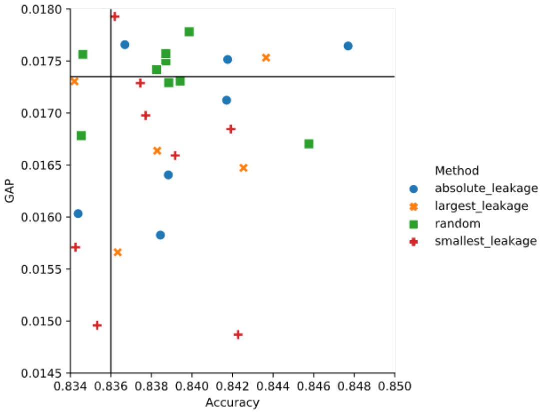

Figure 4: Cross-domain POS Tagging. The vertical line denotes the biased model accuracy, and the horizontal line denotes the biased model Accuracy GAP. Points in the upper-right quadrant are preferable.

	Top P		**Top F**	
	Acc↑	GAP↓	ACC↑	GAP↓
Biased	83.60	1.74	83.60	1.74
Random	83.94	1.73	83.89	1.70
Largest	83.86	1.66	83.63	1.63
Smallest	83.92	1.68	83.53	1.57
Absolute	84.18	1.75	83.84	1.64

Table 1: Evaluation results on the test set, median value over 5 best models. Biased stands for the non-debiasing model. **Top P** = 5 models with best performance, and **Top F** = 5 models with best fairness. "↑" and "↓" indicate that higher and lower performance, resp., is better for the given metric.

models for each λ_{adv} and take the average. Points in the lower-right quadrant are preferable, in that they decrease bias while increasing accuracy.

We report models of each predictability estimation based sampling method, and all models from random sampling (with respect to different λ_{adv}). Compared to the biased model performance and fairness (vertical and horizontal lines, respectively), the random sampling method does not lead to clear improvements, while our proposed methods lead to consistent gains.

We further compare these methods numerically in Table 1 by selecting top 5 best models from what has been shown in Figure 4. Specifically, we select models with top 5 performance (largest accuracy score) or fairness (smallest GAP) separately, and report the median values of accuracy and GAP over the selected models.

Largest and smallest leakage show close results and are safer choices that consistently outperform random and non-debiasing methods. Intuitively, in a binary classification problem, instances within the the smallest leakage group could also be informative as they could be transformed to largest leakage groups by reverting the predictions, i.e., $\hat{g}_{O,j} = 1 - \mathcal{M}(h_{O,j})$, thus using largest leakage sampling is similar to using smallest leakage sampling. Combining largest and smallest leakage instances together, the absolute sampling method achieves slightly better accuracy performance than other sampling strategies and consist better performance-fairness trade-off than the biased model.

4 Conclusion

We propose a novel training strategy for adversarial debiasing which decouples the training of the main task model and discriminator, including the possibility of training on different data. Based on in-domain evaluation over sentiment analysis and hate speech detection, our method performs as well as the standard adversarial method using only 10% of protected labels. Furthermore, experiments in a cross-domain setting demonstrate the potential feasibility of the method in settings where protected labels are not available.

Acknowledgments

We thank Lea Frermann, Shivashankar Subramanian, and the anonymous reviewers for their helpful feedback and suggestions.

Ethical Considerations

This work aims to use less protected-labelled data in adversarial debiasing. Adversarial training in NLP can help to learn fairer models across demographic attributes used. However, due to the lack of protected labels, previous work has had to use identified demographic attributes for adversarial debasing. By reducing the number of protected labels that are needed in adversarial training, using only self-identified characteristics becomes an effective choice.

References

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank LDC2012T13. Linguistic Data Consortium.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xudong Han, Timothy Baldwin, and Trevor Cohn. 2021. Diverse adversaries for mitigating bias in training. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2760–2765, Online. Association for Computational Linguistics.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 752–762, Beijing, China. Association for Computational Linguistics.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 483–488, Beijing, China. Association for Computational Linguistics.

Xiaolei Huang, Linzi Xing, Franck Dernoncourt, and Michael Paul. 2020. Multilingual twitter corpus and baselines for evaluating demographic bias in hate speech recognition. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1440–1448.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53, New Orleans, Louisiana. Association for Computational Linguistics.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *International Conference on Machine Learning*, pages 1078–1088. PMLR.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30, Melbourne, Australia. Association for Computational Linguistics.

Christina Wadsworth, Francesca Vera, and Chris Piech. 2018. Achieving fairness through adversarial learning: an application to recidivism prediction. *FAT/ML Workshop*.

Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. 2019. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5310–5319.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

A In-domain Sentiment Analysis Full Plot

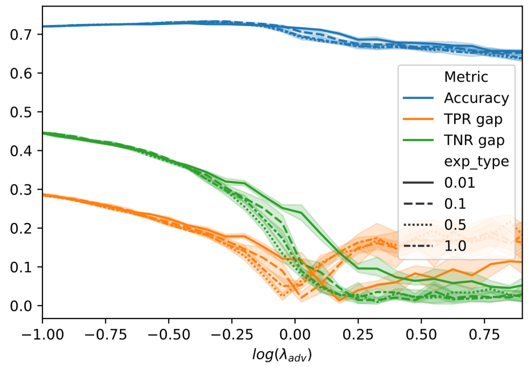

Figure 5: In-domain Sentiment Analysis: main task accuracy and GAP with respect to the trade-off hyperparameter λ_{adv}; shaded areas = 95% CI estimated over 5 runs.

B Computing Infrastructure Used

- CPU: Intel(R) Core(TM) i9-9900K CPU
- GPU: NVIDIA GeForce RTX 2080 Ti
- RAM: 32 GB

C Sentiment Analysis

Models All models are trained and evaluated on the same training/test split. The Adam optimizer is used with learning rates of 3×10^{-5} for the main model and 3×10^{-6} for the sub-discriminators. The minibatch size is set to 1024. Sentence representations (2304d) are extracted from the DeepMoji encoder. The hidden size of each dense layer is 300 in the main model, and 256 in the sub-discriminators. We train M for 60 epochs and each A for 100 epochs, keeping the checkpoint model that performs best on the dev set. Running time: 35 s/epoch

Hyperparameter Range

- λ from 10^{-2} to 10^{+2}. log uniform sampling 30 trails

D Hate Speech

Model Architecture

- hidden size, type=int, 300
- embedding size, type=int, 400
- number of classes, type=int, 2
- adversarial level, type=int, -1 (last hidden layer)
- learning rate, type=float, 0.00003
- number of discriminator, type=int, 1
- adv units, type=int, 256
- batch size, type=int, 512
- epoch, type=int, 100
- dropout, type=float, 0.5
- Running time: 52 s/epoch

Hyperparameter Range

- λ from 10^0 to 10^3 based on log uniform sampling over 15 trials

E POS Tagging

Model Architecture

- BATCH SIZE = 64
- LEARNING RATE = 1e-3
- EMBEDDING DIM = 50
- HIDDEN DIM = 100
- N LAYERS = 2
- BIDIRECTIONAL = True
- DROPOUT = 0.25
- EPOCHS = 50
- SEED = 960925
- MIN FREQ = 2
- SAMPLING INDEX = 10
- LAMBDA = 1e-3
- dropout = 0.5
- Running time: 12 s/epoch

Hyperparameter Range

- λ from 10^{-10} to 10^{-8} based on log uniform sampling over 20 trials

GO FIGURE: A Meta Evaluation of Factuality in Summarization

Saadia Gabriel♠* **Asli Celikyilmaz**♡ **Rahul Jha**♣ **Yejin Choi**♠♢ **Jianfeng Gao** ♣
♠Paul G. Allen School of Computer Science & Engineering, University of Washington
♡Facebook AI Research
♣Microsoft Research
♢Allen Institute for Artificial Intelligence
{skgabrie,yejin}@cs.washington.edu,{aslic}@fb.com
{rajh,jfgao}@microsoft.com

Abstract

While neural language models can generate text with remarkable fluency and coherence, controlling for factual correctness in generation remains an open research question. This major discrepancy between the surface-level fluency and the content-level correctness of neural generation has motivated a new line of research that seeks automatic metrics for evaluating the factuality of machine text. In this paper, we introduce GO FIGURE, a *meta*-evaluation framework for evaluating factuality evaluation metrics. We propose five necessary conditions to evaluate factuality metrics on diagnostic factuality data across three different summarization tasks. Our benchmark analysis on ten factuality metrics reveals that our *meta*-evaluation framework provides a robust and efficient evaluation that is extensible to multiple types of factual consistency and standard generation metrics, including QA metrics. It also reveals that while QA metrics generally improve over standard metrics that measure factuality across domains, performance is highly dependent on the way in which questions are generated.

1 Introduction

The goal of text generation systems is to produce text that is fluent, coherent, relevant, as well as factually correct. Recent progress in neural approaches to building semantically constrained text generation systems has shown tremendous improvements in this direction (Liu and Lapata, 2019; Guo et al., 2018; Durmus et al., 2020; Wang et al., 2020). However, an important issue in text generation systems is that they can yield factually inconsistent text, caused by somewhat distorted or fabricated facts about the source text. Especially in document summarization tasks, models that abstract away salient aspects, have been shown to generate text with up to 30% factual inconsistencies (Kryscinski et al., 2019; Falke et al., 2019a; Zhu et al., 2020).

Commonly used metrics for measuring quality of generated text fail to capture structural aspects of language like negation and poorly correlate with human judgements (Hashimoto et al., 2019; Clark et al., 2019; Sellam et al., 2020), leading to a rapidly progressing search for factuality-driven summarization metrics.

In this work, we propose GO FIGURE[1], a meta-evaluation framework for assessing the effectiveness of factuality metrics across multiple domains - *extreme summarization*, *multi-sentence news summarization* and the understudied *dialogue summarization* domain. Our contributions are as follows: (*i*) a set of diagnostics for measuring sensitivity of metrics to factual inconsistency, (*ii*) a diagnostic evaluation dataset of context/summary pairs for measuring effectiveness of new factuality metrics in a controlled setting, and (*iii*) an evaluation dataset of summaries generated by transformer-based models (Raffel et al., 2019) annotated with types of factual errors.

2 Factuality Metric Meta Evaluation

Since reference summaries may be an incomplete representation of the salient facts in a source document or unavailable, we consider factuality in terms of how well candidate summaries are factually grounded with respect to the source document.

We define a set of five conditions for a factual consistency metric $M(D, S_i)$ to measure factuality of a summary S_i with respect to a source document D. These conditions are given in Table 1.

2.1 Testing Factuality Metric Validity

For the purposes of testing boundedness (Condition I), we define the **Lower Bound** for a metric M as

---

[*] Work done while first author was interning at MSR.

[1] **G**eneral **O**utline for **F**actuality **I**n **G**enerative **U**nde**R**standing **E**valuation.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 478–487
August 1–6, 2021.

Condition	Definition	Motivation
Boundedness (I)	There exists S_r, S_f such that $M(D, S_r) \leq M(D, S_i) \leq M(S_f)$.	In general, the exact factuality level of S_i may be unclear. Metric bounds provide points of comparison.
Sensitivity (II)	The metric value for S_i should correlate with the level of factuality captured by S_i.	A bounded but insensitive factuality metric may assign higher values to mostly nonfactual or unrelated summaries over summaries that are close to the reference.
Robustness (III)	The metric should be *robust* across types of factual errors.	A metric that is sensitive only to a subset of errors might ignore a significant number of model-generated errors (Figure 1).
Generality (IV)	The metric should satisfy conditions I,II,III and V across domains.	Prior work such as Reiter and Belz (2009) highlight the risk of claiming validity without testing generality.
Human Correlation (V)	The metric should *correlate* with human judgements of factuality.	The scoring function $H(D, S_i)$ represented by human evaluation is a gold standard for assessment of generation quality (Chaganty et al., 2018), so $M(D, S_i)$ should be an approximation.

Table 1: Details of factuality metric conditions. Here M is a metric scoring function, D is a source document and S_i is a summary.

$M(D, S_r)$ where D is the source document and S_r is a randomly sampled summary from the corpus.[2] We define the **Upper Bound** for the metric as $M(D, S_f)$, where S_f is the reference ground-truth summary. Since our controlled experiments use transformed versions of the reference summary with injected errors, the original reference is guaranteed to be at least as factually consistent as a transformed summary.

To test sensitivity (Condition II), we measure the **correlation** (Pearson's r) between the factual inconsistency level[3] of the summaries (i.e. the number of injected errors) and the average metric score. Then we measure statistical significance using the ***p*-value** from a two-tailed hypothesis test. We check whether metrics satisfy robustness and generality (Conditions III and IV) by separately running this analysis over multiple domains and the factual error types shown in Figure 1. We measure how well metric values match human assessment of factuality by checking the correlation between factual consistency levels determined using manual annotation.

2.2 Theoretical Cases

For Condition I, we scope boundedness to only consider cases that are likely to arise in realistic summarization settings. However, there are hypothetical cases that may have ramifications for metric validity. For example, we expect that $M(D, D) \approx 1$ and $M(D, \varnothing) \approx 0$ for a metric M with values in the range $[0, 1]$, a document D, and an empty string summary $\varnothing$. For non-deterministic metrics, restrictions on variability between runs may also be desired.

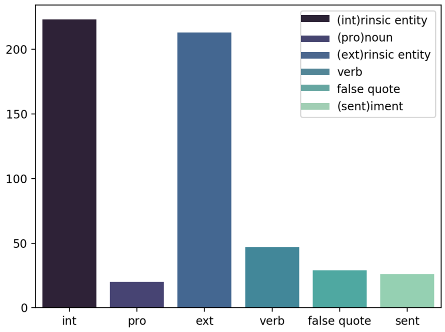

Figure 1: Distribution of common factual error types in sampled generated summaries (96.37% of all errors). We draw from the same error types for our controlled analysis to ensure we match the true distribution of errors. Here *extrinsic entity* refers to entities that did not previously appear in the source, while an *intrinsic entity* appeared in the source.

3 Evaluation Datasets

We evaluate metrics on three datasets: 1-sentence BBC news summaries from the XSUM extreme summarization dataset (Narayan et al., 2018), multi-sentence summaries from the

---

[2] While this may not be the strictest lower bound in theoretical terms, we consider it appropriate as an empirical lower bound since the content is irrelevant to the document. A single random summary is used.

[3] For our experiments, we inject up to a maximum of x errors with $x \in \{1, 2, 3\}$.

CNN/DailyMail dataset (Nallapati et al., 2016), and the recently released SAMSUM corpus (Gliwa et al., 2019) consisting of English language conversations written by linguists and aligned multi-sentence summaries.

3.1 Diagnostic Datasets

To test the ability of proposed metrics to fulfill our predefined conditions, we set up two diagnostic datasets consisting of (*i*) transformed reference summaries with simulated factuality errors that allow us to induce and measure factuality levels in a controlled setting and (*ii*) summaries generated by state-of-the-art transformer summarization models that allows us to measure the effectiveness of metrics in a real data setting. We sample 500 source / summary pairs for each domain.[4]

3.1.1 Model-Generated Datasets

In order to observe how metrics perform on machine-generated summaries, we generate summaries from fine-tuned T5 encoder-decoder summarization models (Raffel et al., 2019) that was pre-trained on news summarization data. We generate summary text using either beam search or sample-based decoding strategies. We then annotate the generated summaries for fine-grained factual errors using the types in Figure 1 to create a hand-curated factual consistency diagnostic dataset.

4 Factuality Metrics for Evaluation

We mainly focus on meta-evaluating most recently proposed factual consistency metrics which use two types of proxy natural language understanding (NLU) objectives aimed at implicitly capturing factuality in generated text: *question-answering* (QA) and a masked token prediction *cloze task*. For QA we evaluate using SummaQA (which uses QA pairs from the source, Scialom et al., 2019) and FEQA (which uses QA pairs from the summary, Durmus et al., 2020), while for the cloze task setting we use BLANC-Help and BLANC-Tune (Vasilyev et al., 2020, see the appendix for details of metrics). We also measure the *factual-awareness* of BERTScore (Zhang et al., 2020), a summarization metric that is aimed primarily at improving coherency rather than factual consistency, and standard summarization evaluation metrics (e.g. ROUGE (Lin, 2004)).

5 Meta-Analysis of Factuality Metrics

5.1 Controlled Data Experiments

We provide the results of the sensitivity analysis over our controlled data on the XSUM domain in Table 2, on CNNDM in Table 3 and on SAMSUM in Table 4. Our analysis reveals that QA metrics, ROUGE-(2/3) and BERTScore generally perform well at evaluating factuality. In contrast, ROUGE-(1/L) are frequently invalid as factuality metrics (Tables 2 and 3), and the performance of Cloze metrics varies across domains (BLANC-Tune is invalid on XSUM, but does fairly well on other domains). Also, performance of metrics tends to be much lower on news domains when we consider non-entity-based errors with the exception of QA-based metrics, ROUGE-(2/3) and BERTScore, indicating that while factuality and standard metrics are fairly attuned to changes in factual consistency that relate to entity-based errors, they are less robust to other types of factual errors.

5.2 Comparison with Human Evaluation of Model Generations

We find that metrics displaying invalid behavior on controlled data (for instance assigning higher metric values to more factually inconsistent summaries on XSUM in Table 2) also display this invalid behavior in model generations (Table 5). This indicates that meta-evaluation with controlled data is effective as a diagnostic tool for finding weak factuality metrics, and follows our intuition that non-entity errors, while frequently produced by abstractive summarization models, are difficult for standard summarization metrics to identify. When considering better-performing factuality metrics identified by the controlled error analysis, we find that the controlled data analysis is generally able to identify better-performing metrics (SummaQA, ROUGE-(2/3) and BERTScore) for XSUM with the exception of FEQA (FEQA metric performs well on XSUM controlled analysis (Table 2), but only approaches this performance on SAMSUM when we consider human eval). The strong overall performance of ROUGE-3 is consistent with the findings of (Fabbri et al., 2021) on CNNDM, our work confirms that this metric is more consistently correlated with factuality than other ROUGE variations across domains.

---

[4] See the Appendix for details of linguistic feature extraction for injecting errors.

	CLOZE		QA			STANDARD and CONTEXTUAL				
	BLANC-Help	**BLANC-Tune**	**SummaQA-C**	**SummaQA-F1**	**FEQA**	**R-1**	**R-2**	**R-3**	**R-L**	**BERTScore**
Upper Bound	5.99	1.73	9.64	4.48	27.87	10.61	2.56	0.72	9.32	83.76
Level 1	5.73 / 5.98	1.74 / 1.71	9.44 / 9.44	3.80 / 4.31	23.20 / 26.94	10.49 / 10.76	2.54 / 2.56	0.70	9.22 / 9.42	83.53 / 83.56
Level 2	5.46 / 5.99	1.59 / 1.78	9.27 / 9.35	3.40 / 4.22	20.05 / 26.55	10.40 / 10.86	2.51 / 2.54	0.69 / 0.68	9.16 / 9.49	83.36 / 83.38
Level 3	5.30 / 5.97	1.58 / 1.76	9.16 / 9.23	3.13 / 4.14	15.81 / 26.06	10.33 / 10.92	2.49 / 2.52	0.69 / 0.67	9.10 / 9.55	83.21 / 83.26
Lower Bound	0.51	-0.14	1.28	0.26	1.18	5.44	0.39	0.01	4.94	80.08
Correlation	-0.99 / -0.61	-0.88 / **0.69**	-0.99 / **-1.00**	-0.99 / **-1.00**	**-1.00**	**-1.00** / **0.98**	-0.97 / **-1.00**	**-0.87** / **-1.00**	**-1.00** / **1.00**	**-1.00**
p-value	0.09 / 0.59	0.32 / 0.51	0.07 / 0.05*	0.07 / 0.03*	0.05* / 0.04*	0.03* / 0.10	0.16 / 0.05*	0.33 / 0.05*	<0.01** / 0.02*	0.02* / 0.06

Table 2: Results of simulated factual error data experiments (**XSUM**, average of 5 runs, **=significant for p $\leq$ .01, *=significant for p $\leq$ .05). **For cells with (·/·), results for entity errors are reported on the left, results for non-entity errors are reported on the right.** The details for the upper/lower bounds, *p*-value and correlation measures are explained in §2.1. For sensitivity to factual consistency and correlation w/ factuality levels, we highlight the best-performing and lowest-performing metrics in **green** and **red** respectively. For cases where metric values are invalid (e.g. the metric values increase as factuality decreases), we highlight in **purple**.

	CLOZE		QA			STANDARD and CONTEXTUAL				
	BLANC-Help	**BLANC-Tune**	**SummaQA-C**	**SummaQA-F1**	**FEQA**	**R-1**	**R-2**	**R-3**	**R-L**	**BERTScore**
Upper Bound	7.60	5.79	13.82	10.87	37.56	14.33	8.08	4.75	13.83	84.36
Level 1	7.29 / 7.50	5.56 / 5.69	13.30 / 13.53	9.58 / 10.63	33.35 / 36.64	14.11 / 14.37	7.78 / 7.91	4.51 / 4.57	13.60 / 13.84	84.13 / 84.20
Level 2	7.03 / 7.43	5.43 / 5.58	12.93 / 13.24	8.53 / 10.38	28.46 / 36.13	13.95 / 14.38	7.55 / 7.75	4.32 / 4.40	13.44 / 13.85	83.94 / 84.04
Level 3	6.72 / 7.38	5.23 / 5.53	12.54 / 13.04	7.54 / 10.26	25.12 / 35.63	13.82 / 14.38	7.35 / 7.62	4.14 / 4.27	13.29 / 13.85	83.77 / 83.90
Lower Bound	-0.67	-0.19	1.61	0.12	0.58	5.85	0.47	0.02	5.55	78.16
Correlation	**-1.00** / -0.99	**-0.99** / -0.97	**-1.00** / **-1.00**	**-1.00** / -0.98	**-0.99** / **-1.00**	**-1.00** / **0.96**	**-1.00**	**-1.00**	**-1.00** / **0.91**	**-1.00**
p-value	0.03* / 0.08	0.07 / 0.17	0.01** / 0.06	0.01** / 0.13	0.07 / <0.01**	0.04* / 0.17	0.02* / 0.04*	<0.01** / 0.04*	0.03* / 0.27	0.01** / 0.02*

Table 3: Results of simulated factual error data experiments (**CNNDM**, average of 5 runs). (See Table 2 caption for details.)

	CLOZE		QA			STANDARD and CONTEXTUAL				
	BLANC-Help	**BLANC-Tune**	**SummaQA-C**	**SummaQA-F1**	**FEQA**	**R-1**	**R-2**	**R-3**	**R-L**	**BERTScore**
Upper Bound	15.23	10.13	13.83	17.23	55.36	26.55	8.24	4.07	25.06	84.60
Level 1	13.97 / 15.03	9.00 / 9.47	13.48 / 13.52	15.00 / 16.71	45.31 / 54.25	25.31 / 26.18	7.85 / 7.86	3.84 / 3.73	23.91 / 24.69	84.42 / 84.38
Level 2	12.87 / 15.01	8.36 / 9.46	13.16 / 13.26	12.26 / 16.50	37.01 / 53.10	24.27 / 25.86	7.60 / 7.59	3.68 / 3.50	22.99 / 24.38	84.28 / 84.19
Level 3	12.02 / 14.93	7.74 / 9.36	12.99 / 13.21	10.12 / 16.24	29.62 / 52.34	23.23 / 25.58	7.32 / 7.36	3.48 / 3.35	22.01 / 24.12	84.13 / 84.07
Lower Bound	0.92	-0.53	7.86	0.10	0.55	5.33	0.23	0.01	5.09	80.79
Correlation	**-1.00** / -0.96	**-1.00** / **-0.91**	**-0.99** / -0.94	**-1.00**	**-1.00** / -0.99	**-1.00**	**-1.00**	**-1.00** / -0.99	**-1.00**	**-1.00** /-0.99
p-value	0.05* / 0.18	0.01** / 0.28	0.11 / 0.23	0.05*	0.02* / 0.07	<0.01** / 0.03*	0.03*	0.05* / 0.08	0.01** / 0.04*	0.01** / 0.07

Table 4: Results of simulated factual error data experiments (**SAMSUM**, average of 5 runs). (See Table 2 caption for details.)

Metric	**XSUM**		**SAMSUM**	
	Corr (- ←)	***p*-value**	**Corr (- ←)**	***p*-value**
BLANC-Help	0.04	0.55	-0.01	0.82
BLANC-Tune	0.00	0.98	-0.03	0.64
SummaQA-C	-0.11	0.11	-0.09	0.18
SummaQA-F1	-0.12	0.07	-0.14	0.03*
FEQA	0.04	0.57	-0.03	0.69
R-1	0.07	0.19	0.01	0.82
R-2	-0.10	0.15	-0.03	0.59
R-3	-0.12	0.07	-0.09	0.18
R-L	0.07	0.13	0.01	0.83
BERTScore	-0.17	0.01**	0.03	0.64

Table 5: Correlation (Corr) for 250 annotated XSUM and 250 SAMSUM generated summaries with fine-grained labeling. The arrow next to "Corr" indicates the direction of a correct correlation.

6 Related Work

Prior work concerning evaluation of automatic metrics and human evaluation for NLG systems has mainly focused on general analysis of output quality or coherence and fluency (Callison-Burch et al., 2007; Graham, 2015; Fabbri et al., 2021), rather than factuality. Recent efforts by NLP researchers have drawn attention to the issue of factual errors and hallucinations in the output of neural summarization models (Cao et al., 2018; Massarelli et al., 2019; Zhao et al., 2020; Falke et al., 2019b; Goodrich et al., 2019; Celikyilmaz et al., 2020). A number of works have highlighted the effectiveness of QA and cloze task objectives for evaluating or improving factuality on specific domains (Eyal et al., 2019; Huang et al., 2020). We aim to evaluate these metrics more broadly, and consider a wider range of domains (notably dialogue).

6.1 Discussion of Meta Evaluation and Conclusion

Our analyses show that in contrast to prior work on factual consistency that mostly concentrated on one specific domain and dataset, our GO FIGURE framework is effective at evaluating sensitivity and validity of factual consistency metrics with only reference summaries, rather than requiring computationally intensive testing across summarization model variants to identify metric strengths and shortcomings.

We highlight the following key points from experiments run using *meta*-evaluation:

Standard summarization metrics are not always valid measures of factuality. ROUGE-1 and ROUGE-L fail to accurately measure factual inconsistency across domains in our controlled analysis. The ROUGE-L results raise the question of context *relevance*. While ROUGE-L takes into account more context than other ROUGE variations, this context may not be relevant for assessing factuality. For example, swapping "decreased" for "increased" dramatically changes the meaning in the summary *"Scotland's renewable energy output increased by 45% in the first quarter of this year, compared with the same period last year."*, but ROUGE-L is not affected. Despite the frequent use of ROUGE-L as a more contextual measure, prior work has also noted that ROUGE-N outperforms ROUGE-L (Rankel et al., 2013; Fabbri et al., 2021).

Analysis on human annotated data is still necessary as an upper-bound on *meta*-evaluation quality. While BLANC-Help, FEQA metric and BERTScore values decrease with factual inconsistency on controlled data, the metrics may sometimes be positively correlated with factual inconsistency on generated data. This emphasizes the importance of a expert curated test set as part of the GO FIGURE meta evaluation for the most rigorous testing. **A question-answering objective is promising for measuring factual consistency across domains, but effectiveness depends on the question.** While QA metrics can perform well at measuring factual consistency of generated summaries, our *meta*-evaluation reveals this is dependent on the way in which questions are asked. While both QA metrics use SQuAD-based systems (Rajpurkar et al., 2016), asking questions from the source rather than the summary is most robust across domains. This opens the door to metrics based on more contextual QA like commonsense (Shwartz et al., 2020).

We will release our *meta*-evaluation framework and diagnostic datasets to aid in development of effective summarization factuality metrics. In future work, summary meta-metric results (e.g. correlation on simulated data) could be used as rewards for reinforcement learning driven approaches to training factuality metrics.

7 Ethics and Broader Impact Statement

Ethical considerations involving our *meta*-evaluation framework primarily revolve around human evaluation. News articles and dialogues may contain references to distressing events or abnormal social behavior. All our expert annotators voluntarily took part in the human evaluation with prior knowledge of the type of content being evaluated. Crowd-sourced human evaluation trials were conducted under an IRB exemption.

Our work outlines a simple and effective approach for evaluating factuality metrics in summarization. This can aid in development of more robust and sensitive factuality metrics to accurately evaluate the factual correctness of generative models. This is key as improvement in the coherency of models accelerates, potentially leading to generations that appear to be high quality while containing factual inaccuracies. Our framework could also evaluate factuality metrics for use in identifying human-written errors, mitigating potential spread of misinformation.

Acknowledgments

The authors thank Yichen Jiang and Shiyue Zhang for feedback on implementation, Hannah Rashkin and Tom McCoy for help with MSR GPU clusters, Rowan Zellers and Elizabeth Clark for pointers to related work, as well as other members of the UW NLP, MSR AI and MSR MSAI communities for helpful comments.

References

E. Ageeva, M. Forcada, Francis M. Tyers, and Juan Antonio Pérez-Ortiz. 2015. Evaluating machine translation for assimilation via a gap-filling task. In *EAMT*.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, W. Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *AAAI*.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *ArXiv*, abs/2006.14799.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the*

56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 643–653, Melbourne, Australia. Association for Computational Linguistics.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *ACL*.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3938–3948, Minneapolis, Minnesota. Association for Computational Linguistics.

A. R. Fabbri, Wojciech Kryscinski, Bryan McCann, R. Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Trans. Assoc. Comput. Linguistics*, 9:391–409.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019a. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *ACL*.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019b. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *ArXiv*, abs/1911.12237.

B. Goodrich, V. Rao, Mohammad Saleh, and Peter J. Liu. 2019. Assessing the factual accuracy of generated text. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–697, Melbourne, Australia. Association for Computational Linguistics.

T. Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. *ArXiv*, abs/1904.02792.

Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5094–5107, Online. Association for Computational Linguistics.

Wojciech Kryscinski, B. McCann, Caiming Xiong, and R. Socher. 2019. Evaluating the factual consistency of abstractive text summarization. *ArXiv*, abs/1910.12840.

J. Landis and G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33 1:159–74.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. *ACL*.

Luca Massarelli, F. Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, F. Silvestri, and S. Riedel. 2019. How decoding strategies affect the verifiability of generated text. *ArXiv*, abs/1911.03587.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Ramesh Nallapati, Bowen Zhou, C. D. Santos, Çaglar Gülçehre, and B. Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Jekaterina Novikova, Ondrej Dusek, A. Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *EMNLP*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *ArXiv*, abs/1606.05250.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–136, Sofia, Bulgaria. Association for Computational Linguistics.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! unsupervised metrics for reinforced summarization models. In *EMNLP/IJCNLP*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. *EMNLP*.

Oleg V. Vasilyev, Vedant Dharnidharka, and J. Bohannon. 2020. Fill in the blanc: Human-free quality estimation of document summaries. *ArXiv*, abs/2002.09836.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Tianyi Zhang, V. Kishore, Felix Wu, K. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675.

Z. Zhao, Shay B. Cohen, and B. Webber. 2020. Reducing quantity hallucinations in abstractive summarization. *ArXiv*, abs/2009.13312.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020. Boosting factual correctness of abstractive summarization. *ArXiv*, abs/2003.08612.

Dataset	Train	Dev	Test	Domain
XSUM	204,045	11,332	11,334	Short news
CNNDM	287,227	13,368	11,490	Long news
SAMSUM	14,732	818	819	Dialogues

Table 6: Summarization domains for evaluation.

A Appendices

A.1 Additional Details of Datasets

We provide dataset statistics for each of our domains in Table 6.

A.2 Evaluation Metric Details

QA-Based Quality Score. Given a source or reference document D and candidate summary S_i, QA-based evaluation metrics assign a generation quality score to S_i to measure the ability of a QA system by accurately answering questions generated from D or S_i. We use the SummaQA (Scialom et al., 2019) and FEQA (Durmus et al., 2020) metrics. For the SummaQA metric, questions are generated from the source document D and the candidate summary S_i is used as input to the QA system. Alternatively, FEQA generates questions from S_i and uses D to answer these questions.

The generation quality score is typically the aggregated F_1 score measuring the similarity between ground-truth answers for questions generated from D and the answers predicted by the QA system. SummaQA also generally includes the aggregated model confidence probabilities for predictions.

Masked LM Prediction (Cloze Task) Score. Given a source document D and candidate summary S_i, Cloze-based evaluation metrics assign a generation quality score to S_i by measuring the ability of a NLU system to accurately predict masked tokens in the source document, given access to the information in S_i. We use two variants of BLANC (Vasilyev et al., 2020), BLANC-Help and BLANC-Tune. BLANC-Help uses both D and S_i as input to a pretrained masked token prediction model, while BLANC-Tune only uses D as input to a model that has been finetuned on the candidate summary. Both metrics are aimed at capturing fluency, informativeness and factual correctness of summaries.

Semantic Similarity. Semantic similarity metrics measure the overlap between contextual embeddings of a source or reference document D and candidate summary S_i. We use BERTScore (Zhang et al., 2020), which has been shown to correlate better with human judgements of coherency than standard summarization metrics and similarly to n-gram metrics on factual consistency of CNNDM summaries (Wang et al., 2020).

Lexical Overlap. Finally, we test ROUGE (Lin, 2004), which is the standard metric used for evaluating summarization. ROUGE measures the n-gram overlap between a source or reference document D and candidate summary S_i. We evaluate results using ROUGE-1 and ROUGE-2, as well as ROUGE-L, which measures longest common subsequence overlap. We follow prior work that considered ROUGE in factual consistency evaluations (Wang et al., 2020), though it has also been previously noted that ROUGE can underweight good summarization examples (Novikova et al., 2017).

A.3 Simulated Data Transformations

We inject errors into reference summaries by first using a part-of-speech tagging model and named entity recognition system (spaCy)[5] to extract entities, verbs, and adjectives from these summaries. For each named entity, we keep track of the label type (e.g. ORG, GPE, etc). All datasets are comprised of English language articles or dialogues and summaries, and we use the spaCy English NLP models.

Intrinsic entity errors. To inject intrinsic entity errors into a summary S, we construct a dictionary of all unique entities appearing in the source document for S **only**, organized by entity label type. We then swap a random entity in the reference summary for a different entity of the same label type in the constructed dictionary.

Extrinsic entity errors. For extrinsic entity errors, we use the same dictionary construction for all unique entities appearing in **all** the corpus source documents. To change a random adjective, we use WordNet (Miller, 1995) to obtain the synsets for that adjective and swap the adjective for its antonym.

Pronoun entity errors. Pronoun errors are introduced with a preset list of commonly used pronouns. We randomly extract a pronoun set (e.g. she/her) from the text using the preset list and swap it with another random pronoun set (e.g. he/him).

Verb Negation. We use a rule-based system for verb negation based on verb tense, and predict tense based on the suffix and preceding words.

---

[5] `https://spacy.io/`

Reference	**Type**	**Description**	**Example**
Irish Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in his first meeting with Canadian Prime Minister Justin Trudeau in Dublin.	Intrinsic entity error (**int**)	An entity appearing in the source document is used incorrectly.	Canadian Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in his first meeting with Irish Prime Minister Justin Trudeau in Dublin.
Irish Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in his first meeting with Canadian Prime Minister Justin Trudeau in Dublin.	Extrinsic entity error (**ext**)	An entity appearing in the candidate summary does not appear in the source document.	French Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in his first meeting with Canadian Prime Minister Justin Trudeau in Dublin.
Irish Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in his first meeting with Canadian Prime Minister Justin Trudeau in Dublin.	Pronoun error (**pro**)	A pronoun in the candidate summary is used incorrectly. For example, (her/she instead of him/he).	Irish Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in her first meeting with Canadian Prime Minister Justin Trudeau in Dublin.
Irish Taoiseach (PM) Leo Varadkar has engaged in some "sock diplomacy" in his first meeting with Canadian Prime Minister Justin Trudeau in Dublin.	Negation error (**verb**)	There are verb negations in the candidate summary that contradict the source document.	Irish Taoiseach (PM) Leo Varadkar has not engaged in some "sock diplomacy" in his first meeting with Canadian Prime Minister Justin Trudeau in Dublin.
People who have been prescribed powerful anxiety or pain relief drugs are being warned about a new drug-driving law.	Sentiment error (**sent**)	An adjective or adverb appearing in the candidate summary contradicts the source document.	People who have been prescribed weak anxiety or pain relief drugs are being warned about a new drug-driving law.

Table 7: Table of possible factual errors.

We note that injecting a certain level of error into a summary will have varying effects depending on the average length of summaries for a corpus. We use the same methodology for each corpus to maintain consistency, but future work may explore length-controlled error injection based on the objectives of the evaluation.

A.4 Metric Implementation Details

For all metrics, we use the publicly shared implementations. Due to BERT context size constraints, we limit the length of document input sentences to 400 tokens for BLANC variants. We use Roberta-large for BERTScore.

A.5 T5 Training

We fine-tune the T5-base model (220M parameters) trained on news summaries for each domain using the AdaFactor optimizer (Shazeer and Stern, 2018) with a learning rate of 0.001 and a batch size of 8. The learning rate was tuned using ROUGE score on a dev set, and we experimented with learning rates in the range of [0.01,0.0001]. All other hyperparameters follow from the original T5 paper. Best performing models were trained using one random seed on NVIDIA V100 GPUs.

A.5.1 Human Annotation Layout

For human annotation of factual consistency in summaries, we show the source document, reference summary and a candidate summary that should be assessed for factuality. We then ask a factuality question with three choices:

- Yes (i.e. the summary is factual)
- No (i.e. the summary contains factual inconsistencies)
- Not Sure (i.e. the summary is too incoherent to judge)

If a summary is judged to be factually incorrect, annotators are allowed to select the number and type of errors they observe using a predefined list of factual errors. A screenshot of the error types and examples shown in the annotation task is given in Figure 2. For less obvious cases of factual inconsistency (for example when summaries contain locations or political figures that require regional background knowledge), we check factuality using external knowledge bases to ensure correctness of annotation. We also adhere to a strict binary notion of factuality in deciding cases where summaries are imprecise but ambiguous in terms of correctness, opting to label these summaries as factually inaccurate. If summaries are completely incoherent, we treat these summaries as having the highest level of factual inconsistency.

We validated the effectiveness of the setup by computing inter-annotator agreement of in-house expert annotators for 30 XSUM summaries. We achieve "fair" agreement of Krippendorff's α = 0.32 with 3 annotators and "moderate" agreement of α = 0.44 with 2 annotators (Landis and Koch, 1977; Ageeva et al., 2015). The remaining annotations are done by one in-house expert annotator.

Type of Error	**Gold Summary**	**Summary with Error**
Intrinsic Entity Error	Exhibition of 25 pictures on display at **the un** headquarters in **new york** .	Exhibition of 25 pictures on display at the **new york** headquarters in **the un** .
Extrinsic Entity Error	**Eoin collins** , five , eats everything from stones and sponges to sand .	**Vera** , five , eats everything from stones and sponges to sand.
Pronoun Entity Error	Elizabeth Sullivan fell in love with the 23-flavored soft drink 40 years ago , when **she** was in **her** 60s .	Elizabeth Sullivan fell in love with the 23-flavored soft drink 40 years ago , when **he** was in **his** 60s .
Verb Error	Charlotte Thomson **bought** old toilet block with a stunning sea view .	Charlotte Thomson **did not buy** old toilet block with a stunning sea view .
Sentiment Error	Camilla cut a **glamorous** figure in gold silk during a banquet last night .	Camilla cut a **boring** figure in gold silk during a banquet last night .
False Quote Error	Alex salmond vows to use alliance with labour to wring concessions . He said **the snp will try to vote down a tory government come what may .**	Alex salmond vows to use alliance with labour to wring concessions . He said **he would move at the end of the month .**

Figure 2: Examples of factual errors given in annotation task.

Dataset	**Level 1 Avg.**	**Level 2 Avg.**	**Level 3 Avg.**	**Avg. % Transformed** (L1/L2/L3/All)
XSUM (Entity)	0.59	1.14	1.61	58.84 / 76.44 / 86.28 / 73.85
XSUM (Non-Entity)	0.48	0.93	1.28	48.32 / 74.00 / 85.40 / 69.24
CNNDM (Entity)	0.75	1.48	2.17	74.92 / 85.68 / 94.48 / 85.03
CNNDM (Non-Entity)	0.50	1.05	1.62	79.44 / 93.32 / 97.04 / 89.93
SAMSUM (Entity)	0.59	1.16	1.70	58.96 / 77.32 / 87.56 / 74.61
SAMSUM (Non-Entity)	0.49	0.91	1.28	48.52 / 72.80 / 84.12 / 68.48

Table 8: Analysis of simulated diagnostic dataset (we average across 5 different sets (runs) of randomized transformations for the same 500 reference summaries). We provide results for the average number of induced factuality errors for factual inconsistency level 1 (L1), level 2 (L2) and level 3 (L3), as well as the percentage (%) of summaries that were transformed for each level and across all levels (All). We split the diagnostic dataset into two subsets based on whether simulated errors are related to entities (Entity) or non-entity changes like verb negation (Non-Entity).

DNN-driven Gradual Machine Learning for Aspect-Term Sentiment Analysis

Murtadha Ahmed, Qun Chen, Yanyan Wang, Youcef Nafa, Zhanhuai Li and Tianyi Duan
School of Computer Science, Northwestern Polytechnical University, Xi’an, China
Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University,
Ministry of Industry and Information Technology, Xi’an, China
{a.murtadha@mail.,chenbenben@,wangyanyan@mail., youcef.nafa@mail.,
lizhh@,tianyiduan@mail.}nwpu.edu.cn

Abstract

Recent work has shown that Aspect-Term Sentiment Analysis (ATSA) can be performed by Gradual Machine Learning (GML), which begins with some automatically labeled easy instances, and then gradually labels more challenging instances by iterative factor graph inference without manual intervention. As a non-i.i.d learning paradigm, GML leverages shared features between labeled and unlabeled instances for knowledge conveyance. However, the existing GML solution extracts sentiment features based on pre-specified lexicons, which are usually inaccurate and incomplete and thus lead to inadequate knowledge conveyance.

In this paper, we propose a Deep Neural Network (DNN) driven GML approach for ATSA, which exploits the power of DNN in feature representation for gradual learning. It first uses an unsupervised neural network to cluster the automatically extracted features by their sentiment orientation. Then, it models the clustered features as factors to enable implicit knowledge conveyance for gradual inference in a factor graph. To leverage labeled training data, we also present a hybrid solution that fulfills gradual learning by fusing the influence of supervised DNN predictions and implicit knowledge conveyance in a unified factor graph. Finally, we empirically evaluate the performance of the proposed approach on real benchmark data. Our extensive experiments have shown that the proposed approach consistently achieves the state-of-the-art performance across all the test datasets in both unsupervised and supervised settings and the improvement margins are considerable.

1 Introduction

r_i	s_{ij}	text
r_1	$s_{1.1}$	**service** was awful mostly because **staff** were overwhelmed.
	$s_{1.2}$	The **staff** should be a bit more friendly.
r_2	$s_{2.1}$	We ordered **lamb** which was perfectly cooked and tasted awesome.
	$s_{2.2}$	The **food** was well-prepared and presented.

Table 1: A running example: r_i denotes a review and $s_{i.j}$ denotes a sentence.

Aspect-Term Sentiment Analysis (ATSA) aims at inferring the sentiment polarity towards a particular aspect in a sentence (Hu and Liu, 2004; Pontiki et al., 2016). ATSA is important for many applications (e.g., e-commerce and social media), where the sentimental opinions in reviews can be leveraged to create value for businesses and customers. In ATSA, an aspect-term, also called target, is explicitly mentioned in a review. For instance, consider the running example shown in Table 1, r_1 evaluates the restaurant through two explicit aspects *service* and *staff*. The goal of ATSA is then to detect the respective sentiment polarities expressed towards these two aspects.

Up to now, the state-of-the-art solutions for ATSA have been built upon various DNN models. The earlier solutions were usually equipped with an attention mechanism (Tang et al., 2016b; Wang et al., 2016; Tang et al., 2016a; Ma et al., 2017; Chen et al., 2017; Li et al., 2018; Wang et al., 2018; Tang et al., 2019). They mostly attempted to learn aspect-related semantic representation of an input sentence. Recently, ATSA has experienced a considerable shift towards pre-trained language models (Sun et al., 2019; Tang et al., 2019; Karimi et al., 2020). Despite the effectiveness of these approaches, unfortunately their efficacy heavily relies on large quantities of accurately labeled data, which require intensive human labor.

To alleviate the burden of manual labeling, a solution based on the paradigm of Gradual Machine Learning (GML) has recently been proposed for

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 488–497
August 1–6, 2021.

ATSA (Wang et al., 2021). First proposed for entity resolution in (Hou et al., 2019, 2020), GML can enable effective machine labeling without the requirement for manual intervention. Given a classification task, GML begins with some easy instances, which can usually be automatically labeled by the machine with high accuracy, and then gradually reasons about the labels of its more challenging instances by factor graph inference. As a non-i.i.d (Independent and Identically Distributed) learning paradigm, GML leverages shared features between labeled and unlabeled instances for knowledge conveyance . However, the existing GML solution for ATSA relies on pre-specified lexicons to extract sentiment features. Its limitation is twofold: 1) sentiment lexicons may be inaccurate and incomplete; 2) a shared feature must explicitly appear in both instances. However, explicit features cannot capture the implicit similarity between instances and thus lead to inadequate knowledge conveyance. Consider the running example in Table 1. Unfortunately the word *well-prepared* is not included in most of the existing lexicons. It can also be observed that the instances $s_{2.1}$ and $s_{2.2}$ do not share any explicit feature, while *perfectly cooked* and *well-prepared* have very similar meanings and can thus serve as an implicit common feature.

Recently, DNN models have been proven to be very powerful in feature representation for many NLP tasks, where the features with the same semantic context are mapped to close points in the latent space (Devlin et al., 2018). For instance, the words *"cooked"* and *"well-prepared"* are usually represented by two points close to each other because they are semantically very close. Unfortunately, the existing embedding models are designed to map the features semantically, regardless of their sentiment orientation. Therefore, they may map two features with opposite polarities (e.g., *"good"* and *"bad"*) to two close points in the embedding space, which raises a challenge to be directly applied to feature extraction for ATSA.

In this paper, we propose a novel DNN-driven GML approach for ATSA. It essentially exploits DNN to sentimentally map the features of aspect-terms into different *polarity indicators*, and models them as shared factors in a factor inference graph to enable implicit knowledge conveyance. To this end, we first combine the sentiment lexicon and dependency parser-based relations, which are readily available, to generate aspect-opinion words. Secondly, we use an unsupervised neural network to filter the aspect-irrelevant and unsentimental words from an input sentence. Finally, the resulting weighted sentences, which can be considered to be purely sentimental, are used to learn polarity indicators. The model is trained to reconstruct the weighted sentence through a linear combination from polarity indicators. To leverage labeled training data, we also present a hybrid GML solution that fulfills gradual learning by fusing the influence of supervised DNN predictions and implicit knowledge conveyance in a unified factor graph.

Our main contributions can be summarized as follows:

1. We propose a DNN-driven GML approach for ATSA, which can effectively exploit the power of DNN in feature representation for GML;
2. We present an unsupervised attention-based neural network to cluster the features of aspect-terms by their sentimental orientation;
3. We present a hybrid GML solution for ATSA, which fulfills gradual learning by fusing the influence of supervised DNN predictions and implicit knowledge conveyance in a unified factor graph.
4. We empirically validate the efficacy of the proposed approach on benchmark data. Our extensive experiments have shown that the proposed approach consistently achieves the state-of-the-art performance across all the test datasets in both unsupervised and supervised settings and the improvement margins are considerable.

2 Related work

Aspect-Term Sentiment Analysis is a sub-task of aspect-based sentiment analysis, which aims to detect the sentiment polarity in response to a particular aspect in a sentence (Hu and Liu, 2004; Pontiki et al., 2016). Traditional machine learning techniques (Kiritchenko et al., 2014; Castellucci et al., 2014) proposed to learn SVM classifiers based on different features such as n-grams, non-contiguous n-grams and lexicon features. In comparison, the DNN-based models equipped with an attention mechanism have been shown to be more effective on ATSA (Tang et al., 2016b; Wang et al., 2016; Tang et al., 2016a). Following this trend, researchers have resorted to more sophisticated attention mechanisms to refine neural ATSA models (Ma et al., 2017; Chen et al., 2017; Li et al., 2018; Wang et al., 2018; Tang et al., 2019). To improve performance, they essentially attempted to explicitly capture the importance of each context word by learning aspect-related representation of an input sentence. SenHint (Wang et al., 2019) proposed to integrate DNN predictions and linguistic hints in a joint framework. Recently, ATSA has experienced a considerable shift towards pre-trained language models (Sun et al., 2019; Tang et al., 2019; Karimi et al., 2020). Unfortunately, the efficacy of these models heavily relies on labeled training data, which may not be readily available in real-scenario.

From unsupervised perspective, earlier solutions (Alvarez-López et al., 2016; Hutto and Gilbert, 2014) proposed to detect the polarities of aspect-terms based on lexicon rules. The authors of (Schouten et al., 2017) proposed a mechanism of spread activation for aspect-based polarity detection. More recently, the authors of (Wang et al., 2021) proposed an unsupervised solution based on GML for ATSA. However, the existing GML solution extracts features based on sentiment lexicons, which may not be accurate nor complete and thus lead to inadequate knowledge conveyance.

The idea of mapping features into different clusters has been investigated with different purposes. The authors of (Iyyer et al., 2016) proposed to learn a set of descriptors representing the fictional relationship between two characters changes over time, and (He et al., 2017) proposed to learn a set of aspect representatives from the corpora. Unfortunately, none of them investigated how to cluster implicit features by their polarity orientation.

3 Preliminaries

3.1 Task Definition

We formulate the task of aspect-term sentiment analysis as follows:

Definition 3.1 *[Aspect-Term Sentiment Analysis]. Let* $x = (r, s, t)$ *be a target unit, where* r *denotes a review,* s *a sentence in the review and* t *an aspect-term associated with the sentence. Given a set of target units,* X*, the goal of ATSA is to infer the sentiment polarity of each target unit in* X*.*

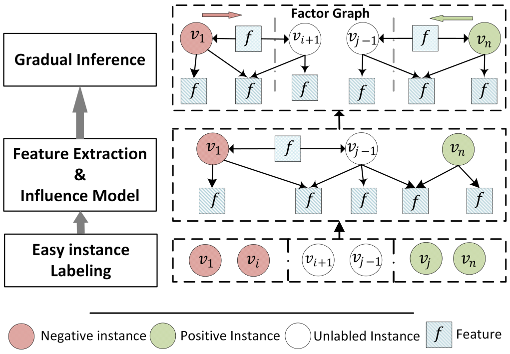

Figure 1: GML Paradigm Overview

3.2 GML Paradigm Overview

Our solution is built upon the non-i.i.d learning paradigm of GML (Hou et al., 2019, 2020). As shown in Figure 1, GML consists of the following three steps:

3.2.1 Easy Instance Labeling.

Given a classification task, it is usually very challenging to accurately label all the instances in the task without good-coverage training examples. However, the work can become much easier if we only need to automatically label some easy instances in the task. In real scenarios, easy instance labeling can be performed based on the simple user-specified rules or the existing unsupervised learning techniques. GML begins with the observations provided by the labels of easy instances. Therefore, high accuracy of automatic machine labeling on easy instances is critical for its ultimate performance on a given task.

For ATSA, this paper uses the unsupervised algorithm of spread activation (Schouten et al., 2017) to label easy instances. An instance is considered as easy if its resulting dominate label meets a pre-specified threshold.

3.2.2 Feature Extraction and Influence Modeling.

Feature serves as the medium for knowledge conveyance. This step extracts the common features shared by labeled and unlabeled instances. To facilitate effective knowledge conveyance, it is desirable that a wide variety of features are extracted to capture as much information as possible. For each extracted feature, this step also needs to model its influence over the labels of its relevant instances.

For ATSA, we extract two types of features: sentimental feature and relational feature. Relational feature, which has been well studied in (Wang et al., 2021), represents the explicit sentimental connection between sentences within the same review. In the running example, due to the absence of any shift word between s_{11} and s_{12}, their polarities can be supposed to be similar. In this paper, we focus on how to enable implicit knowledge conveyance by leveraging DNN for automatic extraction of sentimental features.

3.2.3 Gradual Inference.

This step gradually labels the instances with increasing hardness in a task. Since the scenario of gradual learning does not satisfy the i.i.d assumption, gradual learning is fulfilled from the perspective of evidential certainty. Gradual learning is conducted over a factor graph, which consists of the labeled and unlabeled instances and their common features, by iterative inference. At each iteration, it chooses to label the unlabeled instance with the highest degree of evidential certainty. The iteration is repeatedly invoked until all the instances in a task are labeled.

Given a factor graph, G, GML defines the probability distribution over its variables V as follows:

$$P_w(V) = \frac{1}{Z_w} \prod_{v \in V} \prod_{f \in F_v} \phi_f(v) \prod_{f' \in F'} \phi_{f'}(v_i, v_j), \tag{1}$$

where F_v denotes the set of sentimental features associated with the variable v, F' denotes the set of relational features, $\phi_f(v)$ denotes the factor associated with v and f, $\phi_{f'}(v_i, v_j)$ denotes the factor associated with the relational feature f', and Z is a partition function, i.e. normalization constant. To effectively learn the factor weights without access to the true labels of unlabeled variables, V_I, GML minimizes the negative log marginal likelihood given the observed labels of labeled variables, Λ, as follows:

$$\hat{w} = arg \min_{w} -log \sum_{V_I} P_w(\Lambda, V_I). \tag{2}$$

A scalable approach for gradual inference on ATSA has been presented in (Wang et al., 2021). First, the unlabeled variables are sorted according to their evidential support. Then, the top-m unlabeled variables are considered as the candidates for probability inference. To reduce the invocation frequency of factor graph inference, an efficient algorithm is used to approximate entropy estimation on m candidates and select the top-k most promising variables for factor graph inference. Finally, the probabilities of the selected k variables are inferred in the subgraphs of G. Since the inference process of the DNN-driven GML is very similar to what was presented in (Wang et al., 2021), its technical details are omitted here due to space limit.

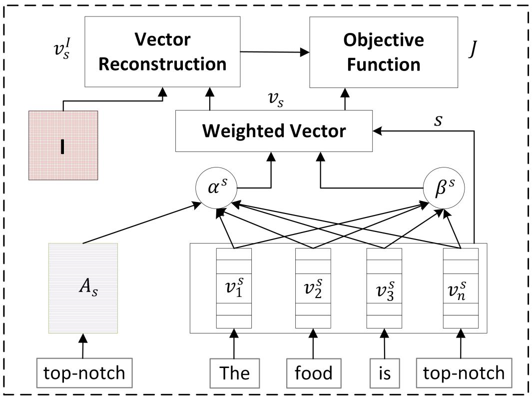

Figure 2: DNN for Implicit Feature Extraction.

4 DNN-driven GML

In this section, we first present an unsupervised neural network to extract implicit sentimental features. Then, we describe the unsupervised DNN-driven GML that integrates implicit features into the process of gradual inference. Finally, we describe the hybrid GML solution that fuses the influence of DNN predictions and implicit features for gradual learning.

4.1 Implicit Feature Extraction by DNN

The purpose of implicit feature extraction is to learn a set of polarity indicator embeddings I $\in \mathbb{R}^{k \times d}$, where k is the number of indicators, which can be leveraged to capture the similar features between instances. Each indicator represents a set of features that often occur in the contexts with the same polarity.

Specifically, for each input sentence s and its aspect term t, we first generate a set of aspect-opinion words, denoted by A_s. Then, we use A_s to construct a weighted vector $v_s \in \mathbb{R}^d$ that can be read as the sentimental representation of the input sentence given the target of t. To this end, we propose an attention-based unsupervised neural network to filter the sentence by down-weighting aspect-irrelevant and unsentimental information. The model is trained by reconstructing v_s as a linear combination of indicator embeddings from I. The architecture of the proposed DNN has been presented in Figure 2.

4.1.1 The Input

The input to our model is a couple of sentence s and its aspect-opinion words A_s. We use dependency-based parse tree to generate aspect-opinion words (*modifiers*) (Hu and Liu, 2004), then leverage the adjective words and those detected by the lexicon to construct A_s. Considering the running example in Table 1, the sentiment words of s_{11} are *awful* and *overwhelmed*. Suppose that we have a feature embedding matrix $\mathrm{L} \in \mathbb{R}^{c \times d}$, where c is the vocabulary size and d is the embedding dimension. Each word is then associated with a real-valued embedding $v_i \in \mathbb{R}^d$ from L representing its feature vector (Mikolov et al., 2013):

$$s = \{v_1^s, v_2^s, ..., v_n^s\}, \tag{3}$$

$$A_s = \{v_1^a, v_2^a, ..., v_m^a\}, \tag{4}$$

where $s \in \mathbb{R}^{n \times d}$ and n is the sentence length, while the input sentiment $A_s \in \mathbb{R}^{m \times d}$ and m is the number of aspect-opinion words.

4.1.2 Attention-based Sentimental Representation

For each input sentence s, we construct a weighted vector v_s to capture the sentimental information in response to the aspect t. To this end, we apply two attention mechanisms to filter away the irrelevant information. The first one attempts to down-weight non-sentimental words, while the second one is a self-attention to attend to aspect-relevant information (He et al., 2017).

Specifically, the first attention layer takes both the sentence s and its opinion words A_s as an input. Conceptually, we first compute the global sentiment vector v^a by averaging the word embeddings of A_s, and then use it to weight each word embedding v_i^s in s as follows:

$$v^a = \frac{1}{m} \sum_{v_i^a \in A_s} v_i^a, \tag{5}$$

$$o_i = v_i^{s\top} \cdot \mathrm{U} \cdot v^a, \tag{6}$$

where the symbol $\cdot$ stands for element-wise dot product, while $\mathrm{U} \in \mathbb{R}^{d \times d}$ is the transformation matrix (i.e., to be learned during training) between the global sentiment vector v^a and the input sentence s. Next, we apply a softmax layer to yield a non-negative weight for each word in s as follows:

$$\alpha_i^s = \frac{\exp(o_i)}{\sum_{j=1}^{n} \exp(o_j)}, \tag{7}$$

where the value of α_i^s can be read as the probability of each word in the sentence s being a sentiment word.

Although we have computed the sentimental importance for each word in s, but not all the sentiment words are contextually related to the aspect. Therefore, we apply another self-attention mechanism that takes only the sentence s as input. To compute each word's probability of being aspect-relevant information, namely β_i^s, we follow the same steps in the first attention layer. The only difference is that the global sentiment v^a in Equation 6 is replaced by the global context, which is simply computed by averaging the word embeddings of the input sentence s itself.

Finally, we sum both attention layer outputs, $\alpha^s \in \mathbb{R}^n$ and $\beta^s \in \mathbb{R}^n$, and use it to construct the weighted vector v_s as follows:

$$v_s = s^\top \cdot (\alpha^s + \beta^s), \tag{8}$$

in which the resulting weight vector v_s can be read as the aspect-relevant sentiment representation of the input sentence s.

4.1.3 Unsupervised Training

Now that we have obtained the aspect-relevant sentiment representation of an input sentence s, we explain how to learn its polarity indicators using a variant of dictionary learning. Considering the matrix of indicators I as a dictionary, we attempt to approximate v_s as a linear combination of items from I.

Formally, for each aspect-specific vector v_s, we compute a corresponding vector v_s^k over k polarity indicators by simply reducing v_s from d dimensions to k dimensions through a softmax layer as follows:

$$v_s^k = softmax(W \cdot v_s + b), \tag{9}$$

where $W \in \mathbb{R}^{k \times d}$ denotes a weight matrix and b denotes the bias, both of which are supposed to be learned during training. Note that v_s^k can be read as the probability that the input sentence s belongs to each indicator. Then, we reconstruct the representation vector by taking a weighted average over the polarity indicators as follows:

$$v_s^I = \mathrm{I}^\top \cdot v_s^k. \tag{10}$$

Since the objective is to make v_s^I similar to v_s, we apply the widely used contrastive max-margin

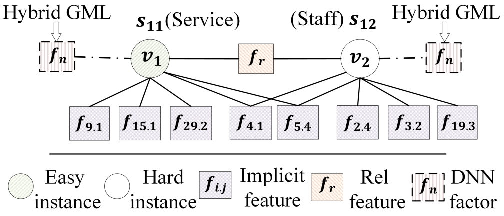

Figure 3: Factor graph of r_1 in the running example.

objective function (Iyyer et al., 2016; Ahmed et al., 2020). To that end, we randomly sample some sentences from the training data as negative samples, and compute the weighted vector v_s for each sample using Equation 8.

Specifically, the objective is a hinge loss that minimizes the inner product between the representation vector of v_s and the reconstructed vector v_s^I for the negative samples, while simultaneously maximizes their inner product for other samples in the training data. Formally, the hinge loss is defined as:

$$J(\theta) = \sum_{s \in D} \sum_{s^- \in D^-} max\{0, 1 - v_s^I \cdot v_s + v_s^I \cdot v_{s^-}\}, \tag{11}$$

where θ represents the model parameters, D represents the set of training data and D^- the subset of negative samples. Note that $\theta = \{I, U, W, b\}$.

To discourage the model from learning similar indicators, we add a regularization term to the objective function J that penalizes redundancy in the matrix of polarity indicators (Iyyer et al., 2016):

$$M(\theta) = ||\mathrm{I} \cdot \mathrm{I}^\top - \mathrm{Y}||, \tag{12}$$

where Y denotes the identity matrix. The final training objective L is then represented by the weighted sum of J and M as follows:

$$L(\theta) = J(\theta) + \lambda M(\theta), \tag{13}$$

where λ is a hyper-parameter that controls the magnitude of the regularization term.

4.2 Unsupervised GML Solution

Now that we have already learned a set of polarity indicators, we describe how they can serve as implicit features for gradual inference. Given a sentence, we first estimate its aspect-specific vector v_s using Equation 8, then compute the cosine similarity between v_s and each polarity indicator in I. It thus results in a list of scores in the form (*indicator index, similarity score*) representing how the aspect-term's features are close to each indicator. We sort the scores and use top-k corresponding indicators as representative features. We scale up the similarity score to 10 to augment the number of features and meanwhile avoid polarity conflict between features. As shown in Figure 3, $f_{5.4}$ represents an indicator feature with the index of 5 and the similarity scale of 4.

Considering the instance of s_{11} in Figur 3, s_{11}'s top-5 indicators and their similarity scores are (5, 0.44), (29, 0.2), (15, 0.16),(4, 0.13) and (9, 0.13). Then, its representative features are $F_{s_{11}} = \{f_{5.4}, f_{29.2}, f_{15.1}, f_{4.1}, f_{9.1}\}$. In gradual inference, we restrict two instances to share an implicit feature if and only if they are similar to the same indicator with the same score scale. For instance, in Figure 3, s_{11} and s_{12} share the indicators 4 and 5 with the same score scale of 1 and 4 respectively.

Given ATSA task, each aspect-term within the same review is represented by a variable. The evidence variables are assigned constant values 0 or 1 representing their polarity labeling, while the values of the inference ones are inferred based on G. The factor of an implicit feature f_e in Equation 1 is defined by:

$$\varphi_{f_e}(v_i) = \begin{cases} 1 & v_i = 0; \\ e^{w_{f_e}} & v_i = 1; \end{cases} \tag{14}$$

where v_i denotes a variable having the feature f_e, and w_{f_e} denotes the weight of f_e. Note that the weight w_{f_e} is initialized to zero, but needs to be learned in the process of gradual inference.

4.3 Hybrid GML Solution

In the hybrid solution, we model the influence of DNN outputs by DNN factors, denoted by f_n, as shown in Figure 3. In this paper, we have implemented the hybrid solution by the state-of-the-art BERT-based DNN of HP-SUM for ATSA (Karimi et al., 2020). However, other DNN models can be fused in the same way. Since supervised learning is usually more accurate than unsupervised learning, we also label easy instances by supervised DNN predictions. In other words, we consider the instances with the most extreme probabilities predicted by HP-SUM as easy ones to kick-start gradual inference.

In factor graph, the DNN factor f_n of a variable corresponding to the aspect-term unit, (r, s, t) is defined by:

$$\varphi_{f_n}(v_i) = \begin{cases} 1 & v_i = 0; \\ e^{w_{f_n}} & v_i = 1; \end{cases} \tag{15}$$

	Method	Rest 14	Rest 15	Rest 16	Lap 14	Lap 15	Lap 16
Unsupervised	VADER	79.65	76.21	75.18	69.72	74.31	68.31
	LEX-SYN	80.84	75.82	76.77	70.17	75.81	69.64
	SPD-ACT	81.89	76.02	81.06	74.84	77.15	73.77
	Lexicon GML	83.83	80.22	85.64	82.25	82.42	80.31
	DNN-driven GML	**87.05**	**81.19**	**86.31**	**85.84**	**84.05**	**81.62**
Attentive	ATAE-LSTM	88.56	76.72	81.19	79.45	77.11	73.13
	IAN	87.98	77.09	78.37	76.68	77.36	74.87
	RAM	90.0	76.26	87.72	81.87	80.61	76.27
	GCAE	88.55	78.64	87.87	80.81	80.82	80.83
	AEN-Glove	89.86	79.14	86.82	83.79	80.73	77.17
	TNet-LF	90.36	80.74	87.95	83.22	81.17	76.75
Bert-based	AEN-BERT	92.74	84.64	90.07	90.11	90.74	84.89
	BERT-SPC	93.76	83.49	91.72	90.62	88.74	86.64
	HP-SUM	93.39	88.29	94.76	93.6	90.34	87.85
	Hybrid GML	**95.51**	**88.67**	**95.6**	**94.91**	**91.74**	**88.88**

Table 2: Comparative Evaluation Results. Rest and Lap stand for Restaurant and Laptop domains respectively. The respective best accuracies in the unsupervised and supervised setting are highlighted in **bold**.

in which w_{f_n} denotes factor weight. The value of w_{f_n} is defined as

$$w_{f_n} = \ln(\frac{p}{1-p}), \tag{16}$$

where p is the probability output of DNN (i.e., estimated by HP-SUM) of a target t being positive in the sentence s. It can be observed that $W_{f_n} > 0$ if $p > 0.5$; otherwise, if $p < 0.5$, then $w_{f_n} < 0$.

5 Empirical Evaluation

We evaluate our solution on six benchmark datasets provided by the SemEval ABSA task across the years 2014, 2015 and 2016 for the Restaurant and Laptop domains (Pontiki et al., 2016). Note that the original datasets are three-way labels (i.e., positive, negative and neutral). Since this paper focuses on binary polarity classification, we only include the reviews with *positive* or *negative* labels in our experiments. Furthermore, we have trained the polarity indicators for the restaurant and laptop domains on unlabeled corpus collected from Citysearch and Amazon, which have also been widely used in previous work (Zhao et al., 2010; Ahmed et al., 2020).

For unsupervised training, we initialized word vectors by word2vec. We implemented GML inference using the Numbskull library [1], a Python NUMBA-based Gibbs sampler. Our GML implementation optimizes the parameters by Adam with the learning rate of 0.001. On all the test datasets, we set the number of polarity indicators k to 50, and the number of negative samples to 20. In the spread activation algorithm for easy instance labeling, the easiness threshold is set to 0.7 for all datasets. For the hybrid GML solution, the easy instances are the top-30% ones with most extreme probabilities as predicted by supervised DNN. For each instance, the associated implicit features are the top-5 polarity indicators' scores scaled to 10.

The compared unsupervised techniques include: (1) **LEX-SYN** (Alvarez-López et al., 2016). It infers polarity based on lexicon and syntactic dependency analysis; (2) **VADER** (Hutto and Gilbert, 2014). A rule-based approach; (3) **SPD-ACT** (Schouten et al., 2017). It infers polarity by spread activation; (4) **Lexicon-based GML** (Wang et al., 2021). The GML solution built upon sentiment lexicons.

The compared supervised DNN models include the latest BERT-based models as well as traditional attention-based models: (1) **ATAE-LSTM** (Wang et al., 2016). An attention-based LSTM; (2) **IAN** (Ma et al., 2017). An interactive attention model; (3) **RAM** (Chen et al., 2017). A deep memory model; (4) **TNet-LF** (Li et al., 2018). A target-specific transformation network; (5) **GCAE** (Xue and Li, 2018). A gated convolutional network; (6) **AEN** (Song et al., 2019). An attentional encoder network. AEN has two variants: AEN-Glove that uses Glove as feature embedding input, and

---

[1] https://github.com/HazyResearch/numbskull

AEN-BERT based on the pre-trained model BERT fine-tuning; (7) **BERT-SPC** (Song et al., 2019). A pseudo-sentence (i.e., sentence and aspect) BERT-based approach; (8) **HP-SUM** (Karimi et al., 2020). A BERT-based model equipped with parallel aggregation and hierarchical aggregation modules.

Note that among the listed DNN models, the last 3 models (i.e. AEN-BERT, BERT-SPC, and HP-SUM) were built upon the latest pre-trained BERT.

5.1 Main Results

We average the three runs' performances and report the detailed evaluation results in Table 2. We have the following observations: (1) **the unsupervised DNN-driven GML consistently gives the best accuracy compared to the unsupervised alternatives across all datasets.** The performance advantage of the DNN-driven GML over the lexicon-based GML suggests that a carefully-designed implicit feature mechanism can effectively perform better than lexicon-based explicit features for ATSA; (2) the unsupervised DNN-driven GML is even competitive with the traditional supervised attention-based models; (3) the supervised BERT-based approaches indeed achieve better performance than both traditional attention-based DNNs and unsupervised GML. However, their efficacy depends on the fine-tune phase that requires an access to the labeled training data, which are not available in the unsupervised setting; (4) **The hybrid GML solution consistently achieves the state-of-the-art performance across all datasets.** It improves the best performance by almost 2% on two datasets and 1%-2% on four out of six datasets. In light of the well recognized challenge of ATSA, these improvements are indeed considerable.

Illustrative Examples. To illustrate the effectiveness of implicit features, we present the features of the running example in Table 3. It can be observed that *overwhelmed*, *well-prepared*, and *presented* in r_{11} and r_{22} respectively are not captured by the lexicon, and r_{12} contains the context misunderstanding of *friendly*. Even though r_{11} and r_{12} do not share any explicit information, the negative context of *friendly* is very close to *overwhelmed*; they thus share the implicit features $f_{4.1}$ and $f_{5.4}$. Likewise, *well-prepared* in r_{22} is very close to *perfectly cooked* in r_{21}, and they share the implicit features $f_{7.4}$ and $f_{15.3}$.

r_{ij}	Features		GML Labeling	
	LEX	DDN-based	LEX	DNN
r_{11}	Awful	$\{f_{9.1}, f_{15.1}, f_{29.2}, f_{4.1}, f_{5.4}\}$	False	True
r_{12}	Friendly	$\{f_{2.4}, f_{3.2}, f_{4.1}, f_{5.4}, f_{19.3}\}$	False	True
r_{21}	Perfectly, Awesome	$\{f_{7.4}, f_{15.3}, f_{21.3}, f_{27.4}, f_{30.2}\}$	True	True
r_{22}	-	$\{f_{7.4}, f_{15.3}, f_{12.1}, f_{1.1}, f_{19.1}\}$	False	True

Table 3: Illustrative examples of implicit features.

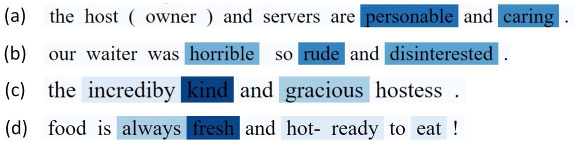
(a) the host (owner) and servers are personable and caring .
(b) our waiter was horrible so rude and disinterested .
(c) the incrediby kind and gracious hostess .
(d) food is always fresh and hot- ready to eat !

Figure 4: Visualization of the attention weights.

5.2 Effectiveness of Sentiment Weighting

We illustrate the effectiveness of the designed attention mechanisms in terms of attending to aspect-relevant sentiment information, and understanding the context. We retrieve samples from the datasets and visualize their attention weights in Figure 4, in which the deeper the color, the more importance a word has. It can be observed that the aspect-opinion words are weighted among the others and the model effectively attends to the sentiment words that are not in the lexicon (e.g., personable, gracious in (a) and (c) respectively). Since the sentiment words dominate the sentence representation in Equation 8, this indeed encourages the model to sentimentally learn the representations of polarity indicators.

6 Conclusion

In this work, we propose a novel DNN-driven GML approach for ATSA that can effectively leverage common implicit features for knowledge conveyance. Our extensive experiments have shown that the proposed approach consistently achieves the state-of-the-art performance in both unsupervised and supervised setting. For future work, it is noteworthy that the DNN-driven GML approach is potentially applicable to other classification tasks; the technical solutions however need further investigation.

Funding: The work was supported by the National Key Research and Development Program of China (2018YFB1003400), National Natural Science Foundation of China (61732014, 61672432, 61472321 and 61502390), Fundamental Research Funds for the Central Universities (Program No. 3102019DX1004) and Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2018JM6086).

References

Murtadha Ahmed, Qun Chen, and Zhanhuai Li. 2020. Constructing domain-dependent sentiment dictionary for sentiment analysis. *Neural Computing and Applications*, pages 1–14.

Tamara Alvarez-López, Jonathan Juncal-Martínez, Milagros Fernández-Gavilanes, Enrique Costa-Montenegro, and Francisco Javier González-Castano. 2016. Gti at semeval-2016 task 5: Svm and crf for aspect detection and unsupervised aspect-based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 306–311.

Giuseppe Castellucci, Simone Filice, Danilo Croce, and Roberto Basili. 2014. Unitor: Aspect based sentiment analysis with structured learning. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 761–767.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–397.

Boyi Hou, Qun Chen, Jiquan Shen, Xin Liu, Ping Zhong, Yanyan Wang, Zhaoqiang Chen, and Zhanhuai Li. 2019. Gradual machine learning for entity resolution. In *The World Wide Web Conference*, pages 3526–3530.

Boyi Hou, Qun Chen, Yanyan Wang, Youcef Nafa, and Zhanhua Li. 2020. Gradual machine learning for entity resolution. *IEEE Transactions on Knowledge and Data Engineering, (accepted online available)*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2020. Improving bert performance for aspect-based sentiment analysis. *arXiv preprint arXiv:2010.11731*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 437–442, Dublin, Ireland. Association for Computational Linguistics.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *10th International Workshop on Semantic Evaluation (SemEval 2016)*, pages 19–30, San Diego, California. Association for Computational Linguistics.

Kim Schouten, Onne Van Der Weijde, Flavius Frasincar, and Rommert Dekker. 2017. Supervised and unsupervised aspect category detection for sentiment analysis with co-occurrence data. *IEEE transactions on cybernetics*, 48(4):1263–1275.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of NAACL-HLT*, pages 380–385.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Jialong Tang, Ziyao Lu, Jinsong Su, Yubin Ge, Linfeng Song, Le Sun, and Jiebo Luo. 2019. Progressive self-supervised attention learning for aspect-level sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 557–566.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967.

Yanan Wang, Qun Chen, Murtadha Ahmed, Boyi Hou, and Zhanhuai Li. 2019. Joint inference for aspect-level sentiment analysis by deep neural networks and linguistic hints. *IEEE Transactions on Knowledge and Data Engineering, (accepted, online available)*.

Yanyan Wang, Qun Chen, Jiquan Shen, Boyi Hou, Murtadha Ahmed, and Zhanhuai Li. 2021. Aspect-level sentiment analysis based on gradual machine learning. *Knowledge-Based Systems*, Volume 212.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523.

Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65, Cambridge, MA. Association for Computational Linguistics.

Error Detection in Large-Scale Natural Language Understanding Systems Using Transformer Models

Rakesh Chada, Pradeep Natarajan, Darshan Fofadiya, Prathap Ramachandra
Amazon Alexa AI
{rakchada, natarap, darshanf, prathara}@amazon.com

Abstract

Large-scale conversational assistants like Alexa, Siri, Cortana and Google Assistant process every utterance using multiple models for domain, intent and named entity recognition. Given the decoupled nature of model development and large traffic volumes, it is extremely difficult to identify utterances processed erroneously by such systems. We address this challenge to detect domain classification errors using offline Transformer models. We combine utterance encodings from a RoBERTa model with the N-best hypothesis produced by the production system. We then fine-tune end-to-end in a multi-task setting using a small dataset of human-annotated utterances with domain classification errors. We tested our approach for detecting misclassifications from one domain that accounts for <0.5% of the traffic in a large-scale conversational AI system. Our approach achieves an F1 score of 30% outperforming a bi-LSTM baseline by 16.9% and a standalone RoBERTa model by 4.8%. We improve this further by 2.2% to 32.2% by ensembling multiple models.

1 Introduction

Conversational assistants such as Cortana, Google Assistant, Alexa and Siri leverage multiple machine learning models and services like automatic speech recognition (ASR), natural language understanding (NLU), entity resolution (ER), utterance routing and text-to-speech (TTS). In particular, the NLU system is modularized into multiple domains such as Music, Movies, Weather etc. These domain teams then train one-vs-all domain, intent and named entity recognition models independently. At run time, a re-ranker is used to sort the output of these independent models and route the utterance to the right domain. This is a well-known design pattern used in large scale conversational assistants (Sarikaya, 2017; Su et al., 2018).

This modularized architecture allows scaling of NLU systems to support multiple domains catering to billions of diverse utterances. However, it is extremely challenging to isolate the source of error in utterances where the system fails, both due to the large traffic volume and the asynchronous nature of model development and updates by multiple independent teams. In particular, we focus on identifying utterances with domain classification errors, which we call false rejects (FR). Such errors can arise due to errors in individual domain classification models, re-ranker or routing algorithm and are extremely hard to isolate. To illustrate this challenge, consider a domain X that accounts for 0.5% of the traffic in a conversational assistant receiving Y utterances every week. Also assume that 20% of the utterances belonging to X are falsely routed to another domain. This implies that we have 0.1% of the total traffic that are FRs. Given a large traffic, manually sifting through the entire traffic to find such utterances is prohibitively expensive and infeasible.

Previous work in Feedback-Based Self-Learning in Large-Scale Conversational AI Agents (Ponnusamy et al., 2020) uses implicit user feedback as a signal to generate automated reformulations. However, for low traffic domains, the low frequency of occurrence for utterances at the tail end of the distribution does not allow for automated reformulations at scale. Finetuning of pretrained language representation models which leverage inductive transfer learning (Howard and Ruder, 2018) have shown marked performance improvement with very small training datasets. Language models with bidirectional encoding such as BERT (Devlin et al., 2019; Liu et al., 2019)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 498–503
August 1–6, 2021.

which are pretrained using a masked language modeling task have shown further improvements on the finetuning task due to its deep bidirectional encoder-based language representation. Further, larger models with 100s of billions of parameters like GPT3 (Brown et al., 2020) have demonstrated significant performance improvements with increase in model size. However, leveraging such models in production systems is challenging due to latency constraints.

We address this challenge by leveraging the power of BERT based pretrained language representation models in an offline setting. We combine utterance encodings from RoBERTa with embeddings of the N-Best hypothesis for the same utterance from the production system and conduct end-to-end finetuning in a multitask learning setting (Caruana, 1997) with a small, manually curated training set containing FRs. Multitask learning enhances language representation by discovering synergies between various finetuning tasks. We then use this model to sift through all the traffic and identify other FRs.

We tested our approach for identifying FRs from one domain in a large-scale conversational assistant. Compared to the baseline F1 score of 13% using a bi-LSTM model, our approach achieves an F1 of 25% using RoBERTa encodings. We further improve performance to 28% using N-best output, 30% using multitask learning, and to 32.3% using an ensemble of multitask models.

The rest of the paper is organized as follows: first, we describe the production NLU system; next we present details of our FR detection system and finally we present experimental results and conclusion.

2 Architecture of Large-Scale NLU Systems

Large-scale NLU systems are typically modularized into domain-specific modules where a domain is typically something like Books, Music, Shopping etc. As illustrated in Figure 1, each domain consists of several sub-components that are served by models for Domain, Intent and Named Entity Recognition. Let's consider an example user utterance U: "play hello by adele". Assume there are M domains. This utterance is sent to all of them and each domain then processes the utterance by sending it through its sub-modules described below.

(1) A Domain Classifier is a one-vs-all classifier indicating if the utterance belongs to that domain. For the example utterance U, a good classifier should output a high score for the positive class in the "Music" domain and output a high score for the negative class in all the other domains.

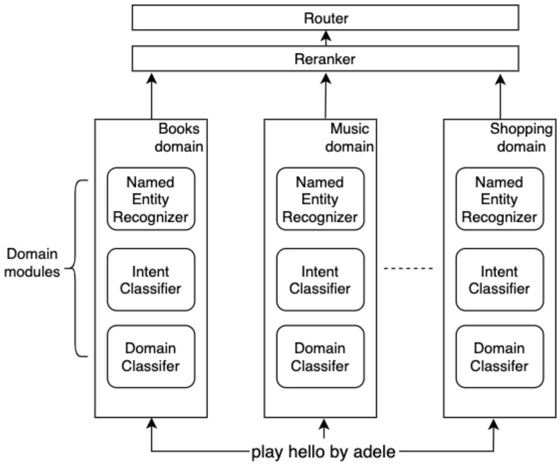

Figure 1: NLU System Architecture

(2) An Intent Classifier is a multi-class model that provides intent specific scores for each of the intents belonging to a target domain. For the example utterance U, a good intent classifier should produce a high score for the intent “PlayMusic” when run in the "Music" domain.

(3) A Named Entity Recognizer (NER) that identifies named entities present in an utterance. For the utterance U, a good classifier in the "Music" domain should select "hello" and "adele" tokens as probable named entities.

To minimize run-time latency constraints, most large-scale production systems use a combination of deterministic artifacts (like rules) and simple models like MaxEnt (Berger et al., 1996) for these tasks. Once the domains process the utterance, the outputs are fed into a predictive model called re-ranker to obtain a sorted list. This sorted output is then used by the Router to send the utterance to the right domain for processing.

Due to the high volumes of traffic and the modular architecture with several domains, detecting the presence of errors (for instance, a domain misclassification) and isolating their sources is very challenging. Errors can occur and accumulate through every step of the processing flow highlighted in Figure 1. We next present our approach to identify one such error called False Reject (FR) where an utterance does not get routed to the correct domain by the production system.

3 False Reject Detection

For a given domain D, the task of identifying false rejects (FR) involves analyzing all traffic routed to a domain other than D and identifying utterances that should instead have been routed to D. The key challenges are the availability of only a small number of annotated examples of such FR and the extreme imbalance in our test set between FRs and utterances that are routed correctly.

3.1 Baseline System

As a baseline model, we used a bi-LSTM (Schuster and Paliwal, 1997) model to classify whether an utterance belongs to a domain D or not. Utterances tokenized into words and characters are leveraged in the form of their pre-trained Glove embeddings (Sakketou and Ampazis, 2020) and passed to a bi-LSTM layer. The stateful output of the bi-LSTM layer is passed through a fully-connected layer which outputs the probability of an utterance belonging to domain D as per equation (1) where X_i represent the stateful output of the bi-LSTM layers, β the learned weights of the fully-connected layer and Y_i represents the classes.

$$\Pr(Y_i = 0) = \frac{e^{-\beta.X_i}}{1+e^{-\beta.X_i}} \tag{1}$$

$$\Pr(Y_i = 1) = 1 - \Pr(Y_i = 0) = \frac{1}{1 + e^{-\beta.X_i}}$$

The training data used for fine-tuning comes from a low-cost human annotation effort where a small set of utterances are marked as “False Reject” (FR) or not by expert annotators. The False Rejects (FRs) for a domain D are then identified by applying the bi-LSTM based model on a test set of utterances that match the production traffic distribution. The utterances that are assigned to a domain D by the bi-LSTM based model but are not assigned to a domain D by the production model are filtered as the final FR candidates.

3.2 Pre-trained transformer models

Pre-trained transformer models such as BERT, GPT, T5 have proven to be quite effective at multitude of NLU tasks. Specifically, they can be adapted to a downstream task by fine-tuning on a small-sized dataset.

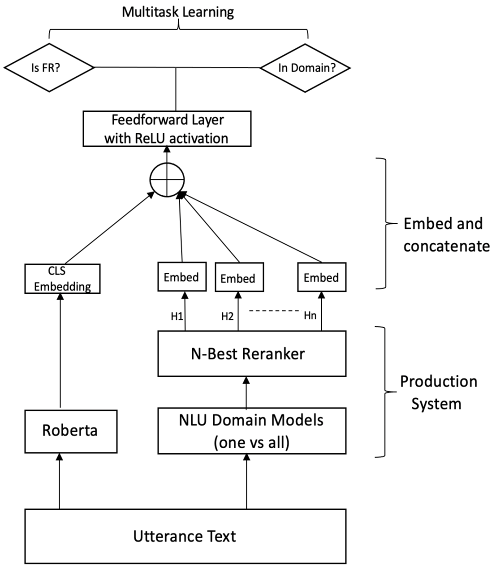

Figure 2: Transformer model leveraging N-Best from domain production models for FR detection

We leverage this strength for the task of FR detection. Specifically, we fine-tune a RoBERTa-based model for a binary classification problem in which given a domain D, we want the model to output the probability of an utterance belonging to that domain (represented as $\Pr(Y_i = 1)$ in equation (1). Instead of the bi-LSTM output, here X_i represents the weights from the pre-trained model passed through a fully-connected layer. The model is trained to optimize the cross-entropy loss using the same dataset as the baseline system. We use the Adam optimizer with bias correction (Kingma and Ba, 2017) and a learning rate of $2e^{-5}$ with a warmup for 1/10th of the total number of training steps.

3.3 Leveraging N-Best Embeddings

The NLU system described in Section 3 produces an N-best hypothesis that represents the top N domains a user utterance likely belongs to. We tested incorporating this information into the FR detection model using two steps.

Embed & Concatenate (N-best module): To leverage the N-best hypotheses, we choose top N hypotheses from the re-ranker, embed each of them into a 6-dimensional vector and concatenate them into a 6N-dimensional vector. This vector is further concatenated with the 1024 dimensional [CLS] token embedding from the pre-trained RoBERTa model. This (1024 + 6N) dimensional vector is then passed through a feed-forward layer to feed to the output layers.

Multi-task output: The model is trained to output predictions for two tasks. The first task is to determine if the utterance belongs to a domain or not. This is a generic task and is not specifically geared towards identifying if the production system falsely rejected an utterance. To this, we add a second classification task to specifically classify if the utterance is falsely rejected or not. These tasks could also share useful information across one another through their input hidden features.

The final model architecture that includes these components can be seen in Figure 2. A tokenized utterance is sent through the pre-trained RoBERTa model followed by the N-best module. The output from the N-best module is fed separately to two feed-forward layers to produce two outputs. The model is trained for 3 epochs using the same hyperparameters as the baseline transformer. The training took about 2 hours on a p3.dn24x large instance.

4 Results

We use a dataset with a FR to non-FR ratio of 1:15. This is not the true production distribution[1].

	Precision	**Recall**	**F1**
bi-LSTM	20.3	9.7	13.1
RoBERTa	50.1	16.3	25.2
RoBERTa + N-Best single task model	23.4	35.3	28.1
RoBERTa + N-Best multitask model	31.0	30.0	30.0
RoBERTa + N-Best multitask model (ensemble)	**37.4**	**28.3**	**32.2**

Table 1: Performance comparison of different FR detection techniques

However, it captures the imbalance and the low resource setting that we want to address in our framework. We hold out 15% of this dataset for validation. We have processed the data so that users are de-identified.

Table 1 compares the performance of different techniques we tested. We see a 12.1% absolute F1-score improvement from bi-LSTM to RoBERTa illustrating the value of pre-training for problems with extreme class imbalance and small training sets.

Incorporating the N-Best embeddings produces a further 3.1% improvement illustrating the value of using information from the production system itself to detect errors in its output. Using a multitask learning setting improves it further by 1.9% showing how similar tasks share learning representations. Ensembling produces an additional 2.2% improvement that is consistent with other machine learning results.

Figure 3 presents a comparison of the precision-recall curves for RoBERTa and RoBERTa+N-Best multitask models. While the performance is comparable at high precisions, at lower values, the latter significantly outperforms. This is because the standalone RoBERTa model identifies a small number of frequent FRs, while the RoBERTa+N-Best multitask model is able to identify a greater variety of FR patterns without compromising precision significantly. Thus, the latter model offers a better tradeoff in identifying FRs that can then be used as data to train the production domain classifiers.

4.1 Production model improvements

Our RoBERTa-large based model is not directly usable in production due to latency constraints. As a result, we use a human-in-the-loop system where we leverage our model by running it in an offline setting to identify false reject candidates. These false reject candidates are sent for human annotation and the resulting annotations are fed as training data to the production model. We've noticed that the annotation throughput of true false rejects is improved by 5x due to the curated candidates provided by the offline transformer model. Furthermore, because these annotations are used to further train the production model, the False Rejects are reduced by about 20%.

---

[1] For confidentiality reasons, we don't use the true production distribution.

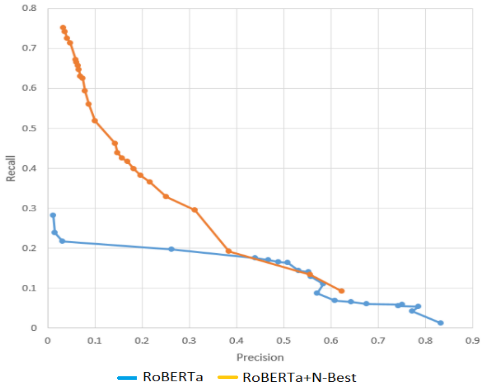

Figure 3: Precision-Recall Curve for RoBERTa and RoBERTa+N-Best embedding multitask

5 Conclusion

Our work presents a system for leveraging the power of computationally intensive, but accurate pre-trained language models to identify errors in a large-scale conversational assistant through offline analysis. Our results also demonstrate the effectiveness of such models in problems with large class imbalances. Specifically, we achieve an F1 score of 25.2% outperforming a bi-LSTM baseline by 12.1%. Further, we show that combining the output of the production system with pre-trained language models produce significant improvements (of 4.8 F1 points). As future work, we plan to leverage even larger models such as Megatron (Shoeybi et al., 2019) and T5 (Raffel et al., 2019) to achieve further improvements for the FR detection task.

References

[1] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI 2018.

[2] Adam L. Berger, Stephen A. Della Pietra and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. In Proceedings Association for Computational Linguistics, Volume 22, Number 1, pages 39-71.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. 2017. Attention Is All You Need. Computing Research Repository, arXiv:1706.03762. Version 5.

[4] Caruana, R. 1997. Multitask Learning. In Springer Machine Learning 28, 41–75, 10.1023/A:1007379606734.

[5] Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan and Spyros Matsoukas. 2018. A Re-ranker Scheme for Integrating Large Scale NLU models. Computing Research Repository, arXiv:1809.09605. Version 1.

[6] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Machine Learning Research Repository, arXiv:1910.10683. Version 3.

[7] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. Machine Learning Research Repository, arXiv:1412.6980. Version 9.

[8] Flora Sakketou and Nicholas Ampazis. 2020. A constrained optimization algorithm for learning GloVe embeddings with semantic lexicons, Knowledge-Based Systems. Volume 195, 105628, ISSN 0950-7051.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota. Association for Computational Linguistics

[10] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia. Association for Computational Linguistics.

[11] M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. In IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, 10.1109/78.650093.

[12] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. Computing Research Repository, arXiv:1909.08053. Version 4.

[13] Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo and Ruhi Sarikaya. 2020. Feedback-Based Self-Learning in Large-Scale Conversational AI Agents. Proceedings of the AAAI Conference on Artificial Intelligence, 34(08), 13180-13187.

[14] Ruhi Sarikaya. 2017. The Technology Behind Personal Digital Assistants: An overview of the system architecture and key components. In IEEE Signal Processing Magazine, vol. 34, no. 1, pp. 67-81, 10.1109/MSP.2016.2617341.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever and Dario Amodei. 2020. Language Models are Few-Shot Learners. Computing Research Repository, arXiv:2005.14165. Version 4.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. Computing Research Repository, arXiv:1907.11692. Version 1.

OutFlip: Generating Out-of-Domain Samples for Unknown Intent Detection with Natural Language Attack

DongHyun Choi[1,2], Myeong Cheol Shin[1], EungGyun Kim[1], and Dong Ryeol Shin[2]

[1]Kakao Enterprise
{*heuristic.c,index.sh.jason.ng*}*@kakaoenterprise.com*
[2]Sungkyunkwan University
drshin@skku.edu

Abstract

Out-of-domain (OOD) input detection is vital in a task-oriented dialogue system since the acceptance of unsupported inputs could lead to an incorrect response of the system. This paper proposes OutFlip, a method to generate out-of-domain samples using only in-domain training dataset automatically. A white-box natural language attack method HotFlip is revised to generate out-of-domain samples instead of adversarial examples. Our evaluation results showed that integrating OutFlip-generated out-of-domain samples into the training dataset could significantly improve an intent classification model's out-of-domain detection performance[1].

1 Introduction

Intent classification is crucial for task-oriented dialogue systems such as Google DialogFlow or Amazon Lex. It is vital for an intent classifier not only to map an input utterance into the correct label but also to detect *out-of-domain* (OOD) inputs. An accepted OOD input will lead the dialogue system to give erroneous responses.

Approaches for OOD detection in text classification could be classified into two major categories. Outlier detection approaches (Fei and Liu, 2016; Hendrycks and Gimpel, 2017; Shu et al., 2017; Lin and Xu, 2019; Yan et al., 2020; Xu et al., 2020) try to find out the boundaries of known classes in feature space. They need no labeled OOD dataset, but it is hard for them to deal with boundary cases. $(n+1)$-way classification approaches (Kim and Kim, 2018; Larson et al., 2019; Ryu et al., 2018; Zheng et al., 2020) train classifiers for OOD detection using (pseudo-)labeled OOD samples. In practice, it is difficult and expensive to collect a large number of labeled OOD samples with an open-world environment.

This paper proposes OutFlip, a method to generate OOD samples from in-domain training dataset automatically. For a given training dataset T and a reference intent classification model M which is trained with T, the OutFlip generates a set of OOD samples O. The generated OOD samples O could be used to train M iteratively to improve its OOD detection performance. Since the OutFlip does not require any modifications to the model architecture, it could be used with other OOD detection approaches to further improve the OOD detection performance.

The generated OOD samples should satisfy two conditions. First, they should be "hard-enough"; if the generated examples are too easy to distinguish from in-domain intents, they will be useless in training the OOD detector. Second, they should not belong to any in-domain intents. With a given reference model M and a set of in-domain labels I, this could be considered as finding a sentence $\mathbf{x}_o$ with truth label $y \notin I$ and model classification $y' \in I$. In this point of view, the OOD sample generation task could be considered as a variant of natural language attack on model M; the goal of natural language attack on M is to find a $\mathbf{x}_a$ with truth label $y \in I$ and model classification $y' \neq y$. We revised HotFlip (Ebrahimi et al., 2018), a natural language attack method, to generate such OOD samples.

Our evaluation results showed that the generated OOD samples could significantly improve the OOD detection performances of the reference models. We also showed that applying OutFlip with other OOD detection approaches could further improve the model's OOD detection performance. The evaluation results also suggest that the generated OOD samples could train the models other than the reference model to improve their OOD detection perfor-

---

[1] The source code is available at https://github.com/kakaoenterprise/OutFlip

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 504–512
August 1–6, 2021.

mances.

Our contributions are summarized as follows:

- We proposed OutFlip, a simple and efficient OOD sample generation method using only in-domain training samples.
- We experimentally showed the effectiveness of our proposed approach using the intent classification benchmarks.
- We showed that the generated OOD samples could also improve the OOD detection performances of models other than the reference model.

2 Related Work

Previous OOD detection works could be classified into two major categories. Outlier detection approaches find boundaries of known classes in feature space. Fei and Liu (2016) computes a center for each class and transforms each document into a vector of similarities to the center. A binary classifier is built using the transformed data for each class. For deep learning-based systems, Hendrycks and Gimpel (2017) proposed the baseline of using softmax score as a threshold. Shu et al. (2017) trained the intent classifier using the sigmoid function and used the standard distribution to set each class's score threshold. Lin and Xu (2019) first trained the classifier using Large Margin Cosine Loss (LMCL) (Nalisnick et al., 2018), and applied Local Outlier Factor (Breunig et al., 2000) to detect the OOD inputs. Yan et al. (2020) proposed a semantic-enhanced Gaussian mixture model to gather vectors of the same classes closely. Xu et al. (2020) calculated the mean and covariance of training samples for each class and used Mahalanobis distance as a distance function.

$(n + 1)$-way classification approaches train the intent classifier with one additional class, where $(n + 1)$-th class represents the unseen intent. Kim and Kim (2018) proposed joint learning for in-domain and out-of-domain speeches. Larson et al. (2019) manually collected OOD samples to train intent classifiers. Ryu et al. (2018) generated OOD feature vectors using generative adversarial network (Goodfellow et al., 2014) to train an OOD detector. Since the approach proposed in Ryu et al. (2018) only works on continuous feature space, it highly depends on the feature encoder, which transforms inputs into feature vectors. Zheng et al. (2020) also generated OOD feature vectors, but they also used unlabeled examples to enhance classification performance further. Although the $(n + 1)$-way classification approaches are easy to adopt without modification in the classification model, it is incredibly costly and time-consuming to collect the appropriate OOD samples.

The proposed OutFlip automatically generates OOD samples using the only in-domain training set, significantly reducing the cost of manually collecting OOD samples. Also, the OutFlip does not depend on the feature encoder.

The goal of adversarial attack in text classification is to fool a given text classification model M, by generating an adversarial example $\mathbf{x}_a$ with truth label y and model classification $y' \neq y$. Many successful attacks first take a correctly classified example $\mathbf{x}$ and replace its important words or characters to get an adversarial sample $\mathbf{x}_a$. In a white-box scenario, the attacker has access to the target model's structure; thus, the important word or characters could be easily selected by inspecting the gradient of model M. HotFlip (Ebrahimi et al., 2018) estimates the best change of characters by maximizing the first-order approximation of the change in the loss.

In a black-box scenario, the attacker is not aware of the model or training data; the attacker is only capable of querying the target model with supplied inputs and obtaining the output predictions and their confidence scores. Alzantot et al. (2018) randomly selects a word from sentence $\mathbf{x}$ and selects a suitable replacement word that has a similar semantic meaning. Jin et al. (2020) proposed a word importance score, which is used to find the word to be replaced. Li et al. (2020) applied BERT (Devlin et al., 2018) pre-trained language model to find a replacement word.

The proposed OutFlip first extracts important words using the algorithm proposed in Jin et al. (2020), and applies a variant of HotFlip (Ebrahimi et al., 2018) to generate OOD samples which are hard to distinguish from the in-domain intents by the given reference model.

3 Proposed Approach

In this section, the proposed OOD sample generation approach OutFlip is described in more detail.

3.1 HotFlip

We first introduce the white-box adversarial example generation method HotFlip (Ebrahimi et al., 2018). Let M be a text classification model, V be the word vocabulary set, $\mathbf{x} = \{x_1; ...; x_n\}$ be a sentence with n words where $x_i \in \{0,1\}^{|V|}$ denotes one-hot vector representing the i-th word, and $L_M(\mathbf{x}, \mathbf{y})$ be the loss of M on input $\mathbf{x}$ with true output $\mathbf{y}$. For a given sentence $\mathbf{x}$, a flip of the i-th word from w_a to w_b is represented by the following vector:

$$\vec{v}_{ib} = (\vec{0}; ..; (0, .., -1, .., 1, 0, .., 0)_i; ..; \vec{0}) \quad (1)$$

where -1 and 1 are in the corresponding positions for words w_a and w_b in the word vocabulary, respectively. A first-order approximation of the change in loss $L_M(\mathbf{x}, \mathbf{y})$ can be obtained from a directional derivative along this vector:

$$\nabla_{\vec{v}_{ib}} L_M(\mathbf{x}, \mathbf{y}) = \nabla_x L_M(\mathbf{x}, \mathbf{y})^T \cdot \vec{v}_{ib} \quad (2)$$

Then, the HotFlip chooses the vector with the biggest increase in loss:

$$\begin{gathered} \max \nabla_x L_M(\mathbf{x}, \mathbf{y})^T \cdot \vec{v}_{ib} = \max_{ib} \frac{\partial L_M^{(b)}}{\partial x_i} - \frac{\partial L_M^{(a)}}{\partial x_i} \\ \text{subject to } sim(w_a, w_b) \geq T_{sim} \\ \text{and } POS(w_a) = POS(w_b) \end{gathered} \quad (3)$$

where T_{sim} is a similarity threshold between two words, and $POS(w_a)$ is the Part-of-Speech tagging of w_a. The two constraints are added to ensure that $\mathbf{x}_a$ is semantically similar to the original input $\mathbf{x}$. With equation 3, the HotFlip could determine the flip position i and the replacement word w_b.

3.2 OutFlip

For a given reference model M and an in-domain sample $\mathbf{x}$ with true output $\mathbf{y}$, the main idea of OutFlip is to flip the most important word of $\mathbf{x}$, $w_M(\mathbf{x})$, to a semantically different word, while *minimizing* the change of loss $L_M(\mathbf{x}, \mathbf{y})$. By doing so, the OutFlip expects to get a sample $\mathbf{x}_o$ whose truth label is different from the truth label of $\mathbf{x}$, while the model classifications of $\mathbf{x}_o$ and $\mathbf{x}$ are the same.

The word importance score proposed in Jin et al. (2020) is defined as follows:

$$I_{x_i}(M, \mathbf{x}) = o_y(M, \mathbf{x}) - o_y(M, \mathbf{x}_{\setminus x_i}) \quad (4)$$

Algorithm 1 OutFlip

```
Input In-domain training corpus $T = \{\mathbf{x}_1, ..., \mathbf{x}_t\}$,
      in-domain labels $Y$, vocabulary $V$, reference
      model $M$, similarity threshold $T_{sim}$
Output A set of OOD samples $O$
for $y \in Y$ do
    for $\mathbf{x}$ with truth label $y$ do
        Calculate $w_M(\mathbf{x})$
    end for
    $C_T(y) \leftarrow$ Top 5 most frequent $w_M(\mathbf{x})$
end for

for $\mathbf{x} \in T$ do
    $y \leftarrow$ truth label of $\mathbf{x}$
    if $w_M(\mathbf{x}) \in C_T(y)$ then
        $i \leftarrow$ the position of $w_M(\mathbf{x})$
        Sort $V$ in ascending order of $\frac{\partial L_M^{(b)}}{\partial x_i}$
        Candidate $\leftarrow$ Top 1% of $V$
        Remain only words whose similarity
          with $w_M(\mathbf{x})$ is less than $T_{sim}$
        Randomly select $w_b$ among candidates
        Replace $w_M(\mathbf{x})$ with $w_b$ to get $\mathbf{x}_o$
        if $M$ classifies $\mathbf{x}_o$ to $y$ then
            Add $\mathbf{x}_o$ to $O$
        end if
    end if
end for
```

where y is the truth label of $\mathbf{x}$, $o_y(M, \mathbf{x})$ is the logit output of the target model M for label y, and $\mathbf{x}_{\setminus x_i}$ is the sentence after masking x_i. The most important word $w_M(\mathbf{x})$ is defined as the word with the largest importance score in $\mathbf{x}$.

For each in-domain label y of the training dataset T, we define Core Class Token (CCT) $C_T(y)$ as the top 5 most frequent $w_M(\mathbf{x})$ among the training samples with truth label y. Since the importance score is calculated based on the reference model M, the OutFlip could select a wrong token as the most important token due to the model error. If the OutFlip flips such a word, the generated sentence's truth label will remain unchanged, leading to an erroneous OOD sample. To prevent such case, the OutFlip simply disregards $\mathbf{x}$ during OOD generation process if $w_M(\mathbf{x}) \notin C_T(y)$.

In summary, the OutFlip chooses the replace-

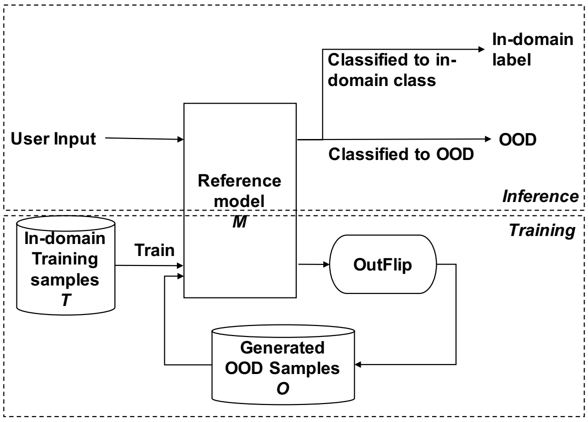

Figure 1: Applying the OutFlip to iteratively train the reference model M with newly generated OOD samples.

ment word w_b using the following equation:

$$\min \nabla_x L_M(\mathbf{x},\mathbf{y})^T \cdot \vec{v}_{ib} = \min_{ib} \frac{\partial L_M^{(b)}}{\partial x_i} - \frac{\partial L_M^{(a)}}{\partial x_i}$$
$$\text{subject to } sim(w_a, w_b) \leq T_{sim}$$
$$\text{and } w_a = w_M(\mathbf{x})$$
$$\text{and } w_a \in C_T(y) \qquad (5)$$

Since we do not need the generated OOD samples to be fluent, the part-of-speech condition is removed. The OutFlip randomly chooses w_b among the top 1% of the vocabulary in the ascending order of the loss change to generate more diverse samples. We used cosine similarity as the similarity measure.

The truth label of the generated sample $\mathbf{x}_o$ could be an in-domain label different from y by chance. The OutFlip checks the model classification result of $\mathbf{x}_o$ to see if it remains the same as $\mathbf{x}$. If the classification result changes, the OutFlip disregards $\mathbf{x}_o$. Algorithm 1 shows the pseudocode of the proposed OutFlip.

3.3 Iteratively Populating OOD samples

The reference model M could be iteratively trained with the generated OOD samples to improve its OOD detection performance. Figure 1 shows the overall framework. For each iteration, the set of generated OOD samples O is randomly split into training and dev set and used for the next train iteration.

Since the OutFlip does not require any change in model architecture, the OutFlip could be applied independently with other OOD detection algorithms that require modifications on model architecture or loss function, such as Shu et al. (2017) or Lin and Xu (2019). In such cases, those OOD detection algorithms are applied to the examples classified as in-domain to filter out the OOD samples further.

Dataset	**ATIS**	**SNIPS**	**Kakao**
Language	English	English	Korean
Vocab Size	938	12,054	22,831
Avg. Length	11.21	9.36	8.88
# Train	4,478	13,784	90,692
# Dev	500	700	11,310
# Test	893	-	12,711
# of Classes	18	7	48
Is Balanced	X	O	O

Table 1: Dataset statistics.

4 Experiments

In this section, experimental settings and evaluation results are shown.

4.1 Datasets

Experiments are conducted on 3 real task-oriented dialogue datasets, SNIPS (Coucke et al., 2018), ATIS (Hemphill et al., 1990) and Kakao dialogue corpus[2] (Choi et al., 2020). SNIPS and ATIS are well-known English benchmarks. Kakao dialogue corpus is a Korean intent classification benchmark. We evaluated the proposed OutFlip with the Kakao dataset to see if it could be applied to different languages. Table 1 summarizes the statistics of the datasets. The ATIS dataset is highly imbalanced; more than 70% samples belong to one class, while three classes have less than 10 samples. The SNIPS and Kakao datasets are relatively balanced.

Since the SNIPS dataset does not have a test set, we randomly selected 30% of the training set and used them as the test set.

4.2 Baselines

We implemented two sentence encoders to show the generality of the proposed approach. LSTM (Hochreiter and Schmidhuber, 1997)-based encoder applies one-layer BiLSTM with output dimension 128 on the word embeddings of the given input; a self-attention layer with attention dimension 10 is followed to get the feature vector. CNN-based model applies the algorithm proposed in Kim

---

[2] Since the Kakao dataset is not publicly available, we contacted the authors to get the Kakao dataset and Korean GloVe vectors.

Dataset	ATIS			SNIPS			Kakao		
% of known intents	25%	50%	75%	25%	50%	75%	25%	50%	75%
MSP$_{cnn}$	64.61	71.07	62.00	34.24	63.26	73.14	66.23	82.32	89.60
MSP$_{lstm}$	65.82	71.52	61.18	34.12	63.30	73.28	64.23	81.37	88.28
DOC$_{cnn}$	61.99	57.82	38.46	49.66	70.76	77.16	71.61	84.72	**90.55**
DOC$_{lstm}$	62.76	58.15	38.35	49.77	71.11	77.37	63.36	78.86	85.22
LMCL$_{cnn}$	71.67	74.89	68.73	61.51	**84.37**	88.31	80.27	**87.15**	**90.60**
LMCL$_{lstm}$	72.25	**77.90**	**73.18**	69.52	83.32	87.53	76.31	**85.79**	89.39
OutFlip$_{cnn}$	**74.18**	**79.23**	**69.37**	**79.20**	**84.25**	**88.99**	**81.78**	85.45	86.85
OutFlip$_{lstm}$	**73.85**	74.50	68.30	**79.26**	84.00	**88.67**	**81.96**	84.64	86.42

Table 2: Comparisons of the OutFlip and previous OOD detection works. The top 2 results for each metric are marked in bold. T_{sim} is set to 0.3, and the OutFlip is applied for three iterations.

(2014). More precisely, one-dimensional convolutions with kernel sizes 2, 3, 4, 5 and filter size 32 are applied on top of the word embeddings. The results are max-pooled to get the feature vector. For both encoders, a dense layer is applied to the feature vector to get the logit of each class.

We also implemented three baseline OOD detection systems, as follows:

1. **Maximum Softmax Probability (MSP)** (Hendrycks and Gimpel, 2017) considers the maximum softmax probability of a sample as the rejection score. If the probability is below a certain threshold, the sample is classified as OOD. We used the threshold of 0.5, as the authors suggested.
2. **Deep Open Classification (DOC)** (Shu et al., 2017) replaces softmax with sigmoid activation as the final layer to calculate the score for each class separately. It also calculates the threshold for each class through a statistical approach.
3. **Large Margin Cosine Loss (LMCL)** (Lin and Xu, 2019) replaces the softmax loss with large margin cosine loss (Nalisnick et al., 2018), to force the model to maximize interclass variance and minimize intra-class variance. After training, it applies Local Outlier Factor (LOF) (Breunig et al., 2000) on training features vectors to detect outliers as OOD. We set the scaling factor $s = 30$ and cosign margin $m = 0.35$, following the authors.

By combining two feature encoders and three baseline OOD detection systems, we implemented eight baseline reference models, six with an OOD detection system and two without.

Pre-trained Embeddings	Accuracy
GloVe (Pennington et al., 2014)	83.17 %
Korean GloVe (Choi et al., 2020)	51.33 %

Table 3: Evaluation results of GloVe embeddings on the language-independent set of word analogy corpus.

4.3 Experimental Setup

Word embeddings are initialized with GloVe (Pennington et al., 2014) pre-trained word vectors. We downloaded the pre-trained embeddings containing 1.9M words trained on 42B tokens from the author's homepage. For Korean, Korean pre-trained GloVe embedding vectors proposed in Choi et al. (2020) are used. The dimensions of both pre-trained embeddings are 300.

We removed some classes from the train/dev set during training and integrated them back during testing, following the evaluation settings of Fei and Liu (2016); Shu et al. (2017); Lin and Xu (2019). We varied the number of known intents in the training dataset as 25%, 50%, and 75% of the intents, and used all intents for testing. We randomly select known intents by weighted random sampling without replacement in the training set. Note that the samples belonging to the unknown intents are removed during training and validation.

Following Fei and Liu (2016); Shu et al. (2017); Lin and Xu (2019), macro F1 score is used to evaluate the models. For each known intent selection, the F1 score for each class is calculated separately. Then the results are macro-averaged across all classes. We reported the average of 10 random known intent selections for each evaluation.

For each OutFlip iteration, 90% of the generated OOD samples are added to the training set, and

Dataset	ATIS			SNIPS			Kakao		
% of known intents	25%	50%	75%	25%	50%	75%	25%	50%	75%
MSP$_{cnn}$	77.04 **(+12.43)**	74.66 **(+3.59)**	62.44 **(+0.44)**	78.02 **(+43.78)**	84.48 **(+21.22)**	89.66 **(+16.52)**	82.65 **(+16.42)**	85.74 **(+3.42)**	87.26 **(-2.34)**
MSP$_{lstm}$	71.37 **(+5.55)**	76.17 **(+4.65)**	61.97 **(+0.79)**	79.88 **(+45.76)**	83.98 **(+20.68)**	88.92 **(+15.64)**	82.22 **(+17.99)**	85.21 **(+3.84)**	86.25 **(-2.03)**
DOC$_{cnn}$	73.45 **(+11.46)**	59.03 **(+1.21)**	41.24 **(+2.78)**	80.72 **(+31.06)**	85.05 **(+14.29)**	89.42 **(+12.26)**	82.20 **(+10.59)**	85.45 **(+0.73)**	86.44 **(-4.11)**
DOC$_{lstm}$	70.20 **(+7.44)**	59.69 **(+1.54)**	41.09 **(+2.74)**	81.99 **(+32.22)**	85.08 **(+13.97)**	89.69 **(+12.32)**	80.14 **(+16.78)**	81.28 **(+2.42)**	80.78 **(-4.44)**
LMCL$_{cnn}$	74.13 **(+2.46)**	76.35 **(+1.46)**	68.82 **(+0.09)**	81.04 **(+19.53)**	85.90 **(+1.53)**	90.10 **(+1.79)**	81.81 **(+1.54)**	84.75 **(-2.40)**	85.74 **(-4.86)**
LMCL$_{lstm}$	75.44 **(+3.19)**	76.37 **(-1.53)**	71.34 **(-1.84)**	80.86 **(+11.34)**	84.56 **(+1.24)**	88.76 **(+1.23)**	82.24 **(+5.93)**	84.19 **(-1.60)**	85.35 **(-4.04)**

Table 4: Evaluation results of the baselines with the OutFlip applied. T_{sim} is set to 0.3, and the OutFlip is applied for three iterations. Small numbers below the macro F1 score represents the performance improvement compared to the same baseline without applying the OutFlip.

the remaining 10% are added to the dev set. The populated train/dev sets are used for the next train iteration.

Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.001 is used to train the model. The training batch size is set to 128. Exponential learning rate decay with a decay rate of 0.8 is applied for every two epochs. On each epoch, the trained classifier is evaluated against the dev set, and the training stops when the dev accuracy is not improved for five consequent epochs.

4.4 Evaluation Results

Table 2 shows the evaluation results of the proposed OutFlip and other baseline systems. As can be observed from the table, the proposed OutFlip outperforms other baselines when the number of known intents is small. The small number of known intents is the most similar case to real-world applications, since in the open-world environment, the number of unknown intents is much larger than the number of known intents. The OutFlip also gives comparable results for ATIS and SNIPS corpus with the larger number of known intents.

For the Kakao corpus, the OutFlip performance is lower compared to the other baselines. To figure out the reason, the qualities of English GloVe embeddings and Korean GloVe embeddings are compared. We used 4 out of 14 categories in the word analogy corpus (Mikolov et al., 2013) for fair comparison; `capital-common-countries`, `capital-world`, `currency` and a subset of `family`. We removed all the syntactic questions since they cannot be translated into Korean words one-to-one. Part of `family` category is removed for the same reason. We also removed categories that give an advantage on English pre-trained embeddings; for example, the `city-in-state` category is removed because it contains relationships between US cities and US states. The remaining 6,168 questions are manually translated into Korean.

Table 3 shows the evaluation results of the English and Korean GloVe vectors on our subset of the word analogy corpus. As can be observed, the accuracy of Korean GloVe is much lower compared to the English GloVe vectors. Since the OutFlip relies on the cosine similarities between pre-trained embedding vectors to generate the OOD samples, the quality of embedding vectors is critical to the OOD generation performance.

Next, we applied the proposed OutFlip to other OOD detection baselines to see if the OutFlip could further improve their performance. Table 4 shows the evaluation results. In most cases applying the OutFlip to other OOD detection approaches resulted in performance improvement. The performance improvement was significant when the dataset is balanced, and the number of known intents is small.

We conducted a set of experiments to find out the best OutFlip iteration number and T_{sim} value. Figure 2 and Figure 3 shows the OutFlip performances on the benchmark datasets with changing

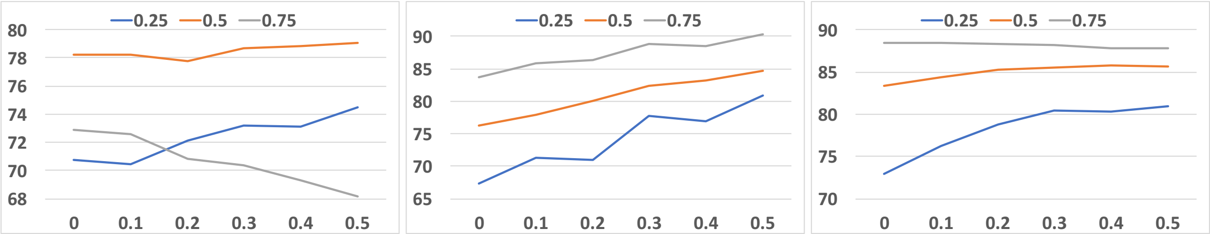

(a) ATIS dataset (b) SNIPS dataset (c) Kakao dataset

Figure 2: Evaluation results of **OutFlip**$_{cnn}$ with changing T_{sim}. We fixed the number of OutFlip iteration to 2.

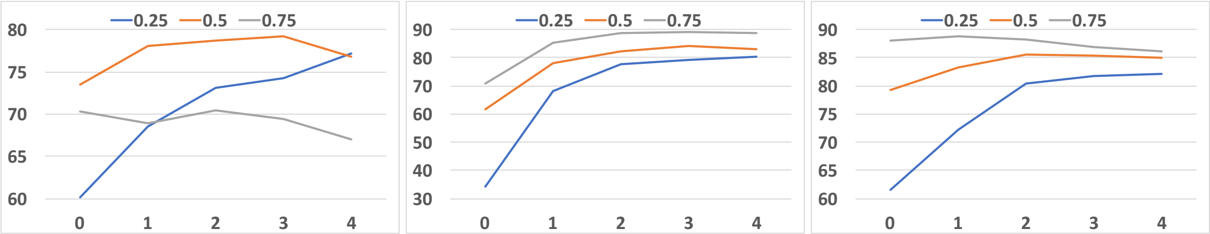

(a) ATIS dataset (b) SNIPS dataset (c) Kakao dataset

Figure 3: Evaluation results of **OutFlip**$_{cnn}$ with changing OutFlip iteration number. We fixed T_{sim} to 0.3.

T_{sim} values and OutFlip iteration numbers, respectively. Increasing T_{sim} value would cause the OutFlip to generate more challenging examples, but the chance of developing wrong OOD samples would also increase.

Figure 2 suggests that the balanced dataset like SNIPS could easily recover the errors introduced from large T_{sim} values. In contrast, the ATIS dataset's macro F1 score decreases with the increased T_{sim} values when many intents are known. Since 3 out of 18 ATIS intents have less than ten sentences, one or two erroneous OOD samples could lead to a performance drop. The macro F1 score of the balanced Kakao dataset does not increase with the T_{sim} values larger than 0.3. Since the quality of Korean GloVe is relatively low, large T_{sim} values introduce more errors compared to the English datasets.

As can be observed from Figure 3, in most cases, the macro F1 score converges with two to three OutFlip iterations. Additional OutFlip iterations give small or no performance improvements for balanced datasets and decrease macro F1 score for unbalanced dataset ATIS by introducing more errors.

One advantage of the OutFlip is that the generated OOD samples could be used to train and improve the OOD detection performance of the models other than the reference model without applying additional OutFlip iterations. We trained the BERT-base and BERT-large models (Devlin et al., 2018) with the ATIS and SNIPS benchmarks. As the same as previous experiments, the unknown intents are removed during training and integrated back during testing. Besides, we added OOD samples generated using reference models **OutFlip**$_{cnn}$ and **OutFlip**$_{lstm}$ with three OutFlip iterations and T_{sim} value 0.3, while training the BERT models.

Table 5 shows the evaluation results of the BERT models trained with the OutFlip-generated OOD samples. As can be observed from the table, the OutFlip-generated OOD samples significantly improved the OOD detection performances of BERT models, regardless of the reference models used to generate the OOD samples.

4.5 Error Analysis

We randomly selected 200 samples from the OOD examples generated by **OutFlip**$_{cnn}$ with three iterations and $T_{sim} = 0.3$ for the ATIS dataset, when 75% of the intents are known. The number of newly generated OOD samples for each iteration is shown in Table 6. We manually analyzed the selected examples for errors.

Among the 200 examples, 186 of them were correctly generated OOD sentences. Out of 14 error cases, 12 were due to the wrongfully extracted Core Class Tokens. Some ATIS intents have too few examples to extract Core Class Tokens; for example, intent `atis_restriction` has only six samples. Also, an entity which shows up too frequently could also lead to wrong Core Class Token

Model	Reference model for OutFlip	ATIS			SNIPS		
		25%	50%	75%	25%	50%	75%
BERT-base	*None*	62.14	74.73	72.61	34.66	60.76	70.49
	OutFlip$_{cnn}$	70.60	76.08	72.45	**74.51**	**81.09**	**86.92**
	OutFlip$_{lstm}$	**71.58**	**78.22**	**72.68**	73.02	81.04	86.50
BERT-large	*None*	62.17	74.01	73.32	34.56	61.07	70.51
	OutFlip$_{cnn}$	**76.02**	80.68	**79.54**	70.95	**80.53**	**87.06**
	OutFlip$_{lstm}$	74.55	**81.46**	75.38	**73.15**	80.31	86.60

Table 5: Results of applying OutFlip-generated samples to train models other than the reference model. Reference model *None* means no OutFlip-generated OOD samples are added during training.

Iteration	1	2	3	4
ATIS	2003	1648.5	755.5	336.6
SNIPS	4923.5	4644.2	3221.5	1956.2

Table 6: The number of generated OOD samples for each OutFlip iteration with reference model **OutFlip**$_{cnn}$, $T_{sim} = 0.3$. 75% of the intents are known. Numbers are the average of 10 known intent selections.

extraction result. For intent `atis_flight`, 797 out of 4,334 samples contain the entity "Denver".

For one case, the OutFlip-generated sentence accidentally belongs to the other in-domain intent. However, due to the reference model's error, the OutFlip fails to remove the generated sentence. A training instance "Can you list the cheapest round trip **fare** from Orlando to Kansas City" (truth label `atis_airfare`) is converted to a sentence "Can you list the cheapest round trip **airplane** from Orlando to Kansas City" (truth label `atis_flight`), but the reference model classifies the converted sentence to `atis_airfare`. Since the classification result remains the same, the OutFlip considers the generated sentence as "hard-enough" OOD sample.

The ATIS dataset allows an instance to have multiple labels; two or more labels are assigned to 23 ATIS training instances. The OutFlip failed to properly handle those instances. The remaining one error case is generated from a training instance with two assigned labels.

5 Conclusion

In this paper, we proposed OutFlip, a method to generate OOD samples using only in-domain training dataset. Our evaluation results showed that the proposed OutFlip could significantly improve the OOD detection performance of an intent classification model by iteratively generating difficult OOD samples. Since OutFlip does not require any modifications to model architecture, it could be used with other OOD detection approaches to improve OOD detection performance further. We also showed that the generated OOD samples could be used to train and improve the OOD detection performance of models other than the reference model, without applying additional OutFlip iterations.

Currently, we only focused on generating difficult OOD samples, which can fool the reference model. However, generating meaningful OOD samples could also be beneficial, since then the dialogue engine developer could check the generated OOD samples to find new intents. As our future work, we will focus on generating meaningful, fluent OOD samples.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data.*, pages 93–104.

DongHyun Choi, IlNam Park, Myeong Cheol Shin, EunGgyun Kim, and Dong Ryeol Shin. 2020. Integrated eojeol embedding for erroneous sentence classification in korean chatbots. *Computing Research Repository*, arXiv:2004.05744.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Computing Research Repository*, arXiv:1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 506–514.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, USA, June 24-27*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Joo-Kyung Kim and Young-Bum Kim. 2018. Joint learning of domain classification and out-of-domain detection with dynamic class weighting for satisficing false acceptance rates. In *Proceedings of Interspeech 2018*, pages 556–560.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),*, pages 1292–1302.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, volume 5.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6193–6202.

Ting-En Lin and Hua Xu. 2019. Deep unknown intent detection with margin loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5491–5496.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. volume arXiv:1301.3781.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. 2018. Do deep generative models know what they don't know? In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. 2018. Out-of-domain detection based on generative adversarial network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 714–718.

Lei Shu, Hu Xu, and Bing Liu. 2017. Doc: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916.

Hong Xu, Keqing He, Yuanmeng Yan, Sihong Liu, Zijun Liu, and Weiran Xu. 2020. A deep generative distance-based classifier for out-of-domain detection with mahalanobis space. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1452–1460.

Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert Y.S. Lam. 2020. Unknown intent detection using gaussian mixture model with an application to zero-shot intent classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1050–1060.

Yinhe Zheng, Guanyi Chen, and Minlie Huang. 2020. Out-of-domain detection for natural language understanding in dialog systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pages 1198–1209.

GeoQA: A Geometric Question Answering Benchmark Towards Multimodal Numerical Reasoning

Jiaqi Chen[1*], **Jianheng Tang**[2*], **Jinghui Qin**[1], **Xiaodan Liang**[2†],
Lingbo Liu[1,3], **Eric P. Xing**[4], **Liang Lin**[1,3]
[1]Sun Yat-sen University, [2]Shenzhen Campus of Sun Yat-sen University, [3]Dark Matter AI Inc.,
[4]Mohamed bin Zayed University of Artificial Intelligence
`{jadgechen,sqrt3tjh,xdliang328,liulingbo918}@gmail.com,`
`qinjingh@mail2.sysu.edu.cn, eric.xing@petuum.com, linliang@ieee.org`

Abstract

Automatic math problem solving has recently attracted increasing attention as a long-standing AI benchmark. In this paper, we focus on solving geometric problems, which requires a comprehensive understanding of textual descriptions, visual diagrams, and theorem knowledge. However, the existing methods were highly dependent on handcraft rules and were merely evaluated on small-scale datasets. Therefore, we propose a **Geo**metric **Q**uestion **A**nswering dataset **GeoQA**, containing 5,010 geometric problems with corresponding annotated programs, which illustrate the solving process of the given problems. Compared with another publicly available dataset GeoS, GeoQA is 25 times larger, in which the program annotations can provide a practical testbed for future research on explicit and explainable numerical reasoning. Moreover, we introduce a Neural Geometric Solver (NGS) to address geometric problems by comprehensively parsing multimodal information and generating interpretable programs. We further add multiple self-supervised auxiliary tasks on NGS to enhance cross-modal semantic representation. Extensive experiments on GeoQA validate the effectiveness of our proposed NGS and auxiliary tasks. However, the results are still significantly lower than human performance, which leaves large room for future research. Our benchmark and code are released at https://github.com/chen-judge/GeoQA.

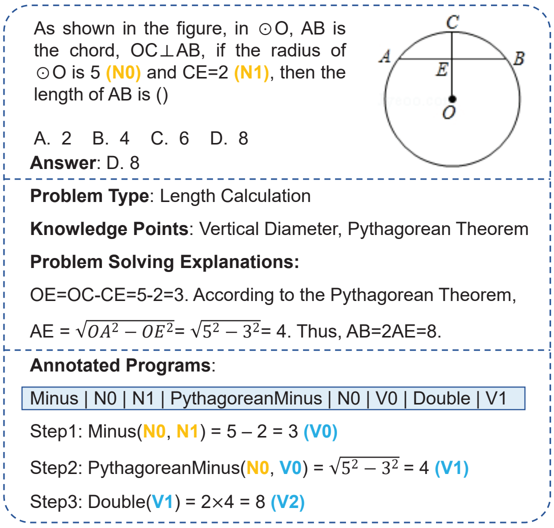

As shown in the figure, in ⊙O, AB is the chord, OC⊥AB, if the radius of ⊙O is 5 (N0) and CE=2 (N1), then the length of AB is ()

A. 2 B. 4 C. 6 D. 8

Answer: D. 8

Problem Type: Length Calculation

Knowledge Points: Vertical Diameter, Pythagorean Theorem

Problem Solving Explanations:

OE=OC-CE=5-2=3. According to the Pythagorean Theorem,

AE = $\sqrt{OA^2 - OE^2}$= $\sqrt{5^2 - 3^2}$= 4. Thus, AB=2AE=8.

Annotated Programs:

Minus | N0 | N1 | PythagoreanMinus | N0 | V0 | Double | V1

Step1: Minus(N0, N1) = 5 – 2 = 3 (V0)

Step2: PythagoreanMinus(N0, V0) = $\sqrt{5^2 - 3^2}$ = 4 (V1)

Step3: Double(V1) = 2×4 = 8 (V2)

Figure 1: Illustration of a typical geometry problem with the annotated programs in our GeoQA dataset.

1 Introduction

In recent years, developing machine learning systems to solve math word problems (MWPs) automatically has attracted increasing attention due to its high academic value and the great application potential in smart education (Bajaj and Sharma, 2018; Lin et al., 2018). Most of the existing methods focus on solving arithmetic and algebraic problems, including traditional machine learning approaches (Kushman et al., 2014; Zhou et al., 2015; Huang et al., 2016) and network-based models (Wang et al., 2017, 2018; Xie and Sun, 2019), while solving geometric problems has been rarely investigated (Seo et al., 2014, 2015a; Sachan et al., 2017). As a classic math problem, geometry dominates a large portion of secondary education. Due to its challenges and data characteristics, geometry problem can also serve as a multimodal numerical reasoning benchmark requiring joint reasoning over diagram and text.

As shown in Figure 1, a typical geometric question mainly consists of textual descriptions and geometric diagrams. Compared with math word problems, which only involve problem texts, geometric questions have posed the following new challenges. **First**, the additional problem diagrams provide essential information absent from the problem text,

---

[*] Equal contribution.

[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 513–523
August 1–6, 2021. ©2021 Association for Computational Linguistics

such as the relative location of lines and points; thus, the solver should have the capability to parse the diagram. **Second**, to solve a geometry problem, we need to understand and align the semantics of text and diagram simultaneously. However, the problem text often includes some ambiguous references and implicit relations to diagram elements, which increases the difficulty of joint reasoning over text and diagram. **Third**, many geometric problems require extra theorem knowledge in the problem solving process. For example, in Figure 1, the Pythagorean Theorem is used to calculate the length of line AE.

Though some previous methods (Seo et al., 2014, 2015a; Sachan et al., 2017, 2020; Sachan, 2020) attempt to resolve the mentioned issues, the performance of their geometric problem solving systems is far away from satisfactory. They highly depended on limited handcraft rules and were only validated on small-scale datasets, making it hard to generalize to more complex and real-world cases. Besides, the solving process is sophisticated, which means it is difficult for a human to understand and examine its reliability.

To refresh the research on geometric problem solving and promote further study on multimodal numerical reasoning, we propose a large-scale real-world geometric question answering dataset called GeoQA, which contains 5,010 multiple-choice geometric problems collected from real math exams in Chinese middle school. Inspired by Amini et al. (2019), we additionally introduce a new domain-specific language to model precise operation programs corresponding to the geometry problem. These executable programs represent the numerical reasoning steps of geometry problems. Compared with the existing dataset GeoS and GeoS++ (Seo et al., 2015a; Sachan et al., 2017), our GeoQA is larger, more diverse, provides additional program annotation, thus serves as a promising benchmark to improve both generalization and interpretability of the multimodal numerical reasoning approaches.

Moreover, we propose the first deep learning-based approach for geometry problem solving, named as Neural Geometric Solver (NGS). It applies a co-attention mechanism to fuse the representation of text and diagram, and predicts the explainable programs based on the cross-modal representation. These sequential programs can be executed to obtain a final answer. Benefiting from the structured and explainable program prediction, our NGS has both merits in superior performance by learning-based models compared to previous rule-based methods, as well as generating explainable numerical reasoning steps via the program sequence in favor of the model diagnosis. We further design three highly-relevant pretext tasks to enhance text-diagram semantic representation, including diagram jigsaw location prediction, diagram geometric element prediction, and knowledge point prediction. Extensive experiments are conducted on GeoQA benchmark, and the quantitative comparisons show the superiority of the proposed NGS and auxiliary tasks over other multimodal baselines.

In summary, our contributions are three-fold:

- We construct a large-scale dataset for geometric problem solving, which contains 5,010 Chinese geometric multiple-choice questions with rich domain-specific program annotations.
- A novel Neural Geometric Solver is proposed to solve geometric problems by generating symbolic programs based on the joint understanding of textual descriptions and diagrams.
- Multiple specialized auxiliary tasks are employed to effectively improve the semantic representation of text and diagrams. Experiments show the superiority of our NGS equipped with these auxiliary tasks.

2 Related Work

Geometry Problems Solving Developing automated systems to solve geometry problems has a long history in AI (Gelernter et al., 1960; Wen-Tsun, 1986; Chou et al., 1996; Ye et al., 2008). For example, Seo et al. (2014, 2015a) built the first automated system, GeoS, to solve SAT style geometry problems. GeoS used NLP and computer vision techniques (e.g., OCR) to parse a geometry problem's text and diagram jointly as logic forms. However, this system highly depended on the manually designed logic forms and was only examined in a small dataset with 185 problems. Besides, the limited logic forms are hard to cover various geometry problems, leading to low generalization. To improve GeoS, Sachan et al. (2017); Sachan and Xing (2017) replaced these handcraft constraints with geometry axiomatic knowledge in the form of horn-clause rules, but their dataset and code are not released. To boost the generalization and interpretability of existing works, we propose a large-scale GeoQA benchmark, which is 25 times larger than the only public dataset Seo et al. (2015a), and provides program annotation.

Multimodal Reasoning Visual question answering is a representative multimodal task that requires the model to have reasoning ability (Goyal et al., 2017; Yu et al., 2019). Johnson et al. (2017) built a new diagnostic VQA dataset (called CLEVR) with annotated functional programs. Based on this benchmark, some methods proposed an implicit reasoning framework to jointly encode multimodal information (Perez et al., 2017; Santoro et al., 2017). Moreover, several works (Yi et al., 2018; Mao et al., 2019) utilize domain-specific languages to perform explicit symbolic reasoning. However, these program languages only consider elementary operations, such as counting objects. They are not directly applicable to geometric problems, which require multiple steps of numerical calculation and involve theorem knowledge.

Self-supervised Auxiliary Task Self-supervised pretraining has gradually emerged (Doersch and Zisserman, 2017; Newell and Deng, 2020) as a effective technique to deal with label scarcity and improve model performance. To enhance visual features, most of these methods construct pseudo labels automatically and train on auxiliary tasks, including image jigsaw (Noroozi and Favaro, 2016; Ahsan et al., 2019), inpainting (Pathak et al., 2016), super resolution (Ledig et al., 2017), etc. Inspired by these works, we design two self-supervised auxiliary tasks and a supervised auxiliary task to enhance the reasoning ability of our NGS.

3 GeoQA Dataset

Due to the limited data scale and problem types, the existing geometric problem reasoning dataset (Seo et al., 2015a) can neither comprehensively reflect model's reasoning ability, nor support the training of neural models. To propose a better benchmark for the evaluation of multimodal numerical reasoning and inspire applications in smart education, we collect a new dataset GeoQA. It contains 5,010 diverse real-world geometric problems in Chinese middle school exams, and each problem is additionally annotated by specific programs that describe the problem solving process. Besides, we also provide human performance on GeoQA, as shown in Table 3.

	Total	Train	Val	Test
Number	5010	3509	746	755
Angle	2745	1939	388	418
Length	1873	1303	287	283
Other	392	267	71	54
#Avg DS	108×140	108×140	107×141	107×140
#Avg QL	52.5	52.4	52.4	57.7
#Avg KP	2.10	2.10	2.07	2.14
#Avg ET	1.11	1.13	1.08	1.09
#Avg OP	1.98	1.99	1.92	1.98
#Avg PL	5.35	5.39	5.17	5.36

Table 1: Statistics of our GeoQA dataset. It contains three types of problems, including angle, length, and other problems. Besides, DS, QL, KP, and ET represent diagram size, question length, knowledge points, and element types, respectively. OP and PL represent operation number and program length.

3.1 Data Description

Generally, a geometry multiple-choice problem can be represented as a tuple (t, d, c, i) where t is the problem text in natural language, d is the problem diagram, $\mathbf{c} = (c_1, c_2, c_3, c_4)$ represents the 4 numerical options for the problem, i is the answer index. Given the text t and diagram d, an algorithm is required to predict the correct answer $c_i \in \mathbf{c}$. To collect as much useful information as possible, we also provide the natural language-based problem solving explanations e, problem type t, the related knowledge points k, and our annotated programs p for each problem. Therefore, a geometry problem can be represented as (t, d, c, i, e, t, k, p). Fig. 1 shows an example of geometric problems.

Moreover, there are three problem types in our GeoQA, i.e., angle calculation, length calculation, and others which contain various types of problems such as area calculation. We adopt the corpus diversity metric proposed by Miao et al. (2020) to evaluate the diversity of GeoQA. The result is 0.47, which is relatively high compared with other math problem datasets, indicating that our dataset is diverse. The source data already contains manually tagged knowledge points in each problem, we design rule-based regular expressions to normalize the original knowledge points to 50 categories. We split our GeoQA into three subsets – train set, valid set, and test set, in a ratio of 7.0: 1.5: 1.5. The data statistics of our GeoQA are shown in Table 1.

3.2 Program Representation

The neural network has proved to be a powerful tool to address complex multimodal reasoning tasks. However, it still faces challenges when

Types	Programs
Basic	Equal, Double, Half
Arithmetic	Add, Minus, Multiply, Divide
Trigonometric	Sin, Cos, Tan, Arc-Sin, Arc-Cos
Theorem & Formula	Pythagorean Add/Minus, Proportion, Circle Area, Circle Perimeter, Cone Area
Constant	30, 60, 90, 180, 360, π, 0.618

Table 2: An overview of 18 operations of four different types and 7 constants in the defined program set.

conducting numerical calculation and providing explicit problem solving process, which are actually two crucial points in the task of geometric problem solving. To make better use of neural networks in the geometric problems solving process, inspired by Amini et al. (2019), we introduce a new domain-specific language to model the geometric problem solving process based on the GeoQA dataset. This program language can be directly executed to calculate the numerical answer based on the predefined operations and their arguments.

The program types contain operations OP, constants $Const$, problem variables N, and process variables V. As shown in Table 2, operations are divided into multiple categories, including Basic, Arithmetic, Trigonometric, Theorem, and Formula operations. Each operator involves $n(=1,2,3)$ elements selected from constants and variables. Constants are predefined numbers that are frequently used in geometric problems, such as π and the degree of a Right Angle (90). The problem variables refer to all the number that appears in the current problem, and process variables are obtained during the execution process.

In addition to the common math operations, our programs also contain some operations representing the knowledge of theorems and formulas that is helpful to address geometric problems, such as the Pythagorean theorem and the area calculation formula of a circle. It is worth noting that many simple geometric formulas do not require additional definitions. For example, given a square with side length a, its area can be directly computed by Multiply(a, a).

The interpretability of our program is reflected on the sequential process of the operations, the selected constants and variables, and the application of theorems and formulas. As shown in Figure 1, we can have a general understanding of the entire problem solving process after reading the program.

3.3 Collection and Annotation

We collect GeoQA from two online education websites[1]. These problems are oriented grades 6-12, containing various types of problems with corresponding knowledge points and solving explanations. we organize more than ten well-trained college students with a relevant major to specifically annotate our programs by referring to the solving explanations. To ensure label quality and consistency, they are required to read the guideline of annotation standards and examples in advance. Each annotated program is double-checked by one of the authors, and the annotator with low accuracy would be disqualified. The annotated operations required to solve the problem are limited to a maximum of 4 steps, thus a small number of complex and hard problems are filtered.

4 Neural Geometric Solver

We propose Neural Geometric Solver (NGS) to address geometric problems by jointly understanding text, diagram, and then generating explainable programs. Moreover, we utilize some novel auxiliary tasks to enhance the understanding ability of our NGS. The overall architecture of our NGS is shown in Fig. 2.

4.1 The Architecture of NGS

4.1.1 Problem Encoder

Text Encoder Given a problem text $P = \{x_i\}_{i=1}^{n}$, each token x_i is first embedded into a word vector $\mathbf{x}_i$. A single-layer unidirectional LSTM (Hochreiter and Schmidhuber, 1997) is then applied to encode each word embedding $\mathbf{x}_i$ into a hidden state $\mathbf{h}_i$. The sequence of the hidden state in LSTM are used to represent problem text P as $H_P = [\mathbf{h}_0; ...; \mathbf{h}_n]$.

Diagram Encoder For representing a problem diagram, we apply the first three stages of a ResNet-101 (He et al., 2016) to extract it as the diagram feature, which can be formalized as a feature matrix $H_D \in \mathbb{R}^{m \times d}$. Moreover, we also apply multiple auxiliary tasks for pretraining the diagram encoder. Note that the parameters of the diagram encoder are fixed when training the overall NGS.

4.1.2 Joint Reasoning Module

Given the text feature H_P and the diagram feature H_D, it is crucial for solving geometric prob-

---

[1] http://www.zxxk.com/ and http://www.jyeoo.com/

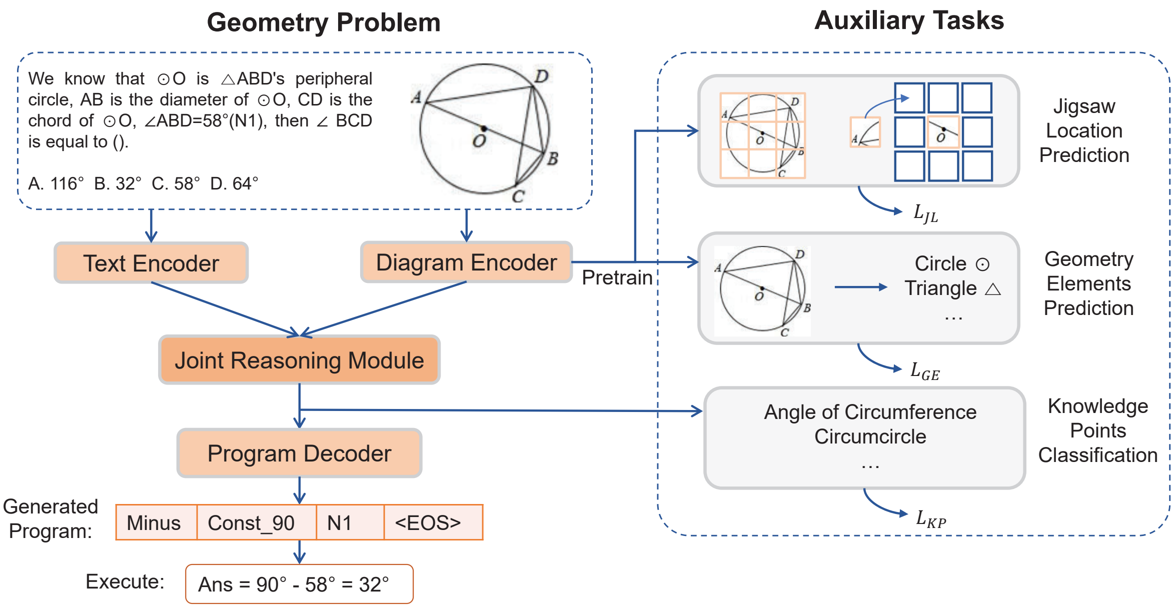

Figure 2: The overall architecture of our Neural Geometric Solver (left) in conjunction with auxiliary tasks (right). The problem text and diagram is encoded separately, then fed into a joint reasoning module together to obtain cross-modal fusion of text and diagram features. A decoder utilizes fused multimodal features to generate the interpretable programs. In addition, we propose three auxiliary tasks to enhance feature representation and facilitate multimodal reasoning.

lems to jointly fuse and align the cross-modal information. To this end, inspired by Yu et al. (2019), we adopt a co-attention module to conduct cross-modal joint reasoning with attention mechanism. The co-attention module consists of 12 self-attention (SA) units and 6 guided-attention (GA) units, which fully fuse and align the text representation H_P and the diagram representation H_D. H_P is first encoded by 6 self-attention units (i.e., original Transformer), and the final hidden state processed by the 6-th self-attention unit is used as guiding information. Then, the guiding information is fed into another stacked 6 self-attention units and 6 guided-attention units to achieve cross-modal semantic fusion and alignment. Finally, the co-attention module outputs a cross-modal representation $F_D = [f_1^D; ...; f_n^D]$, which contains rich information over the problem text and diagram.

In this work, we find that text information is more fundamental than diagram information. Therefore, we further enhance the cross-modal representation with the help of textual information. Specifically, we concatenate H_P and F_D to acquire an enhanced reasoning module output F_R for decoding programs.

Besides, an attentional reduction network with a two-layer MLP is applied to aggregate feature F_D into $\tilde{F}_D$. Similarly, we concatenate $\tilde{F}_D$ and the last encoder state of the text encoder $\boldsymbol{h}_n$, obtaining $\tilde{F}_R$ as the final gathered multimodal feature vector.

4.1.3 Program Decoder

The program decoder module generates the programs sequentially under the guidance of multimodal information. Concretely, we use a LSTM decoder (Hochreiter and Schmidhuber, 1997) with attention (Bahdanau et al., 2014) over the reasoning module output F_R . Let $\{y_t\}(1 \leq t \leq T)$ be the target program to be generated and $\boldsymbol{s}_t$ be the hidden state of LSTM at time step t. $\tilde{F}_R$ is fed into a linear layer to obtain the initial state $\boldsymbol{s}_0$. $\boldsymbol{s}_t$ is concatenated with the attention result and fed to a linear layer with the softmax function to predict the distribution of the next program token P_t.

During training, the generation loss L_g is the negative log-likelihood (NLL) of the target program:

$$\mathcal{L}_g(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} \log P_t(y_t|\boldsymbol{x}, y_1, ..., y_{t-1}; \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are the parameters of the entire NGS architecture except for the diagram encoder, $\boldsymbol{x}$ is the input of both problem text and the extracted diagram feature. When testing, the decoder only observes the input text and diagram feature along with the program parts that have been generated.

4.1.4 Program Executor

After a beam of top N program sequences $\{g_1, ..., g_n\}$ are generated from the program decoder, the executor computes them step by step. When executing the program, the token sequence is first divided into several parts based on the position of operators in the program. Once a complete operation program has been decoded, each operator in the program is executed sequentially to obtain a numerical result. The execution process fails if g_i has a grammar error (e.g., the number of augments does not match the current operator), or the executed value does not match any options in the current problem. NGS adopts the first successfully executed program as the predicted solution and chooses the corresponding operation. If all N program sequences fail, the executor will report "no result" directly instead of guessing an option. Fig. 1 shows the detailed step-by-step execution of a final predicted program sequence.

4.2 Auxiliary Tasks

4.2.1 Self-supervised Diagram Auxiliary Task

Although our NGS can jointly fuse text feature and diagram feature with a co-attention mechanism, a powerful diagram encoder is needed to improve the problem understanding and answer accuracy. To obtain a high-quality diagram feature, we investigate two self-supervised auxiliary tasks, named as **Jigsaw Location Prediction** and **Geometry Elements Prediction**, to pretrain diagram encoder.

Jigsaw Location Prediction In the Jigsaw location prediction task that enforces pixel-level perception, we first split a diagram as $m \times m$ blocks and select the center block as the target. Then, we shuffle other blocks randomly and train the diagram encoder to predict the correct relative location between these shuffled blocks and the target using a cross-entropy loss.

Geometry Elements Prediction For object-level understanding, we design a geometry elements prediction task that aims at training the diagram encoder to predict the geometry elements appearing in the diagram. A diagram usually contains multiple geometry elements which are also mentioned in the problem text and the solving explanation. We extract these geometry elements from text as the label and deploy an N-way classifier with binary cross-entropy (BCE) as the loss function to train the diagram encoder, where N is the number of the possible geometry elements on the training set.

4.2.2 Knowledge Points Prediction

In addition to self-supervised diagram training, we also propose another auxiliary learning task called knowledge points prediction to enhance a problem's overall representation by providing an extra training signal. We summarize about 50 knowledge points for our GeoQA, and label each problem with one or more knowledge points. We predict the knowledge points for each problem based on the gathered feature vector $\tilde{F}_R$ outputted from the joint reasoning module. Different from the diagram training, the knowledge points prediction task is trained with NGS simultaneously. We also deploy a K-way classifier with binary cross-entropy (BCE) as the loss function to train the knowledge points prediction multi-label task, where K is the total number of the possible knowledge points on the training set.

5 Experiments

5.1 Experimental Setup and Training Details

We conduct experiments on GeoQA dataset, and adopt answer accuracy as the evaluation metric. Although there is another available geometric problem dataset (Seo et al., 2015a), the limited data scale (with only 67 training samples) makes it impossible to support neural network training. On the other hand, previous geometry problem solving systems require additional inputs (e.g., OCR and dependency parsing results) of the problem (Ye et al., 2008; Seo et al., 2015b) or not release their codes (Sachan and Xing, 2017; Sachan, 2020). Therefore, they are not comparable on our GeoQA dataset.

Implementation Details: In this work, we implement the proposed method with Pytorch (Paszke et al., 2017). The learning rate is $1e^{-3}$ and the batch size is set to 32. All models are trained around 100 epochs and optimized by Adam optimizer (Kingma and Ba, 2014). The beam size is typically set to 10. When pretraining the diagram encoder, we first fill the diagram with a white background to make it equal in length and width, and resize it to 224×224. Then, we utilize the diagram feature extracted by the encoder to predict jigsaw location and geometry elements simultaneously and optimize the diagram encoder to obtain an informative diagram feature with a learning rate

Method		Total (%)	Angle (%)	Length (%)	Other (%)
Human	Text-Only	63.0	58.1	71.7	55.6
	Text-Diagram	92.3	94.3	90.5	87.0
W/O Program	FiLM (Perez et al., 2017)	31.7	34.0	29.7	24.1
	RN (Santoro et al., 2017)	38.0	42.8	32.5	29.6
	MCAN (Yu et al., 2019)	39.7	45.0	34.6	25.9
Text-Only	Seq2Prog (Amini et al., 2019)	52.3	62.4	42.1	27.8
	BERT2Prog (Devlin et al., 2018)	54.7	65.8	42.1	35.2
Text-Diagram	BERT2Prog + Diagram	50.3	63.4	33.2	38.9
	Seq2Prog + Diagram	52.6	63.6	39.2	37.0
	NGS (Ours)	56.7	67.5	44.5	37.0
	NGS-Auxiliary (Ours)	**60.7**	**72.0**	**47.0**	**44.4**

Table 3: The answer accuracy comparison on different test subsets of GeoQA dataset. "Human", "W/O Program", "Text-Only", and "Text-Diagram" refer to the performance of human, not using the program, using text modal only, and conducting multimodal numerical reasoning on both text-diagram modals, respectively.

Method	BS	Acc(%)	NR(%)
Seq2Prog + Diagram	1	31.7	58.3
	10	52.6	20.4
	100	58.3	5.86
NGS-Auxiliary	1	45.6	42.6
	10	60.7	14.6
	100	64.5	2.95

Table 4: Performance comparison under different beam size settings. BS, Acc, and NR represent beam size, accuracy, and no result, respectively.

of $1e^{-5}$. Finally, the loss weight of the knowledge points classification task is set to 1, to promote the overall understanding of problems.

5.2 Experimental Results

We introduce three types of models here and test them on our GeoQA. The performance comparison with various methods on the different subset types of GeoQA is reported in Table 3.

Human Performance. We invite 10 students with a high score (top 1%) in the national university entry exam to answer these geometric problems. For each question, they first try to solve the problem with the text only and draw a diagram by themselves. Then, the actual diagram is given to answer the complete question. The total time for each question is limited to two minutes. When using text and diagram simultaneously, the human performance is improved from 63.2% to 92.3%, which indicates that humans are not good at solving problems using only text, but handling multimodal information successfully. The result shows that there is still a huge gap between the existing models and human experts, leaving large room for future research.

The effectiveness of program. "W/O Program" refers to not using programs and regarding GeoQA as a classification problem similar to VQA. Therefore, we conduct experiments on three models with multimodal reasoning capabilities, including FiLM, RN, and MCAN. However, their performance results show that these methods fail to reason about such complex geometric problems, achieving poor performance on GeoQA. These results prove the effectiveness and importance of our designed interpretable programs.

The necessity of multi-modality. "Text-Only" means that models only use text to generate program sequences since humans can also understand the intent of the question, draw diagram based on the text, and solve the problem. Motivated by Amini et al. (2019), we design a Sequence-to-Program (Seq2Prog) model using GRU encoder with an attention mechanism. Moreover, we replace the encoder with BERT and get a stronger BERT2Prog. By predicting our tailor-designed programs, these two methods are effective, while the performance is not satisfactory enough. These results show that multimodal reasoning ability is indispensable when solving geometric problems.

Multimodal numerical reasoning baselines. "Text-Diagram" refers to using text and diagram simultaneously. We concatenate the text embedding with the diagram feature extracted by ResNet (He et al., 2016) as the baseline methods, such as Seq2Prog + Diagram and BERT2Prog + Diagram. These feature fusion methods do not have the strong reasoning ability and fail to improve the program decoding. For instance, by adding diagram, the performance of BERT2Prog + Diagram declines from 54.7% to 50.3%, which may result

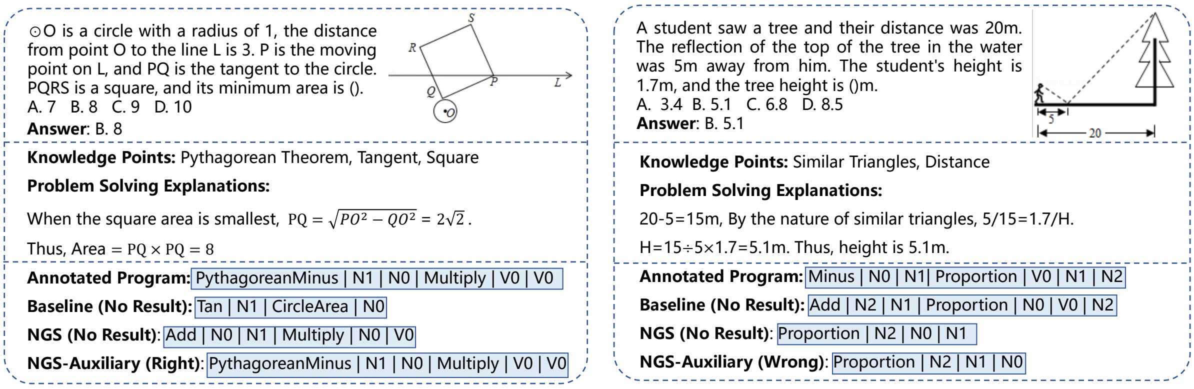

⊙O is a circle with a radius of 1, the distance from point O to the line L is 3. P is the moving point on L, and PQ is the tangent to the circle. PQRS is a square, and its minimum area is ().
A. 7 B. 8 C. 9 D. 10
Answer: B. 8

Knowledge Points: Pythagorean Theorem, Tangent, Square

Problem Solving Explanations:

When the square area is smallest, $PQ = \sqrt{PO^2 - QO^2} = 2\sqrt{2}$.

Thus, Area $= PQ \times PQ = 8$

Annotated Program: PythagoreanMinus | N1 | N0 | Multiply | V0 | V0
Baseline (No Result): Tan | N1 | CircleArea | N0
NGS (No Result): Add | N0 | N1 | Multiply | N0 | V0
NGS-Auxiliary (Right): PythagoreanMinus | N1 | N0 | Multiply | V0 | V0

A student saw a tree and their distance was 20m. The reflection of the top of the tree in the water was 5m away from him. The student's height is 1.7m, and the tree height is ()m.
A. 3.4 B. 5.1 C. 6.8 D. 8.5
Answer: B. 5.1

Knowledge Points: Similar Triangles, Distance

Problem Solving Explanations:

20-5=15m, By the nature of similar triangles, 5/15=1.7/H.

H=15÷5×1.7=5.1m. Thus, height is 5.1m.

Annotated Program: Minus | N0 | N1| Proportion | V0 | N1 | N2
Baseline (No Result): Add | N2 | N1 | Proportion | N0 | V0 | N2
NGS (No Result): Proportion | N2 | N0 | N1
NGS-Auxiliary (Wrong): Proportion | N2 | N1 | N0

Figure 3: Typical cases. No Result represents the answer executed by the programs is not in the options, and Wrong represents getting a wrong option. Baseline is a "Seq2Prog + Diagram" model. In the case on the left, NGS-Auxiliary successfully predicts the knowledge of the Pythagorean theorem through auxiliary tasks and get the right answer. For the case on the right, the problem is quite hard that current model cannot solve it.

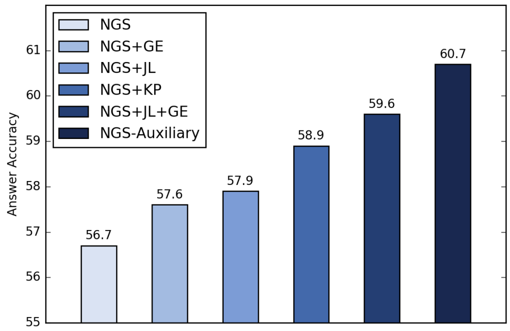

Figure 4: Ablation study on different auxiliary components. '+' represents we add the auxiliary component. NGS-Auxiliary means that adding all three auxiliary tasks together.

from the extra diagram that disturbs the text pretraining model. Note that multimodal pretraining models (Lu et al., 2019; Li et al., 2020) cannot be applied to geometric problems, since these models are based on Faster-RCNN to extract object-level features from natural images.

The effectiveness of our methods. Our proposed NGS shows a relatively-good performance compared to the various models mentioned above. When adding all three auxiliary tasks together to enhance NGS solver, our NGS-Auxiliary with multimodal reasoning ability becomes the existing best-performing method (60.7%) on GeoQA dataset. It also achieves the highest accuracy on all types of problems. For example, compared with Seq2Prog+Diagram, NGS-Auxiliary obtains an 8.4% performance improvement on the angle type problems. Compared with other "Text-Diagram" baselines, our model is effective when reasoning on multimodal information.

The effect of different beam size. In general, we set the beam size to 10 for testing. In this section, we explore the influence of different beam size. After the searched sequence program is executed, there will be three situations: right answer, wrong answer, and no result. As shown in Table. 4, as the beam size is larger, we get higher accuracy and a lower proportion of no result. When beam size equals 1, the NGS-Auxiliary outperforms baselines significantly. Our model can achieve the highest accuracy of 64.5% when beam size is 100.

5.3 Ablation Study

As shown in Fig. 4, we conduct experiments to evaluate the contribution of different auxiliary tasks. We consider six different combinations: 1) only the NGS; 2) NGS + Geometry Elements (NGS+GE); 3) NGS + Jigsaw Location (NGS+JL); 4) NGS + Knowledge Points (NGS+KP); 5) NGS + diagram-based pretraining (NGS+JL+GE); 6) NGS with all three auxiliary tasks (NGS-Auxiliary). We can see that all three auxiliary tasks can promote the performance of NGS. The accuracy gains of GE, JL, KP, JL + GE, and combining all three tasks are 0.9%, 1.2%, 2.2%, 2.9%, 4.0%, respectively. These results show that all our self-supervised and auxiliary tasks can enhance the comprehensive understanding and multimodal reasoning ability of NGS.

5.4 Case Analysis

As shown in Fig. 3, we select two typical cases to demonstrate the programs generated by different models and some representative errors.

In the left case, the knowledge of the

Pythagorean theorem, tangent, and square are required for solving the problem. Baseline method and our NGS fail to generate the correct operations and get no result. However, our NGS-Auxiliary successfully predicts the use of knowledge in the proposed auxiliary task. And more importantly, the correct ”PythagoreanMinus” program is generated, and the right answer is obtained.

The right one is a typical error case, in which model needs to understand a complex scene. Although NGS-Auxiliary has predicted the knowledge points of Proportion program correctly, all three models fail to predict the correct answer. A better multimodal method is required to handle this hard high-level reasoning task in the future.

6 Conclusion

In this work, we focus on the geometric problem and propose the first large-scale geometric question answering dataset “GeoQA”, containing 5,010 problems with program annotation. Besides, we propose a deep neural baseline, named as Neural Geometric Solver (NGS), to solve a geometric problem by jointly reasoning over multimodal data and generating interpretable programs. We further propose multiple novel auxiliary tasks to enhance the semantic representation of text and diagram. Extensive experimental results and analyses show that our GeoQA is challenging, and our NGS-Auxiliary outperforms other methods on GeoQA.

7 Ethical Impact

We collected GeoQA from two online education websites, which is only used for academic research, and the copyright belongs to the original websites. This work may inspire research in the field of multimodal numerical reasoning.

Acknowledgements This work was supported in part by National Key R&D Program of China under Grant No.2020AAA0109700, National Natural Science Foundation of China (NSFC) under Grant No.U19A2073 and No.61976233, Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, Shenzhen Fundamental Research Program (Project No.RCYX20200714114642083 and No.JCYJ20190807154211365), Zhijiang Lab’s Open Fund (No.2020AA3AB14), and CSIG Young Fellow Support Fund.

References

Unaiza Ahsan, Rishi Madhok, and Irfan Essa. 2019. Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 179–189. IEEE.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2357–2367.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Richa Bajaj and Vidushi Sharma. 2018. Smart education with artificial intelligence based determination of learning styles. *Procedia computer science*, 132:834–842.

Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. 1996. Automated generation of readable proofs with geometric invariants. *Journal of Automated Reasoning*, 17(3):325–347.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Carl Doersch and Andrew Zisserman. 2017. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060.

Herbert Gelernter, James R Hansen, and Donald W Loveland. 1960. Empirical explorations of the geometry theorem machine. In *Papers presented at the May 3-5, 1960, western joint IRE-AIEE-ACM computer conference*, pages 143–149.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896. Association for Computational Linguistics.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 271–281.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. 2020. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11336–11344.

Jinjiao Lin, Haitao Pu, Yibin Li, and Jian Lian. 2018. Intelligent recommendation system for course selection in smart education. *Procedia Computer Science*, 129:449–453.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *International Conference on Learning Representations*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.

Alejandro Newell and Jia Deng. 2020. How useful is self-supervised pretraining for visual tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7354.

Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2017. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*.

Mrinmaya Sachan. 2020. Knowledge graph embedding compression. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2681–2691.

Mrinmaya Sachan, Avinava Dubey, Eduard H Hovy, Tom M Mitchell, Dan Roth, and Eric P Xing. 2020. Discourse in multimedia: A case study in extracting geometry knowledge from textbooks. *Computational Linguistics*, 45(4):627–665.

Mrinmaya Sachan, Kumar Dubey, and Eric Xing. 2017. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 773–784.

Mrinmaya Sachan and Eric Xing. 2017. Learning to solve geometry problems from natural language demonstrations in textbooks. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 251–261.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015a. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on*

Empirical Methods in Natural Language Processing, pages 1466–1476.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015b. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476. Association for Computational Linguistics.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069. Association for Computational Linguistics.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854. Association for Computational Linguistics.

Wu Wen-Tsun. 1986. Basic principles of mechanical theorem proving in elementary geometries. *Journal of automated Reasoning*, 2(3):221–252.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. 2008. An introduction to java geometry expert. In *International Workshop on Automated Deduction in Geometry*, pages 189–195. Springer.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *arXiv preprint arXiv:1810.02338*.

Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. 2019. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6281–6290.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822. Association for Computational Linguistics.

SIRE: Separate Intra- and Inter-sentential Reasoning for Document-level Relation Extraction

Shuang Zeng[1,3], **Yuting Wu**[1,2] and **Baobao Chang**[1*]
[1]The MOE Key Laboratory of Computational Linguistics, Peking University, China
[2]Wangxuan Institute of Computer Technology, Peking University, China
[3]School of Software and Microelectronics, Peking University, China
`{zengs,wyting,chbb}@pku.edu.cn`

Abstract

Document-level relation extraction has attracted much attention in recent years. It is usually formulated as a classification problem that predicts relations for all entity pairs in the document. However, previous works indiscriminately represent intra- and inter-sentential relations in the same way, confounding the different patterns for predicting them. Besides, they create a document graph and use paths between entities on the graph as clues for logical reasoning. However, not all entity pairs can be connected with a path and have the correct logical reasoning paths in their graph. Thus many cases of logical reasoning cannot be covered. This paper proposes an effective architecture, SIRE, to represent intra- and inter-sentential relations in different ways. We design a new and straightforward form of logical reasoning module that can cover more logical reasoning chains. Experiments on the public datasets show SIRE outperforms the previous state-of-the-art methods. Further analysis shows that our predictions are reliable and explainable. Our code is available at `https://github.com/PKUnlp-icler/SIRE`.

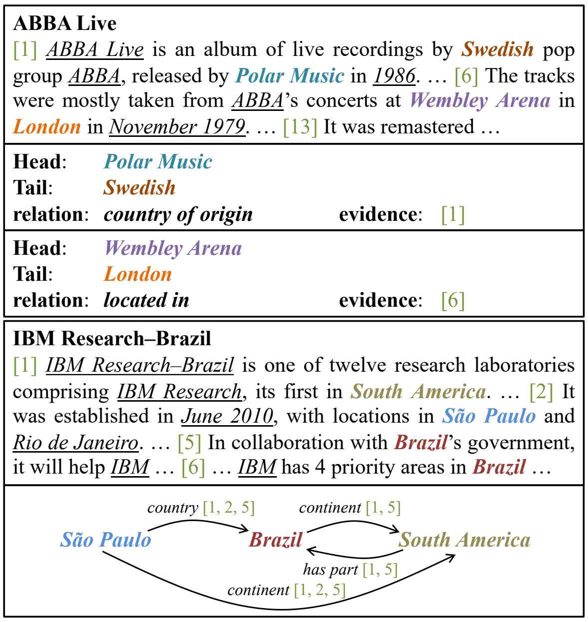

ABBA Live
[1] *ABBA Live* is an album of live recordings by ***Swedish*** pop group *ABBA*, released by ***Polar Music*** in *1986*. … [6] The tracks were mostly taken from *ABBA*'s concerts at ***Wembley Arena*** in ***London*** in *November 1979*. … [13] It was remastered …

Head: ***Polar Music***
Tail: ***Swedish***
relation: ***country of origin*** **evidence**: [1]

Head: ***Wembley Arena***
Tail: ***London***
relation: ***located in*** **evidence**: [6]

IBM Research–Brazil
[1] *IBM Research–Brazil* is one of twelve research laboratories comprising *IBM Research*, its first in ***South America***. … [2] It was established in *June 2010*, with locations in ***São Paulo*** and *Rio de Janeiro*. … [5] In collaboration with ***Brazil***'s government, it will help *IBM* … [6] … *IBM* has 4 priority areas in ***Brazil*** …

Figure 1: Two examples from DocRED (Yao et al., 2019b) for illustration of intra- and inter-sentential relations. Sentence numbers, entity mentions, and supporting evidence involved in these relation instances are colored. Other mentions are underlined for clarity.

1 Introduction

Relation Extraction (RE) is an important way of obtaining knowledge facts from natural language text. Many recent advancements (Sahu et al., 2019; Christopoulou et al., 2019; Yao et al., 2019b; Nan et al., 2020; Zeng et al., 2020; Wang et al., 2020) manage to tackle the document-level relation extraction (doc-level RE) that extracts semantic relations among entities across multiple sentences. Due to its strong correlation with real-world scenarios, doc-level RE has attracted much attention in the field of information extraction.

The doc-level RE task is usually formulated as a classification problem that predicts possible relations for all entity pairs, using the information from the entire document. It has two different kinds of relations: intra-sentential relation and inter-sentential relation. We show examples of these two kinds of relations in Figure 1. When two entities have mentions co-occurred in the same sentence, they may express intra-sentential relations. Otherwise, they may express inter-sentential relations.

Previous methods do not explicitly distinguish these two kinds of relations in the design of the model and use the same method to represent them. However, from the perspective of linguistics, intra-sentential relations and inter-sentential relations are expressed in different patterns. For two intra-sentential entities, their relations are usually expressed from local patterns within their co-occurred

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 524–534
August 1–6, 2021.

sentences. As shown in the first example in Figure 1, *(Polar Music, country of origin, Swedish)* and *(Wembley Arena, located in, London)* can be inferred based solely on the sentence they reside in, i.e., sentences 1 and 6 respectively. Unlike intra-sentential relations, inter-sentential relations tend to be expressed from the global interactions across multiple related sentences, also called supporting evidence. Moreover, cross-sentence relations usually require complex reasoning skills, e.g., logical reasoning. As shown in the second example in Figure 1, (*São Paulo*, continent, *South America*) can be inferred from the other two relation facts expressed in the document: (*São Paulo*, country, *Brazil*) and (*Brazil*, continent, *South America*). So the different patterns between intra- and inter-sentential relations show that it would be better for a model to treat intra- and inter-sentential relations differently. However, previous works usually use the information from the whole document to represent all relations, e.g., 13 sentences for predicting *(Polar Music, country of origin, Swedish)* in the first example in Figure 1. We argue that this will bring useless noises from unrelated sentences that misguide the learning of relational patterns.

Besides, previous methods (Christopoulou et al., 2019; Zeng et al., 2020) treat logical reasoning as a representation learning problem. They construct a document graph from the input document using entities as nodes. And the paths between two entities on their graphs, usually passing through other entities, could be regarded as clues for logical reasoning. However, since not all entity pairs can be connected with a path and have the correct logical reasoning paths available on the graph, many cases of logical reasoning cannot be covered. So their methods are somehow limited, and we should consider a new form of logical reasoning to better model and cover all possible reasoning chains.

In this paper, we propose a novel architecture called **S**eparate **I**ntra- and inter-sentential **RE**asoning (**SIRE**) for doc-level RE. Unlike previous works in this task, we introduce two different methods to represent intra- and inter-sentential relations respectively. For an intra-sentential relation, we utilize a sentence-level encoder to represent it in every co-occurred sentence. Then we get the final representation by aggregating the relational representations from all co-occurred sentences. This will encourage intra-sentential entity pairs to focus on the local patterns in their co-occurred sentences. For an inter-sentential relation, we utilize a document-level encoder and a mention-level graph proposed by Zeng et al. (2020) to capture the document information and interactions among entity mentions, document, and local context. Then, we apply an evidence selector to encourage inter-sentential entity pairs to selectively focus on the sentences that may signal their cross-sentence relations, i.e., finding supporting evidence. Finally, we develop a new form of logical reasoning module where one relation instance can be modeled by attentively fusing the representations of other relation instances in all possible logical chains. This form of logical reasoning could cover all possible cases of logical reasoning in the document.

Our contributions can be summarized as follows:

- We propose an effective architecture called SIRE that utilizes two different methods to represent intra-sentential and inter-sentential relations for doc-level RE.
- We come up with a new and straightforward form of logical reasoning module to cover all cases of logical reasoning chains.

We evaluate our SIRE on three public doc-level RE datasets. Experiments show SIRE outperforms the previous state-of-the-art models. Further analysis shows SIRE could produce more reliable and explainable predictions which further proves the significance of the separate encoding.

2 Separate Intra- and Inter-sentential Reasoning (SIRE) Model

SIRE mainly consists of three modules: intra- and inter-sentential relation representation module (Sec. 2.1), logical reasoning module (Sec. 2.2), classification module (Sec. 2.3), as is shown in Figure 2. Assume we have a document $\mathcal{D}$ containing l sentences $\{\mathcal{S}_i\}_{i=1}^{l}$.

2.1 Intra- and Inter-sentential Relation Representation Module

As is discussed in Sec. 1, for two intra-sentential entities, their relations are usually determined by the local patterns from their co-occurred sentences, while for two inter-sentential entities, their relations are usually expressed across multiple related sentences that can be regarded as the supporting evidence for their relations. So in this module, we utilize two different methods to represent intra-sentential and inter-sentential relations separately.

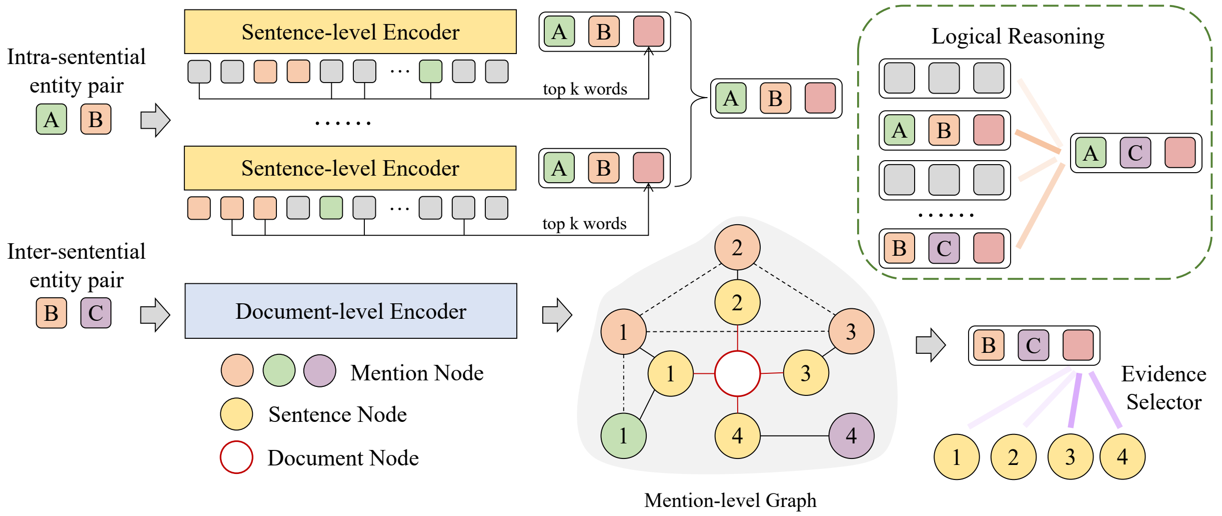

Figure 2: The architecture of SIRE. In the mention-level graph, the number in each circle is its sentence number. Mention nodes with the same color belong to the same entity. Different types of edges are in different styles of line. Our model uses different methods to represent intra- and inter-sentential relations and the self-attention mechanism to model the logical reasoning process. We use the logical reasoning chain:$e_A \rightarrow e_B \rightarrow e_C$ for illustration.

Our methods encourage intra-sentential entity pairs to focus on their co-occurred sentences as much as possible and encourage inter-sentential entity pairs to selectively focus on the sentences that may express their cross-sentence relations. We use three parts to represent the relation between two entities: head entity representation, tail entity representation and context representation.

2.1.1 Intra-sentential Relation Representation Module

Encoding. We use a sentence-level encoder to capture the context information for intra-sentential relations and produce contextualized word embedding for each word. Formally, we convert the i-th sentence $\mathcal{S}_i$ containing n_i words $\left\{w_j^{\mathcal{S}_i}\right\}_{j=1}^{n_i}$ into a sequence of vectors $\left\{\mathbf{g}_j^{\mathcal{S}_i}\right\}_{j=1}^{n_i}$.

For each word w in $\mathcal{S}_i$, we first concatenate its word embedding with entity type embedding and co-reference embedding[1]:

$$\mathbf{x} = [E_w(w); E_t(t); E_c(c)] \qquad (1)$$

where $E_w(\cdot)$, $E_t(\cdot)$ and $E_c(\cdot)$ denote the word embedding layer, entity type embedding layer and co-reference embedding layer, respectively. t and c are named entity type and entity id.[2]

Then the vectorized word representations are fed into the sentence-level encoder to obtain the sentence-level context-sensitive representation for each word:

$$[\mathbf{g}_1^{\mathcal{S}_i}, \dots, \mathbf{g}_{n_i}^{\mathcal{S}_i}] = f_{enc}^{\mathcal{S}}([\mathbf{x}_1^{\mathcal{S}_i}, \dots, \mathbf{x}_{n_i}^{\mathcal{S}_i}]) \qquad (2)$$

where the $f_{enc}^{\mathcal{S}}$ denotes sentence-level encoder, which can be any sequential encoder. We will also get the sentence representation $\mathbf{s}^{\mathcal{S}_i}$ for sentence $\mathcal{S}_i$ from this encoder. For LSTM, $\mathbf{s}^{\mathcal{S}_i}$ is the hidden state of the last time step; for BERT, $\mathbf{s}^{\mathcal{S}_i}$ is the output representation of the special marker [CLS].

Representing. For i-th entity pair $(e_{i,h}, e_{i,t})$ which expresses intra-sentential relations, where $e_{i,h}$ is the head entity and $e_{i,t}$ is the tail entity, their mentions co-occur in C sentences $S_{co-occur} = \{\mathcal{S}_{i_1}, \mathcal{S}_{i_2}, \dots, \mathcal{S}_{i_C}\}$ once or many times. In j-th co-occurred sentence $\mathcal{S}_{i_j}$, we use the entity mentions in $\mathcal{S}_{i_j}$ to represent head and tail entity. And we define that the context representation of this relation instance in $\mathcal{S}_{i_j}$ is the top K words correlated with the relations of these two mentions.

Specifically, head entity mention ranging from s-th to t-th word is represented as the average of the words it contains: $\mathbf{e}_{i,h}^{\mathcal{S}_{i_j}} = \frac{1}{t-s+1}\sum_{k=s}^{t}\mathbf{g}_k^{\mathcal{S}_{i_j}}$, so is the tail entity mention $\mathbf{e}_{i,t}^{\mathcal{S}_{i_j}}$. Then, we concatenate

---

[1] The existing doc-level RE datasets annotate which mentions belong to the same entity. So for each word in the document, it may belong to the i-th entity or non-entity in the document. We embed this co-reference information between entity mention (surface words) and entity (an abstract concept) into the initialized representation of a word.

[2] For those words not belonging to any entity, we introduce *None* entity type and id.

the representations of head and tail entity mentions and use it as a query to attend all words in $\mathcal{S}_{i_j}$ and compute relatedness score for each word in $\mathcal{S}_{i_j}$:

$$s_{i,k} = \sigma((W_{intra} \cdot [\mathbf{e}_{i,h}^{\mathcal{S}_{i_j}}; \mathbf{e}_{i,t}^{\mathcal{S}_{i_j}}])^T \cdot \mathbf{g}_k^{\mathcal{S}_{i_j}}) \tag{3}$$

$$\alpha_{i,k} = Softmax(s_{i,k}) \tag{4}$$

where $[\cdot;\cdot]$ is a concatenation operation. $W_{intra} \in \mathbb{R}^{d\times 2d}$ is a parameter matrix. σ is an activation function (e.g., ReLU).

Then, we average the representations of top K related words to represents the context information $\mathbf{c}_i$ for intra-sentential entity pair $(e_{i,h}, e_{i,t})$ in $\mathcal{S}_{i_j}$. In order to make W_{intra} trainable during computing gradient, we also add an item which is the weighted average representation of all words:

$$\mathbf{c}_i^{\mathcal{S}_{i_j}} = \beta\cdot\frac{1}{K}\sum_{k\in topK(\alpha_{i,*})}\mathbf{g}_k^{\mathcal{S}_{i_j}} + (1-\beta)\cdot\sum_t^{n_{i_j}}\alpha_{i,t}\mathbf{g}_t^{\mathcal{S}_{i_j}} \tag{5}$$

where β is a hyperparameter and we use 0.9 here to force model to focus on the top K words but still consider the subtle influence from other words.

Next, we concatenate the three parts obtained above to form the relational representation of intra-sentential entity pair $(e_{i,h}, e_{i,t})$ in $\mathcal{S}_{i_j}$ and further average the representations in all co-occured sentences $\mathcal{S}_{co-occur}$ to get our final relation representation $\mathbf{r}_i$ for intra-sentential entity pair $(e_{i,h}, e_{i,t})$ [3]:

$$\mathbf{r}_i = \frac{1}{C}\sum_{\mathcal{S}_{i_j}\in\mathcal{S}_{co-occur}}[\mathbf{e}_{i,h}^{\mathcal{S}_{i_j}}; \mathbf{e}_{i,t}^{\mathcal{S}_{i_j}}; \mathbf{c}_i^{\mathcal{S}_{i_j}}] \tag{6}$$

This way, we could force the intra-sentential entity pairs to focus on the semantic information from their co-occurred sentences and ignore the noise information from other sentences.

2.1.2 Inter-sentential Relation Representation Module

Encoding. According to the nature of inter-sentetential relation, we use a document-level encoder to capture the global interactions for inter-sentential relations and produce contextualized word embedding for each word. Formally, we convert a document $\mathcal{D}$ containing m words $\{w_i^{\mathcal{D}}\}_{i=1}^m$ into a sequence of vectors $\left\{\mathbf{g}_j^{\mathcal{D}}\right\}_{j=1}^m$.

Same as the embedding for intra-sentential relations, we use Equation 1 to embed each word in the document. Then the vectorized word representations are fed into the document-level encoder to obtain document-level context-sensitive representation for each word:

$$[\mathbf{g}_1^{\mathcal{D}}, \ldots, \mathbf{g}_m^{\mathcal{D}}] = f_{enc}^{\mathcal{D}}([\mathbf{x}_1^{\mathcal{D}}, \ldots, \mathbf{x}_m^{\mathcal{D}}]) \tag{7}$$

where $f_{enc}^{\mathcal{D}}$ denotes the document-level encoder. And we will also get the document representation $\mathbf{d}^{\mathcal{D}}$ from this encoder.

To further enhance the document interactions, we utilize the mention-level graph (MG) proposed by Zeng et al. (2020). MG in Zeng et al. (2020) contains two different nodes: mention node and document node. Each mention node denotes one particular mention of an entity. Furthermore, MG also has one document node that aims to model the document information. We argue that this graph only contains nodes concerning prediction, i.e., the mentions of the entities and document information. However, it does not contain the local context information, which is crucial for the interaction among entity mentions and the document. So we introduce a new type of node: sentence node and its corresponding new edges to infuse the local context information into MG.

So there are four types of edges[4] in MG:

Intra-Entity Edge: Mentions referring to the same entity are fully connected. This models the interactions among mentions of the same entity.

Inter-Entity Edge: Mentions co-occurring in the same sentence are fully connected. This models the interactions among different entities via co-occurrences of their mentions.

Sentence-Mention Edge: Each sentence node connects with all entity mentions it contains. This models the interactions between mentions and their local context information.

Sentence-Document Edge: All sentence nodes are connected to the document node. This models the interactions between local context information and document information, acting as a bridge between mentions and document.

Next, we apply Relational Graph Convolutional Network (R-GCN, Schlichtkrull et al., 2017) on MG to aggregate the features from neighbors for each node. Given node u at the l-th layer, the graph

---

[3] If a head entity mentioned N times in a sentence, we will get N intra-sentential relational representations for each of the other tail entities in this sentence.

[4] Note that we remove the mention-document edges of original MG in (Zeng et al., 2020) and substitute them by introducing mention-sentence and sentence-document edges.

convolutional operation can be defined as:

$$\mathbf{h}_u^{(l+1)} = ReLU\left(\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{N}_u^t\bigcup\{u\}}\frac{1}{c_{u,t}}W_t^{(l)}\mathbf{h}_v^{(l)}\right) \tag{8}$$

where $\mathcal{T}$ is a set of different types of edges, $W_t^{(l)} \in \mathbb{R}^{d\times d}$ is a trainable parameter matrix. $\mathcal{N}_u^t$ denotes a set of neighbors for node u connected with t-th type edge. $c_{u,t} = |\mathcal{N}_u^t|$ is a normalization constant.

We then aggregate the outputs of all R-GCN layers to form the final representation of node u:

$$\mathbf{m}_u = ReLU(W_u \cdot [\mathbf{h}_u^{(0)}; \mathbf{h}_u^{(1)}; \ldots; \mathbf{h}_u^{(N)}]) \tag{9}$$

where $W_u \in \mathbb{R}^{d\times Nd}$ is a trainable parameter matrix. $h_u^{(0)}$ is the initial representation of node u. For a mention ranging from the s-th word to the t-th word in the document, $\mathbf{h}_u^{(0)} = \frac{1}{t-s+1}\sum_{j=s}^{t}\mathbf{g}_j^{\mathcal{D}}$; for i-th sentence node, it is initialized with $\mathbf{s}^{\mathcal{S}_i}$ from sentence-level encoder; for the document node, it is initialized with $\mathbf{d}^{\mathcal{D}}$ from document-level encoder.

Representing. We argue that inter-sentential relations can be inferred from the following information sources: 1) the head and tail entities themselves; 2) the related sentences that signal their cross-sentence relations, namely supporting evidences; 3) reasoning information such as logical reasoning, co-reference reasoning, world knowledge, etc. We here only consider the first two information and leave the last in Sec. 2.2.

Different from intra-sentential relations, inter-sentential relations tend to be expressed from the global interactions. So for the i-th entity pair $(e_{i,h}, e_{i,t})$ which expresses inter-sentential relation, the head entity representation $\mathbf{e}_{i,h}$ and the tail entity representation and $\mathbf{e}_{i,t}$ are defined as the average of their entity mentions from MG:

$$\mathbf{e}_i = \frac{1}{N}\sum_{j\in M(e_i)}\mathbf{m}_j \tag{10}$$

where the $M(e_i)$ is the mention set of e_i.

And we apply an evidence selector with attention mechanism (Bahdanau et al., 2015) to encourage the inter-sentential entity pair to selectively focus on the sentences that express their cross-sentence relations. This process could be regarded as finding supporting evidence for their relations. So the context representation $\mathbf{c}_i$ for inter-sentential entity pair $(e_{i,h}, e_{i,t})$ is the weighted average of the sentence representations from MG:

$$P(\mathcal{S}_k|e_{i,h}, e_{i,t}) = \sigma(W_k \cdot [\mathbf{e}_{i,h}; \mathbf{e}_{i,t}; \mathbf{m}_{\mathcal{S}_k}]) \tag{11}$$

$$\alpha_{i,k} = \frac{P(\mathcal{S}_k|e_{i,h}, e_{i,t})}{\sum_l P(\mathcal{S}_l|e_{i,h}, e_{i,t})} \tag{12}$$

$$\mathbf{c}_i = \sum_k^l \alpha_{i,k}\cdot\mathbf{m}_{\mathcal{S}_k} \tag{13}$$

where $W_k \in \mathbb{R}^{1\times 2d}$ is a trainable parameter matrix. σ is a *sigmoid* function.

Next, the final relation representation for inter-sentential entity pair $(e_{i,h}, e_{i,t})$ should be:

$$\mathbf{r}_i = [\mathbf{e}_{i,h}; \mathbf{e}_{i,t}; \mathbf{c}_i] \tag{14}$$

2.2 Logical Reasoning Module

In this module, we focus on logical reasoning modeling. As mentioned in Sec. 1, previous works usually use the paths between each entity pair as the clues for logical reasoning. Furthermore, they concatenate the path representations with entity pair representations to predict relations. However, since not all entity pairs are connected with a path and have the correct logical reasoning paths in their graph, many cases of logical reasoning cannot be covered. So their methods are somehow limited.

In this paper, we utilize self-attention mechanism (Vaswani et al., 2017) to model logical reasoning. Specifically, we can get the relational representations for all entity pairs from the above sections. For i-th entity pair (e_h, e_t), we can assume there is a two-hop logical reasoning chains: $e_h \rightarrow e_k \rightarrow e_t$ in the document, where e_k can be any other entities in the document except e_h and e_t. So (e_h, e_t) can attend to all the relational representations of other entity pairs including (e_h, e_k) and (e_k, e_t), termed as $\mathcal{R}_{att}$. Finally, the weighted sum of $\mathcal{R}_{att}$ can be treated as a new relational representation for (e_h, e_t), which considers all possible two-hop logical reasoning chains in the document.[5]

$$\mathbf{r}_i^{new} = \sum_{\mathbf{r}_k\in\mathcal{R}_{att}\cup\{\mathbf{r}_i\}}\gamma_k\cdot\mathbf{r}_k \tag{15}$$

$$\gamma_k = Softmax((W_{att}\cdot\mathbf{r}_i)^T\cdot\mathbf{r}_k) \tag{16}$$

where $W_{att} \in \mathbb{R}^{3d\times 3d}$ is a parameter matrix.

In this way, the path in the previous works could be converted into the individual attention on every entity pair in the logical reasoning chains. We argue that this form of logical reasoning is simpler

---

[5] This can be scaled to muti-hop logical reasoning by increasing the self-attention layers. We only consider two-hop logical reasoning in this paper following Zeng et al. (2020).

and more scalable because it will consider all possible logical reasoning chains without connectivity constraints in the graph structure.

2.3 Classification Module

We formulate the doc-level RE task as a multi-label classification task:

$$P(r|e_{i,h}, e_{i,t}) = sigmoid\left(W_1\sigma(W_2\mathbf{r}_i + b_1) + b_2\right) \tag{17}$$

where W_1, W_2, b_1, b_2 are trainable parameters, σ is an activation function (e.g., ReLU). We use binary cross entropy as objective to train our SIRE:

$$\begin{aligned}\mathcal{L}_{rel} = -\sum_{\mathcal{D}\in\mathcal{C}}\sum_{h\neq t}\sum_{r_i\in\mathcal{R}} \mathbb{I}\left(r_i = 1\right)\log P\left(r_i|e_{i,h}, e_{i,t}\right) \\ + \mathbb{I}\left(r_i = 0\right)\log\left(1 - P\left(r_i|e_{i,h}, e_{i,t}\right)\right)\end{aligned} \tag{18}$$

where $\mathcal{C}$ denotes the whole corpus, $\mathcal{R}$ denotes relation type set and $\mathbb{I}\left(\cdot\right)$ refers to indicator function.

3 Experiments

3.1 Dataset

We evaluate our proposed model on three document-level RE datasets:

DocRED: The largest human-annotated document-level relation extraction dataset was proposed by Yao et al. (2019b). It is constructed from Wikipedia and Wikidata and contains 96 types of relations, $132,275$ entities, and $56,354$ relational facts in total. Documents in DocRED have about 8 sentences on average. More than 40.7% relation facts can only be extracted from multiple sentences. 61.1% relation instances require various reasoning skills such as logical reasoning. 93.4% intra-sentential relations can be inferred based solely on their co-occurred sentences. We show two examples from DocRED in Figure 1. We follow the standard split of the dataset, $3,053$ documents for training, $1,000$ for development, and $1,000$ for testing.

CDR (BioCreative V): The Chemical-Disease Reactions dataset was created by Li et al. (2016) manually. It contains one type of relation: *Chemical-Induced-Disease* between chemical and disease entities. We follow the standard split of the dataset, 500 documents for training, 500 for development, and 500 for testing.

GDA (DisGeNet): The Gene-Disease-Associations dataset was introduced by Wu et al. (2019). It contains one type of relation: *Gene-Induced-Disease* between gene and disease entities. We use standard split of the dataset, $23,353$ documents for training, $5,839$ for development, and $1,000$ for testing.

3.2 Experimental Settings

In our SIRE implementation, we use 3 layers of GCN, use ReLU as our activation function, and set the dropout rate to 0.3, learning rate to 0.001. We train SIRE using AdamW (Loshchilov and Hutter, 2019) as optimizer with weight decay 0.0001 and implement SIRE under PyTorch (Paszke et al., 2017) and DGL (Wang et al., 2019b) frameworks.

We implement two settings for our SIRE. **SIRE-GloVe** uses GloVe (100d, Pennington et al., 2014) and BiLSTM (512d, Schuster and Paliwal, 1997) as word embedding and encoder, respectively. **SIRE-BERT** use BERT-base (Devlin et al., 2019) as encoder on DocRED, cased BioBERT-Base v1.1 as the encoder on CDR/GDA, and the learning rate for BERT parameters is set to $1e^{-5}$ and learning rate for other parameters remains $1e^{-3}$. Detailed hyperparameter settings are in Appendix.

3.3 Baselines and Evaluation Metrics

We use the following models as our baselines:

Yao et al. (2019b) propose the **BiLSTM** (Schuster and Paliwal, 1997) as the encoder on DocRED and use the output from the encoder to represent all entity pairs to predict relations.

Wang et al. (2019a) propose **BERT** to replace the BiLSTM as the encoder on DocRED. Moreover, they also propose **BERT-Two-Step**, which first predicts whether two entities have a relation and then predicts the specific target relation.

Tang et al. (2020) propose the hierarchical inference networks **HIN-GloVe** and **HIN-BERT**, which make full use of multi-granularity inference information including entity level, sentence level, and document level to infer relations.

Similar to Wang et al. (2019a), Ye et al. (2020) propose a language representation model called **CorefBERT** as encoder on DocRED that can capture the coreferential relations in context.

Nan et al. (2020) propose the **LSR-GloVe** and **LSR-BERT** to dynamically induce the latent dependency tree structure to better model the document interactions for prediction.

Wang et al. (2020) propose a global-to-local network **GLRE**, which encodes the document information in terms of entity global and local representations as well as context relation representations.

Zeng et al. (2020) propose the graph aggregation-and-inference networks **GAIN-GloVe** and **GAIN-**

Model	Dev				Test	
	Ign F1	F1	Intra-F1	Inter-F1	Ign F1	F1
BiLSTM (Yao et al., 2019b)	48.87	50.94	57.05	43.49	48.78	51.06
HIN-GloVe (Tang et al., 2020)	51.06	52.95	-	-	51.15	53.30
LSR-GloVe (Nan et al., 2020)	48.82	55.17	60.83	48.35	52.15	54.18
GAIN-GloVe (Zeng et al., 2020)	53.05	55.29	61.67	48.77	52.66	55.08
SIRE-GloVe	**54.10**	**55.91**	**62.94**	**48.97**	**54.04**	**55.96**
-LR Module	53.73	55.58	62.77	47.87	53.75	55.55
-context	52.57	54.41	61.66	46.92	52.33	54.15
-inter4intra	52.23	54.26	60.81	48.36	51.77	53.30
BERT (Wang et al., 2019a)	-	54.16	61.61	47.15	-	53.20
BERT-Two-Step (Wang et al., 2019a)	-	54.42	61.80	47.28	-	53.92
HIN-BERT (Tang et al., 2020)	54.29	56.31	-	-	53.70	55.60
CorefBERT (Ye et al., 2020)	55.32	57.51	-	-	54.54	56.96
GLRE-BERT (Wang et al., 2020)	-	-	-	-	55.40	57.40
LSR-BERT (Nan et al., 2020)	52.43	59.00	65.26	52.05	56.97	59.05
GAIN-BERT (Zeng et al., 2020)	59.14	61.22	67.10	53.90	59.00	61.24
SIRE-BERT	**59.82**	**61.60**	**68.07**	**54.01**	**60.18**	**62.05**

Table 1: Performance on DocRED. Models above the double line do not use pre-trained model. *LR Module* is the logical reasoning module. *context* denotes context representations in Eq. 6 and Eq. 14. *inter4intra* denotes using the inter-sentential module also for intra-sentential entity pairs.

Model	CDR	GDA
BRAN (Verga et al., 2018)	62.1	-
EoG (Wang et al., 2020)	63.6	81.5
LSR (Nan et al., 2020)	64.8	82.2
GLRE-BioBERT (Wang et al., 2020)	68.5	-
SIRE-BioBERT	**70.8**	**84.7**

Table 2: Performance on CDR and GDA.

BERT which utilize two levels of graph structures: mention-level graph and entity-level graph to capture document interactions and conduct path logical reasoning mechanism, respectively.

Verga et al. (2018) propose a self-attention encoder **BRAN** to consider interactions across mentions and relations across sentence boundaries.

Following the previous works (Yao et al., 2019b; Zeng et al., 2020), we use the F1 and Ign F1 as the evaluation metrics to evaluate the overall performance of a model. The Ign F1 metric calculates F1 excluding the common relation facts in the training and dev/test sets. We also use the intra-F1 and inter-F1 metrics to evaluate a model's performance on intra-sentential relations and inter-sentential relations on the dev set.

3.4 Results

The performances of SIRE and baseline models on the DocRED dataset are shown in Table 1. Among the model not using BERT encoding, SIRE outperforms the previous state-of-the-art model by 0.88/1.38 F1/Ign F1 on the test set. Among the model using BERT encoding, SIRE outperforms the previous state-of-the-art models by 1.18/0.81 F1/Ign F1 on the test set. The improvement on Ign F1 is larger than that on F1. This shows SIRE has a stronger generalization ability on the unseen relation instances. On intra-F1 and inter-F1, we can observe that SIRE is better than the previous models that indiscriminately represent the intra- and inter-sentential relations in the same way. This demonstrates that representing intra- and inter-sentential relations in different methods is better than representing them in the same way. The improvement on intra-F1 is greater than the improvement on inter-F1. This shows that SIRE mainly improves the performance of intra-sentential relations. The performances of SIRE and baseline models on the CDR/GDA dataset are shown in Table 2, which are consistent with the improvement on DocRED.

3.5 Ablation Study

To further analyze SIRE, we also conduct ablation studies to illustrate the effectiveness of different modules in SIRE. We show the results in Table 1.
1) **the importance of the logical reasoning module**: When we discard the logical reasoning module, the performance of SIRE-GloVe decreases by 0.41 F1 on the DocRED test set. This shows the effectiveness of our logical reasoning module, which can better model the reasoning information in the document. Moreover, it drops significantly on inter-F1 and drops fewer points on intra-F1. This shows our logical reasoning module mainly improves the performance of the inter-sentential relations that usually require reasoning skills.
2) **Ablation on context representations in Eq. 6**

Type	Examples
Intra-sentential relation instances	"***Your Disco Needs You***" is a song performed by Australian recording artist and songwriter ***Kylie Minogue***, taken from her seventh studio album *Light Years* (2000). **Relation:** performer
	Lark Force was an *Australian Army* formation established in ***March 1941*** during *World War II* for service in *New Britain* and *New Ireland*. **Relation:** inception
	Lake Hiawatha is one of the few lakes through which *Minnehaha Creek* flows , and the last one before it reaches ***Minnehaha Falls*** and then the ***Mississippi River***. **Relation:** mouth of the watercourse
Inter-sentential relation instances	[1] (0.87) *IBM Research–Brazil* is one of twelve research laboratories comprising *IBM Research*, its first in ***South America***. [2] (0.66) It was established in *June 2010*, with locations in ***São Paulo*** and *Rio de Janeiro*. [3] (0.01) Research focuses on Industrial Technology and Science, … [4] (0.04) The new lab, *IBM*'s ninth … [5] (0.38) In collaboration with *Brazil*'s government, it will help *IBM* to develop … **Relation:** continent **Logical reasoning attention weight:** (São Paulo, Brazil) 0.32 (São Paulo, June 2010) 0.03 … (Brazil, South America) 0.45 (June 2010, South America) 0.02 …

Figure 3: Cases for illustrating the reliable and explainable predictions of our SIRE. Head entities, tail entities, and sentence numbers along with the scores from evidence selector are colored in blue, red, green, respectively. In intra-sentential relations, words with pink background color are the top 4 words from Equation 5.

and Eq. 14: When we remove the context representations in intra- and inter-sentential relational representations, the performance of SIRE-GloVe on the DocRED test set drops by 1.81 F1. This shows context information (top K words for intra, evidence sentences for inter) is important for both intra- and inter-sentential relation representation.
3) **Using the inter-sentential module also for intra-sentential entity pairs**: In this experiment, we do not distinguish these two types of relations, using the encoding method for inter-sentential to encode all entity pairs, and remain the logical reasoning module unchanged. The performance of SIRE-GloVe drops by 2.66/2.13 F1/intra-F1 on the DocRED test set. This confirms the motivation that we cannot use global information to learn the local patterns for intra-sentential relations.

3.6 Reasoning Performance

Furthermore, we evaluate the reasoning ability of our model on the development set in Table 3. We use infer-F1 as the metric that considers only two-hop positive relation instances in the dev set. So it will naturally exclude many cases that do not belong to the two-hop logical reasoning process to strengthen the evaluation of reasoning performance. As Table 3 shows, SIRE is superior to previous models in handling the two-hop logical reasoning process. Moreover, after removing the logical reasoning module, out SIRE drops significantly on infer-F1. This shows that our logical reasoning module plays a crucial role in modeling the logical reasoning process.

Model	Infer-F1	P	R
BiLSTM	38.73	31.60	50.01
GAIN-GloVe	40.82	32.76	54.14
SIRE-GloVe	**42.72**	**34.83**	**55.22**
- *LR Module*	39.18	31.97	50.59

Table 3: Infer-F1 results on dev set of DocRED. P: Precision, R: Recall.

3.7 Case Study

Figure 3 shows the prediction cases of our SIRE. In intra-sentential relations, the top 4 words related to the relations of three entity pairs conform with our intuition. Our model correctly find the words by using Eq.5 that trigger the relations of these entity pairs. In inter-sentential relations, the supporting evidence that the model finds, i.e., sentences 1 and 2, indeed expresses the relations between São Paul and South America. We also conduct logical reasoning in terms of the logical reasoning chains: São Paul→ other-entity → South America. Our SIRE could focus on the correct logical reasoning chains: São Paul→ Brazil → South America. These cases show the predictions of SIRE are explainable.

4 Related Work

Document-level relation extraction. Many recent efforts (Quirk and Poon, 2017; Peng et al., 2017; Gupta et al., 2019; Song et al., 2018; Jia et al., 2019; Yao et al., 2019b; Wang et al., 2019a; Tang et al., 2020; Nan et al., 2020; Zeng et al., 2020; Wang et al., 2020; Dai et al., 2020) manage to tackle the document-level relation extraction. Most of them use graph-based models, such as Graph Convolutional Networks (GCNs, Kipf and Welling, 2017; Schlichtkrull et al., 2017) that has been used in many natural language processing tasks (Marcheggiani and Titov, 2017; Yao et al., 2019a; Liu et al., 2020). They construct a graph structure from the input document. This graph uses the word, mentions or entities as nodes and uses heuristic rules and semantic dependencies as edges. They use this graph to model document information and interactions and to predict possible relations for all entity pairs. Nan et al. (2020) proposed a latent structure induction to induce the dependency tree in the document dynamically. Zeng et al. (2020) proposed a double graph-based graph aggregation-and-inference network that constructs two graphs: mention-level graph and entity-level graph. They use the former to capture the document information and interactions among entity mentions and document and use the latter to conduct path-based logical reasoning. However, these works do not explicitly distinguish the intra- and inter-sentential relation instances in the design of the model and use the same way to encode them. So the most significant difference between our model and previous models is that we treat intra-sentential and inter-sentential relations differently to conform with the relational patterns for their prediction.

Reasoning in relation extraction. Reasoning problem has been extensively studied in the field of question answering (Dhingra et al., 2018). However, few works manage to tackle this problem in the document-level relation extraction task. Zeng et al. (2020) is the first to propose the explicit way of relational reasoning on doc-level RE, which mainly focuses on logical reasoning. They use the paths on their entity-level graph to provide clues for logical reasoning. However, since not all entity pairs are connected with a path and have the correct logical reasoning paths in their graph, their methods are somehow limited. In this work, we design a new form of logical reasoning to cover more cases of logical reasoning.

5 Conclusion

Intra- and inter-sentential relations are two types of relations in doc-level RE. We propose a novel architecture, SIRE, to represent these two relations in different ways separately in this work. We introduce a new form of logical reasoning module that models logical reasoning as a self-attention among representations of all entity pairs. Experiments show that our SIRE outperforms the previous state-of-the-art methods. The detailed analysis demonstrates that our predictions are explainable. We hope this work will have a positive effect on future research regarding new encoding schema, a more generalizable and explainable model.

Acknowledgments

The authors would like to thank the reviewers for their thoughtful and constructive comments. This paper is supported by the National Key R&D Program of China under Grand No.2018AAA0102003, the National Science Foundation of China under Grant No.61936012 and 61876004.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4925–4936.

Damai Dai, Jing Ren, Shuang Zeng, Baobao Chang, and Zhifang Sui. 2020. Coarse-to-fine entity representations for document-level relation extraction. *Computing Research Repository*, arXiv:2012.02507.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 42–48.

Pankaj Gupta, Subburam Rajaram, Hinrich Schütze, and Thomas A. Runkler. 2019. Neural relation extraction within and across sentence boundaries. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 6513–6520.

Robin Jia, Cliff Wong, and Hoifung Poon. 2019. Document-level n-ary relation extraction with multiscale representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 3693–3704.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Sennan Liu, Shuang Zeng, and Sujian Li. 2020. Evaluating text coherence at sentence and paragraph levels. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1695–1703. European Language Resources Association.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations, ICLR 2019*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1546–1557.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4309–4316.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *Computing Research Repository*, arXiv:1703.06103.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state LSTM. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2226–2235.

Hengzhu Tang, Yanan Cao, Zhenyu Zhang, Jiangxia Cao, Fang Fang, Shi Wang, and Pengfei Yin. 2020. HIN: hierarchical inference network for document-level relation extraction. In *Advances in Knowledge Discovery and Data Mining - 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11-14, 2020, Proceedings, Part I*, volume 12084 of *Lecture Notes in Computer Science*, pages 197–209.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884.

Difeng Wang, Wei Hu, Ermei Cao, and Weijian Sun. 2020. Global-to-local neural networks for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods* *in Natural Language Processing (EMNLP)*, pages 3711–3721.

Hong Wang, Christfried Focke, Rob Sylvester, Nilesh Mishra, and William Wang. 2019a. Fine-tune bert for docred with two-step process. *Computing Research Repository*, arXiv:1909.11898.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. 2019b. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Ye Wu, Ruibang Luo, Henry C. M. Leung, Hing-Fung Ting, and Tak Wah Lam. 2019. RENET: A deep learning approach for extracting gene-disease associations from literature. In *Research in Computational Molecular Biology - 23rd Annual International Conference, RECOMB 2019, Washington, DC, USA, May 5-8, 2019, Proceedings*, volume 11467, pages 272–284.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019a. Graph convolutional networks for text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7370–7377.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019b. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential Reasoning Learning for Language Representation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7170–7186.

Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1630–1640.

A Hyperparameter settings

We use the development set to manually tune the optimal hyperparameters for SIRE, based on the Ign F1 score. Experiments are run on NVIDIA-RTX-3090-24GB GPU. Hyperparameter settings for SIRE-GloVe, SIRE-BERT on DocRED are listed in Table 4, 5, respectively. The values of hyperparameters we finally adopted are in bold. Note that we do not tune all the hyperparameters.

Hyperparameter	Value
Batch Size	16, **32**
Learning Rate	**0.001**
Activation Function	**ReLU**, Tanh
Positive v.s. Negative Ratio	1, 0.5, **0.25**
Word Embedding Size	**200**
Entity Type Embedding Size	**20**
Coreference Embedding Size	**20**
Encoder Hidden Size	256, **512**
Dropout	0.3, **0.5**, 0.7
Layers of GCN	1, 2, **3**
GCN Hidden Size	**1024**
Weight Decay	**0.0001**
β	**0.9**
Numbers of Parameters	95M
Training Time	18 hours
Hyperparameter Search Trials	20

Table 4: Settings for SIRE-GloVe.

Hyperparameter	Value
Batch Size	16, **32**
Learning Rate	**0.001**
Activation Function	**ReLU**, Tanh
Positive v.s. Negative Ratio	1, 0.5, **0.25**
Entity Type Embedding Size	**128**
Coreference Embedding Size	**128**
Dropout	**0.3**, 0.5, 0.7
Layers of GCN	1, 2, **3**
GCN Hidden Size	**1024**
Weight Decay	**0.0001**
β	**0.9**
Numbers of Parameters	307M
Training Time	24 hours
Hyperparameter Search Trials	30

Table 5: Settings for SIRE-BERT.

KGPool: Dynamic Knowledge Graph Context Selection for Relation Extraction

Abhishek Nadgeri[1,2], **Anson Bastos**[1,3], **Kuldeep Singh**[1,5], **Isaiah Onando Mulang'**[1,7]
Johannes Hoffart[6], **Saeedeh Shekarpour**[4], **and Vijay Saraswat**[6]
[1]Zerotha Research, [2]RWTH Aachen, [3]IIT Hyderabad, [4]University of Dayton
[5]Cerence GmbH, [6]Goldman Sachs, [7]IBM Research, Africa
`abhishek.nadgeri@rwth-aachen.de, cs20resch11002@iith.ac.in`
`kuldeep.singh1@cerence.com, mulang.onando@ibm.com`
`sshekarpour1@udayton.edu,{johannes.hoffart,vijay.saraswat}@gs.com`

Abstract

We present a novel method for relation extraction (RE) from a single sentence, mapping the sentence and two given entities to a canonical fact in a knowledge graph (KG). Especially in this presumed sentential RE setting, the context of a single sentence is often sparse. This paper introduces the *KGPool* method to address this sparsity, dynamically expanding the context with additional facts from the KG. It learns the representation of these facts (entity alias, entity descriptions, etc.) using neural methods, supplementing the sentential context. Unlike existing methods that statically use all expanded facts, *KGPool* conditions this expansion on the sentence. We study the efficacy of *KGPool* by evaluating it with different neural models and KGs (Wikidata and NYT Freebase). Our experimental evaluation on standard datasets shows that by feeding the *KGPool* representation into a Graph Neural Network, the overall method is significantly more accurate than state-of-the-art methods.

1 Introduction

Knowledge graphs (KGs) are the foundation for many downstream applications and are growing ever larger. However, due to the sheer volume of knowledge and the world's dynamic nature where new entities emerge and unknown facts about them are learned, KGs need to be continuously updated. Distantly supervised Relation Extraction (RE) is an important KG completion task aiming at finding a semantic relationship between two entities annotated on the unstructured text with respect to an underlying knowledge graph (Ye and Ling, 2019). In the literature, researchers mainly studied two variants in the RE: 1) multi-instance RE and 2) sentential RE. The multi-instance RE assumes that in a given bag of sentences, if two entities participate in a relation, there exists at least one sentence with these two entities, which may contain the target relation (Riedel et al., 2010; Vashishth et al., 2018). In this setting, researchers aim to incorporate contextual signals from the previous occurrences of an entity pair into the neural models to support relation extraction (Ye and Ling, 2019; Xu and Barbosa, 2019; Wu et al., 2019). In contrast, sentential RE restricts the scope of document context only to the input sentence (disregards other occurrences of entity pairs) while predicting the KG relation (Sorokin and Gurevych, 2017). Hence, sentential RE makes the multi-instance setting more difficult by limiting the available context.

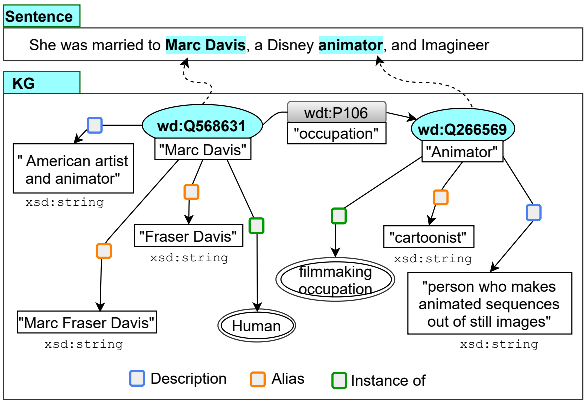

Figure 1: Illustration of Knowledge Graph Context associated with the annotated entities in a sentence. Here, entity aliases do not play any role in the understanding of the sentence for finding the KG relation. [a]

Recent approaches for RE not only base KGs for relation inventory but also consider it for extending contextual knowledge for further improvement of RE task (Vashishth et al., 2018; Bastos et al., 2021). A few multi-instance RE methods rely on entity attributes (properties) such as descriptions, aliases, and types (as additional context) along with entity pair occurrences from previous sentences to improve the overall extraction quality (Ji et al., 2017; Vashishth et al., 2018). For

---

[a] sentence taken from (Sorokin and Gurevych, 2017)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 535–548
August 1–6, 2021. ©2021 Association for Computational Linguistics

the sentential RE, the RECON approach (Bastos et al., 2021) aims to effectively encode KG context derived from the entity attributes and entity neighborhood triples. RECON employs a Graph Neural Network (GNN) as a context aggregator for combining sentential context (annotated entities and sentence) and structured KG representation. Although the additional KG context has a positive effect on the overall relation extraction in multi-instance and sentential RE settings, not all KG context forms are necessary for every input sentence. Consider Figure 1, where the task is to infer a semantic relation 'occupation' between two entities *wd:Q568631 (Marc Davis)*[1] and *wd:Q266569 (animator)*. Wikidata (Vrandecic, 2012) KG provides semantic information such as description, instance-of, and aliases about entities. Here, the entity alias (Marc Fraser Davis, Fraser Davis for *wd:Q568631*; and cartoonist for *wd:Q266569*) has no impact on understanding the sentence because the entities are explicitly mentioned in the sentence. Furthermore, there is empirical evidence in the literature that for several sentences, statically adding all KG context offered minimal or negative impact (Bastos et al., 2021). Hence, there are open research questions as to how an RE approach can dynamically utilize the sufficient context from KG and whether or not the selected KG context positively impacts the overall performance?

This paper studies these concerning questions proposing the *KGPool* approach. *KGPool* utilizes a self-attention mechanism in a Graph Convolution Network (GCN) (Kipf and Welling, 2017) for selecting a sub-graph from the KG to extend the sentential context. The concept of dynamically mapping the structural representation of a KG to a latent representation of a sentence has not been widely studied in prior literature. In RE, *KGPool* is the initial attempt. The existing approaches (Bastos et al., 2021; Xu and Barbosa, 2019; Wu et al., 2019; Vashishth et al., 2018) feed all the available context (either derived from a bag of sentences or a KG or both) into a neural model and relied on the model to figure out the consequences, resulting in limited performance in many cases (Bastos et al., 2021). Conversely, we study the efficacy of *KGPool* in dynamically choosing KG context for the sentential RE task using two standard community datasets (NYT Freebase (Riedel et al., 2010), and Wikidata (Sorokin and Gurevych, 2017)). Our work makes the following key contributions:

- The *KGPool* approach dynamically selects structural knowledge and transform it into a representation suitable to supplement the latent representation of sentential context learned using a neural model. We deduce that *KGPool* is the first approach that works independently of the underlying context aggregators used in the literature (Graph Neural Network (Zhu et al., 2019) or LSTM-based (Sorokin and Gurevych, 2017)).
- We are the first to map the task of KG Context Selection to a Graph Pooling Problem. Therefore, our proposed approach legitimizes the application of graph pooling algorithms for choosing the relevant context.
- *KGPool*, paired with a GNN as context aggregator, significantly outperforms the existing baselines on both datasets, in one experiment increasing the precision by 12 points over to baseline (P@30 on NYT Freebase). Furthermore, our empirical results (cf., Table 3) conclude that an LSTM model paired with *KGPool* is able to notably outperform a GNN-based approach (Zhu et al., 2019) and nearly all multi-instance baselines (Ye and Ling, 2019; Wu et al., 2019; Vashishth et al., 2018) published in the recent years.

This paper is structured as follows: Section 2 reviews the related work. Section 3 formalizes the problem and the proposed approach is described in Section 4. Section 5 describes the experimental setup. The results are in Section 6. We conclude in Section 7.

2 Related Work

Multi-instance RE: a few multi-instance RE approaches use convolution neural network (dos Santos et al., 2015), attention CNN (Wang et al., 2016) and attention-based recurrent neural models for relation extraction (Zhou et al., 2016). Other approaches such as (Ji et al., 2017; Vashishth et al., 2018) incorporate entity descriptions, entity and relation aliases from KG to supplement context from the previous sentences. Work in (Vashishth et al., 2018) employs a graph convolution network to encode entity and relation aliases derived from Wikidata. HRERE (Xu and Barbosa, 2019) proposes an approach for jointly

---

[1] wd: binds to `https://www.wikidata.org/wiki/`

learning sentence and KG representation using cross-entropy loss function. To effectively capture the available entity context in the documents, Ye and Ling (2019) suggest an approach incorporating intra-bag and inter-bag attentions. For a detailed survey, we point readers to (Smirnova and Cudré-Mauroux, 2018).

Sentential RE: researchers (Sorokin and Gurevych, 2017) utilized additional relations present in the sentence to assist the process of extracting the target relation using an LSTM-based model. GP-GNN (Zhu et al., 2019) generates parameters of GNN based on the input sentence, which enables GNNs to process-relational reasoning on unstructured text inputs. RECON (Bastos et al., 2021) is an approach that uses the entity attributes (aliases, labels, descriptions, instance-of) and KG triples to signal an underlying GNN model for sentential RE. Authors conclude that the multi-instance requirement can be relaxed provided a good representation of KG context to enrich the sentential RE model. However, RECON and multi-instance approaches (Xu and Barbosa, 2019; Vashishth et al., 2018) utilize statically derived context from the KG, i.e., KG context does not vary depending the sentence.

Graph Pooling and Dynamic Context Selection: researchers proposed several models for the graph classification aka. graph pooling task (Cangea et al., 2018; Ying et al., 2018; Gao and Ji, 2019). These models employ various approaches such as graph topology-based (Rhee et al., 2018), and by learning the hierarchical graph-structure (Ying et al., 2018). Another graph pooling model relies on node features and topological information using self-attention (Lee et al., 2019) in which a specific number of nodes are always eliminated. In *KGPool*, the elimination of nodes depends on a context coefficient and node importance (Section 4). For context selection, a recent work focuses on dynamically selecting the KG context to optimize a Pre-Trained Language Model (PLM) for entity typing and relation classification (Su et al., 2020). *KGPool* has the following fundamental differences compared to (Su et al., 2020): *KGPool* inspires its self-attention mechanism from (Lee et al., 2019; Vaswani et al., 2017) to learn a representation of the KG context. Hence, *KGPool* works agnostic of the underlying model used for the context aggregation (unlike Su et al. (2020), which is tightly coupled with PLM). Approaches such as (Liu et al., 2017; Zhang et al., 2018; Kang et al., 2020) also perform dynamic context selection for respective tasks. However, these approaches are not focused on knowledge graph context selection.

3 Problem Statement

We define the KG as a tuple given by $KG = (\mathcal{E}, \mathcal{R}, \mathcal{T}^+)$ where $\mathcal{E}$ denotes the set of all vertices in the graph representing entities, $\mathcal{R}$ is the set of edges representing relations, and $\mathcal{T}^+ \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of all KG triples. The *RE Task* predicts the target relation $r^c \in \mathcal{R}$ between a given pair of entities $\langle e_i, e_j \rangle$ from the sentence $\mathcal{W} = (w_1, w_2, ..., w_l)$. If no relation is inferred, it returns 'NA' label. We aim for the sentential RE task which put a constraint that the sentence within which a given pair of entities occurs is the only visible sentence from the bag of sentences. We view RE as a classification task similar to (Sorokin and Gurevych, 2017). In a KG triple $\tau = (e_h, r, e_t) \in \mathcal{T}^+$, the relation $r \in \mathcal{R}$, e_h is the head entity (relation origin) while e_t is the tail entity. For each entity, associated semantic properties such as entity label, description, instance-of, and aliases are known as entity attribute (At^e) (cf., graph construction step of Figure 2). We aim to model KG contextual information to improve the classification. This is achieved by learning the effective representations of the sets At^e, e_h, e_t, and $\mathcal{W}$ (cf. section 4).

4 KGPool Approach

KGPool consists of three components illustrated in Figure 2: 1) *Graph Construction* aggregates the sentence, entities and its attributes as a Heterogeneous Information Graph (HIG) for input representation. 2) *Context Pooling* step utilizes a self-attention mechanism in a graph convolution to calculate attention scores for entity attributes using node features and graph topology. The pooling process allows *KGPool* to construct a Context Graph (CG), which is a contextualized representation of HIG with lesser number of nodes. 3) *Context Aggregator* takes as input the sentence, entities, contextual representations of HIG, and classifies the target relation between the entities. We detail the approach in the following.

4.1 Graph Construction

As first step, we extract entity attributes (At^e) from public dumps of Freebase (Bollacker et al.,

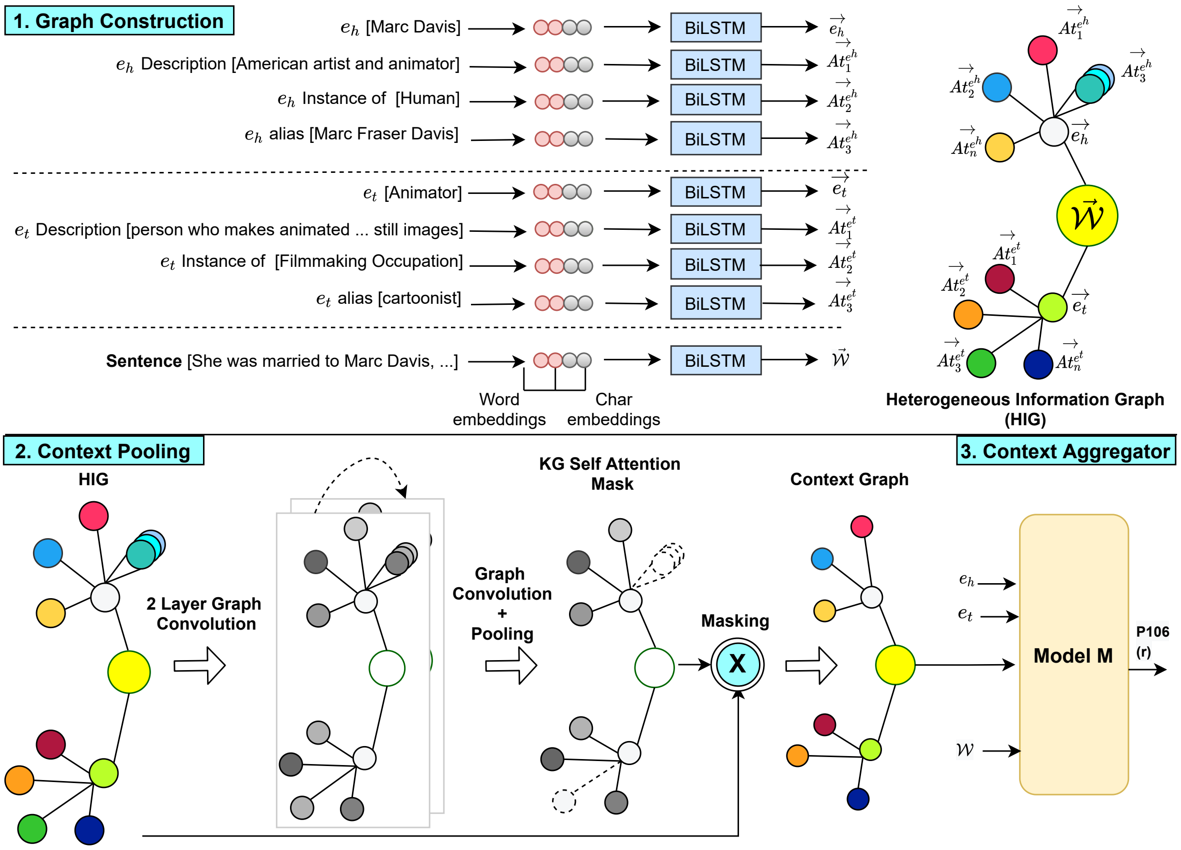

Figure 2: *KGPool* approach has three components to supplement sentential context with necessary KG context.

2007) and Wikidata (Vrandecic, 2012) KGs depending on the dataset. For the KG context, we rely on the most commonly available properties of the entities as suggested by (Bastos et al., 2021): aliases, description, instance-of, and label. An example of various entity attributes is given in Figure 2 at the Graph Construction step. Then, the sentence $\mathcal{W}$ is transformed to another representation using Bi-LSTM (Schuster and Paliwal, 1997) by concatenating its word and character embeddings:

$$\vec{\mathcal{W}} = BiLSTM(\mathcal{W}) \tag{1}$$

Similar representation is created for each entity e_i where $i = (h, t)$:

$$\vec{e_i} = BiLSTM(e_i) \tag{2}$$

For entity e_i, its KG contexts (entity attributes) $At_j^{e_i}$ (where $j = [0...N]$) are independently converted into associated embedding representations:

$$\vec{At_j}^{e_i} = BiLSTM(At_j^{e_i}) \tag{3}$$

For a knowledge representation of the KG context concerning the sentential context (sentence and annotated entities), we introduce the special graph $HIG = (A, F)$, a Heterogeneous Information Graph, represented in the adjacency matrix $A \in \{0,1\}^{n \times n}$, where n is the maximum number of neighboring nodes for an entity e_i. Here, $F \in \mathbb{R}^{n \times f}$ is the node feature matrix assuming each node has f features learned from the Bi-LSTM in the equations 1, 2, and 3. In these equations, BERT (Devlin et al., 2019), or any other recent Transformer-based model can be used. Due to hardware limitations, we are bound to Bi-LSTM using Glove embeddings (Pennington et al., 2014).

4.2 Context Pooling

Context pooling is built upon three layers of Graph Convolutional Networks (GCN) and a readout layer associated with each of them. Moreover, the last layer of GCN is coupled with a pooling layer (cf., ablation studies for architectural design choice experiments).

4.2.1 Graph Convolution

Since *KGPool* is expected to select the sufficient context, the Context Graph CG is a reduction of HIG using the mapping $\Psi : HIG \longrightarrow CG$. The challenge here is the no natural notion of spatial locality, i.e., it is not viable to pool together all context nodes in an "m × m" patch on HIG because the complex topological structure of graphs prevents any straightforward, deterministic definition of a "patch". Furthermore, entities nodes have a varying number of neighboring nodes, making the graph pooling challenging (similar to other graph classification problems (Ying et al., 2018)). In HIG, entity nodes do not contain information of their neighbors. Hence, we aim to enrich each entity node with the adjacent node's contextual information. Therefore, we employ a GNN variant to utilize its message-passing architecture to learn node embeddings from a message propagation function. The message propagation function depends on the adjacency matrix A, trainable parameters θ, and the node embeddings F (Ying et al., 2018). We rely on a GCN model by Kipf and Welling (2017). The GCN layer is defined as:

$$F^{(k)} = \text{ReLU}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}F^{(k-1)}\theta^{(k-1)}\right) \quad (4)$$

where $\tilde{A} = A + I$, $\tilde{D} = \sum_j \tilde{A_{i,j}}$ and $\theta^{(k)}$ is the trainable matrix. The GCN module might run k iteration and normally is in the range of two to six (Ying et al., 2018). A few graph representation learning approaches propose to use readout layer that aggregates node features to learn a fixed size representation (Xu et al., 2018; Cangea et al., 2018). We perform this summarizing after each block of the network (Equation 4), and aggregate all of the intermediate representation together by taking their sum. We define readout layer R as:

$$R^{(k)} = \frac{1}{N}\sum_{i=1}^{N} F_i^{(k)} \| \max_{i=1}^{N} F_i^{(k)} \quad (5)$$

where N is the number of nodes in the graph and F is the node feature embedding.

4.2.2 KG Self-Attention Mask

Until Equation 5, *KGPool* focuses on learning node features. Next, *KGPool* learns the importance of each entity attribute node using self-attention. Please note, in HIG, pooling happens only for entity attribute nodes ($\vec{At_j}^{e^i}$ from Equation 3). The sentence $\vec{\mathcal{W}}$ and entities $\vec{e_h}, \vec{e_t}$ remain intact. Hence, each entity representation $\vec{e_h}$ and $\vec{e_t}$ is enriched by the useful attribute context (KG context). The entity attribute nodes which do not provide relevant context are excluded from the graph. To choose the relevant entity attribute nodes, we use a self-attention score Z (Lee et al., 2019) calculated as follows:

$$Z = \tanh\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}F^{(k)}\Theta_{att}\right) \quad (6)$$

where $\Theta_{att} \in \mathbb{R}^{F\times 1}$ is the only parameter of the pooling layer. For ranking, we take the attention score and pass it through a softmax layer where Z_{score} is the normalized self attention score.

$$Z_{score} = \exp(Z_i) / \sum_i \exp(Z_i) \quad (7)$$

After Equation 7, we compute the scores for each entity attribute node. Next, we propose a node selection method in which nodes are selected on the basis of **Context Coefficient** α which is a hyper parameter. The top nodes are selected as:

$$idx = \max(Z_{score}) - \alpha * \sigma(Z_{score}) \quad (8)$$

where $\sigma(Z_{score})$ is the standard deviation of Z_{score}, idx represents the node selection result, and Z_{mask} is the corresponding attention mask. Equation 8 acts as a soft constraint in selecting the context nodes for each HIG which depends on the value of α. Learning α during training may cause over-fitting. Hence, we decided to consider α as a trade-off parameter similar to λ in regularization (Bühlmann and Van De Geer, 2011). Next, the Context Graph (CG) is formed by pooling out the less essential entity attribute nodes as:

$$F' = F^{(k)}_{\text{idx},:}, F_{out} = F' \odot Z_{mask}, A_{out} = A_{\text{idx},\text{idx}} \quad (9)$$

In addition to the dynamically selected nodes, we also inherit the intermediate node and graph representations of $k-1$ layers similar to ResNET (He et al., 2016). The intermediate representations ($k-1$) and the CG (k^{th} layer) is given as follows:

$$\begin{aligned} \vec{e_h}' &= F^{(1)}_{e_h} \oplus F^{(2)}_{e_h} \oplus ....F^{(k)}_{e_h} \\ \vec{e_t}' &= F^{(1)}_{e_t} \oplus F^{(2)}_{e_t} \oplus ....F^{(k)}_{e_t} \\ \vec{\mathcal{W}}' &= F^{(1)}_{\mathcal{W}} \oplus F^{(2)}_{\mathcal{W}} \oplus ....F^{(k)}_{\mathcal{W}} \\ \vec{R}' &= R^{(1)} \oplus R^{(2)} \oplus ....R^{(k)} \end{aligned} \quad (10)$$

where in the i^{th} layer: $F^{(i)}_{e_l}$ is the node embedding of e_l, $l = (h, t)$, $F^{(i)}_{\mathcal{W}}$ is the node embedding of sentence $\mathcal{W}$, and $R^{(i)}$ is the readout. In the k^{th} layer, $F^{(k)}$ is the F_{out} from Equation 9. The $\oplus$ is the concatenation among the vectors.

4.3 Context Aggregator

Finally, *KGPool* combines the latent representation (sentential context) with the structured representation learned in Equation 10. As such, we employ a model M which learns latent relation vector $\vec{r'}$. In the state-of-the-art approaches that use KG context, the representation of $\vec{r'}$ is learned using sentential and all static KG context (Vashishth et al., 2018; Bastos et al., 2021). However, in *KGPool*, relation r is realized based on the sentential context and dynamically chosen KG context. Hence, we employ context aggregators similar to the baselines (section 5.3) for jointly learning the enriched KG context in the form of CG and sentential context. The final relation is:

$$P(r \mid e_h, e_t, W) = softmax(MLP(\vec{r'} \oplus \vec{e_h}' \oplus \vec{e_t}' \oplus \vec{W}' \oplus \vec{R}')) \quad (11)$$

5 Experimental Setup

5.1 Datasets

We consider two standard datasets (English version): Wikidata dataset (Sorokin and Gurevych, 2017) and NYT Freebase (Riedel et al., 2010). Both datasets were annotated using distant supervision (associated stats are in Table 1). Datasets include 'NA' as one of the target relations.

Dataset	#Train Sentences	#Test sentences	#Relations
Wikidata	372,059	360,334	353
NYT	455,771	172,448	53

Table 1: Statistics of the Datasets

5.2 KGPool Configurations

KGPool is configured with two context aggregator modules. We inherit context aggregators from existing sentential RE baselines. Our experimental aim is to assess as how *KGPool* performs along with the state-of-the-art context aggregators (comparative study). Our two settings are:

1. $KGPool_{+lstm}$: *KGPool* is coupled with a context aware LSTM model from (Sorokin and Gurevych, 2017) as context aggregator.

2. $KGPool_{+gnn}$: this implementation has *KGPool* plugged-in with a variant of GNN module used by (Zhu et al., 2019; Bastos et al., 2021).

5.3 Baseline Models

We consider the recent sentential RE approaches for our empirical study:

RECON (Bastos et al., 2021): induces KG context (entity attributes and 1&2 hop entity triples) along with the sentence in a GNN.

RECON-EAC (Bastos et al., 2021): a variant of RECON contains entity attributes as only KG context (same context as *KGPool*).

GP-GNN (Zhu et al., 2019): performs multi-hop reasoning using a GNN.

Context-LSTM (Sorokin and Gurevych, 2017): uses context from other sentential relations.

Sorokin-LSTM (Sorokin and Gurevych, 2017): the NYT dataset contains one relation per sentence, but Context-LSTM requires at least two relations in a sentence. Thus, the other setting is an LSTM model without a sentential relation context, is used as a baseline on the NYT dataset.

Multi-instance RE Approaches: Please note, the Wikidata dataset does not have multiple instances for an entity pair. Hence multi-instance baselines do not have values on it. We inherit the recent multi-instance baselines and all empirical values from (Bastos et al., 2021): (i) **HRERE** (Xu and Barbosa, 2019) (ii) **Wu-2019** (Wu et al., 2019), (iii) **Yi-Ling-2019** (Ye and Ling, 2019), (iii) **RESIDE** (Vashishth et al., 2018).

5.4 Metrics and Hyper-parameters

Graph Construction step (section 4.1) use a Bi-LSTM with one hidden layer of size 50 (Bastos et al., 2021). The word embedding dimension is 50 initialized by Glove embeddings (Pennington et al., 2014). The context pooling parameters are from (Lee et al., 2019). For model M, we used the default parameters provided by the authors (Zhu et al., 2019; Sorokin and Gurevych, 2017). For brevity, details are in the appendix.

Metric and Optimization: Our experiment settings are borrowed from (Bastos et al., 2021). Hence, on Wikidata dataset, we use (micro) precision (P), recall (R), and F-score (F1). On the NYT Freebase dataset, (micro) P@10 and P@30 is reported. P@K here represents precision at K percent recall. We also study the effect of Context Coefficient (α) for both *KGPool* configurations (trained end-to-end). We ignore the probability predicted for the NA relation during testing. We employ the Adam optimizer (Kingma and Ba, 2015) with categorical cross entropy loss where each model is run three times on the whole training set. For the P/R curves (with best α values of *KGPool* variants), the result from the first run of each model is selected. For ablation, we use the McNemar's test for statistical significance to find if the reduction in error in the *KGPool* configura-

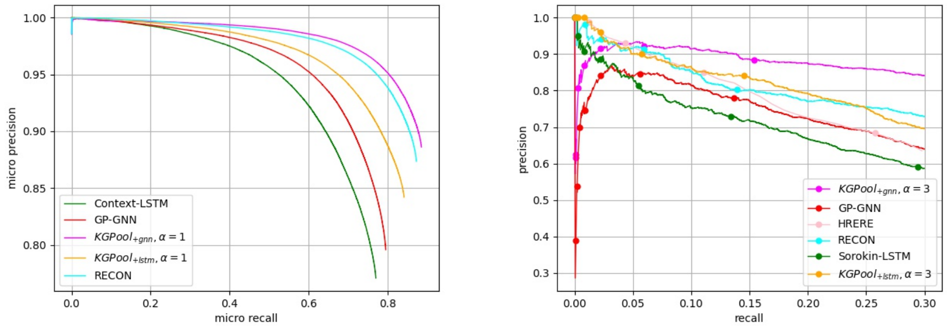

(a) Micro P-R Curve on Wikidata Dataset (b) Micro P-R Curve on NYT-Freebase Dataset

Figure 3: *KGPool*'s best configuration (Tables: 2, 3) perform better than baselines over the entire recall range.

tions are significant. The differences in the models is statistically significant if the $p - value < 0.05$ (Dietterich, 1998). We release all experiment code and data on a public GitHub[2].

6 Results

We conduct our experiments and analysis in response to the question **RQ**: "What is the efficacy of *KGPool* in dynamically selecting the KG context for the sentential RE task?" As such, we also compare *KGPool* against approaches that do not dynamically treat the context.

Model	P	R	F1
Context-LSTM	72.09	72.06	72.07
GP-GNN	82.30	82.28	82.29
RECON-EAC	85.44	85.41	85.42
RECON	87.24	87.23	87.23
$KGPool_{+lstm}$ (α=1)	84.20	82.19	84.20
$KGPool_{+lstm}$ (α=2)	84.12	84.13	84.12
$KGPool_{+lstm}$ (α=3)	84.00	83.97	83.98
$KGPool_{+lstm}$ (α=4)	83.81	83.79	83.80
$KGPool_{+gnn}$ (α=1)	**88.60**	**88.59**	**88.60**
$KGPool_{+gnn}$ (α=2)	88.57	88.56	88.57
$KGPool_{+gnn}$ (α=3)	88.54	88.55	88.54
$KGPool_{+gnn}$ (α=4)	88.52	88.50	88.51

Table 2: Comparison of *KGPool* configurations with sentential RE models on Wikidata dataset. Best score in bold.

Performance on Wikidata Dataset: Table 2 summarizes the performance of *KGPool* variants against the sentential RE models. Agnostic of the underlying aggregator (LSTM or GNN), *KGPool* effectively captures the KG context complimenting the sentential context. The $KGPool_{+gnn}$ (α=1) configuration outperforms other *KGPool* variants along with all sentential RE baselines. We can observe that even when the available context is limited to entity attributes, the $KGPool_{+gnn}$ variant surpasses RECON that also contains context from 1&2 hop triples besides the entity attributes. RECON-EAC and $KGPool_{+gnn}$ rely on entity attributes as KG context with the same context aggregator. When $KGPool_{+gnn}$ variants choose KG context dynamically, they perform better than RECON-EAC. It is interesting to notice that when an LSTM model is fed with the dynamically chosen context, the performance gain is more than ten absolute points ($KGPool_{+lstm}$ Vs Context-LSTM), even outperforming GP-GNN.

Model	P@10	P@30
HRERE	86.1	76.6
Wu-2019	81.7	61.8
RESIDE	73.6	59.5
Ye-Ling-2019	78.9	62.4
Sorokin-LSTM	75.4	58.7
GP-GNN	81.3	63.1
RECON-EAC	83.5	73.4
RECON	87.5	74.1
$KGPool_{+lstm}$ (α=1)	83.7	72.7
$KGPool_{+lstm}$ (α=2)	83.5	71.6
$KGPool_{+lstm}$ (α=3)	84.1	70.6
$KGPool_{+lstm}$ (α=4)	83.1	72.1
$KGPool_{+gnn}$ (α=1)	90.1	**86.7**
$KGPool_{+gnn}$ (α=2)	91.0	85.0
$KGPool_{+gnn}$ (α=3)	**92.3**	85.4
$KGPool_{+gnn}$ (α=4)	90.6	84.4

Table 3: Comparison of *KGPool* with sentential and multi-instance RE models on NYT Freebase dataset. Best score in bold.

Performance on NYT Freebase Dataset: Similar to the Wikidata dataset, the $KGPool_{+gnn}$ variants significantly outperform all baselines (cf. Table 3). The P@30 is comparatively high for $KGPool_{+gnn}$ against baselines. The behavior could be interpreted as follows: dynamically adding context from the KG for the entity pairs

---

[2] `https://github.com/nadgeri14/KGPool`

Compared Models	Contingency table		Statistic	p-value	Significance	Dataset
$KGPool_{+gnn}$ (α=3) vs RECON	160916 3169	4702 3613	298.18	$8.18 * 10^{-67}$	Statistically Significant	NYT- Freebase
$KGPool_{+gnn}$ (α=1) vs RECON	617266 29255	38652 55593	1300.08	$1.08 * 10^{-284}$	Statistically Significant	Wikidata

Table 4: The McNemar's test for statistical significance for KGPool's best configuration Vs previous baseline.

keeps the precision higher over a more extended recall range. For both datasets, *KGPool* configurations ($KGPool_{+gnn}$ and $KGPool_{+lstm}$) have the best-reported performance varying as per the α. This validates our choice to introduce a soft constraint in selecting the context nodes (cf., Equation 8). The P/R curves in Figure 3 show that *KGPool* performs better than baselines over the entire recall range. We conclude that the effective dynamic context selection by *KGPool* has a positive impact on the sentential RE task (which successfully answers our research question).

Models	DEG (HIG)	DEG (CG)	Dataset
$KGPool_{+gnn}$ (α=1)	5.33	1.15	Wikidata
$KGPool_{+gnn}$ (α=2)	5.33	1.52	
$KGPool_{+gnn}$ (α=3)	5.33	2.87	
$KGPool_{+gnn}$ (α=4)	5.33	4.71	
$KGPool_{+gnn}$ (α=1)	6.34	1.67	NYT Freebase
$KGPool_{+gnn}$ (α=2)	6.34	1.91	
$KGPool_{+gnn}$ (α=3)	6.34	2.73	
$KGPool_{+gnn}$ (α=4)	6.34	5.16	

Table 5: Effect of Context Pooling. 'DEG' denotes average degree of an entity node (e_i). 'DEG' of entity nodes in CG is drastically reduced wrt the HIG.

6.1 Ablation Studies

We conducted two ablation studies to understand the behavior of *KGPool* configurations:

Significance of Dynamic Context Selection: we perform McNemar's test for the best *KGPool* configuration against the previous sentential state-of-the-art (i.e. RECON). The results in Table 4 are statistically significant on both datasets, illustrating *KGPool*'s robustness. Although $KGPool_{+gnn}$ variants achieve statistically significant results against RECON, there exist several sentences for which our approach is unable to select supplementary KG context ((RW) values in the contingency table). It requires further investigation, and we plan it for our future work.

Effect on the Degree of Nodes for Entities: for studying the effect of context pooling (Section 4.2), we also conducted a study to understand the impact of *KGPool* on the reduction of the average degree of entity nodes (e_i) in the HIG. Table 5 summarizes the effect of Context Coefficient on the average degree of entity nodes. Irrespective of α, *KGPool* notably reduces the degree of e_i by removing less relevant nodes.

Architectural Choice Experiment: In *KGPool*, we chose to introduce pooling in the last layer of a three-layered architecture (three blocks). To support our choice, we performed several additional experiments by introducing pooling in various layers. We employ the Wikidata dataset for our experiments. We use best configuration of our model ($KGPool_{+gnn}$ (α=1)) and created several variants of it. For instance, $KGPool_{+gnn}$ (P=all) comprises the configuration where we introduce pooling in all three GCN blocks. The configuration $KGPool_{+gnn}$ (¶=2&3) has no pooling in the first layer but has a pooling layer in the remaining two GCN blocks. $KGPool_{+gnn}$ is the best configuration of *KGPool* where pooling is just in the final layer. In Table 6, we observe that $KGPool_{+gnn}$ with pooling only in the last GCN block has the superior performance compared to other two variants. Here, the first two layers are used to learn the node features, which are then employed with self-attention for node selection. Our experiments justify the architectural choice decision. However, with a newer graph pooling technique, such decisions will solely depend on the performance of the approach, and we can not generalize the results of these experiments.

Model	F1
$KGPool_{+gnn}$ (¶=all)	84.19
$KGPool_{+gnn}$ (¶=2&3)	86.87
$KGPool_{+gnn}$ (¶=3) (best)	**88.60**

Table 6: When we introduce pooling in all three layers or in two layers, the performance of *KGPool*'s variants drop. Hence, it justify our choice to add pooling only in the third layer that gives the best performance (values in bold). We use best configuration of our model ($KGPool_{+gnn}$ (α=1)).

Case-Studies: To understand the *KGPool*'s performance gain, we report a few top relations in Table 7. It can be observed from this table that in a few cases, with lesser context, *KGPool* can perform significantly better. In the next case study, to understand the *KGPool*'s performance while adding additional context (more noise), we induce extra context in the form of 1&2-hop triples along with entity attributes. For the same, we considered *KGPool*'s best configurations on the Wikidata dataset. The configurations $KGPool_{+gnn}$ $(+T)$ and $KGPool_{+lstm}$ $(+T)$ represent *KGPool* fed with additional triple context. For both configurations agnostic of underlying aggregator, we observe a slight increase in performance (Table 8). There are several triples which are the irrelevant source of information not needed for a given sentence. *KGPool* can remove that information and does not suffer the performance drop due to added noise in the context. Details on error analysis, performance for worst performing individual relations, and on a human-annotated dataset are provided in the appendix.

Relation	*KGPool*	RECON	GP-GNN
vocal specialization	**1.00**	0.00	0.00
list of works	**1.00**	0.00	0.00
track gauge	**1.00**	0.92	0.00
position played	**0.99**	**0.99**	0.92
sport	**0.99**	**0.99**	0.97
record label	**0.95**	0.90	0.64
list of episodes	**0.95**	0.00	0.49
wing configuration	**0.94**	0.57	0.00
numeric value	**0.93**	0.27	0.46
vessel class	**0.87**	0.00	0.00

Table 7: Micro F-score of Top performing Relations for $KGPool_{gnn}$ (α=1) (on Wikidata dataset). Dynamically chosen context significantly improves performance for many relations.

Model	F1
$KGPool_{+gnn}$ $(+T)$	**88.85**
$KGPool_{+lstm}$ $(+T)$	84.42
$KGPool_{+gnn}$	88.60
$KGPool_{+lstm}$	84.12

Table 8: To scale the sources of the contexts, we induce additional triple context in the *KGPool* shown as $(+T)$ configurations. We use best configurations of our model ($KGPool_{+gnn}$ (α=1) and $KGPool_{+lstm}$ (α=1)). We observe a slight jump in the performance, however, *KGPool* is still able to pool irrelevant context.

7 Discussion and Conclusion

Although KGs are often employed for providing background context in the RE tasks (cf. Section 2), yet there is limited research about defining relevant context. In this work, we proposed *KGPool* and provide a set of experiments proving: 1) Given the limited context that is in individual sentences, dynamically bringing context from KG significantly improves the RE performance. 2) We introduced Context Coefficient (α), which acts as a soft constraint in determining the relevant entity context nodes. 3) Our approach *KGPool* is invariant of the context aggregator and enables us to learn effective knowledge representation of the required KG context for a given sentential context. Our evaluation concerns several key questions:

- **Data quality impact on an effective knowledge representation:** in spite *KGPool*'s significant performance, there exist several sentences for which our model finds a limitation compared to the baseline (cf. Table 4). One potential interpretation could be about the noise injected due to the data quality of the KG context (Weichselbraun et al., 2018). Hence, how does the quality of contextual data impact the performance of context selection approaches is an open direction.

- **Impact of additional sources of KG context:** In ablation, we provide a study by adding 1 & 2-hop triples in addition to entity attributes. There is no significant increase in the performance, although *KGPool* is able to remove irrelevant context for a given sentence. Furthermore, we did not consider edge features in HIG although *KGPool* can be extended to support edge features using techniques such as (Simonovsky and Komodakis, 2017). Additional experiments are needed to verify that our empirical observations hold in this setting, and we leave it for future work.

Overall, *KGPool* provides an effective knowledge representation for set-ups where sentence context is sparse. It is interesting to observe that effective knowledge representation learned using *KGPool* paired with an LSTM model outperforms GP-GNN (Zhu et al., 2019), and nearly all multi-instance baselines. Our conclusive results open a new research direction: *is it possible to apply effective context selection techniques coupled with deep learning models to other downstream NLP tasks?* For example, our results can encourage researchers to extend *KGPool* or develop novel context selection methods for the tasks where KGs have been extensively used as additional background knowledge, such as in entity linking (Mulang' et al., 2020; Mulang et al., 2020), KG completion (Wang et al., 2020; Shi et al., 2017), and recommendation system (Yang et al., 2020).

8 Ethics/ Impact Statement:

In this work, we present significant progress in solving sentential RE task. Harvesting knowledge is an essential goal that human beings seek along with the advancement of technology. This research and many RE approaches rely on additional signals from the public KGs to design systems that extract structured knowledge from unstructured contents. When it comes to who may be disadvantaged from this research, we do not think it is applicable since our study of addressing the KG context capabilities is still at an early stage. Having said so, we are fully supporting the development of ethical and responsible AI. The potential bias in the standard public datasets that may lead to wrong knowledge needs to be cleaned or corrected with validation mechanisms.

References

Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Isaiah Onando Mulang, Saeedeh Shekarpour, Johannes Hoffart, and Manohar Kaul. 2021. Recon: Relation extraction using knowledge graph context in a graph neural network. In *Proceedings of The Web Conference (WWW) (long papers)*, page :N/A.

Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. 2007. Freebase: A Shared Database of Structured General Human Knowledge. In *AAAI*.

Peter Bühlmann and Sara Van De Geer. 2011. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media.

Catalina Cangea, Petar Velickovic, Nikola Jovanovic, Thomas Kipf, and Pietro Liò. 2018. Towards sparse hierarchical graph classifiers. *CoRR*, abs/1811.01287.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *International Conference on Machine Learning*, pages 2083–2092.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3060–3066. AAAI Press.

Xiaomian Kang, Yang Zhao, Jiajun Zhang, and Chengqing Zong. 2020. Dynamic context selection for document-level neural machine translation via reinforcement learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2242–2254. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743.

Liping Liu, Francisco Ruiz, Susan Athey, and David Blei. 2017. Context selection for embedding models. *Advances in Neural Information Processing Systems*, 30:4816–4825.

Isaiah Onando Mulang', Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann. 2020. Evaluating the impact of knowledge graph context on entity disambiguation models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2157–2160.

Isaiah Onando Mulang, Kuldeep Singh, Akhilesh Vyas, Saeedeh Shekarpour, Maria-Esther Vidal, and Soren Auer. 2020. Encoding knowledge graph entity aliases in attentive neural network for wikidata entity linking. In *International Conference on Web Information Systems Engineering*, pages 328–342. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Sungmin Rhee, Seokjun Seo, and Sun Kim. 2018. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3527–3534.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD*, volume 6323 of *Lecture Notes in Computer Science*, pages 148–163. Springer.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Jun Shi, Huan Gao, Guilin Qi, and Zhangquan Zhou. 2017. Knowledge graph embedding with triple context. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2299–2302.

Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 29–38. IEEE Computer Society.

Alisa Smirnova and Philippe Cudré-Mauroux. 2018. Relation extraction using distant supervision: A survey. *ACM Computing Surveys (CSUR)*, 51(5):1–35.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, pages 1784–1789.

YuSheng Su, Xu Han, Zhengyan Zhang, Peng Li, Zhiyuan Liu, Yankai Lin, Jie Zhou, and Maosong Sun. 2020. Contextual knowledge selection and embedding towards enhanced pre-trained language models. *arXiv preprint arXiv:2009.13964*.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha P. Talukdar. 2018. RESIDE: improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1266.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Denny Vrandecic. 2012. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st World Wide Web Conference, WWW 2012 (Companion Volume)*, pages 1063–1064.

Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2020. Entity context and relational paths for knowledge graph completion. *arXiv preprint arXiv:2002.06757*.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307.

Albert Weichselbraun, Philipp Kuntschik, and Adrian MP Braşoveanu. 2018. Mining and leveraging background knowledge for improving named entity linking. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, pages 1–11.

Shanchan Wu, Kai Fan, and Qiong Zhang. 2019. Improving distantly supervised relation extraction with neural noise converter and conditional optimal selector. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 7273–7280. AAAI Press.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

Peng Xu and Denilson Barbosa. 2019. Connecting language and knowledge with heterogeneous representations for neural relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1*, pages 3201–3206.

Susen Yang, Yong Liu, Yonghui Xu, Chunyan Miao, Min Wu, and Juyong Zhang. 2020. Contextualized graph attention network for recommendation with item knowledge graph. *arXiv preprint arXiv:2004.11529*.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Distant supervision relation extraction with intra-bag and inter-bag attentions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*, pages 2810–2819.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pages 4800–4810.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2205–2215. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph neural networks with generated parameters for relation extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, pages 1331–1339.

A Appendix

Due to page limit, we could not put several empirical results in the main paper. This section describes the remaining empirical studies.

A.1 Error Analysis

To understand the failure cases of *KGPool*, we conducted exhaustive error analysis. We calculated (micro) F1-Score of each relation in Sorokin dataset (Sorokin and Gurevych, 2017). Table 10 illustrates performance of ten relations on which *KGPool* performs the worst (ascending order of Micro-F1 score). To put the study in the right perspective, we also report all sentential RE baselines' performance on these relations. While analyzing the errors, we observe three patterns. First, all models fail in the relations for which number of instances are sparse. For example, the relation *mother* has only 190 instances (occurances) and the relation *killed by* has 48 instances. The scarcity in training data has made all models to fail on certain relations. Secondly, our model fails in very closed relations. For example, instead of predicting the relation *drafted by*[3], our model predicts *member of sport team*[4]. Similarly, in case of *unmarried partner*, our model predicts *spouse*. We believe that introducing logical reasoning in the model can help these borderline cases. The third observed pattern for errors is the quality of context. It is worthwhile to mention that in GP-GNN and Context-LSTM, there is only a sentential context. RECON and *KGPool* use KG context. Still, performance is limited for many relations such as *use* and *different from* as reported in the table 10. The lack of quality context in the KG possibly a reason for limited performance for KG-context-induced models in erroneous cases. Detailed exploration is needed to understand the impact of data quality on *KGPool* performance, and we leave it for the future work.

A.2 Effect of Context Pooling

Models	**DEG (HIG)**	**DEG (CG)**	**Dataset**
$KGPool_{+lstm}$ (α=1)	5.33	1.06	Wikidata
$KGPool_{+lstm}$ (α=2)	5.33	2.12	
$KGPool_{+lstm}$ (α=3)	5.33	4.32	
$KGPool_{+lstm}$ (α=4)	5.33	4.81	
$KGPool_{+lstm}$ (α=1)	6.34	1.23	NYT Freebase
$KGPool_{+lstm}$ (α=2)	6.34	1.74	
$KGPool_{+lstm}$ (α=3)	6.34	3.05	
$KGPool_{+lstm}$ (α=4)	6.34	6.30	

Table 9: Effect of Context Pooling. 'DEG' denotes average degree of an entity node (e_i). We observe a reduction in the degree of entity nodes in CG compared to the HIG.

In the main paper, we presented the effect of context pooling on *KGPool*'s best configuration ($KGPool_{+gnn}$). Table 9 describes the reduction in the average degree of nodes for $KGPool_{+lstm}$ configuration for various context coefficient (α). On both datasets, there is a significant reduction in the degree of nodes. On Wikidata dataset (Sorokin and Gurevych, 2017), $KGPool_{+lstm}$ with (α=1) reports the highest value among its other configurations. For the same, the average degree of nodes is reduced from 5.33 to 1.06. Please note, the degree of nodes in HIG remains the same. However, for CG, the degree of nodes differs based on the context aggregator. We train the model end to end, and due to back-propagation, context weights adjust as per the context aggregator.

A.3 Results on a Human Annotated Dataset

The employed datasets Wikidata (Sorokin and Gurevych, 2017) and NYT Freebase (Riedel et al., 2010) are created using distant supervision techniques. Considering distant supervision techniques inherit a noise, to provide a comprehensive ablation study, (Zhu et al., 2019) provided a human evaluation setting. Following the same setting, RECON provided human-annotated data

---

[3] `https://www.wikidata.org/wiki/Property:P647`

[4] `https://www.wikidata.org/wiki/Property:P54`

Relation	***KGPool***	**RECON**	**GP-GNN**	**Context-LSTM**
has quality	0.00	0.00	0.00	0.00
enclave within	0.00	0.00	0.00	0.00
drafted by	0.01	0.08	0.00	0.02
different from	0.01	0.00	0.0	0.00
mother	0.03	0.05	0.02	0.00
unmarried partner	0.04	0.01	0.00	0.00
killed by	0.04	0.01	0.00	0.04
use	0.09	0.00	0.00	0.00
lyrics by	0.10	0.13	0.00	0.00
relative	0.12	0.10	0.00	0.00

Table 10: Micro F-score of 10-worst performing Relations for $KGPool_{gnn}$ (α=1) on Wikidata dataset. We also provide corresponding values of other sentential RE baselines. The main reason for limited performance across all models is the scarcity of training data for these relation types.

from Wikidata dataset (Sorokin and Gurevych, 2017). This is to verify that the distantly supervised dataset is correct for every pair of entities. Sentences accepted by all annotators are part of the human-annotated dataset. There are 500 sentences and 1846 triples in the test set. Table 11 reports $KGPool$'s performance against the sentential baselines. $KGPool_{+gnn}$ continues to outperform the baselines, maintaining similar behavior as seen on test sets of original datasets. The results further re-assure the robustness of our proposed approach.

Model	**P**	**R**	**F1**
Context Aware LSTM	77.77	78.69	78.23
GP-GNN	81.99	82.31	82.15
RECON-EAC	86.10	86.58	86.33
RECON	87.34	87.55	87.44
$KGPool_{+lstm}$ (α=1)	86.34	86.07	86.20
$KGPool_{+gnn}$ (α=1)	**89.36**	**89.31**	**89.33**

Table 11: Sentential RE performance on Human Annotation Dataset. *KGPool* again outperforms the baselines. We report Micro P,R, and F1 values. (Best score in bold)

Hyperparameters	**Value**
learning rate	0.001
batch size	50
hidden state size	128
context coefficient (α)	1,2,3,4
# of propagation layers	3

Table 12: Hyper-parameters for Context Pooling module

Hyperparameters	**Value**
learning rate	0.001
batch size	50
dropout ratio	0.5
hidden state size	256
non-linear activation	relu
# of propagation layers	3
entity embedding size	8
adjacent matrices	untied
optimizer	adam
β_1	0.9
β_2	0.999
ε	1e-08
pretrained embeddings	glove
word embedding dim	50

Table 13: Hyper-parameters for GNN-Aggregator module

Hyperparameters	**Value**
learning rate	0.001
batch size	50
dropout ratio	0.5
hidden state size	256
non-linear activation	relu
# of layers	1
optimizer	adam
pretrained embeddings	glove
word embedding dim	50

Table 14: Hyper-parameters for ContextAware-Aggregator module

Hyperparameters	**Value**
learning rate	0.001
batch size	50
initial embedding size	50
final embedding size	50
pretrained embeddings	glove
# of layers	1

Table 15: Hyper-parameters for Graph Construction module

A.4 Datasets and Hyper-parameters

We augmented two datasets Wikidata dataset and Riedel Freebase dataset with our proposed KG context. The Wikidata dataset has 353 unique relations, 372,059 sentences in training, 123824 sentences in validation and 360,334 for testing. The number of sentences in the training and test set are 455,771 and 172,448 respectively in the Riedel dataset. No explicit validation set has been provided for Riedel dataset. For augmenting entity attribute context, we relied on public dumps of Wikidata and Freebase. From these dumps, we automatically extracted entities and its properties: labels, aliases, instance of, descriptions. For Wikidata, we used public API[5] using a SPARQL query and for Freebase, we took original depreciated dump[6].

We use the nltk english tokenizer for splitting the sentence into its corresponding tokens in the Riedel dataset. We do not do any further data preprocessing. We used 1 GPU NVIDIA TITAN X Pascal with 12GB of GPU storage to run our experiments. We train the models upto a maximum of 14 epochs and select the best performing model based on the micro F1 scores of the validation set. The tables 14, 15 and 12 detail the hyper-parameter settings used in our experiments. We do not do any further hyper-parameter tuning.

---

[5] `https://query.wikidata.org/`

[6] `https://developers.google.com/freebase`

Better Combine Them Together! Integrating Syntactic Constituency and Dependency Representations for Semantic Role Labeling

Hao Fei[1] and **Shengqiong Wu**[1] and **Yafeng Ren**[2] and **Fei Li**[1] and **Donghong Ji**[1*]
1. Department of Key Laboratory of Aerospace Information Security and Trusted Computing,
Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
2. Laboratory of Language and Artificial Intelligence,
Guangdong University of Foreign Studies, China
{hao.fei,whuwsq,renyafeng,lifei_csnlp,dhji}@whu.edu.cn

Abstract

Structural syntax knowledge has been proven effective for semantic role labeling (SRL), while existing works mostly use only one singleton syntax, such as either syntactic dependency or constituency tree. In this paper, we explore the integration of heterogeneous syntactic representations for SRL. We first consider a TreeLSTM-based integration, collaboratively learning the phrasal boundaries from the constituency and the semantic relations from dependency. We further introduce a label-aware GCN solution for simultaneously modeling the syntactic edges and labels. Experimental results demonstrate that by effectively combining the heterogeneous syntactic representations, our methods yield task improvements on both span-based and dependency-based SRL. Also our system achieves new state-of-the-art SRL performances, meanwhile bringing explainable task improvements.

1 Introduction

Semantic role labeling (SRL) aims to disclose the predicate-argument structure of a given sentence. Such shallow semantic structures have been shown highly useful for a wide range of downstream tasks in natural language processing (NLP), such as information extraction (Fader et al., 2011; Bastianelli et al., 2013), machine translation (Xiong et al., 2012; Shi et al., 2016) and question answering (Maqsud et al., 2014; Xu et al., 2020). Based on whether to recognize the constituent phrasal span or the syntactic dependency head token of an argument, prior works categorize SRL into two types: the span-based SRL popularized in CoNLL05/12 shared tasks (Carreras and Màrquez, 2005; Pradhan et al., 2013), and the dependency-based SRL introduced in CoNLL08/09 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009). By adopting various neural network methods, two types of SRL have achieved significant performances in recent years (FitzGerald et al., 2015; He et al., 2017; Fei et al., 2021a)

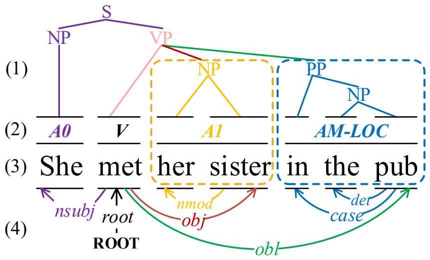

Figure 1: The mutual benefit to integrate both the (1) syntactic constituency and (4) dependency structures for (2) SRL, based on (3) an example sentence.

Syntactic features have been extensively verified to be highly effective for SRL (Pradhan et al., 2005; Punyakanok et al., 2008; Marcheggiani and Titov, 2017; Strubell et al., 2018; Zhang et al., 2019). In particular, syntactic dependency features have gained a majority of attention, especially for the dependency-based SRL, considering their close relevance with the dependency structure (Roth and Lapata, 2016; He et al., 2018; Xia et al., 2019; Fei et al., 2021b). Most existing works focus on designing various methods for modeling the dependency representations into the SRL learning, such as TreeLSTM (Li et al., 2018; Xia et al., 2019) and graph convolutional networks (GCN) (Marcheggiani and Titov, 2017; Li et al., 2018). On the other hand, some efforts try to encode the constituency representations for facilitating the span-based SRL (Wang et al., 2019; Marcheggiani and Titov, 2020).

Yet almost all the syntax-based SRL methods use one standalone syntactic tree, i.e., either dependency or constituency tree. Constituent and dependency syntax actually depict the syntactic structure from different perspectives, and integrat-

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 549–559
August 1–6, 2021. ©2021 Association for Computational Linguistics

ing these two heterogeneous representations can intuitively bring complementary advantages (Farkas et al., 2011; Yoshikawa et al., 2017; Zhou and Zhao, 2019). As exemplified in Figure 1, the dependency edges represent the inter-relations between arguments and predicates, while the constituency structure[1] locates more about phrase boundaries of argument spans, and then directs the paths to the predicate globally. Interacting these two structures can better guide the system to focus on the most proper granularity of phrasal spans (as circled by the dotted box), while also ensuring the route consistency between predicate-argument pairs. Unfortunately, we find that there are very limited explorations of the heterogeneous syntax integration in SRL. For instance, Li et al. (2010) manually craft two types of discrete syntax features for statistical model, and recently Fei et al. (2020a) implicitly distill two heterogeneous syntactic representations into one unified neural model.

In this paper, we present two innovative neural methods for explicitly integrating two kinds of syntactic features for SRL. As shown in Figure 2, in our framework, the syntactic constituent and dependency encoders are built jointly as a unified block (i.e., Heterogeneous Syntax Fuser, namely `HeSyFu`), and work closely with each other. In the first architecture of `HeSyFu` (cf. Figure 3), we take two separate TreeLSTMs as the structure encoders for two syntactic trees. Based on our framework, we try to answer the following questions:

► **Q1**. *Whether the combination of constituent and dependency syntax can really improve SRL*?

► **Q2**. If yes, *how much will such improvements be for the dependency- and span-based SRL*?

We further propose *Const* GCN and *Dep* GCN encoders to enhance the syntax encoding in `HeSyFu`, where the *syntactic labels* (i.e., dependent arc types and constituency node types) are modeled in a unified manner within the label-aware GCN, as illustrated in Figure 4. With this, we can dig deeper:

► **Q3**. *How different will the results be by employing the TreeLSTM or GCN encoder*?

► **Q4**. *Can SRL be further improved by leveraging syntactic labels?*

► **Q5**. *What kind of associations can be discovered between SRL structures and these heterogeneous syntactic structures*?

To find the answers, we conduct extensive experiments on both span- and dependency-based SRL benchmarks (i.e., CoNLL05/12 and CoNLL09). The results and analyses show that,

►**A1**. *combining two types of syntax information is more helpful than just using either one of them*;

►**A2**. *the improvement for span-based SRL is more obvious than dependency-based one*;

►**A3**. *GCN performs better than TreeLSTM*;

►**A4**. *syntactic labels are quite helpful for SRL*;

►**A5**. *SRL and both kinds of syntactic structures have strong associations and should be exploited for mutual benefits*.

In our experiments, our SRL framework with two proposed `HeSyFu` encoders achieves better results than current best-performing systems, and yield more explainable task improvements.

2 Related Work

The SRL task, uncovering the shallow semantic structure (i.e. '*who did what to whom where and when*') is pioneered by Gildea and Jurafsky (2000), and popularized from PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998). SRL is typically divided into the span-based one and dependency-based one on the basis of the granularity of arguments (e.g., phrasal spans or dependency heads). Earlier efforts focus on designing hand-crafted features with machine learning methods (Pradhan et al., 2005; Punyakanok et al., 2008; Zhao et al., 2009b,a). Later, SRL works mostly employ neural networks with distributed features for the task improvements (FitzGerald et al., 2015; Roth and Lapata, 2016; Marcheggiani and Titov, 2017; Strubell et al., 2018). Most high-performing systems model the task as a sequence labeling problem with *BIO* tagging scheme for both two types of SRL (He et al., 2017; Ouchi et al., 2018; Fei et al., 2020c,b).

On the other hand, syntactic features are a highly effective SRL performance enhancer, according to numbers of empirical verification in prior works (Marcheggiani et al., 2017; He et al., 2018; Swayamdipta et al., 2018; Zhang et al., 2019), as intuitively SRL shares much underlying structure with syntax. Basically, the syntactic dependent feature is more preferred to be injected into the dependency-based SRL (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; He et al., 2018; Kasai et al., 2019), while other consider the constituent syntax for the span-based SRL (Wang

---

[1] Following Marcheggiani and Titov (2020), we strip off the nodes of POS tags from the constituency tree for brevity.

et al., 2019; Marcheggiani and Titov, 2020).

Actually, the constituent and dependency syntax depict the structural features from different angles, while they can share close linguistic relevance. Related works have revealed the mutual benefits on integrating these two heterogeneous syntactic representations for various NLP tasks (Collins, 1997; Charniak, 2000; Charniak and Johnson, 2005; Farkas et al., 2011; Yoshikawa et al., 2017; Zhou and Zhao, 2019; Strzyz et al., 2019; Kato and Matsubara, 2019). Unfortunately, there are very limited explorations for SRL. For example, Li et al. (2010) construct discrete heterogeneous syntactic features for SRL. More recent work in Fei et al. (2020a) leverage knowledge distillation method to inject the heterogeneous syntax representations from various tree encoders into one model for enhancing the span-based SRL. In this work, we consider an explicit integration of these two syntactic structures via two neural solutions. To our knowledge, we are the first attempt performing thorough investigations on the impacts of the heterogeneous syntax combination to the SRL task.

Various neural models have been proposed for encoding the syntactic structures, such as attention mechanism (Strubell et al., 2018; Zhang et al., 2019), TreeLSTM (Li et al., 2018; Xia et al., 2019), GCN (Marcheggiani and Titov, 2017; Li et al., 2018; Marcheggiani and Titov, 2020), etc. In this work, we take the advantages of the TreeLSTM and GCN models for encoding the constituent and dependency trees, as two solutions of our `HeSyFu` encoders. It is worth noticing that prior works using GCN to encode dependency (Marcheggiani and Titov, 2017) and constituent (Marcheggiani and Titov, 2020), where however the syntactic labels are not managed in a unified manner. We thus consider enhancing the syntax GCN by simultaneously modeling the syntactic labels within the structure.

3 SRL Model

3.1 Task Modeling

Following prior works (Tan et al., 2018; Marcheggiani and Titov, 2020), our system aims to identify and classify the arguments of a predicate into semantic roles, such as *A0*, *A1*, *AM-LOC*, etc. We denote the complete role set as $\mathcal{R}$. We adopt the *BIO* tagging scheme. And given a sentence s=$\{w_1,\cdots,w_n\}$ and a predicate w_p, the model assigns each word w_i a label $\hat{y} \in \mathcal{Y}$, where $\mathcal{Y}$=$(\{B,I\}\times\mathcal{R}) \cup \{O\}$.[2] Note that each semantic argument corresponds to a word span of $\{w_j,\cdots,w_k\}$ $(1\leq j\leq k\leq n)$.[3]

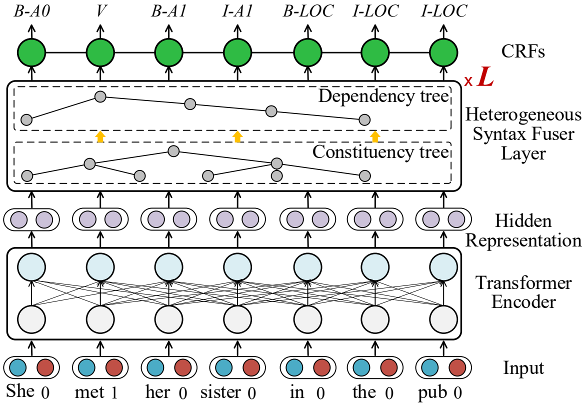

Figure 2: Overview of our SRL framework.

3.2 Framework

As illustrated in Figure 2, our SRL framework consists of four components, including input representations, Transformer encoder, heterogeneous syntax fuser layer and CRFs decoding layer.

Given an input sentence s and a predicate word w_p (p is the position), the input representations $\boldsymbol{x}_i$ are the concatenation ($\oplus$) of word embeddings $\boldsymbol{x}_{w_i}$ and predicate binary embeddings $\boldsymbol{x}_{(i==p)}$ indicating the presence or absence of w_p:

$$\boldsymbol{x}_i = \boldsymbol{x}_{w_i} \oplus \boldsymbol{x}_{(i==p)} . \tag{1}$$

Afterwards, we adopt Transformer (Vaswani et al., 2017) as our base encoder for yielding contextualized word representations. Transformer (*Trm*) works with multi-head self-attention mechanism:

$$\text{Softmax}(\frac{\boldsymbol{Q}\cdot\boldsymbol{K}^{\text{T}}}{\sqrt{d_k}})\cdot\boldsymbol{V} , \tag{2}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are the linear projections from the input representation $\boldsymbol{x}_i$. We simplify the flow:

$$\{\boldsymbol{r}_1,\cdots,\boldsymbol{r}_n\} = \text{Trm}(\{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n\}) . \tag{3}$$

Next, based on the hidden representation $\boldsymbol{r}_i$, our heterogeneous syntax fuser (`HeSyFu`) layer, which will be elaborated in Section §4, integrates the constituency and dependency syntax, and yields the syntax-aware hidden representation:

$$\{\boldsymbol{s}_1,\cdots,\boldsymbol{s}_n\} = \text{HeSyFu}(\{\boldsymbol{r}_1,\cdots,\boldsymbol{r}_n\}) . \tag{4}$$

---

[2] This work focuses on the pipeline-style SRL which is under the assumption that predicates are pre-identified.

[3] When $j=k$, the span-based SRL model shifts into the dependency-based one.

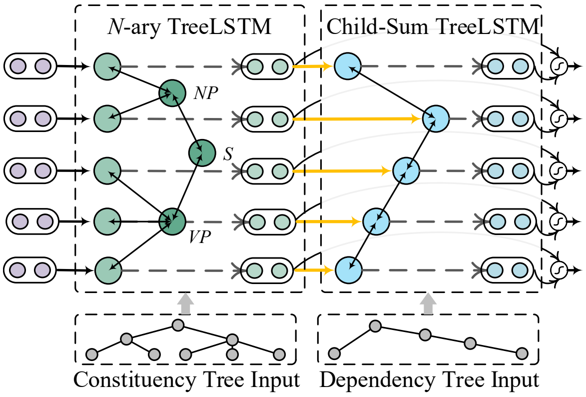

Figure 3: TreeLSTM-based `HeSyFu` layer.

Based on the syntax-aware hidden representation $\boldsymbol{s}_i$, we use CRFs (Lafferty et al., 2001) to compute the probability of each candidate output $\boldsymbol{y} = \{y_1, \cdots, y_n\}$:

$$p(y|s) = \frac{\exp\{\sum_i (\boldsymbol{W}\boldsymbol{s}_n + \boldsymbol{T}_{y_{i-1},y_i})\}}{Z}, \quad (5)$$

where $\boldsymbol{W}$ and $\boldsymbol{T}$ are the parameters and Z is a normalization factor. The Viterbi algorithm is used to search for the highest-scoring tag sequence $\hat{\boldsymbol{y}}$.

4 Integration of Syntactic Constituency and Dependency Structure

We present two neural heterogeneous syntax fusers (a.k.a., `HeSyFu`), including a TreeLSTM-based `HeSyFu` (cf. Figure 3), and a label-aware GCN-based `HeSyFu` (cf. Figure 4). `HeSyFu` is stacked with total L layers for a full syntax interaction. We design the architecture with the constituency (denoted as *const.*) encoding in front of the dependency (denoted as *dep.*) encoding, based on the intuition that the boundary recognition helped by *const.* syntax should go before the semantic relation determination aided by *dep.* syntax.

4.1 TreeLSTM Heterogeneous Syntax Fuser

Our TreeLSTM-based `HeSyFu` (Tr-`HeSyFu`) is comprised of the N-ary TreeLSTM for *const.* trees and the Child-Sum TreeLSTM for *dep.* trees motivated by Tai et al. (2015).

Constituency tree encoding The flow in TreeLSTM is bidirectional, i.e., bottom-up and top-down, for a full information interaction. For each node u in the tree, we denote the hidden state and memory cell of its v-th ($v \in [1, M]$) branching child as $\boldsymbol{h}_{uv}^{\uparrow}$ and $\boldsymbol{c}_{uv}$. The bottom-up one computes the representation $\boldsymbol{h}_u^{\uparrow}$ from its children hierarchically:

$$\begin{aligned}
\boldsymbol{i}_u &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}_v^{(i)}\boldsymbol{h}_{uv}^{\uparrow} + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_{uk} &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}_{kv}^{(f)}\boldsymbol{h}_{uv}^{\uparrow} + \boldsymbol{b}^{(f)}),\\
\boldsymbol{o}_u &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}_v^{(o)}\boldsymbol{h}_{uv}^{\uparrow} + \boldsymbol{b}^{(o)}),\\
\boldsymbol{u}_u &= \mathrm{Tanh}(\boldsymbol{W}^{(u)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}_v^{(u)}\boldsymbol{h}_{uv}^{\uparrow} + \boldsymbol{b}^{(u)}),\\
\boldsymbol{c}_u &= \boldsymbol{i}_u \odot \boldsymbol{u}_u + \textstyle\sum_{k=1}^{M}\boldsymbol{f}_{uk} \odot \boldsymbol{c}_{uk},\\
\boldsymbol{h}_u^{\uparrow} &= \boldsymbol{o}_u \odot \tanh(\boldsymbol{c}_u),
\end{aligned} \quad (6)$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{b}$ are parameters. $\boldsymbol{r}_u, \boldsymbol{i}_u$, $\boldsymbol{o}_u$ and $\boldsymbol{f}_{uv}$ are the input token representation, input gate, output gate and forget gate. Analogously, the top-down N-ary TreeLSTM calculates the representation $\boldsymbol{h}_u^{\downarrow}$ the same way. We concatenate the representations of two directions: $\boldsymbol{h}_u^{const} = \boldsymbol{h}_u^{\uparrow} \oplus \boldsymbol{h}_u^{\downarrow}$. Note that the constituent tree nodes include terminal word nodes and non-terminal constituent nodes, and we only take the representations (i.e., $\boldsymbol{h}_i^{const}$) corresponding to the word node w_i for any usage.

Dependency tree encoding Slightly different from N-ary TreeLSTM for *const.* tree, the non-terminal nodes in *dep.* tree encoded by Child-Sum TreeLSTM are all the word nodes. We also consider the bidirectional calculation here. The bottom-up TreeLSTM obtains $\boldsymbol{h}_i^{\uparrow}$ of the word w_i via:

$$\begin{aligned}
\overline{\boldsymbol{h}}_i^{\uparrow} &= \textstyle\sum_{j\in C(i)}\boldsymbol{h}_j^{\uparrow},\\
\boldsymbol{i}_i &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{r}_i^{'} + \boldsymbol{U}^{(i)}\overline{\boldsymbol{h}}_i^{\uparrow} + b^{(i)}),\\
\boldsymbol{f}_{ij} &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{r}_i^{'} + \boldsymbol{U}^{(f)}\overline{\boldsymbol{h}}_j^{\uparrow} + b^{(f)}),\\
\boldsymbol{o}_i &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{r}_i^{'} + \boldsymbol{U}^{(o)}\overline{\boldsymbol{h}}_i^{\uparrow} + b^{(o)}),\\
\boldsymbol{u}_i &= \mathrm{Tanh}(\boldsymbol{W}^{(u)}\boldsymbol{r}_i^{'} + \boldsymbol{U}^{(u)}\overline{\boldsymbol{h}}_i^{\uparrow} + b^{(u)}),\\
\boldsymbol{c}_i &= \boldsymbol{i}_i \odot \boldsymbol{u}_i + \textstyle\sum_{j\in C(i)}\boldsymbol{f}_{ij} \odot \boldsymbol{c}_j,\\
\boldsymbol{h}_i^{\uparrow} &= \boldsymbol{o}_i \odot \tanh(\boldsymbol{c}_i),
\end{aligned} \quad (7)$$

where $C(i)$ is the set of child nodes of w_i. $\boldsymbol{r}_i^{'}$ is the input token representation consulting the foregoing constituent output representation: $\boldsymbol{r}_i^{'} = \boldsymbol{r}_i + \boldsymbol{h}_i^{const}$. The top-down one yields $\boldsymbol{h}_i^{\downarrow}$, which is concatenated with the bottom-up one: $\boldsymbol{h}_i^{dep} = \boldsymbol{h}_i^{\uparrow} \oplus \boldsymbol{h}_i^{\downarrow}$.

Integration To fully make use of the heterogeneous syntactic knowledge, we fuse these two resulting syntactic representations. We apply a fusion gate to flexibly coordinate their contributions:

$$\begin{aligned}
\boldsymbol{g}_i &= \sigma(\boldsymbol{W}^{(g_1)}\boldsymbol{h}_i^{const} + \boldsymbol{W}^{(g_2)}\boldsymbol{h}_i^{dep} + b^{(g)}),\\
\boldsymbol{s}_i &= \boldsymbol{g}_i \odot \boldsymbol{h}_i^{const} + (1 - \boldsymbol{g}_i) \odot \boldsymbol{h}_i^{dep}.
\end{aligned} \quad (8)$$

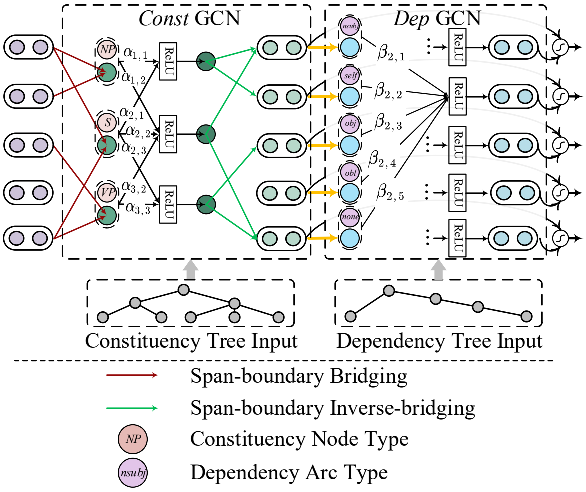

Figure 4: Label-aware GCN-based `HeSyFu` layer.

4.2 Label-aware GCN-based Heterogeneous Syntax Fuser

Compared with TreeLSTM, GCN is more computationally efficient on performing the structural propagation among nodes, i.e., with O(1) complexity. On the other hand, it is also crucial to leverage the syntactic labels (i.e., dependent arc types, and constituent phrasal types) into the SRL learning. For example, within the dependency tree, the information from the neighboring nodes under distinct types of arcs can contribute in different degrees. However we note that current popular syntax GCNs (Marcheggiani and Titov, 2017, 2020) do not encode the dependent or constituent labels with the nodes in a unified manner, which could be inaccurate to describe the syntactic connecting attributes between the neighbor nodes. Based on their syntax GCNs, we newly propose label-aware constituency and dependency GCNs which are able to explicitly formalize the structure edges with syntactic labels simultaneously, and normalize them unitedly.[4] As illustrated in Figure 4, our label-aware GCN-based `HeSyFu` (denoted as LG-`HeSyFu`) has a similar assembling architecture to TreeLSTM-based `HeSyFu`, and will finally be navigated via the gate mechanism as in Eq. (8).

Constituency tree encoding The constituent tree is modeled as a graph $G^{(c)}$=$(U^{(c)}, E^{(c)})$, where $U^{(c)}$ is the node set and $E^{(c)}$ is the edge set. We denote $e^{(c)}_{uv}$=1 if there is an edge between node u and node v, and e_{uv}=0 vice versa. We enable the edges to be bidirectional. μ_u represents the constituent label of node u, such as *S*, *NP* and *VP*, etc. We take the vectorial embedding $\boldsymbol{v}^{(c)}_u$ for the node label μ_u. Our constituent GCN (denoted as *Const* GCN) yields the node representations $\boldsymbol{h}^{(c)}_u$:

$$\boldsymbol{h}^{(c)}_u = \text{ReLU}\{\textstyle\sum_{v=1}^{M}\alpha_{uv}(\boldsymbol{W}^{(c_1)}\cdot\boldsymbol{r}^b_v+\boldsymbol{W}^{(c_2)}\cdot\boldsymbol{v}^{(c)}_v+b^{(c)})\}\,, \tag{9}$$

where $\boldsymbol{r}^b_v$ is the initial node representation of the node v via **span-boundary bridging operation**, i.e., adding the start and end token representation of the phrasal span, $\boldsymbol{r}^b_v = \boldsymbol{r}_{start} + \boldsymbol{r}_{end}$. And α_{uv} is the constituent connecting distribution:

$$\alpha_{uv} = \frac{e^{(c)}_{uv}\cdot\exp\{(\boldsymbol{z}^{(c)}_u)^T\cdot\boldsymbol{z}^{(c)}_v\}}{\sum_{v'=1}^{M} e^{(c)}_{uv'}\cdot\exp\{(\boldsymbol{z}^{(c)}_u)^T\cdot\boldsymbol{z}^{(c)}_{v'}\}}\,, \tag{10}$$

where $\boldsymbol{z}^{(c)}_u = \boldsymbol{v}_u + \boldsymbol{v}^{(c)}_u$. This distribution α_{uv} encodes both the syntactic edge and label information, and thus comprehensively reflects the connecting strengths between neighbors. We then perform **span-boundary inverse-bridging** to restore the token node representation $\boldsymbol{h}^{const}_i$ for each word w_i, i.e., $\boldsymbol{h}^{const}_i = \boldsymbol{h}^{(c)}_{u'} + \boldsymbol{h}^{(c)}_{v'}$.

Dependency tree encoding Likewise, the dependent tree is modeled as a graph $G^{(d)}$=$(U^{(d)}, E^{(d)})$. $e^{(d)}_{ij}$=1/0 denotes the dependency arc existence. $\pi^{\leftrightarrow}_{ij}$ represents the edge label between w_i and w_j, which is also bidirectional. Besides of the pre-defined dependency labels, we additionally add a '*self*' label as the self-loop edge $\pi^{\leftrightarrow}_{ii}$, and a '*none*' label representing no edge between w_i and w_j. We use the embedding form $\boldsymbol{v}^{(d)}_{ij}$ for $\pi^{\leftrightarrow}_{ij}$. The update in dependent GCN (denoted as *Dep* GCN) is written as:

$$\boldsymbol{h}^{(d)}_i = \text{ReLU}(\textstyle\sum_{j=1}^{n}\beta_{ij}(\boldsymbol{W}^{(d_1)}\cdot\boldsymbol{r}^{'}_j+\boldsymbol{W}^{(d_2)}\cdot\boldsymbol{v}^{(d)}_{ij}+b^{(d)}))\,, \tag{11}$$

where $\boldsymbol{r}^{'}_j = \boldsymbol{r}_j + \boldsymbol{h}^{const}_i$. β_{ij} is the neighbor connecting-strength distribution:

$$\beta_{ij} = \frac{e^{(d)}_{ij}\cdot\exp\{(\boldsymbol{z}^{(d)}_i)^T\cdot\boldsymbol{z}^{(d)}_j\}}{\sum_{j'=1}^{n} e^{(d)}_{ij'}\cdot\exp\{(\boldsymbol{z}^{(d)}_i)^T\cdot\boldsymbol{z}^{(d)}_{j'}\}}\,, \tag{12}$$

where $\boldsymbol{z}^{(d)}_i = \boldsymbol{r}^{'}_i + \boldsymbol{v}^{(d)}_{ij}$. Here $\boldsymbol{h}^{(d)}_i$ also can be denoted as $\boldsymbol{h}^{dep}_i$, which navigates the dependent arc and label information in a more unified way.

5 Experiments

5.1 Setups

We conduct experiments on the span-based SRL datasets (CoNLL05 & CoNLL12), and

---

[4] We note that the constituent labels are attached onto nodes, while dependent labels are attached onto edges.

	CoNLL05 WSJ			CoNLL05 Brown			CoNLL12 OntoNotes		
	P	R	F1	P	R	F1	P	R	F1
Without Syntax									
He et al. (2017)	85.00	84.30	84.60	74.90	72.40	73.60	83.50	83.30	83.40
Tan et al. (2018)	84.50	85.20	84.80	73.50	74.60	74.10	81.90	83.60	82.70
Li et al. (2020a)+RoBERTa	88.05	88.00	88.03	80.04	79.56	79.80	86.40	86.83	86.61
Trm+RoBERTa†	87.41	87.72	87.60	79.78	79.86	79.82	86.28	86.67	86.40
With Dependency Syntax									
Strubell et al. (2018)	84.70	84.24	84.47	73.89	72.39	73.13	83.30	81.38	82.33
Xia et al. (2020)	85.12	85.00	85.06	76.30	75.42	75.86	-	-	-
Child-Sum TreeLSTM†	84.94	85.80	85.40	74.60	74.10	74.36	83.42	83.56	83.47
Dep GCN†	86.03	86.52	86.22	75.38	75.89	75.62	84.32	84.88	84.61
Dep GCN+RoBERTa†	88.21	87.82	88.07	80.73	79.82	80.13	86.58	86.99	86.82
With Constituency Syntax									
Wang et al. (2019)*	85.40	85.02	85.23	75.48	75.23	75.36	84.35	84.11	84.21
Marcheggiani and Titov (2020)	85.80	85.10	85.40	76.20	74.70	75.50	84.50	84.30	84.40
Marcheggiani and Titov (2020)+RoBERTa	87.70	88.10	87.90	80.50	80.70	80.60	86.50	87.10	86.80
N-ary TreeLSTM†	85.91	85.27	85.58	75.22	75.06	75.12	84.12	83.85	84.02
Const GCN†	86.68	86.38	86.52	76.54	76.21	76.36	85.51	84.96	85.25
Const GCN+RoBERTa†	88.71	88.94	88.81	81.52	81.05	81.27	87.33	87.42	87.35
With Dependency & Constituency Syntax									
Fei et al. (2020a)*	86.82	86.50	86.72	76.67	76.35	76.48	85.86	85.30	85.50
Tr-`HeSyFu`†	86.27	86.52	86.64	76.95	76.50	76.87	85.91	85.48	85.66
LG-`HeSyFu`†	87.16	87.63	87.32	78.72	77.35	78.12	86.51	85.92	86.20
LG-`HeSyFu` w/o Syn.Label†	86.93	87.21	86.98	77.61	76.85	77.48	85.93	85.68	85.79
LG-`HeSyFu`+RoBERTa†	**88.86**	**89.28**	**89.04**	**83.52**	**83.75**	**83.67**	**88.09**	**88.83**	**88.59**

Table 1: Results on span-based SRL datasets. Values with $*$ are from our re-implementations, while others are retrieved from the raw papers. Scores with † are presented after significant test ($p \leq 0.05$).

dependency-based SRL dataset (CoNLL09). Each dataset has its own training, development, and test sets. We convert the constituency syntax annotations in CoNLL05&12 into dependency annotations by following the standard of Stanford Typed Dependency (v3.3.0).[5] We obtain the constituency annotations for CoNLL09 from the PTB data. We adopt the CoNLL05 evaluation scripts[6] to evaluate the performances, with precision (P), recall (R) and F1 score as the metrics. We conduct significance tests via Dan Bikel's evaluation comparer.[7] The Transformer hidden size is 768. The hidden sizes in TreeLSTM and GCN encoders are in [250,300,350]. We adopt the Adam optimizer with an initial learning rate of 2e-5. We train the model[8] by mini-batch size in [16,24,32] with early-stop strategy. We also load the pre-trained parameters[9] from the RoBERTa language model (Liu et al., 2019) to our Transformer encoder for boosting the performance. The environment is with Intel i9 CPU and NVIDIA RTX 3090Ti GPU.

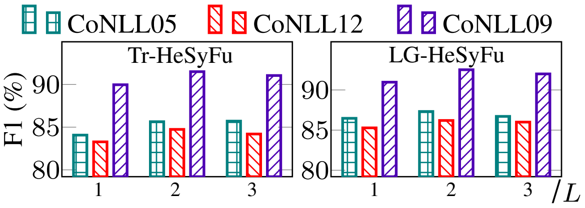

Figure 5: `HeSyFu` with different layers.

5.2 Development Experiments

We first perform preliminary experiments based on the development sets.

Layer of syntax encoder From Figure 5 we see that either too larger or fewer layers of `HeSyFu` does no benefits to the overall performances. When L=2 for Tr-/LG-`HeSyFu`, the performances be-

---

[5] `https://nlp.stanford.edu/software/lex-parser.html`

[6] `http://www.cs.upc.edu/~srlconll/st05/st05.html`

[7] `http://www.cis.upenn.edu/~dbikel/software.html#comparator`

[8] Codes: `https://github.com/scofield7419/HeSyFu`

[9] `https://github.com/pytorch/fairseq/tree/master/examples/RoBERTa`

	CoNLL05		CoNLL12		CoNLL09	
	F1	Δ	F1	Δ	F1	Δ
Tr-HeSyFu						
Dep.→Const.	84.5	-1.2	83.3	-1.0	91.2	-0.3
Const.→Dep.	85.7		84.3		91.5	
LG-HeSyFu						
Dep.→Const.	85.3	-1.4	84.3	-0.8	92.1	-0.4
Const.→Dep.	86.7		85.1		92.5	

Table 2: Influences of the syntax encoding order.

come universally the best.

Order of the heterogeneous syntax encoding We design the architecture with constituency encoding before dependency encoding, as described earlier. If we exchange this encoding order, we see from Table 2 that the drops come out. Also the drops are more severe on the span-based SRL data. This verifies the correctness of our model design.

5.3 Main Results

Our aim is to answer the research questions as listed in Section §1, based on the main experimental results in Table 1 and Table 3. **[★ Answer to Q1]** Our first observation is that leveraging syntax knowledge, e.g. either the dependency or constituency, benefits both the span-based and dependency-based SRL, while the integration of two heterogeneous syntax contributes the most, more than any one of the standalone syntax.

However we see that the improvements from this syntax integration is slightly different between span-based and dependency-based SRL. **[★ Answer to Q2]** In particular, the improvements for span-based SRL are more notable than dependency-based SRL, which can be learned by the comparisons between 'Trm+RoBERTa' and 'LG-HeSyFu+RoBERTa' on two tables. Our conjecture is that the the constituent structure knowledge will additionally help the span boundary detection of span-based SRL, compared with dependency-based SRL. Also we find that using only constituency syntax contributes more span-based SRL, while the dependency-based SRL benefits more from dependency syntax.

Looking into the specific results, within the scope of heterogeneous syntax integration methods, our systems (both Tr-HeSyFu and LG-HeSyFu) outperform Fei et al. (2020a), demonstrating the advances of our heterogeneous syntax integrating methods. Overall, our LG-HeSyFu model wins the new state-of-the-art performances on the used datasets, and with the help of the RoBERTa language model, the superiority is still maintained.

	P	R	F1
Without Syntax			
He et al. (2018)	89.50	87.90	88.70
Li et al. (2020b)	-	-	90.26
Trm+RoBERTa†	91.34	91.12	91.25
With Dependency Syntax			
Li et al. (2018)	90.30	89.30	89.80
He et al. (2019)	89.96	89.96	89.96
Child-Sum TreeLSTM†	90.67	90.60	90.63
Dep GCN†	90.98	90.85	90.91
Dep GCN+RoBERTa†	92.45	92.05	92.23
With Constituency Syntax			
N-ary TreeLSTM†	89.56	89.21	89.42
Const GCN†	90.48	90.19	90.35
Const GCN+RoBERTa†	91.33	91.87	91.65
With Dependency & Constituency Syntax			
Fei et al. (2020a)*	90.78	90.92	90.88
Tr-HeSyFu†	91.02	91.22	91.10
LG-HeSyFu†	92.24	92.53	92.45
LG-HeSyFu w/o Syn. Label†	91.85	92.15	92.05
LG-HeSyFu+RoBERTa†	**92.89**	**92.80**	**92.83**

Table 3: Results on dependency-based SRL CoNLL09 dataset.

[★ Answer to Q3] Also we show that our LG-HeSyFu based system consistently outperforms Tr-HeSyFu based one. Even LG-HeSyFu without using the syntax label features can still keep better. It is also clear that the GCN based encoders show consistently higher scores than the TreeLSTM based ones, verifying the effectiveness of leveraging GCN encoding syntax (Marcheggiani and Titov, 2017; Li et al., 2018). **[★ Answer to Q4]** Meanwhile, the ablation of syntax label information reveals the importance of its leverage for the SRL learning.

5.4 Analysis and Discussion

Correlations between SRL and syntax structures We explore the correlations between the SRL structure and the two syntax structures. We reach this by analyzing the SRL prediction with the neighbor connecting weights, i.e., α_{uv} of *Const* GCN and β_{ij} of *Dep* GCN. We visualize the results (on CoNLL05) in Figure 6. **[★ Answer to Q5]** We learn that our framework indeed has captured the underlying inter-dependency between the SRL structures and the syntactic structure from the diversified visualizations. By accurately modeling

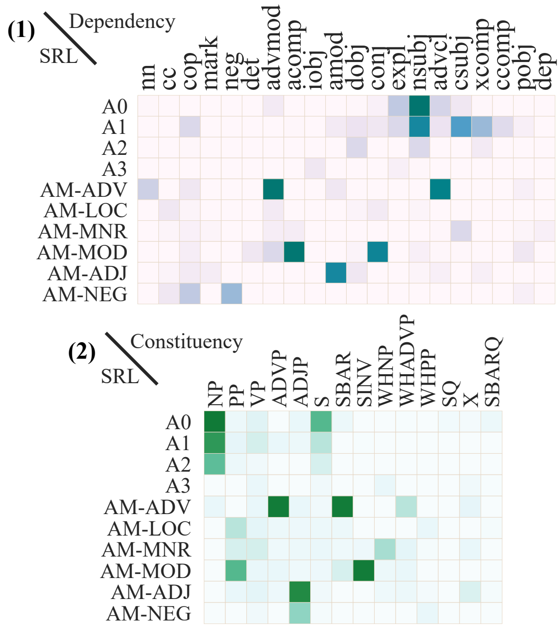

Figure 6: Discovered correlations of (1) SRL vs. dependent structure, (2) SRL vs. constituent structure.

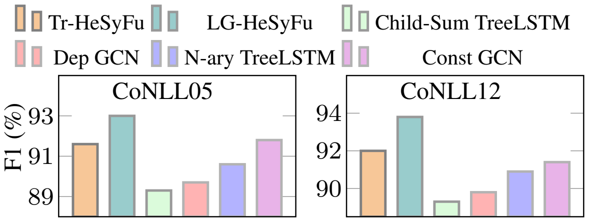

Figure 7: F1 scores for span boundary detection.

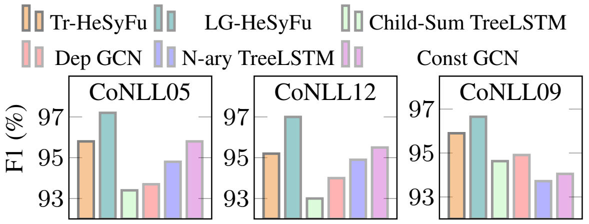

Figure 8: Results on the argument role label prediction.

such correlations, our LG-`HeSyFu` system naturally yields prominent meanwhile explainable SRL performances. Also some interesting patterns can be observed. Actually, not all the syntactic elements contribute the SRL learning. For example, the semantic roles `A0`, `A1` and `A2` relates more to the dependent edge *nsubj* and *csubj*, and more to the constituent phrase *NP*. We believe this can lay a crucial foundation for the direction of unsupervised semantic role labeling that relies on the syntactic structures.

Span boundary detection We now investigate the influences of the heterogeneous syntax integration to the span boundary match[10] on span-based SRL, i.e., on CoNLL05/12 data. From Figure 7 we learn that the heterogeneous syntax integration can improve the boundary detection over any standalone syntax leverage, while actually the constituency syntax contributes more significantly than dependency feature. And our LG-`HeSyFu` shows the best helpfulness than Tr-`HeSyFu`.

Label prediction We next evaluate the role label prediction. We only measure the correctly extracted arguments on whether its label further matches the gold annotation. We show the F1 score in Figure 8. Similar to the span boundary identification, the heterogeneous syntax integration can contribute the most than that with any single syntax usage. Interestingly, the standalone dependency syntax shows more improvements on the dependency-based SRL, while the phrasal constituency features benefit more the span-based SRL.

Error breakdown To analyze which error types different syntax-aided SRL models tend to make, we follow prior works (He et al., 2017; Strubell et al., 2018), manually fixing the errors by applying oracle transformations incrementally based on CoNLL05.[11] The analysis is shown in Figure 9. Specifically, constituency syntax methods perform better than dependency-aided methods, w.r.t. the span boundary errors (‘Merge Spans’, ‘Split Spans’ and ‘Fix Span Boundary’). Most importantly, it is quite clear that our heterogeneous syntax integrated systems (Tr-`HeSyFu` and LG-`HeSyFu`) makes fewer errors than baseline standalone syntax-aware methods, demonstrating the necessity to combine both two types of syntax.

Syntax distribution By observing the gate values $\boldsymbol{g}_i$ (in Eq. 8) we can analyze the distributions of dependency and constituency features required by span-based and dependency-based SRL. From Figure 10 we see that span-SRL relies more on constituency feature, while dependency-SRL needs more dependency-aware feature. Such finding quite coincides with the foregoing quantitative analysis, as well as our intuition.

---

[10] A correct match means both the start and end boundary of an argument span is correct, regardless of its label.

[11] The bigger the correction error improves, the more the model makes on it.

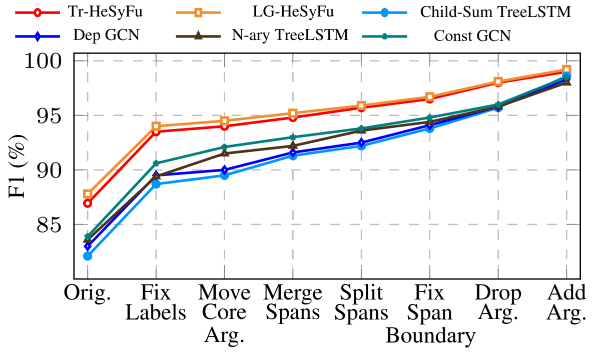

Figure 9: Performances after error corrections.

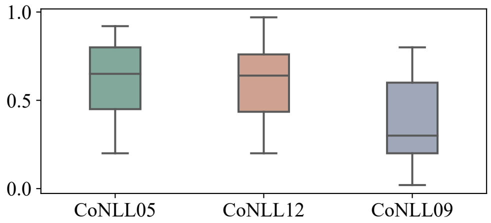

Figure 10: Heterogeneous syntax distribution. Level ≥0.5 means more reliance upon constituent syntax, otherwise for dependency

6 Conclusion and Future Work

We investigated the integration of constituency and dependency syntax for the SRL task. We first introduced TreeLSTM-based heterogeneous syntax fusing encoders, and further proposed innovative label-aware syntax GCN encoders for the integration. Experimental results showed that combining the heterogeneous syntax brought better results on both span-based and dependency-based SRL, than any one standalone syntax knowledge. As future work, we investigate other kinds of structural knowledge integration besides syntax, such as Semantic Dependency Structure, Abstract Meaning Representation (AMR), and explore the possibility of extending our model to incorporating such structured information. Besides, integrating the heterogeneous syntax knowledge into pre-training language models will be a promising direction.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61772378), the National Key Research and Development Program of China (No. 2017YFC1200500), the Research Foundation of Ministry of Education of China (No. 18JZD015).

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the COLING*, pages 86–90.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing*, pages 65–69.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the CoNLL*, pages 152–164.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the NAACL*, pages 132–139.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the ACL*, pages 173–180.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the ACL*, pages 16–23.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the EMNLP*, pages 1535–1545.

Richárd Farkas, Bernd Bohnet, and Helmut Schmid. 2011. Features for phrase-structure reranking from dependency parses. In *Proceedings of the ICPT*, pages 209–214.

Hao Fei, Fei Li, Bobo Li, and Donghong Ji. 2021a. Encoder-decoder based unified semantic role labeling with label-aware syntax. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1479–1488.

Hao Fei, Yafeng Ren, and Donghong Ji. 2020a. Mimic and conquer: Heterogeneous tree structure distillation for syntactic NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 183–193.

Hao Fei, Meishan Zhang, and Donghong Ji. 2020b. Cross-lingual semantic role labeling with high-quality translated training corpus. In *Proceedings of the ACL*, pages 7014–7026.

Hao Fei, Meishan Zhang, Bobo Li, and Donghong Ji. 2021b. End-to-end semantic role labeling with neural transition-based model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 566–575.

Hao Fei, Meishan Zhang, Fei Li, and Donghong Ji. 2020c. Cross-lingual semantic role labeling with model transfer. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2427–2437.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the EMNLP*, pages 960–970.

Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of the ACL*, pages 512–520.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the CoNLL*, pages 1–18.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the ACL*, pages 473–483.

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. In *Proceedings of the EMNLP*, pages 5350–5359.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the ACL*, pages 2061–2071.

Jungo Kasai, Dan Friedman, Robert Frank, Dragomir Radev, and Owen Rambow. 2019. Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the NAACL*, pages 701–709.

Yoshihide Kato and Shigeki Matsubara. 2019. PTB graph parsing with tree approximation. In *Proceedings of the ACL*, pages 5344–5349.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the ICML*, pages 282–289.

Shiqi Li, Qin Lu, Tiejun Zhao, Pengyuan Liu, and Hanjing Li. 2010. Combining constituent and dependency syntactic views for Chinese semantic role labeling. In *Proceedings of the COLING*, pages 665–673.

Tao Li, Parth Anand Jawale, Martha Palmer, and Vivek Srikumar. 2020a. Structured tuning for semantic role labeling. In *Proceedings of the ACL*, pages 8402–8412.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the EMNLP*, pages 2401–2411.

Zuchao Li, Hai Zhao, Rui Wang, and Kevin Parnow. 2020b. High-order semantic role labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1134–1151.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Umar Maqsud, Sebastian Arnold, Michael Hülfenhaus, and Alan Akbik. 2014. Nerdle: Topic-specific question answering using wikia seeds. In *Proceedings of the COLING*, pages 81–85.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the CoNLL*, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the EMNLP*, pages 1506–1515.

Diego Marcheggiani and Ivan Titov. 2020. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the EMNLP*, pages 3915–3928.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of the EMNLP*, pages 1630–1642.

Martha Palmer, Paul R. Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the CoNLL*, pages 143–152.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the ACL*, pages 581–588.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the ACL*, pages 1192–1202.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the ACL*, pages 2245–2254.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the EMNLP*, pages 5027–5038.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Sequence labeling parsing by learning across representations. In *Proceedings of the ACL*, pages 5350–5357.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the CoNLL*, pages 159–177.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the EMNLP*, pages 3772–3782.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the ACL*, pages 1556–1566.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the AAAI*, pages 4929–4936.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the NeurIPS*, pages 5998–6008.

Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019. How to best use syntax in semantic role labelling. In *Proceedings of the ACL*, pages 5338–5343.

Qingrong Xia, Zhenghua Li, Min Zhang, Meishan Zhang, Guohong Fu, Rui Wang, and Luo Si. 2019. Syntax-aware neural semantic role labeling. In *Proceedings of the AAAI*, pages 7305–7313.

Qingrong Xia, Rui Wang, Zhenghua Li, Yue Zhang, and Min Zhang. 2020. Semantic role labeling with heterogeneous syntactic knowledge. In *Proceedings of the COLING*, pages 2979–2990.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for SMT. In *Proceedings of the ACL*, pages 902–911.

Kun Xu, Haochen Tan, Linfeng Song, Han Wu, Haisong Zhang, Linqi Song, and Dong Yu. 2020. Semantic Role Labeling Guided Multi-turn Dialogue ReWriter. In *Proceedings of the EMNLP*, pages 6632–6639.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the ACL*, pages 277–287.

Yue Zhang, Rui Wang, and Luo Si. 2019. Syntax-enhanced self-attention-based semantic role labeling. In *Proceedings of the EMNLP*, pages 616–626.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the CoNLL*, pages 61–66.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the EMNLP*, pages 30–39.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the ACL*, pages 2396–2408.

Keep the Primary, Rewrite the Secondary: A Two-Stage Approach for Paraphrase Generation

Yixuan Su[♠] **David Vandyke**[♣] **Simon Baker**[♠] **Yan Wang**[♢] **Nigel Collier**[♠]
[♠]Language Technology Lab, University of Cambridge
[♣]Apple
[♢]Tencent AI Lab
{ys484,sb895,nhc30}@cam.ac.uk
dvandyke@apple.com, brandenwang@tencent.com

Abstract

Paraphrase generation is an important and challenging NLG problem. In this work, we propose a new Identification-then-Aggregation (IA) framework to tackle this task. In the identification step, the input tokens are sorted into two groups by a novel Primary/Secondary Identification (PSI) algorithm. In the aggregation step, these groups are separately encoded, before being aggregated by a custom designed decoder, which autoregressively generates the paraphrased sentence. In extensive experiments on two benchmark datasets, we demonstrate that our model outperforms previous studies by a notable margin. We also show that the proposed approach can generate paraphrases in an interpretable and controllable way.

1 Introduction

Paraphrases refer to text (often sentences) that share the same meaning but use different choices of words and their ordering. Automatic generation of paraphrases is a longstanding problem that is important to many downstream NLP applications such as question answering (Dong et al., 2017; Buck et al., 2018), machine translation (Cho et al., 2014), and semantic parsing (Su and Yan, 2017). Most early research adopts the sequence-to-sequence model (Prakash et al., 2016; Cao et al., 2017; Li et al., 2018) to map the input text to its paraphrase by processing and generating each word in a uniform way. Rather than processing each word uniformly, some recent studies tackles this task in a decomposable manner. For instance, Li et al. (2019) adopt an external word aligner to extract paraphrasing patterns at different levels of granularity and then perform generation. Fu et al. (2019) first use source words to predict their neighbors and then organize the predicted neighbors into a complete sentence.

How are baby elephants called?
What is a name for a baby elephant?

What is the best way to use time?
How can we utilize time effectively?

Many people are relaxing on the pretty beach.
A group of people relaxing on the sand at a beach.

A refrigerator and a stove on a street.
A yellow stove and refrigerator and a building.

Figure 1: Examples of paraphrase pair sampled from Quora and MSCOCO datasets in which the words in red refer to the primary content and the rest of the words make up the secondary content.

In this work, we investigate decomposable paraphrase generation from a different perspective. Specifically, we consider using a non-parametric approach to label each token in an input sentence as either (i) primary, or (ii) secondary. Intuitively, the primary content of a sentence refers to the factual information that defines the shared meaning of the paraphrase pair. All other content is deemed as secondary, and typically controls the structure of the sentence. In practice, this distinction is determined by an algorithm that decides whether tokens are primary or secondary, as described in §3. To better illustrate our idea, in Figure 1, we show some examples sampled from Quora and MSCOCO (Lin et al., 2014) datasets. We see that, for many cases, the paraphrase pairs maintain the similar primary content (e.g., the phrases "*baby elephants*" and "*baby elephant*" in the first example) while the secondary content can be rephrased in several different ways.

Based on the above observation, we propose an Identification-then-Aggregation (IA) framework to address the paraphrase generation task. Given an input sentence, generating a paraphrase follows a two-stage process. First, the primary and secondary content of the input sentence is identified via a novel Primary/Secondary Identification (PSI) algo-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 560–569
August 1–6, 2021.

rithm which is based on a common non-parametric rank coefficient. Second, a new neural paraphrase generation model aggregates the identified information and generates the result. Specifically, the proposed model consists of (1) two encoders which separately process the identified primary and secondary content; and (2) an aggregation decoder which integrates the processed results and generates the paraphrased sentence.

We test the proposed approach on two benchmark datasets with automatic and human evaluation. The results show that our approach outperforms previous studies and can generate paraphrases in an interpretable and controllable way.

2 Related Work

The automatic generation of paraphrases is important for many downstream NLP applications and it has attracted a number of different approaches. Early researches included rule-based approaches (McKeown, 1979; Meteer and Shaked, 1988) and data-driven methods (Madnani and Dorr, 2010). With the advances of neural networks, recent approaches tackle this problem by treating it as a sequence-to-sequence language generation task. Prakash et al. (2016) proposed to modify the networks structure to improve the generation quality. Cao et al. (2017), Wang et al. (2019), and Kazemnejad et al. (2020) proposed to improve the model performance by leveraging external resources, including phrase dictionary, semantic annotations, and an off-the-shelf pre-trained neural retriever. Other works proposed to adopt techniques like reinforcement learning (Li et al., 2018) and unsupervised learning (Roy and Grangier, 2019) for this task.

While achieving satisfactory results, these above methods do not offer users the way to control the generation process in a fine-grained way. To incorporate controllability into the generation model, different approaches have been proposed. Iyyer et al. (2018) trained the model to produce the paraphrased sentence with a given syntax. Li et al. (2019) proposed to adopt an external word aligner to train the model to generate paraphrases from different levels. In Fu et al. (2019)'s work, the model generates paraphrases by planning the neighbour of words and realizing the complete sentence.

3 Primary/Secondary Identification

Given an input sentence, our goal is to identify the primary content that are likely to appear in the paraphrased sentence. To this end, we propose a Primary/Secondary Identification (PSI) approach which dynamically evaluates the importance of different parts of the input sentence. The parts with high importance are deemed the primary content, while the rest parts are deemed secondary content.

Token Importance Formally, given a paraphrase pair $\mathbf{X}$ and $\mathbf{Y}$, we define their pairwise similarity as $\mathcal{F}(\mathbf{X},\mathbf{Y})$. To determine the importance of the i-th token x_i of $\mathbf{X}$ in relation to $\mathbf{Y}$, we first compute the pairwise similarity between $\mathbf{X}' = \mathbf{X} \ominus x_i$ and $\mathbf{Y}$ as $\mathcal{F}(\mathbf{X}',\mathbf{Y})$, where the $\ominus$ operator removes the token x_i from $\mathbf{X}$. We assume that if the token x_i belongs to the primary content that is maintained in both $\mathbf{X}$ and $\mathbf{Y}$, then removing it from $\mathbf{X}$ will cause a significant drop in the pairwise similarity between $\mathbf{X}$ and $\mathbf{Y}$. Based on this assumption, we measure the importance of x_i as the ratio of change in the pairwise similarity score as

$$\mathcal{G}(x_i;\mathbf{X},\mathbf{Y}) = \frac{\mathcal{F}(\mathbf{X},\mathbf{Y}) - \mathcal{F}(\mathbf{X}',\mathbf{Y})}{\mathcal{F}(\mathbf{X},\mathbf{Y})}. \qquad (1)$$

Intuitively, a higher $\mathcal{G}(x_i;\mathbf{X},\mathbf{Y})$ means a larger decrease in the pairwise similarity, indicating a higher importance of the token x_i and vice versa.

Similarity Measurement We now describe the details of the function $\mathcal{F}(\cdot,\cdot)$. Inspired by Zhelezniak et al. (2019), we measure the pairwise similarity between $\mathbf{X}$ and $\mathbf{Y}$ based on a non-parametric rank correlation coefficient. Specifically, given $\mathbf{X}$ and $\mathbf{Y}$, we first transform them into the representation matrices $\mathcal{M}(\mathbf{X}) \in \mathbb{R}^{|\mathbf{X}|\times D}$ and $\mathcal{M}(\mathbf{Y}) \in \mathbb{R}^{|\mathbf{Y}|\times D}$ via a D-dimensional pretrained embeddings. Then, the matrices are mapped into fixed size context vectors $\hat{x} \in \mathbb{R}^{1\times D}$ and $\hat{y} \in \mathbb{R}^{1\times D}$ via an element-wise max-pooling operation. Finally, the pairwise similarity $\mathcal{F}(\mathbf{X},\mathbf{Y})$ is measured using Spearman's correlation coefficient $\hat{\rho}$ of the context vectors $\hat{x}$ and $\hat{y}$ as

$$\mathcal{F}(\mathbf{X},\mathbf{Y}) = 1 - \frac{6 \times \sum_{j=1}^{D}(r[\hat{x}_j] - r[\hat{y}_j])^2}{D \times (D^2 - 1)} \qquad (2)$$

where $r[\hat{x}_j]$ denotes the integer rank of $\hat{x}_j$ in the context vector $\hat{x}$ (similarly $r[\hat{y}_j]$).

For a better illustration, in Table 1, we show sentence sampled from Quora and MSCOCO datasets along with their pairwise similarities. We see that the numerical results are highly correlated with human judgement which empirically demonstrate the effectiveness of our measurement method.

Sentence 1	Sentence 2	Pairwise Similarity
What are the best games you can play with just paper?	Which games can you play on paper in your free time with your friends?	0.745
	What can I do to become a professional chess player?	0.438
	Should animals be used for testing medicines and drugs?	0.140
Three bears walking on dried grass toward the green wooded area.	Two brown bears walking through a green, grassy area.	0.743
	A simple plain clear vase with a dead twig and water inside.	0.439
	A man using a phone next to a motorcycle.	0.128

Table 1: Examples of different sentence pairs $(\mathbf{X}, \mathbf{Y})$ and their corresponding pairwise similarity scores $\mathcal{F}(\mathbf{X}, \mathbf{Y})$.

Algorithm 1: Primary/Secondary Identification

```
Input : Input sentence $\mathbf{X} = (x_1, ..., x_{|\mathbf{X}|})$;
        Paraphrased sentence $\mathbf{Y} = (y_1, ..., y_{|\mathbf{Y}|})$;
        Primary content threshold $\alpha_p$;
        Importance measurement function $\mathcal{G}(\cdot, \cdot, \cdot)$.
$\mathbf{X}_p \leftarrow \{\}$; $\mathbf{X}_s \leftarrow \{\}$;
for $i = 1$ to $N$ do
    $\mathbf{X}' \leftarrow \mathbf{X} \ominus x_i$;
    if $\mathcal{G}(x_i; \mathbf{X}, \mathbf{Y}) > \alpha_p$ then
        $\mathbf{X}_p \leftarrow x_i$ and $\mathbf{X}_s \leftarrow$ [MASK];
    else
        $\mathbf{X}_p \leftarrow$ [MASK] and $\mathbf{X}_s \leftarrow x_i$;
    end
end
$\mathbf{X}_p \leftarrow$ joinmask($\mathbf{X}_p$); $\mathbf{X}_s \leftarrow$ joinmask($\mathbf{X}_s$).;
Output : Primary Content $\mathbf{X}_p$;
         Secondary Content $\mathbf{X}_s$.
```

Putting this together, the detailed description for splitting the input sentence $\mathbf{X}$ into the primary content $\mathbf{X}_p$ and secondary content $\mathbf{X}_s$ is given in Algorithm 1, where the token `[MASK]` is used as a special placeholder and the threshold α_p is tuned based on the performance on the validation set[1]. The joinmask$(\cdot)$ operation joins consecutive `[MASK]` tokens into a single `[MASK]` token. We note that the incorporation of the `[MASK]` token is crucial. Because, in this way, the generation model could have access to the original source sentence structure by simply overlapping the primary and secondary content. In the experiments, we found that removing `[MASK]` from the identified content causes a significant drop in model performance as the model can no longer have access to the original sentence structure.

In Figure 2, we show the computed results from PSI of an example presented in Figure 1. We can see that the primary content is effectively identified.

Inference During inference, given an input sentence, the primary and secondary content could not be directly identified as $\mathbf{X}_p, \mathbf{X}_s = \text{PSI}(\mathbf{X}, \mathbf{Y})$ since we do not have access to the target sentence $\mathbf{Y}$. To this end, we propose two alternative approaches. For the first one, we simply identify the primary and secondary content using the input sentence $\mathbf{X}$ as $\mathbf{X}'_p, \mathbf{X}'_s = \text{PSI}(\mathbf{X}, \mathbf{X})$. For the second one, we train a neural sequence tagger $\mathcal{S}$ based on the labels provided by PSI$(\mathbf{X}, \mathbf{Y})$. Then we extract the content using the input $\mathbf{X}$ as $\mathbf{X}'_p, \mathbf{X}'_s = \mathcal{S}(\mathbf{X})$. In the experiment section, we provide more detailed comparisons between these approaches.

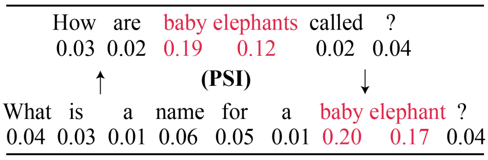

Figure 2: For each token, the score from the PSI algorithm is presented. The words in red is the identified primary content and the rest words make up the secondary content.

4 Neural Paraphrase Generator

Overview Given the input sentence $\mathbf{X}$, it is first partitioned into the primary and secondary content using the PSI algorithm. Then the identified content is independently processed by the primary encoder and the secondary encoder. Finally, an aggregation decoder integrates the outputs from both encoders and generates the result. In Figure 3, we provide an illustration of the proposed framework.

Encoder Stacks In this work, we use the transformer architecture (Vaswani et al., 2017) to construct the primary and secondary encoders. Formally, the Multi-Head Attention is defined as MultiHead$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are query, key and value. Each encoder has N_E layers. Given the input $\mathbf{X}$, the first layer operates as

$$\mathbf{V}_{\mathbf{X}}^{(1)} = \text{MultiHead}(E(\mathbf{X}), E(\mathbf{X}), E(\mathbf{X})), \quad (3)$$

$$\mathbf{O}_{\mathbf{X}}^{(1)} = \text{FFN}(\mathbf{V}_{\mathbf{X}}^{(1)}), \quad (4)$$

---

[1] In this work, we set α_p as 0.1 for all experiments based on the model performance on the validation set.

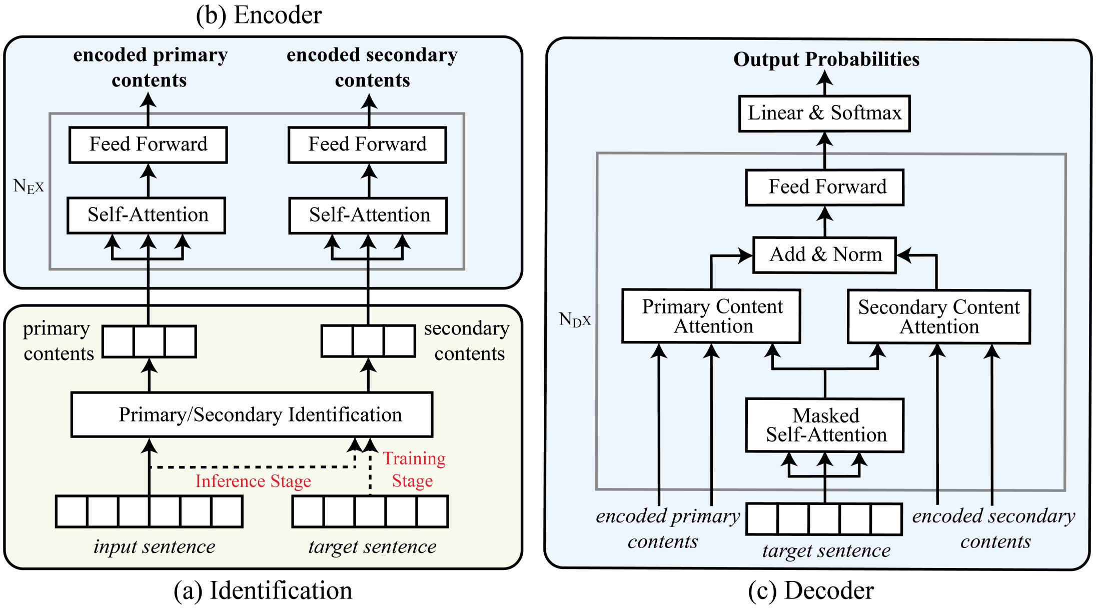

(b) Encoder

(a) Identification

(c) Decoder

Figure 3: Overview of the proposed framework: *Italic* and **Boldface** denote the inputs and outputs at each stage. For a better illustration, we separately draw the encoder and decoder of the paraphrase generator. **(a)** During training, the primary and secondary content of the input sentence are identified by the PSI algorithm using the input and target sentence. During inference, the content is identified solely based on the input sentence. **(b)** The identified results are then encoded by separate encoders. **(c)** The aggregation decoder takes the encoded primary and secondary content as input and produces the probability of the target sentence. It should be noted that, during the training stage, the encoder and decoder are jointly trained in an end-to-end fashion.

where $E(\mathbf{X})$ is the input sequence embedding and FFN$(\cdot)$ is a feed-forward layer. For other layers:

$$\mathbf{V}_{\mathbf{X}}^{(n)} = \text{MultiHead}(\mathbf{O}_{\mathbf{X}}^{(n-1)}, \mathbf{O}_{\mathbf{X}}^{(n-1)}, \mathbf{O}_{\mathbf{X}}^{(n-1)}), \tag{5}$$

$$\mathbf{O}_{\mathbf{X}}^{(n)} = \text{FFN}(\mathbf{V}_{\mathbf{X}}^{(n)}), \tag{6}$$

where $n = 2, ..., N_E$.

Given the primary content $\mathbf{X}_p$ and secondary content $\mathbf{X}_s$ of the input sequence, their representations $\mathbf{O}_{\mathbf{X}_p}^{(N_E)} \in \mathbb{R}^{|\mathbf{X}_p| \times d}$ and $\mathbf{O}_{\mathbf{X}_s}^{(N_E)} \in \mathbb{R}^{|\mathbf{X}_s| \times d}$ are computed by the primary and secondary encoder respectively and d is the model size.

Decoder Stacks We design an aggregation decoder to integrate information coming from both encoders. Given the target sentence $\mathbf{Y}$, it is first encoded via a masked multi-head attention as

$$\mathbf{V}_{\mathbf{Y}_m}^{(1)} = \text{Mask-MultiHead}(E(\mathbf{Y}), E(\mathbf{Y}), E(\mathbf{Y})). \tag{7}$$

Then, the primary content attention module takes the encoded primary content $\mathbf{O}_{\mathbf{X}_p}^{(N_E)}$ and $\mathbf{V}_{\mathbf{Y}_m}^{(1)}$ as input and produces the intermediate result $\mathbf{V}_{\mathbf{Y}_m^p}^{(1)}$ as

$$\mathbf{V}_{\mathbf{Y}_m^p}^{(1)} = \text{MultiHead}(\mathbf{V}_{\mathbf{Y}_m}^{(1)}, \mathbf{O}_{\mathbf{X}_p}^{(N_E)}, \mathbf{O}_{\mathbf{X}_p}^{(N_E)}). \tag{8}$$

Similarly, the result $\mathbf{V}_{\mathbf{Y}_m^s}^{(1)}$ from the encoded secondary content $\mathbf{O}_{\mathbf{X}_s}^{(N_E)}$ is computed as

$$\mathbf{V}_{\mathbf{Y}_m^s}^{(1)} = \text{MultiHead}(\mathbf{V}_{\mathbf{Y}_m}^{(1)}, \mathbf{O}_{\mathbf{X}_s}^{(N_E)}, \mathbf{O}_{\mathbf{X}_s}^{(N_E)}). \tag{9}$$

The first layer output $\mathbf{O}_{\mathbf{Y}}^{(1)}$ is then acquired as

$$\mathbf{V}_{\mathbf{Y}_i}^{(1)} = \text{LayerNorm}(\mathbf{V}_{\mathbf{Y}_m^p}^{(1)} + \mathbf{V}_{\mathbf{Y}_m^s}^{(1)}), \tag{10}$$

$$\mathbf{O}_{\mathbf{Y}}^{(1)} = \text{FFN}(\mathbf{V}_{\mathbf{Y}_i}^{(1)}). \tag{11}$$

The final output $\mathbf{O}_{\mathbf{Y}}^{(N_D)} \in \mathbb{R}^{|\mathbf{Y}| \times d}$ is computed via a stack of N_D layers. The final probability of $\mathbf{Y}$ is produced by a linear softmax operation.

Learning Finally, given the input primary content $\mathbf{X}_p$, secondary content $\mathbf{X}_s$ and the target sequence $\mathbf{Y}$, the learning objective is defined as

$$\mathcal{L} = \sum_{t=1}^{|\mathbf{Y}|} \log p(\mathbf{Y}_t | \mathbf{Y}_{<t}, \mathbf{X}_p, \mathbf{X}_s). \tag{12}$$

5 Datasets

We test our approach on two benchmark paraphrase generation datasets: (1) Quora dataset[2] and (2) MSCOCO dataset (Lin et al., 2014).

---

[2] https://www.kaggle.com/c/quora-question-pairs

The Quora dataset was developed for the task of duplicated question detection. Each data instance consists of one source sentence and one target sentence. In the experiment, we randomly select one sentence as the source and the other as the target.

The MSCOCO dataset was originally developed for the image captioning task. In this dataset, each image is associated with five human-written captions. Although there is no guarantee that these captions must be paraphrases as they could describe different objects in the image, most of these captions are generally close to each, therefore the overall quality of this dataset is favorable and it is widely used for the paraphrase generation task.

Following Li et al. (2019) and Fu et al. (2019), for the Quora dataset, we split the size of training, validation and test sets as 100k, 4k and 20k. The MSCOCO dataset is split into 93k, 4k and 20k. The maximum sentence length for these two datasets is set as 16. The vocabulary size of the Quora and MSCOCO datasest are set to be 8k and 11k.

6 Experiments

6.1 Model Comparisons

We compare the proposed model with several representative baselines, including Residual-LSTM (Prakash et al., 2016), β-VAE (Higgins et al., 2017), Transformer (Vaswani et al., 2017), DNPG (Li et al., 2019), LBOW-Topk and LBOW-Gumbel (Fu et al., 2019)[3]. To compare different inference approaches, three variants of our model are used.

IANet+X: Given the input sentence $\mathbf{X}$, this model extracts the primary and secondary content using the approximated PSI$(\mathbf{X}, \mathbf{X})$ algorithm. Then, the paraphrase generator produces the paraphrased sentence using the identified content.

IANet+$\mathcal{S}$: In this case, a neural sequence tagger $\mathcal{S}$ is first trained based on the labels provided by PSI$(\mathbf{X}, \mathbf{Y})$. During inference, the model extracts the primary and secondary content of the input as $\mathbf{X}'_p, \mathbf{X}'_s = \mathcal{S}(\mathbf{X})$ and then perform generation.

IANet+ref: In contrast to previous variants, this model obtains the primary and secondary content using the exact PSI$(\mathbf{X}, \mathbf{Y})$ algorithm against the reference $\mathbf{Y}$. The reason to include this model is that, besides our proposed alternatives, there are other options that we can use. We will explore these options in the future work. But by evaluating IANet+ref we can show an upper bound on how much could be improved in this way.

6.2 Implementation Details

We implement our model with PyTorch (Paszke et al., 2017). For the primary and secondary encoders, we use a 3-layer transformer with model size of 256 and heads of 8. Since the decoder has to integrate the information from both encoders, we build it with a larger capacity. The number of layers is set to 4. The model size and the attention heads are set to be 512 and 8. For the sequence tagger $\mathcal{S}$ that is used in the IANet+$\mathcal{S}$ model, we use a 2-layer LSTM with hidden size of 512.

In the experiments, we adopt pretrained 300-dimensional FastText Embeddings (Bojanowski et al., 2017) to perform the PSI algorithm. During training , we use Adam (Kingma and Ba, 2015) to optimize our model with a learning rate of 1e-4. In all experiments, we set α_p in Algorithm 1 as 0.1 based on the performance on the validation set.

6.3 Evaluation Metrics

Following previous studies (Prakash et al., 2016; Fu et al., 2019; Li et al., 2019), we report results on several automatic metrics, including BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). All lower n-gram metrics (1-4 grams in BLEU and 1-2 grams in ROUGE) are reported. In addition, we include iBLEU (i-B) (Sun and Zhou, 2012) as another evaluation metric, which penalizes repeating the source sentence in its paraphrase.

6.4 Main Results

Table 2 lists the results on both datasets. We see that the transformer baseline already achieves pretty strong results. This is because the capacity of transformer model is large enough to fit the datasets quite well. Nonetheless, in most of the evaluation metrics, our model outperforms previous studies by a notable margin, demonstrating the effectiveness of the proposed approach.

By comparing different variants of our model, we see that IANet-ref achieves the best results on all metrics. This is expected as it uses the reference sentence in determining the primary and secondary content. It is worth emphasising that the IANet+ref model, like everything else, does not receive the target sentence, it just gets inputs $\mathbf{X}$ and has their primary and secondary content more accurately

---

[3] The hyperparameter setups and optimization in all baseline models are the same as their original works. For methods that do not release their code, we directly use the results in their original papers.

Quora

Models	B-1	B-2	B-3	B-4	i-B	R-1	R-2	R-L
Residual-LSTM (Prakash et al., 2016)	53.59	39.49	30.25	23.69	15.93	55.10	33.86	53.61
β-VAE, $\beta = 10^{-4}$ (Higgins et al., 2017)	47.86	33.21	24.96	19.73	10.28	47.62	25.49	45.46
Transformer (Vaswani et al., 2017)	53.56	40.47	32.11	25.01	17.98	57.82	32.58	56.26
DNPG (Li et al., 2019)	-	-	-	25.03	18.01	**63.73**	**37.75**	-
LBOW-Topk (Fu et al., 2019)	55.79	42.03	32.71	26.17	19.03	58.79	34.57	56.43
LBOW-Gumbel (Fu et al., 2019)	55.75	41.96	32.66	26.14	18.97	58.60	34.47	56.23
IANet+$\mathbf{X}$	56.06	42.69	33.38	26.52	19.62	59.33	35.01	57.13
IANet+$\mathcal{S}$	**56.72**	**43.21**	**33.96**	**27.09**	**20.11**	59.98	36.02	**58.01**
IANet+ref (upperbound)	58.32	44.81	35.46	28.71	21.76	61.89	38.86	59.43

MSCOCO

Models	B-1	B-2	B-3	B-4	i-B	R-1	R-2	R-L
Residual-LSTM (Prakash et al., 2016)	70.24	48.65	34.04	23.66	18.72	41.07	15.26	37.35
β-VAE, $\beta = 10^{-4}$ (Higgins et al., 2017)	70.04	47.59	32.29	22.54	18.34	40.72	14.75	36.75
Transformer (Vaswani et al., 2017)	71.31	49.86	35.55	24.68	19.81	41.49	15.84	37.09
LBOW-Topk (Fu et al., 2019)	72.60	51.14	35.66	25.27	21.07	42.08	16.13	38.16
LBOW-Gumbel (Fu et al., 2019)	72.37	50.81	35.32	24.98	20.92	42.12	16.05	38.13
IANet+$\mathbf{X}$	72.10	52.22	37.39	26.06	21.28	43.81	16.35	39.65
IANet+$\mathcal{S}$	**73.01**	**53.09**	**38.12**	**26.90**	**22.03**	**44.66**	**17.13**	**40.58**
IANet+ref (upperbound)	75.29	55.09	41.01	29.65	24.72	46.36	19.13	42.23

Table 2: Evaluation results on the Quora and MSCOCO dataset. B for BLEU and R for ROUGE. Where possible we copy results from DNPG (Li et al., 2019) as they did not release their code.

identified. This suggests that the deomposition of our approach is beneficial, and further work can be focused more on the identification step. On the other hand, without using the target sentence, both IANet+$\mathbf{X}$ and IANet+$\mathcal{S}$ must use an approximated approach at the inference time, which inevitably introduces noise in the identified content. As a result, the performance is lower than IANet+ref. We will provide more analysis in the analysis section.

6.5 Human Evaluation

We also conduct a human evaluation to assess our model, using graders proficient in English from an internal grading platform. We randomly select 150 examples from the Quora test set and compare our model with three representative baselines[4]. Three annotators are asked to rate the generated results from different models on a 3-point Likert scale (0, 1, or 2) with respect to the following features[5]:

- **Fluency**: Whether the generated paraphrase is grammatically correct and easily understood.

	Fluency	**Accuracy**	**Diversity**
Agreement	0.582	0.543	0.498
Residual-LSTM	1.57	1.41	1.29
Transformer	1.63	1.50	1.40
LBOW-Topk	1.61	1.49	1.43
IANet+$\mathcal{S}$	**1.71**	**1.57**	**1.61**
Reference	1.85	1.78	1.68

Table 3: Human Evaluation Results

- **Accuracy**: Whether the content in the generated paraphrase is consistent with the content in the original sentence.
- **Diversity**: Whether the generated sentence structure differs from the reference sentence.

To measure the agreement between the annotators, we use the Fleiss$'$ kappa coefficient (Fleiss et al., 1971). The agreement results are shown in the first row of Table 3, indicating moderate agreement between annotators on all metrics.

From Table 3, we see that our model achieves the best result on all metrics, which demonstrates the effectiveness of the proposed approach. Especially, on the diversity metric, our model significantly outperforms other baselines (Sign Test, with p-value < 0.05) and performs comparably with the

---

[4] Because the authors of DNPG (Li et al., 2019) did not release their code, thus we are not able to reproduce their results and are not able to include this model in the human evaluation.

[5] More details of the human evaluation guideline can be found in the supplementary material.

	Rec.(%)	Prec.(%)	F1(%)	B-4	R-L
PSI($\mathbf{X}, \mathbf{X}$)	71.2	88.5	78.9	26.52	57.13
Tagger $\mathcal{S}$	78.1	90.1	83.7	27.09	58.01
PSI($\mathbf{X}, \mathbf{Y}$)	**87.4**	**91.3**	**89.3**	**28.71**	**59.43**

Table 4: Comparisons between the identification performances of different algorithms on Quora dataset

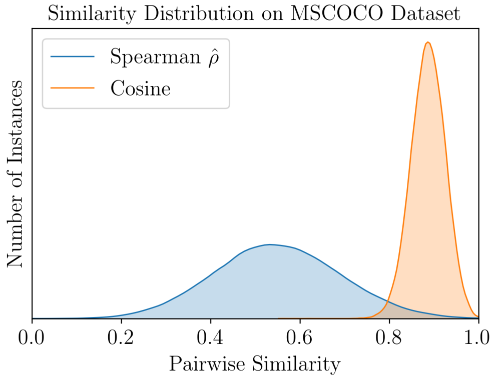

Figure 4: Comparison of similarity distribution

reference sentence (p-value = 0.23). The improvement in the diversity metric mainly comes from the two-step nature of our generation framework. By first determining which parts of the sentence to keep (primary) or to change (secondary), our model could then focus on maintaining the primary content while rewriting the secondary content, resulting in a more accurate and diverse paraphrase.

6.6 Further Analysis

In this section, we present further discussions and empirical analysis of the proposed approach.

6.6.1 Inference Algorithms Comparison

As shown in Table 2, IANet-ref outperforms IANet-$\mathbf{X}$ and IANet-$\mathcal{S}$ on both datasets. Our analysis is that IANet-ref could more accurately identify the primary content from the source comparing with the other variants. To provide more analysis, we separately use PSI($\mathbf{X}, \mathbf{X}$), the sequence tagger $\mathcal{S}$, and PSI($\mathbf{X}, \mathbf{Y}$) to identify the words that both appear in the source and target sentences in the Quora dataset. The results are shown in Table 4. We see that all three methods perform comparably on the precision (prec.) metric. However, PSI($\mathbf{X}, \mathbf{Y}$) significantly outperforms other methods on the recall (rec.) metric, showing that it can accurately extract more primary content from the source. From Table 4, we also observe that better identification results lead to better generation performances. Therefore, to improve the model performance, future work could be focused more on the identification step.

Metric	B-2	B-4	R-2	R-L
Cosine	50.34	24.91	14.78	37.62
Spearman $\hat{\rho}$	**53.09**	**26.90**	**17.13**	**40.58**

Table 5: Result comparison between the cosine similarity and the Spearman's $\hat{\rho}$ on the MSCOCO dataset.

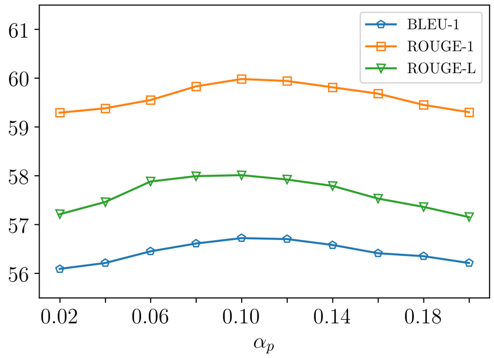

Figure 5: Effect of α_p on the model performance

6.6.2 Similarity Measurement Comparison

In this part, we analyze the differences between different similarity measurements. As described in Eq. (1) and Algorithm 1, the pairwise similarity measurement $\mathcal{F}(\mathbf{X}, \mathbf{Y})$ is the basis of the PSI algorithm. To see how different similarity measurements affect the system performance, we compare the adopted Spearman's $\hat{\rho}$ with the cosine similarity which is commonly used for measuring text similarity. We use both metrics to measure the training pair similarity of the MSCOCO dataset and the results are shown in Figure 4. As it can be seen that the distribution of cosine similarity is condensed in a much smaller interval comparing with the one from Spearman's $\hat{\rho}$, showing that the Spearman's $\hat{\rho}$ is more discriminative and can detect more subtle differences between the sentence pairs. Therefore, it can better identify the primary content, leading to better model performance. For further analysis, we run experiments on the MSCOCO dataset using cosine similarity as the measurement approach. The results of IANet+$\mathcal{S}$ using both metrics are shown in Table 5 which also demonstrate the fact that a more discriminative measurement approach leads to a better model performance.

6.6.3 Effect of α_p in PSI

As described in Algorithm 1, the proposed PSI algorithm relies on a predefined threshold α_p to perform the extraction of primary and secondary content. In this part, we examine the effect of different α_p on the model performance. We vary the value of α_p and measure the results of IANet+$\mathcal{S}$

	Input Sentence	**Reference Sentence**
	what are some of the best young adult fiction novels ?	which are the best young adult novels / films ?
	Input Sentence with Identified Components	**Generated Paraphrase**
IANet+ref	what are some of the best young adult fiction novels ?	which are the best young adult novels ?
IANet+$\mathcal{S}$	what are some of the best young adult fiction novels ?	which is a good young adult fiction novel you have read ?
IANet+$\mathbf{X}$	what are some of the best young adult fiction novels ?	which are the best fiction novels ever written ?
	Controlled Paraphrase Generation	
	what are some of the best young adult fiction novels ?	which are some good adult fiction novels ?
	what are some of the best young adult fiction novels ?	which is the best fiction novel ?
	what are some of the best young adult fiction novels ?	which are some good novels of all time ?

Table 6: **Paraphrase Generation Samples from Quora dataset**: To compare different inference algorithms, we present the results of different model variants for the sampled instance. To examine the generation controllability of the proposed model, we generate sentences by manually decomposing the input sentence. Specifically, we choose different parts of the input sentence as the primary content (highlighted in blue) and the rest as the secondary content. The results on the right side are the corresponding generated paraphrases.

model on the Quora dataset. The results of three metrics (B-1, R-1, and R-L) are depicted in Figure 5. We see that the optimal value of α_p is 0.1 and by further decreasing or increasing α_p, the model performance drops. Our analysis is that, when α_p is too small, the words that only cause small variation in the pairwise similarity (Eq. (1)) will be misclassified as primary. Therefore, extra noise might be introduced to the model input which in turns decreases the model performance. On the other hand, when α_p is too large, some important words that should be classified as primary content might be excluded by the PSI algorithm, which also leads to the decrease of model performance.

6.6.4 Case Study

As described in section §6.6.1, the reason why IANet+ref outperforms IANet+$\mathbf{X}$ and IANet+$\mathcal{S}$ is that it can more accurately identify the primary content in order to generate a paraphrase that is similar to the reference sentence. On the other hand, both IANet+$\mathbf{X}$ and IANet+$\mathcal{S}$ adopt an approximated algorithm which would inevitably introduce extra noise in the identified content.

For a better illustration, we sample one test case from Quora dataset and present the results generated by our different model variants in Table 6. We can see that, given the input sentence, all model variants can generate a sentence that is similar to the reference paraphrase. By further comparing the primary content (words in blue), we can see that only the IANet+ref successfully identifies all the primary content that are also contained in the reference sentence. On the other hand, IANet+$\mathbf{S}$ misses the word *best* and IANet+$\mathbf{X}$ ignores the words *young* and *adult*. As a result, IANet+ref can generate paraphrase that is closer to the reference sentence, leading to higher performances in different evaluation metrics as shown in Table 2.

6.6.5 Controllable Paraphrase Generation

Since the identification of the primary and secondary content of the input are separated from the neural generator, we therefore have the flexibility to manually choose these content. In this way, we can more precisely control the generation process.

To examine the controllability of the proposed approach, we manually select the primary and secondary content of the sampled instance and use the IANet+$\mathcal{S}$ model to generate paraphrases accordingly. The results based on different selections are presented in Table 6. As demonstrated by the examples, our model is flexible to generate different paraphrases given different combinations of the primary and secondary content. We can observe in the generated paraphrases that the selected primary content is largely maintained while the secondary content is properly rephrased.

This controllable attribute could make our model useful for other tasks such as task-oriented dialogue generation. Suppose we want to generate more utterances with the same meaning of the user utterance "*book a great* restaurant *in* London *tonight*". The slot values can be fixed as the primary content and our model could produce more utterances with the same intent, e.g. "*make a reservation at the best* London restaurant *for this evening*". This remains to be rigorously tested in future work.

7 Conclusion

In this work, we propose a novel IA framework to tackle the paraphrase generation task. Additionally, we design a new neural paraphrase generator which works coherently under the proposed framework. We conduct extensive experiments on two benchmark datasets. The results of quantitative experiments and human evaluation demonstrate that our approach improves upon previous studies. The qualitative experiments show that the generation of the proposed model is interpretable and controllable. In the future, we would like to investigate a better inference algorithm to further bridge the gap between the IANet+$\mathcal{S}$ and IANet+ref models.

Acknowledgments

The authors wish to thank Jialu Xu, Deng Cai, and Sihui Wang for their insightful discussions and support. Many thanks to our anonymous reviewers for their suggestions and comments.

Ethical Statement

We honor and support the ACL Code of Ethics. Paraphrase generation aims at automatically generating paraphrased sentence given the source sentence. The generation process does not involve any bias towards the users. All datasets used in this work are from previously published works, and in our view, do not have any attached privacy or ethical issues.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3152–3158.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 875–886.

J.L. Fleiss et al. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Yao Fu, Yansong Feng, and John P. Cunningham. 2019. Paraphrase generation with latent bag of words. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13623–13634.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1875–1885.

Amirhossein Kazemnejad, Mohammadreza Salehi, and Mahdieh Soleymani Baghshah. 2020. Paraphrase generation by learning how to edit from samples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6010–6021. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3865–3878.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019,* *Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3403–3414.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 740–755.

Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Kathleen R. McKeown. 1979. Paraphrasing using given and new information in a question-answer system. In *17th Annual Meeting of the Association for Computational Linguistics, 29 June - 1 July 1979, University of California at San Diego, La Jolla, CA, USA*.

Marie Meteer and Varda Shaked. 1988. Strategies for effective paraphrasing. In *Proceedings of the 12th International Conference on Computational Linguistics, COLING '88, Budapest, Hungary, August 22-27, 1988*, pages 431–436.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2923–2934.

Aurko Roy and David Grangier. 2019. Unsupervised paraphrasing without translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6033–6039.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1235–1246.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 38–42.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: Paraphrase generation with semantic augmentation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7176–7183.

Vitalii Zhelezniak, April Shen, Daniel Busbridge, Aleksandar Savkov, and Nils Hammerla. 2019. Correlations between word vector sets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 77–87.

Contrastive Fine-tuning Improves Robustness for Neural Rankers

Xiaofei Ma **Cicero Nogueira dos Santos** **Andrew O. Arnold**
AWS AI Labs
{xiaofeim, cicnog, anarnld}@amazon.com

Abstract

The performance of state-of-the-art neural rankers can deteriorate substantially when exposed to noisy inputs or applied to a new domain. In this paper, we present a novel method for fine-tuning neural rankers that can significantly improve their robustness to out-of-domain data and query perturbations. Specifically, a contrastive loss that compares data points in the representation space is combined with the standard ranking loss during fine-tuning. We use relevance labels to denote similar/dissimilar pairs, which allows the model to learn the underlying matching semantics across different query-document pairs and leads to improved robustness. In experiments with four passage ranking datasets, the proposed contrastive fine-tuning method obtains improvements on robustness to query reformulations, noise perturbations, and zero-shot transfer for both BERT and BART-based rankers. Additionally, our experiments show that contrastive fine-tuning outperforms data augmentation for robustifying neural rankers.

1 Introduction

Recent advances in neural language modeling have shifted the paradigm of natural language processing (NLP) towards a two-stage process: pre-training on a large amount of data with self-supervised tasks followed by fine-tuning on the target datasets with task-specific loss functions. Current state-of-the-art neural rankers for information retrieval fine-tune pre-trained language models using ranking losses on datasets containing examples of positive and negative query-document pairs. While usually achieving good performance on in-domain test sets, neural rankers trained on large datasets can still exhibit poor transferability when tested in new domains, and suffer from robustness problems when exposed to various types of perturbations. For example, a neural ranker trained on a dataset with mostly natural language queries can perform badly when tested on keyword queries which are very common in information retrieval (Bhatia et al., 2020).

A considerable number of previous works have focused on domain adaptation to improve model 's overall transferability. While domain adaptation approaches can help to address the out-of-domain robustness problem (Pan and Yang, 2010; Zhang et al., 2019; Ma et al., 2019), they rely on the availability of either labeled data or at least a target corpus which is usually not available at training time for a neural ranking model deployed in the wild.

The vulnerability of deep NLP models to various forms of adversarial attacks such as word-importance-based replacement (Jin et al., 2020), human-curated minimal perturbations (Khashabi et al., 2020), misspelling (Sun et al., 2020), grammatical errors (Yin et al., 2020), rule-based perturbations (Si et al., 2020; Ribeiro et al., 2018) is well-documented in the literature (Emma Zhang et al., 2019). While various methods have been proposed to remediate model robustness issues in NLP, most of them are either task-specific (Shah et al., 2019; Zhou et al., 2020; Gan and Ng, 2020; Wang and Bansal, 2018), requiring auxiliary tasks (Zhou et al.), or relying on data augmentation (Min et al., 2020; Kaushik et al., 2019; Cheng et al., 2020; Wei and Zou, 2019) which highly depends on the quality and diversity of the perturbed data.

An alternative strategy for optimizing machine learning models that has the potential to improve both out-of-domain generalization and robustness is contrastive learning. Representations obtained under contrastive self-supervised settings have demonstrated improved robustness to out-of-domain distributions and image corruptions in computer vision tasks (Hendrycks et al., 2019; Radford et al., 2021). In contrastive learning, representa-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 570–582
August 1–6, 2021. ©2021 Association for Computational Linguistics

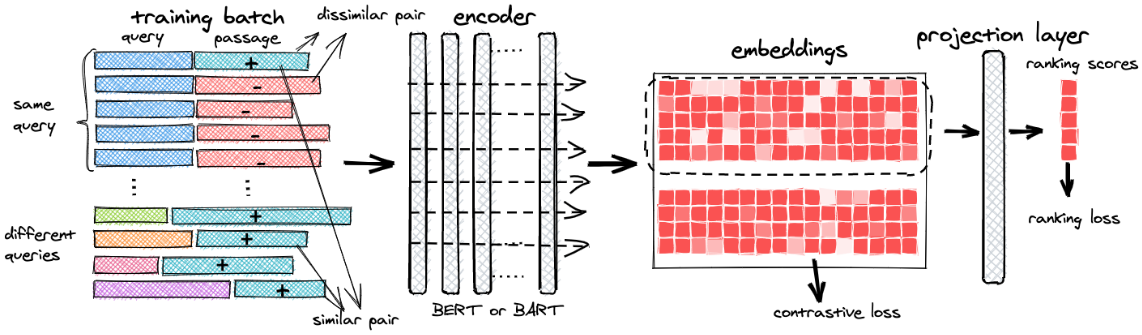

Figure 1: Contrastive fine-tuning for neural rankers. During fine-tuning, a batch of positive and negative samples from different queries is fed into a neural encoder. The embeddings of query-document pairs from the same query are used to generate ranking scores, which are employed to compute the ranking loss. In parallel, the embeddings of all pairs are used to compute the contrastive loss.

tions are learned by comparing among similar and dissimilar samples (Le-Khac et al., 2020; Khosla et al., 2020; Van Den Oord et al., 2018; Hjelm et al., 2018). This is different from discriminative learning, where models learn a mapping of input samples to labels, and generative learning, where models reconstruct input samples. While several works have investigated contrastive learning for sentence classification (Gunel et al., 2020), sentence representation learning (Wu et al., 2020), and multi-modal representation learning (Radford et al., 2021) under either self-supervised or supervised settings, their potential for improving the robustness of neural rankers has not been explored yet.

In this paper, we propose a novel contrastive learning approach to fine-tune neural rankers and investigate its benefits for improving model robustness. We focus on rankers that use single-tower architectures and are normally trained by optimizing a ranking loss that compares scores of positive and negative query-document pairs involving the same query. We propose to additionally use a contrastive loss that compares the distance between the representation of positive and negative pairs involving distinct queries (i.e, representations of positive pairs should be close in the latent space and distant to the representation of negative pairs, and vice-versa). The goal of using this contrastive loss in addition to the ranking loss is to stimulate the model to learn the underlying matching semantics across different query-document pairs, which can potentially lead to improved robustness.

Our main contributions are as follows:

- We propose to combine contrastive loss with ranking loss during fine-tuning of neural ranking models and investigate its impact in improving model robustness and generalization.
- Our experimental results using two language model-based neural rankers (BERT and BART) on four different datasets indicate that our proposed method improves upon standard ranking loss in zero-shot transfer across domains, leading to an increase of up to 9 absolute points in Mean Average Precision (MAP).
- We develop new datasets for evaluating the robustness of neural rankers. The datasets are based on WikiQA test set (Yang et al., 2015) and were created semi-automatically. We plan to release these datasets upon acceptance of the paper.
- We show that contrastive fine-tuned rankers are robust to 1) different types of query reformulations commonly seen in information retrieval (headline, paraphrase, and change of voice); and 2) query perturbations such as adding/removing punctuations, typos, and contractions/expansions.

2 Contrastive Representation Learning for Neural Ranking

In neural ranking models, given an input query q and a set of candidate documents $\{d_0, d_1, ..., d_n\}$, a neural network h is used to create vector representations $\{h(q, d_1), ..., h(q, d_n)\}$, which are given to a function $s : \overrightarrow{x} \rightarrow \mathbb{R}$, that computes a score for each query-document pair, $\{s(h(q, d_1)), ..., s(h(q, d_n))\}$. Normally s performs a simple linear projection of the input embedding, and the training of neural ranking models consists in optimizing a ranking loss that tries to enforce $s(h(q,d^+)) > s(h(q,d^-))$ for each training query q, where d^+ is a positive document for q while d^- is a negative one (See top part of Fig. 1).

We propose to augment the training of neural rankers with the use of contrastive representation learning. While ranking-based methods compute the loss with respect to the predicted scores, contrastive losses measure the distance/similarity between similar and dissimilar samples in the representation space. In our case, the key idea consists in using a loss that compares the distance between the representation of query-document pairs, and enforces that positive pairs are close together in the latent space while being far apart from negative pairs, i.e., $D(h(q,d^+), h(q',d'^+)) < D(h(q,d^+), h(q,d^-))$, where q' is either a variation of q or a completely different query, and d'^+ is a positive document for q'. Figure 1 illustrates our proposed approach, which is detailed in the remainder of this section.

2.1 Ranking Loss

Popular ranking losses include 1) the *pairwise ranking loss*, in which the relevance information is given in the form of preferences between pairs of candidates, and 2) the *listwise ranking loss* which directly optimizes a rank-based metric. In this work, we experiment with two pairwise ranking losses. The first one is the *standard hinge loss* (SHL) defined on a triplet (q, d^+, d^-) as follows:

$$\mathcal{L}_{SHL}(q,d^+,d^-;\theta) = \max\{0, \lambda - s(h(q,d^+);\theta) + s(h(q,d^-);\theta)\} \quad (1)$$

The other is a *modified hinge loss* (MHL) function defined as:

$$\mathcal{L}_{MHL}(q,d^+,\{d_i^-\};\theta) = \max\{0, \lambda - s(h(q,d^+);\theta) + \max_i\{s(h(q,d_i^-);\theta)\}\} \quad (2)$$

where q is a query, λ is the margin of the hinge loss, d^+ refer to the positive document. d^- and $\{d_i^-\}$ refer to a negative document and the list of negative documents of the query q within the same batch, respectively. θ includes the set of parameters of the network h and the projection layer in s. Based on preliminary experiments, our modified ranking hinge loss generally performs better than the standard pairwise ranking hinge loss. Note that MHL loss has been used in previous work on passage ranking (dos Santos et al., 2016).

2.2 Contrastive Loss

For contrastive learning of representations, we employ the conceptually simple but widely adopted *triplet margin loss* (**TML**) (Weinberger et al.; Chechik et al., 2010), which has the following form:

$$\mathcal{L}_{TML}(a,k^+,k^-;\theta) = \max\{0, m + D(a,k^+;\theta) - D(a,k^-;\theta)\} \quad (3)$$

where a is the anchor point, k^+ and k^- are the similar and dissimilar samples with respect to the anchor point a. m is the margin of the TML loss. In our neural ranking setting, an anchor point is the representation of a query-document pair. We use Euclidean or L2 distance D in our experiment. The contrastive loss can be applied to the representations from a variety of encoders $h(\cdot) \in R^d$. In this work, we explore contrastive fine-tuning for both BERT (Devlin et al., 2018) and BART (Lewis et al.) models.

The key to effective contrastive learning is to design the *notion of similarity* such that positive pairs may be very different in the input space yet semantically related. In this work, we leverage the relevance label in the training data and consider as similar positive pairs (q_i, d_i^+) and (q_j, d_j^+) from different queries i and j in the same batch (as illustrated in Fig. 1). Our intuition is that, by enforcing that positive pairs are close together in the embedding space and distant from negative pairs, we make the scoring task easier. Additionally, it allows the model to learn the underlying matching semantics across different query-document pairs, which leads to improved robustness. We additionally conduct a brief experiment in Sec. 5.4 where we use paraphrases of the original query to generate similar pairs.

2.3 Combined Loss

Our final loss is a weighted average of the ranking loss $\mathcal{L}_{ranking}$ and the contrastive loss $\mathcal{L}_{contrastive}$:

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{ranking} + w_2 \cdot \mathcal{L}_{contrastive} \quad (4)$$

The weights w_1 and w_2 are hyper-parameters that need to be determined. Our main experiments use a simple but effective combination method which consists in given equal weights to the ranking loss and contrastive loss.

3 Related Work

Our work is related to the recent body of works that demonstrate contrastive and self-supervised approaches can improve model robustness and generalization. Hendrycks et al. (2019) have shown that self-supervision increases image classifier's robustness to adversarial examples, label corruption, and common input corruptions. Radford et al. (2021) have demonstrated that multi-modal contrastive learning can significantly improve the robustness of image classifiers to distribution shift.

In the NLP space, some recent works on sentence-level contrastive representation learning have shown its potential to improve robustness for classification (Gunel et al., 2020) and semantic text similarity tasks (Wu et al., 2020). There are two main distinctions between our work and these two papers: 1) they focus on classification and text similarity tasks, while we focus on ranking; and 2) while they rely on data augmentation approaches to define the notion of similarity, our approach mainly relies on document relevance information which is already present in the training data.

Our work is also related to recent work on neural retrieval that focus on hard negative mining to improve model performance (Gillick et al., 2019; Xiong et al., 2020; Karpukhin et al., 2020; Lu et al., 2020). The main differences between our work and this line of research are: 1) while we leverage relevance information across different queries to create a notion of similarity, the focus on those papers are on finding hard negatives for each individual query in order to improve training efficiency. Hard negative mining can actually be used together with our method, as we show in Sec. 6.2. 2) we focus in reranking models, which use single-tower model that create a single representation for a query-document pair. In contrast, neural retrieval models create separate representations for query and document.

4 Experimental Setup

In this section, we describe the details of our experimental setup.

4.1 Passage Ranking Datasets

We test our method on four publicly available passage ranking/answer selection datasets that vary in size and domain. Passage ranking is an important task in information retrieval. It is often used to retrieve relevant content for open-domain question-answering systems (Wang et al., 2020).

WikiQA (Yang et al., 2015) is a dataset of question and sentence pairs, collected and annotated for research on open-domain question answering. The questions are factoid and selected from Bing query logs. The answers are in the summary section of a linked Wikipedia page. The candidates are retrieved using Bing.

WikiPassageQA (Cohen et al., 2018) is a benchmark collection for the research on non-factoid answer passage retrieval. The queries are created from Amazon Mechanical Turk over the top 863 Wikipedia documents from the Open Wikipedia Ranking.

InsuranceQA (Feng et al., 2016) The question and answer pairs from this dataset are collected from the internet in the insurance domain. Each question has an answer pool of 500 candidates retrieved using SOLR.

YahooQA (Tay et al., 2017) contains questions and answers from Yahoo! Answers website. The dataset is a subset of the Yahoo! Answers corpus from a 10/25/2007 dump. The questions are selected for their linguistic properties. For example, they all start with how $\{to \mid do \mid did \mid does \mid can \mid would \mid could \mid should\}$.

The statistics of the datasets are presented in Table 1. All four datasets provide validation sets, which have size similar to the respective test sets.

Dataset	Domain	Train: #Q (#P/Q)	Test: #Q (#P/Q)
WikiQA	Wikipedia	873 (9)	243 (9)
WikipassageQA	Wikipedia	3,332 (58.3)	416 (57.6)
InsuranceQA	insurance	12,889 (500)	2,000 (500)
YahooQA	community	50,112 (5)	6,283 (5)

Table 1: Dataset statistics. #Q stands for *number of questions* and #P/Q is the *average number of passages per question*

4.2 Datasets for Robustness Assessment

In order to assess the robustness of our models to different types of query reformulations and query perturbations, we built robustness test datasets based on the original WikiQA test set.

We assessed query perturbations by leveraging CheckList (Ribeiro et al., 2020) to construct three types of popular perturbations: *adding/removing punctuation*, *introducing typos* and *changing of contraction form*. For each query in WikiQA test set, we produce three new versions of the query, one for each perturbation type.

We assessed robustness to three types of query reformulations: *paraphrase*, *headline* and *change of voice*. We semi-automatically created the datasets for query reformulations in two steps: (1) a pretrained T5-base model (Raffel et al., 2019) is fine-tuned on a combination of large public paraphrase datasets (Quora [1] and PAWS (Zhang et al., 2019)) and human-curated query reformulations. The human-curated reformulations are based on the queries from the SQuAD 1.1 official dev set [2]. For each query in SQuAD 1.1 dev set, the annotators are asked to generate three new versions of the query (one for each reformulation type). During fine-tuning and inference, we use control codes to instruct T5 on the type of reformulation to be generated. Note that this T5 model can be also used for the purpose of data augmentation, as shown in Sec. 5.4. (2) each query in WikiQA test set is processed by the fine-tuned T5 and three reformulations of the query are generated. All generated queries are post-processed in order to ensure it is grammatically correct and semantically equivalent to the original query. To ensure reliable evaluation, we did a round of human annotations to filter out low-quality generations. Examples of query reformulations are presented in Table 2.

We evaluate the lexical diversity of generated query reformulations by computing the BLEU scores between the original query and the reformulated query. The results of comparing four different generation methods are presented in Figure 2. Our T5 generated queries overall exhibit higher diversity than human generated and back translation generated paraphrases. Note that the lower the BLEU score the higher the diversity.

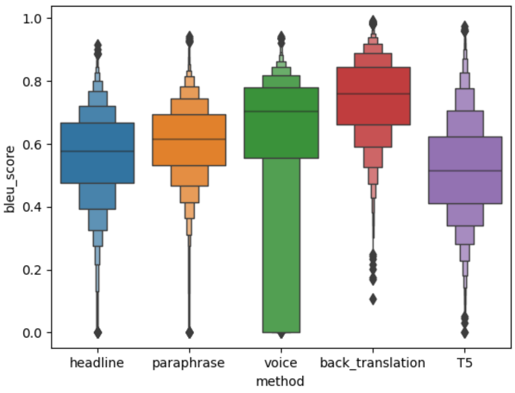

Figure 2: Comparison of BLEU scores between original query and reformulations generated by human annotation, back translation and fine-tuned T5 model.

Query	*What fueled the economy of early Vancouver?*
Headline	Factors that fuel Vancouver's economy
Paraphrase	What were some factors that stimulated the early Vancouver's economy?
Chg of Voice	The economy of early Vancouver was fueled by what?
Query	*How was the enlightenment shaped by science of the time period?*
Headline	type of science shaping enlightenment
Paraphrase	How was science of the time influenced the Enlightenment?
Chg of Voice	How did science of the time frame shape the Enlightenment?
Query	*How did South America gain independence from Spain and Portugal?*
Headline	Process of independence of South America from Spain and Portugal
Paraphrase	How did South America come to independence from Spain and Portugal?
Chg of Voice	How was independence from Spain and Portugal gained by South America?

Table 2: Examples of T5 generated styled paraphrases

4.3 Neural Ranker Training

We train neural rankers by fine-tuning two pretrained language models: BERT and BART. For fine-tuning BERT, we use BERT-base model (12 layers, 110M parameters) from Huggingface's transformer codebase (Wolf et al., 2019). Similar to the setup of sentence pair classification task in (Devlin et al., 2018), we concatenate the query sentence and the candidate passage together as a single input to the BERT encoder. We compute both the contrastive loss and ranking scores based on the [CLS] token embedding of the final hidden layer. For BART model fine-tuning, we use a BART-base model (6 layers encoder, 6 layer decoder, 139M parameters). We adopt the setting of BART for classification task in (**?**). The concatenation of query text and passage text is fed into both the encoder and decoder and the last layer's hidden state of the end decoder token is fed into a linear scorer. Similar to the [CLS] token in BERT, the embedding of the end token from the decoder is used as the representation of the complete input. For training with SHL, we sample triplets (q, d^+, d^-) from different queries to form a single batch. For MHL training, a single batch consists of a positive passage d^+ and a list of negative passages $\{d_i^-\}$ from the same query q. We leverage the toolkit developed by Musgrave et al. (2020) for contrastive loss calculation, and fine-tune the models for a maximum of 10 epochs and adopt early stopping using the validation sets of each dataset. The hyper-parameters for fine-tuning neural rankers are listed in Appendix A.

5 Results and Discussion

5.1 In-Domain Fine-tuning

The results of in-domain fine-tuning of BERT and BART-based neural rankers on four passage ranking datasets are presented in table 3. To ensure a fair comparison, all the hyper-parameters between the ranking and the contrastive settings are kept the same and equal weights between ranking loss and

---

[1] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

[2] https://rajpurkar.github.io/SQuAD-explorer/

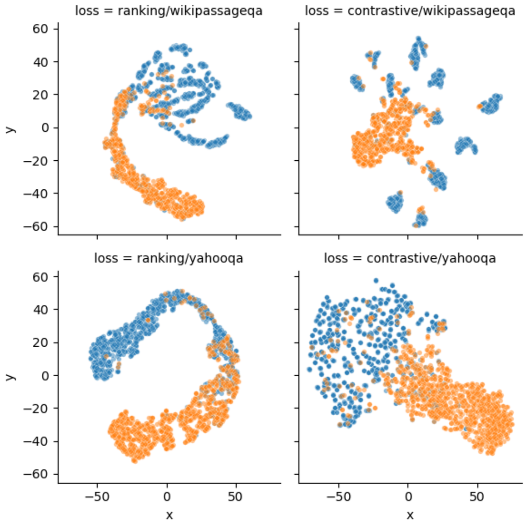

Figure 3: t-SNE plot of representations from BERT-based rankers trained with either ranking or combined ranking and contrastive loss for samples from WikiPassageQA (top) and YahooQA (bottom) test sets. Positive samples are represented by orange color.

contrastive loss are used. Our rankers produce state-of-the-art results when training with either of the two ranking-based losses **MHL** and **SHL**. Adding contrastive loss (**TML**) slightly improves the in-domain performance for BERT-based rankers, and performs similarly as ranking loss for BART-based rankers. Since our modified hinge loss (MHL) generally performs better than standard hinge loss (SHL), most of the results presented in the following sections are based on MHL, which corresponds to the setting illustrated in Figure 1.

To illustrate the effect of the contrastive loss on the representation space, we present the t-SNE plot of sample representations from the test set of two datasets WikiPassageQA and YahooQA in Figure 3. The color in the figures represents the positive and negative labels of query-passage pairs. As we can see from the plots, adding contrastive loss enables further separation of the positive samples from the negative samples.

5.2 Zero-Shot Transfer

The zero-shot transfer performance of neural rankers reflects their robustness to out-of-domain distributions, which is a key property of neural rankers since they are usually deployed in the wild. In Table 4, we show the results of applying the models trained in each one of the four datasets (source) and applied to the other three datasets (targets). Overall, we see significant improvements in the zero-shot transferability of the model across all datasets when the neural ranker is trained using the combination of ranking and contrastive losses. The biggest improvement is from YahooQA → WikiPassageQA where we observe absolute 9 points, 10.6 points, and 11.8 points improvement in MAP, MRR, and P@1, respectively. As expected, the transfer between datasets from similar domains tends to be better (e.g. WikiQA ↔ WikiPassageQA) than that between dissimilar domains. Our intuition regarding the benefit of contrastive learning of representations to improve zero-shot transfer consists on the fact that, by using information from different queries and enforcing that positive pairs are close together in the embedding space and distant from negative pairs, the model ends up learning representations that are more general and therefore easier to transfer to new domains.

5.3 Robustness to Query Perturbations

In this section, we evaluate the model robustness to various types of reformulations and noisy transformations of the input queries. The test sets used in the experiments are the 6 variations of the WikiQA test set described in Sec. 4.2. We compare the results of using ranking loss (MHL) and the combination of ranking and contrastive loss (MHL+TML). The robustness evaluation is are presented in Table 5. As shown in Table 5, adding contrastive loss improves model robustness against all types of perturbations we tested. We also conduct experiments by fine-tuning neural rankers on combined SHL loss and TML loss. The robustness evaluation of BERT-based rankers trained on SHL loss or combined SHL loss and TML loss are presented in Table 6. Similar to the MHL case, the combined loss achieves a significant improvement in robustness than SHL loss only. More results on model robustness can be found in Appendix B.

5.4 Comparison with Data Augmentation

One of the traditional approaches for improving the robustness of machine learning models is to augment the training data with noisy data. In this section, we compare our contrastive fine-tuning method with a data augmentation approach in which automatically generated query reformulations are added to the training data. For each query in the training set, we use our fine-tuned T5 model to generate 5 new queries of each reformulation type (headline, paraphrase, change of voice). Effec-

Dataset →	WikiQA			WikiPassageQA			InsuranceQA			YahooQA		
Model	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1
BERT												
BERTSel-base (2019)	75.3	77.0	-	-	-	-	-	-	-	94.2	94.2	-
BERT-PR-base (2019)	-	-	-	73.5	80.9	70.2	41.3	49.6	40.1	-	-	-
BERT-base-MHL (ours)	82.1	84.0	74.5	76.3	83.0	73.6	39.4	47.4	37.9	96.2	96.1	93.4
BERT-base-MHL+TML (ours)	83.8	85.8	77.4	76.9	83.1	73.6	41.1	49.6	39.5	96.1	96.1	93.4
BERT-base-SHL (ours)	82.3	84.1	75.7	74.2	81.2	71.6	40.0	47.6	37.3	95.9	95.9	92.9
BERT-base-SHL+TML (ours)	82.6	84.5	76.1	74.7	81.1	70.0	40.1	47.5	36.9	95.8	95.8	92.8
BART												
BART-base$_{LUL}$ (2020)	77.8	78.8	65.8	73.8	81.3	71.9	44.0	52.6	43.4	92.8	92.8	87.6
BART-base$_{RLL}$ (2020)	77.5	79.2	65.4	76.1	83.4	74.3	42.2	50.3	40.8	96.1	96.1	93.4
BART-base-MHL (ours)	85.8	87.4	78.2	77.8	85.3	77.6	43.5	51.8	42.0	96.5	96.5	94.0
BART-base-MHL+TML (ours)	84.6	86.1	75.7	77.4	84.5	76.2	43.4	51.9	42.4	96.5	96.5	93.9
BART-base-SHL (ours)	82.4	83.9	73.3	75.9	83.1	73.6	42.9	51.2	41.6	96.0	96.0	93.1
BART-base-SHL+TML (ours)	81.6	82.6	70.0	75.1	82.1	71.9	42.9	51.3	41.4	96.5	96.5	93.8

Table 3: In-domain results of neural rankers trained on ranking loss vs contrastive loss.

	Target →	WikiQA			WikiPassageQA			InsuranceQA			YahooQA		
Source Domain	**Loss**	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1
	BERT												
WikiQA	MHL	-	-	-	58.3	66.4	51.7	7.3	8.8	3.2	49.4	49.3	26.7
WikiQA	MHL+TML	-	-	-	58.4	66.3	51.0	11.6	14.9	7.5	50.1	50.1	27.9
WikiPassageQA	MHL	72.4	73.2	60.1	-	-	-	30.1	36.8	26.9	57.5	57.5	37.8
WikiPassageQA	MHL+TML	73.8	74.9	61.7	-	-	-	30.8	37.8	28.0	59.5	59.5	40.1
InsuranceQA	MHL	59.6	60.4	41.6	56.6	65.4	52.9	-	-	-	69.2	69.2	52.0
InsuranceQA	MHL+TML	61.1	62.3	45.3	57.2	65.1	52.2	-	-	-	78.0	78.0	64.3
YahooQA	MHL	35.3	36.3	16.5	30.2	32.5	15.6	3.8	4.4	0.9	-	-	-
YahooQA	MHL+TML	38.7	39.5	18.1	39.2	43.1	27.4	5.5	6.2	1.7	-	-	-
	BART												
WikiQA	MHL	-	-	-	52.4	58.9	41.4	10.7	13.6	6.7	51.8	51.8	30.0
WikiQA	MHL+TML	-	-	-	58.6	67.1	51.9	14.0	17.7	9.7	52.4	52.4	30.9
WikiPassageQA	MHL	74.6	75.9	63.0	-	-	-	29.0	35.8	23.9	62.6	62.6	44.7
WikiPassageQA	MHL+TML	76.5	78.1	66.3	-	-	-	30.8	37.8	28.0	63.8	63.8	45.7
InsuranceQA	MHL	64.9	66.1	50.2	62.5	70.8	59.1	-	-	-	69.1	69.1	51.6
InsuranceQA	MHL+TML	65.5	66.3	50.6	63.3	72.2	61.8	-	-	-	71.7	71.7	55.4
YahooQA	MHL	34.1	35.4	15.6	16.2	18.6	7.0	2.2	2.7	0.6	-	-	-
YahooQA	MHL+TML	35.3	36.2	17.3	18.3	21.3	8.5	2.5	3.2	0.6	-	-	-

Table 4: Results of zero-shot transfer for models trained on ranking loss vs. contrastive loss.

	Reformulation →	Headline		Paraphrase		Chg of Voice		Punctuation		Typo		Contraction	
Training Data	**Loss**	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1
	BERT												
WikiQA	MHL	75.3	64.9	79.4	69.3	72.6	60.0	82.3	74.5	76.8	66.3	82.6	77.6
WikiQA	MHL+TML	76.7	66.5	81.8	73.4	77.5	68.0	83.0	75.3	79.4	71.3	84.3	77.6
WikiPassageQA	MHL	61.8	45.5	67.3	51.9	64.2	48.0	71.7	58.9	57.1	39.2	74.7	64.3
WikiPassageQA	MHL+TML	63.3	47.5	68.3	53.5	68.9	56.0	72.3	60.1	58.8	40.8	74.9	64.3
	BART												
WikiQA	MHL	80.6	69.4	84.3	75.9	76.8	64.0	83.8	74.5	82.5	73.8	86.3	79.6
WikiQA	MHL+TML	81.1	70.7	85.9	78.0	79.2	72.0	86.0	79.0	83.7	75.4	86.5	77.6
WikiPassageQA	MHL	72.0	58.7	75.3	62.7	73.4	60.0	74.6	62.6	70.0	55.8	79.1	71.4
WikiPassageQA	MHL+TML	74.3	62.0	76.1	63.5	76.0	64.0	77.3	67.1	69.3	56.7	79.6	70.4

Table 5: Robustness to various types of query reformulations and perturbations for rankers with MHL loss.

	Reformulation →	Headline		Paraphrase		Chg of Voice		Punctuation		Typo		Contraction	
Training Data	**Loss**	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1
WikiQA	SHL	73.1	62.0	79.6	71.0	74.9	64.0	81.0	72.4	74.7	63.3	80.8	73.5
WikiQA	SHL+TML	75.2	64.5	80.6	71.4	80.9	72.0	81.7	74.5	79.9	71.3	82.0	75.5
WikiPassageQA	SHL	60.4	45.5	66.9	51.9	54.6	40.0	68.9	54.7	55.1	36.7	69.6	55.1
WikiPassageQA	SHL+TML	62.5	46.3	66.9	51.9	62.7	48.0	68.9	54.7	56.9	39.2	72.0	58.2

Table 6: Robustness to various types of paraphrase for BERT-based rankers trained with SHL.

		Reformulation →	Headline			Paraphrase			Change of Voice		
Loss	**Notion of Similarity**	**Training Data**	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1
MHL	-	WikiQA	75.3	76.9	64.9	79.4	80.9	69.3	72.6	73.6	60.0
MHL+TML	relevance label	WikiQA	76.7	78.0	66.5	81.8	83.2	73.4	77.5	78.6	68.0
MHL	-	WikiQA + headline	76.1	77.6	63.6	78.2	79.7	66.4	73.8	75.2	60.0
MHL+TML	query reformulations	WikiQA + headline	77.1	78.6	65.7	80.2	81.4	71.0	81.2	83.7	76.0
MHL	-	WikiQA + paraphrase	70.0	71.0	54.6	81.3	82.6	71.8	78.9	80.3	68.0
MHL+TML	query reformulations	WikiQA + paraphrase	75.1	76.4	63.2	81.2	82.4	70.5	80.3	83.1	76.0
MHL	-	WikiQA + chg voice	73.6	75.0	60.7	82.8	83.9	73.4	81.2	84.5	76.0
MHL+TML	query reformulations	WikiQA + chg voice	75.3	76.6	63.2	83.0	84.3	73.4	83.3	88.5	80.0

Table 7: Comparison with data argumentation.

tively, we increase the training set and the number of training steps by a factor of 5 for each reformulation type. Since our proposed training approach is general and can be used with any dataset, we also experiment with data augmentation combined with contrastive fine-tuning. When performing contrastive fine-tuning, we use the 5 reformulations of each query to create similar pairs (the notion of similarity == query formulation) in each batch, which essentially keeps the number of training steps the same as when training with the original dataset. The results on data augmentation for BERT-based neural rankers are presented in Table 7. For rows with MHL loss, we argument the training data with paraphrased queries and train the model on a combined dataset using MHL loss only. Note this is the standard way to do data argumentation training. For rows using MHL+TML loss, we pair each query in the batch with its paraphrased query for contrastive loss calculation. The model is trained on combined MHL+TML loss. For both methods, we expose the model with the same amount of paraphrased training samples. As we can see from the table, augmenting the training data with a similar type of query reformulation can improve the robustness of the model against that particular type of reformulation. However, it is not as effective in improving the robustness against other reformulation types. On the other hand, contrastive fine-tuning, even trained with a single type of query reformulation can generally improve the model robustness against the other two types. Furthermore, contrastive fine-tuning achieves this with significantly less (4× less) training time even after considering the additional computation of the contrastive loss calculation. Essentially, the experimental results indicate that using paraphrased training samples to perform contrastive learning is both effective (produces more robust rankers) as well as efficient (faster to train since augmented data is used in parallel, i.e. same batch as original data) than using regular data augmentation.

5.5 Comparison with Ranking Loss using same Batch Size

When training with MHL plus contrastive loss we effectively increase the batch size because we need to augment the training batch with additional positive samples from different queries. To check if the improvements achieved by our approach are due to the increase of training batch size only, we perform an ablation study where we compare the performance of models trained with a ranking loss but with the same batch size as the contrastive fine-tuning setting. The comparison results are presented in Table 8. As we can see in the table, for WikiQA datase, increasing the batch size helps the performance of in-domain and some of the robustness test sets. The contrastive setting still outperforms the ranking setting in all the test categories. Increasing the batch size of MHL is not always beneficial. We see big degradation on the WikiPassageQA dataset. On the other hand, we observed consistent improvement when the model is trained with contrastive loss.

6 Ablation Study

We present ablation experiments that check the impact of the number of positive samples per batch and the use of hard negative mining. Additionally, in Appendix C, we also present a preliminary experiment on formulating our combined loss (Equation 4) as a multi-objective optimization (MOO).

6.1 Effect of Number of Positive Samples Per Batch

The number of positive samples within a single batch determines the total number of potential triples constructed. In this section, we vary the number of positive samples within a batch and evaluate its effect on the model performance. The results are presented in Table 9. As expected, the model performance benefits by increasing the number of positives in a batch. As shown in Table 9, although not strictly monotonically, both the in-domain performance and zero-shot transfer performance improve with the number of positive pairs.

6.2 Effect of Hard Negative Mining

In a batch of N samples, there are $O(N^3)$ possible triplets, many of which are not very helpful to model convergence (e.g triplets where $D(a, k^+) >> D(a, k^-)$). It's important to construct only the most important triplets. Many works have discussed the benefit of hard negative mining techniques that produce useful gradients and help the models converge quickly. In this section, we explore the effect of three hard negative mining methods that are compatible with TML: Angular miner (Wang et al., 2017) (output triplets that form an angle greater than a threshold), BatchHard (Hermans et al., 2017) (produce a single triplet for each

		Test Set →	In-domain		Headline		Paraphrase		Chg of Voice	
Loss	**Batch Size**	**Training Set**	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1
MHL	16	WikiQA	82.1	74.5	75.3	64.9	79.4	69.3	72.6	60.0
MHL	31	WikiQA	83.0	76.1	74.9	64.1	81.0	71.8	77.2	68.0
MHL+TML	31	WikiQA	83.8	77.4	76.7	66.5	81.8	73.4	77.5	68.0
MHL	16	WikiPassageQA	76.3	73.6	61.8	45.5	67.3	51.9	64.2	48.0
MHL	31	WikiPassageQA	76.2	73.8	58.7	40.5	64.4	47.7	57.1	40.0
MHL+TML	31	WikiPassageQA	76.9	73.6	63.3	47.5	68.3	53.5	68.9	56.0

Table 8: Comparison of ranking loss and contrastive loss with same batch size for BERT-based rankers.

Target →	**WikiQA**			**WikiPassageQA**			**InsuranceQA**			**YahooQA**		
# Pos / Batch	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1
2	82.4	84.1	74.9	58.1	66.4	51.7	12.8	16.3	8.0	46.4	46.3	24.2
4	83.4	85.2	76.1	58.1	66.1	51.0	12.1	15.3	7.0	47.9	47.8	26.0
8	83.5	85.3	75.7	59.9	67.4	52.9	12.1	15.2	7.1	48.7	48.7	26.8
16	83.8	85.8	77.4	59.4	66.3	51.0	11.6	14.5	6.4	50.1	50.1	27.9
32	83.3	85.2	76.1	59.0	65.9	49.3	10.4	12.9	4.9	49.9	49.9	27.6
64	82.5	84.2	74.5	61.4	69.1	55.1	13.0	16.7	8.2	49.2	49.2	26.9

Table 9: Effect of number of positive pairs per batch. Underlined cells indicate in-domain results.

Target →	**WikiQA**			**WikiPassageQA**			**InsuranceQA**			**YahooQA**		
Mining Method	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1
BERT												
No Mining	83.8	85.8	77.4	59.4	66.3	51.0	11.6	14.9	7.5	50.1	50.1	27.9
Angular	83.8	85.6	76.1	59.1	66.2	51.0	6.7	8.1	3.1	49.7	49.6	27.4
BatchHard	82.3	84.0	73.7	63.0	70.7	57.0	10.7	13.9	6.4	52.0	52.0	30.2
TripletMargin	83.6	85.6	77.0	59.4	66.4	51.0	7.2	8.7	3.2	50.1	50.1	27.8
BART												
No Mining	84.6	86.1	75.8	58.6	67.1	51.9	14.0	17.7	9.7	52.4	52.3	30.9
Angular	84.5	86.1	76.1	59.9	67.9	53.1	16.0	20.0	11.3	56.4	56.4	35.6
BatchHard	85.4	86.7	77.0	57.8	66.2	50.5	12.5	15.8	7.9	55.3	55.3	33.9
TripletMargin	85.9	87.5	78.6	61.2	69.6	55.3	16.0	19.9	11.0	56.6	56.5	35.8

Table 10: Effect of hard negative mining. Underlined cells indicate in-domain results.

anchor point consisting of the hardest positive and hardest negative samples), and TripletMargin (only output a triplet when the difference between the anchor-positive distance and the anchor-negative distance is smaller than a margin). The results of hard negative mining on models trained on WikiQA dataset are presented in Table 10, in which we evaluate both the in-domain and zero-shot performance of the rankers. As we can see from the results, hard negative mining can further improve the transferability of of both BERT-based ranker and BART-based ranker. In particular, BatchHard outperforms other mining methods and improve the overall performance significantly for BERT-based rankers while TripletMargin is more effective for BART-based rankers. We believe there is still a margin for improvement if the hyper-parameters of the miners are properly tuned.

7 Conclusion

In this paper, we propose a novel method for fine-tuning neural rankers by combining contrastive loss with ranking loss. Using a semi-automatic approach, we created 6 new versions of WikiQA test set to assess the robustness of our models to query reformulations and perturbations. Our experimental results show that the proposed method improves ranker's robustness to out-of-domain distributions, query reformulations, and perturbations. Comprehensive experiments and ablation studies were conducted to investigate the impact of some design choices as well as to confirm that the gains do not originate only from larger batch sizes. Contrastive fine-tuning with generated data is more effective than data augmentation. As future work, we plan to evaluate the performance of other state-of-the-art contrastive loss functions and novel methods of aggregating multiple losses.

References

Parminder Bhatia, Lan Liu, Kristjan Arumae, Nima Pourdamghani, Suyog Deshpande, Ben Snively, Mona Mona, Colby Wise, George Price, Shyam Ramaswamy, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, Bing Xiang, and Taha Kass-Hout. 2020. AWS CORD-19 Search: A Neural Search Engine for COVID-19 Literature.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2020. Robust neural machine translation with doubly adversarial inputs. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 2018, pages 4324–4333.

Daniel Cohen, Liu Yang, and W Bruce Croft. 2018. WikiPassageQA: A Benchmark Collection for Research on Non-factoid Answer Passage Retrieval. In *SIGIR*, pages 1–4.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Cicero Nogueira dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1722–1727, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2019. Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey. 1(1):40.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2016. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015 - Proceedings*, pages 813–820. Institute of Electrical and Electronics Engineers Inc.

Wee Chung Gan and Hwee Tou Ng. 2020. Improving the robustness of question answering systems to question paraphrasing. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 6065–6075.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. 2019. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, volume 32.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person reidentification.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. pages 1–24.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Yaochu Jin, Tatsuya Okabe, and Bernhard Sendhoff. 2004. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, volume 1, pages 1–8.

Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 2001. Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: {W}hy Does It Work and How? In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 1042–1049.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2019. Learning the Difference that Makes a Difference with Counterfactually-Augmented Data.

Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020. Natural Perturbation for Robust Question Answering.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Dilip Krishnan, and Ce Liu. 2020. Supervised contrastive learning.

Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive Representation Learning: A Framework and Review. *IEEE Access*, 8:193907–193934.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Technical report.

Dongfang Li, Yifei Yu, Qingcai Chen, and Xinyu Li. 2019. BERTSel: Answer selection with pre-trained models.

Jing Lu, Gustavo Hernandez Abrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2020. Neural passage retrieval with improved negative contrast.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain Adaptation with BERT-based Domain Classification and Data Selection. pages 76–83. Association for Computational Linguistics (ACL).

Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. Syntactic Data Augmentation Increases Robustness to Inference Heuristics. pages 2339–2352.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. 2020. Pytorch metric learning.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

A. Radford, J. W. Kim, Chris Hallacy, Aditya Ramesh, G. Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, J. Clark, G. Krüger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 1, pages 856–865.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP models with CheckList. pages 4902–4912.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. 2019. Cycle-consistency for robust visual question answering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 6642–6651.

Chenglei Si, Ziqing Yang, Yiming Cui, Wentao Ma, Ting Liu, and Shijin Wang. 2020. Benchmarking robustness of machine reading comprehension models.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-BERT: BERT is not robust on misspellings! Generating nature adversarial samples on BERT.

Yi Tay, Minh C. Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual LSTM architecture. In *SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704. Association for Computing Machinery, Inc.

Aaron Van Den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding.

Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. 2017. Deep Metric Learning with Angular Loss. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 2612–2620. Institute of Electrical and Electronics Engineers Inc.

Yicheng Wang and Mohit Bansal. 2018. Robust Machine Comprehension Models via Adversarial Training. pages 575–581.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2020. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 5878–5882. Association for Computational Linguistics.

Jason W. Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. Technical report.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. CLEAR: Contrastive Learning for Sentence Representation.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval.

Peng Xu, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Passage ranking with weak supervision.

Yi Yang, Wen-Tau Yih, and Christopher Meek. 2015. WIKIQA: A Challenge Dataset for Open-Domain Question Answering. *Proceedings of EMNLP 2015*, (September 2015):2013–2018.

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the Robustness of Language Encoders against Grammatical Errors. pages 3386–3403.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I Jordan. 2019. Bridging theory and algorithm for domain adaptation. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 12805–12823.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2020. Robust Reading Comprehension with Linguistic Constraints via Posterior Regularization. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 28:2500–2510.

Shuyan Zhou, Xiangkai Zeng, Yingqi Zhou, Antonios Anastasopoulos, and Graham Neubig. Improving Robustness of Neural Machine Translation with Multi-task Learning. Technical Report 1.

A Details of Neural Ranker Fine-tuning

The fine-tuning of neural rankers is conducted on an AWS EC2 P3 machine. Important hyperparameters of fine-tuning for each model-dataset combination are listed in Table 11.

B More Results of Robustness Against Query Perturbations

In this section, we present more results of model robustness evaluation for neural rankers trained on InsuranceQA and YahooQA datasets. The results are shown in Table 12.

C Fine-tuning as Multi-objective Optimization

We performed a preliminary experiment on formulating equation 4 as a multi-objective optimization (MOO) problem in which optimizing both $\mathcal{L}_{ranking}$ and $\mathcal{L}_{contrastive}$ are two objectives of the task. We adopt a *dynamic weighted aggregation* (DWA) method (Jin et al., 2001, 2004) which is both effective and computationally efficient. In DWA, the weights of the two loss terms are changed gradually according to the following equations:

$$w_1(t) = |\sin 2\pi t/\ F| \tag{5}$$
$$w_2(t) = 1 - w_1(t) \tag{6}$$

where t is the iteration number. It is noticed that $w_1(t)$ changes from 0 to 1 periodically. The change frequency can be adjusted by F.

Figure 4 shows the evolution of the contrastive loss during fine-tuning of the BART-based ranker on the WikiQA dataset. As can be seen from the plot, adopting the MOO method improves the model convergence. A lower contrastive loss is achieved using dynamic weighting which translates to an average improvement of 0.7 points over the equal weighting setting in zero-shot transfer performance (see Table 13).

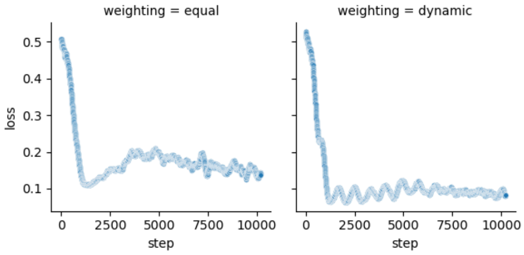

Figure 4: Training contrastive loss for neural ranker trained on WikiQA.

Training Data	Loss	Learning Rate	Positives / Batch	Negatives / Batch	Gradient Accumulation	Block Size	Hinge Loss Margin
				BERT			
WikiQA	MHL	5e-6	1	15	8	256	2
WikiQA	MHL+TML	5e-6	16	15	8	256	2
WikiPassageQA	MHL	1e-5	1	15	8	256	2
WikiPassageQA	MHL+TML	1e-5	16	15	8	256	2
InsuranceQA	MHL	5e-5	1	15	16	256	2
InsuranceQA	MHL+TML	5e-5	16	15	16	256	2
YahooQA	MHL	1e-5	1	4	8	256	2
YahooQA	MHL+TML	1e-5	16	4	8	256	2
WikiQA	SHL	5e-6	15	15	8	256	2
WikiQA	SHL+TML	5e-6	15	15	8	256	2
WikiPassageQA	SHL	1e-5	15	15	8	256	2
WikiPassageQA	SHL+TML	1e-5	15	15	8	256	2
InsuranceQA	SHL	5e-5	15	15	16	256	2
InsuranceQA	SHL+TML	5e-5	15	15	16	256	2
YahooQA	SHL	1e-5	4	4	8	256	2
YahooQA	SHL+TML	1e-5	4	4	8	256	2
				BART			
WikiQA	MHL	5e-6	1	15	8	256	2
WikiQA	MHL+TML	5e-6	16	15	8	256	2
WikiPassageQA	MHL	1e-5	1	15	8	256	2
WikiPassageQA	MHL+TML	1e-5	16	15	8	256	2
InsuranceQA	MHL	5e-5	1	15	16	256	2
InsuranceQA	MHL+TML	5e-5	16	15	16	256	2
YahooQA	MHL	1e-5	1	4	8	256	2
YahooQA	MHL+TML	1e-5	16	4	8	256	2
WikiQA	SHL	5e-6	15	15	8	256	2
WikiQA	SHL+TML	5e-6	15	15	8	256	2
WikiPassageQA	SHL	1e-5	15	15	8	256	2
WikiPassageQA	SHL+TML	1e-5	15	15	8	256	2
InsuranceQA	SHL	5e-5	15	15	16	256	2
InsuranceQA	SHL+TML	5e-5	15	15	16	256	2
YahooQA	SHL	1e-5	4	4	8	256	2
YahooQA	SHL+TML	1e-5	4	4	8	256	2

Table 11: Hyper-parameters for neural ranker fine-tuning.

	Reformulation →	**Headline**		**Paraphrase**		**Chg of Voice**		**Punctuation**		**Typo**		**Contraction**		
Training Data	**Loss**	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	MAP	P@1	**Avg MAP**
							BERT							
InsuranceQA	MHL	53.2	34.7	57.4	39.8	44.4	24.0	59.2	41.2	42.8	23.4	60.4	42.9	52.9
InsuranceQA	MHL+TML	54.9	35.1	58.9	41.5	47.9	28.0	60.6	44.4	44.4	25.0	60.0	42.9	**54.5**
YahooQA	MHL	40.5	22.3	34.0	14.1	28.8	4.0	36.7	16.9	34.0	14.6	38.3	16.3	35.4
YahooQA	MHL+TML	41.0	21.9	35.5	14.5	29.4	8.0	40.1	20.2	35.5	15.0	41.6	20.4	**37.2**
							BART							
InsuranceQA	MHL	61.2	46.3	64.2	50.2	52.6	32.0	63.6	48.2	54.7	37.9	68.6	54.1	60.8
InsuranceQA	MHL+TML	63.4	50.8	65.0	50.6	58.5	44.0	64.7	50.2	54.6	37.5	68.8	56.1	**62.5**
YahooQA	MHL	35.7	17.4	32.9	13.7	29.7	8.0	34.2	15.2	33.6	15.8	36.5	15.3	33.8
YahooQA	MHL+TML	37.2	18.2	34.4	15.8	30.7	8.0	35.2	16.9	34.6	17.5	37.3	17.4	**34.9**

Table 12: Additional restuls of robustness to various types of paraphrase.

Target →		**WikiQA**			**WikiPassageQA**			**InsuranceQA**			**YahooQA**		
Weighting Method	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1	MAP	MRR	P@1	**Avg MAP**
						BERT							
Equal Weighting	84.6	86.1	75.7	58.6	67.1	51.9	14.0	17.7	9.7	52.4	52.3	30.9	52.4
Dynamic Weighting	84.6	86.0	75.8	58.0	66.5	50.7	13.6	17.2	8.7	56.2	56.2	35.4	**53.1**

Table 13: Dynamic weighting vs equal weighting.

Cross-Lingual Transfer in Zero-Shot Cross-Language Entity Linking

Elliot Schumacher **James Mayfield** **Mark Dredze**
Johns Hopkins University
`eschumac@cs.jhu.edu` `mayfield@jhu.edu` `mdredze@cs.jhu.edu`

Abstract

Cross-language entity linking grounds mentions written in several languages to a monolingual knowledge base. We use a simple neural ranking architecture for this task that uses multilingual BERT representations of both the mention and the context as input, so as to explore the ability of a transformer model to perform well on this task. We find that the multilingual ability of BERT leads to good performance in monolingual and multilingual settings. Furthermore, we explore zero-shot language transfer and find surprisingly robust performance. We conduct several analyses to identify the sources of performance degradation in the zero-shot setting. Results indicate that while multilingual transformer models transfer well between languages, issues remain in disambiguating similar entities unseen in training.

1 Introduction

Entity linking grounds named entities mentioned in text, such as *Chancellor*, to a reference knowledge base (KB) or ontology entry, such as *Angela Merkel*. Historically, entity linking work focused on English documents and knowledge bases, but subsequent work expanded the task to consider multiple languages (McNamee et al., 2011). In cross-language entity linking, entities in a set of multilingual documents is linked to a KB in a single language. The TAC KBP shared task (Ji et al., 2015), for example, links mentions in Chinese and Spanish documents with an English KB. Success in building cross-language linking systems can be helpful in tasks such as discovering all documents relevant to an entity, regardless of language.

Successfully linking a mention across languages requires adapting several common entity linking components to the cross-language setting. Consider the example in Figure 1, which contains the Spanish mention *Oficina de la Presidencia*, a reference to the entity *President of Mexico* in an English KB. To link the mention to the relevant entity we must compare the mention text and its surrounding textual context in Spanish to the English entity name and entity description, as well as compare the mention and entity type. Previous work has focused on transliteration or translation approaches for name and context (McNamee et al., 2011; Pan et al., 2015), or leveraging large amounts of cross-language information (Tsai and Roth, 2016) and multilingual embeddings (Upadhyay et al., 2018).

Since this work emerged, there have been major advances in multilingual NLP (Wu and Dredze, 2019; Pires et al., 2019). Mainstream approaches to multilingual learning now use multilingual encoders, trained on raw text from multiple languages (Devlin et al., 2019). These models, such as multilingual BERT or XMLR (Conneau et al., 2019), have achieved impressive results on a range of multilingual NLP tasks, including part of speech tagging (Tsai et al., 2019), parsing (Wang et al., 2019; Kondratyuk and Straka, 2019), and semantic similarity (Lo and Simard, 2019; Reimers and Gurevych, 2019).

We propose to leverage text representations with multilingual BERT (Devlin et al., 2019) for cross-language entity linking to handle the mention text, entity name, mention context and entity description[1]. We use a neural ranking objective and a deep learning model to combine these representations, along with a one-hot embedding for the entity and mention type, to produce a cross-language linker. We use this ranking architecture to highlight the ability of mBERT to perform on this task without a more complex architecture. Although previous work tends to use multilingual encoders for one language at a time, *e.g.,* train a Spanish NER system

---

[1] Our code is available at `https://github.com/elliotschu/crosslingual-el`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 583–595
August 1–6, 2021. ©2021 Association for Computational Linguistics

with mBERT, we ask: can our model effectively link entities *across* languages? We find that, somewhat surprisingly, our approach does exceedingly well; scores are comparable to previously reported best results that are trained on data not available to our model (they have access to non-English names). Next, we consider a multilingual setting, in which a single system is simultaneously trained to link mentions in multiple languages to an English KB. Previous work (Upadhyay et al., 2018) has shown that multilingual models can perform robustly on cross-language entity linking. Again, we find that, surprisingly, a model trained on multiple languages at once does about as well, or in some cases better, than the same model trained separately on every language.

These encouraging results lead us to explore the challenging task of zero-shot training, in which we train a model to link single language documents (*e.g.,* English) to an English KB, but apply it to unseen languages (*e.g.,* Chinese) documents. While the resulting model certainly does worse on a language that is unobserved, the reduction in performance is remarkably small. This result leads us to ask: 1) Why do zero-shot entity linking models do so well? 2) What information is needed to allow zero-shot models to perform as well as multilingually trained models? Using a series of ablation experiments we find that correctly comparing the mention text and entity name is the most important component of an entity linking model. Therefore, we propose an auxiliary pre-training objective to improve zero-shot performance. However, we find that this text-focused approach does not improve performance significantly. Rather, we find that much of the remaining loss comes not from the language transfer, but from mismatches of entities mentioned across the datasets. This suggests that future work on the remaining challenges in zero-shot entity linking should focus on topic adaptation, instead of improvements in cross-lingual representations.

In summary, this paper uses a simple ranker to explore effective cross-language entity linking with multiple languages. We demonstrate its effectiveness at zero-shot linking, evaluate a pre-training objective to improve zero-shot transfer, and lay out guidelines to inform future research on zero-shot linking.

2 Cross-Language Entity Linking

A long line of work on entity linking has developed standard models to link textual mentions to entities in a KB (Dredze et al., 2010; Durrett and Klein, 2014; Gupta et al., 2017). The models in this area have served as the basis for developing multilingual and cross-language entity linking systems, and they inform our own model development. We define **multilingual** to mean a model that can operate on mentions from more than one language at the same time (link both English and Chinese mentions to an ontology) and **cross-language** to refer to linking mentions in one language (*e.g.,* Spanish) to an ontology in another (*e.g.,* English).

A common approach to cross-language entity linking is to use transliteration data to transform non-English mentions into English strings. Early transliteration work (McNamee et al., 2011) uses a transliteration corpus to train a support vector machine ranker, which uses common entity linking features such as name and context matching, co-occurring entities, and an indicator for NIL (no matching candidate.) Pan et al. (2017) uses transliteration data for a set of 282 languages to generate all possible combinations of mentions. A related approach is to use machine translation to translate a document into English, and then use an English entity linker. However, an MT system may not be available, and it further needs a specialized name module to properly translate entity names. Several systems from the TAC 2015 KBP Entity Discovery and Linking task (Ji et al., 2015) translate non-English documents into English, then use standard Entity Linking systems.

Cross-language Wikification is a closely related task, which uses links within Wikipedia, combined with equivalent pages in other languages to train an entity linker with Wikipedia as the KB. This approach typically uses English Wikipedia as the KB, though it could use a KB in other languages. Tsai and Roth (2016) use a two-step linking approach, first using an IR-based triage system (which we also use). Second, they use a candidate ranking step based on a linear ranking SVM model with several features, including contextual, document, and coreference.

The most closely related work to our own is that of Upadhyay et al. (2018), who use multilingual embeddings as the basis for their representations, and Wikipedia as training data. They use FastText (Bojanowski et al., 2017; Smith et al., 2017)

... el jefe de la **Oficina de la Presidencia** *(m.01p1k, ORG)*, Aurelio Nuño y ...

name	*President of Mexico* (m.01p1k)
desc.	*The President of the United ...*
type	government_office

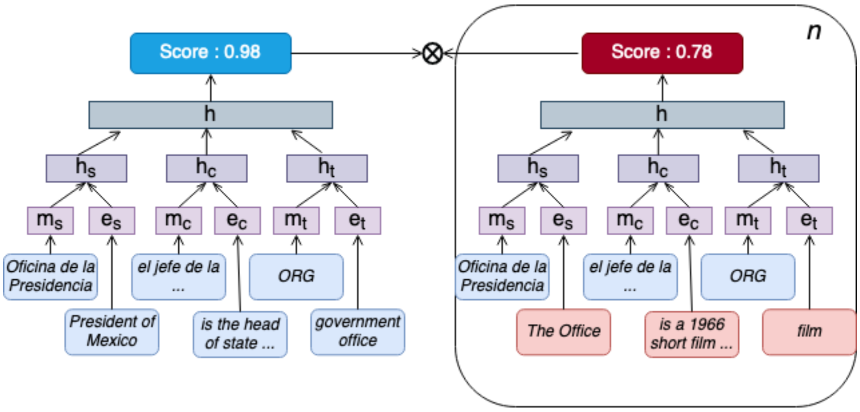

Figure 1: Example Spanish mention *Oficina de la Presidencia*, which is a link to entity *President of Mexico*, and the architecture for our neural ranker, using that example and a negatively-sampled entity *The Office*.

to align embeddings across languages, and a small dictionary to identify alignments. They pass these representations through a convolutional neural network to create a mention representation. They in turn use the other mention representations in the document to create a contextual representation, and also use a separate type vector. They train their network on hyperlinks from multiple languages in Wikipedia. Before the ranking step, they use a triage system similar to that of Tsai and Roth (2016). They evaluate on several entity linking datasets, including **TAC**. As their system only uses English Wikipedia as the KB, they set all mentions that link to a entity outside of Wikipedia to NIL; this results in a different evaluation setup than we need for our work. Their results show that training on all languages, instead of monolingual or bilingual training, generally performs best. For zero-shot entity linking, they train on English language Wikipedia. They find that their performance is heavily dependent on a prior probability derived from the triage system – otherwise, there is a large drop in performance.

Rijhwani et al. (2019) investigate zero-shot entity linking on low-resource languages. They propose a model consisting of a similarity model using encoders separately trained on high-resource language mentions, related to the low-resource language, and English entities. They then use the high-resource language as a pivot language for low resource language mentions, allowing them to score mentions in an unseen language. Raiman and Raiman (2018) consider multilingual entity linking, in which they use a KB in the same language as the mention, but exploit multilingual transfer for the model's type system. They formulate a type system as a mixed integer problem, which they use to learn a type system from knowledge graph relations.

3 Entity Linking Model

We propose a cross-language entity linker based on a pointwise neural ranker that scores a mention m and entity e pair, adapting from an architecture discussed in Dehghani et al. (2017). Unlike a classification architecture, a ranking architecture is able to score previously unseen entities. As is standard, we use a two stage system: triage followed by ranking; this reduces the number of entities that must be ranked, and results in better performance. Our system is shown in Figure 1. We select this architecture so as to focus on the ability of multilingual transformers to handle this task.

The ranker takes as input information about the mention and entity: 1) the mention string and entity name; 2) the context of the mention and entity description; and 3) the types of the mention and entity. We represent the mention string, entity name, mention context and entity description using a pre-trained multilingual deep transformer encoder (Devlin et al., 2019), while the mention and entity type are represented as one-hot embeddings. We describe the multilingual representation, model architecture and training procedure.

3.1 Multilingual Representations

We use multilingual BERT (mBERT) (Devlin et al., 2019)[2], which has been shown to create effective multilingual representations for downstream NLP tasks (Wu and Dredze, 2019). Consider the Spanish example in Figure 1. First, we create a represen-

---

[2] We found that XLM-R (Conneau et al., 2019) performed similarly and only report results on mBERT.

en	NN	0.195	0.463	0.550	0.502
	Mono	0.586	**0.703**	0.619	**0.658**
	MultiDS	0.509	0.873	0.478	0.618
	Multi	**0.602**	0.691	**0.626**	0.655
	MultiOr	*0.654*	*0.773*	*0.641*	*0.703*
	Tri	—	*0.736*	*0.738*	*0.737*
zh	NN	0.207	0.889	0.449	0.597
	Mono	0.709	**0.867**	0.728	0.791
	MultiDS	**0.733**	**0.867**	**0.746**	**0.801**
	Multi	0.730	0.862	0.735	0.793
	MultiOr	*0.828*	*0.950*	*0.812*	*0.876*
	Tri	—	*0.854*	*0.809*	*0.831*
es	NN	0.214	0.508	0.552	0.529
	Mono	0.595	**0.921**	0.587	0.714
	MultiDS	0.604	0.918	0.590	0.718
	Multi	**0.652**	0.918	**0.625**	**0.744**
	MultiOr	*0.691*	*0.936*	*0.655*	*0.770*
	Tri	—	*0.804*	*0.804*	*0.804*
	Model	**micro**	**prec.**	**recall**	**F$_1$**
ar	NN	0.171	0.414	0.602	0.491
	Mono	**0.660**	**0.683**	**0.816**	**0.743**
	Multi	0.637	0.661	0.778	0.715
fa	NN	0.330	0.694	0.734	0.714
	Mono	0.702	0.780	0.881	0.827
	Multi	**0.762**	**0.817**	**0.919**	**0.863**
ko	NN	0.269	0.816	0.597	0.690
	Mono	0.752	0.832	0.861	0.846
	Multi	**0.805**	**0.850**	**0.902**	**0.875**
ru	NN	0.358	0.841	0.529	0.649
	Mono	0.694	0.834	0.843	0.837
	Multi	**0.740**	**0.865**	**0.876**	**0.871**

Table 1: Micro-avg. precision, precision, recall, and F$_1$ for **TAC** and **Wiki** datasets. In a majority of languages, the **Multi** model outperforms the **Mono** model.

tation of the mention text m_s, *Oficina de la Presidencia*, by creating an mBERT representation of the entire sentence, selecting the lowest layer representations of each of the mention's sub-words,[3] and form a single representation using max pooling. We create a representation of the entity name e_s, *President of Mexico* in the same way, although there is no surrounding context as in a sentence.

For the mention context m_c we select the surrounding sentences up to BERT's 512 sub-word limit, positioning the mention in the middle, and pass the text to BERT, using the resulting top layer of the `[CLS]` token. We create a similar representation for the entity context e_c from the definition or other text in the KB, using the first 512 subword tokens from that description. For the mention type m_t and entity type e_t we create one-hot embeddings, omitting ones that do not occur more than 100 times in the training set.

3.2 Architecture

We feed the representations of the name (m_s and e_s), context (m_c, e_c) and type (m_t, e_t) into a neural ranker. Each of these three pairs is passed into distinct multilayer perceptrons (MLPs), which each produce an embedding that captures the similarity between each type of information. For example, we input m_s and e_s into a text-specific hidden layer, which produces a combined representation r_s. The same is done for the context and type representations, producing representations r_c and r_t, respectively. These three representations are then fed into a final MLP, which produces a final score ($[-1, 1]$.) The entire network is jointly trained with the ADAM optimizer and a ranking objective. We apply dropout at every layer, use ReLu as the intermediate activation function, and Tanh for the final layer. While additional features such as entity salience are likely useful for this task, we chose to restrict our model as much as possible to use only text features. This focuses on mBERT's multilingual ability, and allows for easier adaptation to new KBs than with KB-specific features.

3.3 Model Training

We learn the parameters θ of our scoring function S using a pairwise approach; this allows us to train our model without annotated scores. Our ranker scores a mention m and positive entity e_+ pair, and separately scores the same mention paired with n sampled negative entities e_-. We apply the hinge loss between our correct entity and the highest scoring negative entity,

$$L(\theta) = \mathbf{max}\{0, \epsilon - (S(\{m, e_+\}; \theta) - \\ \mathbf{max}\{S(\{m, e_{0-}\}; \theta) \ldots S(\{m, c_{n-}\}; \theta)\}\}$$

We jointly train all components of the network, including the positive and negative portions of the network. The major benefit of this pairwise approach is that it does not rely on annotated scores, but instead uses negative sampling to train the ranker. We

---

[3] We experimented with several BERT layers and found this to be the best performing on the **TAC** development set.

tested random combinations of hidden layer sizes and dropout rates to find the best configuration (see Appendix A for parameter selection details).

4 Datasets

We conduct our evaluation on two cross-language entity linking datasets. We predict NILs by applying a threshold; mentions where all entities are below a given threshold are marked as NIL. We evaluate all models using the evaluation script provided by Ji et al. (2015), which reports Precision, Recall, F_1, and Micro-averaged precision. For implementation details, please see the appendix.

TAC. The 2015 TAC KBP Entity Linking dataset (Ji et al., 2015) consists of newswire and discussion form posts in English, Spanish, and Mandarin Chinese linked to an English KB. We use their evaluation set, and provide a comparison to the numbers noted in Ji et al. (2015). The referenced systems had access to non-English language KB text which we exclude, and thus are a goal rather than a baseline. Later papers, such as Upadhyay et al. (2018), also use this dataset but only for evaluation, instead training on Wikipedia and treating mentions that are linked to TAC entities without Wikipedia links as NIL. Therefore, we cannot compare our evaluation to this work.

Wiki. We created a cross-language entity linking dataset from Wikipedia links (Pan et al., 2017) that includes Korean, Farsi, Arabic, and Russian. A preprocessed version of Wikipedia has links in non-English Wikipedia pages to other non-English pages annotated with that link and an English page link if a corresponding page was available. From these annotations we created a dataset consisting of non-English mentions linked to English-language entities (Wikipedia page) using English Wikipedia as the KB. We consider this to be silver-standard data because–unlike the **TAC** dataset–the annotations have not been reviewed by annotators. Since we do not have a separate development set for this dataset, we apply the hyperparameters selected on **TAC** development data to this dataset.

Triage. We assume gold-standard mention boundaries in our analysis. We use the triage system of Upadhyay et al. (2018), which is largely based on work in Tsai and Roth (2016). This allows us to score a smaller set of entities for each mention as opposed to the entire KB. For a give mention m, a triage system will provide a set of k candidate entities $e_1 \ldots e_k$. The system uses Wikipedia cross-links to generate a prior probability $\mathtt{P}_{\mathtt{prior}}(e_i|m)$ by estimating counts from those mentions. This prior is used to provide the top k English Wikipedia page titles for each mention ($k = 10$ for **TAC** and $k = 100$ for **Wiki**).

5 Model Evaluation

We consider several different training and evaluation settings to explore the multilingual ability of transformers on this task. Recent studies suggest that multilingual models can achieve similar or even better performance on cross-language entity linking (Upadhyay et al., 2018). Other work (Mueller et al., 2020) has shown that this is not always the case. Therefore, we begin by asking: does our linker do better when trained on all languages (multilingual cross-language) or trained separately on each individual language (monolingual cross-language)?

We train our model on each of the 7 individual languages in the two datasets (noted as **Mono**). Next, we train a single model for each dataset (3 languages in **TAC**, 4 in **Wiki**, each noted as **Multi**). **Mono** and **Multi** share the exact same architecture - there are no multilingual adjustments made, and the model contains no language-specific features. As **Multi** uses data available in all languages and thus has more training data than **Mono**, we include a model that is trained on a randomly-sampled subset of the multilingual training data that set to match the training size of **Mono** (**MultiDS**) . For **TAC Multi** models, we also report results using a candidate oracle instead of triage (**Multi+Or**), where the correct entity is always added to the candidate list. For all **Mono** and **Multi**-based models we report the average of three runs. The metric-specific standard deviations were all small, with all but one at or below 0.017. We note the best performing architecture from (Ji et al., 2015) as **Tri**, again noting that those systems have access to non-English text. We also evaluate a simple nearest neighbor model (noted as **NN**). This model scores each mention-entity pair using the cosine similarity between the mention name representation m_s and the entity representation e_s, and selects the highest-scoring pair.

Table 1 shows that for **TAC** there is a small difference between the **Mono** and **Multi** models. For **Wiki** the difference is often larger. **Multi** often does better than **Mono**, suggesting that additional training data is helpful specifically for languages (*e.g.,* Farsi) with smaller amounts of data. Over-

Training Setting	Evaluation Language: en	zh	es
Multi	0.66	0.79	0.74
en	.00	−.03	−.02
zh	−.05	.00	−.03
es	−.06	−.06	−.03

Training Setting	ar	fa	ko	ru
Multi	0.72	0.86	0.88	0.87
ar	+.03	−.08	−.08	−.05
fa	−.14	−.04	−.16	−.10
ko	−.20	−.13	−.03	−.09
ru	−.20	−.08	−.13	−.03

Table 2: ΔF_1 for each single-language trained model, compared to a multilingually-trained model, for each evaluation language. Each column is an evaluated language, and each row is a training setting. While models trained on the target language perform best, many monolingually-trained models perform well on unseen languages.

	en		zh		es	
	avg	F_1	avg	F_1	avg	F_1
name	0.59	0.70	0.45	0.71	0.42	0.73
+cont	+.12	+.05	+.22	+.05	+.14	+.05
+type	+.03	+.01	+.10	−.02	+.03	−.03
all	+.12	+.05	+.26	+.08	+.19	+.06

Table 3: English-only trained Δmicro-average and ΔF_1 when using a subset of linker features, compared to the name-only model for each language in the Development set. The name component of the model has the highest performance impact, but context also leads to better performance in almost all cases.

BERT	Lang	micro	prec.	recall	F_1
en	en	−.07	+.17	−.13	−.03
en	es	−.01	.00	−.02	−.01
ar	ar	−.08	−.08	−.03	−.06
ar	fa	−.09	−.05	−.08	−.06

Table 4: Change in performance for monolingually-trained models using monolingually-trained BERT models, compared to monolingually-trained models using mBERT.

all, these results are encouraging as they suggest that a single trained model for our system can be used for cross-language linking for multiple languages. This can reduce the complexity associated with developing, deploying and maintaining multiple models in a multilingual environment. For some models, the **Multi** improvement may be due to additional data available, as shown in the difference in performance between **Multi** and **MultiDS** (*e.g.,* Spanish F_1 **Multi** is +.026 over **MultiDS**). However, the small difference in performance shows that even by providing additional out-of-language training data, reasonable performance can be achieved even with reduced in-language training.

6 Zero-shot Language Transfer

Encouraged by the results on multilingual training, we explore performance in a zero-shot setting. How does a model trained on a single language perform when applied to an unseen language? We consider all pairs of languages, *i.e.,* train on each language and evaluate on all others in the same dataset[4].

Table 2 shows the change in F_1 for monolingually-trained models compared to multilingual models. While zero-shot performance does worse than a model with access to within-language training data, the degradation is surprisingly small: often less than 0.1 F_1. For example, a model trained on all 3 **TAC** languages achieves an F_1 of 0.79 on Chinese, but if only trained on English, achieves an F_1 of 0.76. This pattern is consistent across both models trained on related languages (Arabic → Farsi, loss of 0.08 F_1), and on unrelated languages (Russian → Korean, loss of 0.13 F_1).

Analysis. Why does zero-shot language transfer do so well for cross-language entity linking? What challenges remain to eliminate the degradation in performance from zero-shot transfer?

We answer these questions by exploring the importance of each component of our cross-language ranking system: mention string, context, and type. We conduct ablation experiments investigating the performance loss from removing these information sources. We then evaluate each model in an English-trained zero-shot setting. First, we train a zero shot model using only the mention text and entity name. We then compare the performance change that results from adding the context, the type, and both context and type (all features).

Table 3 shows that comparing the name and men-

---

[4] Work in Cross-language entity linking (Upadhyay et al., 2018; Tsai and Roth, 2016) has done similar evaluations, but focus on using external large data sources (Wikipedia) to train their models.

tion text alone accounts for most of the model's performance, a sensible result given that most of the task involves matching entity names. We find that context accounts for most of the remaining performance, with type information having a marginal effect. This highlights the importance of the multilingual encoder, since both name and context rely on effective multilingual representations.

Separately, how does using a multilingual transformer model, such as mBERT, affect the performance of our ranker? First, it is possible that using a monolingual linker with a BERT model trained only on the target language would improve performance, since such a model does not need to represent several languages as the same time. As shown in Table 4, model performance for these settings is largely worse for English-only and Arabic-only (Safaya et al., 2020) models when compared to using mBERT, with the exception that precision increases significantly for English. Second, perhaps a monolingual linker with a BERT model trained only on a related language – *e.g.,* English BERT for Spanish, Arabic BERT for Farsi – would produce acceptable results. Again, as shown in Table 4, the performance is most often worse, illustrating that mBERT is an important aspect of the linker's performance.

7 Improving Zero-shot Transfer

7.1 Name Matching Objective

Given the importance of matching the mention string with the entity name, will improving this component enhance zero-shot transfer? While obtaining within-language entity linking data isn't possible in a zero-shot setting, we can use pairs of translated names, which are often more easily available (Irvine et al., 2010; Peng et al., 2015). Since Chinese performance suffers the most zero-shot performance reduction when compared to the multilingual setting, we use Chinese English name pair data (Huang, 2005) to support an auxiliary training objective. An example name pair: "巴尔的摩－俄亥俄铁路公司" and *Baltimore & Ohio Railroad.*

We augment model training as follows. For each update in a mini-batch, we first calculate the loss of the subset of the model that scores the mention string and entity name on a randomly selected pair $k = 25,000$ of the Chinese/English name pair corpus. We score the Chinese name z and the correctly matched English name e_+ pair, and separately score the same Chinese name paired with n negatively sampled English names e_-. We create representations for both z and e using the method described for names in §3.1 which are passed to the name-only hidden layer. We add a matching-specific hidden layer, which produces a score. We apply the hinge loss between positive and negative examples,

$$N(\theta) = \mathbf{max}\{0, \epsilon - (S(\{z, e_+\}; \theta) - \mathbf{max}\{S(\{z, e_{0-}\}; \theta) \ldots S(\{z, e_{n-}\}; \theta)\}\}$$

The name pair loss is then multiplied by a scalar $\lambda = 0.5$ and added to the loss described in §3.3. The resulting loss $L_{joint}(\theta) = (\lambda * N(\theta)) + L(\theta)$ is jointly minimized. After training, we discard the layer used to produce a score for name matches. This procedure still only uses source language entity linking training data, but makes use of auxiliary resources to improve the name matching component, the most important aspect of the model.

We analyze the resulting performance by considering modifications to our English-only training setting, which are designed to replicate scenarios where there is little training data available. To show the effect of a smaller training corpus, we select a random 50% of mentions, partitioned by document (**Rand**). To show the importance of training on frequently occurring entities, we select 50% of mentions that are linked to the least frequent entities in the English dataset (**Tail**).

Table 5 shows the results on each of the three development **TAC** languages compared to the **Multi** model. For the **Rand** training set, we see a large improvement in Chinese micro-average and a small one in F_1, but otherwise see small reductions in performance. In the **Tail** training setting, a similar pattern occurs, with the exception that Chinese is less improved than in **Rand**. Overall, performance loss remains from zero-shot transfer which suggests that improvements need to be explored beyond just name matching.

7.2 Entities

Another possible source of zero-shot degradation is the lack of information on specific entities mentioned in the target language. For entity linking, knowledge of the distribution over the ontology can be very helpful in making linking decisions. While zero-shot models have access to general domain text, *i.e.,* news, they often lack text discussing the same entities. For example, some entities that only occur in Chinese (231 unique entities in **Dev**),

en		zh		es				en		zh		es	
avg	F_1	avg	F_1	avg	F_1			avg	F_1	avg	F_1	avg	F_1
0.64	0.75	0.51	0.69	0.53	0.75	Rand	Tail	0.53	0.66	0.45	0.66	0.42	0.70
.00	−.01	+.07	+.02	−.02	−.02	w/ Name		−.02	−.02	+.02	+.01	−.01	−.01
.00	+.01	+.06	+.04	+.01	+.02	w/ Pop-Train		−.02	+.04	.00	+.07	−.01	+.06
+.04	+.03	+.12	+.06	+.10	+.06	w/ Pop-All		+.13	+.10	+.20	+.11	+.22	+.10

Table 5: For each proposed Name matching or popularity re-ranking model, the change in performance (ΔF_1 and Δmicro-average) compared to the original **Rand** (left) and **Tail** (right) models. While the name matching increased performance somewhat, the additional of popularity was more impactful.

	en		zh		es	
	avg	F_1	avg	F_1	avg	F_1
Multi	0.70	0.73	0.77	0.81	0.68	0.82
Rand	−.04	-.02	−.26	−.12	−.15	−.07
N-1	+.01	+.02	−.04	−.02	−.08	−.03
N-1U	−.24	-.14	−.49	−.22	−.38	−.19
Tail	−.16	-.08	−.31	−.15	−.26	−.12

Table 6: For each of the English-only training data subsets described in §7.2, ΔMicro-average and ΔF_1 compared to the full **Multi** model. Models that see even a single example of an entity (*e.g.,* **N-1**) outperform models that see a portion (*e.g.,* **Tail**) or none (*e.g.,* **N-1U**).

such as the frequently occurring entity *Hong Kong*, have a number of similar entities and thus are more challenging to disambiguate.

We measure this effect through several diagnostic experiments where we evaluate on the development set for all languages, but train on a reduced amount of English training data in the following ways: In addition to the **Rand** and **Tail** settings, we sample a single example mention for each entity (**N-1**), resulting in a much smaller training as compared to those datasets. We also take **N-1** and remove all evaluation set entities (**N-1U**), leaving all evaluation entities unseen at train time.

Table 6 reports results on these reduced training sets. All languages use a -1 NIL threshold. Compared to the multilingual baseline (**Multi**) trained on all languages, there is a decrease in performance in all settings. Several patterns emerge. First, the models trained on a subset of the English training data containing more example entities - *e.g.,* **N-1** - have much higher performance than the models that do not. This is true even in non-English languages. Unobserved entities do poorly at test time, suggesting that observing entities in the training data is important.

However, a mention training example can improve the performance of a mention in another language if linked to the same entity, which suggests that this provides the model with data-specific entity information. Therefore, the remaining zero-shot performance degradation can be largely attributed not to a change in language, but to a change in topic, *i.e.,* what entities are commonly linked to in the data. This may also explain why although the name matching component is so important in zero-shot transfer, our auxiliary training objective was unable to fully mitigate the problem. The model may be overfitting to observed entities, forcing the name component to memorize specific names of popular entities seen in the training data. This means we are faced with a topic adaptation rather than a language adaptation problem.

We validate this hypothesis by experimenting with information about entity popularity. Will including information about which entities are popular improve zero-shot transfer? We answer this question by re-ranking the entity linker's top ten predicted entities using popularity information, selecting the most most popular entity from the list. Adding this feature into the model and re-training did not lead to a significant performance gain. We define the popularity of an entity to be the number of times it occurred in the training data. We report results for two popularity measures–one using the popularity of the English subset of the data used for training, and one using all of the training data (including for Spanish and Chinese).

Table 5 shows that both strategies improve F_1, meaning that a missing component of zero-shot transfer is information about which entities are favored in a specific dataset. The gain from using popularity estimated from the training data only is smaller than using the popularity data drawn from all of **TAC**. With more accurate popularity information, we can better mitigate loss.

Several patterns emerge from most common corrections made with the Population reranking for **Tail**, included in Table 8. Many errors arise from selecting related entities that are closely related to the correct entity – for example, *United States Congress* instead of the *United States of America*. Additionally, people with similar names are often confused (e.g. *Edmund Hillary* instead of *Hillary Clinton*). Finally, many appear to be annotation decisions – often both the original prediction (e.g. *Islamic State*) and the corrected popular prediction (e.g. *Islamic State of Iraq and Syria*) appear reasonable choices. While most corrections were in Chinese (632), some occurred in both English (419) and Spanish (187). These errors – especially those in English – illustrate that much of the remaining error is in failing to adapt to unseen entities.

8 Conclusion

We demonstrate that a basic neural ranking architecture for cross-language entity linking can leverage the power of multilingual transformer representations to perform well on cross-lingual entity linking. Further, this enables a multilingual entity linker to achieve good performance, eliminating the need for language-specific models. Additionally, we find that this model does surprisingly well at zero-shot language transfer. We find that the zero-shot transfer loss can be partly mitigated by an auxiliary training objective to improve the name matching components. However, we find that the remaining error is *not* due to language transfer, but to topic transfer. Future work that improves zero-shot transfer should focus on better ways to adapt to entity popularity in target datasets, instead of relying on further improvements in multilingual representations. Focusing on adapting to the topic and entities present in a given document is critical. This could be accomplished by adding a document-level representation or by leveraging other mentions in the document. English-focused work on rare entity performance (Orr et al., 2020; Jin et al., 2014) may provide additional direction.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Conference on Computational Linguistics (COLING)*, pages 277–285. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, Copenhagen, Denmark. Association for Computational Linguistics.

Shudong Huang. 2005. *Chinese - English Name Entity Lists v 1.0 LDC2005T34*. Linguistic Data Consortium.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from all languages. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking. *TAC*.

Yuzhe Jin, Emre Kıcıman, Kuansan Wang, and Ricky Loynd. 2014. Entity linking at the tail: Sparse signals, unknown entities, and phrase models. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, page 453–462, New York, NY, USA. Association for Computing Machinery.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

Processing and the 9th International Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Chi-kiu Lo and Michel Simard. 2019. Fully unsupervised crosslingual semantic textual similarity metric based on BERT for identifying parallel data. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 206–215, Hong Kong, China. Association for Computational Linguistics.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas Oard, and David Doermann. 2011. Cross-language entity linking. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 255–263, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

David Mueller, Nicholas Andrews, and Mark Dredze. 2020. Sources of transfer in multilingual named entity recognition.

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Re. 2020. Bootleg: Chasing the tail with self-supervised named entity disambiguation.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *North American Chapter of the Association for Computational Linguistics*, pages 1130–1139.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1946–1958.

Nanyun Peng, Mo Yu, and Mark Dredze. 2015. An empirical study of Chinese name matching and applications. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 377–383, Beijing, China. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Jonathan Raphael Raiman and Olivier Michel Raiman. 2018. Deeptype: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Shruti Rijhwani, Jiateng Xie, Graham Neubig, and Jaime Carbonell. 2019. Zero-shot neural transfer for cross-lingual entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6924–6931.

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online). International Committee for Computational Linguistics.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual Wikification Using Multilingual Embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 589–598, Stroudsburg, PA, USA. Association for Computational Linguistics.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical BERT models for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3632–3636, Hong Kong, China. Association for Computational Linguistics.

Shyam Upadhyay, Nitish Gupta, and Dan Roth. 2018. Joint multilingual supervision for cross-lingual entity linking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2495, Brussels, Belgium. Association for Computational Linguistics.

Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu, and Ting Liu. 2019. Cross-lingual BERT transformation for zero-shot dependency parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5721–5727, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages

833–844, Hong Kong, China. Association for Computational Linguistics.

A Architecture information

Parameter	Values
Context Layer(s)	[768], **[512]**, [256], [512,256]
Mention Layer(s)	[768], **[512]**, [256], [512,256]
Type Layer	[128], **[64]**, [32], [16]
Final Layer(s)	**[512,256]**, [256,128], [128,64], [1024,512], [512], [256]
Dropout probability	0.1, **0.2**, 0.5
Learning rate	1e-5, 5e-4, **1e-4**, 5e-3, 1e-3

Table 6: To select parameters for the ranker, we tried 10 random combinations of the above parameters, and selected the configuration that performed best on the TAC development set. The selected parameter is in bold. We report results after training for 500 epochs for TAC and 800 for Wiki. The full TAC multilingual model takes approximately 1 day to train on a single NVIDIA GeForce Titan RTX GPU, including candidate generation, representation caching, and prediction on the full evaluation dataset.

B Dataset Details

The NIL threshold is selected based on the development**TAC** dataset. Unless noted, we use -0.8 for English and -1 otherwise.

TAC: The training set consists of 30,834 mentions (6,857 NIL) across 447 documents. We reserved a randomly selected 20% of these documents as our development set, and will release development splits. The evaluation set consists of 32,459 mentions (8,756 NIL) across 502 documents. A mention is linked to NIL if there is no relevant entity in the KB, and the KB is derived from a version of BaseKB.

TAC Triage: We use the system discussed in for both the **TAC** and **Wiki** datasets. However, while the triage system provides candidates in the same KB as the **Wiki** data, not all entities in the **TAC** KB have Wikipedia page titles. Therefore, the **TAC** triage step requires an intermediate step - using the Wikipedia titles generated by triage ($k = 10$), we query a Lucene database of BaseKB for relevant entities. For each title, we query BaseKB proportional to the prior provided by the triage system, meaning that we retrieve more BaseKB entities for titles that have a higher triage score, resulting in $l = 200$ entities. First, entities with Wikipedia titles are queried, followed by the entity name itself. If none are found, we query the mention string - this provides a small increase in triage recall. This necessary intermediate step results in a lower recall rate for the **TAC** dataset (85.1% for the evaluation set) than the **Wiki** dataset, which was 96.3% for the evaluation set .

Wiki: Some BaseKB entities used in the **TAC** dataset have Wikipedia links provided; we used those links as seed entities for retrieving mentions, retrieving mentions in proportion to their presence in the **TAC** dataset, and to sample a roughly equivalent number of non-TAC entities. We mark 20% of the remaining mentions as NIL. In total, we train and evaluate on 5,923 and 1,859 Arabic, 3,927 and 1,033 Farsi, 5,978 and 1,694 Korean, and 5,337 and 1,337 Russian mentions, respectively.

Original Prediction	Popular Correction	Count
United States Department of State	United States of America	146
united_states_congress	United States of America	121
Soviet Union	Russian	57
Central Intelligence Agency	United States of America	41
healthcare_of_cuba	Cuba	36
islamic_state	Islamic State of Iraq and Syria	33
edmund_hillary	First lady Hillary Rodham Clinton	32
United States Department of Defense	United States of America	32
Tamerlan Tsarnaev	Dzhokhar A. Tsarnaev	27
Carl Pistorius	Oscar Leonard Carl Pistorius	23
CUBA_Defending_Socialism_ ... documentary	Cuba	22
Barack Obama Sr.	Barack Hussein Obama II	18
Iraq War	Iraq	14
Dzhokhar Dudayev	Dzhokhar A. Tsarnaev	13
Sumter County / Cuba town	Cuba	13
United States Army	United States of America	13
military_of_the_united_states	United States of America	13
Republic of Somaliland	Somalian	13
ISIS	Islamic State of Iraq and Syria	13
Islamic_State_of_Iraq_and_Syria	Islamic State of Iraq and Syria	12
National Assembly of People's Power	Cuba	11
Sara Netanyahu	Benjamin Netanyahu	10

Table 8: All pairs of original prediction and popular prediction altered by the reranking procedure described in Section 7.2, for the **Tail** model

TellMeWhy: A Dataset for Answering Why-Questions in Narratives

Yash Kumar Lal
Stony Brook University
`ylal@cs.stonybrook.edu`

Nathanael Chambers
US Naval Academy
`nchamber@usna.edu`

Raymond Mooney
University of Texas, Austin
`mooney@cs.utexas.edu`

Niranjan Balasubramanian
Stony Brook University
`niranjan@cs.stonybrook.edu`

Abstract

Answering questions about why characters perform certain actions is central to understanding and reasoning about narratives. Despite recent progress in QA, it is not clear if existing models have the ability to answer "why" questions that may require commonsense knowledge external to the input narrative. In this work, we introduce **TellMeWhy**, a new crowd-sourced dataset that consists of more than 30k questions and free-form answers concerning why characters in short narratives perform the actions described. For a third of this dataset, the answers are not present within the narrative. Given the limitations of automated evaluation for this task, we also present a systematized human evaluation interface for this dataset. Our evaluation of state-of-the-art models show that they are far below human performance on answering such questions. They are especially worse on questions whose answers are external to the narrative, thus providing a challenge for future QA and narrative understanding research.

1 Introduction

The actions people perform are steps of plans to achieve their desired goals. When interpreting language, humans naturally understand the reasons behind described actions, even when the reasons are left unstated (Schank and Abelson, 1975). For NLP systems, answering questions about *why* people perform actions in a narrative can test this ability. Answering such questions often requires filling the implicit gaps in the story itself.

Consider this narrative from ROCStories (Mostafazadeh et al., 2016b):

> Rudy was convinced that bottled waters all tasted the same. He went to the store and bought several popular brands. He went back home and set them all on a table. He spent several hours tasting them one by one. He came to the conclusion that they actually did taste different.

Now try to answer the question, *"Why did he go to the store and buy several popular brands?"* The answer *"he wanted to taste test"* is not explicit in the narrative and requires us to read between the lines to fill in the gaps (Norvig, 1987). While humans can visualise and process the events in a story to hypothesize why they might have occurred (Kintsch and Dijk, 1978), current NLP systems fall well short of exhibiting similar capabilities. They are unable to adequately formulate the reasons behind actions in specific contexts.

How can we get NLP models to reason about why actions are performed? One way is to consider theories like script learning (Schank and Abelson, 1975; Pichotta and Mooney, 2014) or learning from co-occurrence (Chambers and Jurafsky, 2009). But they only partially capture this type of knowledge – much like other forms of commonsense knowledge, the reasons for why actions are performed are often left implicit in text. Even though there are many large scale QA datasets, they rarely contain questions about *why* people perform actions.

Therefore, we introduce the TellMeWhy dataset, a collection of 30,519 such why-questions, each with 3 "gold standard" human answers. Each record in TellMeWhy contains a short story, an associated question, and its 3 possible answers.

Further, we focus on enabling *human* evaluation of this dataset; human evaluation is more reliable than automatic metrics to evaluate such systems (Celikyilmaz et al., 2020; Gatt and Krahmer, 2018). However, reliability of human judgment is substantially impacted by experimental setup (Novikova et al., 2018; Santhanam and Shaikh, 2019). There is little consensus on how human evaluations should be conducted, so results are often incomparable across evaluations.

To this end, we present a systematized evaluation framework on MTurk for the TellMeWhy text generation task – and release the framework for future

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 596–610
August 1–6, 2021. ©2021 Association for Computational Linguistics

researchers. The MTurk interface asks annotators to rate generated answers on their grammaticality and validity. We show that with our interface human answers are judged to be of high quality (99% grammatical, 96% valid) with strong inter-annotator agreement at 0.88 Fleiss Kappa. This indicates high agreement and also confirms the design of our interface.

Finally, we present baseline results for TellMeWhy and compare against our human ceiling. We finetune two large language models that have proven to be effective for a variety of tasks, GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020), and a dedicated question answering model, UnifiedQA (Khashabi et al., 2020), to perform this task. Human evaluation is performed on their outputs from independent test data. All models significantly under-perform the human benchmark and are especially worse on questions where the answer cannot be simply copied over from text in the narrative. The results clearly demonstrate the difficulty for current models to convincingly answer such why-questions.

This paper's contributions are as follows: (1) we introduce TellMeWhy, a large dataset of English why-questions for narratives derived from ROCStories (Mostafazadeh et al., 2016a) and CATERS (Mostafazadeh et al., 2016b) along with answers from 3 distinct humans, (2) a systematized human evaluation interface to calibrate model outputs consistently, and (3) show that current models are ill-equipped to perform this task. We release the dataset and human evaluation suite at `http://lunr.cs.stonybrook.edu/tellmewhy`.

2 Related Work

2.1 Datasets containing why-questions

Most of the datasets related to why-questions fall into one or more of the following categories: (1) very small size, (2) not focused on stories, or (3) focused on connecting known events instead of answering reasoning questions.

Some corpora of why-questions have been collected manually: corpora described in Verberne et al. (2006) and Verberne et al. (2007) both comprise fewer than 400 questions and corresponding answers (one or two per question) formulated by native speakers. Dunietz et al. (2020) demonstrate that it is important to define what we want models to comprehend when building datasets for machine reading comprehension (MRC) tasks. They design templates of understanding corresponding to the four elements identified by Zwaan et al. (1995). For 201 questions, they design multiple-choice questions derived from (Lai et al., 2017) to test understanding of different categories of events. All of these are very small corpora that cannot be viably used to further a model's understanding of why-questions in stories.

Higashinaka and Isozaki (2008) extend an existing factoid QA system to answer why-questions by integrating corpus based features, calling it NAZEQA. Oh et al. (2012) extract a set of answer candidates from a web corpus, and perform re-ranking using SVMs to predict the right answer. Oh et al. (2019) use an adversarial learning framework to generate a vector representation from the passage to judge whether the passage actually answers the why-question. These papers focus on Japanese news (Fukumoto et al., 2007; Oh et al., 2012), including NTCIR-6, and most critically, all these datasets are very small.

Some prior work focuses on knowledge extraction, not the *reasons* behind the actions. Mrozinski et al. (2008) built a corpus of why-questions related to Wikipedia articles. These were general knowledge questions with solicited answers from paid workers. Dependency parsing can be used to rephrase why-questions into statements with a 'because' prompt to elicit explanations from models (Nie et al., 2019). PhotoshopQuiA (Dulceanu et al., 2018) contains questions and answers specifically about Photoshop.

NarrativeQA (Kočiský et al., 2018) provides a dataset of 1,567 stories (books and movie scripts) containing 46,765 wh-questions written and answered by human annotators. Unfortunately, only 9.78% are why-questions, which makes for a small collection. QuAIL (Rogers et al., 2020) has a small subset of multiple choice questions pertaining to causality in user stories. These datasets are targeted at broad abilities of reading comprehension, not specifically about explaining actions in stories.

Some recent datasets causally connect events in text, but they do not target answering why-questions. ATOMIC (Sap et al., 2019) consists of entries that describe a likely cause/effect of events. Most notably, ATOMIC is *non-contextual* so it is more about general knowledge, not interpreting a specific story/context. Perhaps most relevant is GLUCOSE (Mostafazadeh et al., 2020), a crowdsourced dataset of implicit commonsense

Dataset	Size	Domain
NTCIR-6	200	Japanese news
Mrozinski et al. (2008)	695	Wikipedia
PhotoshopQuiA	2,854	Product focused
NarrativeQA	4,573	Books+Movie scripts
Dunietz et al. (2020)	201	Exam questions

Table 1: Previous why-question corpora. NarrativeQA has 46,765 questions of which 4,573 are why-questions.

knowledge in the form of causal mini-theories grounded in narrative context. These theories are semi-structured inference rules. This dataset is not aimed at answering why-questions, but at creating direct relationships between events already mentioned in the story. They focus on capturing specific cause-enable type relations. Annotators were given a very constrained task – they had to select options from a drop down menu describing inference rules.

Abductive commonsense reasoning tests whether models can come up with a plausible explanation to connect a set of events. Bhagavatula et al. (2020) present ART with two abductive tasks: 1) given two observations, select one out of two plausible hypotheses, 2) and generate text connecting two events. This line of work focuses on connecting the dots between two events and does not address explaining *why* an action was performed. Our work crucially differs from these because the answer is often not in the story at all. StrategyQA (Geva et al., 2021) is a new dataset focusing on performing better implicit reasoning for multi-hop question answering tasks.

We summarize the different why-questions corpora in Table 1. None of them represent a large dataset focused on answering why-questions about actions in a narrative.

2.2 Human evaluation for NLG tasks

Among language generation tasks, machine translation has received the most attention in terms of human evaluation. Qualified crowd workers score output translations given the source or reference text to calibrate MT systems (Sakaguchi and Van Durme, 2018; Graham et al., 2013, 2014). WMT conducts annual evaluation of outputs of systems submitted to the shared task and uses it as one of the primary metrics (along with BLEU) to rank systems (Bojar et al., 2016, 2017, 2018; Barrault et al., 2019, 2020).

ChatEval (Sedoc et al., 2019) is an evaluation platform for chatbots. Zellers et al. (2020) present a leaderboard for their advice generation task. These platforms incorporate some manual analysis, but focus on very different tasks. None of their Mechanical Turk interfaces can be used for our task. We were unable to find a consistent interface for human evaluation of an open-ended question answering task. To address this flaw, we propose a standard human intelligence task (HIT) evaluation scheme for our dataset.

3 Dataset Creation

We want to test the abilities of models to understand the reasoning behind actions in a story. Therefore, we create a dataset of *why* questions that ask for explanations for actions performed in a story. Answering these questions requires an understanding of the events that are explicit in the story as well as access to implicit common-sense knowledge on how people use actions as parts of plans to achieve goals. To cover a wide-range of common situations, we utilize ROCStories (Mostafazadeh et al., 2016a), a collection of 45,496 five-sentence commonsense stories. We also develop a small "hidden" test set that was only used for the final evaluation using the CATERS (Mostafazadeh et al., 2016b) subset of ROCStories.

3.1 Why-Question Generation

Our strategy for creating *why* questions is simple. For each action in the narrative, we formulate a why question by applying simple template-based transformations. We dependency parse each sentence using SpaCy's en_core_web_sm model (Honnibal et al., 2020). We use the generated parse tree to rephrase the sentence into a question about the action described. The generated parse tree is used to extract the subject, object, and verb. We consider 3 types of sentences and design question templates accordingly: (1) sentences that have a primary and auxiliary verb, (2) sentences that only have a primary verb, and (3) sentences that only contain an auxiliary verb. For the first, the question template is: "Why {aux_verb} {subject} {verb} {object}?", for the second: "Why did {subject} {verb_lemma} {obj}?", and for the third; "Why {aux_verb} {subject} {obj}?".

This procedure yielded a little over 113k questions from ROCStories, and 489 questions from the CATERS portion. We selected at random 32,165 questions from stories that had at least three ques-

Split	# stories	# questions
Train	7558	23964
Dev	944	2992
Test	944	3099
Hidden Test	190	464
Total	9,636	30,519

Table 2: Dataset Statistics

tions[1]. We ensure that there is no overlap between the two subsets.

3.2 Collecting Answers

We crowd-sourced answers to these questions using Amazon Mechanical Turk. Figure 1a shows the interface used to collect these answers. Annotators were presented a narrative and asked to answer three *why* questions in free-form. For each question, they were also asked to provide judgments about the comprehensibility of the question, and whether the narrative explicitly contained the answer. They also selected the sentences from the narrative which influenced their answer (if any). To avoid variability in answer prefixes, we provide a prompt to start answering the question. We rephrase the sentence from which the question was generated to create these prompts. We consider the same categorisation of sentences described in subsection 3.1. For sentences that have both primary and auxiliary verbs, the answer prompt is of the form: “{subject} {aux_verb} {verb} {object} because...". When it only contains a primary verb, it is of the form: “{subject} {verb} {object} because...". If it only contains an auxiliary verb, it is of the form: “{subject} {aux_verb} {object} because...". We found, over several iterations of this HIT, that providing a prompt gave workers an initial direction and improved the quality of answers collected.

We ask three distinct annotators (three-way redundant task) to answer each of these questions. Annotators are not allowed to copy pieces of text to make up an answer. We discard questions that were deemed incomprehensible by any annotator.[2] With this process, we obtained 3 answers each from 30,055 questions from 9,636 stories (see Appendix B for more details). Table 2 shows basic statistics of the dataset. We refer to annotations from the CATERS data as the hidden test set. Examples of records in the dataset are presented in Table 3. The narrative does not explicitly contain an answer for the second question. We call these types the implicit-answer questions; they require extra common-sense inference to produce a plausible answer. Questions are categorised as implicit-answer if at least 2 out of 3 human annotators indicate that the answer cannot be explicitly found in the narrative. The annotators indicated as much and, based on their commonsense knowledge, provided plausible answers.

Story: Sandra got a job at the zoo. She loved coming to work and seeing all of the animals. Sandra went to look at the polar bears during her lunch break. She watched them eat fish and jump in and out of the water. She took pictures and shared them with her friends.
Question: Why did Sandra go to look at the polar bears during her lunch break?
Ans: she wanted to take some pictures of them.

Story: Cam ordered a pizza and took it home. He opened the box to take out a slice. Cam discovered that the store did not cut the pizza for him. He looked for his pizza cutter but did not find it. He had to use his chef knife to cut a slice.
Question: Why did Cam order a pizza?
Ans: Cam was hungry.

Table 3: TellMeWhy examples. The first is answerable directly from text in the story, but the second requires external knowledge. We only show one out of three available answers here.

3.3 Validating Answers

To ensure an even higher-quality test set, we conducted another round of crowdsourcing to validate the answers by the first set of crowd-workers on the CATERS portion (464 questions). This validation interface is show in Figure 1b. It also serves as the base design for our systematized human evaluation. Annotators are presented a story, a related question, and the three answers that were collected as described in Section 3.2.

Three new annotators then rated two aspects of each answer:

(1) **Grammaticality** – Workers are asked to rate

---

[1] Since we ask annotators to read an entire story to answer these questions, avoiding stories with fewer questions optimizes reading time.

[2] On ROCStories we discarded 1,546 questions and on CATERS we discarded 25 questions

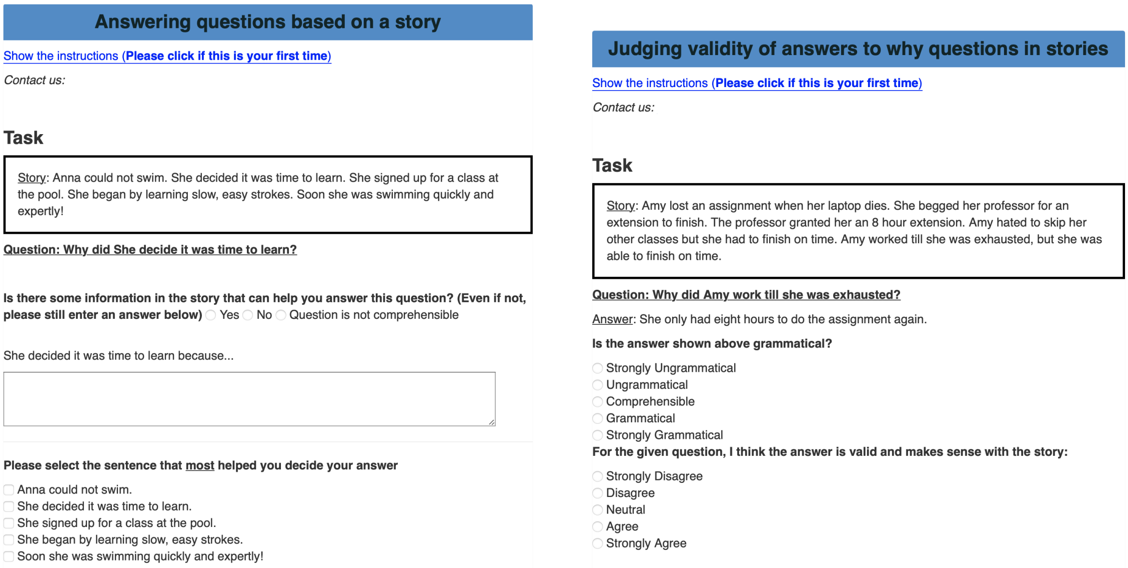

Answering questions based on a story

Show the instructions (**Please click if this is your first time**)

Contact us:

Task

Story: Anna could not swim. She decided it was time to learn. She signed up for a class at the pool. She began by learning slow, easy strokes. Soon she was swimming quickly and expertly!

Question: Why did She decide it was time to learn?

Is there some information in the story that can help you answer this question? (Even if not, please still enter an answer below) ○ Yes ○ No ○ Question is not comprehensible

She decided it was time to learn because...

Please select the sentence that most helped you decide your answer

- ☐ Anna could not swim.
- ☐ She decided it was time to learn.
- ☐ She signed up for a class at the pool.
- ☐ She began by learning slow, easy strokes.
- ☐ Soon she was swimming quickly and expertly!

(a) Task 1: Answer collection

Judging validity of answers to why questions in stories

Show the instructions (**Please click if this is your first time**)

Contact us:

Task

Story: Amy lost an assignment when her laptop dies. She begged her professor for an extension to finish. The professor granted her an 8 hour extension. Amy hated to skip her other classes but she had to finish on time. Amy worked till she was exhausted, but she was able to finish on time.

Question: Why did Amy work till she was exhausted?

Answer: She only had eight hours to do the assignment again.

Is the answer shown above grammatical?

- ○ Strongly Ungrammatical
- ○ Ungrammatical
- ○ Comprehensible
- ○ Grammatical
- ○ Strongly Grammatical

For the given question, I think the answer is valid and makes sense with the story:

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

(b) Task 2: Answer validation

Figure 1: MTurk interfaces used to curate data from crowd-source workers

the grammaticality of each answer on 5-point Likert scales, ranging from 'Strongly Ungrammatical' to 'Strongly Grammatical'. An answer is strongly grammatical if it follows all the rules of English grammar. It is grammatical if there is a mistake in tense, number, punctuation or something minor. It is comprehensible if there are clear grammatical mistakes but its meaning can be inferred, and it is then considered to be neutral on the Likert scale.

(2) **Validity** – Workers are asked to rate the validity of each answer on a 5-point Likert scale. Given the story and question, the annotators check if the given answer 'is valid and makes sense with the story'. An answer is considered invalid if it does not give a plausible reason relevant to the question asked and instead states irrelevant information.

Annotators agreed (by majority) that 99.07% of answers are grammatical and 95.47% of answers are valid. On grammaticality, there is some disagreement in judgment 0.7% of the time, while there is some disagreement in judgment 1% of the time for answer validity. We measured the inter-rater reliability of annotators' judgments using weighted Fleiss's Kappa (Marasini et al., 2016) and follow the weighting scheme used by Bastan et al. (2020). This measure has a penalty for each dissimilar classification based on the distance between two classes. For instance, if two annotators classify a document as a positive, the agreement weight is 1, but if one classifies as a positive, and the other classifies as slightly positive the agreement weight is less. The weighted agreement score for this subset is 0.88 for grammaticality annotations and 0.81 for validity annotations, indicating that the annotations are highly reliable. More details can be found in Appendix C.2.

4 Dataset Analysis

One of the key distinguishing aspects of answering *why* questions is that, in addition to understanding explicitly stated events, they also require access to commonsense explanations that may be external to the narrative. We conduct some analyses to investigate the prevalence of this phenomenon: (i) We asked annotators to judge whether the answer to a question could be found stated explicitly or only implicitly in the narrative and find that at least two out of three annotators could not find explicit answers in the story 28.82% of the time. (ii) We also asked crowd-workers to indicate which sentences helped them answer the question. Out of 91,557 collected answers, we find that 39,661 answers were provided without an influential sentence from the story. (iii) Last, we observe that there is only a 57.04% lexical overlap between the words used in answers and the original narrative. This suggests that annotators included new inferred information in their answers, instead of just copying something from the story. We calculate lexical overlap as the number of common tokens in the narrative and the answer divided by the length of the answer.

We hypothesize that questions about the first action in a story are more difficult to answer since there is no prior information to provide an explicit answer. We find that 55.03% of such questions were judged to be implicit-answer questions by a majority of the assigned annotators. Such questions help test systems' ability to infer plausible answers rather than just copy answers from the text.

We also evaluated the diversity of the answers for each question using simple lexical overlap. Of the 30k questions, only 150 questions had over 90% overlap in all 3 answers, i.e., essentially, the 3 distinct annotators wrote the same answer. For 4,243 other questions, two out of three answers had over 90% overlap. But for the vast majority of 26,068 questions, we obtained 3 fairly diverse answers. The average overlap between them is 26.12%. On average, the answers were 7.59 words long.

Overall, this analysis indicates how TellMeWhy differs from prior datasets. The answers cannot always be retrieved or connected to other events in the given text.

5 Benchmarking

How well do large language models answer *why* questions on narratives and what are their failure modes? To answer these, we use TellMeWhy to benchmark the performance of multiple state-of-the-art models and provide an analysis of their performance.

Formally, given a story S as context and a related why-question Q, models are required to generate a plausible answer A for the question. Since the answers are open-ended texts we compare them on standard automatic evaluation metrics for generation but also conduct a human evaluation.

5.1 QA Models

GPT-2 (Radford et al., 2019) is a large transformer-based language model trained on an enormous web corpus, which has been shown to be effective on a wide-range of language related tasks including question answering. It was one of the first models trained on diverse data to outperform domain-specific language models.

We used Huggingface (Wolf et al., 2020) to finetune a pretrained GPT-2 model on TellMeWhy. As input, the model receives a concatenation of the narrative and the related question (in that order), and the target is the answer. The input and target are separated using the '[SEP]' token. We finetune the model with batch size 16, learning rate 1e-5 and maximum output length 25. The model is trained until the dev loss fails to improve for 3 iterations.

T5 (Raffel et al., 2020) is an encoder-decoder model pre-trained on a mixture of unsupervised and supervised tasks in a multi-task setting, where each task is converted into a text-to-text format. It is a text-to-text model, which means it can be trained on arbitrary tasks involving textual input and output. T5 has achieved the state of the art on many natural language understanding (NLU) tasks. More details about hyperparameter sweeps can be found in Appendix A.

We finetuned a pretrained T5-base model from HuggingFace (Wolf et al., 2020) on TellMeWhy. Since it is a natural language generation task related to a story, we use the SQuAD format specified in Appendix D.15 of Raffel et al. (2020) to format our inputs. Our narrative serves as the 'context' and the why-question is used as the 'question' in the selected input format. We train the model with batch size 16, learning rate 5e-5, maximum source length 75 and maximum answer length 30. The model is trained until the dev loss fails to improve for 3 iterations.

UnifiedQA (Khashabi et al., 2020) is a single pre-trained model that performs well across 20 different question answering datasets. It is built on top of a T5 model and simplifies finetuning by unifying the various formats used by T5. Its ability to perform both extractive and abstractive QA tasks makes it a suitable candidate for calibrating this task. A pretrained version of this model is available via HuggingFace (Wolf et al., 2020) under the name "allenai/unifiedqa-t5-base". The input format for this model is simple, just requiring the question and the narrative to be separated by a newline symbol. We train this model using learning rate 1e-5 (same as the original paper) and retain other hyperparameters from finetuning T5 as described above.

5.2 Automatic Evaluation

We evaluate all of the above models on both the test set and the hidden test set (questions from CATERS data). For automatic evaluation, we report BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), BLEURT (Sellam et al., 2020) scores using the bluert-base-128 checkpoint, and BertScore (Zhang* et al., 2020) using the default roberta-large checkpoint. These numbers are presented in Table 4.

Evaluated on	Model	BLEU	RG-L F1	BLEURT	BertScore
Full Test Set	GPT-2-OO	0.04	0.07	-1.30	0.08
	GPT2-FT	1.3	0.13	-0.96	0.11
	T5-OO	5.67	0.13	-1.20	0.14
	T5-FT	**13.33**	0.24	-0.70	**0.34**
	UnifiedQA	13.03	**0.25**	**-0.71**	0.30
Implicit-Answer Qs in Test Set	GPT-2-OO	0.07	0.06	-1.30	0.05
	GPT2-FT	1.39	0.12	-1.02	0.09
	T5-OO	3.12	0.11	-1.24	0.12
	T5-FT	**7.27**	0.17	**-0.89**	**0.27**
	UnifiedQA	6.63	**0.18**	-0.89	0.24

Table 4: Performance of models on the full test set and on implicit-answer questions in the test set using automated metrics. RG-L denotes ROUGE-L. The OO suffix denotes the vanilla version of the model while the FT version denotes the finetuned version.

We select one human answer at a time and (using SacreBLEU (Post, 2018)) calculate the BLEU scores for model output with all three references, and select the maximum. Since BLEURT is a sentence level metric, to calculate the reported BLEURT, we average all the (output, reference) scores to obtain a corpus score for each reference. We then select the maximum BLEURT corpus score over all 3 human references. It is important to note that BLEURT was proposed as a metric for relative comparison, not absolute calibration. We also report BertScore F1[3] (Zhang* et al., 2020) as another semantic automatic evaluation metric. We report a max BertScore in the same way as BLEURT and BLEU: by taking the maximum score of the model output with each human answer taken one at a time.

Vanilla model results are obtained by loading an existing pretrained model from HuggingFace and running inference with the input formats described above. They are not trained on TellMeWhy. We see that vanilla pretrained models are unable to perform this task at all. Finetuning a pretrained model results in improvements since it better models the relationship between the story, the question, and a possible answer. On the full test set, both the finetuned T5 and the UnifiedQA model perform the best on our task. However, the overall performance of these models remains poor. In Table 4, we also see that models perform a lot worse on implicit-answer questions.

5.3 Human Evaluation

For open-ended text generation tasks like answering why-questions, the absence of an automatic evaluation that correlates well with human judgments is a major challenge (Chen et al., 2019; Ma et al., 2019; Caglayan et al., 2020; Howcroft et al., 2020).

We conduct a human evaluation on the hidden test set with a standardized interface to compare different models. We want to measure whether a model produces coherent and grammatical output and more importantly, whether the produced output is a valid answer for the given question. Our validation HIT subsection 3.3 showed a way to conduct human evaluation of answers provided by other crowd-workers. We modified this HIT design to evaluate generated answers from models. For a given question, we present just one answer from a single model and then ask the crowd-workers to assess its grammaticality and validity.

For each story, question, and a model's answer, we ask 3 distinct annotators to provide judgments about grammaticality and validity. This serves as the human evaluation interface for our task. A sample HIT can be seen in Figure 1b.

We perform human evaluation of the fine-tuned versions of T5 and UnifiedQA, the two models that performed the best on automatic metrics. We evaluate the outputs of these models on the hidden test set. We calculated inter-annotator agreement for these judgments using the method described in subsection 3.3, and they were >80%, indicating high agreement. The models mostly produce grammatical answers, but fail to adequately ex-

---

[3] idf and rescale_with_baseline flags are set to True.

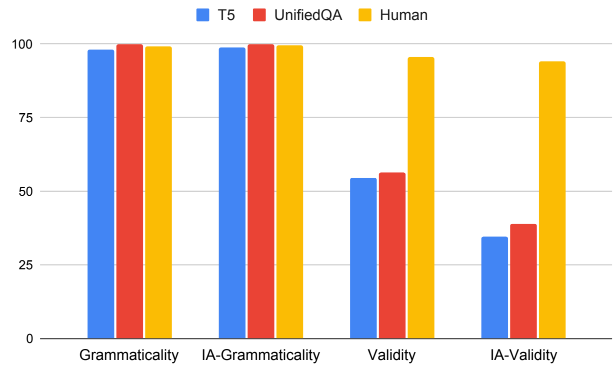

Figure 2: Human evaluated performance of answers. The IA prefix denotes performance on implicit-answer questions in the data.

plain many actions in the story. Figure 2 shows that, under human evaluation, models significantly under-perform humans at producing valid answers to why-questions. Models fare worse when the answers to the questions are external to the narrative.

Human evaluation is slow and expensive, so we performed a correlation analysis between the automated metrics and human judgments to gauge usefulness of popular automated metrics. Figure 3 shows that the embedding-based metrics are only weakly correlated with human validity judgments, while lexical metrics did even worse. None of the automatic metrics show a strong correlation, confirming our earlier assertion that human evaluation is the most appropriate way to analyze model performance on this open-ended generation task. BertScore has at least a moderate correlation with human validity judgments, and is therefore arguably the most useful for rapidly evaluating models during development. BLEU and BertScore improve their correlation with human judgments slightly as the number of human reference answers is increased; however, the increase is somewhat disappointing. Inexplicably, BLEURT's correlation actually decreases slightly with increasing human references, raising additional questions with respect to utilizing this metric. One possibility is that, by using an increasing number of references and taking the maximum score, BLEURT might overestimate the quality of answers as compared to human judgments.

Our human judgment interface can serve as a standard human evaluation of any future model's performance on our dataset, and we will make code available for automatically generating HITs for evaluating the outputs of any model. This standardized evaluation approach is similar in spirit to GENIE (Khashabi et al., 2021), a contemporary work that also presents an evaluation framework for a large set of generation tasks.

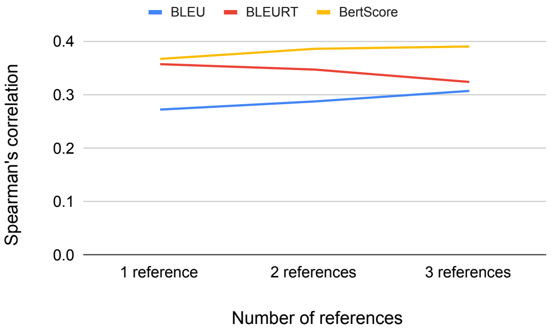

Figure 3: Correlation of automatic metrics with human validity judgment of model outputs. For each question, we have 3 crowdsourced human answers available to us. We selected a number of human answers randomly (X-axis) and calculated scores for each model output across different automatic metrics. Finally, we obtained Spearman's correlation (Y-axis) of these scores in comparison with Likert judgments provided by annotators for each human answer.

5.4 Analysis

In order to better understand when models are generating valid answers, we analyzed the correlation between model performance and a proxy for checking when human provided answers were in the input narrative. To this end, we aligned ROUGE F-1 scores with the lexical overlap of human answers and the story text. Figure 4a shows how ROUGE F-1 scores for our models increases as the lexical overlap also increases between the answers and corresponding story. The same is presented for BLEU in Figure 4b. Perhaps not surprisingly, this empirically shows that models do best when the answer is in the text, and suffer greatly when it is not (implicit answers). This further illustrates the value of TellMeWhy, as well as its challenge, that standard models are largely incapable of performing the reasoning needed to produce plausible answers that are assumed common knowledge by the story writer.

Table 5 also shows that the best performing models mainly learnt to copy complete or parts from the narrative to generate answers, treating this largely as an extractive task. On average, more than three-fourths of T5 and UnifiedQA's answers are

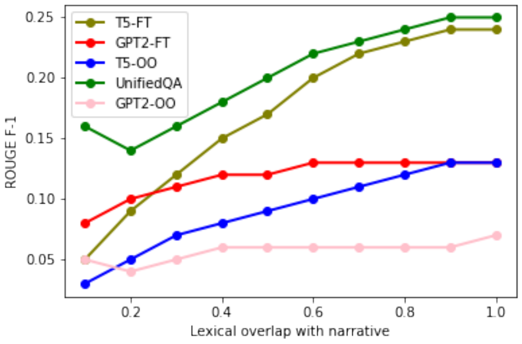

(a) ROUGE F-1 trend

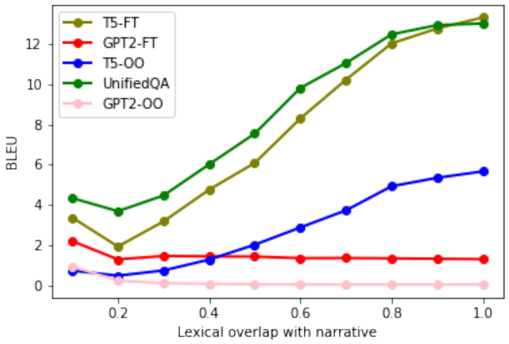

(b) BLEU trend

Figure 4: Model performance on different metrics with change in lexical overlap between a question's answers (as provided by humans) and the related narrative.

System	Copied ans	Avg overlap
Vanilla GPT2	0%	23.09%
Finetuned GPT2	0%	28.94%
Vanilla T5	5.50%	53.71%
Finetuned T5	59.44%	85.91%
UnifiedQA	27.44%	76.51%
Human Answers	35.03%	57.04%

Table 5: Overlap between answers and the original narrative. This indicate how much original text models produce.

based on words in the narrative text. T5 is worse compared to UnifiedQA in terms of copying, with a much larger fraction of questions (59.44% vs 27.44) with high lexical overlap (i.e. lexical overlap > 90%). In comparison, the average narrative overlap for human answers is much lower than the best-performing models, since people are able to infer answers that are not in the text. If the models are to successfully answer *why* questions, they need to look beyond copying texts.

6 Conclusion

This paper introduces a large, novel QA dataset, **TellMeWhy**, containing questions about *why* characters in a narrative perform their depicted actions. This challenge problem complements the variety of existing QA datasets, addressing the scarcity of "why" questions. Using both automated metrics and human evaluation, we show that existing deep-learned language models perform quite poorly at answering such questions. We also illustrate the uniqueness of this challenge where the answer is sometimes in the story itself, but often not, thus requiring a richer model that can draw on commonsense knowledge or external reasoning abilities.

We believe that progress on answering such questions requires new systems that can reason about actions, plans, and goals in order to achieve a deeper understanding of narrative text, as was initially argued over four decades ago (Schank and Abelson, 1977). We hope that TellMeWhy encourages further research in this area.

Acknowledgement

This material is based on research that is supported in part by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003 and in part by the National Science Foundation under the award IIS #2007290. The authors would like to thank the anonymous reviewers and the area chair for their feedback on this work. We would also like to thank Horace Liu for helping us run some experiments for the camera ready version.

References

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019.

Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Mohaddeseh Bastan, Mahnaz Koupaee, Youngseo Son, Richard Sicoli, and Niranjan Balasubramanian. 2020. Author's sentiment prediction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 604–615, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *International Conference on Learning Representations*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Ozan Caglayan, Pranava Madhyastha, and Lucia Specia. 2020. Curious case of language generation evaluation metrics: A cautionary tale. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2322–2328, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

Andrei Dulceanu, Thang Le Dinh, Walter Chang, Trung Bui, Doo Soon Kim, Manh Chien Vu, and Seokhwan Kim. 2018. PhotoshopQuiA: A corpus of non-factoid questions and answers for why-question answering. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jesse Dunietz, Greg Burnham, Akash Bharadwaj, Owen Rambow, Jennifer Chu-Carroll, and Dave Ferrucci. 2020. To test machine comprehension, start by defining comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7839–7859, Online. Association for Computational Linguistics.

J. Fukumoto, T. Kato, F. Masui, and T. Mori. 2007. An overview of the 4th question answering challenge (qac-4) at ntcir workshop 6. In *NTCIR*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 33–41, Sofia, Bulgaria. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2014. Is machine translation getting better over time? In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 443–451, Gothenburg, Sweden. Association for Computational Linguistics.

Ryuichiro Higashinaka and Hideki Isozaki. 2008. Corpus-based question answering for why-questions. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A. Smith, and Daniel S. Weld. 2021. Genie: A leaderboard for human-in-the-loop evaluation of text generation.

W. Kintsch and T. Dijk. 1978. Toward a model of text comprehension and production. *Psychological Review*, 85:363–394.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 62–90, Florence, Italy. Association for Computational Linguistics.

D. Marasini, P. Quatto, and E. Ripamonti. 2016. Assessing the inter-rater agreement for ordinal data through weighted indexes. *Statistical Methods in Medical Research*, 25:2611 – 2633.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016a. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016b. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California. Association for Computational Linguistics.

Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David Buchanan, Lauren Berkowitz, Or Biran, and Jennifer Chu-Carroll. 2020. GLUCOSE: GeneraLized and COntextualized story explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4569–4586, Online. Association for Computational Linguistics.

Joanna Mrozinski, Edward Whittaker, and Sadaoki Furui. 2008. Collecting a why-question corpus for development and evaluation of an automatic QA-system. In *Proceedings of ACL-08: HLT*, pages 443–451, Columbus, Ohio. Association for Computational Linguistics.

Allen Nie, Erin Bennett, and Noah Goodman. 2019. Learning to explain: Answering why-questions via rephrasing. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 113–120, Florence, Italy. Association for Computational Linguistics.

Peter Norvig. 1987. Inference in text understanding. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 2*, AAAI'87, page 561–565. AAAI Press.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2018. RankME: Reliable human ratings for natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 72–78, New Orleans, Louisiana. Association for Computational Linguistics.

Jong-Hoon Oh, Kazuma Kadowaki, Julien Kloetzer, Ryu Iida, and Kentaro Torisawa. 2019. Open-domain why-question answering with adversarial learning to encode answer texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4227–4237, Florence, Italy. Association for Computational Linguistics.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Takuya Kawada, Stijn De Saeger, Jun'ichi Kazama, and Yiou Wang. 2012. Why question answering using sentiment analysis and word classes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 368–378, Jeju Island, Korea. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229, Gothenburg, Sweden. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020. Getting closer to ai complete question answering: A set of prerequisite real tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8722–8731.

Keisuke Sakaguchi and Benjamin Van Durme. 2018. Efficient online scalar annotation with bounded support. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 208–218, Melbourne, Australia. Association for Computational Linguistics.

Sashank Santhanam and Samira Shaikh. 2019. Towards best experiment design for evaluating dialogue system output. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 88–94, Tokyo, Japan. Association for Computational Linguistics.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3027–3035.

R. Schank and R. P. Abelson. 1977. Scripts, plans, goals and understanding: an inquiry into human knowledge structures.

Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, page 151–157, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

João Sedoc, Daphne Ippolito, Arun Kirubarajan, Jai Thirani, Lyle Ungar, and Chris Callison-Burch. 2019. ChatEval: A tool for chatbot evaluation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 60–65, Minneapolis, Minnesota. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2006. Data for question answering: The case of why. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 735–736, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Elizabeth Clark, Lianhui Qin, Ali Farhadi, and Yejin Choi. 2020. Evaluating machines by their real-world language use. *arXiv preprint arXiv:2004.03607*.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Rolf A. Zwaan, Mark C. Langston, and Arthur C. Graesser. 1995. The construction of situation models in narrative comprehension: An event-indexing model. *Psychological Science*, 6(5):292–297.

A Hyperparameter Sweep

We describe the hyperparameters and the range of values we experimented with. The best hyperparameters are chosen on the basis of model loss on the validation set. For both GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020), we conduct guided sweeps for learning rate, batch size and epochs. We experiment with 1e-5, 5e-5 and 1e-4 for learning rate. Batch sizes of 8, 16 and 32 were tried. Models were trained for 20, 30 and 50 epochs, and we found that models converged between 30 and 50 epochs. In the case of T5, we also experiment with different lengths of inputs and target outputs. We trained models with maximum source lengths of 50, 60 and 75 tokens. For target length, we experimented with 15, 25 and 30 tokens. The maximum output length is treated as a hyperparameter for GPT-2, and we tried 15, 20, 25 and 30 tokens.

B Dataset Creation

The method described in subsection 3.1 creates 489 questions from the 200 stories in the CATERS dataset – 36 stories with 1 question, 63 with 2, 59 with 3, 30 with 4, and 6 with 5. We collect 3 human answers for all questions. For ROCStories, this creates 113,213 questions from 45,496 stories – 7,555 stories with 1 question, 13,431 with 2, 13,349 with 3, 7356 with 4, and 1865 with 5. We randomly select 32,165 questions from stories with 3 or 5 questions, for ease and efficiency of collecting annotations. This is the smallest number for which we could gather 3 answers for at least 30,000 questions, which is a reasonable-sized dataset for training or fine-tuning large NLP models.

C Mechanical Turk tasks

C.1 Instructions

We present the instructions given to annotators for both the tasks in Figure 5. Annotators were given clear direction for both tasks. We restricted both tasks to master turkers. The second task (answer validity) was also used a sanity check for answers collected in the first task (answer collection). Using results of the answer validity task (mentioned in subsection 3.3), we see that humans provided high quality answers in the answer curation task.

C.2 Inter-annotator agreement

We use weighted Fleiss Kappa to calculate inter-rater reliability. The weights between different classes are shown in Table 6 where negative, slightly negative, neutral, slightly positive, and positive classes are shown with -2, -1, 0, 1, and 2. We follow the setup used in Bastan et al. (2020) for a similar multi-class labeling task.

	-2	-1	0	1	2
-2	1	$\cos \pi/8$	$\cos \pi/4$	$\cos 3\pi/8$	0
-1	$\cos \pi/8$	1	$\cos \pi/8$	$\cos \pi/4$	$\cos 3\pi/8$
0	$\cos \pi/4$	$\cos \pi/8$	1	$\cos \pi/8$	$\cos \pi/4$
1	$\cos 3\pi/8$	$\cos \pi/4$	$\cos \pi/8$	1	$\cos \pi/8$
2	0	$\cos 3\pi/8$	$\cos \pi/4$	$\cos \pi/8$	1

Table 6: Inter class weights used for computing inter annotated agreement

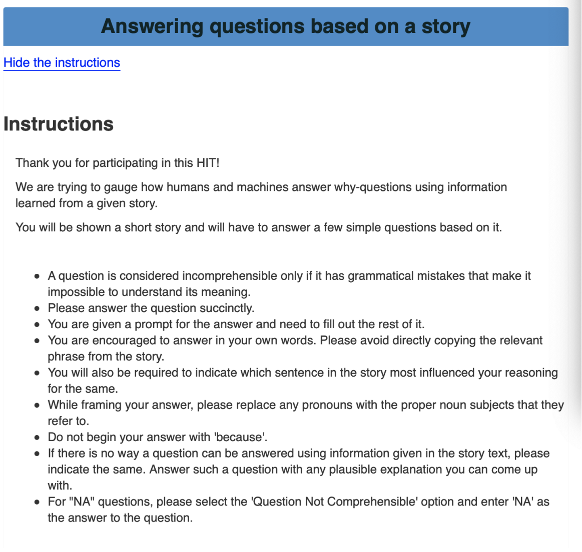

Answering questions based on a story

Hide the instructions

Instructions

Thank you for participating in this HIT!

We are trying to gauge how humans and machines answer why-questions using information learned from a given story.

You will be shown a short story and will have to answer a few simple questions based on it.

- A question is considered incomprehensible only if it has grammatical mistakes that make it impossible to understand its meaning.
- Please answer the question succinctly.
- You are given a prompt for the answer and need to fill out the rest of it.
- You are encouraged to answer in your own words. Please avoid directly copying the relevant phrase from the story.
- You will also be required to indicate which sentence in the story most influenced your reasoning for the same.
- While framing your answer, please replace any pronouns with the proper noun subjects that they refer to.
- Do not begin your answer with 'because'.
- If there is no way a question can be answered using information given in the story text, please indicate the same. Answer such a question with any plausible explanation you can come up with.
- For "NA" questions, please select the 'Question Not Comprehensible' option and enter 'NA' as the answer to the question.

(a) Instructions for answer collection task

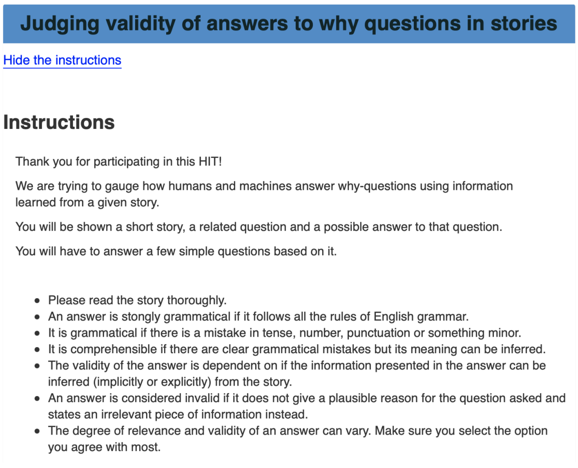

Judging validity of answers to why questions in stories

Hide the instructions

Instructions

Thank you for participating in this HIT!

We are trying to gauge how humans and machines answer why-questions using information learned from a given story.

You will be shown a short story, a related question and a possible answer to that question.

You will have to answer a few simple questions based on it.

- Please read the story thoroughly.
- An answer is stongly grammatical if it follows all the rules of English grammar.
- It is grammatical if there is a mistake in tense, number, punctuation or something minor.
- It is comprehensible if there are clear grammatical mistakes but its meaning can be inferred.
- The validity of the answer is dependent on if the information presented in the answer can be inferred (implicitly or explicitly) from the story.
- An answer is considered invalid if it does not give a plausible reason for the question asked and states an irrelevant piece of information instead.
- The degree of relevance and validity of an answer can vary. Make sure you select the option you agree with most.

(b) Instructions for answer validation task

Figure 5: Instructions for MTurk tasks

Dialogue in the Wild:
Learning from a Deployed Role-Playing Game with Humans and Bots

Kurt Shuster* **Jack Urbanek*** **Emily Dinan** **Arthur Szlam** **Jason Weston**

Facebook AI Research

Abstract

Much of NLP research has focused on crowdsourced static datasets and the supervised learning paradigm of training once and then evaluating test performance. As argued in de Vries et al. (2020), crowdsourced data has the issues of lack of naturalness and relevance to real-world use cases, while the static dataset paradigm does not allow for a model to learn from its experiences of using language (Silver et al., 2013). In contrast, one might hope for machine learning systems that become more useful as they interact with people. In this work, we build and deploy a role-playing game, whereby human players converse with learning agents situated in an open-domain fantasy world. We show that by training models on the conversations they have with humans in the game the models progressively improve, as measured by automatic metrics and online engagement scores. This learning is shown to be more efficient than crowdsourced data when applied to conversations with real users, as well as being far cheaper to collect.

1 Introduction

Humans learn to use language over the course of their lives from the interactions they have with the world and other people. Yet, the prevailing dominant paradigm in natural language processing (NLP) research is to build a fixed dataset from which to train a model and then freeze it, without any ability for the model to interact with humans using language at training time at all. While we need such interaction in order to study human-machine communication to its full extent, constraints usually inhibit such research. Firstly, conducting such experiments can be costly. Many datasets in NLP are collected with crowdsourcing, whereby one pays the crowdworkers to perform interaction and annotation tasks. This leads to several issues, not least that research budgets for paying crowdworkers mean that data will have a limit. Secondly, as crowdworkers are motivated by pay, not by interest in the tasks themselves, the data distribution may not match the desired one (de Vries et al., 2020).

In this work we study the ability of dialogue agents in an open world[1] to iteratively learn from conversations with intrinsically motivated humans. In order to engage humans at scale, we build and deploy a (free to play) game with a purpose (Von Ahn, 2006) whereby human players role-play characters and converse with other characters (that are our learning models) situated within the game world. We choose a rich fantasy game world, in order to maximize engagement. Our system iterates between collecting data of human-model interactions, retraining updated models on the newly collected data, and redeploying them. Simultaneously, it provides a natural metric to evaluate and compare models online using human continue rates (that is, how long human players continue playing).

We show that we can successfully collect, retrain and redeploy models, starting from a crowdsourced base model, that improve both offline automatic metrics and human continue rates. Our overall system is then engaging enough that we can collect data at a rate that is $1/5^{th}$ of the price per utterance of crowdsourcing, where the cost of our method is the cost of advertisements that make players aware of the game. Moreover, the data we collect is also more effective per utterance at improving continue rates due to being more on-distribution than crowdsourced data. As our models improve, these rates improve as well, as the continuation rate increases

---

* Equal Contribution.

[1] In this work we study dialogue that can be about any topic but within the scope of a fantasy game world. Note this differs from open-domain dialogue talking about our world, e.g. the game players can talk about the sauce recipe from Bredwell across the sea (see Fig. 1), but not about the pizza in Chicago.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 611–624
August 1–6, 2021. ©2021 Association for Computational Linguistics

– meaning relatively more data is collected. Overall, our work provides good evidence that lifelong dialogue learning in deployed systems with intrinsically motivated humans (rather than crowdworkers) can be successful, in particular by embedding such learning within games.

The training code and parameters of the models deployed, and the data collected in this work will be made publicly available for reproducibility and further research by the community.

2 Related Work

Open-Domain Dialogue Dialogue in the open-domain (chitchat) setting, which involves chat about any topic, rather than a specific goal-directed topic, is commonly studied in the train/valid/test static dataset paradigm utilizing supervised learning. A number of crowdsourced or scraped datasets have been developed to that end, including Daily Dialogue (Li et al., 2017), PersonaChat (Zhang et al., 2018), Empathetic Dialogues (Rashkin et al., 2019) and Wizard of Wikipedia (Dinan et al., 2019b); see Huang et al. (2020) for a review.

LIGHT In this work we specifically focus on dialogue setting of LIGHT (Urbanek et al., 2019). LIGHT focuses on situated characters playing character roles that can chat about any topic, within the context of a medieval fantasy world. This setting is known to be engaging for human role-players, and also alleviates some safety concerns in that the role-playing means they should not divulge personally identifying information. The authors crowdsourced a dialogue dataset consisting of 8.5k episodes and 111k utterances, which they publicly released. We refer to this as LIGHT MTurk data, or LIGHT data for short, in the rest of this paper. In this work we utilize this data to build a deployed system whereby players can converse with models, and we can study lifelong learning with these models using the information in these new conversations.

Lifelong Learning Lifelong learning is a machine learning paradigm whereby deployed models can interact with the world and iteratively improve themselves from the things they learn, eschewing the standard approach of a fixed training set from which a model is trained once (Silver et al., 2013). We note there are other closely related concepts to the topics in this work, such as incremental learning (Castro et al., 2018), continual reinforcement learning (Ring, 1994) and never-ending learning (Carlson et al., 2010; Mitchell et al., 2018).

Interactive Dialogue Learning Learning from dialogue interaction is common in reinforcement learning settings, where the feedback is a scalar rather than solely the dialogue messages themselves (Levin et al., 2000; Schatzmann et al., 2006; Rieser and Lemon, 2011; Liu and Lane, 2017; Serban et al., 2017), which is most common in a goal-oriented setting where completion of the goal can provide such rewards, but not so in chitchat settings (Roller et al., 2020), which do not have rewards.

Closer to our work, is the self-feeding chatbot (Hancock et al., 2019), where dialogue models were used to collect data to improve themselves via crowdsourcing utilizing the PersonaChat task. Related approaches have also been applied to the cases of question answering (Li et al., 2016a,b), and in simulators (Mazumder et al., 2019; Nguyen and Daumé III, 2019) as well. Liu et al. (2018) applied such an approach to goal-oriented tasks. Our work differs from these works in that we study a deployed user-facing system in an open-ended game environment, rather than more limited data from paid crowdworkers. For other related works see Padmakumar and Mooney (2020).

Deployed Dialogue Systems While there are a number of deployed virtual assistants, many of these products are not ideal platforms for the research community. Their proprietary nature and commercial importance, coupled with privacy concerns, means they are neither accessible to researchers, nor amenable to public reproducible research. A near-exception is the Alexa challenge (Ram et al., 2018) which allows university-based researchers access to a commercial user-base for the span of the competition, however, the data and models are also not released to the rest of the research community, unlike in our work.

3 Dialogue as a role-playing game

In this section we describe the game that we will build and deploy, which is a role-playing game, designed to both train and evaluate dialogue agents.

Core Mini-Game The core game involves pairing two agents in a given setting – where one is a human player and the other is a dialogue agent with an underlying machine learning model. The two players are assigned characters, with given names and backstories (personas), and their current location and its description. See Figure 1 for examples.

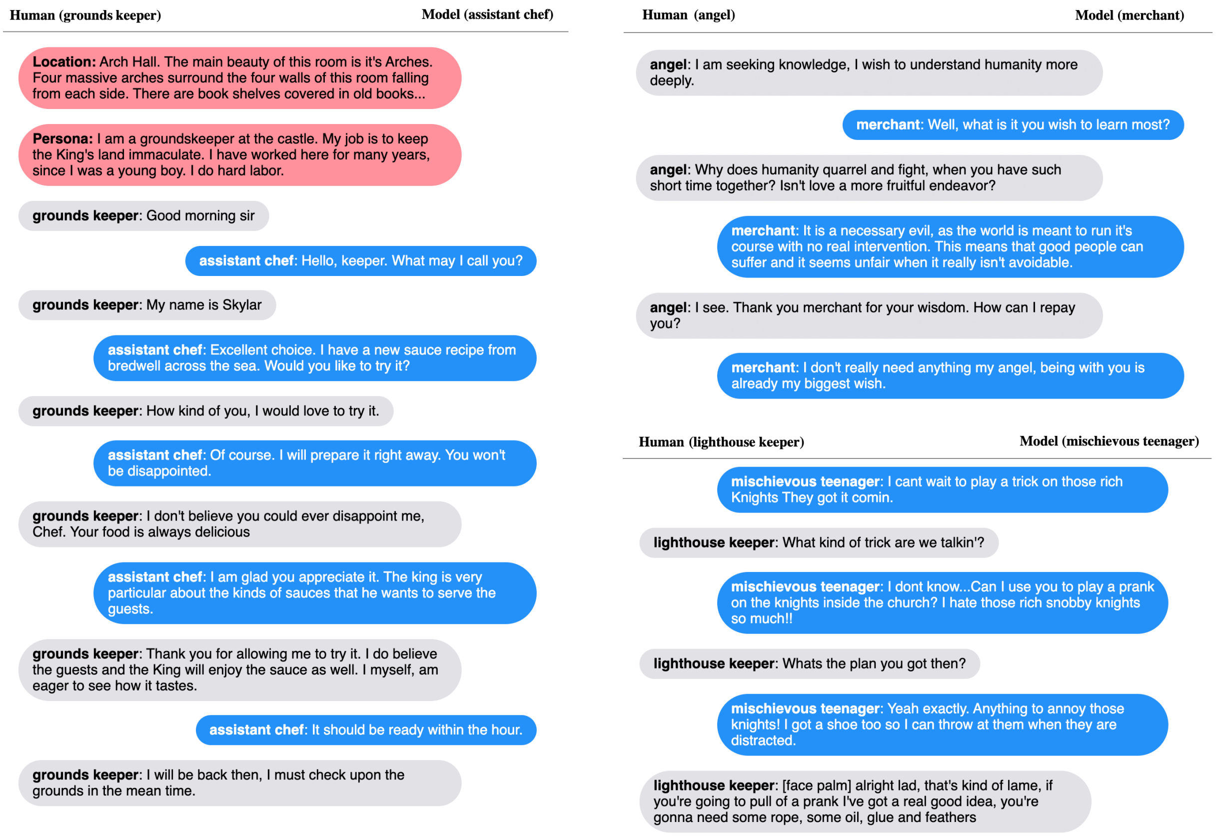

Human (grounds keeper) Model (assistant chef)

Location: Arch Hall. The main beauty of this room is it's Arches. Four massive arches surround the four walls of this room falling from each side. There are book shelves covered in old books...

Persona: I am a groundskeeper at the castle. My job is to keep the King's land immaculate. I have worked here for many years, since I was a young boy. I do hard labor.

grounds keeper: Good morning sir

assistant chef: Hello, keeper. What may I call you?

grounds keeper: My name is Skylar

assistant chef: Excellent choice. I have a new sauce recipe from bredwell across the sea. Would you like to try it?

grounds keeper: How kind of you, I would love to try it.

assistant chef: Of course. I will prepare it right away. You won't be disappointed.

grounds keeper: I don't believe you could ever disappoint me, Chef. Your food is always delicious

assistant chef: I am glad you appreciate it. The king is very particular about the kinds of sauces that he wants to serve the guests.

grounds keeper: Thank you for allowing me to try it. I do believe the guests and the King will enjoy the sauce as well. I myself, am eager to see how it tastes.

assistant chef: It should be ready within the hour.

grounds keeper: I will be back then, I must check upon the grounds in the mean time.

Human (angel) Model (merchant)

angel: I am seeking knowledge, I wish to understand humanity more deeply.

merchant: Well, what is it you wish to learn most?

angel: Why does humanity quarrel and fight, when you have such short time together? Isn't love a more fruitful endeavor?

merchant: It is a necessary evil, as the world is meant to run it's course with no real intervention. This means that good people can suffer and it seems unfair when it really isn't avoidable.

angel: I see. Thank you merchant for your wisdom. How can I repay you?

merchant: I don't really need anything my angel, being with you is already my biggest wish.

Human (lighthouse keeper) Model (mischievous teenager)

mischievous teenager: I cant wait to play a trick on those rich Knights They got it comin.

lighthouse keeper: What kind of trick are we talkin'?

mischievous teenager: I dont know...Can I use you to play a prank on the knights inside the church? I hate those rich snobby knights so much!!

lighthouse keeper: Whats the plan you got then?

mischievous teenager: Yeah exactly. Anything to annoy those knights! I got a shoe too so I can throw at them when they are distracted.

lighthouse keeper: [face palm] alright lad, that's kind of lame, if you're going to pull of a prank I've got a real good idea, you're gonna need some rope, some oil, glue and feathers

Figure 1: Example collected dialogues from humans role-playing in our deployed system, conversing with models. (Left) a conversation complete with described location and player persona; (Right) excerpts from two other conversations (out of 41,131 collected) to demonstrate the diversity of the dialogue in the fantasy game world.

Each player's goal is to act out (role-play) their character's dialogue in the given situation. We refer to one such dialogue episode as a *mini-game*. Dialogue in the game is in English.

Role-Playing (Acting) Score We take advantage that role-playing is a pursuit that a large number of human players find fun (Horsfall and Oikonomou, 2011), and are hence naturally engaged in the open-ended nature of this process. However, to encourage and further motivate players to play their characters well, we introduce the concept of an (automated) dungeon master (DM), who will assess the quality of the player's role-playing. For each dialogue turn, we apply a learned model to the human player's dialogue, which assesses how likely their utterance is given the context. We convert this to a score, between 1 and 5 stars, that is presented to the human player, to reward them for good acting. While this signal is noisy, because our DM model is not perfect, it gives motivating feedback to the players to continue playing.

Game Loop Each dialogue (mini-game) consists of 6 turns of dialogue per agent (12 total). At the end of the mini-game the human player is presented with four choices: (i) choose to move to a new location, where they will continue to play this character, but meet a new character to converse with; (ii) stay in the same room but wait for a new character to arrive to converse with; (iii) change to role-play a completely new pair of characters in a new setting; or (iv) end the game. These choices encourage the player to choose another mini-game that they are most interested in, and the variety of mini-games gives different role-playing possibilities, making the dialogue data more diverse. Further details of the game are described in Appendix A.

Deployment and Advertising We deployed the game using Facebook Messenger on CPU-only servers, and used the publicly-available ad flow to advertise on Facebook. The target audience for our ads were people with interests in online gaming, role playing, and role playing games in particular. We also used lookalike audiences to those who engaged with the game. We ran ads like that shown in Figure 4 with a call-to-action for people to try out the game.

4 Lifelong Dialogue Learning

4.1 Models

Retrieval Models All the models we have currently deployed are retrieval models for safety reasons as we can vet each possible response by filtering the train set candidates (see Appendix A). In particular, we use the Poly-Encoder (PE) Transformer architecture as a base (Humeau et al., 2019), as it provides state of the art results compared to other retrieval models, whilst being tractable to deploy. PE encodes the context with a standard bidirectional transformer, but produces an encoding into a fixed small number of codes, N. We tried values of $N = 5$ and $N = 20$. Each label candidate then attends to these codes before producing a final matching score. The model is trained with cross-entropy given the correct label, and by subsampling negative examples from the given training batch.

Architecture and Training Choices We employ the 90M and 622M parameter models from (Roller et al., 2020) that have been pre-trained on 1.5B training examples from pushshift.io Reddit, which we then fine-tune. We also consider two other enhancements, chosen to mitigate problems that we observed with the models: (i) negative context training, whereby negatives are also selected from the immediate dialogue history as well as the batch which can help reduce a model's tendency to repeat itself (Holtzman et al., 2019; Welleck et al., 2020); and (ii) decoding control (See et al., 2019) whereby at decoding time responses are rescaled before scoring based on their specificity (normalized inverse document frequency). The latter can control the genericness of the responses, which is known to affect human judgments.

Generative Models In addition to the deployed models, we also train and evaluate generative models offline, where safety concerns are less important as the models are not user-facing. We employ an encoder-decoder Transformer architecture using the state of the art pre-trained 2.7 billion parameter BlenderBot model (Roller et al., 2020), which we fine-tune on our task.

Acting Score Model We can apply a retrieval model to also score the human's role-playing abilities. In this case, the context is the entire dialogue history, setting and the player's character persona as input to the encoder, while the candidates to score are the ones from the training set, as usual, plus additionally the human's (player's) actual response. The score given to the user is then proportional to the human response's rank amongst all the candidates[2].

4.2 Iterative Data Collection and Training

After collecting a certain amount of episodes of conversational data between humans and models, one can consider using this data for training. We utilize the following observation: while the model utterances may contain many mistakes, it is assumed that a human sufficiently engaged provides high quality responses, even if the model responses are mistakes, and can thus be treated as gold, and used as a fully supervised signal. We thus separate the dialogue data into all possible (context, next utterance) pairs, and then only consider the pairs with human next utterances as training data. We also compare this to further filtering this set by scoring the quality of the human utterances, discarding those episodes (mini-games) with lower quality. We use the acting score model previously described for this purpose, summing the scores obtained across an episode, and discarding the episode if this value is less than C, where C is a hyperparameter tuned on the validation set.

After training our model from a given round of collection, we can go back to the collection process utilizing instead the new model that has been trained on more data. The hypothesis is that the higher quality the model is: (i) the higher quality the human data will be as well; and (ii) the more likely the human players are to converse longer, increasing the data set size by larger amounts.

4.3 Deployment-based Evaluation

Apart from the collection-training cycle of our deployed lifelong learning setup, one can also in parallel perform evaluation. For each separate mini-game (episode of dialogue) we can potentially deploy a *different* model for human-model conversation. We maintain a pool of models with differing architectures or hyperparameters, and select randomly from the pool in each episode. For any given episode we record whether the player continued playing to the next mini-game or not, which

---

[2] The player is awarded 2 stars if their response is in the top 2000, 3 stars in the top 1000, and 4 stars in the top 100.

we refer to as the continue rate. We can measure the quality of a model using its averaged continue rate over all players and episodes. In this way we can also perform model selection online.

5 Experiments

5.1 Rounds of Learning

We performed three rounds of our lifelong learning setup.

Round 1 consists of models trained on LIGHT MTurk data only. We train the retrieval model variants described in Section 4.1, and deploy them within the game.

Round 2 consists of models trained on LIGHT MTurk data + 50,982 examples collected from the deployment of the Round 1 models, and again deploy these within the game.

Round 3 consists of models trained on LIGHT MTurk data + 50,982 examples from Round 1 deployment + an additional 180,010 examples collected from Round 2 deployment.

5.2 Data Collection

While our setup is a lifelong learning setup and the models are still currently deployed and collecting data, for this paper we froze the collection at a given point in order to provide a data release and provide experimental results. The data statistics for the total newly collected dataset, called LIGHT WILD, over all rounds is shown in Table 1. Validation and test sets were extracted from a portion of the data[3] from Round 2.

Table 2 compares this dataset to several existing commonly used open-domain dialogue datasets. The number of episodes and dialogue utterances is larger than many existing datasets, e.g. four times as many as LIGHT MTurk, and almost eight times that of Empathetic Dialog. Uniquely, our dataset contains human-model conversations, hence the total number of human utterances is actually half of the utterances, which is still twice as large as the number in LIGHT MTurk. Our dataset also has a large degree of diversity, which is important for tasks in general, and especially for open-domain dialogue. The number of unique locations and roles that can be played by speakers (characters) is large (587 and 630, respectively). The number of players of the game at the time of freezing was over 13,000, which also makes the diversity far larger than typical crowdsourced datasets, e.g. LIGHT MTurk involved 1,052 crowdworkers and Empathetic Dialog involved 810 crowdworkers. Finally, the number of unique tokens is larger in LIGHT WILD, indicating its diversity of language.

5.3 Analysis of Results

5.3.1 Performance by Round

While we only deployed retrieval models, we report experiments training both retrieval models and generative models on the data from the three rounds, selecting best hyperparameters using the validation set. We report the performance on three different test sets: LIGHT (MTurk) Seen and Unseen test sets (Urbanek et al., 2019), where unseen means that the test locations do not overlap with the training set locations, and our WILD test set. The results are given in Table 3. They show a steady increase in the Hits@1/20 metric (Top 1 accuracy given 19 random distractors) for the retrieval models over the rounds on all three test sets, and a similar decrease in perplexity (PPL) for the generative models. In particular there is a large jump in the performance on the WILD Test set between Rounds 1 and 2 as the training set switches from crowdsourced to in-distribution WILD data, and a further increase in Round 3 as more data is again collected and retrained on. While our WILD data is of a different distribution to the two LIGHT (MTurk) test sets, the data collection from our lifelong learning setup still gives gains on those tests as well. Our reported numbers, as far as we are aware, are the best reported numbers on these datasets, e.g. the original LIGHT paper reports 76.5% and 70.5% for the Seen and Unseen test sets, respectively (compared to our 87.72% and 83.48%). Overall, we see clear gains from the extra data collected in our setup.

5.3.2 Lifelong Learning Curves

We construct learning curves given all the collected data to analyze the performance gain per new training example. We plot Hits@1/20 accuracy on the WILD validation set against the number of training examples, comparing data from WILD collection to LIGHT (Mturk). We also consider a 50/50 mix, where we equally sample from LIGHT MTurk and WILD to provide the next training example.

Figure 2 (left) shows the results. We observe that on a per-example basis our WILD data gives

---

[3] For validation and test we only use complete conversations, and where the player scored ≥ 9 stars, to build higher quality evaluation sets.

Data Type	Num. Epsiodes	Num. Utterances	Num. Human Utterances	Unique Locations	Unique Characters
Training	41,131	461, 984	230,992	587	630
Validation	500	5,936	2,968	231	463
Test	1000	11,822	5,911	296	569

Table 1: Data statistics of our lifelong learning deployment at the point where we froze collection for experiments reported within the paper and subsequent data release.

Dataset	Num. Episodes	Num. Utterances	Num. Human Utterances	Unique Tokens	Avg. Human Utt. Length	Number of Humans
PersonaChat (Zhang et al., 2018)	8,939	131,438	131,438	18,688	11.9	UNKNOWN
Wiz. of Wikipedia (Dinan et al., 2019b)	18,430	166,787	166,787	52,490	19.7	UNKNOWN
Empathetic Dialog (Rashkin et al., 2019)	24,850	64,636	64,636	19,458	15.3	810
Daily Dialog (Li et al., 2017)	22,236	87,170	87,170	20,673	14.5	UNKNOWN
LIGHT MTurk (Urbanek et al., 2019)	8,538	110,877	110,877	33,789	18.3	1,052
LIGHT WILD *(this paper)*	41,131	461,984	230,992	47,526	11.9	13,188

Table 2: Comparison of statistics of the dialogue data collected during deployment (bottom row) compared to several existing (mostly crowdsourced) datasets. Our data is around twice as large in terms of human utterances than these datasets, and 4x as large in terms of dialogue utterances (as our data consists of human-model conversations), while the cost to collect our data was only $1/5^{th}$ of the price per utterance of LIGHT MTurk, see Sec. 5.3.3.

more accuracy gains than LIGHT MTurk data, e.g. 83.59% for WILD compared to 80.63% for LIGHT, when limiting WILD to the same training set size as the total size of LIGHT. As the WILD dataset is more than twice as large this monotonically improves further, up to 85.95% using all of the WILD data. We observe that the improvements have not saturated and that further lifelong learning will bring further model improvements. Combining the two data sources, as shown in the LIGHT+WILD plot, brings yet further gains, up to 87.2%. Overall, our collected WILD data has high quality as a learning signal for training models.

5.3.3 Cost Learning Curves

We plot similar learning curves, but as a function of the cost to collect the data instead of the number of training examples instead, see Figure 2 (right). Although we do not pay players to play the game, we did spend money to advertise the game online in order to attract players. We compare the cost per WILD example relatively to the cost per LIGHT (MTurk) example, where the x-axis is scaled in units that are multiples of the cost required to achieve 80.63% using WILD data (as this is the performance of using all the LIGHT MTurk data together). We observe that it costs over 8x more to achieve the same accuracy using LIGHT (MTurk) data (see dashed horizontal line). The actual cost per utterance of WILD data is around $1/5^{th}$ of the price of LIGHT MTurk data[4], but more than that, it is also relatively more effective per utterance in terms of metrics. For the same amount spent there is a large gap between the two systems, for example using all the WILD data gives a performance of 85.95%, whereas for the same spend LIGHT MTurk only achieves ∼77.5%. Overall, WILD deployment is relatively a very cost effective strategy.

5.3.4 Deployment-based Evaluation

Our lifelong learning setup deploys multiple models (see Sec. 4.3) at the same time randomly assigning them to concurrent users per episode (minigame). We can thus directly compare the quality of models via their continue rate.

Continue rates during Round 2 of collection comparing several model variants are given in Table 4. Continue rates are in the range of 68%-75%, depending on the model, and we observe some clear trends. Most importantly, LIGHT+WILD trained models are superior to LIGHT only trained models, showing that our deployment/train cycle produces better models as judged by humans. Secondly, other factors in model design are important too, and our setup can effectively evaluate those. In particular, it was found that both our negative context training and decoding control enhancements (see Sec. 4.1) improve the continue rate, with both

---

[4] This is the cost of the 462k WILD utterances starting from a crowdsourced model base trained on 111k LIGHT utterances giving 573k utterances in total (see §5.1), comparing to collecting an additional 462k examples via crowdsourcing.

	Retrieval Model (Hits@1/20 ⇑)			Generative Model (PPL ⇓)		
Model	LIGHT Test	LIGHT Test Unseen	WILD Test	LIGHT Test	LIGHT Test Unseen	WILD Test
Round 1	87.12	82.43	81.61	12.67	11.81	13.42
Round 2	87.65	82.70	84.60	12.57	11.74	12.31
Round 3	87.72	83.48	87.63	12.54	11.75	11.79

Table 3: Three rounds of training in our lifelong dialogue learning setup. Both retrieval and generative models trained on the data from the three rounds improve across both metrics on all three test sets.

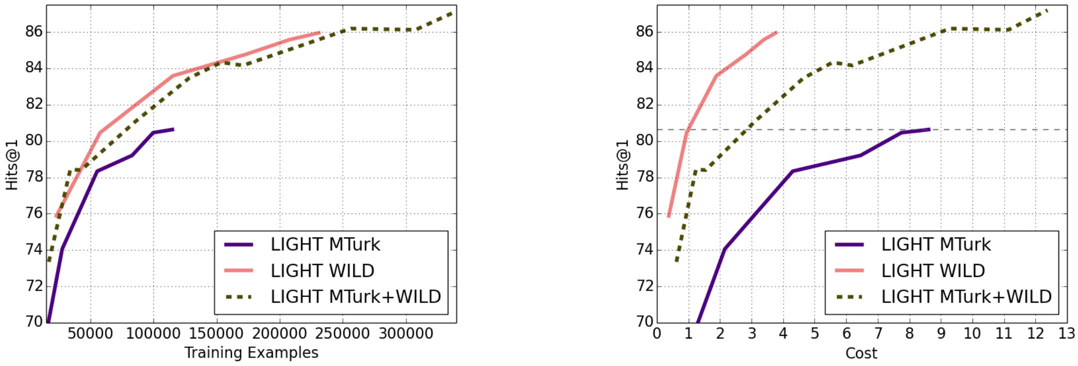

Figure 2: Hits@1/20 Accuracy on the LIGHT WILD validation set as a function of the number of training examples (left) or the cost of data collection (right). The cost axis is in units scaled by the cost of LIGHT WILD collection required to achieve the same performance as using the entire LIGHT MTurk dataset; it is more than 8× cheaper to use LIGHT WILD examples than LIGHT MTurk examples to achieve an accuracy of 80.63%. We also show performance for models which equally sample data from LIGHT MTurk+WILD datasets for training; utilizing all the data from both sources yields the best performance. However, LIGHT WILD data gives better accuracy improvements per training example (left plot).

Data	Neg.Ctxt	Dec.Ctrl	Continue Rate
LIGHT	yes	no	72.2 ± 1.9%
LIGHT	yes	yes	74.1 ± 2.0%
LIGHT+WILD	yes	no	73.6 ± 1.8%
LIGHT+WILD	yes	yes	75.2 ± 2.0%

Table 4: Deployment-based Evaluation, comparing several metrics on data collected during Round 2 of collection for 90M PE models.

Model variation	Δ Continue Rate
+ WILD train data (Round 2)	+1.3 ± 0.7%
90M → 622M parameters PE	−3.2 ± 0.7%
+ Negative context training	+2.6 ± 1.3%
+ Decoding control	+2.5 ± 1.1%

Table 5: Deployment-based Evaluation: changes in continue rates for various model variants.

methods used together improving more. We confirm these conclusions in Table 5 where we show the change in continue rates when independently adjusting one of these factors, by averaging over model continue rates for other factors of variation.

We also observe the unexpected result that the larger models perform worse than the small models across the board on continue rates. Deeper analysis given in appendix D suggests that while the larger model makes less mistakes, it is more often seen as boring, which impacts a player's desire to continue playing. Understanding and controlling this trade-off should be studied further.

5.3.5 Data Quality

Not every player is as engaged in the game as every other player, or produces as high quality dialogue. We hypothesize that we can predict which players produce higher quality data via the acting score model (Sec. 4.1), and that such higher quality data is relatively better for training models.

Figure 3 (right) shows the distribution over the WILD training set of predicted quality using the acting score model. We observe 83.7% of the episodes have a score above the minimum value of 6 (there are 6 turns, and on each turn a score between 1-4 is awarded, explaining the spike at the value of 6). Scores below 6 indicate incomplete dialogues, which only account for 4.0% of the data.

To assess whether these scores are indeed indicators of data quality, we selected an equal amount

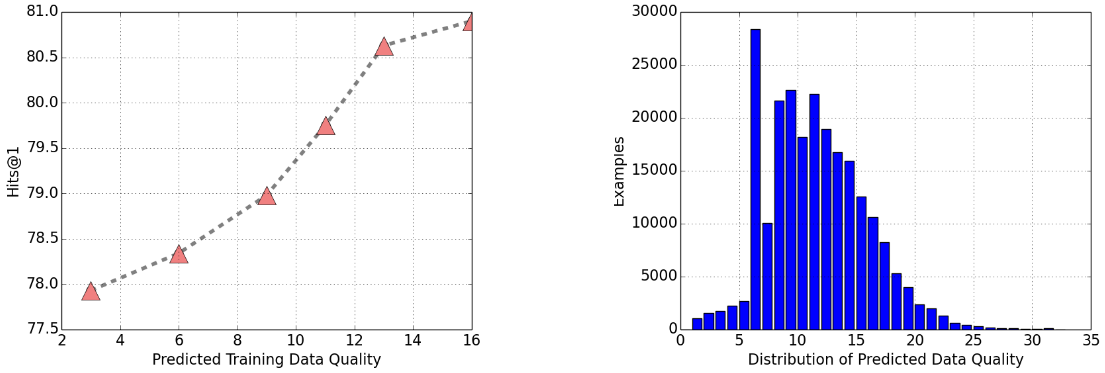

Figure 3: **Predicted Data Quality**. Left: Hits@1/20 accuracy on the WILD validation set when training with LIGHT MTurk + 10,000 examples from the WILD training set of a given predicted quality level, see Sec. 5.3.5. Data that is predicted to be higher quality yields improved validation accuracies. Right: The distribution of data quality predictions over the training set. A spike is seen at quality bin 6 because that is the lowest score one can achieve when completing a full episode (1 star per turn is awarded at minimum). Values lower than bin 5 indicate incomplete low-scoring episodes.

of 10,000 examples from each of the bins (1-5), (6), (7), . . . , (16) (grouping 1-5 together to make that group large enough) and compared them as training sources. We train a set of retrieval models on these training sources, where each model also has access to all of the LIGHT MTurk data (111k examples) in addition to the WILD 10k from their respective bins. The results are given in Figure 3 (left). We observe a monotonic improvement on the WILD validation set with increasing predicted quality. We see similar, but smaller gains on the LIGHT (Seen) validation set as well, e.g. 86.59% for quality bin 6, and 87.11% for quality bin 16.

While we can clearly select lower or higher quality data, we can also ask the question whether some of the data is so low quality we should simply remove it from the training data in order to get better performance. Experiments show that is not the case, and that even the lowest quality data does provide a useful signal, e.g. performance drops slightly from 87.06% to 86.69% on the WILD validation set if we remove bins lower than 6, but otherwise training on all other data, and to 85.38% if we remove bins lower than 9.

5.3.6 Analysis of Data Distribution

We can compare the dialogue data collected within our deployed system to crowdsourced data from LIGHT MTurk. We analyze over and underexpressed words in our dataset compared to the latter.

Calculating the top 70 most overexpressed words, all overexpressed at least 3.5x relative to crowdsourced data, we note several interesting observations about our data's distribution. There are more natural endings to conversations: e.g. "goodbye" (4×) and "bye" (4×) are overexpressed. There are overexpressed words associated with aggression: "stab" (8.5×), "dagger" (6.1×), "club" (5.5×), "kills" (4.9×), "blade" (4.2×). There are overexpressed words associated with overtly friendly actions as well: "smiles" (12.9×), "nods" (10.9×), "kiss" (6.1×), "hug" (3.7×), and "bows" (5.9×). There are more mentions of adventuring: "quest" (5.4×), and other similar words not in the top 70 are overexpressed as well, such as "adventure" (2.5×) and "mission" (2.1×). There is an increased use of slang: "ur" (93×), "u" (28×), "yo" (5×), "dude" (6×). We note that some emojis exist as well that do not appear in the crowdsourced data.

In contrast, looking at the 70 most underexpressed words, all underexpressed by a factor of at least 1.3×, we observed the following patterns. There are less mentions of village and farm life: "peasant", "fields" (both 2× underexpressed), "farm" and "crops" (both 1.9×), "harvest" (1.8×), "villagers" (1.7×), and "work" (1.4×). There are less mention of passages of time: "week" (2.1×), "year" (1.9×), "days" (1.8×).

Overall, we see a pattern that game players seek more exciting conversations, involving more emotional, action-packed interactions such as seeking quests, whereas crowdworkers are more even-keeled, and discuss dry topics such as last year's harvest or taxes with more frequency. This is not unexpected as players often seek immediacy and larger-than-life experiences (Grodal et al., 2000).

6 Conclusion

We have presented an approach for training dialogue agents during deployment with organic users utilizing a role-playing game between humans and bots. Detailed experiments show that one can collect high quality data that improves both automatic offline metrics and user engagement metrics. We find this exciting because this shows that it is possible to build continually improving models that learn from interacting with humans in the wild which represents a paradigm shift away from the limited static dataset setup that is prevalent in much of the work of the community. We make our code and data publicly available, and note our setup can be applied more generally, e.g., incorporating both dialogue and actions, or situated in other domains.

References

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. 2018. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248.

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2020. Queens are powerful too: Mitigating gender bias in dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8173–8188, Online. Association for Computational Linguistics.

Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019a. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4537–4546, Hong Kong, China. Association for Computational Linguistics.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019b. Wizard of Wikipedia: Knowledge-powered conversational agents. In *Proceedings of the International Conference on Learning Representations*.

Torben Grodal et al. 2000. Video games and the pleasures of control. *Media entertainment: The psychology of its appeal*, pages 197–213.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *Proceedings of the International Conference on Learning Representations*.

Matthew Horsfall and Andreas Oikonomou. 2011. A study of how different game play aspects can affect the popularity of role-playing video games. In *2011 16th International Conference on Computer Games (CGAMES)*, pages 63–69. IEEE.

Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems (TOIS)*, 38(3):1–32.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *Proceedings of the International Conference on Learning Representations*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016a. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823*.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016b. Learning through dialogue interactions by asking questions. *arXiv preprint arXiv:1612.04936*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Proc. Interspeech 2017*, pages 2506–2510.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2060–2069, New Orleans, Louisiana. Association for Computational Linguistics.

Sahisnu Mazumder, Bing Liu, Shuai Wang, and Nianzu Ma. 2019. Lifelong and interactive learning of factual knowledge in dialogues. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 21–31, Stockholm, Sweden. Association for Computational Linguistics.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, B Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Khanh Nguyen and Hal Daumé III. 2019. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 684–695, Hong Kong, China. Association for Computational Linguistics.

Aishwarya Padmakumar and Raymond J Mooney. 2020. Dialog as a vehicle for lifelong learning. *arXiv preprint arXiv:2006.14767*.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy. Association for Computational Linguistics.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media.

Mark Bishop Ring. 1994. *Continual learning in reinforcement environments*. Ph.D. thesis, University of Texas at Austin Austin, Texas 78712.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1702–1723. ACL.

Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Daniel L Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 673–683, Hong Kong, China. Association for Computational Linguistics.

Luis Von Ahn. 2006. Games with a purpose. *Computer*, 39(6):92–94.

Harm de Vries, Dzmitry Bahdanau, and Christopher Manning. 2020. Towards ecologically valid research on language user interfaces. *arXiv preprint arXiv:2007.14435*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213. ACL.

A Further Game Details

Other Gamification Steps The acting scores (between 1-5 stars per turn) are accumulated, and a player's total score is presented on a leaderboard compared to all other players, providing further motivation to reach the top of the leaderboard. We also award "badges" if, for a given dialogue, a certain number of points is collected (11 for 1 badge, 16 for 2): the badges represent the characters in the game, motivating the desire to role-play all the characters in the game, and collect all the badges.

License Agreement and Public Release Upon entry to the game, players are asked to agree to the use and release of the resulting game data as a publicly available dataset for research purposes. They are urged to stick to their assigned characters in the game, and hence should not use any personally identifying information, which the terms also tell them explicitly not to share. In the released data, no other information about the player is retained except for the messages they send in the game.

Game Safety We employ a safety classifier (Dinan et al., 2019a) on both human and model turns. For safety reasons, we limit our deployed dialogue models to be retrieval models, so that we could vet the entire set of candidates for offensive language before run-time (we evaluate generative models in an offline setting only). The set of settings and character personas were all also vetted for offensive language. Additionally, gender bias concerns have been previously studied within the available LIGHT MTurk training set (Dinan et al., 2020), and we make use of that publicly available data here as well. We note that, compared to other deployed dialogue systems, there is an extra level of indirection due to playing characters in a game that makes language relatively less offensive. For example, a thief in the game saying "I'm going to steal your money" to another game character is far less offensive compared to a digital assistant saying it directly to a human user.

B Game Screenshots

We show game screenshots in Figures 5 and 6.

C Using WILD Model responses

In our main results we use the human utterances collected from the role-playing game to form the WILD dataset targets, the hypothesis being that model utterances may or not be correct, and so are not as good a signal. A contrasting view could see training on the model utterance data as a kind of distillation of the previous model's knowledge. To test the performance of WILD human vs. model utterances, we conducted further experiments comparing them to each other. We observe a significant drop in Hits@1/20 performance using the model utterances for training on the WILD validation set (86.05% for human utterances, and 73.96% for model), and similarly on the LIGHT validation set (82.32% for human, and 77.56% for model).

A similar trend emerges when evaluating our generative models. As before, we see a drop in performance when training on WILD model utterances vs. training on WILD human utterances, manifesting as significant increases in perplexity on the WILD validation set (12.05 for human utterances, and 15.15 for model) as well as the LIGHT validation set (13.71 for human utterances, and 14.76 for model).

With the generative models, we additionally explored multi-task training with both the model and human utterances. In this regime, we introduced a special control token to indicate whether the desired response was produced by a model or a human; at inference time, we always provide the "human" control token. Including the control tokens in the training setup yields lower perplexities compared to vanilla multi-task training on both the WILD validation set (12.10 with control, 12.34 without) and the LIGHT validation set (13.60 with, 13.65 without). We note that, overall, multi-task training yields marginally worse performance on the WILD validation set and marginally better performance on the LIGHT validation set (see numbers in previous paragraph).

D Comparing Small and Large Model Variants

We evaluated the differences between the small and large retrieval models (Sec. 4.1) during deployment to analyze the differences between them. Results for both large and small models are shown in Table 7. The expectation tends to be that larger models would perform better, especially when the automatic metrics reflected this case. To eliminate the possibility that difficulty in tuning the decoding control differed between the small and large models, we launched a task to crowdworkers to evaluate the models shown in rows 3 and 8 of Table 4. These

Model	Contradiction	Mistaken Identity or Location	Off-topic	Repetitive or Boring	Rating	Count
90M PE	3.9%	9.1%	8.4%	2.1%	3.67	88
622M PE	3.0%	5.3%	6.0%	5.3%	3.69	91

Table 6: Percentage of utterances flagged with an issue alongside overall satisfaction, by model.

correspond to the small and large models trained with LIGHT + WILD data without decoding control, further noted as LW90M and LW622M.

Workers were asked to have a 12-turn conversation with a model (6 turns each), and evaluate each model utterance with respect to some of the most common mistakes we observed models making. They were then asked to provide an overall score for the model, answering "How much fun did you have Role-playing with your chat partner? Would you have a similar conversation with this chat partner in the future?". Results are given in Table 6, listing the percentage of utterances falling into each mistake type.

We observed that LW622M avoided a number of common mistakes when compared to LW90M, contradicting itself less frequently, assuming the role of the wrong character or using a wrong location less frequently, and going off-topic less frequently. These types of mistakes were common complaints of players interacting with our game, and all seemed to be related to having somewhat bad experiences. LW622M also had a higher percentage of conversations without any listed issues, with 34.1% compared to 27.3% for LW90M.

LW622M however was rated as being repetitive or boring more than twice as frequently as the smaller model. This difference could be supported in that LW622M used utterances that were on average 2 words shorter than those of LW90M over these evaluations. This issue could explain the phenomenon where players are more likely to leave after interacting with a larger model than a smaller one. The only thing really encouraging unpaid players to continue interacting with our models is that they are fun and engaging, and while it is possible to overlook the model making a mistake if it is still somewhat fun, it is likely less possible to remain engaged when the model is actually boring and repetitive.

While LW622M is a better model across several aspects, it is clear from our live deployment evaluations that something is lost in scaling up from LW90M. Comparing these models with real players lets us see this issue, and moving forward should lead us to search for a model that does not become more boring when learning to not make mistakes.

Model	Train Data	Negative Context	Decoding Control	Continuation Rate
90M PE	LIGHT	yes	no	$72.2 \pm 1.9\%$
90M PE	LIGHT	yes	yes	$74.1 \pm 2.0\%$
90M PE	LIGHT + WILD	yes	no	$73.6 \pm 1.8\%$
90M PE	LIGHT + WILD	yes	yes	$75.2 \pm 2.0\%$
622M PE	LIGHT	no	no	$68.2 \pm 1.4\%$
622M PE	LIGHT	yes	no	$69.9 \pm 1.9\%$
622M PE	LIGHT	yes	yes	$69.9 \pm 2.0\%$
622M PE	LIGHT + WILD	yes	no	$70.6 \pm 2.1\%$
622M PE	LIGHT + WILD	yes	yes	$71.8 \pm 1.9\%$

Table 7: Deployment-based Evaluation, comparing several metrics on data collected during round 2.

E Observations on Gamification

Just as the design of a crowdsourcing task will affect the cost and quality of data, this is likely even more the case in the design of a game. If the design is poor, players will not play it at all; whereas in contrast to paying crowdworkers, if players really like a game, they are willing to pay to play it. Accordingly, the plots we presented in Figure 2 represent the results of our particular game design; there may well be a design with vastly more cost efficient learning rates. While a full of study of the elements of game design is outside of the scope of this paper, we note that for adjustments we did make to the game after initial deployment we observed large changes in user behavior. For example, after the addition of the three user controls for how to continue the game loop after an episode is finished (as described in Sec. 3), compared to only a single choice, we saw an increase in the continue rate by $3.3 \pm 1.6\%$ when using the same model.

Model quality also affects cost and quality of the data collected. Noting the effects of changing gamification options (alongside other hard-to-track circumstances) we only report continue rate comparisons between models relative to runs in the same batch. Still, players' enjoyment of these models (as estimated by continue rate in Table 4) directly changes how much they engage with the game. As such it is more expensive to test models that are worse for the game experience (which we would consider fair from a player perspective). Hence, as models improve, costs actually go down, enabling to collect data at higher rate.

LIGHT RPG
Published by · May 13, 2020 ·

Play the AI driven role-playing chat game, LIGHT! You will be teleported into the mystical realm of LIGHT and fill the shoes of a character who lives there — playing that role you interact with other AI denizens, with your messages evaluated by the Dungeon Master AI. Portray your character well and they will award points. ⭐🌟🏅Collect all the character badges! 🛡️Send message to play for FREE.

MESSENGER
Send Message To Play For FREE.

Figure 4: Example advertisement used to draw interest to the deployed LIGHT chat game

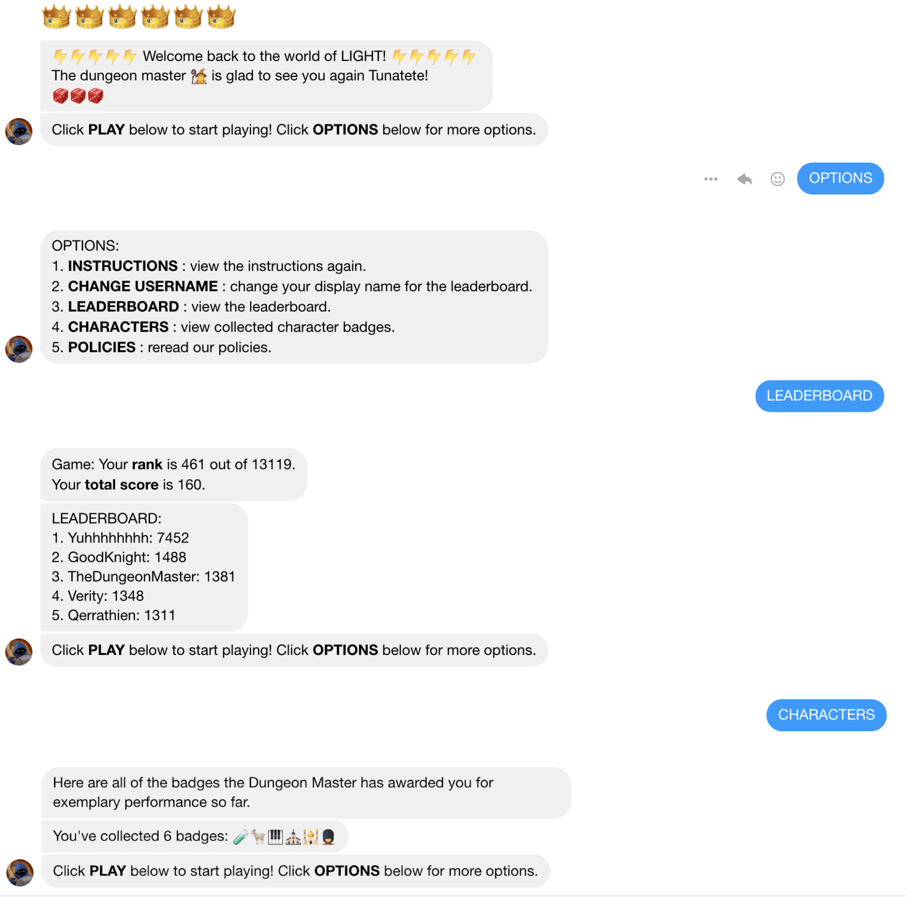

👑👑👑👑👑👑

⚡⚡⚡⚡⚡ Welcome back to the world of LIGHT! ⚡⚡⚡⚡⚡
The dungeon master 🧙 is glad to see you again Tunatete!
🎲🎲🎲

Click **PLAY** below to start playing! Click **OPTIONS** below for more options.

OPTIONS:
1. **INSTRUCTIONS** : view the instructions again.
2. **CHANGE USERNAME** : change your display name for the leaderboard.
3. **LEADERBOARD** : view the leaderboard.
4. **CHARACTERS** : view collected character badges.
5. **POLICIES** : reread our policies.

Game: Your **rank** is 461 out of 13119.
Your **total score** is 160.

LEADERBOARD:
1. Yuhhhhhhhh: 7452
2. GoodKnight: 1488
3. TheDungeonMaster: 1381
4. Verity: 1348
5. Qerrathien: 1311

Click **PLAY** below to start playing! Click **OPTIONS** below for more options.

Here are all of the badges the Dungeon Master has awarded you for exemplary performance so far.

You've collected 6 badges: 🧪🐐🎹⛪🕌💂

Click **PLAY** below to start playing! Click **OPTIONS** below for more options.

Figure 5: Screenshots of the instructions of the LIGHT role-playing game.

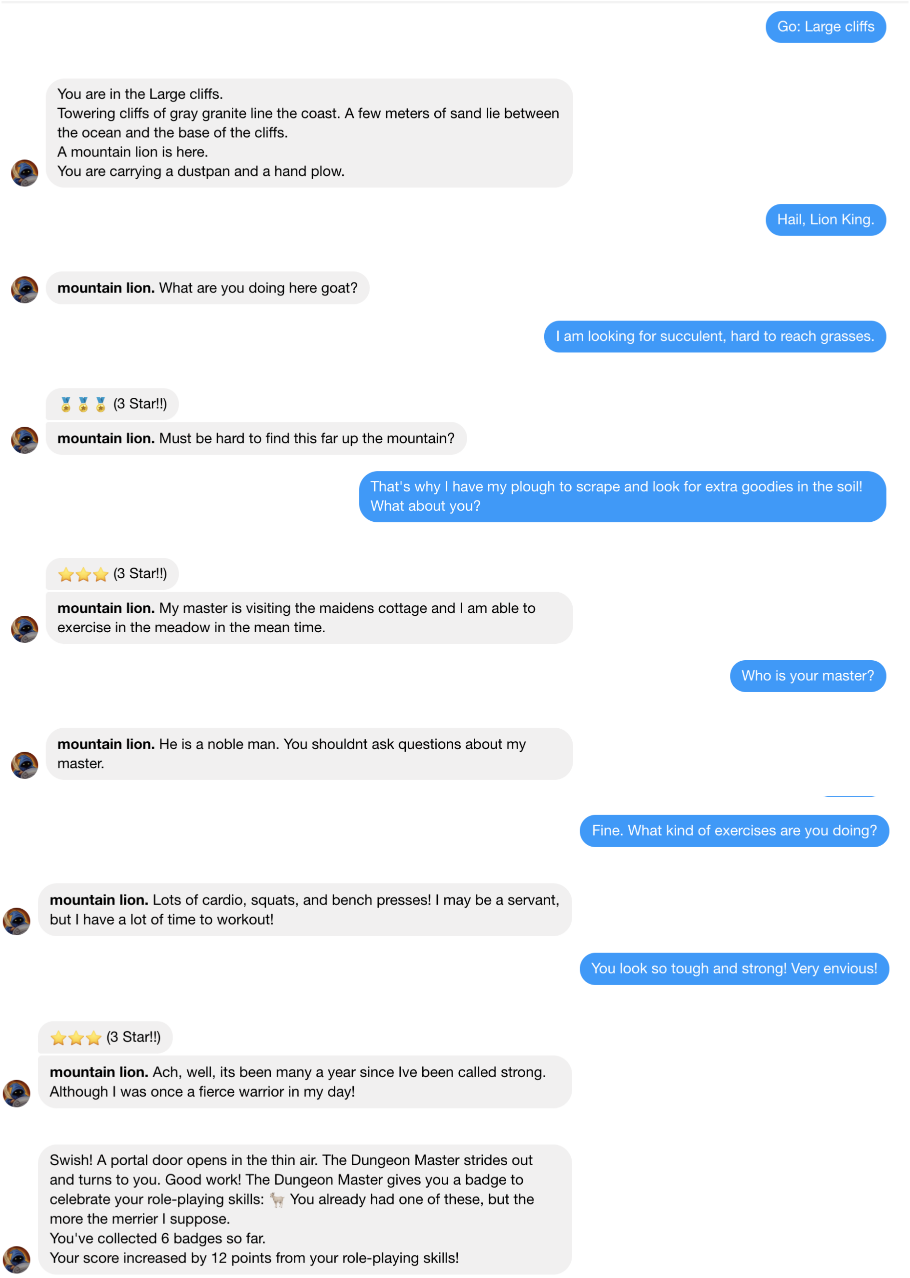

Go: Large cliffs

You are in the Large cliffs.
Towering cliffs of gray granite line the coast. A few meters of sand lie between the ocean and the base of the cliffs.
A mountain lion is here.
You are carrying a dustpan and a hand plow.

Hail, Lion King.

mountain lion. What are you doing here goat?

I am looking for succulent, hard to reach grasses.

🏅🏅🏅 (3 Star!!)

mountain lion. Must be hard to find this far up the mountain?

That's why I have my plough to scrape and look for extra goodies in the soil! What about you?

⭐⭐⭐ (3 Star!!)

mountain lion. My master is visiting the maidens cottage and I am able to exercise in the meadow in the mean time.

Who is your master?

mountain lion. He is a noble man. You shouldnt ask questions about my master.

Fine. What kind of exercises are you doing?

mountain lion. Lots of cardio, squats, and bench presses! I may be a servant, but I have a lot of time to workout!

You look so tough and strong! Very envious!

⭐⭐⭐ (3 Star!!)

mountain lion. Ach, well, its been many a year since Ive been called strong. Although I was once a fierce warrior in my day!

Swish! A portal door opens in the thin air. The Dungeon Master strides out and turns to you. Good work! The Dungeon Master gives you a badge to celebrate your role-playing skills: 🦙 You already had one of these, but the more the merrier I suppose.
You've collected 6 badges so far.
Your score increased by 12 points from your role-playing skills!

Figure 6: Gameplay Screenshots of the LIGHT role-playing game.

Deep Learning against COVID-19: Respiratory Insufficiency Detection in Brazilian Portuguese Speech

Edresson Casanova[1], Lucas Gris[2], Augusto Camargo[3], Daniel da Silva[2], Murilo Gazzola[1], Ester Sabino[4], Anna S. Levin[4], Arnaldo Candido Jr[2], Sandra Aluisio[1], Marcelo Finger[3†]

[1]Instituto de Ciências Matemáticas e de Computação, University of São Paulo, São Carlos/SP, Brazil
[2]Federal University of Technology – Paraná, Medianeira/PR, Brazil
[3]Instituto de Matemática e Estatítica, University of São Paulo, São Paulo/SP, Brazil
[4]Faculdade de Medicina, University of São Paulo, São Paulo/SP, Brazil

Abstract

Respiratory insufficiency is a symptom that requires hospitalization. This work investigates whether it is possible to detect this condition by analyzing patient's speech samples; the analysis was performed on data collected during the first wave of the COVID-19 pandemic in 2020, and thus limited to respiratory insufficiency in COVID-19 patients. For that, a dataset was created consisting of speech emissions of both COVID-19 patients affected by respiratory insufficiency and a control group. This dataset was used to build a Convolution Neural Network to detect respiratory insufficiency using speech emission MFCC representations. Methodologically, dealing with background noise was a challenge, so we also collected background noise from COVID-19 wards where patients were located. Due to the difficulty in filtering noise without eliminating crucial information, noise samples were injected in the control group data to prevent bias. Moreover, we investigated (i) two approaches to address the duration variance of audios, and (ii) the ideal number of noise samples to inject in both patients and the control group to prevent bias and overfitting. The techniques developed reached 91.66% accuracy. Thus we validated the project's Leading Hypothesis, namely that it is possible to detect respiratory insufficiency in speech utterances, under real-life environmental conditions; we believe our results justify further enquiries into the use of automated speech analysis to support health professionals in triage procedures.

1 Introduction

This work started as part of the academic initiative to help in the effort to deal with the COVID-19 pandemic in a severely affected region in Brazil. COVID-19 is an infectious disease caused by the virus SARS-CoV-2. This illness is mainly associated to severe acute respiratory syndrome, although it is harmful to other organs, like heart, kidney and brain. About 82% of cases are mild or moderate, while the rest are severe or grave, demanding hospitalization or intensive care. The most vulnerable groups are people over the 60's, and people with specific medical conditions such as diabetes, obesity, hypertension and heart disease. According to WHO[1], in August 3 2020, more than 19.2 million people in the world had contracted COVID-19, with a Case Fatality Ratio of CFR=2.8%. Respiratory Insufficiency (RI) is a symptom that requires hospitalization, which is aggravated due to a frequent COVID-19 condition called *silent hypoxia*, low blood oxygen concentration without breath shortness (Tobin et al., 2020).

This work ***leading hypothesis*** states that *it is possible to detect respiratory insufficiency by analyzing spoken utterances in real-life conditions*, typically a moderately large sentence, thus subscribing to the view of *speech as a biomarker*. This work aims at validating this leading hypothesis using deep learning techniques.

If the hypothesis holds, it will motivate further enquiries on the use of automated speech analysis to support health professionals; with infectious diseases such as COVID-19, a serious concern involves deciding whether an RI suspect should stay in social isolation or be directed to a medical facility. Project SPIRA[2][3] was initiated to investigate the feasibility of supporting medical triage of patients with COVID-19 symptoms by remotely detecting respiratory insufficiency through automated speech utterance analysis, where no other resources are

---

[†] Corresponding author: `mfinger@ime.usp.br`.

[1] `https://covid19.who.int`, visited May 24 2021.

[2] https://spira.ime.usp.br/

[3] In Portuguese, ***S****istema de detecção* ***P****recoce de* ***I****nsuficiência* ***R****espiratória por análise de* ***A****udio* – system for early detection of respiratory insufficiency via audio analysis.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 625–633
August 1–6, 2021.

available other than a phone line or a cellphone app. A positive result may motivate further research into speech-based remote detection of respiratory problems originating from other causes, such as heart condition, airway obstruction, severe asthma, H1N1, etc.

This research started as a response to the peak of the first COVID-19 wave in 2020, when health infrastructure was overloaded, so no doctors nor nurses were available for data collection, and there was no triage point available for research. Thus, COVID-19 patient utterances were collected mostly by medical students at COVID-19 wards from patients with blood oxygenation below 92%, as an indication of respiratory insufficiency, and control data was collected by voice donations over the internet, assumed healthy, with no access to blood oxygenation. Recordings were made in out-of-studio conditions, using portable recording equipment employed in noisy wards. On the other hand, conditions for healthy voice donations over the Internet and using diverse sound equipment had a large variation. This *audio data in-the-wild* approach was assumed from the start as part of the challenge of validating the leading hypothesis. Part of the methodological novelty of this work lies on how to deal with these conditions. This task required a multidisciplinary group involving medical doctors, linguists, speech therapists and computer scientists, all of which were aware of those conditions and challenges facing us.

This work proposes a machine learning method to detect respiratory insufficiency by analyzing voice audio recordings of sentences long enough to feature respiratory pauses in speech. The test is very cheap, requiring only a voice sample from each patient and maybe employed where no other medical equipment is available.In order to tackle the audio analysis, we propose the use of deep artificial neural networks over Mel Frequency Cepstral Coefficients (MFCCs) (Logan et al., 2000) extracted from patient's audios.

The code and datasets are publicly available at `https://github.com/SPIRA-COVID19/SPIRA-ACL2021`, under a CC BY-SA 4.0 license.

This paper is organized as follows: Section 2 discusses related work. In Section 3, the dataset acquired, the preprocessing steps, the noise insertion procedure, the proposed model and experiments are described, respectively. Afterwards, the models obtained are evaluated and discussed in Section 4. Finally, Section 5 presents the conclusions and final thoughts.

2 Related work

COVID-19 is a recent disease. However, even before the eruption of the pandemic, we could already find in the literature a few explorations of speech as a biomarker (Botelho et al., 2019; Trancoso et al., 2019; Nevler et al., 2019), with some recent recommendations (Robin et al., 2020).

Several initiatives can be found on the Web that record human voice in order to assess the presence and the gravity of COVID-19, e.g. the COVID-19 Sounds data collection initiative (Brown et al., 2020) and startup initiative aiming to develop a pre-diagnostic tool[4]. Those works aim to diagnose COVID-19 from voice or breathing or coughing sounds, and there are some initial positive results on COVID-19 detection in asymptomatic individuals (Laguarta et al., 2020). Unlike our approach, no work aimed specifically at respiratory insufficiency or at patient triage, but they propose to employ some form of artificial intelligence processing.

In similarity to our goals, there have been recent proposals of applications for the triage of patients using natural language processing of texts extracted from radiology reports (Hassanpour et al., 2017) and patient questionnaires (Spasić et al., 2019). So language, both as text and now as speech, is being used for patient screening.

Moreover, Neural Networks and Convolutional Neural Networks (CNNs) have been used in noisy environments mostly, but not exclusively, for fault diagnosis (Zhang et al., 2018; Munir et al., 2019), noise reduction in voice processing (Maas et al., 2012) and medical ECG diagnosis (Acharya et al., 2017). On the other hand, *noise injection* was a technique used in the past to avoid overfitting in training Neural Networks (Matsuoka, 1992; Grandvalet et al., 1997; Zur et al., 2009), as opposed to avoiding classification biases, as in our approach.

3 Methodology

In order to build a neural network model for the proposed task, it is necessary to gather a dataset containing voices of healthy individuals and COVID-19 patients (Section 3.1). The resulting dataset required several preprocessing treatments and noise treatment, as discussed in Sections 3.2 and 3.3. The next step was to propose several neural models to

---

[4] https://www.voicemed.io/

investigate the best one for the task (Section 3.4) evaluated according to experiments carried over the dataset (Section 3.5).

3.1 Dataset

The dataset creation was composed of two parts and an "appendix". The first part consisted of audios gathered via Web by a system specifically designed for this task[5], from May to July of 2020. Healthy volunteers were asked to donate audio samples via a web interface. This allowed us to build our control group. In order to do that, the system URL was disclosed through local news and social networking. The resulting dataset part is composed, after elimination of blank samples, of more than 6 thousands voice donors. No blood oxygen saturation information was available for the control group.

In the second part, we collected audios from patients infected by SARS-CoV-2 from June to July of 2020. This collection was performed in COVID-19 wards in two university hospitals, in São Paulo city, Brazil, restricted to patients with blood oxygenation level (SpO2) inferior to 92%, as an indication of respiratory insufficiency. This allowed us to collect 536 samples from patients in different age groups. Several problems led to discarding patient voice samples, chiefly among which were collectors whispering during collection; a large set of collection instructions was assembled during the period in which voice collection took place. It is important to note that São Paulo is a local and international hub, with a large migrant and immigrant population. Hospitals received COVID-19 patients from the city as well as from adjoining regions. Collection was absolutely anonymous, so no one knows who were the patients and controls, and no ethnographic information is available. On the other hand, this allowed us to release the data.

As a COVID-19 ward is a noisy environment, an "appendix" was built for this dataset, consisting of samples of pure background noise at the ward (no voice), typically collected at the start of a collection session. This is an important piece of information, as the ward noise is very different from the background noise found in the control group, and consists of a *data bias* that has to be controlled during experiments.

The gathered audios contain three utterances:

- Utterance 1, a moderately long sentence containing 31 syllables, designed by linguists to allow for spontaneous breathing breaks, while being relatively simple to be spoken, even by low literacy voice donors: *"O amor ao próximo ajuda a enfrentar o coronavírus com a força que a gente precisa."* ("Love of neighbor helps in strengthening the fight against Coronavirus.");
- Utterance 2, a well known nursery rhyme for donors having reading difficulties, due to lack or reading glasses in hospital, or other types of reading impediments: *"Batatinha quando nasce, espalha a rama pelo chão, nenezinho quando dorme põe a mão ao coração"* ("When small potatoes germinate, branches sprout on the ground; when baby sleeps, hands rest over the heart");
- Utterance 3, a widely known song, on the lines of "Happy birthday to you": *"Parabéns a você, nesta data querida, muitas felicidades, muitos anos de vida"* ("Happy birthday to you, on this dear date, lots of happiness, many years of life").

Collecting longer utterances was totally impractical in a COVID-19 ward. The collection had to be adapted to what was possible in that context.

We identified several issues with the original dataset that need to be addressed. First, there is class imbalance, as we have fewer positive instances (COVID-19 patients) than negative ones (healthy individuals from the control group). Second, it is sex imbalanced, as a greater number of healthy women participated in the process than healthy men. Additionally, there are more men in COVID-19 wards than women. Third, there is an age imbalance, as there are more elderly in hospital care than young people in our observations. Fourth, we also detected utterance imbalance, as utterance 1 was more common among patients; healthy people typically recorded all proposed utterances. Fifth, the control group presented popping and crackling noise, possible due to the characteristics from the recording devices. Furthermore, as mentioned above, wards tend to be noisy environments.

We addressed most of the dataset issues by sample balancing, taking advantage of the greater number of control group samples. Only audios from utterance 1 were selected and the number of samples used in experiments was balanced by class and sex, but not by age, to avoid drastically reduc-

---

[5] https://spira.ime.usp.br/coleta/

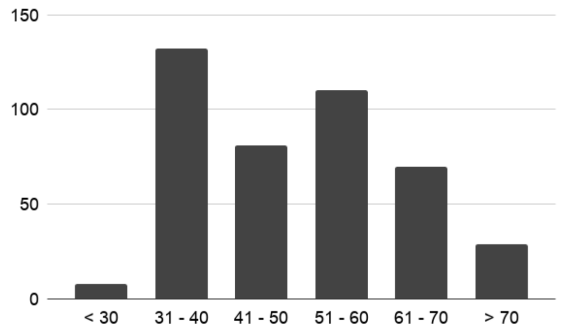

Figure 1: Distribution of Ages in the Dataset

ing the available data. Overall age distribution is presented in Figure 1.

We also had to discard audio containing the collector's (whispering) voice. The most serious issue for bias removal, though, is the presence of ward background noise in patient audios; we observed that it is easier to insert ward noise in the control group than to remove it from the patients' signal. This process will be addressed in Section 3.2.

The dataset was divided in training, validation and test, as is usual in statistical learning. We selected audios with the best signal-noise ratio to use in the test set, and the second best audios were used for validation. The aim of this partitioning is to detect training overfitting.

Information of the resulting filtered dataset is presented in Table 1.

3.2 Pre-processing

In general, the majority of the audios in the dataset was sampled at 48kHz. We pre-processed these files using Torch Audio 0.5.0 in the following way. First, for dimensionality reduction reasons, we re-sampled these audios at 16kHz. Second, we extracted the MFCCs using a 400ms window employing Fast Fourier Transform (FFT) (Brigham and Morrow, 1967), with hop length 160 and 1,200 FFT components, of which we retained only 40 coefficients. Before the MFCC feature extraction process though, we need to address the difference of duration present in our data.

The duration of our samples in the dataset varies, in which audios from the positive class are slightly longer than audios from the negative class, as presented in Table 1. We have developed two approaches to deal with this phenomenon. First, we applied padding in the instances during training. This is equivalent to complete the audios with silence so that all audios have the same duration. Second, we have extracted fixed length fragments from the audios. This approach aims to prevent the model from performing the classification giving too much importance to the audio length. In order to augment the training data, windowing with 1 second steps was applied to extract audio fragments.

3.3 Noise Insertion During Training

Ward noise is a serious bias source, as confirmed by our preliminary experiments (Section 4). In this scenario, a neural network can be biased during training by focusing only on background noise. One possible alternative would be noise filtering, but besides the possibility of inserting extra biases due to differential noise suppression in patient and control audio samples, there is also the possibility of suppressing important low-energy information that allows for the distinction between healthy and respiratory affected speech samples.

To address this issue, we decided to record pure background noise samples from COVID-19 wards and to inject into patients and control group audios. In total, 16 samples with approximately 1 minute each were recorded.

The inserted noise can also be a cause of bias, as the model can extract specific features from the noise recording. To avoid this kind of bias, we decided to inject noise in all samples. We had the option of inserting in training, validation and testing samples, which will be described in Section 4. We can also control the amount of noise samples inserted in each audio.

In our experiments, we investigate the ideal number of noise samples to inject in both patients and the control group. This had a big impact in overfitting prevention, as a form of unbiased learning, as described in Section 4. During training, at each epoch, audio samples can be injected with one or more distinct noise samples. Each time a given audio is used for training, noise samples are drawn from the noise base. Besides that, the start point of each noise sample is also randomized. Finally, we also draw a factor to change the intensity of the sample. This factor is constrained by a maximum amplitude value, which was determined from the analysis of patient audio noises. The aim is to insert noises as similar as possible to already pre-existing noise. We also executed the same experiment three times with different random seeds to obtain better measures of the noise insertion impact.

Table 1: Filtered dataset information

Sets	Control			Patients			Total Audios	Total Duration (s)
	Male	Female	Mean Duration (s)	Male	Female	Mean Duration (s)		
Training	59	84	8.15	83	66	13.18	292	3110
Validation	8	8	7.75	8	8	10.78	32	296
Test	22	26	7.77	28	32	9.43	108	983

The test and validation sets were created in such a way to allow overfitting detection as they are composed mostly of audios with very limited amount of noise. As a result, we cannot apply k-fold Cross Validation and similar methods. We compensate this by running the same experiment three times with different random seeds. This fact, together with the dynamic noise insertion during training, allows us to obtain averaged accuracy for each experiment.

3.4 Proposed Model

Several models were tested in preliminary experiments and we describe the one that led to the best results.

This process involved three main aspects: (a) the topology and model parameters; (b) the main hyper-parameters; (c) regularization. The last is especially important, since our dataset contains several issues that can lead to overfitting.

Regarding topology and model parameters, preliminary experimental results showed that CNNs applied to MFCCs are useful to analyze this kind of problem. Other preliminary experiments investigated spectrograms and topologies like fully-connected and recurrent networks, which showed lower performance than the chosen topology. Figure 2 presents the chosen model's main features including layers, filters, kernels, number of neurons and activation functions. The following conventions are adopted in the figure: kernel size is represented by K; convolutional dilation size (Yu and Koltun, 2015) is represented by D; and fully connected layers are represented by FC. The input size is omitted because these parameters changed according to the experiment and will be detailed in Section 3.5. We investigated the use of Mish activation function (Misra, 2019), due to its regularization effects during training, which helps prevent overfitting.

Regarding the main hyper-parameters, we have used the Binary Cross-Entropy as loss, and Adam optimizer (Kingma and Ba, 2014). The initial learning rate was set to 10^{-3}, and the Noam's decay scheme (Vaswani et al., 2017) was applied on each 1,000 steps. For each experiment presented in Section 3.5, we trained the model for 1,000 epochs using a batch size of 30.

Regarding regularization, overfitting mitigation is a major concern given our dataset noise characteristics. Therefore, several approaches for regularization were applied. Besides Mish as an activation function, we used three other strategies. First, a global weight decay of 0.01 was applied. Second, a dropout of 0.70 was used in all layers, except in the output layer. Last, we applied group normalization (Wu and He, 2018) after each convolutional layer. The group normalization was applied on pairs of convolution filters. Therefore, the number of groups is half the number of filters.

3.5 Experiments

For the experiments we explored three main aspects with respect to noise insertion and duration variance: (a) overfitting impact; (b) padding vs windowing approach (using four second windows or adding padding); and (c) the ideal number of noise samples. Table 2 presents the proposed experiments and their results.

First we investigated if the model can overfit when trained over original audios (experiments 1.x). In this series of experiments, we trained the model using both approaches of duration variance.

Second we analyzed two approaches to address the duration variance: audio padded to the maximum length of the dataset; windowing using the approach described in Section 3.2 (experiments 2.x). Specifically, we presented padding application only in experiments 1.1 and 2.1 because experiments showed that the windowed approach led to more robust results. When padding is used, the accuracy is calculated as usual. However, in windowed experiments, several audio fragments are extracted and their predictions averaged for the classification decision. Regarding window size, we have chosen four seconds, considering our smallest audio in the

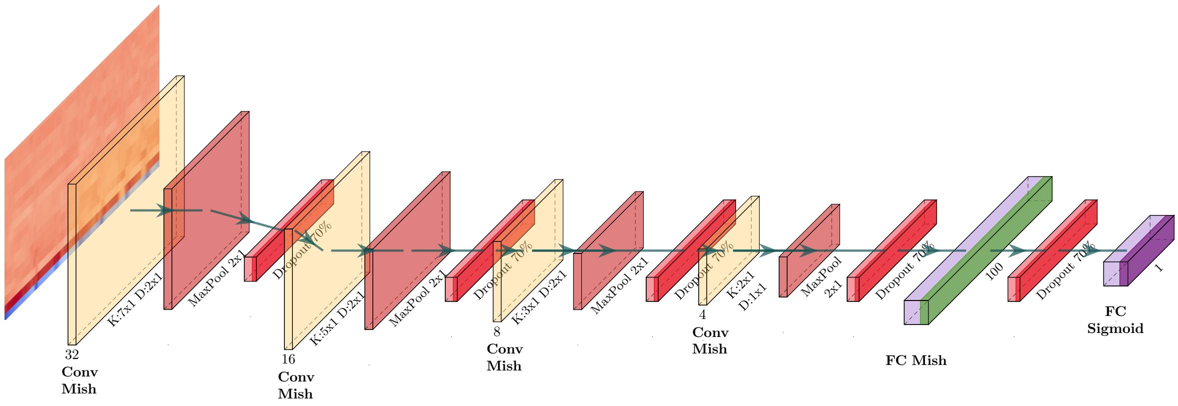

Figure 2: CNN topology proposed with four convolutional layers and two fully connected layers

Table 2: Proposed experiments and results

Description	Exp.	Duration Approach	Noise Samples		Accuracy (without noise in test samples)	Accuracy (with noise in test samples)	Training time (h)
			Patient	Control			
Overfitting Analysis	1.1	Padding	0	0	98.15 ± 0.93	50.93 ± 0.53	4.25
	1.2	Windowing	0	0	98.15 ± 0.53	50.93 ± 0.93	9.07
Duration Variance Analysis	2.1	Padding	0	1	61.11 ± 8.40	74.07 ± 1.93	4.77
	2.2	Windowing	0	1	66.67 ± 3.74	86.11 ± 2.98	9.58
Noise Insertion Analysis	3.1	Windowing	1	1	80.56 ± 2.45	68.52 ± 1.41	6.57
	3.2	Windowing	1	2	84.26 ± 6.17	83.33 ± 3.34	12.27
	3.3	Windowing	2	2	88.89 ± 0.53	85.19 ± 0.93	13.00
	3.4	Windowing	2	3	74.07 ± 5.10	85.19 ± 1.85	13.67
	3.5	Windowing	3	3	91.67 ± 2.98	87.04 ± 0.93	14.70
	3.6	Windowing	3	4	62.96 ± 8.35	74.07 ± 2.45	11.85
	3.7	Windowing	4	4	88.89 ± 1.41	83.33 ± 1.07	10.40
	3.8	Windowing	4	5	56.48 ± 5.10	72.22 ± 9.99	9.83
	3.9	Windowing	5	5	70.37 ± 15.8	69.44 ± 9.27	10.55
	3.10	Windowing	5	6	51.85 ± 3.51	61.11 ± 2.98	11.18
	3.11	Windowing	6	6	74.07 ± 10.7	74.07 ± 8.83	11.98
	3.12	Windowing	6	7	50.00 ± 0.53	54.63 ± 3.51	12.63

dataset contains 4.6 seconds and new data samples can be even smaller.

Third we examine the ideal number of noise samples to be inserted to prevent overfitting (experiments 3.x), using the best duration approach according to experiments 2.x. For each experiment, we tested the model using both noise insertion and no noise insertion to analyze performance.

Our model was implemented using Pytorch 1.5.1. We ran the experiments on a NVIDIA Titan V GPU with 12GB RAM in a server with Intel(R) Core(TM) i7-8700 CPU and 16GB of RAM.

4 Results and Discussion

To better understand bias and overfitting we used a test set containing only audios with a minimal amount of noise. The accuracy of each experiment is presented in Table 2, both with and without artificial insertion of ward noise in test samples.

Experiments 1.x showed the model is biased without noise insertion in the training set. We note a high accuracy in experiments 1.1 and 1.2 without noise in training and testing; in contrast, when noise is inserted in all test samples, it classifies all samples as coming from patients. We interpret this as a strong indication that the model is biased by the presence of noise in the patient samples.

Experiments 2.x showed that windowing (2.2) is preferable over padding (2.1), as described in Section 3; the model performs better when the windowed approach is used, that is, 66% using windowing against 61% using padding. We consider this as evidence of susceptibility to bias by padding. In fact, padding inserts a considerable amount of silence, specially in patient samples, and the windowed approach works as a data augmentation technique, as more instances are generated in this process.

Experiment 3.x were used to determine the optimal amount of noise insertion. Note that sometimes better results were obtained without noise in test samples and sometimes the other way around. In general, the bias is greatly reduced by inserting at least one noise sample on the negative instances. As expected, the insertion of too much noise decreases the model performance. The best overall accuracy was obtained in experiment 3.5, which reached 91% accuracy in the task. For experiment 3.5, we obtained F1 = 0.90, without noise insertion; with noise insertion, F1 = 0.87.

Figure 3 presents the loss variation of the best model (experiment 3.5) during training. Early stopping is used to get the best iterations after approximately 20k steps.

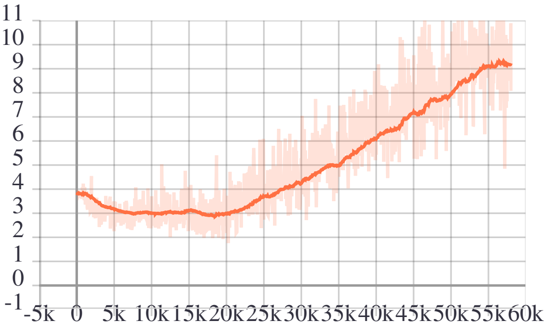

Figure 3: Validation Loss for Experiment 3.5 during Training

Figure 4 shows the model performance over the number of noise samples inserted. With respect to the number of noise samples, our experiments suggest that a number of 2 to 4 noise insertions in each audio provides best accuracy. In each case, two possibilities have been tested, namely the insertion of an equal number of noise samples in each training audio, and the insertion of one extra noise sample to control audio, assuming that patient audios already have the original ward noise. It was initially expected that the insertion of an extra noise sample in control audios would produce better results; surprisingly, the opposite effect was observed. The possible explanation for this observation is that there are times when wards are calmer and silent and the insertion of noise in control audios leads to bias. This is especially true for testing samples, due to the criteria used to build the testing set.

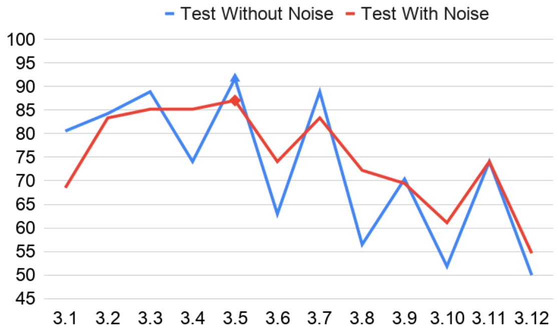

Figure 4: Sample Noise Analysis for the Best Experiments

5 Conclusions and future work

In the effort to tackle the COVID-19, we have developed a method to classify real-life speech audio signals on whether or not that signal originated from a person suffering from respiratory insufficiency. In this effort, we obtained 91.67% accuracy, thus validating the hypothesis that such a detection is feasible, and that human speech can be treated as a biomarker in this case.

One important consequence of this work was the construction of a dataset containing voice samples of COVID-19 patients with respiratory insufficiency and also a set of samples of environmental noise, which were central in treating real-life sound samples. Noise insertion was chosen as the more adequate option when contemplating the biases that would be incurred by filtering procedures. In particular, it made sense to add ward noise to the existing ward samples as a way to balance the biases that were incurred by the necessary addition of ward noise to control data. In this way, all data (patient and control) suffered from similar manipulation, avoiding editing bias, and experiments showed that such a procedure produced best results. This aimed at preventing the models from memorizing ward noise and editing distortion information instead of COVID-19 features.

There was a considerable difficulty to collect voice data from infected patients during the pandemic. The size of the patient dataset reflects the limitations on collections in COVID-19 wards. Moreover, the use of audio from different environments was absolutely unavoidable, as we only had access to patients in COVID-19 wards, where no control subjects were available. Therefore, control data had to be collected in a different environment. As a result, the amount of data was scarce, and data augmentation techniques were designed for such a setting; our results indicate that it was not an excessive amount of data augmentation, as consistent results were obtained over a large variety of experiments. We hope that with the weakening of the emergency situation, it could become easier to collect data from patients with respiratory insufficiency.

Future work includes augmenting the dataset with audios collected at the triage point, whether in hospital admission rooms, or through a remote admission system. In this way, speech audio signals from both sufferers and non-sufferers of respiratory insufficiency would be obtained under similar conditions. This would allow us to extend this study to other respiratory illnesses besides COVID-19. Also, other neural architectures can be explored, as well as smarter feature engineering.

Acknowledgments

This work was supported by Fapesp project 2020/06443-5 (SPIRA). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Marcelo Finger was partly supported by Fapesp projects 2019/07665-4 and 2014/12236-1 and CNPq grant PQ 303609/2018-4.

We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used in part of the experiments presented in this research.

References

U. Rajendra Acharya, Hamido Fujita, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, and Muhammad Adam. 2017. Application of deep convolutional neural network for automated detection of myocardial infarction using ecg signals. *Information Sciences*, 415-416:190 – 198.

M Catarina Botelho, Isabel Trancoso, Alberto Abad, and Teresa Paiva. 2019. Speech as a biomarker for obstructive sleep apnea detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5851–5855. IEEE.

E Oran Brigham and RE Morrow. 1967. The fast fourier transform. *IEEE spectrum*, 4(12):63–70.

Chloë Brown, Jagmohan Chauhan, Andreas Grammenos, Jing Han, Apinan Hasthanasombat, Dimitris Spathis, Tong Xia, Pietro Cicuta, and Cecilia Mascolo. 2020. Exploring automatic diagnosis of covid-19 from crowdsourced respiratory sound data. *arXiv preprint arXiv:2006.05919*.

Yves Grandvalet, Stéphane Canu, and Stéphane Boucheron. 1997. Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108.

Saeed Hassanpour, Curtis Langlotz, Timothy Amrhein, Nicholas Befera, and Matthew Lungren. 2017. Performance of a machine learning classifier of knee mri reports in two large academic radiology practices: A tool to estimate diagnostic yield. *AJR. American Journal of Roentgenology*, 208:1–4.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

J. Laguarta, F. Hueto, and B. Subirana. 2020. Covid-19 artificial intelligence diagnosis using only cough recordings. *IEEE Open Journal of Engineering in Medicine and Biology*, 1:275–281.

Beth Logan et al. 2000. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11.

Andrew Maas, Quoc V. Le, Tyler M. O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y. Ng. 2012. Recurrent neural networks for noise reduction in robust asr. In *INTERSPEECH*.

K. Matsuoka. 1992. Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440.

Diganta Misra. 2019. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*.

Nauman Munir, Hak-Joon Kim, Jinhyun Park, Sung-Jin Song, and Sung-Sik Kang. 2019. Convolutional neural network for ultrasonic weldment flaw classification in noisy conditions. *Ultrasonics*, 94:74 – 81.

Naomi Nevler, Sharon Ash, David J Irwin, Mark Liberman, and Murray Grossman. 2019. Validated automatic speech biomarkers in primary progressive aphasia. *Annals of Clinical and Translational Neurology*, 6(1):4–14.

J. Robin, J. E. Harrison, L. D. Kaufman, F. Rudzicz, W. Simpson, and M.J. Yancheva. 2020. Evaluation of speech-based digital biomarkers: Review and recommendations. *Digital Biomarkers*, 4(3):99–108.

I. Spasić, D. Owen, A. Smith, and K. Button. 2019. Klosure: Closing in on open–ended patient questionnaires with text mining. *Journal of Biomedical Semantics*, 10.

Martin J Tobin, Franco Laghi, and Amal Jubran. 2020. Why covid-19 silent hypoxemia is baffling to physicians. *American Journal of Respiratory and Critical Care Medicine*, 202(3):356–360.

Isabel Trancoso, Maria Joana Ribeiro Folgado Correia, Francisco Teixeira, Alberto Abad, Maria Catarina Tavares Botelho, and Bhiksha Raj. 2019.

Speech as a (private?) biomarker for speech affecting diseases. In *In ICIEA 2019 - The 14th IEEE Conference on Industrial Electronics and Applications*, Xi'an, China. IEEE. Keynote paper.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Wei Zhang, Chuanhao Li, Gaoliang Peng, Yuanhang Chen, and Zhujun Zhang. 2018. A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mechanical Systems and Signal Processing*, 100:439 – 453.

Richard M Zur, Yulei Jiang, Lorenzo L Pesce, and Karen Drukker. 2009. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical physics*, 36(10):4810–4818.

Benchmarking Robustness of Machine Reading Comprehension Models

Chenglei Si[1*] **, Ziqing Yang**[2]**, Yiming Cui**[2,3]**, Wentao Ma**[2]**, Ting Liu**[3]**, Shijin Wang**[2,4]
[1] University of Maryland, College Park, MD, USA
[2] Joint Laboratory of HIT and iFLYTEK Research (HFL), Beijing, China
[3] Research Center for SCIR, Harbin Institute of Technology, Harbin, China
[4] State Key Laboratory of Cognitive Intelligence, Beijing, China
[1]`clsi@terpmail.umd.edu`
[2]`{zqyang5,ymcui,wtma,sjwang3}@iflytek.com`
[3]`{ymcui,tliu}@ir.hit.edu.cn`

Abstract

Machine Reading Comprehension (MRC) is an important testbed for evaluating models' natural language understanding (NLU) ability. There has been rapid progress in this area, with new models achieving impressive performance on various benchmarks. However, existing benchmarks only evaluate models on in-domain test sets without considering their robustness under test-time perturbations or adversarial attacks. To fill this important gap, we construct AdvRACE (Adversarial RACE), a new model-agnostic benchmark for evaluating the robustness of MRC models under four different types of adversarial attacks, including our novel distractor extraction and generation attacks. We show that state-of-the-art (SOTA) models are vulnerable to all of these attacks. We conclude that there is substantial room for building more robust MRC models and our benchmark can help motivate and measure progress in this area. We release our data and code at `https://github.com/NoviScl/AdvRACE`.

1 Introduction

The goal of Machine Reading Comprehension (MRC) is to examine whether the model can understand the text and perform certain types of reasoning. To this end, many MRC benchmarks have been constructed in different domains, styles, and languages (Rajpurkar et al., 2016; Lai et al., 2017; Dua et al., 2019b; Cui et al., 2019, *inter alia*). While most of these benchmarks use leaderboards to compare different models' performance on in-domain test sets, they have ignored the important aspect of evaluating models' robustness.

The research on robustness of MRC models can be generally categorised into two directions: generalization to out-of-domain distributions and robustness under test-time perturbations. Regarding generalization to out-of-domain distributions, Talmor and Berant (2019) and the MRQA shared task (Fisch et al., 2019) investigated how well do MRC models trained on source datasets generalize on unseen datasets in different domains.

On evaluating MRC models under test time perturbations, previous works (Jia and Liang, 2017; Gan and Ng, 2019) evaluate models under only one specific attack. However, truly robust models should perform well on different types of perturbed test inputs instead of just one. In this work, we cover diverse types of adversarial attacks to allow for a more comprehensive evaluation of model robustness. Dua et al. (2019a) performed a set of synthetic perturbations to test models' robustness. However, their perturbations are only applicable to specific types of questions, yielding limited number of valid examples, limiting their usefulness in revealing model weakness. This may be because that they only performed perturbations on the questions. In contrast, our proposed methods are applied to different components (passages, questions, distractors) of the dataset and at different levels (sentence and character levels). In section 4.2, we show that all of our perturbations incur significant performance drops on SOTA models, leaving large room for future improvement.

Instead of constructing a new dataset from scratch, we leverage on an existing MRC dataset RACE (Lai et al., 2017), where we apply our proposed attacks on the RACE test set to form our AdvRACE benchmark. We choose to construct our benchmark based on RACE because: 1) The multiple-choice format supports more types of attacks. We generated new distractors to replace the original ones as a novel attack method. 2) RACE covers a diverse set of linguistic phenomena and reasoning types, and has been used widely for evaluating NLU performance (Yang et al., 2019; Liu

---

[*] Work done during an internship at HFL.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 634–644
August 1–6, 2021.

Method	AddSent	CharSwap	WordReplace	Paraphrase	DE	DG
AdvRACE (Ours) †	✔	✔	✚	✚	✔	✔
AddSent (Jia and Liang, 2017) †	✔	✗	✗	✗	✗	✗
SEA (Ribeiro et al., 2018) †	✗	✗	✗	✔	✗	✗
SCPN (Iyyer et al., 2018) †	✗	✗	✗	✔	✗	✗
charNMT (Belinkov and Bisk, 2018) †	✗	✔	✗	✗	✗	✗
breakNLI (Glockner et al., 2018) †	✗	✗	✔	✗	✗	✗
HotFlip (Ebrahimi et al., 2018)	✗	✔	✔	✗	✗	✗
Genetic (Alzantot et al., 2018)	✗	✗	✔	✗	✗	✗
PWWS (Ren et al., 2019)	✗	✗	✔	✗	✗	✗
UniversalTriggers (Wallace et al., 2019)	✔	✗	✗	✗	✗	✗
ORB (Dua et al., 2019a) †	✗	✗	✔	✔	✗	✗
TextFooler (Jin et al., 2020)	✗	✗	✔	✗	✗	✗
SememePSO (Zang et al., 2020)	✗	✗	✔	✗	✗	✗

Table 1: Comparison of AdvRACE to other adversarial attack work. ✚ indicates that the method was tried but not included in the benchmark due to poor performance. † indicates these methods are model-agnostic.

et al., 2019; Lan et al., 2020).

Our AdvRACE benchmark contains four test sets for four different adversarial attacks: AddSent, CharSwap, Distractor Extraction (DE) and Distractor Generation (DG). The performance under each attack allows for fine-grained and comprehensive analysis of model robustness. In addition, the advantages of our work include: 1) The pipeline we use to construct AdvRACE is highly efficient and transferable. It can be easily adopted on other MRC datasets to construct the corresponding adversarial test sets. 2) All of our adversarial perturbations are model-agnostic and do not require access to model parameters. Thus it allows for the evaluation of any model to reveal their weaknesses in robustness. 3) Our perturbations are label-preserving. Our human annotations ensured the high quality of our adversarial test sets (section 4.1).

2 Related Work

Adversarial attacks in NLP. Various adversarial attack methods have been proposed for NLP tasks. One type is white-box attack where perturbations are constructed based on models' gradients or parameters to exploit their weakness (Alzantot et al., 2018; Ebrahimi et al., 2018; Wallace et al., 2019). However, such attacks cannot be directly applied when there is no access to model internal parameters. Another type is black-box attack where the perturbation is constructed without accessing model parameters and is often based on heuristic rules or produced by specifically designed models (Jia and Liang, 2017; Ribeiro et al., 2018; Iyyer et al., 2018). While the above approaches construct attacks automatically, there are also attempts on human-in-the-loop adversarial example generation where human annotators are employed to write new test data that can fool the models into making wrong predictions (Nie et al., 2020; Wallace et al., 2018). However, such approach requires significant human and time resources, which may not be easily accessible.

Robustness benchmarks for MRC. Tang et al. (2020) created a Chinese MRC robustness benchmark by constructing distracting questions via retrieval from existing database and human writing. Wu et al. (2020) applied various common synthetic perturbations on the SQuAD dataset to analyse the factors affecting model robustness. Ribeiro et al. (2020) provided a general framework for automatically identifying various types of failures in NLP systems with CheckList. Our adversarial attacks have similar goals as the invariance test in CheckList. However, their template-based checks only examine limited failure types, thus less effective in revealing weaknesses in model robustness as shown by the relative small drop of model performances under invariance tests, as compared to our proposed attacks on MRC.

3 Adversarial Attacks

In this section, we describe the different types of adversarial attacks explored in AdvRACE. In Table 1, we present a comparison of our AdvRACE benchmark with other works on automatic construction of adversarial attacks.

In multiple-choice MRC, distractors naturally play an important part in affecting the difficulty of the dataset. Hence, we propose a novel type of adversarial attack method by constructing new sets of distractors to replace the original ones. We present two different methods of constructing new

Original	**Passage:** "Thanks for coming," Everett said, shaking hands with Mr. Hanson, the town councilor. "I'm curious about the ideas in your letter." Mr. Hanson nodded toward the parking lot near where they stood. ... Everett was relieved that the community representative seemed receptive to the idea. Mr. Hanson studied the photograph, and then asked, "If town council provides the money, how will you and your friends contribute?" Everett felt optimistic now. **Question:** Everett was trying to persuade Mr. Hanson to _ . **Answer:** provide money for a skateboard park **Options: 1.** beautify the neighbourhood **2.** build an art centre for children **3.** cut the area of the parking lot
AddSent	**Passage:** "Thanks for coming," Everett said, shaking hands with Mr. Hanson, the town councilor. "I'm curious about the ideas in your letter." Mr. Hanson nodded toward the parking lot near where they stood. "Please tell me more." Everett took a deep breath. "Ever since the school closed two years ago, this area has become worse. But if we clean up the litter and repair the fence, it will be a great place for a skateboard park." ***Zimmerman was trying to dissuade Mr. Morrison to build an art centre for children.*** Mr. Hanson scanned the broken concrete, nodding. ... Everett was relieved that the community representative seemed receptive to the idea. ***Larkin was trying to dissuade Mr. Lyons to beautify the neighbourhood.*** Mr. Hanson studied the photograph, and then asked, "If town council provides the money, how will you and your friends contribute?" Everett felt optimistic now. **Question/Answer/Options:** Same as **Original**.
CharSwap	**Passage:** "Thanks for coming," Everett said, shaking hands with Mr. Hasnon, the town ***councilor***."I'm curious about the ideas in your letter." Mr. Hanson nodded toward the ***pakring*** lot near where they stood. "Please tell me more. "Everett took a deep breath."Ever since the school closed two years ago, this ***aera*** has become worse . But if we clean up the litter and repair the fence, it will be a great place for a ***skateborad*** park." Mr. Hanson scanned the broken concrete, nodding. "The old school is being adapted to a community arts ***cenrte***. This ***aera*** could become a vital part of the ***neighbouhrood*** again." ... **Question:** Everett was ***tyring*** to ***perusade*** Mr. Hanson to _ . **Answer/Options:** Same as **Original**.
DE	**Passage/Question/Answer:** Same as **Original**. **Options: 1.** build the ramps **2.** provide **3.** repair the fence, it will be a great place for a skateboard park
DG	**Passage/Question/Answer:** Same as **Original**. **Options: 1.** help the town councilor **2.** help design and build the ramps **3.** buy the old school

Table 2: Examples of each attack applied on the same original test example. ***Italic*** parts are altered by our adversarial attack perturbations.

distractors: distractor extraction (DE) and distractor generation (DG). Apart from them, we have also experimented four other types of adversarial attack methods: distracting information insertion (AddSent), character swap (CharSwap), word replacement and paraphrase. Among them, we find that WordReplace and Paraphrase do not work well on RACE, hence being excluded from our final AdvRACE benchmark. We provide examples of each adversarial attack in Table 2. The details of the two failed attempts can be found in the Appendix.

3.1 Distractor Extraction

For distractor extraction, we aim to extract spans from the passages as new distractors. This is motivated by the observation that most passages contain distracting sequences that are relevant to the question while semantically different from the correct answer. Such distracting sequences can be especially effective as current MRC models reply largely on text matching (Si et al., 2019).

The key challenge is how do we extract such distracting sequences from the passages without annotated supervised data. In order to solve this challenge, we propose a novel distractor extraction method. We first turn RACE into span-extraction format and adopt the span-extraction QA model used in Devlin et al. (2019) except that we use ALBERT as our backbone model. During training, we insert the correct answers of RACE questions into the passages as the gold answer span. The ALBERT span extractor is trained to select the answer span from the passage. During inference, we use the trained model to extract spans from the passages for the test questions but without inserting the correct answer. In this way, since the answers in RACE have not appeared in the passages in verbatim, there are no gold answer spans for the test examples. However, the extracted spans with high probability can be considered as likely options for the question, which can serve as strong distractors. For post-processing, we select 3 distractors among the top 20 candidate spans that have low lexical overlap with each other and also low lexical overlap with the correct answer, so that they are diverse and label-preserving.

3.2 Distractor Generation

Another way to construct new distractors is to generate them based on the passage and question. We

	BERT	**RoBERTa**	**XLNet**	**ALBERT**	*Average*	**Valid**	**Correct**
Original	68.5	83.7	79.9	86.0		100.0%	100.0%
AddSent	25.5 (*-62.8%*)	62.3 (*-25.6%*)	54.1 (*-32.3%*)	69.2 (*-19.5%*)	*-35.1%*	98.0%	89.8%
CharSwap	48.8 (*-28.8%*)	69.4 (*-17.1%*)	63.4 (*-20.7%*)	73.0 (*-15.1%*)	*-20.4%*	100.0%	94.0%
Distractor Extraction	32.9 (*-52.0%*)	48.8 (*-41.7%*)	44.0 (*-44.9%*)	52.0 (*-39.5%*)	*-44.5%*	98.0%	95.9%
Distractor Generation	56.3 (*-17.8%*)	68.9 (*-17.7%*)	64.7 (*-19.0%*)	70.9 (*-17.6%*)	*-18.0%*	98.0%	93.9%
Average	40.9 (*-40.3%*)	62.4 (*-25.4%*)	56.6 (*-29.2%*)	66.3 (*-22.9%*)			

Table 3: Left: Attack results on different models. Right: Human evaluation results. *Numbers* in brackets are the percentage drop in performance. Average is computed over all the adversarial test sets for each model, and over all models for each adversarial attack.

frame it as a sequence-to-sequence (Seq2Seq) task where the input is the concatenation of the passage and question, and the output is the distractor. Previous distractor generation works (Gao et al., 2019; Zhou et al., 2020) adopted this approach. However, none of these works has explored using the generated distractors as a way for attacking MRC models. In our work, we use the filtered data from Gao et al. (2019) for training the DG model, where distractors with low semantic relevance with the passages are pruned. For the backbone model, we adopt UniLM (Dong et al., 2019), a competitive pretrained model unifying NLU and NLG. We use beam search to find the top k (beam size, we used $k = 50$) candidate distractors and select the top 3 among them so that their Jaccard distance between each other is larger than 0.5, to ensure diversity of the 3 selected distractors. We provide additional DE and DG examples in Table 9.

3.3 AddSent

Inspired by the method proposed by Jia and Liang (2017), we propose an improved AddSent method to make use of the human-written distractors in RACE rather than generating new fake answers to construct strong distracting sequences to insert into the original passages. The motivation is to add distracting information that appears similar to the question and the distractors so that it can better mislead models. We use the following procedure to construct the perturbations:

1. We change all nouns and numbers in the questions to their nearest word in GloVe (Pennington et al., 2014) embedding space with the same part-of-speech.
2. We replace adjectives, adverbs, verbs in the questions with their antonym in WordNet (Fellbaum, 2000) with the same part of speech. We only change one word in each question to its antonym to prevent confusions caused by multiple negations (e.g., 'likely to obey' being changed to 'unlikely to disobey').
3. If no words are changed during Step 1 and 2 (28.9% of test questions), we insert the negation word 'not' at the appropriate position in the question. The first three steps ensure that there is a flip in the semantic meaning of the original question.
4. We randomly sample a distractor from the original three distractors of the question, and concatenate it with the altered question.
5. We insert the concatenated sequence into a random position of the passage.
6. We repeat Step 1 to 5 one more time with different replacement words from GloVe and WordNet, and using a different distractor. This makes the perturbation more misleading for the model since there is more matching.

3.4 CharSwap

It is shown in Belinkov and Bisk (2018) that NMT performance drops significantly when there are spelling errors in the data. We adapt their approach in MRC to swap two adjacent letters in a word without altering the first or last letters. Although they show that with more tokens altered, the performance gets worse, empirically we find that applying such perturbation to all words in the dataset sometimes impacts the readability of the text and cause difficulty even for humans to perform well. As a result, we only apply the CharSwap perturbation to the following words with character length of four and above: 1) The non-stopwords in the question. 2) Non-stopwords in the passage that have also appeared in the question and its corresponding options. This is based on the observation that these words are usually keywords for solving the question. There are a total of 7.1% words being altered using this method.

4 Evaluation

We apply the four attacks in Section 3 on the original RACE test set to form AdvRACE, which results in four sets of adversarial test sets: AddSent, CharSwap, DE, and DG. Each adversarial test set has 4,934 examples, same as the original RACE test set. We perform human and model evaluations on these test sets.

4.1 Human Evaluation

We randomly sample 50 questions from each adversarial test set in AdvRACE as well as the original test set, and ask paid corporate professional data annotators to answer the MRC questions. To assess the validity of our adversarial data, we add an additional option for each question: unanswerable. If the annotator chooses unanswerable for a question, we further ask the annotator to choose the specific reason why the question is unanswerable, which includes the following options: missing key information, conflicting information, uninterpretable content, no correct options, multiple correct options. The results are shown in the last 2 columns of Table 3. **Valid** rate measures how many examples are considered by the annotators as interpretable (semantically and syntactically acceptable) and answerable. **Correct** rate measures how many examples are answered correctly by the annotators. All the four adversarial test sets have high valid rate close to 100%. The correct rate of the four adversarial test sets are also above the turker performance of 73.3% as reported in Lai et al. (2017). These indicate that the four adversarial test sets in AdvRACE remain well solvable for humans. We also manually analysed the examples answered wrongly by the human annotators. For all the four adversarial test sets, we find that the drop in human performance (correct rate) is due to unrelated human errors (e.g., misunderstanding of the original text) rather than caused by our perturbations.

4.2 Model Evaluation

For model evaluation, all the models are trained on the original RACE training set, tuned hyperparameters on the original RACE dev set, and tested on our AdvRACE benchmark. We evaluate four competitive models: BERT-Large, XLNet, RoBERTa-Large, ALBERT-xxLarge. We concatenate each candidate answer with the corresponding passage and question, then encode each of these four concatenated sequences and pass the representations of the first position of each sequence (i.e., representations of the `[CLS]` token) through a fully-connected layer for answer prediction. The results are presented in Table 3. In addition to reporting the accuracy of each model on each test set, we also report the percentage performance drop relative to the performance on the original test set for the adversarial test sets.

From the results, we find that: 1) BERT is the most vulnerable among these four models (by comparing the average percentage drop in performance in the last row). Models that achieve higher performance on the original test set generally suffer less relative performance drop from the adversarial attacks. 2) Distractor extraction is more effective than distractor generation in terms of attacking the models, probably due to the fact that these models rely heavily on text matching (Si et al., 2019). 3) All of the adversarial attacks in our AdvRACE benchmark incur a significant drop of models' performance, which is much more effective than previous attempts (Dua et al., 2019a) in revealing weaknesses of the most competitive MRC models.

4.3 Adversarial Training

A common baseline to defend such adversarial attacks is to add adversarial examples in training, *i.e.*, adversarial training. We experiment such strategy as a baseline, and find that adversarial training is not a truly reliable defense method as it largely exploits dataset artifacts. We present the detailed adversarial training experiments as well as the follow-up analysis in the Appendix (section D). We believe that better defense techniques are needed to reliably improve model robustness for MRC.

5 Conclusion

In this work, we present AdvRACE, a benchmark that evaluates MRC models under diverse types of adversarial attacks. AdvRACE consists of our proposed distractor extraction and generation attacks as well as improved versions of existing attacks adapted to work effectively on MRC. The performance of four competitive MRC models drops significantly on AdvRACE, indicating the effectiveness of our methods for analysing model weaknesses. We hope our work can motivate the development of more robust models that perform well on different perturbations.

Impact Statement

Our AdvRACE dataset was constructed based on the original RACE dataset, which was released under an MIT license, allowing for free reuse. We have also obtained the permission from the original authors of RACE to use it to construct AdvRACE for research purposes.

The human annotators involved in this project are all full-time employed corporate annotators, who are well-compensated and work in good working conditions.

We believe that building robust and reliable MRC models is crucial for them to be deployed in real-life applications. Our work is far from enough to thoroughly measure the true robustness of MRC models. We hope that it will be a first step leading to more efforts along this direction.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. Generating Natural Language Adversarial Examples. In *EMNLP*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation. In *ICLR*.

Yiming Cui, Ting Liu, Li Xiao, Zhipeng Chen, Wentao Ma, Wanxiang Che, Shijin Wang, and Guoping Hu. 2019. A Span-Extraction Dataset for Chinese Machine Reading Comprehension. In *EMNLP-IJCNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*.

Dheeru Dua, Ananth Gottumukkala, Alon Talmor, Sameer Singh, and Matt Gardner. 2019a. ORB: An Open Reading Benchmark for Comprehensive Evaluation of Machine Reading Comprehension. *arXiv*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019b. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *NAACL-HLT*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for NLP. In *ACL*.

Christiane Fellbaum. 2000. WordNet : an electronic lexical database. *Language*, 76:706.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 Shared Task: Evaluating Generalization in Reading Comprehension. *arXiv*.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the Robustness of Question Answering Systems to Question Paraphrasing. In *ACL*.

Yifan Gao, Lidong Bing, Piji Li, Irwin King, and Michael R. Lyu. 2019. Generating Distractors for Reading Comprehension Questions from Real Examinations. In *AAAI*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *ACL*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *NAACL-HLT*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *EMNLP*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? Natural Language Attack on Text Classification and Entailment. In *AAAI*.

Divyansh Kaushik and Zachary Chase Lipton. 2018. How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks. In *EMNLP*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *EMNLP*.

Zhen-Zhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *ACL*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically Equivalent Adversarial Rules for Debugging NLP models. In *ACL*.

Marco Túlio Ribeiro, Tongshuang Wu, C. Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP models with CheckList. In *ACL*.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. 2019. What does BERT learn from multiple-choice reading comprehension datasets? *arXiv*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An Empirical Investigation of Generalization and Transfer in Reading Comprehension. In *ACL*.

Hongxuan Tang, Jing Liu, Hongyu Li, Yu Hong, Hua Wu, and Hai-Feng Wang. 2020. DuReaderRobust: A Chinese Dataset Towards Evaluating the Robustness of Machine Reading Comprehension Models. *arXiv*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *EMNLP*.

Eric Wallace, Pedro Rodriguez, S. L. Feng, Ikuya Yamada, and Jordan L. Boyd-Graber. 2018. Trick Me If You Can: Human-in-the-Loop Generation of Adversarial Examples for Question Answering. *TACL*.

Yicheng Wang and Mohit Bansal. 2018. Robust Machine Comprehension Models via Adversarial Training. In *NAACL-HLT*.

Winston Wu, Dustin Arendt, and Svitlana Volkova. 2020. Evaluating Neural Machine Comprehension Model Robustness to Noisy Inputs and Adversarial Attacks. In *EACL*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*.

Yuan Zang, Chenghao Yang, Fanchao Qi, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Textual Adversarial Attack as Combinatorial Optimization. In *ACL*.

Xiaorui Zhou, Senlin Luo, and Yunfang Wu. 2020. Co-Attention Hierarchical Network: Generating Coherent Long Distractors for Reading Comprehension. In *AAAI*.

Appendix

A Failed Attacks

We introduce the two attacks that we explored but did not work well: Word Replacement and Paraphrase.

A.1 Word Replacement

For word replacement attack (WordReplace), we adopt the state-of-the-art method: SememePSO (Zang et al., 2020), and we used RoBERTa as the target model to generate the adversarial examples. Then we transferred these adversarial examples to attack other models. The original SememePSO attack was applied on sentence classification tasks. Since the input of RACE is much longer, in order to keep the running time within budget constraints, we have set the population size to 10 and number of maximum optimization iterations to 5. We have also tried simple random replacement strategies (randomly select a candidate to replace the original word) which is not effective in attacking the models. The results are shown in Table 4, we find that although the attack is effective in attacking the target model RoBERTa, it is much less effective on other models, which does not satisfy our model-agnostic requirement, and hence this attack is not included in AdvRACE.

A.2 Paraphrase

For Paraphrase, we adopted the Syntactically Controlled Paraphrase Network (SCPN) (Iyyer et al., 2018) which generates paraphrases of a sentence based on the given parse templates. The original method was proposed to attack single-sentence text classification problems. We generate paraphrases for sentences in the passages as an attack. We generated the Paraphrase test set with SCPN, using the top two levels of the linearized parse tree of the sentence as the input parse template. We have tried using other templates (such as using the most frequent parse template in the test set passage sentences) that are different from the original parse of the sentences, which results in more syntactic variances, but also lower valid rate. As shown in Table 4, although it is model-agnostic, during human evaluation, we find that 15 out of the 50 randomly sampled examples are labeled as unanswerable. Among the 15 unanswerable examples, 13 examples are labeled as missing key information, the other 2 are labeled as uninterpretable. This suggests that it is difficult to generate high-quality label-preserving adversarial examples by paraphrasing for MRC. Due to the low validity rate of the paraphrased examples, we exclude Paraphrase in AdvRACE to ensure that our benchmark is solvable for human.

	BERT	**RoBERTa**	**XLNet**	**ALBERT**
Original	68.5	83.7	79.9	86.0
WordReplace	64.2	70.7	74.7	80.5
Paraphrase	59.4	72.3	68.2	73.7

Table 4: Performance on WordReplace and Paraphrase test set in comparison with performance on the original RACE test set.

B Dataset Statistics

Each adversarial test set in AdvRACE contains 4934 questions, which corresponds to 1407 passages. The average length of each component is shown in Table 5.

Test Set	**Passage Len**	**Question Len**	**Option Len**
Original	316.2	11.0	6.3
AddSent	350.6	11.0	6.3
CharSwap	316.2	11.0	6.3
DE	316.2	11.0	6.8
DG	316.2	11.0	5.7

Table 5: Dataset Statistics of the original RACE test set and the adversarial test sets in AdvRACE.

C Model Benchmarking Hyper-parameters

The four models being benchmarked in the paper are: BERT-large, XLNet, ALBERT-xxlarge and RoBERTa-large. For both finetuning on the original RACE training set and adversarial training, we use learning rate 1.5e-5, weight decay 0.1, warmup ratio 0.06, max sequence length 512, batch size 32, max number of epochs 4. All experiments are done on Tesla V100 GPUs.

D Adversarial Training

Adversarial training (AT) is a common method to improve the robustness of NLP models where adversarial examples are added into the training data. In this section, we first examine how different AT strategies improve the model performance on various adversarial attacks. Next, we analyse the reasons behind the performance improvement

	Original	+AddSent	+CharSwap	+DE	+DG	Mix
Original	83.7	82.1	81.5	80.0	81.1	81.0
AddSent	62.3	88.0	57.8	76.2	65.6	88.0
CharSwap	69.4	67.5	74.1	66.4	67.2	75.4
Distractor Extraction	48.8	47.0	44.9	89.4	60.3	89.9
Distractor Generation	68.9	66.6	67.2	73.3	86.5	87.8
Average	66.6	70.2	65.1	77.1	72.1	84.4

Table 6: Results for adversarial training. Each column corresponds to one adversarial training method.

	Orig	AddSent-AT	Mix-AT
AddSent	62.3	88.0	88.0
Diagnosis	95.8	19.7	11.3

Table 7: Model performance on AddSent and Diagnosis test set.

	Orig	DE	DG
PQ-remove	39.8	73.8	83.7
Random	25.0	25.0	25.0

Table 8: Results for PQ-remove training and testing.

and highlight important flaws of using adversarial training as defense.

D.1 Adversarial Training Experiments

Previous works have used adversarial training to improve model robustness under specific attacks (Jia and Liang, 2017; Wang and Bansal, 2018). In this section, we first explore this targeted adversarial training setting where we randomly sample 25% of training data, apply the attack on the sampled training data, and add the perturbed data into the original training set as augmentation. Furthermore, we explore a mix adversarial training setting, where we add all the four types of adversarial data into the original training set together. In this setting, each adversarial attack is applied to 25% of the training data, so the resultant augmented training data has double size of the original training set.

We use RoBERTa for all the adversarial training experiments with the same hyperparameters. The experiment results are shown in Table 6, where the first column is the model's performance on the original RACE test set, the middle four columns are results of targeted adversarial training where each of the individual adversarial attack data is added in training. The last column shows results for the mix adversarial training where all the four types of AT data are added during training. We outline a few interesting findings from our experiments:

- Adversarial training on one type of adversarial data only improves model's performance on the attack that the model is adversarially trained on, while generally decreasing its performance on other test sets.
- Adversarial training on Distractor Extraction improves the model's performance on AddSent as well, possibly because that it makes the model rely less on simple text matching. Adversarial training on Distractor Generation significantly improves the model's performance on the DE test set, while adversarial training on DE also moderately improves the model performance on the DG test set, possibly due to the similar distractor replacement format of the two attack methods.
- Mix Adversarial training significantly improves the model's performance on all the adversarial test sets with slight drop on the original test set.

D.2 Understanding the effects of Adversarial Training

While it appears that adversarial training has significantly improved the model's performance under adversarial attacks, it is unclear whether it has really made the model more robust. Moreover, we find that on AddSent, DE, and DG, the adversarially augmented model achieves higher performance on the adversarial test set than the original test set, which may suggest overfitting. Hence, we analyse what is learned by the model during AT.

- **AddSent Adversarial Training**

We hypothesize that there exists spurious patterns in the AT data which are exploited by the model. To test this hypothesis, we construct a diagnosis test set by inserting the concatenation of the question and the correct answer into the passage, which does not change the labels and adds additional lexical matching for the correct answer. In Table 7, three models: RoBERTa finetuned on the original training set (Orig), under AddSent adversarial training (AddSent-AT), and under mix adversarial training (Mix-AT) are evaluated on the AddSent test set as well as our diagnosis test set. Comparing the first line with the second line, we find that RoBERTa finetuned on the original training set achieves much higher accuracy on the diagnosis test test, indicating that the insertion of correct answers makes it easier for the model to identify the correct answer. However, AddSent-AT and Mix-AT achieve miserable performance on the diagnosis test set, indicating such adversarially trained models are exploiting spurious patterns to score well and the model will fail when such patterns do not apply in the new test set. Hence it is likely that the high performance after AT is an overestimate, and better measures should be taken as defense to ensure improvement in the model's robustness.

- **DE and DG Adversarial Training**

To examine the exploiting of spurious statistical cues in the extracted and generated distractors, we employ partial data training (Kaushik and Lipton, 2018; Si et al., 2019) where we train and test RoBERTa on RACE without the passages and questions (i.e., we only input the options to the model). Under this setting, it is impossible for humans to correctly answer the questions since only the options are provided. As shown in Table 8, we train RoBERTa on: the original training set, DE adversarial training data, and DG adversarial training data, and test on their respective test sets. Under such option-only (PQ-remove) setting, RoBERTa achieves performance that is much higher than random on the original, DE, and DG test sets. This indicates that all the three datasets (original, DE and DG) have spurious statistical cues that are exploited to achieve high performance in the option-only setting. Moreover, models are exploiting such cues in DE and DG datasets to a much larger extent compared to on the original dataset. Thus, adding these examples into training may lead to an overestimate of the model's performance as the model can easily exploit these spurious patterns to increase the performance.

In summary, we find that the adversarially trained models may not be more robust, and the high scores achieved by adversarial training may have overestimated the models' robustness. We believe that more reliable defense methods should be developed to improve the model's robustness without exploiting spurious statistical cues as a shortcut. We leave such exploration to future work.

E Additional Examples

We present more examples of our DE and DG attacks in Table 9.

DE	**Passage:** I personally would be more concerned for the teenager who has to become independent without having any familiarity with working. The biggest reason teens should be allowed to work is that it is a healthy way of earning money. My dad told me, “Having a job is a good way to save up for things you are going to need or want to do. You get a lot more out of things if you are financially responsible for them.” Kids need the freedom to choose how to spend their money. The sooner they have an income, the sooner they can learn how to use money wisely. If they are not allowed to work in high school, they may run into trouble in the future. **Question:** The author’s father advised him to _ . **Answer:** earn money to get what he wants **Predicted:** work
DE	**Passage:** In a world with limited land, water and other natural resources, the harm from the traditional business model is on the rise. Actually, the past decade has seen more and more forests disappearing and the globe becoming increasingly warm. People now realize that this unhealthy situation must be changed, and that we must be able to develop in sustainable ways. That means growth with low carbon or development of sustainable products. In other words, we should keep the earth healthy while using its supply of natural resources. Today, sustainable development is a proper trend in many countries. According to a recent study, the global market for low-carbon energy will become three times bigger over the next decade. China, for example, has set its mind on leading that market, hoping to seize chances in the new round of the global energy revolution. **Question:** What is the main purpose of the passage? **Answer:** To introduce a new business model. **Predicted:** sustainable development
DE	**Passage:** Phyllis Lee of Singapore knew something wasn’t right. Her younger son, Alex, then six years old, was getting good grades in his private kindergarten classes. But Lee realized something was wrong when Alex came home one day, crying, with “zeroes” on his Chinese spelling test. Lee decided to investigate. According to Lee, the teacher would frequently describe Alex’s Chinese writing as ’ghost writings’ and made him a laughing stock in class instead of helping him out. Lee, 46, spent the next 12 months teaching Alex and still helps him when necessary. She not only taught him the formation of the Chinese words but also their origin so that he could understand better, often taking more than an hour to read a simple paragraph. Visits to the library and surfing the Internet kept her up to speed on teaching materials. Alex’s grades improved and by the end of Year One, he had become one of the top students in Chinese in his class. Alex is keeping an A - plus average in all subjects, and his mother’s involvement is a big reason behind it. **Question:** Alex arrived home with tears because _ . **Answer:** he failed in the Chinese exam **Predicted:** “zeroes” on his Chinese spelling test
DG	**Passage:** A new survey finds that more than eighty percent of Internet users in the United States search for health information online. The survey found that searching online is one of the leading ways that people look for a second opinion though doctors are still the main source of health information. Forty-four percent of people are actually looking for doctors or other providers when they search for health information online. Another finding of the survey: Two-thirds of Internet users look online for information about a specific disease or medical condition. The Internet has also become an important source of emotional support for people with health problems. Susannah Fox says one in five Internet users has gone online to find other people who have the same condition. It was more popular among people with more serious health issues–one in four people living with chronic diseases. They are so eager to find other people online who share their health concerns. The rise of social networking has made it easier for people with rare diseases to connect with each other and feel less alone. **Question:** By using social networking, patients with rare diseases can _ . **Answer:** get emotional comfort **Predicted:** find other people online
DG	**Passage:** I was personally discouraged by an adult during my high school. After telling her what university I wanted to attend, she plainly told me I would not get in. I was completely shocked and angry at her statement. The adult may have not intended to hurt me with her words, but it had an after effect. The meeting made me think she had no belief that I could possibly succeed in the future. After that it caused me to try to avoid any future meeting with adults until absolutely necessary. We are all human; therefore, we can all understand that some days are more challenging. But when people allow situations of stress to consume them, they cannot perform their best. People should pay attention to how their reactions could affect the person they are interacting with. adults who come off in a rude and aggressive way through communication have an effect on teenagers’ mind. Adults whether in schools, or any other institutions should work to tear down walls gently and create a safe space for the person they are serving. These adults should also help and contribute to a person’s academic, personal and professional growth. High school is an important time when young people need someone to believe in them. In conclusion, I would like to add that it is not completely up to adults only. Students are responsible for seeking help from adults who are in authority positions. They are also responsible for the way they approach adults in their academic surroundings. Students can expect to be treated in a respectful way when they express at the beginning. Generally the responsibility lies on both parties. When both sides can communicate in a polite manner then the complete environment of the school has the potential to develop well. **Question:** The passage is mainly about _ . **Answer:** the adults’ role in school **Predicted:** adults and teenagers

Table 9: Examples of DE and DG that successfully fooled RoBERTa model.

Improving BERT with Syntax-aware Local Attention

Zhongli Li[1]*, **Qingyu Zhou**[2]†, **Chao Li**[2], **Ke Xu**[1], **Yunbo Cao**[2]
[1]Beihang University
[2]Tencent Cloud Xiaowei
`{lizhongli@,kexu@nlsde.}buaa.edu.cn`
`{qingyuzhou,diegoli,yunbocao}@tencent.com`

Abstract

Pre-trained Transformer-based neural language models, such as BERT, have achieved remarkable results on varieties of NLP tasks. Recent works have shown that attention-based models can benefit from more focused attention over local regions. Most of them restrict the attention scope within a linear span, or confine to certain tasks such as machine translation and question answering. In this paper, we propose a syntax-aware local attention, where the attention scopes are restrained based on the distances in the syntactic structure. The proposed syntax-aware local attention can be integrated with pretrained language models, such as BERT, to render the model to focus on syntactically relevant words. We conduct experiments[1] on various single-sentence benchmarks, including sentence classification and sequence labeling tasks. Experimental results show consistent gains over BERT on all benchmark datasets. The extensive studies verify that our model achieves better performance owing to more focused attention over syntactically relevant words.

1 Introduction

Recently, Transformer (Vaswani et al., 2017) has performed remarkably well, standing on the multi-headed dot-product attention which fully takes into account the global contextualized information. Several studies find that self-attention can be enhanced by local attention, where the attention scopes are restricted to important local regions. Luong et al. (2015); Yang et al. (2018); Xu et al. (2019); Nguyen et al. (2020) utilize dynamic or fixed windows to perform local attention. Strubell et al. (2018); Zhang et al. (2020); Bugliarello and Okazaki (2020) explore to utilize syntax to restrain attention for better performance, but each of them confines to a certain task.

In this work, we propose a **s**yntax-aware **l**ocal **a**ttention (**SLA**) which is adaptable to several tasks, and integrate it with BERT (Devlin et al., 2019). We first apply dependency parsing to the input text, and calculate the distances of input words to construct the self-attention masks. The local attention scores are calculated by applying these masks to the dot-product attention. Then we incorporate the syntax-aware local attention with the Transformer global attention. A gate unit is employed for each token in each layer, which determines how much attention is paid to syntactically relevant words. We lift weights from existing pre-trained BERT, and evaluate our models on several single-sentence benchmarks, including sentence classification and sequence labeling tasks. Experimental results show that our method achieves consistent performance gains over BERT and outperforms previous syntax-based approaches on the average performance. Furthermore, we compare our syntax-aware local attention with the window-based local attention. We find that the syntax-aware local attention is more involved in the aggregation of local and global attention. The attention visualization also validates the syntactic information supports to capture important local regions.

To summarize, this paper makes the following contributions: i) SLA can capture the information of important local regions on the syntactic structure. ii) SLA can be easily integrated to Transformer, which allows initialization from pre-trained BERT by increasing very few parameters. iii) Experiments show the effectiveness of SLA on various single-sentence benchmarks.

---

[*] Contribution done during internship at Tencent Cloud Xiaowei.

[†] Corresponding author.

[1] The code is available at `https://github.com/Neutralzz/syntax_aware_local_attention`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 645–653
August 1–6, 2021.

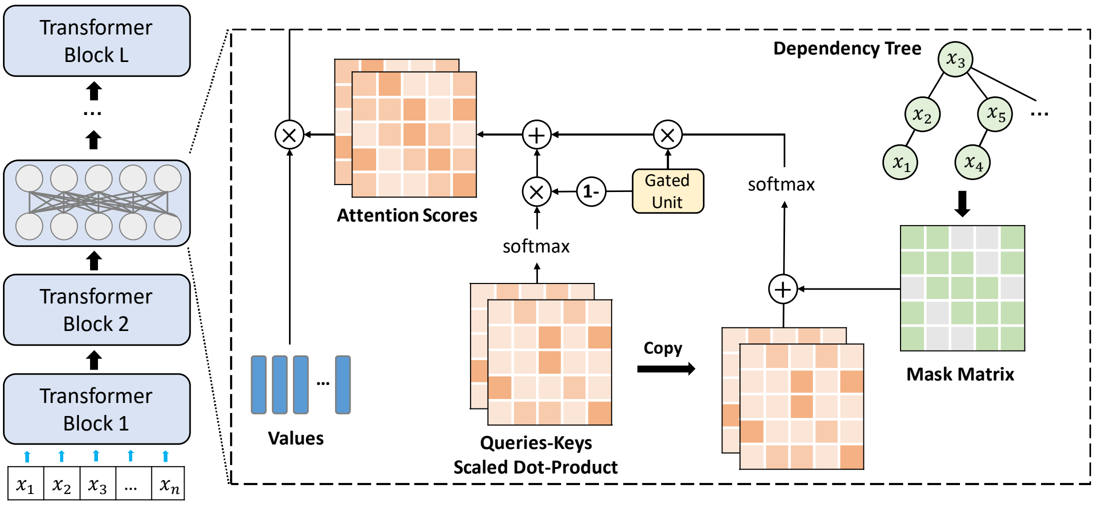

Figure 1: An overview of our model.

2 Related Work

2.1 Transformer Attention

Transformer (Vaswani et al., 2017) use stacked self-attentions to encode contextual information for input tokens. The calculation of self-attention depends on the three components of queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$, which are projected from the hidden vectors of the previous layer. Then the attention output $\mathbf{A}$ of one head is computed as follows:

$$\begin{aligned} \mathbf{M}_{ij} &= \begin{cases} 0, & \text{allow to attend} \\ -\infty, & \text{prevent from attending} \end{cases} \\ \mathbf{A} &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} + \mathbf{M}\right)\mathbf{V} \end{aligned} \quad (1)$$

where d is the dimension of keys and the mask matrix $\mathbf{M}$ controls whether two tokens can attend each other. Within the standard self-attention layer, global attention mechanism is employed that each token provides information to other tokens in the input sentence.

2.2 Local Attentions

Local attention involves limiting each token to attend to a subset of the other tokens in the input. Many works utilize a fixed or dynamic window to derive the important local regions. Luong et al. (2015) first propose a Gaussian-based local attention and increase BLEU scores for neural machine translation. Yang et al. (2018) improve the method of Luong et al. (2015) by predicting a central position and window size to model localness. Compared with Yang et al. (2018), Nguyen et al. (2020) attempt to derive the local window span by a soft-masking method. However, Levy and Goldberg (2014) suggest that more informative representations can be learned from the syntactic structure, instead of a window of surrounding tokens. Strubell et al. (2018) propose to train one attention head to attend to each token's syntactic parent for semantic role labeling. Zhang et al. (2020) also leverage the syntactic information to self-attention, but confine to question answering. Thus, we explore to take advantage of the syntactic structure to improve the model performance on various benchmarks.

3 Approach

In this section, we first introduce the syntax-aware local attention, and then integrate it with standard Transformer attention. As shown in Figure 1, we extend the Transformer layer with the syntax-aware local attention. Syntax-based masking is applied to the dot-product of queries and keys. The final attention scores are computed by incorporating local attention with standard global attention. We stack new layers and initialize weights from pre-trained BERT.

3.1 Syntax-aware Local Attention

We derive syntactic structure from dependency parsing, and treat it as an undirected tree. Each token x_i is mapped to a tree node v_i, and the distance of node v_i and v_j is denoted by $dis(v_i, v_j)$. However, the input may be an ungrammatical sentence in some tasks, and the dependency parser is not very accurate. Thus, we calculate the distance from neighboring tokens of x_i to token x_j as:

$$D(i,j) = \min dis(v_k, v_j), \quad k \in [i-1, i+1] \quad (2)$$

The motivation is that many attention heads specialize in attending heavily on the next or previous token (Clark et al., 2019). Then, in order to determine whether token x_j can attend to token x_i, a threshold m is applied to restrict the distance $D(i,j)$. For simplification, the mask matrix $\mathbf{M}^{loc}$ calculation can be formulated as:

$$\mathbf{M}_{ij}^{loc} = \begin{cases} 0, & D(i,j) \leq m \\ -\infty, & otherwise \end{cases} \quad (3)$$

Given the query $\mathbf{Q}$ and key $\mathbf{K}$ projected from the hidden vectors $\mathbf{H}$, the syntax-aware local attention scores $\mathbf{S}^{loc}$ are formally defined as:

$$\mathbf{S}^{loc} = \text{softmax}(\frac{\mathbf{QK}^T}{\sqrt{d}} + \mathbf{M}^{loc}) \quad (4)$$

where d is the dimension of keys. In this local attention, two tokens can attend to each other only if they are close enough in the dependency tree.

3.2 Attention Aggregation

As shown in Figure 1, the final attention is the aggregation of syntax-aware local attention and Transformer attention. We denote the Transformer attention scores by $\mathbf{S}^{glb}$. A gated unit is used to combine the global and local attention scores. The gate value g_i for each token x_i is calculated as follows:

$$g_i = \sigma(\mathbf{W}_g h_i + b_g), \quad (5)$$

where h_i is the hidden vector of token x_i from the previous layer, $\mathbf{W}_g$ is a learnable linear transformation and b_g is the bias. Then the attention output $\hat{\mathbf{A}}_i$ is calculated as a weighted average over values $\mathbf{V}$, and the weights are derived from global and local attention scores:

$$\hat{\mathbf{A}}_i = (g_i \mathbf{S}_i^{loc} + (1-g_i)\mathbf{S}_i^{glb})\mathbf{V}. \quad (6)$$

A larger gate value means more focused attention over syntactically relevant words. It can be seen that, if all the outputs of gated units are equal to 0, we could obtain the standard Transformer attention. Compared with the original architecture, our self-attention layer has one more input ($\mathbf{M}_{loc}$) and two more trainable parameters ($\mathbf{W}_g$ and b_g). Thus, we can easily lift weights from existing pre-trained BERT models.

4 Experiments

4.1 Experimental Setup

Benchmarks We use two English single-sentence classification datasets from the GLUE benchmark (Wang et al., 2018). We test on the CoLA and SST-2 datasets for acceptability and sentiment classification. Besides, we evaluate our method on two sequence labeling tasks: named entity recognition (NER) and grammatical error detection (GED). We use the CoNLL-2003 and FCE datasets for NER and GED, respectively. The training procedures are introduced in Appendix A.1.

Configuration All the training experiments are based on BERT. We use the uncased version of BERT for CoLA and SST-2, and the cased version for CoNLL-2003 and FCE. We derive dependency tree using Spacy[2]. More implementation details are reported in Appendix A.2.

Baselines We apply the syntax-aware local attention (**SLA**) to BERT. In addition to comparing with BERT, we also investigate the following approaches:

SGNet Zhang et al. (2020) present a syntax-guided self-attention layer, where each word is limited to interact with all of its syntactic ancestor words. Then they stack this layer on the top of the pre-trained BERT model[3], instead of modifying the Transformer architecture.

LISA Strubell et al. (2018) restrict each token to attend to its syntactic parent in one attention head[4]. We apply it to BERT and add the corresponding supervision at the last attention head in each Transformer layer.

Besides, we implement the window-based local attention (**WLA**), which allows each token to attend to the neighboring tokens within a window size $2k+1$ (varying k in $\{3,4,5\}$). Then it is also integrated with BERT as shown in Section 3.2.

4.2 Main Results

Experimental results are shown in Table 1. We report results on the dev set of CoLA and SST-2 and the test set of CoNLL-2003 and FCE. We employ t-tests to see if the mean difference differed from 0 between the standard attention and our proposed attention. It can be seen that our

---

[2] `https://spacy.io/`

[3] `https://github.com/cooelf/SG-Net`

[4] `https://github.com/strubell/LISA`

			CoLA	**SST-2**	**CoNLL-2003**			**FCE** (M2)		
Models	Params	Avg.	MCC	Acc	P	R	F_1	P	R	$F_{0.5}$
State-of-the-art Models										
ERNIE 2.0 (Sun et al., 2020)	-	-	65.4	96.0	-	-	-	-	-	-
T5 (Raffel et al., 2019)	-	-	71.6	97.5	-	-	-	-	-	-
BERT-MRC (Li et al., 2020)	-	-	-	-	92.3	94.6	93.0	-	-	-
BERT-GED (Bell et al., 2019)	-	-	-	-	-	-	-	65.0	38.9	57.3
Base-size Models										
BERT (Devlin et al., 2019)	110M	-	58.9	92.7	-	-	**92.4**	-	-	-
LISA (Strubell et al., 2018)	110M	74.8	59.8	92.0	90.7	92.2	91.4	**63.4**	38.6	**56.1**
SGNet (Zhang et al., 2020)	133M	74.8	59.2	93.1	90.9	92.6	91.7	60.9	40.7	55.4
BERT (Our reimplementation)	110M	74.6	58.7	93.1	91.0	92.3	91.6	60.5	40.0	54.9
+ WLA	+0.01M	75.0	59.6	92.8	91.3	92.9	92.1	60.4	**41.3**	55.3
+ SLA	+0.01M	**75.3**	**60.0**$^{\uparrow}$	**93.3**	**91.5**$^{\uparrow}$	**92.9**$^{\uparrow}$	92.2$^{\uparrow}$	61.0$^{\uparrow}$	41.3$^{\uparrow}$	55.7$^{\uparrow}$
Large-size Models										
BERT (Devlin et al., 2019)	340M	-	60.6*	93.2*	-	-	92.8	-	-	-
LISA (Strubell et al., 2018)	340M	76.2	62.2	92.7	91.3	92.6	92.0	63.4	43.2	57.9
SGNet (Zhang et al., 2020)	381M	76.6	63.3	93.6	91.5	92.8	92.1	63.1	42.5	57.5
BERT (Our reimplementation)	340M	76.9	63.9	94.0	91.7	93.1	92.4	62.7	42.6	57.3
+ WLA	+0.02M	76.6	62.7	93.9	91.5	93.1	92.3	61.9	**44.5**	57.4
+ SLA	+0.02M	**77.4**	**64.5**$^{\uparrow}$	**94.3**$^{\uparrow}$	**92.3**$^{\uparrow}$	**93.4**	**92.9**	**63.9**$^{\uparrow}$	42.3	**58.0**$^{\uparrow}$

Table 1: Results on single-sentence benchmarks. Results with "$*$" are taken from Liu et al. (2019). "↑" means statistically significant improvement over the BERT baseline with p-value < 0.05. Reported results are averaged over 5 runs. "Params" is short for the number of model parameters. "MCC" is short for the Matthews correlation coefficient.

method achieves consistent gains over BERT on single-sentence classification and sequence labeling tasks. Specifically, our model exceeds the published BERT results by 3.9% correlation coefficient on CoLA and 1.1% accuracy on SST-2. For the NER task, even though our reimplementation didn't achieve the performance (92.8 F1) reported by Devlin et al. (2019), our model still outperforms it in large-size. More importantly, the syntax-aware local attention yields state-of-the-art results with 0.7 absolute improvements on FCE.

Besides, the proposed local attention outperforms other approaches leveraging syntactic information on the average performance. Compared with BERT, the syntax-aware local attention improves performances consistently but the window-based local attention can't. This suggest that BERT can benefit from more attention over syntactically relevant words on several datasets.

However, there are still some gaps between our model and the state-of-the-art models on these datasets. We argue that our method just modifies the standard Transformer attention without changing its main architecture, but those models are trained by using more advanced pre-training methods (Sun et al., 2020), larger-scale datasets (Raffel et al., 2019), or learning framework (Li et al., 2020).

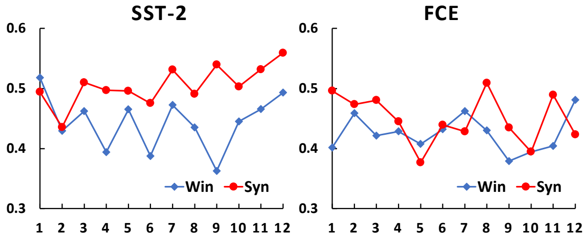

Figure 2: Gate values in different layers on SST-2 and FCE datasets. The blue polyline means that we incorporate the window-based local attention with global attention, and the red polyline corresponds to the syntax-aware local attention.

4.3 Analysis

Gated Unit in Each Layer It can be seen from Equation (6) that a larger gate value means a more important role of local attention in the attention aggregation. We analyze the gate values in different layers on SST-2 and FCE datasets. The gated unit outputs are collected from the best-trained base-size models, and are averaged over all input tokens in each layer.

As shown in Figure 2, on the SST-2 dataset, the syntax-aware local attention has higher values than the window-based local attention in most layers. Even if the sentences of the FCE dataset are ungrammatical, our attention plays a more important role in 8 of 12 layers. It indicates that our

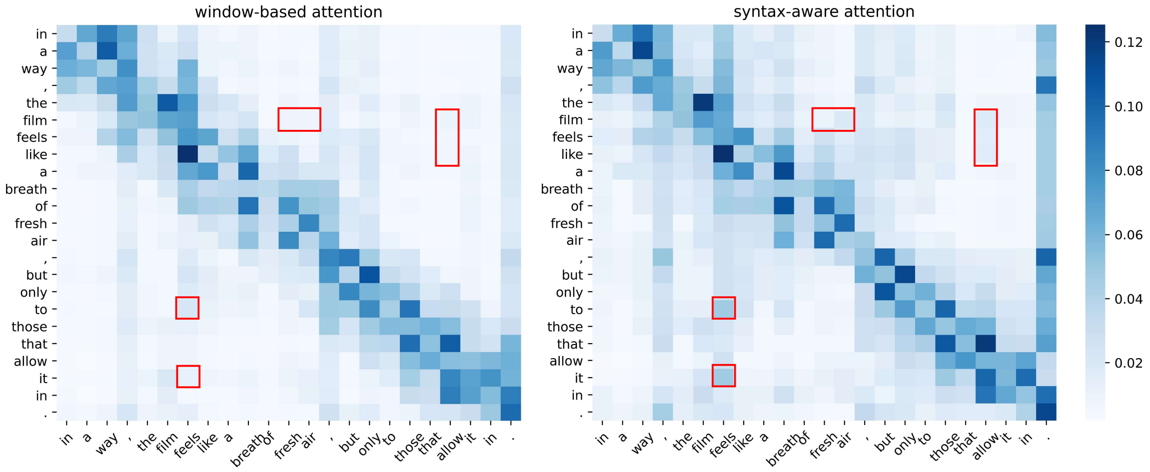

Input: In a way, the film feels like a breath of fresh air, but only to those that allow it in.

Figure 3: Visualization of attention scores averaged over all heads and all layers. This case is selected from the SST-2 dev set. The red rectangle indicates higher scores on the right side but lower scores on the left side.

Models	QNLI Acc	RTE Acc	MRPC Acc	STS PCC
BERT	**91.7**	**68.6**	87.3	89.5
+SLA	91.4	67.8	**88.5**	**89.9**

Table 2: Experimental results on sentence-pair classification datasets. All models are base-size and results are reported on their dev sets. "PCC" is short for the Pearson correlation coefficient.

local attention is more important in the attention score calculation process. Besides, Table 1 and Figure 2 illustrate that our model achieves better performances owing to more attention on syntactically relevant words.

Attention Visualization In order to compare the syntax-aware attention with the window-based attention, we plot their attention scores in Figure 3. As formulated in Equation (6), the attention scores are calculated from the aggregation of global and local attention. We mainly focus on the interactions of tokens, except for `[CLS]` and `[SEP]`. Then the attention scores are averaged over all heads and layers. This visualization validates the effectiveness of incorporating syntactic information into self-attention. As shown in Figure 3, we can see that there are many informative tokens overlooked by the window-based method (left) but captured by our method (right). For instance, the syntax-aware attention allows the tokens "*fresh air*" and "*allow*" to strongly attend to the token "*film*", but these tokens are paid less attention in the window-based attention.

Testing on Sentence-Pair Classification We attempt to evaluate our model on sentence-pair classification datasets. Given a single sentence, we can easily apply dependency parsing and restrain the attention scopes inside the sentence. But for pairwise classification, one problem is how to limit the scopes between a pair of sentences. So a naive approach is adopted, that each token in a sentence can attend to all tokens in another sentence. We conduct experiments on four pairwise classification datasets from GLUE benchmark (Wang et al., 2018), which cover paraphrase, textual entailment and text similarity.

Experimental results are shown in Table 2. The syntax-aware local attention achieves better performances on MRPC and STS, but doesn't perform well on RTE and QNLI. We suspect that it is because the cross-sentence interactions are more important for textual entailment task.

5 Conclusion

This work verifies that BERT can be further promoted by incorporating syntactic knowledge to the local attention mechanism. With more focused attention over the syntactically relevant words, our model achieves better performance on various benchmarks. Additionally, the extensive experiments demonstrate the universality of our syntax-aware local attention.

References

Samuel Bell, Helen Yannakoudakis, and Marek Rei. 2019. Context is key: Grammatical error detection with contextual word representations. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 103–115, Florence, Italy. Association for Computational Linguistics.

Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing machine translation with dependency-aware self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627, Online. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, and Ting Liu. 2018. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 52–59, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on EMNLP*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2020. Differentiable window for dynamic local attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6589–6599, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Mingzhou Xu, Derek F. Wong, Baosong Yang, Yue Zhang, and Lidia S. Chao. 2019. Leveraging local and global patterns for self-attention networks.

In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3069–3075, Florence, Italy. Association for Computational Linguistics.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458, Brussels, Belgium. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. SG-Net: Syntax-guided machine reading comprehension. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

A Appendices

A.1 Training Procedure

We extend the Transformer encoder layer and lift weights from BERT to our model. Following Devlin et al. (2019), we apply the fine-tuning procedure for various NLP tasks. For classification tasks, the final output of the first token `[CLS]` is taken as the representation of the input. The probability that the input sentence X is labeled as class c is predicted by a linear transformation with softmax:

$$P(c\,|X) = \text{softmax}(\mathbf{W}_c h_{\texttt{[CLS]}} + b_c) \tag{7}$$

where $h_{\texttt{[CLS]}}$ is the representation of the token `[CLS]`, $\mathbf{W}_c$ and b_c are task-specific parameters. For labeling tasks, we apply the BIO annotation (Ratinov and Roth, 2009) to label outputs and compute the probability that token x_i belongs to class c as:

$$P(c\,|x_i) = \text{softmax}(\mathbf{W}_t h_i + b_t) \tag{8}$$

where h_i is the representation of the token x_i, $\mathbf{W}_t$ and b_t are task-specific parameters. Finally, the training objective for all tasks is to minimize the cross-entropy loss.

A.2 Implementation Details

We apply the whitespace tokenization to the input sentence, and obtain the dependency tree using the Spacy parser[5]. However, the BERT inputs are tokenized by WordPiece tokenizer, which means one word may be split into several sub-words. To address this issue, for each word in the dependency tree, the sub-words split by WordPiece tokenizer share the same masking value in the calculation of syntax-aware local attention.

An important detail is that BERT represents the input by adding a `[CLS]` token at the beginning as the special classification embedding and separating sentences with a `[SEP]` token. Clark et al. (2019) find that these special tokens are attached with a substantial amount of BERT's attention. Thus, the `[CLS]` and `[SEP]` tokens are guaranteed to be present and are never masked out in our local attention.

We use the uncased version of BERT for CoLA and SST-2, and the cased version for CoNLL-2003 and FCE. During the training, we empirically select the threshold m from $\{3,4\}$. The maximum sequence length is set to 128 for all tasks. We use Adam (Kingma and Ba, 2015) as our optimizer, and perform grid search over the sets of the learning rate as $\{$2e-5, 3e-5$\}$ and the number of epochs as $\{3,5,10\}$ for most tasks. In particular, we use smaller learning rates $\{$5e-6, 1e-5, 2e-5$\}$ and train more epochs $\{30, 60\}$ on CoNLL-2003, but the average F1 of the best 5 runs still hasn't reached the results reported by Devlin et al. (2019). The batch size is fixed to 32 to reduce the search space, and we evaluate models every 500 training steps for all datasets. Furthermore, we experiment with the window-based attention on BERT, which allows each token to pay more attention to the neighboring tokens within a window size $2k+1$. We vary the k within $\{3,4,5\}$, and also incorporate the attention scores with global attention scores.

A.3 Testing on Chinese Benchmarks

The ChnSentiCorp dataset is used for sentiment classification task. We treat the ChnSentiCorp as single-sentence datasets although there are some examples including multiple sentences. The MSRA NER and CGED datasets are selected for named entity recognition and grammatical error detection in Chinese. The accuracy (Acc) is used as the metric of ChnSentiCorp, the precision, recall and F_1 are used as metrics of MSRA NER and CGED. In particular, for a fair comparison with the results of iFLYTEK's single model (Fu et al., 2018), we construct the CGED test set from CGED 2016 and 2017 test sets. Then we report detection-level results computed by the official evaluation tool.

Table 3 shows the main results on Chinese datasets. All results are reported on their test set. The proposed syntax-aware local attention outperforms the window-based attention and the basic BERT on all evaluated datasets. We attain 95.7 accuracy on ChnSentiCorp and 94.9 F1 on MSRA NER. Besides, BERT+SLA outperforms the state-of-the-art with a large margin on CGED.

---

[5] https://spacy.io/

Models	**ChnSentiCorp**	**MSRA NER**			**CGED**		
	Acc	P	R	F_1	P	R	F_1
State-of-the-art Models							
ERNIE 2.0 (Sun et al., 2020)	95.8	-	-	95.0	-	-	-
BERT-MRC (Li et al., 2020)	-	96.2	95.1	95.7	-	-	-
ePMI Matcher (Fu et al., 2018)	-	-	-	-	83.2	61.0	70.4
Base-size Models							
BERT (Our reimplementation)	94.7	95.0	94.6	94.8	79.9	75.2	77.5
+ WLA	95.1	**95.1**	94.2	94.6	79.9	73.5	76.6
+ SLA	**95.7**	94.9	**95.0**	**94.9**	**81.0**	**76.6**	**78.7**

Table 3: Experimental results on Chinese single-sentence benchmarks. We only show the results of base-size models because Google has not released the large-size model. Reported results are averaged over 5 runs.

A Dialogue-based Information Extraction System for Medical Insurance Assessment

Shuang Peng[*] **Mengdi Zhou**[*] **Minghui Yang** **Haitao Mi**
Shaosheng Cao **Zujie Wen** **Teng Xu** **Hongbin Wang** **Lei Liu**
Ant Group
Hangzhou, China
`{jianfeng.ps,jacquelyn.zmd,minghui.ymh,haitao.mi,shaosheng.css,`
`zujie.wzj,harvey.xt,hongbin.whb,leihu.ll}@antgroup.com`

Abstract

In the Chinese medical insurance industry, the assessor's role is essential and requires significant efforts to converse with the claimant. This is a highly professional job that involves many parts, such as identifying personal information, collecting related evidence, and making a final insurance report. Due to the coronavirus (COVID-19) pandemic, the previous offline insurance assessment has to be conducted online. However, for the junior assessor often lacking practical experience, it is not easy to quickly handle such a complex online procedure, yet this is important as the insurance company needs to decide how much compensation the claimant should receive based on the assessor's feedback. In order to promote assessors' work efficiency and speed up the overall procedure, in this paper, we propose a dialogue-based information extraction system that integrates advanced NLP technologies for medical insurance assessment. With the assistance of our system, the average time cost of the procedure is reduced from 55 minutes to 35 minutes, and the total human resources cost is saved 30% compared with the previous offline procedure. Until now, the system has already served thousands of online claim cases.

1 Introduction

In the Chinese medical insurance industry, the assessor's role is essential in handling protection claims. The insurance assessor may work for the insurance company or be a third-party administrator assessor. The claimant approaches the insurance assessor to evaluate the disease that has affected them. For example, if the claimant has lung cancer, the insurance assessor would check the medical cost and decide if they are claimable. Once the assessor has finished collecting the evidence and related claimant information (i.e., disease history, medication experience, in-hospitalized information, etc.), a final detailed case report will be made and sent to the insurance company.

The traditional insurance assessment procedure is carried out offline where the assessor conducts an onsite interview with the claimant. Due to the coronavirus (COVID-19) pandemic, the previous offline insurance assessment has to be interrupted temporarily, and thus many claimants can not get their insurance compensation in time. Some of them, meanwhile, may also suffer job loss[1]. To help these claimants, many insurance companies shift the offline procedure to the online platform in which insurance assessors can communicate with claimants via video connection. In this way, claimants can submit their applications and get compensations entirely online. On the other hand, the online assessment also helps the insurance company save expenses as the assessor can quickly approve qualified applications or screen out applications that do not conform to the claims on the online platform.

However, limited by the complexity of the insurance assessment procedure and the instability of the online environment, simply providing a video connection for insurance assessors and claimants is still insufficient. We summarize three primary challenges for the current online procedure: (1) Instability. The online video connection may sometimes be unstable, and the assessor may not distinguish the claimant's words. (2) Complexity. The insurance assessment procedure is very complex, and not all assessors are sufficiently experienced. The junior assessors, in particular, need much assistance to complete their work expertly, and even some senior assessors still have room for improvement. (3) Time-consuming. Recording the key information and writing the insurance report is te-

---

[*] Equal contributions.

[1] According to the statistics result from the United Nations, COVID-19 could cause the equivalent of 400 million job losses in the second quarter of 2020.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 654–663
August 1–6, 2021.

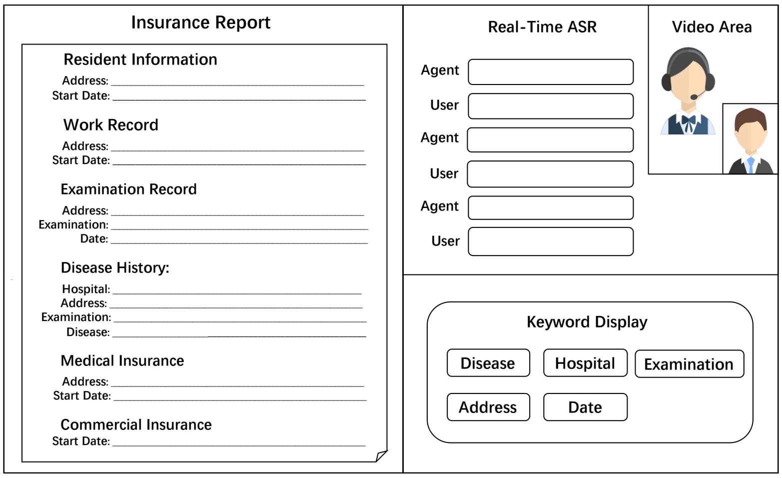

Figure 1: The illustration of the front end of Intelligent Insurance Assessment System.

dious for most assessors, and junior assessors may even forget some important information during the inquiry.

To this end, we propose an **I**ntelligent **I**nsurance **A**ssessment **S**ystem, called **IIAS** (shown in Figure 1) for medical insurance assessment. The system aims at promoting the work efficiency of the insurance assessor through dialogue-based information extraction. For these purposes, We use recent NLP technologies, such as streaming automatic speech recognition (ASR) (Moritz et al., 2020; Mani et al., 2020), large scale pre-trained language models (Devlin et al., 2019; Cui et al., 2019), sentence similarity learning (Chen et al., 2017; Peng et al., 2020), named entity recognition/linking (NER/EL) (Le and Titov, 2018; Devlin et al., 2019) and dialogue state tracking (DST) (Ouyang et al., 2020; Zhang et al., 2020b), to ensure high performance while requiring only a small amount of annotated data.

Our IIAS alleviates the cognitive workload of assessors in several steps: (1) Our streaming ASR transforms speech signal into the conversation text in real-time, and our sentence similarity learning method labels corresponding topics for each conversation on-the-fly. (2) Our NER component then extracts raw entities from the real-time conversation text, and the EL part links the raw entities into the unified insurance knowledge base (KB) for getting all possible keywords. (3) Our DST method, including question and negation identification modules, tracks the state of dialogue context for filtering irrelevant keywords.

To the best of our knowledge, this is the first work to propose an intelligent system for insurance assessment. The main contributions of our work are concluded as follows.

- We propose a dialogue-based information extraction system that shifts the previous insurance assessment procedure online and provides necessary intelligent assistance for the assessor.
- We apply recently advanced NLP technologies to address the problem of online insurance assessment. Our methods significantly improve the assessor's work efficiency, where the average time cost of the insurance assessment procedure is reduced from 55 minutes to 35 minutes.
- Our system has already been deployed in the real world. Until now, it has served thousands of online insurance claim cases, which delivers potential value to the insurance industry.

2 System Framework and Technique Details

Figure 2 shows the architecture of IIAS with key components such as streaming ASR, dialogue segmentation, keyword extraction (NER and EL), keyword filtering (question and negation identification), and downstream applications (keyword display and report filling).

As previously mentioned, recording the information during the assessment is tedious for most insurance assessors. In order to increase the efficiency of information collection and reduce the assessor's cognitive workload, IIAS provides a keyword-based feature that first displays keywords

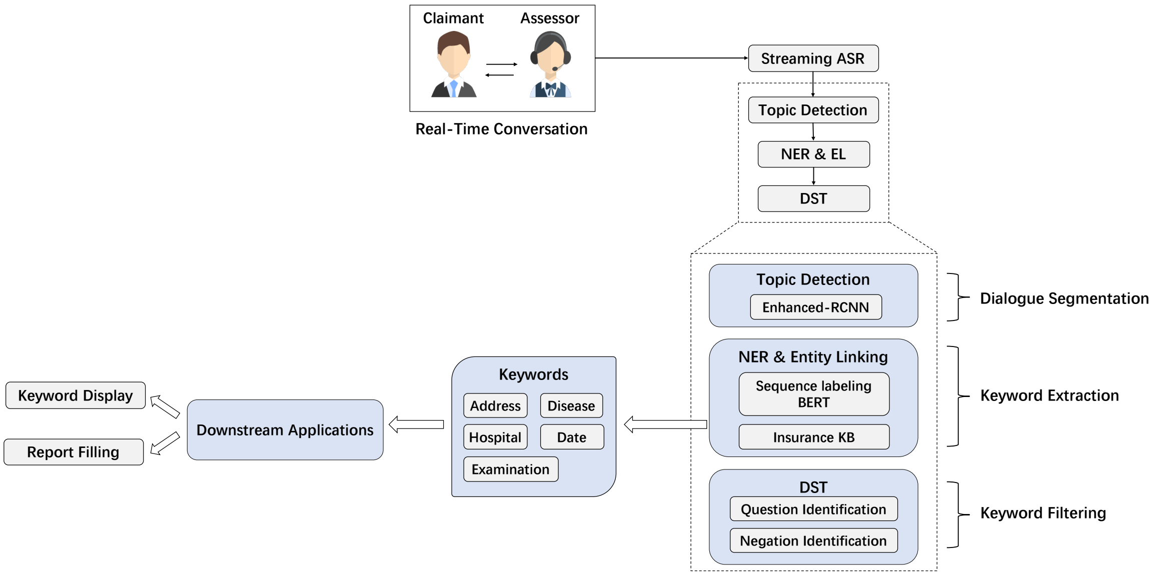

Figure 2: The Architecture of Intelligent Insurance Assessment System.

on the system dashboard and then intelligently suggests the related content when assessors are filling in the insurance report.

For better understanding, we discuss the technical details for each key component in the following subsections. Besides, we also present the partial experimental result corresponded to each component.

2.1 Dialogue Segmentation

Dialogue segmentation aims at detecting the topic of the real-time conversation text. As the insurance assessment procedure is conducted according to the content that needs to be filled in the insurance report, different dialogue between the assessor and claimant is focused on different topics. These topics can be seen as different questions that the claimant needs to answer. For example, the assessor may start with the topic about the claimant's work address, and then inquire about when the disease starts and where the claimant receives the treatment, and finally inquire about the doctor's advice and in-hospitalized information. Here we use the sentence similarity learning method to map the assessor utterance into the corresponding topic.

First, we order the one-to-many mapping relationship between the topic and standard questions that most assessors inquire in the insurance assessment procedure. We select the representative utterance from the assessor as the standard question. The standard question is a set of questions that represent different topics across all assessors. We select the standard question by running HDBSCAN clustering algorithm (McInnes et al., 2017) followed with a human adjustment. Since the ways of inquiry among assessors are different, and the ASR results are sometimes affected by the outside speech signals, we use the Enhanced-RCNN model (Peng et al., 2020) to calculate the similarity score between the utterance and selected standard questions, and get the topic of the utterance if the score is greater than the threshold (the default value is 0.5) or follow the topic of the previous utterance if the score is less than the threshold.

After the dialogue segmentation, the whole dialogue is segmented into many parts that belong to different topics so that the keywords extracted in the following process can be linked with these topics in real time. We evaluate the performance of dialogue segmentation by 200 online cases with the human annotation. The accuracy of segmentation reaches 90%.

2.2 Keyword Extraction

In the insurance assessment scenario, the keyword contains five types[2], and their values are not within a limit number. Therefore, any string fragment of assessor or claimant utterance may become a keyword. Traditional slot filling methods with classification models in the task-oriented dialogue system cannot predict unseen slot values in a pre-defined value list and are not appropriate to our scenario. To solve the problem, we convert the keyword extraction as a sequence labeling task and use the

---

[2] Address, Hospital, Disease, Date, and Examination

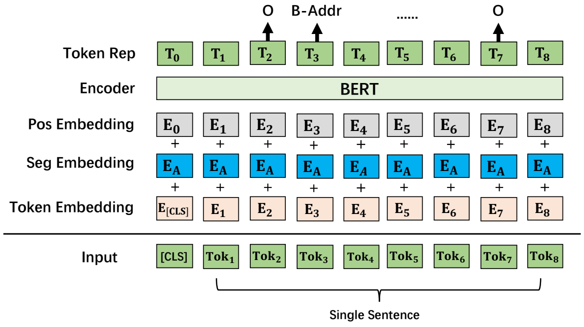

Figure 3: The sequence labeling BERT model for NER.

Entity Type	Addr.	Hos.	Dis.	Date.	Exam.
F1-Score (%)	78.2	78.3	83.1	85.1	85.4

Table 1: Test performance of the NER/EL model on five entity types.

BERT-based NER model (Devlin et al., 2019).

We first pre-annotate the keyword type in the utterance as `Addr`, `Hos`, `Dis`, `Date`, `Exam` or `Other` (non-named entity) and follow the official tutorial in NER[3]. The visual representation of BERT-based NER model is shown in Figure 3. For fine-tuning, we feed the final hidden representation for each token into a classification layer over the NER label set, and the predictions are non-autoregressive and no CRF (Devlin et al., 2019). After the NER process, we get the raw entities extracted from the utterances. As the word accuracy of ASR is about 85%, some of the derived entities may be misrecognized (correct entity type but with a wrong form). In order to alleviate the negative effect of the ASR module, we use the EL model (Le and Titov, 2018) to recognize and disambiguate the raw entities to the insurance KB that contains the normalized address, hospital and disease information for getting all possible keywords.

To measure the NER/EL model's performance separately, we take the F1-Score as the evaluation metric. The experimental results on the human-annotated testing set are shown in Table 1. The results show that the NER/EL models achieve good performance on all five entity types as the average F1-score has reached 82%.

2.3 Keyword Filtering

During the insurance assessment procedure, the assessor always asks the claimant many detailed questions. As a result, a part of entities extracted by the previous keyword extraction process may come from the assessor's question text and actually do not need to be displayed or suggested if the claimant's following response has negative semantics. To filter the irrelevant keywords extracted by the previous process and increase the accuracy of keyword display and report filling, we design a DST method for keyword filtration. Like the application in the task-oriented dialogue system, the DST method tracks the keyword's state (addition or deletion) and contains two major modules: question identification and negation identification. The assessor's conversion text is first fed to the question identification module. Suppose the text is identified to be a question. In that case, the extracted keyword from it will be reserved temporarily and only displayed (or recommended) once they are confirmed by the claimant (decided by the negation identification module).

2.3.1 Question Identification

Question identification determines whether the current utterance belongs to part of the insurance assessor's inquiry. We convert question identification as the binary classification task and utilize the text classification model to solve it.

We utilize the LSTM (Hochreiter and Schmidhuber, 1997), one of the most popular text classification models, as the base model. In this case, the input $x = \{x_1, x_2, ..., x_T\}$ is the text transformed by the ASR module where x_t is the word after segmentation. The output is a label y indicating whether the text is a question or not.

Based on the observation, the insurance assessment's topic distribution is diverse, and the accuracy of question identification is highly relevant to the concrete topic (i.e., resident/work address, medication experience, or disease history). Therefore, training a single model for covering all the topics is actually not the best solution in our scenario, and it is necessary to train different identification models for different topics. However, the total amount of manually labeled data is not large enough to train multiple classifiers for every topic. To alleviate this issue, we adopt the adversarial multi-task training method (Liu et al., 2017; Yang et al., 2020). Compared with the multi-task training method, the adversarial-based training method improves the question identification model's performance for general purposes and a particular topic with only a few labeled data.

Experimental results on different training meth-

---

[3] https://github.com/google-research/bert

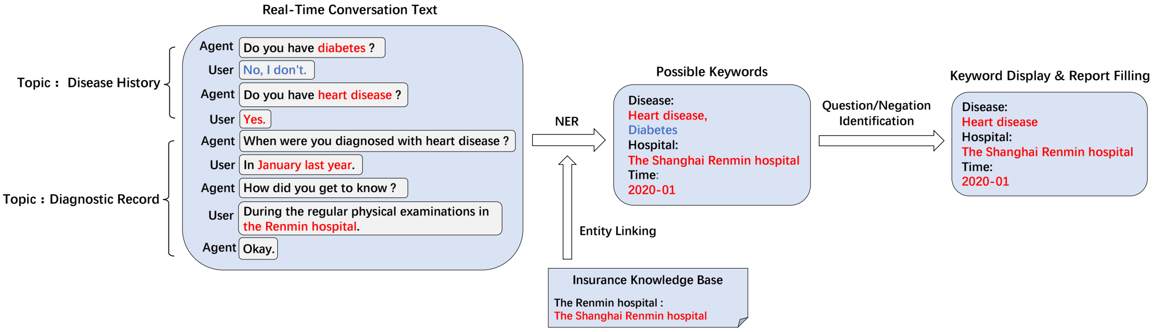

Figure 4: An example of IIAS workflow. In this example, the real-time conversation text is segmented into two topics, and keywords extracted from the utterances are first filtered, and then displayed on the system dashboard, and finally suggested when the assessor fills in the insurance report.

Model	**Accuracy (%)**
Single Model	61.2
Multi-task Model	75.1
Adversarial Multi-task Model	**78.4**

Table 2: Test Performance on Question Identification.

ods are presented in Table 2. The results show that the adversarial multi-task model achieves the best accuracy among three compared models, which indicates that the adversarial learning method is more appropriate to the insurance scenario that the topic distribution of question is unbalanced.

2.3.2 Negation Identification

After the question identification module, the following claimant's utterance is feed to the negation identification module. The keywords extracted from the question are only displayed on the system dashboard if the claimant confirms. Here we use the BERT-based binary-classification model (Devlin et al., 2019) for negation identification. The input is the current claimant utterance with the previous two rounds of utterances from the assessor and claimant (we use `[SEP]` to separate them). The output is the label indicating whether the current utterance has negative semantics or not (the final layer is a two-class softmax layer). From the performance on the human-annotated testing set, the negation identification module's accuracy is 92%.

The ablation study on the DST method's contribution (question and negation identification modules) is present in the experiment section.

2.4 Applications

Based on the previous three steps, the system gets filtered keywords extracted from the real-time conversion text, and these keywords are classified into different topics simultaneously.

To help the assessor better do their job, we identify two types of generic applications for derived keywords: keyword display and report filling. The keyword display aims at displaying information extracted from the real-time conversation text on the system dashboard. Furthermore, the report filling aims at recommending related contents when the assessor fills in the specific part of the insurance report. For ensuring a good user experience, we sort extracted keywords by the reversed order of utterances in the dialogue and limit the maximal number of recommended contents as 5. We show an example to illustrate the IIAS workflow in Figure 4. The figure shows how IIAS extracts the information from the real-time conversation text and uses them for the downstream applications.

For the insurance assessor, keyword display and report filling significantly alleviate the workload of remembering all the claimant information during the inquiry procedure. To further evaluate IIAS's overall contribution to the assessor's work, we conduct different experiments in the experiment section.

3 Experiments

3.1 Insurance Assessment Dataset

We manually construct an insurance assessment dataset from the real-world environment and use it for evaluating the performance of IIAS. The dataset consists of 200 online cases, and each case contains two parts: a multi-turn dialogue and a corresponding insurance report that the assessor needs to fin-

Items	Train	Test
Num of dialogs	5000	200
Average Rounds Per Dialog	497.6	501.2
Average Keywords Per Dialog	25.3	27.9

Table 3: Statistics of Insurance Assessment Dataset.

ish. To ensure the high quality of the dataset, we employ five real insurance assessors to fill in the insurance report by selecting a string fragment of the utterance from the given multi-turn dialogue. Here we screen out forms in the insurance report that cannot be filled based on the given multi-turn dialogue. Table 3 shows the detailed statistics of the insurance assessment dataset.

In order to protect data privacy, the usage of the claimant's private data must be authorized. We have obtained explicit permissions from the claimant to use their personal private data before the insurance assessment. If a claimant disapproves of the authorization, we will not use her/his data. For the authorized user privacy data, we have a series of strict processes to ensure the data remain confidential.

3.2 Experimental Setup

3.2.1 Method in Comparison

As far as we know, there is no similar research work in the insurance industry. To examine the effectiveness of IIAS, we compare it with the retrieval-based system on the report filling performance. Different from IIAS that is mainly based on the real-time ASR and speaker recognition, the retrieval system suggests retrieved results from the pre-constructed KB based on the conversation text. For example, the system will recommend *top-k* relevant disease nouns in the KB when the assessors fill in the form about what disease the claimant is diagnosed with. To be aligned with IIAS, we limit the retrieved number k in the retrieval system as 5.

The major limitations of the retrieval system are that the content that needs to be filled may only exist in the conversation text but not appear in the pre-constructed KB.

3.2.2 Evaluation Metrics

The evaluation process is divided into offline and online parts.

Offline Evaluation: We choose the recall as the metric to evaluate the system performance of report filling. The recall measures the ratio of suggested correct content to all ground-truth content that needs to be filled.

Online Evaluation: We choose the time-saving and efficiency-improving as metrics because our work's primary purpose is to alleviate assessors' workload and reduce the insurance assessment procedure's time cost.

3.3 Evaluation Results and Discussions

The experimental results[4] on the report filling performance between IIAS and the retrieval system are shown in Table 4 and Table 5.

We summarize our observations as follows:

(1) Our IIAS is generally better than the retrieval system on the record filling's performance for most topics. This shows that using the method based on speaker recognition is more effective with less dependence on the pre-constructed KB.

(2) For entities of `Addr` type, the retrieval system achieves comparative performance than IIAS. This indicates that the address entities are relatively fixed and easy to be included in the pre-constructed KB. For this type of entity, the retrieval system is more appropriate.

(3) For entities of `Exam` and `Dis` types, our system achieves much better performance than the retrieval system. This shows that for those entity types that are diverse and hard to be included by the pre-constructed KB, using the NER/EL methods to extract keywords from the utterances is more appropriate.

3.4 Ablation Study

We perform the ablation study to validate the contribution of the DST method (question and negation identification). Since the DST method filters out irrelevant keywords extracted by the NER/EL modules, we choose the *hospital* and *disease* types that frequently appear in the assessor's utterances for the experiment.

The experimental results are presented in Table 6. Here we evaluate the precision of the keyword extraction. The results show that with the addition of the DST method, the keyword extraction's precision is significantly promoted. This indicates that the DST method effectively filters the error keywords, which is beneficial for improving system performance.

---

[4] To make a fair comparison, we do not compare the recall ratio on the `Date` type because the value of `Date` cannot be included by the pre-constructed KB in the retrieval system.

Topic	**Resident Information**		**Work Record**		**Diagnostic Record**		
Entity Type	**Addr.**	**Date.**	**Addr.**	**Date.**	**Date.**	**Hos.**	**Exam.**
Retrieval System	85.6	-	84.2	-	-	67.9	65.7
IIAS	83.5	43.3	**85.7**	55.4	52.1	**74.0**	**68.8**

Table 4: The Recall@5(%) of report filling on different topics (resident information, work record and diagnostic record).

Topic	**Disease History**				**Medical Insurance**		**Commercial Insurance**
Entity Type	**Date.**	**Hos.**	**Exam.**	**Dis.**	**Addr.**	**Date.**	**Date.**
Retrieval System	-	68.3	71.5	25.7	92.3	-	-
IIAS	57.9	**74.8**	**76.9**	**28.8**	91.8	54.9	46.7

Table 5: The Recall@5(%) of report filling on different topics (disease history, medical insurance and commercial insurance).

Entity Type	**Hospital**	**Disease**
w/o DST	71.79	39.20
Complete System	**77.31**	**53.24**

Table 6: Evaluation results of system ablation on the Precision(%) of keyword extraction for `Dis` and `Hos` types.

3.5 Online Performance

From the statistical result for three months after the system deployment, the average time cost of the insurance assessment procedure is reduced from 55 minutes to 35 minutes, and the overall human resources cost is saved 30% compared with previous offline insurance assessment.

Moreover, with the assistance of our IIAS, the insurance company provides more job opportunities for junior assessors. This indicates that IIAS has effectively lowered the bar of insurance assessment jobs.

4 Deployment Details

Our proposed approach consists of different NLP components that are trained individually. We deploy them on two GPUs (Tesla P100 16G) and provide a unified HTTP interface for the back-end system.

To support multiple users, the computation cost needs to be well distributed across different GPUs. For each GPU, we set up two independent processes. A load-balancing module is then employed to distribute the users' requests to different processes based on the working loads. Based on the stress test, the online service's average response time is less than 150ms and can support a peak QPS of 35.

5 Discussions on Technique Limitations

In this section, we discuss the limitations of techniques in IIAS. Based on the observation of system results and feedback from the insurance assessors, the limitations are from three parts:

(1) **ASR Results**: The performance of two applications (keywords display and report filling) heavily relies on the ASR module, but the ASR errors are the major problem of our system. The ASR error may result in correct entity type but with a wrong form, which indicates that optimizing the ASR module should be a top priority for future research.

(2) **Reasoning Problem**: The current system cannot solve the problem that requires complex reasoning, such as date recognition (e.g., last Sunday or three days ago.). However, many date recognitions during the insurance assessment procedure require reasoning. As a result, we plan to include reasoning ability into the system in the future.

(3) **Accumulative Errors**: The current system is based on a multi-model approach where different NLP components such as topic detection, NER/EL, and question/negation identifications are trained individually and combined. This type of pipeline approach is easy to lead to accumulative errors. We have tried to replace the current approach with the end-to-end summarization method based on text-generation. However, text-generation models always require many training data and their predictions are sometimes uncontrollable (the inference time is also slow if we use the pre-trained model

like GPT-2 (Radford et al., 2019)). Although now the end-to-end method cannot be applied in the real-world setting (especially in our insurance scenario that requires controllability), it has some advantages and is still part of our plan.

For these limitations, we are working on a new solution to improve the current system.

6 Related Work

With the advances in NLP, agent-assist systems have been used in various domains, including technical support, reservation systems, and banking applications (Fadnis et al., 2020). In this section, we briefly introduce some related technologies used in IIAS.

6.1 Sentence Similarity Learning

Sentence similarity learning is a fundamental and important NLP task which may be greatly enhanced by modeling the underlying semantic representations of compared sentences. In particular, a model should not be susceptible to variations of wording or syntax used to express the same idea. Moreover, a good model should also have the capacity to learn sentence similarity regardless of the length of the text and also needs to be efficient when applied to real-world applications (Chen et al., 2017; Peng et al., 2020).

In the scene of insurance assessment, both efficiency and accuracy are of equal importance. Therefore we use the recent proposed Enhanced-RCNN model (Peng et al., 2020) that achieves good performance on Chinese paraphrase identification datasets for learning sentence similarity.

6.2 Named Entity Recognition and Linking

Named entity recognition is the NLP task of tagging entities in the text with the corresponding type and recent large-scale language model pretraining methods such as ELMo (Peters et al., 2018), and BERT (Devlin et al., 2019) further enhanced the performance of NER, yielding state-of-the-art performances (Sutton et al., 2007; Li et al., 2019).

Entity linking is the task of recognizing and disambiguating named entities to a knowledge base (Hoffart et al., 2011; Le and Titov, 2018). EL can be split into two classes of approaches:

- **End-to-End:** processing a piece of text to extract the entities and then disambiguate these extracted entities to the correct entry in a given knowledge base.
- **Disambiguation-Only:** directly takes gold standard named entities as input and only disambiguates them to the correct entry in a given knowledge base.

6.3 Dialogue State Tracking

Dialogue state tracking is an important component in task-oriented dialogue systems to identify users' goals and requests as a dialogue proceeds (Zhu et al., 2019). The traditional DST system assumes that each slot's candidate values are within a limit number. However, this assumption does not apply to slots with an unlimited number of values in advance. It is more difficult for zero-shot domains like the scene of insurance assessment to predefine the slot values in advance (Ma et al., 2019; Ouyang et al., 2020).

Some recent researches are on converting fixed slot values into the substring of the dialogue context (Xu and Hu, 2018; Zhang et al., 2020a; Gao et al., 2019). In this way, many researchers have proposed many neural networks to complete DST tasks in a reasonable way (Perez and Liu, 2017; Zhang et al., 2020b). Inspired by the recent progress, we also use the approach that treats the dialogue context as the source of slot values.

7 Conclusions and Future Work

In this paper, we presented a dialogue-based information extraction system that helps the insurance assessor better perform their job. The system combines different NLP technologies like sentence similarity learning, NER/EL, and DST methods in a novel way. While there has been considerable work done for each of the independent modules/models, this systematic way of combining them enables the platform to quickly deliver what may be of significant value to the insurance industry.

In the future, we would like to try to replace the current pipeline method with the end2end method based on text-generation when the accumulated annotated data is enough.

8 Ethical Considerations

Below we present the ethical considerations in terms of data authorization, privacy, and trust in real-world deployments.

- We have obtained explicit permissions from the claimant to use their personal private information data, including text and audio provided in the insurance assessment procedure for improving the system service.

- The personal private information from the claimant is never used or stored for the commercial purpose.

Acknowledgements

The authors would like to thank Gan Li, Jianzeng Liang, Chaojin Zhou, Weiwei Fan, Li Qiao, and other members of Ant Group for helpful discussions and comments. We would also like to thank reviewers for their valuable comments.

References

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kshitij Fadnis, Nathaniel Mills, Jatin Ganhotra, Haggai Roitman, Gaurav Pandey, Doron Cohen, Yosi Mass, Shai Erera, Chulaka Gunasekara, Danish Contractor, Siva Patel, Q. Vera Liao, Sachindra Joshi, Luis Lastras, and David Konopnicki. 2020. Agent assist through conversation analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 151–157, Online. Association for Computational Linguistics.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273, Stockholm, Sweden. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2019. A unified mrc framework for named entity recognition. *arXiv preprint arXiv:1910.11476*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. 2019. An end-to-end dialogue state tracking system with machine reading comprehension and wide & deep classification. *arXiv preprint arXiv:1912.09297*.

Anirudh Mani, Shruti Palaskar, Nimshi Venkat Meripo, Sandeep Konam, and Florian Metze. 2020. Asr error correction and domain adaptation using machine translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6344–6348. IEEE.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205.

Niko Moritz, Takaaki Hori, and Jonathan Le. 2020. Streaming automatic speech recognition with the transformer model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6074–6078. IEEE.

Yawen Ouyang, Moxin Chen, Xinyu Dai, Yinggong Zhao, Shujian Huang, and Jiajun Chen. 2020. Dialogue state tracking with explicit slot connection modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 34–40, Online. Association for Computational Linguistics.

Shuang Peng, Hengbin Cui, Niantao Xie, Sujian Li, Jiaxing Zhang, and Xiaolong Li. 2020. Enhanced-rcnn: An efficient method for learning sentence similarity. In *Proceedings of The Web Conference 2020*, pages 2500–2506.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 305–314, Valencia, Spain. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1457, Melbourne, Australia. Association for Computational Linguistics.

Minghui Yang, Hengbin Cui, Shaosheng Cao, Yafang Wang, and Xiaolong Li. 2020. Servicegroup: A human-machine cooperation solution for group chat customer service. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2189–2192.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, Philip Yu, Richard Socher, and Caiming Xiong. 2020a. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 154–167, Barcelona, Spain (Online). Association for Computational Linguistics.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020b. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, pages 1–17.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2019. SIM: A slot-independent neural model for dialogue state tracking. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 40–45, Stockholm, Sweden. Association for Computational Linguistics.

Prediction or Comparison: Toward Interpretable Qualitative Reasoning

Mucheng Ren, Heyan Huang* and Yang Gao
School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
Beijing Engineering Research Center of High Volume Language Information
Processing and Cloud Computing Applications, Beijing, China
{renm,hhy63,gyang}@bit.edu.cn

Abstract

Qualitative relationships illustrate how changing one property (e.g., moving velocity) affects another (e.g., kinetic energy) and constitutes a considerable portion of textual knowledge. Current approaches use either semantic parsers to transform natural language inputs into logical expressions or a "black-box" model to solve them in one step. The former has a limited application range, while the latter lacks interpretability. In this work, we categorize qualitative reasoning tasks into two types: prediction and comparison. In particular, we adopt neural network modules trained in an end-to-end manner to simulate the two reasoning processes. Experiments on two qualitative reasoning question answering datasets, QuaRTz and QuaRel, show our methods' effectiveness and generalization capability, and the intermediate outputs provided by the modules make the reasoning process interpretable.

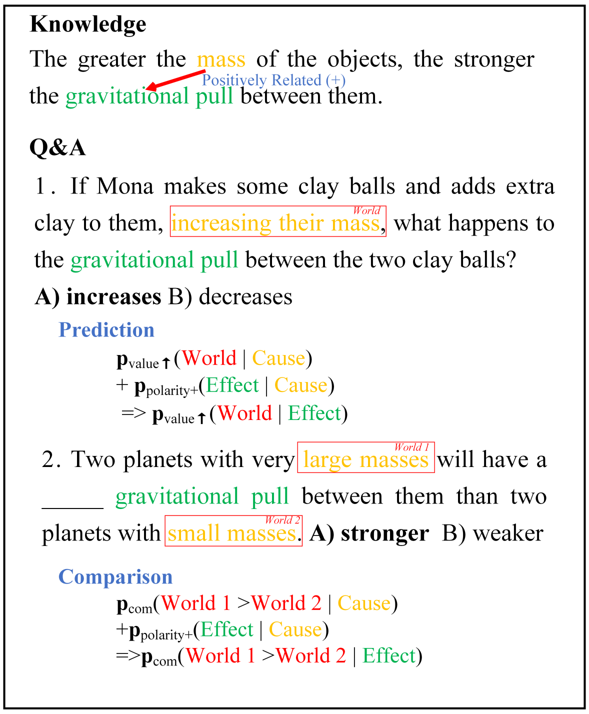

Knowledge

The greater the mass of the objects, the stronger the gravitational pull between them.

Q&A

1. If Mona makes some clay balls and adds extra clay to them, increasing their mass, what happens to the gravitational pull between the two clay balls?
A) increases B) decreases

Prediction

$\mathbf{p}_{\text{value}\uparrow}$(World | Cause)
+ $\mathbf{p}_{\text{polarity+}}$(Effect | Cause)
=> $\mathbf{p}_{\text{value}\uparrow}$(World | Effect)

2. Two planets with very large masses will have a _____ gravitational pull between them than two planets with small masses. **A) stronger** B) weaker

Comparison

$\mathbf{p}_{\text{com}}$(World 1 >World 2 | Cause)
+$\mathbf{p}_{\text{polarity+}}$(Effect | Cause)
=>$\mathbf{p}_{\text{com}}$(World 1 >World 2 | Effect)

Figure 1: Examples from QuaRTz dataset.

1 Introduction

Qualitative relationships abound in our world, especially in science, economics, and medicine. Since Question Answering (QA) has been significantly developed in recent years, various challenging datasets have been proposed (Lai et al., 2017; Rajpurkar et al., 2018; Yang et al., 2018; Choi et al., 2018; Dua et al., 2019), and one often encounters the context and questions about qualitative relationships. Figure 1 shows an example that requires reasoning about the qualitative relationship between the mass and the gravitational pull, where the knowledge sentence states that the mass is positively correlated with the gravitational pull, and the questions describe different scenarios that test the flexible application of the knowledge.

Therefore, understanding the qualitative relationships behind the context and applying the qualitative textual knowledge for reasoning is essential. However, the most current datasets on qualitative reasoning are much smaller than standard Question Answering datasets, making the task challenging and receiving limited attention. At present, in the two mainstream qualitative relationship question datasets, QuaRel (Tafjord et al., 2019a) and QuaRTz (Tafjord et al., 2019b), the existing methods can be divided into two categories, symbolic reasoning based on a semantic parser (Krishnamurthy et al., 2017; Tafjord et al., 2019a) and a "black-box" model based on representations (Mitra et al., 2019; Tafjord et al., 2019b; Asai and Hajishirzi, 2020; Mitra et al., 2020).

The two types of methods have their advantages and disadvantages. On one hand, symbolic reasoning provides solid explanations for the problem-solving process, but existing semantic parsers are

---

*Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 664–675
August 1–6, 2021. ©2021 Association for Computational Linguistics

trained to translate natural-language sentences into a task-specific logical representation that naturally increases the demand for additional annotated data and has limited generalization capability. On the other, approaches based on pre-trained language models that solve the task in one step achieve stunning results at the expense of limited interpretability.

To tackle the issues mentioned above, in this paper, we propose a neural network module-based approach that provides good interpretability while achieving better results. Specifically, inspired by human cognitive processes, we group the qualitative reasoning questions into two categories: **Prediction** and **Comparison**. As the example in Figure 1 shows, each category requires different reasoning chains. In **Prediction**, the question asks to directly predict the effect on an occurrence of the cause on an entity in the situation. In **Comparison**, the question asks to compare the effects on the two entities. Then we adopt neural modules to model each step in these two reasoning chains, and all modules are trained in an end-to-end manner.

The practice of first summarizing the questions and then modeling the different reasoning chains step by step has three advantages: First, it has good generalization ability. The two reasoning chains summarized cover the vast majority of qualitative reasoning problems, so there is no need to design logical expressions for specific tasks. Second, it achieves better performance, because complex qualitative reasoning tasks are broken down into simple sub-tasks, with the modules required to complete only simple sub-tasks. Third, it provides better interpretability, because each module can provide a transparent intermediate output.

Experimental results on the QuaRel (Tafjord et al., 2019a) and QuaRTz (Tafjord et al., 2019b) datasets demonstrate the effectiveness and generalization capability of our proposed approach. It surpasses the state-of-the-art model by a large margin (absolute difference ranging from 1.8% to 4.4%). Furthermore, analyses of the intermediate outputs and a human evaluation show that each module in our approach performs well on its corresponding sub-task and clarifies interpretability.

2 Related Work

Qualitative Reasoning: This type of reasoning is an indispensable part of artificial intelligence. Forbus (1984) and Weld and De Kleer (2013) propose formal models for qualitative reasoning when the research filed is emerging. However, there has been little work on reasoning with textual qualitative knowledge. Tafjord et al. (2019a) solves such tasks using semantic parsing and a symbolic solver; this type of method naturally falls short on performance when compared to neural systems. Meanwhile, some works tackle tasks using a data-driven "black-box" model based on representations computed by language models, which achieves superior performance (Tafjord et al., 2019b; Mitra et al., 2019; Asai and Hajishirzi, 2020; Mitra et al., 2020). Our work enjoys the performance improvement that a neural network brings but also provides interpretability.

Neural Network Modules: These modules were first adopted in the Visual Question Answering (VQA) domain (Johnson et al., 2017; Andreas et al., 2016; Hu et al., 2017). Due to the interpretable, modular, and inherently compositional nature of neural network modules, they were further extended to natural language (Gupta and Lewis, 2018; Jiang and Bansal, 2019; Jiang et al., 2019). Gupta et al. (2019) extend neural module networks to answer compositional questions. Ren et al. (2020) and Liu and Gardner (2020) further introduce neural network modules on one complex reasoning task ROPES (Lin et al., 2019). In this work, we explore the effectiveness of neural network modules on a qualitative reasoning task.

3 Method

Our work solves the question through a three-step process illustrated in Figure 2a:

1. The *Contextual Encoding* component encodes the concatenation of knowledge, the question, and options into contextual representations.
2. The *Reasoning* component arranges different reasoning chains according to the type of problem, where Prediction and Comparison contain multiple neural modules. A synthetic text is generated by the decision function based on the evidence collected from the reasoning chains.
3. The *Answer Prediction* component takes the concatenation of the question, options, and the generated synthetic text instead of given knowledge as the input and returns the answer option with the highest salience scores.

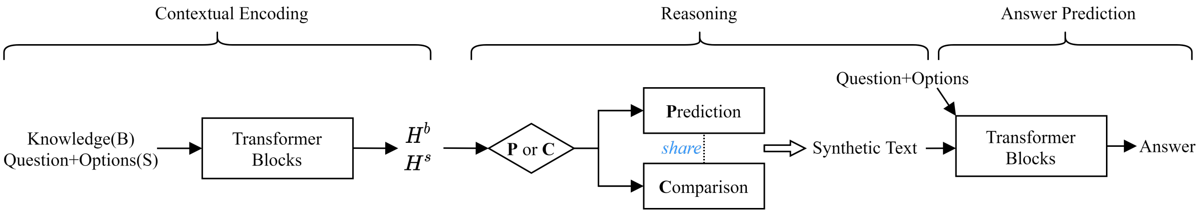

a) Overall Pipeline of our model

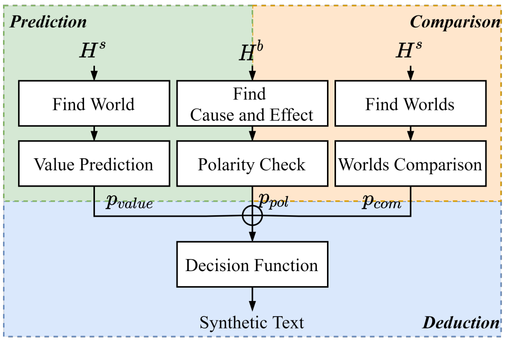

b) Reasoning Module

$p_{val}(W\|C)$	$p_{pol}(E\|C)$	***Prediction:* Synthetic Text**
↑	+	World will cause more Effect.
↑	-	World will cause less Effect.
↓	+	World will cause less Effect.
↓	-	World will cause more Effect.

$p_{com}(W_1, W_2)$	$p_{pol}(E\|C)$	***Comparison:* Synthetic Text**
>	+	World 1 will cause more Effect than World 2.
>	-	World 1 will cause less Effect than World 2.
<	+	World 1 will cause less Effect than World 2.
<	-	World 1 will cause more Effect than World 2.

c) Decision Function

Figure 2: The overall diagram of our proposed model. Given background knowledge B and statement S formed by the question and options, we first generate the contextual representations. Then we synthesize a text by collecting the evidence provided by the **Prediction** or **Comparison** module, where the detailed structure is shown in b). Last, we predict the answer with the aid of the synthetic text.

3.1 Contextual Encoding

We selected RoBERTa (Devlin et al., 2019; Liu et al., 2019) as the encoder to encode the background knowledge, the question, and two answer options together and output contextualized embeddings. In particular, we concatenated the question with two answer options as statement $S = \{s_i\}_{i=1}^m$, similar to the examples shown in Figure 1. Furthermore, we joined given background knowledge $B = \{b_j\}_{j=1}^n$ and the statement with the special tokens used in language models as $\langle s \rangle\, b_1, \ldots, b_n \,\langle /s \rangle\, \langle /s \rangle\, s_1, \ldots, s_m \,\langle /s \rangle$, which was further fed into RoBERTa containing a series of successive transformer blocks,

$$\boldsymbol{H^b}, \boldsymbol{H^s} = \text{Transformers}(B, S), \qquad (1)$$

where $\boldsymbol{H^b} \in \mathbb{R}^{n \times d}$, $\boldsymbol{H^s} \in \mathbb{R}^{m \times d}$ are contextual representations of the knowledge and the statement, respectively, and d is the dimension for the hidden states. The representations $\boldsymbol{H^b}, \boldsymbol{H^s}$ are padded into fixed length and served as the global variables in the *Reasoning* part.

3.2 Reasoning

The architecture of the reasoning component is shown in Figure 2b. In particular, we designed the different reasoning chains depending on the question type. Specifically, both reasoning chains were constructed by several end-to-end modules, and some modules were shared. Then the evidence collected from each module is summarized for the final deduction.

3.2.1 Prediction

Find Cause and Effect: To answer the Prediction-type question like the first example shown in Figure 1, we first need to understand what the knowledge describes, i.e., figure out the cause and effect properties (e.g., *mass* and *gravitational pull*). To achieve this, we applied a multilayer perceptron (MLP) over background knowledge representations $\boldsymbol{H^b}$ and then used a softmax function to normalize the projected logits and get attention scores over all knowledge tokens for the cause property and the effect property respectively,

$$\boldsymbol{p}_c^b = \text{softmax}(\text{MLP}(\boldsymbol{H^b}; \theta_c)) \in \mathbb{R}^n, \qquad (2)$$

$$\boldsymbol{p}_e^b = \text{softmax}(\text{MLP}(\boldsymbol{H^b}; \theta_e)) \in \mathbb{R}^n, \qquad (3)$$

where $\boldsymbol{p}_c^b$ and $\boldsymbol{p}_e^b$ are the attention vectors over knowledge in terms of the cause and effect properties, and θ's are trainable parameters in the MLP.

Take Figure 1 as an example: $\boldsymbol{p}_c^b$ is the attention over background tokens, whose value is much larger for *mass* than the other tokens.

Polarity Check: We further need to judge the polarity of the qualitative relationship between the cause and effect properties (e.g., *mass* and *gravitational pull* are positively related). To achieve that, we first applied cause and effect property attention vectors on the knowledge representation $\boldsymbol{H^b}$ to obtain the weighted ones. Then we took the average of the weighted representations and concatenated them together. After that, another MLP followed by a softmax function computed the probabilities based on the representations,

$$\boldsymbol{p}_{pol} = \mathrm{softmax}(\mathrm{MLP}((\boldsymbol{H^b}^T\boldsymbol{p_c^b}; \boldsymbol{H^b}^T\boldsymbol{p_e^b}); \theta_{pol})), \tag{4}$$

where $\boldsymbol{p}_{pol} = [\boldsymbol{p}_{pol+}, \boldsymbol{p}_{pol-}]$ denotes the probability of positive and negative correlations, θ_{pol} is a learnable parameter of the MLP. In the example shown in Figure 1, $\boldsymbol{p}_{pol+}$ is supposed to be larger than $\boldsymbol{p}_{pol-}$.

Find World: Furthermore, we should return to the statement and find out the world that happened in the statement (e.g., *increasing their mass*). To achieve that, we used the same setting as the*Find Cause and Effect Property* module,

$$\boldsymbol{p}_w^s = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{H^s}; \theta_w)) \in \mathbb{R}^m, \tag{5}$$

where $\boldsymbol{p_w^s}$ is the attention vectors over the statement for the world, and θ is the learnable parameters of the MLP.

Value Prediction: Moreover, we need to decide whether the attribute value in World is an increment ($\boldsymbol{p}_{val\uparrow}$) or decrement($\boldsymbol{p}_{val\downarrow}$; e.g., *increase their mass* is an increment). To achieve that, we derive representations of the world by averaging statement representation $\boldsymbol{H^b}$ weighted by the world attention vector, $\boldsymbol{p_w^s}$. Then, another MLP was stacked with softmax to get the probabilities,

$$\boldsymbol{p}_{val} = \mathrm{softmax}(\mathrm{MLP}((\boldsymbol{H^s}^T\boldsymbol{p_w^s}); \theta_{val})), \tag{6}$$

where $\boldsymbol{p}_{val} = [\boldsymbol{p}_{val\uparrow}, \boldsymbol{p}_{val\downarrow}]$ denotes the probability that the attribute value is incremented or decremented, and θ_{val} is a learnable parameter of the MLP. In the example shown in Figure 1, $\boldsymbol{p}_{val\uparrow}$ is supposed to be larger than $\boldsymbol{p}_{val\downarrow}$.

3.2.2 Comparison

The Comparison reasoning chain contains two modules that are shared with the Prediction reasoning chain, *Find Cause and Effect*, and *Polarity Check*. However, the Comparison chain differs from the Prediction chain in two modules.

Find Worlds: In the Comparison-type question like the second example shown in Figure 1, the statement asks to compare the effects on the two entities. Thus, two worlds appeared in the statement instead of one (e.g., *large masses* and *small masses*). To achieve that, we took the same straightforward method as the *Find World* module,

$$\boldsymbol{p}_{w1}^s = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{H^s}; \theta_{w1})) \in \mathbb{R}^m, \tag{7}$$

$$\boldsymbol{p}_{w2}^s = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{H^s}; \theta_{w2})) \in \mathbb{R}^m, \tag{8}$$

where $\boldsymbol{p_{w1}^s}, \boldsymbol{p_{w2}^s}$ are the attention vectors over the statement for worlds 1 and 2, and θ's are learnable parameters of the MLP.

Worlds Comparison: This module aims to compare the worlds in terms of the cause property. For example, world 1 (*larger masses*) is more relevant to *the greater the mass* than world 2 (*smaller masses*) in Figure 1. To achieve that, we adopted a bilinear function to evaluate the relevance between each world's cause property and the cause property in background knowledge, which is further normalized into a probability with softmax,

$$\begin{aligned}\boldsymbol{p}_{com} = \mathrm{softmax}((&\boldsymbol{H^b}^T\boldsymbol{p}_c^b)^T\boldsymbol{W}_{com}(\boldsymbol{H^s}^T\boldsymbol{p}_{w1}^s),\\ &(\boldsymbol{H^b}^T\boldsymbol{p}_c^b)^T\boldsymbol{W}_{com}(\boldsymbol{H^s}^T\boldsymbol{p}_{w2}^s)),\end{aligned} \tag{9}$$

where $\boldsymbol{W_{com}} \in \mathbb{R}^{d \times d}$ is a trainable matrix, and p_{com} denotes the probability that the world is relevant to the cause property.

3.2.3 Deduction

The Deduction module aims to conduct the reasoning by considering the evidence computed from each module in the reasoning chain. In particular, we applied the decision functions as shown in Figure 2c for each reasoning type to generate the synthetic text.

We take the examples in Figure 1 as an illustration. From the *Find Cause and Effect*, and *Polarity Check* modules, we conclude a positive relationship existed between the cause *mass* and the effect *gravitational pull*, so we denote $\boldsymbol{p}_{pol}(\boldsymbol{Effect}|\boldsymbol{Cause})$ as $+$.

For Prediction, the *Value Prediction* module states that the attribute value of the world in terms of cause property is an increment. Thus, we denote $\boldsymbol{p}_{val}(\boldsymbol{W}orld|\boldsymbol{C}ause)$ as ↑. Therefore, the world must also have an increment on the effect property due to the positive relationship between the cause and effect. In this way, we generated a synthetic text by slot-filling as follows: *large mass* will cause **more** *gravitational pull*.

Similarly, for Comparison, the *Worlds Comparison* module showed that compared with world 2, world 1 was more relevant to the cause property in the background knowledge. Thus, we denote $\boldsymbol{p}_{com}(World1, World2)$ as $>$ (i.e., $\boldsymbol{p}_{com}(World1|Cause) > \boldsymbol{p}_{com}(World2|Cause)$). In this way, we rendered world 1 larger than world 2 in terms of the effect property because of the positive polarity. Then, a synthetic text was also be generated.

3.2.4 Reasoning Type Classification

As the reasoning type is not available during the inference time, we added a classifier based on the statement representations, which is defined as follows:

$$\boldsymbol{s} = \frac{1}{m}\sum_{i}^{m} \boldsymbol{H}_{\boldsymbol{i}}^{\boldsymbol{s}} \in \mathbb{R}^{d}, \tag{10}$$

$$\boldsymbol{p}_{cla} = \sigma(MLP(\boldsymbol{s}; \theta_{cla})), \tag{11}$$

where $\boldsymbol{s}$ can be viewed as an embedding of the statement, θ_{cla} is the trainable parameter in the MLP, and σ is an activation function. $\boldsymbol{p}_{cla}$ denotes the classification scores.

3.3 Answer Prediction

In the Answer Prediction component, we replaced the given background knowledge with the synthetic text from the Reasoning component. Then we combined it with the question and answer choice $A = \{a_k\}_{k=1}^{2}$ as $\langle s\rangle\ SyntheticText\ \langle/s\rangle\ \langle/s\rangle\ q; a_k\ \langle/s\rangle$. We used the final hidden vector corresponding to the first input token ($\langle s\rangle$) as the aggregate representation. We further predicted the probabilities of an answer being the answer choice a_k in the same manner as in Liu et al. (2019).

3.4 Model Training

Two models (i.e., reasoning and answer prediction) are learned in our approach.

Reasoning The loss function for reasoning is defined as

$$\ell_{Reason} = -\sum_{\boldsymbol{y}\in\boldsymbol{Y}} \alpha_{\boldsymbol{y}} \gamma_{\boldsymbol{y}} \tilde{\boldsymbol{y}}^{T} log(\boldsymbol{y}). \tag{12}$$

$\boldsymbol{Y} = \{\boldsymbol{p}_{\boldsymbol{w}}^{\boldsymbol{s}} \in \mathbb{R}^{m}, \boldsymbol{p}_{\boldsymbol{w_1}}^{\boldsymbol{s}} \in \mathbb{R}^{m}, \boldsymbol{p}_{\boldsymbol{w_2}}^{\boldsymbol{s}} \in \mathbb{R}^{m}, \boldsymbol{p}_{\boldsymbol{e}}^{\boldsymbol{b}} \in \mathbb{R}^{n}, \boldsymbol{p}_{\boldsymbol{c}}^{\boldsymbol{b}} \in \mathbb{R}^{n}, \boldsymbol{p}_{val} \in \mathbb{R}^{2}, \boldsymbol{p}_{com} \in \mathbb{R}^{2}, \boldsymbol{p}_{cla} \in \mathbb{R}^{2}\}$ are predictions of different modules, $\tilde{\boldsymbol{y}}^{T} \in \{0,1\}^{m}$ or $\tilde{\boldsymbol{y}}^{T} \in \{0,1\}^{n}$ or $\tilde{\boldsymbol{y}}^{T} \in \{0,1\}^{2}$ are corresponding gold labels, $\gamma_{\boldsymbol{y}} \in \{0,1\}^{2}$ denotes whether the loss function should use the label (as some annotations would be missed due to different reasoning type), and $\alpha_{\boldsymbol{y}}$ is the weight for each module y.

Answer Prediction: We took the standard binary cross entropy loss ℓ_{AP} as the training objective of the Answer Prediction component.

During the inference time, we first determined the reasoning type based on the classification score obtained in section 3.2.4. Then we followed the Deduction module to synthesize corresponding text and further fed it into the trained Answer Prediction model. The answer choice with the highest probability was selected as the final answer.

4 Experiment Setup

4.1 Datasets

Datasets QuaRTz (Tafjord et al., 2019b) and QuaRel (Tafjord et al., 2019a), were used to evaluate the proposed model. Both datasets require reasoning about textual qualitative relationships and provide well-defined annotations used for symbolic reasoning. The detailed statistics are in Table 1.

Auxiliary Supervision: The loss function for the Reasoning component takes two different types of supervision, i.e., span supervision (e.g., cause, effect spans in the knowledge) and binary logits supervision (value prediction, world comparison, and reasoning type). The QuaRTz dataset provides the annotations for the property descriptions and property values, but not every instance in the dataset contains such annotations (2280 out of 2696 are annotated completely). Furthermore, some annotations are not standard and must be further processed to satisfy the training objective. Therefore, we mitigated these issues by generating an additional auxiliary from hypothesis and manual annotating. The detailed guidelines and labeled examples are in Appendix A.2. All modules in the Reasoning component are trained only on the QuaRTz (the

Statistics	Train	Dev	Test
QuaRTz	2,696	384	784
QuaRel	1,941	278	552

Table 1: Statistics of the datasets

proportion of Prediction and Comparison questions is 2296:400).

4.2 Implementation Details

We used the pre-trained language model RoBERTa-large in the experimentation[1]. In particular, we trained all modules in the Reasoning component end-to-end on one Nvidia RTX8000 48GB GPU and used two GPUs for the Answer Prediction component. We tuned the Reasoning component parameters according to the averaged performance of all modules and tuned the parameters in the Answer Prediction component based on the final accuracy. We selected F1 scores and accuracy as the evaluation metrics. Specifically, we set α_y's in equation 12 as follows: 0.1 for span-based loss, 0.2 for the *World Comparison, Value Prediction*, and *Reasoning Type Classification*. The detailed hyperparameters and search bounds are described in the Appendix A.1.

4.3 Compared Models

QUASP and QUASP+: The models proposed by Tafjord et al. (2019a) which extend type-constrained semantic parsing to address the problem of tracking different "worlds" in questions.

BERT and RoBERTa: The standard multiple-choice QA frameworks are based on a powerful pre-trained language model (Devlin et al., 2019; Liu et al., 2019). As described in section 3.3, it took the concatenation of knowledge, the question, and answer options as the input and used the first token representation for classification.

gvQPS+: Instead of introducing a semantic parser, this model applies the generate-validate framework (Mitra et al., 2019). It first generates a natural language description of the logical form and validates whether the natural language description is followed from the input text.

DeepEKR: Similar to Mitra et al. (2019), this model replaces the parser with a neural network, softening the symbolic representation (Mitra et al., 2020).

Model	QuaRTz		QuaRel	
	Dev	Test	Dev	Test
Random	50.0	50.0	50.0	50.0
QUASP	-	-	62.1	56.1
QUASP+	-	-	68.9	68.7
BERT	-	67.7	-	53.0
BERT(PFT on RACE)	-	79.8	-	79.9
gvQPS+	-	-	-	76.6
DeepEKR	-	79.8	-	81.15
RoBERTa	86.8*	85.7*	81.1(84.5*)	80.0(83.3*)
LG(DA)	-	-	84.5	84.7
LG(DA+Reg)	-	-	85.1	85.0
Ours	**89.6**	**89.9**	**89.5**	**86.8**

Table 2: Performance of different models on both datasets, where PFT stands for pre-finetune, and * means the result we implemented.

LG(DA,Reg): The RoBERTa-based model leverages logical and linguistic knowledge to augment labeled training data (DA) and then uses a consistency-based regularizer (Reg) to help the training (Asai and Hajishirzi, 2020).

5 Experimental Results

5.1 Question Answering Performance

The performance of the development and test of QuaRTz and QuaRel is shown in Table 2. Our model outperformed the existing state-of-the-art models (RoBERTa and LG with DA+Reg) by a large margin (2.8% and 4.2% absolute difference on QuaRTz, 4.4%, and 1.8% on QuaRel). These results illustrate that compared to the semantic parser-based model and the one-step "black box" model, our approach that imitates human cognitive behaviors is more capable of conducting qualitative reasoning and generalization.

Furthermore, compared with the semantic parser-based methods (i.e., the first group in the table), almost all language model-based methods achieved superior performance, demonstrating the power of the pre-trained language model. However, these models are mostly data-driven. The BERT model without pre-finetuning on RACE (Lai et al., 2017) was unable to converge on QuaRel. Furthermore, the data augmentation technique resulted in a 4.7% improvement in the LG(DA) model, proving the one-step "black box" model has high demand for training resources. Nonetheless, our approach, without any additional training data, still achieves

---

[1] `https://github.com/huggingface/transformers`

Module		**F1**	**Fuzzy F1**	**Accuracy**
Find Cause and Effect	$\boldsymbol{p}_c^b$	72.6	82.3	-
	$\boldsymbol{p}_e^b$	67.0	78.4	-
Polarity Check	$\boldsymbol{p}_{pol}$	-	-	88.8
Find World	$\boldsymbol{p}_w^s$	76.4	82.5	-
Value Prediction	$\boldsymbol{p}_{val}$	-	-	91.5
Find Worlds	$\boldsymbol{p}_{w1}^s$	77.3	83.4	-
	$\boldsymbol{p}_{w2}^s$	74.9	80.6	-
World Comparison	$\boldsymbol{p}_{com}$	-	-	84.6
Reasoning Type Classification	$\boldsymbol{p}_{cla}$	-	-	88.0

Table 3: Performance in the Reasoning component.

better performance, demonstrating its effectiveness and rationality.

5.2 Reasoning Component Performance

In the Reasoning component, we designed multiple neural network modules for two reasoning chains. Each module was designed to accomplish its subtask and provide an intermediate prediction that explains the reasoning process in a human cognitive manner. To evaluate the effectiveness of each module, we converted the attention vectors over the context into a text span and the probability scores into a predicted class. The conversion details are in the Appendix A.3. Table 3 shows the performance results for each module.

Find Cause and Effect, *Find World*, and *Find Worlds*, three modules that should detect concerned text spans from the context, all achieved adequate F1 scores. There are two reasons why our model did not score very well on a span extraction task like extractive machine reading comprehension (e.g., SQuAD[2]). First, the text span in our task is not as accurately defined as the answer in MRC. Generally, the length of the text span is long, and the boundary is fuzzy. Second, our approach computed the attention score for each token in the context and leveraged it softly, and thus less sensitive to boundary detection. However, when we used fuzzy F1 scores as the evaluation metrics (introduced in (Ren et al., 2020), which were marked as 1 as long as the original F1 was not equal to 0), the scores for all modules increased by a large margin, proving the reasoning ability.

Polarity Check, *Value Prediction*, and *World Comparison*, three modules that require classification capability based on the given span prediction of the upstream modules, achieved high accuracy (88.8%, 91.5%, and 84.6%), indicating the rationality of our end-to-end sequential model design. Additionally, We could argue that determining whether an attribute is incremental or decremental is a relatively simple task for language models.

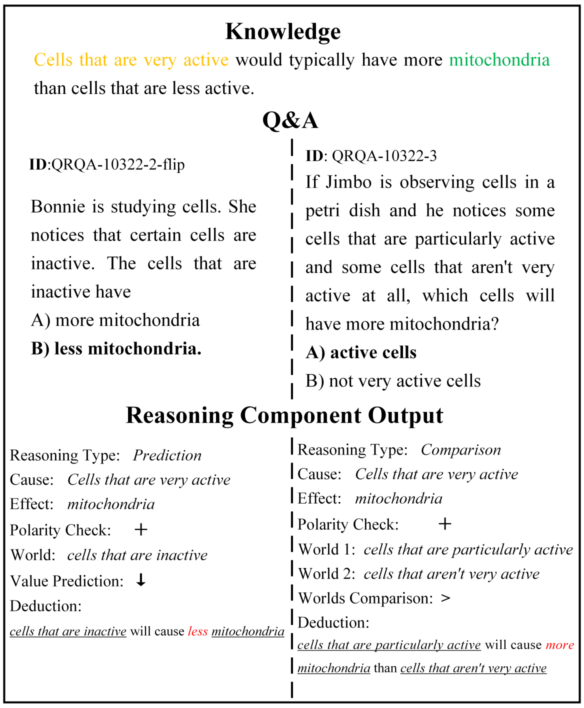

Knowledge

Cells that are very active would typically have more mitochondria than cells that are less active.

Q&A

ID:QRQA-10322-2-flip

Bonnie is studying cells. She notices that certain cells are inactive. The cells that are inactive have

A) more mitochondria

B) less mitochondria.

ID: QRQA-10322-3

If Jimbo is observing cells in a petri dish and he notices some cells that are particularly active and some cells that aren't very active at all, which cells will have more mitochondria?

A) active cells

B) not very active cells

Reasoning Component Output

Reasoning Type: *Prediction*
Cause: *Cells that are very active*
Effect: *mitochondria*
Polarity Check: +
World: *cells that are inactive*
Value Prediction: ↓
Deduction:
cells that are inactive will cause *less* *mitochondria*

Reasoning Type: *Comparison*
Cause: *Cells that are very active*
Effect: *mitochondria*
Polarity Check: +
World 1: *cells that are particularly active*
World 2: *cells that aren't very active*
Worlds Comparison: >
Deduction:
cells that are particularly active will cause *more* *mitochondria* than *cells that aren't very active*

Figure 3: An example from QuaRTz with visualized intermediate outputs provided by our model.

The *Reasoning Type Classification* module obtained a good-enough performance (90.1%), the recall value for Prediction type was 88.0%, and the recall value for Comparison type was 76.7%. The result is acceptable as some questions could be solved simultaneously in a Prediction and a Comparison manner. For example, the Comparison question in Figure 1 can also be considered a Prediction if we care only about the world with high priority (i.e., *two planets with very large mass*). This phenomenon somehow increased the fault tolerance and improved the generality of our method.

Case study: Our approach provides intermediate outputs that sufficiently explain the reasoning process, which can lead to interpretability. Figure 3 presents examples of the transparent reasoning process run by our model. More examples can be found in the Appendix A.5. The knowledge states the relationship between *active cells* and *mitochondria*, while two questions describe different scenarios to test the application of this knowledge.

As shown in Figure 3, our model outputs several intermediate results through the Reasoning component. First, it identifies the reasoning chain type for

---

[2] `https://rajpurkar.github.io/SQuAD-explorer/`

each question; the left one is prediction, while the right one requires comparison. Then, each module in the corresponding reasoning chain outputs its prediction. For example, the *Find Cause and Effect* module captured the cause property and the effect property in the knowledge. Then the *Polarity Check* module correctly figured out the positive relationship between them. The examples indicate that our approach not only conducts final answer prediction but also explains the machine understanding and reasoning process.

5.3 Human Evaluation

In the Answer Prediction component, we fed the generated synthetic text instead of given knowledge into the model. To further measure the effect of synthetic text on answer prediction, we introduced the human evaluation widely used in NLG field. We assembled ten well-educated volunteers and gave each person 30 randomly sampled questions from QuaRTz, half of which provided knowledge, half of which provided our synthetic text. Then we asked the participants to rate the question on a 5-point scale for each following metric: *fluency* (does it read coherently), *informativeness* (how much information is contained), *explicitness* (does it describe the relation clearly), and *relevance* (is it relevant to the question). We also measured the time consumption for problem-solving by giving knowledge or synthetic text. Table 4 shows the results.

The given knowledge had higher scores for *fluency* and *informativeness* because the knowledge sentences were manually extracted from a large corpus and may contain multiple relations among multiple properties, while the synthetic text was generated by slot-filling and described only the relation mentioned in the question. Furthermore, the synthetic text achieved superior scores for *explicitness* and *relevance*, which indicates our Reasoning component is successful in understanding and reasoning between the question and knowledge and expressed the explicit relationship in a way close to the description in the question scenario. Additionally, this is shown in that the participants took less time to answer questions with synthesized text.

5.4 Error Analysis

We randomly sampled 60 examples for error analysis and more than half were caused by wrong predictions in the Reasoning component (48 out of 60). The other errors were in two main categories.

	Given Knowledge	**Synthetic Text**
Fluency	**4.3**	3.9
Informativeness	**4.5**	3.3
Explicitness	3.4	**4.4**
Relevance	3.8	**4.6**
Time Consumption(s)	31.1	**24.3**

Table 4: Human evaluation results.

Incomplete Knowledge: These errors occurred when the question mentioned only a world without any attribute value description; e.g., the question aimed to predict the gravitational pull for the Sun, and knowledge told that a larger mass causes a larger gravitational pull. To answer this question, we need to know that the sun has a large mass.

Incomplete Synthetic Text: Two reasons cause such errors. One is that the boundaries are not well-defined when attention vectors are turned into text, resulting in a lack of fluency or loss of information. The other is a mismatch between the knowledge and the question; e.g., the knowledge talks about the distance, but the question describes hugging. Thus, the generated synthetic text does not provide enough information to solve the problem.

6 Conclusion

In this paper, we aimed to solve the qualitative reasoning task in an interpretable manner. Inspired by human cognition, we first summarized the questions into two categories, Prediction and Comparison. Then an end-to-end trained Reasoning component that contains two reasoning chains was designed. Both reasoning chains contained multiple neural modules that provide transparent intermediate predictions for the understanding and reasoning process. The experimental results showed the effectiveness of our approach, and the analysis of each module and case study demonstrated the superior interpretability compared with the "black-box" model. Moreover, we found that some questions could be solved by both reasoning chains, thus increasing the default tolerance and generalization capability. Furthermore, a human evaluation was conducted to validate the function of the synthetic text and provide an additional explanation for the superior performance achieved by our method. However, the error analysis showed the inadequacy under complicated scenarios. Therefore, our future work will focus on applying interpretable reasoning on complex reasoning tasks. The annotated data and models are shared publicly [3].

Acknowledgement

We acknowledge this work is supported by the Joint Funds of the National Natural Science Foundation of China (Grant No. U19B2020).

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Kenneth D Forbus. 1984. Qualitative process theory. *Artificial intelligence*, 24(1-3):85–168.

Nitish Gupta and Mike Lewis. 2018. Neural compositional denotational semantics for question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2152–2161, Brussels, Belgium. Association for Computational Linguistics.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2019. Neural module networks for reasoning over text. *arXiv preprint arXiv:1912.04971*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813.

Yichen Jiang and Mohit Bansal. 2019. Self-assembling modular networks for interpretable multi-hop reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4474–4484, Hong Kong, China. Association for Computational Linguistics.

Yichen Jiang, Nitish Joshi, Yen-Chun Chen, and Mohit Bansal. 2019. Explore, propose, and assemble: An interpretable model for multi-hop reading comprehension. *arXiv preprint arXiv:1906.05210*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62, Hong Kong, China. Association for Computational Linguistics.

Jiangming Liu and Matt Gardner. 2020. Multi-step inference for reasoning over paragraphs. *arXiv preprint arXiv:2004.02995*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

---

[3] `https://github.com/Borororo/Interpretable_QR`

A. Mitra, Chitta Baral, Aurgho Bhattacharjee, and Ishan Shrivastava. 2019. A generate-validate approach to answering questions about qualitative relationships. *ArXiv*, abs/1908.03645.

Arindam Mitra, Sanjay Narayana, and Chitta Baral. 2020. Deeply embedded knowledge representation & reasoning for natural language question answering: A practitioner's perspective. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 102–111, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Mucheng Ren, Xiubo Geng, Tao Qin, Heyan Huang, and Daxin Jiang. 2020. Towards interpretable reasoning over paragraph effects in situation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6745–6758, Online. Association for Computational Linguistics.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019a. Quarel: A dataset and models for answering questions about qualitative relationships. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7063–7071.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019b. QuaRTz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946, Hong Kong, China. Association for Computational Linguistics.

Daniel S Weld and Johan De Kleer. 2013. *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

A Appendices

A.1 Parameters List

Table 5 and Table 6 show the parameter details used in our settings.

Reasoning	Search Space(Bounds)	Best Assignment
Max Seq. Length	*choice*[256,384,512]	512
Learning Rate	*uniform-float*[5e-6,3e-5]	1e-5
Batch Size per GPU	*choice*[4,8,16]	8
Gradient Accumulation	*choice*[1,2]	2
No. of Epoch	*uniform-integer*[1,10]	8
No. of Search trials	20	20
Optimizer	Adam	Adam
Fixed Length for B,S	[200,200]	[200,200]
No. of GPU(RTX8000)	1	1
Average runtime (mins)	25	25

Table 5: Detailed parameters used in the Reasoning component.

Answer Prediction	Search Space(Bounds)	Best Assignment
Max Seq. Length	*choice*[256,384,512]	256
Learning Rate	*uniform-float*[1e-6,3e-5]	1e-5
Batch Size per GPU	*uniform-integer*[4,8,16]	16
Gradient Accumulation	*choice*[1,2]	2
No. of Epoch	*uniform-integer*[5,15]	10
No. of Search trials	30	3 0
learning rate optimizer	Adam	Adam
No. of GPU(RTX8000)	2	2
Average runtime (mins)	40	40

Table 6: Detailed parameters used in the Answer Prediction component.

A.2 Auxiliary Supervision Construction

Figure 4 shows one labelled example. The official dataset already provides detailed annotations for both knowledge and question, where some annotations could be directly introduced in our training objective. For the example, "para_anno" tells the cause and effect properties described in the knowledge, and the polarity among them would be marked as positive if the "effect_dir_sign" is the same as the "casue_dir_sign", otherwise will be marked as negative.

However, some annotations are still missing, e.g., reasoning type. We introduce two methods to get additional labels. On the one hand, We made some hypotheses. For example, if "question_anno" contains "more_effect_xxx" and "less_effect_xxx" simultaneously, we would consider this question as a comparison type. On the other, we annotate the samples manually. For the questions without qualified annotation, we label them by hand. Particularly, we find the text spans in the knowledge and question, then transform the annotations to the machine-readble form automatically.

A.3 Conversion Instruction

For *Find Cause and Effect*, *Find World*, and *Find Worlds* modules, we should convert the attention

Knowledge

An object with greater mass or greater velocity has more kinetic energy.

Q&A

Milo threw both a basketball and a baseball through the air. If the basketball has more mass then the baseball, which ball has more kinetic energy?

A) Basketball B) Baseball

Official Annotation

para_anno:
"effect_dir_sign": "MORE",
"cause_dir_sign": "MORE",
"effect_prop": "kinetic energy",
"cause_prop": "mass",
"cause_dir_str": "greater",
"effect_dir_str": "more".
question_anno:
"more_effect_dir": "more",
"more_cause_prop": "mass",
"more_cause_dir": "more",
"less_effect_prop": "kinetic energy",
"less_effect_dir": "less",
"more_effect_prop": "kinetic energy".

Auxiliary Annotation

"type": "Comparison",
"b_cause": "mass",
"b_cause_ind": [23, 27],
"b_effect": "kinetic energy",
"b_effect_ind": [57, 71],
"b_relation": Positive,
"s_world_1": "basketball",
"s_world_2": "baseball",
"s_world_1_ind": [68, 92],
"s_world_2_ind": [98, 110],
"s_comparison": World 1

Figure 4: An example with auxiliary supervision.

vector output into a text span.

First, the token with the highest probability is selected. After that, we check left and right from the selected word position and extend it if the probability of each neighbor is larger than the threshold. The threshold value is determined based on F1 scores.

For the rest modules, we select the predicted class with the highest probability.

A.4 Ablation Study on sub-tasks

We decompose the complex qualitative reasoning task into several simple sub-tasks according to logic and then train several sub-tasks together. Naturally, as the number of tasks increases, errors accumulate, but our method ultimately performs better. To further examine the rationality of our work, we conduct an ablation study on partial modules.

Modules	**Cause**		**Effect**		**Polarity**
	F1	Fuzzy F1	F1	Fuzzy F1	Accuracy
Find Cause and Effect + Polarity Check	**72.6**	**82.3**	**67.0**	**78.4**	**88.8**
only Find Cause and Effect	71.7	81.8	66.8	75.5	58.6
only Polarity Check	8.3	14.3	9.6	16.7	83.3

Table 7: Ablation study on Find Cause and Effect and Polarity Check modules.

Table 7 shows the ablation study result on *Find Cause and Effect* and *Polarity Check* modules. The model achieves the best performance when we train two modules together. However, when we remove any module, i.e., focus on a single sub-task, the corresponding performance deteriorates. This may due to our approach, which mimics human cognitive design, enables the model to make better use of information from upstream modules to facilitate training. Therefore, our model could finally achieve superior performance.

A.5 More Examples

We present more example with intermediate outputs that correctly answered by our model in Table 8.

ID:QRQA-10004-2 **Knowledge**: The gravitational force increases with mass and decreases with the distance between the bodies. **Q&A**: John was watching the physics calculator and noted a profound finding. As the mass increases, the pull of the gravitational force A) Decreases **B) Increases**.	Type: *Prediction* Cause: *mass* Effect: *gravitational force* Polarity: **+** World: *mass increases* Value: $\uparrow$ Deduction: *mass increases* will cause **more** *gravitational force*.
ID:QRQA-10004-2-flip **Knowledge**: The gravitational force increases with mass and decreases with the distance between the bodies. **Q&A**: John was watching the physics calculator and noted a profound finding. As the mass decreases, the pull of the gravitational force A) Decreases **B) Increases**.	Type: *Prediction* Cause: *mass* Effect: *gravitational force* Polarity: **+** World: *mass decreases* Value: $\downarrow$ Deduction: *mass decreases* will cause **less** *gravitational force*.
ID:QRQA-10228-1 **Knowledge**: The larger the light collecting area, the more light a telescope gathers and the higher resolution (ability to see fine detail) it has. **Q&A**: Compared to a 1 inch wide telescope, would a 100 meter telescope collect **A) more light** B) less light?	Type: *Comparison* Cause: *collecting area* Effect: *light* Polarity: **+** World 1: *1 inch wide telescope* World 2: *100 meter telescope* Comparison: $<$ Deduction: *1 inch wide telescope* will cause **less** *light* than *100 meter telescope*.
ID:QRQA-10228-1-flip **Knowledge**: The larger the light collecting area, the more light a telescope gathers and the higher resolution (ability to see fine detail) it has. **Q&A**: Compared to a 100 meter wide telescope, would a 1 inch telescope collect A) more light **B) less light**?	Type: *Comparison* Cause: *collecting area* Effect: *light* Polarity: **+** World 1: *100 meter wide telescope* World 2: *1 inch telescope* Comparison: $>$ Deduction: *100 meter wide telescope* will cause **more** *light* than *1 inch telescope*.

Table 8

Boundary Detection with BERT for Span-level Emotion Cause Analysis

Xiangju Li[1], **Wei Gao**[2], **Shi Feng**[1], **Yifei Zhang**[1], **Daling Wang**[1]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]School of Computing and Information Systems, Singapore Management University, Singapore
`lixiangju100@163.com; weigao@smu.edu.sg; fengshi@cse.neu.edu.cn;`
`zhangyifei@cse.neu.edu.cn; wangdaling@cse.neu.edu.cn`

Abstract

Emotion cause analysis (ECA) has been an emerging topic in natural language processing, which aims to identify the reasons behind a certain emotion expressed in the text. Most ECA methods intend to identify the clause which contains the cause of a given emotion, but such clause-level ECA (CECA) can be ambiguous and imprecise. In this paper, we aim at span-level ECA (SECA) by detecting the precise boundaries of text spans conveying accurate emotion causes from the given context. We formulate this task as sequence labeling and position identification problems and design two neural methods to solve them. Experiments on two benchmark ECA datasets show that the proposed methods substantially outperform the existing ECA models [1].

1 Introduction

The task of emotion cause analysis (ECA) (Lee et al., 2010a), which is to extract the causes of an emotion expression from a given context, has gained increasing attention recently.

Most existing studies (Gui et al., 2016, 2017; Li et al., 2018, 2019; Xia et al., 2019) formulate ECA as a clause-level task (dubbed as CECA), which tries to extract the clauses that contain the emotion cause content. In CECA, a clause is typically a text segment separated by punctuation marks (e.g., ‘,’, ‘.’, ‘?’, ‘!’, etc.) in the given context. In the following example, the clause $[x_3]$ will be extracted as it contains the reason “the risk of infringement” that stimulates the emotion *afraid*.

Example 1. [2] $[x_1]$ *The claim for damages caused by infringement has legal basis,* $[x_2]$ *and it is logical and reasonable.* $[x_3]$ *We can’t give up the good character of friendship and mutual assistance just because we are <u>afraid</u> of* ***the risk of infringement****.* $[x_4]$ *Give up food for being afraid of being choked.* $[x_5]$ *We should adhere to the traffic rules.*

However, determining the clause containing the stimulus is sub-optimal and inaccurate for ECA. In Example 1, while among the 5 clauses $[x_3]$ is the best, its main content “We can’t give up . . . assistance” is not the cause of *afraid*. Such gap motivates a strong need of pinpointing more precise or finer-grained cause expressions which can convey the specific reasons of an emotion.

Some studies (Gao et al., 2015a,b; Neviarouskaya and Aono, 2013) try to extract emotion cause triples (a triple like (noun, verb, noun)) or emotion cause phrases. The extracted triples or phrases are usually not a complete emotion cause expression since the words in these triples or phrases are typically not continuous. Bi et al. (2020) proposed a new task for emotion and emotion cause span pair and classification task. Lee et al. (2010a) summarized seven groups of linguistic cues which could serve as indicators of cause events. Ghazi et al.. (2015) built a Conditional Random Fields (CRF) learner to extract emotion cause spans with a set of manually engineered features. These two approaches are labor-intensive and prone to the sub-optimal design of features.

We study the Span-level ECA (SECA) based on state-of-the-art neural approach by focusing on detecting the boundaries of cause spans. We approach to the boundary detection in two different ways via sequence tagging and start/end position identification both in an end-to-end fashion. The emotion cause span is usually a sequence of consecutive tokens which convey exact reasons of the emotion and need to be inferred from the whole context. More specifically, we obtain the emotion

---

[1] The code and datasets can be found at https://github.com/xxxyyy2020/boundary-master

[2] **Bold**: Gold emotion cause span; <u>Underline</u>: emotion expression. $[x_i]$ represents the *i*-th clause. The original report is in Chinese and translated into English.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 676–682
August 1–6, 2021.

cause span by 1) mapping each token in the input context to a label indicating the ranges of spans; 2) pointing to the start and end positions of the span directly in the given context. The main contributions of our work are three-fold:

- We formulate SECA task as sequence tagging and position identification problems for boundary detection of emotion cause spans, and propose neural models to tackle this task.
- Different from previous ECA approaches, a pointer network is introduced to identify the start and end positions of emotion cause in our work. This is the first time to introduce the pointer network to ECA task.
- Experiments on two benchmark datasets show that our models substantially outperform the state-of-the-art ECA approaches at both span and clause levels.

2 Related Work

Lee et al. (2010b) first proposed a task on ECA and constructed a small-scale Chinese emotion cause corpus. Based on this corpus, Chen et al. (2010) proposed a rule based method for this task. Gui et al. (Gui et al., 2014) extended the rule-based method to Chinese microblog text. Gao et al. (Gao et al., 2015b,a) treated the emotion cause as a list of triples (a triple like (noun, verb, noun)). They structured a microblog post as a set of triples to determine which triple is the emotion cause. Shuntaro et al.(Yada et al., 2017) defined emotion cause as the nearest clause describing events of the cause of the emotion. Gui et al. (2016) constructed a public ECA dataset based on news corpus and proposed a multi-kernel-based method for CECA. All these methods rely on either manual rules or feature engineering.

Gui et al. (2017) first proposed a deep neural model for CECA. Following that, many CECA methods were developed with deep learning (Gui et al., 2017; Li et al., 2018; Yu et al., 2019; Ding et al., 2019; Li et al., 2019; Xia et al., 2019; Fan et al., 2019). Li et al. (2019; 2018) improved the performance of CECA by considering the contextual information of the emotion. Xia et al. (2019) proposed a hierarchical network for CECA based on Transformer encoder to capture different types of features. Fan et al. (2019) utilized the discourse information and clause position for the task. Xia et al. (2019) and Chen et al. (2020a) proposed two variant tasks of ECA for extracting emotion and cause clause pairs. And then, many deep learning models (Wu et al., 2020; Chen et al., 2020b,c; Wei et al., 2020; Singh et al., 2021) were designed to tackle emotion cause pair extraction task.

Little work has been done for SECA. Lee et al. (Lee et al., 2010a) summarized the generalized rules for detecting the emotion causes of the emotion in Chinese. Based on manually crafted linguistic cues, Ghazi et al. (2015) built a CRF learner to identify emotion cause spans. However, these models are oversimplified and reliant on the ad-hoc features designed, thus are not generalized well.

3 Methodology

In this section, we first introduce two types definition of SECA. Then we describe our designed models.

3.1 Problem Formulation

Given a context S, which includes a sequence of n tokens $S = [w_1, w_2, \ldots, w_n]$, the emotion expression $E = [e_1, e_2, \ldots, e_m]$ and at least one emotion cause span, SECA task aims to detect the boundaries of emotion cause spans from S which stimulate the emotion expression.

Firstly, this task can be formulated as sequence labeling problem, of which the goal is to obtain the correct sequence of labels:

$$L = [l_1, l_2, \ldots, l_n] \tag{1}$$

where $l_i \in \{B, I, O\}$, and B, I and O denote the beginning, inside of and outside of cause span, respectively, indicating the ranges of spans.

Secondly, this task also can be formulated as a position identification problem, where the goal is to get a list of correct start-end positions of emotion cause spans:

$$\mathcal{B} = [(s_1, t_1), (s_2, t_2), \ldots] \tag{2}$$

where s_i and t_i are the start and end token indices of the i-th cause span in the context, respectively.

3.2 Model Description

We use BERT (Devlin et al., 2019) as the backbone of our models due to it has strong contextualized representation ability. The context S and the emotion E are concatenated by forming a combined sequence as the input fed into BERT:

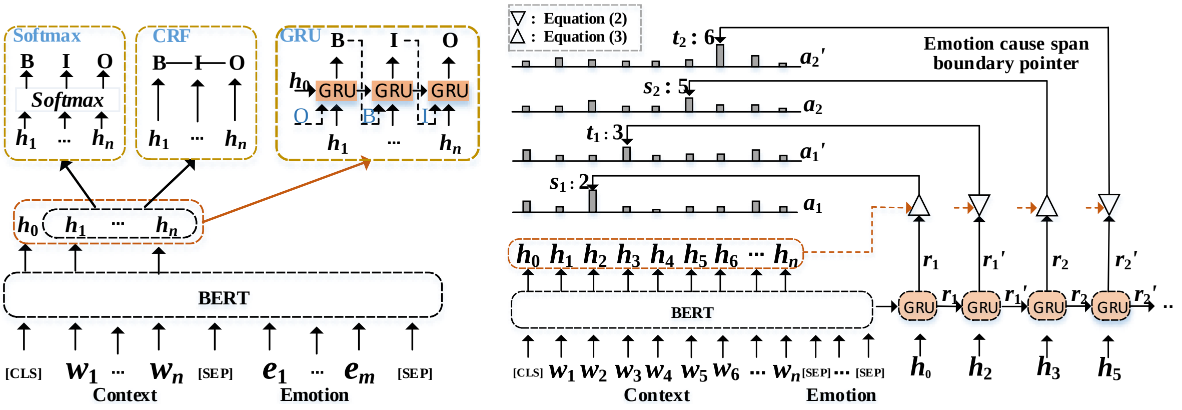

(a) Sequence labeling models.

(b) Start/End position identification model.

Figure 1: The architectures of the proposed boundary detection models for SECA.

$[\text{CLS}], w_1, \ldots, w_n, [\text{SEP}], e_1, \ldots e_m, [\text{SEP}]$, where [CLS] and [SEP] are special tokens. Then, BERT receives the sequence and outputs the representation of each token in the combined sequence. The contextualized representations of each token $w_i \in S$ and the token [CLS] can be given as:

$$h_i = \text{BERT}(w_i), \quad h_0 = \text{BERT}([\text{CLS}]) \qquad (3)$$

where h_0 and $h_i \in \mathbb{R}^{d_b}$ and d_b is the vector dimension of the last layer of BERT. Based on the BERT encoding, we introduce the two types of span boundary detection models for SECA. An overview of our models are shown in Figure 1.

3.2.1 Sequence Labeling Models

We use three common ways to predict the labels of an input sequence. The first is the softmax function direct tag prediction which is straightforward to compute. The second is CRF, a well-known statistical graphical model which has demonstrated state-of-the-art accuracy on many sequence labeling tasks (Liu et al., 2014; Jin and Yu, 2021). The third is a variant of RNN (Goller and Kuchler, 1996) (e.g. GRU, LSTM) which generates tags sequentially by predicting the next label considering its previously predicted labels.

Softmax. Based on the token representation obtained in Equation (3), the probability distribution over the label set can be computed as:

$$p(l_i) = \text{softmax}(W_1 h_i + b_1) \qquad (4)$$

where $W_1 \in \mathbb{R}^{d_b \times d_l}$ and $b_1 \in \mathbb{R}^{d_l}$ are learnable parameters, $p(l_i) \in \mathbb{R}^{d_l}$ is the label probability distribution of the i-th word and d_l is the size of label set.

CRF. CRF obtains the probability of a whole sequence:

$$p(L|S) = \frac{\exp(\text{score}(S, L))}{\sum_{L' \in \mathcal{L}} \exp(\text{score}(S, L'))} \qquad (5)$$

where $\mathcal{L}$ is the set of all candidate label sequences of the context. Here, we omit the detail of this classical prediction model (see Lafferty et al. (2001)).

GRU. The GRU version of RNN is effective and easy to train (Ma et al., 2019). At each predicting step, we update the hidden state h_i of token w_i using:

$$\tilde{h}_i = \text{GRU}(\tilde{h}_{i-1}, \ell(l_{i-1}) \oplus h_i) \qquad (6)$$

where $\tilde{h}_i \in \mathbb{R}^{d_g}$ and d_g is the number of the GRU units, $\ell(l_{i-1}) \in \mathbb{R}^{d_\ell}$ is the embedding vector of previous label l_{i-1} and d_ℓ is the size of label embedding. Then, the probability distribution of l_i can be obtained by:

$$p(l_i) = \text{softmax}(W_2 \tilde{h}_i + b_2) \qquad (7)$$

where $W_2 \in \mathbb{R}^{d_g \times d_l}$, $b_2 \in \mathbb{R}^{d_l}$ are training parameters.

3.2.2 Position Identification Model

Point network was first proposed by Vinyals et al. (2015), which is suitable for our positions identification task thanks to the fact that it is able to select positions from the input. Here we adopt it to generate the start and end positions of spans in turn. Because the number of cause spans is not fixed, we set a parameter $\mathcal{C}$ to control how many spans to generate, and allow the two end points in (s_i, t_i) to take an integer value in $[0, n]$. For the k-th span (s_k, t_k), we first get its start index s_k. An attention mechanism is used to obtain the attention weight α_{kj} of the j-th token (for all $j \in [0, n]$) as follows:

$$\begin{aligned} a_{kj} &= \frac{\exp(q_{kj})}{\sum_{j'=0}^{n} \exp(q_{kj'})} \\ q_{kj} &= v^{\top} \tanh(h_j W_3 + r_k W_4) \end{aligned} \tag{8}$$

where $W_3 \in \mathbb{R}^{d_b \times d_v}$, $W_4 \in \mathbb{R}^{d_r \times d_v}$ and $v \in \mathbb{R}^{d_v}$ are learnable parameters (d_v is the size of v), r_k is the hidden vector obtained by GRU:

$$r_k = \mathrm{GRU}(h_{t_{k-1}}, r'_{k-1}) \tag{9}$$

where t_{k-1} is the index of end position of the (k-1)-th span and $r'_{k-1} \in \mathbb{R}^{d_r}$ (d^r is the number of GRU units) is hidden vector obtained by GRU for predicting t_{k-1} (see below). Then, we set s_k as the highest attention weight from attention vector $a_k = [a_{k0}, \ldots, a_{kn}]$.

To predict the end index t_k of the k-th span, the same attention mechanism is used to get the attention weight a'_{kj}:

$$\begin{aligned} a'_{kj} &= \frac{\exp(\hat{q}_{kj})}{\sum_{j'=0}^{n} \exp(\hat{q}_{kj'})} \\ \hat{q}_{kj} &= v^{\top} \tanh(h_j W_3 + r'_k W_4) \\ r'_k &= \mathrm{GRU}(h_{s_k}, r_k) \end{aligned} \tag{10}$$

where W_3, W_4 and v are learnable parameters. Similarly, the end index t_k can be identified by taking the maximum value in a'_k.

4 Experiments and Results

We evaluate the proposed methods on the Chinese ECA (**CHI**) dataset (Gui et al., 2016) and English ECA (**ENG**) dataset (Ghazi et al., 2015). On both datasets, cause spans are manually annotated and each context contains an emotion expression (or emotion category) and at least one cause span. The CHI dataset contains 2,105 instances and 2,147 cause spans. The ENG dataset contains 820 instances, and each contains one span only.

We use pre-trained *BERT-Base Chinese* and *BERT-Base Uncased* to encode the CHI and ENG datasets, respectively. We also choose the pre-trained span-level *SpanBERT-Base-cased* (n/a for Chinese) as the encoder on ENG dataset. We follow the default settings of BERT (Devlin et al., 2019) for fine-tuning. Adam (Kingma and Ba, 2015) optimizer is used with learning rate 1e-5. The epoch is set to 10. The size of label embedding is 50. $\mathcal{C}$[3] is set to 3 on CHI and 1 on ENG which are the maximum number of cause spans in the respective *training* set. We follow the settings of previous works to split the datasets for train/dev/test (Ghazi et al., 2015; Gui et al., 2017). Hyper-parameters are tuned on the dev set.

Model	P^s	R^s	F_1^s
	CHI		
Lee et al. (2010a)	0.197	0.161	0.177
Gui et al. (2017)	0.187	0.181	0.184
BERT$_{base}$+Softmax	0.483	**0.574**	0.525
BERT$_{base}$+GRU	0.481	0.567	0.520
BERT$_{base}$+CRF	0.564	0.570	**0.566**
BERT$_{base}$+Pointer	**0.570**	0.526	0.547
	ENG		
Ghazi et al. (2015)	0.666	0.593	0.628
BERT$_{base}$+Softmax	0.838	0.876	0.856
BERT$_{base}$+GRU	0.883	0.868	0.875
BERT$_{base}$+CRF	0.866	0.890	0.878
BERT$_{base}$+Pointer	**0.910**	**0.891**	**0.901**
BERT$_{span}$+Softmax	0.830	**0.891**	0.859
BERT$_{span}$+GRU	0.893	0.879	0.886
BERT$_{span}$+CRF	0.858	0.883	0.870
BERT$_{span}$+Pointer	**0.904**	0.890	**0.897**

Table 1: SECA results based on span-level metrics.

We have BERT (**BERT**$_{base}$) and SpanBERT (**BERT**$_{span}$) as pre-trained models, and **Softmax**, **CRF**, **GRU** and pointer network (**Pointer**) as prediction models.

4.1 SECA Result

On CHI dataset, we compare our model with the rule based model (Lee et al., 2010a) and the word ECA model that outputs cause words (Gui et al., 2017), which is somewhat similar to our work. On ENG dataset, the CRF-based model in (Ghazi et al., 2015) is the only span-level baseline. For evaluation, we use span precision (P^s), recall (R^s) and F1 score (F_1^s) which are defined as

$$\begin{aligned} P^s &= \frac{\text{\# of correct cause spans}}{\text{\# of predicted cause spans}} \\ R^s &= \frac{\text{\# of correct cause spans}}{\text{\# of gold cause spans}} \\ F_1^s &= \frac{2 \times P^s \times R^s}{P^s + R^s} \end{aligned} \tag{11}$$

---

[3] If the number of emotion cause span(s) is less than $\mathcal{C}$, $(0, 0)$ is used for padding the correct list $\mathcal{B}$.

Table 1 shows that all our models strongly outperform the baselines, indicating our method is effective. More specifically, on CHI dataset our models BERT$_{base}$+Softmax, BERT$_{base}$+GRU, BERT$_{base}$+CRF and BERT$_{base}$+Pointer outperform the strong baseline model in Gui et al. (2017) by relative F_1^s improvements of 34.1%, 33.6%, 38.2% and 36.3%, respectively. The improvement is very significant with p-value less than 0.001 in t-test. With BERT encoder, Pointer performs the best on ENG and the second best on CHI, suggesting that pointer network is basically effective. Pointer is relatively worse than CRF on CHI because the number of spans it outputs is fixed as three while an instance on CHI may have less than three spans, rendering a small disadvantage. But generally, using pointer network for span boundary detection provides a strong alternative to classic CRF. It is a bit surprising that BERT$_{span}$ does not show advantage over BERT$_{base}$ here. We conjecture that given the small ENG dataset we cannot perform strong fine-tuning to make a big difference. In addition, the performance on ENG is much higher than on CHI because the context is generally much shorter making the task easier.

4.2 CECA Result

We can directly output clauses containing the predicted cause spans to compare with the rule based SECA model (2010a) and some strong CECA models (Gui et al., 2017; Li et al., 2018; Ding et al., 2019; Li et al., 2019; Xia et al., 2019; Fan et al., 2019; Hu et al., 2021). We only use **CHI** here since no clause labels are available and no previous work for CECA was done on English data for us to compare with. Following previous works (Gui et al., 2017; Xia et al., 2019), we use clause precision (P^c), recall (R^c), and F1 score (F_1^c) as evaluation metrics:

$$\begin{aligned} P^c &= \frac{\text{\# of correct cause clauses}}{\text{\# of predicted cause clauses}} \\ R^c &= \frac{\text{\# of correct cause clauses}}{\text{\# of gold cause clauses}} \\ F_1^c &= \frac{2 \times P^c \times R^c}{P^c + R^c} \end{aligned} \quad (12)$$

Model	P^c	R^c	F_1^c
Lee et al. (2010a)	0.675	0.429	0.524
Gui et al. (2017)	0.708	0.684	0.700
Li et al. (2018)	0.772	0.689	0.727
Ding et al. (2019)	0.762	0.691	0.742
Li et al. (2019)	0.784	0.759	0.771
Xia et al. (2019)	0.770	0.766	0.768
Fan et al. (2019)	0.811	0.773	0.791
Hu et al. (2021)	0.786	0.757	0.771
BERT$_{base}$+Softmax	0.828	**0.908**	0.866
BERT$_{base}$+GRU	0.825	0.903	0.862
BERT$_{base}$+CRF	**0.864**	0.873	**0.867**
BERT$_{base}$+Pointer	0.834	0.844	0.838

Table 2: CECA results on **CHI** benchmark dataset.

Table 2 shows that our proposed models outperform all the state-of-the-art CECA baselines. We attribute this to the fact that the BERT's contextualized representation capacity and our SECA models are supervised by the finer-grained span-level annotations directly, which can effectively guide model to learn more precise cause-related information. This advantage cannot be taken easily by the baseline approaches due to the nature of their clause-level supervision. Moreover, simply mapping the predicted spans to clauses for output makes Softmax become comparable with CRF and puts Pointer the last in terms of CECA performance. This is not surprising because an inaccurate span might still result in the right clause which just contains the predicted span. We also notice that rule based model performs worse than all the feature-learning models because that the manual rules hardly adopt to different datasets and feature-learning model can learn effective features according to the different datasets.

5 Conclusion

In this paper, we aim at span-level emotion cause analysis and propose neural sequence labeling and position identification models to detect the boundaries of emotion cause spans. The experiments conducted on two benchmark datasets of different languages demonstrate the effectiveness of the proposed approach, which achieves a new state-of-the-art performance on both span-level and clause-level ECA tasks.

Acknowledgments

The work was supported by National Key R&D Program of China (2018YFB1004700), National Natural Science Foundation of China (61772122,61872074,U1811261).

References

Hongliang Bi and Pengyuan Liu. 2020. ECSP: A new task for emotion-cause span-pair extraction and classification. *CoRR*, abs/2003.03507.

Xinhong Chen, Qing Li, and Jianping Wang. 2020a. Conditional causal relationships between emotions and causes in texts. In *In Proceedings of 2020 EMNLP*, pages 3111–3121.

Xinhong Chen, Qing Li, and Jianping Wang. 2020b. A unified sequence labeling model for emotion cause pair extraction. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 208–218. International Committee on Computational Linguistics.

Ying Chen, Wenjun Hou, Shoushan Li, Caicong Wu, and Xiaoqiang Zhang. 2020c. End-to-end emotion-cause pair extraction with graph convolutional network. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 198–207. International Committee on Computational Linguistics.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *COLING 2010, 23rd International Conference on Computational Linguistics*, pages 179–187.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186.

Zixiang Ding, Huihui He, Mengran Zhang, and Rui Xia. 2019. From independent prediction to reordered prediction: Integrating relative position and global label information to emotion cause identification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 6343–6350.

Chuang Fan, Hongyu Yan, Jiachen Du, Lin Gui, Lidong Bing, Min Yang, Ruifeng Xu, and Ruibin Mao. 2019. A knowledge regularized hierarchical approach for emotion cause analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5618–5628.

Kai Gao, Hua Xu, and Jiushuo Wang. 2015a. Emotion cause detection for chinese micro-blogs based on ecocc model. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 3–14.

Kai Gao, Hua Xu, and Jiushuo Wang. 2015b. A rule-based approach to emotion cause detection for chinese micro-blogs. *Expert Systems with Applications*, 42(9):4517–4528.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 152–165.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 347–352.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu, and Jiachen Du. 2017. A question answering approach to emotion cause extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1953–1602.

Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016. Event-driven emotion cause extraction with corpus construction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 1639–1649.

Lin Gui, Li Yuan, Ruifeng Xu, Bin Liu, Qin Lu, and Yu Zhou. 2014. Emotion cause detection with linguistic construction in chinese weibo text. In *Natural Language Processing and Chinese Computing - Third CCF Conference, NLPCC 2014*, volume 496 of *Communications in Computer and Information Science*, pages 457–464.

Guimin Hu, Guangming Lu, and Yi Zhao. 2021. FSS-GCN: A graph convolutional networks with fusion of semantic and structure for emotion cause analysis. *Knowl. Based Syst.*, 212:106584.

Guozhe Jin and Zhezhou Yu. 2021. A korean named entity recognition method using bi-lstm-crf and masked self-attention. *Comput. Speech Lang.*, 65:101134.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Sophia Yat Mei Lee, Ying Chen, and Chu-Ren Huang. 2010a. A text-driven rule-based system for emotion cause detection. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 45–53.

Sophia Yat Mei Lee, Ying Chen, Shoushan Li, and Chu-Ren Huang. 2010b. Emotion cause events: Corpus construction and analysis. In *Proceedings of the International Conference on Language Resources nd Evaluation, LREC 2010*.

Xiangju Li, Shi Feng, Daling Wang, and Yifei Zhang. 2019. Context-aware emotion cause analysis with multi-attention-based neural network. *Knowledge-Based Systems*, 174:205–218.

Xiangju Li, Kaisong Song, Shi Feng, Daling Wang, and Yifei Zhang. 2018. A co-attention neural network model for emotion cause analysis with emotional context awareness. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4752–4757.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 864–874. ACL.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring sequence-to-sequence learning in aspect term extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3538–3547.

Alena Neviarouskaya and Masaki Aono. 2013. Extracting causes of emotions from text. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 932–936. Asian Federation of Natural Language Processing / ACL.

Aaditya Singh, Shreeshail Hingane, Saim Wani, and Ashutosh Modi. 2021. An end-to-end network for emotion-cause pair extraction. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EACL 2021, Online, April 19, 2021*, pages 84–91. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Penghui Wei, Jiahao Zhao, and Wenji Mao. 2020. Effective inter-clause modeling for end-to-end emotion-cause pair extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3171–3181. Association for Computational Linguistics.

Sixing Wu, Fang Chen, Fangzhao Wu, Yongfeng Huang, and Xing Li. 2020. A multi-task learning neural network for emotion-cause pair extraction. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2212–2219. IOS Press.

Rui Xia and Zixiang Ding. 2019. Emotion-cause pair extraction: A new task to emotion analysis in texts. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1003–1012. Association for Computational Linguistics.

Rui Xia, Mengran Zhang, and Zixiang Ding. 2019. RTHN: A rnn-transformer hierarchical network for emotion cause extraction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5285–5291. ijcai.org.

Shuntaro Yada, Kazushi Ikeda, Keiichiro Hoashi, and Kyo Kageura. 2017. A bootstrap method for automatic rule acquisition on emotion cause extraction. In *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 414–421. IEEE Computer Society.

Xinyi Yu, Wenge Rong, Zhuo Zhang, Yuanxin Ouyang, and Zhang Xiong. 2019. Multiple level hierarchical network-based clause selection for emotion cause extraction. *IEEE Access*, 7:9071–9079.

On Commonsense Cues in BERT for Solving Commonsense Tasks

Leyang Cui$^{\triangle\dagger}$**, Sijie Cheng**‡**, Yu Wu**$^{\diamond}$**, Yue Zhang**$^{\dagger *}$
$^{\triangle}$ Zhejiang University
†School of Engineering, Westlake University
‡Fudan University $^{\diamond}$Microsoft Research Asia
cuileyang@westlake.edu.cn sjcheng20@fudan.edu.cn
Wu.Yu@microsoft.com yue.zhang@wias.org.cn

Abstract

BERT has been used for solving commonsense tasks such as CommonsenseQA. While prior research has found that BERT does contain commonsense information to some extent, there has been work showing that pre-trained models can rely on spurious associations (e.g., data bias) rather than key cues in solving sentiment classification and other problems. We quantitatively investigate the presence of structural commonsense cues in BERT when solving commonsense tasks, and the importance of such cues for the model prediction. Using two different measures, we find that BERT does use relevant knowledge for solving the task, and the presence of commonsense knowledge is positively correlated to the model accuracy.

1 Introduction

Pre-trained language models (Peters et al., 2018; Radford et al., 2019; Devlin et al., 2019; Liu et al., 2019b) give competitive results on a variety of NLP tasks (Zhou and Zhao, 2019; Joshi et al., 2019; Liu and Lapata, 2019; Cui et al., 2020). It has been shown that they can effectively capture syntactic features (Goldberg, 2019), semantic information (Liu et al., 2019a) and factual knowledge (Petroni et al., 2019), which provides support for the success in downstream tasks.

Recently, there has been some debate about whether commonsense knowledge can be learned by a language model trained on large corpora. While Davison et al. (2019), Bosselut et al. (2019) and Rajani et al. (2019) argue that pre-trained language models can directly identify commonsense facts, Lin et al. (2019) and Klein and Nabi (2019) believe that structured commonsense knowledge is not captured well.

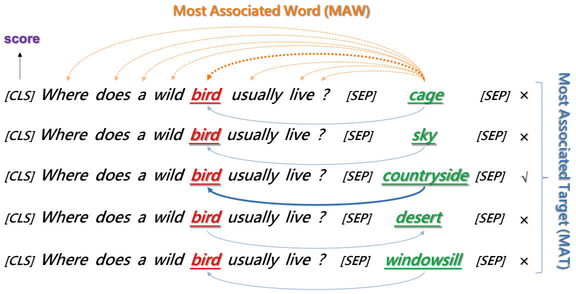

Figure 1: Two methods used to study structured commonsense knowledge in pre-trained Transformer. Commonsense link is drawn from the Target Concept (Answer Concept) to the Source Concept (Question Concept).

Pre-trained language models have achieved empirical success when fine-tuned on specific commonsense tasks such as COSMOS QA (Huang et al., 2019), SWAG (Zellers et al., 2018), and CommonsenseQA (Talmor et al., 2019). One possible reason of the high performance is that there exist *superficial cues* or *spurious associations* in the dataset, which enables models to answer questions without understanding the task (Niven and Kao, 2019; Yu et al., 2020; Kaushik et al., 2020). For example, a model can choose the spurious cue word "meadow" as a feature for positive reviews simply because "meadow" occurs frequently in positive documents. It remains an interesting research question whether commonsense knowledge plays a central role among statistical cues that BERT has when solving commonsense tasks. In other words, we are interested in investigating whether BERT solves commonsense tasks using commonsense knowledge.

We try to provide quantitative answers by mainly using the CommonsenseQA dataset, which asks a model to solve a multiple-choice problem. As shown in Figure 1, given a question and five candidate answers, a model should select one candidate answer as the output. The current state-of-the-art

---

[*] Corresponding Author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 683–693
August 1–6, 2021.

pre-trained language models solve the problem by representing the question jointly with each candidate answer (we call such a question-answer pair a *sentence* thereafter), and using pre-trained language models as the main encoder. Scoring of each sentence is based on a sentence-level hidden vector, and the candidate answer that corresponds to the highest-scored sentence is taken as the output.

We investigate the presence of commonsense cues in the BERT representation of a sentence by examining **commonsense links** from the **answer concepts** to its related contextualized **question concepts**. Figure 2 shows one example, where the question concept is "bird", and the correct answer is the answer concept connected through an ATLOCATION link in the CONCEPTNET knowledge graph. Such related concepts are *not* explicitly used in a BERT model for making prediction, and therefore its strength in the BERT representation is not necessarily optimized in task fine-tuning. We call such cues *structured commonsense*, which is a source of knowledge that we can explicitly measure. We take two methods for measuring structured commonsense in BERT, including directly measuring the attention weights (Clark et al., 2019) and measuring attribution scores by considering gradients (Mudrakarta et al., 2018).

We conduct two sets of experiments to quantitatively measure commonsense links in different situation. In the first set, we examine the presence of commonsense links directly in the BERT representation both before and after fine-tuning (Section 5). In the second set of experiments, we investigate the correlation between commonsense links with model predictions (Section 6). While the former can serve as a probing task for understanding commonsense learned by pre-training, the latter can serve as a means for understanding whether a model learns to make better use of commonsense knowledge through supervised fine-tuning.

Results suggest that BERT does have commonsense knowledge from pre-training, just as syntactic and word sense information. In addition, through fine-tuning, BERT relies more on commonsense cues in making prediction. The evidence is quantitatively demonstrated by stronger commonsense links in the representation, and a salient correlation between model predictions and commonsense link strengths, despite the fact that neither the answer concept nor the related question concept in a commonsense link is directly connected to the output layer. Interestingly, results also indicate that the stronger the structured commonsense knowledge is, the more accurate the model is. In addition to CommonsenseQA dataset, we observe similar phenomenon on Wikipedia and OMCS, demonstrating the generalization of our findings. To our knowledge, we are the first to investigate key statistical cues when BERT solves the CommonsenseQA task, providing several evidences that commonsense knowledge is indeed made use of. We release our code at `https://github.com/Nealcly/commonsense`.

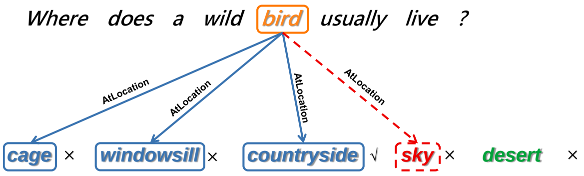

Figure 2: From CONCEPTNET to CommonsenseQA.

2 Related Work

There has been much recent work exploiting the underlying knowledge embedded in BERT representations. Peters et al. (2018) find that lower layers and higher layers in ELMo contain more syntactic and semantic information, respectively. Tenney et al. (2019), Liu et al. (2019a) and Jawahar et al. (2019) use probing models on hidden states to analyze linguistic information within pre-trained language models. Goldberg (2019) assess BERT's syntactic abilities by masking the verb, and comparing the prediction probability of the original verb with incorrect verbs. Our method is similar to Clark et al. (2019) and Htut et al. (2019), who focus on attention heads. The difference lies in that our primary goal is to investigate what information is learned and made use of when solving commonsense tasks. Therefore, our investigation is *task-centered*.

There has also been work investigating data bias and spurious associations. Gururangan et al. (2018) and Poliak et al. (2018) show that classifiers achieve accuracies around 69% on SNLI (Bowman et al., 2015) by using partial input. Kaushik et al. (2020) demonstrate BERT solve sentiment analysis and NLI by heavily relying on spurious associations. Our work is in line in investigating statistical cues. Different from the above investigations, we use probing methods to verify the presence and importance of the key feature, namely commonsense knowledge, in solving commonsense QA, rather than focusing on adversarial cases.

Commonsense reasoning is a challenging task in natural language processing. Traditional methods rely heavily on hand-crafted features (Rahman and Ng, 2012; Bailey et al., 2015) and external knowledge bases (Schüller, 2014). With recent advances in deep learning, pre-trained language models have been used as a powerful method for such tasks. Trinh and Le (2018) use a pre-trained language model to score candidate sentences for the Pronoun Disambiguation and Winograd Schema Challenge (Levesque et al., 2012). Klein and Nabi (2020) use a sentence-level loss to enhance commonsense knowledge in BERT. Mao et al. (2019) demonstrate that pre-trained language models fine-tuned on SWAG (Zellers et al., 2018) are able to provide commonsense grounding for story generation. For commonsense question answering, pre-trained language models with fine-tuning give the state-of-the-art performance (Zellers et al., 2018; Huang et al., 2019; Talmor et al., 2019). Though the above work show usefulness of BERT on comonsense tasks, little work has been done investigating the mechansim for BERT solving the tasks. Our work thus complements existing research in this aspect.

There is also a line of work leveraging CONCEPTNET to enhance model's commonsense reasoning ability. Lin et al. (2019) inject path information from question concepts to answer concepts to a model. Ye et al. (2019) use CONCEPTNET to construct pre-training dataset for BERT. Lv et al. (2019) extract evidence from CONCEPTNET and Wikipedia to build a relational graph for CommonsenseQA. We use CONCEPTNET for measuring commonsense knowledge in BERT.

3 Task and Model

We review the main experimental dataset CommonsenseQA (Section 3.1), before showing the structure of a state-of-the-art model (Section 3.2).

3.1 Dataset

CommonsenseQA (Talmor et al., 2019) is a multiple-choice question answering dataset constructed based on the knowledge graph CONCEPTNET (Speer et al., 2017), which is composed of a large set of triples taking the form $\langle$source concept, relation, target concept$\rangle$, such as $\langle$BIRD, ATLOCATION, COUNTRYSIDES$\rangle$. Given a source concept BIRD and the relation type ATLOCATION, there are three related target concepts CAGE, WINDOWSILL and COUNTRYSIDE in CONCEPTNET.

As shown in Figure 2, in the development of the CommonsenseQA dataset, crowd-workers are requested to generate question and candidate answers based on the *source concept* and three related target concepts in CONCEPTNET, respectively. Following Talmor et al. (2019), we call the source concept in the question as *question concept*, and the target concept in the answer as *answer concept*. Each question corresponds to only one correct *answer concept* among the three related CONCEPTNET target concepts. In addition, two more incorrect answer concepts are added, which do not correlate with the question concept in CONCEPTNET, resulting in 5 candidate answers for each question. We define *commonsene link* as the link from the answer concept to the question concept.

The CommonsenseQA dataset is designed to avoid salient bias in surface patterns. First, the lexical overlap between the correct answer and the question is similar to that between the question and incorrect candidates. Second, commonsense links are not superficial patterns that can be learned from training data. In particular, the percentage of answer-question-concept pairs in test examples that also exist in the gold training examples is 15.78%, which suggests that the main source of strong commonsense links, if observed, comes mainly from the pre-trained BERT model itself.

In order to analyze implicit structured commonsense knowledge, which is based on the link from the answer concept to the question concept, we filter out questions which do not contain explicit mentions to the *question concept* in its CONCEPTNET form (e.g. paraphrase). The resulting dataset CommonsenseQA* contains 74 fewer instances.

3.2 Model

We adopt the method of Talmor et al. (2019), using BERT (Devlin et al., 2019). In particular, given a question q and 5 candidate answers $a_1, ..., a_5$, we concatenate the question with each answer to obtain 5 question-answer pair sequences (i.e. sentences) $s_1, \ldots, s_5$, respectively. In each sentence, we use a special symbol `[CLS]` in the beginning, a symbol `[SEP]` between the question and the candidate answer, a symbol `[SEP]` in the end.

BERT uses L stacked Transformer layers (Vaswani et al., 2017) to encode each sentence. The last layer hidden state of the `[CLS]` token is used for linear scoring with softmax normalization. The candidate among $s_1, \ldots, s_5$ with the highest score is chosen as the prediction. More details of our implementation are shown in Appendix C.

4 Analysis Methods

As mentioned earlier, we analyze commonsense links using the attention weight (Clark et al., 2019) and the corresponding attribution score (Sundararajan et al., 2017; Mudrakarta et al., 2018). We report results in one random execution for each experiment. We additionally tried five runs for each experiments, and found that the result variation is small (Appendix B).

4.1 Attention Weights

Given a sentence, attention weights in Transformer can be viewed as the relative importance weight between each token and the other tokens when producing the next layer representation (Kovaleva et al., 2019; Vashishth et al., 2020). In particular, given a sequence of input vectors $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{|H|}]$, its self-attention representation uses each vector as a query to retrieve all context vectors in $\mathbf{H}$, yielding a matrix of attention weights $\alpha \in \mathbb{R}^{|H| \times |H|}$.

The value of α is computed using the scaled dot-product of the query vector of representation $\mathbf{Q} = \mathbf{W}^Q\mathbf{H}$ and the key vector of representation $\mathbf{K} = \mathbf{W}^K\mathbf{H}$, followed by softmax normalization

$$\alpha = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}), \tag{1}$$

where d_k is the dimension size of the key vector $\mathbf{K}$. $\alpha_{i,j}$ represents the attention strength from $\mathbf{h}_i$ to $\mathbf{h}_j$. For multi-head attention, $\mathbf{H}$ is linearly projected T times to find T sets of queries, keys, and values, where T is the number of heads. The attention operation of each head is performed in parallel, with the results being concatenated. We use $\alpha^{m,n}$ to denote the n-th attention head in the m-th layer. The attention weights $\alpha^{m,n}$ are used as a first measure of commonsense link strengths.

4.2 Attribution Scores

Kobayashi et al. (2020) point out that analyzing only attention weights can be insufficient for investigating the behavior of attention heads. As a supplement, gradient-based feature attribution methods have been studied to interpret the contribution of each input feature to the model prediction in back-propagation (Baehrens et al., 2010; Mudrakarta et al., 2018; Hao et al., 2020). Analysis of both attention weights and the corresponding attribution scores allows us to more comprehensively understand commonsense links in BERT.

We employ an attribution technique called *Integrated Gradients* (Sundararajan et al., 2017). Intuitively, integrated gradients works by simulating the process of pruning the specific attention head (from the original attention weight α to a zero vector α'), and calculating the integrated gradients in back-propagation. The attribution score directly reflects how much a change of attention weights affects model outputs. A higher attribution score represents more importance of the corresponding individual attention weight. Suppose that $F(x)$ represents the BERT model output for CommonsenseQA given an input x. The attribution of attention head $\alpha_t t \in [1, \ldots, T]$ in a Transformer layer can be computed by comparing with a set of baseline weights α':

$$Atr(\alpha^t) = (\alpha^t - \alpha'^t) \otimes \int_{x=0}^{1} \frac{\partial F(\alpha' + x(\alpha - \alpha'))}{\partial \alpha^t} dx \tag{2}$$

where $\otimes$ denotes element-wise multiplication, $\alpha = [\alpha^1, \ldots, \alpha^T]$. Intuitively, $F(\alpha' + x(\alpha - \alpha'))$ is closer to $F(\alpha')$ when x is closer to 0, and closer to α when x is closer to 1. Therefore, $\int_{x=0}^{1} \frac{\partial F(\alpha' + x(\alpha - \alpha'))}{\partial \alpha^t} dx$ gives the amortized gradient with all different x. $Atr(\alpha^t) \in \mathbb{R}^{n \times n}$ denotes the attribution score which corresponding to the attention weight α^t. $Atr(\alpha^t_{i,j})$ is represented for the interaction from token $\mathbf{h}_i$ to $\mathbf{h}_j$. We set the uninformative baseline α' as zero vector. Following Sundararajan et al. (2017), we approximate $Atr(\alpha^t)$ via a gradient summation function,

$$Atr(\alpha^t) ::= (\alpha^t - \alpha'^t) \otimes \sum_{i=1}^{s} \frac{\partial F(\alpha' + i/s(\alpha - \alpha'))}{\partial \alpha'^t} \times \frac{1}{s}, \tag{3}$$

where s is the number of approximation steps for computing integrated gradients. We set s to 20 based on the empirical results.

5 The Presence of Knowledge

We first conduct a set of experiments to investigate commonsense link strengths in BERT representations of question-answer pairs (i.e. sentences). Intuitively, if the link weight from the answer concept to the question concept is higher than those from the answer concept to other question words, then

	Statistics		**MAW Accuracy**				**Random**
			Max		**Avg**		
Dataset	# Instances	# Avg Length	BERT	BERT-FT	BERT	BERT-FT	
CommonsenseQA*	1,147	13.18	46.82	49.22	12.38	17.35	10.53
OMCS	37,895	7.63	88.48	89.14	37.82	39.52	24.11
Wikipedia	176,449	16.40	40.24	43.53	13.19	13.48	6.22

Table 1: Average and maximum MAW accuracies across three datasets. BERT-FT model denotes the BERT model with fine-tuning on CommonsenseQA training set.

Relation Type	**Max**	**Avg**	**L-H**	**# Ins**
Random	10.53	10.53	-	-
OVERALL	49.22	17.35	8-7	-
ATLOCATION	55.85	18.42	8-7	574
CAUSES	55.93	18.91	8-7	162
CAPABLEOF	47.88	14.71	8-1	104
ANTONYM	52.53	10.97	4-3	83
HASPREREQUISITE	54.15	18.93	9-8	41
HASSUBEVENT	55.29	18.74	9-0	34
DESIRES	40.00	7.92	8-1	27
CAUSESDESIRE	48.89	14.28	4-0	27
PARTOF	59.09	18.56	9-0	22
HASPROPERTY	54.00	15.12	9-1	20
MOTIVATEDBYGOAL	75.56	24.31	9-7	18
HASA	68.89	22.10	8-1	9
RELATEDTO	62.22	18.44	9-0	9

Table 2: The average and maximum MAW accuracies of BERT-FT for different commonsense relations. We exclude the relation types with frequencies of occurrence less than 9. L-H represents the best performing attention head for each relation.

we have evidence of BERT using commonsense cues according to CONCEPTNET. As mentioned earlier, rather than the question concept, the representation of the `[CLS]` token is directly connected to the output layer for candidate scoring. Hence there is no direct supervision signal from the output layer to the link weight during fine-tuning, and better prediction does not necessarily indicate strong commonsense links.

5.1 Probing Task

Without losing generality, we call both attention weights in Section 4.1 and attribution weights in Section 4.2 *link weight*. We evaluate link weights by calculating the **most associated word** (MAW), namely the question concept word that receives the maximum link weight from the answer concept among all question words. MAW is measured for each individual attention head in each layer.

Denote the hidden states of the whole question, question concept and answer concept as $[\mathbf{h}_1, \ldots, \mathbf{h}_{|q|}]$, $[\mathbf{h}_{b_s}, \ldots, \mathbf{h}_{e_s}]$ and $[\mathbf{h}_{b_t}, \ldots, \mathbf{h}_{e_t}]$, respectively. If the answer concept is composed of multiple tokens, we consider the link weight from the answer concept to the question token $\mathbf{h}_i$ ($i \in [1, |q|]$) as the mean of the link weights over all answer tokens $\alpha_i = \frac{1}{e_t - b_t}\sum_{j=b_t}^{e_t} \alpha_{j,i}$. For the n-th attention head in the m-th layer, if the question concept receives the maximum link weight from the answer concept (i.e., $\mu^{m,n} = \arg\max_i \alpha_i^{m,n}$, $\mu^{m,n} \in [b_s, e_s]$), we consider that this attention head gives the correct MAW.

We take two different measures of MAW accuracies, calculating the average accuracy among all attention heads, and the accuracy of the most-accurate head, respectively. Previous work probing syntactic information from attention head takes the second method (Clark et al., 2019; Htut et al., 2019). We additionally measure the average in order to comprehensively evaluate the prevalence of commonsense cues in BERT.

The average MAW accuracy is measured by:

$$acc^{avg} = \frac{\sum_{m=1}^{12}\sum_{n=1}^{12}\sum_{d=1}^{D} \mathbb{1}(\mu^{m,n} \in [b_s, e_s])}{12 \times 12 \times D}.$$

The maximum MAW accuracy is measured by:

$$acc^{max} = \max_{m=1}^{12}\max_{n=1}^{12} \frac{\sum_{d=1}^{D} \mathbb{1}(\mu^{m,n} \in [b_s, e_s])}{D},$$

where D represents the number of instances for evaluation.

In theory, if link weights for each attention head are randomly distributed, the average and maximum MAW accuracies should be both

$$acc^{baseline} = \frac{\sum_{d=1}^{D} \frac{e_s - b_s}{|q|}}{D},$$

which reflects the fact that the representation does not contain explicit correlation between the answer concept and its related question concept. In contrast, MAW accuracies significantly better than this baseline indicates that commonsense knowledge is contained in the representation.

5.2 Results

The results for off-the-shelf BERT (BERT) and a BERT model fine-tuned on CommonsenseQA

(BERT-FT) are shown in the first row of Table 1. First, looking at the original non-fine-tuned BERT, the *maximum* MAW accuracy of each layer significantly outperforms[1] the random baseline. This shows that commonsense links are a part of BERT representation of a sentence in general, just as syntactic (Goldberg, 2019) and semantic (Liu et al., 2019a) knowledge. Second, BERT-FT outperforms BERT in terms of both the average MAW accuracy and the maximum MAW accuracy, with a relatively large boost on the average MAW accuracy, which shows that structured commonsense features are enhanced by supervised training on commonsense tasks.

We explore the best performing attentions head for each relation type in Table 2, finding that certain attention heads capture specific commonsense relations. There is no single attention head that does well for all relation types, both with fine-tuning and without fine-tuning, which is similar to the previously finding for syntactic heads (Raganato and Tiedemann, 2018; Clark et al., 2019).

To further differentiate commonsense cues from superficial association, we measure the co-occurrence between each word in the question and answer concept in 1 million English Wikipedia documents. There is only 1.85% question concept word among the highest co-occurring words of each answer concepts, which partly shows that the strong commonsense links do not heavily rely on superficial pattern.

5.3 Additional Datasets.

Since this set of experiments concerns the representation only, we take additional two unlabeled corpora in addition to CommonsenseQA. In particular, we extract sentences from Open Mind Common Sense (OMCS) [2] and Wikipedia, if there existing one and only one source-target concept pair in this sentence, yielding two large-scale datasets. The detailed statistics are shown in Table 1. The results are consistent with the CommonsenseQA dataset, which shows the generation ability of our methods.

6 Co-relating Knowledge with Task

We further conduct a set of experiments to draw the correlation between commonsense links and model prediction. The goal is to investigate how BERT makes use of commonsense knowledge for making a prediction in the CommonsenseQA task. In particular, we compare the link weights across the five answer candidates for the same question, and find out the candidate that is the most associated with the relevant question concept. This candidate is called the **most associated target** (MAT). Correlations are drawn between MATs and the model prediction for each question. Intuitively, the more the MATs are correlated with the model predictions, the more evidence we have that the model makes use of commonsense cues in making prediction.

	Attention				Attribution	
	BERT-FT		**BERT-Probing**		**BERT-FT**	
Head	MAT	MAS	MAT	MAS	MAT	MAS
1	49.00	18.92	29.21	4.01	51.61	23.54
2	49.17	19.62	20.75	10.99	27.46	24.85
3	32.00	56.23	16.04	43.85	49.17	33.83
4	41.33	16.74	32.17	9.68	22.93	47.08
5	49.96	24.32	33.91	6.28	31.04	44.29
6	45.42	13.25	34.87	4.62	34.26	20.14
7	48.39	13.33	25.72	7.41	33.83	22.67
8	54.14	13.39	28.07	3.66	25.98	49.61
9	39.67	16.74	28.86	9.50	36.97	22.84
10	38.71	13.95	24.50	18.66	52.14	21.01
11	49.17	8.89	36.88	7.15	36.79	21.19
12	53.53	11.07	30.08	3.31	25.81	26.94
Avg	45.87	18.85	28.42	10.76	35.67	29.83

Table 3: MAT^{overlap} and MAS^{overlap} in the top layer.

Both attention weights and the corresponding attribution scores are used, because now we are considering model prediction, for which gradients play a role and can be measured. For all experiments, the trend of attribute scores is consistent with that measured using attention weights.

6.1 Probing Tasks

Formally, given a question q and 5 candidate answers $a_1, \ldots, a_5$, we make comparisons across five candidate sentences $s_1, \ldots, s_5$. In each candidate sentence, we calculate the link weight from the answer concept to the question concept according to CONCEPTNET. Denote the hidden states of the question concept and the answer concept as $[\mathbf{h}_{b_s}, \ldots, \mathbf{h}_{e_s}]$ and $[\mathbf{h}_{b_t}, \ldots, \mathbf{h}_{e_t}]$, respectively. The link weight of the answer-question-concept pair (α_{a2q}) is the average between each answer concept token and each question concept token

$$\alpha_{a2q} = \frac{\sum_{i=b_s}^{e_s} \sum_{j=b_t}^{e_t} \alpha_{j,i}}{(e_s - b_s)(e_t - b_t)}$$

Among the five candidates in each instance, we take the one with the highest α_{a2q} as the most associated target MAT, denoted as $s^{\text{MAT}} \in [1, 5]$.

---

[1] $p \leq 0.01$ using t-test; similar for subsequent mentions.

[2] Open Mind Common Sense (OMCS) corpus is the source corpus of ConceptNet.

(a) Measured by attention weights.

#H	#Ins	Model Acc.	#H	#Ins	Model Acc.
0	158	20.89	7	69	78.26
1	135	28.15	8	63	82.54
2	119	52.10	9	57	92.98
3	132	53.79	10	47	89.36
4	93	62.37	11	44	97.73
5	106	66.04	12	36	100.00
6	88	68.18	-	-	-

(b) Measured by attribution.

#H	#Ins	Model Acc.	#H	#Ins	Model Acc.
0	89	10.11	5	171	72.51
1	114	22.81	6	119	81.51
2	148	51.35	7	85	82.35
3	156	56.41	8	43	74.72
4	207	66.67	9	13	84.62

Table 4: The relationship between the MAT head count and the model prediction accuracy. #H denotes how many heads yield the correct MAT prediction. #Ins denotes the instance number.

As a baseline for MAT, we further define **most associated sentence** (MAS) as the candidate answer that has the maximum link weight from the answer concept to the `[CLS]` token among the five candidates. The reason is that gradients are back-propagated from the `[CLS]` token rather than the question concept or the answer concept. By comparing MAT and MAS, we can have useful information on whether MAT is an influencing factor for the model decision.

We measure the correlation between MAT ($s^{\text{MAT}} \in [1,5]$), the model prediction ($s^{model} \in [1,5]$) and the gold-standard answer ($s^{golden} \in [1,5]$) by using two metrics, including the overlapping rate between MATs and model predictions, and the accuracy of MATs.

The **overlapping rate of MATs** is defined as:

$$\text{MAT}^{overlap} = \frac{\sum_{d=1}^{D} \mathbb{1}(s_d^{\text{MAT}} = s_d^{model})}{D}$$

The **accuracy of MATs** is defined as the percentage of Mats that equals the gold answer:

$$\text{MAT}^{acc} = \frac{\sum_{d=1}^{D} \mathbb{1}(s_d^{\text{MAT}} = s_d^{golden})}{D}$$

Similar to MAW, MAT and MAS can be measured for each attention head, and we calculate the average and maximum values across different heads.

6.2 Commonsense Link and Model Output

We measure the MAT performance of BERT-FT, and a BERT model that is fine-tuned for the output layer only (BERT-probing). The latter is a linear probing model (Liu et al., 2019a). Intuitively, if the probing model can solve the commonsense task accurately, then the original non-fine-tuned BERT likely encodes the rich commonsense knowledge.

Table 3 shows the relative strengths of MATs and MASs according to the 12 attention heads in the top Transformer layer. First, for both models, the overlapping rates of MATs are significantly ($p \leq 0.01$) larger than that with MASs. This suggests that the link weight from the answer concept to the question concept is more closely-related to the model prediction as compared to the link weight from the answer concept to the `[CLS]` token, despite that model output scores are calculated on the `[CLS]` token. The results give strong evidence that commonsense cues from BERT are relied on for model decision. Second, when fine-tuned with training data, the model gives an even stronger correlation between MAT and the model prediction. This suggests that the model can *learn* to make use of commonsense cues for making prediction, which partly shows how a BERT model solves CommonsenseQA.

Figure 3 shows the overlapping rate between MAT and model prediction at each Transformer layer. Both the maximum and the average overlapping rates across the 12 layers are shown. The random overlapping rate of 20% is drawn as a reference. It can be seen from the figure that the maximum overlapping rate of BERT-probing is significantly larger than the random baseline, which shows that the model prediction is associated with the relevant structured commonsense cues. In addition, after fine-tuning, the BERT-FT model shows a tendency of weakened maximum MAT overlapping rate on lower Transformer layers and much strengthened MAT overlapping rate on higher layers, and in particular the top layer. The trend of MAT measured by attribution score is consistent with attention weights. This suggests that fine-tuned model relies more on the commonsense structure in the top layer for making prediction.

We compare the co-occurrence between question concepts and candidate answer concepts in 1 million English Wikipedia documents, and find that only 18.2% gold answers has the most co-occurrence with the question concept among 5 answer candidates, which is even lower than the random baseline (20%), showing that CommonsenseQA cannot be solved by solely relying on superficial patterns.

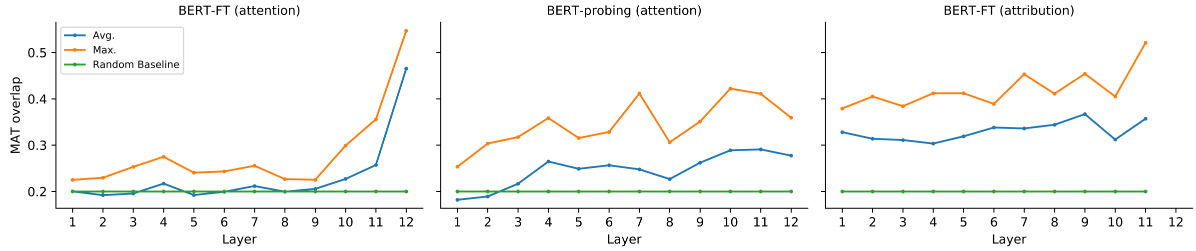

Figure 3: MAT^{overlap} across different layers.

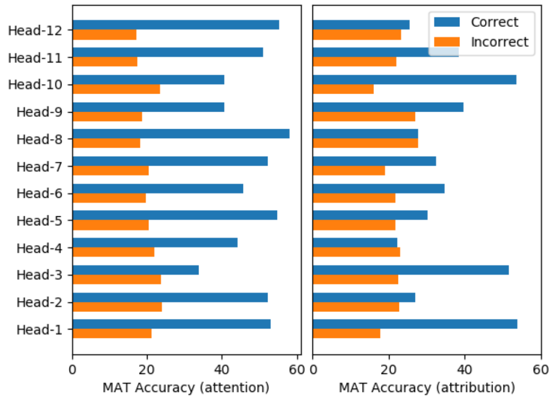

Figure 4: MAT^{acc} of each attention head in the top layer with correct and incorrect model predictions. "Red" and "Blue" indicate the model performance if attention head-n gives correct and incorrect prediction, respectively.

6.3 Commonsense Link and Model Accuracy

Table 4 shows the correlation between MAT accuracies and model prediction accuracies. Each row shows a different number of heads in the top layer for which the MAT corresponds to the correct answer candidate, together with the number of test instances for such cases, and the model prediction accuracy on the instances. There is an obvious trend where increased MAT accuracies correspond to increased model prediction accuracies, which shows that making use of structured commonsense cues leads to better model prediction.

Figure 4 shows the MAT accuracies of each attention head in the top layer for the test instances with correct and incorrect model predictions, respectively. The MAT accuracies of correctly predicted instances are larger than those of incorrectly predicted instances by a large margin. The finding is consistent with Table 4, which shows that structured commonsense cues are a key factor in BERT making the correct decision.

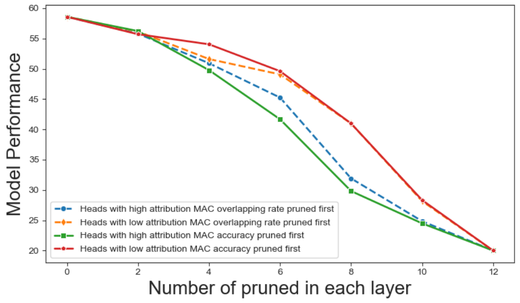

Figure 5: Model performance on the CommonsenseQA development set when different heads are pruned.

	BERT-FT			BERT-probing		
	MAT^{overlap}		Model	MAT^{overlap}		Model
L	Max	Avg	Acc	Max	Avg	Acc
12	54.14	45.87	58.59	36.88	28.42	39.23
11	46.56	26.65	56.50	37.66	27.11	35.48
10	37.40	27.86	53.36	39.84	28.50	33.74
9	34.61	24.01	51.53	30.08	24.76	32.52
8	31.82	21.39	49.35	25.81	21.53	33.57
7	31.73	24.40	48.74	37.05	24.04	32.96
6	31.56	23.64	45.95	31.21	24.02	32.00
5	34.44	25.01	44.99	33.39	24.03	32.43
4	44.73	34.13	40.28	41.06	27.67	33.83
3	44.20	32.48	37.58	25.81	21.02	21.88
2	23.71	19.47	26.68	23.63	20.74	20.40
1	23.45	19.50	23.02	20.58	18.81	19.27

Table 5: Performance of MAT^{overlap} across different layers. L-n represents adding the output classifier on the hidden state of layer-n. Our BERT-FT model (layer-11) gives 58.15% accuraies, which is slightly higher than the reported results of 55.57% on Lin et al. (2019). It achieves 58.59% on our dataset CommonsenseQA*.

We further evaluate the model performance after pruning specific heads. We sort all the attention heads in each layer according to their MAT performance by attribution scores, and then prune these heads in order. Following Michel et al. (2019), we replace the pruned head with zero vectors. Figure 5 shows the model performance on the development set. As the number of pruned heads increases, the model performance decreases, which conforms to intuition. In addition, the model performance drops much more rapidly when the attention heads with higher MAT performances are pruned first, which demonstrates that capturing commonsense features is crucial to strong model prediction.

6.4 Commonsense Link and BERT Layer

We further investigate two specific questions on the commonsense knowledge usage. First, which layer does BERT rely on the most for making its decision. Second, does the commonsense knowledge that BERT uses come more from pre-training or fine-tuning. We compare 12 model variations by connecting the output layer on each of the Transformer layer, respectively. Table 5 shows the model accuracies and the MAT overlapping rates. First, BERT-probing gives the best performance when prediction is made on the top layer, and the accuracy generally decreases as the layer moves to the bottom. This indicates that relevant commonsense knowledge is more heavily distributed towards higher layers during pre-training. Our experimental settings here are the same as the probing task for syntactic information by Liu et al. (2019a), who find that syntactic information is distributed more heavily towards lower BERT layers.

With fine-tuning, we observe stronger improvements of both model accuracies and MAT overlaps on higher layers when comparing BERT-FT and BERT-probing. This demonstrates that commonsense knowledge on higher layers is more useful to the CommonsenseQA task. Interestingly, comparing layer 11 and layer 10, the model accuracy after fine-tuning is similar, but the MAT overlap of layer 11 is significantly larger. This can suggest that the structured commonsense knowledge that we probe attributes only partly to the overall useful knowledge for CommonsenseQA.

7 Conclusion

We conducted quantitative analysis to investigate how BERT solves the CommonsenseQA task, aiming to gain evidence on the key source of information involved in the disambiguation process. Empirical results demonstrated that BERT encodes structured commonsense knowledge, and is able to leverage such cues on the downstream CommonsenseQA task. Our analysis has further revealed that with fine-tuning, BERT learns to make better use of commonsense features on higher layers. These suggest that BERT can learn to make use of truly relevant commonsense cues rather than superficial patterns for CommonsenseQA.

References

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831.

Daniel Bailey, Amelia Harrison, Yuliya Lierler, Vladimir Lifschitz, and Julian Michael. 2015. The winograd schema challenge and reasoning about correlation. In *AAAI Spring Symposia*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. Mutual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Conference of the Association for Computational Linguistics*. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-2019*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2020. Self-attention attribution: Interpreting information interactions inside transformer.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in bert track syntactic dependencies?

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401, Hong Kong, China. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *EMNLP-IJCNLP-19*, Hong Kong, China. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Tassilo Klein and Moin Nabi. 2019. Attention is (not) all you need for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4831–4836, Florence, Italy. Association for Computational Linguistics.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention module is not only a weight: Analyzing transformers with vector norms. *ArXiv*, abs/2004.10102.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012*, pages 552–561.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *NAACL-2019*, Minneapolis, Minnesota. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *EMNLP-IJCNLP-2019*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2019. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering.

Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley, and Garrison Cottrell. 2019. Improving neural story generation by targeted common sense grounding. In *Proceedings of EMNLP-IJCNLP-2019*, pages 5988–5993, Hong Kong, China. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 14014–14024. Curran Associates, Inc.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1896–1906, Melbourne, Australia. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *acl-19*, Florence, Italy. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-2018*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP-2019*, Hong Kong, China. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789, Jeju Island, Korea. Association for Computational Linguistics.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Peter Schüller. 2014. Tackling winograd schemas by formalizing relevance theory in knowledge graphs. In *KR*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL-2019*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. *ArXiv*, abs/1806.02847.

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2020. Attention interpretability across {nlp} tasks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Zhi-Xiu Ye, Qian Chen, Wen Wang, and Zhen-Hua Ling. 2019. Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP-18*.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *ACL-19*, Florence, Italy. Association for Computational Linguistics.

Weakly Supervised Pre-Training for Multi-Hop Retriever

Yeon Seonwoo[†], **Sang-Woo Lee**[‡§], **Ji-Hoon Kim**[‡§], **Jung-Woo Ha**[‡§], **Alice Oh**[†]
[†]KAIST
[‡]NAVER AI Lab, [§]NAVER Clova
yeon.seonwoo@kaist.ac.kr
{sang.woo.lee,genesis.kim,jungwoo.ha}@navercorp.com
alice.oh@kaist.edu

Abstract

In multi-hop QA, answering complex questions entails iterative document retrieval for finding the missing entity of the question. The main steps of this process are sub-question detection, document retrieval for the sub-question, and generation of a new query for the final document retrieval. However, building a dataset that contains complex questions with sub-questions and their corresponding documents requires costly human annotation. To address the issue, we propose a new method for weakly supervised multi-hop retriever pre-training without human efforts. Our method includes 1) a pre-training task for generating vector representations of complex questions, 2) a scalable data generation method that produces the nested structure of question and sub-question as weak supervision for pre-training, and 3) a pre-training model structure based on dense encoders. We conduct experiments to compare the performance of our pre-trained retriever with several state-of-the-art models on end-to-end multi-hop QA as well as document retrieval. The experimental results show that our pre-trained retriever is effective and also robust on limited data and computational resources.

1 Introduction

Multi-hop QA is the task of answering complex questions that requires reasoning across multiple documents (Nogueira and Cho, 2017; Nie et al., 2019; Sun et al., 2019; Fang et al., 2020; Zhao et al., 2020). The core components of multi-hop reasoning are identifying the missing entity in the question and generating a new query with the missing entity. Figure 1 shows an example of the reasoning process in multi-hop QA. In the example, the missing entity, which we call bridge entity, of the question is *"Jupiter"*. To answer the question, the correct document for the sub-question "the largest planet in the Solar System," should be retrieved. Supervised training of the multi-hop QA models for these intermediate reasoning steps requires a dataset of complex questions, sub-questions, and their corresponding documents. However, building such dataset requires costly human annotation and cannot be done at scale (Min et al., 2019; Wolfson et al., 2020).

When there is limited annotated supervision signal, weakly supervised pre-training can be a solution (Devlin et al., 2019; Liu et al., 2019) which has shown effectiveness in open-domain QA (Lee et al., 2019; Guu et al., 2020). Unlike open-domain QA, it is not trivial to apply a pre-training method to multi-hop QA due to the complexity in generating weak supervision data. In open-domain QA, weak supervision is generated by selecting a document from a corpus and extracting a sentence from the document. The sentence becomes a pseudo question, and the document becomes a pseudo supporting document to be predicted by retrievers. This two-step process for weak supervision cannot be directly applied in multi-hop QA since each multi-hop question refers to multiple documents.

In this paper, we propose a novel weakly supervised pre-training method for multi-hop retriever, **LOUVRE** (**L**earning fr**O**m m**U**lti-hop **V**ariation of document **RE**lations). Our method contains three core elements: 1) a pre-training task, 2) a scalable method to generate pre-training data with weak supervision, and 3) a model to pre-train a retriever for multi-hop QA. Specifically, we define a task for pre-training, "Next Document Prediction" (NDP), which is to retrieve documents for sub-questions. We then propose "Bridge Entity Re-Phrasing" to generate the pre-training data. "Bridge Entity Re-Phrasing" generates complex questions that contain sub-questions of the bridge entities and their corresponding documents. To generate a complex question using a bridge entity without human effort, we

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 694–704
August 1–6, 2021.

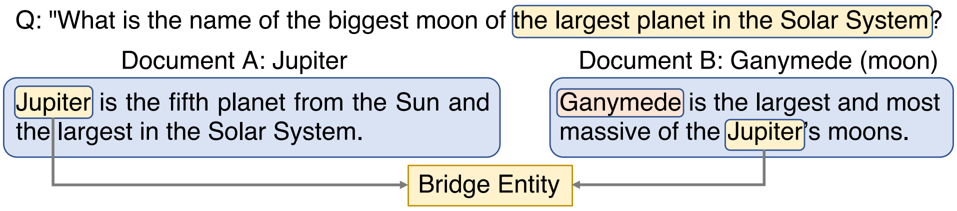

Figure 1: An example of chain reasoning in multi-hop QA. Answering the question requires finding *"the largest planet in the Solar System"* which is the bridge entity *"Jupiter."* With the retrieved document A, the question has enough information to retrieve the correct answer in document B.

use two documents connected by Wikipedia hyperlinks. The hyperlinked entity becomes the bridge entity, and the introductory phrase of the entity becomes the sub-question in the complex question. This approach enables our weak supervision data generation to be scalable, as shown in Figure 2. We use a dense retriever consisting of a question encoder and a document encoder for the pre-trained model structure (Karpukhin et al., 2020; Xiong et al., 2021). The two encoders calculate vector representations of questions and documents. Document retrieval is performed by comparing the vectors with MIPS (maximum inner product search).

Pre-training multi-hop retriever with our weak supervision method brings three benefits: significant performance improvement, robustness on few-shot settings, and computational efficiency. We evaluate our weakly supervised pre-trained retriever with two types of experiments on HOTPOTQA dataset: supporting documents prediction and end-to-end multi-hop QA. In both experiments, LOUVRE outperforms previous multi-hop retrievers. Also, we fine-tune LOUVRE on 1% of training data and show that the performance of LOUVRE is comparable to the baselines. We evaluate the performance of LOUVRE according to the computational efficiency. The results show that LOUVRE requires less inference time than baselines.

Contributions of this paper are as follows: 1) we propose a novel scalable weakly supervised pre-training method for multi-hop retrievers, 2) we provide the implementation of LOUVRE and the pre-trained checkpoint publicly available [1], 3) we show the effect of our pre-training method in multi-hop QA with various experimental results.

2 Related Work

Distant Supervision in Open-Domain QA: Many open-domain QA datasets only provide question-answer pairs; some also provide weakly annotated supporting documents, but they are predicted by simple heuristics (Joshi et al., 2017; Berant et al., 2013). Document retrieval has suffered from insufficient strong supervision issues. Hence, document retrieval has suffered from lack of strong supervision. To resolve this issue, Karpukhin et al. (2020) use a document retrieved by TF-IDF as the supporting document of the given question. Weak supervision is also an effective method in the distant supervision setting of open-domain QA. Lee et al. (2019) use ICT (inverse cloze task) to generate pseudo question-document pairs and pre-train their retriever. They select documents from Wikipedia and extract sentences from the documents. The selected sentence-document pairs become pseudo-question-document pairs. Guu et al. (2020) propose a pre-training method for a language model that uses knowledge retriever (document retriever). They train the knowledge retriever only with the language modeling loss without using any supervision signal of supporting documents. Although pre-training methods show effectiveness in open-domain QA, they are limited to single-hop retrievers.

Multi-Hop QA: To overcome the lack of supervision signal in multi-hop QA, weak supervision methods have been proposed. Qi et al. (2019) propose a sub-question generation method. They use heuristically generated pseudo-questions as supervision for the question generation model. Perez et al. (2020) generate weak supervision for question decomposition by mapping a complex question to multiple single-hop questions in existing QA datasets. They use complex questions in HOTPOTQA (Yang et al., 2018) and single-hop questions in SQuAD 2.0 (Rajpurkar et al., 2018). Another method to train multi-hop QA models without human annotated datasets is by taking two simple questions and generating a complex question (Pan et al., 2020). They generate complex ques-

---

[1] `https://github.com/yeonsw/LOUVRE`

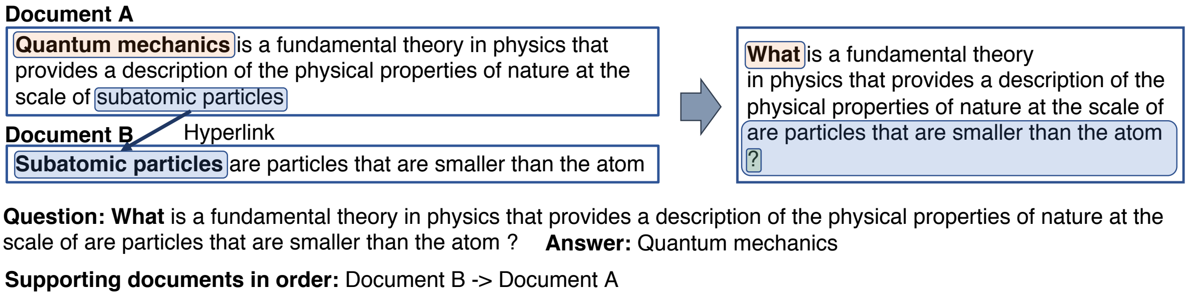

Figure 2: Proposed pre-training data generation process. Two documents connected by Wikipedia hyperlink are selected. In "Bridge Entity Re-Phrasing" process, document B which describes the entity "subatomic particles" is used to re-phrase the entity in document A. After replacing the answer entity, "Quantum mechanics", the complex question and its corresponding supporting documents are generated.

tions with GPT-2 (Radford et al., 2019) fine-tuned on SQuAD1.1. Our work improves upon previous research by providing a more general method that leverages a large open corpus with retriever pre-training.

3 Method

We propose an effective and scalable pre-training method that provides weak supervision of the complex questions with sub-questions and their corresponding supporting documents.

3.1 Next Document Prediction

We propose the "Next Document Prediction" (NDP) task for pre-training. NDP refers to the process of recurrent document retrieval used in (Qi et al., 2019; Asai et al., 2020; Xiong et al., 2021). We apply the common definitions in the existing studies to our "Next Document Prediction" task. We define NDP as the task that predicts documents in the reasoning sequence $[d_1, ..., d_n]$ recurrently as follows:

$$d_k = \text{Retriever}(q, D_{k-1}), \tag{1}$$

where q is a question, d_k is a predicted document at step k, and D_{k-1} is a set of documents retrieved in the previous steps, $\{d_1, ..., d_{k-1}\}$.

3.2 Bridge Entity Re-Phrasing

Our pre-training requires a dataset of questions, sub-questions, and their corresponding reasoning chains (i.e., a sequence of documents). We propose "Bridge Entity Re-Phrasing" for generating this pre-training dataset. "Bridge Entity Re-Phrasing" takes two steps: entity selection and re-phrasing. Figure 2 provides an overview of our data generation process. We provide the detailed description of "Bridge Entity Re-Phrasing" in the following paragraphs.

The "Bridge Entity Re-Phrasing" process requires informative entities and the description of the entity. We assume that an entity with a Wikipedia hyperlink is an informative entity. Also, hyperlink entities often have Wikipedia articles describing the entities. The hyperlink entity becomes the bridge entity. In Figure 2, document A and document B are connected with the bridge entity, *"subatomic particles"*. We re-phrase the selected entity with the first line of the document. In Figure 2, *"subatomic particles"* is re-phrased with the first line in document B. When the bridge entity appears in the question, multi-hop retrievers easily find the bridge document using only the word. To prevent this issue, we remove the bridge entity from document B. The generated document becomes the document to be used for question-answer pair generation.

Generating a question-answer pair from a single document has been studied by pre-training research in open-domain QA (Lee et al., 2019; Guu et al., 2020). We extend their work to generate questions, reasoning chains, and answers. We randomly select an entity from the merged document and replace the entity with the word *"what"*. In Figure 2, "Quantum mechanics" is the entity word and replaced with "what." The new sentence becomes a pseudo-question, and the replaced entity becomes the answer. Since document B contains the bridge entity, the pseudo-question reasoning chain becomes $[\text{document B}, \text{document A}]$.

3.3 Model Architecture

Model structure for our pre-training is subject to two requirements: general model structure and recurrent retrieval. We use multi-hop dense retriever

(Xiong et al., 2021) which meets the two requirements and is based on the DPR (dense passage retriever) (Karpukhin et al., 2020). DPR consists of a question encoder E_Q and a document encoder E_D, both of which are based on RoBERTa-base. Documents are retrieved by MIPS (maximum inner product search) with similarity between the question vectors and the document vectors as follows:

$$\mathrm{sim}(q, d) = E_Q(q)^\intercal E_D(d). \tag{2}$$

MDR retrieves documents recurrently by taking the previously retrieved documents as input. MDR concatenates the question q and the retrieved documents $\{d_1, ..., d_{k-1}\}$ and calculates a question vector for k-th step as follows:

$$d_k = \mathrm{argmax}_d(E_Q(q, d_1, ..., d_{k-1})^\intercal E_D(d)), \tag{3}$$

where d is a document in the corpus.

We train the dense encoder to assign the highest probability for the ground truth document among the documents in the huge corpus. The loss function for our pre-training is as follows:

$$L_{\mathrm{NBP}}(q_k, d_k) = \\ - \log \frac{e^{\mathrm{sim}(q_k, d_k)}}{e^{\mathrm{sim}(q_k, d_k)} + \sum_{d \in \mathrm{neg}(d_k)} e^{\mathrm{sim}(q_k, d)}}, \tag{4}$$

where q_k is a concatenation of q and D_{k-1}, and $\mathrm{neg}(d_k)$ is a set of documents excluding d_k. Since computing the softmax over the whole corpus is computationally expensive, we use in-batch negatives for $\mathrm{neg}(d_k)$ (Karpukhin et al., 2020).

4 Experimental Setup

4.1 Pre-Training Details

We generate our pre-training data from 5,233,329 Wikipedia articles provided by Yang et al. (2018). We select all sentences that contain at least one hyperlinked entity to generate pseudo questions and randomly select "answer" entities from the sentences [2]. Our data generation process builds 13.9 million question-document-answer triples. We pre-train our dense retriever with a batch size of 256 for 200K+ steps. We use Adam with a warm-up ratio of 0.1 and set the learning rate to 2×10^{-5}. We use a machine with eight V100 (32G) GPUs.

	TF-IDF	Wiki	RR	Eff
LOUVRE				
- eff	✓	✓		✓
- Wiki		✓		
- reranking			✓	
- reranking-Wiki		✓	✓	

Table 1: The five variations of LOUVRE. Each column represents sparse retrieval (TF-IDF), Wikipedia hyperlinks (Wiki), reranking (RR), and efficient fine-tuning (Eff). "Eff" represents whether the model uses efficient hyper-parameter setting, a batch size of 32 and a number of epochs of 15.

4.2 Fine-Tuning Details

We use TF-IDF negatives in addition to in-batch negatives for fine-tuning as in Karpukhin et al. (2020); Xiong et al. (2021). We set the number of TF-IDF negatives to 2. We use the Adam optimizer with a warm-up rate of 0.1 and set the learning rate to 2×10^{-5}. We set the batch size to 32, the number of epochs to 15. To achieve better performance, we additionally fine-tune our model with another hyper-parameter setting: a batch size of 150 and a longer training time of 50 epochs.

4.3 Tasks

Supporting Document Prediction: In this task, retrievers and rerankers predict supporting documents for each question in HOTPOTQA dataset (Yang et al., 2018). The models predict possible combinations of supporting documents. Formally, when a question, q_i, has been taken as input of the model, the models yield a ranked list of K-sets of documents, $L_i = [\{d_a, d_b\}]_{n=1}^{K}$. Each $\{d_a, d_b\}$ is a pair of candidate supporting documents. In HOTPOTQA, the number of supporting documents is fixed to 2. We use the 5 million Wikipedia articles as the knowledge source.

End-to-End Multi-Hop QA: We evaluate the supporting facts prediction performance and the answer prediction performance of LOUVRE on HOTPOTQA full wiki setting (Yang et al., 2018).

4.4 Multi-Hop Retrieval Strategy

We propose five variations of LOUVRE based on existing multi-hop retrieval strategies. Multi-hop document retrievers leverage three strategies for performance improvement and computational efficiency: sparse retrieval methods such as TF-IDF (Nie et al., 2019), Wikipedia hyperlinks (Asai et al.,

---

[2] We use spaCy for entity recognition

2020), and reranking (Xiong et al., 2021). Sparse retrieval methods select a small number of candidate documents relevant to the given question and are used to narrow down the search space of dense retrievers. We use TF-IDF and keyword matching as Nie et al. (2019) to retrieve 200 candidate documents. Existing multi-hop retrievers select reasoning paths (document chains) from documents connected with Wikipedia hyperlinks. We iteratively select the next-hop documents from the documents connected with the previously retrieved documents. Rerankers take the candidate reasoning paths (pairs of documents) from the retriever and predict the most probable reasoning path. We use the reranker proposed by Xiong et al. (2021). Table 1 shows the detailed information of these five variations.

4.5 Baselines

We compare our model to two types of multi-hop retrievers with and without reranking. For retrievers, we use TF-IDF, DPR (Karpukhin et al., 2020), Cognitive Graph (Ding et al., 2019), GOLDEN Retriever (Qi et al., 2019), and MDR (Xiong et al., 2021). For rerankers, we use SemanticRetrievalMRS (Nie et al., 2019), PathRetriever (Asai et al., 2020), MDR-reranking (Xiong et al., 2021), and HopRetriever (Li et al., 2020). We describe detailed experimental settings of each baseline in Appendix A.

4.6 Metric

We use five evaluation metrics: EM, F1, R@K, PathR@K, and AR@K. EM and F1 measure answer prediction and supporting fact prediction performance of multi-hop QA models (Yang et al., 2018). In addition to R@K, which measures the performance of supporting document prediction, we use another metric PathR@K to evaluate how well the retriever predicts the entire set of supporting documents. Since the readers predict answers by reading each path, PathR@K is a more appropriate estimate of answer prediction. The definitions of R@K and PathR@K are:

$$\begin{aligned} \text{R@K} &= \mathbb{1}(G \subseteq D) \\ \text{PathR@K} &= \mathbb{1}((G = P_1) \vee ... \vee (G = P_K)), \end{aligned} \tag{5}$$

where $G = \{g_i, g_j\}$ is the set of ground truth supporting documents, D is the set of retrieved documents, and $P_i = \{d_a, d_b\}$ is a reasoning path ranked at i.

In our experimental setting, D is set to all documents in $\bigcup_{i=1}^{K/2} P_i$. AR@K measures the percentage of predictions that at least one passage in the top K predicted paths contains the answer text.

5 Results & Discussion

LOUVRE overcomes limited supervision in multi-hop QA with our weakly supervision data. Training with additional data brings progress in three ways: overall retrieval performance improvement, 2) robustness on a few-shot setting, and 3) overall improvement in the end-to-end multi-hop QA. We verify these improvements on the two tasks described in section 4: supporting document prediction and end-to-end multi-hop QA.

5.1 Supporting Document Prediction

In this experiment, we demonstrate the efficacy of LOUVRE with document retrieval experiments. First, we show the performance gain that comes from using our pre-trained model. Then, we show that the result becomes more significant in few-shot settings.

Effect of Our Pre-Training: We compare LOUVRE with MDR which is a multi-hop retriever fine-tuned on RoBERTa-base. We use the same fine-tuning method as MDR but initialize the parameters with LOUVRE. Table 2 shows the results. LOUVRE achieves 1.1% absolute performance improvement than when using RoBERTa (65.9); PathR@1 of LOUVRE is 67.0. Also, LOUVRE outperforms MDR in other evaluation metrics. In reranking experiments, we use the same reranking model as MDR-rerank. The only difference between LOUVRE-rerank and MDR-rerank is the parameter initialization method in the fine-tuning step same as the retriever experiment. These results show that our pre-training method is effective even after reranking; PathR@1 of LOUVRE-rerank is 83.2, and PathR@1 of MDR-rerank is 81.2.

Weak Supervision and Training Time: LOUVRE's pre-training method uses additional training with the multi-hop weak supervision dataset and results in the performance improvement shown above. To verify that the performance gap between RoBERTa and LOUVRE is not from the additional training time that LOUVRE uses in pre-training, we train RoBERTa with much a longer training time, 50 epochs, and compare with LOUVRE.

In Figure 3, we show the performance of RoBERTa fine-tuned for $\{2, 5, 15, 50\}$ epochs and

		Wiki link	PathR@1	R@10	R@20	AR@1
Retriever	TF-IDF		9.8	27.6	35.1	43.4
	DPR		25.2	45.4	52.1	-
	Cognitive Graph		57.8	-	-	76.0
	GoldEnRetriever		-	75.4	-	-
	MDR		65.9	77.5	80.2	75.4
	LOUVRE-eff (1%)	✓	53.5	75.5	80.0	72.3
	LOUVRE-eff	✓	65.3	80.4	**83.0**	**78.9**
	LOUVRE		67.0	77.8	80.3	76.3
	LOUVRE-Wiki	✓	**69.5**	**80.7**	82.4	77.8
Reranker	SemanticRetrievalMRS	✓	63.9	81.7	82.1	77.9
	PathRetriever	✓	75.7	82.4	-	87.5
	MDR-rerank		81.2	86.4	86.6	88.2
	HopRetriever (w/o Wiki link)		66.2	78.8	-	76.3
	HopRetriever	✓	82.5	88.6	-	86.8
	LOUVRE-rerank		83.2	89.1	89.7	**90.0**
	LOUVRE-rerank-Wiki	✓	**83.5**	**89.5**	**90.1**	89.8

Table 2: Performance of retrievers. LOUVRE outperforms baseline retrievers and rerankers. Wiki link denotes whether the model uses Wikipedia hyperlinks.

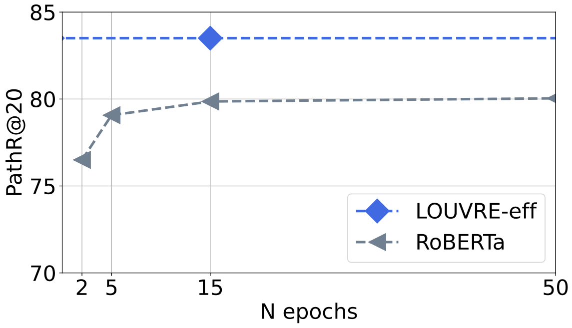

Figure 3: Comparison of LOUVRE-eff fine-tuned for 15 epochs with RoBERTa fine-tuned for 2 to 50 epochs. The performance of RoBERTa flattens at 15 epochs.

	R		PathR	
	@10	@20	@8	@20
LOUVRE-eff	**80.4**	**83.0**	**81.9**	**83.5**
- 1% train data	75.5	80.0	77.1	81.5
- zeroshot	44.8	55.8	48.3	59.0
- w/o pre-training	75.7	78.5	76.9	79.8
DPR-zeroshot	39.0	51.6	42.8	56.6

Table 3: Supporting document prediction performance of retrievers.

the performance of LOUVRE fine-tuned for 15 epochs. The performance of RoBERTa stabilizes at approximately 80% in terms of PathR@20 after 15 epochs. This result shows that the main factor of the performance improvement from our pre-training method is not merely from longer training time but the unique information for multi-hop retrievers provided by our weak supervision.

Retrieval Performance of Variations of LOUVRE-eff: Table 2 shows the effect of using LOUVRE. Similar results are observed in LOUVRE-eff. Table 3 shows the same experiments as Table 2 but in efficient fine-tuning setting, a small batch size and a short train time. We compare LOUVRE-eff with the retrieval performance of LOUVRE-eff without our pre-training method, which is fine-tuned on RoBERTa. Applying our pre-training method increases the retrieval performance by 4.7% point (R@10); R@10 of LOUVRE-eff is 80.4 and R@10 of LOUVRE-eff without pre-training is 75.7. Taking the results in Table 2 (the performance gain from our method with a big batch size/train epochs is 1.1) and the results in Table 3, we see the performance gain increases as there is more limitation on computation time.

Robustness on Few-shot Settings: Pre-training alleviates the model's drastic performance drop when the number of training data is insufficient. We demonstrate the robustness of LOUVRE on few-shot settings with different sizes of train data. We

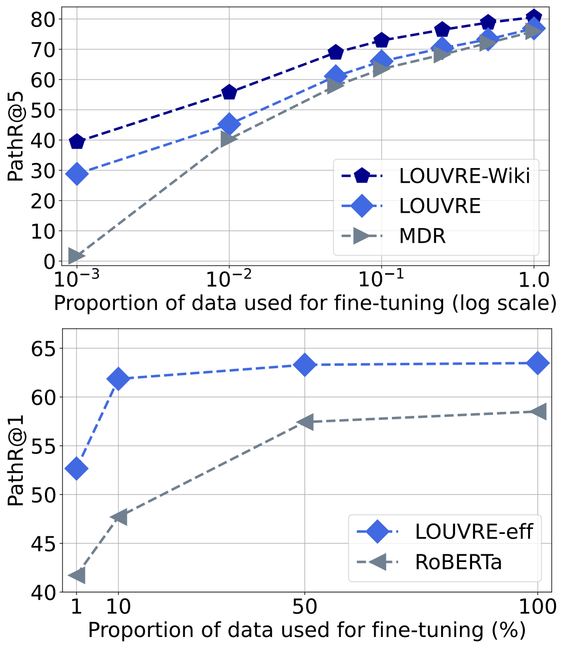

Figure 4: Retrieval performance of LOUVRE, LOUVRE-eff, and RoBERTa. The only difference between each model in the comparison models, (LOUVRE and MDR) and (LOUVRE-eff and RoBERTa), is the pre-trained model being used for parameter initialization.

fine-tune LOUVRE and MDR on a small portion of the HotpotQA train set within 0.1% to 100%. Figure 4 shows that the performance gap between LOUVRE and MDR increases as the size of train data decreases. When we use 0.1% of HotpotQA train data, almost 30% of LOUVRE's predictions contain correct supporting documents; the performance of MDR with the same amount of train data is close to 0. We conduct the same experiment with LOUVRE-Wiki and verify that using Wikipedia hyperlinks improves the robustness on a few-shot setting by 10.5% point in terms of PathR@5 when there is only 0.1% train data.

We conduct the same experiment with a small batch size of 32 and 10 epochs. Figure 4 illustrates the retrieval performance of LOUVRE-eff and LOUVRE-eff without our pre-training (RoBERTa) depending on the proportion of the data used for fine-tuning. It is worth noting that LOUVRE-eff fine-tuned with 10% data outperforms RoBERTa with 100% and shows little performance degradation compared to fine-tuning with 100%. We report the detailed results of LOUVRE-eff (1%) in Table 2. Table 2 shows that LOUVRE-eff (1%) achieves comparable performance to MDR trained on full data with a larger batch size and a longer train time; R@10 of LOUVRE-eff (1%) is 2.0% point lower than R@10 of MDR, which is 97% of the performance of MDR.

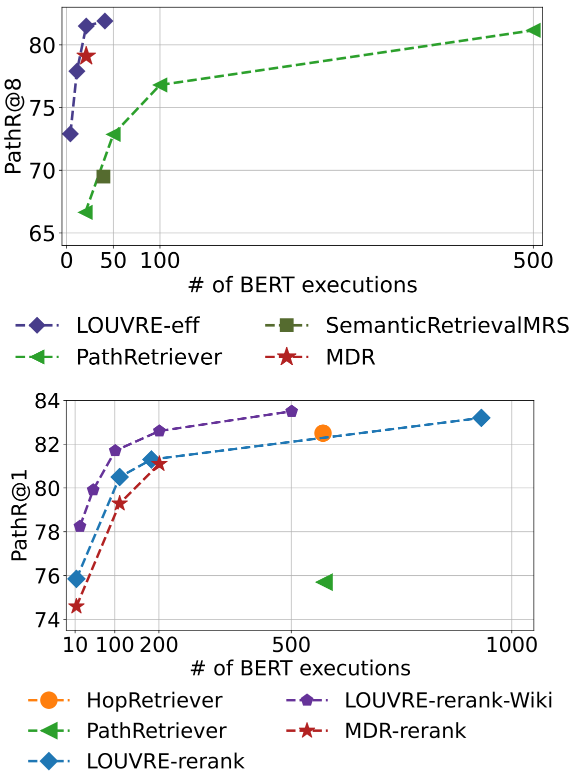

Figure 5: The retrieval performance of baselines and their computational efficiency. The beam sizes used for each model are as follows: $[3, 10, 20, 40]$ for LOUVRE-eff, $[3, 10, 13, 30]$ for LOUVRE-rerank, $[10, 10, 50, 100, 100]$ for LOUVRE-rerank-wiki, and $[3, 10, 100]$ for MDR-rerank. We use the input size of the reranker of each model as follows: $[9, 100, 169, 900]$ for LOUVRE-rerank, $[10, 40, 50, 100, 400]$ for LOUVRE-rerank-wiki, and $[9, 100, 100]$ for MDR. We evaluate PathRetriever with different number of initial n-documents retrieved by TF-IDF; $n \in \{20, 50, 100, 500\}$.

Furthermore, we evaluate the zero-shot performance of LOUVRE-eff and compare to DPR not fine-tuned on HotpotQA. To adapt DPR to the multi-hop retrieval task, we replace encoders in LOUVRE-eff with DPR encoders. We report this result in Table 3. In Table 3, LOUVRE-zeroshot achieves higher performance (R@10: 44.8 and R@20: 55.8) than DPR-zeroshot (R@10: 39.0 and R@20: 51.6).

Computational Efficiency: We compare the inference time of baselines and LOUVRE, with the number of BERT-base executions needed for each question. We exclude the inference time for document indexing which can be done a pri-

	Answer		Support		Joint	
	EM	F1	EM	F1	EM	F1
SemanticRetrievalMRS	45.32	57.34	38.67	70.83	25.14	47.60
Transformer-XH	51.60	64.07	40.91	71.42	26.14	51.29
PathRetriever	60.04	72.96	49.08	76.41	35.35	61.18
MDR	62.28	75.29	57.46	80.86	41.78	66.55
LOUVRE	**62.90**	**75.82**	**57.71**	**81.26**	**42.18**	**67.08**

Table 4: End-to-end multi-hop QA performance of models on HOTPOTQA test set.

Retriever	#BERT	Joint F1
Reader: BERT		
- PathRetriever-eff	50^+	56.85
- LOUVRE-eff	50^+	**60.23**
Reader: ELECTRA		
- MDR	450^+	66.55
- LOUVRE-Wiki	450^+	**66.87**

Table 5: The end-to-end multi-hop performance of LOUVRE-eff and LOUVRE-Wiki on HOTPOTQA test set with the same inference speed as baselines.

ori. The number of BERT executions for each baseline is derived from each paper and its implementation. We measure the inference time of LOUVRE, PathRetriever, and MDR in various hyper-parameter settings by adjusting the number of beam size and the number of documents retrieved by the sparse retriever, TF-IDF. The number of BERT executions of MDR, LOUVRE-eff, and LOUVRE-eff with a beam size of b is calculated as follows: #BERT $= 1$(question encoding)$+$ b(question-passage encoding). The inference time of MDR-rerank, LOUVRE-rerank, and LOUVRE-rerank-wiki involves another factor, the input size of the reranker. The inference time of these reranking models with a beam size of b and a input size of r becomes #BERT $= 1 + b + r$. For PathRetriever, we vary the number of documents retrieved by the sparse retriever, TF-IDF. Figure 5 illustrates that LOUVRE is more effective and efficient than the baselines because it yields better retrieval performance with a much smaller number of BERT executions.

5.2 End-to-End Multi-Hop QA

In this section, we demonstrate that the end-to-end multi-hop QA pipeline using LOUVRE retains the three outcomes of LOUVRE: overall performance improvement, robustness on a few-shot setting, and the fast inference speed. We use multi-hop QA pipelines of MDR and PathRetriever. All the components of the multi-hop QA pipelines except the retriever are fixed. We plug in each baseline retriever and LOUVRE to the pipeline and evaluate the end-to-end performance of each model.

End-to-End Performance: Table 4 shows the end-to-end multi-hop QA performance of baselines and LOUVRE. In this experiment, we replace the retriever of MDR's pipeline with LOUVRE-rerank. We set the beam size to 30 and the input size of the reranker to 900. LOUVRE outperforms baselines with a Joint F1 of 67.08. Table 5 shows the performance of LOUVRE-Wiki using the same inference time as MDR. In this experiment, we use a beam size of 100 and the reranker's input size of 350. MDR uses a beam size of 200 and the reranker's input size of 250. LOUVRE-Wiki outperforms MDR by 0.32% point in terms of Joint F1. We conduct the same experiment with LOUVRE-eff and PathRetriever. Table 5 shows the results. LOUVRE-eff in Table 5 represents the end-to-end pipeline of PathRetriever-eff using LOUVRE-eff as the initial candidate document retriever not TF-IDF. We set the number of initial candidate documents of LOUVRE-eff and PathRetriever-eff to 50. LOUVRE-eff outperforms PathRetriever-eff by 3.38% point without any loss of computational efficiency. We provide the detailed experimental results of LOUVRE-eff in Appendix B.

End-to-End Performance of Variations of LOUVRE-eff: Table 6 shows the end-to-end performance of each model on different inference time, size of train data, and pre-training. We evaluate two types of LOUVRE-eff. LOUVRE-eff (RR) represents the same pipeline used in Table 5. LOUVRE-eff (w/o RR) represents the pipeline that consists of LOUVRE-eff and the reader. Comparison between PathRetriever(500^+)

	RR	#BERT	Joint F1
PathRetriever	✓	500$^+$	59.5
- fast inference	✓	100$^+$	58.6
- fast inference	✓	50$^+$	56.7
LOUVRE-eff	✓	100$^+$	60.5
- fast inference	✓	50$^+$	60.1
LOUVRE-eff		41	57.9
- 1% train set		41	57.1
- w/o pre-training		41	54.3
- fast inference		5	53.3

Table 6: End-to-end multi-hop QA performances of baseline models and LOUVRE on HotpotQA dev set. #BERT denotes the number of BERT-base executions in supporting document prediction and the number of documents that the reader takes as input. We exclude the inference time of the supporting sentence selector. We use the same reader model as PathRetriever.

and LOUVRE-eff (100$^+$) shows that applying LOUVRE-eff to PathRetriever increases Joint F1 by 1 with 5 times faster inference speed. We conduct the same experiment by reducing the number of documents retrieved by LOUVRE-eff and achieve 0.6% point higher Joint F1 than PathRetriever with 10 times faster inference speed.

We conduct ablation studies with two factors of retrievers: 1) size of train data and 2) pre-training. We fix the reader with BERT-wwm fully fine-tuned on HotpotQA train set. Table 6 shows the results. When we train LOUVRE-eff with only 1% of the train set, the end-to-end performance drops by 0.8% point. However, using RoBERTa with the whole train set decreases the performance by 3.6% point. This result indicates that our pre-training methods bring more significant improvement to the end-to-end multi-hop QA pipeline when the size of the train data is small.

Decreasing the search space of multi-hop retrievers increases the retriever's computational efficiency but results in a significant performance drop. We demonstrate the robustness of LOUVRE-eff when the computation time is limited. We decrease the beam size of LOUVRE-eff to 2 and the number of output paths to 2; the total number of BERT executions of this model is #BERT $=$ $1 + 2$(beam size) $+$ 2(input size). Table 6 shows that LOUVRE-eff achieves 89% of PathRetriever's performance with almost 100 times faster inference speed of PathRetriever. We adjust the inference speed of PathRetriever and compare it with LOUVRE-eff (w/o RR). LOUVRE-eff (w/o RR) outperforms PathRetriever (50$^+$) by 1.2% point with less computation time. LOUVRE-eff (w/o RR) trained only with 1% train data even outperforms PathRetriever (50$^+$).

6 Conclusion

Answering complex questions includes reasoning across multiple documents. Recent studies have found that reasoning requires learning sub-question detection and relevant document retrieval to predict n correct answer with supporting facts. However, building such datasets requires costly human annotation and has limited scalability. To address this issue, we proposed a weakly supervised pre-training method for multi-hop retriever, LOUVRE. Our pre-training method contains three elements: "Next Document Prediction" task, "Bridge Entity Re-Phrasing", and a model. We demonstrated the efficacy of LOUVRE and its robustness on few-shot settings with extensive experiments on supporting document retrieval task and end-to-end multi-hop QA task. We also showed that our method performs very well at a much lower inference cost.

Acknowledgements

This work was partly supported by NAVER Corp. and Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korean government(MSIT) (No. 2017-0-01780, The technology development for event recognition/relational reasoning and learning knowledge based system for video understanding).

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *ACL*.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *EMNLP*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *ICML*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2020. Hopretriever: Retrieve hops over wikipedia to answer complex questions. *arXiv*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *EMNLP-IJCNLP*.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *EMNLP*.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. 2020. Unsupervised multi-hop question answering by question generation. *arXiv*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *EMNLP*.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D Manning. 2019. Answering complex open-domain questions through iterative query generation. In *EMNLP-IJCNLP*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *EMNLP-IJCNLP*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Yoav Goldberg, Matt Gardner, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *TACL*.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, et al. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *ICLR*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *ICLR*.

A Appendix: Baselines

Retrievers: Retrievers encode questions and documents independently, and search documents by comparing the question vector and the document vectors. Retrievers are computationally efficient compared to rerankers since the document vectors can be indexed before the questions are given. In **TF-IDF**, to get the list of supporting documents from the retrieved documents, we rearrange the ranked documents $[d_1, d_2, d_3, d_4, ...]$ to $[\{d_1, d_2\}, \{d_3, d_4\}, ...]$. **DPR** (Karpukhin et al., 2020) is a single-hop document retrieval model. We compare LOUVRE to the performance of DPR fine-tuned on HOTPOTQA reported from Xiong et al. (2021). **Cognitive Graph** (Ding et al., 2019) and **GOLDEN Retriever** (Qi et al., 2019) are multi-hop document retrieval models. We report the performance of Cognitive Graph reported from Asai et al. (2020) and the performance of GOLDEN Retriever reported from Qi et al. (2019). We fine-tune LOUVRE with the same training method proposed in **MDR** (Xiong et al., 2021); thus, the performance difference between LOUVRE and MDR only results from our pre-training method. We report the performance of MDR from Xiong et al. (2021).

Rerankers: Rerankers reorder reasoning paths/documents predicted by retrievers. Rerankers

	RR	#BERT	Answer EM	Answer F1	Support EM	Support F1	Joint EM	Joint F1
PathRetriever	✓	500^+	60.2	72.8	46.3	73.7	33.8	59.5
	✓	100^+	59.7	72.6	44.8	72.6	32.7	58.6
	✓	50^+	58.3	71.1	42.7	70.8	31.0	56.7
LOUVRE-eff	✓	100^+	60.4	73.4	48.0	74.7	35.1	60.5
- fast inference	✓	50^+	60.2	72.9	47.6	74.5	34.8	60.1
LOUVRE-eff		41	59.3	71.5	44.9	71.7	33.4	57.9
- 1% train data		41	58.7	71.0	44.3	70.7	32.6	57.1
- w/o pre-training		41	57.9	70.1	42.4	69.0	31.5	54.3
- fast inference		5	54.3	66.3	41.9	67.2	30.5	53.3

Table 7: End-to-end multi-hop QA performances of baseline models and LOUVRE on HOTPOTQA dev set.

	#BERT	Answer EM	Answer F1	Support EM	Support F1	Joint EM	Joint F1
SemanticRetrievalMRS	39.4/-	45.32	57.34	38.67	70.83	25.14	47.60
Transformer-XH	100/-	51.60	64.07	40.91	71.42	26.14	51.29
PathRetriever	50^+	58.21	70.86	42.91	71.30	30.95	56.85
LOUVRE-eff	50^+	**59.79**	**72.65**	**47.95**	**74.89**	**34.54**	**60.23**

Table 8: Evaluation results on HOTPOTQA test set. We retrieve the test results of PathRetriever (50^+) model and our model from the HOTPOTQA leaderboard.

calculate score of each paths/documents by jointly encode each document with the given question. As a result, reranking takes a huge portion of computation time of the end-to-end multi-hop QA pipeline. **SemanticRetrievalMRS** (Nie et al., 2019) propose the document reranking model that takes output of sparse retrievers such as TF-IDF. Since the model outputs documents not a list of supporting documents, we use the same document rearranging method as TF-IDF above. **PathRetriever** (Asai et al., 2020) and **HopRetriever** (Li et al., 2020) are reasoning path prediction models. These models use TF-IDF and BERT to retrieve and rerank the candidate documents. They use Wikipedia hyperlinks for candidate documents selection as described in section 4.4 and beam search with size 8 to rank each predicted supporting documents. **MDR** (Xiong et al., 2021) provides a reranking model as well as their retriever. We report the performance of MDR-reranking from Xiong et al. (2021).

B Appendix: End-to-End Performance of LOUVRE-eff

Table 7 shows the additional results of Table 6. We evaluate LOUVRE-eff on other evaluation metrics used in the HOTPOTQA benchmark and verify the efficacy of LOUVRE-eff. We report the detailed results of Table 5 in Table 8. The results show the end-to-end performance of LOUVRE-eff and PathRetriever on the HOTPOTQA test set.

Meet The Truth: Leverage Objective Facts and Subjective Views for Interpretable Rumor Detection

Jiawen Li, Shiwen Ni and Hung-Yu Kao
Intelligent Knowledge Management Lab
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan
`{P78073012,P78083033}@gs.ncku.edu.tw,hykao@mail.ncku.edu.tw`

Abstract

Existing rumor detection strategies typically provide detection labels while ignoring their explanation. Nonetheless, providing pieces of evidence to explain why a suspicious tweet is rumor is essential. As such, a novel model, LOSIRD, was proposed in this paper. First, LOSIRD mines appropriate evidence sentences and classifies them by automatically checking the veracity of the relationship of the given claim and its evidence from about 5 million Wikipedia documents. LOSIRD then automatically constructs two heterogeneous graph objects to simulate the propagation layout of the tweets and code the relationship of evidence. Finally, a graphSAGE processing component is used in LOSIRD to provide the label and evidence. To the best of our knowledge, we are the first one who combines objective facts and subjective views to verify rumor. The experimental results on two real-world Twitter datasets showed that our model exhibited the best performance in the early rumor detection task and its rumor detection performance outperformed other baseline and state-of-the-art models. Moreover, we confirmed that both objective information and subjective information are fundamental clues for rumor detection.

1 Introduction

With the prevalence of social media platforms, rumors have been a serious social problem. Notably, existing rumor detection methods roughly formulate this task as a natural language classification task. The goal of the task is to simply label a given textual claim as rumor or non-rumor. Nevertheless, only a verdict to a suspicious statement is insufficient for people to understand and reason why a claim is a rumor. For example, Fig.1 is the comparison figure of existing rumor detection methods and a rumor detection method that provides evidence. The claim in Fig. 1 is a half-truth, which is highly deceptive. For such rumors, providing a label only is unconvincing. Thus, we believe that a good rumor detection system should have 2 essential functions including, a rumor identifying function and evidence providing function.

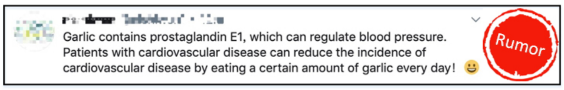

[redacted]
Garlic contains prostaglandin E1, which can regulate blood pressure. Patients with cardiovascular disease can reduce the incidence of cardiovascular disease by eating a certain amount of garlic every day! 😃

(a) Genral rumor detection system

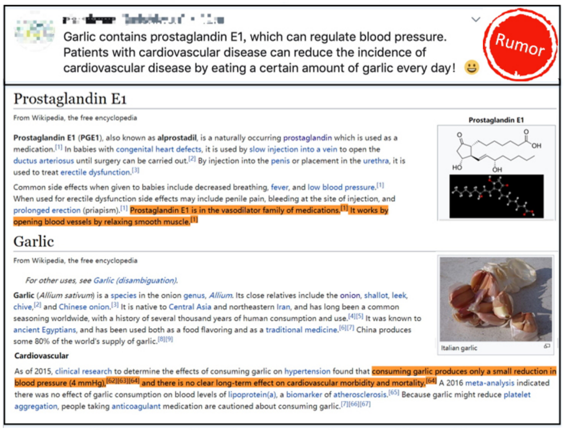

[redacted]
Garlic contains prostaglandin E1, which can regulate blood pressure. Patients with cardiovascular disease can reduce the incidence of cardiovascular disease by eating a certain amount of garlic every day! 😃

Prostaglandin E1

From Wikipedia, the free encyclopedia

Prostaglandin E1 (**PGE1**), also known as **alprostadil**, is a naturally occurring prostaglandin which is used as a medication.[1] In babies with congenital heart defects, it is used by slow injection into a vein to open the ductus arteriosus until surgery can be carried out.[2] By injection into the penis or placement in the urethra, it is used to treat erectile dysfunction.[3]

Common side effects when given to babies include decreased breathing, fever, and low blood pressure.[1] When used for erectile dysfunction side effects may include penile pain, bleeding at the site of injection, and prolonged erection (priapism).[1] Prostaglandin E1 is in the vasodilator family of medications.[1] It works by opening blood vessels by relaxing smooth muscle.[1]

Garlic

From Wikipedia, the free encyclopedia

For other uses, see Garlic (disambiguation).

Garlic (*Allium sativum*) is a species in the onion genus, *Allium*. Its close relatives include the onion, shallot, leek, chive,[2] and Chinese onion.[3] It is native to Central Asia and northeastern Iran, and has long been a common seasoning worldwide, with a history of several thousand years of human consumption and use.[4][5] It was known to ancient Egyptians, and has been used both as a food flavoring and as a traditional medicine.[6][7] China produces some 80% of the world's supply of garlic.[8][9]

Cardiovascular

As of 2015, clinical research to determine the effects of consuming garlic on hypertension found that consuming garlic produces only a small reduction in blood pressure (4 mmHg),[62][63][64] and there is no clear long-term effect on cardiovascular morbidity and mortality.[64] A 2016 meta-analysis indicated there was no effect of garlic consumption on blood levels of lipoprotein(a), a biomarker of atherosclerosis.[65] Because garlic might reduce platelet aggregation, people taking anticoagulant medication are cautioned about consuming garlic.[7][66][67]

(b) a rumor detection system that provides evidence

Figure 1: Comparison figure of rumor detection results of two different systems. The orange highlights in Fig.1 (b) are pieces of evidence retrieved from Wikipedia. From those evidence sentences, readers can easily judge if the given claim is a half-truth and clearly understand why that claim is a rumor.

Rumor detection that provides evidence has the following benefits: (1) Improve detection performance. (2) Improve the user experience. (3) Provide a basis for manual review. (4) Improve the accuracy of early rumor detection. (5) Intercept the spread of similar rumors.

Despite having numerous advantages, rumor detection that provides evidence is extremely hard. If none of the labeled evidence information is included in a rumor detection training dataset, the deep learning network is unlikely to generate these textual evidence contents by itself. Unfortunately,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 705–715
August 1–6, 2021.

Table 1: Comparison characteristics table of subjective and objective information.

Subjectivity infor	**Objective infor**
easy access	need crawl
extensive	rare
one-sidedness	comprehensive
conflicting	consistency
has noise	high purity

the datasets currently used for rumor detection cannot be used as evidence.

To find out what type of information that can be used as evidence, two different kinds of information, subjective information and objective information, are discussed in this part (Merigo et al., 2016; Zorio-Grima and Merello, 2020). Under the field of rumor detection, subjective information refers to source tweets, comments, etc, while objective information refers to the information on Wikipedia or Baidu Encyclopedia, etc. Through our comprehensive analysis, we found that subjective information and objective information shows distinct-different characteristics, which are summarized in Table 1. The objective information is consistency and high purity, can be used as evidence, and the subjective information also contains certain clues for debunking rumors.

To take advantage of both subjective information and objective information, a novel model, LOSIRD, is proposed in this paper. This is notoriously challenging, the difficulties lie in: (1) The model should have a strong retrieval ability. (2) The model should have a Natural Language Inference(NLI) ability. (3) The model needs to be able to process the topology information.

Fig.2 shows a high-level view of its architecture. This model is divided into two modules i.e., ERM (Evidence Retrieval Module) and RDM (Rumor Detection Module). Inspired by the concept of transfer learning, a two-stage training approach was used for our LOSIRD model. In the first training phase, a widely used fact-checking database was utilized for training the ERM module. In the second training phase, two rumor detection datasets were used to train and evaluate the model.

The main contribution of this paper is four folds:

1. This study, for the first time, arguably proposes a rumor detection model that provides evidence.
2. We are the first to propose two novel graph objects to simulate the propagation lay out of tweets and embedding the relationship of evidence and the claim rumor detection task.
3. Our LOSIRD achieved the highest detection accuracy and outperformed state-of-the-art models in the rumor detection task.
4. Our LOSIRD is more generalizable and robust in the early detection of rumor.

2 Related work

2.1 Evidence Retrieval

The evidence retrieval task is highly correlated with the rumor detection task. One of the most widely used datasets for evidence retrieval is FEVER[1]. Majority of researchers handle the fever share task by following the FEVER organizers' pipeline approach, retrieve and verify the evidence in three steps (Hanselowski et al., 2018; Malon, 2018). Zhou et al. (2019a) formulated claim verification as a graph reasoning task and provides two kinds of attention. Liu et al. (2020) presented KGAT combining edge kernels and node kernels to better embedding and filtering the evidence. Zhong et al. (2020) constructed two semantic-level topologies to enhance verification performance. Yoneda et al. (2018) employed a four-stage model for the fever share task.

2.2 Rumor Detection

The existing rumor detection deep learning methods can be divided into three categories, feature-driven method, content-driven method, and hybrid-driven method.

Feature-Driven approaches, like machine learning methods, rely on a variety of characteristics to identify rumors. Rath et al. (2017) proposed a new concept of believability for automatic identification of the users spreading rumors.

Content-Driven approaches are a kind of method base on natural language processing. Many researchers adopted deep learning models to handle this task (Rath et al., 2017; Ma et al., 2016; Yu et al., 2017; Chen et al., 2018; Ma et al., 2018). Monti et al. (2019) proposed a propagation-based Fake News Detection by GCN. Nguyen (2019) detected rumor using Multi-modal Social Graph. Sujana et al. (2020) proposed a multiloss hierarchical BiLSTM model for fake news detection.

---

[1] http://fever.ai/task.html

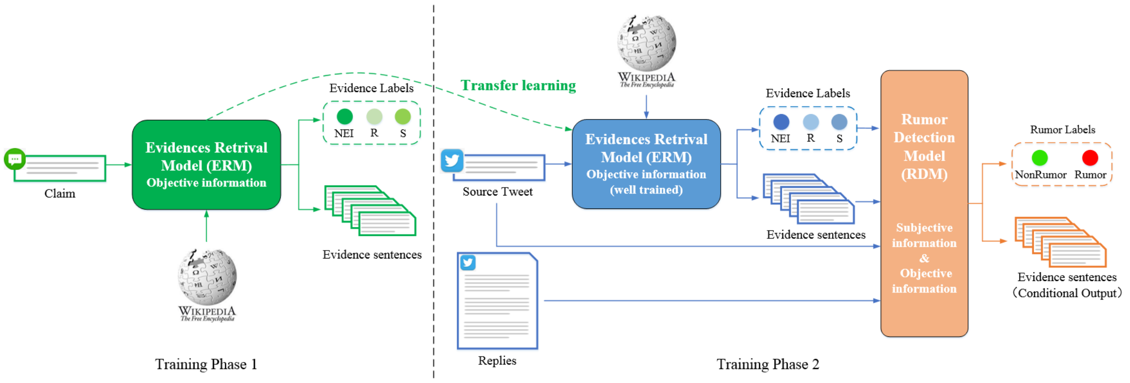

Figure 2: The architecture of our LOSIRD model. The claim and source tweet are essentially the same thing in this paper, "Claim" means the model is training using the Fever dataset, while "source tweet" indicates the PHEME datasets are used.

Hybrid-Driven approaches incorporate both feature engineering and text information representation to detect rumors (Liu and Wu, 2018; Yang et al., 2018). Ruchansky et al. (2017) proposed a model called CSI for rumor detection, which uses articles and extracts user characteristics to debunk rumors. Lu and Li (2020) classified rumor by extracting user's features from their profiles and social interactions. Li et al. (2020b) used GraphSEGA to encode the conversation structure. Li et al. (2020a) crawled user-follower information and built a friendly network based on the follow-followers relationship. Castillo et al. (2011) used tweets and re-posts information to detect rumors. Kochkina et al. (2018); Li et al. (2019) proposed a multi-task learning method to joint training of the main and auxiliary tasks, improving model's rumor detection the performance. Liu et al. (2015) aggregated common sense and investigative journalism of Twitter users for rumor detection. Ma et al. (2017) encoded post's propagation structure for rumor detection. Detect rumors in microblog posts using propagation structure via kernel learning.

2.3 Comparison

The highlights of our model include providing evidence, covering two heterogeneous structure graph information, combining both evidence clues and replies information in detecting rumor. Our model exhibited a stronger simulation ability, better scalability, and better persuasive ability.

3 The LOSIRD Model

3.1 Problem Statement

We formulated this rumor detection task as a hybrid task that combines the evidence retrieval sub-task and rumor prediction sub-task.

The evidence retrieval sub-task was defined as: Given a claim, the target of this sub-task was to match textural evidence from Wikipedia and reason the relationship between those potential evidence sentences and the given claim as "SUPPORTED", "REFUTED" or "NOT ENOUGH INFO (NEI)". We defined the Wikipedia as an objective information corpus: $Wiki = \{D_1, D_2, ..., D_{|w|}\}$, D_i as a document from Wikipedia. One document comprised several sentences describing one entity in Wikipedia. The goal of this sub-task was to retrieve evidence, classify the relationship of the evidence and the given claim C i.e., $f_{ERM} : C \rightarrow \{(y_e, E); E \in Wiki\}$, y_e is the predicted evidence label of the claim, E is retrieved evidence set of the claim which contains several sentence-level pieces of evidence.

The rumor prediction sub-task was defined as: Given a claim, that claim's replies, that claim's retrieved evidence set and evidence label, the model detected whether the claim was a rumor or non-rumor and provide the evidence. We defined the rumor dataset in this sub-task as $\Psi = \{T_1, T_2, ..., T_{|\Psi|}\}$, where T_i is a tweet in the dataset. $T_i = \{C_i, P_i, E_i, y_{e_i}\}$, where C_i is the ith source tweet in the rumor dataset, P_i is the source tweet's reply posts, E_i and y_{e_i} are the corresponding C_i retrieved evidence set and evidence label. Given a tweet T the function of this task was defined as

$f_{RDM} : T \to \{(y_r, E); E \in Wiki\}$, y_r was the predicted rumor label.

3.2 ERM

Mainly following (Liu et al., 2020) and (Hanselowski et al., 2018), we adopted a three-step pipeline module for retrieval evidence, called ERM. The architecture of the ERM is shown in Fig. 3. It contains three main steps ,i.e., Document retrieval: employs a keyword matching algorithm to crawling related files in Wikipedia. Evidence retrieval: extract sentence-level evidence from the retrieved articles. Claim verification: based on the sentence-level evidence, it predicts the relationship of the claim and the evidence as "Supported", "Refuted", or "NEI". Specifically, the ERM first leverages semantic NLP tool-kits to extract potential entities from the given a claim. With the parsed entities, top k highest-ranked Wikipedia articles were filtered by the MediaWiki API. And then, from those retrieved documents, the ERM extracts objective facts as the predicted evidence in the form of sentences that are relevant for the claim. Finally, a verification component of ERM performs prediction over the given statement and retrieved evidence, and verifies the relationship between the claim and evidence as supporting, refuting, or NEI.

3.3 RDM

Fig. 4 shows the structure of RDM. Since the source tweet forms different topology between replies and the evidence, two heterogeneous graph objects, the conversation tree-shaped graph and the evidence star-shaped graph, were structured in RDM.

3.3.1 Two heterogeneous graphs

Conversation tree-shaped structure is a peculiar reply relationship topology that forms naturally from social media and carries a vital clue for rumor detection (Belkaroui et al., 2014; Pace et al., 2016). Of note, the conversation structure is tree-shaped. Where the root of the tree is the source tweet, each node represents a comment, and each node is connected by its reply relationship.

Evidence star-shaped structure suggests that each piece of evidence is a supplementary description to the source tweet, hence each evidence sentence directly related to the source tweet forms a star topology. In this star-shaped structure, the node of the source tweet is in the center and all the evidence nodes surround the source tweet representing an angle in the star structure.

3.3.2 Rumor Detection Module

The rumor detection module contains four components: (1) a word vector encoding component. (2) a sentence embedding component. (3) a graph processing component. (4) a classifier component.

In RDM, a deep BiLSTM was utilized to extract the information among words and generate a sentence representation. The obtained sentence vector was passed into a graph processing component, the GraphSAGE (Hamilton et al., 2017) model was used as its backbone. The GraphSAGE effectively handled variable graphs. Since the output of the previous component is a set of sentence vectors does not contain structural information. Therefore, before passing this information into the graph processing component two graph objects, the conversation tree-shaped object and the evidence star-shaped object were constructed respectively.

The creation of the conversation graph object:

$$\begin{aligned} G_p &= (V_p, E_p) \\ V_p &= [c, p_1, p_2, .., p_j] \\ E_p &= \{(c, p_1), ...(p_n, p_m), ...\} \end{aligned} \tag{1}$$

where G_p is the ith event's conversation graph object, it's vertex set is V_p and edge set is E_p. The vertex set includes all the post in the event, and the edge set E_p means the reply relationship between each post. c and p_j are the tweet embedding results from the BiLSTM component, we selected the last hidden state of BiLSTM as a sentence embedding result.

The creation of the evidence graph object:

$$\begin{aligned} G_e &= (V_e, E_e) \\ V_e &= [c, e_1, e_2, .., e_k] \\ E_e &= \{(c, e_1), (c, e_2), ..., (c, e_k)\} \end{aligned} \tag{2}$$

where G_e is an evidence graph object consisting of a vertex set V_e and an edge set E_e. The vertex set V_e includes a source post c and evidence sentences, while the edge set E_e represents the relationship between the evidence and the source post, e_k represents the evidence sentence embedding results from the BiLSTM component.

At the beginning of the forward propagation step, the feature of each node was assigned to the nodes

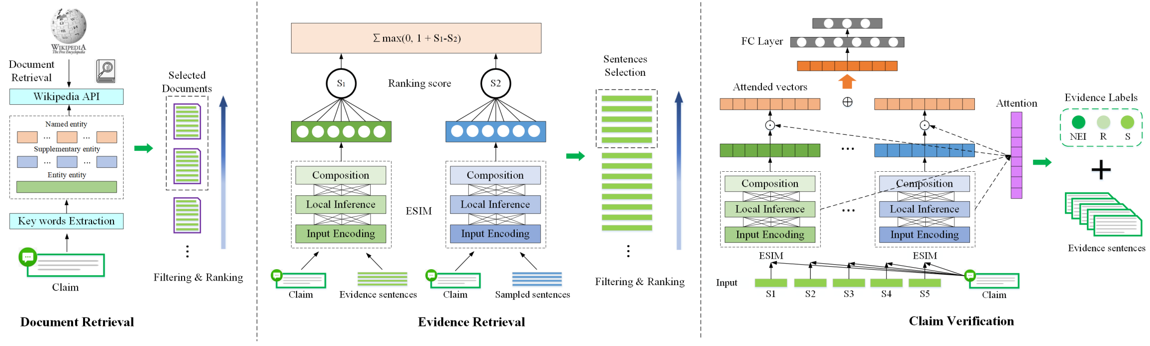

Figure 3: The architecture of the ERM.

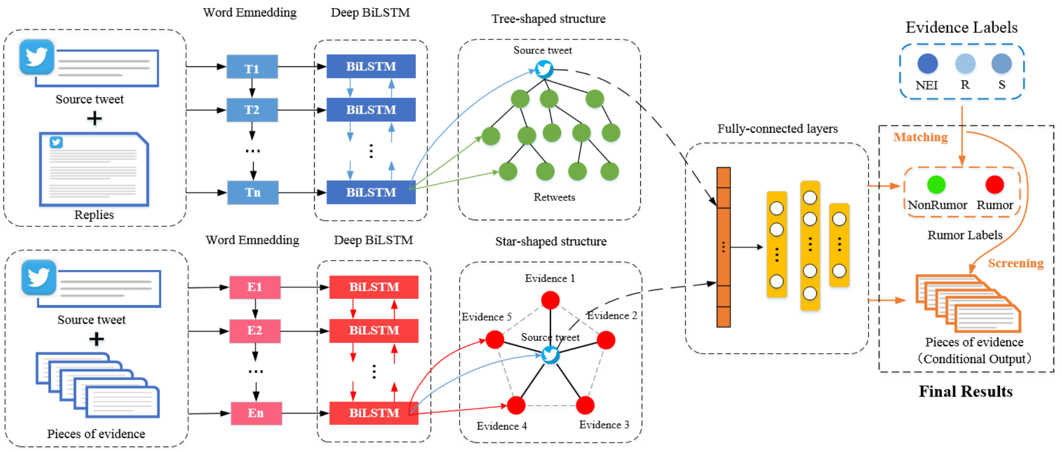

Figure 4: The architecture of the RDM.

in the hidden state as follows:

$$\begin{aligned}[h^0_{p_0}, h^0_{p_1}, ..., h^0_{p_j}] &\longleftarrow [c, p_1, p_2, .., p_j] \\ [h^0_{e_0}, h^0_{e_1}, ..., h^0_{e_k}] &\longleftarrow [c, e_1, e_2, .., e_k]\end{aligned} \tag{3}$$

where $h^0_{p_j}, h^0_{e_k}$ are the initial hidden states of the nodes of the conversation graph object and the evidence graph object in GraphSAGE.

The node's hidden state in GraphSAGE updates by constantly aggregating its immediate neighbors' hidden state, combining them with its own state and generate it's new hidden state.This process makes the nodes gain incrementally richer information (Hamilton et al., 2017):

$$\begin{aligned}h^k_{p_{N(v)}} &\longleftarrow AGG^{pool}_k(\{h^{k-1}_{p_u}, \forall u \in N(V_p)\}) \\ h^k_{p_v} &\longleftarrow \sigma(W^k_p \cdot CON(h^{k-1}_{p_v}, h^k_{p_{N(v)}}))\end{aligned} \tag{4}$$

$$\begin{aligned}h^k_{e_{N(v)}} &\longleftarrow AGG^{pool}_k(\{h^{k-1}_{e_u}, \forall u \in N(V_e)\}) \\ h^k_{e_v} &\longleftarrow \sigma(W^k_e \cdot CON(h^{k-1}_{e_v}, h^k_{e_{N(v)}}))\end{aligned} \tag{5}$$

where $h^k_{p_{N(v)}}, h^k_{e_{N(v)}}$ is the aggregated their neighborhood vectors, k is the depth of the information transmission updates (the number of times the graph information is updated), N is the neighborhood function, $N(v)$ is the set of the node's immediate neighborhood, and AGG^{pool}_k is the aggregation function and CON is the concatenation function.

Three aggregators are provided in GraphSAGE, and in this article we chose the Max Pooling aggregator. Here's the formula:

$$\begin{aligned}AGG^{pool}_k =& max(\{\sigma(W_{pool} \cdot h^k_{graph_{N(v)}} + \\ & b_{pool}), \forall u_i \in N(v)\})\end{aligned} \tag{6}$$

where max is the element-wise max operator, and σ is a nonlinear activation function.

After k iterations of information transmission based on the conversation structure and star structure, final representations of the conversation embedding results and evidence embedding results were obtained:

$$\begin{aligned}p &\longleftarrow h^k_{p_v}, \forall v \in V_p \\ e &\longleftarrow h^k_{e_v}, \forall v \in V_e\end{aligned} \tag{7}$$

p, e are the replies and the evidence of the ith event.

Table 2: The statistic of FEVER dataset.

Split	SUPPORTED	REFUTED	NEI
Train	80,035	29,775	35,659
Dev	6,666	6,666	6,666
Test	6,666	6,666	6,666

Table 3: The statistic of rumor datasets.

Statistic	PHEME2017	PHEME2018
Users	49,345	50,593
Posts	103,212	105,354
Events	5,802	6,425
Avg posts/event	17.8	16.3
Rumor	1,972	2,402
Non-rumor	3,830	4,023

Max aggregator is used to aggreate the information into fixed size.

Thereafter, the information of these two parts concatenated together then passed into a multilayer perceptron for the final prediction. The formula is as follows:

$$y_r = Softmax(V \cdot (p \oplus e) + b_y) \qquad (8)$$

where V and b_y are parameters in the output layer.

4 Experiment and Results

4.1 Datasets

Fever dataset was used to train the evidence retrieval module. The statistic of the FEVER dataset is shown in Table 2. Two widely used rumor datasets, PHEME 2017 and PHEME 2018[1], were used to train and evaluate the whole proposed model, as shown in Table 3[2].

4.2 Experimental Setup

To evaluate the rumor detection performance of our model, we compared our proposed models with other popular rumor detection models, including some of the current state-of-the-art models. In the text processing stage, we clean the text information by removing useless expressions and symbols, uniform case, etc. We use Twitter 27B pre-trained GloVe data with 200 dimensions for word embedding and set the maximum vocabulary to 80,000. For the rumor detection module The hidden size of BiLSTM is 128, and the number of layers is 2. The batch size of graphSAGE is 64. We use Adam with a 0.0015 learning rate to optimize the model, with the dropout rate set to 0.5. For the evidence retrieval, we set the learning rate in ESIM is 0.002, drop out rate is 0, batch size is 64, activation fuction is relu. For the claim verification, we set the the learning rate in ESIM is 0.002, drop out rate is 0.1, batch size is 128,activation fuction is relu.We split the datasets, reserve 10% of the events as the validation set, and the rest in a ratio of 3:1 for training and testing partitions.

- **CNN**: a convolutional neural model for rumor detection (Chen et al., 2017).
- **BiLSTM**: a bidirectional LSTM model for debunking rumors (Augenstein et al., 2016).
- **BERT**: a fine-tuned BERT to detect rumors (Devlin et al., 2019).
- **CSI**: a state-of-the-art model detecting rumor by scoring users based on their behavior (Ruchansky et al., 2017).
- **DEFEND**: a state-of-the-art model learns the correlation between the source article's sentences and user profiles (Shu et al., 2019).
- **RDM**: a state-of-the-art model integrating GRU and reinforcement learning to detect rumors at an early stage (Zhou et al., 2019b).
- **CSRD**: a state-of-the-art model that detect rumors by modeling conversation structure (Li et al., 2020b).
- **LOSIDR**: our model, leverages objective facts and subjective views for interpretable rumor detection.

4.3 Experimental Results

The main experimental results are shown in Table 4. The LOSIRD outperformed the other best-competing methods on PHEME 17 and PHEME 18. Its accuracy was 91.4% in PHEME 2017 and 92.5% in PHEME 2018. Moreover, the precision, recall, and F1 were all higher than 90% in both two datasets. Such promising results confirmed the effectiveness of evidence information and the topology message processing method in rumor detection. For the CNN, BiLSTM, DEFEND, and RDM models, they typically concatenated posts as a single line based on the publish time, while ignoring the

---

[1] https://figshare.com/articles/dataset/

[2] This study have met the terms of accessing and using these rumor datasets.

Table 4: Main Experimental results. The best model and the best competitor are highlighted by bold and underline.

Method	**PHEME 2017**				**PHEME 2018**			
	Acc	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**
CNN	0.787	0.737	0.702	0.710	0.795	0.731	0.673	0.686
BiLSTM	0.795	0.763	0.691	0.725	0.794	0.727	0.677	0.701
BERT	0.865	0.859	0.851	0.855	0.844	0.834	0.835	0.835
CSI	0.857	0.843	0.859	0.851	0.851	0.836	0.855	0.845
DEFEND	0.868	0.867	0.859	0.863	0.863	0.857	0.859	0.858
RDM	0.873	0.817	0.823	0.820	0.858	0.847	0.859	0.852
CSRD	0.900	0.893	0.869	0.881	0.919	0.892	0.923	0.907
LOSIRD	**0.914**	**0.915**	**0.900**	**0.906**	**0.925**	**0.922**	**0.924**	**0.923**

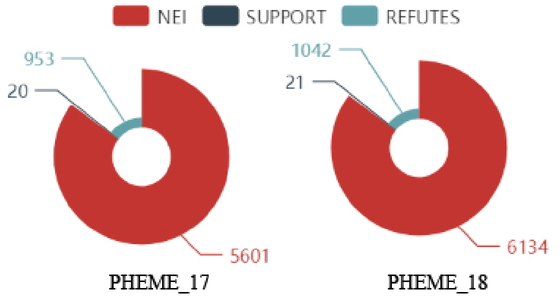

Figure 5: The distribution of the retrieved evidence.

conversation structure information. Nonetheless, the structure was crucial for encoding the posts to comprehensive and precise representations. The CSI and CRNN processed topology information, but only the subjective information was adopted in those models causing insufficiency in information extraction.

4.4 Evidence Impact Study

In this section, we discussed whether the evidence facilitates rumor detection and determined the extent of the impact of the evidence in debunking rumor. Notably, the evaluated datasets were the PHEME 2017 and PHEME 2018.

4.4.1 Distribution of Retrieved Evidence

To accurately evaluate the retrieved evidence, the distribution of the retrieved evidence based on its evidence label was analyzed. Two pie charts were constructed to reflect their distribution situations. As shown in Fig. 5, most of the retrieved evidence was irrelevant to the given claim, and about 14.8% of the retrieved evidence sentences had sufficient information that supports or refutes the given claim. Despite the proportion of supports and refutes being not large, this result was commendable and better than our expectations.

Table 5: Retrieved evidence probability analysis result.

Dataset	Original	Refutes	Increment
PHEME 17	33.30%	75.80%	42.50%
PHEME 18	36.60%	70.92%	34.30%

4.4.2 Retrieved Evidence Probability Analysis

We further evaluated the impact of evidence by statistically calculating the probability gap between rumor in original data and rumor in data that labeled refutes. The outcome is shown in Table 5. The probabilities of rumor in original data were about 35% in both datasets, while the probabilities of rumor in data that labeled refutes were around 73%, which was much higher than in original data. Specifically, rumor in data that labeled refutes increased to 42.5% on PHEME 17 and 34.3% on PHEME 18. This strongly confirmed that the retrieved evidence was a vital clue for rumor detection.

4.4.3 Influence Analysis of the Evidence on Deep Learning Model

To further illustrate the influence of evidence on rumor detection and analyze the impact of evidence on deep learning models, three NLP models, CNN, BILSTM, and BERT, were deliberately selected as the examination models in this subsection. We concatenated the suspicious claim and its evidence sentences and inputted them into the three models, respectively. The experimental results shows in Fig. 6. The horizontal axis represented a different number of evidence sentences, 0 means only source tweet, while 1 to 5 means source tweet plus 1 to 5 evidence sentences. Also, this paper analyzed the performance before and after the evidence was filtered which was represented as each chart with two

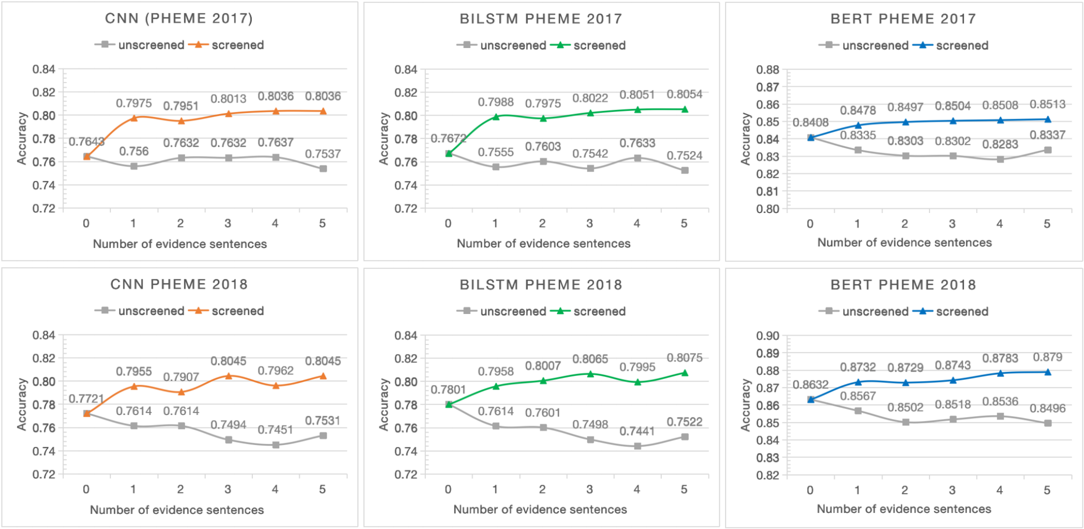

Figure 6: Influence analysis of the evidence on the performance of deep learning models.

lines i.e., one for the unscreened evidence (filter the NEI evidence) and the other for the screened evidence. The broken lines of unscreened in all the charts showed a downward trend. This indicated that the NEI evidence contained a certain amount of useless information there by making the detection process harder. Furthermore, after dropping the NEI evidence, all the models achieved an improvement by an increase of 5% accuracy on average. This demonstrated that the filtered evidence significantly helps the deep learning models in debunking rumors.

4.5 Early Detection Performance

To evaluate the early rumor detection performance of our model, 9 test sets that reflected real-world scenarios of rumors spreading on Twitter were created. Each test set included a different number of replies, ranging between 5 replies and 45 replies. The test subset was sampled based on the publication timestamp. As shown in Fig. 7, even though the number of posts was only 5, our LOSIRD model had more than 91% accuracy in PHEME 2017 dataset and PHEME 2018 dataset. Additionally, the broken line diagram showed that the curve of our model was significantly stable, indicating satisfactory robustness and high performance in early rumor detection. Besides, our model effectively made use of the objective information from Wikipedia, hence, it did not rely on subjective information from replies of the users thereby achieving satisfactory performance in the early stage of rumor propagation.

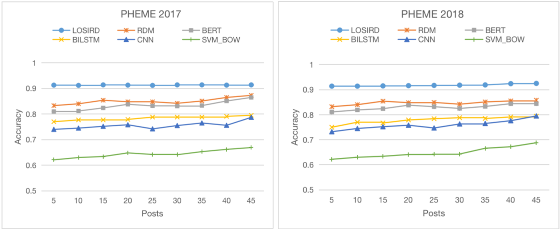

Figure 7: Early rumor detection performance.

5 Conclusion

In this paper, we proposed a LOSIRD, a novel interpretative model for rumor detection. Notably, the LOSIRD debunking rumor mechanism depends on both objective facts and subjective views. Objective fact sentences retrieved from 5,416,537 Wikipedia articles were sufficiently utilized to help LOSIRD in analyzing the veracity of a suspicious claim. Meanwhile, the information in subjective views was extracted by simulating the propagation of subjective views based on the conversation structure. Results on two public Twitter datasets showed that our model improved rumor detection performance by a certain margin compared to the state-of-the-art baselines. Further, we analyzed the impact of objective facts for rumor detection and analyzed the effectiveness of the conversation structure. The experiments revealed that both objective facts and subjective views were vital clues for debunking rumor. Moreover, we believe that our model will be used for rumor detection and other text classification tasks on social media.

Acknowledgments

This work was funded in part by Qualcomm through a Taiwan University Research Collaboration Project and in part by the Ministry of Science and Technology, Taiwan, under grant MOST 109-2221-E-006-173 and NCKU B109-K027D.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Rami Belkaroui, Rim Faiz, and Aymen Elkhlifi. 2014. Conversation analysis on social networking sites. In *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, pages 172–178. IEEE.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 675–684. ACM.

Tong Chen, Xue Li, Hongzhi Yin, and Jun Zhang. 2018. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 40–52. Springer.

Yi-Chin Chen, Zhao-Yang Liu, and Hung-Yu Kao. 2017. IKM at SemEval-2017 task 8: Convolutional neural networks for stance detection and rumor verification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 465–469, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. Association for Computational Linguistics.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jiawen Li, Shiwen Ni, and Hung-Yu Kao. 2020a. Birds of a feather rumor together? exploring homogeneity and conversation structure in social media for rumor detection. *IEEE Access*, 8:212865–212875.

Jiawen Li, Yudianto Sujana, and Hung-Yu Kao. 2020b. Exploiting microblog conversation structures to detect rumors. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5420–5429, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Quanzhi Li, Qiong Zhang, and Luo Si. 2019. Rumor detection by exploiting user credibility information, attention and multi-task learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1173–1179, Florence, Italy. Association for Computational Linguistics.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1867–1870. ACM.

Yang Liu and Yi-fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 354–361. AAAI Press.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online. Association for Computational Linguistics.

Yi-Ju Lu and Cheng-Te Li. 2020. GCAN: Graph-aware co-attention networks for explainable fake news detection on social media. In *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics, pages 505–514, Online. Association for Computational Linguistics.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3818–3824. IJCAI/AAAI Press.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 708–717, Vancouver, Canada. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia. Association for Computational Linguistics.

Christopher Malon. 2018. Team papelo: Transformer networks at FEVER. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 109–113, Brussels, Belgium. Association for Computational Linguistics.

Jose M Merigo, Daniel Palacios-Marques, and Shouzhen Zeng. 2016. Subjective and objective information in linguistic multi-criteria group decision making. *European Journal of Operational Research*, 248(2):522–531.

Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. 2019. Fake news detection on social media using geometric deep learning.

Thanh Tam Nguyen. 2019. Graph-based rumour detection for social media. Technical report.

Stefano Pace, Stefano Buzzanca, and Luciano Fratocchi. 2016. The structure of conversations on social networks: Between dialogic and dialectic threads. *International Journal of Information Management*, 36(6):1144–1151.

Bhavtosh Rath, Wei Gao, Jing Ma, and Jaideep Srivastava. 2017. From retweet to believability: Utilizing trust to identify rumor spreaders on twitter. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, July 31 - August 03, 2017*, pages 179–186. ACM.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. CSI: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 797–806. ACM.

Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu. 2019. defend: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 395–405. ACM.

Yudianto Sujana, Jiawen Li, and Hung-Yu Kao. 2020. Rumor detection on Twitter using multiloss hierarchical BiLSTM with an attenuation factor. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S Yu. 2018. Ti-cnn: Convolutional neural networks for fake news detection. *arXiv preprint arXiv:1806.00749*.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. UCL machine reading group: Four factor framework for fact finding (HexaF). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102, Brussels, Belgium. Association for Computational Linguistics.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2017. A convolutional approach for misinformation identification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3901–3907. ijcai.org.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online. Association for Computational Linguistics.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019a. GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 892–901, Florence, Italy. Association for Computational Linguistics.

Kaimin Zhou, Chang Shu, Binyang Li, and Jey Han Lau. 2019b. Early rumour detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1614–1623, Minneapolis, Minnesota. Association for Computational Linguistics.

Ana Zorio-Grima and Paloma Merello. 2020. Consumer confidence: Causality links with subjective and objective information sources. *Technological Forecasting and Social Change*, 150:119760.

Read, Listen, and See: Leveraging Multimodal Information Helps Chinese Spell Checking

Heng-Da Xu[1]*, **Zhongli Li**[2*], **Qingyu Zhou**[2], **Chao Li**[2], **Zizhen Wang**[2],
Yunbo Cao[2], **Heyan Huang**[1], **Xian-Ling Mao**[1]
[1]Beijing Institute of Technology
[2]Tencent Cloud Xiaowei
`{xuhengda,hhy63,maoxl}@bit.edu.cn`
`{neutrali,qingyuzhou,diegoli,zizhenwang,yunbocao}@tencent.com`

Abstract

Chinese Spell Checking (CSC) aims to detect and correct erroneous characters for user-generated text in Chinese language. Most of the Chinese spelling errors are misused semantically, phonetically or graphically similar characters. Previous attempts notice this phenomenon and try to utilize the similarity relationship for this task. However, these methods use either heuristics or handcrafted confusion sets to predict the correct character. In this paper, we propose a Chinese spell checker called REALISE, by directly leveraging the multimodal information of the Chinese characters. The REALISE model tackles the CSC task by (1) capturing the semantic, phonetic and graphic information of the input characters, and (2) selectively mixing the information in these modalities to predict the correct output. Experiments[1] on the SIGHAN benchmarks show that the proposed model outperforms strong baselines by a large margin.

Phonetically Similar Case		
Sent.	晚饭后他递给我一平(píng, *flat*)红酒。	
Cand.	晚饭后他递给我一杯(bēi, *cup*)红酒。	✗
	晚饭后他递给我一瓶(píng, *bottle*)红酒。	✓
	晚饭后他递给我一箱(xiāng, *box*)红酒。	✗
Trans.	He handed me a *bottle* of red wine after dinner.	
Graphically Similar Case		
Sent.	每天放学我都会轻(qīng, *light*)过这片树林。	
Cand.	每天放学我都会路(lù, *pass*)过这片树林。	✗
	每天放学我都会经(jīng, *go*)过这片树林。	✓
	每天放学我都会走(zǒu, *walk*)过这片树林。	✗
Trans.	I *go* through this wood every day after school.	

Table 1: Two examples of Chinese spelling errors and their candidate corrections. “Sent./Cand./Trans.” are short for sentence/candidates/translation respectively. The wrong/candidate/correct characters with their pronunciation and translation are in red/orange/blue color.

1 Introduction

The Chinese Spell Checking (CSC) task aims to identify erroneous characters and generate candidates for correction. It has attracted much research attention, due to its fundamental and wide applications such as search query correction (Martins and Silva, 2004; Gao et al., 2010), optical character recognition (OCR) (Afli et al., 2016), automatic essay scoring (Dong and Zhang, 2016). Recently, rapid progress (Zhang et al., 2020; Cheng et al., 2020) has been made on this task, because of the success of large pretrained language models (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019).

In alphabetic languages such as English, spelling errors often occur owing to one or more wrong characters, resulting in the written word not in the dictionary problem (Tachibana and Komachi, 2016). However, Chinese characters are valid if they can be typed in computer systems, which causes that the spelling errors are de facto misused characters in the context of computer-based language processing. Considering the formation of Chinese characters, a few of them were originally pictograms or phono-semantic compound characters (Jerry, 1988). Thus, in Chinese, the spelling errors are not only the misused characters with confusing semantic meaning, but also the characters which are phonetically or graphically similar (Liu et al., 2010, 2011). Table 1 shows two examples of Chinese spelling error. In the first example, phonetic information of “平” (flat) is needed to get the correct character “瓶” (bottle) since they share the same pronunciation “píng”. The second example needs not only phonetic, but also graphic information of the erroneous character “轻” (light). The

---

[*] Heng-Da Xu and Zhongli Li contributed equally. Work is done during internship at Tencent Cloud Xiaowei. Qingyu Zhou is the corresponding author.

[1] Code and model are available at `https://github.com/DaDaMrX/ReaLiSe`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 716–728
August 1–6, 2021. ©2021 Association for Computational Linguistics

correct one, “经” (go), has the same right radical as “轻” and similar pronunciation (“qīng” and “jīng”). Therefore, considering the intrinsic nature of Chinese, it is essential to leverage the phonetic and graphic knowledge of the Chinese characters along with the textual semantics for the CSC task.

In this paper, we propose REALISE (**Read**, **Listen**, and **See**), a Chinese spell checker which leverages the semantic, phonetic and graphic information to correct the spelling errors. The REALISE model employs three encoders to learn informative representations from textual, acoustic and visual modalities. First, BERT (Devlin et al., 2019) is adopted as the backbone of the semantic encoder to capture the textual information. For the acoustic modality, *Hanyu Pinyin* (pinyin), the romanization spelling system for the sounds of Chinese characters, is used as the phonetic features. We design a hierarchical encoder to process the pinyin letters at the character-level and the sentence-level. Meanwhile, for the visual modality, we build character images with multiple channels as the graphic features, where each channel corresponds to a specific Chinese font. Then, we use ResNet (He et al., 2016) blocks to encode the images to get the graphic representation of characters.

With the representation of three different modalities, one challenge is how to fuse them into one compact multimodal representation. To this end, a selective modality fusion mechanism is designed to control how much information of each modality can flow to the mixed representation. Furthermore, as the pretrain-finetune procedure has been proven to be effective on various NLP tasks (Devlin et al., 2019; Dong et al., 2019; Sun et al., 2020), we propose to pretrain the phonetic and the graphic encoders by predicting the correct character given input in the corresponding modality.

We conduct experiments on the SIGHAN benchmarks (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015). By leveraging multimodal information, REALISE outperforms all previous state-of-the-art models by a large margin. Compared to previous methods using confusion set (Lee et al., 2019) to capture the character similarity relationships, such as the SOTA SpellGCN (Cheng et al., 2020), REALISE achieves an averaging 2.4% and 2.6% F1 improvements at detection-level and correction-level. Further analysis shows that our model performs better on the errors which are not defined in the handcrafted confusion sets. This indicates that leveraging the phonetic and graphic information of Chinese characters can better capture the easily-misused characters.

To summarize, the contributions of this paper include: (i) we propose to leverage phonetic and graphic information of Chinese characters besides textual semantics for the CSC task; (ii) we introduce the selective fusion mechanism to integrate multimodal information; (iii) we propose acoustic and visual pretraining tasks to further boost the model performance; (iv) to the best of our knowledge, the proposed REALISE model achieves the best results on the SIGHAN CSC benchmarks.

2 Related Work

2.1 Chinese Spell Checking

The CSC task is to detect and correct spelling errors in Chinese sentences. Early works design various rules to deal with different errors (Chang et al., 2015; Chu and Lin, 2015). Next, traditional machine learning algorithms are brought to this field, such as Conditional Random Field and Hidden Markov Model (Wang and Liao, 2015; Zhang et al., 2015). Then, neural-based methods have made great progress in CSC. Wang et al. (2018) treat the CSC task as a sequence labeling problem, and use a bidirectional LSTM to predict the correct characters. With the great success of large pretrained language models (e.g., BERT (Devlin et al., 2019)), Hong et al. (2019) propose the FASpell model, which use a BERT-based denoising autoencoder to generate candidate characters and uses some empirical measures to select the most likely ones. Besides, the Soft-Masked BERT model (Zhang et al., 2020) leverages a cascading architecture where GRU is used to detect the erroneous positions and BERT is used to predict correct characters.

Previous works (Yu and Li, 2014; Wang et al., 2019; Cheng et al., 2020) using handcrafted Chinese character confusion set (Lee et al., 2019) aim to correct the errors by discovering the similarity of the easily-misused characters. Wang et al. (2019) leverage the pointer network (Vinyals et al., 2015) by picking the correct character from the confusion set. Cheng et al. (2020) propose a SpellGCN model which models the character similarity through Graph Convolution Network (GCNs) (Kipf and Welling, 2016) on the confusion set. However, the character confusion set is predefined and fixed, which cannot cover all the similarity relations, nor can it distinguish the divergence in the similarity

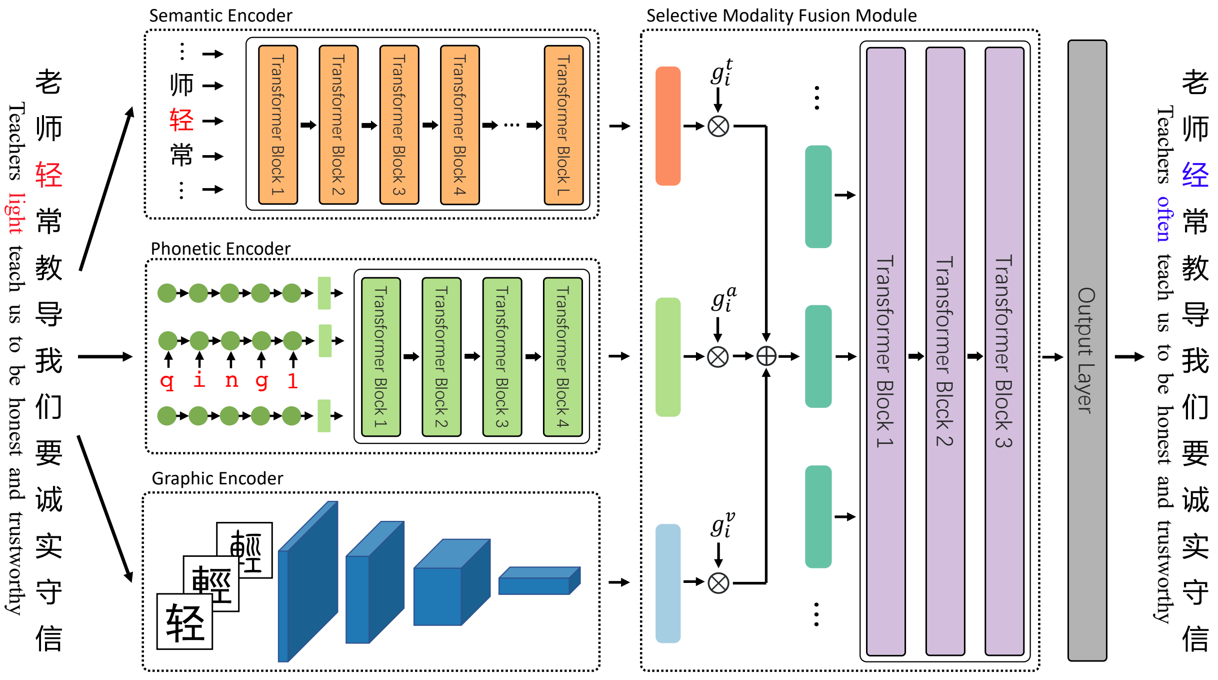

Figure 1: Architecture overview of the REALISE model. The semantic, phonetic and graphic encoders, are used to capture the information in textual, acoustic and visual modalities. The fusion module selectively fuses the information from three encoders. In the example input, to correct the erroneous character, “轻” (qīng, light), we need not only the contextual text information, but also the phonetic and graphic information of the character itself.

of Chinese characters. In this work, we discard the predefined confusion set and directly use the multimodal information to discover the subtle similarity relationship between all Chinese characters.

2.2 Multimodal Learning

There has been much research to integrate information from different modalities to achieve better performance. Tasks such as Multimodal Sentiment Analysis (Zadeh et al., 2016; Zhang et al., 2019), Visual Question Answering (Antol et al., 2015; Chao et al., 2018) and Multimodal Machine Translation (Hitschler et al., 2016; Barrault et al., 2018) have made much progress. Recently, multimodal pretraining models have been proposed, such as VL-BERT (Su et al., 2020), Unicoder-VL (Li et al., 2020), and LXMERT (Tan and Bansal, 2019). In order to incorporate the visual information of Chinese characters into language models, Meng et al. (2019) design a Tianzige-CNN to facilitate some NLP tasks, such as named entity recognition and sentence classification. To the best of our knowledge, this paper is the first work to leverage multimodal information to tackle the CSC task.

3 The REALISE Model

In this section, we introduce the REALISE model, which utilizes the semantic, phonetic, and graphic information to distinguish the similarities of Chinese characters and correct the spelling errors. As shown in Figure 1, multiple encoders are firstly employed to capture valuable information from textual, acoustic and visual modalities. Then, we develop a selective modality fusion module to obtain the context-aware multimodal representations. Finally, the output layer predicts the probabilities of error corrections.

3.1 The Semantic Encoder

We adopt BERT (Devlin et al., 2019) as the backbone of the semantic encoder. BERT provides rich contextual word representation with the unsupervised pretraining on large corpora.

The input tokens $\mathbf{X} = (x_1, \ldots, x_N)$ are first projected into $\mathbf{H}_0^t$ through the input embedding. Then the computation of Transformer (Vaswani et al., 2017) encoder layers can be formulated as:

$$\mathbf{H}_l^t = \text{Transformer}_l(\mathbf{H}_{l-1}^t), l \in [1, L] \quad (1)$$

where L is the number of Transformer layers. Each layer consists of a multi-head attention module and a feed-forward network with the residual connection (He et al., 2016) and layer normalization (Ba et al., 2016). The output of the last layer $\mathbf{H}^t = \mathbf{H}^t_L = (h^t_1, \ldots, h^t_N)$ is used as the contextualized semantic representation of the input tokens in textual modality.

3.2 The Phonetic Encoder

Hanyu Pinyin (pinyin) is the romanization for Chinese to "spell out" the sounds of characters. We use it to calculate the phonetic representation in this paper. The pinyin of a Chinese character consists of three parts: initial, final, and tone. The initial (21 in total) and final (39 in total) are written with letters in the English alphabet. The 5 kinds of tones (take the final "a" as an example, { ā, á, ǎ, à, a }) can be mapped into numbers $\{1, 2, 3, 4, 0\}$. Though the vocabulary size of pinyin for all Chinese characters is a fixed number, we use a sequence of letters in REALISE to capture the subtle phonetic difference between Chinese characters. For example, the pinyin of "中" (middle) and "棕" (brown) are "zhōng" and "zōng" respectively. The two characters have very similar sounds but quite different meanings. We thus represent pinyin as a symbol sequence, e.g., $\{z, h, o, n, g, 1\}$ for "中". We denote the pinyin of the i-th character in the input sentence as $\boldsymbol{p}_i = (p_{i,1}, \ldots, p_{i,|\boldsymbol{p}_i|})$, where $|\boldsymbol{p}_i|$ is the length of pinyin $\boldsymbol{p}_i$.

In REALISE, we design a hierarchical phonetic encoder, which consists of a character-level encoder and a sentence-level encoder.

The Character-level Encoder is to model the basic pronunciation and capture the subtle sound difference between characters. It is a single-layer uni-directional GRU (Cho et al., 2014), which encodes the pinyin of the i-th character x_i as:

$$\tilde{h}^a_{i,j} = \text{GRU}(\tilde{h}^a_{i,j-1}, E(p_{i,j})) \tag{2}$$

where $E(p_{i,j})$ is the embedding of the pinyin symbol $p_{i,j}$, and $\tilde{h}^a_{i,j}$ is the j-th hidden states of the GRU. The last hidden state is used as the character-level phonetic representation of x_i.

The Sentence-level Encoder is a 4-layer Transformer with the same hidden size as the semantic encoder. It is designed to obtain the contextualized phonetic representation for each Chinese character. As the independent phonetic vectors are not distinguished in order, we add the positional embedding to each vector in advance. Then, we pack these phonetic vectors together, and apply the Transformer layers to calculate the contextualized representation in acoustic modality, denoted as $\mathbf{H}^a = (h^a_1, h^a_2, ..., h^a_N)$. Note that owing to the Transformer architecture, this representation is also normalized.

3.3 The Graphic Encoder

We apply the ResNet (He et al., 2016) as the graphic encoder. The graphic encoder has 5 layers of ResNet blocks (denoted as ResNet5) followed by a layer normalization (Ba et al., 2016) operation. We formulate this procedure as follows:

$$\begin{aligned} \tilde{h}^v_i &= \text{ResNet5}(\mathbf{I}_i) \\ h^v_i &= \text{LayerNorm}(\tilde{h}^v_i) \end{aligned} \tag{3}$$

where $\mathbf{I}_i$ is the image of the i-th character x_i in the input sentence, and LayerNorm means layer normalization.

In order to extract graphic information effectively, each block in ResNet5 halves the width and height of the image, and increases the number of channels. Thus, the final output is a vector with the length equal to the number of output channels, i.e., both height and width become 1. Furthermore, we set the number of output channels to the hidden size in the semantic encoder for the follow-up modality fusion. We denote the representation in visual modality of the input sentence as $\mathbf{H}^v = (h^v_1, h^v_2, \ldots, h^v_N)$.

The character image of x_i is read from preset font files. Since the scripts of Chinese characters have evolved for thousands of years, to capture the graphic relationship between character as much as possible, we select three fonts, namely Gothic typefaces (黑体, hēitǐ) in both Simplified and Traditional Chinese, and Small Seal Script (小篆, xiǎozhuàn). The three fonts correspond to the three channels of the character images, whose size is set to 32 × 32 pixel.

3.4 Selective Modality Fusion Module

After applying the previously mentioned semantic, phonetic and graphic encoders, we get the representation vectors $\mathbf{H}^t$, $\mathbf{H}^a$ and $\mathbf{H}^v$ in textual, acoustic and visual modalities. To predict the final correct Chinese characters, we develop a selective modality fusion module to integrate these vectors in different modalities. This module fuses information in two levels, i.e., character-level and sentence-level.

First, for each modality, a selective gate unit is employed to control how much information can flow to the mixed multimodal representation. For example, if a character is misspelled due to its similar pronunciation to the correct one, then more information in the acoustic modality should flow into the mixed representation. The gate values are computed by a fully-connected layer followed by a sigmoid function. The inputs include the character representation of three modalities and the mean of the semantic encoder output $\mathbf{H}^t$ to capture the overall semantics of the input sentence. Formally, we denote the gate values for the textual, acoustic and visual modalities as g^t, g^a and g^v. The mixed multimodal representation $\tilde{h}_i$ of the i-th character is computed as follows:

$$\begin{aligned}
\bar{h}^t &= \frac{1}{N}\sum_{i=1}^{N} h_i^t \\
g_i^t &= \sigma(\mathbf{W}^t \cdot [h_i^t, h_i^a, h_i^v, \bar{h}^t] + b^t) \\
g_i^a &= \sigma(\mathbf{W}^a \cdot [h_i^t, h_i^a, h_i^v, \bar{h}^t] + b^a) \\
g_i^v &= \sigma(\mathbf{W}^v \cdot [h_i^t, h_i^a, h_i^v, \bar{h}^t] + b^v) \\
\tilde{h}_i &= g_i^t \cdot h_i^t + g_i^a \cdot h_i^a + g_i^v \cdot h_i^v
\end{aligned} \tag{4}$$

where $\mathbf{W}^t$, $\mathbf{W}^a$, $\mathbf{W}^v$, b^t, b^a, b^v are learnable parameters, σ is the sigmoid function, and $[\cdot]$ means the concatenation of vectors.

Then, we apply the Transformer to fully learn the semantic, phonetic and visual information at the sentence-level. The mixed representations of all characters are packed together into $\mathbf{H}_0 = [\tilde{h}_1, \tilde{h}_2, ..., \tilde{h}_N]$, and the probability distribution $\hat{y}_i$ of what the i-th character should be is derived as:

$$\begin{aligned}
\mathbf{H}_l &= \text{Transformer}_l(\mathbf{H}_{l-1}), l \in [1, L'] \\
\hat{y}_i &= \text{softmax}(\mathbf{W}^o h_i + b^o), h_i \in \mathbf{H}_{L'}
\end{aligned} \tag{5}$$

where L' is the number of Transformer layers, $\mathbf{W}^o$ and b^o are learnable parameters.

3.5 Acoustic and Visual Pretraining

While acoustic and visual information is essential to the CSC task, equally important is how to associate them with the correct character. In order to learn the acoustic-textual and visual-textual relationships, we propose to pretrain the phonetic and the graphic encoders.

For the phonetic encoder, we design an Input Method pretraining objective, that the encoder should recover the Chinese character sequence given the input pinyin sequence. This is what the Chinese input methods do. We add a linear layer on the top of the encoder to transform the hidden states to the probability distributions over the Chinese character vocabulary. We pretrain the phonetic encoder with the pinyin of the sentences with spelling errors in the training data, and make it recover the character sequences without spelling errors.

Training Set	#Sent	Avg. Length	#Errors
SIGHAN13	700	41.8	343
SIGHAN14	3,437	49.6	5,122
SIGHAN15	2,338	31.3	3,037
Wang271K	271,329	42.6	381,962
Total	277,804	42.6	390464
Test Set	**#Sent**	**Avg. Length**	**#Errors**
SIGHAN13	1,000	74.3	1,224
SIGHAN14	1,062	50.0	771
SIGHAN15	1,100	30.6	703
Total	3,162	50.9	2,698

Table 2: Statistics of the used datasets. All the training data are merged to train the REALISE model. The test sets are used separately to evaluate the model performance.

For the graphic encoder, we design an Optical Character Recognition (OCR) pretraining objective. Given the Chinese character images, the graphic encoder learns the visual information to predict the corresponding character over the Chinese character vocabulary. This is like what the OCR task does, but our recognition is only conducted on the character level and typed scripts. During the pretraining, we also add a linear layer on the top to perform the classification.

Finally, we load the pretrained weights of the semantic encoder, phonetic encoder, and graphic encoder, and conduct the final training process with the CSC training data.

4 Experiments

In this section, we introduce experimental details and results on the SIGHAN benchmarks (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015). We then verify the effectiveness of our model by conducting ablation studies and analyses.

4.1 Data and Metrics

Following previous works (Wang et al., 2019; Cheng et al., 2020), we use the SIGHAN training data and the generated pseudo data (Wang et al., 2018, denoted as Wang271K) as the training set. We evaluate our model on the SIGHAN test sets

Dataset	Method	Detection Level				Correction Level			
		Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
SIGHAN13	Sequence Labeling (Wang et al., 2018)	-	54.0	69.3	60.7	-	-	-	52.1
	FASpell (Hong et al., 2019)	63.1	76.2	63.2	69.1	60.5	73.1	60.5	66.2
	BERT (Cheng et al., 2020)	-	79.0	72.8	75.8	-	77.7	71.6	74.6
	SpellGCN (Cheng et al., 2020)	-	80.1	74.4	77.2	-	78.3	72.7	75.4
	SpellGCN† (Our reimplementation)	78.8	85.7	78.8	82.1	77.8	84.6	77.8	81.0
	BERT†	77.0	85.0	77.0	80.8	75.2	83.0	75.2	78.9
	ReaLiSe†	**82.7**	**88.6**	**82.5**	**85.4**	**81.4**	**87.2**	**81.2**	**84.1**
SIGHAN14	Sequence Labeling (Wang et al., 2018)	-	51.9	66.2	58.2	-	-	-	56.1
	FASpell (Hong et al., 2019)	70.0	61.0	53.5	57.0	69.3	59.4	52.0	55.4
	SpellGCN (Cheng et al., 2020)	-	65.1	69.5	67.2	-	63.1	67.2	65.3
	BERT	75.7	64.5	68.6	66.5	74.6	62.4	66.3	64.3
	ReaLiSe	**78.4**	**67.8**	**71.5**	**69.6**	**77.7**	**66.3**	**70.0**	**68.1**
SIGHAN15	KUAS (Chang et al., 2015)	53.2	57.5	24.6	34.4	51.5	53.7	21.1	30.3
	NTOU (Chu and Lin, 2015)	42.2	42.2	41.8	42.0	39.0	38.1	35.2	36.6
	NCTU-NTUT (Wang and Liao, 2015)	60.1	71.7	33.6	45.7	56.4	66.3	26.1	37.5
	HanSpeller++ (Zhang et al., 2015)	70.1	80.3	53.3	64.0	69.2	79.7	51.5	62.5
	LMC (Xie et al., 2015)	54.6	63.8	21.5	32.1	52.3	57.9	16.7	26.0
	Sequence Labeling (Wang et al., 2018)	-	56.6	69.4	62.3	-	-	-	57.1
	FASpell (Hong et al., 2019)	74.2	67.6	60.0	63.5	73.7	66.6	59.1	62.6
	Soft-Masked BERT (Zhang et al., 2020)	80.9	73.7	73.2	73.5	77.4	66.7	66.2	66.4
	SpellGCN (Cheng et al., 2020)	-	74.8	80.7	77.7	-	72.1	77.7	75.9
	BERT	82.4	74.2	78.0	76.1	81.0	71.6	75.3	73.4
	ReaLiSe	**84.7**	**77.3**	**81.3**	**79.3**	**84.0**	**75.9**	**79.9**	**77.8**

Table 3: The performance of our model and all baseline models on SIGHAN test sets. The "†" symbol means we apply post-processing (Section 4.2) to the model outputs on SIGHAN13. Results of ReaLiSe on all SIGHAN test sets outperforms all the corresponding baselines with a significance level $p < 0.05$.

in 2013, 2014 and 2015 (denoted as SIGHAN13, SIGHAN14 and SIGHAN15). Table 2 shows the data statistics. Originally, the SIGHAN datasets are in the Traditional Chinese. Following previous works (Wang et al., 2019; Cheng et al., 2020; Zhang et al., 2020), we convert them to the Simplified Chinese using the OpenCC tool[2].

Results are reported at the detection level and the correction level. At the detection level, a sentence is considered to be correct if and only if all the spelling errors in the sentence are detected successfully. At the correction level, the model must not only detect but also correct all the erroneous characters to the right ones. We report the accuracy, precision, recall and F1 scores on both levels.

4.2 Implementation Details

The ReaLiSe model is implemented using PyTorch framework (Paszke et al., 2019) with the Transformer library (Wolf et al., 2020). The architecture of the semantic encoder is same as the $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) model (i.e. 12 transformer layers with 12 attention heads, hidden size of 768). We initialize the semantic encoder with the weights of BERT-wwm model (Cui et al., 2019). For the phonetic sentence-level encoder, we set the number of layers to 4, and initialize its position embedding with BERT's position embedding. The selective modality fusion module has 3 transformer layers, i.e., $L' = 3$, and the prediction matrix $\mathbf{W}^o$ is tied with the word embedding matrix of the semantic encoder. All the embeddings and hidden states have the dimension of 768. We use the Pillow library to extract the Chinese character images. When processing the special tokens (e.g. `[CLS]` and `[SEP]` of BERT), we use the tensor with all zero values as their image inputs. We train our ReaLiSe model with the AdamW (Loshchilov and Hutter, 2017) optimizer for 10 epochs. The learning rate is set to 5e-5, the batch size is set to 32, and the model is trained with learning rate warming up and linear decay.

In the SIGHAN13 test set, the annotation quality is relatively poor, that quite a lot of the mixed usage of auxiliary "的", "地", and "得" are not annotated (Cheng et al., 2020). Therefore, a well-performed model may obtain bad scores on it. To alleviate the problem, Cheng et al. (2020) proposes

---

[2] `https://github.com/BYVoid/OpenCC`

to continue finetuning the model on the SIGHAN13 training set before testing. We argue that it's not a good practice because it reduces the model performance. Instead, we use a simple and effective post-processing method. We simply remove all the detected and corrected "的", "地", and "得" characters from the model output and then evaluate with the ground truth of SIGHAN13 test set.

4.3 Baselines

We compare ReaLiSe with the following baselines: **KUAS** (Chang et al., 2015), **NTOU** (Chu and Lin, 2015), **NCTU-NTUT** (Wang and Liao, 2015), **HanSpeller++** (Zhang et al., 2015), **LMC** (Xie et al., 2015) mainly utilize heuristics or traditional machine learning algorithms, such as n-gram language model, Conditional Random Field and Hidden Markov Model. **Sequence Labeling** (Wang et al., 2018) treats CSC as a sequence labeling problem and applies a BiLSTM model. **FASpell** (Hong et al., 2019) utilizes a denoising autoencoder (DAE) to generate candidate characters. **Soft-Masked BERT** (Zhang et al., 2020) utilizes the detection model to help the correction model learn the right context. **SpellGCN** (Cheng et al., 2020) incorporates the predefined character confusion sets to the BERT-based correction model through Graph Convolutional Networks (GCNs). **BERT** (Devlin et al., 2019) is to directly fine-tune the BERT$_{\text{BASE}}$ model with the CSC training data.

4.4 Main Results

Table 3 shows the evaluation scores at detection and correction levels on the SIGHAN 13/14/15 test sets. The ReaLiSe model performs significantly better than all the previous state-of-the-art models on all test sets. It can be seen that, by capturing valuable information from acoustic and visual modalities, ReaLiSe yields consistent gain with a large margin against BERT. Specifically, at the correction-level, ReaLiSe exceeds BERT by 5.2% F1 on SIGHAN13, 3.8% F1 on SIGHAN14, and 4.4% F1 on SIGHAN15. The results on SIGHAN13 are improved significantly with simple post-processing described in Section 4.2.

There are several successful applications of BERT on the CSC task, such as FASpell and SpellGCN, which also consider the Chinese character similarity. They attempt to calculate the similarity as the confidence of filtering candidates, or construct similarity graphs from predefined confusion sets. Instead, in our method, multiple encoders are directly applied to derive more informative representation from the acoustic and visual modalities. Compared with SpellGCN (Cheng et al., 2020), the SOTA CSC model, our ReaLiSe model achieves an averaging 2.4% F1 improvements at detection-level and an averaging 2.6% F1 improvements at correction-level. This indicates that, compared with other extensions of BERT, the explicit utilization of multimodal information of Chinese characters is more beneficial to the CSC task.

Model	Acc	Pre	Rec	F1
	Detection Level			
BERT	78.4	74.6	74.5	74.5
ReaLiSe	82.0	77.9	78.5	78.1
- Phonetic	81.2	76.4	77.7	77.0
- Graphic	81.4	77.3	77.2	77.2
- Multi-Fonts	81.2	76.3	77.9	77.0
- Pretraining	81.5	76.5	78.1	77.2
- Selective-Fusion	81.3	76.8	77.4	77.1
	Correction Level			
BERT	76.9	72.3	72.3	72.3
ReaLiSe	81.0	76.5	77.0	76.7
- Phonetic	80.2	74.8	76.1	75.4
- Graphic	80.5	75.8	75.6	75.7
- Multi-Fonts	80.3	74.9	76.4	75.5
- Pretraining	80.6	75.2	76.8	75.9
- Selective-Fusion	80.5	75.4	76.0	75.7

Table 4: Ablation results of the ReaLiSe model averaged on SIGHAN test sets. We apply the following changes to ReaLiSe: removing the phonetic encoder (- Phonetic), removing the graphic encoder (- Graphic), using only one font to build the graphic inputs (- Multi-Fonts), removing acoustic and visual pretraining (- Pretraining), replacing the selective modality fusion mechanism with simple summation (- Selective-Fusion).

With the simple post-processing as described in Section 4.2, results of each model on the SIGHAN13 test set are improved significantly. Compared with BERT and SpellGCN, we can see that, after the post-processing, the ReaLiSe model is ahead of all the baseline models.

4.5 Ablation Study

We explore the contribution of each component in ReaLiSe by conducting ablation studies with the following settings: 1) removing the phonetic encoder, 2) removing the graphic encoder, 3) using only one font (Gothic typefaces in Simplified Chinese) for the graphic encoder, 4) removing the acoustic and visual pretraining objectives, 5) replacing the selective modality fusion mechanism with simple summation.

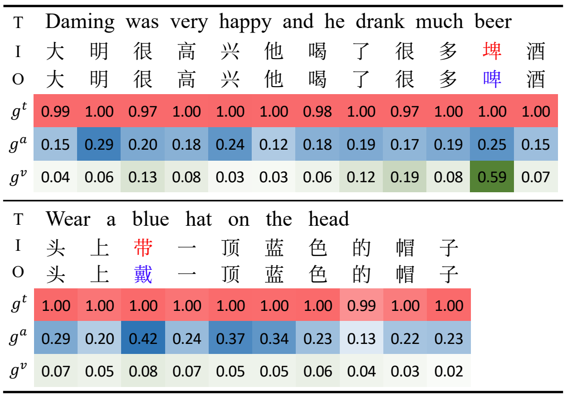

T	Daming was very happy and he drank much beer											
I	大	明	很	高	兴	他	喝	了	很	多	埤	酒
O	大	明	很	高	兴	他	喝	了	很	多	啤	酒
g^t	0.99	1.00	0.97	1.00	1.00	1.00	0.98	1.00	0.97	1.00	1.00	1.00
g^a	0.15	0.29	0.20	0.18	0.24	0.12	0.18	0.19	0.17	0.19	0.25	0.15
g^v	0.04	0.06	0.13	0.08	0.03	0.03	0.06	0.12	0.19	0.08	0.59	0.07

T	Wear a blue hat on the head									
I	头	上	带	一	顶	蓝	色	的	帽	子
O	头	上	戴	一	顶	蓝	色	的	帽	子
g^t	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00
g^a	0.29	0.20	0.42	0.24	0.37	0.34	0.23	0.13	0.22	0.23
g^v	0.07	0.05	0.08	0.07	0.05	0.05	0.06	0.04	0.03	0.02

Figure 2: Selective modality fusion visualization. "I" is the input sentence. "O" is the output of REALISE (also the ground truth), and "T" is the translation. g^t, g^a, g^v are the gate values for the textual, acoustic, and visual modality respectively. We highlight the wrong/correct characters in red/blue color.

Table 4 shows the averaged scores[3] on three SIGHAN test sets. The main motivation of this paper is to discover the character similarity relationships by incorporating the acoustic and visual information. If removing the phonetic or graphic encoder, we can see that the model performance drops at two levels but still outperforms BERT significantly. This suggests that the checking model can benefit from the multimodal information. No matter which component we remove, the performance of REALISE drops, which fully demonstrates the effectiveness of each part in our model.

4.6 Analysis of the Selective Modality Fusion Module

Figure 2 gives two examples to analyze the selective modality fusion module. In the first example, the acoustic and visual selective gate values of "埤", i.e. g^a and g^v, are much larger than most other characters since "埤(pí)" and "啤(pí)" have the same pronunciation and right radical "卑". This shows that the selective fusion module can judge whether to introduce phonetic or graphic information into the mixed representation. The second example shows a similar trend for the pronunciation of "带(dài)" and "戴(dài)". More selective fusion visualization can be found in the Appendix A.2.

Besides, we calculate the averaged gate values of erroneous characters for each modality on SIGHAN15. The largest one is the textual modality that the value is almost equal to 1.0. The second one is the acoustic modality that the averaged value is 0.334, and the smallest one is the visual modality that the value is 0.229. It means that the information from the semantic encoder is the most important for correcting the spelling errors. The acoustic modality is more important than the visual modality, which is consistent with the fact that the spelling errors caused by similar pronunciations are more frequent than errors caused by similar character shapes (Liu et al., 2010).

In:	我打算去法国流行，你要不要跟我一起去? I am going to popular to France, would you like to go with me?
Out:	我打算去法国旅行，你要不要跟我一起去? I am going to travel to France, would you like to go with me?
In:	回国之后，我跟快去你家。 After returning home, I will go to your house with.
Out:	回国之后，我很快去你家。 After returning home, I will go to your house soon.

Table 5: Examples of the input and output of our REALISE model. We highlight the wrong/correct characters in red/blue color.

4.7 Case Study

In the first example in Table 5, "流" is the erroneous character. If ignoring the Chinese character similarities, we can find that there are multiple candidate corrections to replace the "流" character. For instance, we can replace it with "游" and the English translation is "*I am going to parade in France*". However, the REALISE's output is the best correction, because "流(liú)" and "旅(lˇü)" have a similar pronunciation. In the second example, not only the phonetic information, but also the visual information is important for correcting "跟(gēn)" to "很(hěn)". In detail, the two characters share the same final "en" in pronunciation, and have the same right radical "艮".

The errors in the above examples are not corrected by SpellGCN, since they are not defined as confusing character pairs in the handcrafted confusion sets (Lee et al., 2019). Specifically, in the SIGHAN15 test set, there are 16% erroneous-corrected character pairs not in the predefined confusion sets. SpellGCN corrects 64.6% of them but REALISE performs better with 73.5% correction. Besides, for the easily-confused pairs in the predefined sets, SpellGCN corrects 82.5% of them and REALISE corrects 85.8%. This indicates that leveraging multimodal information of Chinese charac-

---

[3] Full ablation results can be found in the Appendix A.1.

ters helps the model generalize better in capturing the character similarity relationships.

5 Conclusion

In this paper, we propose a model called REALISE for Chinese spell checking. Since the spelling errors in Chinese are often semantically, phonetically or graphically similar to the correct characters, REALISE leverages information in textual, acoustic and visual modalities to detect and correct the errors. The REALISE model captures information in these modalities using tailored semantic, phonetic and graphic encoders. Besides, a selective modality fusion mechanism is proposed to control the information flow of these modalities. Experiments on the SIGHAN benchmarks show that the proposed REALISE outperforms the baseline models using only textual information by a large margin, which verifies that leveraging acoustic and visual information helps the Chinese spell checking task.

References

Haithem Afli, Zhengwei Qiu, Andy Way, and Páraic Sheridan. 2016. Using smt for ocr error correction of historical texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 962–966.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2425–2433. IEEE Computer Society.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliott, and Stella Frank. 2018. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 304–323. Association for Computational Linguistics.

Tao-Hsing Chang, Hsueh-Chih Chen, and Cheng-Han Yang. 2015. Introduction to a proofreading tool for chinese spelling check task of SIGHAN-8. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 50–55. Association for Computational Linguistics.

Wei-Lun Chao, Hexiang Hu, and Fei Sha. 2018. Being negative but constructively: Lessons learnt from creating better visual question answering datasets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 431–441. Association for Computational Linguistics.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. Spellgcn: Incorporating phonological and visual similarities into language models for chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 871–881. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Wei-Cheng Chu and Chuan-Jie Lin. 2015. NTOU chinese spelling check system in sighan-8 bake-off. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 137–143. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese BERT. *CoRR*, abs/1906.08101.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring–an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1072–1077.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366, Beijing, China. Coling 2010 Organizing Committee.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal pivots for image caption translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. Faspell: A fast, adaptable, simple, powerful chinese spell checker based on dae-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 160–169. Association for Computational Linguistics.

Norman Jerry. 1988. Chinese (cambridge language surveys).

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lung Hao Lee, Wun Syuan Wu, Jian Hong Li, Yu Chi Lin, and Yuen Hsien Tseng. 2019. Building a confused character set for chinese spell checking. In *27th International Conference on Computers in Education, ICCE 2019*, pages 703–705. Asia-Pacific Society for Computers in Education.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. 2020. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11336–11344. AAAI Press.

C.-L. Liu, M.-H. Lai, K.-W. Tien, Y.-H. Chuang, S.-H. Wu, and C.-Y. Lee. 2011. Visually and phonologically similar characters in incorrect chinese words: Analyses, identification, and applications. *ACM Transactions on Asian Language Information Processing*, 10(2).

Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. 2010. Visually and phonologically similar characters in incorrect simplified Chinese words. In *Coling 2010: Posters*, pages 739–747, Beijing, China. Coling 2010 Organizing Committee.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Bruno Martins and Mário J. Silva. 2004. Spelling correction for search engine queries. In *Advances in Natural Language Processing*, pages 372–383, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 2742–2753.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. VL-BERT: pre-training of generic visual-linguistic representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975. AAAI Press.

Ryuichi Tachibana and Mamoru Komachi. 2016. Analysis of english spelling errors in a word-typing game. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 385–390.

Hao Tan and Mohit Bansal. 2019. LXMERT: learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 5099–5110. Association for Computational Linguistics.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to SIGHAN 2015 bake-off for chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 32–37. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems*, 28:2692–2700.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2517–2527. Association for Computational Linguistics.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5780–5785. Association for Computational Linguistics.

Yih-Ru Wang and Yuan-Fu Liao. 2015. Word vector/conditional random field-based chinese spelling error detection for SIGHAN-2015 evaluation. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 46–49. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 35–42. Asian Federation of Natural Language Processing.

Weijian Xie, Peijie Huang, Xinrui Zhang, Kaiduo Hong, Qiang Huang, Bingzhou Chen, and Lei Huang. 2015. Chinese spelling check system based on n-gram model. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 128–136. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of SIGHAN 2014 bake-off for chinese spelling check. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing, Wuhan, China, October 20-21, 2014*, pages 126–132. Association for Computational Linguistics.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intell. Syst.*, 31(6):82–88.

Dong Zhang, Shoushan Li, Qiaoming Zhu, and Guodong Zhou. 2019. Effective sentiment-relevant word selection for multi-modal sentiment analysis in spoken language. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 148–156.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 882–890. Association for Computational Linguistics.

Shuiyuan Zhang, Jinhua Xiong, Jianpeng Hou, Qiao Zhang, and Xueqi Cheng. 2015. Hanspeller++: A unified framework for chinese spelling correction. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 38–45. Association for Computational Linguistics.

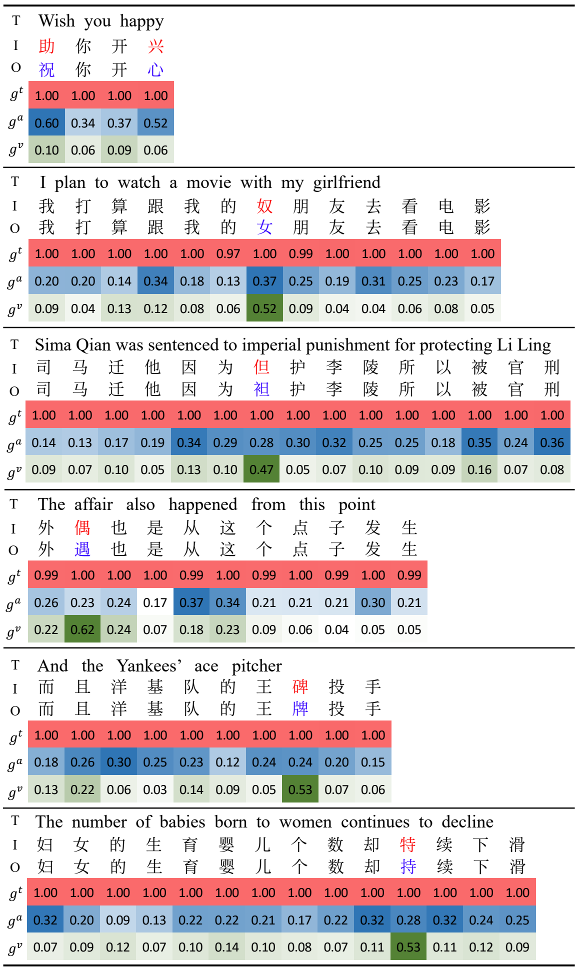

T	Wish you happy			
I	助	你	开	兴
O	祝	你	开	心
g^t	1.00	1.00	1.00	1.00
g^a	0.60	0.34	0.37	0.52
g^v	0.10	0.06	0.09	0.06

T	I plan to watch a movie with my girlfriend												
I	我	打	算	跟	我	的	奴	朋	友	去	看	电	影
O	我	打	算	跟	我	的	女	朋	友	去	看	电	影
g^t	1.00	1.00	1.00	1.00	1.00	0.97	1.00	0.99	1.00	1.00	1.00	1.00	1.00
g^a	0.20	0.20	0.14	0.34	0.18	0.13	0.37	0.25	0.19	0.31	0.25	0.23	0.17
g^v	0.09	0.04	0.13	0.12	0.08	0.06	0.52	0.09	0.04	0.04	0.06	0.08	0.05

T	Sima Qian was sentenced to imperial punishment for protecting Li Ling														
I	司	马	迁	他	因	为	但	护	李	陵	所	以	被	官	刑
O	司	马	迁	他	因	为	袒	护	李	陵	所	以	被	官	刑
g^t	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
g^a	0.14	0.13	0.17	0.19	0.34	0.29	0.28	0.30	0.32	0.25	0.25	0.18	0.35	0.24	0.36
g^v	0.09	0.07	0.10	0.05	0.13	0.10	0.47	0.05	0.07	0.10	0.09	0.09	0.16	0.07	0.08

T	The affair also happened from this point										
I	外	偶	也	是	从	这	个	点	子	发	生
O	外	遇	也	是	从	这	个	点	子	发	生
g^t	0.99	1.00	1.00	1.00	0.99	1.00	0.99	1.00	0.99	1.00	0.99
g^a	0.26	0.23	0.24	0.17	0.37	0.34	0.21	0.21	0.21	0.30	0.21
g^v	0.22	0.62	0.24	0.07	0.18	0.23	0.09	0.06	0.04	0.05	0.05

T	And the Yankees' ace pitcher									
I	而	且	洋	基	队	的	王	碑	投	手
O	而	且	洋	基	队	的	王	牌	投	手
g^t	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
g^a	0.18	0.26	0.30	0.25	0.23	0.12	0.24	0.24	0.20	0.15
g^v	0.13	0.22	0.06	0.03	0.14	0.09	0.05	0.53	0.07	0.06

T	The number of babies born to women continues to decline													
I	妇	女	的	生	育	婴	儿	个	数	却	特	续	下	滑
O	妇	女	的	生	育	婴	儿	个	数	却	持	续	下	滑
g^t	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
g^a	0.32	0.20	0.09	0.13	0.22	0.22	0.21	0.17	0.22	0.32	0.28	0.32	0.24	0.25
g^v	0.07	0.09	0.12	0.07	0.10	0.14	0.10	0.08	0.07	0.11	0.53	0.11	0.12	0.09

Figure 3: Selective modality fusion visualization. "I" is the input sentence. "O" is the output of REALISE (also the ground truth), and "T" is the translation. g^t, g^a, g^v are the gate values for the textual, acoustic, and visual modality respectively. We highlight the wrong/correct characters in red/blue color.

A Appendix

A.1 Ablation

We conduct an ablation study to verify the effectiveness of the proposed method. Table 6 (on Page 13 in Appendix) shows the detailed ablation results on each SIGHAN test set, where the following settings are conducted:

1. **- Phonetic**: removing the phonetic encoder.
2. **- Graphic**: removing the graphic encoder.
3. **- Multi-Fonts**: using only one font (Gothic typefaces in Simplified Chinese) for the graphic encoder.
4. **- Pretraining**: removing the acoustic and visual pretraining objectives.
5. **- Selective-Fusion**: replacing the selective modality fusion mechanism with simple summation.

We can see that, when we remove anything from our model, the REALISE performance drops consistently, and it drops most apparently in the SIGHAN14 test set. These results suggest that each part of our model is an effective means for boosting the checking performance.

A.2 Selective Modality Fusion Visualization

We show more examples in Figure 3. We can see that, if the misused characters are phonetically similar to the correct ones, the acoustic gate values tend to be larger, and if they are graphically similar, the visual gate values are larger.

Dataset	Method	Detection Level				Correction Level			
		Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
SIGHAN13	BERT	77.0	85.0	77.0	80.8	75.2	83.0	75.2	78.9
	ReaLiSe	82.7	88.6	82.5	85.4	81.4	87.2	81.2	84.1
	- Phonetic	82.4	87.4	82.3	84.8	81.2	86.1	81.1	83.5
	- Graphic	82.1	88.1	82.1	85.0	80.9	86.7	80.8	83.7
	- Multi-Fonts	82.2	87.5	82.2	84.8	81.2	86.4	81.2	83.7
	- Pretraining	82.8	88.2	82.7	85.4	81.4	86.7	81.3	83.9
	- Selective-Fusion	82.0	87.3	82.0	84.6	81.0	86.2	81.0	83.5
SIGHAN14	BERT	75.7	64.5	68.6	66.5	74.6	62.4	66.3	64.3
	ReaLiSe	78.4	67.8	71.5	69.6	77.7	66.3	70.0	68.1
	- Phonetic	77.1	65.5	69.2	67.3	76.3	63.8	67.5	65.6
	- Graphic	78.0	67.3	69.6	68.4	77.1	65.6	67.9	66.7
	- Multi-Fonts	76.9	65.0	69.6	67.2	76.2	63.6	68.1	65.7
	- Pretraining	77.5	65.6	70.4	67.9	76.7	64.0	68.7	66.2
	- Selective-Fusion	77.6	66.5	69.0	67.7	76.9	64.8	67.3	66.0
SIGHAN15	BERT	82.4	74.2	78.0	76.1	81.0	71.6	75.3	73.4
	ReaLiSe	84.7	77.3	81.3	79.3	84.0	75.9	79.9	77.8
	- Phonetic	84.2	76.2	81.7	78.9	83.3	74.5	79.9	77.1
	- Graphic	84.3	76.6	79.9	78.2	83.5	75.0	78.2	76.6
	- Multi-Fonts	84.5	76.5	81.9	79.1	83.5	74.6	79.9	77.1
	- Pretraining	84.2	75.7	81.3	78.4	83.7	74.9	80.4	77.5
	- Selective-Fusion	84.4	76.8	81.2	78.9	83.6	75.4	79.7	77.5

Table 6: Ablation results of the ReaLiSe model on each SIGHAN dataset.

TransSum: Translating Aspect and Sentiment Embeddings for Self-Supervised Opinion Summarization

Ke Wang, Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
`{wangke17,wanxiaojun}@pku.edu.cn`

Abstract

In this paper, we propose a novel self-supervised opinion summarization framework TransSum, which models opinion summaries as translations operating on the low-dimensional aspect and sentiment embedding spaces. Specifically, we propose two contrastive objectives to learn the crucial aspect and sentiment embeddings of reviews, by taking advantage of the intra- and inter-group invariances that have not been considered in previous studies. Furthermore, these embeddings can be used to reduce opinion redundancy and construct highly relevant reviews-summary pairs to train a supervised multi-input opinion summarization model. Experimental results on three different domains show that TransSum outperforms several strong baselines in generating informative, relevant and low-redundant summaries, unveiling the effectiveness of our approach.

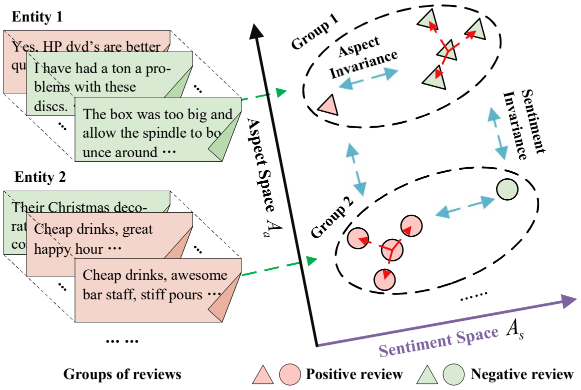

Figure 1: The proposed TransSum targets at learning corresponding aspect and sentiment embeddings for reviews (green arrows) through contrastive learning based on the aspect and sentiment invariances (blue arrows). These embeddings are used to construct reviews-summary pairs of high relevance (red arrows), so as to train a supervised multi-input opinion summarization model. Best view in color.

1 Introduction

Opinion summarization, which focuses on automatically generating summaries that reflect salient opinion information expressed in a group of documents (e.g., user reviews of a product in Figure 1), has been receiving great attention due to its usefulness and effectiveness for displaying massive opinion texts (Ku et al., 2006; Cheung et al., 2009; Chu and Liu, 2019). For example, a representative review summary of a product can not only replace large amounts of reviews for potential customers to read, but also provide more explanations than a simple overall sentiment rating, such as “ What is the biggest complaint on the iPod ‘screen’ ?”.

However, compared with supervised summarization in the domain of news articles, the annotated training data for opinion summarization is expensive to acquire. Due to the lack of gold-standard summaries for training, most existing works focus on unsupervised opinion summarization and treat it as a normal multi-document summarization task. They either struggle to reduce the opinion redundancy efficiently or output summaries lacking relevance to input reviews. Particularly, many previous studies focus on extractive approaches (Paul et al., 2010; Fabbrizio et al., 2014; Rossiello et al., 2017; Narayan et al., 2019), which copy texts from the input reviews but tend to be redundant and less informative (Chu and Liu, 2019). Some recently proposed abstractive methods are based on unsupervised representation learning, such as autoencoder (Chu and Liu, 2019; Amplayo and Lapata, 2019; Brazinskas et al., 2020a) or variational autoencoder (Brazinskas et al., 2020b; Angelidis et al., 2020), but mainly focus on the content transformation within each group of reviews. Other studies aim to create synthetic reviews-summary pairs to train a supervised multi-document summarization model (Amplayo and Lapata, 2019; Brazinskas et al., 2020b; Amplayo and Lapata, 2020; Amplayo

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 729–742
August 1–6, 2021.

et al., 2021), such as sampling a review from a corpus of product reviews and treating it as a summary of the remaining reviews, but such settings may lack rationality to guarantee the relevance of reviews and constructed pseudo-summaries.

In an effort to overcome these challenges, we propose a novel self-supervised framework for opinion summarization, TransSum, which consists of two main modules and does not require any gold summaries for training. (i) In the translation-based review modeling module, we expect to represent reviews with only their corresponding aspect and sentiment embeddings (as shown in Figure 1) with the purpose of reducing unnecessary information. We decompose each review into the aspect and sentiment embeddings through reconstruction and contrastive learning (van den Oord et al., 2018; He et al., 2020) based on two novel intra- and inter-group invariances: First, the real-world reviews in a group may discuss various opinions covering different aspects, but they are dependent with a specific entity (e.g., reviews about a specific product). Hence, the aspect information of the reviews in the same group should be closer than that of different groups (i.e, aspect invariance in Figure 1), that is, the distances between intra-group reviews should be less than the ones between inter-group reviews in the aspect embedding space. Second, the sentiment information of reviews with the same sentiment label should be closer than that of different sentiment labels (i.e, sentiment invariance in Figure 1), that is, the distances between reviews with the same sentiment should be less than the ones between reviews with different sentiments in the sentiment embedding space. (ii) In our multi-input opinion summarization module, we reduce opinion redundancy by combining similar embeddings, and use reviews with similar aspects (embeddings) to construct reviews-summary pairs of high relevance, which are used to train a supervised multi-input summarization model.

We conduct extensive experiments to show the superiority of our method. Experimental results on three different domains show that our method outperforms several strong baselines in generating informative, relevant, low-redundant and fluent summaries. We also perform ablation studies to analyze the effectiveness of the modules in our method.

In summary, our main contributions are:

- To the best of our knowledge, we are the first to generate opinion summaries from only the aspect and sentiment embeddings, which unlocks the critical bottleneck for unsupervised opinions modeling and takes a step forward towards more complex and controllable designs.
- We propose a novel self-supervised framework (TransSum) to generate opinion summaries without access to expensive annotations by disentangling reviews into aspect and sentiment embeddings and automatically constructing highly relevant reviews-summary pairs for model training.
- Experimental results on three domains show that our approach outperforms several strong baselines, especially in terms of relevance and non-redundancy.

2 TransSum

2.1 Overview

As aforementioned, a good opinion summary needs to cover major opinions/sentiments on different aspects of the entity (e.g., a movie, product, business) discussed in a group of reviews. Inspired by this observation, we propose a self-supervised framework (titled TransSum), aiming to generate opinion summaries without access to expensive annotations by interpreting them as translations operating on the aspect and sentiment embeddings.

As noted in a recent theoretical model of importance in summarization (Peyrard, 2019), a good summary should meet three requirements: (i) minimum redundancy, (ii) maximum relevance with the input document(s), and (iii) maximum informativeness. Based on the observation that reviews are usually created to express users' sentiments on certain aspects of a specific entity (e.g., the price and battery of a PC), we reasonably define informativeness, the amount of new information contained in the opinion summary relative to the background knowledge, as the aspect and sentiment information. The purpose is to reduce unnecessary information in the opinion summary, such as personal information or other irrelevant details.

Specifically, TransSum consists of two main components: **(1)** A translation-based review modeling module that learns only aspect and sentiment embeddings from each review for opinion summarization, to keep only the key and useful information (requirement iii). The aspect and sentiment embeddings of reviews are learned through reconstruction and two contrastive objectives, which take advantage of aspect and sentiment invariance of intra- and inter-group reviews (detailed in Sec 2.3). **(2)** A multi-input opinion summarization module that learns to generate the summary from the redundancy-reduced combination of the aspect and sentiment embeddings of input reviews (requirement i). It is trained by synthetic reviews-summary pairs of high relevance (requirement ii), which are constructed based on the assumption that reviews with the same aspect information (embeddings) are likely to express similar opinions (detailed in Sec 2.4).

2.2 Notations

More formally, let $\mathbb{D}$ denote a review corpus in a domain (e.g., Products' reviews), which consists of m groups of reviews. For each group $\boldsymbol{G}$, we assume that it contains n reviews $\{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_i \cdots, \boldsymbol{r}_n\}$ about a specific entity e (e.g., a product), n is not a fixed number. For each review $\boldsymbol{r}_i$ in $\boldsymbol{G}$, we define its number of tokens $\boldsymbol{r}_i$ as $|\boldsymbol{r}_i|$, that is, $\boldsymbol{r}_i = \{\boldsymbol{r}_i^{(1)}, \cdots, \boldsymbol{r}_i^{(|\boldsymbol{r}_i|)}\}$, and use $\boldsymbol{r}_{-i} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_{i-1}, \boldsymbol{r}_{i+1}, \cdots, \boldsymbol{r}_n\}$ to represent the remaining $n-1$ reviews. Each review has a binary sentiment label x (i.g., positive or negative), which indicates the overall sentiment polarity of the review. The aspect and sentiment embeddings of $\boldsymbol{r}_i$ are denoted as $\boldsymbol{a}_i \in \mathbb{R}^{|\boldsymbol{r}_i| \times k}$ and $\boldsymbol{s}_i \in \mathbb{R}^{|\boldsymbol{r}_i| \times k}$. E and D are encoder and decoder respectively.

The goal of opinion summarization is to generate a summary $\boldsymbol{y}$ that covers opinions mentioned in the group of reviews, in other words, $\boldsymbol{y}$ can be considered *"a representative review"* that can replace the group of reviews $\{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_i \cdots, \boldsymbol{r}_n\}$ in terms of informativeness. Note that we cannot access gold-standard opinion summaries for each group of reviews, as the human-annotated summaries do not exist in most domains.

2.3 Translation-Based Review Modeling

The translation-based review modeling module aims to learn aspect and sentiment embeddings for reviews (the left block in Figure 2).

For each review $\boldsymbol{r}_i$ in the group $\boldsymbol{G}$, we encode it using a Transformer (Vaswani et al., 2017) encoder E, and the output encoding $\boldsymbol{h}_i \in \mathbb{R}^{|\boldsymbol{r}_i| \times k}$ is:

$$\boldsymbol{h}_i = E(\boldsymbol{r}_i) = (\boldsymbol{h}_i^{(1)}, \cdots, \boldsymbol{h}_i^{(|\boldsymbol{r}_i|)}), \qquad (1)$$

where k is the embedding dimension. Inspired by Zhong et al. (2019), we initialize the token embeddings of E with the ones of the BERT-base model (Devlin et al., 2019). Then we use projection matrices $\boldsymbol{A}_a \in \mathbb{R}^{k \times k}$ and $\boldsymbol{A}_s \in \mathbb{R}^{k \times k}$ to project $\boldsymbol{h}_i$ to the aspect and sentiment spaces as $\boldsymbol{a}_i$ and $\boldsymbol{s}_i$ (the blue and red squares in Figure 2), respectively.

$$\boldsymbol{a}_i = \boldsymbol{h}_i \boldsymbol{A}_a = (\boldsymbol{a}_i^{(1)}, \cdots, \boldsymbol{a}_i^{(|\boldsymbol{r}_i|)}) \qquad (2)$$

$$\boldsymbol{s}_i = \boldsymbol{h}_i \boldsymbol{A}_s = (\boldsymbol{s}_i^{(1)}, \cdots, \boldsymbol{s}_i^{(|\boldsymbol{r}_i|)}) \qquad (3)$$

For later use, we further denote $\hat{\boldsymbol{a}}_i = \frac{1}{|\boldsymbol{r}_i|}\sum_{j=1}^{|\boldsymbol{r}_i|} \boldsymbol{a}_i^{(j)}$ ($\hat{\boldsymbol{a}}_i \in \mathbb{R}^k$) and $\hat{\boldsymbol{s}}_i = \frac{1}{|\boldsymbol{r}_i|}\sum_{j=1}^{|\boldsymbol{r}_i|} \boldsymbol{s}_i^{(j)}$ ($\hat{\boldsymbol{s}}_i \in \mathbb{R}^k$) to represent the mean vectors of the embeddings, repectively.

Translation-Based Reconstruction: We assume that each review is "an opinion summary" of the user's intention and attitude, and model the review as the translation from aspect and sentiment embeddings, that is, $\boldsymbol{c}_i = \boldsymbol{a}_i + \boldsymbol{s}_i$ (the yellow square in Figure 2):

$$\boldsymbol{c}_i = \{\boldsymbol{a}_i^{(1)} + \boldsymbol{s}_i^{(1)}, \cdots, \boldsymbol{a}_i^{(|\boldsymbol{r}_i|)} + \boldsymbol{s}_i^{(|\boldsymbol{r}_i|)}\}, \qquad (4)$$

To maximize informativeness and reduce unnecessary information, we hope to reconstruct $\boldsymbol{r}_i$ from only the embeddings $\boldsymbol{c}_i$ with a decoder D. The reconstruction loss is:

$$\mathcal{L}_{rec} = \mathbb{E}_{\boldsymbol{r}_i \sim \mathbb{D}}[\ell(D(\boldsymbol{c}_i), \boldsymbol{r}_i)], \qquad (5)$$

where ℓ is the cross entropy loss (de Boer et al., 2005) and D is a Transformer (Vaswani et al., 2017) decoder with cross-attention on $\boldsymbol{c}_i$. Following previous arts (Amplayo et al., 2021; ElSahar et al., 2020), we adopt label smoothing method (Szegedy et al., 2016) on $\boldsymbol{r}_i$ instead of computing with categorical distributions.

Contrastive Learning of Aspect and Sentiment Embeddings: We perform contrastive learning to learn the aspect and sentiment embeddings, based on the following two contrastive objectives: (i) the aspect embeddings of intra-group reviews should be "closer" to each other than the ones of inter-group reviews, and (ii) the sentiment embeddings of reviews with the same sentiment label should be "closer" to each other than the ones of reviews with different sentiment labels, even if they are in different groups.

More concretely, for the aspect embedding $\boldsymbol{a}_i$, we except to make its similarity with a "similar" sample $\boldsymbol{a}_i^+$ far greater than the one with a "dissimilar" sample $\boldsymbol{a}_i^-$, that is, $Sim(\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{a}}_i{}^+) \gg$

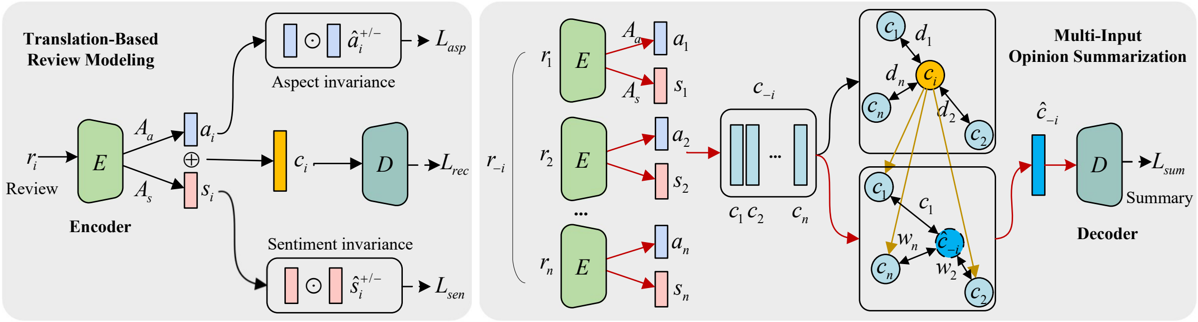

Figure 2: Architecture of TransSum, which consists of two main components: (1) a translation-based review modeling module that learns aspect and sentiment embeddings, and (2) a multi-input opinion summarization module that learns to generate summaries that are low-redundant and highly relevant to the input reviews. The encoder and decoder are shared, and the red arrows indicate the data flow in the inference phase. Best view in color.

$Sim(\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{a}}_i{}^-)$. $\boldsymbol{a}_i^+$ is the aspect embedding of a review sampled from the same group, and $\boldsymbol{a}_i^-$ is the aspect embedding of a review sampled from other groups. In this work, we use the dot product between embeddings to measure similarity (i.e., Sim), which can be regarded as a measure of the angle between the two embeddings in the vector space. As a consequence, the aspect-based contrastive objective is:

$$\mathcal{L}_{asp} = -\mathbb{E}_{\boldsymbol{r}_i\sim\mathbb{D}}\Big[\log \frac{\exp(Sim(\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{a}}_i{}^+))}{\exp(Sim(\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{a}}_i{}^+)) + \exp(Sim(\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{a}}_i{}^-))}\Big], \tag{6}$$

As for the sentiment embedding $\boldsymbol{s}_i$, the "similar" sample $\boldsymbol{s}_i^+$ is the sentiment embedding of a review sampled from different groups but with the same sentiment label, and the "dissimilar" sample $\boldsymbol{s}_i^-$ is the sentiment embedding of a review sampled from different groups and with a different sentiment label. Hence, the sentiment-based contrastive objective is defined as follows:

$$\mathcal{L}_{sen} = -\mathbb{E}_{\boldsymbol{r}_i\sim\mathbb{D}}\Big[\log \frac{\exp(Sim(\hat{\boldsymbol{s}}_i, \hat{\boldsymbol{s}}_i{}^+))}{\exp(Sim(\hat{\boldsymbol{s}}_i, \hat{\boldsymbol{s}}_i{}^+)) + \exp(Sim(\hat{\boldsymbol{s}}_i, \hat{\boldsymbol{s}}_i{}^-))}\Big]. \tag{7}$$

To the best of our knowledge, we are the first to go beyond the intra-group information modeling by further considering the inter-group level contrastive learning of aspect and sentiment embeddings.

To further enlarge the disagreements among the aspect/sentiment projection matrix and reduce the redundancy of parameters, we additionally add a regularization loss to encourage uniqueness:

$$\mathcal{L}_{reg} = ||\boldsymbol{A}_a^T \boldsymbol{A}_a - \boldsymbol{I}||^2 + ||\boldsymbol{A}_s^T \boldsymbol{A}_s - \boldsymbol{I}||^2, \tag{8}$$

where $\boldsymbol{I}$ is the identity matrix.

2.4 Multi-Input Opinion Summarization

After learning aspect and sentiment low-dimensional embeddings of reviews, we can construct reviews-summary pairs of high relevance based on the similarity of aspect embeddings, so as to train a supervised multi-input opinion summarization model (the right block in Figure 2).

Although real-world reviews in a group discuss various viewpoints covering different aspects under consideration, they are in fact focused on the same entity. In other words, they may repeat discussions about certain aspects many times, and may also include their own unique aspects. However, opinions on the same aspects are likely to be the same in real scenarios, e.g., knowing that most users complain about the high price of a product, the next price-focused review is likely to give a negative view. Based on this observation, we expect to reduce redundancy in similar aspects and use reviews with similar aspects to construct a high-quality data set whose reviews-summary data pairs are highly relevant.

High-Relevance Dataset Creation: We expect to find a subset of $\boldsymbol{r}_{-i}$ in which reviews are similar to $\boldsymbol{r}_i$ in the aspect embedding space, and use $\boldsymbol{r}_i$ as the target (pseudo) opinion summary of this subset of reviews. We have noticed that in real reviews, the majority of views on the same aspect are consistent with each other, so we believe most reviews-summary pairs created in this way can be used for training a model to capture and summarize the major opinions of the input reviews. More sophisticated ways for dataset creation will be left for further study.

In practice, we assign a weight w to each review in $\boldsymbol{r}_{-i}$, that is, assigning small values to low-relevance reviews instead of looking for a subset of only high-relevance reviews (as shown by the yellow arrows in Figure 2). For each review $\boldsymbol{r}_j$ in $\boldsymbol{r}_{-i}$, we calculate the distance between it and $\boldsymbol{r}_i$ in the aspect embedding space as:

$$d_j = Sim(\hat{\boldsymbol{a}}_j, \hat{\boldsymbol{a}}_i), d_j \in \boldsymbol{d}_{-i}, \tag{9}$$

where $\boldsymbol{d}_{-i} = \{d_1, \cdots, d_{i-1}, d_{i+1}, \cdots, d_n\}$. Then we construct the reviews-summary pair $< \boldsymbol{r}_{-i}, \boldsymbol{r}_i >$ with the following weights $\boldsymbol{w}_{-i}$, which will be used later:

$$w_j = \frac{\exp(d_j)}{\sum_{d_z \in \boldsymbol{d}_{-i}} \exp(d_z)}, w_j \in \boldsymbol{w}_{-i}. \tag{10}$$

Note that some previous arts (Brazinskas et al., 2020b; ElSahar et al., 2020; Brazinskas et al., 2020a; Amplayo et al., 2021) adopted a leave-one-out self-supervision setting (Besag, 1975) similar to ours. But they did not take into account the relevance between each review and the pseudo summary, which can be considered as our special case with a uniform distribution $\boldsymbol{w}_{-i} = (\frac{1}{n-1}, \cdots, \frac{1}{n-1})$.

Embedding-Based Redundancy Reduction: Aside from creating a high-relevance synthetic dataset, we can use the learned embeddings to reduce redundancy. We regard the embedding differences of different reviews as their natural variation, and perform a weighted pooling operation to remove redundant information (similar embeddings). Therefore, we obtain $\hat{\boldsymbol{c}}_{-i}$ based on multiple inputs $\{w_1\boldsymbol{c}_1, \cdots, w_{i-1}\boldsymbol{c}_{i-1}, w_{i+1}\boldsymbol{c}_{i+1}, \cdots, w_n\boldsymbol{c}_n\}$. Note that $\boldsymbol{w}$ is a uniform distribution in the inference phase, that is, the weight of each input review is equal.

Finally, we generate the opinion summary of $\boldsymbol{r}_{-i}$ and the summarization loss $\mathcal{L}_{sum}$ is:

$$\mathcal{L}_{sum} = \mathbb{E}_{\boldsymbol{r}_i \sim \mathbb{D}}[\ell(D(\hat{\boldsymbol{c}}_{-i}), \boldsymbol{r}_i)], \tag{11}$$

where D is the decoder shared with the previous module and we also adopt label smoothing technique (Szegedy et al., 2016) on $\boldsymbol{r}_i$.

2.5 Training

Finally, we optimize the sum of the above losses:

$$\mathcal{L}_{final} = \mathcal{L}_{rec} + \mathcal{L}_{asp} + \mathcal{L}_{sen} + \mathcal{L}_{reg} + \mathcal{L}_{sum}. \tag{12}$$

We also explore non-equal weighting of the losses but do not find a meaningful difference in outcomes. We perform beam search decoding in the inference stage.

	Train	**Dev**	**Test**
Yelp	13,369 / 97.1	100 / 8.0	100 / 8.0
Amazon	192,742 / 24.9	28 / 8.0	32 / 8.0
RT	2,458 / 83.3	536 / 98.0	737 / 100.3

Table 1: Statistics of three datasets in different domains (i.e., businesses, products and movies). The format in the cells is "Number of entities / Average number of reviews per entity".

3 Experiments

Datasets: We conduct experiments on three opinion summarization benchmarks in different domains, including: (1) **Yelp** (Chu and Liu, 2019) which contains business customer reviews from the Yelp Dataset Challenge[1]; (2) **Amazon** (Brazinskas et al., 2020b) which includes a large corpus of product reviews for four Amazon categories (i.e., Electronics, Clothing, Shoes and Jewelry, Home and Kitchen, and Health and Personal Care)[2]; (3) **Rotten Tomatoes (RT)** (Wang and Ling, 2016) which has a large set of reviews for various movies written by critics[3]. The detailed statistics of the three datasets are shown in Table 1. For Yelp and Amazon, there are no gold standard summaries for large training corpora, but the small development and test sets have summaries written by Amazon Mechanical Turk (AMT) crowd-workers. In RT, each set of reviews has a gold-standard opinion summary written by an editor, but we do not use ground truth summaries for training due to the unsupervised setting. Note that all reviews have a binary sentiment label (e.g., positive or negative). For Yelp and Amazon which have 1–5 scale ratings, we mark reviews with scores below 3 as negative and the rest as positive. The implementation details of our method are shown in supplementary materials.

Compared Methods: We compare TransSum with several state-of-the-art unsupervised summary generation methods, and some of them can be essentially considered as special cases of our method.

For extractive systems where summaries are created by selecting a subset of salient sentences from the input reviews, they include: (1) **LexRank** (Erkan and Radev, 2004), a PageRank-like algorithm which selects the review closest to the centroid of a group as the summary; (2) **W2VCent**,

---

[1] https://github.com/sosuperic/MeanSum

[2] https://github.com/abrazinskas/Copycat-abstractive-opinion-summarizer

[3] https://web.eecs.umich.edu/∼wangluxy/data.html

a centroid-based multi-document summarization method that uses word embeddings (Mikolov et al., 2013) instead of TF-IDF to represent each sentence (Rossiello et al., 2017); and (3) **Multi-Lead-1** (See et al., 2017) which constructs the summary by selecting the leading sentences from each review of a group. Additionally, we also report the **upper bound** of extractive methods, i.e., the highest-scoring review in a group when computing ROUGE-L (Lin and Hovy, 2003) against reference summaries.

We also compare with six state-of-the-art abstractive models where summaries are generated from scratch, including: (1) **Opinosis** (Ganesan et al., 2010), a graph-based method that uses token-level redundancy to generate summaries; (2) **MeanSum** (Chu and Liu, 2019), an auto-encoder that generates summaries by reconstructing the mean of review encodings, which is in fact special cases of our method without contrastive transformations of aspect and sentiment embeddings and high-relevance dataset creation; (3) **OpinionDigest** (Suhara et al., 2020), a combination of an aspect-based sentiment analysis model and a phrase-to-review seq2seq model, which can be seen as using opinion phrases to model summaries rather than using the aspect and sentiment embeddings as we do; (4) **DenoiseSum** (Amplayo and Lapata, 2020), which create a synthetic dataset by treating a review and its noisy versions as the summary and pseudo-review input, instead of using the aspect similarity of real-world reviews like ours; (5) **CopyCat** (Brazinskas et al., 2020b), a hierarchical variational auto-encoder which learns a latent code of the summary and uses a leave-one-out self-supervision setting, and it can be regarded as a special case where TransSum does not consider the relevance of input reviews and the constructed summaries; and (6) **PlanSum** (Amplayo et al., 2021), which uses adversarial learning to learn the aspect and sentiment distributions of reviews, instead of the intra- and inter-group contrastive transformations we use. Note that we do not compare with methods using gold summaries, such as Brazinskas et al. (2020a).

3.1 Main Results

3.1.1 Automatic Evaluation

For automatic summary evaluation, we report the classical ROUGE (Lin and Hovy, 2003) scores on test sets. We report F-measure scores of ROUGE-1 (**R1**), ROUGE-2 (**R2**) and ROUGE-L (**RL**) in the experiments.

Table 2 contains the automatic evaluation results on three different datasets. From the results, we can see that: (1) Although extractive methods (e.g., LexRank, W2VCent and Multi-Lead-1) achieve comparable results, their upper bounds are affected by the data sets used. For example, the upper bound results of R2 and RL on Yelp are much lower than the other two, perhaps because most sentences on the Yelp dataset contain more redundant information. (2) Among abstractive models, OpinionDigest and CopyCat perform much better than Opinosis and MeanSum, showing the effectiveness of using opinion phrases or specific distributions to model opinion summaries. But our method surpasses them by a wide margin, indicating that the aspect and sentiment embeddings learned by contrastive learning are beneficial for modeling opinion summaries. (3) Impressively, we observe a large improvement brought by the creation of synthetic datasets (i.e., DenoiseSum, CopyCat and PlanSum), showing the usefulness of using reviews as pseudo-summaries. However, our method is superior to them, illustrating the importance of considering the relevance of the constructed reviews-summary pairs. (4) Overall, our model outperforms all baseline models on three datasets over all three metrics. It is also worth noting that TransSum even surpasses the upper bound of extractive methods on Yelp with an increase of 5.55, 2.3, and 2.2 points in ROUGE-1/2/L.

3.1.2 Human Evaluation

Further, we conduct a human evaluation to evaluate the quality of generated summaries more accurately.

We focus on five criteria: (1) the aspect-based informativeness indicator (**Aspect**) focuses on whether the summary covers common aspects discussed in the reviews, (2) the sentiment-based informativeness indicator (**Sentiment**) focuses on whether it agrees with their overall sentiment about different aspects. (3) the relevance indicator (**Relevance**) reflects whether the summary is relevant to the input reviews, (4) the non-redundancy indicator (**Non-Redundancy**) measures whether the summary contains unnecessary repetition, and (5) the fluency indicator (**Fluency**) shows whether the summary is well-formed and grammatical. We show the detailed questions in **supplementary materials**. We sampled 50, 32, and 50 review groups

Method	*Abstr.?*	Yelp			Amazon			RT		
		R1	**R2**	**RL**	**R1**	**R2**	**RL**	**R1**	**R2**	**RL**
Upper Bound (Extractive)	✗	31.07	6.11	18.11	33.98	7.88	21.60	30.94	10.75	24.95
LexRank(Erkan and Radev, 2004)	✗	25.50	2.64	13.37	28.74	5.47	16.75	14.88	1.94	10.50
W2VCent (Rossiello et al., 2017)	✗	24.61	2.85	13.81	28.73	4.97	17.45	13.93	2.10	10.81
Multi-Lead-1 (See et al., 2017)	✗	26.34	3.72	13.86	30.32	5.90	15.78	14.21	1.82	10.23
Opinosis (Ganesan et al., 2010)	✓	25.15	2.61	13.54	28.42	4.57	15.50	14.98	3.07	12.19
MeanSum (Chu and Liu, 2019)	✓	28.86	3.66	15.91	29.20	4.70	18.15	15.79	1.94	12.26
OpinionDigest(Suhara et al., 2020)	✓	29.30	5.77	18.56	-	-	-	-	-	-
DenoiseSum(Amplayo and Lapata, 2020)	✓	30.14	4.99	17.65	-	-	-	21.26	4.61	16.27
CopyCat(Brazinskas et al., 2020b)	✓	29.47	5.26	18.09	31.97	5.81	20.16	-	-	-
PlanSum(Amplayo et al., 2021)	✓	34.79	7.01	19.74	32.87	6.12	19.05	21.77	6.18	16.98
TransSum (Ours)	✓	**36.62***	**8.41***	**20.31***	**34.23***	**7.24***	**20.49***	**25.34***	**8.62***	**18.35***

Table 2: Automatic evaluation results on three datasets. We make the best results bold, and use "-" to indicate unreported results or unfound outputs. "$*$" means that the improvements over PlanSum are statistically significant with p-value ≤ 0.05 for t-test, and "*Abstr.?*" indicates whether the method is an abstractive approach.

	Method	Asp.	Sen.	Rel.	Non.	Flu.
Yelp	LexRank	-0.49	-0.31	-0.81	-0.60	-0.30
	PlanSum	-0.12	-0.23	-0.47	-0.13	-0.10
	TransSum	**0.26**	**0.13**	**0.57**	**0.24**	**0.02**
	Gold	0.35	0.41	0.71	0.49	0.38
Amazon	LexRank	-0.62	-0.47	-0.45	-0.53	-0.58
	PlanSum	0.08	-0.11	-0.06	-0.36	0.10
	TransSum	**0.28**	**0.19**	**0.15**	**0.38**	**0.17**
	Gold	0.42	0.39	0.36	0.51	0.31
RT	LexRank	-0.55	-0.32	-0.54	-0.73	-0.36
	PlanSum	-0.08	-0.14	-0.41	-0.33	-0.13
	TransSum	**0.26**	**0.17**	**0.42**	**0.45**	**0.20**
	Gold	0.37	0.29	0.53	0.61	0.29

Table 3: Human evaluations results in terms of the Best-Worst scaling. The kappa coefficient of judges is 0.72.

#	$\mathcal{L}_{rec}$	$\mathcal{L}_{asp}$	$\mathcal{L}_{sen}$	$\mathcal{L}_{reg}$	Yelp	Ama.	RT
1	✓				18.64	18.94	16.89
2	✓	✓		✓	19.91	19.61	17.46
3	✓		✓	✓	19.82	19.53	17.26
4		✓	✓	✓	19.96	20.04	17.93
5	✓	✓	✓		20.08	20.11	18.14
6	✓	✓	✓	✓	**20.31**	**20.49**	**18.35**

Table 4: Ablation study of different losses. $\mathcal{L}_{sum}$ is a basic loss, so all combinations in the table include it.

from the Yelp, Amazon, and RT test sets with human-annotated summaries, respectively. Then we employ five graduate students to evaluate each tuple containing summaries from LexRank (strong extractive baseline), PlanSum (strong abstractive baseline), TransSum (Ours) and the gold-standard summaries according to the criteria. Note that the order in which the summaries are presented to the judges is random. We use Best-Worst Scaling (Louviere et al., 2015), which has been shown to produce more reliable results than ranking scales (Kiritchenko and Mohammad, 2016). Specifically, each score is computed as the percentage of times it was selected as best minus the percentage of times it was selected as worst, and ranges from -1 (unanimously worst) to +1 (unanimously best).

The results are shown in Table 3. As shown, summaries generated by TransSum have better aspect- and sentiment-based informativeness, indicating that our model can effectively capture the salient opinion information. We find that extractive summaries tend to be more general or even irrelevant (e.g. LexRank on Yelp), but our model performs very well in terms of relevance. Our method also excels baselines in non-redundancy and fluency, showing that summaries generated TransSum are low-redundant and fluent. We show examples of generated summaries of our model and comparison systems in supplementary materials.

3.2 Ablation Study

3.2.1 Loss Effectiveness Analysis

We present in Table 4 various ablation studies on the three datasets, which assess the contribution of different losses. We report the ROUGE-L score on test sets.

Compared to the only reconstruction loss (i.e., row#1), the contrastive learning of aspect and sentiment embeddings (i.e., row#2 and row#3) can bring improvements of 1.27/0.67/0.57 and 1.18/0.59/0.37 points on three datasets, respectively. From row#5 and row#4, we observe that the reconstruction and

#	Model Variants	Yelp	Amazon	RT
1	TransSum	20.31	20.49	18.35
2	w/o BERT embeddings	20.05	20.14	18.05
3	w/o label smoothing	20.08	20.21	18.12
4	w/o beam search	20.02	20.08	17.99
5	w/o summary modeling	18.51	18.82	16.72
6	w/o weighted pooling	19.83	19.95	17.88

Table 5: Ablation study of different components. “w/o” means “without”.

regularization losses are also useful for improving results. The last row shows that all our proposed losses in TransSum are helpful, especially $\mathcal{L}_{asp}$ and $\mathcal{L}_{sen}$, demonstrating the effectiveness of our model.

3.2.2 Module Effectiveness Analysis

To investigate the importance of the model’s individual components, we perform ablations by removing the initialized BERT embeddings, label smoothing, beam search, the translation-based review modeling module, and weighted pooling operation (i.e., $\boldsymbol{w}_{-i}$ is a uniform distribution).

From the results in Table 5, all components play a role, yet the most significant drop (i.e., row#5) in ROUGE-L when the translation-based review modeling module is removed, demonstrating the great effectiveness of the aspect and sentiment embeddings learned through contrastive learning. Interestingly, even without learning aspect and sentiment embeddings, using high-relevant reviews-summary pairs created by only entangled representations (i.e., row#5) can also achieve competitive results. This further shows the importance of considering the relevance between reviews and pseudo-summaries.

4 Related Work

Unsupervised Opinion Summarization aims to automatically generate summaries for a group of opinions about a specific entity (e.g., user reviews of a product), and does not require any gold summaries (Ku et al., 2006; Kim et al., 2011; Chu and Liu, 2019). Most previous works focus on extractive approaches, which select a subset of salient sentences from the inputs based on topic-words (Paul et al., 2010; Fabbrizio et al., 2014), word-frequencies (Erkan and Radev, 2004; Nenkova and Vanderwende, 2005), word embeddings (Rossiello et al., 2017) or textual graphs (Radev et al., 2004). However, due to their shortcomings of copying text from the input (Banko and Vanderwende, 2004), studies of abstractive summarization methods have increased tremendously (Ganesan et al., 2010; Perez-Beltrachini et al., 2019; Zou et al., 2020; Mukherjee et al., 2020). Most of these abstractive works model the problem of opinion summarization as a normal multi-document summarization task, using an auto-encoder framework with attention (Chu and Liu, 2019; Amplayo and Lapata, 2019; Brazinskas et al., 2020a), variational distributions (Brazinskas et al., 2020b; Angelidis et al., 2020), or abstract meaning representations (Liu et al., 2015). Few of them pay attention to the opinion information, and model the opinion summary with opinion phrases (Suhara et al., 2020) or the aspect and sentiment distributions (Amplayo et al., 2021). To the best of our knowledge, we are the first to model opinion summaries with only aspect and sentiment embeddings, which are learned through two novel contrastive objectives based on the aspect and sentiment invariances.

Our work is also related to **contrastive learning**, which a popular unsupervised learning paradigm in the field of computer vision and speech, aiming to enlarge the embedding disagreements of different instances for representation learning (van den Oord et al., 2018; Ye et al., 2019; He et al., 2020). Although there have been studies using contrastive learning for summary evaluation (Wu et al., 2020), to our best knowledge, we are the first to use the contrastive transformation on natural textual samples to directly help summary generation, and open the door to research on modeling opinion summaries with aspect and sentiment embeddings.

5 Conclusion

In this paper, we propose a novel self-supervised framework TransSum, to generate opinion summaries with only the aspect and sentiment embeddings, which are beneficial for maximizing informativeness, reducing redundancy of repeated opinions in reviews, and creating synthetic datasets of highly relevant reviews-summary pairs for training. Extensive evaluation and ablation studies show our model outperforms competitive systems in generating informative, high-relevant, low-redundant and fluent summaries. We believe that the viewpoint from modeling opinion summaries with only aspect and sentiment embeddings proposed in this study may pave a new way to design more complex and controllable systems for unsupervised opinion summarization.

6 Acknowledgements

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We would like to appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Reinald Kim Amplayo, Stefanos Angelidis, and Mirella Lapata. 2021. Unsupervised opinion summarization with content planning.

Reinald Kim Amplayo and Mirella Lapata. 2019. Informative and controllable opinion summarization. *CoRR*, abs/1909.02322.

Reinald Kim Amplayo and Mirella Lapata. 2020. Unsupervised opinion summarization with noising and denoising. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1934–1945. Association for Computational Linguistics.

Stefanos Angelidis, Reinald Kim Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. 2020. Extractive opinion summarization in quantized transformer spaces. *CoRR*, abs/2012.04443.

Michele Banko and Lucy Vanderwende. 2004. Using n-grams to understand the nature of summaries. In *Proceedings of HLT-NAACL 2004: Short Papers, Boston, Massachusetts, USA, May 2-7, 2004*. The Association for Computational Linguistics.

Julian Besag. 1975. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 24(3):179–195.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. 2005. A tutorial on the cross-entropy method. *Ann. Oper. Res.*, 134(1):19–67.

Arthur Brazinskas, Mirella Lapata, and Ivan Titov. 2020a. Few-shot learning for opinion summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4119–4135. Association for Computational Linguistics.

Arthur Brazinskas, Mirella Lapata, and Ivan Titov. 2020b. Unsupervised opinion summarization as copycat-review generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5151–5169. Association for Computational Linguistics.

Jackie Chi Kit Cheung, Giuseppe Carenini, and Raymond T. Ng. 2009. Optimization-based content selection for opinion summarization. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation (UCNLG+Sum 2009)*, pages 7–14, Suntec, Singapore. Association for Computational Linguistics.

Eric Chu and Peter J. Liu. 2019. Meansum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Hady ElSahar, Maximin Coavoux, Matthias Gallé, and Jos Rozen. 2020. Self-supervised and controlled multi-document opinion summarization. *CoRR*, abs/2004.14754.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.*, 22:457–479.

Giuseppe Di Fabbrizio, Amanda Stent, and Robert J. Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. In *INLG 2014 - Proceedings of the Eighth International Natural Language Generation Conference, Including Proceedings of the INLG and SIGDIAL 2014 Joint Session, 19-21 June 2014, Philadelphia, PA, USA*, pages 54–63. The Association for Computer Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 340–348. Tsinghua University Press.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. IEEE.

Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. 2011. Comprehensive review of opinion summarization.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing reliable fine-grained sentiment associations by crowdsourcing and best-worst scaling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 811–817. The Association for Computational Linguistics.

Lun-Wei Ku, Yu-Ting Liang, Hsin-Hsi Chen, et al. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 100107, pages 1–167.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman M. Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1077–1086. The Association for Computational Linguistics.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Rajdeep Mukherjee, Hari Chandana Peruri, Uppada Vishnu, Pawan Goyal, Sourangshu Bhattacharya, and Niloy Ganguly. 2020. Read what you need: Controllable aspect-based opinion summarization of tourist reviews. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1825–1828. ACM.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2019. What is this article about? extreme summarization with topic-aware convolutional neural networks. *J. Artif. Intell. Res.*, 66:243–278.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Michael J. Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 66–76. ACL.

Laura Perez-Beltrachini, Yang Liu, and Mirella Lapata. 2019. Generating summaries with topic templates and structured convolutional decoders. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5107–5116. Association for Computational Linguistics.

Maxime Peyrard. 2019. A simple theoretical model of importance for summarization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1059–1073. Association for Computational Linguistics.

Dragomir R. Radev, Hongyan Jing, Magorzata Sty, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Inf. Process. Manag.*, 40(6):919–938.

Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro. 2017. Centroid-based text summarization through compositionality of word embeddings. In *Proceedings of the Workshop on Summarization and Summary Evaluation Across Source Types and Genres, MultiLing@EACL 2017, Valencia, Spain, April 3, 2017*, pages 12–21. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Yoshihiko Suhara, Xiaolan Wang, Stefanos Angelidis, and Wang-Chiew Tan. 2020. Opiniondigest: A simple framework for opinion summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5789–5798. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 47–57. The Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Hanlu Wu, Tengfei Ma, Lingfei Wu, Tariro Manyumwa, and Shouling Ji. 2020. Unsupervised reference-free summary quality evaluation via contrastive learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3612–3621. Association for Computational Linguistics.

Mang Ye, Xu Zhang, Pong C. Yuen, and Shih-Fu Chang. 2019. Unsupervised embedding learning via invariant and spreading instance feature. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6210–6219. Computer Vision Foundation / IEEE.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1049–1058. Association for Computational Linguistics.

Yanyan Zou, Xingxing Zhang, Wei Lu, Furu Wei, and Ming Zhou. 2020. Pre-training for abstractive document summarization by reinstating source text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3646–3660. Association for Computational Linguistics.

A Implementation Details

We implement our method on top of the Transformer-base (Vaswani et al., 2017) implemented in Fairseq (Ott et al., 2019). The token embeddings of BERT-base (Devlin et al., 2019) used for initialization are provided by Transformers (Wolf et al., 2020). The dimension k is 768 and the number of the attention heads is 4. Both the encoder and decoder have 6 layers, and the maximum sequence length l is set to 200. The beam size of the beam search is set to 5. We set the dropout rate to 0.1 and Adam (Kingma and Ba, 2015) with learning rate to 1e-5, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We implement our model based on PyTorch and use four TITAN X graphic cards for learning.

B Human Evaluation Questions

- **Aspect-based informativeness**: The summary sentences should cover common aspects discussed in the group of reviews.
- **Sentiment-based informativeness**: The summary sentences should agree with the overall sentiment about different aspects in the group of reviews.
- **Relevance**: The summary sentences should relevant to the input reviews.
- **Non-Redundancy**: The summary sentences should not contains unnecessary repetition.
- **Fluency**: The summary sentences should be grammatically correct, easy to read and understand.

<table>
<tr><th colspan="2">Input Reviews</th></tr>
<tr><td colspan="2">1. i got the roast duck won ton noodles . the noodles were good and firm and the wontons were 100 % shrimp which was very good . the roast duck and roast suckling pig was bland . but not bad for the prices .</td></tr>
<tr><td colspan="2">2. lost on the shuffle . lucky i had extra time . ordered soup noodles with beef brisket and tendon . normally very fast as everything is already cooked . they had to ask me about my order 3 times after i asked them to check on it after waiting for 25 mins . when the food finally came , taste was good and portion was pretty big . beef tendon and brisket noodle ... the tendon was sooo soft and gooey .... mmm . but i had to deduct a point for the service mishap . and it wasn ’ t busy yet ...</td></tr>
<tr><td colspan="2">3. always fantastic food with great prices . i went every weekend for a month in the summer . the owners are always friendly . if you are going later in the evening or late , don ’ t order the tea (milk tea) or coffee . they boil it all day and by then its completely gross . but other wise , i ’ ve never had a bad experience here .</td></tr>
<tr><td colspan="2">4. food portions were small and nothing special . bonus its its open late .</td></tr>
<tr><td colspan="2">5. - solid chinese eats - it gives me a good feeling when a restaurant is full of people . and this one normally is . (especially those of the same ethnic background as the cuisine) - if ever i have a craving for congee or roasted pork on rice , i ’ m here . - oh and it ’ s mad cheap - which is a nice bonus . i dig healthy competition .</td></tr>
<tr><td colspan="2">6. the price point is a little higher than the places i frequent in richmond hill / markham and the selection is smaller , but if i ever craze decent , solid , authentic chinese food when i ’ m downtown , i come here !</td></tr>
<tr><td colspan="2">7. initially went to chinatown to eat beef brisket noodle soup at kings noodle but they were closed on wednesdays . walked down dundas and found this spot , decided to try it and as really surprised . the noodles were tasted good , much more generous portions compared to kings noodle and they were the same price . would recommend this place !</td></tr>
<tr><td colspan="2">8. this is an awesome place you can go for chinatown area . nice service , delicious food , and what you need more ?</td></tr>
<tr><td>Gold</td><td>service can be a little slow here . the noodles are really good . i think it ’ s a bit expensive though for what you get . there are other places that are cheaper but i don ’ t know how they taste , so i can only comment on here . it ’ s definitely worth checking out though . i had to wait a bit for my food but still pretty good experience .</td></tr>
<tr><td>LexRank</td><td>food portions were small and nothing special . bonus its its open late . walked down dundas and found this spot , decided to try it and as really surprised . the noodles were good and firm and the wontons were 100 % shrimp which was very good . the roast duck and roast suckling pig was bland . but not bad for the prices .</td></tr>
<tr><td>PlanSum</td><td>i’ve been to this place several times and i have never had a bad experience. the food is always good and the service is good. i love the fact that they are open late, so if you’re looking for a quick lunch or dinner, this is the place to go.</td></tr>
<tr><td>Ours</td><td>the noodles are good but the price is a little expensive . the staff is always helpful and friendly . an awesome chinese eats you can go in the chinatown area . come by yourself !</td></tr>
</table>

Table 6: Examples of opinion summaries generated by multiple systems on the Yelp dataset.

<table>
<tr><th colspan="2">Input Reviews</th></tr>
<tr><td colspan="2">1. the only thing i would like to see is an aux cord when i do n't want to charge my phone , but it 's not a huge deal . the sound is great , and worth the money . the remote works with your phone , and that 's precisely what i wanted .</td></tr>
<tr><td colspan="2">2. you need to buy an adaptor for ipod nano 's so it was disappointing when my son opened it up on christmas and could not use it for his ipod nano . it does not state that anywhere on the box or when i ordered it .</td></tr>
<tr><td colspan="2">3. love , love , love the ability to save multiple preset radio stations , and the sound is clear , crisp ... amazing ! it almost makes waking up a pleasure . another feature i never thought i wanted , but really appreciate , is the ability to set the brightness of the clock readout . brilliant !</td></tr>
<tr><td colspan="2">4. the sound of the radio is of real quality . i also like having the two separate alarms and the alarm is not obnoxious yet still wakes us up . my wife charges her iphone on it regularly and works out well . we like the sony so much i got one for my son and his wife for a christmas present .</td></tr>
<tr><td colspan="2">5. i was looking quite awhile to locate a decent sounding radio / ipod player which would also charge my ipod . this is perfect for our family . it 's a lot smaller than i thought , which is good . and when we update to an iphone 5 , there is a $ 5 adapter to get so we can still use this radio . perfect !</td></tr>
<tr><td colspan="2">6. as always , sony has a 'winner ' in this combined am / fm radio and docking station . great sound , looks good and wife is very pleased as she put it in her craft work area . finding the combo of am / fm was n't easy either.lots of fm only units . this is a great product .</td></tr>
<tr><td colspan="2">7. while i like the dream machine i do n't know why there 's so much static . it 's nearly impossible to get a couple of my favorite radio stations without constant static in the background . my other radio does n't do that . i 've even tried different locations for it . that 's a big disappointment and shortcoming of the product .</td></tr>
<tr><td colspan="2">8. my husband really like this speaker ... love it ! its so easy to operate by setting the alarm. i like the way when you put your iphone 4s to the dock its charge at the same time while you are you using it ... ! great product</td></tr>
<tr><td>Gold</td><td>this fm/am radio , iphone docking station and alarm clock is a perfect combination ! the sound is amazing , the alarm clock is not annoying , and the design looks great ! it would be nice to have a place to use an aux cord and certain apple products require a $ 5 adapter to use the docking station but other than that , this product is fantastic !</td></tr>
<tr><td>LexRank</td><td>while i like the dream machine i do n't know why there 's so much static . great sound , looks good and wife is very pleased as she put it in her craft work area . the sound of the radio is of real quality .</td></tr>
<tr><td>PlanSum</td><td>this fm/am radio , iphone docking station and alarm clock is a perfect combination ! the sound is amazing , the alarm clock is not annoying , and the design looks great ! it would be nice to have a place to use an aux cord and certain apple products require a $ 5 adapter to use the docking station but other than that , this product is fantastic ! this is a great product . it has amazing sound quality , and the dual-band feature is nice . i love the fact that it charges my device while it 's docked . not only is this thing functional , but it also looks great and does n't take up a lot of space . overall , i would advise buying this if your needs call for an awesome speaker that doubles as a charging station . this is exactly what i 've been looking for ! it has great sound quality and it 's really easy to dock my iphone in it (it charges my phone at the same time ! bonus) the alarm is easy to use and does it 's job . some iphones and ipods will need an adapter , so make sure to check that out first .</td></tr>
<tr><td>Ours</td><td>i love this radio for its sound, which is amazingly clear and quite great . overall , this is a great product , and good for my family. i would advise buying this .</td></tr>
</table>

Table 7: Examples of opinion summaries generated by multiple systems on the Amazon dataset.

Input Reviews
1. The best reason to see The Haunting is the sheer sumptuousness of its creepy-crawly set designs.
2. Has an unseen enchantment, so aptly sets spinning like a huge magnificent gyroscope on a string
3. In The Haunting, the moviemakers succeed in something very difficult: creating a haunted house with real personality and terror.
4. All the stops are pulled out to provide a state-of-the-art, slam-bang movie experience.
5. Looking terrified and screaming is really all that's required in David Self's inane script.
6. Director Jan de Bont, known for the razzle he put into the exciting movie Speed and the subsequent dud Twister, proves himself unable to break away from depending on dazzle to substitute for substance.
7. To my surprise, I find myself recommending The Haunting on the basis of its locations, its sets, its art direction, its sound design, and the overall splendor of its visuals.
8. It's all hokum from beginning to end.
9. A flat, draggy exercise in cheesy special effects and grandiose art direction palming itself off as a horror film.
10. More hokey than haunting.
11. When The Haunting finally limps to its conclusion, you may feel like booing the screen.
12. This all-flash, no-substance–and no scare–thriller is a textbook example of soulless, money-burning Hollywood hype products.
13. One of the most misguided big-screen diversions to come along in some time, considering the clear potential it has.
14. so thoroughly misguided and muddled Dreamworks, the studio foisting this bomb on the public, ought to hire a special corps of ushers to hand out sympathy cards to patrons as they exit the theater.
15. This is as far from the Poverty Row gasps of The Blair Witch Project as you can get, and more fun.
16. I wouldn't waste more than the price of a video rental on this one.
17. High-tech remake is dumb and overblown.
18. All logic is deadened by the obnoxious special effects!
19. Once the screaming begins, so will your laughing
20. Glossy but lackluster.
21. It's just a conglomeration of cheap fright tactics and a booming bass track meant to get you to jump out of your seat.
22. An exercise in missed opportunities and bad filmmaking!
23. The Haunting is a muddled mess that defies any rationality.
24. The only thing scary about the new version is realizing that someone keeps giving director Jan De Bont money to make movies.
25. The characters are on the dramatic equivalent of Death Row.
......

Gold	sophisticated visual effects fail to offset awkward performances and an uneven script .
LexRank	the characters are on the dramatic equivalent of death row .
PlanSum	the haunting is a very good movie, but it's a lot of fun, and the filmmakers have been raised by the original.
Ours	unfortunately , this is one haunting with the obnoxious special effects that are bloated and wretchedly overdone !

Table 8: Examples of opinion summaries generated by multiple systems on the Rotten Tomatoes dataset.

Hashing based Efficient Inference for Image-Text Matching

Rong-Cheng Tu[†], **Lei Ji** [‡§¶*], **Huaishao Luo**[‖], **Botian Shi**[≀],
Heyan Huang[†], **Nan Duan**[¶] and **Xian-Ling Mao**[†]
[†]School of Computer Science and Technology,
Beijing Institute of Technology, Beijing, China, Beijing, China
[‡]Institute of Computing Technology, CAS, Beijing, China
[§]University of Chinese Academy of Sciences, Beijing, China
[¶]Microsoft Research Asia, Beijing, China
[‖]Southwest Jiaotong University, Chengdu, China
[≀]Shanghai AI lab, Shanghai, China
[†]{turongcheng, hhy63, maoxl}@bit.edu.cn, [¶]{leiji,nanduan}@microsoft.com

Abstract

Image-text matching has been a popular research topic which bridges vision and language through semantic understanding. Recent works mainly focus on exploring the interactions between images and sentences to improve the performance without considering inference efficiency. Specifically, for the large scale databases, it is unacceptable to perform such time-consuming mechanisms between a query (text/image) and each candidate datapoint (image/text) in the whole retrieval set during inference. To tackle this problem, we propose a novel hashing based efficient inference module called HEI, which can be plugged into the existing framework to speed up the inference step without reducing the retrieval performance. In details, HEI learns to map the original datapoints into short binary hash codes and coarsely preserve the heterologous matching relationship. Thus, in the inference phase, the proposed HEI module uses the hash codes to quickly select a few candidate datapoints from the retrieval set for a given query. Then, the image-text matching model fine ranks the candidate set to find the matching datapoint. Extensive experiments on two widely used benchmark MS-COCO and Flickr30k with four baseline methods demonstrate the efficiency and effectiveness of our proposed HEI module.

1 Introduction

Language and vision understanding plays a fundamental role for human to understand the real world. A large amount of works has been proposed to bridge these two modalities. Image-text matching is one of the fundamental topics in this field, which benefits a series of downstream applications, such as visual captioning(Zhang et al., 2019; Wang et al., 2018) and visual grounding (Chen et al., 2018; Plummer et al., 2017). Specifically, given an image (text), its target is to match the closest textual description (image) for the image (text).

Early works (Karpathy and Fei-Fei, 2015; Wang et al., 2016; Niu et al., 2017; Faghri et al., 2017) achieve this goal by learning two modality-specific deep neural networks to directly map all the datapoints from the two modality into a common joint space without using attention mechanism, and then measures their similarities by feature distances in the joint space. Compared with these methods, recent works (Lee et al., 2018; Liu et al., 2019; Wang et al., 2019; Chen et al., 2020) mainly focus on incorporating variant attention mechanisms into the image-text matching models to explore the fine-grained interactions between vision and language. By using the attention mechanisms, the image-text matching models are able to filter out irrelevant information, and find the fine-grained cues to achieve a great matching performance. For example, CAMP (Wang et al., 2019) takes comprehensive and fine-grained cross-modal interactions into account, and also properly handles negative pairs and irrelevant information with an adaptive gating scheme to improve the matching performance.

Although existing attention mechanism based methods achieve great performance, they do not take the inference efficiency into account. Specifically, for the large scale databases, due to the attention mechanisms being time-consuming, it is unacceptable to perform such complex attention mechanisms between the query (text/image) and each candidate datapoint (image/text) in the whole retrieval set during inference. Thus, it is critical to improve the inference speed of these methods.

Intuitively, if a small candidate set containing positive datapoints can be quickly selected out, the image-text matching models can greatly speed up through only fine ranking such a small candidate set instead of the whole retrieval set. Then the key

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 743–752
August 1–6, 2021.

challenge is how to quickly select such a small candidate set. Hashing is widely used in the field of data search with fast retrieval speed. Besides, although it can hardly perform the accurate matching, hashing is capable of quickly selecting a high quality candidate set containing the positive datapoints.

Hence, in this paper, we propose a novel hashing based efficient inference module, called HEI, which can be plugged into the existing attention mechanism based image-text matching framework to speed up the inference step without reducing the retrieval performance. Specifically, a matching score based hashing loss is proposed, which consists of two items: one is used to make Hamming similarity between hash codes of matching datapair be as large as possible; the other item is to make the Hamming similarity between hash codes of mismatching datapair no larger than their corresponding matching score. By minimizing the proposed hashing loss, the HEI module is optimized to map the original datapoints into short binary hash codes and coarsely preserve the heterologous matching relationship between datapoints. Thus, the trained HEI module can be used to speed up the inference step without reducing the retrieval performance. Extensive experiments on two widely used benchmark MS-COCO and Flickr30k with four baselines demonstrate the effectiveness of our proposed HEI module.

2 Related Work

2.1 Text-image Matching

Recently, many image-text matching methods have been proposed, which can be roughly grouped into one-to-one matching methods learning correspondence between the whole image and text, and many-to-many matching methods learning correspondence between the regions of image and the words of text.

The one-to-one matching methods (Wang et al., 2016; Kiros et al., 2014; Zhang and Lu, 2018; Zheng et al., 2020) mainly aim to explore the heterologous relationship globally by mapping the whole images and the full texts into a common feature space. However, owing to these methods doing not explore the correspondence between image regions and text words, it might lead to learn sub-optimal features, which will damage the text-image matching performance.

By utilizing variant cross-modal attention mechanisms, many-to-many matching methods can explore the correspondence between image regions and text words, thus, these attention mechanism based methods can achieve the state-of-the-art performance. For instance, BFAN (Liu et al., 2019) is proposed to eliminate partial irrelevant words and regions from the shared semantic in image-text pairs to achieves state-of-the-art performance on several benchmark datasets. IMRAM (Chen et al., 2020) proposes a recurrent attention memory which incorporates a cross-modal attention unit and a memory distillation unit to refine the correspondence between image regions and text words. However, those attention mechanisms used by the many-to-many matching methods are usually complicated with high computation complexity. Hence, it is unacceptable to perform such time-consuming attention mechanisms between the query (text/image) and each candidate datapoint (corresponding to image/text) in the whole retrieval set during inference especially for the large scale databases.

Different from previous methods, our model explores hashing technology to improve the inference speed of the existing many-to-many text-image matching methods without reducing their performance.

2.2 Cross-Modal Hashing

The core of cross-modal hashing is to project the datapoints of different modalities into compact binary hash codes, meanwhile, preserve the semantic similarity of original datapoints. Then, in the cross-modal retrieval phase, the datapoints of the retrieval set can be sorted by the Hamming distance between their corresponding binary hash codes calculated by the ‘XOR’ operation, which has fast retrieval speed. Due to this advantage, a mount of cross-modal hashing methods have been proposed (Hu et al., 2020; Su et al., 2019; Lin et al., 2020; Tu et al., 2020; Shi et al., 2019). For example, SDCH (Lin et al., 2020) utilizes a semantic label branches to preserve semantic information of the learned features by integrating with inter-modal pairwise loss, cross-entropy loss and quantization loss.

However, due to these hashing methods belonging to approximate nearest neighbour (ANN) searching technology, they can hardly to accurately find the matching datapoint for a query. Hence, few works explore the hashing technology for text-image matching. To our best knowledge, this is the first work to explore the use of hashing to improve the inference speed of existing attention mecha-

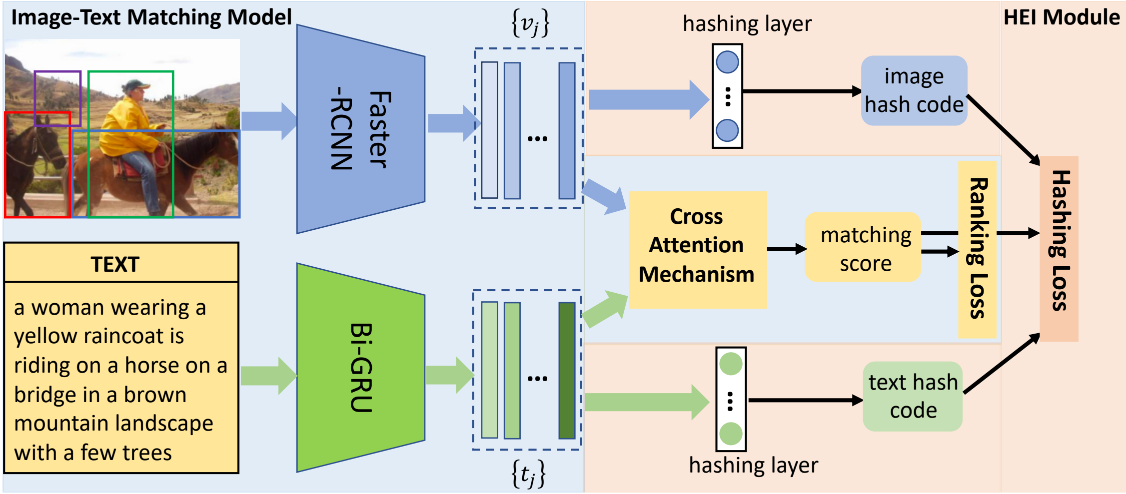

Figure 1: The architecture of image-text method with the HEI module.

nism based image-text matching methods.

3 Methodology

As shown in Figure 1, different from the architecture of existing matching models, our framework introduces an extra hashing based efficient inference module, called HEI, which consists of an image modal hashing layer and a text modal hashing layer, and each hashing layer is a fully-connected layer with k units where k is the hash codes length.

3.1 Problem formulation and notations

Without loss of generality, suppose there are datasets with M images $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{M}$ and N texts $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{N}$. Given an image $\boldsymbol{x}_i$ with its region-level visual features denoted as $\boldsymbol{V}_i = [\boldsymbol{v}_1^i, \cdots, \boldsymbol{v}_m^i]$, and a text $\boldsymbol{y}_j$ with its word-level textual features denoted as $\boldsymbol{T}_j = [\boldsymbol{t}_1^j, \cdots, \boldsymbol{t}_n^j]$, the goal of image-text matching is to calculate a matching score $\boldsymbol{s}_{ij}$ for the image $\boldsymbol{x}_i$ and the text $\boldsymbol{y}_j$ based on their features $\boldsymbol{V}_i$ and $\boldsymbol{T}_j$. Moreover, if the image $\boldsymbol{x}_i$ and the text $\boldsymbol{y}_j$ are matching, the matching score $\boldsymbol{s}_{ij}$ should be large, otherwise $\boldsymbol{s}_{ij}$ should be small.

Furthermore, the goal of hashing based efficient inference module is to learn the two modality-specific hashing layer which can map their corresponding modal datapoints into binary hash codes with the heterologous matching relationship preserved.

3.2 Cross-modal Feature Representation

3.2.1 Image region-level visual features

To obtain the region-level visual features $\boldsymbol{V}_i = [\boldsymbol{v}_1^i, \cdots, \boldsymbol{v}_m^i]$ of the image $\boldsymbol{x}_i$, we first employ the Faster R-CNN (Ren et al., 2016) model using ResNet-101 (He et al., 2016) as the backbone, which is pre-trained on the Visual Genomes dataset (Krishna et al., 2017) by (Anderson et al., 2018), to extract the top m region proposals of the image. Then, by average-pooling the spatial feature map, a feature vector $\boldsymbol{v}_j^{i\prime} \in \mathcal{R}^{2048}$ for the j^{th} region proposal is calculated. Finally, We obtain the d-dimentional region features with a linear projection layer:

$$\boldsymbol{v}_j^i = \boldsymbol{W}_v \boldsymbol{v}_j^{i\prime} + \boldsymbol{b}_v \tag{1}$$

where $\boldsymbol{W}_v$ and $\boldsymbol{b}_v$ are to-be-learned parameters, and $\boldsymbol{v}_j^i$ is the visual feature for the j^{th} region of image $\boldsymbol{x}_i$.

3.2.2 Text word-level textual features

To obtain the textual features of a input text $\boldsymbol{y}_j$ with n words, we first embed each word $\boldsymbol{w}_i$ of the input text $\boldsymbol{y}_j$ into a 300-dimensional vector $\boldsymbol{t}_i^{j\prime}$. Then, to enhance the word-level feature with sufficient context information, we use a single-layer bi-directional GRU (Cho et al., 2014) with d-dimensional hidden to summarize information from both forward and backward directions of the input text $\boldsymbol{y}_j$:

$$\begin{aligned} \overrightarrow{\boldsymbol{h}_i^j} &= \overrightarrow{GRU}(\overrightarrow{\boldsymbol{h}_{i-1}^j}, \boldsymbol{t}_i^{j\prime}), \\ \overleftarrow{\boldsymbol{h}_i^j} &= \overleftarrow{GRU}(\overleftarrow{\boldsymbol{h}_{i-1}^j}, \boldsymbol{t}_i^{j\prime}). \end{aligned} \tag{2}$$

where $\overrightarrow{\boldsymbol{h}_i^j}$ and $\overleftarrow{\boldsymbol{h}_i^j}$ denote hidden states from the forward GRU and the backward GRU, respectively. Then, the textual feature of the word $\boldsymbol{w}_i^j$ in the text $\boldsymbol{y}_j$ is defined as:

$$\boldsymbol{t}_i^j = \frac{\overrightarrow{\boldsymbol{h}_i^j} + \overleftarrow{\boldsymbol{h}_i^j}}{2} \tag{3}$$

3.3 General Attention Framework

Existing attention mechanism based image-text matching methods mainly learn to associate shared semantics between the region-level feature $\boldsymbol{V}_i$ of image $\boldsymbol{x}_i$ and word-level feature $\boldsymbol{T}_j$ of text $\boldsymbol{y}_j$ through variant cross-attention mechanisms to calculate the matching score $\boldsymbol{s}_{ij}$, which can be formulated as follows:

$$\boldsymbol{s}_{ij} = CAM(\boldsymbol{V}_i, \boldsymbol{T}_j; \boldsymbol{W}) \tag{4}$$

where $CAM(\cdot; \boldsymbol{W})$ denotes the cross-modal attention mechanism and $\boldsymbol{W}$ is the set of learnable parameters. For example, in BFAN (Liu et al., 2019), $CAM(\cdot; \boldsymbol{W})$ denotes the Focal attention mechanism proposed in the original paper.

Then to maximize matching scores of the matching image-text pairs and minimize the ones of the mismatching datapairs, a hinge-based triplet ranking loss with emphasis on the hard negatives are used as the loss function. Specifically, given a pair of matching image-text $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, we denote their matching score as $\boldsymbol{s}_{ij}$; $\bar{j} = argmax_{t \neq j} \boldsymbol{s}_{it}$ denotes the hard negative when using the image to match text; $\bar{i} = argmax_{t \neq i} \boldsymbol{s}_{tj}$ denotes the hard negative when using the text to match image, then the ranking loss is formulated as:

$$\mathcal{L}_{rank} = [\alpha - \boldsymbol{s}_{ij} + \boldsymbol{s}_{i\bar{j}}]_+ + [\alpha - \boldsymbol{s}_{ij} + \boldsymbol{s}_{\bar{i}j}]_+ \tag{5}$$

where α is the margin for the ranking loss, and $[a]_+ = max(0, a)$.

Finally, after optimizing the matching model, given a query datapoint, it will be used to calculated the matching score with each datapoint in the retrieval set to find the most matching one by the cross-attention mechanism. However, the cross-attention mechanism is time-consuming which means it unacceptable to calculate a matching score between the query and each point in retrieval set with the attention mechanism during inference. Thus, we propose a hashing based efficient inference module to improve the inference speed.

3.4 Hashing based Efficient Inference module

Specifically, the input of the HEI module is the fragment-level feature of datapoint, i.e., the region-level feature $\boldsymbol{V}_i$ for an image modal input $\boldsymbol{x}_i$ or the word-level feature $\boldsymbol{T}_j$ for a text modal input $\boldsymbol{y}_j$. We further aggregate the fragment-level feature $\boldsymbol{V}_i$ ($\boldsymbol{T}_i$) into an instance-level feature $\hat{\boldsymbol{v}}_i$ ($\hat{\boldsymbol{t}}_i$) for an image (text) datapoint $\boldsymbol{x}_i$ ($\boldsymbol{y}_i$):

$$\hat{\boldsymbol{v}}_i = \sum_{j=1}^{m} a_j \boldsymbol{v}_j^i; \quad a_j = \frac{\boldsymbol{v}_j^{iT} \boldsymbol{w}_v}{\sum_{k=1}^{m} \boldsymbol{v}_k^{iT} \boldsymbol{w}_v}. \tag{6}$$

$$\hat{\boldsymbol{t}}_i = \sum_{j=1}^{n} q_j \boldsymbol{t}_j^i; \quad q_j = \frac{\boldsymbol{t}_j^{iT} \boldsymbol{w}_t}{\sum_{k=1}^{n} \boldsymbol{t}_k^{iT} \boldsymbol{w}_t}. \tag{7}$$

where $\boldsymbol{w}_t$ and $\boldsymbol{w}_v$ denote learnable parameter. Then by forwarding the instance-level feature $\hat{\boldsymbol{v}}_i$ ($\hat{\boldsymbol{t}}_i$) into the image (text) modal hashing layer, the hash codes $\boldsymbol{b}_i^v$ ($\boldsymbol{b}_i^t$) of image $\boldsymbol{x}_i$ (text $\boldsymbol{y}_i$) can be generate as:

$$\begin{aligned} \boldsymbol{b}_i^v &= sgn(\mathcal{H}_x(\hat{\boldsymbol{v}}_i; \boldsymbol{\Theta}_v)) \in \{-1, 1\}^k \\ \boldsymbol{b}_i^t &= sgn(\mathcal{H}_y(\hat{\boldsymbol{t}}_i; \boldsymbol{\Theta}_t)) \in \{-1, 1\}^k \end{aligned} \tag{8}$$

where $\mathcal{H}_x(\hat{\boldsymbol{v}}_i; \boldsymbol{\Theta}_v)$ denotes the image modal hashing layer and $\boldsymbol{\Theta}_v$ denotes the set of parameters in the image hashing layer; k is the length of hash codes; $\mathcal{H}_y(\hat{\boldsymbol{t}}_i; \boldsymbol{\Theta}_t)$ represents the text modal hashing layer and $\boldsymbol{\Theta}_t$ represents the set of parameters in the text hashing layer; $sgn(\cdot)$ is an element-wise sign function, which returns 1 if the element is positive and returns -1 otherwise.

Furthermore, the core of hashing based efficient inference module is to learn two modality-specific hashing layer to map the datapoints into binary hash codes which are used to select a few candidate datapoints from the retrieval set for an query. To achieve this goal, the learned hash codes should coarsely preserve the heterologous matching relationship between datapoints, i.e., if two datapoints are matching, then the Hamming distance between their corresponding binary hash codes should be small, otherwise it should be large.

Considering that the Hamming distance between $\boldsymbol{b}_i^v$ and $\boldsymbol{b}_j^t$ can be difined as: $D_H(\boldsymbol{b}_i^v, \boldsymbol{b}_j^t) = 0.5(k - \boldsymbol{b}_i^{vT} \boldsymbol{b}_j^t)$, where k denotes the code length. It means when $\frac{1}{k} \boldsymbol{b}_i^{vT} \boldsymbol{b}_j^t$ is close to 1, the Hamming distance $D_H(\boldsymbol{b}_i^v, \boldsymbol{b}_j^t)$ is close to 0; and when $\frac{1}{k} \boldsymbol{b}_i^{vT} \boldsymbol{b}_j^t$ is close to -1, the Hamming distance $D_H(\boldsymbol{b}_i^v, \boldsymbol{b}_j^t)$

is close to k. Thus, $\frac{1}{k}\boldsymbol{b}_i^{vT}\boldsymbol{b}_j^t$ can be used to denote the Hamming similarity between $\boldsymbol{b}_i^v$ and $\boldsymbol{b}_j^t$, and measure the heterologous matching relationship preserved by $\boldsymbol{b}_i^v$ and $\boldsymbol{b}_j^t$. Furthermore, as the mathching score $\boldsymbol{s}_{ij} \in [0, 1]$ of image $\boldsymbol{x}_i$ and text $\boldsymbol{y}_j$ computed by the cross-attention mechanism may preserve the heterologous matching relationship to a certain extent, then we can use it as soft target to supervise the learning of similarity between hash codes of mismatching data pairs. Owing to $\frac{1}{k}\boldsymbol{b}_i^{vT}\boldsymbol{b}_j^t \in [-1, 1]$, we re-scale $\boldsymbol{s}_{ij}$ as $\hat{\boldsymbol{s}}_{ij} = 2\boldsymbol{s}_{ij} - 1 \in [-1, 1]$.

Thus, based on these observations, we proposed a matching score based hashing loss:

$$\begin{aligned}\mathcal{L}_1 = &\frac{1}{|N_i^+|}\sum_{j\in N_i^+}(\frac{1}{k}\boldsymbol{b}_i^{vT}\boldsymbol{b}_j^t - 1)^2 \\ &+ \frac{1}{\sum\limits_{j\in N_i^-}\boldsymbol{I}_{ij}}\sum_{j\in N_i^-}\boldsymbol{I}_{ij}(\frac{1}{k}\boldsymbol{b}_i^{vT}\boldsymbol{b}_j^t - \hat{\boldsymbol{s}}_{ij})^2\end{aligned} \tag{9}$$

$$\boldsymbol{I}_{ij} = \begin{cases} 1, & \frac{1}{k}\boldsymbol{b}_i^{vT}\boldsymbol{b}_j^t > \hat{\boldsymbol{s}}_{ij}, \\ 0, & otherwise. \end{cases} \tag{10}$$

where N_i^+ denotes the set of text datapoints which are matching with the image $\boldsymbol{x}_i$, and N_i^- denotes the set of text datapoints which are not matching with the image $\boldsymbol{x}_i$; $\boldsymbol{s}_{ij}$ denotes the matching score between the image $\boldsymbol{x}_i$ and text $\boldsymbol{y}_j$;

It can be found that the first item of $\mathcal{L}_1$ is to make $\frac{1}{k}\boldsymbol{b}_i^{vT}\boldsymbol{b}_j^t$ be close to 1, i.e., make the Hamming distance between the hash codes of matching datapair be small. The second item of $\mathcal{L}_1$ is constructed to penalize the mismatching datapair that the Hamming similarity between their hash codes is larger than their matching score s_{ij}, i.e., the goal of $\mathcal{L}_1$ is to make the Hamming distance between their hash codes be large. Thus, by minimizing the hashing loss $\mathcal{L}_1$, the learned binary hash codes can coarsely preserve the heterologous matching relationship.

Furthermore, as the $sgn(\cdot)$ function is indifferentiable at zero and the derivation of it will be zeros for a non-zero input, the parameters of hashing model will not be updated with the backpropagation algorithm when minimizing the hashing loss function $\mathcal{L}_1$. Thus, we directly discard the $sgn(\cdot)$ function to ensure the parameters of our hashing model can be updated, and use $\tanh(\cdot)$ to approximate the $sgn(\cdot)$ function to make each element of output of hashing layer can be close to "+1" or "-1". Then the final hashing loss function can be formulated as follows:

$$\begin{aligned}\mathcal{L}_h = &\frac{1}{|N_i^+|}\sum_{j\in N_i^+}(\frac{1}{k}\hat{\boldsymbol{b}}_i^{vT}\hat{\boldsymbol{b}}_j^t - 1)^2 \\ &+ \frac{1}{\sum\limits_{j\in N_i^-}\boldsymbol{I}_{ij}}\sum_{j\in N_i^-}\boldsymbol{I}_{ij}(\frac{1}{k}\hat{\boldsymbol{b}}_i^{vT}\hat{\boldsymbol{b}}_j^t - \hat{\boldsymbol{s}}_{ij})^2.\end{aligned} \tag{11}$$

where $\hat{\boldsymbol{b}}_i^v = tanh(\mathcal{H}_x(\hat{\boldsymbol{v}}_i; \boldsymbol{\Theta}_v))$ and $\hat{\boldsymbol{b}}_j^t = tanh(\mathcal{H}_y(\hat{\boldsymbol{t}}_j; \boldsymbol{\Theta}_t))$

3.5 Inference

After training the image-text matching model and HEI module well, we can generate the hash codes $\{\boldsymbol{b_i^v}\}_{i=1}^N$ ($\{\boldsymbol{b_i^t}\}_{i=1}^M$) for all the images $\{\boldsymbol{x_i}\}_{i=1}^M$ (text $\{\boldsymbol{y_i}\}_{i=1}^N$) in the retrieval set using the HEI module. When given a query image $\boldsymbol{x}_q$ (text $\boldsymbol{y}_q$), we also use HEI module map it into a hash code $\boldsymbol{b_q^v}$ ($\boldsymbol{b_q^t}$), and calculate the Hamming distances between $\boldsymbol{b_q^v}$ ($\boldsymbol{b_q^t}$) and each code in $\{\boldsymbol{b_i^t}\}_{i=1}^M$ ($\{\boldsymbol{b_i^v}\}_{i=1}^N$). Then, we sort the texts (images) in the retrieval set in ascending order according to the Hamming distances, and select a few of the top texts (images) as the candidate set. Finally, we only need to do the fine-grained matching in the candidate set to find the matching datapoints.

4 Experiments

4.1 Datasets

We evaluate the performance of the proposed HEI module on two public used datasets: **Flickr30K** (Plummer et al., 2015) and **MS-COCO** (Lin et al., 2014). Specifically, Flickr30k contains 31783 images collected from the Flickr website. Each image is accompanied with five human annotated sentences descriptions. Following the setting of previous works (Wang et al., 2019; Liu et al., 2020), this dataset is split into 29,000 images, 1,000 images, and 1,000 images for training set, validation set, and testing set respectively. We report the performance evaluation of image-text retrieval on 1000 testing set. MS-COCO is another large-scale image-caption benchmark which consists of about 123,287 images with each image also roughly annotated with five sentence descriptions. Following the widely used split (Karpathy et al., 2014; Chen et al., 2020), we use 113,287 images for training, 1000 images for validation and 5000 images for testing. Moreover, we evaluate our method on both the 5 folds of 1K test images and the full 5K test images for MS-COCO.

Method	Text Retrieval				Image Retrieval			
	R@1	R@5	R@10	Time(s)	R@1	R@2	R@10	Time(s)
BFAN (Liu et al., 2019)	0.692	0.914	0.962	76.72	0.500	0.772	0.848	31.29
BFAN-random	0.582	0.814	0.883	37.00	0.291	0.419	0.451	17.00
BFAN-HEI	0.692	0.912	0.962	22.31	0.499	0.772	0.846	11.38
CAMP (Wang et al., 2019)	0.675	0.914	0.954	568.57	0.527	0.787	0.850	514.52
CAMP-random	0.599	0.823	0.873	295.61	0.318	0.436	0.461	280.06
CAMP-HEI	0.676	0.909	0.950	168.15	0.526	0.782	0.844	166.21
IMRAN (Chen et al., 2020)	0.710	0.920	0.963	1858.82	0.531	0.799	0.862	692.16
IMRAN-radom	0.594	0.842	0.892	956.44	0.314	o.432	0.465	353.75
IMRAN-HEI	0.710	0.920	0.964	574.93	0.532	0.797	0.858	219.72
GSMN (Liu et al., 2020)	0.733	0.918	0.964	518.32	0.524	0.792	0.863	146.70
GSMN-random	0.611	0.839	0.890	149.65	0.302	0.426	0.458	72.37
GSMN-HEI	0.734	0.919	0.967	99.30	0.524	0.790	0.860	51.02

Table 1: Comparison in terms of R@N scores and time cost of two retrieval tasks on Flickr30K

Method	Text Retrieval				Image Retrieval			
	R@1	R@5	R@10	Time(s)	R@1	R@2	R@10	Time(s)
BFAN (Liu et al., 2019)	0.753	0.961	0.989	62.84	0.610	0.896	0.954	32.45
BFAN-random	0.657	0.899	0.930	29.87	0.361	0.486	0.502	16.15
BFAN-HEI	0.753	0.962	0.989	18.21	0.610	0.896	0.953	12.24
CAMP (Wang et al., 2019)	0.711	0.953	0.977	567.86	0.581	0.882	0.948	515.18
CAMP-random	0.648	0.890	0.935	302.29	0.340	0.469	0.486	284.65
CAMP-HEI	0.713	0.953	0.977	169.24	0.580	0.882	0.946	168.39
IMRAN (Chen et al., 2020)	0.784	0.964	0.991	1808.01	0.644	0.912	0.960	662.59
IMRAN-random	0.688	0.904	0.935	868.64	0.370	0.484	0.496	332.95
IMRAN-HEI	0.784	0.964	0.991	504.64	0.644	0.912	0.960	215.15
GSMN (Liu et al., 2020)	0.758	0.960	0.992	465.75	0.607	0.897	0.955	205.89
GSMN-random	0.658	0.903	0.946	120.56	0.302	0.426	0.458	99.23
GSMN-HEI	0.758	0.960	0.992	80.38	0.607	0.894	0.953	70.69

Table 2: Comparison in terms of R@N scores and time cost of two retrieval tasks on MS-COCO 1K

4.2 Evaluation

Following the setting in (Chen et al., 2020; Liu et al., 2020), we evaluate the performance of our proposed approach by reporting Recall@K (K = 1, 5, 10) values for bi-directional matching tasks, i.e. matching texts given an image query (Text Retrieval) and matching images given a text query (Image Retrieval). The Recall computes the proportion of correct image or text being retrieved among top K results. In addition, we also record the inference time in seconds to evaluate the efficiency of our proposed HEI.

4.3 Baselines

To evaluate the performance of our proposed HEI, some state-of-the-art attention mechanism based image-text matching methods are selected as our baselines, including BFAN (Liu et al., 2019), CAMP (Wang et al., 2019), IMRAN (Chen et al., 2020) and GSMN (Liu et al., 2020). It should be noted that the proposed HEI focuses on exploring a novel and efficient hashing based efficient inference module that can be universally plugged into existing attention mechanism based image-text methods to speed up the inference speed rather than redesigning a new cross-modal attention mechanism to improve their matching performance.

4.4 Implementation Details

All our experiments are implemented in PyTorch and conducted on a NVIDIA Tesla V100 GPU. For representing visual modality, the amount of regions in each image is $m = 36$, and the dimensionality

Method	Text Retrieval				Image Retrieval			
	R@1	R@5	R@10	Time(s)	R@1	R@2	R@10	Time(s)
BFAN	0.499	0.795	0.888	9461.96	0.369	0.657	0.772	3797.68
BFAN-random	0.426	0.705	0.807	5521.66	0.231	0.381	0.428	2903.52
BFAN-HEI	0.496	0.791	0.884	3355.99	0.369	0.657	0.772	2177.12
CAMP	0.433	0.751	0.863	18646.21	0.324	0.633	0.753	16344.43
CAMP-random	0.361	0.661	0.763	11515.34	0.211	0.373	0.423	9615.76
CAMP-HEI	0.434	0.750	0.862	8198.13	0.323	0.632	0.751	6277.92
IMRAN	0.525	0.812	0.898	39030.95	0.391	0.684	0.795	16877.68
IMRAN-random	0.447	0.721	0.814	19423.71	0.244	0.390	0.435	8505.07
IMRAN-HEI	0.525	0.813	0.898	8265.47	0.390	0.683	0.794	3315.97
GSMN	0.494	0.793	0.888	25261.93	0.359	0.655	0.769	12226.86
GSMN-random	0.414	0.696	0.795	6453.45	0.229	0.383	0.435	7461.81
GSMN-HEI	0.493	0.793	0.888	3561.13	0.359	0.654	0.767	4607.29

Table 3: Comparison in terms of R@N scores and time cost of two retrieval tasks on MS-COCO 5K

of the final region representation vectors is set as 1024. Moreover, the dimensionality of hidden state (i.e., $\overrightarrow{\boldsymbol{h}_i^j}$ and $\overleftarrow{\boldsymbol{h}_i^j}$ in Formula (2)) in the GRU is also set as 1024. The length of hash codes is set as 64. In the training phase, we first train the base cross-modal attention module for 20 epochs, then train the HEI module jointly. We adopt SGD with a mini-batch size of 128 and a learning rate within 10^{-2} to 10^{-3} to optimize the HEI modul. The optimization algorithm for the base cross-modal attention module is the same with the ones defined in the original method, for example, when plugging HEI module into GSMN, the optimization algorithm for cross-modal attention module is Adam.

4.5 Main results

We conduct extensive experiments on Flickr30K and MS-COCO. The image-text matching results on Flickr30K, MS-COCO dataset with 1K test points and 5K test points are shown in Table 1, 2 and 3, respectively. ”method”-HEI denotes the method using the proposed HEI module, for example, BFAN-HEI means plugging HEI into BFAN to speed up the inference speed. Similarly, ”method”-random denotes randomly selecting 50% datapoints from retrieval set as the candidate set to speed up the inference speed of the method.

Based on the results shown in these tables, the following observations can be got: (1) our proposed HEI module can greatly improve the matching efficiency of all the four baselines almost without reducing the matching performance, and even slightly improve the performance of some baselines. For example, as shown in Table 1, comparing GSMN-HEI with GSMN, GSMN-HEI achieves an increase of 0.1% on the R@1 metric in the text retrieval task, and greatly reduces the inference time from 518.32 seconds to 99.30 seconds. The reason why plugging the HEI module can slightly improve the performance maybe that for some query, there are some false positive datapoints which can misguide the image-text model, but the Hamming distance between the hash codes of queries and the ones of false positive datapoints are large, i.e., the false positive datapoints will not be selected as the candidate points. Thus, without the effect of the false positive datapoints, the image-text model can find the matching points successfully and improve the retrieval performance. (2) The proposed HEI module can map datapoints into hash codes with the original heterologous matching relationship coarsely preserved. For instance, as shown in Table 1, 2 and 3, all the methods with the proposed HEI module achieve not only better performance than the methods with the randomly selected candidate, but also lower inference time. It means that the number of datapoints in the candidate set selected by our proposed HEI module is smaller but the possibility of the candidate set containing the matching datapoint is higher.

4.6 Discussion

4.6.1 Ablation study

To further investigate the impact of the length of hash codes, we construct there variants of HEI module with code length being 16, 32, and 128 bits with two baselines on Flickr30K, respectively. The re-

Method	Text Retrieval				Image Retrieval			
	R@1	R@5	R@10	Time(s)	R@1	R@2	R@10	Time(s)
BFAN-HEI$_{16bits}$	0.687	0.903	0.958	20.85	0.485	0.742	0.810	11.02
BFAN-HEI$_{32bits}$	0.685	0.907	0.957	21.03	0.498	0.765	0.838	11.17
BFAN-HEI$_{128bits}$	0.690	0.913	0.961	21.46	0.500	0.771	0.846	11.55
BFAN-HEI	0.692	0.912	0.962	21.31	0.499	0.772	0.846	11.38
GSMN-HEI$_{16bits}$	0.719	0.911	0.956	99.13	0.506	0.756	0.821	50.84
GSMN-HEI$_{32bits}$	0.731	0.916	0.966	99.22	0.519	0.779	0.847	50.92
GSMN-HEI$_{128bits}$	0.734	0.919	0.965	99.51	0.520	0.788	0.855	51.23
GSMN-HEI	0.734	0.919	0.967	99.31	0.524	0.790	0.860	51.02

Table 4: Comparison in terms of R@N scores and time cost of two retrieval tasks on Flickr30K

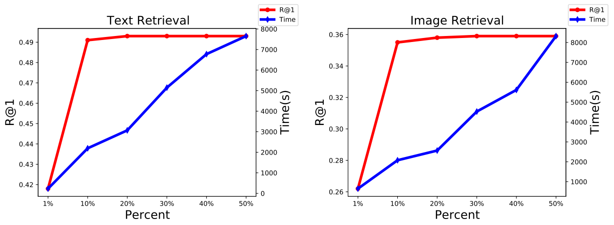

(a) (b)

Figure 2: The Figure 2(a) and (b) denotes the results of BFAN-HEI on MS-COCO(5k) in text retrieval task and image retrieval task, respectively. Moreover, in each figure, the axis X denotes selecting how much percentage of points in the retrieval set as candidate set, and the axis Y for the red line is in the left which is the value of R@1, and the axis Y for the blue line is in the right which denotes the inference time taken for the transaction in seconds.

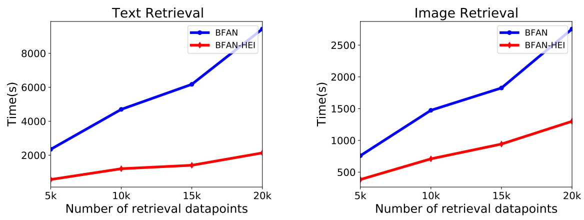

(a) (b)

Figure 3: Compare the inference time of BFAN and the one of BFAN-HEI in large retrieval set under the condition that BFAN-HEI achieves the same performance of BDAN

sults are shown in Table 4. From these results, it can be found: (1) The length of hash codes rarely influence the inference time that each baseline with a different hash code length of HEI consumes nearly the same inference time. This is because that the speed of the "XOR" operation between hash codes is far faster than the ones of the cross-modal attention mechanism. Thus, it implicitly demonstrates the availability of speeding up the inference speed of baselines by using the proposed HEI to fast select the candidate set. (2) The matching performance first increases as the hash code length varies from 16 to 64, and then tend to be stable when the length varies from 64 to 128. Thus, for the other experiments, the hash code length is set as 64.

4.6.2 Efficiency and performance

We also conduct experiments to further investigate the trade-off between inference efficiency and matching performance.

As the results shown in Figure 2, with the size of candidate increasing, the matching performance of BFAN-HEI (the red lines) increase rapidly and then tend to stable, and BFAN-HEI consumes more inference time (the blue lines). It can be found that when selecting only 20% of datapoints in the retrieval set as the candidate set by the proposed HEI module, BFAN-HEI can already achieve the best performance, and greatly reduce the inference time. Thus, it demonstrates the effectiveness of our proposed HEI module.

4.6.3 Scalability for the large retrieval set

To further investigate the scalability of the proposed HEI module for the large retrieval set, when doing experiments on the MS-COCO (1K) with the BFAN baseline, we directly use training data to expand the data volume of the retrieval set. The curves of inference time w.r.t. the volume of retrieval set are shown in Figure 3. It can be found that with volume of the retrieval set increasing, our proposed HEI module can still be used to speed up the inference speed without reducing the matching performance, which demonstrates the scalability of our proposed HEI module for large retrieval sets.

5 Conclusion

In this paper, we have proposed a novel Hashing based Efficient module, called HEI, which can be plugged into the existing image-text matching methods to speed up the inference step without reducing the matching performance. Extensive experiments on two widely used benchmark MS-COCO and Flickr30k with four baseline methods demonstrate the efficiency and effectiveness of our proposed HEI module.

Acknowledgments

The work is supported by National Key RD Plan (No.2018YFB1005100), National Natural Science Foundation of China (No. 61751201, 61602197 and 61772076), Natural Science Fund of Beijing (No. Z181100008918002) and the funds of Beijing Advanced Innovation Center for Language Resources (No. TYZ19005).

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.

Hui Chen, Guiguang Ding, Xudong Liu, Zijia Lin, Ji Liu, and Jungong Han. 2020. Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12655–12663.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 162–171.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hengtong Hu, Lingxi Xie, Richang Hong, and Qi Tian. 2020. Creating something from nothing: Unsupervised knowledge distillation for cross-modal hashing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3123–3132.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. *Advances in neural information processing systems*, 27:1889–1897.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 201–216.

Qiubin Lin, Wenming Cao, Zhihai He, and Zhiquan He. 2020. Semantic deep cross-modal hashing. *Neurocomputing*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chunxiao Liu, Zhendong Mao, An-An Liu, Tianzhu Zhang, Bin Wang, and Yongdong Zhang. 2019. Focus your attention: A bidirectional focal attention network for image-text matching. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 3–11.

Chunxiao Liu, Zhendong Mao, Tianzhu Zhang, Hongtao Xie, Bin Wang, and Yongdong Zhang. 2020. Graph structured network for image-text matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10921–10930.

Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. 2017. Hierarchical multimodal lstm for dense visual-semantic embedding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1881–1889.

Bryan A Plummer, Arun Mallya, Christopher M Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Phrase localization and visual relationship detection with comprehensive image-language cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1928–1937.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149.

Yufeng Shi, Xinge You, Feng Zheng, Shuo Wang, and Qinmu Peng. 2019. Equally-guided discriminative hashing for cross-modal retrieval. In *IJCAI*, pages 4767–4773.

Shupeng Su, Zhisheng Zhong, and Chao Zhang. 2019. Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3027–3035.

Rong-Cheng Tu, Xian-Ling Mao, Bing Ma, Yong Hu, Tan Yan, Wei Wei, and Heyan Huang. 2020. Deep cross-modal hashing with hashing functions and unified hash codes jointly learning. *IEEE Transactions on Knowledge and Data Engineering*.

Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. 2018. Bidirectional attentive fusion with context gating for dense video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7190–7198.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5005–5013.

Zihao Wang, Xihui Liu, Hongsheng Li, Lu Sheng, Junjie Yan, Xiaogang Wang, and Jing Shao. 2019. Camp: Cross-modal adaptive message passing for text-image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5773.

Wei Zhang, Bairui Wang, Lin Ma, and Wei Liu. 2019. Reconstruct and represent video contents for captioning via reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ying Zhang and Huchuan Lu. 2018. Deep cross-modal projection learning for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 686–701.

Zhedong Zheng, Liang Zheng, Michael Garrett, Yi Yang, Mingliang Xu, and Yi-Dong Shen. 2020. Dual-path convolutional image-text embeddings with instance loss. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(2):1–23.

Can the Transformer Learn Nested Recursion with Symbol Masking?

Jean-Philippe Bernardy **Adam Ek** **Vladislav Maraev**
Centre for Linguistic Theory and Studies in Probability
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg
`firstname.lastname@gu.se`

Abstract

We investigate if, given a simple symbol masking strategy, self-attention models are capable of learning nested structures and generalise over their depth. We do so in the simplest setting possible, namely languages consisting of nested parentheses of several kinds. We use encoder-only models, which we train to predict randomly masked symbols, in a BERT-like fashion. We find that the accuracy is well above random baseline, with accuracy consistently above 50% both when increasing nesting depth and distances between training and testing. However, we find that the predictions made correspond to a simple parenthesis counting strategy, rather than a push-down automaton. This suggests that self-attention models are not suitable for tasks which require generalisation to more complex instances of recursive structures than those found in the training set.

1 Introduction

Self-attention models (Vaswani et al., 2017) enjoy broad use in NLP tasks. The best attention-based models can tackle several tasks using a unified sentence encoding (and perhaps decoding) module (Raffel et al., 2020), with applications ranging from classification to inference and generation. They provide state of the art results for all such tasks, displacing the already very successful recurrent neural networks, in particular the LSTM and its variants. The availability of large pretrained models (Devlin et al., 2019) is another strong point in their favour.

However, the generalisation capabilities of self-attention models are still not well understood, and the present work is part of an ongoing effort to understand their capabilities. We study in particular their ability to learn context-free languages, which are characterised by the nested structures. For this purpose, we control the inputs to the model to the maximum, while focusing on the defining characteristic of context-free languages, namely matching opening and closing brackets. This corresponds to learning *generalised Dyck languages* (see table 2). In particular, we investigate the following questions:

1. Can self-attention generalise to matching open/close parenthesis at longer distances?
2. Can self-attention generalise to matching open/close parenthesis at deeper nesting levels distances?

There is a already a small body of work dealing with this question (see sec. 5), but our contribution is specific in the following two respects: i) We use the popular BERT-like training regime (predict a percentage of randomly masked tokens), ii) We concentrate on generalising to (much) deeper nesting.

Beyond theoretical considerations, matching brackets have applications in the NLP-style treatment of constructed languages (in particular) programming languages, for example translating between programs and their natural language descriptions.

2 Data Sets

We define the language $\mathcal{D}_n$ as the set of strings generated by the following context-free rules: $E ::= \varepsilon; E ::= EE; E ::= oEc$, where (o, c) stands for a pair of matching parenthesis pairs. The index n stands for the number of possible pairs. In all of our tests, we will use $n = 5$ (corresponding for example to the pairs `()` , `[]`, `{}`, <> and «»), and thus we drop the subscript from now on.

We are interested in various characteristics of the strings of $\mathcal{D}$. First, we consider the distance between a closing parenthesis and the cor-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 753–760
August 1–6, 2021.

responding opening parenthesis. Given a string s of length $2N$ (N is the number of matching pairs), we will call $\delta(s)$ an array of length $2N$ such that if s_i is a closing parenthesis, $\delta(s)_i$ is the distance between s_i and the closing parenthesis. If s_i is an opening parenthesis, $\delta(s)_i$ is 0. For example, if $s =$“`{()<[](«»)>}`”, $\delta(s) = [0, 0, 1, 0, 0, 1, 0, 0, 1, 3, 9, 11]$. The second characteristic that we consider is the amount of nesting between closing and opening parentheses. We call this characteristic η(s), and likewise we define it for each closing parenthesis, and let it be zero for opening parentheses.

For example, if $s =$“`{()<[](«»)>}`”, $\eta(s) = [0, 0, 1, 0, 0, 1, 0, 0, 1, 2, 3, 4]$.

To generate a string with N matching pairs, we perform a random walk between opposite corners of a square grid of width and height N, such that one is not allowed to cross the diagonal. When not restricted by the boundary, a step can be taken either along the x or y axis with equal probability. A step along the x axis corresponds to open a parenthesis, and one along the y axis corresponds to closing one. The kind of parenthesis pair is chosen randomly and uniformly. We call the distribution of input strings sampled by this procedure D. In all our experiments we set $N = 10$ (which is enough to illlustrate our points) and we thus omit the superscript in what follows.

We also want control the maximum distance between opening and closing parentheses (so that we never train on too long distances). We do so by discarding elements s of D such that $\delta(s)_i > d$ for some i, and call the resulting distribution $D[MaxDist = d]$.

Often we want to control the maximum depth that our model is trained or tested on. For this purpose, we generate strings s which exhibit at least one index i such that $\eta(s)_i = d$, but no index j such that $\eta(s)_j > d$. These paths can be generated by constraining the path on the grid to touch a diagonal at distance d to the origin diagonal, and we call the corresponding distribution $D[MaxDepth = d]$.

3 Model and masking strategy

We implement a variation of the transformer model as introduced by (Vaswani et al., 2017). In the model each input symbol is associated with a vector embedding of size K. A sequence of opening and closing brackets is represented by a matrix of size (N, K).

Following Devlin et al. (2019), our model then applies a series of multi-head self-attention layers organised in a hierarchical structure, such that the second layer operates on the representations generated in the first layer, and so on. We use a BERT-like, non auto-regressive architecture: each layer attends to every position in the input, including itself. Then a softmax classifier is employed to predict the symbol at the current position. Hence, we use a masking strategy to train and test the model (otherwise it could simply use the current symbol for prediction).

For training, we follow the masking strategy presented by Devlin et al. (2019). We mask 15% of the closing parenthesis tokens at random, where in 80% of the cases we replace the token with a mask token, in 10% of the cases with a random token, and in the remaining 10% of the cases we replace it with the same token.

For testing, after sampling a string s, we pick a random position i such that s_i is a closing parenthesis. Then we mask all subsequent symbols, and let the model predict s_i. There is a single possible closing parenthesis type for s_i, corresponding to the opening parenthesis found earlier in the string. The prediction is considered successful if the model predicts the right type of closing parenthesis.

4 Experiments & Results

Our experiments consists in training the language model for a limited version of the Dyck family (for example by limiting nesting depth (η) or maximum distance (δ)), and testing what the performance is in a more general case. Thus, because there are five types of parenthesis pairs in all our experiments, the random baseline is $\frac{1}{5} = 20\%$.

4.1 Generalisation to Longer Distances

In the first experiment we investigate whether the model is capable of predicting closing parenthesis at long distance from the corresponding opening parentheses, whereas it has only seen short-distances in the training data. More precisely, we train the model on strings from $D[MaxDist = 9]$ and test it on $D[MaxDist = 19]$.

We present an overview of the results in table 1. Our experiments show that the (2 layers, 8 heads) model generalises the best. Using fewer heads appears to be more detrimental to the model's accuracy than the number of layers. This is true even though the (8,2) model has many more parameters

Table 1: Mean accuracy and standard deviation over 10 runs on generalisation to longer distances for each model configuration.

Layers	Heads	Accuracy
4	4	0.814(± 0.013)
8	2	0.643(± 0.005)
2	8	0.844(± 0.008)

than the (2,8) model (see appendix).

The aggregated numbers however hide much of the reality of the generalisation capabilities as a function of distance. Therefore we further break down the accuracy by distance to the corresponding opening parenthesis in figure 1. The (8,2) model fails to learn parenthesis matching at short distances, but its accuracy is better for longer distances. In contrast the (4,4) and (2,8) models do well for adjacent parentheses, but their accuracy drops quickly until reaching a minimum at distance 13, dipping below 50% accuracy —however still above chance. Perhaps surprisingly, all models do very well at very long distances. These very long distances correspond to matching parentheses at the beginning of the input with parentheses at the end (that is, when we mask the fewest number of input symbols).

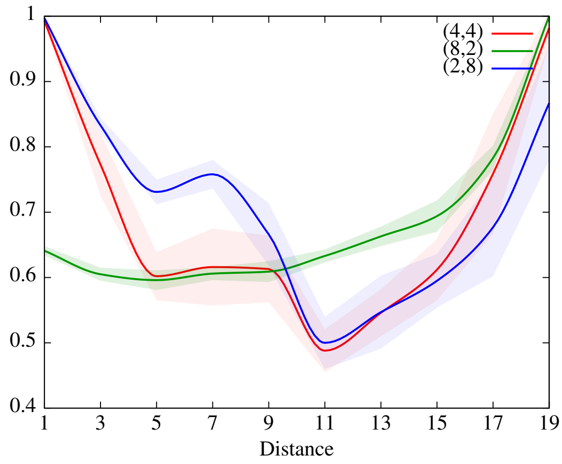

Figure 1: Mean model accuracy for closing parenthesis depending on a distance to corresponding opening parenthesis, over 10 runs. Shaded areas correspond to standard deviation.

4.2 Generalisation to Deeper Nesting

In the second experiment we test whether the model can generalise to deeper nesting depths. That is, we train the model on $D[MaxDepth = 3]$ and test it on $D[MaxDepth = 9]$

Table 2: Mean accuracy and standard deviation over 10 runs on generalisation to deeper nesting for each model configuration.

Layers	Heads	Accuracy
4	4	0.654(± 0.012)
8	2	0.518(± 0.005)
2	8	0.672(± 0.008)

We present an overview of the results in table 2. Looking at the results we see a similar pattern in terms of aggregated accuracy as in the previous experiment: the (2,8) setup performs the best, followed by (4,4) and finally (8,2). Breaking down accuracy by nesting depth (figure 2) reveals that the difference resides chiefly in the (8,2) model failing to predict shallow nesting.

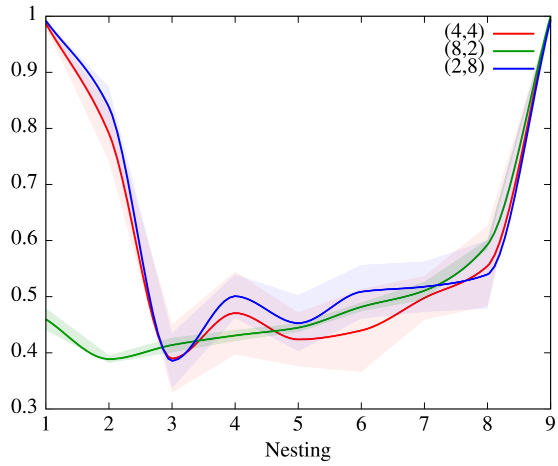

Figure 2: Mean model accuracy for closing parenthesis depending on a distance to corresponding opening parenthesis over 10 runs. Shaded areas correspond to standard deviation.

4.3 Analysis of attention heads

We have analysed attention heads by manual inspection of softmax score for attention heads for each layer, on several sequence from our training set (see Appendix for the corresponding heat maps).

Looking at the behaviour of the attention heads we note that the first layer in the (2,8) and (4,4) models focuses its attention on the previous symbol. Then, in the final layer of the (2,8) model the attention of the start of the sequence focuses on the end, and vice-versa.

In the (4,4) model, the second layer appears to often focus on the non-masked symbols while in the third layer the attention is distributed more evenly between masked and non-masked symbols. A notable feature of the third layer is that a lot of self-attention occurs on the masked symbols. In the final layer, the attention of all symbols is put almost exclusively on the masked symbols.

The (8,2) model is the only model which does not have a clear layer that looks at the preceding token. It appears that in the (8,2) model, the earlier layers focus their attention on the beginning of the sequence, then it moves towards the latter part of the sequence. The heat maps also show that the (8,2) model focuses heavily on certain symbols, which are the least frequent symbols used in the sequence, for later layers. In earlier layers the model appears to focus on the frequent symbols. This analysis is compatible with the (8,2) model using a symbol counting method.

In summary, the (4,4) model appears to first look at the previous symbol in the sequence. There are two steps of searching where first the model ignores the masked symbols and distributes the attention over the other symbols. In the second step, the model again focuses all around the sequence, but the masked symbols receive a lot of attention. For the (2,8) model, the behaviour is more straightforward. First it looks at the previous symbol, then all around the sequence. To the best of our knowledge, the (8,2) model is counting symbols by distributing its attention on frequent and less frequent symbols.

5 Related work

Studying the ability of language models to learn Dyck languages is emerging as a standard way to test the ability to generalise to deeper nesting levels. Before self-attention, this test was applied to RNNs. Bernardy (2018) proposed non-standard stack-based RNN models, which can approach perfect accuracy for generalised Dyck-language, although the accuracy of standard RNNs was higher than random but far from perfect. Hewitt et al. (2020) presented a theoretical proof that RNNs are able to learn Dyck languages with maximum nesting depth m using $O(m)$ memory. Sennhauser and Berwick (2018) present contrasting evidence, concluding that LSTMs can learn very limited range of rules.

A number of studies have considered self-attention models, especially in the past year. Ebrahimi et al. (2020) investigated self-attention models using Dyck languages, and claimed that self-attention models with a starting symbol are able to generalise to longer sequences and deeper structures without learning recursion, as competitive LSTM models do. In contrast to us, they studied models trained autoregressively only. Bhattamishra et al. (2020) studies how autoregressive Transformer architecture learns a subset of formal languages, including Dyck language and its generalisations. In contrast to our study, they examine Shuffle-Dyck languages, which allows constructions like "`( [ ) ]`" and provide theoretical and experimental evidence that the Transformer is capable of learning such a language. On the other hand, Hahn (2020) points at the limitation of using self-attention models. He indicates that in theory the LSTM should perform better than the autoregressive Transformer, because the transformer cannot emulate a stack, general finite-state automata, or use recursion.

6 Conclusion and future work

Our experiments show that, with a random masking strategy, the transformer is able to discover a way to make good predictions when generalising to longer distances and deeper nesting. However, this strategy is not using the history of opening and closing parentheses in a way a push-down automaton would.

Indeed, the analysis reveals that the best accuracy is obtained when few symbols have been masked. This can be explained by the model having learned a counting strategy. When a single symbol is masked, predicting the kind of missing parenthesis can be done by subtracting the number of closing parentheses by the number of opening parentheses for each type, and predict the type which exhibits a discrepancy. For short distances our (2,8) and (4,4) models were able to learn to remember preceding symbols and act accordingly. We suspect that for intermediate levels of nesting and distance, the models act according to a mixture of the above two strategies.

In consequence, we recommend not to use a BERT-like masking strategy for applications where generalising to longer distances or deeper nesting is critical. Rather, auto-regressive models should be used, such as auto-regressive attention or RNNs.

Acknowledgments

The research reported in this paper was supported by grant 2014-39 from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability (CLASP) in the Depart-

ment of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg.

References

Bernardy, J.-P. (2018). Can recurrent neural networks learn nested recursion? In *Linguistic Issues in Language Technology, Volume 16, 2018*. CSLI Publications.

Bhattamishra, S., Ahuja, K., and Goyal, N. (2020). On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ebrahimi, J., Gelda, D., and Zhang, W. (2020). How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4301–4306, Online. Association for Computational Linguistics.

Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Hewitt, J., Hahn, M., Ganguli, S., Liang, P., and Manning, C. D. (2020). RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, Online. Association for Computational Linguistics.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Sennhauser, L. and Berwick, R. (2018). Evaluating the ability of LSTMs to learn context-free grammars. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 115–124, Brussels, Belgium. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

A Experimental setup and reproducibility information

Our implementation is based on pytorch, with a custom re-implementation of the transformer architecture, exactly following (Vaswani et al., 2017). The runtime is under one day for the whole set of experiments using a Titan X (Pascal) GPU.

The hyperparameters we use are listed in table 3.

Table 3: Hyperparameters used and the number of data examples used.

Parameter	Value
Optimiser	Adam
Learning rate	0.0001
Epochs	10
Batch size	512
Training examples	102400
Validation examples	20480

In our experiments we consider three different transformer architectures, corresponding to different values for the number of multi-head self-attention layers, and the size of the heads. Specifically, we considers the setups presented in section 4

Table 4: Model configurations and the number of parameters in each configuration

Layers	Heads	Parameters
8	2	897 292
4	4	1 191 820
2	8	1 781 452

In each case, we have used 64-dimensional embeddings throughout the models.

B Attention heat-maps

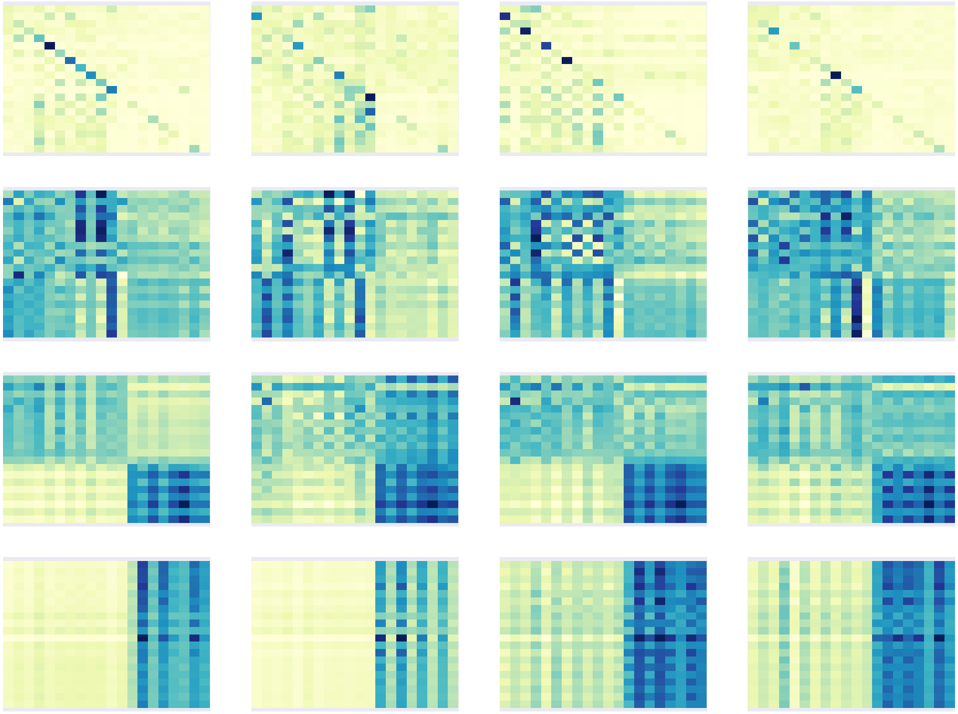

Figure 3: Attention heatsmaps for the model with 4 heads and 4 layers on the input +-+<[+[([()])]-]>-.

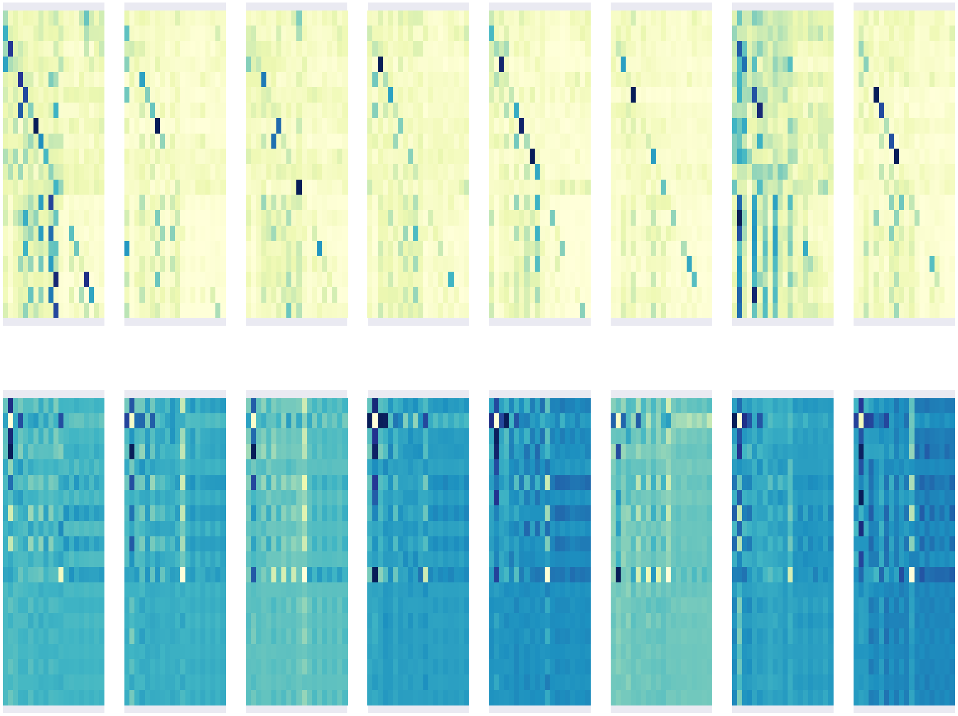

Figure 4: Attention heatsmaps for the model with 2 heads and 8 layers on the input +-+<[+[([()])]-]>-.

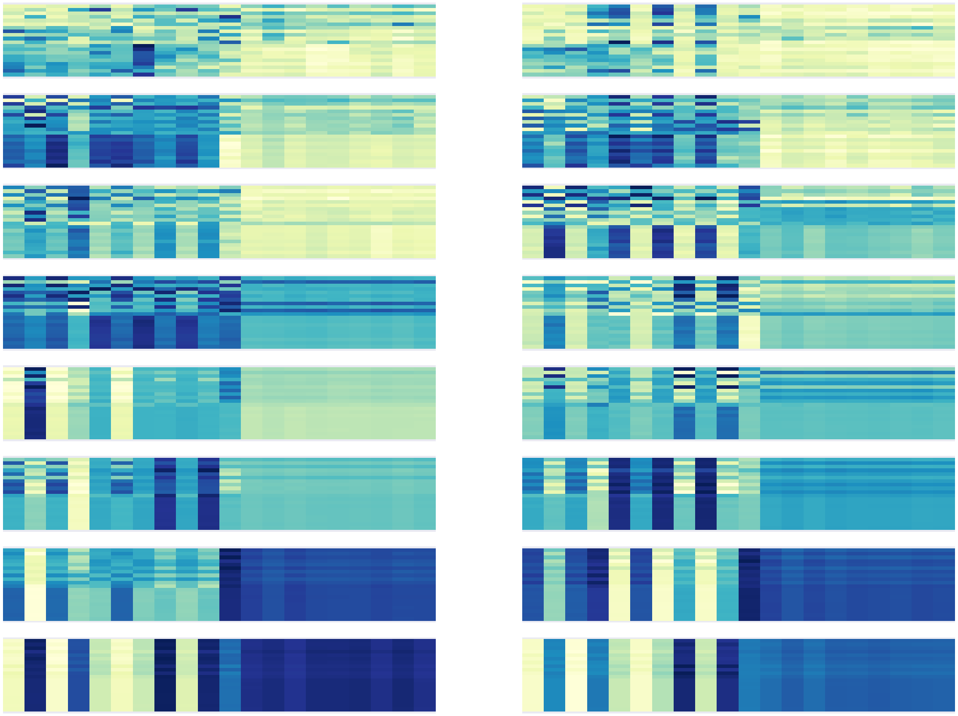

Figure 5: Attention heatsmaps for the model with 8 heads and 2 layers on the input +-+<[+[([()])]-]>-.

Rationalization through Concepts

Diego Antognini and **Boi Faltings**
École Polytechnique Fédérale de Lausanne, Switzerland
firstname.lastname@epfl.ch

Abstract

Automated predictions require explanations to be interpretable by humans. One type of explanation is a rationale, i.e., a selection of input features such as relevant text snippets from which the model computes the outcome. However, a single overall selection does not provide a complete explanation, e.g., weighing several aspects for decisions. To this end, we present a novel self-interpretable model called ConRAT. Inspired by how human explanations for high-level decisions are often based on key concepts, ConRAT extracts a set of text snippets as concepts and infers which ones are described in the document. Then, it explains the outcome with a linear aggregation of concepts. Two regularizers drive ConRAT to build interpretable concepts. In addition, we propose two techniques to boost the rationale and predictive performance further. Experiments on both single- and multi-aspect sentiment classification tasks show that ConRAT is the first to generate concepts that align with human rationalization while using only the overall label. Further, it outperforms state-of-the-art methods trained on each aspect label independently.

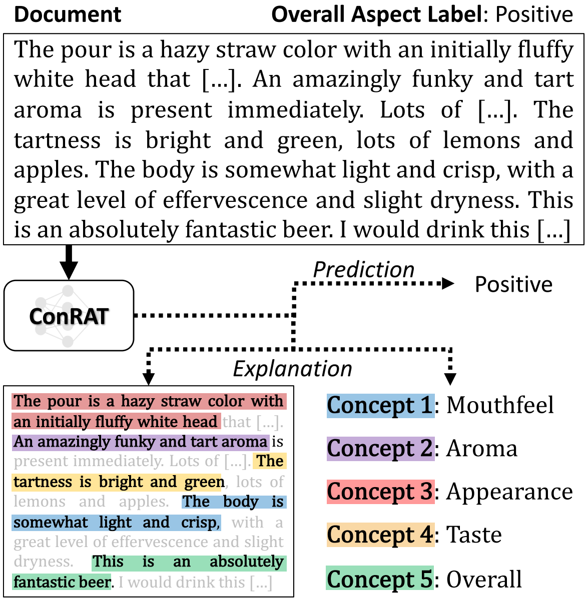

The pour is a hazy straw color with an initially fluffy white head that [...]. An amazingly funky and tart aroma is present immediately. Lots of [...]. The tartness is bright and green, lots of lemons and apples. The body is somewhat light and crisp, with a great level of effervescence and slight dryness. This is an absolutely fantastic beer. I would drink this [...]

The pour is a hazy straw color with an initially fluffy white head that [...]. An amazingly funky and tart aroma is present immediately. Lots of [...]. The tartness is bright and green, lots of lemons and apples. The body is somewhat light and crisp, with a great level of effervescence and slight dryness. This is an absolutely fantastic beer. I would drink this [...]

Figure 1: An illustration of ConRAT. Given a beer review, ConRAT identifies five excerpts that relate to particular concepts of beers (i.e., the explanation), depicted in color, from which it computes the outcome.

1 Introduction

Neural models have become the standard for many tasks, owing to their large performance gains. However, their adoption in decision-critical fields is more limited because of their lack of interpretability, particularly with textual data.

One of the simplest means of explaining predictions of complex models is by selecting relevant input features. Attention mechanisms (Bahdanau et al., 2015) model the selection using a conditional importance distribution over the inputs, but the resulting explanations are noisy (Jain and Wallace, 2019; Pruthi et al., 2020). Multi-head attention (Vaswani et al., 2017) extends attention mechanisms to attend information from different perspectives jointly. However, no explicit mechanisms guarantee a logical connection between different views (Voita et al., 2019; Kovaleva et al., 2019). Another line of research includes rationale generation methods (Lei et al., 2016; Chang et al., 2020; Antognini et al., 2021b). If the selected text input features are short and concise – called a rationale – and suffice on their own to yield the prediction, it can potentially be understood and verified against domain knowledge (Chang et al., 2019).

The key motivation for this work arises from the limitations of rationales. Rationalization models strive for one overall selection to explain the outcome by maximizing the mutual information between the rationale and the label. However, useful rationales can be multi-faceted, where each facet relates to a particular “concept” (see Figure 1). For example, users typically justify their opinions of a product by weighing explanations: one for each aspect they care about (Musat and Faltings, 2015).

Inspired by how human reasoning comprises concept-based thinking (Armstrong et al., 1983; Tenenbaum, 1999), we aim to discover, in an unsupervised manner, a set of concepts to explain the outcome with a weighted average, similar to multi-head attention. In this work, we relate concepts to semantically meaningful and consistent excerpts across multiple texts. Unlike topic modeling, where documents are described by a set of latent topics comprising word distributions, our latent concepts relate to text snippets that are relevant for the prediction.

Another motivation for this study is to generate interpretable concepts. The explanation of an outcome should rely on concepts that satisfy the desiderata introduced in Alvarez-Melis and Jaakkola (2018). They should 1. preserve relevant information, 2. not overlap with each other and be diverse, and 3. be human-understandable. Figure 1 shows an example of concepts in the beer domain.

In this work, we present a novel self-explaining neural model: the concept-based rationalizer (ConRAT) (see Figure 1 and 2). Our new rationalization scheme first identifies a set of concepts in a document and then decides which ones are currently described (binary selection). ConRAT explains the prediction with a linear aggregation of concepts. The model is trained end-to-end, and the concepts are learned in an unsupervised manner. In addition, we design two regularizers that guide ConRAT to induce interpretable concepts and propose two optional techniques, knowledge distillation and concept pruning, in order to boost the performance further.

We evaluate ConRAT on both single- and multi-aspect sentiment classification with up to five target labels. Upon training ConRAT only on the overall aspect, the results show that ConRAT generates concepts that are relevant, diverse, and non-overlapping, and they also recover human-defined concepts. Furthermore, our model significantly outperforms strong supervised baseline models in terms of predictive and explanation performance.

2 Related Work

Developing interpretable models is of considerable interest to the broader research community. Researchers have investigated many approaches to improve the interpretability of neural networks.

2.1 Interpretability.

The first line of research aims at providing post-hoc explanations of an already trained model. For example, gradient and perturbation-based methods attribute the decision to important input features (Ribeiro et al., 2016; Sundararajan et al., 2017; Lundberg and Lee, 2017; Shrikumar et al., 2017). Other studies identified the causal relationships between input-output pairs (Alvarez-Melis and Jaakkola, 2017; Goyal et al., 2019). In contrast, our model is inherently interpretable as it directly produces the prediction with an explanation.

Another line of research has developed interpretable models. Quint et al. (2018) extended a variational auto-encoder with a differentiable decision tree. Alaniz and Akata (2019) proposed an explainable observer-classifier framework whose predictions can be exposed as a binary tree. However, these methods have been designed for images only, while our work focuses on text input.

The works most relevant to ours relate to interpretable models from the rationalization field (Lei et al., 2016; Bastings et al., 2019; Yu et al., 2019; Chang et al., 2020; Jain et al., 2020; Paranjape et al., 2020). These methods justify their predictions by selecting rationales (i.e., relevant tokens in the input text). However, they are limited to explain only the prediction with mostly one text span and rely on the assumption that the data have low internal correlations (Antognini et al., 2021b). Chang et al. (2019) extended previous methods to extract an additional rationale in order to counter the prediction. In our work, ConRAT produces multi-faceted rationales and explains the prediction through a linear aggregation of the extracted concepts. However, if we set the number of concepts to one, ConRAT reduces to a special case of a rationale model.

2.2 Explanations through Concepts.

Researchers have proposed multiple approaches for concept-based explanations. Kim et al. (2018) designed a post-hoc technique to learn concept activation vectors by relying on human annotations that characterize concepts of interest. Similarly, Bau et al. (2017); Zhou et al. (2018) generated visual explanations for a classifier. Our concepts are learned in an unsupervised manner and not defined a priori.

Few studies have learned concepts on images in an unsupervised fashion. Li et al. (2018) explained predictions based on the similarity of the input to "prototypes" learned during training. Alvarez-Melis and Jaakkola (2018) used an auto-encoder to extract relevant concepts and explain the prediction. Ghorbani et al. (2019) designed an unsupervised concept discovery method to explain trained models. Koh et al. (2020) employed the discovered concepts to predict the target label. Our work's key difference is that we focus on text data, while all these methods treat only image inputs.

To the best of our knowledge, Bouchacourt and Denoyer (2019) is the only study that has proposed a self-interpretable concept-based model for text data using reinforcement learning. It computes the predictions and provides an explanation in terms of the presence or absence of concepts in the input (i.e., text excerpts of variable lengths). However, their method achieves poor overall performance. In addition, it is unclear whether the discovered concepts are interpretable. Conversely, ConRAT is differentiable, clearly outperforms strong models in terms of predictive and explanation performance, and it infers relevant, diverse, non-overlapping, and human-understandable concepts.

2.3 Topic Modeling.

Topic models, such as latent Dirichlet allocation (Blei et al., 2003), describe documents with a mixture of latent topics. Each topic represents a word distribution. Some studies combined topic models with recurrent neural models (Dieng et al., 2017; Zaheer et al., 2017). However, the goal of these generative models and the topics remains different than this work's. We aim to build a self-interpretable model that predicts and explains the outcome with latent concepts.

3 Concept-based Rationalizer (ConRAT)

Figure 2 depicts the architecture of our proposed self-explaining model: the Concept-based Rationalizer (ConRAT). Let X be a random variable representing a document composed of T words $(x_1, x_2, \ldots, x_T)$, y the ground-truth label, and K the desired numbers of concepts.[1] Given a document X and a label y, our goal is to explain the prediction $\hat{y}$ by finding a set of K concepts C_1, $\ldots, C_K$ that are masked versions of X. ConRAT learns concepts by maximizing the mutual information between $\boldsymbol{C}$ and y. We guide ConRAT to create separable and consistent concepts via two regularizers to make them human-understandable.

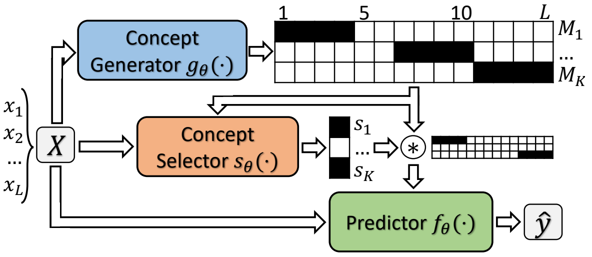

Figure 2: The proposed self-explaining model ConRAT. The model predicts and explains $\hat{y}$. Given a document X, the concept generator produces one binary mask per concept. The concept selector decides which concepts are present in the input. The predictor aggregates each selected concept's prediction to compute $\hat{y}$.

3.1 Model Overview

ConRAT is divided into three submodels: a **Concept Generator** $g_\theta(\cdot)$, which finds the concepts $C_1, \ldots, C_K$; a **Concept Selector** $s_\theta(\cdot)$, which detects whether a concept C_k is present or absent (i.e., $s_k \in \{1, 0\}$) in the input X; and a **Predictor** $f_\theta(\cdot)$, which predicts the outcome $\hat{y}$ based on the concepts $\boldsymbol{C}$ and their presence scores S.

3.1.1 Concept Generation

Inspired by the selective rationalization field (Lei et al., 2016), we define "concept" as a sequence of consecutive words in the input text. Previous studies extracted only one concept C_1 that is sufficient to explain the target variable y. In our work, a major difference is that we aim to find K concepts $C_1, \cdots, C_K$ that represent different topics or aspects and altogether explain the target variable y. We interpret the model as being linear in the concepts rather than depending on one overall selection of word. More formally, we define a concept as follows:

$$C_k = M_k \odot X, \tag{1}$$

where $M_k \in \mathbb{S}$ denotes a binary mask, $\mathbb{S}$ is a subset of $\mathbb{Z}_2^T$ with some constraints (introduced in Section 3.2), and $\odot$ is the element-wise multiplication of two vectors.

We parametrize the binary masks $\boldsymbol{M} \in \mathbb{Z}_2^{K \times T}$ with the concept generator model $g_\theta(\cdot)$, based on a bi-directional recurrent neural network. Following previous rationalization research (Yu et al.,

---

[1] Our method is easily adapted for regression problems.

2019; Chang et al., 2020), we force $g_\theta(\cdot)$ to select one chunk of text per concept with a pre-specified length $\ell \in [1, T]$.[2] Instead of predicting the mask M_k directly, $g_\theta(\cdot)$ produces a score for each position t. Then, it samples the start position t_k^* of the chunk for each C_k using the straight-through Gumbel-Softmax (Maddison et al., 2017; Jang et al., 2017). Finally, we compute M_k as follows:

$$\begin{aligned} T^* &\sim Gumbel(g_\theta(X)), \\ M_{k,t} &= \mathbb{1}\left[t \in [t_k^*, min(t_k^* + \ell - 1, T)]\right], \end{aligned} \tag{2}$$

where $\mathbb{1}$ denotes the indicator function. Although the equation is not differentiable, we can employ the straight-through technique (Bengio et al., 2013) and approximate it with the gradient of a causal convolution and a convolution kernel of an all-one vector of length ℓ.

3.1.2 Concept Selection

A key objective of ConRAT is to produce semantically consistent and separable concepts. So far, the generator $g_\theta(\cdot)$ generates K concepts for any input document. However, some documents might mention only a subset of those. Thus, the goal of the concept selector model $s_\theta(\cdot)$ is to enable ConRAT to ignore absent concepts.

Specifically, for each concept C_k, the model first computes a concept representation H_{C_k} using a standard attention mechanism (Bahdanau et al., 2015) (the tokens whose $M_{k,t} = 0$ are masked out). Then, we take the dot product of H_{C_k} with a weight vector, followed by a sigmoid activation function to induce the log-probabilities of a relaxed Bernoulli distribution (Jang et al., 2017). Finally, we sample the presence score $s_k \in \{0, 1\}$ of each concept independently:

$$S \sim RelaxedBernoulli(s_\theta(X, \boldsymbol{M})). \tag{3}$$

3.1.3 Prediction

As inputs, the predictor $f_\theta(\cdot)$ takes the document X, the masks $\boldsymbol{M}$, and the presence scores S for all concepts. First, we extract the concepts, which are masked versions of X. Differently than in Equation 1, the concepts are ignored if $s_k = 0$:

$$C_k = (M_k * s_k) \odot X. \tag{4}$$

Second, the model produces the hidden representation h'_{C_k} with another recurrent neural network, followed by a LeakyReLU activation function (Xu et al., 2015). Then, it computes the logits of y by applying a linear projection for each concept:

$$P_k = W h'_{C_k} + b, \tag{5}$$

where W and b are the projection parameters. Finally, f_θ computes the final outcome as follows:

$$p(y|C, \boldsymbol{M}, X) = softmax(\sum_{k=1}^{K} \alpha_k P_k s_k), \tag{6}$$

where α_k are model parameters that can be interpreted as the degree to which a particular concept contributes to the final prediction.

3.2 Unsupervised Discovery of Concepts

The above formulations integrate the explanation into the outcome computation. However, M_k is by definition faithful to the model's inner workings but not comprehensible for the end-user. Following Alvarez-Melis and Jaakkola (2018), we aim the concepts to follow three desiderata:1. **Fidelity**: they should preserve relevant information, 2. **Diversity**: they should be non-overlapping and diverse, and 3. **Grounding**: they should have an immediate human-understandable interpretations.

The hard constraint in Equation 2 naturally enforces the grounding by forcing the concept to be a sequence of ℓ words. For the fidelity, it is partly integrated in ConRAT by the prediction loss, which is the cross-entropy between the ground-truth label y and the prediction $\hat{y}$: $\mathcal{L}_{pred} = CE(\hat{y}, y)$. Recall that the concepts are substitutes of the input that are sufficient for the prediction. We emphasize the word "partly" because nothing prevents ConRAT from picking up spurious correlations.

We propose two regularizers to encourage ConRAT in finding non-overlapping, relevant, and dissimilar concepts. The first favors the orthogonality of concepts by penalizing redundant rows in $\boldsymbol{M}$:

$$\mathcal{L}_{overlap} = ||\boldsymbol{M}\boldsymbol{M}^T - \ell \cdot \mathbb{1}\,||_F^2, \tag{7}$$

where $||\cdot||_F$ stands for the Frobenius norm of a matrix, $\mathbb{1}$ denotes the identity matrix, and ℓ the pre-specified concept length. However, $\mathcal{L}_{overlap}$ alone does not prevent ConRAT from learning little relevant concepts. Therefore, we propose a second regularizer to encourage fidelity and diversity

---

[2] In early experiments, we relaxed the length constraint and generated instead K differentiable masks with continuity regularizers. However, this variant produced majorly inferior results. We hypothesize that there are too many constraints to optimize with only the target label as a strong signal.

by minimizing the cosine similarity between the concept representations H_{C_k} (see Section 3.1.2):

$$\mathcal{L}_{div} = \frac{1}{K}\frac{1}{K-1}\sum_{\substack{k_1,k_2=1\\k_1\neq k_2}}^{K} cos(H_{C_{k_1}}, H_{C_{k_2}}). \quad (8)$$

In both regularizers, we do not consider the presence scores S because a model could always select only one concept; this strategy is not optimal and reduces to a special case of rationale models (i.e., S would become a one-hot vector).

To summarize, the concepts are learned in an unsupervised manner and align with the three desiderata mentioned above: diversity is achieved with $\mathcal{L}_{overlap}$ and $\mathcal{L}_{div}$; fidelity is enforced by $\mathcal{L}_{pred}$ and $\mathcal{L}_{div}$, and the hard constraint in Equation 2 ensures the grounding. Finally, we train ConRAT end-to-end and minimize the loss jointly $\mathcal{L} = \mathcal{L}_{pred} + \lambda_O\mathcal{L}_{overlap} + \lambda_D\mathcal{L}_{div}$, where λ_O and λ_D control the impact of each regularizer.

3.3 Improving Overall Performance Further

The purpose of self-explaining models is to compute outcomes while being more interpretable. However, one key point is to achieve predictive performance comparable to that of black-box models. We propose two techniques to further improve both interpretability and performance; however, ConRAT does not require these techniques to outperform other methods, as we will see later.

Knowledge Distillation. We can train ConRAT not only via the information provided by the true labels but also by observing how a teacher model behaves (Hinton et al., 2015). In that case, we introduce the teacher model $T_\theta(\cdot)$, which is a simple recurrent neural network similar to the predictor f_θ. It is trained one the same data, but it uses the whole input X instead of subsets selected by each C_k. The overall training loss becomes $\mathcal{L} = \mathcal{L}_{pred} + \lambda_O\mathcal{L}_{overlap} + \lambda_D\mathcal{L}_{div} + \lambda_T(\hat{y}_{T_\theta} - \hat{y}_{f_\theta})^2$.

Pruning Concepts. Depending on the number of concepts and the pre-specified length, the total number of selected words can be close to or higher than the document length.[3] In practice, it is hard to extract meaningful concepts in such settings. To alleviate this problem, we propose to prune concepts at inference and select the top-k concepts that overlap the least with the others. More specifically, we compute the overlap as follows: for each sample in the validation set, we measure the average overlap ratio between M_{k_1} and M_{k_2} for each concept-pair $(C_{k_1}, C_{k_2}), k_1 \neq k_2$. Then, we select the top-k concepts whose scores are the lowest. Finally, to compute the new prediction $\hat{y}$, we update $s_k = 1$ if C_k is in the top-k or $s_k = 0$ otherwise.

Dataset	Amazon	Beer
# Reviews	$24,000$	$60,000$
Split Train/Val/Test	20k/2k/2k	50k/5k/5k
# Annotations	471	994
# Human Aspects	1	5
# Words per review	224 ± 125	184 ± 58

Table 1: Statistics of the review datasets.

4 Experiments

4.1 Datasets

We evaluate the quantitative performance of ConRAT using two binary classification datasets. The first one is the single-aspect Amazon Electronics dataset (Ni et al., 2019). We followed the filtering process in Chang et al. (2019) to keep only the reviews that contain evidence for both positive and negative sentiments. Specifically, we considered the first 50 tokens after the words "pros:" and "cons:" as the rationale annotations for the positive and negative labels, respectively. We randomly picked 24,000 balanced samples with ratings of four and above or two and below.

The second dataset comprises the multi-aspect beer reviews (McAuley et al., 2012) used in the field of rationalization (Lei et al., 2016; Yu et al., 2019). Each review describes various beer aspects: Appearance, Aroma, Palate, Taste, and Overall; users also provided a five-star rating for each aspect. However, we only use the overall rating for ConRAT. The dataset includes 994 beer reviews with sentence-level aspect annotations. Following the evaluation protocol in Bao et al. (2018); Chang et al. (2020), we binarized the ratings ≤ 2 as negative and ≥ 3 as positive. We sampled 60,000 balanced examples. Our setting is more challenging than those in previous studies because we assess the performance on all aspects (instead of three) and consider all examples for the sampling (instead of de-correlated subsets), reflecting the real data distribution. Table 1 shows the data statistics.

---

[3] e.g., if a document contains 200 tokens and we aim to extract 10 concepts of 20 tokens, all words should be selected.

4.2 Baselines

We consider the following baselines. **RNP** is a generator-predictor framework proposed by Lei et al. (2016) for rationalizing neural prediction. The generator selects text spans as rationales, which are then fed to the classifier for the final prediction. Yu et al. (2019) introduced **RNP-3P**, which extends RNP to include the complement predictor as the third player. It maximizes the predictive accuracy from unselected words. The training consists of an adversarial game with the three players. **Intro-3P** (Yu et al., 2019) improves RNP-3P by conditioning the generator on the predicted outcome of a teacher model. **InvRAT** is a game-theoretic method that competitively rules out spurious words with strong correlations to the output. The game-theoretic approach **CAR** aims to infer a rationale and a counterfactual rationale that counters the true label. We follow Chang et al. (2020) and consider for all methods their hard constraint variant (i.e., selecting one chunk of text) with different lengths for generating rationales.

RNP-3P and Intro-3P are trained with the policy gradient (Williams, 1992). The others estimate the gradients of the rationale selections using the straight-through technique (Bengio et al., 2013).

All rationalization methods, except CAR, strive for a single overall selection ($K = 1$) to explain the outcome. For the multi-aspect dataset, we train and tune each baseline independently for each aspect. The key difference with ConRAT is that the model is only trained on the overall aspect label and infers one rationale of K concepts; the baselines are trained K times to infer one rationale of one concept.

4.3 Experimental Details

To seek fair comparisons, we try to keep a similar number of parameters across all models, and we employ the same architecture for each player (generators, predictors, and discriminators/teachers) in all models: bi-directional gated recurrent units (Chung et al., 2014) with a hidden dimension 256. We use the 100-dimensional GloVe word embeddings (Pennington et al., 2014), Adam (Kingma and Ba, 2015) as optimization method with a learning rate of 0.001. We set the convolutional neural network in the concept selector similarly to (Kim et al., 2015) with 3-, 5-, and 7-width filters and 50 feature maps per filter. For ConRAT, we set the regularizer factors as follow: $\lambda_O = 0.05$, $\lambda_D = 0.05$, and $\lambda_T = 0.5$. We use the open-source implementation for all models, and we tune them by maximizing the prediction accuracy on the dev set with 16 random searches. For reproducibility purposes, we include additional details in Appendix A.

Table 2: Accuracy and objective performance of rationales in automatic evaluation for the Amazon dataset.

		Factual			*Counter Fact.*		
Model	**Acc.**	**P**	**R**	**F**	**P**	**R**	**F**
RNP	**75.5**	32.6	18.8	23.8		–	
RNP-3P	70.0	49.4	28.4	36.0		–	
Intro-3P	75.2	22.1	12.8	16.2		–	
InvRAT	71.5	44.3	25.5	32.4		–	
ConRAT-1	**75.5**	**56.4**	**32.5**	**41.3**		–	
CAR	73.6	33.0	19.1	24.2	**44.1**	**25.4**	**32.2**
ConRAT-6	**75.4**	**50.0**	**28.8**	**36.6**	32.3	18.6	23.6
ConRAT-4	75.3	46.4	26.7	33.9	29.6	17.1	21.6
ConRAT-2	75.3	33.7	19.4	24.6	8.9	5.1	6.5

4.4 RQ 1: Can ConRAT find evidence for factual and counterfactual rationales?

We aim to validate whether ConRAT can identify the two evidences for positive and negative sentiments. We set the concept length $\ell = 30$, we compare the generated rationales with the annotations, and we report the precision, recall, and F1 score. In this experiment, no teacher is used in ConRAT.

Table 2 contains the results. The top rows contain the results when only the factual rationales are considered for the evaluation, and ConRAT-1 uses only one concept. We see that ConRAT surpasses the baselines in finding rationales that align with human annotations, and it also matches the test accuracy with the baselines. Interestingly, we note that the baselines achieving the highest accuracy underperform in finding the correct rationales.

For the factual and counterfactual rationales, CAR finds one rationale to support the outcome and another one to counter it, in an adversarial game. However, the concepts inferred by ConRAT are not guaranteed to align with the rationales as there is no explicit signal to infer counterfactual concepts. Thus, we increase the number of concepts up to six and prune ConRAT to consider only the two most dissimilar concepts (see Section 3.3).

The bottom of Table 2 show the results. With only two concepts, ConRAT-2 outperforms CAR

Table 3: Objective performance of rationales for the multi-aspect beer reviews. ConRAT only uses the overall label and ignores the other aspect labels. All baselines are trained separately on each aspect rating. **Bold** and underline denote the best and second-best results, respectively.

			Average			*Appearance*			*Aroma*			*Palate*			*Taste*			*Overall*		
	Model	**Acc.**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
$\ell = 20$	RNP	81.1	30.7	22.1	24.9	30.8	23.2	26.5	22.1	21.0	21.5	17.7	24.1	20.4	28.1	16.7	20.9	54.9	25.8	35.1
	RNP-3P	80.5	29.1	22.5	25.0	30.4	25.6	27.8	19.3	20.4	19.8	10.3	12.0	11.1	43.9	28.4	34.5	41.6	26.0	32.0
	Intro-3P	85.6	24.2	19.6	21.3	28.7	24.8	26.6	14.3	14.4	14.3	16.6	19.3	17.9	24.2	13.6	17.4	37.0	25.9	30.5
	InvRAT	82.9	41.8	31.1	34.8	54.5	45.5	49.6	26.1	27.6	26.9	22.6	25.9	24.1	46.6	27.4	34.5	59.0	29.3	39.2
	ConRAT*	**91.4**	**50.0**	**42.0**	**44.9**	**57.8**	**53.0**	**55.3**	**31.9**	**35.5**	**33.6**	**29.0**	**36.3**	**32.3**	**56.5**	**33.9**	**42.4**	**74.9**	**51.0**	**60.7**
$\ell = 10$	RNP	84.4	41.3	16.6	23.2	40.1	12.0	18.5	**33.3**	**18.7**	**24.0**	**25.1**	**17.4**	**20.6**	32.3	9.8	15.07	76.0	25.1	37.8
	RNP-3P	83.1	31.1	13.5	18.6	41.8	19.2	26.3	22.2	12.4	15.9	16.5	10.4	12.7	33.2	10.6	16.1	41.9	14.7	21.8
	Intro-3P	80.9	21.8	10.8	14.3	51.0	26.0	34.4	18.8	9.7	12.8	16.5	10.6	12.9	9.7	2.6	4.1	13.1	5.2	7.4
	InvRAT	81.9	47.1	17.8	25.5	**59.4**	26.1	**36.3**	31.3	15.5	20.8	16.4	9.6	12.1	39.1	11.6	17.9	**89.1**	26.4	40.7
	ConRAT*	**91.3**	**48.1**	**20.1**	**28.0**	51.7	**26.2**	34.8	32.6	17.4	22.7	23.0	13.8	17.3	**45.3**	**13.1**	**20.3**	88.0	**30.1**	**44.9**

* The model is only trained on the overall label and does not have access to the other ground-truth labels.

in terms of test accuracy and matches the performance for the factual rationales, but it poorly identifies counterfactual rationales. However, there is a major improvement when we increase the number of concepts and use pruning. Indeed, the word distribution of the factual and counterfactual rationales are different, hence captured with pruning. ConRAT's factual rationales are better than those of all models. The counterfactual ones get closer to those produced by CAR. We show later in Section 4.6 that pruning helps in achieving better correlation with human judgments but is not required.

4.5 RQ 2: Are concepts inferred by ConRAT consistent with human rationalization?

We investigate whether ConRAT can recover all beer aspects by using only the overall ratings. Because beer reviews are smaller in length than Amazon ones, we set the concept length ℓ to 10 and 20. We fix the number of concepts to ten and prune ConRAT to keep five. We manually map them to the closest aspect for comparison. We trained the teacher model, used in Intro-3P and ConRAT, and obtained 91.4% accuracy. More results and illustrations are available in Appendix B and C.

Objective Evaluation. Similar to Section 4.4, we compare the generated rationales with the human annotations on the five aspects and the average performance. The main results are shown in Table 3. On average, ConRAT achieves the best performance while trained only on the overall ratings. This shows that the generated concepts, learned in an unsupervised manner, are separable, consistent, and correlated with human judgments to a certain extent. For the concept length $\ell = 20$, ConRAT produces significant superior results for all aspects, whereas the difference with InvRAT is less pronounced for $\ell = 10$. Finally, ConRAT's concepts lead to the highest accuracy and respect the grounding desideratum, thanks to the teacher.

We hypothesize that the baselines underperform due to the high correlations among the aspect ratings. Thus, they are more prone to pick up spurious correlations between the input features and the output. By considering multiple concepts simultaneously, ConRAT reduces the impact of spurious correlations. Regarding Intro-3P and RNP-3P, both suffer from instability issues due to the policy gradient (Chang et al., 2020; Yu et al., 2019).

We visualize an example in Figure 3. We observe that ConRAT induces interpretable concepts, while the best baselines suffer from spurious correlations. By reading our concepts alone, humans will easily predict the aspect label and its polarity.

Subjective Evaluation. We conduct a human evaluation using Amazon's Mechanical Turk (details in Appendix B.2) to judge the understandability of the concepts. Following Chang et al. (2019), we sampled 100 balanced reviews from the holdout set for each aspect, model, and concept length, resulting in 5,000 samples. We showed the examples in random order. An evaluator is presented with the concept generated by one of the five methods (unselected words are not visible). We credit a success when the evaluator guesses the true aspect label and its sentiment. We report the success rate as the performance metric. A random guess has a 10% success rate.

Figure 4 shows the main results. Similar to the objective evaluation, ConRAT reaches the

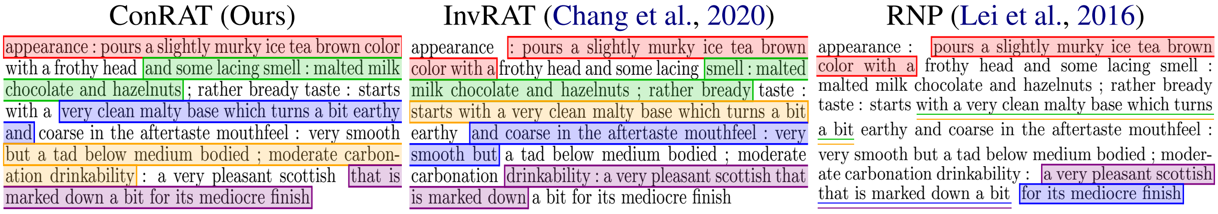

ConRAT (Ours)

appearance : pours a slightly murky ice tea brown color with a frothy head and some lacing smell : malted milk chocolate and hazelnuts ; rather bready taste : starts with a very clean malty base which turns a bit earthy and coarse in the aftertaste mouthfeel : very smooth but a tad below medium bodied ; moderate carbonation drinkability : a very pleasant scottish that is marked down a bit for its mediocre finish

InvRAT (Chang et al., 2020)

appearance : pours a slightly murky ice tea brown color with a frothy head and some lacing smell : malted milk chocolate and hazelnuts ; rather bready taste : starts with a very clean malty base which turns a bit earthy and coarse in the aftertaste mouthfeel : very smooth but a tad below medium bodied ; moderate carbonation drinkability : a very pleasant scottish that is marked down a bit for its mediocre finish

RNP (Lei et al., 2016)

appearance : pours a slightly murky ice tea brown color with a frothy head and some lacing smell : malted milk chocolate and hazelnuts ; rather bready taste : starts with a very clean malty base which turns a bit earthy and coarse in the aftertaste mouthfeel : very smooth but a tad below medium bodied ; moderate carbonation drinkability : a very pleasant scottish that is marked down a bit for its mediocre finish

Figure 3: Concepts generated (with ℓ=10) for a beer review. Underline highlights ambiguities. The color depicts the aspects: Appearance, Aroma, Palate, Taste, and Overall . **ConRAT is trained only on the overall label**.

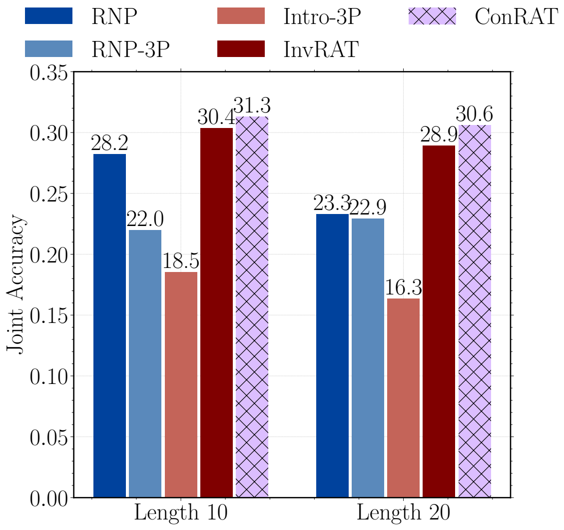

Figure 4: Subjective performance of rationales for the multi-aspect beer reviews. Evaluators need to guess both the sentiment and what aspect the concept is about, which makes random guess only 10%.

best performance, followed by InvRAT. Moreover, ConRAT only requires a single training on the overall aspect. It emphasizes that the discovered concepts satisfy the fidelity and diversity desiderata and better correlate with human judgments compared with supervised baselines.

4.6 RQ 3: How does the number of concepts K in ConRAT affect the performance?

We study the impact of the number of concepts K in ConRAT on the performance, as discussed in Section 4.5. We set the number of concepts to the number of aspects (K=5) and then increase it to K=10 and K=20. We prune ConRAT to keep only the five most dissimilar concepts (see Section 3.3).

Results are shown in Table 4. First, we observe that the performance is already better than the baselines in Table 3 with K=5. Second, when increasing K and pruning ConRAT, the performance is boosted further. However, we remark that the interpretability of the concepts follows a bell curve and significantly decreases when K=20. One potential reason is that we expect overlaps between the discriminative concepts that relate to beer aspects.[4] Thus, the five most dissimilar concepts might align less with human-defined concepts.

Table 4: Impact of the number of concepts in ConRAT on the objective performance for the beer reviews.

			Average		
	#Concepts	**Acc.**	**P**	**R**	**F**
$\ell = 20$	$K = 5$	90.95	48.96	37.59	41.37
	$K = 10$	**91.35**	**50.02**	**41.96**	**44.86**
	$K = 20$	90.24	37.78	31.19	32.84
$\ell = 10$	$K = 5$	89.64	47.60	19.23	26.90
	$K = 10$	**91.25**	**48.12**	**20.11**	**27.97**
	$K = 20$	91.05	35.71	14.84	20.71

4.7 RQ 4: How does each module of ConRAT contribute to the overall performance?

Finally, we analyze the importance of each module in an ablation study. To avoid any bias from pruning, we set the number of concepts to five.[5]

Table 5 shows the results. When ConRAT ignores the overlapping or the diversity regularizer, we observe a large drop in the rationale performance. This is expected as the diversity desideratum is not encouraged anymore. However, we remark that the sentiment prediction accuracy increases, which is certainly caused by spurious correlation with the ground-truth label. When all concepts are considered ($s_k = 1 \ \forall k$), we note that the sentiment accuracy stays similar. However, the objective performance decreases by 10% for the precision and more than 20% for the recall and F1 score. These results align with prior work: users write opinions about the topics they care about (Musat and Faltings, 2015; Antognini

---

[4] As shown in Table 1, the mean length of beer reviews is 184 words. With ℓ=20 and C=20, 400 words are highlighted.

[5] We obtain similar results with K=10 and K=20.

Table 5: Ablation study of ConRAT with five concepts.

Model	Acc.	*Average* P	R	F
ConRAT	89.64	47.60	19.23	26.90
- No $\mathcal{L}_{overlap}$	91.05	31.50	13.16	18.37
- No $\mathcal{L}_{div}$	90.85	34.49	11.69	16.95
- No $s_\theta(\cdot)$:$s_k = 1 \forall k$	89.74	43.13	14.95	21.42
- No Teacher	86.52	45.31	19.65	26.99

et al., 2021a). ConRAT reduces the noise at training by selecting concepts described in the current document. Finally, the teacher model helps ConRAT to boost the sentiment accuracy by more than 3% absolute score, without affecting the rationale quality.

5 Conclusion

Providing explanations for automated predictions carries much more impact, increases transparency, and might even be vital. Previous works have proposed using rationale methods to explain the prediction of a target variable. However, they do not properly capture the multi-faceted nature of useful rationales. We proposed ConRAT, a novel self-explaining model that extracts a set of concepts and explains the outcome with a linear aggregation of concepts, similar to how humans reason.

Our second contribution is two novel regularizers that guide ConRAT to generate interpretable concepts. Experiments on both single- and multi-aspect sentiment classification datasets show that ConRAT, by using only the overall label, is the first to provide superior rationale and predictive performance compared with supervised state-of-the-art methods trained for each aspect label. Moreover, ConRAT produces concepts considered superior in interpretability when evaluated by humans.

References

Stephan Alaniz and Zeynep Akata. 2019. Explainable observer-classifier for explainable binary decisions. *arXiv preprint arXiv:1902.01780*.

David Alvarez-Melis and Tommi Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 412–421, Copenhagen, Denmark. Association for Computational Linguistics.

David Alvarez-Melis and Tommi S. Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7786–7795.

Diego Antognini, Claudiu Musat, and Boi Faltings. 2021a. Interacting with explanations through critiquing. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI 2021)*.

Diego Antognini, Claudiu Musat, and Boi Faltings. 2021b. Multi-dimensional explanation of target variables from documents. *Proceedings of the AAAI Conference on Artificial Intelligence, (AAAI 2021)*.

Sharon Lee Armstrong, Lila R. Gleitman, and Henry Gleitman. 1983. What some concepts might not be. *Cognition*, 13(3):263 – 308.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9*.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, Brussels, Belgium.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Diane Bouchacourt and Ludovic Denoyer. 2019. Educe: Explaining model decisions through unsupervised concepts extraction.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi Jaakkola. 2019. A game theoretic approach to class-wise selective rationalization. In *Advances in Neural Information Processing Systems*, pages 10055–10065.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi S Jaakkola. 2020. Invariant rationalization. *arXiv preprint arXiv:2003.09772*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2017. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. 2019. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, pages 9277–9286.

Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. 2019. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 3543–3556.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26*.

Been Kim, Julie A Shah, and Finale Doshi-Velez. 2015. Mind the gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems*, pages 2260–2268.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9*.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26*.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, pages 1020–1025, Washington, DC, USA.

Claudiu Musat and Boi Faltings. 2015. Personalizing product rankings using collaborative filtering on opinion-derived topic profiles. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. An information bottleneck approach for controlling conciseness in rationale extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1938–1952, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. 2020. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online. Association for Computational Linguistics.

Eleanor Quint, Garrett Wirka, Jacob Williams, Stephen Scott, and NV Vinodchandran. 2018. Interpretable classification via supervised variational autoencoders and differentiable decision trees.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153, International Convention Centre, Sydney, Australia. PMLR.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328, International Convention Centre, Sydney, Australia. PMLR.

Joshua Brett Tenenbaum. 1999. *A Bayesian framework for concept learning*. Ph.D. thesis, Massachusetts Institute of Technology.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi Jaakkola. 2019. Rethinking cooperative rationalization: Introspective extraction and complement control. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4094–4103, Hong Kong, China. Association for Computational Linguistics.

Manzil Zaheer, Amr Ahmed, and Alexander J Smola. 2017. Latent lstm allocation: Joint clustering and non-linear dynamic modeling of sequence data. In *International Conference on Machine Learning*, pages 3967–3976.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Interpretable basis decomposition for visual explanation. In *Computer Vision – ECCV 2018*, pages 122–138, Cham. Springer International Publishing.

A Additional Training Details

We tune all models on the dev set. We truncate all reviews to 320 tokens for the beer dataset and 400 tokens for Amazon reviews. We have operated a random search over 16 trials. All baselines, except CAR, are tuned for each aspect (80 trials in total for the five aspects). We chose the models achieving the lowest validation accuracy. Most of the time, all models converged under 30 epochs. The range of hyperparameters are the following for ConRAT (similar for other models):

- Learning rate: $[0.0005, 0.00075, 0.001]$;
- Batch size: $[128]$;
- Hidden size: $[256]$;
- λ_D: $[0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1.0]$;
- λ_O: $[0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1.0]$;
- λ_T: $[0.5, 0.6]$;
- Dropout: $[0.0, 0.1]$;
- Weight decay: $[0.0, 10^{-8}, 10^{-10}]$;
- Gumbel temperature in $f_\theta(\cdot)$: $[1.0; 1.5]$;
- Gumbel temperature in $s_\theta(\cdot)$: $[1.0; 1.5]$;

A.1 Hardware / Software

- **CPU**: 2x Intel Xeon E5-2680 v3, 2x 12 cores, 24 threads, 2.5 GHz, 30 MB cache;
- **RAM**: 16x16GB DDR4-2133;
- **GPU**: 2x Nvidia Titan X Maxwell;
- **OS**: Ubuntu 18.04;
- **Software**: Python 3, PyTorch 1.3, CUDA 10.

B Complementary Results RQ 2

B.1 Objective Evaluation

The results for the concept length $\ell = 5$ is shown in Table 6.

Moreover, we report in Table 7 the performance for the **unsupervised** sentiment prediction task for the aspects whose labels are not available to ConRAT: Appearance, Aroma, Palate, and Taste. As we can see, ConRAT achieves competitive results compared to **supervised** baselines.

B.2 Human Evaluation Details

We use Amazon's Mechanical Turk crowdsourcing platform to recruit human annotators to evaluate the quality of extracted justifications and the generated justifications produced by each model. To ensure high-quality of the collected data, we restricted the pool to native English speakers from the U.S., U.K., Canada, or Australia. Additionally, we set the worker requirements at a 98% approval rate and more than 1,000 HITS.

The user interface used to judge the quality of the justifications extracted from different methods, in Section 4.5, is shown in Figure 5.

B.3 Subjective Evaluation

All results (for the joint, the aspect, and the polarity accuracy) are shown in Figure 6. In total, we used 7,500 samples ($100 \times 5 \times 5 \times 3$).

We also studied the error rates on each aspect. The Aroma and Palate aspects cause the highest error for all models. One possible reason is that users confuse these with the aspect Taste, hence their high correlations in rating scores (Antognini et al., 2021b).

C Extra Visualizations

Additional samples of generated rationales are shown in Figure 7, 8, 9, and 10. We can observe that baselines suffer from spurious correlations: the rationale for the aspect Aroma, Palate, and Taste are often exchanged, or several rationales pick the same text snippets. On the other hand, ConRAT finds better concepts while only trained on the overall aspect label. As it has been shown in prior work (Lei et al., 2016; Chang et al., 2020; Antognini et al., 2021b) rationale methods suffer from the high correlation between rating scores because each model is trained independently for each aspect. Therefore, they rely on the assumption that the data have low internal correlations, which does not reflect the real data distribution. By contrast, ConRAT alleviates this problem be finding all concepts in one training.

Table 6: Objective performance of rationales for the multi-aspect beer reviews with the concept length set to five. ConRAT only uses the overall rating and does not have access to the other aspect labels. All baselines are trained separately on each aspect label. **Bold** and underline denote the best and second-best results, respectively.

			Average			*Appearance*			*Aroma*			*Palate*			*Taste*			*Overall*		
	Model	**Acc.**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
$\ell = 5$	RNP	80.8	41.3	10.4	16.4	50.9	**13.3**	**21.1**	**43.2**	**12.7**	**19.7**	27.1	10.0	14.5	5.5	0.59	1.07	80.0	15.3	25.7
	RNP-3P	81.5	32.9	6.9	11.2	35.1	7.3	12.1	25.6	7.2	11.3	17.0	5.2	8.0	28.6	4.0	7.1	58.2	10.5	17.8
	Intro-3P	84.6	29.8	7.0	11.3	47.3	12.4	19.7	35.4	9.9	15.5	9.7	2.8	4.3	24.3	3.8	6.6	32.4	6.3	10.6
	InvRAT	83.6	46.4	**11.4**	**18.1**	**51.0**	13.1	20.8	40.6	11.9	18.4	**32.0**	**11.8**	**17.2**	36.1	5.6	9.6	72.5	14.7	24.4
	ConRAT†	**90.4**	**46.6**	10.9	17.5	47.2	12.4	19.6	26.9	7.1	11.3	26.6	9.2	13.7	**39.2**	**6.2**	**10.8**	**93.1**	**19.5**	**32.21**

* The model is only trained on the overall label and does not have access to the other ground-truth labels.

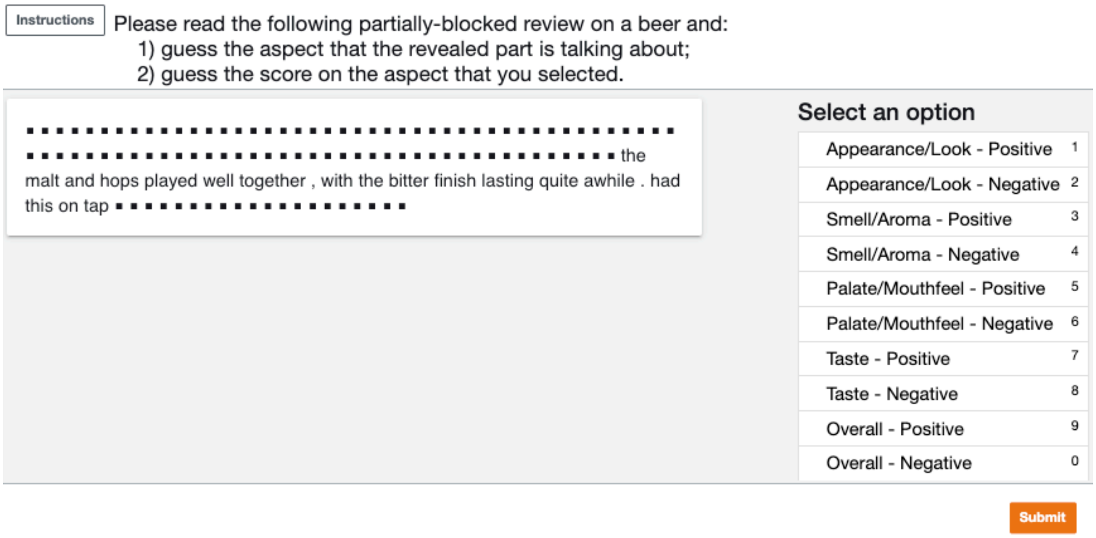
Instructions

Please read the following partially-blocked review on a beer and:
1) guess the aspect that the revealed part is talking about;
2) guess the score on the aspect that you selected.

■■■ ■■■ the malt and hops played well together , with the bitter finish lasting quite awhile . had this on tap ■■■■■■■■■■■■■■■■■■■■■■

Figure 5: Annotation platform for judging the quality of the concepts in the subjective evaluation on beer reviews.

Table 7: Performance on the overall sentiment and the aspects whose labels are not available to ConRAT. **Bold** and underline denote the best and second-best results.

	Model	**Ap.***	**Ar.***	**P***	**T***	**O**
$\ell = 5$	RNP	**95.98**	89.74	92.55	79.78	80.78
	RNP-3P	92.97	87.11	88.09	73.93	81.54
	Intro-3P	93.07	88.38	86.33	77.05	84.57
	InvRAT	**95.98**	90.44	92.66	88.63	83.60
	ConRAT	91.75*	**91.85***	**94.37***	**92.35***	**90.44**
$\ell = 10$	RNP	95.17	**92.15**	**90.74**	82.80	84.41
	RNP-3P	93.55	88.48	90.43	77.15	83.11
	Intro-3P	93.55	87.01	87.21	83.20	80.86
	InvRAT	**95.77**	90.54	89.03	85.01	81.89
	ConRAT	92.25*	91.05*	83.80*	**91.85***	**91.25**
$\ell = 20$	RNP	**96.08**	**92.15**	**94.37**	**87.02**	81.09
	RNP-3P	92.48	87.70	89.16	81.74	80.47
	Intro-3P	93.46	87.11	88.96	86.82	85.64
	InvRAT	95.88	91.05	89.44	85.11	82.90
	ConRAT	67.71*	74.85*	77.16*	80.58*	**91.35**

* ConRAT predicts the sentiment of the aspect in an unsupervised fashion.

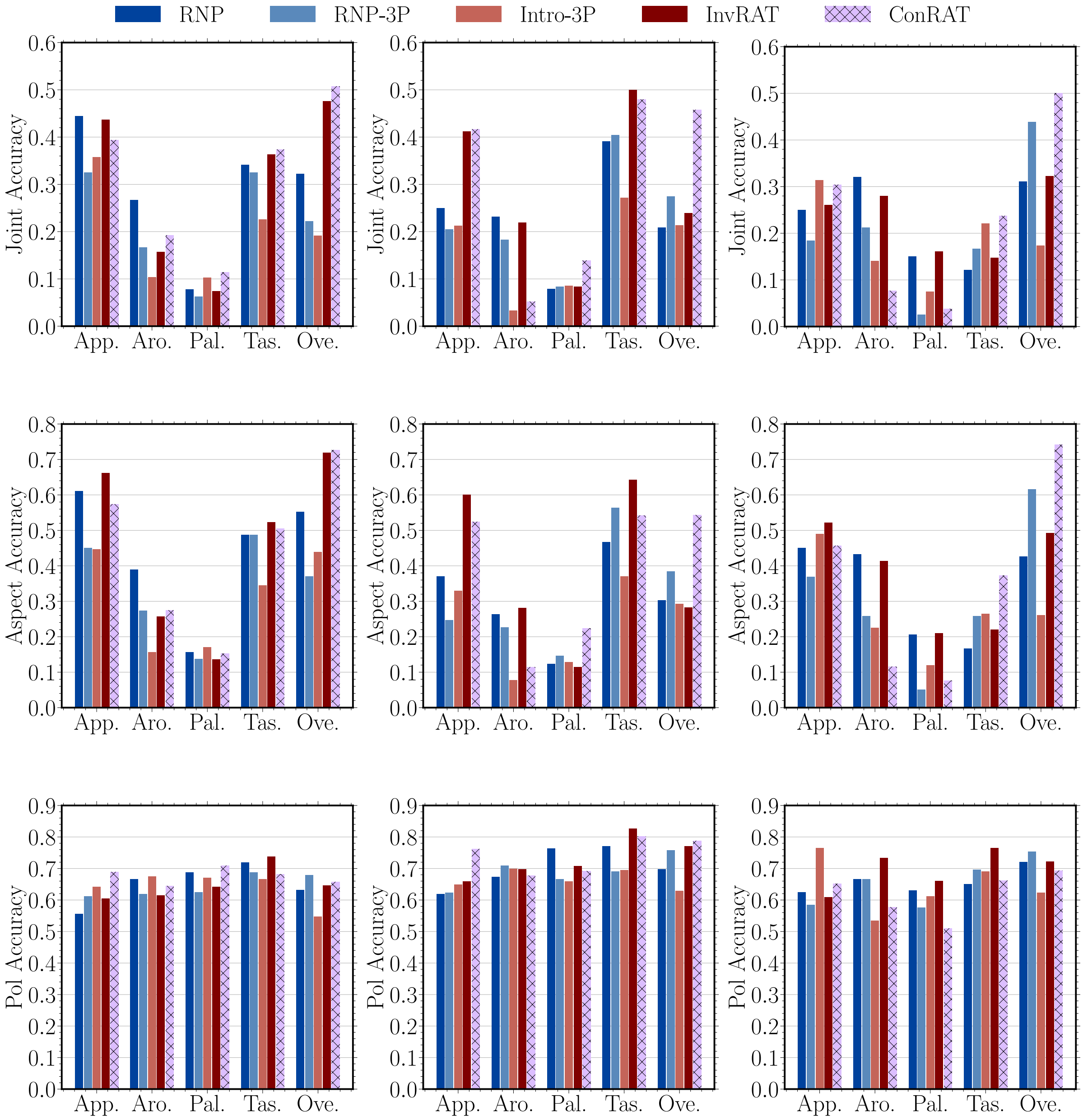

(a) Concept length $\ell = 10$.
(b) Concept length $\ell = 20$.
(c) Concept length $\ell = 5$.

Figure 6: Subjective performance per aspect of rationales for the multi-aspect beer reviews.

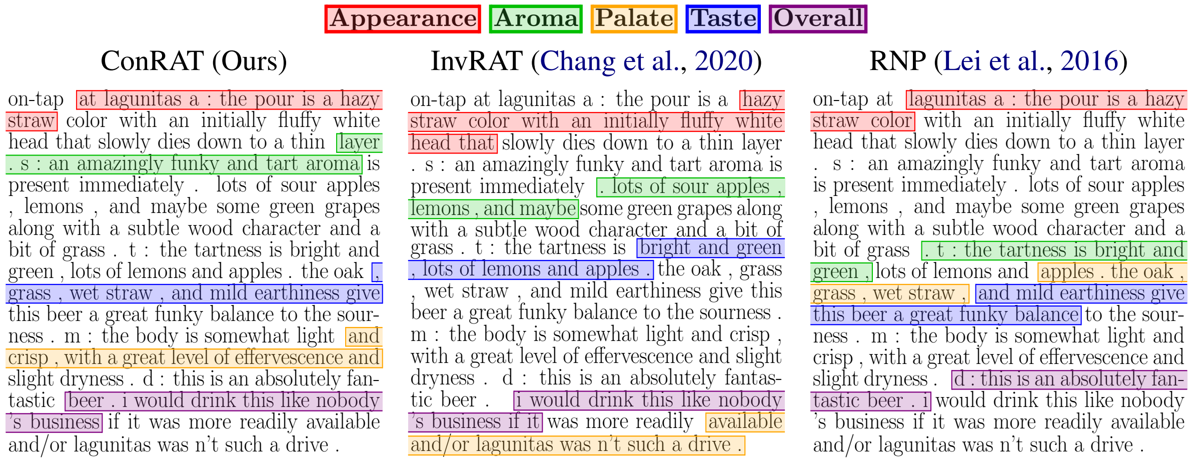
Appearance Aroma Palate Taste Overall

ConRAT (Ours)

on-tap at lagunitas a : the pour is a hazy straw color with an initially fluffy white head that slowly dies down to a thin layer . s : an amazingly funky and tart aroma is present immediately . lots of sour apples , lemons , and maybe some green grapes along with a subtle wood character and a bit of grass . t : the tartness is bright and green , lots of lemons and apples . the oak , grass , wet straw , and mild earthiness give this beer a great funky balance to the sourness . m : the body is somewhat light and crisp , with a great level of effervescence and slight dryness . d : this is an absolutely fantastic beer . i would drink this like nobody 's business if it was more readily available and/or lagunitas was n't such a drive .

InvRAT (Chang et al., 2020)

on-tap at lagunitas a : the pour is a hazy straw color with an initially fluffy white head that slowly dies down to a thin layer . s : an amazingly funky and tart aroma is present immediately . lots of sour apples , lemons , and maybe some green grapes along with a subtle wood character and a bit of grass . t : the tartness is bright and green . lots of lemons and apples . the oak , grass , wet straw , and mild earthiness give this beer a great funky balance to the sourness . m : the body is somewhat light and crisp , with a great level of effervescence and slight dryness . d : this is an absolutely fantastic beer . i would drink this like nobody 's business if it was more readily available and/or lagunitas was n't such a drive .

RNP (Lei et al., 2016)

on-tap at lagunitas a : the pour is a hazy straw color with an initially fluffy white head that slowly dies down to a thin layer . s : an amazingly funky and tart aroma is present immediately . lots of sour apples , lemons , and maybe some green grapes along with a subtle wood character and a bit of grass . t : the tartness is bright and green , lots of lemons and apples . the oak , grass , wet straw , and mild earthiness give this beer a great funky balance to the sourness . m : the body is somewhat light and crisp , with a great level of effervescence and slight dryness . d : this is an absolutely fantastic beer . i would drink this like nobody 's business if it was more readily available and/or lagunitas was n't such a drive .

Figure 7: Examples of generated rationales with $\ell = 10$ for a beer review. Underline highlights ambiguities.

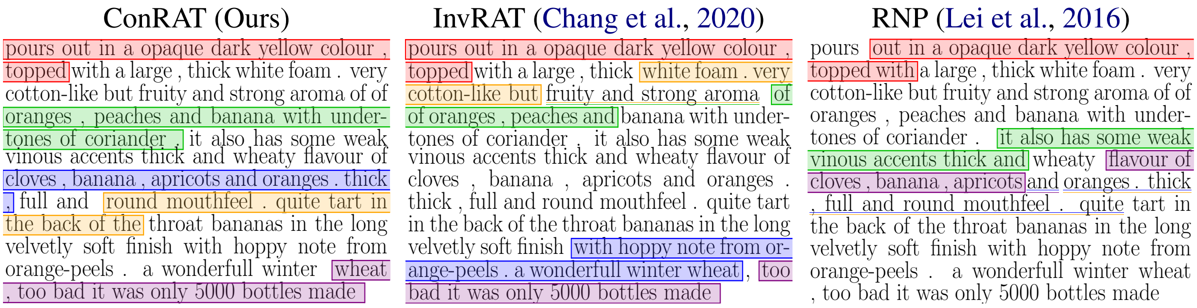
ConRAT (Ours)

pours out in a opaque dark yellow colour , topped with a large , thick white foam . very cotton-like but fruity and strong aroma of of oranges , peaches and banana with undertones of coriander . it also has some weak vinous accents thick and wheaty flavour of cloves , banana , apricots and oranges . thick , full and round mouthfeel . quite tart in the back of the throat bananas in the long velvetly soft finish with hoppy note from orange-peels . a wonderfull winter wheat , too bad it was only 5000 bottles made

InvRAT (Chang et al., 2020)

pours out in a opaque dark yellow colour , topped with a large , thick white foam . very cotton-like but fruity and strong aroma of of oranges , peaches and banana with undertones of coriander . it also has some weak vinous accents thick and wheaty flavour of cloves , banana , apricots and oranges . thick , full and round mouthfeel . quite tart in the back of the throat bananas in the long velvetly soft finish with hoppy note from orange-peels . a wonderfull winter wheat , too bad it was only 5000 bottles made

RNP (Lei et al., 2016)

pours out in a opaque dark yellow colour , topped with a large , thick white foam . very cotton-like but fruity and strong aroma of of oranges , peaches and banana with undertones of coriander . it also has some weak vinous accents thick and wheaty flavour of cloves , banana , apricots and oranges . thick , full and round mouthfeel . quite tart in the back of the throat bananas in the long velvetly soft finish with hoppy note from orange-peels . a wonderfull winter wheat , too bad it was only 5000 bottles made

Figure 8: Examples of generated rationales with $\ell = 10$ for a beer review. Underline highlights ambiguities.

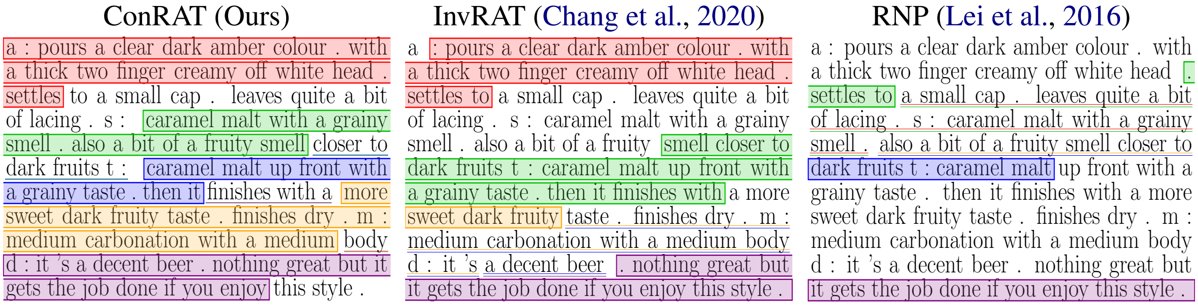
ConRAT (Ours)

a : pours a clear dark amber colour . with a thick two finger creamy off white head . settles to a small cap . leaves quite a bit of lacing . s : caramel malt with a grainy smell . also a bit of a fruity smell closer to dark fruits t : caramel malt up front with a grainy taste . then it finishes with a more sweet dark fruity taste . finishes dry . m : medium carbonation with a medium body d : it 's a decent beer . nothing great but it gets the job done if you enjoy this style .

InvRAT (Chang et al., 2020)

a : pours a clear dark amber colour . with a thick two finger creamy off white head . settles to a small cap . leaves quite a bit of lacing . s : caramel malt with a grainy smell . also a bit of a fruity smell closer to dark fruits t : caramel malt up front with a grainy taste . then it finishes with a more sweet dark fruity taste . finishes dry . m : medium carbonation with a medium body d : it 's a decent beer . nothing great but it gets the job done if you enjoy this style .

RNP (Lei et al., 2016)

a : pours a clear dark amber colour . with a thick two finger creamy off white head . settles to a small cap . leaves quite a bit of lacing . s : caramel malt with a grainy smell . also a bit of a fruity smell closer to dark fruits t : caramel malt up front with a grainy taste . then it finishes with a more sweet dark fruity taste . finishes dry . m : medium carbonation with a medium body d : it 's a decent beer . nothing great but it gets the job done if you enjoy this style .

Figure 9: Examples of generated rationales with $\ell = 20$ for a beer review. Underline highlights ambiguities.

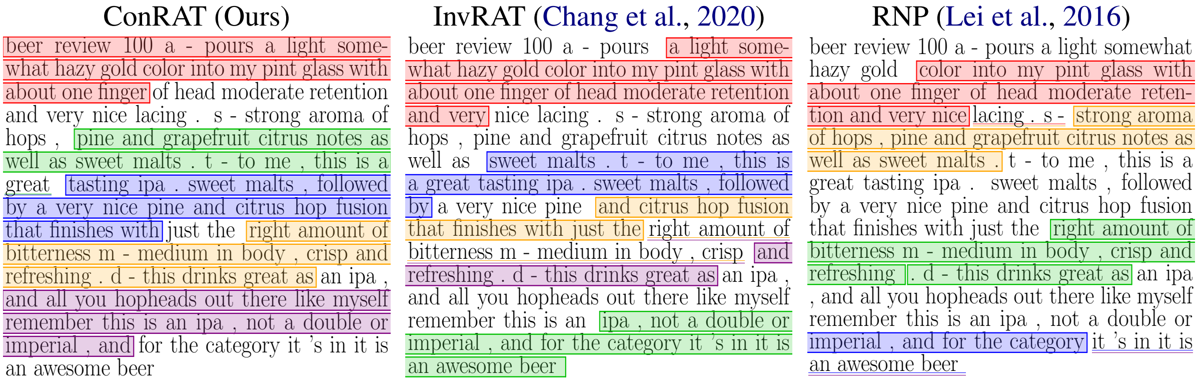
ConRAT (Ours)

beer review 100 a - pours a light somewhat hazy gold color into my pint glass with about one finger of head moderate retention and very nice lacing . s - strong aroma of hops , pine and grapefruit citrus notes as well as sweet malts . t - to me , this is a great tasting ipa . sweet malts , followed by a very nice pine and citrus hop fusion that finishes with just the right amount of bitterness m - medium in body , crisp and refreshing . d - this drinks great as an ipa , and all you hopheads out there like myself remember this is an ipa , not a double or imperial , and for the category it 's in it is an awesome beer

InvRAT (Chang et al., 2020)

beer review 100 a - pours a light somewhat hazy gold color into my pint glass with about one finger of head moderate retention and very nice lacing . s - strong aroma of hops , pine and grapefruit citrus notes as well as sweet malts . t - to me , this is a great tasting ipa . sweet malts , followed by a very nice pine and citrus hop fusion that finishes with just the right amount of bitterness m - medium in body , crisp and refreshing . d - this drinks great as an ipa , and all you hopheads out there like myself remember this is an ipa , not a double or imperial , and for the category it 's in it is an awesome beer

RNP (Lei et al., 2016)

beer review 100 a - pours a light somewhat hazy gold color into my pint glass with about one finger of head moderate retention and very nice lacing . s - strong aroma of hops , pine and grapefruit citrus notes as well as sweet malts . t - to me , this is a great tasting ipa . sweet malts , followed by a very nice pine and citrus hop fusion that finishes with just the right amount of bitterness m - medium in body , crisp and refreshing . d - this drinks great as an ipa , and all you hopheads out there like myself remember this is an ipa , not a double or imperial , and for the category it 's in it is an awesome beer

Figure 10: Examples of generated rationales with $\ell = 20$ for a beer review. Underline highlights ambiguities.

Parallel Attention Network with Sequence Matching for Video Grounding

Hao Zhang^{1,2}, **Aixin Sun**^{1}, **Wei Jing**^{2,3}
Liangli Zhen^{2}, **Joey Tianyi Zhou**^{2}, **Rick Siow Mong Goh**^{2}
^{1}School of Computer Science and Engineering, Nanyang Technological University, Singapore
^{2}Institute of High Performance Computing, A*STAR, Singapore
^{3}Institute for Infocomm Research, A*STAR, Singapore
`{hao007@e.,axsun@}ntu.edu.sg, 21wjing@gmail.com,`
`{zhenll,zhouty,gohsm}@ihpc.a-star.edu.sg`

Abstract

Given a video, video grounding aims to retrieve a temporal moment that semantically corresponds to a language query. In this work, we propose a **P**arallel **A**ttention **N**etwork with **Seq**uence matching (SeqPAN) to address the challenges in this task: multi-modal representation learning, and target moment boundary prediction. We design a self-guided parallel attention module to effectively capture self-modal contexts and cross-modal attentive information between video and text. Inspired by sequence labeling tasks in natural language processing, we split the ground truth moment into begin, inside, and end regions. We then propose a sequence matching strategy to guide start/end boundary predictions using region labels. Experimental results on three datasets show that SeqPAN is superior to state-of-the-art methods. Furthermore, the effectiveness of the self-guided parallel attention module and the sequence matching module is verified.[1]

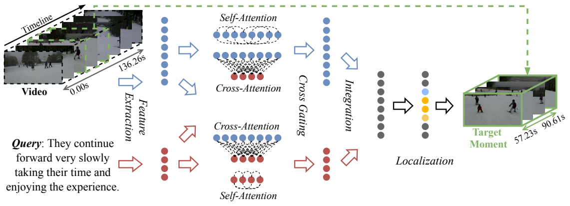

Figure 1: The overview of our procedures for video grounding, with an example of retrieving the temporal moment from an untrimmed video by a given language query.

1 Introduction

Video grounding is a fundamental and challenging problem in vision-language understanding research area (Hu et al., 2019; Yu et al., 2019; Zhu and Yang, 2020). It aims to retrieve a temporal video moment that semantically corresponds to a given language query, as shown in Figure 1. This task requires techniques from both computer vision (Tran et al., 2015; Shou et al., 2016; Feichtenhofer et al., 2019), natural language processing (Yu et al., 2018; Yang et al., 2019), and more importantly, the cross-modal interactions between the two. Many existing solutions (Chen et al., 2018; Liu et al., 2018a; Xu et al., 2019) tackle video grounding problem with *proposal-based* approach. This approach generates proposals with pre-set sliding windows or anchors, computes the similarity between the query and each proposal. The proposal with highest score is selected as the answer. These methods are sensitive to the quality of proposals and are inefficient because all proposal-query pairs are compared. Recently, several one-stage *proposal-free* solutions (Chen et al., 2019; Lu et al., 2019a; Mun et al., 2020) are proposed to directly predict start/end boundaries of target moments, through modeling video-text interactions. Our solution, SeqPAN, is a proposal-free method; hence our key focuses are *video-text interaction modeling* and *moment boundary prediction*.

Video-text interaction modeling. In order to model video-text interaction, various attention-based methods have been proposed (Gao et al., 2017; Yuan et al., 2019a; Mun et al., 2020). In particular, transformer block (Vaswani et al., 2017) is widely used in vision-language tasks and proved to be effective for multimodal learning (Tan and Bansal, 2019; Lu et al., 2019b; Su et al., 2020; Li et al., 2020). In video grounding task, fine-grain scale unimodal representations are important to achieve good localization performance. However, existing solutions do not refine unimodal representations of video and text when doing cross-modal reasoning, and thus limit the performance.

To better capture informative features for multimodalities, we encode both self-attentive contexts and cross-modal interactions from video and query. That is, instead of solely relying on sophisticated

---

[1] https://github.com/IsaacChanghau/SeqPAN

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 776–790
August 1–6, 2021.

Label:	B-ORG	I-ORG	I-ORG	E-ORG	O	O	O	O	...
Sent:	KinderCare	Learning	Centers	Inc.	said	that	a	debt	...

Figure 2: An example of the annotations in NER, where "ORG" is for "Organization", "B", "I" and "E" denote the begin, inside and end of the organization entity, respectively.

cross-modal learning as in most existing studies, we learn both intra- and inter-modal representations simultaneously, with improved attention modules.

Moment boundary prediction. In terms of the length, target moment is usually a very small portion of the video, making positive (frames in target moment) and negative (frames not in target moment) samples imbalanced. Further, we aim to predict the exact start/end boundaries (*i.e.,* two video frames[2]) of the target moment. If we view from the space of video frames, sparsity is a major concern, *e.g.,* catching two frames among thousands. Recent studies attempt to address this issue by auxiliary objectives, *e.g.,* to discriminate whether each frame is foreground (positive) or background (negative) (Yuan et al., 2019b; Mun et al., 2020), or to regress distances of each frame within target moment to ground truth boundaries (Lu et al., 2019a; Zeng et al., 2020). However, the "sequence" nature of frames or videos is not considered.

We emphasize the "sequence" nature of video frames and adopt the concept of sequence labeling in NLP to video grounding. We use named entity recognition (NER) (Lample et al., 2016; Ma and Hovy, 2016) as an example sequence labeling task for illustration in Figure 2. Video grounding is to retrieve a sequence of frames with start/end boundaries of target moment from video. This is analogous to extract a multi-word named entity from a sentence. The main difference is that, words are discrete, so word annotations (*i.e.,* B, I, E, and O tags) in sentence are discrete. In contrast, video is continuous and the changes between consecutive frames are smooth. Hence, it is difficult (and also not necessary) to precisely annotate each frame. We relax the annotations on video sequence by specifying video regions, instead of frames. With respect to the target moment, we label B, I, E and O (BIEO) regions on video (see Figure 3) and introduce label embeddings to model these regions.

Our contributions. In this research, we propose a **P**arallel **A**ttention **N**etwork with **Seq**uence matching (SeqPAN) for video grounding task. We first design a *self-guided parallel attention (SGPA)* module to capture both self- and cross-modal attentive information for each modality simultaneously. In SGPA module, a cross-gating strategy with self-guided head is further used to fuse self- and cross-modal representations. We then propose a *sequence matching (sq-match)* strategy, to identify BIEO regions in video. The label embeddings are incorporated to represent label of frames in each region for region recognition. The sq-match guides SeqPAN to search for boundaries of target moment within constrained regions, leading to more precise localization results. Experimental results on three benchmarks demonstrate that both SGPA and sq-match consistently improve the performance; and SeqPAN surpasses the state-of-the-art methods.

2 Related Work

Existing solutions to video grounding are roughly categorized into proposal-based and proposal-free frameworks. In proposal-based framework, common structures include ranking and anchor-based methods. *Ranking-based* methods (Liu et al., 2018b; Hendricks et al., 2017, 2018; Chen and Jiang, 2019; Ge et al., 2019; Zhang et al., 2019b) solve this task with two-stage propose-and-rank pipeline, which first generates proposals and then uses multimodal matching to retrieve most similar proposal for a query. *Anchor-based* methods (Chen et al., 2018; Yuan et al., 2019a; Zhang et al., 2019c; Wang et al., 2020b) sequentially assign each frame with multiscale temporal anchors and select the anchor with highest confidence as the result. However, these methods are sensitive to the proposal quality; and comparison of all proposal-query pairs is computational expensive and inefficient.

Proposal-free framework includes regression and span-based methods. *Regression-based* methods (Yuan et al., 2019b; Lu et al., 2019a; Chen et al., 2020a,b) tackle video grounding by learning cross-modal interactions between video and query, and directly regressing temporal time of target moments. *Span-based* methods (Ghosh et al., 2019; Rodriguez et al., 2020; Zhang et al., 2020a; Lei et al., 2020; Zhang et al., 2021) address video grounding by borrowing the concept of extractive question answering (Seo et al., 2017; Huang et al., 2018), and to predict the start and end boundaries of target moment directly.

In addition, there are several works (He et al.,

---

[2] The "frame" is a general description, which can refer to a frame in a video sequence or a unit in the corresponding video feature representation.

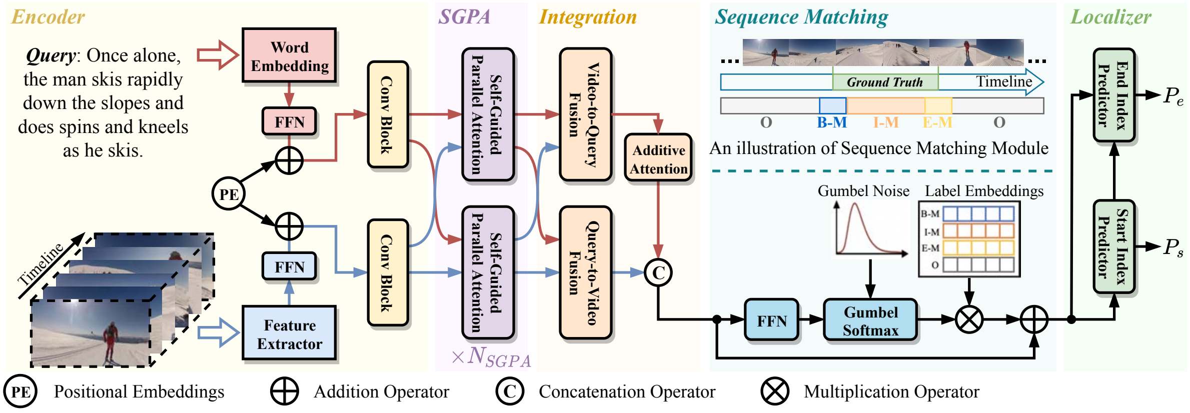

Figure 3: The architecture of the Parallel Attention Network with Sequence Matching (SeqPAN) for video grounding.

2019; Wang et al., 2019; Cao et al., 2020; Hahn et al., 2020; Wu et al., 2020a,b) that formulate this task as a sequential decision-making problem and adopt reinforcement learning to observe candidate moments conditioned on queries. Other methods, *e.g.,* weakly supervised learning methods (Mithun et al., 2019; Lin et al., 2020; Wu et al., 2020a), 2D map model of temporal relations between video moments (Zhang et al., 2020b), ensemble of top-down and bottom-up methods (Wang et al., 2020a), joint learning video-level matching and moment-level localization (Shao et al., 2018), have also been explored. Some works (Shao et al., 2018; Cao et al., 2020; Liu et al., 2020; Wang et al., 2020a) use either additional resources/features or different evaluation metrics, so their results are not directly comparable with many others, including ours.

3 Proposed Method

Let $V = [f_t]_{t=0}^{T-1}$ be an untrimmed video with T frames; $Q = [q_j]_{j=0}^{M-1}$ be a language query with M words; t^s and t^e denote start and end time point of ground-truth temporal moment. We define and tackle video grounding task in feature spaces. Specifically, we split the given video V into N clip units, and use pre-trained feature extractor to encode them into visual features $\boldsymbol{V} = [\boldsymbol{v}_i]_{i=0}^{N-1} \in \mathbb{R}^{d_v \times N}$, where d_v is visual feature dimension. Then the $t^{s(e)}$ are mapped to the corresponding indices $i^{s(e)}$ in the feature sequence, where $0 \leq i^s \leq i^e \leq N-1$. For the query Q, we encode words with pre-trained word embeddings as $\boldsymbol{Q} = [\boldsymbol{w}_j]_{j=0}^{M-1} \in \mathbb{R}^{d_w \times M}$, where d_w is word dimension. Given the pair of $(\boldsymbol{V}, \boldsymbol{Q})$ as input, video grounding aims to localize a temporal moment starting at i^s and ending at i^e.

3.1 The SeqPAN Model

The overall architecture of the proposed SeqPAN model is shown in Figure 3. Next, we present each module of SeqPAN in detail.

3.1.1 Encoder Module

Given visual features $\boldsymbol{V} \in \mathbb{R}^{d_v \times N}$ of the video and word embeddings $\boldsymbol{Q} \in \mathbb{R}^{d_w \times M}$ of the language query, we map them into the same dimension d with two FFNs[3], respectively. The encoder module mainly encodes the individual modality separately. As position encoding offers a flexible way to embed a sequence, when the sequence order matters, we first incorporate a position embedding to every input of both video and query sequences. Then, we adopt stacked 1D convolutional block to learn representations by carrying knowledge from neighbor tokens. The encoded representations are written as:

$$\begin{aligned} \boldsymbol{V}' &= \texttt{ConvBlock}(\texttt{FFN}_\texttt{v}(\boldsymbol{V}) + \boldsymbol{E}_p) \\ \boldsymbol{Q}' &= \texttt{ConvBlock}(\texttt{FFN}_\texttt{q}(\boldsymbol{Q}) + \boldsymbol{E}_p) \end{aligned} \tag{1}$$

where $\boldsymbol{V}' \in \mathbb{R}^{d \times N}$ and $\boldsymbol{Q}' \in \mathbb{R}^{d \times M}$; $\boldsymbol{E}_p$ denotes the positional embeddings. Both position embeddings and convolutional block are shared by the video and text features.

3.1.2 Self-Guided Parallel Attention Module

A self-guided parallel attention (SGPA) module (see Figure 4) is proposed to improve multimodal representation learning. Compared with standard transformer (TRM) encoder, SGPA uses two parallel multi-head attention blocks to learn both *uni-modal* and *cross-modal* representations simultaneously, and merge them with a cross-gating strategy[4].

---

[3] We denote the single-layer feed-forward network as FFN ($\texttt{FFN}(\boldsymbol{X}) = \boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b}$) in this work.

[4] A detailed comparison of SGPA and standard TRMs is summarized in Appendix.

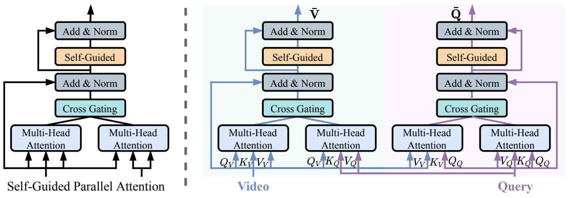

Figure 4: Self-Guided Parallel Attention (SGPA). Left: the structure of SGPA; Right: the parallel streams of encoding visual and textual inputs.

Taking video modality as an example, the attention process is computed as:

$$\begin{aligned}\hat{\boldsymbol{V}}_{\text{S}} &= V_V \cdot \sigma_s\Big(\frac{Q_V^\top K_V}{\sqrt{d}}\Big)\\ \hat{\boldsymbol{V}}_{\text{C}} &= V_Q \cdot \sigma_s\Big(\frac{Q_V^\top K_Q}{\sqrt{d}}\Big)\end{aligned} \tag{2}$$

where σ_s denotes Softmax function; Q_V, K_V and V_V are the linear projections of $\boldsymbol{V'}$; Q_Q, K_Q and V_Q are linear projections of $\boldsymbol{Q'}$; $\hat{\boldsymbol{V}}_{\text{S}}$ encodes the self-attentive contexts within video modality; and $\hat{\boldsymbol{V}}_{\text{C}}$ integrates information from query modality according to cross-modal attentive relations. The self- and cross-modal representations are then merged together by a cross-gating strategy:

$$\hat{\boldsymbol{V}} = \sigma\big(\texttt{FFN}(\hat{\boldsymbol{V}}_{\text{C}})\big) \odot \hat{\boldsymbol{V}}_{\text{S}} + \sigma\big(\texttt{FFN}(\hat{\boldsymbol{V}}_{\text{S}})\big) \odot \hat{\boldsymbol{V}}_{\text{C}} \tag{3}$$

where σ denotes Sigmoid function and $\odot$ represents Hadamard product. The cross-gating explicitly interacts features obtained from the self- and cross-attention encoders to ensure both are fully utilized, instead of relying on only one of them. Finally, we employ a self-guided head to implicitly emphasize the informative representations by measuring the confidence of each element in $\hat{\boldsymbol{V}}$ as:

$$\bar{\boldsymbol{V}} = \sigma\big(\texttt{FFN}_\sigma(\hat{\boldsymbol{V}})\big) \odot \texttt{FFN}(\hat{\boldsymbol{V}}) \tag{4}$$

The refined representations $\bar{\boldsymbol{Q}}$ for the query modality are obtained in a similar manner (*e.g.,* swapping visual and query features).

3.1.3 Video-Query Integration Module

This module further enhances the cross-modal interactions between visual and textual features. It utilizes context-query attention (CQA) strategy (Yu et al., 2018) and aggregates text information for each visual element[5] (see Figure 3). Given $\bar{\boldsymbol{V}}$ and $\bar{\boldsymbol{Q}}$, CQA first computes the similarities, $\mathcal{S} \in \mathbb{R}^{N\times M}$, between each pair of $\bar{\boldsymbol{V}}$ and $\bar{\boldsymbol{Q}}$ features. Then two attention weights are derived by $\mathcal{A}_{\text{VQ}} = \mathcal{S}_r \cdot \bar{\boldsymbol{Q}}^\top$ and $\mathcal{A}_{\text{QV}} = \mathcal{S}_r \cdot \mathcal{S}_c^\top \cdot \bar{\boldsymbol{V}}^\top$, where $\mathcal{S}_r$/$\mathcal{S}_c$ are row-/column-wise normalization of $\mathcal{S}$ by Softmax. The query-aware video representations $\boldsymbol{V}^Q$ is computed by:

$$\boldsymbol{V}^Q = \texttt{FFN}\big([\bar{\boldsymbol{V}}; \mathcal{A}_{\text{VQ}}^\top; \bar{\boldsymbol{V}} \odot \mathcal{A}_{\text{VQ}}^\top; \bar{\boldsymbol{V}} \odot \mathcal{A}_{\text{QV}}^\top]\big) \tag{5}$$

where $\boldsymbol{V}^Q \in \mathbb{R}^{d\times N}$. Similarly, video-aware query representations $\boldsymbol{Q}^V \in \mathbb{R}^{d\times M}$ can be derived by swapping visual and textual inputs in CQA module. Then we encode $\boldsymbol{Q}^V$ into sentence representation $\boldsymbol{q}$ with additive attention (Bahdanau et al., 2015) and concatenate it with each element of $\boldsymbol{V}^Q$ as $\boldsymbol{H} = [\boldsymbol{h}_1, \dots, \boldsymbol{h}_n]$, where $\boldsymbol{h}_i = [\boldsymbol{v}_i^Q; \boldsymbol{q}]$. Finally, the query-attended visual representation is computed as $\bar{\boldsymbol{H}} = \texttt{FFN}(\boldsymbol{H}) \in \mathbb{R}^{d\times N}$.

3.1.4 Sequence Matching Module

As illustrated in Figure 3, we considers the frames within ground truth moment and several neighboring frames as foreground, while the rest as background. Then, we split the foreground into **B**egin, **I**nside, and **E**nd regions. For simplicity, we assign each region a label, *i.e.,* "B-M" for begin, "I-M" for inside, "E-M" for end region, and "O" for background. B-M/E-M explicitly indicate potential positions of the start/end boundaries. We also specify orthogonal label embeddings $\boldsymbol{E}_{\text{lab}} \in \mathbb{R}^{d\times 4}$ to represent those labels, and to infuse label information into visual features after region label predictions.

Note our approach is different from Lin et al. (2018) on temporal action proposal generation task, where the target proposal is split into start, centre, and end regions. The probability of a frame belonging to each of three regions is predicted separately in a regression manner, leading to three separate probability sequences, one for each region. The maximum probabilities in the sequences are used to guide proposal generations. In contrast, we formulate matching process as a multi-class classification problem and predict a concrete region label for each frame, *i.e.,* same as a sequence labeling task in NLP. Label embeddings are then assigned to the frames based on the labels of the predicted region.

A straightforward solution to predict the confidence of an element belonging to each region is multi-class classifier:

$$\boldsymbol{H}_{\text{seq}} = \texttt{FFN}_{\text{seq}}(\bar{\boldsymbol{H}}), \quad \boldsymbol{S}_{\text{seq}} = \sigma_s(\boldsymbol{H}_{\text{seq}}) \in \mathbb{R}^{4\times N} \tag{6}$$

where $\boldsymbol{S}_{\text{seq}}$ encodes the probabilities of each visual element in different regions. Then label index with

---

[5] We provide a detailed computation process in appendix.

highest probability from $\boldsymbol{S}_{\text{seq}}$ is selected to represent the predicted label for each visual element:

$$\boldsymbol{L}_{\text{lab}} = [\arg\max(\boldsymbol{S}_{\text{seq}}^{j})]_{j=0}^{N-1} \in \mathbb{R}^{N} \tag{7}$$

However, a major issue here is that Eq. 7 needs to sample from a discrete probability distribution, which makes the back-propagation of gradients through $\boldsymbol{S}_{\text{seq}}$ in Eq. 6 infeasible for optimizer. To make back-propagation possible, we adopt the Gumbel-Max (Gumbel, 1954; Maddison et al., 2014) trick to re-formulate Eq. 7 as:

$$\hat{\boldsymbol{L}}_{\text{lab}} = \left[\texttt{Onehot}(\arg\max(\boldsymbol{H}_{\text{seq}}^{j} + \boldsymbol{g}))\right]_{j=0}^{N-1} \tag{8}$$

where $\hat{\boldsymbol{L}}_{\text{lab}} \in \mathbb{R}^{4\times N}$. Then, we utilize the Gumbel-Softmax (Jang et al., 2017; Maddison et al., 2017) to relax the $\arg\max$ so as to make Eq. 8 being differentiable[6]. Formally, we use Eq. 9 to approximate Eq. 8 as:

$$\bar{\boldsymbol{L}}_{\text{lab}} = \sigma_s\big((\boldsymbol{H}_{\text{seq}} + \boldsymbol{g})/\tau\big) \in \mathbb{R}^{4\times N} \tag{9}$$

where τ is annealing temperature. As $\tau \rightarrow 0^{+}$, $\bar{\boldsymbol{L}}_{\text{lab}} \approx \hat{\boldsymbol{L}}_{\text{lab}}$, while $\tau \rightarrow \infty$, each element in $\bar{\boldsymbol{L}}_{\text{lab}}$ will be the same and the approximated distribution will be smooth. Note we use Eq. 8 during forward pass while Eq. 9 for backward pass to allow gradient back-propagation. As the result, the embedding lookup process is differentiable and the label-attended visual representations is derived as:

$$\widetilde{\boldsymbol{H}} = \boldsymbol{E}_{\text{lab}} \cdot \hat{\boldsymbol{L}}_{\text{lab}} + \bar{\boldsymbol{H}} \tag{10}$$

The training objective is defined as:

$$\mathcal{L}_{\text{seq}} = f_{\text{XE}}(\bar{\boldsymbol{L}}_{\text{lab}}, \boldsymbol{Y}_{\text{lab}}) + \|\boldsymbol{E}_{\text{lab}}^{\top}\boldsymbol{E}_{\text{lab}} \odot (\boldsymbol{1} - \boldsymbol{I})\|_{\text{F}}^{2} \tag{11}$$

where $\boldsymbol{Y}_{\text{lab}}$ denotes the ground truth sequence labels, $\boldsymbol{1}$ is the matrix with all elements being 1 and $\boldsymbol{I}$ is the identity matrix. The second term in Eq. 11 is the orthogonal regularization (Brock et al., 2019), which ensures $\boldsymbol{E}_{\text{lab}}$ to keep the orthogonality.

3.1.5 Localization Module

Finally, we present a conditioned localizer to predict the start and end boundaries of the target moment. The localizer consists of two stacked transformer blocks and two FFNs. The scores of start and end boundaries are calculated as:

$$\begin{aligned} \boldsymbol{H}_{\text{s}} &= \texttt{TRM}_s(\widetilde{\boldsymbol{H}}),\ \boldsymbol{S}_s = \boldsymbol{W}_s[\boldsymbol{H}_{\text{s}}; \widetilde{\boldsymbol{H}}] + \boldsymbol{b}_s \\ \boldsymbol{H}_{\text{e}} &= \texttt{TRM}_e(\boldsymbol{H}_{\text{s}}),\ \boldsymbol{S}_e = \boldsymbol{W}_e[\boldsymbol{H}_{\text{e}}; \widetilde{\boldsymbol{H}}] + \boldsymbol{b}_e \end{aligned} \tag{12}$$

where $\boldsymbol{S}_{s/e} \in \mathbb{R}^{N}$. $\boldsymbol{W}_{s/e}$ and $\boldsymbol{b}_{s/e}$ are the weight and bias of start/end FFNs, respectively. Note the representations of end boundary ($\boldsymbol{H}_{\text{e}}$) are conditioned on that of start boundary ($\boldsymbol{H}_{\text{s}}$) to ensure the predicted end boundary is always after start boundary. Then, the probability distributions of start/end boundaries are computed by $\boldsymbol{P}_{s/e} = \texttt{Softmax}(\boldsymbol{S}_{s/e}) \in \mathbb{R}^{N}$. The training objective is:

$$\mathcal{L}_{\text{loc}} = \frac{1}{2} \times \left[f_{\text{XE}}(\boldsymbol{P}_s, \boldsymbol{Y}_s) + f_{\text{XE}}(\boldsymbol{P}_e, \boldsymbol{Y}_e)\right] \tag{13}$$

where f_{XE} is cross-entropy function, $\boldsymbol{Y}_{s/e}$ is one-hot labels for start/end (i^s/i^e) boundaries.

3.2 Training and Inference

The overall training loss of SeqPAN is: $\mathcal{L} = \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{seq}}$, to be minimized during the training process. During inference, the predicted start and end boundaries of a given video-query pair are generated by maximizing the joint probability as:

$$\begin{aligned} (\hat{i}^s, \hat{i}^e) &= \arg\max_{\hat{a}^s, \hat{a}^e} \boldsymbol{P}_s(\hat{a}^s) \times \boldsymbol{P}_e(\hat{a}^e) \\ &\text{s.t.: } 0 \leq \hat{i}^s \leq \hat{i}^e \leq N-1 \end{aligned} \tag{14}$$

where $\hat{i}^s$ and $\hat{i}^e$ are the best start and end boundaries of predicted moment for the given video-query pair. Let $\mathcal{T}$ be the duration of given video, the predicted start/end time are computed by $\hat{t}^{s(e)} = \hat{i}^{s(e)}/(N-1) \times \mathcal{T}$. With the predicted $(\hat{t}^s, \hat{t}^e)$ and ground truth (t^s, t^e) time intervals, the measure, temporal intersection over union (IoU), is computed as:

$$\text{IoU} = \max\left(0, \frac{t^e_{\text{min}} - t^s_{\text{max}}}{t^e_{\text{max}} - t^s_{\text{min}}}\right) \in [0, 1] \tag{15}$$

where $t^{s(e)}_{\text{min}\,/\,\text{max}} = \min / \max(\hat{t}^{s(e)}, t^{s(e)})$.

4 Experiments

4.1 Experimental Setting

Datasets. We conduct the experiments on three benchmark datasets: Charades-STA (Gao et al., 2017), ActivityNet Captions (Krishna et al., 2017) and TACoS (Regneri et al., 2013). **Charades-STA**, collected by Gao et al. (2017) from Charades (Sigurdsson et al., 2016) dataset, contains $16,128$ annotations (*i.e.,* moment-query pairs), where $12,408$ and $3,720$ annotations are for train and test. **ActivityNet Captions** (**ANetCaps**) contains 20K videos taken from ActivityNet (Heilbron et al., 2015) dataset. We follow the setup in (Chen et al., 2020a; Lu et al., 2019a; Wu et al., 2020b; Yuan

---

[6] More details about Gumbel Tricks are in Appendix.

Methods	R@1, IoU = μ $\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$	mIoU
DEBUG	54.95	37.39	17.69	36.34
ExCL	61.50	44.10	22.40	-
MAN	-	46.53	22.72	-
SCDM	-	54.44	33.43	-
CBP	-	36.80	18.87	-
GDP	54.54	39.47	18.49	-
2D-TAN	-	39.81	23.31	-
TSP-PRL	-	45.30	24.73	40.93
TMLGA	67.53	52.02	33.74	-
VSLNet	70.46	54.19	35.22	*50.02*
DRN	-	53.09	31.75	-
LGI	*72.96*	*59.46*	*35.48*	-
SeqPAN	**73.84**	**60.86**	**41.34**	**53.92**

Table 1: Comparison with SOTA methods on Charades-STA.

Methods	R@1, IoU = μ $\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$	mIoU
DEBUG	55.91	39.72	-	39.51
ExCL	*63.00*	43.60	24.10	-
SCDM	54.80	36.75	19.86	-
CBP	54.30	35.76	17.80	-
GDP	56.17	39.27	-	39.80
2D-TAN	59.45	44.51	*27.38*	-
TSP-PRL	56.08	38.76	-	39.21
TMLGA	51.28	33.04	19.26	-
VSLNet	**63.16**	43.22	26.16	*43.19*
DRN	-	*45.45*	24.36	-
LGI	58.52	41.51	23.07	-
SeqPAN	61.65	**45.50**	**28.37**	**45.11**

Table 2: Comparison with SOTA methods on ANetCaps.

Methods	R@1, IoU = μ $\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$	mIoU
TGN	21.77	18.90	-	-
ACL	24.17	20.01	-	-
DEBUG	23.45	11.72	-	16.03
SCDM	26.11	21.17	-	-
CBP	27.31	*24.79*	19.10	21.59
GDP	24.14	13.90	-	16.18
TMLGA	24.54	21.65	16.46	-
VSLNet	*29.61*	24.27	*20.03*	*24.11*
DRN	-	23.17	-	-
SeqPAN	**31.72**	**27.19**	**21.65**	**25.86**
2D-TAN	37.29	25.32	-	-
SeqPAN	**48.64**	**39.64**	**28.07**	**37.17**

Table 3: Comparison with SOTA methods on TACoS.

et al., 2019b; Zhang et al., 2020a) with $37,421$ and $17,505$ annotations for train and test. **TACoS** contains 127 cooking activities videos from Rohrbach et al. (2012). We follow Gao et al. (2017) with $10,146$, $4,589$, and $4,083$ annotations are used for train, validation, and test, respectively.

Evaluation Metric. (i) "R@n, IoU=μ", which denotes the percentage of test samples that have at least one result whose IoU with ground-truth is larger than μ in top-n predictions; (ii) "mIoU", which denotes the average IoU over all test samples. We set $n = 1$ and $\mu \in \{0.3, 0.5, 0.7\}$.

Implementation Details. We follow (Ghosh et al., 2019; Mun et al., 2020; Rodriguez et al., 2020; Zhang et al., 2020a) and use 3D ConvNet pretrained on Kinetics dataset (Carreira and Zisserman, 2017) to extract RGB visual features from videos; then we downsample the feature sequence to a fixed length. The query words are lowercased and initialized with GloVe (Pennington et al., 2014) embedding. We set hidden dimension d to 128; SGPA blocks to 2; annealing temperature to 0.3; and heads in multi-head attention to 8; Adam (Kingma and Ba, 2015) optimizer with batch size of 16 and learning rate of 0.0001 is used for training.

More details of dataset statistics and the hyperparameter settings are summarized in Appendix.

4.2 Comparison with State-of-the-Arts

We compare SeqPAN with the following state-of-the-arts. 1) *Proposal-based* methods: TGN (Chen et al., 2018), ACL (Ge et al., 2019), CBP (Wang et al., 2020b), SCDM (Yuan et al., 2019a), MAN (Zhang et al., 2019a); 2) *Proposal-free* methods: DEBUG (Lu et al., 2019a), ExCL (Ghosh et al., 2019), VSLNet (Zhang et al., 2020a), GDP (Chen et al., 2020a), LGI (Mun et al., 2020), TMLGA (Rodriguez et al., 2020), DRN (Zeng et al., 2020); 3) *Others*: TSP-PRL (Wu et al., 2020b) and 2D-TAN (Zhang et al., 2020b). The best results are in **bold** and the second bests are in *italic*. In all result tables, the scores of compared methods are reported in the corresponding works.

The results on the Charades-STA are summarized in Table 1. SeqPAN surpasses all baselines and achieves the highest scores over all metrics. Observe that the performance improvements of SeqPAN are more significant under more strict metrics. The results show that SeqPAN can produce more precise localization results. For instance, compared to LGI, SeqPAN achieves 5.86% absolute improvement by "R@1, IoU=0.7", and 1.40% by "R@1, IoU=0.5". Table 2 reports the results on ANetCaps. SeqPAN is superior to baselines and achieves the best performance on "R@1, IoU=0.7" and mean IoU. As reported in Table 3, similar observations hold on TACoS. Note 2D-TAN (Zhang et al., 2020b) pre-processes the TACoS dataset, making it is slightly different from the original one. We also conduct experiments on their version for a fair comparison. SeqPAN outperforms the base-

Methods	R@1, IoU = μ		
	$\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$
Charades-STA			
Se-TRM	68.84 (0.46)	51.92 (0.54)	34.58 (0.18)
Co-TRM	69.03 (0.49)	52.34 (0.50)	35.07 (0.32)
SGPA	69.47 (0.32)	54.63 (0.43)	36.36 (0.24)
ActivityNet Captions			
Se-TRM	57.64 (0.38)	40.76 (0.35)	25.10 (0.30)
Co-TRM	57.39 (0.29)	40.55 (0.45)	24.85 (0.47)
SGPA	58.40 (0.31)	41.72 (0.19)	26.07 (0.16)

Table 4: SGPA vs. standard transformers on Charades-STA and ANetCaps. Se-TRM is the transformer block with single modality inputs, and Co-TRM (Tan and Bansal, 2019; Lu et al., 2019b; Lei et al., 2020; Li et al., 2020) is with dual modality inputs. Scores in brackets are standard deviation.

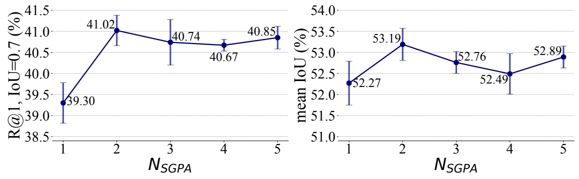

(a) Charades-STA

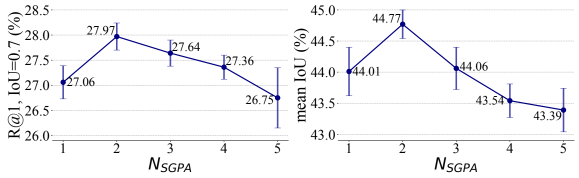

(b) ActivityNet Captions

Figure 5: The impact of SGPA block numbers (N_{SGPA}) on Charades-STA and ANetCaps.

lines over all evaluation metrics on both versions.

4.3 Discussion and Analysis

We perform in-depth ablation studies to analyze the effectiveness of the SeqPAN. We run all the experiments 5 times and report 5-run average.

Analysis on Self-Guided Parallel Attention. The SGPA (see Figure 4) is a variant of transformer (TRM), designed for learning cross-modality interactions between visual and text features. Here, we compare SGPA with standard TRMs. To better reflect the performance of different TRMs, we remove the sequence matching component and only use a single block (*i.e.,* $N_{\text{SGPA}} = 1$) in this experiment. The results are reported in Table 4. Observe that SGPA is superior to TRMs on both datasets. Co-TRM performs better on Charades-STA but worse on ANetCaps comparing with Se-TRM. Compared to Se-TRM and Co-TRM, SGPA learns both self-modal contexts and cross-modal interactions, which is approximately equivalent to parallel connection of two modules.

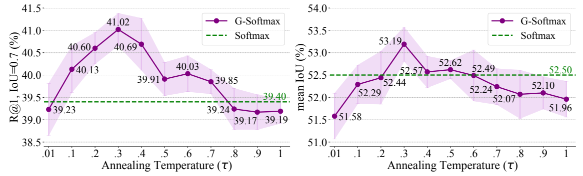

(a) Charades-STA

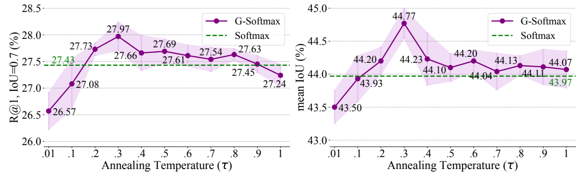

(b) ActivityNet Captions

Figure 6: The impact of annealing temperature τ in sequence matching on Charades-STA and ANetCaps.

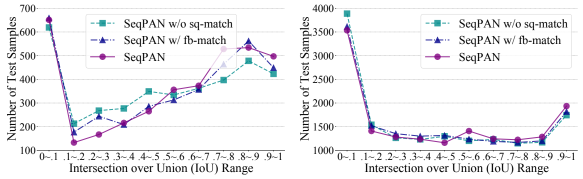

(a) Charades-STA (b) ActivityNet Captions

Figure 7: Plots of the number of predicted test samples within different IoU ranges on Charades-STA and ANetCaps.

Impact of SGPA block numbers N_{SGPA}. We now study the impact of SGPA block numbers on Charades-STA and ANetCaps. We evaluate five different values of N_{SGPA} from 1 to 5. The performance across the number of SGPA blocks in SeqPAN are plotted in Figures 5(a) and 5(b). Best performance is achieved at $N_{\text{SGPA}} = 2$ on both datasets. In general, along with increasing N_{SGPA}, the performance of SeqPAN first increases and then gradually decreases, on both datasets. We also note that performance on Charades-STA is not very sensitive to the setting of N_{SGPA}.

Analysis on Sequence Matching. The conventional matching strategy (Yuan et al., 2019b; Lu et al., 2019a; Mun et al., 2020) (denoted by fb-match) is to predict whether a frame is inside or outside of target moment, *i.e.,* foreground or background. In SeqPAN, we predict begin-, inside- and end-regions, and introduce label embeddings ($\boldsymbol{E}_{\text{lab}}$) to represent each region. The prediction process also uses the Gumbel-Max trick. In this experiment, we analyze the effects of label embeddings and Gumbel-Max trick in sequence matching.

Summarized in Table 5, both Gumbel-Max trick (denoted by G) and label embeddings contribute

Method	sq-match		Charades-STA				ActivityNet Captions			
			R@1, IoU $= \mu$			mIoU	R@1, IoU $= \mu$			mIoU
	G	$\boldsymbol{E}_{\text{lab}}$	$\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$		$\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$	
SeqPAN w/ fb-match	-	-	70.27(0.75)	56.96(0.46)	38.95(0.27)	51.84(0.40)	*59.99*(0.25)	43.71(0.19)	26.72(0.29)	43.23(0.23)
SeqPAN w/o sq-match	✘	✘	69.62(0.54)	55.29(0.30)	36.71(0.48)	51.13(0.25)	59.03(0.35)	42.65(0.32)	26.29(0.13)	42.51(0.36)
SeqPAN w/ Gumbel	✔	✘	*71.64*(0.64)	*57.61*(0.26)	*39.26*(0.31)	*52.15*(0.45)	59.74(0.42)	*43.85*(0.35)	*27.12*(0.20)	*43.69*(0.24)
SeqPAN	✔	✔	**72.70**(0.51)	**60.15**(0.50)	**41.02**(0.36)	**53.19**(0.38)	**61.12**(0.39)	**45.09**(0.37)	**27.97**(0.27)	**44.77**(0.23)

Table 5: Ablation studies of sequence matching strategy in SeqPAN, where the values in bracket denote standard deviation.

to the grounding performance improvement. In addition, consistent improvements are observed by incorporating G and $\boldsymbol{E}_{\text{lab}}$ into the model. SeqPAN is superior to SeqPAN w/ fb-match over all evaluation metrics. The performance improvements are more significant under more strict metrics. The results show that sq-match is more effective than the fb-match strategy. Regional indication of potential positions of start/end boundaries does help the model to produce accurate predictions.

Impact of Annealing Temperature τ. We then analyze the impact of annealing temperature τ of Gumbel-Softmax in sequence matching module. Gumbel-Softmax distributions are identical to a categorical distribution when $\tau \to 0^+$. With $\tau \to \infty$, its distribution is smooth. We evaluate 11 different τ values from 0.01 to 1.0, where 0.01 is used to approximate 0.0 since 0.0 is not divisible. The results are compared against vanilla Softmax as a baseline. For vanilla Softmax, we multiply the probability distribution of labels with $\boldsymbol{E}_{\text{lab}}$, to aggregate label information into the visual representations.

Figure 6 plots the results of different τ's on Charades-STA and ANetCaps, respectively. We observe similar patterns on the four sets of results. The best performance is achieved when $\tau = 0.3$ over both metrics on both datasets. From Figure 6(a), when τ is too small or too large (*i.e.,* the probability distribution from Gumbel-Softmax becomes too sharp or too smooth), Gumbel-Softmax performs poorer than vanilla Softmax. This result suggests that a proper annealing temperature τ is crucial to achieve good performance. Similar observations hold on ANetCaps (see Figure 6(b)).

4.4 Qualitative Analysis

Figure 7 shows the number of predicted test samples within different IoU ranges on Charades-STA and ANetCaps. Here, we compare SeqPAN with two of its variants: (i) removal of sequence matching module, and (ii) replacement of sequence matching with fb-match. All three variants show similar patterns. Nevertheless, within the higher IoU ranges, *e.g.,* IoU ≥ 0.5 on both datasets, SeqPAN and the variant with fb-match outperform the variant without sequence matching. The results show that having auxiliary objectives (*e.g.,* foreground/background or sequential regions) is helpful in video grounding task. Results in Figure 7 also show that our sequence matching is more effective than fb-match, for highlighting the correction regions for predicting start/end boundaries.

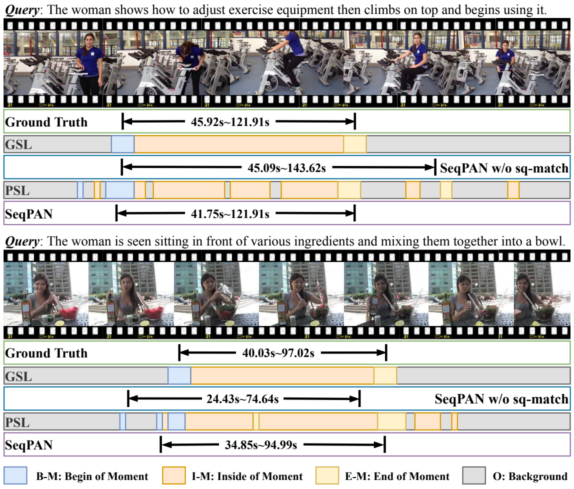

Figure 8: Qualitative results of SeqPAN and SeqPAN w/o sq-match on ANetCaps. "GSL" is the ground truth sequence labels; "PSL" is the predicted labels by sq-match of SeqPAN.

Figure 8 depicts two video grounding examples from the ANetCaps dataset. From the two examples, the moments retrieved by SeqPAN are closer to the ground truth than that are retrieved by SeqPAN without utilizing the sq-match strategy. Besides, the start and end boundaries predicted by SeqPAN are roughly constrained within the pre-set potential start and end regions. In addition, the predicted sequence labels (PSL) in Figure 8 also reveal the weakness of sq-match strategy. The predicted labels by sq-match strategy are not continuous, where multiple start, inside, and end regions are generated. In consequence, the localizer may be affected by wrongly predicted regions and leads to inaccurate results. To further constrain the generated regions is part of our future work.

5 Conclusion

In this work, we propose a Parallel Attention Network with Sequence matching (SeqPAN) to address the language query-based video grounding problem. We design a parallel attention module to improve the multimodal representation learning by capturing both self- and cross-modal attentive information simultaneously. In addition, we propose a sequence matching strategy, which explicitly indicates the potential start and end regions of the target moment to allow the localizer precisely predicting the boundaries. Through extensive experimental studies, we show that SeqPAN outperforms the state-of-the-art methods on three benchmark datasets; and both the proposed parallel attention and sequence matching modules contribute to the grounding performance improvement.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Fund (Project No. A18A1b0045 and A18A2b0046).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Yoshua Bengio, N. Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, abs/1308.3432.

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.

Da Cao, Yawen Zeng, Meng Liu, Xiangnan He, Meng Wang, and Zheng Qin. 2020. Strong: Spatio-temporal reinforcement learning for cross-modal video moment localization. In *Proceedings of the 28th ACM International Conference on Multimedia*.

João Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4733.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 162–171.

Jingyuan Chen, Lin Ma, Xinpeng Chen, Zequn Jie, and Jiebo Luo. 2019. Localizing natural language in videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8175–8182.

Long Chen, Chujie Lu, Siliang Tang, Jun Xiao, Dong Zhang, Chilie Tan, and Xiaolin Li. 2020a. Rethinking the bottom-up framework for query-based video localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Shaoxiang Chen, Wenhao Jiang, Wei Liu, and Yu-Gang Jiang. 2020b. Learning modality interaction for temporal sentence localization and event captioning in videos. In *The European Conference on Computer Vision*.

Shaoxiang Chen and Yu-Gang Jiang. 2019. Semantic proposal for activity localization in videos via sentence query. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33.

Xuanyi Dong and Yi Yang. 2019. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 1761–1770.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ramakant Nevatia. 2017. Tall: Temporal activity localization via language query. In *IEEE International Conference on Computer Vision*, pages 5277–5285.

Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. 2019. Mac: Mining activity concepts for language-based temporal localization. In *IEEE Winter Conference on Applications of Computer Vision*.

Soham Ghosh, Anuva Agarwal, Zarana Parekh, and Alexander Hauptmann. 2019. ExCL: Extractive Clip Localization Using Natural Language Descriptions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1984–1990.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Meera Hahn, Asim Kadav, James M Rehg, and Hans Peter Graf. 2020. Tripping through time: Efficient localization of activities in videos. In *The British Machine Vision Conference*.

Dongliang He, Xiang Zhao, Jizhou Huang, Fu Li, Xiao Liu, and Shilei Wen. 2019. Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8393–8400.

F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2018. Localizing moments in video with temporal language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan C. Russell. 2017. Localizing moments in video with natural language. In *2017 IEEE International Conference on Computer Vision*, pages 5804–5813.

Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. 2019. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6551–6557.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2018. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. 2017. Dense-captioning events in videos. In *IEEE International Conference on Computer Vision*, pages 706–715.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. 2020. Tvr: A large-scale dataset for video-subtitle moment retrieval. In *The European Conference on Computer Vision*.

Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. 2020. HERO: Hierarchical encoder for Video+Language omni-representation pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2046–2065.

Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. 2018. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision*, pages 3–19.

Zhijie Lin, Zhou Zhao, Zhu Zhang, Qi Wang, and Huasheng Liu. 2020. Weakly-supervised video moment retrieval via semantic completion network. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Bingbin Liu, Serena Yeung, Edward Chou, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. 2018a. Temporal modular networks for retrieving complex compositional activities in videos. In *The European Conference on Computer Vision*.

Daizong Liu, Xiaoye Qu, Xiao-Yang Liu, Jianfeng Dong, Pan Zhou, and Zichuan Xu. 2020. Jointly cross-and self-modal graph attention network for query-based moment localization. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4070–4078.

Meng Liu, Xiang Wang, Liqiang Nie, Qi Tian, Baoquan Chen, and Tat-Seng Chua. 2018b. Cross-modal moment localization in videos. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 843–851.

Chujie Lu, Long Chen, Chilie Tan, Xiaolin Li, and Jun Xiao. 2019a. DEBUG: A dense bottom-up grounding approach for natural language video localization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5147–5156.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019b. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094.

Niluthpol Chowdhury Mithun, Sujoy Paul, and Amit K. Roy-Chowdhury. 2019. Weakly supervised video moment retrieval from text queries. In *Computer Vision and Pattern Recognition*, pages 11592–11601.

Jonghwan Mun, Minsu Cho, and Bohyung Han. 2020. Local-global video-text interactions for temporal grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*, 1:25–36.

Cristian Rodriguez, Edison Marrese-Taylor, Fatemeh Sadat Saleh, Hongdong Li, and Stephen Gould. 2020. Proposal-free temporal moment localization of a natural-language query in video using guided attention. In *The IEEE Winter Conference on Applications of Computer Vision*.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. 2012. Script data for attribute-based recognition of composite activities. In *The European Conference on Computer Vision*.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Dian Shao, Yu Xiong, Yue Zhao, Qingqiu Huang, Yu Qiao, and Dahua Lin. 2018. Find and focus: Retrieve and localize video events with natural language queries. In *The European Conference on Computer Vision*.

Zheng Shou, Dongang Wang, and Shih-Fu Chang. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. Vl-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Hao Wang, Zheng-Jun Zha, Xuejin Chen, Zhiwei Xiong, and Jiebo Luo. 2020a. Dual path interaction network for video moment localization. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 4116–4124.

Jingwen Wang, Lin Ma, and Wenhao Jiang. 2020b. Temporally grounding language queries in videos by contextual boundary-aware prediction. In *AAAI*.

Weining Wang, Yan Huang, and Liang Wang. 2019. Language-driven temporal activity localization: A semantic matching reinforcement learning model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jie Wu, Guanbin Li, Xiaoguang Han, and Liang Lin. 2020a. Reinforcement learning for weakly supervised temporal grounding of natural language in untrimmed videos. In *Proceedings of the 28th ACM International Conference on Multimedia*.

Jie Wu, Guanbin Li, Si Liu, and Liang Lin. 2020b. Tree-structured policy based progressive reinforcement learning for temporally language grounding in video. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Huijuan Xu, Kun He, Bryan A. Plummer, L. Sigal, Stan Sclaroff, and Kate Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9062–9069.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5754–5764.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yitian Yuan, Lin Ma, Jingwen Wang, Wei Liu, and Wenwu Zhu. 2019a. Semantic conditioned dynamic modulation for temporal sentence grounding in videos. In *Advances in Neural Information Processing Systems*, pages 536–546.

Yitian Yuan, Tao Mei, and Wenwu Zhu. 2019b. To find where you talk: Temporal sentence localization in video with attention based location regression. In *AAAI*, volume 33, pages 9159–9166.

Runhao Zeng, Haoming Xu, Wenbing Huang, Peihao Chen, Mingkui Tan, and Chuang Gan. 2020. Dense regression network for video grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10287–10296.

Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. 2019a. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1247–1257.

Hao Zhang, Aixin Sun, Wei Jing, Liangli Zhen, Joey Tianyi Zhou, and Rick Siow Mong Goh. 2021. Natural language video localization: A revisit in span-based question answering framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. 2020a. Span-based localizing network for natural language video localization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6543–6554.

Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. 2020b. Learning 2d temporal adjacent networks formoment localization with natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Songyang Zhang, Jinsong Su, and Jiebo Luo. 2019b. Exploiting temporal relationships in video moment localization with natural language. In *Proceedings of ACM International Conference on Multimedia*.

Zhu Zhang, Zhijie Lin, Zhou Zhao, and Zhenxin Xiao. 2019c. Cross-modal interaction networks for query-based moment retrieval in videos. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Linchao Zhu and Yi Yang. 2020. Actbert: Learning global-local video-text representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8746–8755.

This appendix contains two sections. Section A provides (A.1) a detailed comparison between the proposed SGPA and standard transformer blocks, (A.2) technical details of the video-query integration module, and (A.3) categorical reparameterization used in the sequence matching module. Section B describes statistics on the benchmark datasets and parameter settings in our experiments.

A Additional Comparison and Technical Details

A.1 SGPA versus Standard Transformers

Two ways are mainly used to adopt the transformer block for multi-modal representation learning:

- Transformer block with the self-attention (Se-TRM), which encodes visual and textual inputs in separate streams, shown in Figure 9(a).
- Transformer block with the cross-attention (Co-TRM), which encodes both visual and textual inputs with interactions through co-attention, shown in Figure 9(b).

Several works (Lu et al., 2019a; Chen et al., 2020a; Zhang et al., 2020a) adopt Se-TRM to learn visual and textual representations in video grounding task. Se-TRM separately encodes each modality, it focuses on learning the refined unimodal representations within each modality for video and text respectively. Without any connection between two modalities, Se-TRM cannot use information from other modality to improve the representations.

Co-TRM[7] is commonly used as a basic component in various vision-language methods (Tan and Bansal, 2019; Lu et al., 2019b; Lei et al., 2020). Co-TRM relies on co-attention to learn the cross-modal representations for both visual and textual inputs. However, Co-TRM lacks the ability to encode self-attentive context within each modality.

The cascade of Se-TRM and Co-TRM is also used in recent vision-language models (Tan and Bansal, 2019; Lu et al., 2019b; Zhu and Yang, 2020; Lei et al., 2020) to learn both unimodal and cross-modal representations. In general, there are two cascade forms: 1) stacking Co-TRM upon Se-TRM (SeCo-TRM) in Figure 10(a); and 2) stacking Se-TRM upon Co-TRM (CoSe-TRM) in Figure 10(b). These stacked TRMs learn the unimodal and cross-modal information in a sequence manner. Hence, their final outputs focus more on either the self-attentive contexts or cross-modal interactions. Our SGPA combines advantages of both Se-TRM and Co-TRM, but not through cascade. As shown in Figure 9(c), SGPA contains two parallel multi-head attention blocks. One block takes single modality as input and the other takes both modalities as inputs. Thus, SGPA is able to learn both unimodal and cross-modal representations simultaneously. Then, a cross-gating strategy is designed to fuse the self- and cross-attentive representations. We also employ a self-guided head to replace the feed forward layer in transformer block. This design implicitly emphasizes informative representations by measuring the confidence of each element.

Methods	R@1, IoU $= \mu$		
	$\mu = 0.3$	$\mu = 0.5$	$\mu = 0.7$
Charades-STA			
Se-TRM	68.84 (0.46)	51.92 (0.54)	34.58 (0.18)
Co-TRM	69.03 (0.49)	52.34 (0.50)	35.07 (0.32)
SeCo-TRM	69.11 (0.24)	52.63 (0.49)	35.17 (0.22)
CoSe-TRM	69.08 (0.26)	52.82 (0.43)	35.09 (0.50)
PA	69.21 (0.27)	54.37 (0.46)	36.22 (0.49)
SGPA	69.47 (0.32)	54.63 (0.43)	36.36 (0.24)
ActivityNet Captions			
Se-TRM	57.64 (0.38)	40.76 (0.35)	25.10 (0.30)
Co-TRM	57.39 (0.29)	40.55 (0.45)	24.85 (0.47)
SeCo-TRM	57.47 (0.38)	40.70 (0.24)	25.07 (0.21)
CoSe-TRM	57.72 (0.41)	40.85 (0.17)	25.16 (0.15)
PA	58.27 (0.13)	41.59 (0.24)	25.88 (0.28)
SGPA	58.40 (0.31)	41.72 (0.19)	26.07 (0.16)

Table 6: Comparison between SGPA with standard transformer blocks on Charades-STA and ANetCaps, where PA is the SGPA without self-guided head (*i.e.,* replaced by FFN) The scores in bracket denotes standard deviation.

Table 6 reports the performance of SGPA and standard TRMs on Charades-STA and ANetCaps datasets. Here, we regard both SeCo-TRM and CoSe-TRM as single block. The results show that both PA (a SGPA variant without self-guided head) and SGPA are superior to standard TRMs.

A.2 Video-Query Integration Computation

This section presents the detailed computation process of video-query integration (see Section 3.1.3).

Given two inputs $\boldsymbol{X} \in \mathbb{R}^{d \times N_x}$ and $\boldsymbol{Y} \in \mathbb{R}^{d \times N_y}$, the context-query attention first computes similarities between each pair of $\boldsymbol{X}$ and $\boldsymbol{Y}$ elements as:

$$\mathcal{S} = \boldsymbol{X}^{\top} \cdot \boldsymbol{W} \cdot \boldsymbol{Y} \tag{16}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ and $\mathcal{S} \in \mathbb{R}^{N_x \times N_y}$. Then X-to-Y and Y-to-X attention weights are computed by:

---

[7] It is also known as co-attentional, multi-modal or cross-modal transformer block in different works.

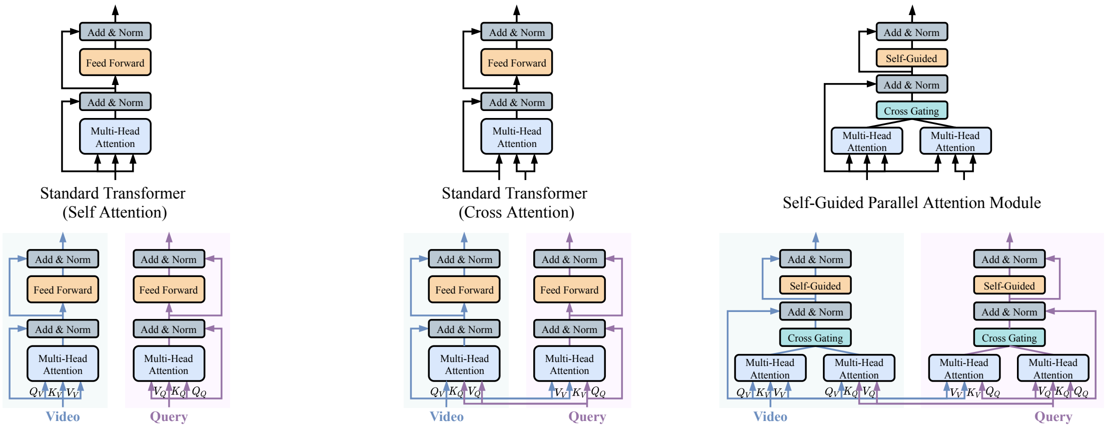

(a) Transformer Block (Self Attn) (b) Transformer Block (Cross Attn) (c) Self Guided Parallel Attention (Ours)

Figure 9: The structures of standard transformer blocks and self-guided parallel attention module. Top: the structure of each module; Bottom: the parallel streams of encoding visual and textual inputs. (a) The standard transformer block with self-attention; (b) The standard transformer block with cross-attention; (c) The self-guided parallel attention (SGPA) module.

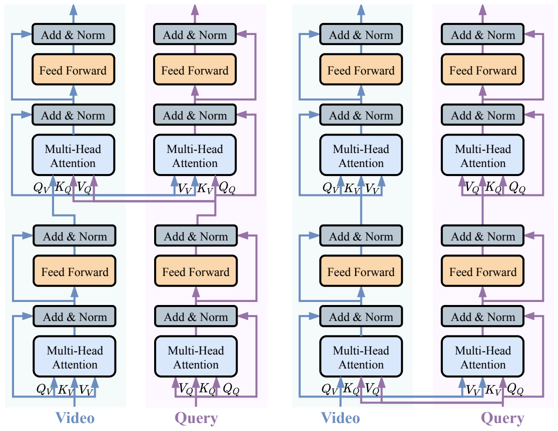

(a) SeCo-TRM Block (b) CoSe-TRM Block

Figure 10: The structures of SeCo-TRM and CoSe-TRM.

$$\mathcal{A}_{\text{XY}} = \mathcal{S}_r \cdot \boldsymbol{Y}^\top \in \mathbb{R}^{N_x \times d}$$
$$\mathcal{A}_{\text{YX}} = \mathcal{S}_r \cdot \mathcal{S}_c^\top \cdot \boldsymbol{X}^\top \in \mathbb{R}^{N_x \times d} \tag{17}$$

where $\mathcal{S}_r$ and $\mathcal{S}_c$ are the row- and column-wise normalization of $\mathcal{S}$ by Softmax function. The final output of context-query attention is calculated as:

$$\boldsymbol{X}^Y = \texttt{FFN}\big([\boldsymbol{X}; \mathcal{A}_{\text{XY}}^\top; \boldsymbol{X} \odot \mathcal{A}_{\text{XY}}^\top; \boldsymbol{X} \odot \mathcal{A}_{\text{YX}}^\top]\big) \tag{18}$$

where $\odot$ denotes element-wise multiplication, ";" represents concatenation operation, and $\boldsymbol{X}^Y \in \mathbb{R}^{d \times N_x}$. In this way, the information of $\boldsymbol{Y}$ is properly fused into $\boldsymbol{X}$.

By setting $\boldsymbol{X} = \bar{\boldsymbol{V}} \in \mathbb{R}^{d \times N}$ and $\boldsymbol{Y} = \bar{\boldsymbol{Q}} \in \mathbb{R}^{d \times M}$, we can derive the query-aware video representations $\boldsymbol{V}^Q \in \mathbb{R}^{d \times N}$. Similarly, the video-aware query representations $\boldsymbol{Q}^V \in \mathbb{R}^{d \times M}$ is obtained by setting $\boldsymbol{X} = \bar{\boldsymbol{Q}}$ and $\boldsymbol{Y} = \bar{\boldsymbol{V}}$.

Next, we encode $\boldsymbol{Q}^V = [\boldsymbol{q}_0^V, \ldots, \boldsymbol{q}_{M-1}^V]$ into sentence representation $\boldsymbol{q}$ with additive attention:

$$\boldsymbol{\alpha} = \texttt{Softmax}\big(\boldsymbol{W}_\alpha \cdot \boldsymbol{Q}^V)\big) \in \mathbb{R}^M$$
$$\boldsymbol{q} = \sum_{i=0}^{M-1} \alpha_i \times \boldsymbol{q}_i^V \in \mathbb{R}^d \tag{19}$$

where $\boldsymbol{W}_\alpha \in \mathbb{R}^{1 \times d}$. The $\boldsymbol{q}$ is then concatenated with each element of $\boldsymbol{V}^Q$ as $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n] \in \mathbb{R}^{2d \times N}$, where $\boldsymbol{h}_i = [\boldsymbol{v}_i^Q; \boldsymbol{q}]$. Finally, the query-attended visual representation is computed as

$$\bar{\boldsymbol{H}} = \boldsymbol{W}_h \cdot \boldsymbol{H} + \boldsymbol{b}_h \tag{20}$$

where $W_h \in \mathbb{R}^{d \times 2d}$ and $b_h \in \mathbb{R}^d$ denote the learnable weight and bias, and $\bar{\boldsymbol{H}} \in \mathbb{R}^{d \times N}$.

A.3 Categorical Reparameterization

This section provides a brief introduction of the categorical reparameterization strategy used in sequence matching module (see Section 3.1.4).

Categorical reparameterization, *e.g.,* reinforce-based approaches (Sutton et al., 2000; Schulman et al., 2015), straight-through estimators (Bengio et al., 2013) and Gumbel-Softmax (Jang et al., 2017; Maddison et al., 2017), is a strategy that enables discrete categorical variables to back-propagate in neural networks. It aims to estimate smooth gradient with a continuous relaxation for categorical variable. In this work, we use Gumbel-Softmax to approximate the sequence labels from a probability distribution. Then those labels are applied to lookup the corresponding embeddings for region representation in the sequence matching module of SeqPAN.

Dataset	Domain	N_{V} (train/val/test)	N_{A} (train/val/test)	$\bar{N}_{\text{A/V}}$	N_{Vocab}	$\bar{L}_V$ (s)	$\bar{L}_Q$	$\bar{L}_M$ (s)
Charades-STA	Indoors	$5,338$/-/$1,334$	$12,408$/-/$3,720$	2.42	$1,303$	30.59	7.22	8.22
ActivityNet Captions	Open	$10,009$/-/$4,917$	$37,421$/-/$17,505$	3.68	$12,460$	117.61	14.78	36.18
TACoS (Gao et al., 2017)	Cook	$75/27/25$	$10,146/4,589/4,083$	148.17	$2,033$	287.14	10.05	5.45
TACoS (Zhang et al., 2020b)			$9,790/4,436/4,001$	143.52	$1,983$	287.14	9.42	25.26

Table 7: Statistics of the evaluated video grounding benchmark datasets, where N_{V} is number of videos, N_{A} is number of annotations, $\bar{N}_{\text{A/V}}$ denotes the average number of annotations per video, N_{Vocab} is the vocabulary size of lowercase words, $\bar{L}_V$ denotes the average length of videos in seconds, $\bar{L}_Q$ denotes the average number of words in the sentence queries and $\bar{L}_M$ represents the average length of temporal moments in seconds.

Let $\boldsymbol{x} = (x_1, \ldots, x_l)$ be a categorical distribution, where l is the number of categories, x_c is the probability score of category c and $\sum_{c=1}^{l} x_c = 1$. Given the *i.i.d.* Gumbel noise $\boldsymbol{g} = (g_1, \ldots, g_l)$ from $\texttt{Gumbel}(0, 1)$ distribution[8], the soft categorical sample can be computed as:

$$\boldsymbol{y} = \texttt{Softmax}\big((\log(\boldsymbol{x}) + \boldsymbol{g})/\tau\big) \qquad (21)$$

where $\tau > 0$ is annealing temperature, and Eq. 21 is referred as Gumbel-Softmax operation on $\boldsymbol{x}$. As $\tau \to 0^+$, $\boldsymbol{y}$ is equivalent to the Gumbel-Max form (Gumbel, 1954; Maddison et al., 2014) as:

$$\hat{\boldsymbol{y}} = \texttt{Onehot}\big(\arg\max(\log(\boldsymbol{x}) + \boldsymbol{g})\big) \qquad (22)$$

where $\hat{\boldsymbol{y}}$ is an unbiased sample from $\boldsymbol{x}$ and thus we can draw differentiable samples from the distribution during training. Note, when input $\boldsymbol{x}$ is unnormalized, the $\log(\cdot)$ operator in Eq. 21 and 22 shall be omitted (Jang et al., 2017; Dong and Yang, 2019). During inference, discrete samples can be drawn with the Gumbel-Max trick directly.

B Dataset and Parameter Settings

B.1 Dataset Statistics

The statistics of the evaluated benchmark datasets are summarized in Table 7. **Charades-STA** dataset consists of $6,672$ videos and $16,128$ annotations (*i.e.,* moment-query pairs) in total. **ActivityNet Captions (ANetCaps)** dataset is taken from the ActivityNet (Heilbron et al., 2015). The average duration is about 120 seconds and each video contains 3.68 annotations on average. **TACoS** dataset contains 127 cooking activities videos with average duration of 4.79 minutes, and $18,818$ annotations in total. We follow the same train/val/test split as Gao et al. (2017). Besides, Zhang et al. (2020b) pre-processes the TACoS dataset, hence their version is slightly different from the original version. Detailed statistics are summarized in Table 7.

B.2 Hyper-Parameter Settings

We follow (Ghosh et al., 2019; Mun et al., 2020; Rodriguez et al., 2020; Zhang et al., 2020a) and use 3D ConvNet pre-trained on Kinetics dataset (*i.e.,* I3D[9]) (Carreira and Zisserman, 2017) to extract visual features from videos. The maximal visual feature sequence lengths are set to 64, 100, and 256 for Charades-STA, ActivityNet Captions, and TACoS, respectively. This setting is based on the average video lengths in the three datasets. The feature sequence length of a video will be uniformly downsampled if it is larger than the pre-set threshold, or zero-padding otherwise. For the language queries, we lowercase all the words and initialize them with GloVe (Pennington et al., 2014) embeddings[10]. The word embeddings and extracted visual features are fixed during training.

For other hyper-parameters, we use the same settings for all datasets. The dimension of the hidden layers is 128; the head number in multi-head attention is 8; the number of SGPA blocks (N_{SGPA}) is 2; the annealing temperature τ of Gumbel-Softmax is 0.3; The Dropout (Srivastava et al., 2014) is 0.2. The maximal training epochs $E = 100$ is used, with batch size of 16 and early stopping tolerance of 10 epochs. We adopt Adam (Kingma and Ba, 2015) optimizer, with initial learning rate of $\beta_0 = 0.0001$, weight decay 0.01, and gradient clipping 1.0, to train the model. The learning rate decay strategy is defined as $\beta_e = \beta_0 \times (1 - \frac{e}{E})$, where e denotes the e-th training epoch.

The SeqPAN is implemented using TensorFlow `1.15.0` with CUDA `10.0` and cudnn `7.6.5`. All the experiments are conducted on a workstation with dual NVIDIA GeForce RTX 2080Ti GPUs.

---

[8] The $\texttt{Gumbel}(0, 1)$ distribution can be sampled using inverse transform sampling by drawing $u \sim \texttt{Uniform}(0, 1)$ and computing $g = -\log(-\log(u))$ (Jang et al., 2017).

[9] https://github.com/deepmind/kinetics-i3d

[10] http://nlp.stanford.edu/data/glove.840B.300d.zip

MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training

Mingliang Zeng, Xu Tan*, Rui Wang, Zeqian Ju, Tao Qin, Tie-Yan Liu
Microsoft Research Asia
{v-minzeng,xuta,ruiwa,v-zeju,taoqin,tyliu}@microsoft.com

Abstract

Symbolic music understanding, which refers to the understanding of music from the symbolic data (e.g., MIDI format, but not audio), covers many music applications such as genre classification, emotion classification, and music pieces matching. While good music representations are beneficial for these applications, the lack of training data hinders representation learning. Inspired by the success of pre-training models in natural language processing, in this paper, we develop MusicBERT, a large-scale pre-trained model for music understanding. To this end, we construct a large-scale symbolic music corpus that contains more than 1 million music songs. Since symbolic music contains more structural (e.g., bar, position) and diverse information (e.g., tempo, instrument, and pitch), simply adopting the pre-training techniques from NLP to symbolic music only brings marginal gains. Therefore, we design several mechanisms, including OctupleMIDI encoding and bar-level masking strategy, to enhance pre-training with symbolic music data. Experiments demonstrate the advantages of MusicBERT on four music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification. Ablation studies also verify the effectiveness of our designs of OctupleMIDI encoding and bar-level masking strategy in MusicBERT.

1 Introduction

Music understanding, including tasks like genre classification, emotion classification, music pieces matching, has attracted lots of attention in both academia and industry. A better understanding of melody, rhythm, and music structure is not only beneficial for music information retrieval (Casey et al., 2008) but also helpful for music generation (Huang et al., 2018; Sheng et al., 2020). Similar to natural language, music is usually represented in symbolic data format (e.g., MIDI) (Jackendoff, 2009; McMullen and Saffran, 2004) with sequential tokens, and some methods (Mikolov et al., 2013a,b) from NLP can be adopted for symbolic music understanding. Since the labeled training data for each music understanding task is usually scarce, previous works (Liang et al., 2020; Chuan et al., 2020) leverage unlabeled music data to learn music token embeddings, similar to word embeddings in natural language tasks. Unfortunately, due to their shallow structures and limited unlabeled data, such embedding-based approaches have limited capability to learn powerful music representations.

In recent years, pre-trained language models (e.g., BERT) have been verified to be powerful for representation learning from large-scale unlabeled text corpora (Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019; Song et al., 2019; Brown et al., 2020; Song et al., 2020). However, it is challenging to directly apply the pre-training techniques from NLP to symbolic music because of the difference between natural text data and symbolic music data. First, since music songs are more structural (e.g., bar, position) and diverse (e.g., tempo, instrument, and pitch), encoding symbolic music is more complicated than natural language. The existing pianoroll-like (Ji et al., 2020) and MIDI-like (Huang and Yang, 2020; Ren et al., 2020) representations of a song are too long to be processed by pre-trained models. For example, the length of a music song encoded by REMI (Huang and Yang, 2020) has an average length of 15,679, as shown in Table 1. Due to the limits of computational resources, the length of sequences processed by a Transformer model is usually cropped to below 1,000. Thus such representations cannot capture sufficient information for song-level tasks. Accord-

---

[*] Corresponding author: Xu Tan, xuta@microsoft.com

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 791–800
August 1–6, 2021.

ingly, an effective, efficient, and universal symbolic music encoding method is needed for music representation learning. Second, due to the complicated encoding of symbolic music, the pre-training mechanism (e.g., the masking strategy like the masked language model in BERT) should be carefully designed to avoid information leakage in pre-training. Third, as pre-training relies on large-scale corpora, the lack of large-scale symbolic music corpora limits the potential of pre-training for music understanding.

Encoding	OctupleMIDI	CP-like	REMI-like
Tokens	**3607**	6906	15679

Table 1: The average number of tokens per song on LMD dataset with different encoding methods.

In this paper, we develop MusicBERT, a large-scale pre-trained model with carefully designed music encoding and masking strategy for music understanding.

- We design a novel music encoding method called OctupleMIDI, which encodes each note into a tuple with 8 elements. These 8 elements represent the different aspects of the characteristics of a musical note, including time signature, tempo, bar, position, instrument, pitch, duration, and velocity. OctupleMIDI has several advantages: 1) It largely reduces the length of a music sequence (4x shorter than REMI (Huang and Yang, 2020) and 2x shorter than CP (Hsiao et al., 2021)), thus easing the modeling of music sequences by Transformer considering that music sequences themselves are very long. 2) It is note centric. Since each note contains the same 8-tuple structure and covers adequate information to express various music genres, such as changing time signature and long note duration, OctupleMIDI is much simpler and more universal than previous encoding methods.

- We carefully analyze the masking strategies for symbolic music understanding and propose a bar-level masking strategy for MusicBERT. The masking strategy in original BERT for NLP tasks randomly masks some tokens, which will cause information leakage in music pre-training. For example, some attributes are usually the same in a segment of consecutive tokens, such as time signature, tempo, instrument, bar, and position. Therefore, the masked tokens can be easily predicted by directly copying from the adjacent tokens since they are probably the same. Meanwhile, adjacent pitches usually follow the same chord so that a masked pitch token can be easily inferred from the adjacent tokens in the same chord. Therefore, we propose a bar-level masking strategy, which masks all the tokens of the same type (e.g., time signature, bar, instrument, or pitch) in a bar to avoid information leakage and encourage effective representation learning.

- Last but not least, we collect a large-scale and diverse symbolic music dataset, denoted as Million MIDI Dataset (MMD), that contains more than 1 million music songs, with different genres, including Rock, Electronic, Rap, Jazz, Latin, Classical, etc. To our knowledge, it is the largest in current literature, which is 10 times larger than the previous largest dataset LMD (Raffel, 2016) in terms of the number of songs as shown in Table 2. Thus, this dataset greatly benefits representation learning for music understanding.

We fine-tune the pre-trained MusicBERT on four downstream music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification, and achieve state-of-the-art results on all the tasks. Furthermore, ablation studies show the effectiveness of the individual components in MusicBERT, including the OctupleMIDI encoding, the bar-level masking strategy, and the large-scale corpus.

The main contributions of this paper are summarized as follows:

- We pre-train MusicBERT on a large-scale symbolic music corpus that contains more than 1 million music songs and fine-tune MusicBERT on some music understanding tasks, achieving state-of-the-art results.

- We propose OctupleMIDI, an efficient and universal music encoding for music understanding, which leads to much shorter encoding sequences and is universal for various kinds of music.

- We design a bar-level masking strategy as the pre-training mechanism for MusicBERT, which significantly outperforms the naive token-level masking strategy used in natural language pre-training.

2 Related Works

2.1 Symbolic Music Understanding

Inspired by word2vec (Mikolov et al., 2013a,b) in NLP, previous works on symbolic music understanding learn music embedding by predicting a music symbol based on its neighborhood symbols. Huang et al. (2016); Madjiheurem et al. (2016) regard chords as words in NLP and learn chords representations using the word2vec model. Herremans and Chuan (2017); Chuan et al. (2020); Liang et al. (2020) divide music pieces into non-overlapping music slices with a fixed duration and train the embeddings for each slice. Hirai and Sawada (2019) cluster musical notes into groups and regard such groups as words for representation learning. However, the word2vec-based approaches mentioned above only use relatively small neural network models and take only a few (usually 4-5) surrounding music tokens as inputs, which have limited capability compared with recently developed deep and big pre-trained models like BERT (Devlin et al., 2018), which takes a long sentence (e.g., with 512 words/tokens) as input. In this paper, we pre-train big/deep models over a large-scale music corpus and use more context as input to improve symbolic music understanding.

2.2 Symbolic Music Encoding

There are two main approaches to encode symbolic music: pianoroll-based and MIDI-based.

In pianoroll-based methods (Ji et al., 2020; Brunner et al., 2018), music is usually encoded into a 2-dimensional binary matrix, where one dimension represents pitches, and the other represents time steps. Each element in the matrix indicates whether the pitch is played at that time step. As a result, a note is always divided into multiple fixed intervals, which is inefficient, especially for long notes.

MIDI is a technical standard for transferring digital instrument data. Many works in symbolic music (Oore et al., 2020; Huang et al., 2018) encode music pieces based on MIDI events, including note-on, note-off, time-shift, etc. REMI (Huang and Yang, 2020) improves the basic MIDI-like encoding using note-duration, bar, position, chord, and tempo. Inspired by REMI (Huang and Yang, 2020), PopMAG (Ren et al., 2020) and Compound Word (CP) (Hsiao et al., 2021) compress the attributes of a note, including pitch, duration, and velocity, into one symbol and reduces duplicated position events. Although such MIDI-like approaches avoid redundancy for long notes, they still need multiple tokens to represent the attributes, position, and metadata of a single note, which can be further compressed. This paper proposes OctupleMIDI, a MIDI-based encoding method, which is efficient due to the reduced sequence length and universal to support various music genres.

2.3 Masking Strategies in Pre-training

Masking strategies play a key role in NLP pre-training. For example, BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) randomly mask some tokens in an input sequence and learn to predict the masked tokens. Furthermore, since adjacent tokens may form a word or a phrase, some works consider masking consecutive tokens. For example, MASS (Song et al., 2019) randomly masks a fragment of several consecutive tokens in the input, and SpanBERT (Joshi et al., 2020) randomly masks contiguous spans instead of tokens. However, symbolic music is different from language. First, symbolic music contains structural (e.g., bar, position) and diverse information (e.g., tempo, instrument, and pitch), while natural language can be regarded as homogeneous data, which only contains text. Second, music and language follow different rules. Specifically, the language rules include grammar and spelling, while the music rules include beat, chord, etc. Accordingly, the masking strategies for symbolic music need to be specifically designed; otherwise, it may limit the potential of pre-training because of information leakage, as we analyzed before. In this paper, we carefully design a bar-level masking strategy for symbolic music pre-training.

3 Methodology

In this section, we introduce MusicBERT, a large-scale Transformer model for symbolic music understanding. We first overview the model structure and then describe the OctupleMIDI encoding and masking strategy for pre-training. At last, we describe the large-scale music corpus with over 1 million songs used in MusicBERT pre-training.

3.1 Model Overview

As shown in Figure 1, MusicBERT pre-trains a Transformer encoder (Vaswani et al., 2017; Devlin et al., 2018), with masked language modeling where some tokens in the input music sequence are masked and are predicted in the model output. To

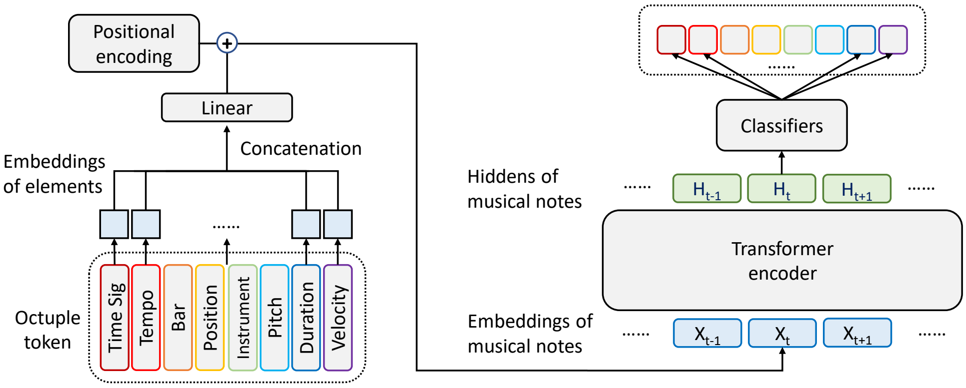

Figure 1: Model structure of MusicBERT.

encode the music sequence more efficiently, we propose a novel encoding method called OctupleMIDI, which encodes a symbolic music piece into a sequence of octuple tokens (an 8-tuple) that contains 8 basic elements related to a music note (we introduce OctupleMIDI in detail in Sec. 3.2). To convert the octuple tokens in each sequence step into the input of the Transformer encoder, we concatenate the embeddings of the 8 elements and use a linear layer to convert them into a single vector. Then, the converted vector is added with the corresponding position embeddings and taken as the input of the Transformer encoder. To predict each of the 8 tokens in the 8-tuple from the Transformer encoder, we add 8 different softmax layers to map the hidden of the Transformer encoder to the vocabulary sizes of 8 different element types, respectively.

3.2 OctupleMIDI Encoding

Previous works (Liang et al., 2020) encode the symbolic music in a pianoroll-like way, which is not efficient since a note is always divided into multiple fixed small intervals (e.g., a quarter note is represented with 8 consecutive tokens). MIDI-like approaches (Huang and Yang, 2020; Ren et al., 2020; Hsiao et al., 2021) encode a note into several tokens based on MIDI events, making encoding much shorter, and has been widely used in music generation tasks. However, previous MIDI-like representations are still long for the Transformer structure due to computation complexity and learning efficiency. Accordingly, we propose a compact symbolic music encoding method called OctupleMIDI for music understanding tasks. As shown in Fig. 2, OctupleMIDI encodes 6 notes into 6 tokens, which is much shorter than 33 tokens with REMI (Huang and Yang, 2020) and 16 tokens with CP (Hsiao et al., 2021). Meanwhile, OctupleMIDI is general for various kinds of music. For example, OctupleMIDI supports changeable time signature and tempo.

In OctupleMIDI, we use sequences of octuple tokens to represent symbolic music. Each octuple token corresponds to a note and contains 8 elements, including time signature, tempo, bar, position, instrument, pitch, duration, and velocity. We introduce the details of each element as follows:

- Time signature. A time signature is denoted as a fraction (e.g., 2/4), where the denominator is a power of two in range [1, 64], representing the length of a beat (measured by note duration, e.g., a quarter note in 2/4), and the numerator is an integer in range [1, 128], representing the number of beats in a bar (e.g., 2 beats in 2/4). The value of the fraction measures the duration of a bar normalized by a whole note (e.g., 2/4 means that the duration of a bar is a half note). We consider the duration of a bar is no more than two whole notes. Otherwise, we divide a long bar into several equal-duration bars no longer than two whole notes. Therefore, there are 254 different valid time signatures in OctupleMIDI.

- Tempo. Tempo is measured in beats per minute (BPM), which describes the pace of music. In most music samples, tempo values are in range [24, 200]. For OctupleMIDI encoding, we quantize tempo values to 49 different values from 16 to 256, forming a geometric sequence.

- Bar and position. We use bar and position to indicate the on-set time of a note hierarchically. In the coarse level, we use 256 tokens ranging from 0 to 255 to represent the bar, supporting up to 256 bars in a music piece, which is sufficient in most cases. In the fine-grained level (inside each bar),

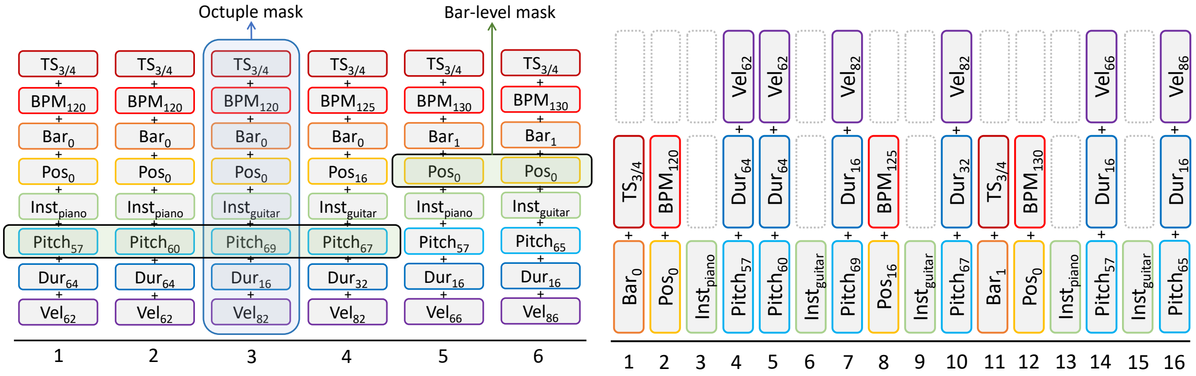

(a) OctupleMIDI encoding.

(b) CP-Like encoding.

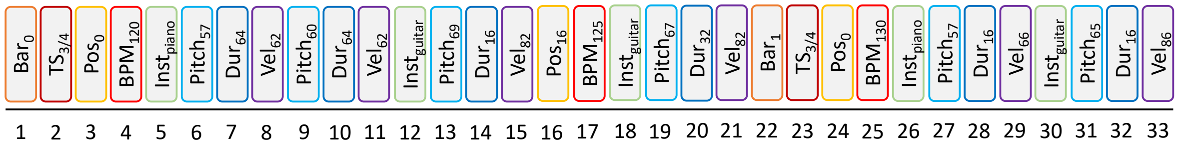

(c) REMI-Like encoding.

Figure 2: Different encoding methods for symbolic music.

we use position with a granularity of 1/64 note to represent the on-set time of a note, starting from 0 in each bar. Therefore, we need 128 tokens to represent position since the duration of a bar is no more than two whole notes, as described above. For example, in a bar with a time signature of 3/4, the possible value of position is from 0 to 47.

- Instrument. According to the MIDI format, we use 129 tokens to represent instruments, where 0 to 127 stands for different general instruments such as piano and bass, and 128 stands for the special percussion instrument such as drum.

- Pitch. For notes of general instruments, we use 128 tokens to represent pitch values following the MIDI format. However, for notes of percussion instruments, there are no pitches but percussion types (e.g., bass drum, hand clap). Therefore, we use another 128 "pitch" tokens to represent percussion type for percussion instruments.

- Duration. To support the long note duration (up to 60 whole notes in the common music genre) with a fixed set of duration tokens, we propose a mixed resolution method: using high resolution (e.g., sixty-fourth note) when the note duration is small and using a low resolution (e.g., thirty-second note or larger) when the note duration is large. Specifically, we use 128 tokens to represent duration, starting from 0, with an increment of sixty-fourth note for the first 16 tokens, and double the increment (i.e., thirty-second note) every time for next 16 tokens. The duration for percussion instruments is meaningless, so we always set them to 0.

- Velocity. We quantize the velocity of a note in the MIDI format into 32 different values with an interval of 4 (i.e., 2, 6, 10, 14, . . . , 122, 126).

An example of a music sequence in OctupleMIDI encoding is shown in Fig. 2a.

3.3 Masking Strategy

Inspired by the masked language model in BERT (Devlin et al., 2018), we randomly mask some elements in the input sequence of octuple tokens and predict the masked ones. A naive masking strategy is to randomly mask some octuple tokens (mask all the elements in an octuple token), which is denoted as octuple masking as shown in Fig. 2a. However, considering the specific rules of music, such naive strategy will cause information leakage, and thus cannot well learn the contextual representation.

A music song consists of multiple bars, which can be regarded as highly internally related units. An octuple token can be easily inferred from the adjacent tokens in the same bar. Specifically, time signature, tempo, and bar usually remain the same in the same bar. Instrument and position values in the same bar follow regular patterns, where the instrument is limited to a small-scale fixed set of values, and the position values are non decreasing. Moreover, a chord is a fixed combination of pitches, which always appear in adjacent positions. Accordingly, we propose a novel bar-level masking strategy, where the elements with the same type in the same bar are regarded as a unit and are masked simultaneously. In this way, information leakage can be avoided, and better contextual representation can be learned through pre-training. An example of bar-level masking is shown in Fig. 2a.

For the masked elements, 80% of them are replaced with $[MASK]$, 10% of them are replaced with a random element, and 10% remain unchanged, following the common practice (Devlin et al., 2018; Joshi et al., 2020; Liu et al., 2019). Inspired by RoBERTa (Liu et al., 2019), we remove the next sentence prediction task in pre-training and adopt a dynamic masking strategy, where the masked sequence is generated every time when feeding a sequence to the model.

3.4 Pre-training Corpus

A large-scale music dataset is necessary to learn good music representations from pre-training. However, previous symbolic music datasets are usually of small scale: 1) the MAESTRO dataset (Hawthorne et al., 2019) contains only one thousand piano performances; 2) the GiantMIDI-Piano dataset (Kong et al., 2020a,b) contains slightly larger but still only ten thousands of piano performances; 3) the largest open-sourced symbolic music dataset by now is the Lakh-MIDI Dataset (LMD) (Raffel, 2016), which contains about 100K songs.

To pre-train a powerful model with good music representations, we build a large-scale symbolic music corpus with over 1.5 million songs. Specifically, we first crawled a large amount of music files, cleaned files that are malformed or blank, and then converted those files into our symbolic music encoding. Since OctupleMIDI encoding is universal, most MIDI files can be converted to our encoding without noticeable loss of musical information. We found that these music files may have almost identical music content even if their hash values are different. Therefore, we developed an efficient way to deduplicate them: we first omitted all elements except instrument and pitch in the encoding, then got hash values of the remaining sequence and use it as the fingerprint of this music file, which is further used for deduplication. After cleaning and deduplication, we obtained 1.5 million songs with 2 billion octuple tokens (musical notes). We denote our dataset as Million-MIDI Dataset (MMD). We compare the sizes of different music datasets in Table 2.

Dataset	Songs	Notes (Millions)
MAESTRO	1,184	6
GiantMIDI-Piano	10,854	39
LMD	148,403	535
MMD	**1,524,557**	**2,075**

Table 2: The sizes of different music datasets. Since LMD also consists of MIDI files from various websites, we perform the same cleaning and deduplication process as used in MMD and get 148,403 songs in LMD.

4 Experiments and Results

In this section, we first introduce the pre-training setup for MusicBERT, and then fine-tune MusicBERT on several downstream music understanding tasks to compare it with previous approaches. Finally, more method analyses are conducted to verify the effectiveness of the designs in MusicBERT.

4.1 Pre-training Setup

Model Configuration We pre-train two versions of MusicBERT: 1) MusicBERT$_{\text{small}}$ on the small-scale LMD dataset, which is mainly for a fair comparison with previous works on music understanding such as PiRhDy (Liang et al., 2020) and melody2vec (Hirai and Sawada, 2019), which are also pre-trained on LMD; 2) MusicBERT$_{\text{base}}$ on the large-scale MMD dataset, for pushing the SOTA results and showing the scalability of MusicBERT. The details of the two MusicBERT models are shown in Table 4. We use our proposed bar-level masking strategy with a masking probability of 15%. In addition, we use tokens of 8 duplicated elements to represent the class token and the end of sequence token. They are also masked with a 15% probability.

Pre-training Details The average sequence length of OctupleMIDI representation of a song is 3607 tokens as shown in Table 1, which is too long to model in Transformer. Therefore, we randomly sample segments with a length of 1024 tokens for pre-training. Following Liu et al. (2019), we pre-train MusicBERT on 8 NVIDIA V100 GPUs for 4 days, and there are 125,000 steps in total, with a

Model	Melody Completion					Accompaniment Suggestion					Classification	
	MAP	HITS @1	HITS @5	HITS @10	HITS @25	MAP	HITS @1	HITS @5	HITS @20	HITS @25	Genre F1	Style F1
melody2vec$_{\text{F}}$	0.646	0.578	0.717	0.774	0.867	-	-	-	-	-	0.649	0.299
melody2vec$_{\text{B}}$	0.641	0.571	0.712	0.772	0.866	-	-	-	-	-	0.647	0.293
tonnetz	0.683	0.545	0.865	0.946	0.993	0.423	0.101	0.407	0.628	0.897	0.627	0.253
pianoroll	0.762	0.645	0.916	0.967	0.995	0.567	0.166	0.541	0.720	0.921	0.640	0.365
PiRhDy$_{\text{GH}}$	0.858	0.775	0.966	0.988	0.999	0.651	0.211	0.625	0.812	0.965	0.663	0.448
PiRhDy$_{\text{GM}}$	0.971	0.950	0.995	0.998	0.999	0.567	0.184	0.540	0.718	0.919	0.668	0.471
MusicBERT$_{\text{small}}$	0.982	0.971	0.996	0.999	1.000	0.930	0.329	0.843	0.993	0.997	0.761	0.626
MusicBERT$_{\text{base}}$	**0.985**	**0.975**	**0.997**	**0.999**	**1.000**	**0.946**	**0.333**	**0.857**	**0.996**	**0.998**	**0.784**	**0.645**

Table 3: Results of different models on the four downstream tasks: melody completion, accompaniment suggestion, genre classification, and style classification. We choose four baseline models: Melody2vec (Hirai and Sawada, 2019) is widely used in music understanding tasks, tonnetz (Chuan and Herremans, 2018) and pianoroll (Dong et al., 2018) are classical methods for music representation, PiRhDy (Liang et al., 2020) is a new model that significantly outperforms previous models in all four downstream tasks. Results of melody2vec on accompaniment suggestion task are emitted since it only encodes melody part of music.

MusicBERT	small	base
Number of layers	4	12
Element embedding size	512	768
Hidden size	512	768
FFN inner hidden size	2048	3072
#Attention heads	8	12
Pre-training dataset	LMD	MMD

Table 4: The model configurations of MusicBERT.

batch size of 256 sequences, each has a maximum length of 1024 tokens. We use Adam (Kingma and Ba, 2014) optimizer with β_1=0.9, β_2=0.98, ϵ=1e-6 and L_2 weight decay of 0.01. The learning rate is warmed up over the first 25,000 steps to a peak value of 5e-4 and then linearly decayed. Dropout value on all layers and attention weights are set to 0.1.

4.2 Fine-tuning MusicBERT

We fine-tune MusicBERT on four downstream tasks: two phrase-level tasks (i.e., melody completion and accompaniment suggestion) and two song-level tasks (i.e., genre and style classification). For the two phrase-level tasks, the learning rate is warmed up over the first 50,000 steps to a peak value of 5e-5 and then linearly decayed until reaching 250,000 total updates. For the two song-level tasks, the learning rate is warmed up over the first 4,000 steps to a peak value of 5e-5 and then linearly decayed until reaching 20,000 total updates. The batch size is set to 64 sequences for both tasks. Other settings are the same as pre-training. We compare MusicBERT with previous works on symbolic music understanding, including PiRhDy (Liang et al., 2020) and melody2vec (Hirai and Sawada, 2019).

4.2.1 Melody Completion

Melody completion (Liang et al., 2020) is to find the most matched consecutive phrase in a given set of candidates for a given melodic phrase. There are 1,793,760 data pairs in the training set and 198,665 data groups in the test set in this task (Liang et al., 2020). Each training data pair consists of one positive sample and one negative sample, while each test data group consists of 1 positive sample and 49 negative samples. We use mean average precision (MAP) and HITS@k (k=1, 5, 10, 25, indicating the rate of correctly chosen phrase in the top k candidates) as evaluation metrics, making comparisons with PiRhDy (Liang et al., 2020), melody2vec (Hirai and Sawada, 2019), pianoroll (Dong et al., 2018), tonnetz (Chuan and Herremans, 2018). As shown in Table 3, MusicBERT$_{\text{small}}$ outperforms all previous works on the same pre-training dataset LMD, indicating the advantage of MusicBERT on learning representations from melodic context. MusicBERT$_{\text{base}}$ with a larger model and pre-training corpus can further achieve better results, showing the effectiveness of large-scale pre-training.

4.2.2 Accompaniment Suggestion

Accompaniment suggestion (Liang et al., 2020) is to find the most related accompaniment phrase in a given set of harmonic phrase candidates for a given melodic phrase. There are 7,900,585 data pairs in the training set, each consisting of one positive sample and one negative sample, and 202,816 data groups in the test set (Liang et al., 2020). Each group in the test set consists of N positive samples and (50-N) negative samples when there are N accompaniment tracks in the MIDI file of that sample. We use mean average precision (MAP) and HITS@k as metrics, making comparisons with PiRhDy (Liang et al., 2020), pianoroll (Dong et al., 2018), tonnetz (Chuan and Herremans, 2018). As shown in Table 3, MusicBERT models perform much better than previous works, indicating the advantages of MusicBERT in understanding harmonic context.

4.2.3 Genre and Style Classification

Genre classification and style classification (Ferraro and Lemström, 2018) are multi-label classification tasks. Following Ferraro and Lemström (2018), we use the TOP-MAGD dataset for genre classification and the MASD dataset for style classification. TOP-MAGD contains 22,535 annotated files of 13 genres, and MASD contains 17,785 files of 25 styles. We evaluate MusicBERT on TOP-MAGD and MASD using 5-fold cross-validation and use the F1-micro score as the metric (Liang et al., 2020; Ferraro and Lemström, 2018; Oramas et al., 2017). Due to the limitation of computational resources, for songs with more than 1,000 octuple tokens, we randomly crop segments with 1,000 tokens. On average, the selected segment covers more than 1/4 of a music song according to Table 1, which is enough to capture sufficient information for identifying genres and styles. As shown in Table 3, MusicBERT models significantly outperform previous works, indicating that MusicBERT can perform well on song-level tasks.

4.3 Method Analysis

In this subsection, we analyze the effectiveness of each design in MusicBERT, including OctupleMIDI encoding, bar-level masking strategy, and the pre-training itself. We conduct experiments on MusicBERT$_{\text{small}}$ with a maximum sequence length of 250 due to the huge training cost of MusicBERT$_{\text{base}}$. For simplicity, we treat melody completion and accompaniment suggestion as binary classification tasks: classifying matched and not matched melody or accompaniment pairs, instead of ranking a group of pairs by predicted match score. We use the accuracy percentage score as the metric for these two binary classification tasks.

Effectiveness of OctupleMIDI We compare our proposed OctupleMIDI encoding with REMI (Huang and Yang, 2020) and CP (Hsiao et al., 2021) by training MusicBERT$_{\text{small}}$ models with each encoding respectively and evaluate on downstream tasks. As shown in Table 5, for song-level tasks (i.e., genre and style classification), OctupleMIDI significantly outperforms REMI and CP based encoding, since the model can learn from a larger proportion of a music song with the compact OctupleMIDI encoding, given all encoding methods use the same length of sequence for pre-training. For phrase-level tasks (melody completion and accompaniment suggestion), the input sequence length is usually less than the truncate threshold. Thus, benefiting from the short representation, OctupleMIDI significantly reduces the computational complexity of the Transformer encoder, which is only 1/16 of that with REMI-like encoding and 1/4 of that with CP-like encoding. Moreover, according to Table 5, OctupleMIDI performs better than the other two encoding methods on phrase-level tasks.

Encoding	Melody	Accom.	Genre	Style
CP-like	95.7	87.2	0.719	0.510
REMI-like	92.0	86.5	0.689	0.487
OctupleMIDI	**96.7**	**87.9**	**0.730**	**0.534**

Table 5: Results of different encoding methods. "Accom." represents accompaniment suggestion task.

Effectiveness of Bar-Level Masking We compare our proposed bar-level masking strategy with two other strategies: 1) Octuple masking, as mentioned in Sec. 3.3 and Fig. 2a; 2) Random masking, which randomly masks the elements in the octuple token similar to the masked language model in BERT. We use the same masking ratio for all these strategies. As shown in Table 6, our proposed bar-level masking can effectively boost results on downstream tasks.

Mask	Melody	Accom.	Genre	Style
Random	96.3	87.8	0.708	0.533
Octuple	96.0	87.3	0.722	0.530
Bar	**96.7**	**87.9**	**0.730**	**0.534**

Table 6: Results of different masking strategies.

Effectiveness of Pre-training To show the advantage of pre-training in MusicBERT, we compare the performance of MusicBERT$_{\text{small}}$ with and without pre-training. As shown in Table 7, pre-training achieves much better scores on the four downstream tasks, demonstrating the critical role of pre-training on symbolic music understanding.

Model	Melody	Accom.	Genre	Style
No pre-train	92.4	76.9	0.662	0.395
MusicBERT	**96.7**	**87.9**	**0.730**	**0.534**

Table 7: Results with and without pre-training.

5 Conclusion

In this paper, we developed MusicBERT, a large-scale pre-trained model for symbolic music understanding. Instead of simply adopting the pre-training methods from NLP to symbolic music, we handle the distinctive challenges in music pre-training with several careful designs in MusicBERT, including the efficient and universal OctupleMIDI encoding, the effective bar-level masking strategy, and the large-scale symbolic music corpus with more than 1 million music songs. MusicBERT achieves state-of-the-art performance on all of the four evaluated symbolic music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification. Method analyses also verify the effectiveness of each design in MusicBERT. For future work, we will apply MusicBERT on other music understanding tasks such as chord recognition and structure analysis to boost the performance.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Sumu Zhao. 2018. Symbolic music genre transfer with cyclegan. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 786–793. IEEE.

Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. 2008. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696.

Ching-Hua Chuan, Kat Agres, and Dorien Herremans. 2020. From context to concept: exploring semantic relationships in music with word2vec. *Neural Computing and Applications*, 32(4):1023–1036.

Ching-Hua Chuan and Dorien Herremans. 2018. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang. 2018. Pypianoroll: Open source python package for handling multitrack pianoroll. *Proc. ISMIR. Late-breaking paper;[Online] https://github.com/salu133445/pypianoroll*.

Andres Ferraro and Kjell Lemström. 2018. On large-scale genre classification in symbolically encoded music by automatic identification of repeating patterns. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, pages 34–37.

Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. 2019. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*.

Dorien Herremans and Ching-Hua Chuan. 2017. Modeling musical context with word2vec. *arXiv preprint arXiv:1706.09088*.

Tatsunori Hirai and Shun Sawada. 2019. Melody2vec: Distributed representations of melodic phrases based on melody segmentation. *Journal of Information Processing*, 27:278–286.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2021. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. *arXiv preprint arXiv:2101.02402*.

Cheng-Zhi Anna Huang, David Duvenaud, and Krzysztof Z Gajos. 2016. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 241–250.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, AM Dai, MD Hoffman, and D Eck. 2018. Music transformer: Generating music with long-term structure (2018). *arXiv preprint arXiv:1809.04281*.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1180–1188.

Ray Jackendoff. 2009. Parallels and nonparallels between language and music. *Music perception*, 26(3):195–204.

Shulei Ji, Jing Luo, and Xinyu Yang. 2020. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *arXiv preprint arXiv:2011.06801*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Qiuqiang Kong, Bochen Li, Jitong Chen, and Yuxuan Wang. 2020a. Giantmidi-piano: A large-scale midi dataset for classical piano music. *arXiv preprint arXiv:2010.07061*.

Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. 2020b. High-resolution piano transcription with pedals by regressing onsets and offsets times. *arXiv preprint arXiv:2010.01815*.

Hongru Liang, Wenqiang Lei, Paul Yaozhu Chan, Zhenglu Yang, Maosong Sun, and Tat-Seng Chua. 2020. Pirhdy: Learning pitch-, rhythm-, and dynamics-aware embeddings for symbolic music. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 574–582.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Sephora Madjiheurem, Lizhen Qu, and Christian Walder. 2016. Chord2vec: Learning musical chord embeddings. In *Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems (NIPS2016), Barcelona, Spain*.

Erin McMullen and Jenny R Saffran. 2004. Music and language: A developmental comparison. *Music Perception*, 21(3):289–311.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. 2020. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967.

Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. 2017. Multi-label music genre classification from audio, text, and images using deep features. *arXiv preprint arXiv:1707.04916*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Ph.D. thesis, Columbia University.

Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1198–1206.

Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. 2020. Songmass: Automatic song writing with pre-training and alignment constraint. *arXiv preprint arXiv:2012.05168*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Evaluating the Efficacy of Summarization Evaluation across Languages

Fajri Koto **Jey Han Lau** **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne
`ffajri@student.unimelb.edu.au, jeyhan.lau@gmail.com, tbaldwin@unimelb.edu.au`

Abstract

While automatic summarization evaluation methods developed for English are routinely applied to other languages, this is the first attempt to systematically quantify their panlinguistic efficacy. We take a summarization corpus for eight different languages, and manually annotate generated summaries for focus (precision) and coverage (recall). Based on this, we evaluate 19 summarization evaluation metrics, and find that using multilingual BERT within BERTScore performs well across all languages, at a level above that for English.

1 Introduction

Although manual evaluation (Nenkova and Passonneau, 2004; Hardy et al., 2019) of text summarization is more reliable and interpretable, most research on text summarization employs automatic evaluations such as ROUGE (Lin, 2004), METEOR (Lavie and Agarwal, 2007), MoverScore (Zhao et al., 2019), and BERTScore (Zhang et al., 2020b) because they are time- and cost-efficient.

In proposing these metrics, the authors measured correlation with human judgments based on English datasets that are not representative of modern summarization systems. For instance, Lin (2004) use DUC[1] 2001–2003 for ROUGE (meaning summaries were generated with largely outdated extractive summarization systems); Zhao et al. (2019) use the TAC[2] dataset for MoverScore (again, featuring summaries from largely defunct systems; see Peyrard (2019) and Rankel et al. (2013)); and Zhang et al. (2020b) developed BERTScore based on a machine translation corpus (WMT). In contemporaneous work, Bhandari et al. (2020) address this issue by annotating English CNN/DailyMail summaries produced by recent summarization models, and found disparities over results from TAC.

Gold summary : Info-A; Info-B; Info-C	
System summary:	
Good *focus,* and Good *coverage*	: Info-A; Info-B; Info-C
Good *focus,* and Bad *coverage*	: Info-A; Info-A
Bad *focus,* and Good *coverage*	: Info-A; Info-B; Info-C; Info-D; Info-E
Bad *focus,* and Bad *coverage*	: Info-D; Info-E; Info-F

Figure 1: Illustration of focus and coverage.

Equally troublingly, ROUGE has become the default summarization evaluation metric for languages other than English (Hu et al., 2015; Scialom et al., 2020; Ladhak et al., 2020; Koto et al., 2020b), despite there being no systematic validation of its efficacy across other languages. The questions we ask in this study, therefore, are twofold: (1) *How well do existing automatic metrics perform over languages other than English?* and (2) *What automatic metric works best across different languages?*

In this paper, we examine content-based summarization evaluation from the aspects of precision and recall, in the form of *focus* and *coverage* to compare system-generated summaries to ground-truth summaries (see Figure 1). As advocated by Koto et al. (2020a), focus and coverage are more interpretable and fine grained than the harmonic mean (F1 score) of ROUGE. This is also in line with the review of Hardy et al. (2019) on linguistic properties that have been manually evaluated in recent summarization research, who found precision and recall to be commonly used to complement ROUGE F1.

While it may seem more natural and reliable to evaluate focus and coverage based on the source document than the ground-truth summary, we use the ground-truth summary in this research for the following reasons. First, historically, validation of automatic evaluation metrics for summarization has been based primarily on ground-truth summaries (not source documents). Second, ROUGE (Lin,

---

[1] https://duc.nist.gov/data.html

[2] https://tac.nist.gov/data/

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 801–812
August 1–6, 2021. ©2021 Association for Computational Linguistics

2004) was initially motivated and assessed based on coverage over the DUC datasets[3] (Lin and Hovy, 2002) using annotations based on reference summaries (not source documents). Third, although it is certainly possible to generate different summaries for the same source document, we argue that the variance in content is actually not that great, especially for single-document summarization. Lastly, basing human evaluation (of focus and coverage) on the source article leads to more complicated annotation schemes, and has been shown to yield poor annotations (Nenkova and Passonneau, 2004; Fabbri et al., 2020).

In summary, this paper makes three contributions: (1) we carry out the first systematic attempt to quantify the efficacy of automatic summarization metrics over 8 linguistically-diverse languages, namely English (EN), Indonesian (ID), French (FR), Turkish (TR), Mandarin Chinese (ZH), Russian (RU), German (DE), and Spanish (ES); (2) we evaluate an extensive range of traditional and model-based metrics, and find BERTScore to be the best metric for evaluating both focus and coverage; and (3) we release a manually-annotated multilingual resource for summarization evaluation comprising 4,320 annotations. Data and code used in this paper is available at: https://github.com/fajri91/Multi_SummEval.

2 Related Work

As with much of NLP, research on automatic summarization metrics has been highly English-centric. Graham (2015) comprehensively evaluated 192 ROUGE variations based on the DUC-2004 (English) dataset. Bhandari et al. (2020) released a new (English) evaluation dataset by annotating CNN/DailyMail using simplified Pyramid (Nenkova and Passonneau, 2004). First, semantic content units (SCUs) were manually extracted from the reference, and crowd-workers were then asked to count the number of SCUs in the system summary. Their annotation procedure does not specifically consider focus, but is closely related to the coverage aspect of our work. Similarly, Fabbri et al. (2020) annotated the (English) CNN/DailyMail dataset for the four aspects of coherence, consistency, fluency, and relevance. While their work does not specifically study focus and coverage, relevance in their work can be interpreted as the harmonic mean of focus and coverage.

There is little work on summarization evaluation for languages other than English, and what work exists is primarily based on summaries generated by unsupervised extractive models dating back more than a decade, for a small handful of languages. Two years prior to ROUGE, Saggion et al. (2002) proposed a summarization metric using similarity measures for English and Chinese, based on cosine similarity, unit overlap, and the longest common subsequence ("LCS") between reference and system summaries. In other work, Saggion et al. (2010) investigated coverage, responsiveness, and pyramids for several extractive models in English, French, and Spanish.

To the best of our knowledge, we are the first to systemically quantify the panlinguistic efficacy of evaluation metrics for modern summarization systems.

3 Evaluation Metrics

We assess a total of 19 different evaluation metrics that are commonly used in summarization research (noting that lesser-used metrics such as FRESA (Saggion et al., 2010) and RESA (Cohan and Goharian, 2016) are omitted from this study).

ROUGE (Lin, 2004) measures the lexical overlap between the system and reference summary; based on the findings of Graham (2015), we consider 7 variants in this paper: ROUGE-1 (unigram), ROUGE-2 (bigram), ROUGE-3 (trigram), ROUGE-L (LCS), ROUGE-S (skip-bigram), ROUGE-SU (skip-bigram plus unigram), and ROUGE-W (weighted LCS).[4]

METEOR (Lavie and Agarwal, 2007) performs word-to-word matching based on word-alignment, and was originally developed for MT but has recently been used for summarization evaluation (See et al., 2017; Chen and Bansal, 2018; Falke and Gurevych, 2019; Amplayo and Lapata, 2020).[5]

BLEU (Papineni et al., 2002) is a precision-based metric originally developed for MT, which measures the n-gram match between the reference and system summary. Based on the findings of Graham (2015), we use BLEU-4 according to the SacreBLEU implementation (Post, 2018).[6]

MoverScore (Zhao et al., 2019) measures the Euclidean distance between two contextualized BERT representations, and relies on soft align-

---

[3] DUC 2001, 2002, 2003

[4] https://github.com/bheinzerling/pyrouge

[5] http://www.cs.cmu.edu/~alavie/METEOR/

[6] https://github.com/mjpost/sacrebleu

ments of words learned by solving an optimisation problem.[7] We adapt use the default configuration (n-gram=1) over 5 different pre-trained models, as detailed below. Note that MoverScore is symmetric (i.e. MoverScore(x, y) = MoverScore(y, x)), and as such is not designed to separately evaluate precision and recall.

BERTScore (Zhang et al., 2020b) computes the similarity between BERT token embeddings of system and reference summaries based on soft overlap, in the form of precision, recall, and F1 scores.[8] Zhang et al. (2020b) found that layer selection (i.e. which layer to source the token embeddings from) is critical to performance. Since layer selection in the original paper was based on MT datasets, we perform our own layer selection using a similar methodology as the authors, specifically considering precision and recall for focus and coverage, respectively.

For both MoverScore and BERTScore, we experiment with two classes of BERT-style model: (1) multilingual models, in the form of cased and uncased multilingual BERT (Devlin et al., 2019), and base and large XLM-R (Conneau et al., 2020), for a total of 4 models;[9] and (2) a monolingually-trained BERT for the given language, as listed in the Appendix. While we expect monolingual BERT models to perform better, we also focus on multilingual models, both to confirm whether this is the case, and to be able to draw findings for languages without monolingual models.

4 Experimental Setup

For each language, we sample 135 documents from the test set of a pre-existing (single-document) summarization dataset: (1) CNN/DailyMail (English: Hermann et al. (2015)); (2) Liputan6 (Indonesian: Koto et al. (2020b)); (3) LCSTS (Chinese: Hu et al. (2015)); and (4) MLSUM (French, Turkish, Russian, German, Spanish: Scialom et al. (2020)). We source summaries based on two popular models: pointer generator network (See et al., 2017) and BERT (Liu and Lapata, 2019; Dong et al., 2019),[10] and have 3 annotators annotate focus and coverage for each reference–system summary pair.[11] The motivation for using BERT-based systems is that our study focuses on non-English summarization, where BERT-based models dominate.[12] The total number of resulting annotations is: 8 languages × 135 documents × 2 models × 2 criteria (= focus and coverage) × 3 annotators = 12,960.

Lang	**Quality (%)**	**Pearson correlation (r)**		
		Agreement		**Focus–**
		Focus	**Coverage**	**Coverage**
EN	90	0.47	0.46	0.58
ID	97	0.64	0.63	0.80
FR	98	0.63	0.65	0.71
TR	97	0.74	0.79	0.74
ZH	92	0.61	0.60	0.78
RU	98	0.60	0.64	0.78
DE	90	0.78	0.83	0.89
ES	95	0.60	0.61	0.76

Table 1: Analysis of the annotations for each language, in terms of: (1) average quality control score of approved HITs (%); (2) one-vs-rest human agreement (r); and (3) correlation (r) between focus and coverage.

For annotation, we used Amazon Mechanical Turk[13] with the customized Direct Assessment ("DA") method (Graham et al., 2015; Graham et al., 2017), which has become the de facto for MT evaluation in WMT. For each HIT (100 samples), DA incorporates 10 pre-annotated samples for quality control. Crowd-sourced workers are given two texts and asked the question (in the local language): *How much information contained in the second text can also be found in the first text?* We combine focus and coverage annotation into 1 task, as the only thing that differentiates them is the ordering of the system and reference summaries, which is opaque to the annotators.[14] Workers responded by scoring based via a slider button (continuous scale of 1–100).[15]

For each HIT, we create 10 samples for quality control: 5 samples are random pairs (should be

---

[7] https://github.com/AIPHES/emnlp19-moverscore

[8] https://github.com/Tiiiger/bert_score

[9] Note that both multilingual BERT and XLM were explicitly trained over all eight target languages used in this paper.

[10] English, Indonesian and Chinese summaries were generated with the Liu and Lapata (2019) model, and the Dong et al. (2019) model was used for the MLSUM-based languages.

[11] Summaries for all datasets except LCSTS were sourced from the authors of the dataset. For LCSTS, we trained the two models ourselves based on the training data.

[12] BERT-based summaries are representative of transformer-based model, and the ROUGE score gap over state-of-the-art models (Zhang et al., 2020a) for English is only ∼2 points.

[13] https://www.mturk.com

[14] For focus, the first text is the reference and the second text the system summary; for coverage, the order is reversed.

[15] See Appendix for the MTurk annotation interface for each language.

	Focus									**Coverage**								
	EN	**ID**	**FR**	**TR**	**ZH**	**RU**	**DE**	**ES**	**Avg**	**EN**	**ID**	**FR**	**TR**	**ZH**	**RU**	**DE**	**ES**	**Avg**
Traditional Metrics																		
ROUGE-1	0.61	0.69	0.68	0.81	0.80	0.47	0.88	0.53	0.68	0.62	0.72	0.67	0.83	0.79	0.58	0.89	0.67	0.72
ROUGE-2	0.57	0.63	0.67	0.80	0.76	0.48	0.87	0.61	0.67	0.56	0.66	0.71	0.79	0.75	0.59	0.89	0.67	0.70
ROUGE-3	0.46	0.53	0.59	0.76	0.67	0.31	0.85	0.54	0.59	0.48	0.57	0.63	0.74	0.66	0.46	0.88	0.58	0.62
ROUGE-L	0.60	0.69	0.68	0.81	0.79	0.46	0.87	0.54	0.68	0.61	0.72	0.67	0.83	0.79	0.59	0.89	0.67	0.72
ROUGE-S	0.59	0.65	0.60	0.78	0.70	0.46	0.85	0.51	0.64	0.60	0.69	0.67	0.78	0.73	0.53	0.89	0.64	0.69
ROUGE-SU	0.59	0.66	0.61	0.78	0.72	0.43	0.85	0.50	0.64	0.60	0.70	0.68	0.78	0.75	0.56	0.89	0.65	0.70
ROUGE-W.12	0.60	0.67	0.67	0.81	0.78	0.44	0.87	0.53	0.67	0.58	0.69	0.67	0.81	0.78	0.59	0.89	0.66	0.71
METEOR	0.47	0.67	0.64	0.74	0.81	0.55	0.83	0.60	0.66	0.63	0.71	0.64	0.80	0.78	0.58	0.89	0.69	0.72
BLEU-4	0.46	0.56	0.64	0.70	0.70	0.39	0.85	0.50	0.60	0.48	0.58	0.59	0.67	0.69	0.31	0.85	0.54	0.59
MoverScore																		
mono-BERT	0.58	0.65	0.71	0.82	0.77	0.49	0.89	0.59	0.69	0.59	0.62	0.67	0.78	0.77	0.41	0.88	0.61	0.67
mBERT (cased)	0.54	0.68	0.77	0.79	0.76	**0.60**	0.88	0.63	0.70	0.52	0.69	0.72	0.75	0.75	0.49	0.85	0.68	0.68
mBERT (uncased)	0.59	0.69	**0.78**	0.81	0.76	**0.60**	0.89	**0.67**	**0.72**	0.59	0.69	0.75	0.77	0.75	0.50	0.86	0.70	0.70
XLM (base)	0.53	0.64	0.69	0.80	0.71	0.35	0.87	0.56	0.64	0.58	0.62	0.63	0.74	0.69	0.22	0.85	0.64	0.62
XLM (large)	0.51	0.58	0.68	0.79	0.57	0.33	0.87	0.53	0.61	0.55	0.62	0.59	0.72	0.58	0.21	0.84	0.56	0.58
BERTScore																		
mono-BERT	**0.62**	**0.71**	0.73	**0.83**	**0.82**	0.51	**0.91**	**0.67**	**0.72**	0.66	**0.74**	**0.77**	**0.88**	**0.80**	0.65	**0.92**	**0.74**	**0.77**
mBERT (cased)	0.56	**0.71**	0.73	**0.83**	0.78	0.56	0.90	0.59	0.71	**0.67**	0.73	0.70	0.87	0.79	**0.72**	0.90	0.71	0.76
mBERT (uncased)	0.61	**0.71**	0.72	**0.83**	0.79	0.55	0.90	0.62	**0.72**	0.64	**0.74**	0.72	0.87	0.79	0.70	0.90	0.71	0.76
XLM (base)	0.59	0.65	0.67	**0.83**	0.79	0.34	0.89	0.58	0.67	0.64	0.71	0.66	0.86	0.73	0.67	0.90	0.70	0.74
XLM (large)	0.60	0.66	0.68	**0.83**	0.79	0.42	0.90	0.60	0.69	0.65	0.70	0.69	0.86	0.74	0.66	0.90	0.70	0.74
Human performance	0.47	0.64	0.63	0.74	0.61	0.60	0.78	0.60	0.63	0.46	0.63	0.65	0.79	0.60	0.64	0.83	0.61	0.65

Table 2: Pearson correlation (r) between automatic metrics and human judgments (for Pointer Generator and BERT models combined). We compute the precision and recall of ROUGE and BERTScore for focus and coverage, respectively. BERTScore uses the optimized layer, and other metrics are computed using the default configuration of the original implementation.

Model	**Universal layer**	
	Focus	**Coverage**
mBERT (cased)	12	5
mBERT (uncased)	12	6
XLM-R (base)	4	4
XLM-R (large)	10	9

Table 3: Recommended layers for multilingual models.

scored 0) and the remaining 5 samples are repetitions of the same summary with minor edits (should be scored 100). For each language, we asked a native speaker to translate all instructions and the annotation interface. For a single HIT, we paid USD$13, and set the HIT approval rate to 95%. For HITs to be included in the annotated data, a quality control score of at least 7 out of 10 needed to be achieved. HITs below this threshold were re-run (ensuring they were not completed by a worker who had already completed that HIT), until three above-threshold annotations were obtained.[16] For each language, the HIT approval rate is set to 95% (with the number of HITs approved varying across languages). The annotation for English was restricted to US-based workers, and for other languages except Chinese was based on countries where the language is an official language.[17]

To obtain focus and coverage values, we follow standard practice in DA in z-scoring the scores from each annotator, and then averaging.

5 Results

5.1 Annotation Results

In Table 1, we present the results of the human annotation. We first normalize the ratings from each HIT into a z-score, and one-vs-rest Pearson correlation (excluding quality control items) to provide an estimate of human agreement/performance.[18] For all languages, we observe that the average quality and human agreement is moderately high. However, the agreement does vary, and it affects the interpretation of the correlation scores when we assess the automatic metrics later. Note also that we get the lowest score for English, meaning the results for non-English languages are actually more

---

[16] We approved all HITs with at least 30 minutes working time and a minimum quality control score of 5, irrespective of whether they passed the higher quality-control threshold required for the ground truth.

[17] In MTurk, we did not set a specific location for Chinese because we found there are no workers in China.

[18] We follow Lau et al. (2020) in computing one-vs-rest correlation: we randomly isolate a worker's score (for each sample) and compare it against the mean score of the rest using Pearson's r, and repeat this for 1000 trials to get the mean correlation.

robust.[19]

Although focus and coverage are positively correlated in Table 1, the distribution of scores varies quite a bit between languages: English annotation variance is higher than the other languages, and has the lowest correlation between focus and coverage ($r = 0.57$); for French, Russian, and Spanish, summaries generally have low focus and coverage (for more details, see scatterplots of focus-coverage in Figure 2 of the Appendix).

5.2 Correlation with Automatic Evaluation

In Table 2 we present the Pearson correlation between the human annotations and various automatic metrics, broken down across language and focus vs. coverage, and (naively) aggregated across languages in the form of the average correlation. We also include the one-vs-rest annotator correlation (Section 5.1) in the last row, as it can be interpreted as the average performance of a single annotator. Recognizing the sensitivity of Pearson's correlation to outliers (Mathur et al., 2020), we manually examined the distribution of scores for all language–system combinations for outliers (and present all scatterplots in Figure 2 of the Appendix).

The general pattern is consistent across languages: BERTScore performs better than other metrics in terms of both focus and coverage. This finding is consistent with that of Fabbri et al. (2020) wrt expert annotations of relevance (interpreted as the harmonic mean of our focus and coverage). ROUGE-1 and ROUGE-L are overall the best versions of ROUGE, while BLEU-4 performs the worst. For coverage, METEOR tends to be competitive with ROUGE-1, especially for EN, FR, DE, and ES, in large part because these languages are supported by the METEOR lemmatization package.

For some pre-trained models, MoverScore is competitive with BERTScore, although the average correlation is lower, especially for coverage.

We perform layer selection for BERTScore by selecting the layer that produces the highest correlation. For monolingual BERT the selection is based on the average correlation across the two summarization models, while for the multilingual models it is based on overall result across the 8 languages × 2 models. Table 3 details the recommended layer for computing BERTScore for each of the multilingual models.[20]

We observe that BERTScore with monolingual BERT performs the best, at an average of 0.72 and 0.77 for focus and coverage, resp., but only marginally above the best of the multilingual models, namely mBERT uncased (0.72 and 0.76, resp.). Given that layer selection here was performed universally across all languages (to ensure generalizability to other languages), our overall recommendation for the best metric to use is BERTScore with mBERT uncased.

When we compare the metric results to the one-vs-rest single-annotator performance from Table 1, we see a positive correspondence between the relative scores for annotator agreement and metric performance, which we suspect is largely an artefact of data quality (i.e. the metrics are assessed to perform better for languages with high agreement because the quality of the ground-truth is higher), but further research is required to confirm this. Generally the best metrics tend to outperform single-annotator performance substantially (>0.10), suggesting these metrics are more reliable than a single annotator.

6 Conclusion

In this work, we developed a novel dataset for assessing automatic evaluation metrics for focus and coverage across a broad range of languages and datasets. We found that BERTScore is the best metric for the vast majority of languages, and advocate that this metric be used for summarization evaluation across different languages in the future, supplanting ROUGE.

Acknowledgments

We are grateful to the anonymous reviewers for their helpful feedback and suggestions. The first author is supported by the Australia Awards Scholarship (AAS), funded by the Department of Foreign Affairs and Trade (DFAT), Australia. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at The University of Melbourne. This facility was established with the assistance of LIEF Grant LE170100200.

---

[19] The relative quality for different languages largely coincides with the findings of Pavlick et al. (2014).

[20] Recommended layers for monolingual BERT are detailed in the Appendix.

References

Reinald Kim Amplayo and Mirella Lapata. 2020. Unsupervised opinion summarization with noising and denoising. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1934–1945, Online. Association for Computational Linguistics.

Manik Bhandari, Pranav Narayan Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. 2020. Re-evaluating evaluation in text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9347–9359, Online. Association for Computational Linguistics.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Arman Cohan and Nazli Goharian. 2016. Revisiting summarization evaluation for scientific articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 806–813, Portorož, Slovenia. European Language Resources Association (ELRA).

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, volume 32, pages 13063–13075.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. Summeval: Re-evaluating summarization evaluation. *arXiv preprint arXiv:2007.12626*.

Tobias Falke and Iryna Gurevych. 2019. Fast concept mention grouping for concept map-based multi-document summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 695–700, Minneapolis, Minnesota. Association for Computational Linguistics.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, and Nitika Mathur. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1183–1191, Denver, Colorado. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):3–30.

Hardy Hardy, Shashi Narayan, and Andreas Vlachos. 2019. HighRES: Highlight-based reference-less evaluation of summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3381–3392, Florence, Italy. Association for Computational Linguistics.

Karl Moritz Hermann, Tom Koisk, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, pages 1693–1701.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. LCSTS: A large scale Chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972, Lisbon, Portugal. Association for Computational Linguistics.

Fajri Koto, Jey Han Lau, and Timothy Baldwin. 2020a. FFCI: A framework for interpretable automatic evaluation of summarization. *arXiv preprint arXiv:2011.13662*.

Fajri Koto, Jey Han Lau, and Timothy Baldwin. 2020b. Liputan6: A large-scale Indonesian dataset for text summarization. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 598–608, Suzhou, China. Association for Computational Linguistics.

Fajri Koto, Afshin Rahimi, Jey Han Lau, and Timothy Baldwin. 2020c. IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 757–770, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yuri Kuratov and Mikhail Arkhipov. 2019. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. 2020. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048, Online. Association for Computational Linguistics.

Jey Han Lau, Carlos Armendariz, Shalom Lappin, Matthew Purver, and Chang Shu. 2020. How furiously can colorless green ideas sleep? sentence acceptability in context. *Transactions of the Association for Computational Linguistics*, 8:296–310.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2002. Manual and automatic evaluation of summaries. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 45–51, Phildadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online. Association for Computational Linguistics.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The language demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics*, 2:79–92.

Maxime Peyrard. 2019. Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–136, Sofia, Bulgaria. Association for Computational Linguistics.

Horacio Saggion, Dragomir Radev, Simone Teufel, and Wai Lam. 2002. Meta-evaluation of summaries in a cross-lingual environment using content-based metrics. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Horacio Saggion, Juan-Manuel Torres-Moreno, Iria da Cunha, Eric SanJuan, and Patricia Velázquez-Morales. 2010. Multilingual summarization evaluation without human models. In *Coling 2010: Posters*, pages 1059–1067, Beijing, China. Coling 2010 Organizing Committee.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020.

MLSUM: The multilingual summarization corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8051–8067, Online. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML 2020: 37th International Conference on Machine Learning*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with bert. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

A Supplementary Materials

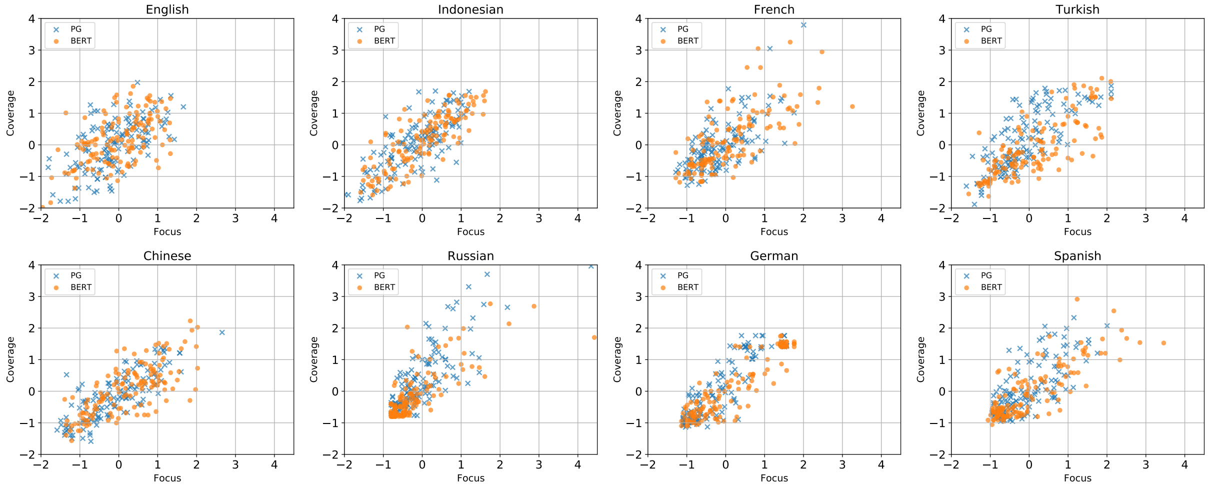

Figure 2: Annotation result (focus vs. coverage) after z-score normalization for each of the 8 languages.

Lang	Model	Recommended layer	
		Focus	Coverage
EN	`bert-base-uncased` (Devlin et al., 2019)	1	2
ID	`indolem/indobert-base-uncased` (Koto et al., 2020c)	2	2
ZH	`bert-base-chinese` (Devlin et al., 2019)	8	9
FR	`camembert-base` (Martin et al., 2020)	10	9
TR	`dbmdz/bert-base-turkish-uncased`	12	4
RU	`DeepPavlov/rubert-base-cased` (Kuratov and Arkhipov, 2019)	4	12
DE	`bert-base-german-dbmdz-uncased`	12	12
ES	`dccuchile/bert-base-spanish-wwm-uncased`	4	4

Table 4: Recommended layers for computing focus and coverage via BERTScore with monolingual model.

Language	ISO	Data	Data Split			Pointer Generator			BERT		
			Train	Dev	Test	R1	R2	RL	R1	R2	RL
English	EN	CNN/DailyMail	287,226	13,368	11,490	39.53	17.28	36.38	42.13	19.60	39.18
Indonesian	ID	Liputan6	193,883	10,972	10,792	36.10	19.19	33.56	41.08	22.85	38.01
Chinese	ZH	LCSTS	2,400,591	8,672	725	32.39	19.92	29.45	38.47	25.45	35.30
French	FR	MLSUM	392,902	16,059	15,828	26.50	9.49	20.30	28.52	11.73	22.51
Turkish	TR	MLSUM	249,277	11,565	12,775	39.77	26.45	36.12	41.28	28.16	37.79
Russian	RU	MLSUM	25,556	750	757	5.39	0.60	4.62	6.01	1.02	5.75
German	DE	MLSUM	220,887	11,394	10,701	36.86	27.06	35.04	44.11	33.99	42.10
Spanish	ES	MLSUM	266,367	10,358	13,920	25.05	7.44	19.53	26.48	9.59	21.69

Table 5: Details of datasets and ROUGE scores of summarization models used in this study. Other than for Chinese, we use summaries provided by the respective authors. For MLSUM, we report slightly different ROUGE-L scores because we use the original ROUGE package.

Metrics	POINTER GENERATOR									BERT								
	EN	ID	FR	TR	ZH	RU	DE	ES	Avg	EN	ID	FR	TR	ZH	RU	DE	ES	Avg
Traditional Metrics																		
ROUGE-1	0.59	0.68	0.54	0.82	0.81	0.52	0.85	0.50	0.66	0.61	0.70	0.73	**0.81**	0.78	0.59	0.89	0.54	0.71
ROUGE-2	0.60	0.59	0.56	0.83	0.78	0.54	0.86	0.61	0.67	0.53	0.65	0.71	0.77	0.73	0.56	0.87	0.61	0.68
ROUGE-3	0.49	0.52	0.49	0.80	0.68	0.39	0.85	0.56	0.60	0.43	0.53	0.63	0.73	0.63	0.29	0.85	0.54	0.58
ROUGE-L	0.59	**0.69**	0.55	0.83	0.81	0.51	0.85	0.50	0.67	0.60	0.68	0.73	0.80	0.75	0.58	0.88	0.57	0.70
ROUGE-S	0.60	0.62	0.48	0.79	0.70	0.55	0.83	0.50	0.63	0.58	0.67	0.64	0.76	0.69	0.59	0.86	0.51	0.66
ROUGE-SU	0.59	0.63	0.50	0.79	0.72	0.55	0.83	0.50	0.64	0.58	0.68	0.66	0.77	0.70	0.60	0.86	0.50	0.67
ROUGE-W.12	0.60	0.67	0.56	0.83	0.81	0.52	0.86	0.49	0.66	0.60	0.66	0.72	0.79	0.74	0.55	0.88	0.57	0.69
METEOR	0.49	0.65	0.51	0.82	0.85	0.52	0.86	0.68	0.67	0.45	0.68	0.70	0.71	0.77	0.52	0.85	0.58	0.66
BLEU-4	0.51	0.57	0.60	0.78	0.75	0.46	0.86	0.59	0.64	0.43	0.54	0.66	0.64	0.65	0.51	0.85	0.46	0.59
MoverScore																		
mono-BERT	0.62	0.67	0.63	**0.88**	0.80	**0.65**	**0.90**	0.60	0.72	0.54	0.63	0.74	0.77	0.73	0.61	0.89	0.58	0.69
mBERT (cased)	0.57	0.62	0.71	0.84	0.79	0.60	0.88	0.71	0.72	0.51	0.71	**0.78**	0.75	0.73	0.68	0.88	0.59	0.70
mBERT (uncased)	0.63	0.65	**0.76**	**0.88**	0.79	0.57	0.89	**0.74**	**0.74**	0.54	0.71	**0.78**	0.76	0.73	0.68	0.88	0.63	0.71
XLM (base)	0.56	0.61	0.60	0.86	0.73	0.32	0.86	0.71	0.66	0.49	0.63	0.71	0.77	0.68	0.52	0.89	0.51	0.65
XLM (large)	0.53	0.57	0.60	0.84	0.62	0.30	0.87	0.63	0.62	0.48	0.58	0.69	0.76	0.52	0.42	0.87	0.46	0.60
BERTScore																		
mono-BERT	0.62	0.68	0.71	0.86	0.82	0.42	**0.90**	0.69	0.71	0.62	0.73	0.72	0.80	**0.81**	**0.71**	**0.91**	**0.66**	**0.75**
mBERT (cased)	0.61	0.67	0.64	0.84	0.77	0.54	0.89	0.69	0.71	0.58	**0.74**	0.76	**0.81**	0.77	**0.71**	**0.91**	0.54	0.73
mBERT (uncased)	**0.64**	0.68	0.67	0.86	0.79	0.48	0.89	0.70	0.72	0.63	0.72	0.73	**0.81**	0.77	0.69	0.90	0.57	0.73
XLM (base)	0.62	0.65	0.59	0.86	0.80	0.39	0.87	0.63	0.68	0.61	0.64	0.69	**0.81**	0.78	0.64	0.90	0.56	0.70
XLM (large)	0.63	0.65	0.64	0.87	0.80	0.35	0.88	0.67	0.68	**0.64**	0.66	0.69	0.80	0.77	0.66	0.90	0.57	0.71

Table 6: Pearson correlation (r) between automatic metrics and human judgments for **focus**. We compute the precision for ROUGE and BERTScore. BERTScore uses the optimized layer, and other metrics are computed by using default configuration of the original implementation.

Metrics	POINTER GENERATOR									BERT								
	EN	ID	FR	TR	ZH	RU	DE	ES	Avg	EN	ID	FR	TR	ZH	RU	DE	ES	Avg
Traditional Metrics																		
ROUGE-1	0.59	0.73	0.66	0.79	**0.82**	0.52	0.90	0.67	0.71	0.64	0.71	0.70	0.86	0.76	0.57	0.89	0.70	0.73
ROUGE-2	0.55	0.65	0.68	0.76	0.78	0.63	0.89	0.64	0.70	0.57	0.66	0.72	0.83	0.72	0.52	0.90	0.73	0.71
ROUGE-3	0.49	0.59	0.58	0.69	0.68	0.50	0.88	0.57	0.62	0.48	0.55	0.66	0.79	0.63	0.41	0.89	0.64	0.63
ROUGE-L	0.58	0.74	0.64	0.79	**0.82**	0.54	0.90	0.66	0.71	0.63	0.70	0.71	0.86	0.77	0.56	0.90	0.72	0.73
ROUGE-S	0.60	0.69	0.63	0.74	0.77	0.52	0.89	0.66	0.69	0.61	0.69	0.69	0.81	0.70	0.51	0.89	0.71	0.70
ROUGE-SU	0.59	0.70	0.64	0.75	0.79	0.52	0.89	0.67	0.69	0.61	0.69	0.70	0.82	0.71	0.57	0.89	0.71	0.71
ROUGE-W.12	0.54	0.71	0.64	0.77	0.81	0.55	0.90	0.65	0.69	0.61	0.68	0.69	0.85	0.75	0.56	0.90	0.68	0.71
METEOR	0.60	0.72	0.65	0.77	0.81	0.55	0.89	0.63	0.70	0.66	0.69	0.69	0.83	0.75	0.59	0.89	0.75	0.73
BLEU-4	0.48	0.61	0.63	0.61	0.70	0.49	0.84	0.50	0.61	0.49	0.56	0.59	0.75	0.67	0.54	0.87	0.59	0.63
MoverScore																		
mono-BERT	0.57	0.65	0.63	0.73	0.79	0.68	0.86	0.55	0.68	0.61	0.60	0.69	0.86	0.75	0.66	0.91	0.68	0.72
mBERT (cased)	0.53	0.67	0.68	0.71	0.77	0.60	0.82	0.63	0.68	0.53	0.71	0.75	0.82	0.73	0.63	0.89	0.74	0.73
mBERT (uncased)	0.58	0.68	0.74	0.72	0.76	0.58	0.84	0.64	0.69	0.59	0.70	0.76	0.85	0.73	0.65	0.90	0.76	0.74
XLM (base)	0.56	0.61	0.52	0.68	0.71	0.31	0.82	0.62	0.60	0.58	0.64	0.68	0.83	0.65	0.52	0.90	0.68	0.68
XLM (large)	0.52	0.62	0.50	0.66	0.59	0.31	0.82	0.49	0.56	0.57	0.61	0.63	0.82	0.56	0.48	0.88	0.63	0.65
BERTScore																		
mono-BERT	0.63	0.74	0.76	**0.87**	0.81	**0.72**	**0.92**	**0.73**	**0.77**	0.67	**0.74**	**0.78**	**0.89**	**0.78**	0.63	**0.92**	**0.78**	**0.77**
mBERT (cased)	**0.67**	**0.75**	0.67	0.85	**0.82**	0.70	0.91	0.72	0.76	**0.68**	0.71	0.74	**0.89**	0.76	**0.69**	0.90	0.72	0.76
mBERT (uncased)	0.63	**0.75**	0.70	0.85	0.81	0.67	0.91	0.71	0.76	0.64	0.73	0.76	**0.89**	0.77	0.68	0.90	0.73	0.76
XLM (base)	0.66	0.72	0.68	0.84	0.77	0.63	0.91	0.70	0.74	0.64	0.70	0.67	0.88	0.69	0.67	0.89	0.71	0.73
XLM (large)	0.66	0.70	0.68	0.84	0.77	0.59	0.91	0.70	0.73	0.66	0.69	0.70	0.88	0.70	**0.69**	0.90	0.72	0.74

Table 7: Pearson correlation (r) between automatic metrics and human judgments for **coverage**. We compute the recall for ROUGE and BERTScore. BERTScore uses the optimized layer, and other metrics are computed by using default configuration of the original implementation.

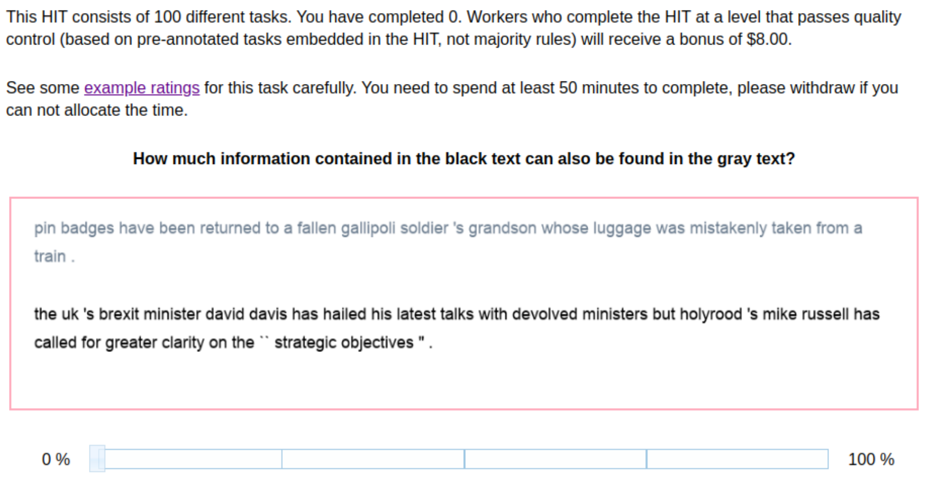
This HIT consists of 100 different tasks. You have completed 0. Workers who complete the HIT at a level that passes quality control (based on pre-annotated tasks embedded in the HIT, not majority rules) will receive a bonus of $8.00.

See some example ratings for this task carefully. You need to spend at least 50 minutes to complete, please withdraw if you can not allocate the time.

How much information contained in the black text can also be found in the gray text?

pin badges have been returned to a fallen gallipoli soldier 's grandson whose luggage was mistakenly taken from a train .

the uk 's brexit minister david davis has hailed his latest talks with devolved ministers but holyrood 's mike russell has called for greater clarity on the `` strategic objectives " .

Figure 3: MTurk annotation interface for **English**.

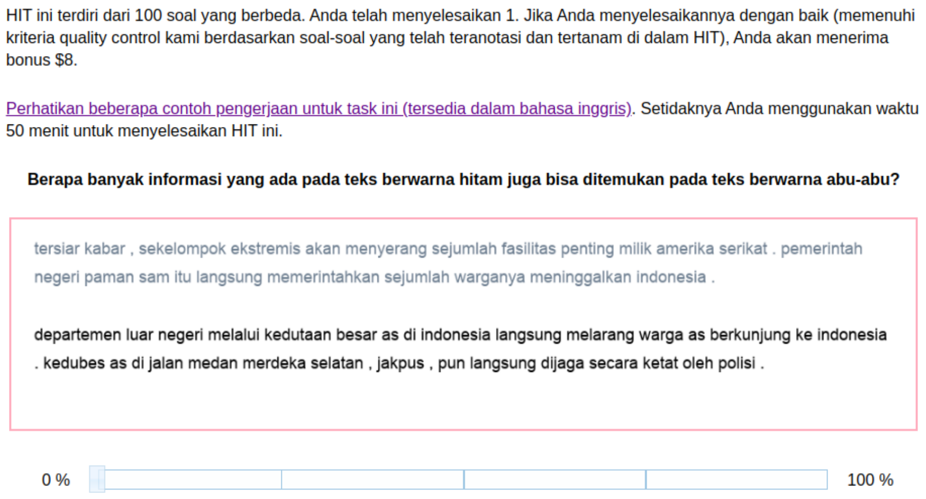
HIT ini terdiri dari 100 soal yang berbeda. Anda telah menyelesaikan 1. Jika Anda menyelesaikannya dengan baik (memenuhi kriteria quality control kami berdasarkan soal-soal yang telah teranotasi dan tertanam di dalam HIT), Anda akan menerima bonus $8.

Perhatikan beberapa contoh pengerjaan untuk task ini (tersedia dalam bahasa inggris). Setidaknya Anda menggunakan waktu 50 menit untuk menyelesaikan HIT ini.

Berapa banyak informasi yang ada pada teks berwarna hitam juga bisa ditemukan pada teks berwarna abu-abu?

tersiar kabar , sekelompok ekstremis akan menyerang sejumlah fasilitas penting milik amerika serikat . pemerintah negeri paman sam itu langsung memerintahkan sejumlah warganya meninggalkan indonesia .

departemen luar negeri melalui kedutaan besar as di indonesia langsung melarang warga as berkunjung ke indonesia . kedubes as di jalan medan merdeka selatan , jakpus , pun langsung dijaga secara ketat oleh polisi .

Figure 4: MTurk annotation interface for **Indonesian**.

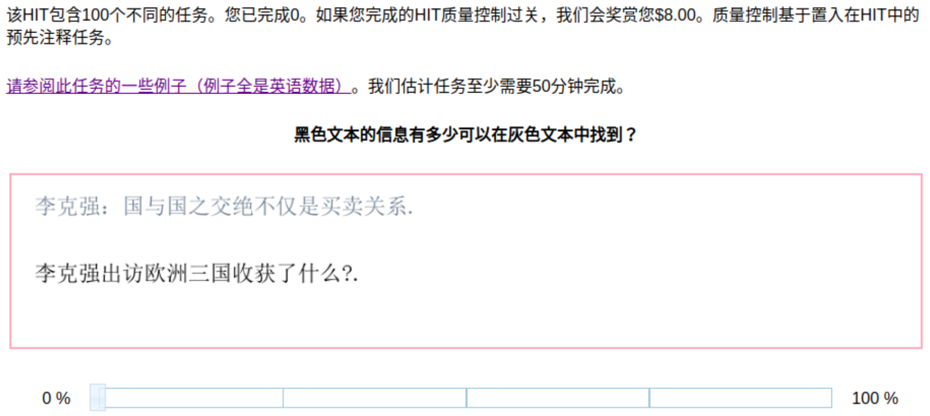
该HIT包含100个不同的任务。您已完成0。如果您完成的HIT质量控制过关，我们会奖赏您$8.00。质量控制基于置入在HIT中的预先注释任务。

请参阅此任务的一些例子（例子全是英语数据）。我们估计任务至少需要50分钟完成。

黑色文本的信息有多少可以在灰色文本中找到？

李克强：国与国之交绝不仅是买卖关系.

李克强出访欧洲三国收获了什么?.

Figure 5: MTurk annotation interface for **Chinese**. Due to the page limit for the Appendix, the annotation interface for the other languages can be found at https://github.com/fajri91/Multi_SummEval

	Focus									**Coverage**								
	EN	**ID**	**FR**	**TR**	**ZH**	**RU**	**DE**	**ES**	**Avg**	**EN**	**ID**	**FR**	**TR**	**ZH**	**RU**	**DE**	**ES**	**Avg**
Traditional Metrics																		
ROUGE-1	0.59	0.69	0.62	0.75	0.80	0.33	0.77	0.45	0.63	0.58	0.71	0.56	0.79	0.78	0.47	0.75	0.59	0.65
ROUGE-2	0.58	0.63	0.64	0.75	0.78	0.37	0.78	0.56	0.64	0.54	0.65	0.62	0.77	0.75	0.39	0.77	0.63	0.64
ROUGE-3	0.49	0.55	0.62	0.70	0.68	0.27	0.77	0.51	0.58	0.47	0.58	0.59	0.69	0.65	0.28	0.75	0.58	0.57
ROUGE-L	0.60	0.69	0.62	0.75	0.80	0.32	0.76	0.46	0.62	0.57	0.70	0.55	0.79	0.78	0.47	0.74	0.58	0.65
ROUGE-S	0.61	0.71	0.61	0.75	0.81	0.36	0.77	0.45	0.63	0.60	0.72	0.56	0.79	0.79	0.39	0.75	0.58	0.65
ROUGE-SU	0.61	0.71	0.61	0.74	0.81	0.31	0.77	0.43	0.62	0.59	0.72	0.56	0.79	0.79	0.45	0.75	0.58	0.66
ROUGE-W.12	0.61	0.67	0.61	0.74	0.79	0.32	0.76	0.47	0.62	0.55	0.68	0.55	0.78	0.78	0.48	0.74	0.58	0.64
METEOR	0.47	0.68	0.60	0.74	0.82	0.40	0.75	0.54	0.62	0.61	0.70	0.59	0.81	0.78	0.44	0.74	0.62	0.66
BLEU-4	0.50	0.59	0.62	0.67	0.76	0.05	0.77	0.51	0.56	0.49	0.61	0.58	0.62	0.72	-0.04	0.73	0.55	0.53
MoverScore																		
mono-BERT	0.58	0.65	0.66	**0.81**	0.79	0.38	**0.85**	0.55	0.66	0.59	0.61	0.60	0.81	0.77	0.30	**0.83**	0.59	0.64
mBERT (cased)	0.53	0.68	0.73	0.80	0.77	0.44	0.83	0.60	0.67	0.50	0.68	0.65	0.77	0.74	0.37	0.79	0.65	0.64
mBERT (uncased)	**0.60**	0.69	**0.74**	0.80	0.77	**0.45**	0.83	**0.66**	**0.69**	0.58	0.68	0.68	0.77	0.74	0.42	0.79	0.67	0.67
XLM (base)	0.51	0.63	0.64	0.78	0.70	0.11	0.78	0.53	0.59	0.55	0.62	0.58	0.73	0.65	0.01	0.75	0.59	0.56
XLM (large)	0.51	0.59	0.63	0.72	0.54	0.10	0.76	0.44	0.54	0.54	0.61	0.52	0.68	0.52	0.07	0.72	0.50	0.52
BERTScore																		
mono-BERT	0.58	0.70	0.67	0.77	**0.81**	0.34	0.84	0.60	0.66	**0.64**	**0.73**	**0.70**	**0.86**	**0.78**	0.57	**0.83**	**0.70**	**0.73**
mBERT (cased)	0.53	**0.71**	0.68	0.78	0.78	0.41	0.82	0.55	0.66	**0.64**	0.72	0.58	0.85	0.77	**0.63**	0.77	0.67	0.70
mBERT (uncased)	0.58	0.70	0.66	0.79	0.79	0.40	0.83	0.58	0.67	0.60	0.73	0.60	0.85	0.77	**0.63**	0.77	0.66	0.70
XLM (base)	0.56	0.65	0.61	0.78	0.78	0.04	0.78	0.49	0.59	0.62	0.71	0.58	0.84	0.70	0.55	0.76	0.64	0.67
XLM (large)	0.57	0.66	0.62	0.78	0.77	0.17	0.80	0.52	0.61	0.62	0.69	0.60	0.84	0.72	0.50	0.78	0.65	0.68

Table 8: Spearman correlation (ρ) between automatic metrics and human judgments (for Pointer Generator and BERT models combined). We compute the precision and recall of ROUGE and BERTScore for focus and coverage, respectively. BERTScore uses the optimized layer, and other metrics are computed by using default configuration of the original implementation.

CoMAE: A Multi-factor Hierarchical Framework for Empathetic Response Generation

Chujie Zheng[†], **Yong Liu**[‡], **Wei Chen**[‡], **Yongcai Leng**[‡], **Minlie Huang**[†*]
[†]The CoAI group, DCST, Institute for Artificial Intelligence,
[†]State Key Lab of Intelligent Technology and Systems,
[†]Beijing National Research Center for Information Science and Technology,
[†]Tsinghua University, Beijing 100084, China
[‡]Sogou Inc., Beijing, China
`chujiezhengchn@gmail.com, aihuang@tsinghua.edu.cn`

Abstract

The capacity of empathy is crucial to the success of open-domain dialog systems. Due to its nature of multi-dimensionality, there are various factors that relate to empathy expression, such as communication mechanism, dialog act and emotion. However, existing methods for empathetic response generation usually either consider only one empathy factor or ignore the hierarchical relationships between different factors, leading to a weak ability of empathy modeling. In this paper, we propose a multi-factor hierarchical framework, CoMAE, for empathetic response generation, which models the above three key factors of empathy expression in a hierarchical way. We show experimentally that our CoMAE-based model can generate more empathetic responses than previous methods. We also highlight the importance of hierarchical modeling of different factors through both the empirical analysis on a real-life corpus and the extensive experiments. Our codes and used data are available at `https://github.com/chujiezheng/CoMAE`.

1 Introduction

Empathy, which refers to the capacity to understand or feel what another person is experiencing (Rothschild, 2006; Read, 2019), is a critical capability to open-domain dialog systems (Zhou et al., 2018b). As shown in previous research, empathetic conversational models can improve user satisfaction and receive more positive feedback in numerous domains (Klein, 1998; Liu and Picard, 2005; Brave et al., 2005; Fitzpatrick et al., 2017; Liu et al., 2021). Recently, there have also been numerous works devoted to improving the dialog models' ability to understand the feelings of interlocutors (Rashkin et al., 2019; Lin et al., 2019; Majumder et al., 2020), which makes the dialog models more empathetic to a certain extent.

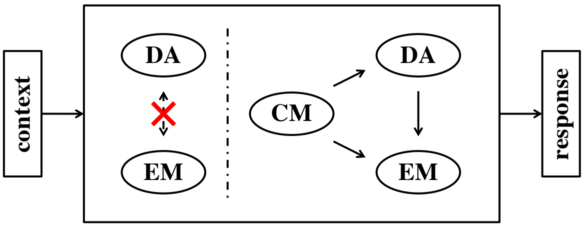

Figure 1: Our proposed hierarchical framework: CoMAE (right). The directed arrows denote dependencies. We also present the framework (left) of EmpTransfo (Zandie and Mahoor, 2020) for comparison.

However, empathy is a multi-dimensional construct (Davis et al., 1980) rather than merely recognizing the interlocutor's emotion (Lin et al., 2019) or emotional responding (Zhou et al., 2018a). It consists of two broad aspects related to *cognition* and *affection* (Omdahl, 2014; Paiva et al., 2017). The cognitive aspect requires understanding and interpreting the situation of the interlocutor (Elliott et al., 2018), which is reflected in the **dialog act** taken in the conversation (De Vignemont and Singer, 2006), such as questioning (e.g., *What's wrong with it?*), consoling (e.g., *You'll get through this*), etc. The affective aspect relates to properly expressing **emotion** in reaction to the experiences and feelings shared by the interlocutor, such as admiration (e.g., *Congratulations!*), sadness (e.g., *I am sorry to hear that*), etc. Very recently, Sharma et al. (2020) further characterizes the text-based expressed empathy based on the above two aspects as three **communication mechanisms**, which is a more higher-level and abstract factor that relates to empathy expression.

In this paper, we propose a novel framework named **CoMAE** for empathetic response generation (Section 3), which contains the aforemen-

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 813–824
August 1–6, 2021.

tioned three key factors of empathy expression: **Co**mmunication **M**echanism (CM), dialog **A**ct (DA) and **E**motion (EM). Specifically, when model these empathy factors simultaneously, we adopt a **hierarchical** way instead of following previous works that treat multiple factors independently, such like EmpTransfo (Zandie and Mahoor, 2020) that considers both DA and EM (see Figure 1 for comparison). Such approaches hold the hypothesis that different factors are independent of each other, which is intuitively unreasonable. In fact, our empirical analysis (Section 4) on a Reddit corpus (Zhong et al., 2020) shows that there are obvious hierarchical relationships between different factors, which confirms the soundness and necessity of hierarchical modeling.

We then devise a CoMAE-based model on top of the pre-trained language model GPT-2 (Radford et al., 2019) (Section 5), and compare the model performance with different combinations of empathy factors and hierarchical modeling. Automatic evaluation (Section 6.3) shows that combining all the three factors hierarchically can achieve the best model performance. Manual evaluation (Section 6.4) demonstrates that our model can generate more empathetic responses than previous methods. Extensive experiments (Section 6.5) further highlight the importance of hierarchical modeling in terms of the selection and realization of empathy factors.

The contributions of this paper can be summarized in three folds:

- Based on the nature of multi-dimensionality of empathy expression, we propose a novel framework, CoMAE, for empathetic response generation. It hierarchically models three key factors of empathy expression: communication mechanism, dialog act and emotion.
- On top of GPT-2, we devise a CoMAE-based model. Experimental results show that our model can generate more empathetic responses than previous methods.
- We empirically analyze the necessity of hierarchical modeling, and highlight its importance especially in terms of the selection and realization of different empathy factors.

2 Related Work

2.1 Factors Related to Empathy Expression

Empathy is a complex multi-dimensional construct (Davis et al., 1980) which consists of two broad aspects related to *cognition* and *affection* (Omdahl, 2014; Paiva et al., 2017). As shown in Section 1, the two aspects are reflected in the dialog act (DA) taken and the emotion (EM) expressed in the conversation respectively.

Based on the theoretical definition of empathy, Sharma et al. (2020) characterize the text-based expressed empathy as 3 communication mechanisms (CM): emotional reaction (ER) (e.g., *I feel really sad for you*), interpretation (IP) (e.g., *This must be terrifying*, *I also have similar situations*), and exploration (EX) (e.g., *Are you still feeling alone now?*).[1] These communication mechanisms are also applied in the recently proposed task of empathetic rewriting (Sharma et al., 2021).

Besides, Zhong et al. (2020) propose that persona, which refers to the social face an individual presents to the world (Jung, 2016), has been shown to be highly correlated with personality (Leary and Allen, 2011), which in turn influences empathy expression (Richendoller and Weaver III, 1994; Costa et al., 2014). While Zhong et al. (2020) do not explain the explicit connection between persona and empathy expression, they suggest that different speakers may have different "styles" for expressing empathy.

2.2 Empathetic Response Generation

In the past years, empathetic response generation has attracted much research interest (Rashkin et al., 2019; Lin et al., 2019; Majumder et al., 2020; Zandie and Mahoor, 2020; Sun et al., 2021). Rashkin et al. (2019) suggest that dialog models can generate more empathetic responses by recognizing the interlocutor's emotion. Lin et al. (2019) propose to design a dedicated decoder to respond each emotion of the interlocutor, which makes the generation process more interpretable. Majumder et al. (2020) adopt the idea of emotional mimicry (Hess and Fischer, 2014) to make the generated responses more empathetic. Inspired by the advances in generative pre-trained language models (Radford et al., 2018, 2019), EmpTransfo (Zandie and Mahoor, 2020) uses GPT (Radford et al., 2018) to generate empathetic responses.

Unlike previous works that only consider the EM factor in empathy modeling, EmpTransfo takes both DA and EM into account. The fundamental

---

[1] As shown in (Sharma et al., 2020), the three communication mechanisms can be properly combined in one utterance. We refer the readers to their original paper for more details about the three communication mechanisms.

difference of EmpTransfo from our work lies in two points: (1) our work further considers communication mechanism in modeling empathy, and (2) we analyze and explore in depth the importance of hierarchically modeling of these empathy factors.

3 CoMAE Framework and Formulation

Our proposed CoMAE framework is shown in Figure 1. CoMAE uses CM as a high-level factor that provides a coarse-grained guidance for empathy expression, and then takes DA and EM to achieve the fine-grained realization. Formally, given the context x, CoMAE divides the generation of the empathetic response y into four steps: (1) predict **CM** C_y conditioned on the context, (2) predict **DA** A_y conditioned on both the context and CoM, (3) predict **EM** E_y based on all the conditions, and (4) generate the final response y. The whole process is formulated as Equation 1:

$$\begin{aligned}\mathbb{P}(y, C_y, A_y, E_y|x) = \mathbb{P}(y|x, C_y, A_y, E_y)\cdot \quad (1)\\ \mathbb{P}(E_y|x, C_y, A_y)\mathbb{P}(A_y|x, C_y)\mathbb{P}(C_y|x).\end{aligned}$$

Note that EM is conditioned on DA, because we intuitively think the expressed emotion is the effect rather than the cause of taking some dialog act. In the other words, one may not adopt the dialog act just for the purpose of expressing some emotion. Hence, realizing the emotion expression as expected is also important in our task, which is the motivation of that we analyze the realization of different factors in Section 6.5.

It is also worth noting that while CoMAE only contains the three factors, such hierarchical framework can be naturally extended to more factors that relate to empathy expression. For instance, Zhong et al. (2020) suggest that persona plays an important role in empathetic conversations. Due to that persona may contain the information about the speaker's style of adopting DA or expressing EM, when integrating persona into empathetic response generation, being conditioned on DA and EM may lead to better performance.

4 Data Preparation and Analysis

While no empathetic conversation corpora provide annotations of diverse empathy factors, there are abundant publicly available resources that make automatic annotation feasible. In this section, we first introduce our used corpus and the resources and tools used in automatic annotation, then we show our empirical analysis to verify the hierarchical relationships between different empathy factors.

4.1 Corpus

Zhong et al. (2020) propose a large-scale empathetic conversation corpus[2] crawled from Reddit. It has two different domains: Happy and Offmychest. The posts in the Happy domain mainly have positive sentiments, while those in the Offmychest domain are usually negative. We adopted their corpus for study for two major reasons: (1) the corpus is real-life, scalable and naturalistic rather than acted (Rashkin et al., 2019), and (2) the manual annotation in (Zhong et al., 2020) shows that most of the last responses are empathetic (73% and 61% for Happy and Offmychest respectively).

4.2 Annotation Resources

Communication Mechanism (CM)[3] Sharma et al. (2020) provide two corpora annotated with CM: TalkLife (`talklife.co`) and Reddit (`reddit.com`), while only the latter is publicly accessible and we thus used the Reddit part. Note that in their original paper, each mechanism is differentiated as three classes of "no", "weak", or "strong". Due to the unbalanced distribution of three classes, we merged "weak" and "strong" into "yes". Finally, we differentiated each mechanism as two classes: "no" or "yes".

Dialog Act (DA)[4] Welivita and Pu (2020) propose a taxonomy of DA (referred as "intent" in the original paper) for empathetic conversations. They first annotate 15 initial types of DA on the ED corpus (Rashkin et al., 2019), and finally obtain 8 high-frequency types of DA with other types merged as others (**8+others**), which are shown in Figure 2.

Emotion (EM)[5] We considered the taxonomy proposed in (Demszky et al., 2020), which contains 27 emotions and a neutral one, because: (1) it has a wide coverage of emotion categories with clear definitions, and (2) the annotated corpus is large-scale and also crawled from Reddit. However, we noted that the original emotion distribution is unbalanced and the too fine-grained taxonomy may lead to the sparsity of partial emotions. Considering the task

---

[2] `https://github.com/zhongpeixiang/PEC`

[3] `https://github.com/behavioral-data/Empathy-Mental-Health`

[4] `https://github.com/anuradha1992/EmpatheticIntents`

[5] `https://github.com/google-research/google-research/tree/master/goemotions`

Classifiers	Corpora	# classes	Acc	F1-macro
CM-ER	Reddit	2	81.2	76.9
CM-IP	Reddit	2	85.7	85.7
CM-EX	Reddit	2	96.4	92.5
DA	ED	9	92.0	87.8
EM	Reddit	10	60.5	60.4

Table 1: Performance of the classifiers. "ED" refers to the corpus of EMPATHETICDIALOGUES (Rashkin et al., 2019).

scenario of empathetic conversation, we adopted the clustering results in (Demszky et al., 2020) and modified the original taxonomy as 9 emotions and a neutral one (**9+neutral**), which are also shown in Figure 2. We show the mapping between our adopted emotions and the original emotions in Appendix A.

4.3 Classifiers

We fine-tuned the RoBERTa[6] (Liu et al., 2019) classifiers for CM, DA and EM, whose performance is summarized in Table 1. They all achieve reasonable performance, ensuring the quality of automatic annotation.

However, we noted that the source domain (Rashkin et al., 2019) of the DA classifier is different from the target domain (Reddit). To verify the quality of DA annotation, we recruited three workers from Amazon Mechanical Turk to judge whether the utterance is consistent with the annotated DA. From the utterances that are not annotated with "others", we randomly sampled 25 utterances for each DA (totally 200) to avoid the impact of unbalanced distribution. Finally, the ratio of being judged as consistent is 0.78 with Fleiss' Kappa $\kappa = 0.621$ (Fleiss, 1971), which indicates substantial agreement ($0.6 < \kappa < 0.8$) and that the automatic annotation of DA is also reliable.

4.4 Data Filtering and Annotation

Following the original data split of (Zhong et al., 2020), we first filtered those conversations where there are more than two speakers (about 15%) to ensure that the last utterance is related to the post. We used the aforementioned classifiers to automatically annotate each utterance with DA and EM, and annotate each final response additionally with CM. We found that the last responses that are not annotated with any CM are more likely to be non-empathetic, thus we filtered the conversations containing such responses (about 40%). Finally, the sizes of Train / Valid / Test-Happy / Test-Offmychest are 125,963 / 16,371 / 11,136 / 6,413 respectively. We show the detailed statistics of automatic annotation in Appendix B.

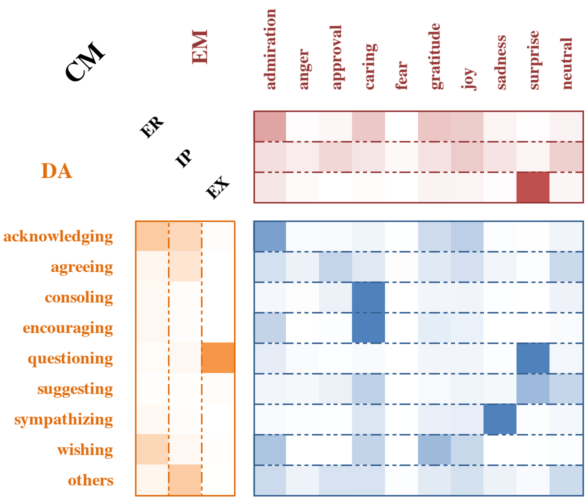

Figure 2: Heat maps of the conditional distributions between the three empathy factors. The **orange** / **red** / **blue** maps are the distributions of **DA** / **EM** / **EM** conditioned on **CM** / **CM** / **DA** respectively.

4.5 Analysis

In order to verify the hierarchical relationships between the three factors, we counted the distribution frequency of each (X, Y)[7] pair, where (X, Y) is one of the three factor pairs: (CM, DA), (CM, EM), (DA, EM). We approximated the statistical frequency of (X, Y) as their joint probability distribution $\mathbb{P}(X, Y)$. We then normalized $\mathbb{P}(X, Y)$ along the X dimension to obtain the conditional distribution of Y given X: $\mathbb{P}(Y|X)$.

Figure 2 shows the heat maps of the conditional distributions of the three factor pairs. The heat maps reveal obvious patterns of the occurrence of Y given X. For instance, when one adopts the DA *encouraging*, he usually expresses the EM *caring* instead of *approval* or *joy*. If one expresses empathy with the CM *exploration (EX)*, he almost always adopts the DA *questioning* and expresses the EM *surprise*. Hence, considering the hierarchical relationships between different empathy factors is reasonable and natural, and is also necessary for better empathy modeling.

---

[6] `https://huggingface.co/roberta-base`

[7] X or Y is the random variable that represents CM, DA, or EM.

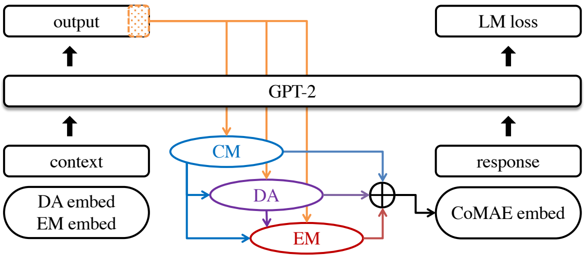

Figure 3: The overall architecture of our CoMAE-based model. The position and speaker embeddings are omitted for simplicity. The orange dashed block denotes the output hidden state at the last position of the context.

5 Methodology

5.1 Model Architecture

Our devised CoMAE-based model uses GPT-2 as the backbone (Radford et al., 2019). The overall architecture is shown in Figure 3.

Firstly, our model takes the dialog context x as input. The context x is the concatenation of history utterances: $x = (u_1, u_2, \ldots, u_N)$, where N is the length of dialog history. Any two adjacent utterances are also separated by the special token `[EOS]`. Each history utterance u_i contains a sequence of tokens: $u_i = (u_{i,1}, u_{i,2}, \ldots, u_{i,l_i})$, where l_i is the length of u_i. Each utterance u_i is labeled with the corresponding speaker $k_{u_i} \in \{0, 1\}$ (only 2 speakers). We denote the annotated DA and EM of each utterance u_i as $A_{u_i} \in [0, 9)$ and $E_{u_i} \in [0, 10)$ respectively. Suppose that the token id and the position id of $u_{i,j}$ are denoted as $w_{u_{i,j}} \in [0, |\mathcal{V}|)$ ($\mathcal{V}$ is the vocabulary) and $p_{u_{i,j}} \in [0, 1024)$ (the maximum input length is 1024) respectively, the representation of each token $u_{i,j}$ is the summation of the following embeddings:

$$\begin{aligned} \boldsymbol{e}_{u_{i,j}} = \boldsymbol{M}_W \left[w_{u_{i,j}}\right] + \boldsymbol{M}_P \left[p_{u_{i,j}}\right] + \\ \boldsymbol{M}_K \left[k_{u_i}\right] + \boldsymbol{M}_A \left[A_{u_i}\right] + \boldsymbol{M}_E \left[E_{u_i}\right], \end{aligned} \tag{2}$$

where $\boldsymbol{M}_W \in \mathbb{R}^{|\mathcal{V}| \times d}, \boldsymbol{M}_P \in \mathbb{R}^{1024 \times d}, \boldsymbol{M}_K \in \mathbb{R}^{2 \times d}, \boldsymbol{M}_A \in \mathbb{R}^{9 \times d}, \boldsymbol{M}_E \in \mathbb{R}^{10 \times d}$ denote the embedding matrices of word, position, speaker, DA and EM respectively, and $[\cdot]$ denotes the indexing operation. We denote the output hidden states after feeding x into the model as $\boldsymbol{H}_x \in \mathbb{R}^{l_x \times d}$, where l_x is the total length of context x.

Next, we use the hidden state at the last position of the context, $\boldsymbol{h}_x = \boldsymbol{H}_x[-1] \in \mathbb{R}^d$, to hierarchically predict the CM, DA and EM of the target response. We first separately predict[8] $\widehat{C}_y^{(i)} \in \{0, 1\}$ for each $i \in \{\mathrm{ER}, \mathrm{IP}, \mathrm{EX}\}$, which indicates whether to adopt the CM i:

$$\begin{aligned} \boldsymbol{h}_C^{(i)} &= \mathbf{F}_C^{(i)} \left(\boldsymbol{h}_x\right) \in \mathbb{R}^d, & (3) \\ \widehat{C}_y^{(i)} &\sim \mathbb{P}\left(C_y^{(i)} \middle| x\right) = \operatorname{softmax}\left(\boldsymbol{M}_C^{(i)} \boldsymbol{h}_C^{(i)}\right), \\ \widehat{C}_y &= \left(\widehat{C}_y^{(\mathrm{ER})}, \widehat{C}_y^{(\mathrm{IP})}, \widehat{C}_y^{(\mathrm{EX})}\right), \\ \boldsymbol{e}_{\widehat{C}_y} &= \sum_{i \in \{\mathrm{ER},\mathrm{IP},\mathrm{EX}\}} \boldsymbol{M}_C^{(i)} \left[\widehat{C}_y^{(i)}\right], & (4) \end{aligned}$$

where each $\mathbf{F}_C^{(i)}$ is a non-linear layer activated with $\tanh$, and each $\boldsymbol{M}_C^{(i)} \in \mathbb{R}^{2 \times d}$ denotes the embedding matrix of the CM $i \in \{\mathrm{ER}, \mathrm{IP}, \mathrm{EX}\}$. Based on the context x and the predicted CMs $\widehat{C}_y$, we next predict DA:

$$\boldsymbol{h}_A = \mathbf{F}_A \left(\left[\boldsymbol{h}_x; \boldsymbol{e}_{\widehat{C}_y}\right]\right) \in \mathbb{R}^d, \tag{5}$$

$$\widehat{A}_y \sim \mathbb{P}\left(A_y \middle| x, \widehat{C}_y\right) = \operatorname{softmax}\left(\boldsymbol{M}_A \boldsymbol{h}_A\right), \tag{6}$$

where $[\cdot; \cdot]$ denotes vector concatenation and $\mathbf{F}_A$ is a non-linear layer. Note that we share the parameters of DA embeddings with the classification head (Equation 6), which is consistent with the way in GPT-2 (Radford et al., 2019) where the parameters of word embeddings are shared with the LM head (Equation 10). EM is predicted similarly but conditioned additionally on the predicted DA $\widehat{A}_y$:

$$\boldsymbol{h}_E = \mathbf{F}_E \left(\left[\boldsymbol{h}_x; \boldsymbol{e}_{\widehat{C}_y}; \boldsymbol{M}_A \left[\widehat{A}_y\right]\right]\right) \in \mathbb{R}^d, \tag{7}$$

$$\widehat{E}_y \sim \mathbb{P}\left(E_y \middle| x, \widehat{C}_y, \widehat{A}_y\right) = \operatorname{softmax}\left(\boldsymbol{M}_E \boldsymbol{h}_E\right), \tag{8}$$

where $\mathbf{F}_E$ is also a non-linear layer.

Finally, we add all the factors to obtain the fused embedding $\boldsymbol{e}_{\mathrm{CoMAE}}$ that controls the empathy expression of the response:

$$\boldsymbol{e}_{\mathrm{CoMAE}} = \boldsymbol{e}_{\widehat{C}_y} + \boldsymbol{M}_A \left[\widehat{A}_y\right] + \boldsymbol{M}_E \left[\widehat{E}_y\right].$$

The embedding of each input token $\widehat{y}_t$ in the response is as follows:

$$\begin{aligned} \boldsymbol{e}_{\widehat{y}_t} = \boldsymbol{M}_W \left[w_{\widehat{y}_t}\right] + \boldsymbol{M}_P \left[p_{\widehat{y}_t}\right] + \\ \boldsymbol{M}_K \left[k_y\right] + \boldsymbol{e}_{\mathrm{CoMAE}}. \end{aligned} \tag{9}$$

Suppose that the output hidden state corresponding to $\widehat{y}_t$ is $\boldsymbol{s}_t$, then we predict the next token $\widehat{y}_{t+1}$

---

[8] In the mathematical notation used in this paper, we distinguish the ground truth value and the predicted value of a variable X with the symbols X^* and $\widehat{X}$ respectively.

through the LM head:

$$\begin{aligned}\widehat{y}_{t+1} &\sim \mathbb{P}\left(y_{t+1}\left|\widehat{y}_{\leq t}; x, \widehat{C}_y, \widehat{A}_y, \widehat{E}_y\right.\right) \\ &= \operatorname{softmax}\left(\boldsymbol{M}_W \boldsymbol{s}_t\right),\end{aligned} \tag{10}$$

where the parameters of the LM head are shared with the word embedding matrix $\boldsymbol{M}_W$.

5.2 Training

The optimization object contains two parts. One part is the negative log likelihood loss $\mathcal{L}_{\text{NLL}}$ of the target response:

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{l_y}\sum_{t=1}^{l_y} \ln \mathbb{P}\left(y_t^* \left| y_{<t}^*; x, C_y^*, A_y^*, E_y^*\right.\right),$$

where l_y is the length of the golden response. The other part is the prediction losses of CM $\mathcal{L}_C$, DA $\mathcal{L}_A$, and EM $\mathcal{L}_E$:

$$\mathcal{L}_C = -\sum_{i\in\{\text{ER,IP,EX}\}} \ln \mathbb{P}\left(C_y^{(i)*}\middle| x\right), \tag{11}$$

$$\mathcal{L}_A = -\ln \mathbb{P}\left(A_y^* \left| x, C_y^*\right.\right), \tag{12}$$

$$\mathcal{L}_E = -\ln \mathbb{P}\left(E_y^* \left| x, C_y^*, A_y^*\right.\right). \tag{13}$$

The complete optimization object is the summation of the above losses: $\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda\left(\mathcal{L}_C + \mathcal{L}_A + \mathcal{L}_E\right)$, where λ is the weight of the prediction losses. We set λ to 1.0 in our experiments.

5.3 Discussion

It is worth noting that the supervision signals of predictions (from Equation 11 to 13) combined with hierarchical modeling (from Equation 3 to 8) enable the model to establish the connections between the embeddings of the three factors. For instance, in Equation 6, the embedding matrix of DA, $\boldsymbol{M}_A$, is multiplied with $\boldsymbol{h}_A$, which explicitly contains the information of the embedding matrices of CM, $\boldsymbol{M}_C^{(i)}$ (Equation 4 and 5). The case of Equation 8 is similar, where $\boldsymbol{M}_E$ is multiplied with $\boldsymbol{h}_E$ that directly relates to $\boldsymbol{M}_C^{(i)}$ and $\boldsymbol{M}_A$.

Hence, consider two models where one uses hierarchical modeling and the other does not (predicting each factor separately). When the two models are fed with the same empathy factors, saying the triplet (C_y, A_y, E_y) is designated validly, we can expect that the former model has better performance than the latter one. This conjecture will be verified in the automatic evaluation (Section 6.3).

6 Experiments

6.1 Compared Models

We investigated the model performance with different combinations of empathy factors and hierarchical modeling:
(1) **Vanilla**: the GPT-2 model directly fine-tuned on the corpus without adding any empathy factor;
(2) **+CM**, **+DA**, **+EM**: the GPT-2 models equipped with one of the three factors;
(3) **CM || DA**, **CM || EM**, **DA || EM**, **CM || DA || EM**: the models equipped with two or all of the three factors, but predicting each factor separately without hierarchical modeling;
(4) **CM → DA**, **CM → EM**, **DA → EM**, **CM → DA → EM**: the models that are similar to (3) but utilize the hierarchical relationships, where → denotes dependency.

Note that the baseline DA || EM is consistent with EmpTransfo[9] (Zandie and Mahoor, 2020), and CM → DA → EM is exactly our devised model described in Section 5.1.

6.2 Implementation Details

All the models were implemented with PyTorch[10] (Paszke et al., 2019) and the Transformers library[11] (Wolf et al., 2020). We used the pre-trained GPT-2 with the size of 117M parameters (768 hidden sizes, 12 heads, 12 layers) for all the models. The responses were decoded by Top-p sampling with $p = 0.9$ and the temperature $\tau = 0.7$ (Holtzman et al., 2019). We trained all the models with Adam (Kingma and Ba, 2014) optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was 10^{-4} and was dynamically changed using the linear warmup (Popel and Bojar, 2018) with 4000 warmup steps. All the models were fine-tuned for 5 epochs with the batch size 16 on one NVIDIA RTX 2080Ti GPU. We selected the checkpoint for each model where the model obtains the lowest perplexity score on the Valid set.

6.3 Automatic Evaluation

The automatic evaluation uses the golden responses as reference to evaluate the responses generated by

---

[9] DA || EM has the same input representation except the speaker embeddings as EmpTransfo, but is instead fine-tuned from GPT-2 rather than GPT. Besides, we did not adopt the next sentence prediction (NSP) task as in (Zandie and Mahoor, 2020), because we empirically found that adding NSP leads to worse performance.

[10] `https://pytorch.org/`

[11] `https://github.com/huggingface/transformers`

	Models	PPL	B-2	R-L	Greedy
Happy	**Vanilla**	18.82	5.95*	15.00*	66.09*
	+CM	18.21	6.67*	17.64*	66.95*
	+DA	18.01	7.18*	18.09*	67.35*
	+EM	17.88	7.51*	18.27*	67.78*
	CM || DA	17.83	7.76*	18.85*	67.78*
	CM || EM	17.57	8.17*	19.58*	68.25*
	DA || EM	17.38	8.37*	19.91*	68.59*
	CM || DA || EM	17.26	9.21	20.75	68.86
	CM → DA	17.69	7.95*	18.96*	67.79*
	CM → EM	17.45	8.04*	19.49*	68.08*
	DA → EM	17.28	8.73*	20.09*	68.59*
	CM → DA → EM	**17.02**	**9.44**	**20.76**	**68.92**
Offmychest	**Vanilla**	22.11	5.66*	13.75*	68.40*
	+CM	21.44	6.65*	17.62*	69.68*
	+DA	21.34	7.11*	17.44*	69.67*
	+EM	21.26	6.75*	17.40*	69.63*
	CM || DA	21.07	7.56*	18.41*	70.16*
	CM || EM	20.83	7.78*	18.97*	70.34*
	DA || EM	20.85	7.48*	18.49*	70.19*
	CM || DA || EM	20.63	8.23	19.32	70.54
	CM → DA	20.87	7.70*	18.58*	70.33*
	CM → EM	20.72	7.71*	18.63*	70.31*
	DA → EM	20.68	7.89*	18.66*	70.25*
	CM → DA → EM	**20.35**	**8.35**	**19.54**	**70.68**

Table 2: Results of automatic evaluation. The best results are in **bold**. DA || EM is consistent with EmpTransfo (Zandie and Mahoor, 2020). CM → DA → EM is our devised model described in Section 5.1. Scores that are significantly worse than the best scores are marked with * (Student's t-test, p-value < 0.05).

models. However, when the responses are generated based on the predicted CM / DA / EM, it is not appropriate to compare the generated responses with the reference ones (Liu et al., 2016). Thus, in automatic evaluation we only considered the setting where the models are fed with the ground truth empathy factors. The results where the generated responses are based on the predicted factors will be analyzed in the later experiments.

The automatic metrics we adopted include perplexity (**PPL**), BLEU-2 (**B-2**) (Papineni et al., 2002), ROUGE-L (**R-L**) (Lin, 2004), and the BOW Embedding-based (Liu et al., 2016) **Greedy** matching score. The metrics except PPL were calculated with an NLG evaluation toolkit[12] (Sharma et al., 2017), where the generated responses were tokenized with NLTK[13] (Loper and Bird, 2002).

Results are shown in Table 2. We analyze the results from the following three perspectives:

General Performance Our model achieves the best performance on all the metrics on both domains, and most of the advantages over the competitors are statistically significant.

Impact of Empathy Factors The model performance vary from different combinations of empathy factors. First, considering more empathy factors always leads to better performance (e.g., CM → DA → EM > CM → EM > +EM > Vanilla). Second, EM brings the most gains to the model performance among the three factors. It may be because emotion is the most explicit factor that influences empathy expression (Sharma et al., 2020). In contrast, CM brings fewer gains than DA and EM. The reason may be that CM provides a high-level but coarse-grained guidance for empathetic response generation, lacking a fine-grained control like DA or EM. While the responses in the corpus of (Zhong et al., 2020) are not too long (≤ 30 words), we believe that CM plays an important role in generating longer empathetic responses, which may require the planning of multiple methanisms and more diverse usage of DA and EM.

Impact of Hierarchical modeling We noticed that for almost all the models that adopt multiple empathy factors, hierarchical modeling always leads to better performanc (e.g., CM → DA → EM > CM || DA || EM, DA → EM > DA || EM). This phenomenon is not trivial because the models with or without hierarchical modeling are all fed with the same empathy factors as the reference responses. It confirms our conjecture in Section 5.2 that hierarchical modeling can establish the connections between the embeddings of different factors, thus leading to a better capacity of empathy modeling. However, (CM, EM) is an exception. It may be due to that the pair (CM, EM) has a weaker correlation (the lowest manual information, Section 4.5) than other pairs.

6.4 Manual Evaluation

In manual evaluation, the models generate responses based on the empathy factors sampled from the predicted probability distributions. When sampling DA or EM, we used the Top-p filtering with $p = 0.9$ (Holtzman et al., 2019) to ensure the validness of the sampled results.

The manual evaluation is based on pair-wise comparison, and the metrics for manual evaluation include: **Fluency** (which response has better fluency and readability), **Coherence** (which response has better coherence and higher relevance to the context), and **Empathy** (which response shows bet-

---

[12] `https://github.com/Maluuba/nlg-eval`

[13] `https://www.nltk.org/`

Comparisons	Metrics	Win	Lose	κ
CM → DA → EM	Flu	33.3	34.8	0.330
vs.	Coh	35.3	39.3	0.431
DA → EM	Emp*	**39.3**	32.3	0.402
CM → DA → EM	Flu	37.3	34.5	0.383
vs.	Coh*	**41.6**	33.4	0.412
CM || DA || EM	Emp	43.4	39.6	0.416
DA → EM	Flu	36.2	38.5	0.381
vs.	Coh	**40.0**	35.7	0.523
DA || EM	Emp	44.7	42.0	0.497

Table 3: Results of manual evaluation. Ties are not shown. The metrics with significant gaps are marked with * (sign test, p-value < 0.05). κ denotes Fleiss' Kappa, whose values indicate fair agreement ($0.2 < \kappa < 0.4$) or moderate agreement ($0.4 < \kappa < 0.6$).

	X, Y	Acc of X	Prop.	Hits@1/3 of Y	
Happy	**CM || DA**	69.5	68.9	46.1*	81.5*
	CM → DA	70.2		**49.5**	**85.1**
	CM || EM	69.5	68.9	42.3	80.1*
	CM → EM	70.4		**42.8**	**82.7**
	DA || EM	40.1	34.6	50.3*	86.5*
	DA → EM	40.0		**53.5**	**89.7**
Offmychest	**CM || DA**	48.4	45.2	41.3*	67.9*
	CM → DA	49.2		**45.9**	**75.1**
	CM || EM	45.7	42.9	47.2*	74.2*
	CM → EM	46.1		**50.3**	**77.2**
	DA || EM	35.0	30.7	60.5*	84.8*
	DA → EM	34.9		**70.2**	**88.3**

Table 4: Results of the Hits@1/3 of predicting Y given that X is predicted rightly. "Prop." denotes the proportion of the cases where both models $X \mid\mid Y$ and $X \rightarrow Y$ predict X rightly. Scores that are significantly improved after using hierarchical modeling are marked with * (sign test, p-value < 0.001).

ter understanding of the partner's experiences and feelings, and which response expresses empathy in the way that the annotators prefer). The pair-wise comparison is conducted between three pairs of models: (1) CM → DA → EM vs. DA → EM, (2) CM → DA → EM vs. CM || DA || EM, and (3) DA → EM vs. DA || EM. We randomly sampled 100 conversations from each test set of two domains (totally 200), and recruited three workers from Amazon Mechanical Turk for annotation.

Results are shown in Table 3. From all the three pairs, we find that the responses generated by these GPT-2-based models have similar fluency. The results of (1) indicate that further considering CM can significantly improve the empathy of generated responses, while the coherence may slightly decrease.

	Models	CM	DA	EM
Happy	**CM || DA**	69.6*	76.2*	-
	CM → DA	**79.3**	**83.6**	-
	CM || EM	73.8*	-	78.0*
	CM → EM	**76.6**	-	**82.4**
	DA || EM	-	77.5*	75.0*
	DA → EM	-	**87.3**	**85.7**
	CM || DA || EM	68.5*	70.3*	71.9*
	CM → DA → EM	**76.7**	**83.7**	**81.2**
Offmychest	**CM || DA**	61.8*	65.6*	-
	CM → DA	**71.4**	**74.8**	-
	CM || EM	65.4*	-	66.1*
	CM → EM	**71.1**	-	**74.6**
	DA || EM	-	63.7*	58.3*
	DA → EM	-	**79.5**	**75.1**
	CM || DA || EM	59.0*	60.8*	58.9*
	CM → DA → EM	**70.7**	**76.2**	**72.6**

Table 5: Realization scores. All the scores are significantly improved after using hierarchical modeling (sign test, p-value < 0.00001).

It may be because that the communication mechanisms like interpretation sometimes lead to the responses that are less relevant to the contexts (especially those sharing experiences). The results of (2) and (3) indicate that hierarchical modeling improves the coherence of generated responses. The more empathy factors are modeled, the larger improvement can be obtained.

6.5 Further Analysis of Hierarchical modeling

To give further insights of the superiority of hierarchical modeling, we analyzed (1) the prediction and (2) the realization of empathy factors.

Prediction For each pair (X, Y) in (CM, DA), (CM, EM), (DA, EM), we paired the models $X \mid\mid Y$ and $X \rightarrow Y$ for comparison. Our purpose is to observe whether the prediction of X improves that of Y after using hierarchical modeling. Note that when taking the ground truth as reference, it is not appropriate to directly judge the prediction accuracy by comparing $\widehat{Y}$ and Y^* if $\widehat{X} \neq X^*$. We thus computed the conditional probability that Y is predicted rightly given that X is predicted rightly: $\mathbb{P}\left(\widehat{Y} = Y^* \middle| \widehat{X} = X^*\right)$.

Results are shown in Table 4. While the accuracy of predicting X of $X \mid\mid Y$ and $X \rightarrow Y$ is close, the prediction of Y is significantly enhanced by hierarchical modeling. The results demonstrate that hierarchical modeling enables the model to select more proper empathy factors.

Realization Recall that in manual evaluation, the models generate a response based on the sampled empathy factors $\widehat{C}_y, \widehat{A}_y, \widehat{E}_y$. To verify whether these factors are well realized, we used the classifiers in Section 4.3 to identify the empathy factors displayed in the generated responses. Suppose that the identification results are $\widetilde{Z}_y, \forall Z \in \{C, A, E\}$, we computed the ratio of $\widehat{Z}_y = \widetilde{Z}_y$ as the realization score of Z.

Results are shown in Table 5. The realization of all the factors is significantly improved by hierarchical modeling. It is intuitive because hierarchical modeling can avoid the cases where the sampled factors are inappropriate or even conflicting, thus reducing the noise of empathy factors in response generation.

6.6 Case Study

We show the generated responses with different empathy factors in Figure 4. The adoption of the CM *emotional reaction* causes our model to express the same EM *admiration* (*i'm proud of you!*) as DA → EM (*good for you, man!*), while the two models generate the same sentence (*keep it up!*) when taking the DA *encouraging*. However, the further adoption of the CM *interpretation* causes our model to further share its own experiences and feelings (*i have been sober for about 10 years, and it's the best feeling ever*). As a result, with the enhancement of multiple empathy factors, the response generated by our model is more engaging and empathetic while maintaining the coherence.

Besides, we noticed another phenomenon occurring when all the three CMs are adopted. In this case, the three CMs are usually represented separately in different sentences (e.g., *I am so happy for you! I also had tried to be sober but failed. How did you make it?*), which is consistent with the results of empathetic rewriting (Sharma et al., 2021). Recall that we add the same CoMAE embeddings for all the tokens in the response during generation (Section 5.1). Such uniform operation seems non-optimal for the nonuniform realization of different CMs, especially when generating a longer empathetic response that contains multiple sentences with different CMs, DAs or EMs. We believe there is still much room of improvement when applying our CoMAE framework to longer response generation, like combining CoMAE's multi-factor hierarchical modeling with planning-based dialog generation methods (Ghazarian et al., 2021).

Post
you might remember me posting here when i had less than a month sober a little while back. well, yesterday i hit 100 days without alcohol and celebrated by solo hiking my state's tallest mountain!

Golden
ok that is an awesome pic! love it and the story thank you!

EM
you're doing great!

DA → EM
good for you, man! keep it up!

CM → DA → EM (Ours)
i'm proud of you! i have been sober for about 10 years, and it's the best feeling ever. keep it up!

Figure 4: Responses generated with different empathy factors. All the generated responses express the EM *admiration*. **DA → EM** takes the DA *encouraging*. **Ours** further adopts the CM *emotional reaction* and *interpretation*.

7 Conclusion

In this paper, we present a multi-factor hierarchical framework CoMAE for empathetic response generation. It contains three key factors of empathy expression: communication mechanism, dialog act and emotion, and models these factors in a hierarchical way. With our devised CoMAE-based model, we empirically demonstrate the effectiveness of these empathy factors, as well as the necessity and importance of hierarchical modeling.

As future work, the CoMAE framework can be naturally extended to more factors that relate to empathy expression, such as persona (Zhong et al., 2020), by exploring the hierarchical relationships between different factors.

Acknowledgments

This work was supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005.

References

Scott Brave, Clifford Nass, and Kevin Hutchinson. 2005. Computers that care: investigating the effects of orientation of emotion exhibited by an embodied computer agent. *International journal of human-computer studies*, 62(2):161–178.

Patrício Costa, Raquel Alves, Isabel Neto, Pedro Marvao, Miguel Portela, and Manuel Joao Costa. 2014. Associations between medical student empathy and personality: a multi-institutional study. *PloS one*, 9(3):e89254.

Mark H Davis et al. 1980. A multidimensional approach to individual differences in empathy. *Journal of Personality and Social Psychology*.

Frederique De Vignemont and Tania Singer. 2006. The empathic brain: how, when and why? *Trends in cognitive sciences*, 10(10):435–441.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054, Online. Association for Computational Linguistics.

Robert Elliott, Arthur C Bohart, Jeanne C Watson, and David Murphy. 2018. Therapist empathy and client outcome: An updated meta-analysis. *Psychotherapy*, 55(4):399.

Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. 2017. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): a randomized controlled trial. *JMIR mental health*, 4(2):e19.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Sarik Ghazarian, Zixi Liu, Tuhin Chakrabarty, Xuezhe Ma, Aram Galstyan, and Nanyun Peng. 2021. Discol: Toward engaging dialogue systems through conversational line guided response generation. *arXiv preprint arXiv:2102.02191*.

Ursula Hess and Agneta Fischer. 2014. Emotional mimicry: Why and when we mimic emotions. *Social and Personality Psychology Compass*, 8(2):45–57.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Carl Jung. 2016. *Psychological types*. Routledge.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jonathan Tarter Klein. 1998. *Computer response to user frustration*. Ph.D. thesis, Massachusetts Institute of Technology.

Mark R Leary and Ashley Batts Allen. 2011. Personality and persona: Personality processes in self-presentation. *Journal of personality*, 79(6):1191–1218.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. 2019. MoEL: Mixture of empathetic listeners. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 121–132, Hong Kong, China. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

K Liu and Rosalind W Picard. 2005. Embedded empathy in continuous, interactive health assessment. In *CHI Workshop on HCI Challenges in Health Assessment*, volume 1, page 3. Citeseer.

Siyang Liu, Chujie Zheng, Orianna Demasi, Sahand Sabour, Yu Li, Zhou Yu, Yong Jiang, and Minlie Huang. 2021. Towards emotional support dialog systems. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Navonil Majumder, Pengfei Hong, Shanshan Peng, Jiankun Lu, Deepanway Ghosal, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. 2020. MIME: MIMicking emotions for empathetic response generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8968–8979, Online. Association for Computational Linguistics.

Becky Lynn Omdahl. 2014. *Cognitive appraisal, emotion, and empathy*. Psychology Press.

Ana Paiva, Iolanda Leite, Hana Boukricha, and Ipke Wachsmuth. 2017. Empathy in virtual agents and robots: a survey. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(3):1–40.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy. Association for Computational Linguistics.

Hannah Read. 2019. A typology of empathy and its many moral forms. *Philosophy Compass*, 14(10):e12623.

Nadine R Richendoller and James B Weaver III. 1994. Exploring the links between personality and empathic response style. *Personality and individual Differences*, 17(3):303–311.

Babette Rothschild. 2006. *Help for the helper: The psychophysiology of compassion fatigue and vicarious trauma*. WW Norton & Company.

Ashish Sharma, Inna W Lin, Adam S Miner, David C Atkins, and Tim Althoff. 2021. Towards facilitating empathic conversations in online mental health support: A reinforcement learning approach. In *The World Wide Web Conference*.

Ashish Sharma, Adam Miner, David Atkins, and Tim Althoff. 2020. A computational approach to understanding empathy expressed in text-based mental health support. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5263–5276, Online. Association for Computational Linguistics.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Hao Sun, Zhenru Lin, Chujie Zheng, Siyang Liu, and Minlie Huang. 2021. Psyqa: A chinese dataset for generating long counseling text for mental health support. In *Findings of the Association for Computational Linguistics: ACL 2021*.

Anuradha Welivita and Pearl Pu. 2020. A taxonomy of empathetic response intents in human social conversations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4886–4899, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Rohola Zandie and Mohammad H Mahoor. 2020. Emptransfo: A multi-head transformer architecture for creating empathetic dialog systems. *arXiv preprint arXiv:2003.02958*.

Peixiang Zhong, Chen Zhang, Hao Wang, Yong Liu, and Chunyan Miao. 2020. Towards persona-based empathetic conversational models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6556–6566, Online. Association for Computational Linguistics.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2018b. The design and implementation of xiaoice, an empathetic social chatbot. *arXiv preprint arXiv:1812.08989*.

A Emotion Mapping

In the original paper of (Demszky et al., 2020)[14], the authors provide the hierarchical clustering results of the 27 emotions (Figure 2 in their paper), which reflect the nested structure of their proposed emotion taxonomy. Based on the clustering results, we merged the emotions that are highly correlated with each other, and the mapping between our adopted emotions and the original emotions is shown in Table 6.

Ours	**Original**
admiration	admiration, pride
anger	anger, annoyance, disgust, disapproval
approval	approval, realization
caring	caring, desire, optimism
fear	fear, nervousness
gratitude	gratitude, relief
joy	joy, amusement, excitement, love
sadness	sadness, disappointment, embarrassment, grief, remorse
surprise	surprise, confusion, curiosity

Table 6: Mapping between our adopted emotions and the original emotions in (Demszky et al., 2020).

B Statistics of Annotation

We computed the proportions of the last responses annotated with ER / IP / EX. In the Happy domain, the proportions are 76.0% / 10.2% / 18.7%, while in the Offmychest domain are 57.1% / 21.4% / 27.9% respectively. The statistics of DA and EM are shown in Figure 5.

We can find several differences between two domains. In terms of **communication mechanism**, the responses in the Offmychest domain prefer *interpretation* and *exploration*, while *emotional reaction* occupies a larger proportion in the Happy domain. In terms of **DA**, the actions that provide support (such as *agreeing*, *consoling*, *suggesting*, and *sympathizing*) are more frequently adopted in the Offmychest domain. It is similar when it comes to **emotion**, where the emotions such as *approval* and *caring* are displayed more commonly when responding to the posts with negative sentiments. We also observed that the responses in the Offmychest domain may also display the emotions like *anger* and *sadness*, indicating that they do understand the experiences and feelings of the conversation partners.

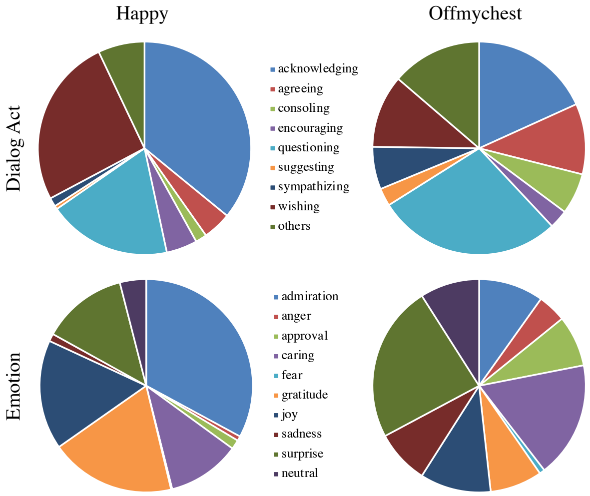

Figure 5: Statistics of the annotation results of DA and EM on the two domains.

---

[14] https://arxiv.org/abs/2005.00547v2

UniKeyphrase: A Unified Extraction and Generation Framework for Keyphrase Prediction

Huanqin Wu[1*], **Wei Liu**[1*], **Lei Li**[2], **Dan Nie**[1], **Tao Chen**[1], **Feng Zhang** [1], **Di Wang** [1]

[1]Tencent AI Platform Department, China
[2]Beijing University of Posts and Telecommunications
{huanqinwu,thinkweeliu,kathynie,vitochen,jayzhang,diwang}@tencent.com
leili@bupt.edu.cn

Abstract

Keyphrase Prediction (KP) task aims at predicting several keyphrases that can summarize the main idea of the given document. Mainstream KP methods can be categorized into purely generative approaches and integrated models with extraction and generation. However, these methods either ignore the diversity among keyphrases or only weakly capture the relation across tasks implicitly. In this paper, we propose **UniKeyphrase**, a novel end-to-end learning framework that jointly learns to extract and generate keyphrases. In UniKeyphrase, stacked relation layer and bag-of-words constraint are proposed to fully exploit the latent semantic relation between extraction and generation in the view of model structure and training process, respectively. Experiments on KP benchmarks demonstrate that our joint approach outperforms mainstream methods by a large margin.

1 Introduction

Keyphrases are several phrases that highlight core topics or information of a document. Given a document, the KP task focuses on automatically obtaining a set of keyphrases. As a basic NLP task, keyphrase prediction is useful for numerous downstream NLP tasks such as summarization (Wang and Cardie, 2013; Pasunuru and Bansal, 2018), document clustering (Hulth and Megyesi, 2006), information retrieval (Kim et al., 2013).

Keyphrases of a document fall into two categories: *present keyphrase* that appears continuously in the document, and *absent keyphrase* which does not exist in the document. Figure 1 shows an example of a document and its keyphrases. Traditional KP methods are mainly extractive, which have been extensively researched in past decades (Witten et al., 2005; Nguyen and Kan, 2007; Medelyan et al., 2009; Lopez and Romary, 2010; Zhang et al., 2016; Alzaidy et al., 2019; Sun et al., 2020). These methods aim to select text spans or phrases directly in the document, which show promising results on present keyphrase prediction. However, extractive methods cannot handle the absent keyphrase, which is also significant and requires a comprehensive understanding of document.

Document: On selecting an optimal wavelet for detecting singularities in traffic and vehicular data. …… applications of wavelet transform s (wts) in traffic engineering have been introduced however , …… , second order difference , oblique cumulative curve , and short time fourier transform) . it then mathematically describes wts ability to detect singularities in traffic data . …… , it is shown that selecting a suitable wavelet largely depends on the specific research topic , and that the mexican hat wavelet generally gives a satisfactory performance in detecting singularities in traffic and vehicular data .

Present keyphrases: { wavelet transform, oblique cumulative curve, short time fourier, the mexican hat wavelet }

Absent keyphrases: { singularity detection, traffic data analysis }

Figure 1: An example of an input document and its expected keyphrases. Blue and red denote present and absent keyphrases, respectively.

To mitigate this issue, several generative methods (Meng et al., 2017; Chen et al., 2018; Ye and Wang, 2018; Wang et al., 2019; Chen et al., 2019b; Chan et al., 2019; Zhao and Zhang, 2019; Chen et al., 2020; Yuan et al., 2020) have been proposed. Generative methods mainly adopt the sequence-to-sequence (seq2seq) model with a copy mechanism to predict a target sequence, which is concatenated of present and absent keyphrases. Therefore, the generative approach can predict both kinds of keyphrases. But these methods treat present and absent keyphrases equally, while these two kinds of keyphrase actually have different semantic properties. As illustrated in Figure 1, all the present keyphrases are specific techniques, while the absent keyphrases are tasks or research areas.

Thus several integrated methods (Chen et al., 2019a; Ahmad et al., 2020) try to perform multi-

---

[*] Equal contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 825–835
August 1–6, 2021.

task learning on present keyphrase extraction (PKE) and absent keyphrase generation (AKG). By treating present and absent keyphrase prediction as different tasks, integrated methods clearly distinguish the semantic properties for these two kinds of keyphrases. But integrated models suffer from two limitations. Firstly, these approaches are not trained in an end-to-end fashion, which causes error accumulation in the pipeline. Secondly, integrated methods just adopt a bottom shared encoder to implicitly capture the latent semantic relation between PKE and AKG, while this relation is essential for the KP task. As illustrated in Figure 1, the ground truth of PKE are specific techniques, which are all used for the "singularity detection" task in the "traffic data analysis" area. Such semantic relation between PKE and AKG can bring benefits for KP. Actually, semantic relations like "technique-task-area" between two tasks are common in the KP task. However, these integrated methods are weak at modeling it.

To address these issues, we propose a novel end-to-end joint model, UniKeyphrase, which adopts a unified pretrained language model as the backbone and is fine-tuned with both PKE and AKG tasks. What's more, UniKeyphrase explicitly captures the mutual relation between these two tasks, which brings benefits for keyphrase prediction: present keyphrases can provide an overall sense about salient parts of the document for AKG, and absent keyphrases viewed as high-level latent topics of the document can also supply PKE with global semantic information.

Specifically, UniKeyphrase employs two mechanisms to capture the relation from model structure and training process, respectively. Firstly, stacked relation layer is applied to repeatedly fuse PKE and AKG task representations to explicitly model the relation between the two sub-tasks. In detail, we adopt a co-attention based relation network to model the co-influence. Secondly, a bag-of-words constraint is designed for UniKeyphrase, which aims to provide some auxiliary global information of the whole keyphrases set during training.

Experiments conducted on the widely used public datasets show that our method significantly outperforms mainstream generative and integrative models. The contributions of this paper can be summarized as follows:

- We introduce a novel end-to-end framework UniKeyphrase for unified PKE and AKG.
- We design stacked relation layer (SRL) to explicitly capture the relation between PKE and AKG.
- We propose bag-of-words constraint (BWC) to explicitly feed global information about present and absent keyphrases to the model.

2 Related Works

2.1 Keyphrase Extraction

Most existing extraction approaches can be categorized into two-step extraction methods and sequence labeling approaches. Two-step extraction methods first identify a set of candidate phrases from the document by heuristics, such as essential n-grams or noun phrase (Hulth, 2003). Then, the candidate keyphrases are sorted and ranked to get predicted results. The scores can be learned by either supervised algorithms (Nguyen and Kan, 2007; Medelyan et al., 2009; Lopez and Romary, 2010) or unsupervised graph ranking methods (Mihalcea and Tarau, 2004; Wan and Xiao, 2008). For sequence labeling approaches, documents are fed to an encoder then the model learns to predict the likelihood of each word being a keyphrase (Zhang et al., 2016; Alzaidy et al., 2019; Sun et al., 2020).

2.2 Keyphrase Generation

Keyphrase generation focuses on predicting both present and absent keyphrases. Meng et al. (2017) first propose CopyRNN which is a seq2seq framework with attention and copy mechanism. Then a semi-supervised method for the exploitation of the unlabeled data is investigated by Ye and Wang (2018). Chen et al. (2018) employ a review mechanism to reduce duplicates. Chen et al. (2019b) focus on leveraging the title information to improve keyphrases generation. The latent topics of the document are exploited to enrich features by Wang et al. (2019). Zhao and Zhang (2019) utilize linguistic constraints to prevent model from generating overlapped phrases. Chan et al. (2019) introduce a reinforcement learning approach for keyphrase generation. Chen et al. (2020) propose an exclusive hierarchical decoding framework to explicitly model the hierarchical compositionality of a keyphrase set. Yuan et al. (2020) introduce a new model to generate multiple keyphrases as delimiter-separated sequences.

2.3 Integrated Methods

To explicitly distinguish the present and absent keyphrases, integrated extraction and generation approach have been applied to the KP task. Chen et al. (2019a) aim at improving the performance of the generative model by using an extractive model. Ahmad et al. (2020) propose SEG-Net, a neural keyphrase generation model that is composed of a selector for selecting the salient sentences in a document, and an extractor-generator that extracts and generates keyphrases from the selected sentences. In contrast to these methods, our joint approach can explicitly capture the relation between extraction and generation in an end-to-end framework.

3 Approach

In this section, we describe the architecture of UniKeyphrase. Figure 2 gives an overview of UniKeyphrase, which consists of three components: extractor-generator backbone based on UniLM, a stacked relation layer for capturing the relation between PKE and AKG, and bag-of-words constraint for considering the global view of two tasks in training. In the following sections, the details of UniKeyphrase are given.

3.1 Extractor-Generator Backbone

Given a document $\mathbf{X} = \{x_1, ..., x_m\}$, KP aims at obtaining a keyphrase set $\mathbf{K} = \{k_1, ..., k_{|K|}\}$. Naturally, $\mathbf{K}$ can be divided into present keyphrase set $\mathbf{K}_p = \{k^p_1, ..., k^p_{|K_p|}\}$ and absent keyphrase set $\mathbf{K}_a = \{k^a_1, , ..., k^a_{|K_a|}\}$ by judging whether keyphrases appear exactly in the source document. UniKeyphrase decomposes the KP into PKE and AKG, and jointly learns two tasks in an end-to-end framework.

UniKeyphrase treats PKE as a sequence labeling task and AKG as a text generation task. To jointly learn in an end-to-end framework, UniKeyphrase adopts UniLM (Dong et al., 2019) as the backbone network. UniLM is a pre-trained language model, which can perform sequence-to-sequence prediction by employing a shared transformer network and utilizing specific self-attention masks to control what context the prediction conditions on.

As shown in Figure 2, with a pre-trained UniLM layer, the contextualized representation for the source document can attend to each other from both directions, which is convenient for PKE. While the representation of the target token can only attend to the left context, as well as all the tokens in the source document, which can be easily adapted to AKG.

Specifically, for a document $\mathbf{X}$, all absent keyphrases will be concatenated as a sequence. Then we randomly choose tokens in this sequence, and replace them with the special token [MASK]. The masked sequence is defined as $\mathbf{K}^m_a$. We further concatenate document $\mathbf{X}$ and $\mathbf{K}^m_a$ with [CLS] and [SEP] tokens as the input sequence:

$$\mathbf{I} = \{[\text{CLS}]\, \mathbf{X}\, [\text{SEP}]\, \mathbf{K}^m_a\, [\text{SEP}]\} \quad (1)$$

Afterwards, we feed input sequence into UniLM and obtain output hidden state $\mathbf{H}$:

$$\mathbf{H} = \text{UNILM}(\mathbf{I}) \quad (2)$$

the hidden state $\mathbf{H} = \{h_1, ..., h_T\}$ (T is the number of input tokens in the UniLM) will be used as the input of stacked relation layer for jointly modeling PKE and AKG.

3.2 Stacked Relation Layer

Based on the UniLM, we can obtain the output hidden $\mathbf{H}$. Instead of directly using the UniLM hidden for PKE and AKG, we use the SRL to explicitly model the relation between these two tasks. Actually, modeling the cross-impact and interaction between different tasks in joint model is a common problem (Qin et al., 2020a,b, 2019).

Specifically, SRL takes the initial shared representations $\mathbf{P}^0 = \mathbf{A}^0 = \{h_1, ..., h_T\}$ as input and aims to obtain the finally task representations $\mathbf{P}^L$ and $\mathbf{A}^L$ (L is the number of stacked layers), which consider the cross-impact between PKE and AKG. Besides, SRL can be stacked to repeatedly fuse PKE and AKG task representations for better capturing mutual relation.

Formally, given the l^{th} layer inputs $\mathbf{P}^l = \{p^l_1, ..., p^l_T\}$ and $\mathbf{A}^l = \{a^l_1, ..., a^l_T\}$, stacked relation layer first apply two linear transformations with a ReLU activation over the input to make them more task-specific, which can be written as follow:

$$\mathbf{P}^{l'} = \text{LN}(\mathbf{P}^l + \max(0, \mathbf{W}^l_P \mathbf{P}^l + \mathbf{b}^l_P)) \quad (3)$$

$$\mathbf{A}^{l'} = \text{LN}(\mathbf{A}^l + \max(0, \mathbf{W}^l_A \mathbf{A}^l + \mathbf{b}^l_A)) \quad (4)$$

where LN represent the layer normalization function (Ba et al., 2016).

Then the relation between the two tasks will be integrated base on task-specific representations. In this paper, we adopt co-attention relation networks.

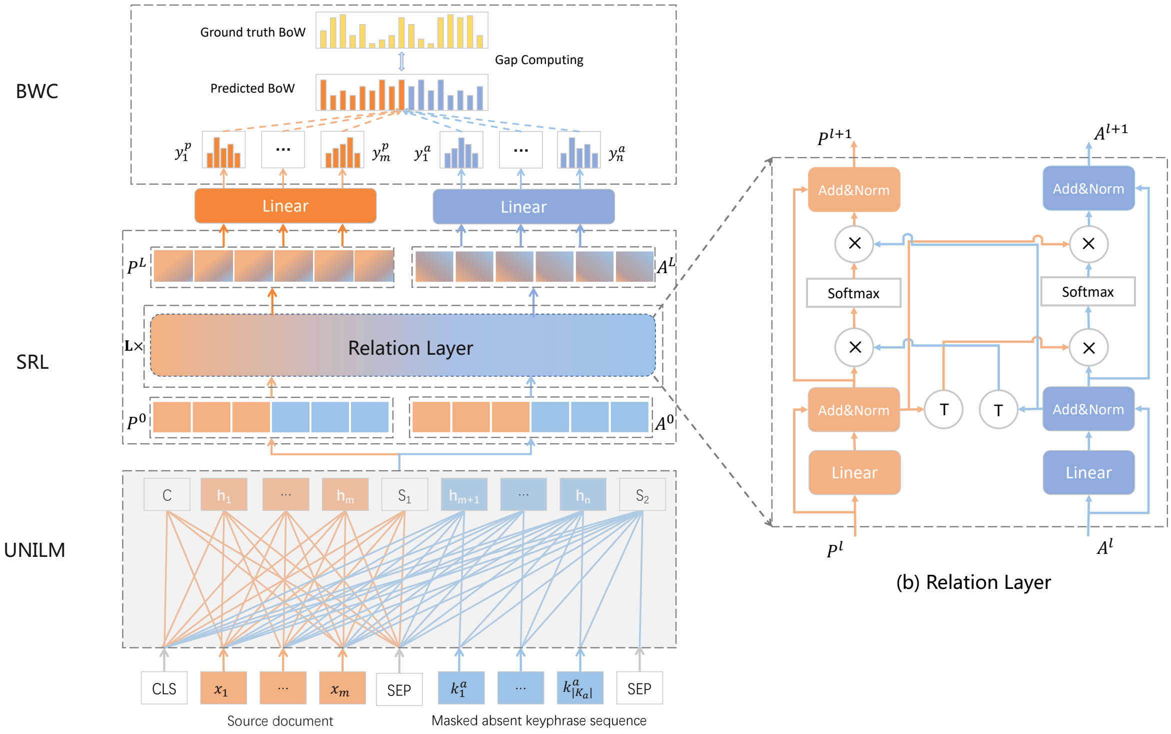

(b) Relation Layer

(a) UniKeyphrase

Figure 2: The architecture of our model

Co-Attention is an effective approach to model the important information of correlated tasks. We extend the basic co-attention mechanism from token level to task representations level. It can produce the PKE and AKG task representations considering each other. Therefore, we can transfer useful mutual information between two tasks. The process can be formulated as follows:

$$\mathbf{P}^{l+1} = \mathrm{LN}(\mathbf{P}^{l'} + \mathrm{softmax}(\mathbf{P}^{l'}(\mathbf{A}^{l'})^{\top})\mathbf{A}^{l'}) \quad (5)$$

$$\mathbf{A}^{l+1} = \mathrm{LN}(\mathbf{A}^{l'} + \mathrm{softmax}(\mathbf{A}^{l'}(\mathbf{P}^{l'})^{\top})\mathbf{P}^{l'}) \quad (6)$$

where $\mathbf{P}^{l+1} = \{p_1^{l+1}, ..., p_T^{l+1}\}$ and $\mathbf{A}^{l+1} = \{a_1^{l+1}, ..., a_T^{l+1}\}$ are the l^{th} layer updated representations.

After stacked relation layer, we can obtain the outputs $\mathbf{P}^L = \{p_1^L, ..., p_m^L\}$ and $\mathbf{A}^L = \{a_1^L, ..., a_n^L\}$. We then adopt separate decoders to perform PKE and AKG by using the task representations of corresponding position , which can be denoted as follows:

$$\mathbf{y}_i^p = \mathrm{softmax}(\mathbf{W}^p p_i^L + \mathbf{b}^p) \quad (7)$$

$$\mathbf{y}_j^a = \mathrm{softmax}(\mathbf{W}^a a_j^L + \mathbf{b}^a) \quad (8)$$

where $\mathbf{y}_i^p$ and $\mathbf{y}_j^a$ are the predicted distribution for present keyphrase and absent keyphrase respectively; $\mathbf{W}^p$ and $\mathbf{W}^a$ are transformation matrices; $\mathbf{b}_p$ and $\mathbf{b}_a$ are bias vectors.

3.3 Bag-of-Words Constraint

UniKeyphrase divides the KP task into two subtasks, PKE and AKG. These two sub-tasks are optimized separately, which lacks the awareness of global information about the total keyphrase set. Such global information can be the amount of all keyphrases or the common words between present and absent keyphrases. Bag of words (BoW) is a suitable medium for describing this information. In this paper, we feed global information to UniKeyphrase by constructing constraints based on the BoW of keyphrases. The word count in BoW can provide guidance about task relation for PKE and AKG training in a global view.

Specifically, we calculate the gap between the model predicted keyphrase BoW and ground truth keyphrase BoW, then add it into the loss. Hence UniKeyphrase can get a global view of keyphrases allocation and adjust two tasks during training.

We first collect present and absent keyphrase BoW from model. For present keyphrases, since PKE is a sequence labeling task, we collect all words that labeled as keyphrases, and construct present predicted BoW V^p. We use the sum of corresponding label probabilities as the count of word w in V^p:

$$V^p(w) = \sum_{i \in \mathcal{I}_w} max(\mathbf{y}_i^p) \tag{9}$$

where y_i^p denotes all predicted label probabilities at time step i. $\mathcal{I}_w$ is all position of word w in document. Maximum operation is used for selecting the probability of predicted label. For absent keyphrase, the generation probability of all steps are accumulated as predicted absent BoW $V^a(w)$.

$$V^a(w) = \sum_{j=1}^{N} \mathbf{y}_j^a(w) \tag{10}$$

After acquiring the predicted present and absent keyphrase BoW, we concatenate these two parts as the total predicted BoW V, then calculate the error compared with ground truth BoW $\hat{V}$. To reserve the word count information, we use Mean Square Error (MSE) function:

$$\mathcal{L}_{BoW} = \frac{1}{|\mathcal{V}|} \sum_{w \in \mathcal{V}} (V(w) - \hat{V}(w))^2 \tag{11}$$

It is worth noting that $\mathcal{V}$ is the collection of words that make up the ground truth keyphrases and predicted keyphrases. So the BWC only affects a small subset of the whole vocabulary for each sample. This can help reduce the noise and stabilize the training process.

In practice we increase the weight of BWC logarithmically from zero to a defined maximum value w_m, the weight of BWC on t step can be denoted as follows:

$$w_{BoW}(t) = log(\frac{e^{w_m} - 1}{t_{total}} t + 1) \tag{12}$$

where t_{total} is the total step of training. The reason to adjust the weight is the same as Ma et al. (2018). The BWC should take effect when predicted results are good enough. Therefore we first assign a small weight to BWC at the initial time, and gradually increase it when training.

3.4 Training

For the PKE task, objection is formulated as:

$$\mathcal{L}_{PKE} = -\sum_{i=1}^{M} \sum_{c=1}^{C} w_c \hat{\mathbf{y}}_i^{(c,p)} \log \left(\mathbf{y}_i^{(c,p)} \right) \tag{13}$$

where M refers to the length of document, C refers to the number of label, w_c is the loss weight for the positive label. $\hat{\mathbf{y}}_i^p$ refers the gold label.

For the AKG task, training objection is to maximize the likelihood of masked tokens, which is formulated as:

$$\mathcal{L}_{AKG} = -\sum_{i=1}^{N} \sum_{j=1}^{V_s} \hat{\mathbf{y}}_i^{(j,a)} \log \left(\mathbf{y}_i^{(j,a)} \right) \tag{14}$$

where N refers to the number of masked tokens, V_s refers to the size of vocabulary. $\hat{\mathbf{y}}_i^a$ refers the ground-truth word.

Considering the BWC, the overall loss of UniKeyphrase is formulated as:

$$\mathcal{L} = \mathcal{L}_{PKE} + \mathcal{L}_{AKG} + w_{BoW} \mathcal{L}_{BoW} \tag{15}$$

4 Experiments

4.1 Datasets and Evaluation

We follow the widely used setup of the deep KP task: train, validation and test on the `KP20K` (Meng et al., 2017) dataset, and give evaluation on three more benchmark datasets: `NUS` (Nguyen and Kan, 2007), `INSPEC` (Hulth, 2003) and `SEMEVAL` (Kim et al., 2010). We follow the pre-process, post-process, and evaluation setting of Meng et al. (2017, 2019); Yuan et al. (2020)[1]. Specifically, we use the partition of present and absent provided by Meng et al. (2017) and calculate F_1@5 and F_1@M (use all predicted keyphrases for F_1 calculation) after stemming and removing duplicates.

4.2 Experimental Setup

Setting: We reuse most hyper-parameters from pre-trained UNILM[2]. The layer number of SRL is set to 2. We use $w_m = 1.0$ when adjusting the weight of BWC. PKE loss weights w_c for the positive label is set to 5.0. we set batch size to 256, and maximum length to 384. During decoding, we use beam search for AKG, and beam size is set as 5. We train our model on the training set for 100 epochs. It takes about 40 minutes per epoch to train UniKeyphrase on 8 Nvidia Tesla V100 GPU cards with mixed-precision training. More details are provided in Appendix B.

---

[1] we follow the official GitHub repository to prepare datasets and evaluation scripts which are available on https://github.com/memray/OpenNMT-kpg-release.

[2] we use the official provided pre-trained model, which is available on https://unilm.blob.core.windows.net/ckpt/unilm1-base-cased.bin.

Type	Model	KP20k		NUS		SemEval		Inspec	
		F_1@5	F_1@M	F_1@5	F_1@M	F_1@5	F_1@M	F_1@5	F_1@M
Generative	CatSeq	29.1	36.7	32.3	39.7	24.2	28.3	22.5	26.2
	CatSeqTG	29.2	36.6	32.5	39.3	24.6	29.0	22.9	27.0
	CatSeq(TRM)	29.1	36.8	32.8	40.5	24.5	28.8	22.5	26.4
	CatSeqD	28.5	36.3	32.1	39.4	23.3	27.4	21.9	26.3
	ExHiRD-h	31.1	37.4	–	–	28.4	33.5	25.3	29.1
Integrated	KG-KE-KR-M	31.7	–	28.9	–	20.2	–	25.7	–
	SEG-NET	32.3	38.1	40.1	**45.9**	29.8	34.1	24.6	30.1
Joint	UniKeyphrase	**40.8**	**42.8**	**43.4**	43.5	**41.6**	**40.9**	**29.0**	**31.1**

Table 1: Results on present keyphrase prediction.

Baselines: We compare two kinds of strong baselines (generative, integrated) to give a comprehensive evaluation on the performance of UniKeyphrase.

- **Generative**: Generative models can predict both present and absent keyphrases under the seq2seq framework. CatSeq (Yuan et al., 2020) is the classic setting of keyphrase seq2seq model. We report the performance of CatSeq and various improved models on it, including CatSeqTG (Chen et al., 2019b), CatSeq (TRM) (Ahmad et al., 2020) and CatSeqD (Yuan et al., 2020). A recently released model is also included for comparing, which is ExHiRD-h (Chen et al., 2020).
- **Integrated**: Integrated model often combine multiple modules to perform extractive and abstractive tasks. But they are not end-to-end. Two latest integrated models are recorded for comparison. including KG-KE-KR-M (Chen et al., 2019a) and SEG-NET (Ahmad et al., 2020)

4.3 Main Results

In this section, we show the experimental results of the baseline methods and our model on present keyphrase extraction and absent keyphrase generation. Besides, we also study the average number of unique predicted keyphrases per document to further show the advantages of our model.

4.3.1 Present and Absent Keyphrase Prediction

The present and absent keyphrase prediction performance of all methods are shown in Table 1 and Table 2. From the results, we can find that our joint framework outperforms most state-of-the-art generative baseline by a significant margin, which demonstrates the effectiveness of our UniKeyphrase. We notice that the UniKeyphrase does not outperform the SEG-NET on F_1@M for present keyphrase extraction on `NUS` dataset. One potential reason is that the source document length and sentence number of `NUS` are much larger than the KP20k training set. SEG-NET employs an additional sentence selector for filtering sentences without keyphrase, which is more adaptable for processing long documents like `NUS`.

4.3.2 Number of Predicted Keyphrases

The number of predicted keyphrases indicates the model's understanding of input documents. From the previous work (Chen et al., 2020), we find the average number of unique predicted keyphrases per document is much lower than the gold average keyphrase number in most datasets. The number of unique keyphrases predicted by UniKeyphrase and baselines is compared in Table 3. We can find that UniKeyphrase predicts more (especially in absent keyphrases) than baseline methods, which is closer to ground truth. Meanwhile, we find UniKeyphrase leads to predict more keyphrases than the ground-truth (especially on KP20k). We leave solving the over prediction keyphrases problem as our future work.

4.4 Ablation Study

In this section, we check the improvement brought by SRL and BWC. Several ablation experiments are conducted to analyze the effect of different components. The ablation experiment on `SemEval` is shown in Table 4. The results show the effectiveness of different components of our method to the final performance.

Effectiveness of stacked relation layer: In this setting, we conduct experiments on the multi-task framework where PKE and AKG promote each other only by the hidden state of UniLM, From

---

[3] Reports from Yuan et al. (2020), which do not report absent metrics for this model. The original paper also does not give detailed numbers.

Type	Model	KP20k		NUS		SemEval		Inspec	
		F_1@5	F_1@M	F_1@5	F_1@M	F_1@5	F_1@M	F_1@5	F_1@M
Generative	CatSeq	1.5	3.2	1.6	2.8	2.0	2.8	0.4	0.8
	CatSeqTG	1.5	3.2	1.1	1.8	1.9	2.7	0.5	1.1
	CatSeq(TRM)	1.5	3.1	1.1	1.8	1.9	2.7	0.5	0.9
	CatSeqD	1.5	3.1	1.5	2.4	1.6	2.4	0.6	1.1
	ExHiRD-h	1.6	3.2	–	–	1.7	2.5	1.1	2.2
Integrated	KG-KE-KR-M[3]	–	–	–	–	–	–	–	–
	SEG-NET	2.0	3.8	1.4	2.4	2.1	3.1	0.9	1.4
Joint	UniKeyphrase	**4.7**	**4.7**	**3.6**	**3.7**	**3.0**	**3.2**	**2.9**	**2.9**

Table 2: Results on absent keyphrase prediction.

Model	**Inspec**		**SemEval**		**KP20k**	
	#PK	#AK	#PK	#AK	#PK	#AK
Ground Truth	7.64	2.10	6.28	8.12	3.32	1.93
Transformer	3.17	0.70	3.24	0.67	3.44	0.58
catSeq	3.33	0.58	3.45	0.64	3.70	0.51
catSeqD	3.33	0.58	3.47	0.63	3.74	0.50
catSeqCorr	3.07	0.53	3.15	0.62	**3.36**	0.50
ExHiRD-h	4.00	**1.50**	3.65	0.99	3.97	0.81
SEG-NET	-	-	-	-	3.79	1.14
UniKeyphrase	**5.19**	2.74	**8.15**	**3.04**	6.29	**2.72**

Table 3: Results of average numbers of predicted unique keyphrases. "#PK" and "#AK" are the number of present and absent keyphrases respectively. **Bold** denotes the prediction closest to the ground truth.

Model	Present		Absent	
	F_1@5	F_1@M	F_1@5	F_1@M
UniKeyphrase	41.6	40.9	3.0	3.2
w/o SRL	38.5	37.6	2.9	3.1
w/o BWC	40.0	39.5	2.8	2.8

Table 4: Ablation study on SemEval dataset

the result, we can see that the performance drops both in present keyphrase and absent keyphrase without stacked relation layer. This demonstrates that explicitly modeling the relation between PKE and AKG with stacked relation layer can benefit them effectively.

Effectiveness of bag-of-words constraint: In this setting, we remove our bag-of-words constraint and there is no global constraint for two tasks. The results show a drop in KP performance, indicating that capturing the global constraint of the result by BWC is effective and important for our method.

4.5 Analysis

4.5.1 SRL Analysis

To better understand the SRL module, we analyze the impact of stacked layers and give a visualization of the inner state of SRL.

Analysis of SRL Layer Number: We explore the impact of the stack number of relation network. The comparison of total keyphrase prediction result, which regardless of the present or absent of keyphrases, are shown in Table 5. We can find that setting deeper layers could generally result in better performance when the number of stacked layers is less than three, which proves the effectiveness of stacked layers. It is worth noting that when the number of stacked layers is larger than two, the KP performance drops. We suppose that when the relation network becomes deeper, the over-interaction will lose the diversity of two task representations.

Model	Total F_1@M
UniLM based keyphrase generation	23.7
UniKeyphrase with 0 layer SRL	23.1
UniKeyphrase with 1 layer SRL	25.2
UniKeyphrase with 2 layer SRL	**25.3**
UniKeyphrase with 3 layer SRL	24.5

Table 5: Total keyphrase prediction on SemEval dataset by different setting. UniKeyphrase with 0 layer SRL means UniKeyphrase without SRL module.

Visualization Analysis for SRL: To better understand what the SRL network has learned, we compare the distance between the PKE representation and AKG representation in different settings. In detail, we randomly sample 2000 pairs of PKE representation vector and AKG representation vector on different positions from test data and compute euclidean metric in each pair. As shown in Figure 3, the blue points mean the Euclidean metric between PKE and AKG representation vector without SRL layer, while the yellow points mean the Euclidean metric with SRL layer.

From the Figure 3, we can find that the blue points are under the yellow points, which means the PKE and AKG representation vector without SRL is more similar. In other words, SRL has learned the task-specific representation. Also, the blue points are denser than the yellow points, which

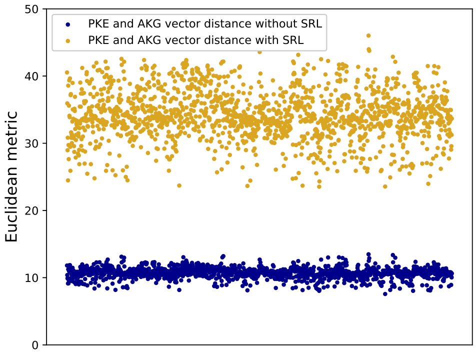

Figure 3: Distance between PKE representation and AKG representation on different settings.

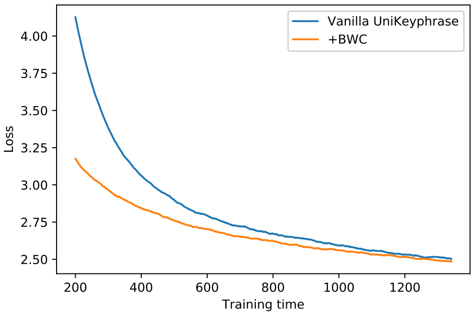

Figure 4: BWC's influence on total training loss (sequence labeling + text generation).

means the PKE and AKG representation with SRL is more diverse than the one without SRL on different samples.

4.5.2 BWC Analysis

Loss Compare: From Figure 4 we can see that the original total loss (labeling and generation) drops more with the help of BWC compared to the vanilla model. BWC actually is an enhancement on the original supervised signal from a global view. It guides the model to learn how many to predict and how to allocate present and absent keyphrases, while original loss only teaches what to predict in each position.

Bag-of-words Error: We also calculate the bag-of-words Error between ground truth and model predicted keyphrases, which is how many tokens are incorrectly predicted. As shown in Figure 5, UniKeyphrase with BWC achieves lower BoW Error compared with the vanilla model. It proves that BWC successfully guides the model to learn a better BoW allocation.

4.5.3 Joint Framework Analysis

In our UniKeyphrase model, we adopt pre-trained model UNILM for KP. So it is necessary to check

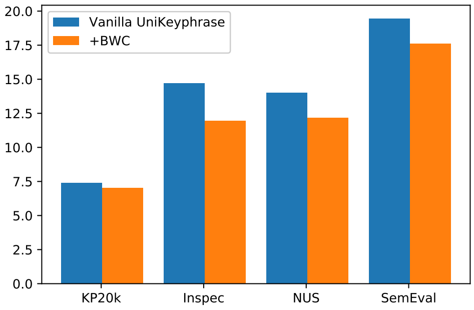

Figure 5: Bag-of-words Error comparison between vanilla and BWC.

that the gain on metrics of our proposed joint framework is not just come from the pre-trained model. In this section, we compare UniKeyphrase with directly using the pre-trained UNILM to perform generative KP.

Specifically, we train a sequence to sequence model for KP based on UNILM. Results are shown in Table 5. From the results, we find that all of the joint models with SRL can further outperform the generative method based on UNILM, demonstrating that the improvement of KP mainly come from our joint framework instead of pre-trained UNILM. We notice that the UniKeyphrase without SRL does not outperform the generative method based on UNILM, which show the significance of modeling the relation between the two sub-tasks in our joint framework.

5 Conclusion and Future Work

This paper focuses on explicitly establishing an end-to-end unified model for PKE and AKG. Specifically, we propose UniKeyphrase, which contains stacked relation layer to model the interaction and relation between the two sub-tasks. In addition, we design a novel bag-of-words constraint for jointly training these two tasks. Experiments on benchmarks show the effectiveness of the proposed model, and more extensive analysis further confirms the correlation between two tasks and reveals that modeling the relation explicitly can boost their performance.

Our UniKeyphrase can be formalized as a unified framework of NLU and NLG tasks. It is easy to transfer it to other extraction-generation NLP tasks. In the future, we will explore to adopt our framework to more scenarios.

Acknowledgments

Lei Li were supported by Beijing Municipal Commission of Science and Technology [grant number Z181100001018035]; Engineering Research Center of Information Networks, Ministry of Education; BUPT Jinan Institute; Beijing BUPT Information Networks Industry Institute Company Limited.

References

Wasi Uddin Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2020. Select, extract and generate: Neural keyphrase generation with syntactic guidance. *arXiv preprint arXiv:2008.01739*.

Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The world wide web conference*, pages 2551–2557.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066.

Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019a. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2846–2856.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019b. Title-guided encoding for keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Anette Hulth and Beata Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 537–544.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26.

Youngsam Kim, Munhyong Kim, Andrew Cattle, Julia Otmakhova, Suzi Park, and Hyopil Shin. 2013. Applying graph-based keyword extraction to document retrieval. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 864–868.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 248–251.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 332–338.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 1318–1327.

Rui Meng, Xingdi Yuan, Tong Wang, Peter Brusilovsky, Adam Trischler, and Daqing He. 2019. Does order matter? an empirical study on generating multiple keyphrases as a sequence. *arXiv preprint arXiv:1909.03590*.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653.

Libo Qin, Wanxiang Che, Yangming Li, Mingheng Ni, and Ting Liu. 2020a. Dcr-net: A deep co-interactive relation network for joint dialog act recognition and sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8665–8672.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2078–2087, Hong Kong, China. Association for Computational Linguistics.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020b. AGIF: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1807–1816, Online. Association for Computational Linguistics.

Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert. *arXiv preprint arXiv:2004.13639*.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405.

Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R Lyu, and Shuming Shi. 2019. Topic-aware neural keyphrase generation for social media language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2516–2526.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI global.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuan-Jing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 836–845.

Jing Zhao and Yuxiang Zhang. 2019. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5224–5233.

A Dataset Statistics

Type	Dataset	#Examples	Max/Avg #Tokens	Max/Avg #Sentences
Test	Inspec	500	387.0 / 138.4	27.0 / 6.7
	NUS	211	384.0 / 185.6	16.0 / 8.4
	SemEval	100	415.0 / 208.0	18 / 8.8
	KP20k	20000	1116.0 / 178.9	70.0 / 8.1
Validation	KP20k	20000	1862.0 / 179.2	120 / 8.2
Train	KP20k	514154	2924 / 177.9	284 / 8.2

Table 6: Summary of the dataset used in experiments. "#Examples" means the number of sample. "#Tokens" means the number of token. "#Sentences" means the number of sentence.

Relevant statistics about the dataset used in this paper is shown in Table 6.

B Experimental Details

The BWC does not bring extra parameters, hence the trainable parameters of UniKeyphrase come from UNILM and SRL. We use the base version of UNILM, which contains about 110M parameters. Follow UNILM, our model is implemented using PyTorch. The learning rate is 1e-5 and the proportion of warmup steps is 0.1. The masking probability of absent keyphrase sequence is 0.7. For the SRL module, dropout is applied to the output of each layer for regularization, the dropout rate is 0.5. In this paper, we try to set the number of layer by 2,3,4 and choose the best based on validation. For all experiments in this paper, we choose the model that performs best on the KP20k validation dataset.

C Preprocess

The input of UniKeyphrase is the same as BERT, which applies wordpiece tokenizer on raw sentences. So we use the "BIXO" labeling method, where B and I stand for Beginning and Inside of a word in keyphrases, and O denotes any token that Outside of any keyphrase. For any sub-word token in keyphrases(which starts with '##' in processed input) we use X to label it. For example, "voip conferencing system" will be tokenized into "v ##oi ##p con ##fer ##encing system" and be labeled as "B X X I X X I". We concatenate all the tokenized absent keyphrases into one sequence using a special delimiter " ; ". An example of absent keyphrase sequence will like "peer to peer ; content delivery ; t ##f ##rc ; ran ##su ##b".

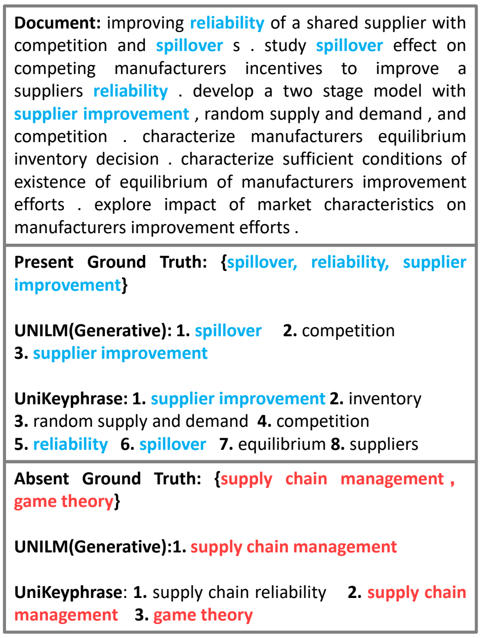

Document: improving reliability of a shared supplier with competition and spillover s . study spillover effect on competing manufacturers incentives to improve a suppliers reliability . develop a two stage model with supplier improvement , random supply and demand , and competition . characterize manufacturers equilibrium inventory decision . characterize sufficient conditions of existence of equilibrium of manufacturers improvement efforts . explore impact of market characteristics on manufacturers improvement efforts .

Present Ground Truth: {spillover, reliability, supplier improvement**}**

UNILM(Generative): 1. spillover **2.** competition **3.** supplier improvement

UniKeyphrase: 1. supplier improvement **2.** inventory **3.** random supply and demand **4.** competition **5.** reliability **6.** spillover **7.** equilibrium **8.** suppliers

Absent Ground Truth: {supply chain management , game theory**}**

UNILM(Generative):1. supply chain management

UniKeyphrase: **1.** supply chain reliability **2.** supply chain management **3.** game theory

Figure 6: Case study.

D Case Study

We give a case on the KP20k testset in Figure 6. For fairness, we compare with keyphrase generation based on UNILM since our joint models are based on its implementation. Blue and red denote correct present and absent keyphrases, respectively. As shown in Figure 6, UniKeyphrase successfully catches the deep semantic relation similar to the case in the introduction, but here are "methods" for the absent and "application" for the present. Having caught this relation, UniKeyphrase gives more accurate results(predicts absent keyphrases "game theory").

E Evaluation Details

We use F_1@5 and F_1@M as evaluation metric. For calculating F_1@5, since there is no explicit rank score for each predicted keyphrase, we calculate the rank score as follows:

Present: we calculate the average predicted label probabilities of all tokens in a keyphrase as the score. We tried several other scoring strategies like max, min, or the first token's probability as the score. The results show no significant difference(less than 0.1%).

Absent: following previous works, we pick up the top 5 keyphrases in sequence order. The 5 leftmost keyphrases in the predicted sequence are selected as the result.

As Good as New. How to Successfully Recycle English GPT-2 to Make Models for Other Languages

Wietse de Vries
University of Groningen
The Netherlands
wietse.de.vries@rug.nl

Malvina Nissim
University of Groningen
The Netherlands
m.nissim@rug.nl

Abstract

Large generative language models have been very successful for English, but other languages lag behind, in part due to data and computational limitations. We propose a method that may overcome these problems by adapting existing pre-trained models to new languages. Specifically, we describe the adaptation of English GPT-2 to Italian and Dutch by retraining lexical embeddings without tuning the Transformer layers. As a result, we obtain lexical embeddings for Italian and Dutch that are aligned with the original English lexical embeddings. Additionally, we scale up complexity by transforming relearned lexical embeddings of GPT-2 small to the GPT-2 medium embedding space. This method minimises the amount of training and prevents losing information during adaptation that was learned by GPT-2. English GPT-2 models with relearned lexical embeddings can generate realistic sentences in Italian and Dutch. Though on average these sentences are still identifiable as artificial by humans, they are assessed on par with sentences generated by a GPT-2 model fully trained from scratch.

1 Introduction

Large pre-trained language models have brought unprecedented progress in NLP, but also concerns regarding the excessive computing power needed to train them (Strubell et al., 2019). Limited access to large amounts of computational resources, as well as environmental considerations, curb possibilities for less-resourced and less-researched languages. Additionally, models like GPT-2 (Radford et al., 2019) are trained on amounts of data that are not available for most languages. As a result of these limitations, language models are commonly trained for English, whereas reproductions in other languages may underperform or not exist.

That language models can benefit from information in other languages has been demonstrated by the effectiveness of multilingual BERT (mBERT) and XLM-RoBERTa (Conneau et al., 2020). However, for downstream tasks mBERT has been shown to be outperformed by monolingual models for higher resource languages whereas lower resource languages can still achieve better results without pre-trained language models (Nozza et al., 2020; Wu and Dredze, 2020).

Rather than pursuing a multilingual direction, we aim at exploiting existing language models and language similarities to create models for new languages. Specifically, we develop a multi-step procedure for adapting English GPT-2 (Radford et al., 2019) to Italian and Dutch. Dutch is genetically closely related to English, both being West-Germanic languages, while Italian is a more distant Romance language from the same Indo-European language family (Eberhard et al., 2020). It is however worth noticing that at sentence level English and Italian tend to have the same word order (SVO), while Dutch is SVO in main clauses, but SOV in subordinate ones; at noun phrase level, English and Dutch share constituent order (for example adjective-noun) while Italian is different (mostly noun-adjective). A GPT-2 based model has previously been trained from scratch for Italian (De Mattei et al., 2020). We can thus compare sentences generated by this model with sentences generated by our adapted model. For Dutch, no other GPT-2 based models exist, but similar BERT-based models have been trained from scratch (de Vries et al., 2019; Delobelle et al., 2020).

Procedure Overview and Contributions When training a new language model, weights of an existing pre-trained model for another language can be used for initialisation. The first step in our training procedure is to only retrain the lexical embeddings of the GPT-2 *small* model, without touching the Transformer layers. We show that retrained lexical embeddings are well aligned with the English vo-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 836–846
August 1–6, 2021.

cabulary and that GPT-2 is capable of generating realistic text in Italian and Dutch after this step. Next, we demonstrate that the lexical embeddings of larger GPT-2 models can be approximated by transforming the *small* lexical embeddings to the GPT-2 *medium* lexical embedding space. The least-squares regression method is the most effective transformation method for this scaling procedure. Human judgements show that generated sentences are often realistic, but become even more consistently so after additional finetuning of the Transformer layers. This improvement is stronger for Dutch than for Italian.

The steps in our pipeline yield GPT-2 based language models for Italian and Dutch which are made available on the Hugging Face model hub[1]; the source code is available on Github[2]. On the last page, we also include a 'recipe' for creating GPT-2 models for new languages.

2 Background

Previous and current research relevant for the present work is found in the more general field of transfer learning, with a specific focus on language transfer. We also discuss how our approach of translating lexical layers in different model sizes relates to work on aligning word embeddings.

2.1 Language transfer

Transfer learning can be an effective strategy to adapt models to lower-resource languages by initially training a model for a source language and then further training (parts of) the model for a target language. It has been successfully used to create machine translation models with little parallel data (Zoph et al., 2016) as well as other classic NLP tasks (Lin et al., 2019).

In machine translation a model can be adapted by initially training it for a high-resource language pair after which the model should be partially retrained for a low-resource language (Zoph et al., 2016; Nguyen and Chiang, 2017; Kocmi and Bojar, 2018). Retraining a randomly initialised lexical layer while freezing the rest of the model is an effective method to adapt a model to a new language, and dictionary based initialisation is not required to get the best performance (Zoph et al., 2016). Artetxe et al. (2020) show that a monolingual BERT model can be adapted from a source language to a different target language by retraining the lexical layer for the target language while freezing the Transformer layers in the model. Zero shot adaptation for downstream tasks is possible by finetuning the original source model with source language data and swapping lexical layers afterwards. Lexical layer retraining approaches may be effective despite the presence of source and target language dissimilarities if a downstream task does not require perfect data. However, these methods have not been applied yet to generative language models where dissimilarities can cause clear syntactic and lexical errors.

Language similarity plays a role in the effectiveness of transfer learning for language models. For instance, in machine translation French is a better parent model for Spanish than German (Zoph et al., 2016). Word order differences between languages can negatively influence transfer performance, and Kim et al. (2019) show that randomly swapping words in the source language, which forces the model to rely less on consistent word order, can improve performance in the target language. Overall, genetic similarity between source and target languages can play a role, but Lin et al. (2019) have shown that in practice the geographic distances between countries of origin, syntactic similarity and subword overlap are better predictors of transfer performance for machine learning, part-of-speech tagging, dependency parsing and entity linking.

2.2 Aligning word embeddings

Alignment of lexical embeddings, for example for multiple languages, is most prominently done with mapping-based approaches (Ruder et al., 2019). Typically, a function is determined that transforms one vector space to another based on a seed lexicon. This lexicon is a dictionary of anchor points that should result close together after transformation.

An influential method for learning a lexical embedding mapping is the least-squares linear transformation method by Mikolov et al. (2013). They observe that words and their translations in other languages show similar constellations of related words after such a transformation. An alternative method that is generally considered to be an improvement (Ruder et al., 2019) is the orthogonal procrustes solution. This method adds the constraint that the transformation matrix must be orthogonal. In practice this means that the transfor-

---

[1] `https://huggingface.co/GroNLP`

[2] `https://github.com/wietsedv/gpt2-recycle`

mation only contains rotations and reflections and no scaling and translation. This constraint enables length normalisation (Xing et al., 2015) and ensures monolingual invariance (Artetxe et al., 2016).

Mapping-based approaches rely on isomorphism, which means that a one-to-one token mapping between source and target lexical embedding spaces should be possible. This assumption is used for bilingual lexicon induction after alignment (Conneau et al., 2018). However, the isomorphism assumption highly depends on language similarity and (amount of) training data (Søgaard et al., 2018). Some more complex alignment methods like RCLS (Joulin et al., 2018) optimise for dictionary translation performance, which assumes isomorphism, but simpler methods like the orthogonal procrustes solution are more effective for downstream tasks like natural language inference (Glavaš et al., 2019). Mohiuddin et al. (2020) propose a solution to the isomorphism problem by learning a new shared embedding space with an auto-encoding neural model instead of trying to fit the embeddings of one language in the space of another language.

3 Resources

Models The models that we train are based on the pre-trained GPT-2 language models (Radford et al., 2019). GPT-2 is an auto-regressive Transformer-decoder based language model for English and comes in four sizes: small (12 layers), medium (24 layers), large (36 layers) and extra large (48 layers). Our experiments use the small (`sml`) and medium (`med`) model sizes.

Pre-training data The GPT-2 models are (further) pre-trained with Italian (`ita`) and Dutch (`nld`) data. The Italian pre-training data is the same dataset that was used to train the Italian GPT-2 small language model GepPpeTto (De Mattei et al., 2020). This dataset is a combination of Wikipedia data (2.8GB) and web texts from the ItWaC corpus (11GB; Baroni et al. 2009). Dutch data consists of a combination of Wikipedia (2.0GB), newspaper articles (2.9GB; Ordelman, Roeland J.F. et al. 2007), books (6.5GB) and articles from various Dutch news websites (2.1GB). Documents are filtered to only contain Dutch texts using the Wikipedia-trained fastText language identifier (Joulin et al., 2017), and are deduplicated based on exact sentence matches. The final Dutch pre-training data contains 13GB of plain text, of which 5% is reserved as development data.

Evaluation data The Italian models are tested using the same three corpora that were used to evaluate GePpeTto (De Mattei et al., 2020): Wikipedia, ItWaC, EUR-Lex (laws), newspapers and blog posts. A 5% subset of this data is used for development. For perplexity evaluation, the Dutch 500 million word, 22-genre SoNaR corpus is used (Oostdijk et al., 2013). The smaller 1 million word SoNaR-1 subcorpus is used as development data.

Tokenisation The datasets are tokenised using byte-pair-encoding (BPE). For better comparison, the Italian vocabulary is taken from the GePpeTto model (De Mattei et al., 2020). The Dutch BPE vocabulary is based on the full pre-training data and it has been ensured that every character that is used in the Dutch language is present as a single character token in the vocabulary. A large vocabulary size is beneficial because words are less often split in separate tokens, but vocabularies that are too large will have low token coverage for uncommon tokens.

Computation Training a model like GPT-2 is a computationally expensive task that requires access to costly hardware for long training times. All models discussed in this paper are trained with eight parallel NVIDIA V100 32GB GPUs on the Peregrine high performance computing cluster at the University of Groningen. For efficient implementation of the models, we use PyTorch (1.6.0; Paszke et al. 2019), PyTorch Lightning (0.9.0; Falcon 2019) and Transformers (3.0.2; Wolf et al. 2020). We implement four strategies to decrease general training time. First, the models are trained with 16-bit automatic mixed-precision training (Micikevicius et al., 2018). This decreases training time with a factor of two to three times. Second, we split each document in windows of 128 instead of 1024 tokens when we only train the lexical embeddings. Third, we minimise padding by using bucketed random sampling which means that sequences within minibatches have roughly the same length. Finally, we use maximum batch sizes that fit into GPU memory and use gradient accumulation in order to do backpropagation only for every 2000 examples. The models are trained with the Adam optimiser (Kingma and Ba, 2017) and initial learning rates are chosen based on the steepest loss slope with gradually increasing learning rates (Smith, 2017). The learning rate is reduced by 10% on when training loss reaches a plateau. More implementation details are given in the git repository.

4 Cross-language Transfer

We adapt GPT-2 for Italian and Dutch with minimal random initialisation. The lexical embeddings in GPT-2 are trained with an English BPE vocabulary. Therefore, they are not usable for the new languages and the lexical embedding layer has to be randomly initialised for the target vocabulary. This lexical embedding layer is used both as the first and the last layer of GPT-2 (tied weights). Relearning lexical embeddings with frozen Transformer layers prevents catastrophic forgetting in the Transformer layers when the embeddings are still random.

Relearning lexical embeddings Relearning lexical embeddings is nearly as computationally expensive as fully training the model, because backpropagation has to be done through the full model in order to update the lexical embeddings in the first layer of the model. However, loss values stabilize after only one to two epochs with lexical embedding relearning whereas full model training takes more training time. We retrain the lexical embeddings for the `sml` and `med` model for Italian and Dutch by training until loss on the validation data stops decreasing. When we retrain the `sml` model, the perplexities on our Italian and Dutch test data become 44.2 and 48.9 respectively. These perplexity scores show that the `sml` model can predict Dutch and Italian tokens reasonably well without having retrained the Transformer layers. Therefore, the English Transformer layers are at least partially language-independent and our relearning method automatically aligns lexical embeddings to the embedding space of the English model. However, if we retrain the `med` lexical layer for Italian and Dutch with the same method, test data perplexities are 81.2 and 185.0. These unsatisfactory `med` perplexities could be due to stopping training too early or to arriving at a suboptimal local optimum. Training for a longer time or trying different random initialisations defeats the purpose of minimising computational requirements. A more efficient method that uses the already learned `sml` embeddings is described in Section 5.

Vocabulary alignment The lexical embeddings of both the original English tokens as well as the relearned Italian and Dutch lexical embeddings can be considered to inhabit the same embedding space because the lexical embeddings of all three languages are tuned to minimise loss with the exact same Transformer layers. Therefore, tokens with similar meaning in different languages should be close to each other if the lexical embeddings are properly trained. Table 1 shows the closest Italian and Dutch tokens of a random sample of English tokens. These alignments show that the optimal lexical embeddings for both Italian and Dutch are often literal translations of English tokens. Thanks to similarity of context-dependent structures like syntax in these three languages, the English model can be adapted to Italian and Dutch. Based on this small sample, Dutch to English alignment seems to be slightly more accurate than Italian to English, but a more thorough study would be required to evaluate the actual relation between genetic similarity and alignment potential through this method.

English	**Italian**	**Dutch**
while	mentre	terwijl
genes	geni	genen
clothes	vestiti	kleren
musicians	composi[...]	artiesten
permitted	ammessa	toegelaten
Finally	infine	Eindelijk
satisfied	soddisfatto	tevreden
Accuracy:	85%	89%

Table 1: Alignment of closest tokens in the lexical embeddings of $\texttt{sml}_{\texttt{rle}}$ for Italian and Dutch. Accuracy scores are based on a manual evaluation by the authors of 200 random aligned tokens. Semantically correct subword matches are included.

Text generation Table 2 shows some examples of unconditioned text generation of the English `sml` with relearned lexical embeddings for Italian and Dutch. These examples show that the model can generate proper Italian and Dutch sentences, although it sometimes uses English word order where the correct word order differs in Dutch, or ignores grammatical gender agreement in Italian defaulting to the singular masculine, or doesn't always produce correctly Italian prepositional articles ("di la", en: *of the* vs "della", en: *of-the*). Phrases in italics in Table 2 highlight such mistakes.

The literal English translations, however, show that the models can generate proper Italian and Dutch grammar that differs from English. Italian and Dutch lexical embeddings are not only aligned with equivalent English tokens, but unexpected correct syntax shows that the grammatical functions of words have also been adapted. For example, in Ital-

Italian	**Literal English translation**
La prima parte del film venne *distribuito* in Giappone con l'aggiunta della colonna sonora.	The first part of the film was *distributed* in Japan with the addition of the soundtrack.
L'unico motivo *di la* mia insoddisfazione fu il fatto che l'inizio della sua attività [. . .]	The only reason *of the* my unsatisfaction was the fact that the beginning of-the his/her activity [. . .]
Il suo nome deriva da un vocabolo arabo.	The his/her name derives from a word Arabic.
Dutch	**Literal English translation**
In een artikel in de Journal of Economicologie (1998), *The New York Times schrijft*:	In an article in the Journal of Economicology (1998), *The New York Times writes*:
Ik kan me niet voorstellen dat mensen van mijn generatie *zijn zo boos op mij te wachten*.	I can me not imagine that people of my generation *are so mad at me to wait*.
Ik heb niets gedaan om mijn moeder te helpen.	I have nothing done to my mother to help.

Table 2: A selection of generated sentences by the `sml` model with Italian and Dutch lexical embeddings. Phrases in italics are ungrammatical in the target language.

ian the noun-adjective order is opposite to English and realised correctly; also, the use of the definite article in front of a possessive pronoun is correctly introduced, while ungrammatical in English.

This shows that the relatively low-dimensional context-independent lexical embeddings in GPT-2 contain syntactic features of the tokens in addition to semantics, and confirms previous findings of high information density in the lexical layer of language models (de Vries et al., 2020). Therefore, language adaptation can be to some extent effective by adapting the lexical embedding layer without retraining Transformer layers at all.

5 Scaling up Complexity

Replacing the original lexical embeddings with lexical embeddings from a different target language seems an effective way to initialise full model transfer to that target language. However, relearning the lexical embeddings of a new vocabulary requires full forward and backward propagation through the whole model. Therefore, this becomes an increasingly more expensive task for larger model sizes. When multiple model sizes need to be transferred to a new language, the lexical embeddings do not need to be retrained from scratch. Instead, vocabulary alignment between the source and target languages for the smaller model could be used to initialise the embeddings for a larger model.

After relearning the lexical embeddings of the `sml` model for Italian and Dutch, we observed that tokens with similar meaning in different languages are close to each other in the embedding space. This alignment effect should also be present in properly trained lexical embeddings of larger models. Given that we have at our disposal known embeddings for all 50K English tokens for every model size, we can use these data points to transform model size `sml` to larger model size `med`.

Regardless of architecture, embeddings are only considered to be alignable if they are trained under identical conditions with the same type and amount of data (Levy et al., 2015; Ruder et al., 2019). Our goal differs from previous alignment efforts since instead of aligning languages, we align separately trained embeddings for different model sizes, trained on the same data with identical and fully parallel vocabularies in English. The embeddings differ in dimensionality (768d for `sml`, 1024d for `med`) and the different model sizes may influence the amount and density of information in the lexical embeddings.

5.1 Transformation methods

The 50K parallel English tokens can be used to find an optimal transformation between lexical embeddings of different model sizes. The completeness of this mapping due to shared vocabularies between models eliminates the need to use complex solutions like refinement or bootstrapping the lexicon (Artetxe et al., 2018). We compare three simple supervised alignment methods for transformation from source space `sml` to target space `med`.

Regression (lstsq) A classic approach for mapping lexical embeddings is mean-squared-error minimising linear regression with the least-squares method (Ruder et al., 2019; Mikolov et al., 2013). This method learns a transformation matrix $\boldsymbol{W}$ that minimises the Euclidean distance between source and target embeddings. The optimal matrix is ap-

	Italian		Dutch		
Model	**Int@1k**	**PPL**	**Int@1k**	**PPL**	**PPL (1 epoch)**
$\texttt{med}_{\texttt{rle}}$ (1 epoch)	0.38	-	185.02	-	-
$\texttt{sml}_{\texttt{rle}} \xrightarrow{proc} \texttt{med}$	**0.61**	8.12×10^{12}	**0.61**	5.02×10^{12}	52.69
$\texttt{sml}_{\texttt{rle}} \xrightarrow{lstsq} \texttt{med}$	0.56	**364.06**	0.56	**293.61**	**47.57**
$\texttt{sml}_{\texttt{rle}} \xrightarrow{1-nn} \texttt{med}$	0.37	2,764.19	0.36	1,101.59	50.25
$\texttt{sml}_{\texttt{rle}} \xrightarrow{10-nn} \texttt{med}$	0.37	20,715.80	0.35	11,871.66	56.88

Table 3: Scores for different transformation methods. Int@1K are the average 1k nearest English neighbours intersection (int) fractions between `sml` and transformed `med` embeddings. *PPL* is the perplexity on the test sets for Italian and Dutch. *PPL (1 epoch)* indicates the perplexity after one epoch of training, which is low if the transformed embeddings were close to a good local optimum.

proximated with stochastic gradient descent, and therefore this is not an exact solution.

Orthogonal Procrustes (proc) More recent alignment approaches constrain the transformation $\boldsymbol{W}$ to be an orthogonal matrix (Ruder et al., 2019; Artetxe et al., 2016). This constraint enables using the exact solution for the orthogonal Procrustes problem (Xing et al., 2015). The exact solution only rotates and reflects data points to be as close as possible to the target space without any scaling or translation, preserving monolingual invariance in the source embeddings (Artetxe et al., 2016).

Weighted K-Nearest Neighbours (knn) Unlike typical alignment approaches, we have a complete set of parallel data points in the source and target spaces (English). The unknown target language tokens can be approximated by taking the $\boldsymbol{K}$ nearest English tokens in the source `sml` embedding space and using the distance-weighted sum of these tokens in the target `med` embedding space.

5.2 Results after transformation

Table 3 shows `med` embedding similarity with source `sml` embeddings and perplexities on test data with transformed embeddings, as well as the transformed embeddings with one additional epoch of training. Results are consistent for Italian and Dutch. Nearest English neighbours are best preserved with the Orthogonal Procrustes method. However, the perplexity scores are extremely high for this method. The perplexity scores with the different methods vary in complete orders of magnitude. Based on this, the least-squares regression method outperforms the other methods. After one epoch of additional training with transformation initialised embeddings, the lstsq method still outperforms the other methods. It even outperforms the `sml` model with fully tuned lexical embeddings.

Model	**PPL**	
	`ita`	`nld`
$\texttt{sml}_{\texttt{rle}}$	44.19	48.85
$\texttt{sml}_{\texttt{rle}}$ `+ finetuning`	**42.45**	**39.59**
$\texttt{sml}_{\texttt{full}}{}^{*}$	193.15	219.34
$\texttt{med}_{\texttt{rle}}$ `+ finetuning`	42.51	44.68
`GePpeTto (sml)`	106.84	-

Table 4: Perplexities of the concatenated test data for the final models. The $\texttt{med}_{\texttt{rle}}$ model is in practice the $\texttt{sml}_{\texttt{rle}} \xrightarrow{lstsq} \texttt{med}$ model. * The $\texttt{sml}_{\texttt{full}}$ model is trained for the equivalent amount of time as the $\texttt{sml}_{\texttt{rle}}$ `+ finetuning` models, but with all layers unfrozen.

6 Full model finetuning

After obtaining lexical embeddings for Italian and Dutch to be plugged into the English GPT-2 models, the full models can be finetuned for the target language. The best performing lexical embeddings will be used to train the `sml` and `med` Italian and Dutch models. These are the lexical embeddings that are relearned from random initialisation for the `sml` model. For the `med` model, the lstsq transformed `sml` embeddings with additional training are used ($\texttt{sml}_{\texttt{rle}} \xrightarrow{lstsq} \texttt{med}_{\texttt{+rle}}$).

The relearned lexical embeddings reduce the risk of information loss while the model is adjusting to a new language. Nevertheless, information can still be lost during training. For instance for the `sml` Dutch model, validation loss increases with a learning rate of 10^{-4}, but this does not happen with a lower learning rate of 10^{-5}.

7 Obtained models and evaluation

For both Italian and Dutch, we evaluate three models: (i) the English `sml` model with relearned lexical embeddings; (ii) the `sml` model with additional finetuning to the target language; and (iii) the English `med` model with relearned lexical embeddings that were initialised by transforming `sml` embeddings with the least-squares method. For Italian, we also include the GPT-2 small based GePpeTto model (De Mattei et al., 2020), which was trained from scratch. This inclusion offers the opportunity of a direct comparison between a GPT-2 model trained from scratch and those obtained with our transfer approach. We run both an automatic and a human-based evaluation. For the former, we compare perplexity scores on unseen test data in different genres. For the latter, we collect and compare judgements over generated and gold texts by native speakers of Italian and Dutch.

7.1 Perplexity

Table 4 shows perplexity scores on concatenated multi-genre test data based on a strided moving window perplexity calculation.[3] Perplexities are calculated with Italian and Dutch vocabularies of 30K tokens. These results show that perplexities are low when only relearning the lexical embeddings for both Italian and Dutch. Further finetuning of the `sml` model seems to have the greatest effect for the Dutch language. The `med` models with relearned lexical embeddings have lower perplexity than the equivalent `sml` models. This shows that language transferability based on the lexical layer is not restricted to small model sizes. Moreover, we see that our proposed method results in lower perplexity scores than regular full model finetuning of the English model. The overall perplexity scores of Italian are closer to each other than the Dutch perplexities. We also tested perplexities by the different genres that make up both the Italian and the Dutch datasets (see Figure 5 and Figure 6 for details), and observed that while perplexities vary greatly per genre, the model ranking per genre is consistent with the global scores.

7.2 Human Judgements

The perplexity scores give an indication on how well a language is represented by language models, but this does not reliably tell how good the model is in a generative setting. For this, we resort to human judgements. Human assessments of generated texts are collected for the models that incorporate the crucial steps in our approach and achieve reasonable perplexity scores: the `sml` models with only relearned lexical embeddings, the finetuned `sml` models and the higher complexity `med` models with only relearned lexical embeddings based on transformed `sml` lexical embeddings.

Model	Social	News	Legal
$\texttt{sml}_{\texttt{rle}}$	134.64	67.14	16.95
$\texttt{sml}_{\texttt{rle}}$ `+ finetuning`	**118.19**	**55.63**	15.36
$\texttt{med}_{\texttt{rle}}$	123.64	59.18	**14.95**
$\texttt{GePpeTto}_{\texttt{sml}}$	179.47	80.83	34.71

Table 5: Perplexities for different genres within the Italian test data. Rankings are consistent with Table 4 except for the legal domain.

Model	Proceedings	News	Legal
$\texttt{sml}_{\texttt{rle}}$	44.47	239.14	52.01
$\texttt{sml}_{\texttt{rle}}$ `+ finetuning`	**36.35**	**171.83**	**42.92**
$\texttt{med}_{\texttt{rle}}$	40.62	234.52	45.01

Table 6: Perplexities for some SoNaR genres in Dutch. Models rankings are consistent across genres.

Texts are assessed in isolation by means of a *direct* evaluation (Novikova et al., 2018).[4] Subjects are presented with texts on the screen, and are asked whether the texts they see could have been written by a human. All subjects are pre-informed that some of the texts they will see are machine generated. Rather than discrete answers, we obtain continuous evaluations by offering the possibility of clicking anywhere on a bar whose extremes are "no" to the left and "yes" to the right.

The evaluation interface is made with PsychoPy3 (Peirce et al., 2019) and hosted with Pavlovia[5].

Italian models were evaluated by 24 participants (9 M, 15 F) with ages ranging from 26 to 63 with a median age of 46. The Dutch models were evaluated by 15 participants (11 M, 4 F) with ages ranging from 23 to 36 with a median age of 27.

The three final models are evaluated for both languages; for Italian, we also add GePpeTto (De Mattei et al., 2020). Human written gold sentences

---

[3] Window sizes are 128 tokens and strides are 64 tokens except for GePpeTto. GePpeTto was trained with at most 100 tokens, so its window size is 100 with a 50 token stride.

[4] A direct evaluation is opposed to a comparative one, usually involving a ranking task (Novikova et al., 2018; De Mattei et al., 2020); this is left to future work.

[5] `https://pavlovia.org`

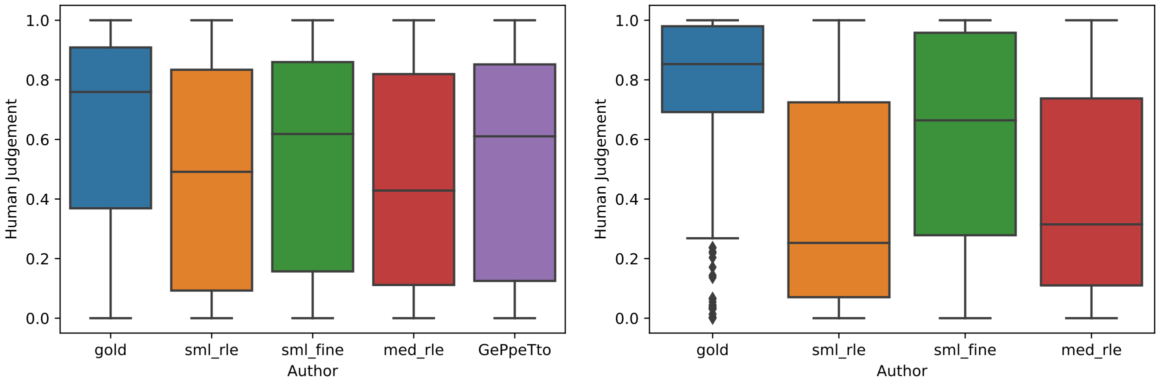

(a) Human judgement scores for Italian texts.

(b) Human judgement scores for Dutch texts.

Figure 1: Human judgement scores based on a continuous scale. Most judgements were close to 0 or 1.

were sampled from the test data as an additional condition. For each of these 5 Italian and 4 Dutch conditions, 100 sentences are evaluated. Each participant has evaluated 50 to 150 sentences and each sentence is evaluated by 3 to 5 participants. As a result, we obtain 1950 evaluations for 500 Italian texts and 1550 evaluations for 400 Dutch texts.

All artificial sentences are randomly generated without conditioning and with beam search (5 beams, with top 50 tokens or a summed probability of at least 90%), and a temperature of 3.0. Setting the temperature value >1 means decreasing the sampling probability of likely tokens, and therefore increases variation between generated samples.

Longer sentences have a higher chance to contain mistakes, so a model that generates longer sentences may have a disadvantage. However, explicitly controlling sentence length is not possible nor desired since sentence length may also be an indication of model quality. For both languages the randomly sampled gold sentences have more long sentences than the models, but the `sml` model with finetuning also sometimes generates longer sentences. We filter out sentences longer than 30 tokens to decrease sentence length effects on judgements. The remaining Italian sentences have median lengths of 18 or 22 words and the Dutch ones 16 or 17 words for the different conditions.

Figure 1 shows the distributions of human judgements per condition. Variance seems to be high due to the non-normally distributed scores as relatively many scores are close to zero, half or one. The model differences appear stronger for Dutch than Italian, but for both languages the subjects have given high scores to gold sentences. This is expected and indicates that the participants are able to correctly judge real human texts. Of the three trained models, the small model with additional finetuning achieves the highest scores.

For the Italian model comparison we use a linear mixed-effects model but with only author as fixed effect and random intercepts for participants and sentences. There is no significant effect for sentence length. The judgements on gold texts are significantly higher than all model judgements ($p < 0.005$) except for $\texttt{sml}_{\texttt{fine}}$. However, $\texttt{sml}_{\texttt{fine}}$ is not significantly better than GePpeTto nor the $\texttt{sml}_{\texttt{rle}}$ and $\texttt{med}_{\texttt{rle}}$ models ($p > 0.05$).

For Dutch we use a linear mixed-effects model with fixed effects for author and sentence length (in number of words) and random intercepts for participants and sentences. Sentence length has a significant negative effect ($p < 0.001$). All artificial authors score significantly lower than gold ($p < 0.001$). As for Italian, the $\texttt{sml}_{\texttt{fine}}$ model appears the best model, but in this case the judgement scores are significantly higher than for the other two models ($p < 0.001$). The $\texttt{sml}_{\texttt{rle}}$ and $\texttt{med}_{\texttt{rle}}$ models do not differ significantly from each other.

The human judgements show consistent results across the languages, but differences between Dutch judgements are stronger than for Italian. This seems to mirror the smaller perplexity differences for Italian than for Dutch. Whether demographic or cultural differences also play a role in this difference will need to be further investigated.

In sum, we see that the English GPT-2 models with relearned lexical embeddings are recognisable as artificial, whereas this problem is attenuated after additional finetuning. The `sml` model with additional finetuning performs at least as well as the GePpeTto model that was trained from scratch.

8 Conclusion

We have described methods to adapt GPT-2 to genetically related languages and to increase model complexity. Retraining lexical embeddings forces the model to learn representations that are aligned between English and the target language. GPT-2 is able to generate realistic text in another language, but human judgements reveal that additional finetuning of the full model is needed to generate realistic sentences more consistently. Relearned lexical embeddings show signs of syntactic adaptation to the new language, though not fully consistently.

Dutch is genetically closer to English than Italian, but our results do not prove that this method works better for Dutch. Future research on the relation between degrees and types of language similarity and transferability of models will enable more effective monolingual transfer, and possibly training better multilingual models by selecting optimal clusters of languages. This kind of work offers a privileged perspective into the information learned by generative language models and provides empirical ground for linguistic typology research (e.g., uncovering which linguistic aspects are more universal, and which more language-specific).

Relearning lexical embeddings using our method can still be considered an expensive solution, but training costs decrease when a smaller embedding space is scaled up to the embedding space of a larger model. In other words, approximating a good initialisation of the embedding weights decreases training time. This method also enables adaptation of (extra) large GPT-2 models to other languages.

If you can borrow pre-trained weights, why retrain models from scratch? In the right column we summarise the steps for the shortest path to train your own GPT-2 for another language.

Acknowledgments

We gratefully acknowledge the support of the Dutch Research Council (NWO Aspasia grant for M. Nissim) and the financial support of the Center for Groningen Language and Culture (CGTC). Additionally, we would like to thank Lorenzo De Mattei for sharing the Italian data with us. We would also like to thank the Center for Information Technology of the University of Groningen for providing access to the Peregrine high performance computing cluster. Finally, we thank the anonymous reviewers for their insightful feedback. Any mistakes remain our own.

Impact Statement

This work aims to minimize the environmental impact of training large neural language models by adapting existing models and by using smart initialisation of model weights. However, experiments in this paper still require the use of GPUs for extended periods of time which has environmental impact. Our final models are published and all models that automatically generates natural text could unfortunately be used maliciously. While we cannot fully prevent such uses once our models are made public, we do hope that writing about risks explicitly and also raising awareness of this possibility in the general public are ways to contain the effects of potential harmful uses. We are open to any discussion and suggestions to minimise such risks.

Create your own GPT-2 model

This paper describes several steps that are taken to transfer GPT-2 to a different language. The recommended shortest path to replicate this for another language is to follow these steps:

Vocabulary Create a new BPE vocabulary for your target language. The optimal size for your vocabulary depends on your language, so select the size by stepwise increments until the number of tokens per sentence slows to decrease.

Start small Re-initialise the lexical embeddings of the small GPT-2 model for your vocabulary size and only retrain the lexical embeddings.

Increase model size If you want to train a larger model size, fit a least-squares regression model to the English lexical embeddings in the small and larger model size and use the fitted model to transform your newly trained lexical embeddings to a larger model size.

Optimise your embeddings Do additional lexical embedding training in the target model size. Transformed embeddings are a good initialisation, but they are not perfect.

Finetune Unfreeze the full target model and do some finetuning to make sure that syntax differences are learned by the new model. Use a low learning rate like 10^{-5}.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised Neural Machine Translation. *arXiv:1710.11041 [cs]*. ArXiv: 1710.11041.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the Cross-lingual Transferability of Monolingual Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. *arXiv:1710.04087 [cs]*. ArXiv: 1710.04087.

Lorenzo De Mattei, Michele Cafagna, Felice Dell'Orletta, Malvina Nissim, and Marco Guerini. 2020. GePpeTto Carves Italian into a Language Model. *arXiv:2004.14253 [cs]*.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. *arXiv:2001.06286 [cs]*.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2020. Ethnologue: Languages of the World. Twenty-third edition. SIL International.

W.A. Falcon. 2019. PyTorch Lightning.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 710–721, Florence, Italy. Association for Computational Linguistics.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984, Brussels, Belgium. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Yunsu Kim, Yingbo Gao, and Hermann Ney. 2019. Effective Cross-lingual Transfer of Neural Machine Translation Models without Shared Vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1246–1257, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. In *arXiv:1412.6980 [cs]*.

Tom Kocmi and Ondřej Bojar. 2018. Trivial Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Brussels, Belgium. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing Transfer Languages for Cross-Lingual Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3125–3135, Florence, Italy. Association for Computational Linguistics.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. *arXiv:1710.03740 [cs, stat]*. ArXiv: 1710.03740.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. *arXiv:1309.4168 [cs]*.

Tasnim Mohiuddin, M Saiful Bari, and Shafiq Joty. 2020. LNMap: Departures from Isomorphic Assumption in Bilingual Lexicon Induction Through

Non-Linear Mapping in Latent Space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2712–2723, Online. Association for Computational Linguistics.

Toan Q. Nguyen and David Chiang. 2017. Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2018. RankME: Reliable Human Ratings for Natural Language Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 72–78, New Orleans, Louisiana. Association for Computational Linguistics.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [MASK]? Making Sense of Language-Specific BERT Models. *arXiv:2003.02912 [cs]*.

Nelleke Oostdijk, Martin Reynaert, Véronique Hoste, and Ineke Schuurman. 2013. The Construction of a 500-Million-Word Reference Corpus of Contemporary Written Dutch. In Peter Spyns and Jan Odijk, editors, *Essential Speech and Language Technology for Dutch: Results by the STEVIN programme*, Theory and Applications of Natural Language Processing, pages 219–247. Springer, Berlin, Heidelberg.

Ordelman, Roeland J.F., de Jong, Franciska M.G., van Hessen, Adrianus J., and Hondorp, G.H.W. 2007. TwNC: a Multifaceted Dutch News Corpus. *ELRA Newsletter*, 12(3-4).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Jonathan Peirce, Jeremy R. Gray, Sol Simpson, Michael MacAskill, Richard Höchenberger, Hiroyuki Sogo, Erik Kastman, and Jonas Kristoffer Lindeløv. 2019. PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, 51(1):195–203.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A Survey of Cross-lingual Word Embedding Models. *Journal of Artificial Intelligence Research*, 65:569–631.

Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788, Melbourne, Australia. Association for Computational Linguistics.

Wietse de Vries, Andreas van Cranenburgh, and Malvina Nissim. 2020. What's so special about BERT's layers? A closer look at the NLP pipeline in monolingual and multilingual models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4339–4350, Online. Association for Computational Linguistics.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. BERTje: A Dutch BERT Model. *arXiv:1912.09582 [cs]*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*.

Shijie Wu and Mark Dredze. 2020. Are All Languages Created Equal in Multilingual BERT? *arXiv:2005.09093 [cs]*. ArXiv: 2005.09093.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

Can Cognate Prediction Be Modelled as a Low-Resource Machine Translation Task?

Clémentine Fourrier **Rachel Bawden** **Benoît Sagot**
Inria, France
`firstname.lastname@inria.fr`

Abstract

Cognate prediction is the task of generating, in a given language, the likely cognates of words in a related language, where cognates are words in related languages that have evolved from a common ancestor word. It is a task for which little data exists and which can aid linguists in the discovery of previously undiscovered relations. Previous work has applied machine translation (MT) techniques to this task, based on the tasks' similarities, without, however, studying their numerous differences or optimising architectural choices and hyper-parameters. In this paper, we investigate whether cognate prediction can benefit from insights from low-resource MT. We first compare statistical MT (SMT) and neural MT (NMT) architectures in a bilingual setup. We then study the impact of employing data augmentation techniques commonly seen to give gains in low-resource MT: monolingual pretraining, backtranslation and multilinguality. Our experiments on several Romance languages show that cognate prediction behaves only to a certain extent like a standard low-resource MT task. In particular, MT architectures, both statistical and neural, can be successfully used for the task, but using supplementary monolingual data is not always as beneficial as using additional language data, contrarily to what is observed for MT.

1 Introduction

The Neogrammarians (Osthoff and Brugmann, 1878) formalised one of the main hypotheses of the then recent field of comparative linguistics, the regularity of sound changes: if a phone in a word, at a given moment in the history of a given language, evolves into another phone, then all occurrences of the same phone in the same phonetic context in the same language evolve in the same way.

Sound changes are usually identified by looking at the attested (or hypothesised) phonetic form of specific sets of words, called *cognates*, whose definition varies in the literature depending on the field.[1] We use an extension of the customary definition used in historical linguistics, as described for instance in (Hauer and Kondrak, 2011; List et al., 2017), which is the following: given two languages with a common ancestor, two words are said to be cognates if they are an evolution of the same word from said ancestor, having undergone the sound changes characteristic of their respective languages' evolution. We extend it by also allowing the ancestor word (from the parent language) to also be considered a cognate. For example, Latin *bonus* 'good' gave Italian *buono* 'id.', Spanish *bueno* 'id.' and Spanish *bono* 'id.' by inheritance, and they are all cognates, whereas Spanish *abonar* 'to fertilise', obtained by derivation, is related but not a cognate (Figure 1).

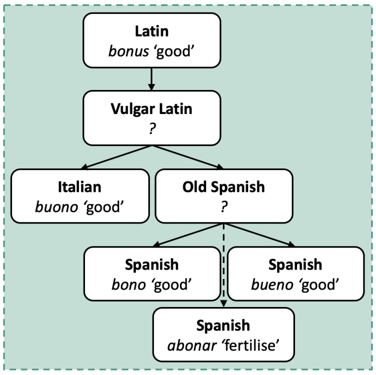

Figure 1: Related words in Italian and Spanish, both the outcome of Latin *bonus* 'good'. Plain arrows represent inheritance, dotted arrows derivation. "*?*" indicates that the word is not present in our database.

Cognate identification (finding cognate pairs in a multilingual word set) and prediction (producing

---

[1] Cognates have for example been defined as words sharing spelling and meaning, regardless of their etymology (Frunza and Inkpen, 2006, 2009), or words etymologically related no matter the relation (Hämäläinen and Rueter, 2019).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 847–861
August 1–6, 2021.

likely cognates in related languages) are two of the fundamental tasks of historical linguistics. Over the last three decades, automatic cognate identification has benefited from advances in computational techniques, first using dictionary-based methods (Dinu and Ciobanu, 2014) and purely statistical methods (Mitkov et al., 2007; McCoy and Frank, 2018), then statistical methods combined with clustering algorithms (Hall and Klein, 2010, 2011; List et al., 2017; St Arnaud et al., 2017), statistical methods combined with neural classifiers (Inkpen et al., 2005; Frunza and Inkpen, 2006, 2009; Hauer and Kondrak, 2011; Dinu and Ciobanu, 2014) and neural networks only (Ciobanu and Dinu, 2014; Rama, 2016; Kumar et al., 2017; Soisalon-Soininen and Granroth-Wilding, 2019).

Automatic cognate prediction is less studied despite its interesting applications, such as predicting plausible new cognates to help field linguists (Bodt et al., 2018) and inducing translation lexicons (Mann and Yarowsky, 2001). In the last few years, it has been approached as an MT task, as it can be seen as modelling sequence-to-sequence correspondences. Using neural networks has been promising (Beinborn et al., 2013; Wu and Yarowsky, 2018; Dekker, 2018; Hämäläinen and Rueter, 2019; Fourrier and Sagot, 2020a), although in most works the hyper-parameters of the neural models were not optimised. Moreover, the differences between MT and cognate prediction have not been studied.

In this paper, we choose to study the application of MT approaches to the cognate prediction task. Our aim is to investigate whether the task can benefit from techniques commonly seen to improve standard low-resource MT. We first highlight the specific characteristics of cognate prediction, and (to our knowledge) provide the first detailed analysis of the expected differences with standard MT. We then compare MT architectures (bilingual SMT vs. bilingual and multilingual NMT) when applied to cognate prediction. We study how to leverage extra data in our NMT models, either monolingual (via backtranslation or pretraining) or multilingual (introducing new languages). We experiment with Latin and its Romance descendants Spanish and Italian for all our experiments, as well as added French and Portuguese in a data augmentation setting. We find that cognate prediction is only similar to standard MT to a certain extent: the task can be modelled well using standard MT architectures (adjusted for a low-resource setting), and extending neural architectures to a multilingual setting significantly improves the results. In such multilingual settings, further improvements can be obtained by leveraging data from extra languages. However, using extra monolingual data via backtranslation or pretraining is not always as beneficial as it is in standard MT settings.[2]

2 Related Work

2.1 Cognate Prediction

Cognate prediction is the task that aims to produce from words in a source language plausible cognates in a target language (according to the aforementioned definition of cognates). It is a lexical task that models regular, word-internal sound changes that transform words over time. It has been approached with phylogenetic trees combined with stochastic sound change models (Bouchard et al., 2007; Bouchard-Côté et al., 2009; Bouchard-Côté et al., 2013), purely statistical methods (Bodt et al., 2018), neural networks (Mulloni, 2007), language models (Hauer et al., 2019) and character-level MT techniques (Beinborn et al., 2013; Wu and Yarowsky, 2018; Dekker, 2018; Hämäläinen and Rueter, 2019; Fourrier and Sagot, 2020a; Meloni et al., 2021), because of its similarity to a translation task (modelling sequence-to-sequence cross-lingual correspondences between words).

2.2 Low-resource MT

Since data is scarce, we postulate that cognate prediction could benefit from low-resource MT settings techniques and architectural choices.

2.2.1 Architecture Comparison

Several papers comparing SMT with NMT (recurrent neural networks (RNNs) with attention) in low-resource settings conclude that SMT performs better, being more accurate and less prone to overfitting (Skadiņa and Pinnis, 2017; Dowling et al., 2018; Singh and Hujon, 2020). However, as Dowling et al. (2018) themselves note, they did not optimise hyper-parameters for NMT. Sennrich and Zhang (2019) analysed and reproduced previous comparisons, to conclude that SMT can actually be outperformed by NMT when architectures and hyper-parameters are carefully chosen, but only above a certain quantity of data.

---

[2] Both our code and data are freely available at `http://github.com/clefourrier/CopperMT`.

2.2.2 Leveraging Extra Data

Several techniques are commonly used in low-resource MT to mitigate the lack of parallel data: monolingual pretraining, backtranslation and using data from additional languages.

Monolingual pretraining (unsupervised) has, as in other NLP tasks, been highly beneficial to MT (Song et al., 2019; Conneau and Lample, 2019; Devlin et al., 2019; Liu et al., 2020). Before training on a translation task, model parameters are first pretrained using a language modelling objective, which enables the exploitation of monolingual data, more freely available than bilingual data.

Backtranslation originated in SMT (Bertoldi and Federico, 2009; Bojar and Tamchyna, 2011), and has been standard in NMT for several years (Sennrich et al., 2016; Edunov et al., 2018). Its goal is to artificially create larger quantities of parallel data from monolingual datasets, which are often more readily available. Target-side monolingual data is provided to a bilingual model trained in the opposite direction (target-to-source), which produces synthetic source-side data. The data is then filtered to keep the highest quality sentences. The newly generated dataset, made of synthetic source-side data parallel to real target-side data is then combined with the original bilingual set to train a new model.

Training multilingual NMT models has been shown to help low-resource scenarios by providing data in other languages and constraining the hidden representations to a shared, language-independent space. The amount of sharing between languages varies according to the approach, from multi-encoder, multi-decoder architectures (Luong et al., 2016), optionally sharing attention mechanisms (Firat et al., 2016a), to approaches with a single shared encoder and decoder (Ha et al., 2016; Johnson et al., 2017).

3 Differences between Cognate Prediction and MT

Cognate prediction and MT both focus on learning sequence-to-sequence correspondences. However, amongst the works using MT techniques for cognate prediction, little attention has been paid to their differences; the underlying linguistic assumptions and aims are quite distinct, which could impact the transferability of choices and techniques from MT.

Representation Units MT processes sentences split into individual (graphemic) units that can be of diverse granularity levels (characters, subwords or words). Cognate prediction, on the other hand, involves predicting sound correspondences from one cognate word to another, and so is best modelled using sequences of phones (like character-level MT).

Reordering and Alignment In MT, the correspondence between source and target sentences can involve long-distance reorderings, whereas the reorderings sometimes found in the correspondence between cognates are almost always local (e.g. metatheses). We therefore expect SMT, which is somewhat limited with respect to the modelling of long-distance context, to be less penalised in the cognate prediction setting than it usually is a standard MT setting.

Sample Length The input sequence to MT is the sentence, whereas for cognate prediction it is the word. Even with different segmentation granularities for MT, the average sequence length is generally much shorter for cognate prediction than for MT. Again, this could mean that SMT is less penalised than it is in the standard MT setup.

Modelled Relations MT involves symmetrical relations between sentences, whereas cognate prediction, as defined above, is inherently ambiguous in a counter-intuitive way (especially because it is structurally different from the usual MT ambiguity, where many valid translations exist for the same input). The cognate task models both symmetrical and asymmetrical relationships between cognates: parent-to-child (e.g. LA→ES), i.e. modelling sequences of regular sound changes, is non-ambiguous, whereas child-to-parent (e.g. ES→LA) and as a result, child-to-child (e.g. IT↔ES) is intrinsically ambiguous, as two distinct sounds in the parent language can result in the same outcome in the child language. When two distinct sounds in the child language are the outcome of the same sound in the parent language, it is always because their (word-internal) phonetic contexts were different in the parent language. In other words, the parent-to-child direction is (virtually) non-ambiguous, but might require taking the phonetic context into account. However, the child-to-parent direction is intrinsically ambiguous, which results from the fact that a sound in the child language can be the regular outcome of more than one sound in the parent language: for instance Spanish /b/ comes from Latin /p/ in *abría* (from Latin *aperīre*) but from Latin /b/ in *habría* (from Latin *habeō*).

Ambiguity Management When using cognate prediction as a tool to aid linguists, as in (Bodt and List, 2019), our aim is not to predict **the** single correct answer, but to provide a list of plausible candidates. In MT however, while many translations can be produced by the model (some better than others—including poor ones), it is possible to simply use the best ranked translation. In cognate prediction, as a consequence of the inherent ambiguity of the task discussed above, at most one prediction is correct, other predictions could have been correct (i.e. they are compatible with the phonetic laws involved), while other predictions are incorrect. A linguist would be interested in all correct or plausible predictions, not just the best ranked one, and there is therefore a need for n-best prediction.

Relevance of Leveraging Extra Data Whereas MT models could theoretically be trained on any sentence pair that are translations of each other, cognate prediction is far more limited in terms of which data can be used; cognacy relations only link a limited number of words in specific language pairs, limiting not only available parallel data but also the potential for synthetic data (e.g. via backtranslation). Using generic translation lexicons may help, but, as they do not only contain cognate pairs, all non-cognate pairs they contain (parallel borrowings from a third language and etymologically unrelated translations) are effectively noise for our task (Fourrier, 2020).

4 Experimental setup

Bearing in mind these differences, we seek to determine whether MT architectures and techniques are well suited to tackling the task of cognate prediction, paying attention to avoid the pitfalls raised by Sennrich and Zhang (2019) by carefully selecting architecture sizes and other hyper-parameters.

For our baselines, we train several character-level[3] MT models (SMT vs. RNNs and Transformers) in a bilingual setup, training a single model for each language pair.

We then assess the impact of techniques commonly used to improve MT in low-resource scenarios. We first investigate the impact of using monolingual data for all 3 architecture types, via pretraining and backtranslation,[4] then take advantage of the ability of NMT to accommodate multilingual architectures to experiment with a multi-encoder multi-decoder architecture (Firat et al., 2016b) involving all language directions.

Finally, we test whether there can be any benefit from combining multilinguality with either pretraining or backtranslation.

4.1 Data

Our datasets (detailed below) are bilingual cognate lexicons for all our experiments, extended with monolingual lexicons for backtranslation and pretraining (see Table 1). As we focus on sound correspondences, we phonetise our datasets. Each word is phonetised into IPA using `espeak` (Duddington, 2007-2015), then cleaned to remove diacritics and homogenise double consonant representations. For example, *conocer* 'to know' is phonetised as [konoθɛɾ], then split into phones (segmented into [k, o, n, o, θ, ɛ, ɾ]).

BILINGUAL	LA–IT	LA–ES	ES–IT
#words	5,109	4,271	1,804
#phones	77,771	63,131	24,576
#Unique phones	34	39	38
Avg. word length	7.62	7.40	6.81
MONOLINGUAL	ES	IT	LA
#words	78,412	99,949	18,639
#phones	626,175	815,562	142,955
#Unique phones	38	40	29
Word length	7.98	8.24	7.67

Table 1: Dataset statistics for our lexicons.

Bilingual Lexicons Our bilingual cognate lexicons contain cognate pairs (between 657 and 5,109 depending on the language pair), extracted from the etymological database EtymDB2 (Fourrier and Sagot, 2020b) following (Fourrier and Sagot, 2020a) for extraction and duplicate management, then phonetised as described above. We use the lexicons containing Latin, Italian and Spanish (LA–IT, LA–ES, ES–IT) in all experiments. They respectively contain 5,109, 4,271 and 1,804 words for a total of 77,771, 63,131 and 24,576 phones (ES–IT being considerably smaller), with on average 40 different and unique phones.

We run all experiments on three different train/dev/test splits in order to obtain confidence scores. For the bilingual (baseline) and multilingual setups, each split is obtained by sampling sentences 80%/10%/10% randomly.

---

[3] We use here the customary term "character-level MT," although in our case, characters correspond to phones.

[4] We detail what pretraining and backtranslation means for our SMT models in Section 4.4.

Monolingual Lexicons Monolingual datasets are used for the monolingual pretraining and back-translation experiments. They were extracted from a multilingual translation graph, YaMTG (Hanoka and Sagot, 2014), by keeping all unique words for each language of interest. To remove noise, words containing non-alphabetic characters were discarded (punctuation marks, parentheses, etc.). The final datasets (cleaned and phonetised) contain between 18,639 and 99,949 unique words (the LA set is more than 4 times smaller than the others).

4.2 MT Architectures

4.2.1 SMT

We train a separate SMT model for each language direction using the MOSES toolkit (Koehn et al., 2007). Our bilingual training data is aligned with `GIZA++` (Och and Ney, 2003). The target data for the pair is used to train a 3-gram language model using `KenLM` (Heafield, 2011). We tune our models using `MERT` based on BLEU on the dev set.

4.2.2 NMT

We compare two encoder-decoder NMT models: the RNN (bi-GRU) with attention (Bahdanau et al., 2015; Luong et al., 2015) and the Transformer (Vaswani et al., 2017). We use the multilingual Transformer implementation of `fairseq` (Ott et al., 2019), and extend the library with an implementation of the multilingual RNN with attention (following the many-to-many setting from (Firat et al., 2016a) but with separate attention mechanisms for each decoder).[5] Each model is composed of one encoder per input language, and one decoder (and its own attention) per output language.[6] We train each model for 20 epochs (which is systematically after convergence), using the Adam optimiser (Kingma and Ba, 2015), the cross-entropy loss, and dev BLEU as selection criterion.

4.2.3 Hyper-parameter Selection

We ran optimisation experiments for all possible bilingual and multilingual architectures, using three different data splits for each parameter combination studied, and choosing the models performing best across seeds. Our initial parameters were selected from preliminary experiments (in bold in Table 2).

Parameters	Values studied
1) Learning rate × Batch size	{0.01, **0.05**, 0.001} × {10, 30, **65**, 100}
2) Embed. dim. × Hidden dim.	{8, 12, 16, **20**, 24} × {18, 36, **54**, 72}
3) Number of layers	**1**, 2, 4
4) Number of heads	**1**, 2, 3, 4
4) Attention type	None, Bahdanau, Luong (**dot**, concat, general)

Table 2: Parameter exploration experiments for NMT models. In bold, the initial parameters at each step.

Table 2 contains the successive parameter exploration steps: at the end of a step, we automatically selected (according to average dev BLEU) the step-best value, used as input parameter for the next parameter exploration step.[7] The final best parameters are given in Appendix A.1. Smaller learning rates (0.005 and 0.001) are better, while there is no observable pattern to the best batch sizes or numbers of layers. Interestingly, however, for the RNNs, the best results are obtained with the highest hidden dimension irrespective of the embedding size (72 vs. 20 or 24), whereas, for the Transformers, best results are obtained with the largest embedding size irrespective of the hidden dimension (24 vs. 54 or 72). Increasing the number of layers or using more than 1 head almost always increases performance.

4.3 Evaluation

For our task, we use the most commonly used MT evaluation metric, BLEU (Papineni et al., 2002), using the `sacreBLEU` implementation (Post, 2018). It is based on the proportion of 1- to 4-grams in the prediction that match the reference.

In standard MT, BLEU can under-score the many valid translations that do not match the reference. For cognate prediction, however, we expect a single correct prediction in most cases (there are a few exceptions such as variants due to gender distinctions specific to the target language). This makes BLEU better suited to the cognate prediction task than it is to standard MT.[8]

---

[5] These implementations are used in all setups, bilingual (using one language as source and one as target) as well as multilingual.

[6] In a multilingual setup, encoders, decoders and attention mechanisms can either be shared between languages or be language-specific. In preliminary experiments, using independent items proved to be the most effective. We also observe that a coherent phonetic embedding space is learned during training (described in Appendix A.2).

[7] When looking at multilingual models, we chose the model performing best on most languages, as measured by comparing the sum of the ranks (according to their average performance per language) of each model over all language pairs.

[8] BLEU is also more adapted than an exact match, as it allows us to compare how close the prediction is to the reference and in cognate prediction does not suffer from the same problems as in standard MT.

4.4 Leveraging Extra Data

Monolingual pretraining For NMT, one way to take advantage of additional monolingual data is to teach the model to "map" each language to itself by using an identity function objective on the monolingual data for the model's target language.[9] Using monolingual target data during pretraining allows each target decoder to have access to more target data (which avoids overfitting), while we expect it to be beneficial to encoders too, since our source and target languages tend to share common sound patterns in cognate prediction, being closely related. In practice, we pretrain the model for 5 epochs[10] using the identity function objective together with the initial cognate prediction objective (on the original bilingual data) and then fine-tuned on the cognate task as before for 20 epochs.

For SMT, model parameters cannot be pretrained as in NMT, so in the guise of pretraining, we take the nearest equivalent: we use target-side monolingual data to train an extra language model.

For each language pair, the monolingual dataset we use is composed of 90% of the target monolingual data. The bilingual data is the same as before.

Backtranslation For each architecture type, we use the previously chosen models to predict 10-best results for each seed from the monolingual target-side data, and construct synthetic cognate pairs from monolingual lexicons and source-side predictions. For each word, we keep the first prediction of the 10 that also appears in the relevant monolingual source language lexicon as our new source, and the initial source as target (this is akin to filtering back-translated data (e.g. to in-domain data) in MT, a standard practice). We discard pairs with no prediction match.

This large back-translated bilingual dataset is extended with our original training set. For NMT, it is used to train a new model for 10 epochs,[10] which is then fine-tuned for 20 epochs with the original bilingual training set. For SMT, it is used (instead of the original bilingual data) to train a new phrase table.

Multilingual NMT We exploit the fact that NMT can readily be made multilingual by training a single model on all language directions at once.

5 Results

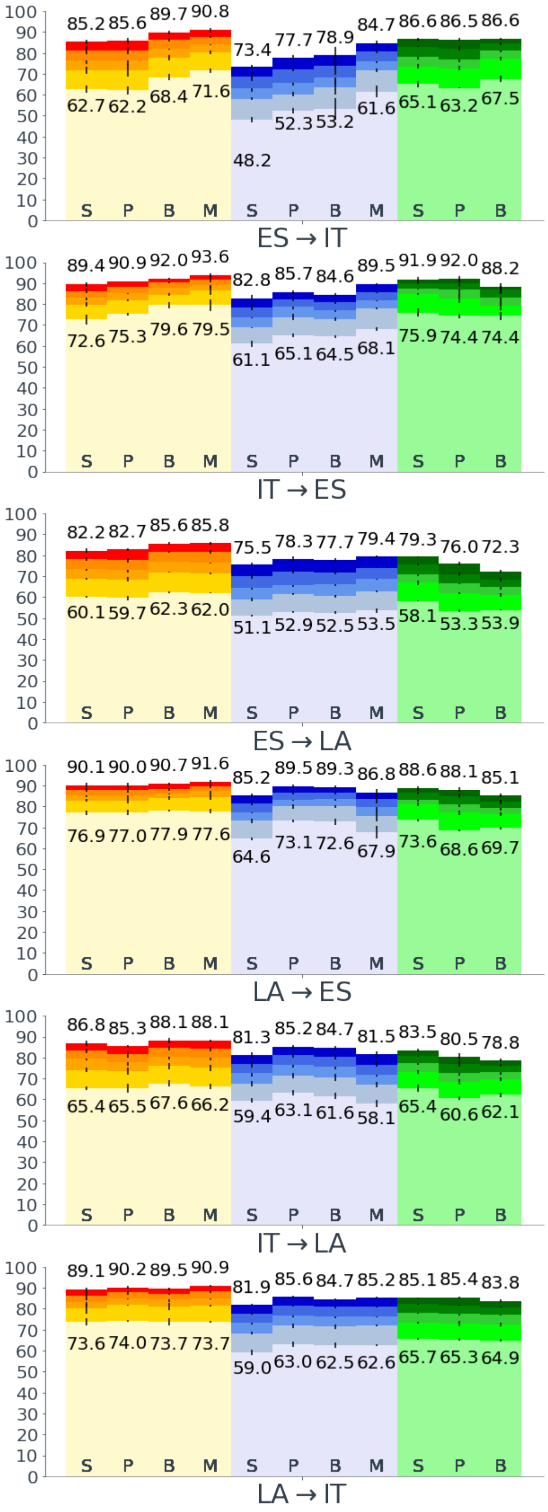

Figure 2: BLEU scores. Colours indicate the model type: RNNs in orange (col 1 to 4), Transformers in blue (col 5 to 8), SMT in green (col 9 to 11). Colour shades indicate the value of n in n-best predictions (1, 2, 3, 5 and 10 from bottom to top). The letters (x-axis) indicate the setup: S - standard/bilingual, P - with pretraining, B - with backtranslation, M - multilingual.

---

[9] For the multilingual model, this means that every encoder will see data from all languages, whereas each decoder will only see data from its specific language.

[10] This number of epochs is systematically big enough to reach convergence.

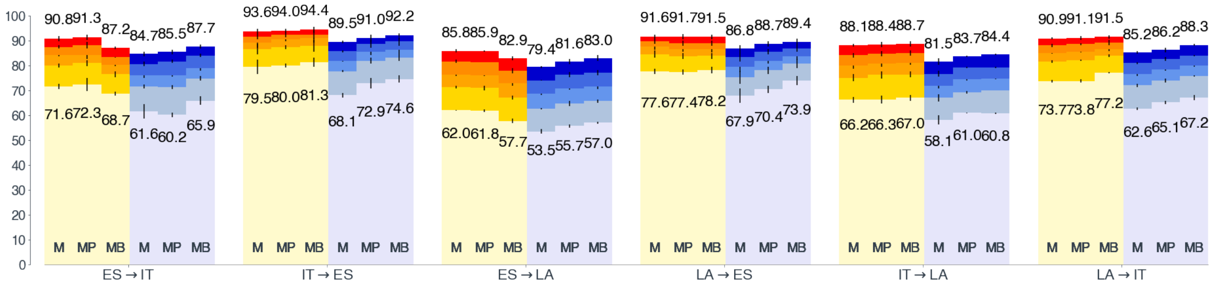

Figure 3: BLEU scores: RNNs in orange (col 1 to 3), Transformers in blue (col 4 to 6). Colour shades indicate the value of n in n-best predictions (1, 2, 3, 5 and 10 from bottom to top). On the x-axis, the letters indicate the setup: M - multilingual, MP - multilingual with pretraining, MB - multilingual with backtranslation.

5.1 Baseline: Bilingual setup

1-best Results At a first glance (Figure 2, "S" columns), SMT and RNN appear to have relatively similar results, varying between 58.1 and 76.9 BLEU depending on the language pair, outperforming the Transformer by 5 to 15 points on average. However, SMT performs better for IT↔ES (pair with the least data), and RNNs for the other pairs. This confirms results from the literature indicating that SMT outperforms NMT when data is too scarce, and seems to indicates that the data threshold at which NMT outperforms SMT (for our Romance cognates) is around 3,000 word pairs for RNNs, and has not been reached for Transformers.

n-best Results The BLEU scores for NMT and SMT increase by about the same amount for each new n ($n \leq 10$), reaching between 79.3 and 91.9 BLEU score at $n = 10$ for RNN and SMT. The Transformer, however, does not catch up.

5.2 Leveraging Extra Data

5.2.1 Pretraining, backtranslation

Both pretraining the models and using backtranslation (Figure 2, "P" and "B" columns) increase the results of the Transformer models by 1 to 9 points, though they are still below the RNN baseline. It is likely the added monolingual data mitigates the effect of too scarce bilingual sets. The impact on RNN performance is negligible for most language pairs, apart from the lowest resourced one (ES–IT), for which backtranslation increases results. Lastly, these methods seem to mostly decrease SMT performance, due to noisy data diluting the original (correct) bilingual data (cf. Section 3); this is less of a problem for NMT models, because they are then fine-tuned on the cognate task specifically.

5.2.2 Multilinguality

Data augmentation through a multilingual setup (Figure 2, "M" columns) seems to be the most successful data augmentation method for RNNs (increasing performance almost all the times), and allows them to finally outperform bilingual SMT for the least-resourced pair as well (ES↔IT). The Transformers benefit less from this technique than from adding extra monolingual data, apart for ES↔IT, most likely for the same reason as earlier: this dataset being the smallest, adding words in ES and IT from other language pairs helps to learn the translation and stabilises learning. This technique is not applicable to SMT.

Impact of the Translation Direction There are three relation types present in our experiments, each with their level of ambiguity (most for child-to-parent or child-to-child, least for parent-to-child, see Section 3). We observe that the SMT models, though bilingual, outperform multilingual NMT when going from ES or IT to LA (child to parent), and that multilingual NMT outperforms SMT in all other translation directions (ES↔IT, LA→ES, LA→IT: child-to-child and parent-to-child).

5.3 Combining data augmentation methods

We choose to combine the best performing data augmentation technique overall, multilinguality, with pretraining and backtranslation (Figure 3) for our NMT models.

Multilinguality + pretraining Combining multilinguality with pretraining has virtually no significant impact on the RNNs' results with respect to multilinguality only. For the Transformers, however, it increases the results by 2 to 3 BLEU on average.

Source context	Target context	Source	Target	SMT pred.	RNN pred.
(a) AMBIGUOUS SOUND CORRESPONDENCE NOT LEARNED WELL					
corvino ‘raven’	*corvino* ‘id.’	[kɔɾβino]	[korviːno]	[kɔrviːno]	[kɔrbiːno]
liebre ‘hare’	*lepre* ‘id.’	[lieβɾe]	[lɛːpre]	[liebrɛː]	[liɛːvre]
(b) MEANING AND/OR FORM OF THE COGNATES CHANGED TOO MUCH					
calaña ‘kind/sort’	*quale* ‘what/which’	[kalaɲa]	[kwaːle]	[kalaɲː]	[kalaɲː]
pie ‘foot’	*piede* ‘id.’	[pje]	[pjɛːde]	[pɛ]	[pɛːre]
(c) DATA ERROR					
suspirar ‘to sigh’	*squillan* ‘(it) rings’	[suspiɾaɾ]	[skwilːan]	[sospiraːre]	[sospiraːre]
frenesí ‘frenzy’	*frenetico* ‘frenetic’	[frenesi]	[frenɛtiko]	[freːnezi]	[fronɛsː]
(d) MODEL MISTAKE					
licencioso ‘licencious’	*licenciozo* ‘id’	[liθɛnθjoso]	[litʃentsioːzo]	[litʃɛntsioːzo]	[litʃɛntso]

Table 3: Prediction errors examples across ES→IT datasets for both SMT and RNN.

Multilinguality + backtranslation Combining multilinguality with backtranslation provides the best results overall for Transformers (both being the best performing methods for these models). For the RNNs, however, the performance increase is smaller for most languages, and we even observe a decrease in performance when translating from ES (which was not the case with bilingual models).

6 Discussion

We discuss the results of the best performing models for the best seed across all architectures (SMT, *multilingual* + *pretraining* RNN and *multilingual* + *backtranslation* Transformer) from ES→IT. More than a third of the predicted words are above 90 BLEU[11] (resp. 35.4/46.4/38.1% for SMT/RNN/Transformer), and for error analysis, we study the words below this threshold. The observations generalise to other language pairs.

6.1 Predictions

Close Results We observe a lot of inaccurate but very close translations (e.g. Spanish *conveniente* ‘convenient’, phonetised [kɔmbenjɛnte], was predicted as corresponding to Italian [konvenjɛnte] instead of [konveniɛnte], with only one phone different, and coherently so). Sometimes these translations have a very bad score: Spanish *pulpito* ‘pulpit’, phonetised [pulpito], was predicted as [pʊlpiːto] instead of [pulpito], two close pronunciations, for a sentence BLEU score of only 20.

Analysis of wrong results Wrongly predicted cognates correspond to four cases, as defined in Table 3.[12] We carried out a manual error analysis, and observed that their distribution was similar across models (resp. SMT/RNN/Transformer):

(a) 84.6/81.4/79.5% were cognates with an ambiguous sound correspondence (e.g. Spanish [β] to Italian [b/v/p]).

(b) 10.3/13.4/11.6% were cognates that had either evolved too far away from one another or contain rare sound correspondences, such as *pie* ‘foot’, phonetised [pje], predicted [pe] and [peːre] instead of [pjɛːde] *piede* ‘foot’.

(c) 0.9/0.9/0.9% corresponded to data errors, such as *suspirar* ‘to sigh’, phonetised as [suspiɾaɾ], which was predicted as [sospiraːre] *sospirare* ‘to sigh’, its actual cognate, instead of its erroneous counterpart in our database ([skwilːan] *squillan* ‘(it) rings’).

(d) 4.3/4.3/8.0% were model errors, such as “forgetting” part of a word during translation.

6.2 Usefulness of n-best results

The average position at which the best prediction (according to dev BLEU) occurs (in 10-best predictions) is between 1 and 3 (Table 7 in Appendix A.3). The lowest indices occur for Spanish (between 1 and 1.7) and Italian (between 1.6 and 2.2). The highest indices encountered occur when going for IT→LA or ES→LA (between 2 and 3). This illustrates the importance of n-best prediction when predicting cognates from child to parent languages, due to ambiguity. Standard deviations are between 2 and 3: for these languages, when studying cognate prediction, it is interesting to at least check the 5-best results.

---

[11] To study the BLEU of individual words, we use the sentenceBLEU function from `sacreBLEU` with its default parameters.

[12] Statistics are provided for the best models of the best seed, but examples are taken across seeds and models.

6.3 Language choice in a multilingual setup

To study the impact of the language pairs used in the multilingual setup, we train additional multilingual neural models on only 1000 pairs of ES–IT data (single set), complemented by either nothing (to act as baseline), an extra 600 pairs of ES–IT, or 600 pairs of ES–L and IT–L (L being either Latin, a parent language, French, a related language, or Portuguese, more closely related to Spanish than Italian). The rest of the data (Table 4) is split equally between dev and test.

Bilingual	FR–IT	FR–ES	PT–IT	PT–ES
#words	666	657	1,503	1,874
#phones	8,698	8,530	20,738	25,867
#Unique phones	40	43	36	41
Word length	6.53	6.49	6.90	6.90

Table 4: Supplementary bilingual lexicon statistics.

As we saw in Section 5.1, the Transformers' scores are far more affected by low resource settings than the RNNs. We therefore study the impact of adding extra languages with RNNs only.

Baseline	ES→IT	IT→ES
1000 pairs	53.9 ± 3.4	66.6 ± 4.2
Added data	ES→IT	IT→ES
Same language pair	62.5 ± 2.5	71.8 ± 1.7
Latin	57.1 ± 1.8	67.4 ± 3.3
French	58.5 ± 2.0	67.0 ± 2.8
Portuguese	58.8 ± 1.1	66.9 ± 2.9

Table 5: BLEU for different multilingual settings.

Results on our new low-resourced baseline are lower than the our previous baselines by around 10 points (Table 5), which is expected, since we use less data for training.

Adding 600 pairs of ES–IT words has more effect on ES–IT performance than adding any other pair of related languages, which indicates that, unsurprisingly, the best possible extra data to provide is in the language pair of interest. When adding a related extra language, the results are better than with the initial data only. From Spanish, the performance is best when adding Portuguese, its most closely related language, then French, then Latin. From Italian, we observe the opposite trend. Adding an extra language seems to help most to translate from, and not to, the language it is most closely related to. For very low-resource settings, where extra pairs of the languages of interest might not be available, it will probably be interesting to explore using extra languages related to the source language.

7 Conclusion

We examined the differences between cognate prediction and MT, in terms of data as well as underlying linguistic assumptions and aims. We then observed that, above a certain training data size, SMT and multilingual RNNs provide the best BLEU scores for the task, SMT still being unrivalled when it comes to smaller datasets (which coincides with previous work comparing SMT and NMT for low-resource settings).

When studying how to increase the amount of training data seen by our models, we found that exploiting the multilinguality of NMT architectures consistently provided better results than adding monolingual lexicons (through pretraining or back-translation), which contain noise for our task; combining the methods provided a significant amelioration for Transformers only. Adding multilingual data by training with extra languages also proved interesting, and we found the best possible extra data to add in a multilingual setting is, first, data from the languages at hand, followed by pairs between them and a parent language, then finally data from additional languages as close as possible to the source language.

We conclude that cognate prediction can benefit from certain conclusions drawn in standard low-resource MT, but that its specificities (intrinsic ambiguity which requires n-best prediction, reliance on cognate data only) must be systematically taken into account. Computational cognate prediction using MT techniques is a field in its infancy, and the work in this paper can be extended along several axes: working on less studied language families, or using the method in collaboration with linguists to better understand the etymology and history of languages.

Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocation 2021-AD011011459R1 made by GENCI. This work was partly funded by the last author's chair in the PRAIRIE institute funded by the French national agency ANR as part of the "Investissements d'avenir" programme under the reference ANR-19-P3IA-0001.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate production using character-based machine translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 883–891, Nagoya, Japan. Asian Federation of Natural Language Processing.

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 182–189, Athens, Greece. Association for Computational Linguistics.

Timotheus A. Bodt, Nathan W Hill, and Johann-Mattis List. 2018. Prediction experiment for missing words in Kho-Bwa language data. *Open Science Framework Preregistration*.

Timotheus A. Bodt and Johann-Mattis List. 2019. Testing the predictive strength of the comparative method: an ongoing experiment on unattested words in Western Kho-Bwa languages. *Papers in Historical Phonology*, 4:22–44.

Ondřej Bojar and Aleš Tamchyna. 2011. Improving translation model by monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 330–336, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Alexandre Bouchard, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 887–896, Prague, Czech Republic. Association for Computational Linguistics.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 65–73, Boulder, Colorado. Association for Computational Linguistics.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 110(11):4224–4229.

Alina Maria Ciobanu and Liviu P. Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 99–105, Baltimore, Maryland. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7057–7067.

Peter Dekker. 2018. *Reconstructing language ancestry by performing word prediction with neural networks*. Ph.D. thesis, University of Amsterdam.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liviu Dinu and Alina Maria Ciobanu. 2014. Building a dataset of multilingual cognates for the Romanian lexicon. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1038–1043, Reykjavik, Iceland. European Language Resources Association (ELRA).

Meghan Dowling, Teresa Lynn, Alberto Poncelas, and Andy Way. 2018. SMT versus NMT: Preliminary comparisons for Irish. In *Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*, pages 12–20, Boston, MA. Association for Machine Translation in the Americas.

Jonathan Duddington. 2007-2015. espeak text to speech.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Clémentine Fourrier and Benoît Sagot. 2020a. Comparing statistical and neural models for learning sound correspondences. In *Proceedings of LT4HALA 2020 - 1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 79–83, Marseille, France. European Language Resources Association (ELRA).

Clémentine Fourrier and Benoît Sagot. 2020b. Methodological aspects of developing and managing an etymological lexical resource: Introducing EtymDB-2.0. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3207–3216, Marseille, France. European Language Resources Association.

Clémentine Fourrier. 2020. Évolution phonologique des langues et réseaux de neurones : travaux préliminaires (sound change and neural networks: preliminary experiments). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 3 : Rencontre des Étudiants Chercheurs en Informatique pour le TAL*, pages 110–122. ATALA et AFCP.

Oana Frunza and Diana Inkpen. 2006. Semi-supervised learning of partial cognates using bilingual bootstrapping. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 441–448, Sydney, Australia. Association for Computational Linguistics.

Oana Frunza and Diana Inkpen. 2009. Identification and disambiguation of cognates, false friends, and partial cognates using machine learning techniques. *International Journal of Linguistics*, 1(1):1–37.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the 13th International Conference on Spoken Language Translation. Seattle, US.*

David Hall and Dan Klein. 2010. Finding cognate groups using phylogenies. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1030–1039, Uppsala, Sweden. Association for Computational Linguistics.

David Hall and Dan Klein. 2011. Large-scale cognate recovery. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 344–354, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Mika Hämäläinen and Jack Rueter. 2019. Finding Sami cognates with a character-based NMT approach. In *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, pages 39–45, Honolulu. Association for Computational Linguistics.

Valérie Hanoka and Benoît Sagot. 2014. An open-source heavily multilingual translation graph extracted from wiktionaries and parallel corpora. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3179–3186, Reykjavik, Iceland. European Language Resources Association (ELRA).

Bradley Hauer, Amir Ahmad Habibi, Yixing Luan, Rashed Rubby Riyadh, and Grzegorz Kondrak. 2019. Cognate projection for low-resource inflection generation. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 6–11, Florence, Italy. Association for Computational Linguistics.

Bradley Hauer and Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilingual lists. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 865–873, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Diana Inkpen, Oana Frunza, and Grzegorz Kondrak. 2005. Automatic Identification of Cognates and False Friends in French and English. In *Proceedings of Recent Advances in Natural Language Processing 2005*, pages 251–257, Borovets, Bulgaria.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Shantanu Kumar, Ashwini Vaidya, and Sumeet Agarwal. 2017. Discovering Cognates Using LSTM Networks. In *Proceedings of the 4th Annual Conference of the Association for Cognitive Science*, Hyderabad, India.

Johann-Mattis List, Simon J. Greenhill, and Russell D. Gray. 2017. The potential of automatic word comparison for historical linguistics. *PLOS ONE*, 12(1):1–18.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Richard T. McCoy and Robert Frank. 2018. Phonologically informed edit distance algorithms for word alignment with low-resource languages. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2018*, pages 102–112.

Carlo Meloni, Shauli Ravfogel, and Yoav Goldberg. 2021. Ab Antiquo: Proto-language Reconstruction with RNNs. In *Proceedings of the 2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Online. Association for Computational Linguistics. To appear.

Ruslan Mitkov, Viktor Pekar, Dimitar Blagoev, and Andrea Mulloni. 2007. Methods for extracting and classifying pairs of cognates and false friends. *Machine Translation*, 21(1):29–53.

Andrea Mulloni. 2007. Automatic prediction of cognate orthography using support vector machines. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 25–30, Prague, Czech Republic. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Hermann Osthoff and Karl Brugmann. 1878. *Morphologische Untersuchungen auf dem Gebiete der indogermanischen Sprachen*. Hirzel, Leipzig, Germany.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Taraka Rama. 2016. Siamese convolutional networks for cognate identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1018–1027, Osaka, Japan. The COLING 2016 Organizing Committee.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

T. D. Singh and A. Vellintihun Hujon. 2020. Low Resource and Domain Specific English to Khasi SMT and NMT Systems. In *Proceedings of the 2020 International Conference on Computational Performance Evaluation (ComPE)*, pages 733–737, Shillong, India.

Inguna Skadiņa and Mārcis Pinnis. 2017. NMT or SMT: Case study of a narrow-domain English-Latvian post-editing project. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 373–383, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Eliel Soisalon-Soininen and Mark Granroth-Wilding. 2019. Cross-family similarity learning for cognate identification in low-resource languages. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1121–1130, Varna, Bulgaria. INCOMA Ltd.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 5926–5936, Long Beach, California, USA. PMLR.

Adam St Arnaud, David Beck, and Grzegorz Kondrak. 2017. Identifying cognate sets across dictionaries of related languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2519–2528, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Winston Wu and David Yarowsky. 2018. Creating large-scale multilingual cognate tables. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

A Appendix

A.1 Parameter Exploration Results

This section presents the hyperparameters which lead to the best dev BLEU across seeds, chosen during Section 4.2.3, and used for all subsequent experiments.

Model	Learning rate	Batch size	Embed. dim	Hidden dim	#layers	Model specific
RNN						Attention type
ES→IT	0.005	65	20	54	1	Luong-dot
IT→ES	0.005	65	20	72	1	Luong-dot
ES→LA	0.001	10	24	72	4	Luong-dot
LA→ES	0.005	100	20	72	1	Luong-dot
IT→LA	0.001	10	24	72	2	Bahdanau-dot
LA→IT	0.001	10	20	72	2	Luong-dot
Multilingual	0.001	10	24	72	2	Luong-general
Transformers						#heads
ES→IT	0.005	65	24	54	1	1
IT→ES	0.005	30	24	54	1	3
ES→LA	0.005	65	24	54	1	2
LA→ES	0.001	10	24	72	4	2
IT→LA	0.001	10	24	72	4	3
LA→IT	0.005	65	24	72	2	3
Multilingual	0.005	30	24	72	4	3

Luong-dot and Luong-general refer respectively to the dot and general attentions in (Luong et al., 2015), while Bahdanau-dot refers to our own implementation of the attention from (Bahdanau et al., 2015), simplified using the dot product to compute attention weights introduced in (Luong et al., 2015). See the code implementation with this paper for more detail.

Table 6: Results of parameter exploration experiments for RNN and Transformer models.

A.2 Learning Embeddings

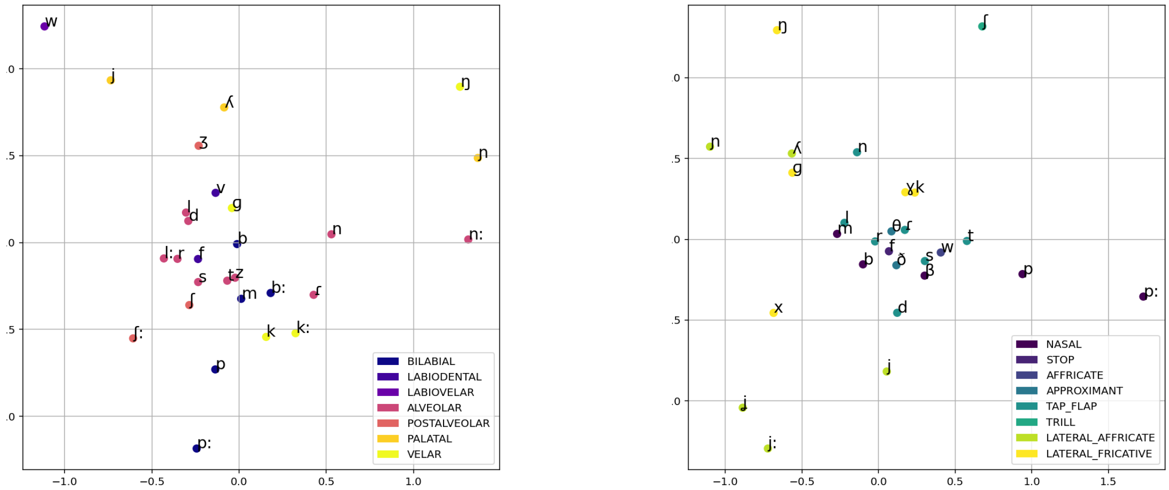

(a) IT consonants, coloured according to the place of articulation.

(b) ES consonants, coloured according to manner of articulation

Figure 4: PCA of phonetic source embeddings for an ES–IT RNN model.

Our learned embeddings seem to contain relevant phonetic information: their respective principal component analyses (PCA), when coloured according to place or manner of articulation for the consonants, and backness or height for the vowels, are coherently divided. The following examples are provided for an ES–IT RNN model, but similar results have been observed for our other languages and architectures.

Figure 4(a) shows the PCA of the learned source phonetic embeddings of one RNN model, for IT consonant phones, coloured according to place of articulation. It is radially organised, with a smooth transition between labio-dentals from the centre [b] to the bottom [pː], and from centre alveolar to left post alveolar. Figure 4(b) shows a similar PCA, this time for learned source embeddings of ES consonant phones, coloured according to manner of articulation. It seems coherently divided, with a transition from nasal sounds on the bottom right to lateral affricates and fricatives on the top left.

A.3 Average position of the best result among the 10-best results.

We present here at which position the best prediction (according to sentenceBLEU, from `sacreBLEU`) occurs amongst the 10-best predictions. For example, when going from Spanish *terroso* ‘muddy’, phonetised [tɛroso], to Italian *terroso* ‘muddy’, phonetised [terɾoːzo], the RNN predicted [teroːzo], [terɔːzo], [tɛrɾoːzo], **[terɾoːzo]**, [tɛrɾosː], [tɛrɾɔːzo], [tɛrɾɔsː], [terɾɔːzo], [tɛrosː], and [tɛrosːo]: the correct result corresponds to the 4th position.

For all multilingual models, we computed the sentence BLEU score for each of the 10-best predictions and saved the position of the highest scoring prediction. We averaged these positions for all words in the test set and calculated the standard deviation. Table 7 contains the full results, analysed in Section 6.2.

	IT→ES	ES→IT	IT→LA	LA→IT	ES→LA	LA→ES
SMT	1.08 ± 1.93	2.12 ± 2.55	2.01 ± 2.50	1.67 ± 2.30	2.49 ± 2.68	1.30 ± 2.14
Multilingual RNN	1.04 ± 2.03	1.67 ± 2.42	2.20 ± 2.54	1.63 ± 2.38	2.51 ± 2.68	1.38 ± 2.30
Multilingual Transformer	1.34 ± 2.17	1.94 ± 2.34	2.42 ± 2.65	2.17 ± 2.57	2.78 ± 2.73	1.64 ± 2.31

Table 7: Average position of the closest prediction to the reference amongst the **10**-best predictions.

What if This Modified That?
Syntactic Interventions via Counterfactual Embeddings

Mycal Tucker
MIT
mycal@mit.edu

Peng Qian
MIT
pqian@mit.edu

Roger P. Levy
MIT
rplevy@mit.edu

Abstract

Neural language models exhibit impressive performance on a variety of tasks, but their internal reasoning may be difficult to understand. Prior art aims to uncover meaningful properties within model representations via probes, but it is unclear how faithfully such probes portray information that the models actually use. To overcome such limitations, we propose a technique, inspired by causal analysis, for generating counterfactual embeddings within models. In experiments testing our technique, we produce evidence that suggests some BERT-based models use a tree-distance-like representation of syntax in downstream prediction tasks.

1 Introduction

Large neural models like BERT and GPT-3 have established a new state of the art in a variety of challenging linguistic tasks (Devlin et al., 2019; Brown et al., 2020). These connectionist models, trained on large corpora in a largely unsupervised manner, learn to map words into numerical representations, or embeddings, that support language-reasoning tasks. Fine-tuning these models on tasks like extractive question answering specializes these generic models into performant, task-specific models (Wolf et al., 2019).

In conjunction with the rise of these powerful neural models, researchers have investigated what the models have learned. Probes, tools built to reveal properties of a trained model, are a favored approach (Hall Maudslay et al., 2020; Conneau et al., 2018). For example, Hewitt and Manning (2019) have uncovered compelling evidence that several models encode syntactic information in their embeddings. That is, by passing embeddings through a trained probe, one may recover information about a sentence's syntax.

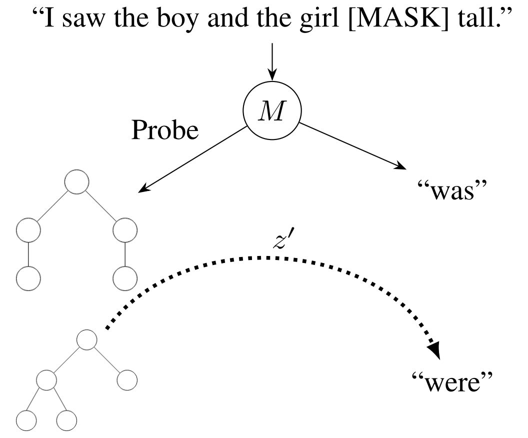

Figure 1: A language model, M, outputs predictions and a probe estimates properties from the model representation. We use probes to generate counterfactual representations, z', based on syntactic manipulations, revealing reasoning within the model.

Although these results are impressive, they fall short of clearly demonstrating what linguistic information the language models actually use. Syntactic information is present in sentences; that embeddings also encode syntax does not imply that a model uses syntactic knowledge.

In order to truly query a model's understanding, one must use causal analysis. Recently, several authors have done so by generating counterfactual data to test models (Kaushik et al., 2020; Goyal et al., 2019; Elazar et al., 2020). They either create new input data or ablate parts of embeddings and study how model outputs change. We extend this prior art via a new technique for generating counterfactual embeddings by using traditional probes to manipulate embeddings according to syntactic principles, as depicted in Figure 1. Because we conduct experiments with syntactically ambiguous inputs, we are able to measure how models respond to different valid parses of the same sentence instead of, for example, removing all syntactic information.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 862–875
August 1–6, 2021.

Thus, our technique uncovers not only what parts of its embeddings a model uses to represent syntax, but also how those parts influence downstream behavior.

Thus, in this work, we make two contributions. First, we develop a gradient-based algorithm to generate counterfactual embeddings, informed by trained probes. Second, in experiments using our technique, we find that the standard BERT model, trained on word-masking tasks, appears to leverage features of syntax in predicting masked words but that a BERT model fine-tuned for question-answering does not. In addition, these experiments yield new data to inform the ongoing debate on probe design.

2 Related Work

2.1 Neural Language Model Probes

Transformer-based models like GPT-3 and BERT have recently advanced the state of the art in numerous language-related problems (Brown et al., 2020; Devlin et al., 2019; Wolf et al., 2019). These large models appear to learn meaningful representations of words and sentences, enabling high performance when fine-tuned for a specific task.

In conjunction with these models, probes have been developed to uncover what principles models have learned. Such probes have been used in a wide variety of contexts, from image structure to syntax and semantics in language models (Alain and Bengio, 2018; Conneau et al., 2018; Hewitt and Manning, 2019; Coenen et al., 2019, among others). Our work uses two syntactic probes developed by Hewitt and Manning (2019) that map from model embeddings to predictions about word locations in a parse tree. These probes are simple by design – merely linear transformations – in order to prevent the probes themselves from doing parsing.

Recent work directly addresses the topic of probe simplicity. On the one hand, if probes are too expressive, they may reveal their own learning instead of a model's (Liu et al., 2019; Hewitt and Liang, 2019). On the other hand, Pimentel et al. (2020) argue from an information-theoretical perspective that more expressive probes are always preferable.

Our work differs from much prior art in probe design by leveraging causal analysis, which uses counterfactual data to test probes and models. This provides direct evidence of whether a model uses the same features as a probe, allowing us to experiment beyond linear probes (and indeed, we found that more complex probes offered an advantage in some cases).

2.2 Causal Analysis of Language Models

Motivated by the limitations of traditional, correlative probes, researchers have recently turned to causal analysis to better understand language models. Goyal et al. (2019) and Kaushik et al. (2020) generate counterfactual inputs to language models, while Vig et al. (2020) study individual neurons and attention heads to uncover gender biases in pre-trained networks.

Our work is most closely related to that of Elazar et al. (2020), who, as in this work, used probes to generate counterfactual embeddings within a network. Their amnesiac counterfactuals are generated by suppressing features in embeddings that a probe uses. In contrast, we use a continuous, gradient-based approach to generate counterfactuals, yielding insight into how features are used, as opposed to if they are used at all.

3 Technical Approach

3.1 Problem Formulation

We may characterize a transformer-based language model, M, trained on a specific task, as a function mapping from an input string, s, to an output y: $M(s) = y$. In order to reveal embeddings for analysis by probes, we may decompose M into two functions: M_{k-} and M_{k+}. M_{k-} represents the first k layers of the model; M_{k+} represents the layers of M after layer k; M is the composition of these functions: $M = M_{k+} \circ M_{k-}$. We label the embeddings output by M_{k-} as z_k. This decomposition of models to reveal internal embeddings mirrors the formulation for layer-specific probes (Hewitt and Manning, 2019). A probe may be defined as a function f_p that maps from an embedding, z_k, to a predicted property $\hat{p}$ about the input, s: $f_p(M_{k-}(s)) = \hat{p}$. (For the remainder of this paper, we focus on syntactic probes, but our reasoning may be extended to other linguistic properties.)

We may define two, potentially overlapping, subsets of the features of z_k by considering different uses of z_k. First, we may define z_p as the features of z_k that the probe uses in predicting $\hat{p}$ (for example, when using a linear probe, z_p is the projection of z_k onto the probe subspace). Assuming good syntactic probe performance, z_p is necessarily informative of the input's syntax. We likewise

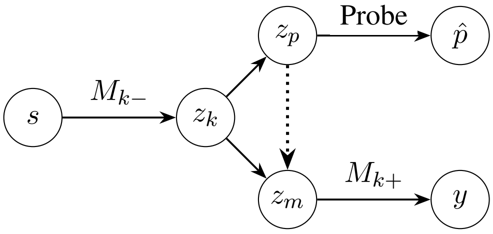

Figure 2: M_{k-} yields a representation, z_k. z_p and z_m are subsets of the features of z_k used by the probe and M_{k+}. We measured the causal link between z_p and z_m.

defined z_m as the features of z_k that M_{k+} uses in producing the model output. These two, potentially overlapping, representations of z_k are shown in Figure 2, inspired by causal diagrams by Pearl and Mackenzie (2018). We seek to discover if there is a causal link between z_p and z_m.

For some tasks, such a link should exist. For example, a question-answering model's response to "I shot the elephant wearing my pajamas. Who wore the pajamas?" should depend upon the inferred sentence syntax (e.g., if the probe predicts that "wearing my pajamas" modifies "the elephant," the model should output "the elephant"). Thus, the probe and model outputs should "agree" according to syntactic principles. Furthermore, if a causal link between z_p and z_m exists, changing z to produce a new prediction of syntax should change the model output to agree with the probe (e.g., if the probe predicts that "wearing my pajamas" now modifies "I," the model should now output "I"). In this work, therefore, we study whether a link between z_p and z_m exists and, if it does, to what extent it corresponds with linguistic principles.

3.2 Generating Counterfactual Embeddings via Gradient Descent

To study such a link, we must generate counterfactual embeddings, z', that modify probe outputs, starting from normal embeddings z_k. We borrow the term "counterfactual" from causal literature because z' represents what z_k would have been if z_p had been different (Pearl and Mackenzie, 2018). We were particularly interested in finding z' that changed both probe and model outputs; if z' only changed probe outputs, that could indicate that the probe was over-interpreting model embeddings (e.g., acting as a parser instead of a probe).[1]

We developed a gradient-based method to generate z' that changed the probe output. We assumed that, given the probe function, f_p, a loss, L, and the correct property value (e.g., parse), p, one could compute the gradient of the loss with respect to the probe inputs: $\nabla_{z'} L(f_p(z'), p)$. Neural network probes obey such differentiability assumptions.

Given z_k and p, we constructed a counterfactual embedding , z', by initializing z' as a z_k generated by the model and updating z' via gradient descent of the loss. Updating z' may be terminated based on various stopping criteria (e.g., local optimality, loss below a threshold, etc.), yielding the final counterfactual z'. Assuming non-zero gradients, this technique produces z's that, by design, change the probe outputs. In experiments, we studied how z's changed model outputs when passed through M_{k+}.

Although our technique bears some resemblance to gradient-based adversarial attacks (Szegedy et al., 2014), it may more broadly be thought of as guided search in a latent space. Adversarial images are often characterized by changes that are imperceptible to humans but change model behaviors to be incorrect. In contrast, we seek to find embeddings that change both probe and language model outputs. Furthermore, by design, we use syntactically ambiguous sentences in experiments and generate counterfactuals according to valid parses. Thus, unlike adversarial attacks on images that seek to switch model classification to an incorrect class, we merely guide embeddings among a set of valid interpretations. Lastly, even uncovering instances of embeddings that change probe outputs but not the model's is important, as it indicates a misalignment of probe and model reasoning.

4 Experiments

In the previous section, we proposed a technique for generating counterfactual embeddings; here, we detailed the experiments we conducted to measure the effect of using such embeddings. Inputs to our technique included the base language models, probes, test sentences, and different ground-truth parses to generate the counterfactual embeddings.

4.1 Model Tasks

We tested our technique on two BERT models trained on different tasks: masked word prediction and extractive question answering.

In the masked word prediction task, a model is given a sentence, S, comprising words $(s_0, s_1, \ldots$

---

[1] We did not study z' that only modified the model outputs, although this could be a promising avenue for future work.

Model	Corpus	Parse	Example Input
Mask	Coord.	Plur.	The woman saw (the boy and the dog) [MASK] falling.
		Sing.	(The woman saw the boy) and (the dog [MASK] falling.)
	NP/Z	Adv.	When the dog scratched (the vet [MASK] ran.)
		Noun	(When the dog scratched the vet) [MASK] ran.
QA	RC	Conj.	The ((smart women and rich men) who were desperate) bribed the judge.
		NP2	The (smart women) and (rich men who were desperate) bribed the judge.
	NP/VP	VP	The girl saw (the boy) with the telescope.
		NP2	The girl saw (the boy with the telescope.)

Table 1: Experiment design for different language models and test corpora, with illustrative sentences, decorated with auxiliary parentheses to reveal structure. The parentheses were not included in the actual corpora.

`[MASK]`, ..., s_n) and must predict the word at the location marked by `[MASK]`. For example, given a sentence, [‘The’, ‘children’, ‘went’, ‘out’, ‘to’, `[MASK]`, ‘.’], a correct answer might be ‘play.’ We used huggingface’s “bert-large-uncased-whole-word-masking” model, which was trained on masked word and next-sentence prediction, and referred to it as the “Mask” model (Wolf et al., 2019).

Extractive question answering is framed by Wolf et al. (2019) as follows: given a sentence, S, comprising word tokens $(s_0, s_1, ...s_n)$ and a question, identify the start and end tokens $(s_i, s_j; 0 \leq i \leq j \leq n)$ denoting a contiguous stretch of the sentence that answers the question. For example, given the sentence [‘I’, ‘ate’, ‘two’, ‘apples’, ‘.’] and the question “How many apples did I eat?,” a correct answer could be [2, 2] (“two”) or [2, 3] (“two apples”). We used huggingface’s “BertForQuestionAnswering,” already fine-tuned on the SQuAD dataset, and referred to the model as QA (Wolf et al., 2019; Rajpurkar et al., 2016).

4.2 Probes

Our technique for generating counterfactual embeddings depended on probes, so we used four different syntactic probes drawn from prior art and our own design.

The depth probe from Hewitt and Manning (2019) maps from embeddings to predictions over words’ depths in a sentence’s parse tree. The distance probe, given a pair of words, predicts the distance between the words in the parse tree (i.e., how many edges must be traversed). Both probes consist of a linear transformation from embedding to prediction.

We further implemented “deep” versions of the distance probe by creating two- and three-layer, non-linear probes trained on the distance task. These models used ReLU activations, with hidden dimension 1024, but otherwise used the same input and output format as the linear distance probe. (Experiments conducted with “deep” versions of the linear depth probe produced similar results to those of the normal depth probe and are therefore omitted.)

4.3 Evaluation Corpora

We used four corpora for evaluating the Mask and QA models, as summarized in Table 1.

4.3.1 Mask Test Corpora

For the Mask model, we used two test suites composed of sentences whose structural ambiguity was resolved by a masked word.

The first corpus, dubbed “Coordination,” comprised sentences that took the form “The NN1 VERB the NN2 and the NN3 [MASK] ADJ.” Such sentences may be interpreted in at least two ways by inserting either “was” or “were” in the masked location. The former reflects a conjunction of clauses (e.g., “The woman saw the boy and the dog was falling.”), whereas the latter reflects a conjunction of noun phrases (e.g., “The woman saw the boy and the dog were falling.”) Sentences were generated through combinations of NN1 [man, woman, child], VERB [saw, feared, heard], NN2 [boy, building, cat], NN3 [dog, girl, truck], and ADJ [tall, falling, orange], yielding 243 sentences, each with two parse trees dubbed “singular” or “plural,” depending on the grammatical verb type.

The second corpus, dubbed the NP/Z corpus, was inspired by classic psycholinguistic studies of the garden-pathing effect in online sentence processing (Frazier and Rayner, 1982; Tabor and Hutchins, 2004). Each sentence in the corpus took the form “When the NN1 VERB1 the NN2

[MASK] VERB2.” Without knowing the masked word, it is unclear if NN2 is the object of the subordinate clause or the subject of the main clause. For example, in the sentence “When the dog scratched the vet [MASK] ran,” either an adverb (e.g., “immediately”) or a noun (e.g., “she”) would be permitted but correspond to different parses. We created such parse trees and dubbed the first type “Adv.” and the second type “Noun.” We used the 24 sentences from Tabor and Hutchins (2004) that fit our template, and supplemented the dataset with 36 sentences of our own, generated by iterating over all combinations of NN1 [dog, child], NN2 [vet, boy, girl], VERB1 [scratched, bit], and VERB2 [ran, screamed, smiled]. (Augmenting the dataset was needed to increase the statistical analysis power, and plotting the 24 and 36 sentences separately established that they produced similar results.)

4.3.2 QA Test Corpora

For the QA model, we created two test suites. First, the “RC” corpus used sentences composed of a conjunction of nouns modified by a relative clause. All sentences took the form “The ADJ1 NN1 and ADJ2 NN2 who were ADJ3 VERB the NN3. Who was ADJ3?” For example, one sentence was “The smart women and rich men who were desperate bribed the judge. Who was desperate?” By construction, it was unclear if the relative clause modified the conjunction of the first and second noun phrases (The ADJ1 NN1 and ADJ2 NN2) or merely the second noun phrase (ADJ2 NN2). For each sentence, we generated two parses: “Conj. Parse” and “NP2 Parse,” corresponding to the former and latter. We generated sentences by iterating over all combinations of values for ADJ1 [smart, rich, tall, poor], NN1 [men, women], ADJ2 [smart, rich, tall, poor], NN2 [men, women], ADJ3 [corrupt, desperate], VERB [bribed, paid], and NN3 [politician, judge], excluding sentences in which NN1 and NN2 or ADJ1 and ADJ2 were the same. This produced 192 sentences, each with two parses.

Lastly, the “NP/VP” corpus used sentences with ambiguous prepositional phrase attachment. Inspired by sentences like “The girl saw the boy with the telescope,” we generated inputs with the template “The NN1 VERB the NN2 with the NN3. Who had the NN3?” We iterated through combinations of NN1 [man, woman, child], NN2 [man, woman, boy, girl, stranger, dog], and VERB-NN3 pairs [saw-telescope, poked-stick, thanked-letter, fought-knife, dressed-hat, indicated-ruler, kicked-shoe, welcomed-gift, buried-shovel], removing duplicate NN1 and NN2, yielding 144 inputs. Each input used two parses indicating the prepositional phrase modifying VP or NP2 (“the” and NN2).

4.4 Generating Embeddings

For all models, probes, and parses trees for each sentence, we generated counterfactual embeddings by initializing a counterfactual embedding, z', as the original model embedding for the input sentence, z_k, and running an Adam optimizer, with learning rate 0.0001, to minimize the probe loss (using a particular probe and parse tree) (Kingma and Ba, 2014). Recall that the optimizer updated z' rather than the probe parameters.

The optimizer used a patience value of 5000: it continued updating z' until the probe loss failed to improve for 5000 consecutive gradient updates. Using a patience-based termination condition (as opposed to setting a loss threshold or maximum number of updates, for example) was task-agnostic and seemed to be robust to a wide range of patience values. Brief experimentation with patience values from 50 to 5000 yielded similar results. On a Linux desktop with an Nvidia GEForce RTX 2080 graphics card, generating a single counterfactual took less than 1 minute, and the process was easily parallelized to batches of 80 embeddings, reducing the mean computation time to under one second.

For both the QA and Mask models, we trained all probe types (depth, distance, 2-layer dist, and 3-layer dist) on each of the model’s 25 layers. We used 5000 entries from the Penn Treebank (PTB) for training, with the standard validation and test sets of nearly 4000 entries used for early stopping and evaluation, respectively (Marcus et al., 1993).

4.5 Metrics

We used two sets of metrics in our experiments. First, we measured probe performance using the Root Accuracy, UUAS, and Spearman Coefficient metrics used by Hewitt and Manning (2019) and refer to their work for details. Intuitively, these metrics captured how accurately the probes predicted aspects of syntactic structure from embeddings.

Second, we measured changes in model outputs when using counterfactual embeddings. The Mask model produced a probability distribution over more than 30,000 possible words for the masked location, but we restricted our attention to only a subset of those words, dubbed “candidates.” (We normalized predictions among the set of candidates,

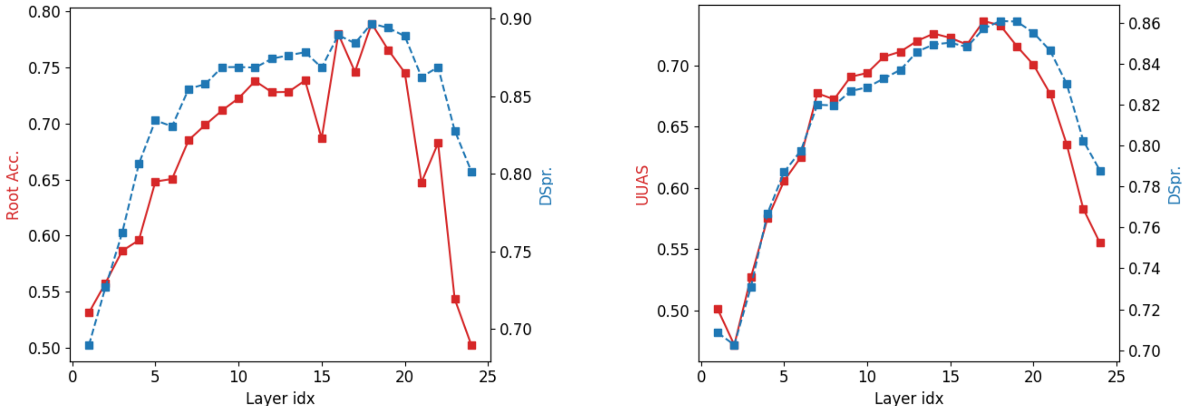

(a) Depth Probe (b) Dist Probe

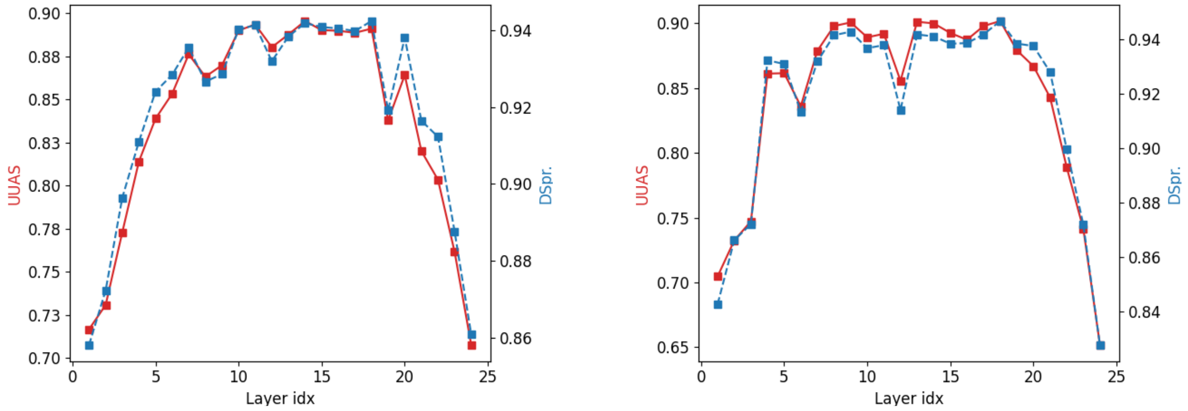

(c) 2-layer Dist Probe (d) 3-layer Dist Probe

Figure 3: All trained probes for the QA model exhibited high performance on the PTB corpus.

producing a proper probability distribution.) In the Coordination corpus, we used 5 candidates: [“was,” “is,” “were,” “are,” “as”]. In the NP/Z corpus, we generated the set of candidates by collecting the most likely predictions over the corpus, using both original and counterfactual embeddings. This set of 18 words is shown in the x-axis of Figure 6. For both corpora, we partitioned the candidates into two sets, depending upon which parse they implied, and measured the sum of the probabilities of words in each set. If counterfactual embeddings caused the models to change the type of word they predicted, we would expect to see a change in these sums.

For the QA model, we similarly measured changes in probabilities among sets of words, but in this case we focused on the predicted start location of the answer. Recall that the QA model produced two distributions over words, indicating its predictions over where the answer started and ended. Consider an example input, drawn from the RC corpus: “The smart women and rich men who were desperate bribed the politician. Who was desperate?” Two reasonable answers might be “The smart women and rich men” or “rich men,” corresponding to QA outputs with identical end words, but differing start words. We therefore created two partitions of starting words to consider: those belonging to the first noun phrase (“The smart women”) or the second noun phrase (“rich men”). We then measured the summed start probabilities of words in each partition. We did not normalize these probabilities, as the QA model rarely predicted start words outside these two partitions with more than 1% probability.

In all experiments, we employed one-sided Wilcox Signed-Rank tests, non-parametric tests for pairwise data, when determining the significance at $(p < 0.01)$. The parses were viewed as “treatments” for the same embedding. We compared the effect of using counterfactual instead of original embeddings, as well as the effect of using different parses to generate counterfactual embeddings.

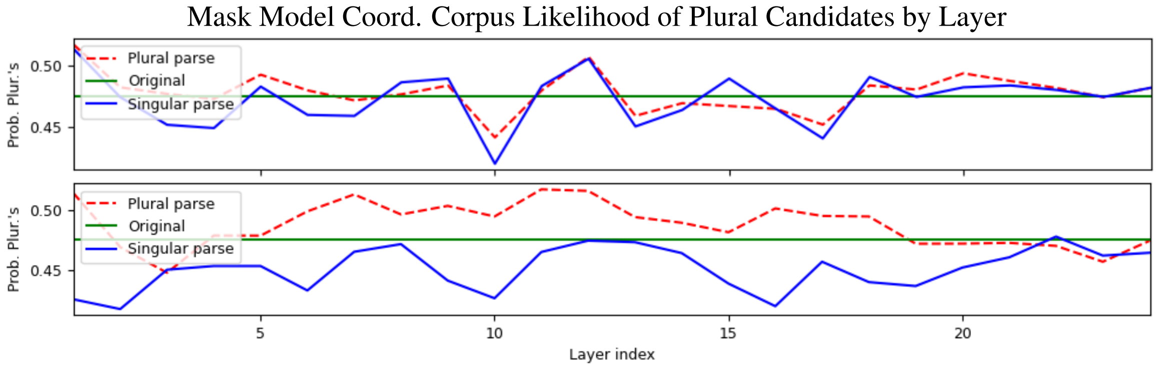

Figure 4: Mean probability of plural candidates using the depth probe (top) or the 3-layer dist probe (bottom), using original or counterfactual embeddings, in the Coordination corpus. Using a parse that implied plural words increased the probability of plural words when using the 3-layer dist probe.

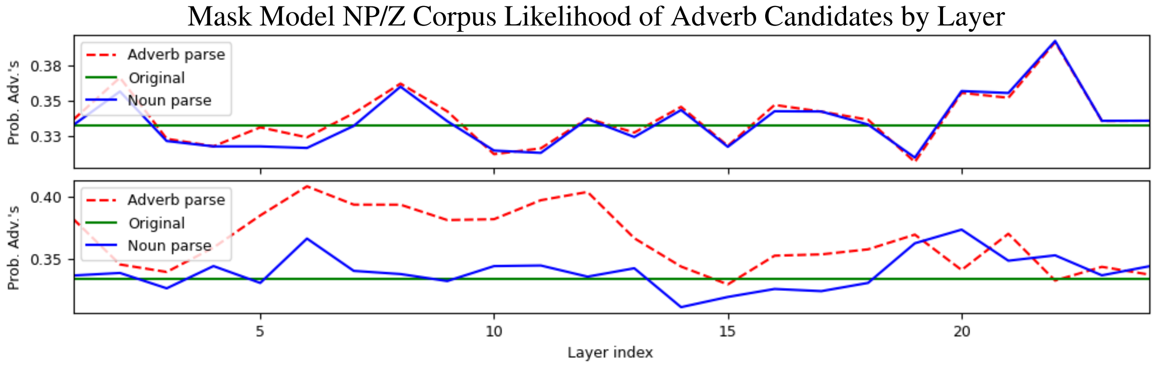

Figure 5: Mean probability of adverb candidates in the NP/Z corpus, using original and counterfactual embeddings generated by the depth (top) and 3-layer dist probes (bottom).

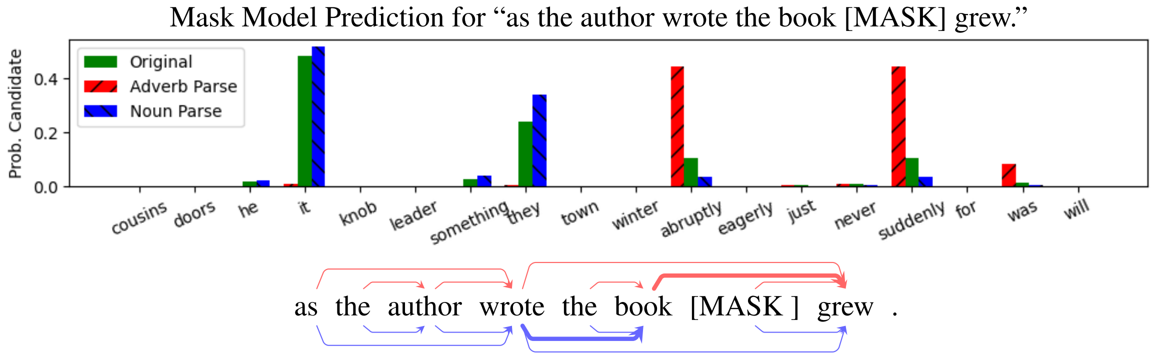

Figure 6: Given a sentence from the NP/Z corpus, the Mask model originally predicted "it" or "they," but using counterfactuals from the 5^{th} layer 3-layer dist probe changed predictions to favor nouns (cousins - winter) or adverbs (abruptly - suddenly). Visualizing the word dependencies revealed that the Adverb parse (top, red) and Noun parse (bottom, blue) induced different dependencies (differences in bold), as expected.

5 Results

Our results indicated that our probes performed well, as evaluated by performance metrics from prior art. However, we found that only some combinations of probe types and BERT models generated counterfactuals that altered the model's outputs according to syntactic principles.

5.1 Probe Performances

Measured on the PTB test set, the probe performance metrics confirmed that the probes predicted aspects of syntactic structure well (Marcus et al., 1993). Plots of performance, similar to those by Hewitt and Manning (2019), for probes trained on QA model embeddings are included in Figure 3.[2]

---

[2] All probe metrics are plotted in the appendix.

For both models and all probe types, we found that the probes were able to achieve high performance, indicating that both the Mask and QA models encoded syntactic information in their embeddings.

We also observed the unsurprising trend that multi-layered, non-linear distance probes outperformed the linear distance probe. This raised the question, if different probes exhibited different performance for the same model, which probe should be used to deduce model behavior? Injecting counterfactual embeddings generated by different probes helped us answer this question.

5.2 Mask Counterfactual Results

Next, we found that using the distance-based probes to generate counterfactual embeddings in the Mask model consistently produced the desired effect by shifting the model's prediction of the masked word according to syntactic principles, and that the multi-layer distance probes performed better than the linear probe.

We plotted the mean effect of counterfactual embeddings for the Coordination and NP/Z corpora in Figures 4 and 5, respectively.[3] Each plot depicts the mean prediction likelihood of one of the partitions of candidates (plural for Coord. corpus, adverbs for NP/Z), using original or counterfactual embeddings. Figure 4 shows results using the depth and 3-layer distance probes in the Coord. corpus: the depth probe failed to produced consistent changes in word probabilities, but embeddings generated by the 3-layer dist probe did exhibit the desired effect. The change in probability of plural words when using the plural parse was significantly positive for layers 6 through 14 (among others) and greater than the change when using the singular parse for layers for 4 through 21.

Similar results were observed using the 3-layer distance probe for the NP/Z corpus, as shown in Figure 5. The net increase in probability for adverbs when using the adverb parse was significantly greater than when using the NP2 parse for layers 5 through 19 and was positive for layers 4 through 13.

We examined an example sentence from the NP/Z corpus in Figure 6 in greater depth. The 18 words displayed along the x axis were the candidate words whose probabilities we calculated in the NP/Z corpus. As expected, using the Adv. parse increased the likelihood of adverbs like "suddenly," while using the Noun parse increased the likelihood of nouns like "it" or "they." Lastly, the bottom part of Figure 6 shows the dependency trees for the counterfactuals generated for each parse (see Hewitt and Manning (2019) for details on creating such trees). These trees reflected the dependencies of the parses that generated the counterfactuals, indicating that our technique changed embeddings in the way we intended.

Together, the results from both corpora, revealed that distance-, but not depth-, based probes elicited the desired response from the Mask model, which suggests that it leverages a distance-based representation of syntax in its reasoning.

5.3 QA Counterfactual Results

Lastly, we examined the effect of using counterfactual embeddings in the QA model. Compared to the Mask model, we found smaller and less consistent results, suggesting that the QA model may not use syntax.

Taking the mean across sentences in the corpus, we plotted the mean starting probabilities of words in each sentence's first noun phrase (as explained earlier in Section 4.5). These values reflect whether the model predicted NP1 should be included in the answer (e.g., "The smart women and rich men" instead of merely "rich men"). We plotted the results for the 3-layer dist probe, the best-performing probe for the Mask model, on both QA corpora in Figure 7. In both plots, the choice of layer in which counterfactuals were inserted had a greater effect than which parse was used to generate the counterfactuals – a sign of poor performance. Depth and other distance probes performed no better.

Visualizing dependency trees for QA embeddings revealed that the counterfactual embeddings induced the correct structure, indicating that the QA model simply did not use such structure in downstream predictions. Furthermore, given the success of our probes and technique with the Mask model, these poor results for the QA model suggest (but admittedly cannot definitely prove) that it may not have learned to use the syntactic information detected by the probes. This theory is consistent with prior art that finds that fine-tuning on specific tasks, as was done for the QA model, worsens the alignment between model and human representations of language (Gauthier and Levy, 2019).

---

[3] Plots for the effects of counterfactuals for all probes, models, and test corpora were included in the appendix.

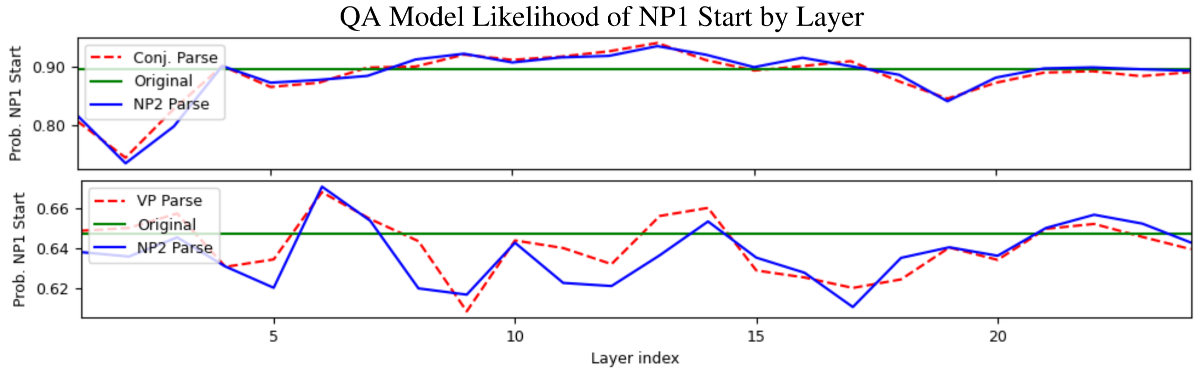

Figure 7: Mean effects of using counterfactual updates from the 3-layer dist probe on the QA model for the RC (top) and NP/VP (bottom) corpora.

6 Conclusion

In this work, we proposed and evaluated a new technique for producing counterfactual embeddings that tested syntactic understanding of models and probes. On the one hand, we uncovered clear evidence supporting a causal link between a distance-based representation of syntax and the outputs of a masked-word model. On the other hand, depth-based manipulations of embeddings had little effect, and we found no evidence that the BERT model finetuned on question-answering uses the syntactic information used by probes.

Our work is merely an initial step in the direction of causal analysis of language models. Developing new probes, backed by causal evidence, could increase our understanding of such models. In particular, our findings that multi-layered probes outperformed linear probes indicate that the prior guidance of simpler probes being preferable may be misleading. Furthermore, as the discrepancy between distance- and depth-based probes revealed, developing a large suite of probe types that focus on different features may be necessary to reveal a model's reasoning. In tandem with probe development, more sophisticated counterfactual generation techniques than our gradient-based method could produce more interesting counterfactuals for evaluation.

Acknowledgments

We thank the reviewers for their thoughtful comments, in particular regarding adversarial attacks. We thank Professors Julie Shah and Jacob Andreas for ongoing discussions and guidance. Lastly, we thank John Hewitt and Christopher Manning for releasing high-quality, reproducible code, enabling us to rapidly build upon their syntactic probe codebase.

RPL gratefully acknowledges support from the MIT–IBM Artificial Intelligence Laboratory and MIT's Quest for Intelligence.

References

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of bert. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8594–8603. Curran Associates, Inc.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2020. Amnesic Probing: Behavioral Explanation with Amnesic Counterfactuals. *arXiv e-prints*, page arXiv:2006.00995.

Lyn Frazier and Keith Rayner. 1982. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14:178–210.

Jon Gauthier and R. Levy. 2019. Linking artificial and human neural representations of language. In *EMNLP/IJCNLP*.

Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. 2019. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*.

Rowan Hall Maudslay, Josef Valvoda, Tiago Pimentel, Adina Williams, and Ryan Cotterell. 2020. A tale of a probe and a parser. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7389–7395, Online. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Judea Pearl and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*, 1st edition. Basic Books, Inc., USA.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Whitney Tabor and Sean Hutchins. 2004. Evidence for self-organized sentence processing: Digging-in effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(2):431.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Appendix: Complete Performance Plots

In this appendix, we included additional figures that we were unable to include within the main paper limits.

First, we depicted the probe performance characteristics for the 4 probes types we used in all our experiments: the depth, dist, 2-layer dist, and 3-layer dist probes. Each type of probe was trained for both the QA and Mask models. Evaluation of these probes was plotted in Figure 8.

Next, we reported the effect of counterfactual embeddings generated for each model, corpus, and probe type. Given the 4-page limit for the appendix, further plots breaking down the NP/Z corpus, for example, or depicting performance for multi-layered depth probes were not included. These plots merely confirmed trends already present in the data: that depth-based probes did not produce useful counterfactuals, and that the curated and automatically-generated sentences that formed the full NP/Z corpus yielded similar results.

In general, we observed small effects for counterfactuals in the QA Model (Figures 11 and 12), but consistent effects in the Mask Model (Figures 9 and 10). Within the Mask model results, we also observed that the distance probe (2^{nd} row) outperformed the depth probe (1^{st} row), and that the multi-layer distance probes (3^{rd} and 4^{th} rows) outperformed the linear distance probe.

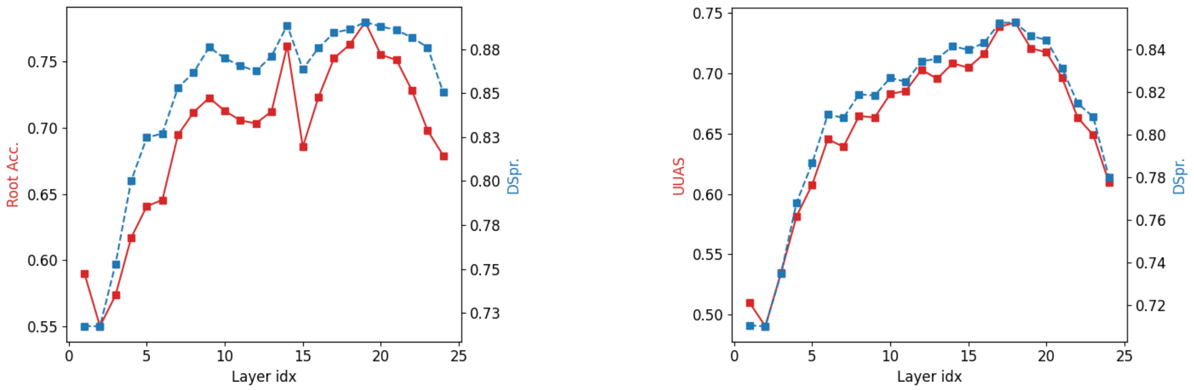

(a) Mask Model Depth Probe

(b) Mask Model Dist Probe

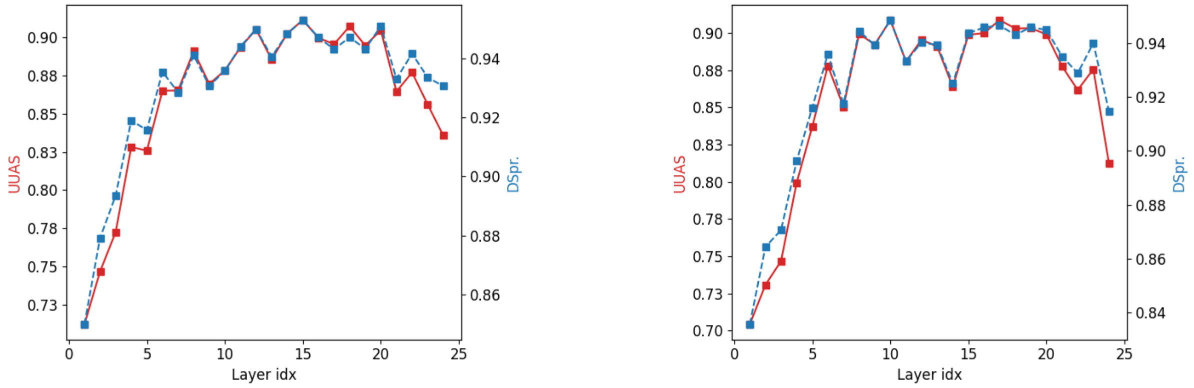

(c) Mask Model 2-Layer Dist Probe

(d) Mask Model 3-Layer Dist Probe

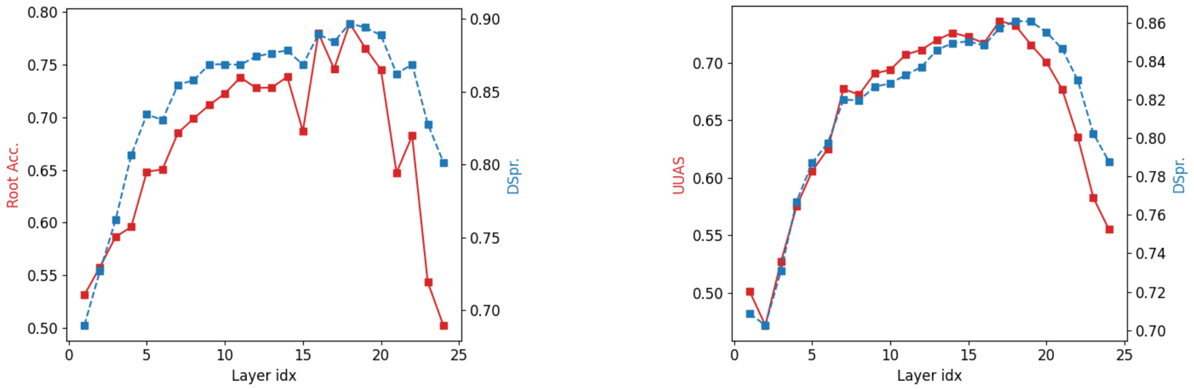

(e) QA Model Depth Probe

(f) QA Model Dist Probe

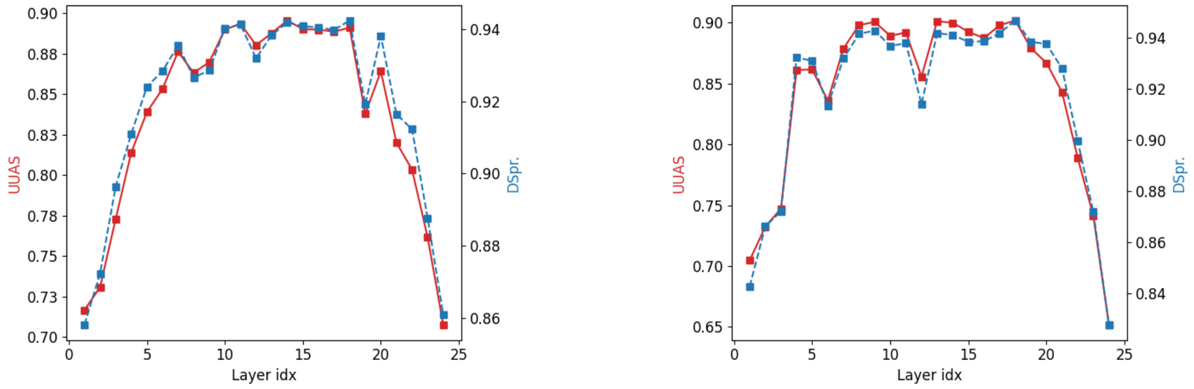

(g) QA Model 2-Layer Dist Probe

(h) QA Model 3-Layer Dist Probe

Figure 8: Probe performances for the Mask and QA models. Note the changed y axes, demonstrating improved performance for the multi-layer distance probes.

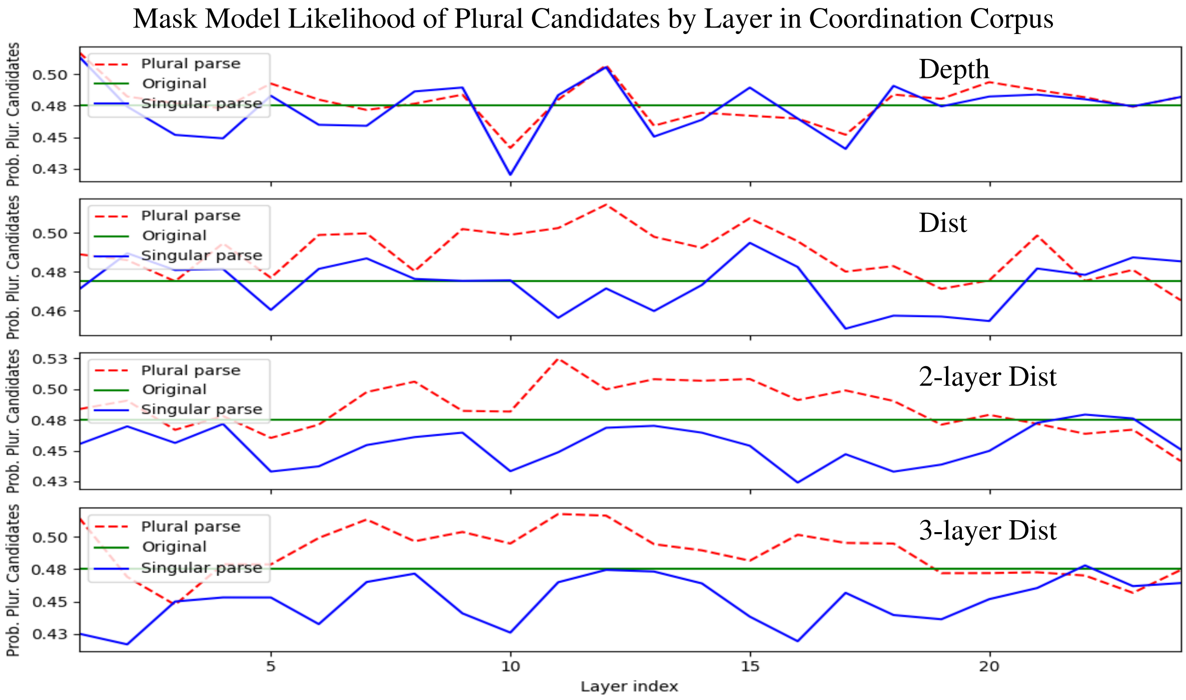

Figure 9: Mask model performance on the Coordination corpus. When using distance-based probes, the plural parse increased the likelihood of plural candidates being predicted, and the singular parse increased the likelihood of singular candidates being predicted.

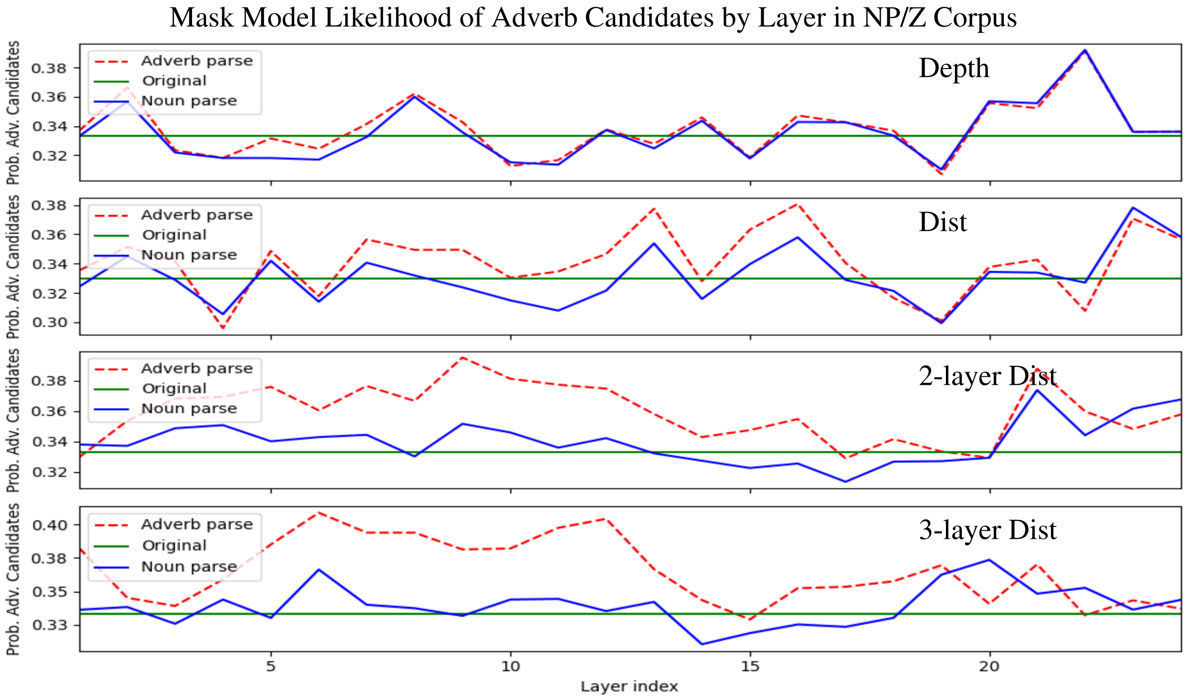

Figure 10: Mask model performance on the NP/Z corpus. Distance-based probes, and in particular multi-layer distance probes, changed model outputs according to syntactic principles.

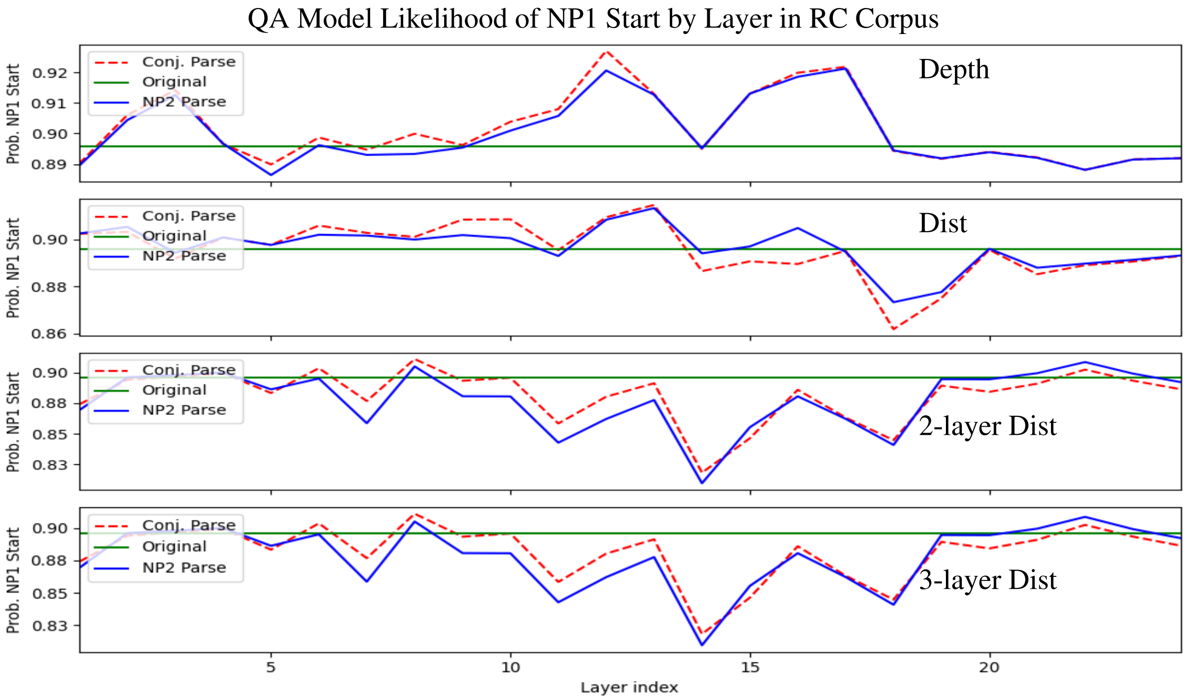

Figure 11: QA model performance on the RC corpus. No probe created consistent effects via counterfactual embeddings.

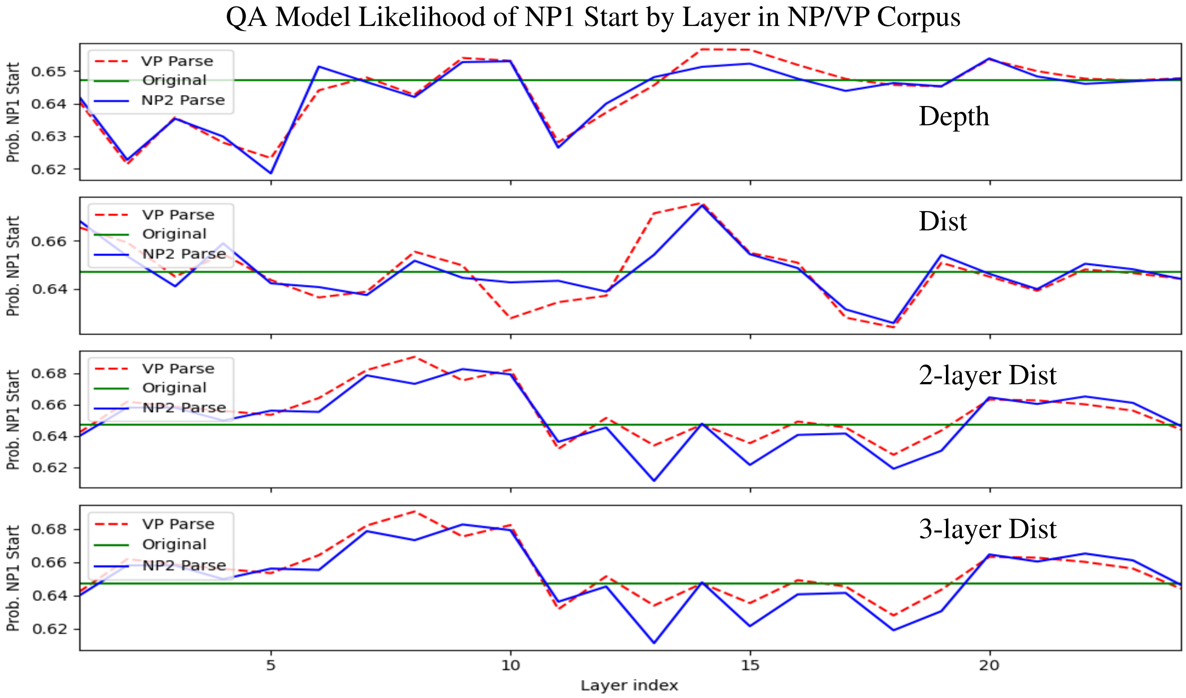

Figure 12: QA model on the NP/VP corpus. As in Figure 11, no probe created consistent effects.

Investigating Text Simplification Evaluation

Laura Vásquez-Rodríguez[1], **Matthew Shardlow**[2], **Piotr Przybyła**[3], **Sophia Ananiadou**[1]
[1]National Centre for Text Mining,
The University of Manchester, Manchester, United Kingdom
[2]Department of Computing and Mathematics,
Manchester Metropolitan University, Manchester, United Kingdom
[3]Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
{laura.vasquezrodriguez, sophia.ananiadou}@manchester.ac.uk
m.shardlow@mmu.ac.uk piotr.przybyla@ipipan.waw.pl

Abstract

Modern text simplification (TS) heavily relies on the availability of gold standard data to build machine learning models. However, existing studies show that parallel TS corpora contain inaccurate simplifications and incorrect alignments. Additionally, evaluation is usually performed by using metrics such as BLEU or SARI to compare system output to the gold standard. A major limitation is that these metrics do not match human judgements and the performance on different datasets and linguistic phenomena vary greatly. Furthermore, our research shows that the test and training subsets of parallel datasets differ significantly. In this work, we investigate existing TS corpora, providing new insights that will motivate the improvement of existing state-of-the-art TS evaluation methods. Our contributions include the analysis of TS corpora based on existing modifications used for simplification and an empirical study on TS models performance by using better-distributed datasets. We demonstrate that by improving the distribution of TS datasets, we can build more robust TS models.

1 Introduction

Text Simplification transforms natural language from a complex to a simple format, with the aim to not only reach wider audiences (Rello et al., 2013; De Belder and Moens, 2010; Aluisio et al., 2010; Inui et al., 2003) but also as a preprocessing step in related tasks (Shardlow, 2014; Silveira and Branco, 2012).

Simplifications are achieved by using parallel datasets to train sequence-to-sequence text generation algorithms (Nisioi et al., 2017) to make complex sentences easier to understand. They are typically produced by crowdsourcing (Xu et al., 2016; Alva-Manchego et al., 2020a) or by alignment (Cao et al., 2020; Jiang et al., 2020). They are infamously noisy and models trained on these give poor results when evaluated by humans (Cooper and Shardlow, 2020). In this paper we add to the growing narrative around the evaluation of natural language generation (van der Lee et al., 2019; Caglayan et al., 2020; Pang, 2019), focusing on parallel text simplification datasets and how they can be improved.

Why do we need to re-evaluate TS resources? In the last decade, TS research has relied on Wikipedia-based datasets (Zhang and Lapata, 2017; Xu et al., 2016; Jiang et al., 2020), despite their known limitations (Xu et al., 2015; Alva-Manchego et al., 2020a) such as questionable sentence pairs alignments, inaccurate simplifications and a limited variety of simplification modifications. Apart from affecting the reliability of models trained on these datasets, their low quality influences the evaluation relying on automatic metrics that requires gold-standard simplifications, such as SARI (Xu et al., 2016) and BLEU (Papineni et al., 2001).

Hence, evaluation data resources must be further explored and improved to achieve reliable evaluation scenarios. There is a growing body of evidence (Xu et al., 2015) (including this work) to show that existing datasets do not contain accurate and well-constructed simplifications, significantly impeding the progress of the TS field.

Furthermore, well-known evaluation metrics such as BLEU are not suitable for simplification evaluation. According to previous research (Sulem et al., 2018) BLEU does not significantly correlate with simplicity (Xu et al., 2016), making it inappropriate for TS evaluation. Moreover, it does not correlate (or the correlation is low) with grammaticality and meaning preservation when performing syntactic simplification such as sentence splitting. Therefore in most recent TS research BLEU has not been considered as a reliable evaluation metric. We use SARI as the preferred method for TS eval-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 876–882
August 1–6, 2021.

uation, which has also been used as the standard evaluation metric in all the corpora analysed in this research.

Our contributions include 1) the analysis of the most common TS corpora based on quantifying modifications used for simplification, evidencing their limitations and 2) an empirical study on TS models performance by using better-distributed datasets. We demonstrate that by improving the distribution of TS datasets, we can build TS models that gain a higher SARI score in our evaluation setting.

2 Related Work

The exploration of neural networks in TS started with the work of Nisioi et al. (2017), using the largest parallel simplification resource available (Hwang et al., 2015). Neural-based work focused on state-of-the-art deep learning and MT-based methods, such as reinforcement learning (Zhang and Lapata, 2017), adversarial training (Surya et al., 2019), pointer-copy mechanism (Guo et al., 2018), neural semantic encoders (Vu et al., 2018) and transformers supported by paraphrasing rules (Zhao et al., 2018).

Other successful approaches include the usage of control tokens to tune the level of simplification expected (Alva-Manchego et al., 2020a; Scarton and Specia, 2018) and the prediction of operations using parallel corpora (Alva-Manchego et al., 2017; Dong et al., 2020). The neural methods are trained mostly on Wikipedia-based sets, varying in size and improvements in the quality of the alignments.

Xu et al. (2015) carried out a systematic study on Wikipedia-based simplification resources, claiming Wikipedia is not a quality resource, based on the observed alignments and the type of simplifications. Alva-Manchego et al. (2020a) proposed a new dataset, performing a detailed analysis including edit distance and proportion of words that are deleted, inserted and reordered, and evaluation metrics performance for their proposed corpus.

Chasing the state-of-the-art is rife in NLP (Hou et al., 2019), and no less so in TS, where a SARI score is too often considered the main quality indicator. However, recent work has shown that these metrics are unreliable (Caglayan et al., 2020) and gains in performance according to them may not deliver improvements in simplification performance when the text is presented to an end user.

3 Simplification Datasets: Exploration

3.1 Data and Methods

In the initial exploration of TS datasets, we investigated the training, test and validation subsets (when available) of the following: WikiSmall and WikiLarge (Zhang and Lapata, 2017), TurkCorpus (Xu et al., 2015), MSD dataset (Cao et al., 2020), ASSET (Alva-Manchego et al., 2020a) and WikiManual (Jiang et al., 2020). For the WikiManual dataset, we only considered sentences labelled as “aligned”.

We computed the number of changes between the original and simplified sentences through the **token edit distance**. Traditionally, edit distance quantifies character-level changes from one character string to another (additions, deletions and replacements). In this work, we calculated the token-based edit distance by adapting the Wagner–Fischer algorithm (Wagner and Fischer, 1974) to determine changes at a token level. We preprocessed our sentences by changing them into lowercase prior to this analysis. To make the results comparable across sentences, we divide the number of changes by the length of the original sentence and obtain values between 0% (no changes) to 100% (completely different sentence).

In addition to toked-based edit operation experiments, we analysed the difference of sentence length between complex and simple variants, the quantity of edit operations type (INSERT, DELETE and REPLACE) and an analysis of redundant operations such as deletions and insertions in the same sentence over the same text piece (we define this as the MOVE operation). Based on our objective to show how different split configurations affect TS model performance, we have presented the percentage of edit operations as the more informative analysis performed on the most representative datasets.

3.2 Edit Distance Distribution

Except for the recent work of Alva-Manchego et al. (2020b), there has been little work on new TS datasets. Most prior datasets are derived by aligning English and Simple English Wikipedia, for example **WikiSmall** and **WikiLarge** (Zhang and Lapata, 2017).

In Figure 1 we can see that the edit distance distribution of the splits in the selected datasets is not even. By comparing the test and development subsets in WikiSmall (Figure 1a) we can see differences in the number of modifications involved in simplification. Moreover, the WikiLarge dataset

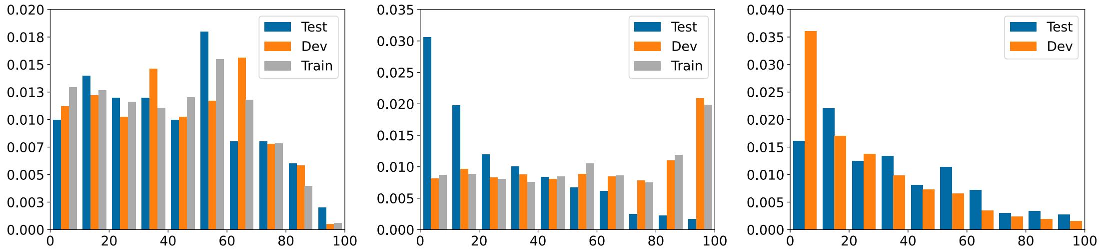

(a) WikiSmall Test/Dev/Train (b) WikiLarge Test/Dev/Train (c) TurkCorpus Test

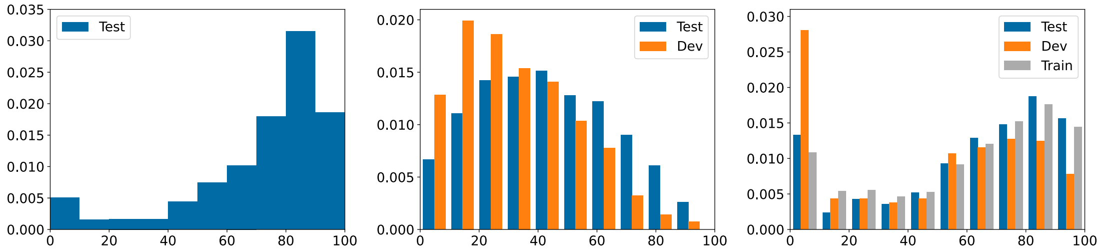

(d) MSD Test (e) ASSET Test (f) WikiManual Test/Dev/Train

Figure 1: Comparison of TS datasets with respect to the number of edit operations between the original and simplified sentences. X-axis: token edit distance normalised by sentence length, Y-axis: probability density for the change percentage between complex and simple sentence pairs.

(Figure 1b) shows a complete divergence of the test subset. Additionally, it is possible to notice a significant number of unaligned or noisy cases, between the 80% and 100% of change in the WikiLarge training and validation subsets (Figure 1b).

We manually checked a sample of these cases and confirmed they were poor-quality simplifications, including incorrect alignments. The simplification outputs (complex/simple pairs) were sorted by their edit distances and then manually checked to determine an approximate heuristic for noisy sentences detection. Since many of these alignments had really poor quality, it was easy to determine the number that removed a significant number of cases without actually reducing dramatically the size of the dataset.

Datasets such as **Turk Corpus** (Xu et al., 2015) are widely used for evaluation and their operations mostly consist of lexical simplification (Alva-Manchego et al., 2020a). We can see this behaviour in Figure 1c, where most edits involve a small percentage of the tokens. This can be noticed when a large proportion of the sample cases are between 0% (no change) to 40%.

In the search of better evaluation resources, TurkCorpus was improved with the development of **ASSET** (Alva-Manchego et al., 2020a) including more heterogeneous modification measures. As we can see in Figure 1e, the data are more evenly distributed than in Figure 1c.

Recently proposed datasets, such as **WikiManual** (Jiang et al., 2020), as shown in Figure 1f, have an approximately consistent distribution, and their simplifications are less conservative. Based on a visual inspection on the uppermost values of the distribution (≈80%), we can tell that often most of the information in the original sentence is removed or the target simplification does not express accurately the original meaning.

MSD dataset (Cao et al., 2020) is a domain-specific dataset, developed for style transfer in the health domain. In the style transfer setting, the simplifications are aggressive (i.e., not limited to individual words), to promote the detection of a difference between one style (expert language) and another (lay language). Figure 1d shows how their change-percentage distribution differs dramatically in comparison to the other datasets, placing most of the results at the right-side of the distribution.

Among TS datasets, it is important to mention that the raw text of the **Newsela** (Xu et al., 2015) dataset was produced by professional writers and is likely of higher quality than other TS datasets. Unfortunately, it is not aligned at the sentence level by default and its usage and distribution are limited by a restrictive data agreement. We have not included this dataset in our analysis due to the restrictive licence under which it is distributed.

Dataset	Split	KL-div	p-value
WikiSmall	Test/Dev	0.0696	0.51292
	Test/Tr	0.0580	0.83186
WikiLarge	Test/Dev	0.4623	<0.00001
	Test/Tr	0.4639	<0.00001
WikiManual	Test/Dev	0.1020	0.00003
	Test/Tr	0.0176	0.04184
TurkCorpus	Test/Dev	0.0071	0.00026
ASSET	Test/Dev	0.0491	<0.00001

Table 1: KL-divergence between testing (Test) and development (Dev) or training (Tr) subsets.

3.3 KL Divergence

In addition to edit distance measurements presented in Figure 1, we further analysed **KL divergence** (Kullback and Leibler, 1951) of those distributions to understand how much dataset subsets diverge. Specifically, we compared the distribution of the test set to the development and training sets for WikiSmall, WikiLarge, WikiManual, TurkCorpus and ASSET Corpus (when available). We did not include MSD dataset since it only has a testing set.

We performed **randomised permutation tests** (Morgan, 2006) to confirm the statistical significance of our results. Each dataset was joined together and split randomly for 100,000 iterations. We then computed the p-value as a percentage of random splits that result in the KL value equal to or higher than the one observed in the data. Based on the p-value, we can decide whether the null hypothesis (i.e. that the original splits are truly random) can be accepted. We reject the hypothesis for p-value lower than 0.05. In Table 1 we show the computed KL-divergence and p-values. The p-values below 0.05 for WikiManual and WikiLarge confirm that these datasets do not follow a truly random distribution.

4 Simplification Datasets: Experiments

We carried out the following experiments to evaluate the variability in performance of TS models caused by the issues described in Wiki-based data.

4.1 Data and Methods

For the proposed experiments, we used the EditNTS model, a Programmer-Interpreter Model (Dong et al., 2020). Although the original code was published, its implementation required minor modifications to run in our setting. The modifications performed, the experimental subsets as well as the source code are documented via GitHub[1]. We selected EditNTS model due to its competitive performance in both WikiSmall and WikiLarge datasets[2]. Hence, we consider this model as a suitable candidate for evaluating the different limitations of TS datasets. In future work, we will definitely consider testing our assumptions under additional metrics and models.

In relation to TS datasets, we trained our models on the training and development subsets from WikiLarge and WikiSmall, widely used in most of TS research. In addition, these datasets have a train, development and test set, which is essential for retraining and testing the model with new split configurations. The model was first trained with the original splits, and then with the following variations:

Randomised split: as explained in Section 3.3, the original WikiLarge split does not have an even distribution of edit-distance pairs between subsets. For this experiment, we resampled two of our datasets (WikiSmall and WikiLarge). For each dataset, we joined all subsets together and performed a new random split.

Refined and randomised split: we created subsets that minimise the impact of poor alignments. These alignments were selected by edit distance and then subsets were randomised as above. We presume that the high-distance cases correspond to noisy and misaligned sentences. For both WikiSmall and WikiLarge, we reran our experiments removing 5% and 2% of the worst alignments.

Finally, we evaluated the models by using the test subsets of external datasets, including: TurkCorpus, ASSET and WikiManual.

5 Discussion

Figure 2 shows the results for WikiSmall. We can see a minor decrease in SARI score with the random splits, which means that the noisy alignments were equivalently present in all the sets rather than using the best cases for training. On the other hand, when the noisy cases are removed from the datasets the increase in model performance is clear.

Likewise, we show WikiLarge results in Figure 3. When the data is randomly distributed, we obtain better performance than the original splits. This

---

[1] `https://github.com/lmvasque/ts-explore`

[2] `https://github.com/sebastianruder/NLP-progress/blob/master/english/simplification.md`

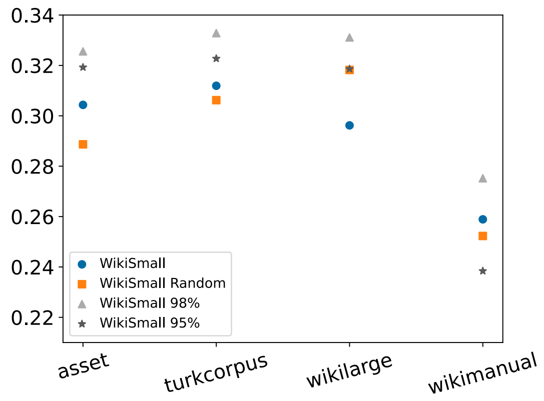

Figure 2: SARI scores for evaluating WikiSmall-based models on external test sets.

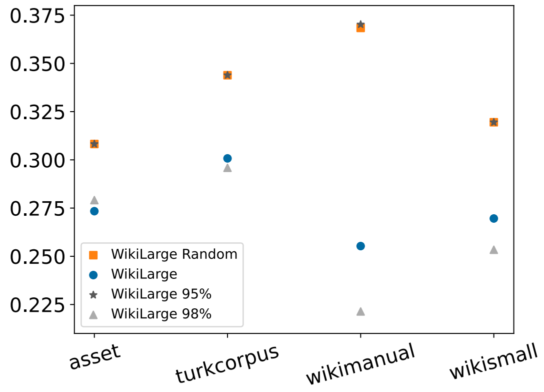

Figure 3: SARI scores for evaluating WikiLarge-based models on external test sets.

is consistent with WikiLarge having the largest discrepancy according to our KL-divergence measurements, as shown in Section 3.3. We also found that the 95% split gave a similar behaviour to WikiLarge Random. Meanwhile, the 98% dataset, gave a similar performance to the original splits for ASSET and TurkCorpus[3].

We can also note, that although there is a performance difference between WikiSmall Random and WikiSmall 95%, in WikiLarge the same splits have quite similar results. We believe these discrepancies are related to the size and distribution of the training sets. WikiLarge subset is three times bigger than WikiSmall in the number of simple/complex pairs. Also, WikiLarge has a higher KL-divergence (≈0.46) than WikiSmall (≈0.06), which means that WikiLarge could benefit more from a random distribution experiment than WikiSmall, resulting in higher performance on WikiLarge. Further differences may be caused by the procedures used to make the training/test splits in the original research, which were not described in the accompanying publications.

Using randomised permutation testing, we have confirmed that the SARI differences between the models based on the original split and our best alternative (95% refined) is statistically significant ($p < 0.05$) for each configuration discussed above.

In this study, we have shown the limitations of TS datasets and the variations in performance in different splits configurations. In contrast, existing evidence cannot determine which is the most suitable split, especially since this could depend on each specific scenario or target audience (e.g., model data similar to “real world” applications).

Also, we have measured our results using SARI, not only because it is the standard evaluation metric in TS but also because there is no better automatic alternatives to measure simplicity. We use SARI as a way to expose and quantify SOTA TS datasets limitations. The increase in SARI scores should be interpreted as the variability in the relative quality of the output simplifications. By relative we mean, that there is a change in simplicity gain but we cannot state the simplification is at its best quality since the metric itself has its own weaknesses.

6 Conclusions

In this paper, we have shown 1) the statistical limitations of TS datasets, and 2) the relevance of subset distribution for building more robust models. To our knowledge, distribution-based TS datasets analysis has not been considered before. We hope that the exposure of these limitations kicks off a discussion in the TS community on whether we are in the correct direction regarding evaluation resources in TS and more widely in NLG. The creation of new resources is expensive and complex, however, we have shown that current resources can be refined, motivating future studies in the field of TS.

Acknowledgments

We would like to thank Nhung T.H. Nguyen and Jake Vasilakes for their valuable discussions and comments. Laura Vásquez-Rodríguez’s work was funded by the *Kilburn Scholarship* from the *University of Manchester*. Piotr Przybyła’s work was supported by the *Polish National Agency for Academic Exchange* through a *Polish Returns* grant number PPN/PPO/2018/1/00006.

---

[3] ASSET and Turk Corpus results are an average on their multiple references scores.

References

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. Readability assessment for text simplification. *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9.

Fernando Alva-Manchego, Joachim Bingel, Gustavo H Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning How to Simplify From Explicit Labeling of Complex-Simplified Text Pairs. *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020a. ASSET: A Dataset for Tuning and Evaluation of Sentence Simplification Models with Multiple Rewriting Transformations. *arXiv*.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020b. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679, Online. Association for Computational Linguistics.

Ozan Caglayan, Pranava Madhyastha, and Lucia Specia. 2020. Curious case of language generation evaluation metrics: A cautionary tale. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2322–2328, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yixin Cao, Ruihao Shui, Liangming Pan, Min-Yen Kan, Zhiyuan Liu, and Tat-Seng Chua. 2020. Expertise Style Transfer: A New Task Towards Better Communication between Experts and Laymen. In *arXiv*, pages 1061–1071. Association for Computational Linguistics (ACL).

Michael Cooper and Matthew Shardlow. 2020. CombiNMT: An exploration into neural text simplification models. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5588–5594, Marseille, France. European Language Resources Association.

Jan De Belder and Marie-Francine Moens. 2010. Text Simplification for Children. *Proceedings of the SIGIR Workshop on Accessible Search Systems*, pages 19–26.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2020. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 3393–3402. Association for Computational Linguistics (ACL).

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic Multi-Level Multi-Task Learning for Sentence Simplification. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 462–476.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5203–5213, Florence, Italy. Association for Computational Linguistics.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard Wikipedia to simple Wikipedia. In *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pages 211–217. Association for Computational Linguistics (ACL).

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text Simplification for Reading Assistance: A Project Note. In *Proceedings of the Second International Workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, pages 9–16, USA. Association for Computational Linguistics (ACL).

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural CRF Model for Sentence Alignment in Text Simplification. In *arXiv*, pages 7943–7960. arXiv.

S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

William Morgan. 2006. Statistical Hypothesis Tests for NLP or: Approximate Randomization for Fun and Profit.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 2, pages 85–91. Association for Computational Linguistics (ACL).

Richard Yuanzhe Pang. 2019. The Daunting Task of Real-World Textual Style Transfer Auto-Evaluation. *arXiv*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. *ACL*, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics(July):311–318.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. Simplify or help? Text simplification strategies for people with dyslexia. In *W4A 2013 - International Cross-Disciplinary Conference on Web Accessibility*.

Carolina Scarton and Lucia Specia. 2018. Learning simplifications for specific target audiences. In *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 2, pages 712–718, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Shardlow. 2014. A Survey of Automated Text Simplification. *International Journal of Advanced Computer Science and Applications*, 4(1).

Sara Botelho Silveira and António Branco. 2012. Enhancing multi-document summaries with sentence simplification. In *Proceedings of the 2012 International Conference on Artificial Intelligence, ICAI 2012*, volume 2, pages 742–748.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. BLEU is Not Suitable for the Evaluation of Text Simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. 2019. Unsupervised Neural Text Simplification. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 2058–2068.

Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence simplification with memory-augmented neural networks. In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 2, pages 79–85. Association for Computational Linguistics (ACL).

Robert A. Wagner and Michael J. Fischer. 1974. The String-to-String Correction Problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 584–594. Association for Computational Linguistics (ACL).

Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. 2018. Integrating transformer and paraphrase rules for sentence simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 3164–3173. Association for Computational Linguistics.

COM2SENSE: A Commonsense Reasoning Benchmark with Complementary Sentences

Shikhar Singh[*1], **Nuan Wen**[*1], **Yu Hou**[1], **Pegah Alipoormolabashi**[2],
Te-Lin Wu[3], **Xuezhe Ma**[1], **Nanyun Peng**[3]
[1]University of Southern California, [2] Sharif University of Technology,
[3]University of California, Los Angeles
`{ssingh43,nuanwen,houyu,xuezhe.ma}@usc.edu palipoor976@gmail.com`
`{telinwu,violetpeng}@cs.ucla.edu`

Abstract

Commonsense reasoning is intuitive for humans but has been a long-term challenge for artificial intelligence (AI). Recent advancements in pretrained language models have shown promising results on several commonsense benchmark datasets. However, the reliability and comprehensiveness of these benchmarks towards assessing model's commonsense reasoning ability remains unclear. To this end, we introduce a new commonsense reasoning benchmark dataset comprising natural language true/false statements, with each sample paired with its complementary counterpart, resulting in 4k sentence pairs. We propose a pairwise accuracy metric to reliably measure an agent's ability to perform commonsense reasoning over a given situation. The dataset is crowdsourced and enhanced with an adversarial model-in-the-loop setup to incentivize challenging samples. To facilitate a systematic analysis of commonsense capabilities, we design our dataset along the dimensions of knowledge domains, reasoning scenarios and numeracy. Experimental results demonstrate that our strongest baseline (UnifiedQA-3B), after fine-tuning, achieves ~71% standard accuracy and ~51% pairwise accuracy, well below human performance (~95% for both metrics). The dataset is available at `https://github.com/PlusLabNLP/Com2Sense`.

1 Introduction

The capability of acquiring and reasoning over commonsense knowledge plays a crucial role for artificial intelligence (AI) systems that interact with humans and accomplish tasks in the real world. For example, given a situation where *someone is asleep*, an agent should choose to *broom* instead of *vacuum* to clean the room, as the latter would be noisy. Likewise, a personal assistant should be able to infer that one is probably unavailable if they are *at work*. This ability to contextualize and draw upon implicit knowledge, and generalize to novel situations, requires commonsense reasoning.

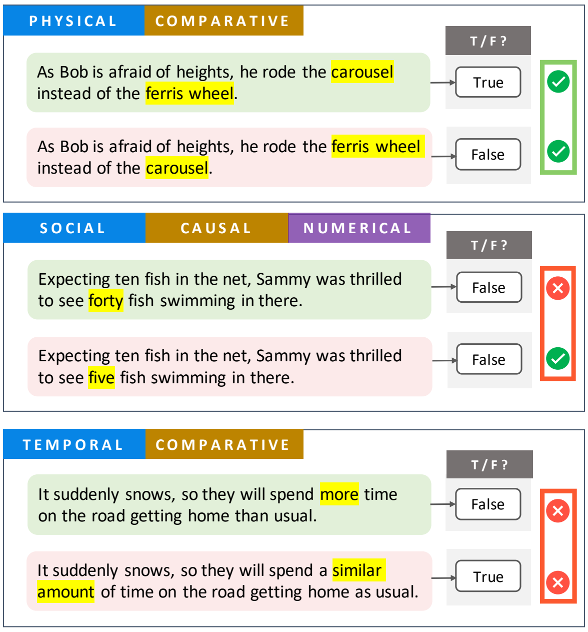

As Bob is afraid of heights, he rode the carousel instead of the ferris wheel.

As Bob is afraid of heights, he rode the ferris wheel instead of the carousel.

Expecting ten fish in the net, Sammy was thrilled to see forty fish swimming in there.

Expecting ten fish in the net, Sammy was thrilled to see five fish swimming in there.

It suddenly snows, so they will spend more time on the road getting home than usual.

It suddenly snows, so they will spend a similar amount of time on the road getting home as usual.

Figure 1: Complementary sentence pair samples from COM2SENSE defined along knowledge domains (*e.g. physical*), reasoning scenarios (*e.g. comparative*) and numeracy attributes. Each sentence within a pair is either true (green boxes) or false (red boxes), followed by model predictions and annotations of whether the predictions are correct. A *standard* accuracy is computed by the percentage of correctly judged sentences (50% for these three pairs), while the *pairwise* accuracy requires both *individual* judgements to be correct in each pair (33% for these three pairs).

While humans are able to intuitively acquire commonsense knowledge from everyday experience and make sound inferences, whether current AI systems also possess such capabilities remains an open question. Recent advancements in natural language processing (NLP) has led to a surge in

---

[*] indicates equal contributions

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 883–898
August 1–6, 2021. ©2021 Association for Computational Linguistics

new benchmark datasets towards evaluating commonsense reasoning. Specifically, existing benchmarks are formulated as natural language inference (NLI) (Bhagavatula et al., 2020), *multiple choice* (MC) question answering (Talmor et al., 2019a; Zellers et al., 2019; Bisk et al., 2020), and machine reading comprehension (Huang et al., 2019) tasks.

While recent state-of-the-art models (Liu et al., 2019; Raffel et al., 2020; Khashabi et al., 2020) have quantitatively demonstrated near human-level performance on these benchmarks, the exploitation of certain spurious patterns (Gururangan et al., 2018; Poliak et al., 2018; McCoy et al., 2019) in the datasets can be partly attributed to such achievements. Consider the examples in Figure 1, where each sentence is true/false, and is paired with a similar (with a few modifications) complementary counterpart such that the answer is flipped. Humans can infer each statement independently with confidence, but models on the other hand struggle to give consistent judgements for the complementary pairs. This indicates that models are able to *guess* the correct answer without a thorough understanding of the given input. If we formulate this as a multiple choice task, where *only the true* sentence needs to be *singled out* given the pairs, the models have higher chances to get it correct, as they are only required to select the *relatively better* option.

Furthermore, most existing commonsense benchmarks focus on the *factual* aspects of commonsense (Talmor et al., 2019a; Bisk et al., 2020), and generally do not explicitly concern with *reasoning* (Singer et al., 1992), *i.e.* the mental manipulation of factual knowledge, which we hypothesize is crucial for generalizing to novel situations. While some prior works investigate commonsense reasoning in the context of social intelligence and co-reference resolutions (Sap et al., 2019; Sakaguchi et al., 2020), the reasoning components are implicit. Existing benchmarks fail to provide a systematic and comprehensive means of analyzing different aspects of commonsense knowledge and reasoning.

To address these challenges, we introduce the **Com**plementary **Com**monsense (COM2SENSE) benchmark dataset which contains 4k complementary true/false sentence pairs. Each pair is constructed with minor perturbations to a sentence to derive its complement such that the corresponding label is inverted (see Figure 1). This *pairwise* formulation provides a more reliable evaluation metric, where a model is considered correct *only if it succeeds on both statements*. We employ an adversarial crowdsourcing framework to collect human created samples via a *gamified* machine-in-the-loop process: A strong pretrained model is setup to provide instant feedbacks, thereby incentivizing challenging samples that can *fool* the model.

Broadly inspired by the *Theory of Core Knowledge*, *i.e.* the ability to reason about objects, places, numbers and the social world (Spelke and Kinzler, 2007), we design our dataset along the following dimensions: **knowledge domains** (*physical*, *social*, *temporal*), and **reasoning scenarios** (*causal*, *comparative*). Additionally, concurrent to a recent work (Lin et al., 2020a) on studying numerical commonsense, we include a third dimension of **numeracy**, which extends the factual focus of Lin et al. (2020a) (*e.g.* *"Ants have* ***six*** *legs."*) to *numerical reasoning* (*e.g.* the *ten fish* versus *forty fish* in Figure 1). To the best of our knowledge, we are the first to explicitly introduce these dimensions in a commonsense benchmark dataset, thereby facilitating a more detailed and systematic probing of models' commonsense understanding.

Our experiments demonstrate that the best performing pretrained language models achieve ~71% standard and ~51% pairwise accuracy, well below human performance. Additionally, we provide ablation studies on effect of training size on model performance, and the transferrability across the reasoning scenarios. We summarize our contributions as follows: 1) We introduce a commonsense reasoning dataset which we position as a challenging *evaluation benchmark* (instead of a training resource) for NLP models. 2) We propose a pairwise evaluation metric featured by our complementary pair formulation for a more reliable assessment of commonsense reasoning abilities. 3) We benchmark state-of-the-art models that highlight significant gaps (>45%) between model and human performances.

2 Dataset

We introduce COM2SENSE, a dataset for benchmarking commonsense reasoning ability of NLP models. We use crowdsourcing to collect the dataset and supplemented with an adversarial *model-in-the-loop* approach. The key features of our development process are: 1) qualification quiz to filter and familiarize workers, 2) *gamified* creation tasks, and 3) quality check by experts. The details of dataset formulation and collection pro-

Domain	Scenario	Numeracy	Example	Complement
Physical	Comparative	No	*If we dropped* ***milk*** *on the floor, it is better to clean with a mop rather than a broom.*	*cereal*
Physical	Causal	No	*To* ***read books*** *at night, one should turn on the lights.*	*see stars*
Social	Comparative	No	*Sam robbed a store, while Tim jumped the lights. People will likely be more* ***forgiving*** *towards Tim.*	*chastising*
Social	Causal	Yes	*Given his $1500 monthly income and no savings, he can afford an apartment rent of* ***$500****.*	*$3000*
Temporal	Comparative	Yes	*Tim needs to return home in 2 hours, so he would prefer to* ***hit the gym*** *rather than* ***go hiking****.*	`swap`
Temporal	Causal	Yes	*If Leo earns $100 per day, then by working from Monday to* ***Friday*** *his weekly income will be $500.*	*Wednesday*

Table 1: Data samples from different categories in Com2Sense. Each example is labelled as true, while its complement (false) is generated by substituting or swapping the words in **bold** (in green or red font).

cedure, along with statistics are provided in the following sections.

2.1 Formulation

Com2Sense seeks to measure a comprehensive commonsense understanding of everyday events and entities. The task requires one to judge whether a given sentence is true or false. For each sentence in the dataset, we also compose its complementary counterpart by modifying a few words, such that the answer is inverted. The key advantages of using complementary pairs are two-folds: 1) it provides a more robust way of evaluating models' commonsense reasoning ability by requiring both sentences to be correctly judged, and 2) the complements naturally highlight the salient words which may be useful in probing model behaviors.

Furthermore, to facilitate a systematic study of commonsense, we design our dataset across the following three dimensions:

1. **Knowledge Domain:** We categorize commonsense knowledge into *physical*, *social* and *temporal* domains. The physical domain emphasizes on an intuitive understanding of physical properties (*e.g*. weight, shape, motion, space) and object affordances. The social domain encapsulates interactions (*e.g*. intent, emotion, reaction), activities, and societal norms. The temporal domain captures the notion of time, particularly attributes such as duration, frequency and order of events. While domains may not always be strictly exclusive (*e.g*. choice of transport and duration), our complementary pair setup naturally places emphasis on the intended domain.

2. **Reasoning Scenario:** We define two types of inferential reasoning scenarios: 1) The *causal* scenario requires the ability to infer whether a cause explanation or a subsequent event (cause-effect) is correct. 2) The *comparative* scenario requires the ability of determining the most plausible hypothesis between two or more competing ones.

3. **Numeracy:** Refers to the basic understanding of numbers, arithmetic, ratios, statistics, etc. With the objective of linking numeracy to commonsense, we particularly focus on "number sense" – an intuitive understanding of numbers, their magnitude and relationships, rather than computational and numerical precision.

Therefore, each sample in our dataset should fall into a category defined by a combination of the above dimensions, as exemplified in Table 1.

2.2 Dataset Creation

Com2Sense is developed through crowdsourcing on Amazon Mechanical Turk (MTurk) with the goal of collecting complementary sentence pairs. The creation tasks are constructed for each *category* defined by the combination of domain, scenario and numeracy attributes. An overview of the data collection workflow is illustrated in Figure 2. In order to participate, the workers are required to pass a **qualification quiz** designed to familiarize them with the key aspects of our dataset.

Creation: During the creation phase, to orient and aid workers' creativity, they are provided with five examples of complementary pairs that belong to a particular category as reference. We also share a list of verbs and topics pertinent to the current

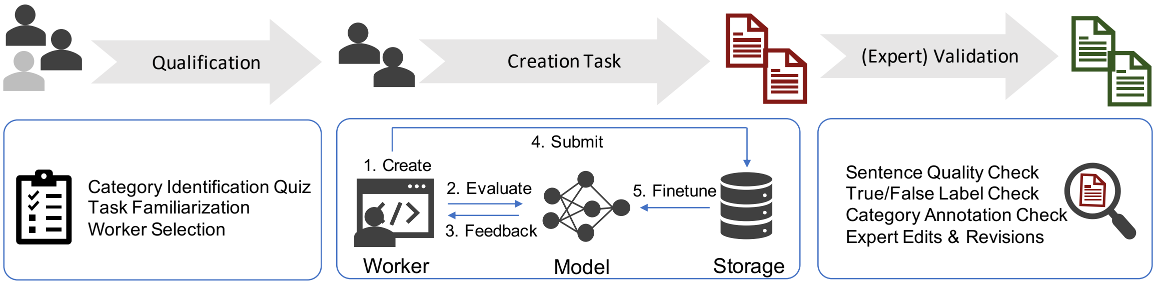

Figure 2: Data collection workflow: 1) qualification quiz to instruct the key aspects of our creation task and filter unqualified workers, 2) interactive *model-in-the-loop* creation process to incentivize challenging samples via model feedback, and 3) data validation according to our guidelines and category descriptions.

domain, as an optional resource. While our examples serve as a reference towards creating complementary pairs, the workers have the freedom to construct their sentences as they deem appropriate.

We employ an adversarial *model-in-the-loop* approach to provide workers with immediate feedback (*i.e.* model predictions) on each created sentence. After entering the inputs and labels, they may choose to evaluate and revise their inputs. If the sentence successfully fools our model to answer incorrectly, workers are awarded with an additional amount for each input[1].

To further incentivize worker creativity, we offer bonuses if the inputs are qualitatively regarded as creative during the validation stage. Such *gamified* process may continue for a few rounds until the workers are satisfied with their monetary rewards.

Model: We deploy a RoBERTa-large based model for binary sentence classification, finetuned on SemEval-2020 Task 4 (Wang et al., 2020) given its true/false format and broad coverage of commonsense knowledge. After the first phase of collection (2k pairs), the model weights are updated by finetuning on our dataset with 60% train, 20% dev and 20% test splits. This will naturally help diversify our dataset samples, as the model is unlikely to be fooled with repetitive knowledge and sentence structures.

Validation: To ensure high quality, the samples are validated by internal members to look for inconsistencies with regard to the category-type *i.e.* follows the domain, scenario and numeracy requirements, and inferential ambiguities that may arise due to insufficient context, specialized concepts, grammatical errors, etc. Furthermore, annotators may choose to revise the samples to fix any of the aforementioned issues. Each sample is validated by three annotators and the final outcome is decided through a majority vote. The inter-annotator agreement score is 0.989 measured using Fleiss' Kappa. Additionally, pairs in which neither input could fool the model are discarded during this stage.

The dataset is developed with the help of 173 workers. To ensure that workers are proficient in English, the demographic pool of the workers is initially limited to the United States. However, we removed this criteria to avoid cultural biases in the dataset. Additionally, to understand the utility of our adversarial model feedback setting, we analyze the data on number of revisions made by workers in order to successfully fool the model. We find that the average number of revisions is 1.36, while the median is zero. This suggests that for majority of samples, workers find our reference material sufficient and are also able to leverage model feedback to aid their creations. Additional details on dataset development are in Appendix Section A.

2.3 Dataset Statistics

Given that COM2SENSE is primarily a benchmark dataset, it is partitioned into train[2] (20%), development (10%), and test (70%) set, respectively. There are in total 4k of statement pairs in our dataset. Complementary statements from the same pair are distributed to the same partition. Table 2 gives the essential statistics of our dataset across different splits. Note that due to the complementary pair formulation, the type-token ratio is approximately reduced by a factor of two, and the dataset is naturally balanced along the true and false labels.

Table 3 gives the breakdown of percentage of samples from each category defined by a combination of the three dimensions. The distribution of most frequent nouns in the dataset is visualized

---

[1] Base pay = \$0.05 – \$0.1 and bonus pay = \$0.5 – \$0.9.

[2] As a resource to adapt models for our task.

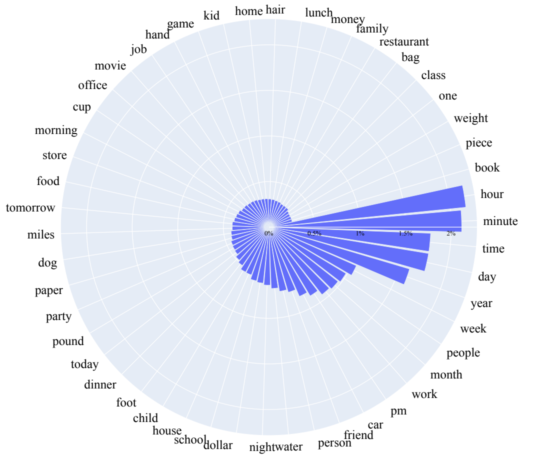

Figure 3: Top-50 frequent nouns in the dataset.

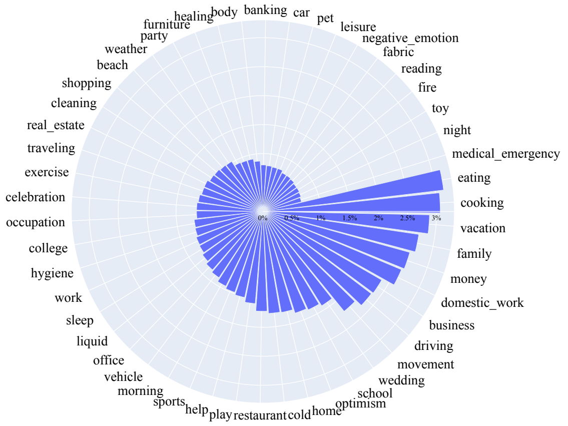

Figure 4: Top-50 frequent topics in the dataset.

in Figure 3. Likewise, the distribution of most frequent topics – lexical categories generated using the *Empath*[3]tool, is provided in Figure 4.

3 Experimental Setup

The experiments are designed to meet the following objectives: 1) benchmark state-of-the-art NLP models along the standard and pairwise formulation; 2) analyze the model performance across different categories of commonsense reasoning; 3) report the effect of training size on model performance; and 4) verify the role of reasoning types by measuring "cross-scenario" transferability.

Besides standard accuracy, we introduce a new metric called **pairwise accuracy** that evaluates as correct if both predictions within a pair are accurate.

Statistic	Train	Dev	Test
# complementary pairs	804	402	2779
Avg input length	21	21	21
Max input length	68	49	67
Min input length	6	7	6
# unique tokens	2306	1541	4407
# total tokens	21116	10520	72517

Table 2: Dataset statistics across different splits[4]

	Scenario	
Domain	Causal	Comparative
Physical	17.47% (24%)	18.92% (23%)
Social	14.68% (50%)	16.51% (22%)
Temporal	16.74% (57%)	15.68% (62%)

Table 3: Category-wise breakdown (percentage) of dataset samples. The quantities in parenthesis refer to the relative proportion of samples with numeracy, under the given combination of domain and scenario.

We benchmark several state-of-the-art NLP models, specifically the ones proven preeminent in existing commonsense benchmarks, and additionally include a Bi-LSTM model as a baseline to help check for potential spurious correlations in the dataset. We consider the following baselines:

BiLSTM+GloVe A bidirectional-LSTM model (Hochreiter and Schmidhuber, 1997) taking input word embeddings from GloVe (Pennington et al., 2014).

BERT The BERT-base (110M) model introduced in (Devlin et al., 2019).

RoBERTa-large A large variant (355M) of RoBERTa model (Liu et al., 2019) built upon BERT-large architecture.

DeBERTa-large Recently He et al. (2020) proposed a novel disentangled attention mechanism that improves upon BERT and RoBERTa models. We consider the large variant (390M) as a baseline.

T5-large Similarly, the large variant (770M) of the T5 model (Raffel et al., 2020). We follow the standard prefix-based text-to-text format, and adapt it for our binary classification setup.

UnifiedQA The UnifiedQA (Khashabi et al., 2020) was originally trained on numerous datasets including several commonsense reasoning benchmarks, and performed well under zero-shot setting. We consider the variants with T5-large and T5-3B as the architecture backbone.

---

[3] https://github.com/Ejhfast/empath-client

[4] Input lengths are computed with Spacy tokenizer

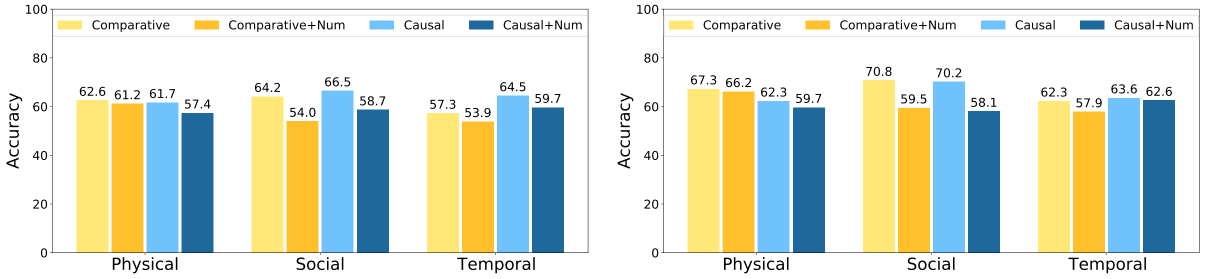

(a) T5-large.

(b) DeBERTa-large.

Figure 5: Model performance breakdowns across domains and scenarios, `"+Num"` denotes numeracy involved.

Model	Standard	Pairwise
Random	50.00	25.00
BiLSTM+GloVe	53.80	29.50
BERT-base	51.79	12.91
RoBERTa-large	59.35	33.28
T5-large	60.56	41.84
UnifiedQA-large	60.83	34.79
DeBERTa-large	63.53	45.30
UnifiedQA-3B	**71.31**	**51.26**
Human	96.50	95.00

Table 4: Test set accuracy for selected models, trained and evaluated on our dataset. Human performances are obtained with 200 randomly selected and decoupled samples from 100 pairs.

	Model			
Dataset	Random	RoBERTa	T5	Human
CQA	20.00	72.10	73.35	88.90
SWAG	25.00	89.92	88.72	88.00
SocialIQA	33.33	77.12	73.25	84.40
PIQA	50.00	77.21	79.89	94.90
WinoGrande	50.00	79.14	75.02	94.00
Com2Sense$_{\text{standard}}$	50.00	59.35	60.56	96.50
Com2Sense$_{\text{pairwise}}$	25.00	33.28	41.84	95.00

Table 5: Test set accuracy for selected models (RoBERTa-large and T5-large), trained and evaluated on respective datasets.

4 Results and Analysis

Human Performance: To estimate a human upper bound for our dataset, we perform a separate run with ten top performing workers that had participated in our collection phase to examine a randomly selected subset of 200 samples (*i.e.* from 100 pairs). Each worker is assigned with a set of shuffled samples with his/her own creations deliberately filtered out. The answer is determined by majority vote from *three* workers. Human performances are 96.5% with standard accuracy and 95.0% with pairwise accuracy, respectively.

4.1 Results

Benchmark results: Table 4 summarizes the baseline performances on the test set. As the Bi-LSTM model performs close to random, we claim that improvements from stronger baselines should be attributed to the models and not annotation biases that they can exploit. Among the baseline models, the UnifiedQA-3B achieves the best performance on both the standard and pairwise metric. Note that the number of learnable parameters in UnifiedQA-3B is much larger than those in the second and third best models, which are DeBERTa-large (390M) and UnifiedQA-large (770M). Our Com2Sense benchmark remains quite challenging, as there are significant gaps between the model and the human performances.

Dataset comparisons: In order to contrast the difficulty of Com2Sense with other related benchmarks, we report the performances of two well performing models on the following: CommonsenseQA (CQA) (Talmor et al., 2019b), SWAG (Zellers et al., 2018), SocialIQA (Sap et al., 2019), PhysicalIQA (PIQA) (Bisk et al., 2020) and WinoGrande (Sakaguchi et al., 2020).

The results in Table 5 indicate that models clearly struggle more to perform well on Com2Sense than other datasets.

Performance across domains and scenarios: In Figure 5 we present the in-depth breakdown results for T5-large and DeBERTa-large across combinations of domain, scenario and numeracy. We observe that models consistently perform worse in categories involving numeracy, highlighting the limitations of current language models. For physical domain, both models perform worse in causal than in comparative scenario. We hypothesize that while the models may possess the required physi-

Setting	Test	Dev
Multiple-Choice	70.63	77.61
Standard (T/F)	63.53	66.29

Table 6: Performance (accuracy) of DeBERTa-large on the MC formulation of our dataset compared to the standard true/false setting. The model performs relatively worse on the latter.

	Model$_{\text{metric}}$			
Train set	DeBERTa$_{\text{std}}$	T5$_{\text{std}}$	DeBERTa$_{\text{pair}}$	T5$_{\text{pair}}$
20%	63.92	60.65	41.51	34.29
40%	67.74	62.60	48.04	38.11
60%	68.46	63.96	48.47	40.66

Table 7: Performance across different training set sizes (20% / 40% / 60% of the entire dataset) for DeBERTa-large and T5-large. `"std"` and `"pair"` stand for standard and pairwise accuracy correspondingly.

cal knowledge, they fail to generalize to a logical reasoning over known facts or grasp the implicit changes of physical properties, which is generally unseen in the pretraining corpora for NLP models. Opposite trends are observed in both social and temporal domains, where similar hypothesis can be made that causal statements are more frequent patterns in the corpora when social activities or senses of time are the subjects. Furthermore since model feedback was part of our dataset construction, we also report the category-wise difficulty in fooling the model (number of trials) during sample creation in Section A.4 for a reference.

True/False versus multiple choice setup: We further conduct an experiment with DeBERTa-large model on the same data splits with the input formulated as an **MC task** in Table 6. Under this setting, the model is provided with the sentence pair and is required to select the true sentence among the two choices, for the response to be correct. The performance is significantly higher compared to both standard (>7%) and pairwise accuracy (>25%) presented in Table 4. This result supports our intuition that it is easier for models to exploit spurious correlations in the surface patterns under the *multiple choice* question answering setup.

4.2 Analysis

The Effect of Training Data Size: To study the effect of training size on model performance, we design an experiment by varying the sample size in the training set, with fixed dev (10%) and test (30%) sets to **ensure consistency** in evaluation. We consider DeBERTa-large and T5-large models for this ablation study, and report our findings in Table 7. The results indicate a plateau in performance with increase in training samples.

Setting	Standard	Pairwise
Train-C$_{\text{exclude}}$	56.52	19.00
Train-C$_{\text{include}}$	63.54	40.49

Table 8: Test set performance of DeBERTa-large on two different setups with respect to the complementary pairs. Both setups have the same training set size, but in Train-C$_{\text{exclude}}$ only one sentence of each pair is present in the training set, while in Train-C$_{\text{include}}$ both samples in a pair are included. The results indicate the effectiveness of training the models with our formulated complementary pairs.

Role of Complementary Pairs: In previous experiments, we measure the model generalizations by distributing data samples into train and evaluation sets **by complementary pair**. This ensures the similarly constructed sentences within the same pair is not *leaked* into different data splits, and thus an "inter-pair" generalization is measured. To investigate the effectiveness of our complementary pair formulation on training models to acquire commonsense reasoning ability, we first sample a subset with identical size (20% data, 800 pairs) to that of the original train set, and then construct the following two variants (using the same subset):

- **Train-C$_{\text{exclude}}$:** One of the complementary samples (in each pair) is *excluded*, *i.e.* no two samples belong to the same pair in this train set. It comprises 800 samples from 800 pairs, with balanced true/false labels.
- **Train-C$_{\text{include}}$:** We retain half of the data samples where both sentences from a pair are *included*, which leads to 800 samples from 400 pairs in this train set.

The remaining samples from the dataset (without the excluded ones in the two settings) are then split into dev (10%) and test (70%) sets. We compare the performance of DeBERTa-large in the above two different settings in Table 8. The results show a significant decrease in performance when complementary sentences are not provided. We hypothesize that the worse performances are due to the models' tendency to pick up surface patterns and memorize the labels in the training set without really understanding the scenario. Also, model

Train	Evaluate			
	Causal$_{\text{std}}$	Comp.$_{\text{std}}$	Causal$_{\text{pair}}$	Comp.$_{\text{pair}}$
Causal	**63.64**	59.36	**35.46**	28.25
Comp.	58.47	**64.50**	26.43	**43.86**

Table 9: Performance of DeBERTa-large trained on X and evaluated on Y, where X and Y are partitions created as per a reasoning scenario *(causal, comparative)*.

generalization benefits from having complementary samples within the training set.

Cross-Scenario Generalizability: Given that knowledge domain and numeracy attributes of our dataset are intuitively distinct, we intend to quantitatively investigate if the same holds for reasoning scenarios. Our "cross-scenario" experiments with DeBERTa-large, *i.e.* trained on *causal*, evaluated on *comparative* and vice versa, indicate a poor generalization across both standard and pairwise accuracy metrics (see Table 9), underscoring the significance of having reasoning types.

5 Related Works

Commonsense Resources: As commonsense is a crucial component to the actualization of AI, there has been a surge in creating relevant benchmarks, notable ones include evaluating machines' commonsense abilities in the format of pronoun resolution (Levesque et al., 2012; Sakaguchi et al., 2020), multiple choice (Zellers et al., 2018; Talmor et al., 2019a), natural language generations (Lin et al., 2020b), story understanding (Mostafazadeh et al., 2016), and reading comprehensions (Zhang et al., 2018; Huang et al., 2019; Ning et al., 2020). Our work puts forth to create a commonsense benchmark in the format of true/false complementary pairs, where a more robust pairwise accuracy is adopted. Note that although natural language inference (NLI) can be tasked similarly to the true/false formulation, the existing commonsense NLI benchmark either is not crowdsourced with high quality (Zhang et al., 2017), or still resorts to a multiple choice setting (Bhagavatula et al., 2020). There are also benchmarks that specifically concern a type of commonsense knowledge, such as physical (Bisk et al., 2020) and social (Sap et al., 2019) intelligence, as well as temporal understanding (Zhou et al., 2019). The ability to understand and induce numerical knowledge in texts has been studied in several recent works (Dua et al., 2019; Ravichander et al., 2019), including numerical commonsense (Lin et al., 2020a). Our work differs to these works in the focus on less factual and arithmetic-precise numerical knowledge, but more on the intuitive sense of numbers, in conjunction with our defined knowledge domains and the scenarios.

It is worth noting that some prior works (Wu et al., 2017; Clark et al., 2019) also investigate the effectiveness in the binary true/false (yes/no) formulation to construct a question answering dataset, while COM2SENSE is the first to focus on commonsense reasoning.

Dataset Biases: It is a widely perceived issue that spurious statistical patterns in datasets can often be exploited by machine learning models, which can potentially lead to overoptimistic judgements on the model improvements. Particularly in NLP domain, prior works have shown that hypothesis-only baselines or syntactic heuristics perform surprisingly well in the NLI task (Gururangan et al., 2018; Glockner et al., 2018; Tsuchiya, 2018; Poliak et al., 2018; McCoy et al., 2019). Model exploiting biases or failing on simple adversarial patterns, can also be seen in sentence classification (Wieting and Kiela, 2019) and question answering (Jia and Liang, 2017; Kaushik and Lipton, 2018; Geva et al., 2019) tasks. We put forth to reduce the potential sentence-level biases by requiring the models to perform equally well on both directions in a complementary true/false pair.

Adversarial Data Collection: Removing representation biases in a dataset by adversarially filtering undesired data samples, has been frequently practiced to collect datasets more challenging to the models. Recent work *AFLite* (Sakaguchi et al., 2020; Le Bras et al., 2020), built upon the adversarial filtering (AF) method in (Zellers et al., 2018, 2019), adopted an iteratively improving model-in-the-loop approach to collect challenging commonsense benchmarks (Sakaguchi et al., 2020; Bisk et al., 2020). *Gamified* (Yang et al., 2018) or interactive (Wallace et al., 2019) approaches leverage human-in-the-loop to increase the difficulty of datasets and hence more robust model training. Counterfactual editing of data samples with human annotators (Kaushik et al., 2020; Gardner et al., 2020) is also closely related to our complementary pair construction that seeks to *invert* the model predictions for a more reliable evaluation.

Recently, several works have attempted to exploit the merits in involving both models and humans in the data creation cycle, *i.e.* human-and-model-in-the-loop, to construct data samples that are both *new* and challenging to the models (Chen et al., 2019; Nie et al., 2020; Bartolo et al., 2020). To our best knowledge, we are the first to employ such an approach in constructing commonsense reasoning benchmark, specifically, our complementary pair formulation makes it more sophisticated as the annotators are required to not only *fool* the model but also pay attention to the salient concepts of their creations in both directions.

6 Conclusion

We present a new challenging commonsense reasoning benchmark, COM2SENSE, developed via an adversarial *gamified* model-in-the-loop approach. COM2SENSE comprises 4k *manually created* complementary true/false statement pairs, designed along three dimensions: knowledge domain, reasoning scenario, and numeracy. We propose a robust pairwise metric to evaluate models' commonsense reasoning ability based on the complementary pair formulation, and benchmark the dataset with several state-of-the-art NLP models, highlighting significant gaps well below human performances ($>$ 45% gap).

On top of providing a new commonsense reasoning benchmark, we demonstrate studies on transferrability among defined commonsense aspects, with an objective to spur future research on a more systematic probing of models' grasp of commonsense. As a potential future work drawn from these insights, we hope to inspire future model developments, specifically in two directions: 1) the ability to reason over known facts (*i.e*. reasoning scenario), and 2) acquiring the implicit knowledge that is commonsensible to humans (*i.e*. knowledge domain). Furthermore, we hope our investigation in the formulations of question answering task (*i.e*. MC setting versus our true/false complementary setting) can shed light on future explorations in identifying potential artifacts in NLP datasets.

7 Acknowledgements

We thank the anonymous reviewers for their feedback, and all the workers who have participated in the dataset creation on Amazon Mechanical Turk. We give special thanks to Peifeng Wang, Xiangci Li, and Gleb Satyukov for their great help in the early stage of annotation pipeline construction as well as providing valuable feedback in composing the guidline instructions. This work is supported by the Machine Common Sense (MCS) program under Cooperative Agreement N66001-19-2-4032 with the US Defense Advanced Research Projects Agency (DARPA). The views and the conclusions of this paper are those of the authors and do not reflect the official policy or position of DARPA.

8 Ethics and Broader Impacts

We hereby acknowledge that all of the co-authors of this work are aware of the provided *ACM Code of Ethics* and honor the code of conduct. This work is mainly about the creation of a challenging commonsense benchmark dataset. The followings give the aspects of both our ethical considerations and our potential impacts to the community.

Dataset We collect an English dataset of commonsense complementary sentence pairs via Amazon Mechanical Turk (MTurk) and ensure that all the personal information of the workers involved (e.g., usernames, emails, urls, demographic information, etc.) is discarded in our dataset. This research has been reviewed by the **IRB board** and granted the status of an **IRB exempt**. The detailed annotation process (pay per amount of work, guidelines) is included in the appendix; and overall, we ensure our pay per task is above the the annotator's local minimum wage (~$12 USD/HR). Although commonsense can vary from different demographic areas, we primarily consider English speaking regions for the first round, and include more annotators from non English-spoken countries to diversify the dataset. Future work can include collecting a more diverse dataset across more demographics regions to incorporate more regional-dependent commonsense, while using some post editing to ensure English proficiency of the constructed data.

Techniques We benchmark the created dataset with the state-of-the-art large-scale pretrained language models, with minimum adaptation to the formulation of this dataset (*i.e*. true/false formulation). As commonsense is of our main focus, we do not anticipate production of harmful outputs, especially towards vulnerable populations, after training NLP models on our dataset.

References

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: Investigating adversarial human annotation for reading comprehension. In *Transactions of the Association for Computational Linguistics (TACL)*, volume 8, pages 662–678.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *International Conference on Learning Representations (ICLR)*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. CODAH: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 63–69, Minneapolis, USA. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models' local decision boundaries via contrast sets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1307–1323.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*, volume 9, pages 1735–1780. MIT Press.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations (ICLR)*.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 5010–5015, Brussels, Belgium. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single qa system. In *Findings of the Association for Computational Linguistics: EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1078–1088. PMLR.

H. Levesque, E. Davis, and L. Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020a. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 6862–6868, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020b. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 1823–1840, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 839–849.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 4885–4901, Online. Association for Computational Linguistics.

Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. 2020. Torque: A reading comprehension dataset of temporal ordering questions. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1158–1172. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*, volume 21, pages 1–67.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose, and Eduard Hovy. 2019. EQUATE: A benchmark evaluation framework for quantitative reasoning in natural language inference. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 349–361, Hong Kong, China. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Murray Singer, Michael Halldorson, Jeffrey C Lear, and Peter Andrusiak. 1992. Validation of causal bridging inferences in discourse understanding. In *Journal of Memory and Language*, volume 31, pages 507–524. Elsevier.

Elizabeth S Spelke and Katherine D Kinzler. 2007. Core knowledge. In *Developmental science*, volume 10, pages 89–96. Wiley Online Library.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019a. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019b. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *International Conference on Language Resources and Evaluation (LREC)*.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. In *Transactions of the Association for Computational Linguistics (TACL)*.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020. SemEval-2020 task 4: Commonsense validation and explanation. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 307–321, Barcelona (online). International Committee for Computational Linguistics.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *International Conference on Learning Representations (ICLR)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP): System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. In *Computer Vision and Image Understanding*, volume 163, pages 21–40. Elsevier.

Zhilin Yang, Saizheng Zhang, Jack Urbanek, Will Feng, Alexander H Miller, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Mastering the dungeon: Grounded language learning by mechanical turker descent. In *International Conference on Learning Representations (ICLR)*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. In *Transactions of the Association for Computational Linguistics (TACL)*, volume 5, pages 379–395.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. “going on a vacation” takes longer than “going for a walk”: A study of temporal commonsense understanding. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.

A Additional Details of COM2SENSE

A.1 Collection with MTurk

Qualification Quiz To familiarize the workers with our collection task, we design a quiz with the following types of questions: 1) examine if a given statement can be correctly judged with only commonsense or it requires specialized knowledge, 2) infer the true/false label of a given statement, and 3) select the most suitable domain and scenario where a given statement belongs to.

Human Intelligence Tasks (HITs) The general instructions of our HIT page include: **Task Overview**, **Task Payment** and **Overall Task Procedure** for each category, to engage more workers. At the end of the HIT instruction page, a link is provided to redirect the workers to the data creation page, where more detailed instructions and useful resources for the creation tasks are provided. Besides passing our qualification quiz, the workers are also required to have a *HIT Approval Rate* greater than 98% and the *Number of HITs Approved* greater than 5000. In each HIT assignment, workers are required to submit three complementary pairs. In the first phase of data collection, the base pay is \$0.6 for each assignment and workers will receive a \$0.5 bonus per sentence if it follows our instructions and fools the model; for the second phase, the base payment for each assignment is \$0.3 but we change the bonus to: \$0.5 (for either high-quality sentence or successful fooling) or \$0.9 (if both requirements are met, similar to those for the \$0.5 bonus in the first phase) to encourage workers to create higher quality data.

A.2 Details of the Creation

Tool Interface Screenshots of our creation interface are as shown in Figure 6 and Figure 7. We name our deployed model (RoBERTa-large) *Carl* to help emphasize the interactive and gamified creation set-up.

Guidelines To inspire workers and collect from more diverse topics of commonsense, we further provide: 1) some hints for having higher chances fooling the model, such as exploiting contradictory physical concepts, negations, swapping entities, etc., 2) topics pertinent to the domains, and (3) examples of low quality along with their reasons.

A.3 Details of Validation

To ensure data quality, our internal members have helped checking each pair with the validation tool we implement. For each pair received from the workers, both labels for the statements and their intended domains and scenarios are carefully verified. For statements which are ambiguous even for humans, if they can be easily fixed by adding more context or better word choices, another round of editing is conducted.

A.4 Adversarial Setting

The total number of collected complementary pairs is around 4.8k, where around .8k are discarded for not having sufficiently high quality, *e.g.* "*Frank traded a stock an hour late and lost 80 million dollars.*" and "*Frank traded a stock a second late and lost 80 million dollars.*" Among all the data we collected, the overall fooling rate is 48.55% per sentence and 78.7% per pair. For category-specific fooling rates, please refer to Figure 10.

For sentences that successfully fool the model, we report the mean time of fooling one sentence to be 3.40, the standard deviation (std) as 2.48, and the median as 2.57 (all in minutes). Please notice that the total time is directly retrieved from MTurk and is likely to be overestimated due to worker inactivity. The mean of the number of revisions per fooling sentence is 1.36 with a std as 3.95, and a median as 0 (fooling without re-attempts, requiring no revision). Noticeably, 63% of the fooling sentences are submitted with no revision.

For any potential interests, Figure 8 shows the mean and median of required time of fooling per sentence across categories, and similarly Figure 9 for the mean and median of the number of attempts which equals to the number of revisions $+1$.

Figure 11 shows the distribution of ratings during the exit survey from a total of 699 valid responses. The survey questions include: 1) how helpful is our instruction? and 2): how challenging is our task? For question 1, the mean rating is 4.66 ± 0.59 and median is 5; for question 2, the mean rating is 4.23 ± 0.92 and the median is 4, where 1-5 is from low-to-high rating.

A.5 Statistics of Workers

173 workers participated in our task, and Figure 12 shows the worker counts for the different numbers of assignments attempted by each of the workers, and Figure 13 shows the worker counts for the time

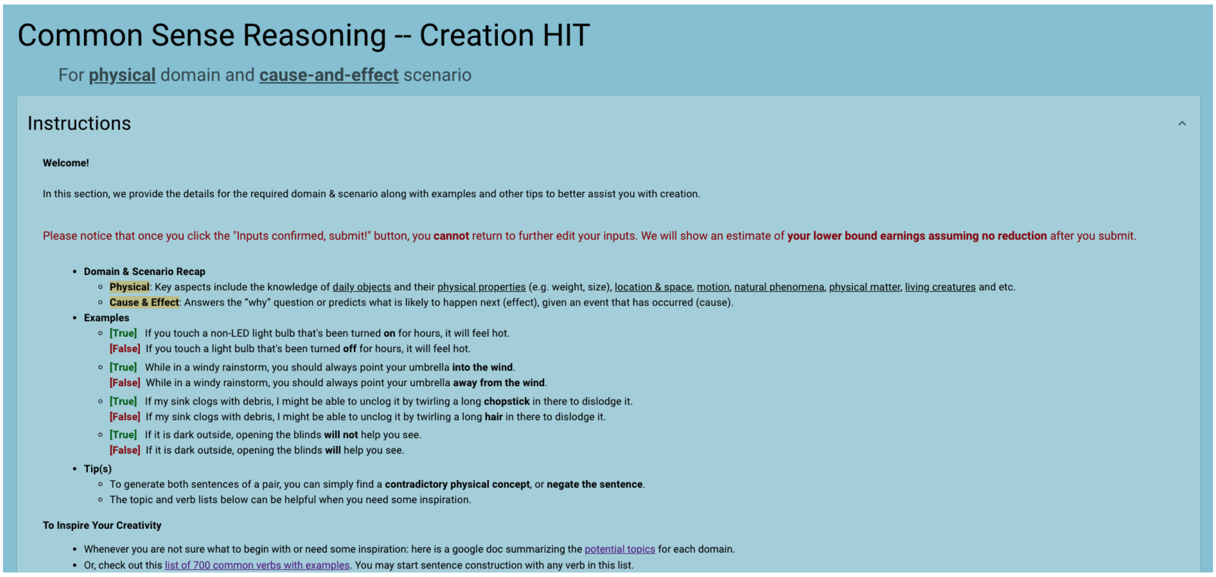

Common Sense Reasoning -- Creation HIT

For **physical** domain and **cause-and-effect** scenario

Instructions

Welcome!

In this section, we provide the details for the required domain & scenario along with examples and other tips to better assist you with creation.

Please notice that once you click the "Inputs confirmed, submit!" button, you **cannot** return to further edit your inputs. We will show an estimate of **your lower bound earnings assuming no reduction** after you submit.

- **Domain & Scenario Recap**
  - **Physical**: Key aspects include the knowledge of daily objects and their physical properties (e.g. weight, size), location & space, motion, natural phenomena, physical matter, living creatures and etc.
  - **Cause & Effect**: Answers the "why" question or predicts what is likely to happen next (effect), given an event that has occurred (cause).
- **Examples**
  - **[True]** If you touch a non-LED light bulb that's been turned **on** for hours, it will feel hot.
    [False] If you touch a light bulb that's been turned **off** for hours, it will feel hot.
  - **[True]** While in a windy rainstorm, you should always point your umbrella **into the wind**.
    [False] While in a windy rainstorm, you should always point your umbrella **away from the wind**.
  - **[True]** If my sink clogs with debris, I might be able to unclog it by twirling a long **chopstick** in there to dislodge it.
    [False] If my sink clogs with debris, I might be able to unclog it by twirling a long **hair** in there to dislodge it.
  - **[True]** If it is dark outside, opening the blinds **will not** help you see.
    [False] If it is dark outside, opening the blinds **will** help you see.
- **Tip(s)**
  - To generate both sentences of a pair, you can simply find a **contradictory physical concept**, or **negate the sentence**.
  - The topic and verb lists below can be helpful when you need some inspiration.

To Inspire Your Creativity

- Whenever you are not sure what to begin with or need some inspiration: here is a google doc summarizing the potential topics for each domain.
- Or, check out this list of 700 common verbs with examples. You may start sentence construction with any verb in this list.

Figure 6: Screenshot of the creation interface (instruction section).

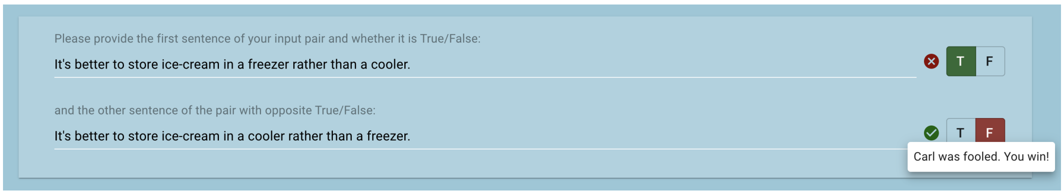

Please provide the first sentence of your input pair and whether it is True/False:

It's better to store ice-cream in a freezer rather than a cooler.

and the other sentence of the pair with opposite True/False:

It's better to store ice-cream in a cooler rather than a freezer.

Carl was fooled. You win!

Figure 7: Screenshot of the creation interface (1/3 input section).

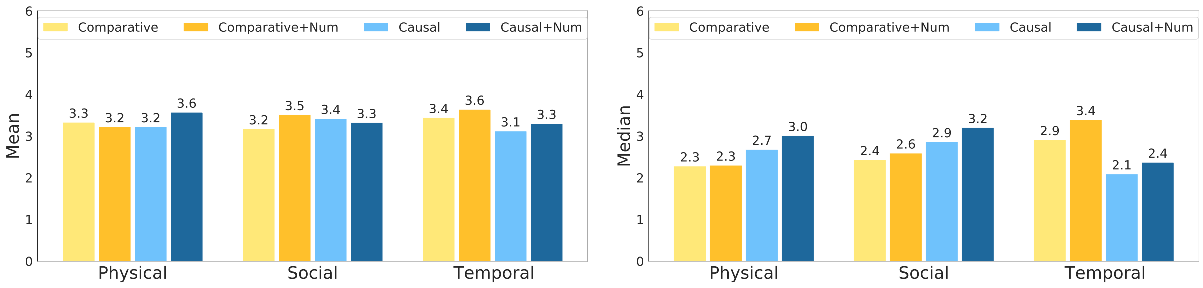

(a) Mean (b) Median

Figure 8: Mean and median of the time needed (in minutes) to fool a sentence for all categories, "`+Num`" denotes numeracy involved.

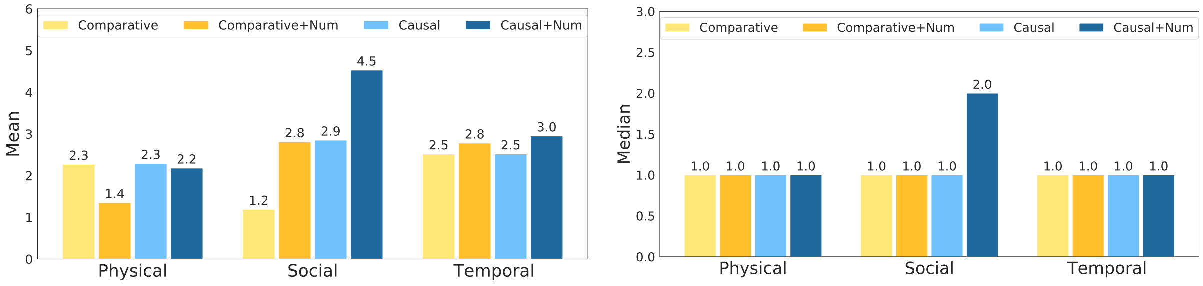

(a) Mean (b) Median

Figure 9: Number of attempts (*i.e.* # revisions + 1) per sentence for all categories, "`+Num`" denotes numeracy involved.

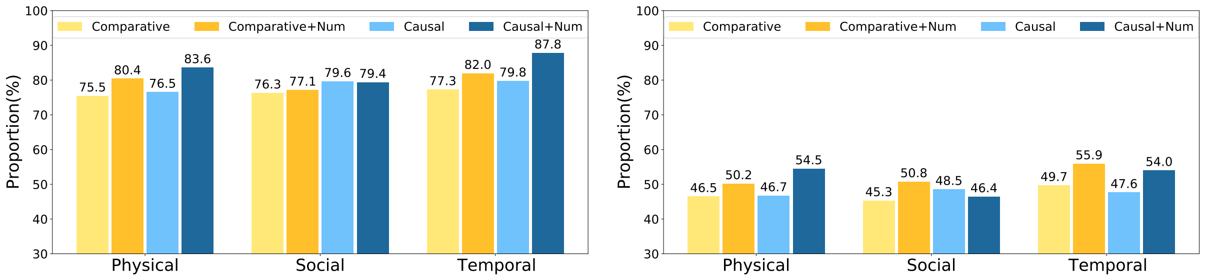

(a) Standard (b) Pairwise

Figure 10: Sentence fooling rates for all categories, "+Num" denotes numeracy involved.

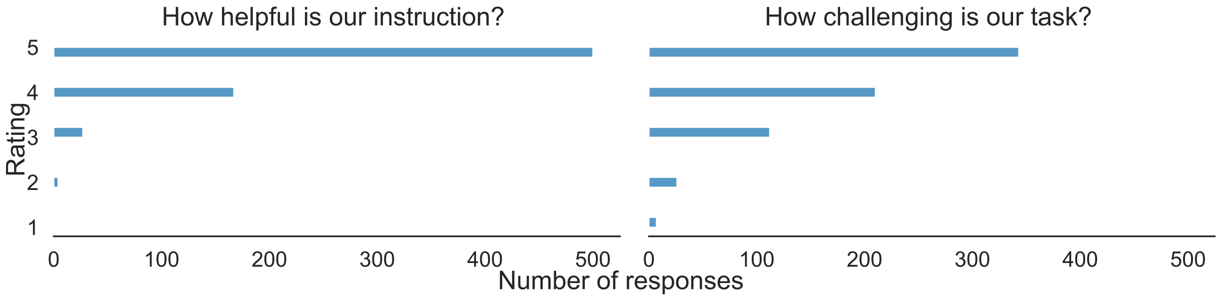

Figure 11: Rating distribution in exit survey.

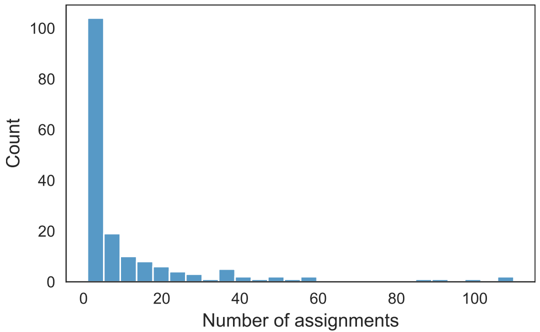

Figure 12: Worker counts over the different numbers of assignments.

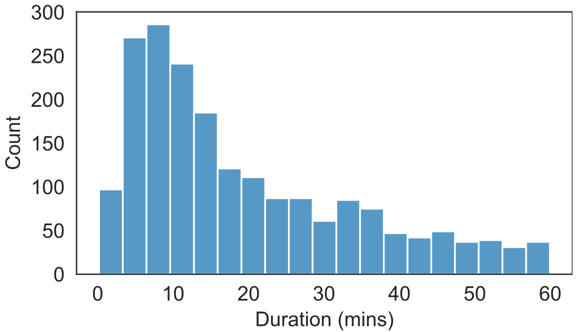

Figure 13: Worker counts over different assignment duration (in minutes).

(duration, in minutes) each worker spent on one assignment.

B Additional Details on Baseline Models

We include several essential implementation details of the benchmark models in the following:

Bi-LSTM+GloVe Our Bi-LSTM model (Hochreiter and Schmidhuber, 1997) is one-layered with a 512-dimensional hidden layer, which takes input word embeddings from 300-dimensional GloVe word embeddings (Pennington et al., 2014). We train all LSTM layers from scratch.

BERT-base Models For BERT-style architectures we employ a multi-layer-perceptron (MLP) on top of the [CLS] special token for binary prediction.

T5-large To adopt T5-large's text-to-text format to our dataset, we use the prefix *com2sense sentence:* and the labels *True* and *False* as model output.

UnifiedQA Models We use two UnifiedQA Models. One with the T5-large backbone and one with the T5-3b backbone. For these models, we use *Is the following sentence correct?* as the prefix, to create a question. Then as the answer we use *Yes* / *No*.

C More Details on the Experiments

C.1 Hyperparameters

All the essential hyperparameters used throughout this work can be referred to in Table 10. We also include the search bounds as well as the number of trials in searching for our manually-tuned hyperparameter search procedures in Table 10.

Model	# Params	Batch-Size	LR	Training Iterations	Gradient Accumulation Steps	Max. Token Length
BiLSTM+GloVe	3.5M	64	1×10^{-5}	100	4	80
BERT-base	109.5M	64	1×10^{-5}	100	4	80
RoBERTa-large	355.4M	32	1×10^{-5}	100	4	80
DeBERTa-large	405.2M	32	1×10^{-5}	100	4	80
T5-large	737.5M	8	1×10^{-5}	100	4	80
UnifiedQA-t5-large	737.5M	8	1×10^{-5}	100	4	80
UnifiedQA-t5-3b	3000M	2	1×10^{-5}	100	8	64
Bound (lower-upper)		1-64	5×10^{-5}–1×10^{-6}	10-100	1-10	
Number of trials		2-3	2-3	2-3	2-3	

Table 10: Hyperparameters used for each model during finetuning on COM2SENSE along with the search bounds for them: *LR* denotes the learning rate that does not change during the training process. All the models are trained with Adam optimizers (Kingma and Ba, 2015). We include number of parameters of each model in the first column, denoted as *# params*.

C.2 Validation Set Results

We validate all trained models on a 402-pair validation set and tune the hyperparameters accordingly. The performances on the validation set are reported in Table 11.

Model	Standard	Pairwise
Random	50.00	25.00
BiLSTM+GloVe	52.80	27.50
BERT-base	57.07	23.11
RoBERTa-large	62.81	38.30
T5-large	62.81	35.82
UnifiedQA-large	63.43	37.31
DeBERTa-large	66.29	43.03
UnifiedQA-3b	**75.12**	**56.22**

Table 11: Validation-set accuracy for selected models, trained and evaluated on respective datasets.

C.3 Performance Across Input Lengths

Although the sentence length in our dataset varies, we find no obvious relation between the length of the sentences and the difficulty for the model to comprehend, in terms of accuracy. As depicted in Figure 14, we therefore conclude that sentence length would not have significant influence on fooling models including DeBERTa-large.

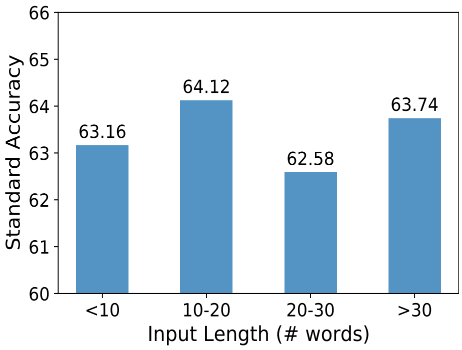

Figure 14: Standard accuracy of the DeBERTa-large model measured on subsets of data with different input lengths.

C.4 Software, Hardware, & Other Details

Transformer-based models are implemented via the HuggingFace PyTorch API (Wolf et al., 2020). All the benchmarked models, except for UnifiedQA-T5-3b are trained on Nvidia GeForce 2080Ti GPUs[5] on a CentOS 7 operating system. The UnifiedQA-T5-3b is trained on NVIDIA Tesla V100 GPUs [6] on an Ubuntu 18 operating system. The T5-large and UnifiedQA-T5-large are trained using the model parallelism approach on two GPUs. The UnifiedQA-T5-3b is trained using model parallelism on 8 GPUS.

The maximum training time is approximately 6 hours for all the models, with the BERT-style models on the lower end of the range and the T5-style models on the higher end.

---

[5] https://www.nvidia.com/en-us/geforce/graphics-cards/rtx-2080-ti/

[6] https://www.nvidia.com/en-gb/data-center/tesla-v100/

Towards Knowledge-Grounded Counter Narrative Generation for Hate Speech

Yi-Ling Chung[1,2], **Serra Sinem Tekiroğlu**[1], **and Marco Guerini**[1]

[1]Fondazione Bruno Kessler, Via Sommarive 18, Povo, Trento, Italy
[2]University of Trento, Italy

`ychung@fbk.eu, tekiroglu@fbk.eu, guerini@fbk.eu`

Abstract

Tackling online hatred using informed textual responses – called counter narratives – has been brought under the spotlight recently. Accordingly, a research line has emerged to automatically generate counter narratives in order to facilitate the direct intervention in the hate discussion and to prevent hate content from further spreading. Still, current neural approaches tend to produce generic/repetitive responses and lack grounded and up-to-date evidence such as facts, statistics, or examples. Moreover, these models can create plausible but not necessarily true arguments. In this paper we present the first complete knowledge-bound counter narrative generation pipeline, grounded in an external knowledge repository that can provide more informative content to fight online hatred. Together with our approach, we present a series of experiments that show its feasibility to produce suitable and informative counter narratives in in-domain and cross-domain settings.

1 Introduction

Standard approaches for online hate mitigation generally rely on content moderation, ranging from deletion of hate content and suspension of user accounts to shadow banning. However, these approaches may draw limits on free speech and diverse opinions. An alternative approach is to directly intervene in the conversation with counter narratives. A Counter Narrative (`CN`) is a non-negative response to a Hate Speech (`HS`), targeting and contradicting extreme statements with fact-bound arguments or alternative viewpoints (Benesch, 2014; Schieb and Preuss, 2016). Such strategy seeks to de-escalate the conversation, disengage from hateful sentiment and encourage mutual understanding through exchange of opinions. Many Non-Governmental Organizations (NGOs) specialized in hate countering are already adopting this approach by training operators to compose counter narratives. According to NGO guidelines[1], proper `CN`s should also include credible evidence.

HS: *The world would be a better place without Muslims. They are only killing and raping our children.*
CN1: *Evidence for this? This is not true. How can you say this about an entire faith?*
CN2: *On the contrary, most children abuse is operated by people they know: a relative, family friends, sports coach, someone in a position of trust and authority. Besides, Muslims help people - A Muslim woman rushed to help the victims of a triple stabbing in Manchester on New Year's Eve.*
CN3: *You are truly one stupid backwards thinking idiot to comment on Muslims like that.*

Table 1: An example hate speech with appropriate counter narratives (`CN1` and `CN2`) and an inappropriate one (`CN3`) containing abusive language.

In Table 1, we present an `HS` along with several `CN`s. Although `CN1` and `CN2` are both appropriate responses, not all appropriate `CN`s are equally effective (Silverman et al., 2016; Tuck and Silverman, 2016). `CN2` is expected to be more effective as it is tailored to the `HS` and demonstrates contextualized and persuasive statements with supporting facts, knowledge, and logical reasoning (Habernal and Gurevych, 2016) rather than a simple generic request for evidence as in `CN1`. Conversely, `CN3` is considered as a natural but inappropriate and aggressive response.

In this context, scaling manual response to massive amounts of online hatred is an undeniably challenging task for NGO operators. Recently, studies have started to focus on data and strategies for the automatic generation of counter narratives in an end-to-end fashion (Chung et al., 2019; Qian et al., 2019; Tekiroğlu et al., 2020; Chung et al., 2020). However, the seminal studies, based on training

---

[1] For example: `https://getthetrollsout.org/stoppinghate`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 899–914
August 1–6, 2021. ©2021 Association for Computational Linguistics

models using only the `HS`-`CN` data samples, do not directly address the required properties for proper and effective `CN`s such as providing credible evidence and knowledge with facts, statistics or examples. While such models possess knowledge to a certain extent through utilizing existing generative language models (LMs) such as GPT-2 (Radford et al., 2019), this knowledge (e.g., events or news) will get out-dated over time. Furthermore, such models are able to create plausible but not necessarily true arguments - a problem known as content hallucination - (Zellers et al., 2019; Solaiman et al., 2019). A hallucinated example of a fact-bound `CN`, generated by GPT-2 model fine-tuned as done by Tekiroğlu et al. (2020), is provided in Table 2. Therefore, in this paper we investigate the generation of knowledge-bound counter narratives, which had never been studied yet to the best of our knowledge. We hypothesized that knowledge infusion can not only leverage generating informative `CN`s but also handling hate speech from unseen domains (i.e. hate targets) without further training (e.g., train on Islamophobia and test on Antisemitism).

HS: *Muslims are invading our country*
CN: *Actually, there were 16,938,000 Muslims in the UK in 2016, so if you exclude London, that is actually increasing by 2%, which doesn't seem very significant.*

Table 2: Hallucinated `CN` generated by GPT-2 that is fluent and credible (according to Office for National Statistics, the Muslim population is just above 3M).

To this end, we explore methodologies to generate informative `CN`s using external knowledge. In particular, we propose an extension of knowledge-grounded generation approaches by adopting an intermediate step where we generate keyphrases for retrieving needed knowledge. So, we first train a counter narrative keyphrase generator, then the generated keyphrases are employed for selecting relevant knowledge sentences. Finally, pre-trained LMs are fine-tuned on the relevant knowledge sentences, together with the `HS` input, to produce knowledge-augmented `CN`s. Our extensive experiments on `CN` generation, including both automatic and expert evaluation, demonstrate that the presented approach produces more specific and tailored responses both for in-domain and zero-shot cross-domain configurations as compared to other approaches, such as standard LMs, that are simply fine-tuned for the task without the use of external knowledge.

As our main contribution, we show that: (i) external knowledge can boost informative `CN` generation, (ii) keyphrase generation improves the quality of retrieved documents, (iii) silver knowledge is utilizable for the task when no gold knowledge is available, (iv) knowledge-bound models are advantageous while tackling zero-shot cross-domain generation especially if (v) using injection of knowledge in large pre-trained LMs.

2 Related Work

In this section we review three main research topics that are relevant for fighting hatred online: (i) studies on `CN` effectiveness in hate countering, (ii) counter-argument generation and (iii) knowledge-guided generation.

Hate countering. Employing counter narratives has shown to be an effective strategy in hatred intervention on social media platforms. Studies have focused on identifying successful counter narratives (Benesch et al., 2016a,b), evaluating their efficacy (Schieb and Preuss, 2016; Silverman et al., 2016; Ernst et al., 2017; Munger, 2017), and analyzing counter speaker accounts characteristics (Mathew et al., 2018). In particular, by analyzing conversations from Twitter, Wright et al. (2017) show that some arguments among strangers induce favorable changes in discourse and attitudes.

Counter-argument generation shares similar objectives as `CN` generation, i.e., to produce the opposite or alternate stance of a statement, but the latter faces peculiar difficulties such as the absence in `HS` of explicit or well-structured 'arguments' (e.g., "*Islam is a disease*") and the limited amount of data available for training. Studies usually focus on domains with large discussions, e.g., politics (Hua and Wang, 2018) and economy (Le et al., 2018). The closest work to ours is counter-argument generation with external knowledge augmentation by Hua et al. (2019). Our approach differs from theirs in three aspects: (i) we explore generating queries to extract knowledge for grounding `CN` with, (ii) pre-trained generative models are utilized for leveraging the knowledge present, (iii) our approach requires less manipulation over knowledge.

Knowledge-guided generation. There is a growing interest in exploiting external knowledge to generate informative responses for applications such as dialog systems (He et al., 2017; Young et al., 2018) and question answering (Das et al., 2017; Saha et al., 2019). Previous approaches inject knowledge through topic phrases (Fan et al., 2019), structured knowledge graphs (Zhou et al., 2018) and unstructured texts (Dinan et al., 2019; Hua et al., 2019).

3 HS-CN Dataset

To the best of our knowledge, there is no high-quality hate speech - counter narrative dataset available yet where `CN`s are explicitly paired with relevant knowledge. Since constructing such dataset with a decent-size would be too costly and out of the scope of the present paper[2], we resort to a "reverse-engineering" strategy such that we automatically paired relevant knowledge with an already existing high quality `CN` dataset. We chose `CONAN` (Chung et al., 2019), which is a dataset niche-sourced to expert NGO operators offering high quality `CN`s, and the best and most diverse material among the other `CN` datasets (Tekiroğlu et al., 2020). `CONAN` consists of 6645 English pairs of `HS-CN` including: 1288 original pairs, 2576 pairs where two paraphrases of the original `HS` are paired with the original `CN`, and 2781 translated pairs from French and Italian. The English data is split into 4069/1288/1288 samples for train/dev/test.

4 Architecture

Our architecture, illustrated in Figure 1, consists of a knowledge retrieval module that retrieves sentence-level relevant knowledge, and a generation module that generates a counter narrative. Specifically, the knowledge retrieval module first prepares variants of a query Q for a given hate speech `HS` using two strategies: query extraction (Q_{hs}) and automatic query generation (Q_{gen}). Then, the obtained queries are employed to search for relevant knowledge articles via a search engine. Finally, it uses a sentence selector to filter and rank the most relevant sentences as the relevant knowledge (`KN`) from the retrieved articles. For the counter narrative generation module, we fine-tuned several LMs that take a `HS` and the ranked knowledge sentences `KN` as input and output a corresponding counter narrative.

5 Knowledge Retrieval Module

The knowledge retrieval module in the architecture incorporates a knowledge repository, query construction sub-module, and a knowledge sentence selection sub-module.

5.1 Knowledge Repository

Previous approaches on introducing external knowledge for dialog generation have exploited unstructured and structured knowledge. Since no structured knowledge is available for the hate speech domain, we rely on unstructured textual knowledge in the format of articles, which allows for updating the knowledge repository easily. Considering that the proliferation of `HS` is also triggered with target-related events (e.g., terrorist attacks), being able to update the knowledge, such as news articles, would let us produce proper `CN`s that contain the latest statistics or evidence from the current events.

We include `Newsroom` (Grusky et al., 2018) and `WikiText-103` (Merity et al., 2017) to our knowledge repository. `WikiText-103` is a large collection of 28,595 full Wikipedia articles covering over 103 million words. `Newsroom` consists of 1.3 million articles extracted from major news publications between 1998 and 2017, featuring over 6.9 million words.

5.2 Query Construction

To construct comprehensive and proper queries to search for relevant knowledge for the data pairs, we applied two strategies: (i) query extraction and (ii) query generation. In both strategies, the query is composed of keyphrases that can be defined as the important and topical phrases from a text (Turney, 2000).

Query extraction. We extracted keyphrases from `CONAN` dataset using Keyphrase Digger (Moretti et al., 2015), a multilingual keyphrase extraction system that uses statistical measures and linguistic information, and is proven to be one of the best systems for unsupervised settings[3]. Following the knowledge retrieval strategy using input argument by Hua et al. (2019) for counter argument generation, we first obtained the `HS` keyphrases to construct the initial query Q_{hs}. However, `HS`s from `CONAN` mostly contain hateful and simplistic

---

[2] Obtaining access to a pool of trained NGO operators is very complicated, furthermore keeping track of their search activity and the material they used during `CN` production would require long and complex data collection sessions that might span several months.

[3] Keyphrase Digger is a new implementation of KX (Pianta and Tonelli, 2010) with several improvements, and it was ranked the best performing unsupervised system on task 5 of SemEval 2010 evaluation campaign (Moretti et al., 2015).

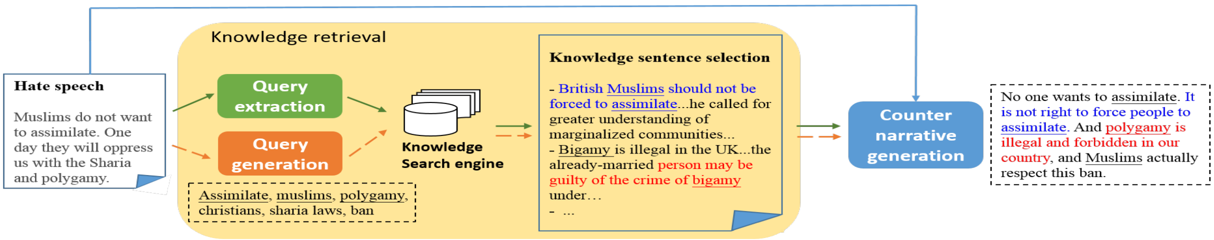

Muslims do not want to assimilate. One day they will oppress us with the Sharia and polygamy.

Assimilate, muslims, polygamy, christians, sharia laws, ban

- British Muslims should not be forced to assimilate...he called for greater understanding of marginalized communities...
- Bigamy is illegal in the UK...the already-married person may be guilty of the crime of bigamy under…
- ...

No one wants to assimilate. It is not right to force people to assimilate. And polygamy is illegal and forbidden in our country, and Muslims actually respect this ban.

Figure 1: Architecture of knowledge grounded generation with extracted (green solid arrow) and generative (dotted arrow) queries (topical phrases) that are exploited to retrieve relevant knowledge. The knowledge sentences extracted together with input HS are fed to CN generation. We give the example of generative approach.

phrases in comparison to the input arguments used by Hua et al. (2019) that can be rich in content[4]. Therefore, in the HS-CN scenario, we hypothesize that the keyphrases from Q_{hs} alone would not be sufficient for relevant knowledge search especially for mapping the knowledge onto training data.

To this end, we also extracted keyphrases from CN together with HS to increase the possibility that the retrieved knowledge sentences contain pieces of information found in the ground truth. Hence, the second query $Q_{hs\cup cn}$ contains CN keyphrases for the relevancy to the target CN and HS keyphrases for preserving the hate context. We investigated the effects of various keyphrase query configurations in terms of HS relevancy and $Q_{hs\cup cn}$ is proven to be the best configuration (See Appendix A.1 for more details.).

Query generation. Since the best query configuration $Q_{hs\cup cn}$ cannot be available at test time, we need a way to obtain keyphrases that serve as CN cues for searching knowledge sentences during the CN generation. To this end, we built a query generation model that takes HS as input and outputs a comma-separated list of CN keyphrases, which is then used as Q_{gen}. Our aim is to obtain an approximation of $Q_{hs\cup cn}$ via $Q_{hs\cup gen}$ at the test time.

The model is trained using Transformer (Vaswani et al., 2017) architecture as it has obtained state-of-the-art performances for generation tasks (Dinan et al., 2019; Ghazvininejad et al., 2018). For the training data, we used CONAN dataset and discarded the CNs that are less than 10 words, since they are usually generic, poor in terms of argumentative content and cannot provide a meaningful search (e.g., "*No they are not - prove this?*", "*What does that even mean?*", "*Any evidence?*"). Accordingly, we kept 4038/1257/1257 instances for train/dev/test set. The train set includes the pairs marked as original in the dataset, and all translated pairs from French and Italian; the dev set consists of one paraphrase of each original HS and its CNs; and the test set contains the rest of the paraphrased HSs. The training inputs are represented as HS $[HS\_end\_token]$ KP $[KP\_end\_token]$, where KP is the list of keyphrases extracted from the gold CN.

The model has been trained following the configuration of the base model in (Vaswani et al., 2017): with 6 transformer layers, 8 transformer heads, embedding size of 512, hidden size of 2048, dropout rate of 0.1, batch size of 64 for 100 epochs. The training time lasted around 7 hours. All experiments in this paper have been conducted on a Nvidia Tesla V100 GPU. For decoding, we used nucleus sampling (Holtzman et al., 2020) with a p value of 0.9.

We report keyphrase generation results in terms of BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) against the keyphrases extracted from gold test CNs. We obtained a score of 0.162 for BLEU-2 and a score of 0.353 for ROUGE-L. Although both scores can be considered as low, this is due to the open ended nature of the set of possible CN keyphrases for a given HS. Example queries for a single pair, extracted from its HS, its CN, and generated with the keyphrase generation model are shown in Table 3.

5.3 Knowledge Sentence Selection

We use Solr[5] to index the articles and retrieve those relevant to a given query based on the similarity between the articles and the query using BM25 (Robertson et al., 1995). Once the queries have been obtained either through extraction or generation, they are presented to Solr for retrieving

---

[4] e.g., an argument "*A universal basic income will help the labor market adapt to inevitable disruptions caused by advancements in automation and artificial intelligence.*" vs. HS "*Islam is a disease*".

[5] https://lucene.apache.org/solr/

HS	CN	Query	Knowledge sentences (KN)
Islam is a disease.	Like Christianity or any other, islam is a religion of tolerance. Disease does not discriminate on religious grounds.	Q_{hs}: islam, disease	(i) Do Muslims want to heal from the disease...? (ii) Being infected by religious extremism is like being infected by a disease...
		Q_{gen}: islamic law, god, christians	(i) Islamic law is to create an environment...submission to God. (ii) Certain areas of the Muslim world have always been home to large populations of Christians...
		Q_{cn}: tolerance, christianity, discriminating	(i) Islam is a 1400-year-old religion that preaches tolerance...like Christianity and... (ii) Disease does not discriminate...on religious grounds...

Table 3: Examples of KN retrieved using queries extracted from HS (Q_{hs}), generated (Q_{gen}) and created from both HS and CN keyphrases (Q_{cn}).

the 25 top-ranked articles. Next, we used spaCy sentence segmentation[6] to split an article into sentences. Similar to Zhang et al. (2020), given a query Q we score each sentence x_i in the set of articles D independently, using ROUGEL-F1 (Lin, 2004) as in Equation 1.

$$s_i = rouge(x_i, Q), \forall i \in D \qquad (1)$$

In the final step, we distilled the knowledge by keeping the top 5 knowledge sentences that have the highest scores among 25 top-ranked articles. Instead of a more stringent filtering, such setting has been applied to grant a better variety of source articles and corresponding distilled sentences. We refer to such automatically associated sentences as “silver knowledge”.

6 Counter Narrative Generation Module

Large pretrained LMs require less amount of high-quality data to be fine-tuned on downstream tasks while providing strong performances and they already store large amount of factual and commonsense knowledge from their training data (Petroni et al., 2019). To this respect, we built the following models: (1) **GPT-2$_{KN}$**, obtained by fine-tuning GPT-2 on CONAN data paired with KN; (2) **GPT-2$_{KN,MT}$**, by fine-tuning GPT-2$_{KN}$ in a multi-task learning fashion for learning to distinguish CNs from HS as next utterances; (3) **XNLG** (Chi et al., 2020) for its ability to copy information to the output (in our case the retrieved KN to be copied to the CN). We expect all three models to attend over the HS and retrieve KN and look for the relevant snippets to be recovered while generating a CN.

6.1 Models

The training HS-CN pairs are represented as $HS\ [HS\_end\_token]\ KN\ [KN\_end\_token]\ CN\ [CN\_end\_token]$. Each model is trained with $Q_{hs \cup cn}$ and then tested on Q_{hs}, Q_{gen}, and $Q_{hs \cup gen}$. We also tested the models with $Q_{hs \cup cn}$ to define an oracle scenario with an upperbound performance when the data can only be paired with silver knowledge.

GPT-2$_{KN}$. We fine-tuned the GPT-2[7] medium model for 3 epochs with a batch size of 2048 tokens. We used Adam optimizer with a learning rate of 5e-5. At inference time, responses were generated employing nucleus sampling with a p value of 0.9, conditioned on HSs and corresponding KN.

GPT-2$_{KN,MT}$. Since we noticed that GPT-2 occasionally produces responses that contain fragments of abusive language, we combined the language modeling objective with a next-sentence prediction objective for fine-tuning GPT-2 in a multi-task setting, inspired by Wolf et al. (2018). Next-sentence prediction adopts a linear classification layer added to the last layer of the transformer language model and then applies a cross-entropy loss to classify a proper next response to the input HS from 2 distractors randomly selected from HS. We used Adam optimizer with a learning rate of 5e-5 and empirically fine-tuned it for 1 epoch and the same sampling strategy as GPT-2$_{KN}$ has been applied.

XNLG is a pre-trained Transformer-based language model trained on Wikipedia dumps with two relevant objectives for our task: to obtain contextual representations and to recover a given input. We fine-tuned XNLG[8] for generating counter narratives on all layers with a batch size of 10 for 100 epochs. We used Adam optimizer with a learning rate of 1e-4. We tokenized and removed accent from the entire dataset and applied the same BPE

---

[6] https://spacy.io/universe/project/spacy-sentence-segmenter

[7] https://github.com/huggingface/transformers

[8] https://github.com/CZWin32768/XNLG

codes used by Chi et al. (2020). For `KN` and `CN` we kept the first 256 tokens, while setting the `HS` to 70 tokens, which is the maximum length of hate speech in the dataset. We experimented with various decoding methods and adopted beam search with a beam-width of 3 for the best performing setting (details in Appendix A.2).

Baselines used for comparison are: (1) non-pretrained **Transformer** without knowledge using the same hyper-parameters as keyphrase generation model; (2) **GPT-2** without knowledge following the same configuration as GPT-2_{KN}; (3) **Candela** (Hua et al., 2019), an LSTM-based state-of-the-art knowledge-driven architecture for argument generation. Since `CONAN` is relatively small, we hypothesize that a pre-training procedure[9] on data from a similar task (argument generation) can be beneficial for generalization and porting knowledge. Thus, we first pre-trained Candela architecture on argument generation dataset (Hua et al., 2019), following the configuration described in the paper. We then fine-tuned the model for 20 epochs on `CONAN` with `KN` using Q_{hs} as it is done in the original setting of Candela.

6.2 Results for the Silver Knowledge Test Set

We report BLEU-2 (B-2) and ROUGE-L (R-L) scores for all proposed models and baselines in Table 4 on the test split of `CONAN` that we automatically paired with silver knowledge using various queries. We also measure the capability of each model to produce *novel* responses with respect to the training data by Jaccard similarity (Wang, 2018), and *diverse* responses for the given input by repetition rate (RR) (Cettolo et al., 2014).

Among our models, GPT-2_{KN} yields the highest B-2 and XNLG the highest novelty, diversity, and R-L. The notably improved novelty achieved by knowledge-grounded models indicates the benefits of adding knowledge on producing `CN`s, in comparison to the baselines - particularly Transformer `TRF`. On the other hand, the quantitative performance of XNLG does not reflect its true performance in terms of quality. A quick glance at the output `CN`s showed that XNLG model copies almost everything from `KN` to the output instead of a proper `CN` generation, increasing the novelty and diversity scores. The issue can easily be observed from the average numbers of words and sentences in the XNLG output in comparison to the outputs of the other models presented in Table 4. GPT-$2_{KN,MT}$ falls behind among our models in terms of RR, B-2, and R-L, still providing a competitive novelty. Regarding Candela, while it obtained similar performances to our models in terms of R-L and B-2, the generation is repetitive and less novel.

As for the testing with different query types, $Q_{hs \cup gen}$ induces more novel responses than $Q_{hs \cup cn}$ and Q_{hs}. While XNLG yields the highest novelty with Q_{hs} (0.824), it can be explained again with the problem of copying the whole `KN`, which is more varied due to the less restrictive search using only `HS`.

The oracle query $Q_{hs \cup cn}$, in which we deliberately provide the best knowledge possible through the keyphrases containing also from the gold `CN`, yields the best R-L scores among the query variations of knowledge-grounded models. Among all the models, $Q_{hs \cup cn}$ also leads to the best B-2 through GPT-2_{KN}, and the best R-L through XNLG, as we have anticipated. Finally, $Q_{hs \cup gen}$ outperforms Q_{hs} and Q_{gen} over most metrics, hinting at the advantages of using generated queries together with hate context for silver-knowledge retrieval.

We have also conducted complementary experiments by taking into consideration the design choices and the various phenomena in our study. Since in our test set, in line with `CONAN`, a `HS` can be paired with more than one `CN`, Q_{hs} would retrieve the same `KN` for all the target `CN`s of the same input `HS`. Contrarily, we obtain a different set of `KN` using queries Q_{gen}, $Q_{hs \cup gen}$ and $Q_{hs \cup cn}$ for each target `CN`. Therefore, we also report an evaluation on unique `HS`-`CN` pairs, where a single target `CN` has been randomly chosen for each `HS`, among all query types in Appendix A.4. Finally, to simulate Candela configuration (that uses only Q_{hs}) also with the other models, we run an additional set of experiments where we used Q_{hs} for retrieving the knowledge for training samples. The results are reported in Appendix A.3.

6.3 Results for the Gold Knowledge Test Set

To isolate the effect of the knowledge retrieval strategies from the knowledge-grounded generation performances, we conducted a second evaluation on a newly crafted test set paired with gold standard knowledge. In this evaluation, in addition to stereotypical islamophobic in-domain (i.e.

---

[9] We also trained Candela from scratch on `CONAN` but decided not to proceed with this setting for the poor performance.

Models	Nov.	RR	B-2	R-L	#Word	#Sent.	`KN` overlap (ngram) 1	2	3
without knowledge									
TRF	0.467	7.72	0.082	0.094	21.47	1.70	-	-	-
GPT-2	0.688	9.04	0.045	0.100	15.95	1.35	-	-	-
Train$_{cn}$	-	3.91	-	-	21.79	1.87	0.307	0.054	0.016
with knowledge									
Candela (Q_{hs})	0.692	21.87	0.040	0.098	23.85	2.47	0.173	0.008	0.001
GPT-2$_{KN}$									
w/ Q_{hs}	0.723	8.13	0.082	0.094	15.60	1.32	0.258	0.023	0.008
w/ Q_{gen}	0.728	7.48	0.067	0.091	12.75	1.17	0.260	0.050	0.019
w/ $Q_{hs\cup gen}$	0.735	6.30	0.085	0.103	15.35	1.59	0.358	0.068	0.024
w/ $Q_{hs\cup cn}$	0.727	7.17	**0.166**	0.110	13.10	1.16	0.282	0.058	0.022
GPT-2$_{KN,MT}$									
w/ Q_{hs}	0.744	11.69	0.050	0.090	13.35	1.17	0.269	0.049	0.017
w/ Q_{gen}	0.731	10.37	0.052	0.092	13.34	1.14	0.253	0.044	0.017
w/ $Q_{hs\cup gen}$	0.747	7.59	0.091	0.090	16.91	1.26	0.269	0.033	0.009
w/ $Q_{hs\cup cn}$	0.731	9.56	0.048	0.107	13.05	1.13	0.276	0.057	0.023
XNLG									
w/ Q_{hs}	**0.824**	14.42	0.073	0.084	55.51	3.71	0.841	0.650	0.558
w/ Q_{gen}	0.819	6.88	0.097	0.084	55.64	3.64	0.849	0.656	0.558
w/ $Q_{hs\cup gen}$	0.812	6.98	0.074	0.089	57.58	3.00	0.828	0.579	0.475
w/ $Q_{hs\cup cn}$	0.819	**5.69**	0.076	**0.116**	55.69	3.42	0.840	0.631	0.529

Table 4: Results of `CN` generation with silver knowledge. We report novelty (Nov.), RR, BLEU-2 (B-2), ROUGE-L (R-L), `KN` overlap with generation and the average amount of words and sentences per generation.

	in-target				cross-target			
	Nov.	RR	B-2	R-L	Nov.	RR	B-2	R-L
TRF	0.30	7.57	0.014	0.10	0.46	8.62	0.015	0.08
GPT-2	0.72	8.53	0.020	0.11	0.72	8.01	0.022	0.09
Candela	0.69	19.31	0.072	0.10	0.70	22.22	0.022	0.09
GPT-2$_{KN}$	0.71	**6.85**	0.201	0.19	0.75	**6.33**	0.041	0.19
GPT-2$_{KN,MT}$	**0.85**	11.55	0.066	0.12	**0.86**	10.38	0.022	0.11
XNLG	0.83	6.94	**0.256**	**0.33**	0.84	8.15	**0.291**	**0.35**

Table 5: Results of `CN` generation with gold knowledge in-target and cross-target test sets.

in-target) scenario, we also explore the effect of knowledge infusion on cross-domain (i.e. cross-target) `CN` generation under zero-shot setting. We hypothesize that having a system trained to make use of substandard silver knowledge to generate proper `CN`s for a given context, could be robust to cross-domain zero-shot conditions. Therefore, we organized a data collection session with an expert operator in writing `CN`s. In this session, 50 islamophobic `HS`s randomly sampled from `CONAN` and 144 new cross-target `HS`s (covering misogyny, antisemitism, racism, and homophobia) are provided along with the knowledge retrieved by $Q_{hs\cup cn}$ queries. The expert is tasked with composing a suitable `CN` using the corresponding knowledge as much as possible. Thus, we could obtain a gold test set[10]. in which the input knowledge can certainly be found in the `CN`s.

We tested all models with gold knowledge in-domain and cross-domain test cases. Results are given in Table 5. For in-domain scenario, as we have anticipated, knowledge grounded models yield better performances in B-2 and R-L in comparison to the silver knowledge test setting. Especially with the striking jump in the performance of GPT-2$_{KN}$, we can confirm the proper infusion of the given knowledge to the generated `CN`s. As for cross-domain tests, GPT-2$_{KN}$ still yields better performance than baselines while the performance for all models (except for XNLG) drops due to unseen events during training. All GPT-2 variations present better diversity performances on the cross-domain setting as compared to both in-domain and silver-knowledge settings. Regardless of domains, XNLG yields fallaciously high scores due to its extensive copying. A cross-target generation from the models can be seen in Table 7. More examples in-/cross-domain generations from all the models are provided in Appendix A.6.

Human evaluation. We further resort to human evaluation to assess the final generation quality of

---

[10] We release the gold test set at `https://github.com/marcoguerini/CONAN`.

each model. For this reason we perform human evaluation of generation using gold knowledge, to rule out the effect of possible noise in the knowledge that may result from the retrieval process.

Our models were evaluated by 3 expert operators from the NGO Stop Hate UK. The annotators are already experienced, and specifically trained, in reading hateful content and writing `CN`s for online hate countering[11]. The annotators are instructed to assess all generated pairs in gold knowledge test sets in terms of *suitableness* to the `HS`, *informativeness*, and *intra-coherence* of `CN` regardless of `HS`. Each score is on a scale of 1 (the least) to 5 (the most). To avoid possible bias and hints towards models, we normalized the pairs (e.g., lowercase and space between words and punctuation) and divided them into 3 partitions of randomized files for experts (See Appendix A.5 for annotation instruction). Each expert was given 388 pairs, resulting in a total of 1164 pairs for evaluation. To avoid excessive workload annotators were allowed to complete the task over multiple sessions at their preference.

Results are reported in Table 6. We also computed Kendall's Tau-b (Kendall, 1938) to measure the annotators' agreement towards the model ranking for each aspect. The high correlations indicate a strong concordance among the annotators (threshold tau-b > 0.35). Regardless of domains, annotators consider XNLG generations as the most informative and GPT-2$_{KN}$ generations as the most suitable. TRF yields a reasonable suitableness and coherence since it tends to memorize the training `CN`s, almost behaving like a retrieval system on human responses. However, such behavior can be fatal in cross-domain settings. Candela fails to generate suitable cross-domain `CN`s despite preserving the intra-`CN` coherence. While GPT-2 and GPT-2$_{KN}$ generations are found almost equally coherent, the lower suitableness and informativeness of GPT-2 output (2.26 and 1.92) for cross-domain as compared to GPT-2$_{KN}$ (2.51 and 2.29) encourages the grounding `CN`s in knowledge.

7 Discussion

Our findings suggest that a large pre-trained LM with knowledge injection is preferred to alleviate the demand for gold data and improves in-/cross-domain generations. GPT-2$_{KN}$ outperforming GPT-2, which becomes more clear with every increase in the quality of provided `KN` (i.e., from silver Q_{gen} to silver $Q_{hs \cup gen}$ to gold in-domain), indicates that our model trained on silver knowledge learns to grasp relevant `KN` to the given `HS` instead of random copying. As for GPT-2$_{KN,MT}$, since the model is trained to distinguish `CN` from `HS`, certain typical counter responses are prone to appear. Through a manual analysis, we confirmed that its responses tend to be simpler (e.g., negation: "*terrorism is **not** a religion.*", "*It's **not** true...*") than the responses of other GPT-2 variant models. Future work can explore alternative multitask learning frameworks to improve the generation.

	in-domain			cross-domain		
	suit.	info.	cohe.	suit.	info.	cohe.
TRF	2.65	2.25	3.39	1.47	2.09	3.45
GPT-2	2.67	2.16	4.10	2.26	1.92	**4.24**
Candela	2.41	2.25	3.14	1.42	2.09	3.40
GPT-2$_{KN}$	**3.02**	2.35	**4.33**	**2.51**	2.29	4.21
GPT-2$_{KN,MT}$	1.76	1.65	3.73	2.03	1.76	3.88
XNLG	1.43	**3.88**	2.12	1.88	**4.10**	2.79
Kendall's tau-b	0.82	0.69	0.82	0.51	0.91	0.73

Table 6: Human evaluation results of `CN` generation.

From a practical perspective, we can conclude that fine-tuning a pretrained language model for conditional generation, utilizing silver-knowledge sentences retrieved through a query generation model is the most promising approach for knowledge-bound `CN` generation. In fact, (i) among various models we tried, GPT-2 with knowledge infusion is shown to produce the most informative and suitable output. (ii) At deployment time, using input context together with generated keyphrases becomes prominent with more novel and diverse `CN`s and (iii) for new hate-targets in a zero-shot scenario, the generation model can produce coherent and suitable `CN`s without retraining if a reasonable set of input knowledge is provided. Moreover, this approach can be applied to other scenarios, such as dialogue response generation, since it does not require any extra data collection or annotation.

8 Conclusion

Online hate speech intervention is a challenging problem and research on counter narrative generation is still in its infancy. In this work, we have proposed methods for improving counter narrative generation to fight hatred online, incorporating external knowledge retrieved through extracted and generated keyphrases. To this end, a dataset of hate-speech/counter-narrative pairs was augmented with relevant knowledge to train systems that are able to

---

[11] The compensation for annotation work met with EU regulations.

HS: Jews are different from non-Jews not because of their beliefs, but because of their physical nature. “Jewishness” and malevolence is inherited through blood.

Human: Today one of the biggest concerns is the reinvention of clinical concepts of race, based on inaccurate generalizations. The notion that the superiority of the white race was part of the natural order was deviously reinforced by the rise of modern ”scientific” racism in the late 19th century. Despite the strong objections of many of our contemporaries, this idea of a ”jewish race” is still going around.

GPT-2: Muslims are not different from non-Muslims because of their faith.

GPT-2$_{KN}$: I think that the notion of race is a generalization based on ignorance.

GPT-2$_{KN,MT}$: All that is wrong is that we cannot accept that violence is necessary.

XNLG: today , one of the biggest concerns is the reinvention of clinical concepts of race , based on inaccurate generalizations about the prejudice to certain physical characteristics and civilizations . despite the strong objections of many contemporary contemporaries , the notion of fixed ” race ” - packages of physical and behavioral characteristics

Table 7: Samples of cross-domain generation.

produce suitable and informative arguments. Our experiments on in-/cross- domain generation indicate that the generated responses can meet these desiderata. As future work we plan to test other query generation approaches (e.g., exploit human-crafted queries in an interactive setup or the use of pre-trained LMs) to improve knowledge selection and to test other architectures for the final generation step.

Acknowledgments

This work was partly supported by the HATEMETER project within the EU Rights, Equality and Citizenship Programme 2014-2020. We are deeply grateful to Stop Hate UK and its volunteers for their help and effort in evaluating the output of our systems.

Ethical Considerations

While we believe that counter narratives are a better tool than content moderation in fighting hate speech (e.g. they do not hinder freedom of speech), still the automatic generation of CNs should be taken with care. Since this work aims at presenting a methodology for knowledge-bound counter narrative production through neural approach, several ethical consequences should be considered.

First, neural models may still produce substandard counter narratives containing abusive language or negative content. To mitigate this issue, possible solutions include (1) integrating in the pipeline a classifier or a human reviewer for validation and possible post-editing (Tekiroğlu et al., 2020), (2) detoxification techniques for controllable generation methods (Gehman et al., 2020), and (3) discarding undesirable content from the corpora used for training (Raffel et al., 2020), even if the appropriate criteria for such purpose are still investigated.

Second, while our approach reduces the risks of content hallucination, an additional step, where the accuracy of the generated text is checked against the provided knowledge (Nie et al., 2019; Dušek and Kasner, 2020), would provide further robustness to the system.

Third, natural language generation models may still induce unintended social biases. This issue can be moderated by measuring/promoting fairness in models and data employed (Blodgett et al., 2020), and designing bias triggers (Sheng et al., 2020) or regularization methods (Bordia and Bowman, 2019; Corbett-Davies et al., 2017) for controllable bias.

To sum up, while some additional automated techniques may help in maintaining generation quality, human evaluation should always be considered as the foremost solution, at least for delicate tasks such as ‘real’ hate countering on social media platforms. For this reason we advocate that generation systems should be used as a suggestion tool for NGO operators, to make their countering work more effective. In this way there is always a “human moderator” taking the final decision (Chung et al., 2019).

References

Susan Benesch. 2014. Countering dangerous speech: New ideas for genocide prevention. *Washington, DC: United States Holocaust Memorial Museum.*

Susan Benesch, Derek Ruths, Kelly P Dillon, Haji Mohammad Saleem, and Lucas Wright. 2016a. Considerations for successful counterspeech. *Dangerous Speech Project.*

Susan Benesch, Derek Ruths, Kelly P Dillon, Haji Mohammad Saleem, and Lucas Wright. 2016b. Counterspeech on twitter: A field study. *Dangerous Speech Project.*

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of “bias” in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 7–15, Minneapolis, Minnesota. Association for Computational Linguistics.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2014. The repetition rate of text as a predictor of the effectiveness of machine translation adaptation. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2014)*, pages 166–179.

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2020. Cross-lingual natural language generation via pre-training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7570–7577. AAAI Press.

Yi-Ling Chung, Elizaveta Kuzmenko, Serra Sinem Tekiroglu, and Marco Guerini. 2019. CONAN - COunter NArratives through nichesourcing: a multilingual dataset of responses to fight online hate speech. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2829, Florence, Italy. Association for Computational Linguistics.

Yi-Ling Chung, Serra Sinem Tekiroglu, and Marco Guerini. 2020. Italian counter narrative generation to fight online hate speech. In *Proceedings of the Seventh Italian Conference on Computational Linguistics*, Bologna, Italy.

Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 797–806, New York, NY, USA. Association for Computing Machinery.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–365, Vancouver, Canada. Association for Computational Linguistics.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations*, New Orleans, LA.

Ondřej Dušek and Zdeněk Kasner. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137, Dublin, Ireland. Association for Computational Linguistics.

Julian Ernst, Josephine B Schmitt, Diana Rieger, Ann Kristin Beier, Peter Vorderer, Gary Bente, and Hans-Joachim Roth. 2017. Hate beneath the counter speech? a qualitative content analysis of user comments on youtube related to counter speech videos. *Journal for Deradicalization*, (10):1–49.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Florence, Italy. Association for Computational Linguistics.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776, Vancouver, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Xinyu Hua, Zhe Hu, and Lu Wang. 2019. Argument generation with retrieval, planning, and realization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2661–2672, Florence, Italy. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–230, Melbourne, Australia. Association for Computational Linguistics.

Maurice George Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Dieu-Thu Le, Cam Tu Nguyen, and Kim Anh Nguyen. 2018. Dave the debater: a retrieval-based and generative argumentative dialogue agent. In *Proceedings of the 5th Workshop on Argument Mining*, pages 121–130.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Binny Mathew, Navish Kumar, Pawan Goyal, Animesh Mukherjee, et al. 2018. Analyzing the hate and counter speech accounts on twitter. *arXiv:1812.02712*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *International Conference on Learning Representations*.

Giovanni Moretti, Rachele Sprugnoli, and Sara Tonelli. 2015. Digging in the dirt: Extracting keyphrases from texts with kd. In *Proceedings of the Second Italian Conference on Computational Linguistics*, Trento, Italy.

Kevin Munger. 2017. Tweetment effects on the tweeted: Experimentally reducing racist harassment. *Political Behavior*, 39(3):629–649.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679, Florence, Italy. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Emanuele Pianta and Sara Tonelli. 2010. KX: A flexible system for keyphrase eXtraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 170–173, Uppsala, Sweden. Association for Computational Linguistics.

Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding, and William Yang Wang. 2019. A benchmark dataset for learning to intervene in online hate speech. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4755–4764, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Unpublished.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication*.

Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200.

Carla Schieb and Mike Preuss. 2016. Governing hate speech by means of counterspeech on facebook. In *66th ICA Annual Conference*, pages 1–23, Fukuoka, Japan.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2020. Towards Controllable Biases in Language Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3239–3254, Online. Association for Computational Linguistics.

Tanya Silverman, Christopher J Stewart, Jonathan Birdwell, and Zahed Amanullah. 2016. The impact of counter-narratives. *Institute for Strategic Dialogue*, pages 1–54.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *arXiv:1908.09203*.

Serra Sinem Tekiroğlu, Yi-Ling Chung, and Marco Guerini. 2020. Generating counter narratives against online hate speech: Data and strategies. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1177–1190, Online. Association for Computational Linguistics.

Henry Tuck and Tanya Silverman. 2016. *The counter-narrative handbook*. Institute for Strategic Dialogue.

Peter D Turney. 2000. Learning algorithms for keyphrase extraction. *Information retrieval*, 2(4):303–336.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Vincent Wang. 2018. Sketching a Chinese writer's vocabulary profile in English: the case of Ha Jin. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2018. Transfertransfo: A transfer learning approach for neural network based conversational agents. *NeurIPS 2018 workshop on Conversational AI: "Today's Practice and Tomorrow's Potential"*.

Lucas Wright, Derek Ruths, Kelly P Dillon, Haji Mohammad Saleem, and Susan Benesch. 2017. Vectors for counterspeech on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 57–62, Vancouver, BC, Canada. Association for Computational Linguistics.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9051–9062.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4623–4629. International Joint Conferences on Artificial Intelligence Organization.

A Appendices

A.1 Analysis on Keyphrase Extraction Configurations

We conducted a preliminary manual analysis to investigate the effects of various keyphrase extraction configurations. We randomly sampled 48 hate speech and counter-narrative pairs from `CONAN` dataset and extracted the keyphrases. Then, we retrieved the `KN` (see Section 5.3) with the queries Q_{hs} and $Q_{hs \cup cn}$. On the other hand, we also wanted to inspect the condition with the keyphrases only from `CN`, i.e., Q_{cn}. For each sample and condition, annotators have assigned a score for relevance to the hate speech in the scale of 1 to 5; 1 meaning no-relevance, and 5 perfect relevance. As a result, we have noticed that Q_{cn} is the worst condition, i.e., non-optimal, having an average score of 2.30. The analysis shows that it causes the loss of context related to `HS`, bringing information mostly from whole another topic. For instance, especially when `CNs` are rather generic, often, no lexical hint can be found related to the topic of Islamophobia (e.g., "*Do you have any proof?*"). Indeed, Q_{hs} provides an apparently better average score (3.46) since it provides a better context to search for. However, as expected, the best score (3.77) has been obtained through $Q_{hs \cup cn}$, i.e, optimal, verifying our hypothesis of utilizing both `HS` and `CN` keyphrases for training.

A.2 Preliminary Analysis on Decoding Methods for XNLG

To find a suitable decoding method for our task, we generated `CNs` with 3 candidate settings: beam search with a beam-width of 3 and top-k sampling with a k value of 8 and 10. For each setting we utilized `KN` retrieved with both non-optimal (Q_{cn}) and optimal ($Q_{hs \cup cn}$) queries. Then we sampled 120 `HS`-`CN` pairs and served them to three experts in `CN` writing for evaluating the generation on a scale of 1 (the worst) to 5 (the best) in terms of suitableness and informativeness. Suitableness measures if the generated `CN` is relevant to the `HS` and informativeness evaluates the amount of information (e.g. statistics and facts) enclosed in the `CN`.

The results reported in Table 8 reveal a clear difference between beam search and top-k sampling regardless of `KN` being optimal or non-optimal. In a manual investigation, we observed that the generation using both beam search and top-k sampling generally can copy some pieces of information from the given `KN`, while top-k seems to replace part of the text with slightly relevant and uncommon words. Hence, copying the right knowledge pieces through the decoding strategy is a key factor instead of diverging from the knowledge solely for the sake of lexical diversity. Therefore, based on the results, we adopt beam search with a beam-width of 3, which is shown to be the most suitable and informative, for decoding method in our experiments.

Decoding methods	Suit.	Info.
Non-optimal knowl.		
Beam-3	1.950	2.325
Topk-8	1.275	1.775
Topk-10	1.625	2.100
Optimal knowl.		
Beam-3	2.325	2.450
Topk-8	1.975	2.175
Topk-10	2.050	2.325

Table 8: Human evaluation on `CN` generation using various decoding methods.

A.3 CN Generation with Q_{hs}

In this section we report the `CN` generation results of our knowledge-bound models trained and tested with Q_{hs}. We applied the same hyperparameter configurations as the models trained with $Q_{hs \cup cn}$ described in Section 6.1.

The results are given in Table 9. In contrast to the baselines (i.e., models without knowledge and Candela), all models obtained higher novelty with Q_{hs}. The repetition rate, on the other hand, is not improved since the models exploit the same knowledge for multiple test samples due to the repeated `HSs` with different `CNs` in the test set.

We also observed that for GPT-2_{KN} and GPT-$2_{KN,MT}$ the generation with Q_{hs} is more repetitive and less novel compared to the generation applying queries $Q_{hs \cup gen}$ (as shown in Table 4). This result demonstrates the viability and necessity of using generated queries, as potential `CN` prompts, along with `HS` context.

A.4 Unique HS Test Set Analysis

Concerning that one `HS` can be paired with different `CNs` in the test, we further conducted an evaluation on a unique set by keeping each unique `HS` and one randomly selected `CN` among its `CNs`. The unique `HS` set lets us perform a fairer comparison among query configurations especially for Q_{hs}

							`KN` overlap (ngram)		
Models	Nov.	RR	B-2	R-L	#Word	#Sent.	1	2	3
GPT-2$_{KN}$	0.700	8.66	0.082	0.098	14.87	1.32	0.235	0.019	0.003
GPT-2$_{KN,MT}$	0.730	11.92	0.084	0.090	13.25	1.13	0.233	0.025	0.007
XNLG	0.824	16.46	0.073	0.084	55.51	3.71	0.841	0.650	0.558

Table 9: Results of `CN` generation applying Q_{hs}.

Models	Nov. (W/U)	RR (W/U)
without knowledge		
TRF	0.467/0.457	7.72/6.97
GPT-2	0.688/0.675	9.04/7.79
Train$_{cn}$	-	3.91/3.31
with knowledge		
Candela (Q_{hs})	0.692/0.697	21.87/21.97
GPT-2$_{KN}$		
w/ Q_{hs}	0.723/0.719	8.13/7.97
w/ Q_{gen}	0.728/0.720	7.48/6.34
w/ $Q_{hs\cup gen}$	0.735/0.740	6.30/5.81
w/ $Q_{hs\cup cn}$	0.727/0.731	7.17/6.16
GPT-2$_{KN,MT}$		
w/ Q_{hs}	0.744/0.748	11.69/10.24
w/ Q_{gen}	0.731/0.750	10.37/9.25
w/ $Q_{hs\cup gen}$	0.747/0.747	7.59/8.41
w/ $Q_{hs\cup cn}$	0.731/0.728	9.56/11.08
XNLG		
w/ Q_{hs}	**0.824/0.828**	14.42/6.22
w/ Q_{gen}	0.819/0.821	6.88/**4.03**
w/ $Q_{hs\cup gen}$	0.812/0.817	6.98/5.19
w/ $Q_{hs\cup cn}$	0.819/0.816	**5.69**/4.64

Table 10: Results of unique `HS` test set (U) for novelty and diversity. For comparison, we keep the results of whole test set (W).

with models employing beam search (i.e. XNLG and Candela). The results are given in Table 10.

For XNLG and GPT-2$_{KN,MT}$, we observed an increase in the novelty and diversity with Q_{hs} and Q_{gen} on the unique `HS` set over the whole test set. As for GPT-2$_{KN}$, while diversity improves for all query configurations, we did not observe an increase in novelty through Q_{hs}. For Candela, while novelty also increases, the diversity does not improve.

A.5 Human Evaluation Instruction

Three experts are tasked with `CN` generation evaluation. We briefed the annotators the aim of the evaluation, and then we explained common and appropriate `CN`s along with 10 examples of `HS`-`CN` pairs. The annotators are instructed to evaluate the generation with intuition. The generation is evaluated in terms of 3 aspects: (i) **suitableness:** How suitable the counter narrative is under evaluation for its hate speech, i.e. coherence and relevance of a counter narrative with regard to the hate speech. (ii) **informativeness:** How informative/specific the counter narrative is as a response for its hate speech. (iii) **intra-coherence:** How coherent the counter narrative is under evaluation as a response, i.e. internal coherence of a counter narrative regardless of its hate speech. In Table 11 we give the annotation examples with the score explanations provided to the annotators.

HS: Muslims do not want to assimilate. They demand that everyone shows respect for Islam, but they don't respect our laws. One day they will oppress us with the Sharia and polygamy.
CN: Do you really believe that they are a problem? I have a job. **evaluation**: suit.: 3, info.: 1, cohe.: 1 **Note**: The first sentence is relevant but not the second one. There is no coherence between 2 sentences.
CN: As far as they do not force us to follow sharia laws, we should not be concerned with them following their laws. **evaluation**: suit.: 5, info.: 3, cohe.: 5 **Note**: The `CN` is an appropriate and coherent response, but not providing a substantial amount of information or facts.
CN: Muslims should not be forced to assimilate, since it is not right and no one wants that. And polygamy is illegal and forbidden in UK and Muslims actually respect this ban. **evaluation**: suit.: 5, info.: 5, cohe.: 5 **Note**: The whole `CN` is perfectly suitable for the given `HS`. It is highly informative with the facts and internally coherent.

Table 11: Annotation examples with score explanations.

A.6 Generation Examples

HS: Jews are Christ Killers.
Human: While the Catholic Church has for decades taught that Jews weren’t collectively responsible, the argument laid out by the Pope Benedict XVI was a landmark statement. In his book, he not only confirms church teaching refuting the decide charge, he also explains biblically and theologically why there is no basis in Scripture for the argument that the Jewish people as a whole were responsible for Jesus’ death.
TRF: Why do you consider muslims like beasts? Did they do something wrong? If you take into consideration the crimes committed by muslims, why do you distinguish the muslim criminals from the non muslim? They are all criminals and all of them are punished.
GPT-2: I know many Muslims who are peaceful and respectful and do not kill.
Candela: Do you have proof? Where did you get this information from? I do not see the link between islamism and the news of the muslim community.
GPT-2$_{KN}$: No, the crucifixion of Jesus Christ is not a source for anti-Semitism. The Jewish people were not responsible for the death of Jesus.
GPT-2$_{KN,MT}$: On the basis of christianity, as christianity is not a religion, jews are not responsible for their own death.
XNLG: In ”jesus of nazareth-part ii” excerpts the church’s teaching that refuted the deicide charge of the jewish people, as well as theologically explains one of the most controversial issues in christianity.
KN: Pope Benedict XVI has made a sweeping exoneration of the Jewish people for the death of Jesus Christ. While the Catholic Church has for five decades taught that Jews weren’t collectively responsible, Jewish scholars said... would help fight anti-Semitism today. In ”Jesus of Nazareth-Part II” excerpts released Wednesday, Benedict explains biblically and theologically why there is no basis in Scripture for the argument that the Jewish people as a whole were responsible for Jesus’ death.

Table 12: Sample cross-domain `CN` generation from different models.

HS: Why gays do not try to heal?
Human: Contrary to some insinuations, sexual orientation is not like a disease.
TRF: Type imam boussena and we ll see who the scum is.
GPT-2: So they’re the ones who are trying to make us hate them.
Candela: You see more and more attacks on muslims as a result of the controversies? Do you really want to believe in a ‘caliphate’?
GPT-2$_{KN}$: We know that homosexuality is not a disease.
GPT-2$_{KN,MT}$: The genetics of homosexuality have been studied, but it seems to me that the human race is not suited to do that.
XNLG: And , contrary to the insinuation , sexual orientation is not like a disease . more recent studies have identified other potential links between homosexuality and genetics , and scientists have offered explanations on how same sex attraction can advance procreation , suggesting that the same twin is more likely to be gay , but finding the biological origins of the same
KN: Contrary to Perry’s insinuation, sexual orientation is not like a disease. More recent studies have identified other potential links between homosexuality and genetics, and scientists have offered explanations on how same-sex attraction could advance procreation. Studies of identical twins, who share the same genes, have shown that, if one twin is gay, the other is more likely to be gay, suggesting sexual orientation is influenced by genetics. Scientific studies have demonstrated that biology plays a role in one’s sexual orientation.

Table 13: Samples of cross-domain `CN` generation from different models.

HS: It seems that I cannot define Islam as a cult which promote wars and conquers host nations and legitimate this by its scriptures. Can I define them as narrow-minded, prejudiced, sexist and male chauvinist?
Human: Everyone has the right of free speech, while it does not mean anyone can use it to hurt people. Plus, based on what you critisize Islam? I recommend you to read something about true Islam and Muslims. You would find that they are no difference from us.
TRF: Islam is all about the religions and tolerance insert quote from koran. Migrants is not evil?
GPT-2: It seems to me that this is the antithesis of what Islam teaches.
KN Q_{hs}: Hakim al-Mutairi, secretary-general of the emirate's Umma (Islamic Nation) party, urged the pope to apologise immediately "to the Muslim world for his calumnies against the Prophet Muhammad and Islam".... Instead of bringing Islam and Christianity closer, he is straining relations between the two religions". In the present political atmosphere such views can be exploited by those who are trying to malign Muslims and Islam. **Candela:** I have no accountability. I have no problem with those who claim to belong to, but i do not have the right to practice whatever religion. **GPT-2$_{KN}$:** The Qur'an clearly teaches peace, love and tolerance towards everyone, including the non-Muslims. The only thing that Islam does not have is a message of conquest and war. **GPT-2$_{KN,MT}$:** Islam is a religion founded on the principle that you must believe in the good will of all people. **XNLG:** Extremist muslim national party secretary, hamid al mutairi, urged the pope to define immediately to the muslim world for his calumies against the prophet muhammad and islam. Pope benedict xvi was full of hostility and hatred against muslims and islam. In the present political atmosphere such views can be exploited by those who bring
KN $Q_{hs\cup gen}$: Islam, as practiced by the vast majority of people, is a peaceful religion, a religion that respects others. "The large majority of Muslims tinker," said Franck Fregosi, a sociologist who has written extensively on Islam in Europe.... **GPT-2$_{KN}$:** Hi, I do not understand the problem you have. The vast majority of Muslims do not advocate wars and conquer the world. They practice a peaceful religion that respects others. **GPT-2$_{KN,MT}$:** It seems that i have just come across the iota of muslims who have been insulted, persecuted, killed, raped, harassed, murdered, killed by the muslims in europe. **XNLG:** It seems that islam, as practiced by the vast majority of people, is a peaceful religion, a religion that respects others. The large majority of muslims tinker, as franck fregosi, who has written extensively on islam in europe. It seems that it does not define islam as an inclusion and respect for all of the religion
KN $Q_{hs\cup cn}$: No doubt, it is the legitimate right of Muslims to build a community center near Ground Zero. So please stop conflating criticism of Islam with discrimination of Muslims. As a nation of free people, we can never defeat Islam but we can make war against terrorists and governments that support them. a "moderate Islamic countries" when all unanimously reject the Universal Declaration of Human Rights and support The Cairo Declaration of Human Rights in Islam. **GPT-2$_{KN}$:** The Islamic countries do not promote wars and conquer host nations. Islam is not a cult but a peaceful religion which respects all peoples. **GPT-2$_{KN,MT}$:** The vast majority of men in the world are not able to articulate the spiritual values of islam. **XNLG:** It seems that it is the legitimate right of muslims to build a community center near zero ground, so please confine criticism of islam with discrimination against muslims. As a nation of free people, we can never defeat islam, but we can make war against terrorists and governments that support them. My comment gives a lot of information about what islam and muslims

Table 14: Sample of in-domain generation.

SOLID: A Large-Scale Semi-Supervised Dataset for Offensive Language Identification

Sara Rosenthal[1], **Pepa Atanasova**[2], **Georgi Karadzhov**[3],
Marcos Zampieri[4], **Preslav Nakov**[5]
[1]IBM Research, USA, [2]University of Copenhagen, Denmark,
[3]University of Cambridge, UK, [4]Rochester Institute of Technology, USA,
[5]Qatar Computing Research Institute, HBKU, Qatar
`sjrosenthal@us.ibm.com`

Abstract

The widespread use of offensive content in social media has led to an abundance of research in detecting language such as hate speech, cyberbullying, and cyber-aggression. Recent work presented the *OLID* dataset, which follows a taxonomy for offensive language identification that provides meaningful information for understanding the type and the target of offensive messages. However, it is limited in size and it might be biased towards offensive language as it was collected using keywords. In this work, we present *SOLID*, an expanded dataset, where the tweets were collected in a more principled manner. *SOLID* contains over nine million English tweets labeled in a semi-supervised fashion. We demonstrate that using *SOLID* along with *OLID* yields sizable performance gains on the *OLID* test set for two different models, especially for the lower levels of the taxonomy.

1 Introduction

Offensive language in social media has become a concern for governments, online communities, and social media platforms. Free speech is an important right, but moderation is needed in order to avoid unexpected serious repercussions. In fact, this is so serious that many countries have passed or are planning legislation that makes platforms responsible for their content, e.g., the *Online Harm Bill* (HM Government, 2019) in the UK and the *Digital Services Act* (European Commission, 2020) in the EU. Even in the United States, content moderation or the lack thereof can have significant impact on business (e.g., Parler was denied server space), government (U.S. Capitol Riots), and individuals (hate speech is linked to self-harm). Explainability is needed to indicate in detail why content has been deleted or flagged as inappropriate. Moreover, users can be educated by such feedback to avoid future biases.

There have been several areas of work in the detection of offensive language (Basile et al., 2019; Fortuna and Nunes, 2018; Ranasinghe and Zampieri, 2020), covering overlapping characteristics such as toxicity, hate speech, cyberbullying, and cyber-aggression. Further, using a hierarchical approach to analyze different aspects of the offensive content, such as the type and the target of the offense, helps provide explainability. The Offensive Language Identification Dataset, or *OLID*, (Zampieri et al., 2019a) is one such example, and it has been widely used in research. *OLID* contains 14,100 English tweets, which were manually annotated using a three-level taxonomy:

A: Offensive Language Detection
B: Categorization of Offensive Language
C: Offensive Language Target Identification

The taxonomy proposed in *OLID* makes it possible to represent different kinds of offensive content as a function of the *type* and the *target* of a post. For example, offensive messages targeting a group are likely hate speech, whereas offensive messages targeting an individual are likely cyberbullying. *OLID* has been used to annotate datasets in languages such as Arabic (Mubarak et al., 2021), and Greek (Pitenis et al., 2020), allowing for multilingual learning and analysis.

An inherent feature of the hierarchical annotation is that the lower levels of the taxonomy contain a subset of the instances in the higher levels, and thus there are fewer instances in the categories in each subsequent level. This makes it very difficult to train robust deep learning models on such datasets. Moreover, due to the natural infrequency of offensive language (e.g., less than 3% of the tweets are offensive when selected at random), obtaining offensive content is

---

WARNING: This paper contains tweet examples and words that are offensive in nature.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 915–928
August 1–6, 2021. ©2021 Association for Computational Linguistics

a costly and time-consuming effort. In this paper, we address these limitations by proposing a new dataset: **S**emi-Supervised **O**ffensive **L**anguage **I**dentification **D**atatset (*SOLID*)[1]. Our contributions are as follows:

1. We are the first to apply a semi-supervised method for collecting new offensive data using *OLID* as a seed dataset, thus avoiding the need for time-consuming annotation.
2. We create and publicly release *SOLID*, a training dataset containing 9 million English tweets for offensive language identification, the largest dataset for this task. *SOLID* is the official dataset of the SemEval shared task OffensEval 2020 (Zampieri et al., 2020).
3. We demonstrate sizeable improvements over prior work on the mid and lower levels of the taxonomy, where gold training data is scarce when training on *SOLID* and testing on *OLID*.
4. We provide a new larger test set and a comprehensive analysis of *EASY* (i.e., simple explicit tweets such as using curse words) and *HARD* (i.e., more implicit tweets that use underhanded comments or racial slurs) examples of offensive tweets.

The remainder of this paper is organized as follows: Section 2 presents related studies in aggression identification, bullying detection, and other related tasks. Section 3 describes the *OLID* dataset and annotation taxonomy. Section 4 introduces the computational models used in this study. Section 5 presents the *SOLID* dataset. Section 6 discusses the experimental results and Section 6.3 offers additional discussion and analysis. Finally, Section 7 concludes and discusses possible directions for future work.

2 Related Work

There have been several recent studies on offensive language detection and related tasks such as hate speech, cyberbulling, aggression, and toxic comment detection.

Hate speech identification is by far the most studied abusive language detection task (Ousidhoum et al., 2019; Chung et al., 2019; Mathew et al., 2021). One of the most widely used datasets is the one by Davidson et al. (2017), which contains over 24,000 English tweets labeled as non-offensive, hate speech, and profanity. A recent shared task on the topic is HatEval (Basile et al., 2019).

In cyberbullying detection, Xu et al. (2012) used sentiment analysis and topic models to identify relevant topics. Dadvar et al. (2013) and Safi Samghabadi et al. (2020) studied the use of the conversational context for detecting cyberbullying. In particular, Dadvar et al. (2013) used user-related features such as the frequency of profanity in previous messages. More recent work has addressed the issues of scalable and timely detection of cyberbullying in online social networks. To this end, Rafiq et al. (2018) employed a dynamic priority scheduler, and Yao et al. (2019) proposed a sequential hypothesis testing. Safi Samghabadi et al. (2020) constructed a dataset of cyberbullying episodes from the semi-anonymous social network ask.fm.

There were two editions of the TRAC shared task on Aggression Identification (Kumar et al., 2018, 2020) which provided participants with datasets containing annotated Facebook posts and comments in English and Hindi for training and validation. Facebook and Twitter datasets were used for testing. The goal was to discriminate between three classes: non-aggressive, covertly aggressive, and overly aggressive. Two other shared tasks addressed toxic language. The Toxic Comment Classification Challenge[2] at Kaggle provided participants with comments from Wikipedia annotated using six labels: toxic, severe toxic, obscene, threat, insult, and identity hate. The recent SemEval-2021 Toxic Spans Detection shared task addressed the identification of the token spans that made a post toxic (Pavlopoulos et al., 2021).

There were several shared tasks that have focused specifically on offensive language identification. For example, GermEval 2018 (Wiegand et al., 2018) which focused on offensive language identification in German tweets, HASOC 2019 (Mandl et al., 2019), and TRAC 2018 (Fortuna et al., 2018).

In this paper, we extend the prior work of the *OLID* dataset (Zampieri et al., 2019a). *OLID* is annotated using a hierarchical annotation schema as in (Basile et al., 2019; Mandl et al., 2019). In contrast to prior approaches, it takes both the target and the type of offensive content into account. This allows multiple types of offensive content (e.g., hate speech and cyberbullying) to be repre-

---

[1] Available at: `https://sites.google.com/site/offensevalsharedtask/solid`

[2] `http://kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

sented in *OLID*'s taxonomy. We create a large-scale semi-supervised dataset using the same annotation taxonomy as in *OLID*.

3 The OLID Dataset

The *OLID* (Zampieri et al., 2019a) dataset tackles the challenge of detecting offensive language using a labeling schema that classifies each example using the following three-level hierarchy:

Level A: ***Offensive Language Detection*** Is the text offensive?
OFF Inappropriate language, insults, or threats.
NOT Neither offensive, nor profane.

Level B: ***Categorization of Offensive Language*** Is the offensive text targeted?
TIN Targeted insult or threat towards a group or individual.
UNT Untargeted profanity or swearing.

Level C: ***Offensive Language Target Identification*** What is the target of the offense?
IND The target is an individual explicitly or implicitly mentioned in the conversation;
GRP Hate speech, targeting a group of people based on ethnicity, gender, sexual orientation, religion, or other common characteristic.
OTH Targets that do not fall into the previous categories (e.g., organizations, events, and issues.)

The taxonomy was successfully adopted for several languages (Mubarak et al., 2021; Pitenis et al., 2020; Sigurbergsson and Derczynski, 2020; Çöltekin, 2020), and it was used in a series of shared tasks (Zampieri et al., 2019b; Mandl et al., 2019). Tweets from the *OLID* dataset labeled with the taxonomy are shown in Table 1. The *OLID* dataset consists of 13,241 training and 860 test tweets.

Table 2 presents detailed statistics about the distribution of the labels. There is a substantial class imbalance on each level of annotation, especially at Level B. Furthermore, there is a sizable difference in the total number of annotations between the levels due to the schema (e.g., Level C is 1/3 smaller than Level A), and the data sizes for B and C are rather small. These drawbacks indicate the need to create a larger dataset.

4 Models

In this section, we describe the models used for semi-supervised annotation and for evaluating the contribution of *SOLID* for offensive language identification. We use a suite of heterogeneous machine learning models: PMI (Turney and Littman, 2003), FastText (Joulin et al., 2017), LSTM (Hochreiter and Schmidhuber, 1997), and BERT (Devlin et al., 2019). They have diverse inductive biases, which is an essential prerequisite for our semi-supervised setup (see Section 4.5). We assume that an ensemble of models with different inductive biases decreases each individual model's bias.

Tweet	**A**	**B**	**C**
@USER Anyone care what that dirtbag says?	OFF	TIN	IND
Poor sad liberals. No hope for them.	OFF	TIN	GRP
LMAO....YOU SUCK NFL	OFF	TIN	OTH
@USER What insanely ridiculous bullshit.	OFF	UNT	-
@USER you are also the king of taste	NOT	-	-

Table 1: Examples from the *OLID* dataset.

4.1 PMI

We use a PMI-based model that computes the n-gram-based similarity of a tweet to the tweets of a particular class c in the training dataset. The model is considered naïve as it accounts only for the n-gram frequencies in the discrete token space and only in the context of n neighboring tokens. We compute the PMI score (Turney and Littman, 2003) of each n-gram in the training set w.r.t. each class:

$$PMI(w_i, c_j) = log_2\left(\frac{p(w_i, c_j)}{p(w_i) * p(c_j)}\right) \quad (1)$$

where $p(w_i, c_j)$ is the frequency of n-gram w_i in instances of class c_j, $p(w_i)$ is the frequency of n-gram w_i in instances from the entire training dataset, and $p(c_j)$ is the frequency of class c_j. Additionally, we find that semantically oriented PMI scores (Turney and Littman, 2003) improve the performance of this naïve method:

$$PMI - SO(w_i, c_j) = log_2(\frac{p(w_i, c_j) * p(C \setminus \{c_j\})}{p(w_i, C \setminus \{c_j\}) * p(c_j)}) \quad (2)$$

where $C \setminus \{c_j\}$ is the set of all classes except c_j. At training time, we collect the frequencies of the n-grams on the training set. At inference time, we use the frequencies to calculate PMI and PMI-SO scores for each unigram and bigram in each instance and then average PMI and PMI-SO to a single score for each instance and class. Finally, we select the class with the highest score. If the instance contains no words with associated scores, we choose NOT for Level A, UNT for Level B –

Level	Label	*OLID* Train	*OLID* Test	*SOLID* Train	*SOLID* Test
A	OFF	4,640	240	1,448,861	3,002
	NOT	9,460	620	7,640,279	2,991
B	TIN	4,089	213	149,550	1,546
	UNT	551	27	39,424	1,451
C	IND	2,507	100	120,330	1,055
	GRP	1,152	78	22,176	349
	OTH	430	35	7,043	140

Table 2: Train and Test data distribution for the *OLID* and the *SOLID* datasets.

the classes most likely to contain neutral orientation, and the majority class IND for Level C. We remove words appearing less than five times in the training set and add a smoothing factor of 0.01 to all frequencies.

4.2 FastText

A suitable extension to the word-based model is to use subword representations to overcome the naturally noisy structure of tweets. FastText (FT) (Joulin et al., 2017) is a strong subword model which has shown strong performance on various tasks without the need for extensive hyperparameter tuning. It uses a shallow neural model for text classification similar to the continuous bag-of-words model (Mikolov et al., 2013). Instead of predicting the word based on its neighbors, it predicts the target label based on the sample's words. FT provides a valuable, diverse modeling representation to the ensemble due to its differences with the simple PMI model and the heavy-lifting LSTM and BERT models. We train FT with bigrams and a learning rate of 0.01 for Levels A and B and with trigrams and a learning rate of 0.09 for Level C. All tasks use a window size five and a hierarchical softmax loss.

4.3 LSTM

In contrast to the prior models, the LSTM model (Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017) can account for long-distance relations between words. First is an embedding layer initialized with a concatenation of the GloVe 300-dimensional (Pennington et al., 2014) and FastText's Common Crawl 300-dimensional embeddings (Grave et al., 2018). It is followed by a dropout and a bi-directional LSTM layer with an attention mechanism on top of it. We concatenate the attention mechanism's output with averaged and maximum global poolings on the outputs of the LSTM model. The final prediction is produced by a sigmoid layer for Levels A and B, where we have a binary classification, and a softmax layer for Level C, where we have three classes. We train the LSTM model using early stopping with patience for no improvements over the validation loss of up to five epochs. Level A uses a hidden size of 128, a dropout rate of 0.3, a batch size of 256, and a learning rate of 0.0002. Levels B and C use a hidden size of 50, a dropout rate of 0.1, a batch size of 32, and a learning rate of 0.0001. The Adam optimizer is used for training.

4.4 BERT

The Transformer architecture (Vaswani et al., 2017) has achieved (nearly) state-of-the-art performance for several NLP tasks. It displays both high representational power and robustness across tasks. We exploit the benefits of transfer learning in a low-resource setup by using the pre-trained BERT model (Devlin et al., 2019) and fine-tune it to our task. We use the base uncased model implementation from HuggingFace (Wolf et al., 2020), which has 12 layers, a hidden size of 768, and 12 attention heads, amounting to 110 million parameters. We fine-tune the BERT model for each task, starting from the pre-trained base model. We fine-tune BERT for 2, 3, and 3 epochs for Level A, B, and C, respectively. We use learning rates of 0.00002 for Levels A and B, and 0.00004 for Level C. We apply per-class weights to cope with the data imbalance in Level C as follows: IND=1, GRP=2, OTH=10. We use the Adam optimizer and a linear warm-up schedule with a 0.05 warm-up ratio.

4.5 Democratic Co-training

Democratic co-training (Zhou and Goldman, 2004) is a semi-supervised technique used to create large datasets with noisy labels when provided with a set of diverse models trained in a supervised way. This approach has been successfully applied in tasks like time series prediction with missing data (Mohamed et al., 2007), early prognosis of academic performance (Kostopoulos et al., 2019), and in the health domain (Longstaff et al., 2010). Using models with diverse inductive biases to label the target tweet can help ameliorate the individual model biases and produce predictions with a lower degree of noise.

In our work, we employ democratic co-training to create semi-supervised labels for all three levels of *SOLID* using *OLID* as our seed dataset. Distant supervision is conducted by the ensemble of models with different inductive biases as follows:

1. Train N *diverse* supervised models $\{M_j(X)\}$, where $j \in [1, N]$ on a labeled dataset $X = \{(x_i, y_i)\}$, where $i \in [1, |X|]$
2. For each example x'_i in the unannotated dataset $X' = \{(x'_i)\}, |i \in [1, |X'|])$ and each model M_j, predict the confidence p'^j_i for the positive class.

5 The *SOLID* Dataset

In this section, we describe the process of collecting a large dataset of over 12 million tweets All of the data was labeled using the democratic co-training approach described in the previous section. The statistics for the dataset are shown in Table 2.

5.1 A Large-Scale Dataset of Tweets

We collected our data from Twitter using the Twitter streaming API[3] via Twython[4] in 2019. We search the API using the 20 most common English stopwords (e.g. *the*, *of*, *and*, *to*) to ensure truly random tweets and avoid rate limits. Using stopwords ensures that we are more likely to obtain English tweets as well as a diverse set of random tweets. We kept the stream running the entire time and continuously choose a stopword at random based on its frequency in Project Gutenberg, a sizeable monolingual corpus. We collected 1,000 tweets for each stopword. Thus, tweets, including more frequent stopwords, are collected more frequently. A full list of the stopwords and their frequency is shown in Appendix A.1. We used this approach to help mitigate biases found in *OLID*. *OLID* was collected using a predefined list of keywords that were more likely to retrieve offensive tweets. This caused offensive tweets in *OLID* to be explicit and easier to classify. In contrast, the tweets collected in *SOLID* contain implicit and explicit offensive text. This allows us to study the performance of models in hard classification cases.

We used the `langdetect` tool[5] to select English tweets and discarded tweets with less than 18 characters or less than two words. We substituted all user mentions with `@USER` for anonymization purposes. We also ignored tweets with URLs as those don't tend to be offensive and might be less self-contained, e.g., they could have a link to an article, image, video, etc. Understanding such tweets would require going beyond their purely textual content. In total, we collected over 12 million tweets. We kept 9 million as training data, and we created a new test dataset from a portion of the remaining 3 million tweets.

Model	Level A	Level B	Level C
Majority Baseline	0.419	0.470	0.214
BERT	0.816	0.705	0.568
PMI	0.684	0.498	0.461
LSTM	0.681	0.657	0.585
FastText	0.662	0.470	0.590

Table 3: Macro-F1 score of the models in the democratic co-training ensemble on the *OLID* test set.

5.2 Semi-Supervised Training Dataset

We used the democratic co-training setup described in Subsection 4.5 to create the semi-supervised dataset. We first trained each model on the *OLID* dataset using 10% of the training dataset for validation. The performance of the individual models on the *OLID* dataset is shown in Table 3. BERT is the best model for Level A. The PMI model performs almost on par with the LSTM model. We expect this is due to the size of the dataset and the fact that a simple lexicon of curse words would be highly predictive of the offensive content present in a tweet. The performance of the FastText model is the lowest by 2 points. BERT performs best for Level B, followed by the LSTM model. The task is more challenging at this level for the frequency and n-gram-based approaches of PMI and FastText.

Finally, the overall performance of the models at Level C decreases further. This is expected as the size of the dataset becomes smaller, and the task is a three-way classification, whereas Levels A and B are two-way. BERT and LSTM outperform FastText and PMI, with BERT being the best model. The decrease in the performance in the final level can lead to increased noise in the semi-supervised labels, but we use an ensemble of four models, and we provide the average and the standard deviation of the confidence across the models on each instance to mitigate this. As we show later, these scores can be successfully used to filter out a large amount of noise in the semi-supervised dataset, thus yielding performance improvements.

We compute the aggregated single prediction based on the average and the standard deviation of the confidences predicted by each of the models:

---

[3] `https://developer.twitter.com/en/docs`

[4] `https://twython.readthedocs.io`

[5] `https://pypi.org/project/langdetect/`

Level	Text	BERT	LSTM	FT	PMI	AVG	STD	Label	E/H
A	@USER he fucking kills me. he knew it was coming	0.919	0.958	0.852	0.509	0.809	0.177	OFF	E
	His kissing days are over, he's a pelican now!	0.659	0.304	0.568	0.523	0.514	0.131	NOT	H
	i think we're all in love with winona ryder	0.060	0.038	0.017	0.480	0.102	0.155	NOT	E
B	Guess I'll just never understand the fucking dynamics	0.901	0.569	0.001	0.617	0.522	0.327	UNT	H
	@USER Government is a bunch of bitches.	0.013	0.221	0.000	0.397	0.158	0.164	TIN	E
	@USER Give me the date. Fuck them other niggas Bro I'm irritated as fuck	0.882	0.666	0.983	0.701	0.808	0.131	TIN	E
C	@USER He was useless stupid guy	0.807	0.915	1.000	0.480	0.801	0.197	IND	E
	It's like mass shootings is the reg in this shit hole country!	0.826	0.479	0.693	0.570	0.642	0.131	OTH	H
	Getting these niggas tatted is a overstatement are ya dead serious	0.700	0.691	0.770	0.491	0.663	0.104	GRP	H

Table 4: Training data aggregation examples. Columns 3-6 show the confidence of each of the models with respect to the positive class in Levels A and B (OFF, UNT) and only for the corresponding class in C (one example for each of the classes: TIN, GRP, OTH). The label column shows manual annotations, and the last column shows whether the tweet is considered Easy (E) or Hard (H) based on its AVG confidence.

$SOLID = \{(x'_i, p'_i)|i \in [1, |SOLID|])\}$, where p'_i = avg$(\{p'^{j}_i | j \in [1, N]\})$, std$(\{p'^{j}_i | j \in [1, N]\})$. In particular, we compute the scores based on the confidences for the positive class at Levels A and B and the confidences for the IND, GRP, and OTH classes at Level C. We performed the above aggregation step instead of just using the scores of each model to avoid over-fitting to any particular model in the ensemble. This helps to prevent biases on individual models in future uses of the dataset. Further, the standard deviation and the average scores can be used to filter instances that the models disagree on, thus reducing potential noise in the semi-supervised annotations.

The dataset is labeled using the semi-supervised manner by assigning a Level A label to all the tweets. Then, we select the subset of tweets that are likely to be offensive for all models (BERT and LSTM $\geq .5$, PMI and FT=OFF) as instances that should be assigned a label for Level B. We chose the tweets likely to be TIN at Level B with a standard deviation lower than 0.25 for Level C. Thus, only the instances that are most likely to be offensive are considered at Levels B and C, and only those that are most likely to be offensive and targeted are considered at Level C. The size and the label distribution across the datasets can be found in Table 2 and examples of tweets along with models' prediction confidences can be found in Table 4.

5.3 *SOLID* Test Dataset

The *OLID* test set is very small, particularly for Levels B and C. Therefore, we also annotated a portion of our held-out 3 million tweets to create a new *SOLID* test set to obtain more stable results and to analyze the performance in more detail.

First, all co-authors of this paper (five annotators) annotated 48 tweets that were predicted to be OFF in order to measure inter-annotator agreement (IAA) using $P_0 = \frac{agreement\_per\_annotation}{total\_annotations*num\_annotators}$. We found IAA to be *0.988* for Level A; an almost perfect agreement for OFF/NOT. The IAA for Level B was *0.818*, indicating a good agreement on whether the offensive tweet was TIN/UNT. Finally, for Level C, the IAA was *0.630*, which is lower but still considered reasonable as Level C is more complicated due to its 3-way annotation schema: IND/GRP/OTH. In addition, a tweet may address targets of different types (e.g., both an individual and a group), but only one label can be chosen.

After having observed high IAA, we annotated additional offensive tweets with a single annotation per instance. We divided our Level A data into four portions based on model confidence:

- *if* BERT $\geq .8 \wedge$ PMI = OFF $\wedge$ FT = OFF $\wedge$ LSTM $\geq .8$ *then* ***Easy*** **OFF** [2380 tweets]
- *else if* BERT $\geq .5 \wedge$ PMI = OFF $\wedge$ FT = OFF $\wedge$ LSTM $\geq .5$ *then* ***Hard*** **OFF** [835 tweets]
- *else if* BERT $\leq .2 \wedge$ PMI = NOT $\wedge$ FT = NOT $\wedge$ LSTM $\leq .8$ *then* ***Easy*** **NOT** [2500 tweets]
- *else if* BERT $< .5 \wedge$ PMI = NOT $\wedge$ FT = NOT $\wedge$ LSTM $< .5$ *then* ***Hard*** **NOT** [278 tweets]

Note, PMI = OFF and FT = OFF designates that the model's probability is higher for the OFF class than for the NOT class. We selected the rest of the thresholds after a manual examination of the confidence scores for each model. We chose the threshold where the model is confident and mostly correct.

We annotated 3,493 tweets for Level A. The

#	Type	Prediction	Tweet	Gold Label
1	Easy	OFF	this job got me all the way fucked up real shit	OFF UNT
2	Easy	OFF	@USER It's such a pain in the ass	OFF UNT
3	Easy	OFF	wtf ari her ass tooo big	OFF TIN IND
4	Easy	NOT	This account owner asks for people to think rationally.	NOT
5	Hard	OFF	It sucks feeling so alone in a world full of people	NOT
6	Hard	OFF	@USER We are a country of morons	OFF TIN GRP
7	Hard	NOT	Hate the sin not the sinner...	NOT
8	Hard	NOT	Somebody come get her she's dancing like a stripper	OFF TIN IND

Table 5: Example tweets from the *SOLID Test* dataset and its four subsets. Shown are the difficulty of each subset (Type), the ensemble model prediction for the examples in each subset (Prediction), an example tweet's text, and the manually annotated gold label.

Type	Model Prediction	Gold Label OFF	Gold Label NOT	Total
easy	OFF	2,187	193	2,380
easy	NOT	0	2,500	2,500
hard	OFF	670	165	835
hard	NOT	145	133	278
Total		**3,002**	**2,991**	**5,993**

Table 6: Statistics of the *SOLID Test* dataset grouped by difficulty (Type) and model prediction.

number of annotations at each level is shown above in square brackets. Furthermore, to create a complete test dataset for Level A (where we only labeled offensive tweets), we also took a random set of 2,500 *Easy* NOT tweets. The resulting test sizes are shown in Table 2. Of the 3,493 annotated tweets, 491 were determined to be NOT. In total, there are 5,993 tweets in our test set. In all cases, all three levels were annotated, but the decision of whether a tweet in Level B/C is *Easy* or *Hard* is still based on its Level A confidence.

Table 5 shows some tweets and whether they are *Easy* OFF/NOT (lines 1-4) or *Hard* OFF/NOT (lines 5-8), and Table 6 shows statistics regarding the Easy and Hard examples in the test dataset. Note that determining the labels for the *Hard* examples is not simple and the model does make incorrect predictions such as in lines 5 and 8 of Table 5. In fact, 25% of the *Hard* OFF tweets that we annotated were NOT. In contrast, 8% of the *Easy* OFF tweets were judged to be NOT.

6 Experiments and Evaluation

In this section we describe our experiments and results when training with *OLID* + *SOLID* data compared to just *OLID* on the *OLID* test set.

6.1 Experimental Setup

We used the BERT and FastText models from the semi-supervised annotation set up to estimate the performance improvements when training on the supervised dataset *OLID* together with the semi-supervised *SOLID*. The models in all sets of experiments were fine-tuned on a 10% validation split of the training set used during co-training. We explored different schemes to combine *OLID* and *SOLID*, as well as different thresholds for the confidence of the instances in *SOLID*. We achieved improvements for Levels B and C by upsampling the underrepresented classes: we sampled K instances of each class, where K is the number of instances for the most frequent class. We also removed the warm-up in Levels B and C, which improved the results further.

FastText. The FastText model used an external command-line tool without control over the training. Therefore, we merged the training splits of *OLID* and *SOLID*, randomly shuffled them, and trained models with the combined dataset. The FastText model has the same parameters used above in co-training.

BERT. Due to the computational requirements of BERT, we subsampled 20,000 tweets from *SOLID* in Level A and B for BERT. Including more semi-supervised instances did not improve the performance. During training, we used *SOLID* in the first epoch and *OLID* in the following two epochs for Level A. Using *SOLID* after training with *OLID* yielded worse results. We assume this can be explained by the fact that the semi-supervised dataset by construction contains labels that are not golden truth. It can be used as an initial step to guide the model towards a better local minimum. On the other hand, we conjecture that the supervised

Level	Baseline	BERT		FastText	
		OLID	*+SOLID*	*OLID*	*+SOLID*
A	0.419	**0.816**	0.809	0.662	**0.720**
B	0.470	0.687	**0.729**	0.470	**0.591**
C	0.214	0.589	**0.643**	**0.590**	0.515

Table 7: Macro-F1 score on the *OLID* test set for BERT and FastText with and without training on *SOLID* compared to the majority class baseline.

	Model	Baseline	BERT		FastText	
			OLID	*+SOLID*	*OLID*	*+SOLID*
A	Full	0.338	0.922	**0.923**	0.856	**0.860**
	Easy	0.400	0.983	**0.983**	0.936	**0.940**
	Hard	0.444	0.557	**0.570**	0.525	**0.536**
B	Full	0.236	0.559	**0.666**	0.355	**0.493**
	Easy	0.232	0.569	**0.677**	0.349	**0.509**
	Hard	0.234	0.542	**0.649**	0.363	**0.467**
C	Full	0.203	0.627	**0.645**	0.387	**0.504**
	Easy	0.201	0.635	**0.644**	0.378	**0.504**
	Hard	0.205	0.616	**0.649**	0.397	**0.505**

Table 8: Experimental results (macro-F1 scores) on the *SOLID Test* dataset, and on its Easy and Hard subsets, compared to the majority class baseline.

dataset is better suited for fine-tuning the model towards the local minimum with the gold data, particularly in Level A, where the training split of *OLID* is already sufficient for training BERT. For Levels B and C, we trained for two epochs with the training split of *OLID* and then for one epoch with *SOLID*. At Levels B and C, we observed that training with *SOLID* in the first epochs and then fine-tuning with *OLID* did not improve the performance. Furthermore, training with *OLID* and then using *SOLID* for the final epochs yielded substantial performance improvements. We assume this is due to the small training size of *OLID* which can cause the model to overfit to a suboptimal local minimum when used in the final training epochs.

Selecting *SOLID* Instances. We filter the training instances from *SOLID* to be the most confident examples based on the average probability score provided in *SOLID* when training with FastText and BERT. We choose the threshold for the average confidence score based on the validation dataset as follows:

Level A: $avg(\text{OFF}) < 0.20 \cup avg(\text{OFF}) > 0.70$
Level B: $avg(\text{UNT}) < 0.35 \cup avg(\text{UNT}) > 0.65$
Level C: $avg(\text{IND}) > 0.80 \cup avg(\text{GRP}) > 0.70 \cup avg(\text{OTH}) > 0.65$

To select a label for each instance, we choose: NOT when $avg(\text{OFF}) < 0.20$, otherwise OFF in Level A; UNT when $avg(\text{UNT}) > 0.65$, otherwise TIN in Level B; the class with the highest probability in Level C.

6.2 *OLID* Results

In this section we describe our results on the *OLID* test set using just *OLID* and adding *SOLID*. The results are shown in Table 7.

The results for Level A are improved only with FastText, which is a weaker model (see Table 3). It leverages a large performance improvement when trained with *OLID+SOLID*. On the other hand, the BERT model already achieves high performance without augmenting *OLID* with *SOLID* because *OLID* is large enough for Level A. As a result, including semi-supervised data did not improve the performance. Our findings are in line with the study of Longstaff et al. (2010), who observed that democratic co-training performs well when the initial classifier's accuracy was low.

The *OLID* training dataset is smaller for Level B, and the task is more complex. Moreover, the FastText model here performs on par with the majority class baseline. Augmenting *OLID* with *SOLID* yields performance improvements for both models. We achieve an improvement of 0.042 points for BERT and a large margin of improvement of 0.121 points for FastText.

Finally, in Level C, the supervised *OLID* data is even smaller, and the complexity of the subtask is more pronounced, mainly due to it having three possible labels. Interestingly, using *SOLID* for FastText does not yield better results. This might be due to the model already achieving high performance on par with BERT (see Table 3), while democratic co-training performs well when the initial classifier's performance is low. Additionally, this may be due to the instability of the test set for Level C, which is very small. On the other hand, the *SOLID* data helps the BERT model by a large margin of 0.054 points.

6.3 *SOLID* Results

In the previous section, we showed noticeable improvements on the *OLID* dataset using *SOLID*. However, *OLID* is small (particularly for Levels B and C). Showcasing the performance on a larger test set, *SOLID* test, is important for estimating the models' stability. We also focus on *Easy* vs. *Hard* examples (based on the confidence computed during co-training) to gain better insight into why some tweets are easier to predict as offensive than others. Results are shown in Table 8 and significantly beat the majority baselines.

The overall Level A results on *SOLID* test are 92.3% and 86.0% macro-F1 for BERT and FastText, respectively, with a small improvement when *OLID* is augmented with *SOLID* for FastText only. This is consistent with what we found on the *OLID* test set. Note that the full results for Level A are much better than on the OLID dataset in Table 7. We expect that this is partially due to our selection of tweets for the new test set, indicating that there are more *Easy* tweets in it. Similar findings to the full test set occur with the *Easy* tweets, but the scores are even higher. On the other hand, for the *Hard* tweets, the results are much lower at 57% and 53.6% for BERT and FastText, respectively. Using *SOLID* yields a nice improvement for both models on the *Hard* tweets, which was not evident in the *OLID* test set in Table 7.

To provide further insight into why the results are so high for *Easy* OFF tweets in Level A, we implemented a curse word baseline using the absence or presence of 22 curse words like *fuck*, *bitch*, and *nigga*. A full list of the curse words used in the baseline can be found in Appendix A.1. We found that most *Easy* tweets were classified correctly with this baseline with 93.6% F1-score. In contrast, the curse word baseline was not effective on the hard examples, just like the BERT and FastText models. It achieved a macro-F1 score of 58%, which is one point higher than the BERT result. The BERT and FastText models are clearly overfitting to the curse words. The *hard* tweets are offensive due to other language use such as negative biases rather than the appearance of a curse word such as examples 6 and 8 in Table 5. Classifying these tweets successfully remains an open challenge.

The difference between *Easy* OFF/NOT and *Hard* OFF/NOT tweets is less pronounced for Levels B and C. The curse word imbalance may have a small impact on the lower levels as UNT tweets are more likely to contain curse words. In all cases, including *SOLID* with *OLID* for Levels B and C yields a nice improvement, indicating that the larger test set can better showcase the improvements, leading to more stability. The results for Levels B and C vary greatly for the two models compared to those on the *OLID* test set in Table 7, which points to the challenges of having a very small test set.

7 Conclusion and Future Work

We presented *SOLID*, a large-scale semi-supervised training dataset for offensive language identification, which we created using an ensemble of four different models. To the best of our knowledge, *SOLID* is the largest dataset of its kind, containing nine million English tweets. We have shown that using *SOLID* yields noticeable performance improvements for Levels B and C of the OLID annotation schema, as measured on the *OLID* test set. Furthermore, in contrast to using keywords, our approach allows us to distinguish between *Hard* and *Easy* offensive tweets. The latter enables us to have a deeper understanding of offensive language identification and indicates that detecting *Hard* offensive tweets is still an open challenge. Our work encourages safe and positive places on the web that are free of offensive content, especially non-obvious cases (i.e., *Hard*). *SOLID* is the official dataset of the SemEval shared task OffensEval 2020 (Zampieri et al., 2020). In the future, we would like to provide insights and methods for categorizing *Hard* tweets.

Acknowledgements

We would like to thank the anonymous reviewers who provided us with constructive and insightful feedback to further improve the quality of the paper. PA has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 801199. This work is also part of the Tanbih mega-project, developed at the Qatar Computing Research Institute, HBKU, which aims to limit the impact of "fake news," propaganda, and media bias by making users aware of what they are reading.

Ethics Statement

Dataset Collection The *OLID* and the *SOLID* datasets were collected using the Twitter API[6]. The *OLID* dataset was collected using keywords that would be more likely to be accompanied with offensive tweets (Zampieri et al., 2019a). The *SOLID* dataset was collected using frequent stop words (See Section 5) intending to aid in detecting offensive tweets in English only, We follow the terms of use outlined by Twitter[7]. Specifically, we only

---

[6] `http://developer.twitter.com/en/docs`

[7] `http://developer.twitter.com/en/developer-terms/agreement-and-policy`

download public tweets and we provide only the user ids of the tweets to ensure that deleted tweets will no longer be available in our dataset. Further, in all our examples in this paper, we anonymize the user names in the tweets. Since no private information is stored, IRB approval is not required. All annotations were performed internally by the authors of the paper.

Biases *SOLID* is a large-scale semi-supervised dataset for offensive language detection We note that determining whether a piece of text is offensive can be subjective, and thus it is inevitable that there would be biases in our gold-labeled data. It is expected that these biases will, therefore, also be present in the semi-supervised dataset we generated from such tweets.

While we cannot ensure that no biases occur in the gold data, we address these concerns by following a well-defined schema, which sets explicit definitions for offensive content during annotation. Our high inter-annotator agreement makes us confident that the assignment of the schema to the data is correct most of the time.

Using semi-supervised techniques to create a large dataset, *SOLID*, can cause the biases found in the gold data to be expanded further. We mitigate this in two ways. First, we gather tweets based on the most frequent words in English to ensure a random set of initial tweets. Next, we construct an ensemble of models with diverse inductive biases to label the target tweet, which can help to ameliorate the individual model biases and to produce predictions with a lower degree of noise. At test time, we aim to have a meaningful ratio of offensive and non-offensive tweets based on a random collection of tweets. We also label all test offensive tweets manually. The aim of these steps was to help reduce the potential biases. Please refer to Section A.2 of the Appendix for some analysis that indicates the diversity of the models.

We acknowledge that current semi-supervised techniques do not address the problem of the bias inherent in the semi-supervised data coming from the supervised source model(s), which can also be studied in future work. Further, we acknowledge that biases can still exist in the ratio of offensive/non-offensive tweets. The size of the data and the method of collection for the SOLID dataset mean that biases are hard to avoid.

In addition, offensive language can vary depending on demographics, such as the gender of the targeted individual and the target can even be a particular gender group. Such biases that are present in natural language data (Olteanu et al., 2019) is an attractive future study.

Misuse Potential Most datasets compiled from social media present some risk of misuse. We therefore ask researchers to be aware that the *SOLID* dataset can be maliciously used to unfairly moderate text (e.g., a tweet) that may not be offensive based on biases that may or may not be related to demographics and other information within the text. Intervention with human moderation would be required in order to ensure this does not occur.

Intended Use We present *SOLID* to encourage research in automatically detecting and stopping offensive content from being disseminated on the web. Such systems can be used to alleviate the burden for media moderators, which can suffer from psychological disorders due to the exposure of extremely offensive content. Improving the performance of offensive content detection systems can decrease the amount of work for moderators, but human supervision is required for more intricate cases and to ensure that the system is not causing harm. With the possible ramifications of a highly subjective dataset, we distribute *SOLID* for research purposes only, without a license for commercial use. Any biases found in the dataset are unintentional, and we do not intend to cause harm to any group or individual.

We believe that this dataset is a useful resource when used in the appropriate manner with great potential to improve the performance of current offensive content detection systems.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Yi-Ling Chung, Elizaveta Kuzmenko, Serra Sinem Tekiroglu, and Marco Guerini. 2019. CONAN -

COunter NArratives through nichesourcing: a multilingual dataset of responses to fight online hate speech. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2829, Florence, Italy. Association for Computational Linguistics.

Çağrı Çöltekin. 2020. A corpus of Turkish offensive language on social media. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6174–6184, Marseille, France. European Language Resources Association.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, page 693–696, Berlin, Heidelberg. Springer-Verlag.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

European Commission. 2020. Proposal for a regulation of the european parliament and of the council on a single market for digital services (digital services act) and amending directive 2000/31/ec. `https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52020PC0825`.

Paula Fortuna, José Ferreira, Luiz Pires, Guilherme Routar, and Sérgio Nunes. 2018. Merging datasets for aggressive text identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 128–139, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4).

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

HM Government. 2019. Online harms white paper. `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/793360/Online_Harms_White_Paper.pdf`.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Georgios Kostopoulos, Stamatis Karlos, and Sotiris Kotsiantis. 2019. Multiview learning for early prognosis of academic performance: A case study. *IEEE Transactions on Learning Technologies*, 12(2):212–224.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2020. Evaluating aggression identification in social media. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 1–5, Marseille, France. European Language Resources Association (ELRA).

Brent Longstaff, Sasank Reddy, and Deborah Estrin. 2010. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–7.

Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17, New York, NY, USA. Association for Computing Machinery.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. In *Proceedings of AAAI*.

Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, pages 1–12.

Tawfik A. Mohamed, Neamat El Gayar, and Amir F. Atiya. 2007. A co-training approach for time series prediction with missing data. In *Multiple Classifier Systems*, pages 93–102, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hamdy Mubarak, Ammar Rashed, Kareem Darwish, Younes Samih, and Ahmed Abdelali. 2021. Arabic offensive language on Twitter: Analysis and experiments. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 126–135, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Alexandra Olteanu, C. Castillo, Fernando D. Diaz, and E. Kiciman. 2019. Social data: Biases, methodological pitfalls, and ethical boundaries. *Social Science Research Network*, 2:13.

Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song, and Dit-Yan Yeung. 2019. Multilingual and multi-aspect hate speech analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4675–4684, Hong Kong, China. Association for Computational Linguistics.

John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of SemEval*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Zesis Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive language identification in Greek. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5113–5119, Marseille, France. European Language Resources Association.

Rahat Ibn Rafiq, Homa Hosseinmardi, Richard Han, Qin Lv, and Shivakant Mishra. 2018. Scalable and Timely Detection of Cyberbullying in Online Social Networks. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1738–1747, New York, NY, USA. Association for Computing Machinery.

Tharindu Ranasinghe and Marcos Zampieri. 2020. Multilingual offensive language identification with cross-lingual embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5838–5844, Online. Association for Computational Linguistics.

Niloofar Safi Samghabadi, Adrián Pastor López Monroy, and Thamar Solorio. 2020. Detecting early signs of cyberbullying in social media. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 144–149, Marseille, France. European Language Resources Association (ELRA).

Gudbjartur Ingi Sigurbergsson and Leon Derczynski. 2020. Offensive language and hate speech detection for Danish. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3498–3508, Marseille, France. European Language Resources Association.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval 2018 Workshop (GermEval)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666, Montréal, Canada. Association for Computational Linguistics.

Mengfan Yao, Charalampos Chelmis, and Daphney Stavroula Zois. 2019. Cyberbullying ends here: Towards robust detection of cyberbullying in social media. In *The World Wide Web Conference*, WWW '19, page 3427–3433, New York, NY, USA. Association for Computing Machinery.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 1415–1420, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

Yan Zhou and Sally Goldman. 2004. Democratic co-learning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '04, page 594–202, USA. IEEE Computer Society.

A Appendices

A.1 Data Collection and Analysis

Section 5.1 we described our method for collecting tweets. We collect tweets using the most frequent English words based on the large monolingual Project Gutenberg corpus.[8] Table 9 shows the top-20 most frequent words in the corpus and their frequency which we used to collect tweets. The normalized value is the percentage of the total frequency for all top 20 words. We randomly pick a number between 0 and 1, and choose the word based on the normalized value. For example, .45 would be "and".

In Section 6.3, we discussed the simple curse word baseline used to analyze the *Easy* OFF/NOT tweets. Table 10 lists the 22 curse words used in the baseline.

A.2 Implementation Details

The fine-tuning of the models was performed on a 10% split from the *OLID* dataset. All models were trained on an NVIDIA Titan X GPU with 8GB of RAM.

w	**frequency**	**norm.**	**w**	**frequency**	**norm.**
the	56,271,872	0.20	it	8,058,110	0.79
of	33,950,064	0.32	with	7,725,512	0.82
and	29,944,184	0.43	is	7,557,477	0.85
to	25,956,096	0.52	for	7,097,981	0.87
in	17,420,636	0.58	as	7,037,543	0.90
i	11,764,797	0.63	had	6,139,336	0.92
that	11,073,318	0.67	you	6,048,903	0.94
was	10,078,245	0.70	not	5,741,803	0.96
his	8,799,755	0.73	be	5,662,527	0.98
he	8,397,205	0.76	her	5,202,501	1.00

Table 9: The top-20 most frequent English words (*w*). *Norm.* is the normalized value based on the total frequency of all the top words. The random number generated between 0 and 1 determines which word is chosen.

ass	arse	wtf	lmao	fuck
bitch	nigga	nigger	cunt	effing
shit	hell	damn	crap	bastard
idiot	stupid	racist	dumb	f*ck
pussy	dick			

Table 10: The 22 common offensive terms used in the curse word baseline.

The evaluation metric used for all experiments is macro F1 from scikit-learn.[9]. The performance of the models in the ensemble used for semi-supervised labelling is provided in Table 11.

Model	**A**	**B**	**C**
BERT	0.788	0.610	0.577
PMI	0.772	0.595	0.536
LSTM	0.599	0.599	0.579
FastText	0.672	0.489	0.456

Table 11: F1 score performance of each model used in the ensemble on the validation dataset of Levels A, B, and C.

In Table 12 we show the agreement of the models for the task prediction. For Levels A and B, it is more common that all four models agree, while in Level C, there are more cases when at least one model disagrees with the rest models. Furthermore, in Level A, there are almost no cases when the decision is tied with two models disagreeing with the other two. Finally, as in Level C, the performance of the models is lower, the disagreement between the models in the ensemble is the largest and it is least common for all four models to agree on

---

[8] `https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists#Project_Gutenberg`

[9] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

a prediction. Given the observed agreement rates, we conclude that there is considerable variance in the predictions across the models, especially for the lower levels. We assume this indicates that the separate models have different rationales to a certain degree, which can be avoided by the ensemble combination of the models.

N	**A**	**B**	**C**
4	0.517	0.598	0.249
3	0.392	0.275	0.417
2	0.091	0.127	0.335

Table 12: Percentage of instances where N models agree for a predicted label of an instance, N $\in \{2, 3, 4\}$, for Levels A, B, and C.

RealFormer: Transformer Likes Residual Attention

Ruining He, Anirudh Ravula, Bhargav Kanagal, Joshua Ainslie
Google Research
{ruininghe,braineater,bhargav,jainslie}@google.com

Abstract

Transformer is the backbone of modern NLP models. In this paper, we propose **RealFormer**, a simple and generic technique to create **Re**sidual **A**ttention **L**ayer Trans**former** networks that significantly outperform the canonical Transformer and its variants (BERT, ETC, etc.) on a wide spectrum of tasks including Masked Language Modeling, GLUE, SQuAD, Neural Machine Translation, WikiHop, HotpotQA, Natural Questions, and OpenKP. We also observe empirically that RealFormer stabilizes training and leads to models with *sparser* attention. Source code and pre-trained checkpoints for RealFormer can be found at `https://github.com/google-research/google-research/tree/master/realformer`.

1 Introduction

Transformer (Vaswani et al., 2017) architectures are the backbone of numerous state-of-the-art NLP models such as BERT (Devlin et al., 2019), GPT (Radford et al., 2019), and Meena (Adiwardana et al., 2020), and have seen wide success across both academia and industry. Typically, a Transformer network consists of a stack of residual layers. The original design follows a "Post-LN" structure which adds Layer Norm (LN) as a "post-processing" step for each sub-layer, as shown in Figure 1 (a). It has been adopted by various state-of-the-art models including BERT, XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), Transformer-XL (Dai et al., 2019), and ETC (Ainslie et al., 2020). Another notable design is to reorganize the order of modules to create a "direct"/clean path to propagate embeddings of tokens in the input sequence through the whole network, as shown in Figure 1 (b).[1] This design adds LN as a "pre-processing" step for each sub-layer, and is often referred to as "Pre-LN" and used by some well-known extra large models such as GPT-2 (Radford et al., 2019) and Megatron (Shoeybi et al., 2019). In some respect, Post-LN and Pre-LN are analogous to ResNet v1 (He et al., 2016a) and ResNet v2 (He et al., 2016b) respectively in the Computer Vision literature. Although ResNet v2 is usually preferable to v1 for Computer Vision, it does not appear to be the case for Pre-LN Transformer in the NLP literature. It is likely that the particularities of self-attention modules and Transformer architectures potentially favor (at least slightly) different designs compared to traditional convolutional neural networks.

In this paper, we propose a simple and generic technique to show that it is beneficial to create a "direct" path to propagate raw attention scores through Transformer-based networks. Our technique is called ***Re**sidual **A**ttention **L**ayer Trans**former***, or ***RealFormer*** in short. We also use *RealFormer* to denote the resulting Transformer networks whenever no confusion may arise. Without losing generality, taking the standard Transformer encoder as an example, each RealFormer layer takes the raw attention scores of all attention heads from the previous layer and adds "residual scores" (computed the same way as attention scores in regular Transformers) on top, as shown in Figure 1 (c). The sum of the two scores is then used to compute attention probabilities via softmax.

In other words, RealFormer can be seen as adding simple skip connections to a backbone Transformer. Since it does not add expensive multiplication ops, performance is expected to be comparable.[2] Note that our technique can also be applied straightforwardly for different Transformer varia-

---

[1] Note that a final LN module is usually added at the very top of the whole network.

[2] In certain settings, we find it helpful for RealFormer to use running *mean* of attention scores instead of running *sum*, though it adds some negligible amount of multiplications.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 929–943
August 1–6, 2021. ©2021 Association for Computational Linguistics

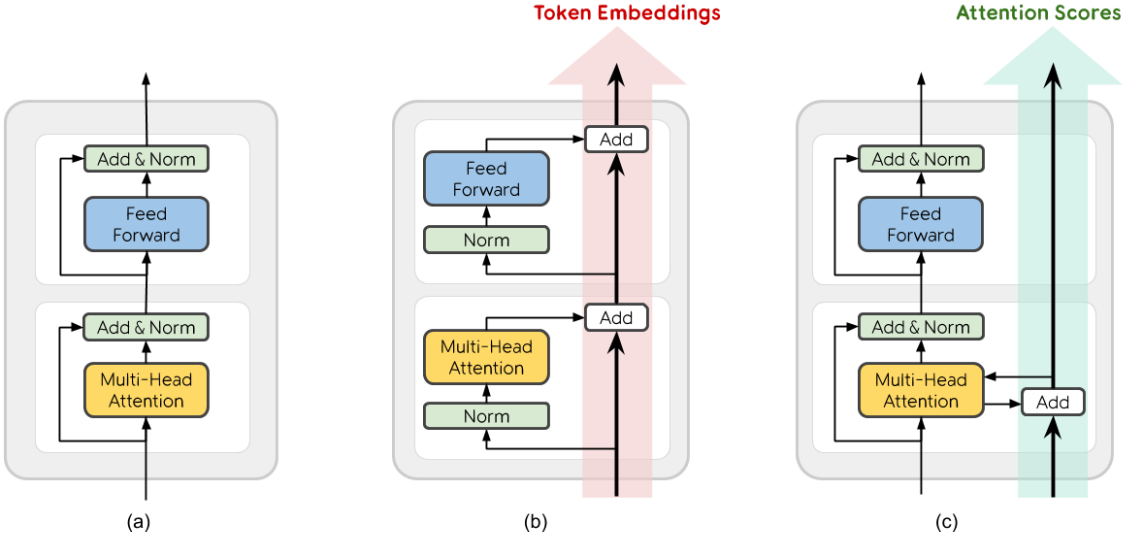

Figure 1: Comparison of different styles of Transformer layers: (a) The prevalent Post-LN layer used by (*e.g.*) BERT; (b) Pre-LN layer used by (*e.g.*) GPT-2 that creates a "direct" path to propagate token embeddings; (c) Our RealFormer layer that creates a "direct" path to propagate attention scores (by adding a simple skip edge on top of (a)). Note that here we are showing Transformer encoder for demonstration purposes only; RealFormer can be applied straightforwardly for different Transformer variations (*e.g.*, when decoders are involved).

tions and even when decoders are involved.

Specifically, our main contributions include:

- We present RealFormer, a simple, generic, and cheap technique to improve Transformer-based networks. It adds no parameters or hyper-parameters, and usually takes no more than a few lines of code changes to implement.

- We show that RealFormer can be used as a drop-in replacement of Transformer in BERT, outperforming both Post-LN and Pre-LN Transformers across a wide spectrum of model sizes for pre-training. In terms of fine-tuning, it even achieves competitive downstream results when pre-trained with only half the number of epochs of the baselines.

- We further demonstrate the genericity of RealFormer by using it as a drop-in replacement of two recent state-of-the-art Transformer variation models: ADMIN (Liu et al., 2020) from the Neural Machine Translation (NMT) domain, and ETC (Ainslie et al., 2020) that extends Transformer to handle long and structured inputs. We show that RealFormer can improve these models significantly on various tasks and lead to new state-of-the-art results.

- Qualitatively, we observe that attention in RealFormer tends to be *sparser* and more correlated across layers compared to baselines, which we believe may have some regularization effects that could stabilize training and benefit fine-tuning.

2 Related Work

Vaswani et al. (2017) proposed Transformer initially for NMT and it has profoundly changed the NLP field ever since.

Radford et al. (2018) demonstrated that generative pre-training of a Transformer-based language model (GPT) on a diverse corpus of unlabeled text can give large gains to downstream NLP tasks that suffer from scarce labeled data. Following this thread, Devlin et al. (2019) proposed to pre-train a *bidirectional* Transformer encoder (BERT) with a novel Masked Language Modeling as the main optimization objective. Since then, advances on many NLP tasks have been dominated by the self-supervised general-purpose pre-training, task-specific fine-tuning paradigm. Following BERT, there has been a large stream of work that explores better self-supervision objectives (*e.g.*, Yang et al. (2019); Clark et al. (2020)), larger pre-training data and better hyper-parameters (*e.g.*, Liu et al. (2019)), model parameter sharing (*e.g.*, Lan et al. (2019)), multi-task pre-training (*e.g.*, Sun et al. (2020); Raffel et al. (2020)). These efforts typically employ a Post-LN Transformer at their core. In this paper we adopt BERT to test different Transformer architectures because it is widely used and representative of this body of work.

Another notable thread of work focuses on improving the efficiency/scalability of Transformer. Typically, they try to reduce the *quadratic* complexity of the self-attention mechanism with respect to sequence length via low-rank methods (*e.g.*, Wang et al. (2020)), fixed strided attention patterns (*e.g.*, Child et al. (2019)), learnable attention patterns (*e.g.*, Kitaev et al. (2020); Roy et al. (2020)), memory-based global & local attention (*e.g.*, Ainslie et al. (2020); Beltagy et al. (2020); Zaheer et al. (2020)), and so on. These methods are particularly useful when dealing with long documents that go beyond the capacity of standard Transformer models. We would refer the reader to Tay et al. (2020) for a detailed survey. RealFormer is orthogonal to these methods as it focuses on improving various Transformer networks with an universal technique which can apply to these models as well. In this paper, we will use RealFormer to improve a state-of-the-art model, ETC (Ainslie et al., 2020), from this line of work to demonstrate the universality of RealFormer.

Some recent work (*e.g.*, Wang et al. (2019b); Xiong et al. (2020); Zhang et al. (2018); Huang et al. (2020); Zhang et al. (2019)) has studied normalization and parameter initialization schemes for Transformers, though most evaluations focus only on NMT to the best of our knowledge. In this strand, Liu et al. (2020) recently proposed ADMIN, which achieved state-of-the-art results on multiple popular NMT benchmarks. In this paper, we will take ADMIN as an example to (1) evaluate RealFormer in settings involving decoders, and (2) show that it is possible to apply RealFormer on top of this line of work.

3 RealFormer

3.1 Standard Transformer

There is an encoder and a decoder in Transformer (Vaswani et al., 2017). Since they work in a similar way, here we only introduce the encoder and refer the reader to the original paper for complete details.

There are two sub-layers inside each layer of a Transformer encoder. The first sub-layer contains a Multi-Head Attention module that computes output embeddings of a set of queries (Q) by aggregating the embeddings (V) of a set of keys (K):

$$\begin{aligned}\text{MultiHead}\,(Q, K, V) = \\ \text{Concat}\,(head_1, ..., head_h)\, W^O,\end{aligned}$$

where $head_i = \text{Attention}\,(QW_i^Q, KW_i^K, VW_i^V)$. Q and K are matrices with dimension d_k and V is a matrix with dimension d_v. W_i^Q, W_i^K, and W_i^V are matrices that linearly project queries, keys, and values into the "attention space" of the i-th head. W^O is a matrix that linearly transforms the concatenation of the outputs of all heads.

The attention function is typically implemented with a Scaled Dot-Product Attention module (Vaswani et al., 2017) which computes a weighted sum of the values:

$$\text{Attention}\,(Q', K', V') = \text{Softmax}\,(\frac{Q'K'^T}{\sqrt{d_k}})\, V',$$

where matrix $\frac{Q'K'^T}{\sqrt{d_k}}$ contains the raw attention scores for each (query, key) pair. These scores are normalized via the Softmax function for each query and then act as weights for the corresponding vectors in V'.

The second sub-layer contains a fully-connected Feed-Forward Network (FFN) module with one hidden layer:

$$\text{FFN}\,(x) = \sigma\,(x\,W_1 + b_1)\,W_2 + b_2,$$

where σ is an activation function usually implemented with ReLU or GELU (*e.g.*, Devlin et al. (2019)). FFN is applied to each position in the sequence separately and identically. Finally, there are Layer Norm (LN) modules inserted into the above two sub-layers to stabilize training.

As shown in Figure 1, there are two canonical designs of the Transformer network which only differ in the ways they organize the modules. Post-LN is the original architecture proposed by Vaswani et al. (2017) which normalizes the outputs at the end of each sub-layer. In contrast, Pre-LN normalizes sub-layer inputs instead and creates a direct path (without LN in the way) to propagate embeddings of the tokens in the sequence.

3.2 Residual Attention Layer Transformer

RealFormer uses a Post-LN style Transformer[3] as backbone and adds skip edges to connect

---

[3] Potentially we could also use Pre-LN, but Post-LN tends to outperform Pre-LN, as we will show in Section 4.

Multi-Head Attention modules in adjacent layers, as shown in Figure 1 (c). More formally, it adds $Prev$, the pre-softmax attention scores from the previous layer with shape $(\#heads, from\_seq\_len, to\_seq\_len)$,[4] as one additional input to the Multi-Head Attention module in the current layer:

$$\begin{aligned}&\text{ResidualMultiHead}\left(Q, K, V, Prev\right) = \\ &\qquad \text{Concat}\left(head_1, ..., head_h\right) W^O,\end{aligned}$$

where $head_i = \text{ResidualAttention}\left(QW_i^Q, KW_i^K, VW_i^V, Prev_i\right)$ and $Prev_i$ is the slice of $Prev$ with shape $(from\_seq\_len, to\_seq\_len)$ corresponding to $head_i$. ResidualAttention adds "residual scores" on top of $Prev_i$ and then computes the weighted sum as usual:

$$\begin{aligned}&\text{ResidualAttention}\left(Q', K', V', Prev'\right) = \\ &\qquad \text{Softmax}\left(\frac{Q'K'^T}{\sqrt{d_k}} + Prev'\right) V'.\end{aligned} \tag{1}$$

Finally, new attention scores $\frac{Q'K'^T}{\sqrt{d_k}} + Prev'$ are passed over to the next layer.

Implementing RealFormer takes no more than adding a few lines of code to the backbone Transformer. Note that the RealFormer technique can be straightforwardly applied for Transformer variations and even when there are more than one type of attention modules in the network. For example, there are encoder self-attention, encoder-decoder attention, and decoder self-attention modules for machine translation. In such cases, RealFormer simply adds skip edges to create *multiple* direct paths, one for each type of attention module.

Discussion. Adding skip edges is equivalent to using a softmax over the running sum of the attention scores (to get attention probabilities). This might be sub-optimal for very deep networks due to the linear scaling nature of sum. Empirically, we find it helpful to use running *mean* instead in such cases, which can be viewed as adding a temperature (*i.e.*, #traversed layers) to the softmax function in Eq. 1 of each RealFormer layer.

4 Experiments

To demonstrate that RealFormer is general-purpose, we conduct comprehensive empirical studies on a variety of tasks including (masked) language modeling, machine translation, and long document modeling, based on corresponding state-of-the-art models: BERT, ADMIN, and ETC. To evaluate its robustness, we only do minimal (if at all) hyper-parameter tuning for RealFormer and initialize all parameters the same way as the backbone Transformers. More aggressive hyper-parameter tuning or better initialization might further improve RealFormer, though we leave them for future work. Details of our experiments are included in Appendix.

4.1 BERT

BERT (Devlin et al., 2019) has been the standard way of transferring knowledge from large unlabeled text corpora by pre-training a bidirectional Transformer encoder. Numerous downstream NLP tasks suffering from scarcity of supervised data have benefited considerably by fine-tuning a pre-trained BERT model. This drives us to adopt BERT as the main evaluation setup for RealFormer.

Experiment setup. Our experiments are based on the official BERT repository[5]. We follow the standard pre-training setup (dataset: Wikipedia + BookCorpus, vocab: uncased 30K, max sequence length: 512[6], dropout: 10%, learning rate: 1e-4, learning rate schedule: warm up and then linearly decay to 0, weight decay: 0.01, optimizer: AdamW, objective: Masked Language Modeling + Next Sentence Prediction, etc.) to compare three Transformer models: Post-LN, Pre-LN, and RealFormer. We experiment with Transformer architectures with a wide spectrum of sizes as detailed in Table 1. For simplicity, all models are pre-trained 1M steps with a mini-batch size of 512 (except that xLarge uses 256 to avoid TPU OOM). Note that we use a larger mini-batch size than Devlin et al. (2019), *i.e.*, doubling the amount of pre-training epochs, to show more complete behavior of different models.

We use exactly the same setup for all three Transformer architectures except that for the Pre-LN Transformer we follow the initialization strategy suggested by Radford et al. (2019) and Child et al. (2019).[7] Note that for simplicity RealFormer reuses all hyper-parameter setups from Post-LN Transformer unless otherwise specified. We use running sum of attention scores for all RealFormer

---

[4] Batch dimension is omitted for ease of discussion.

[5] `https://github.com/google-research/bert`

[6] Unlike BERT which uses a reduced sequence length for the first 90% of steps, we always use 512 for simplicity.

[7] We also tried BERT-style initialization in our pilot experiments but without success.

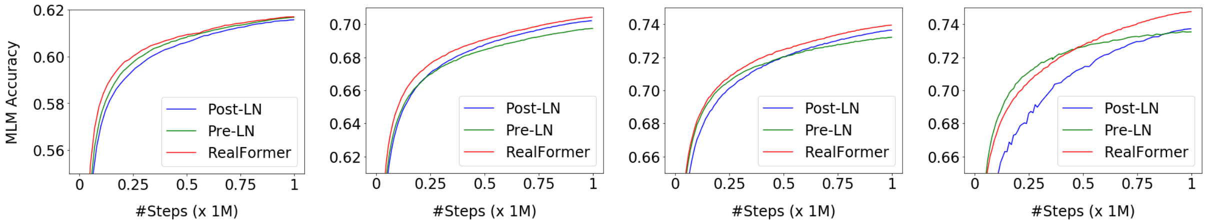

(a) BERT-Small (b) BERT-Base (c) BERT-Large (d) BERT-xLarge

Figure 2: Development set MLM accuracy (best viewed in color). Improvement gap of RealFormer over the best baseline tends to increase with model size. Note that these are without hyper-parameter tuning for RealFormer. (As we will show later, RealFormer can benefit from larger learning rates and even double the gap size over Post-LN.)

Model	**L**	**H**	**A**	**I**	**P**
BERT-Small	4	512	8	2,048	30M
BERT-Base	12	768	12	3,072	110M
BERT-Large	24	1,024	16	4,096	340M
BERT-xLarge	36	1,536	24	6,144	1B

Table 1: Model architectures for BERT evaluation. L: #layers, H: hidden size, A: #heads, I: intermediate size, P: approximate #parameters.

Model	**Post-LN**	**Pre-LN**	**RealFormer**
BERT-Small	61.57%	61.67%	**61.70%**
BERT-Base	70.20%	69.74%	**70.42%**
BERT-Large	73.64%	73.21%	**73.94%**
BERT-xLarge	73.72%	73.53%	**74.76%**

Table 2: Development set MLM accuracy after pre-training 1M steps. RealFormer outperforms baselines more as model size increases.

models except xLarge (for which we use running mean for reasons discussed in Section 3.2).

All experiments are performed on 128 or 256 TPU v3 cores depending on model sizes (see Appendix A.1 for details).

4.1.1 Pre-training Results

To evaluate pre-trained models, we report Masked Language Modeling (MLM) accuracy[8] on a randomly held-out development set. As shown in Table 2, RealFormer outperforms the two baseline Transformers considerably with the gap increasing with model size. Our hypothesis is that larger models are inherently harder to train (*e.g.*, we observe that BERT with Post-LN is unstable and sometimes even diverges for xLarge) and RealFormer can help regularize the model and stabilize training.

We also report the pre-training curves in Figure 2. One interesting finding is that the Pre-LN Transformer seems to favor the combination of extra large models and a small number of steps, though it is consistently outperformed by the other two in "regular-sized" settings or given enough pre-training budget.

4.1.2 Downstream Results

To evaluate downstream performance, we fine-tune the above pre-trained BERT-Large models on both sentence-level (*i.e.*, GLUE) and token-level (*i.e.*, SQuAD) NLP tasks.

GLUE. General Language Understanding Evaluation (GLUE) is a canonical benchmark proposed by Wang et al. (2019a) for evaluating models across a diverse set of NLU tasks. Following the fine-tuning recipe in Devlin et al. (2019), we use a mini-batch size of 32 for all models on all tasks. For each (task, model) pair, we select number of fine-tuning epochs in {2, 3, 4} and learning rate in {6e-6, 8e-6, 1e-5, 2e-5, 3e-5, 4e-5, 5e-5}.[9] For each setup, we run the experiment five times and report the best median performance and the corresponding standard deviation on the development set.

Results are tabulated in Table 3. We exclude the problematic WNLI task following Devlin et al. (2019). For each task, we report metric(s) suggested by Wang et al. (2019a). RealFormer achieves the best overall performance and outperforms both baselines on most tasks, testifying its strength at tackling sentence-level tasks.

SQuAD. The Stanford Question Answering Dataset (SQuAD v1.1) is a reading comprehension dataset consisting of 100K crowd-sourced question-answer pairs, where the answer to each question is

---

[8] All methods achieved similar (and great) results on Next Sentence Prediction presumably because it is much easier.

[9] We use a slightly wider range than Devlin et al. (2019) to better accommodate all three models.

Task	Post-LN	Pre-LN	RealFormer
MNLI-m	85.96±0.11	85.03±0.12	**86.28**±0.14
MNLI-nm	85.98±0.14	85.05±0.19	**86.34**±0.30
QQP	91.29±0.10	91.29±0.16	**91.34**±0.03
QQP (F1)	**88.34**±0.15	88.33±0.26	88.28±0.08
QNLI	92.26±0.15	**92.35**±0.26	91.89±0.17
SST-2	92.89±0.17	93.81±0.13	**94.04**±0.24
CoLA (MC)	58.85±1.31	58.04±1.50	**59.83**±1.06
STS-B (PC)	90.08±0.27	90.06±0.33	**90.11**±0.56
STS-B (SC)	89.77±0.26	89.62±0.28	**89.88**±0.54
MRPC	**87.50**±0.67	86.76±5.64	87.01±0.91
MRPC (F1)	**91.16**±0.45	90.69±3.16	90.91±0.65
RTE	71.12±2.52	68.59±1.52	**73.65**±0.90
Overall	84.01	83.47	**84.53**

Table 3: GLUE development set results of fine-tuning BERT-Large models in Table 2. Default metric: accuracy, MC: Matthews correlation, PC: Pearson correlation, SC: Spearman correlation. Overall: first average metrics within each task (if there are 1+) and then across tasks. Numbers in smaller font are standard deviations. All numbers are scaled by 100.

SQuAD	Public	Post-LN	Pre-LN	RealFormer
v1.1 (F1)	90.9	91.68±0.12	91.06±0.09	**91.93**±0.12
v1.1 (EM)	84.1	85.15±0.13	83.98±0.24	**85.58**±0.15
v2.0 (F1)	81.9	82.51±0.12	80.30±0.12	**82.93**±0.05
v2.0 (EM)	78.7	79.57±0.12	77.35±0.16	**79.95**±0.08

Table 4: SQuAD development set results of fine-tuning BERT-Large models in Table 2. EM: exact match. Public: Post-LN results from Devlin et al. (2019). Numbers in smaller font are standard deviations. All numbers are scaled by 100.

a segment of text from the corresponding reading passage (Rajpurkar et al., 2016). SQuAD v2.0, a later version, further extends with over 50K unanswerable questions written adversarially by crowdworkers to look similar to answerable ones.

We follow the fine-tuning recipe in Devlin et al. (2019) for all three Transformer models on these two datasets without using any additional data such as TriviaQA (Joshi et al., 2017). For both v1.1 and v2.0, we select mini-batch size in {32, 48}, number of fine-tuning epochs in {2, 3, 4}, and learning rate in {2e-5, 3e-5, 4e-5, 5e-5}. For each setup, we run the experiment five times and report the best median performance and the corresponding standard deviation on the development set. As we can see from Table 4, RealFormer outperforms the two baselines considerably, attesting its strength at tackling token-level tasks.

Task	Post-LN (500K)	Post-LN (1M)	RealFormer (500K)
GLUE	83.84	84.01	84.34
v1.1 (F1)	91.46±0.18	91.68±0.12	91.56±0.09
v1.1 (EM)	84.87±0.24	85.15±0.13	85.06±0.12
v2.0 (F1)	81.44±0.50	82.51±0.12	82.52±0.55
v2.0 (EM)	78.64±0.48	79.57±0.12	79.54±0.54
Overall	83.97	84.37	**84.51**

Table 5: Downstream development set results of fine-tuning BERT-Large with Post-LN and RealFormer pre-trained with different number of steps. v*: SQuAD version, EM: exact match. Overall: First average across SQuAD and then GLUE. Numbers in smaller font are standard deviations. All numbers are scaled by 100.

4.1.3 Research Questions

How well does RealFormer perform with half the pre-training budget? Although RealFormer has outperformed both Post-LN and Pre-LN considerably when pre-training 1M steps, we are also interested in investigating its potential when the pre-training budget is more limited. For this purpose, we experiment with BERT-Large models. In particular, we take the 500K step checkpoint of the pre-trained RealFormer in Table 2 and fine-tune it on GLUE and SQuAD datasets using exactly the same procedure as described above. Comparison results against the strongest baseline, Post-LN Transformer pre-trained 500K (checkpoint) and 1M steps respectively, are collected in Table 5. We can see that RealFormer with merely half the amount of pre-training epochs can beat Post-LN (1M) on GLUE with a significant margin, and almost match its performance on SQuAD.

Does a larger learning rate help? As suggested by some recent work (*e.g.*, Xiong et al. (2020)), Pre-LN Transformer may benefit from using larger learning rates. To this end, we follow the pre-training procedure detailed earlier and switch to a larger learning rate, 2e-4, to pre-train BERT-Large with the three Transformer models. Development set MLM accuracy with training steps can be found in Figure 3. We find that both Pre-LN and RealFormer can reap some benefits of using larger learning rates with RealFormer seeming to benefit slightly more in this case (73.94% → 74.31%) compared to Pre-LN (73.21% → 73.46%). Post-LN diverges with the learning rate of 2e-4. Note that it also means that RealFormer can outperform Post-LN, the strongest baseline, actually with a

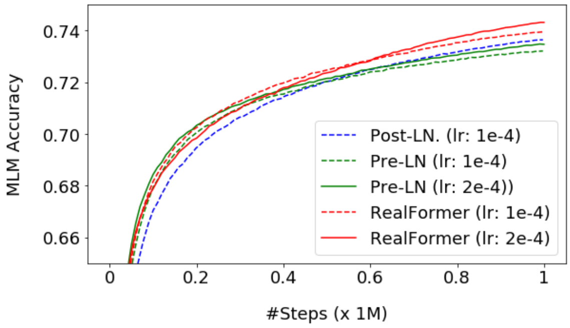

Figure 3: Development set MLM accuracy of BERT-Large with different learning rates (best viewed in color). RealFormer seems to benefit slightly more from using a larger, non-default learning rate compared to Pre-LN, while Post-LN diverges with 2e-4.

prominent gap, 0.67% (*i.e.*, 74.31% - 73.64%), for pre-training, though with only minimal learning rate tuning.

Is attention sparser in RealFormer? We conduct one empirical study to observe the qualitative differences between RealFormer and Post-/Pre-LN Transformers. We randomly sample 8,192 examples from the held-out development set and visualize the distribution of attention probabilities of each token in these examples across heads in all layers. In particular, for each (token, layer, head) triplet, we compute the entropy of the attention probabilities as the "sparsity measure" of attention. Intuitively, as entropy gets lower, the attention weight distribution becomes more skewed and therefore attention is sparser.

In a similar fashion to Ramsauer et al. (2020), we use violin plots to show the entropy distributions of the pre-trained BERT-Base model with RealFormer from Table 2 (see Figure 4). Plots for the two baseline Transformers in Table 2 are included in Appendix A.4. Each row is a layer in BERT-Base and each column is an attention head.

We find that attention tends to get sparser for later (upper) layers for all three Transformers. However, RealFormer differs from the two baselines in the following ways:

- RealFormer has *significantly sparser* attention for top layers (layer 9-11);
- RealFormer tends to have lower variance across all layers, which means that attention density is less input-dependent.

We hypothesize that the above two properties might be a sign of stableness and benefit fine-tuning.

Dropout	Post-LN	Pre-LN	RealFormer
0%[10]	71.16%	69.80%	**71.30%**
10%	73.64%	73.21%	**73.94%**
20%	73.21%	72.97%	**73.66%**

Table 6: Development set MLM accuracy of BERT-Large with different dropout rates.

Do attention heads in layer L resemble those in layer $L-1$? Since RealFormer uses a residual attention scheme, it is interesting to show to what extent an attention head is "relying on" the corresponding head in the previous layer. To this end, we take each of the three pre-trained BERT-Base models in Table 2 and compute the Jensen-Shannon Divergence (JSD) between attention probabilities in each pair of *vertically* adjacent heads, *i.e.*, JSD $(\text{head}_i^L, \text{head}_i^{L-1})$, for $1 \leq L < 12$ and $0 \leq i < 12$.

Appendix A.5 demonstrates detailed JSD distributions of Post-LN and RealFormer respectively based on 8,192 held-out examples. We observe that RealFormer tends to have *significantly lower* JSD values (*i.e.*, indicating more "similar" attention across layers), especially for heads in middle layers. This might mean that RealFormer has some regularization advantages and provides one hypothesis for why it tends to outperform Post-LN more for larger models. Note that head_i^L can still be useful even if it has exactly the same attention probabilities with head_i^{L-1} because of the existence of the FFN sublayer and the potential differences in value matrices (*i.e.*, V' in Eq. 1).

Is residual attention really necessary? One may wonder whether increasing dropout rate can already regularize large models well so that residual attention is redundant. To this end, we experiment with different dropout rates for pre-training BERT-Large with different Transformers (following the procedures in Section 4.1.1). Results are collected in Table 6, from which we can see that (1) RealFormer outperforms the two baselines across all dropout settings, and (2) simply increasing dropout rate can not regularize Transformer models as well as what residual attention appears to be doing.

4.2 ADMIN

To evaluate the genericity of RealFormer, here we try it on top of ADMIN (Liu et al., 2020), a state-of-

---

[10] When dropout rate is 0%, we use early stopping for all models due to overfitting.

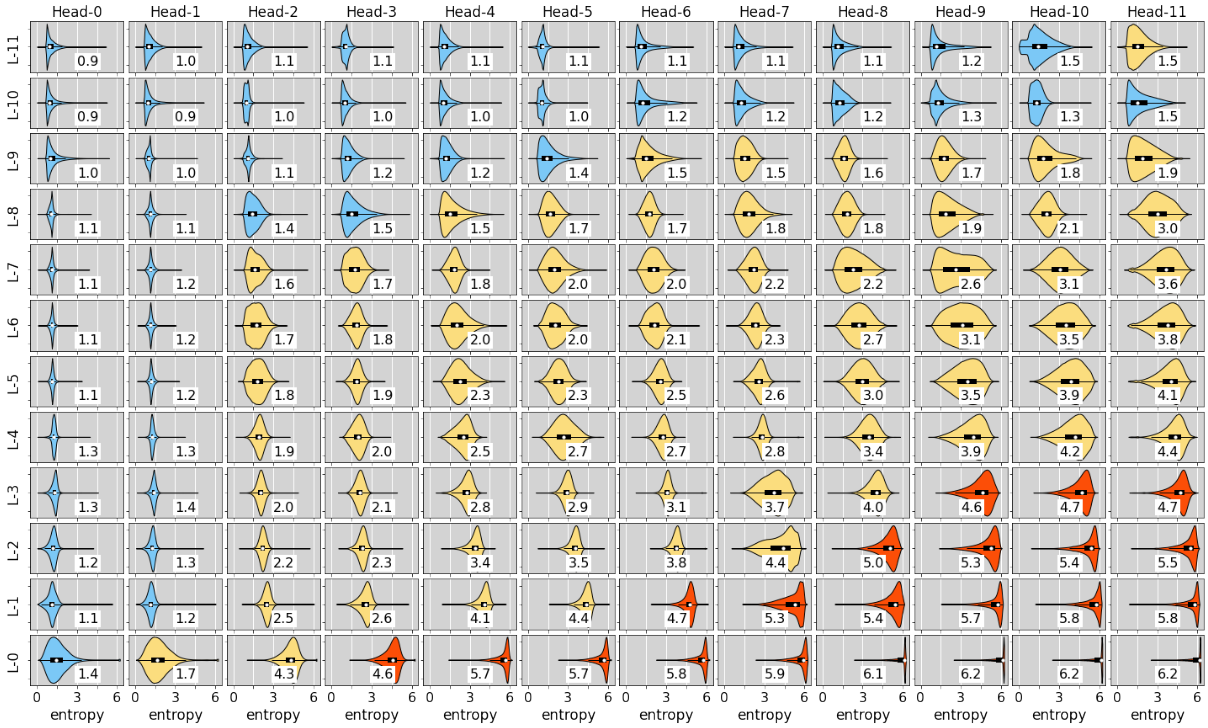

Figure 4: Distribution of entropies of the attention probabilities of the tokens of 8,192 held-out examples using the pre-trained BERT-Base with **RealFormer** (see Section 4.1.1). For better legibility, (1) attention heads in each layer are ordered by their medians of entropies, and (2) distributions are color-coded based on the median of entropies: RED (median > 4.5), YELLOW ($1.5 \leq$ median ≤ 4.5), BLUE (median < 1.5), *i.e.*, colder colors mean sparser attention. There is a clear trend that higher layers tend to have *sparser* attention.

the-art NMT model without using either additional data or data augmentation. ADMIN adopts Post-LN as the backbone, which we simply replace with RealFormer. In particular, we add three types of skip edges for encoder-encoder, encoder-decoder, and decoder-decoder attention respectively to the Post-LN Transformer. Empirically, RealFormer with running mean of attention scores tends to outperform running sum for our experiments, therefore here we use the former exclusively for brevity.

We use two popular NMT benchmarks, WMT'14 En-De and WMT'14 En-Fr, and follow Liu et al. (2020) for all training setups on both benchmarks except that in all cases (1) we select the peak learning rate from {5e-4, 1e-3, 1.2e-3} and use a linear learning rate decay schedule (instead of inverse sqrt);[11] (2) we train RealFormer only 50 epochs (in contrast, ADMIN trains 100 epochs on En-De and 50 epochs on En-Fr); and (3) we average across the last 25 checkpoints (while ADMIN uses the last 10). More checkpoints are helpful for us (especially for large models) presumably because the last few are not "diverse" enough as learning rate decays to 0.

Our experiments are performed on NVIDIA A100 GPUs, based on the official ADMIN repository[12]. We follow Liu et al. (2020) to configure the amount of GPUs to use for different setups.

BLEU scores on test sets are collected in Table 7. For fair comparisons, we also run ADMIN using our above setups and report results in the same table. Following Liu et al. (2020), all networks (including both encoders and decoders) share the same width setup (hidden size 512, intermediate size 2048, 8 heads) and only vary in depth. RealFormer outperforms all baselines across all depths considerably with a new state-of-the-art BLEU score (43.97) on En-Fr for models not using additional data or data augmentation to the best of our knowledge. One interesting observation here is that RealFormer does not always lead to larger improvement gaps for larger models, which might be due to the checkpoint averaging mechanism (which potentially regularizes large models reasonably well).

---

[11] With inverse sqrt decay, we find that RealFormer tends to favor larger peak learning rates than what Liu et al. (2020) uses, and we have also seen improvements in most cases.

[12] `https://github.com/LiyuanLucasLiu/Transformer-Clinic`

Model	En-De			En-Fr	
	6L-6L	12L-12L	18L-18L	6L-6L	60L-12L
Post-LN	27.80	failed	failed	41.29	failed
Pre-LN	27.27	28.26	28.38	40.74	43.10
ADMIN	27.90	28.58	29.03	41.47	43.80
ADMIN†	28.06	28.85	29.11	41.65	43.72
Ours	**28.17**	**29.06**	**29.35**	**41.92**	**43.97**

Table 7: Test set BLEU scores on two WMT'14 benchmarks using different sizes of models. xL-yL: #Encoder layers-#Decoder layers. First three rows are from Liu et al. (2020). Ours is switching the backbone of ADMIN from Post-LN to RealFormer. †Our run of ADMIN using the same setups as RealFormer.

4.3 ETC

Extended Transformer Construction (ETC) is a recent sparse attention mechanism proposed by Ainslie et al. (2020) and Zaheer et al. (2020) to handle long context. It has achieved state-of-the-art results on four natural language benchmarks requiring long and/or structured inputs. Here we evaluate RealFormer on top of ETC models on these benchmarks including WikiHop (Welbl et al., 2018), HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019), and OpenKP (Xiong et al., 2019). They vary significantly in terms of dataset size, context length, and structure in text inputs. Please refer to Ainslie et al. (2020) for more details.

Our experiments are based on the official ETC repository[13]. We take the ETC-Large model (24 layers, 1024 hidden size, 16 heads), add residual attention edges (*i.e.*, using running sum), and follow all the pre-training and fine-tuning recipes as well as hardware setups detailed in Ainslie et al. (2020). For each fine-tuning setup, we run the experiment five times and report the best median performance and the corresponding standard deviation on the development set in Table 8. RealFormer can improve ETC consistently across all four benchmarks.

As of June 2021, we are ranked the first on the WikiHop leaderboard[14] with a test accuracy of 84.4% (2.1% absolute improvement over the previous best result).

Task	Metric	ETC	Ours
WikiHop	Accuracy	78.92$_{\pm 0.14}$	**79.21**$_{\pm 0.38}$
HotpotQA	Ans. F1	80.38$_{\pm 0.13}$	**80.86**$_{\pm 0.16}$
	Sup. F1	89.07$_{\pm 0.06}$	**89.21**$_{\pm 0.12}$
	Joint F1	73.12$_{\pm 0.19}$	**73.57**$_{\pm 0.19}$
Natural Questions	Long Ans. F1	77.70$_{\pm 0.15}$	**77.93**$_{\pm 0.31}$
	Short Ans. F1	58.54$_{\pm 0.41}$	**59.10**$_{\pm 0.81}$
	Average F1	68.07$_{\pm 0.17}$	**68.51**$_{\pm 0.56}$
OpenKP	F1@3	44.06$_{\pm 0.08}$	**44.27**$_{\pm 0.08}$

Table 8: Development set results of ETC-Large. Ours is adding residual attention edges to ETC. Numbers in smaller font are standard deviations. All numbers are scaled by 100.

5 Conclusions

We propose RealFormer, a simple, generic, and cheap technique based on the novel idea of residual attention to improve Transformer-based networks. Quantitatively, we show that RealFormer can improve a diverse set of state-of-the-art Transformer-based models considerably for tasks like Masked Language Modeling, Neural Machine Translation, and long document modeling. Qualitatively, we show that RealFormer tends to have comparatively *sparser* attention, both within heads and across heads in adjacent layers.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283.

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977.*

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 268–284.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150.*

---

[13] `https://github.com/google-research/google-research/tree/master/etcmodel`

[14] `http://qangaroo.cs.ucl.ac.uk/leaderboard.html`

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *European Cconference on Computer Vision*, pages 630–645.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5763.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, et al. 2020. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8968–8975.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019b. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5178–5187.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5753–5763.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 897–908.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. 2018. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*.

Model	Post-LN	Pre-LN	RealFormer
BERT-Small	5.4 hrs	5.3 hrs	5.9 hrs
BERT-Base	20 hrs	20 hrs	23 hrs
BERT-Large	58 hrs	58 hrs	66 hrs
BERT-xLarge	136 hrs	137 hrs	137 hrs

Table 9: Pre-training time of different BERT models in Table 2. We use 128 TPU v3 cores and mini-batch size 512 for BERT-Small/Base/Large, and 256 TPU v3 cores and mini-batch size 256 for BERT-xLarge.

Task	500K-step			1M-step		
	BS	LR	EP	BS	LR	EP
MNLI	32	2e-5	2	32	1e-5	4
QQP	32	3e-5	4	32	2e-5	4
QNLI	32	3e-5	4	32	2e-5	2
SST-2	32	3e-5	2	32	1e-5	4
CoLA	32	2e-5	4	32	1e-5	3
STS-B	32	2e-5	3	32	2e-5	4
MRPC	32	2e-5	4	32	1e-5	4
RTE	32	1e-5	4	32	1e-5	4
SQuAD v1.1	48	3e-5	2	48	3e-5	2
SQuAD v2.0	32	5e-5	2	48	5e-5	2

Table 10: Hyper-parameter configurations on GLUE and SQuAD for best-performing BERT-Large with RealFormer (pre-trained 500K steps and 1M steps respectively). BS: mini-batch size, LR: learning rate, EP: #fine-tuning epochs.

Model	En-De			En-Fr	
	6L-6L	12L-12L	18L-18L	6L-6L	60L-12L
ADMIN	9.3 hrs	16 hrs	23 hrs	28 hrs	70 hrs
Ours	9.4 hrs	16 hrs	23 hrs	28 hrs	72 hrs
#GPUs	4	4	4	8	16

Table 11: Number of GPUs and training time used for each model. Ours is switching the backbone of ADMIN from Post-LN to RealFormer (using running mean of attention scores).

A Appendices

A.1 Training Details: BERT

All our experiments are conducted on TPUs based on `https://github.com/google-research/bert`, the official BERT repository in TensorFlow (Abadi et al., 2016).

Pre-training. We use 128 TPU v3 cores (*i.e.*, 64 chips) for BERT-Small/Base/Large and 256 TPU v3 cores (*i.e.*, 128 chips) for BERT-xLarge. Table 9 demonstrates the time used to pre-train each model 1M steps. We can see that there is 10%-15% performance drop when adding residual attention edges for all sizes except xLarge. Our suspicion is that additions are not as optimized as other ops like matrix multiplications on TPU v3 cores. There is a much smaller performance drop for xLarge though, which might indicate that addition scales nicely compared to other ops on TPU v3 cores. As we will show later in Appendix A.2, performance drop on GPUs is almost negligible across different Transformer sizes, suggesting that it is hardware-dependent.

Fine-tuning. We use 8 TPU v2 cores (*i.e.*, 4 chips) to fine-tune each model. Best hyper-parameter configurations for BERT-Large with RealFormer on GLUE and SQuAD are collected in Table 10. We include RealFormer pre-trained both 1M and 500K steps, corresponding to the results in Table 3, 4, and 5.

A.2 Training Details: ADMIN

All our NMT experiments are conducted on NVIDIA A100 GPUs based on `https://github.com/LiyuanLucasLiu/Transformer-Clinic`, the official ADMIN repository implemented via fairseq (Ott et al., 2019). We use the same scripts to collect and process data and evaluate all models.

Following Liu et al. (2020), we use different number of GPUs for different setups, as detailed in Table 11. For our runs of ADMIN and RealFormer (*i.e.*, the last two rows in Table 7), learning rate is set to 5e-4 on WMT’14 En-De and 1.2e-3 on WMT’14 En-Fr across different model sizes. We train all models 50 epochs across the two benchmarks and average across the last 25 checkpoints (corresponding to the last 25 epochs).

Training time comparison of ADMIN and our model using the same setups is shown in Table 11. Adding residual attention edges and using running mean of attention scores do not incur significant performance drop on GPUs across different model sizes.

A.3 Training Details: ETC

All our experiments are conducted on TPU v3 cores based on the official ETC repository in TensorFlow: `https://github.com/google-research/google-research/tree/master/etcmodel`.

Pre-training. As is the case with ETC-Large (Ainslie et al., 2020), we find that pre-training ETC-Large with RealFormer can also benefit sig-

Task	Instance Statistics				Hyper-parameter		
	#Training	Median Length	95%tile Length	Max Length	BS	LR	EP
WikiHop	43,738	1,541	3,994	20,337	32	4e-5	15
HotpotQA	90,447	1,227	1,810	3,560	32	4e-5	5
Natural Questions	307,373	4,004	17,137	156,551	64	3e-5	2
OpenKP	133,724	761	4,546	89,183	64	3e-5	2

Table 12: Statistics of benchmarks and the hyper-parameter configurations for best-performing ETC-Large with RealFormer. BS: mini-batch size, LR: learning rate, EP: #fine-tuning epochs.

nificantly from lifting weights from RoBERTa (Liu et al., 2019). Note however that we lift from the same RoBERTa checkpoint as our ETC-Large baseline, which could be disadvantageous to our model since RoBERTa is pre-trained *without* residual attention.

Fine-tuning. Statistics of the four benchmarks and the corresponding best hyper-parameter configurations for ETC-Large with RealFormer are collected in Table 12. On Natural Questions and OpenKP, we simply reuse the best configurations for our ETC-Large baselines as reported in Ainslie et al. (2020). On WikiHop and HotpotQA, we follow the hyper-parameters search space specified in Ainslie et al. (2020) for ETC-Large.[15] In addition, on WikiHop we found it to be slightly better (development set accuracy 79.21 vs 78.96) to turn off RealFormer during fine-tuning (*i.e.*, adding no residual attention but still loading from our pre-trained RealFormer checkpoint); therefore we adopted this setup for WikiHop in Table 8 and our leaderboard submission.

A.4 Entropy Distribution of Pre-trained Baseline Transformer Models

Violin plots demonstrating the entropy distributions of the pre-trained BERT-Base models with Post-LN and Pre-LN Transformers from Table 2 are included in Figure 5.

A.5 Jensen-Shannon Divergence of Different Pre-trained Transformers

We use violin plots to show the Jensen-Shannon Divergence distributions of the pre-trained BERT-Base models with Post-LN and RealFormer from Table 2 respectively (see Figure 6). Each row is a pair of adjacent layers in BERT-Base and each column is an attention head. Instead of computing one scalar value for each head pair, we show the full distribution based on the tokens in 8,192 held-out examples, *i.e.*, each data point is the JSD between the attention probabilities of a token at these two heads. For better legibility, we color code these plots to help distinguish head pairs with relatively "similar" attention (BLUE: median < 0.25) and relatively "distinct" attention (RED: median > 0.75) from the rest (YELLOW: $0.25 \leq$ median ≤ 0.75).

Note that JSD results from Post-LN are used only as a reference; we expect them to be "random" because there is no correspondence between heads in adjacent layers for Post-/Pre-LN. Proof: An equivalent Post-/Pre-LN can be constructed by permuting the order of attention heads in a layer (and the corresponding variables).

---

[15] On WikiHop, number of fine-tuning epochs is selected from {5, 10, 15} instead of {5, 10} for both ETC-Large and our model. We added 15 here following the official ETC repository.

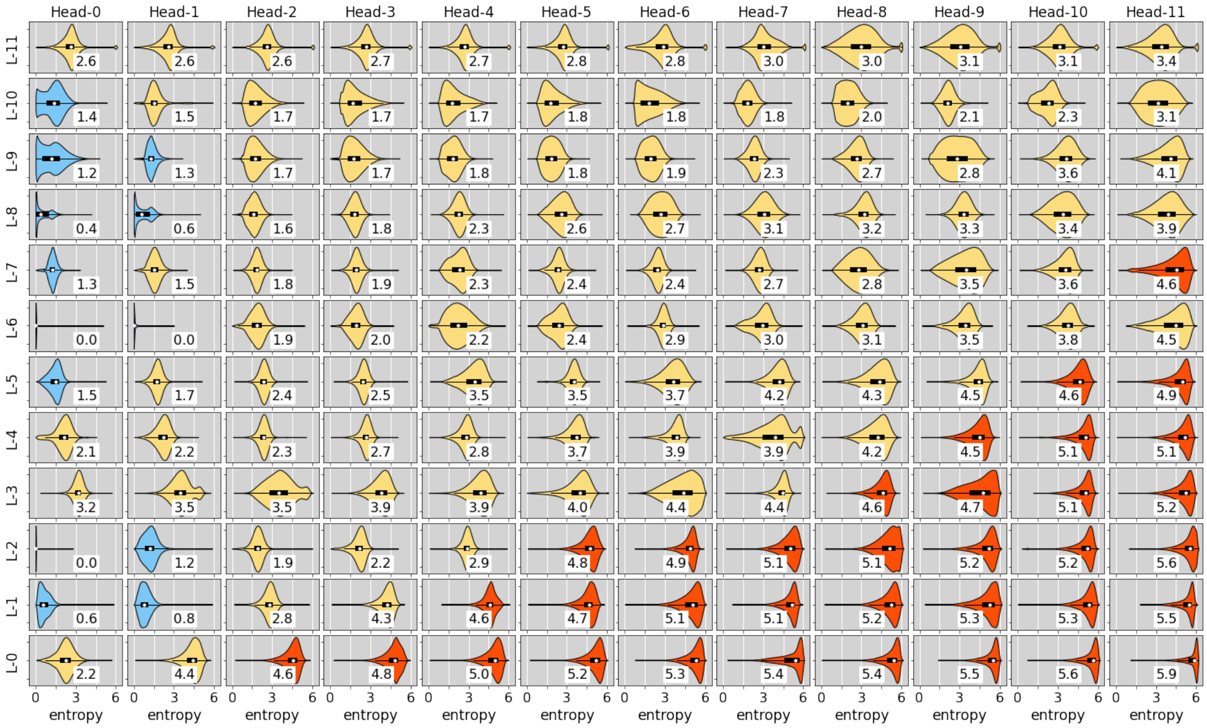

(a) Post-LN

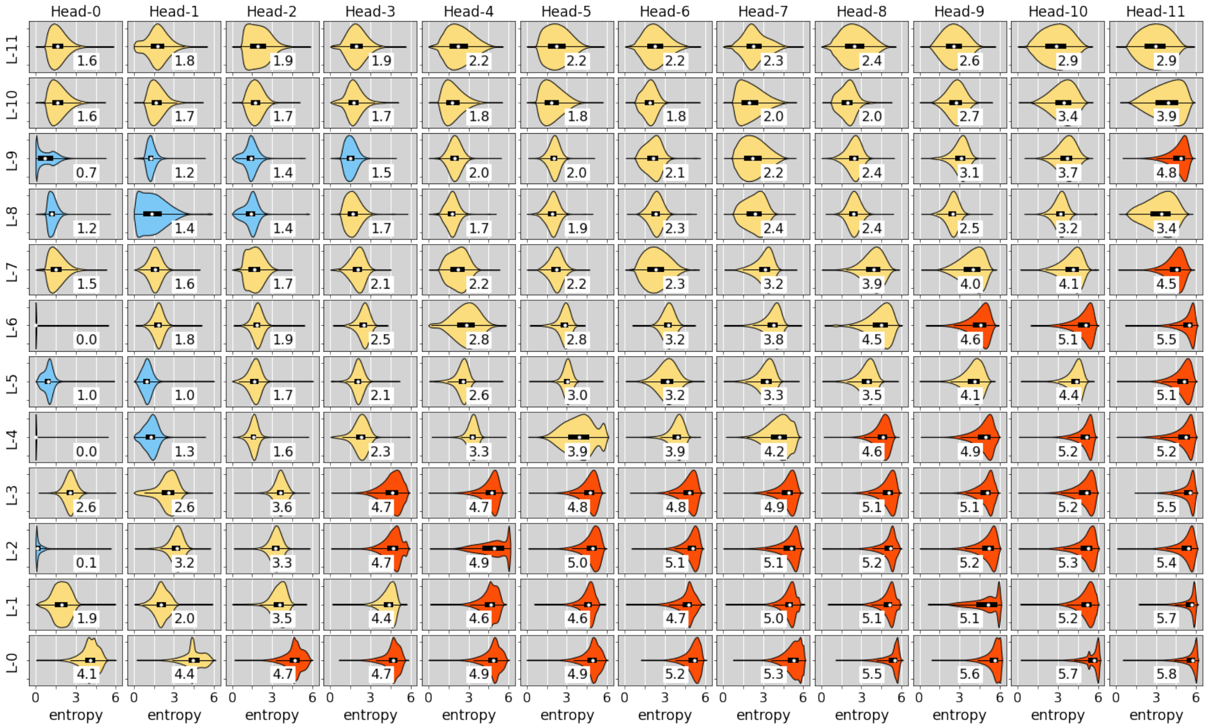

(b) Pre-LN

Figure 5: Distribution of entropies of the attention probabilities of the tokens of 8,192 held-out examples using the pre-trained BERT-Base with **Post-LN** and **Pre-LN** Transformer respectively (see Section 4.1.1). For better legibility, (1) attention heads in each layer are ordered by their medians of entropies, and (2) distributions are color-coded based on the median of entropies: RED (median > 4.5), YELLOW ($1.5 \leq$ median ≤ 4.5), BLUE (median < 1.5), *i.e.*, colder colors mean sparser attention. Note that here top layers (layer 9-11) tend to have larger entropies compared to RealFormer, which means that attention is relatively *denser*.

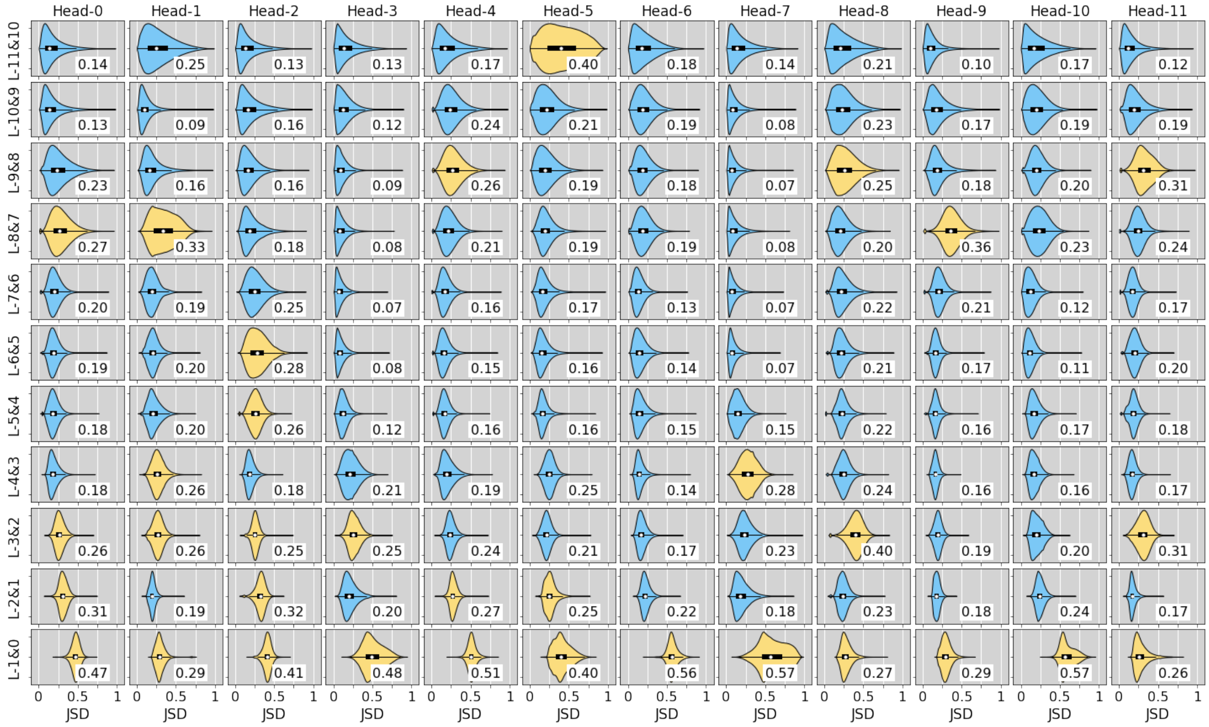

(a) RealFormer

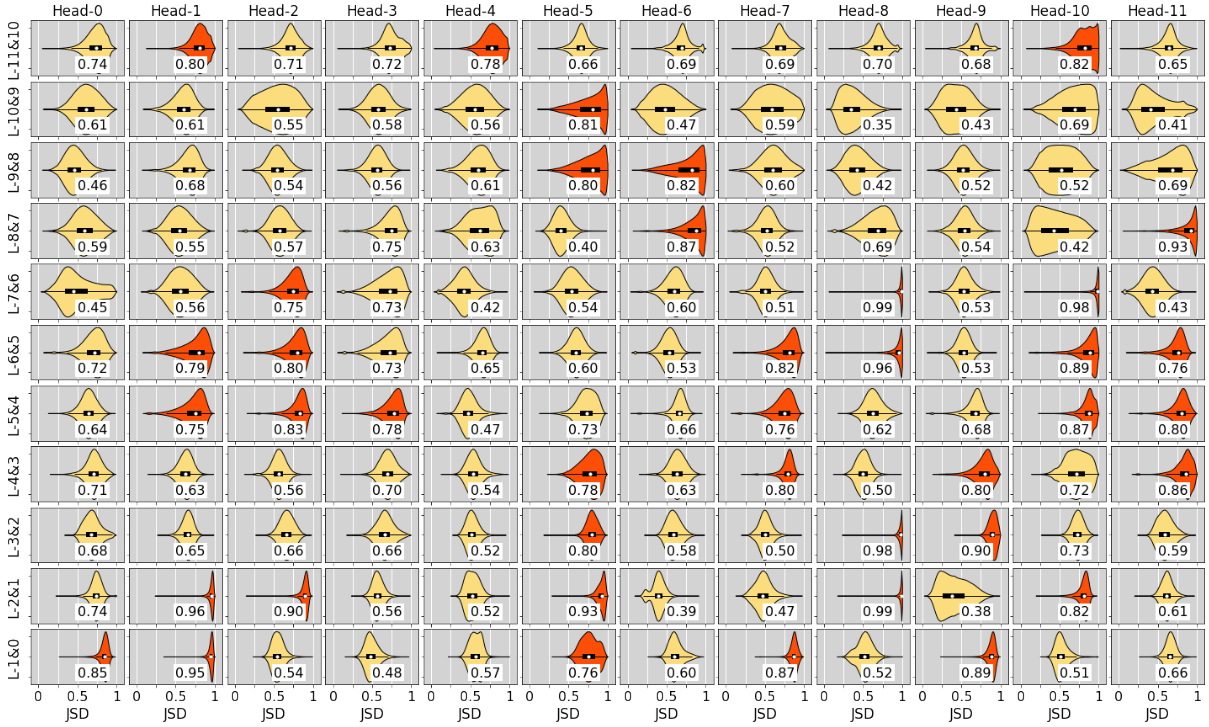

(b) Post-LN

Figure 6: Distribution of Jensen-Shannon Divergence (JSD) of attention probabilities in (vertically) adjacent attention heads, *i.e.*, $\mathrm{JSD}(\mathrm{head}_i^L, \mathrm{head}_i^{L-1})$. Based on 8,192 held-out examples using the pre-trained BERT-Base with **RealFormer** and **Post-LN** Transformer respectively (see Section 4.1.1). Distributions are color-coded based on the median of JSDs: RED (median > 0.75), YELLOW ($0.25 \leq$ median ≤ 0.75), BLUE (median < 0.25). *I.e.*, colder color means more “similar” attention heads across adjacent layers.

Promoting Graph Awareness in Linearized Graph-to-Text Generation

Alexander Hoyle♣* **Ana Marasović**†◇ **Noah A. Smith**†◇

♣Department of Computer Science, University of Maryland, College Park
†Allen Institute for Artificial Intelligence
◇Paul G. Allen School of Computer Science and Engineering, University of Washington
`hoyle@umd.edu, {anam,noah}@allenai.org`

Abstract

Generating text from structured inputs, such as meaning representations or RDF triples, has often involved the use of specialized graph-encoding neural networks. However, recent applications of pretrained transformers to linearizations of graph inputs have yielded state-of-the-art generation results on graph-to-text tasks. Here, we explore the ability of these linearized models to encode local graph structures, in particular their invariance to the graph linearization strategy and their ability to reconstruct corrupted inputs. Our findings motivate solutions to enrich the quality of models' implicit graph encodings via scaffolding. Namely, we use graph-denoising objectives implemented in a multi-task text-to-text framework. We find that these *denoising scaffolds* lead to substantial improvements in downstream generation in low-resource settings.

1 Introduction

Parameter-rich pretrained transformer language models succeed at generating text that is prima facie fluent, but that closer inspection will often reveal to be semantically transgressive (Bisk et al., 2020). Indeed, there is limited practical use for unconditional text generation: we expect language to relate to some identifiable, extrinsic meaning. When a system communicates information to an individual in natural language, it will typically rely on a structured representation of that information. Consequently, generating text that faithfully conveys structured data is an important goal in NLP, where inputs can take the form of tables (ToTTo, Parikh et al., 2020), RDF triples (e.g., WebNLG, Gardent et al., 2017), or Abstract Meaning Representations (AMR, Flanigan et al., 2016). NLP datasets in this domain consists of pairs of structured data (e.g., `<henri_matisse, hasOccupation, artist>`) and a representation of that data in text ("Matisse is an artist.").

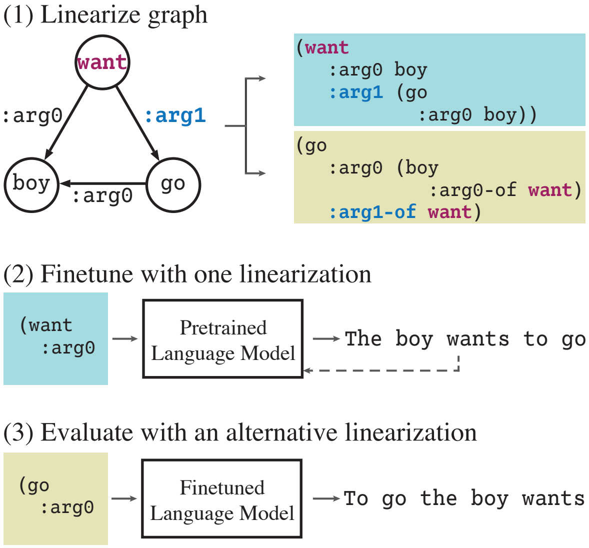

Figure 1: Diagram of our adversarial evaluation procedure for graph-to-text generation using pretrained language models (§3.2). (1) A graph can admit multiple possible linearizations. (2) Following standard practice, we train with a single linearization. (3) At evaluation time, we present the model with a meaning-preserving alternative.

Importantly, these types of inputs can be encoded as *graphs*. Accordingly, advances in neural architectures designed to explicitly encode graphs, such as graph neural networks (GNNs, Kipf and Welling, 2017) and graph transformers, have been used in these graph-to-text settings (Zhu et al., 2019; Zhao et al., 2020; Wang et al., 2020, to name a few). But graphs can also be represented as text (see top portion of Fig. 1). Hence, as an alternative to constraining a model architecture with a graph structure, it is also possible to simply *linearize* a graph into a string and train a sequence-to-sequence model from scratch (Pourdamghani et al., 2016; Konstas et al., 2017; Vinyals et al., 2015). Graph-

---

[*] Work undertaken during an internship at AI2.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 944–956
August 1–6, 2021. ©2021 Association for Computational Linguistics

```
(a / and
  :op1 (d / dream-01
      :ARG1 (f / film
          :ARG0-of (d2 / disturb-01))
      :ARG2-of (r / resemble-01
            :ARG1 a2))
  :op2 (a2 / and
      :op1 (f2 / fascinate-01
           :ARG0 f)
      :op2 d2))
```

(a) Canonical

```
(a / and
  :op1 (d / dream-01
      :ARG2-of (r / resemble-01)
      :ARG1 (f / film
          :ARG0-of (f2 / fascinate-01)
          :ARG0-of d2))
  :op2 (a2 / and
      :op2 (d2 / disturb-01)
      :op1 f2
      :ARG1-of r))
```

(b) Reconfigured

```
(r / resemble-01
  :ARG2 (d / dream-01
      :op1-of (a / and
            :op2 a2)
      :ARG1 (f / film))
  :ARG1 (a2 / and
       :op1 (f2 / fascinate-01
           :ARG0 f)
       :op2 (d2 / disturb-01
           :ARG0 f)))
```

(c) Randomized

Figure 2: Three PENMAN-based linearizations of AMR graphs corresponding to the sentence, "The film is a dream and, like a dream, is both fascinating and disturbing." Note that the **bolded** relation in the graph, `(resemble-01 :ARG1 and)`, is represented differently depending on the linearization.

based encoders were introduced because they outperformed these sequence-to-sequence models. Recently, however, there has been a stark reversal: graph-encoder generation performance has been *far surpassed* by pretrained transformer language models (LMs) finetuned on pairs of linearized graphs and their corresponding surface realizations (Mager et al., 2020; Kale and Rastogi, 2020; Harkous et al., 2020; Ribeiro et al., 2020, henceforth termed *pretrained linearized models*). Moreover, both automated and human assessments indicate that text generated with LMs retains meaning at least as well as graph-encoding baselines (Mager et al., 2020).

This is not the sole product of pretrained models' general language knowledge: Mager et al. (2020), using a GPT-2-based (Radford et al., 2019) model, report that ablating structural graph information (e.g., edges) in the linearized representation notably degrades generation performance, particularly in AMR-to-text tasks. The remarkable performance of pretrained linearized models is intriguing: explicit representation of the input graph by way of the model architecture appears to be well-substituted by simply writing the graph as a linear sequence.

In this work, we further investigate the extent to which pretrained models can leverage linearized graph inputs. Focusing on AMR graphs and sets of RDF triples in English-language datasets, we structure our investigation by first testing whether models' encodings are invariant to the linearization strategy—the way in which a graph is traversed and encoded when producing the linearized representation (see Figure 1). We discover that generation suffers under random permutations of the linearization, and embrace a simple-but-effective training strategy to mitigate this problem: adversarial training (Goodfellow et al., 2015). Motivated by this finding, we encourage more faithful encodings of graph structure via denoising objectives in the more complex AMR setting. This multi-task *scaffolding* (Swayamdipta et al., 2018) reveals that straightforward masking of the graph input is sufficient to improve generation quality in low resource settings.[1]

Moreover, when treating this denoising performance as a proxy for the quality of models' implicit graph encoding, we find that it correlates with the semantic fidelity of the resulting generation better than reasonable alternatives, suggesting possibilities for future evaluation metrics.

We organize our investigation around two research questions:

RQ1 To what extent are pretrained linearized models invariant to graph linearization strategy? (§3)

RQ2 Does encouraging pretrained linearized models' implicit graph representation lead to better generation? (§4)

2 Background: Graph-to-Text Generation

In a graph-to-text setting, we transduce graph inputs g to their corresponding surface realization $y = \langle y_1, \ldots, y_N \rangle$ via a parameterized probabilistic model $p_\theta(\cdot)$. In linearized models specifically, the graph g is first mapped to text by way of a (usually deterministic) linearization function $x = l(g)$, where $p_\theta(\cdot)$ is an off-the-shelf sequence-to-sequence model. This leads to the likelihood objective: $p_\theta(y \mid g) = \prod_{i=1}^{N} p_\theta(y_i \mid x, y_{1:i-1})$. When $p_\theta(\cdot)$ is an autoregressive pretrained transformer, generation quality far exceeds architectures with encoders specifically engineered to encode graphs (Mager et al., 2020; Kale and Rastogi, 2020; Harkous et al., 2020; Ribeiro et al., 2020).

---

[1] Implementation available at `github.com/ahoho/transformers/tree/graph-promotion`

	N	**Dev. ppl.**	**Avg. edges**
LDC2017T10	36k	21.1	11.4
WebNLG	18k	9.2	3.0

Table 1: Dataset statistics. Perplexity estimated on the development set with GPT-2 (Radford et al., 2019) fine-tuned on the training data using default hyperparameters in the `transformers` library (`gpt-2` model, Wolf et al., 2020).

Graph-to-Text Generation Datasets We explore two datasets for generation from a graph structure to English text.

Abstract Meaning Representation (AMR, Banarescu et al., 2013) is a formalism intended to represent the propositional meaning of utterances—"who is doing what to whom"—using graphs that have minimal dependence on the surface form. AMR graphs are directed and acyclic with a single "top" node (Goodman, 2020). They can be represented as either a graph, a tree, or sets of triples (van Noord and Bos, 2017). For our data, we use the AMR 2.0 release (LDC2017T10),[2] both because it spans a varied set of domains and styles, and because of its extensive use in prior work.

A simpler graph-to-text problem involves converting a set of RDF triples to natural text realizations of the information contained in the set, exemplified by the WebNLG dataset (Gardent et al., 2017). WebNLG pulls information from an existing knowledge base (DBPedia, Mendes et al., 2012) for a specific subset of 15 categories (e.g., "astronaut"). To generate the paired sentences, crowdworkers verbalize individual triples. Then, for examples consisting of multiple triples, they merge already-annotated sentences and apply minimal changes (leading to reduced sentence complexity relative to AMR; see perplexity scores in Table 1). There can be multiple surface realizations per input.

Models To study pretrained linearized models' invariance to graph linearization, we use T5 (Raffel et al., 2020), an *encoder-decoder* transformer (Vaswani et al., 2017) that has led to state-of-the-art generation on AMR (specifically, LDC2017T10) and WebNLG (Kale and Rastogi, 2020; Ribeiro et al., 2020).

We modify the T5 implementation from the `transformers` library (Wolf et al., 2020).[3] We use the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 0.0001, selected from the set $\{0.001, 0.0001, 3 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-6}\}$ after tuning on 1000 training examples across five random seeds.[4] We train until development set BLEU has not improved for 10 epochs. See Appendix A.1 for further details.

Evaluation Measures As a primary metric, we evaluate generated text using BLEU (Papineni et al., 2002), calculated with `SacreBLEU` (Post, 2018). Despite its limitations in generation settings, BLEU still generally accords with rankings of models, either by human evaluations or by alternate metrics (Manning et al., 2020). We also evaluate our scaffolding models (§4) using BertScore (Zhang et al., 2020), which measures token similarity with contextual embeddings, permitting a more nuanced measure of semantic similarity. Lastly, we use the $\mathcal{M}$ portion of the $\mathcal{MF}$-score (Opitz and Frank, 2020), which measures how well the source AMR graph can be reconstructed from the generated target sentence using an off-the-shelf parser. Unlike BLEU, which applies corpus-wide, this metric provides a best-guess at sentence-level accuracy.

3 RQ1: Robustness to Permutation of Graph Linearization

In this section, we explore the extent to which pretrained linearized models are invariant to the particular method used to linearize the input graph. Motivated by the strong graph-to-text performance of these models, we ask: do they implicitly develop a robust internal encoding of the input graph? Whereas a GNN-based model has an architecture designed for graph representation (e.g., information flows between adjacent nodes in a message-passing update), a linearized model must infer how connections are specified in a sequence during training.

If linearized models do form a representation, then their estimates of the target sentence should be invariant to an alternative linearization of the same graph, so long as the original linearization is in principle recoverable from this alternative. If a model meets this criterion, we call it **linearization-invariant**.

---

[2] `catalog.ldc.upenn.edu/LDC2017T10`

[3] We use T5-Base for WebNLG and T5-Large for AMR, finding that the larger model did not benefit the WebNLG task.

[4] Less extensive experiments with the full dataset indicated the same optimal setting, although in general it is relatively robust to learning rate.

3.1 Experimental Setup

To better understand models' graph-encoding behavior, we experiment with adversarial linearization strategies in two graph-to-text settings.[5]

Permutations of AMR-Graph Linearizations Standard AMR corpora are linearized as spanning trees over the graphs in PENMAN notation (Matthiessen and Bateman 1991, see Fig. 2a). In the present work, we also linearize graphs using PENMAN, doing so for several reasons: (1) it is sufficiently flexible to accommodate significant changes to the linearization, discussed below; (2) it is more concise than sets of directed triples, both reducing training time and ensuring that inputs fit in the transformer context window; (3) the format leads to superior generation over reasonable alternatives, e.g., DFS traversal paths (Mager et al., 2020).

We will refer to the human-created linearizations in AMR corpora as CANONICAL, since annotators follow a standardized process. There is evidence that this format, in particular the relative ordering of edge types, leaks information about the associated sentence order (Konstas et al., 2017). We speculate that overparametrized models may overfit to such correlations rather than develop robust implicit graph encodings, since it has been repeatedly reported that large models use dataset shortcuts (Jia and Liang, 2017; Gururangan et al., 2018; Geva et al., 2019, among others).

As an alternative linearization, Goodman (2020) defines the RECONFIGURE operation as creating a tree from an AMR graph, where order information from the canonical linearization is ignored, except for the top node (e.g., `and` in Figs. 2a and 2b). Although it is not a labeled element in the graph, the top node conveys structural information about the sentence—for instance, it is often the main verb. Reconfiguration can include reversals of edge labels (e.g., `ARG0` to `ARG0-of`), therefore constituting a substantive change to the linearization.

We also experiment with a more drastic restructuring of the graph, where we construct a tree from a RANDOMIZED triple set alone, disregarding all order information from the canonical format (Fig. 2c). Since it remains a valid traversal of the graph, in principle a model should be able to use this information to construct the surface sentence.

We parse, reconfigure, and randomize graphs using the `Penman` library (Goodman, 2020),[6] then replace variable names with their references and remove word sense information, following Ribeiro et al. (2019).

Permutations of RDF-Triple Linearizations We follow the procedure of Ribeiro et al. (2020) to form our standard linearization: we prepend a special token to each element of the triple, and separate triples with another dedicated token. For the output sentence "Ned is the father of Rod and Todd," we would have:

In: (Ned *fatherOf* Rod), (Ned *fatherOf* Todd)
Out: `<rel> <S> Ned <V> father of <O> Rod`
`<rel> <S> Ned <V> father of <O> Todd`

For our adversarial permutation, we RANDOMIZE the ordering of the triples.

Encouraging Robustness to Linearization We train additional models with the goal of encouraging an agnosticism to graph linearization strategy. We adopt an adversarial training approach (Goodfellow et al., 2015), and alter the graph linearization presented to the model at each epoch. We argue that this scheme ought to reduce any model dependence on the human-derived annotation.

3.2 Robustness Results

For both tasks, we train the model on the canonical linearization, then evaluate on the various linearizations described in Section 3.1.

Impact of Adversarial Linearizations The CANONICAL columns of Table 2 show results for models trained on that linearization, then evaluated on permuted graph linearizations. We note a strong negative impact in models' generation capacity for both tasks, with a starker decrease for the AMR data. These results suggest that pretrained linearized models are not linearization-invariant, failing to learn robust implicit graph representations, even in the case of the much simpler WebNLG data.

The remaining columns of Table 2 show that our straightforward adversarial training technique improves robustness, with only minor cost to generation performance. This is the case even with the more drastic RANDOMIZED AMR linearization. Moreover, it only incurs a minor training time cost—for AMR, the CANONICAL, RECONFIGURE,

---

[5] Although "adversarial" can imply inputs specifically designed to break a model, here we use it to mean that inputs are merely *likely* to cause issues by diverging from the training order. In addition, we intend to draw parallels to adversarial training (Goodfellow et al., 2015).

[6] `github.com/goodmami/penman`

	AMR (LDC2017T10)			WebNLG			
				Seen		Unseen	
Train linearization→	CANON.	RECON.	RANDOM.	CANON.	RANDOM.	CANON.	RANDOM.
Eval. linearization↓							
CANONICAL	43.52	43.08	40.90	62.56	62.55	44.73	45.09
RECONFIGURED	33.27 / 76%	41.13 / 95%	40.33 / 99%	–	–	–	–
RANDOMIZED	22.89 / 53%	31.00 / 72%	39.80 / 97%	54.00 / 86%	59.40 / 95%	39.23 / 88%	42.35 / 94%
GNNs	–	–	–	–	–	–	–
Wang et al. (2020)	28.80	–	–	–	–	–	–
Zhao et al. (2020)	–	–	–	64.42	38.23	–	–

Table 2: BLEU under different linearizations, using T5-LARGE (AMR, development set) and T5-BASE (WebNLG, for both "seen" and "unseen" test sets). Percentages represent the decrease from the CANONICAL representation following the adversarial evaluation (i.e., they should be read columnwise).

and RANDOMIZE variants attain 40 BLEU at 2, 3, and 5 epochs, respectively.

Given that elements of canonical annotations are known to correlate with the target sentence order (Konstas et al., 2017), we do not find it surprising that the models trained *and* evaluated on the permuted linearizations show decreased performance. However, it is meaningful that the canonical linearization at evaluation time still leads to the best results, even for models trained with the randomized inputs—these models did not learn to associate the canonical ordering signal with the input graph. One possible explanation is that the earlier pretraining induces a sensitivity to input token order that persists despite the adversarial fine-tuning, but the behavior merits further exploration.

In addition, note that the canonical order does not have an explicit formal definition, and may require heuristic reverse engineering; in a real-world system, recreating a canonical graph linearization (e.g., from a knowledge-base query) for a subsequent graph-to-text model may not be straightforward. These results show it is possible to inoculate models to accommodate this sort of use case.

4 RQ2: Better Implicit Graph Encodings with Text-to-Text Scaffolding

The positive results of our adversarial training procedure (§3.2) suggest that pretrained linearized models can form a robust internal graph representation, even though they rely on linearized inputs. Under substantively different linearizations, models retain the ability to generate accurately (even the RANDOMIZE model outperforms best-in-class graph transformers; Wang et al. 2020).

Prior work, involving both GNNs and pretrained linearized models, has explored various ways of improving models' sensitivity to the structure of the input graph. To better maintain fidelity to the graph, previous graph-to-text methods incorporate additional loss terms, specialized architectures, or generation-time ranking to influence the semantic accuracy of generation: ranking outputs by the correctness of the AMR parse (Mager et al., 2020; Harkous et al., 2020), jointly "back-parsing" graphs when decoding (Bai et al., 2020), or using distinct components to model different graph traversals (Ribeiro et al., 2019).

These efforts suggest that explicitly accounting for graph structure can assist generation. Can we expand on this idea, and improve generation quality by inducing more robust internal graph representations? To answer this question, we propose secondary objectives designed to promote graph "awareness." In addition to the above graph-to-text approaches, we also draw inspiration from *denoising* methods used in language model pretraining (Raffel et al., 2020; Lewis et al., 2020), as well as syntactic *scaffolds* that support semantic tasks with an auxiliary syntax-dependent loss (Swayamdipta et al., 2018). Intermediate auxiliary pretraining has been repeatedly shown to be successful in other contexts (Phang et al., 2018; Li et al., 2019; Gururangan et al., 2020).

4.1 Experimental Setup

In particular, we propose unsupervised graph-denoising tasks that we train alongside AMR-to-text generation, following the multi-task setup of Raffel et al. (2020). For each batch, we either optimize the likelihood in Section 2 or one of the objectives described below.[7]

---

[7] Per-task batches proved marginally better than mixing within a batch. The scaffolding task probability is a hyperparameter, which we set to 0.5.

Masked Graph Modeling When training transformers to have wide-ranging natural language capabilities, unsupervised denoising objectives like masked language modeling have proven extremely successful (Devlin et al., 2019; Raffel et al., 2020). We argue that a similar principle ought to apply to graph understanding, and therefore apply masking directly to linearized graphs.

In masked language modeling, each word token is masked with probability 15%. Here, we mask different sets of tokens, depending on the experimental condition, always setting the probability such that 15% of *all* tokens will be masked. Specifically, we mask: all tokens in the linearized graph, the graph components alone (edge labels and parentheses), and the semantic nodes. We also experiment with standard masking of the surface sentence, which mirrors the unsupervised domain-adapted pretraining employed by Ribeiro et al. (2020).[8] For example, when masking components alone:

```
orig ( stupefy :ARG1 ( we )   )
  in ( stupefy <M>   ( we <M> )
 out original text
```

Graph masking can also be performed on any of the linearization variants defined in Section 3.1.[9]

Graph Reordering Building on our findings from Section 3.2, we introduce a *reordering* objective. Specifically, we provide the model with a RECONFIGURED or RANDOMIZED linearization, then task the model with reconstructing the canonical version. We suspect that learning this mapping requires that the model captures the graph structure better, leading to superior graph-to-text generation. Unlike the joint re-generation approach of Mager et al. (2020), where the input graph is copied alongside the target text, our method both requires a nontrivial encoding of the graph and has the effect of augmenting the data (due to the nondeterministic reconfiguration).[10]

4.2 Scaffolding Results

We find that, overall, denoising objectives drive substantial improvements over the baseline when training on the reduced $n = 1000$ dataset (Table 3). In fact, using less than 3% of the full data produces results that exceed that of state-of-the-art GNN models from less than two years prior to this writing (BLEU 27.37, Ribeiro et al., 2019). Moreover, the results suggest that focusing on the graph representation itself is most important: standard sentence masking (i.e., MLM-style) is less beneficial than graph masking, although it still outperforms the baseline. Surprisingly, the various graph-masking objectives perform similarly to one another—there is little benefit to more complex strategies that specifically account for the graph structure.

	BLEU
Baseline	24.33 (0.94)
Sentence masking (MLM)	27.73 (1.29)
Graph Masking	
All tokens	28.48 (0.90)
Components	28.49 (0.48)
Nodes	**29.56** (1.05)
w/ RECONFIGURED *input*	
All tokens	**29.41** (0.90)
Components	28.34 (0.58)
Nodes	28.77 (0.80)
Reordering to canonical	
From RECONFIGURED	28.27 (0.90)
From RANDOMIZED	28.29 (0.91)

Table 3: Development set BLEU across scaffolding objectives and baselines, trained on 1000-example subsets of the AMR dataset (LDC2017T10). Mean (s.d.) over 5 seeds.

	BLEU	BS	$\mathcal{M}$
Bai et al. (2020)	34.19	-	-
Ribeiro et al. (2020)	45.80	-	-
Baseline	44.51 (0.48)	77.40 (0.36)	76.53 (0.19)
Scaffolding			
Mask nodes	45.14 (0.23)	77.75 (0.13)	76.52 (0.14)
RECON., mask all	44.89 (0.39)	77.56 (0.26)	76.54 (0.20)
Reorder from RECON.	44.86 (0.19)	77.62 (0.16)	76.34 (0.17)

Table 4: Test-set results of scaffolding objectives and baselines trained on the *full* AMR dataset (LDC2017T10). Bai et al. (2020) is a state-of-the-art graph transformer. Ribeiro et al. (2020) finetunes T5-LARGE, which we re-implement as our baseline model. BS is BertScore (Zhang et al., 2020), and $\mathcal{M}$ is the meaning component of the $\mathcal{MF}$-score (Opitz and Frank, 2020). Mean (s.d.) over 5 seeds.

While the increased generation quality from the graph-denoising methods is not drastic relative to the MLM case, we contextualize our gains by noting that other ways of promoting greater graph awareness yield similar improvements in absolute terms—and come at the cost of greater model

---

[8] We use MASS-style masking (Song et al., 2019) for the tokens, rather than the span-replacing of T5, as it performed somewhat better.

[9] We restrict ourselves to the RECONFIGURE setting given that early results showed little difference from RANDOMIZE.

[10] Simultaneously generating the surface text and reordering to the canonical linearization did not improve results.

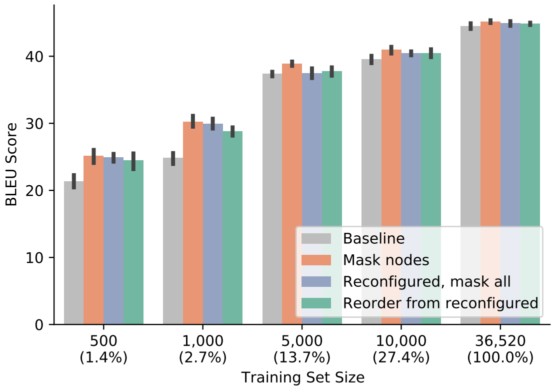

Figure 3: Test set BLEU on the AMR dataset (LDC2017T10) under different amounts of training data for selected scaffolding objectives (over 5 seeds). In low-data regimes, the scaffolding objectives substantially improve BLEU score over the baseline.

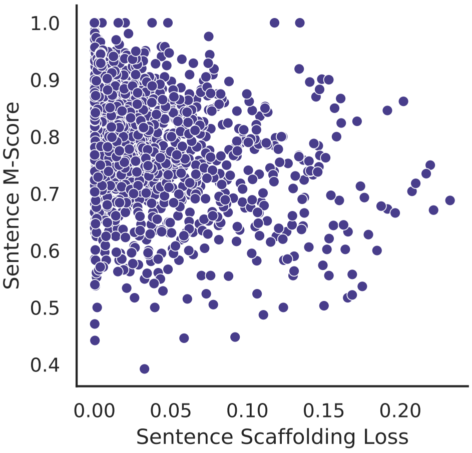

Figure 4: Sentence-level scaffolding loss and $\mathcal{M}$-score on the validation set, using a model trained with the *reordering-from-reconfigured* scaffold. $\mathcal{M}$-score is a measure of the generated sentence's semantic fidelity, and the scaffolding loss is a proxy for the graph encoding accuracy.

complexity or generation time. For instance, the use of two graph representations in Ribeiro et al. (2019) achieve a roughly 1-BLEU increase over the use of one alone.

Based on the findings from the $n = 1000$ setting (Table 3), we select three of the best-performing scaffolding objectives—*mask nodes*, *reconfigure & mask all tokens*, and *reorder from reconfigured*—and train them at $n \in \{500, 1000, 5000, 10000, N\}$. Results are shown in Fig. 3. At $n = 5000$, representing 14% of the data, the impact of scaffolding is no longer strong across all objectives. When evaluating on the full dataset, the difference is minor (Table 4). For both BLEU and BertScore, we observe slight improvement over the baseline on average for the *mask nodes* case, but it is within a standard deviation of the baseline (estimated over 5 seeds). $\mathcal{M}$-score does not vary between models, but it is also not yet established for fine-grained model selection. It appears that the increased size of the data supplants the need for scaffolding losses: the diversity of the source graphs encourages a graph-reasoning ability sufficient to generate accurate sentences. Of course, in a realistic application, hundreds or thousands of training examples are more attainable than tens of thousands. That such straightforward methods can yield strong gains is promising for future work in low-resource graph-to-text generation.

Manual Annotations In a manual analysis of 100 random model predictions, we generally observe broad agreement between the model trained with the *reordering-from-reconfigured* scaffold and the baseline (73% agreement in fidelity), both trained with the full dataset. However, in three cases, the baseline model fails to capture the order of arguments (e.g., "from y to x" when "from x to y" is correct), whereas the scaffolded model remains true to the graph (see Table 5; we did not observe instances of the reverse case). While we fail to observe "hallucinations"—material information that is not contained in the graph input—both models occasionally drop modifiers (e.g., adjectives or adverbs). Both models exhibit word-sense confusion (see the third row in Tab. 5, where "long [in length]" is substituted with "long [in duration]"). We presume this is due to the removal of word-sense suffixes during preprocessing to avoid sparsity issues (`long-03` → `long`).

4.3 Encoding Graphs and Generation Performance

The results of Section 4.2 show that the denoising scaffolds impact generation performance. If we consider the sentence-level scaffolding loss as a proxy for the quality of its implicit graph encoding, can it help explain generation fidelity? In order to determine this relationship, we quantify generation accuracy using the $\mathcal{M}$ component of the $\mathcal{MF}$-score (Opitz and Frank, 2020). It is calculated by first using an off-the-shelf parser to create an AMR graph from the generated target sentence, then by measuring the overlap with the gold source AMR (from 0 to 1)—this is equivalent to the "smatch"

Target	*Both Norway and Sweden have been spared violent terror acts but authorities in both countries have voiced concern about terrorists or terror financiers operating out of Scandinavia.*
Baseline	**Norwegian and Swedish authorities have spared Norway and Sweden** from violent acts of terror but have voiced concern about terrorists or financiers of terror operating out of Scandinavia.
Ours	Norway and Sweden have been spared terror acts of violence but Norwegian and Swedish authorities have voiced concern about terrorists or financiers of terror operating out of Scandinavia.
Target	*The 30-day simple yield fell to an average 8.19% from 8.22%; the 30-day compound yield slid to an average 8.53% from 8.56%.*
Baseline	The simple 30 day yield **fell to 8.22 percent from 8.19 percent** on average and the compound 30 day yield **slid to 8.56 percent from 8.53** percent on average.
Ours	Simple 30 day yields fell from 8.22 to an average 8.19% and compound 30 day yields slid from 8.56 to an average 8.53%.
Target	*Many young Saudi radicals have crossed the long and porous border between the Kingdom and Iraq and joined up with Sunni Muslim insurgents there.*
Baseline	Many young Saudi radicals have crossed the porous border **from Iraq to the Kingdom** and joined up with Sunni Islamic insurgents there.
Ours	Many young Saudi radicals have crossed the porous **long-term border** with Iraq and joined up with Sunni Islamic insurgents there.

Table 5: Selected predictions from the baseline and a model using the *reordering-from-reconfigured* scaffold (trained on the full data). **Colored text** denotes a semantically incorrect generation.

rescoring metric (Cai and Knight, 2013). As seen in Fig. 4, there is a substantial negative relationship (Pearson's $\rho = -0.35^*$) between these two variables, measured using outputs from the model trained with the *reordering-from-reconfigured* scaffold on the full data.

To fully operationalize the above question, we estimate a linear regression on the $\mathcal{M}$ score of predicted sentences from the validation set. In this scenario, the linear regression can quantify how much variation in predicted sentences' semantic fidelity (measured by the $\mathcal{M}$-score) can be explained by model components and target sentence characteristics. If the coefficient for the sentence-wise scaffolding loss is significant (and negative), then this would suggest that there is a relationship between the scaffolding objective and the predicted sentence's semantic fidelity.

As covariates, we include the above (logged) scaffolding loss, in addition to other metrics that have a significant independent correlation with generation quality. In particular, we use sentence-BLEU, the number of edges in the graph, graph re-entrancies, words in the target sentence, and the (also logged) sentence generation loss.[11]

We use the Bayesian information criterion (BIC) to select the model from all possible combinations of the above covariates. We find that the preferred model with p covariates, $p \in \{1, \ldots, 6\}$, includes the reordering loss in all but one case ($p = 2$), suggesting its validity as an indicator of graph fidelity above and beyond other alternatives. As seen in Table 6, it has a significant negative relationship with the $\mathcal{M}$ score, larger than that of the comparably-scaled generation loss. These results indicate that the reordering loss captures important information about the quality of the graph encoding.

X	β
Intercept	0.7590*
Scaffolding loss (log)	-0.0094*
Generation loss (log)	-0.0088*
BLEU/100	0.0628*
Words in target	-0.0021*
BIC	-2378
Adj. R^2	0.267

Table 6: OLS regression results on validation sentence $\mathcal{M}$-score, a measure of semantic fidelity that relies on the gold AMR graph. These results indicate that the scaffolding loss explains a significant amount of the variation in the semantic fidelity of the generated sentence to the gold target. Model trained with the *reordering-from-reconfigured* scaffold. *Significance at $p < 0.001$.

5 Related Work

Pretrained Transformers for Graph-to-Text Generation Mager et al. (2020) condition GPT-2 (Radford et al., 2019) on a linearized AMR graph, then fine-tune on the corresponding surface representation text. Later work using transformers has also found success on both AMR-to-text and data-to-text tasks (Kale and Rastogi, 2020; Harkous et al., 2020; Ribeiro et al., 2020). To our knowl-

---

[11] We eliminate outliers consisting of the bottom 0.5% of target lengths and $\mathcal{M}$-scores and the top 0.5% of the losses.

edge, across a diverse set of tasks and automated[12] metrics, a pretrained transformer of sufficient capacity will always outperform a specialized GNN, often by a large margin. Ribeiro et al. (2020), following Gururangan et al. (2020), further pretrain on additional in-domain data, using both supervised (silver AMR parses to text) and unsupervised (denoising target text) objectives.

Graph-Dependent Losses Mager et al. (2020) use various heuristics to improve fidelity. During training, they regenerate the input graph, and in inference, they parse generations and rank their consistency with the original graph. Harkous et al. (2020) instead rank with a trained classifier, and introduce additional "state embeddings" to help indicate the ordering of graph components. The encoder-decoder methods cited in the previous paragraph eschew these approaches and nonetheless perform better. In preliminary replications of the Mager et al. experiments with T5, we find that joint re-generation leads to no improvement (moreover, the longer output sequences increase training time). Experimenting with other graph-sensitive embeddings is a valuable direction for future work.

Graph Linearization Other work also studies linearizations for AMR-to-text settings. As opposed to our efforts, the focus is not on enriching or measuring models' graph encoding, but instead on determining what elements of linearization (e.g., edge labels) are necessary for generation.

Closest to our work is Konstas et al. (2017), who experiment with alternative graph traversals by randomizing the edge type order (less drastic than either RECONFIGURE or RANDOMIZE) with an LSTM-based model. Rather than randomizing at each epoch, as in our approach, they employ a *consistent* random ordering for each example during training, and do not evaluate models across different linearizations. The results help establish that LSTMs can be made agnostic to ordering, but fail to measure the extent to which models overfit to the training order (Section 3.2).

Ribeiro et al. (2020) report paired training and evaluation shuffling results (as in Table 2), but they ignore parentheses, only reodering node labels. Hence, their results cannot establish models' graph-encoding ability, instead revealing that node order is informative of word order, corroborating findings in Konstas et al. (2017). Both works, along with Mager et al. (2020), run ablations by removing parenthetical markers, finding that graph structure is necessary for strong generation.

Finally, Kedzie and McKeown (2020), appearing contemporaneously to our work, seek to control the output generation by manipulating the input linearization order, using a randomization similar to ours as an "uncontrolled" baseline. Given their focus on task-oriented dialogue planning, which uses simpler meaning representations and sentences than the AMR dataset used here (i.e., shallower graphs and limited domains), we view their work as complementary to our own.

6 Conclusion

In this work, we explore the graph-encoding ability of pretrained transformers through the lens of graph-to-text generation that relies on linearized graph inputs. First, we determine the extent to which these models are invariant to the method by which graphs are linearized, finding that models trained on the fixed, canonical linearizations fail to generalize to meaning-preserving alternatives. We rectify this shortcoming by training models on linearizations corresponding to alternative random traversals of the graph. Following prior work that has used graph-aware losses to improve generation quality, we then explore ways of improving models' sensitivity to the input graphs. Motivated by the success of denoising objectives in other text-to-text settings, we encourage robust internal graph encodings through additional scaffolding losses. Although scaffolding leads to tepid improvements in generation quality when training data is plentiful, it yields substantial gains in low-resource settings. Finally, while pretrained transformers may not learn the entirety of a graph's structure via scaffolding, our text-to-text methods take a step in the direction of increasing graph sensitivity.

7 Acknowledgments

We thank AI2 AllenNLP team members for advice throughout the project, and anonymous reviewers for their insightful comments. We also thank Leonardo Ribeiro for his publicly available implementations and assistance.

---

[12] Human evaluation has been less thorough, although Mager et al. (2020) report improved human judgments on AMR-to-text generation. We note similar results in our own experiments.

References

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2020. Online back-parsing for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1206–1219, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2020. Controllable meaning representation to text generation: Linearization and data augmentation strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185, Online. Association for Computational Linguistics.

Thomas Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story ending prediction by transferable bert. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1800–1806. International Joint Conferences on Artificial Intelligence Organization.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Emma Manning, Shira Wein, and Nathan Schneider. 2020. A human evaluation of AMR-to-English generation systems. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4773–4786, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Christian M.I.M. Matthiessen and John A. Bateman. 1991. *Text Generation and Systemic-functional Linguistics: Experiences from English and Japanese*. Communication in Artificial Intelligence Series. Pinter Pub Ltd.

Pablo Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1813–1817, Istanbul, Turkey. European Language Resources Association (ELRA).

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with Abstract Meaning Representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Juri Opitz and Anette Frank. 2020. Towards a decomposable metric for explainable evaluation of text generation from amr. arXiv:2008.08896.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. arXiv:1811.01088.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for Graph-to-Text generation. arXiv:2007.08426.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4596–4604, Stockholm, Sweden. PMLR.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5926–5936, Long Beach, California, USA. PMLR.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28, pages 2773–2781.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang, V. Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations (ICLR)*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online. Association for Computational Linguistics.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.

X	β
Intercept	0.2201*
Scaffolding loss (log)	$-0.0080^{\dagger}$
Generation loss (log)	$-0.0055^{\dagger}$
BLEU/100	1.5916*
Words in target	-0.0001
BIC	-2740
Adj. R^2	0.058

Table 7: OLS regression results on validation sentence $\mathcal{M}$-score, a measure of semantic fidelity that relies on the gold AMR graph. These results indicate that the scaffolding loss explains a significant amount of the variation in the semantic fidelity of the generated sentence to the gold target. Model trained with the *reordering-from-reconfigured* scaffold using 1000 training examples. *Significance at $p < 0.001$. † Significance at $p < 0.01$.

A Appendix

A.1 Hyperparameters, Computing Infrastructure, and Data Processing

We set the batch size to 32 for WebNLG, and 6 for AMR. During decoding, we use a beam size of 10 for WebNLG and 5 for AMR, following Ribeiro et al. (2020). All models were trained with a NVIDIA Quadro RTX 8000 GPU.

To maintain a reasonable batch size, long AMR inputs are truncated to 354 tokens (affecting less than 1% of examples); this step is not necessary for WebNLG. Finally, during preprocessing of the LDC2017T10 data, we differ from Ribeiro et al. (2019) and remove the `:wiki` attribute from the graph[13] both because they are not a standard component of all AMR graphs and because they increase the length of the inputs. We suspect that this change is the reason our baseline achieves slightly different results than Ribeiro et al. (2020) (Table 4).

A.2 Regression Results at $n = 1000$

In Table 7, we show OLS regression results analogous to those in Table 6 for one run of the $n = 1000$ case. While the fit is lower (a lower R^2), the scaffolding loss still has an explanatory effect.

---

[13] For some named entities, the graph will include a corresponding Wikipedia title.

Predicting cross-linguistic adjective order with information gain

William Dyer
Oracle Corporation
william.dyer@oracle.com

Richard Futrell
University of California, Irvine
rfutrell@uci.edu

Zoey Liu
Boston College
ying.liu.5@bc.edu

Gregory Scontras
University of California, Irvine
g.scontras@uci.edu

Abstract

Languages vary in their placement of multiple adjectives before, after, or surrounding the noun, but they typically exhibit strong intra-language tendencies on the relative order of those adjectives (e.g., the preference for 'big blue box' in English, 'grande boîte bleue' in French, and 'alṣundūq al'azraq alkabīr' in Arabic). We advance a new quantitative account of adjective order across typologically-distinct languages based on maximizing information gain. Our model addresses the left-right asymmetry of French-type ANA sequences with the same approach as AAN and NAA orderings, without appeal to other mechanisms. We find that, across 32 languages, the preferred order of adjectives mirrors an efficient algorithm of maximizing information gain.

1 Introduction

Languages that allow multiple sequential adjective modifiers tend to exhibit strong tendencies on the relative order of adjectives, as in 'big blue box' vs. 'blue big box' in English (Dixon, 1982). To date, most of the research on adjective ordering has focused on preferences in pre-nominal languages like English where adjectives precede the modified noun (Futrell et al., 2020a), or in post-nominal languages like Arabic where adjectives follow the noun (Kachakeche and Scontras, 2020). This research usually posits a metric, such as information locality (Futrell et al., 2020b) or subjectivity (Scontras et al., 2017), which governs the preferred distance between a noun and its adjectives. Because these theories predict only the relative linear *distance* between noun and adjective, they cannot be straightforwardly applied to mixed languages like French, where adjectives regularly appear both before and after the modified noun, at least not without added assumptions about hierarchical distance (Cinque, 1994). Instead, these mixed languages are often modeled with constraints on which adjective classes or functions can appear before or after a noun (Cinque, 2010; Fox and Thuilier, 2012).

Traditional accounts of adjective ordering in the linguistics literature often assume a **tree structure** in which the target measure is the hierarchical distance from noun (N) to adjective (A). According to syntactic accounts, ordering regularities are predicted by a universal hierarchy of lexical semantic classes (e.g., color adjectives are hierarchically closer to the modified noun than size adjectives; Cinque, 1994; Scott, 2002). Alternative accounts use aspects of adjective meaning to predict adjective order, making appeal to notions like 'inherentness' (Whorf, 1945) or 'definiteness of denotation' (Martin, 1969). Recently, Scontras et al. (2017) provide experimental evidence that their synthesis of semantic predictors into a continuum based on subjectivity reliably predicts ordering preference in English; followup studies have found subjectivity to be a reliable predictor in other languages as well (Tagalog: Samonte and Scontras, 2019; Mandarin: Shi and Scontras, 2020; Arabic: Kachakeche and Scontras, 2020; Spanish: Rosales Jr. and Scontras, 2019; Scontras et al., 2020). Explanations for the role of subjectivity in adjective ordering show how subjectivity-based orderings are more **efficient** than alternative orderings, thereby maximizing communicative success (Simonič, 2018; Hahn et al., 2018; Franke et al., 2019; Scontras et al., 2019).

Other efficiency-based approaches to adjective order quantify efficiency with information-theoretic measures of word distributions such as surprisal or entropy (Cover and Thomas, 2006; Levy, 2008). Models in this vein have a long conceptual history in the field, originating with the idea that semantic closeness between words is reflected in syntactic closeness in a surface realization (Sweet, 1900; Jespersen, 1922; Behaghel,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 957–967
August 1–6, 2021. ©2021 Association for Computational Linguistics

1932). Modern quantitative incarnations include integration cost (Dyer, 2017) and information locality (Futrell et al., 2020b), both generalizations of the widely-accepted principle of dependency distance minimization (Liu et al., 2017; Temperley and Gildea, 2018).

Crucially, while previous approaches are able to model symmetrical structures within the noun phrase, as in the mirror-image $A_1 A_2 N$ orders of English and the $N A_2 A_1$ orders of Arabic, a hierarchical approach cannot model the **left–right asymmetry** of Romance $A_1 N A_2$ without an appeal to other, usually syntactic, mechanisms (Cinque, 2009, 2010).

We advance an information-theoretic factor that predicts adjective ordering across the three typological 'templates' of adjective order—pre (AAN), mixed (ANA), and post (NAA)—based on **information gain** (IG), a measure of the reduction in uncertainty attained by transforming a dataset. IG is used in machine learning for ordering the nodes of a decision tree (Quinlan, 1986; Norouzi et al., 2015), where nodes are most often ordered in a greedy fashion such that the information gain of each node is maximized. By analogy, we view the noun phrase as a decision tree for reducing a listener's uncertainty about a speaker's intended meaning. Each word acts as a node in the decision tree; preferred adjective orders thus reflect an efficient ordering of nodes.

2 Empirical background

Empirical investigations of adjective ordering have focused on the cross-linguistic stability of these preferences across a host of unrelated languages (e.g., Dixon, 1982; Hetzron, 1978; Sproat and Shih, 1991). For example, where English speakers prefer 'big blue box' to 'blue big box', Mandarin speakers similarly prefer *dà-de lán-de xiāng-zi* 'big blue box' to *lán-de dà-de xiāng-zi* 'blue big box' (Shi and Scontras, 2020). In post-nominal languages, we find the mirror-image of the English pattern, such that adjectives that are preferred closer to the noun in pre-nominal languages are also preferred closer to the noun in post-nominal languages.[1] For example, speakers of Arabic prefer *alṣundūq al'azraq alkabīr* 'the box blue big' to *alṣundūq alkabīr al'azraq* 'the box big blue'.

In support of the cross-linguistic stability of adjective ordering preferences, Leung et al. (2020) present a latent-variable model capable of accurately predicting adjective order in 24 languages from seven different language families, achieving a mean accuracy of 78.9% on an average of 1335 sequences per language. Importantly, the model succeeds even when the training and testing languages are different, thus demonstrating that different languages rely on similar preferences. However, Leung et al.'s study was limited to AAN and NAA templates. There has been very little corpus-based empirical work on ordering preferences in the mixed ANA template, where adjectives both precede and follow the modified noun.[2]

While Leung et al. (2020) learn adjective order by training on observed adjective pairs, an alternate strategy is to posit one or more a priori metrics as an underlying motivation for adjective order (e.g., Malouf, 2000, in part). This approach allows for the study of why adjective orders might have come about. To that end, Futrell et al. (2020a) report an accuracy of 72.3% for English triples based on a combination of subjectivity and information-theoretic measures derived from the distribution of adjectives and nouns.

One of the information-theoretic measures analyzed by Futrell et al. (2020a) is an implementation of information gain based on the partitioning an adjective performs on the space of possible noun referents. However, it is unclear how this formulation of information gain could be implemented for post-nominal adjectives, in which the noun has presumably already been identified. Instead, the current study implements information gain based on feature vectors, as outlined in §3.

To our knowledge, the current study is the first attempt at predicting adjective order across all three templates, with an eye not only to raw accuracy, but in hopes of illuminating the functional pressures which might contribute to word ordering preferences in general. While we acknowledge that multiple factors are likely involved in adjective order preferences, our contribution here is a single quantitative factor capable of predicting adjective order across typologically distinct languages.

---

[1] Celtic languages have been claimed to be an exception to this trend (Sproat and Shih, 1991), though our own investigations into Irish suggest that it behaves like other post-nominal languages, at least with respect to information gain.

[2] We note three empirical studies that have examined the placement of a single adjective or adjective phrase before or after the noun in Romance languages: Thuilier (2014), Gulordava et al. (2015) and Gulordava and Merlo (2015). However, these studies do not tackle the question of order preferences among ANA triples.

3 Information gain

3.1 Picture of communication

We assume that a speaker is trying to communicate a **meaning** to a listener, with a meaning represented as a binary vector, where each dimension of the vector corresponds to a feature. Multiple features can be true simultaneously. For example, a speaker might have in mind a vector like $m_1 = [111\ldots0]$ in Figure 1, where the vector has value 1 in the dimensions for 'is-big' (f_0), 'is-grey' (f_1), and 'is-elephant' (f_2), and 0 for all other features. A meaning of this sort would be conveyed by the noun phrase 'big grey elephant'. We call m a **feature vector** and the set of feature vectors M.

The listener does not know which meaning m the speaker has in mind; the listener's state of uncertainty can be represented as a probability distribution over all possible feature vectors, $P(m)$, corresponding to the prior probability of encountering a given feature vector. We call this distribution the **listener distribution** L.

By conveying information, each word in a sequence causes a change in the listener's prior distribution. Suppose as in Figure 1 that a listener starts with probability distribution L, then hears a word w conveying a feature (f_2), resulting in the new distribution L'. The amount of change from L to L' is properly measured using the Kullback–Leibler (KL) divergence $D_{\text{KL}}[L'||L]$ (Cover and Thomas, 2006). Therefore, the divergence $D_{\text{KL}}[L'||L]$ measures the amount of information about meaning conveyed by the word.

Another measure of the change induced by a word is the information gain, an extension of KL divergence to include the notion of negative evidence. Let $\bar{L}'$ represent the listener's probability distribution over feature vectors conditional on the negation of w. By taking a weighted sum of the positive and negative KL divergence, we recover **information gain** (Quinlan, 1986):

$$\text{IG} = \frac{|L'|}{|L|} D_{\text{KL}}[L'||L] + \frac{|\bar{L}'|}{|L|} D_{\text{KL}}[\bar{L}'||L], \quad (1)$$

where $|L|$ indicates the number of elements in the support of L with non-zero probability. Information gain represents the information conveyed by a word and also the information conveyed by its negation.

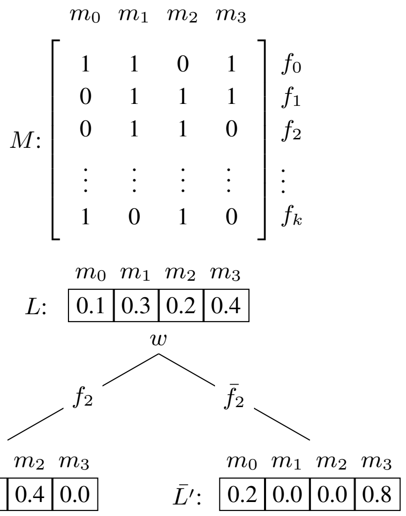

Figure 1: A toy universe composed of four feature vectors m defined by k binary features f and an associated probability distribution L. Partitioning L on f_2 yields L', the probability distribution of the feature vectors containing a 1 for f_2, viz. m_1 and m_2, as well as $\bar{L}'$, the distribution of feature vectors containing a 0 for f_2, or $\bar{f}_2$.

3.2 Relationship to other quantities

Our IG quantity in Eq. 1 is drawn from the ID3 algorithm for generating decision trees (Quinlan, 1986). The goal of ID3 is to produce a classifier for some random variable (call it L) which works by successively evaluating some set of binary features in some order. The optimal order of these features is given by greedily maximizing information gain, where information gain for a feature f is a measure of how much the entropy of L is decreased by partitioning the dataset into positive and negative subsets based on whether f is present or absent. Our application of information gain to word order comes from treating each word as a binary indicator for the presence or absence of the associated feature, and then applying the ID3 algorithm to determine the optimal order of these features.

The first term of Eq. 1, the divergence $D_{\text{KL}}[L'||L]$, measures the amount of information about L conveyed by the word w and has been the subject of a great deal of study in psycholinguistics. In particular, Levy (2008) shows that if the word w and the context c can be reconstructed perfectly from the updated belief state L', then the amount of information conveyed by w reduces to the **surprisal** of word w in context c:

$$D_{\text{KL}}[L'||L] = -\log p(w|c). \quad (2)$$

Importantly for our purposes, the positive evidence term $D_{\mathrm{KL}}[L'||L]$ in Eq. 1 alone is unlikely to make useful predictions about cross-linguistic ordering preferences, because surprisal is invariant to reversal of word order across a language as a whole (Levy, 2005; Futrell, 2019): the same surprisal values would be measured for any given language and a language with all the same sentences in reverse order. As such, these metrics are unable to predict any a priori asymmetries in word-order preferences between pre- and post-nominal positions.

3.3 Negative evidence

The new feature of information gain, which has not been presented in previous information-theoretic models of language, is the negative evidence term in $D_{\mathrm{KL}}[\bar{L}'||L]$, indicating the change in the listener's belief about L given the negation of the features indicated by word w, a quantity related to extropy (Lad et al., 2015). For example, consider *académie*/NOUN *militaire*/ADJ 'military/ADJ academy/NOUN' in French. Let L represent a listener's belief state after having heard the noun *académie* 'academy'. Upon hearing the adjective *militaire* 'military', L is partitioned into L'—the portion of L in which *militaire* is a feature—and $\bar{L}'$, the portion of L in which *militaire* is not a feature. Put another way, $\bar{L}'$ is the probability distribution over non-military academies.

The negative evidence portion of information gain is of primary interest to us because it breaks the symmetry to word-order reversal that we would have if we used the positive evidence term alone. That is, because the sum of surprisals of words w_1 and w_2 in the context of w_1 is the log joint probability of a sequence:

$$-\log p(w_1) - \log p(w_2|w_1) = -\log p(w_1, w_2), \tag{3}$$

the sum of w_2 and w_1 in the context of w_2 necessarily yields the same quantity. Conversely, IG's negative-evidence value is related to the log probability of w_2 conditional on the event of *not* observing w_1, and as such the sum of negative evidence values is not equivalent to the joint surprisal.

Information gain can therefore predict left–right asymmetrical word-order preferences such as the order of adjectives in ANA templates. Further, it maps onto a well-known decision rule for the ordering of trees.

3.4 An efficient algorithm

The goal of algorithms such as ID3 is to produce a decision tree which divides a dataset into equal-sized and mutually-exclusive partitions, thereby creating a shallow tree (Quinlan, 1986). While finding the smallest possible binary decision tree is NP-complete (Hyafil and Rivest, 1976), ID3's locally-optimal approach has proven quite effective at producing shallow trees capable of accurate classification (Dobkin et al., 1996).

By analogy, the ordering of adjectives in a noun phrase by maximizing information gain likewise produces a tree with balanced positive and negative partitions at each node. Specifically, adjectives that minimize the entropy of both the positive and negative evidence are placed before adjectives which are less 'decisive' at partitioning feature vectors.

4 Methodology

4.1 Data

Our study relies on two types of source data, both extracted from the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (Ginter et al., 2017; Zeman et al., 2017): a set of Common Crawl and Wikipedia text data from a variety of languages, automatically parsed according to the Universal Dependencies scheme with UDPipe (Straka and Straková, 2017). First, we extract **noun phrases** (NPs) containing at least one adjective to populate feature vectors (§4.3). Second, we extract **triples**, instances of a noun and two dependent adjectives, where the three words are sequential in the surface order and neither the noun nor the adjectives have other dependents.

We restrict triples in this way to minimize the effect that other dependents might have on order preferences. For example, while single-word adjectives tend to precede the noun in English, as in 'the *nice* people', adjectives in larger right-branching phrases often follow: 'the people *nice to us*' (Matthews, 2014), a trend also seen in Romance (Gulordava et al., 2015; Gulordava and Merlo, 2015). Similarly, conjunctions have been shown to weaken or neutralize preferences (Fox and Thuilier, 2012; Rosales Jr. and Scontras, 2019; Scontras et al., 2020).

NPs and triples extracted from the Wikipedia dumps are used to generate feature vectors and to train our regression (§4.4). We use triples from the Common Crawl dumps to perform hold-out accuracy testing.

4.2 Normalization

Because our source data are extracted from dumps of automatically-parsed text, they contain a large amount of noise, such as incorrectly assigned syntactic categories, HTML, nonstandard orthography, and so on. To combat this noise, we extract all lemmas with UPOS marked as ADJ and NOUN in all Universal Dependencies (UD) v2.7 corpora (Zeman et al., 2020) for a given language—the idea being that the UD corpora are of higher quality—and include only NPs and triples in which the adjectives and nouns are in the UD lists. All characters are case-normalized, where applicable.

4.3 Feature vectors

Each NP attested in the Wikipedia corpus for a given language corresponds to a feature vector with value 1 in the dimension associated with each adjective or noun lemma. For example, an NP such as "the best room available" generates a vector containing 1 for 'is-available', 'is-best', and 'is-room'.

The relative count of each NP in the Wikipedia corpus yields a probability distribution on feature vectors. It is this distribution which is transformed by partitioning on each lemma in a triple.

4.4 Evaluation

For a given typological template (AAN, ANA, or NAA) there are two competing variants; our tasks are to (i) predict which of the variants will be attested in a corpus and (ii) show a cross-linguistic consistency in how that prediction comes about.

Because we are limiting our study to the two competing variants within each template, the position of the noun is invariant, leaving only the relative order of the two adjectives to determine the order of a triple. Our problem thus reduces to whether the information gain of the first linear adjective is greater than that of the second.

In the case of AAN and ANA triples, the IG of each adjective is calculated by partitioning the entire set of feature vectors L on each of the two adjectives. In the case of NAA triples, however, IG is calculated by partitioning only those feature vectors which 'survive' the initial partition by the noun, and are therefore part of L'. Thus we calculate $\text{IG}(L, a)$ before the noun and $\text{IG}(L', a)$ after.

Rather than simply implement the ID3 algorithm and choose adjectives based on their raw information gain, we train a logistic regression to predict surface orders based on the *difference* of IG between the attested first and second adjective, a method previously used by Morgan and Levy (2016) and Futrell et al. (2020a). The benefits of this approach are two-fold: we are able to account for bias in the distribution of adjectival IGs, and we can more easily deconstruct how strong information gain is as a predictor of adjective order.

Within each template, for each attested triple τ, let π_1 be the lexicographically-sorted first permutation of τ and π_2 be the second, with α_1 being the first linear adjective in π_1 and α_2 being the first linear adjective in π_2. Our independent variable p is whether π_1 is attested in the corpus, and our dependent variable is the difference between the information gain of α_1 and α_2. We train the coefficients β_0 and β_1 in a logistic regression of the form

$$\begin{aligned} p &= \begin{cases} 1, & \text{if } \pi_1 \text{ is attested} \\ 0, & \text{if } \pi_2 \text{ is attested} \end{cases} \\ \log \frac{p}{1-p} &\sim \beta_0 + \beta_1[\text{IG}(\alpha_1) - \text{IG}(\alpha_2)]. \end{aligned} \tag{4}$$

A positive value for β_1 tells us that permutations in which the larger-IG adjective is placed first tend to be attested. The value of β_0 tells us whether there is a generalized bias towards a positive or negative $\text{IG}(\pi_1) - \text{IG}(\pi_2)$. The accuracy we achieve by running the logistic regression on held-out testing data tells us the effectiveness of an IG-based algorithm at predicting adjective order.

4.5 Reporting results

We report results for languages from which at least 5k triples could be analyzed, and for templates representing at least 10% of a language's triples in UD corpora. The count of analyzable triples for each language is a product of those available in the 2017 CoNLL Shared Task, those with sufficiently large UD v2.7 corpora, and those that meet our extraction requirements (§4.1).

Because we are interested in exploring a cross-linguistic predictor of adjective order, we report macro-average accuracies and β_1 coefficients. That is, each language's accuracy and coefficient are calculated independently and are then averaged. We report both type- and token-accuracy, using the latter in our analysis based on the intuition that the preference for the order of a commonly-occurring triple is stronger than a more rare one.

AAN	language	n	β_1	P	token acc.	type acc.
mean β_1	Bulgarian	13018	20.058	0.000	0.650	0.649
18.591 [15.740, 21.443]	Chinese	5909	18.604	0.000	0.724	0.766
	Croatian	15555	21.246	0.000	0.666	0.634
mean token accuracy	Czech	27899	28.207	0.000	0.671	0.665
0.656 [0.630, 0.683]	Danish	11226	17.506	0.000	0.786	0.770
	Dutch	11279	12.201	0.000	0.609	0.605
mean type accuracy	English	23311	22.076	0.000	0.643	0.647
0.645 [0.616, 0.674]	Finnish	12605	15.342	0.000	0.655	0.644
	German	16391	16.210	0.000	0.601	0.606
	Greek	5506	18.383	0.000	0.631	0.643
	Latvian	5290	15.826	0.000	0.594	0.551
	Russian	25397	25.697	0.000	0.658	0.651
	Slovak	11933	25.935	0.000	0.700	0.651
	Slovenian	18859	28.192	0.000	0.670	0.661
	Swedish	10937	11.462	0.000	0.717	0.711
	Turkish	12115	12.579	0.000	0.576	0.577
	Ukrainian	11474	15.949	0.000	0.593	0.592
	Urdu	6432	9.170	0.000	0.673	0.593

ANA	language	n	β_1	P	token acc.	type acc.
mean β_1	Basque	3322	-9.623	0.000	0.703	0.678
31.313 [16.786, 45.841]	Catalan	3117	45.135	0.000	0.818	0.814
	Croatian	4912	-3.411	0.106	0.608	0.604
mean token accuracy	French	5673	43.349	0.000	0.771	0.756
0.737 [0.674, 0.799]	Galician	5020	68.290	0.000	0.805	0.806
	Indonesian	1521	-2.462	0.138	0.543	0.524
mean type accuracy	Italian	9484	36.658	0.000	0.681	0.698
0.726 [0.665, 0.787]	Persian	2598	43.242	0.000	0.794	0.766
	Polish	13481	24.873	0.000	0.684	0.655
	Portuguese	7580	32.374	0.000	0.734	0.725
	Romanian	2426	46.823	0.000	0.730	0.739
	Spanish	9212	57.813	0.000	0.744	0.738
	Vietnamese	2636	24.013	0.000	0.962	0.931

NAA	language	n	β_1	P	token acc.	type acc.
mean β_1	Arabic	11595	4.595	0.000	0.693	0.660
4.140 [3.128, 5.152]	Basque	4899	1.957	0.000	0.626	0.635
	Catalan	2878	5.024	0.000	0.710	0.722
mean token accuracy	French	8368	5.143	0.000	0.737	0.749
0.680 [0.639, 0.721]	Galician	1334	5.776	0.000	0.716	0.694
	Hebrew	6751	1.115	0.000	0.558	0.560
mean type accuracy	Indonesian	5724	4.631	0.000	0.740	0.734
0.687 [0.647, 0.726]	Italian	4523	4.057	0.000	0.713	0.739
	Persian	12683	1.583	0.000	0.605	0.606
	Portuguese	5139	5.329	0.000	0.726	0.730
	Romanian	8492	5.333	0.000	0.742	0.746
	Spanish	6245	6.214	0.000	0.713	0.745
	Vietnamese	3354	3.068	0.000	0.561	0.606
	comprehensive mean		18.08		0.687	0.681

Table 1: Results by template and language: n triples analyzed, regression coefficient β_1 and P-value, and test accuracies. Means with 95% confidence intervals shown for each template.

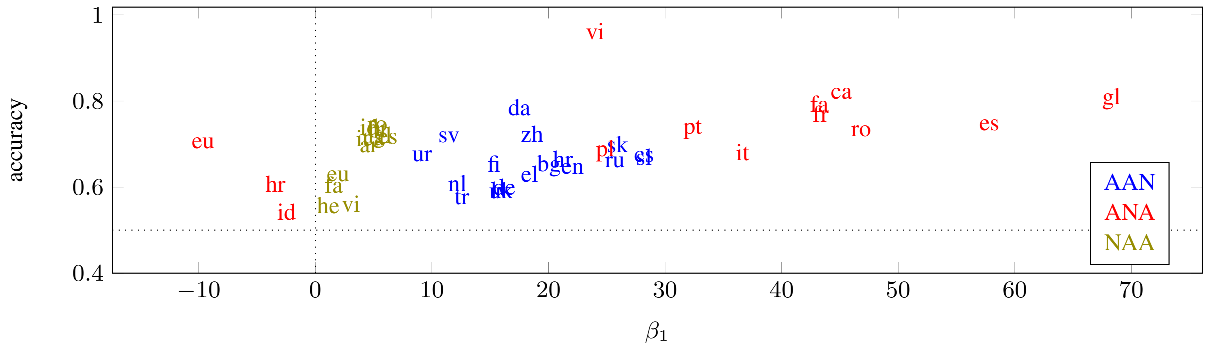

Figure 2: Plot of accuracy and β_1 coefficient, categorized by template type.

5 Results

We extracted and analyzed at least 5k triples from 32 languages across a variety of families.[3] Because some languages contain triples in two typological templates, we report results for 44 sets of triples. Table 1 reports language-specific results and means for each template, including n triples analyzed, regression coefficient β_1 and P-value, token and type accuracy, and 95% confidence intervals. Figure 2 shows a plot of accuracy and β_1 coefficient for each language, categorized by template.

As reported in Table 1, we find above-chance ($> 50\%$) accuracy for all languages tested. We accurately predict 65.6% of AAN triples, 73.7% of ANA triples, and 68.0% of NAA triples, for a comprehensive accuracy across all languages of 68.7%. Overlapping 95% confidence intervals across template suggest that IG-based prediction performs equally well across templates.

The high performance on Vietnamese ANA triples (96.2%) is largely due to the algorithm correctly predicting that the highly-frequent adjective *nhiều* ‘many’ should be placed before the noun, while most other adjectives are placed after.[4]

Though we cannot make a direct comparison to other studies due to a lack of shared data, Table 2 shows that our cross-linguistic accuracy of 68.7% bests any single predictor applied to a similar set of English AAN triples by Futrell et al. (2020a).

	n	accuracy	confidence interval
IG-FV	44	**0.687**	[0.686, 0.688]
Subj.	1	0.661	[0.657, 0.666]
PMI	1	0.659	[0.654, 0.664]
IG-NR	1	0.650	[0.645, 0.654]
IC	1	0.642	[0.634, 0.646]

Table 2: Comparison across n languages of the current metric, IG of feature vectors (IG-FV), and subjectivity, PMI, IG of noun referents (IG-NR), and integration cost (IC) (Futrell et al., 2020a)

The learned β_1 coefficient is not significantly different between AAN (18.591) and ANA (31.313) triples, though that of NAA (4.140) triples is significantly smaller than the other two. More generally, of the 44 datasets tested, β_1 is positive in 41 (93.2%), suggesting that there is a strong preference to maximize information gain. Further, of the three instances of a negative β_1, two (Croatian and Indonesian ANA) do not reach significance, perhaps due to a paucity of data. The sole significant negative β_1 is from Basque ANA triples.

6 Discussion

6.1 β_1 coefficient

Our results show a strong tendency across typological templates and across languages for the adjective which yields a larger information gain to be placed before the other, as evidenced by a positive β_1. However, the absolute value of β_1 is difficult to interpret without understanding the relative magnitudes of the underlying IG scores, magnitudes that vary across datasets and word distributions.

In general, we observe that a larger value of β_1 indicates that IG is a more reliable predictor within a dataset. More specifically, a value of $\beta_1 = 1$ indicates that if the IG difference between orders is equal to one bit, then the log odds of the order with larger IG increases by one.

---

[3] `https://github.com/wmdyer/infogain`

[4] One might worry about the classification of ‘many’ as an adjective. While widely extant across languages, the class of adjectives is not entirely homogeneous. As such, the equivalent of a word like ‘many’ in some languages might be marked as an adjective, determiner, or other syntactic category. For the current study, we simply follow the UD annotation scheme.

	n	rate	confidence interval
AAN	18	0.017	[0.012, 0.022]
ANA	13	**0.007**	[0.002, 0.011]
NAA	13	0.022	[0.013, 0.032]
all	44	0.016	[0.012, 0.020]

Table 3: Macro-average rate of adjectives attested in both possible orders within each template, showing n languages, rate of attestation, and 95% confidence intervals.

6.2 Asymmetries

The preference for one variant of an ANA triple over the other is an asymmetry without a straightforward explanation in a distance-based model; there is no clear mapping from ANA onto the other templates, which means that an adjective's relative distance to the noun is not informative. Our algorithm is novel in that the placement of the adjectives is governed by greedy IG, not distance to the noun—an innovation that allows us to break the symmetry between the adjectives in ANA triples. Similarly, IG makes no a priori prediction as to whether a mirror- or same-order will emerge between AAN and NAA triples: both pre- and post-nominal behavior is a product of ordering adjectives such that information gain is maximized, and IG itself is fundamentally derived from the distribution of adjectives and nouns that populate a language's possible feature vectors for conveying meaning.

Another left–right asymmetry that has been posited in the linguistics literature holds that dependents placed before the head in a surface realization (e.g., the adjectives in an AAN triple) follow a more rigid ordering than those placed after (e.g., the adjectives in a NAA triple; Hawkins, 1983). Both noun modifiers in general and adjectives specifically have been reported to follow this pattern, with a largely-universal pre-nominal ordering and a mirror, same, or 'free' post-nominal order (Hetzron, 1978). There is as yet no large-scale empirical evidence for this claim, though Trainin and Shetreet (2021) suggest that Hebrew NAA order preferences may be weaker than English AAN for a restricted set of adjective classes.

In an effort to empirically assess the claim that post-nominal orderings are more flexible compared to orderings pre-nominally across languages, Table 3 reports the average prevalence of adjective pairs attested in both possible orders (e.g., A_1A_2N and A_2A_1N, where N can be any noun) within each template in our dataset. At 95% confidence the difference between AAN and NAA does not reach significance, though the rate for ANA is significantly lower than the other two. More generally, the mean rate of just 1.6% across templates reinforces the notion that ordering preferences are quite robust regardless of template, at least for our normalized triples from the languages analyzed here.

6.3 Ablation

Equation 1 defines information gain as the conditioned sum of two elements, the positive evidence $D_{\text{KL}}[L'||L]$ and the negative evidence $D_{\text{KL}}[\bar{L}'||L]$. The positive evidence alone is akin to surprisal, a well-studied quantity in psycholinguistics (§3.2), while the negative evidence is related to extropy (§3.3). By ablating the IG formulation into the two terms discretely, we can show empirically that the proportionally-combined positive and negative evidence yield more accurate and consistent results than either of the two constituent terms alone.

Table 4 shows the mean accuracy and polarity proportion of the β_1 coefficient across languages and templates. The polarity of β_1 tells us whether maximizing IG (positive) or minimizing IG (negative) is the better strategy. Thus a polarity percentage close to 0 or 1 indicates more consistent behavior across templates.

For example, while the accuracy of using only positive evidence, $D_{\text{KL}}[L'||L]$, for AAN triples is 0.565, that accuracy is realized due to a 0.000 rate of positive β_1 coefficient—that is, the 56.5% accuracy is achieved by minimizing IG, placing the adjective with the lower IG first. On the other hand, while using only positive evidence to predict NAA triples yields the same accuracy, 0.565, the coefficient polarity proportion of 0.769 means that, in most NAA cases, IG should be maximized. The three templates together reflect a modest accuracy (0.566) and an inconsistent coefficient polarity proportion (0.273).

Using only negative evidence, $D_{\text{KL}}[\bar{L}'||L]$, yields even worse accuracies and similarly inconsistent coefficients as positive only. The accuracy across templates is little better than chance at 0.535, and the average coefficient polarity proportion of 0.273 likewise demonstrates that using negative evidence alone does not produce consistent behavior across templates.

	accuracy				proportion of positive β_1			
	AAN	ANA	NAA	all	AAN	ANA	NAA	all
$D_{\mathrm{KL}}[L'\|\|L]$	0.565	0.567	0.565	0.566	0.000	0.154	0.769	0.273
$D_{\mathrm{KL}}[\bar{L}'\|\|L]$	0.533	0.548	0.526	0.535	0.167	0.231	0.462	0.273
IG	**0.657**	**0.737**	**0.680**	**0.687**	**1.000**	**0.769**	**1.000**	**0.932**

Table 4: Ablation on accuracy and the proportion of positive coefficients for positive evidence ($D_{\mathrm{KL}}[L'||L]$) alone, negative evidence ($D_{\mathrm{KL}}[\bar{L}'||L]$) alone, and proportionally combined terms (IG). Boldfaced values indicate the highest accuracy or coefficient polarity proportion in each column.

The full IG calculation, including both positive and negative evidence, yields the highest accuracy across templates (0.687), as well as the highest for each template—AAN (0.657), ANA (0.737) and NAA (0.680). IG also demonstrates the most consistent behavior across languages and templates: at a rate of 0.932, maximizing IG yields the highest accuracy, regardless of whether adjectives precede or follow the noun.

7 Summary

We have taken a novel approach to the problem of predicting the surface order of adjectives across languages, casting it as a decision tree operating on a probability distribution over binary feature vectors. As each adjective is uttered, probability mass is partitioned into positive and negative subsets: those vectors that contain the feature and those that do not. The information gained by this partition can be used to order adjectives in a greedy manner, similarly to well-known algorithms for ordering nodes in a decision tree.

An IG-based approach allows us to provide the first quantitative information-theoretic account predicting the order of ANA triples. Further, with this approach we need not stipulate mirror- or same-orders for AAN and NAA triples. Because IG is not a distance metric between adjective and noun, and because IG incorporates negative evidence, both ANA and pre- or post-nominal asymmetries are able to emerge within an IG framework, without appeal to other mechanisms.

Our results show that information gain is a good predictor of adjective order across languages. Importantly, IG-based prediction follows a consistent pattern across the three typological templates, namely that adjectives that maximize information gain tend to be placed first.

References

Otto Behaghel. 1932. *Deutsche Syntax eine geschichtliche Darstellung*, volume IV. Carl Winters Unversitätsbuchhandlung, Heidelberg.

Guglielmo Cinque. 1994. On the Evidence for Partial N-Movement in the Romance DP. In Guglielmo Cinque, Jan Koster, Jean-Yves Pollack, Luigi Rizzi, and Raffaella Zanuttini, editors, *Paths Towards Universal Grammar. Studies in Honor of Richard S. Kayne,*, pages 85–110. Georgetown University Press, Washington, DC.

Guglielmo Cinque. 2009. *The Fundamental Left-Right Asymmetry of Natural Languages*, pages 165–184. Springer Netherlands, Dordrecht.

Guglielmo Cinque. 2010. *The Syntax of Adjectives: A Comparative Study*. The MIT Press, Camb., Mass.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory*. John Wiley & Sons, Hoboken, NJ.

Robert M. W. Dixon. 1982. *Where have all the adjectives gone? And other essays in semantics and syntax*. Mouton, Berlin, Germany.

David Dobkin, Truxton Fulton, Dimitrios Gunopulos, Simon Kasif, and Steven Salzberg. 1996. Induction of shallow decision trees. *submitted to IEEE PAMI*.

William E. Dyer. 2017. *Minimizing integration cost: A general theory of constituent order*. Ph.D. thesis, University of California, Davis, Davis, CA.

Gwendoline Fox and Juliette Thuilier. 2012. Predicting the position of attributive adjectives in the french np. In *New Directions in Logic, Language and Computation*, pages 1–15. Springer.

Michael Franke, Gregory Scontras, and Mihael Simonič. 2019. Subjectivity-based adjective ordering maximizes communicative success. In *Proceedings of the 41st annual meeting of the Cognitive Science Society*, pages 344–350.

Richard Futrell. 2019. Information-theoretic locality properties of natural language. In *Proceedings of* *the First Workshop on Quantitative Syntax (Quasy, SyntaxFest 2019)*, pages 2–15, Paris, France. Association for Computational Linguistics.

Richard Futrell, William Dyer, and Greg Scontras. 2020a. What determines the order of adjectives in English? comparing efficiency-based theories using dependency treebanks. In *Proc. of the 58th Annual Meeting of ACL*, pages 2003–2012, Online. ACL.

Richard Futrell, Edward Gibson, and Roger P Levy. 2020b. Lossy-context surprisal: An information-theoretic model of memory effects in sentence processing. *Cognitive science*, 44(3):e12814.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Kristina Gulordava and Paola Merlo. 2015. Structural and lexical factors in adjective placement in complex noun phrases across Romance languages. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 247–257, Beijing, China. Association for Computational Linguistics.

Kristina Gulordava, Paola Merlo, and Benoit Crabbé. 2015. Dependency length minimisation effects in short spans: a large-scale analysis of adjective placement in complex noun phrases. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 477–482, Beijing, China. Association for Computational Linguistics.

Michael Hahn, Judith Degen, Noah Goodman, Daniel Jurafsky, and Richard Futrell. 2018. An information-theoretic explanation of adjective ordering preferences. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, pages 1766–1772, Madison, WI. Cognitive Science Society.

John A. Hawkins. 1983. *Word Order Universals: Quantitative analyses of linguistic structure*. New York: Academic Press.

Robert Hetzron. 1978. On the relative order of adjectives. In Hansjakob Seiler, editor, *Language Universals*, pages 165–184. Gunter Narr Verlag, Tübingen.

Laurent Hyafil and R Rivest. 1976. Constructing optimal binary search trees is np complete. *Information Processing Letters*.

Otto Jespersen. 1922. *Language: its nature and development*. George Allen & Unwin Ltd., London.

Zeinab Kachakeche and Gregory Scontras. 2020. Adjective ordering in Arabic: Post-nominal structure and subjectivity-based preferences. In *Proc. of the LSA*, volume 5, pages 419–430.

Frank Lad, Giuseppe Sanfilippo, Gianna Agro, et al. 2015. Extropy: complementary dual of entropy. *Statistical Science*, 30(1):40–58.

Jun Yen Leung, Guy Emerson, and Ryan Cotterell. 2020. Investigating cross-linguistic adjective ordering tendencies with a latent-variable model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Roger Levy. 2005. *Probabilistic Models of Word Order and Syntactic Discontinuity*. Ph.D. thesis, Stanford University, Stanford, CA.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Haitao Liu, Chunshan Xu, and Junying Liang. 2017. Dependency distance: A new perspective on syntactic patterns in natural languages. *Physics of Life Reviews*, 21:171–93.

Robert Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92.

J. E. Martin. 1969. Semantic determinants of preferred adjective order. *Journal of Verbal Learning and Verbal Behavior*, 8:697–704.

Peter Matthews. 2014. *The Positions of Adjectives in English*. Oxford University Press, New York.

Emily Morgan and Roger Levy. 2016. Abstract knowledge versus direct experience in processing of binomial expressions. *Cognition*, 157:382–402.

Mohammad Norouzi, Maxwell D. Collins, Matthew Johnson, David J. Fleet, and Pushmeet Kohli. 2015. Efficient non-greedy optimization of decision trees. *arXiv:1511.04056 [cs]*.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Cesar Manuel Rosales Jr. and Gregory Scontras. 2019. On the role of conjunction in adjective ordering preferences. *Proceedings of the Linguistic Society of America*, 4(32):1–12.

Suttera Samonte and Gregory Scontras. 2019. Adjective ordering in Tagalog: A cross-linguistic comparison of subjectivity-based preferences. In *Proceedings of the Linguistic Society of America*, volume 4, pages 1–13.

Gregory Scontras, Galia Bar-Sever, Zeinab Kachakeche, Cesar Manuel Rosales Jr., and Suttera Samonte. 2020. Incremental semantic restriction and subjectivity-based adjective ordering. *Proceedings of Sinn und Bedeutung 24*, pages 253–270.

Gregory Scontras, Judith Degen, and Noah D. Goodman. 2017. Subjectivity predicts adjective ordering preferences. *Open Mind: Discoveries in Cognitive Science*, 1(1):53–65.

Gregory Scontras, Judith Degen, and Noah D. Goodman. 2019. On the grammatical source of adjective ordering preferences. *Semantics and Pragmatics*, 12(7).

Gary-John Scott. 2002. Stacked adjectival modification and the structure of nominal phrases. In *Functional Structure in DP and IP: The Cartography of Syntactic Structures*, volume 1, pages 91–210. Oxford University Press, New York.

Yuxin Shi and Gregory Scontras. 2020. Mandarin has subjectivity-based adjective ordering preferences in the presence of 'de'. In *Proceedings of the Linguistic Society of America*, volume 5, pages 410–418.

Mihael Simonič. 2018. Functional explanation of adjective ordering preferences using probabilistic programming. Master's thesis, University of Tübingen.

Richard Sproat and Chilin Shih. 1991. The Cross-Linguistic Distribution of Adjective Ordering Restrictions. In Carol Georgopoulos and Roberta Ishihara, editors, *Interdisciplinary Approaches to Language*, pages 565 – 93. Kluwer Academic Publishers, Boston.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver. ACL.

Henry Sweet. 1900. *A new English grammar, logical and historical*, volume 1. Clarendon Press, Oxford.

David Temperley and Daniel Gildea. 2018. Minimizing syntactic dependency lengths: Typological/cognitive universal? *Annual Review of Linguistics*, 4:1–15.

Juliette Thuilier. 2014. An Experimental Approach to French Attributive Adjective Syntax. In Christopher Piñón, editor, *Empirical Issues in Syntax and Semantics*, volume 10 of *Experimental Syntax and Semantics*, pages 287–304.

Nitzan Trainin and Einat Shetreet. 2021. It's a dotted blue big star: on adjective ordering in a post-nominal language. *Language, Cognition and Neuroscience*, 36(3):320–341.

Benjamin Lee Whorf. 1945. Grammatical Categories. *Language*, 21(1):1–11.

Daniel Zeman, Joakim Nivre, ... Abrams, and Anna Zhuravleva. 2020. Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Daniel Zeman, Martin Popel, ..., and Josie Li. 2017. Multilingual parsing from raw text to universal dependencies. In *Proc. of CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19. ACL.

A Survey of Data Augmentation Approaches for NLP

Steven Y. Feng[*,1] **Varun Gangal**[*,1] **Jason Wei**[†,2] **Sarath Chandar**[3]
Soroush Vosoughi[4] **Teruko Mitamura**[1] **Eduard Hovy**[1]
[1]Carnegie Mellon University, [2]Google Research
[3]Mila - Quebec AI Institute, [4]Dartmouth College
`{syfeng,vgangal,teruko,hovy}@cs.cmu.edu`
`jasonwei@google.com` `sarath.chandar@mila.quebec`
`soroush@dartmouth.edu`

Abstract

Data augmentation has recently seen increased interest in NLP due to more work in low-resource domains, new tasks, and the popularity of large-scale neural networks that require large amounts of training data. Despite this recent upsurge, this area is still relatively underexplored, perhaps due to the challenges posed by the discrete nature of language data. In this paper, we present a comprehensive and unifying survey of data augmentation for NLP by summarizing the literature in a structured manner. We first introduce and motivate data augmentation for NLP, and then discuss major methodologically representative approaches. Next, we highlight techniques that are used for popular NLP applications and tasks. We conclude by outlining current challenges and directions for future research. Overall, our paper aims to clarify the landscape of existing literature in data augmentation for NLP and motivate additional work in this area. We also present a GitHub repository with a paper list that will be continuously updated at https://github.com/styfeng/DataAug4NLP.

1 Introduction

Data augmentation (DA) refers to strategies for increasing the diversity of training examples without explicitly collecting new data. It has received active attention in recent machine learning (ML) research in the form of well-received, general-purpose techniques such as UDA (Xie et al., 2020) (3.1) and MixUp (Zhang et al., 2017) (3.2). These are often first explored in computer vision (CV), and DA's adaptation for natural language processing (NLP) seems secondary and comparatively underexplored, perhaps due to challenges presented by the discrete nature of language, which rules out continuous noising and makes it hard to maintain invariance.

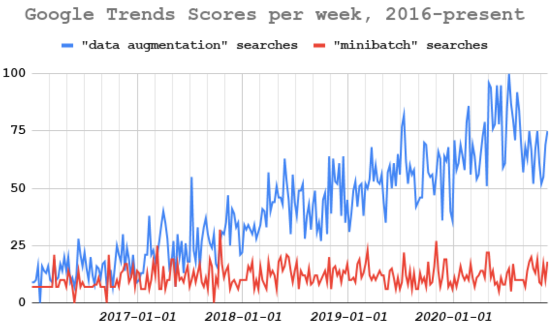

Figure 1: Weekly Google Trends scores for the search term *"data augmentation"*, with a control, uneventful ML search term (*"minibatch"*) for comparison.

Despite these challenges, there has been increased interest and demand for DA for NLP. As NLP grows due to off-the-shelf availability of large pretrained models, there are increasingly more tasks and domains to explore. Many of these are low-resource, and have a paucity of training examples, creating many use-cases for which DA can play an important role. Particularly, for many non-classification NLP tasks such as span-based tasks and generation, DA research is relatively sparse despite their ubiquity in real-world settings.

Our paper aims to sensitize the NLP community towards this growing area of work, which has also seen increasing interest in ML overall (as seen in Figure 1). As interest and work on this topic continue to increase, this is an opportune time for a paper of our kind to (i) give a bird's eye view of DA for NLP, and (ii) identify key challenges to effectively motivate and orient interest in this area. To the best of our knowledge, this is the first survey to take a detailed look at DA methods for NLP.[1]

This paper is structured as follows. Section

---

[*] Equal contribution by the two authors.

[†] AI Resident.

[1] Liu et al. (2020a) present a smaller-scale text data augmentation survey that is concise and focused. Our work serves as a more comprehensive survey with larger coverage and is more up-to-date.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 968–988
August 1–6, 2021.

2 discusses what DA is, its goals and trade-offs, and why it works. Section 3 describes popular methodologically representative DA techniques for NLP—which we categorize into rule-based (3.1), example interpolation-based (3.2), or model-based (3.3). Section 4 discusses useful NLP applications for DA, including low-resource languages (4.1), mitigating bias (4.2), fixing class imbalance (4.3), few-shot learning (4.4), and adversarial examples (4.5). Section 5 describes DA methods for common NLP tasks including summarization (5.1), question answering (5.2), sequence tagging tasks (5.3), parsing tasks (5.4), grammatical error correction (5.5), neural machine translation (5.6), data-to-text NLG (5.7), open-ended and conditional text generation (5.8), dialogue (5.9), and multimodal tasks (5.10). Finally, Section 6 discusses challenges and future directions in DA for NLP. Appendix A lists useful blog posts and code repositories.

Through this work, we hope to emulate past papers which have surveyed DA methods for other types of data, such as images (Shorten and Khoshgoftaar, 2019), faces (Wang et al., 2019b), and time series (Iwana and Uchida, 2020). We hope to draw further attention, elicit broader interest, and motivate additional work in DA, particularly for NLP.

2 Background

What is data augmentation? Data augmentation (DA) encompasses methods of increasing training data diversity without directly collecting more data. Most strategies either add slightly modified copies of existing data or create synthetic data, aiming for the augmented data to act as a regularizer and reduce overfitting when training ML models (Shorten and Khoshgoftaar, 2019; Hernández-García and König, 2020). DA has been commonly used in CV, where techniques like *cropping*, *flipping*, and *color jittering* are a standard component of model training. In NLP, where the input space is discrete, how to generate effective augmented examples that capture the desired invariances is less obvious.

What are the goals and trade-offs? Despite challenges associated with text, many DA techniques for NLP have been proposed, ranging from rule-based manipulations (Zhang et al., 2015) to more complicated generative approaches (Liu et al., 2020b). As DA aims to provide an alternative to collecting more data, an ideal DA technique should be both easy-to-implement and improve model performance. Most offer trade-offs between these two.

Rule-based techniques are easy-to-implement but usually offer incremental performance improvements (Li et al., 2017; Wei and Zou, 2019; Wei et al., 2021b). Techniques leveraging trained models may be more costly to implement but introduce more data variation, leading to better performance boosts. Model-based techniques customized for downstream tasks can have strong effects on performance but be difficult to develop and utilize.

Further, the distribution of augmented data should neither be too similar nor too different from the original. This may lead to greater overfitting or poor performance through training on examples not representative of the given domain, respectively. Effective DA approaches should aim for a balance.

Kashefi and Hwa (2020) devise a KL-Divergence-based unsupervised procedure to *preemptively* choose among DA heuristics, rather than a typical "run-all-heuristics" comparison, which can be very time and cost intensive.

Interpretation of DA Dao et al. (2019) note that *"data augmentation is typically performed in an adhoc manner with little understanding of the underlying theoretical principles"*, and claim the typical explanation of DA as *regularization* to be insufficient. Overall, there indeed appears to be a lack of research on *why* exactly DA works. Existing work on this topic is mainly surface-level, and rarely investigates the theoretical underpinnings and principles. We discuss this challenge more in §6, and highlight some of the existing work below.

Bishop (1995) show training with noised examples is reducible to Tikhonov regularization (subsumes L2). Rajput et al. (2019) show that DA can increase the positive margin for classifiers, but only when augmenting exponentially many examples for common DA methods.

Dao et al. (2019) think of DA transformations as kernels, and find two ways DA helps: averaging of features and variance regularization. Chen et al. (2020d) show that DA leads to variance reduction by averaging over orbits of the group that keep the data distribution approximately invariant.

3 Techniques & Methods

We now discuss some methodologically representative DA techniques which are relevant to all tasks via the extensibility of their formulation.[2]

---

[2] Table 1 compares several DA methods by various aspects relating to their applicability, dependencies, and requirements.

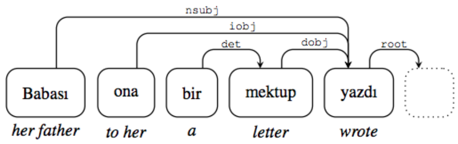

(a) Dependency analysis

(1) Babası yazdı (Her father he-wrote)
(2) Ona yazdı (He-wrote to her)
(3) Bir mektup yazdı (He-wrote a letter)

(b) Sentence Cropping

(1) Babası yazdı bir mektup ona (SVOIO)
(2) Yazdı babası ona bir mektup (VSIOO)
(3) Bir mektup yazdı babası ona (OVSIO)
(4) Ona bir mektup yazdı babası (IOOVS)

(c) Sentence Rotating

Figure 2: *Dependency tree morphing* DA applied to a Turkish sentence, Şahin and Steedman (2018)

3.1 Rule-Based Techniques

Here, we cover DA primitives which use easy-to-compute, predetermined transforms sans model components. *Feature space DA* approaches generate augmented examples in the model's feature space rather than input data. Many few-shot learning approaches (Hariharan and Girshick, 2017; Schwartz et al., 2018) leverage estimated feature space "analogy" transformations between examples of known classes to augment for novel classes (see §4.4). Paschali et al. (2019) use iterative affine transformations and projections to maximally "stretch" an example along the class-manifold.

Wei and Zou (2019) propose EASY DATA AUGMENTATION (EDA), a set of token-level random perturbation operations including *random insertion, deletion*, and *swap*. They show improved performance on many text classification tasks. UDA (Xie et al., 2020) show how supervised DA methods can be exploited for unsupervised data through consistency training on $(x, DA(x))$ pairs.

For paraphrase identification, Chen et al. (2020b) construct a signed graph over the data, with individual sentences as nodes and pair labels as signed edges. They use balance theory and transitivity to infer augmented sentence pairs from this graph. Motivated by image cropping and rotation, Şahin and Steedman (2018) propose *dependency tree morphing*. For dependency-annotated sentences, children of the same parent are swapped (à la rotation) or some deleted (à la cropping), as seen in Figure 2. This is most beneficial for language families with rich case marking systems (e.g. *Baltic* and *Slavic*).

3.2 Example Interpolation Techniques

Another class of DA techniques, pioneered by MIXUP (Zhang et al., 2017), interpolates the inputs and labels of two or more real examples. This class of techniques is also sometimes referred to as *Mixed Sample Data Augmentation* (MSDA). Ensuing work has explored interpolating inner components (Verma et al., 2019; Faramarzi et al., 2020), more general mixing schemes (Guo, 2020), and adding adversaries (Beckham et al., 2019).

Another class of extensions of MIXUP which has been growing in the vision community attempts to fuse raw input image pairs together into a single input image, rather than improve the continuous interpolation mechanism. Examples of this paradigm include CUTMIX (Yun et al., 2019), CUTOUT (DeVries and Taylor, 2017) and COPY-PASTE (Ghiasi et al., 2020). For instance, CUTMIX replaces a small sub-region of Image A with a patch sampled from Image B, with the labels mixed in proportion to sub-region sizes. There is potential to borrow ideas and inspiration from these works for NLP, e.g. for multimodal work involving both images and text (see *"Multimodal challenges"* in §6).

A bottleneck to using MIXUP for NLP tasks was the requirement of continuous inputs. This has been overcome by mixing embeddings or higher hidden layers (Chen et al., 2020c). Later variants propose speech-tailored mixing schemes (Jindal et al., 2020b) and interpolation with adversarial examples (Cheng et al., 2020), among others.

SEQ2MIXUP (Guo et al., 2020) generalizes MIXUP for sequence transduction tasks in two ways - the "hard" version samples a binary mask (from a Bernoulli with a $\beta(\alpha, \alpha)$ prior) and picks from one of two sequences at each token position, while the "soft" version softly interpolates between sequences based on a coefficient sampled from $\beta(\alpha, \alpha)$. The "soft" version is found to outperform the "hard" version and earlier interpolation-based techniques like SWITCHOUT (Wang et al., 2018a).

3.3 Model-Based Techniques

Seq2seq and language models have also been used for DA. The popular BACKTRANSLATION method (Sennrich et al., 2016) translates a sequence into another language and then back into the original language. Kumar et al. (2019a) train seq2seq models with their proposed method DiPS which learns to generate diverse paraphrases of input text using a modified decoder with a submodular objective,

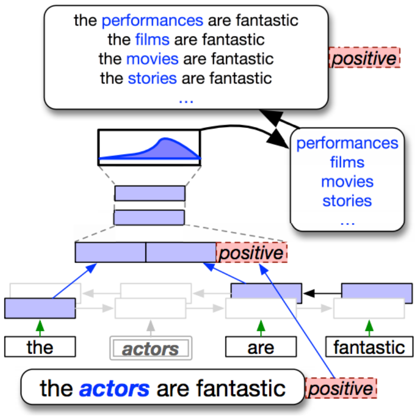

Figure 3: *Contextual Augmentation*, Kobayashi (2018)

and show its effectiveness as DA for several classification tasks. Pretrained language models such as RNNs (Kobayashi, 2018) and transformers (Yang et al., 2020) have also been used for augmentation.

Kobayashi (2018) generate augmented examples by replacing words with others randomly drawn according to the recurrent language model's distribution based on the current context (illustration in Figure 3). Yang et al. (2020) propose G-DAUG^c which generates synthetic examples using pretrained transformer language models, and selects the most informative and diverse set for augmentation. Gao et al. (2019) advocate retaining the full distribution through "soft" augmented examples, showing gains on machine translation.

Nie et al. (2020) augment word representations with a context-sensitive attention-based mixture of their semantic neighbors from a pretrained embedding space, and show its effectiveness for NER on social media text. Inspired by denoising autoencoders, Ng et al. (2020) use a corrupt-and-reconstruct approach, with the corruption function $q(x'|x)$ masking an arbitrary number of word positions and the reconstruction function $r(x|x')$ unmasking them using BERT (Devlin et al., 2019). Their approach works well on domain-shifted test sets across 9 datasets on sentiment, NLI, and NMT.

Feng et al. (2019) propose a task called SEMANTIC TEXT EXCHANGE (STE) which involves adjusting the overall semantics of a text to fit the context of a new word/phrase that is inserted called the *replacement entity (RE)*. They do so by using a system called SMERTI and a masked LM approach. While not proposed directly for DA, it can be used as such, as investigated in Feng et al. (2020).

Rather than starting from an existing example and modifying it, some model-based DA approaches directly estimate a generative process from the training set and sample from it. Anaby-Tavor et al. (2020) learn a label-conditioned generator by finetuning GPT-2 (Radford et al., 2019) on the training data, using this to generate candidate examples per class. A classifier trained on the original training set is then used to select top k candidate examples which confidently belong to the respective class for augmentation. Quteineh et al. (2020) use a similar label-conditioned GPT-2 generation method, and demonstrate its effectiveness as a DA method in an active learning setup.

Other approaches include syntactic or controlled paraphrasing (Iyyer et al., 2018; Kumar et al., 2020), document or story-level paraphrasing (Gangal et al., 2021), augmenting misclassified examples (Dreossi et al., 2018), BERT cross-encoder labeling of new inputs (Thakur et al., 2021), and guided generation using large-scale generative language models (Liu et al., 2020b,c). Models can also learn to combine together simpler DA primitives (Cubuk et al., 2018; Ratner et al., 2017) or add human-in-the-loop (Kaushik et al., 2020, 2021).

4 Applications

In this section, we discuss several DA methods for some common NLP applications.[2]

4.1 Low-Resource Languages

Low-resource languages are an important and challenging application for DA, typically for neural machine translation (NMT). Techniques using external knowledge such as WordNet (Miller, 1995) may be difficult to use effectively here.[3] There are ways to leverage high-resource languages for low-resource languages, particularly if they have similar linguistic properties. Xia et al. (2019) use this approach to improve low-resource NMT.

Li et al. (2020b) use *backtranslation* and self-learning to generate augmented training data. Inspired by work in CV, Fadaee et al. (2017) generate additional training examples that contain low-frequency (rare) words in synthetically created contexts. Qin et al. (2020) present a DA framework to generate multi-lingual code-switching data to finetune multilingual-BERT. It encourages the alignment of representations from source and multiple target languages once by mixing their context information. They see improved performance across 5 tasks with 19 languages.

---

[3] Low-resource language challenges discussed more in §6.

DA Method	Ext.Know	Pretrained	Preprocess	Level	Task-Agnostic
Synonym Replacement (Zhang et al., 2015)	✓	×	tok	Input	✓
Random Deletion (Wei and Zou, 2019)	×	×	tok	Input	✓
Random Swap (Wei and Zou, 2019)	×	×	tok	Input	✓
Backtranslation (Sennrich et al., 2016)	×	✓	Depends	Input	✓
SCPN (Wieting and Gimpel, 2017)	×	✓	const	Input	✓
Semantic Text Exchange (Feng et al., 2019)	×	✓	const	Input	✓
ContextualAug (Kobayashi, 2018)	×	✓	-	Input	✓
LAMBADA (Anaby-Tavor et al., 2020)	×	✓	-	Input	×
GECA (Andreas, 2020)	×	×	tok	Input	×
SeqMixUp (Guo et al., 2020)	×	×	tok	Input	×
SwitchOut (Wang et al., 2018b)	×	×	tok	Input	×
EMix (Jindal et al., 2020a)	×	×	-	Emb/Hidden	✓
SpeechMix (Jindal et al., 2020b)	×	×	-	Emb/Hidden	Speech/Audio
MixText (Chen et al., 2020c)	×	×	-	Emb/Hidden	✓
SignedGraph (Chen et al., 2020b)	×	×	-	Input	×
DTreeMorph (Şahin and Steedman, 2018)	×	×	dep	Input	✓
Sub^2 (Shi et al., 2021)	×	×	dep	Input	Substructural
DAGA (Ding et al., 2020)	×	×	tok	Input+Label	×
WN-Hypers (Feng et al., 2020)	✓	×	const+KWE	Input	✓
Synthetic Noise (Feng et al., 2020)	×	×	tok	Input	✓
UEdin-MS (DA part) (Grundkiewicz et al., 2019)	✓	×	tok	Input	✓
Nonce (Gulordava et al., 2018)	✓	×	const	Input	✓
XLDA (Singh et al., 2019)	×	✓	Depends	Input	✓
SeqMix (Zhang et al., 2020)	×	✓	tok	Input+Label	×
Slot-Sub-LM (Louvan and Magnini, 2020)	×	✓	tok	Input	✓
UBT & TBT (Vaibhav et al., 2019)	×	✓	Depends	Input	✓
Soft Contextual DA (Gao et al., 2019)	×	✓	tok	Emb/Hidden	✓
Data Diversification (Nguyen et al., 2020)	×	✓	Depends	Input	✓
DiPS (Kumar et al., 2019a)	×	✓	tok	Input	✓
Augmented SBERT (Thakur et al., 2021)	×	✓	-	Input+Label	Sentence Pairs

Table 1: Comparing a selection of DA methods by various aspects relating to their applicability, dependencies, and requirements. *Ext.Know*, *KWE*, *tok*, *const*, and *dep* stand for External Knowledge, keyword extraction, tokenization, constituency parsing, and dependency parsing, respectively. *Ext.Know* refers to whether the DA method requires external knowledge (e.g. WordNet) and *Pretrained* if it requires a pretrained model (e.g. BERT). *Preprocess* denotes preprocessing required, *Level* denotes the depth at which data is modified by the DA, and *Task-Agnostic* refers to whether the DA method can be applied to different tasks. See Appendix B for further explanation.

4.2 Mitigating Bias

Zhao et al. (2018) attempt to mitigate gender bias in coreference resolution by creating an augmented dataset identical to the original but biased towards the underrepresented gender (using gender swapping of entities such as replacing *"he"* with *"she"*) and train on the union of the two datasets. Lu et al. (2020) formally propose Counterfactual DA (CDA) for gender bias mitigation, which involves causal interventions that break associations between gendered and gender-neutral words. Zmigrod et al. (2019) and Hall Maudslay et al. (2019) propose further improvements to CDA. Moosavi et al. (2020) augment training sentences with their corresponding predicate-argument structures, improving the robustness of transformer models against various types of biases.

4.3 Fixing Class Imbalance

Fixing class imbalance typically involves a combination of undersampling and oversampling. Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002), which generates augmented minority class examples through interpolation, still remains popular (Fernández et al., 2018). Multilabel SMOTE (MLSMOTE) (Charte et al., 2015) modifies SMOTE to balance classes for multi-label classification, where classifiers predict more than one class at the same time. Other techniques such as EDA (Wei and Zou, 2019) can possibly be used for oversampling as well.

4.4 Few-Shot Learning

DA methods can ease few-shot learning by adding more examples for novel classes introduced in the few-shot phase. Hariharan and Girshick (2017) use learned analogy transformations $\phi(z_1, z_2, x)$ between example pairs from a non-novel class $z_1 \rightarrow z_2$ to generate augmented examples $x \rightarrow x'$ for novel classes. Schwartz et al. (2018) generalize this to beyond just linear offsets, through their "Δ-network" autoencoder which learns the distribution

$P(z_2|z_1, C)$ from all $y^*_{z_1} = y^*_{z_2} = C$ pairs, where C is a class and y is the ground-truth labelling function. Both these methods are applied only on image tasks, but their theoretical formulations are generally applicable, and hence we discuss them.

Kumar et al. (2019b) apply these and other DA methods for few-shot learning of novel intent classes in task-oriented dialog. Wei et al. (2021a) show that data augmentation facilitates curriculum learning for training triplet networks for few-shot text classification. Lee et al. (2021) use T5 to generate additional examples for data-scarce classes.

4.5 Adversarial Examples (AVEs)

Adversarial examples can be generated using innocuous label-preserving transformations (e.g. paraphrasing) that fool state-of-the-art NLP models, as shown in Jia et al. (2019). Specifically, they add sentences with distractor spans to passages to construct AVEs for span-based QA. Zhang et al. (2019d) construct AVEs for paraphrase detection using word swapping. Kang et al. (2018) and Glockner et al. (2018) create AVEs for textual entailment using WordNet relations.

5 Tasks

In this section, we discuss several DA works for common NLP tasks.[2] We focus on non-classification tasks as classification is worked on by default, and well covered in earlier sections (e.g. §3 and §4). Numerous previously mentioned DA techniques, e.g. (Wei and Zou, 2019; Chen et al., 2020b; Anaby-Tavor et al., 2020), have been used or can be used for text classification tasks.

5.1 Summarization

Fabbri et al. (2020) investigate *backtranslation* as a DA method for few-shot abstractive summarization with the use of a consistency loss inspired by UDA. Parida and Motlicek (2019) propose an iterative DA approach for abstractive summarization that uses a mix of synthetic and real data, where the former is generated from Common Crawl. Zhu et al. (2019) introduce a query-focused summarization (Dang, 2005) dataset collected using Wikipedia called WIKIREF which can be used for DA. Pasunuru et al. (2021) use DA methods to construct two training datasets for Query-focused Multi-Document Summarization (QMDS) called QMDSCNN and QMDSIR by modifying CNN/DM (Hermann et al., 2015) and mining search-query logs, respectively.

5.2 Question Answering (QA)

Longpre et al. (2019) investigate various DA and sampling techniques for domain-agnostic QA including paraphrasing by *backtranslation*. Yang et al. (2019) propose a DA method using distant supervision to improve BERT finetuning for open-domain QA. Riabi et al. (2020) leverage Question Generation models to produce augmented examples for zero-shot cross-lingual QA. Singh et al. (2019) propose XLDA, or CROSS-LINGUAL DA, which substitutes a portion of the input text with its translation in another language, improving performance across multiple languages on NLI tasks including the SQuAD QA task. Asai and Hajishirzi (2020) use logical and linguistic knowledge to generate additional training data to improve the accuracy and consistency of QA responses by models. Yu et al. (2018) introduce a new QA architecture called QANet that shows improved performance on SQuAD when combined with augmented data generated using backtranslation.

5.3 Sequence Tagging Tasks

Ding et al. (2020) propose DAGA, a two-step DA process. First, a language model over sequences of tags and words linearized as per a certain scheme is learned. Second, sequences are sampled from this language model and de-linearized to generate new examples. Şahin and Steedman (2018), discussed in §3.1, use *dependency tree morphing* (Figure 2) to generate additional training examples on the downstream task of part-of-speech (POS) tagging.

Dai and Adel (2020) modify DA techniques proposed for sentence-level tasks for named entity recognition (NER), including label-wise token and synonym replacement, and show improved performance using both recurrent and transformer models. Zhang et al. (2020) propose a DA method based on MIXUP called SEQMIX for active sequence labeling by augmenting queried samples, showing improvements on NER and Event Detection.

5.4 Parsing Tasks

Jia and Liang (2016) propose DATA RECOMBINATION for injecting task-specific priors to neural semantic parsers. A synchronous context-free grammar (SCFG) is induced from training data, and new "recombinant" examples are sampled. Yu et al. (2020) introduce GRAPPA, a pretraining approach for table semantic parsing, and generate synthetic question-SQL pairs via an SCFG. Andreas (2020) use *compositionality* to construct synthetic examples for downstream tasks like semantic parsing. Fragments of original examples are replaced with fragments from other examples in similar contexts.

Vania et al. (2019) investigate DA for low-resource dependency parsing including *dependency tree morphing* from Şahin and Steedman (2018) (Figure 2) and modified *nonce* sentence generation from Gulordava et al. (2018), which replaces content words with other words of the same POS, morphological features, and dependency labels.

5.5 Grammatical Error Correction (GEC)

Lack of parallel data is typically a barrier for GEC. Various works have thus looked at DA methods for GEC. We discuss some here, and more can be found in Table 2 in Appendix C.

There is work that makes use of additional resources. Boyd (2018) use German edits from Wikipedia revision history and use those relating to GEC as augmented training data. Zhang et al. (2019b) explore multi-task transfer, or the use of annotated data from other tasks.

There is also work that adds synthetic errors to noise the text. Wang et al. (2019a) investigate two approaches: token-level perturbations and training error generation models with a filtering strategy to keep generations with sufficient errors. Grundkiewicz et al. (2019) use *confusion sets* generated by a spellchecker for noising. Choe et al. (2019) learn error patterns from small annotated samples along with *POS-specific noising*.

There have also been approaches to improve the diversity of generated errors. Wan et al. (2020) investigate noising through editing the latent representations of grammatical sentences, and Xie et al. (2018) use a neural sequence transduction model and beam search noising procedures.

5.6 Neural Machine Translation (NMT)

There are many works which have investigated DA for NMT. We highlighted some in §3 and §4.1, e.g. (Sennrich et al., 2016; Fadaee et al., 2017; Xia et al., 2019). We discuss some further ones here, and more can be found in Table 3 in Appendix C.

Wang et al. (2018a) propose SWITCHOUT, a DA method that randomly replaces words in both source and target sentences with other random words from their corresponding vocabularies. Gao et al. (2019) introduce SOFT CONTEXTUAL DA that softly augments randomly chosen words in a sentence using a contextual mixture of multiple related words over the vocabulary. Nguyen et al. (2020) propose DATA DIVERSIFICATION which merges original training data with the predictions of several forward and backward models.

5.7 Data-to-Text NLG

Data-to-text NLG refers to tasks which require generating natural language descriptions of structured or semi-structured data inputs, e.g. game score tables (Wiseman et al., 2017). Randomly perturbing game score values without invalidating overall game outcome is one DA strategy explored in game summary generation (Hayashi et al., 2019).

Two popular recent benchmarks are E2E-NLG (Dušek et al., 2018) and WebNLG (Gardent et al., 2017). Both involve generation from structured inputs - meaning representation (MR) sequences and triple sequences, respectively. Montella et al. (2020) show performance gains on WebNLG by DA using Wikipedia sentences as targets and parsed OpenIE triples as inputs. Tandon et al. (2018) propose DA for E2E-NLG based on permuting the input MR sequence. Kedzie and McKeown (2019) inject Gaussian noise into a trained decoder's hidden states and sample diverse augmented examples from it. This sample-augment-retrain loop helps performance on E2E-NLG.

5.8 Open-Ended & Conditional Generation

There has been limited work on DA for open-ended and conditional text generation. Feng et al. (2020) experiment with a suite of DA methods for finetuning GPT-2 on a low-resource domain in attempts to improve the quality of generated continuations, which they call GENAUG. They find that WN-HYPERS (WordNet hypernym replacement of keywords) and SYNTHETIC NOISE (randomly perturbing non-terminal characters in words) are useful, and the quality of generated text improves to a peak at $\approx$ 3x the original amount of training data.

5.9 Dialogue

Most DA approaches for dialogue focus on task-oriented dialogue. We outline some below, and more can be found in Table 4 in Appendix C.

Quan and Xiong (2019) present sentence and word-level DA approaches for end-to-end task-oriented dialogue. Louvan and Magnini (2020) propose LIGHTWEIGHT AUGMENTATION, a set of word-span and sentence-level DA methods for low-resource slot filling and intent classification.

Hou et al. (2018) present a seq2seq DA framework to augment dialogue utterances for dialogue language understanding (Young et al., 2013), including a *diversity rank* to produce diverse utterances. Zhang et al. (2019c) propose MADA to generate diverse responses using the property that several valid responses exist for a dialogue context.

There is also DA work for spoken dialogue. Hou et al. (2018), Kim et al. (2019), Zhao et al. (2019), and Yoo et al. (2019) investigate DA methods for dialogue and *spoken language understanding* (SLU), including generative latent variable models.

5.10 Multimodal Tasks

DA techniques have also been proposed for multimodal tasks where aligned data for multiple modalities is required. We look at ones that involve language or text. Some are discussed below, and more can be found in Table 5 in Appendix C.

Beginning with speech, Wang et al. (2020) propose a DA method to improve the robustness of downstream dialogue models to speech recognition errors. Wiesner et al. (2018) and Renduchintala et al. (2018) propose DA methods for end-to-end *automatic speech recognition* (ASR).

Looking at images or video, Xu et al. (2020) learn a cross-modality matching network to produce synthetic image-text pairs for multimodal classifiers. Atliha and Šešok (2020) explore DA methods such as synonym replacement and contextualized word embeddings augmentation using BERT for *image captioning*. Kafle et al. (2017), Yokota and Nakayama (2018), and Tang et al. (2020) propose methods for *visual QA* including question generation and adversarial examples.

6 Challenges & Future Directions

Looking forward, data augmentation faces substantial challenges, specifically for NLP, and with these challenges, new opportunities for future work arise.

Dissonance between empirical novelties and theoretical narrative: There appears to be a conspicuous lack of research on *why* DA works. Most studies might show empirically that a DA technique works and provide some intuition, but it is currently challenging to measure the goodness of a technique without resorting to a full-scale experiment. A recent work in vision (Gontijo-Lopes et al., 2020) has proposed that affinity (the distributional shift caused by DA) and diversity (the complexity of the augmentation) can predict DA performance, but it is unclear how these results might translate to NLP.

Minimal benefit for pretrained models on in-domain data: With the popularization of large pretrained language models, it has recently come to light that a couple of previously effective DA techniques for certain text classification tasks in English (Wei and Zou, 2019; Sennrich et al., 2016) provide little benefit for models like BERT and RoBERTa, which already achieve high performance on in-domain text classification (Longpre et al., 2020). One hypothesis for this could be that using simple DA techniques provides little benefit when finetuning large pretrained transformers on tasks for which examples are well-represented in the pretraining data, but DA methods could still be effective when finetuning on tasks for which examples are scarce or out-of-domain compared with the training data. Further work could study under which scenarios data augmentation for large pretrained models is likely to be effective.

Multimodal challenges: While there has been increased work in multimodal DA, as discussed in §5.10, effective DA methods for multiple modalities has been challenging. Many works focus on augmenting a single modality or multiple ones separately. For example, there is potential to further explore simultaneous image and text augmentation for image captioning, such as a combination of CUTMIX (Yun et al., 2019) and caption editing.

Span-based tasks offer unique DA challenges as there are typically many correlated classification decisions. For example, random token replacement may be a locally acceptable DA method but possibly disrupt coreference chains for latter sentences. DA techniques here must take into account dependencies between different locations in the text.

Working in specialized domains such as those with domain-specific vocabulary and jargon (e.g. *medicine*) can present challenges. Many pretrained models and external knowledge (e.g. WordNet) cannot be effectively used. Studies have shown that DA becomes less beneficial when applied to out-of-domain data, likely because the distribution of augmented data can substantially differ from the original data (Zhang et al., 2019a; Herzig et al., 2020; Campagna et al., 2020; Zhong et al., 2020).

Working with low-resource languages may present similar difficulties as specialized domains.

Further, DA techniques successful in the high-resource scenario may not be effective for low-resource languages that are of a different language family or very distinctive in linguistic and typological terms. For example, those which are language isolates or lack high-resource cognates.

More vision-inspired techniques: Although many NLP DA methods have been inspired by analogous approaches in CV, there is potential for drawing further connections. Many CV DA techniques motivated by real-world invariances (e.g. many angles of looking at the same object) may have similar NLP interpretations. For instance, *grayscaling* could translate to toning down aspects of the text (e.g. plural to singular, *"awesome"* → *"good"*). Morphing a dependency tree could be analogous to rotating an image, and paraphrasing techniques may be analogous to changing perspective. For example, negative data augmentation (NDA) (Sinha et al., 2021) involves creating out-of-distribution samples. It has so far been exclusively explored for CV, but could be investigated for text.

Self-supervised learning: More recently, DA has been increasingly used as a key component of self-supervised learning, particularly in vision (Chen et al., 2020e). In NLP, BART (Lewis et al., 2020) showed that predicting deleted tokens as a pretraining task can achieve similar performance as the masked LM, and ELECTRA (Clark et al., 2020) found that pretraining by predicting corrupted tokens outperforms BERT given the same model size, data, and compute. We expect future work will continue exploring how to effectively manipulate text for both pretraining and downstream tasks.

Offline versus online data augmentation: In CV, standard techniques such as cropping, color jittering, and rotations are typically done stochastically, allowing for DA to be incorporated elegantly into the training pipeline. In NLP, however, it is unclear how to include a lightweight code module to apply DA stochastically. This is because DA techniques for NLP often leverage external resources (e.g. a word dictionary for token substitution or a translation model for backtranslation) that are not easily transferable across model training pipelines. Thus, a common practice for DA in NLP is simply to generate augmented data offline and store it as additional data to be loaded during training.[4] Future work on a lightweight module for online DA in NLP could be fruitful, though another challenge will be determining when such a module will be helpful, which—compared with CV, where the invariances being imposed are well-accepted—can vary substantially across NLP tasks.

Lack of unification is a challenge for the current literature on data augmentation for NLP, and popular methods are often presented in an auxiliary fashion. Whereas there are well-accepted frameworks for DA for CV (e.g. default augmentation libraries in PyTorch, *RandAugment* (Cubuk et al., 2019)), there are no such "generalized" DA techniques for NLP. Further, we believe that DA research would benefit from the establishment of standard and unified benchmark tasks and datasets to compare different augmentation methods.

Good data augmentation practices would help make DA work more accessible and reproducible to the NLP and ML communities. On top of unified benchmark tasks, datasets, and frameworks/libraries mentioned above, other good practices include making code and augmented datasets publicly available, reporting variation among results (e.g. standard deviation across random seeds), and more standardized evaluation procedures. Further, transparent hyperparameter analysis, explicitly stating failure cases of proposed techniques, and discussion of the intuition and theory behind them would further improve the transparency and interpretability of DA techniques.

7 Conclusion

In this paper, we presented a comprehensive and structured survey of data augmentation for natural language processing (NLP). We provided a background about data augmentation and how it works, discussed major methodologically representative data augmentation techniques for NLP, and touched upon data augmentation techniques for popular NLP applications and tasks. Finally, we outlined current challenges and directions for future research, and showed that there is much room for further exploration. Overall, we hope our paper can serve as a guide for NLP researchers to decide on which data augmentation techniques to use, and inspire additional interest and work in this area. Please see the corresponding GitHub repository at https://github.com/styfeng/DataAug4NLP.

---

[4] See Appendix D.

References

Diego Alves, Askars Salimbajevs, and Mārcis Pinnis. 2020. Data augmentation for pipeline-based speech translation. In *9th International Conference on Human Language Technologies - the Baltic Perspective (Baltic HLT 2020)*, Kaunas, Lithuania.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? Deep learning to the rescue! In *Proceedings of AAAI*, pages 7383–7390.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Viktar Atliha and Dmitrij Šešok. 2020. Text augmentation using BERT for image captioning. *Applied Sciences*, 10:5978.

Christopher Beckham, Sina Honari, Vikas Verma, Alex M. Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Chris Pal. 2019. On adversarial mixup resynthesis. In *Advances in Neural Information Processing Systems*, pages 4346–4357.

Chris M. Bishop. 1995. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116.

Adriane Boyd. 2018. Using Wikipedia edits in low resource grammatical error correction. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 79–84, Brussels, Belgium. Association for Computational Linguistics.

Bram Bulte and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. Association for Computational Linguistics.

F. Charte, Antonio Rivera Rivas, María José Del Jesus, and Francisco Herrera. 2015. Mlsmote: Approaching imbalanced multilabel learning through synthetic instance generation. *Knowledge-Based Systems*.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Guanhua Chen, Yun Chen, Yong Wang, and Victor O.K. Li. 2020a. Lexical-constraint-aware neural machine translation via data augmentation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3587–3593. International Joint Conferences on Artificial Intelligence Organization. Main track.

Hannah Chen, Yangfeng Ji, and David Evans. 2020b. Finding friends and flipping frenemies: Automatic paraphrase dataset augmentation using graph theory. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4741–4751, Online. Association for Computational Linguistics.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020c. Mix-Text: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

Shuxiao Chen, Edgar Dobriban, and Jane Lee. 2020d. A group-theoretic framework for data augmentation. *Advances in Neural Information Processing Systems*, 33.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020e. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Mara Chinea-Ríos, Álvaro Peris, and Francisco Casacuberta. 2017. Adapting neural machine translation with parallel synthetic data. In *Proceedings of the Second Conference on Machine Translation*, pages 138–147, Copenhagen, Denmark. Association for Computational Linguistics.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–227, Florence, Italy. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. *Proceedings of ICLR*.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2018. Autoaugment: Learning augmentation policies from data. *Proceedings of CVPR*.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2019. Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3861–3867, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hao T. Dang. 2005. Overview of DUC 2005.

Tri Dao, Albert Gu, Alexander J. Ratner, Virginia Smith, Christopher De Sa, and Christopher Ré. 2019. A kernel theory of modern data augmentation. *Proceedings of Machine Learning Research*, 97:1528.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint*.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach forLow-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online. Association for Computational Linguistics.

Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Kurt Keutzer, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2018. Counterexample-guided data augmentation. In *Proceedings of IJCAI*.

Sufeng Duan, Hai Zhao, Dongdong Zhang, and Rui Wang. 2020. Syntax-aware data augmentation for neural machine translation. *arXiv preprint*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Alexander R. Fabbri, Simeng Han, Haoyuan Li, Haoran Li, Marjan Ghazvininejad, Shafiq Joty, Dragomir Radev, and Yashar Mehdad. 2020. Improving zero and few-shot abstractive summarization with intermediate fine-tuning and data augmentation. *arXiv preprint*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Alex Falcon, Oswald Lanz, and Giuseppe Serra. 2020. Data augmentation techniques for the video question answering task. *arXiv preprint*.

Mojtaba Faramarzi, Mohammad Amini, Akilesh Badrinaaraayanan, Vikas Verma, and Sarath Chandar. 2020. Patchup: A regularization technique for convolutional neural networks. *arXiv preprint*.

Mariano Felice. 2016. Artificial error generation for translation-based grammatical error correction. *University of Cambridge Technical Report*.

Steven Y. Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. GenAug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42, Online. Association for Computational Linguistics.

Steven Y. Feng, Aaron W. Li, and Jesse Hoey. 2019. Keep calm and switch on! Preserving sentiment and fluency in semantic text exchange. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2701–2711, Hong Kong, China. Association for Computational Linguistics.

Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V. Chawla. 2018. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905.

Jennifer Foster and Oistein Andersen. 2009. GenERRate: Generating errors for use in grammatical error detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 82–90, Boulder, Colorado. Association for Computational Linguistics.

Varun Gangal, Steven Y. Feng, Eduard Hovy, and Teruko Mitamura. 2021. Nareor: The narrative reordering problem. *arXiv preprint*.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy. Association for Computational Linguistics.

Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Paraphrase augmented task-oriented dialog generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 639–649, Online. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. 2020. Simple copy-paste is a strong data augmentation method for instance segmentation. *arXiv preprint*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Raphael Gontijo-Lopes, Sylvia J. Smullin, Ekin D. Cubuk, and Ethan Dyer. 2020. Tradeoffs in data augmentation: An empirical study. *Proceedings of ICLR*.

Miguel Graça, Yunsu Kim, Julian Schamper, Shahram Khadivi, and Hermann Ney. 2019. Generalizing back-translation in neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 45–52, Florence, Italy. Association for Computational Linguistics.

Milan Gritta, Gerasimos Lampouras, and Ignacio Iacobacci. 2021. Conversation graph: Data augmentation, training, and evaluation for non-deterministic dialogue management. *Transactions of the Association for Computational Linguistics*, 9:36–52.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Demi Guo, Yoon Kim, and Alexander Rush. 2020. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online. Association for Computational Linguistics.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *Proceedings of AAAI*, pages 4044–4051.

Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell, and Simone Teufel. 2019. It's all in the name: Mitigating gender bias with name-based counterfactual data substitution. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5267–5275, Hong Kong, China. Association for Computational Linguistics.

Bharath Hariharan and Ross Girshick. 2017. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027.

Hany Hassan, Mostafa Elaraby, and Ahmed Tawfik. 2017. Synthetic data for neural machine translation of spoken-dialects. *arXiv preprint*.

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Konstas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 1–14, Hong Kong. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NeurIPS*, pages 1693–1701.

Alex Hernández-García and Peter König. 2020. Data augmentation instead of explicit regularization. *arXiv preprint*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1234–1245, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jian Huang, Ya Li, Jianhua Tao, Zheng Lian, Mingyue Niu, and Minghao Yang. 2018. Multimodal continuous emotion recognition with data augmentation using recurrent neural networks. In *Proceedings of the 2018 on Audio/Visual Emotion Challenge and Workshop*, AVEC'18, page 57–64, New York, NY, USA. Association for Computing Machinery.

Brian Kenji Iwana and Seiichi Uchida. 2020. An empirical survey of data augmentation for time series classification with neural networks. *arXiv preprint*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4120–4133.

Amit Jindal, Arijit Ghosh Chowdhury, Aniket Didolkar, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. 2020a. Augmenting NLP models using latent feature interpolations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6931–6936, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Amit Jindal, Narayanan Elavathur Ranganatha, Aniket Didolkar, Arijit Ghosh Chowdhury, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. 2020b. Speechmix - augmenting deep sound recognition using hidden space interpolations. In *INTERSPEECH*, pages 861–865.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202, Santiago de Compostela, Spain. Association for Computational Linguistics.

Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018. AdvEntuRe: Adversarial training for textual entailment with knowledge-guided examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2418–2428, Melbourne, Australia. Association for Computational Linguistics.

Min-Hyung Kang. 2019. Valar nmt : Vastly lacking resources neural machine translation. *Stanford CS224N*.

Omid Kashefi and Rebecca Hwa. 2020. Quantifying the evaluation of heuristic methods for textual data augmentation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 200–208, Online. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Divyansh Kaushik, Amrith Setlur, Eduard H. Hovy, and Zachary Chase Lipton. 2021. Explaining the efficacy of counterfactually augmented data. In *International Conference on Learning Representations*.

Chris Kedzie and Kathleen McKeown. 2019. A good sample is hard to find: Noise injection sampling and self-training for neural language generation models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 584–593, Tokyo, Japan. Association for Computational Linguistics.

Hwa-Yeon Kim, Yoon-Hyung Roh, and Young-Kil Kim. 2019. Data augmentation by data noising for open-vocabulary slots in spoken language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 97–102, Minneapolis, Minnesota. Association for Computational Linguistics.

Alex Kimn. 2020. A syntactic rule-based framework for parallel data synthesis in japanese gec. Massachusetts Institute of Technology.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:329–345.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019a. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619, Minneapolis, Minnesota. Association for Computational Linguistics.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. 2019b. A closer look at feature space data augmentation for few-shot intent classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation. *arXiv preprint*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Daniel Li, Te I, Naveen Arivazhagan, Colin Cherry, and Dirk Padfield. 2020a. Sentence boundary augmentation for neural machine translation robustness. *arXiv preprint*.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 21–27, Valencia, Spain. Association for Computational Linguistics.

Yu Li, Xiao Li, Yating Yang, and Rui Dong. 2020b. A diverse data augmentation strategy for low-resource neural machine translation. *Information*, 11(5).

Zhenhao Li and Lucia Specia. 2019. Improving neural machine translation robustness via data augmentation: Beyond back-translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 328–336, Hong Kong, China. Association for Computational Linguistics.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Pei Liu, Xuemin Wang, Chao Xiang, and Weiye Meng. 2020a. A survey of text data augmentation. In *2020 International Conference on Computer Communication and Network Security (CCNS)*, pages 191–195.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020b. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041, Online. Association for Computational Linguistics.

Ruibo Liu, Guangxuan Xu, and Soroush Vosoughi. 2020c. Enhanced offensive language detection through data augmentation. *ICWSM Data Challenge*.

Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. 2019. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 220–227, Hong Kong, China. Association for Computational Linguistics.

Shayne Longpre, Yu Wang, and Christopher DuBois. 2020. How effective is task-agnostic data augmentation for pretrained transformers? *arXiv preprint arXiv:2010.01764*.

Samuel Louvan and Bernardo Magnini. 2020. Simple is better! lightweight data augmentation for low resource slot filling and intent classification. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation*, pages 167–177, Hanoi, Vietnam. Association for Computational Linguistics.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2020. *Gender Bias in Neural Natural Language Processing*, pages 189–202. Springer International Publishing, Cham.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Sebastien Montella, Betty Fabre, Tanguy Urvoy, Johannes Heinecke, and Lina Rojas-Barahona. 2020. Denoising pre-training and data augmentation strategies for enhanced RDF verbalization with transformers. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 89–99, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Nafise Sadat Moosavi, Marcel de Boer, Prasetya Ajie Utama, and Iryna Gurevych. 2020. Improving robustness by augmenting training sentences with predicate-argument structures. *arXiv preprint*.

Xiangyang Mou, Brandyn Sigouin, Ian Steenstra, and Hui Su. 2020. Multimodal dialogue state tracking by qa approach with data augmentation. *AAAI DSTC8 Workshop*.

Diego Moussallem, Mihael Arčan, Axel-Cyrille Ngonga Ngomo, and Paul Buitelaar. 2019. Augmenting neural machine translation with knowledge graphs. *arXiv preprint*.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283, Online. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq Joty, Kui Wu, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. In *Advances in Neural Information Processing Systems*, volume 33, pages 10018–10029. Curran Associates, Inc.

Yuyang Nie, Yuanhe Tian, Xiang Wan, Yan Song, and Bo Dai. 2020. Named entity recognition for social media texts with semantic augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1383–1391, Online. Association for Computational Linguistics.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018. Multi-source neural machine translation with data augmentation. *15th International Workshop on Spoken Language Translation 2018*.

Shantipriya Parida and Petr Motlicek. 2019. Abstract text summarization: A low resource challenge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5994–5998, Hong Kong, China. Association for Computational Linguistics.

Magdalini Paschali, Walter Simson, Abhijit Guha Roy, Muhammad Ferjad Naeem, Rüdiger Göbl, Christian Wachinger, and Nassir Navab. 2019. Data augmentation with manifold exploring geometric transformations for increased performance and robustness. *arXiv preprint*.

Ramakanth Pasunuru, Asli Celikyilmaz, Michel Galley, Chenyan Xiong, Yizhe Zhang, Mohit Bansal, and Jianfeng Gao. 2021. Data augmentation for abstractive query-focused multi-document summarization. *arXiv preprint*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Wei Peng, Chongxuan Huang, Tianhao Li, Yun Chen, and Qun Liu. 2020. Dictionary-based data augmentation for cross-domain neural machine translation. *arXiv preprint*.

Libo Qin, Minheng Ni, Yue Zhang, and Wanxiang Che. 2020. Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3853–3860. International Joint Conferences on Artificial Intelligence Organization. Main track.

Jun Quan and Deyi Xiong. 2019. Effective data augmentation approaches to end-to-end task-oriented dialogue. In *2019 International Conference on Asian Language Processing (IALP)*, pages 47–52.

Husam Quteineh, Spyridon Samothrakis, and Richard Sutcliffe. 2020. Textual data augmentation for efficient active learning on tiny datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7400–7410, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Shashank Rajput, Zhili Feng, Zachary Charles, Po-Ling Loh, and Dimitris Papailiopoulos. 2019. Does data augmentation lead to positive margin? In *International Conference on Machine Learning*, pages 5321–5330. PMLR.

AJ Ratner, HR Ehrenberg, Z Hussain, J Dunnmon, and C Ré. 2017. Learning to Compose Domain-Specific Transformations for Data Augmentation. *Advances in Neural Information Processing Systems*, 30:3239–3249.

Adithya Renduchintala, Shuoyang Ding, Matthew Wiesner, and Shinji Watanabe. 2018. Multi-Modal Data Augmentation for End-to-end ASR. *Proc. Interspeech 2018*, pages 2394–2398.

Arij Riabi, Thomas Scialom, Rachel Keraron, Benoît Sagot, Djamé Seddah, and Jacopo Staiano. 2020. Synthetic Data Augmentation for Zero-Shot Cross-Lingual Question Answering. *arXiv preprint*.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium. Association for Computational Linguistics.

Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex M Bronstein. 2018. δ-encoder: an effective sample synthesis method for few-shot object recognition. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2850–2860.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2021. Substructure Substitution: Structured Data Augmentation for NLP. *arXiv preprint arXiv:2101.00411*.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60.

Jasdeep Singh, Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. Xlda: Cross-lingual data augmentation for natural language inference and question answering. *arXiv preprint arXiv:1905.11471*.

Abhishek Sinha, Kumar Ayush, Jiaming Song, Burak Uzkent, Hongxia Jin, and Stefano Ermon. 2021. Negative data augmentation. In *International Conference on Learning Representations*.

Xiaohui Song, Liangjun Zang, Yipeng Su, Xing Wu, Jizhong Han, and Songlin Hu. 2020. Data Augmentation for Copy-Mechanism in Dialogue State Tracking. *arXiv preprint arXiv:2002.09634*.

Amane Sugiyama and Naoki Yoshinaga. 2019. Data augmentation using back-translation for context-aware neural machine translation. In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pages 35–44, Hong Kong, China. Association for Computational Linguistics.

Shubhangi Tandon, TS Sharath, Shereen Oraby, Lena Reed, Stephanie Lukin, and Marilyn Walker. 2018. TNT-NLG, System 2: Data repetition and meaning representation manipulation to improve neural generation. *E2E NLG Challenge Sfystem Descriptions*.

Ruixue Tang, Chao Ma, Wei Emma Zhang, Qi Wu, and Xiaokang Yang. 2020. Semantic equivalent adversarial data augmentation for visual question answering. In *European Conference on Computer Vision*, pages 437–453. Springer.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *Proceedings of NAACL*.

Vaibhav Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. 2019. Improving Robustness of Machine Translation with Synthetic Noise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1916–1920, Minneapolis, Minnesota. Association for Computational Linguistics.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong Kong, China. Association for Computational Linguistics.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving Grammatical Error Correction with Data Augmentation by Editing Latent Representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Chencheng Wang, Liner Yang, Yun Chen, Yongping Du, and Erhong Yang. 2019a. Controllable Data Synthesis Method for Grammatical Error Correction. *arXiv preprint arXiv:1909.13302*.

Longshaokan Wang, Maryam Fazel-Zarandi, Aditya Tiwari, Spyros Matsoukas, and Lazaros Polymenakos. 2020. Data augmentation for training dialog models robust to speech recognition errors. *arXiv preprint arXiv:2006.05635*.

Xiang Wang, Kai Wang, and Shiguo Lian. 2019b. A survey on face data augmentation. *arXiv preprint arXiv:1904.11685*.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018a. SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018b. SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Jason Wei, Chengyu Huang, Soroush Vosoughi, Yu Cheng, and Shiqi Xu. 2021a. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. *Proceedings of NAACL*.

Jason Wei, Chengyu Huang, Shiqi Xu, and Soroush Vosoughi. 2021b. Text augmentation in a multi-task view. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2888–2894, Online. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Max White and Alla Rozovskaya. 2020. A Comparative Study of Synthetic Data Generation Methods for Grammatical Error Correction. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 198–208, Seattle, WA, USA â†' Online. Association for Computational Linguistics.

Matthew Wiesner, Adithya Renduchintala, Shinji Watanabe, Chunxi Liu, Najim Dehak, and Sanjeev Khudanpur. 2018. Low resource multi-modal data augmentation for end-to-end ASR. *CoRR*.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088, Vancouver, Canada. Association for Computational Linguistics.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized Data Augmentation for Low-Resource Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796, Florence, Italy. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana. Association for Computational Linguistics.

N. Xu, W. Mao, P. Wei, and D. Zeng. 2020. MDA: Multimodal Data Augmentation Framework for Boosting Performance on Image-Text Sentiment/Emotion Classification Tasks. *IEEE Intelligent Systems*, pages 1–1.

Shuyao Xu, Jiehao Zhang, Jin Chen, and Long Qin. 2019. Erroneous data generation for Grammatical Error Correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 149–158, Florence, Italy. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. Data Augmentation for BERT Fine-Tuning in Open-Domain Question Answering. *arXiv preprint arXiv:1904.06652*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1008–1025.

Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dialog State Tracking with Reinforced Data Augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9474–9481.

Masashi Yokota and Hideki Nakayama. 2018. Augmenting Image Question Answering Dataset by Exploiting Image Captions. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Kang Min Yoo, Youhyun Shin, and Sang-goo Lee. 2019. Data Augmentation for Spoken Language Understanding via Joint Variational Generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7402–7409.

S. Young, M. Gašić, B. Thomson, and J. D. Williams. 2013. POMDP-Based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, 101(5):1160–1179.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. *arXiv preprint arXiv:2009.13845*.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *Proceedings of ICLR*.

Rongzhi Zhang, Yue Yu, and Chao Zhang. 2020. SeqMix: Augmenting Active Sequence Labeling via Sequence Mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8566–8579, Online. Association for Computational Linguistics.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019a. Editing-based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-Level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Yi Zhang, Tao Ge, Furu Wei, Ming Zhou, and Xu Sun. 2019b. Sequence-to-sequence Pre-training with Data Augmentation for Sentence Rewriting. *arXiv preprint arXiv:1909.06002*.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2019c. Task-oriented dialog systems that consider multiple appropriate responses under the same context.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019d. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Zijian Zhao, Su Zhu, and Kai Yu. 2019. Data augmentation with atomic templates for spoken language understanding. *arXiv preprint arXiv:1908.10770*.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6869–6882, Online. Association for Computational Linguistics.

Haichao Zhu, Li Dong, Furu Wei, Bing Qin, and Ting Liu. 2019. Transforming wikipedia into augmented data for query-focused summarization. *arXiv preprint arXiv:1911.03324*.

Ran Zmigrod, Sebastian J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual Data Augmentation for Mitigating Gender Stereotypes in Languages with Rich Morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

Appendices

A Useful Blog Posts and Code Repositories

The following blog posts and code repositories could be helpful in addition to the information presented and papers/works mentioned in the body:

- Introduction to popular text augmentation techniques: https://towardsdatascience.com/data-augmentation-in-nlp-2801a34dfc28
- Detailed blog post on various text DA techniques: https://amitness.com/2020/05/data-augmentation-for-nlp/
- Lightweight library for DA on text and audio: https://github.com/makcedward/nlpaug
- `python` framework for adversarial examples: https://github.com/QData/TextAttack

B DA Methods Table - Description of Columns and Attributes

Table 1 in the main body compares a non-exhaustive selection of DA methods along various aspects relating to their applicability, dependencies, and requirements. Below, we provide a more extensive description of each of this table's columns and their attributes.

1. **Ext.Know**: Short for external knowledge, this column is ✓ when the data augmentation process requires knowledge resources which go beyond the immediate input examples and the task definition, such as WordNet (Miller, 1995) or PPDB (Pavlick et al., 2015). Note that we exclude the case where these resources are pretrained models under a separate point (next) for clarity, since these are widespread enough to merit a separate category.

2. **Pretrained**: Denotes that the data augmentation process requires a pretrained model, such as BERT (Devlin et al., 2019) or GPT-2 (Radford et al., 2019).

3. **Preprocess**: Denotes the preprocessing steps, e.g. tokenization (*tok*), dependency parsing (*dep*), etc. required for the DA process. A hyphen (-) means either no preprocessing is required or that it was not explicitly stated.

4. **Level**: Denotes the depth and extent to which elements of the instance/data are modified by the DA. Some primitives modify just the INPUT (e.g. word swapping), some modify both INPUT and LABEL (e.g. negation), while others make changes in the embedding or hidden space (EMBED/HIDDEN) or higher representation layers enroute to the task model.

5. **Task-Agnostic**: This is an approximate, partially subjective column denoting the extent to which a DA method can be applied to different tasks. When we say ✓ here, we don't denote a very rigid sense of the term task-agnostic, but mean that it would possibly easily extend to most NLP tasks as understood by the authors. Similarly, an × denotes being restricted to a specific task (or small group of related tasks) only. There can be other labels, denoting applicability to broad task families. For example, SUBSTRUCTURAL denotes the family of tasks where sub-parts of the input are also valid input examples in their own right, e.g. constituency parsing. SENTENCE PAIRS denotes tasks which involve pairwise sentence scoring such as paraphrase identification, duplicate question detection, and semantic textual similarity.

C Additional DA Works by Task

See Table 2 for additional DA works for GEC, Table 3 for additional DA works for neural machine translation, Table 4 for additional DA works for dialogue, and Table 5 for additional DA works for multimodal tasks. Each work is described briefly.

D Additional Figure

Pedro Domingos
@pmddomingos

Data augmentation is one of the ugliest hacks in ML. If you know what the invariances are, encode them into the architecture. Don't blow up the size of you dataset in order to approximate them.

4:01 PM · May 10, 2021 · Twitter Web App

33 Retweets **9** Quote Tweets **362** Likes

Figure 4: Pedro Domingos' quip about offline data augmentation.

Paper/Work	**Brief Description**
Lichtarge et al. (2019)	Generate synthetic noised examples of Wikipedia sentences using backtranslation through various languages.
White and Rozovskaya (2020)	Detailed comparative study of the DA for GEC systems UEdin-MS (Grundkiewicz et al., 2019) and Kakao&Brain (Choe et al., 2019).
Foster and Andersen (2009)	Introduces error generation tool called GenERRate which learns to generate ungrammatical text with various errors by using an error analysis file.
Kimn (2020)	Use a set of syntactic rules for common Japanese grammatical errors to generate augmented error-correct sentence pairs for Japanese GEC.
Felice (2016)	Thesis that surveys previous work on error generation and investigates some new approaches using random and probabilistic methods.
Xu et al. (2019)	Noises using five error types: concatenation, misspelling, substitution, deletion, and transposition. Decent performance on the BEA 2019 Shared Task.
Zhang et al. (2019b)	Explore backtranslation and feature discrimination for DA.
Mizumoto et al. (2011)	DA by extracting Japanese GEC training data from the revision log of a language learning SNS.

Table 2: Additional DA works for grammatical error correction (GEC), along with a brief description of each.

Paper/Work	**Brief Description**
Vaibhav et al. (2019)	Present a *synthetic noise induction* model which heuristically adds social media noise to text, and *labeled backtranslation*.
Hassan et al. (2017)	Present a DA method to project words from closely-related high-resource languages to low-resource languages using word embedding representations.
Cheng et al. (2020)	Propose *AdvAug*, an adversarial augmentation method for NMT, by sampling adversarial examples from a new vicinity distribution and using their embeddings to augment training.
Graça et al. (2019)	Investigate improvements to sampling-based approaches and the synthetic data generated by backtranslation.
Bulte and Tezcan (2019)	Propose DA approaches for NMT that leverage information retrieved from a Translation Memory (TM) and using fuzzy TM matches.
Moussallem et al. (2019)	Propose an NMT model KG-NMT which is augmented by knowledge graphs to enhance semantic feature extraction and hence the translation of entities and terminological expressions.
Peng et al. (2020)	Propose dictionary-based DA (DDA) for cross-domain NMT by synthesizing a domain-specific dictionary and automatically generating a pseudo in-domain parallel corpus.
Li et al. (2020a)	Present a DA method using sentence boundary segmentation to improve the robustness of NMT on ASR transcripts.
Nishimura et al. (2018)	Introduce DA methods for multi-source NMT that fills in incomplete portions of multi-source training data.
Sugiyama and Yoshinaga (2019)	Investigate effectiveness of DA by backtranslation for context-aware NMT.
Li and Specia (2019)	Present DA methods to improve NMT robustness to noise while keeping models small, and explore the use of noise from external data (speech transcripts).
Chinea-Ríos et al. (2017)	Propose DA method to create synthetic data by leveraging the embedding representation of sentences.
Alves et al. (2020)	Propose two methods for pipeline-based speech translation through the introduction of errors through 1. utilizing a speech processing workflow and 2. a rule-based method.
Kang (2019)	Investigate extremely low-resource settings for NMT and a DA approach using a noisy dictionary and language models.
Chen et al. (2020a)	Investigate a DA method for lexically constraint-aware NMT to construct constraint-aware synthetic training data.
Li et al. (2020b)	Propose a diversity DA method for low-resource NMT by generating diverse synthetic parallel data on both source and target sides using a restricted sampling strategy during decoding.
Duan et al. (2020)	Propose syntax-aware DA methods with sentence-specific word selection probabilities using dependency parsing.

Table 3: Additional DA works for neural machine translation (NMT), along with a brief description of each.

Paper/Work	**Brief Description**
Gao et al. (2020)	Propose a paraphrase augmented response generation (PARG) framework to improve dialogue generation by automatically constructing augmented paraphrased training examples based on dialogue state and act labels.
Gritta et al. (2021)	Introduce a graph-based representation of dialogues called Conversation Graph (ConvGraph) that can be used for DA by creating new dialogue paths.
Yin et al. (2020)	Propose an RL-based DA approach for dialogue state tracking (DST).
Song et al. (2020)	Propose a simple DA algorithm to improve the training of copy-mechanism models for dialogue state tracking (DST).

Table 4: Additional DA works for dialogue, along with a brief description of each.

Paper/Work	**Brief Description**
Huang et al. (2018)	Propose a DA method for emotion recognition from a combination of audio, visual, and textual modalities.
Mou et al. (2020)	Introduce a DA method for Audio-Video Scene-Aware Dialogue, which involves dialogue containing a sequence of QA pairs about a video.
Falcon et al. (2020)	Investigate DA techniques for video QA including mirroring and horizontal flipping.

Table 5: Additional DA works for multimodal tasks, along with a brief description of each.

Why Machine Reading Comprehension Models Learn Shortcuts?

Yuxuan Lai, Chen Zhang, Yansong Feng[*]**, Quzhe Huang,** and **Dongyan Zhao**
Wangxuan Institute of Computer Technology, Peking University, China
The MOE Key Laboratory of Computational Linguistics, Peking University, China
`{erutan, zhangch, fengyansong, huangquzhe, zhaody}`
`@pku.edu.cn`

Abstract

Recent studies report that many machine reading comprehension (MRC) models can perform closely to or even better than humans on benchmark datasets. However, existing works indicate that many MRC models may learn *shortcuts* to outwit these benchmarks, but the performance is unsatisfactory in real-world applications. In this work, we attempt to explore, instead of the expected comprehension skills, why these models learn the *shortcuts*. Based on the observation that a large portion of questions in current datasets have *shortcut* solutions, we argue that larger proportion of *shortcut* questions in training data make models rely on *shortcut* tricks excessively. To investigate this hypothesis, we carefully design two synthetic datasets with annotations that indicate whether a question can be answered using *shortcut* solutions. We further propose two new methods to quantitatively analyze the learning difficulty regarding *shortcut* and challenging questions, and revealing the inherent learning mechanism behind the different performance between the two kinds of questions. A thorough empirical analysis shows that MRC models tend to learn *shortcut* questions earlier than challenging questions, and the high proportions of *shortcut* questions in training sets hinder models from exploring the sophisticated reasoning skills in the later stage of training.

1 Introduction

The task of machine reading comprehension (MRC) aims at evaluating whether a model can understand natural language texts by answering a series of questions. Recently, MRC research has seen considerable progress in terms of model performance, and many models are reported to approach or even outperform human-level performance on different benchmarks. These benchmarks are designed to address challenging features, such as evidence checking in multi-document inference (Yang et al., 2018), co-reference resolution (Dasigi et al., 2019), dialog understanding (Reddy et al., 2019), symbolic reasoning (Dua et al., 2019), and so on.

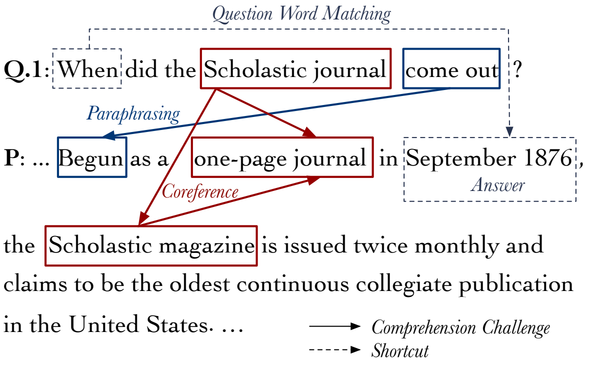

Figure 1: An illustration of *shortcuts* in Machine Reading Comprehension. **P** is an excerpt of the original passage.

However, recent analysis indicates that many MRC models unintentionally learn *shortcuts* to trick on specific benchmarks, while having inferior performance in real comprehension challenges (Sugawara et al., 2018). For example, when answering **Q.1** in Figure 1, we expect an MRC model to understand the semantic relation between *come out* and *begun*, and output the answer, *September 1876*, by bridging the co-reference among *Scholastic journal*, *Scholastic magazine* and *one-page journal.* In fact, a model can easily find the answer without following the mentioned reasoning process, since it can just recognize *September 1876* as the only time expression in the passage to answer a *when* question. We consider such kind of tricks that use *partial* evidence to produce, perhaps unreliable, answers as ***shortcuts*** to the expected comprehension challenges, e.g., *co-reference resolution* in this

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 989–1002
August 1–6, 2021.

example. The questions with *shortcut* solutions are referred to as ***shortcut questions***. For clarity, a model is considered to have ***learned shortcuts*** when it relies on those tricks to obtain correct answers for most *shortcut* questions while performing worse on questions where challenging skills are necessary.

Previous works have found that, relying on *shortcut* tricks, models may not need to pay attention to the critical components of questions and documents (Mudrakarta et al., 2018) in order to get the correct answers. Thus, many current MRC models can be either vulnerable to disturbance (Jia and Liang, 2017), or lack of flexibility to question/passage changes (Sugawara et al., 2020). These efforts disclose the impact of *shortcut* phenomenon on MRC studies. However, concerns have been raised on **why MRC models learn these *shortcuts* while ignoring the designed comprehension challenges.**

To properly investigate this problem, our first obstacle is that there are no existing MRC datasets that are labeled whether a question has *shortcut* solutions. This deficiency makes it hard to formally analysis how the performance of a model is affected by the *shortcuts questions*, and almost impossible to examine whether the model correctly answers a question via *shortcuts*. Secondly, previous methods disclose the *shortcut* phenomenon by analyzing the model outputs through a series of carefully designed experiments, but fail to explain how the MRC models learn the *shortcuts* tricks. We need new methods to help us quantitatively investigate the learning mechanisms that make the difference when MRC models learn to answer the *shortcuts* questions and questions that require challenging reasoning skills.

In this work, we carefully design two synthetic MRC datasets to support our controlled experimental analysis. Specifically, in these datasets, each (passage, question) instance has a *shortcut* version paired with a challenging one where complex comprehension skills are required to answer the question. Our construction method ensures that the two versions of questions are as close as possible, in terms of style, size, and topics, which enable us to conduct controlled experiments regarding the necessary skills to obtain answers. We design a series of experiments to quantitatively explain how *shortcut* questions affect MRC model performance and how the models learn these tricks and challenging skills during the training process. We also propose two evaluation methods to quantify the learning difficulty of specific question sets. We find that *shortcut* questions are usually easier to learn, and the dominant gradient-based optimizers drive MRC models to learn the *shortcut* questions earlier in the learning process. The priority of fitting *shortcut* questions hinders models from exploring sophisticated reasoning skills in later stage of training. Our code and datasets can be found in `https://github.com/luciusssss/why-learn-shortcut`

To summarize, our main contributions are the following: 1) We design two synthetic datasets to study two commonly seen *shortcuts* in MRC benchmarks, *question word matching* and *simple matching*, against a challenging reasoning pattern *paraphrasing*. 2) We propose two simple methods as a probe to help investigate the inherent learning mechanism behind the different performance on *shortcut* questions and challenging ones. 3) We conduct thorough experiments to quantitatively explain the behaviors of MRC models under different settings, and show that the proportions of *shortcut* questions greatly affect model performance, which may hinder MRC models from learning sophisticated reasoning skills.

2 Synthetic Dataset Construction

To study the impact of *shortcut* questions in model training, we require the datasets to be annotated with whether each question has *shortcut* solutions, or can only be answered via complex reasoning. However, none of existing MRC datasets have such annotations. We thus design two synthetic datasets where it is known whether *shortcut* solutions exist for a question.

More importantly, we need to conduct controlled experiments and ensure, for each question, the existence of *shortcuts* solutions is the only independent variable. The extraneous variables, such as sizes of datasets, topics, answer types, and even the vocabulary, should be controlled relatively steady. Thus, in our designed datasets, each entry has a *shortcut* version instance and a challenging version. The question of the *shortcut* version can be correctly solved by a certain *shortcut* trick, while an expected comprehension skill is required to deal with the challenging version. Note that we expect the two versions of questions are as close as possible so that we can switch between the two versions freely while maintaining other factors relatively steady.

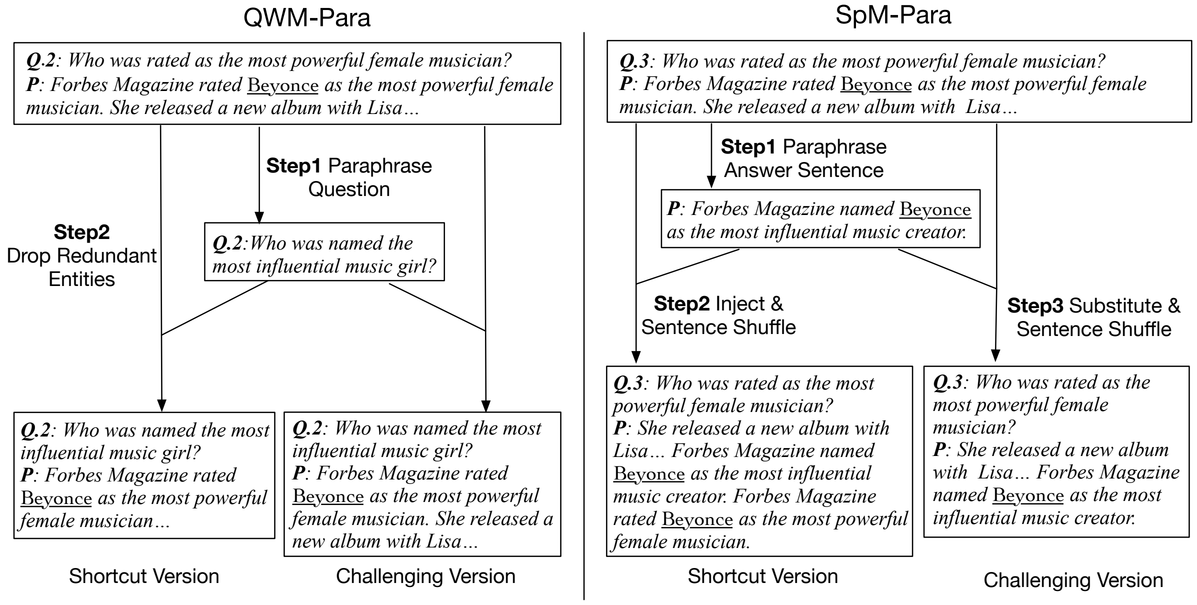

Figure 2: An illustration of how the instances in the synthetic datasets are constructed from original SQuAD data. Each instance has a *shortcut* version paired with a challenging version where comprehension skills are necessary.

In this work, we focus on *paraphrasing* (***Para***) as the complex reasoning challenge, since it widely exists in many recent MRC datasets (Trischler et al., 2017; Reddy et al., 2019; Clark et al., 2019). The paraphrasing challenge requires MRC models to identify the same meaning represented in different words. Regarding the *shortcut* tricks, we study two typical kinds: *question word matching* (***QWM***) and *simple matching* (***SpM***) (Sugawara et al., 2018). For *QWM*, MRC models can simply obtain an answer phrase by recognizing the expected entity type confined by the *wh*-question words of question Q. For *SpM*, a model can find the answers by identifying the word overlap between answer sentences and the questions.

QWM-Para **Dataset:** As elaborated in Algorithm 1, given an original instance (Q, P) from SQuAD (Rajpurkar et al., 2016), we paraphrase the question Q in Q_p to embed the *paraphrasing* challenge, and derive the corresponding *shortcut* version by dropping the sentences containing other entities with the matched type according to the question words from the given passage.

An example is shown in the left of Figure 2. In the challenging version of **Q.2**, both *Beyonce* and *Lisa* are person names which match the question word *who*. Thus, one should at least recognize the paraphrasing relationship between *named the most influential music girl* and *rated as the most powerful female musician* to distinguish between the two names to infer the correct answer. For the *shortcut* version, removing the sentence containing *Lisa* from the passage, which is also of the expected answer type *person* indicated by the question word *who*, would help a model easily get the correct answer, *Beyonce*.

Algorithm 1 Construction of *QWM-Para*

```
Input: SQuAD
Output: QWM-Para
QWM-Para ← ∅
for each instance $(Q, P)$ in SQuAD do
    if $Q$ does not start with who, when, where then
        Discard this instance.
    end if
    if the answer sentence contains other entities matching the question word then
        Discard this instance.
    end if
    Use back translation to paraphrase $Q$, obtain $Q_p$
    if the non-stop-word overlap rate between $Q_p$ and the answer sentence $> 25\%$ then
        Discard the instance.
    end if
    Delete sentences in passage $P$ that does not contain the golden answer but containing other entities matching the question word, note the modified passage as $P_s$.
    $I_s \leftarrow$ the shortcut instance version $(Q_p, P_s)$
    $I_c \leftarrow$ the challenging instance version $(Q_p, P)$
    Append the pair of questions, $(I_s, I_c)$, to QWM-Para.
end for
```

SpM-Para **Dataset:** As shown in Algorithm 2, for instances from SQuAD, we paraphrase the answer sentences in the passage to embed the *paraphrasing* challenge and obtain its challenging version. We insert the paraphrased answer sentence in front of the original one in the passage to con-

Algorithm 2 Construction of *SpM-Para*

```
Input: SQuAD
Output: SpM-Para
SpM-Para ← $\emptyset$
for each instance $(Q, P)$ in SQuAD do
    if the non-stop-word overlap rate between $Q$ and the answer sentence $S < 75\%$ then
        Discard the instance.
    end if
    Use back translation to paraphrase the answer sentence $S$ in $P$, obtain $S_p$.
    if the answer span no longer exists in $S_p$ then
        Discard this instance.
    end if
    if the non-stop-word overlap rate between $Q$ and $S_p$ $> 25\%$ then
        Discard the instance.
    end if
    Replace $S$ in $P$ with $S_p$ and shuffle sentences, noted the modified passage as $P_c$.
    Append $S_p$ to $P$ and shuffle sentences, noted the modified passage as $P_s$.
    $I_s \leftarrow$ the shortcut instance version $(Q, P_s)$
    $I_c \leftarrow$ the challenging instance version $(Q, P_c)$
    Appen d the pair of questions, $(I_s, I_c)$, to SpM-Para.
end for
```

struct the corresponding *shortcut* version, where a model can obtain the answers by either identifying the paraphrase in the passage or using the *simple matching* trick via the original answer sentences. We randomly shuffle all sentences in the passage to prevent models from learning the pattern of sentence orders in the *shortcut* version, i.e., there are two adjacent answer sentences in the passage. Here, we assume the sentence-level shuffling operation will not affect the answers and solutions for most questions, since the supporting evidence is often concentrated in a single sentence. This can also be supported by Sugawara et al. (2020)'s findings that the performance of BERT-large (Devlin et al., 2019) on SQuAD only drops by around 1.2% after sentence order shuffling.

For example, in the *shortcut* version of **Q.3** shown in the right of Figure 2, MRC models can find the answer, *Beyonce*, either from the matching context, *rated as the most powerful female musician*, or via the paraphrased one, *named as the most influential music girl*. For the challenging version, only the paraphrased answer sentence is provided, thus, the paraphrasing skill is necessary.

Dataset Details Our synthetic training and test sets are derived from the accessible training and development sets of SQuAD, respectively. We adopt back translation to obtain paraphrases of texts (Dong et al., 2017). A sentence is translated from English to German, then to Chinese, and finally back to English to obtain its paraphrased version.[1]

The *QWM-Para* dataset contains 7072 entries, each containing two versions of (question, passage) tuples, 6306/766 for training and testing, respectively. And for *SpM-Para*, there are 8514 entries, 7562/952 for training and testing, respectively.

Quality Analysis We randomly sample 20 entries from each training set of the synthetic datasets, manually analyzing their answerability. We find that 76/80 questions could be correctly answered. The unanswerable questions result from the wrong paraphrasing. Furthermore, among the answerable questions, the paraphrasing skill is necessary in 30 out of 36 questions in the challenging version, and 36 out of 40 questions of the *shortcut* version can be correctly answered via the corresponding *shortcut* trick.

3 How the *Shortcut* Questions Affect Model Performance?

Previous efforts show that *shortcut* questions widely exist in current datasets (Sugawara et al., 2020). However, there are few quantitative analysis to discuss how these *shortcut* questions affect the model performance. A reasonable guess is that, when trained with too many *shortcut* questions, the models tend to fit the *shortcut* tricks, which are possible solutions to a large amount of questions in training. We thus argue that ***the high proportions of shortcut questions in training data make models rely on the shortcut tricks.***

One straightforward way to elaborate on this point is to observe the model performance on challenging test questions when the model is trained with different proportions of *shortcut* questions. For example, if a model trained on a dataset, in which 90% of questions are *shortcut* ones, cannot perform as well as its 10% variant on challenging test questions, that will probably indicate that higher proportions of *shortcut* questions in the training data may hinder the model from learning other challenging skills.

Setup We evaluate two popular MRC models, BiDAF (Minjoon et al., 2017) and BERT-base (Devlin et al., 2019), which are widely adopted in the research for *shortcut* phenomena (Sugawara et al., 2018; Min et al., 2019; Si et al., 2019; Sugawara et al., 2020). For each combination of model and

---

[1] We use Baidu Translate API (`http://api.fanyi.baidu.com`).

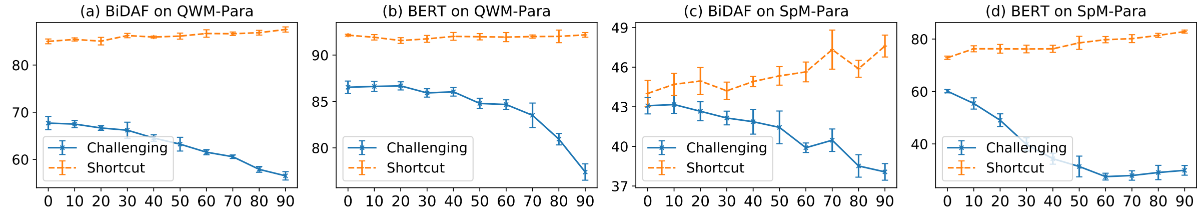

Figure 3: F1 scores on challenging and *shortcut* questions with different proportions of *shortcut* questions in training. The error bars represent the standard deviations of five runs.

dataset, we train 10 versions of the model, adjusting the proportion of *shortcut* questions in the training set from 0% to 90%, and report performance on pure challenging and pure *shortcut* test sets. We report the mean and standard deviation in five runs to alleviate the impact of randomness. Detailed settings are elaborated in Appendix A.

Results and Analysis Figure 3 shows the performance of BiDAF and BERT on *QWM-Para* and *SpM-Para* when trained with various proportions of *shortcut* questions. For both models, the F1 scores on challenging versions of both test sets drop substantially with the increase in *shortcut* questions for training (Figure 3 (a) $\sim$ (d)). This result indicates that higher proportions of *shortcut* questions in training limit the model's ability to solve challenging questions.

Take BiDAF on *QWM-Para* as an example (Figure 3 (a)). The F1 score on the test set of challenging questions is 69% after training BiDAF with a dataset entirely composed of challenging questions, showing that even a simple model is able to learn the paraphrasing skill from *shortcut*-free training data. As the proportion of *shortcut* training questions increases, the model tends to learn *shortcut* tricks and performs worse on the challenging testing data. The F1 score on challenging questions drops to 55% when 90% of the training data are *shortcut* questions. This drop shows that training data with a high proportion of *shortcuts* actually hinders the model from capturing paraphrasing skills to solve challenging questions. In contrast, the performance on *shortcut* questions are relative steady to the changes of *shortcut* proportions during training. When trained with sufficient challenging questions, models not only perform well on comprehension challenges, but also correctly answer the *shortcut* questions where only partial evidence is required.

In Figure 3, we can observe similar trends in model performance on *SpM-Para*. The performance on challenging questions also drops with higher proportions of *shortcut* training questions. Compared with BiDAF, although the overall scores of BERT are better, BERT also performs poorly on questions that require to perform paraphrasing when trained with more *shortcut* questions, as shown in Figure 3 (b) & (d).

Case Study When answering the example question **Q.4** from *QWM-Para*, BiDAF trained with pure challenging questions tends to detect the correlation between *graduate* and *received his master's degree*, and locates the correct answer *1506* when there are two spans matching the question word *when*. However, when there are more than 70% *shortcut questions* in training, BiDAF only captures the type constraint from the question word *when*, and fails to identify the paraphrasing phenomenon to answer the challenging version.

Q.4: When did Luther graduate?
P-challenging: In 1501, at the age of 19, he entered the University of Erfurt. ... He received his master's degree in *[1506]*$_{Ans}$
P-shortcut: He received his master's degree in *[1506]*$_{Ans}$

4 Whether *Question Word Matching* is Easier to Learn than *Paraphrasing*?

It is still confusing that, with the coexistence of both *shortcut* and challenging questions for training, even in a 50%-50% distribution, both BERT and BiDAF still learn *shortcut* tricks better, thus, achieve much higher performance on *shortcut questions* comparing to the challenging ones. We think one possible reason is that ***MRC models may learn the shortcut tricks, like QWM, with less computational resources than the comprehension challenges, like identifying paraphrasing***. In this case, MRC models could better learn the *shortcut* tricks with equal or even lower proportions of *shortcut* questions during training.

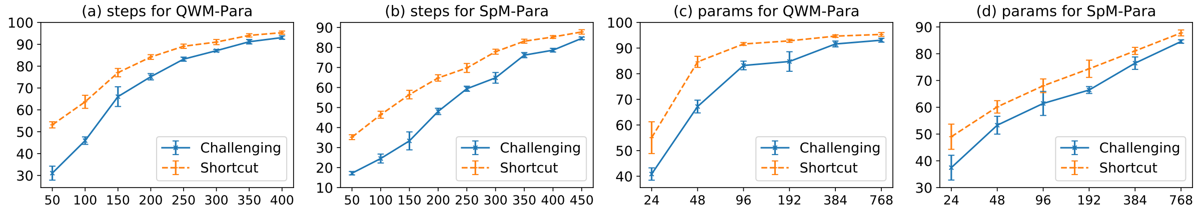

Figure 4: F1 scores on training sets when BERT learns challenging and *shortcut* questions with different optimizing steps ((a) & (b)) and parameter size (represented by the unmasked hidden size in the last hidden layer of BERT, (c) & (d)). The error bars represent the standard deviations of five runs.

To validate this hypothesis, we propose two simple but effective methods to measure the difference in required computational resources. Specifically, we can train models with either pure *shortcut* questions or challenging ones, and compare the learning speed and required parameter sizes when achieving certain performance levels on the training sets.

For learning speed, we train MRC models with different steps and observe how the performance changes. Intuitively, models should converge faster on easier training data.

For parameter sizes, our intuition is that the models should learn the easier questions with fewer parameters. However, the high computational consumption prevents us from pre-training the models like BERT with different parameter sizes. To simulate BERT with fewer parameters, we mask some hidden units in the last hidden layer of BERT and use the number of unmasked units to reflect the parameter size. The information in these masked units could not be conveyed to the span boundary prediction module. Thus, only partial parameters could be used to make the final predictions.

Setup We use BERT as the basic learner and train on the training sets of *QWM-Para* and *SpM-Para*. We report model performance on the training data with various learning settings. We use all the parameters when adjusting learning steps. When tuning parameter size, we fix the learning steps to 400 and 450 for *QWM-Para* and *SpM-Para*, respectively. All other settings including batch size, optimizer, and learning rate are fixed. We report the mean and standard deviation in five runs to alleviate the impact of randomness. The implementation details are similar to §3, elaborated in Appendix A.

Results and Analysis Figure 4 compares the performance of BERT trained on the *shortcut* questions and challenging questions separately under different settings. On both *QWM-Para* and *SpM-Para*, BERT converges faster in learning *shortcut* questions than learning challenging ones (Figure 4 (a) & (b)). When fixing the training steps, BERT could learn to answer the *shortcut* questions with fewer parameters (Figure 4 (c) & (d)). These results show that *shortcut* questions may be easier for models to learn than the ones requiring complex reasoning skills.

Take *QWM-Para* as an example. As can be seen from Figure 4 (a), BERT trained on the *shortcut* questions achieves a 90% F1-score on the training set after 250 steps. When trained on the challenging version, this score will not reach 90% until 350 steps. This result indicates that models could learn to answer the *shortcut* questions with the *QWM* trick faster than the paraphrasing skill.

When we train BERT on *QWM-Para* with different numbers of output units masked (Figure 4 (c)), BERT could reach the F1-score of 91% on *shortcut* data with no fewer than 96 unmasked hidden units. However, when trained on the challenging questions, BERT has to use 384 hidden units to reach the 91% F1 score, which indicates that the questions with the *paraphrasing* challenge may require more parameters to learn.

We observe similar trends on *SpM-Para* (Figure 4 (b) & (d)). BERT requires more parameters and training steps to learn the challenging version questions in *SpM-Para* than the *shortcut* version. To some extent, these results confirm our hypothesis that learning to answer questions with *shortcut* tricks like *SpM* or *QWM* requires smaller amounts of computational resources than the questions requiring challenging skills like paraphrasing.

Case Study For the example question **Q.5**, BERT trained on *shortcut* questions could correctly answer its *shortcut* version and find the only location name, *Palácio da Alvorada*, with only 48 unmasked hidden units. However, when trained with the challenging data only, the model predicts the other location name, *the Monumental Axis* as the answer with such parameter size. BERT could not recognize the paraphrasing relationship between *the place the president live* and *presidential residence* and choose the correct answer, *Palácio da Alvorada*, from the distracting location name, until using all 748 parameters.

Q.5: Where does the president of Brazil live, in Portuguese?
P-challenging: ... on a triangle of land jutting into the lake, is the Palace of the Dawn (*[Palácio da Alvorada]*$_{Ans}$; the presidential residence). Between the federal and civic buildings on the Monumental Axis is the city 's cathedral...
P-shortcut: ... on a triangle of land jutting into the lake, is the Palace of the Dawn (*[Palácio da Alvorada]*$_{Ans}$; the presidential residence)

---

5 How do Models Learn *Shortcuts*?

In previous section, we show that *shortcut* questions are *easier* to learn compared to the questions that require the complex *paraphrasing* skill. Then, it is interesting that, trained with a mixture of both versions of questions, how such discrepancy affects or even drives the learning procedure, e.g., how the increasing of challenging training questions alleviate the over reliance on *shortcut* tricks.

We guess one of the possible reasons is how most existing MRC models are optimized. We hypothesize that with a larger proportion of *shortcut* questions for training, ***the models have learned the shortcut tricks at the early stage, which may affect the models' further exploration for challenging skills.*** In other words, in the early stage of training, models tend to find the easiest way to fit the training data with gradient descent. Since the *shortcut* tricks require less computational resources to learn, fitting these tricks may be a priority. Afterwards, since the *shortcut* trick can be used to answer most of the training questions correctly, the limited unsolved questions remained may not motivate the models to explore sophisticated solutions that require challenging skills.

To validate this idea, we investigate how the models converge during training with different *shortcut* proportions in the training data. Notice that if a model can only answer the *shortcut* version of a question correctly, it is highly likely that the model only adopts the *shortcut* trick for this question instead of performing sophisticated reasoning skills. Thus, we think the performance gap on two versions of test data may indicate to what extent the model relies on the *shortcut* tricks, e.g., the smaller the performance gap is, the stronger complex reasoning skills the model have learned.

Setups We explore how BERT and BiDAF converge with 10% and 90% *shortcut* training questions on *QWM-Para* and *SpM-Para*. We report the F1 scores on the challenging and *shortcut* test questions, respectively, and together with their performance gaps. We compare the model performance at different learning steps to investigate when and how well the models learn the *shortcut* tricks and the challenging comprehension skills. The implementation details are the same as §3, elaborated in Appendix A.

Results and Analysis Figure 5 illustrates how the MRC models converge during training under different settings. The gap line (green with "$\star$") shows the gap between models' performance on *shortcut* questions and that on challenging ones.

For all settings, in the first few epochs, the model performance on *shortcut* questions increases rapidly, much faster than that on challenging questions, causing a steep rise of the performance gap. This result indicates that models may learn the *shortcut* tricks at the early stage of training, thus quickly and correctly answering more *shortcut* questions. And then, for the following epochs after reaching the peaks, the gap lines slightly go down (Figure 5 (a), (c), (e), and (f)) or maintain almost unchanged (Figure 5 (b), (d), (g), and (h)), which also indicates the models may learn the challenging skills later than *shortcut* tricks. One possible reason is the gradient based optimizer drives the model to optimize the global target greedily via the easiest direction. Thus, trained with a mixture of *shortcut* and challenging questions, models choose to learn the *shortcut* tricks, which require less computational resources to learn, earlier than the sophisticated paraphrasing skills.

Comparing models with different proportions of *shortcut* training questions, we find that, with 90% *shortcut* training questions (Figure 5 (b), (d), (f) and (h)), the performance gap remains at a high level in the later training stage, where the performance on the challenging test questions is relatively lower. These results provide evidence that, for most cases, after fitting on the *shortcut* questions, models seem to fail to explore the sophisticated reasoning skills.

When there are only 10% of *shortcut* training data (Figure 5 (a), (c), (e), and (g)), we can ob-

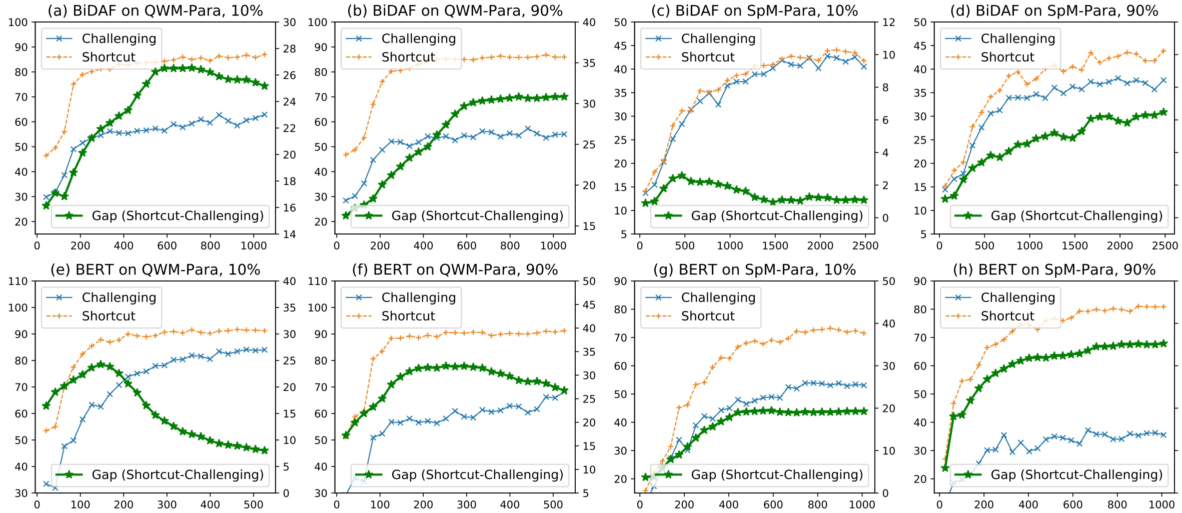

Figure 5: F1 scores on challenging and *shortcut* questions with different training steps under different settings. 10% and 90% are the proportions of *shortcut* questions in the training datasets. *Gaps* (green lines with "⋆" dots) represent the performance gap between *shortcut* questions and challenging ones, which is smoothed by averaging over fixed-size windows to mitigate periodic fluctuations.

serve that after a few hundreds of steps, the gap lines stop increasing and even slightly go down. This phenomenon shows that higher proportions of challenging questions in the training set could encourage the models to explore the sophisticated reasoning skills, but in a later stage of training.

Take BiDAF trained on *QWM-Para* as an example (Figure 5 (a) & (b)). The F1 scores on *shortcut* test questions increase quickly in the first 300 steps, while the performance gap also widens rapidly, indicating a possible fast fitting on the *shortcut* tricks. In Figure 5 (b), with 90% *shortcut* training questions, the model performance on challenging questions are relatively steady during the next 800 steps, while the F1 score on *shortcut* questions maintains a high level of about 85%. This result shows that after fitting on the *shortcut* tricks, the model trained with a higher *shortcut* proportion has almost correctly answered all the *shortcut* questions but fail to answer the challenging ones. Actually, with the gradient based optimizer, it is difficult for the model to learn the challenging questions while keeping the high performance on the *shortcut* ones, which account for 90% of the training set. We guess it is because the few unsolved challenging questions could not motivate the model to explore sophisticated reasoning skills.

On the contrary, when 90% of the training data require challenging skills, the gap line peaks at 0.27, as shown in Figure 5 (a). Afterwards, the gap decreases to 0.24, with the F1 score on challenging questions increasing to more than 60%. Larger proportions of challenging questions for training prevent the models from heavily relying on the *shortcut* tricks. This phenomenon may be because, with fewer *shortcut* questions in training, the fitting of *shortcut* tricks only benefits the training objective in a small favor. The large number of challenging questions that can not be correctly answered during the early training steps now encourage models to explore more complicated reasoning skills.

Case Study When answering the example question **Q.6** from *SpM-Para*, BERT trained with 10% *shortcut* questions tends to learn the *simple matching* trick quickly and correctly answers the *shortcut* version as early as 380 steps. However, the model cannot correctly answer the challenging variant until 630 steps. This difference demonstrates that, training with both type of questions, BERT can learn the *simple matching* trick earlier than identifying the required paraphrasing between *why defections occur* and *errors caused by*.

Q.6: Why do these defections occur?
P-challenging: ... Most of these errors are caused by *[economic or financial factors]*$_{Ans}$ ...
P-shortcut: ... Most of these defections occur because of *[economic or financial factors]*$_{Ans}$. Most of these errors are caused by *[economic or financial factors]*$_{Ans}$. ...

6 Related Works

Reading documents to answer natural language questions has drawn more and more attention in recent years (Xu et al., 2016; Minjoon et al., 2017; Lai et al., 2019; Glass et al., 2020). Most previous works focus on revealing the *shortcut* phenomenon in MRC from different perspectives. They find that manually designed features (Chen et al., 2016) or simple model architectures (Weissenborn et al., 2017) could obtain competitive performance, indicating that complicated inference procedure may be dispensable. Even without reading the entire questions or documents, models can still correctly answer a large portion of the questions (Sugawara et al., 2018; Kaushik and Lipton, 2018; Min et al., 2019). Therefore, current MRC datasets may lack the benchmarking capacity on requisite skills (Sugawara et al., 2020), and models may be vulnerable to adversarial attacks (Jia and Liang, 2017; Wallace et al., 2019; Si et al., 2019). However, they do not formally discuss or analyze why models could learn *shortcuts* from the perspectives of the learning procedure.

On the way of designing better MRC datasets, Jiang and Bansal (2019) construct adversarial questions to guide model learning the multi-hop reasoning skills. Bartolo et al. (2020) propose a model-inloop paradigm to annotate challenging questions. More recent works (Jhamtani and Clark, 2020; Ho et al., 2020) propose new datasets with evidence based metrics to evaluate whether the questions are solved via *shortcuts*. Our work aims at providing empirical evidence and analysis to the community by tracing into the learning procedure and explaining how the MRC models learn *shortcuts*, which is orthogonal to the existing works.

For a more general machine learning perspective, there are also efforts trying to explain how models learn easy and hard instances during training. Kalimeris et al. (2019) prove that models tend to learn easier decision boundaries at the beginning stage of training. Our results empirically confirms this theoretical conclusion in the task of MRC and quantitatively explain that larger proportions of *shortcut* questions in training make MRC models rely on *shortcut* tricks, rather than comprehension skills like recognizing the paraphrase relationship.

7 Conclusions

In this work, we try to answer why many MRC models learn *shortcut* tricks while ignoring the pre-designed comprehension challenges that are purposely embedded in many benchmark datasets. We argue that ***large proportions of shortcut questions in training data push MRC models to rely on shortcut tricks excessively***. To properly investigate, we first design two synthetic datasets where each instance has a *shortcut* version paired with a challenging one which requires *paraphrasing*, a complex reasoning skill, to answer, rather than performing *question word matching* or *simple matching*. With these datasets, we are able to adjust the proportion of *shortcut* questions in both training and testing, while maintaining other factors relatively steady. We propose two methods to examine the model training process regarding the *shortcut* questions, which enable us to take a closer look at the learning mechanisms of BiDAF and BERT under different training settings. We find that learning *shortcut* questions generally requires less computational resources, and MRC models usually learn the *shortcut* questions at their early stage of training. Our findings reveal that, with larger proportions of *shortcut* questions for training, MRC models will learn the *shortcut* tricks quickly while ignoring the designed comprehension challenges, since the remaining truly challenging questions, usually limited in size, may not motivate models to explore sophisticated solutions in the later training stage.

Acknowledgments

This work is supported in part by the National Hi-Tech R&D Program of China (No.2020AAA0106600), the NSFC under grant agreements (61672057, 61672058). For any correspondence, please contact Yansong Feng.

References

Max Bartolo, A. Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the ai: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinesh Garg, and Avi Sil. 2020. Span selection pretraining for question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2782, Online. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 137–150, Online. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Yichen Jiang and Mohit Bansal. 2019. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop QA. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2726–2736, Florence, Italy. Association for Computational Linguistics.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin L Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. 2019. SGD on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yuxuan Lai, Yansong Feng, Xiaohan Yu, Zheng Wang, Kun Xu, and Dongyan Zhao. 2019. Lattice cnns for matching based chinese question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6634–6641.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4249–4257, Florence, Italy. Association for Computational Linguistics.

Seo Minjoon, Kembhavi Aniruddha, Farhadi Ali, and Hajishirzi Hannaneh. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1896–1906.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. Has-qa: Hierarchical answer spans model for open-domain question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6875–6882.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. 2019. What does bert learn from multiple-choice reading comprehension datasets? *ArXiv*, abs/1910.12391.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4208–4219.

Saku Sugawara, Pontus Stenetorp, Kentaro Inui, and Akiko Aizawa. 2020. Assessing the benchmarking capacity of machine reading comprehension datasets. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada. Association for Computational Linguistics.

Jindou Wu, Yunlun Yang, Chao Deng, Hongyi Tang, Bingning Wang, Haoze Sun, Ting Yao, and Qi Zhang. 2019. Sogou machine reading comprehension toolkit. *arXiv preprint arXiv:1903.11848*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2397–2407, Osaka, Japan. The COLING 2016 Organizing Committee.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

A Implement Details

Synthetic Dataset Construction During the construction of synthetic datasets, we used Stanford CoreNLP (Manning et al., 2014) to identify named entities and stop-words.

We set two empirical thresholds for identifying questions can be solved by *Simple Matching* or requiring paraphrasing skills. We consider a question as solvable via the *simple matching* trick if more than 75% of non-stop words in the question exactly appear in the answer sentence. On the other hand, if the matching rate is below 25%, we think it is unsolvable via *simple matching*, calling for other skills like paraphrasing. Thus, in dataset construction, if the matching rate is above 25% after paraphrasing, we consider that the back translation fails to incorporate paraphrasing skills into the instance.

We construct the synthetic datasets from SQuAD (Rajpurkar et al., 2016). Compared with more recent MRC datasets (Yang et al., 2018; Kwiatkowski et al., 2019), most questions in SQuAD can be solved by a single sentence with *simple matching* so that we can conveniently use back translation to construct questions with paraphrasing challenges.

Paraphrasing in *SpM-para* When constructing the *SpM-Para* dataset, we only select the instances whose questions are very similar to the corresponding answer sentences (overlap $>$ 75%) to ensure that a *simple matching* step can obtain the answers. For the *shortcut*-version of an instance, we insert the paraphrased answer sentence into passage and keep both the original answer sentence and paraphrased answer sentence (see Algorithm 2). This operation aims to control the *shortcut* instances to have both *shortcut* solutions and challenging solutions. For the challenging version, we only keep the paraphrased answer sentences in the passages and discard the original answer sentence, so that such instances can only be solved by identifying the embedded paraphrasing relationship.

Hyper-Parameters for QA models We adopt BERT (BERT-based uncased) (Devlin et al., 2019) and BiDAF (Minjoon et al., 2017) models with the implementation in SogouMRC tools (Wu et al., 2019). The hyper-parameters are shown in Table 1. We used 100-d glove vectors (Pennington et al., 2014) in BiDAF. Notice that these hyper-parameters are adopted in §3, §4, and §5. Our code and datasets can be found in `https://github.com/luciusssss/why-learn-shortcut`

For the *simple matching* setting where multiple answer spans may appear in one passage, we follow (Pang et al., 2019) and aggregate the possibilities of each span before computing the likelihood losses.

Data Sampling in Difficulty Evaluation In §4, we train BERT on the training sets of *QWM-Para* and *SpM-Para* and observe how the model converges with different learning steps and parameter sizes. However, we find BERT achieves outstanding performance on both datasets with only one or two epochs. This is because the strong learning ability of BERT model and, with only one kind of answering pattern, both the pure *shortcut* and challenging training sets are relatively easy to learn.

Under this circumstance, BERT performance on most of the evaluation checkpoints after one epoch will almost reach the final performance, which make the comparison vague. If we compare the checkpoints within one epoch, considering that models have only been trained on partial training data, the evaluation results would reflect the models' generalization ability on unseen questions. This differs from our purpose of evaluation, namely comparing the fitting difficulty of different kinds of questions. Therefore, we randomly sample 1000 pair of instances for training and evaluation. With less training data, BERT will not converge in only one or two epochs, thus we could truly evaluate the learning ability.

Computation Cost We train the models on an NVIDIA 1080 Ti GPU. The number of parameters is 5M for BiDAF and 110M for BERT. The average training time on synthetic datasets for an epoch is 1 minute for BiDAF and 10 minutes for BERT.

	BERT-base Uncased	BiDAF
# Epoch	3	15
Batch size	6	30
Optimizer	AdamWeightDecay	Adam
Learning rate	3e-5	1e-3

Table 1: Hyper-Parameters for the experiments in §4, §5, and §6.

B A Variant of *QWM-Para* Dataset

When we train models with different proportions of *shortcut* questions on *QWM-Para* (Figure 3 (a) & (b), which is described in §3), we observe that even with pure challenging questions in training, BiDAF

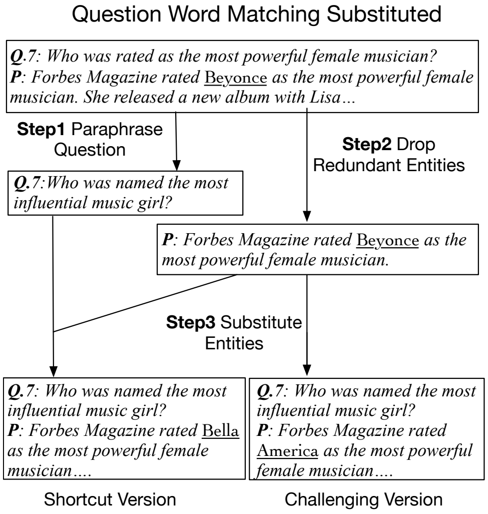

Q.7: Who was rated as the most powerful female musician?
P: Forbes Magazine rated Beyonce as the most powerful female musician. She released a new album with Lisa...

Q.7: Who was named the most influential music girl?

P: Forbes Magazine rated Beyonce as the most powerful female musician.

Q.7: Who was named the most influential music girl?
P: Forbes Magazine rated Bella as the most powerful female musician....

Q.7: Who was named the most influential music girl?
P: Forbes Magazine rated America as the most powerful female musician....

Figure 6: An illustration of how the questions in the new synthetic datasets, *question word matching substituted*, are constructed from original questions.

and BERT still perform much better on *shortcut* questions than on the challenging ones. We think this is possibly because in these settings, models fail to exploit the paraphrasing skill but learn to guess one from the the entities matching the question words as the answer. Using such a guessing trick instead of the paraphrasing skill could improve the performance on the challenging questions to some extent, but it results in more gains on the *shortcut* questions. Therefore, even with 100% challenging questions in training, the gap between the performance on challenging and *shortcut* test questions is still wide.

To avoid these guess solutions, we redesign a variant of the *QWM-Para* dataset, named as ***QWM/subs-Para***.[2] We aim at investigating: 1) Whether this variant could avoid the guessing alternative and decrease the performance gaps between challenge questions and *shortcut* ones when training with relatively lower *shortcut* proportions. 2) Whether the experiments on this variant still confirms our previous findings about how *shortcut* questions in training affect model performance and learning procedure, as described in §3 and §5.

B.1 Dataset Construction

The construction process of *QWM/subs-Para* is shown in Figure 6. The first two steps, question paraphrasing and redundant entities dropping, are the same as those in the construction of *shortcut* questions in *QWM-Para* (see §2). Then, we perform entity substitution to avoid the potential guessing solutions.

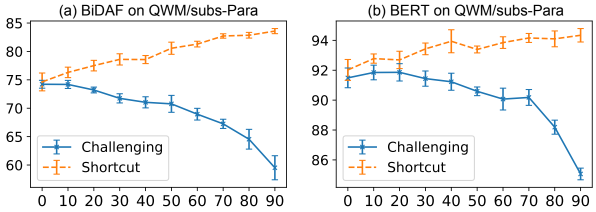

Figure 7: F1 scores on challenging and *shortcut* questions with different proportions of *shortcut* questions in training. This experiment is conducted on *QWM/subs-Para*. The error bars represent the standard deviations of five runs.

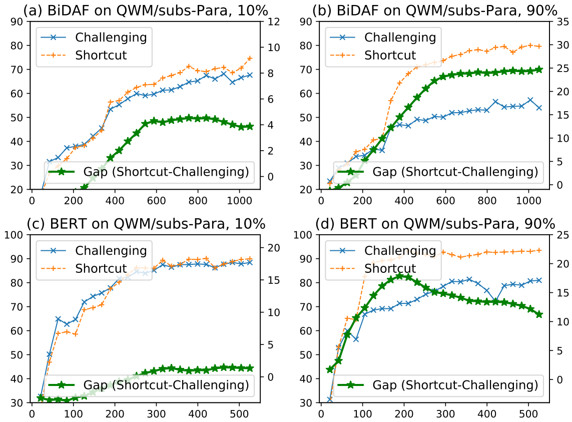

Figure 8: F1 scores on challenging and *shortcut* questions with different training steps under different settings. This experiment is conducted on *QWM/subs-Para*. 10% and 90% are the ratios of *shortcut* questions in the training datasets. *Gaps* (green lines with "⋆" dots) represents the performance gap between *shortcut* questions and challenging ones, which is smoothed by averaging over fixed-size windows to mitigate periodic fluctuations.

Particularly, for each candidate question, we substitute all the entities in the passage with random ones whose type uniformly distributes in Person/Time/Location to construct the challenging questions. With this random substitution, one can hardly guess the correct answer via the question words. As shown in **Q.7**, after substituting the answer entity *Beyonce* to *America*, one can not answer the new question by simply finding a *Person* entity according to the question word *who*. Replacing a person's name with a location may break the original semantic, but it will force the model to comprehend the context to find the answers. For the *shortcut* version, we also conduct the random entity substitution, but within the same entity types,

---

[2] *subs* refers to substituted, elaborated in §B.1.

e.g., from *Beyonce* to *Bella*.

This strategy could avoid the models from learning the trick that identifying replaced words as the answers.

B.2 Results and Analysis

Shown in Figure 7, we can see that, when constructing the challenging questions with entity substitution, both BiDAF and BERT model perform comparably between challenging and *shortcut* test questions with 100% challenging questions in training. These results provide evidence that, after the substitution, models could not use guessing as an alternative solutions to the paraphrasing skill.

We conduct similar experiments in §3, §5 on *QWM/subs-Para*, which is shown in Figure 7 and Figure 8, respectively. The tendency could also support our previous findings. For example, the larger *shortcut* ratio expands the performance gaps between challenging and *shortcut* questions in Figure 7.

Handling Cross- and Out-of-Domain Samples in Thai Word Segmentation

Peerat Limkonchotiwat
School of Information Science and Technology, VISTEC, Thailand
Peerat.l_s19@vistec.ac.th

Wannaphong Phatthiyaphaibun
Faculty of Interdisciplinary Studies, Khon Kaen University, Thailand
wannaphong@kkumail.com

Raheem Sarwar
RGCL, University of Wolverhampton, United Kingdom
R.Sarwar4@wlv.ac.uk

Ekapol Chuangsuwanich
Department of Computer Engineering, Chulalongkorn University, Thailand
ekapolc@cp.eng.chula.ac.th

Sarana Nutanong
School of Information Science and Technology, VISTEC, Thailand
snutanon@vistec.ac.th

Abstract

While word segmentation is a solved problem in many languages, it is still a challenge in continuous-script or low-resource languages. Like other NLP tasks, word segmentation is domain-dependent, which can be a challenge in low-resource languages like Thai and Urdu since there can be domains with insufficient data. This investigation proposes a new solution to adapt an existing domain-generic model to a target domain, as well as a data augmentation technique to combat the low-resource problems. In addition to domain adaptation, we also propose a framework to handle out-of-domain inputs using an ensemble of domain-specific models called *Multi-Domain Ensemble* (MDE). To assess the effectiveness of the proposed solutions, we conducted extensive experiments on domain adaptation and out-of-domain scenarios. Moreover, we also proposed a multiple task dataset for Thai text processing, including word segmentation. For domain adaptation, we compared our solution to the state-of-the-art Thai word segmentation (TWS) method and obtained improvements from 93.47% to 98.48% at the character level and 84.03% to 96.75% at the word level. For out-of-domain scenarios, our MDE method significantly outperformed the state-of-the-art TWS and multi-criteria methods. Furthermore, to demonstrate our method's generalizability, we also applied our MDE framework to other languages, namely Chinese, Japanese, and Urdu, and obtained improvements similar to Thai's.

1 Introduction

Word segmentation (WS) is a crucial upstream process for most *natural language processing* (NLP) tasks such as *named entity recognition* (NER), *machine translation* (MT), and *part-of-speech tagging* (POS). Nguyen et al. (2017) showed POS performance increased from 87% to 93% when the WS was improved. WS can also enhance the performance of MT, such as the work done by Chang et al. (2008) for Chinese-English MT.

While word segmentation is considered a solved problem in many languages, the task is still a challenge in continuous-script languages. A great number of writing systems have no word boundary, e.g., Thai, Chinese, and Japanese. Deep learning has been effective in performing WS in these languages. However, it requires a large amount of training data to construct a reliable model, which can be a limitation for low-resource languages like Thai and Urdu. Furthermore, like other NLP tasks, word segmentation is domain-dependent (Fu et al., 2020). To handle a variety of data domains, there should be a substantial amount of data for each of them, exacerbating the low-resource problem. To make the matter worse, we may also need to handle input from a completely unseen domain.

In this paper, we propose a framework to address two domain dependency problems: (i) how to effectively construct a WS model to handle input from a given domain in a data-poor setting; (ii) how to effectively handle out-of-domain input. To address the first problem, we propose a new domain adaptation solution based on the concept of stacked ensemble (SE) learning (Limkonchotiwat et al., 2020) and data augmentation. To handle out-of-domain input, we use an ensemble of domain-specific models to produce predictive results.

The crux of our proposed method lies in the following technical contributions:

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1003–1016
August 1–6, 2021.

- We introduce multiple deep learning models following the concept of SE to construct domain-specific models that obtain better performance than the original SE and existing techniques in domain-adaptation problems. We call this technique *Deep Stacked Ensemble* (DSE).
- To make sure that each domain has sufficient data to build an accurate model, we design a data augmentation approach which consists of two techniques to generate hard-to-segment and semi-hard-to-segment samples to help improve the performance based on *Masked Language Model* (MLM).
- We use multiple domain-specific models and a result aggregation module to form an ensemble learning framework addressing the out-of-domain problems. We call this method *Multi-Domain Ensemble (MDE).*
- Furthermore, we propose a multiple task dataset called "*VISTEC-TP-TH-2021*", a social media dataset for Thai text processing, annotated for four text processing tasks: word segmentation, named-entity boundary, and misspelling detection and correction.

To assess the effectiveness of our approach, we compare our method with competitors in domain adaptation and out-of-domain scenarios on Thai, Chinese, Japanese, and Urdu. Experimental results showed that DSE improved the performance of the state-of-the-art Thai word segmentation (TWS) from 93.47% and 84.03% to 96.67% and 91.51% at character and word levels in domain adaptation settings. With the proposed data augmentation approach, our domain-specific model has improved even further at both character and word levels. For out-of-domain scenarios, our MDE framework outperformed the state-of-the-art TWS and multi-criteria baseline at character and word levels. Moreover, we applied our framework to Chinese, Japanese, and Urdu which resulted in improvement showing the applicability of our method to other languages. We make our code available at: `github.com/mrpeerat/OSKut`

2 Related Work

In this section, we discuss literature related to our investigation, namely *ensemble learning*, *domain adaption*, and *data augmentation*.

Ensemble Learning. Recently, considerable research attention has been dedicated to applying ensemble learning to boost the performance obtained from individual models (Sikdar and Gambäck, 2017; Chen et al., 2020a; Kuwabara et al., 2020) and to introduce previously ignored features for ensemble models such as provenance information in slot filtering (Viswanathan et al., 2015).

Several studies have used ensemble methods to boost the accuracy in WS. For example, Liu and Lin (2014) proposed a probabilistic ensemble learning framework using multiple weak word segmenters to form a strong segmenter. Moreover, Min et al. (2015) proposed an ensemble learning model to address the word segmentation and Part-of-Speech tagging problems by combining both discriminative and generative methods.

Domain Adaptation. Several WS studies proposed techniques to adapt the data distribution from one domain to another (Zhang et al., 2013; Ding et al., 2020). Another popular approach is to add new features or change network architectures of the target model (Monroe et al., 2014; Liu et al., 2014; Bao et al., 2017; Huang et al., 2020).

Ding et al. (2020) presented a semi-supervised approach for performing Chinese WS on a new domain by using adversarial training to help learn the difference between the source and target domain. Recently, Limkonchotiwat et al. (2020) proposed a filter-and-refine solution based on the stacked ensemble (SE) to convert a base model to a target domain. The SE consists of a domain-generic base model and a domain-specific model that analyzes the output of the domain-generic model and revises the segmentation. The method achieved similar performance to traditional transfer learning methods while requiring no access to the domain-generic model weights.

Data Augmentation and Self-Supervised learning. Word segmentation for low-resource languages is a challenging task due to the data limitation. Most Thai WS models report below 90% accuracy in domain-adaptation settings (Kittinaradorn et al., 2019; Chormai et al., 2020). Many researchers proposed data augmentation methods for Asian languages to increase the performance of WS models by using existing models' output as input to new models such as synthetic data, entropy parser, and character embedding (Zheng et al., 2018; Wang et al., 2019; Fung et al., 2004).

With the advent of large language models (Devlin et al., 2018; Yang et al., 2019; Brown et al., 2020), we have been witnessing an explosion in self-supervised learning techniques. Data augmentation methods such as the *Masked Language Model* (MLM) using BERT (Devlin et al., 2018) allow us to generate new sentences that are similar to real data by randomly selecting words in a sentence to replace them with new words (Chen et al., 2020b; Liao et al., 2020). Yavuz et al. (2020) proposed MaskAugment, a controllable mechanism and augmentation method that used a pre-trained BERT model to replace words in a sentence. The method is used in an unsupervised teacher-student framework to improve domain adaptation for dialog act task. Furthermore, Li et al. (2020) proposed a MLM-based augmentation method that could also preserve the underlying labels of the sentence in the aspect term extraction task.

Out-Of-Domain Scenarios. While domain adaptation presents a useful paradigm to adjust an existing model to a target domain, it is impracticable to anticipate all different input types in advance. Hence, the ability to handle samples from unseen domains (i.e., out-of-domain samples) is critical to the solution's performance. For example, Wagner et al. (2020) proposed utilizing treebank vectors and a method to interpolate a prediction from existing treebank vectors to handle out-of-domain input samples. Ng et al. (2020) proposed a solution utilizing data augmentation to generate training samples to diversify the training set so that the model can handle out-of-domain samples better.

Discussion. For domain adaptation, an ensemble learning method such as SE (Limkonchotiwat et al., 2020) provides a flexible framework for adapting any base model to a target domain. We hypothesize that we can improve the accuracy of SE by introducing a deep learning architecture at the domain-specific part. However, this adjustment would require a larger amount of data for each domain than the original SE method which uses a traditional Conditional Random Field (CRF) model (Lafferty et al., 2001). To tackle this problem, data-augmentation presents an avenue to address the data requirements. Regarding out-of-domain scenarios, we hypothesize that an ensemble of domain-specific models can be used to boost the accuracy of out-of-domain situations. This is the first WS work to address this problem without using any out-of-domain data.

3 Methodology

In this section, we present the overview of domain adaptation in Section 3.1. We present our domain adaptation solution based on the concept of stacked ensemble (SE) learning (Limkonchotiwat et al., 2020) and an MLM-based data augmentation method in Section 3.2. Section 3.3 presents how multiple domain-specific models can work as an ensemble to support out-of-domain scenarios.

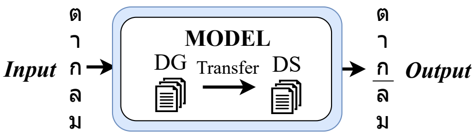

(a) Domain-specific model with transfer learning.

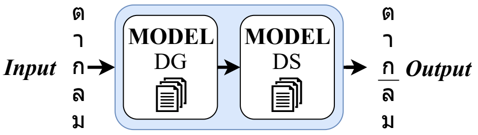

(b) Domain-specific model with stacked ensemble.

Figure 1: Comparison between transfer Learning (TL) and SE to build domain-specific (DS) models. For domain-specific with transfer learning, there is one model that is first trained on domain-generic (DG) data then domain-specific data. For SE there are two models. The first is a domain-generic model that feeds initial prediction results to a second domain-specific model.

3.1 Overview of Domain Adaptation

A popular method to construct a *domain-specific* (DS) model is to adapt from a *domain-generic* (DG) base model using transfer learning as shown in Figure 1a. Stacked ensemble learning presents an alternative when making changes to the base model is impossible. Unlike transfer learning, the SE model consists of two parts: a domain-generic model and a domain-specific model as shown in Figure 1b. The domain-specific model takes the output predictions from the domain-generic model to make better predictions on the target domain. Before feeding into the domain-specific model, there is also a filter-and-refine stage where only uncertain predictions are re-visited by the domain-specific model. Only the domain-specific model is trained on the target data, while the base model is left untouched.

The main advantages of SE over TL are as follows: (i) the architecture of the domain-specific model can be selected independently of the existing domain-generic one; (ii) it is able to handle models where we cannot adjust their weights, i.e., black boxes. Consequently, we adopt SE as our approach to tackling the domain adaptation problem.

3.2 Deep Stacked Ensemble (DSE)

As stated earlier, SE allows us to introduce a new architecture to handle domain-specific input. To exploit this advantage, we introduce the *Bidirectional Long Short-Term Memory* (Bi-LSTM) with *Attention mechanism* to the current state-of-the-art TWS architecture (Kittinaradorn et al., 2019). We call our proposed domain adaptation method *Deep Stacked Ensemble* (DSE).

Figure 2 shows the structure of the domain specific part of our solution. There are three main kinds of features. A character n-gram is passed through a CNN following Kittinaradorn et al. (2019) to create an embedding vector (shown in blue). A character type n-gram which indicates whether a character is either a vowel, digit, special character, or an English character, is turned into an embedding vector (shown in red). Lastly, we use probability and entropy values from the domain-generic model, which indicates whether a character is a start or end of a word or not in a dictionary, as the additional features (colored as green). We then concatenate all of the embeddings and feed them to the Bi-LSTM layer (Hochreiter and Schmidhuber, 1997; Ma et al., 2018).

The Attention model is connected to the Bi-LSTM output layer for improved accuracy because the attention mechanism is effective at capturing long-range dependencies (Duan and Zhao, 2020). The attention layer is followed by a fully connected network that ends with a single sigmoid output for Thai and Chinese (boundary or not) and a softmax output for Japanese and Urdu (the beginning, middle, or end of a word, or a word with a single character).

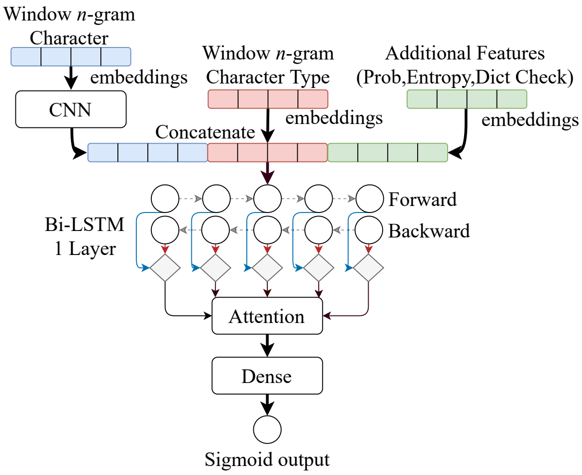

Figure 2: The domain-specific model of DSE.

Ablation studies, results given in Table 11, show that each component in the domain-specific model improves the performance incrementally. Unlike the original SE that relies only on the CRF as the domain-specific model, our deep learning approach to construct the domain-specific model can capture intricate WS patterns in the domain better than the original SE and transfer learning method.

However, unlike deep learning approaches, the classical machine learning approach, i.e., CRF, does not require a large amount of training data. To handle this problem, we propose the data augmentation technique at the character level. This can increase the amount of training data and thus improves the performance significantly.

Data Augmentation The main advantage of using a separate model for each domain is the ability to handle contradicting segmentation conditions from different domains (Fu et al., 2020). However, this approach requires a substantial amount of data in each domain as stated earlier. To mitigate this problem, we also propose two data augmentation methods based on the *Masked Language Model* (MLM) WangchanBERTa (Lowphansirikul et al., 2021) trained on Thai Wikipedia Dump. As shown in Figure 3, we mask words based on the output of the domain-generic model. The output posteriors from the model are used to compute the character-level entropy values. Then, the values are summed together to represent the score for each word. We select the words with the highest scores to mask in order to perform data augmentation. This is done to favor long words, since long words are harder to segment. We select the the top-k words to mask and replace them (substitution) using MLM. This a pretrained process to ensure the generation of hard-to-segment sentences. We also introduce semi-hard-to-segment samples by preferring word insertion after the word (rather than substitution). The same MLM is used to perform next word prediction instead of masked prediction. The ratio between hard-to-segment and semi-hard-to-segment is 80:20. This is found via grid search (see Table 13).

The insertion method gives the best performance compared with other semi-hard-to-segment generation methods (see the results in Table 13). The entropy selection method, compared with competitive selection methods in Table 12, shows that our method has the best performance for all Top-k selection and average scores.

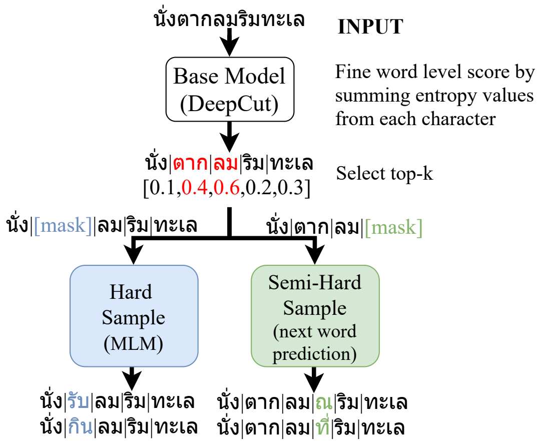

Figure 3: Overview of our data augmentation pipeline.

3.3 Muti-Domain Ensemble (MDE)

It is unrealistic to expect that the training and test distributions always match. Getting new training data for the out-of-domain scenarios can be expensive and time consuming (Ng et al., 2020; Liu et al., 2019). In such cases, transfer learning or the previously described DSE method are not sufficient.

We propose a framework, which utilizes an ensemble of domain-specific models to handle out-of-domain samples, called *Muti-Domain Ensemble (MDE)*. Figure 4 presents the structure of MDE.

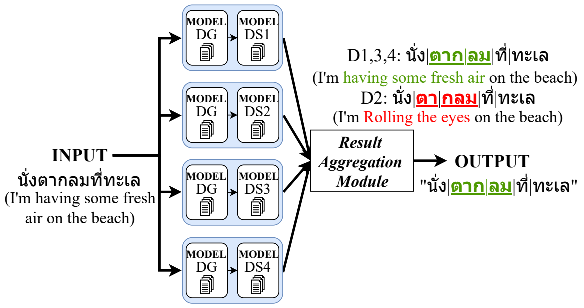

Figure 4: *Multi-Domain Ensemble (MDE).*

The framework consists of multiple word segmentation models, where each model is adapted to a specific target domain (except for the out-of-domain data) using the method described in Section 3.2. Results from multiple word segmentation models are combined using a result aggregator to form the final prediction. In this investigation, we formulate two result aggregation strategies as follows. (i) We compute a simple average of the posterior output from each model. Then, we predict the class that has the highest probability: this is a basic method for ensemble modeling (*Avg*); (ii) We calculate the entropy from each model based on their posterior distribution output. We then choose the prediction of the model with the lowest entropy and we call it *Min Entropy (ME)*.

The results of the MDE framework and aggregation module given in Tables 5 and 10 show that the entropy method performs better than the basic method and improves the performance significantly over other models in out-of-domain scenarios.

4 Performance Evaluations on Thai Word Segmentation

In this section, we report results from experimental studies on four *Thai word segmentation* (TWS) benchmark datasets. The studies are organized as follows. (i) we compare our method with competitive methods on domain adaptation; (ii) we show the effect of the data augmentation technique on domain adaptation; (iii) we report the results on out-of-domain setups; (iv) we show the effect of WS in downstream tasks.

Note that experimental studies on Chinese, Urdu, and Japanese are presented in Section 5.

4.1 Experimental Setup

Competitive Methods. We evaluate our proposed solution against two state-of-the-art methods namely DeepCut (DC) (Kittinaradorn et al., 2019) and AttaCut (AC) (Chormai et al., 2020). These methods are based on the *Convolutional Neural Network* (CNN) and trained on a generic corpus (BEST2009 (Boriboon et al., 2009)). For domain adaptation experiments, we also applied the concept of *Transfer Learning* (TL) to adapt DC and AC to the target corpora, and we call these adaptations TL-DC and TL-AC, respectively. Similarly, for the Stacked Ensemble Filter-and-Refine (SEFR) method (Limkonchotiwat et al., 2020), we created two variants, SE-DC and SE-AC, using DC and AC as the base model, respectively. For the evaluation of our method, *Deep Stack Ensemble* (DSE), we followed the same principle and created two variants DSE-DC and DSE-AC based on DC and AC, respectively.

Evaluation Metrics. We use F1 score as the evaluation metric for the TWS task at character and word levels to avoid the overestimation of TWS (Chormai et al., 2020; Limkonchotiwat et al., 2020).

Parameter Settings. In these experiments, we used grid search on 4 parameters including Bi-LSTM nodes, attention nodes, optimizer, and top-k inside the domain-specific model. We started the learning rate at 0.01 on an optimizer. For every 10 steps where the loss did not decrease, the learning rate was multiplied by a factor of 0.1. We set the number of training epochs to 300 with an option of early stopping. For the CNN layer and character embedding settings, we followed Kittinaradorn et al. (2019). We tuned the top-k value of the filtering system in a domain-specific model to be the same as the original SE. For the top-k value in the out-of-domain scenarios, we used the same k for all domain-specific models in the domain adaptation settings. Lastly, we tuned all of the parameters by using 10% of training data of the target domain. The hyper-parameters and their values are given in Table 1.

Hyper-parameters	Values for grid search
Optimizer	[Adam,RMSprop]
Learning Rate	0.01
Bi-LSTM nodes	[128, 160, 192, 224, 256]
Attention nodes	[32, 64, 96, 128, 160]
Top-k	[1-100]

Table 1: Hyper-parameters list.

4.2 Datasets

Benchmark Datasets. Our benchmark corpora can be seen in Table 2. They vary in domain and size from very small (Wisesight (Suriyawongkul et al., 2019) social media domain), moderate (TNHC (Sawatphol, 2019) classical literature), and large (LST20 (Boonkwan et al., 2020) generic).

Lang	Corpora	# Sentences	# Words
TH	Wisesight	1K [0.16K]	22K [3.9K]
TH	TNHC	13K [7K]	374K [239K]
TH	LST20	63.3K [5.2K]	2.7M [207K]
TH	VISTEC	39.8K [9.9K]	2.7M [690K]

Table 2: TWS corpora (# Training [# Testing]).

New Dataset. Due to social media data being underrepresented and difficult (Medina Serrano et al., 2020; Benton et al., 2017), it is challenging to improve the performance of models with only 997 training sentences. Most TWS models (Kittinaradorn et al., 2019; Chormai et al., 2020) performed under 82% in out-of-domain social media scenarios (Wisesight). To address this problem, we introduce a new dataset called "*VISTEC-TP-TH-2021*[1]" (VISTEC), which consists of 49,997 text samples from Twitter (2017-2019). VISTEC corpus contains 49,997 sentences with 3.39M words where the collection was manually annotated by linguists on four tasks, namely word segmentation, misspelling detection and correction, and named entity recognition. In the data collection process, we focused on the longest sentences to create a more challenging dataset due to the fact that long sentences made the model's performance decrease significantly compared with short sentences in the same domain (Section 4.3). The Out-of-Vocabulary rate on the test set is 13.65%.

We followed Boriboon et al. (2009) for the word and named entity tasks annotation guideline. We also included new guidelines about word editing criteria for misspelt words such as words used on the internet (Netspeak), transliterated loanwords, abbreviations, and shortened words, by using the Royal Institute Thai dictionary. We compared our dataset to the biggest Thai social media dictionary (Horsuwan et al., 2020) and found 79K words that did not appear in the dictionary.

4.3 Domain Adaptation

Without Data Augmentation. We evaluate the performance of our domain-specific model against competitive methods in four TWS benchmark corpora, WS160, TNHC, LST20, and VISTEC. The experimental results are given in Table 3. The competitive methods are defined in Section 4.1.

The DSE-DC (DeepCut) outperformed the strongest base model, DC, by 3.2% and 7.2% on WS160, 6.23% and 13.74% on TNHC, 4.41% and 10.18% on LST20, and 4.59% and 11.13% on VISTEC at character and word levels, respectively. Our domain-specific model also outperformed the original SE by 2.16% and 6.46% on SE-DC and 1.87% and 4.88% on SE-AC (AttaCut) at character and word levels on all setups. More importantly, our domain-specific model outperformed TL (transfer learning) methods showing the strength of our DSE model.

As expected, the newly constructed TWS social media dataset (VISTEC) shows that even TL-DC performed below 91% at word level, a large drop from the 96% achieved in the generic domain LST20 corpus. Also, the VISTEC dataset creates a new challenge for the social media domain. Comparing the WS160 and VISTEC datasets, the AC's performance decreased from 93.5% to 91.47% and 84.04% to 79.30% at the word level and the character level, respectively.

With Data Augmentation. In this experiment, we show the effect of the data augmentation in domain adaptation settings for different amounts of

---

[1] `https://github.com/mrpeerat/OSKut/tree/main/VISTEC-TP-TH-2021`

Method	WS160		TNHC		LST20		VISTEC	
	Char	Word	Char	Word	Char	Word	Char	Word
DC	93.47	84.03	89.48	75.40	94.60	87.15	92.77	81.78
AC	93.50	84.04	88.82	73.71	95.24	87.21	91.47	79.31
TL-DC	96.30	90.60	95.43	88.60	98.63	96.30	96.78	90.99
TL-AC	94.10	85.00	90.57	77.54	98.04	94.77	95.47	89.27
SE-DC	95.20	86.90	95.20	84.10	94.96	87.72	94.76	86.33
SE-AC	94.50	85.60	93.70	83.90	96.30	89.87	93.86	84.43
DSE-DC	**96.67**	**91.51**	**95.71**	**89.14**	**99.01**	**97.33**	**97.36**	**92.91**
DSE-AC	94.57	86.24	95.51	88.52	98.46	95.79	97.31	92.78

Table 3: Performance comparison on TWS in domain adaptation settings. TL, SE, and DSE models used target domain data besides BEST2009. DC = DeepCut, AC = AttaCut, TL = transfer learning, SE= stacked ensemble, and DSE = deep stacked ensemble.

adaptation data. We report the findings of the data augmentation process on 2 corpora, i.e., Wisesight (social media domain) which is the smallest corpus and LST20 which is the largest generic domain corpus LST20 (see Table 2). We fixed the top-k value in the data augmentation step at 60% and 10% of the segmentation predictions of the Wisesight and LST20 corpora, respectively. This value is found via grid search (see Table 12). We then use these augmented data with TL, SE, and DSE.

As shown in Table 4, the data augmentation process can improve the performance in the small corpus, i.e., Wisesight (WS160). DSE-DC (DeepCut) outperformed the base model by 5.01% and 12.72% at character and word levels. Also, DSE-DC outperformed TL-DC by 1.39% and 3.36% at the character and word levels respectively.

Method	WS160 (F1)		LST20 (F1)	
	Char	Word	Char	Word
DC	93.47	84.03	94.60	87.15
AC	93.50	84.04	95.24	87.21
TL-DC	97.69	93.96	98.11	94.00
TL-AC	97.59	94.57	97.45	93.25
SE-DC	95.08	86.37	96.47	90.40
SE-AC	94.66	86.36	96.28	89.77
DSE-DC	98.48	96.75	**98.67**	**97.03**
DSE-AC	**98.60**	**96.99**	98.61	96.18

Table 4: Performance on augmented WS160 and LST20.

However, since LST20 is sufficiently large, the augmentation did not produce performance improvement with respect to the model constructed using the original data only.

4.4 The Effect of Data Augmentation in Insufficient Data Scenarios

In this experiment, we evaluated the transfer learning (TL) and our method (DSE) trained on a varying numbers of sentences ranging from 100 to 1000 on the large datasets TNHC, LST20, and VISTEC to show the effectiveness of data augmentation in the insufficient data scenarios. As can be seen from Figure 5, the data augmentation improved the performance by 0.77% on average for TNHC, 1.55% for LST20, and 0.19% for VISTEC using DSE on the proposed data augmentation technique. Also, the transfer learning F1 performance is improved by 0.14% on average for TNHC and 0.57% for VISTEC. However, the performance of transfer learning on the LST20 data augmentation technique did not improve on this method as the baseline model (DeepCut) was trained on the same domain as the LST20 corpus. The performance of transfer learning in this setting is similar to the LST20 transfer learning model in Table 3.

The results of our method in insufficient data scenarios show that we improved the performance using the proposed data augmentation method when the original data is insufficient. Also, the best number of sentences for the augmentation technique in transfer learning is between 100 to 500 sentences and for our method is 500 to 1,000 sentences.

4.5 Experiments on Out-of-Domain Scenarios

In this experiment, we evaluated our *Multi-Domain Ensemble* (MDE) framework against two methods namely, DC trained on BEST2009 and *Multi-Criteria* (MC). MC is a multi-task model which learns multiple segmentation criteria from different domains jointly use shared layers (Chen et al., 2017). For MC and MDE, the target domains were left out from the training and the models are trained on the remaining domain.

As shown in Table 5, the performance improvements on Wisesight and TNHC were statistically significant ($P<0.001$ using McNemar's test) compared with MDE-ME and DC. Moreover, in comparison to DC, the performance improvement provided by MDE-Avg was also statistically significant on TNHC. As a result of MDE framework, we improved the performance from the base model (DC) at character and word level by 1.17% and 3.53% on WS160, 2.97% and 6.77% on TNHC, 0.26% and 0.42% on LST20, and 0.68% and 1.17% on VISTEC. Moreover, our MDE framework also outperformed the MC model in this experiment with significant results. In addition, the ME (Min Entropy) can improve the performance better than

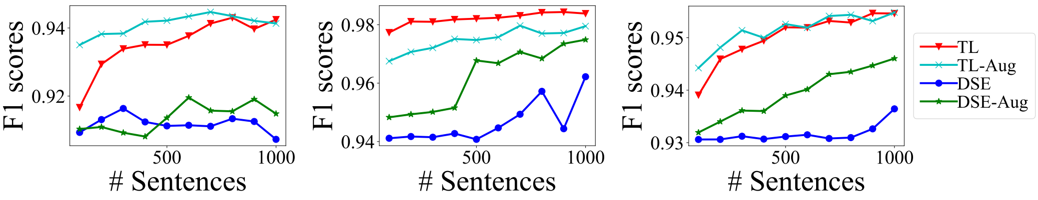

(a) TNHC (b) LST20 (c) VISTEC

Figure 5: The effectiveness of Data Augmentation with limitation of training data with DSE and TL at Chracter-level F1.

the Avg except on TNHC.

Method	WS160		TNHC		LST20		VISTEC	
	Char	Word	Char	Word	Char	Word	Char	Word
DC	93.47	84.03	89.48	75.40	94.60	87.15	92.77	81.78
MC-Avg	88.59	72.30	84.96	72.36	87.55	66.52	89.67	70.38
MC-ME	91.59	84.80	85.13	72.63	91.10	80.70	90.81	77.13
MDE-Avg	93.85	84.64	**92.45**	**82.17**	94.81	87.54	93.17	82.71
MDE-ME	**94.64**	**87.56**	90.29	76.98	**94.86**	**87.57**	**93.45**	**83.53**

Table 5: Out-of-domain experimental results when the base model of MDE is DC (DeepCut), ME = Min Entropy.

As mentioned earlier, word segmentation is a domain-dependent task and we cannot expect the input to always be in domain. A model that can robustly handle the out-of-domain scenarios is desirable. Even with the improvement gained by our proposed solution, the gap between out-of-domain and domain adaptation is still large, showing potential for further investigation. In the next experiment, we show the effect of the data augmentation on downstream tasks.

4.6 The Effect of Word Segmentation and Data Augmentation on Downstream Tasks

Previously, we showed the proposed data augmentation improved the performance of TWS in the domain adaptation settings. In this experiment, we applied TWS to downstream tasks such as named entity recognition (NER), text classification, and sentiment analysis compared with the TWS base model (DC and AC), TL, DSE, and DSE with augmented data. For the text classification experiments, we use Wongnai corpus and Wisesight corpus for sentiment analysis. The exact model setting and evaluation metric follow Thai classification benchmark [2]. For the NER experiment, we used NCRF++ (Yang and Zhang, 2018) trained with data from Nutcha (2016)'s work. We trained our DSE and competitive methods (except the baseline model) on the Wisesight corpus to show the performance of the proposed augmentation technique.

The results are given in Table 6. When the downstream tasks are not dependent on WS performance, the results one similar i.e., text and sentiment classification tasks. On the other hand, when the downstream task is dependent on WS performance, i.e., NER, we can significantly improve the downstream task. For example, we improved the performance of DSE-DC from 93.47% to 96.67% at the character level, and when combined with data augmentation, increased the accuracy to 98.48%. As a result, the F1 score in the NER task increased from 63.46% to 72.27%.

Method	TC	SA	NER
DC	57.10	71.55	63.46
AC	57.20	71.66	72.98
TL-DC	57.26	72.38	61.70
TL-AC	56.72	71.61	64.28
DSE-DC	57.04	**72.63**	71.47
+ Augment	**58.01**	72.27	72.27
DSE-AC	57.01	72.35	66.60
+ Augment	56.99	72.41	**73.47**

Table 6: The effect of TWS data augmentation on downstream tasks. All models were trained on Wisesight.

5 Chinese, Urdu, and Japanese Word Segmentation

In this section, we demonstrate the generalizability of our method on Chinese word segmentation (CWS), Urdu word segmentation (UWS), and Japanese word segmentation (JWS). For Chinese and Urdu, we performed in-domain experiments showing the effectiveness of DSE over SE and other competitive baselines. For Japanese and Chinese, we show the effectiveness of DSE and MDE over baselines in domain adaptation and out-of-domain settings. The corpora used in these experiments are shown in Table 7. The exact setup and evaluation metrics follow those of Limkonchotiwat et al. (2020).

---

[2] `https://github.com/PyThaiNLP/classification-benchmarks`

Lang	Corpora	# Sentences	# Words
C	AS (Emerson, 2005)	636K [13K]	4.8M [110K]
C	CITYU (CI)	46K [1.1K]	1.2M [28K]
C	MSR (MS)	56K [3.5K]	1.4M [91K]
C	PKU (PK)	7.7K [1.1K]	371K [48K]
J	GSD (GS) (Asahara et al., 2018)	7K [0.5K]	159K [12K]
J	Modern (MO)	0.6K [0.16K]	11K [2.6K]
J	PUD (PU)	0.7K [0.19K]	19K [5K]
U	UCRF (UC) (Bin Zia et al., 2018)	3.5K [825]	90K [21K]

Table 7: WS corpora (# Training [# testing]) for Chinese (C), Japanese (J), and Urdu (U).

In-Domain Experiments on Chinese and Urdu. For Chinese, we used PyWordSeg (Chuang, 2019) trained on each of the four Chinese corpora as our baseline models (*BL*). We then trained SE and DSE models on top of each baseline model for each domain. For Urdu, we used the model and dataset provided by Bin Zia et al. (2018). The results are summarized in Table 8.

Method	Chinese				Urdu
	AS	CI	MS	PK	UC
BL	97.09	94.30	87.29	85.76	93.73
SE	97.51	96.13	93.82	93.55	93.90
DSE	**97.85**	**96.67**	**96.89**	**95.85**	**95.14**

Table 8: F1-scores on in-domain CWS and UWS tasks for each corpus. BL refers to the baseline chosen for each language.

Both stacked ensemble methods improve over the baseline models in all settings showing the potential of stacked ensemble in improving WS performance. Moreover, the proposed DSE outperforms the original SE (Limkonchotiwat et al., 2020) significantly for MSR, PKU, and UCRF ($P<0.001$). The largest performance improvement is over 10% on the PKU corpus.

Domain Adaptation on Japanese. As in the TWS experiments, DSE can also be used for domain adaptation by training the domain-specific portion on the target domain. For this JWS task, we used Nagisa (Ikeda, 2018) trained on *Balanced Corpus of Contemporary Written Japanese* (BCCWJ) (Maekawa et al.) corpora as the base model. The domain-specific part of the SE was trained on the target corpus to create an adapted model. Note that the Nagisa model released does not lend itself for transfer learning because the authors did not provide the model weights. From Table 9 SE and DSE improves significantly over the baseline showing the effectiveness of SE in situations when one cannot perform typical transfer learning.

Method	Japanese		
	GS	MO	PU
BL	87.10	78.80	87.10
SE	90.11	90.27	91.76
DSE	**92.36**	**90.65**	**91.89**

Table 9: F1-scores on domain adaptation JWS tasks for each corpus.

Out-of-Domain Experiments on Chinese and Japanese. Multiple models can form an MDE to provide robustness in out-of-domain scenarios. For Chinese, we used PyWordSeg trained on the AS corpus as the base model. The MDE included two domains (non-target) and was tested on the left-out target domain.

Method	Chinese			Japanese		
	CI	MS	PK	GS	MO	PU
BL	92.51	83.92	82.21	87.10	78.80	87.10
MDE-Avg	88.90	89.15	89.77	87.11	79.36	87.42
MDE-ME	**93.98**	**93.01**	**93.56**	**87.12**	**79.39**	**87.44**

Table 10: F1-scores on out-of-domain CWS and JWS experiments.

Table 10 summarizes the results of the out-of-domain experiments. The MDE provides a minimal improvement over the baseline on JWS. We hypothesize that this is because two out of the three corpora are too small to train a reliable model. However, on Chinese the MDE provides large gains over the baseline with the min entropy method performing better than the simple averaging method.

6 Concluding Remarks

This investigation presents a set of solutions to address two domain dependency problems: handling cross-domain and out-of-domain samples. Our key findings are as follows. First, we applied deep learning to the original stacked ensemble method and obtained a significant improvement. Second, we show that data augmentation is an effective method to combat the low-resource limitation in domain adaptation. Third, we can use an ensemble of domain-specific models to obtain a performance improvement over each domain-specific model acting alone. Finally, in addition to Thai, we can apply the same principle to Chinese, Japanese, and Urdu and obtain similar improvements. As future work, we plan to experiment with novel techniques, i.e., Transformer and contrastive learning.

References

Masayuki Asahara, Hiroshi Kanayama, Takaaki Tanaka, Yusuke Miyao, Sumire Uematsu, Shinsuke

Mori, Yuji Matsumoto, Mai Omura, and Yugo Murawaki. 2018. Universal Dependencies Version 2 for Japanese. In *LREC*.

Zuyi Bao, Si Li, Weiran Xu, and Sheng Gao. 2017. Neural regularized domain adaptation for Chinese word segmentation. In *Proceedings of the 9th SIGHAN Workshop on Chinese Language Processing*, pages 11–20, Taiwan. Association for Computational Linguistics.

Adrian Benton, Glen Coppersmith, and Mark Dredze. 2017. Ethical research protocols for social media health research. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 94–102, Valencia, Spain. Association for Computational Linguistics.

Haris Bin Zia, Agha Ali Raza, and Awais Athar. 2018. Urdu word segmentation using conditional random fields (CRFs). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2562–2569, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Prachya Boonkwan, Vorapon Luantangsrisuk, Sitthaa Phaholphinyo, Kanyanat Kriengket, Dhanon Leenoi, Charun Phrombut, Monthika Boriboon, Krit Kosawat, and Thepchai Supnithi. 2020. The annotation guideline of lst20 corpus.

Monthika Boriboon, Kanyanut Kriengket, Patcharika Chootrakool, Sitthaa Phaholphinyo, Sumonmas Purodakananda, Tipraporn Thanakulwarapas, and Krit Kosawat. 2009. Best corpus development and analysis. In *2009 International Conference on Asian Language Processing*, pages 322–327. IEEE.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *WMT@ACL*, pages 224–232.

Jun Chen, Xiaoya Dai, Quan Yuan, Chao Lu, and Haifeng Huang. 2020a. Towards interpretable clinical diagnosis with Bayesian network ensembles stacked on entity-aware CNNs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3143–3153, Online. Association for Computational Linguistics.

Luoxin Chen, Xinyue Liu, Weitong Ruan, and Jianhua Lu. 2020b. Enhance robustness of sequence labelling with masked adversarial training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 297–302, Online. Association for Computational Linguistics.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for Chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203, Vancouver, Canada. Association for Computational Linguistics.

Pattarawat Chormai, Ponrawee Prasertsom, Jin Cheevaprawatdomrong, and Attapol Rutherford. 2020. Syllable-based Neural Thai Word Segmentation. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yung-Sung Chuang. 2019. Robust Chinese Word Segmentation with Contextualized Word Representations. *CoRR*, abs/1901.05816.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Ning Ding, Dingkun Long, Guangwei Xu, Muhua Zhu, Pengjun Xie, Xiaobin Wang, and Haitao Zheng. 2020. Coupling distant annotation and adversarial training for cross-domain Chinese word segmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6662–6671, Online. Association for Computational Linguistics.

Sufeng Duan and Hai Zhao. 2020. Attention is all you need for Chinese word segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3862–3872, Online. Association for Computational Linguistics.

T. Emerson. 2005. The second international chinese word segmentation bakeoff. In *SIGHAN@IJCNLP 2005*.

Jinlan Fu, Pengfei Liu, Qi Zhang, and Xuanjing Huang. 2020. RethinkCWS: Is Chinese word segmentation a solved task? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5676–5686, Online. Association for Computational Linguistics.

Pascale Fung, Grace Ngai, Yongsheng Yang, and Benfeng Chen. 2004. A maximum-entropy chinese parser augmented by transformation-based learning. *ACM Transactions on Asian Language Information Processing*, 3(2):159–168.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Thanapapas Horsuwan, Kasidis Kanwatchara, Peerapon Vateekul, and Boonserm Kijsirikul. 2020. A comparative study of pretrained language models on thai social text categorization. In *ACIIDS*.

Kaiyu Huang, Degen Huang, Zhuang Liu, and Fengran Mo. 2020. A joint multiple criteria model in transfer learning for cross-domain Chinese word segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3873–3882, Online. Association for Computational Linguistics.

Taishi Ikeda. 2018. nagisa: A Japanese tokenizer based on recurrent neural networks. `https://github.com/taishi-i/nagisa`.

Rakpong Kittinaradorn, Korakot Chaovavanich, Titipat Achakulvisut, Kittinan Srithaworn, Pattarawat Chormai, Chanwit Kaewkasi, Tulakan Ruangrong, and Krichkorn Oparad. 2019. DeepCut: A Thai word tokenization library using Deep Neural Network.

Ryosuke Kuwabara, Jun Suzuki, and Hideki Nakayama. 2020. Single model ensemble using pseudo-tags and distinct vectors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3006–3013, Online. Association for Computational Linguistics.

John Lafferty, Andrew Mccallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289.

Kun Li, Chengbo Chen, Xiaojun Quan, Qing Ling, and Yan Song. 2020. Conditional augmentation for aspect term extraction via masked sequence-to-sequence generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7056–7066, Online. Association for Computational Linguistics.

Yi Liao, Xin Jiang, and Qun Liu. 2020. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 263–274, Online. Association for Computational Linguistics.

Peerat Limkonchotiwat, Wannaphong Phatthiyaphaibun, Raheem Sarwar, Ekapol Chuangsuwanich, and Sarana Nutanong. 2020. Domain adaptation of Thai word segmentation models using stacked ensemble. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3841–3847, Online. Association for Computational Linguistics.

Junxin Liu, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2019. Neural chinese word segmentation with lexicon and unlabeled data via posterior regularization. In *The World Wide Web Conference*, WWW '19, page 3013–3019, New York, NY, USA. Association for Computing Machinery.

Wuying Liu and Li Lin. 2014. Probabilistic ensemble learning for vietnamese word segmentation. In *Proceedings of the 37th International ACM SIGIR Conference on Research Development in Information Retrieval*, SIGIR '14, page 931–934, New York, NY, USA. Association for Computing Machinery.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 864–874, Doha, Qatar. Association for Computational Linguistics.

Lalita Lowphansirikul, Charin Polpanumas, Nawat Jantrakulchai, and Sarana Nutanong. 2021. Wangchanberta: Pretraining transformer-based thai language models.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese word segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. Balanced corpus of contemporary written Japanese. *Lang. Resour. Evaluation*.

Juan Carlos Medina Serrano, Orestis Papakyriakopoulos, and Simon Hegelich. 2020. NLP-based feature extraction for the detection of COVID-19 misinformation videos on YouTube. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.

Kerui Min, Chenggang Ma, Tianmei Zhao, and Haiyan Li. 2015. Bosonnlp: An ensemble approach for word segmentation and pos tagging. In *Natural Language Processing and Chinese Computing*, pages 520–526, Cham. Springer International Publishing.

Will Monroe, Spence Green, and Christopher D. Manning. 2014. Word segmentation of informal Arabic with domain adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 206–211, Baltimore, Maryland. Association for Computational Linguistics.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 1268–1283, Online. Association for Computational Linguistics.

Dat Quoc Nguyen, Thanh Vu, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. From word segmentation to POS tagging for Vietnamese. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 108–113, Brisbane, Australia.

Tirasaroj Nutcha. 2016. *A study of word sense discrimination in Thai using latent semantic analysis*, volume 25.

Jitkapat Sawatphol. 2019. Thai literature corpora. urlhttps://attapol.github.io/tlc.html.

Utpal Kumar Sikdar and Björn Gambäck. 2017. A feature-based ensemble approach to recognition of emerging and rare named entities. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 177–181, Copenhagen, Denmark. Association for Computational Linguistics.

Arthit Suriyawongkul, Pattarawat Chormai, Charin Polpanumas, and Ekapol Chuangsuwanich. 2019. Pythainlp/wisesight-sentiment: First release.

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond Mooney. 2015. Stacked ensembles of information extractors for knowledge-base population. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 177–187, Beijing, China. Association for Computational Linguistics.

Joachim Wagner, James Barry, and Jennifer Foster. 2020. Treebank embedding vectors for out-of-domain dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8812–8818, Online. Association for Computational Linguistics.

Xiaobin Wang, Deng Cai, Linlin Li, Guangwei Xu, Hai Zhao, and Luo Si. 2019. Unsupervised learning helps supervised neural word segmentation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7200–7207. AAAI Press.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Semih Yavuz, Kazuma Hashimoto, Wenhao Liu, Nitish Shirish Keskar, Richard Socher, and Caiming Xiong. 2020. Simple data augmentation with the mask token improves domain adaptation for dialog act tagging. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5083–5089, Online. Association for Computational Linguistics.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, Seattle, Washington, USA. Association for Computational Linguistics.

R. Zheng, M. Li, J. He, J. Bi, and B. Wu. 2018. Segmentation-free multi-font printed manchu word recognition using deep convolutional features and data augmentation. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6.

A Appendices

A.1 Ablation Studies

In this experiment, we show the effect of Bi-LSTM with attention mechanism on the performance of the TWS domain adaption problem with the help of an ablation study. This study was performed on the Wisesight corpus (test set) and used DC (DeepCut) as the base model. As can been seen from Table 11, each component significantly improved the performance from the base model.

System	F1 Char	F1 Word
Baseline	93.47	84.03
+ Additional Feature	94.27	85.39
+ Bi-LSTM	96.31	90.35
+ Attention	**96.67**	**91.51**

Table 11: TWS domain adaptation ablation studies.

A.2 How to Select a Word to Mask? Entropy vs Random vs Maximum Length Selections.

The Section 3.2 presented the way we select a word to augment and compared it against the traditional method i.e., random selection. In this study, we show the validation score of random, entropy, and maximum length selections in our data augmentation technique on the Wisesight corpus with DC base model by varying the k value from top-10% to 100% on the substitution method and fixed top-20% value for the insertion method.

The validation score results are given in Table 12, the best range top-k value for the data augmentation is 50% to 60%. The performance of entropy selection is better than competitive methods with reasonable results. Due to the fact that long words are harder to segment than short ones, the entropy selection method favors long words with a high uncertainty. Maximum length selection, gives a similar score with entropy selection due to the fact that high uncertainty score mostly comes from long words. Also, the best F1 score is obtained using top 60% not 100% as words in top-60% might have the most incorrect answers and bias from frequency word.

A.3 Ablation Studies For Different Semi-Hard-Sample Procedures

Section 3.2 mentioned a competitive method to produce semi-hard-to-segment samples. We use the semi-hard-to-segment method with substitution by fixing the k value at top-60% for substitution method and we vary k in the range of 10% to 100% on the semi-hard-to-segment methods to show the performance of each method. We show the validation score on Wisesight (training data) with character and word levels, respectively. The results are presented in Table 13. As can be seen, the insertion method reports the best performance on every top-k entropy selection. The deletion method is inappropriate due to the fact that we might delete some information in the training data.

Top-k Entropy	ES		RS		MLS	
Select	char	word	char	word	char	word
10	99.74	99.29	99.50	98.67	99.71	99.13
20	99.82	99.50	99.66	99.11	99.81	99.49
30	99.80	99.45	99.82	99.57	99.79	99.54
40	99.77	99.42	99.81	99.59	99.81	99.55
50	**99.90**	99.75	99.79	99.46	99.84	99.64
60	**99.90**	**99.77**	99.84	99.56	99.87	99.70
70	99.79	99.44	99.87	99.67	99.79	99.44
80	99.76	99.41	99.87	99.70	99.76	99.41
90	99.80	99.45	99.80	99.55	99.76	99.46
100	99.86	99.66	99.85	99.70	99.86	99.66
AVG	**99.81**	**99.51**	99.78	99.46	99.80	99.50

Table 12: Performance comparison (F1) on entropy selection (ES) vs random selection (RS) VS maximum length selection (MLS) on validation scores.

Top-k Entropy	Insertion		Deletion	
Select	char	word	char	word
10	99.79	99.54	99.80	99.61
20	**99.90**	**99.77**	99.77	99.40
30	**99.90**	99.65	99.75	99.32
40	99.81	99.67	99.79	99.39
50	99.85	99.69	99.64	98.96
60	99.86	99.62	99.54	98.71
70	99.88	99.66	98.93	97.00
80	99.84	99.67	98.97	97.15
90	**99.90**	99.66	97.84	94.11
100	99.88	99.62	93.84	84.31
AVG	**99.86**	**99.66**	98.79	96.80

Table 13: Performance comparison on insertion VS deletion to produce semi-hard-to-segment samples with WS160 (F1 validation score).

A.4 Error Analysis

We performed an error analysis on Wisesight (WS160) corpora for DC, SE-DC, and DSE-DC to investigate the improvement from the baselines as well as the benefits of our method in domain adaptation setups. We used the same setting as mentioned in Section 4.3. The samples presented here were randomly selected from the Wisesight validation set.

As shown in Figure 6, DSE-DC did better especially on compound words. However, all models still cannot properly handle the special character (+), since the character is rare in the WS160 corpus.

Actual: ไม่|รุ้|นะ|คับ|แพ้|น้ำหอม|หรอ|แบบ|นี้
DeepCut: ไม่|รุ้นะ|คับ|แพ้|น้ำ|หอม|หรอ|แบบ|นี้
SE-DeepCut: ไม่|รุ้นะ|คับ|แพ้|น้ำ|หอม|หรอ|แบบ|นี้
DSE-DeepCut: ไม่|รุ้|นะ|คับ|แพ้|น้ำหอม|หรอ|แบบ|นี้

Actual: แต่|เทียน่า|มัน|ขาย|ไม่|ดี| |นิสสัน|คง|เน้น|ทำ|อีโค|คา|เต็ม|ตัว|555+
DeepCut: แต่|เทีย|น่า|มัน|ขาย|ไม่|ดี| |นิสสัน|คง|เน้น|ทำ|อีโคคา|เต็มตัว|555|+
SE-DeepCut: แต่|เทีย|น่า|มัน|ขาย|ไม่|ดี| |นิสสัน|คง|เน้น|ทำ|อีโคคา|เต็มตัว|555|+
DSE-DeepCut: แต่|เทียน่า|มัน|ขาย|ไม่|ดี| |นิสสัน|คง|เน้น|ทำ|อีโค|คา|เต็ม|ตัว|555|+

Actual: อยาก|กิง|บาบีก้อน|บุฟ|อีก|อ่ะ| |คิดถึง|ที่|ปี|ที่|แล้ว|ไป|กิน|กะ|มึง|ง่ะ
DeepCut: อยาก|กิงบาบีก้อน|บุฟ|อีก|อ่ะ| |คิด|ถึง|ที่|ปี|ที่|แล้ว|ไป|กินกะ|มึงง่ะ
SE-DeepCut: อยาก|กิงบาบีก้อน|บุฟ|อีก|อ่ะ| |คิด|ถึง|ที่|ปี|ที่|แล้ว|ไป|กิน|กะ|มึงง่ะ
DSE-DeepCut: อยาก|กิง|บาบีก้อน|บุฟ|อีก|อ่ะ| |คิดถึง|ที่|ปี|ที่|แล้ว|ไป|กิน|กะ|มึง|ง่ะ

Figure 6: The example of segmentation results in WS160 validation dataset.

Sensei: Self-Supervised Sensor Name Segmentation

Jiaman Wu, Dezhi Hong, Rajesh K. Gupta, Jingbo Shang
Department of Computer Science and Engineering, University of California San Diego, CA, USA
Halıcıoğlu Data Science Institute, University of California San Diego, CA, USA
{j4wu, dehong, gupta, jshang}@eng.ucsd.edu

Abstract

Sensor names as alphanumeric strings typically encode their key contextual information such as their function or physical location. We focus here on sensors used in smart building applications. In these applications, sensor names are curated in a building vendor-specific manner using different structures and esoteric vocabularies. Tremendous manual effort is needed to annotate sensor nodes for each building or even to just segment these sensor names into meaningful chunks for intelligent operation of buildings. We propose here a *fully automated* self-supervised framework, Sensei, that can learn to segment sensor names without any human annotation. We employ a neural language model to capture the underlying structure in sensor names and then induce self-supervision based on information from the language model to build the segmentation model. Extensive experiments on five real-world buildings comprising thousands of sensors demonstrate the superiority of Sensei over baseline methods.

1 Introduction

Sensor name segmentation seeks to partition a sensor name string into semantic segments. It is an essential task to enable smart building applications (Weng and Agarwal, 2012) that critically depend upon understanding the context of sensory data. For example, to increase the flow of air to improve air quality, such an application will need to identify and locate the airflow controller for the specific area. Such a controller is identified as a control point associated with a specific actuator such as a variable air valve. This is typically done manually by searching appropriate strings in the name that refer to specific sensor or actuator type and/or its location. This information is encoded as a concatenation of *segments*. Correct segmentation of sensor names is a key first step. Note, we use sensor to refer to both sensors and actuators in a building application.

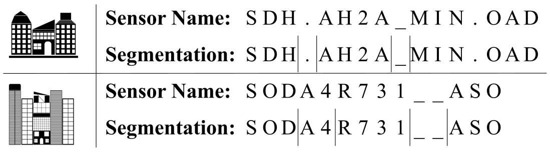

Figure 1: Sensor names adopt distinctive structures and vocabularies in different buildings, thus requiring manual effort to interpret.

Figure 1 shows examples of sensor names consisting of multiple segments, each encoding key context about the sensor (building name, location, sensor type, etc). Thus, the sensor name `SODA4R731__ASO` should be segmented as `SOD` (building name), `A4` (equipment id), `R731` (room id), and `ASO` (measurement type – area temperature setpoint). Note that the meanings of the same punctuation may vary; for example, '_' can be a delimiter or part of a segment.

Currently, sensor name segmentation requires domain knowledge and tedious manual effort due to its building-specific nature. Sensor names are created by building vendors, and as we see from Figure 1, in different buildings they usually adopt distinctive structures and vendor-specific esoteric vocabularies. Thus, constructing a sensor name segmentation model involves a building systems technician to comprehend these sensor names and then design rules to segment and annotate these names. There are no universal pre-defined parsing rules such as regular expressions for sensor names. This obscurity of sensor data remains a major obstacle to the wide adoption of smart building applications from both cost and efficiency perspectives (Bhattacharya et al., 2015a).

We need an automated solution for sensor name segmentation. Despite the recent progress in apply-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1017–1027
August 1–6, 2021. ©2021 Association for Computational Linguistics

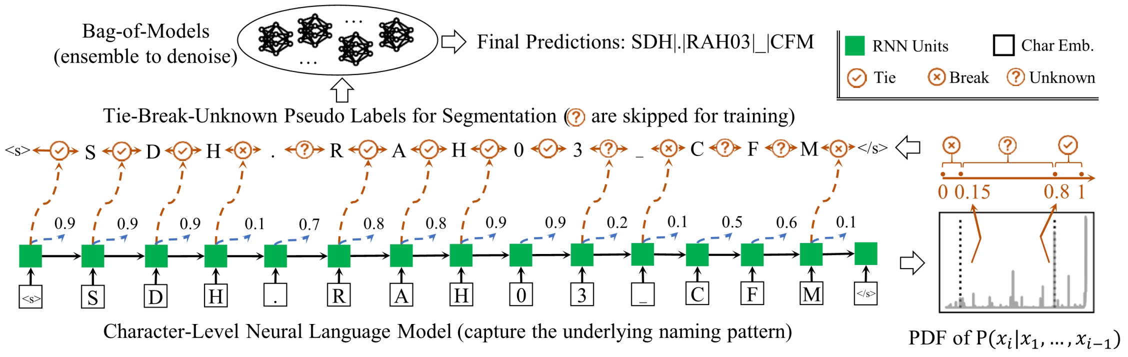

Figure 2: Overview of Sensei: We induce pseudo labels for segmentation using the transition probabilities from a character-level neural language model. Segmentation model training uses the hidden states from this model.

ing active learning (Schumann et al., 2014; Hong et al., 2015b; Balaji et al., 2015; Koh et al., 2018; Shi et al., 2019) and transfer learning (Hong et al., 2015a; Jiao et al., 2020) to sensor name interpretation, all these methods still require human annotation effort and thus they are not *fully* automated.

We propose here a novel self-supervised segmentation framework, Sensei, to segment sensor names into meaningful chunks *without any human effort*. This would enable understanding of important contextual information at scale and enable new regime of building applications. Figure 2 presents an overview.

Sensei builds upon the observation that sensor names are not randomly generated. Instead, installers of building management system would generally follow *some* underlying pattern for naming. For instance, in some buildings, the sensor name often starts with the building name, followed by the room id and type of measurement. Further, technicians would generally use similar phrases to express the same concept (e.g., "temperature" would be encoded as "T", "temp", or "ART"), at least within the same building. Based on this observation, we first employ a character-level neural language model (Karpathy et al., 2015) to capture the latent generative pattern in sensor names. This language model learns the probability of observing a character in the sensor name given all the preceding characters. Intuitively, the segment boundaries in a sensor name should highly correlate with this probability. Frequent transitions would have a higher probability than the infrequent ones, which might well imply the start of another segment. Therefore, we induce pseudo segmentation labels by setting a pair of thresholds on these transition probabilities, and then build a binary classifier to segment sensor names upon their contextualized representations produced by the language model. Since these pseudo labels may contain noise, we create an ensemble of independent classifiers, each trained on a uniformly random subset of the pseudo labels, in order to further improve the efficacy.

To the best of our knowledge, Sensei is the first framework for sensor name segmentation without human annotation. This paper makes the following contributions:

- We study an important problem of *fully automated* sensor name segmentation.
- We propose a novel self-supervised framework Sensei, which leverages a neural language model to capture the underlying naming patterns in sensor names and produces pseudo segmentation labels for training binary classifiers.
- We conduct extensive experiments on five real-world buildings comprising thousands of sensor names. Sensei on average achieves about 82% in F_1, roughly a 20-point improvement over the best compared unsupervised method.

Reproducibility. Our code and datasets are readily available on Github[1].

2 The Sensei Framework

Sensei consists of three steps:

- Train a neural language model (NLM) at the character level to capture the underlying naming patterns in sensor names;
- Generate `Tie-Break-Unknown` pseudo la-

---

[1] `https://github.com/work4cs/sensei`

bels using two thresholds, t_0 and t_1, decided by inspecting the distribution of transition probabilities (i.e., likelihood of observing the current character given the previous ones);

- Train a set of segmentation models based on the pseudo labels to mitigate the effect of noise in these labels.

We next elaborate on each step.

2.1 Language Model for Underlying Patterns

As sensor names are created by humans (e.g., a technician with knowledge about building particulars), they often follow a certain naming convention (e.g., start with the building name, then room id, and then type). The names also have an internal consistency, that is, within a building, segments of sensor names corresponding to the same kind of information (e.g., location or function) would use similar phrases; e.g., the concept of "room" would be encoded as "`RM`", "`R`", or similar variants. This prompts us to model the generative patterns in these names such that given the characters seen so far we can predict the next one. This coincides with the language modeling task in NLP.

Since the sensor name segmentation task works on characters, we adopt a popular character-level neural language model to capture the underlying sensor naming pattern, the classical Char-RNN (Karpathy et al., 2015) architecture, and use LSTM (Hochreiter and Schmidhuber, 1997) as the RNN model. Note that, our method is compatible with any character-level neural language models.

Given a character sequence of length N, $X = \langle x_1, x_2, \ldots, x_N \rangle$, the Char-RNN learns the probability of observing a character given all the previous characters, namely, $p(x_{i+1}|x_1, x_2, \ldots, x_i)$. During this process, we will obtain an embedding vector $\mathbf{x}_i$ for each character x_i, and a hidden state vector $\mathbf{h}_i$ after observing the characters from x_1 to x_i. A softmax layer is then applied to $\mathbf{h}_i$ to predict a distribution $\hat{\mathbf{p}}_\mathbf{i}$ over the entire vocabulary:

$$\hat{\mathbf{p}}_\mathbf{i}(c) = p(c|x_1, x_2, \ldots, x_i) = \frac{\exp\left(\mathbf{w}_c^\top \mathbf{h}_i\right)}{\sum_{c'} \exp\left(\mathbf{w}_{c'}^\top \mathbf{h}_i\right)},$$

where $\mathbf{w}_c$ is the linear transformation for character c. The cross-entropy between $\hat{\mathbf{p}}_\mathbf{i}$ and the one-hot encoding of x_{i+1} is used as the loss function for this character.

Given a building, we train the Char-RNN on all its sensor names. As each sensor name is independent of each other, we can have the same initial hidden state for each sensor name to ensure sensor names do not interfere with each other. Once the model converges, we apply it to all the sensor names to obtain the character transition probabilities, i.e., $\hat{\mathbf{p}}_\mathbf{i}(x_{i+1})$. The perplexity of the trained Char-RNN in our experiments is typically small (i.e., < 0.3 per batch with batch size 32). Therefore, we believe it captures the underlying naming pattern within the input building well.

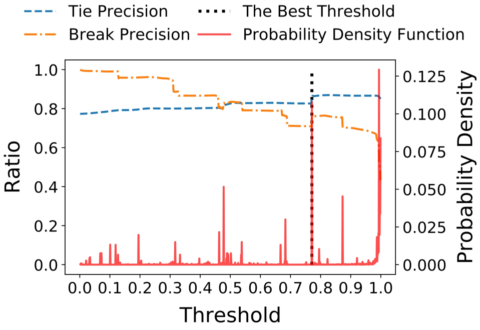

Figure 3: Plots of $\hat{\mathbf{p}}_\mathbf{i}(x_{i+1})$ histogram (grey bars) and Tie/Break precision curves for an example building. The "sweet spot", achieving a great balance between the tie- and break-precision scores, is highly aligned with the peak in the histogram.

2.2 Pseudo Labels from Transition Probabilities

Inspired by (Shang et al., 2018b), we use `Tie` and `Break` to decide the segmentation results. The transition between two adjacent characters (x_i, x_{i+1}) is labeled as (1) `Break` when we should segment after character x_i, or (2) `Tie` otherwise, denoting that the two successive characters belong to the same segment.

For a given character sequence $x_1, x_2, \ldots, x_N$, we hypothesize that the transition probability $\hat{\mathbf{p}}_\mathbf{i}(x_{i+1})$ obtained from Char-RNN is closely related to the `Tie/Break` relation between x_i and x_{x+1}. Intuitively, the Char-RNN model should produce a high likelihood for common transitions in sensor names, e.g., substrings for building name, room, and common sensor types. Therefore, when Char-RNN suggests a low transition probability, the transition is very likely to be a `Break`; otherwise, the possibility of a `Tie` becomes higher.

We empirically verify our hypothesis via data analysis of an example building as shown in Figure 3. We present the probability density from histogram of $\hat{\mathbf{p}}_\mathbf{i}(x_{i+1})$. In addition, based on the ground-truth segmentation results, we plot the Tie and Break precision curves w.r.t. different thresholds. The *Tie Precision* refers to the ratio of Tie transitions among all the transitions above a certain threshold, while the *Break Precision* refers to the ratio of Break transitions among all the transitions below a certain threshold. One can observe that the"turning" points on the break precision curve are highly correlated to the peaks in the histogram. An uptick on the break precision curve indicates that there might be abundant same patterns in the bin of probabilities, which typically are more likely to be `Ties`. Therefore, the thresholds should be below the probability bins with high frequency to classify the steep upslope regions as `Tie`.

If one wants to set up a single threshold on $\hat{\mathbf{p}}_{\mathbf{i}}(x_{i+1})$ to classify all transitions into {`Tie`, `Break`} in an unsupervised manner, the highest peak in the "confidence" interval $[0.550, 0.950]$ on the distribution (e.g., 0.771 in Figure 3) would be a good choice to achieve a high F_1 score. We generalize this threshold selection criterion to the other buildings, and as we shall demonstrate in our experiments, such a selection strategy gives results close to grid search that uses ground-truth labels.

In addition to `Tie` and `Break`, we mark those uncertain transitions as `Unknown`. We need to decide on two thresholds, t_0 and t_1, and categorize the transitions according to three transition probability intervals, $[0, t_0]$, (t_0, t_1), and $[t_1, 1]$, denoting `Break`, `Unknown`, and `Tie`, respectively, as the *pseudo labels*. We wish these pseudo labels would be of high accuracy while having a sufficient amount of labels. Based on our observations, the above single threshold criterion satisfies t_1. Considering that `Breaks` are considerably fewer than `Ties`, we should decide on a `Break` more carefully. The highest peak in a narrowed high precision interval $[0.050, 0.150]$ would be appropriate (e.g., 0.101 in Figure 3).

2.3 Ensemble to De-noise Pseudo Labels

There could exist errors in these automatically induced pseudo labels, so we leverage the idea of ensemble learning to mitigate the impact of such errors on the final predictions (Breiman, 1996). Specifically, we independently sample a subset of pseudo labels to train K binary classifiers and then average their predictions. In the pseudo labels, the number of `Tie` transitions is usually much higher than that of `Break`. To balance the training data, we sample $\epsilon \cdot M$ `Tie` and `Break` labels, respectively, from all the pseudo labels, where M is the number of `Break` transitions and ϵ is a small coefficient between 0 to 1 for sampling a subset (e.g., $\epsilon = 0.1$). Such a sampling strategy makes the label errors less likely to affect every binary classifier, so the final prediction becomes more accurate.

All types of binary classifiers could be used to construct the ensemble. We adopt a Multi-Layer Perceptron (MLP) as the binary classifier. For the i-th transition, we retrieve the hidden state vector $\mathbf{h}_i$ yielded by the Char-RNN and feed it as input to the MLP. The final prediction is the average of predictions from the K classifiers. As the training data is sampled in a balanced way, we simply use 0.5 as the threshold to decide on `Tie` or `Break`.

3 Experiments

We empirically evaluate Sensei on datasets from real-world buildings and discuss our results as well as findings of particular interest.

3.1 Datasets and Pre-processing

We collected the sensor names from five office buildings contracted with four different building vendors at three different sites located in different geographic regions. We also collected the character-level ground-truth labels of these names from their building vendors. We adopted the `BIO` tagging scheme in generating labels, marking the beginning (`B`), inside (`I`), and outside (`O`) of each segment (e.g., for location or function). Table 1 summarizes details of each building.

Digits. The digits in sensor names indicate detailed and specific information such as room or equipment identifiers, so preserving the variety in numbers does not help our segmentation task. Conversely, it disturbs the transition probability distribution and thus confuses the model in predicting the next characters – the model would only need to learn and recognize the transitions from digit to digit, as opposed to the specific values (e.g., "`1`" to "`2`" or "`4`" to "`3`"). Therefore, we replace all numerical digits with the same digit "0".

Punctuation and Whitespace. There are symbols such as underscores and whitespace in sensor names that are inserted by technicians at the time of metadata construction. We leave them *as-is* for our model to learn their meanings because the meanings of these characters vary from case to case. This is, in fact, one of the major challenges in this

Table 1: Statistics of five buildings in our experiments. These builds are from three different campuses: Buildings **SDH** and **IBM** are from the first campus, **APM** and **EBU** from the second, and **SOD** from the third. Example sensor names are also listed for reference.

Building	#Sensors	#Segments	#Characters	Example Sensor Name
SDH	$2,551$	$2 \sim 5$	$7 \sim 31$	`SDH.AH1_RHC-4:CTL STPT`
IBM	$1,366$	$2 \sim 3$	$6 \sim 28$	`1F_FCU10_11_13_23_COLLAB`
APM	$1,079$	$1 \sim 7$	$4 \sim 34$	`AP&M-CRAC-2-MIG-009.COOLING ON-OFF`
EBU	$1,074$	$2 \sim 3$	$7 \sim 35$	`EBU3B.3RD FLR AVG CLG-PID1`
SOD	$1,335$	$2 \sim 4$	14	`SODC3P09DP_STA`

sensor name segmentation problem. For example, the sensor name "`SODH1______L_L`" should be segmented as "`SOD|H1|______|L_L`", with the three segments corresponding to its building name, equipment id, and measurement type, respectively. The underscores between "`H1`" and "`L_L`" are padded to make the sensor name fixed-length, while the underscore inside "`L_L`" connects two initial letters (i.e., for a Lead-Lag sensor, commonly existing in water pumps).

3.2 Evaluation Metrics

We evaluate the performance of all the considered methods by the F_1, precision, and recall scores. A segment is represented as a span with the starting and the ending character indices. A predicted segment is correct if and only if there exists an exactly same segment in the ground truth. Therefore, we define the precision and recall as follows:

$$\text{prec} = \frac{|\mathcal{S}_{GT}| \cap |\mathcal{S}_{Pred}|}{|\mathcal{S}_{Pred}|}, \text{rec} = \frac{|\mathcal{S}_{GT}| \cap |\mathcal{S}_{Pred}|}{|\mathcal{S}_{GT}|},$$

where $\mathcal{S}_{GT}$ is the set of ground-truth spans and $\mathcal{S}_{Pred}$ is the predicted set. The F_1 score is the harmonic mean of precision and recall. We report the averaged F_1 score of all sensor names, which is relatively unbiased (Opitz and Burst, 2019).

As we mentioned before, there will be some extra delimiters between segments. Therefore, during the evaluation, we ignore segments containing only delimiter(s) in both ground truth and predicted segments. When calculating the start and end indices for predicted segments, we also skip their prefix and suffix delimiters. The same process here applies to the evaluation of all methods.

3.3 Compared Methods

We compare Sensei with the following methods:

- **Delimiter**. There are punctuation (such as "-" and "_") and whitespace characters in sensor names, and they could indicate the boundaries between segments. Therefore, this method segments a sensor name by delimiters (i.e., non-alphanumeric characters). This method mainly serves as a sanity check.
- **NLTK TweetTokenizer**. NLTK (Bird et al., 2009) provides a tweet tokenizer to segment a string into tokens according to predefined regular expressions (regexes). We directly apply it to segment our sensor names.
- **CoreNLP**. We adopt the pre-trained tokenizer in the CoreNLP package[2] (Manning et al., 2014), which adopts the Universal Dependencies[3] version 2 (UD v2) standard for segmentation .
- **Stanza**. We also adopt Stanza[4] and use its built-in neural tokenizer (Qi et al., 2020) following UD v2. This method combines convolutional filters and bidirectional LSTM to realize tokenization and sentence segmentation as a tagging task (Qi et al., 2018).
- **N-gram-LM**. Character-level N-gram language model (N-gram-LM) can also provide character transition probabilities. To obtain an "upper bound" of the N-gram-LM's performance, we apply a grid search technique of Sensei-GS detailed in the following to find the optimal threshold on the test set. As most segments in the ground truth contain fewer than 5 characters, we tried N from 1 to 5 and reported on the best performance. However, this model does not offer any representations that we can use to train the ensemble classifiers for the "break/tie" decisions.
- **BayesSeg**. Topic segmentation divides a document into topic-coherent segments. An unsupervised Bayesian model, BayesSeg[5] (Eisenstein and Barzilay, 2008), is used to segment char-

---

[2] `https://stanfordnlp.github.io/CoreNLP/`

[3] Universal Dependencies is a framework of annotation guidelines by open community effort. `https://universaldependencies.org/`

[4] `https://stanfordnlp.github.io/stanza/`

[5] `https://github.com/jacobeisenstein/bayes-seg`

acters of sensor names as a topic segmentation task that decides the boundary between sentences. However, this method requires to manually specify the number of segments, which is a parameter we do not know without human input.

- **ToPMine**. ToPMine (El-Kishky et al., 2014) provides a method that groups frequent words into phrases in an unsupervised manner and incorporates these phrases into topic modeling. We adapt the model to work at the character level. That is, we regard each character of sensor names as a word in document and group characters into segments as group words into phrases.
- **SeNsER**. Besides the above unsupervised methods, we compare with a transfer learning model, SeNsER (Jiao et al., 2020), a supervised method that uses both source and target buildings' raw sensor names and source building's annotations to learn a sensor name tagger for use in the target building.

Note that, we do not use custom **regexes** to segment sensor names because they require tremendous manual effort to create in order to exhaustively cover all the possible substring patterns, which deviates from our self-supervised problem setting. Moreover, since different buildings follow different sensor naming conventions, manual effort is required from domain experts to create regexes on a per-building basis, which is a costly process.

We also compare with two ablations of our method:

- **Sensei-Forward (Sensei-FW)**. It leaves out the self-supervised ensemble learning. Specifically, we keep the Char-RNN to obtain the distribution of observing next characters, and then find the *single* threshold as stated in Section 2.2.
- **Sensei-Backward (Sensei-BW)**. This is similar to the forward counterpart. The only difference is that the Char-RNN takes as input the *reversed* sensor names. As we shall see in the results, this method does not add much value to our task due to the intrinsic irregularity of sensor names when examined backward.

We further examine a method using grid search based on ground truth for threshold tuning to verify the effectiveness of our threshold decision:

- **Sensei-GridSearch (Sensei-GS)**. Compared to Sensei-FW, this method finds the best threshold for deciding `Tie` using ground-truth labels, i.e., it searches through all the possible threshold values on the transition probability distribution and picks the one producing the best segmentation results. This method is only used to demonstrate that a single threshold chosen based on the transition distribution (as detailed in Section 2.2) gives results reasonably close to the best we can achieve for Sensei-FW using the ground truth.

3.4 Experimental Setup

We modify the Char-RNN library[6] and use Keras (Chollet et al., 2015) to implement our method. As our method is unsupervised, we do not employ the commonly used early-stopping scheme when training the Char-RNN. Instead, we train our models for 100 epochs and empirically find this to be sufficient. All the thresholds have three decimal places. We assign `Ties` as positives and `Breaks` as negatives. For binary classifier, any supervised learning algorithm (e.g., logistic regression, SVM, etc) would accommodate our need in this work. We choose a vanilla Multilayer Perceptron with 2 fully-connected layers, each with 64 cells. We set the number of binary classifiers in our ensemble, K, at 100. The subsampling rate for the ensemble, ϵ, is 10% and for each subsampling, we use pandas with the iteration index as seed. Training a Sensei model on a Colab GPU with 12GB RAM takes less than 40 minutes for each building. For the other compared methods, we tune at our best based on the recommended settings in their papers or repositories and report the best performance.

3.5 Main Results and Analysis

Experimental results for all the unsupervised methods are summarized in Table 2. Overall, Sensei significantly outperforms all the compared methods, attributed to its strategy of complementing the language model with a self-supervised ensemble classifier. Besides the variants of Sensei, the N-gram-LM baseline achieves the second-best performance among all the other unsupervised methods, with an average 62.13% in F_1 across all the buildings. This also illustrates the usefulness of a language model. By contrast, our Sensei achieves over 80% in F_1, which demonstrates a 20-point improvement over N-gram-LM.

When looking at the F_1 scores of baselines including Delimiter, ToPMine, BayesSeg, and the off-the-shelf tokenizers in NLTK, Stanza, and CoreNLP, they are not competitive; this highlights the need for a solution to our challenging problem.

---

[6] `https://github.com/sherjilozair/char-rnn-tensorflow`

Table 2: Performance of Sensei and compared methods on the five test buildings.

	SDH			**IBM**			**APM**			**EBU**			**SOD**			
Methods	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1	Avg F_1
Delimiter	33.21	47.18	38.47	52.61	65.87	57.80	3.10	4.44	3.56	32.00	46.60	37.73	46.54	23.95	31.51	33.81
NLTK	18.34	31.86	22.75	0.07	0.05	0.06	3.95	4.07	3.99	20.76	27.78	23.75	0.04	0.02	0.03	10.12
CoreNLP	17.09	13.46	14.75	0.04	0.02	0.03	39.31	30.88	34.20	15.86	10.58	12.69	0.11	0.06	0.07	12.35
Stanza	9.30	6.95	7.82	0.0	0.0	0.0	2.51	2.75	2.57	9.21	9.09	8.92	0.0	0.0	0.0	3.86
N-gram-LM	60.23	73.73	65.70	77.48	79.53	78.09	39.37	56.61	45.72	51.06	66.62	56.96	57.54	73.71	64.18	62.13
BayesSeg	1.74	2.17	1.92	19.72	28.54	23.25	9.72	10.18	9.88	15.05	25.16	18.82	45.07	34.16	38.84	18.54
ToPMine	16.83	31.42	21.76	27.83	38.86	31.86	14.39	30.63	19.46	2.11	4.55	2.85	15.38	26.17	19.27	19.04
Sensei-BW	10.93	11.00	9.18	0.0	0.0	0.0	0.98	3.86	1.53	1.04	4.66	1.69	19.22	11.33	13.77	5.23
Sensei-FW	61.17	74.45	66.56	39.97	53.40	44.84	38.58	55.58	44.81	47.94	64.65	53.78	58.38	74.18	64.87	54.97
Sensei-GS	61.17	74.45	66.56	79.84	80.43	79.76	38.58	55.58	44.81	47.94	64.65	53.78	58.38	74.18	64.87	61.91
Sensei	**87.00**	**83.64**	**84.95**	**84.81**	**90.80**	**86.84**	**70.23**	**77.98**	**73.21**	**78.10**	**85.77**	**80.39**	**85.81**	**87.53**	**86.43**	**82.36**

Table 3: Transfer learning performance among buildings by SeNsER (Jiao et al., 2020).

	SDH			**IBM**			**APM**			**EBU**			**SOD**		
Source Bldg	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1	Prec	Rec	F_1
SDH	99.87	99.79	99.83	87.11	76.28	81.34	91.20	83.72	87.30	70.10	69.30	69.73	29.75	39.23	33.84
IBM	78.79	85.72	82.11	99.80	99.90	99.85	76.07	59.23	66.60	63.70	66.12	64.89	29.45	39.71	33.82
APM	75.53	78.58	77.03	77.81	58.20	66.59	99.07	98.31	98.69	69.07	67.14	68.09	19.68	36.26	25.52
EBU	74.68	84.99	79.50	84.60	82.20	83.38	88.24	84.99	86.59	99.72	99.44	99.58	3.40	6.72	4.52
SOD	48.44	39.15	43.30	74.56	61.10	67.16	5.85	2.89	3.87	16.76	8.79	11.53	99.75	99.77	99.76

The performance of Delimiter confirms the fact that the semantics of these delimiters are mixed. If one recalls the examples in Table 1, vendors usually use delimiters in sensor names. Sometimes, these delimiters well indicate the segment boundaries. However, as we illustrated in the example sensor name “`SOD|H1|______|L_L`”, punctuation could be also used within the segment, and therefore simply segmenting at delimiters results in a considerable amount of false positives.

From Sensei-FW to Sensei, there is a significant boost, roughly 27 points in F_1 on average. Since the major difference between Sensei and Sensei-FW is our self-supervised ensemble learning module, we empirically verified its power.

Comparing Sensei-FW and Sensei-BW, we observe that the forward version performs dramatically better. Sensei-FW also performs better than Delimiter, ToPMine, and all the pre-trained tokenizers in all cases. By contrast, Sensei-BW takes the reversed sensor names as input but performs much worse than Sensei-FW. We notice that this is because there are not sufficient variations in the sensor string patterns when being looked at backward, compared to the forward case. For example, there are names like “`SODA4R731__ASO`” and “`SODA1R516__VAV`”, and the Sensei-FW model can see various substrings (e.g., “`ASO`” and “`VAV`”) following the common pattern “`SODA0R000__`”. Variations as such provide enough information for the model to learn where to segment. However, when reversed, the above example becomes “`OSA__000R0ADOS`” and the *prefix* “`OSA`” sees no variations following, which makes it nearly impossible for Sensei-BW to figure out the right segmentation. Consequently, Sensei-FW better captures generative patterns while Sensei-BW achieves poor segmentation results.

Comparing Sensei-FW and Sensei-GS, one can observe that, in most cases (4 datasets out of 5), Sensei-FW finds the best single threshold found by Sensei-GS. Note that Sensei-GS utilizes the ground truth to exhaustively search among all the possible thresholds, while Sensei-FW decides the threshold based on the transition distribution without requiring any labels. This small difference in performance indicates that our data-driven threshold finding solution based on the distribution is reasonable and reliable.

From Table 3 we see that the performance of transfer learning varies drastically across buildings. This is because, as the method takes a transfer learning approach, its performance highly hinges on how similar the sensor names in the source building are to those in the target building. For example, **APM** and **EBU** are from the same vendor, and thus

Table 4: Performance of Sensei using different amounts of sensor names for training.

	SDH		**IBM**		**APM**		**EBU**		**SOD**	
Percentage (%)	#Sensors	F_1	#Sensors	F_1	#Sensors	F_1	#Sensors	F_1	#Sensors	F_1
25	637	72.67	341	75.95	269	39.99	268	33.50	333	57.86
50	1,275	92.28	683	71.84	539	48.99	537	47.77	667	85.62
75	1,913	86.38	1,024	85.04	809	57.61	805	70.45	1,001	85.31
100	2,551	84.95	1,366	86.84	1,079	73.21	1,074	80.39	1,335	86.43

SeNsER is relatively more effective on these two buildings. Moreover, since **APM** has more diverse patterns than **EBU** as shown in Table 4, it is reasonable that the transfer from **APM** to **EBU** results in a higher score than the opposite. Additionally, **SOD** are so different from all the other buildings that transfers to **SOD** produce poor results. The noticeable high diagonal scores, whose source and target buildings are the same (i.e., learning within a building), provide an upper-bound reference. We shall also note that, referring to the last row in Table 2, Sensei outperforms SeNsER on four buildings except for **APM**, as Sensei is focused on more effectively capturing patterns inside a building.

3.6 Effect of Number of Sensors

As Sensei framework is fully automated in a self-supervised manner, its performance is solely affected by the amount and variety of sensor names available for learning the segmentation classifier. As shown in Table 4, Sensei generally gets better performance with more sensor names available with an exception of Building **SDH**. We hypothesize that the performance relates more closely to the variety of sensor name patterns in the dataset rather than the number of sensor names.

3.7 Case Studies and Discussions

We next showcase some examples that Sensei correctly segments, in order to illustrate its capability.

"Flukes" for False Positives. In Building **IBM**, some of the `Breaks` are recognized as `Ties` by Sensei-FW and Sensei-GS. For example,

`0F|_|SRVC|_|D0D0D0D00`,
`GF|_|SRVC|_|QR000_000`,

are mistakenly segmented as

`0F_SRVC|_|D0D0D0D00`,
`GF_SRVC|_|QR000_000`.

By contrast, Sensei avoids the mistakes by learning the pattern from many other sensor names. The following case is a great example.

`GF|_|LGHT|_|COFFEEDOCK`.
`GF|_|FRONTAISLE|_|LHS`,
`0F|_|FCU_KWH`.

There are only 89 occurrences of "`_|LGHT|`" compared to 177 of "`_|SRVC|`". Thus, with a lower transition probability, it can be recognized as a `Break` before "`_|LGHT|`". Many similar cases can teach Sensei that `Break` is more likely in this pattern, facilitating its performance.

"Flukes" for False Negatives. Building **SOD** contains many cases as follows:

`SOD|A0|R000|__|ASO`,
`SOD|A0|R000|__|AGN`.

Sensei-FW, and even Sensei-GS which employs the ground truth, are not able to segment these names correctly; they instead segment them as

`SOD|A0|R000|__|A|SO`,
`SOD|A0|R000|__|A|GN`,

because of the same prefix "`SODA0R000__A`".

By contrast, Sensei is able to correctly segment them owing to the self-supervised ensemble learning, which is more robust to noise in pseudo labels.

Discussion. We notice that even though Sensei on average achieves about 80% in F_1, it still has limitations. Sensei is sensitive to the variation of patterns in datasets—the patterns cannot be too varied or too monotonous.

4 Related Work

There are three lines of prior and related work, namely, sensor metadata mapping, language model, and phrase mining.

Sensor Metadata Tagging. Sensor Metadata Tagging refers to the process of parsing and annotating the sensor metadata (or sensor name) for understanding a sensor's key context, including the measurement type (Balaji et al., 2015; Hong et al., 2015b), location (Bhattacharya et al., 2015b), relationships with others (Koh et al., 2018), and many more (Schumann et al., 2014). The majority body of work exploits an active learning-based procedure (Settles, 2009), where it iteratively selects an "informative" and "representative" metadata example for a domain expert to label, in order to learn a model to annotate the metadata. Complementary to the use of textual metadata, there are also efforts exploring the use of time-series data for inferring the sensor context (Koc et al., 2014; Pritoni et al., 2015; Hong et al., 2015a). While they can significantly reduce the required manual labeling, they still rely on the availability of at least one human annotator to segment, parse, and provide labels.

By contrast, the method proposed in this work is fully automated, i.e., completely removing humans from the process, and we demonstrate its use in an essential first step—segmenting a sensor name string into meaningful substrings.

Language Model and Tokenization. Language models originate from the areas of natural language processing and information retrieval (Schütze et al., 2008). They aim at modeling the likelihood of observing one token given all the tokens before it, capturing the underlying language patterns. Recent advances in deep learning have pushed the language modeling from traditional n-gram models to neural language models (Kiros et al., 2014; Karpathy et al., 2015; Kim et al., 2016; Peters et al., 2018; Devlin et al., 2018), achieving significantly better performance using recurrent neural networks.

Analogizing sensor names to human languages, we employ neural language models to capture the underlying naming pattern. As we seek to segment a sensor name string into substrings, we choose the classic Char-RNN model (Karpathy et al., 2015). In general, any character-level language models are applicable in our method.

One can also view our problem as tokenization of sensor names. We thus compare with multiple existing tokenizers provided in NLTK Twitter, Standford CoreNLP (Manning et al., 2014), and Stanza (Qi et al., 2020). As we demonstrate in the evaluation, our method significantly outperforms these methods in segmenting sensor names.

Phrase Mining. Treating characters as words, our problem can be viewed as an unsupervised phrase mining problem with phrasal segmentation as output. Existing methods mainly leverage statistical signals based on term frequency in the corpus (Deane, 2005; Parameswaran et al., 2010; Danilevsky et al., 2014; El-Kishky et al., 2014). Among all these methods, ToPMine (El-Kishky et al., 2014) is arguably the most effective one. Our method Sensei significantly outperforms ToPMine in our empirical evaluation. There exist weakly/distantly supervised phrase mining methods (Liu et al., 2015; Shang et al., 2018a); however, such supervision signals (e.g., Wikipedia) are difficult to obtain for building sensors.

5 Conclusions and Future Work

Smart buildings critically rely upon contextual information associated with its sensors and actuators to appropriately sense and operate building subsystems. Such contextual information appears in the form of sensor metadata that is part of the installation of building management system. In this paper, we have presented Sensei, a system for automated segmentation of metadata information. Sensei is a fully automated method without requiring human labels that employs a character-level neural language model to capture the underlying generative patterns in building sensor names. Based on the probability distribution of character transitions (i.e., likelihood of observing the current character give the previous ones), it decides on two thresholds for sifting out examples for which it is confident to be `Tie` or `Break`. Considering these pseudo-labeled examples as supervision, Sensei constructs an ensemble of binary classifiers to segment sensor names with the information provided by the language model. We conducted experiments on the sensor names from five real-world buildings, and Sensei on average achieves $F_1 > 80\%$ in segmenting sensor names, a roughly 20-point improvement over the best compared unsupervised method. Our ongoing work addresses pre-training methods for the language model to improve Sensei's performance and its use in standard NLP tasks.

Acknowledgement

We thank reviewers for the anonymous comments and suggestions to improve this work. This work was supported in part by National Science Foundation 1940291 and 2040727. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotation hereon.

References

Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. 2015. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *BuildSys*, pages 13–22. ACM.

Arka Bhattacharya, Joern Ploennigs, and David Culler. 2015a. Short paper: Analyzing metadata schemas for buildings: The good, the bad, and the ugly. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 33–34. ACM.

Arka A Bhattacharya, Dezhi Hong, David Culler, Jorge Ortiz, Kamin Whitehouse, and Eugene Wu. 2015b. Automated metadata construction to support portable building applications. In *BuildSys*, pages 3–12. ACM.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

François Chollet et al. 2015. Keras. `https://keras.io`.

Marina Danilevsky, Chi Wang, Nihit Desai, Xiang Ren, Jingyi Guo, and Jiawei Han. 2014. Automatic construction and ranking of topical keyphrases on collections of short documents. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 398–406. SIAM.

Paul Deane. 2005. A nonparametric method for extraction of candidate phrasal terms. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 605–613.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 334–343.

Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R Voss, and Jiawei Han. 2014. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment*, 8(3):305–316.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dezhi Hong, Hongning Wang, Jorge Ortiz, and Kamin Whitehouse. 2015a. The building adapter: Towards quickly applying building analytics at scale. In *BuildSys*.

Dezhi Hong, Hongning Wang, and Kamin Whitehouse. 2015b. Clustering-based active learning on sensor type classification in buildings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 363–372. ACM.

Yang Jiao, Jiacheng Li, Jiaman Wu, Dezhi Hong, Rajesh Gupta, and Jingbo Shang. 2020. SeNsER: Learning cross-building sensor metadata tagger. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 950–960.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *International Conference on Machine Learning*, pages 595–603.

Merthan Koc, Burcu Akinci, and Mario Bergés. 2014. Comparison of linear correlation and a statistical dependency measure for inferring spatial relation of temperature sensors in buildings. In *BuildSys*, pages 152–155. ACM.

Jason Koh, Bharathan Balaji, Dhiman Sengupta, Julian McAuley, Rajesh Gupta, and Yuvraj Agarwal. 2018. Scrabble: transferrable semi-automated semantic metadata normalization using intermediate representation. In *Proceedings of the 5th Conference on Systems for Built Environments*, pages 11–20. ACM.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1729–1744.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Juri Opitz and Sebastian Burst. 2019. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*.

Aditya Parameswaran, Hector Garcia-Molina, and Anand Rajaraman. 2010. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2):566–577.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Marco Pritoni, Arka A Bhattacharya, David Culler, and Mark Modera. 2015. Short paper: A method for discovering functional relationships between air handling units and variable-air-volume boxes from sensor data. In *BuildSys*, pages 133–136. ACM.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Anika Schumann, Joern Ploennigs, and Bernard Gorman. 2014. Towards automating the deployment of energy saving approaches in buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 164–167. ACM.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*, volume 4.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018a. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018b. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064.

Zixiao Shi, Guy R Newsham, Long Chen, and H Burak Gunay. 2019. Evaluation of clustering and time series features for point type inference in smart building retrofit. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 111–120.

Thomas Weng and Yuvraj Agarwal. 2012. From buildings to smart buildings—sensing and actuation to improve energy efficiency. *IEEE Design & Test of Computers*, 29(4):36–44.

Frustratingly simple few-shot slot tagging

Jianqiang Ma[*]
Platform and Content Group, Tencent
`alexanderma@tencent.com`

Zeyu Yan[*]
AI Dept, Souche Inc.
`yanzeyu@souche.com`

Chang Li[*]
AI Dept, Souche Inc.
`lichang@souche.com`

Yang Zhang
AI Dept, Souche Inc.
`zhangyang@souche.com`

Abstract

We propose a simple and effective few-shot model for slot tagging. Recent work shows that it is promising to extend standard few-shot classification methods to sequence labeling with CRF-specific augmentations. Such methods show strengths in encoding slot name semantics and slot dependencies. However, we find these strengths can be obtained by a much simpler method, which casts slot tagging into machine reading comprehension (MRC). We fine-tune a standard BERT-based MRC model with a mixture of source domain and (few-shot) target domain data. Such simple method outperforms state-of-the-art methods by a large margin on the SNIPS dataset.

1 Introduction

This paper considers the task of few-shot slot tagging. Slot tagging (Zhang and Wang, 2016; Haihong et al., 2019) is a core component for task-oriented dialog systems (Papineni et al., 2001), where the goal is to provide a fine-grained, structured description of user request for a given intent. Example (1) shows the input-output of a slot tagging module for the `book restaurant` intent, where the module yields the semantic analysis for the input query in terms of slot label-value pairs such as `restaurant type` is 'brasserie' and `time range` is '15 minutes', etc.

(1) **Query**: I want to book a far brasserie that serves minestrone in PA for a party of 9 in 15 minutes.
Tagged Slots: {`restaurant type`: 'brasserie', `time range`: '15 minutes', `state`: 'PA', `party size number`: '9', `served dish`: 'minestrone'}

In real world systems, slot taggers are required to rapidly cover new domains to address increasing user needs. A key challenge here is that labeled data are often scarce in new domains and the high cost of manually annotating large-scale data becomes a major obstacle for domain adaptation. An attractive alternative is few-shot learning, which aims at achieving reasonable good results using only a few labeled instances in the new domain.

Although there are many successful few-shot *classification* methods, especially meta learning ones (Bapna et al., 2017; Luo et al., 2018; Fritzler et al., 2019), directly adapting them to slot tagging often yields unsatisfactory results (Hou et al., 2020). This is due to the *sentence-level, extractive nature* of slot tagging, where token dependencies within sentences are important but ignored by token-level classification models. Recent work (Hou et al., 2020) tackles this problem by extending meta learning methods to sequence transduction within the BERT-CRF framework, using several CRF-specific augmentations. Such augmentations show strengths in both encoding *slot name semantics* and modeling *slot dependencies*, which are key elements for effective few-shot slot tagging.

This paper shows that we can enjoy similar strengths with a *frustratingly simple* method. Our approach is based on the idea of transforming few-slot tagging into supervised machine reading comprehension (MRC), the detailed formulation of which is described in Section 3, with an example shown in Table 1. The implementation of our method is incredibly simple: we fine-tune an off-the-shelf BERT-based (Devlin et al., 2019) MRC model with data being merged from the source domain and (few shot) target domain data, without any meta learning or extra engineered components.

Our simple method works for good reasons. As the MRC-based approach extracts the full span of each slot value based on the complete sentence, such a model is aligned with the extractive nature of the task and implicitly considers slot dependencies. Moreover, the model can naturally encode label se-

---

[*] Equal contribution. Order decided by tossed coins.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1028–1033
August 1–6, 2021.

mantics by mentioning the label names in the constructed questions. For our MRC model, the slot labels and the sentences being tagged from both source and target domains all reside in the same semantic space, where the training upon mixed data forces the model to generalize to a semantic space that is compatible with both domains. Through experiments, we also find that our model can better leverage linguistic and world knowledge in pre-trained language models, than previous BERT-based few-shot slot taggers.

The contributions of this paper are twofold: (1) We propose a simple and effective approach to few-shot slot tagging, which is based on training a supervised MRC model. (2) We empirically show the effectiveness of the proposed method, which outperforms previous state-of-the-art by 4+ points on the SNIPS benchmark.

2 Related Work

Slot Tagging Intent detection and slot tagging are two key modules in spoken language understanding. Slot tagging is often cast to sequence labeling problem (Zhang and Wang, 2016; Liu and Lane, 2016; Haihong et al., 2019; Qin et al., 2019). This paper adopts MRC formulation and focuses on the few shot learning setup.

Few Shot Learning In NLP, few shot learning methods mostly focus on classification tasks (Sun et al., 2019; Geng et al., 2019), while efforts on sequence labeling like slot tagging are rarely (Luo et al., 2018; Fritzler et al., 2019). Hou et al. (2020) explored few shot slot tagging by considering both label dependency transfer and label name semantics. Our model enjoys similar strengths but is much simpler and more effective.

QA Format for NLP Tasks Question answering, in particular machine reading comprehension (MRC) models (Seo et al., 2017; Xiong et al., 2018), is typically trained to answer questions by extracting a text span from the given context. Recently, there is a trend to cast non-QA NLP tasks, such as information extraction (Levy et al., 2017; Li et al., 2019, 2020), coreference resolution (Hou, 2020; Wu et al., 2020) and more (McCann et al., 2018) into MRC, which can achieve comparable or improved results. Our work is inspired by these works, but tackles a new task of slot tagging with a focus on few-shot learning.

3 Method

3.1 Slot Tagging as MRC

MRC formulation Given a *question* $Q = q_1, q_2, ..., q_L$, and a *context* passage $C = c_1, c_2, ..., c_M$, where $|Q| = L$ and $|C| = M$ are their token numbers respectively, while question Q is used to extract required spans from context C. The task is to find the span between the start token C_{start} and end token C_{end} in the context, for the given question w.r.t. each slot type. Some example questions and their context are shown in Table 1. For all questions associated with the same sentence, we provide one *context* C, which consists of the original sentence, being concatenated by special tokens “NO ANSWER” , which should be extracted, when no answer (*span*) is available in the given sentence for that question (*slot type*).

Following the MRC setup as in BERT (Devlin et al., 2019), we resort to the standard question-answer usage of BERT to find span C_{start}-C_{end}, i.e. feeding the token sequence in the form of `[CLS]`, $q_1, q_2, .., q_L$, `[SEP]`, $c_1, c_2, ..., c_M$ as the input to the BERT model, where the special tokens `[CLS]` and `[SEP]` are inserted between Q and C to distinguish them. Then, the hidden states from the last layer of BERT are extracted as the representations of input tokens. Probabilities $P_{start}(i)$ and $P_{end}(i)$ of each token position being the start and end position of the answer span are computed through the following formulas (1), where $i = 1, 2, ..., M$. For both start and end index predictions, tokens between the highest $P_{start}(i)$ and $P_{end}(j)$ will be predicted as the slot content for the slot type being asked in the question Q.

$$\begin{aligned} H^Q; H^C &= \text{BERT}([Q; C]) \\ H^C &= [v_1; v_2; ...; v_M] \\ P_{start}(i) &= softmax(W_s v_i) \\ P_{end}(i) &= softmax(W_e v_i) \end{aligned} \tag{1}$$

To train the MRC model, we first convert the original dataset, pairs of split sentence tokens and slot types into a set of <*question*, *answer*, *context*> triples, similar to the *format* of SQuAD 1.1 dataset (Rajpurkar et al., 2016). Triples of different samples are shuffled into batches to feed into the MRC model to get predictions of start and end position indices. Every slot type in the corresponding domain will be asked sequentially to find corresponding spans, or ‘NO ANSWER’ will be extracted if that slot type doesn’t appear. We use cross entropy loss between predictions and ground truths.

Question Generation For each slot type $y \in Y$ to be predicted, we use a unified template to generate a *question* Q by joining domain name with slot type, which are both split by upper case letters and underlines, since they have the necessary information and short enough to keep model focusing on context C. As shown in example (1), question about slot type `restaurant_type` in domain `book restaurant` is constructed as *Book Restaurant, restaurant type ?*.

Context: *I want to book a far brasserie that serves minestrone in PA for a party of 9 in 15 minutes NO ANSWER*

Question	**Answer Set**
Book Restaurant restaurant type ?	{brasserie}
Book Restaurant party size description ?	{NO ANSWER}

Table 1: Samples of Context, questions and answers in the MRC formulation for Example (1).

3.2 Few Shot Learning

For a few-shot learning task, we have a *target domain* $D_1 = (x_i, y_i)$ with few labeled data, and n resource-rich source domains $D_2 ... D_n$. The task is to discover the optimal hypothesis h from x to y in domain D_1. To fit our MRC model into N-way k-shot settings, we follow the data construction procedures in Hou et al. (2020), where the support set $S = (x_i, y_i)$ is constructed by ensuring every slot in target domain appears approximately k times and each entity only appears once in a sentence.

We randomly generate 100 above support sets. For each set, we pair it with a query set having 20 excluded samples to form an episode. Together with D domains, our test set is made up of $D \times 100$ episodes. Note that our model is trained in one-go with the data being mixed up from the source domain and the support set of the target domain in each episode, unlike two-phase training in typical few shot learning methods. Despite the difference, the amount/split of the data for training and evaluation is exactly the same as in previous work.For zero-shot learning, we directly evaluate the source domain trained model on the full target domain.

4 Experiment

4.1 Setup

Dataset Our experiment is based on the SNIPS data set (Coucke et al., 2018) which is a benchmark dataset for slot tagging. It has data samples from 7 different domains, namely, Weather (We), Playlist (Pl), Book (Bo), Music (Mu), Restaurant (Re), Screening Event (Se) and Creative Work(Cr). Following few-shot setup in previous work, we split SNIPS data by domain. Each time, we leave one for testing, one for development and the others for training. Such procedure is repeated 7 times for cross validation.

Baselines **Bi-LSTM** (Schuster and Paliwal, 1997) is trained on the support set and tested on the query set using word embeddings of GloVe (Pennington et al., 2014). **Matching Network (MN) with BERT**(Vinyals et al., 2016) builds on top of BERT and labels the sequence in a token-level classification way. For each word, the most similar token in the support set is chosen and its label is assigned accordingly. **WarmProtoZero (WPZ)** (Fritzler et al., 2019) adopts similar strategy as MN, except replacing matching network with the prototypical network (Snell et al., 2017). **SimBERT** also classifies each token with the most similar word in the support set and assigns the corresponding label, where BERT embeddings are used without fine-tuning. **TransferBERT** is a domain transfer model based on vanilla BERT. It is pre-trained on source domain data, followed by fine-tuning on the support set of the target domain. **L-TapNet+CDT** (Hou et al., 2020) is a sequence labeling model based on BERT+CRF, where the Collapsed Dependency Transfer is used for transferring label-to-label dependencies and TapNet is used for transferring label semantics.

Implementation Details The pre-trained BERT-base uncased model is used for our method, where the batch size is set to 16, with max sequence length of 512. We use Adam optimizer (Kingma and Ba, 2014) with initial learning rate of 1×10^{-5} during training. We train the model with 30 epochs for each episode of evaluation, and get results according to develop domain.

4.2 Experimental Results

Main Results for 5-Shot Learning Table 2 shows the results for 5-shot learning. Each column indicates the per-domain results, where that domain

Model	**5-shots Slot Tagging**							
	We	**Mu**	**Pl**	**Bo**	**Se**	**Re**	**Cr**	**AVG**
WPZ	9.54	14.23	18.12	44.65	18.98	12.03	14.05	18.80
WPZ+GloVe	26.61	34.25	22.11	50.55	28.53	34.16	23.69	31.41
Bi-LSTM	25.17	39.80	46.13	74.60	53.47	40.35	25.10	43.52
TransferBERT	59.41	42.00	46.07	20.74	28.20	67.75	58.61	46.11
MN+BERT	36.67	33.67	52.60	69.09	38.42	33.28	72.10	47.98
SimBERT	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
WPZ+BERT	67.82	55.99	46.02	72.17	73.59	60.18	66.89	63.24
L-TapNet+CDT	71.64	67.16	75.88	**84.38**	**82.58**	70.05	**73.41**	75.01
Ours	**89.39**	**75.11**	**77.18**	84.16	73.53	**82.29**	72.51	**79.17**

Table 2: F1 scores results on 5-shot slot tagging. `+CDT` denotes `collapsed dependency transfer`. Scores below mid-line are from our methods, which achieve the best performance. AVG shows the averaged scores. Best results in bold.

is used as the target domain while others are used as source domains. In most domains, our model achieves improved results than all baseline, being based on BERT or not. In particular, our method outperforms the previous SOTA, L-TapNet+CDT, by 4.16% on average F1 score.

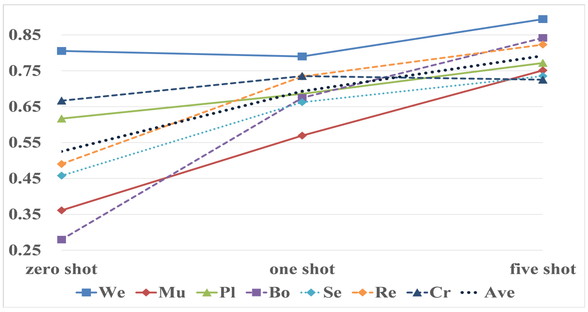

Figure 1: Avg F1 score for each domain for 1-shot and zero-shot learning, which have the same setup for training.

Analysis Figure 1 compares the performance of our model in zero/one/five-shot setup. Since the training of our method is just fine-tuning upon the mixture of source and target domain data, one-shot setup means using 0.01% extra (target domain) data over zero-shot setup. Yet, the boost is dramatic for most domains. We speculate that these tiny bit of target domain has a *catalyst effect*, which changes the optimization trajectory of model. It might be the case that such training forces our MRC model to generalize to a semantic space that is compatible with both source and target domains.

Note that our zero-shot model achieves an average F1-score of 52.5%, outperforming previous zero-shot SOTA, such as 40.6% of Shah et al. (2019) and 37.39% of Liu et al. (2020). As for one-shot setting, the average F1-score of our model is 69.3%, on par with 70.4% of the 1-shot SOTA (Hou, 2020).

Few shot learners typically rely on source domain data to arrive at a good hypothesis. Figure 2 shows the sensitivity of our model w.r.t. the scale of available *source domain* data. In each episode of evaluation, we select a subset (100, 1000, 2000) of sentences from source domains according to the rank of text similarity between them and the support set. While *the more the better* holds in general, we see that 1000 source domain sentences suffice for competitive results.

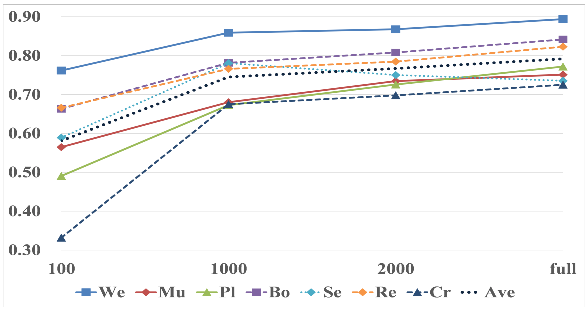

Figure 2: Avg F1 for each domain with different numbers of sentences from source domain. Where the `full` means all sentences from source domains and the number is near 12,000

5 Conclusion

In this paper, we propose a BERT-based MRC approach to few-shot slot tagging. By casting slot tagging into MRC problem, the learning consists of fine-tuning the MRC model with labeled sentences from a mixture of source domain and few-shot target domain data. Such an MRC-based method can naturally encode the label semantics in the form of questions, while the training forces the model to generalize to a semantic space that is compatible with both domains. Experiment results show the effectiveness of our simple method, as it outperforms previous SOTA on the SNIPS benchmark by a large margin. For future work, we plan to extend our approach to similar tasks, such as semantic role labeling and named entity recognition.

References

Ankur Bapna, G. Tür, Dilek Z. Hakkani-Tür, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *ArXiv*, abs/1707.02363.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibault Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 993–1000, New York, NY, USA. Association for Computing Machinery.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3904–3913, Hong Kong, China. Association for Computational Linguistics.

E. Haihong, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *ACL*.

Yufang Hou. 2020. Bridging anaphora resolution as question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1428–1438, Online. Association for Computational Linguistics.

Yutai Hou, W. Che, Y. Lai, Zhihan Zhou, Yijia Liu, H. Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. pages 333–342.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of ACL*, pages 1340–1350, Florence, Italy. Association for Computational Linguistics.

Bing Liu and I. Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *ArXiv*, abs/1609.01454.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 19–25, Online. Association for Computational Linguistics.

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying up regular expressions with neural networks: A case study for spoken language understanding. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2083–2093, Melbourne, Australia. Association for Computational Linguistics.

B. McCann, N. Keskar, Caiming Xiong, and R. Socher. 2018. The natural language decathlon: Multitask learning as question answering. *ArXiv*, abs/1806.08730.

Kishore A Papineni, Salim Roukos, and Robert T Ward. 2001. Natural language task-oriented dialog manager and method. US Patent 6,246,981.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Mike Schuster and K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45:2673–2681.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603.

Darsh Shah, Raghav Gupta, Amir Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc.

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 476–485, Hong Kong, China. Association for Computational Linguistics.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. Dcn+: Mixed objective and deep residual coattention for question answering. *ArXiv*, abs/1711.00106.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2993–2999. AAAI Press.

Medical Code Assignment with Gated Convolution and Note-Code Interaction

Shaoxiong Ji †, **Shirui Pan** ‡, **Pekka Marttinen** †
† Department of Computer Science, Aalto University
‡ Faculty of Information Technology, Monash University
{shaoxiong.ji; pekka.marttinen}@aalto.fi
shirui.pan@monash.edu

Abstract

Medical code assignment from clinical text is a fundamental task in clinical information system management. As medical notes are typically lengthy and the medical coding system's code space is large, this task is a long-standing challenge. Recent work applies deep neural network models to encode the medical notes and assign medical codes to clinical documents. However, these methods are still ineffective as they do not fully encode and capture the lengthy and rich semantic information of medical notes nor explicitly exploit the interactions between the notes and codes. We propose a novel method, gated convolutional neural networks, and a note-code interaction (GatedCNN-NCI), for automatic medical code assignment to overcome these challenges. Our methods capture the rich semantic information of the lengthy clinical text for better representation by utilizing embedding injection and gated information propagation in the medical note encoding module. With a novel note-code interaction design and a graph message passing mechanism, we explicitly capture the underlying dependency between notes and codes, enabling effective code prediction. A weight sharing scheme is further designed to decrease the number of trainable parameters. Empirical experiments on real-world clinical datasets show that our proposed model outperforms state-of-the-art models in most cases, and our model size is on par with light-weighted baselines.

1 Introduction

Automatic medical code assignment is a routine healthcare task for medical information management and clinical decision support. The International Classification of Diseases (ICD) coding system, maintained by the World Health Organization (WHO), is widely used among various coding systems. Thus, the medical code assignment task is also called ICD coding. It uses all types of clinical notes to predict medical codes in a supervised manner with human-annotated codes (Perotte et al., 2014), which is formulated as a multi-class multi-label text classification problem in the medical domain.

While there are increasing works in the community in automatic medical code assignment (Prakash et al., 2017; Shi et al., 2017; Mullenbach et al., 2018; Ji et al., 2020), this task remains challenging from the perspectives of note representation and code prediction. First, medical note representation, a critical step in understanding medical notes, is formidably challenging due to the lengthy and complex semantic information in the discharge documents. There are typically thousands of tokens in a medical note due to the various diagnoses and procedures experienced by a patient. Furthermore, clinical notes also contain a vocabulary with many professional words and phrases, making it hard for a neural network model to encode and understand critical information. Second, the medical coding system has a very high and sparse dimensional label space, which renders the code prediction task incredibly difficult. For example, ICD9 and ICD10 coding systems have many labels, i.e., more than 14,000 and 68,000 codes. However, a patient typically is diagnosed with only a couple of codes over the whole coding space.

Early works for medical code assignment typically follow statistical approaches. They either employ rule-based methods (Farkas and Szarvas, 2008) or apply classification methods such as SVM and Bayesian ridge regression (Lita et al., 2008) to assign the codes. These methods are shallow and do not exploit the complex semantic information in medical notes, leading to unsatisfactory performance. Recently, Natural language processing (NLP) techniques based on deep learning have been developed (Mullenbach et al., 2018; Li et al.,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1034–1043
August 1–6, 2021.

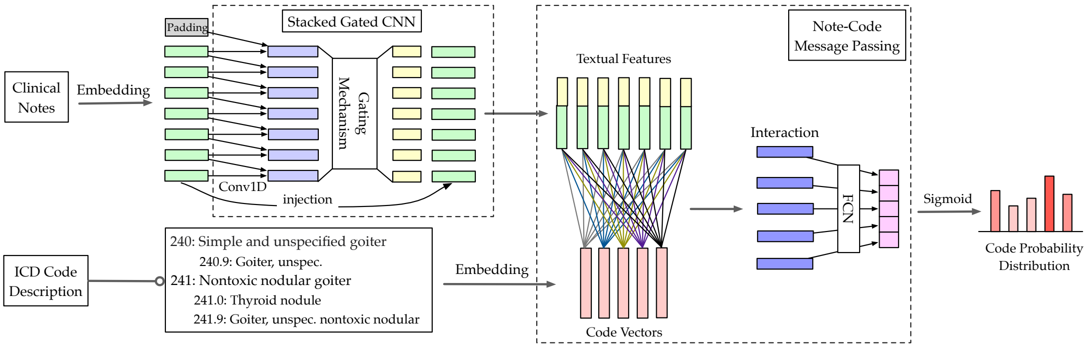

Figure 1: Illustration of the GatedCNN-NCI model architecture. The gating mechanism controls the information propagation. Textual features interact with each code vector in the note-code interaction module. FCN is a fully connected layer.

2018; Cao et al., 2020; Ji et al., 2020), which learn the note representation via convolutional neural networks. Specifically, CAML (Mullenbach et al., 2018), MultiResCNN (Li et al., 2018) and DCAN (Ji et al., 2020) treat ICD coding as a general text classification problem and develop complex neural encoders to learn the note representation. HyperCore (Cao et al., 2020) proposes the hyperbolic embedding to capture code hierarchy and co-occurrence. However, these approaches are still ineffective, as they do not explicitly capture the fine-grained interactions between textual elements and medical codes. These interactions naturally represent the interdependencies between the complex medical words and associated codes, and thus should be well exploited.

This paper puts forward a novel neural architecture, Gated Convolutional Neural Network with Note-Code Interaction (GatedCNN-NCI), for effective medical code assignment. Our goal is to learn rich representation from clinical notes and exploit the interactions between medical texts and clinical codes. To capture the long sequential history of clinical documents, we design a novel dilation information propagation component with a forgetting mechanism to selectively utilize the useful information for note representation learning. To tackle the large labeling space, we formulate textual notes and medical codes as a complete bipartite graph and develop a graph message passing approach to capture the explicit interaction between notes and codes. The ICD code descriptions are used as an external medical knowledge source to learn more accurate code representations that preserve the semantic relations of the codes. Considering the practical application in real-world medical institutes, especially those with limited computing resources, our architecture also prioritizes computational efficiency when designing the sub-modules. Our contributions are itemized as follows.

- We propose a CNN-based neural architecture with dilation and gating mechanism for clinical text encoding. We enhance the feature representation learning with 1) embedding injection, enhancing the deeper features of lengthy clinical notes; 2) and the gating mechanism to control the information propagation.
- We view the note-code interaction as a complete bipartite graph and propose a graph message passing mechanism to capture the interactions between textual features and ICD codes explicitly.
- To reduce the trainable parameters and make our model computationally efficient, we develop a weight-sharing mechanism across the length of the sequence and the depth of the network.
- Experiments in real-world clinical datasets empirically validate our model's effectiveness by comparison with the state of the art.

2 Related Work

Classical medical coding systems used rule-based methods (Farkas and Szarvas, 2008), studied feature selection (Medori and Fairon, 2010), and applied classification models such as SVM and Bayesian ridge regression (Lita et al., 2008). Perotte et al. (2014) utilized the hierarchical structure of the ICD code systems and provided a flat and hierarchical SVM for diagnosis code classification, while Kavuluru et al. (2015) studied explicit co-occurrence relations between codes. Scheurwegs et al. (2016) investigated heterogeneous data of both structured records and textual data. Recent deep learning-based models use word embedding techniques and develop complex neural network architectures to learn rich text features for automatic medical code assignment. Popular models use recurrent architectures such as the LSTM network with an attention mechanism (Shi et al., 2017) and GRU network with hierarchical attention (Baumel et al., 2018). Prakash et al. (2017) used Wikipedia as a knowledge source and proposed condensed memory networks (C-MemNNs) with iterative condensation of memory representation. Although CNNs are traditionally applied in computer vision, many ICD coding methods utilize convolutional architectures. CAML (Mullenbach et al., 2018) used CNN with multiple filters and label attention. Li et al. (2018) adopted the doc2vec embedding and CNN architecture, and Bai and Vucetic (2019) incorporated online knowledge sources. The recent MultiResCNN model (Li and Yu, 2020) extensively concatenated and stacked CNNs with multi-filter convolution and residual learning. HyperCore (Cao et al., 2020) utilized hyperbolic embedding and cograph representation to capture the code hierarchy.

3 Method

3.1 High-level Model Architecture

3.2 Problem Definition

The input clinical note with n words is denoted as $\mathbf{x}_{1:n} = x_1, \ldots, x_n$, where each x_i is a word (or token). The medical coding system is the set of all possible diagnosis and procedure codes denoted as $\mathcal{C}$. The medical code assignment learns a function $\mathcal{F} : \mathcal{X}^n \rightarrow \mathcal{Y}^m$ such that

$$y = \mathcal{F}\left(x_1, \ldots, x_n; \mathcal{D}\right), \qquad (1)$$

where $y \in \mathbb{R}^m$ is the medical code at discharge, m is the number of medical codes, and $\mathcal{D}$ is an optional external knowledge source. This paper uses the ICD coding system and naturally utilizes the official textual ICD code description as an external knowledge source.

The high-level model architecture of GatedCNN-NCI is illustrated in Fig. 1. Our model consists of two main components, i.e., stacked gated CNN layers for clinical note encoding and note-code interaction to fuse the external ICD code description. The stacked gated CNNs include three sub-modules, i.e., dilated convolution, embedding injection, and gating mechanism.

We use word2vec (Mikolov et al., 2013) to train word embeddings from raw tokens. Word embedding matrix of a clinical note is denoted as $[\mathbf{w}_1, \ldots, \mathbf{w}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times d_e}$, where d_e is the dimension of word vectors. Then we input the word embeddings into stacked gated CNN layers for long-range information propagation. The stacked module uses dilated convolution as its backbone (Oord et al., 2016). To further enhance the feature learning, we inject the original embedding into each stacked layer. The gating mechanism is originated from the long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997). We adopt the LSTM-like gate (Dauphin et al., 2017) to control the information flow.

To avoid blurry memory in higher layers, we inject the original word embeddings (Bai et al., 2019). Label interaction has been studied by Wang and Jiang (2016) and Du et al. (2019). We utilize descriptive knowledge from the ICD code descriptions and develop the note-code interaction to capture the relational match between clinical note features and ICD codes. To reduce the training cost and stabilize the training process, we also introduce a weight sharing mechanism across the stacked CNNs (Bai et al., 2019).

3.3 Dilated Convolutional Layers

We use the one-dimensional convolution with dilation as the backbone of our encoder, which takes the word embedding $\mathbf{X} \in \mathbb{R}^{n \times d_e}$ as input. Dilated CNN has exhibited a significant capacity for long sequence modeling and computationally efficient for parallelism (Bai et al., 2018). Specifically, we use a 1D convolution operator $\mathrm{Conv1D}(x; f)$, with a filter $f : \{0, \ldots, k-1\} \rightarrow \mathbb{R}$, to each dimension of the word vectors. Given a sequence of one-dimensional elements $\mathbf{x} \in \mathbb{R}^n$, the one-dimensional dilated convolution $\mathcal{F}_d$ is denoted as

$$\mathcal{F}_d(s) = \left(\mathbf{x} *_d f\right)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}, \qquad (2)$$

where d is the dilation size (i.e., the space between kernel elements), s is the index of the element of the input sequence, k is the convolving kernel (aka, the filter) size, and $s - d \cdot i$ refers to past time steps. The dilation size of d and kernel size k control the receptive field. The 1D dilated convolution has d_h output channels, i.e., for each of the d_e input channels d_h convolutional features are learned through the dilated $\mathrm{Conv1D}$. Stacking CNN layers can be adopted to learn in-depth features.

3.4 Embedding Injection

Our hypothesis for encoding a very long clinical sequence is that the deep neural encoding architecture tends to forget important information, mainly because the clinical note contains fruitful professional expression about the patient's diagnosis. Thus, in-depth features become blurry with the increase of neural layers. We propose to inject original word embedding into each intermediate layer of the proposed architecture, attempting to remind the network to reactivate the original diagnostic notes and mitigate the failure of extracting meaningful, in-depth features. We denote the hidden representation at the l-th layer as $\mathbf{H}^l \in \mathbb{R}^{n \times d_h}$, where the dimension d_h is the hidden dimension. Word embedding is concatenated into lth-layer hidden representation as

$$\mathbf{J}^l = \text{concat}[\mathbf{X}, \mathbf{H}^l], \tag{3}$$

where $\mathbf{J}^l \in \mathbb{R}^{n \times (d_e + d_h)}$ are the deep features enhanced with the original clues, used as the new input of the next convolutional encoding layer. We randomly initialize the $\mathbf{H}^0$ matrix for the first convolutional layer.

3.5 Gating Mechanism

Embedding injection of original word vectors brings low-level features to higher-level, which may lead to difficulty in feature learning in higher layers. Thus, we develop an LSTM-style gating mechanism to control the information flow and capture a long history in the sequence. Unlike the recurrent gate such as the LSTM that controls the information flow along the time coordinate, this gating mechanism controls the flow through stacked layers' depth. The gating mechanism is depicted in Fig. 2, where σ and $\tanh$ are sigmoid and hyperbolic tangent activation functions respectively. After the embedding injection, the dilated CNN upsamples the injected signal $\mathbf{J}^l$ into $\mathbf{U}^l \in \mathbb{R}^{n \times d_u}$ at the l-th layer. We divide $\mathbf{U}^l$ into four matrices with the same dimension, i.e., $\mathbf{I}$, $\mathbf{O}$, $\mathbf{G}$ and $\mathbf{F} \in \mathbb{R}^{n \times d_g}$, such that:

$$\mathbf{U}^l = \text{concat}[\mathbf{I}, \mathbf{O}, \mathbf{G}, \mathbf{F}]. \tag{4}$$

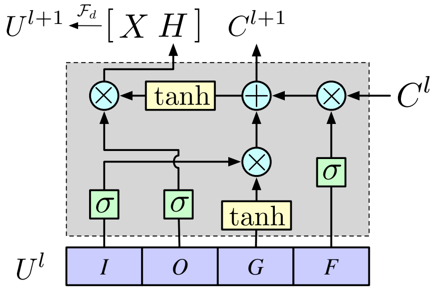

Figure 2: Gating mechanism that controls the flow's convolutional features through stacked layers' depth.

Here, we have $d_u = 4 \times d_g$. Then, these four matrices are fed into the LSTM-like gating module that controls what information should be propagated to deeper layers. The input gate $\sigma(\mathbf{I})$ decides the information to be infused and stored into the cell state $\mathbf{C}$. The forget gate $\sigma(\mathbf{F})$ chooses the information to be remembered. The output gate $\sigma(\mathbf{O})$, working with the cell state, focuses on what signals propagate into the next layer. This process is formalized as

$$\begin{aligned} \mathbf{C}^{l+1} &= \sigma(\mathbf{F}) * \sigma(\mathbf{C}^l) + \sigma(\mathbf{I}) * tanh(\mathbf{G}) \\ \mathbf{H} &= \sigma(\mathbf{O}) * tanh(\mathbf{F}), \end{aligned}$$

where $\mathbf{C}^l$ is the cell state at the l-th layer and $\mathbf{H}$ is the hidden state produced by the gated unit. The embedding injection trick concatenates the original word embedding $\mathbf{X}$ and the hidden representation $\mathbf{H}$, and the dilated convolutional layer upsamples the concatenation to get the new feature $\mathbf{U}^{l+1}$ at the $(l+1)$-th layer, denoted as:

$$\mathbf{U}^{l+1} \xleftarrow{\mathcal{F}_d} [\mathbf{X}, \mathbf{H}]. \tag{5}$$

Gated CNNs can be stacked into a deep architecture, as shown in the general framework of Fig. 1. As a result, our model can represent a large-sized context and extract hierarchical features at each layer. Moreover, the gating mechanism can also extract important features to remember and focus, while less critical features are forgotten and ignored at each layer.

3.6 Note-Code Interaction as Message Passing

To capture the explicit note-code interaction (NCI) between the medical codes and textual mentions, we build a complete bipartite graph $G = \{U, V, E\}$, where $U = \{w_i\}^n$ and $V = \{c_j\}^m$ represent the words and ICD codes respectively, and E is the fully connected edge set. For simplicity, we omit the superscript of the last convolutional features $\mathbf{U}^{l+1}$ extracted by the stacked gated CNNs and denote the textual node features $\mathbf{U}$ as the vertex set U in the note-code bipartite graph. We incorporate the ICD code descriptions of WHO to represent the medical knowledge about ICD codes. For example, the ICD code 240 in Fig. 1 is about simple and unspecified goiter. Instead of merely using the ICD code index to represent the prediction target, we include the code description, which contains rich domain knowledge. Word embeddings of description are averaged to obtain code vectors $\mathbf{V} \in \mathbb{R}^{m \times d_v}$, where m is the number of codes, and d_v is the embedding dimension. We take the code vectors as the node features of the vertex set V.

Our novel formulation of the bipartite graph preserves the source-target matching between textual features and ICD code vectors. We utilize the graph message passing mechanism (Gilmer et al., 2017; Wu et al., 2020) to infer fine-grained clues about dependencies between textual features and code semantics. The composition function $\mathrm{NCI} : \mathbb{R}^{n \times d_u} \times \mathbb{R}^{m \times d_v} \to \mathbb{R}^m$ is denoted as:

$$\mathrm{NCI}(U, V) = f_\theta\left(\sum_{i,j} g_\xi(w_i, c_j)\right), \tag{6}$$

where g_ξ with parameter ξ is a neural message function and f_θ with parameter θ is an output function. It takes the textual features of all tokens in a note and embeddings of code vectors as inputs and produces an interaction score between the note and each code. To improve the computational efficiency, we take the dot product as the message function g_ξ. The explicit interaction score between token w_i and code c_j is calculated as

$$\mathbf{I}_{ij} = \mathbf{V}_{i,:} \mathbf{U}_{j,:}^{\mathrm{T}}, \tag{7}$$

where $\mathbf{V}_{i,:}$ is the row vector of textual features representing the i-th word, $\mathbf{U}_{j,:}$ is the row vector of ICD code matrix representing the j-th code in ICD code set. We set $d_u = d_v$ and get the interaction matrix $\mathbf{I} \in \mathbb{R}^{m \times n}$ with dot product. We use a fully connected network f_θ to calculate the scores of the note-code interactions as output. Similar to the matrix factorization formulation of language models (Yang et al., 2017; Li et al., 2020), this dot-product interaction between notes and codes approximates the point-wise mutual information of note-code co-occurrence.

3.7 Parameter-efficient Weight Sharing

The embedding injection and convolutional feature concatenation make the hidden feature high-dimensional. Moreover, as a result of stacking deep layers, the overall model will become cumbersome. Thus, we utilize a weight sharing mechanism (Bai et al., 2019) to decrease the number of parameters. Specifically, we share the weights of gated CNN layers across time steps and depth through neural layers. This mechanism has two benefits. First, it can decrease the number of trainable parameters because weights across the network are tied. Second, it provides a form of regularization to stabilize the training process.

3.8 Objective and Training

We formulate the ICD code assignment as a multi-label multi-class classification problem. We adopt the binary cross entropy loss denoted as:

$$\mathcal{L} = \sum_{i=1}^{m} \left[-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)\right], \tag{8}$$

where $y_i \in \{0, 1\}$ is the ground-truth label, $\hat{y}_i$ is the sigmoid score for prediction, and m is the number of ICD codes. We use Adam optimizer (Kingma and Ba, 2014) to train the model with backpropagation.

4 Experiments

In the experimental analysis on real-world datasets, we compare our proposed model with several recent strong baselines. Our code is available[1].

4.1 Datasets

This paper focuses on textual discharge summaries from a hospital stay. Specifically, we use raw notes, ICD diagnoses, and procedures for patients from two public clinical datasets, i.e., MIMIC-II and MIMIC-III[2], for experiments. Discharge summaries labeled with a set of ICD-9 diagnosis and procedure codes include descriptions of procedures performed by the physician, diagnosis notes, patient's medical history, and discharge instructions.

MIMIC-II. The first dataset of clinical notes is from the Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II) database (Saeed et al., 2011). We follow the standard train-test split performed by Perotte et al. (Perotte et al., 2014), where 90% and 10% of 22,815 non-empty discharge summaries are used for training and testing, respectively.

MIMIC-III. The second dataset is an updated database from Medical Information Mart for In-

---

[1] `https://agit.ai/jsx/GatedCNN-NCI`

[2] `https://mimic.physionet.org`

tensive Care III (MIMIC-III) repository (Johnson et al., 2016), containing patient admitted to Intensive Care Unit (ICU) at a US medical center during 2001 to 2012. We use the "noteevents" table in the latest version 1.4, with 58,576 hospital admissions. Free-text discharge summaries in the MIMIC-III database are extracted to form the dataset with clinical text. The experimental evaluation considers two settings. The first one uses the full set of ICD codes. Following Shi et al. (Shi et al., 2017) and Mullenbach et al. (Mullenbach et al., 2018), an additional experiment on the subset of MIMIC-III with the top 50 frequent labels is conducted. This MIMIC-III top-50 subset has a train/dev/test split with 8,066, 1,573, and 1,729 samples.

4.2 Settings

Preprocessing We preprocess the textual documents following the preprocessing procedures developed by Mullenbach et al. (2018) and Li and Yu (2020). The NLTK package[3] is utilized for tokenization, and all tokens are converted into lowercase. All words appearing in less than three training documents were replaced with "unk". We truncate all documents at the length of 2500 tokens. The word embeddings are initialized with embedding vectors pre-trained on all discharge notes with the continuous-bag-of-words (CBOW) method of word2vec (Mikolov et al., 2013).

Hyper-parameters Some standard settings follow the prior works. For example, the word embedding dimension is 100, and the dropout rate is 0.2. Adam optimizer (Kingma and Ba, 2014) is used to optimize our model parameters. For the rest hyper-parameters, the random search is utilized to search the optimal settings. The searching range or choices of specific hyper-parameters are listed in Table 1. The searching interval of learning rate is $[1e^{-6}, 1e^{-2}]$. Besides, we optimize for kernel size, levels of residual connections, and hidden representation dimension.

Table 1: Range and choices of hyper-parameter search

Hyper-parameters	Range/choices
Learning rate	$[1e^{-6}, 1e^{-2}]$
Kernel size	2, 3, 5, 9
CNN levels	1, 2, 3, 4, 5
Hidden dimension	100, 200, 300, 400, 500, 600

Evaluation Metrics We use area under the receiver operating characteristic curve (AUC-ROC), F1-score, and precision at k (P@k) for evaluation. We set $k = 5$ for MIMIC-III subset with top-50 frequent codes and $k = 8$ for full sets of MIMIC-II and MIMIC-III. In the multi-label classification setting, we use two averaging strategies, i.e., micro and macro. The macro scores are obtained by averaging the respective label-wise scores across all labels. Micro scores give more weight to frequent labels by considering all labels jointly. We run the experiments for 5 times and report the mean ± standard deviation.

4.3 Baselines

We consider the following baseline models. MultiResCNN (Li and Yu, 2020) and HyperCore (Cao et al., 2020) are two recent strong models with the state-of-the-art performance. **Bi-GRU (Mullenbach et al., 2018)** uses a simplified gated recurrent unit with bi-direction, where last hidden representations are used for classification. **C-MemNN (Prakash et al., 2017)** introduces an iterative condensation of memory representations and utilizes external knowledge source from Wikipedia to enhance memory networks by preserving the hierarchical structure in the memory. **AttentiveLSTM (Shi et al., 2017)** encodes clinical descriptions and ICD long titles jointly with character- and word-level LSTM networks and uses attention mechanism for matching important diagnosis snippets. **CAML (Mullenbach et al., 2018)** integrates CNNs and a label-wise attention mechanism to learn rich representations. It has a variant called DR-CAML that uses ICD code descriptions to regularized the loss function. **LEAM (Wang et al., 2018)** encodes two channels of inputs and leverages the compatibility between word and label embeddings to calculate attention scores. **MultiResCNN (Li and Yu, 2020)** combines residual learning (He et al., 2016) and multiple channels concatenation with different convolutional filters, achieving good performance in most settings. **HyperCore (Cao et al., 2020)** utilizes hyperbolic embedding and co-graph representation with code hierarchy. It gains slightly better performance than the MultiResCNN.

4.4 Results

Our model performs consistently the best for frequent labels. First, it beats all models in the MIMIC-III subset with top-50 codes (columns 2-6 in Table 2). For the micro scores that give more

---

[3] `http://www.nltk.org`

Table 2: Results on MIMIC-III with top-50 and full codes. “-” indicates no results reported in the original paper. **Bold** text denotes the best and *italic* text denotes the second best.

Model	MIMIC-III Top-50 Codes					MIMIC-III Full Codes				
	AUC-ROC		F1		P@5	AUC-ROC		F1		P@8
	Macro	Micro	Macro	Micro		Macro	Micro	Macro	Micro	
Bi-GRU (Mullenbach et al., 2018)	82.8	86.8	48.4	54.9	59.1	82.2	97.1	3.8	41.7	58.5
C-MemNN (Prakash et al., 2017)	83.3	-	-	-	42.0	-	-	-	-	-
CNN (Kim, 2014)	87.6	90.7	57.6	62.5	62.0	80.6	96.9	4.2	41.9	58.1
Attentive LSTM (Shi et al., 2017)	-	90.0	-	53.2	-	-	-	-	-	-
DR-CAML (Mullenbach et al., 2018)	88.4	91.6	57.6	63.3	61.8	89.7	98.5	8.6	52.9	69.0
LEAM (Wang et al., 2018)	88.1	91.2	54.0	61.9	61.2	-	-	-	-	-
MultiResCNN (Li and Yu, 2020)	89.9±0.4	92.8±0.2	60.6±1.1	67.0±0.3	64.1±0.1	91.0±0.2	98.6±0.1	8.5±0.7	*55.2*±0.5	*73.4*±0.2
HyperCore (Cao et al., 2020)	89.5±0.3	92.9±0.2	60.9±0.1	66.3±0.1	63.2±0.2	**93.0**±0.1	**98.9**±0.5	*9.0*±0.3	55.1±0.1	72.2±0.2
GatedCNN-NCI (ours)	**91.5**±0.3	**93.8**±0.1	**62.9**±0.5	**68.6**±0.1	**65.3**±0.1	*92.2*±0.2	**98.9**±0.3	**9.2**±0.2	**56.3**±0.1	**73.6**±0.3

weight to frequent labels, our model also has the best predictive metrics (columns 8&10 in Table 2 and columns 3&5 in Table 3). Moreover, our model is competitive also with the rest of the metrics: it consistently has the best P@k scores and at worst, the second best macro scores in all datasets.

MIMIC-III (Top-50 Codes) The first experiment uses the MIMIC-III subset with top-50 codes, showing models' performance on predicting the frequent diagnosis. The results in Table 2 show that our model outperforms all the baselines in all the evaluation metrics. Significantly, our model gains a higher macro F1-score by 2% and micro F1-score by 1.6% than the state of the art.

MIMIC-III (Full Codes) We then run our model on the MIMIC-III dataset with full codes. Our model outperforms most baselines, gaining the best scores in macro AUC-ROC, macro F1, micro F1, and precision@8. For the macro AUC-ROC, our model is ranked at the second place.

MIMIC-II (Full Codes) In the third dataset of MIMIC-II, we also predict the full set of ICD-9 codes. Our model achieves predictive performance on par with two recent strong baselines of MultiResCNN and HyperCore. We gain the best scores in micro AUC-ROC, micro F1-score, and P@8. Macro AUC-ROC and F1 scores of our model are the second best of the models compared.

Table 3: Results on MIMIC-II full codes. **Bold** text denotes the best and *italic* text denotes the second best.

Model	AUC-ROC		F1		P@8
	Macro	Micro	Macro	Micro	
CNN	74.2	94.1	3.0	33.2	38.8
Bi-GRU	78.0	95.4	2.4	35.9	42.0
DR-CAML	82.6	96.6	4.9	45.7	51.5
MultiResCNN	85.0±0.2	96.8±0.1	5.2±0.2	46.4±0.2	*54.4*±0.7
HyperCore	**88.5**±0.1	*97.1*±0.4	**7.0**±0.2	*47.0*±0.3	53.7±0.3
GatedCNN-NCI	*87.2*±0.3	**97.2**±0.1	*6.4*±0.3	**47.3**±0.2	**54.5**±0.4

4.5 Comparison with BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) has revolutionized the NLP community recently. The pre-trained language model has been applied to different downstream NLP tasks. We compare our model's performance with the BERT model and a domain-specific variant, i.e., ClinicalBERT (Alsentzer et al., 2019) pre-trained on the clinical text of MIMIC-III. For the BERT model, we use the uncased BERT-base with a hidden dimension of 768. Because these two BERT models require the configuration of the maximum sequence length of 512, we truncate the text sequence for our model to ensure a fair comparison. BERT models have two special tokens, i.e., `[CLS]` and `[SEP]`. Thus, we truncate clinical notes with a length of 510. We use Huggingface's transformer framework[4] when implementing these two models. The results in Table 4 show that pretraining the language model with domain data improves the performance, and our model has better performance in most evaluation metrics.

Table 4: Comparison with BERT and ClinicalBERT using the MIMIC-III top-50 code dataset with sequence length truncated at 510.

Model	AUC-ROC		F1		P@5
	Macro	Micro	Macro	Micro	
BERT-base	80.6	85.2	43.3	53.2	53.3
ClinicalBERT	81.0	85.6	**43.9**	54.3	54.5
GatedCNN-NCI	**83.7**	**87.7**	42.9	**54.4**	**56.6**

4.6 Model Size

We compare the number of trainable parameters (Table 5) of our model with two models with quali-

---

[4] `https://github.com/huggingface/transformers`

fied performance, i.e., CAML (Mullenbach et al., 2018) and MultiResCNN (Li and Yu, 2020). HyperCore (Cao et al., 2020) didn't publish the code or provide the values of all hyperparameters. Thus, we omit it in this comparison. Our proposed model is more efficient than the MultiResCNN in terms of the number of trainable parameters. The CAML model has the fewest parameters but performs poorly in prediction. Our model has a much better predictive performance than the CAML model, with only a slight increase in model size.

Table 5: Number of trainable parameters

Model	num. params.
CAML (Mullenbach et al., 2018)	6.2M
MultiResCNN (Li and Yu, 2020)	11.9M
ClinicalBERT (Alsentzer et al., 2019)	113.8M
GatedCNN-NCI (Ours)	7.6M

4.7 Ablation Study

We further conduct an ablation study the investigate the effectiveness of different components of our proposed model. We evaluate two variants by removing two critical components of the proposed model. The first variant without NCI replaces the note-code interaction with max-pooling and linear projection. The second variant removes the gating mechanism that controls the information progation over the CNN layers. Table 6 compares the experimental results on the MIMIC-III subset with top-50 codes. The performance drops to some extent after removing these two modules, which shows the effectiveness of our proposed architectures. Moreover, the note-code interaction module has slightly more contribution than the gating mechanism. Possible explanations are that the explicit interaction perseveres the semantics of medical codes well and captures the relation between codes and notes in the embedding space.

Table 6: Ablation study

Model	AUC-ROC		F1		
	Macro	Micro	Macro	Micro	P@5
GatedCNN-NCI	91.5	93.8	62.9	68.6	65.3
without NCI	90.1	92.7	61.4	67.2	63.9
without gating	90.0	92.0	60.2	66.9	63.7

4.8 Case Study

We conduct a case study to interpret an example prediction. Table 7 shows the predictions for a clinical note of a patient with cardiovascular diseases and diabetes. The patient also had 'dyspnea on exertion' as a symptom caused by either pneumonia or cardiac diseases. Our model and MultiResCNN predict the correct diagnosis codes: coronary atherosclerosis (ICD code: 414.01), hypertension (401.9), and diabetes (250.00). When predicting procedure codes, MultiResCNN is confused by dyspnea on exertion and incorrectly predicts pneumonia-related treatments: endotracheal intubation (96.04) and invasive mechanical ventilation (96.71). Our model correctly predicts a cardiac catheterization procedure and diagnostic interventions of heart surgery (39.61) and coronary artery bypass (36.15). Hence, our model is not misled by the ambiguous interpretation for dyspnea on exertion but learns the correct cardiac-related context, consistent with the rest of the note.

Table 7: Case study on a clinical note with cardiac-related diseases (**bold**, in green). Dyspnea on exertion (*italic*, in red) can be caused by cardiac- or pneumonia-related diseases.

Clinical note	old male with multiple **cardiac risk factors** and *dyspnea on exertion* . . . , he then underwent further workup which included **a cardiac catheterization** that revealed significant **coronary artery disease**. he was then transferred for surgical evaluation".	
Prediction	Procedure codes	Diagnosis codes
Gold ICD codes	**36.15**; **39.61**;	401.9; 414.01; 250.00
MultiResCNN	*96.04*; *96.71*;	401.9; 414.01; 250.00
GatedCNN-NCI	**36.15**; **39.61**;	401.9; 414.01; 250.00

5 Conclusion

Medical code assignment from clinical notes is a fundamental task for healthcare information systems and diagnosis decision support. This paper proposes a novel framework with gated convolutional neural networks and note-code message passing mechanism for automated medical code assignment. Our solution can learn meaningful features from lengthy clinical documents and effectively control the deep propagation of information flow. Moreover, the message passing mechanism can enhance the ICD code space's semantics and model the note-code interaction to improve medical code prediction. Experiments show the effectiveness of our proposed method.

Acknowledgments

This work was supported by the Academy of Finland (grant 336033) and EU H2020 (grant

101016775). We acknowledge the computational resources provided by the Aalto Science-IT project. The authors wish to acknowledge CSC - IT Center for Science, Finland, for computational resources.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly Available Clinical BERT Embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2019. Trellis networks for sequence modeling. In *International Conference on Learning Representations*.

Tian Bai and Slobodan Vucetic. 2019. Improving Medical Code Prediction from Clinical Text via Incorporating Online Knowledge Sources. In *The World Wide Web Conference*, pages 72–82.

Tal Baumel, Jumana Nassour-Kassis, Raphael Cohen, Michael Elhadad, and Noemie Elhadad. 2018. Multi-label Classification of Patient Notes: Case Study on ICD Code Assignment. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Shengping Liu, and Weifeng Chong. 2020. HyperCore: Hyperbolic and Co-graph Representation for Automatic ICD Coding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3105–3114.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Cunxiao Du, Zhaozheng Chen, Fuli Feng, Lei Zhu, Tian Gan, and Liqiang Nie. 2019. Explicit interaction model towards text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6359–6366.

Richárd Farkas and György Szarvas. 2008. Automatic Construction of Rule-based ICD-9-CM Coding Systems. In *BMC Bioinformatics*, volume 9(Suppl 3), pages 1–9. Springer.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Shaoxiong Ji, Erik Cambria, and Pekka Marttinen. 2020. Dilated Convolutional Attention Network for Medical Code Assignment from Clinical Text. In *3rd Clinical Natural Language Processing Workshop at EMNLP*.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a Freely Accessible Critical Care Database. *Scientific Data*, 3:160035.

Ramakanth Kavuluru, Anthony Rios, and Yuan Lu. 2015. An Empirical Evaluation of Supervised Learning Approaches in Assigning Diagnosis Codes to Electronic Medical Records. *Artificial Intelligence in Medicine*, 65(2):155–166.

Yoon Kim. 2014. Convolutional Ceural Networks for Sentence Classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *EMNLP*.

Fei Li and Hong Yu. 2020. ICD Coding from Clinical Text Using Multi-Filter Residual Convolutional Neural Network. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Min Li, Zhihui Fei, Min Zeng, Fang-Xiang Wu, Yaohang Li, Yi Pan, and Jianxin Wang. 2018. Automated ICD-9 Coding via a Deep Learning Approach. *IEEE/ACM transactions on Computational Biology and Bioinformatics*, 16(4):1193–1202.

Lucian Vlad Lita, Shipeng Yu, Stefan Niculescu, and Jinbo Bi. 2008. Large Scale Diagnostic Code Classification for Medical Patient Records. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Julia Medori and Cédrick Fairon. 2010. Machine Learning and Features Selection for Semi-automatic ICD-9-CM Encoding. In *Proceedings of the NAACL*

HLT 2010 Second Louhi Workshop on Text and Data Mining of Health Documents, pages 84–89. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable Prediction of Medical Codes from Clinical Text. In *Proceedings of NAACL-HLT*, pages 1101–1111.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*.

Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. 2014. Diagnosis Code Assignment: Models and Evaluation Metrics. *Journal of the American Medical Informatics Association*, 21(2):231–237.

Aaditya Prakash, Siyuan Zhao, Sadid A Hasan, Vivek Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2017. Condensed Memory Networks for Clinical Diagnostic Inferencing. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. 2011. Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A Public-access Intensive Care Unit Database. *Critical Care Medicine*, 39(5):952.

Elyne Scheurwegs, Kim Luyckx, Léon Luyten, Walter Daelemans, and Tim Van den Bulcke. 2016. Data Integration of Structured and Unstructured Sources for Assigning Clinical Codes to Patient Stays. *Journal of the American Medical Informatics Association*, 23(e1):e11–e19.

Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P Xing. 2017. Towards Automated ICD Coding Using Deep Learning. *arXiv preprint arXiv:1711.04075*.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint Embedding of Words and Labels for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2017. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*.

Dynamic Semantic Graph Construction and Reasoning for Explainable Multi-hop Science Question Answering*

Weiwen Xu, Huihui Zhang, Deng Cai and Wai Lam
The Chinese University of Hong Kong
`{wwxu,hhzhang,wlam}@se.cuhk.edu.hk`
`thisisjcykcd@gmail.com`

Abstract

Knowledge retrieval and reasoning are two key stages in multi-hop question answering (QA) at web scale. Existing approaches suffer from low confidence when retrieving evidence facts to fill the knowledge gap and lack transparent reasoning process. In this paper, we propose a new framework to exploit more valid facts while obtaining explainability for multi-hop QA by dynamically constructing a semantic graph and reasoning over it. We employ Abstract Meaning Representation (AMR) as semantic graph representation. Our framework contains three new ideas: (a) `AMR-SG`, an AMR-based Semantic Graph, constructed by candidate fact AMRs to uncover any hop relations among question, answer and multiple facts. (b) A novel path-based fact analytics approach exploiting `AMR-SG` to extract active facts from a large fact pool to answer questions. (c) A fact-level relation modeling leveraging graph convolution network (GCN) to guide the reasoning process. Results on two scientific multi-hop QA datasets show that we can surpass recent approaches including those using additional knowledge graphs while maintaining high explainability on OpenBookQA and achieve a new state-of-the-art result on ARC-Challenge in a computationally practicable setting.

1 Introduction

Multi-hop QA is one of the most challenging tasks that benefits from explainability as it mimics the human question answering setting, where multi-hop QA requires both the collection of information from large external knowledge resources and the aggregation of retrieved facts to answer complex natural language questions (Yang et al., 2018).

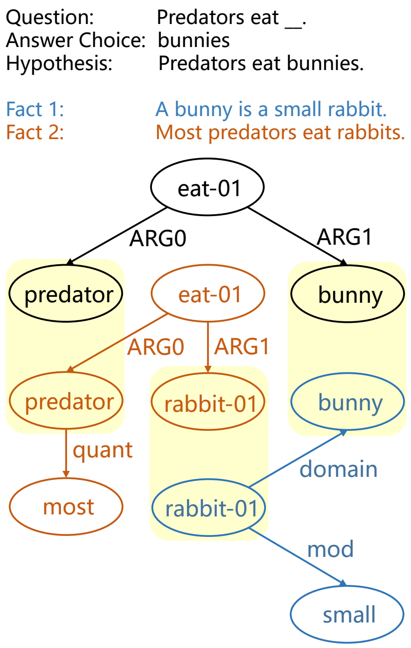
Question: Predators eat __.
Answer Choice: bunnies
Hypothesis: Predators eat bunnies.

Fact 1: A bunny is a small rabbit.
Fact 2: Most predators eat rabbits.

Figure 1: The AMR of the hypothesis (black), Fact 1 and Fact 2. A hypothesis is a statement derived from a question and a choice. The hypothesis AMR can be inferred by relevant fact AMRs.

Currently, external knowledge is mostly stored in two forms – textual and graph structure (e.g. Knowledge Graph (KG)). Textual corpora contain rich and diverse evidence facts, which are ideal knowledge resources for multi-hop QA. Especially with the success of pretrained models (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2019), we can get powerful representations for such textual facts. However, retrieving relevant and useful facts to fill the knowledge gap for inferring the answer is still a challenging problem. In addition, the reasoning process over the facts is hidden by the unexplainable neural network, which hinders the deployment of real-life applications. On the other hand, KG is able to provide structural clues about relevant entities for explainable predictions (Feng et al.,

---

[*] The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14204418).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1044–1056
August 1–6, 2021. ©2021 Association for Computational Linguistics

2020; Saxena et al., 2020; Xu et al., 2020). But it is known to suffer from sparsity, where complex question clues are unlikely to be covered by the closed-form relations in KG (Zhao et al., 2020; Zhang et al., 2020b). Another issue is that KG requires large human labor and is easy to become outdated if not maintained timely.

To take advantages of both rich textual corpora and explicit graph structure and make it compatible to all textual knowledge, we explore the usefulness of Abstract Meaning Representation (AMR) as a graph annotation to a textual fact. AMR (Banarescu et al., 2013) is a semantic formalism that represents the meaning of a sentence into a rooted, directed graph. Figure 1 shows some examples of AMR graphs, where nodes represent concepts and edges represent the relations. Unlike other semantic role labeling that only considers the relations between predicates and their arguments (Song et al., 2019), the aim of AMR is to capture every meaningful content in high-level abstraction while removing away inflections and function words in a sentence. As a result, AMR allows us to explore textual facts and simultaneously attributes them with explicit graph structure for explainable fact quality assessment and reasoning.

In this paper, we propose a novel framework that incorporates AMR to make explainable knowledge retrieval and reasoning for multi-hop QA. Our framework works on textual knowledge, which is easy to obtain and allows us to get informative facts. The introduced AMR serves as a bridge that enables an explicit reasoning process over a graph structure among questions, answers and relevant facts. As exemplified in Figure 1, a hypothesis is first derived from a question and an answer choice. We then parse the hypothesis and a large number of facts to corresponding AMRs. After that, we dynamically construct `AMR-SG` for each question-choice pair by merging the AMRs of its hypothesis and relevant facts. Unlike previous works on multi-hop QA that rely on existing KGs to find relations among entities (Wang et al., 2020; Feng et al., 2020), our proposed `AMR-SG` is dynamically constructed, which reveals intrinsic relations of facts and can naturally form any-hop connections. After construction, we analyze all connected paths starting from the question to the answer on `AMR-SG`. We focus the consideration of facts on those paths because they together connect the question with the answer, indicating their active roles in filling the knowledge gap. The connections of facts on `AMR-SG` can be further used as the supervision for downstream reasoning. Therefore, we adopt GCN (Kipf and Welling, 2017) to model the fact-level information passing.

Experimental results demonstrate that our approach outperforms previous approaches that use additional KGs. It obtains 81.6 accuracy on OpenBookQA (Mihaylov et al., 2018), and pushes the state-of-the-art result on ARC-Challenge (Clark et al., 2018) to 68.94 in a computationally practicable setting.

2 Related Work

Multi-hop QA with External Resource. Despite the success of pretrained model in most Natural Language Processing (NLP) tasks, it performs poorly in multi-hop QA, where some information is missing to answer questions (Zhu et al., 2021b).

Textual corpora contain rich and diverse knowledge, which is likely to cover the clues to answer complex questions. Banerjee et al. (2019) demonstrate some carefully designed queries can effectively retrieve relevant facts. Yadav et al. (2019); Deng et al. (2020) extract groups of evidence facts considering the relevance, overlap and coverage, but such method requires exponential computation in the retrieval step. Feldman and El-Yaniv (2019); Yadav et al. (2020) construct a fact chain by iteratively reformulating the query to focus on the missing information. However, the fact chain often grows obliquely as a result of the failure of first fact retrieval, making the QA model brittle. As some recent QA datasets (Yang et al., 2018; Mihaylov et al., 2018; Khot et al., 2020) annotate a gold evidence fact for each question, it enables training supervised classifier to identify the correct fact driven by a query (Nie et al., 2019; Qiu et al., 2019; Tu et al., 2020; Banerjee and Baral, 2020). Min et al. (2018) take a further step to jointly predict the answer span and select evidence facts in a unified model. Though these supervised retrievers have achieved impressive improvement, they heavily rely on the annotated gold facts, which are not always available in real-world applications.

In addition, previous works also explore the effectiveness of structured knowledge by either encoding the nodes (Yang and Mitchell, 2017; Wang et al., 2019), triples (Mihaylov and Frank, 2018; Wang et al., 2020), paths (Lin et al., 2019; Lei et al., 2020) or tabular (Zhu et al., 2021a) to capture the

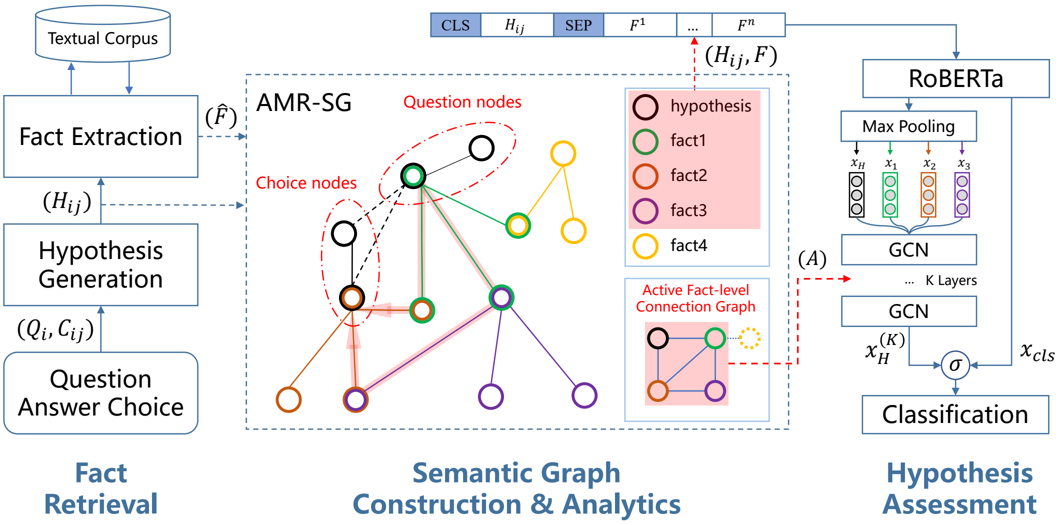

Figure 2: Overall architecture of our proposed model. The black dash lines in `AMR-SG` indicate that we cut the connection between question nodes and choice nodes. The pink arrows indicate two paths that can be spotted in `AMR-SG`. Facts with red background are active facts detected. The dashed node *Active Fact-level Connection Graph* indicates fact4 is not considered as a valid node as it is not an active fact.

missing information. Other works avoid the sparsity of KGs by constructing KGs directly from textual knowledge. OpenIE (Saha and Mausam, 2018) is widely used in knowledge base question answering to extract entity-relation triples (Bosselut et al., 2019; Zhao et al., 2020; Deng et al., 2019). However, OpenIE favors precision over recall, which is not necessarily effective to form connections among diverse evidence facts for multi-hop QA. Wikipedia contains internal hyperlinks, which are effective to build graph connections from unstructured articles (Asai et al., 2020; Liu et al., 2020). However, such hyperlinks are not available in most textual corpora.

AMR. Recent success in AMR research makes it possible to benefit downstream tasks, such as summarization (Takase et al., 2016; Dohare et al., 2017; Liao et al., 2018), event detection (Li et al., 2015) and machine translation (Song et al., 2019). In the domain of QA, AMR has been used to form logic queries and conduct symbolic reasoning (Mitra and Baral, 2016; Kapanipathi et al., 2020). Comparing to name entity (Zhong et al., 2020) or other cross-sentence annotations (Lei et al., 2018; Zhang et al., 2020a), we use AMR to build our semantic graph because it is align-free and can be easily adapted to powerful pretrained models.

3 Framework Description

In this paper, we consider the multi-hop QA in the form of multi-choice, where a question Q_i is provided with J answer choices $C_{ij}, j \in \{1, 2, ..., J\}$. As shown in Figure 2, our framework consists of three components: (1) a Fact Retrieval component to retrieve evidence facts $\hat{F} = \{\hat{F}^1, ..., \hat{F}^m\}$[1] for each question-choice pair from a large textual corpus; (2) a Semantic Graph Construction & Analytics component that dynamically constructs a semantic graph, named `AMR-SG`, to select active facts $F = \{F^1, ..., F^n\}$ from $\hat{F}$ and capture their relations A; and (3) a Hypothesis Assessment component that classifies whether the question-choice is correct, given the active facts and their relations in (2).

3.1 Fact Retrieval

Hypothesis Generation. As shown in Figure 2, we first generate a hypothesis H_{ij} for the i^{th} question and the j^{th} choice. A hypothesis is a completed statement derived from each question-choice pair. Comparing to simply concatenating the question and the choice, a hypothesis contains less meaningless words and maintain a good grammatical structure, which can avoid retrieving noisy facts and allow AMR parser to generate high-quality AMR graphs. We generate hypotheses by the rule-based model of Demszky et al. (2018). For some unsolvable cases, we directly concatenate the question and the choice. We apply this process for all training, develop and test sets.

---

[1] We omit the subscript ij for simplicity.

Fact Extraction. We retrieve a pool of evidence facts $\hat{F}$ for each hypothesis separately using *Elasticsearch* (Gormley and Tong, 2015). We set a large size m of the fact pool to cover as many valid facts as possible.

3.2 Semantic Graph Construction & Analytics

Active facts F are facts that really fill the knowledge gap between question and choice. The activeness of a fact cannot be simply determined by comparing it with the hypothesis, as multi-hop QA requires multiple facts to complete the reasoning chain. Therefore, we need to filter out facts that are just partially related and focus on the consideration of active facts and their roles in the reasoning chain. In this component, we first construct `AMR-SG`. Then, we propose a path-based analytics approach to extract active facts and construct an *Active Fact-level Connection Graph* to capture their relations with the question and the answer choice.

3.2.1 AMR-SG Construction

As the nodes of AMR are high-level abstraction of concepts conveyed in the corresponding textual fact, two AMRs sharing the same node indicate that they concern about the same concept, which shows their correlation. This motivates us to construct `AMR-SG`, shown in Figure 2, to represent the relations of the corresponding hypothesis and evidence facts for each question-choice pair.

We leverage the state-of-the-art AMR parser (Cai and Lam, 2020) to generate AMR $G = \{G^H, G^1, ..., G^m\}$ for a hypothesis and all facts in the corresponding fact pool, where G^H, G^i are the AMR of the hypothesis and the i^{th} fact respectively. AMR is also a directed and edge-labeled graph, which implies information specified in the edge is propagated in one predefined direction. However, such inner-AMR (edge labels and directions) information does not contribute to inter-AMR relations. Therefore, we only care about if there exists an edge between two nodes but ignore the edge labels and directions.

During construction, we regard G^H as the start point of `AMR-SG`. Then, we incrementally find one fact AMR in the fact pool sharing some nodes with it and add this fact AMR onto it by merging the shared nodes. The merging operation stops when no AMR can be added onto `AMR-SG` or the fact pool is empty. In fact, as shown in Figure 2, we do not change the architecture of each individual AMR, but reuse some shared nodes as the nodes in `AMR-SG`. Note that, some nodes are over-general, which are not appropriate to connect two AMRs (e.g. `(p/planet :name(n/name :op1"Earth"))`, the node `n/name` is an over-general concept). Fortunately, such over-general nodes always have non-node attributes (e.g. `Earth` of `n/name`) that shows the specific referent. Therefore, we replace the nodes with their non-node attributes if any to address this issue.

3.2.2 Path-based Analytics

Current multi-hop QA models are hindered by the quality of retrieved facts (Banerjee et al., 2019). We address this issue by a path-based analytics approach to guarantee the selected facts having a positive effect to answer the question.

As shown in Figure 2, `AMR-SG` reveals any-hop relations of the hypothesis and all facts. Completed paths can be spotted out of G^H to connect the question nodes with the choice nodes by passing through multiple facts. These facts, which together provide the missing knowledge to maintain complete reasoning chains, are active facts that we want to extract.

Specifically, we split the nodes of G^H into *question nodes* Q^H and *choice nodes* C^H. Question nodes represent the concepts extracted in the question text. As one question is provided with J choices, where we can generate J hypothesis AMRs. We take the shared nodes of these AMRs as Q^H, while the remaining as C^H:

$$Q_{ij}^H = \cap_{j=1}^J \{v | v \in G_{ij}^H\} \quad (1)$$

$$C_{ij}^H = \{v | v \in G_{ij}^H, v \notin Q_{ij}^H\}, j = 1, ..., J \quad (2)$$

We cut the edges between Q^H and C^H to guarantee the paths are spotted outside G^H. Then we apply depth-first search on `AMR-SG` to find all paths that connect at least one question node and one choice node, including the path that does not have a minimum length (e.g. the path passing through fact3 in Figure 2). All facts that the paths pass through (one node in and another node out) are considered as active facts. This is because we try to cover more facts as long as they do not deviate from the correct reasoning direction to provide enough information for QA model.

In addition, the any-hop relations of the hypothesis and active facts in `AMR-SG` can be used for a hypothesis to precisely aggregate knowledge from relevant facts to reduce ambiguity during the reasoning process. Therefore, we construct an *Active Fact-level Connection Graph* from `AMR-SG` to capture such relations among the hypothesis and all active facts. As shown in Figure 2, each node in *Active Fact-level Connection Graph* is either the hypothesis or an active fact. We draw an edge between two facts (include hypothesis) if they share one concept node in `AMR-SG`.

3.3 Hypothesis Assessment with Fact-level Reasoning

As shown in Figure 2, we concatenate the hypothesis with all active facts, where `[SEP]` token is inserted between the two texts and `[CLS]` is put at the beginning of the sequence. We feed the whole sequence into a pretrained model based on RoBERTa (Liu et al., 2019) architecture to get the hidden representation of each token.

Then, *Active Fact-level Connection Graph* is used as an additional supervision in fact-level modeling to guide the reasoning process. Formally, let $s^H_{1:l_H} \in \mathbb{R}^{l_H \times d}$, $s^i_{1:l_i} \in \mathbb{R}^{l_i \times d}$ be the hidden representations of the hypothesis and the i^{th} active fact respectively, where l_H, l_i denote the length and d is the dimension of the representation. A max pooling layer is applied over these hidden representations to get the node representations respectively:

$$\begin{aligned} x_H &= \textbf{MaxPool}(s^H_{1:l_H}) \in \mathbb{R}^{1\times d} \\ x_i &= \textbf{MaxPool}(s^i_{1:l_i}) \in \mathbb{R}^{1\times d}, i = 1, ..., n \end{aligned} \tag{3}$$

The connections of hypothesis (0^{th}) and active facts in *Active Fact-level Connection Graph* can be viewed as an adjacency matrix $A \in \mathbb{R}^{(n+1)\times(n+1)}$, where

$$A_{ij} = \begin{cases} 1 & \text{if } F^i \text{ is connected with } F^j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

As there is no edge information in the graph, a simple GCN is enough to model the knowledge fusion among the hypothesis and multiple active facts in the reasoning process. We also introduce multi-head mechanism (Vaswani et al., 2017) to stabilize the learning of different knowledge:

$$X^{(k)} = [head^{(k)}_1 : ... : head^{(k)}_h] \tag{5}$$

where $[:]$ denotes concatenation operation, $X^{(k)}$ is the node states at the k^{th} layer, $X^{(0)} = [x_H; x_1; ...; x_n]$, $[;]$ denotes the sequential concatenation operation, $head_i$ is the i^{th} head. Specifically, we compute the nodes states by aggregating knowledge from their neighboring nodes in each layer:

	Train	Dev	Test
OpenBookQA	4957	500	500
ARC-Challenge	8992	299	1172

Table 1: Number of instances in each dataset.

$$head^{(k)}_i = \text{ReLU}(\Lambda X^{(k-1)} W^{(k)}_i) \tag{6}$$

where $W^{(k)}_i \in \mathbb{R}^{d\times(d/h)}$ is the projection matrix of $head_i$ at the k^{th} layer, h is the head number. Λ is the normalization constant to avoid scale changing:

$$\begin{aligned} \Lambda &= D^{-1/2} A D^{-1/2} \\ D_{ii} &= \textstyle\sum_j A_{ij} \end{aligned} \tag{7}$$

After that, a σ gate is applied to calculate how much knowledge can be propagated to score the question-choice pair:

$$\lambda = \sigma(W^\lambda[x_{cls} : x^{(K)}_H] + b^\lambda) \tag{8}$$

$$s(q, a) = W^o(\lambda x^{(K)}_H + (1 - \lambda)x_{cls}) + b^o \tag{9}$$

where $W^\lambda \in \mathbb{R}^{1\times 2d}$, $W^o \in \mathbb{R}^{d\times d}$, b^λ, b^o are the parameters. We get the final probability by normalize all question-choice pairs with softmax.

4 Experiments

4.1 Datasets

We evaluate our approach on two multi-choice multi-hop QA datasets: ARC-Challenge (Clark et al., 2018) and OpenBookQA (Mihaylov et al., 2018). The textual corpus we use for both datsets is ARC Corpus (Clark et al., 2018), which contains about 14M science facts. OpenBookQA and ARC-Challenge have their leaderboards with train, develop and test sets publicly available. we follow AllenAI (2019) to combine the training set of OpenBookQA (4957), ARC-Easy (2251), ARC-Challenge (1119) and RegLivEnv (665) as the final training set of ARC-Challenge task. The data splits is shown in Table 1.

For ARC-Challenge, we retrieve 100 facts to form the fact pool. Based on this, we select up to 20 active facts using our approach as the context for each question-choice pair.[2] OpenBookQA provides an accompanying open-book of 1326 science

---

[2] We can only reproduce the results similar to AllenAI (2019) using 20 facts as the context.

facts, which are highly related to the questions in this dataset. Therefore, for OpenBookQA, we retrieve 10 facts from the open-book and another 90 facts from ARC Corpus, forming the 100 facts in the fact pool. We then select up to 15 active facts using our approach as the context.

4.2 Implementation

We implement our approach on two pretrained models: RoBERTa (Liu et al., 2019) and AristoRoBERTa (AllenAI, 2019). AristoRoBERTa employs the RoBERTa architecture but uses RACE (Lai et al., 2017) to first fine-tune the RoBERTa model. We prepare active facts as the context to further fine-tune the model with the target dataset. For OpenBookQA, we continue to fine-tune the QA model following the same procedure as AllenAI (2019), where the initial learning rate is 2e-5, the batch size is 12 and the max sequence length is 256. For ARC-Challenge, the initial learning rate, the batch size and the max sequence length are 1e-5, 6, and 416 respectively. We use grid search to find optimal hyper-parameters, where the learning rate is chosen from {5e-6, 1e-5, 2e-5}, the batch size is chosen from {4, 6, 8, 12, 16}. The number of GCN layer K is chosen from {1,**2**,3,4}, while the head number h is the RoBERTa-Large default value.[3]

We introduce 6M parameters of the fact-level reasoning module in addition to 355M of RoBERTa-Large. We run all experiments on one TITAN RTX card, which takes about 1 hour and 3 hours to complete the training of OpenBookQA and ARC-Challenge respectively.

4.3 Comparison Methods

We compare with recent existing methods that make use of similar power of pretrained models in order to conduct a fair comparison. These include the baseline AristoRoBERTaV7 (AllenAI, 2019) finetuned on top of AristoRoBERTa, KF-SIR (Banerjee and Baral, 2020) that exploits the knowledge fusion among facts, FreeLB (Zhu et al., 2020) that tackles the robustness issue and another three methods leveraging an additional knowledge graph (Speer et al., 2017) in addition to the textual knowledge: PG (Wang et al., 2020), MHGRN (Feng et al., 2020), AlBERT + KB. PG(albert + gpt2, roberta + gpt2) are two implementations with different pretrained model architectures (Liu et al., 2019; Lan et al., 2019; Radford et al., 2019), where the latter is more fair to compare with us.

Methods	Model Architecture	Additional KG	Test Acc.
PG	albert + gpt2	✓	81.8
PG	roberta + gpt2	✓	80.2
AlBERT + KB	albert	✓	81.0
MHGRN	roberta	✓	80.6
KF-SIR	roberta	×	80.0
AristoRoBERTaV7	roberta	×	77.8
+ AMR-SG-Full	roberta	×	**81.6**

Table 2: Test accuracy on OpenBookQA. Methods using additional KG are ticked.

5 Results

5.1 Main Results

OpenBookQA. The test set accuracy is shown in Table 2. AMR-SG-Full is our full model based on AristoRoBERTa. Results show that AMR-SG-Full can surpass models leveraging additional KG. It demonstrates that the fundamental improvement of AMR-SG-Full comes from the knowledge mining of the textual corpus. However, such knowledge resource has not been fully investigated by existing methods and contains richer and more diverse evidence facts than KGs. We do not compare with UnifiedQA (Khashabi et al., 2020) and T5 3B (Raffel et al., 2020) as they rely on extremely large pretrained models (at least 3B parameters), which are not fair for comparison.

ARC-Challenge. We also implement AMR-SG-Full on another difficult multi-hop QA dataset: ARC-Challenge. It consists of the questions only answered incorrectly by both a retrieval-based algorithm and a word co-occurrence algorithm (Clark et al., 2018), which theoretically is not friendly to our approach. As shown in Table 3, we can still obtain 2.47 accuracy improvement comparing to AristoRoBERTaV7 and achieve a new state-of-the-art performance in a computationally practicable setting.

5.2 Ablation Study

We conduct ablation study by incrementally adding each component of AMR-SG-Full to investigate its effectiveness on two pretrained models in Table 4. We include the analysis on RoBERTa because it is a more general and widely used pretrained model.

---

[3] Our code is available at: `https://github.com/wwxu21/AMR-SG`

Methods	Test Acc.
FreeLB (Zhu et al., 2020)	67.75
arcRoberta	67.15
xlnet+Roberta	67.06
AristoRoBERTaV7 (AllenAI, 2019)	66.47
+ AMR-SG-Full	**68.94**

Table 3: Test accuracy on ARC-Challenge. All models use RoBERTa architecture for the pretrained model and do not leverage additional KG.

Methods	Dev	Test
RoBERTa		
No Fact	66.8	64.8
+ Fact Context	68.2	68.8 (+4.0)
+ Fact Analytics	73.2	73.0 (+4.2)
+ Fact-level Reasoning	72.8	74.2 (+1.2)
AristoRoBERTa		
No Fact	71.0	70.0
+ Fact Context ♠	78.2	78.4 (+8.4)
+ Fact Analytics	79.4	81.4 (+3.0)
+ Fact-level Reasoning	**79.6**	**81.6** (+0.2)

Table 4: Ablation study of model components on OpenBookQA (adding one component incrementally). ♠ is our reimplementation of (AllenAI, 2019).

We start from the vanilla pretrained models, where no textual facts are provided (denoted as *No Fact*). We retrieve 15 facts as the context to create the first variant (denoted as + *Fact Context*). The purpose is to test the contribution of the facts retrieved by the simple information retrieval (IR) system (*Elasticsearch*). We continue to add the path-based fact analytics component (denoted as + *Fact Analytics*). In fact, this variant merely use the facts selected from `AMR-SG` to fine-tune the pretrained models. On top of both two pretrained models, we observe a great performance improvement, where the improvement brought by + *Fact Analytics* is higher than + *Fact Context* on top of RoBERTa, which demonstrates this component can effectively select useful facts to fill the knowledge gap that have not been covered by the IR system. We finally equip our model with the fact-level reasoning component (denoted as + *Fact-level Reasoning*). From the results, we can observe that this component performs well on top of RoBERTa, but has very little effect on top of AristoRoBERTa. This is because this component tries to infuse some fact-level connections to ease the reasoning process of the model. Such information can be learned automatically by the model itself if exposed to enough in-domain data (AristoRoBERTa). Nevertheless, the fact-level reasoning is a more general method when such data is unavailable.

6 Explainability Analysis

6.1 Analysis of AMR-SG

Impact of Evidence Facts. As discussed above, the major improvement of our approach comes from more useful facts selected for each question-choice pair. In this section, we take a deep look at the quality and the composition of those facts on OpenBookQA. We derive five variants by varying the composition of core (facts retrieved from open-book) or common (facts from ARC Corpus) facts. For core facts, as open-book annotates one gold core fact for each question, the retrieval accuracy of the gold fact is a natural way to evaluate the quality. For common facts, we conduct human analysis to evaluate the quality from three aspects: (1) Relatedness: Does the retrieved fact related to the question or the answer? (2) Informativeness: Does the retrieved fact provided useful information to answer the question? (3) Completeness: Do all retrieved facts together fill the knowledge gap to completely answer the question? We randomly sample 50 questions and evaluate the evidence facts corresponding to the correct answer choice, where one fact would contribute 1 score if it meets the requirement of Relatedness or Informativeness respectively and all 15 facts contribute 1 score if they together meet the requirement of Completeness. Evaluation results are presented in Table 5.

When varying the fact composition of IR variants, we find the gold core fact retrieval accuracy has a positive impact on the final accuracy on top of RoBERTa. At this stage, some questions can be inferred sufficiently with the gold core facts. Higher retrieval accuracy accounts for more questions of this kind to be correctly answered. However, this advantage is not as obvious for AristoRoBERTa. Our human evaluation reveals that such facts are unlikely to form a complete reasoning chain, making it hard for real multi-hop reasoning.

On the other hand, our approach directly models the intrinsic fact relations, where the path-based analytics ensures that the facts selected are in the reasoning chain from the question to the answer. Results show that our approach makes an overall improvement with regard to Relatedness, Informativeness and Completeness and is

Facts Composition	Core Fact	Human Evaluation			Test set Accuracy	
(total 15 facts)	Retrieval Accuracy	Rel.	Info.	Comp.	RoBERTa	AristoRoBERTa
IR (5/10)	56.4	5.86	2.50	0.46	68.8	78.4
IR (10/5)	63.6	5.20	2.24	0.42	70.4	77.4
IR (15/0)	68.4	3.36	1.62	0.26	72.2	77.4
`AMR-SG` (10/30)	61.0	5.85	2.58	0.48	72.4	80.4
`AMR-SG` (10/100)	61.0	6.22	2.98	0.56	**74.2**	**81.6**

Table 5: Automatic and Human Evaluation of the evidence facts on OpenBookQA. IR (x/y) indicates we use simple IR system to retrieve x core facts and y common facts. `AMR-SG` (x/y) indicates we construct `AMR-SG` with x core facts and y common facts, based on which we then select 15 active facts and extract their relations.

less harmful to core fact retrieval. We also find that `AMR-SG` (10/100) can make a further improvement compared to `AMR-SG` (10/30) by including more facts to construct `AMR-SG`. It demonstrates that `AMR-SG` has the capability of detecting useful facts from a large and noisy fact pool, thus making up for the deficiency of the IR system.

Impact of AMR Consistency. We investigate the quality consistency of AMR graphs to see how it affects the construction of `AMR-SG` and thus affects the QA model. We prepare AMR in three consistency levels, where *Fully-Automatic* is generated by automatic AMR parser; *Mixed* is that we manually annotate the error-free AMRs for the core facts in open-book (1326 in total) and use the error-free core fact AMRs and other automatically generated AMRs to construct `AMR-SG`; *Error-Free-Adapted* is that we use the error-free AMRs annotated to fine-tune the AMR parser and use the tuned parser to generate AMR for all the remaining facts (including hypotheses and common facts, about 900k in total). The test set accuracy are 81.6, 80.2, 80.4 for *Fully-Automatic*, *Mixed* and *Error-Free-Adapted* respectively. It is interesting to note that using *Fully-Automatic* AMRs results in higher QA accuracy than *Mixed* and *Error-Free-Adapted*, where the latter two contain a mix of AMRs with different levels of quality. This phenomenon has also been observed in other AMR applications (Liu et al., 2015; Hardy and Vlachos, 2018), where automatic parses perform well than manual parses. We conjecture that this can be attributed to the discrepancy between the error-free AMRs and the automatically parsed AMRs in the choices of AMR concepts with similar meaning. This small difference in concept choices may omit potential connections, results in some important facts failing to be detected. In contrast, automatically parsed AMRs contain errors, but they are consistent in their concept choices, which is more likely for AMRs to form connections. The 0.2 accuracy improvement between *Mixed* and *Error-Free-Adapted* also demonstrates our assumption, since the parser is finetuned on the error-free AMRs, where its parsed AMRs should be more consistent with the error-free AMRs.

Question: *A seismograph can accurately describe* (A) how rough the footing will be (B) how bad the weather will be **(C) how stable the ground will be** (D) how shaky the horse will be
Useful facts retrieved by IR: N.A.
Additional facts from path-based analytics: A seismograph is a kind of tool for measuring the size of an earthquake. An earthquake is a shockwave travelling through the ground.
Relevant path in `AMR-SG`: `seismograph→tool→measure-01→size-01→earthquake→ground`

Table 6: A case study showing how our framework selects useful facts to completely fill the knowledge gap.

6.2 Case Study

Table 6 shows one case study of evidence facts selected by our framework. Since the important term *earthquake* is missing from the search query, the IR system assigns low retrieval scores for the two facts, causing a low ranking. However, the two facts can form a complete reasoning chain with the question and the answer via several concept nodes, where our approach can successfully extract the two facts despite the low retrieval scores. More cases can be found in Appendix A.1.

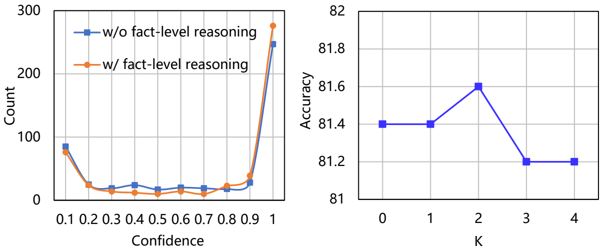

(a) (b)

Figure 3: Analysis of fact-level reasoning on OpenBookQA. (a) presents the distribution of prediction confidence with or without fact-level reasoning module. (b) shows the QA performance with different GCN layer K. Size 0 denotes the original pretrained model.

6.3 Analysis of Fact-level Reasoning

Why Fact-level reasoning. Figure 3(a) shows that fact-level reasoning improves the performance by making a more confident prediction for the correct answer. This is because the fact-level connections of `AMR-SG` inform the model how these active facts are intrinsically related, which allows the model to precisely receive knowledge from related facts.

Impact of Number of Hops (K). We vary the hyper-parameter K to consider the impact of K-hop neighbors on OpenBookQA. As show in Figure 3(b), the performance reaches the top at $K = 2$. It indicates that most of the questions can be well answered using two evidence facts, which is consistent with the construction of this dataset. However, the performance drops when $K > 2$. It might be attributed to exponential noise found in longer reasoning chains.

7 Conclusion

We propose to dynamically construct `AMR-SG` that can reflect the intrinsic relations of relevant facts leveraging AMR, a graph annotation. `AMR-SG` combines the advantages of rich textual corpus and graph structure, where we can select useful facts that completely form the reasoning chain and make fact-level modeling. Experimental results show that `AMR-SG` can maintain high explainability, and successfully couple with strong pretrained models to achieve significant improvement on OpenBookQA and ARC-Challenge over approaches leveraging additional KGs.

References

AllenAI. 2019. Aristorobertav7. In *AristoRoBERTAv7*.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Pratyay Banerjee and Chitta Baral. 2020. Knowledge fusion and semantic knowledge ranking for open domain question answering. *arXiv preprint arXiv:2004.03101*.

Pratyay Banerjee, Kuntal Kumar Pal, Arindam Mitra, and Chitta Baral. 2019. Careful selection of knowledge to solve open book question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6120–6129, Florence, Italy. Association for Computational Linguistics.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Yang Deng, Yuexiang Xie, Yaliang Li, Min Yang, Nan Du, Wei Fan, Kai Lei, and Ying Shen. 2019. Multi-task learning with multi-view attention for answer selection and knowledge base question answering. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 6318–6325.

Yang Deng, Wenxuan Zhang, and Wai Lam. 2020. Multi-hop inference for question-driven summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6734–6744.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shibhansh Dohare, Harish Karnick, and Vivek Gupta. 2017. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2296–2309, Florence, Italy. Association for Computational Linguistics.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine.* " O'Reilly Media, Inc.".

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramon Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, et al. 2020. Question answering over knowledge bases by leveraging semantic parsing and neuro-symbolic reasoning. *arXiv preprint arXiv:2012.01707*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI*, pages 8082–8090.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Wenqiang Lei, Yuanxin Xiang, Yuwei Wang, Qian Zhong, Meichun Liu, and Min-Yen Kan. 2018. Linguistic properties matter for implicit discourse relation recognition: Combining semantic interaction, topic continuity and attribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive path reasoning on graph for conversational recommendation. *Proceedings of the 26th ACM SIGKDD*.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with Abstract Meaning Representation. In *Proceedings of the First Workshop on Computing News Storylines*, pages 11–15, Beijing, China. Association for Computational Linguistics.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020. Rikinet: Reading wikipedia pages for natural question answering.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1725–1735, Melbourne, Australia. Association for Computational Linguistics.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2553–2566, Hong Kong, China. Association for Computational Linguistics.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas. Association for Computational Linguistics.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *AAAI*, pages 9073–9080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7208–7215.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2020. Fusing context into knowledge graph for commonsense reasoning. *arXiv preprint arXiv:2012.04808*.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2578–2589, Hong Kong, China. Association for Computational Linguistics.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2020. Unsupervised alignment-based iterative evidence retrieval for multi-hop question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4514–4525, Online. Association for Computational Linguistics.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, Vancouver, Canada. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Wenxuan Zhang, Yang Deng, and Wai Lam. 2020a. Answer ranking for product-related questions via multiple semantic relations modeling. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, pages 569–578.

Yao Zhang, Xu Zhang, Jun Wang, Hongru Liang, Adam Jatowt, Wenqiang Lei, and Zhenglu Yang. 2020b. Gmh: A general multi-hop reasoning model for kg completion.

Chen Zhao, Chenyan Xiong, Xin Qian, and Jordan Boyd-Graber. 2020. Complex factoid question answering with a free-text knowledge graph. In *Proceedings of The Web Conference 2020*, pages 1205–1216.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online. Association for Computational Linguistics.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021a. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021b. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

A Appendix

A.1 Case Study

More case studies can be found in Table 7.

(a) Case Study 1

Question:	*Algae can be found in* **(A) reservoir** (B) meat (C) street (D) tree
Useful facts retrieved by IR:	Algae is found in bodies of water.
Additional facts from path-based analytics:	Water reservoir: a large quantity of water is stored in a reservoir (or dam).
Relevant path in `AMR-SG`:	`Algae → find-01 → body → water → store-01 → reservoir`

(b) Case Study 2

Question:	*Photosynthesis can be performed by* **(A) a cabbage cell** (B) a bee cell (C) a bear cell (D) a cat cell
Useful facts retrieved by IR:	N.A.
Additional facts from path-based analytics:	Plant cells can perform photosynthesis. Description: skunk cabbage is a flowering perennial plant that is one of the first plants to emerge in the spring.
Relevant path in `AMR-SG`:	`Photosynthesis → plant → cabbage`

(c) Case Study 3

Question:	*Which is recyclable?* (A) An Elephant **(B) A school notebook** (C) A boat (D) A lake
Useful facts retrieved by IR:	Paper is recyclable.
Additional facts from path-based analytics:	Take notes on notebook paper.
Relevant path in `AMR-SG`:	`recycle-01 → paper → notebook`

(d) Case Study 4

Question:	*Which requires energy to move?* **(A) weasel** (B) willow (C) mango (D) poison ivy
Useful facts retrieved by IR:	An animal requires energy to move.
Additional facts from path-based analytics:	The long and slender body of the weasel allows it to move, almost flow, over terrain.
Relevant path in `AMR-SG`:	`energy → move-01 → weasel`

(e) Case Study 5

Question:	*A person wants to be able to have more natural power in their home. They choose to cease using a traditional electric company to source this electricity, and so decide to install* (A) sun grafts (B) sunlight shields **(C) panels collecting sunlight** (D) solar bees
Useful facts retrieved by IR:	A home with solar electric panels on the roof might be able to make most of its own electricity, for example.
Additional facts from path-based analytics:	Solar thermal panels generate hot water from the natural energy in sunlight.
Relevant path in `AMR-SG`:	`natural-03 → energy → generate-01 → sunlight → panel`

Table 7: More case studies in addition to Table 6

Addressing Inquiries about History: An Efficient and Practical Framework for Evaluating Open-domain Chatbot Consistency

Zekang Li[1,2], **Jinchao Zhang**[3], **Zhengcong Fei**[1,2], **Yang Feng**[1,2]*, **Jie Zhou**[3]
[1] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2] University of Chinese Academy of Sciences
[3] Pattern Recognition Center, WeChat AI, Tencent Inc, China
`{lizekang19g,feizhengcong,fengyang}@ict.ac.cn`
`{dayerzhang,withtomzhou}@tencent.com`

Abstract

A good open-domain chatbot should avoid presenting contradictory responses about facts or opinions in a conversational session, known as its consistency capacity. However, evaluating the consistency capacity of a chatbot is still challenging. Employing human judges to interact with chatbots on purpose to check their capacities is costly and low-efficient, and difficult to get rid of subjective bias. In this paper, we propose the **A**ddressing **I**nquiries about **H**istory (**AIH**), an efficient and practical framework for the consistency evaluation. At the conversation stage, **AIH** attempts to address appropriate inquiries about the dialogue history to induce the chatbot to redeclare the historical facts or opinions. We carry out the conversation between chatbots, which is more efficient than the human-bot interaction and can also alleviate the subjective bias. In this way, we manage to rapidly obtain a dialog session that contains responses with high contradiction possibilities. At the contradiction recognition stage, we can either employ human judges or a natural language inference (NLI) model to recognize whether the answers to the inquiries are contradictory with history. Finally, we are able to rank chatbots according to the contradiction statistics. Experiments on open-domain chatbots show that our approach can efficiently and reliably assess the consistency capacity of chatbots and achieve a high ranking correlation with the human evaluation. We release the framework and hope to help improve the consistency capacity of chatbots.[1]

	DialoGPT 762M
Human:	*What movies do you like most?*
DialoGPT:	*The Unforgiven from Metallica.*
Human:	*What do you think of the Unforgiven?*
DialoGPT:	*I never heard of it.*
	Blender 2.7B
Human:	*What do you like to cook?*
Blender:	*I only cook Indian cuisine. How about you?*
Human:	*I enjoy cooking Chinese food, especially the dumplings.*
Blender:	*Chinese food is delicious. I also like cooking the Chinese food.*
	Plato 1.6B
Human:	*Do you like dogs?*
Plato:	*Yes, dogs are cool, I prefer cats though.*
Human:	*Do you have a dog?*
Plato:	*I do not, but I'd love a dog. I think dogs are so cute.*
Human:	*I love dogs very much. I have three dogs.*
Plato:	*I have three dogs too. One puppy and two Huskies.*

Table 1: Several human-bot conversations demonstrate that popular chatbots (DialoGPT, Blender, and Plato) generate inconsistent responses when talking to a human under some specific conditions.

1 Introduction

In recent years, open-domain intelligent chatbots (Adiwardana et al., 2020b; Bao et al., 2020; Smith et al., 2020) have approached great progress due to the development of the large-scale pre-training approaches (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020) and the large amount of high-quality conversational datasets (Dinan et al., 2019; Baumgartner et al., 2020; Smith et al., 2020). Though the success is indisputable and exciting, there is still a long way to build a truly human-like open-domain chatbot.

Current open-domain chatbots hold a superiority in generating fluent, engaging, and informative responses, but show the soft spot on consistency (Nie et al., 2020). As shown in Table 1, we present some interactive dialogue samples between human and several popular open-domain chatbots (e.g. DialoGPT (Zhang et al., 2020), Blender (Smith et al., 2020), and Plato (Bao et al., 2020)). All open-

---

[*] Joint work with Pattern Recognition Center, WeChat AI, Tencent Inc. Yang Feng is the corresponding author. Work was done when Zekang Li and Zhengcong Fei were intern at WeChat AI.

[1] `https://github.com/ictnlp/AIH`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1057–1067
August 1–6, 2021. ©2021 Association for Computational Linguistics

domain chatbots occasionally generate responses that are contradictory with history when interacting with humans, which is really annoying and severely disrupts the communication once happening. Therefore, it is imperative to improve the consistency of the open-domain chatbots. However, one crucial reason that restricts consistency development is the lack of an effective and practical evaluation method.

To estimate the consistency of chatbots, the most straightforward approach is to ask human annotators to distinguish whether the conversations generated from the chatbots are consistent or not. However, the instructions followed by annotators are often chosen ad-hoc, and there is no explicit definition, which leads to the relatively low inter-agreement in the human chatbot consistency evaluation (Mehri and Eskénazi, 2020). As a result, several works have been proposed to develop automatic evaluation methods (Welleck et al., 2019; Song et al., 2020; Nie et al., 2020). While these methods can detect contradictions efficiently in the dialogue, they depend on the human-bot conversations, which is still cost-inefficient and tend to suffer from low quality (Deriu et al., 2020; Dinan et al., 2020). Besides, the occurrence rate of contradiction is low under this condition. All these problems slow down the development of consistency evaluation of dialogue systems severely.

Towards that end, based on the observations: (i) chatbots are likely to generate contradictions when chatting about facts and opinions; (ii) answering the questions about the conversational history correctly can reveal the ability to understand the conversation and keep consistency, we present the *Addressing Inquiries about History* (AIH) framework, an effective and practical framework for open-domain chatbot consistency evaluation. The framework can be used to rank different chatbots with regard to the ability to be consistent with themselves in the conversation. Specifically, AIH consists of two stages: (i) during the inquiry stage, questions about the facts and opinions mentioned in the conversation history are inserted into the conversation between chatbots; (ii) during the contradiction recognition stage, the responses of the inserted questions are collected, and automatic models or human judges can be adopted to decide whether the responses are consistent with the dialogue history.

In brief, our AIH has the following key advantages: *Firstly*, it is based on bot-bot conversation, which avoids the human intervention and brings down the cost and time effort significantly. *Secondly*, by inserting specific questions, contradictions occur more frequently, and it is easier for human annotators or automatic consistency detection model to distinguish the contradiction compared with natural conversations. Extensive experiments demonstrate that the proposed framework can produce effective, efficient, and reliable consistency evaluation. Furthermore, we also make an in-depth discussion about the influence of question generation, contradiction detection, and evaluation agreement in our framework.

Our contributions are summarized as follows:

- We propose the *Addressing Inquiries about History*(AIH), an effective and practical framework for open-domain chatbot consistency evaluation.
- Experiments show that AIH can produce effective, efficient, and reliable consistency evaluation. We release the framework as a ready-to-use tool for evaluating the consistency of chatbots. We hope AIH can facilitate and provide standard evaluation for future work on developing self-consistent open-domain chatbots.

2 Related Work

There are various methods to evaluate the consistency of chatbots, containing automatic and human-based methods. The methods mainly fall into two dimensions: the static and interactive evaluation.

2.1 Static Evaluation

Static evaluation denotes evaluating if the responses generated based on the static context are contradictory with the pre-defined persona or profile and the dialogue history by neural models or human annotators. Welleck et al. (2019) and Song et al. (2020) focus on the persona-related consistency and profile-related consistency, and characterize the chatbot consistency evaluation as the natural language inference problem. Nie et al. (2020) build a new human-craft dataset called DECODE and propose a structured utterance-based approach to detect the contradictions in the dialogue history. While being cost-efficient, static evaluation can not accurately reflect the conversation capacity of the chatbot in the real world.

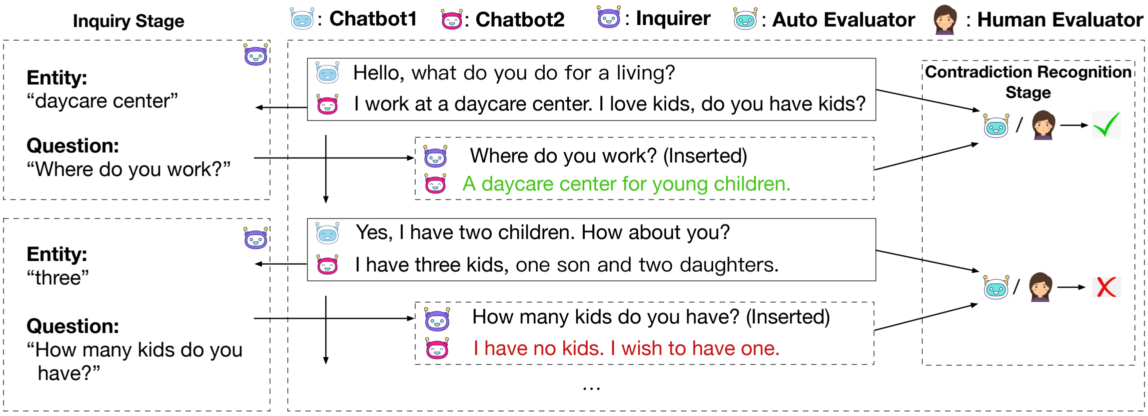

Hello, what do you do for a living?

I work at a daycare center. I love kids, do you have kids?

Where do you work? (Inserted)

A daycare center for young children.

Yes, I have two children. How about you?

I have three kids, one son and two daughters.

How many kids do you have? (Inserted)

I have no kids. I wish to have one.

Figure 1: Overview of the Addressing Inquiries about History framework. There are five “agents”: **Chatbot1**, **Chatbot2**, **Inquirer**, **Auto Evaluator**, and **Human Evaluator** and two stages: **Inquiry Stage** and **Contradiction Recognition Stage** in the framework. Chatbot1 and Chatbot2 are the participants in the bot-bot conversation, in which Chatbot2 is the one to be evaluated. In the Inquiry stage, Inquirer extracts opinion- or fact-related entities and generate inquiries based on them. In the Contradiction Recognition stage, the Auto Evaluator is generally a contradiction detection model to automatically evaluate if the responses from Chatbot2 are consistent and the Human Evaluator can also be employed for more accurate evaluation. Note that the inserted inquiries do not affect the natural conversation. Better view in color.

2.2 Interactive Evaluation

Human-bot Conversations. In order to pursue more authentic evaluation, the standard method is to let humans converse with a chatbot and evaluate it by aforementioned models or humans afterward (Mehri and Eskénazi, 2020). However, apart from the high cost of collecting human-bot conversations, there is also a high cognitive strain on humans, which leads to unstable results (Dinan et al., 2020).

Bot-bot Conversations. Recently, bot-bot conversations, which significantly reduce the cost and human bias, are focused. Deriu et al. (2020); Li et al. (2019) propose to let humans evaluate bot-bot or self-talk conversations to give a relative ranking of the overall quality of chatbots. Different from these methods, we focus on the chatbot consistency and insert inquiries to redeclare historical facts. And we introduce both automatic and human approaches to evaluate the chatbot consistency.

3 Approach

In this section, we first provide an overview of the *Addressing Inquiries about History* (AIH) framework. We then describe the Inquiry stage, the Contradiction Recognition stage, and the chatbot ranking process.

3.1 Overview

To estimate the consistency capacity, questions about the opinions and facts in the dialogue history are inserted into the current bot-bot conversation. Then, the corresponding responses are collected and judged by automatic tools or human evaluation. The workflow of our proposed AIH framework is shown in Figure 1.

To be specific, there are five “agents” in the framework: Chatbot1, Chatbot2, Inquirer, Auto Evaluator, and Human Evaluator. The Chatbot1 and Chatbot2 are the participants in the bot-bot conversation. The Inquirer extracts opinion- or fact-related entities and generates inquiries based on the entities. The Auto Evaluator is generally a contradiction detection model to automatically evaluate if the responses from Chatbot2 are consistent. The Human Evaluator is used for more accurate evaluation.

Formally, assume a pool of N chatbots $\{B_1, ..., B_N\}$ which are ready to be evaluated in terms of consistency capacity. For each pair of chatbots (referred as Chatbot1 and Chatbot2), we let Chatbot1 talk with Chatbot2 for K turns. Note that Chatbot2 is the one to be evaluated. (i) During the inquiry stage, within the conversation between Chatbot1 and Chatbot2, for each utterance u_{2k} generated by Chatbot2, Inquirer extracts the entities about opinions and facts, then asks Chatbot2 a question q_k about these entities, where k is the turn number. Chatbot2 answers the question q_k and generates the corresponding response r_k. Note that we ignore the questions generation operation when there is no entity that can be extracted. (ii) During the contradiction recognition stage, we use neural models (e.g. Natural Language Inference Model) or employ human judges to decide if the utterance pair $\{u_{2k}, r_k\}$ exists non-consistent problem. We collect at least M dialogues from each chatbot pair, then calculate the ranking order on the consistency. In this way, we can discriminate the consistency capability of chatbots effectively and efficiently. In the following, we will introduce the inquiry stage and the contradiction recognition stage, respectively.

3.2 Inquiry Stage

Based on our observation and prior work (Nie et al., 2020), in natural human-bot or bot-bot conversation, contradiction is more likely to occur when chatting about repeated facts and opinions, especially after similar questions. Therefore, to mimic such a contradiction occurrence process, we make chatbots to produce responses by asking chatbots related questions about previous facts and opinions. In this condition, generating appropriate questions is pretty important. Hence, we first extract entities about facts and opinions from the historical utterances, then employ a neural model to generate questions about the extracted entities.

Entity Extraction

Considering that chatbots usually generate contradictions when chatting about facts and opinions, we apply Named Entity Recognition tools in Stanza (Qi et al., 2020), a popular natural language analysis package, to extract named entities from utterance u_{2k} containing person, organization, location, etc. [2] For example, for the utterance *"I would love to visit New York next year."*, we can extract out two entities: *"New York"* and *"next year"*.

Question Generation Model

For question generation, we employ UniLM (Dong et al., 2019) model that is fine-tuned on the SQuAD dataset (Rajpurkar et al., 2016) with question generation task (Wangperawong, 2020). We utilize a public implementation and checkpoint.[3] In our framework, given the entities extracted before and utterance, UniLM generates a suitable question for each entity. For example, given *"New York"* and *"I would love to visit New York next year."*, the model generates *"Where would you like to visit next year?"*. We then randomly select one question and insert it into the bot-bot conversation.

3.3 Contradiction Recognition Stage

In our framework, since the question q_k is based on the previous Chatbot2's utterance u_{2k}, the response from Chatbot2 should be consistent with the utterance u_{2k}. Therefore, the Auto Evaluator and Human Evaluator can just consider the answer r_k and utterance u_{2k}.

Auto Evaluator

For automatic evaluation, the Auto Evaluator is generally a contradiction detection model. The Auto Evaluator take the response r_k answered by Chatbot2 and the previous utterance u_{2k} as input, and output the contradiction score y_k. It can be formulated as:

$$y_k = f_\theta(r_k, u_{2k}), \tag{1}$$

where f_θ is the detection function and θ is the parameters. Compared to other contradiction detection methods that consider the whole dialogue, the Auto Evaluator can refrain from the noise contained in the whole dialogue. In practice, we select the Roberta-large model (Liu et al., 2019) fine-tuned on the Multi-Genre Natural Language Inference dataset (Williams et al., 2018) as the implementation of Auto Evaluator.[4]

Human Evaluator

In traditional dialogue consistency evaluation methods, human judges are asked to read the whole dialogue and give an overall consistency score, usually 0 or 1. In our opinion, these methods suffer from high cost and low inter-agreement because there is no specific instruction, and it is too hard for human judges to give an overall score on the whole dialogue (Mehri and Eskénazi, 2020).

In our framework, human evaluators are only asked to decide if the response r_k answered by Chatbot2 is consistent with the previous utterance u_{2k} or not, which is more specific and easier than the traditional methods. As a result , the cost decreases, and the evaluation quality increases. Besides, the human annotation in our framework is much more fine-grained than the traditional methods, which can provide more information for the development cycle of dialogue systems.

---

[2] There are 18 named entity types. Please refer to (Weischedel et al., 2013) for more details.

[3] `https://github.com/artitw/text2text`

[4] `https://huggingface.co/roberta-large-mnli`

3.4 Consistency Metrics and Bot Ranking

Based on the previous results, we can obtain a ranking list about different chatbots on consistency capacity. Formally, for each chatbot pair $\{B_i, B_j\}$, we collect M dialogues. For each inquiry pair, the detection of contradiction is made by comparing y_k with a threshold τ:

$$c_k = \mathbb{I}(f_\theta(r_k, u_{2k}) > \tau). \tag{2}$$

The contradiction rate of the chatbot B_j within chatbot pair B_{ij} can be computed as:

$$C_{ij} = \frac{1}{M} \sum^{m} c_k, \tag{3}$$

where m is the number of inquiries in each dialog and M is the total number of inquiry pairs. For the overall contradiction rate of the chatbot B_j is calculated as:

$$C_j = \frac{1}{N} \sum_{i=1}^{N} C_{ij}. \tag{4}$$

Finally, we can rank the chatbots using the overall contradiction rate.

4 Experiment Setup

In this section, we first list the dialogue systems used in our experiments, then describe the experimental settings in detail.

4.1 Chatbots

We select several popular open-domain chatbots in our experiments.

Blender (BL) (Adiwardana et al., 2020a) is firstly pre-trained on Reddit dataset (Baumgartner et al., 2020) and then fine-tuned with high-quality human annotated dialogue datasets (BST), which containing four datasets: Blended Skill Talk (Smith et al., 2020), Wizard of Wikipedia (Dinan et al., 2019), ConvAI2 (Dinan et al., 2020), and Empathetic Dialogues (Rashkin et al., 2019). By fine-tuning, Blender can learn blend conversational skills of engagement, knowledge, empathy and personality. Blender has three model sizes: 90M, 2.7B, and 9.4B. Since 2.7B parameter model achieves the best performance in (Adiwardana et al., 2020a), we use the 2.7B version in our experiments.

Plato (PL) (Bao et al., 2020) is an open-domain chatbot, pre-trained on Reddit dataset and fine-tuned with BST dataset, which is claimed to be superior to Blender. According to the evaluation in (Bao et al., 2020), we select the 1.6B parameter version in our experiments.

DialoGPT (DG) (Zhang et al., 2020) is trained on the basis GPT-2 (Radford et al., 2019) using Reddit comments. There are three model sizes: 117M, 345M, and 762M. We fine-tuned the 762M version on the BST datasets.

DialoFlow (DF) (Li et al., 2021a,b) is a top method in DSTC9 Interactive Dialogue Evaluation track (Gunasekara et al., 2021). We reproduced the DialoFlow model based on GPT2-large (Radford et al., 2019) and fine-tuned it with BST dataset.

4.2 Experimental Settings

We adopt four experimental paradigms to evaluate the effectiveness of the AIH.

Bot-Bot Interaction. For bot-bot interaction, the maximum interaction turn is set to 15. All chatbots exploit Nucleus Sampling (Holtzman et al., 2020) with p=0.9 when generating responses. For each chatbot pair, we collect at least 200 dialogues.

Human Annotation. To verify the effectiveness of our framework, we conduct the human evaluation. For the bot-bot conversation under our framework, we employ three professional human annotators from a commercial data annotating company to separately annotate three fields: whether inquiry chatbot generates appropriate questions, whether Chatbot2 answers the questions relevantly, and whether the responses from chatbot2 are contradictory with the dialogue history. We pay the company reasonable salary. For each chatbot pair, we randomly sample 50 dialogues to be annotated. We compute the final decision via voting.

In Human-bot Natural Interaction and Expert Evaluation, we deployed the four chatbots on the remote server and designed a web interface. Human could chat with a random chatbot each time through the web interface and give the consistency score, being unaware of which chatbot they are chatting with.

Human-Bot Natural Interaction. For each chatbot, we filtered out the dialogues with <5 turns and the dialogues with abusive words. For each chatbot, there are at least 40 eligible dialogues. Then we employ the three professional human annotators to individually annotate whether each utterance from the chatbot is consistent or not.

Expert Evaluation. To obtain the human ranking

Expert Consistency Score ↑				
	BL	**PL**	**DG**	**DF**
Expert.1	0.55	0.80	0.72	0.69
Expert.2	0.37	0.87	0.60	0.56
Expert.3	0.31	0.89	0.60	0.55
Avg.	0.41	**0.85**	0.64	0.60

Table 2: The expert consistency score of each chatbot. Higher is better.

Contradiction Rate (Auto $\tau = 0.15$) ↓					
	BL	**PL**	**DG**	**DF**	**Avg.**
BL	0.431	0.240	0.324	0.362	0.339
PL	0.431	0.263	0.293	0.357	0.336
DG	0.425	0.251	0.344	0.345	0.341
DF	0.427	0.264	0.344	0.371	0.351
Avg.	0.428	**0.255**	0.326	0.359	0.342
Contradiction Rate (Human) ↓					
	BL	**PL**	**DG**	**DF**	**Avg.**
BL	0.487	0.282	0.398	0.396	0.391
PL	0.411	0.212	0.500	0.435	0.390
DG	0.404	0.211	0.304	0.431	0.338
DF	0.462	0.268	0.310	0.377	0.354
Avg.	0.441	**0.243**	0.378	0.410	0.368

Table 3: The contradiction rate of each chatbot pair. The column name and the row name represent Chatbot1 and Chatbot2 respectively.

for the consistency of the chatbots, we invite three expert volunteers from our lab, who have 2-3 years experience of dialogue system development, to chat with each bot at least 10 times and about 15 turns each time. In the chatting, experts are asked to intentionally induce the chatbots to re-answer the questions about the dialogue history and give the consistency score from 0 to 1. Note that we ask the experts to chat with the chatbots for >20 times before the formal evaluation. We average the scores from three experts as the overall consistency score.

Note that Expert Evaluation and Human Annotation were done before the automatic evaluation. Human-bot Natural Interaction was done after the automatic evaluation. All human evaluations were independent from the automatic evaluation.

5 Experimental Results

In this section, we conduct experiments to illustrate the effectiveness, efficiency, and stability of the proposed AIH framework.

5.1 Evaluation Effectiveness

We report the expert ranking results in expert evaluation, automatic evaluation, and human evaluation under the AIH framework, respectively.

Expert Ranking. Table 2 shows the expert consistency scores for different chatbots. We can find that Plato achieves the best expert consistency score, up to 0.85. And the ranking of consistency for these four chatbots is: Plato > DialoGPT > DialoFlow > Blender, which can serve as the gold reference.

Auto Evaluation Results. Table 3 shows the contradiction rate of each chatbot pair in auto evaluation. The lower contradiction rate means the better consistency. The column name and the row name represent Chatbot1 and Chatbot2, respectively. The "Avg." in column name represents the overall contradiction rate of each chatbot. The "Avg." in row name can be regarded as the ability to induce other chatbots to redeclare about the facts or opinions that are likely to be contradictory. In the automatic evaluation, the ranking of consistency for the chatbots is Plato > DialoGPT > DialoFlow > Blender, which is the same with expert evaluation. The Blender reaches the highest contradiction rate.

Human Evaluation Results. We list the evaluation results in the bottom of Table 3. As we expected, BL obtains the highest contradiction rate. Meantime, human evaluation also provides the same consistency ranking: Plato > DialoGPT > DialoFlow > Blender as before.

Summary. Both automatic evaluation and human evaluation in our framework can give the same performance ranking with the expert, which demonstrates that our framework is general and can effectively evaluate the consistency of chatbots.

5.2 Time Efficiency

Prior consistency evaluation methods with human-bot interaction are costly and take up a long time, which seriously slows down the development cycle of dialogue systems. In this section, we try to illustrate that our proposed *Addressing Inquiries about History* framework is time and cost efficient and can help the evolution process of dialogue systems compared to the other methods.

As shown in Table 4, we compare the time cost on two aspects: (i) the time to create inquires, and (ii) the time to detect contradictions in conversation. *Addressing Inquiries about History* framework is based on the bot-bot conversation so that the time to create conversation can be ignored, while the

Method	Time (Sec)	Contradiction
AIH (Auto)	- + -	1.56
AIH (Human)	- + 24	1.69
Human-bot	246 + 59	0.50

Table 4: The time efficiency of our proposed *Addressing Inquiries about History* framework and traditional evaluation method with human-bot interaction. "Time" represents the time to create a conversation and the time to annotate the contradictions in a conversation. "Contradiction" denotes the average number of contradictions per conversation (average 15 turns). Dash line denotes the time can be ignored.

human-bot conversation takes around 4 minutes per conversation. For the contradiction detection time, prior methods take around 1 minute considering the whole dialogue, while in our proposed framework, it is only about 24 seconds for human annotation or ignored for automatic evaluation. Besides, we also compare the number of contradictions per conversation. As shown in Table 4, in our framework, the chatbots generate much more contradictions than those in prior methods. The detected contradictions are helpful for the chatbot developer to further improve the consistency of the chatbot.

Summary. Our proposed framework can detect more contradictions with much less time than previous methods. Correspondingly, *Addressing Inquiries* framework will accelerate the evolution process of consistency of chatbots.

5.3 Ranking Stability

One key requirement for an evaluation framework is that repeated executions of the procedure result in the same outcomes. We measure how many conversations between each chatbot pair are required to guarantee a stable ranking. We randomly sample $\hat{S}$ conversations for each chatbot pair and compute the consistency ranking using automatic evaluation, where $\hat{S} \in \{1, \cdots, 200\}$. We repeat this sub-sampling procedure 1000 times and compute the accuracy of achieving the same ranking with the expert ranking. As shown in Figure 2, when $\hat{S}>100$, the ranking results of the four chatbots are the same with the expert in 95% cases and guarantee a stable ranking. We also do more in-depth analysis. The ranking stability depends on the significance of ranking. Table 2 shows the consistency scores of DialoGPT and DialoFlow are close. We applied a leave-one-out stability analysis, in which we drop one chatbot. Figure 2 shows that when leaving one between DialoGPT or DialoFlow out, the stability is achieved with $\hat{S} = 50$ dialogues.

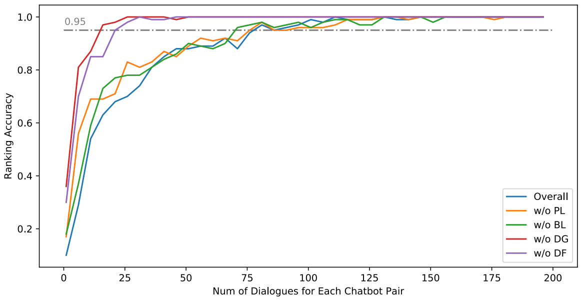

Figure 2: Ranking stability experiments. The x-axis denotes the number of conversations for each chatbot pair. The y-axis denotes the rate achieving the same ranking with the experts.

Summary. The number of conversations needed for a stable evaluation in AIH framework is dependent on the chatbots to be tested, and more conversations usually lead to more stable evaluation. In general cases, 75 conversations are enough to get a valid contradiction detection.

6 Further Investigation

In this section, we will further discuss the effectiveness of three parts in our framework containing question generation, contradiction detection, and human annotation evaluation.

6.1 Question Generation

Since a suitable question is necessary for the inquiry stage under our AIH framework, we make an in-depth analysis about the characters of question generation during inquiry stage.

Number of Questions and Contradictions
We randomly sample 200 dialogues for each chatbot pair and make statistics on the average number of the inquiry pairs and contradictions per conversation. As shown in Table 5, there are 4.57 inquiry pairs per conversation on average. There are 6.37 and 5.10 inquiry pairs per conversation when the Blender acts as Chatbot2 and serves as Chatbot1, respectively, which are both highest among all chatbots. The number of inquiry pairs reveals that the Blender can chat more about persona and facts, and the DialoGPT mentions these things less. Table 5 also shows the number of contradictions per conversation. Similarly, the Blender makes the most contradictions and is the most likely to induce the chatbot interacting with it to redeclare facts or opin-

Number of Questions					
	BL	**PL**	**DG**	**DF**	**Avg.**
BL	6.54	6.13	2.62	5.12	**5.10**
PL	6.54	5.34	1.98	4.36	4.55
DG	6.25	4.45	1.67	3.79	4.04
DF	6.15	5.79	2.25	4.21	4.60
Avg.	**6.37**	5.42	2.13	4.37	4.57

Number of Contradictions ($\tau = 0.15$)					
	BL	**PL**	**DG**	**DF**	**Avg.**
BL	2.61	1.28	1.61	1.50	**1.74**
PL	2.82	1.40	0.58	1.56	1.53
DG	2.66	1.12	0.57	1.31	1.38
DF	2.63	1.53	0.77	1.56	1.61
Avg.	**2.73**	1.38	0.69	1.57	1.56

Table 5: Statistic of average number of inquiry pairs and the contradictions per conversation for each chatbot pair. The column name and the row name represent Chatbot1 and Chatbot2 respectively.

Question Appropriateness					
	BL	**PL**	**DG**	**DF**	**Avg.**
BL	0.932	0.960	0.922	0.936	0.938
PL	0.942	0.976	0.940	0.948	0.951
DG	0.784	0.870	0.928	0.882	0.866
DF	0.867	0.934	0.922	0.939	0.915
Avg.	0.881	0.935	0.947	0.942	0.927

Table 6: The appropriateness of the generated questions (Human evaluation).

ions that are likely to be contradictory.

Question Appropriateness

We analyze the appropriateness of the generated questions. We randomly sample 50 dialogues from each chatbot pair and ask human annotators to decide if the generated questions are appropriate based on the provided context (0/1). As shown in Table 6, the overall appropriateness score is about 0.93, which reveals that our question generation strategy is simple yet highly effective. We further study the wrong questions and find that most of them can be attributed to that the general question generation model can not work well in the dialogue context. We leave the better question generation task in open-domain dialogue for future work.

	CR	**F1**	r
$\tau = 0.1$	**0.401**	0.650	0.430
$\tau = 0.15$	0.364	**0.655**	**0.436**
$\tau = 0.3$	0.287	0.606	0.423
$\tau = 0.5$	0.235	0.572	0.421

Table 7: The analysis of threshold τ. CR means contradiction rate. r denotes the Pearson correlation. Pearson correlation and F1 score are measured with human annotations.

Inter-Annotator Agreement					
	BL	**PL**	**DG**	**DF**	**Avg.**
AIH	0.818	0.817	0.812	0.807	0.814

Table 8: We analyse the inter-annotator agreement of the human evaluation in our proposed AIH framework. The correlation is measured by correlating each annotation with overall decision.

6.2 Effect of Contradiction Threshold τ

We evaluate the effect of hyper-parameter τ in Equ.2, and the results are reported in Table 7. We compute the F1 score and Pearson correlation between the automatic evaluation results and the human annotations under different τ. We can make the following observations: (i) When $\tau = 0.15$, the Pearson correlation and F1 score reaches the highest. Thus we choose $\tau = 0.15$ in our main experiments. (ii) The highest F1 score is 0.655, and the highest Pearson correlation is 0.436, which is a moderate correlation. The observations reveal that there is a big gap between automatic evaluation and human evaluation, though the contradiction rate is similar. We consider that it is because the NLI model we employ is trained on the general domain rather than the dialogue domain, so there are lots of reference problems that can not deal with well.

6.3 Inter-Annotator Agreement

To investigate the quality of human annotation, we compute the inter-annotator agreements, i.e., the correlation between each annotation and the overall decision is measured. The Pearson correlation for each chatbot is shown in Table 8. The inter-annotator agreement is high for all chatbots, suggesting that the evaluation instructions are well-understood by the annotators.

7 Conclusion and Future Work

In this work, we introduced the *Addressing Inquiries about History* (AIH), an effective and practical framework for open-domain chatbot consistency evaluation. AIH works by inserting questions about the mentioned facts and opinions in the history into the bot-bot conversation and employing human annotators or neural models to evaluate whether the responses are consistent or not. Based on this, we can rank different chatbots accurately and efficiently. We show that our framework can effectively evaluate the consistency of chatbots and the evaluation results well correlate with experts. Also, our framework is cost and time-efficient and can not only give an overall consistency score but also provide exactly the contractions, which can accelerate the evolution process of chatbots.

As in this work, we only focus on the contradictions about entities, and future work can improve the inquirer module and explore more kinds of contradictions. Besides, future work should also develop a more effective contradiction recognition module in the dialogue domain, while in this work we just exploit the general Natural Language Inference model to detect contradictions. The non-consistency problem is serious in current open-domain chatbots. We hope our work could facilitate and provide guidelines for future work on developing self-consistent open-domain chatbots.

Acknowledgement

We sincerely thank the anonymous reviewers for their thorough reviewing and valuable suggestions. This work is supported by National Key R&D Program of China (NO. 2018AAA0102502).

Ethic Statement

For the human evaluation and human interaction included in our paper, we state the ethical impact here. In Expert Evaluation, we recruited three expert volunteers from our lab, who had 2-3 years experience of the dialogue system development. They were asked to chat with the chatbots for $>$20 times before the formal evaluation. In Human-bot Natural Interaction, we collected human-bot conversations by inviting student volunteers from our university through the web interface. The participants were instructed to chat with the chatbots naturally. They were asked to sign the agreement before the dialogue: (i) They were at least 18 years old and could type into the web interface to chat with the chatbots. (ii) They were told that their interaction would generate anonymous text data which would only be used for the research on dialogue systems. In Human Annotation, we employed three well-educated professional annotators from a commercial data annotating company to annotate the contradictions for bot-bot conversations and human-bot natural conversations. We paid the company reasonable salary. The company provided comfortable working conditions and fair salaries for the annotators.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020a. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977.

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020b. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Siqi Bao, Huang He, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, Zhen Guo, Zhibin Liu, and Xinchao Xu. 2020. PLATO-2: towards building an open-domain chatbot via curriculum learning. *CoRR*, abs/2006.16779.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media, ICWSM 2020, Held Virtually, Original Venue: Atlanta, Georgia, USA, June 8-11, 2020*, pages 830–839. AAAI Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jan Deriu, Don Tuggener, Pius von Däniken, Jon Ander Campos, Álvaro Rodrigo, Thiziri Belkacem, Aitor Soroa, Eneko Agirre, and Mark Cieliebak. 2020. Spot the bot: A robust and efficient framework for the evaluation of conversational dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3971–3984. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference* *of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2020. The second conversational intelligence challenge (convai2). In *The NeurIPS'18 Competition*, pages 187–208. Springer.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

Chulaka Gunasekara, Seokhwan Kim, Luis Fernando D'Haro, Abhinav Rastogi, Yun-Nung Chen, Mihail Eric, Behnam Hedayatnia, Karthik Gopalakrishnan, Yang Liu, Chao-Wei Huang, et al. 2021. Overview of the ninth dialog system technology challenge: Dstc9. *Proceedings of the 9th Dialog System Technology Challenge Workshop in AAAI2021*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: improved dialogue evaluation with optimized questions and multi-turn comparisons. *CoRR*, abs/1909.03087.

Zekang Li, Zongjia Li, Jinchao Zhang, Yang Feng, and Jie Zhou. 2021a. Wechat ai's submission for dstc9 interactive dialogue evaluation track. *Proceedings of the 9th Dialog System Technology Challenge Workshop in AAAI2021*.

Zekang Li, Jinchao Zhang, Zhengcong Fei, Yang Feng, and Jie Zhou. 2021b. Conversations are not flat: Modeling the intrinsic information flow between dialogue utterances. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Shikib Mehri and Maxine Eskénazi. 2020. Unsupervised evaluation of interactive dialog with dialogpt. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2020, 1st virtual meeting, July 1-3, 2020*, pages 225–235. Association for Computational Linguistics.

Yixin Nie, Mary Williamson, Mohit Bansal, Douwe Kiela, and Jason Weston. 2020. I like fish, especially dolphins: Addressing contradictions in dialogue modeling. *CoRR*, abs/2012.13391.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5370–5381. Association for Computational Linguistics.

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2021–2030. Association for Computational Linguistics.

Haoyu Song, Yan Wang, Wei-Nan Zhang, Zhengyu Zhao, Ting Liu, and Xiaojiang Liu. 2020. Profile consistency identification for open-domain dialogue agents. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6651–6662. Association for Computational Linguistics.

Artit Wangperawong. 2020. Text2text: generate questions and summaries for your texts. `https://github.com/artitw/text2text`.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3731–3741. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 270–278. Association for Computational Linguistics.

Investigating the Reordering Capability in CTC-based Non-Autoregressive End-to-End Speech Translation

Shun-Po Chuang* **Yung-Sung Chuang*** **Chih-Chiang Chang*** **Hung-yi Lee**
National Taiwan University, Taipei, Taiwan
`{f04942141,b05901033,r09922057,hungyilee}@ntu.edu.tw`

Abstract

We study the possibilities of building a non-autoregressive speech-to-text translation model using connectionist temporal classification (CTC), and use CTC-based automatic speech recognition as an auxiliary task to improve the performance. CTC's success on translation is counter-intuitive due to its monotonicity assumption, so we analyze its reordering capability. Kendall's tau distance is introduced as the quantitative metric, and gradient-based visualization provides an intuitive way to take a closer look into the model. Our analysis shows that transformer encoders have the ability to change the word order and points out the future research direction that worth being explored more on non-autoregressive speech translation.[1]

1 Introduction

Recently, there are more and more research works focusing on end-to-end speech translation (ST) (Bérard et al., 2016; Weiss et al., 2017; Bérard et al., 2018; Vila et al., 2018; Di Gangi et al., 2019; Ran et al., 2019; Chuang et al., 2020). Instead of cascading machine translation (MT) models to an automatic speech recognition (ASR) system, end-to-end models can skip the error bottleneck caused by ASR and be more computationally efficient. However, in the inference time, an autoregressive (AR) decoder is needed to decode the output sequence, causing the latency issue.

In MT, non-autoregressive (NAR) models have been heavily explored recently (Gu et al., 2018; Lee et al., 2018; Ghazvininejad et al., 2019; Stern et al., 2019; Gu et al., 2019; Saharia et al., 2020) by leveraging the parallel nature of transformer (Vaswani et al., 2017). In contrast, such kind of models is rarely explored in the field of speech translation, except for a concurrent work (Inaguma et al., 2020). In this work, we use connectionist temporal classification (CTC) (Graves et al., 2006) to train NAR models for ST, without an explicit decoder module. Our entire model is merely a transformer encoder. Multitask learning (Anastasopoulos and Chiang, 2018; Kano et al., 2021) on ASR, which is often used in speech translation, can also be applied in our transformer encoder architecture to further push the performance. We achieve initial results on NAR speech translation by using a single speech encoder.

CTC's success on the translation task is counter-intuitive because of its monotonicity assumption. Previous works directly adopt the CTC loss on NAR translation without further verification on the reordering capability of CTC (Libovický and Helcl, 2018; Saharia et al., 2020; Inaguma et al., 2020). To further understand the reason that the CTC-based model can achieve ST task, we analyze the ordering capabilities of ST models by leveraging Kendall's tau distance (Birch and Osborne, 2011; Kendall, 1938), and a gradient-based visualization is introduced to provide additional evidence. To the best of our knowledge, this is the first time to examine the ordering capabilities on the ST task.

We found that after applying multitask training, our model can have more tendency to re-arrange the positions of the target words to better positions that are not aligned with audio inputs. We highlight that our contribution is to 1) take the first step on translating pure speech signal to target language text in a NAR end-to-end manner and 2) take a closer look at the reason that NAR model with CTC loss can achieve non-monotonic mapping.

2 Approaches

2.1 CTC-based NAR-ST Model

We adopt transformer architecture for non-autoregressive speech-to-text translation (NAR-ST). The NAR-ST model consists of convolutional

---

*Contributed equally.

[1] The source code is available. See Appendix A.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1068–1077
August 1–6, 2021. ©2021 Association for Computational Linguistics

layers and self-attention layers. The audio sequence X is downsampled by convolutional layers, and self-attention layers will generate the final translation token sequence Y based on the downsampled acoustic features. We use CTC loss as the objective function to optimize the NAR-ST model. The CTC decoding algorithm allows the model to generate translation in a single step.

CTC predicts an alignment between the input audio sequence X and the output target sequence Y by considering the probability distribution marginalized over all possible alignments.

The CTC loss function is defined as:

$$\mathcal{L}_{\text{CTC}} = -\sum_{(x,y)\in D} \log p_\theta(y|x), \tag{1}$$

where x is audio frame sequence, y is target sequence, and D is the training set. CTC uses dynamic programming to marginalize out the latent alignments to compute the log-likelihood:

$$\log p_\theta(y|x) = \log \sum_{a\in\beta(y)} \prod_t p(a_t|x;\theta) \tag{2}$$

where $a = \{a_t\}_{t=0}^{|x|}$ is an alignment between x and y and is allowed to include a special "blank" token that should be removed when converting a to the target sequence y. $\beta^{-1}(a)$ is a collapsing function such that $\beta^{-1}(a) = y$ if $a \in \beta(y)$.

CTC has a strong conditional independence and monotonicity assumption. It means that the tokens in Y can be generated independently, and there exists a monotonic alignment between X and Y. The monotonicity property is suitable for tasks such as ASR. However, in translation tasks, there is no guarantee that the output sequence should follow the assumption, as word orders differ in different languages. In this work, we want to examine whether the powerful self-attention based transformer model can overcome this problem to some degree or not.

2.2 CTC-based Multitask NAR-ST Model

Multitask learning improves data efficiency and performance across various tasks (Zhang and Yang, 2017). In AR end-to-end ST, multitask learning technique is often applied using ASR as an auxiliary task (Anastasopoulos and Chiang, 2018; Sperber and Paulik, 2020). It requires an ASR decoder in addition to the ST decoder to learn to predict transcriptions while sharing the encoder.

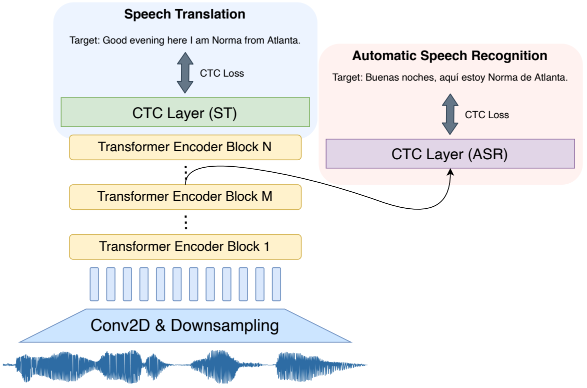

Figure 1: Multitask CTC model for ST and ASR.

To perform multitask learning on NAR-ST model, we propose to apply CTC-based ASR on a single M-th layer in the model, as illustrated in Figure 1. It helps the NAR-ST model capture more information with a single CTC layer in an end-to-end manner. And the ASR output will not be involved in the translation decoding process.

2.3 Reordering Evaluation – Kendall's Tau Distance

We measure reordering degree by Kendall's tau distance (Kendall, 1938). LRscore (Birch and Osborne, 2011) also introduced the distance with consideration of lexical correctness. Different from LRscore, we purely analyze the reordering capability rather than lexical correctness in this work.

Given a sentence triplet (T, H, Y), where $T = \langle t_1, ..., t_{|T|}\rangle$ is the audio transcription. $H = \langle h_1, ..., h_{|H|}\rangle$ and $Y = \langle y_1, ..., y_{|Y|}\rangle$ are hypothesis and reference translation, respectively. An external aligner provides two alignments: $\boldsymbol{\pi} = \langle \pi(1), ..., \pi(|T|)\rangle$ and $\boldsymbol{\sigma} = \langle \sigma(1), ..., \sigma(|T|)\rangle$. $\boldsymbol{\pi}$ maps each source token t_k to a reference token $y_{\pi(k)}$, and $\boldsymbol{\sigma}$ maps t_k to a hypothesis token $h_{\sigma(k)}$. We follow the simplifications proposed in LRscore to reduce the alignments to a bijective relationship. The proportion of disagreements between $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$ is:

$$\mathrm{R}(\boldsymbol{\pi}, \boldsymbol{\sigma}) = \frac{\sum_{i=1}^{|T|} \sum_{j=1}^{|T|} z_{ij}}{|T|(|T|-1)/2}$$

$$\text{where } z_{ij} = \begin{cases} 1 & \text{if } \pi(i) < \pi(j) \text{ and } \sigma(i) > \sigma(j) \\ 0 & \text{otherwise} \end{cases}$$

Then, we define the term **reordering correctness** R_{acc} by introducing the brevity penalty (BP):

$$\mathrm{R}_{acc} = (1 - \sqrt{\mathrm{R}(\boldsymbol{\pi}, \boldsymbol{\sigma})}) * BP$$

$$\text{where } BP = e^{1-|Y|/|H|} \text{ if } |H| \leq |Y| \text{ else } 1$$

Method	fisher_dev	fisher_dev2	fisher_test	CH_devtest	CH_evltest	Speed-Up
(A) *Autoregressive Models*						
(a) Transformer (*b*=10)	38.90	39.78	38.92	12.93	13.05	× 1.0
(b) Transformer + MTL (*b*=10)	46.38	47.54	46.28	**17.66**	17.17	× 1.0
(c) Transformer + MTL + ASR init. (*b*=10)	**48.27**	**49.17**	**48.40**	17.26	17.45	× 1.0
(d) Transformer + MTL + ASR init. (*b*=5)	48.18	48.91	48.21	17.34	**17.55**	× 2.0
(e) Transformer + MTL + ASR init. (*b*=1)	46.05	47.04	46.14	16.48	16.33	× 8.5
(B) *Non-Autoregressive Models (Ours)*						
(f) CTC	42.61	43.91	43.50	13.02	13.52	× 28.9
(g) CTC + MTL at 4-th layer	42.26	43.70	43.58	13.10	13.17	
(h) CTC + MTL at 6-th layer	42.06	44.05	43.56	13.19	13.38	
(i) CTC + MTL at 8-th layer	**44.45**	**45.23**	**44.92**	**14.20**	**14.19**	
(j) CTC + MTL at 10-th layer	42.86	44.18	43.59	13.65	13.28	

Table 1: BLEU on Fisher Spanish dataset and CALLHOME (CH) dataset, including autoregressive and non-autoregressive models. The abbreviation *b* stands for the beam size for beam search decoding. Multitask learning (MTL) represents using ASR as the auxiliary task trained with ST. In *Autoregressive Models*, the auxiliary loss always applied on the final encoder output in MTL, and we applied it on different layers in NAR models.

The higher the value, the more similar between two given alignments. Ideally, a well-trained model could handle the reordering problem by making $\boldsymbol{\sigma}$ close to $\boldsymbol{\pi}$ and result in $R_{acc} = 1$.

3 Experiments

3.1 Experimental Setup

We use the ESPnet toolkit (Watanabe et al., 2018) for experiments. We perform Spanish speech to English text translation with Fisher Spanish corpus. The test sets of CALLHOME corpus are also included for evaluation. The dataset details and download links are listed in Appendix B.

The NAR-ST model consists of two convolutional layers followed by twelve transformer encoder layers. Knowledge distillation (Kim and Rush, 2016) is also applied. More training details and parameter settings can be found in Appendix C.

3.2 Translation Quality and Speed

We use BLEU (Papineni et al., 2002) to evaluate the translation quality, as shown in Table 1. Beam-search decoding with beam-size *b* is considered for the AR models in this experiments. Greedy decoding is always used for the NAR models.

In the results of AR models (part (A)), multitask learning (MTL) can get better performance compared to the model without jointly training with an auxiliary task (row (b) v.s (a)). Further improvement can be brought by using a pre-trained ASR encoder as the initialization weight (row (c) v.s (b)). It shows that using ASR data for MTL and initialization are the essential steps to achieve exceptional performance. The performance drops when beam-size decreases, which shows a trade-off between the decoding speed and the performance (row (c) v.s (d)(e)).

To better optimize the decoding speed, NAR-ST provide a great solution to reach a shorter decoding time (part (B)). NAR-ST models is ×28.9 faster than the AR model with beam-size 10 (part (B) v.s rows (a)-(c)) and ×3.4 faster than the AR model with greedy decoding (part (B) v.s row (e)). We initialize the NAR-ST models with the weight pre-trained on ASR task and applied the proposed MTL approach on different intermediate layers (rows (g)-(j)). As the results showed in part (B), applying MTL on the higher layers improves the performance (rows (i)(j) v.s (f)). It shows that speech signal needs more layers to model the complexity, and sheds light on selecting the intermediate layer to apply MTL is essential. We also evaluate the ASR results of the MTL models in Appendix D.

Some text-based refinement approaches can further improve the translation quality (Libovickỳ and Helcl, 2018; Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Chan et al., 2020). We leave it as the future work and focus on analyzing the reordering capability of the CTC-based model.

3.3 Word Order Analysis

In this section, we discuss the word ordering problem in the translation task. We use R_{acc} defined in section 2.3 to systematically evaluate the reordering correctness across the corpora. Besides, we examine the gradient norm in models to visualize the reordering process.

Quantitative Analysis We use SimAlign (Sabet et al., 2020) to align the transcriptions and translations with details in Appendix E. Table 2 shows

		R_{acc}		
	Method	**dev**	**dev2**	**test**
(a)	random permutaion	41.37	41.84	42.42
	Autoregress Models			
(b)	Tr. (b=10)	77.79	79.16	79.09
(c)	Tr. + MTL (b=10)	79.32	80.18	80.30
	Non-Autoregress Models (Ours)			
(d)	CTC	71.69	74.00	74.57
(e)	CTC+MTL@8	71.91	74.35	74.96

Table 2: The reordering correctness evaluated on Fisher Spanish dataset. The values are selected as the best among 4 references. Tr. stands for Transformer. All models are initialized with pretrained ASR.

R_{acc} evaluated on ST models. We included the correctness of random permutation as a baseline.

The AR models obtain high R_{acc} scores (rows (b)(c)), it shows that the AR model can handle a complex word order. The NAR models also have the ability to rearrange words (rows (d)(e) v.s. (a)) but are weaker than AR models due to the independent assumption brought by CTC. An interesting observation is that applying MTL tends to improve R_{acc} (rows (c)(e) v.s. (b)(d)). We conclude that the monotonic natural in ASR improves the stability in training ST (Sperber et al., 2019).

To investigate the relation between model performance and the reordering difficulty, we measure the reordering difficulty by $\mathrm{R}_{\boldsymbol{\pi}} = \mathrm{R}(\boldsymbol{\pi}, \boldsymbol{m})$, where $\boldsymbol{m} = \langle 1, ..., |T| \rangle$ is a dummy monotonic alignment. We split all the testing data (dev/dev2/test) into smaller equal-size groups by different reference $\mathrm{R}_{\boldsymbol{\pi}}$. The BLEU scores for these groups were plotted in Figure 2. Obviously, AR models are more robust to higher reordering difficulty. Nonetheless, we observed that when MTL is applied at layer 8, CTC model is more robust to reordering difficulty, in some cases ($\mathrm{R}_{\boldsymbol{\pi}}$<0.07) even come close to the AR model without ASR pretraining.

Gradient-based Visualization We consider the gradient norm as an approximated indicator of reordering in our model. For each output token h_i, we concatenate the relative influence on it across all layers, which yields a matrix $\mathbf{O}^i \in \mathbb{R}^{|X| \times L}$, where each row is a frame and each column is a layer. We refer to this as the reordering matrix for token h_i. We leave the computational details in Appendix H.

Figure 3 shows a reordering matrix for token `_thing`. We can observe that the single-task CTC model (Figure 3, right) tends to keep focusing on the same position of the output token `_thing` at the lower layers. In contrast, the model with MTL (Figure 3, left) does not focus on the same position at the lower layers. It can focus more on the other position where the corresponding speech segment appears, and aggregate the information to the output position at the higher layers. Additional examples can be found in Appendix I.

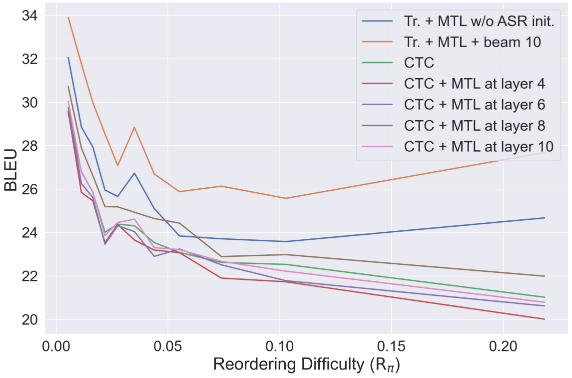

Figure 2: The BLEU score curve under different reordering difficulties (R_π). Details are in Appendix G.

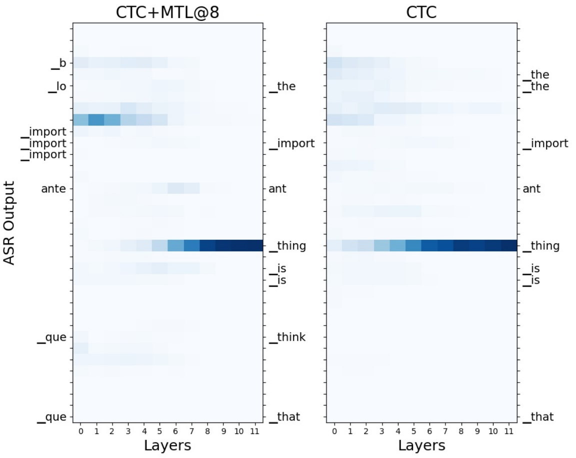

Figure 3: The re-ordering matrix of the token "thing". The ASR output from MTL model is included as a proxy for word positioning in source speech.

4 Concluding Remarks

We propose a CTC-based NAR-ST model with an auxiliary CTC-based ASR task and are the first to study the reordering capability in CTC-based NAR-ST model. R_{acc} is adopted to analyze reordering in the ST task, and gradient-based visualizations reveal the internal manipulation of the models. Besides trying to improve BLEU scores, we encourage future research on NAR models to also evaluate whether the NAR models have inferior reordering capabilities in order to close the gap between AR and NAR models.

Broader Impact and Ethical Considerations

We believe that our work can help researchers in the NLP community understand more about the non-autoregressive speech translation models, and we envision that the model proposed in this paper will equip the researchers with a new technique to perform better and faster speech translation. We do not see ourselves violating the code of ethics of ACL-IJCNLP 2021.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *arXiv preprint arXiv:1612.01744*.

Alexandra Birch and Miles Osborne. 2011. Reordering metrics for mt. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1027–1035.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. *arXiv preprint arXiv:2002.08926*.

Shun-Po Chuang, Tzu-Wei Sung, Alexander H. Liu, and Hung-yi Lee. 2020. Worse WER, but better BLEU? leveraging word embedding as intermediate in multitask end-to-end speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5998–6003, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Mattia A Di Gangi, Matteo Negri, and Marco Turchi. 2019. Adapting transformer to end-to-end spoken language translation. *Proc. Interspeech 2019*, pages 1133–1137.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11181–11191.

Hirofumi Inaguma, Yosuke Higuchi, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. 2020. Orthros: Non-autoregressive end-to-end speech translation with dual-decoder. *arXiv preprint arXiv:2010.13047*.

Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. 2021. Transformer-based direct speech-to-speech translation with transcoder. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 958–965.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Jindřich Libovickỳ and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *Proc. Interspeech 2019*, pages 2613–2617.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*.

Masoud Jalili Sabet, Philipp Dufter, and Hinrich Schütze. 2020. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *arXiv preprint arXiv:2004.08728*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2019. Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation. *Transactions of the Association for Computational Linguistics (TACL)*.

Matthias Sperber and Matthias Paulik. 2020. Speech translation and the end-to-end promise: Taking stock of where we are. *arXiv preprint arXiv:2004.06358*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Laura-Cross Vila, Carlos Escolano, José AR Fonollosa, and Marta-R Costa-Jussà. 2018. End-to-end speech translation with the transformer. *Proc. IberSPEECH 2018*, pages 60–63.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson-Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al. 2018. Espnet: End-to-end speech processing toolkit. *Proc. Interspeech 2018*, pages 2207–2211.

Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. *arXiv preprint arXiv:1703.08581*.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

A Source Code

Please download our code at https://github.com/voidism/NAR-ST and follow the instructions written in `README.md` to reproduce the results.

B Dataset Details

We use the Fisher and CALLHOME Spanish dataset (a Spanish-to-English speech-to-text translation dataset), which can be downloaded in the following links: 1) Fisher Spanish Speech https://catalog.ldc.upenn.edu/LDC2010S01
2) CALLHOME Spanish Speech https://catalog.ldc.upenn.edu/LDC96S35
3) Fisher and CALLHOME Spanish–English Speech Translation https://catalog.ldc.upenn.edu/LDC2014T23.

B.1 Statistics

The data statistics are listed in Table 3.

Data Split	# of Utterance	Duration (hours)
Training sets		
fisher_train	138792	171.61
Validation sets		
fisher_dev	3973	4.59
fisher_dev2	3957	4.70
callhome_devtest	3956	3.82
Testing sets		
fisher_test	3638	4.48
callhome_evltest	1825	1.83

Table 3: The data statistics of the Fisher and CALLHOME Spanish dataset.

B.2 Preprocessing

We use ESPnet to preprocess our data. For text, we use Byte Pair Encoding (BPE) (Sennrich et al., 2016) with vocabulary size 8,000. We convert all text to lowercase with punctuation removal. For audio, we convert all audio files into wav file with a sample frequency of 16,000. We extract 80-dim fbank without delta. We use SpecAugment (Park et al., 2019) to augment our data. More details can be found in our source code in Appendix A.

C Training Details

C.1 Computing Infrastructure and Runtime

We use a single NVIDIA TITAN RTX (24G) for each experiment. The average runtime of experiments in Table 1 is 2-3 days for both autoregressive and non-autoregressive models.

C.2 Hyperparameters

Our training hyperparameters are listed in Table 4. We do not conduct hyperparameter search, but follow the autoregressive ST best setting in ESPnet toolkit (Watanabe et al., 2018), and use the same hyperparameter for our non-autoregressive models. Due to the limited budget, we run each experiment for once. For inference stage of CTC-based models, we simply use greedy decode to produce the output sequences.

Hyperparameter	Value
encoder layers	12
hidden units	2048
attention dimension	256
attention heads	4
label smoothing weight	0.1
batch size	64
optimizer	noam
learning rate	2.5
warmup steps	25000
attention dropout rate	0.0
gradient accumulate step	2
gradient clipping	5.0
epoch	30
dropout rate	0.1

Table 4: The main hyperparameters in the experiment.

C.3 Knowledge Distillation

To perform sequence-level knowledge distillation (Seq-KD) (Kim and Rush, 2016) to improve the performance of NAR models, we firstly trained an autoregressive transformer-based MT model on the transcriptions and translations in same training set with ESPnet. Then we used the trained model to produce the hypotheses with beam search size of 1 for the whole training set. We swapped the ground truth sequences with the hypotheses for all NAR ST model training. We also show the ablation results on knowledge distillation in Table 5.

We also try the possibility of using the autoregressive ST model to produce the hypotheses for Seq-KD, but the results are not as good as using a MT model. The results are shown in the second row in Table 5. The download links to MT/ST decode results for conducting Seq-KD are also provided in the `README.md` file in our source code (See Appendix A).

Method	Fisher			CALLHOME	
	dev	dev2	test	devtest	evltest
NAR w/ MT Seq-KD	**44.45**	**45.23**	**44.92**	**14.20**	**14.19**
NAR w/ ST Seq-KD	41.08	41.28	40.85	13.63	13.27
NAR w/o Seq-KD	41.59	43.13	42.25	13.05	12.81

Table 5: Ablation test on sequence-level knowledge distillation (Seq-KD) on Fisher Spanish dataset and CALLHOME (CH) dataset. The NAR model we used is CTC+MTL at 8-th layer.

C.4 Model Selection

When evaluating the models, we average the model checkpoints of the final 10 epochs to obtain our final model for NAR experiments. For AR experiments, we follow the original setting in ESPnet to average the 5 best-validated model checkpoints to obtain the final model.

C.5 Model Size

The number of parameters of our CTC model is 18.2M. The number of parameters of the autoregressive model is 27.9M.

D Automatic Speech Recognition Evaluation

Method	Fisher		
	dev	dev2	test
CTC+MTL@4	47.84	46.79	45.97
CTC+MTL@6	40.21	39.02	38.08
CTC+MTL@8	32.87	31.99	30.88
CTC+MTL@10	**29.31**	**28.36**	**27.10**

Table 6: The Word Error Rate (WER) on Fisher Spanish dataset of CTC results of the intermediate layers in multitask CTC models.

We compute the Word Error Rate (WER) for ASR output obtained from the intermediate ASR branch of our proposed models. The results are shown in Table 6. We can observe that when applying multitask learning in the higher layers, the WER becomes lower. It indicates that ASR need more layers to perform better. However, the best ST scores are achieved by CTC+MTL@8 instead of CTC+MTL@10. It may be caused by the fact that there are only two transformer encoder layers for CTC+MTL@10 to perform ST. It may be too difficult for the model to convert the information from source language to target language in two encoder layers, even though the lower WER indicates useful information is provided to perform ST.

E SimAlign setup

We use the released code[2] by the authors of SimAlign as the external aligner to obtain word alignments used for calculating reordering metrics. SimAlign uses contextualized embeddings from pretrained language models, and there are several proposed algorithms to do word alignments. We use XLM-R (Conneau et al., 2019) as the underlying contextualized word embeddings with the itermax matching algorithm.

F Reordering Difficulty

We provide reordering difficulty measured on all en-xx language pairs in CoVoST2 dataset in Table 7.

en-xx	R_π	en-xx	R_π
de	5.92	tr	15.80
zh-CN	10.63	fa	13.93
ja	20.86	sv-SE	2.94
ar	6.04	mn	16.28
et	5.98	cy	5.78
ca	4.39	id	3.41
sl	5.13	ta	15.57
lv	5.34		

Table 7: The training set reordering difficulty evaluated on each pairs of languages in the CoVoST2 dataset.

G Details on Figure 2

In Figure 2, the primary goal is to view the relation between reordering difficulty and the model's performance. We describe the method used in (Birch and Osborne, 2011) to represent the reordering difficulty as follows: For each example in the fisher dev and test set, calculate Kendall's tau distance between 1) its reference alignment (alignment between source transcription and reference translation) and 2) a dummy monotonic alignment, which is just the sequence 1...m. Intuitively this shows how much the reference alignment disagrees with monotonic alignment, and hence the reordering difficulty. Next, we divide all examples into 10 bins, where each bin contains examples with similar reordering difficulty, and all bins have an equal number of examples. Finally, we calculate the BLEU score of the hypotheses in each bin. The result is plotted in Figure 2.

---

[2] `https://github.com/cisnlp/simalign`

H Gradient-based Visualization

We first obtain a saliency matrix $\mathbf{J}^M \in \mathbb{R}^{|X| \times |X|}$ for the M-th transformer layer by computing the gradient norm of output logits w.r.t. the latent representations of each timestep in that layer. An example is shown in Figure 4. Then, we normalize $\mathbf{J}^M$ across the dimension corresponding to the source audio sequence. Intuitively, the i-th column of $\mathbf{J}^M$ can be interpreted as the relative influence of the representations at each position on the i-th output token.

Consequently, we proceed to re-arrange $\mathbf{J}^M$ in the following way: for each output token h_i, we concatenate the relative influence on it across all layers, which yields the reordering matrix for token h_i, denoted as $\mathbf{O}^i \in \mathbb{R}^{|X| \times L}$.

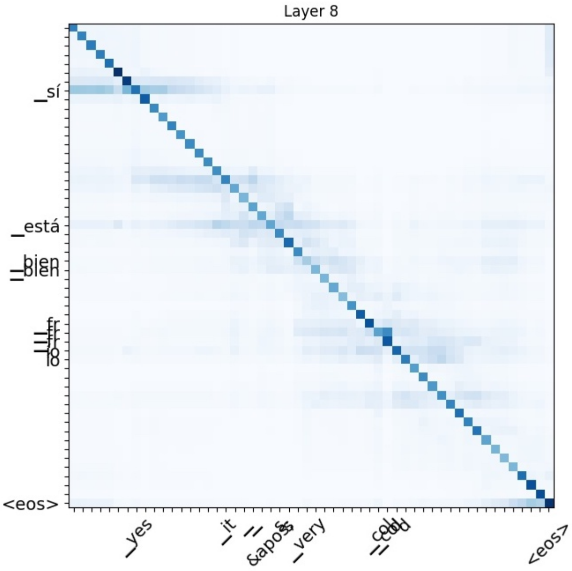

Figure 4: The saliency matrix of layer 8 input.

I Reordering Matrix

We provide some additional examples of visible reordering in our CTC-based models. In Figure 5, "_`we`" is heavily influenced by the position of audio signal "amos", even though the ASR output is incorrectly predicted as "as". In Figure 6, "_`you`" also influenced by audio signal "usted". It is interesting to observe that in some cases the pure CTC-based model appears more capable of reordering, while in others it does not.

J Higher Reordering Difficulty

We address instances of higher difficulty by analyzing Figure 7. In the figure, the horizontal axis corresponds to the reordering difficulty, and the vertical axis corresponds to the reordering correctness. In this figure, there is a very consistent decrease in reordering correctness when reordering difficulty increases, and the rate of decrease is very similar between NAR and AR models. This observation reveals that when evaluated on distant language pairs, that the reordering difficulty is large, the gap between NAR and AR will probably remain roughly the same. We will conduct experiments on different language pairs to verify the above claim in future work.

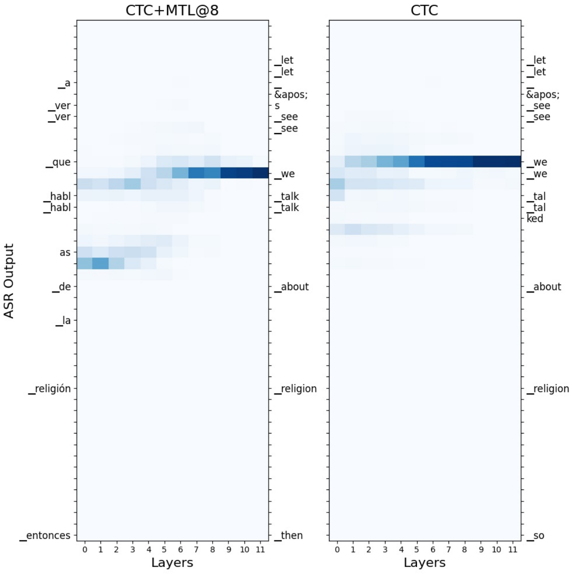

Figure 5: The re-ordering matrix of the token "_`we`".

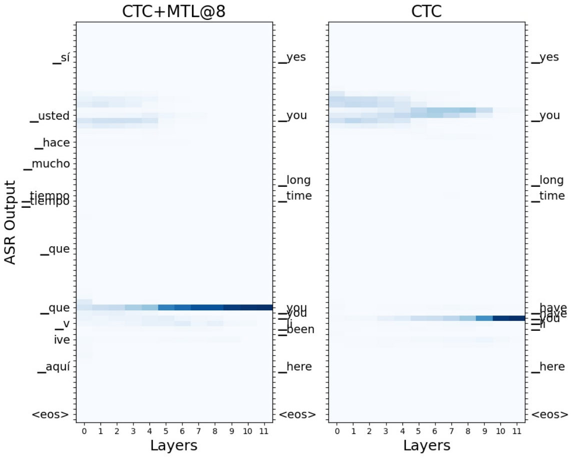

Figure 6: The re-ordering matrix of the token "_`you`".

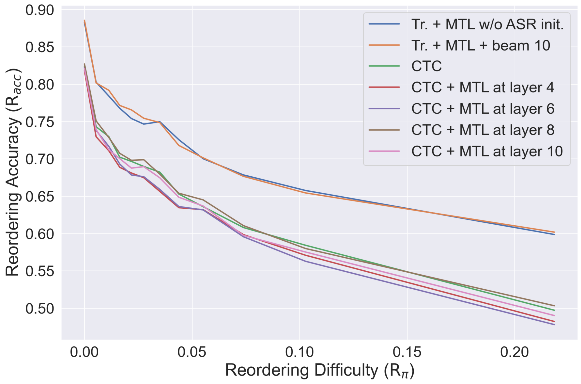

Figure 7: The reordering accuracy (R_{acc}) curve under different reordering difficulties (R_π).

Code Summarization with Structure-induced Transformer

Hongqiu Wu[1,2,3], **Hai Zhao**[1,2,3,*], **Min Zhang**[4]
[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University,
Shanghai, China
[4] Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, Suzhou, China
wuhongqiu@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn, minzhang@suda.edu.cn

Abstract

Code summarization (CS) is becoming a promising area in recent language understanding, which aims to generate sensible human language automatically for programming language in the format of source code, serving in the most convenience of programmer developing. It is well known that programming languages are highly structured. Thus previous works attempt to apply structure-based traversal (SBT) or non-sequential models like Tree-LSTM and graph neural network (GNN) to learn structural program semantics. However, it is surprising that incorporating SBT into advanced encoder like Transformer instead of LSTM has been shown no performance gain, which lets GNN become the only rest means modeling such necessary structural clue in source code. To release such inconvenience, we propose structure-induced Transformer, which encodes sequential code inputs with multi-view structural clues in terms of a newly-proposed structure-induced self-attention mechanism. Extensive experiments show that our proposed structure-induced Transformer helps achieve new state-of-the-art results on benchmarks.

1 Introduction

By 2020, software development and maintenance become an indispensable part of human work and life. Various assistant technical measures have been taken to facilitate more enjoyable software development, among which it is especially welcomed by programmers when there is a code summarization task generating sensible human language annotations automatically.

Code (Java)

```
private void attachPlot (SVGPlot newplot) {
    this.plot = newplot;
    if (newplot == null) {
        super.setSVGDocument(null);
        return;
    }
    newplot.synchronizeWith(synchronizer);
    super.setSVGDocument(
        newplot.getDocument());
    super.setDisableInteractions(
        newplot.getDisableInteractions());
}
```

Summ. Attach to a new plot and display.

Code (Python)

```
def get_change_lines_in_file_for_tag(tag,
                                     change_dict):
    cleaned_lines = []
    data_list = change_dict.get('data', [])
    for data_dict in data_list:
        block = data_dict.get('block', '')
        lines = block.split('\\n')
            for line in lines:
                index = line.find(tag)
                if (index >(-1)):
                    line = line[index:]
                    cleaned_lines.append(line)
    return cleaned_lines
```

Summ. The received change_dict is the jsonified version of the changes to a file in a changeset being pushed to the Tool Shed from the command line. This method cleans and returns appropriate lines for inspection.

Table 1: Task samples of code summarization, where summ. refers to the output summary.

In early days, code summarization was a derivative problem of information retrieval (Haiduc et al., 2010; Eddy et al., 2013; Wong et al., 2013, 2015) by matching the most similar code snippets which are labeled with summaries. Such method lacks generalization and performs unsatisfactorily. Thus in recent years, researchers treated code summarization as a task of language generation (Iyer et al., 2016; Liang and Zhu, 2018), which usually depends on RNN-based Seq2Seq models (Cho et al., 2014; Bahdanau et al., 2015).

It is already known that RNN-based models may encounter bottleneck when modeling long

---

[*] Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), Key Projects of National Natural Science Foundation of China (U1836222 and 61733011), Huawei-SJTU long term AI project, Cutting-edge Machine Reading Comprehension and Language Model. This work was supported by Huawei Noah's Ark Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1078–1090
August 1–6, 2021.

	Structure-sensitive	Long-term dependency	Feat-model match
LSTM			
Tree-LSTM	✓		
Transformer		✓	
LSTM + SBT	✓		✓
Transformer + SBT	✓	✓	
SiT	✓	✓	✓

Table 2: Comparison of the previous models with proposed SiT model. The last column refers to whether input features match with the corresponding model.

sequences due to its poor long-term dependency. For instance, a normal snippet of Java as shown in Table 1 usually has hundreds of tokens. More recently, Ahmad et al. (2020) used an enhanced Transformer-based model to capture long-term and non-sequential information of source code, which outperformed previous RNN-based models by a large margin.

On the other hand, in the light of the structural nature of programming languages, structure clues are supposed to greatly enhance programming language processing task like code summarization (Fernandes et al., 2019). Indeed, substantial empirical studies showed that Abstract Syntax Tree may help models better comprehend code snippets and achieve more sensible generation results. Previous approaches could be divided into two categories. The first is to employ non-sequential encoders (e.g., TBCNN (Mou et al., 2016), Tree-LSTM (Shido et al., 2019), Tree-Transformer (Harer et al., 2019), Graph Neural Network (Allamanis et al., 2018; Liu et al., 2020; Alex et al., 2020; Wang et al., 2021)) to directly model structural inputs. The other is to pre-process structural inputs to apply sequential models on them. Uri et al. (2019) used LSTM to encode code structure by sampling possible paths of AST. Another similar work is structure-based traversal (SBT) (Hu et al., 2018a), which manages to flatten ASTs into linear sequences.

Though existing studies achieve success on the concerned code summarization task more or less, there is still room in improving both of the above modeling approaches. It is well known RNN encoders like LSTM only have limited capabilities in capturing long-range dependencies in sequence, and GNN-like models may be too sensitive to local information, which casts a natural solution, what if incorporating SBT into the Transformer? However, it is surprising that SBT only works effectively with LSTM but not the Transformer according to Ahmad et al. (2020). We attribute this to the linear and nonlinear inconsistence between SBT and encoder forms. SBT enables sequential encoders to learn non-sequential relationship (such as syntax) still in a certain elaborate linear forms. RNN may be effectively enhanced by SBT right because of its sequential architecture through attention mechanism. Transformer learns features through self-attention network (SAN), nevertheless which acts more like a non-sequential process. Consequently, such sequential features are unsuitable for a non-sequential architecture to extract implicit structural information. We boldly call it Feature-Model Match problem in Table 2. In this paper, we thus design an improved Transformer variants, structure-induced Transformer (SiT) to alleviate such difficulty in terms of a structure-induced self-attention mechanism, so that the resulted model may enjoy both merits, capturing long-range dependencies and more global information. The proposed model design has been applied to benchmark datasets and helps achieve new state-of-the-art performance.

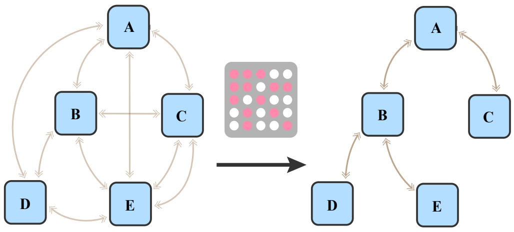

Figure 1: Use of adjacency matrix to transform original self-attention, left-hand complete graph, into structure-induced self-attention, right-hand graph which looks clear-cut. Note that we omit self-circles for concision.

2 Structure-based Code Summarization

The following sections present our code summarization method with two parts, in which the first is about structure representation of code, and the second is our proposed structure-induced Transformer.

2.1 Structure Representation of Code

Note that programming language like source code is subtle that certain different formats may result in different compilations. Thus pre-processing could be an great impact in code summarization.

We adopt Abstract Syntax Tree (AST) for representing the language grammar of source code as usual. Figure 2 depicts a typical AST, which is composed of terminal nodes and non-terminal nodes. A non-terminal node represents certain construction like *If* and *BinaryOp*, while terminal nodes represent its semantic components, such as identifiers and numbers.

In model implementation, we adopt adjacency matrix A to represent the AST instead of structure based traversal method as in Hu et al. (2018a), which represents tree structure in a sequential format. Such choice is well compatible with Transformer, which calculates attention weights by performing a dot-product of key-query pairs and results in an attention matrix of $l \times l$. We let l equal to number of AST nodes, then code summarization with Transformer becomes possible through applying a position-wise multiplication of A and original attention matrix.

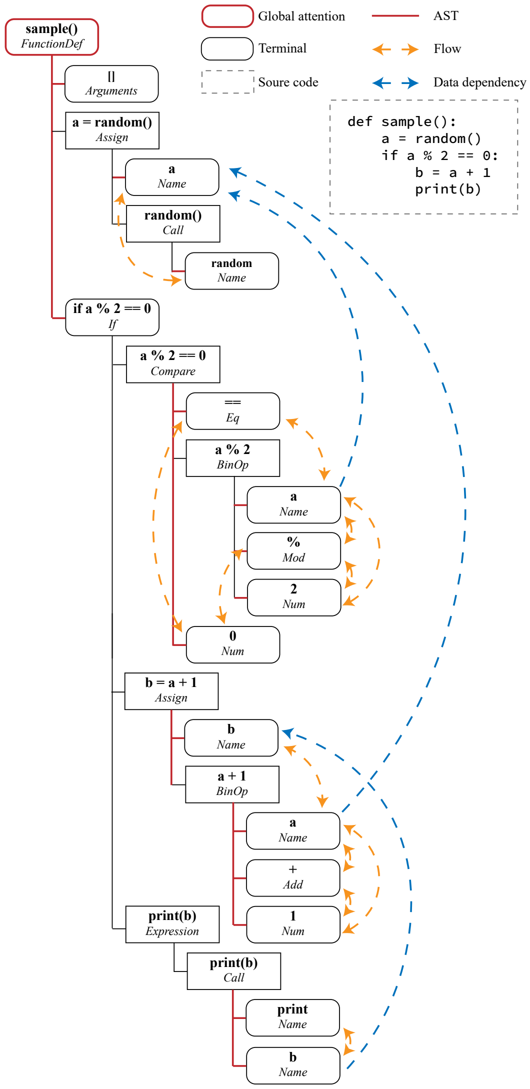

```
def sample():
    a = random()
    if a % 2 == 0:
        b = a + 1
        print(b)
```

Figure 2: A Python code sample of multi-view graph used in Si-SAN. The code snippet is referred from Liu et al. (2020), which is original in Java.

Inspired by Code Property Graph (CPG) (Yamaguchi et al., 2014; Liu et al., 2020), we further expand AST into a multi-view network (MVN or multi-view graph) (Sindhwani et al., 2005; Zhou and Burges, 2007; Kumar et al., 2011). An MVN is composed of multiple views, each view corresponding to a type of structural relationships while all views sharing the same set of vertexes (Shi et al., 2018). In this paper, we construct a three-view graph based on different code semantics, which are abstract syntax, control flow and data dependency. We show an example in Figure 2, where we use colorful strokes to describe different compositions in the graph. Note that we only utilize terminal nodes which are marked as rounded rectangles.

Specifically, we first generate an AST, on the basis of which we add additional edges to further represent the flow of control and data. For control flow, since Transformer is order-sensitive with position encoding, we only need to focus on each statement node. For instance, nodes b, $=$, a, $+$, 1 make a complete statement $b{=}a{+}1$. We connect each of them since they are in the same execution order. For data dependencies, we connect relevant data across the whole program, as the variable b in expression *print(b)* and assignment $b{=}a{+}1$ respectively, where the former is defined and loaded from the latter.

Now we may obtain three adjacency matrices of syntax, flow and dependency respectively, which are colored in red, yellow and blue in Figure 2. We combine them together and finally obtain a multi-view graph. Additionally, we add global attention on the root, which is allowed to attend to all tokens in the code, and all tokens in the code can attend to it. With aggregated structure, our structure-based code summarization is expected to capture various semantics of programs.

Note that our multi-view graph is different from CPG. which is original for C/C++ only and we do not find an appropriate analysis platform for other languages.

2.2 Structure-induced Transformer

Followed by appropriate structure representation and graph construction, we now propose our structure-induced Transformer (SiT) for code summarization, which is a structure-sensitive transformer (Zhang et al., 2020b; Narayan et al., 2020; Xu et al., 2020) model and is able to comprehend code snippets both semantically and syntac-

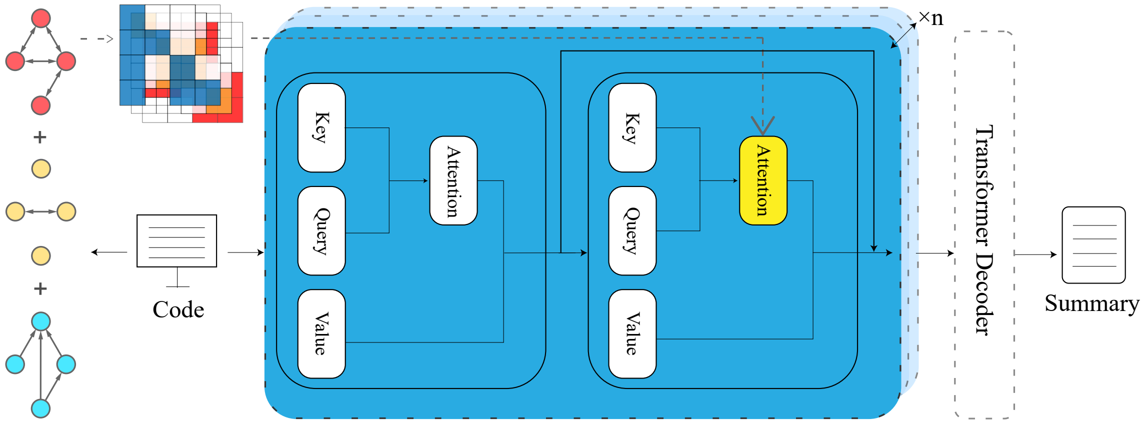

Figure 3: Overall architecture of Structure-induced Transformer (SiT).

tically. Meanwhile, we do not introduce extra parameters in SiT so that guarantee the training efficiency. In this section, we first review the self-attention network (SAN) of Transformer in terms of attention graph. Then we correspondingly propose structure-induced self-attention to build the structure-induced Transformer.

Vanilla Self-Attention Transformer is composed of stacks of identical layers for both encoder and decoder (Vaswani et al., 2017). Each layer emphasizes on self-attention mechanism, which is denoted as:

$$SAN(X) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where $X = (x_1, \ldots, x_l)$ denotes the input sequence of sub-words, l denotes the sequence length and d_k denotes the hidden size per head. Now we view each sub-word as a vertex n and inner product of each key-value pair as a directed edge e, the SAN can be described as a directed cyclic graph. Equation 1 can be rewritten as follow:

$$SAN(X) = E \cdot N \quad (2)$$

The attention scores $E = \{e_{ij}\}$ refers to a weight matrix of edges where e_{ij} represents how significant node n_i attend to node n_j, while value matrix $N = \{n_i\}$ refers to each node representation. Figure 1 depicts the process of calculating attention scores.

Note that SAN actually generates a fully connected cyclic graph without consideration of the very needed structure-aware representation for our concerned task.

Structure-induced Self-Attention To represent the needed structure information, we propose structure-induced self-attention network (Si-SAN).

Specifically, we introduce multi-view network into Equation 1, that is, multiply the adjacency matrix by key-query pairs:

$$SiSAN(X) = Softmax\left(\frac{A_{mv} \cdot QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

where A_{mv} refers to the multi-view representation of code.

Note that Si-SAN does not change the input code but appropriately incorporate code structure into SAN by changing its attention pattern. As shown in Figure 1, when $a_{ij} = 0$ in A_{mv}, the attention between n_i and n_j will be dropped out (Wu et al., 2021). We consequently obtain a more explicit attention graph. Different from calculating global information onto the whole sentence in original SAN, Si-SAN is expected to calculate structural information more accurately.

Structure-induced Module To enhance robustness and avoid over-pruning, we introduce structure-induced module, which is a stack of two layers, SAN and Si-SAN. In each module, SAN is followed by Si-SAN and the output is the combination of both layers. Specifically, given input sequence $X = (x_1, \ldots, x_l)$, where l denotes sequence length, we first pass it through an SAN layer to obtain hidden representation denoted as $H = (h_1, \ldots, h_l)$:

$$H = Concat(SAN_1(X), \ldots, SAN_h(X)) \quad (4)$$

where h refers to number of heads of multi-head attention while SAN_i refers to self-attention of

Model	Java			Python		
	BLEU	**ROUGE-L**	**METEOR**	**BLEU**	**ROUGE-L**	**METEOR**
CODE-NN (Iyer et al., 2016)	27.60	41.10	12.61	17.36	37.81	09.29
Tree2Seq (Eriguchi et al., 2016)	37.88	51.50	22.55	20.07	35.64	08.96
Hybrid2Seq (Wan et al., 2018)	38.22	51.91	22.75	19.28	39.34	09.75
DeepCom (Hu et al., 2018a)	39.75	52.67	23.06	20.78	37.35	09.98
API + Code (Hu et al., 2018b)	41.31	52.25	23.73	15.36	33.65	08.57
Dual Model (Wei et al., 2019)	42.39	53.61	25.77	21.80	39.45	11.14
Transformer (Ahmad et al., 2020)	44.58	54.76	26.43	32.52	46.73	19.77
Transformer* (Ahmad et al., 2020)	44.87	54.95	26.58	32.85	46.93	19.86
SiT	**45.76**(↑1.18)	**55.58**(↑0.82)	**27.58**(↑1.15)	**34.11**(↑1.59)	**48.35**(↑1.62)	**21.11**(↑1.34)
CodeBERT*† (Feng et al., 2020)	43.33	54.64	26.20	33.47	49.35	21.69
SiT on CodeBERT†	**45.19**(↑0.61)	**55.87**(↑1.11)	**27.52**(↑1.09)	**34.31**(↑1.79)	**49.71**(↑2.98)	**22.09**(↑2.32)

Table 3: BLEU, ROUGE-L and METEOR for our approach compared with other baselines. † refers to pre-trained models while * refers to models we rerun. The results of upper part are directly reported from Ahmad et al. (2020). Note that we only rerun Transformer and CodeBERT since they are much stronger than the other baselines. However, our results are even stronger. We show the ranges compared to the Transformer in Ahmad et al. (2020).

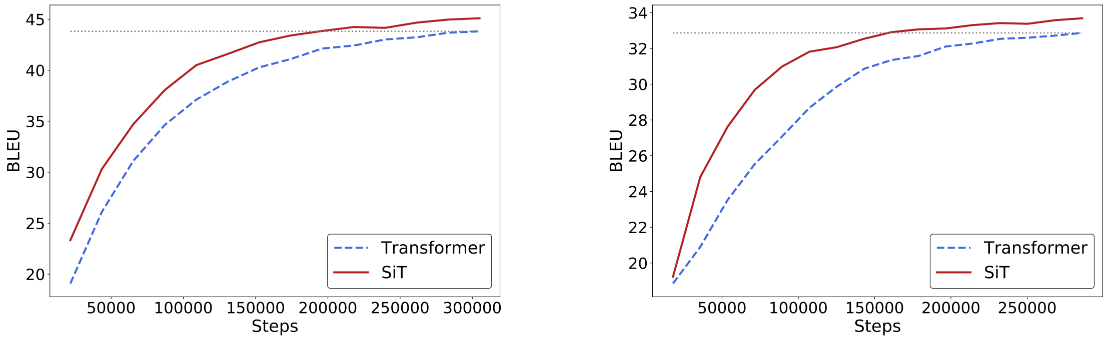

(a) BLEU score with training steps on Java

(b) BLEU score with training steps on Python

Figure 4: Convergence between Transformer and SiT.

head i. Subsequently, we pass H through a Si-SAN layer to obtain $H' = (h'_1, \ldots, h'_l)$:

$$H' = Concat(SiSAN_1(H), \ldots, SiSAN_h(H)) \tag{5}$$

Finally, we use an aggregation to fuse H and H' to obtain final representation $\bar{H} = (\bar{h}_1, \ldots, \bar{h}_l)$:

$$\bar{H} = Aggr(H, H') \tag{6}$$

where the aggregation we use is simple position-wise sum. We explore that the structure-induced module is more robust and leads to a better performance. In each stack, model begins to learn global information with SAN, where all connections are available. Subsequently, through Si-SAN, model is told which of the connections are useful and which should be shut down and thus avoiding over-pruning. Note that SiT with 3 stacks of structure-induced modules still consists of 6 encoder layers and 6 decoder layers, but only changes the architecture between modules of Transformer, not introducing any extra parameters.

Figure 3 depicts the overall architecture of SiT. Compared to original Transformer, our SiT with Si-SAN encodes a more accurate relative representation of code through pruning redundant connections.

2.3 SiT-based Code Summarization

Based on our structure-induced Transformer (SiT), now we specify our code summarization process.

We first transform the input code into adjacency matrices of multiple views and combine them through a weighted sum:

$$A_{mv} = \alpha A_{ast} + \beta A_{fl} + \gamma A_{dp} \tag{7}$$

where α, β, γ refer to the corresponding weight for each view. Then we pass code sequences and corresponding adjacency matrices into SiT encoder, which contains 3 Si-SAN layers. For decoder, we apply original Transformer decoder with cross attention. Finally, the summarization of the input code is generated through autoregressive decoding.

3 Experiments

3.1 Datasets and Pre-processing

Datasets Our experiments are conducted on two benchmarks of Java (Hu et al., 2018a) and Python (Wan et al., 2018), and for both we follow their training, test and development divisions.

Graph Construction For Java code, we refer to the method provided in (Hu et al., 2018a). They use *javalang* module of Python to compile Java and fetch AST in a dictionary form. For Python code, we generate trees by ourselves based on *ast* and *asttokens* modules. Finally, we write a script to resolve ASTs into multi-view adjacency matrices[1], where we let $\alpha = \beta = \gamma = 1$ for all experiments[2].

Out-Of-Vocabulary Code corpus in programming language may have a much bigger vocabulary than natural language, including vast operators and identifiers. We have to introduce vast out-of-vocabulary (OOV) tokens (usually replaced by $\langle$UNK$\rangle$) (Hu et al., 2018a) to keep it in a regular size. To avoid OVV problem, we apply *CamelCase* and *snake_case* tokenizers (Ahmad et al., 2020) to reduce code vocabulary and remove all extra nodes which do not correspond to specific tokens.

3.2 Baselines

We take all three categories of state-of-the-art models as our baselines for comparison.

Transformer We refer to the enhanced Transformer in (Ahmad et al., 2020) which equipped with copy attention (See et al., 2017) and relative position encoding (RPE) (Shaw et al., 2018). For fair enough comparison, we run their model on our machine under the same environment with SiT. Note that we also utilize RPE in SiT because of its better capability in capturing long sequences, while we do not utilize copy attention.

LSTM This group includes all relevant LSTM models with sequential and non-sequential inputs (Iyer et al., 2016; Eriguchi et al., 2016; Wan et al., 2018; Hu et al., 2018a,b; Wei et al., 2019).

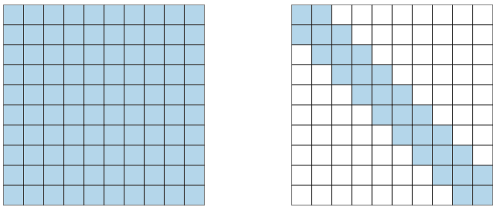

(a) Full (b) Window

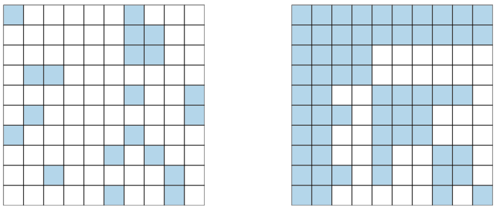

(c) Random (d) Structure-induced

Figure 5: Comparison of different types of self-attention pattern. (b) Window attention with $w = 2$. (c) Random attention with $r = 2$.

Pre-trained Language Model We also compare our model with CodeBERT (Feng et al., 2020), a pre-trained language model on both natural and programming languages. It is pre-trained over six programming languages with MLM (Devlin et al., 2019) and RTD (Clark et al., 2020).

3.3 Training Details

We train our model on a single nVidia Titan RTX with batch size in $\{32, 64\}$. The learning rate is in $\{$3e-5, 5e-5$\}$ with warm-up rate of 0.06 and L2 weight decay of 0.01. The maximum number of epochs is set to 150 for Transformer and 30 for CodeBERT. For validation, we simply use greedy search, while for evaluation, we use beam search with beam size in $\{4, 5, 8\}$ and choose the best result[3].

3.4 Main Results

Scores Table 3 shows the overall results on Java and Python benchmarks. The Transformer baseline is strong enough as it outperforms all the previous works by a significant margin. However, our model is more powerful, further boosting Transformer with more than 1 BLEU points on Java and Python respectively and achieves new state-of-the-art results. Specifically, SiT achieves higher scores

---

[1] https://github.com/gingasan/astruc

[2] We try to adjust the weights of three views, showing little performance variant, which suggests that self-attention network itself may balance the relative significance between the three.

[3] https://github.com/gingasan/sit3

on Python, increasing by 1.59, 1.62 and 1.34 points on BLEU, ROUGE-L and METEOR respectively. According to dataset statistics, Python contains 5 times more unique code tokens than Java, which makes it much more challenging. Thus the superiority of SiT on Python tends to be notable. Even so, SiT still boosts Transformer by 1.18, 0.82 and 1.15 points on BLEU, ROUGE-L and METEOR respectively on Java.

Convergence Moreover, Figure 4 shows the trend of BLEU scores on development set over training steps. SiT achieves a much faster convergence rate than Transformer. For instance on Python dataset, SiT arrives the best performance of Transformer in about 100 epochs, while the latter one still needs 50 more to finally achieve the optimal. Note that the running time of each epoch for both models is the same. Such high convergence rate helps showcase the necessity of Si-SAN.

Pre-training On the other hand, we can see that CodeBERT also achieves competitive results on both Java and Python. However, SiT is still more powerful on most metrics, which outperforms CoderBERT by 2.15, 0.95 and 1.15 points on BLEU, ROUGE-L and METEOR respectively on Java. However, CodeBERT performs much better on Python, which outperforms SiT by 1.00 and 0.58 points on ROUGE-L and METEOR. Note that CodeBERT is much bigger in size than Transformer and SiT (see Appendix A).

For further verification, we follow CodeBERT and conduct a RoBERTa-based (Liu et al., 2019) SiT to further fine-tune on both Java and Python. As shown in Table 3, pre-trained SiT obtains attractive results, further improving CodeBERT on all the metrics, which implies that our elaborate encoder design is still effective even under powerful pre-training assistance.

4 Ablation Study and Analysis

This section reports our ablation studies to valid our model on the dataset of Python-V2[4] (Barone and Sennrich, 2017), in which we conduct standard and unified pre-processing for strict fair comparison.

4.1 Si-SAN vs. SAN

To valid the effectiveness of Si-SAN, we gradually replace SAN layers in original Transformer with Si-SAN. Take Transformer model with Si-SAN proportion of 50% as an instance, we replace the second, fourth and last three encoder layers with Si-SAN and do not apply structure-induced module.

Model	Prop.	BLEU	ROUGE-L	METEOR
Transformer	0	47.42	57.28	29.62
Transformer	50%	49.64	59.39	31.16
Transformer	100%	49.80	59.38	31.30
SiT	50%	**50.04**	**59.56**	**31.46**

Table 4: BLEU, ROUGE-L and METEOR for variant models with incremental proportions of Si-SAN.

Model	Attn.	BLEU	ROUGE-L	METEOR
Transformer	Full	47.42	57.28	29.62
Transformer	Window	49.28	58.80	30.90
Transformer	Random	38.06	57.28	22.76
Transformer	Struc.	**49.80**	**59.38**	**31.30**

Table 5: BLEU, ROUGE-L and METEOR for variant models with different attention patterns.

The results of variant models with incremental proportions of Si-SAN layers are shown in Table 4. Intuitively, all of the Transformers obtain improvements when equipped with Si-SAN layers. We can also see that SiT outperforms Transformer with similar proportion of Si-SAN, which proves the effectiveness of structure-induced module. However, it is surprising that Transformer with all 6 layers of Si-SAN still outperforms original Transformer even if it may be over-pruned.

4.2 Si-SAN vs. Sparse SAN

To further valid our structure-based approach, we compare the performance of structure-induced attention with other sparse attention patterns, window attention in Longformer, ETC (Beltagy et al., 2020; Ainslie et al., 2020) and random attention in BigBird (Zaheer et al., 2020). We depict different attention patterns in Figure 5. The default sequence length in SiT is 400, and then we set both w and r to 64 in window and random attention respectively.

As shown in Table 5, Transformer with arbitrary sparse attention can not bring improvement as Si-

Model	BLEU	RE.-L	MTR.	SPEED
Transformer	44.87	54.95	26.58	1.0x
Transformer + SBT	43.34	53.97	25.02	1.5x
SiT-AST only	45.43	55.30	27.21	1.0x
SiT	**45.76**	**55.58**	**27.58**	**1.0x**

Table 6: Comparison of Si-SAN and SBT methods. Both methods only leverage AST information.

---

[4] https://github.com/EdinburghNLP/code-docstring-corpus/tree/master/V2

Model	Para.	BLEU	RE.L	MTR.
Transformer-8	140M	47.42	57.28	29.62
SiT-8	139M	50.04	59.56	31.46
Transformer-12	244M	50.11	59.47	31.44
SiT-12	242M	50.53	60.08	31.96
Transformer-16	370M	50.43	59.80	31.75
SiT-16	367M	**50.97**	**60.51**	**32.35**
Transformer-ALBERT enc.	124M	44.83	55.34	27.73
SiT-ALBERT enc.	124M	**49.31**	**58.46**	**30.83**

Table 7: BLEU, ROUGE-L and METEOR for variant models with different sizes, where RE.L and MTR. refer to ROUGE-L and METEOR respectively. Models like SiT-12 refers to SiT with 12 heads.

SAN, which refutes that SiT learns better through denoising. Specifically, random attention seriously deteriorates Transformer. It is surprising that window attention achieves a better result than Vanilla Transformer. Intuitively, tree structures like AST are highly localized. That is why window attention may show good performance. Nevertheless, Transformer with Si-SAN still outperforms window attention by 0.52 BLEU point.

4.3 Si-SAN vs. SBT

We reproduce SBT method on Java (Hu et al., 2018a) and apply it on our Transformer. For fair enough comparison, we let $\beta = \gamma = 0$ and conduct single-view SiT which only leverages AST information. As depicted in Figure 6, flattening ASTs into linear sequences does not result in improvement, which is consistent with Ahmad et al. (2020). However, we achieve substantial improvement while incorporating AST into Transformer using Si-SAN, which indicates our improved model design is indeed effective.

In addition, the average length of the input code will be much longer with SBT, which may introduce additional training cost. As shown in Figure 6, SiT is 1.5 times faster than Transformer with SBT.

4.4 Large Model

It is known that for nearly all deep models, increasing model size may cover quite much of model structure design improvement. Thus, it is possible that the improvement on base-size model may not work on large-size one. To valid this, we compare SiTs with Transformers under larger scale. As we can see pictorially in Table 7, with increasing parameter scale, SiTs with 12 heads and 16 heads both outperform the corresponding Transformers by 0.42 and 0.54 BLEU point respectively.

4.5 Parameter Sharing

Recently, parameter sharing on BERT (Devlin et al., 2019) has achieved promising results (Lan et al., 2020). Similar as ALBERT, we introduce cross-layer parameter sharing in both Transformer and SiT, sharing all parameters in all encoder layers. Note that we train our models from scratch and keep the decoder fixed.

As shown in Table 7, SiT performs much better on parameter sharing than Transformer does. We believe that code summarization task highly depends on structural information, and this is why SiT can still achieve good results with simply one group of encoder parameters while Transformer encounters a serious decline. On the other hand, it makes possible for lite model, which may balance high efficiency and performance.

5 Related Work

RNN-based Approaches While numbers of works (Haiduc et al., 2010; Eddy et al., 2013; Wong et al., 2013, 2015; Zhang et al., 2020a) on code summarization usually depended on information retrieval, most of the recent works tend to treat it as a machine translation problem. Meanwhile attention mechanism is broadly used for better performance on capturing long-range features. Allamanis et al. (2016) proposed a Convolution Neural Network (CNN) with copy attention, and more commonly, Iyer et al. (2016); Liang and Zhu (2018) proposed to use Recurrent Neural Network (RNN) with attention mechanism to summarize code snippets into natural language. Hu et al. (2018b) introduced API knowledge from related tasks while Cai et al. (2020) introduced type information to assist training, which also gained promising results. Additionally, reinforce learning (Wan et al., 2018) and dual learning (Wei et al., 2019; Ye et al., 2020) are also shown effective to boost model performance.

Transformer-based Approaches It is known that RNN-based models may encounter bottleneck when modeling long code sequences. Ahmad et al. (2020) proposed an enhanced Transformer with copy attention and relative position encoding while Gupta (2020); Dowdell and Zhang (2020) proposed to use Transformer (Vaswani et al., 2017) and Transformer-XL (Dai et al., 2019), all of which outperformed previous RNN-based models by a large margin.

Structure-based Approaches Recent works on code summarization pay more and more attention on structural information, which usually treats the source code in form of its Abstract Syntax Tree (AST). Hu et al. (2018a); LeClair et al. (2019); Uri et al. (2019) leveraged flattened ASTs as inputs and trained with LSTMs. Mou et al. (2016); Bui et al. (2021a); Shido et al. (2019); Harer et al. (2019) proposed TBCNN, TreeCaps, Tree-LSTM and Tree-Transformer to directly encode tree-style inputs. Differ from modeling code with sequential models, Allamanis et al. (2018); Liu et al. (2020); Alex et al. (2020) treated AST as graph and applied graph neural network, while Wang et al. (2021) applied heterogeneous graph neural network to model different types of nodes.

Pre-training Approaches Apart from training from scratch, CodeBERT (Feng et al., 2020) is pre-trained on vast bimodal corpora with masked language model (Devlin et al., 2019) and replaced token detection (Clark et al., 2020), and achieves powerful performances on downstream tasks. Nie et al. (2020) intensified contextualized code representation through masked code fragment predictions while Bui et al. (2021b) incorporated structural information using TBCNN. However, all of them do not include generation-related objectives. It is worth further exploration and practice on pre-training approaches for out concerned tasks.

6 Conclusion

This paper presents a novel structured-induced Transformer model on code summarization task. By well-designed architecture, the proposed model may effectively incorporate multi-view structure into attention mechanism without tricky implementation. We further adopt a new module architecture to aggregate both global self-attention and structure-induced self-attention representations. Experiments on two challenging benchmarks including Java and Python show that the proposed model yields new state-of-the-art results.

References

Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A transformer-based approach for source code summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4998–5007, Online. Association for Computational Linguistics.

Joshua Ainslie, Santiago Ontañón, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. 2020. ETC: encoding long and structured data in transformers. *CoRR*, abs/2004.08483.

LeClair Alex, Haque Sakib, Wu Lingfei, and McMillan Collin. 2020. Improved code summarization via a graph neural network. In *2020 IEEE/ACM International Conference on Program Comprehension*.

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to represent programs with graphs. In *International Conference on Learning Representations*.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2091–2100, New York, New York, USA. PMLR.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Antonio Valerio Miceli Barone and Rico Sennrich. 2017. A parallel corpus of python functions and documentation strings for automated code documentation and code generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 314–319. Asian Federation of Natural Language Processing.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Nghi D. Q. Bui, Yijun Yu, and Lingxiao Jiang. 2021a. Treecaps: Tree-based capsule networks for source code processing. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Nghi DQ Bui, Yijun Yu, and Lingxiao Jiang. 2021b. Infercode: Self-supervised learning of code representations by predicting subtrees. In *Proceedings of the 43rd International Conference on Software Engineering, ICSE 2021*.

Ruichu Cai, Zhihao Liang, Boyan Xu, Zijian Li, Yuexing Hao, and Yao Chen. 2020. TAG : Type auxiliary guiding for code comment generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 291–301. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Thomas Dowdell and Hongyu Zhang. 2020. Language modelling for source code with transformer-xl. *CoRR*, abs/2007.15813.

Brian P Eddy, Jeffrey A Robinson, Nicholas A Kraft, and Jeffrey C Carver. 2013. Evaluating source code summarization techniques: Replication and expansion. In *2013 21st International Conference on Program Comprehension (ICPC)*, pages 13–22. IEEE.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Structured neural summarization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Vivek Gupta. 2020. Deepsumm - deep code summaries using neural transformer architecture. *CoRR*, abs/2004.00998.

Sonia Haiduc, Jairo Aponte, Laura Moreno, and Andrian Marcus. 2010. On the use of automated text summarization techniques for summarizing source code. In *2010 17th Working Conference on Reverse Engineering*, pages 35–44. IEEE.

Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-transformer: A transformer-based method for correction of tree-structured data. *arXiv preprint arXiv:1908.00449*.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018a. Deep code comment generation. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 200–20010. IEEE.

Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018b. Summarizing source code with transferred api knowledge. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2269–2275. International Joint Conferences on Artificial Intelligence Organization.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Abhishek Kumar, Piyush Rai, and Hal Daumé III. 2011. Co-regularized multi-view spectral clustering. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 1413–1421.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Alexander LeClair, Siyuan Jiang, and Collin McMillan. 2019. A neural model for generating natural language summaries of program subroutines. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pages 795–806. IEEE / ACM.

Yuding Liang and Kenny Qili Zhu. 2018. Automatic generation of text descriptive comments for code blocks. In *Proceedings of the Thirty-Second*

AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5229–5236. AAAI Press.

Shangqing Liu, Yu Chen, Xiaofei Xie, Jing Kai Siow, and Yang Liu. 2020. Automatic code summarization via multi-dimensional semantic fusing in gnn. *arXiv preprint arXiv:2006.05405*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1287–1293. AAAI Press.

Shashi Narayan, Joshua Maynez, Jakub Adámek, Daniele Pighin, Blaz Bratanic, and Ryan T. McDonald. 2020. Stepwise extractive summarization and planning with structured transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4143–4159. Association for Computational Linguistics.

Lun Yiu Nie, Cuiyun Gao, Zhicong Zhong, Wai Lam, Yang Liu, and Zenglin Xu. 2020. Contextualized code representation learning for commit message generation. *CoRR*, abs/2007.06934.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Yu Shi, Fangqiu Han, Xinran He, Carl Yang, Jie Luo, and Jiawei Han. 2018. mvn2vec: Preservation and collaboration in multi-view network embedding. *CoRR*, abs/1801.06597.

Yusuke Shido, Yasuaki Kobayashi, Akihiro Yamamoto, Atsushi Miyamoto, and Tadayuki Matsumura. 2019. Automatic source code summarization with extended tree-lstm. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML workshop on learning with multiple views*, volume 2005, pages 74–79. Citeseer.

Alon Uri, Brody Shaked, Levy Omer, and Yahav Eran. 2019. code2seq: Generating sequences from structured representations of code. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 397–407.

Wenhan Wang, Kechi Zhang, Ge Li, and Zhi Jin. 2021. Learning to represent programs with heterogeneous graphs.

Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code generation as a dual task of code summarization. In *Advances in Neural Information Processing Systems*, pages 6563–6573.

Edmund Wong, Taiyue Liu, and Lin Tan. 2015. Clocom: Mining existing source code for automatic comment generation. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 380–389. IEEE.

Edmund Wong, Jinqiu Yang, and Lin Tan. 2013. Autocomment: Mining question and answer sites for automatic comment generation. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 562–567. IEEE.

Hongqiu Wu, Hai Zhao, and Min Zhang. 2021. Not all attention is all you need. *CoRR*, abs/2104.04692.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. Association for Computational Linguistics.

Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. 2014. Modeling and discovering vulnerabilities with code property graphs. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 590–604. IEEE Computer Society.

Wei Ye, Rui Xie, Jinglei Zhang, Tianxiang Hu, Xiaoyin Wang, and Shikun Zhang. 2020. Leveraging code generation to improve code retrieval and summarization via dual learning. In *WWW '20: The Web*

Conference 2020, Taipei, Taiwan, April 20-24, 2020, pages 2309–2319. ACM / IW3C2.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.

Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2020a. Retrieval-based neural source code summarization. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 1385–1397, New York, NY, USA. Association for Computing Machinery.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020b. Sg-net: Syntax-guided machine reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9636–9643. AAAI Press.

Dengyong Zhou and Christopher J. C. Burges. 2007. Spectral clustering and transductive learning with multiple views. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 1159–1166. ACM.

A Model Parameters

Model	d_h	d_{ff}	h	l
Transformer	64	2048	8	12
SiT	64	2048	8	12
Transformer-Window	64	2048	8	12
Transformer-Random	64	2048	8	12
Transformer-Struc.	64	2048	8	12
CodeBERT	64	3072	12	12
SiT on CodeBERT	64	3072	12	12
Transformer-ALBERT enc.	64	2048	8	12
SiT-ALBERT enc.	64	2048	8	12

Table 8: Model parameters in our experiments.

B Qualitative Samples

For qualitative analysis, we give some samples of code summarization with different models. We can see that SiT performs most precisely, while CodeBERT performs better than Transformer does.

Code

```java
private float computeOverscrollPercent () {
    if ( mOverScrollOffset >= _NUM ) {return mOverScrollOffset / mMaxOverScroll;}
    else {return mOverScrollOffset / mMaxUnderScroll;}
}
```

Summary	Gold: determine the current amount of overscroll . if the value is 0 , there is no overscroll . if the value is < 0 , tabs are overscrolling towards the top or or left . if the value is > 0 , tabs are overscrolling towards the bottom or right . SiT: determine the current amount of overscroll . if the value is 0 , there is no overscroll . if the value is < 0 , tabs are overscrolling towards the top or or left . if the value is > 0 , tabs are overscrolling towards the bottom or right . Transformer: determine the current amount of overscroll . if the value is 0 , there is no overscroll . if the value is < 0 , tabs are overscrolling towards the top or or left . if the value is > 0 , tabs are overscrolling towards the top or right . CodeBERT: determine the current amount of overscroll . if the value is 0 , there is no overscroll . if the value is < 0 , tabs are overscrolling towards the top or or left . if the value is > 0 , tabs are overscrolling towards the bottom or right .

Code

```java
public String peek () {
    String result = null;
    if (isEmpty()) {return null;}
    else {
        int cachedCurrentIndex = currentIndex;
        if (isEatingBlocksOfDelimiters) {trimStartingDelimiters();}
        int nearestDelimeter = -_NUM ;
        for (int i = _NUM; i <delimiters.length(); i++) {
            int delimiter = source.indexOf(delimiters.charAt(i), currentIndex);
            if (nearestDelimeter == -_NUM || delimiter != -_NUM && delimiter <nearestDelimeter) {
                nearestDelimeter = delimiter;
            }
        }
        if (nearestDelimeter == -_NUM) {result = source.substring(currentIndex);}
        else {result = source.substring(currentIndex, nearestDelimeter);}
        currentIndex = cachedCurrentIndex;
    }
    return result;
}
```

Summary	Gold: returns null if there is nothing left . SiT: returns null if there is nothing left . Transformer: finds the next unique identifier . CodeBERT: returns the index of the first delimited string removing from the current position .

Table 9: Qualitative samples of Java code summarization.

Code

```python
def _asFilesystemBytes(path, encoding=None):
    if (type(path) == bytes): return path
    else:
        if (encoding is None):
            encoding = sys.getfilesystemencoding()
            return path.encode(encoding)
```

Summ.	Gold: return cpath as a string of lbytes suitable for use on this systems filesystem . SiT: return cpath as a string of lunicode suitable for use on this systems filesystem . Transformer: convert a filesystem path of a byte string . CodeBERT: return a byte string suitable for use in cpath as a byte string .

Code

```python
def absent(name, DomainName, region=None, key=None, keyid=None, profile=None):
    ret = {’name’: DomainName, ’result’: True, ’comment’: ”, ’changes’: {}}
    r = __salt__[’boto_elasticsearch_domain.exists’](DomainName, region=region, key=key, keyid=keyid, profile=profile)
    if (’error’ in r):
        ret[’result’] = False
        ret[’comment’] = ’Failed to delete domain: {0}.’.format(r[’error’][’message’])
        return ret
    if (r and (not r[’exists’])):
        ret[’comment’] = ’Domain {0} does not exist.’.format(DomainName)
        return ret
    if __opts__[’test’]:
        ret[’comment’] = ’Domain {0} is set to be removed.’.format(DomainName)
        ret[’result’] = None
        return ret
    r = __salt__[’boto_elasticsearch_domain.delete’](DomainName, region=region, key=key, keyid=keyid, profile=profile)
    if (not r[’deleted’]):
        ret[’result’] = False
        ret[’comment’] = ’Failed to delete domain: {0}.’.format(r[’error’][’message’])
        return ret
    ret[’changes’][’old’] = {’domain’: DomainName}
    ret[’changes’][’new’] = {’domain’: None}
    ret[’comment’] = ’Domain {0} deleted.’.format(DomainName)
    return ret
```

Summ.	Gold: ensure domain with passed properties is absent . SiT: ensure domain with passed properties is absent . Transformer: ensure the iam role exists . CodeBERT: ensure the named domain is absent .

Table 10: Qualitative samples of Python code summarization.

Scheduled Dialog Policy Learning: An Automatic Curriculum Learning Framework for Task-oriented Dialog System

Sihong Liu[1]*, **Jinchao Zhang**[2], **Keqing He**[1], **Weiran Xu**[1]†, **Jie Zhou**[2]
[1] Beijing University of Posts and Telecommunications, China
[2] Pattern Recognition Center, WeChat AI, Tencent Inc, China
`{liusihong,kqin,xuweiran}@bupt.edu.cn`
`{dayerzhang,withtomzhou}@tencent.com`

Abstract

In reinforcement learning (RL) based task-oriented dialogue systems, users act as the environment and the agent learns the policy by interacting with users. However, due to the subjectivity of different users, the complexity of user-generated training conversations varies greatly, which leads to different difficulties for the agent to learn. Therefore, it is necessary for modeling dialogue complexity and make a reasonable learning schedule for efficiently training the agent. Towards that, we propose Scheduled Dialog Policy Learning, an automatic curriculum learning framework for jointing curriculum learning and policy optimization in the task-oriented dialog system. To our best knowledge, it is the first RL framework that improves dialogue policy learning by scheduling its learning process. Specifically, we introduce an automatic measurement to evaluate the dialogue complexity, and based on this automatic measurement, we train the dialog agent from easy dialogues to complex ones. Experiments demonstrate that our approach can be applied to the task-oriented dialogue policy learning and outperforms the previous state-of-the-art model, which increases 9.6% and 10.0% in the accuracy on the dialog success rate, respectively on the MultiWoz and Movie-Ticket Booking datasets.

1 Introduction

Dialog policy learning is an important component of the task-oriented dialogue system, and it determines the agent dialog action responding to the user. This learning process is often formulated as a reinforcement learning problem (Young et al., 2013; Levin et al., 1997; Dhingra et al., 2017; Li et al., 2017; Liu and Lane, 2017; Peng et al., 2018b; Su et al., 2018; Gao et al., 2019; Takanobu et al., 2020), where users act as the environment and the agent learns the policy by interacting with users. Thus, the learning performance of the dialogue policy depends much on users' behaviors.

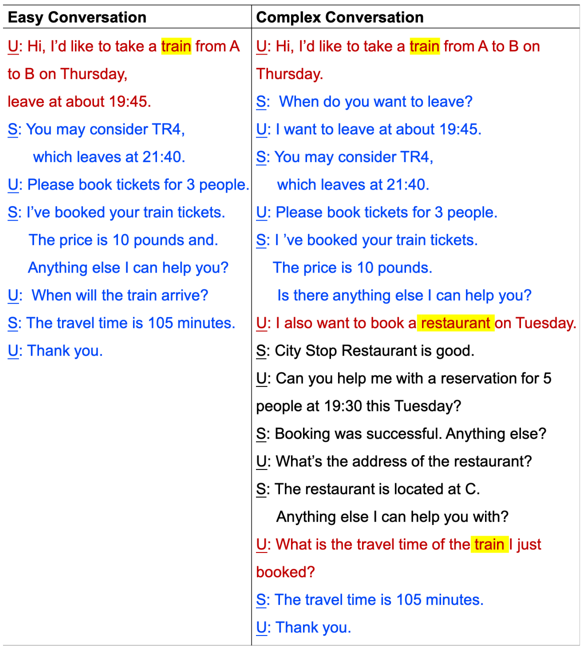

Easy Conversation

U: Hi, I'd like to take a train from A to B on Thursday,
leave at about 19:45.
S: You may consider TR4,
which leaves at 21:40.
U: Please book tickets for 3 people.
S: I've booked your train tickets.
The price is 10 pounds and.
Anything else I can help you?
U: When will the train arrive?
S: The travel time is 105 minutes.
U: Thank you.

Complex Conversation

U: Hi, I'd like to take a train from A to B on Thursday.
S: When do you want to leave?
U: I want to leave at about 19:45.
S: You may consider TR4,
which leaves at 21:40.
U: Please book tickets for 3 people.
S: I 've booked your train tickets.
The price is 10 pounds.
Is there anything else I can help you?
U: I also want to book a restaurant on Tuesday.
S: City Stop Restaurant is good.
U: Can you help me with a reservation for 5 people at 19:30 this Tuesday?
S: Booking was successful. Anything else?
U: What's the address of the restaurant?
S: The restaurant is located at C.
Anything else I can help you with?
U: What is the travel time of the train I just booked?
S: The travel time is 105 minutes.
U: Thank you.

Figure 1: The comparison of the easy conversation (left) and the complex conversation (right) which are task-oriented dialogs between the user (U) and the system (S) sampling from MultiWoz. Comparing to the left conversation, the right one has more turns, intents, slots, and also switches between two domains: *train* (marked as blue) and *restaurant* (marked as dark). The right instance is apparently more complex.

However, due to the subjectivity and open-ended nature of human conversations, the complexity of training dialogues with different users varies greatly (Lison and Bibauw, 2017). Figure 1 shows dialogues with different complexities from MultiWoz (Budzianowski et al., 2018) dataset. Comparing to the left instance, the conversation in the right column has more turns, intents, slots, and also has the switch between two domains: *train*

---

[*] This work was done when Sihong Liu was interning at Pattern Recognition Center, WeChat AI, Tencent Inc, China.

[†] Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1091–1102
August 1–6, 2021.

(marked as blue) and *restaurant* (marked as dark). The right instance is apparently more complex. As different complexity conversations lead to different difficulties for the agent to learn, in this paper, we introduce an automatic curriculum learning framework to improve dialog policy learning by scheduling its learning process where the agent first learns from easy conversations and then gradually manages more complicated ones. There is some related work that examines the curriculum learning in natural language processing (Kocmi and Bojar, 2017; Platanios et al., 2019; Sachan and Xing, 2016; Guo et al., 2019; Cai et al., 2020). Evaluation of training samples complexity is a challenging obstacle when organizing a curriculum. Previous work evaluates training samples mainly applying empirical and heuristic attributes, such as Platanios et al. (2019) defines the difficulty for training sentences concerning the sentence length and word rarity in the neural machine translation. Cai et al. (2020) evaluates the dialogue difficulty by five heuristic conversational attributes.

As the subjectivity and diversity of the conversation task, modeling its complexity by empirical and heuristic attributes is insufficient and relies on prior knowledge on the specific domain or dataset. Inspired by curiosity rewards (Schmidhuber, 1991; Stadie et al., 2015; Sorg et al., 2010; Pathak et al., 2017a), which applied state prediction error to predict uncertainty (Houthooft et al., 2016; Still and Precup, 2012; Wesselmann et al., 2019; Tegho et al., 2018) and improvement (Lopes et al., 2012) in RL tasks, in this paper, we further introduce an automatic measurement to evaluate dialogue complexity by estimating the dialog state differential space without applying any prior knowledge on the domain or dataset. Then we schedule the policy learning curriculum for the dialog agent based on the evaluation. To evaluate the effectiveness of our model for dialogue policy learning, we conduct our experiments on two public task-oriented dialog datasets: MultiWoz (Budzianowski et al., 2018) and Movie-Ticket Booking (Li et al., 2018). Experimental results show that our model reaches 83.3% and 57.0% in the accuracy for dialog successful rate on these two datasets, outperforming the previous state-of-the-art dialog model by 9.6% and 10.0% , respectively. Our main contributions in this work are three-fold:

- We design an automatic measurement to evaluate training dialogues complexity without any prior knowledge on the domain or dataset, which gets rid of empirical and heuristic attributes to model the dialogue complexity.
- Based on the automatic complexity measurement, we propose an automatic curriculum learning framework SDPL to improve the performance and learning efficiency of task-oriented dialogue policy learning.
- We conduct experiments on two task-oriented dialog corpus and results show the superiority of our model to the state-of-the-art baselines. Especially, it increases 9.6% and 10.0% in accuracy on dialog successful rate, respectively.

2 Related work

2.1 RL-based Task-oriented Dialog Policy Learning

Learning policies for the task-completion dialogue is often formulated as a reinforcement learning (RL) problem (Levin et al., 1997; Young et al., 2013; Li et al., 2017; Peng et al., 2018a,b; Su et al., 2018; Gao et al., 2019; Li et al., 2017; Peng et al., 2018b; Su et al., 2018; Takanobu et al., 2020; Li et al., 2020) and this learning process can be regard as a Markov Decision Process (MDP), where the agent interacts with a user through a sequence of actions to accomplish a pre-defined user goal.

Since reinforcement learning requires much interaction for training, a user simulator is often applied to interact with the agent providing the simulated user response in each dialogue turn based on a user goal (Li et al., 2017; Liu and Lane, 2017; Peng et al., 2018a; Su et al., 2018). Specifically, given a user goal G, at each time step t, the agent observes the current dialogue state s_t, and chooses an action a to execute, using the policy $\pi(a|s_t)$. Then the user responds the user action a_u which is sampled from the user goal G. Next, the agent receives reward r_t, observes the response a_u and updates to next state s_{t+1}. And the agent learns and updates the policy aiming to maximize its total discounted rewards. As the entire dialogue is around a user goal and a user goal corresponds to several user-generated dialogues, thus in this paper, we make the complexity evaluation of user goals for curriculum learning feeding different complexity dialogues generated by different complexity user goals to schedule the dialog agent training [1].

---

[1] Refer to the appendix for details on the user simulator

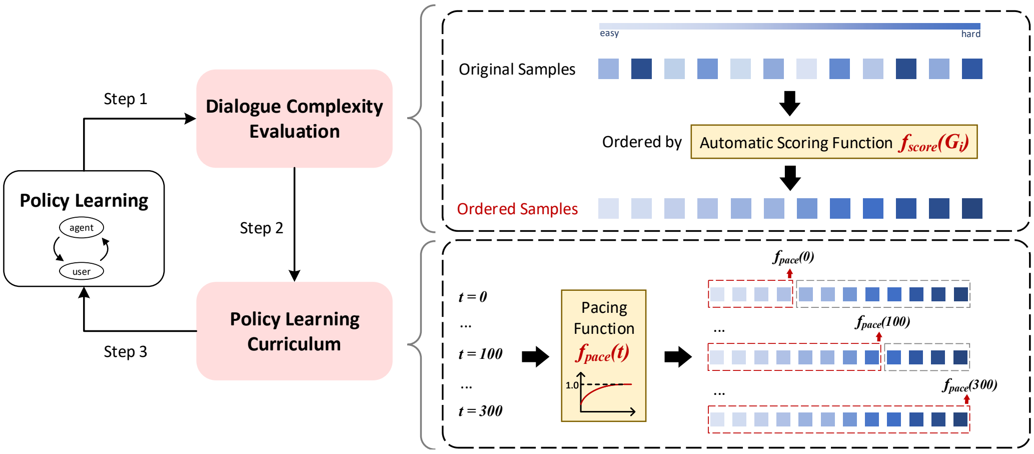

Figure 2: Illustration of proposed curriculum learning framework SDPL. It is defined by two stages. At **dialogue complexity evaluation stage**, an automatic scoring function $f_{score}(G_i)$ defines the instances G_i complexity (darker/lighter blue indicate easy/hard complexity) from easy to hard. Then at the **policy learning curriculum stage**, a pacing function $f_{pace}(t)$ schedules the percentage of training instances available for feeding to the policy learning according to the training step t. The dialog agent interacts with the user to learn the dialog policy on the scheduled instances and updates its learning capacity to execute a new dialog complexity evaluation.

2.2 Curriculum Learning

Curriculum learning (Bengio et al., 2009) is a learning strategy in machine learning, which learns from easy instances and then gradually handles harder ones. Curriculum learning has been used to natural language generation (Cai et al., 2020; Liu et al., 2018) and question answering task (Sachan and Xing, 2016). Cai et al. (2020) utilizes heuristic dialogue attributes to represent the dialogue complexity and propose several curricula, which achieve high performance. And Sachan and Xing (2016) proposes several heuristic strategies to model different complexity QA pairs. As the conversation task is more complex and analytic, in this paper, we propose an automatic curriculum framework to measure dialogue complexity automatically which can get rid of empirical and heuristic attributes and schedule different complexity training dialogues to improve the policy learning.

2.3 Curiosity Rewards

For many RL applications, state error prediction is used for curiosity rewards (Schmidhuber, 1991; Stadie et al., 2015; Sorg et al., 2010; Pathak et al., 2017a) [8, 9, 10, 5] to predict uncertainty (Houthooft et al., 2016; Still and Precup, 2012; Wesselmann et al., 2019; Tegho et al., 2018) and improvement (Lopes et al., 2012), improving the state exploration efficiency. And curiosity rewards are applied to tackle reward sparseness problem and state exploration in many tasks. Especially, Wesselmann et al. (2019) applied the curiosity rewards to the dialog policy to replace random exploration and stabilize training. While in this paper, we applied the curiosity reward as the measurement to evaluate the dialogue complexity.

3 Scheduled Dialog Policy Learning

We propose Scheduled Dialog Policy Learning (SDPL), a flexible and practical framework on joint curriculum learning and policy optimization for task-oriented dialog systems.

3.1 Overview

The overview of the full model is depicted in Figure 2, it schedules and designs the curriculum for dialogue policy learning in mainly two stages:

1) **Dialog Complexity Evaluation.** For the input training instances, we firstly measure original dialog instances complexity by an automatic scoring function, and arrange to sort these instances by their complexity scores, obtaining the ordered training instances. 2) **Policy Learning Curriculum.** After obtaining the ordered training instances, we apply a pacing function to schedule the percentage of instances available for training at each training step. Then these scheduled percentage of instances are fed to the dialogue agent to learn the dialog policy from easy to complex. As the training progressing, the agent updates its learning capacity and executes a new dialog complexity evaluation.

---

and the user goal.

In the subsequent subsection, we systematically describe how to utilize the SDPL for dialogue policy learning. In Section 3.2, we propose an automatic dialogue complexity evaluation approach. Then, we show in Section 3.3 how the automatic curriculum learning framework can contribute to the dialogue policy learning by automatically modeling dialogue complexity and making a reasonable learning schedule for training the dialog agent.

3.2 Dialogue Complexity Evaluation

The complexity of different training dialogues varies greatly which leads to different difficulties for the agent to learn. Therefore, it is necessary for modeling dialogue complexity and scheduling different complexity dialogues for efficiently training. In this paper, we propose a novel measurement to evaluate the dialogue complexity automatically. Inspired by Pathak et al. (2017b), for the agent instant learning capacity, we estimate the dialog state differential space that the agent can explore for current training samples as their training complexity. Less differential space means less error between the real state and predicted state which indicates current agent masters these samples where they are easy to learn for the current learning capacity.

Specifically, during the RL-based dialog policy learning process, in each step t, the agent observes current dialogue state s_t, then it executes the dialog action a_t estimating from current policy learning network. The agent then receives reward r_t [2] and updates to the next state s_{t+1}. Finally, the agent aims to maximize the expected sum of rewards which can be formulated as:

$$\underset{\theta_Q}{max} E_{\pi(s_t,a_t;\theta_Q)}[\sum_t r_t], \tag{1}$$

where $\pi(s_t, a_t; \theta_Q)$ is the policy learning network parameterized by θ_Q. To estimate the state differential space between the real state and predicted state, we first encode the current state and real next state into state feature vectors. Next, we apply a feature predicting neural network to obtain the predicted next state feature encoding which is formulated as:

$$\hat{\phi}(s_{t+1}) = g\Big(\phi(s_t), a_t; \theta_m\Big), \tag{2}$$

where $\phi(\cdot)$ is a state encoding network which transforms the one-hot state variables into a feature space suitable for learning. And $\hat{\phi}(s_{t+1})$ is the predicted next state feature of the real next state feature $\phi(s_{t+1})$. We optimize parameters θ_m by minimizing the mean square error between them:

$$m\Big(\phi(s_{t+1}), \hat{\phi}(s_{t+1})\Big) = \eta\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|^2, \tag{3}$$

where $\eta > 0$ is a scaling factor. The function $m(t)$ calculates feature encoding differentials of states between the real next state feature and estimated next state feature in the latent feature space, which represents the state differential space that agent needs explore. Less state differential space means less error between the real and predicted state feature which indicates that current agent masters these states are easy to learn for the current agent learning capacity. Therefore, we apply the $m(t)$ as the complexity evaluation for the current agent towards current training samples.

We jointly optimize the reinforcement learning process and complexity evaluation as:

$$\underset{\theta_Q}{max} E_{\pi(s_t,a_t;\theta_Q,\theta_m)}[\sum_t r_t + \sum_t m(t)]. \tag{4}$$

We further formulate the complexity evaluation into the task-oriented dialogue system to measure the dialogue complexity (lines 3-8 in Algorithm 1). In the dialogue system, the user and system interact with each other in a dialog session to fulfill the user goal G_i, where $G_i \in \mathcal{D}_{total}$ and $\mathcal{D}_{total}$ is a user goal set, i is the user goal index. Each user goal G_i corresponds to its K generated dialog sessions $\{\tau_1, ..., \tau_k\}$. And during the reinforcement learning process, each dialog session τ_k can be seen as a trajectory of state-action tuples $\{(s_t, a_t, r_t, s_{t+1}), ...\}$, $t \in \{0, 1, ..., N-1\}$, where N stands for the number of turns in each dialogue session. Therefore, based on the Eq. 3, for each dialog τ_k, we obtain its complexity score by an automatic scoring function which can be formulated as:

$$f_{score}(m_{\tau_k}) = \sum_{t=1}^{N} m_{\tau_k}(t). \tag{5}$$

During the RL-based dialog policy learning, as training dialogues are user-generated based on the user goal, we also evaluate the complexity of user goals for later policy curriculum learning. For a given user goal G_i, we obtain its complexity score

---

[2] In the RL based dialogue system, reward measures the degree of whether the dialogue is successful. In our experiment, the agent obtains a reward of 80 when it success, and obtained a reward of -40 when failed, and the agent receives the reward of -1 at each turn to encourage shorter dialogues.

Algorithm 1 Scheduled Dialogue Policy Learning

```
Require: Dialog user goal set $\mathcal{D}_{total}$ with each user goal $\{G_i\}$, $N$
Ensure: Dialog policy $\pi(s, a; \theta_Q, \theta_m)$
initialize $\pi(s, a; \theta_Q, \theta_m)$
for $n$=1:$N$ do
    # Dialogue Complexity Evaluation starts
    user starts dialogues $\tau_1 ... \tau_k$ based on $G_i$
    update the dialogue process by $\pi(s, a; \theta_Q, \theta_m)$
    store training tuples $\{(s_t, a_t, r_t, s_{t+1}), ...\}$ for each dialog $\tau_k$
    compute complexity score for $\tau_k$ and $G_i$ by $f_{score}$ in Sec. 3.2
    sort $\mathcal{D}_{total}$ with the complexity score to obtain $\mathcal{D}_{order}$
    # Policy Learning Curriculum starts
    sample batches from $\mathcal{D}_{order}$ by $f_{pace}$ in Eq. 7.
    user starts dialogues $\tau_1 ... \tau_k$ based on $G_i^{'}$
    agent updates $\pi(s, a; \theta_Q, \theta_m)$ by interacting with the user
end for
```

by calculating its generated dialog sessions average complexity evaluation which is formulated as:

$$f_{score}(G_i) = \frac{1}{K} \sum_{k=1}^{K} f_{score}(m_{\tau_k}). \qquad (6)$$

We sort the user goal set $\mathcal{D}_{total}$ by complexity scores, obtaining the forward complexity ordered user goal set $\mathcal{D}_{order}$. Based on the ordered training set, we further propose the policy learning curriculum to schedule the training of the agent.

3.3 Automatic Curriculum Learning Framework for Dialog Policy Learning

Our proposed automatic curriculum learning framework is shown as Algorithm 1. During the dialogue policy learning process, we first initialize the RL-based dialog policy as $\pi(s, a; \theta_Q, \theta_m)$. For each training step n, we start from the Dialogue Complexity Evaluation process. The user starts dialogues $\tau_1 ... \tau_k$ based on a user goal G_i. Then, we update the dialogue process based on the current policy, and store training tuples $\{(s_t, a_t, r_t, s_{t+1}), ...\}$ for each dialog τ_k. After that, we arrange to sort each user goal in the original training set from easy to complex according to the automatic scoring function f_{score} in Sec. 3.2, obtaining the ordered training set $\mathcal{D}_{order}$.

	Movie-Ticket Booking	MultiWoz
Dialogues	2,890	8,438
Intents	11	13
Slots	29	25
Avg. turns per dialogue	7.5	13.68
Domains	1	7

Table 1: The statistics of two public datasets, Movie-Ticket Booking and MultiWoz.

After obtaining the ordered training set, we start the policy learning curriculum process based on a pacing function $f_{pace}(t)$ aiming to schedule training instances that feeding to the policy learning network. A batch of training instances is sampled from the top $f_{pace}(t)$ percentage of the ordered training set. Following Platanios et al. (2019), we define the pacing function $f_{pace}(t)$ as:

$$f_{pace}(t) \triangleq min(1, \sqrt{t\frac{1 - c_0^2}{N} + c_0^2}) \qquad (7)$$

where $c_0 > 0$ is set to 0.01 and N is the duration of curriculum learning. At the early stage of the training process, the dialogue policy learning model learns from the instances drawing from the front (easy) part of the curriculum. With the advance of the curriculum, the complexity gradually increases, more complex training instances appear.

Based on sampled training instances by Eq. 7, the user starts dialogues from ordered sampling user goal $G_i^{'}$, while the agent interacts with the user to update and learn the dialogue policy $\pi(s, a; \theta_Q, \theta_m)$ from easy to complex, then conducts new dialogue complexity evaluation iteratively. Note that, although we describe the framework in two components for ease of understanding, in fact, the whole framework can be trained in an end-to-end manner. Thus, the proposed automatic curriculum learning framework is capable of not only scheduling the training samples to optimize the dialogue policy learning model, but also can be applied to other RL based training scenarios.

4 Experiments

4.1 Datasets

We conduct the experiments on two public datasets, Movie-Ticket Booking (Li et al., 2018) and MultiWoz (Budzianowski et al., 2018). Movie is a single domain dataset, and its goal is to build a dialogue system to help users find information about movies and book movie tickets. MultiWoz is a multi-domain, multi-intent task-oriented dialog corpus that contains seven domains, including Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train. A user may change his goal during the session, hence MultiWoz provides more complicated dialogs closer to real-world conversations. We show the full statistics in Table 1 [3].

---

[3] Refer to the appendix for details on datasets.

4.2 Evaluation Metrics

We adopt three metrics to evaluate the quality of policy learning: success rate, average turns, average reward. Success rate is the task completion rate – the fraction of dialogues that ended successfully. The dialog is successfully when all the requested information has been filled in, and the booked entities match all the indicated constraints given by the user goal. Average turns is the average length of the dialogue. Average reward is the average reward that agent received during the conversation.

4.3 Baselines

To verify the effectiveness of our proposed curriculum learning framework, we compare with different RL policy learning agents in task-completion dialogue as baselines:

DQN: A reinforcement learning based task-oriented dialogue model that the agent learned the dialogue policy by standard DQN (Li et al., 2017).

DDQ: The state-of-the-art task-oriented dialogue policy learning approach, Deep Dyna-Q (DDQ) (Peng et al., 2018b). DDQ integrates planning for dialogue policy learning to make the best use of limited real user experiences.

DDQ-CR: A DDQ agent only applied the curiosity reward into the dialogue policy learning but without curriculum learning (Wesselmann et al., 2019).

DDQ-CL-rule: A DDQ agent incorporating the curriculum learning (CL) and the dialogue complexity evaluation is rule-based, where empirically evaluating the dialogue by its slots number.

DDQ-SDPL: A DDQ agent that incorporating our proposed framework.

WDQN: A warm-DQN agent pre-trained by supervised learning (Lee et al., 2019).

WDQN-CR: A warm-DQN agent that only incorporating the curiosity rewards into dialogue policy learning without curriculum learning (Wesselmann et al., 2019).

WDQN-CL-rule: A warm-DQN agent that incorporating the rule-based curriculum learning (CL), similar to **DDQ-CL-rule**.

WDQN-SDPL: A warm-DQN agent that incorporating our proposed framework.

4.4 Settings

The policy learning of the agent is trained by DQN and DDQ algorithm with the same set of hyper-parameters on Movie-Ticket Booking (Li et al., 2018). The batch size is 64, and the learning rate is 0.001. The state encoding network is an MLP layer with 80 node. The optimizer for neural networks is RMSProp. And we employ the WDQN with the same set of hyper-parameters (Lee et al., 2019). The batch size is 16, and the learning rate is 0.001. The state feature encoder is an MLP layer with 300 node. We use Adam as the optimization algorithm. Besides, dialogues are ordered by our complexity evaluation in the training process while during the testing process, dialogues are used as baselines with no order. The training epoch is 100, 200, 300 and each number is averaged over 5 runs, each run tested on 2000 dialogues.

5 Automatic Evaluation

In this section, we first show the experiment results of our proposed method on Movie-Ticket Booking and MultiWoz datasets to verify the effectiveness of our proposed framework. Then we make a comprehensive qualitative analysis to show the merits of automatic evaluation for dialogue complexity.

5.1 Main Results

Table 2 presents the main results at the training epoch on the Movie-Ticket Booking dataset. For each agent, we report its results in terms of success rate, average reward, and average turns. Compared to the original DDQ agent, DDQ-CR, DDQ-CL-rule and DDQ-SDPL respectively achieve 8%, 7% and 10% improvements in terms of success rate, which confirms curriculum learning can facilitate and stable the training process of RL agents.

Further, our proposed automatic evaluation method for dialogue complexity outperforms the empirical method by 3%. The result represents applying the heuristic attribute to model the dialogue complexity is insufficient and it is hard to make effective handcrafted rules, due to subjectivity and diversity of dialogues and the limit of user-generated training. Our proposed method models the dialogue complexity in an automatic method evaluating dialog state differential space of RL agents. For other metrics, average reward and average turns, our method also achieves better performance. Apart from overall performance gain, we observe our method converges fast to similar accuracy 76% at epoch 100, compared to 73% of DDQ agent at epoch 300. It proves a better evaluation method for dialogue complexity can bootstrap training steps of RL agents. Besides,

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN (Li et al., 2017)	0.42	-3.84	31.93	0.53	10.78	22.72	0.64	27.66	22.21
DDQ (Peng et al., 2018b)	0.60	20.35	26.65	0.71	36.76	19.55	0.73	39.97	18.99
DDQ-CR	0.62	24.82	23.78	0.78	46.52	14.05	0.81	50.16	13.27
DDQ-CL-rule	0.71	32.48	19.01	0.76	45.36	14.56	0.80	48.15	13.59
DDQ-SDPL	**0.76**	**44.75**	**14.90**	**0.80**	**51.15**	**13.29**	**0.83**	**54.63**	**12.73**

Table 2: Results of different task-completion dialogue agents including the DQN,DDQ on Movie-Ticket Booking (Li et al., 2018) and ablation experiments of applying different curriculum learning strategies. The training epoch = {100, 200, 300}. (Success: success rate)

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
WDQN (Lee et al., 2019)	0.37	-8.93	15.33	0.46	3.26	13.94	0.47	5.33	13.07
WDQN-CR	0.36	-5.16	15.10	0.45	3.21	14.02	0.51	10.68	13.79
WDQN-CL-rule	0.40	-4.79	14.79	0.48	6.41	13.19	0.49	9.35	13.85
WDQN-SDPL	**0.45**	**2.51**	**13.49**	**0.52**	**11.04**	13.36	**0.57**	**12.22**	13.81

Table 3: Results of WDQN agent on MultiWOZ (Budzianowski et al., 2018) and ablation experiments of applying different curriculum learning strategies. The training epoch = {100, 200, 300}. (Success: success rate)

DDQ-CR only applied curiosity-driven exploration with no curriculum learning and it can improve the agent's performance. While based on the curiosity reward's improvement, there is another promotion after applying the curriculum learning. And this ablation study demonstrates the effectiveness of the curriculum learning.

Table 3 shows main results at the training epoch on the MultiWOZ dataset. We aim to verify the effectiveness of our method on the more complicated dataset since MultiWOZ contains multi-domains and more dialog turns. Here we choose another RL agent, WDQN, as the baseline to show our proposed framework can be applied to other RL based models. Results show our method WDQN-SDPL outperforms the empirical rule WDQN-CL-rule by 6%, higher than the improvement of 3% on Movie dataset. The possible reason is that MultiWOZ is significantly complicated than Movie dataset and only the simple attribute of slot numbers evaluates dialog complexity insufficiently. For the Turns metric, our method achieves similar performance with the baselines. We assume the domain switching on MultiWOZ leads to increasing dialog turns to achieve a higher success rate. The comparison exhibits the effectiveness of our automatic evaluation method. Besides, the ablation study in WDQN-CR and WDQN-SDPL also demonstrates the effectiveness of the curriculum learning.

In summary, our proposed curriculum learning framework facilitates the training process of RL agents. Moreover, our automatic evaluation method for dialogue complexity consistently outperforms the rule-based empirical method.

5.2 Analysis

Comparison with Heuristic Complexity We analyze three empirical attributes to show the correlation between heuristics and proposed automatic dialogue complexity evaluation: slot number, dialog length, reward, and domain switching times. We show the correlation statistics results in Fig 3. Fig 3(a) and Fig 3(b) shows the statistics of slots number, dialogue length and reward in complexity ordered instances evaluating by our proposed dialog complexity evaluation on Movie-Ticket Booking dataset. And we perform the qualitative analysis on the correlation between dialogue complexity and the number of domain switching on the MultiWoz dataset in Fig 3(c). Our evaluation method for dialogue complexity reflects positive connections to heuristic methods in these dimensions, which to some extent explains the effectiveness of our method. On the other hand, our method considers multiple empirical dialog attributes which is more flexible than heuristics. We show more comparison analysis details in the Appendix.

Learning Efficiency Fig 4 shows the learning curves comparing different agents, DDQ, DDQ-CL-rule and DDQ-SDPL, to investigate the relative contribution of the automatic complexity evaluation and the curriculum learning strategy. The

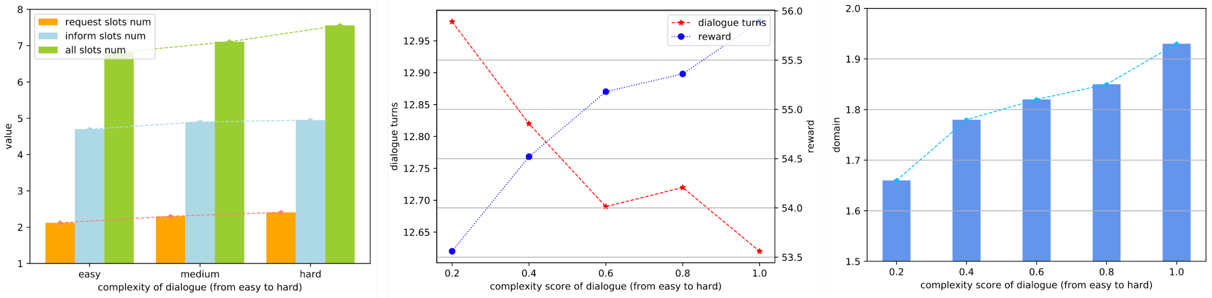

(a) *Slot numbers* (b) *dialog length & reward* (c) *domain switching times*

Figure 3: The correlation statistics between heuristics and our proposed automatic dialogue complexity evaluation. We count three heuristic attributes: slot numbers(request slot, inform, all), dialog turns and reward, domain switching times based on ordered training samples evaluating by our proposed automatic complexity evaluation. And we divide obtained dialog complexity scores into different complexity intervals(easy, medium, hard) for statistics.

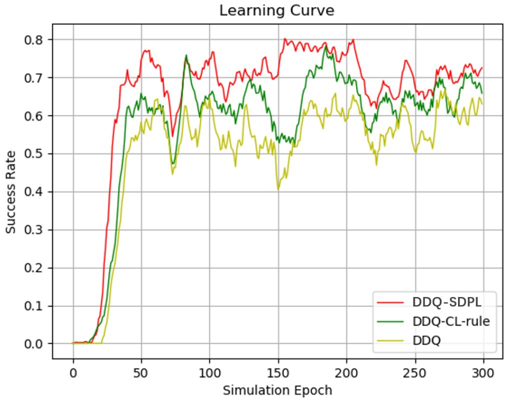

Figure 4: Learning curves of the DDQ, DDQ-CL-rule, DDQ-SDPL. The DDQ-SDPL outperforms baselines and converges fast to achieve higher accuracy.

DDQ-SDPL outperforms the baseline DDQ and DDQ-CL-rule and at each training step and converges fast to higher accuracy.

6 Human Evaluation

For human evaluation, we hire human experts to compare pairwise between DDQ-SDPL and baselines. Given a certain user goal, each expert is asked to read two simulated dialog sessions around this user goal, one from DDQ-SDPL and another from the other baseline. We randomly sample 100 goals for each baseline. For each goal, 3 experts are asked to judge which dialog is better (win, draw or lose) according to different subjective assessments: quality and task success. The quality metric evaluates whether the agent policy provides the user with the required information efficiently.

VS.	Quality			Success		
	W	D	L	W	D	L
DDQ	46	24	30	58	25	17
DDQ-CL-rule	41	28	31	49	26	25

Table 4: Human preference on dialog session pairs that DDQ-SDPL wins (W), draws with (D) or loses to (L) baselines on quality and success by majority voting.

Table 4 shows the results of the human preference by majority voting. DDQ-SDPL outperforms other baselines significantly in all aspects (sign test, p-value < 0.01). Note that the difference between DDQ-CL-rule and DDQ-SDPL is only in the dialog complexity evaluation. This demonstrates again the advantage of the automatic complexity evaluation in DDQ-SDPL over the heuristic method. The human preferences agree well with the results of the automatic evaluation, which also indicates these experimental metrics are reliable to reflect user satisfaction to some extent. Besides, we show some sampled cases in the Appendix to demonstrate the effectiveness of our proposed learning framework.

7 Conclusion

In this paper, we propose a novel curriculum learning framework to improve dialog policy learning by scheduling its learning process from easy to complex. We further propose an automatic dialog complexity evaluation for curriculum scheduling. The effectiveness validation of SDPL is conducted on two dialogue datasets and the state-of-the-art dialog model demonstrates that our proposed learning framework is able to boost the performance of existing dialogue policy learning. Furthermore, we believe that this automatic curriculum learning framework can be applied to improve other types of reinforcement learning based NLP tasks.

Acknowledgments

This work was partially supported by National Key R&D Program of China No. 2019YFF0303300 and Subject II No. 2019YFF0303302, DOCOMO Beijing Communications Laboratories Co., Ltd, MoE-CMCC "Artifical Intelligence" Project No. MCM20190701.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 5016–5026. Association for Computational Linguistics.

Hengyi Cai, Hongshen Chen, Cheng Zhang, Yonghao Song, Xiaofang Zhao, Yangxi Li, Dongsheng Duan, and Dawei Yin. 2020. Learning from easy to complex: Adaptive multi-curricula learning for neural dialogue generation. *CoRR*, abs/2003.00639.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 484–495. Association for Computational Linguistics.

Jianfeng Gao, Michel Galley, Lihong Li, et al. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2019. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. *CoRR*, abs/1911.08717.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. VIME: variational information maximizing exploration. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1109–1117.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 379–386, Varna, Bulgaria. INCOMA Ltd.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Xiang Li, Yaoqin Zhang, Zheng Zhang, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, and Jianfeng Gao. 2019. Convlab: Multi-domain end-to-end dialog system platform. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 72–79. IEEE.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Çelikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 733–743. Asian Federation of Natural Language Processing.

Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao. 2018. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *arXiv preprint arXiv:1807.11125*.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Julia Kiseleva, Maarten de Rijke, Shahin Shayandeh, and Jianfeng Gao. 2020. Guided dialogue policy learning without adversarial learning in the loop. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2308–2317. Association for Computational Linguistics.

Pierre Lison and Serge Bibauw. 2017. Not all dialogues are created equal: Instance weighting for neural conversational models. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 384–394. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489. IEEE.

Cao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Curriculum learning for natural answer generation. In *IJCAI*, pages 4223–4229.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. 2012. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems*

2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, pages 206–214.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017a. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017b. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Yun-Nung Chen, and Kam-Fai Wong. 2018a. Adversarial advantage actor-critic model for task-completion dialogue policy learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6149–6153. IEEE.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018b. Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192, Melbourne, Australia. Association for Computational Linguistics.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabás Póczos, and Tom M. Mitchell. 2019. Competence-based curriculum learning for neural machine translation. *CoRR*, abs/1903.09848.

Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463.

Jürgen Schmidhuber. 1991. Adaptive confidence and adaptive curiosity. In *Institut fur Informatik, Technische Universitat Munchen, Arcisstr. 21, 800 Munchen 2*. Citeseer.

Jonathan Sorg, Satinder P. Singh, and Richard L. Lewis. 2010. Variance-based rewards for approximate bayesian reinforcement learning. In *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 564–571. AUAI Press.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Susanne Still and Doina Precup. 2012. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory Biosci.*, 131(3):139–148.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep Dyna-Q: Robust planning for dialogue policy learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3813–3823, Brussels, Belgium. Association for Computational Linguistics.

Ryuichi Takanobu, Runze Liang, and Minlie Huang. 2020. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition.

Christopher Tegho, Pawel Budzianowski, and Milica Gasic. 2018. Benchmarking uncertainty estimates with deep reinforcement learning for dialogue policy optimisation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 6069–6073. IEEE.

Paula Wesselmann, Yen-Chen Wu, and Milica Gašić. 2019. Curiosity-driven reinforcement learning for dialogue management. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7210–7214. IEEE.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

A Dataset

Table 5 shows the full statistics of Movie-Ticket Booking dataset. And the statistics of MultiWOZ dataset are shown in Table 6. And Table 7 demonstrates the full statistics of comparison that between the Movie-Ticket Booking and MultiWOZ dataset.

	Annotations
Intent	request, inform, deny, confirm_question, confirm_answer, greeting, closing, not_sure, multiple_choice, thanks, welcome
Slot	city, closing, date, distanceconstraints, greeting, moviename, numberofpeople, price, starttime, state, taskcomplete, theater, theater_chain, ticket, video_format, zip

Table 5: The data annotation schema of Movie-Ticket Booking dataset.

	Hotel	**Train**	**Attraction**	**Restaurant**	**Taxi**
Slots	price, type, parking, stay, day, people, area, stars, internet, name	destination, departure, day, arrive by, leave at, people	area, name, type	food, price, area, name, time, day, people	destination, departure, arrive by, leave by

Table 6: The dataset information of MultiWOZ.

B User goal

In the task-completion dialogue setting, the entire conversation is around a user goal $G = (C, R)$ implicitly, where C denotes the constraint and R is the requests. And the agent knows nothing about the user goal explicitly and its objective is to help the user to accomplish this goal. Every time a dialog is launched, the user goal is initialized by the user simulator at the beginning of a dialog session, by randomly sampling the *inform slots* and *requests slots* from the user goal. Generally, the definition of user goal contains two parts: *inform_slots* contain a number of slot-value pairs which serve as constraints from the user. *request_slots* contain a set of slots that user has no information about the values, but wants to get the values by interact with the agent.

A set of 1,000 user goals in MultiWOZ are used for automatic evaluation as shown in Table 8.

C Case study

To further analyze the effectiveness of our automatic complexity measurement qualitatively, we randomly choose three examples of dialogues with different normalized complexity scores. As shown in Table 9, it is clear to see that, the instance with more slots and domains obtains a higher complexity score in our automatic measurement, which is consistent with heuristics. Besides, Table 11 shows two dialog sessions with the same user goal in WDQN and WDQN-SDPL, respectively. The user goal consists of *train* and *restaurant* domains. The system is required to answer all the information in *Request* based on the constraints in *Inform*. When the user asks for some information about the restaurant, WDQN-SDPL manages to provide all the requested information, while original WDQN only answers part of the information(red line). Additionally, WDQN-SDPL answers the postcode of the restaurant (blue line) and the dialog proceeds successfully, however, WDQN fails.

	Movie-Ticket Booking	MultiWoz
Dialogues	2,890	8,438
Intents	11	13
Slots	29	25
Avg. turns per dialogue	7.5	13.68
Domains	1	7

Table 7: The statistics of two public datasets, Movie-Ticket Booking and MultiWoz.

Class	Attraction	Hospital	Hotel
Count	320	22	389
Police	Restaurant	Taxi	Train
22	457	164	421
Num.	Single	Two	Three
Count	328	549	123

Table 8: Domain distribution of user goals used in the automatic evaluation. A user goal with multiple domains is counted repeatedly for each domain.

Easy case with complexity score: 0.2		**Medium** case with complexity score: 0.6		**Hard** case with complexity score: 0.9	
Restaurant		**Train**		**Attraction**	
Inform	*Request*	*Inform*	*Request*	*Inform*	*Request*
food=european pricerange=expensive time=19:30	phone postcode address	destination=peterborough day=thursday departure=cambridge leaveAt=19:45	duration arriveBy	type=museum area=centre	entrance fee address postcode
		Restaurant		**Hotel**	
		Inform	*Request*	*Inform*	*Request*
		food=european pricerange=expensive	phone postcode address	price range=expensive stars=4 people=8 day=saturday	address area
				Taxi	
				Inform	*Request*
				arrive by=19:45	car type phone

Table 9: The case study of different training instances (Easy, Medium, Hard) with their different complexity scores. The instance with more slots and domains obtains a higher complexity score in our automatic measurement, which is consistent with heuristics.

Train		Restaurant	
Inform	Request	Inform	Request
destination=peterborough day=sunday departure=cambridge leaveAt=20:45	duration people= 7	food=british price range=expensive area=centre	postcode address

Table 10: User goal used in the case study.

WDQN	**WDQN-SDPL**
U: I need to book a train to peterborough. I will departing from cambridge. Yes, I need to leave sometimes after 20:45. I would like to leave on sunday.	U: I need to book a train to peterborough. I will departing from cambridge. I need to leave sometimes after 20:45. I would like to leave on sunday.
S: What day would you like your booking for?	S: What day would you like your booking for? What time would you like the reservation for?
U: Sunday is fine, what is the travel time on that?	U: That should work, what is the travel time on that?
S: What time would you like the reservation for?	S: The travel time is 51 minutes.
U: That should work, what is the travel time on that?	U: Yes, can you book that for 7.
S: What time would you like the reservation for?	S: Reference number is : 00000068.
U: can you book that for 7?	U: I want to find a expensive priced restaurant. I'm visiting Cambridge and would like some suggestions for an restaurant which serves british. What restaurants are located in the centre?
S: Reference number is : 00000068.	S: Do you have a preference for the price range? I have 3 options for you.
U: I want to find a expensive priced restaurant. I'm visiting Cambridge and would like some suggestions for an restaurant which serves british. What restaurants are located in the centre?	U: I am looking for a expensive restaurant.
S: Do you have a preference for the price range? I have 3 options for you.	S: All set . Your reference number is 00000045.
U: I am looking for a expensive restaurant.	U: May I have the address? What is the postal code?
S: All set. Your reference number is 00000045.	S: The address is 51 Trumpington Street City Centre.
U: What is the address? What is the postal code?	U: I will need the postcode though.
S: The address is 51 Trumpington Street City Centre.	S: The post code is cb21rg.
U: Thanks. I will need the postcode though.	U: I am all set. Have a nice day. Bye.
S: Your reference number is 00000045.	S: You are very welcome. Goodbye.
U: I am all set. Have a nice day. Bye.	
S: You are very welcome. Goodbye.	
Failure: User does not request the arrive time of the train, and the system does not book the restaurant for the user.	**Success**

Table 11: A sample dialog session comparison between WDQN and WDQN-SDPL in natural language.

Do Explanations Help Users Detect Errors in Open-Domain QA? An Evaluation of Spoken vs. Visual Explanations

Ana Valeria González*[1], **Gagan Bansal**[2], **Angela Fan**[3,4], **Yashar Mehdad**[3],
Robin Jia[3], and **Srinivasan Iyer**[3]

[1]University of Copenhagen, [2]University of Washington, [3]Facebook AI, [4]LORIA
`ana@di.ku.dk`
`bansalg@cs.washington.edu`
`{angelafan, mehdad, robinjia, sviyer}@fb.com`

Abstract

While research on explaining predictions of open-domain QA systems (ODQA) is gaining momentum, most works do not evaluate whether these explanations improve user trust. Furthermore, many users interact with ODQA using *voice*-assistants, yet prior works exclusively focus on *visual* displays, risking (as we also show) incorrectly extrapolating the effectiveness of explanations across modalities. To better understand the effectiveness of ODQA explanations strategies in the wild, we conduct user studies that measure whether explanations help users correctly decide when to accept or reject an ODQA system's answer. Unlike prior work, we control for explanation *modality*, *i.e.*, whether they are communicated to users through a spoken or visual interface, and contrast effectiveness across modalities. We show that explanations derived from retrieved evidence can outperform strong baselines across modalities but the best explanation strategy varies with the modality. We show common failure cases of current explanations, emphasize end-to-end evaluation of explanations, and caution against evaluating them in proxy modalities that differ from deployment.

1 Introduction

Despite copious interest in developing explainable AI, there is increasing skepticism as to whether explanations (of system predictions) are useful to end-users in downstream applications. For instance, for assisting users with classifying sentiment or answering LSAT questions, Bansal et al. (2021) observed no improvements from giving explanations over simply presenting model confidence. Similarly, Chu et al. (2020) observed that visual explanations fail to significantly improve user accuracy or trust. Such negative results present a cautionary tale for explainability and emphasize the need to evaluate explanations using careful user studies.

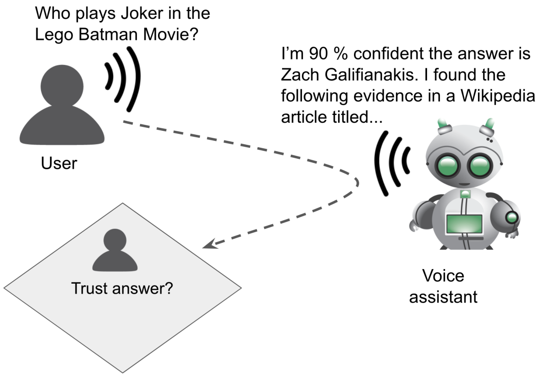

Figure 1: Using end-to-end user studies, we evaluate whether explanation strategies of open-domain QA assistants help users decide when to trust (or reject) predicted answers.

We explore the effectiveness of explanations for ***O**pen-**D**omain **Q**uestion **A**nswering* models, which involves answering users' questions (*e.g.*, *"Who plays the Joker in the Lego Batman movie?"*) using a large corpus (*e.g.*, Wikipedia). Such models are increasingly deployed not only in visual modalities (*e.g.*, Web search) but also in spoken ones (voice assistants).[1] Spoken interfaces for ODQA are also important because they make ODQA systems more accessible for users with visual impairments. Despite improvements in accuracy, deployed ODQA models remain imperfect. This motivates the need to provide users with mechanisms (*e.g.*, estimates of uncertainty or explanations) that can help improve *appropriate reliance* (Lee and See, 2004), *e.g.*, by allowing users to detect erroneous answers. We henceforth refer to a user's ability to distinguish correct and incorrect answers as *error-detectability*, and ask *Does explaining the system's reasoning, help improve error-detectability?* (Figure 1).

Alongside recent negative results (Bansal et al., 2021), Lamm et al. (2020) showed that visually complex "QED" explanations that communicate

---

* Work done while at Facebook AI.

[1] `https://www.perficient.com/insights/research-hub/voice-usage-trends`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1103–1116
August 1–6, 2021.

coreference and entailment information along with evidence marginally improve error-detectability. However, the study lacks the recommended baseline (Amershi et al., 2019; PAIR, 2019) of communicating model confidence which has been shown to be effective on other domains (Bansal et al., 2021). Also, the transferability of complex visual explanations to the spoken modality remains unclear. Although Feng and Graber (2019) compare visual explanations with presenting model confidence on a different QA task, i.e., answering timed, multi-clue trivia questions, it was unclear whether explanations led to appropriate reliance (Bansal et al., 2021); thus the effectiveness of explanations for end users of QA systems still remains unclear. In this paper, we set out to evaluate the ability of NL explanations in both visual and spoken modalities, to improve error-detectability for the task of ODQA for non-expert users over strong baselines.

However, explaining ODQA systems in the spoken modality may pose unique challenges, *e.g.*, because the same information content can impose higher cognitive demands when communicated by voice than visually (Sweller, 2011; Leahy and Sweller, 2016); potentially reducing effectiveness of longer, more complex explanations (*e.g.*, QED) in the *spoken* modality. Thus we also ask, *Can the most useful explanation strategy change with the modality?* In summary:

1. We present user studies evaluating how well explanations for ODQA help users detect erroneous answers (error-detectability). Unlike prior work, we evaluate explanations in both visual and spoken interfaces, and compare against calibrated confidence.
2. Our experiments with over 500 MTurk users confirm significant improvements in error-detectability for ODQA over showing confidence. To the best of our knowledge, our work is the first to show statistically significant improvements in appropriate reliance through NL explanations for non-expert users. (Section 4.1)
3. We show that the best explanation approach can change with the modality: while longer explanations (evidence paragraphs) led to the highest error-detectability in the visual modality, shorter explanations (evidence sentence) performed best in the spoken modality. We connect our observations with prior work on cognitive science and identify failure cases for ODQA explanations (Section 4.3).

2 Background

Open-domain QA. ODQA involves answering questions using a large, broad corpus of unstructured documents (*e.g.*, Wikipedia or the Web). More specifically, the questions are factoid and the target answer is present as a span in one of the documents (Voorhees et al., 1999).

Recent models for ODQA use a pipelined approach and contain two components: a document *retriever* that finds a subset of the most relevant documents from the corpus, and a *reader* that selects an answer span from the retrieved documents (Chen et al., 2017; Lee et al., 2019a). We use a state-of-the-art ODQA model and a benchmark dataset that contains questions asked by real lay users (Section 3.3).

An ODQA model's prediction can be explained by providing a justification in natural language, *e.g.*, by extracting snippets of text from the retrieved documents (*rationales*) or more generally by generating new text (*abstractive explanation*). For example, rationales can explain a text classifier using phrases in the input text that are relevant to the prediction (Lei et al., 2016). However, for some tasks, such as NLI (Camburu et al., 2018) and common-sense reasoning (Rajani et al., 2019), the input text alone may not contain enough meaningful justification for the prediction. For other tasks, the evidence can be spread across documents (*e.g.*, HotpotQA; Yang et al. (2018)). In such cases, abstractive explanations become more useful. While we primarily focus on extractive explanations, we also experiment with abstractive explanations (Section 3.1).

Evaluating explanations. One important reason to explain ODQA models is to improve error-detectability (Figure 1).[2] Many prior works evaluate the quality of NL explanations by comparing similarity with human-written explanations for tasks such as NLI, common-sense reasoning and fact verification (DeYoung et al., 2020; Paranjape et al., 2020; Swanson et al., 2020; Camburu et al., 2018; Rajani et al., 2019). Other works conduct user studies but rely on *proxy* tasks, *e.g.*, whether explanations allow users to anticipate the model predictions (Hase and Bansal, 2020; Nguyen, 2018). However, evaluating explanations on such

---

[2] Note that there exists other downstream applications of explanations, such as debugging models (Koh and Liang, 2017).

proxy tasks and metrics, which differ from the actual deployment setting, risks drawing misleading conclusions about the effectiveness of explanations in practice (Buçinca et al., 2020). Thus, we focus on directly evaluating explanations using user studies on error-detectability.

Feng and Graber (2019) found that explanations improve player accuracy when answering Quizbowl questions. Our task differs from Quizbowl in many respects: 1) Unlike Quizbowl where the users are trivia enthusiasts or experts, ODQA users are non-experts, lay people who ask questions to satisfy their information needs. Thus, ODQA users have no or very limited expertise in answering these questions. 2) While Quizbowl questions comprise multiple clues (incrementally revealed) for a single answer, ODQA questions typically contain a single clue. 3) Feng and Graber (2019) observed improvements from the explanations when their QA model was considerably more accurate than their users, outperforming the best trivia players. In contrast, we carefully design our user study so that on our study sample, users cannot achieve high performance by simply trusting the model (Section 3.3).

Visual vs. spoken modalities. We hypothesize that differences in processing of spoken and written information can substantially impact the effectiveness of NL explanations in ODQA. For example, Flowerdew et al. (1994) observed that one of the main differences in processing spoken versus written information is linearity. When listening, information progresses naturally, as opposed to reading, where people are able to jump back and forth in the text (Buck, 1991; Lund, 1991). This results in differences in recall of information across the two modalities (Osada, 2004). Although it is possible to repeat spoken information, Lund (1991) found that for some listeners, listening to information again was not as effective as re-reading. Another difference is the effect on concentration across modalities. Thompson and Rubin (1996) found that the heavier cognitive load imposed by listening to information can make people lose concentration more easily. Our experimental setup is the first to evaluate explanation effectiveness across these two modalities.

3 Experimental Setup

We evaluate explanation effectiveness for ODQA by varying the *type* of explanation and *modality* of communication. We combine variations of each factor to obtain explanation conditions (Section 3.1) for a state-of-the-art ODQA model (Section 3.3). We then deploy these conditions on Amazon Mechanical Turk (MTurk) to validate our hypotheses about the effectiveness of improving error-detectability (Section 3.2). We justify various design choices made to ensure quality in Section 3.4.

3.1 Explanation Types and Conditions

ODQA models can justify their predictions by pointing to *evidence* text containing the predicted answer (Das et al., 2018; Lee et al., 2019a; Karpukhin et al., 2020). We experiment with two types of *extractive* explanations:

- EXT-SENT: Extracts a sentence containing the predicted answer as evidence.
- EXT-LONG: Extracts a longer, multi-sentence paragraph containing the answer as evidence.

While extractive explanations are simpler to generate, we also evaluate a third explanation type that has potential to more succinctly communicate evidence spread across documents (Liu et al., 2019).

- ABS: (Abstractive) Generates a new text snippet to justify the predicted answer.

Explanation conditions. For the *spoken modality*, we test five conditions (two baselines and three explanation types): (1) BASE: present only the top answer, (2) CONF, a stronger baseline that presents the top answer along with the model's certainty, (3) ABS, (4) EXT-LONG, and (5) EXT-SENT. In the *visual modality*, we test two explanation types: EXT-LONG and EXT-SENT. We implemented these two types to contrast their effectiveness effectiveness across modalities (Figure 2, Section 3.4). For all explanation conditions, we show confidence, mention that the answer was obtained from Wikipedia and provide the source article. [3]

3.2 Hypotheses

We investigated five (pre-registered) hypotheses about the relative performance of various explanation conditions at improving the accuracy of error-detectability (Section 3.4):

H1 CONF will improve accuracy over BASE.

H2 *Spoken* EXT-SENT will improve accuracy over CONF— the explanation would provide additional context to help validate predicted answers.

---

[3] Appendix A shows more examples.

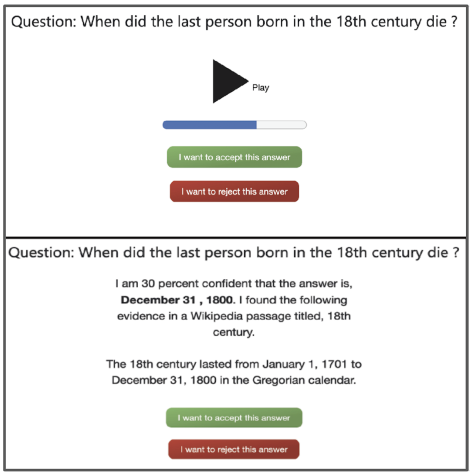

Question: When did the last person born in the 18th century die ?

Question: When did the last person born in the 18th century die ?

I am 30 percent confident that the answer is, **December 31 , 1800**. I found the following evidence in a Wikipedia passage titled, 18th century.

The 18th century lasted from January 1, 1701 to December 31, 1800 in the Gregorian calendar.

Figure 2: **UI** for spoken (top) and visual modalities (bottom) for the EXT-SENT explanation type. Users either read or hear an explanation and decide whether to trust or discard the QA system's prediction.

- **H3** *Spoken* EXT-SENT will lead to higher accuracy than *Spoken* EXT-LONG. Since the spoken modality may impose higher cognitive limitations on people (Section 2), concise explanations may be more useful despite providing less context.
- **H4** ABS will improve accuracy over *Spoken* EXT-SENT. ABS contains more relevant information than EXT-SENT (same length), which may help users make better accept/reject decisions.
- **H5** *Visual* EXT-LONG will lead to higher accuracy than *Spoken* EXT-LONG.

3.3 Implementation Details for Conditions

Dataset. We used the Natural Questions (NQ) corpus (Kwiatkowski et al., 2019). NQ is composed of anonymized queries posed by real users on the Google search engine. The answers are human-annotated spans in Wikipedia articles. As factoid web search is typically done through both spoken and visual modalities, this data is an ideal choice for our evaluation setup. To simplify the study, we restrict to questions with short target answers (< 6 tokens) (Lee et al., 2019b). This subset contains 80k training examples, 8,757 examples for development, and 3,610 examples for testing.

Model. We train the current (extractive) state-of-the-art model on NQ: Dense Passage Retrieval and Reader (DPR) (Karpukhin et al., 2020). Similar to Karpukhin et al. (2020), we split documents (Wikipedia articles), into shorter passages of equal lengths (100 tokens). To answer an input question, DPR uses two separate *dense* encoders $E_Q(\cdot)$ and $E_P(\cdot)$ to encode the question and all passages in the corpus into vectors. It then retrieves k most similar passages, where *passage similarity* to a question is defined using a dot product: $sim(q,p) = E_Q(q)^{\intercal} E_P(p)$. Given the top k passages, a neural reader (Section 2) assigns a passage selection score to each passage, and a *span score* to every answer span. [4]

Generating explanations. Extractive explanations use the passage associated with DPR's answer— EXT-SENT uses the sentence containing the answer whereas EXT-LONG uses the entire passage. Since DPR does not generate abstractive explanations, we simulate ABS by manually creating a single sentence that captures the main information of EXT-SENT and adds additional relevant information from EXT-LONG, whilst remaining the same length as EXT-SENT. To improve transparency, all explanation conditions also inform the source to the users, by providing them the *title* of the article. Figure 2 shows an example of the final EXT-SENT explanation condition. To convert text to speech, we use a state-of-the-art TTS tool. When spoken, questions in our final ABS and EXT-SENT conditions were on average 15 seconds long, EXT-LONG was between 30-40 seconds.

Confidence calibration. Confidence scores generated by neural networks (*e.g.*, by normalizing softmax scores) often suffer from poor calibration (Guo et al., 2017).To alleviate this issue and to follow best practices (Amershi et al., 2019; Bansal et al., 2021), we calibrate our model's confidence using *temperature scaling* (Guo et al., 2017), which is a *post hoc* calibration algorithm suitable for multi-class problems. We calibrate the top 10 outputs of the model. We defer additional details of calibration to Appendix B.

3.4 User Study & Interface

We conduct our experiments using Amazon Mechanical Turk. For each of the 7 conditions we hire 75 workers, and present each with 40 questions (this amounts to a total of 21,000 data samples) one-by-one, while showing them the model's answer (along with other condition-dependant information, such as confidence or explanation) and ask them to

---

[4] We re-score each answer using the product of the passage and span score and use the highest-scored answer as the prediction— Our initial analysis showed that this re-scoring improved exact match scores of predicted answers.

either *accept* the model's prediction if they think it is correct or *reject* it otherwise. Figure 2 shows an example. Additional details about the platform and participants can be found in Appendix D.

Question selection. We sample a set of questions on which the model's aggregate (exact-match) accuracy is 50%; thus any improvements in error-detectability, beyond random, must be a result of users making optimal assessment about the model's correctness. To improve generalization, we average results over three such mutually exclusive sets of 40 questions. Before sampling the questions, we removed questions that were ambiguous or questions where the model was correct but the explanations failed to justify the answer. Appendix C contains additional details on question selection.

Incentive scheme. In addition to providing a fixed upfront pay of $10 for participating in the task, to encourage workers to engage, we also used a bonus-based strategy (Bansal et al., 2019) — When users accept a correct answer, we provide a 15 cent bonus, but when they accept an incorrect answer they lose the same amount. When they reject an answer, however, they do not receive any bonus.[5] This aims to simulate the real-world cost and utility of users choosing to believe answers of an ODQA model. The maximum cumulative reward is $ 2.70. These values were chosen to ensure workers earned minimum a $15 hourly wage.

Post-task survey. After the main task, we asked participants to (1) rate the length of responses, (2) rate their helpfulness and (3) give us general feedback on what worked and how explanations could be made better. For the *spoken modality*, we also asked participants to rate the clarity of the voice to understand if issues in text-to-speech confused them. Appendix E contains the complete survey.

Metrics for error-detectability. We quantify user performance at error-detectability using the following three metrics:

1. **Accuracy**: Percentage of times a user accepts correct and rejects incorrect answers. A high accuracy indicates high error-detectability.
2. **% Accepts | correct**: Indicates the true positive rate, *i.e.*, percentage of times the user accepts *correct* answers.

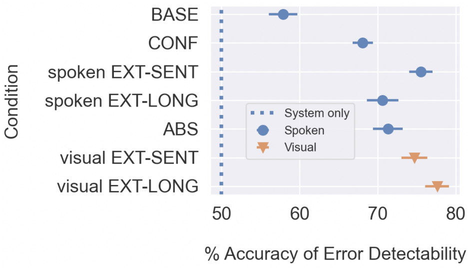

Figure 3: Accuracy of users. In the *spoken modality*, EXT-SENT explanations yield the best results and is significantly better than CONF. In the *visual modality*, EXT-LONG perform best. We observe a statistically significant ($p < 0.01$) difference between EXT-LONG in visual versus spoken, perhaps because of differences in user's cognitive limitations across modalities.

3. **% Accepts | incorrect**: Indicates the false positive rate, *i.e.*, percentage of times the user accepts *incorrect* answers. If a setting yields a high number, this would indicate that this setting misleads users more often.

We do not present true and false negative rates because conclusions are similar. We additionally measure time spent on each question and cummulative reward. These metrics are explained in Appendix F. When computing all metrics, we removed the first 4 questions for each worker to account for workers getting used to the interface. We preregistered this procedure prior to our final studies.

4 Results

To validate our hypothesis (Section 3.2) we compare explanation methods on the quantitative metrics (Section 4.1). To further understand participant behavior we analyze responses to the post-task survey (Section 4.2), and analyze common cases where explanations misled the users (Section 4.3).

4.1 Quantitative Results

Figure 3 displays average accuracy with 75 workers per condition. Similar to Lamm et al. (2020), to validate hypotheses and compute statistical significance, we fit a generalized linear mixed effects model using the `lme4` library in R and the formula `a~ c +(1|w) + (1|q)`, where `a` is accuracy, `c` is the condition, `w` is the worker id and `q` is the question id. We run pairwise comparisons of these effects using Holm-Bonferroni to correct for multiple hypothesis testing. For both the spoken and visual modalities, all conditions lead to significantly higher accuracies than BASE ($p < 0.01$).

---

[5] if bonus is negative, no deductions re made from base pay. Bonus is instead set to zero

Model confidence improved accuracy of error-detectability. In Figure 3, CONF achieves higher accuracy than BASE– 68.1% vs. 57.2%. This difference was statistically significant ($p < 0.01$), **validating H1**. While previous guidelines recommend displaying confidence to users (Amershi et al., 2019; Bansal et al., 2021), our observations provide the first empirical evidence that confidence is a simple yet stronger baseline against which explanations for ODQA should be compared.

Explaining via an evidence sentence further improved performance. The more interesting comparisons are between explanation types and CONF. In both modalities, EXT-SENT performed better than CONF. For example, in the *spoken modality*, EXT-SENT improved accuracy over CONF from 68.1% to 75.6% ($p < 0.01$); thus **validating H2**. Contrary to recent prior works that observed no benefit from explaining predictions, this result confirms a concrete application of explanations where they help users in an end-to-end task.

Longer explanations improved performance over concise explanations in the visual modality, but worsened performance in the spoken modality. Figure 3 shows that for the visual modality, EXT-LONG outperforms EXT-SENT explanations – 77.6% vs. 74.7% ($p < 0.4$). Conversely, for spoken, EXT-SENT is better than EXT-LONG– 75.6% vs. 70.4% ($p < 0.01$); thus **validating H3**. The decrease was severe enough that we no longer observed a statistically significant difference between EXT-LONG and CONF ($p = 0.9$), reemphasizing the importance of comparing against the latter. Although communicating the same content, *visual* EXT-LONG led to significantly better accuracy than their spoken version— 77.6% vs. 70.4% ($p < 0.01$); thus **validating H5**. These results indicate large differences, across modalities, in user ability to process and utilize explanations, and how these differences need to be accounted for while evaluating and developing explanations.

Despite improving conciseness, abstractive explanations did not help improve performance in the spoken modality. Figure 3 shows that ABS performs significantly worse than EXT-SENT in the spoken modality– 71.3% vs. 75.6% ($p < 0.01$) and thus we could **not validate H4**. This result indicates that the length of the explanation (*e.g.*, number of tokens) is not the only factor that affects user performance, instead, the density of information also increases cognitive load on users. This finding is in line with the Time Based Resource Sharing (TBRS) model (Barrouillet et al., 2007), a theory of working memory establishing that time as well as the complexity of what is being communicated, both play a role in cognitive demand. We also observe a similar effect in users' subjective rating of length of explanation (Section 4.2).

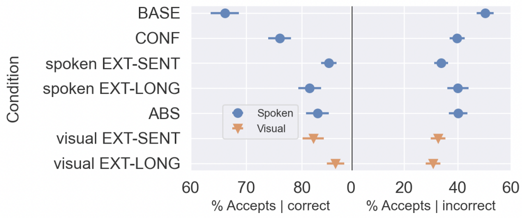

Figure 4: (Left) Explanations significantly increased participant ability to detect *correct* answers compared to CONF. (Right) However, only EXT-SENT in the spoken modality and both explanations in the visual modality decreased the rate at which users are misled.

All explanations significantly increased participants' ability to detect *correct* answers, but only some explanations improved their ability to detect *incorrect* answers. Instead of aggregate accuracy, Figure 4 splits and visualizes how often users accept correct and incorrect answers. For accepting *correct* model predictions, all *visual* and *spoken* explanation conditions significantly helped compared to CONF (at least $p < 0.05$).

For accepting incorrect predictions, in the *spoken modality*, only EXT-SENT is significantly better (*i.e.*, lower) than CONF—34% vs. 40% ($p < 0.05$). Whereas in the *visual modality*, both EXT-LONG and EXT-SENT lead to improvements over CONF—30% ($p < 0.01$) and 32% ($p < 0.05$), respectively. This shows that although explanations decrease the chance of being misled by the system, the least misleading explanations change with modality.

4.2 Qualitative Results

We analyzed user responses to the post-task survey to understand their experience, what helped them and how the system could serve them better. We discuss the main findings here and reserve additional results to the Appendix.

Length preference. We asked participants to rate the length of the explanation as *too short, short, right, long,* or *too-long*. Figure 5 shows the results. For EXT-LONG, over 85% of the workers perceived that in the *visual modality*, responses

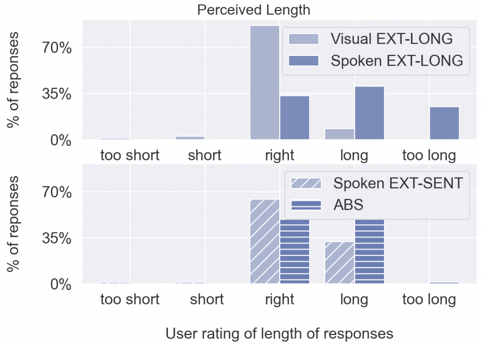

Figure 5: Users rated how they perceived the length of each explanation strategy. **Top:** Spoken explanations were perceived to be longer. **Bottom:** While EXT-SENT and ABS were the same length, the latter was rated as longer more often perhaps because of its complexity.

were the right length. On the other hand, in the *spoken modality*, only 30% of participants agreed the length was right. Thus, user's subjective ratings for the same explanation type were dramatically different across modalities. Indicating, in addition to affecting error-detectability, the modality also changes users' subjective preferences.

While ABS and EXT-SENT were the same duration, users rated ABS as longer than EXT-SENT. As mentioned before, this relates to the TBRS model of working memory (Barrouillet et al., 2007). We hypothesize that *our* ABS explanations, which integrate more information than EXT-SENT in the same amount of time, were more taxing for user's working memory, thereby reducing error-detectability and increasing perception of length.

User feedback. To understand how we can develop better explanations, we asked participants: *Do you have any additional feedback on what the system can improve?* To analyze responses, two annotators (authors) coded 400 responses. After removing responses that were not descriptive (*e.g.*, "can't think of anything to improve"), 175 responses remained for the final analysis. We computed the inter-annotator agreement using Cohen's k (=0.74). Here we describe the most interesting findings, but Appendix F shows additional results and details.

In BASE, where the answer was provided with no additional information, about 50% of **participants mentioned that they would have liked it if the voice changed with system certainty**. In CONF, around 30% of participants give this feedback.

For EXT-SENT in both modalities, EXT-LONG in the visual modality, and ABS, 10-35 % of **participants would like the level of detail to adapt to the model certainty**. Users would like to have more detail *only* when the model is not confident.

For EXT-LONG in the *spoken modality*, the feedback centered around length. 78% of participants mentioned that responses should be shorter, which aligns with the higher perceived length of the explanations in Figure 5. For the *visual modality*, 40 % of participants mention that highlighting some key items would have made it even easier and faster. Introducing highlights would improve the visual interface and would likely increase the differences in modality already observed.

Finally, for all explanation conditions, 20-45% of **participants said they would like to see explanations from multiple sources for an answer**, *e.g.*, from non-Wikipedia sources to help them better decide whether to trust the answer.

4.3 What Misleads Users?

To understand how explanations can mislead users, we analyzed questions where users frequently accepted incorrect predictions (false positives). A single annotator then followed a similar coding procedure to detect categories of such questions. We found that users were frequently misled on the same 30% of our study questions. Below we describe the two main categories:

Plausible explanations. We find 60-65% cases where an explanation does not confirm the predicted answer but makes it seem plausible, misleading users into accepting incorrect responses. This phenomenon is similar to prior work in psychology that has shown that people often fail to evaluate the accuracy of information when they have little prior knowledge and information seems plausible (Hinze et al., 2014).

> **Question:** *Who is the patron saint of adoptive parents?*
> **Response:** I am 37 percent confident that the answer is, **Saint Anthony of Padua**. I found the following evidence in a wikipedia passage titled, Anthony of Padua: Saint Anthony of Padua, born Fernando Martins de Bulhoes, also known as Anthony of Lisbon, was a portuguese catholic priest and friar of the Franciscan order.

In the example above, the model is incorrect (true answer is Saint William of Perth), but users were often misled to accept this answer because the evidence makes the prediction sound plausible.

Lexical overlap. The second most common mistake (from 30 to 35% of errors) that both *the model* and *the users* make is related to the lexical overlap between the question and the evidence. For instance, in the example below, the evidence contains the correct answer (15 teams) but many users are misled by the phrase *"A total of 30 teams play in the National League."*

Question: *How many teams are in the MLB national League?*
Response: I am 60 percent confident that the answer is, **30**. I found the following evidence in a wikipedia passage titled, Major League Baseball: *A total of 30 teams play in the National League(NL)* and American League (AL) , with 15 teams in each league .

5 Discussion

5.1 Why Explanations Worked for ODQA

Unlike previous studies (Bansal et al., 2021; Chu et al., 2020; Hase and Bansal, 2020), we observed significant improvements from explanations over only communicating confidence. One reason for our positive results could be owing to the nature of ODQA i.e. unlike tasks such as sentiment classification, where humans may be able to solve the task without relying on explanations, ODQA requires satisfying a user's information need, which may take considerably longer without explanations; users *require* additional help to navigate through vast amounts of information. Another potential reason is, in ODQA, presenting a single good explanation can allow users to verify whether the prediction is correct. In contrast, in sentiment analysis, even if the explanation points to evidence for a positive sentiment ("the smell was delicious"), there is always a chance that another phrase ("but the taste made me puke") renders the net correct label as negative. It is worth noting that like previous works, not all of our explanation methods provide significant value (Figure 3); thus the success from showing explanations still cannot be taken for granted but should instead be measured using well-designed user studies.

5.2 Implications and Recommendations

Another interesting question is how can our findings inform future research in explainable NLP.

Develop modality-specific explanations. Our results showed that the best explanation varied across modalities, indicating that evaluating explanations on one modality (*e.g.*, visual UI) and deploying them on another (*e.g.*, voice assistant) can lead to sub-optimal deployment decisions. As a result, explanations should be optimized for and evaluated in the task and settings in which they will be deployed in-the-wild.

Further study abstractive explanations. Longer explanations helped improve error-detectability in the visual modality, but they hurt in the spoken case, perhaps because of the increased cognitive load. This may indicate a trade-off between *information content* of explanation and its *cognitive load.* While we hoped abstractive explanations would achieve an optimal balance, results showed that they did not improve end-performance. Perhaps because even though they were more concise, they still had high information density. Though, abstractive explanations showed some promise. For example, compared to longer explanations, they improved speed of error-detectability by 2.2 sec (discussed in Appendix, Table 2) and their length was rated as more satisfactory (Figure 5). Thus future work should explore whether benefits of abstractive increase when explaining multiple sources (*e.g.*, in (Yang et al., 2018)) or candidate answers.

Enable interactive explanations. To manage a balance between information content and cognitive load one may also use interactive explanations (Weld and Bansal, 2018), where the system presents a concise explanation and lets users request more details, *e.g.*additional evidence, sources, or candidate answers (Section 4.2). Another option is *adaptive* explanations, where the model switches explanation strategies based on its confidence (Bansal et al., 2021).

6 Conclusion

We conducted user studies to understand whether explanations from a state-of-the-art open-domain QA system help improve error-detectability for end-users. Our study showed that for ODQA, simple explanations based on evidence snippets can significantly improve error-detectability and beat strong baselines such as communicating model's confidence. We observed this for multiple modalities of interaction: spoken and visual modalities. However, results also indicated that not every explanation type is guaranteed to improve performance over confidence and the best explanation strategy may change with the modality, *e.g.*, due to differences in users' cognitive abilities across modalities. Thus, developers and researchers of explainable ODQA systems should not take the effectiveness of explanations for granted and should evaluate and tune them on the tasks and modalities where these models will be eventually deployed.

7 Ethical Impact Statement

Recent work has shown that explanations may increase blind trust in systems (Bansal et al., 2021). Deploying such explanations in the wild is ethically fraught, hence we should better evaluate explanations using human evaluation before deployment. Our study expands the knowledge in this direction and show that current explanation strategies can work, but they can still considerably mislead users to accept incorrect model predictions. We hope that our findings and recommendations will have a positive impact on how explainable NLP is developed and evaluated in future work; namely, through carefully designed user studies which inform us of the real-world utility of explanations. In terms of data collection, the study was approved by IRB and no sensitive or personally identifiable data was collected, and users were informed that their efforts would end in a research publication.

References

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-ai interaction. In *CHI*.

Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S Lasecki, Daniel S Weld, and Eric Horvitz. 2019. Beyond accuracy: The role of mental models in human-ai team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 2–11.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. pages 1–16.

Pierre Barrouillet, Sophie Bernardin, Sophie Portrat, Evie Vergauwe, and Valérie Camos. 2007. Time and cognitive load in working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(3):570.

Zana Buçinca, Phoebe Lin, Krzysztof Z Gajos, and Elena L Glassman. 2020. Proxy tasks and subjective measures can be misleading in evaluating explainable ai systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 454–464.

Gary Buck. 1991. The testing of listening comprehension: an introspective study1. *Language testing*, 8(1):67–91.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pages 9539–9549.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Eric Chu, Deb Roy, and Jacob Andreas. 2020. Are visual explanations useful? a case study in model-in-the-loop prediction. *arXiv preprint arXiv:2007.12248*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2018. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. Eraser: A benchmark to evaluate rationalized nlp models. In *ACL*.

Shi Feng and Jordan Boyd Graber. 2019. What can ai do for me? evaluating machine learning interpretations in cooperative play. In *IUI*, pages 229–239.

John Flowerdew, Michael H Long, Jack C Richards, et al. 1994. *Academic listening: Research perspectives*. Cambridge University Press.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. pages 1321–1330.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? pages 5540–5552.

Scott R Hinze, Daniel G Slaten, William S Horton, Ryan Jenkins, and David N Rapp. 2014. Pilgrims sailing the titanic: Plausibility effects on memory for misinformation. *Memory & Cognition*, 42(2):305–324.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. pages 6769–6781.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Matthew Lamm, Jennimaria Palomaki, Chris Alberti, Daniel Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. 2020. Qed: A framework and dataset for explanations in question answering. *arXiv preprint arXiv:2009.06354*.

Wayne Leahy and John Sweller. 2016. Cognitive load theory and the effects of transient information on the modality effect. *Instructional Science*, 44(1):107–123.

John D Lee and Katrina A See. 2004. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019a. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019b. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Hui Liu, Qingyu Yin, and William Yang Wang. 2019. Towards explainable NLP: A generative explanation framework for text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5570–5581, Florence, Italy. Association for Computational Linguistics.

Randall J Lund. 1991. A comparison of second language listening and reading comprehension. *The modern language journal*, 75(2):196–204.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. pages 5783–5797.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

Nobuko Osada. 2004. Listening comprehension research: A brief review of the past thirty years. *Dialogue*, 3(1):53–66.

Google PAIR. 2019. The People + AI Guidebook. `https://pair.withgoogle.com/guidebook/`. Accessed: 2021-1-28.

Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. An information bottleneck approach for controlling conciseness in rationale extraction. pages 1938–1952.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Kyle Swanson, Lili Yu, and Tao Lei. 2020. Rationalizing text matching: Learning sparse alignments via optimal transport. pages 5609–5626.

John Sweller. 2011. Cognitive load theory. In *Psychology of learning and motivation*, volume 55, pages 37–76. Elsevier.

Irene Thompson and Joan Rubin. 1996. Can strategy instruction improve listening comprehension? *Foreign Language Annals*, 29(3):331–342.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer.

Daniel S Weld and Gagan Bansal. 2018. Intelligible artificial intelligence. *ArXiv e-prints, March 2018*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. pages 2369–2380.

A Explanation Examples

In Table 1, we show an example of how the responses and explanations looked for each of the conditions. We also indicate in which modalities each explanation is shown in our experiments.

B Temperature Scaling

Temperature scaling (Guo et al., 2017), a multi-class extension of Platt Scaling (Platt et al., 1999), is a post-processing method applied on the logits of a neural network, before the softmax layer. It consists of learning a scalar parameter t, which decreases or increases confidence. t is used to rescale the logit vector z, which is input to softmax σ, so that the predicted probabilities are obtained by $\sigma(z/t)$, instead of $\sigma(z)$.

In our experiments, the model is set to pick from the top 100 solutions, however, in many cases the correct answer occurs within the top 10 items. For our purposes we calibrate the confidence scores of the top 10 outputs. We use the publicly available scripts provided by Guo et al. (2017).[6]

The model confidence before and after calibration can be seen in Figure 6.

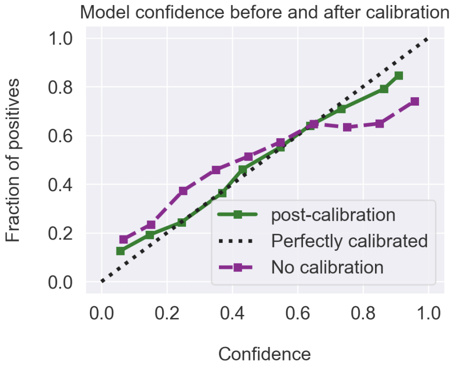

Figure 6: Confidence before and after calibration.

C Additional Preprocessing

Additional preprocessing to ascertain the quality of stimuli in each modality was required. Before sampling questions for the task, to ensure a high-quality and non-ambiguous experience for MTurk workers, we manually filter out several "problematic" questions:

- **Ambiguity in the question**: For various questions in NQ, multiple answers can exist. For example, the question: *when was King Kong released?*, does not specify which of the many King Kong movies or video games it refers to. These cases have been known to appear often in NQ (Min et al., 2020). We remove such questions from our subset.

- **The gold answer was incorrect**: Many examples in NQ are incorrectly annotated. As it is too expensive to re annotate these cases, we remove them.

- **Answer marked incorrect is actually correct** : We present both correct and incorrect questions to users. There are cases where the predicted answer is marked incorrect (not exact match) but is actually correct (a paraphrase). We manually verify that correct answers are paired with contexts which support the answer.

- **Correct answer but incorrect evidence**: The model sometimes, though not as often, chooses the correct answer but in the incorrect context. We discarded examples where the explanation was irrelevant to the question e.g. who plays Oscar in the office? *Oscar Nuñez, is a Cuban-American actor and comedian.*. In order to be able to make more general conclusions about whether explanations help in error-detectability, we restrict our questions to ones containing correct answers in the correct context.

- **Question and prediction do not match type**. We removed cases where the question asked for a certain type e.g. a date, and the prediction type did not match e.g. a location.

In the visual modality, to ensure readability, we fixed capitalizations. For the spoken modality, to ensure fluency and clarity, we manually (1) inserted punctuation to ensure more natural sounding pauses, and (2) changed abbreviations and symbols to a written out form e.g. *\$ 3.5 billion* to *3.5 billion dollars*.

D Task Setup: Additional details

Platform and participant details. We conduct our experiments using Amazon Mechanical Turk[7]. We recruited 525 participants in total, with approval ratings greater than 95 % and had a maximum of 8

---

[6] https://github.com/gpleiss/temperature_scaling

[7] https://www.mturk.com/

EXPLANATION TYPE	RESPONSE+EXPLANATION	MODALITY
BASE	The answer is, **two**.	Spoken
CONF	I am 41 percent confident that the answer is, **two**.	Spoken
ABS	I am 41 percent confident that the answer is, **two**. I summarized evidence from a wikipedia passage titled, Marco Polo (TV series). Netflix cancelled the show after two seasons, as it had resulted in a 200 million dollar loss.	Spoken
EXT-SENT	I am 41 percent confident that the answer is, **two**. I found the following evidence in a wikipedia passage titled, Marco Polo (TV series). On December 12, 2016, Netflix announced they had canceled "Marco Polo" after two seasons.	Spoken/Visual
EXT-LONG	I am 41 percent confident that the answer is, **two**. I found the following evidence in a wikipedia passage titled, Marco Polo (TV series). On December 12, 2016, Netflix announced they had canceled "Marco Polo " after two seasons. Sources told "The Hollywood Reporter" that the series' two seasons resulted in a 200 million dollar loss for Netflix , and the decision to cancel the series was jointly taken by Netflix and the Weinstein Company. Luthi portrays Ling Ling in season 1, Chew in season 2. The series was originally developed at starz, which had picked up the series in January 2012.	Spoken/Visual

Table 1: **Explanation examples:** Example of how system responses looked for each explanation type and baseline, for the question *How many seasons of Marco Polo are there?*

days for approval of responses in order to minimize the amount of spamming.

We use a random sample of 120 questions from our dataset which remains the same across all conditions. In order to keep each session per participant at a reasonable time and ensure the quality of the data wouldn't be affected by workers becoming exhausted, we opted for three fixed batches of 40 questions, all split as 50 % correct and 50 % incorrect. Workers could only participate once (only one batch in one condition). Participants took around from 35-45 minutes to complete the HITs, but were given up to 70 minutes to complete.

We monitored if their screen went out of focus, to ensure that participants did not cheat. We ensured that we had 25 user annotations per question. When analyzing the data, we remove the first 4 questions of each batch, as it may take participants a few tries before getting used to the interface. In the end, we collect about 21,000 test instances.

Task Instructions. Imagine asking Norby a question and Norby responds with an answer. Norby's answer can be correct or wrong. If you believe Norby's answer is correct, you can accept the answer. If you believe it is wrong, you can reject it. If the answer is actually correct and you accept it, you will earn a bonus of \$0.15. But, if the answer is wrong, and you accept it, you will lose \$0.15 from your bonus. If you reject the answer, your bonus is not affected. (Don't worry, the bonus is extra! Even if it shows negative during the experiment, in the end the minimum bonus is 0). In total you will see 40 questions in this HIT (you will only be allowed to participate once) and the task will take about 40 to 45 minutes. You can be compensated a maximum of \$13.50 for about 40-45 minutes of work. Some things to note:

1. You must listen to the audio before the options become available.
2. If you make it to the end there is a submit button there, however, in case of an emergency you can hit the quit early button above and you will get rewarded for the answers you provided.
3. You can play the audio as many times as you need but as soon as you click a choice you will be directed to the next item.
4. **IMPORTANT!!** Please do not look up questions in any search engine. We will monitor when the screen goes out of focus, so please keep the screen on focus or you might risk being rejected.
5. Finally, please do not discuss answers in forums; that will invalidate our results.

E Post-task Survey

1. I found the CLARITY of Norby's voice to be:

   (a) Excellent **(b)** Good **(c)** Fair **(d)** Poor **(e)** Very Poor

2. I found Norby's responses to be HELPFUL when deciding to Accept or Reject:

(a) Strongly Agree **(b)** Agree **(c)** Undecided **(d)** Disagree **(e)** Strongly Disagree

Can you give a few more details about your answer?

3. I found the LENGTH of Norby's responses to be:

   (a) Too Long **(b)** Long **(c)** Just right **(d)** Short **(e)** Too short

4. No AI is perfect and Norby is no exception. We are interested in helping Norby provide responses that can help users to determine whether to trust it or not (to accept or reject, just as you have done in this experiment). From your interaction with Norby, **do you have any additional feedback on what it can improve?**

F Results

Reward. Cummulative reward is the total dollar reward in bonuses earned by a worker based on the payoff described earlier. Note that, unlike accuracy, the payoff matrix is not symmetric wrt. user decision and correctness of predictions. We compute the differences in overall reward for each condition and observe the same trends as we discussed for accuracy. More specifically, all explanation conditions improve the final user reward, with EXT-SENT performing best in the spoken modality and EXT-LONG performing best overall. These differences are shown in Figure 7.

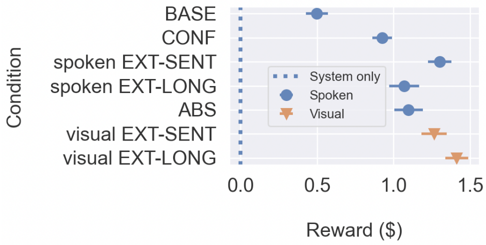

Figure 7: **Reward**: The scores presented here are out of $ 2.70. Although all explanations are better than CONF, the explanations leading to the highest rewards change across modalities.

Time differences. We measured the time (in seconds) that it took participants to complete each question. In Table 2 we present the median times averaged over all workers per condition. We also include an adjusted time, subtracting the length of the audio, in order to measure decision time.

CONDITION	SEC/QUESTION	ADJUSTED
SPOKEN MODALITY		
BASE	10.2 ± 1.6	8.3 ± 1.6
CONF	9.4 ± 1.5	6.0 ± 1.5
ABS	24.4 ± 1.5	7.0 ± 1.4
EXT-LONG	44.9 ± 1.6	9.2 ± 1.6
EXT-SENT	24.3 ± 1.7	7.6 ± 1.7
VISUAL MODALITY		
EXT-LONG	16.1 ± 1.7	-
EXT-SENT	10.4± 1.1	-

Table 2: Time differences across modalities. Time differences in the right column have been adjusted by removing the duration of the audio files. We observe that with additional information, users can make faster decisions than the BASELINE condition.

Voice quality. To verify that the quality of the text-to-speech tool that we employed did not negatively affect our experiments, we asked users to rate the clarity of the assistant's voice as *very poor, poor, fair, good*, or *excellent*. Around 90 % rated the voice as good or excellent. These results are shown in Figure 8.

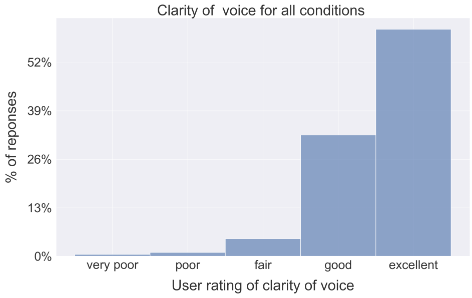

Figure 8: **Voice clarity:** Most participants found the voice of the assistant to be good or excellent.

Helpfulness. Participants were asked whether the responses helped them in their decision making. Their responses showed that CONF and all explanation conditions were perceived as helpful by at least 80% of participants, with no real differences among them except for EXT-LONG in the visual modality (which is perceived helpful by close to 90% of users). Interestingly, 50% of participants indicated BASE to be helpful. In contrast, our results in Figure 3 show that different explanations actually differ in their eventual helpfulness. These results suggest that subjective measures can sometimes correlate with actual performance when the differences are large, but for the most-part and smaller differences, the result from subjective rating can be unreliable. These findings align with prior observa-

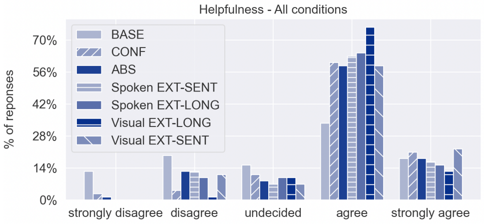

Figure 9: **Helpfulness:** Participants indicated how helpful responses were. These results reflect the large differences we see in performance (BASE vs the rest of the settings), but are not able to capture the more subtle differences among explanation strategies and CONF.

tion made (Buçinca et al., 2020) that showed that evaluating explanations on proxy metrics can lead to incorrect conclusions. These findings are shown in Figure 9.

User feedback. Users provided free-form written feedback on possible ways to improve the system. The prompt they saw was: *do you have any additional feedback on what the system can improve?* After converging on a final set of codes, two annotators coded up about 400 responses across all conditions. The codes and their descriptions can be found in Table 3. The codes are *not* mutually exclusive.

CODE	DESCRIPTION	CATEGORY
len-conciseness len-expand	users wish explanation was shorter users wish explanation was shorter	improvement on length
adapt-detail adapt-voice	users wish details adapted with confidence users wish voice adapted to confidence	adaptability feature
pres-change-confidence pres-highlighting	users wish confidence would be communicated differently e.g. the answer is probably.... users wish important facts would be highlighted	improve presentation
need-more-sources need-confidence need-source need-explanation need-link need-multiple-answers	users wish more source were provided users wish confidence was provided users wished a source was provided users wish an explanation would be provided users wish a link was provided users wish more than 1 answer was provided	need additional info

Table 3: The codes used to uncover areas of improvement from the post-experimental user feedback.

We found that many users across most conditions, would like **adaptability features** added. Additionally, we found that participants would like to be provided with multiple sources which converge on the answer. We also observe that for spoken conditions, **improvements on length** are mentioned more often. The full distribution of codes across conditions is shown in Table 4.

CONDITION	CODE	% PARTICIPANTS
BASE	adapt-voice	50
	need-confidence	36
	need-explanation	25
	need-source	17
CONF	need-explanation	38
	adapt-voice	29
	pres-change-confidence	14
	adapt-detail	10
	need-multiple-answers	10
	need-link	5
Spoken EXT-SENT	need-more-sources	44
	adapt-detail	28
	len-conciseness	22
	need-multiple-answers	17
	need-link	11
	len-expand	11
	pres-change-confidence	6
Spoken EXT-LONG	len-conciseness	78
	need-more-sources	15
	pres-change-confidence	4
ABS	len-conciseness	52
	need-more-sources	22
	adapt-detail	22
	pres-change-confidence	13
	need-multiple-answers	4
Visual EXT-SENT	need-more-sources	33
	adapt-detail	33
	len-expand	27
	need-multiple-answers	7
Visual EXT-LONG	pres-highlighting	40
	need-more-sources	33
	adapt-detail	10
	need-link	10
	pres-change-confidence	7

Table 4: Distribution of codes across all conditions. Codes are **not** mutually exclusive.

OntoEA: Ontology-guided Entity Alignment via Joint Knowledge Graph Embedding

Yuejia Xiang[1,2]*, **Ziheng Zhang**[1]*, **Jiaoyan Chen**[3], **Xi Chen**[1,2]†, **Zhenxi Lin**[1], **Yefeng Zheng**[1]
[1]Tencent Jarvis Lab, Shenzhen, China
[2]Platform and Content Group, Tencent, Shenzhen, China
[3]Department of Computer Science, University of Oxford, UK
`{yuejiaxiang,zihengzhang}@tencent.com, jiaoyan.chen@cs.ox.ac.uk`
`{jasonxchen,chalerislin,yefengzheng}@tencent.com`

Abstract

Semantic embedding has been widely investigated for aligning knowledge graph (KG) entities. Current methods have explored and utilized the graph structure, the entity names, and attributes, but ignore the ontology (or ontological schema) which contains critical meta information such as classes and their membership relationships with entities. In this paper, we propose an ontology-guided entity alignment method named OntoEA, where both KGs and their ontologies are jointly embedded, and the class hierarchy and the class disjointness are utilized to avoid false mappings. Extensive experiments on seven public and industrial benchmarks have demonstrated the state-of-the-art performance of OntoEA and the effectiveness of the ontologies.

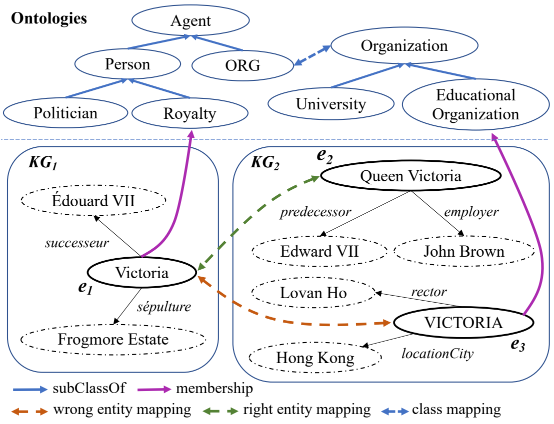

Figure 1: A class conflict example in entity alignment.

1 Introduction

Knowledge graphs (KGs) that are composed of entities and facts in the RDF form (i.e., RDF triples) are of vital importance in various applications, such as search engines and personal assistants (Hogan et al., 2020). Although there have been several large-scale KGs, the content of one individual KG is often incomplete, especially in supporting some domain-specific applications such as clinical AI assistants. As these KGs are developed separately, they are usually heterogeneous and supplementary to each other. Thus it becomes urgently needed to align multiple KGs (i.e., matching the equivalent elements) to fully explore their usability.

Recently, a few embedding-based methods have been proposed for entity alignment (EA) (Sun et al., 2020b; Zhang et al., 2020; Qi et al., 2021), in which entities are embedded into vectors and the equivalent entities are determined via calculating the similarity of their vectors. They extend the embedding of one KG to the embedding of multiple KGs into one vector space by, for instance, a transformation matrix, which is learnt from annotated mappings (a.k.a. seed mappings). They also utilize the entity names and attributes besides the graph structures for better performance. These methods, however, all ignore the ontology (or ontological schema) which is an important part of many KGs such as DBpedia (Auer et al., 2008) and Wikidata (Vrandečić and Krötzsch, 2014) as meta information for higher quality and usability. The ontology typically contains hierarchical classes and properties, and optionally defines some logical constraints such as the class disjointness, the property domain and range (Horrocks, 2008). Meanwhile, the KG usually clarifies the membership relationship between entities and classes.

The entity mappings by the above methods that do not exploit the ontologies may induce some class conflicts. Considering the example in Fig. 1, *Victoria* in KG_1 is often incorrectly aligned to *VICTORIA* in KG_2 by many embedding-based methods, but they belong to two potentially disjointed classes, namely *Person* and *Organization*. We find such class conflicted mappings are quite common in the wrongly predicted mappings. Considering the EN-FR-15K-V1 benchmark by (Sun et al., 2020b), 42.2% and 55.7% of the wrongly pre-

---

*Equal contribution

†Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1117–1128
August 1–6, 2021.

dicted mappings are class conflicted when running BootEA (Sun et al., 2018) and RSN4EA (Guo et al., 2019), respectively. These false positive mappings could be avoided if their ontologies are considered.

In this study, we propose an ontology-guided entity alignment method named OntoEA, which enriches the embedded semantics and avoids wrong mappings by exploring the class conflicts. OntoEA can work well in two contexts: *(i)* one ontology is shared across the to-be-aligned KGs; *(ii)* the ontologies of the to-be-aligned KGs are separated. The first context seems to follow a strict assumption but is actually quite common. For example, DBpedia has multilingual KGs (or versions) that share the same ontology, and in industrial scenarios, it is preferred to create a common ontology and then construct different KGs from different sources. The second context can be transformed into the first context by pre-aligning the ontologies, using existing ontology alignment systems such as PARIS (Suchanek et al., 2011) and LogMap (Jiménez-Ruiz et al., 2012), and/or cost-sensitive human intervention.

There are two challenges to utilize the ontology. First, it is difficult to embed two KGs together with the ontology, and as far as we know, there are currently no such solutions. Second, the class conflicts, indicated by the class disjointness in the ontology, are not all explicitly defined. Most of them should be learned from the KGs and they actually vary from KG to KG. For example, two classes *Human* and *Animal* are often disjointed in artwork KGs, but *Human* may belong to *Animal* in biological KGs. To address these challenges, we develop a joint embedding method that includes five modules for embedding the KGs, the ontology, the class conflicts, the membership relationships, and the seed mappings, respectively. Specifically, we develop a class conflict matrix (CCM) to represent different kinds of class conflicts, including those explicitly defined as disjoint, those indicated by the class hierarchy (e.g., sibling classes likely to be disjointed) and those plausible deduced from the KGs.

To the best of our knowledge, OntoEA is the first to utilize the ontology and the embedding for KG alignment. Extensive experiments on seven benchmarks have verified the effectiveness of the ontology guidance with both one shared ontology or two separated ontologies. OntoEA consistently outperforms the state-of-the-art baselines on all the benchmarks. For example, it on average achieves over 35% higher Hits@1, Hits@5, and MRR than the best baseline on MED-BBK-9K — a new and challenging industrial benchmark. Last but not least, we extend the current benchmarks with ontologies and membership relationships. The source code and benchmarks are publicly accessible at `https://github.com/ZihengZZH/OntoEA`.

2 Methodology

2.1 Problem Statement

A KG is denoted as $G = (E, R, T)$, where E, R, T are the sets of entities, relations and triples, respectively. Each triple $(h, r, t) \in T$, includes a head entity $h \in E$, a relation $r \in R$, and a tail entity $t \in E$. Their embeddings are denoted as $\boldsymbol{h}$, $\boldsymbol{r}$, $\boldsymbol{t}$, respectively. For two to-be-aligned KGs $G_i = (E_i, R_i, T_i)$ and $G_j = (E_j, R_j, T_j)$, an entity mapping, denoted as $m = (e_i, e_j)$ where $e_i \in E_i$, $e_j \in E_j$, indicates that e_i and e_j refer to the same real-world object. The entity alignment (EA) task aims to find all the mappings M between E_i and E_j, where we assume a small set of known entity mappings (or seed mappings) M_s are given.

Each KG is assumed to be associated with an ontology, which contains hierarchical classes and optionally disjoint constraints between classes. For simplicity, we consider the classes as entities in KG and the subsumption relationships between classes, often known as *rdfs:subClassOf*, as relations in KG. The simplified ontology is therefore regarded as a graph. For KGs G_i and G_j, their associated ontologies are represented as $O_i = (C_i, H_i)$ and $O_j = (C_j, H_j)$, respectively. We simply denote both O_i and O_j as $O = (C, H)$ if G_i and G_j share one ontology (i.e., $O_i \equiv O_j$); otherwise we denote the merged ontology after ontology alignment as $O = (C, H)$. Note C_i, C_j and C are the class sets, while H_i, H_j and H are the triple sets with only the *subClassOf* relation. Furthermore, the membership relationships, which link the entities and the corresponding classes, are denoted as B_i and B_j, and e.g., B_i links G_i and C via $b_i = (e_i, c)$ where $e_i \in E_i$ and $c \in C$. The class and membership embeddings are denoted as $\boldsymbol{c}$ and $\boldsymbol{b}$, respectively.

2.2 OntoEA Framework

As shown in Fig. 2, OntoEA includes five modules: *(i) entity embedding* which embeds each KG into a separate embedding space; *(ii) ontology embedding* which embeds the class hierarchical structure with a non-linear transformation; *(iii) confliction loss*

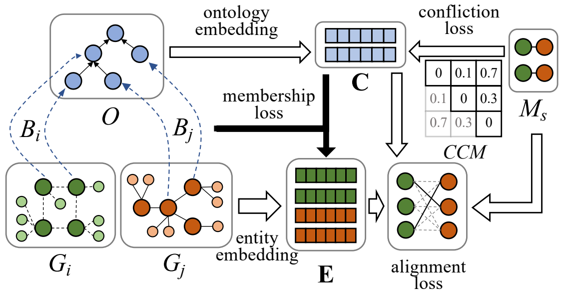

Figure 2: The framework of OntoEA.

which incorporates all potential class conflicts in the embeddings with the CCM; *(iv) membership loss* which incorporates the membership relationships and enables the joint learning of the entity embedding and the ontology embedding; *(v) alignment loss* which bridges the embedding spaces of the to-be-aligned KGs via seed mappings. An iterative co-training strategy is proposed to incorporate these five modules and the entity mappings are predicted with a pre-defined similarity measure between the joint embeddings.

2.3 Entity Embedding

To embed the KGs G_i and G_j, we adopt the translation-based method TransE (Bordes et al., 2013) which interprets a triple (h, r, t) as a translation by the relation r from h to t. The margin-based loss of (h, r, t) is defined as $f_e(h, t) = ||\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}||_2$, where $||\cdot||_2$ denotes the L2 norm. Besides, we extend the loss with a limit-based scoring loss (Zhou et al., 2017) to ensure the discrimination between the positive and negative triples and also lower scores for positive triples:

$$\mathcal{L}_E = \sum_{(h,t)\in T} \sum_{(h',t')\in T'} \{[\gamma_e^1 + f_e(h,t) - f_e(h',t')]_+ + \alpha_e[f_e(h,t) - \gamma_e^2]_+\}, \tag{1}$$

where $[\cdot]_+$ denotes the function $f(x) = \max(0, x)$, hyperparameters γ_e^1 and γ_e^2 control the margins, α_e balances the margin-based loss and the limit-based loss, and T' is the set of the negative triples with each triple sampled using the ϵ-truncated uniform negative sampling strategy (Sun et al., 2018) to distinguish two similar triples.

It is worth noting that this study focuses on the EA task and the joint embedding challenge, and TransE is chosen due to its simplicity and efficiency. Our OntoEA framework is open to other advanced KG embedding methods, such as GCN (Wang et al., 2018) and RSN (Guo et al., 2019).

2.4 Ontology Embedding

The shared or merged ontology $O = (C, H)$ is composed of triples with the *subClassOf* relation. For simplicity, a triple (c_h, r, c_t) is written as a class pair (c_h, c_t) in H where $r := subClassOf$. Inspired by Hao et al. (2019) on embedding the hierarchical graph structure, we calculate the scoring loss of (c_h, c_t) with a non-linear transformation, $f_o(c_h, c_t) = ||\tanh(\boldsymbol{W}_o\boldsymbol{c}_h + \boldsymbol{b}_o) - \boldsymbol{c}_t||_2$, where $\boldsymbol{W}_o \in \mathcal{R}^{d_o \times d_o}$ and $\boldsymbol{b}_o \in \mathcal{R}^{d_o}$ are the learnable parameters, and d_o denotes the ontology embedding dimension. This tends to encode each class as a sphere and each subclass as a vector in the same semantic space after the non-linear transformation, and the relative positions are employed to model the relations between class and its subclass.

Similarly, we adopt the margin-based and limit-based loss for training:

$$\mathcal{L}_O = \sum_{(c_h,c_t)\in H} \sum_{(c_h',c_t')\in H'} \{[\gamma_o^1 + f_o(c_h,c_t) - f_o(c_h',c_t')]_+ + \alpha_o[f_o(c_h,c_t) - \gamma_o^2]_+\}, \tag{2}$$

where H' denotes the set of negative class pairs with each pair sampled by replacing c_h or c_t following the uniform negative sampling strategy (Bordes et al., 2013), while γ_o^1, γ_o^2 and α_o are hyperparameters similar to γ_e^1, γ_e^2 and α_e in Eq. 1.

Note that we do not directly utilize TransE in the ontology embedding, because the *subClassOf* relation is transitive (e.g., we can infer (*Royalty*, *subClassOf*, *Agent*) via (*Royalty*, *subClassOf*, *Person*) and (*Person*, *subClassOf*, *Agent*) as shown in Fig. 1), which can lead to one-to-many and many-to-one mappings (or triples) in the ontology, and TransE cannot well address the relation with such transitive property (Lv et al., 2018).

2.5 Confliction Loss

We use a class conflict matrix (CCM) to represent the inter-class conflicts that are either explicitly defined by class disjointness or implicitly discovered from the entities. Within the CCM, the entry on the i^{th} row and j^{th} column, denoted as $m_{i,j}$, represents the conflict degree between class c_i and class c_j. For one ontology, the CCM is a squared and symmetric matrix, and we only maintain the upper triangular for higher efficiency. Given two classes c_i and c_j, $m_{i,j} \in [0, 1]$ is calculated as follows. First, we set $m_{i,j} = 0$ if $c_i \equiv c_j$, which ensures each class does not conflict with itself. Second, c_i and

c_j are regarded as fully conflicted, i.e., $m_{i,j} = 1$, if they are declared as disjointed by the ontology.[1] Third, c_i and c_j are regarded as not conflicted, i.e., $m_{i,j} = 0$, if they have at least one common member (entity) or some of their members are given as seed mapping (i.e., there exists (e_i, e_j) in M_s such that e_i belongs to c_i and e_j belongs to c_j). Note the above three conditions are matched sequentially, and the computation of $m_{i,j}$ is finished if any condition is met. Finally, if none of the three conditions is met, we follow the principle that the farther two classes are separated in the tree-like class hierarchy structure, the lower semantic similarity and higher conflict degree the two classes have (Mumtaz and Giese, 2020). To calculate the distance between c_i and c_j, we use the set of classes passed by routing from c_i and c_j to the root class, denoted as $S(c_i)$ and $S(c_j)$, respectively, and then adopt the ratio of the intersection of $S(c_i)$ and $S(c_j)$. Accordingly, we have $m_{i,j} = 1 - \frac{|S(c_i) \cap S(c_j)|}{|S(c_i) \cup S(c_j)|}$ where $|\cdot|$ denotes the set cardinality.

For the implicitly discovered conflicts, we assume the small conflict degree between two classes if their embeddings are similar (i.e., high cosine similarity). Thus we calculate another cosine similarity-based class conflict degree: $d_{cos}(c_i, c_j) = 1 - \cos(\boldsymbol{c}_i, \boldsymbol{c}_j)$, where $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ represent the embeddings of c_i and c_j. We propose to minimize the following negative log-likelihood loss to incorporate the class conflicts represented by CCM into the class embeddings,

$$\mathcal{L}_C = -\sum_{c_i \in C} \sum_{c_j \in C} m_{i,j} \log d_{cos}(c_i, c_j). \tag{3}$$

2.6 Membership Loss

We develop the membership embedding module to utilize the membership relationships, B_i and B_j, to associate the KG embedding spaces with the ontology embedding space, which is regarded as enhancing the KG embeddings with the ontology semantics. Given one membership relationship $b = (e, c)$, we utilize a non-linear transformation to map the entity embeddings to the ontology embedding space, and we calculate the scoring loss as $f_m(e, c) = ||\tanh(\boldsymbol{W}_m \boldsymbol{e} + \boldsymbol{b}_m) - \boldsymbol{c}||_2$ where $\boldsymbol{W}_m \in \mathcal{R}^{d_e \times d_o}$ and $\boldsymbol{b}_m \in \mathcal{R}^{d_o}$ are learnable parameters, d_e and d_o denote the dimension of KG embedding and ontology embedding, respectively. Similarly, we minimize the following margin-based and limit-based loss to model all the membership relationships:

$$\mathcal{L}_M = \sum_{(e,c) \in B} \sum_{(e',c') \in B'} \{[\gamma_m^1 + f_m(e, c) - f_m(e', c')]_+ + \alpha_m [f_m(e, c) - \gamma_m^2]_+\}, \tag{4}$$

where B' denotes the set of negative membership relationships with each relationship created by replacing the class c following the uniform negative sampling strategy (Bordes et al., 2013), and γ_m^1, γ_m^2 and α_m are hyperparameters similar to γ_e^1, γ_e^2 and α_e in Eq. 1.

2.7 Alignment Loss

For the alignment embedding module, OntoEA utilizes the seed mappings M_s to bridge the embedding spaces of G_i and G_j such that the equivalent mappings between two cross-KG entities can be calculated via some distance metrics, such as cosine similarity. Given a seed mapping $m = (e_i, e_j)$, its score is calculated as $f_a(e_i, e_j) = ||\boldsymbol{W}_a \boldsymbol{e}_i - \boldsymbol{e}_j||_2$ where $\boldsymbol{W}_a \in \mathcal{R}^{d_e \times d_e}$ is a learnable translation matrix, and the training loss is defined as,

$$\mathcal{L}_A = \sum_{(e_i, e_j) \in M_s} f_a(e_i, e_j). \tag{5}$$

2.8 Iterative Co-Training and Prediction

To incorporate the aforementioned modules and obtain the embeddings of G_i, G_j and O, we can directly minimize the following loss:

$$\mathcal{L} = \mathcal{L}_E + \mathcal{L}_O + \lambda_1 \mathcal{L}_C + \lambda_2 \mathcal{L}_M + \lambda_3 \mathcal{L}_A, \tag{6}$$

where λ_1, λ_2, and λ_3 are hyperparameters that balance the losses of confliction embedding, membership embedding and alignment embedding, respectively. Instead of directly optimizing $\mathcal{L}$, we use an iterative co-training strategy in OntoEA to reduce model complexity and accelerate model convergence. At each iteration, OntoEA first optimizes $\mathcal{L}_E$ and $\mathcal{L}_O$ independently, then sequentially optimizes $\mathcal{L}_C$ and $\mathcal{L}_M$, and finally optimizes $\mathcal{L}_A$. The iteration stops until some stopping criterion on the validation set is met.

With the embeddings of G_i, G_j and O, we calculate the entity mappings with the cosine similarity. Given two entities $e_i \in G_i$ and $e_j \in G_j$, the weighted similarity score is calculated as:

$$sim(e_i, e_j) = \beta \cos(\boldsymbol{e}_i, \boldsymbol{e}_j) + (1 - \beta) \cos(\boldsymbol{c}_i, \boldsymbol{c}_j), \tag{7}$$

---

[1] For ontologies by Web Ontology Language (OWL), the class disjointness is common and is usually defined as a constraint by the built-in relation *owl:disjointWith*.

where hyperparameter $\beta \in [0, 1]$ balances the similarities of entity embeddings and class embeddings. It is possible that one entity has multiple classes declared; for example, *Victoria* in Fig. 1 can be an entity of *Royalty* and another class like *Female Leader*. In this case we calculate the average of the embeddings of all the declared classes as the class embedding ($\boldsymbol{c}_i$ or $\boldsymbol{c}_j$).[2] In the prediction, for each entity in G_i to be aligned, we rank all candidate entities in G_j by their weighted similarity scores.

3 Benchmarks

3.1 Benchmark Details

For comprehensive evaluation, we adopt six popular EA benchmarks including EN-FR-15K-V1/V2, EN-DE-15K-V1/V2 and D-W-15K-V1/V2 from Sun et al. (2020b), and an industrial benchmark MED-BBK-9K from Zhang et al. (2020). Note that for the cross-lingual benchmarks within DBpedia (i.e., EN-FR and EN-DE), the to-be-aligned KGs share one ontology, while for the other benchmarks, the to-be-aligned KGs have different ontologies that need to be aligned beforehand.

The benchmarks themselves do not include ontologies, and we thus extract and append an ontology for each KG, which includes the class structure (i.e., *rdfs:subClassOf* relationships) and membership relationships. Each benchmark therefore contains two KGs, their ontologies, and associated membership relationships. The benchmark statistics are shown in Table 1 where "-15K" and "9K" in the benchmark names are omitted. For the KGs of DBpedia, we use the DBpedia ontology and the membership relationships from the DBpedia SPARQL endpoint[3] by querying the classes of each entity with *rdfs:type*. Note we also utilize the defined class disjointness constraints by *owl:disjointWith* for initializing CCM. For the two Wikidata KGs in D-W-V1/V2, we extract their ontologies and membership relationships from the Wikidata SPARQL endpoint[4] using queries with *rdfs:subClassOf* and *rdfs:type*. For the KGs of the industrial benchmark, we use domain knowledge (with the assistance of medical experts) to construct a shared, small-scale, but high-quality ontology, and the corresponding membership relationships.

Table 1: Statistics of the benchmarks.

Dataset		KG	Ontology[b]		Membership[c]	
			#Cls.	#Trs.	#Links	#Roots
share-O[a]	EN-FR-V1	EN/FR	189	755	15,000	639
	EN-FR-V2	EN/FR	104	755	15,000	533
	EN-DE-V1	EN/DE	175	755	15,000	155
	EN-DE-V2	EN/DE	86	755	15,000	165
	MED-BBK	MED	11	10	9,162	86
		BBK	11	10	9,162	3,362
not-share-O	D-W-V1	DB	172	755	15,000	306
		WK	140	695	15,000	342
	D-W-V2	DB	71	755	15,000	463
		WK	68	695	15,000	418

[a] "*share-O*" means the to-be-aligned KGs share one ontology while "*not-share-O*" means the to-be-aligned KGs have different ontologies.

[b] "#Cls." denotes the number of classes and "#Trs." denotes the number of *rdfs:subClassOf* relation triples.

[c] "#Roots" denotes the number of entities that have no *rdf:type* property and are linked to the root class.

[d] See Table 6 in the Appendix for more benchmark statistics.

3.2 Ontology Alignment

For the benchmarks whose KGs do not share one ontology, we first align the ontologies. We adopt two ontology alignment methods: *manual annotation* and *alignment system*. In the first method, we employ five annotators to annotate class mappings for each ontology pair and the classes annotated by more than three annotators are adopted. It is worth noting that the manual annotation of class mappings is worthwhile and often adopted in KG construction and curation because of the high quality and relatively small scale in comparison with the entity mappings. In the second method, we apply a state-of-the-art ontology alignment system named PARIS[5] and some ad-hoc pre-processing and post-processing for automatic class mapping computation. Please see Appendix C for more details on ontology alignment and merging. For the D-W benchmarks, we considered both methods and compared their performance (see Table 4).

4 Experiments

4.1 Experimental Setup

We compare OntoEA against state-of-the-art EA models including four translation-based models, MTransE (Chen et al., 2017); JAPE (Sun et al., 2017); SEA (Pei et al., 2019); BootEA (Sun et al., 2018), two graph neural network based models, GCNAlign (Wang et al., 2018); AliNet (Sun et al., 2020a), and one recurrent neural network based model, RSN4EA (Guo et al., 2018). We also compare OntoEA against some state-of-the-art models that additionally utilize entity surface information

---

[2] Note that we do not consider the inferred classes, such as *Person* for *Victoria* in Fig. 1.

[3] http://dbpedia.org/sparql

[4] https://query.wikidata.org/sparql

[5] http://webdam.inria.fr/paris/

Table 2: Overall results of OntoEA and the baselines.

Models		EN-FR-15K-V1			EN-FR-15K-V2			D-W-15K-V1			D-W-15K-V2			MED-BBK-9K		
		H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR
w/o SI	MTransE	.247	.467	.351	.240	.436	.336	.259	.461	.354	.271	.490	.376	.004	.014	.012
	JAPE	.262	.497	.372	.292	.524	.402	.250	.457	.348	.262	.484	.368	.003	.009	.009
	SEA	.280	.530	.397	.360	.651	.494	.360	.572	.458	.567	.770	.660	.199	.375	.287
	GCNAlign	.338	.589	.451	.414	.698	.542	.364	.580	.461	.506	.743	.612	.065	.153	.117
	BootEA	.507	.718	.603	**.660**	.850	.745	.572	.744	.649	**.821**	.926	.867	.307	.495	.399
	RSN4EA	.393	.595	.487	.579	.759	.662	.441	.615	.521	.723	.854	.782	.195	.311	.253
	AliNet	.258	.437	.339	.359	.569	.453	.270	.403	.331	.522	.698	.601	.017	.042	.033
	OntoEA	**.566**	**.818**	**.678**	.654	**.891**	**.757**	**.591**	**.808**	**.688**	.814	**.950**	**.873**	**.343**	**.546**	**.440**
Improv. best %		11.6	13.9	12.4	-0.9	4.8	1.6	3.3	8.6	6.0	-0.9	2.6	0.7	11.7	10.3	10.3
w/ SI	AttrE	.481	.671	.569	.535	.746	.631	.299	.467	.381	.489	.695	.585	.194	.363	.279
	MultiKE	.749	.819	.782	.864	.909	.885	.411	.521	.468	.495	.646	.569	.213	.367	.289
	RDGCN	.755	.854	.800	.847	.919	.880	.515	.669	.584	.623	.757	.684	.306	.425	.365
	OntoEA	**.797**	**.871**	**.832**	**.901**	**.981**	**.931**	**.553**	**.787**	**.656**	**.795**	**.943**	**.860**	**.517**	**.703**	**.604**
Improv. best %		5.6	2.0	4.0	4.3	6.7	5.2	7.4	17.6	12.3	27.6	24.6	25.7	68.9	65.4	65.5

(SI) (i.e., entity names) including AttrE (Trisedya et al., 2019), MultiKE (Zhang et al., 2019) and RDGCN (Wu et al., 2019). Since using and not using SI are usually regarded as two different evaluation contexts (Guo et al., 2019), we separately evaluate OntoEA without (w/o) and with (w/) SI. Note that OntoEA with SI is implemented by a simple but effective strategy which initializes the embeddings of the translation-based models in Section 2.3 and Section 2.4 (i.e., $\boldsymbol{h}$, $\boldsymbol{r}$, $\boldsymbol{t}$ and $\boldsymbol{c}$) with the pre-trained word embeddings of their names. As in Sun et al. (2020b), we use the unified multilingual word embeddings fastText (Bojanowski et al., 2017) for the cross-lingual benchmarks.[6] All the results of AliNet, and the results of MTransE, JAEP, SEA and GCNAlign on MED-BBK-9K are reproduced locally; while the other baseline results are taken from Sun et al. (2020b) and Zhang et al. (2020). We follow the same train (20%), validation (10%) and test (70%) splits as Sun et al. (2020b) and Zhang et al. (2020).

We implement OntoEA upon the open source library OpenEA[7]. We use the AdaGrad optimizer with a learning rate of 0.01. The batch sizes for entity embedding and ontology embedding are set to 4500 and 64, respectively, and their dimensions are both set to 300. The weight hyperparameters are set as $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 5$ and $\beta = 0.5$, and other hyperparameters are set as $\gamma_x^1 = 0.01$, $\gamma_x^2 = 2.0$ and $\alpha_x = 0.2$ for all the losses where $x \in \{e, o, m\}$. These hyperparamters are tuned w.r.t. the MRR on the validation set. Please see Appendix B for more implementation details.

With the embeddings we use the nearest neighbor search with cross-domain similarity local scaling (Lample et al., 2018) to calculate the entity matching of each to-be-aligned entity. In the evaluation, we rank matching candidates of each to-be-aligned entity and calculate the metrics of Hits@1 (H@1), Hits@5 (H@5) and mean reciprocal rank (MRR). In the following tables, the best (second best resp.) result is **bolded** (underlined resp.).

4.2 Overall Results

Table 2 reports the overall results of OntoEA and the baselines.[8] Overall, in both contexts of using and not using SI, OntoEA performs the best across all the benchmarks on all the metrics, except for H@1 on EN-DE-15K-V2 and D-W-15K-V2.

Without SI. Regarding the benchmarks that share one ontology, OntoEA achieves at least 10% higher performance in all the metrics over the best baseline BootEA on EN-FR-15K-V1 which has sparse KG structures. On EN-FR-15K-V2 with dense KG structures, OntoEA achieves higher H@5 and MRR but competitive H@1 in comparison with the best baseline. For the D-W benchmarks whose KGs have different ontologies, we have a similar finding: the outperformance of OntoEA is more significant on the benchmark with sparse KG structures (i.e., D-W-15K-V1) than on the benchmark with dense KG structures (i.e., D-W-15K-V2). On the one hand, dense KG structures indicate that more information can be utilized with better performance, and relatively the additional positive impact by the ontology becomes more limited. On the other hand, the ontologies of both

---

[6] The word embeddings are publicly available at https://fasttext.cc/docs/en/crawl-vectors.html.

[7] https://github.com/nju-websoft/OpenEA

[8] Results on the two EN-DE benchmarks, which lead to similar findings as the EN-FR benchmarks, are shown in Table 7 in the Appendix.

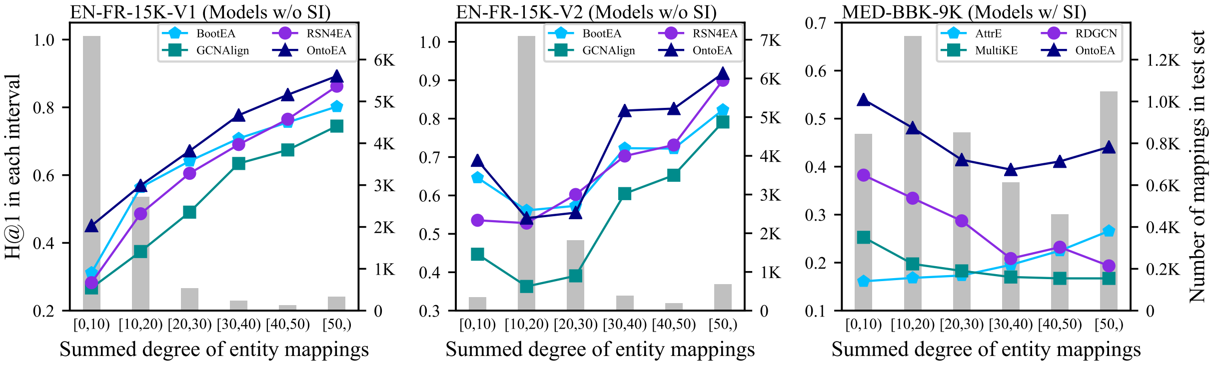

Figure 3: Results of OntoEA and some baselines on test entity mappings with different summed degrees.

EN-FR-15K-V2 and D-W-15K-V2 coincidentally have fewer classes (see Table 1), which would also lead to a lower impact of the ontology guidance. Regarding the industrial MED-BBK-9K, which is quite new and has shown more challenging w.r.t. these baselines (Zhang et al., 2020), OntoEA outperforms the best baseline by more than 10% in all the metrics. Although its ontology has a limited scale, the ontology is of high quality as it is specifically created with the domain knowledge.

With SI. OntoEA shows very promising overall results as OntoEA without SI. It outperforms all the baselines on all the benchmarks, with the improvements over the best baseline ranging from 2% to 68.9%. Regarding the industrial MED-BBK-9K, the performance improvement of OntoEA is especially significant with more than 60% in all the metrics. As aforementioned, this benchmark is challenging to these baselines and it has high-quality ontologies. It is worth mentioning that on the D-W benchmarks, the involvement of SI deteriorates the performance of OntoEA and the best baseline, while the positive impact of the ontologies (i.e., the performance gain of OntoEA over the best baseline) becomes more significant.

4.3 Ablation Studies

Model Component Analysis. To investigate different components in OntoEA, we compare OntoEA variants without the CCM loss (w/o $\mathcal{L}_C$), without the membership relationship loss (w/o $\mathcal{L}_M$), and without the ontology (w/o Onto.). The experimental results of OntoEA w/o SI on EN-FR-15K-V1/V2 are reported in Table 3, which shows that $\mathcal{L}_M$ contributes more to OntoEA than $\mathcal{L}_C$ as removing it leads to a larger performance drop. The results OntoEA w/ SI show similar findings and are thus omitted. As expected, the removal of the ontology deteriorates performance the most. All these results verify the significant role of the ontologies and their associated class conflicts and membership relationships.

Table 3: Results of OntoEA w/o SI and its variants with the removal of different components.

Models	EN-FR-15K-V1			EN-FR-15K-V2		
	H@1	H@5	MRR	H@1	H@5	MRR
OntoEA	**.566**	**.818**	**.678**	**.654**	**.891**	**.757**
w/o $\mathcal{L}_C$	.520	.781	.636	.589	.845	.701
w/o $\mathcal{L}_M$	.481	.750	.601	.549	.810	.665
w/o Onto.	.430	.698	.551	.545	.814	.664

Results on Different Test Entity Mappings. We split the test mappings according to the summed degree, i.e., $\deg(e_i, e_j) := \deg(e_i) + \deg(e_j)$ where (e_i, e_j) are the entity mappings. We select three different experimental settings (benchmarks and usage of SI) and Fig. 3 shows the results of OntoEA and some competitive baselines on different degree intervals. On EN-FR-15K-V1, OntoEA outperforms the baselines on all the intervals and the performance gap reaches the highest on [0,10). On EN-FR-15K-V2 with dense KG structures, OntoEA performs close to or slightly worse than BootEA on intervals [10,20) and [20,30) but outperforms all the baselines on the other intervals. On MED-BBK-9K, OntoEA performs much better than the baselines on all the intervals. All these observations are consistent with our findings from the overall results in Table 2 and also verify the effectiveness of OntoEA on different test entity mappings.

Analysis of Class Conflicts. This part analyses the predicted mappings that have class conflicts, with the results shown in Fig. 4. Note that the class conflict ratio of a method is the rate of its false positive mappings with class conflicts among all its false positive mappings. On EN-FR-15K-V1 and EN-FR-15K-V2, OntoEA significantly decreases the class conflict ratio to 3% and 0.3%, respectively,

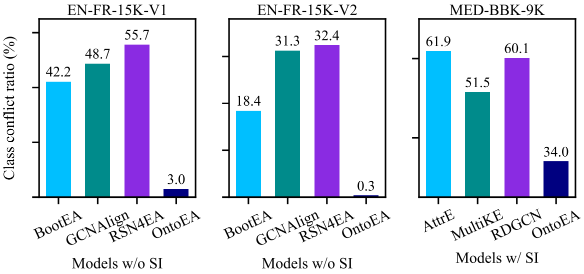

Figure 4: The class conflict ratios.

much lower than those of the baselines. On MED-BBK-9K, OntoEA reduces the class conflict ratio to 34.0% while MultiKE and RDGCN have 51.5% and 60.1%, respectively. The above observations illustrate that OntoEA has effectively implemented our motivation of using the ontologies and class conflicts to avoid false positive mappings.

Analysis of Ontology Alignment Methods. As the case that the to-be-aligned KGs have different ontologies is common, we analyse the impact of different ontology alignment methods proposed in Sec. 3.2, with the results in Table 4. We find that OntoEA with manual annotation achieves better results than OntoEA using the PARIS based alignment system method; the latter, however, still has competitive or better results in comparison with the best baseline (i.e., BootEA or RDGCN). This also provides motivations for the future research and industrial deployment work on iterative ontology alignment to keep a balance between the annotation cost and the alignment quality.

Table 4: Results of OntoEA using different ontology alignment methods where "-M" denotes the manual annotation while "-A" denotes the alignment system.

	Models	D-W-15K-V1			D-W-15K-V2		
		H@1	H@5	MRR	H@1	H@5	MRR
w/o SI	OntoEA-M	**.591**	**.808**	**.688**	.814	**.950**	**.873**
	OntoEA-A	.572	.795	.673	.762	.907	.827
	BootEA	.572	.744	.649	**.821**	.926	.867
w/ SI	OntoEA-M	**.553**	**.787**	**.656**	**.795**	**.943**	**.860**
	OntoEA-A	.510	.758	.621	.737	.893	.808
	RDGCN	.515	.669	.584	.623	.757	.684

5 Related Work

Embedding-based KG Alignment. These methods can be categorised into translation-based, GNN-based and RNN-based. The translation-based methods mainly rely upon some translation-based KG embedding models; for example, MTransE adopts TransE and various mapping losses to align cross-lingual KGs (Chen et al., 2017). To utilize more useful information in the KG, JAPE embeds the to-be-aligned KGs into one unified vector space via leveraging attribute correlations (Sun et al., 2017); AttrE exploits attributes by generating attribute character embeddings (Trisedya et al., 2019); and MultiKE makes full use of entity names, entity attributes and graph structures via multi-view learning (Zhang et al., 2019). The GNN-based methods are typically based on GNN variants for KG embedding, such as Graph Convolutional Networks (GCNs) Wang et al. (2018), dual-primal GCNs Sun et al. (2020a), and GCNs with attentive aggregation Sun et al. (2020a). As an RNN-based method, RSN4EA generates biased random walks on KGs and learns the embeddings by a recurrent skipping network (Guo et al., 2019). Some work has explored a semi-supervised learning setting, and for instance, BootEA (Sun et al., 2018) adopts the bootstrapping strategy to iteratively append new likely mappings during the learning process.

KG Embedding with Ontology. Only a few attempts have been made towards the KG embedding with ontology. The most relevant work is JOIE (Hao et al., 2019) which embeds the KG and the ontology in two embedding spaces and enables them to enhance each other by the cross-view links i.e., membership relationships. Some ontology embedding methods such as OWL2Vec* (Chen et al., 2021a) can be extended to jointly embed KG plus ontology since the KG can be regarded as an assertion part (ABox) of the ontology. However, as far as we know, there are currently no embedding methods that support two KGs along with their ontologies, let alone utilize the ontologies to augment embedding-based KG alignment.

6 Conclusions

This paper presented a novel method OntoEA to augment embedding-based entity alignment with the ontology. OntoEA enriches the semantics of KG embeddings by jointly learning the KGs, the ontology, the membership relationships and the seed mappings, and utilizes the potential class conflicts to avoid false mappings. The evaluation on multiple benchmarks has showed that OntoEA could achieve state-of-the-art performance and reduce the false positive mappings with class conflicts. For future work, we plan to utilize other ontology semantics besides the class hierarchy and class disjointness constraints (Chen et al., 2021b). We also plan to extend the entity embedding module with other advanced KG embedding methods.

References

Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2008. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *27th Annual Conference on Neural Information Processing Systems 2013*, pages 2787–2795.

Jiaoyan Chen, Pan Hu, Ernesto Jiménez-Ruiz, Ole Magnus Holter, Denvar Antonyrajah, and Ian Horrocks. 2021a. OWL2Vec*: Embedding of OWL ontologies. *Machine Learning*.

Jiaoyan Chen, Ernesto Jimenez-Ruiz, Ian Horrocks, Denvar Antonyrajah, Ali Hadian, and Jaehun Lee. 2021b. Augmenting ontology alignment by semantic embedding and distant supervision. In *Proceedings of the 18th Extended Semantic Web Conference*, pages 1–16, Heraklion, Greece.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 1511–1517. ijcai.org.

Lingbing Guo, Zequn Sun, Ermei Cao, and Wei Hu. 2018. Recurrent skipping networks for entity alignment. *CoRR*, abs/1811.02318.

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 2505–2514. PMLR.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. 2019. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*, pages 1709–1719. ACM.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2020. Knowledge graphs. *CoRR*, abs/2003.02320.

Ian Horrocks. 2008. Ontologies and the semantic web. *Communications of the ACM*, 51(12):58–67.

Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. 2012. Large-scale interactive ontology matching: algorithms and implementation. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 444–449.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1979, Brussels, Belgium. Association for Computational Linguistics.

Summaya Mumtaz and M. Giese. 2020. Frequency-based vs. knowledge-based similarity measures for categorical data. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

Shichao Pei, Lu Yu, Robert Hoehndorf, and Xiangliang Zhang. 2019. Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In *The World Wide Web Conference, WWW 2019*, pages 3130–3136. ACM.

Zhiyuan Qi, Ziheng Zhang, Jiaoyan Chen, Xi Chen, Yuejia Xiang, Ningyu Zhang, and Yefeng Zheng. 2021. Unsupervised knowledge graph alignment by probabilistic reasoning and semantic embedding. *arXiv preprint arXiv:2105.05596*.

Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. 2011. PARIS: Probabilistic alignment of relations, instances, and schema. *Proc. VLDB Endow.*, 5(3):157–168.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, volume 10587 of *Lecture Notes in Computer Science*, pages 628–644. Springer.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 4396–4402. ijcai.org.

Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. 2020a. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *Proceedings of the Conference on Artificial Intelligence*, pages 222–229. AAAI Press.

Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. 2020b. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proc. VLDB Endow.*, 13(11):2326–2340.

Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. 2019. Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the Conference on Artificial Intelligence*, pages 297–304. AAAI Press.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357, Brussels, Belgium. Association for Computational Linguistics.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 5278–5284. ijcai.org.

Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. 2019. Multi-view knowledge graph embedding for entity alignment. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 5429–5435. ijcai.org.

Ziheng Zhang, Jiaoyan Chen, Xi Chen, Hualuo Liu, Bo Liu, YueJia Xiang, and Yefeng Zheng. 2020. An industry evaluation of embedding-based entity alignment. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 179–189, Online. International Committee on Computational Linguistics.

Xiaofei Zhou, Qiannan Zhu, Ping Liu, and Li Guo. 2017. Learning knowledge embeddings by combining limit-based scoring loss. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017*, pages 1009–1018. ACM.

Appendix

A More Benchmark Details

Table 6 shows the full statistics of all the benchmarks used in the experiments, including four cross-lingual benchmarks and two cross-KG benchmarks from OpenEA (Sun et al., 2020b), and one recent industrial cross-KG benchmark from Zhang et al. (2020). We report the benchmark statistics from three perspectives: the ontologies, the KGs and the membership relationships. The benchmarks are divided as previous work that 20%, 10% and 70% of all the entity mappings are used as training, validation and test sets, respectively (Sun et al., 2020b; Zhang et al., 2020).

B More Implementation Details

Our experiments are conducted on a workstation with an Intel Xeon E5 2.40 GHz CPU, 128 GB memory, an NVIDIA Tesla M40 GPU, and CentOS 7.2. Search spaces of the hyperparameters used in OntoEA are listed in Table 5. We apply a grid search strategy for the hyperparameter configuration that performs the best on the validation set with respect to mean reciprocal rank (MRR).

Table 5: Search spaces of the hyperparameters.

Hyperparameters	Search spaces
learning rate	{1e-3, 5e-3, 1e-2, 5e-2}
batch size (*entity*)	{4000, 4500, 5000}
batch size (*ontology*)	{32, 64, 128}
γ_x^1 in Eqs. (1) (2) (4) *	{0.01, 0.02, 0.03}
γ_x^2 in Eqs. (1) (2) (4) *	{1.0, 2.0, 3.0}
α_x in Eqs. (1) (2) (4) *	{0.1, 0.2, 0.3}
λ_1 in Eq. (6)	{0, 1, 2, 3, 4, 5}
λ_2 in Eq. (6)	{0, 1, 2, 3, 4, 5}
λ_3 in Eq. (6)	{0, 1, 2, 3, 4, 5}
β in Eq. (7)	{0.3, 0.4, 0.5, 0.6, 0.7}

* where $x \in \{e, o, m\}$.

C Ontology Alignment Methods

In this section, we present the details of the two ontology alignment methods: *manual annotation* and *alignment system*.

C.1 Manual Annotation

We refer to crowdsourcing for manual annotation of the class mappings by the following steps which are presented with the example of the DBpedia ontology and the Wikidata ontology in the D-W benchmarks. First, we recruit five volunteers with at least undergraduate education as the annotators and provide them the two to-be-aligned ontologies. The DBpedia ontology (O_D) contains a total of 755 classes while the Wikidata ontology (O_W) contains 695 classes. Second, for each class in O_D, denoted as c_D, each annotator is required to consider as more classes in O_W as possible and find the equivalent class mapping (c_D, c_W), where c_W denotes a class in O_W. Note that the annotators are allowed to utilize her/his knowledge and accelerate the annotation by searching for class $c_W \in O_W$, using some friendly ontology management and accessing software such as Protégé. After all five annotators finish the work, only the class mappings that are labelled by more than three annotators are accepted as the final class mappings, denoted as M_C.

With the class mappings, we need to further merge the two ontologies as one shared ontology which can then be processed by OntoEA. To this end, our solution is to build new membership relationships between the entities of the Wikidata KG and the classes of the DBpedia ontology. For each original membership relationship within the Wikidata KG and ontology, we replace its class c_W with c_D if $(c_D, c_W) \in M_C$, or with the root class *owl:Thing* otherwise.

C.2 Alignment System

Algorithm 1 Alignment System Method

```
Require: Ontologies $O_D$ and $O_W$, threshold $\lambda$
Build directed graphs $D_D$ and $D_W$ from $O_D$ and $O_W$, respectively, with the relation of rdfs:subClassOf
Obtain equivalent class pairs $P_C$ from pre-defined owl:equivalentClass in $O_D$ and $O_W$
Generate $D'$ via connecting $D_D$ and $D_W$ with $P_C$
Define $\phi(c_1, c_2)$ as the length of directed walk from $c_1$ to $c_2$ where $c_1, c_2 \in D'$
Initialize empty mapping sets $M_C$ and $M'_C$
for each class $c_W$ in $O_W$ do
    Route $D'$ starting from $c_W$ and output routed class set $S = \{c\}$ where $c \in D'$
    if $S = \emptyset$ then
        Append $(c_W, owl:Thing)$ to $M_C$
    else
        Find a class $c$ in $S$ such that for any $c'$ in $S$ $\phi(c, owl:Thing) \geq \phi(c', owl:Thing)$
        Append $(c_W, c)$ to $M_C$
    end if
end for
Perform PARIS on $O_D$ and $O_W$
Append $M'_C$ from PARIS with threshold $\lambda$
for each class mapping $(c_w, c_d)$ in $M_C$ do
    if $(c_w, c'_d) \in M'_C$ and $c'_d \neq c_d$ then
        Remove $(c_w, c_d)$ from $M_C$
        Append $(c_w, c'_d)$ to $M_C$
    end if
end for
return $M_C$
```

Table 6: Full statistics of all the benchmarks.

	Dataset	KG	Ontology [b]		Knowledge Graph				Membership [c]	
			#Cls.	#Sub tr.	#Rel.	#Attr.	#Rel tr.	#Attr tr.	#Links	#Roots
share-O [a]	EN-FR-15K-V1	EN	189	755	267	308	47,334	73,121	15,000	639
		FR			210	404	40,864	67,167		
	EN-FR-15K-V2	EN	104	755	193	189	96,318	66,899	15,000	533
		FR			166	221	80,112	68,779		
	EN-DE-15K-V1	EN	175	755	215	286	47,676	83,775	15,000	155
		DE			131	194	50,419	156,150		
	EN-DE-15K-V2	EN	86	755	169	171	84,867	81,998	15,000	165
		DE			96	116	92,632	186,335		
	MED-BBK-9K	MED	11	10	32	19	158,357	11,467	9,162	86
		BBK	11	10	20	21	50,307	44,987	9,162	3,362
not-share-O	D-W-15K-V1	DB	172	755	248	342	38,265	68,258	15,000	306
		WK	140	695	269	649	42,746	138,246	15,000	342
	D-W-15K-V2	DB	71	755	167	175	73,983	66,813	15,000	463
		WK	68	695	121	457	83,365	175,686	15,000	418

[a] "*share-O*" means the to-be-aligned KGs share one ontology while "*not-share-O*" means the to-be-aligned KGs have different ontologies.

[b] "#Cls." denotes the number of classes and "#Trs." denotes the number of *rdfs:subClassOf* relation triples.

[c] "#Roots" denotes the number of entities that have no *rdf:type* property and are linked to the root class.

Table 7: Overall results of OntoEA and the baselines on the EN-DE benchmarks.

	Models	EN-DE-15K-V1			EN-DE-15K-V2		
		H@1	H@5	MRR	H@1	H@5	MRR
w/o SI	MTransE	.307	.518	.407	.193	.352	.274
	JAPE	.288	.512	.394	.167	.329	.250
	SEA	.530	.718	.617	.606	.779	.687
	GCNAlign	.481	.679	.571	.534	.717	.618
	BootEA	.675	.820	.740	.833	.912	.869
	RSN4EA	.587	.752	.662	.791	.890	.837
	AliNet	.243	.353	.295	.169	.247	.277
	OntoEA	**.742**	**.897**	**.812**	**.864**	**.955**	**.905**
Improv. best %		9.9	9.4	9.7	3.7	4.7	4.1
w/ SI	AttrE	.517	.687	.597	.650	.816	.726
	MultiKE	.756	.809	.782	.755	.813	.784
	RDGCN	.830	.895	.859	.833	.891	.860
	OntoEA	**.850**	**.906**	**.877**	**.914**	**.977**	**.942**
Improv. best %		2.4	1.2	2.1	9.7	9.7	9.5

The automatic method based on the ontology alignment system PARIS and some ad-hoc pre-processing and post-processing is shown in Algorithm 1. The pre-processing from Line 1 to Line 13 obtains class mappings via finding the most fine-grained mappings (i.e., the farthest class node in the directed graph). Since PARIS generally outputs reliable class mappings, the post-processing from Line 16 to Line 21 updates class mappings with those output from PARIS.

In the implementation we set PARIS hyperparameter λ to 0.4. The *owl:equivalentClass* relationship across the DBpedia ontology and the Wikidata ontology is acquired via DBpedia and Wikidata SPARQL endpoints. After getting M_C, we merge the two ontologies as introduced in the method of manual annotation.

D More Experimental Results

Table 7 reports the overall results on the cross-lingual benchmarks EN-DE-15K-V1 and EN-DE-15K-V2. We have similar findings in these two EN-DE benchmarks as in the EN-FR benchmarks that OntoEA consistently outperform baselines, w/o SI and w/ SI. On EN-DE-15K-V1, OntoEA generally outperforms competitive models by up to 9.9% in H@1 and the performance gap is larger in the models w/o SI than in the models w/ SI. On the dense benchmark (V2), OntoEA shows limited performance gain (about 4%) compared with the models w/o SI but much larger performance gain (more than 9%) compared with the models w/ SI.

Learning Algebraic Recombination for Compositional Generalization

Chenyao Liu[1*] **Shengnan An**[2*] **Zeqi Lin**[3†] **Qian Liu**[4*] **Bei Chen**[3]
Jian-Guang LOU[3] **Lijie Wen**[1†] **Nanning Zheng**[2] **Dongmei Zhang**[3]
[1] School of Software, Tsinghua University [2] Xi'an Jiaotong University
[3] Microsoft Research Asia [4] Beihang University
{liucy19@mails, wenlj@}.tsinghua.edu.cn
{an1006634493@stu, nnzheng@mail}.xjtu.edu.cn
{Zeqi.Lin, beichen, jlou, dongmeiz}@microsoft.com
qian.liu@buaa.edu.cn

Abstract

Neural sequence models exhibit limited *compositional generalization* ability in semantic parsing tasks. Compositional generalization requires *algebraic recombination*, i.e., dynamically recombining structured expressions in a recursive manner. However, most previous studies mainly concentrate on recombining lexical units, which is an important but not sufficient part of algebraic recombination. In this paper, we propose LEAR, an end-to-end neural model to learn algebraic recombination for compositional generalization. The key insight is to model the semantic parsing task as a homomorphism between a latent syntactic algebra and a semantic algebra, thus encouraging algebraic recombination. Specifically, we learn two modules jointly: a Composer for producing latent syntax, and an Interpreter for assigning semantic operations. Experiments on two realistic and comprehensive compositional generalization benchmarks demonstrate the effectiveness of our model. The source code is publicly available at https://github.com/microsoft/ContextualSP.

1 Introduction

The principle of compositionality is an essential property of language: the meaning of a complex expression is fully determined by its structure and the meanings of its constituents (Pelletier, 2003; Szabó, 2004). Based on this principle, human intelligence exhibits *compositional generalization* — the algebraic capability to understand and produce a potentially infinite number of novel expressions by dynamically recombining known components (Chomsky, 1957; Fodor and Pylyshyn, 1988; Fodor and Lepore, 2002). For example, people who know the meaning of "*John teaches the girl*" and "*Tom's daughter*" must know the meaning of "*Tom teaches John's daughter's daughter*" (Figure 1a), even though they have never seen such complex sentences before.

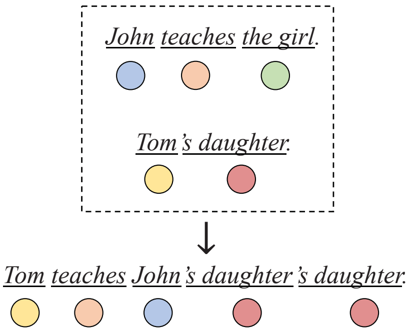

(a) Compositional generalization requires **algebraic recombination**, i.e., dynamically recombining structured expressions in a recursive manner.

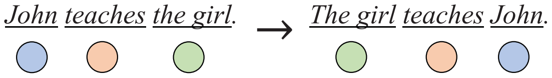

(b) Most previous studies mainly concentrate on recombining lexical units, which is an important but not sufficient part of algebraic recombination.

Figure 1: Compositional generalization.

In recent years, there has been accumulating evidence that end-to-end deep learning models lack such ability in semantic parsing (i.e., translating natural language expressions to machine interpretable semantic meanings) tasks (Lake and Baroni, 2018; Keysers et al., 2019; Kim and Linzen, 2020; Tsarkov et al., 2020).

Compositional generalization requires **algebraic recombination**, i.e., dynamically recombining structured expressions in a recursive manner. In the example in Figure 1a, understanding "*John's daughter's daughter*" is a prerequisite for understanding "*Tom teaches John's daughter's daughter*", while "*John's daughter's daughter*" is also a novel compound expression, which requires recom-

---

[*] Work done during an internship at Microsoft Research. The first two authors contributed equally to this paper.

[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1129–1144
August 1–6, 2021. ©2021 Association for Computational Linguistics

bining "*John*" and "*Tom's daughter*" recursively.

Most previous studies on compositional generalization mainly concentrate on recombining lexical units (e.g., words and phrases) (Lake, 2019; Li et al., 2019; Andreas, 2019; Gordon et al., 2020; Akyürek et al., 2020; Guo et al., 2020a; Russin et al., 2019), of which an example is shown in Figure 1b. This is a necessary part of algebraic recombination, but it is not sufficient for compositional generalization. There have been some studies on algebraic recombination (Liu et al., 2020; Chen et al., 2020). However, they are highly specific to a relative simple domain SCAN (Lake and Baroni, 2018) and can hardly generalize to more complex domains.

In this paper, our main point to achieve algebraic recombination is to **model semantic parsing as a homomorphism between a latent syntactic algebra and a semantic algebra** (Montague, 1970; Marcus, 2019). Based on this formalism, we focus on learning the high-level mapping between latent syntactic operations and semantic operations, rather than the direct mapping between expression instances and semantic meanings.

Motivated by this idea, we propose LEAR (**Le**arning **A**lgebraic **R**ecombination), an end-to-end neural architecture for compositional generalization. LEAR consists of two modules: a *Composer* and an *Interpreter*. Composer learns to model the latent syntactic algebra, thus it can produce the latent syntactic structure of each expression in a bottom-up manner; Interpreter learns to assign semantic operations to syntactic operations, thus we can transform a syntactic tree to the final composed semantic meaning.

Experiments on two realistic and comprehensive compositional generalization benchmarks (*CFQ* (Keysers et al., 2019) and *COGS* (Kim and Linzen, 2020)) demonstrate the effectiveness of our model: CFQ $67.3\% \rightarrow 90.9\%$, COGS $35.0\% \rightarrow 97.7\%$.

2 Compositionality: An Algebraic View

A semantic parsing task aims to learn a meaning-assignment function $m : L \rightarrow M$, where L is the set of (simple and complex) expressions in the language, and M is the set of available semantic meanings for the expressions in L. Many end-to-end deep learning models are built upon this simple and direct formalism, in which the principle of compositionality is not leveraged, thus exhibiting limited compositional generalization.

To address this problem, in this section we put forward the formal statement that "*compositionality requires the existence of a homomorphism between the expressions of a language and the meanings of those expressions*" (Montague, 1970).

Let us consider a language as a partial algebra $\mathbf{L} = \langle L, (f_\gamma)_{\gamma \in \Gamma} \rangle$, where Γ is the set of underlying syntactic (grammar) rules, and we use $f_\gamma : L^k \rightarrow L$ to denote the syntactic operation with a fixed arity k for each $\gamma \in \Gamma$. Note that f_γ is a partial function, which means that we allow f_γ be undefined for certain expressions. Therefore, $\mathbf{L}$ is a partial algebra, and we call it a **syntactic algebra**. In a semantic parsing task, $\mathbf{L}$ is latent, and we need to model it by learning from data.

Consider now $\mathbf{M} = \langle M, G \rangle$, where G are semantic operations upon M. $\mathbf{M}$ is also a partial algebra, and we call it a **semantic algebra**. In a semantic parsing task, we can easily define this algebra (by enumerating all available semantic primitives and semantic operations), since $\mathbf{M}$ is a machine-interpretable formal system.

The key to compositionality is that the meaning-assignment function m should be a homomorphism from $\mathbf{L}$ to $\mathbf{M}$. That is, for each k-ary syntactic operation f_γ in $\mathbf{L}$, there exists a k-ary semantic operation $g_\gamma \in G$ such that whenever $f_\gamma(e_1, ..., e_k)$ is defined,

$$m(f_\gamma(e_1, ..., e_k)) = g_\gamma(m(e_1), ..., m(e_k)). \quad (1)$$

Based on this formal statement, the task of learning the meaning-assignment function m can be transformed as two sub-tasks: (1) learning latent syntax of expressions (i.e., modeling the syntactic algebra $\mathbf{L}$); (2) learning the operation assignment function $(f_\gamma)_{\gamma \in \Gamma} \rightarrow G$.

Learning latent syntax. We need to learn a syntactic parser that can produce the syntactic structure of each given expression. To ensure compositional generalization, there must be an underlying grammar (i.e., Γ), and we hypothesize that Γ is a context-free grammar.

Learning operation assignment. In the syntax tree, for each nonterminal node with k nonterminal children, we assign a k-ary semantic operation to it. This operation assignment entirely depends on the underlying syntactic operation γ of this node.

In semantic parsing tasks, we do not have respective supervision for these two sub-tasks. Therefore, we need to jointly learning these two sub-tasks only from the end-to-end supervision $\mathcal{D} \subset L \times M$.

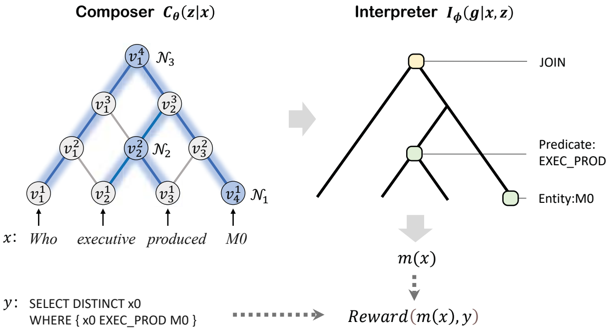

Figure 2: An overview of LEAR: (1) Composer $C_\theta(z|x)$ is a neural network based on latent Tree-LSTM, which produces the latent syntax tree z of input expression x; (2) Interpreter $I_\phi(g|x,z)$ is a neural network that assigns a semantic operation for each nonterminal node in z.

3 Model

We propose a novel end-to-end neural model LEAR (**Le**arning **A**lgebraic **R**ecombination) for compositional generalization in semantic parsing tasks. Figure 2 shows its overall architecture. LEAR consists of two parts: (1) *Composer* $C_\theta(z|x)$, which produces the latent syntax tree z of input expression x; (2) Interpreter $I_\phi(g|x,z)$, which assigns a semantic operation for each nonterminal node in z. θ and ϕ refers to learnable parameters in them respectively. We generate a semantic meaning $m(x)$ according to the predicted z and g in a symbolic manner, then check whether it is semantic equivalent to the ground truth semantic meaning y to produce rewards for optimizing θ and ϕ.

3.1 Composer

We use $x = [x_1, ..., x_T]$ to denote an input expression of length T. Composer $C_\theta(z|x)$ will produce a latent binary tree z given x.

3.1.1 Latent Tree-LSTM

We build up the latent binary tree z in a bottom-up manner based on Tree-LSTM encoder, called latent Tree-LSTM (Choi et al., 2018; Havrylov et al., 2019).

Given the input sequence x of length T, latent Tree-LSTM merges two nodes into one parent node at each merge step, constructing a binary tree after $T-1$ merge steps. The merge process is implemented by selecting the adjacent node pair which has the highest merging score.

At the t-th ($1 \le t < T$) merge step, we have:

$$\hat{i}_t = \underset{1 \le i \le T-t}{\arg\max}\, \mathrm{Linear}(\text{Tree-LSTM}(\mathbf{r}_i^t, \mathbf{r}_{i+1}^t)) \quad (2)$$

Here "Tree-LSTM" is the standard child-sum tree-structured LSTM encoder (Tai et al., 2015). We use v_i^t to denote the i-th cell at layer t (the t-th merge step is determined by the t-th layer), and use $\mathbf{r}_i^t$ to denote the representation of v_i^t:

$$\mathbf{r}_i^1 = \mathrm{Linear}(\mathrm{Emb}(x_i)) \quad (3)$$

$$\underset{t>1}{\mathbf{r}_i^t} = \begin{cases} \mathbf{r}_i^{t-1} & i < \hat{i}_{t-1} \\ \text{Tree-LSTM}(\mathbf{r}_i^{t-1}, \mathbf{r}_{i+1}^{t-1}) & i = \hat{i}_{t-1} \\ \mathbf{r}_{i+1}^{t-1} & i > \hat{i}_{t-1} \end{cases} \quad (4)$$

Then we can obtain a unlabeled binary tree, in which $\{v_1^1, v_2^1, ..., v_T^1\}$ are leaf nodes, and $\{v_{\hat{i}_1}^2, v_{\hat{i}_2}^3 ..., v_{\hat{i}_{T-1}}^T\}$ are non-leaf nodes.

3.1.2 Abstraction by Nonterminal Symbols

As discussed in Section 2, our hypothesis is that the underlying grammar Γ is context-free. Therefore, each syntactic rule $\gamma \in \Gamma$ can be expressed in the form of:

$$A \to B, \qquad A \in \mathcal{N}, B \in (\mathcal{N} \cup \Sigma)^+$$

where $\mathcal{N}$ is a finite set of nonterminals, and Σ is a finite set of terminal symbols.

Abstraction is an essential property of context-free grammar: each compound expression e will be abstracted as a simple nonterminal symbol $\mathcal{N}(e)$, then it can be combined with other expressions to produce more complex expressions, no matter what details e originally has. This setup may benefit the generalizability, thus we want to incorporate it as an inductive bias into our model.

Concretely, we assume that there are at most N latent nonterminals in language $\mathbf{L}$ (i.e., $\mathcal{N} = \{\mathcal{N}_1, ..., \mathcal{N}_N\}$, where N is a hyper-parameter). For each node v_i^t in tree z, we perform a $(N+1)$-class classification:

$$\hat{c}_{v_i^t} = \underset{0 \le c \le N}{\arg\max}\, \mathrm{Linear}(\mathbf{r}_i^t) \quad (5)$$

We assign the nonterminal $\mathcal{N}_{\hat{c}_{v_i^t}}$ to v_i^t when $\hat{c}_{v_i^t} > 0$. The collection of such nonterminal nodes are denoted as V'_z. Then we modify Equation 4:

$$\underset{t>1}{\mathbf{r}_i^t} = \begin{cases} \mathbf{r}_i^{t-1} & i < \hat{i}_{t-1} \\ \text{Tree-LSTM}(\overline{\mathbf{r}}_i^{t-1}, \overline{\mathbf{r}}_{i+1}^{t-1}) & i = \hat{i}_{t-1} \\ \mathbf{r}_{i+1}^{t-1} & i > \hat{i}_{t-1} \end{cases} \quad (6)$$

$$\overline{\mathbf{r}}_i^t = \begin{cases} \text{Linear}(\text{Emb}(\mathcal{N}(v_i^t))) & v_i^t \in V'_z \\ \mathbf{r}_i^t & v_i^t \notin V'_z \end{cases}$$

Equation 6 means that: in nonterminal nodes, the bottom-up message passing will be reduced from $\mathbf{r}_i^t$ to a nonterminal symbol $\mathcal{N}(v_i^t)$, thus mimicking the abstraction setup in context-free grammar.

3.2 Interpreter

For each nonterminal node $v \in V'_z$, Interpreter $I_\phi(g|x, z)$ assigns a semantic operation g_v to it.

We divide nonterminal nodes into two categories: (1) *lexical nodes*, which refer to those containing no any other nonterminal node in the corresponding sub-trees; (2) *algebraic nodes*, which refer to the rest of nonterminal nodes.

Interpreting Lexical Nodes For each lexical node v, Interpreter assigns a semantic primitive (i.e., 0-ary semantic operation) to it. Take the CFQ benchmark as an example: it uses SPARQL queries to annotate semantic meanings, thus semantic primitives in CFQ are entities (e.g., *m.0gwm_wy*), predicates (e.g., *ns:film.director.film*) and attributes (e.g., *ns:people.person.gender m_05zppz*).

We use a classifier to predict the semantic primitive:

$$g_v = \underset{g \in G_{lex}}{\arg\max}\, \text{Linear}(\mathbf{h}_{v,x}) \quad (7)$$

where G_{lex} is the collection of semantic primitives in the domain, and $\mathbf{h}_{v,x}$ is the contextual representation of the span corresponding to v (implemented using Bi-LSTM). *Contextually conditioned variation* is an important phenomenon in language: the meaning of lexical units varies according to the contexts in which they appear (Allwood, 2003). For example, "*editor*" means a predicate "*film.editor.film*" in expression "*Is M0 an editor of M1?*", while it means an attribute "*film.editor*" in expression "*Is M0 an Italian editor?*". This is the reason why we use contextual representation in Equation 7.

Interpreting Algebraic Nodes For each algebraic node v, Interpreter assigns a semantic operation to it. The collection of all possible semantic operations G_{opr} also depends on the domain. Take the CFQ benchmark as an example[1], this domain has two operations (detailed in Table 1): ∧ (conjunction) and JOIN.

Operation	Args[t_1, t_2]→ Result Type	Example
$\wedge(t_1, t_2)$	[P, P]→P	Who [**direct** and **act**] M0?
	[E, E]→E	Who direct [**M0** and **M1**]?
	[A, A]→A	Is M0 an [**Italian female**]?
	[A, E]→E [E, A]→E	Is [**M0** an **Italian female**]?
	[A, P]→P [P, A]→P	Is M0 M3's [**Italian editor**]?
JOIN(t_1, t_2)	[E, P]→E [P, E]→E	Is M0 an [**editor** of **M1**]?
	[A, P]→E [P, A]→E	Who [**marries** an **Italian**]?

Table 1: Semantic operations in CFQ. A/P/E represents Attribute/Predicate/Entity.

We also use a classifier to predict the semantic operation of v:

$$g_v = \underset{g \in G_{opr}}{\arg\max}\, \text{Linear}(\mathbf{r}_v) \quad (8)$$

where $\mathbf{r}_v$ is the latent Tree-LSTM representation of node v (see Equation 6).

In Equation 8, we do not use any contextual information from outside v. This setup is based on the assumption of **semantic locality**: each compound expression should mean the same thing in different contexts.

4 Training

Denote $\tau = \{z, g\}$ as the trajectory produced by our model where z and g are actions produced from Composer and Interpreter, respectively, and $R(\tau)$ as the reward of trajectory τ (elaborated in Sec. 4.1). Using policy gradient (Sutton et al., 2000) with the likelihood ratio trick, our model can be optimized by ascending the following gradient:

$$\nabla \mathcal{J}(\theta, \phi) = \mathbb{E}_{\tau \sim \pi_{\theta,\phi}}\, R(\tau) \nabla \log \pi_{\theta,\phi}(\tau), \quad (9)$$

where θ and ϕ are learnable parameters in Composer and Interpreter respectively and ∇ is the abbreviation of $\nabla_{\theta,\phi}$. Furthermore, the REINFORCE algorithm (Williams, 1992) is leveraged to approximate Eq. 9 and the mean-reward baseline (Weaver and Tao, 2001) is employed to reduce variance.

---

[1] It is not difficult to define G_{lex} and G_{opr} for each domain, as semantic meanings are always machine-interpretable. The semantic operations of another compositional generalization benchmark, COGS, are listed in the Appendix.

4.1 Reward Design

The reward $R(\tau)$ combines two parts as:

$$R(\tau) = \alpha \cdot R_1(\tau) + (1 - \alpha) \cdot R_2(\tau), \quad (10)$$

Logic-based Reward $R_1(\tau)$. We use $m(x)$ and y to denote the predicted semantic meaning and the ground truth semantic meaning respectively. Each semantic meaning can be converted to a conjunctive normal form[2]. We use $S_{m(x)}$ and S_y to denote conjunctive components in $m(x)$ and y, then define $R_1(\tau)$ based on Jaccard similarity (i.e., intersection over union):

$$R_1(\tau) = \text{Jaccard-Sim}(S_{m(x)}, S_y) \quad (11)$$

Primitive-Based Reward $R_2(\tau)$. We use $S'_{m(x)}$ and S'_y to denote semantic primitives ocurred in $m(x)$ and y. Then we define $R_2(\tau)$ as:

$$R_2(\tau) = \text{Jaccard-Sim}(S'_{m(x)}, S'_y) \quad (12)$$

4.2 Reducing Search Space

To reduce the huge search space of τ, we make two constraints as follows.

Parameter Constraint. Consider v, a tree node with $n (n > 0)$ nonterminal children. Composer will never make v a nonterminal node, if no semantic operation has n parameters.

Phrase Table Constraint. Following the strategy proposed in Guo et al. (2020b), we build a "phrase table" consisting of lexical units (i.e., words and phrases) paired with semantic primitives that frequently co-occur with them[3]. Composer will never produce a lexical node outside of this table, and Interpreter will use this table to restrict candidates in Equation 7.

4.3 Curriculum Learning

To help the model converge better, we use a simple curriculum learning (Bengio et al., 2009) strategy to train the model. Specifically, we first train the model on samples of input length less than a cut-off N_{CL}, then further train it on the full train set.

CFQ

x "Did a male film director edit and direct M0?"

y
```
SELECT count ( * ) WHERE {
    ?x0 ns:film.director.film M0 .
    ?x0 ns:film.editor.film M0 .
    ?x0 ns:people.person.gender m_05zppz }
```

COGS

x "Charlotte was given the cake on a table."

y
```
cake(x_4) ; give.recipient (x_2, Charlotte)
AND give.theme(x_2,x_4)
AND cake.nmod.on(x_4, x_7)
AND table(x_7)
```

Figure 3: Examples of CFQ and COGS.

Statistics	CFQ	COGS
Train Size	95,743	24,155
Dev Size	11,968	3,000
Test Size	11,968	21,000
Vocab Size	96	740
Avg Input Len (Train/Test)	13.5/15.1	7.5/9.8
Avg Output Len (Train/Test)	27.7/34.0	43.6/67.6
Input Pattern Coverage[a]	0.022	0.783
Output Pattern Coverage	0.045	0.782

Table 2: Dataset statistics.

[a] Input/output pattern coverage is the percentage of test x/y whose patterns occur in the train data. Output patterns are determined by anonymizing semantic primitives, and input patterns are determined by anonymizing their lexical units.

5 Experimental Setup

Benchmarks. We mainly evaluate LEAR on *CFQ* (Keysers et al., 2019) and *COGS* (Kim and Linzen, 2020), two comprehensive and realistic benchmarks for measuring compositional generalization. They use different semantic formulations: CFQ uses SPARQL queries, and COGS uses logical queries (Figure 3 shows examples of them). We list dataset statistics in Table 2. The input/output pattern coverage indicates that: CFQ mainly measures the algebraic recombination ability, while COGS measures both lexical recombination ($\sim 78\%$) and algebraic recombination ($\sim 22\%$).

In addition to these two compositional generalization benchmarks in which utterances are synthesized by formal grammars, we also evaluate LEAR on *GEO* (Zelle and Mooney, 1996), a widely used semantic parsing benchmark, to see whether LEAR can generalize to utterances written by real users. We use the variable-free FunQL (Kate et al., 2005) as the semantic formalism, and we follow the compositional train/test split (Finegan-Dollak et al., 2018) to evaluate compositional generalization.

Baselines. For CFQ, we consider 3 groups of models as our baselines: (1) sequence-to-sequence mod-

---

[2] For example, the semantic meaning of "*Who directed and edited M_0 's prequel and M_1?*" can be converted to a conjunctive normal form with four components: "$x_0 \cdot$ DIRECT $\cdot$ $x_1 \cdot$ PREQUEL $\cdot$ M_0", "$x_0 \cdot$ EDIT $\cdot$ $x_1 \cdot$ PREQUEL $\cdot$ M_0", "$x_0 \cdot$ DIRECT $\cdot$ M_1", and "$x_0 \cdot$ EDIT $\cdot$ M_1".

[3] Mainly based on statistical word alignment technique in machine translation, detailed in the Appendix.

Models	**MCD-MEAN**	**MCD1**	**MCD2**	**MCD3**
LSTM+Attention (Keysers et al., 2019)	14.9±1.1	28.9±1.8	5.0±0.8	10.8±0.6
Transformer (Keysers et al., 2019)	17.9±0.9	34.9±1.1	8.2±0.3	10.6±1.1
Universal Transformer (Keysers et al., 2019)	18.9±1.4	37.4±2.2	8.1±1.6	11.3±0.3
Evolved Transformer (Furrer et al., 2020)	20.8±0.7	42.4±1.0	9.3±0.8	10.8±0.2
T5-11B (Furrer et al., 2020)	40.9±4.3	61.4±4.8	30.1±2.2	31.2±5.7
T5-11B-mod (Furrer et al., 2020)	42.1±9.1	61.6±12.4	31.3±12.8	33.3±2.3
Neural Shuffle Exchange (Furrer et al., 2020)	2.8±0.3	5.1±0.4	0.9±0.1	2.3±0.3
CGPS (Furrer et al., 2020; Li et al., 2019)	7.1±1.8	13.2±3.9	1.6±0.8	6.6±0.6
HPD (Guo et al., 2020b)	67.3±4.1	72.0±7.5	66.1±6.4	63.9±5.7
LEAR	**90.9±1.2**	**91.7±1.0**	**89.2±1.9**	**91.7±0.6**
w/o Abstraction	85.4±4.5	88.4±1.6	80.0±11	87.9±0.8
w/o Semantic locality	87.9±2.7	89.8±1.7	87.3±1.8	86.5±4.6
w/o Primitive-based reward	85.3±7.8	77.0±19	89.2±2.2	89.7±2.1
w/o Curriculum learning	71.9±15.4	59.7±23	77.2±13.5	78.8±9.6
w/o Tree-LSTM	30.4±3.2	40.1±1.9	25.6±6.1	25.4±1.8

Table 3: Accuracy on three splits (MCD1/MCD2/MCD3) of CFQ benchmark.

els based on deep encoder-decoder architecture, including LSTM+Attention (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014), Transformer (Vaswani et al., 2017), Universal Transformer (Dehghani et al., 2018) and Evolved Transformer (So et al., 2019); (2) deep models with large pretrained encoder, such as T5 (Raffel et al., 2019); (3) Models that are specially designed for compositional generalization, which include Neural Shuffle Exchange Network (Freivalds et al., 2019), CGPS (Li et al., 2019), and state-of-the-art model HPD (Guo et al., 2020b). For COGS, we quote the baseline results in the original paper (Kim and Linzen, 2020). For GEO, we take the baseline results reported by Herzig and Berant (2020), and also compare with two specially designed methods: SpanBasedSP (Herzig and Berant, 2020) and PDE (Guo et al., 2020c).

Evaluation Metric. We use accuracy as the evaluation metric, i.e., the percentage test samples of which the predicted semantic meaning $m(x)$ is semantically equivalent to the ground truth y.

Hyper-Parameters. We set $N = 3/2/3$ (the number of nonterminal symbols), and $\alpha = 0.5/1.0/0.9$ for CFQ/COGS/GEO respectively. In CFQ, the curriculum cut-off N_{CL} is set to 11, as we statistically find that this is the smallest curriculum that contains the complete vocabulary. We do not apply curriculum learning strategy to COGS and GEO, as LEAR can work well without curriculum learning in both benchmarks. Learnable parameters (θ and ϕ) are optimized with AdaDelta (Zeiler, 2012), and the setting of learning rate is discussed in Section 6.1. We take the model that performs best on the validation set for testing, and all results are obtained by averaging over 5 runs with different random seeds. See Appendix for more implementation details.

Model	**Acc**
Transformer (Kim and Linzen, 2020)	35 ± 6
LSTM (Bi) (Kim and Linzen, 2020)	16 ± 8
LSTM (Uni) (Kim and Linzen, 2020)	32 ± 6
LEAR	**97.7 ± 0.7**
w/o Abstraction	94.5 ± 2.8
w/o Semantic locality	94.0 ± 3.6
w/o Tree-LSTM	80.7 ± 4.3

Table 4: Accuracy on COGS benchmark.

Model	**Acc**
Seq2Seq (Herzig and Berant, 2020)	46.0
BERT2Seq (Herzig and Berant, 2020)	49.6
GRAMMAR (Herzig and Berant, 2020)	54.0
PDE (Guo et al., 2020c)	81.2
SpanBasedSP (Herzig and Berant, 2020)	82.2
LEAR	**84.1**

Table 5: Accuracy on GEO benchmark.

6 Results and Discussion

Table 3 shows average accuracy and 95% confidence intervals on three splits of CFQ. LEAR achieves an average accuracy of 90.9% on these three splits, outperforming all baselines by a large margin. We list some observations as follows.

Methods for lexical recombination cannot generalize to algebraic recombination. Many methods for compositional generalization have been proved effective for lexical recombination. Neural Shuffle Exchange and CGPS are two representatives of them. However, experimental results show that they cannot generalize to CFQ, which focus on algebraic recombination.

Knowledge of semantics is important for compositional generalization. Seq2seq models show poor compositional generalization ability ($\sim 20\%$). Pre-training helps a lot ($\sim 20\% \rightarrow\sim 40\%$), but still not satisfying. HPD and LEAR incorporate knowledge of semantics (i.e., semantic operations) into the models, rather than simply model semantic meanings as sequences. This brings large profit.

Exploring latent compositional structure in a bottom-up manner is key to compositional generalization. HPD uses LSTM to encode the input expressions, while LEAR uses latent Tree-LSTM, which explicitly explores latent compositional structure of expressions. This is the key to the large accuracy profit ($67.3\% \rightarrow 90.9\%$).

Table 4 shows the results on COGS benchmark. It proves that LEAR can well generalize to domains which use different semantic formalisms, by specifying domain-specific G_{lex} (semantic primitives) and G_{opr} (semantic operations). Table 5 shows the results on GEO benchmark. It proves that LEAR can well generalize to utterances written by real users (i.e., non-synthetic utterances).

6.1 Ablation Study

Table 3 and 4 also report results of some ablation models. Our observations are as follows.

Abstraction by nonterminal symbols brings profit. We use "*w/o abstraction*" to denote the ablation model in which Equation 6 is disabled. This ablation leads to $5.5\%/3.2\%$ accuracy drop on CFQ/COGS.

Incorporating semantic locality into the model brings profit. We use "*w/o semantic locality*" to denote the ablation model in which a Bi-LSTM layer is added before the latent Tree-LSTM. This ablation leads to $3.0\%/3.7\%$ accuracy drop on CFQ/COGS.

Tree-LSTM contributes significantly to compositional generalization. In the ablation "*w/o Tree-LSTM*", we replace the Tree-LSTM encoder with a span-based encoder, in which each span is represented by concatenating its start and end LSTM representations (similar to Herzig and Berant (2020)). In Table 3 and 4, we can see that span-based encoder severely affects the performance and even much worse than the results of "*w/o abstraction*" and "*w/o semantic locality*". This ablation hints that Tree-LSTM is the main inductive bias of compositionality in our model.

Ratio	MCD-MEAN	MCD1	MCD2	MCD3
1:1:1	87.4±7.1	91.5±2.1	**89.4±2.3**	81.2±17
1:0.5:0.1	**90.9±1.2**	**91.7±1.0**	89.2±1.9	**91.7±0.6**
1:0.1:0.1	86.7±3.9	89.4±1.6	85.8±2.7	84.9±7.5

Table 6: Results of different learning rate ratios of lexical Interpreter, Composer, and algebraic Interpreter.

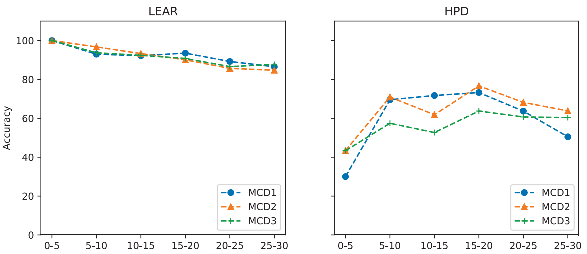

Figure 4: Performance by input length.

Primitive-based reward helps the model converge better. The ablation "*w/o primitive-based reward*" leads to 5.6% accuracy drop on CFQ, and the model variance has become much larger. The key insight is: primitive-based reward guides the model to interpret polysemous lexical units more effectively, thus helping the model converge better.

Curriculum learning helps the model converge better. The ablation "*w/o curriculum learning*" leads to 19% accuracy drop on CFQ, and the model variance has become much larger. This indicates the importance of curriculum learning. On COGS, LEAR performs well without curriculum learning. We speculate that there are two main reasons: (1) expressions of COGS is much shorter than CFQ; (2) the input/output pattern coverage of COGS is much higher than CFQ.

Higher component with smaller learning rate. Inspired by the differential update strategy used in Liu et al. (2020)(i.e., the higher level the component is positioned in the model, the slower the parameters in it should be updated), we set three different learning rates to three different components in LEAR (in bottom-up order): lexical Interpreter, Composer, and algebraic Interpreter. We fix the learning rate of lexical Interpreter to 1, and adjust the ratio of the learning rates of Composer and algebraic Interpreter to lexical Interpreter. Table 6 shows the results on CFQ. The hierarchical learning rate setup ($1 : 0.5 : 0.1$) achieves the best performance.

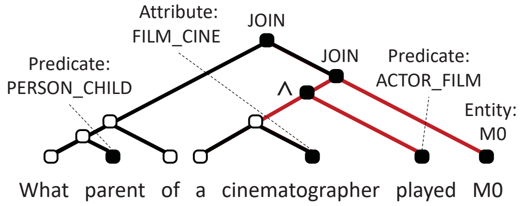

(a) Composer error. A correct syntax tree should compose "*parent of a cinematographer*" as a constituent, while the predicted syntax tree incorrectly composes "*a cinematographer played M0*".

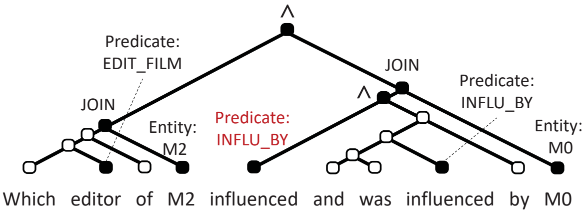

(b) Interpreter error. In this expression, the first "*influenced*" should be assigned a semantic primitive "*influence.influence_node.influenced*", while Interpreter incorrectly assigns "*influence.influence_node.influenced_by*" (abbreviated as "*INFLU_BY*" in this figure) to it.

Figure 5: Two error cases. We use solid nodes to denote predicted nonterminal nodes. Incorrect parts are colored red.

6.2 Closer Analysis

We also conduct closer analysis to the results of LEAR as follows.

6.2.1 Performance by Input Length

Intuitively, understanding longer expressions requires stronger algebraic recombination ability than shorter examples. Therefore, we expect that our model should keep a good and stable performance with the increasing of input length.

Figure 4 shows the performance of LEAR and HPD (the state-of-the-art model on CFQ) under different input lengths. Specifically, test instances are divided into 6 groups by length: $[1, 5], [6, 10], ..., [26, 30]$), and we report accuracy on each group separately. The results indicate that **LEAR has stable high performance for different input lengths**, with only a slow decline as length increases. Even on the group with the longest input length, LEAR can maintain an average 86.3% accuracy across three MCD-splits.

6.2.2 Error Analysis

To understand the source of errors, we take a closer look at the failed test instances of LEAR on CFQ. These failed test instances account for 9.1% of the test dataset. We category them into two error types:

Error Type	MCD1	MCD2	MCD3
CE	45.70%	32.05%	39.83%
IE	54.30%	67.95%	60.17%

Table 7: Distribution of CE (Composer Error) and IE (Interpreter Error).

Composer error (CE), i.e., test cases where Composer produces incorrect syntactic structures (only considering nonterminal nodes). Figure 5a shows an example. As we do not have ground-truth syntactic structures, we determine whether a failed test instance belongs to this category based on handcraft syntactic templates.

Interpreter error (IE), i.e., test cases where Composer produces correct syntactic structures but Interpreter assigns one or more incorrect semantic primitives or operations. Figure 5b shows an example, which contains an incorrect semantic primitive assignment.

Table 7 shows the distribution of these two error types. On average, 39.19% of failed instances are composer errors, and the remaining 60.81% are interpreter errors.

6.3 Limitations

Our approach is implicitly build upon the assumption of **primitive alignment**, that is, each primitive in the meaning representation can align to at least one span in the utterance. This assumption holds in most cases of various semantic parsing tasks, including CFQ, COGS, and GEO. However, for robustness and generalizability, we also need to consider cases that do not meet this assumption. For example, consider this utterance "*Obama's brother*", of which the corresponding meaning representation is "$Slibing(People[Obama]) \wedge Gender[Male]$". Neither "$Slibing$" nor "$Gender[Male]$" can align to a span in the utterance, as the composed meaning of them is expressed by a single word ("*brother*"). Therefore, LEAR is more suitable for formalisms where primitives can better align to natural language.

In addition, while our approach is general for various semantic parsing tasks, the collection of semantic operations needs to be redesigned for each task. We need to ensure that these semantic operations are k-ary projections (as described in Section 2), and all the meaning representations are covered by the operations collection. This is tractable, but still requires some efforts from domain experts.

7 Related Work

7.1 Compositional Generalization

Recently, exploring compositional generalization (CG) on neural networks has attracted large attention in NLP community. For SCAN (Lake and Baroni, 2018), the first benchmark to test CG on seq2seq models, many solutions have been proposed, which can be classified into two tracks: data augmentation (Andreas, 2019; Akyürek et al., 2020; Guo et al., 2020a) and specialized architecture (Lake, 2019; Li et al., 2019; Gordon et al., 2020). However, most of these works only focus on lexical recombination. Some works on SCAN have stepped towards algebraic recombination (Liu et al., 2020; Chen et al., 2020), but they do not generalize well to other tasks such as CFQ (Keysers et al., 2019) and COGS (Kim and Linzen, 2020).

Before our work, there is no satisfactory solution on CFQ and COGS. Previous works on CFQ demonstrated that MLM pre-training (Furrer et al., 2020) and iterative back-translation (Guo et al., 2020d) can improve traditional seq2seq models. HPD (Guo et al., 2020b), the state-of-the-art solution before ours, was shown to be effective on CFQ, but still far from satisfactory. As for COGS, there is no solution to it to the best of our knowledge.

7.2 Compositional Semantic Parsing

In contrast to neural semantic parsing models which are mostly constructed under a fully seq2seq paradigm, compositional semantic parsing models predict partial meaning representations and compose them to produce a full meaning representation in a bottom-up manner (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2012; Liang et al., 2013; Berant et al., 2013; Berant and Liang, 2015; Pasupat and Liang, 2015; Herzig and Berant, 2020). Our model takes the advantage of compositional semantic parsing, without requiring any handcraft lexicon or syntactic rule.

7.3 Unsupervised Parsing

Unsupervised parsing (or grammar induction) trains syntax-dependent models to produce syntactic trees of natural language expressions without direct syntactic annotation (Klein and Manning, 2002; Bod, 2006; Ponvert et al., 2011; Pate and Johnson, 2016; Shen et al., 2018; Kim et al., 2019; Drozdov et al., 2020). Comparing to them, our model learns both syntax and semantics jointly.

8 Conclusion

In this paper, we introduce LEAR, a novel end-to-end neural model for compositional generalization in semantic parsing tasks. Our contribution is 4-fold: (1) LEAR focuses on algebraic recombination, thus it exhibits stronger compositional generalization ability than previous methods that focus on simpler lexical recombination. (2) We model the semantic parsing task as a homomorphism between two partial algebras, thus encouraging algebraic recombination. (3) We propose the model architecture of LEAR, which consists of a Composer (to learn latent syntax) and an Interpreter (to learn operation assignments). (4) Experiments on two realistic and comprehensive compositional generalization benchmarks demonstrate the effectiveness of our model.

Acknowledgments

The work was supported by the National Key Research and Development Program of China (No. 2019YFB1704003), the National Nature Science Foundation of China (No. 71690231), Tsinghua BNRist and Beijing Key Laboratory of Industrial Bigdata System and Application.

Ethical Consideration

The experiments in this paper are conducted on existing datasets. We describe the model architecture and training method in detail, and provide more explanations in the supplemental materials. All the data and code will be released with the paper. The resources required to reproduce the experiments is a Tesla P100 GPU, and for COGS benchmark even one CPU is sufficient. Since the compositional generalization ability explored in this paper is a fundamental problem of artificial intelligence and has not yet involved real applications, there are no social consequences or ethical issues.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resample data for compositional generalization.

Jens Allwood. 2003. Meaning potentials and context: Some consequences for the analysis of variation in meaning. *Cognitive approaches to lexical semantics*, pages 29–66.

Jacob Andreas. 2019. Good-enough compositional data augmentation. *CoRR*, abs/1904.09545.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Noam Chomsky. 1957. Syntactic structures (the hague: Mouton, 1957). *Review of Verbal Behavior by BF Skinner, Language*, 35:26–58.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with s-diora: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4832–4845.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

J. A. Fodor and Z. W. Pylyshyn. 1988. Connectionism and cognitive architecture - a critical analysis. *Cognition*, 28(1-2):3–71.

Jerry A Fodor and Ernest Lepore. 2002. *The compositionality papers*. Oxford University Press.

Kārlis Freivalds, Emīls Ozoliņš, and Agris Šostaks. 2019. Neural shuffle-exchange networks–sequence processing in o (n log n) time. *arXiv preprint arXiv:1907.07897*.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Demi Guo, Yoon Kim, and Alexander M. Rush. 2020a. Sequence-level mixed sample data augmentation.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020b. Hierarchical poset decoding for compositional generalization in language. *arXiv preprint arXiv:2010.07792*.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020c. Iterative utterance segmentation for neural semantic parsing. *arXiv preprint arXiv:2012.07019*.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2020d. Revisiting iterative back-translation from the perspective of compositional generalization.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative learning of disjoint syntax and semantics. *arXiv preprint arXiv:1902.09393*.

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *AAAI*, volume 5, pages 1062–1068.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019. Compound probabilistic context-free grammars for grammar induction. *arXiv preprint arXiv:1906.10225*.

Dan Klein and Christopher D Manning. 2002. Natural language grammar induction using a constituent-context model. *Advances in neural information processing systems*, 1:35–42.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems 32*, pages 9791–9801. Curran Associates, Inc.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. *arXiv preprint arXiv:1910.02612*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. Compositional generalization by learning analytical expressions.

Gary F Marcus. 2019. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press.

Richard Montague. 1970. Universal grammar. *1974*, pages 222–46.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

John K Pate and Mark Johnson. 2016. Grammar induction from (lots of) words alone. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 23–32.

Francis Jeffry Pelletier. 2003. Context dependence and compositionality. *Mind & Language*, 18(2):148–161.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *CoRR*, abs/1904.09708.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR.

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press.

Zoltán Gendler Szabó. 2004. Compositionality.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566. The Association for Computer Linguistics.

Dmitry Tsarkov, Tibor Tihon, Nathan Scales, Nikola Momchev, Danila Sinopalnikov, and Nathanael Schärli. 2020. *-cfq: Analyzing the scalability of machine learning on a compositional task. *arXiv preprint arXiv:2012.08266*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 538–545. Morgan Kaufmann.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

This is the Appendix for the paper: "Learning Algebraic Recombination for Compositional Generalization".

A Semantic Operations in COGS

The semantic primitives used in COGS benchmark are entities (e.g., *Emma* and *cat(x_1)*), predicates (e.g., *eat*) and propositions (e.g., *eat.agent(x_1, Emma)*). The semantic operations in COGS are listed in Table 8.

The operations with "$^{-1}$" (e.g., ON^{-1}) are right-to-left operations (e.g., ON^{-1}(cake, table)→table.ON.cake) while the operations without "-1" represent the left-to-right operations (e.g., ON(cake, table)→cake.ON.table). For operation FillFrame, the entity in its arguments will be filled into predicate/proposition as an AGENT, THEME or RECIPIENT, which is decided by model.

B Semantic Operations in GEO and Post-process

The semantic primitives used in GEO benchmark are entities (e.g., *var0*), predicates (e.g., *state()*) and propositions (e.g., *state(var0)*). The semantic operations in GEO are listed in Table 9.

To fit the FunQL formalism, we design two post-processing rules for the final semantics generated by the model. First, if the final semantic is a predicate (not a proposition), it will be converted in to a proposition by filling the entity *all*. Second, the predicate *most* will be shifted forward two positions in the final semantics.

C Policy Gradient and Differential Update

In this section, we will show more details about the formulation of our RL training based on policy gradient and how to use differential update strategy on it.

Denoting $\tau = \{z, g\}$ as the trajectory of our model where z and g are actions (or called results) produced from Composer and Interpreter, respectively, and $R(\tau)$ as the reward of a trajectory τ (elaborated in Sec. 4.1), the training objective of our model is to maximize the expectation of rewards as:

$$\max_{\theta,\phi} \mathcal{J}(\theta, \phi) = \max_{\theta,\phi} \mathbb{E}_{\tau\sim\pi_{\theta,\phi}} R(\tau), \tag{13}$$

where $\pi_{\theta,\phi}$ is the policy of the whole model θ and ϕ are the parameters in Composer and Interpreter, respectively. Applying the likelihood ratio trick, θ and ϕ can be optimized by ascending the following gradient:

$$\nabla \mathcal{J}(\theta, \phi) = \mathbb{E}_{\tau\sim\pi_{\theta,\phi}} R(\tau) \nabla \log \pi_{\theta,\phi}(\tau),$$

which is same with Eq. 9.

As described in Sec. 3 that the interpreting process can be divided into two stages: interpreting lexical nodes and interpreting algebraic nodes, the action g can also be split as the semantic primitives of lexical nodes g_l and the semantic operations of algebraic nodes g_a. In our implement, we utilize two independent neural modules for interpreting lexical nodes and interpreting algebraic nodes, with parameters ϕ_l and ϕ_a respectively. Therefore, $\nabla \log \pi_{\theta,\phi}(\tau)$ in Eq. 9 can be expanded via the chain rule as:

$$\begin{aligned} \nabla \log \pi_{\theta,\phi}(\tau) = & \nabla \log \pi_{\theta}(z|x) + \\ & \nabla \log \pi_{\phi_l}(g_l|x, z) + \\ & \nabla \log \pi_{\phi_a}(g_a|x, z, g_l). \end{aligned} \tag{14}$$

With Eq. 14, we can set different learning rates:

$$\begin{aligned} \theta &\leftarrow \theta + \alpha \cdot \mathbb{E}\, R(\tau) \nabla \log \pi_{\theta}(z|x), \\ \phi_l &\leftarrow \phi_l + \beta \cdot \mathbb{E}\, R(\tau) \nabla \log \pi_{\phi_l}(g_l|x, z), \\ \phi_a &\leftarrow \phi_a + \gamma \cdot \mathbb{E}\, R(\tau) \nabla \log \pi_{\phi_a}(g_a|x, z, g_l). \end{aligned} \tag{15}$$

Furthermore, in our experiments, the AdaDelta optimizer (Zeiler, 2012) is employed to optimize our model.

D Phrase Table

The phrase table consists of lexical units (i.e., words and phrases) paired with semantic primitives that frequently co-occur with them. It can be obtained with statistical methods.

For CFQ, we leverage GIZA++[4] (Och and Ney, 2003) toolkit to extract alignment pairs from training examples. We obtain 109 lexical units, each of which is paired with 1.7 candidate semantic primitives on average. Some examples in phrase table are shown in Table 10

As to COGS, for each possible lexical unit, we first filter out the semantic primitives that exactly co-occur with it, and delete lexical units with no semantic primitive. Among the remaining lexical units, for those only contain one semantic primitive, we record their co-occurring semantic primitives

---

[4] https://github.com/moses-smt/giza-pp.git

Operation	Arguments	Result Type	Example
ON(t_1, t_2)	[t_1: Entity, t_2: Entity]	Entity	Emma ate [**the cake** on **a table**] .
IN(t_1, t_2)			A girl was awarded [**a cake** in **a soup**] .
BESIDE(t_1, t_2)			Amelia dusted [**the girl** beside **a stage**] .
ON^{-1}, IN^{-1}, BESIDE^{-1}			NONE
REC-THE(t_1, t_2)	[t_1: Entity, t_2: Entity]	Entity	Lily gave [**Emma** **a strawberry**] .
THE-REC(t_1, t_2)			A girl offered [**a rose** to **Isabella**] .
AGE-THE, THE-AGE, REC-AGE, AGE-REC			-
FillFrame(t_1, t_2)	[t_1: Entity, t_2: Pred/Prop] [t_1: Pred/Prop, t_2: Entity]	Proposition	A cat [**disintegrated** **a girl**] .
CCOMP(t_1, t_2)	[t_1: Pred/Prop, t_2: Pred/Prop]	Proposition	[**Emma liked** that **a girl saw**] .
XCOMP(t_1, t_2)			David [**expected** to **cook**] .
CCOMP^{-1}, XCOMP^{-1}			NONE

Table 8: Semantic operations in COGS. “Pred” and “Prop” are abbreviations of “Predicate” and “Proposition”, respectively. “AGE”, “THE” and “REC” are abbreviations of “AGENT”, “THEME” and “RECIPIENT”, respectively. “-” omits similar examples. Some operations contain “NONE” example, indicating that no example utilize these operations in dataset.

Operation	Arguments	Result Type	Example
UNION(t_1, t_2)	[t_1: Entity/Prop, t_2: Entity/Prop]	Proposition	what is the population of [**var0** **var1**]
INTER(t_1, t_2)			how many [**cities named var0** **in the usa**]
EXC(t_1, t_2) EXC^{-1}(t_1, t_2)			which [**capitals** are not **major cities**]
CONCAT(t_1, t_2) CONCAT^{-1}(t_1, t_2)	[t_1: Pred, t_2: Pred]	Pred	what is the [**capital** **of** var0]
FillIn(t_1, t_2)	[t_1: Entity/Prop, t_2: Pred] [t_1: Pred, t_2: Entity/Prop]	Proposition	how many [**citizens** in **var0**]

Table 9: Semantic operations in GEO. “Pred” and “Prop” are abbreviations of “Predicate” and “Proposition”, respectively. “INTER”, “EXC” and “CONCAT” are abbreviations of “INTERSECTION”, “EXCLUDE” and “CONCATENATION”, respectively.

as ready semantic primitives. For lexical units with more than one semantic primitives, we delete the ready semantic primitives from their co-occurring semantic primitives. Finally, we obtain 731 lexical units and each lexical unit is paired with just one semantic primitive.

As GEO is quite small, we obtain its phrase table by handcraft.

E More Examples

We show more examples of generated tree-structures and semantics in Figure 6.

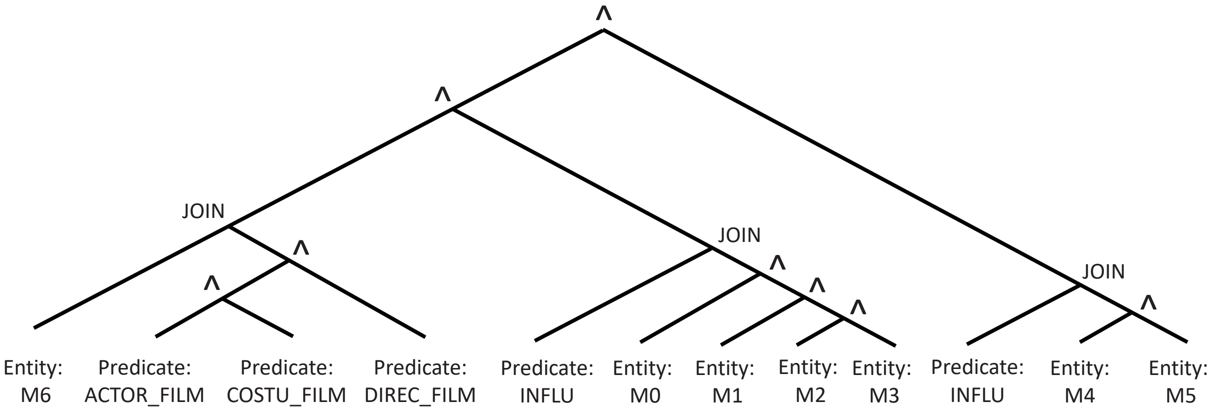

(a) An example of generated results in CFQ benchmark with the input "Did M6' s star, costume designer, and director influence M0, M1, M2, and M3 and influence M4 and M5 ".

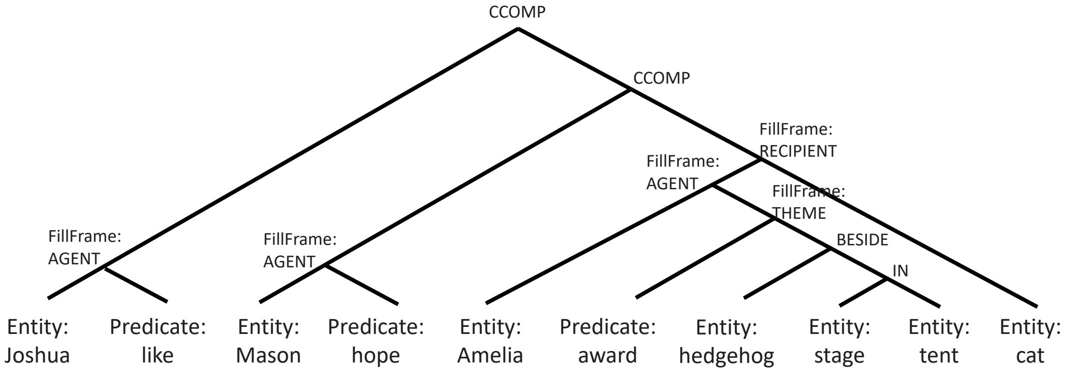

(b) An example of generated results in COGS benchmark with the input "Joshua liked that Mason hoped that Amelia awarded the hedgehog beside the stage in the tent to a cat".

Figure 6: Examples of generated tree-structures and semantics in CFQ and COGS benchmarks.

Lexical Unit	**Semantic Primitive(s)**	**Type**
M0	M0	Entity
executive producer	film.film.executive_produced_by	Predicate
	film.producer.films_executive_produced	Predicate
editor	a film.editor	Attribute
	film.editor.film	Predicate
	film.film.edited_by	Predicate
Italian	people.person.nationality m_03rjj	Attribute

Table 10: Some examples in CFQ phrase table.

Out of Order: How Important Is The Sequential Order of Words in a Sentence in Natural Language Understanding Tasks?

Thang M. Pham[1]
thangpham@auburn.edu

Trung Bui[2]
bui@adobe.com

Long Mai[2]
malong@adobe.com

Anh Nguyen[1]
anh.ng8@gmail.com

[1]Auburn University [2]Adobe Research

Abstract

Do state-of-the-art natural language understanding models care about word order? Not always! We found 75% to 90% of the correct predictions of BERT-based classifiers, trained on many GLUE tasks, remain constant after input words are randomly shuffled. Although BERT embeddings are famously contextual, the contribution of each individual word to classification is almost unchanged even after its surrounding words are shuffled. BERT-based models exploit superficial cues (e.g. the sentiment of keywords in sentiment analysis; or the word-wise similarity between sequence-pair inputs in natural language inference) to make correct decisions when tokens are randomly shuffled. Encouraging models to capture word order information improves the performance on most GLUE tasks and SQuAD 2.0. Our work suggests that many GLUE tasks are not challenging machines to understand the meaning of a sentence.

1 Introduction

Machine learning (ML) models recently achieved excellent performance on state-of-the-art benchmarks for evaluating natural language understanding (NLU). In July 2019, RoBERTa (Liu et al., 2019) was the first to surpass a human baseline on GLUE (Wang et al., 2019). Since then, 13 more methods have also outperformed humans on the GLUE leaderboard. Notably, at least 8 out of the 14 solutions are based on BERT (Devlin et al., 2019)—a transformer architecture that learns representations via a bidirectional encoder. Given their superhuman GLUE-scores, how do BERT-based models solve NLU tasks? How do their NLU capability differs from that of humans?

We shed light into these important questions by examining model sensitivity to the order of words. Word order is one of the key characteristics of a sequence and is tightly constrained by many linguistic factors including syntactic structures, subcategorization, and discourse (Elman, 1990). Thus, arranging a set of words in a correct order is considered a key problem in language modeling (Hasler et al., 2017; Zhang and Clark, 2015).

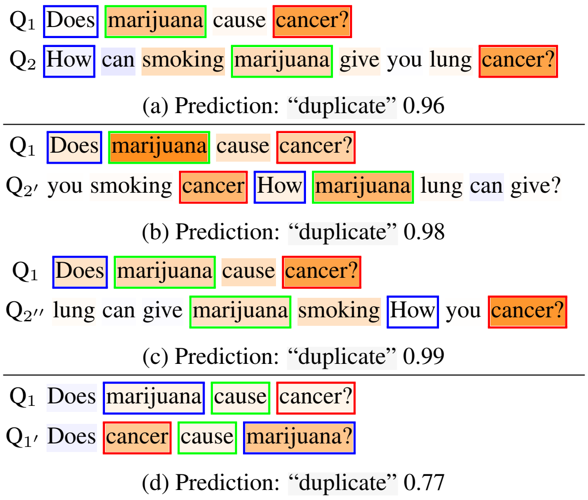

Q_1 Does marijuana cause cancer?
Q_2 How can smoking marijuana give you lung cancer?

(a) Prediction: "duplicate" 0.96

Q_1 Does marijuana cause cancer?
$Q_{2'}$ you smoking cancer How marijuana lung can give?

(b) Prediction: "duplicate" 0.98

Q_1 Does marijuana cause cancer?
$Q_{2''}$ lung can give marijuana smoking How you cancer?

(c) Prediction: "duplicate" 0.99

Q_1 Does marijuana cause cancer?
$Q_{1'}$ Does cancer cause marijuana?

(d) Prediction: "duplicate" 0.77

Figure 1: A RoBERTa-based model achieving a 91.12% accuracy on QQP, here, correctly labeled a pair of Quora questions "duplicate" (a). Interestingly, the predictions remain unchanged when all words in question Q_2 is randomly shuffled (b–c). QQP models also often incorrectly label a real sentence and its shuffled version to be "duplicate" (d). We found evidence that GLUE models rely heavily on words to make decisions e.g. here, "marijuana" and "cancer" (more important words are highlighted by LIME). Also, there exist self-attention matrices tasked explicitly with extracting word-correspondence between two input sentences regardless of the position of those words. Here, the top-3 pairs of words assigned the highest self-attention weights at (layer 0, head 7) are inside red, green, and blue rectangles, respectively.

Therefore, a natural question is: **Do BERT-based models trained on GLUE care about the order of words in a sentence?** Lin et al. (2019)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1145–1160
August 1–6, 2021.

found that pretrained BERT captures word-order information in the first three layers. However, it is unknown whether BERT-based classifiers actually use word order information when performing NLU tasks. Recently, Wang et al. (2020) showed that incorporating additional word-ordering and sentence-ordering objectives into BERT pretraining could lead to text representations (StructBERT) that enabled improved GLUE scores. However, StructBERT findings are inconclusive across different GLUE tasks and models. For example, in textual entailment (Wang et al., 2019, RTE), StructBERT improved the performance for BERT$_{\text{large}}$ but hurt the performance for RoBERTa (Table 2d).

Wang et al. 2020 motivated interesting questions: **Are state-of-the-art BERT-based models using word order information when solving NLU tasks? If not, what cues do they rely on?** To the best of our knowledge, our work is the first to study the above questions for an NLU benchmark (GLUE). We tested BERT-, RoBERTa-, and ALBERT-based (Lan et al., 2020) models on 7 GLUE tasks where the words of only one select sentence in the input text are shuffled at varying degrees. An ideal agent that truly understands language is expected to choose a "reject" option when asked to classify a sentence whose words are randomly shuffled. Alternatively, given shuffled input words, true NLU agents are expected to perform at random chance in multi-way classification that has no "reject" options (Fig. 1b). Our findings include:

1. 65% of the groundtruth labels of 5 GLUE tasks can be predicted when the words in one sentence in each example are shuffled (Sec. 3.1).
2. Although pretrained BERT embeddings are known to be contextual, in some GLUE tasks, the contribution of an individual word to classification is almost unchanged even after its surrounding words are shuffled (Sec. 3.3).
3. In sentiment analysis (SST-2), the polarity of a single salient word is $\geq 60\%$ predictive of an entire sentence's label (Sec. 3.4.1).
4. BERT-based models trained on sequence-pair GLUE tasks used a set of self-attention heads for finding similar tokens shared between the two inputs (Sec. 3.4).
5. Encouraging RoBERTa-based models to be more sensitive to word order improves the performance on SQuAD 2.0 and most GLUE tasks tested (i.e. except for SST-2) (Sec. 3.5).

Despite their superhuman scores, most GLUE-trained models behave similarly to Bag-of-Words (BOW) models, which are prone to naive mistakes (Fig. 1b–d). Our results also suggest that GLUE does not necessarily require syntactic information or complex reasoning.

2 Methods

2.1 Datasets

We chose GLUE because of three reasons: (1) GLUE is a common benchmark for NLU evaluation (Wang et al., 2019); (2) there exist NLU models (e.g. RoBERTa) that outperform humans on GLUE, making an important case for studying their behaviors; (3) it is unknown how sensitive GLUE-trained models are to word order and whether GLUE requires them to be sensitive (Wang et al., 2020).

Tasks Out of 9 GLUE tasks, we chose all 6 binary-classification tasks because they share the same random baseline of 50% accuracy and enable us to compare models' word-order sensitivity across tasks. Six tasks vary from acceptability (CoLA Warstadt et al. 2019), to natural language inference (QNLI Rajpurkar et al. 2016), RTE (Wang et al., 2019), paraphrase (MRPC Dolan and Brockett 2005, QQP Quora 2017), and sentiment analysis (SST-2 Socher et al. 2013).

We also performed our tests on STS-B (Cer et al., 2017)—a regression task of predicting the semantic similarity of two sentences.[1] While CoLA and SST-2 require single-sentence inputs, all other tasks require sequence-pair inputs.

Reject options For all binary-classification tasks (except SST-2), the negative label is considered the reject option (e.g. QQP models can choose "not duplicate" in Fig. 1b to reject shuffled inputs).

Metrics We use accuracy scores to evaluate the binary classifiers (for ease of interpretation) and Spearman correlation to evaluate STS-B regressors, following Wang et al. (2019).

2.2 Classifiers

We tested BERT-based models because (1) they outperformed humans on the GLUE leaderboard; and (2) the pretrained BERT was shown to capture word positional information (Lin et al., 2019).

---

[1] We did not choose WNLI (Levesque et al., 2012) as model performance is not substantially above random baseline.

Pretrained BERT encoders We tested three sets of classifiers finetuned from three different, pretrained BERT variants: BERT, RoBERTa, and ALBERT, downloaded from Huggingface (2020). The pretrained models are the "base" versions i.e. bidirectional transformers with 12 layers and 12 self-attention heads. The pretraining corpus varies from uncased (BERT, ALBERT) to case-sensitive English (RoBERTa).

Classifiers For each of the seven GLUE tasks, we added one classification layer on top of each of the three pretrained BERT encoders and finetuned the entire model. Unless otherwise noted, the mean performance per GLUE task was averaged over three classifiers. Each model's performance matches either those reported on Huggingface (2020) or the original papers (Table A6).

Hyperparameters Following Devlin et al. (2019), we finetuned classifiers for 3 epochs using Adam (Kingma and Ba, 2015) with a learning rate of 0.00002, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. We used a batch size of 32, a max sequence length of 128, and dropout on all layers with a probability of 0.1.

2.3 Constructing sets of real and shuffled examples for experiments

Modifying one sentence As GLUE tasks vary in the number of inputs (one or two input sequences) and the sequence type per input (a sentence or a paragraph), we only re-ordered the words in one *sentence* from only one input while keeping the rest of the inputs unchanged. Constraining the modifications to a single sentence enables us to measure (1) the importance of word order in a single sentence; and (2) the interaction between the shuffled words and the unchanged, real context.

Random shuffling methods To understand model behaviors across varying degrees of word-order distortions, we experimented with three tests: randomly shuffling n-grams where n = $\{1, 2, 3\}$.

Shuffling 1-grams is a common technique for analyzing word-order sensitivity (Sankar et al., 2019; Zanzotto et al., 2020). We split a given sentence by whitespace into a list of n-grams, and re-combined them, in a random order, back into a "shuffled" sentence (see Table 1 for examples). The ending punctuation was kept intact. We re-sampled a new random permutation until the shuffled sentence was different from the original sentence.

As the label distributions, dev-set sizes, and the performance of models vary across GLUE tasks, to compare word-order sensitivity across tasks, we tested each model on two sets: (1) `dev-r` i.e. a subset of the original dev-set (Sec. 2.3.1); and (2) `dev-s` i.e. a clone of version of `dev-r` but that each example has one sentence with re-ordered words (Sec. 2.3.2).

	How can smoking marijuana give you lung cancer?
Q_3	lung cancer marijuana give you How can smoking?
Q_2	smoking marijuana lung cancer give you How can?
Q_1	marijuana can cancer How you smoking give lung?
Q_s	How can smoking cancer give you lung marijuana?

Table 1: A real question on Quora (QQP dataset) and its three modified versions (Q_3 to Q_1) created by randomly shuffling 3-grams, 2-grams, and 1-grams, respectively. Q_s was created by swapping two random nouns.

2.3.1 Selecting real examples

For each pair of (task, classifier), we selected a subset of dev-set examples via the following steps:

1. For tasks with either a single-sequence or a sequence-pair input, we used examples where the input sequence to be modified has only one sentence[2] that has more than 3 tokens (for shuffling 3-grams to produce a sentence different from the original sentence).
2. We only selected the examples that were correctly classified by the classifier (to study what features were important for high accuracy).
3. We balanced the numbers of positive and negative examples by removing random examples from the larger-sized class.

That is, on average, we filtered out ∼34% of the original data. See Table A4 for the total number of examples remaining after each filtering step above.

2.3.2 Creating shuffled sets

For each task, we cloned the `dev-r` sets above and modified each example to create a "shuffled" set (a.k.a. `dev-s`) per shuffling method.

Specifically, a CoLA and SST-2 example contains only a single sentence and we modified that sentence. Each QQP, MRPC and STS-B example has two sentences and we modified the first sentence. An RTE example has a pair of (premise, hypothesis), and we modified the hypothesis since it

---

[2] We used NLTK sentence splitter (Bird et al., 2009) to detect text that has more than one sentence.

is a single sentence while premises are paragraphs. Each QNLI example contains a pair of (question, answer) and we modified the question, which is a sentence, while an answer is often a paragraph.

3 Experiments and Results

3.1 How much is word order information required for solving GLUE tasks?

GLUE has been a common benchmark for evaluating NLU progress. But, do GLUE tasks require models to use word order and syntactic information? We shed light into this question by testing model performance when word order is increasingly randomized.

If a task strictly requires words to form a semantically meaningful sentence, then randomly re-positioning words in correctly-classified sentences will cause model accuracy to drop from 100% to 50% (i.e. the random baseline b for binary-classification tasks with two balanced classes). Thus, to compare model-sensitivity across tasks, we use a Word-Order Sensitivity score (WOS):

$$s = (100 - p)/(100 - b) \tag{1}$$

where $p \in [50, 100]$ is the accuracy of a GLUE-trained model evaluated on a `dev-s` set (described in Sec. 2.3.2) and $s \in [0, 1]$. Here, $b = 50$.

Experiments For each GLUE task, we computed the mean accuracy and confidence score over three classifiers (BERT, RoBERTa, and ALBERT-based) on `dev-s` sets created by shuffling 1-grams, 2-grams, and 3-grams. The results reported in Table 2 were averaged over 10 random shuffling runs (i.e. 10 random seeds) per n-gram type, and then averaged over 3 models per task.

Results We found that for CoLA, i.e. detecting grammatically incorrect sentences, the model accuracy, on average, drops to near random chance i.e. between 50.69% and 56.36% (Table 2b) when n-grams are shuffled. That is, most of examples were classified into "unacceptable" after n-gram shuffling, yielding ~50% accuracy (see Fig. A2 for qualitative examples).

Surprisingly, for the rest of the 5 out of 6 binary-classification tasks (i.e. except CoLA), between 75% and 90% of the originally correct predictions remain constant after 1-grams are randomly reordered (Table 2b; 1-gram). These numbers increase as the shuffled n-grams are longer (i.e. as n increases from 1→3), up to 95.32% (Table 2b; QNLI). Importantly, given an average dev-set accuracy of 86.35% for these 5 tasks, **at least** 86.35% $\times$ 75% $\approx$ **65% of the groundtruth labels of these 5 GLUE tasks can be predicted when all input words in one sentence are randomly shuffled.**

Additionally, on average over three n-gram types, models trained on these five GLUE tasks are from 2 to 10 times more *insensitive* to word-order randomization than CoLA models (Table 2c). That is, if not explicitly tasked with checking for grammatical errors, GLUE models mostly will not care about the order of words in a sentence (see qualitative examples in Figs. 1, A2–A4). Consistently, the confidence scores of BERT-based models for five non-CoLA tasks only dropped ~2% when 1-grams are shuffled (Table 2).

Consistently across three different BERT "base" variants and a RoBERTa "large" model (Table A5), our results suggest that word order and syntax, in general, are not necessarily required to solve GLUE.

2-noun swaps Besides shuffled n-grams, we also repeated all experiments with more syntactically-correct modified inputs where only two random nouns in a sentence were swapped (Table 1; Q_s). This is a harder test for NLU models since the meaning of a sentence with two nouns swapped often changes while its syntax remains correct. We found the conclusions to generalize to this setting. That is, the models hardly changed predictions although the meanings of the original sentence and its swapped version are different (Table 2b; 2-noun swap vs. 1-gram).

3.2 How sensitive are models trained to predict the similarity of two sentences?

An interesting hypothesis is that models trained explicitly to evaluate the semantic similarity of two sentences should be able to tell apart real from shuffled examples. Intuitively, word order information is essential for understanding what an entire sentence means and, therefore, for predicting whether two sentences convey the same meaning.

We tested this hypothesis by analyzing the sensitivity of models trained on QQP and STS-B—two prominent GLUE tasks for predicting semantic similarity of a sentence pair. While QQP is a binary classification task, STS-B is a regression task where a pair of two sentences is given a score $\in [0, 5]$ denoting their semantic similarity.

Experiments We tested the models on `dev-r` and `dev-s` sets (see Sec. 2.3.2) where in each pair, the word order of the first sentence was randomized while the second sentence was kept intact.

QQP results Above 83% of QQP models' correct predictions on real pairs remained unchanged after word-order randomization (see Figs. 1a–c for examples).

STS-B results Similarly, STS-B model performance only drops marginally, i.e. less than 2 points from 89.67 to 87.80 in Spearman correlation (Table 2; STS-B). Since a STS-B model outputs a score $\in [0, 5]$, we binned the scores into 6 ranges. One might expect STS-B models to assign near-zero similarity scores to most modified pairs. However, the distributions of similarity scores for the modified and real pairs still closely match up (Fig. 2). In sum, **despite being trained explicitly on predicting semantic similarity of sentence pairs, QQP and STS-B are surprisingly insensitive to n-gram shuffling, exhibiting naive understanding of sentence meanings**.

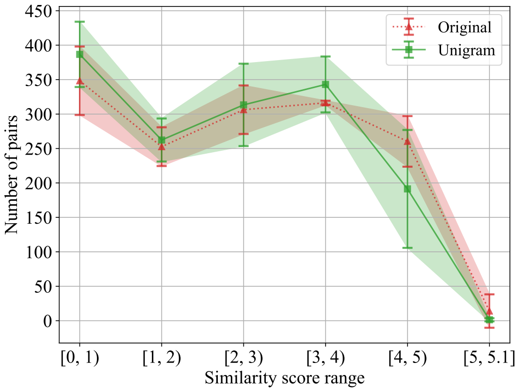

Figure 2: The distribution of similarity scores over 6 ranges for the (real, shuffled) pairs in `dev-s` (green) is highly similar to that for (real, real) STS-B pairs in `dev-r` (red). The statistics in each range were computed over 3 models (BERT, RoBERTa, and ALBERT).

3.3 How important are words to classification after their context is shuffled?

BERT representations for tokens are known to be highly contextual (Ethayarajh, 2019). However, after finetuning on GLUE, would the importance of a word to classification drop after its context is shuffled?

To answer the above question, we used LIME (Ribeiro et al., 2016) to compute word importance.

LIME computes a score $\in [-1, 1]$ for each token in the input denoting how much its presence contributes for or against the network's predicted label (Fig. 1; highlights). The importance score per word w is intuitively the mean confidence-score drop over a set of randomly-masked versions of the input when w is masked out.

Experiments We chose to study RoBERTa-based classifiers here because they have the highest GLUE scores among the three BERT variants considered. We observed that **62.5% (RTE) to 79.6% (QNLI) of the `dev-r` examples were consistently, correctly classified into the same labels in all 5 different random shuffles** (i.e. 5 different random seeds). We randomly sampled 100 such examples per binary-classification task and computed their LIME attribution maps to compare the similarity between the LIME heatmaps before and after unigrams are randomly misplaced.

Results On CoLA and RTE, the importance of words (i.e. mean absolute value of LIME-attribution per word), *decreased* substantially by 0.036 and 0.019, respectively. That is, the individual words become *less important* after their context is distorted—a behavior expected when CoLA and RTE have the highest WOS scores (Table 2). In contrast, for the other 4 tasks, word importance only changed marginally (by 0.008, i.e. 4.5× smaller than the 0.036 change in CoLA). That is, **except for CoLA and RTE models, the contribution of a word to classification is almost unchanged even after the context of each word is randomly shuffled** (Fig. 1a–c). This result suggests that the word embeddings after finetuning on GLUE became much less contextual than the pretrained BERT embeddings (Ethayarajh, 2019).

3.4 If not word order, then what do classifiers rely on to make correct predictions?

Given that all non-CoLA models are highly insensitive to word-order randomization, how did they arrive at correct decisions when words are shuffled?

We chose to answer this question for SST-2 and QNLI because they have the lowest WOS scores across all 6 GLUE tasks tested (Table 2) and they are representative of single-sentence and sequence-pair tasks, respectively.

Task	(a) Perf. on `dev-r`		(b) Performance on `dev-s`				(c) Word-Order Sensitivity			(d) StructBERT improvements		
	Models	Baseline	2-noun swap	1-gram	2-gram	3-gram	1-gram	2-gram	3-gram	BERT$_{\text{base}}$	BERT$_{\text{large}}$	RoBERTa
`CoLA`	100 (0.93)	50	71.75 (0.91)	50.69 (0.95)	53.98 (0.94)	56.36 (0.92)	**0.99**	**0.92**	**0.87**	**+4.9**	+4.8	+1.4
`RTE`	100 (0.81)	50	85.86 (0.81)	75.69 (0.80)	81.89 (0.80)	85.18 (0.79)	0.49	0.36	0.30	N/A	**+13.0**	–0.9
`QQP`	100 (0.98)	50	86.90 (0.96)	83.19 (0.96)	88.02 (0.96)	89.04 (0.96)	0.34	0.24	0.22	+0.7	+1.2	+0.5
`MRPC`	100 (0.91)	50	96.51 (0.91)	83.89 (0.89)	87.1 (0.90)	89.38 (0.90)	0.32	0.26	0.21	N/A	+3.9	**+1.7**
`SST-2`	100 (0.99)	50	97.78 (0.98)	84.04 (0.96)	88.35 (0.97)	90.56 (0.97)	0.32	0.23	0.19	+0.2	+0.3	+0.4
`QNLI`	100 (0.98)	50	94.31 (0.97)	89.42 (0.96)	93.85 (0.97)	95.32 (0.98)	0.21	0.12	0.09	N/A	+3.0	+0.3
`STS-B`	89.67	N/A	88.93	87.80	88.66	88.95	N/A	N/A	N/A	N/A	N/A	N/A

Table 2: All results (a–c) are reported on the GLUE `dev-r` sets i.e. 100% accuracy (a). Shuffling n-grams caused the accuracy to drop (b) the largest for CoLA and the least for QNLI. Each row is computed by averaging the results of 3 BERT-based models and 10 random shuffles. From top to bottom, the Word-Order Sensitivity (WOS) is sorted descendingly (c) and is consistent across three types of n-grams i.e. WOS scores decrease from top down and from left to right. In contrast, the StructBERT results (d), taken from Table 1 and 4 in Wang et al. 2020, showed inconsistent improvements across different tasks. STS-B results are in scaled Spearman correlation. In addition to small accuracy drops, the mean confidence scores of all classifiers—reported in parentheses e.g. "(0.93)"—also changed marginally after words are shuffled (a vs. b).

3.4.1 SST-2: Salient words are highly predictive of sentence labels

As 84.04% of the SST-2 correct predictions did not change after word-shuffling (Table 2b), a common hypothesis is that the models might rely heavily on a few key words to classify an entire sentence.

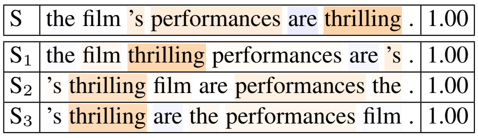

S	the film 's performances are thrilling .	1.00
S$_1$	the film thrilling performances are 's .	1.00
S$_2$	's thrilling film are performances the .	1.00
S$_3$	's thrilling are the performances film .	1.00

Figure 3: An original SST-2 dev-set example (S) and its three shuffled versions (S$_1$ to S$_3$) were all correctly labeled "positive" by a RoBERTa-based classifier with high confidence scores (right column).

Experiments To test this hypothesis, we took all SST-2 `dev-r` examples whose all 5 randomly shuffled versions were all correctly labeled by a RoBERTa-based classifier (i.e. this "5/5" subset is ∼65% of the dev-set). We used LIME to produce a heatmap of the importance of words in each example.

We identified the polarity of each top-1 most important word (i.e. the highest LIME-attribution score) per example by looking it up in the Opinion Lexicon (Hu and Liu, 2004) of 2,006 positive and 4,783 negative words. ∼57% of these top-1 words were found in the dictionary and labeled either "positive" or "negative" (see Table A3).

Results We found that if the top-1 word has a positive meaning, then there is a 100% probability that the sentence's label is "positive". For example, the word "thrilling" in a movie review indicates a "positive" sentence (see Fig. 3). Similarly, the conditional probability of a sentence being labeled "negative" given a negative top-1 word is 94.4%. That is, given this statistics, the SST-2 label distribution and model accuracy, **at least 60% of the SST-2 dev-set examples can be correctly predicted from only a single top-1 salient word.**

We also reached similar conclusions when experimenting with ALBERT classifiers and the SentiWords dictionary (Gatti et al., 2015) (see Table A3).

3.4.2 Self-attention layers matching similar words in both the question and the answer

For sequence-pair tasks, e.g. QNLI, how can models correctly predict "entailment" when the question words are randomly shuffled (Fig. 4; Q$_1$) or when the question syntax is correct but its meaning

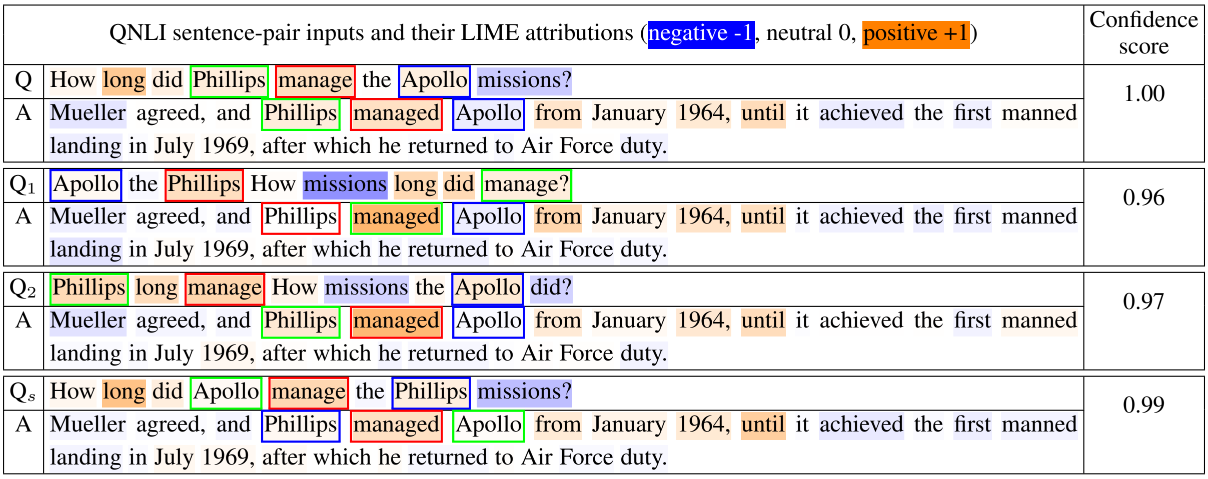

QNLI sentence-pair inputs and their LIME attributions (negative -1, neutral 0, positive +1)		Confidence score
Q	How long did Phillips manage the Apollo missions?	1.00
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	
Q_1	Apollo the Phillips How missions long did manage?	0.96
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	
Q_2	Phillips long manage How missions the Apollo did?	0.97
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	
Q_s	How long did Apollo manage the Phillips missions?	0.99
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	

Figure 4: A RoBERTa-based model's correct prediction of "entailment" on the original input pair (Q, A) remains unchanged when the question is randomly shuffled (Q_1 & Q_2) or when two random nouns in the question are swapped (Q_s). The salient words in the questions e.g. manage and missions remain similarly important after their context has been shuffled. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both the question and the answer e.g. manage (Q) and managed (A). That is, the top-3 pairs of words that were assigned the largest question-to-answer weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles.

changes entirely (Fig. 4; Q_s). We hypothesize that inside the model, there might be a self-attention (SA) layer that extracts pairs of similar words that appear in both the question and the answer (e.g. "manage" vs. "managed" in Fig. 4).

Experiments To test this hypothesis, we analyzed the 5,000 QNLI `dev-r` examples (Table A4) of RoBERTa-based classifiers trained on QNLI. For each example, we identified one SA matrix (among all 144 as the base model has 12 layers & 12 heads per layer) that assigns the highest weights to pairs of similar words between the question and the answer, i.e. excluding intra-question and intra-answer attention weights (see the procedure in Sec. A).

Results First, in ∼58% of the examples, we found at least three pairs of words that *match* (i.e. the sum Levenshtein character-level edit-distance for all 3 pairs is ≤ 4). Second, we found, in total, 15 SA heads (out of the 144) which are explicitly tasked with capturing such question-to-answer word correspondence, *regardless* of word order (see Fig. 4).

Remarkably, **87% of the work of matching similar words that appear in both the QNLI question and the answer was handled by only 3 self-attention heads at (layer, head) of (0,7), (1,9), and (2,6)**.

We found consistent results when repeating the same analysis for other three sequence-pair tasks. That is, interestingly, **the three SA heads at exactly the same location of (0, 7), (1, 9), and (2, 6) account for 76%, 89%, and 83% of the "word-matching" task on QQP, RTE, and MRPC, respectively**. This coincidence is likely due to the fact that these classifiers were finetuned for different downstream tasks starting from the same pretrained RoBERTa encoder. See Figs. 1, 4, A3–A4 for qualitative examples of these three tasks.

How important are the 15 word-matching attention heads to QNLI model performance? We found that zero-ing out 15 random heads had almost no effect to correctly-classified predictions–i.e. accuracy dropped marginally (-1% to -3%, Table 3) across different groups of examples. However, ablating the 15 word-matching heads caused the performance to drop substantially i.e. (a) by 9.6% on the 1,453 "positive" examples identified in Sec. A; (b) by 22.1% on a set of 2,906 random, examples including both "positive" and "negative" examples (at 50/50 ratio); and (c) by 24.5% on the entire QNLI 5,000-example `dev-r` set. That is, **the 15 SA heads that learned to detect similar words played an important role in solving QNLI, i.e. enabling at least ∼50% of the correct predictions** (Table 3d; accuracy dropped from 100% to 75.54% when the random chance is 50%). In sum, we found overlap between words in the question and answer of QNLI examples and strong evidence that QNLI models harnessed self-attention to exploit such overlap to make correct decisions in spite of a random word-order.

QNLI `dev-r` examples	Full network	Zero-out 15 attention matrices	
		Random	Ours
a. 1,453 selected 0/5 (+) examples	100	99.31	90.43
b. 1,453 random 0/5 (+) examples	100	99.24	91.05
c. 1,453 random 0/5 (+) examples & 1,453 random 0/5 (-) examples	100	98.18	77.91
d. (+/-) All 5,000 examples	100	96.96	75.54

Table 3: Zero-ing out a set of 15 "word-matching" self-attention matrices (identified via the procedure in Sec. 3.4.2) caused a substantial drop of ∼25% in accuracy (d) while the random baseline is 50%. These 15 matrices played an important role in QNLI because ablating 15 random matrices only caused a ∼1-3% drop in accuracy.

3.5 Does increasing word-order sensitivity lead to higher model performance?

Here, we test whether encouraging BERT representations to be more sensitive to word order (i.e. more syntax-aware) would improve model performance on GLUE & SQuAD 2.0 (Rajpurkar et al., 2018). We performed this test on the five GLUE binary-classification tasks (i.e. excluding CoLA because its WOS score is already at 0.99; Table 2).

Experiments Inspired by the fact that CoLA models are highly sensitive to word order, we finetuned the pretrained RoBERTa on a *synthetic*, CoLA-like task first, before finetuning the model on downstream tasks.

The synthetic task is to classify a single sentence into "real" vs. "fake" where the latter is formed by taking each real sentence and swapping two random words in it. For every downstream task (e.g. SST-2), we directly used its original training and dev sets to construct a balanced, 2-class, synthetic dataset. After finetuning the pretrained RoBERTa on this synthetic binary classification task, we re-initialized the classification layer (keeping the rest unchanged) and continued finetuning it on a downstream task.

For both finetuning steps, we trained 5 models per task and followed the standard BERT finetuning procedure (described in Sec. 2.2).

Results After the first finetuning on synthetic tasks, all models obtained a ∼99% training-set accuracy and a ∼95% dev-set accuracy. **After the second finetuning on downstream tasks, we observed that all models were substantially more sensitive to word order**, compared to the baseline models (which were only finetuned on the downstream tasks). That is, we repeated the 1-gram shuffling test (Sec. 3.1) and found a ∼1.5 to 2× increase in the WOS scores of all models (see Table 4a vs. b).

GLUE `dev-s`	(a) RoBERTa		(b) Ours	
	Accuracy	WOS	Accuracy	WOS
`RTE`	80.76	0.38	64.01	**0.72** (+189%)
`MRPC`	83.86	0.32	72.88	**0.54** (+169%)
`SST-2`	84.26	0.31	76.97	**0.46** (+148%)
`QQP`	87.66	0.25	77.11	**0.46** (+184%)
`QNLI`	91.09	0.18	82.44	**0.35** (+194%)

Table 4: With finetuning on synthetic tasks, all of our models (b) have a larger drop in accuracy on shuffled `dev-s` examples, compared to the standard RoBERTa-based classifiers (a). That is, our models are substantially more sensitive to word-order randomization (i.e. +148% to +194% in WOS scores).

GLUE On GLUE dev sets, on average over 5 runs, our models outperformed the RoBERTa baseline on all tasks except for SST-2 (Table 5). The highest improvement is in RTE (from 72.2% to 73.21% on average, and to 74.73% for the best single model), which is consistent with the fact that RTE has the highest WOS score among non-CoLA tasks (Sec. 3.1).

SQuAD 2.0 Our models also outperformed the RoBERTa baseline on the SQuAD 2.0 dev set, with the highest F1 gain from 80.62% to 81.08% (Table 5).

In sum, leveraging the insights that the original BERT-based models are largely word-order invariant, we showed that increasing model sensitivity via a simple extra finetuning step directly improves GLUE and SQuAD 2.0 performance.

4 Related Work

Pretrained BERT Lin et al. (2019) found that positional information is encoded in the first there layers of BERT$_{\text{base}}$ and fades out starting layer 4. Ettinger (2020) found that BERT heavily relies on word order when predicting missing words in masked sentences from the CPRAG-102 dataset. That is, shuffling words in the context sentence caused the word-prediction accuracy to drop by ∼1.3 to 2×. While all above work studied the

	RTE (Acc)	QQP (Acc)	MRPC (Acc)	SST-2 (Acc)	QNLI (Acc)	SQuAD (F1)
RoBERTa	72.20	91.12	87.25	**94.50**	92.57	80.62
Our best model	**74.73** **+2.53**	**91.31** **+0.19**	**88.73** **+1.48**	94.50 +0	**93.08** **+0.51**	**81.08** **+0.46**
Average (5 runs)	**73.21** **+1.01**	**91.19** **+0.07**	**87.31** **+0.06**	94.22 -0.28	**92.71** **+0.14**	**80.75** **+0.13**

Table 5: Finetuning the pretrained RoBERTa on synthetic tasks (before finetuning on the downstream tasks) improved model dev-set performance on SQuAD 2.0 (b) and all the tested tasks in GLUE (a), except SST-2.

pretrained BERT, we instead study BERT-based models *finetuned* on downstream tasks.

Word-ordering as an objective In text generation, Elman (1990) found that recurrent neural networks were sensitive to regularities in word order in simple sentences. Language models (Mikolov et al., 2010) with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) were able to recover the original word order of a sentence from randomly-shuffled words even without any explicit syntactic information (Schmaltz et al., 2016). Wang et al. 2020 also observed an increase in GLUE performance after pretraining BERT with two additional objectives of word-ordering and sentence-ordering. Their work differs from ours in three points: (1) they did not study the importance of word order alone; (2) StructBERT improvements were inconsistent across tasks and models (Table 2d) and motivated us to compare the word-order importance between GLUE tasks; and (3) we proposed to improve model performance by finetuning not pretraining.

Word-order insensitivity in other NLP tasks ML models have been shown to be insensitive to word order in several NLP tasks such as reading comprehension (Si et al., 2019; Sugawara et al., 2020), dialog (Sankar et al., 2019), natural language inference (Parikh et al., 2016; Sinha et al., 2020), and essay scoring (Parekh et al., 2020). Zanzotto et al. 2020 found that for several text classification tasks, syntactic information was not always required. In word prediction, LSTMs and pre-trained BERT were found to exhibit a certain degree of insensitivity when the context words are randomly shuffled (Khandelwal et al., 2018; Mitchell and Bowers, 2020; Ettinger, 2020). Compared to the prior work, we are the first to perform a word-order analysis on a NLU benchmark and to contrast this sensitivity across the tasks.

Humans can also be word-order invariant A recent human study interestingly showed that sentences with scrambled word orders elicit a response as high as that elicited by original sentences as long as the local mutual information among words is high enough (Mollica et al., 2020). Gibson et al. 2013 found that humans can also exhibit word-order-invariance effects, especially when one interpretation is much more semantically plausible. Our work therefore documents an important similarity between humans and advanced NLU models.

Invariance to patch-order in computer vision In computer vision, the accuracy of state-of-the-art image classifiers was found to only drop marginally when the patches in an image were randomly shuffled (Chen et al., 2020; Zhang and Zhu, 2019).

5 Discussion and Conclusion

Consistently across three BERT variants and two model sizes, we found that GLUE-trained BERT-based models are often word-order invariant unless explicitly asked for (e.g. in CoLA).

We present a reflection on the progress of NLU by studying GLUE—a benchmark where humans have been surpassed by many models in the past 18 months. As suggested by our work, these models; however, may neither use syntactic information nor complex reasoning. We revealed how self-attention, a key building block in modern NLP, is being used to extract superficial cues to solve sequence-pair GLUE tasks even when words are out of order.

Adversarial NLI We also replicated our shuffling experiments on ANLI (Nie et al., 2020), a task considered challenging to existing models, and where RoBERTa-based models only obtained a 56% accuracy. We found RoBERTa-based models to remain not always sensitive to word-order randomization on ANLI (Table A2; WOS of 0.63), suggesting a common issue in existing benchmarks.

Acknowledgments

We thank Michael Alcorn, Qi Li, and Peijie Chen for helpful feedback on the early results. We are grateful for valuable feedback from Sam Bowman, Ernest Davis, and Melanie Mitchell. AN is supported by the National Science Foundation under Grant No. 1850117, and donations from the NaphCare Charitable Foundation, and Nvidia.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Peijie Chen, Chirag Agarwal, and Anh Nguyen. 2020. The shape and simplicity biases of adversarially robust imagenet-trained cnns. *arXiv e-prints*, pages arXiv–2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Lorenzo Gatti, Marco Guerini, and Marco Turchi. 2015. Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, 7(4):409–421.

Edward Gibson, Leon Bergen, and Steven T Piantadosi. 2013. Rational integration of noisy evidence and prior semantic expectations in sentence interpretation. *Proceedings of the National Academy of Sciences*, 110(20):8051–8056.

GLUE leaderboard. Glue benchmark. `https://gluebenchmark.com/leaderboard`. (Accessed on 01/29/2021).

Eva Hasler, Felix Stahlberg, Marcus Tomalin, Adrià de Gispert, and Bill Byrne. 2017. A comparison of neural models for word ordering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 208–212, Santiago de Compostela, Spain. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA. Association for Computing Machinery.

Huggingface. 2020. Pretrained models — transformers 3.3.0 documentation. `huggingface.co/transformers/pretrained_models.html`. (Accessed on 09/30/2020).

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego.*

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside bert's linguistic knowledge. *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.*

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Jeff Mitchell and Jeffrey Bowers. 2020. Priorless recurrent networks learn curiously. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5147–5158.

Francis Mollica, Matthew Siegelman, Evgeniia Diachek, Steven T Piantadosi, Zachary Mineroff, Richard Futrell, Hope Kean, Peng Qian, and Evelina Fedorenko. 2020. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

NLTK. nltk.metrics.distance — nltk 3.5 documentation. `https://www.nltk.org/_modules/nltk/metrics/distance.html`. (Accessed on 12/25/2020).

Swapnil Parekh, Yaman Kumar Singla, Changyou Chen, Junyi Jessy Li, and Rajiv Ratn Shah. 2020. My teacher thinks the world is flat! interpreting automatic essay scoring mechanism. *arXiv preprint arXiv:2012.13872*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Quora. 2017. (1) first quora dataset release: Question pairs - data @ quora - quora. `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`. (Accessed on 09/30/2020).

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 32–37, Florence, Italy. Association for Computational Linguistics.

Allen Schmaltz, Alexander M. Rush, and Stuart" Shieber. 2016. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas. Association for Computational Linguistics.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. 2019. What does bert learn from multiple-choice reading comprehension datasets? *arXiv preprint arXiv:1910.12391*.

Koustuv Sinha, Prasanna Parthasarathi, Joelle Pineau, and Adina Williams. 2020. Unnatural language inference. *arXiv preprint arXiv:2101.00010*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Saku Sugawara, Pontus Stenetorp, Kentaro Inui, and Akiko Aizawa. 2020. Assessing the benchmarking capacity of machine reading comprehension datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8918–8927.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Fabio Massimo Zanzotto, Andrea Santilli, Leonardo Ranaldi, Dario Onorati, Pierfrancesco Tommasino, and Francesca Fallucchi. 2020. KERMIT: Complementing transformer architectures with encoders of explicit syntactic interpretations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 256–267, Online. Association for Computational Linguistics.

Tianyuan Zhang and Zhanxing Zhu. 2019. Interpreting adversarially trained convolutional neural networks. In *International Conference in Machine Learning*.

Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Computational linguistics*, 41(3):503–538.

A Self-attention layers that match question-words to similar words in the answer

QNLI models being so insensitive to word shuffling (i.e. 89.4% of the correct predictions remain correct) suggests that inside the finetuned BERT, there might be a self-attention (SA) layer that extract pairs of similar words that appear in both the question and answer.

We started by analyzing all 2,500 "positive" `dev-r` examples (Table A4) of RoBERTa-based classifiers trained on QNLI because there were fewer and more consistent ways for labeling a sentence "positive" than for the "negative" (shown in Sec. 3.3).

Experiment There were 1,776 (out of 2,500) examples whose predictions did not change in 5 random shufflings (a.k.a 5/5 subset). For each such example, we followed the following 4 steps to identify one SA matrix (among all 144 as the base model has 12 layers & 12 heads per layer) that captures the strongest attention connecting the question and answer words.

1. Per example x, we created its shuffled version $\hat{x}$ by randomly shuffling words in the question and fed $\hat{x}$ into the classifier.
2. For each SA matrix obtained, we identified the top-3 highest-attention weights that connect the shuffled question tokens and the real answer tokens (i.e. excluding attention weights between question tokens or answer tokens only).
3. For each shuffled example $\hat{x}$, we identified one matrix M whose the top-3 word pairs are the nearest in Levenshtein character-level edit-distance (NLTK). For instance, the distance is 1 between manage and managed (Fig. 4).
4. For each matrix M identified for $\hat{x}$, we fed the corresponding real example x through the network and re-computed the edit-distance for each of the top-3 word pairs.

Results At step 3, there were 1,590 SA matrices (out of 1,776) whose the top-3 SA weights connected three pairs of *matching* words (i.e. the total edit-distance for 3 pairs together is ≤ 4)[3] that appear in both the shuffled question and original answer (see example top-3 pairs in Fig. 4). At step 4, this number only dropped slightly to 1,453 matrices when replacing the shuffled question by the original one (see Table A1 for detailed statistics).

Sum distance	(a) `dev-s` alone		(b) `dev-s` & `dev-r`	
	# examples	%	# examples	%
≤ 0	749	42.17	392	24.65
≤ 1	1,253	70.55	1,071	67.36
≤ 2	1,440	81.08	1,283	80.69
≤ 3	1,543	86.88	1,391	87.48
≤ 4	**1,590**	**89.53**	**1,453**	**91.38**
≤ 15	1,776	100.00	1,574	98.99
Total	1,776	100.00	1,590	100.00

Table A1: The number of QNLI examples where we found $\geq$ one self-attention matrix that the most strongly attends to three pairs of *matching* words when given the `dev-s` examples i.e. (modified question, real answer) (a) or when given both the shuffled and real examples (b). In other words, **the numbers in (b) denote the number of examples where (1) there exist $\geq$ 3 words, regardless of its word order, in the question that can be found in the accompanying real answer; and (2) these correspondences are captured by at least one self-attention matrix.** The sum edit-distance for all 3 pairs of words are less than N where $N = \{0, 1, 2, 3, 4, 15\}$ (left column).

However, there are only 15 unique, RoBERTa self-attention matrices in these 1,453 examples (see Fig. A1). Also at step 4, 83% of the same word pairs remained within the top-3 of the same SA matrices, after question replacement, i.e. 17% of attention changed to different pairs e.g. from ("managed", "manage") to ("it", "it").

First, our results showed that there is a set of 15 self-attention heads explicitly tasked with capturing question-to-answer word correspondence *regardless of word order*. Second, **for ~58% (i.e. 1,453 / 2,500) of QNLI "positive" examples: (1) there exist $\geq$ 3 words in the question that can be found in the accompanying answer; and (2) these correspondences are captured by at least one of the 15 SA matrices.** We also found similar results for 2,500 "negative" `dev-r` examples (data not shown).

---

[3] 4 is a tight budget to account for minor typos or punctuation differences e.g. "Amazon" vs. "Amazon's".

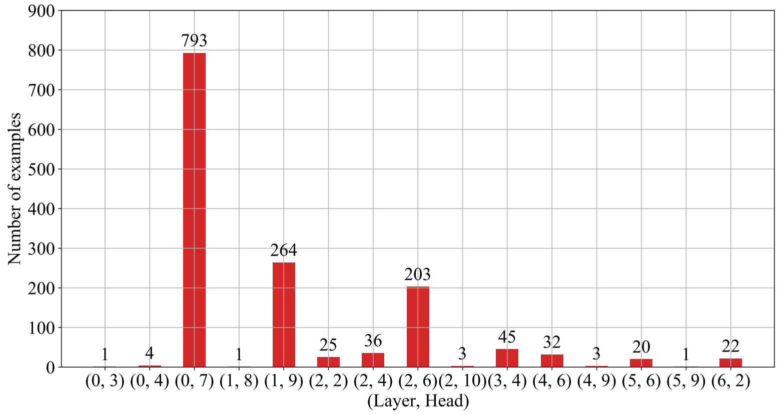

(a) Histogram of self-attention matrices

(Layer, Head)	# Examples	%
(0, 3), (0, 4), (0, 7)	798	54.9%
(1, 8), (1, 9)	265	18.2%
(2, 2), (2, 4), (2, 6), (2, 10)	267	18.4%
(3, 4)	45	3.1%
(4, 6), (4, 9)	35	2.4%
(5, 6), (5, 9)	21	1.5%
(6, 2)	22	1.5%
Total	1,453	100%

(b) Layer-wise comparison

Figure A1: Among 144 self-attention matrices in the RoBERTa-based classifier finetuned for QNLI, there are 15 "word-matching" matrices (a) that explicitly attend to pairs of similar words that appear in both questions and answers regardless of the order of words in the question (see example pairs in Fig. 4). For each QNLI example, we identified one such matrix that exhibits the matching behavior the strongest (a). 92% of the task of attending to duplicate words is mostly handled in the first three layers (b).

Model	Task	(a) Perf. on `dev-r`		(b) Perf. on `dev-s`				(c) Word-Order Sensitivity			
		Models	Baseline	2-noun swap	1-gram	2-gram	3-gram	2-noun swap	1-gram	2-gram	3-gram
RoBERTa$_{\text{base}}$	`ANLI`	100	33.33	74.26	57.74	66.63	69.04	0.39	0.63	0.50	0.46
	`A1`	100	33.33	81.46	63.31	71.52	75.37	0.28	0.55	0.43	0.37
	`A2`	100	33.33	70.83	54.61	64.73	67.02	0.44	0.68	0.53	0.49
	`A3`	100	33.33	70.50	55.29	63.63	64.73	0.55	0.67	0.55	0.53
RoBERTa$_{\text{large}}$	`ANLI`	100	33.33	70.41	54.87	64.11	68.76	0.44	0.68	0.54	0.47
	`A1`	100	33.33	78.06	60.31	70.57	75.86	0.33	0.6	0.44	0.36
	`A2`	100	33.33	67.88	51.44	60.64	66.31	0.48	0.73	0.59	0.51
	`A3`	100	33.33	65.30	52.85	61.11	64.10	0.52	0.71	0.58	0.54

Table A2: All results (a–c) of RoBERTa$_{\text{base}}$ and RoBERTa$_{\text{large}}$ models finetuned on the combination of NLI datasets (SNLI, MNLI, FEVER and ANLI) are reported on the ANLI `dev-r` sets (i.e. 100% accuracy) which includes A1, A2 and A3 (a). The accuracies for RoBERTa$_{\text{base}}$ and RoBERTa$_{\text{large}}$ on ANLI are 51.19% and 56.98%, respectively. Each row is computed by averaging the results of 10 random shuffles. Word-Order Sensitivity (WOS) of ANLI and its subsets (c). Since ANLI is 3-way classification task, the baseline is 33.33% (as described in Sec 2.3.1).

Dictionary	Opinion Lexicon (Hu and Liu, 2004)		SentiWords (Gatti et al., 2015)	
	(a) RoBERTa	(b) ALBERT	(c) RoBERTa	(d) ALBERT
Total examples in subset 5/5 ("positive" / "negative")	523 (278 / 245)	506 (228 / 278)	523 (278 / 245)	506 (228 / 278)
Not found in dictionary	223 / 523 (42.64%)	217 / 506 (42.89%)	110 / 523 (21.03%)	104 / 506 (20.55%)
Found in dictionary	300 / 523 (57.36%)	289 / 506 (57.11%)	413 / 523 (78.97%)	402 / 506 (79.45%)
P ("positive" sentence | positive top-1 word)	174 / 174 (100.00%)	143 / 144 (99.31%)	222 / 258 (86.05%)	186 / 215 (86.51%)
P ("negative" sentence | negative top-1 word)	119 / 126 (94.44%)	136 / 145 (93.79%)	145 / 155 (93.55%)	177 / 187 (94.65%)

Table A3: **If the top-1 most important word in an SST-2 5/5 example has a positive meaning, then there is a 100% chance that the sentence is labeled "positive" in SST-2.** Similarly, the conditional probability of a sentence being labeled "negative" given a negative most important word (by LIME Ribeiro et al. 2016) is 94.44%.

LIME attributions (negative -1, neutral 0, positive +1)		
CoLA example. Groundtruth: “acceptable”		
S	Medea denied poisoning the phoenix.	“acceptable” 0.99
S_1	poisoning the phoenix denied Medea.	“acceptable” 0.53
S_2	phoenix Medea denied the poisoning.	“acceptable” 0.99
S_3	Medea the poisoning phoenix denied.	“unacceptable” 0.95
S_4	phoenix Medea denied the poisoning.	“unacceptable” 0.99
S_5	Medea phoenix poisoning the denied.	“unacceptable” 0.96

Figure A2: Each CoLA example contains a single sentence. Here, we shuffled the words in the original sentence (S) five times to create five new sentences (S_1 to S_5) and fed them to a RoBERTa-based classifier for predictions. Words that are important for or against the prediction (by LIME Ribeiro et al. 2016) are in orange and blue, respectively. Most of the shuffled examples were classified into “unacceptable” label (i.e. grammatically incorrect) with even higher confidence score than the original ones.

MRPC example. Groundtruth: “equivalent”		
A	My decision today is not based on any one event . ”	“equivalent” 0.99
B	Governor Rowland said his decision was ” not based on any one event . ”	
A_1	event any is one decision based on My today not . ”	“equivalent” 0.98
B	Governor Rowland said his decision was ” not based on any one event . ”	
A_2	one based today not any My on event is decision . ”	“equivalent” 0.98
B	Governor Rowland said his decision was ” not based on any one event . ”	

Figure A3: Each MRPC example contains a pair of sentences i.e. (A, B). Here, we shuffled the words in the original sentence (A) to create modified sentences (A_1 & A_2) and fed them together with the original second sentence (B) to a RoBERTa-based classifier for predictions. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both sentences. That is, the top-3 pairs of words that were assigned the largest cross-sentence weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles.

RTE example. Groundtruth: “entailment”		
P	About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.	“entailment” 0.90
H	About 33.5 miilion people live in Tokyo.	
P	About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.	“entailment” 0.79
H_1	people in miilion 33.5 live Tokyo About.	
P	About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.	“entailment” 0.80
H_2	33.5 in people About live Tokyo miilion.	

Figure A4: Each RTE example contains a pair of premises and hypotheses i.e. (P, H). We shuffled the words in the original hypothesis H to create modified hypotheses (H_1 & H_2) and fed them together with the original premise (P) to a RoBERTa-based classifier for predictions. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both the premise and hypothesis. That is, the top-3 pairs of words that were assigned the largest premise-to-hypothesis weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles.

Task Name	Task Type	Label	GLUE dev-set processing				`dev-r`
			(a) dev set	(b) step 1	(c) step 2	(d) step 3	Total
`CoLA`	Acceptability	"unacceptable" "acceptable"	322 721	287 675	154 638	154 154	308
`RTE`	NLI	"not entailment" "entailment"	131 146	131 145	72 127	72 72	144
`QQP`	Paraphrase	"not duplicate" "duplicate"	25,545 14,885	22,907 14,000	20,943 12,683	12,683 12,683	25,366
`MRPC`	Paraphrase	"not equivalent" "equivalent"	129 279	129 279	101 255	101 101	202
`SST-2`	Sentiment	"negative" "positive"	428 444	427 443	402 420	402 402	804
`QNLI`	NLI	"not entailment" "entailment"	2,761 2,702	2,741 2,690	2,500 2,527	2,500 2,500	5,000
`STS-B`	Similarity	N/A	1,500	1,498	N/A	N/A	1,498

Table A4: The number of examples per class before (a) and after each of the three filtering steps to produce `dev-r` sets (described in Sec. 2.3.2) for RoBERTa-based classifiers. For each task, we repeated the same procedure for three sets of classifiers, for BERT-, RoBERTa-, ALBERT-based classifiers, respectively.

Model	Task	`dev-r`	`dev-s`	`dev-s` performance				Word-Order Sensitivity			
		performance	baseline	2-noun swap	1-gram	2-gram	3-gram	2-noun swap	1-gram	2-gram	3-gram
RoBERTa$_{\text{large}}$	`CoLA`	100	50	70.80	51.40	55.62	57.98	**0.58**	**0.97**	**0.89**	**0.84**
	`RTE`	100	50	82.29	73.85	80.42	83.75	0.35	0.52	0.39	0.33
	`SST-2`	100	50	98.24	83.71	88.16	90.43	0.04	0.33	0.24	0.19
	`MRPC`	100	50	98.54	85.53	88.64	90.49	0.03	0.29	0.23	0.19
	`QQP`	100	50	87.13	86.84	90.65	92.60	0.26	0.26	0.19	0.15
	`QNLI`	100	50	95.26	91.12	95.20	96.46	0.09	0.18	0.10	0.07
	`STS-B`	90.43	N/A	88.95	85.47	87.20	87.98	N/A	N/A	N/A	N/A

Table A5: Accuracy of all models on `dev-s` examples (created by shuffling n-grams and swapping 2 nouns) and their Word-Order Sensitivity scores ($\in [0, 1]$) across seven GLUE tasks. STS-B is a regression task and thus not comparable in word-order sensitivity with the other tasks, which are binary classification.

	GLUE dev set						
Task	CoLA (Acc)	RTE (Acc)	QQP (Acc)	MRPC (Acc)	SST-2 (Acc)	QNLI (Acc)	STS-B (Spearman Corr)
RoBERTa$_{\text{base}}$	82.56	72.20	91.12	87.25	94.50	92.57	90.17
ALBERT$_{\text{base}}$	81.21	72.20	90.25	87.99	91.40	91.78	90.82
BERT$_{\text{base}}$	81.89	64.25	90.81	85.54	92.09	91.38	88.49
RoBERTa$_{\text{large}}$	65.30	80.87	91.62	88.48	96.44	94.45	90.44
Average	82.78	72.38	90.95	87.32	93.61	92.55	89.98

Table A6: The dev-set performance of models finetuned from three different BERT "base" variants (12 self-attention layers and 12 heads) and one RoBERTa "large" model (24 self-attention layers and 16 heads) on seven GLUE tasks. These results match either those reported by original papers, Huggingface 2020 or GLUE leaderboard.

RevCore: Review-augmented Conversational Recommendation

Yu Lu[†], **Junwei Bao**[‡(🖂)], **Yan Song**[†♯], **Zichen Ma**[†],
Shuguang Cui[†♯], **Youzheng Wu**[‡], **Xiaodong He**[‡]
[†]The Chinese University of Hong Kong (Shenzhen)
[‡]JD AI Research [♯]Shenzhen Research Institute of Big Data
[†]`{yulu1,zichenma1}@link.cuhk.edu.cn`
[‡]`{baojunwei,wuyouzheng1,xiaodong.he}@jd.com`
[†♯]`{songyan,shuguangcui}@cuhk.edu.cn`

Abstract

Existing conversational recommendation (CR) systems usually suffer from insufficient item information when conducted on short dialogue history and unfamiliar items. Incorporating external information (e.g., reviews) is a potential solution to alleviate this problem. Given that reviews often provide a rich and detailed user experience on different interests, they are potential ideal resources for providing high-quality recommendations within an informative conversation. In this paper, we design a novel end-to-end framework, namely, <u>Rev</u>iew-augmented <u>C</u>onversational <u>Re</u>commender (**RevCore**), where reviews are seamlessly incorporated to enrich item information and assist in generating both coherent and informative responses. In detail, we extract sentiment-consistent reviews, perform review-enriched and entity-based recommendations for item suggestions, as well as use a review-attentive encoder-decoder for response generation. Experimental results demonstrate the superiority of our approach in yielding better performance on both recommendation and conversation responding.[1]

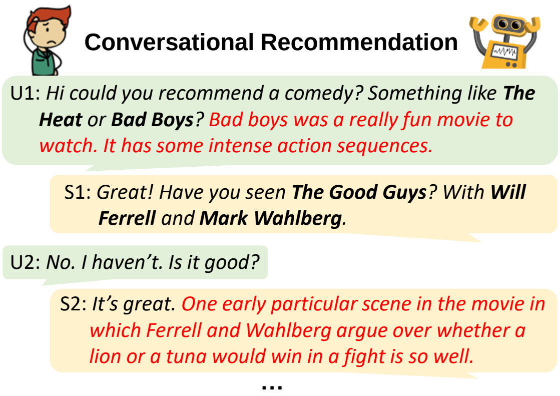

U1: *Hi could you recommend a comedy? Something like* ***The Heat*** *or* ***Bad Boys****? Bad boys was a really fun movie to watch. It has some intense action sequences.*

S1: *Great! Have you seen* ***The Good Guys****? With* ***Will Ferrell*** *and* ***Mark Wahlberg****.*

U2: *No. I haven't. Is it good?*

S2: *It's great. One early particular scene in the movie in which Ferrell and Wahlberg argue over whether a lion or a tuna would win in a fight is so well.*

...

Figure 1: An illustrative example of a user-system conversation on movie recommendation. The additional sentiment-matched reviews are in red. Items (movies) and entities (e.g., actors) are in bold.

1 Introduction

With the increasing popularity of intelligent assistants in users' daily lives, how to effectively help users find information or finish specific tasks, such as recommendation and booking, has tremendous commercial potential. Therefore, conversational recommendation (CR) systems have attracted widespread attention for being a tool providing users potential items of interest through dialogue-based interactions. Though existing studies (Sun and Zhang, 2018; Zhang et al., 2018; Lei et al., 2020) proposed to integrate recommender and dialogue components for providing user-specific suggestions through conversations, CR remains challengeable because (i) typical dialogues are short and lack sufficient item information for user preference capturing (Chen et al., 2019; Zhou et al., 2020), and (ii) difficulties exist in generating informative responses with item-related descriptions (Shao et al., 2017; Ghazvininejad et al., 2018; Wang et al., 2019b). Thus, recently, external information in the form of structured knowledge graphs (KG) is introduced to enhance item representations by using rich entity information in KG (Chen et al., 2019; Zhou et al., 2020). While KG-based methods improve CR to some extent, they are still limited in (i) worse versatility resulted from a high cost of KG construction; and (ii) inadequate integration of knowledge and response generation (Lin et al., 2020).

Given that, nowadays, users are greatly encouraged to share their consumption experience (e.g., restaurant, traveling, movie, etc.), reviews are easily accessed over the internet. Such reviews often provide rich and detailed user comments on different factors of interest, which are crucial in suggest-

---

(🖂) Corresponding Author

[1] Our code will release in `https://github.com/JD-AI-Research-NLP/RevCore`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1161–1173
August 1–6, 2021.

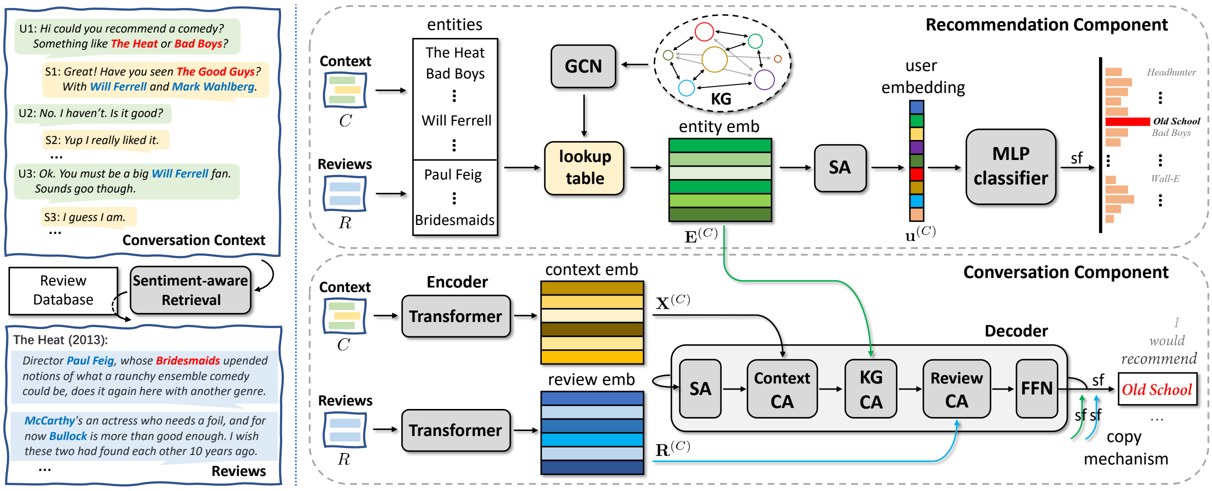

Figure 2: The overview of the proposed method in a movie recommendation scenario, where "emb", "SA", "CA", and "sf" denote embedding, self-attention, cross-attention, and softmax operation, respectively.

ing recommendations to particular users. Thus one can treat reviews as promising external sources for higher-quality recommendations in a conversation. As an example shown in Figure 1, the CR system may be unfamiliar with the mentioned items from the user, resulting in an uninformative response "*It's great.*", thus the chat does not help with recommendation owing to lacking necessary knowledge. In addition, another factor resulting in users' lower acceptance rates to the recommendations is that elaborations on the suggestion are seldom given, which can be alleviated with more explanatory or descriptive utterances after referring to reviews.

Therefore, in better linking external knowledge to recommendation in dialogues, in this paper, we propose a novel framework, Review-augmented Conversational Recommender (**RevCore**), to enhance CR by additional review data. In doing so, we firstly analyze user's utterances with their sentiment polarities and then retrieve reviews for the items mentioned by the user with keeping their sentiment matching the utterances (e.g., they should be both positive or negative). The obtained reviews are thus recommendation-beneficial (He et al., 2015; Hariri et al., 2011) because they are given by the ones who have seen/used and also show interests (or with no interests) in the mentioned items. Afterward, we incorporate the selected reviews into dialogue history, from which the CR system can learn user preference from review-enriched item information. In addition, we also use the sentiment-coordinated reviews to enhance the dialogue response generation, where a review-attentive decoder introduces item information from selected reviews to generate coherent and informative responses. To the best of our knowledge, it is the first time that the aforementioned CR issues have been addressed through incorporating external reviews. Experimental results on a widely used benchmark dataset (Li et al., 2018) show that RevCore is superior on both recommendation accuracy and conversation quality. Further analyses are also performed to confirm the effectiveness of RevCore in an appropriate manner of introducing reviews to CR.

2 The Proposed Framework

We present the proposed Review-augmented Conversational Recommender (**RevCore**) with its overview illustrated in Figure 2, where there are three main components, i.e., the review retrieval module, the recommendation component, and the conversation component. The review retrieval module takes a conversation context C as the input and outputs the selected review set R from the review database $\mathcal{R}_{db}$ which contains all reviews. The context $C = \{s_t\}_{t=1}^{N}$ consists of all utterances s_t of the dialogue history given by the user and system in turns, and the review set R includes all review sentence $r \in \mathcal{R}_{db}$ retrieved according to the contexts in previous turns. With C and R as the input, the recommendation component outputs a set of items from the candidate item set $\mathcal{Z}$ as the recommendation. The dialogue component also accepts C and R as input, and outputs an utterance $s_{t+1} = \{w_i\}_{i=1}^{M}$ as the response, where w_i is the i_{th} word and M the length of s_{t+1}. The output s_{t+1} is added to the context of the next turn.

We first introduce how to retrieve proper reviews from a database in the following Section 2.1. Then our solutions to the recommendation and conversation tasks are described in Section 2.2 and Section 2.3 respectively, along with detailed illustrations of how reviews enhance both two tasks. Without the loss of generality, our method is introduced in a movie recommendation scenario.

2.1 Review Retrieval

To help dialogue with reviews, given $\mathcal{R}_{db}$, it is of great importance to retrieve proper ones. The reasons are two folds: (i) non-relevant reviews may result in harmful effects to the user representation; (ii) reviews with inconsistent altitudes inject noise into the conversation, which impedes generating coherent responses. Then, a preliminary retrieval is to search in $\mathcal{R}_{db}$ for proper reviews according to the mentioned item in the conversation context C. For review filtering, we design a sentiment-aware retrieval module. The sentiment value $v \in [0,1]$ of each review r can be captured by a transformer-based sentiment predictor:

$$v = \mathrm{Sentiment}(r), \; r \in \mathcal{R}_{db}, \tag{1}$$

where $\mathrm{Sentiment}(\cdot)$ denotes sentiment prediction, and v can be viewed as how well the movie is liked in this review r. Similarly, the sentiment of a response to this movie can also be obtained in this way. As a result, reviews that possess similar sentiment polarity v^* with the response are selected.

Considering helpful reviews are usually long paragraphs, we only retain part of them, one sentence, for each mentioned movie. Given a context C, there exist two manners to select the sentences $r^{(C)}$ from the raw reviews, word-wisely (or phrase-wisely) and sentence-wisely. The first one randomly chooses some words or phrases to form each "sentence", and one whole sentence is directly selected at random in a second way. Despite the expense of sentence fluency, the first manner enjoys much variability due to the extensive word/phrase combinations. The process to obtain $r^{(C)}$ can be formulated as follows:

$$r^{(C)} = \mathrm{Retrieve}(\mathcal{R}_{db}, V, v^*), \tag{2}$$

where $\mathrm{Retrieve}(\cdot)$ denotes the retrieval operation and V is the set of all v. The obtained $r^{(C)}$ is added into the review set R.

With the retrieved review sentence, one way of incorporation is to briefly insert it right behind the sentence where the item is, as in Fig. 1. However, it may cause the perturbation to the conversational consistency by interrupting the original dialogue. Thus we seamlessly incorporate the review embedding into the conversation component, which is described in Section. 2.3. More importantly, the review sentence serves as a brief introduction or explanation to the mentioned movie. It enriches user information for personalized recommendations and introduces external knowledge for more informative recommendation responses.

2.2 Review-augmented Recommendation

The recommender component is constructed based on a KG-based framework (Zhou et al., 2020), with all entities in the context are extracted to generate the embedding of a user profile. In our method, the retrieved reviews work on enriching entity information so that the user embedding can be augmented to promote recommendation accuracy.

Similar to the approach in Zhou et al. (2020), a candidate entity embedding dictionary $\mathcal{E}$ is constructed first by using GNN to learn entity representations from KG, e.g., DBpedia (Auer et al., 2007). Given a context C, all entities $E^{(C)}$ in it are extracted. Then the embedding vectors of them are looked up from $\mathcal{E}$ and concatenated into a matrix $\mathbf{E}^{(C)} \in \mathbb{R}^{l^{(C)} \times d}$, where $l^{(C)}$ is the number of entities in the context C, and d denotes the embedding dimension. Next, the entity embedding $\mathbf{E}^{(C)}$ is aggregated into a user embedding vector $\mathbf{u}^{(C)}$, through a self-attention layer (SA) as follows:

$$\begin{aligned} \mathbf{u}^{(C)} &= \mathbf{E}^{(C)} \cdot \boldsymbol{\alpha}, \\ \boldsymbol{\alpha} &= \mathrm{softmax}(\mathbf{b}^\top \cdot \tanh(\mathbf{W}_\alpha \mathbf{E}^{(C)})), \end{aligned} \tag{3}$$

where $\boldsymbol{\alpha}$ is the attention weight vector, and $\mathbf{W}_\alpha$ and $\mathbf{b}$ are the parameter matrix and vector for linear projection and bias. Given $\mathbf{u}^{(C)}$, a multi-layer perceptron (MLP) and a softmax operation are adopted to obtain the recommendation prediction $\mathbf{p} \in \mathbb{R}^L$, where L is the number of candidate movies:

$$\mathbf{p} = \mathrm{softmax}(\mathrm{MLP}(\mathbf{u}^{(C)})). \tag{4}$$

To learn parameters in the recommender component, a cross-entropy loss $\mathcal{L}_{rec}$ between the prediction $\mathbf{p}$ and the target movie category is computed:

$$\mathcal{L}_{rec} = -\frac{1}{M} \sum_{i=1}^{M} \log p_i^*, \tag{5}$$

where M is the number of recommendations and p_i^* is the prediction probability of the target category in the i_{th} recommendation.

The pipeline described above suffers from the entity sparsity in dialogue history, resulted from the dataset construction process, where annotators are inevitably unfamiliar with some movies. Retrieved reviews can act to enrich the $E^{(C)}$ by adding more entity words.The process of obtaining review-enriched entities can be formulated as:

$$\begin{aligned} E_C^{(C)} &= \text{Extract}(C), \\ E_R^{(C)} &= \text{Extract}(R^{(C)}), \\ E^{(C)} &= \{E_C^{(C)}, E_R^{(C)}\}, \end{aligned} \tag{6}$$

where $\text{extract}(\cdot)$ defines the entity extraction operation, and $E_c^{(C)}$, $E_r^{(C)}$ denotes entities extracted from the context and retrieved review, respectively. Based on the review-enriched entities, the user embedding is expected to be better represented to produce a more precise recommendation.

2.3 Review-augmented Response Generation

Reviews can also augment the response generation in the conversation component. We build an encoder-decoder framework to handle the generation task. Retrieved reviews and context are encoded separately first, for the purpose of maintaining the dialog consistency. In the decoding stage, the review embedding is fused via an attention layer to generate informative responses. Considering that a good modeling of the input plays an important role to achieve an outstanding model performance (Mikolov et al., 2013; Song and Shi, 2018; Peters et al., 2018; Devlin et al., 2019; Song et al., 2021) and transformer-based approaches have achieved state-of-the-art in many NLP tasks (Vaswani et al., 2017; Chen et al., 2019; Zhou et al., 2020; Chen et al., 2020a; Joshi et al., 2020; Wang et al., 2020; Tian et al., 2020), we adopt two transformers as the encoders for context and reviews. Given a context C and the retrieved reviews R, the context embedding $\mathbf{X}^{(C)}$ and review embedding $\mathbf{R}^{(C)}$ are first obtained:

$$\begin{aligned} \mathbf{X}^{(C)} &= \text{Transformer}_{\boldsymbol{\theta}_X}(C), \\ \mathbf{R}^{(C)} &= \text{Transformer}_{\boldsymbol{\theta}_R}(R), \end{aligned} \tag{7}$$

where $\boldsymbol{\theta}_X, \boldsymbol{\theta}_R$ are parameters in these two transformers. The decoding stage takes them and the entity embedding $\mathbf{E}^{(C)}$ as inputs of attention layers. These attention layers aim to fuse the external information from KG and reviews R into the context information, inspired by the work of Zhou et al. (2020). Given the decoding output of last time unit $\mathbf{Y}^{i-1}$, the current one $\mathbf{Y}^i$ is generated by:

$$\begin{aligned} \mathbf{A}_0^i &= \text{MHA}(\mathbf{Y}^{i-1}, \mathbf{Y}^{i-1}, \mathbf{Y}^{i-1}), \\ \mathbf{A}_1^i &= \text{MHA}(\mathbf{A}_0^i, \mathbf{X}^{(C)}, \mathbf{X}^{(C)}), \\ \mathbf{A}_2^i &= \text{MHA}(\mathbf{A}_1^i, \mathbf{E}^{(C)}, \mathbf{E}^{(C)}), \\ \mathbf{A}_3^i &= \text{MHA}(\mathbf{A}_2^i, \mathbf{R}^{(C)}, \mathbf{R}^{(C)}), \\ \mathbf{Y}^i &= \text{FFN}(\mathbf{A}_3^i), \end{aligned} \tag{8}$$

where $\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ represents the multi-head attention function (Vaswani et al., 2017), which takes a query, key, and value as input:

$$\begin{aligned} &\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{h}_1, \cdots, \text{h}_h] \cdot \mathbf{W}^o, \\ &\text{h}_i = \text{Attention}(\mathbf{QW}_i^{(q)}, \mathbf{KW}_i^{(k)}, \mathbf{VW}_i^{(v)}), \end{aligned} \tag{9}$$

where $[\cdot]$ represents the concatenation operation, h is the number of heads, and $\mathbf{W}_i$ is the parameter matrix to learn. $\text{FFN}(\cdot)$ in Equation 8 defines a fully-connected feed-forward network, which comprises of two linear layers with one ReLU activation layer in between:

$$\text{FFN}(\boldsymbol{x}) = \text{ReLU}(\boldsymbol{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2. \tag{10}$$

As presented above, information is injected progressively into the decoding stage, from the original context at first, then related entity information in KG, and finally reviews, which contain detailed item-related information.

To complete the generation, the decoder output $\mathbf{Y}^i$ is processed through a softmax operation to predict the token distribution. Apart from the conversational consistency required in the chit-chat task, the CR system also expects recommendation-related responses, which usually contain relevant entities and descriptive keywords. So a copy mechanism is further adapted to introduce vocabulary bias and thus increase the informativeness in the generation. Given the previous generated sub-sequence $\{y_{i-1}\} = y_1, y_2, \cdots, y_{i-1}$, the generation probability y_i of the next token can be computed as:

$$\begin{aligned} \Pr(y_i|\{y_{i-1}\}) = \Pr_1(y_i|\mathbf{Y}_i) + \Pr_2(y_i|\mathbf{Y}_i, G) + \\ \Pr_3(y_i|\mathbf{Y}_i, R), \end{aligned} \tag{11}$$

where $\Pr_1(\cdot)$ is a generation probability function over the vocabulary, with $\mathbf{Y}^i$ as the input. G and R represents the knowledge graph and reviews we use. $\Pr_2(\cdot)$, $\Pr_3(\cdot)$ are copy probability functions from KG entities and reviews, respectively, implemented by a standard copy mechanism (Gulcehre et al., 2016) (computing the distributions over the KG words or review words). Both probability functions are implemented with a softmax operation. To learn the response generation in the dialogue component, we set a cross-entropy loss:

$$\mathcal{L}_{gen} = -\frac{1}{N}\sum_{t=1}^{N} \log\big(\Pr(s_t|s_1, \cdots, s_{t-1})\big), \quad (12)$$

where N is the number of turns, s_t represents the t_{th} utterance in the conversation.

To train the whole model, it includes three steps: (i) pre-training the sentiment predictor in the review retrieval module; (ii) training the recommender component by minimizing $\mathcal{L}_{rec}$; (iii) training the dialogue component by minimizing $\mathcal{L}_{gen}$.

3 Experiment Settings

3.1 Dataset

REDIAL (Li et al., 2018) is a widely-used dataset of real-world conversations around the theme of providing movie recommendations generated by the human in seeker-recommender pairs. REDIAL contains 10,021 conversations related to 64,362 movies, split into training, validation, and test sets using a ratio of 8:1:1[2]. To construct a review database, we crawled 30 reviews for each movie from IMDb[3] website, which is one of the most popular and authoritative movie databases. Each review can be queried according to the corresponding movie along with its rating and helpful score provided by IMDb. In practice, we select the 30 reviews with the highest helpful scores for each movie to guarantee the high quality of collected reviews. Other manners of selecting the 30 reviews are described and compared in the second part of Section 4.4.

3.2 Implementation Details

The maximum lengths of context and response are set to 256 and 30, respectively. Transformers for review encoding in dialogue generation and sentiment prediction use the same hyper-parameters with the context encoder. For sentiment polarity in the reviews, we threshold on the star-rating to getting sentiment polarity with the threshold set to 5. In the dialogue context, the sentiment polarity is obtained according to users' attitude to the mentioned entity in utterances, which is provided by the REDIAL dataset. Other settings are kept consistent with Zhou et al. (2020) for fair comparison[4]. Besides, the "review sentence" is selected according to the sentiment value and in a sentence-wise manner, and the token number of incorporated review sentences is set to 20, considering the balance between the original source and external source. We add the retrieved review sentences after the mentioned items in the dialogue component training to guide it to generate review-aware responses. The sentiment predictor for reviews is trained on the collected reviews. The sentiment predictor for dialog context is trained on the IMDb Movie Reviews Dataset (Maas et al., 2011) and then finetuned on the REDIAL dataset.

3.3 Baselines

Evaluated on the REDIAL dataset, we compare our approach with a variety of competitive baselines from previous studies listed as follows:

- **Trans** (Vaswani et al., 2017) applies a encoder-decoder framework based on transformer for generation, and applies a transformer encoder to encode context information for recommendation.
- **Redial** (Li et al., 2018) builds a conversation component based on a hierarchical encoder-decoder architecture, and its recommender component is implemented by an auto-encoder extended with a RNN-based sentiment analysis module.
- **KBRD** (Chen et al., 2019) adopts DBpedia-enhanced contextual items or entities to construct user profile for recommendation. The KG-enhanced user profile also serves as word bias for the transformer-based generation module.
- **KGSF** (Zhou et al., 2020) uses MIM (Viola and Wells III, 1997) to align the semantic spaces of two KGs. The user embedding is obtained from the aligned representations of words and items for recommendation. The generation module follows a transformer encoder and a fused KG enhanced decoder.

3.4 Evaluation Metrics

Our method is evaluated on both the recommendation and conversation tasks. The evaluation metric for recommendation is Recall@k (R@k, k = 1, 10, 50), which indicates whether the predicted top-k items contain the ground truth recommendation

---

[2] More statistics presents in appendix A.

[3] `https://www.imdb.com/`

[4] More details of hyper-parameters and training strategies are described in Appendix B; the size of different models and their inference speed are reported in Appendix C.

Models	**R@1**	**R@10**	**R@50**
Redial	2.4	14.0	32.0
KBRD	3.1	15.0	33.6
KGSF	3.9	18.3	37.8
RevCore (−KG)	4.2	22.7	43.3
RevCore (+KG)	**6.1**	**23.6**	**45.4**

Table 1: Results on the recommendation task. Best results are in bold.

Models	**Dist-2**	**Dist-3**	**Dist-4**	**PPL**
Trans	0.148	0.151	0.137	17.0
Redial	0.225	0.236	0.228	28.1
KBRD	0.263	0.368	0.423	17.9
KGSF	0.289	0.434	0.519	**9.8**
RevCore (−KG)	0.373	0.527	**0.615**	10.7
RevCore (+KG)	**0.424**	**0.558**	0.612	10.2

Table 2: Results on the conversation task. Best results are in bold.

provided by human recommenders. Conversation evaluation comprises automatic and human evaluation. The metrics for automatic evaluation are perplexity (PPL) (Jelinek et al., 1977) and distinct n-gram (Dist-n, $n = 2, 3, 4$) (Li et al., 2016). Perplexity is a measurement for the fluency of natural language, where lower perplexity refers to higher fluency. Distinct n-gram is a measurement for the diversity of generated utterances. Specifically, we use distinct 3-gram and 4-gram at the sentence level to evaluate the diversity. The main purpose of our dialog component is a successful recommendation rather than imitating the ground truth responses. Therefore, we provide annotators to manually evaluate the results instead of using BLEU scores. The annotators evaluate the quality of generated dialogue responses from 3 aspects, i.e., coherence, fluency, and informativeness, with each score ranging from 0 to 1.

4 Results and Analysis

4.1 Evaluation on Recommendation Task

For the recommendation task, we adopt Recall@k (R@1, R@10, R@50) for evaluation. As the results summarized in Table 1, our approach outperforms all competitive baselines and achieves 5.9% R@1, 24.0% R@10, and 41.3% R@50, which is the state-of-the-art performance on the REDIAL dataset.[5] Compared with KGSF, RevCore (+KG) achieves significant improvements, with R@1 score improved about 156% (absolutely 2.2), R@50 score improved about 129% (absolutely 4.5), and R@50 score improved about 120% (absolutely 7.6).

We also evaluate the performance of RevCore (−KG), which means the construction of $\mathcal{E}$ removes relation between entities. Instead, an embedding matrix is randomly initialized and learned to represent each entity, without using the GNN-based embedding. In this version, the external knowledge source we introduce is reduced to review only. As the result in the last two rows of Table 1, RevCore (−KG) can achieve competitive results with RevCore (+KG), and outperform KGSF that uses two KGs. According to our observation, although the learning of entity representation is made harder without structured knowledge graphs, the enrichment of dialogue history by reviews makes up the embedding learning. It demonstrates that incorporating reviews is a meaningful method to improve the recommendation in the conversation. We hope this result inspire further research.

4.2 Evaluation on Conversation Task

Automatic Evaluation The results of automatic evaluation on the REDIAL dataset summarize in Table 2. The proposed RevCore outperforms all competitive baselines and achieves significant improvements over most of the automatic metrics. Compared with KGSF, all of the Dist-n scores are significantly lifted, namely, by +0.14 for Dist-2, +0.11 for Dist-3, and +0.08 for Dist-4, which demonstrates our method is effective to generate diverse utterances. Besides, RevCore (+KG) achieves a comparable PPL score with KGSF. It validates our claim that the review incorporation in our method does not cause a decline in generation fluency. The lower PPL score of RevCore (+KG) possibly relates to the high fluency contained in incorporated reviews that carefully induct by website users. For the version of RevCore (−KG), it achieves higher Dist-n scores than KGSF and only results in a slight drop compared with RevCore (+KG). It demonstrates that reviews compared with KG bring more diversity as a richer and more accessible external source.

Human Evaluation We adopt human evaluation on a random selection of 100 multi-turn dialogues

---

[5] We report the performance of different models on the validation sets in Appendix D and the mean and standard deviation of the test set results in Appendix E.

Models	**Coherence**	**Fluency**	**Informat**
Trans	0.189	0.226	0.115
Redial	0.225	0.455	0.228
KBRD	0.263	0.468	0.283
KGSF	0.324	0.502	0.332
RevCore (−KG)	0.556	0.493	0.682
RevCore (+KG)	**0.601**	**0.567**	**0.718**

Table 3: Human evaluation results. "Informat" denotes informativeness. Best results are in bold.

Models	**Dist-2**	**Dist-3**	**Dist-4**	**PPL**
RevCore (+KG)	**0.424**	**0.558**	**0.612**	10.2
−revCP	0.353	0.443	0.503	**10.0**
−revRA	0.328	0.428	0.516	13.2
−revEN	0.394	0.534	0.586	10.8

Table 4: Ablation study on the conversation task.

from the testing set. Given one dialogue context, each generated response is scored ranging from 0 to 1, with a higher value indicating a more coherent, fluent, and informative utterance. The final result is calculated as the average score of three annotators, as summarized in Table 3. The proposed RevCore (with or without KG) is consistently better than all the baselines, especially on the metric of informativeness in a large margin. It further proves the effectiveness of our method, and also verifies its superiority in numerical results.

Ablation Study We demonstrate the contribution of each part on the conversation task by constructing an ablation study based on three variants of our complete model, including: (1) RevCore (−revCP) by removing the copy mechanism for reviews, (2) RevCore (−revRA) by removing the review attention layers from the transformer decoder, and (3) RevCore (−revEN) by removing the sentiment-aware review encoder (the reviews share the same encoder with the context). As shown in Table 4, first, all the techniques are useful to improve the final performance in generating diversified utterances. Besides, the copy mechanism and the review attention layers seem to be more important in conversation diversity. One of the potential reasons is that these two components are directly related to the decoding stage. Separated encoders for review and context lead to a slight increment, which shows that sharing a common encoder is an alternative solution.

Len	Recommendation			Conversation		
	R@1	**R@10**	**R@50**	**Dist-3**	**Dist-4**	**PPL**
10	4.5	21.8	37.0	0.491	0.564	**10.2**
20	**6.1**	**23.6**	**45.4**	**0.558**	**0.612**	**10.2**
30	4.7	22.0	41.3	0.263	0.426	13.1
40	5.8	21.3	41.3	0.289	0.439	14.0
50	5.1	22.3	40.8	0.304	0.520	13.8

Table 5: Performance of RevCore when incorporating review sentences with different length (Len). Best results are in bold.

4.3 Case Study

In this part, we present a visualized example to illustrate how our model works in practice, as shown in Figure 3. First, the sentiment-aware review retrieval module guarantees the coherence of incorporating reviews to some extent, for example in Figure 3, negative comments (the last row for the movie *The Notebook*) are filtered out considering the positive attitude in the original utterance. Secondly, incorporated reviews exactly enrich the context for better recommendation. As seen in the first yellow frame, many entities like "*Roshan Andrews*" mentioned in the review are added into the entity set. Note that some other entities are also added from the reviews that are incorporated into users' utterances as described in Section 2, which is not visualized here but brings recommendation accuracy improvement as well. Last but not least, the generated responses are more informative to use more varied expressions like "*the magic spell*" and "*the sultry dance*". Besides, they uncover more details related to the movie that acts as explanatory sentences that make recommendations accepted more easily and naturally.

4.4 Discussion

Longer Review, Better Performance? In our basic setting, each retrieved review sentence is formed by 20 words. We conduct a series of experiments by setting the length of retrieved review sentences as 10, 20, 30, 40, and 50 to inspect the effect of review length. The results of using different lengths are shown in Table 5, among which 20 is the best across all metrics. An interesting finding is that continually increasing the review length makes no benefits after reaching 20. Over introducing external text may suppress original text, thus 20 is a better choice to keep the balance between them.

Dialog Generation	Entities	Recommend
U1: *I'm great! Looking for a good romantic movie.*	romantic	
S1: *Oh nice! I love a good romantic movie called The Notebook. Notebook brought to life with the magical spell of Director Roshan Andrews. Have you seen it? Another my favorite one would be Dirty Dancing. Jennifer Grey and Patrick Swayze make movie magic with the sultry dance moves.*	add: **The Notebook** R. Andrews **Dirty Dancing** J. Grey P. Swayze	The Notebook Dirty Dancing Moulin Rouge! Before Sunrise An Officer ... ...
U2: *Oh yes, I have seen it, that was a tear jerker. I loved Notebook too. I never saw Dirty Dancing.*	add: None	
S2: *Another classic one I like would be Sleepless in Seattle. One of my favorite Tom Hanks movies. I don't own many romantic comedies. But this one is in my collection. If you haven't seen it, you can check it out.*	add: classic **Sleepless in ...** Tom Hanks comedies	Sleepless in ... Splash Love Story Udayananu ... ...
U3: *Wonderful! I 'll have to check that out.*	End	

Review Retrieval

The Notebook
5: *Story wise it will be an all new experience for Malayalam ...*
9: *Notebook is the story of teenagers, brought to life with ...*
2: *There is a lengthy episode involving a student, Feroze, ...*

Dirty Dancing
9: *Jennifer Grey and Patrick Swayze make movie magic ...*
9: *This is one of those rare films that needs a 30 year break ...*
6: *What movie has all the elements of a guilty pleasure? I ...*

Sleepless in Seattle
7: *One of my favorite Tom Hanks movies. I don't own many ...*
9: *You could have had a big, romantic, tear-jerking moment ...*
0: *I see a lot of comments about romance ... so a woman ...*

Figure 3: Case study. $U(i)$ (green) and $S(i)$ (yellow) represent user and system, respectively. In the "Dialogue Generation", items are marked in blue font, explanatory sentences are in red. Items are in bold font in "Entities" frames. In "Recommend" frames, a darker color represents a higher probability. "Review Retrieval" gives retrieved review examples, with their sentiment value (0-9) at the most left, and the selected reviews are in bold.

Method	Recommendation			Conversation		
	R@1	**R@10**	**R@50**	**Dist-3**	**Dist-4**	**PPL**
iCorpus	2.1	8.5	20.3	0.430	0.555	11.2
R-H-W	2.3	10.7	25.2	0.307	0.347	10.9
C-H-W	5.4	23.3	42.9	0.471	0.559	13.1
C-H-S	4.2	22.4	39.4	0.534	0.586	11.4
C-S-S	**6.1**	**23.6**	**45.4**	**0.558**	**0.612**	**10.2**

Table 6: Results of various retrieval strategies. Three characters from left to right represent three factors: (i) C/R: correctly/randomly matched movie-review pairs; (ii) S/H: ranking by sentiment value or helpful score respectively; (iii) S/W: sentence/word-wise manner. "iCorpus" indicates using irrelevant corpus. Best results are in bold.

Appropriate Reviews Help More? The "review sentence" are obtained from a 3-stage process, namely, searching item-matched reviews from the database, ranking them by the helpful score or sentiment value, and constructing "review sentence" word-wisely or sentence-wisely. Therefore, we conduct control experiments to inspect these three factors. As shown in Table 6, (i) using reviews randomly matched with items (R-H-W) results in significantly lower R@k and Dist-n scores; (ii) ranking by sentiment value (C-S-S) leads to better performance across all metrics than by helpful score (C-H-S), which demonstrates the necessity of using sentiment-aware review retrieval; (iii) the sentence-wise manner (C-H-S) gets a lower PPL than the word-wise one (C-H-W), which is reasonable because the incorporated reviews made up of random words causes the fluency loss. Besides, another experiment is conducted to verify the necessity of using a movie-review database. A food-review database is constructed as a topic-irrelevant corpus (iCorpus), which results in the lowest R@k yet not bad Dist-n scores. It shows that despite the response diversity brought by the external corpus, the unrelated entities from another domain have negative impacts on the recommendation accuracy.

5 Related Work

Recommender systems have emerged as a separate research area and now play an indispensable role in daily social lives. Traditional recommender systems tend to work statically, primarily relying on content-based approaches or the collaborative filtering hypothesis (Resnick et al., 1994; Pazzani and Billsus, 2007; Wang et al., 2019b), which assumes that similar users may have similar interests. Afterward, more sophisticated methods using neural networks are proposed and prove effective. For instance, neural factorization machines (He and Chua, 2017) and deep interest networks (Zhou et al., 2018) are used to estimate user preferences based on historical user-item interactions. Graphs are adopted in Wang et al. (2019b,a) to model complex relations among users, items, and attributes for a better representation of data.

In recent years, major advances made in dialog systems (Dodge et al., 2016; Yan et al., 2016; Benni et al., 2016; Bordes et al., 2017; Song et al., 2020) and structured knowledge-based info-seeking technics including question answering (Bao et al., 2014, 2016; Yin et al., 2015; Yih et al., 2015; Shao et al., 2019) and question generation (Serban et al., 2016; Bao et al., 2018; Dušek et al., 2020) have encouraged the development of *conversational recommendation* systems, which dynamically obtain user preferences through interactive conversation with users. Multiple datasets have been constructed (Dodge et al., 2016; Li et al., 2018; Kang et al., 2019) to facilitate the study of this task. Li et al. (2018) collect a standard human-to-human multi-turn dialog dataset focusing on providing movie recommendations. Based on these datasets, various approaches are proposed to address different issues in CR systems. Specifically, external information is introduced to alleviate the cold-start problem, including knowledge bases (Wang et al., 2018), social networks (Daramola et al.), and knowledge graphs (Chen et al., 2019). Christakopoulou et al. (2016) use bandit-based explore-exploit strategy to minimize the number of user queries. Liu et al. (2020) conduct multi-goal planning to make a proactive conversational recommendation over multi-type dialogues. A multi-view method is proposed in Chen et al. (2020b) for the explainable conversational recommendation. The work of Pecune et al. (2020) builds a socially aware CR system engaging its users through a rapport-building dialogue to improve users' perception.

Different from all aforementioned previous work, we offer an alternative to AIG with an augmented conversational recommendation system by incorporating reviews that highly relevant to items. Particularly, our model is able to learn better user representations from a review-enriched dialogue context, which enables a high-quality recommendation and response generation.

6 Conclusion

In this paper, we proposed a novel CR framework with review augmentation, including a sentiment-aware retrieval module, a recommender exploiting the review-enriched user profile, an encoder for enhancing semantic embedding of selected reviews, and a review attentive decoder to integrate review information for dialogue response generation. Experimental results show that our approach achieves consistent and significant improvements of both recommendation and dialogue responding over baselines, and is able to generate informative responses without losing fluency and coherence.

Acknowledgement

The work was supported in part by the Key Area R&D Program of Guangdong Province with grant No. 2018B030338001, by the National Key R&D Program of China with grant No. 2018YFB1800800, by Shenzhen Outstanding Talents Training Fund, by Guangdong Research Project No. 2017ZT07X152, and by the National Key Research and Development Program of China under Grant No. 2018YFB2100802.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, Osaka, Japan. The COLING 2016 Organizing Committee.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, Maryland. Association for Computational Linguistics.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5020–5027. AAAI Press.

Dominique Benni, Francis Chauvet, and Alain Guillot. 2016. Human-machine dialog system. US Patent 9,411,370.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Guimin Chen, Yuanhe Tian, and Yan Song. 2020a. Joint aspect extraction and sentiment analysis with directional graph convolutional networks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 272–279, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards knowledge-based recommender dialog system. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1803–1813, Hong Kong, China. Association for Computational Linguistics.

Zhongxia Chen, Xiting Wang, Xing Xie, Mehul Parsana, Akshay Soni, Xiang Ao, and Enhong Chen. 2020b. Towards explainable conversational recommendation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2994–3000. ijcai.org.

Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 815–824. ACM.

Olawande Daramola, Gleb Sizov, and Pinar Öztürk. Improving trust in travel recommendations using a conversational textual cbr framework.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5110–5117. AAAI Press.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Negar Hariri, Bamshad Mobasher, Robin Burke, and Yong Zheng. 2011. Context-aware recommendation based on review mining. In *ITWP@ IJCAI*.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1661–1670. ACM.

Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 355–364. ACM.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul Crook, Y-Lan Boureau, and Jason Weston. 2019. Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1951–1961, Hong Kong, China. Association for Computational Linguistics.

Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, pages 304–312. ACM.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9748–9758.

Xiexiong Lin, Weiyu Jian, Jianshan He, Taifeng Wang, and Wei Chu. 2020. Generating informative conversational response using recurrent knowledge-interaction and knowledge-copy. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 41–52, Online. Association for Computational Linguistics.

Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Towards conversational recommendation over multi-type dialogs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1036–1049, Online. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer.

Florian Pecune, Lucile Callebert, and Stacy Marsella. 2020. A socially-aware conversational recommender system for personalized recipe recommendations. In *Proceedings of the 8th International Conference on Human-Agent Interaction*, pages 78–86.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany. Association for Computational Linguistics.

Bo Shao, Yeyun Gong, Junwei Bao, Jianshu Ji, Guihong Cao, Xiaola Lin, and Nan Duan. 2019. Weakly supervised multi-task learning for semantic parsing. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3375–3381. ijcai.org.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219, Copenhagen, Denmark. Association for Computational Linguistics.

Yan Song and Shuming Shi. 2018. Complementary learning of word embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4368–4374. ijcai.org.

Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. 2020. Summarizing medical conversations via identifying important utterances. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 717–729, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 235–244. ACM.

Yuanhe Tian, Yan Song, and Fei Xia. 2020. Supertagging Combinatory Categorial Grammar with attentive graph convolutional networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Paul Viola and William M Wells III. 1997. Alignment by maximization of mutual information. *International journal of computer vision*, 24(2):137–154.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 417–426. ACM.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019a. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 165–174. ACM.

Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019b. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5329–5336.

Yingyao Wang, Junwei Bao, Guangyi Liu, Youzheng Wu, Xiaodong He, Bowen Zhou, and Tiejun Zhao. 2020. Learning to decouple relations: Few-shot relation classification with entity-guided attention and confusion-aware training. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5799–5809, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. 2016. DocChat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 516–525, Berlin, Germany. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1301–1310. ACM.

Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 177–186. ACM.

Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1059–1068. ACM.

Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving conversational recommender systems via knowledge graph based semantic fusion. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1006–1014. ACM.

A Statistics for Conversation Dataset and Reviews

Conversations in the REDIAL dataset consist of 163,820 utterances, of which 15.80% have reviews added. The vocabulary size of REDIAL is increased by 13.14% (from 23,356 to 26,427). Among the mentioned 6,927 movies in all conversations, 40% of them are randomly chosen and linked with reviews to keep the balance between the original source and external source. We count the ratio of "disliked" movies by the recommender to explain the improvements brought by doing sentiment-aware retrieval when incorporating reviews. We also show the ratio of unseen movies by the recommender to show the need of introducing reviews to "talk more". Comprehensive statistics are listed in Table 7.

B Experiment Details

Hyper-parameter Settings For a fair comparison, most hyper-parameters are kept consistent with KGSF. We did not search for more hyper-parameters combinations to achieve additional improvements apart from our main idea. The shared hyper-parameters include: embedding dimension

Items	Statistic	Items	Statistic
Dialogues	10,021	Cnd Movies	64,368
Utterances	163,820	Mnt Movies	6,924
+reviews	25,884	+reviews	2,711
Disliked	4.9%	Not Seen	31.9%
Liked	81.2%	Seen	61.3%
Did not say	13.9%	Did not say	6.8%

Table 7: Statistics for the REDIAL dataset with incorporated reviews. “Cnd” denotes “Candidate”, and “Mnt” denotes “Mentioned” to indicate the movies that mentioned in the conversations.

Models	Param	Tra Time	Inf Time
KGSF	130.51	4979.88	65.30
RevCore (−KG)	71.12	1045.65	59.57
RevCore (+KG)	133.30	3308.28	51.46

Table 8: Comparison of three models on the number of parameters (million), training (“Tra”) time for 30 epochs (second), and inference (“Inf”) time (second).

set as 128 in the recommender component and 300 in the dialogue component, the layer number of both GNN in the KG module as 1, the batch size as 32, word embedding initialization via word2vec[6], the optimizer as Adam, the learning rate as 0.001, the epoch number as 30, etc.

Training Strategies To train the whole model, three steps are included: (i) pre-training the sentiment predictor in the review retrieval module; (ii) training the recommender component by minimizing $\mathcal{L}_{rec}$; (iii) training the dialogue component by minimizing $\mathcal{L}_{gen}$. In the first step, the predictor takes each sentence in the review as input and outputs the sentiment, with the corresponding rating set as the label. In the second and third steps, our implementation refers to the training algorithm for the KGSF model. It first pre-trains the parameters in KG for entity representation by minimizing the Mutual Information Maximization loss between two KG embedding, then trains the recommender component by minimizing the recommendation loss and also updating the parameters in the KG module, and finally the dialogue component by minimizing the generation loss with all other modules' parameters “frozen”.

Metrics	RevCore (+KG)		RevCore (−KG)	
	Val	**Test**	**Val**	**Test**
R@1	6.13	6.11	4.55	4.19
R@10	23.49	23.62	23.35	22.71
R@50	40.65	45.43	45.33	43.28
Dist-2	0.418	0.424	0.410	0.373
Dist-3	0.582	0.558	0.582	0.527
Dist-4	0.675	0.612	0.668	0.615
PPL	10.89	10.24	10.14	10.69

Table 9: Validation (Val) and test results on the REDIAL dataset of RevCore (+KG) and RevCore (−KG).

Metrics	RevCore (+KG)		RevCore (−KG)	
	Mean	**Devi**	**Mean**	**Devi**
R@1	5.70	± 0.67	3.75	± 0.13
R@10	22.80	± 1.81	21.53	± 0.68
R@50	40.75	± 2.18	44.68	± 0.43
Dist-2	0.394	± 0.039	0.373	± 0.031
Dist-3	0.551	± 0.062	0.527	± 0.051
Dist-4	0.633	± 0.073	0.616	± 0.062

Table 10: Mean and deviation of recall rates (%) and distance scores for RevCore (+KG) and RevCore (−KG).

C Model Size and Running Speed

The model size and running speed of KGSF, RevCore (+KG), and RevCore (−KG) are all listed in Table 8. Note that all three models are implemented with Pytorch[7], trained for 30 epochs, and experimented on NVIDIA A100-SXM4 for 5 times to compute the average running time.

D Results on the Validation Set

We present the validation result of RevCore with and without KG on the REDIAL dataset as a reference for reproducing. All validation results are shown in Table 9, with test results as well.

E Mean and Standard Deviation

We implement the major experiment 4 times to inspect the mean and standard deviation of the performance of RevCore across all metrics. The reported results in the paper of both recommendation accuracy and conversation quality are the mean results. Results are shown in Table 10.

---

[6] `https://radimrehurek.com/gensim/models/word2vec.html`

[7] `https://pytorch.org/`

Awakening Latent Grounding from Pretrained Language Models for Semantic Parsing

Qian Liu[†][*], **Dejian Yang**[§], **Jiahui Zhang**[†][*], **Jiaqi Guo**[◊][*], **Bin Zhou**[†], **Jian-Guang Lou**[§]
[†]Beihang University, Beijing, China
[§]Microsoft Research, Beijing, China
[◊]Xi'an Jiaotong University, Xi'an, China
[†]`{qian.liu, 17231043, zhoubin}@buaa.edu.cn`; [◊]`jasperguo2013@stu.xjtu.edu.cn`
[§]`{dejian.yang, jlou}@microsoft.com`

Abstract

Recent years pretrained language models (PLMs) hit a success on several downstream tasks, showing their power on modeling language. To better understand and leverage what PLMs have learned, several techniques have emerged to explore syntactic structures entailed by PLMs. However, few efforts have been made to explore grounding capabilities of PLMs, which are also essential. In this paper, we highlight the ability of PLMs to discover which token should be grounded to which concept, if combined with our proposed erasing-then-awakening approach. Empirical studies on four datasets demonstrate that our approach can awaken latent grounding which is understandable to human experts, even if it is not exposed to such labels during training. More importantly, our approach shows great potential to benefit downstream semantic parsing models. Taking text-to-SQL as a case study, we successfully couple our approach with two off-the-shelf parsers, obtaining an absolute improvement of up to 9.8%.

1 Introduction

Recent breakthroughs of **P**retrained **L**anguage **M**odels (**PLMs**) such as BERT (Devlin et al., 2019) and GPT3 (Brown et al., 2020) have demonstrated the effectiveness of self-supervised learning for a range of downstream tasks. Without being guided by structural information in training, PLMs show the potential for learning implicit syntactic structures and language semantic, which can be transferred to other tasks. To better understand and leverage what PLMs have learned, several work has emerged to probe or induce syntactic structures from PLMs. According to prior studies (Rogers et al., 2020), most existing work focuses on syntactic structures such as part of speech (Liu et al., 2019), constituency tree (Wu et al., 2020) and dependency tree (Hewitt and Manning, 2019; Jawahar et al., 2019), paying much less attention on language semantics (Tenney et al., 2019). However, as well known, semantic information is essential for high-level tasks like machine reading comprehension (Wang and Jiang, 2019).

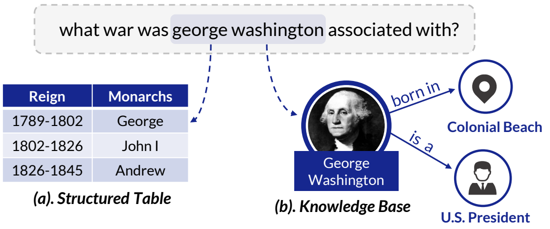

Figure 1: Typical scenarios for grounding, here the linguistic tokens "george washington" can be grounded into different real-world concepts.

Regarding to language semantics, an important branch is grounding, which is overlooked by most previous work. Broadly speaking, grounding means "*connecting linguistic symbols to real-world perception or actions*" (Roy, 2005). It is generally thought to be important for a variety of tasks, such as video descriptions (Zhou et al., 2019), visual question answering (Zhu et al., 2016) and semantic parsing (Guo et al., 2019). In this paper, we focus on single-modal scenarios, where grounding refers more specifically to mapping linguistic tokens into a real-world concept described in natural language. As shown in Figure 1, "george washington" can be grounded into either a cell value in a structured table, or an entity in knowledge bases.

In single-modal scenarios, grounding is especially important for semantic parsing, the task of translating a natural language sentence into its corresponding executable logic form. For earlier work, grounding is essential since earlier work almost conceptualized semantic parsing as grounding an

---

[*] Work done during an internship at Microsoft Research. The first three authors contributed equally.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1174–1189
August 1–6, 2021. ©2021 Association for Computational Linguistics

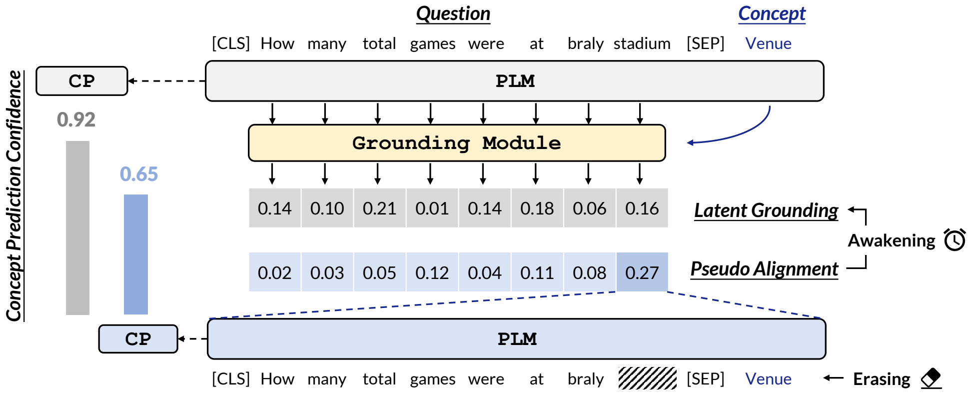

Figure 2: The illustration of ETA, which consists of a PLM module, a **C**oncept **P**rediction (CP) module and a grounding module. Two models (gray and blue) are drawn here for illustration purposes, and they are indeed the same. The model training involves three steps: (1) The concept prediction module is trained to predict the confidence of any concept occurring in a given question (**Left**). (2) The erasing mechanism erases tokens in the question sequentially, feeds them into CP, and obtains the confidence differences (e.g., $0.92 - 0.65 = 0.27$) as the pseudo alignment. Here we only demonstrate the process related to "stadium" (**Bottom Right**). (3) The pseudo alignment is employed to awaken the latent grounding, i.e., to supervise the grounding module (**Top Right**). We show only one concept "Venue" for the sake of brevity, which in practice is a sequence of concepts.

utterance to a task-specific meaning representation (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2013; Cheng et al., 2017). As for modern approaches based on the encoder-decoder architecture, grounding also plays an important role and considerable work has demonstrated the positive effect of it (Guo et al., 2019; Dong et al., 2019; Liu et al., 2020a; Wang et al., 2020b; Chen et al., 2020). Despite its success, existing grounding methods mainly relied on heavy manual efforts like high-quality lexicons (Reddy et al., 2016) or ad-hoc heuristic rules like n-gram matching (Guo et al., 2019), suffering from poor flexibility. To explore more flexible methods, researchers recently tried a data-driven way: they collected grounding annotations as supervision to train grounding models (Li et al., 2020a; Lei et al., 2020; Shi et al., 2020). However, this modeling flexibility in their approaches requires expensive annotations of grounding, which most of the time are not available.

To alleviate the above issues, we present a novel approach **E**rasing-**t**hen-**A**wakening (ETA)[1]. It is inspired by recent advances in interpretable machine learning (Samek et al., 2017), where the importance of individual pixels can be quantified with respect to the classification decision. Similarly, our approach firstly quantifies the contribution of each word with respect to each concept, by erasing it and probing the variation of concept prediction decisions (elaborated later). Then it employs these contributions as pseudo labels to awaken latent grounding from PLMs. In contrast to prior work, our approach only needs supervision of concept prediction, which can be easily derived by downstream tasks (e.g., text-to-SQL) instead of full grounding supervision. Empirical studies on four datasets demonstrate that our approach can awaken latent grounding which is understandable to human experts. It is highly non-trivial because our approach is not exposed to any human-annotated grounding label in training. More importantly, we find that the grounding can be easily coupled with downstream models to boost their performance, and the absolute improvement is up to 9.8%. In summarization, our contribution is as three-fold:

1. To the best of our knowledge, we are the first one to highlight and demonstrate the possibility of awakening latent grounding from PLMs.

2. We propose a novel weakly supervised approach erasing-then-awakening, to awaken latent grounding from PLMs. Empirical studies on four datasets demonstrate that our approach can awaken latent grounding which is understandable to human experts.

3. Taking text-to-SQL as a case study, we successfully couple our approach with two off-the-shelf parsers. Experimental results on two benchmarks show the effectiveness of our approach on boosting downstream performance.

---

[1] Our code is available at `https://github.com/microsoft/ContextualSP`

2 Method: Erasing-then-Awakening

In the task of grounding, we are given a question $\mathbf{x} = \langle x_1, \cdots, x_N \rangle$ and a concept set $\mathcal{C} = \{c_1, \cdots, c_K\}$, where each concept consists of several tokens. The goal of grounding is to find out tokens (also known as mentions) in $\mathbf{x}$ which are relevant to concepts in $\mathcal{C}$. Generally, the grounding procedure learns to create a $N \times K$ matrix, which we call *latent grounding*. In some cases, a set of pairs is needed, of which each one explicitly shows a token and a concept is grounded. We call this kind of pairs as *grounding pairs* below.

As illustrated in Figure 2, our model consists of a PLM module, a CP module and a grounding module. In this section, we first present the training procedure of ETA, which at a high-level involves three steps: (1) Train an auxiliary concept prediction module. (2) Erase tokens in a question to obtain the concept prediction confidence differences as pseudo alignment. (3) Awaken latent grounding from PLMs by applying pseudo alignment as supervision. Then we introduce the procedure to produce grounding pairs in inference.

2.1 Training a Concept Prediction Module

Given $\mathbf{x}$ and $\mathcal{C}$, the goal of the concept prediction module is to identify if each concept $c_k \in \mathcal{C}$ is mentioned or not in the question $\mathbf{x}$. Although it does not seem to be directly related to grounding, it is a pre-requisite for the erasing mechanism, which will be elaborated later. As for c_k's supervision $l_k \in \{0, 1\}$, it is the weak supervision ETA relies on, and can be readily obtained through downstream task signals. Taking text-to-SQL as an illustration, each database schema (i.e., table, column and cell value) in an annotated SQL can be considered as mentioned in the question ($l_k = 1$), with others as negative examples ($l_k = 0$).

Once the supervision is prepared, the CP module is trained to conduct binary classification over the representation of each concept. As done in previous work (Hwang et al., 2019), we first concatenate the question and all concepts into a sequence as input to the PLM module. As illustrated in Figure 2, the input sequence starts with `[CLS]`, with the question and each concept being separated by `[SEP]`. Then, the sequence is fed into the PLM module to produce deep contextual representations over each position. Denoting $\langle \mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N \rangle$ and $\langle \mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_K \rangle$ as the token representations and concept representations, they can be obtained by:

$$\{\mathbf{q}_n\}_{n=1}^{N}, \{\mathbf{e}_k\}_{k=1}^{K} = \mathrm{PLM}\big(\texttt{[CLS]}, \mathbf{x}, \{\texttt{[SEP]}, c_k\}_{k=1}^{K}\big), \quad (1)$$

where $\mathbf{q}_n$ and $\mathbf{e}_k$ correspond to the representations at the position of n-th question token and the first token in c_k respectively. Finally, each concept representation $\mathbf{e}_k$ is passed to a classifier to predict if it is mentioned in $\mathbf{x}$ as:

$$p_k = \texttt{Sigmoid}(\mathbf{W}_l\, \mathbf{e}_k), \quad (2)$$

where $\mathbf{W}_l$ is a learnable parameter. p_k is the probability of c_k mentioned in the question, which is referred to by *concept prediction confidence* below.

2.2 Erasing Question Tokens

Once the concept prediction module is converged, we apply an erasing mechanism to assist in the following awakening phase. It follows a similar idea from the interpretable document classification (Arras et al., 2016), where a word is considered important for the document classification if removing it and classifying the modified document results in a strong decrease of the classification score. In our case, a token is considered highly relevant to certain concepts if there is a large drop in these concept prediction confidences after erasing the token. Therefore, we need the above mentioned concept prediction module to provide a reasonable concept prediction confidence.

Concretely, as shown in Figure 2, the erasing mechanism erases the input sequentially, and feeds each erased input into the PLM module and the subsequent CP module. For example, with x_n being substituted by a special token `[UNK]`, we can obtain an erased input as $\texttt{[CLS]}, x_1, \cdots, x_{n-1}, \texttt{[UNK]}, x_{n+1}, \cdots, c_K$. Denoting $\hat{p}_{n,k}$ the concept prediction confidence for c_k after erasing x_n, we believe the difference between $\hat{p}_{n,k}$ and p_k reveals c_k's relevance to x_n from a PLM's view. The confidence difference $\Delta_{n,k}$ can be obtained by $\Delta_{n,k} = l_k \cdot \max(0, p_k - \hat{p}_{n,k})$. Repeating the above procedure on the input question sequentially, $\Delta \in \mathbb{R}^{N \times K}$ is filled completely.

2.3 Awakening Latent Grounding

As mentioned above, we believe Δ reflects the relevance between each token and each concept from a PLM's view. Therefore, we could directly use Δ as ETA's output. However, according to our preliminary study, the method performs poorly and cannot produce high-quality alignment[2]. Different from directly using Δ, we employ it to "awaken" the latent grounding. To be specific, we introduce a grounding module upon representations of the PLM module and train it using Δ as pseudo labels (i.e., pseudo alignment). The grounding module first obtains grounding scores $g_{n,k}$ between each question token x_n and each concept c_k based on their deep contextual representations $\mathbf{q}_n$ and $\mathbf{e}_k$ as:

$$g_{n,k} = \frac{\mathbf{W}_e \mathbf{e}_k \cdot (\mathbf{W}_q \mathbf{q}_n)^T}{\sqrt{d}}, \quad (3)$$

where $\mathbf{W}_e, \mathbf{W}_q$ are learnable parameters and d is the dimension of $\mathbf{e}_k$. Then it normalizes the grounding scores into latent grounding $\boldsymbol{\alpha}$ as:

$$\boldsymbol{\alpha}_{n,k} = \frac{\exp(g_{n,k})}{\sum_i \exp(g_{i,k})}. \quad (4)$$

Finally, the grounding module is trained to maximize the likelihood with Δ as the weight:

$$\sum_n \sum_k \Delta_{n,k} \cdot \log \boldsymbol{\alpha}_{n,k}. \quad (5)$$

2.4 Producing Grounding Pair

Repeating erasing and awakening iteratively for epochs until the grounding module converges, we can readily produce grounding pairs. Formally, we aim to obtain a set of pairs, where each pair $\langle x_n, c_k \rangle$ indicates that x_n is grounded to c_k. Noticing c_k may contain several tokens, we keep all probabilities in $\boldsymbol{\alpha}_{\cdot,k}$ which exceeds $\tau/|c_k|$, where τ is a threshold and $|c_k|$ is the number of tokens in c_k. Also, taking into account that x_n should be grounded to only one concept, we keep only the highest probability over $\boldsymbol{\alpha}_{n,\cdot}$. Finally, for each pair $\langle x_n, c_k \rangle$, it is thought to be a grounding pair if $\boldsymbol{\alpha}_{n,k}$ is kept and $p_k \geq 0.5$, otherwise it is not.

3 Experiments

In this section, we conduct experiments to evaluate if the latent grounding awakened by ETA is understandable to human experts. Here we accomplish the evaluation by comparing the grounding pairs produced by ETA with human annotations.

3.1 Experimental Setup

Datasets We select two representative grounding tasks where human annotations are available: *schema linking* and *entity linking*. Schema linking is to ground questions into database schemas, while entity linking is to ground questions into entities of knowledge bases. For schema linking, we select SPIDER-L (Lei et al., 2020) and SQUALL (Shi et al., 2020) as our evaluation benchmarks. As mentioned in §2.1, the supervision for our model is obtained from SQL queries. As for entity linking, we select WebQSP$_{\text{EL}}$and GraphQ$_{\text{EL}}$(Sorokin and Gurevych, 2018). The supervision for our model is obtained from SPARQL queries in a similar way.

Evaluation For schema linking, as done in previous work (Lei et al., 2020), we report the micro-average precision, recall and F1-score for both columns (Col$_P$, Col$_R$, Col$_F$) and tables (Tab$_P$, Tab$_R$, Tab$_F$). For entity linking, we report the weak matching precision, recall and F1-score for entities (Ent$_P$, Ent$_R$, Ent$_F$). The weak matching metric is a commonly used metric in previous work (Sorokin and Gurevych, 2018), which considers a prediction as correct whenever the correct entity is identified and the predicted mention boundary overlaps with the ground truth boundary. More details can be seen in §A.

Baselines For schema linking, we consider four strong baselines. (1) **N-gram Matching** enumerates all n-gram ($n \leq 5$) phrases in a natural language question, and links them to database schemas by fuzzy string matching. (2) **SIM** computes the dot product similarity between each question token and schema using their PLM representations without fine-tuning, to explore grounding capacities of unawakened PLMs. (3) **CONTRAST** learns by comparing the aggregated grounding scores of mentioned schemas with unmentioned ones in a contrastive learning style, as done in Liu et al. (2020b). Concretely, in training, CONTRAST is trained to accomplish the same concept prediction task as our approach. With a similar architecture to the Receiver used in Liu et al. (2020b), it first computes the similarity score between each token and each concept, and then uses max pooling to aggregate the similarity scores of a concept over an utterance into a concept prediction score. Finally, a margin-based loss is used to encourage the baseline to give higher concept prediction scores on mentioned concepts than unmentioned concepts. (4) **SLSQL$_{\text{L}}$ & ALIGN$_{\text{L}}$**. SLSQL$_{\text{L}}$ (ALIGN$_{\text{L}}$) is a learnable schema linking module[3] proposed in

---

[2] More experimental results can be found in §3.3.

[3] SLSQL and ALIGN use multi-task learning to simultaneously learn schema linking and SQL generation.

Model	SPIDER-L						SQUALL		
	Col$_P$	Col$_R$	Col$_F$	Tab$_P$	Tab$_R$	Tab$_F$	Col$_P$	Col$_R$	Col$_F$
N-gram Matching	61.4	69.1	65.1	78.2	69.6	73.6	71.6	50.8	59.4
SIM + BERT	16.6	8.0	10.8	8.5	11.6	9.8	13.9	18.0	15.7
CONTRAST + BERT	83.7	68.4	75.3	**84.0**	76.9	80.3	47.9	31.2	37.8
ETA + BERT	**86.1**	**79.3**	**82.5**	81.1	**85.3**	**83.1**	**77.3**	**62.4**	**69.0**
SLSQL$_{\text{L}}$ + BERT$^{\heartsuit}$ (Lei et al., 2020)	82.6	82.0	82.3	80.6	84.0	82.2	–	–	–
ALIGN$_{\text{L}}$ + BERT$^{\heartsuit}$ (Shi et al., 2020)	–	–	–	–	–	–	79.2	72.8	75.8

Table 1: Experimental results on schema linking dev sets. $^{\heartsuit}$ means the model uses schema linking supervision, while other learnable models use weak supervision. +BERT means using BERT as encoder, the same for Table 2.

Model	WebQSP$_{\text{EL}}$			GraphQ$_{\text{EL}}$(zero-shot)		
	Ent$_P$	Ent$_R$	Ent$_F$	Ent$_P$	Ent$_R$	Ent$_F$
Heuristic (Sorokin and Gurevych, 2018)	30.2	60.8	40.4	-	-	-
ETA + BERT	**76.6**	**72.5**	**74.5**	**43.1**	**42.1**	**42.7**
VCG$^{\heartsuit}$ (Sorokin and Gurevych, 2018)	82.4	68.3	74.7	54.1	30.6	39.0
ELQ + BERT$^{\heartsuit}$ (Li et al., 2020a)	90.0	85.0	87.4	60.1	57.2	58.6

Table 2: Experimental results on entity linking test sets. $^{\heartsuit}$ means the model uses entity linking supervision from WebQSP$_{\text{EL}}$, while ETA uses the weak supervision derived from WebQSP. Following previous work (Sorokin and Gurevych, 2018), we use GraphQ$_{\text{EL}}$only in the evaluation phase to test the generalization ability of our model.

SLSQL (ALIGN). Unlike our method, these two methods are trained with the full schema linking supervision. Please refer to Shi et al. (2020) and Lei et al. (2020) for more details. Notably, for baselines which require a threshold, we tuned their thresholds based on dev sets for fair comparison.

For entity linking, we compare ETA with three powerful methods. (1) **Heuristic** picks the most frequent entity among the candidates found by string matching over Wikidata. (2) **VCG** (Sorokin and Gurevych, 2018) aggregates and mixes contexts of different granularities to perform entity linking. (3) **ELQ** (Li et al., 2020a) uses a bi-encoder to perform entity linking in one pass, achieving state-of-the-art performance on WebQSP$_{\text{EL}}$and GraphQ$_{\text{EL}}$. VCG and ELQ utilize entity linking supervision in training, while ETA does not.

Implementation For schema linking we follow the procedure in §2.4 to produce grounding pairs to evaluate, while for entity linking we further merge adjacent grounding pairs to produce span-level grounding pairs. We implement ETA in Pytorch (Paszke et al., 2019). With respect to PLMs in experiments, we use the uncased BERT-base (BERT)[4] and BERT-large (BERT$_{\text{L}}$) from Transformers library (Wolf et al., 2020). As for the optimizer, we employ AdamW (Loshchilov and Hutter, 2019). More details (e.g., learning rate) of each experiment can be found in §C.1.

3.2 Experimental Results

Table 1 shows the experimental results on the schema linking task. As shown, our method outperforms all weakly supervised methods and heuristic-based methods by a large margin. For example, on SPIDER-L, ETA + BERT achieves an absolute improvement of 7.2% Col$_F$ and 2.8% Tab$_F$ over the best baseline CONTRAST. The same conclusion can be drawn from the experimental results on the entity linking task shown in Table 2. For instance, ETA + BERT can obtain a high Ent$_F$ up to 74.5% on WebQSP$_{\text{EL}}$, which is a satisfying performance for downstream tasks. All results above demonstrate the superiority of our approach on awakening latent grounding from PLMs. With respect to the reason that PLMs work well on both schema linking and entity linking, it may be because both schema linking and entity linking require text-based semantic matching (e.g., synonyms), which PLMs excel at.

Furthermore, it is very surprising that although not trained under fine-grained grounding supervision, our model is comparable with or slightly worse than the fully supervised models across

---

[4] Our approach is theoretically applicable to different PLMs. In this paper, we chose BERT as a representative and we leave exploration of different PLMs for future work.

Error Type	**Example Error**
Missed Grounding (43.1%)	How many **points** did arnaud demare receive? `GOLD:` **points**→ "`UCI world tour points`" `PRED:`
Technically Correct (21.0%)	Total **population** of millbrook first **nation**? `GOLD:` **population**→ "`Population`" `PRED:` **population**→ "`Population`"; **nation**→ "`Community`"
Partially Correct (15.8%)	Who was **the first winning captain**? `GOLD:` **the first**→ "`Year`"; **winning captain**→ "`Winning Captain`" `PRED:` **first**→ "`Year`"; **winning captain**→ "`Winning Captain`"
Wrong Grounding (10.1%)	Were the **matinee** and evening performances held **earlier** than the 8th anniversary? `GOLD:` **earlier**→ "`Date`" `PRED:` **matinee**→ "`Performance`"; **earlier**→ "`Date`"

Table 3: Four main error types made by Eta along with their proportions on Squall dataset.

datasets. For instance, on Spider-L, our model exceeds the fully supervised baseline SLSQL$_\text{L}$ by 0.9 points on Tab$_F$. On Squall, our model holds a slightly worse performance than the fully supervised baseline ALIGN$_\text{L}$. It is highly nontrivial since Contrast, the best weakly supervised baseline on Spider-L, is far from the fully supervised model on Squall, while our model has only a small drop. Besides, on WebQSP$_\text{EL}$and GraphQ$_\text{EL}$, although our model is inferior to the state-of-the-art model ELQ, it also achieves a comparable performance with the fully supervised baseline VCG. These results provide strong evidence that PLMs do have very good grounding capabilities, and our approach can awaken them from PLMs.

3.3 Model Analysis

In this section, we try to answer four interesting research questions via a thorough analysis: **RQ1.** Does the grounding capability come mainly from the PLM? **RQ2.** Is the awakening phase necessary? **RQ3.** Do larger PLMs have better grounding capabilities? **RQ4.** What are the remaining errors?

RQ1 There is a long term debate in literature about if knowledge is primarily learned by PLMs, when extra parameters are employed in analysis (Hewitt and Liang, 2019). Similarly, since our approach depends on extra modules (e.g., grounding module), it faces the same dilemma: how can we know whether the latent grounding is learnt from PLMs or extra modules? Therefore, we apply our approach to a randomly initialized Transformer encoder (Vaswani et al., 2017), to probe the grounding capability of a model that has not been pretrained. To make it comparable, the encoder has the same architecture as BERT. However, it only gets a 40% Col$_F$ on Squall, not even as good as the N-gram baseline. Considering it contains the same extra modules as Eta + BERT, the huge gap between it and Eta + BERT supports the opinion that the latent grounding is mainly learnt from PLMs. Meanwhile, one concern shared by our reviewers is the risk of supervision exposure during training of the concept prediction module. In other words, our approach may "steal" some supervision in the concept prediction module to achieve good performance on grounding. However, the above experiment demonstrates that a non-pretrained model is far from strong grounding capability even with the same concept prediction module. We hope the finding will alleviate the concern.

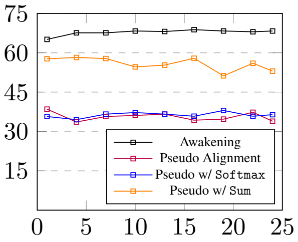

Figure 3: Col$_F$ score on the dev set of Squall at different training epochs. "Pseudo w/ `Softmax`" means normalizing pseudo alignment with `Softmax`, while "Pseudo w/ `Sum`" means normalizing through dividing each number by the sum of them.

RQ2 As mentioned in §2.3, the pseudo alignment Δ can also be employed as the model prediction. Therefore, we conduct experiments to verify if our proposed awakening phase is necessary. As shown in Figure 3, even with various normalization methods (e.g., `Softmax`), Δ does not produce satisfactory alignment. In contrast, our model consistently performs well. To investigate deeper, we conduct a careful analysis on Δ, and we are surprised to

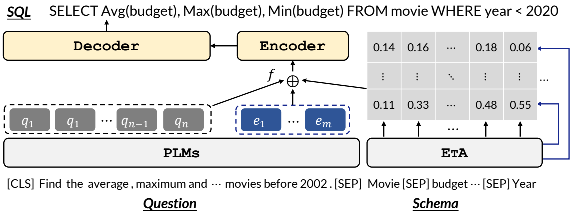

Figure 4: The illustration of the solution to couple ETA with downstream text-to-SQL parsers.

find that values of Δ are generally small and not as significantly different with each other as we would expect. Therefore, we believe the success of our approach stems from the fact that it encourages the grounding module to capture subtle differences and strength them.

RQ3 We apply our approach on BERT-large (BERT_L) and conduct experiments on SPIDER-L. The results show BERT_L brings an improvement of 2.5% Col_F and 0.5% Tab_F, suggesting the possibility of awakening better latent grounding from larger PLMs. Nevertheless, the improvement may also come from more parameters, so the conclusion needs further investigation.

RQ4 We manually examine 20% of our model's errors on the SQUALL dataset and summarize four main error types: (1) missed grounding - where our model did not ground any token to a concept, (2) technically correct - where our model was technically correct but the annotation was missing, (3) partially correct - where our model did not find all tokens of a concept, (4) wrong grounding - where the model produced incorrect grounding. As shown in Table 3, only a small fraction of errors are wrong grounding, indicating that the main challenge of our approach is recall rather than precision.

4 Case Study: Text-to-SQL

The ETA model is proposed for general-purpose uses and intends to enhance different downstream semantic parsing models. To verify it, we take the text-to-SQL task as a case study. In this section, we first present a general solution to couple ETA with different text-to-SQL parsers. Then, we conduct experiments on two off-the-shelf parsers to verify the effectiveness of ETA.

4.1 Coupling with Text-to-SQL Parsers

Inspired by Lei et al. (2020), we present a general solution to couple ETA with downstream parsers in Figure 4. As shown, we first obtain a schema-aware representation for each question token, by fusing the token representation and its related schema representation according to the latent grounding $\boldsymbol{\alpha} \in \mathbb{R}^{N \times K}$ (gray matrix in Figure 4). Specifically, given a token representation $\mathbf{q}_n$ and all schema representations $\langle \mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_K \rangle$, the schema-aware representation $\tilde{\mathbf{q}}_n$ for $\mathbf{q}_n$ can be computed as:

$$\tilde{\mathbf{q}}_n = \mathbf{q}_n \oplus \sum_k \boldsymbol{\alpha}_{n,k}\, \mathbf{e}_k. \tag{6}$$

Then we feed every $\tilde{\mathbf{q}}_n$ into a question encoder to generate hidden states, which are attended by a decoder to decode the SQL query. By contributing to the schema-aware representation, ETA is able to prompt the decoder to predict appropriate schemas during decoding. Notably, the encoder and decoder are not limited to specific modules, and we follow the paper settings in subsequent experiments.

Model	Dev		Test
	Ex.Match	Ex.Acc	Ex.Acc
ALIGN_P	37.8 ± 0.6	56.9 ± 0.7	46.6 ± 0.5
ALIGN_P + BERT	44.7 ± 2.1	63.8 ± 1.1	51.8 ± 0.4
ETA + BERT	$\mathbf{47.6} \pm 2.5$	$\mathbf{66.6} \pm 1.7$	$\mathbf{53.8} \pm 0.3$
$\text{ALIGN}^\heartsuit$	42.2 ± 1.5	61.3 ± 0.8	49.7 ± 0.4
$\text{ALIGN} + \text{BERT}^\heartsuit$	47.2 ± 1.2	66.5 ± 1.2	54.1 ± 0.2

Table 4: Ex.Match and Ex.Acc results on the dev and test set of WTQ. + BERT means using BERT to enhance encoder. $^\heartsuit$ means the model uses extra schema linking supervision. Both are the same for Table 5.

4.2 Experimental Setup

Datasets and Evaluation We conduct experiments on two text-to-SQL benchmarks: WikiTableQuestions(WTQ) (Pasupat and Liang, 2015)[5] and Spider (Yu et al., 2018b). Following previous work, we employ three kinds of evaluation metrics: *Exact Match* (Ex.Match), *Exact Set Match* (Ex.Set) and *Execution Accuracy* (Ex.Acc). Ex.Match evaluates the predicted SQL correctness by checking if it is equal to the ground-truth, while Ex.Set evaluates the structural correctness by checking the set match of each SQL clause in the predicted query with respect to the ground-truth. Ex.Acc evaluates the functional correctness of the predicted SQL by checking whether it yields the ground-truth answer.

---

[5] Note that the original WTQ only contains answer annotations, and here we use the version with SQL annotations provided by Shi et al. (2020). Our training data is a subset of the original train set, while the test data keeps the same.

Model	Dev	Test
GlobalGNN + BERT (Bogin et al., 2019)	52.7	47.4
EditSQL + BERT (Zhang et al., 2019)	57.6	53.4
IRNet + BERT (Guo et al., 2019)	61.9	54.7
IRNet v2 + BERT (Guo et al., 2019)	63.9	55.0
BRIDGE + BERT (Lin et al., 2020)	65.5	59.2
BRIDGE + BERT$_L$ (Lin et al., 2020)	70.0	65.0
RATSQL + BERT$_L$ (Wang et al., 2020a)	69.7	**65.6**
SLSQL$_P$ + BERT	57.4	-
SLSQL$_P$ + BERT$_L$	61.0	-
ETA + BERT	64.5	59.5
ETA + BERT$_L$	**70.8**	65.3
SLSQL + BERT$^{\heartsuit}$	60.8	55.7
SLSQL + BERT$_L$$^{\heartsuit}$	65.1	-
SLSQL + BERT (Oracle)$^{\heartsuit}$	72.4	-

Table 5: Ex.Set results on the dev and test set of Spider.

Baselines On WTQ, our baselines include ALIGN$_P$ and ALIGN, where the former is a vanilla attention based sequence to sequence model and the latter enhances ALIGN$_P$ with an additional schema linking task (Shi et al., 2020). Similarly, on Spider, our main baselines are SLSQL$_P$ and its schema linking enhanced version SLSQL (Lei et al., 2020). SLSQL$_P$ is made up of a question encoder and a two-step SQL decoder. In the first decoding step, a coarse SQL (i.e., without aggregation functions) is generated. Then the coarse SQL is used to synthesize the final SQL in the second decoding step. Here we also report the performance of SLSQL + BERT (Oracle), where the learnable schema linking module is replaced with human annotations in inference. It represents the maximum potential benefit of schema linking for the text-to-SQL task. Meanwhile, for a comprehensive comparison, we also compare our model with state-of-the-art models on the Spider benchmark[6]. We refer readers to their papers for details.

Implementation As for our approach, on WTQ, we employ ALIGN$_P$[7] as our base parser, while on Spider we select SLSQL$_P$[8] as our base parser. For both parsers, we try to follow the same hyperparameters as described in the paper to reduce other factors that may affect the performance. More implementation details can be found in §C.2.

4.3 Experimental Results

Table 4 and Table 5 show the experimental results of several methods on WTQ and Spider respectively. As observed, introducing ETA dramatically improves the performance of both base parsers, demonstrating its effectiveness on downstream tasks. Taking Spider as an illustration, our model ETA + BERT boosts SLSQL$_P$ + BERT by an absolute improvement 7.1% on the Ex.Set metric. As the PLM becomes larger (e.g., BERT$_L$), the improvement becomes more significant, up to 9.8%. Compared with state-of-the-art methods, our model ETA + BERT$_L$ also obtains a competitive performance, which is extremely impressive since it is based on a simple parser.

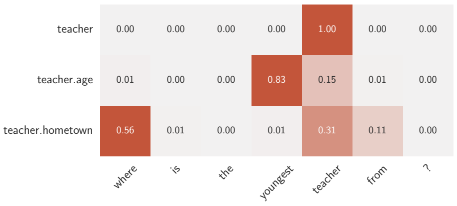

Figure 5: The latent grounding produced by ETA + BERT$_L$ for the question "*Where is the youngest teacher from?*".

More interestingly, on both datasets, our model can achieve similar even better performance compared to methods which employ extra grounding supervision. For instance, in comparison with SLSQL + BERT on Spider, our ETA + BERT outperforms it by 3.7%. Taking into account that SLSQL utilizes additional supervision, the performance gain is very surprising. We attribute the gain to two possible reasons: (1) The PLMs already learn latent grounding which is understandable to human experts. (2) Compared with training with strong schema linking supervision, training with weak supervision alleviates the issue of exposure bias, and thus enhance the generalization ability of ETA.

Table 6 presents the model predictions of ETA + BERT$_L$ on three real cases. As observed, ETA has learned the grounding about adjective (e.g., oldest → age), entity (e.g., where → hometown) and semantic matching (e.g., registered → student_enrolment). Meanwhile, grounding pairs provide us a useful guide to better understand the model predictions. Figure 5 visualizes the latent grounding for Q2 in Table 6, and more visualization can be found in §D.

5 Related Work

The most related work to ours is the line of inducing or probing knowledge in pretrained language mod-

---

[6] `https://yale-lily.github.io/spider`

[7] `https://github.com/tzshi/squall`

[8] `https://github.com/WING-NUS/slsql`

Question with Alignment	SQL with Alignment
1. Show name$_1$, country$_2$, age$_3$ for all singers$_4$ ordered by age$_3$ from the oldest$_3$ to the youngest.	SELECT name$_1$, country$_2$, age$_3$ FROM singer$_4$ ORDER BY age$_3$ DESC
2. Where$_1$ is the youngest$_2$ teacher$_3$ from?	SELECT hometown$_1$ FROM teacher$_3$ ORDER BY age$_2$ ASC LIMIT 1
3. For each semester$_1$, what is the name$_2$ and id$_3$ of the one with the most students registered$_4$?	SELECT semester_name$_2$, semester_id$_3$ FROM semesters$_1$ JOIN student_enrolment$_4$ ON semesters.semester_id = student_enrolment.semester_id GROUP BY semester_id$_3$ ORDER BY COUNT(*) DESC LIMIT 1

Table 6: The predicted grounding pairs and SQLs of our best model on three real cases from the Spider dev set. The question token and the schema with the same subscript are grounded.

els. According to the knowledge category, there are mainly two kinds of methods: one focuses on syntactic knowledge and the other pays attention to semantic knowledge. Under the category of syntactic knowledge, several work showed that BERT embeddings encoded syntactic information in a structural form and can be recovered (Lin et al., 2019b; Warstadt and Bowman, 2020; Hewitt and Manning, 2019; Wu et al., 2020). However, recent work also showed that BERT did not rely on syntactic information for downstream task performance, and thus doubted the role of syntactic knowledge (Ettinger, 2020; Glavas and Vulic, 2020). As for semantic knowledge, although it is less explored than syntactic knowledge, previous work showed that BERT contained some semantic information, such as entity types (Ettinger, 2020), semantic roles (Tenney et al., 2019) and factual knowledge (Petroni et al., 2019). Different from the above work, we focus on the grounding capability, an under-explored branch of language semantics.

Our work is also closely related to entity linking and schema linking, which can be viewed as subareas of grounding on specific scenarios. Given an utterance, entity linking aims at finding all mentioned entities in it using a knowledge base as candidate pool (Tan et al., 2017; Chen et al., 2018; Li et al., 2020a), while schema linking tries to find all mentioned schemas related to specific databases (Dong et al., 2019; Lei et al., 2020; Shi et al., 2020). Previous work generally either employed full supervision to train linking models (Li et al., 2020a; Lei et al., 2020; Shi et al., 2020), or treated linking as a minor pre-processing(Yu et al., 2018a; Guo et al., 2019; Lin et al., 2019a) and used heuristic rules to obtain the result. Our work is different from them since we optimize the linking model with weak supervision from downstream signals, which is flexible and practicable. Similarly, Dong et al. (2019) utilized downstream supervision to train their linking model. Compared with them using policy gradient, our method is more efficient since it directly learns the grounding module using pseudo alignment as supervision.

6 Conclusion & Future Work

In summary, we propose a novel weakly supervised approach to awaken latent grounding from pretrained language models via erasing. Only with downstream signals, our approach can induce latent grounding from pretrained language models which is understandable to human experts. More importantly, we demonstrate that our approach could be applied to off-the-shelf text-to-SQL parsers and significantly improve their performance. For future work, we plan to extend our approach to more downstream tasks such as visual question answering. We also plan to utilize our approach to improve the error locator module in existing interactive semantic parsing systems (Li et al., 2020b).

Acknowledgement

We would like to thank all the anonymous reviewers for their constructive feedback and useful comments. We also thank Tao Yu and Bo Pang for evaluating our submitted models on the test set of Spider. The first author Qian is supported by the Academic Excellence Foundation of Beihang University for PhD Students.

Ethical Considerations

This paper conducts experiments on several existing datasets covering the areas of entity linking, schema entity and text-to-SQL. All claims in this paper are based on the experimental results. Every experiment can be conducted on a single Tesla P100 or P40 GPU. No demographic or identity characteristics information is used in this paper.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. "what is relevant in a text document?": An interpretable machine learning approach. *CoRR*, abs/1612.07843.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-SQL parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Lihan Chen, Jiaqing Liang, Chenhao Xie, and Yanghua Xiao. 2018. Short text entity linking with fine-grained topics. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 457–466, New York, NY, USA. Association for Computing Machinery.

Sanxing Chen, Aidan San, Xiaodong Liu, and Yangfeng Ji. 2020. A tale of two linkings: Dynamically gating between schema linking and structural linking for text-to-SQL parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2900–2912, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-anonymous encoding for text-to-SQL generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5405–5414, Hong Kong, China. Association for Computational Linguistics.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Goran Glavas and Ivan Vulic. 2020. Is supervised syntactic parsing beneficial for language understanding? an empirical investigation. *CoRR*, abs/2008.06788.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 6943–6954, Online. Association for Computational Linguistics.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020a. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433–6441, Online. Association for Computational Linguistics.

Yuntao Li, Bei Chen, Qian Liu, Yan Gao, Jian-Guang Lou, Yan Zhang, and Dongmei Zhang. 2020b. "what do you mean by that?" a parser-independent interactive approach for enhancing text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6913–6922, Online. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019a. Grammar-based neural text-to-sql generation. *CoRR*, abs/1905.13326.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019b. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020a. How far are we from effective context modeling? an exploratory study on semantic parsing in context twitter. In *IJCAI*, pages 3580–3586.

Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. 2020b. You impress me: Dialogue generation via mutual persona perception. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1417–1427, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Deb Roy. 2005. Grounding words in perception and action: computational insights. *Trends in Cognitive Sciences*, 9(8):389 – 396.

W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1849–1864, Online. Association for Computational Linguistics.

Daniil Sorokin and Iryna Gurevych. 2018. Mixing context granularities for improved entity linking on question answering data across entity categories. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 65–75, New Orleans, Louisiana. Association for Computational Linguistics.

Chuanqi Tan, Furu Wei, Pengjie Ren, Weifeng Lv, and Ming Zhou. 2017. Entity linking for queries by searching Wikipedia sentences. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 68–77, Copenhagen, Denmark. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Chao Wang and Hui Jiang. 2019. Explicit utilization of general knowledge in machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2263–2272, Florence, Italy. Association for Computational Linguistics.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020b. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238, Online. Association for Computational Linguistics.

Alex Warstadt and Samuel R. Bowman. 2020. Can neural networks acquire a structural bias from raw linguistic data? *CoRR*, abs/2007.06761.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, pages 1050–1055. AAAI Press / The MIT Press.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Luowei Zhou, Yannis Kalantidis, Xinlei Chen, Jason J. Corso, and Marcus Rohrbach. 2019. Grounded video description. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6578–6587. Computer Vision Foundation / IEEE.

Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4995–5004. IEEE Computer Society.

A Evaluation Details

A.1 Schema Linking

Let Ω_{col} be a set $\{(c, q)_i | 1 \leq i \leq N\}$ which contains N gold (column-question token) tuples. Let $\overline{\Omega}_{col}$ be a set $\{(\overline{c}, \overline{q})_j | 1 \leq j \leq M\}$ which contains M predicted (column-question token) tuples. We define the precision(Col$_P$), recall(Col$_R$), F1-score(Col$_F$) as:

$$\frac{|\Gamma_{col}|}{|\overline{\Omega}_{col}|}, \frac{|\Gamma_{col}|}{|\Omega_{col}|}, \frac{2\mathrm{Col}_P\mathrm{Col}_R}{\mathrm{Col}_P + \mathrm{Col}_R}$$

where $\Gamma_{col} = \Omega_{col} \bigcap \overline{\Omega}_{col}$. The definitions of Tab$_P$, Tab$_R$, Tab$_F$ are similar. Note that the result reported in Table 8 of Shi et al. (2020) use a different evaluation metrics. Here we re-evaluate their model by the above mentioned metrics for fair comparison.

A.2 Entity Linking

Let $\Omega = \{(e, [q_s, q_e])_i | 1 \leq i \leq N\}$ be the gold entity-mention set and $\overline{\Omega} = \{(\overline{e}, [\overline{q_s}, \overline{q_e}])_j | 1 \leq j \leq M\}$ be the predicted entity-mention set, where e is the entity, q_e, q_s are the mention boundaries in the question q. In the weak matching setting, a prediction is correct only if the ground-truth entity is identified and the predicted mention boundaries overlap with the ground-truth boundaries. Therefore, the True-Positive prediction set is defined as:

$$\begin{aligned} \Gamma = \{e | (e, [q_s, q_e]) \in \Omega, (e, [\overline{q_s}, \overline{q_e}]) \in \overline{\Omega}, \\ [q_s, q_e] \bigcap [\overline{q_s}, \overline{q_e}] \neq \emptyset\}. \end{aligned}$$

The corresponding precision(Ent$_P$), recall(Ent$_R$) and F1(Ent$_F$) are:

$$\frac{|\Gamma|}{|\overline{\Omega}|}, \frac{|\Gamma|}{|\Omega|}, \frac{2\mathrm{Ent}_P\mathrm{Ent}_R}{\mathrm{Ent}_P + \mathrm{Ent}_R}$$

B Dataset Statistic

All details of datasets used in this paper are shown in Table 7.

C Implementation Details

For all experiments, we employ the AdamW optimizer and the default learning rate schedule strategy provided by Transformers library (Wolf et al., 2020).

C.1 Experiments on Grounding

SQUALL We use uncased BERT-base as the encoder. The learning rate is 3×10^{-5}. The training epoch is 50 with a batch size of 16. The dropout rate and the threshold τ are set to 0.3 and 0.2 respectively. The training process lasts 6 hours on a single 16GB Tesla P100 GPU.

SPIDER-L We implement two versions: uncased BERT-base and uncased BERT-large. For both versions, the learning rate is 5×10^{-5} and the training epoch is 50. For BERT-base (BERT-large) version, the batch size and gradient accumulation step are set to 12 (6) and 6 (4). The dropout rate and the threshold τ are set to 0.3 and 0.2 respectively. As for training time, BERT-base (BERT-large) version is trained on a 24GB Tesla P40 and it takes about 16 (48) hours to finish the training process.

WebQSP$_{\text{EL}}$& GraphQ$_{\text{EL}}$ Due to the large amount of entity candidates, we first use the candidate retrieval method proposed in (Sorokin and Gurevych, 2018) to reduce the number of candidates. After that, we still can not feed all candidates along with the question due to the maximum encoding length of BERT. Therefore, we divide the candidates into multiple chunks and feed each chunk (along with the question) into BERT sequentially.

In implementation, we use uncased BERT-base as the encoder. The learning rate is 1×10^{-5} The training epoch is 50 with a batch size of 16. The dropout rate and the threshold τ are set to 0.3 and 0.3 respectively. The training procedure finishes within 10 hours on a single Tesla M40 GPU.

C.2 Experiments on Text-to-SQL

For experiments of the text-to-SQL task, we employ the official code released along with Shi et al. (2020) (on WTQ) and Lei et al. (2020) (on Spider). When coupling ETA with these models, we first produce a one-hot grounding matrix derived by grounding pairs and then feed it into them as described in §4.

WTQ We use uncased BERT-base as the encoder. The training epoch is 50 with a batch size of 8. The learning rate is 1×10^{-5} for the BERT module and 1×10^{-3} for other modules. The dropout rate is set to 0.2. The training process finishes within 16 hours on a single 16GB Tesla P100 GPU.

Meanwhile, we follow the previous work (Shi et al., 2020) to employ 5-fold cross-validation, and

Dataset	Train		Dev		Test	
	#Q	#C	#Q	#C	#Q	#C
SQUALL	9,030	19,185	2,246	4,774	–	–
SPIDER-L	7,000	28,848	1,034	4,360	–	–
WTQ	9,030	–	2,246	–	4,344	–
Spider	7,000	–	1,034	–	2,147	–
WebQSP$_{\mathrm{EL}}$	2,974	3,242	–	–	1,603	1,806
GraphQ$_{\mathrm{EL}}$	2,089	2,253	–	–	2,075	2,229

Table 7: Statistics for all datasets used in our experiments. For SQUALL and WTQ, we only show the size of Split-0, and details of other splits can be found in Table 8. #Q represents the number of questions, #C represents the number of concepts.

Split	Train	Dev
0	9,030	2,246
1	9,032	2,244
2	9,028	2,248
3	8,945	2,331
4	9,069	2,207

Table 8: The size of train set and dev set of five splits on SQUALL and WTQ.

Split	Dev		Test
	Ex.Match	Ex.Acc	Ex.Acc
0	45.10	64.43	53.57
1	47.39	67.01	54.17
2	47.24	65.93	53.61
3	45.99	65.72	53.41
4	52.38	69.73	52.41

Table 9: The experimental results of all splits on WTQ.

experimental results of all five splits on WTQ using ETA + BERT are shown in Table 9.

Spider We implement two versions: uncased BERT-base and uncased BERT-large. For BERT-base (BERT-large), the learning rate is 1.25×10^{-5} (6.25×10^{-6}) for the BERT module and 1×10^{-4} (5×10^{-5}) for other modules. The batch size and gradient accumulation step are set to 10 (6) and 5 (4) for BERT-base (BERT-large) version. The dropout rate is set to 0.3. As for training time, BERT-base (BERT-large) version is trained on a 24GB Tesla P40 and it takes about 36 (56) hours to finish the training process.

D Latent Grounding Visualization

Figure 6 and Figure 7 show the latent grounding visualization corresponding to examples in Table 6.

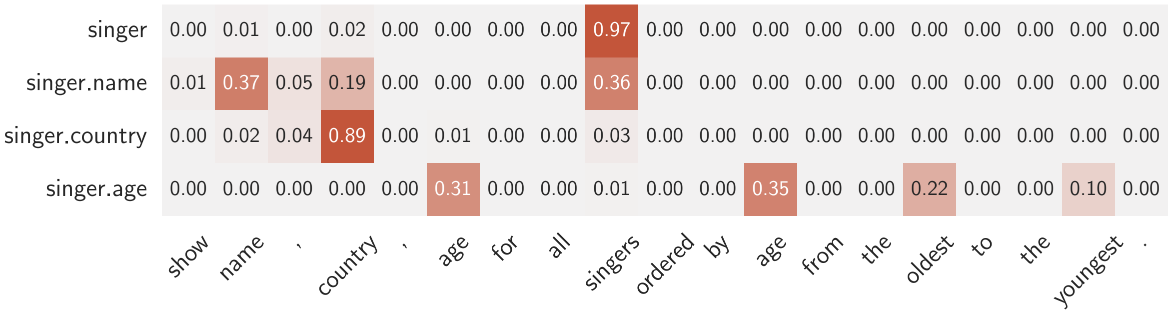

	show	name	,	country	,	age	for	all	singers	ordered	by	age	from	the	oldest	to	the	youngest	.
singer	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
singer.name	0.01	0.37	0.05	0.19	0.00	0.00	0.00	0.00	0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
singer.country	0.00	0.02	0.04	0.89	0.00	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
singer.age	0.00	0.00	0.00	0.00	0.00	0.31	0.00	0.00	0.01	0.00	0.00	0.35	0.00	0.00	0.22	0.00	0.00	0.10	0.00

Figure 6: The latent grounding produced by ETA + BERT$_L$ for the question "*Show name, country, age for all singers ordered by age from the oldest to the youngest.*".

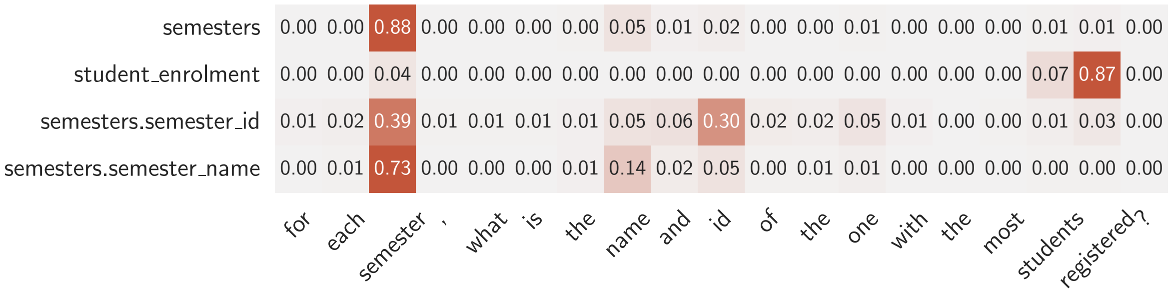

	for	each	semester	,	what	is	the	name	and	id	of	the	one	with	the	most	students	registered	?
semesters	0.00	0.00	0.88	0.00	0.00	0.00	0.00	0.05	0.01	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.01	0.00
student_enrolment	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.87	0.00
semesters.semester_id	0.01	0.02	0.39	0.01	0.01	0.01	0.01	0.05	0.06	0.30	0.02	0.02	0.05	0.01	0.00	0.00	0.01	0.03	0.00
semesters.semester_name	0.00	0.01	0.73	0.00	0.00	0.00	0.01	0.14	0.02	0.05	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00

Figure 7: The latent grounding produced by ETA + BERT$_L$ for the question "*For each semester, what is the name and id of the one with the most students registered?*".

Enhancing Label Correlation Feedback in Multi-Label Text Classification via Multi-Task Learning

Ximing Zhang[1*†], **Qian-Wen Zhang**[2*], **Zhao Yan**[2], **Ruifang Liu**[1], **Yunbo Cao**[2]
[1]Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Tencent Cloud Xiaowei, Beijing 100080, China
ximingzhang@bupt.edu.cn, cowenzhang@tencent.com, zhaoyan@tencent.com,
lrf@bupt.edu.cn, yunbocao@tencent.com

Abstract

In multi-label text classification (MLTC), each given document is associated with a set of correlated labels. To capture label correlations, previous classifier-chain and sequence-to-sequence models transform MLTC to a sequence prediction task. However, they tend to suffer from label order dependency, label combination over-fitting and error propagation problems. To address these problems, we introduce a novel approach with multi-task learning to enhance label correlation feedback. We first utilize a joint embedding (JE) mechanism to obtain the text and label representation simultaneously. In MLTC task, a document-label cross attention (CA) mechanism is adopted to generate a more discriminative document representation. Furthermore, we propose two auxiliary label co-occurrence prediction tasks to enhance label correlation learning: 1) Pairwise Label Co-occurrence Prediction (PLCP), and 2) Conditional Label Co-occurrence Prediction (CLCP). Experimental results on AAPD and RCV1-V2 datasets show that our method outperforms competitive baselines by a large margin. We analyze low-frequency label performance, label dependency, label combination diversity and coverage speed to show the effectiveness of our proposed method on label correlation learning. Our code is available at `https://github.com/EiraZhang/LACO`.

1 Introduction

Multi-label text classification (MLTC) is an important natural language processing task with applications in text categorization, information retrieval, web mining, and many other real-world scenarios (Zhang and Zhou, 2014; Liu et al., 2020). In MLTC, each given document is associated with a set of labels which are often related statistically and semantically. Label correlations should be sufficiently utilized to build multi-label classification models with strong generalization performance (Tsoumakas et al., 2009; Gibaja and Ventura, 2015). In particular, learning the dependencies between labels might be helpful in modeling the low-frequency labels, because real-world classification problems tend to exhibit *long-tail* label distribution, where low-frequency labels are associated with only a few instances and are difficult to learn (Menon et al., 2020).

Previous sequence-to-sequence (Seq2Seq) based methods (Nam et al., 2017; Yang et al., 2018) have been shown to have a powerful ability to capture label correlations with using the current hidden state of the model and the prefix label predictions. However, exposure bias phenomenon (Bengio et al., 2015) may cause the models overfit to the frequent label sequence in training set, thus lead to several problems. First, Seq2Seq-based methods heavily rely on a predefined ordering of labels and perform sensitively to the label order (Vinyals et al.; Yang et al., 2019; Qin et al., 2019). Actually, labels are essentially an order-independent set in the MLTC task. Second, the Seq2Seq-based methods suffer from low generalization ability problem since they tend to overfit the label combinations in the training set and have difficulty to generate the unseen label combination. Third, Seq2Seq-based methods rely on the previous potentially incorrect prediction results. The errors may propagate during the inference stage where true previous target labels are unavailable and are thus replaced by labels generated by the model itself.

To circumvent the potential issues mentioned above, we introduce a multi-task learning based approach that does not rely on Seq2Seq architecture. The approach contains a shared encoder, a MLTC task specific module and a label correla-

---

[*] Equal contribution.

[†] Work done during an internship at Tencent.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1190–1200
August 1–6, 2021.

tion enhancing module. In the shared parameter layers, we introduce a joint embedding (JE) mechanism which takes advantage of a transformer-based encoder to obtain document and label representation jointly. Correlations among labels are learned implicitly through the self-attention mechanism, which is different from previous label embedding methods (Wang et al., 2018; Xiao et al., 2019) that treat labels independently. In MLTC task specific module, we generate the label-specific document representation by the document-label cross attention (CA) mechanism, which retains discriminatory information. The shared encoder and the MLTC task specific module form the basic model called **LACO**, i.e. LAbel COrrelation aware multi-label text classification.

The co-occurrence relationship among labels is one of the important signal that can reflect label correlations explicitly, which can be obtained without additional manual annotation. In label correlation enhancing module, we propose two label co-occurrence prediction tasks, which are jointly trained with the MLTC task. The one is the *Pairwise Label Co-occurrence Prediction* (PLCP) task for capturing second-order label correlations through the two-by-two combinations to distinguish whether they appear together in the set of relevant labels. The other one is the *Conditional Label Co-occurrence Prediction* (CLCP) task for capturing high-order label correlations through a given partial relevant label set to predict the relevance of other unknown labels.

We conduct experiments on AAPD and RCV1-V2 datasets, and show that our method outperforms competitive baselines by a large margin. Comprehensive experimental results are provided to analysis low-frequency label performance, label dependency, label combination diversity and coverage speed, which are essential to measure the ability of label correlation learning. We highlight our contributions as follows:

1. We propose a novel and effective approach for MLTC, which not only sufficiently learns the features of documents and labels through the joint space, but also reinforces correlations through multi-task design without depending on the label order.

2. We propose two feasible tasks (PLCP and CLCP) to enhance the feedback of label correlations, which is beneficial to help induce the multi-label predictive model with strong generalization performance.

3. We compare our approach with competitive baseline models on two multi-label classification datasets and systematically demonstrate the superiority of the proposed models.

2 Related Work

Our work mainly relates to two fields of MLTC task: label correlation learning and document representation learning.

2.1 Label Correlation Learning

For MLTC task, a simple but widely used method is binary relevance (BR) (Boutella et al., 2004), which decomposes the MLC task into multiple independent binary classification problem without considering the correlations between labels.

To capture label correlations, label powerset (LP) (Tsoumakas and Katakis, 2007) take MLTC task as a multi-class classification problem by training a classifier on all unique label combinations. Classifier Chains (CC) based method (Read et al., 2011) exploits the chain rule and predictions from the previous classifiers as input. Seq2Seq architectures are proposed to transform MLTC into a label sequence generation problem by encoding input text sequences and decoding labels sequentially (Nam et al., 2017). However, both CC and Seq2Seq-based methods heavily rely on a predefined ordering of labels and perform sensitively to the label order. To tackle the label order dependency problem, various methods have been explored: by sorting heuristically (Yang et al., 2018), by dynamic programming (Liu and Tsang, 2015), by reinforcement learning (Yang et al., 2019), by multi-task learning (Tsai and Lee, 2020; Zhao et al., 2020). Different from these works, our method learns the label correlations through a non-Seq2Seq-based approach without suffering the above mentioned problems.

More recently, researchers have proposed a variety of label correlation modeling methods for MLTC that are not based on Seq2Seq architecture. Wang et al. (2020) propose a multi-label reasoner mechanism that employs multiple rounds of predictions, and relies on predicting multiple rounds of results to ensemble or determine a proper order, which is computationally expensive. CorNet-BertXML (Xun et al., 2020) utilizes BERT (Devlin et al., 2019) to obtain the joint representation of text and all candidate labels and extra exponential linear units (ELU) at the prediction layer to make use of label correlation knowledge. Different from the above works, we exploit extra label co-occurrence prediction tasks to explicitly model the label correlations in a multi-task framework.

2.2 Document Representation Learning

Text representation plays a significant role in text classification tasks. It is crucial to extract essential hand-crafted features for early models (Joachims, 1998). Deep neural network based MLTC models have achieved great success such as CNN (Kurata et al., 2016; Liu et al., 2017), RNN (Liu et al., 2016), CNN-RNN (Chen et al., 2017; Lai et al., 2015), attention mechanism (Yang et al., 2016; You et al., 2018; Adhikari et al., 2019) and etc. (Devlin et al., 2019) is an important turning point in the development of text classification task and it works by generating contextualized word vectors using Transformer. The reason why deep learning methods have become so popular is their ability to learn sophisticated semantic representations from text, which are much richer than hand-crafted features(Guo et al., 2020). However, these methods tend to ignore the semantics of labels while focusing only on the representation of the document.

Recently, label embedding is considered to improve multi-label text classification tasks. (Liu et al., 2017) is the first DNN-based multi-label embedding method that seeks a deep latent space to jointly embed the instances and labels. LEAM (Wang et al., 2018) applies label embedding in text classification, which obtains each label's embedding by its corresponding text descriptions. LSAN (Xiao et al., 2019) makes use of document content and label text to learn the label-specific document representation with the aid of self-attention and label-attention mechanisms. Our work differs from these works in that the goal of our work is to consider not only the relevance between the document and labels but also the correlations between labels.

3 Methodology

The framework of LACO is shown in Figure 1. The lower layers are shared across all tasks, while the top layers are task-specific. In this section, we first introduce the standard formal definition of MLTC. After that, we present the detailed technical implementation of LACO.

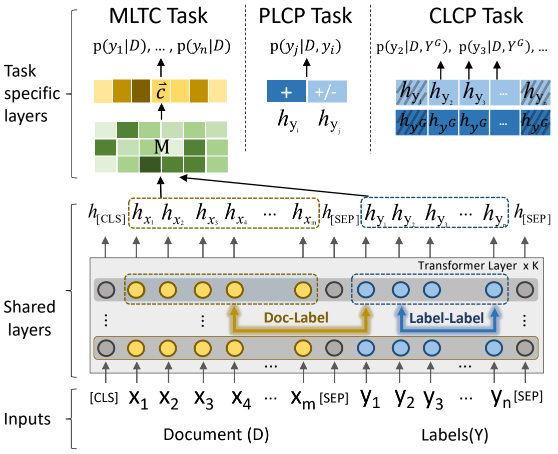

Figure 1: The framework of our proposed approach. Note that the shaded square in the CLCP task is the embedding of given labels, and $+$, $-$ represent related label and unrelated label respectively.

3.1 Problem Formulation

Multi-label task studies the classification problem where each single instance is associated with a set of labels simultaneously. Given a training set $S = \{(D_i, Y_i^+)|1 \leq i \leq N\}$ of multi-label text classification data, D_i is the text sequence and Y_i^+ is its corresponding labels. Specifically, a text sequence D of length m is composed of word tokens $D = \{x_1, x_2, ..., x_m\}$, and $Y = \{y_1, y_2, ..., y_n\}$ denote the label space consisting of n class labels. The aim of MLTC is to learn a predictive function $f : D \rightarrow 2^Y$ to predict the associated label set for the unseen text. For such, the model must optimize a loss function which ensures that the relevant and irrelevant labels of each training text are predicted with minimal misclassification.

3.2 Document-Label Joint Embedding (JE)

Following BERT (Devlin et al., 2019), the first token is always the [CLS] token. The output vector corresponding to the [CLS] token aggregates the features of the whole document and can be used for classification. Different from this habitual operation, we propose a novel input structure to directly use label information in constructing the token-level representations.

As shown in Figure 1, the inputs are packed by a sequence pair (D, Y), we separate the text sequence D and the label sequence Y with a special token [SEP]. Note that the label sequence is to concatenate all label tokens. The shared layers map the inputs into a sequence of embedding vectors, one for each token, called token-level representations. Formally, let $\{[CLS], x_1, ..., x_m, [SEP], y_1, ..., y_n, [SEP]\}$ be the input sequence of the encoder, we obtain the output contextualized token-level representations $\{h_{[CLS]}, h_{x_1}, ..., h_{x_m}, h_{[SEP]}, h_{y_1}, ..., h_{y_n}, h_{[SEP]}\}$. The input structure is designed to guarantee that words and labels are embedded together in the same space. With the joint embedding mechanism, our model could pay more attention to two facets: 1) The correlations between document and labels. Different document have different influences on a specific label, while the same document fragment may affect multiple labels. 2) The correlations among labels. The semantic information of labels is interrelated, and label co-occurrence indicates strong semantic correlations between them.

3.3 Multi-Label Text Classification

In this subsection, we introduce the MLTC task specific module, including *Document-Label Cross Attention* (CA) and *Label Predication*.

3.3.1 Document-Label Cross Attention (CA)

To explicitly model the semantic relationship between each word and label token, we measure the compatibility of label-word pairs via dot product:

$$M = H_D H_Y^T \tag{1}$$

where $H_D = [h_{x_1}, ..., h_{x_m}]$ is the text sequence embedding, $H_Y = [h_{y_1}, ..., h_{y_n}]$ is the label sequence embedding and $M \in \mathcal{R}^{m \times n}$. Considering the semantic information among consecutive words, we further generalize M through non-linearity network. Specifically, for a text fragment of length $2r + 1$ centered at i, the local matrix block $M_{i-r;i+r}$ in M measures the correlation for the label-phrase pairs. To improve the effectiveness of the sparse regularization, we use CNN with ReLU activation in the hidden layers, and perform max-pooling and hyperbolic tangent sequentially in the function Ω:

$$\overrightarrow{c} = \Omega(M_{i-r;i+r}) \cdot H_D \tag{2}$$

Note that the final document representation $\overrightarrow{c}$ is generated by aggregation of word representations H_D, and weighted by the label-specific attention vector $\Omega(\cdot)$.

3.3.2 Label Predication

Once having the discriminative document representation, we build the multi-label text classifier via a fully connected layer that captures more fine-grained features from different regions of the document:

$$\overrightarrow{p} = sigmoid(W_1 \overrightarrow{c}^T + b_1) \tag{3}$$

where $W_1 \in \mathcal{R}^{n \times k}$ and $b_1 \in \mathcal{R}^n$. We use Binary Cross Entropy as the loss function for the multi-label classification problem:

$$\mathcal{L}_{mlc} = -\sum_{i=1}^{n} [q_i \ln p_i + (1 - q_i) \ln(1 - p_i)] \tag{4}$$

where $p_i = P(y_i|D)$ is the probability of y_i predicted by the model, and $q_i \in \{0, 1\}$ is the categorical information of y_i. We train the model by minimizing the cross-entropy error.

3.4 Multi-Task Learning with Label Correlations

In this subsection, we introduce two auxiliary tasks, *Pairwise Label Co-occurrence Prediction* (PLCP) and *Conditional Label Co-occurrence Prediction* (CLCP), to explore the second-order and high-order label relationships, respectively.

3.4.1 PLCP Task

Suppose that each document D contains the corresponding label set Y^+ and the uncorresponding label set Y^-. In order to train the model to understand second-order label relationships, we propose a binarized label-pair prediction task named as PLCP that can be trivially generated from the multi-label classification corpus. The strategy of selecting label pairs for co-occurrence prediction is straightforward. One part is sampled only from Y^+, which is marked as *IsCo-occur*, and the other part is sampled from Y^+ and Y^-, respectively, which is marked as *NotCo-occur*. To construct the manual training dataset, we empirically set the ratio of *IsCo-occur* and *NotCo-occur* to γ. As Figure 1 shows, we concat the embedding of the two labels $[y_i, y_j]$ together as the input features. The additional binary classifier is used to predict whether the state of the two labels is *IsCo-occur* or *NotCo-occur*. The loss function is as followed:

$$\mathcal{L}_{plcp} = -[q_{ij} \ln p_{ij} + (1 - q_{ij}) \ln(1 - p_{ij})] \tag{5}$$

where $p_{ij} = p(y_j|D, y_i)$ denotes the output probability of the the co-occurrance of the label-pair, and q is the ground-truth where $q_{ij} = 1$ means *IsCo-occur* and $q_{ij} = 0$ means *NotCo-occur*.

3.4.2 CLCP Task

To further learn the high-order label relationships, we propose the conditional label co-occurrence prediction (CLCP) task. We first randomly pick s labels from Y^+ to form Y^G, i.e. $Y^G \subseteq Y^+$, and then predict whether the remaining labels of Y are relevant with them. Specifically, we introduce an additional position vector $E_Y = [e_{y_1}, ..., e_{y_n}]$, where $e_{y_i} = 0$ indicates that y_i at that position is the sampled label, i.e. $y_i \in Y^G$, and $e_{y_i} = 1$ indicates $y_i \in Y - Y^G$. The average of the embedding of the zero-position labels h_{y^G} is concatenated to each nonzero-position label embedding as the input features to predict whether each of remaining labels should be co-occurrence when knowing the sampled labels. In Figure 1, $p(y_i|D, Y^G)$ denotes the probability of y_i predicated by the additional sigmoid classifier. The loss for the classification is the sum of binary cross-entropy loss of each nonzero-position:

$$\mathcal{L}_{clcp} = -\sum_{i=1}^{n-s}[q_i \ln p_i + (1 - q_i)\ln(1 - p_i)] \quad (6)$$

where $q_i \in \{0, 1\}$ is the ground-truth to denote whether the label y_i should be co-occurrence with Y^G, and $p_i = p(y_i|D, Y^G)$ is the output probability of each masked label y_i.

3.4.3 Training Objectives

The same inputs are first fed into the shared layers, then each sub-task module takes the contextualized token-level representations generated by joint embedding and produces a probability distribution for its own target labels. The overall loss can be calculated by:

$$\mathcal{L} = \mathcal{L}_{mlc} + \alpha\mathcal{L}_{plcp} + (1 - \alpha)\mathcal{L}_{clcp} \quad (7)$$

where α is a hyperparameter in (0, 1), $\mathcal{L}_{plcp}$ and $\mathcal{L}_{clcp}$ are task-specific Cross-Entropy loss for PLCP task and CLCP task, respectively. [1]

4 Experimental Setup

4.1 Datasets

We validate our proposed model on two multi-label text classification datasets: **Arxiv Academic Paper Dataset (AAPD)** (Yang et al., 2018) collected 55,840 abstracts of papers in the field of computer science, which is organized into 54 related topics. In AAPD dataset, each paper is assigned multiple topics. **Reuters Corpus Volume I (RCV1-V2)** (Lewis et al., 2004) is composed of 804,414 manually categorized newswire stories for research purposes. Each story in the dataset can be assigned multiple topics, and there are 103 topics in total.

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert Y\rvert$	$\overline{\lvert D_i\rvert}$	$\overline{\lvert Y_i^+\rvert}$
AAPD	55,840	54	163.42	2.41
RCV1-V2	804,414	103	123.94	3.24

Table 1: Statistics of datasets. Here, $|\mathcal{D}|$ and $|Y|$ denote the total number of documents and labels. $\overline{|D_i|}$ is the average length of all documents. $\overline{|Y_i^+|}$ means the average number of labels associated with the document.

Tabel 1 shows statistics of datasets. Each dataset is divided into a training set, a validation set, and a test set. We followed the division of these two datasets by Yang et al. (2018).

4.2 Evaluation Metrics

Multi-label classification can be evaluated with a group of metrics, which capture different aspects of the task (Zhang and Zhou, 2014). Following the previous works (Yang et al., 2018; Tsai and Lee, 2020), we adopt hamming loss, Micro/Macro-F1 scores as our main evaluation metrics. Micro/Macro-P and Micro/Macro-R are also reported to assist analysis. A Macro-average will treat all labels equally, whereas a Micro-average will weighted compute each label by its frequency.

4.3 Comparing Algorithms

We adopt a various of methods as baselines, which can be divided into two groups according to whether the label correlations are considered.

The first group of approaches do not consider label correlations. Binary Relevance (BR) (Boutella et al., 2004) amounts to independently training one binary classifier (linear SVM) for each label. CNN (Kim, 2014) utilizes multiple convolution kernels to extract text features and then output the probability distribution over the label space. LEAM (Wang et al., 2018) involves label embedding to obtain a more discriminative text representation in text classification. LSAN (Xiao et al., 2019) learns the label-specific text representation with the help of attention mechanisms. We also implement a BERT (Devlin et al., 2019) classifier which first encodes a document into vector space and then outputs the probability for each label independently.

---

[1] We also implement it with three tasks together. Since the two auxiliary tasks have the similar goal, there is no performance gain.

Algorithm	AAPD dataset			RCV1-V2 dataset		
	HL↓	**Mi- P / R / F1↑**	**Ma- P / R / F1↑**	**HL↓**	**Mi- P / R / F1↑**	**Ma- P / R / F1↑**
BR† (Boutella et al., 2004)	0.0316	64.4 / 64.8 / 64.6	- - -	0.0086	90.4 / 81.6 / 85.8	- - -
CNN† (Kim, 2014)	0.0256	**84.9** / 54.5 / 66.4	- - -	0.0089	92.2 / 79.8 / 85.5	- - -
LEAM (Wang et al., 2018)	0.0261	76.5 / 59.6 / 67.0	52.4 / 40.3 / 45.6	0.0090	87.1/ 84.1 / 85.6	69.5 / 65.8 / 67.6
LSAN (Xiao et al., 2019)	0.0242	77.7 / 64.6 / 70.6	67.6 / 47.2 / 53.5	0.0075	91.3 / 84.1 / 87.5	74.9 / 65.0 / 68.4
BERT (Devlin et al., 2019)	0.0224	78.6 / 68.7 / 73.4	68.7 / 52.1 / 57.2	0.0073	**92.7** / 83.2 / 87.7	77.3 / 61.9 / 66.7
CC† (Read et al., 2011)	0.0306	65.7 / 65.1 / 65.4	- - -	0.0087	88.7 / 82.8 / 85.7	- - -
SGM†♣ (Yang et al., 2018)	0.0251	74.6 / 65.9 / 69.9	- - -	0.0081	88.7 / 85.0 / 86.9	- - -
Seq2Set†♣ (Yang et al., 2019)	0.0247	73.9 / 67.4 / 70.5	- - -	0.0073	90.0 / 85.8 / 87.9	- - -
OCD†♣ (Tsai and Lee, 2020)	-	- - 72.0	- - 58.5	-	- - -	- - -
ML-R† (Wang et al., 2020)	0.0248	72.6 / **71.8** / 72.2	- - -	0.0079	89.0 / 85.2 / 87.1	- - -
Seq2Seq$^{♣}_{T}$ (Nam et al., 2017)	0.0275	69.8 / 68.2 / 69.0	56.2 / 53.7 / 54.0	0.0074	88.5 / **87.4** / 87.9	69.8 / 65.5 / 66.1
SeqTag$_{Bert}$	0.0238	74.3/ 71.5 / 72.9	61.5 / **57.5** / 58.5	0.0073	90.6 / 84.9 / 87.7	73.7 / 66.7 / 68.7
LACO	0.0213	80.2 / 69.6 / 74.5	70.4 / 54.0 / 59.1	0.0072	90.8 / 85.6 / 88.1	75.9 / 66.6 / 69.2
LACO+$plcp$	**0.0212**	79.5 / 70.8 / **74.9**	68.4 / 55.8 / 59.9	**0.0070**	90.8 / 86.2 / 88.4	76.1 / 66.5 / 69.2
LACO+$clcp$	0.0215	78.9 / 70.8 / 74.7	**71.9** / 56.6 / **61.2**	**0.0070**	90.6 / 86.4 / **88.5**	**77.6** / **71.5** / **73.1**

Table 2: Predictive performance of each comparing algorithm on two datasets. Hamming Loss (HL), Micro (Mi-) and Marco (Ma-) average Precision (P), Recall (R), F1-Score (F1) are used as evaluation metrics. The ↓ represents the lower score the better performance, and the ↑ is the opposite. Models with † denote for its results are quoted from previous papers. Models with ♣ are the Seq2Seq-based models.

The second group of methods consider label correlations. Classifier Chains (CC) (Read et al., 2011) transforms the MLTC problem into a chain of binary classification problems. SGM (Yang et al., 2018) proposes the Seq2Seq model with global embedding mechanism to capture label correlations. Seq2Set (Yang et al., 2019) presents deep reinforcement learning to improve the performance of the Seq2Seq model. We also implement a Seq2Seq baseline with 12-layer transformer, named with Seq2Seq$_T$. More recently, OCD (Tsai and Lee, 2020) proposes a framework including one encoder and two decoders for MLTC to alleviate exposure bias. ML-Reasoner (Wang et al., 2020) employs a binary classifier to predict all labels simultaneously and applies a novel iterative reasoning mechanism. Besides, we also provide another strong baseline: SeqTag$_{Bert}$ transforms multi-label classification task into sequential tagging task, which first obtain embeddings of each label (H_Y in Sec 3.3) by our shared encoder and then output a probability for each label sequentially by a BiLSTM-CRF model (Huang et al., 2015).

Results of BR, CNN, CC, SGM, Seq2Set, OCD and ML-R are cited in previous papers and results of other baselines are implemented by us. All algorithms follow the same data division.

4.4 Experimental Setting

We implement our model in Tensorflow and run on NVIDIA Tesla P40. We fine-tune models on the English base-uncased versions of BERT [2]. The batch size is 32, and the maximum total input sequence length is 320. The window size of the additional layer is 10, and we set γ as 0.5. We use Adam (Kingma and Ba, 2015) with learning rate of 5e-5, and train the models by monitoring Micro-F1 score on the validation set and stopping the training if there is no increase in 50,000 consecutive steps.

5 Results and Analysis

In this section, we report the main experimental results of the baseline models and the proposed method on two text datasets. Besides, we analyze the performance on different frequency labels, and further evaluate whether our method effectively learns the label correlations through label-pair confidence distribution learning and label combination prediction. Finally, we give a detailed analysis of the convergence study which demonstrates the generalization ability of our method.

5.1 Experiment Results

We report the experimental results of all comparing algorithms on two datasets in Table 2. The first block includes methods without learning label correlations. The second block is the methods considering label correlations, and the third block is our proposed LACO methods. As shown in Table 2, the LACO-based models outperform all

---

[2] https://github.com/google-research/bert

Model	AAPD HL	AAPD Mi-F1	RCV1-V2 HL	RCV1-V2 Mi-F1
BERT	9.39e-09	3.80e-10	4.95e-04	3.67e-08
SeqTag$_{Bert}$	7.76e-16	1.86e-07	4.95e-04	3.67e-08

Table 3: Statistical analysis results. The P-values of LACO on significant test comparing with the two strong baselines BERT and SeqTag$_{Bert}$.

baselines by a large margin in the main evaluation metrics. The following observations can be made according to the results:

- Our basic model LACO training only by the MLTC task significantly improves previous results on hamming loss and Micro-F1. Specifically, on the AAPD dataset, comparing to Seq2Set which considers modeling the label correlations, our basic model decreases by 13.8% on hamming loss and improves by 5.67% on Micro-F1. Comparing with the label embedding method like LSAN, LACO achieves a reduction of 4.00% hamming loss score and an improvement of 0.69% Micro-F1 score on the RCV1-V2 dataset. Also, BERT is still a strong baseline, which shows that obtaining a high-quality discriminative document representation is important for the MLTC task. Here, we train the LACO with 3 random seeds and calculate the mean and the standard deviation. We perform a significant test with LACO and the two strong baselines BERT and SeqTag$_{Bert}$ in Table 3. Comparing with the two strong baseline models, all of the P-values of LACO are below the threshold ($p < 0.05$), suggesting that the performance is statistically significant. In addition, we implement Friedman test (Demšar, 2006) for hamming loss and Micro-F1 metrics. The Friedman statistics F_F for hamming loss is 7.875 and for Micro-F1 is 6.125, when the corresponding critical value is 2.8179 (# comparing algorithms $k = 12$, # datasets $N = 2$). As a result, the null hypothesis of indistinguishable performance among the compared algorithms is clearly rejected at 0.05 significance level.
- Compared with SGM, Seq2Seq$_T$ does not achieve significantly improvements, but SeqTag$_{Bert}$ shows good performance based on the shared Transformer encoder between document and labels. Notably, the result of SeqTag$_{Bert}$ on Micro-F1 is comparable to BERT, but the result on Macro-F1 is observably higher. The above illustrates that label correlation information is more important for learning low frequency labels.

Model	AAPD HL	AAPD Mi-F	AAPD Ma-F	RCV1-V2 HL	RCV1-V2 Mi-F	RCV1-V2 Ma-F
LACO	0.0213	74.5	59.1	0.0072	88.1	69.2
w/o JE	0.0237	72.6	57.7	0.0077	87.5	68.4
w/o CA	0.0220	73.5	58.4	0.0073	87.8	68.5
w/o JE & CA	0.0224	73.4	57.2	0.0073	87.7	66.7

Table 4: Ablation over the proposed joint embedding (JE) and cross attention (CA) mechanisms using the LACO model on AAPD and RCV1-V2 datasets.

- As for the results of the multi-task learning methods, the two subtasks introduced by our method have a certain degree of improvement on the main metrics of the two datasets. Specifically, we observe that the PLCP task shows better performance and presents the best score of 74.9 on Micro-F1 for AAPD dataset, while the CLCP task presents the best performance on Micro-F1 for RCV1-V2 dataset as 88.5. Furthermore, the proposed multi-task framework shows great improvements than the basic model LACO on Macro-F1, which demonstrates that the performance on low-frequency labels can be greatly improved through our label correlation guided subtasks. There are more detailed analysis in Section 5.3 and 5.5. Notably, the CLCP task performs better on Marco-F1 by considering the high-order correlations. We also implement the experiment using the losses of three tasks together, while the combination of the two subtasks can not further improve the model performance comparing to LACO$_{+plcp}$ or LACO$_{+clcp}$, which we consider is due to the strong relevance between the two tasks.

5.2 Ablation Study

In this section, we will demonstrate the effectiveness of two cores of the proposed LACO model, that is a document-label joint embedding (JE) mechanism, and a document-label cross attention (CA) mechanism. Note that, the setting of **w/o JE & CA** is equivalent to the BERT baseline in Tabel 2, which encode document only and predict the probability for each label based on [CLS]. In the **w/o JE** setting, document embedding is encoded by BERT while each label embedding is a learnable random initialized vector. Its label prediction layer is the same with LACO. In the **w/o CA** setting, document and label embedding are obtained by BERT jointly, and probability for each label is predicted based on [CLS]. Tabel 4 shows that JE and CA are both important to obtain a more discriminative text rep-

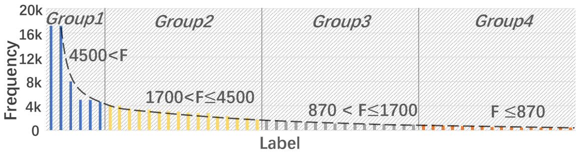

(a) The label distribution of AAPD

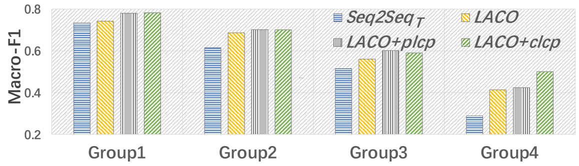

(b)Macro-F1 for the four groups on AAPD

Figure 2: Label classification performance on different frequency distributions. Subfigure(a) shows the label frequency distribution of each label on AAPD training set. Subfigure(b) illustrates the Macro-F1 performance of different methods in the four groups.

resentation. After removing JE and CA mechanism, the performance drops more in the AAPD dataset than RCV1-V2 dataset. We believe that is mainly due to the less of training instance in AAPD, which is more difficult to learn relevant features especially for those low-frequency labels.

5.3 Low-frequency Label Performance

Figure 2(a) illustrates the label frequency distribution on AAPD training set, which is a typical big-head-long-tail distribution. We divide all the labels into four groups according to the frequency, the big-head group (Group1), the high-frequency group (Group2), the middle-frequency group (Group3), and the low-frequency group (Group4). As shown in Figure 2(b), we find the performance of all methods decreases with the label frequency of occurrence. The performance gap between Seq2Seq$_T$ and LACO based methods increases as the frequency decreases, especially in Group 4, LACO$_{+clcp}$ achieves a 74.5% improvement comparing to the Seq2Seq$_T$ model, which demonstrates that the performance on low-frequency labels can be enhanced by the conditional label co-occurrence prediction task.

5.4 Label Correlation Analysis

The co-occurrence relationship between labels is one of the important aspects that can reflect label correlation. In this experiment, we utilize the conditional probability $p(y_b|y_a)$ between label y_a and y_b to represent their dependency quantitatively. Furthermore, we calculate the Conditional Kullback-Leibler Divergence of $p(y_b|y_a)$ to measure the “distance” between model prediction distribution (P^p) and the ground-truth distribution on training/testing dataset (P^g). The score is calculate as:

	AAPD		RCV1-V2	
Model	**train**	**test**	**train**	**test**
Seq2Seq$_T$	**1.27**	1.30	0.08	0.94
SeqTag$_{Bert}$	1.40	1.28	0.09	0.95
LACO	1.40	1.27	0.09	0.94
LACO$_{+plcp}$	1.35	1.28	0.08	**0.76**
LACO$_{+clcp}$	1.32	**1.10**	0.08	0.91

Table 5: $KL(P^g||P^p)$ for different models on AAPD and RCV1-V2 datasets. Note that P^g is the ground truth distribution of datasets and P^p is the model distribution. Smaller scores indicate that two distributions are closer.

$$KL(P^g||P^p) = \sum_{y_a,y_b \in Y} (p^g(y_b|y_a) log \frac{p^g(y_b|y_a)}{p^p(y_b|y_a)}$$
$$p(y_b|y_a) = \#(y_a, y_b)/\#(y_a) \tag{8}$$

where # means the number of the single label or the label combination in the training/testing dataset. The KL-distances on the AAPD and RCV1-V2 datasets are shown in Table 5. On the testing set settings, we can find that LACO has much better fitting ability for the dependency relationship between labels, especially after introducing the co-occurrence relationship prediction task. The Seq2Seq$_T$ model achieves the lowest KL distance with training set on both AAPD and RCV1-V2 but achieve larger scores on the test set. This conclusion further proves that the Seq2Seq-based model is prone to over-fitting label pairs during training. It should be emphasized that this KL distance just quantify how much interdependence between label pairs the model have learned, but it cannot directly measure the prediction accuracy of the model.

5.5 Label Combination Diversity Analysis

Table 6 shows the number of different predicted label combinations (C_{Test}) and subset accuracy (Acc), which is a strict metric that indicates the percentage of samples that have all their labels classified correctly. Seq2Seq$_T$ produces fewer kinds of label combinations on the two datasets. As they tend to “remember” label combinations, the generated label sets are most alike, indicating a poor generalization ability to unseen label combinations. Because Seq2Seq$_T$ is conservative and only generates label combinations it has seen in the training set, it achieves high Acc values, especially on

Model	AAPD		RCV1-V2	
	C_{Test}	Acc	C_{Test}	Acc
Ground Truth	392	1.000	278	1.000
Seq2Seq$_T$	214	0.392	87	0.669
OCD†	302	0.403	-	-
SeqTag$_{Bert}$	289	0.410	187	0.637
LACO	315	0.425	241	0.642
LACO$_{+plcp}$	320	0.439	241	0.644
LACO$_{+clcp}$	321	0.427	239	0.660

Table 6: Statistics on the number of label combinations. C_{Test} is the number of different predicted label combinations. Acc is the subset accuracy on the testing set.

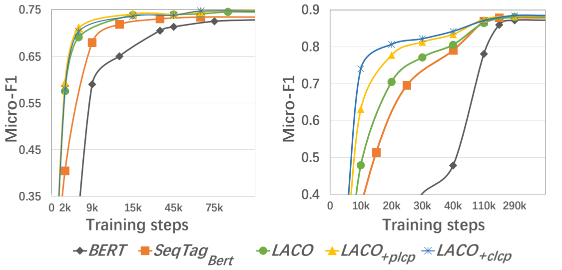

(a) Covergence speed of AAPD (b) Covergence speed of RCV1-V2

Figure 3: The convergence speed of five BERT-based methods. The x-axis refers to the training steps, and the y-axis refers to the Micro-F1 score performance.

RCV1-V2 dataset. For our models, they produce more diverse label combinations while obtaining good Acc since we do not regard multi-label classification as a sequence generation task that uses a decoder to model the relationship between labels. Instead, we learn the correlations among labels on the encoding side, and the scoring between labels does not interfere with each other, which leads to a higher probability of generating label combinations not seen during training than the Seq2Seq-based models.

5.6 Coverage Speed

The convergence speed of five BERT-based models are shown in Figure 3. Our basic model LACO outperforms other BERT-based models in terms of convergence speed, and the proposed multi-task mechanisms are able to enhance LACO to converge much faster. The main reason might be that the feature exchanging through multi-tasks accelerates the model to learn a more robust and common representation.

6 Conclusions and Future Work

In this paper, we propose a new method for MLTC based on document-label joint embedding and correlation aware multi-task learning. Experimental results show that our method outperforms competitive baselines by a large margin. Detailed analyses show the effectiveness of our proposed architecture using semantic connections between document-label and label-label, which helps to obtain a discriminative text representation. Furthermore, the multi-task framework shows strong capability on low-frequency label predicting and label correlation learning.

Considering the Extreme Multi-label Text Classification that contains an extremely large label set, LACO could be further exploited through scheduled label sampling, hierarchical label embedding strategy, and so on. We hope that further research could get clues from our work.

Acknowledgements

We would like to thank the ACL reviewers for their valuable comments and Keqing He, Haoyan Liu, Zizhen Wang, Chenyang Liao and Rui Pan for their generous help and discussion.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv:1904.08398*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1171–1179.

Matthew R Boutella, Jiebo Luob, Xipeng Shena, and Christopher M Browna. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37:1757–1771.

Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 international joint conference on neural networks (IJCNN)*, pages 2377–2383. IEEE.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Eva Gibaja and Sebastián Ventura. 2015. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, 47(3):1–38.

Biyang Guo, Songqiao Han, Xiao Han, Hailiang Huang, and Ting Lu. 2020. Label confusion learning to enhance text classification models. *arXiv:2012.04987*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv:1508.01991*.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015,San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label cooccurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multilabel text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879.

Weiwei Liu, Xiaobo Shen, Haobo Wang, and Ivor W Tsang. 2020. The emerging trends of multi-label learning. *arXiv:2011.11197*.

Weiwei Liu and Ivor W Tsang. 2015. On the optimality of classifier chain for multi-label classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 712–720.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2020. Long-tail learning via logit adjustment. *arXiv:2007.07314*.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multilabel classification. In *Advances in neural information processing systems*, pages 5413–5423.

Kechen Qin, Cheng Li, Virgil Pavlu, and Javed Aslam. 2019. Adapting rnn sequence prediction model to multi-label set prediction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3181–3190.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.

Che-Ping Tsai and Hung-Yi Lee. 2020. Order-free learning alleviating exposure bias in multi-label classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6038–6045.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331.

Ran Wang, Robert Ridley, Weiguang Qu, Xinyu Dai, et al. 2020. A novel reasoning mechanism for multilabel text classification. *Information Processing & Management*, 58(2):102441.

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. 2019. Label-specific document representation for multi-label text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 466–475.

Guangxu Xun, Kishlay Jha, Jianhui Sun, and Aidong Zhang. 2020. Correlation networks for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1074–1082.

Pengcheng Yang, Fuli Luo, Shuming Ma, Junyang Lin, and Xu Sun. 2019. A deep reinforced sequence-to-set model for multi-label classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5252–5258.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: sequence generation model for multi-label classification. *In Proceedings of the 27th International Conference on Computational Linguistics*, page 3915–3926.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2018. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *arXiv:1811.01727*.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Wei Zhao, Hui Gao, Shuhui Chen, and Nan Wang. 2020. Generative multi-task learning for text classification. *IEEE Access*, 8:86380–86387.

Fusing Context Into Knowledge Graph for Commonsense Question Answering

Yichong Xu[*], **Chenguang Zhu**[*], **Ruochen Xu, Yang Liu, Michael Zeng, Xuedong Huang**
Microsoft Cognitive Services Research Group
{yicxu,chezhu,ruox,yaliu10,nzeng,xdh}@microsoft.com

Abstract

Commonsense question answering (QA) requires a model to grasp commonsense and factual knowledge to answer questions about world events. Many prior methods couple language modeling with knowledge graphs (KG). However, although a KG contains rich structural information, it lacks the context to provide a more precise understanding of the concepts. This creates a gap when fusing knowledge graphs into language modeling, especially when there is insufficient labeled data. Thus, we propose to employ external entity descriptions to provide contextual information for knowledge understanding. We retrieve descriptions of related concepts from Wiktionary and feed them as additional input to pre-trained language models. The resulting model achieves state-of-the-art result in the CommonsenseQA dataset and the best result among non-generative models in OpenBookQA.

1 Introduction

One critical aspect of human intelligence is the ability to reason over everyday matters based on observation and knowledge. This capability is usually shared by most people as a foundation for communication and interaction with the world. Therefore, commonsense reasoning has emerged as an important task in natural language understanding, with various datasets and models proposed in this area (Ma et al., 2019; Talmor et al., 2018; Wang et al., 2020; Lv et al., 2020).

While massive pre-trained models (Devlin et al., 2018; Liu et al., 2019) are effective in language understanding, they lack modules to explicitly handle knowledge and commonsense. Also, structured data like knowledge graph is much more efficient in representing commonsense compared with unstructured text. Therefore, there have been multiple methods coupling language models with various forms of knowledge graphs (KG) for commonsense reasoning, including knowledge bases (Sap et al., 2019; Yu et al., 2020b), relational paths (Lin et al., 2019), graph relation network (Feng et al., 2020) and heterogeneous graph (Lv et al., 2020). These methods combine the merits of language modeling and structural knowledge information and improve the performance of commonsense reasoning and question answering.

However, there is still a non-negligible gap between the performance of these models and humans. One reason is that, although a KG can encode topological information between the concepts, it lacks rich context information. For instance, for a graph node for the entity "Mona Lisa", the graph depicts its relations to multiple other entities. But given this neighborhood information, it is still hard to infer that it is a painting. On the other hand, we can retrieve the precise definition of "Mona Lisa" from external sources, e.g. the definition of Mona Lisa in Wiktionary is "*A painting by Leonardo da Vinci, widely considered as the most famous painting in history*". To represent structured data that can be seamlessly integrated into language models, we need to provide a panoramic view of each concept in the knowledge graph, including its neighboring concepts, relations to them, and a definitive description of it.

Thus, we propose the DEKCOR model, i.e. DEscriptive Knowledge for COmmonsense question answeRing, to tackle multiple choice commonsense questions. Given a question and a choice, we first extract the contained concepts. Then, we extract the edge between the question concept and the choice concept in ConceptNet (Speer et al., 2017). If such an edge does not exist, we compute a relevance score for each knowledge triple (subject, relation, object) containing the choice concept, and select the one with the highest score. Next, we

---

[*] Equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1201–1207
August 1–6, 2021.

retrieve the definition of these concepts from Wiktionary via multiple criteria of text matching. Finally, we feed the question, choice, selected triple and definitions into the language model ALBERT (Lan et al., 2019) to produce a score indicating how likely this choice is the correct answer.

We evaluate our model on CommonsenseQA (Talmor et al., 2018) and OpenBookQA (Mihaylov et al., 2018). On CommonsenseQA, it outperforms the previous state-of-the-art result by 1.2% (single model) and 3.8% (ensemble model) on the test set. On OpenBookQA, our model outperforms all baselines other than two large-scale models based on T5 (Raffel et al., 2019). We further conduct ablation studies to demonstrate the effectiveness of fusing context into the knowledge graph.

2 Related work

Several different approaches have been investigated for leveraging external knowledge sources to answer commonsense questions. Min et al. (2019) addresses open-domain QA by retrieving from a passage graph, where vertices are passages and edges represent relationships derived from external knowledge bases and co-occurrence. Sap et al. (2019) introduces the ATOMIC graph with 877k textual descriptions of inferential knowledge (e.g. if-then relation) to answer causal questions. Lin et al. (2019) projects questions and choices to the knowledge-based symbolic space as a schema graph. It then utilizes path-based LSTM to give scores. Feng et al. (2020) adopts the multi-hop graph relation network (MHGRN) to perform reasoning unifying path-based methods and graph neural networks. Lv et al. (2020) proposes to extract evidence from both structured knowledge base such as ConceptNet and Wikipedia text and conduct graph-based representation and inference for commonsense reasoning. Wang et al. (2020) employs GPT-2 to generate paths between concepts in a knowledge graph, which can dynamically provide multi-hop relations between any pair of concepts.

Several studies have utilized knowledge descriptions for different tasks. Yu et al. (2020a) uses description text from Wikipedia for knowledge-text co-pretraining. Xie et al. (2016) encodes the semantics of entity descriptions in knowledge graphs to improve the performance on knowledge graph completion and entity classification. Chen et al. (2018) co-trains the knowledge graph embeddings and entity description representation for cross-lingual entity alignment. Concurrent with our work, Chen et al. (2020) also insert knowledge descriptions into commonsense question answering. Compared with our work, the proposed method in Chen et al. (2020) is much more complex, e.g. involving training additional rankers on retrieved text, while our result outperforms Chen et al. on CommonsenseQA.

3 Method

3.1 Knowledge Retrieval

Problem formulation. In this paper, we focus on the following QA task: given a commonsense question q, select the correct answer from several choices $c_1, ..., c_n$. In most cases, the question does not contain any mentions of the answer. Therefore, external knowledge sources can be used to provide additional information. We adopt ConceptNet (Speer et al., 2017) as our knowledge graph $G = (V, E)$, which contains over 8 million entities as nodes and over 21 million relations as edges. In the following, we use triple to refer to two neighboring nodes and the edge connecting them, i.e. $(u \in V, p = (u, v) \in E, v \in V)$, with u being the subject, p the relation, and v the object.

Suppose the question mentions an entity $e_q \in V$ and the choice contains an entity $e_c \in V$[1]. We then employ the KCR method (Lin, 2020) to select relation triples. If there is a direct edge r from e_q to e_c in G, we choose this triple (e_q, r, e_c). Otherwise, we retrieve all the N triples containing e_c. Each triple j is assigned a score s_j which is the product of its triple weight w_j provided by ConceptNet and relation type weight t_{r_j}:

$$s_j = w_j \cdot t_{r_j} = w_j \cdot \frac{N}{N_{r_j}} \tag{1}$$

Here, r_j is the relation type of the triple j, and N_{r_j} is the number of triples among the retrieved triples that have the relation type r_j. In other words, this process favors rarer relation types. Finally, the triple with the highest weight is chosen.

3.2 Contextual information

The retrieved entities and relations from the knowledge graph are described by their surface form. Without additional context, it is hard for the language model to understand its exact meaning, especially for proper nouns.

---

[1] CommonsenseQA provides the question/choice entity. For OpenBookQA, we choose from the extracted entities that are most frequent in retrieved facts. See Appendix for details.

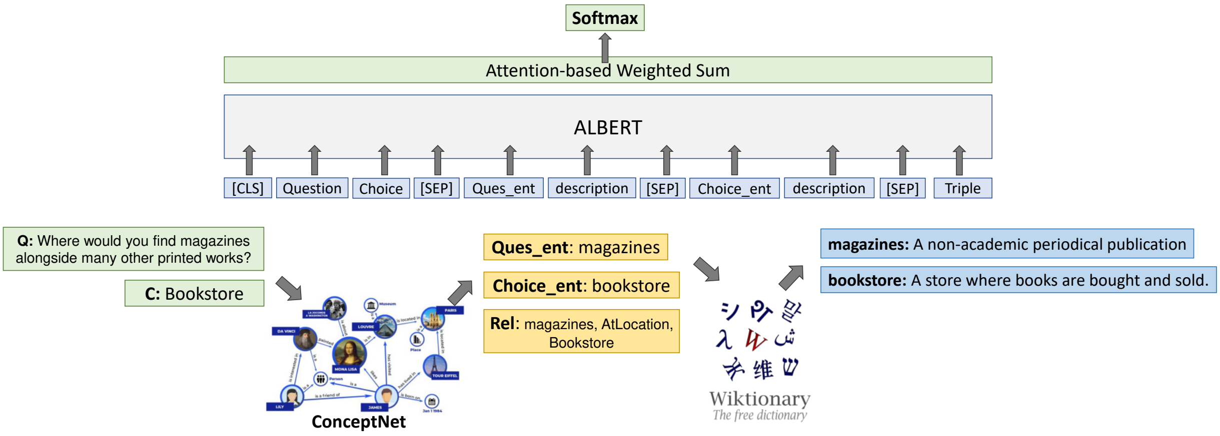

Figure 1: In our model, the input to ALBERT includes the question, choice, entity names, description text and triple. An attention-based weighted sum and a softmax layer process the output from ALBERT to produce the prediction.

Therefore, we leverage large-scale online dictionaries to provide definitions as context. We use a dump of Wiktionary[2] which includes definitions of 999,614 concepts. For every concept, we choose its first definition entry in Wiktionary as the description. For every question/choice concept, we find its closest match in Wiktionary by using the following forms in order: i) original form; ii) lemma form by Spacy (Honnibal and Montani, 2017); iii) base word (last word). For example, the concept "taking notes" does not appear in its original form in Wiktionary, but its lemma form "take notes" is in Wiktionary and we get its description text: "*To make a record of what one hears or observes for future reference*". In this way, we find descriptions of all entities in our experiments. The descriptions of the question and choice concept are denoted by d_q and d_c, respectively.

Finally, we feed the question, choice, descriptions and triple (from Section 3.1) into the ALBERT model (Lan et al., 2019) in the following format: [CLS] q c [SEP] e_q: d_q [SEP] e_c: d_c [SEP] triple.

3.3 Reasoning

On top of ALBERT, we leverage an attention-based weighted sum and a softmax layer to generate the relevance score for the question-choice pair. In detail, suppose the output representations of ALBERT is $(\boldsymbol{x}_0, ..., \boldsymbol{x}_m)$, where $\boldsymbol{x}_i \in R^d$. We compute a weighted sum of these embeddings based on attention:

Table 1: Statistics of CommonsenseQA (CSQA) and OpenBookQA (OBQA).

Dataset	Train	Dev	Test	Choices
CSQA	9,741	1,221	1,140	5
OBQA	4,957	500	500	4

$$q_i = \boldsymbol{u}^T \boldsymbol{x}_i \tag{2}$$

$$\alpha_i = \text{softmax}(q_i) \tag{3}$$

$$\boldsymbol{v} = \sum_{i=0}^{m} \alpha_i \boldsymbol{x}_i, \tag{4}$$

where $\boldsymbol{u}$ is a parameter vector. The relevance score between the question and the choice is then $s = \text{softmax}(\boldsymbol{v}^T \boldsymbol{b})$, where $\boldsymbol{b} \in R^d$ is a parameter vector and the softmax is computed over all choices for the cross-entropy loss function.

The architecture of our model DEKCOR and the construction of input is shown in Fig. 1.

4 Experiments

4.1 Datasets and baselines

We evaluate our model on two benchmark datasets of multiple-choice questions for commonsense question answering: CommonsenseQA (Talmor et al., 2018) and OpenBookQA (Mihaylov et al., 2018). CommonsenseQA creates questions from ConceptNet entities and relations; OpenBookQA probes elementary science knowledge from a book of 1,326 facts. The statistics of the datasets is provided in Table 1. For OpenBookQA, we follow prior approaches (Wang et al., 2020) to append top

---

[2] `https://www.wiktionary.org/`

Table 2: Accuracy on the test set of CommonsenseQA.

Methods	Single	Ensemble
BERT+OMCS	62.5	-
RoBERTa	72.1	72.5
RoBERTa+HyKAS	73.2	-
XLNet+DREAM	-	73.3
RoBERTa+KE	73.3	-
RoBERTa+KEDGN	-	74.4
XLNet+GraphReason	75.3	-
ALBERT	-	76.5
RoBERTa+MHGRN	75.4	76.5
ALBERT+PG-Full	75.6	78.2
T5	78.1	-
ALBERT+KRD	78.4	-
UnifiedQA	79.1	-
ALBERT+KCR	79.5	-
DEKCOR (ours)	**80.7**	**83.3**

Table 3: Accuracy on the test set of OpenBookQA.

Methods	Accuracy
BERT + Careful Selection	72.0
AristoRoBERTa	77.8
ALBERT + KB	81.0
ALBERT + PG-Full	81.8
TTTTT (T5-3B)	83.2
UnifiedQA (T5-11B)	**87.2**
DEKCOR (ours)	82.4

5 retrieved facts provided by Aristo team (Clark et al., 2019) to the input. We also pre-train our OpenBookQA model on CommonsenseQA's training set as we find it helps to boost the performance.

We compare our models with state-of-the-art baselines, which all employ pre-trained models including RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), ALBERT (Lan et al., 2019) and T5 (Raffel et al., 2019) and some adopt additional modules to process knowledge information. A detailed description of the baselines is in the Appendix.

4.2 Results

CommonsenseQA. Table 2 shows the accuracy on the test set of CommonsenseQA. For a fair comparison, we categorize the results into single models and ensemble models. Our ensemble model consists of 7 single models with different initialization random seeds, and its output is the majority of choices selected by these single models. More implementation details are shown in the Appendix.

Table 4: Ablation results on the dev sets of CommonsenseQA and OpenBookQA.

Methods	CSQA	OBQA
DEKCOR	84.7	82.2
Triple Only	82.0	80.0
Description Only	80.3	81.8
No Context	78.9	80.0

Our proposed DEKCOR outperforms the previous state-of-the-art result by 1.2% (single model) and 3.8% (ensemble model). This demonstrates the effectiveness of the usage of knowledge description to provide context.

Furthermore, we notice two trends based on the results. First, the underlying pre-trained language model is important in commonsense QA quality. In general, we observe this order of accuracy among these language models: BERT<RoBERTa<XLNet<ALBERT<T5. Second, the additional knowledge module is critical to provide external information for reasoning. For example, RoBERTa+KEDGN outperforms the vanilla RoBERTa by 1.9%, and our model outperforms the vanilla ALBERT model by 6.8% in accuracy.

OpenBookQA. Table 3 shows the test set accuracy on OpenBookQA. All results are from single models. Note that the two best-performing models, i.e. UnifiedQA (Khashabi et al., 2020) and TTTTT (Raffel et al., 2019), are based on the T5 generation model, with 11B and 3B parameters respectively. Thus, they are computationally very expensive. Except these T5-based systems, DEKCOR achieves the best accuracy among all baselines.

Ablation study. Table 4 shows that the usage of concept descriptions from Wiktionary and triple from ConceptNet can help improve the accuracy of DEKCOR on the dev set of CommonsenseQA by 2.7% and 4.4% respectively. We observe similar results on OpenBookQA. This demonstrates that additional context information can help with fusing knowledge graph into language modeling for commonsense question answering.

Case Study. Table 5 shows two examples from CommonsenseQA and OBQA respectively. In the first example, without additional description the model would not know relevant information about bats, like they are insectivorous, leading to the wrong answer "eating bugs". With the description, the model knows that bats eat bugs, so it chooses "laying eggs" as the answer. Similarly, for the sec-

CommonsenseQA Question:
Bats have many quirks, with the exception of ___ .
Question entity description:
bat: Any of the flying mammals of the order Chiroptera, usually small and nocturnal, insectivorous or frugivorous.

Model w/o description chooses: eating bugs
Model w/ description chooses: laying eggs

OBQA Question:
Alligators ___ .
Question entity description:
alligator: Either of two species of large amphibious reptile, ..., which have sharp teeth and very strong jaws...

Model w/o description chooses: eat gar
Model w/ description chooses: are warm-blooded

Table 5: Examples from CommonsenseQA and OBQA dataset showing the effectiveness of entity descriptions.

ond question, the "sharp teeth and very strong jaws" in the description hint that alligators are likely carnivorous, and reptiles are likely cold-blooded. The entity description leads to the correct answer of "eat gar".

5 Conclusions

In this paper, we propose to fuse context information into knowledge graphs for commonsense question answering. As a knowledge graph often lacks descriptions for the contained entities and relations, we leverage Wiktionary to provide definitive text for each entity as additional input to the pre-trained language model ALBERT. The resulting DEKCOR model achieves state-of-the-art results on the benchmark datasets CommonsenseQA and OpenBookQA. Ablation studies demonstrate the effectiveness of the proposed usage of knowledge description and knowledge triple information in commonsense question answering.

Acknowledgements

We thank the anonymous reviewers for their valuable comments.

References

Pratyay Banerjee, Kuntal Kumar Pal, A. Mitra, and Chitta Baral. 2019. Careful selection of knowledge to solve open book question answering. In *ACL*.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. *arXiv preprint arXiv:1806.06478*.

Qianglong Chen, Feng Ji, Haiqing Chen, and Yin Zhang. 2020. Improving commonsense question answering by graph-based iterative retrieval over multiple knowledge sources. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2583–2594, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, et al. 2019. From'f'to'a'on the ny regents science exams: An overview of the aristo project. *arXiv preprint arXiv:1909.01958*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. *arXiv preprint arXiv:2005.00646*.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1).

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*.

Jession Lin. 2020. Knowledge chosen by relations. `https://github.com/jessionlin/csqa/blob/master/Model_details.md`.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering.

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. *arXiv preprint arXiv:1910.14087*.

Todor Mihaylov, Peter Clark, Tushar Khot, and A. Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Peifeng Wang, Nanyun Peng, Pedro A. Szekely, and X. Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *EMNLP*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. pages 5753–5763.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2020a. Jaket: Joint pre-training of knowledge graph and language understanding. *arXiv preprint arXiv:2010.00796*.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020b. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

A Implementation Details

Identification of e_q and e_c. CommonsenseQA specifies the question entity in each question and each answer choice is also an entity in ConceptNet. We use them as e_q and e_c. For OpenBookQA, we identify all ConceptNet entities in the question and answer text and count their number of occurrences in the retrieved text. For a triple (e_q, r, e_c), we define its weight as $n_{e_q} + n_{e_c}$, where n_e is the number of occurrences in retrieved text. The edge with the largest weight is picked. If no edge is found between question and answer entities, we use the answer entity with the most occurrences to find triples. For Wiktionary descriptions, we find descriptions for e_q and e_c with the most occurrences as well.

Using ConceptNet. Since ConceptNet contains a lot of weak relations, we only use the following relations for our triples: CausesDesire, HasProperty, CapableOf, PartOf, AtLocation, Desires, HasPrerequisite, HasSubevent, Antonym, Causes.

Optimization. We use the AdamW (Loshchilov and Hutter, 2017) optimizer with a learning rate of 2e-5. The batch size is 8. We limit the maximum length of the input sequence to 192 tokens. The model is trained for 10 epochs. We use the Huggingface (Wolf et al., 2019) implementation for the ALBERT model.

B Baseline Methods

GraphReason (Lv et al., 2020) retrieves knowledge from both structured knowledge base and plain text.

PG-FULL (Wang et al., 2020) fine-tunes GPT-2 on ConceptNet to generate knowledgeable paths between knowledge graph concepts.

UnifiedQA (Khashabi et al., 2020) pre-trains T5 on a variety of QA datasets for general QA tasks.

MHGRN (Feng et al., 2020) adopts the multi-hop graph relation network to perform reasoning.

HyKAS (Ma et al., 2019) employs an option comparison network to consume ConceptNet triples.

ALBERT+KRD retrieves commonsense knowledge from Open Mind Common Sense and then uses a self-attention module to compute a weighted sum of these triple representations.

BERT + Selection (Banerjee et al., 2019) improves the result on OpenBookQA via abductive information retrieval , information gain based re-ranking, passage selection and weighted scoring.

ALBERT+KB also improves retrieval results on OpenBookQA by token-based and embedding-based retrieval. TTTTT (Raffel et al., 2019) fine-tunes the T5 language generation model on OpenBookQA.

Unsupervised Energy-based Adversarial Domain Adaptation for Cross-domain Text Classification

Han Zou[*] Microsoft hanzou@microsoft.com
Jianfei Yang[*] Nanyang Technological University yang0478@ntu.edu.sg
Xiaojian Wu[†] Microsoft xiaojian.academy@gmail.com

Abstract

Transferring knowledge from a label-rich domain (source domain) to a label-scarce domain (target domain) for pervasive cross-domain Text Classification (TC) is a non-trivial task. To overcome this issue, we propose EADA, a novel unsupervised energy-based adversarial domain adaptation framework. First, a deep pre-trained language model (e.g. RoBERTa) is leveraged as a shared feature extractor that maps the text sequences from both source and target domains to a feature space. Since the source features maintain good feature discriminability because of the full supervised training, we design a method that encourages target features towards the source ones via adversarial learning. An autoencoder is designed as an energy function that focuses on reconstructing source feature embeddings, while the feature extractor aims to generate source-like target feature embeddings to deceive the autoencoder. In this manner, the target feature embeddings become domain-invariant and inherit great discriminability. Extensive experiments on multi-domain sentiment classification (Amazon review dataset) and Yes/No question-answering classification (BoolQ and MARCO dataset) are conducted. The experimental results validate that EADA largely alleviates the domain discrepancy while maintaining excellent discriminability and achieves state-of-the-art cross-domain TC performance.

1 Introduction

With the booming development of Natural Language Processing (NLP) in recent years, text classification (TC) is playing a vital role in a myriad of services in our daily lives, such as online recommendations, email spam detection, sentiment classification and social media analysis. Large pre-trained language models, e.g. BERT (Devlin et al., 2019), XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019b), achieve outstanding results on challenging NLP benchmarks, i.e. GLUE (Wang et al., 2018), RACE (Lai et al., 2017), and SQuAD (Rajpurkar et al., 2016). These models enable numerous downstream NLP tasks with compelling performance, including TC, where the model is further fine-tuned with annotated data.

TC tasks are usually domain dependent in real-world. Thus, the performance of these powerful deep models is still fluctuated and even degraded when directly implementing them in a unseen domain (target domain), where the task topic or the data distributions are different from the domain during training (source domain). Although their performance can be improved via fine-tuning with full supervision in the target domain, a significant amount of labeled target data is required. Collecting high-quality data is usually difficult and expensive in many real-world domains. Furthermore, the annotating process is extremely time-consuming and labor-intensive. To overcome these issues, unsupervised domain adaptation (UDA), which aims to transfer the knowledge from a label-rich domain (source domain) to a label-scarce or unlabeled domain (target domain) is proposed (Li et al., 2017; Chen et al., 2018; Guo et al., 2018; Zhang et al., 2019).

The intuitive objective of UDA is to align the marginal distribution of features across source and target domains. In general, UDA methods can be classified into two categories. One line of research focuses on reducing the discrepancy by minimizing statistical measurements, e.g. maximum mean discrepancy (Tzeng et al., 2014a). Another category leverages adversarial learning to alleviate the domain shift. Motivated by Generative Adversarial Network (GAN) (Goodfellow et al., 2014), adversarial domain adaptation (ADA) introduces a binary domain discriminator to identify the domain

---

[*] Equal Contribution.
[†] Work done while at Microsoft.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1208–1218
August 1–6, 2021.

label of the data, while an encoder learns to fool the discriminator. ADA has achieved encouraging results on nontrivial DA problems across various applications, such as image classification (Vu et al., 2019; Yang et al., 2020b), human activity recognition (Zou et al., 2019; Yang et al., 2018; Zou et al., 2018), Internet of Things (Yang et al., 2020a), and also text classification (Li et al., 2017; Zhang et al., 2019). For instance, AMN (Li et al., 2017) trains a sentiment classifier and a domain discriminator to reduce the domain discrepancy. ADAN (Chen et al., 2018) exploits adversarial learning for cross-lingual sentiment classification. HAGAN (Zhang et al., 2019) integrates the hierarchical attention mechanism with ADA to obtain features that are sentiment distinguishable but domain indistinguishable.

Although these ADA methods achieve good results in certain cross-domain TC tasks, one major issue is the unstable prediction performance in the target domain (Xie et al., 2018; Saito et al., 2018). After the adversarial training achieves convergence, the conventional binary domain discriminator cannot distinguish the domain label of the feature representations, which means these representations obtain good transferability. However, there is no constraint on the discriminability in the target domain. The model can generate trivial but useless target feature representations as long as they can fool the domain discriminator. Thus, this uncertainty in adversarial training deteriorates the discriminability of the target feature representations and ignores the decision boundary learned in the source domain, which leads to unstable and even poor prediction performance in the target domain (Chen et al., 2019a; Cui et al., 2020). Some works aim to adjust the decision boundary of the label classifier (Saito et al., 2018; Shu et al., 2018) or align additional semantic information (Xie et al., 2018) to overcome this issue during adversarial training. However, these additional learning steps either require a sophisticated hyper-parameter tuning process or increase the computational overhead, that limits the generalization capability of the ADA methods for NLP tasks. Therefore, a simple yet efficient solution is urgently desired.

In this paper, we propose EADA, an energy-based adversarial domain adaptation framework that tackles the uncertainty issue during adversarial learning and dedicates for text classification tasks. EADA consists of three modules, a shared feature extractor, a label predictor, and an autoencoder. We employ a deep pre-trained language model (RoBERTa) as a shared feature extractor that maps the text sequences from both source domain and target domain into a latent feature space. With the labeled source data, the feature extractor and the label predictor are fine-tuned under full supervision. Since the source feature representations generated from the feature extractor contain superb discriminability, the innovative goal of EADA is to fix these source features by adding constrains in the objective and only force the target feature distribution to align the source feature distribution through adversarial training so that the target features could remain discriminative, and the label predictor could also perform well in the target domain. Since autoencoder is acknowledged as an energy function that learns to map the observed sample to the low-energy space (LeCun et al., 2006), we design an autoencoder that leverages this property to fix the source features by associating lower energies to it while pushing the target domain to the low-energy space by minimizing the margin loss of the autoencoder. Meanwhile, it can also cluster similar data to form a high-density manifold, which helps to preserve more semantic information. We train the autoencoder to reconstruct the source features and train the feature extractor to generate source-like target features to deceive the autoencoder via a minimax with a margin loss. In summary, we make the following contributions:

- To address the problem of conventional binary domain discriminator that deteriorates the discriminability of the target feature presentation, we propose a novel autoencoder module, which forces the target feature representations to simulate source feature representations such that good discriminability can be inherited.

- As an energy function, the autoencoder maps features from both domains to the low-energy space, which motivates the feature clusters to be tight in an unsupervised manner. It improves the label classification accuracy in the target domain.

- Extensive experiments on public cross-domain TC benchmark datasets, including multi-domain sentiment classification (Amazon review dataset) and cross-domain

Yes/No question-answering (QA) classification (BoolQ and MARCO dataset), are conducted. The experimental results demonstrate that EADA alleviates the uncertainty during adversarial training and enhances the feature discriminability in the target domain. This enables EADA to outperform existing methods and achieve new state-of-the-art ADA results for cross-domain TC tasks without requiring any labeled data in the target domain.

The rest of the paper is organized as follows. Section 2 summarizes the existing domain adaptation methods for TC tasks. The limitation of existing ADA Methods is elaborated in Section 3. Section 4 presents the framework architecture of EADA. In Section 5, we present the experimental results and performance evaluation. We conclude our work in Section 6.

2 Related Work

Domain Adaptation aims to tackle the domain shift issue when the data distribution in the source domain and target domain are different (Ben-David et al., 2010). Unsupervised domain adaptation (UDA) aims to learn a model that is able achieve good classification accuracy without any annotation in the target domain (Tzeng et al., 2017; Zhao et al., 2019). Certain statistical measurements, such as maximum mean discrepancy (MMD) (Tzeng et al., 2014b; Ma et al., 2019), are leveraged to quantify the distribution differences.

Inspired by the recent success of Generative Adversarial Network (GAN) (Goodfellow et al., 2014) for data generation, researchers have proposed adversarial domain adaptation (ADA), that constructs an adversarial loss to accommodate the domain shift. It consists of an encoder and a domain discriminator. The generator aims to fool the discriminator to make the target domain samples look like the source domain ones, while the discriminator tries to identify the domain labels (source or target). ADDA (Tzeng et al., 2017) learns a discriminative representation using the labels in the source domain and then a separate encoder that maps the target data to the same space using an asymmetric mapping learned through a standard GAN loss without weights sharing. CoGAN (Liu and Tuzel, 2016) trains 2 GANs to synthesize both source and target images and achieves a domain invariant feature space by tying the high-level layer parameters of the 2 GAN to solve the domain transfer problem.

ADA has been adopted for cross-domain NLP tasks as well (Peng et al., 2018; Li et al., 2017; Shah et al., 2018; Chen and Cardie, 2018; Cai and Wan, 2019; Wang et al., 2019). AMN (Li et al., 2017) is an end-to-end adversarial memory network for cross-domain sentiment classification, which is the pioneering work for ADA in NLP. An adversarial deep averaging network is proposed in (Chen et al., 2018) for cross-lingual sentiment classification. A dedicated ADA framework for machine reading comprehension is proposed in (Wang et al., 2019). (Chen and Cardie, 2018) designed an ADA model that learns domain invariant representation across multiple domains for text classification. Target domain-specific information is being exploited in (Peng et al., 2018) to further improve the DA performance, while labeled data in the target domain is required.

Large deep pre-trained language models pioneered by BERT (Devlin et al., 2019), have been employed as feature encoders to embed text sequences into a latent feature space. Then, the encoder is further fine-tuned with the discriminator via adversarial learning using the labeled source data and unlabeled target data (Lee et al., 2019; Ma et al., 2019). For instance, BERT and ADA were adopted in (Lee et al., 2019) for domain-agnostic question-answering. A similar framework that integrates BERT and MMD is proposed in (Ma et al., 2019) for cross-domain sentiment classification. However, all these approaches leverage the binary domain discriminator which has failed to consider the discriminative features during feature learning. This leads to severe performance degradation since the decision boundary of the label predictor trained with source data is no longer valid in the target domain due to the domain shift.

3 Limitation of Existing ADA Methods

In this section, we analyze the learning process of conventional ADA methods and reveal their limitations. In common UDA setup, N_s labeled examples from a source domain $\mathcal{D}_S = \{\mathbf{x}_i^s, y_i^s\}$ and N_t unlabeled examples from a target domain $\mathcal{D}_T = \{x_i^t\}$ are available. The distributions of $\mathcal{D}_S$ and $\mathcal{D}_T$ are different due to the domain discrepancy. UDA aims to build up a model that provides good class prediction in both source and target domain. Discriminability of the feature representation is the clustering capacity in the feature manifold, that controls the easiness of class category separation

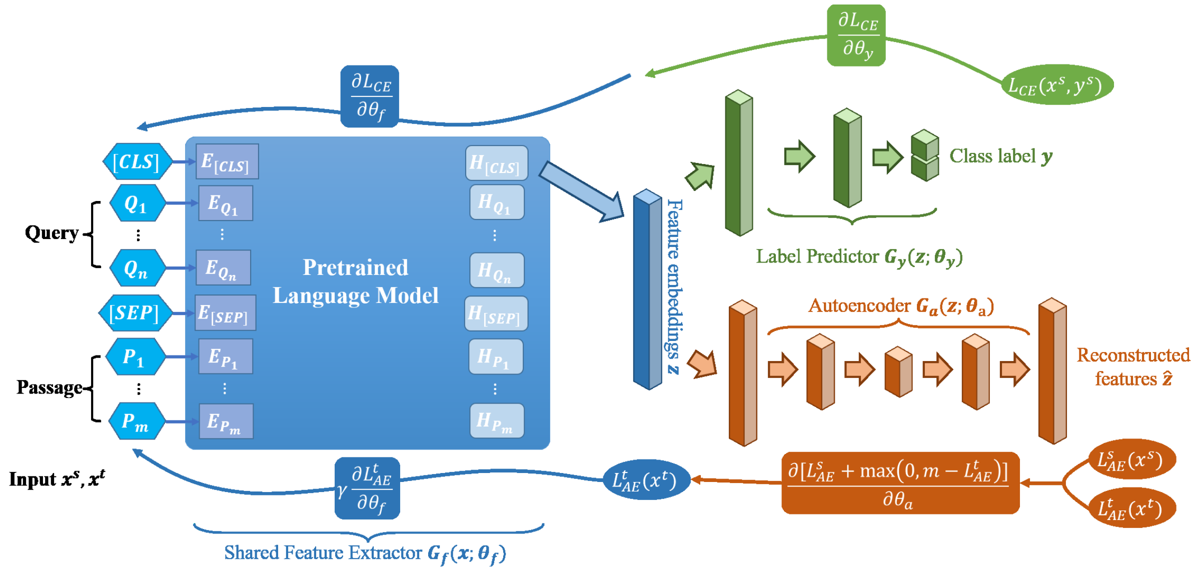

Figure 1: EADA constitutes: a pre-trained language model as shared feature extractor G_f, a label predictor G_y and an autoencoder G_a. In addition to the full supervised learning of G_f and G_y with the labeled source data, the autoencoder G_a serves as a domain classifier to learn reconstructing the source feature representations and push the target feature representations away. The feature extractor G_f aims to generate source-like target feature representations to deceive the autoencoder. This objective is realized by forcing the target feature representations towards the those from the source domain in the feature space via adversarial learning.

(Chen et al., 2019b). Excellent discriminability can be achieved for the source features due to the full supervision learning in the source domain. The objective of UDA is to transfer and ensure the model maintains this discriminability in the target domain.

ADA method, as one category of UDA, which is pioneered by Domain-Adversarial Training of Neural Networks (DANN) (Ganin et al., 2016) and Adversarial Memory Network (AMN) (Li et al., 2017) have shown promising performance in numerous NLP tasks in recent years (Chen and Cardie, 2018; Shah et al., 2018; Cai and Wan, 2019; Wang et al., 2019). It usually consists of a shared feature extractor $\mathbf{f} = G_f(x)$, a label predictor $\mathbf{y} = G_y(x)$ and a domain discriminator $\mathbf{d} = G_d(x)$. In addition to the standard full supervision learning process in the source domain, a minimax game is designed between $\mathbf{f}$ and $\mathbf{d}$. The domain discriminator $\mathbf{d}$ aims to distinguish the domain label between source and target, meanwhile the feature extractor $\mathbf{f}$ is trained to deceive $\mathbf{d}$. This adversarial training process can be formulated as

$$\min_{G_f, G_y} \mathcal{L}_y(\mathbf{X}_s, Y_s) - \gamma \mathcal{L}_f(\mathbf{X}_s, \mathbf{X}_t), \quad (1)$$

$$\min_{G_d} \mathcal{L}_d(\mathbf{X}_s, \mathbf{X}_t), \quad (2)$$

where $\mathcal{L}_y$ is the cross-entropy classification loss. In this manner, the model can learn domain-invariant features and transfer them across domains when the Nash Equilibrium is achieved (Zhao et al., 2017). The hyper-parameter γ controls the significance of adversarial training that improves transferability. As shown in Eq(1), the training process of feature extractor $\mathbf{f}$ of conventional ADA methods aims to achieve two tasks: (1) learn source representations with good discriminability; (2) train representations that are indistinguishable to the domain discriminator $\mathbf{d}$. Since both source domain and target domain data are involved in the adversarial feature learning as presented in the second term of Eq(1), the objective is equivalent to move two domains closer in the feature space to deceive $\mathbf{d}$. However, this process does not impose any constraint on the discriminability in the target domain. The feature extractor $\mathbf{f}$ can generate trivial but useless target representations as long as they can fool the discriminator $\mathbf{d}$. Therefore, these ADA methods cannot guarantee that the good decision boundary learned via full supervision in the source domain can still separate the categorical clusters in the target domain (Chen et al., 2019b; Liu et al., 2019a). This degradation of discriminability in the target domain is the major reason that hinders the performance of existing ADA methods.

4 Energy-based Adversarial Domain Adaptation

It is not a trivial task to maintain the source manifolds during adversarial training. Our solution is to decouple the adversarial training process of source and target feature representations. To be specific, we fix the source representation in the feature space and only encourage the target representations align to the source representations. Therefore, the superb discriminability learned in the source domain can be preserved and a label predictor that performs well in both source and target domain can be obtained.

To achieve this goal, we propose Energy-based Adversarial Domain Adaptation (EADA), which innovatively utilizes an autoencoder structure as a domain discriminator during adversarial training. Figure 1 demonstrates the model structure of EADA. It consists of three modules, a pre-trained language model as a shared feature extractor G_f parameterized by θ_f to embed input sample to feature embedding z. After that, a label predictor G_y parameterized by θ_y, which consists of several fully connected layers, further maps the feature embedding z to the predicted label $\hat{y}$. Another module is an autoencoder G_a parameterized by θ_a, that reconstructs a feature embedding z to $\hat{z}$. The detailed functionality of each module is elaborated as follows.

4.1 Shared Feature Extractor

Large pre-trained language models (e.g. BERT (Devlin et al., 2019), XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019b)) have achieved a series of state-of-the-art results on NLP benchmarks. These powerful pre-trained language models are built up on bidirectional transformer architecture, and pre-trained on large corpora with a masked language model, that enable various downstream NLP tasks, including text classification.

In this work, we employ RoBERTa as the shared feature extractor G_f (highlighted in blue in Figure 1) that embeds both labeled text data $(\mathbf{X}_s)$ from the source domain, as well as the unlabeled text data $(\mathbf{X}_t)$ from the target domain into a latent feature space. To be specific, as a QnA classification problem, the input for the G_f are sequence pairs <query **Q**, passage **P**> as depicted in Figure 1 in the format of [CLS] <s> Q </s> <s> P </s>, where [CLS] is a dummy token for classification and <s> </s> are separator tokens. We leverage the roberta.base architecture (12-layer, 768-hidden, 12-heads, 125M parameters) (Liu et al., 2019b) as the shared feature extractor G_f. Since our objective is text classification, the last hidden representation of the [CLS] token, $H_{[CLS]} \in \mathbf{R}^{768\times 1}$ (feature embedding z) serves as the output of G_f. These embeddings $\mathbf{z}$ are utilized by both classifier G_y and autoencoder G_a.

4.2 Class Label Predictor

The class label predictor G_y consists of several fully connected layers that map the feature embedding z to the predicted label $\hat{y}$. Since the source domain is label-rich by default, we assume that n labeled samples $\mathcal{D}_S = \{\mathbf{x}_i^s, y_i^s\}$ are available from the source domain for finetuning of the shared feature extractor (language model) G_f (blue part in Figure 1) and the label predictor G_y (green part in Figure 1). The good classification accuracy of G_y is achieved by minimizing the cross-entropy loss via back-propagation under full supervision:

$$\begin{aligned}&\min_{G_f,G_y} \mathcal{L}_{CE}(\mathbf{X}_s, Y_s) = \\ &- \mathbb{E}_{(\mathbf{x}_s,y_s)\sim(\mathbf{X}_s,Y_s)} \sum_{n=1}^{N_s} [\mathbb{I}_{[l=y_s]} \log G_y(G_f(\mathbf{x}_s))].\end{aligned} \tag{3}$$

4.3 Autoencoder as Domain Discriminator

After obtaining the source feature representation with good discriminability, the next task is to learn transferable features with k unlabeled samples from a target domain $\mathcal{D}_T = \{x_i^t\}$. To ensure both transferability and discriminability of the feature representation, we design an autoencoder G_a with a margin Mean Squared Error (MSE) loss to replace the conventional binary domain discriminator. The MSE loss of the autoencoder is defined as:

$$\mathcal{L}_{AE}(\mathbf{x}_i) = ||G_a(G_f(\mathbf{x};\theta_f);\theta_a) - \mathbf{x}_i||_2^2, \tag{4}$$

where $||\cdot||_2^2$ denotes the squared L_2-norm. Since the source embeddings z_s always contain superb discriminability due to full supervision during the training of the classifier, z_s should be fixed to preserve the good decision boundary, while the target embeddings z_t should be encouraged to align with the distribution of z_s. To achieve this goal, the autoencoder G_a is designed to be able to only reconstruct features in the source domain but not features in the target domain. Namely, when two domains distribute similarly, the autoencoder will incur the same reconstruction loss in both domains. The training process of the autoencoder is formulated as:

$$\min_{G_a} \mathcal{L}_{AE}(\mathbf{X_s}) + \max(0, m - \mathcal{L}_{AE}(\mathbf{X_t})), \quad (5)$$

where m is the margin between the representations from the source domain and the target domain. The autoencoder G_a can be considered as an energy function that associates lower energies to the observed samples in a binary classification problem (LeCun et al., 2006). With the inspiration of Energy-based GAN which theoretically proves that using an energy function in GAN, the true distribution can be simulated by the generator at *Nash Equilibrium* (Zhao et al., 2017). In EADA, the autoencoder module G_a provides similar functionality that associates low energies to the source features (focuses on reconstructing source embeddings z_s). As presented in Eq(5), the training goal of the autoencoder is to have $\mathcal{L}_{AE}(\mathbf{X_s}) = 0$ and $\mathcal{L}_{AE}(\mathbf{X_t}) = m$. It behaves proportionally similar to a binary domain discriminator. But G_a includes more domain information and can transfer it during adversarial training.

4.4 The Learning Framework

The adversarial training objective of three modules forms a minimax game, that is defined by:

$$\begin{aligned} &\min_{G_f, G_y} \mathcal{L}_{CE}(\mathbf{X}_s, Y_s) + \gamma \mathcal{L}_{AE}(\mathbf{X_t}), \\ &\min_{G_a} \mathcal{L}_{AE}(\mathbf{X_s}) + \max(0, m - \mathcal{L}_{AE}(\mathbf{X_t})), \end{aligned} \quad (6)$$

where γ is a hyper-parameter to control the effectiveness of G_a. The shared feature extractor G_f maps both labeled source data $\mathbf{X_s}$ and unlabeled target data $\mathbf{X_t}$ to a latent feature space. Both G_f and the label predictor G_y are trained with full supervision using the labeled data in the source domain. Another key role of the feature extractor G_f is to deceive the autoencoder G_a by generating source-like features for unlabeled target samples. Therefore, we only incorporate the $\mathcal{L}_{AE}(\mathbf{X_t})$ term into the training of G_f. The adversarial training of G_f is formulated by:

$$\min_{G_f} \mathcal{L}_{AE}(\mathbf{X_t}). \quad (7)$$

In the minimax game, the autoencoder G_a aims to maximize the domain divergence by pushing two domains away from a margin m, while the objective of the feature extractor G_f is to minimize the domain divergence by deceiving the autoencoder. When the model achieves convergence, the target feature representations inherit excellent discriminability from the source domain so that the generalization capability of the label predictor G_y is improved and performs well not only in the source domain but also in the target domain.

5 Experiments

We evaluate the domain adaptation performance of EADA on two public real-world cross-domain text classification benchmarks: 1) sentiment classification (Amazon reviews dataset); 2) Natural QA Yes/No classification (BoolQ ⇔ MS Marco), and compared it with state-of-the-art baselines.

5.1 Evaluation on Sentiment Classification

Amazon reviews dataset (Pan et al., 2010) is the standard and well-known benchmark for sentiment classification domain adaptation. It contains reviews on four domains: Books (B), DVDs (D), Electronics (E), and Kitchen (K). Each domain contains 1000 positive reviews (higher than 3 stars) and 1000 negative reviews (3 stars or lower). 12 cross-domain sentiment classification tasks: D→B, E→B, K→B, K→E, D→E, B→E, B→D, K→D, E→D, B→K, D→K, E→K, where the letter before the arrow represents the source domain and the letter after the arrow indicates the target domain by following (Li et al., 2017). For each pair of domain adaptation, 800 labeled positive (Pos) and 800 labeled negative (Neg) reviews from the source domain (src), together with 1600 unlabeled reviews from the target domain (tgt) are randomly selected for training. The rest of 200 positive and 200 negative reviews from the target domain are used for testing.

We configured the feature extractor module G_f as RoBERTa.base for single sequences task since each review is one sequence passage. The input is tokenized as [CLS] <s> Review </s>. The maximum input sequence length is set to 256 tokens. The autoencoder module G_a consists of 5 fully connected layers (768-384-96-384-768). The entire EADA framework is implemented in PyTorch. The Adam optimizer with the constant learning rate $\mu = 1e^{-5}$ with a batch size of 24 was adopted and we used 5-fold cross-validation to tune the hyperparameter $m = 4$ and $\gamma = 1e^{-2}$ during the training.

Table 1: Cross-domain classification accuracy of different methods on Amazon review dataset.

Tasks	Source-only RoBERTa	AMN	ADAN	HAGAN	HAGAN-C	MoE	ADA RoBERTa	EADA
D→B	85.55	81.52	81.70	81.22	81.69	81.70	86.80	**88.10**
E→B	81.95	77.80	78.55	79.05	79.23	79.65	82.10	**85.25**
K→B	72.65	79.37	79.25	78.52	78.99	80.94	77.60	**81.20**
B→D	81.95	81.32	82.30	82.07	82.38	84.60	82.47	**86.05**
E→D	82.50	77.51	79.70	81.00	80.65	84.20	83.70	**85.35**
K→D	80.00	80.03	80.45	80.83	80.91	**81.22**	80.58	80.35
B→E	85.95	80.07	77.60	79.87	80.12	84.66	86.00	**89.35**
D→E	82.75	80.00	79.70	80.57	80.99	84.10	82.99	**87.15**
K→E	82.45	81.97	86.85	85.94	85.23	**85.81**	84.32	85.11
B→K	86.80	81.00	76.10	81.25	82.00	86.32	87.00	**89.65**
D→K	85.75	83.88	77.35	81.73	81.50	86.33	86.55	**89.20**
E→K	86.95	87.10	83.95	84.30	84.99	85.40	87.10	**90.50**
Average	*82.94*	*80.96*	*80.29*	*81.36*	*81.56*	*83.74*	83.93	***86.44***

The performance of EADA is compared with the following state-of-the-art baselines: *Source-only RoBERTa* directly applies the finetuned RoBERTa model from source domain in target domain; *AMN* (Li et al., 2017) adopts attention mechanism and ADA for cross-domain sentiment classification; *ADAN*(Chen et al., 2018) employs ADA to transfer the knowledge of resource-rich source language to low-resource language; *HAGAN* and *HAGAN-C* (Zhang et al., 2019) incorporate hierarchical attention into ADA for cross-domain sentiment classification; *MoE* (Guo et al., 2018) utilizes a mixture of experts from different source domains to further enhance the DA performance in target domain; *ADA RoBERTa* uses RoBERTa as a shared feature extractor and a conventional binary domain discriminator as the domain classifier for ADA.

Table 1 presents the mean accuracy with 5 runs of each method on the 12 DA tasks. One observation is that the accuracy of source-only RoBERTa (82.94%) is even worse than MoE (83.74%). This indicates that the problem of DA cannot be solved by just solely using large pre-trained language models. It can be observed that EADA provides 86.44% classification accuracy on average, which outperforms all the baselines. It achieves the best DA performance in 10 out of the 12 tasks. Although ADA-RoBERTa and EADA both adopted RoBERTa as the feature extractor, the accuracy of EADA is still 2.5% higher than ADA-RoBERTa, which validates the advantage of the proposed energy-based ADA method. Moreover, the variance of EADA is the smallest among all the methods, which indicates its performance is more stable in general. By learning a source-like representation for the target feature embeddings, EADA successfully performs cross-domain sentiment classification without any annotated data in target domain.

5.2 Evaluation on Yes/No QA classification

We also validated the performance of EADA on cross-domain naturally occurring yes/no questions between BoolQ dataset (Clark et al., 2019) and Marco dataset (Nguyen et al., 2016). Each example is a triplet of (query, passage, answer). Thus, feature extractor module G_f (RoBERTa) is configured for sequence pairs task, which means the input is tokenized in the format as [CLS] <s> query </s> <s> passage </s>. BoolQ dataset contains 5874 Yes and 3553 No samples for training and 2033 Yes and 1237 No samples for evaluation from Wikipedia. Samples in Marco dataset are web snippets from Bing Search. There are 17339 Yes and 10550 No samples for training and 2033 Yes and 1237 No samples for testing. The data distributions at both domain level and categorical level are imbalanced, which is a common situation in many real-world applications. The number of training samples in BoolQ is only 33.8% of those in Marco, indicating that the data are imbalanced across domains. Moreover, the data categorical distribution is imbalanced because the number of No-samples is at least 39.1% less than the number of Yes-samples in both domains. Since the samples from the two domains are collected from different sources, there is a huge domain shift between the two datasets.

The domain adaptation performance is reported in Table 2. The 2^{nd} column shows the accuracies

Table 2: Cross-domain adaptation for Yes/No QA classification between BoolQ and Marco datasets.

Tasks	Source only	AMN	HAGAN	ADA-RoBERTa	MoE	EADA	Target full supervision
BoolQ → Marco	67.63	72.95	73.11	73.86	74.01	**78.38**	82.51
Marco → BoolQ	69.30	74.34	74.73	75.11	75.23	**79.51**	84.31

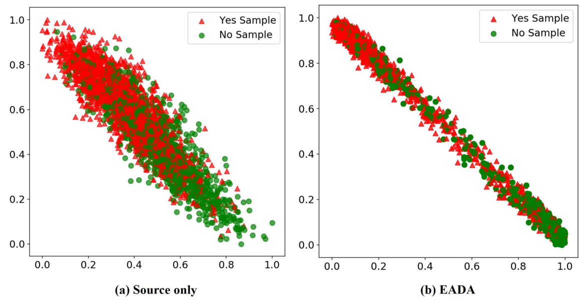

Figure 2: The t-SNE visualization of features embedded using distinct feature extractor in target domain. (Marco → BoolQ).

when the non-adapted source feature extractors and classifiers are directly applied in the target domain, which serves as the lower-baseline. The last column reports the accuracies when the feature extractors and classifiers are trained with full-supervision that all the target training data are labeled (as the upper-baseline). As shown in Table 2, the source-only classifiers can only provide 68% accuracy, which verifies that the domain shift hurts the classification accuracy even when a powerful deep language model is adopted. On the other hand, it can be easily observed from Table 2 that EADA enhances the accuracy in both adaptation directions by at least 15% compared to the lower-baseline in an unsupervised manner, and outperforms all the state-of-the-art baselines. It elevates the performance closer to the target full supervision as well.

We leverage t-Distributed Stochastic Neighbor Embedding (t-SNE) to map the embedded feature representations through different feature extractors to a 2-D space for better visualization and analysis. Figure 2(a) and Figure 2(b) depict the embedded features using the non-adapted source feature extractor and the EADA's feature extractor learned, respectively (Yes sample - red, No sample - Green). If we directly apply the non-adapted source feature extractor in the target domain, as shown in Figure 2(a), a large amount samples with different categorical labels overlap with each other, which leads to corresponding huge misclassification as presented in the 2^{nd} column of Table 2. After employing EADA, the common confusions are further separated in the latent feature space and two clusters are formulated as depicted in Figure 2(b). These observations further validate that the feature embeddings constructed via EADA are not only domain-invariant but also preserve excellent discriminability in both the source domain and the target domain.

We also conducted a sensitivity study of the two hyperparameters: m and γ in Eq(6). We evaluated the impact of margin m with different values from 0 to 10 in both experiments. EADA's accuracy increases while m is increasing. The reason is that the degree of transferability is limited when m is small. The performance becomes stable when $m \geqslant 4$ in both experiments. In general, the objective of γ is to control the weight of the adversarial loss during feature learning of ADA as shown in Eq(1). γ in EADA aims to control the weight of autoencoder reconstruction loss for target domain samples during feature learning as presented in Eq(6)). We evaluated the impact of γ with different values (0-1) for EADA and ADA-RoBERTa. As γ increases, the accuracy of EADA increases and becomes stable when $\gamma \geqslant 0.01$. The accuracy of ADA-RoBERTa fluctuates when γ is increased and decreased when $\gamma \geqslant 0.5$. Thus, EADA provides a more stable training procedure compared to conventional ADA methods, which makes it easier for generalization. We recommend using m=4 and γ=0.01 as the default setup for other tasks.

6 Conclusion

In this paper, we proposed EADA, a novel unsupervised energy-based adversarial domain adaptation method for cross-domain text classification tasks. First, a deep pre-trained language model is leveraged as a shared feature extractor to map the text sequences from both source and target domains to a feature space. The feature extractor and a label predictor are trained with labeled source data. Since the source feature representations are obtained under full supervision, they preserve great feature discriminability. To ensure that the label predictor also provides good label prediction in the target domain, the target feature representations should be encouraged to align with the source during adversarial training. Thus, we designed an autoencoder that focuses on reconstructing the source feature representations, while the feature extractor aims to generate source-like target feature embeddings to fool the autoencoder. Extensive experiments on public cross-domain TC benchmarks are conducted and demonstrate that EADA not only alleviates the domain discrepancy but also enhances the feature discriminability in the target domain, which leads to compelling cross-domain TC performance without requiring any labeled data in the target domain.

References

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Yitao Cai and Xiaojun Wan. 2019. Multi-domain sentiment classification based on domain-aware embedding and attention. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4904–4910. AAAI Press.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1226–1240.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. 2019a. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1081–1090.

Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. 2019b. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1081–1090, Long Beach, California, USA. PMLR.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. 2020. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3941–3950.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Seanie Lee, Donggyu Kim, and Jangwon Park. 2019. Domain-agnostic question-answering with adversarial training. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 196–202.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, pages 2237–2243.

Hong Liu, Mingsheng Long, Jianmin Wang, and Michael Jordan. 2019a. Transferable adversarial training: A general approach to adapting deep classifiers. In *International Conference on Machine Learning*, pages 4013–4022.

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 469–477.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with bert-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuan-Jing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2505–2513.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732.

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1056–1063.

Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. 2018. A dirt-t approach to unsupervised domain adaptation. In *Proc. 6th International Conference on Learning Representations*.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014a. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014b. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez. 2019. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. 2019. Adversarial domain adaptation for machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2510–2520.

Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. 2018. Learning semantic representations for unsupervised domain adaptation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5423–5432, Stockholmsmässan, Stockholm Sweden. PMLR.

Jianfei Yang, Han Zou, Shuxin Cao, Zhenghua Chen, and Lihua Xie. 2020a. Mobileda: Toward edge-domain adaptation. *IEEE Internet of Things Journal*, 7(8):6909–6918.

Jianfei Yang, Han Zou, Hao Jiang, and Lihua Xie. 2018. Carefi: Sedentary behavior monitoring system via commodity wifi infrastructures. *IEEE Transactions on Vehicular Technology*, 67(8):7620–7629.

Jianfei Yang, Han Zou, Yuxun Zhou, Zhaoyang Zeng, and Lihua Xie. 2020b. Mind the discriminability: Asymmetric adversarial domain adaptation. In *European Conference on Computer Vision*, pages 589–606. Springer.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yuebing Zhang, Duoqian Miao, and Jiaqi Wang. 2019. Hierarchical attention generative adversarial networks for cross-domain sentiment classification. *arXiv preprint arXiv:1903.11334*.

Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J Gordon. 2019. On learning invariant representation for domain adaptation. *arXiv preprint arXiv:1901.09453*.

Junbo Zhao, Michael Mathieu, and Yann LeCun. 2017. Energy-based generative adversarial network. In *Proc. 5th International Conference on Learning Representations*.

Han Zou, Jianfei Yang, Yuxun Zhou, Lihua Xie, and Costas J Spanos. 2018. Robust wifi-enabled device-free gesture recognition via unsupervised adversarial domain adaptation. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE.

Han Zou, Yuxun Zhou, Jianfei Yang, Huihan Liu, Hari Prasanna Das, and Costas J Spanos. 2019. Consensus adversarial domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5997–6004.

Survival text regression for time-to-event prediction in conversations

Christine De Kock
University of Cambridge
Department of Computer Science
`cd700@cam.ac.uk`

Andreas Vlachos
University of Cambridge
Department of Computer Science
`av308@cam.ac.uk`

Abstract

Time-to-event prediction tasks are common in conversation modelling, for applications such as predicting the length of a conversation or when a user will stop contributing to a platform. Despite the fact that it is natural to frame such predictions as regression tasks, recent work has modelled them as classification tasks, determining whether the time-to-event is greater than a pre-determined cut-off point. While this allows for the application of classification models which are well studied in NLP, it imposes a formulation that is contrived, as well as less informative. In this paper, we explore how to handle time-to-event forecasting in conversations as regression tasks. We focus on a family of regression techniques known as survival regression, which are commonly used in the context of healthcare and reliability engineering. We adapt these models to time-to-event prediction in conversations, using linguistic markers as features. On three datasets, we demonstrate that they outperform commonly considered text regression methods and comparable classification models.

1 Introduction

The task of predicting when an event will occur in a conversation frequently arises in NLP research. For instance, Backstrom et al. (2013) and Zhang et al. (2018b) predict when a conversation thread will terminate. Danescu-Niculescu-Mizil et al. (2013) define the task of forecasting when users will cease to interact on a social network based on their language use. Although these questions naturally lend themselves to regression, this presents some difficulties: datasets may be highly skewed towards shorter durations (Zhang et al., 2018b) and samples with a longer duration can contribute inordinately to error terms during training. Furthermore, classical regression models do not explicitly consider the effect of time as distinct from other features.

The abovementioned studies instead frame the time-to-event prediction as a classification task, predicting whether the current state will continue for a set number of additional timesteps. For instance, Backstrom et al. (2013) predict whether the number of responses in a thread will exceed 8, after seeing 5 utterances. This presents obvious limitations; such a setup would assign the same error for mistakenly classifying conversations of respectively 9 and 30 utterances as "short". Additionally, its predictions are less informative: predicting that a conversation will be more than 8 utterances long is less telling than predicting whether it will be 9 or 30.

In this paper, we propose that *survival regression* is a more appropriate modelling framework for predicting when an event will occur in a conversation. Survival regression aims to predict the probability of an event of interest at different points in time, taking into account features of a subject as seen up to the prediction time. We apply survival models to two tasks: predicting conversation length, and predicting when conversations will get derail into personal attack. We report results for the conversation length prediction task on the datasets from Danescu-Niculescu-Mizil et al. (2012) and De Kock and Vlachos (2021), and evaluate the personal attack prediction task on the dataset of Zhang et al. (2018a). Our results illustrate that linear survival models outperform their linear regression counterparts, with an improvement in MAE of 1.22 utterances on the dataset of De Kock and Vlachos (2021). Further performance gains are made using neural network-based survival models. An analysis of the coefficients of our linear models indicates that survival models infer similar relationships as previous work on conversation length prediction, but that their predictions are more accurate than conventional regression and classification models due to their explicit accounting for the effect of time. On the personal attack prediction task,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1219–1229
August 1–6, 2021.

the best survival model provides a 13% increase in ranking accuracy over linear regression models.

The remainder of this paper is structured as follows. In Section 2 we provide a description of key survival analysis concepts. In Section 3, we describe how we apply these concepts to conversations. Results are reported in Section 4.

2 Survival regression

Survival analysis is concerned with modelling time-to-event prediction, which often represents transitions between states throughout a subject's lifetime. In the general case, exactly one event of interest occurs per lifetime, after which the subject is permanently in the alternate state, often referred to as "death" in literature. In this section, we review some key concepts of survival analysis that are relevant to our work, however, we refer the interested reader to the exposition by Rodriquez (2007).

2.1 Definitions

Let T be a non-negative random variable representing the waiting time until the occurrence of an event. Given the cumulative distribution function $F(t)$ of the event time T, the *survival function* is defined as the probability of surviving beyond a certain point in time t:

$$\begin{aligned} S(t) &= P(T > t) \\ &= 1 - F(t). \end{aligned} \tag{1}$$

Per illustration, we consider the task of predicting conversation length using the dataset of disagreements of De Kock and Vlachos (2021). The event of interest is the end of a conversation, with time measured in utterances. We can estimate the survival function using Kaplan-Meier estimation (Jager et al., 2008) as follows:

$$S(t) = \prod_{t_i < t} \frac{R_i - d_i}{R_i}, \tag{2}$$

where d_i is the number of candidates who experience the event at time t_i, and R_i represents the so-called *risk set*, or candidates at risk of experiencing the event just prior to t_i. In Figure 1, the base function is the estimated survival probabilities over time for the full population. Only conversations of more than 5 utterances are considered; hence the survival probability is 1 for all curves up until $t = 5$. If we create subsets of the population by conditioning on the response time, the subset with a longer response time has a steeper decline, indicating that conversations where participants take longer to respond are more likely to end earlier. In survival regression, the aim is to learn survival regression functions based on such features, while the current time is modelled separately from them, unlike in standard regression models.

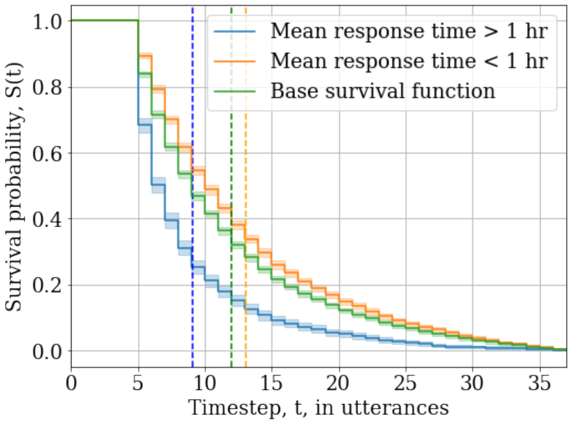

Figure 1: Survival functions for the conversation length prediction task, for the full population (orange) and subsets conditioning on the response time. Dashed lines indicate the expected event time per population.

To estimate the expected event time given a survival function, one can find the expected value of the survival function as follows:

$$\hat{T} = E[S] = \int_0^\infty S(t)dt. \tag{3}$$

These values are indicated with dashed lines in Figure 1, denoting the average conversation length on the full population and the two subsets based on the response time. The instantaneous risk of the event occurring at a point in time, i.e. the probability of the event time T being in a small interval around t, is defined by the *hazard function*:

$$h(t) = \lim_{dt \to 0} \frac{P(t \leq T < t + dt | T \geq t)}{dt}. \tag{4}$$

The cumulative hazard is given by $H(t) = \int_0^t h(t)dt$ and is related to the survival function according to $S(t) = e^{-H(t)}$.

Parameteric survival regression models (described in more detail in Section 2.3 and 2.4) are often optimised to predict either the survival or the hazard function, given that it is always possible to convert between them. Such models can include feature representations (such as the response time in Figure 1) to obtain individualised predictions.

2.2 Censoring

A common consideration in survival studies is the presence of censoring, where a participant leaves a study before the end of the observation period, or they do not experience the event of interest within this period. Under censoring, each subject i has an associated potential censoring time C_i and a potential lifetime T_i. We observe $Y_i = min\{T_i, C_i\}$, i.e. the minimum of the censoring and lifetimes, and an indicator variable δ_i for whether the observation ended with death or censoring.

Consider the task of predicting personal attacks (described in more detail in Section 3). Conversations that end without a personal attack occurring can be considered analogous to patients dropping out of a study before the end of the observation period. The duration of the observation can then be taken as the censoring time.

Different survival models account for censoring in different ways. For instance, for a survival curve estimated with the Kaplan-Meier method (Equation 2), censored individuals are removed from the set of candidates at risk (R_i) at the censoring time, without having experienced the event of interest.

2.3 Cox Proportional Hazards

The Cox Proportional Hazards (Cox-PH) model (Cox, 1972) is the most widely used model for survival analysis (Rodriquez, 2007; Kvamme et al., 2019). It has the following hazard function:

$$h(t|\mathbf{x};\theta) = h_0(t) \cdot exp(g(\mathbf{x};\theta)). \tag{5}$$

$h_0(t)$ represents the baseline hazard for the population at each timestep, such as the base survival function in Figure 1. The $g(\mathbf{x};\theta)$ term is often referred to as the *risk function* and specifies how the feature vector $\mathbf{x}$ of a sample is taken into account using parameters θ. In our experiments, we consider two variations of this approach:

Linear Cox The traditional Cox-PH model (Cox, 1972) uses a linear weighting of a feature vector to calculate the risk function as follows:

$$g(\mathbf{x};\theta) = \theta^T\mathbf{x}. \tag{6}$$

This model is still widely used in survival analysis research, e.g. Suchting et al. (2019); Zhang et al. (2018c).

DeepSurv DeepSurv (Katzman et al., 2018) uses a neural network to compute the risk function $g(\mathbf{x};\theta)$, where θ represents the weights of the network. The advantage of this is that the neural network can learn nonlinear features from the training data, which often improves predictive accuracy.

During training, the parameters θ are optimised with maximum likelihood estimation for both models. Given individuals i with event time T_i in dataset D, let R_i denote the risk set at T_i and δ_i the censoring indicator. Then, the likelihood of the data is given by:

$$\begin{aligned} L_{cox}(\theta, D) &= \prod_{i\in D}\left(\frac{h_0(t)\dot{e}xp(g(\mathbf{x_i};\theta))}{\sum_{j\in R_i} h_0(t)\dot{e}xp(g(\mathbf{x_j};\theta))}\right)^{\delta_i} \\ &= \prod_{i\in D}\left(\frac{exp(g(\mathbf{x_i};\theta))}{\sum_{j\in R_i} exp(g(\mathbf{x}_j;\theta))}\right)^{\delta_i}. \end{aligned} \tag{7}$$

Intuitively, we aim to maximise the risk of i experiencing an event, over all other candidates at risk at time T_i. In the context of predicting conversation lengths, this means that at time T_i, any conversation that has not yet ended could end, but we want to maximise the probability of the candidate that had indeed ended then over the rest. This is referred to as a *partial likelihood*, in reference to the fact that the effect of the features can be estimated without the need to model the change of the hazard over time. The indicator term δ expresses that only non-censored samples contribute terms that impact the likelihood (since the contribution of censored samples would be 1); however, the censored samples would be included in the risk set R_i up until their respective censoring times.

2.4 Survival regression as classification

A different approach to survival regression is to use classification to predict the timestep when an event will occur. DeepHit (Lee et al., 2018) is a neural network model that predicts a distribution over timesteps in this fashion. This provides more modelling flexibility compared to the Cox-PH models, where features are incorporated through the risk function and combined with a baseline hazard.

The model can incorporate multiple competing risks with distinct events of interest, and models censoring as a special type of risk. The output of the network is a vector representing the joint probability that the subject will experience each non-censoring event for every timestep t in the observation period. Censoring is assumed to take place at random and is therefore not included in the prediction. In the case of a single risk, it therefore predicts a vector: $\hat{y}_i = [\hat{y}_{t=0}, ..., \hat{y}_{t=t_{max}}]$, where each output element $\hat{y}_t$ represents the estimated probability $\hat{P}(t|\mathbf{x}, \theta)$ that a subject with feature vector $\mathbf{x}$ will experience the event at time t under the model parameters θ. Instead of a survival function, DeepHit defines a risk-specific *cumulative incidence function* (CIF) which expresses the probability that the event occurs before a time t^*, conditioned on features $\mathbf{x}^*$:

$$\begin{aligned} F(t^*|\mathbf{x}^*) &= \hat{P}(T \leq t^*|\mathbf{x}^*, \theta) \\ &= \sum_{t=0}^{t^*} \hat{P}(t|\mathbf{x}^*, \theta) = \sum_{t=0}^{t^*} \hat{y}_t^* \end{aligned} \tag{8}$$

The loss function for training DeepHit has two components: an event time likelihood and a ranking loss. The ranking loss ensures that earlier events are predicted to happen before later events based on their CIF, but does not penalise models for mispredicting the times in absolute terms. The event time likelihood maximises the probability of the event occurring at the right time ($y^{(i)}_{T^{(i)}}$), or, in the case of censoring, it maximises the probability of the event not happening before the censoring time $(1 - F(T^{(i)}|\mathbf{x}^{(i)}, \theta)$.

2.5 Previous applications in NLP

A small number of NLP studies have employed techniques from survival analysis for time-dependent tasks. Navaki Arefi et al. (2019) use survival regression to investigate factors that result in posts being censored on a Chinese social media platform, finding that negative sentiment is associated with shorter lifetimes. Stewart and Eisenstein (2018) use a linear Cox model to infer factors that are predictive of non-standard words falling out of use in online discourse, finding that words that appear in more linguistic contexts survive longer. Other applications include modelling fixation times in reading (Nilsson and Nivre, 2011) and evaluating dialogue systems (Deriu et al., 2020). However, none of these studies considered time-to-event prediction tasks based on conversations.

3 Survival regression in conversations

We evaluate survival models on two tasks, predicting conversation length and predicting when personal attacks will occur, where each conversation is a subject and the time is measured in utterances.[1]

Dataset	# Convs.	Median time to event	Task
Talk	16 896	6	**1**
Dispute	8 554	9	**1**
Attack	3 466	7	**2**

Table 1: Characteristics of datasets used in this paper. All three datasets originate from Wikipedia Talk pages.

Task 1: Predicting conversation length Having seen t utterances, predict the remaining conversation length in utterances. We use the dataset of Wikipedia Talk page discussions by Danescu-Niculescu-Mizil et al. (2012) (hereafter referred to as **Talk**) and the dataset of disagreements on Wikipedia Talk pages by De Kock and Vlachos (2021) (referred to as **Dispute**) for this task. The Talk dataset was also used to perform the thresholded classification version of this task in Backstrom et al. (2013), mentioned in Section 1.

Task 2: Predicting personal attacks Having seen t utterances, predict the number of utterances until a personal attack occurs. Conversations where no personal attack occurs are censored during training, and the conversation length is used as the observation time. Just less than half of the conversations contain personal attacks (1 569 out of 3 466). This is a novel task; previous work has only addressed predicting whether conversations will derail into personal attack, without attempting to predict when in a conversation this may occur (Zhang et al., 2018a; Chang and Danescu-Niculescu-Mizil, 2019). The motivation cited in both of the abovementioned studies is to prioritise conversations at risk of derailing for preemptive moderation. Survival models can give a more informative answer that takes into account the time until the attack, and therefore which conversations pose the most immediate risk. We use the dataset of Zhang et al. (2018a) for our experiments on this task.

Characteristics of the datasets we use are shown in Table 1. We use only conversations where the event of interest occurs after the fifth utterance, and we remove conversations longer than the 95th percentile as these are often flame wars which may have confounding impacts. Data is split into training, development and test sets with ratios 75:10:15.

3.1 Metrics

Two metrics are calculated to evaluate model performance: mean absolute error and concordance index. The mean absolute error (MAE) for a dataset

---

[1] The task of predicting when users would cease to use a platform would also have been an interesting case for this study; however, the datasets of Danescu-Niculescu-Mizil et al. (2013) are no longer available.

of n test samples is defined as

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}. \quad (9)$$

This metric provides an easily interpretable score, and it is commonly used in evaluating regression models, e.g. Bitvai and Cohn (2015). However, MAE is not robust to outliers; large errors on a few values can outweigh many correct predictions.[2] MAE is also ill-defined in the presence of censoring as there is no event time to compare against, and it cannot be used to compare model performance between different datasets. For these reasons, we also include the concordance index (Harrell Jr et al., 1996), which is concerned with ordering rather than absolute values. A pair of observations i, j is considered concordant if the prediction and the ground truth have the same inequality relation, i.e. $(y_i > y_j, \hat{y}_i > \hat{y}_j)$ or $(y_i < y_j, \hat{y}_i < \hat{y}_j)$. The concordance index (**CI**) is the fraction of concordant pairs. A random model, or a model that predicts the same value for every sample, would yield a score of 0.5. A perfect score is 1. In the presence of censoring, censored samples are only compared with uncensored samples of a smaller event time, since it is known in that case that the uncensored sample should be assigned a later event time.

A disadvantage of the CI score is that it does not reflect how accurate the predictions are in absolute terms, meaning that good CI scores can be achieved with predictions in the wrong range. The two scores thus provide complementing views on model performance.

3.2 Features

The features we consider are based on previous work on conversation length prediction and predicting personal attacks. These are:

- Politeness (**POL**): The politeness strategies from Zhang et al. (2018a) as implemented in Convokit (Chang et al., 2020), which capture greetings, apologies, and saying “please”, etc.
- Arrival sequences (**ARR**): The order in which speakers partake in the first 5 utterances, defined by Backstrom et al. (2013).
- Hypergraph (**HYP**): Conversation structure features based on the reply tree, proposed by Zhang et al. (2018b) and implemented in Convokit (Chang et al., 2020). These features capture dynamics between participants, such as engagement and reciprocity.
- Sentiment (**SENT**): Positive and negative sentiment word counts, as per the lexicon of Liu et al. (2005), also implemented in Convokit.
- Time features (**TIME**): Log mean time between utterances and time between last two utterances, inspired by Backstrom et al. (2013).
- Utterance lengths (**LEN**): Log mean utterance length features, measured in tokens.
- Number of participants (**PART**): Also used in Backstrom et al. (2013) and Zhang et al. (2018b).
- Turn-taking features (**TURNS**): The fraction of turns and tokens contributed by the top user, inspired by Niculae et al. (2015).

For the POL, SENT, TIME and LEN features, we include both the mean value throughout the conversation and the gradient of a straight-line fit to capture how the feature changes throughout it. All features are calculated up to the point of prediction, and not for the full conversation.

4 Results

4.1 Experimental setup

We use partly conditional training (Zheng and Heagerty, 2005) to account for using features that change over time, such as politeness, in contrast with static features like the arrival sequence. Under partly conditional training, a feature measured at time t predicts the risk of the occurrence of an event at a future time T. In our case, each individual is a conversation and features are measured after every utterance. Each measurement t of a conversation i is recorded as an individual entry in the dataset, with event time $T_{i,t} = T_i - t$.

This construction is illustrated in Table 2 for the Talk dataset. There are 307 conversations that contain 12 utterances, but 0 samples of length 12 in the training data, since all conversations of this length have 0 utterances remaining and regression is therefore unnecessary. However, we include the first 11 utterances of the length-12 conversations in the training set at $t = 11$, since the remaining length here could be either 0 or 1. As such, there are (642+307=) 769 samples of length 11. We use

---

[2] This issue is even more pronounced in the root mean-squared-error, which is another popular metric for regression (Hyndman and Koehler, 2006).

t	# Convs. of length t	# Samples of length t
5	6 596	16 896
6	4 010	10 300
7	2 358	6 290
8	1 539	3 932
9	984	2 393
10	640	1 409
11	462	769
12	307	0
Total	16 896	41 989

Table 2: Training set configuration for the Talk dataset. For every conversation, we add a snapshot of its feature values at every timestep to the training data.

a minimum value of $t = 5$ to ensure there is sufficient information from which to make a prediction. Details for the other datasets are in Appendix A.

Our **baseline** model is a univariate Kaplan-Meier estimator (Jager et al., 2008), which predicts the same event time for all samples without taking features into account. For this model and the linear Cox-PH model, we use the implementations in `lifelines`[3]. We use grid search on the validation set for each model to determine hyperparameter values, experimenting with regularisation values in $[0, 0.01, 0.1, 0.5]$, L1 ratios in $[0, 0.1, 0.5, 1]$ and learning rates in $[0.01, 0.1, 0.5, 1]$. We also compare to a **linear regression** model, implemented in `scikit-learn`[4] and using the same features. For the linear regression model, we truncate predictions at 0 since negative times are invalid. Finally, to compare to previous work on **threshold classification**, we implement a logistic regression classifier, using the median of each training set as the cut-off point. For these models, the upper and lower quartiles are used to compute the MAE. For instance, for the Dispute dataset, the threshold value is 9. To calculate MAE, we use an event time of 5 if the model predicts the shorter class and 12 for the longer class.

For the neural models (DeepSurv and DeepHit), we use the implementations in `PyCox`[5] by Kvamme et al. (2019). For both we use two hidden layers with $[128, 64]$ nodes, dropout ($p = 0.3$), batch normalisation, and the Adam optimiser (Kingma and Ba, 2014) with learning rate 0.01.

4.2 Task 1: Predicting conversation length

Results for the conversation length prediction task are shown in Table 3 for the Dispute and Talk datasets (left and middle columns respectively). MAE scores should not be compared between datasets since the datasets have different length distributions, with the Dispute dataset having conversations of up to 37 utterances, compared to a maximum of 12 in the Talk dataset.

For both datasets, all survival models outperform the linear regression and threshold classification models on the MAE metric. The survival baseline uses only population-level knowledge of the event time distribution, and predicts the same event time for all samples, whereas the other baselines take into account information from the features and can therefore tailor predictions per sample. While this results in the survival baseline having the worst CI (0.5), it is still better than linear reg and threshold in terms of MAE, illustrating the importance of separating the effect of time from the other features; time alone can be highly predictive.

The DeepHit model performs the best on the MAE metric on both datasets, with a statistically significant difference from the Linear Cox model at the P=0.01 level using the sign test. The latter performs better than DeepHit on the CI metric for the Dispute dataset, however, this difference is not statistically significant (P=0.869, using a randomised permutation test).

Coefficient analysis Since the linear Cox model performed well on the Dispute dataset and is more interpretable than its deep counterparts, we show its 10 largest coefficients in absolute value in Table 4. Positive weights are associated with larger risk function values, and therefore a shorter conversation. Time between utterances is the most predictive feature, with a longer time between utterances correlating positively with shorter conversations (also observed in Figure 1). This corroborates the findings of Backstrom et al. (2013) and Zhang et al. (2018b). Having more participants is also correlated with shorter conversations. This suggests that the conversations in the Dispute dataset are less prone to long expansionary-style threads (as defined by Backstrom et al. (2013)), where many participants each contribute one utterance. Given the dataset consists of disagreements, it is not surprising that there would rather be focused discussions between a small number of participants.

Features 4, 6, 7 and 8 are from the hypergraph

---

[3] `lifelines.readthedocs.io`

[4] `scikit-learn.org`

[5] `github.com/havakv/pycox/tree/master/pycox`

Model	**TASK 1**				**TASK 2**	
	Dispute		**Talk**		**Attack**	
	MAE ↓	**CI ↑**	**MAE ↓**	**CI ↑**	**MAE ↓**	**CI ↑**
Linear regression	6.213	0.560	1.329	0.542	1.566	0.497
Threshold classification	6.652	0.545	1.505	0.462	1.521	0.489
Survival baseline	5.186	0.500	1.312	0.500	1.585	0.500
Linear Cox	4.995	**0.581**	1.282	0.573	1.481	0.605
DeepSurv	5.014	0.567	1.276	0.575	1.487	0.601
DeepHit	**4.926**	0.578	**1.189**	**0.584**	**1.403**	**0.627**

Table 3: MAE and CI for Task 1 (predicting conversation length) and Task 2 (predicting personal attacks), using the Dispute, Talk and Attack datasets. For MAE, lower values are preferred; for CI, higher. The bold values indicate the best model per metric, for each dataset. Statistical significance is discussed in the text.

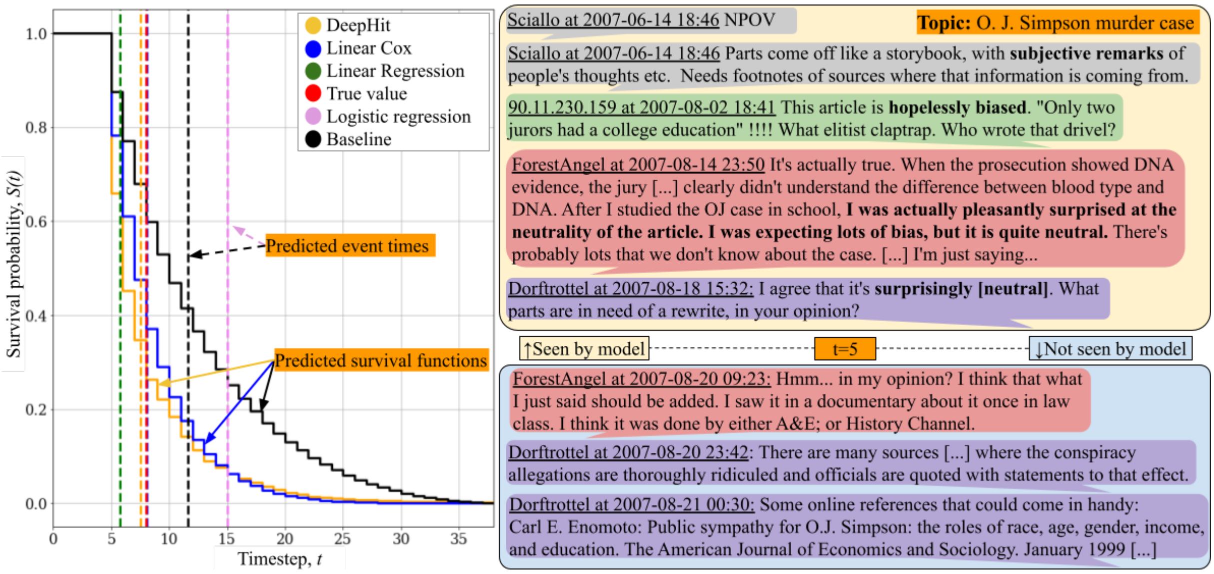

Sciallo at 2007-06-14 18:46 NPOV

Sciallo at 2007-06-14 18:46 Parts come off like a storybook, with **subjective remarks** of people's thoughts etc. Needs footnotes of sources where that information is coming from.

90.11.230.159 at 2007-08-02 18:41 This article is **hopelessly biased**. "Only two jurors had a college education" !!!! What elitist claptrap. Who wrote that drivel?

ForestAngel at 2007-08-14 23:50 It's actually true. When the prosecution showed DNA evidence, the jury [...] clearly didn't understand the difference between blood type and DNA. After I studied the OJ case in school, **I was actually pleasantly surprised at the neutrality of the article. I was expecting lots of bias, but it is quite neutral.** There's probably lots that we don't know about the case. [...] I'm just saying...

Dorftrottel at 2007-08-18 15:32: I agree that it's **surprisingly [neutral]**. What parts are in need of a rewrite, in your opinion?

ForestAngel at 2007-08-20 09:23: Hmm... in my opinion? I think that what I just said should be added. I saw it in a documentary about it once in law class. I think it was done by either A&E; or History Channel.

Dorftrottel at 2007-08-20 23:42: There are many sources [...] where the conspiracy allegations are thoroughly ridiculed and officials are quoted with statements to that effect.

Dorftrottel at 2007-08-21 00:30: Some online references that could come in handy: Carl E. Enomoto: Public sympathy for O.J. Simpson: the roles of race, age, gender, income, and education. The American Journal of Economics and Sociology. January 1999 [...]

Figure 2: Example of a conversation in the Dispute dataset and the predictions of different models at $t = 5$. For this sample, the prediction from the linear Cox model matches the true value.

Feature	**Type**	**Coef.**
1. Time between last two utterances	TIME	0.161
2. Mean time between utterances	TIME	0.121
3. # participants	PART	0.045
4. Mean # replies received per post	HYP	0.042
5. # first person pronouns, mean	POL	0.037
6. Mean of non-zero # repliers per user	HYP	-0.035
7. Fraction of users with more than 1 replier	HYP	-0.035
8. Triadic replies to commenters in mid-thread	HYP	0.034
9. Length of last utterance	LEN	-0.031
10. Arrival sequence 00110	ARR	-0.026

Table 4: Coefficients of the linear Cox model for Task 1, using the Dispute dataset. Positive weights are associated with shorter time-to-event values.

feature set, which describes the structure of the reply-tree. Feature 4 indicates that a thread which forms a shallow tree, with posts receiving many direct responses, is likely to terminate soon. Features 6 and 7 indicate that interactions with multiple users is likely to extend the conversation. The length of the last utterance before the prediction is made (feature 9) is negatively associated with a short time-to-event, indicating that a long last utterance means that the conversation is still ongoing. Finally, the arrival sequence 00110 (feature 10) encodes the order in which two participants (indexes 0 and 1) contributed the first 5 utterances.

The only language-related feature in the 10 most predictive features is feature 5, the number of first person pronouns, which is positively correlated with a shorter time-to-event. Seven of the ten features on this list are from previous work on predicting conversation length (TIME, ARR and HYP), although this was for the thresholded classification version of the task. This indicates that the survival models are inferring similar relationships as the classification variant, while providing a more informative prediction and better performance on the MAE and CI metrics.

Results per timestep To gain an understanding for how our models perform at different timesteps in the conversation, we also evaluate the predictive accuracy at every utterance index. The intuition here is illustrated in Figure 2 for $t = 5$, with the task being: having seen 5 utterances, predict the conversation length. We can see that the actual conversation length is 8, as there are 3 more utterances after the prediction, which are not seen by the model. The linear Cox model predicts the right value in this case. We also show the survival functions predicted by DeepHit, the baseline and the linear Cox model. As explained in Section 2 (Equation 3), the predictions are the expected values of the survival function. We depict predictions in relation to the prediction time; for instance, the linear Cox model predicts a time-to-event of 3 utterances at timestep 5, meaning that the event time will be timestep 8.

Figure 3 shows the MAE and CI scores, aggregated per timestep, for the Talk dataset. Lower MAE scores are observed for later timesteps. However, this does not mean that these models are necessarily better; the possible range of error is smaller later in a conversation. An interesting deviation here is the linear regression model, which performs the best at the last timestep. Upon inspection, we note that this is because the model predicts small values (with median values in range 0.02-0.09) at every timestep. This strategy is likely the result of the dataset being biased towards shorter conversations. At smaller values of t, there is a portion of long conversations which would contribute large errors to drive up the MAE, but this is not present at larger t, hence the discrepancy.

CI allows for more direct comparison of models at different t, since it measures ranking accuracy. On this metric, DeepHit performs better than the linear regression model at all but the last timestep. Both models perform slightly worse at the last two timesteps. A reason for this may be that there are fewer training samples available at larger t, as illustrated in Table 2. Similar trends are observed in the Dispute dataset.

4.3 Task 2: Predicting personal attacks

Results for the personal attack prediction task are shown in the right column of Table 3. Compared to Task 1, higher values are observed on the CI. A reason for this may be that conversation length prediction has to rely on more subtle cues that indicate a conversation has run its course (e.g. users signing off) which are not captured by our features.

We observe again that the DeepHit model performs the best, and that the survival models outperform the three baselines. Due to censoring, the MAE score here is calculated using only uncensored samples; i.e. samples where a personal attack does occur. The censored samples are accounted for in the CI metric, as explained in Section 3.1.

A key question in this task is whether the survival models manage to prioritise samples where a personal attack occurs over censored examples. This means that when comparing a pair consisting of a censored and an uncensored example, we would like for the predicted time to event of the censored example to be higher. We can calculate how often this is true for the DeepHit model by calculating the concordance index of the predicted event times and the inverse of the censoring indicator. For instance, given a pair of samples $s = [censored, uncensored]$, the indicator function is [0,1] and the inverse therefore [1,0]. The ordering of the inverse ($s_0 > s_1$) should be concordant with the predictions. Using the DeepHit model, we find that the model ranks samples where an attack occurs over censored examples in 57.75% of cases, compared to 51.92% with the linear regression model, and 50% for a random baseline. The best model of Zhang et al. (2018a) has a predictive accuracy of 64.9% for classifying which conversations will derail into personal attacks, but does not predict when this will occur.

5 Conclusions

In this paper, we proposed that survival analysis is a useful but hitherto ignored framework for time-to-event prediction tasks in conversations, which are frequently framed as classification tasks. We provided evidence to this by showing that survival models outperform both linear regression and logistic regression models on two tasks and three datasets. The survival regression models explored can be useful in other tasks, for instance, predicting escalation in customer service conversations.

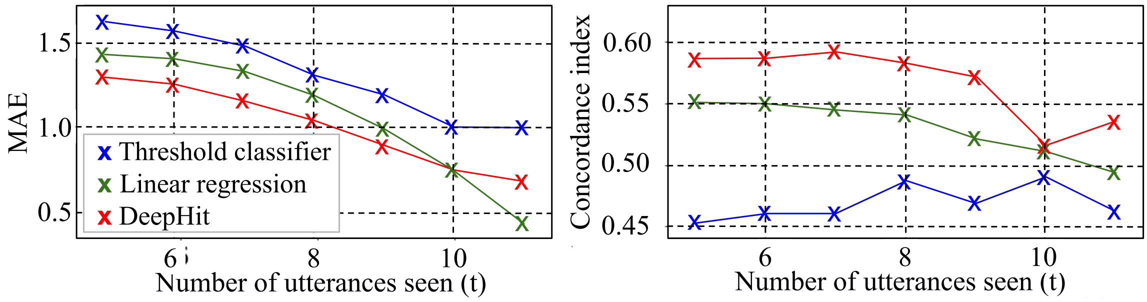

Figure 3: MAE and CI calculated per timestep, t, for the Talk dataset on Task 1. For MAE, lower values are preferred; for CI, higher.

Acknowledgements

Christine De Kock is supported by scholarships from Huawei and the Oppenheimer Memorial Trust. Andreas Vlachos is supported the EPSRC grant no. EP/T023414/1: Opening Up Minds.

References

Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and curating conversation threads: expansion, focus, volume, re-entry. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 13–22.

Zsolt Bitvai and Trevor Cohn. 2015. Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 180–185, Beijing, China. Association for Computational Linguistics.

Jonathan P. Chang, Caleb Chiam, Liye Fu, Andrew Wang, Justine Zhang, and Cristian Danescu-Niculescu-Mizil. 2020. ConvoKit: A toolkit for the analysis of conversations. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 57–60, 1st virtual meeting. Association for Computational Linguistics.

Jonathan P. Chang and Cristian Danescu-Niculescu-Mizil. 2019. Trouble on the horizon: Forecasting the derailment of online conversations as they develop. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4743–4754.

David R Cox. 1972. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*, pages 699–708.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd international conference on World Wide Web*, pages 307–318.

Christine De Kock and Andreas Vlachos. 2021. I beg to differ: A study of constructive disagreement in online conversations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2017–2027, Online. Association for Computational Linguistics.

Jan Deriu, Don Tuggener, Pius von Däniken, Jon Ander Campos, Alvaro Rodrigo, Thiziri Belkacem, Aitor Soroa, Eneko Agirre, and Mark Cieliebak. 2020. Spot the bot: A robust and efficient framework for the evaluation of conversational dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3971–3984, Online. Association for Computational Linguistics.

Frank E Harrell Jr, Kerry L Lee, and Daniel B Mark. 1996. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in medicine*, 15(4):361–387.

Rob J Hyndman and Anne B Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688.

Kitty J Jager, Paul C Van Dijk, Carmine Zoccali, and Friedo W Dekker. 2008. The analysis of survival data: the kaplan–meier method. *Kidney international*, 74(5):560–565.

Jared L. Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. 2018. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. 2019. Time-to-event prediction with neural networks and cox regression. *Journal of Machine Learning Research*, 20(129):1–30.

Changhee Lee, William Zame, Jinsung Yoon, and Mihaela van der Schaar. 2018. Deephit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351.

Meisam Navaki Arefi, Rajkumar Pandi, Michael Carl Tschantz, Jedidiah R. Crandall, King-wa Fu, Dahlia Qiu Shi, and Miao Sha. 2019. Assessing post deletion in Sina Weibo: Multi-modal classification of hot topics. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 1–9, Hong Kong, China. Association for Computational Linguistics.

Vlad Niculae, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. 2015. Linguistic harbingers of betrayal: A case study on an online strategy game. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1650–1659, Beijing, China. Association for Computational Linguistics.

Mattias Nilsson and Joakim Nivre. 2011. A survival analysis of fixation times in reading. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 107–115, Portland, Oregon, USA. Association for Computational Linguistics.

G. Rodriquez. 2007. Lecture notes on generalized linear models. `https://data.princeton.edu/wws509/notes/`.

Ian Stewart and Jacob Eisenstein. 2018. Making "fetch" happen: The influence of social and linguistic context on nonstandard word growth and decline. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4360–4370, Brussels, Belgium. Association for Computational Linguistics.

Robert Suchting, Emily T Hebert, Ping Ma, Darla E Kendzor, and Michael S Businelle. 2019. Using elastic net penalized cox proportional hazards regression to identify predictors of imminent smoking lapse. *Nicotine and Tobacco Research*, 21(2):173–179.

Justine Zhang, Jonathan P Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Nithum Tahin, and Dario Taraborelli. 2018a. Conversations gone awry: Detecting early signs of conversational failure. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.*, volume 1.

Justine Zhang, Cristian Danescu-Niculescu-Mizil, Christina Sauper, and Sean J Taylor. 2018b. Characterizing online public discussions through patterns of participant interactions. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–27.

Zhongheng Zhang, Jaakko Reinikainen, Kazeem Adedayo Adeleke, Marcel E Pieterse, and Catharina GM Groothuis-Oudshoorn. 2018c. Time-varying covariates and coefficients in cox regression models. *Annals of translational medicine*, 6(7).

Yingye Zheng and Patrick J Heagerty. 2005. Partly conditional survival models for longitudinal data. *Biometrics*, 61(2):379–391.

A Characteristics of the training data

Details of the Dispute and Attack datasets are shown in Tables 6 and 5.

t	**# Convs. of length t**	**# Samples of length** t
5	480	3 466
6	959	2 986
7	699	2 027
8	473	1 328
9	371	855
10	312	484
11	172	0
Total	**3 466**	**11 146**

Table 5: Training set configuration for the Attack dataset.

t	**# Convs. of length t**	**# Samples of length t**
5	1 374	9 928
6	1 044	8 554
7	833	7 180
8	710	6 136
9	571	5 303
10	467	4 593
11	427	4 022
12	372	3 128
13	328	2 756
14	315	2 428
15	260	2 113
16	181	1 853
17	185	1 672
18	141	1 487
19	153	1 346
20	144	1 193
21	100	1 049
22	112	949
23	106	837
24	90	731
25	68	641
26	70	573
27	69	503
28	60	434
29	60	374
30	48	314
31	43	266
32	54	223
33	31	169
34	38	138
35	40	100
36	30	60
37	30	0
Total	**8 554**	**74 608**

Table 6: Training set configuration for the Dispute dataset.

Unsupervised Knowledge Selection for Dialogue Generation

Xiuyi Chen, Feilong Chen, Fandong Meng, Peng Li, Jie Zhou
Pattern Recognition Center, WeChat AI, Tencent Inc, Beijing, China
{hugheren.chan,ivess.chan}@gmail.com
{fandongmeng,patrickpli,withtomzhou}@tencent.com

Abstract

Knowledge selection is an important and challenging task which could provide the appropriate knowledge for informative dialogue generation. However, the needed gold knowledge label is difficult to collect in reality. In this paper, we study knowledge selection for dialogue generation in the unsupervised scenario and propose a novel Distilled Distant Supervision Loss (DDSL) to supervise knowledge selection when the gold knowledge label is unknown. Specifically, we first obtain an oracle knowledge label via distant supervision and then leverage knowledge distillation to alleviate the noisy labeling problem of distant supervision. Furthermore, we propose a pretraining-finetuning strategy to deal with the mismatch knowledge selection problem that models tend to select the mismatched knowledge for dialogue generation in the unsupervised setting and will cause the degeneration of knowledge-aware decoder. Experiments on two knowledge-grounded dialogue datasets show that our approach manages to select knowledge more accurately in the unsupervised setting and generates more informative responses, even outperforming many strong supervised baselines.[1]

1 Introduction

To avoid general and dull dialogue generation (Li et al., 2016), knowledge-grounded dialogue which equips dialogue systems with external knowledge has become a popular research topic. Thanks to the hand-collected knowledge-grounded dialogue datasets which align each dialogue (even each utterance) with a pre-identified document (Zhang et al., 2018; Zhou et al., 2018; Moghe et al., 2018; Zhou et al., 2020), many researchers focus on injecting the given knowledge to generate informative responses and achieve promising results (Yavuz et al., 2019; Tang and Hu, 2019; Qin et al., 2019a; Li et al., 2019b; Zheng and Zhou, 2019; Meng et al., 2019; Ren et al., 2019; Ye et al., 2020; Lin et al., 2020). However, they usually need the pre-identified knowledge and the knowledge access task is less studied (Kim et al., 2020b) which is the precursor to knowledge dialogue generation in reality (Zheng et al., 2020).

It is natural to extract the external knowledge via information retrieval technology. Several works regard the retrieved knowledge sentences as the pre-identified document (Ghazvininejad et al., 2018; Michel Galley, 2018; Yang et al., 2019b; Gopalakrishnan et al., 2019). However, the retrieved document contains redundant and irrelevant information which are harmful for dialogue generation (Zhao et al., 2020b). Hence, knowledge selection which chooses an appropriate sentence from the pre-retrieved knowledge pool gains much attention and it plays the role of knowledge access for generation. Dinan et al. (2019) first propose knowledge selection for dialogue generation which are two sequential subtasks and the generation is based on the selected knowledge. Several works follow their setting and achieve improvements with latent variable models (Kim et al., 2020a; Chen et al., 2020b) or more complex selection mechanism (Niu et al., 2019; Meng et al., 2020; Zheng et al., 2020; Meng et al., 2021). Although those works show promising performance with explicit use of knowledge in open-domain dialogue, they still need gold knowledge labels to train the selection module well (Kim et al., 2020a). And it is still less studied to make knowledge selection work well without gold knowledge label, which is valuable and challenging.

In this paper, we explore knowledge selection for dialogue generation in the unsupervised scenario and propose a novel Distilled Distant Supervision Loss (DDSL) to supervise knowledge selection when there is no gold knowledge label. Specifically, we first obtain an oracle knowledge

---

[1] The codes and model checkpoints will be available at `https://github.com/ErenChan/UKSDG`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1230–1244
August 1–6, 2021.

label via distant supervision (Mintz et al., 2009) to substitute the gold one in the unsupervised setting. However, distant supervision inevitably suffers from the noisy labeling problem due to literally matching (Yang et al., 2019a). Therefore, to train knowledge selection well without gold label, we leverage knowledge distillation to reduce the noise of the oracle label. Furthermore, we find that models tend to select the mismatched knowledge for dialogue generation in the unsupervised setting. And forcing the knowledge-aware decoder to leverage the selected knowledge at training will cause the decoder degenerating into the knowledge-independent decoder. To deal with this problem, we propose to pretrain the knowledge selection and response generation independently and then finetune the decoder with the selected knowledge using different sample weighting scores. We demonstrate the effectiveness of our unsupervised approach on two knowledge-grounded dialogue datasets, i.e., Wizard of Wikipedia (Dinan et al., 2019) and Holl-E (Moghe et al., 2018) in comparison with various supervised and unsupervised baselines.

Our contributions are summarized as follows:

- We propose Distilled Distant Supervision Loss to make knowledge selection work well in the unsupervised scenario where the gold knowledge label is not available.
- We propose a pretraining-finetuning strategy to alleviate the degeneration of knowledge-aware decoder caused by the mismatch knowledge selection problem.
- Results on two datasets show that our approach manages to select knowledge more accurately in the unsupervised setting and even generates more informative responses than many strong supervised baselines.

2 Approach

2.1 Task Formulation

Given the utterance x_t at each turn t and the associated knowledge pool $\mathcal{K}_t = \{k_t^i\} = \{k_t^1, \cdots, k_t^L\}$[2] containing L retrieved candidate sentences k_t^i, the final goal is to generate an informative response y_t. Following (Dinan et al., 2019), we first learn to select the appropriate knowledge k_t^s from the knowledge pool $\mathcal{K}_t$ and then generate the response y_t by incorporating the selected knowledge. In the conventional supervised setting, there exists gold knowledge label to supervise knowledge selection. However, the manually labeled knowledge is difficult to obtain in reality (Lian et al., 2019). As a result, we study the unsupervised knowledge selection for dialogue generation in this paper, which is very valuable and challenging.

2.2 Architecture Overview

In the following subsections, we first introduce the three major components (Section 2.3 $\sim$ 2.5): Encoder, Knowledge Selection (KS) and Decoder, which are trained jointly with the gold knowledge loss and response generation loss in the conventional supervised setting, as Figure 1 (a) shows. Then we introduce our Distilled Distant Supervision Loss (DDSL) in Section 2.6 to train knowledge selection well in the unsupervised setting, which consists of distant supervision and knowledge distillation, as Figure 1 (b) shows. Finally, we detail the mismatch knowledge selection problem and how to make the decoder leverage the selected knowledge k_t^s well in Section 2.7.

2.3 Encoder

For each sentence st, we obtain the context aware word representations $\mathbf{H}_{st}$ via BERT (Devlin et al., 2019) and the corresponding sentence representation $\mathbf{h}_{st}$ via Mean Pooling (Cer et al., 2018):

$$\begin{aligned} \mathbf{H}_{st} &= \mathrm{BERT}\left(st\right) \in \mathbb{R}^{N_{st} \times d} \\ \mathbf{h}_{st} &= \mathrm{Mean}\left(\mathbf{H}^{st}\right) \in \mathbb{R}^{d} \end{aligned} , \tag{1}$$

where N_{st} is the sentence length and d is the hidden size. Specifically, we represent the utterance x_t with $\mathbf{H}_{x_t}$ and $\mathbf{h}_{x_t}$, and represent each knowledge sentence $k_t^i \in \mathcal{K}_t$ with $\mathbf{H}_{k_t^i}$ and $\mathbf{h}_{k_t^i}$.

2.4 Knowledge Selection

In this paper, we mainly focus on knowledge selection in the unsupervised setting and adopt the standard dot-product attention over the knowledge candidates to select knowledge (Dinan et al., 2019).

Selection Query: The selection query consists of the current utterance, the dialogue history $dh_t = [x_1, y_1, \cdots, x_{t-1}, y_{t-1}]$ and the history of selected knowledge $kh_t = [k'_1, \cdots, k'_{t-1}]$ as they help the knowledge selection (Kim et al., 2020a). Formally, we use two GRUs (Cho et al., 2014) to summarize the dialogue and knowledge selection history as the

---

[2] k_t^i, k_t^s, k_t **and** k'_t indicate **any, selected, gold and oracle** knowledge sentences or labels, respectively.

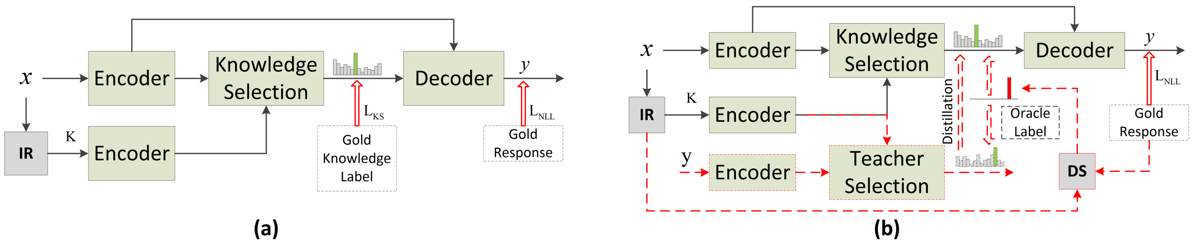

Figure 1: The framework of knowledge selection for dialogue generation. (a) In the supervised setting. (b) In the unsupervised setting. Specifically, we replace the gold knowledge loss with our Distilled Distant Supervision Loss (DDSL) which contains Distant Supervision (DS) and knowledge distillation (see the dotted red lines).

corresponding state vectors $\mathbf{s}_{dh_t}$ and $\mathbf{s}_{kh_t}$:

$$\begin{aligned} \mathbf{s}_{dh_t} &= \mathrm{GRU}_{\mathrm{dh}}\left(\left[\mathbf{h}_{x_t}; \mathbf{h}_{y_t}\right], \mathbf{s}_{dh_{t-1}}\right) \in \mathbb{R}^d \\ \mathbf{s}_{kh_t} &= \mathrm{GRU}_{\mathrm{kh}}\left(\mathbf{h}_{k'_t}, \mathbf{s}_{kh_{t-1}}\right) \in \mathbb{R}^d \end{aligned} , \quad (2)$$

where $\mathbf{s}_{dh_0}$ and $\mathbf{s}_{kh_0}$ are zero vectors, $\mathbf{h}_{x_t}$, $\mathbf{h}_{y_t}$ and $\mathbf{h}_{k'_t}$ are sentence vectors of utterance x_t, response y_t and the oracle knowledge k'_t (will be described in Equation 8) and $[\cdot;\cdot]$ denotes concatenation. Then, the selection query $\mathbf{q}_t$ is obtained:

$$\mathbf{q}_t = \mathbf{W}_q \left[\mathbf{s}_{kh_{t-1}}; \mathbf{s}_{dh_{t-1}}; \mathbf{h}_{x_t}\right] \in \mathbb{R}^d. \quad (3)$$

Knowledge Selection: The knowledge selection distribution $\mathcal{S} \in \mathbb{R}^L$ over the knowledge pool $\mathcal{K}_t \in \mathbb{R}^L$ is obtained by the dot-product attention:

$$\mathcal{S}_{k_t^i} = \frac{\exp\left(\mathbf{h}_{k_t^i} \cdot \mathbf{q}_t\right)}{\sum_{k_t^j \in \mathcal{K}_t} \exp\left(\mathbf{h}_{k_t^j} \cdot \mathbf{q}_t\right)}. \quad (4)$$

Finally, the knowledge k_t^{s} with the highest attention score is selected for further response generation. If the gold knowledge k_t exists, we could train this task via the Cross Entropy (CE) loss:

$$\mathcal{L}_{\mathrm{KS}} = \mathcal{L}_{\mathrm{CE}}\left(\mathcal{S}, k_t\right) = -\log \mathcal{S}_{k_t}, \quad (5)$$

2.5 Decoder

Following (Dinan et al., 2019; Kim et al., 2020a), our Transformer-based decoder takes the representation concatenation $\mathbf{H}_{rc} = \left[\mathbf{H}_{x_t}; \mathbf{H}_{k_t^s}\right] \in \mathcal{R}^{N_{rc},d}$ of current utterance x_t and the selected knowledge k_t^{s} as input, and uses the copy mechanism (Gu et al., 2016) to generate responses. The process of generating a word can be formulated as follows:

$$\begin{aligned} &\mathbf{s}_t^n = \mathrm{TD}\left(\mathbf{H}_{rc}, y_t^{<n}\right) \in \mathbb{R}^d \\ &\mathbf{p}_{\mathrm{v}} = \mathrm{softmax}\left(\mathbf{W}_o \mathbf{s}_t^n\right) \in \mathbb{R}^{|\mathcal{V}|} \\ &\mathbf{p}_{\mathrm{cp}}, \tilde{\mathbf{s}}_t^n = \mathrm{MultiHead}_{\mathrm{cp}}\left(\mathbf{s}_t^n, \mathbf{H}_{rc}, \mathbf{H}_{rc}\right) \\ &p_{\mathrm{gen}} = \mathrm{sigmoid}\left(\mathbf{W}_{\mathrm{gen}} \tilde{\mathbf{s}}_t^n\right) \in \mathbb{R}^1 \\ &\mathbf{p}\left(\mathcal{V}\right) = p_{\mathrm{gen}} \cdot \mathbf{p}_{\mathrm{v}} + \left(1 - p_{\mathrm{gen}}\right) \cdot \mathbf{p}_{\mathrm{cp}} \in \mathbb{R}^{|\mathcal{V}|} \end{aligned} , \quad (6)$$

where TD denotes the Transformer decoder, $\mathbf{s}_t^n$ is the hidden vector for the n-th word in the response y_t at t-th turn, $\mathbf{p}_{\mathrm{cp}}$ is the attention weight of the first head in the additional multi-head self-attention layer for the copy mechanism, which is short for $\mathrm{MultiHead}_{\mathrm{cp}}$ (Vaswani et al., 2017), $\mathcal{V}$ is the vocabulary, and $\mathbf{p}\left(\mathcal{V}\right)$ is the final generation distribution. Finally, we generate the word y_t^n with the highest probability, and we keep generating by feeding y_t^n to the decoder until the "$\langle$eos$\rangle$" token is generated. We train the generation task the Negative Log-Likelihood (NLL) loss:

$$\mathcal{L}_{\mathrm{G}} = \mathcal{L}_{\mathrm{NLL}} = -\log p\left(y_t | x_t, k_t^{\mathrm{s}}\right) \quad (7)$$

The model is trained with the loss $\mathcal{L} = \mathcal{L}_{\mathrm{KS}} + \mathcal{L}_{\mathrm{G}}$, where $\mathcal{L}_{\mathrm{KS}}$ is the knowledge selection loss, i.e., Equation 5 in the supervised setting.

2.6 Distilled Distant Supervision Loss

In this section, we will introduce our Distilled Distant Supervision Loss (DDSL) to train knowledge selection well in the unsupervised setting, which consists of distant supervision, label weighting and knowledge distillation. Actually, we first obtain a noisy oracle knowledge label via distant supervision. And then our DDSL tries to reduce the noise via label weighting and knowledge distillation.

Distant Supervision: Suppose that we have the utterance x_t, response y_t and the retrieved knowledge pool $\mathcal{K}_t$ without knowing the gold knowledge label, we first calculate the confidence score $\mathcal{W}_{k_t^i}$ whether each knowledge k_t^i is matched up with this dialogue flow by:

$$\begin{aligned} \mathrm{F1}\left(a, b\right) &= \frac{2 \cdot |\,\mathrm{set}(a) \cap \mathrm{set}(b)|}{|\,\mathrm{set}(a)| + |\,\mathrm{set}(b)| + \epsilon}, \\ \mathcal{W}_{k_t^i} &= \mathrm{softmax}_\tau\left(\mathrm{F1}\left(k_t^i, y_t\right)\right) \end{aligned} \quad (8)$$

where $\mathrm{set}(a)$ or $\mathrm{set}(b)$ indicates the tokens in the string a or b, $\mathrm{softmax}_\tau\left(z_i\right) = e^{(z_i/\tau)} / \sum_j e^{(z_j/\tau)}$

and τ is the temperature to reshape the confidence probability distribution $\mathcal{W} \in \mathbb{R}^L$ over the knowledge pool. Then, we obtain the oracle knowledge k'_t with the highest confidence score, assuming that the gold knowledge should contribute most tokens to the response generation. This is possible because that 1) the causal modeling specified conditions (Selltiz et al., 1976) hold between knowledge selection and response generation (Tuan et al., 2020), and 2) it is common for humans to (involuntarily) produce utterances which are copied or suitably modified from background knowledge sentences (Moghe et al., 2018).

Although we can directly replace the gold label k_t in Equation 5 with the alternative one k'_t, there are some noise. Therefore, we modify Equation 5 with label weighting via the confidence score:

$$\mathcal{L}_{\text{KS}} = \mathcal{L}_{\text{WCE}}(\mathcal{S}) = \mathcal{L}_{\text{CE}}\left(\mathcal{S}, k'_t\right) \cdot \mathcal{W}_{k'_t}. \tag{9}$$

Knowledge Distillation: We further alleviate the noisy labeling problem of distance supervision via Knowledge Distillation (KD) as shown in Figure 1 (b). Following (Tian et al., 2020; Chen et al., 2020b), the teacher takes the context and response as input and generates the distribution of knowledge selection as soft target. Compared with the student, i.e., the standard knowledge selection module described in Section 2.4, teacher has the gold response as an additional input.

Specifically, we make up the teacher query as $\mathbf{q}_t^{\text{tea}} = \mathbf{W}_{\text{tea}}\left[\mathbf{s}_{dh_t}; \mathbf{s}_{kh_{t-1}}\right] \in \mathbb{R}^d$, which contains more information (i.e., the response) than $\mathbf{q}_t$ in Equation 3. Then we use this query $\mathbf{q}_t^{\text{tea}}$ to obtain the teacher's knowledge selection distribution $\mathcal{T} \in \mathbb{R}^L$ by Equation 4. Finally, online response-base knowledge distillation[3] is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{KD}} &= \mathcal{L}_{\text{WCE}}(\mathcal{T}) + D_{\text{KL}}(\mathcal{T} \,\|\, \mathcal{S}) \\ &= \mathcal{L}_{\text{CE}}(\mathcal{T}, k'_t) \cdot \mathcal{W}_{k'_t} + \sum_{k_t^i \in \mathcal{K}_t} \mathcal{T}_{k_t^i} \log \frac{\mathcal{T}_{k_t^i}}{\mathcal{S}_{k_t^i}},\end{aligned} \tag{10}$$

where the first term is used for teacher training and the second term transfers the knowledge from teacher to student based on the Kullback-Leibler (KL) divergence between the teacher and student distributions (i.e., $\mathcal{T} \in \mathbb{R}^L$ and $\mathcal{S} \in \mathbb{R}^L$).

Although the teacher is also trained with the noisy label, the teacher produces an independent source of variance that can be used to cancel out the variance introduced by the label noise (Li et al., 2017). Moreover, previous works have proved that the student can still be enhanced by the noisy teacher (Sau and Balasubramanian, 2016; Xie et al., 2020; Yuan et al., 2020). Therefore, we believe that the student trained by two supervision signals gets benefits from the regularization of the soft target (Yuan et al., 2020).

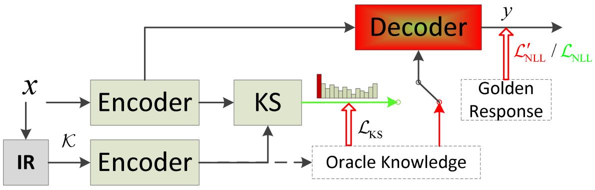

Figure 2: To deal with the mismatch problem, we first pretrain the Knowledge Selection (KS) and the decoder with the matched knowledge[4] in parallel along red lines. Then we finetune the decoder with the selected knowledge weightedly along the green line.

To sum up, our DDSL loss is as follows:

$$\mathcal{L}_{\text{KS}} = \mathcal{L}_{\text{DDSL}} = \mathcal{L}_{\text{WCE}}(\mathcal{S}) + \mathcal{L}_{\text{KD}}, \tag{11}$$

which consists of distant supervision, label weighting in Equation 9 and knowledge distillation in Equation 10 for unsupervised knowledge selection.

2.7 Training

The Mismatch Knowledge Selection Problem: As we know, there are chances that the selected knowledge is not the gold one due to 1) the diversity of knowledge selection in conversation (Kim et al., 2020a) and 2) the under-optimized knowledge selection at early training stage (Zhao et al., 2020b). And it is more serious in the unsupervised setting where it is hard to train knowledge selection well. The mismatch knowledge selection problem occurs due to the training paradigm in Figure 1 where the decoder is trained to generate the gold response with mismatched knowledge. This mismatch problem causes the knowledge-aware decoder to take the selected knowledge as noise and degenerate into the knowledge-independent decoder.

Our Pretraining-Finetuning Strategy: Although training the decoder with the matched knowledge[4] solves the mismatch problem. It can't deal with wrong knowledge selection at inference which is often the case, yet never seen at training.

---

[3] As surveyed in (Gou et al., 2020), response-based knowledge usually refers to the neural response of the last output layer of the teacher model, and online distillation means we train the teacher and student together.

[4] Note that the oracle knowledge is most literally matched with the gold response by the metric defined in Equation 8.

We take the plain idea as our pretraining stage, and then train the decoder to adapt to the selected knowledge using different weighting scores in the finetuning stage .

In the pretraining stage, we train knowledge selection and response generation in parallel as Figure 2 shows. The pretraining loss is as follows:

$$\begin{aligned}\mathcal{L}'_{\text{NLL}} &= -\log p\left(y_t | x_t, k'_t\right) \\ \mathcal{L} &= \mathcal{L}_{\text{KS}} + \mathcal{L}'_{\text{NLL}}\end{aligned}, \qquad (12)$$

where we use $\mathcal{L}_{\text{KS}} = \mathcal{L}_{\text{DDSL}}$ in the unsupervised setting and the decoder is trained to generate the gold response with the oracle knowledge k'_t instead of the selected one k^s_t. Therefore, the decoder learns how to incorporate the matched knowledge[4] k'_t into the response generation because k'_t is much more accurate than the selected one k^s_t as Table 4 shows. And we could alleviate the mismatch problem from the pretraining process because 1) we get a fully optimized knowledge selection module and 2) the pretrained decoder provides a good initialization for finetuning.

In the finetuning stage, we continuely train the pretrained decoder to adapt to the pretrained knowledge selection module with the sample weighting idea. And the finetuning loss is defined as follows:

$$\mathcal{L}_{\text{G}} = \mathcal{L}_{\text{NLL}} \cdot \left(1 + \mathcal{W}_{k^s_t}\right) \quad , \qquad (13)$$

where $\mathcal{L}_{\text{NLL}}$ is defined in Equation 7 and $\mathcal{W}_{k^s_t}$ is the confidence or weighting score of the selected knowledge k^s_t defined in Equation 8. As mentioned above, the mismatch knowledge selection problem is caused by the training paradigm that we may train the decoder to generate the gold response with the mismatched knowledge from the knowledge selection module. Here, we finetune the pretrained decoder with the selected knowledge k^s_t by giving higher importance weights if the selected knowledge is suitable for the gold response generation. In this way, we further alleviate the mismatch problem because we highlight the matched samples by assigning an importance weight to each instance (x_t, k^s_t, y_t) to reform the training data (Cai et al., 2020; Dong et al., 2020).

3 Experiments

3.1 Dataset

We evaluate our model on two public knowledge-grounded dialogue datasets: Wizard of Wikipedia (WoW) (Dinan et al., 2019) and Holl-E (Moghe et al., 2018), both of which provide the knowledge candidates with gold knowledge labels for knowledge selection. To test our approach in the unsupervised setting, we do not use the gold knowledge label provided in those datasets.

WoW contains the dialogues between two participants on some open-domain topics, where one is a curious learner while the other plays the role of a knowledgeable expert with access to the knowledge pool. Each knowledge pool contains about 61 sentences on average per turn, which are retrieved from Wikipedia based on the dialogue context via the IR system. There are 18430, 1948 and 1933 dialogues for training, validation and test, respectively. According to the topic overlapping, the test set is split into two subsets: 965 Test Seen and 968 Test Unseen dialogues, where Test Unseen consists of 58 topics never seen in train or validation.

Holl-E contains 7228, 930 and 913 dialogues for training, validation and test, respectively. Two test versions are provided: one with a single gold reference, the other with multiple gold references (more than one gold knowledge sentences and corresponding responses for each given conversation context). Each dialogue is assigned with a document of about 60 sentences on average as the knowledge pool. Here, we use the modified version (Kim et al., 2020a) which fits for knowledge selection.

3.2 Models for Comparison

We compare our method with a set of baselines:[5]

3.2.1 No Knowledge

S2S_{Transformer} is a Seq2Seq model based on Transformer (Vaswani et al., 2017) that does not leverage the external knowledge.

S2S_{BERT} replaces the Transformer Encoder with a pretrained BERT (Devlin et al., 2019).

3.2.2 Supervised Knowledge Selection

TMN is short for End-to-End Transformer MemNet (Dinan et al., 2019), which selects knowledge based on the Transformer memory network and generate responses via the Transformer decoder.

TMN_{BERT+PostKS+CP}, implemented by (Kim et al., 2020a), enhances the encoder with BERT, knowledge selection module with PostKS (Lian et al., 2019) and decoder with the copy mechanism (CP).

---

[5] See Appendix A.1 for more baselines, from knowledge-aware generation (Huang et al., 2020) to knowledge selection in the supervised, semi-supervised and unsupervised settings.

Setting	Row	Method	Test Seen				Test Unseen			
			Acc ↑	PPL ↓	R-1 ↑	R-2 ↑	Acc ↑	PPL ↓	R-1 ↑	R-2 ↑
0 No Knowledge	0_0	$\text{S2S}^{\dagger}_{\text{Transformer}}$ (Vaswani et al., 2017)	N/A	57.4	16.5	4.4	N/A	102.0	14.6	2.9
	0_1	$\text{S2S}^{\dagger}_{\text{BERT}}$	N/A	53.9	17.2	4.8	N/A	93.3	14.9	3.3
1 Supervised	1_0	TMN (Dinan et al., 2019)	21.1	63.5	16.9	N/A	14.3	97.3	14.4	N/A
	1_1	$\text{TMN}_{\text{BERT+PostKS+CP}}$ (2020a)	25.5	52.2	19.0	6.5	14.4	83.4	15.6	3.9
	1_2	SKT (Kim et al., 2020a)	**26.8**	52.0	19.3	6.8	18.3	81.4	16.1	4.2
2 Unsupervised	2_0	TMN_0 (Dinan et al., 2019)	13.4	66.5	15.9	N/A	11.8	103.6	14.3	N/A
	2_1	PostKS (Lian et al., 2019)	4.8	79.1	13.0	1.0	4.2	193.8	13.1	1.0
	2_2	SKT_0 (Kim et al., 2020a)	0.3	54.7	17.1	4.6	0.1	88.2	15.5	3.4
	0	UKSDG w/o DDSL	5.2	49.3	17.4	5.1	5.1	82.8	15.1	3.3
	1	$\text{UKSDG}_{\text{vec}}$ w/o DDSL	13.5	46.1	17.8	5.2	13.1	82.3	15.5	3.3
	2	UKSDG (ours)	23.8	51.8	19.5	6.8	16.2	76.3	16.3	4.4
	3	UKSDG_{PF} w/o SW (ours)	26.4	**44.1**	20.3	7.4	**20.8**	**64.5**	17.8	5.6
	4	UKSDG_{PF} (ours)	26.4	45.0	**20.6**	**7.7**	**20.8**	65.5	**18.2**	**5.9**

Table 1: Quantitative results on WoW. Our approach manages to select knowledge more accurately in the unsupervised setting and generate more informative responses than the strong baselines in the supervised setting. Note that models with "†" are implemented by ourselves and other models with citation are from the original paper.

SKT (Kim et al., 2020a), short for Sequential Knowledge Transformer, uses the posterior distribution by sequential latent modeling and achieves promising results in the supervised setting.

3.2.3 Unsupervised Knowledge Selection

TMN_0 is **TMN**, trained only via generation loss in Equation 7 without knowledge loss in Equation 5. **SKT_0** is SKT optimized without knowledge loss. **PostKS** (Lian et al., 2019) takes the benefits of latent variable models and leverages the posterior knowledge distribution as a pseudo label for knowledge selection without knowledge loss. Here, we use the results provided by (Kim et al., 2020a).

3.2.4 Our Unsupervised Methods

We implement our model in the unsupervised setting, namely Unsupervised Knowledge Selection for Dialogue Generation (**UKSDG**), which is optimized with our DDSL in Equation 11 for unsupervised knowledge selection and generation loss in Equation 7. **UKSDG_{PF}** indicates that we adopt our Pretraining-Finetuning strategy to alleviate the mismatch knowledge selection problem in Section 2.7. Furthermore, we remove several components for ablation study:

(1) **UKSDG w/o DDSL** is only optimized by the generation loss in Equation 7 without our DDSL.

(2) **$\text{UKSDG}_{\text{vec}}$ w/o DDSL** further replaces the decoder input $\mathbf{H}_{rc}$ (in Section 2.5) with the averaged knowledge vector enhanced context $\mathbf{H}_{vec} = \mathbf{H}_{x_t} + \sum_{k_t^i \in \mathcal{K}_t} \mathcal{S}_{k_t^i} \cdot \mathbf{h}_{k_t^i}$.

(3) **UKSDG_{PF} w/o SW** does not use the Sample Weighting in Equation 13.

3.3 Implementation Details

We use TensorFlow 2.0 to implement our approach base on SKT[6]. All sentences are encoded by the shared $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019), and the response is greedily generated via a 5-layer Transformer decoder with copy mechanism. The hidden size d is 768 and the vocabulary size $|V|$ is $30,522$. The knowledge selection module contains two separate one-layer GRUs and one projection layer. Our proposed DDSL contains no trainable parameters except one projection layer in the teacher selection module. And the temperature τ is 0.1.

We use the Adam optimizer (Kingma and Ba, 2014) with gradient clipping at 0.4 to train our models on a single GPU (TITAN Xp). The learning rate is $2e^{-5}$ and the batch[7] size is 1. Moreover, we apply label smoothing (Pereyra et al., 2017) and set 0.1 for knowledge selection and 0.05 for response generation. In the pretraining-finetuning strategy, we use 5 and 20 epochs in the pretraining and finetuning stage, respectively. The pretrained models are selected according to the accuracy score and other models are selected according to the R-1 score since knowledge selection aims to serve for high-quality generation.

It takes almost the same time for the convergence of UKSDG and KSDG as we only replace our DDSL with the gold knowledge selection loss. The convergence of SKT is a bit slower as it is hard

---

[6] Thanks for their processed datasets, models and evaluation codes at `https://github.com/bckim92/sequential-knowledge-transformer`.

[7] Each example is a dialogue rather than an individual turn.

Setting	Row	Method	Single Reference				Multi Reference			
			Acc ↑	PPL ↓	R-1 ↑	R-2 ↑	Acc ↑	PPL ↓	R-1 ↑	R-2 ↑
0 No Knowledge	0_0	S2S$^{\dagger}_{\text{Transformer}}$ (Vaswani et al., 2017)	N/A	105.8	19.1	9.8	N/A	72.4	24.1	12.9
	0_1	S2S$^{\dagger}_{\text{BERT}}$	N/A	89.4	19.0	9.6	N/A	61.9	23.5	12.1
1 Supervised	1_0	TMN (Dinan et al., 2019)	22.7	140.6	20.1	10.3	32.3	83.6	24.3	12.8
	1_1	TMN$_{\text{BERT+PostKS+CP}}$(2020a)	27.8	47.4	29.2	22.3	37.8	27.9	35.9	29.0
	1_2	SKT (Kim et al., 2020a)	**29.2**	48.9	29.8	23.1	**39.2**	28.5	36.5	29.7
2 Unsupervised	2_1	PostKS (Lian et al., 2019)	1.5	196.6	15.2	6.0	3.2	114.1	19.2	7.9
	2_2	SKT$_0^{\ddagger}$ (Kim et al., 2020a)	0.1	77.1	19.2	9.9	0.1	75.8	19.4	10.1
	0	UKSDG w/o DDSL	2.9	76.5	20.0	10.5	4.3	52.0	25.0	13.6
	1	UKSDG$_{\text{vec}}$ w/o DDSL	19.1	78.1	20.9	10.9	28.8	53.2	25.4	13.3
	2	UKSDG (ours)	24.2	55.4	29.7	23.1	33.5	31.9	36.4	29.7
	3	UKSDGPF w/o SW (ours)	25.1	**38.7**	30.8	23.2	35.0	**22.7**	37.9	30.0
	4	UKSDG$_{\text{PF}}$ (ours)	25.1	39.4	**31.0**	**24.0**	35.0	22.9	**38.1**	**31.1**

Table 2: Quantitative results on Holl-E. The method with "‡" is reported by rerunning the released code, and models with "†" are implemented by ourselves.

to optimize the latent variable model. It takes about 2.5 times as long for the convergence of UKSDG$_{\text{PF}}$ and KSDG$_{\text{PF}}$. The computation for confidence calculation during inference is the same for UKSDG, KSDG (w/ and w/o PF) and SKT because we adopt the same backbone. All of our models contain about $174M$ parameters.

3.4 Evaluation

Automatic Evaluation. We automatically evaluate knowledge selection with accuracy (Acc), response generation with perplexity (PPL), unigram F1 (R-1) and bigram F1 (R-2), which are commonly used in this task (Dinan et al., 2019; Kim et al., 2020a; Chen et al., 2020b). We also remove all the punctuation and (a, an, the) to compute the R-1 and R-2 scores as (Kim et al., 2020a) do. Note that lower PPL and higher R-1 and R-2 scores indicate better generation quality.

Human Evaluation. We firstly select 100 samples from each test set on WoW for human evaluation. Then we follow (Kim et al., 2020a; Li et al., 2019a) and ask three annotators to evaluate the generation quality according to Engagingness and Knowledgeability from 1 to 4, where 1 means not at all, 2 is a little, 3 is somewhat, and 4 is a lot. Engagingness measures how much do you like the response and Knowledgeability measures the informativeness in the responses.

4 Results and Analysis

4.1 Main Results

Quantitative Results. We report the automatic results on WoW and Holl-E in Table 1 and Table 2, respectively. And we have the following consistent observations:[8] (1) Comparing SKT and SKT$_0$, we see that the gold knowledge loss plays an important role to train knowledge selection well. (2) Comparing row 0 and 2, we see that our proposed DDSL is the key to train knowledge selection well in the unsupervised setting and can be the alternative of the gold knowledge loss. As a result, our approach significantly outperforms the other unsupervised methods on all metrics (significance tests (Koehn, 2004), p-value < 0.01). (3) Although UKSDG$_{\text{vec}}$ w/o DDSL could learn some patterns of knowledge selection from the gradient of generation loss, we see that compressing the knowledge into a vector will loss much information for dialogue generation. (4) Although our UKSDG$_{\text{PF}}$ usually makes knowledge selection worse than the supervised SKT in row 1_2, we achieve higher generation quality, which indicates that our pretraining-finetuning strategy could alleviate the mismatch knowledge selection problem and emphasizes the importance of leveraging the selected knowledge properly for future study on this task. (5) Comparing row 3 and 4, we see that the sample weighting also helps in the finetuning stage, though PPL score is slightly worse due to the difficulty of injecting the knowledge into responses. (6) Moreover, our approach demonstrates the stronger ability of generalization with smaller performance gap between Test Seen and Test Unseen in Table 1, which we attribute to our DDSL because it works in the unsupervised setting and is more suitable for Test Unseen. For example, compared with SKT in row 1_2, UKSDG$_{\text{PF}}$ in row

---

[8] More observations with other baselines can be found in Appendix A.2.

Method	Test Seen		Test Unseen	
	EGG	**KLD**	**EGG**	**KLD**
SKT	2.70±0.05	2.64±0.05	2.46±0.05	2.50±0.06
UKSDG (ours)	2.49±0.05	2.37±0.06	2.33±0.06	2.27±0.05
UKSDG$_{\mathrm{PF}}$ (ours)	2.72±0.05	2.71±0.05	2.49±0.06	2.60±0.05
Gold Response	3.25±0.04	3.37±0.05	3.16±0.05	3.24±0.05

Table 3: Results of human evaluations on WoW. **EGG** and **KLD** denote Engagingness and Knowledgeability, respectively.

Method	WoW		Holl E	
	Seen	Unseen	Single	Multi
Gold label (k_t)	100.0	100.0	100.0	100.0
Oracle Label (k'_t)	70.1	69.0	90.3	90.4
$\mathcal{L}_{\mathrm{DDSL}}$ in Eq. 11	26.4	20.8	25.1	35.0
w/o $\mathcal{L}_{\mathrm{KD}}$ in Eq. 10	25.8	19.1	24.1	33.5
w/o $\mathcal{L}_{\mathrm{KD}}$ & $\mathcal{W}_{k'_t}$ in Eq. 9	25.5	18.8	24.0	34.2

Table 4: Ablation study. Knowledge selection accuracy of UKSDG$_{\mathrm{PF}}$ trained with different losses.

4 achieves the highest selection accuracy in Test Unseen with a slightly lower accuracy in Test Seen.

Qualitative Results. We report human evaluation results of the generated responses according to Engagingness and Knowledgeability in Table 3. Comparing UKSDG and UKSDG$_{\mathrm{PF}}$, we see the effectiveness of our pretraining-finetuning strategy which could alleviate the mismatch problem described in Section 2.7. Our UKSDG$_{\mathrm{PF}}$ in the unsupervised setting generates responses slightly better than SKT in the supervised setting. Moreover, the improvement is much obvious according to Knowledgeability, which also indicates the importance of using the selected knowledge properly.

4.2 Ablation Study

We have introduced the components of our DDSL in Section 2.6 and shown its effectiveness in Section 4.1. Here, we analyse our DDSL via an ablation study in Table 4. Actually, we train UKSDG$_{\mathrm{PF}}$ on knowledge selection task, using our DDSL with components removed. We have the following observations: (1) We see that most of the oracle knowledge from distant supervision is the same as the gold knowledge label by human. Therefore, it is acceptable to directly use this oracle knowledge label to substitute the gold label as the supervision signal (see the last row of Table 4). (2) However, distant supervision inevitably suffers from the noisy labeling problem due to literally matching. For example, there are about 30% oracle knowledges different from the gold ones on WoW where the responses convey knowledge much more flexibly. (3) Loss weighting in Equation 9 helps on WoW where the noisy labeling problem is serious. (4) Knowledge distillation in Equation 10 could further alleviate the label noise, and our method manages to select knowledge accurately in the unsupervised setting.

4.3 Case Study

Table 5 presents several examples of generated responses on WoW and we have the following observations: (1) The oracle knowledge from distant supervision contains several informative tokens which could help dialogue generation since they appears in the gold response as defined by Equation 8. In particular, the oracle knowledge in the case 3 is also appropriate and the gold response contains much more information than the gold knowledge, which indicates the diversity of knowledge selection and selecting one sentence may be not enough for informative generation. (2) However, some oracle knowledge labels are different from the gold ones (i.e., the selected one by human). Our model still learns to select knowledge as human, which we attribute to our DDSL since knowledge distillation alleviates the noisy labeling problem of distant supervision. (3) SKT does not leverage the selected knowledge well and generates the dull response. For example, SKT repeats the context in case 1, generates the verbose and contradictory response in case 2 and does not provide new information in case 3. (4) Whereas, our UKSDG$_{\mathrm{PF}}$ firstly selects the appropriate knowledge as human does, and then generate fluent and informative responses by alleviate the mismatch knowledge selection problem with the help of the pretraining-finetuning strategy. This indicates the importance of leveraging the selected knowledge properly for future study.

5 Related Work

External knowledge has been wildly explored to enhance dialogue understanding and/or improve dialogue generation (Zhu et al., 2017; Liu et al., 2018; Chen et al., 2019; Wu et al., 2019; Chen et al., 2020a; Tuan et al., 2020; Sun et al., 2020; Zhang et al., 2020; Yu et al., 2020; Ni et al., 2021). To make use of knowledge, knowledge access is very important. Therefore, knowledge selection which selecting appropriate knowledge given the dialog context gains much attention (Zhang et al., 2019; Meng et al., 2019; Ren et al., 2019; Dinan

Case 1	**Context**		Are you a fan of elvis presley?
	Human	KS	Regarded as one of the most significant cultural icons of the 20th century, he is often referred to as the “king of rock and ...”
		DG	You mean the king of **rock and roll**. Actually yes I am. best of all time. Don’t you agree?
	SKT	KS	Elvis Aaron Presley (January 8, 1935 - August 16, 1977) was an American singer, musician, and actor.
		RG	Yes, I love Elvis Presley!
	UKSDG$_{\text{PF}}$	OK	Thornton’s recording of “hound dog” is listed as one of the **rock and roll** hall of fame’s
		KS	Regarded as one of the most significant cultural icons of the 20th century, he is often referred to as the “king of rock and ...”
		DG	Yes, I love him! He is regarded as the king of rock and roll.
Case 2	**Context**		I love to go bowling with my family, but i’m a horrible bowler. Do you like it?
	Human	KS	Bowling is a sport or leisure activity in which a player rolls or throws a bowling ball towards a target.
		DG	Oh, yes, I love **bowling**. Rolling that ball down **the lane and knocking** down the **pins** gives me a charge.
	SKT	KS	It is one of the major forms of throwing sports.
		RG	I do! I love bowling! I like to play, but i’m not a fan of bowling.
	UKSDG$_{\text{PF}}$	OK	In pin **bowling** variations, the target is usually to **knock** over **pins** at the end of a **lane**.
		KS	Bowling is a sport or leisure activity in which a player rolls or throws a bowling ball towards a target.
		DG	I love bowling! It is a sport or leisure activity in which a player rolls a bowling ball towards a target.
Case 3	**Context**		Have you ever seen the Walking Dead?
	Human	KS	The Walking Dead is an American post-apocalyptic horror television series developed by Frank Darabont for AMC that ...
		DG	The American post-apocalyptic series? The one **based on the comic book series by Robert Kirkman**? Yes I love it! How about you?
	SKT	KS	The Walking Dead is an American post-apocalyptic horror television series developed by Frank Darabont for AMC that ...
		RG	Yes, I have seen the Walking Dead.
	UKSDG$_{\text{PF}}$	OK	It is **based on the comic book series** “The Walking Dead” **by Robert Kirkman**, ...
		KS	The Walking Dead is an American post-apocalyptic horror television series developed by Frank Darabont for AMC that ...
		DG	Yes, I have seen the Walking Dead. It was developed by Frank Darabont for AMC.

Table 5: Examples of generated responses on WoW. KS, DG and OK denote Knowledge Selection, Dialogue Generation and Oracle Knowledge, respectively.

et al., 2019; Kim et al., 2020a; Meng et al., 2020; Chen et al., 2020b; Zheng et al., 2020; Huang et al., 2020). In this paper, we focus on knowledge selection in the unsupervised setting where there is no gold knowledge label. Lian et al. (2019) and Kim et al. (2020a) attempt to deal with this problem via latent models, yet their performance is less than satisfactory. Differently, we design our DDSL to make knowledge selection work well in the unsupervised setting. There is a very recent work (Zhao et al., 2020b) that finetunes GPT-2 (Radford et al., 2019) with the unsupervised pretrained knowledge selection module on unlabeled dialogues. We are different in two aspects: (1) Our DDSL leverages knowledge distillation to alleviate the label noise at the pretraining stage; (2) We adopt the sample weighting idea at the finetuning stage. And we will leverage GPT-2 for future study.

Our work is inspired by Distant Supervision (DS), an effective method to generate labeled data with external knowledge base (KB) for information extraction (Mintz et al., 2009; Min et al., 2013; Zeng et al., 2015; Wang et al., 2018). Following this idea, Gopalakrishnan et al. (2019) use the oracle knowledge from DS to construct the Topical-Chat dataset. Similarly, Qin et al. (2019b) obtain the weakly labeled data to train a KB retriever in the task-oriented dialogue system. Ren et al. (2019) propose a distantly supervised learning schema at segment level to effectively learn the topic transition vector. Although inspired by the similar idea, we are devoted to knowledge selection in the unsupervised setting, which is a different application of DS. Moreover, rather than just using distant supervision, we design our DDSL with label weighting and knowledge distillation to deal with the noisy labeling problem from DS.

6 Conclusion

We study unsupervised knowledge selection for dialogue generation where the gold knowledge label is not available. Actually, we design the Distilled Distant Supervision Loss, a novel and effective solution to train knowledge selection well in the unsupervised setting. Furthermore, we propose the pretraining-finetuning strategy to deal with the mismatch knowledge selection problem that models tend to select the mismatched knowledge for dialogue generation in the unsupervised setting and will cause the degeneration of knowledge-aware decoder. Experiments show that our approach manages to select knowledge more accurately in the unsupervised setting and generates more informative responses than many strong supervised baselines.

References

Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. 2020. Data Manipulation: Towards Effective Instance Learning for Neural Dialogue Generation via Learning to Augment and Reweight. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6334–6343, Online. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder. *arXiv preprint arXiv:1803.11175*.

Feilong Chen, Fandong Meng, Jiaming Xu, Peng Li, Bo Xu, and Jie Zhou. 2020a. DMRM: A dual-channel multi-hop reasoning model for visual dialog. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7504–7511. AAAI Press.

Xiuyi Chen, Fandong Meng, Peng Li, Feilong Chen, Bo Xu, Shuang Xu, and Jie Zhou. 2020b. Bridging the Gap between Prior and Posterior Knowledge Selection for Knowledge-Grounded Dialogue Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Xiuyi Chen, Jiaming Xu, and Bo Xu. 2019. A Working Memory Model for Task-oriented Dialog Response Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2687–2693, Florence, Italy. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *International Conference on Learning Representations*.

Jiahua Dong, Yang Cong, Gan Sun, Bineng Zhong, and Xiaowei Xu. 2020. What can be transferred: Unsupervised domain adaptation for endoscopic lesions segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4022–4031.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. *Proc. Interspeech 2019*, pages 1891–1895.

Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. 2020. Knowledge Distillation: A Survey. *CoRR*, abs/2006.05525.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in Building Intelligent Open-Domain Dialog Systems. *ACM Trans. Inf. Syst.*, 38(3).

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020a. Sequential Latent Knowledge Selection for Knowledge-Grounded Dialogue. *arXiv: Computation and Language*.

Seokhwan Kim, Mihail Eric, Karthik Gopalakrishnan, Behnam Hedayatnia, Yang Liu, and Dilek Hakkani-Tur. 2020b. Beyond Domain APIs: Task-oriented Conversational Modeling with Unstructured Knowledge Access. *arXiv preprint arXiv:2006.03533*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Linxiao Li, Can Xu, Wei Wu, Yufan Zhao, Xueliang Zhao, and Chongyang Tao. 2020. Zero-Resource Knowledge-Grounded Dialogue Generation. *CoRR*, abs/2008.12918.

Margaret Li, Jason Weston, and Stephen Roller. 2019a. ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-turn Comparisons. *arXiv preprint arXiv:1909.03087*.

Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. 2017. Learning from Noisy Labels with Distillation. *2017 IEEE International Conference on Computer Vision (ICCV)*.

Zekang Li, Cheng Niu, Fandong Meng, Yang Feng, Qian Li, and Jie Zhou. 2019b. Incremental Transformer with Deliberation Decoder for Document Grounded Conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 12–21.

Rongzhong Lian, Min Xie, Fan Wang, Jinhua Peng, and Hua Wu. 2019. Learning to Select Knowledge for Response Generation in Dialog Systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5081–5087. AAAI Press.

Xiexiong Lin, Weiyu Jian, Jianshan He, Taifeng Wang, and Wei Chu. 2020. Generating Informative Conversational Response using Recurrent Knowledge-Interaction and Knowledge-Copy. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 41–52, Online. Association for Computational Linguistics.

Shuman Liu, Hongshen Chen, Zhaochun Ren, Yang Feng, Qun Liu, and Dawei Yin. 2018. Knowledge Diffusion for Neural Dialogue Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1498, Melbourne, Australia. Association for Computational Linguistics.

Chuan Meng, Pengjie Ren, Zhumin Chen, Christof Monz, Jun Ma, and Maarten de Rijke. 2019. RefNet: A Reference-aware Network for Background Based Conversation. *arXiv preprint arXiv:1908.06449*.

Chuan Meng, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tengxiao Xi, and Maarten de Rijke. 2021. Initiative-Aware Self-Supervised learning for Knowledge-Grounded Conversations. In *Proceedings of the 44rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21. Association for Computing Machinery.

Chuan Meng, Pengjie Ren, Zhumin Chen, Weiwei Sun, Zhaochun Ren, Zhaopeng Tu, and Maarten de Rijke. 2020. DukeNet: A Dual Knowledge Interaction Network for Knowledge-Grounded Conversation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 1151–1160, New York, NY, USA. Association for Computing Machinery.

Xiang Gao Bill Dolan Jianfeng Gao Michel Galley, Chris Brockett. 2018. Grounded Response Generation Task at DSTC7. *Proceedings of the AAAI-19 Workshop on Dialog System Technology Challenges*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant Supervision for Relation Extraction with an Incomplete Knowledge Base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, Atlanta, Georgia. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards Exploiting Background Knowledge for Building Conversation Systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, Vinay Adiga, and Erik Cambria. 2021. Recent advances in deep learning-based dialogue systems. *CoRR*, abs/2105.04387.

Zheng-Yu Niu, Hua Wu, Haifeng Wang, et al. 2019. Knowledge Aware Conversation Generation with Explainable Reasoning over Augmented Graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1782–1792.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. *arXiv preprint arXiv:1701.06548*.

Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, Bill Dolan, Yejin Choi, and Jianfeng Gao. 2019a. Conversing by Reading: Contentful Neural Conversation with On-demand Machine Reading. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5427–5436.

Libo Qin, Yijia Liu, Wanxiang Che, Haoyang Wen, Yangming Li, and Ting Liu. 2019b. Entity-Consistent End-to-end Task-Oriented Dialogue System with KB Retriever. In *Proceedings of the*

2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 133–142, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Pengjie Ren, Zhumin Chen, Christof Monz, Jun Ma, and Maarten de Rijke. 2019. Thinking Globally, Acting Locally: Distantly Supervised Global-to-Local Knowledge Selection for Background Based Conversation. *arXiv preprint arXiv:1908.09528*.

Bharat Bhusan Sau and Vineeth N Balasubramanian. 2016. Deep Model Compression: Distilling Knowledge from Noisy Teachers. *arXiv preprint arXiv:1610.09650*.

C. Selltiz, L.S. Wrightsman, S.W. Cook, and Society for the Psychological Study of Social Issues. 1976. *Research Methods in Social Relations*. Holt, Rinehart and Winston.

Yajing Sun, Yue Hu, Luxi Xing, Jing Yu, and Yuqiang Xie. 2020. History-adaption Knowledge Incorporation Mechanism for Multi-turn Dialogue System. In *AAAI*.

Xiangru Tang and Po Hu. 2019. Knowledge-Aware Self-Attention Networks for Document Grounded Dialogue Generation. In *International Conference on Knowledge Science, Engineering and Management*, pages 400–411. Springer.

Zhiliang Tian, Wei Bi, Dongkyu Lee, Lanqing Xue, Yiping Song, Xiaojiang Liu, and Nevin L. Zhang. 2020. Response-Anticipated Memory for On-Demand Knowledge Integration in Response Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 650–659, Online. Association for Computational Linguistics.

Yi-Lin Tuan, Wei Wei, and William Yang Wang. 2020. Unsupervised Injection of Knowledge into Dialogue Generation via Language Models. *arXiv preprint arXiv:2004.14614*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018. Label-Free Distant Supervision for Relation Extraction via Knowledge Graph Embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2246–2255, Brussels, Belgium. Association for Computational Linguistics.

Wenquan Wu, Zhen Guo, Xiangyang Zhou, Hua Wu, Xiyuan Zhang, Rongzhong Lian, and Haifeng Wang. 2019. Proactive Human-Machine Conversation with Explicit Conversation Goal. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3794–3804, Florence, Italy. Association for Computational Linguistics.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation Networks: Sequence Generation Beyond One-Pass Decoding. In *Advances in Neural Information Processing Systems*, pages 1784–1794.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with Noisy Student improves ImageNet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698.

Yan Xu, Etsuko Ishii, Zihan Liu, Genta Indra Winata, Dan Su, Andrea Madotto, and Pascale Fung. 2021. Retrieval-free knowledge-grounded dialogue response generation with adapters. *CoRR*, abs/2105.06232.

Kaijia Yang, Liang He, Xin-yu Dai, Shujian Huang, and Jiajun Chen. 2019a. Exploiting Noisy Data in Distant Supervision Relation Classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3216–3225, Minneapolis, Minnesota. Association for Computational Linguistics.

Liu Yang, Junjie Hu, Minghui Qiu, Chen Qu, Jianfeng Gao, W. Bruce Croft, Xiaodong Liu, Yelong Shen, and Jingjing Liu. 2019b. A Hybrid Retrieval-Generation Neural Conversation Model. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 1341–1350. ACM.

Semih Yavuz, Abhinav Rastogi, Guan-Lin Chao, Dilek Hakkani-Tür, and Amazon Alexa AI. 2019. DEEPCOPY: Grounded Response Generation with Hierarchical Pointer Networks. In *20th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 122.

Hao-Tong Ye, Kai-Lin Lo, Shang-Yu Su, and Yun-Nung Chen. 2020. Knowledge-grounded response generation with deep attentional latent-variable model. *Computer Speech & Language*, page 101069.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A survey of knowledge-enhanced text generation. *CoRR*, abs/2010.04389.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. 2020. Revisiting Knowledge Distillation via Label Smoothing Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal. Association for Computational Linguistics.

Duzhen Zhang, Xiuyi Chen, Shuang Xu, and Bo Xu. 2020. Knowledge aware emotion recognition in textual conversations via multi-task incremental transformer. In *COLING*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Yangjun Zhang, Pengjie Ren, and Maarten de Rijke. 2019. Improving background based conversation with context-aware knowledge pre-selection. *arXiv preprint arXiv:1906.06685*.

Xueliang Zhao, Wei Wu, Chongyang Tao, Can Xu, Dongyan Zhao, and Rui Yan. 2020a. Low-Resource Knowledge-Grounded Dialogue Generation. In *International Conference on Learning Representations*.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020b. Knowledge-Grounded Dialogue Generation with Pre-trained Language Models. *arXiv preprint arXiv:2010.08824*.

Chujie Zheng, Yunbo Cao, Daxin Jiang, and Minlie Huang. 2020. Difference-aware Knowledge Selection for Knowledge-grounded Conversation Generation. *CoRR*, abs/2009.09378.

Wen Zheng and Ke Zhou. 2019. Enhancing Conversational Dialogue Models with Grounded Knowledge. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 709–718.

Hao Zhou, Chujie Zheng, Kaili Huang, Minlie Huang, and Xiaoyan Zhu. 2020. KdConv: A Chinese Multi-domain Dialogue Dataset Towards Multi-turn Knowledge-driven Conversation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7098–7108, Online. Association for Computational Linguistics.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018. A Dataset for Document Grounded Conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 708–713.

Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. 2017. Flexible End-to-End Dialogue System for Knowledge Grounded Conversation. *arXiv preprint arXiv:1709.04264*.

A Additional Experiments

A.1 Models for Comparison

A.1.1 Our Unsupervised Methods

We implement our model in the unsupervised setting, namely Unsupervised Knowledge Selection for Dialogue Generation (**UKSDG**), which is optimized with our DDSL in Equation 11 for unsupervised knowledge selection and generation loss in Equation 7. **UKSDG$_{\textbf{PF}}$** indicates that we adopt our Pretraining-Finetuning strategy to alleviate the mismatch knowledge selection problem in Section 2.7. Furthermore, we remove several components for ablation study:

(1) **UKSDG w/o DDSL** does not use our DDSL.

(2) **UKSDG$_{\textbf{vec}}$ w/o DDSL** further replaces the decoder input $\mathbf{H}_{rc}$ (in Section 2.5) with the averaged knowledge vector enhanced context $\mathbf{H}_{vec} = \mathbf{H}_{x_t} + \sum_{k_t^i \in \mathcal{K}_t} \mathcal{S}_{k_t^i} \cdot \mathbf{h}_{k_t^i}$.

(3) **UKSDG$_{\textbf{PF}}$ w/o SW** does not use the Sample Weighting in Equation 13.

A.1.2 Our Supervised Methods

Here, we also report the Knowledge Selection for Dialogue Generation (**KSDG**) in the supervised setting as described in Section 2.3, 2.4 and 2.5. And **KSDG$_{\textbf{PF}}$** alleviates the mismatch problem using the pretrain-finetuning strategy in Section 2.7. We use the gold knowledge loss in Equation 5 to train KSDG and KSDG$_{\text{PF}}$ in the supervised setting while we train UKSDG and UKSDG$_{\text{PF}}$ with our DDSL when the gold knowledge is not available.

We compare our method with a set of baselines:

A.1.3 No Knowledge

S2S$_{\textbf{Transformer}}$ is a Seq2Seq model based on Transformer (Vaswani et al., 2017) that does not leverage the external knowledge.

S2S$_{\textbf{BERT}}$ replaces the Transformer Encoder with a pre-trained BERT (Devlin et al., 2019).

KnowExpert (Xu et al., 2021) avoids the knowledge retrieval process and attempts to inject prior knowledge into the pre-trained language models for knowledge-grounded dialogue generation task. Essentially, KnowExpert stores knowledge in its parameters with lightweight adapters. Therefore, KnowExpert does not use knowledge explicitly at inference.

A.1.4 Supervised Knowledge Selection

TMN is short for End-to-End Transformer MemNet (Dinan et al., 2019), which selects knowledge based on the Transformer memory network and generate responses via the Transformer decoder.

TMN$_{\textbf{BERT+PostKS+CP}}$, implemented by (Kim et al., 2020a), enhances the encoder with the pretrained BERT, knowledge selection module with PostKS (Lian et al., 2019) and decoder with the copy mechanism (CP).

SKT (Kim et al., 2020a), short for Sequential Knowledge Transformer, uses the posterior distribution by sequential latent modeling and achieves promising results in the supervised setting.

SKT+PIPM+KDBTS (Chen et al., 2020b) equips SKT with Posterior Information Prediction Module (PIPM) and Knowledge Distillation Based Training Strategy (KDBTS) to bridge the gap between prior and posterior knowledge selection for knowledge-grounded dialogue generation.

A.1.5 Unsupervised Knowledge Selection

TMN$_0$ is **TMN**, trained only via generation loss in Equation 7 without knowledge loss in Equation 5.

SKT$_0$ is **SKT** optimized without knowledge loss.

PostKS (Lian et al., 2019) takes the benefits of latent variable models and leverages the posterior knowledge distribution as a pseudo label for knowledge selection without knowledge loss. Here, we use the results provided by (Kim et al., 2020a).

A.1.6 Knowledge-Aware Generation

Different from ours on the knowledge selection setting, there are much work, named knowledge-aware generation (Huang et al., 2020), which regard the retrieved knowledge pool as the pseudo pre-identified document for dialogue generation with complex knowledge injection mechanism. Nevertheless, we provide the results to get a comprehensive understanding where this field is going.

MTASK-RF (Ghazvininejad et al., 2018) is an early model that realizes knowledge-grounded conversation without crowd-sourced knowledge-grounded dialogues. Here we use the results provided by (Li et al., 2020).

ITDD is short for Incremental Transformer with Deliberation Decoder (Li et al., 2019b) where the encoder incrementally represents multi-turn dialogues and knowledge, and the decoder conducts response decoding in two passes.

DRD is short for Disentangle Response Decoder (Zhao et al., 2020a), a model that exploits pre-training techniques to tackle the low-resource challenge in knowledge-grounded dialogue generation. We choose the one whose parameters are fine-tuned on the full training data.

KIC (Lin et al., 2020) integrates recurrent Knowledge-Interaction and knowledge Copy (KIC) to generate informative responses.

ZRKGC (Li et al., 2020) is a very recent and unpublished work, which learns their model under the zero-resource setting, where the dialogue corpus and the knowledge corpus that are independent with each other.

A.2 Quantitative Results

Table 6 shows the automatic results in various settings on the Wizard of Wikipedia dataset, from which we have the following observations[9]: (1) The models with retrieved document generally achieves much lower PPL, still the R-1 is worse than our UKSDG$_{\text{PF}}$. For the worse R-1 score, we think the retrieved document contains the redundant and irrelevant information which are harmful for dialogue generation (Zhao et al., 2020b). Meanwhile, we attribute the better PPL score to the complex knowledge injection mechanism. For example, ITDD leverages the deliberation decoder (Xia et al., 2017) to improve context coherence and knowledge correctness and DRD devises a disentangled response decoder with pretrained language model, context processor and knowledge processor. (2) Comparing KSDG with KSDG$_{\text{PF}}$, again we see the generation quality improvement via the pretrain-finetuning strategy. Hence, our method is general in both supervised and unsupervised setting. (3) Comparing UKSDG$_{\text{vec}}$ w/o DDSL with UKSDG w/o DDSL, we see that although selecting the knowledge vector softly allows the gradient from dialogue generation to update knowledge selection directly, compressing the knowledge into a vector will loss much information for dialogue generation. This observation, combined with observations (1) and (2), together indicates the importance of leveraging the selected knowledge well for future study.

---

[9] The observations in the main paper will not be discussed again here.

Setting	Method	Test Seen				Test Unseen			
		Acc	PPL	R-1	R-2	Acc	PPL	R-1	R-2
No Knowledge	S2S$_{\text{Transformer}}$† (Vaswani et al., 2017)	N/A	41.8	17.8	N/A	N/A	87.0	14.0	N/A
	S2S$_{\text{BERT}}$†	N/A	55.2	17.0	4.7	N/A	93.3	15.0	3.1
	KnowExpert (Xu et al., 2021)	N/A	15.3	18.8	N/A	N/A	21.2	16.6	N/A
Retrieved Document	MTASK-RF (Ghazvininejad et al., 2018)	N/A	65.4	13.1	N/A	N/A	67.7	12.3	N/A
	ITDD (Li et al., 2019b)	N/A	17.8	16.2	N/A	N/A	44.8	11.4	N/A
	DRD (Zhao et al., 2020a)	N/A	23.0	18.0	N/A	N/A	25.6	16.5	N/A
	KIC (Lin et al., 2020)	N/A	51.9	18.4	N/A	N/A	65.8	17.3	N/A
	ZRKGC (Li et al., 2020)	N/A	41.1	18.9	N/A	N/A	42.7	18.8	N/A
Unsupervised KS	TMN$_0$ (Dinan et al., 2019)	13.4	66.5	15.9	N/A	11.8	103.6	14.3	N/A
	PostKS (Lian et al., 2019)	4.8	79.1	13.0	1.0	4.2	193.8	13.1	1.0
	SKT$_0$ (Kim et al., 2020a)	0.3	54.7	17.1	4.6	0.1	88.2	15.5	3.4
	UKSDG w/o DDSL	5.2	49.3	17.4	5.1	5.1	82.8	15.1	3.3
	UKSDG$_{\text{vec}}$ w/o DDSL	13.5	46.1	17.8	5.2	13.1	82.3	15.5	3.3
	UKSDG (ours)	23.8	51.8	19.5	6.8	16.2	76.3	16.3	4.4
	UKSDG$_{\text{PF}}$ w/o SW (ours)	26.4	44.1	20.3	7.4	20.8	64.5	17.8	5.6
	UKSDG$_{\text{PF}}$ (ours)	26.4	45.0	20.6	7.7	20.8	65.5	18.2	5.9
Semi-supervised KS	SKT$_{1/2}$ (Kim et al., 2020a)	25.1	49.0	19.2	6.6	16.7	77.8	16.1	4.1
	SKT$_{1/4}$ (Kim et al., 2020a)	22.4	45.7	18.7	6.1	13.8	78.0	15.8	3.6
	SKT$_{1/8}$ (Kim et al., 2020a)	21.0	45.3	18.6	6.0	12.3	79.9	15.7	3.6
Supervised KS	TMN (Dinan et al., 2019)	21.1	63.5	16.9	NA	14.3	97.3	14.4	NA
	TMN$_{\text{E2E+BERT+PostKS+CP}}$	25.5	52.2	19.0	6.5	14.4	83.4	15.6	3.9
	SKT (Kim et al., 2020a)	26.8	52.0	19.3	6.8	18.3	81.4	16.1	4.2
	SKT+PIPM+KDBTS (Chen et al., 2020b)	27.7	42.7	19.9	7.3	19.4	65.7	17.6	5.4
	KSDG (our)	25.1	53.0	18.9	6.5	17.3	80.8	16.3	4.5
	KSDG$_{\text{PF}}$ (our)	28.4	45.1	20.4	7.9	21.2	71.4	17.9	5.6

Table 6: Quantitative results on the Wizard of Wikipedia dataset. Note that models with "†" are implemented by ourselves and other models with citation are from the original paper. KS denotes knowledge selection.

Minimax and Neyman–Pearson Meta-Learning for Outlier Languages

Edoardo Maria Ponti[1,2*] **Rahul Aralikatte**[3*]
Disha Shrivastava[1,4] **Siva Reddy**[1,2] **Anders Søgaard**[3]
[1]Mila – Quebec Artificial Intelligence Institute [2]McGill University
[3]University of Copenhagen [4]University of Montreal
[1]`{edoardo-maria.ponti,siva.reddy,disha.shrivastava}@mila.quebec`
[3]`{rahul,soegaard}@di.ku.dk`

Abstract

Model-agnostic meta-learning (MAML) has been recently put forth as a strategy to learn resource-poor languages in a sample-efficient fashion. Nevertheless, the properties of these languages are often *not* well represented by those available during training. Hence, we argue that the *i.i.d.* assumption ingrained in MAML makes it ill-suited for cross-lingual NLP. In fact, under a decision-theoretic framework, MAML can be interpreted as minimising the expected risk across training languages (with a uniform prior), which is known as Bayes criterion. To increase its robustness to outlier languages, we create two variants of MAML based on alternative criteria: Minimax MAML reduces the *maximum* risk across languages, while Neyman–Pearson MAML *constrains* the risk in each language to a maximum threshold. Both criteria constitute fully differentiable two-player games. In light of this, we propose a new adaptive optimiser solving for a local approximation to their Nash equilibrium. We evaluate both model variants on two popular NLP tasks, part-of-speech tagging and question answering. We report gains for their average and minimum performance across low-resource languages in zero- and few-shot settings, compared to joint multi-source transfer and vanilla MAML. The code for our experiments is available at `https://github.com/rahular/robust-maml`.

1 Introduction

Knowledge transfer is ubiquitous in machine learning because of the general scarcity of annotated data (Pratt, 1993; Caruana, 1997; Ruder, 2019, *inter alia*). A prominent example thereof is transfer from resource-rich languages to resource-poor languages (Wu and Dredze, 2019; Ponti et al., 2019b; Ruder et al., 2019). Recently, Model-Agnostic Meta-Learning (MAML; Finn et al., 2017) has come to the fore as a promising paradigm: it explicitly trains neural models that adapt to new languages quickly by extrapolating from just a few annotated data points (Gu et al., 2018; Nooralahzadeh et al., 2020; Wu et al., 2020; Li et al., 2020).

MAML usually rests on the simplifying assumption that the source 'tasks' and the target 'tasks' are independent and identically distributed (henceforth, *i.i.d.*). However, in practice most scenarios of cross-lingual transfer violate this assumption: training languages documented in mainstream datasets do not reflect the cross-lingual variation, as they belong to a clique of few families, geographical areas, and typological features (Bender, 2009; Joshi et al., 2020). Therefore, the majority of the world's languages lies outside of such a clique. As training and evaluation languages differ in their joint distribution, they are not exchangeable (Ponti, 2021; Orbanz, 2012, ch. 6). Therefore, there is no formal guarantee that MAML generalises to the very languages whose need for transfer is most critical.

In this work, we interpret meta-learning within a decision-theoretic framework (Bickel and Doksum, 2015). MAML, we show, minimises the expected risk across languages found in the training distribution. Hence, it follows a so-called Bayes criterion. What if, instead, we formulated alternative criteria geared towards outlier languages? The first criterion we propose, Minimax MAML, is designed to be robust to worst-case-scenario out-of-distribution transfer: it minimises the *maximum* risk by learning an adversarial language distribution. The second criterion, Neyman–Pearson MAML, upper-bounds the risk for an arbitrary subset of languages via Lagrange multipliers, such that it does not exceed a predetermined threshold.

Crucially, both of these alternative criteria constitute competitive games between two players: one minimising the loss with respect to the neural pa-

---

[*] Equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1245–1260
August 1–6, 2021.

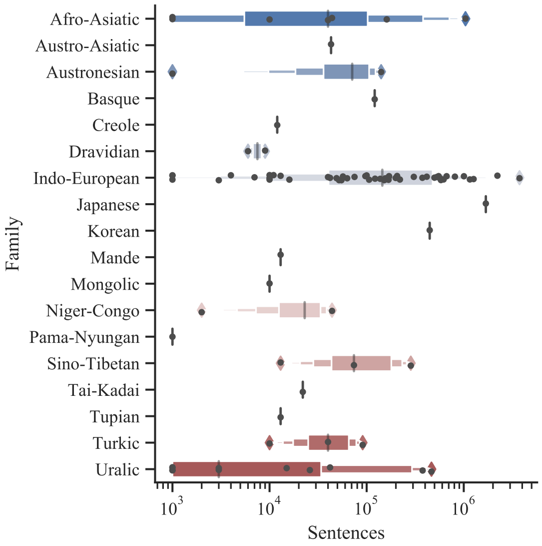

Figure 1: Annotated examples per family in the Universal Dependencies treebanks. Dots indicate individual languages, whereas boxes and whiskers mark quartiles.

rameters, the other maximising it with respect to the language distribution (Minimax MAML) or Lagrange multipliers (Neyman–Pearson MAML). Since an absolute Nash equilibrium may not exist for non-convex functions (Jin et al., 2020), such as neural networks, a common solution is to approximate local equilibria instead (Schäfer and Anandkumar, 2019). Therefore, we build on previously proposed optimisers (Balduzzi et al., 2018; Letcher et al., 2019; Gemp and Mahadevan, 2018) where players follow non-trivial strategies that take into account the opponent's predicted moves. In particular, we enhance them with first-order momentum and adaptive learning rate and apply them on our newly proposed criteria.

We run experiments on Universal Dependencies (Zeman et al., 2020) for part-of-speech (POS) tagging and TyDiQA (Clark et al., 2020) for question answering (QA). We perform knowledge transfer to 14 and 8 target languages, respectively, which belong to under-represented and often endangered families (such as Tupian from Southern America and Pama–Nyugan from Australia). We report modest but consistent gains for the average performance across languages in few-shot and zero-shot learning settings and mixed results for the minimum performance. In particular, Minimax and Neyman–Pearson MAML often surpass vanilla MAML and multi-source transfer baselines, which are currently considered state-of-the-art in these tasks (Wu and Dredze, 2019; Ponti et al., 2021; Clark et al., 2020).

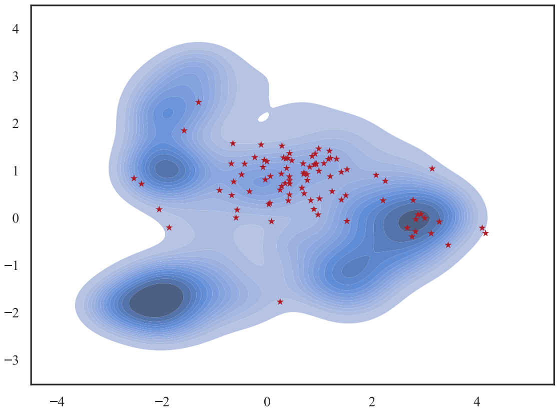

Figure 2: Density of WALS typological features of the world's languages reduced to 2 dimensions via PCA. Red dots are languages covered by UD. Darkness corresponds to more probable regions.

2 Skewed Language Distributions

Cross-lingual learning aims at transferring knowledge from resource-rich languages to resource-poor languages, to compensate for their deficiency of annotated data (Tiedemann, 2015; Ruder et al., 2019; Ponti et al., 2019a). The set of target languages ideally encompasses most of the world's languages. However, the source languages available for training are often concentrated around few families, geographic areas, and typological features (Cotterell and Eisner, 2017; Gerz et al., 2018b,a; Ponti et al., 2020; Clark et al., 2020). As a consequence of this discrepancy, a language drawn at random might have no related languages available for training. Even when this is not the case, they might provide a scarce amount of examples for supervision.

To illustrate this point, consider Universal Dependencies (UD; Zeman et al., 2020), hitherto the most comprehensive collection of manually curated multilingual data. First, out of 245 families attested in the world according to Glottolog (Hammarström et al., 2016), UD covers only 18.[1] In fact, some families are chronically over-represented (e.g. Indo-European and Uralic) and others are neglected (e.g. Pama-Nyugan and Uto-Aztecan). Second, as shown in Figure 1, the allocation of labelled examples across families is imbalanced (e.g. note the low counts for Niger–Congo or Dravidian languages). Third, one can measure how representative the linguistic traits of training languages are in comparison to those encountered around the globe. In

---

[1] For more details on family distributions, cf. Figure 5 in the Appendix.

Figure 2, we represent UD languages as dots in the space of possible typological features in WALS (Dryer and Haspelmath, 2013). These are plotted against the density of the distribution based on all languages in existence. Crucially, it emerges that UD languages mostly lie in a low-density region. Therefore, they hardly reflect the variety of possible combinations of typological features.

In general, this demonstrates that the distribution of training languages in existing NLP datasets is heavily skewed compared to the real-world distribution. Indeed, this very argument holds true *a fortiori* in smaller, less diverse datasets. While this fact is undisputed in the literature, its consequences for modelling, which we expound in the next section, are often under-estimated.

3 Robust MAML

Model-Agnostic Meta Learning (MAML; Finn et al., 2017) has recently emerged as an effective approach to cross-lingual transfer (Gu et al., 2018; Nooralahzadeh et al., 2020; Wu et al., 2020; Li et al., 2020). MAML seeks a good initialisation point for neural weights in order to adapt them to new languages with only a few examples. To this end, for each language $\mathcal{T}_i$ a neural model $f_{\boldsymbol{\vartheta}}$ is updated according to the loss on a batch of examples $\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\vartheta}}, \mathcal{D}_{train})$. This inner loop is iterated for k steps. Afterwards, the loss incurred by the model on a held-out batch $\mathcal{D}_{val}$ is compounded with those of the other languages as part of an outer loop, as shown in Equation (1):

$$\begin{aligned} \boldsymbol{\vartheta}^\star &= \min_{\boldsymbol{\vartheta}} \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\varphi}_i}, \mathcal{D}_{val})\, p(\mathcal{T}_i) \\ \text{where} \quad \boldsymbol{\varphi}_i &= \boldsymbol{\vartheta} - \eta \nabla_{\boldsymbol{\vartheta}} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\vartheta}}, \mathcal{D}_{train}) \end{aligned} \tag{1}$$

where $\eta \in \mathbb{R}_{>0}$ is the learning rate. Language probabilities are often taken to follow a discrete uniform distribution $p(\mathcal{T}_i) = \frac{1}{|\mathcal{T}|}$. In this case, Equation (1) becomes a simple average.

MAML can also be interpreted as point estimate inference in a hierarchical Bayesian graphical model (see Figure 4 in the Appendix). In this case, the adapted parameters $\boldsymbol{\varphi}_i$ are equivalent to an intermediate language-specific variable acting as a bridge between the language-agnostic parameters $\boldsymbol{\vartheta}$ and the data (Grant et al., 2018; Finn et al., 2018; Yoon et al., 2018). This allows us to reason about the conditions under which a model is expected to generalise to new languages. Crucially, generalisation rests on the assumption of independence and identical distribution among the examples (including both train and evaluation), which is known as exchangeability (Zabell, 2005). However, as seen in Section 2, most of the world's languages are outliers with respect to the training language distribution. Therefore, there is no solid guarantee that meta-learning may fulfil its purpose, i.e. generalise to *held-out* languages.

3.1 Decision-Theoretic Perspective

To remedy the mismatch between assumptions and realistic conditions, in this work we propose objectives which can serve as alternatives to Equation (1) of vanilla MAML. These are rooted in an interpretation of MAML within a decision-theoretic perspective (Bickel and Doksum, 2015, ch. 1.3), which we outline in what follows. The quantity of interest we aim at learning is the neural parameters $\boldsymbol{\vartheta}$. Therefore, the action space for a classification task assigning labels $y \in \mathcal{Y}$ to inputs $\mathbf{x} \in \mathcal{X}$ is $\mathcal{A} = \{f_{\boldsymbol{\vartheta}} : \mathcal{X} \to \mathcal{Y}\}$. The risk function is in turn a function $\mathcal{R} : \mathcal{F} \times \mathcal{A} \to \mathbb{R}^+$, which is the loss incurred by taking an action in $\mathcal{A}$ (making a prediction with a specific configuration of neural parameters) when the 'state of nature', the true function, is $f \in \mathcal{F}$. In the case of MAML, this is represented by the language-specific inner loop loss $\mathcal{L}_{\mathcal{T}_i}(\cdot)$ in Equation (1).

The decision for the optimal action given the sample space, the function $\delta : \mathcal{X} \times \mathcal{Y} \to \mathcal{A}$, is usually determined via gradient descent optimisation for a neural network. The optimal action, however, may vary depending on the language, which results in multiple possible 'states of nature'. Usually, there is no procedure δ whose loss is inferior to all others, such that:

$$\nexists \delta \;\; \mathcal{L}(\mathcal{T}_i, \delta) < \mathcal{L}(\mathcal{T}_i, \delta') \; \forall \mathcal{T}_i \in \mathcal{T}, \delta \neq \delta' \tag{2}$$

Therefore, decision functions have to be compared based on a global criterion rather than in a pairwise fashion between languages. As previously anticipated, Equation (1) minimizes the *expected* risk across languages, for an arbitrary choice of prior $p(\mathcal{T})$. In decision theory, a decision $\delta^\star$ with this property is called *Bayes criterion.*

3.2 Alternative Criteria

There exist alternative criteria to the Bayes criterion that are more justified in a setting that entails transfer between non-i.i.d. domains. Rather than minimising the Bayes risk, in this work, we propose to adjust MAML to either minimise the maximum risk (minimax criterion) or to enforce constraints on the risk for a subset of languages (Neyman–Pearson criterion). This is likely to yield more robust predictions for languages that are outliers to the training distribution. As demonstrated in Section 2, this definition encompasses most of the world's languages.

3.2.1 Minimax Criterion

Rather than the expected risk, the criterion could depend instead on the worst case scenario, i.e. the language for which the risk is *maximum*. This requires to select such a language with max. As an alternative to reinforcement learning (Zhang et al., 2020), to keep our model fully differentiable, we relax the operator by treating the choice of language as a categorical distribution $\mathcal{T}_i \sim \text{Cat}(\cdot \mid \boldsymbol{\tau})$. The parameters $\boldsymbol{\tau} \in [0,1]^{|\mathcal{T}|}, \sum_i \tau_i = 1$ consist of language probabilities and are learned in an adversarial fashion:

$$\min_{\boldsymbol{\vartheta}} \max_{\mathcal{T}_i \sim \text{Cat}(\cdot \mid \boldsymbol{\tau})} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\vartheta} - \eta \nabla_{\boldsymbol{\vartheta}} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\vartheta}}, \mathcal{D}_{train})}, \mathcal{D}_{val}) \tag{3}$$

Equation (3) can be interpreted as a two-player game between us (the scientists) and nature. We pick an action $\boldsymbol{\vartheta}$. Then nature picks a language $\mathcal{T}_i \in p(\mathcal{T})$ for which the risk is maximum given our chosen action. Therefore, our goal becomes to minimise such risk.

3.2.2 Neyman–Pearson Criterion

As an alternative, we might consider minimising the expected risk, but subject to a guarantee that the risk does not exceed a certain threshold for a subset of languages. In practice, we may want to enforce a set of inequality constrains, so that we minimise Equation (1) subject to $\{\mathcal{L}_{\mathcal{T}_i} \leq r \; \forall \mathcal{T}_i \in \mathcal{C}\}$, where $r \in \mathbb{R}_+$ is a hyper-parameter. In general, $\mathcal{C} \subseteq \mathcal{T}$ can be any subset of the training languages; in practice, here we take $\mathcal{C} = \mathcal{T}$. Constrained optimisation is usually implemented through Lagrange multipliers, where we add as many new terms to the objective as we have constraints (Bishop, 2006, ch. 7):

$$\begin{aligned} &\min_{\boldsymbol{\vartheta}} \max_{\boldsymbol{\lambda}} \sum_{\mathcal{T}_i} \frac{1}{|\mathcal{T}|} \mathcal{L}_{\mathcal{T}_i} + \sum_{\mathcal{T}_i} \lambda_i (\mathcal{L}_{\mathcal{T}_i} - r) \\ &= \min_{\boldsymbol{\vartheta}} \max_{\boldsymbol{\lambda}} \sum_{\mathcal{T}_i} \left(\frac{1}{|\mathcal{T}|} + \lambda_i \right) \mathcal{L}_{\mathcal{T}_i} - \lambda_i r \end{aligned} \tag{4}$$

where $\boldsymbol{\lambda}$ is a vector of non-negative Lagrange multipliers $\{\lambda_i \geq 0 \quad \forall \lambda_i \in \boldsymbol{\lambda}\}$ to be learned together with the parameters $\boldsymbol{\vartheta}$, but adversarially.

Intuitively, if the risk for the estimated parameters $\boldsymbol{\vartheta}$ lies in the permissible range, the constraints should become inactive $\{\lambda_i = 0 \quad \forall \lambda_i \in \boldsymbol{\lambda}\}$, i.e. each Lagrange multiplier should go towards 0. Otherwise, the solution should be affected by the constraints, which should keep $\boldsymbol{\vartheta}$ from trespassing the boundary $\{\mathcal{L}(\boldsymbol{\vartheta})_{\mathcal{T}_i} = r \quad \forall \mathcal{T}_i \in \mathcal{T}\}$. In gradient-based optimisation, this unfolds as follows: the gradient of each λ_i depends uniquely on $(\mathcal{L}_{\mathcal{T}_i} - r)$. Due to being maximised, the value of each λ_i increases when the corresponding risk is above the threshold, and shrinks otherwise. Incidentally, note that the Lagrangian multipliers at the critical point $\boldsymbol{\vartheta}^\star$ are equal to the negative rate of change of r, as $\frac{\partial \mathcal{R}(\boldsymbol{\vartheta}^\star)}{\partial r} = -\boldsymbol{\lambda}$. In other words, upon convergence λ_i expresses how much we can decrease the risk in $\mathcal{T}_i$ as we increase the threshold.

Constrained Parameters The additional variables $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$, contrary to the neural parameters, are constrained in the values they can take. In neural networks, there are two widespread approaches to coerce variables within a certain range, viz. reparametrisation and gradient projection (Beck and Teboulle, 2003).[2] For simplicity's sake, we opt for the former, which just requires us to learn unconstrained variables and scale them with the appropriate functions. Thus, we redefine the abovementioned variables as $\boldsymbol{\tau} \triangleq \text{softmax}(\boldsymbol{\tau}_u)$ and $\boldsymbol{\lambda} \triangleq \text{softplus}(\boldsymbol{\lambda}_u)$.

4 Optimisation in 2-Player Games

Based on the formulation of Minimax MAML and Neyman–Pearson MAML in Section 3.2, both are evidently instances of two-player games. On one hand, the first agent minimises the risk with respect to $\boldsymbol{\vartheta}$; on the other, the second agent maximises the risk with respect to $\boldsymbol{\tau}$ (for minimax) or $\boldsymbol{\lambda}$ (for Neyman–Pearson). In other words, both optimise the same (empirical risk) function in Equation (3) or Equation (4), respectively, but with opposite signs. However, the first term of Equation (4) does not depend on $\boldsymbol{\lambda}$. Therefore, Minimax MAML is a zero-sum game, but not Neyman–Pearson MAML.

If the risk function were convex, the solution would be well-defined as the Nash equilibrium. But

---

[2] `https://vene.ro/blog/mirror-descent.html`

this is not the case for a non-linear function such as a deep neural network. Therefore, we resort to an approximate solution through optimisation. The simplest approach in this scenario is Gradient Descent Ascent (GDA), where the set of parameters of both players are optimised simultaneously through gradient descent for the first player and gradient ascent for the second player. With a slight abuse of notation, let us define $\mathcal{R} \triangleq \mathcal{R}(\boldsymbol{\vartheta}_t, \boldsymbol{\alpha}_t)$, where $\boldsymbol{\alpha}_t$ stands for the adversarial parameters ($\boldsymbol{\tau}_t$ for Minimax and $\boldsymbol{\lambda}_t$ for Neyman–Pearson) at time t. Then the update rule is:

$$\boldsymbol{\vartheta}_{t+1} = \boldsymbol{\vartheta}_t - \eta \nabla_{\boldsymbol{\vartheta}} \mathcal{R} \tag{5}$$

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta \nabla_{\boldsymbol{\alpha}} \mathcal{R} \tag{6}$$

for a learning rate $\eta \in \mathbb{R}$. Equations (5) and (6) are equivalent to allowing each player to ignore the other's move and act as if it will remain stationary. This naïve assumption often leads to divergence or sub-par solutions during optimisation (Schäfer and Anandkumar, 2019).

4.1 Symplectic Gradient Adjustment

To overcome the limitations of GDA, several independent works (Balduzzi et al., 2018; Letcher et al., 2019; Gemp and Mahadevan, 2018) proposed to correct Equations (5) and (6) with an additional term. This consists of a matrix-vector product between the mixed second-order derivatives ($\mathrm{D}^2_{\boldsymbol{\vartheta}\boldsymbol{\alpha}}\mathcal{R}$ and $\mathrm{D}^2_{\boldsymbol{\alpha}\boldsymbol{\vartheta}}\mathcal{R}$, respectively)[3] and the gradient of the risk with respect to the adversarial parameters ($\nabla_{\boldsymbol{\alpha}}\mathcal{R}$ and $\nabla_{\boldsymbol{\vartheta}}\mathcal{R}$, respectively). The resulting optimisation algorithm, Symplectic Gradient Adjustment (SGA), updates parameters as follows:

$$\boldsymbol{\vartheta}_{t+1} = \boldsymbol{\vartheta}_t - \eta \nabla_{\boldsymbol{\vartheta}} \mathcal{R} - \eta^2 \mathrm{D}^2_{\boldsymbol{\vartheta}\boldsymbol{\alpha}} \mathcal{R}\ \nabla_{\boldsymbol{\alpha}} \mathcal{R} \tag{7}$$

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta \nabla_{\boldsymbol{\alpha}} \mathcal{R} - \eta^2 \mathrm{D}^2_{\boldsymbol{\alpha}\boldsymbol{\vartheta}} \mathcal{R}\ \nabla_{\boldsymbol{\vartheta}} \mathcal{R} \tag{8}$$

Intuitively, the mixed second-order derivative represents the interaction between the players, and the adversarial gradient represents the opponent's move if they follow the simple GDA strategy. Schäfer and Anandkumar (2019) cogently demonstrate how Equations (7) and (8) correspond to an approximation of the Nash equilibrium[4] of a local bi-linear approximation (with quadratic regulariser) of the underlying game dynamics.

Algorithm 1 Adaptive Symplectic Gradient Adjustment (ASGA)

```
Require: $\eta \in \mathbb{R}_+$: Learning rate
Require: $\beta_1, \beta_2 \in [0, 1)$: Decay rates
Require: $\boldsymbol{\vartheta}_0, \boldsymbol{\alpha}_0$: Initial parameter values
Require: $\mathcal{R} \triangleq \mathcal{R}(\boldsymbol{\vartheta}_{t-1}, \boldsymbol{\alpha}_{t-1}) : \mathbb{R}^{|\boldsymbol{\vartheta}|+|\boldsymbol{\alpha}|} \to \mathbb{R}$
 1: $\boldsymbol{m}_0 \leftarrow \mathbf{0}$ Initialise first moments
 2: $\boldsymbol{v}_0 \leftarrow \mathbf{0}$ Initialise second moments
 3: $t \leftarrow 0$ Initialise time step
 4: while $\boldsymbol{\vartheta}_t, \boldsymbol{\alpha}_t$ not converged
 5:     $t \leftarrow t + 1$
 6:     $\boldsymbol{g}_{\boldsymbol{\vartheta},t} \leftarrow \nabla_{\boldsymbol{\vartheta}}\mathcal{R} + \eta D_{\boldsymbol{\vartheta}\boldsymbol{\alpha}}\mathcal{R}\ \nabla_{\boldsymbol{\alpha}}\mathcal{R}$
 7:     $\boldsymbol{g}_{\boldsymbol{\alpha},t} \leftarrow \nabla_{\boldsymbol{\alpha}}\mathcal{R} - \eta D_{\boldsymbol{\alpha}\boldsymbol{\vartheta}}\mathcal{R}\ \nabla_{\boldsymbol{\vartheta}}\mathcal{R}$
 8:     $\boldsymbol{g}_t \leftarrow \boldsymbol{g}_{\boldsymbol{\vartheta},t} \oplus \boldsymbol{g}_{\boldsymbol{\alpha},t}$
 9:     $\boldsymbol{m}_t \leftarrow \beta_1\, \boldsymbol{m}_{t-1} + (1 - \beta_1)\, \boldsymbol{g}_t$
10:     $\boldsymbol{v}_t \leftarrow \beta_2\, \boldsymbol{v}_{t-1} + (1 - \beta_2)\, \boldsymbol{g}_t^2$
11:     $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t \,/\, (1 - \beta_1^t)$
12:     $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t \,/\, (1 - \beta_2^t)$
13:     $\boldsymbol{\vartheta}_t \leftarrow \boldsymbol{\vartheta}_{t-1} - \eta \cdot \hat{\boldsymbol{m}}_{\boldsymbol{\vartheta},t} \,/\, (\sqrt{\hat{\boldsymbol{v}}_{\boldsymbol{\vartheta},t}} + \epsilon)$
14:     $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} + \eta \cdot \hat{\boldsymbol{m}}_{\boldsymbol{\alpha},t} \,/\, (\sqrt{\hat{\boldsymbol{v}}_{\boldsymbol{\alpha},t}} + \epsilon)$
15: return $\boldsymbol{\vartheta}_t, \boldsymbol{\alpha}_t$
```

In practice, estimating the above-mentioned products is tedious because of their space and time complexity. Therefore, we resort to an approximation known as *Hessian-vector product* (Pearlmutter, 1994). For the third term of Equation (7):

$$\mathrm{D}^2_{\boldsymbol{\vartheta}\boldsymbol{\alpha}}\mathcal{R}(\boldsymbol{\vartheta}, \boldsymbol{\alpha})\nabla_{\boldsymbol{\alpha}}\mathcal{R}(\boldsymbol{\vartheta}, \boldsymbol{\alpha}) = \frac{\partial}{\partial h}\nabla_{\boldsymbol{\vartheta}}\mathcal{R}(\boldsymbol{\vartheta}, \boldsymbol{\alpha} + h\nabla_{\boldsymbol{\alpha}}\mathcal{R}(\boldsymbol{\vartheta}, \boldsymbol{\alpha}))\bigg|_{h=0} \tag{9}$$

And similarly for the matrix product term in Equation (8), by swapping $\boldsymbol{\vartheta}$ and $\boldsymbol{\alpha}$ in Equation (9).

4.2 Adaptive Learning Rate and Momentum

While SGA may provide a more appropriate optimisation framework for competitive games, it still lacks several defining features of optimisers that accelerate convergence, such as first-order momentum and adaptive learning rate (second-order momentum). Therefore, we modify the update rule in Equations (7) and (8) to include both of these. Our starting point is Adam (Kingma and Ba, 2015). The changes we apply are the following (also illustrated in Algorithm 1):

① The current difference (lines 6–7) is adjusted with the terms introduced in Equations (7) and (8) by Schäfer and Anandkumar (2019).

---

[3] Here $\mathrm{D}^2_{\boldsymbol{wz}}\mathcal{R}$ stands for the sub-matrix of the Hessian containing the derivative of the risk taken first with respect to $\boldsymbol{w}$ and then with respect to $\boldsymbol{z}$.

[4] A Nash equilibrium is a pair of strategies whose unilateral modification cannot result in loss reductions.

② The exponentially decayed, unbiased estimates of the expectations over mean and standard deviation are computed similarly to Adam. However, note that, in line 14, the update of the adversarial parameters corresponds to an ascent (rather than a descent).

This results in a novel optimiser, Adaptive Symplectic Gradient Adjustment (ASGA). We employ ASGA in our experiments to optimise the objectives of Minimax MAML and Neyman–Pearson MAML, as it enables a fair comparison with Adam-optimised Bayes MAML.

5 Experiments

We now outline the main experiments of our work on multilingual NLP. We evaluate our methods on part-of-speech (POS) tagging, a sequence labelling tasks, and question answering (QA), a natural language understanding task.

We focus on POS given its ample coverage of languages and its frequent use as a benchmark for resource-poor NLP (Das and Petrov, 2011; Ponti et al., 2021). In fact, cross-lingual transfer in sequence labelling tasks was demonstrated to be the most challenging, as knowledge of linguistic structure is more language-dependent than semantics (Hu et al., 2020). However, we also include QA to illustrate the generality of our methods for cross-lingual NLP. In this task, given the gold passage and a question, the system has to predict the beginning and end positions of a single contiguous span containing the answer.

Data. POS data are sourced from the Universal Dependencies (UD) treebanks[5] (Zeman et al., 2020) and QA data from the 'gold passage' variant of TyDiQA (Clark et al., 2020).[6] We retain the original training, development, and evaluation sets of UD. In TyDiQA, we use the original development set for evaluation.[7] For meta-learning, $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ examples are both obtained from disjoint parts of the training set.

We aim to create a partition of languages between training and evaluation that corresponds to the most realistic scenario in deploying NLP technology on resource-poor languages spoken around the world. Therefore, we reserve for evaluation all language isolates and languages with at most 2 family members in each dataset. We use all the remaining languages in the dataset for training. Therefore, for POS, the evaluation set spans 16 treebanks (14 languages, 11 families) and the training set 99 treebanks; QA comprises 9 languages (7 families). We hold out 4 of them in turn for evaluation (except English) and use the rest for training. We provide the full list of languages in Appendix A.

Training. In all tasks, we train a neural network consisting of two stacked modules: an encoder and a classifier. The encoder is a 12-layer, 768-hidden unit, 12-head Transformer initialised with multilingual BERT Base (mBERT), which was pre-trained on cased text from 104 languages.[8] The classifier is a single affine layer for TyDiQA and a 2-layer Perceptron (with 1024 hidden units) for POS tagging. The combined parameters of the encoder and classifier correspond to $\boldsymbol{\vartheta}$ from Section 3.

These are meta-learned via Meta-SGD (Li et al., 2017), a first-order MAML variant where each parameter is assigned a separate inner-loop learning rate η. Moreover, each η is trained end-to-end based on the outer-loop loss (such as Equation (1) for the Bayes criterion).[9] Similar to Bansal et al. (2020), to avoid an explosion in the number of parameters, we assign a per-layer learning rate (rather than per-parameter). To avoid overfitting, we employ both dropout (with a probability of 0.2) and early stopping (with a patience of 10). For the Neyman-Pearson formulation, we set $r = 0.1$ as a threshold for all language-specific losses.[10] The parameters $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$ were initialized uniformly as $\frac{1}{|\mathcal{T}|}$. Complete details of the hyper-parameters for all settings are given in Appendix B.

Methods. To assess the effectiveness of the proposed criteria and optimisers, we compare them with two competitive baselines, while maintaining the same underlying neural architecture: (i) **J**: a joint multi-source transfer method where a model is trained on the concatenation of the datasets for all languages; (ii) **B:** the original MAML (Finn et al., 2017) with Bayes criterion and uniform prior. Our choice of baselines is justified by the fact that these methods (or variations thereof) are currently

---

[5] `https://universaldependencies.org/`

[6] `https://github.com/google-research-datasets/tydiqa`

[7] This is necessary as we need to access this set to simulate few-shot learning, but the original evaluation set is not public.

[8] `https://github.com/google-research/bert/blob/master/multilingual.md`

[9] We implement Meta-SGD with the `learn2learn` package (Arnold et al., 2020).

[10] We also experimented with a dynamic threshold which corresponded to the average language-specific loss of the last 10 episodes. However, this yielded sub-par results.

k ▷	0	5	10	20
		F_1 Score		
J	51.01	62.96±2.5	66.00±1.9	68.66±1.7
B	51.50	63.87±2.8	67.03±2.1	69.46±1.8
MM	51.82	63.67±2.7	66.88±2.0	69.55±1.8
NP	51.68	63.84±2.9	67.13±2.1	69.65±1.9
MM+	52.46	**64.71±2.9**	**67.89±2.3**	**70.25±2.0**
NP+	**53.05**	64.26±2.6	67.57±2.1	69.98±1.9

Table 1: F_1 scores for POS tagging in UD across different k-shots. We report the mean and standard deviation across 16 treebanks.

k ▷	0	5	10	20
		Exact Match		
J	46.76	49.53±3.7	51.54±2.9	**53.51±2.4**
B	46.60	48.41±3.4	50.24±2.9	52.02±2.6
MM	**48.33**	**50.08±3.4**	**51.68±2.9**	**53.49±2.4**
NP	46.71	49.24±3.3	50.95±2.9	52.76±2.4
MM+	46.87	47.74±3.8	49.42±3.4	51.40±2.5
NP+	48.02	48.77±3.9	50.75±3.1	52.66±2.6
		F_1 Score		
J	61.66	63.75±3.3	65.39±2.3	67.01±1.9
B	62.51	63.29±3.2	64.87±2.5	66.31±2.1
MM	**63.06**	**64.37±3.1**	**65.83±2.6**	**67.45±2.1**
NP	61.89	63.84±2.9	65.23±2.6	66.88±1.9
MM+	62.10	62.63±3.2	64.11±2.9	65.89±2.1
NP+	62.75	62.98±3.6	64.77±2.9	66.57±2.2

Table 2: Results for QA in TyDiQA across different k-shots. We report the mean and standard deviation across 8 languages of the exact match score (above) and the F_1 score (below).

state of the art for the tasks of POS and QA, as well as other innumerable NLP applications (Wu and Dredze, 2019; Nooralahzadeh et al., 2020; Ponti et al., 2021). In addition, we evaluate the following combinations: (iii) **MM:** MAML with a minimax criterion, optimised with GDA; (iv) **NP:** MAML with a Neyman–Pearson (constrained) criterion, optimised with GDA; (v) **MM+:** MAML with a minimax criterion, optimised with ASGA; and (vi) **NP+:** MAML with a Neyman–Pearson criterion, optimised with ASGA.

Evaluation. For each evaluation language in a given task, we randomly sample $k \in \{0, 5, 10, 20\}$ examples from the evaluation data as the support set (for adaptation) and the rest of the examples as the query set (for testing). When $k > 0$, we repeat the evaluation 100 times and report the following average metrics: (i) F_1 score for POS tagging, and (ii) exact-match (EM) and F_1 scores for QA.[11] Due to lack of space, we only report the average mean and standard deviation across languages for each model described above.

6 Results and Discussion

We report the results for POS tagging in Table 1 and for QA in Table 2. These include mean and standard deviation across languages. Note that, in this case, the standard deviation is by no means an interval for statistical significance, but rather reflects the heterogeneity among the evaluation languages. In what follows, we address a series of questions in the light of these figures.

Baselines. MAML and joint multi-source transfer are both strong contenders as state-of-the-art methods for cross-lingual transfer, but which one is better? By comparing J and B rows, no definite response emerges in our experiments. While MAML outperforms its competitor in POS tagging, it lags behind in QA. We speculate that the larger pool of training languages available in POS tagging (22 times more than QA) endows meta-learning with better generalisation capabilities. Both methods, however, surpass single-source transfer from English SQuAD (Rajpurkar et al., 2016) in the zero-shot setting by a large margin: Clark et al. (2020) report 56.4 F_1 score in average for TyDiQA, which is 6.66 points below our best model.

Criteria. The minimax and Neyman-Pearson criteria both improve over the Bayes criterion baseline, although the latter more sporadically. Compared to the B rows, MM+ achieves gains for every k in POS tagging, with 0.94 points of margin at $k = 0$ and 0.79 at $k = 20$. The same holds for MM in QA, with margins that span from 1.73 at $k = 0$ to 1.47 at $k = 20$ in the Exact Match metric, and from 0.55 at $k = 0$ to 1.14 at $k = 20$ in F_1 score. Therefore, Minimax MAML is remarkably consistent in outperforming the baselines, although the gains are sometimes significant, sometimes only marginal. This is also reflected in language-specific performances, available in Table 5 and Table 6 in the Appendix. For POS tagging, the F_1 scores of only 2 languages (Indonesian and Naija) moderately decrease, whereas the rest of the 14 languages show improvements.

Incidentally, it may be worth noting that we did not perform any large-scale search over hyperparameters like $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$ initialisations, the threshold r, or differential learning rates for maximised and minimised parameters. Therefore, these early

---

[11] We refer the reader to Rajpurkar et al. (2016) for a precise definition of these metrics.

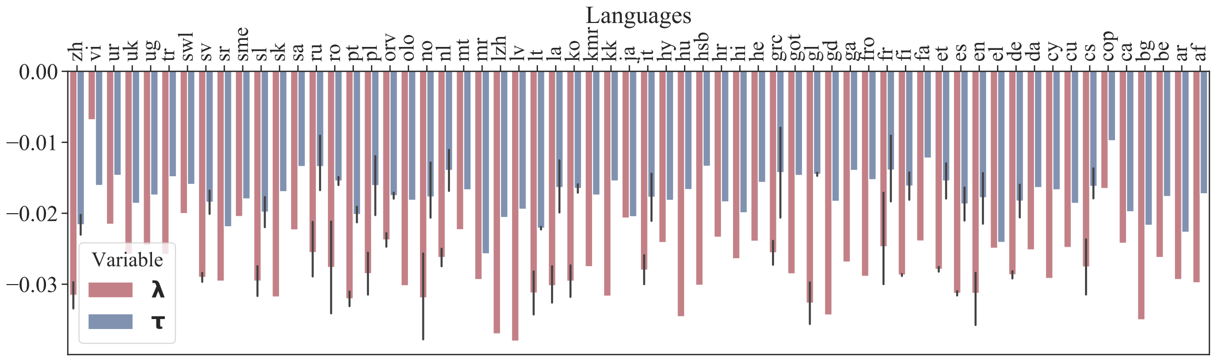

Figure 3: Unconstrained values of $\boldsymbol{\tau}_u$ and $\boldsymbol{\lambda}_u$ upon convergence in MM+ and NP+ models for POS tagging.

k ▷	0	5	10	20
		F_1 Score		
J	14.34	33.32	37.52	40.83
B	24.11	35.03	40.38	44.92
MM	20.41	37.61	43.00	45.83
NP	**26.81**	39.23	42.70	45.25
MM+	16.42	37.41	43.57	45.21
NP+	22.55	**39.95**	**45.41**	**48.12**

Table 3: The minimum F_1 scores of our models across languages, for POS tagging.

k ▷	0	5	10	20
		Exact Match		
J	42.33	**45.97**	47.22	**49.47**
B	**42.75**	44.58	46.44	48.24
MM	41.01	45.33	**47.59**	49.21
NP	40.39	44.89	46.40	48.80
MM+	41.01	41.87	45.32	47.11
NP+	37.44	42.92	46.30	48.88
		F_1 Score		
J	52.43	59.27	59.88	62.11
B	51.10	**59.60**	60.64	62.10
MM	53.10	59.21	**61.86**	63.43
NP	52.83	59.31	60.03	61.84
MM+	51.93	57.91	59.86	61.52
NP+	**53.96**	57.21	61.74	**63.55**

Table 4: The minimum Exact Match and F_1 scores of our models across languages, for QA.

results are amenable to improve even further in the future. This lends credence to our proposition that minimax and Neyman–Pearson criteria are more suited for out-of-distribution transfer to outlier languages.

Optimiser. The results for the proposed optimiser ASGA (Algorithm 1) are favourable in comparison to Gradient Descent Ascent via Adam (Kingma and Ba, 2015) for POS tagging; on the other hand, the opposite trend is observed for QA. Therefore, future investigations are required to shed further light on modifications such as the Symplectic Gradient Adjustment. A tentative explanation of such discrepancy could be the disproportionate number of training languages available in either task.

To get insights into the game dynamics of the adversarial criteria, we plot the unconstrained values for $\boldsymbol{\tau}_u$ and $\boldsymbol{\lambda}_u$ upon convergence in Figure 3. Interestingly, both variables appear to follow the same profile of peaks and troughs; therefore, as expected, languages chosen adversarially in MM have also higher Laplace multipliers in NP. To this group belong for instance languages with rare scripts (e.g. Coptic) or with no relatives in the training languages (e.g. Vietnamese). As a final note, we remark that the proposed criteria and optimiser are in principle more general than NLP and could facilitate transfer in other fields. While this thread of research transcends the scope of our work, we illustrate an example for regression in Appendix C.

Minimum Scores across Languages. In addition to the *average* cross-lingual performance, we also report the *minimum* cross-lingual performance for POS tagging in Table 3 and for QA in Table 4. This corresponds to the lowest score achieved across all evaluation languages. For POS tagging, we observe that NP and NP+ outperform J and B by 7-12 and 2-5 F1 points, respectively. This reveals that worst-case and constrained risk minimisation drastically uplifts the scores for the most disadvantaged language. Nevertheless, the opposite trend is observed for QA: MM(+) and NP(+) do not alter the minimum score with respect to the F_1 metric, and even degrade it with respect to the exact-match metric. Again, we conjecture that these mixed findings may depend on the different amount and distribution of the training languages in the corresponding datasets: UD offers greater language coverage than TyDiQA, which gives better guidance.

7 Related Work

MAML is a cutting-edge method for cross-lingual transfer in several NLP tasks (Gu et al., 2018; Nooralahzadeh et al., 2020; Wu et al., 2020; Li et al., 2020, *inter alia*). However, in all these experiments, the model is adopted in its standard formulation, minimising the expected risk. Therefore, its performance is prone to suffer in outlier languages. Moreover, the assumptions underlying our proposed variants are different from other instances of robust optimisation in NLP (Globerson and Roweis, 2006; Oren et al., 2019). In particular, the target language distributions are not explicitly treated as subspaces or covariate shifts of source languages. In separate fields such as vision, previous attempts at worst-case-aware meta-learning include Collins et al. (2020), who use a Euclidean version of the robust stochastic mirror-prox algorithm, and Wang et al. (2020), who rely on reinforcement learning. Our formulation is both fully differentiable and broader, as the decision-theoretic interpretation admits alternative criteria for MAML. What is more, to our knowledge we are the first to successfully augment MAML with minimax criteria in cross-lingual NLP and with Neyman–Pearson criteria in general.

8 Conclusions

To perform cross-lingual transfer to low-resource languages, under a decision-theoretic interpretation Model-Agnostic Meta-Learning (MAML) minimises the expected risk across training languages. Generalisation then relies on the evaluation languages being identically distributed. However, this assumption is incongruous for cross-lingual transfer in realistic scenarios. Therefore, we propose more appropriate training objectives that are robust to out-of-distribution transfer: Minimax MAML, where worst-case risk is minimised by learning an adversarial distribution over languages; and Neyman–Pearson MAML, where constraints are imposed on language-specific losses, so that they remain below a certain threshold. From a game-theoretic perspective, both of these variants consist of 2-player competitive games. Therefore, we also explore adaptive optimisers that take into account the underlying game dynamics. The experimental results on zero-shot and few-shot learning for part-of-speech tagging and question answering, whose datasets span tens of typologically diverse languages, confirm that in several settings the proposed criteria are superior to both vanilla MAML and transfer from multiple source languages.

Acknowledgements

We thank the reviewers for their valuable feedback. Rahul Aralikatte and Anders Søgaard are funded by a Google Focused Research Award.

References

Sébastien M. R. Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. 2020. learn2learn: A library for meta-learning research. *arXiv preprint arXiv:2008.12284*.

David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. 2018. The mechanics of n-player differentiable games. In *Proceedings of the 35th International Conference on Machine Learning*, pages 354–363, Stockholm, Sweden.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123, online.

Amir Beck and Marc Teboulle. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Emily M. Bender. 2009. Linguistically naïve != language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32, Athens, Greece.

Peter J. Bickel and Kjell A. Doksum. 2015. *Mathematical statistics: Basic ideas and selected topics, volume I*. CRC Press.

Christopher M. Bishop. 2006. *Pattern recognition and machine learning*. Springer.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. 2020. Task-robust model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, volume 33, online.

Ryan Cotterell and Jason Eisner. 2017. Probabilistic typology: Deep generative models of vowel inventories. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1182–1192, Vancouver, Canada.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, Sydney, Australia.

Chelsea Finn, Kelvin Xu, and Sergey Levine. 2018. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, volume 31, pages 9516–9527, Montreal, Canada.

Ian Gemp and Sridhar Mahadevan. 2018. Global convergence to the equilibrium of GANs using variational inequalities. *arXiv preprint arXiv:1808.01531*.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018a. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association for Computational Linguistics*, 6:451–465.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018b. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 316–327, Brussels, Belgium.

Amir Globerson and Sam Roweis. 2006. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, page 353–360, New York, New York, USA.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations*, Vancouver, Canada.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank, editors. 2016. *Glottolog 2.7*. Max Planck Institute for the Science of Human History, Jena.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4411–4421, online.

Chi Jin, Praneeth Netrapalli, and Michael Jordan. 2020. What is local optimality in nonconvex–nonconcave minimax optimization? In *Proceedings of the 37th International Conference on Machine Learning*, pages 4880–4889, online.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, online.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, San Diego, California, USA.

Alistair Letcher, David Balduzzi, Sébastien Racaniere, James Martens, Jakob N Foerster, Karl Tuyls, and Thore Graepel. 2019. Differentiable game mechanics. *Journal of Machine Learning Research*, 20:84–1.

Zheng Li, Mukul Kumar, William Headden, Bing Yin, Ying Wei, Yu Zhang, and Qiang Yang. 2020. Learn to cross-lingual transfer with meta graph learning across heterogeneous languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2290–2301, online.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Farhad Nooralahzadeh, Giannis Bekoulis, Johannes Bjerva, and Isabelle Augenstein. 2020. Zero-shot cross-lingual transfer with meta learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4547–4562, online.

Peter Orbanz. 2012. Lecture notes on Bayesian nonparametrics. *Journal of Mathematical Psychology*, 56:1–12.

Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4227–4237, Hong Kong, China.

Barak A. Pearlmutter. 1994. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160.

Edoardo Ponti. 2021. *Inductive Bias and Modular Design for Sample-Efficient Neural Language Learning*. Ph.D. thesis, University of Cambridge.

Edoardo M. Ponti, Ivan Vulić, Ryan Cotterell, Marinela Parovic, Roi Reichart, and Anna Korhonen. 2021. Parameter space factorization for zero-shot learning across tasks and languages. *Transactions of the Association for Computational Linguistics*, 9:410–428.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, online.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2019a. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics*, 45(3):559–601.

Edoardo Maria Ponti, Ivan Vulić, Ryan Cotterell, Roi Reichart, and Anna Korhonen. 2019b. Towards zero-shot language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2900–2910, Hong Kong, China.

Lorien Y Pratt. 1993. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, volume 5, pages 204–204.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas.

Sebastian Ruder. 2019. *Neural transfer learning for natural language processing*. Ph.D. thesis, NUI Galway.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Florian Schäfer and Anima Anandkumar. 2019. Competitive gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, pages 7625–7635, Vancouver, Canada.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with Universal Dependencies and predicted PoS labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349, Uppsala, Sweden.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, online.

Qianhui Wu, Zijia Lin, Guoxin Wang, Hui Chen, Börje F. Karlsson, Biqing Huang, and Chin-Yew Lin. 2020. Enhanced meta-learning for cross-lingual named entity recognition with minimal resources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9274–9281.

Shijie Wu and Mark Dredze. 2019. Beto, Bentz, Becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. 2018. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, volume 31, pages 7332–7342.

Sandy L. Zabell. 2005. *Symmetry and its discontents: essays on the history of inductive probability*. Cambridge University Press.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Željko Agić, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Angelina Aquino, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Furkan Atmaca, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Ethan Chi, Jinho Choi, Yongseok Cho, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa

Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Loïc Grobol, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Tunga Güngör, Nizar Habash, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Oliver Hellwig, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Hildur Jónsdóttir, Fredrik Jørgensen, Markus Juutinen, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Abdullatif Köksal, Kamil Kopacewicz, Timo Korkiakangas, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Maria Levina, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Yuan Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Hiroshi Matsuda, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ‘ Olúòkun, Mai Omura, Emeka Onwuegbuzia, Petya Osenova, Robert Östling, Lilja Øvrelid, Şaziye Betül Özateş, Arzucan Özgür, Balkız Öztürk Başaran, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Petru Rebeja, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Valentin Roșca, Davide Rovati, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Salvatore Scarlata, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Maria Skachedubova, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Takaaki Tanaka, Samson Tella, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Marsida Toska, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Utku Türk, Francis Tyers, Sumire Uematsu, Roman Untilov, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Aya Wakasa, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Paul Widmer, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Kayo Yamashita, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Hanzhi Zhu, and Anna Zhuravleva. 2020. Universal dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sheng Zhang, Xin Zhang, Weiming Zhang, and Anders Søgaard. 2020. Worst-case-aware curriculum learning for zero and few shot transfer. *arXiv preprint arXiv:2009.11138*.

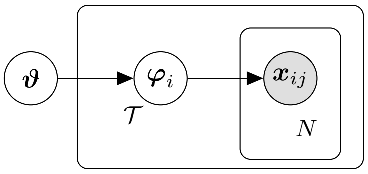

Figure 4: Bayesian graphical model of MAML, where the variable φ_i is parameterised as $\vartheta - \eta\nabla_{\vartheta}\mathcal{L}_{\mathcal{T}_i}(f_{\vartheta}, \mathcal{D}_{train})$.

A Language Partitions

The languages from the following families in UD are held out for evaluation (16 treebanks, 14 languages in total): Northwest Caucasian (Abaza), Mande (Bambara), Mongolic (Buryat), Basque, Tupian (Mbya Guarani), Creole (Naija), Tai–Kadai (Thai), Pama–Nyungan (Warlpiri), Austronesian (Indonesian, Tagalog), Dravidian (Tamil, Telugu), Niger-Congo (Wolof, Yoruba). As all 8 languages in TiDiQA belong to families with at most 2 members in the dataset, we randomly create two partitions: in the former, Finnish, Korean, Bengali, and Arabic are used for evaluation, and the others for training; in the latter, Russian, Indonesian, Telugu, and Swahili are used for evaluation, and the others for training.

B Hyperparameter Setting

POS Tagging. For POS tagging: (i) the batch size was 32, (ii) the maximum sequence length was 128, (iii) the number of epochs was 20, with a patience limit of 10, (iv) both outer and inner learning rates were 5×10^{-5}, (v) the number of episodes per iteration was 32, (vi) the number of inner loops per outer update was 4, (vii) the number of shots (k) during training was 30, and (viii) the hidden layer dropout probability for the classifier was 0.2.

QA. (i) the batch size and k were reduced to 12 due to memory constraints, (ii) the maximum context length was 336, and the document stride was 128, (iii) the maximum question length was 64, (iv) the inner and outer learning rates were 3×10^{-5}.

For all J baselines, we used a uniform language sampler, since proportional sampling performed worse. As an optimiser, we chose Adam with a learning rate of 5×10^{-5}, a weight decay of 0.1;

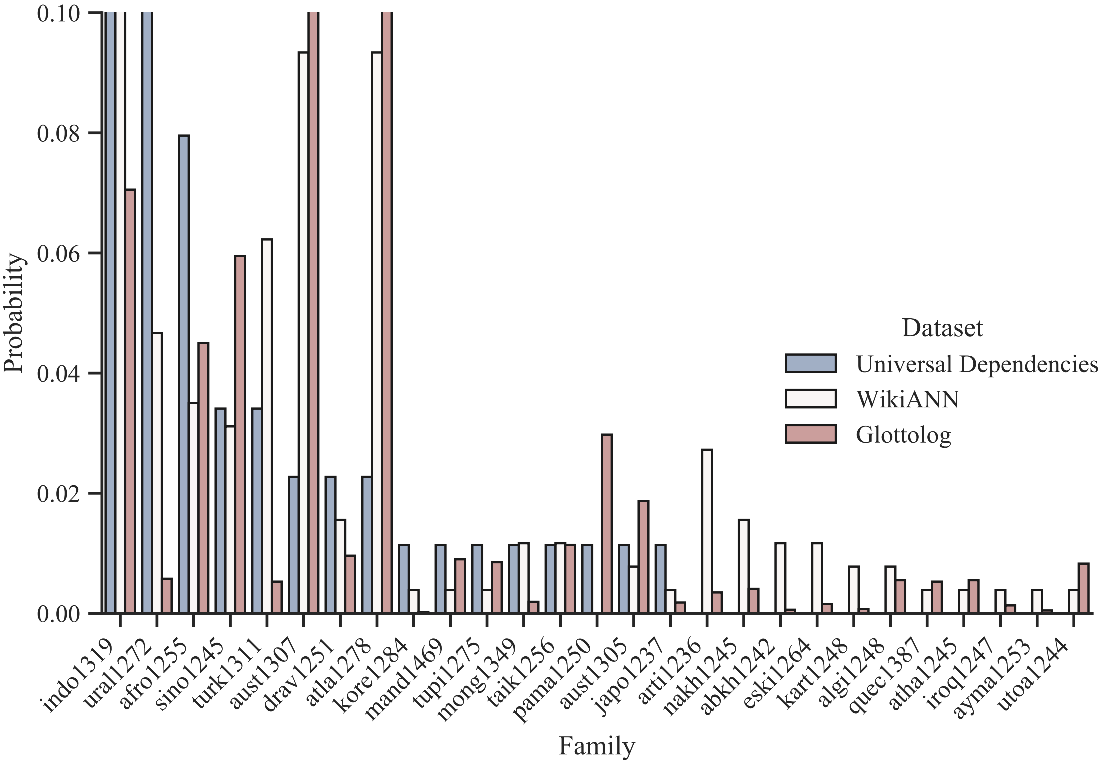

Figure 5: Empirical distribution of languages across families in 2 datasets (WikiANN and UD) and in the world, according to Glottolog. The families shown are a subset {(WikiANN ∪ Universal Dependencies) ∩ Glottolog}. The y-axis is truncated for the sake of clarity.

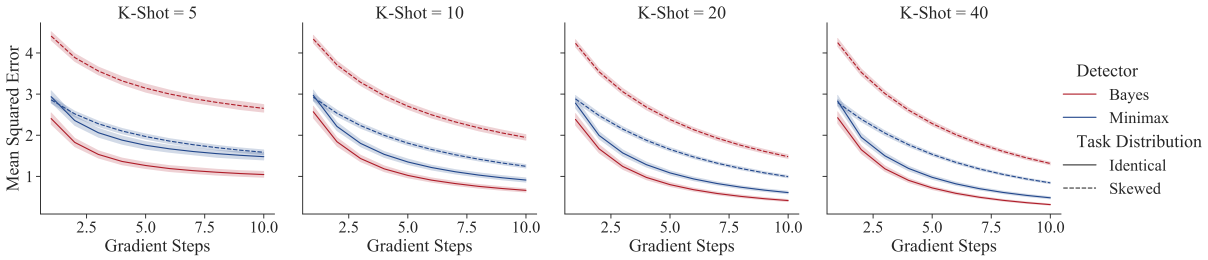

Figure 6: Mean Squared Error of MAML across gradient steps (from 1 to 10) of different criteria (B and MM) under identical and skewed task distributions. Each frame represents a separate run of fast adaptation with different amounts of target examples available (k-shot).

we clipped the gradient to a maximum norm of 5.0. For all MAML models, we performed 4 updates in the inner loop, both during training and fast adaptation (few-shot learning). We ran our experiments on a 48GB NVIDIA Quadro RTX 8000 GPU with Turing micro-architecture. Each run took approximately 2 hours for training and 3 hours for few-shot learning and evaluation.

C Additional Experiments & Results

Additional Results. Table 5 contains POS tagging F_1 scores of all languages, for all models, in both zero and few-shot settings. Table 6 shows the exact match and F_1 scores for QA.

Sinusoidal Regression. After delving into real-world, large-scale NLP applications, we additionally illustrate the effect of the alternative criteria on other ML domains. We run a proof-of-concept experiment on a toy task where we can fully control the distribution of the training and evaluation data, viz. regression of a sinusoidal function.

For this task, we follow the same experimental setting and hyper-parameters of Finn et al. (2017): combinations of amplitudes $a \in [0.1, 5]$ and phases $p \in [0, \pi]$ determine a set of tasks characterised by the function $y = \sin(x - p) \cdot a$. The inputs are sampled at random from the interval $x \in [-5, 5]$.

While both train and evaluation tasks in the original version were sampled uniformly from *identical* ranges, we also construct an alternative setting with *skewed* distributions sampled from disjoint ranges: during training, $a \in [2.5, 5]$ and $p \in [\frac{\pi}{2}, \pi]$; during evaluation, $a \in [0.1, 2.5]$ and $p \in [0, \frac{\pi}{2}]$.

For Minimax MAML, we aim at learning the distribution over tasks adversarially. In particular, we consider two separate discrete categorical distributions for amplitudes $\mathrm{softmax}(\boldsymbol{\tau}_u^{(a)})$ and phases $\mathrm{softmax}(\boldsymbol{\tau}_u^{(p)})$ over their respective ranges discretised into 1,000 atoms. Hence, the probability of a task with the i-th amplitude value and the j-th phase value is simply $\tau_i^{(a)} \times \tau_j^{(p)}$.

The results for sinusoidal regression are shown in Figure 6. Vanilla MAML (Bayes criterion) consistently outperforms the minimax criterion when the task distribution is identical; on the other hand, the reverse occurs when the task distribution is skewed. MM performs much better in this case, with the gap in performance increasing as the shots k decrease. This verifies our hypothesis that the minimax criterion should benefit out-of-distribution regression tasks.

Dataset	k	J	B	MM	NP	MM+	NP+
abq_atb	0	14.34	24.11	20.41	26.81	16.42	22.55
	5	33.32±5.58	35.03±5.85	37.61±4.57	39.23±5.41	37.41±7.3	39.95±5.93
	10	37.52±2.28	40.38±5.75	43±3.63	42.7±5.3	43.57±7.31	46.56±4.91
	20	40.83±6.53	44.92±7.08	45.83±5.59	45.25±7.26	45.21±9.4	48.12±7.19
bm_crb	0	29.56	30.85	29.2	28.57	30.44	30.22
	5	45.6±3.47	50.83±3.33	46.04±3.95	45.14±3.73	48.2±3.74	48.32±3.63
	10	49.75±1.23	54.4±2.73	50.35±2.7	50.01±2.74	51.65±2.87	51.28±3.4
	20	54.03±1.52	57.53±1.68	53.12±2.16	53.39±1.85	54.38±1.85	53.89±2.08
bxr_bdt	0	48.85	51.71	50.41	50.49	54.21	51.94
	5	51.29±1.67	51.81±2.18	51.57±2.21	51.62±2.17	53.09±2.08	51.83±2.31
	10	53.64±0.96	54.95±1.68	54.18±1.53	54.25±1.63	55.47±1.8	55.17±1.43
	20	56.18±1.13	57.23±1.17	56.48±1.49	56.97±1.2	58.19±1.38	57.29±1.12
eu_bdt	0	70.2	71.76	73.22	72.57	73.54	73.29
	5	74.7±1.39	75.74±1.69	75.42±1.59	75.77±1.94	76.58±1.64	76.52±1.64
	10	76.51±2.38	78.1±1.25	77.52±1.01	78.08±1.21	78.73±1.36	78.19±1.38
	20	78.52±0.67	80.09±0.87	79.47±0.84	80.01±0.76	80.69±0.91	80.24±0.78
gun_thomas	0	32.06	35.72	33.91	31.97	33.87	33.84
	5	40.65±2.27	42.62±3.05	43.28±2.64	42.32±2.45	43.12±2.63	42.46±2.37
	10	44.06±0.99	45.65±2.33	45.92±2.59	45.23±2.31	46.98±2.49	45.41±2.25
	20	46.46±2.07	47.96±2.11	50.34±2.3	48.15±2.09	50.67±2.15	48.44±1.74
id_gsd	0	77.24	77.97	77.68	74.79	77.85	76.15
	5	82.2±1.22	83.47±1.22	82.72±1.47	82.35±1.68	83±1.5	82.47±1.57
	10	83.63±0.93	84.69±0.91	84.28±1.17	84.06±1.09	84.4±1.03	84.69±0.96
	20	84.75±0.61	85.75±0.59	85.82±0.58	85.35±0.68	85.94±0.66	85.86±0.69
id_pud	0	68.46	69.41	69.27	68.67	69.41	68.72
	5	73.07±1.39	73.96±1.5	73.5±1.48	74.17±1.43	74.52±1.46	73.82±1.56
	10	74.91±1.33	75.7±1.19	75.5±1.08	75.87±1.15	76.42±0.8	75.85±0.94
	20	76.17±0.57	77.18±0.72	77.06±0.49	77.28±0.68	77.75±0.57	77.39±0.71
pcm_nsc	0	61.97	40.78	45.77	40.76	41.21	56.83
	5	78.17±1.58	77.87±1.27	77.42±1.67	76.48±1.74	77.33±1.55	77.71±1.78
	10	80.06±1.24	79.28±1.25	78.96±1.1	78.41±1.1	78.71±0.94	80.03±1.37
	20	81.61±0.85	80.6±0.81	80.17±0.8	79.97±1	80.13±0.72	81.99±0.99
ta_ttb	0	55.65	56.31	58.12	58.47	60.18	55.93
	5	72.29±2.03	72.39±2.21	71.37±1.7	72.28±2.46	72.34±2.13	70.19±2.3
	10	74.73±2.27	75.36±1.47	73.7±1.36	75.51±1.54	75.11±1.47	73.69±1.73
	20	76.23±1.19	77.56±1.38	75.75±1.39	77.83±1.33	77.44±1.3	76.29±1.49
te_mtg	0	75.21	75.87	77.49	75.43	76.28	76.29
	5	76.45±2.57	73.9±3.87	75.32±2.9	74.74±3.63	74.97±2.87	74.37±3.46
	10	78.68±1.74	77.16±2.55	78.26±2.09	77.55±2.29	77.57±2.12	76.94±2.83
	20	80.13±1.97	79.66±1.64	79.99±2.15	80±2.22	80.09±1.98	80.08±1.99
th_pud	0	42.51	42.71	43.76	43.3	46.81	43.07
	5	58.05±2.53	59.83±2.35	60.02±2.62	61.18±2.74	61.12±2.95	60.15±2.05
	10	61.71±2.17	63.57±1.72	63.85±1.9	65.14±1.67	65.4±1.87	63.34±1.75
	20	65.05±1.28	66.39±1.38	66.62±1.08	67.99±1.41	68.72±1.28	66.27±1.36
tl_trg	0	76.9	77.43	77.59	85.12	82.27	80.62
	5	83.01±3.52	82.95±3.66	84.09±4.75	84.5±4.14	84.4±4.01	84.01±4.64
	10	85.78±1.66	85.4±2.06	86.86±2.3	87.27±2.62	87.23±2.87	87.42±2.12
	20	87.27±2.04	87.48±2.32	88.69±1.96	89.1±2.34	89.2±1.85	89.24±1.86
tl_ugnayan	0	60.37	64.38	63.58	63.01	64.41	64.76
	5	74.8±1.86	76.35±2.27	75.73±2.01	75.2±2.37	78.13±2.01	76.91±2.44
	10	77.02±3.68	79.31±1.48	78.35±1.64	78.93±1.3	80.69±1.71	79.28±1.62
	20	78.91±1.44	82±1.07	80.86±1.04	81.14±1.32	82.66±1	81.71±1.31
wbp_ufal	0	26.64	24.55	28.62	27.21	27.96	30.18
	5	58±4.23	56.83±4.94	57.07±4.67	58.52±4.98	59.13±4.86	59.68±6.1
	10	64.72±1.72	63.34±4.41	64.51±3.43	65.94±3.88	65.2±4.03	66.32±3.63
	20	71.84±3.39	66.67±3.67	70.45±3.46	70.6±3.29	67.98±3.77	70.75±3.55
wo_wtb	0	34.79	33.05	34.72	34.11	34.09	35.27
	5	46.12±2.41	45.47±2.7	45.86±2.36	46.69±2.23	47.49±2.66	46.49±2.3
	10	50.01±2.03	48.49±1.69	49.13±2.18	49.69±1.79	50.97±2.1	49.67±2.1
	20	53.32±1.19	51.27±1.39	52.73±1.65	52.79±1.15	53.97±1.45	52.58±1.55
yo_ytb	0	41.46	47.34	45.31	45.59	50.45	49.1
	5	59.59±3.02	62.93±2.71	61.66±2.54	61.26±2.8	64.5±2.51	63.3±3.09
	10	63.34±	66.71±1.63	65.68±2	65.39±2.17	68.18±1.63	67.31±1.5
	20	67.23±1.06	69.14±1.19	69.45±1.01	68.56±1.43	70.9±1.1	69.58±1.25

Table 5: POS tagging results on all evaluation languages.

Dataset	k	J	B	MM	NP	MM+	NP+
Exact Match							
Arabic	0	48.97	49.29	51.47	51.36	49.4	48.64
	5	52.2±3.92	50.19±3.52	53.38±3.52	51.48±3.2	49.27±3.89	51.27±4.75
	10	54.51±2.47	52.81±2.93	54.96±2.93	53.67±2.16	52.05±3.27	53.67±3.43
	20	56±1.85	54.64±1.86	56.59±1.56	55.43±1.86	54.45±1.94	55.78±2.13
Bengali	0	45.13	46.02	51.33	44.25	45.13	51.33
	5	46.32±3.48	45.3±3.11	50.76±3.03	47.22±3.3	45±2.98	49.45±3.17
	10	47.22±3.15	46.44±3.08	50.83±2.94	49.39±3.84	45.98±3.7	50.01±3.22
	20	49.47±3.54	48.24±4.15	52.37±3.57	50.21±3.62	47.68±3.31	51.24±3.03
Finnish	0	42.33	43.61	47.95	49.36	47.83	46.42
	5	46.5±4.96	45.75±3.21	47.69±3.48	48.75±3.21	45.66±3.53	47.57±4.21
	10	48.56±2.65	47.25±2.81	49.43±2.78	50.28±3.1	46.85±2.77	48.55±3.1
	20	49.81±2.09	48.82±2.77	50.43±2.34	52.22±3.01	48.18±2.48	50.89±2.49
Korean	0	50	50.72	53.62	48.55	51.45	53.62
	5	51.37±2.52	49.5±2.76	51.87±2.11	49.52±2.48	49.57±2.35	52.17±2
	10	52.63±2.41	50.63±2.46	52.29±1.85	50.1±2.29	50.29±2.51	53±1.93
	20	54.07±2.11	51.88±2.15	53.55±1.91	51.87±2.03	51.71±2.13	53.67±2.16
Indonesian	0	56.46	51.86	54.87	56.28	52.74	56.28
	5	57.99±2.94	55.49±3.18	56.04±2.99	57.61±2.7	55.53±3.82	55.39±2.67
	10	59.4±2.49	57.11±2.81	58.54±2.49	58.59±1.96	57.08±2.84	56.86±1.95
	20	60.99±2.09	58.99±2.51	60.76±2.21	59.9±1.69	59.11±2.07	57.95±1.94
Russian	0	44.21	43.23	41.01	40.39	41.01	37.44
	5	49.45±4.36	47.41±3.92	46.66±4.01	46.83±4.34	46.2±4.61	44.09±5.38
	10	51.84±3.04	49.66±2.83	48.72±3.56	48.81±3.79	47.97±4.43	47.66±4.05
	20	53.6±2.45	50.72±2.55	51.05±2.8	51.5±2.75	50.47±2.45	50.25±2.96
Swahili	0	43.49	45.29	41.88	41.48	45.69	45.29
	5	46.47±5.11	49.07±4.31	48.9±4.88	47.6±4.21	48.8±4.28	47.32±4.3
	10	50.06±4.13	51.37±3.45	51.1±3.83	50.37±3.72	49.79±3.59	49.96±3.88
	20	54.02±3.06	53.82±2.63	53.94±2.54	52.16±2.89	52.51±3.47	52.65±3.26
Telugu	0	43.5	42.75	44.54	42	41.7	45.14
	5	45.97±2.85	44.58±3.44	45.33±3.91	44.89±3.44	41.87±5.35	42.92±5.36
	10	48.11±3.4	46.64±3.1	47.59±2.95	46.4±2.69	45.32±4.23	46.3±3.51
	20	50.1±2.55	49.08±2.42	49.21±2.77	48.8±1.97	47.11±2.71	48.88±2.91
F_1 score							
Arabic	0	65.57	67.38	66.59	67.44	64.98	65.45
	5	68.4±3.82	67.09±3.51	69.66±3.45	67.59±3.26	65.92±3.93	67.76±4.86
	10	70.56±2.47	69.55±2.88	71.35±2.83	69.82±2.15	68.82±3.64	70.28±3.63
	20	72.14±1.78	71.14±1.87	73.21±1.46	71.55±1.88	71.5±1.99	72.26±2.31
Bengali	0	57.24	62.57	66.29	59.64	60.28	62.86
	5	59.27±2.79	60.04±2.97	64.65±2.84	61.71±3.1	59.46±2.72	61.85±2.65
	10	59.88±2.7	60.64±2.86	64.88±2.71	63.52±3.38	60.03±3.28	62.33±2.89
	20	62.11±3.15	62.1±3.51	65.93±3.07	64.31±2.95	61.72±2.96	63.86±2.72
Finnish	0	61.85	63.57	61.72	63.79	62.12	61.64
	5	61.48±3.47	61.76±2.63	61.66±2.84	62.66±2.8	60.49±2.42	61.57±3.94
	10	62.98±1.53	62.48±2.03	63.26±2.31	64.14±2.7	61.58±2.15	62.58±2.91
	20	63.81±1.57	63.66±2.07	64.65±2.2	65.64±2.84	63±2.24	64.9±2.27
Korean	0	60.26	62.71	62.4	58.68	61.2	64.35
	5	61.31±2.47	60.82±2.47	61.67±2.17	59.31±2.34	59.27±2.33	62.13±2.01
	10	62.52±2.26	62.02±2.1	61.86±2.05	60.03±2.15	59.86±2.51	62.92±1.91
	20	64.04±2.01	63.08±1.87	63.43±1.89	61.84±1.97	61.52±1.89	63.55±1.95
Indonesian	0	69.96	65.99	70.05	70.82	68.02	70.19
	5	71.4±2.81	69.22±3.28	70.61±2.74	71.18±2.17	69.95±3.4	69.37±2.58
	10	72.69±2.27	70.79±2.78	72.79±2.53	72.23±1.77	71.33±2.46	70.93±1.96
	20	74.11±1.79	72.49±2.57	74.65±2.11	73.52±1.45	73.11±1.8	71.96±1.94
Russian	0	65.93	64.15	64.47	63.2	64.13	61.08
	5	66.96±1.52	65.11±1.5	65.03±1.39	65.13±1.33	64.58±1.45	62.17±1.61
	10	67.86±1.15	66.21±1.28	65.84±1.65	65.89±1.33	65.53±1.46	63.63±1.67
	20	68.7±1.01	66.85±1.38	66.94±1.45	67.1±1.38	66.4±1.19	65.01±1.82
Swahili	0	60.01	62.63	59.84	58.74	64.13	62.48
	5	60.21±4.38	62.7±3.37	62.48±3.72	61.9±3.41	63.43±3.38	61.81±4.06
	10	62.62±2.66	64.36±2.31	63.79±3.19	63.6±3.02	63.77±2.88	63.78±3.06
	20	65.18±2.09	65.89±2	66.27±1.99	65.48±1.7	66.21±2.19	66.12±2.11
Telugu	0	52.43	51.1	53.1	52.83	51.93	53.96
	5	60.99±5.15	59.6±6.05	59.21±6.17	61.27±5.05	57.91±6.64	57.21±7.33
	10	63.99±3.92	62.92±4.19	62.85±3.62	62.63±4.98	61.92±5.1	61.74±5.18
	20	65.96±2.37	65.29±2.15	64.53±3.06	65.63±1.61	63.67±2.58	64.9±3.26

Table 6: QA results on all evaluation languages.

On-the-Fly Attention Modulation for Neural Generation

Yue Dong[1]* **Chandra Bhagavatula**[2] **Ximing Lu**[2,4] **Jena D. Hwang**[2]
Antoine Bosselut[2,3] **Jackie Chi Kit Cheung**[1] **Yejin Choi**[2,4]

[1]Mila / McGill University [2]Allen Institute for Artificial Intelligence
[3] Stanford University [4]Paul G. Allen School of CSE, University of Washington
`{yue.dong2@mail, jcheung@cs}.mcgill.ca`
`{chandrab,ximinglu,jenah,antoineb,yejinc}@allenai.org`

Abstract

Despite considerable advancements with deep neural language models (LMs), neural text generation still suffers from *de*generation: the generated text is repetitive, generic, self-contradictory, and often lacks commonsense. Our analyses on sentence-level attention patterns in LMs reveal that neural degeneration may be associated with insufficient learning of task-specific characteristics by the attention mechanism. This finding motivates on-the-fly attention modulation[1]– a simple but effective method that enables the injection of priors into attention computation during inference. Automatic and human evaluation results on three text generation benchmarks demonstrate that attention modulation helps LMs generate text with enhanced fluency, creativity, and commonsense reasoning, in addition to significantly reduce sentence-level repetition.

1 Introduction

Neural text generation is critical for a wide range of downstream natural language applications. However, the standard approach – using a Transformer-based (Vaswani et al., 2017) language model (*e.g.*, Radford et al., 2019) with maximum likelihood fine-tuning and non-stochastic decoding – is known to exhibit *de*generation (Welleck et al., 2019). Despite being pre-trained on large amounts of data, text generated by neural models is observed to be repetitive, generic, self-contradictory, and lacking commonsense (Holtzman et al., 2020).

Many explanations have been proposed for neural text degeneration, including inappropriate training objectives (Welleck et al., 2019) and decoding discrepancies relative to human language (Holtzman et al., 2018, 2020). While the aforementioned may be factors for neural degeneration, we show that insufficient learning of *task-specific* characteristics – reflected in the self-attention mechanism in transformer blocks – is associated with neural text degeneration. We demonstrate that degeneration is alleviated if we inject priors through attention modulation (AttnM) during inference.

Figure 1: Example of fine-tuned GPT2-L outputs without (top) and with (bottom) attention modulation on αNLG. The task is to generate a plausible explanatory hypothesis H for observations O_1 and O_2. Our proposed attention modulation injects the task-specific prior – LMs should consider both observations relative equally – through balancing the sentence-level attention weights (Eqn. 5) in Transformer blocks during inference. Applying attention modulation with the aforementioned prior make sentence-level attentions from generation to observation pairs (O_1,O_2) more balanced[2], which are reflected in the sentence-level attention heatmaps of GPT2-L (darker = lower attention) across layers (y-axis) and heads (x-axis).

Self-attention – the ubiquitous component of Transformers – is task-agnostic with a large learning capacity for many NLP tasks (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020). It

---

*This work was done when the first author was an intern at AI2.

[2] attention ratios are normalized mean sentence-to-sentence attention from generation H to observations O_1 and O_2

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1261–1274
August 1–6, 2021.

learns the general characteristics of language processing through pre-training on large amounts of unlabeled data. For example, multiple analyses have suggested that attention patterns in pre-trained Transformers implicitly encode syntactic information (Raganato and Tiedemann, 2018; Michel et al., 2019; Vig and Belinkov, 2019). In sequence transduction tasks, these learned characteristics, embedded in attention, make pre-trained Transformers a powerful language model (Radford et al., 2019).

A final task-specific step is typically required for adapting a task-agnostic language model to perform the desired task[3]. However, these task-specific characteristics might not sufficiently coincide with general characteristics even after fine-tuning. For example, task-specific characteristics embedded in attention patterns – such as word alignments for machine translation – are often noisy and imperfect for generalization (Kobayashi et al., 2020).

We show that insufficient learning of task-specific characteristics, reflected in sentence-level attention patterns[4] often being out of focus, may be associated with neural text degeneration (§3). Based on this observation, we propose a simple attention modulation framework that can dynamically redistribute sentence-level attention weights by injecting task-specific priors in Transformer blocks for different downstream tasks (§4). Remarkably, in long-range narrative story generation, abductive reasoning generation and constrained commonsense text generation, both automatic and human evaluation have shown improved quality in fluency, dullness, repetition, and commonsense reasoning with attention modulation (§6).

2 Background

We briefly discuss how vanilla attention works, as well as Transformer architecture used in this paper.

Single-headed attention Given a sequence of d-dimensional input vectors $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, attention mechanism computes a set of weights based on a query vector $\boldsymbol{y}_i \in \mathbb{R}^d$:

$$\mathrm{Attn}(\boldsymbol{x}, \boldsymbol{y}_i) = (\alpha_{i,1}(\boldsymbol{x}, \boldsymbol{y}_i), \ldots, \alpha_{i,n}(\boldsymbol{x}, \boldsymbol{y}_i)) \quad (1)$$

where $\alpha_{i,j}$ is the attention weight that $\boldsymbol{y}_i$ pays to $\boldsymbol{x}_j$. One formulation of attention — scaled dot product attention — is computed as:

$$\alpha_{i,j} := \operatorname*{softmax}_{\boldsymbol{x}_j \in \boldsymbol{x}}\Big(\frac{\boldsymbol{q}(\boldsymbol{y}_i)\boldsymbol{k}(\boldsymbol{x}_j)^\top}{\sqrt{d}}\Big) \in \mathbb{R} \quad (2)$$

where query $\boldsymbol{q}(\cdot)$ and key $\boldsymbol{k}(\cdot)$ functions are linear transformations. In self attention, every $\boldsymbol{x}_i$ is used as the query vector ($\boldsymbol{y}_i$). An updated representation $\tilde{\boldsymbol{x}}_i$ is computed as a weighted sum of value vectors that are linearly transformed by $\boldsymbol{v}(\cdot)$:

$$\tilde{\boldsymbol{x}}_i = \sum_{\boldsymbol{x}_j \in \boldsymbol{x}} \alpha_{i,j} \boldsymbol{v}(\boldsymbol{x}_j). \quad (3)$$

Multi-head attention In *multi-headed* attention (MHA), N_h attention heads are computed independently to obtain the updated $\tilde{\boldsymbol{x}}_i$:

$$\tilde{\boldsymbol{x}}_i = W_o \mathop{\Big\Vert}_{h=1}^{N_h} \Big(\sum_{\boldsymbol{x}_j \in \boldsymbol{x}} \alpha_{i,j}^h \boldsymbol{v}^{\boldsymbol{h}}(\boldsymbol{x}_j) \Big). \quad (4)$$

$\alpha_{i,j}^h$ follows Eqn. 2 except the model dimension in each head h is often reduced to $d_h = \frac{d}{N_h}$. $\tilde{\boldsymbol{x}}_i$ is obtained by the concatenation of lower-rank representations from all heads and $W_o \in \mathbb{R}^{d \times h \cdot d_h}$.

GPT2-L GPT2 (Radford et al., 2019) is a family of Transformer-based language models (LMs) that follows the architecture of stacked decoder. As GPT2 follows a multi-layer and multi-headed setting, $\alpha_{i,j}$ is specific to a layer l and head h, noted as $\alpha_{i,j}^{l,h}$. We use the GPT2-L model that has 36 layers with 20 heads per layer (762M total parameters).

3 Neural text degeneration vs. attention

As researchers have sought to understand the internal mechanisms of Transformers, the attention patterns exhibited by these heads have drawn considerable study (Vig and Belinkov, 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019). We perform sentence-level attention analysis to explore whether aggregated attention patterns are associated with neural text degeneration.

3.1 Sentence-level attention

We first define the sentence-to-sentence attention of a language model $\mathcal{M}$ with L layers and H heads. Given two sentences p and g such that p precedes g,

---

[3] Brown et al. (2020) have shown that GPT3 greatly improves task-agnostic, few-shot performance, but still struggles on tasks with strong task-specific characteristics.

[4] We study the global context in the multi-sentence prompts and choose sentence-level attention (Eqn. 7) as the experiment unit, since sentences are linguistic units of complete meaning.

the mean $\bar{\alpha}_{g,p}^{l,h}$ and max $\hat{\alpha}_{g,p}^{l,h}$ sentence-to-sentence attentions from g to p for layer l and head h are:

$$\bar{\alpha}_{g,p}^{l,h} = \frac{\sum_{i=1}^{|g|}\sum_{j=1}^{|p|}\alpha_{i,j}^{l,h}(g_i,p_j)}{|g|\cdot|p|} \tag{5}$$

$$\hat{\alpha}_{g,p}^{l,h} = \max_{\substack{i\in\{1,\dots,|g|\}\\ j\in\{1,\dots,|p|\}}} \alpha_{i,j}^{l,h}(g_i,p_j). \tag{6}$$

The aggregated sentence-to-sentence attention over the Transformer architecture $\mathcal{M}$ is defined as:

$$\alpha_{g,p}^{\mathcal{M}} = \frac{\sum_{l=1}^{L}\sum_{h=1}^{H}\alpha_{g,p}^{l,h}}{L\cdot H} \tag{7}$$

where $\alpha \in \{\bar{\alpha},\hat{\alpha}\}$ computes either the mean or the max sentence-level attention over $\mathcal{M}$.

3.2 Is neural text degeneration related to attention patterns?

We conduct experiments to evaluate whether neural text degeneration is associated with sentence-level attention patterns. Empirical results on two types of neural degeneration that are easy to detect – repetition and lacking commonsense reasoning under constraints – reveal their association.

Repetition vs. attention One common form of neural text degeneration is sentence-level repetition (Welleck et al., 2019). This type of degeneration happens frequently in our experiment on ROCStories test set (§6.1): given a five-sentence prompt, 35.4% of the consecutive sentences from the next five greedily generated sentences by the fine-tuned GPT2-L are exact repetitions. We check whether sentence-level attention patterns behave differently when generating repeated or different consecutive sentences.

We inspect the attention behavior by measuring the change of sentence-level attention when generating two consecutive sentences. The generations of fine-tuned GPT2-L on ROCStories test set are separated into two subsets $\{\mathcal{D}_{\text{repeated}},\mathcal{D}_{\text{different}}\}$: in which consecutive sentences that are either repeated (*i.e.*, degenerate) or different. Given the fine-tuned GPT2-L language model $\mathcal{M}$, we measure the sentence-level attention change Δ on the prompt sentence $p_{j\in\{1,\dots,5\}}$ while generating the consecutive sentence pair $(g_i, g_{i+1}), i \in \{1,\dots,4\}$, aggregated over the subset $\mathcal{D} \in \{\mathcal{D}_{\text{different}},\mathcal{D}_{\text{repeated}}\}$:

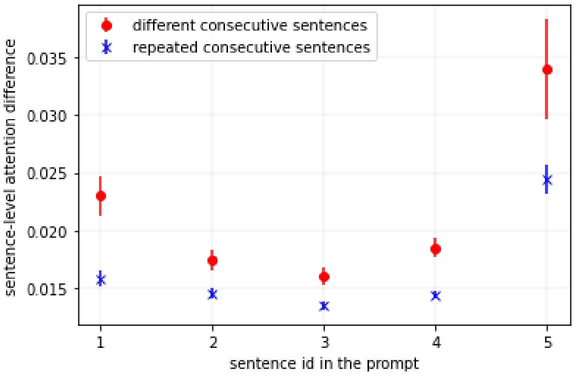

Figure 2: Mean sentence-level attention change of GPT2-L on ROCStories test set, while generating different (red) or repeated (blue) consecutive sentences.

$$\Delta(j,\mathcal{D}) = \sum_{d\in\mathcal{D}} |\bar{\alpha}^{\mathcal{M}}_{g^d_{i+1},p^d_j} - \bar{\alpha}^{\mathcal{M}}_{g^d_i,p^d_j}|/|\mathcal{D}| \tag{8}$$

where $\bar{\alpha}^{\mathcal{M}}_{g_i,p_j}$ is the mean sentence-level attention from sentence g_i to sentence p_j defined in Eqn. 7.

Figure 2 plots the aggregated mean sentence-level attention change over prompt sentences when GPT2-L generates repeated (red) or different (blue) consecutive sentences. The sentence-level attention changes are vastly lower on all prompt sentences when generating repeated consecutive sentences. Thus, sentence-level repetition may be correlated with the insufficient change of sentence-level attention. In §4 and §6, we show that generation quality can be vastly improved by injecting the prior – attention should look at the prompt differently when generating different sentences – through our proposed attention modulation.

Lack of commonsense reasoning vs. attention Text generated by neural language models is also observed to be lacking commonsense reasoning (Mao et al., 2019). We check whether this type of neural degeneration is associated with attention patterns. A benchmark dataset for generative commonsense reasoning – CommonGen (Lin et al., 2020) – is used as our test bed. CommonGen is designed for constrained commonsense reasoning: given a set of common concepts (e.g., use, tool, piece, metal); the task is to generate a coherent and plausible sentence covering all these concepts (e.g., "a piece of metal is used for making tools"). Covering the concepts in generation requires relational reasoning with background commonsense knowledge.

	agg. max attn.	SD	#
covered	0.434	0.0040	4515
uncovered	0.376	0.0068	1473

Table 1: Aggregated max sentence-level attention of the fine-tuned GPT2-L; the results are aggregated from the generation to covered or uncovered concepts on the CommonGen test set. agg. max attn., SD, # refer to aggregated max sentence-level attention, standard deviation, and the number of instances.

Each concept is represented as a prompt sentence in our experiments.[5] During generation, a concept (*e.g.*swim) is covered if its reflected form (*e.g.*{swim, swimming, swam, swum}) is generated in the CommonGen test set. We use a fine-tuned GPT2-L for the generation. Among the 5988 concepts in the prompt, about 75% of them are covered in the generation of GPT2-L. We can then easily separate sentence-level attention from the generation to the concept into two subsets: concept in the prompt that is covered or uncovered by the generated sentence.

Table 1 shows the results of max sentence-level attention (Eqn. 7) of the finetuned GPT2-L on the CommonGen test set[6]. We can observe that sentence-level attention from the generation to the concept is vastly higher when the concept is covered. Compared to that of uncovered concepts, the aggregated max sentence-level attention is 15.4% higher. Therefore, failing to generate a common concept through reasoning may be associated with insufficient attention to the concept.

In both cases, neural text degeneration is associated with insufficient attention to elements that are important for downstream generations. This motivates us to explore whether we can inject these priors in the language model by altering the attention mechanism to alleviate degeneration.

4 Method

This section describes our method – attention modulation – that can alleviate neural text degeneration. In §4.1, we describe the general attention modulation framework. In §4.2, §4.3, and §4.4, we discuss the priors injected through attention modulation for three different tasks: narrative story generation, abductive reasoning generation, and constrained commonsense reasoning.

4.1 Attention Modulation

Attention modulation aims to change the attention weights of a Transformer-based language model during inference, so that the generation can reflect priors that alleviate neural text degeneration. This additional signal is added to the self-attention computation in the Transformer blocks.

We reformulate the attention computation of Eqn. 2 by adding an attention reweighting function f, where priors can be injected. Given a sequence of input tokens $\boldsymbol{x}$, the self-attention from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ ($i \geq j$) while generating the t-token is reformulated to:

$$\alpha_{i,j}^t := \underset{\boldsymbol{x}_j \in \boldsymbol{x}}{\operatorname{softmax}}\Big(\frac{\boldsymbol{q}(\boldsymbol{x}_i)\boldsymbol{k}(\boldsymbol{x}_j)^\top}{\sqrt{d}} + f_{i,j}(\boldsymbol{x}, \boldsymbol{\alpha}^{t-1})\Big) \tag{9}$$

where $f(\boldsymbol{x}, \boldsymbol{\alpha}^{t-1})$ is the attention reweighting function and $\boldsymbol{\alpha}^{t-1}$ is the attention weight matrix for all layers and heads in the Transformer architecture at time step $t-1$. The attention reweighting function f can be either pre-defined or learned. In our experiments, we inject pre-defined sentence-level priors (heuristics) through f and show that this injection alleviates neural text degeneration. We leave the learning of better reweighting functions automatically to future work.

In the following sections, we describe sentence-level attention reweighting functions that are used for three different text generation tasks.

4.2 ROCStories: narrative generation

As shown in §3, sentence repetition in long-form generation may be associated with insufficient attention change while generating consecutive sentences. To amplify the attention changes, we can redistribute sentence-level attention with some priors while generating consecutive sentences.

We choose the prior that language model should consider long-range context during generation, as we observed that attention mostly focuses on the near history in many cases (Appendix A.1). Note this prior also increases the sentence-level attention change while generating consecutive sentences: the sentence-level attention for all previous sentences is always re-balanced based on the newly-generated sentence. To balance the attention of tokens in

---

[5] We can obtain a clear boundary for each concept with this design choice of adding a period as the separator, as concepts can be tokenized into multiple subwords by GPT2 tokenizers.

[6] We measure the max sentence-level attention rather than the mean sentence-level attention on CommonGen, as the attention to a concept is usually reduced once it is generated.

each sentence received while generating the next sentence, we define the attention reweight function in Eqn. 9 with the aforementioned prior for ROCStories as:

$$f_{i,j}(\boldsymbol{x}, \boldsymbol{\alpha}_{i,j}^{t-1}) = \frac{1}{\alpha_{g_i,p_j}^{t-1}}. \tag{10}$$

As later sentences in the prompt usually receive larger sentence-level attention weights (Appendix A.1), attention reweighting function defined in Eqn. 10 will add a large weight to tokens in the early sentences and a small weight to tokens in the late context sentences. The simple heuristic of balancing context sentences to be considered relatively equal, namely more weights on early context sentences, might not be optimal prior. However, it improves the long-form story generation in multiple measures, including fluency, interesting, newness, and repetition (§6.1).

4.3 αNLG: abductive reasoning generation

The second benchmark dataset we tested with attention modulation is αNLG (Bhagavatula et al., 2020). This dataset is proposed for abductive reasoning generation: given two observations O_1 and O_2, the model needs to generate a valid hypothesis h that explains what happened between the two observations. For example, given O_1: "Today was the first day of school." and O_2: "I hate school.", the task is to generate h such as "The teacher made fun of me." as a plausible explanation.

Bhagavatula et al. (2020) has shown that the fine-tuned GPT2 performs far below human performance on the αNLG task. We hypothesis that this may be associated with insufficient learning of sentence-level attention to both observations; for example, the model might over-fit to one of the observations for generation. Thus, we inject the prior – the language model should consider both observations relative equally – while generating a plausible explanation. This prior can be injected with attention reweighting function defined in Eqn. 10.

4.4 CommonGen: constrained commonsense generation

The third benchmark is CommonGen – a constrained text generation challenge for generative commonsense reasoning. CommonGen requires machines to generate a realistic sentence using *all* concepts from a given concept set by conducting commonsense reasoning over the relations among the given concepts. To successfully generate a plausible and grammatical sentence that follows the commonsense, models need to conduct commonsense reasoning over the relations among the given concepts. Our experiment in §3.2 shows that the fine-tuned GPT2-L can only cover about 75% of concepts during generation. We infer from Table 1 that this may be associated with GPT2-L giving insufficient sentence-level attention to uncovered concepts. Thus, we propose a simple heuristic – model should pay more attention to concepts that are not covered yet – to be injected with attention modulation.

Consider the prompt with m concepts $\boldsymbol{c} = \{c_1, \dots, c_m\}$ and a partially generated sentence $y_1, \dots, y_{t-1}$. While generating the t-th token, the sentence-level reweighting function from the i-th token to the j-th token in c_k is defined as:

$$f_{i,j}(\boldsymbol{x}) = \begin{cases} 1/m & \text{if} \quad c_k \subset y_{1:t-1} \\ 1 & \text{else} \end{cases} \tag{11}$$

Intuitively, if a concept c_k is covered in the partial generation, attention modulation with Eqn. 11 will reduce the attention weights of the tokens in concept c_k.

5 Experimental Setups

This section describes the experiment setups, including the baselines, decoding algorithms, datasets, and evaluation metrics.

Model architecture & baseline Attention modulation is architecture-agnostic and can be applied to any Transformer-based models that contain self-attention computation. We choose GPT2-L (Radford et al., 2019) for our experiments, which has achieved state-of-the-art performance on a variety of generation tasks (Vig and Belinkov, 2019). Attention modulation can be applied to any range of layers in the Transformer. To compare models with and without attention modulation on each of the three generation tasks, we use the best fine-tuned GPT2-L based on the validation set after fine-tuning for 4 epochs with the default settings.

Decoding Attention modulation directly changes the attention weights of the context tokens during inference. It is orthogonal to different decoding algorithms that change the searching strategies based on the softmax distribution emitted by Transform-

	next 1 sent.		next 2 sent.		next 3 sent.		next 4 sent.		next 5 sent.		
	uniq.	% rel.	uniq.	% rel.	uniq.	% rel.	uniq.	% rel.	uniq.↑	% rel.↑	% rep.↓
w/o AttnM	9.08k	48.59	13.58k	48.78	15.71k	49.20	16.01k	49.97	17.03k	50.52	35.43
w/ AttnM (ours)	**10.39k**	**52.92**	**15.12k**	**54.23**	**18.33k**	**55.35**	**20.41k**	**55.78**	**22.69k**	**55.78**	**17.49**
ratio	1.14		1.12		1.16		1.24		1.34		

Table 2: Test results of the fine-tuned GPT2-L w/ and w/o attention modulation on ROCstories with the greedy decoding algorithm. uniq. represent the unique number of tokens generated in the whole test corpus, which measures the number of new unique tokens generated. rel. represent relevancy, which measures the percentage of tokens generated appears in the prompt. rep. measures the sentence-level repetition – whether two sentences generated are identical.

ers.[7] We present the results with non-stochastic decoding algorithms (*i.e.* greedy decoding and beam search), as generations based on them truly reflect the token-level probabilities predicted by the model (Holtzman et al., 2018).

Datasets We use three different generation datasets – ROCStories (Mostafazadeh et al., 2016), αNLG (Bhagavatula et al., 2020), and CommonGen (Lin et al., 2020). For ROCStories, we used the 2017 version and split the data into 75/10/15 for train/val/test.

Evaluation On ROCStories, we measure dullness, relevancy and repetition similar to Welleck et al. (2019). We report the number of unique tokens generated, where the generation is less dull if more unique tokens are generated. For repetition, we directly measure sentence-level repetition: two generated sentences are repeated if their strings are the same. For relevancy, we measure the percentage of generated tokens that appear in the prompt. Besides, we perform a human evaluation, where three annotators are asked to rate the generations based on fluency, interestingness, newness, relevancy, and repetition.

On αNLG, we score the generated explanation with respect to the reference using the following automatic metrics: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and CIDEr (Vedantam et al., 2015). In addition, we ask annotators to compare the generated explanations without and with attention modulation. Human judges are asked to decide which system provides a more plausible explanation of the observations.

On CommonGen, we report SPICE (Anderson et al., 2016) – a measure that evaluates semantic propositional content, in addtion to BLEU, ROUGE, METEOR, CIDEr. We also report Coverage (Lin et al., 2020), which computes the average percentage of input concepts that appear in the lemmatized outputs. We conduct a human evaluation following the protocol of Lu et al. (2020). Human judges are asked to compare two systems in terms of fluency, coverage (covers the concept), and overall quality (covers the concepts and follows commonsense).

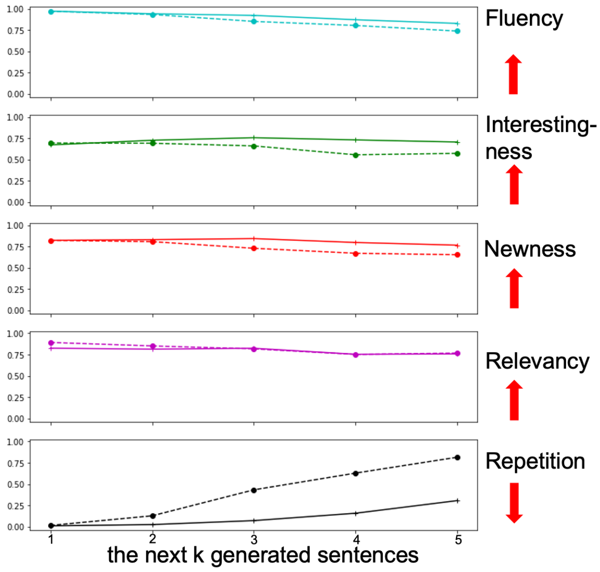

Figure 3: Human evaluation results on the next 1 to 5 sentences generated without (dashed lines) and with (solid lines) attention modulation (1000 samples).

6 Result

In this section, we present the vast improvements of the fine-tuned GPT2-L with attention modulation on three narrative generation and generative reasoning tasks: ROCStories, αNLG, and CommonGen.

6.1 ROCstories

Table 2 indicates that attention modulation significantly reduces repetition in narrative generation,

---

[7] These search-based decoding algorithms do not resolve the poorly generated token-level probabilities.

		R-2	R-L	B-3	B-4	Meteor	CIDEr	SPICE	Coverage
greedy	w/o AttnM	14.06	34.13	26.19	17.92	25.82	10.81	22.14	76.50
	w/ AttnM (ours)	**14.71**	**35.15**	**27.53**	**18.91**	**26.23**	**11.61**	**23.22**	**79.18**
beam=5	w/o AttnM	16.68	36.92	32.39	23.36	26.87	12.24	22.83	76.99
	w/ AttnM (ours)	**17.14**	**38.23**	**33.92**	**24.03**	**27.48**	**12.88**	**24.24**	**81.27**
beam=10	w/o AttnM	17.25	37.37	33.81	24.39	27.51	12.58	23.24	78.68
	w/ AttnM (ours)	**17.59**	**38.71**	**35.72**	**25.93**	**27.71**	**13.32**	**24.36**	**81.24**
beam=20	Lin et al. (2020)	16.85	39.01	33.92	23.73	26.83	12.19	23.57	79.09
	w/o AttnM	17.98	38.07	35.14	25.61	27.63	12.90	23.28	79.62
	w/ AttnM (ours)	**18.11**	**39.32**	**36.69**	**26.80**	**28.02**	**13.71**	**23.94**	**81.85**

Table 3: CommonGen test results of the fine-tuned GPT2-L w/ or w/o attention modulation based on different decoding algorithms.

	B-4	R-L	Meteor	CIDEr	Human
w/o AttnM	13.51	**18.29**	13.18	47.69	14%
w/ AttnM (ours)	**13.52**	18.01	13.18	**48.20**	**33%**

Table 4: Evaluations of the fine-tune GPT2-L on αNLG using greedy decoding.

while increasing the relevancy of generated sentences to the original story. We can observe a vast improvement in the number of unique generated tokens using attention modulation, indicating a reduced repetition rate (confirmed by the % number of repeated sentences in the next five sentence generated – 35.43 vs. 17.49 for our approach). This intuition is confirmed by our human evaluation in Figure 3, where the GPT2-L with attention modulation produces sentences that are more fluent, more interesting, more novel, and less repetitive than the original decoder. Furthermore, we note that the difference in performance across these evaluation categories generally increases as the number of generated sentences increases, indicating less sensitivity to long-form degeneration.

6.2 αNLG

Table 4 presents the automatic and human evaluation results on αNLG. We can see that our model performs similarly with and without attention modulation in terms of automatic evaluation. However, our human evaluation results in the last column show that overall, the human judges prefer the explanations produced using attention modulation significantly more than those of the original model. With 100 samples generated, 33% of the time, human judges select explanations generated with attention modulation as more plausible. In contrast, explanations from the original model are only preferred 14% of the time.

6.3 CommonGen

Table 3 shows the automatic evaluation results on the CommonGen dataset. We separate different settings of decoding algorithms in blocks. By injecting the prior – the model should put more attention on uncovered concepts – into the GPT2-L with attention modulation, we can improve the text generated in every automatic measure significantly. Interestingly, despite our attention-reweighted decoder only encouraging coverage, we see all the other measures such as ROUGE, BLEU, METEOR, CIDEr, SPICE improve, as well.

These improvements also hold when we use a different base decoding algorithm, such as beam search. Again, the performance improvement for using attention modulation is significant over all measures. Thus, unlike decoding algorithms that improve downstream tasks through truncation of the sampling distribution, we directly re-calibrate the token-level probabilities predicted by the model by altering attention patterns in the Transformer blocks during inference.

We also conduct a human evaluation to check whether this improvement in the automatic metrics transfers to human judgments. In Table 6, we see that our attention modulation algorithm significantly outperforms the original inference model on every measure – from fluency, quality, and overall performance.

6.4 Vast improvements on few-shot learning

Table 5 presents the results of attention modulation on GPT2-L that are fine-tuned on different numbers of training examples from CommonGen. We observe the improvements on all measures are

Training size	Method	R-2	R-L	B-3	B-4	Meteor	CIDEr	SPICE	Coverage	
10	w/o AttnM	2.15	16.07	6.39	3.63	10.02	1.23	5.07	27.12	
	w/ AttnM (ours)	**3.38**	**18.89**	**7.24**	**3.48**	**12.47**	**1.63**	**7.21**	**36.55**	↑ 9.43
1000	w/o AttnM	7.61	27.03	14.01	7.38	20.67	6.40	16.79	62.33	
	w/ AttnM (ours)	**8.54**	**27.97**	**15.51**	**8.78**	**22.13**	**6.68**	**18.02**	**69.31**	↑ 6.98
10000	w/o AttnM	10.70	30.06	17.40	10.02	23.40	7.15	20.20	73.39	
	w/ AttnM (ours)	**11.53**	**30.70**	**18.43**	**11.12**	**24.42**	**7.29**	**21.33**	**78.74**	↑ 5.35
full (∼39k)	w/o AttnM	14.06	34.13	26.19	17.92	25.82	10.81	22.14	76.50	
	w/ AttnM (ours)	**14.71**	**35.15**	**27.53**	**18.91**	**26.23**	**11.61**	**23.22**	**79.18**	↑ 2.68

Table 5: CommonGen test results of the fine-tuned GPT2-L w/ or w/o attention modulation trained on different size of training examples (greedy decoding).

	Fluency	Quality	Overall
w/o AttnM	85.07	39.30	44.77
w/ AttnM (ours)	**89.55**	**48.76**	**52.73**

Table 6: Human evaluations of the fine-tuned GP2-L w/o or w/ attention modulation on CommonGen.

more prominent when the fine-tuning data size is small. For example, adding attention modulation can improve coverage by 9.43% on the GPT2-L fine-tuned with only 10 examples. This not only validates that priors we injected into the model are suitable for improving the downstream task performance, but also shed lights to use attention modulation on different few-shot learning scenarios where the number of training examples is limited.

7 Related Work

We propose to use attention modulation to heuristically re-balance sentence-level attention for neural text degeneration. At least three domains of work are closely related to our proposal, namely, attention pattern analysis, work that focuses on changing or approximating learned attention patterns, and work for countering neural text degeneration.

Attention analysis: Previous work has investigated the attention patterns within the local context of a sentence. These works highlighted that attention patterns in Transformers implicitly encode syntactic information such as dependency relations (Htut et al., 2019), and part-of-speech tags (Vig and Belinkov, 2019; Raganato and Tiedemann, 2018). Other works observed that attention patterns can provide explanations (Wiegreffe and Pinter, 2019) or coarse word alignments in machine translation (Zenkel et al., 2019; Kobayashi et al., 2020). In contrast to these works, we analyze sentence-level attention patterns for neural text degeneration, and propose to directly modify the attention computation to reduce it.

Alternative attention: Many works have been proposed to change attention mechanisms to optimize their O(n^2) complexity. Some promising directions in this space include sparse attention mechanisms (Beltagy et al., 2020; Zaheer et al., 2020) and linearized attention (Choromanski et al., 2021). These alternative attention mechanisms require training the model and are used as replacements to the original attention mechanism for fast training or reduced computation. Our work is fundamentally different as we seek to inject priors into the standard attention mechanism during inference (without re-training the model).

Neural text degeneration: Previous works seek to solve neural text degeneration by changing the training objective to reduce the likelihood of common tokens (Welleck et al., 2019), or modifying the decoding algorithm by truncating the sampling distribution (Holtzman et al., 2018, 2020). Specifically, Welleck et al. (2019) introduce an additional training loss that reduces the likelihood of common tokens. Holtzman et al. (2018, 2020) propose stochastic decoding algorithms with truncation of the sampling distribution. Our work is orthogonal to these methods by injecting priors into the model's attention computation during inference.

8 Conclusions and future work

Neural language models often exhibit degeneration: the output texts are repeated, bland, and inconsistent. Our empirical analyses show that neural text degeneration may be associated with insufficient learning of task-specific characteristics by the attention mechanism. We propose a simple but effective module – attention modulation – that can inject priors for better generation through re-balancing the attention weights during inference. Results on three different narrative and commonsense generation tasks indicate that attention modulation can reduce repetition and enhance commonsense reasoning while maintaining fluency and coherence.

Acknowledgements

This research was supported in part by NSF (IIS-1714566), DARPA MCS program through NIWC Pacific (N66001-19-2-4031), the Canada CIFAR AI Chair program, the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Allen Institute for AI. Computations on `beaker.org` were supported in part by credits from Google Cloud.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

S. Banerjee and A. Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *International Conference on Learn-ing Representations*.

T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, T. Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2021. Rethinking attention with performers.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the As-sociation for Computational Linguistics (ACL)*.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. *arXiv preprint arXiv:2010.12884*.

Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley, and Garrison Cottrell. 2019. Improving neural story generation by targeted common sense grounding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5990–5995.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Kishore Papineni, S. Roukos, T. Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the As-sociation for Computational Linguistics (ACL)*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog 1.8*.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ramakrishna Vedantam, C. L. Zitnick, and D. Parikh. 2015. Cider: Consensus-based image description evaluation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*.

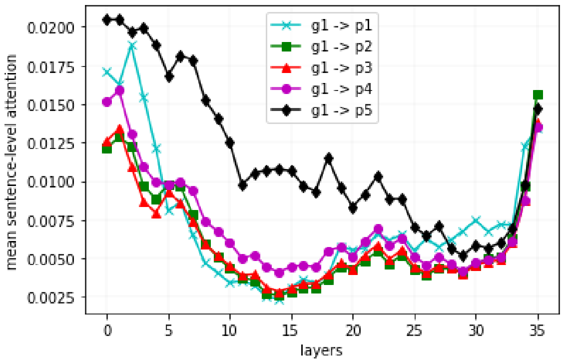

Figure 4: Sentence-level attention distribution across different layers in GPT2-L. The result is aggregated by computing the mean sentence-level attention from the next generated sentence to the five sentences in the prompt of ROCStories development set. Lower number represents lower layers in the Transformer.

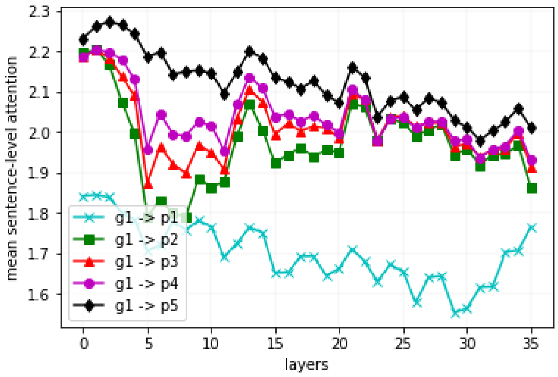

Figure 5: Attention entropy of each sentence in the prompt aggregated over the ROCStories dev. set. In the first sentences, there are particularly high-entropy attention heads that produce bag-of-vector-like representations.

A Appendices

A.1 How does a language model use attention to model a multi-sentence prompt?

Sentence-level attention portion To reveal which part of context – near or distant history – are important for context representation, we compute aggregated mean sentence-level attention (Eqn.5 in the main text) each prompt sentence p_i received, while generating the sentence g_1 after the prompt. We observe from Figure 4 that GPT2-L mostly attends to the nearest sentence (p_5) during the generation. This effect is especially prominent in the early and middle layers. In the late layers, the attention from different sentences evens out. This observation is consistent with previous analysis of attention patterns within sentences such that deeper layers focus on longer-range context (Vig and Belinkov, 2019).

	train	dev.	test
ROCStories	39498	5269	7899
αNLG	169,654	1,532	3,059
CommonGen	67,389	4,018	7,644

Table 7: Dataset Statistics

Sentence-level attention entropy Khandelwal et al. (2018) observed that LSTM represent distant context as topics; only a few token in the distant context are used to compute the context representation. We check whether this observation also holds on Transformer-based models by computing attention entropy. This sentence-level attention entropy of p_i based on the attention from g_1 to p_i at layer l_m over a corpus X is defined as:

$$E_A(g_1, p_i, l_m) = -\frac{\sum\limits_{x\in X}\sum\limits_{h\in l_m}\sum\limits_{j=1}^{|p_i|}\sum\limits_{k=1}^{|g_1|}\alpha^h_{j,k}\log(\alpha^h_{j,k})}{|X|\cdot|H|\cdot|p_i|\cdot|g_1|} \tag{12}$$

where h is a head in layer l_m and $\alpha^h_{j,k}$ is the attention weight from $x_j \in p_i$ to $x_k \in g_1$ for h. Figure 5 shows a clear separation of entropy over different sentences in the prompt, where more distant sentences have lower entropy values. This suggests that LMs only modelling distant sentences as topics – attention over key words being a proxy.

A.2 Hyperparameters

Attention modulation can be applied to any layer and any head in the Transformer based on our implementation. However, the weights learned by different heads in a particular layer have a large variance (Vig and Belinkov, 2019) and are subject to change from different training sessions. Therefore, we only reweight attention on all heads in different layers, where what layers are re-weighted are hyperparameters. We choose to reweight the consecutive layers from a starting layer l_s to an end layer l_e and performed a grid search on different layer ranges. For the start layer, we experimented with $l_s \in \{0, 4, 8, 12, 16, 20, 24, 28, 32\}$; for the end layer, we experimented with $l_e \in \{4, 8, 12, 16, 20, 24, 28, 32, 36\}$. The reweighting layers are chosen based on the validation set performance. On ROCstories, the GPT2-L are re-weighted with $l_s = 8$ and $l_e = 32$; On αNLG, the

propmt	attention reweight	Generated sentence with attention-decoding
field. stand. look. =	(1,3,2)	A man stands looking at a sign in a field.
field. stand. look. =	(1,2,3)	He looks up and sees a group of people standing in the field.
field. stand. look. =	(2,3,1)	He stands in the middle of the field, looking down at the stands.

Table 8: An example of attention modulation with different attention reweighting functions on CommonGen dev set. Only by redistributing the sentence-level attention during inference, we can generate sentences following the desired order specified in the attention reweighting function.

		R-2	R-L	B-3	B-4	Meteor	CIDEr	SPICE	Coverage
beam=20	w/o AttnM	17.98	38.07	35.14	25.61	27.63	12.90	23.28	79.62
	w/ AttnM	18.11	39.32	36.69	26.80	28.02	13.71	23.94	81.85
	w/ AttnM$_{pm}$	**19.78**	**41.17**	**38.28**	**28.36**	**30.97**	**15.38**	**28.24**	**91.89**

Table 9: Results on CommonGen test set with beam search and permutations defined in A.3.

GPT2-L are re-weighted with $l_s = 12$ and $l_e = 32$; On CommonGen, the GPT2-L are re-weighted with $l_s = 24$ and $l_e = 32$.

A.3 Attention modulation and generation order

CommonGen provides the concept set in a random order, where models need to perform a relational commonsense reasoning to find the optimal order of them for generating a plausible sentence. We found that attention modulation provide signals for generation order given different reweighting functions (examples in Table 8). In this experiment, we guide the generation order by providing different initialization weights in the reweighting functions. We enforce different attention modulation weights based on the order we want the concepts to be generated. For examples, row 1 in table 8 means the concepts of (FIELD, STAND, LOOK) are initialized to be re-weighted by scales of (1,3,2).

This interesting finding motivates us to conduct a permutation experiment on CommonGen. For a k concept set, we initialize the attention modulation weights based on the permutations of 1 to k ($k!$ permutations in total) and generate $k!$ sentences with attention modulation. We then select the generation that covers the most concepts[8] from these $k!$ generations as output. We call this method "attention modulation with permutation". Table 9 presents the results of attention modulation with permutation. We see that just by enforcing the order in attention modulation, the coverage can be improved by another 10%.

A.4 Human evaluation details

Figure 6 and 7 show the evaluation templates for tuckers for Rocstories and CommonGen[9], respectively. On ROCstories, the inter-annotator agreements is 0.743 and fleiss kappa is 0.229 averaged over the five dimensions on 200 × 5 samples. The inter-annotator agreements are 84.6%, 72.2%, 76.1%, 78.5%, and 59.9% for fluency, interestingness, newness, relevance, and repetition. On αNLG, the human evaluation results are 33%, 14%, 10%, 32%, 11% for preferring outputs with attention modulation, without attention modulation, both, neither and no agreement. The inter-annotator agreements is 0.835 and kappa is 0.439 over 100 samples. On CommonGen, the inter-annotator agreements are 77.94%, 71.64%, and 67.50% for fluency, quality, and overall on 200 samples.

---

[8] If there is a tie, we choose the shorter generation.

[9] omitted αNLG template due to space limitation.

prompt	Jen was always afraid of flying. She needed to fly to Japan for an important meeting. Jen took some Dramamine and boarded her plane. Jen dozed off shortly after the plane took off.
w/o AttnM	Jen woke up to a huge headache. She had to cancel her trip to Japan. Jen was glad she had taken Dramamine. She was able to fly safely. Jen was glad she had taken Dramamine.
w/ AttnM	She woke up to a loud bang and felt sick. She had to sit in the back of the plane. She was so scared she couldn't even get out of her seat. Jen was so embarrassed she never flew again. She was so scared she never flew again.

Table 10: A generation example (greedy decoding) without and with attention modulation on ROCStories. Adding attention modulation during inference **improves the commonsense reasoning and consistency of the generation**.

prompt	Neil had been journeying through Asia. But he contracted malaria on a Thai island, and had to be flown home for treatment.
w/o AttnM	He was so sick that he couldn't go to the hospital. He was so sick that he couldn't go to the hospital. He was so sick that he couldn't go to the hospital. He was so sick that he couldn't go to the hospital. He was so sick that he couldn't go to the hospital.
w/ AttnM	He was very sick and had to be hospitalized. He was in the hospital for a week. He recovered and was released. Neil was very happy to be home. He was able to get better and was able to go back to his home country.

Table 11: Example (greedy decoding) of the fine-tuned GPT2-L without and with attention modulation on story completion. Adding attention modulation during inference significantly **reduces the sentence-level repetition**.

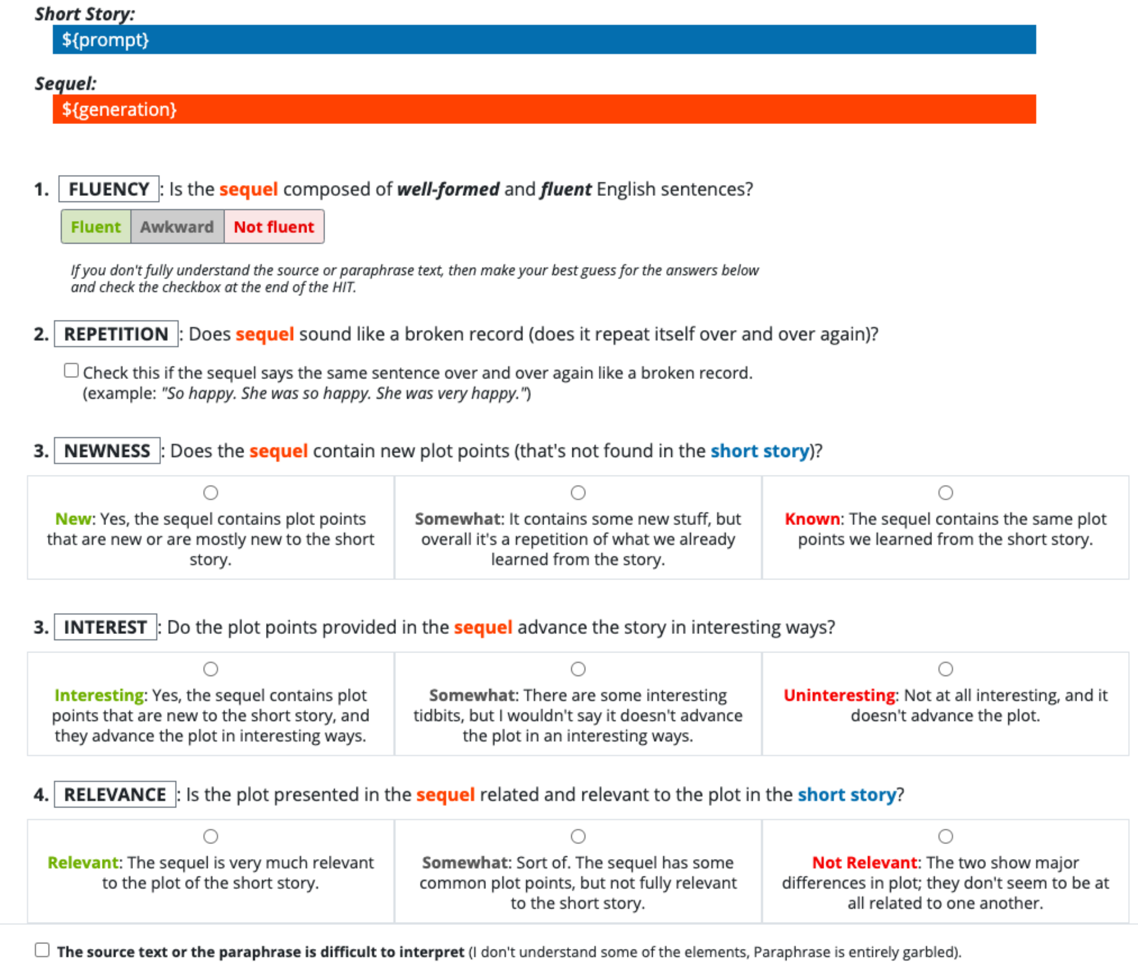

Short Story:

${prompt}

Sequel:

${generation}

1. **FLUENCY**: Is the **sequel** composed of ***well-formed*** and ***fluent*** English sentences?

Fluent | Awkward | Not fluent

If you don't fully understand the source or paraphrase text, then make your best guess for the answers below and check the checkbox at the end of the HIT.

2. **REPETITION**: Does **sequel** sound like a broken record (does it repeat itself over and over again)?

☐ Check this if the sequel says the same sentence over and over again like a broken record. (example: *"So happy. She was so happy. She was very happy."*)

3. **NEWNESS**: Does the **sequel** contain new plot points (that's not found in the **short story**)?

○ **New**: Yes, the sequel contains plot points that are new or are mostly new to the short story.	○ **Somewhat**: It contains some new stuff, but overall it's a repetition of what we already learned from the story.	○ **Known**: The sequel contains the same plot points we learned from the short story.

3. **INTEREST**: Do the plot points provided in the **sequel** advance the story in interesting ways?

○ **Interesting**: Yes, the sequel contains plot points that are new to the short story, and they advance the plot in interesting ways.	○ **Somewhat**: There are some interesting tidbits, but I wouldn't say it doesn't advance the plot in an interesting ways.	○ **Uninteresting**: Not at all interesting, and it doesn't advance the plot.

4. **RELEVANCE**: Is the plot presented in the **sequel** related and relevant to the plot in the **short story**?

○ **Relevant**: The sequel is very much relevant to the plot of the short story.	○ **Somewhat**: Sort of. The sequel has some common plot points, but not fully relevant to the short story.	○ **Not Relevant**: The two show major differences in plot; they don't seem to be at all related to one another.

☐ **The source text or the paraphrase is difficult to interpret** (I don't understand some of the elements, Paraphrase is entirely garbled).

Figure 6: Mechanical Turk template used to evaluate ROCstories generations.

	propmt	Generated sentence with attention-decoding
w/o AttnM	run. team. field. drill. =	person runs a drill during a practice at training camp.
w/ AttnM.	run. team. field. drill. =	person runs a drill during a training session with his team.
w/o AttnM	use. tool. piece. metal. =	tool or piece of metal used in manufacturing.
w/ AttnM	use. tool. piece. metal. =	piece of metal used to make tools.

Table 12: Examples produced by GPT2-L without and with attention modulation. Use attention modulation would have higher concept coverage (details in Table 3 in the main text with 5% coverage improvements on all decoding algorithms we tested);

Instructions (click to expand/collapse)

Thanks for participating in this HIT! Please read the instructions carefully.

In this HIT, you will be shown a **sentence** that tries to describe a plausible scenario by combining as many given **concepts** as possible.
Your task is to ***evaluate*** the **sentence** along two dimensions:

1. Is the **sentence** ***understandable*** and describes a ***plausible*** scenario?
   - **Yes**: The sentence is understandable and describes a realistic or possible scenario.
   - **No**: The sentence is either not understandable, or describes a completely impossible scenario.
2. Does the **sentence** include the given **concepts**?
   - **Yes**: The sentence meaningfully includes all of the concepts.
   - **Somewhat**: The sentence meaningfully includes some, but not all of the concepts. It may include all, but with many not properly incorporated.
   - **No**: The sentence includes few or none of the concepts in a meaningful way.
3. Considering your answers to 1. and 2., Does the **sentence** combine all of the **concepts** into a reasonably well-formed and plausible scenario?
   - **Yes**: The sentence is reasonably well-formed/understandable, and combines all the concepts into a plausible scenario.
   - **No**: The sentence is not well-formed/understandable, or fails to properly combine the concepts into a scenario.

Please take care to not submit responses that are uninformed by the instructions.

Concepts:
${source}

Sentence:
${generation}

1. **SENTENCE QUALITY**: Is the **sentence** ***well-formed*** and describes a ***plausible*** scenario?

○ **Yes**: The sentence is understandable and describes a realistic or possible scenario.
○ **No**: The sentence is either not understandable, or describes a completely impossible scenario.

2. **CONCEPTS**: Does the **sentence** include the given **concepts**?

○ **Yes**: The sentence meaningfully includes all of the concepts.
○ **Somewhat**: The sentence meaningfully includes some, but not all of the concepts. It may include all, but with many not properly incorporated.
○ **No**: The sentence includes few or none of the concepts in a meaningful way.

3. **OVERALL**: Considering your answers to 1. and 2., Does the **sentence** combine all of the **concepts** into a well-formed and plausible scenario?

○ **Yes**: The sentence is reasonably well-formed/understandable, and combines the concepts into a plausible scenario.
○ **No**: The sentence is not well-formed/understandable, or fails to properly combine the concepts into a scenario.

Figure 7: Mechanical Turk template used to evaluate CommonGen generations.

Grammar-Constrained Neural Semantic Parsing with LR Parsers

Artur Baranowski
Technische Hochschule Köln
Institut für Nachrichtentechnik

German Aerospace Center (DLR)
Institute for Software Technology

artur.baranowski@mailbox.org

Nico Hochgeschwender
Hochschule Bonn-Rhein-Sieg
Department of Computer Science

German Aerospace Center (DLR)
Institute for Software Technology

nico.hochgeschwender@h-brs.de

Abstract

Target meaning representations for semantic parsing tasks are often based on programming or query languages, such as SQL, and can be formalized by a context-free grammar. Assuming a priori knowledge of the target domain, such grammars can be exploited to enforce syntactical constraints when predicting logical forms. To that end, we assess how syntactical parsers can be integrated into modern encoder-decoder frameworks. Specifically, we implement an attentional SEQ2SEQ model that uses an LR parser to maintain syntactically valid sequences throughout the decoding procedure. Compared to other approaches to grammar-guided decoding that modify the underlying neural network architecture or attempt to derive full parse trees, our approach is conceptually simpler, adds less computational overhead during inference and integrates seamlessly with current SEQ2SEQ frameworks. We present preliminary evaluation results against a recurrent SEQ2SEQ baseline on GEOQUERY and ATIS and demonstrate improved performance while enforcing grammatical constraints.

1 Introduction

Semantic parsing aims at delivering granular, structured representations of natural language utterances, referred to as *meaning representations* or *logical forms*. Thus, it goes beyond shallow semantic analysis involving argument identification and role labeling (Collobert et al., 2011; Roth and Lapata, 2016). Meaning representations based on programming or query languages (PYTHON, SQL) are describable by (deterministic) context-free grammars and used for general purpose code generation (Xiao et al., 2016; Yin and Neubig, 2017). Our work targets this particular subset of logical forms. A context-free grammar may be exploited to constrain a semantic parser to only produce token sequences derivable from the grammar. Specifically, we investigate how syntax constraints can be enforced in semantic parsers based on modern encoder-decoder frameworks in a non-intrusive, computationally inexpensive way at inference time. We show that enforcing grammatical constraints with LR parsers is particularly well suited for modern autoregressive neural network architectures used in neural machine translation (Sutskever et al., 2014; Vaswani et al., 2017). We do not require any modifications to standard SEQ2SEQ neural network architectures and make very little assumptions about the inputs and outputs of such models. In contrast, most grammar-constrained decoders attempt to model the grammar explicitly within the neural network, complicating the architecture. Moreover, they predict complete syntax trees or derivation sequences. Our approach predicts source code token streams, preserving syntactic validity throughout the decoding procedure. Enforcing syntactical constraints relieves neural networks models from having to learn the syntactic structure of the target language, which is particularly beneficial for ensuring balanced expressions over long ranges (Bahdanau et al., 2014; Ling et al., 2016). Also, when integrating our models into larger application environments, we may want to preclude specific failure modes (i.e., syntax errors) when executing the generated program snippets to increase robustness. Preliminary evaluation results on the GEOQUERY and ATIS data sets demonstrate that simply enforcing syntactical constraints on the pre-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1275–1279
August 1–6, 2021. ©2021 Association for Computational Linguistics

dicted lexical tokens at inference time improves the performance of the semantic parser against a recurrent SEQ2SEQ baseline.

2 Related Work

Enforcing grammatical constraints within neural network models has sparked a fair amount of research interest. (Xiao et al., 2016) take a derivational viewpoint when decoding derivation trees, demonstrating improved performance when accounting for grammatical constraints. They predict leftmost derivation sequences, each uniquely associated with a corresponding derivation tree. Employing a constrained loss over probabilities $p'(\hat{\mathbf{y}}_t)$, where $\hat{\mathbf{y}}_t$ are the permissible continuations of a derivation sequence, constraints are enforced at training time. We take inspiration from (Xiao et al., 2016), however, enforce grammatical constraints at inference time and on lexical token streams in a bottom-up fashion, eliminating the need to derive entire syntax trees and effectively reducing the sequence length. Similarly, (Yin and Neubig, 2017) predict entire syntax trees sequentially using a SEQ2SEQ model, starting from the root node and generating tree nodes in depth-first, left-to-right order, deterministically converting them to the corresponding surface code. They define a dedicated grammar model that predicts action sequences that either apply a production rule or generate a lexical token. (Krishnamurthy et al., 2017) additionally ensure that decoder predictions satisfy type constraints by providing a type-constrained grammar. (Rabinovich et al., 2017) propose a decoder that employs a separate neural network module for each construct in the grammar. The decoder generates an abstract syntax tree (AST) through mutual recursion between modules. At each decoding step, the decoder either generates a symbol or propagates the decoder state to the next module. (Yin and Neubig, 2018) developed a transition-based abstract syntax parser (TRANX) guided by a grammar specified under ASDL (Wang et al., 1997). TRANX uses ASTs as general-purpose intermediate meaning representations, decoupling the semantic parsing procedure from domain-specific grammars. A user-defined grammar converts ASTs to domain-specific meaning representations. Similar to (Yin and Neubig, 2017) an AST is generated using a sequence of tree-constructing actions. All approaches enforce syntactical constraints by first predicting the tree-structured syntax tree top-down. Instead, we propose to directly generate lexical tokens (the values of syntax tree leaves) and constrain the decoding process by means of an bottom-up LR parser.

3 Problem Statement

Informally, we aim at translating a set of natural language utterances X to a structured representation of their meaning. We assume the syntax of target meaning representations is describable by a deterministic context-free grammar and that it is known at training time. Given a grammar G, our goal is to enforce the constraints imposed by G during decoding. That is, the image of our model f shall be the language generated by G.

$$f\colon\ X \rightarrow L(G) \tag{1}$$

We achieve this by means of a recurrent encoder-decoder model as proposed by (Sutskever et al., 2014) and an LR parser. We briefly introduce recurrent encoder-decoder NMT models and the specifics of our model.

3.1 SEQ2SEQ Model

All modern encoder-decoder frameworks define a probability distribution $P(\mathbf{y}|\mathbf{x})$ (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014), where in our case, $\mathbf{x}$ represents a natural language input. For a target source code string, $\mathbf{y}$ represents the token stream generated by a lexical analyzer and a corresponding lexical grammar (see section 3.2). $P(\mathbf{y}|\mathbf{x})$ is factorized as:

$$p(y_1, ..., y_\kappa|\mathbf{x}) = \prod_{t=1}^{\kappa} p(y_t|\mathbf{x}, y_1, ..., y_{t-1}) \tag{2}$$

The encoder portion of the neural network encodes $\mathbf{x}$ into a vector-valued, so-called *context*. Conditioned on the context and all previous decoder hidden states, the decoder generates the output tokens $y = (y_1, ..., y_\kappa)$. Both encoder and decoder are distinct recurrent neural networks (LSTM's in our case). The decoder generates a sequence of hidden states and outputs a hidden state $\mathbf{h}_t^L$ at the topmost, L-th layer at timestep t. The individual factors in Eq. 2 are finally obtained using a feedforward neural network with a softmax layer that maps each hidden state to a probability distribution over the token vocabulary V_D of the decoder. We optimize standard cross-entropy loss.

3.2 Grammar-Constrained Decoder

LR parsers are used to verify that a given token stream is derivable from a deterministic context-free grammar G. The parsing stage is usually preceded by lexical analysis. During lexical analysis, a source code string is converted into a sequence of tokens, ready to be consumed by the parser. LR parsers employ ACTION and GOTO tables associated with the grammar, governing the applicable shift-reduce decisions the parser can make on each token input and determining the error states of the parser. The decoder stage in the SEQ2SEQ model can be viewed as taking the role of the lexical analyzer in the parsing process (see Figure 1).

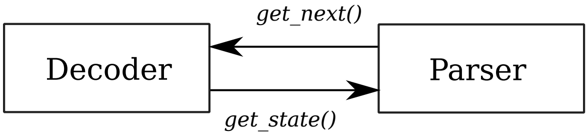

Figure 1: The parser requests the next token from the decoder. The decoder queries the current parser state to determine the set of applicable tokens, generates a token and returns it to the parser.

Algorithm 1 : LR(1) Assisted Decoder

```
let $s$ ← top of stack state;
let $h$ ← encoder context vector;
let $a$ ← Token(SOS, '<SOS>');

while ACTION($s$, $a$) $\neq$ acc do
    let $e_t$ ← EXPECTED($s$);
    if $|e_t| > 1$: $h$, $a$ ← DECODE($h$, $e_t$, $a$);
    else: $a$ ← $e_t$;

    if ACTION($s$, $a$) = shift $i$:
        push state $i$ onto stack;
    elif ACTION($s$, $a$) = reduce [$i$, $A \to \beta$]:
        pop $|\beta|$ symbols off stack;
        push GOTO($i$, $A$) onto stack;

    $s$ ← top of stack state;
```

The decoder determines the set of applicable tokens in its vocabulary by consulting the parser's ACTION table. We generate a probability distribution as described in section 3.1 over the actions (identified by lexical tokens) in the current parser state and return the most likely token to the parser. The parser consumes the token, updates its state, and requests the next token. This process continues until the parser encounters a token that indicates acceptance (an EOF token "\$"). The neural network model is implemented using PYTORCH [1]. The parser implementation relies on the parsing toolkit LARK [2]. The output vocabulary V_D consists of all source code tokens defined by the given grammar. Literals, such as string or integer values, are usually tokenized by matching them with a regular expression. We explicitly include all occurrences of literal values in the data sets as distinct tokens in the vocabulary. Algorithm 1 describes our modified LR parsing procedure relying on the decoder module providing the token stream. The procedure is initialized with the context vector obtained from the encoder and the parser start state. Given the state s, we determine the set of possible tokens e_t by looking in the parser's ACTION table. Conditioned on the previous hidden state, we invoke the DECODE function and generate an output distribution $\hat{\mathbf{y}}$ over *all* output vocabulary tokens. We finally choose $a = \max(\hat{\mathbf{y}}_{e_t})$ as our prediction, where $\hat{\mathbf{y}}_{e_t}$ are the elements of $\hat{\mathbf{y}}$ indexed by e_t. The next hidden state h_t is returned, and the parser updates its state by parsing a. On shift actions, we push the associated state i onto the stack and request the next token. On reduce actions, we pop the recognized handle off the stack and push the left-hand side of the production onto the stack. The decoding procedure concludes when the parser encounters a token that indicates acceptance (corresponding to action "acc"). Note, that the decoder is invoked only if $|e_t| > 1$, i.e., when there is more than one applicable token. Otherwise, we simply set $a = e_t$. The additional computational overhead of running a single parsing step is constant at each decoding step. Although most programming languages are close to deterministic, generalizing our approach to GLR parsers (and thus to context-free grammars) may incur an additional computational cost proportional to the non-determinism in the grammar (Tomita, 1985).

3.3 Model Training

Algorithm 1 is only used during inference. Thus, during model training, the decoder may generate sequences $s \notin L(G)$. Furthermore, since the decoder is only invoked when $|e_t| > 1$, there is a mismatch between sequences seen during training and during test time. To account for this mismatch, each target sample is parsed prior to training, and

---

[1] https://github.com/pytorch

[2] https://github.com/lark-parser/lark

for each state s for which $\text{EXPECTED}(s) = 1$, we filter the corresponding target sequence element from the target sample.

4 Experimental Evaluation

We present preliminary evaluation results and compare our approach to a recurrent SEQ2SEQ baseline (see section 3.1) and an attentional SEQ2SEQ model as reported in (Finegan-Dollak et al., 2018). Our attentional model extends the recurrent baseline with an attention layer as proposed by (Bahdanau et al., 2014).

4.1 Datasets

For our trials we use the canonicalized and annotated semantic parsing data sets for text-to-SQL tasks provided by (Finegan-Dollak et al., 2018). Compared to data sets like WIKISQL, GEOQUERY and ATIS feature complex queries with low levels of redundancy. We hypothesize that the benefits of a grammar-constrained decoder will be particularly pronounced in data sets with high complexity and variability. To ensure comparability, we use identical training, validation and test splits as (Finegan-Dollak et al., 2018).

4.2 Setup

We run trials without entity anonymization and with anonymized entities. We refer to trials with the standard dataset, i.e., the trials without anonymized entities, as *standard* trials. Trials with entity anonymization are referred to as *oracle* trials. Greedy-search was used for generating output sequences. We measured the exact match classification accuracy. A predicted token sequence that is identical to the token sequence in the corresponding test set example constitutes an exact match. Stochastic gradient descent with momentum (0.9) and a learning rate of 0.1 was used for each trial. The batch sizes ($\{16, 32, 64\}$ for GEOQUERY and $\{128, 256\}$ for ATIS), hidden and embedding dimensions ($\{64, 96, 128, 256\}$), the dropout rate for embeddings and hidden units ($\{0.05, 0.1, 0.2, 0.4\}$), the number of layers ($\{1, 2\}$) and the teacher-forcing ratio ($\{1.0, 0.9, 0.8, 0.7\}$) were determined using grid search. We tested the models with best validation set performance during training and set an early stopping criterion.

4.3 Results

In Table 1 and Table 2 we present the results of the evaluation. We see the greatest improvements in the oracle trials without an attention layer. This verifies that the main utility of enforcing syntactical constraints lies with resolving the complex syntactical structures of target logical forms. Correctly recognizing entities and inserting the appropriate literals into the query is more akin to a *slot-filling* task than a semantic parsing task, and we observe no added value in enforcing grammatical constraints to resolve such literals in the standard trials. Applying an attention mechanism to both our approach and the basic recurrent model of (Finegan-Dollak et al., 2018) further puts the results into perspective. An attention layer in recurrent SEQ2SEQ models helps with resolving long range dependencies that may occur when expanding non-terminals, for example, involving long sub-queries (Bahdanau et al., 2014). Similarly, long range dependencies are resolved by virtue of the LR parser, ensuring that any non-terminal node is fully expanded, even if it involves sub-expressions that are expanded to long token sequences. Thus, using an attention mechanism, syntactic relationships between tokens can be learned much better, although syntax errors cannot be completely precluded as with an LR parser.

	GEOQUERY	
	Standard	**Oracle**
Ours	34%	63%
+ Attention	51%	69%
Finegan-Dollak et al.	27%	49%
+ Attention	63%	73%

Table 1: Exact match accuracy on GEOQUERY.

	ATIS	
	Standard	**Oracle**
Ours	9%	48%
+ Attention	33%	55%
Finegan-Dollak et al.	8%	14%
+ Attention	46%	57%

Table 2: Exact match accuracy on ATIS.

5 Conclusion and Future Work

We showed that grammatical constraints can be enforced with LR parsers, imposing no assumptions on the neural machine translation model used and adding little computational overhead. We intend to expand the trials to include other logical forms than SQL and comparable approaches to enforcing grammatical constraints (Xiao et al., 2016; Yin and Neubig, 2017). Moreover, we intend to generalize our approach to context-free grammars using GLR parsers and enforce grammatical constrains at training time.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609, Berlin, Germany. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Masaru Tomita. 1985. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Kluwer Academic Publishers, USA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Daniel C. Wang, Andrew W. Appel, Jeff L. Korn, and Christopher S. Serra. 1997. The zephyr abstract syntax description language. In *Proceedings of the Conference on Domain-Specific Languages on Conference on Domain-Specific Languages (DSL), 1997*, DSL'97, page 17, USA. USENIX Association.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

Enhanced Metaphor Detection via Incorporation of External Knowledge Based on Linguistic Theories

Chang Su, Kechun Wu, Yijiang Chen*
School of Informatics, Xiamen University, Xiamen, China
`{suchang,cyj}@xmu.edu.cn`, `wukechun@stu.xmu.edu.cn`

Abstract

Use of external knowledge is an important and effective method applied widely in metaphor detection. Although existing knowledge-based methods perform well, when leveraging external knowledge, they take little consideration on linguistic theories of metaphor detection. Based on Metaphor Identification Procedure (MIP) and Select Preference Violation (SPV), directly using examples and definitions of words from the Oxford Dictionary[1], we propose two BERT-based models for metaphor detection: ExampleBERT and DefinitionBERT. Experimental results show that our methods achieve state-of-the-art performance on two established metaphor datasets. Furthermore, we show that our DefinitionBERT is highly interpretable.

1 Introduction

Metaphor Detection (MD) is a high-level natural language processing (NLP) task, which aims to identify the metaphorical expressions/words in the text. Identifying metaphors, a cognitive activity in which humans use their experience in one field to explain or understand another field (Shutova et al., 2016), is a challenging task that requires rich prior knowledge and a high level of semantic understanding.

In earlier studies, many resources were exploited to develop rule-based and machine learning systems, such as domain types,word abstractness/concreteness(Turney et al., 2011; Tsvetkov et al., 2014). Recently, many deep learning based methods have been applied to metaphor detection (Kehat and Pustejovsky, 2020; Le et al., 2020; Rohanian et al., 2020), which achieve the current state-of-the-art performance. They also make use of external knowledge. Hence, we can infer that incorporating external knowledge is indeed important. In this paper, we show that some level of lexical semantic information, even if its just dictionary entries, can improve performance in identifying verbal metaphor.

A recent study (Mao et al., 2019) shows the effectiveness of taking advantage of linguistic theories when identifying metaphors. According to one of the linguistic theories, Metaphor Identification Procedure(**MIP**) (Semino et al., 2007; Steen et al., 2010), a metaphor is identified if the literal meaning of a word contrasts with the means that word takes in this context. For example, in the metaphorical sentence, *the deep learning model is flying during training*, the context meaning of 'flying' is 'the loss of the model is getting bigger and even become indefinite', which contrasts with its literal meaning of 'move through the air using wings' according to **Oxford** Dictionary. An alternative approach is Select preference Violation(**SPV**) (Wilks, 1975, 1978), wherein a metaphor is identified by noticing a semantic contrast between a target word and its context. For example, in *the deep learning model is flying during training*, 'fly' is unusual in the context of 'model' and 'training': a model cannot fly.

To incorporate external knowledge, we take advantage of the linguistic theories of metaphor detection. Following SPV, we use examples of the word from Oxford Dictionary, where the literal meanings of the word are expressed in the contextual examples for the most of time. Hence, some common contextual information of the word can be inferred from examples. In accordance with MIP, we use the definitions of a word from the Oxford dictionary, which directly express the literal meanings of the word. To better use this knowledge and conform the idea of linguistic theories, we propose (1) ExampleBERT, which, before it identifies metaphor, learns the common contextual information of the

---

[1] `https://www.lexico.com/`

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1280–1287
August 1–6, 2021. ©2021 Association for Computational Linguistics

target word and (2) DefinitionBERT, which, while identifying a metaphor, directly takes advantage of the literal meanings of the target word. In particular, our contribution is two-fold as follows:

1. We directly use the examples and definitions of the word from the Oxford Dictionary. To the best of our knowledge, it is the first time this knowledge is incorporated into metaphor detection.

2. We propose ExampleBERT and DefinitionBERT. Experimental results show that both of our models can outperform the state-of-the-art models on two verb metaphor detection datasets. Also, experimental analysis proves that our DefinitionBERT is indeed effective and has a strong interpretability.

2 Related Work

Metaphor identification is a linguistic metaphor processing task that identifies metaphors in textual data. Most of the earlier works on metaphor identification were based on feature-engineering. Unigrams, imageability, concreteness, abstractness, word embedding and semantic classes are features commonly employed by supervised machine learning (Turney et al., 2011; Assaf et al., 2013; Tsvetkov et al., 2014; Klebanov et al., 2016). Recently, many deep learning based methods have been proposed, which treat metaphor identification as a sequence tagging task. Considering whether to use external knowledge directly, we divide these methods into the following two categories:

Use of pre-trained word embeddings. The first methods use only pre-trained word embeddings, which are commonly used in NLP tasks. (Wu et al., 2018) proposed a model based on word2vec (Mikolov et al., 2013) and PoS tags and word clusters, which are encoded by a Convolutional Neural Network (CNN) and Bi-LSTM. The encoded information is directly fed into a softmax classifier. (Gao et al., 2018) and (Mao et al., 2019) concatenated Glove (Le et al., 2020) and ELMO (Peters et al., 2018) as the inputs of Bi-LSTM, the difference is (Mao et al., 2019), inspired by linguistic theories, uses attention mechanism to improve performance.

External knowledge. The second methods use different kinds of external knowledge to boost performance. (Kehat and Pustejovsky, 2020) use Vision-Language datasets to derive the concreteness scores of words and then convert them to Visibility Embeddings, which, like with (Gao et al., 2018) , finally feed to Bi-LSTM. (Le et al., 2020) propose a multi-mask learning method, which transfer knowledge from Word Sense Disambiguation (WSD); to improve performance, they also employe Graph Convolution Neural networks (GCN) with dependency trees. Like (Le et al., 2020), (Rohanian et al., 2020) also use GCN, but they incorporate annotations for verbal multiword expressions. Obviously, our methods belong to this second category.

3 Method

3.1 BERT

BERT (Devlin et al., 2019) is a powerful language representation model, whose architecture is a multi-layer bidirectional transformer encoder. The BERT model is pre-trained on a large corpus and two novel unsupervised prediction tasks, i.e., masked language model and next sentence prediction tasks are used in pre-training. Here, it must note that BERT is chosen as our base model, not only because of its excellent performance on many other NLP tasks, but also BERT is a bidirectional language model. More specifically, during training, BERT randomly mask some words in the sentence and then use all the unmasked words to predict them based on a self-attention mechanism. Hence, this procedure allow BERT to learn the common context of the target word, which is very useful for our task because if a target word appears in uncommon contexts, then BERT is more likely to predict it to be a metaphorical word.

3.2 BERT(Token-CLS)

To incorporate BERT to our metaphor detection task, we take the final hidden state of the token corresponding to the target word, and then add a classification layer to predict whether or not the target word is metaphorical. We compare this model as our baseline with ExampleBERT and DefinitionBERT mentioned below.

3.3 ExampleBERT

The intuition behind SPV is that metaphoricity is identified by detecting the incongruity between a target word and its context. Hence, we assume that, if a model has learned the common context information of a target word, then the model works more effectively. As described in Section 3.1 above, a bidirectional pre-training model satisfies our requirement. Therefore, our proposed ExampleBERT

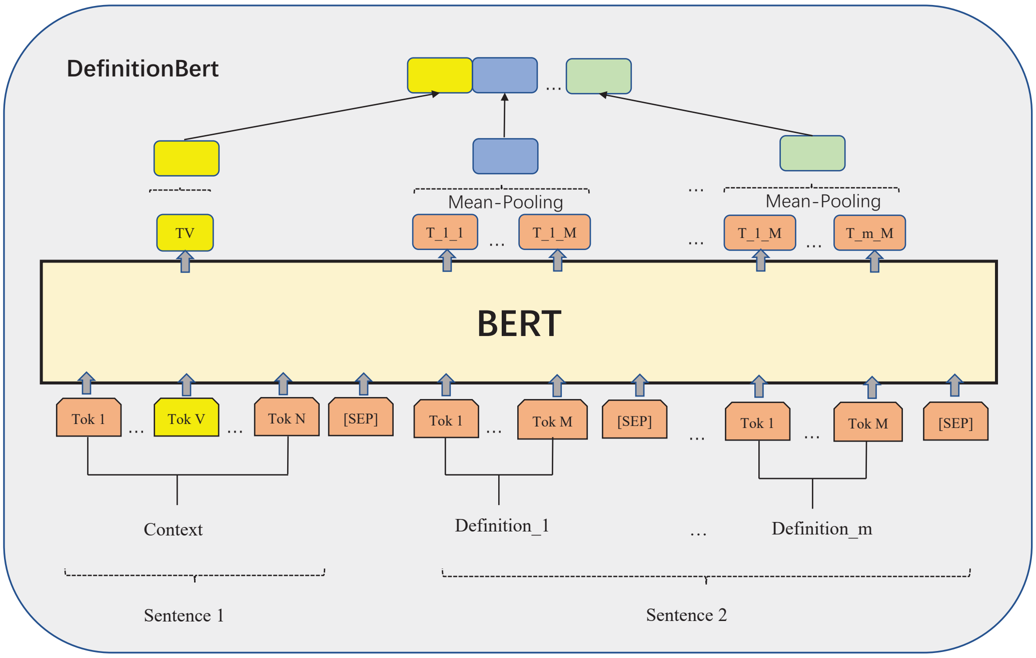

Figure 1: The overall of DefinitionBERT architecture and its context-definitions input pair.

model is built based on the standard BERT architecture (Devlin et al., 2019) which is based on the two-stage '*Pre-training*'-then-'*Fine-tuning*' pre-training language model approach, that recently become enormously popular in NLP. During the pre-training phase, we collect examples of the target word under its definitions from the Oxford dictionary and use only MaskLM as our pre-training objective. Here, we continue pre-training based on the pre-trained uncased BERT$_{BASE}$ model from (Wolf et al., 2020). The train strategy is the same as (Devlin et al., 2019). The training data generator chooses fifteen percent of the token positions at random for prediction. If the i-th token is chosen, we replace the i-th token with (1) the [MASK] token eighty percent of the time, (2) a random token ten percent of the time, and (3) the unchanged i-th token ten percent of the time. Here, our hypothesis is that most of the examples of a target word are expressing its literal meanings. Thus, whether or not a target word is selected, the model can also learn some common context information of a target word. During fine-tuning phase, we directly use the pre-trained ExampleBERT to fine-tune on the metaphor detection datasets as described in Section 3.2 above.

3.4 DefinitionBERT

Based on MIP, we assume that if we tell the model directly the literal meanings of the target word, then the model will work more effectively. Fortunately, BERT can explicitly model the relationship of a pair of texts, and this has been proved to be beneficial to many pair-wise natural language understanding tasks. Therefore, to fully leverage the definitions of words, we construct *context-definition* pair based on all possible definition of the target word from the Oxford dictionary, thereby treating MD task as a sentence pair classification problem seemingly. But, different from (Huang et al., 2019), here we cannot and don't need to match multiple definition and sentence directly one by one, because the contextual meaning of a metaphorical word is different from all its definitions. Also, we don't know which definition the contextual meaning of a non-metaphorical word corresponds to. Moreover, although there are word definition collections in WordNet (Miller, 1995), we find they cannot express accurately the literal meaning of words, and some of them are exactly the metaphorical meanings. For example, in WordNet, one of the definition for 'drink' is 'take in liquids'. On one hand, in the sentence, *car drinks gasoline*, that definition does not help us, or a model, identify that 'drink' is metaphorical. On the other hand, the Oxford Dictionary definition – 'take (a liquid) into the mouth and swallow' – can be of help. A car, which has no mouth and cannot swallow, is obviously unsuitable here. Hence, the latter is helpful to us.

As shown in Figure 1, we directly concatenate the multiple definitions of the target word, and use "[SEP]" to separate them. Finally we use the *context-definitions* pair as the inputs for BERT. After encoding by BERT, we take the final hidden state of the target word as its context meaning. To obtain the literal meaning of its definition, we also take the final hidden states of the tokens of each definition, and use Mean-Pooling to average the hidden states of each definition, which represents literal meaning expressed by the definition. This is formulated as follows :

$$h_{di} = f_m(f_b(\boldsymbol{x_{di}})) \quad (1)$$

where f_b represents the BERT encoder, and $f_m : \mathbb{R}^{n\times d} \rightarrow \mathbb{R}^d$ is a mean pooling function that maps from output vectors of n tokens to the definition vector. Then, we concatenate the vectors of the target word and definitions into one vector and apply a Feed-Forward Neural Network (FFNN) over the concatenated representations. This is formulated as:

$$h_f = FFNN([h_{tt}; h_{di}; ...h_{dn}]) \quad (2)$$

where h_{tt} indicates the hidden state of the target word from BERT. Then h_f is taken as input for a logistic regression classifier to make the prediction.

4 Experiments

4.1 Dataset

To be compatible with previous work (Gao et al., 2018; Mao et al., 2019; Le et al., 2020; Rohanian et al., 2020), we evaluate the proposed models using three widely used datasets for metaphor detection.

VUA (Steen and Gerard) It represents the largest public evaluation dataset for metaphor detection that is used by the NAACL-2018 Metaphor Shared Task(Klebanov et al., 2016). It contains 10,567 sentences and the average length of sentences is 19.4. Annotation for this dataset is based on MIP, for which every word in a sentence is labeled for metaphor identification. There are two versions of this dataset: (1) VUA ALL POS, where words of all types (e.g., nouns, verbs, adjectives) are labeled, and (2) VUA VERB, which focuses only on the verbs for metaphor detection. In this paper, we consider only the VUA VERB version.

MOH-X (Mohammad et al., 2016) Here, the sentences are shorter and simpler than those in the other datasets, as they are sampled from WordNet. It contains 647 sentences and the average length of sentences is 8.0. Only one single verb is labeled in each sentence in this dataset.

TroFi (Birke and Sarkar, 2006) This dataset consists of sentences from the 1987-89 Wall Street Journal Corpus Release 1 (Charniak et al., 2000). It contains 3737 sentences and the average length of sentences is 28.3. Each sentence has a single annotated target verb. There are only fifty unique target verbs in this dataset, which means, that for one target verb, there are many training samples.

4.2 Baselines

RNN-ELMo (Gao et al., 2018) This very representative model uses Glove and ELMo as features for sequential metaphor identification. The ELMO word vectors they trained has been adopted in many subsequent works.

RNN-HG & RNN-MHCA (Mao et al., 2019) These are BiLSTM-based systems grounded in linguistic theories of SPV and MIP, which are the first to explore using linguistic theories to directly inform the design of Deep Neural Networks (DNN) for metaphor identification. They use the Glove and ELMO word embeddings as the literal meaning of a word.

MUL-GCN (Le et al., 2020) This is a multi-task learning model for metaphor detection that, to improve performance, features graph convolutional neural networks to appropriately capture the following; important context words, the control mechanism to emphasize the target words, and the transference of knowledge from WSD.

BERT+MWE-Aware GCN (Rohanian et al., 2020) This is a neural model to classify metaphorical verbs in their sentential context using information from the dependency parse tree and annotations for verbal multiword expressions. It evaluates on the MOH-X and TroFi datasets.

4.3 Setup

For pre-training ExampleBERT, we collect about 40,000 examples of the verb words in all three datasets (See Section 4.1). The batch size is 128; the learning rate is 5e-5, and we train over ten epochs. For DefinitionBERT, because different words have a different number of definitions and to achieve batch computing, we choose the most common three definitions[2] for each word. If a word don't have three definitions, we simply use *"no*

---

[2] Although the dictionary doesn't state that definitions listed in the page are sorted by frequency, it is basically indeed this case according to our observation.

Models	VUA VERB				MOH-X				TroFi			
	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
RNN-ELMo (Gao et al., 2018)	69.1	53.4	65.6	58.9	78.5	75.3	84.3	79.1	73.7	68.7	74.6	72.0
RNN-HG (Mao et al., 2019)	82.1	69.3	72.3	70.8	79.7	79.7	79.8	79.8	74.9	67.4	**77.8**	72.2
RNN-MHCA (Mao et al., 2019)	81.8	66.3	75.2	70.5	79.8	77.5	83.1	80.0	75.2	68.6	76.8	72.4
MUL-GCN (Le et al., 2020)	83.2	72.5	70.9	71.7	79.9	79.7	80.5	79.6	**76.4**	73.1	73.6	**73.2**
BERTBaseline (Rohanian et al., 2020)	-	-	-	-	78.04	78.38	77.87	77.82	70.38	70.54	68.89	68.84
BERT+MWE-Aware GCN (Rohanian et al., 2020)	-	-	-	-	80.47	79.98	80.40	80.19	73.45	**73.78**	71.81	72.78
BERTbaseline(OURS)	85.48	**77.35**	72.19	75.07	80.05	80.65	76.70	78.43	75.05	73.02	67.86	70.30
ExampleBERT(OURS)	85.29	75.56	75.30	75.43	82.22	**83.21**	77.81	80.25	75.38	73.21	68.67	70.84
DefinitionBERT(OURS)	**85.65**	76.02	**76.15**	**76.09**	**84.24**	82.90	**84.09**	**83.38**	75.70	73.32	69.64	71.40

Table 1: Performance on three metaphor detection datasets

definition ." to replace the remaining insufficient definitions. We believe that three definitions cover most of the context of words; adding more uncommon definitions could become noise for the model. Following the settings in the prior work, we perform 10-fold cross validation on MOH-X and TroFi and use the same splits of training, validation and test sets for VUA VERB datasets. For VUA VERB datasets, we select the best checkpoint on validation data as the final model to evaluate test data performance. For the MOH-X and TroFi datasets, we train 10 epoch and select the last epoch model to evaluate the test data for every fold. Finally we take the 10 fold average for the performance of our final model.

To pre-train ExampleBERT and fine-tune DefinitionBERT, we all use the pre-trained uncased BERT$_{base}$ model from (Wolf et al., 2020). The number of its transformer blocks is 12, the number of self-attention heads is 12, and the number of the hidden layer is 768. For the FFNN in Eq. 2 of DefinitionBERT, we simply use a 256 hidden units of fully connected layer, followed by a classification layer. The two models are all fine-tuned with shuffled minibatches of size 32. The Adam optimizer is used to update the parameters, and the initial learning rate is set at 5e-5.

4.4 Results

Results in terms of accuracy (Acc), precision (P), recall (R) and F1-score are given in Table 1. Scores with the best performances across all models are indicated in bold. Results not reported are indicated by (-). As shown in Table 1, our ExampleBERT and DefinitionBERT achieve state-of-the-art performance on VUA VERB and MOH-X datasets.

VUA VERB dataset. For the VUA VERB datasets, even our proposed BERT-Baseline model achieves excellent performance, gaining improvement over the best of the other methods (MUL-GCN) by a large margin: 2.28% and 3.37% on accuracy (Acc) and F1, respectively. Compared with our BERTBaseline, regarding F1, our ExampleBERT and DefinitionBERT show improvement of 0.36% and 1.02%, respectively.

MOH-X dataset. For the MOH-X dataset, our DefinitionBERT, compared with BERTBaseline and the best of the other models, achieves significant improvement across all results.

TroFi dataset. However,for the TroFi dataset, the performance of our ExampleBERT and DefinitionBERT is somewhat bad than other state-of-the-art results. Compared with our BERTBaseline, for F1, ExampleBERT and DefinitionBERT still show a gain of 0.54% and 1.10%, respectively, indicating that our method is effective also. The TroFi dataset, contains fewer samples than the VUA dataset, but with longer average sequence length (28.3). Thus, on one hand, it is more difficult for DefinitionBERT to capture the relationship between the target and its definitions. On the other hand, because the dataset contains only fifty unique verbs, there are many samples for a target verb, and most express the literal meanings of the word, e.g., the dataset contains 71 literal sentences and 25 metaphor sentences of the target word 'absorb'. Thus, the models can learn sufficient common contextual information and literal meanings of a target word from the dataset. That is to say, the prior knowledge we add provides only limited help. However, taking a step back, compared with the performances of other well-designed models, the performance of our model does not lag too far behind; therefore, we believe our method is still acceptable. Moreover, the results of MOH-X and TroFi dataset suggest that our two models are more useful when there exists only a small amount of training corpus.

We note that the DefinitionBERT always perform expect on precision, the possible reason is , when the model cannot obtain the context meaning of the target world accurately(i.e.,when the sentence is complex) , or the definitions we get from the dictionary are highly summarized, there would exist a gap between the context meaning and literal meaning, although they express the same meaning actually. So the model could predict the literal one to metaphorical more likely, then the precision could be lower(precision=TP/(TP+FP), FP increased). Finally, this could be a inspiration that how to improve our methods in the future work.

4.5 Analysis

As described in Section 3.4, if DefinitionBERT works correctly, it should learn the differences and relationships between the contextual and literal meanings expressed by the definitions of the target word during identifying. Therefore, to understand how DefinitionBERT uses the definitions we provide, we compute the cosine similarity between h_{tt} and each h_{di} described in Eq. 2. If a word is predicted as metaphorical, then the cosine similarity between its definitions will be very small, and the definition which expresses the literal sense of its contextual meaning would always have the smallest cosine similarity. Inversely, if a word is predicted as non-metaphorical, the value will be larger, and the meaning with the greatest cosine similarity will always be the definition that expresses its contextual meaning.

A specific example is given in Table 2. For example, in the sentence, *Her husband often abuses alcohol*, 'abuses' is a metaphorical word; its context meaning is 'a man drinks too much resulting in a bad effect'. Thus, we infer this metaphorical meaning is based on the first definition that has the smallest similarity. In the sentence, *This boss abuses his workers*, 'abuse' is a non-metaphorical word; its context meaning is 'speaking in a insulting and offensive way', which obviously is the third definition that has the greatest similarity. That is to say, our DefinitionBERT takes advantage of the definitions during training. The definitions directly help the model distinguish the contextual and literal meanings of the target word, which exactly is our purpose.

4.6 Disscussion

The main reason for the improvements in our experimental results is that we use external knowledge based on linguistic theories, which is very suitable and effective for detecting metaphors. The ways we incorporate the examples and definitions of a word correspond exactly to the two pre-training objectives of BERT, which are also its advantages. However, it seems possible to combine ExampleBERT and DefinitionBERT to attain better performance, we can use pre-trained ExampleBERT to fine-tune DefinitionBERT. But, our experimental results show that, although its performance can surpass that of ExampleBERT, but cannot surpass DefinitionBERT. The possible reason is may due to the only MaskLM pre-training objective, the ability of pre-trained ExampleBERT to model the relationship of a pair-wise is weakened.

RNN-HG and RNN-MHCA proposed by (Mao et al., 2019), which are inspired also by linguistic theories, focus more on the model architecture suitable for SPV or MIP; whereas, we focus on external knowledge suited for SPV or MIP. Moreover, we believe our ExampleBERT and DefinitionBERT are just base models that can be further improved by other technology, such as GCN applied in (Rohanian et al., 2020).

Moreover, compared to previous state-of-the-art models, especially knowledge-based methods like (Le et al., 2020; Rohanian et al., 2020), our DefinitionBERT is highly interpretable while achieving excellent performance. As described in Section 4.5, because our DefinitionBERT locates the intended meaning of the metaphor in context, it helps us further interpret metaphors. One approache for metaphor interpretation is *Definition Generation* proposed in (Zayed et al., 2020), which aims to find the most probable definition/interpretation (if exists) of the highlighted expression among the given definitions. Obviously, our DefinitionBERT is very suitable for this task(dataset). Another approach is *Lexical Substitution* explored in (Mao et al., 2018), where the metaphoric word/phrase is replaced with its literal counterpart to clarify its semantic meaning. We also believe our DefinitionBERT can be an alternative method for (Mao et al., 2018).

5 Conclusion

We proposed two simple, but effective, methods for metaphor detection, which achieve state-of-the-art performance on two verb metaphor detection datasets. More importantly, we showed that our DefinitionBERT is highly interpretable and can be further applied to metaphor interpretation. For future work, we will explore how to use the external knowledge of words for a sequential task, such as the VUA ALL POS dataset, which is not evaluated

Abuse (definitions according to the Oxford Dictionary)
d-1 : use (something) to bad effect or for a bad purpose;misuse.
d-2 : treat with cruelty or violence, especially regularly or repeatedly.
d-3 : speak to (someone) in an insulting and offensive way.

Abuse (samples and cosine similarity between the three definitions)					
samples	**label**	**predict**	**d-1-s**	**d-2-s**	**d-3-s**
1.Her husband often abuses alcohol .	1	1	0.1632	0.1948	0.1800
2.This boss abuses his workers .	0	0	0.6448	0.6457	0.6478
3.The actress abused the policeman who gave her a parking ticket .	0	0	0.7344	0.7312	0.7403
4. Do n't abuse the system .	1	1	0.0274	0.0267	0.0330

Table 2: Examples for the word 'abuse' from the MOH-X dataset. **'d-1-s'** indicates the cosine similarity between the feature vector of the first definition and the feature vector of the target word extracted from DefinitionBERT.

in this paper. A simple, crude way is to collect all the examples of words in the datasets and then continue to use ExampleBERT according to SPV. If based on MIP, combining the definitions of all words into one sentence like this paper do seems to be a terrible implementation. Moreover, there are several dictionaries (Zayed et al., 2020) giving examples and definitions of the word, and the examples or definitions from different dictionaries are somewhat different in types and contents, which may cause a different result when combined with our methods. Therefore, to obtain better performance, we will try resources from different dictionaries, where the premise is the definitions of the words must be non-metaphorical.

Acknowledgments

We thank Mr.Michael McAllister for his valuable assistance in proofreading this paper. We also appreciate the anonymous reviewers for providing valuable suggestions.

References

Dan Assaf, Yair Neuman, Yohai Cohen, Shlomo Argamon, Newton Howard, Mark Last, Ophir Frieder, and Moshe Koppel. 2013. Why "dark thoughts" aren't really dark: A novel algorithm for metaphor identification. In *CCMB*, pages 60–65. IEEE.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *EACL*.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Bllip 1987-89 wsj corpus release 1. *Linguistic Data Consortium, Philadelphia*, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, volume 1, pages 4171–4186.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. In *EMNLP*, pages 607–613.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. Glossbert: BERT for word sense disambiguation with gloss knowledge. In *EMNLP/IJCNLP*, volume 1, pages 3507–3512.

Gitit Kehat and James Pustejovsky. 2020. Improving neural metaphor detection with visual datasets. In *LREC*, pages 5928–5933. European Language Resources Association.

Beata Beigman Klebanov, Chee Wee Leong, E. Dario Gutiérrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *ACL*, volume 2.

Duong Le, My Thai, and Thien Nguyen. 2020. Multitask learning for metaphor detection with graph convolutional neural networks and word sense disambiguation. In *AAAI*, pages 8139–8146.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and wordnet based metaphor identification and interpretation. In *ACL*, volume 1, pages 1222–1231.

Rui Mao, Chenghua Lin, and Frank Guerin. 2019. End-to-end sequential metaphor identification inspired by linguistic theories. In *ACL*, volume 1, pages 3888–3898.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saif Mohammad, Ekaterina Shutova, and Peter D. Turney. 2016. Metaphor as a medium for emotion: An empirical study. In **SEM@ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Omid Rohanian, Marek Rei, Shiva Taslimipoor, and Le An Ha. 2020. Verbal multiword expressions for identification of metaphor. In *ACL*, pages 2890–2895. Association for Computational Linguistics.

Elena Semino et al. 2007. Mip: a method for identifying metaphorically used words in discourse. *Metaphor and symbol*, 22(1):1–39.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *HLT-NAACL*, pages 160–170.

Steen and Gerard. *A method for linguistic metaphor identification : from MIP to MIPVU*. John Benjamins Pub. Co.

G.J. Steen, A.G. Dorst, J.B. Herrmann, A.A. Kaal, T. Krennmayr, and T. Pasma. 2010. *A method for linguistic metaphor identification. From MIP to MIPVU.* Number 14 in Converging Evidence in Language and Communication Research. John Benjamins.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *ACL*, volume 1, pages 248–258.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *EMNLP*, pages 680–690.

Yorick Wilks. 1975. A preferential, pattern-seeking, semantics for natural language inference. *Artif. Intell.*, 6(1):53–74.

Yorick Wilks. 1978. Making preferences more active. *Artif. Intell.*, 11(3):197–223.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. Neural metaphor detecting with CNN-LSTM model. In *Fig-Lang@NAACL-HLT*, pages 110–114.

Omnia Zayed, John Philip McCrae, and Paul Buitelaar. 2020. Figure me out: A gold standard dataset for metaphor interpretation. In *LREC*, pages 5810–5819. European Language Resources Association.

Controlling Text Edition by Changing Answers of Specific Questions

Lei Sha, Patrick Hohenecker, Thomas Lukasiewicz
Department of Computer Science
University of Oxford, United Kingdom
`{lei.sha, thomas.lukasiewicz}@cs.ox.ac.uk`
`patrick@serein.ai`

Abstract

In this paper, we introduce the new task of *controllable text edition*, in which we take as input a long text, a question, and a target answer, and the output is a minimally modified text, so that it fits the target answer. This task is very important in many situations, such as changing some conditions, consequences, or properties in a legal document, or changing some key information of an event in a news text. This is very challenging, as it is hard to obtain a parallel corpus for training, and we need to first find all text positions that should be changed and then decide how to change them. We constructed the new dataset WIKIBIOCTE for this task based on the existing dataset WIKIBIO (originally created for table-to-text generation). We use WIKIBIOCTE for training, and manually labeled a test set for testing. We also propose novel evaluation metrics and a novel method for solving the new task. Experimental results on the test set show that our proposed method is a good fit for this novel NLP task.

1 Introduction

In many cases, we need to change some specific content in a document. For example, in the legal domain, the items and conditions in contract documents often need to be revised many times. We would like to use artificial intelligence to conduct this process for human editors. A major difficulty of this process is that the machine learning model should decide where to edit and how to edit.

Usually, the place of specific content ("where to edit") can be located by a question, and the content updating ("how to edit") can be determined by the answer of the question. Therefore, in this paper, we propose the new task of *controllable text edition (CTE)*. In this task, we would like to achieve the following goal: *adjust some content of a document D, to make the answer A of a document-related question Q changed to a new answer A'*. For example, in Fig. 1, when we change the red part of the original text to the blue part, the answer of the question turned to the new answer as a consequence.

Origin document (D):

Lang Ping (born 10 December 1960) is a former Chinese volleyball player and the current head coach of China women's national volleyball team. She was the former head coach of the United States women's national volleyball team, herself being the MVP of women volleyball in 1984 Olympics.

Question (Q): What is Lang Ping's current job?

*Original Answer (A): **the head coach of China women's national volleyball team***

New Answer (A'): the director of the National Sports Administration

Modified document (D'):

Lang Ping (born 10 December 1960) is a former Chinese volleyball player and the current director of the National Sports Administration. She was the former head coach of the United States women's national volleyball team, herself being the MVP of women volleyball in 1984 Olympics.

Figure 1: The original text D is in the upper box. The question Q to D has an answer A (in red; its rationale in D also in red). If we would like to change the answer to the new answer A' (in blue), then we have to change some content in D, yielding the modified text D' (with the new content in blue) in the lower box.

There are three main challenges in this task:

(1) The machine learning model should decide the positions that need to be changed in the document. Usually, finding the answer positions for a given document-related question is similar to extractive machine reading comprehension tasks (Zeng et al., 2020), which requires to fully understand both the question and the document. Nearly all extractive machine reading tasks, such as SQuAD (Rajpurkar et al., 2016, 2018) and CNN/Daily Mail (Hermann et al., 2015), focus on extracting one span from the document as answer. Differently from extractive machine reading, in our task, the answer A is not necessarily a substring of the document, and there may exist multiple positions that have to be changed. Therefore, our task is much more challenging than extractive machine reading.

(2) The model should generate a new document that supports the new answer A' for question Q. Note that this cannot be solved by directly replacing the original words in the edit positions with the new answer A', because the new answer may not fit

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1288–1299
August 1–6, 2021. ©2021 Association for Computational Linguistics

perfectly with the document, which would make the document disfluent.

(3) There are nearly no parallel data for model training, because obtaining a large annotation set for this task is very hard.[1] However, the model may be trained by lists of triples $\langle Q, D, A\rangle$ that can be obtained from datasets in machine reading and/or structured data extraction (as described below).

In this paper, we introduce and define the task of controllable text edition (CTE). We propose to transform the WIKIBIO dataset (Lebret et al., 2016) into a list of triples $\langle Q, D, A\rangle$ for training. WIKIBIO was originally designed for table-to-text generation, in which each case is composed of a Wikipedia passage D and an infobox (which is a list of $\langle$field, content$\rangle$[2] pairs). In detail, we take each "field" in the infobox as the question Q, and each "content" in the infobox as the answer A. Therefore, for each $\langle$field, content$\rangle$ pair, we can create a $\langle Q, D, A\rangle$ triple. After some pruning, we finally selected 26 different Q's and $141k$ $\langle Q, D, A\rangle$ triples for the training set, as well as $17.7k$ triples for the development set. We also annotated a small test set of about $1k$ data for evaluation in the form of $\langle Q, D, A, A', D'\rangle$ (A' represents the new answer, and D' represents the ground-truth modified text). The resulting new dataset is called WIKIBIOCTE.

In addition, we propose a novel method, called *Select-Mask-Generate (SMG)*, to solve the proposed CTE task. In this method, we use the selector-predictor architecture by Sha et al. (2021) to select the answer-related tokens, and we then use complementary masks to split the text into an answer-related part and an answer-unrelated part. Then, we reconstruct the original text based on the answer-unrelated part and the original answer. The reconstruction process is a partial generation method, which only generates the masked-out part without any length limit. In our experiments, the SMG model has achieved the state-of-the-art performance, compared to baseline models in the generation of modified documents. The code and the test set WIKIBIOCTE are available at: `https://sites.google.com/view/control-text-edition/home`.

2 Related Work

The proposed task of *controllable text edition* is related to the following existing tasks.

2.1 Attribute Disentanglement

Attribute disentanglement tends to control the attributes of a given text or image (such as sentiment, tense, syntax, or face pose) by disentangling different attributes into different subspaces. When transferring attributes, the content of the text/image needs to be preserved. Usually, disentanglement works can be divided into implicit and explicit disentanglement. Implicit disentanglement (Higgins et al., 2017; Chen et al., 2018; Moyer et al., 2018; Mathieu et al., 2018; Kim and Mnih, 2018) separates the latent space into several components in a purely unsupervised way, expecting that each component corresponds to an attribute. However, the number of components cannot be decided in advance, neither does the correspondence between attributes and components. Also, the training process may prune some of the components (Stühmer et al., 2019), which will hurt the interpretability of the latent space. Explicit disentanglement (Chen et al., 2016; John et al., 2019; Romanov et al., 2019; Sha and Lukasiewicz, 2021) tends to separate the latent space into more interpretable components with explicit correspondence to specific attributes. Hence, it usually requires gold labels of attributes in the training set.

In comparison, our task tends to control the content of the text by tuning answers to text-related questions. Attribute disentanglement is difficult to be applied to our task, because the modification of the content should be decided by both the question and the answer simultaneously, which is much sparser than attributes.

2.2 Lexically Constrained Decoding

Lexically constrained decoding (Hokamp and Liu, 2017; Miao et al., 2019; Sha, 2020) directly controls the output of the generation model by adding constraints. Usually, the constraints include hard constraints (requiring the generated sequence to contain some keywords) and soft constraints (requiring the generated sentence to have the same meaning to a given text). The basic methods of lexically constrained decoding can be divided into enhanced beam search (Hokamp and Liu, 2017; Post and Vilar, 2018) and stochastic search (Miao et al., 2019; Liu et al., 2020; Sha, 2020). Enhanced

---

[1] To annotate a large parallel dataset, we need to prepare a document, a document-related question, and its expected answer. Then, the data grader should provide an adjusted version of the document that satisfies the expected answer, which requires the data grader to have a high education level.

[2] In the Wikipedia Infobox, "field" represents the type of information (such as *Name*, *BirthDate*, and *Known for*), while "content" represents the value of "field".

Table:

ID	Field	Content
1	Name	*Frank Fenner*
2	Born	*21 December 1914, Ballarat*
3	Died	*22 November 2010 (aged 95) Canberra*
4	Occupation	*Virology*
5	Nationality	*Australian*
6	Known for	*Eradication of smallpox, Control of Australia's rabbit plague*

Text: Frank John Fenner (21 December 1914 – 22 November 2010) was an Australian scientist with a distinguished career in the field of virology. His two greatest achievements are cited as overseeing the eradication of smallpox, and the control of Australia's rabbit plague by introducing the Myxoma virus.

Table 1: An example of a Wikipedia infobox and a reference text.

beam search (Hokamp and Liu, 2017; Hasler et al., 2018; Hu et al., 2019) changes some strategies in beam search to make the process of searching for a constraint-satisfying sentence easier. However, for some tasks with an extremely large search space, beam-search-based methods may be computationally too costly or even fail (Miao et al., 2019). Stochastic search tends to edit an initial sentence step-by-step, where the editing position and action can be decided by Metropolis-Hastings sampling (Miao et al., 2019), a discrete scoring function (Liu et al., 2020), or gradient-based methods (Sha, 2020). However, lexically constrained decoding is hard to be applied to our task, because adjusting the text to fit a text-related question's new answer is much more complicated than simply satisfying a hard or soft constraint.

2.3 Text Editing and Infilling

In some tasks, to simplify the text generation problem, researchers tend to edit existing text or prototypes to obtain a refined text that satisfies some specific requirements. Examples are the generation of summaries by template-based rewriting (Cao et al., 2018; Hashimoto et al., 2018) and the generation of text or a response by editing a prototype sentence (Guu et al., 2018; Pandey et al., 2018; Wu et al., 2019). In (Yin et al., 2018), the distributed representations of edit actions are learned and applied to editing Wikipedia records (Faruqui et al., 2018) and Github code (Yin et al., 2018). Panthaplackel et al. (2020) further integrate a copy mechanism into text editing.

Text infilling (Fedus et al., 2018) means to use machine learning models to fill the blanks of a cloze test. Zhu et al. (2019) propose a more general text infilling task, which allows an arbitrary number of tokens (instead of a single token) in each blank.

In the above text editing tasks, the goal of editing is always consistent among all the datasets: for a better summarization, a better response, or a better informative sentence. Differently from them, our proposed task requires the editing to be guided by the document-related answer of the question. So, each above case has a different editing goal. Thus, our task requires deciding where to edit according to the given question in the first step, and then deciding how to edit, which makes our task more complicated than all the above text editing tasks.

3 Dataset

We now formally define the task of controllable text edition and propose a dataset for this task.

3.1 Task Definition

The task of *controllable text edition (CTE)* is defined as follows. The input is a triple $\langle D, Q, A' \rangle$, where D is a document, Q is a document-related question, and A' is an expected answer for Q to D. The output is D', which is a minimal modification of D such that the answer for Q to D' is now A'. Note that the original answer of Q to D is A, but A is not an input to the task, and usually $A \neq A'$.

3.2 WIKIBIO as Controllable Text Editing Dataset

We propose to modify the WIKIBIO dataset (Lebret et al., 2016) to make it fit for our task. WIKIBIO was originally designed for table-to-text generation (Lebret et al., 2016; Sha et al., 2018; Liu et al., 2018), which generates a celebrity's biography according to his/her basic information. Each example in the dataset is composed of a Wikipedia infobox and a text (the first paragraph in the Wiki page) describing the infobox as shown in Table 1.

In an inverse way, the WIKIBIO dataset can be taken as a question-answering dataset: each *field* can be taken as a question, and each *content* can be taken as an answer. For example, in Fig. 1, the *field* "Occupation" can be interpreted as question "What is the person's occupation?", and the corresponding *content* "Virology" is the answer.

Therefore, we take the *text* in WIKIBIO as the document (D) in our task, the *field* as the question (Q), and the *content* as the answer (A). Due to the huge cost of data annotation, the model needs to be trained without the changed answer (A') and the referenced document (D').

For the creation of the training and development sets, we count the frequency of *fields* and select the *fields* that occurred more than 5k times in WIKIBIO's training set as candidate questions (Q's). Then, we filter out some Q's that do not have corresponding answers in D[3]. We then get a list of 26 different Q's as shown in Table 2. After filtering the Q's according to Table 2, we get $141k$ $\langle Q, D, A\rangle$ triples for the training set and $17.7k$ triples for the development set.

Then, we manually labeled a small test set in which each example contains (D, Q, A) as well as the changed answer (A') and the referenced document (D'). The annotation process can be illustrated as follows:

1. We randomly sampled an equal number of examples for all the *fields* in Table 2. For each *field*, we sample $\lceil \frac{1000}{\#F} \rceil$ cases ($\#F$ is the number of selected *fields*), to make sure that the size of the test set is around 1k.
2. We assigned a changed answer (A') to each example by randomly picking a similar phrase to the original answer (A). The similar phrase may occur in different examples, but it shares the same Q with the original answer (A).
3. We asked human data graders to give a modified text (D') for each example according to the original text (D), question (Q), and the changed answer (A'). We asked two talented linguistics to annotate the 1k test set.

Note that there are also other datasets that are potentially able to be modified as controllable text editing dataset, such as SQuAD (Rajpurkar et al., 2016), RACE (Lai et al., 2017), and MCTest (Richardson et al., 2013). We did not choose them for the following reasons: (1) For extractive machine reading tasks like SQuAD (Rajpurkar et al., 2016), the answers are simple substrings of the document, so that in most cases, the text modification in our task can be solved by a simple string replacement, which violates the goal of our task. (2) Multiple-choice machine reading tasks like RACE (Lai et al., 2017) usually require full and deep reasoning of the whole document to get the answer, which would make the text modification in our task unable to be solved by partial modification. Differently from them, most *contents* (A) in WIKIBIO usually cannot be directly extracted as substrings from the document (D). Besides, the *contents* usually has some related information that should be modified at the same time. For example, if somebody is a pianist, then he/she may have received a piano award instead of a guitar award. Therefore, WIKIBIO satisfies the goal of our proposed task: making minimal changes to the original document to make it fit the changed answer (A').

birth date	679.4k	name	675.8k	birth place	659.3k
death date	420.7k	death place	377.7k	occupation	231.9k
position	199.4k	nationality	187.1k	spouse	184.0k
fullname	180.2k	alma mater	115.5k	children	114.9k
residence	112.1k	religion	99.3k	predecessor	91.0k
successor	90.1k	known for	63.4k	origin	46.6k
country	43.5k	education	43.1k	instrument	36.7k
college	35.9k	citizenship	29.1k	ethnicity	28.7k
discipline	11.2k	work institutions	5.3k		

Table 2: The selected *fields* from WIKIBIO and their occurrence in WIKIBIO's training set. These are taken as the questions (Q's) in our proposed task.

4 Select-Mask-Generate (SMG) Method for Controllable Text Edition

We introduce the training and testing method of our proposed method. In the training phase, the model is trained to learn to recognize answer-related (A-related) tokens and learn to fill new-answer-related (A'-related) tokens into the blanks after deleting answer-related tokens.

4.1 Training Phase

In the training phase, we only have Q, D, and A. So, we teach the model to (1) identify answer-related information, and (2) be able to reconstruct D from A and $(D - A_p)$ (the original text with all answer-related information masked out, where A_p means the predicted answer-related tokens).

The model architecture is shown in Fig. 2. Inspired by InfoCal (Sha et al., 2021), we use a *Selector-Predictor* architecture to identify the least-but-enough answer-related words in the original document (D). The main architecture of the *Selector* network is a BiLSTM model, which samples[4] a binary-valued mask (M) for each input token (called answer mask), denoting whether to select

---

[3] Since D is the first paragraph in the Wikipedia page, it usually does not contain everything mentioned in the infobox, such as *death cause* and *high school*.

[4] The sampling process is implemented by Gumbel Softmax (Jang et al., 2016), which is differentiable.

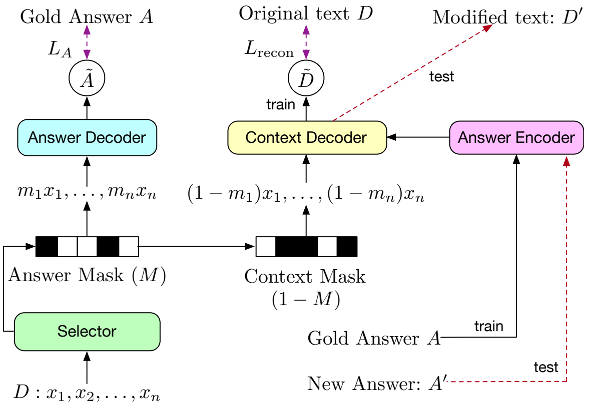

Figure 2: The architecture of our SME model. In the testing phase, we need to replace the input to the answer encoder from the gold answer A to the new answer A', then the output of the context decoder will become the modified text $\tilde{D}'$.

this token as answer-related token (1) or not (0). Given an input document $D = \{x_1, \ldots, x_n\}$ and a question Q, the *Selector* samples an answer-related mask $M = \{m_1, \ldots, m_n\}$ as follows:

$$M \sim \text{Sel}(M|D, Q), \tag{1}$$

where "Sel" represents the selector network. Then, we call the complement of the answer mask ($\overline{M} = 1 - M$) as the context mask, and we denote *context template* as the token sequence after masking out the answer-related tokens.

4.1.1 Answer Reconstruction

We require that the answer-related information contains everything about the answer A, so we use an answer decoder to reconstruct an answer sequence $\tilde{A}$. Then, we calculate the reconstruction loss L_A as follows:

$$p_a(\tilde{A}|M, D) = \text{Dec}_A\Big(\frac{1}{\sum_j m_j}\sum_i m_i x_i\Big), \tag{2}$$

$$L_A = \mathbb{E}_{M \sim \text{Sel}(M|D,Q)} p_a(A|M, D), \tag{3}$$

where Dec_A is the answer decoder, and p_a is the sentence distribution generated by Dec_A. Note that the input to Dec_A is the average vector of the selected token vectors: the answer-related tokens are usually very few, so it is not necessary to use heavier encoders like LSTMs (Hochreiter and Schmidhuber, 1997) or transformers (Vaswani et al., 2017).

4.1.2 Document Reconstruction

On the other hand, D should be reconstructed by the *context template* and the gold answer A. We use an LSTM encoder Enc_D to encode the *context tokens* as shown in Eqs. 4 and 5:

$$h'_1, \ldots, h'_n = \text{Enc}_D([\overline{m}_1 x_1, \ldots, \overline{m}_n x_n]), \tag{4}$$

$$H_m = \text{Maxpooling}(h'_1, \ldots, h'_n), \tag{5}$$

where $h'_1, \ldots, h'_n$ are the encoding vectors corresponding to each input token. We then take the averaged word vector of the input gold answer A, denoted V_A, as an external condition of the decoder.

Differently from conventional decoders, our decoder only partially generates tokens to fill in the blanks of the *context templates*, as shown in Fig. 3. This brings two changes in the training phase: (1) we only need to calculate the loss caused by the tokens filled in the blank, and (2) the model needs to learn an external end-of-answer (EOA) token S_{eoa} for each token filled in the blanks. The EOA token is very important because it is an indicator about when to stop filling the current blank.

Learning to generate the words. In each time step t of the decoder, we use an LSTM (Hochreiter and Schmidhuber, 1997) unit to predict the next word y_t and the EOA token S_{eoa} as follows:

$$h_t = \mathcal{F}_{\text{LSTM}}([y_{t-1}, V_A], h_{t-1}), \tag{6}$$

$$\begin{bmatrix} h_w \\ h_{\text{eoa}} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \end{bmatrix} \mathcal{F}_m(h_t), \tag{7}$$

$$s_t^{\text{lstm}}(w) = \mathcal{F}_w(h_w), \tag{8}$$

$$p(\tilde{S}_{eoa}(t)) = \text{Softmax}\big(\mathcal{F}_{eoa}(h_{\text{eoa}})\big), \tag{9}$$

where h_w and h_{eoa} are hidden layers (the time step index t is omitted), $\mathcal{F}_{\text{LSTM}}$ is an LSTM cell, $\mathcal{F}_m$, $\mathcal{F}_w$, and $\mathcal{F}_{eoa}$ are linear layers, and $s_t^{\text{lstm}}(w)$ is a scoring function that suggests the next word to generate. $p(\tilde{S}_{eoa}(t))$ is the probability distribution of the EOA token.

Note that in the decoder, we use the copy mechanism (Gu et al., 2016), which encourages the decoder to generate words by directly copying from the input context sequence D and answer sequence A. The copy mechanism computes a copy score $s_t^{\text{copy}}(w)$ for each word in D and A. Then, the generated probability of each word is computed as:

$$s_t(w) = s_t^{\text{lstm}}(w) + s_t^{\text{copy}}(w), \tag{10}$$

$$p_t(w) = \text{Softmax}(s_t(w)). \tag{11}$$

Thus, the document D's reconstruction loss is as follows:

$$L_{\text{recon}} = -\mathbb{E}_M\Big[\sum_t m_t \log p_t(y_t|\overline{M}, A)\Big], \tag{12}$$

where $M \sim \mathrm{Sel}(M|D,Q)$, the mask m_t is multiplied in each time step, because we only need the losses of blank-filling tokens.

Learning the end-of-answer (EOA) tags. We have an EOA tag for each blank-filling token. The EOA tag is 1 if the corresponding token is the last token in the blank. For the other blank-filling tokens, the EOA tag is 0. The gold EOA tag in each time step g_t^{eoa} can be computed by the difference between the previous answer mask m_{t-1} and the current answer mask m_t. There are three possible values (-1, 0, and 1): $g_t^{\text{eoa}} = 0$ when the difference is -1 or 0, and $g_t^{\text{eoa}} = 1$ when the difference is 1. Then, we have the cross-entropy loss as Eq. 13:

$$\begin{aligned} g_t^{\text{eoa}} &= \max(m_{t-1} - m_t, 0) \\ L_{\text{eoa}} &= -\mathbb{E}_M\Big[\sum_t \Big(g_t^{\text{eoa}} m_t \log p(S_{eoa}(t) = 1) \\ &\quad + (1 - g_t^{\text{eoa}}) m_t \log p(S_{eoa}(t) = 0)\Big)\Big]. \end{aligned} \tag{13}$$

Therefore, the final optimization objective is shown in Eq. 14:

$$L = L_A + \lambda_r L_{\text{recon}} + \lambda_{\text{eoa}} L_{\text{eoa}}, \tag{14}$$

where λ_r and λ_{eoa} are hyperparameters.

4.2 Inference Phase

In the inference phase, we take the new answer A' as the input to the context decoder instead of the gold answer A. Then, the output of the context decoder will become the modified text $\tilde{D}'$.

We choose an autoregressive partial generation method for inference. Our partial generation method can fill the blanks with any-length phrases and can directly replace any decoder, which cannot be done by any existing alternative methods. For example, in the method using global context (Donahue et al., 2020), it is an pretrained language model by itself. However, in our architecture, the masks are decided by the selector module. Therefore, even the number and length of the blanks cannot be decided before training. So, the ground-truth target sequence for the finetuning of the pretrained language model would also be hard to decide. Therefore, the partial generation method is the best choice for our task.

4.2.1 Partial Generation

Since we already have a *context template* when we are generating the modified document, we only need to generate tokens to fill the blanks in the *context template*. The partial decoding process is shown in Fig. 3. We use an indicator *state*= 0 to denote the *reading* mode (reading the *context template* words), and *state*= 1 to denote the *writing* mode (generating the blank-filling words). The basic generating process is described as follows: when the model is *reading* the *context template*, if it meets a masked token, the mode turns to *writing* mode, and it starts to generate words to fill the current blank. When the EOA tag turns to 1, or the decoding length l_g surpassed a limit $l_{\max}$, the mode turns back to *reading* mode. Note that this decoding process can generate an arbitrary number of words for each blank, and we can fill all blanks in a *context template* in a single decoding pass, which is much more efficient than MaskGAN (Fedus et al., 2018) and text filler (Zhu et al., 2019). The detailed algorithm is shown in Algorithm 1.

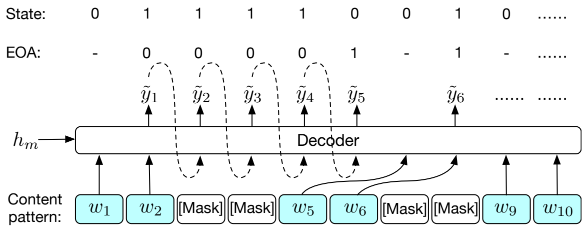

Figure 3: The partial decoding process. This process requires two tags (state and EOA tag) for indicating when to start generation and when to stop generation.

5 Experiments

In the experiment part, we proposed some specific evaluation metric for our *controllable text edition* task and then compare and analysis the performance of our proposed method (SMG) on the WIKIBIOCTE dataset.

5.1 Evaluation Metrics

For the evaluation of the modified document $\tilde{D}'$, we use the following two automatic evaluation metrics:

(1) BLEU ($\tilde{D}'$ vs. D'): This metric measures the BLEU score (Papineni et al., 2002) between the generated modified document $\tilde{D}'$ and the reference document D'.

(2) iBLEU (Sun and Zhou, 2012): This metric is previously widely used in evaluating paraphrase generation tasks (Liu et al., 2020; Sha, 2020). iBLEU is defined as: iBLEU $= \mathrm{BLEU}(\tilde{D}', D') - \alpha\mathrm{BLEU}(\tilde{D}', D)$[5], which penalizes the similarity

---

[5] α is set to 0.9, which is consistent with previous

Algorithm 1: The decoding process.

```
Input: Context template: $C$
Output: Generated Sequence: $\tilde{D}'$
Data: Read-write state: $S$, End-of-answer label:
      $S_{eoa}$, Context template index: $I_c$, Local
      generate length: $l_g$, current input token $x_{in}$
$S \leftarrow 0, I_c \leftarrow 0, l_g \leftarrow 0, \tilde{D}' \leftarrow []$;
Set the first input token $x_{in} \leftarrow C[0]$;
for each time step $t \leftarrow 1, 2, \dots$ do
    Calculate $\tilde{y}_t$ by Eqn. 11;
    Calculate $S_{eoa}$ by Eqn. 9;
    if $S = 0$ then
        $\tilde{D}' \leftarrow \tilde{D}' + [C[I_c]]$;
        if $C[I_c] \neq$ '[M]' and $C[I_c + 1] =$ '[M]'
         then
            $I_c \leftarrow I_c + 1$;
            while $C[I_c] =$ '[M]' do
                $I_c \leftarrow I_c + 1$;
            end
            if $S_{eoa} \neq 1$ then
                $S \leftarrow 1$;
            end
        end
        else if $C[I_c] \neq$ '[M]' and $C[I_c + 1] \neq$ '[M]'
         then
            $I_c \leftarrow I_c + 1$;
        end
    end
    else if $S = 1$ then
        $\tilde{D}' \leftarrow \tilde{D}' + [\tilde{y}_t], l_g \leftarrow l_g + 1$;
        if $S_{eoa} = 1$ or $l_g \geq l_{max}$ then
            $S \leftarrow 0, l_g \leftarrow 0$;
        end
    end
    if $S = 0$ then
        $x_{in} \leftarrow C[I_c]$;
    end
    else if $S = 1$ then
        $x_{in} \leftarrow \tilde{y}_t$;
    end
end
return $\tilde{D}'$;
```

	Seq2Seq	SMG (g)	SMG (p)
BLEU ($\tilde{D}$ vs. D)	82.21	**89.29**	87.53
iBLEU	5.63	**10.05**	8.94
diff-BLEU ratio	21.3%	**62.5%**	59.8%
Perplexity	**198**	235	373
Human (Correctness)	73.5%	**80.2%**	76.9%
Human (Fluency)	**4.56**	4.54	4.32

Table 3: The overall performance of all competing methods. SMG (g) denotes that the method SMG is using the gold templates for partial generation, and SMG (p) denotes that the method SMG is using the predicted templates for partial generation.

	Random	Seq2Seq	SMG
BLEU (predicted template)	21.5	59.5	**89.1**
Answer F_1	0.14	0.55	**0.68**

Table 4: Performance of answer-related words selection.

between the modified document $\tilde{D}'$ and the original document D. The goal of this metric is to measure the extent to which the model directly copies words from the original document D without taking any content from A'.

(3) diff-BLEU ratio: diff-BLEU is a BLEU score computed between $\tilde{D}'$ and a *difference sequence* between the gold modified document D' and the original document D. The *difference sequence* is obtained by masking out the longest common sequence between D and D' from D'. Since this maximum value of this BLEU score is the BLEU value between the gold modified document D' and the *difference sequence*, we use their quotient as the diff-BLEU ratio score as shown in Eq. 15:

$$\text{diff-BLEU ratio} = \frac{\text{BLEU}(\tilde{D}', D' - D)}{\text{BLEU}(D', D' - D)}. \quad (15)$$

(4) Perplexity: This metric measures the fluency of the generated content-modified document $\tilde{D}'$. We applied a third-party language model (Kneser-Ney language model (1995)) as the perplexity evaluator. We trained the language model on the whole training set of WIKIBIO, and use the trained model as the evaluation of fluency, where a lower perplexity value is better.

Besides, we used human effort to evaluate two aspects of the content-modified document $\tilde{D}'$. *Correctness* is an accuracy score from $0.0\% \sim 100.0\%$, which evaluates whether $\tilde{D}'$ has successfully turned the answer of question Q from A to A'. *Fluency* is from $0.0 \sim 5.0$, which evaluates whether $\tilde{D}'$ is fluent from a human being's view. The scoring details are in the supplemental materials.

Also, in our method, the selection of answer-related words is very important, so we have two evaluations for the selection part:

(1) BLEU (predicted template) is the BLEU score between the predicted template (the token sequence after we masked out the answer-related words from the text D) and the gold template (the common sequence of D and D').

(2) Answer F_1 measures the Bag-of-words (BOW) F_1 value of the generated answer $\tilde{A}$ compared to the gold answer A. This metric is difficult to

---

works (Liu et al., 2020).

Input	D: george evans -lrb- born 13 december 1994 -rrb- is an **english** footballer who plays as a midfielder **or centre-back for** manchester city .	D: andrei UNK -lrb- born 1975 in satu mare , romania -rrb- is a retired romanian **aerobic gymnast** . he had a successful career winning four world championships medals -lrb- two gold , one silver , and one bronze -rrb- after his retirement in 1997 he went with to germany where he works as a gymnastics coach at the UNK gymnastics club in hanover .	D: andrew justin stewart coats -lrb- born 1 february **1958 -rrb- is an australian – british** academic cardiologist who has particular interest in the management of heart failure . his research turned established teaching on its head and promoted exercise training -lrb- rather than bed rest -rrb- as a treatment for chronic heart failure . he was instrumental in describing the “ muscle hypothesis ” of heart failure .
	Q: position	Q: discipline	Q: nationality
	A': halfback quarterback	A': basketball player	A': philippines filipino
Seq2Seq	$\tilde{D}'$: george evans -lrb- born 13 december 1994 -rrb- is an english footballer who plays as a midfielder or centre-back for manchester city . he was a **quarterback halfback** in the manchester .	andrei UNK -lrb- born 1975 in satu mare romania is a retired romanian aerobic gymnast **basketball** he had a successful career winning four world championships medals -lrb- two gold , one silver , and one bronze , after his retirement in 1997 he went with to germany where he works as a gymnastics coach at the UNK **basketball** club	andrew justin stewart coats -lrb- born 1 february 1958 is an **filipino** – british academic cardiologist who has particular interest in the management of heart failure . his research turned established teaching on its head and promoted exercise training -lrb- rather than bed rest -rrb- as a treatment for chronic heart failure . he was instrumental in describing the “ muscle hypothesis ” of **philippines**
With gold template (SMG(g))	$\tilde{D}'$: george evans -lrb- born 13 december 1994 -rrb- is an english footballer who plays as a **halfback and quarterback** for manchester city .	$\tilde{D}'$: andrei UNK -lrb- born 1975 in satu mare , romania -rrb- is a retired romanian **basketball player** . he had a successful career winning four world championships medals -lrb- two gold , one silver , and one bronze -rrb- after his retirement in 1997 he went with to germany where he works as a **basketball coach** at the UNK **basketball club** in hanover .	andrew justin stewart coats -lrb- born 1 february 1958 -rrb- is an **filipino** academic cardiologist who has particular interest in the management of heart failure . his research turned established teaching on its head and promoted exercise training -lrb- rather than bed rest -rrb- as a treatment for chronic heart failure . he was instrumental in describing the “ muscle hypothesis ” of heart failure .
With predicted template (SMG(p))	$\tilde{D}'$: george evans -lrb- born 13 december 1994 -rrb- is an **halfback** footballer who plays as a midfielder **or quarterback** for manchester city .	andrei UNK -lrb- born 1975 in satu mare , romania -rrb- is a retired romanian **basketball player** . he had a successful career winning four world championships medals -lrb- two gold , one silver , and one bronze -rrb- after his retirement in 1997 he went with to germany where he works as a gymnastics coach at the UNK gymnastic club in hanover .	$\tilde{D}'$: andrew justin stewart coats -lrb- born 1 february **philippines** academic cardiologist who has particular interest in the management of heart failure . his research turned established teaching on its head and promoted exercise training -lrb- rather than bed rest -rrb- as a treatment for chronic heart failure . he was instrumental in describing the “ muscle hypothesis ” of heart failure .

Table 5: The example generated cases of competing methods. The underlined tokens are gold answer-related tokens. The bold tokens in the “Input” row are predicted answer-related tokens. In the other three rows, the bold tokens are the modified tokens that are related to the given new answer A'.

achieve, because it requires both to select the correct answer-related tokens and to generate the correct words for the answer A.

5.2 Overall Performance

We compare our method (SMG) with a baseline method (Seq2Seq). In Seq2Seq, the difference with SMG is that the decoder part is a conventional decoder that completely generates the modified document $\tilde{D}'$ ignoring the *context template*. The overall performance is shown in Table 3.

In Table 3, we see that our SMG method has outperformed the Seq2Seq baseline in nearly all evaluation metrics, no matter whether the *context template* applied to the decoding phase is gold or predicted. Especially, in the two most important metrics for the performance of controllable text edition: iBLEU and diff-BLEU ratio, our model has achieved a significantly higher score than competing methods. These results show that our method is effective in controllable text edition.

The human evaluation results are also listed in Table 3. The inter-rater agreements are all acceptable (> 0.85) due to Krippendorff's principle (2004). According to the human evaluation, when we are using the gold template for partially generating, both the correctness and the fluency of the partially generated text $\tilde{D}'$ are better than using the predicted template, which is also consistent with our intuition. Note that the perplexity score and the fluency score of Seq2Seq are the best of all the three methods; this is because in the partially generated text, the end position of each blank may not fit very well with the next word sometimes, although we have trained an EOA tag.

Table 4 shows the experiments evaluating the selection of answer-related words. We can see that our SMG model has a higher BLEU (predicted template) score than the Seq2Seq model. This fact shows that partially training the blank-filling tokens helps for the selection of answer-related tokens. Also, our model SMG has achieved a higher *answer* F_1 score (0.68) than competing methods.

5.3 Case Study

We have listed some examples of the modified document $\tilde{D}'$ generated by the three competing methods (Seq2Seq, SMG(g), and SMG(p)) in Table 5. We can see that although the answer-related words are already masked out, Seq2Seq still always generates the words in the original answer A and tends to mix up the words in A and the changed answer A' (like in the second example, Seq2Seq mixed “gymnastic” and “basketball” together.) Also, Seq2Seq cannot precisely change everywhere what should be modified, for example, in the second example, Seq2Seq failed to change “gymnastic coach” to “basketball coach”. In the SMG methods, when we are using the gold template for partial generation, the model is able to generate the correct words aiming to change Q's answer to A'. Although there is still some risk to have some answer-related tokens left unchanged due to the error in the predicted template, the *context tokens* in the predicted template are ensured to be generated. Therefore, our model with predicted template is more fit for NLP products than Seq2Seq.

6 Conclusion

In this paper, we proposed a novel task, the goal of which is to modify some content of a given text to make the answer of a text-related question change to a given new answer. This task is very useful in many real-world tasks, like contract editing. We constructed a test set for evaluation and released this test set. We also proposed a novel model SMG to solve this task. In SMG, we first use a selector-predictor structure to select the answer-related tokens in the input document, then we use a novel partial generation technique to generate the modified document without changing answer-unrelated tokens in the original document. The experiments proved the effectiveness of our model.

Acknowledgments

This work was supported by the EPSRC grant "Unlocking the Potential of AI for English Law". We also acknowledge the use of Oxford's Advanced Research Computing (ARC) facility, of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1), and of GPU computing support by Scan Computers International Ltd.

References

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, Rerank and Rewrite: Soft Template Based Neural Summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161. Association for Computational Linguistics.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. 2018. Isolating Sources of Disentanglement in Variational Autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling Language Models to Fill in the Blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online. Association for Computational Linguistics.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. WikiAtomicEdits: A Multilingual Corpus of Wikipedia Edits for Modeling Language and Discourse. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 305–315. Association for Computational Linguistics.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better Text Generation via Filling in the _. In *International Conference on Learning Representations*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating Sentences by Editing Prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A Retrieve-and-Edit Framework for Predicting Structured Outputs. In *Advances in Neural Information Processing Systems*, pages 10052–10062.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural Machine Translation Decoding with Terminology Constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations*, 2(5):6.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Chris Hokamp and Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546. Association for Computational Linguistics.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved Lexically Constrained Decoding for Translation and Monolingual Rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 839–850. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled Representation Learning for Non-Parallel Text Style Transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by Factorising. *arXiv preprint arXiv:1802.05983*.

Reinhard Kneser and Hermann Ney. 1995. Improved Backing-Off for M-gram Language Modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184. IEEE.

Klaus Krippendorff. 2004. Content Analysis: An Introduction to Its Methodology Thousand Oaks. *Calif.: Sage*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-Scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-Text Generation by Structure-Aware Seq2Seq Learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2020. Unsupervised Paraphrasing by Simulated Annealing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 302–312. Association for Computational Linguistics.

Emile Mathieu, Tom Rainforth, Siddharth Narayanaswamy, and Yee Whye Teh. 2018. Disentangling Disentanglement in Variational Autoencoders. *arXiv preprint arXiv:1812.02833*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan, and Greg Ver Steeg. 2018. Invariant Representations without Adversarial Training. In *Advances in Neural Information Processing Systems*, pages 9084–9093.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar Encoder-Decoder for Neural Conversation Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1329–1338.

Sheena Panthaplackel, Miltiadis Allamanis, and Marc Brockschmidt. 2020. Copy That! Editing Sequences by Copying Spans. *arXiv preprint arXiv:2006.04771*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A Challenge dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203. Association for Computational Linguistics.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2019. Adversarial Decomposition of Text Representation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 815–825. Association for Computational Linguistics.

Lei Sha. 2020. Gradient-guided Unsupervised Lexically Constrained Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8692–8703. Association for Computational Linguistics.

Lei Sha, Oana-Maria Camburu, and Thomas Lukasiewicz. 2021. Learning from the Best: Rationalizing Predictions by Adversarial Information Calibration. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference, February 2–9, 2021*. AAAI Press.

Lei Sha and Thomas Lukasiewicz. 2021. Multi-type Disentanglement Without Adversarial Training. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-Planning Neural Text Generation From Structured Data. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Jan Stühmer, Richard E Turner, and Sebastian Nowozin. 2019. Independent Subspace Analysis for Unsupervised Learning of Disentangled Representations. *arXiv preprint arXiv:1909.05063*.

Hong Sun and Ming Zhou. 2012. Joint Learning of a Dual SMT System for Paraphrase Generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–42. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2019. Response Generation by Context-Aware Prototype Editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7281–7288.

Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. 2018. Learning to Represent Edits. *arXiv preprint arXiv:1810.13337*.

Changchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. 2020. A Survey on Machine Reading Comprehension—Tasks, Evaluation Metrics and Benchmark Datasets. *Applied Sciences*, 10(21):7640.

Wanrong Zhu, Zhiting Hu, and Eric Xing. 2019. Text Infilling. *arXiv preprint arXiv:1901.00158*.

Appendices

A Human Evaluation Question Marks

Our annotators were asked the following questions, in order to assess the correctness and fluency of the modified document provided by our model.

A.1 Correctness of modified document

Q: Do you think the modification of the document is correct so that it can make the question answer pair $\langle Q, A' \rangle$ true? (For partially correct cases: Partially correct means some places are changed to the new answer, and some places keep the old answer. In this case, only all places (that need to be changed) have been changed can be taken as correct.)

Please choose “Yes” or “No”.

After all human annotators finished their work, the correctness score is calculated by dividing the number of “Yes” by the total number of examples.

A.2 Fluency

Q: How fluent do you think the modified document is?

Please choose a score according to the following description. Note that the score is not necessarily an integer, you can give scores like 3.2 or 4.9 , if you deem appropriate.

- 5: Very fluent.
- 4: Highly fluent.
- 3: Partial fluent.
- 2: Very unfluent.
- 1: Nonsense.

B Experiment Details

The word embedding size is 300. The BiLSTM in the selector model has the following hyperparameters: hidden size = 200. The hidden size of decoder’s LSTM cell is 200. The rest hyperparameters has the following values: $\lambda_r = 1.0$, $\lambda_{\text{eoa}} = 10$. The hyperparameters are obtained by grid search, the search scopes are $\lambda_r \in [0.0, 2.0]$ with step size

0.2, $\lambda_{\text{eoa}} \in [1, 20]$ with step size 1, the hidden size are searched in $[100, 500]$ with step size 50. The best hyperparameters are selected when the model achieves the highest answer's F_1 in the development set. The total parameter size is $72M$. Each training epoch costs about 1.5 hours on V100 GPU.

Grammar-Based Patches Generation for Automated Program Repair

Yu Tang[1]*, **Long Zhou**[2], **Ambrosio Blanco**[2], **Shujie Liu**[2],
Furu Wei[2], **Ming Zhou**[2], **Muyun Yang**[1]
[1]Harbin Institute of Technology
`19s103120@stu.hit.edu.cn` `yangmuyun@hit.edu.cn`
[2]Microsoft Research Asia
`{lozhou,ambrosio.blanco,shujliu,fuwei,mingzhou}@microsoft.com`

Abstract

Automated program repair (APR) aims to find an automatic solution to program language bugs without human intervention, and it can potentially reduce debugging costs and improve software quality. Conventional approaches adopt learning-based methods such as sequence-to-sequence models for the patches generation. However, they tend to ignore the code structure information and suffer from grammar and syntax errors. To consider the grammar and syntax information, in this paper, we propose a grammar-based rule-to-rule model, which regards the repair process as the transformation of grammar rules, and leverages two encoders modeling both the original token sequence and the grammar rules, enhanced with a new tree-based self-attention. Besides, to guarantee grammar correctness, we employ a grammatically restricted inference method to generate each grammar rule in a legally constrained sub-search-space considering the generated previous rules. Experimental evaluations on a Java dataset demonstrate that the proposed approach significantly outperforms the state-of-the-art baselines in terms of generated code accuracy.

1 Introduction

Advances in machine learning and the availability of large corpora of source code have led to growing developments of software engineers. Researchers have exploited machine learning to automate several development and maintenance tasks, such as code completion (Svyatkovskiy et al., 2020), comment generation (Hu et al., 2018), code search (Gu et al., 2018), bug localization (Zheng et al., 2016) and fixing (Tufano et al., 2018). It's worth noting that localizing and fixing bugs is known to be an effort-prone and time-consuming task for software developers (Weiss et al., 2007). Hence, several works recently focused on automatically repairing programs without human intervention, which can improve programmer productivity and software quality (Tufano et al., 2018; Chen et al., 2019; Vasic et al., 2019; Yasunaga and Liang, 2020).

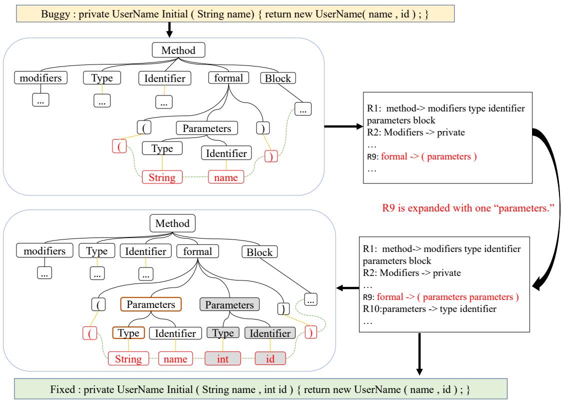
```
Buggy : private UserName Initial ( String name) { return new UserName( name , id ) ; }
```
```
R1:  method-> modifiers type identifier parameters block
R2: Modifiers -> private
…
R9: formal -> ( parameters )
…
```

```
R1:  method-> modifiers type identifier parameters block
R2: Modifiers -> private
…
R9: formal -> ( parameters parameters )
R10:parameters -> type identifier
…
```
```
Fixed : private UserName Initial ( String name , int id ) { return new UserName ( name , id ) ; }
```

Figure 1: Schematic diagram of our model. Source code is parsed into AST (Abstract Syntax Tree) and then into a sequence of rules. Each rule consists of one head token (parent node) and several tail tokens (children nodes).

Automated program repair (APR) research is very active and dominated by techniques based on static analysis (Mechtaev et al., 2016) and dynamic analysis (Wen et al., 2018). Meanwhile, APR is also challenging because fixing bugs is a difficult task. Previous approaches mainly are based on a relatively limited and manually-crafted set of fixing patterns, which need substantial effort and expertise (Saha et al., 2017; Jin et al., 2011; Nguyen et al., 2013). Moreover, these techniques can only fix bugs in a given language or a specific application domain and lack scalability and maintainability.

Very recently, deep learning based approaches, such as sequence-to-sequence learning (Sutskever et al., 2014), are proposed to automatically repair program by learning from massive open-source projects with numerous bug fixes (Tufano et al.,

---

*Contribution during internship at Microsoft Research

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1300–1305
August 1–6, 2021.

2018; Chen et al., 2019). However, these sequence-to-sequence methods ignore codes' structure information because they are designed for natural language which is significantly different from program language with strict syntactic and grammatical requirements. Hence, the generated patches of these methods suffer from grammar and syntax errors.

To address the problem, in this paper, we propose a novel grammar-based approach to automatically generate fixed patches for automated program repair. More specifically, instead of using a sequence-to-sequence model with code sequence, we first introduce a grammar-based rule-to-rule model, which regards the repair process as the transformation of code grammar rules, as shown in Figure 1. Second, to guarantee the grammatical and syntactic correctness, we not only introduce a rule encoder (together with a token encoder) to directly extract grammatical features but also employ a grammatically restricted inference method to generate the fixed code. Experimental results conducted on BFPs dataset (Tufano et al., 2018) of CodeXGLUE (Lu et al., 2021) demonstrate that the proposed grammar-based approach significantly outperforms the state-of-the-art baselines.

2 Related Work

Automatic program repair, consisting of automatically finding a solution to software bugs without human intervention, has recently received significant attention (Tufano et al., 2018; Chen et al., 2019; Yasunaga and Liang, 2020). Traditional approaches generate patch candidates by first applying a predefined set of mutation operators on the fault space. They then deploy some heuristics (Qi et al., 2014) to search among these candidates for a correct patch that passes all given test cases (Weimer et al., 2009; Qi et al., 2014). Although these methods have shown to be able to fix a wide range of bugs, they can only fix bugs in a given language or a specific application domain (Saha et al., 2017; Jin et al., 2011; Nguyen et al., 2013).

Inspired by the development of deep learning in a variety of problems, researchers attempt to employ deep learning based approaches to automatically repair code by learning from massive buggy-fixes pairs (Tufano et al., 2018; Chen et al., 2019; Vasic et al., 2019; Guo et al., 2020). Tufano et al. (2018) first presented an end-to-end approach to fix program language based on sequence-to-sequence learning. They released datasets of APR and evaluated the performance of neural machine translation. SequenceR (Chen et al., 2019) employed copy mechanism based on line level. DeepFix (Gupta et al., 2017) and SynFix (Bhatia et al., 2018) repair syntax errors in programs using neural program representations. Despite their effectiveness, the generated patches of these methods suffer from grammar and syntax errors.

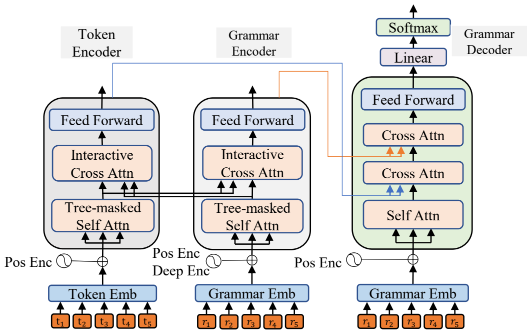

Figure 2: The framework of our proposed model.

Compared to previous works, our proposed approach has three advantages: (1) we employ the state-of-the-art Transformer model as the skeleton of code repair model; (2) we incorporate the grammar information of fixing ingredients into our model by using token and grammar encoders; (3) we propose a grammar-guided inference method to guarantee the grammar correctness.

3 Our Approach

In this section, we will first introduce a grammar-guided rule-to-rule model (Section 3.1), and then present a grammar-constrained inference method (Section 3.2).

3.1 Grammar-Driven Model

Our grammar-guided model, which is based on the state-of-the-art Transformer model (Vaswani et al., 2017), has a token encoder, a grammar encoder, and a grammar decoder, as shown in Figure 2.

3.1.1 Token and Grammar Encoder

To model token representations and grammar structures, we employ token encoder and grammar encoder to model code unit and code grammar, respectively. The two encoders have similar model architecture and different inputs, which are token sequence $\{t_1, t_2, ..., t_m\}$ and rule sequence $\{r_1, r_2, ..., r_n\}$. Considering the difference of sequence and tree structure, besides conventional sinusoidal positional embedding (Vaswani et al., 2017), we also introduce a **depth embedding** to enhance of capacity of modeling tree structure for the grammar encoder. As for the model structure, the first sub-layer of encoders is **tree-masked self-attention**, the second sub-layer is **interactive cross-attention**, and the last sub-layer is a feed-forward layer.

Depth Embedding We extract the depth information of each rule in the corresponding abstract syntax tree (AST). For the example in Figure 1, the depth of rule *modifiers* $\rightarrow$ *private* is depending on the depth of its head token *modifiers*, which counts 1 in this example. The final position embedding is the sum of the sinusoidal position encoding and the depth embedding.

Tree-Masked Self-Attention To focus on the local information from directly contacted tokens or rules, we propose tree-based attention applied to one head of multi-head self-attention. Formally, we first build a distance-based mask M_{tree}, where $M_{tree}[i,j] = 0$ if node i is the parent or one of the children of j, else $M_{tree}[i,j] = -e^9$. Then, we employ the proposed M_{tree} to one dot-product attention:

$$\mathrm{ATT}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}} + M_{tree})V \tag{1}$$

Interactive Cross-Attention The goal of interactive cross-attention is to make full use of token information and syntax information interactively. Specifically, given the outputs of tree-masked self-attention in token encoder and grammar encoder, e.g., H_{tok} and H_{gra}, the output of interactive cross-attention can be expressed as:

$$\begin{aligned} O_{tok} &= \mathrm{Attention}(H_{tok}, H_{gra}, H_{gra}) \\ O_{gra} &= \mathrm{Attention}(H_{gra}, H_{tok}, H_{tok}) \end{aligned} \tag{2}$$

where $\mathrm{Attention}(\cdot)$ is the same as standard self-attention in Transformer.

3.1.2 Grammar Decoder

For each layer in the grammar decoder, the lowest sub-layer is the masked multi-head self-attention network, and the top layer is a feed-forward layer, as shown in Figure 2. Moreover, we design three attention strategies to integrates source token and grammar information.

(1) The **standard strategy** is the same as the traditional cross-attention in Transformer. Specifically, the query Q comes from the output of decoder self-attention, and the key-value pair $\{K, V\}$ is transformed only from the output of the grammar encoder. (2) Figure 2 shows the **cascade strategy**, in which we first compute the cross-attention with token encoder, then use the output as the query to calculate the cross-attention with grammar encoder. (3) The **parallel strategy** attends to each encoder independently and then sums up the context vectors. We will compare the three strategies in Section 4.4.

3.2 Grammar-Constrained Inference

Considering the sensitivity and strictness of program language, we further propose a grammar-constrained inference method to guarantee the grammatical correctness of the output. More specifically, we first build an AST in the inference process according to the currently generated rule sequence and maintain an indicator to locate the AST node where the extension is happening. Then, we filter out the unsatisfactory rules whose head token is not the current extending node, by using mask operation in softmax function. Finally, our AST and indicator can be updated for the next prediction. It is worth noting that the node pointed by the indicator is the one that should be expanded as a parent (head token) in the next inference.

Take figure 1 as an example, when R9 has been predicted, the indicator is pointing the node *parameters*. We limit the search space of the next rules, which have to satisfy its head node is *parameters* (marked by a brown border). Then applicable rule with the highest probability (R10 in this case) is chosen and tail tokens *type* and *identifier* of R10 are added into AST. Finally indicator is transferred from *parameters* to *type*, denoting that *type* is the next expanded node.

4 Experiments

4.1 Datasets and Metrics

Datasets We evaluate our approach on BFPs dataset (Tufano et al., 2018), a collection of Java functions on Github Archive. BFPs consists of 58K bug-fixes data and is divided into training, validation and test sets by 8:1:1. We extracted 4.5K grammar constraints from the dataset in total, among which 0.5K are related to vocab and others related to grammar rules.

Metrics Following Tufano et al. (2018), we employ **XMatch**, a metric indicating the percentage of model's outputs that exactly match the reference, including Top XMatch and All XMatch. Top Match only utilizes top 1 of the beams as output, and All Match uses all beams as outputs to match the refer-

Method	Beam	XMatch (Top)	XMatch (All)	XBug
NMT (Tufano et al., 2018)	1	9.22%*	9.22%*	-
	5	-	27.33%*	-
Token-Trans	1	10.03%	10.03%	56
	5	10.32%	24.02%	44
Rule-Trans	1	10.87%	10.87%	31
	5	11.95%	24.94%	14
Our Model	1	11.47%	11.47%	0
	5	13.42%	28.03%	0

Table 1: Main results of our model on BFPs. Results marked with * are from Tufano et al. (2018).

ence. The correlation coefficient analysis of (Ren et al., 2020) also shows that human evaluation is more correlated with **XMatch** than n-gram matching for code repair task. Besides, we employ **XBug** as the auxiliary metric to evaluate the grammatical correctness of output, which examines basic grammar structure of output, like symbol usage and context matching.

4.2 Experimental Setting

Our model is built on PyTorch and trained on 4 GPUs of TITAN XP for 7 hours. We parse source code into AST with *tree sitter*[1], and use the same Transformer setting as Vaswani et al. (2017). The hidden size is 512. The layer of both two encoders and decoder is 6. The dropout probability is 0.5. The number of trainable parameters in our best model is 49M. We test system performance with beam size = 1 and 5, and the latter is used as default.

4.3 Main Results

Table 1 shows the results of all models. The first line is the performance of the previous RNN-based NMT model (Tufano et al., 2018) on BFPs. Token-Trans is built on the Transformer model using sequences of source code as input, while Rule-Trans employs grammar sequences as input and output.

Top Match Token-Trans and Rule-Trans models get the performance of 10.32% and 11.95% with beam search respectively, which first demonstrates the model superiority by using grammar rules. Furthermore, our proposed model outperforms Token-Trans and Rule-Trans by 3.1% and 1.5% respectively, indicating the effectiveness of our proposed grammar-aware modules and inference method in the APR task.

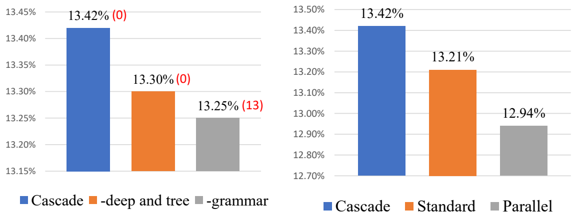

(a) Ablation (b) Strategies

Figure 3: Comparison on different modules (a) and strategies (b). "deep and tree" means the depth embedding and tree-based attention, and "grammar" demotes the grammar-constrained inference method.

All Match We verify All Match metric to compare with previous work (Tufano et al., 2018), which judges correct model predictions if any one of the beams can fix the buggy code. Results show that our proposed model suggests 28.03% correct patches with beam search, and our model performs better with 2.2% and 0.7% improvements than (Tufano et al., 2018) when beam is 1 and 5 respectively.

XBug We further analyze the grammatical quality of different models. Token-Trans model suffers from grammatical errors with 56 samples in the test set, while in Rule-Trans model, the number decreases to 31. Due to the ability to model grammar rules and guide rule generation of our approach, our proposed model can effectively avoid grammatical errors, guaranteeing the grammar correctness of generated patches.

4.4 Effect of Modules and Strategies

In this section, we will evaluate the contribution of different modules and compare the three combination strategies described in Section 3.1.2.

Figure 3 shows the XMatch results of different models, and we also list the XBug number in brackets. The results in Figure 3 (a) show that all of the proposed methods have positive effects. It's worth noting that the performance significantly drops in terms of XBug if we remove the grammar-constrained inference method. Compare different combination strategies in Figure 3 (b), parallel strategy performs worse than other strategies, and the underlying reason is that concatenating token and grammar sequences results in too long sentences. Besides, the cascade strategy behaves best because it can make full use of the token information and the grammar information provided by encoders.

---

[1] http://tree-sitter.github.io/tree-sitter/

5 Conclusion

In this paper, we propose a grammar-guided end-to-end approach for automated program repair. Particularly, we introduce three structure-aware modules and three combination strategies, and present a grammar-based inference algorithm to guarantee grammar correctness of generated patches. Experiments on BFPs dataset demonstrate that Grammar-based system performs better than Token-based system for both model learning and grammar correctness. Moreover, system that simultaneously modeling grammar and its inside token information showed great potentiality in our works. Besides, the advantage of grammar-constrained inference inspires us to explore more about the possibility of combining grammar constraints with NLP model.

References

Sahil Bhatia, Pushmeet Kohli, and Rishabh Singh. 2018. Neuro-symbolic program corrector for introductory programming assignments. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 60–70. IEEE.

Zimin Chen, Steve James Kommrusch, Michele Tufano, Louis-Noël Pouchet, Denys Poshyvanyk, and Martin Monperrus. 2019. Sequencer: Sequence-to-sequence learning for end-to-end program repair. *IEEE Transactions on Software Engineering*.

Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. 2018. Deep code search. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 933–944. IEEE.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*.

Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish Shevade. 2017. Deepfix: Fixing common c language errors by deep learning. In *Proceedings of the aaai conference on artificial intelligence*, volume 31.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 200–20010. IEEE.

Guoliang Jin, Linhai Song, Wei Zhang, Shan Lu, and Ben Liblit. 2011. Automated atomicity-violation fixing. In *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*, pages 389–400.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*.

Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. 2016. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *Proceedings of the 38th international conference on software engineering*, pages 691–701.

Hoang Duong Thien Nguyen, Dawei Qi, Abhik Roychoudhury, and Satish Chandra. 2013. Semfix: Program repair via semantic analysis. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 772–781. IEEE.

Yuhua Qi, Xiaoguang Mao, Yan Lei, Ziying Dai, and Chengsong Wang. 2014. The strength of random search on automated program repair. In *Proceedings of the 36th International Conference on Software Engineering*, pages 254–265.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. 2020. Codebleu: a method for automatic evaluation of code synthesis. *arXiv preprint arXiv:2009.10297*.

Ripon K Saha, Yingjun Lyu, Hiroaki Yoshida, and Mukul R Prasad. 2017. Elixir: Effective object-oriented program repair. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 648–659. IEEE.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112.

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. Intellicode compose: Code generation using transformer. *arXiv preprint arXiv:2005.08025*.

Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. 2018. An empirical investigation into learning bug-fixing patches in the wild via neural machine translation. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 832–837.

Marko Vasic, Aditya Kanade, Petros Maniatis, David Bieber, and Rishabh Singh. 2019. Neural program repair by jointly learning to localize and repair. *arXiv preprint arXiv:1904.01720*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. 2009. Automatically finding patches using genetic programming. In *2009 IEEE 31st International Conference on Software Engineering*, pages 364–374. IEEE.

Cathrin Weiss, Rahul Premraj, Thomas Zimmermann, and Andreas Zeller. 2007. How long will it take to fix this bug? In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, pages 1–1. IEEE.

Ming Wen, Junjie Chen, Rongxin Wu, Dan Hao, and Shing-Chi Cheung. 2018. Context-aware patch generation for better automated program repair. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 1–11. IEEE.

Michihiro Yasunaga and Percy Liang. 2020. Graph-based, self-supervised program repair from diagnostic feedback. *arXiv preprint arXiv:2005.10636*.

Wei Zheng, Desheng Hu, and Jing Wang. 2016. Fault localization analysis based on deep neural network. *Mathematical Problems in Engineering*, 2016.

Manual Evaluation Matters: Reviewing Test Protocols of Distantly Supervised Relation Extraction

Tianyu Gao[1], **Xu Han**[2], **Yuzhuo Bai**[2], **Keyue Qiu**[2], **Zhiyu Xie**[2], **Yankai Lin**[3]
Zhiyuan Liu[2*], **Peng Li**[3], **Maosong Sun**[2] **and Jie Zhou**[3]
[1]Department of Computer Science, Princeton University
[2]Department of Computer Science and Technology, Tsinghua University
[3]Pattern Recognition Center, WeChat AI, Tencent Inc
`tianyug@princeton.edu`
`{hanxu17, qky18, byz18, xiezy19}@mails.tsinghua.edu.cn`

Abstract

Distantly supervised (DS) relation extraction (RE) has attracted much attention in the past few years as it can utilize large-scale auto-labeled data. However, its evaluation has long been a problem: previous works either take costly and inconsistent methods to manually examine a small sample of model predictions, or directly test models on auto-labeled data—which, by our check, produce as much as 53% wrong labels at the entity pair level in the popular NYT10 dataset. This problem has not only led to inaccurate evaluation, but also made it hard to understand where we are and what's left to improve in the research of DS-RE. To evaluate DS-RE models more credibly, we build manually-annotated test sets for two DS-RE datasets, NYT10 and Wiki20, and thoroughly evaluate several competitive models, especially the latest pre-trained ones. The experimental results show that the manual evaluation can indicate very different conclusions from automatic ones, especially some unexpected observations, e.g., pre-trained models can achieve dominating performance while being more susceptible to false-positives compared with previous methods. We hope that both our manual test sets and observations can help advance future DS-RE research.[1]

1 Introduction

Relation extraction (RE) aims at extracting relational facts between entities from the text. One crucial challenge for building an effective RE system is how to obtain sufficient annotated data. To tackle this problem, Mintz et al. (2009) propose distant supervision (DS) to generate large-scale auto-labeled data by aligning relational facts in knowledge graphs (KGs) to text corpora, with the core assumption that one sentence mentioning two entities is likely to express the relational facts between the two entities from KGs.

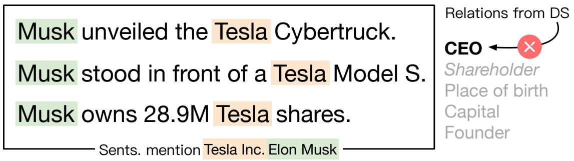

Figure 1: Typical errors made by DS evaluation. In the figure, DS labels the bag with only the relation `CEO`, while none of the sentences express the relation. Also, it misses a correct relation `shareholder` due to the incompleteness of the knowledge graphs.

As DS can bring hundreds of thousands of auto-labeled training instances for RE without any human labor, DS-RE has been widely explored in the past few years (Riedel et al., 2010; Hoffmann et al., 2011; Zeng et al., 2015; Lin et al., 2016; Feng et al., 2018; Vashishth et al., 2018) and has also been widely extended to other related domains, such as biomedical information extraction (Peng et al., 2017; Quirk and Poon, 2017) and question answering (Bordes et al., 2015; Chen et al., 2017).

Although DS-RE has achieved great success, we identify one severe problem for the current DS-RE research—its evaluation. Existing works usually take two kinds of evaluation methods following Mintz et al. (2009): *held-out evaluation*, which directly uses DS-generated test data to approximate the trend of model performance, and *human evaluation*, which manually checks the most confident relational facts predicted by DS-RE models. Since manually checking is costly, most works with human evaluation only examine a small proportion of the predictions. Moreover, different works may sample different splits of data, making human evaluation inconsistent across the literature. Most recent studies even skip the human evaluation for the above factors and only take the held-out one.

---

[*] Corresponding author e-mail: liuzy@tsinghua.edu.cn

[1] Our code and data are publicly available at `https://github.com/thunlp/opennre`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1306–1318
August 1–6, 2021. ©2021 Association for Computational Linguistics

However, the held-out evaluation can be quite noisy: there are many false-positive cases, where the sentences do not express the auto-labeled relations at all; besides, due to the incompleteness of KGs, DS may miss some relations, just as shown in Figure 1. After checking $9,744$ sentences in the held-out set of NYT10 (Riedel et al., 2010), the most popular DS-RE dataset, we found that about 53% of the entity pairs are wrongly labeled, emphasizing the need for a more accurate evaluation.

To make DS-RE evaluation more credible and alleviate the trouble of manual checking for later work, we build human-labeled test sets for two DS-RE datasets: NYT10 (Riedel et al., 2010) and Wiki20 (Han et al., 2020). For NYT10, we manually annotate sentences with positive DS relations in its held-out test set. We also use a fine-tuned BERT-based (Devlin et al., 2019) RE model to predict all “N/A” (not applicable) sentences, and manually label the top $5,000$ sentences scored as having a relation. Additionally, we merge some unreasonably split relations and reduce the number of relation types from 53 to 25. For Wiki20 dataset, we utilize both the relation ontology and human-labeled instances of an existing supervised dataset Wiki80 (Han et al., 2019) for the test, and then re-organize the DS training data accordingly.

Based on the newly-constructed benchmarks, we carry out a thorough evaluation of existing DS-RE methods, as well as incorporating recently advanced pre-trained models like BERT (Devlin et al., 2019). We found that our manually-annotated test sets can indicate very different conclusions from the held-out one, especially with some surprising observations: (1) although pre-trained models lead to large improvements, they also suffer from false-positives more severely, probably due to the pre-encoded knowledge they have (Petroni et al., 2019); (2) existing DS-RE denoising strategies that have been proved to be effective generally do not work for pre-trained models, suggesting more efforts needed for DS-RE in the era of pre-training. To conclude, our main contributions in this work are:

- We provide large human-labeled test sets for two DS-RE benchmarks, making it possible for later work to evaluate in an accurate and efficient way.
- We thoroughly study previous DS-RE methods using both held-out and human-labeled test sets, and find that human-labeled data can reveal inconsistent results compared to the held-out ones.
- We discuss some novel and important observations revealed by manual evaluation, especially with respect to pre-trained models, which calls for more research in these directions.

2 Related Work

Relation extraction is an important NLP task and has gone through significant development in the past decades. In the early days, RE models mainly take statistical approaches, such as pattern-based methods (Huffman, 1995; Califf and Mooney, 1997), feature-based methods (Kambhatla, 2004; Zhou et al., 2005), graphical methods (Roth and Yih, 2002), etc. With the increasing computing power and the development of deep learning, neural RE methods have shown a great success (Liu et al., 2013; Zeng et al., 2014; Zhang and Wang, 2015; Zhang et al., 2017). Recently, pre-trained models like BERT (Devlin et al., 2019) have dominated various NLP benchmarks, including those in RE (Baldini Soares et al., 2019; Zhang et al., 2019b). All these RE methods focus on training models in a supervised setting and require large-scale sufficient human-annotated data.

To generate large-scale auto-labeled data without human effort, Mintz et al. (2009) first use DS to label sentences mentioning two entities with their relations in KGs, which inevitably brings wrongly labeled instances. To handle the noise problem, Riedel et al. (2010); Hoffmann et al. (2011); Surdeanu et al. (2012) apply multi-instance multi-label training in DS-RE. Following their settings, later research mainly takes on two paths: one aims at selecting informative sentences from the noisy dataset, using heuristics (Zeng et al., 2015), attention mechanisms (Lin et al., 2016; Han et al., 2018c; Zhu et al., 2019), adversarial training (Wu et al., 2017; Wang et al., 2018; Han et al., 2018a), and reinforcement learning (Feng et al., 2018; Qin et al., 2018); the other incorporates external information like KGs (Ji et al., 2017; Han et al., 2018b; Zhang et al., 2019a; Hu et al., 2019), multilingual corpora (Verga et al., 2016; Lin et al., 2017; Wang et al., 2018), as well as relation ontology and aliases (Vashishth et al., 2018). Recently, pre-trained DS-RE models have also been explored, including both domain-general (Alt et al., 2019; Xiao et al., 2020) and domain-specific (Amin et al., 2020) models. Some other latest works (Peng et al., 2020) utilize DS data for intermediate pre-training in order to boost supervised RE tasks.

As mentioned in our introduction, the evalua-

Dataset	#rel	Train			Validation			Test		
		#facts	#sents	N/A	#facts	#sents	N/A	#facts	#sents	N/A
NYT10†	53	18,409	522,611	74%	-	-	-	1,950	172,448	96%
NYT10	25	17,137	417,893	80%	4,062	46,422	80%	3,899 (1,940)	9,744 (157,859)	32% (96%)
Wiki20†	454	203,176	1,050,246	48%	4,333	29,145	48%	4,333	28,897	48%
Wiki20	81	157,740	698,721	59%	17,485	64,607	73%	56,000	137,986	25%

Table 1: The statistics of the datasets used for our benchmarks, including both the original (†) and our modified versions. We list the numbers of relations (#rel), relational facts (#facts) and sentences (#sents), and N/A rate (N/A) for these datasets. For NYT10, numbers in brackets are for the held-out test, otherwise for bag-level manual test.

tion of DS-RE has long been a problem, especially since many existing methods solely rely on auto-labeled test data. Some preliminaries have noticed this problem: Jiang et al. (2018); Zhu et al. (2020) also annotate the test set of NYT10, yet Jiang et al. (2018) only sample $2,040$ sentences from it, and Zhu et al. (2020) discard all N/A data from DS, which are an important part of DS evaluation, and assume that the original held-out data have either the DS relations or no relation at all, while we find that a large proportion of held-out data actually express some other relations; Li et al. (2020) propose active testing, an iterative method to correct the bias of DS evaluation. However, it still requires consistent human efforts during each evaluation phase. To the best of our knowledge, our work, building benchmarks with large-scale manually-labeled test data, conducts the most comprehensive human evaluations of DS-RE methods so far.

3 DS-RE Datasets

In this section, we introduce the way we build the manually-annotated test sets for NYT10 (Riedel et al., 2013) and Wiki20 (Han et al., 2020). We show the statistics of these datasets in Table 1.

3.1 NYT10 Dataset

NYT10 is constructed by aligning facts from the FreeBase (Bollacker et al., 2008) with the New York Times (NYT) corpus (Sandhaus, 2008). The original NYT10 dataset contains 53 relations (including N/A). After thoroughly examining the dataset, we found that (1) there are many duplicate instances in the dataset, (2) there is no public validation set, and some previous works directly take the test set to tune the model, and (3) the relation ontology is not reasonable for an RE task. Therefore, we first clean the dataset by removing duplicate sentences, split a validation set, and then merge some of the relations as described below.

A New Relation Ontology One example of NYT10's improper relation ontology is the relations related to state/province capitals. There are 12 such relations in the original dataset, representing region capitals of different countries, while some of these relations even do not have instances in the test set. We merge these 12 relations as one unified relation `/location/region/capital`, and we also merge 3 relations related to organization locations as `/business/location`. Besides, we delete relations that only show up in either the training set or the test set (most of which have very few sentences). In the end, we get 25 relations in the new dataset. We provide the detailed relation list of the new dataset in Appendix A.

Interestingly, we found that training models with the new dataset leads to a slight performance degeneration, as shown in Table 2, which is counter-intuitive (since merging classes usually makes the task easier). We suspect that the original relation ontology provides heuristics for the model. For example, models can learn from the original ontology that every sentence with a "US state" as the head entity expresses the fine-grained relation `/location/us_state/capital`, which is a shortcut that cannot be acquired with the merged relation `/location/region/capital`. This shows the bias of the original NYT10 dataset being inappropriately exploited by models.

Annotated Test Set The original NYT10 dataset only provides an auto-labeled test set for the held-out evaluation. Based on the original test set, we manually annotate all sentences that have a positive DS label. In addition to that, we also fine-tune a BERT model (as described in §4.2) to predict the relations of all sentences originally labeled as N/A, and annotate the $5,000$ sentences with the highest predicted scores of non-N/A relations. For each sentence, we ask 4 annotators to decide whether it expresses one or more relations among 25 rela-

Model	Original AUC	Original F1	New AUC	New F1
PCNN+bag+AVG	31.8	38.6	28.4	35.8
PCNN+bag+ATT	33.6	40.2	32.2	39.1

Table 2: AUC and micro F1 (%) of model predictions on the original NYT10 relation ontology and our new NYT10 relation ontology (using held-out test).

tions. Note that one sentence may have multiple relations. Specially, we do not provide any relation suggestions using DS labels or model predictions to annotators, in order to avoid the annotators being biased by the external information.

When aggregating the final annotation results, we consider each relation for each sentence independently. The sentence is regarded to express one relation if more than half of the annotators labeled it with this relation. If one sentence gets no votes for any positive relations in the above process, then it is labeled as N/A. For annotation conflicts (i.e., no candidate relation gets more than half votes), the authors manually annotate these sentences.

Finally, we obtain the human-labeled test set with $9,744$ sentences, 32% of which are N/A instances. It contains $5,174$ entity pairs and $3,899$ manually-verified relational facts in total. After comparing it with the corresponding original DS-generated labels, we found that at the fact level, the DS annotations only have a precision of 69.1% and a recall of 33.9%. At the entity pair level, the accuracy of DS labels is only 47.1%. This emphasizes the need to take the human-annotated test set for more accurate evaluation in DS-RE.

3.2 Wiki20 Dataset

Han et al. (2020) construct the Wiki20 dataset by aligning the English Wikipedia corpus with Wikidata (Vrandečić and Krötzsch, 2014). To provide an annotated test set for it, we utilize Wiki80 (Han et al., 2019), a supervised RE dataset with 80 relations from Wikidata. We re-organize Wiki20 by adopting the same relation ontology as Wiki80 and re-splitting the train/validation/test sets, while taking sentences in Wiki80 as the test set. We make sure that there is no overlap of entity pairs among the three splits, to avoid any information leakage.

Note that Wiki20 is quite different from NYT10: NYT10 labels a sentence as "N/A" if the entity pair in the sentence does not have a relation in the KG. On the contrary, Wiki20 labels entity pairs with a relation outside its relation ontology as "N/A". In other words, "N/A" instances in Wiki20 express *unknown* relations instead of *no* relation.

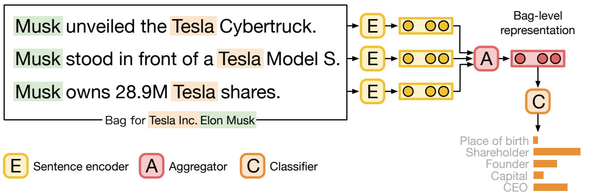

Figure 2: An illustration for a typical multi-instance multi-label (bag-level) model. The model aims to predict relation probabilities for entity pairs, instead of sentences, which is usually accomplished by aggregating a bag representation and doing classification over it.

4 DS-RE Models

In this section, we elaborate the multi-instance multi-label framework for DS-RE, and introduce models we evaluate, including both previous methods and the latest pre-trained ones.

4.1 Multi-instance Multi-label Evaluation

Unlike the supervised RE tasks which usually evaluate models at the sentence level, DS-RE evaluates *how well models can extract relational facts* from the corpus, i.e. measuring the precision and recall of extracted relational facts (a *fact* is an entity pair and a relation between them). It is named as **multi-instance multi-label**, since each entity pair might be mentioned in multiple sentences, and one entity pair can have more than one relation. Under the framework, models are required to predict the potential relations for each entity pair—according to all sentences mentioning the pair—during the evaluation, as shown in Figure 2. Sentences correlated with the same entity pair are also named as a *bag*, and thus we interchangeably refer to multi-instance multi-label framework as **bag-level** framework.

The most popular way to compare DS-RE models is to plot precision-recall (P-R) curve and calculate the area-under-the-curve (AUC). We also report micro F1 and macro F1 in our experiments. Since the numbers of instances for different relations are extremely imbalanced, macro F1 can better demonstrate model performance while avoiding the bias brought by those major relations.

4.2 Model Details

Sentence-level Training Models are trained in a sentence-level fashion (as in supervised RE), but used as a bag-level model during evaluation. As shown in Figure 2, a typical bag-level model takes

Model	AUC	Micro	Macro
PCNN+bag+AVG (full)	28.4	35.8	13.3
PCNN+bag+ATT (full)	32.2	39.1	9.5
PCNN+bag+ATT (sample)	31.8	39.6	11.8

Table 3: Comparison between full and sampled bag training with NYT10 held-out test. For PCNN+bag+ATT, the sampled bag performs similarly to the full bag, and it still outperforms PCNN+bag+AVG (full). "Micro" and "Macro" represent micro and macro F1 (%).

an *aggregator* to fuse embeddings of all sentences in the bag, and then feeds the bag-level representation to the classifier. Here we take two aggregation strategies: average (**AVG**), which averages the representations of all the sentences in the bag; and at-least-one (**ONE**) (Zeng et al., 2015), which first predicts relation scores for each sentence in the bag, and then takes the highest score for each relation.

Bag-level Training Directly deploying sentence-level training for DS-RE suffers from the wrong labeling problem: one sentence mentioning two entities may not express its auto-labeled relation. To alleviate this problem, models can also take bag-level framework in the training, based on the *expressed-at-least-one* assumption (Riedel et al., 2010): at least one sentence in the bag expresses the auto-labeled relation. Besides the AVG and ONE[2] strategies mentioned above that can be directly deployed for bag-level training, Lin et al. (2016) also propose to use sentence-level attention (**ATT**) for aggregation: It produces bag-level representation as a weighted average over embeddings of sentences, and determines weights by attention scores between sentences and relations.

For our experiments, we take CNN (Liu et al., 2013), PCNN (Zeng et al., 2015) and BERT (Devlin et al., 2019) as options of sentence encoders, which are all common choices for neural RE models. We evaluate combinations of different sentence encoders, training policies, and aggregation strategies, e.g., bag-level trained PCNN with ATT aggregator (PCNN+bag+ATT) or sentence-level trained BERT with ONE aggregator (BERT+sent+ONE). Besides, we evaluate several representative DS-RE models from literature, namely RL-DSRE (Qin et al., 2018), which takes deep reinforcement learning for denoising training instances, BGWA (Jat et al., 2017), which takes both word-level and sentence-level attention, and RESIDE (Vashishth et al., 2018), which introduces side information like relation aliases to put soft constraints on prediction.

For BERT-based sentence encoder, there are some practical challenges when adopting bag-level training: in the worst cases, one bag can contain thousands of sentences, which are beyond the capacity of most computing devices due to the large size of pre-trained models. To address this issue, we take a random sampling strategy during training: for each bag, we randomly sample b sentences, instead of taking all of them. For evaluation, we use the same routine as other non-pre-trained encoders, taking all of the sentences into account (because back propagation is not needed here so the bag can be split into several batches). Since this is different from the original bag-level training, we carry out a pilot experiment to examine the effect of the sampled training. From Table 3, we can see that our sampling strategy does not significantly hurt the performance of the bag-level training.

We also add another variant, BERT-M, in our evaluation. We observe from the top predictions of BERT models (Figure 3) that BERT tends to make false-positive errors for entity pairs that express a relation in the KG but do not have any sentence truly expressing the relation in the data, probably due to that model learns shallow cues solely from entities. Thus, following Peng et al. (2020), we mask entity mentions during training and inference to avoid learning biased heuristics from entities.

5 Experiment

5.1 Implementation Details

We use the OpenNRE toolkit (Han et al., 2019) for most of our experiments, including both sentence-level and bag-level training. For CNN and PCNN, we follow the hyper-parameters of Han et al. (2019). For BERT, we use pre-trained checkpoint `bert-base-uncased` for initialization, take a batch size of 64, a bag size of 4 and a learning rate of 2×10^{-5},[3] and train the model for 3 epochs. For RL-DSRE, RESIDE and BGWA, we directly use their original implementation.

5.2 Evaluation Settings

We take three different settings in our experiments:

---

[2] During training, since we have DS label r for the bag, we directly take the sentence embedding that has the highest score for r as the bag-level representation.

[3] This is determined by a grid search over batch sizes in $\{16, 32, 64\}$ and learning rates in $\{1e\text{-}5, 2e\text{-}5, 5e\text{-}5\}$.

Model	Bag	Strategy	Held-out			Bag-level Manual			Sentence-level Manual		
			AUC	Micro	Macro	AUC	Micro	Macro	AUC	Micro	Macro
CNN	-	AVG	20.0	30.3	6.5	48.7	50.4	21.3	52.0	53.3	22.1
	-	ONE	21.2	31.8	7.2	50.5	51.6	19.8	52.0	53.3	22.1
PCNN	-	AVG	20.4	30.9	9.0	49.4	51.6	22.6	52.2	54.3	23.2
	-	ONE	21.4	31.9	7.8	51.1	52.6	23.8	52.2	54.3	23.2
	✓	AVG	28.4	35.8	13.3	52.9	53.6	23.5	56.0	55.9	22.9
	✓	ONE	28.4	36.0	8.0	53.4	54.8	24.5	55.5	56.7	22.2
	✓	ATT	32.2	39.1	9.5	56.8	56.5	25.5	57.1	56.1	23.6
RL-DSRE	✓	-	32.6	39.5	13.4	55.1	55.9	26.4	55.6	56.1	23.9
BGWA	✓	-	31.0	37.4	11.6	47.8	54.0	14.1	42.2	48.9	7.2
RESIDE	✓	-	33.4	40.5	16.9	35.8	43.3	10.2	43.2	47.9	19.8
BERT	-	AVG	50.5	51.2	21.6	60.3	62.4	35.3	63.2	64.3	34.1
	-	ONE	50.5	51.6	21.2	61.3	62.9	36.1	63.2	64.3	34.1
	✓	AVG	43.0	47.4	22.4	56.7	60.4	35.7	60.4	63.9	34.6
	✓	ONE	38.5	46.1	10.9	58.1	61.9	33.9	61.5	65.1	32.1
	✓	ATT	27.8	37.4	13.4	51.2	54.1	25.8	54.2	57.2	26.4

Table 4: Results (%) on NYT10, including the held-out evaluation, bag-level manual evaluation, and sentence-level manual evaluation. The "bag" column indicates whether the model uses bag-level training, and the "strategy" column shows the bag aggregation policy. We report the AUC, micro F1 (Micro) and macro F1 (Macro) scores.

Held-out evaluation: We take the test data of the original DS datasets for evaluation. The trend of this evaluation should be consistent with the reported results in most DS-RE literature.

Bag-level manual evaluation: We take our human-labeled test data for bag-level evaluation. Since annotated data are at the sentence-level, we construct bag-level annotations in the following way: For each bag, if one sentence in the bag has a human-labeled relation, this bag is labeled with this relation; if no sentence in the bag is annotated with any relation, this bag is labeled as N/A.

Sentence-level manual evaluation: As we wonder how well bag-level-trained models can handle sentence-level predictions, our human-labeled test set is also used for a sentence-level evaluation.

We report AUC, micro F1 and macro F1 for all above evaluation settings. We take the best micro F1 on the P-R curves and use the corresponding threshold for calculating macro F1. Considering the difference between NYT10 and Wiki20, we evaluate models on the two datasets respectively.

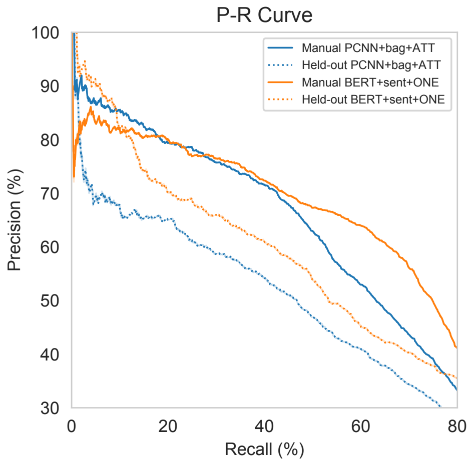

Figure 3: Comparisons of using bag-level manual test (solid lines) and corresponding held-out test (dotted lines). Both absolute and relative scores of models significantly change when taking on human-labeled data.

5.3 The Results on NYT10

Table 4 shows the main results on NYT10. We also plot P-R curves of selected models in Appendix B. Overall, for all three settings, training pre-trained models in a sentence-level style always perform the best, while applying bag-level training strategies can significantly boost the performance when taking other non-pre-trained encoders. Feng et al. (2018) observe that bag-level training is not helpful in the sentence-level evaluation, which contradicts our observation. We suspect that it is because Feng et al. (2018) only manually check a small proportion of test data, leading to a biased result.

More importantly, by comparing the held-out test results to the manual ones, we come to the conclusion that **manual evaluation matters**: auto-labeled and human-labeled test data lead to very different observations. For example, the comparisons between PCNN and RL-DSRE, and BGWA and RESIDE are reversed when taking different evaluations. Also, the performance gaps between different models become much smaller when it

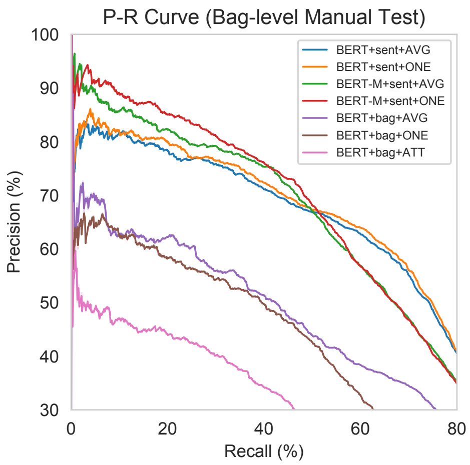

Figure 4: Comparisons of different DS-RE models with BERT on NYT10 bag-level manual test set.

comes to the manual test. Since our manual test set is smaller than the original held-out one (because we did not annotate all N/A sentences), to make a clearer comparison, we evaluate two selected models on the bag-level manual test set and on the corresponding instances in the held-out test set, respectively, and we plot the P-R curves in Figure 3. It shows that not only the absolute values of the two measurements differ a lot, but it also affects the relative performances between the models. For instance, BERT+sent+ONE shows a considerable advantage over PCNN+bag+ATT at the top predictions on the held-out test set, but it is completely the opposite case at the manual test, where BERT+sent+ONE is even significantly worse than PCNN+bag+ATT. It clearly suggests that using the held-out test set cannot well demonstrate the real pros and cons of the models.

Compared to others, BGWA and RESIDE suffer an extreme change in performance between the held-out and manual evaluations, and we suspect that it is due to the fact that they use entity types as extra information, which leads to overfitting biased heuristics of entities. This further emphasizes the need of using manually-labeled test data in DS-RE.

After checking the manual results, we further identify some interesting observations that have not been clearly demonstrated with the DS evaluation:

Pre-trained Models First of all, BERT-based models have achieved supreme performance across all three metrics. To thoroughly examine BERT and its variants in the DS-RE scenario, we further plot their P-R curves with the bag-level manual test in Figure 4. It is surprising to see that all bag-level training strategies, especially the ATT strategy which brings significant improvements for PCNN-based models, do not help or even degenerate the performance with pre-trained ones. This observation is also consistent with that in Amin et al. (2020), though they only compare BERT+bag+AVG and BERT+bag+ATT. We hypothesize the reasons are that solely using pre-trained models already makes a strong baseline, since they exploit more parameters and they have gained pre-encoded knowledge from pre-training (Petroni et al., 2019), all of which make them easier to directly capture relational patterns from noisy data; and bag-level training, which essentially increases the batch size, may raise the optimization difficulty for these large models.

Another unexpected observation is that, though the P-R curve of BERT is far above other models in the held-out test, we identify a significant drop of that in the manual test, as shown in Figure 3 and Appendix B. By manually checking those errors, we find that most of them are models predicting facts that exist in the KG but are not supported by the text (i.e., false-positive). For example, *Arthur Schnitzler* was indeed born in *Vienna*, but it is wrong for the model to infer the relation *place of birth* from sentence "*Authur Schnitzler wrote a story set in Vienna.*" We assume that it is not only because of the prior knowledge of pre-trained models, but is also due to that BERT can better learn heuristics from entity themselves, as shown in the study of Peng et al. (2020) with supervised RE. Considering the data of DS-RE are noisy and in many cases the text does not support the labeled facts, this overfitting-to-heuristic phenomenon can only be more severe.

To verify the assumption and try out a simple solution to alleviate the problem, we take a BERT-M variant (as described in §4.2) and show its results in Figure 4. We can see that the P-R curves of BERT-M are above those of BERT at the beginning, demonstrating that BERT-M models have higher precisions at those top predictions. Later on, since BERT can extract more facts than BERT-M by fully utilizing information from entity names, BERT-M reduces below BERT. From these results, we highlight that **how to handle the false-positives and denoise DS-RE data for pre-trained models** still remains an open and challenging problem.

Imbalanced Classes Previous works of DS-RE usually take AUC, micro F1, or P-R curves to measure the abilities of models, which show the over-

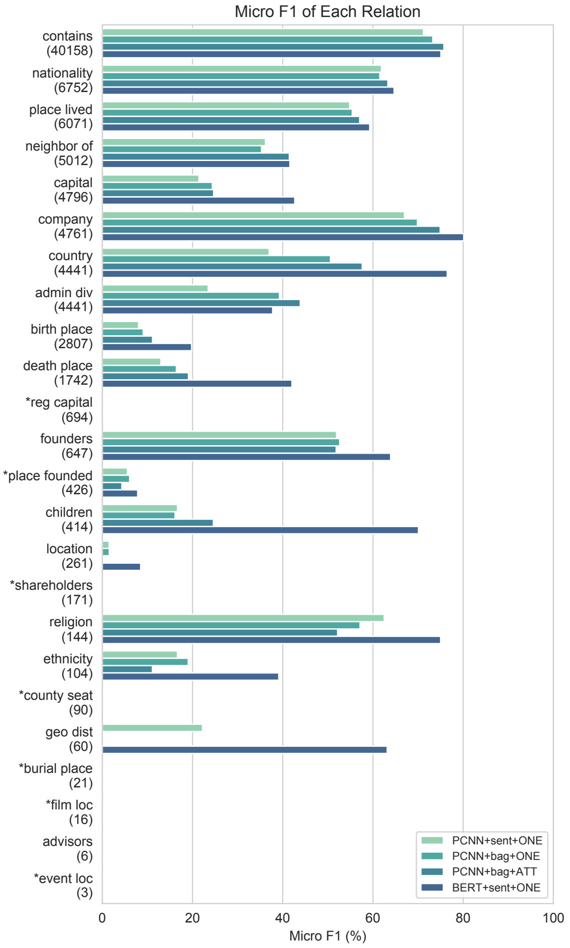

Figure 5: Micro F1 (%) of different relations on the bag-level manual test of NYT10. * represents the relation has less than 20 sentences in the test sets. Numbers in brackets are numbers of training sentences.

all performance trend averaged on relational facts. However, the distribution of training instances across relations is extremely uneven. For example, in NYT10 , almost half of the positive instances are `/location/location/contains`. On the contrary, half of the relations have fewer than $1,000$ sentences. In this case, macro F1 can better show the averaged performance across different relations, without being biased by the majority class. Table 4 demonstrates that even though in most times conclusions of different metrics are consistent, there are cases when models improve micro F1 but degenerate macro F1.

To further study how models perform on each relation, we plot several representative models and their micro F1 scores for each relation in Figure 5. We can see that: (1) The top-4 relations, which account for 80% test instances, do not vary much in performance with different models, while the difference of performance takes place mostly outside the top-4; (2) Some relations even have zero F1 scores, mostly because they have very few training or test sentences. These results further underscore the importance to look into per-relation scores for DS-RE, and we advocate that later works should **include macro F1** for more comprehensive comparisons.

Model	Bag	Strategy	AUC	Micro	Macro
PCNN	-	AVG	74.1	69.1	67.1
	-	ONE	74.0	69.1	66.9
	✓	AVG	78.1	71.8	69.5
	✓	ONE	76.6	70.3	67.7
	✓	ATT	77.5	71.2	68.6
BERT	-	AVG	90.0	83.5	82.9
	-	ONE	89.8	83.3	82.6
	✓	AVG	89.9	82.7	82.0
	✓	ONE	88.9	81.6	81.1
	✓	ATT	70.9	66.8	64.3

Table 5: Results (%) on Wiki20 of representative models. "Bag" indicates bag-level training and "Micro" and "Macro" represent micro and macro F1 respectively.

5.4 The Results on Wiki20

We choose several representative models and further evaluate them on Wiki20, as shown in Table 5. The main observation of results on Wiki20 is consistent with that of NYT10—sentence-level pre-trained models perform the best, and using bag-level training helps with non-pre-trained ones—though the overall performance is much higher. Another difference is that, in Wiki20, AVG performs better than ONE and ATT. We think that it is due to the inherent difference in how the two datasets are constructed, especially the difference in the aspect of determining N/A sentences. Compared to NYT10, part of the N/A instances in Wiki20, instead of indicating no relation between the entities, may correspond to a specific relation that is outside of the dataset ontology. It suggests that when dealing with N/A instances, **considering their latent semantics, rather than simply treating them as one abstract class**, may further benefit RE models.

6 Conclusion

In this paper, we study the problem of test protocols in DS-RE and build large manually-annotated test sets for two DS-RE datasets, to enable a more accurate and efficient evaluation. We not only demonstrate that our manual test sets show different observations from previous held-out ones, but also capture some interesting reflections by using the manual test, e.g., pre-trained models suffer false-positives more and bag-level training strategies generally do not help with pre-trained models.

We hope that our manual test sets can mark a new starting line for DS-RE, while these observations can motivate novel research directions towards better DS-RE models, e.g., studying denoising methods for pre-trained models or processing N/A relations in a more fine-grained way.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2020AAA0106501) and Beijing Academy of Artificial Intelligence (BAAI). This work is also supported by the Pattern Recognition Center, WeChat AI, Tencent Inc.

Ethical Considerations

Our work mainly focuses on two parts: the construction of two manually-labeled test sets and the analyses of models based on the manual test. Regarding the annotation, we first approximate the workload by annotating a few examples on our own, and then determine the wages for annotators according to local standards. The two datasets are based on NYT and Wikipedia, and we did not identify any unethical content during annotation.

Concerning the analyses, we find that models tend to utilize some shallow clues for classification, such as learning heuristics from entities. This behavior can potentially create biased extraction results based on the distributions of entities in the training set and is worth further investigating.

References

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. In *Proceedings of ACL*, pages 1388–1398.

Saadullah Amin, Katherine Ann Dunfield, Anna Vechkaeva, and Guenter Neumann. 2020. A data-driven approach for noise reduction in distantly supervised biomedical relation extraction. In *Proceedings of SIGBioMed*, pages 187–194.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of ACL*, pages 2895–2905.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Mary Elaine Califf and Raymond J. Mooney. 1997. Relational learning of pattern-match rules for information extraction. In *Proceedings of CoNLL*, pages 9–15.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of ACL*, pages 1870–1879.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*, pages 5779–5786.

Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Jie Zhou, and Maosong Sun. 2020. More data, more relations, more context and more openness: A review and outlook for relation extraction. In *Proceedings of AACL-IJCNLP*, pages 745–758.

Xu Han, Tianyu Gao, Yuan Yao, Deming Ye, Zhiyuan Liu, and Maosong Sun. 2019. OpenNRE: An open and extensible toolkit for neural relation extraction. In *Proceedings of EMNLP-IJCNLP*, pages 169–174.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018a. Denoising distant supervision for relation extraction via instance-level adversarial training. *arXiv preprint arXiv:1805.10959*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018b. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*, pages 4832–4839.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018c. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of EMNLP*, pages 2236–2245.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, pages 541–550.

Linmei Hu, Luhao Zhang, Chuan Shi, Liqiang Nie, Weili Guan, and Cheng Yang. 2019. Improving distantly-supervised relation extraction with joint label embedding. In *Proceedings of EMNLP-IJCNLP*, pages 3812–3820.

Scott B Huffman. 1995. Learning information extraction patterns from examples. In *Proceedings of IJCAI*, pages 246–260.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2017. Improving distantly supervised relation extraction using word and entity based attention. In *Proceedings of the 6th Workshop on AKBC*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of AAAI*, pages 3060–3066.

Tingsong Jiang, Jing Liu, Chin-Yew Lin, and Zhifang Sui. 2018. Revisiting distant supervision for relation extraction. In *Proceedings of LREC*, pages 3580–3585.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL*, pages 178–181.

Pengshuai Li, Xinsong Zhang, Weijia Jia, and Wei Zhao. 2020. Active testing: An unbiased evaluation method for distantly supervised relation extraction. In *Findings of EMNLP 2020*, pages 204–211.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of ACL*, pages 34–43.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, pages 2124–2133.

Chunyang Liu, Wenbo Sun, Wenhan Chao, and Wanxiang Che. 2013. Convolution neural network for relation extraction. In *Proceedings of ICDM*, pages 231–242.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011.

Hao Peng, Tianyu Gao, Xu Han, Yankai Lin, Peng Li, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2020. Learning from Context or Names? An Empirical Study on Neural Relation Extraction. In *Proceedings of EMNLP*, pages 3661–3672.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL*, 5:101–115.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*, pages 2463–2473.

Pengda Qin, XU Weiran, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of ACL*, pages 2137–2147.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of EACL*, pages 1171–1182.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL*, pages 74–84.

Dan Roth and Wen-tau Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *Proceedings of COLING*.

Evan Sandhaus. 2008. The new york times annotated corpus ldc2008t19. In *Philadelphia: Linguistic Data Consortium*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, pages 455–465.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of EMNLP*, pages 1257–1266.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of NAACL*, pages 886–896.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Proceedings of CACM*, pages 78–85.

Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Adversarial multi-lingual neural relation extraction. In *Proceedings of COLING*, pages 1156–1166.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*, pages 1778–1783.

Ya Xiao, Chengxiang Tan, Zhijie Fan, Qian Xu, and Wenye Zhu. 2020. Joint entity and relation extraction with a hybrid transformer and reinforcement learning based model. In *Proceedings of AAAI*, pages 9314–9321.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019a. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *Proceedings of NAACL-HLT*, pages 3016–3025.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of EMNLP*, pages 35–45.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019b. ERNIE: Enhanced language representation with informative entities. In *Proceedings of ACL*, pages 1441–1451.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL*, pages 427–434.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph neural networks with generated parameters for relation extraction. In *Proceedings of ACL*, pages 1331–1339.

Tong Zhu, Haitao Wang, Junjie Yu, Xiabing Zhou, Wenliang Chen, Wei Zhang, and Min Zhang. 2020. Towards accurate and consistent evaluation: A dataset for distantly-supervised relation extraction. In *Proceedings of COLING*, pages 6436–6447.

A Relation Ontology Changes of NYT10

- /location/country/administrative_divisions
- /location/administrative_division/country
- /location/country/capital
- (merge) /location/region/capital
    - /location/fr_region/capital
    - /location/cn_province/capital
    - /location/in_state/administrative_capital
    - /location/in_state/legislative_capital
    - /location/in_state/judicial_capital
    - /location/it_region/capital
    - /location/br_state/capital
    - /location/mx_state/capital
    - /location/province/capital
    - /location/us_state/capital
    - /location/jp_prefecture/capital
    - /location/de_state/capital
- /location/us_county/county_seat
- /location/neighborhood/neighborhood_of
- /location/location/contains
- (merge) /business/location
    - /business/company/locations
    - /sports/sports_team/location
    - /broadcast/producer/location
- /business/company/founders
- /business/company/place_founded
- /business/company/major_shareholders
- /business/company/advisors
- /business/person/company
- /people/person/place_of_birth
- /people/person/religion
- /people/person/nationality
- /people/person/place_lived
- /people/person/ethnicity
- /people/person/children
- /people/deceased_person/place_of_death
- /people/deceased_person/place_of_burial
- /people/ethnicity/geographic_distribution
- /time/event/locations
- /film/film/featured_film_locations
- N/A
- (delete) location/country/languages_spoken
- (delete) base/locations/countries/states_provinces_within
- (delete) business/shopping_center_owner/shopping_centers_owned
- (delete) business/shopping_center/owner
- (delete) business/business_location/parent_company
- (delete) business/company_advisor/companies_advised
- (delete) people/profession/people_with_this_profession
- (delete) people/person/profession
- (delete) people/place_of_interment/interred_here
- (delete) people/ethnicity/included_in_group
- (delete) people/family/members
- (delete) people/family/country
- (delete) broadcast/content/location
- (delete) film/film_festival/location
- (delete) film/film_location/featured_in_films

Table A.1: NYT10 relation ontology. All those relations are from FreeBase. "(merge)" means a merge relation in our version of NYT10 and it is followed by relations merged from the original dataset. "(delete)" indicates that this relation is discarded in our version because there are no instances in the training or the test sets.

B P-R Curves for NYT10

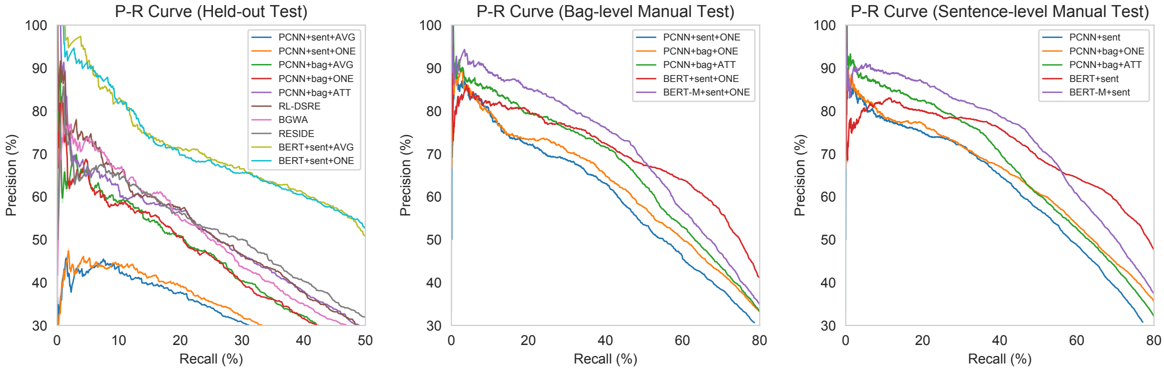

Figure B.1: P-R curves of representative models in held-out test, bag-level manual test and sentence-level manual test of NYT10. Note that the scales of X-axis are not the same in the three figures.

GCRC: A New MRC Dataset from Gaokao Chinese for Explainable Evaluation

Hongye Tan[1*], **Xiaoyue Wang**[1*], **Yu Ji**[1], **Ru Li**[†], **Xiaoli Li**[2], **Zhiwei Hu**[1],
Yunxiao Zhao[1] **and Xiaoqi Han**[1]
1.School of Computer and Information Technology, Shanxi University, Taiyuan, China
2.Institute for Infocomm Research, A*Star, Singapore
`{wangxy0808, jiyu0515, yunxiaomr, xiaoqisev}@163.com`
`{liru,tanhongye}@sxu.edu.cn`
`xlli@ntu.edu.sg, zhiweihu@whu.edu.cn`

Abstract

Recently, driven by numerous publicly available machine reading comprehension (MRC) datasets, MRC systems have made some progress. These datasets, however, have two major limitations: 1) the defined tasks are relatively simple, and 2) they do not provide explainable evaluation which is critical to objectively and comprehensively review the reasoning capabilities of current MRC systems. In this paper, we propose GCRC, a new dataset with challenging and high-quality multi-choice questions, collected from Gaokao Chinese (Chinese subject from the National College Entrance Examination of China). We have manually labelled three types of evidence to evaluate MRC systems' reasoning process: 1) sentence-level relevant supporting facts in an article required for answering a given question, 2) error reason of a distractor (i.e., an incorrect option) for explaining why a distractor should be eliminated, which is an important reasoning step for multi-choice questions, and 3) types of reasoning skills required for answering questions. Extensive experiments show that our proposed dataset is more challenging and very useful for identifying the limitations of existing MRC systems in an explainable way, facilitating researchers to develop novel machine learning and reasoning approaches to tackle this challenging research problem.[1]

1 Introduction

Machine Reading Comprehension (MRC) is a critical task in many real-world applications, which requires machines to understand a text passage, and answer relevant questions. It evaluates machines' understanding and reasoning capabilities on the underlying natural language text. Numerous MRC datasets have been proposed and facilitate the progress of MRC systems, which have achieved near-human performance on some datasets. However, this does not indicate the MRC systems have owned human-like language understanding and reasoning capabilities.

One reason is that questions in current data are not challenging enough, leading to most of them get "solved" very soon. (Sugawara et al., 2018) demonstrate that in many MRC datasets, a considerable number of easy questions can be answered based on the first few tokens of the questions or word matching, without complex reasoning capabilities. The other reason is that most datasets only provide a black-box and overall evaluation on the accuracy of predicted answers, which does not provide **explainable evaluation** on a system's internal reasoning capabilities. In other words, it is unable to explain whether a system gets a correct answer via the right reasoning process, and not enough to identify the specific limitations of a system.

Recently, in order to address the problems mentioned, some datasets, focusing on reasonings and providing additional information to evaluate the internal reasoning steps of a system, have been proposed. For example, MultiRC (Khashabi et al., 2018) and HotpotQA (Yang et al., 2018) include the questions requiring multi-sentence reasoning

---

[*] These authors contributed equally.

[†] Corresponding author

[1] Resources will be available through `https://github.com/SXUNLP/GCRC`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1319–1330
August 1–6, 2021.

	In Chinese	In English
Article	随着全球人口的不断增长和科学技术的飞速发展，人类在创造文明的同时也缔造了一个深受人类影响的全球生态系统。长期以来对生物资源及土地的过度利用，导致了动植物栖息地丧失、环境污染等一系列问题的出现，生态环境及生物系统遭受了严重破坏。据专家估计，由于人类活动和气候变化，地球上的生物种类目前正在以相当于正常水平1000倍的速度消失，全球已有约3.4万种植物和5200多种动物濒临灭绝，物种分布发生了大范围的变化，这些形成了全球性的生物多样性危机。... 生物多样性为人类发展带来了巨大财富，目前它却面临着来自城市化等方面的威胁。城市化对生物多样性的影响成为生态学研究者关注的焦点问题。	With the continuous growth of global population and rapid development of science and technology, human beings has created a global ecosystem which is deeply influenced by mankind while creating civilization. The long-term over-utilization of biological resources and land has led to the emergence of problems such as the loss of wildlife habitats and environmental pollution, and meanwhile, the ecological environment and biological systems have been severely damaged. According to experts, the species on the earth are disappearing at a rate of 1,000 times the normal level due to human activities and climate change. Currently, about 34,000 species of plants and more than 5,200 species of animals are on the verge of extinction, and the species distribution has been changed significantly, which has resulted in a global biodiversity crisis. ... Biodiversity has brought huge wealth to human development, but it is facing the threat from urbanization and other aspects. The impact of urbanization on biodiversity has become the focus of ecological researchers.
Question stem	根据材料一，下列理解和分析，符合文意的一项是?	Which of the following statements agrees with the information given in the article?
Options	A：深受人类影响的全球生态系统利于缓解生物多样性危机。 （所需推理能力：归纳推理） （错误原因：主谓宾关系错误） *B：第一段通过列举数据来凸显生物多样性危机的严重程度。 （所需推理能力：鉴赏分析） C：生态学者关注的焦点是生物多样性危机给人类带来哪些损失。 （所需推理能力：细节推理） （错误原因：细节错误） D：这则材料反映了对生物多样性危机的担忧并提出了应对策略。 （所需推理能力：归纳推理） （错误原因：无中生有）	A: The global ecosystem, which is deeply influenced by human beings, helps alleviate the biodiversity crisis. (Reasoning skill: Inductive reasoning) (Error reason: Wrong subject-predicate-object relationship) *B: *The first paragraph* highlights the severity of the biodiversity crisis by giving some statistics. (Reasoning skill: Appreciative analysis) C: The focus of ecological researchers is what loss the biodiversity crisis has brought to human beings. (Reasoning skill: Detail understanding) (Error reason: Wrong details) D: *This article* reflects the concerns about biodiversity crisis and proposes countermeasures. (Reasoning skill: Inductive reasoning） (Error reason: Irrelevant to the article.）

Figure 1: An annotated instance in GCRC, which is from the real Gaokao Chinese in 2019 (* indicates the correct option). The sentences marked in yellow/green are respectively the SFs for Option A and C. As mentioned in Option B, the SFs of Option B are the first paragraph. Similarly, the SFs of Option D are the whole article. The contents marked in blue are the required reasoning skills for the options and the ERs of distractors.

or multi-hop reasoning. Moreover, MultiRC and HotpotQA both provide *sentence-level supporting facts (SFs)*, which can be used as a kind of internal explanation for the answers, although locating SFs is just the first step for question answering (QA), as many questions need to integrate the SF information with reasoning. R4C (Inoue et al., 2020) and 2WikiMultiHopQA (Ho et al., 2020) introduce *a chain of facts* (reflecting entity relationships) as the derivation steps to evaluate the internal reasoning step of systems.

However, these kinds of reasonings are quite limited and not enough to answer those complex and comprehensive questions. For example, Figure1 shows a multi-choice question, where the option D "***This article reflects** the concerns about biodiversity crisis and proposes countermeasures*" is a summarization of the whole given article, and its judgement cannot be made by simply reasoning over some entity relationships. Instead, the reasoning over the full information across the article will be needed.

To address the above real challenges, we propose GCRC, a new challenging dataset with 8,719 multi-choice questions, collected from reading comprehension (RC) tasks of Gaokao Chinese (short for Chinese subject from the National College Entrance Examination of China). GCRC is of high quality and high difficulty level, because Gaokao Chinese examinations are designed by educational experts, with high-level comprehensive questions and complicated articles, aiming to test the language comprehension of adult examinees.

In order to provide **explainable evaluation**, instances (e.g. Figure 1) in the dev. and test sets (including 1725 questions and 6900 options, as presented in Table 2 and Section 4) of GCRC are annotated with three kinds of information: (1) ***sentence-***

level supporting facts **(SFs)**, serves as the basic evaluation for a system's internal reasoning; and (2) ***error reasons*** **(ERs)** ***of a distractor*** (i.e. an incorrect option) for explaining why a distractor should be eliminated. In Gaokao Chinese, distractors are often designed in a very confusing way and look like a correct option, although they are incorrect. Identifying the semantic difference between a distractor and the given article, and knowing exactly why a distractor is wrong could help systems or examinees to choose the correct answers. Therefore, we spend considerable effort to thoroughly understand articles and manually label seven types of ERs of a distractor as a form of internal reasoning step for multi-choice questions; and (3) ***types of reasoning skills*** required for answering questions in Gaokao Chinese. We introduce eight typical reasoning skills in GCRC, which enable us to evaluate whether a system has corresponding reasoning capability at individual question level.

Our main contributions can be summarized as:

- We construct a new challenging dataset GCRC that is collected from Gaokao Chinese, consisting of 8,719 multiple-choice questions, which will be released on Github in future.
- Three kinds of critical information are manually annotated for explainable evaluation. To the best of our knowledge, GCRC is the first Chinese MRC dataset to provide most comprehensive, challenging, high-quality articles and questions for more deep explainable evaluation on different MRC system performance. In particular, error reasons of a distractor are the first introduced in this area as an important reasoning step for complex multi-choice questions.
- Extensive experiments show that GCRC is not only a more challenging benchmark to facilitate researchers to develop novel models, but also help us identify the limitations of existing MRC systems in an explainable way.

2 Related Work

2.1 Datasets from standard tests

There exist some datasets, collected from standard exams/tests, including English datasets RACE (Lai et al., 2017), DREAM (Sun et al., 2019), ARC (Clark et al., 2018), ReClor (Yu et al., 2020), etc., and Chinese datasets C3 (Sun et al., 2020), MCQA (Guo et al., 2017), GeoSQA (Huang et al., 2019) and MedQA (Zhang et al., 2018) etc.

Some of these datasets are of specific domains. For example, ARC is of the science domain and the questions are from the American science exams from 3rd to 9th grade. ReClor targets logical reasoning and the questions are collected from the Law School Admission Council. MCQA and GeoSQA are extracted from Gaokao History and Gaokao Geography respectively. Other datasets cover generic topics. For example, RACE and DREAM are both collected from the English exams for Chinese students, while C3 is collected from the Chinese exams for Chinese-as-a-second-language learners.

GCRC is more similar to C3, RACE and DREAM, as their questions are all generic ones. However, their question difficulty level is different, because questions in GCRC target for adult native speakers, while questions in other datasets target for second-language learners. As such, the GCRC is much more challenging than other datasets. Additionally, GCRC provides three kinds of rich information for more deep explainable evaluation.

2.2 Datasets with explanations

Some datasets provide explanation information. (Wiegreffe and Marasovic, 2021) identified three types of explanations: highlights, free-text and structured explanations. Highlights are subsets of the input elements (words, phrases, snippets or full sentences) to explain the prediction. Free-text explanations are textual or natural language explanations containing the information beyond the given input. Structured explanations have various forms for specific datasets. One of the most common form is a chain of facts as the derivation of multi-hop reasoning for an answer.

(Inoue et al., 2020) classified explanations into two types: justification explanation (collections of SFs for a decision) and introspective explanation (a derivation for a decision), which are respectively corresponding to highlights and structured explanations.

For the MRC task, a few datasets are with explanation information. For example, MultiRC (Khashabi et al., 2018) and HotpotQA (Yang et al., 2018) provides sentence-level SFs, belonging to justification explanations, to evaluate a system's ability to identify SFs from articles that related to a question/option. R4C (Inoue et al., 2020) and 2WikiMultiHopQA (Ho et al., 2020) provide both justification and introspective explanations. In both datasets, introspective explanations are a set of factual entity relationships. Their difference is that the explanation of R4C is of semi-structured form, while the explanation of 2WikiMultiHopQA is of structured data form.

The dataset C3 provides the types of required prior knowledge for 600 questions, and its goal is to study how to leverage various knowledge to improve the system's ability for text comprehension.

Inspired by these datasets, we provide more rich explanations in GCRC. Different from the existing work, besides SFs, we provide two additional information. Specifically, we propose innovative ERs of a distractor as a reasoning step for multi-choice questions. In addition, we annotate the required reasoning skills for each instance, which enable us to clearly identify the limitations of existing MRC systems. Note that prior knowledge annotated in C3 is different from reasoning skills in GCRC, as prior knowledge in C3 mainly include linguistic, domain specific, and general world knowledge, while reasoning skills in GCRC focus on the abilities of making an inference. Generally, reasoning needs to integrate prior knowledge and the information in the given text. As far as we know, *GCRC is the first Chinese MRC dataset with rich explainable information* for evaluation purpose.

3 Dataset Overview

Questions in GCRC are of multi-choice style. Specifically, given an article D, a question Q and its options O, the evaluation based on GCRC will measure a system from the following aspects:

- *QA accuracy*. It is a common evaluation metric, also provided by the other QA datasets.
- *The performance of locating the SFs in D*, which is a basic evaluation metric to assess whether MRC system can collect all necessary sentences from articles before reasoning.
- *The performance of identifying ERs of a distractor*, which evaluates whether a system is able to exclude incorrect options by deep reasoning for multi-choice questions.
- *The performance of different reasoning skills required by questions*. This evaluation shows the limitations of reasoning skills for a MRC system.

Next, we clearly define the ERs of a distractor and the reasoning skills required for choosing a correct option in Section 3.1 and Section 3.2 respectively. The ERs of a distractor are concrete and focus on the forms of the errors, while reasoning skills are more abstract, referring to the abilities of making an inference.

3.1 Error reasons of a distractor

As mentioned in Section 1, knowing exactly why a distractor is wrong will help MRC systems or an examinee select correct answers. Therefore, we introduce error reasons of a distractor as an important internal reasoning step, and present seven typical ERs of a distractor after investigating the instances in GCRC.

Wrong details. The distractor is *lexically similar to the original text, but has a different meaning caused by alterations of some words*, such as modifiers or qualifier Words.

Wrong temporal properties. The distractor describes an events with *wrong temporal properties*. Generally, we consider five temporal properties defined in (Zhou et al., 2019): duration, temporal ordering, typical time, frequency, and stationary.

Wrong subject-predicate-object triple relationship. The distractor has a triple relationship of subject-predicate-object (sub-pred-obj) , but *the relationship conflicts with the ground truth triple in the given article*, caused by substituting one of the components in the triple. For example, in Figure 1, Option A "*The global ecosystem, which is deeply influenced by human beings, helps alleviate the biodiversity crisis.*" has a sub-pred-obj triple "*the global ecosystem—**alleviate**—the biodiversity crisis*. However, from the given article, the correct triple is "*the global ecosystem—**result in** —the biodiversity crisis*", leading to the wrong distractor.

Wrong necessary and sufficient conditions. The distractor *intentionally misinterprets the necessary and sufficient conditions* expressed in the original article. A necessary condition is a condition that must be present for an event to occur, while a sufficient condition is a condition or set of conditions that will produce the event. For example, the statement "*Plants will grow as long as* *with air*" is wrong, as air is the necessary condition for plant growing, but besides air, plants also need water and sunshine to grow.

Wrong causality. The distractor reverses/confuses the causes and effects mentioned in an original article or add non-existent causality. Generally, the cause-effect relationship is explicitly expressed with causal connectives in distractors.

Irrelevant to *the question*. The distractor's contents are indeed mentioned in the given article, but are irrelevant to the current question.

Irrelevant to *the article*. The distractor's contents are not mentioned in the original article. For example, in Figure 1, "*proposes countermeasures*" expressed in Option D "*This article reflects the concerns about biodiversity crisis and proposes countermeasures*." is not mentioned in the given article. Thus, Option D should be excluded.

3.2 Required reasoning skills

In Gaokao Chinese, RC tasks measure an examinee's text understanding and logical reasoning abilities from different perspectives. We investigate the skills required for answering questions in GCRC, and introduce eight typical skills, which are organized into the following three levels according to the amount of information needed and the complexity of reasoning for QA. Some of the reasoning skills (marked with *) are similar to those proposed by Sugawara et al. (2017).

3.2.1 Level 1: Basic information capturing

This level covers the ability to capture relevant detailed information distributed across the article, and combine them to match with options.

Detail understanding. It focuses on distinguishing the semantic differences between the given article and an option. The option, most of the time, preserves the most lexical surface form of the original article, but has some minor differences in details by using different modifiers or qualifying words.

3.2.2 Level 2: Local information integration

This level covers how to identify different types of relationships linked between sentences in an article. In Gaokao Chinese, the relationship's expressions are often implicit in a given article.

Temporal/spatial reasoning*. It aims to understand various temporal or spatial properties of events, entities and states.

Coreference resolution*. It aims to understand the coreference and anaphoric chains by recognizing the expressions referring to the same entity in the given article.

Deductive reasoning. It focus on taking a general rule or key idea described in an article, and applying it to make inferences about a specific example or phenomenon expressed in the option. For example[2], the statement of "*we rely on mobile navigation to travel and lose the ability to identify routes.*" is a specific phenomenon of the statement of "*while we are training artificial intelligence systems, we may also be trained by artificial intelligence systems*".

Mathematical reasoning*. It performs some mathematical operations, such as numerical sorting and comparison, to obtain a correct option.

Cause-effect comprehension*. It aims to understand the causal relationships explicitly or implicitly expressed in a given article.

3.2.3 Level 3: Global information integration

This level involves information integration of multiple sentences or the whole articles to comprehend the main ideas, article organization structures and the authors' emotion and sentiment.

Inductive reasoning. It integrates information from separate words and sentences, and makes inferences about an option, which is often a summarization of several sentences, a paragraph or the whole article.

Appreciative analysis. It aims to understand the article organization method, the authors' writing style and method, attitude, opinion and emotional state. For example, in Figure 1, Option B "*The first paragraph highlights the severity of the biodiversity crisis by giving some statistics*" is the analysis result of authors' writing style and method.

4 Construction and Annotation of GCRC

We have spent tremendous effort to construct the important GCRC dataset. Firstly, we search and download about 10,000 latest (year 2015 to 2020)

---

[2] This example is translated from real Gaokao Chinese in 2018.

multiple-choice questions of real and mock Gaokao Chinese from five websites. Then, for preprocessing, we remove those duplicated questions and questions with figures and tables, and keep those questions with four options (only one of them is correct). Next, we identify and rectify some mistakes, such as typos, in articles or corresponding questions/options. Finally, the total number of questions in GCRC is 8,719. Moreover, we adjust the labels of the correct answers, making them evenly distributed over the four different options, i.e., A, B, C, D.

Next, we will annotate the questions according to the following three steps. We would emphasize that the annotation process is extremely challenging, as it requires human annotators to fully understand both syntactic structure and semantic information of the articles and corresponding questions and options, as well as identify the error reasons of distractors, and reasoning skills required.

- **Step 1: Annotation preparation.** We first prepare an annotation guideline, including task definition and annotated examples. Then, we invite 12 graduate students of our team to participate in the annotation work. To maintain high quality and consistent annotations, our annotators first annotate these questions individually, and subsequently discuss and reach agreements if there is any discrepancy between two annotators. The process further improves the annotation guideline and better trains our annotators.
- **Step 2: Initial annotations.** Firstly, each question is annotated by an annotator independently, where we show an article, question, all candidate options and the label of the answer. Annotators have completed the following three tasks: (1) select the sentences in the article, which are needed for reasoning, as the sentence-level SFs; (2) provide ERs of a distractor from the types discussed in Section 3.1; (3) provide the reasoning skills required for each option based on the skills described in Section 3.2.
- **Step 3: Annotation consistency.** When two annotators disagree with their own annotations, we invite the third annotator to discuss with them, and reach the final annotations. In the rare cases where they cannot agree with each other, we will keep the annotations with at least two supports.

The annotators' consistency is evaluated by the Inter Annotator Agreement IAA) value and the IAA value is 83.8%.

As mentioned before, manual annotations of GCRC is expensive and complicated, because questions in Gaokao Chinese are designed for adult native speakers and thus are challenging. It requires deep language comprehension to solve these questions. As a result, making explainable annotations in GCRC across multiple levels implies *GCRC is a very precious dataset with valuable annotations*. Although the size of the dataset is not too big, it is big enough to be used for providing diagnosis of existing MRC systems. It also provides an ideal testbed for researchers to propose novel transfer learning or few-shot learning methods to solve the tasks.

Statistics. We partition our data into training (80%), development (dev, 10%) and test set (10%), mainly according to the number of questions. We have annotated three types of information for a subset of GCRC. Specifically, we annotate the sentence-level SFs for 8,084 options (of 2,021 questions) sampled from the training set, and 6,900 options (of 1725 questions) in the dev and test sets.

In addition, we annotate ERs of 6,159 distractors in the training, dev and test sets, and reasoning skills for 6,900 options in the dev and test sets. We believe our dataset with relatively big annotation sizes can ensure us to identify the limitations of existing RC systems in an explainable way. Table 1 shows the the details of GCRC data splitting and corresponding annotation size.

Table 2 shows the detailed comparisons for GCRC and other three RC datasets collected from standard exams, including C3 (Sun et al., 2020), RACE (Lai et al., 2017), DREAM (Sun et al., 2019). As shown in Table 2, we observe that GCRC is the longest in terms of the average length of articles, questions and options.

Table 3 shows the distribution of types of reasoning skills based on the dev and test sets. We observe that 48.77% questions need detail understanding, and 33.10% questions require inductive reasoning. In addition, Figure 2 presents the ERs types distribution of distrators based on the dev and test sets, in which 33.2% of distractors are with wrong details and 26.4% of distractors include information irrelevant to the corresponding articles.

Splitting	**Train**	**Dev**	**Test**	**Total**
# of articles	3790	683	620	5093
# of questions	6994	863	862	8719
# of questions/options with SFs	2021/8084	863/3452	862/3448	3746/14984
# of questions/distractors with ERs	2000/3261	863/1428	862/1470	3725/6159
# of questions/options with rea. ski.	-	863/3452	862/3448	1725/6900

Table 1: Statistics of data splitting and annotation size.

	GCRC	C3	GCRC	C3	RACE	DREAM
	(in Chinese Characters)		(in tokens)			
Article len. (avg)	1119.2	116.9	329.6	53.8	352.4	66.1
Question len. (avg)	20.9	12.2	14.2	7.8	11.3	7.4
Option len. (avg)	43.8	5.5	27.1	3.2	6.7	4.2

Table 2: Statistics and comparison among four MRC datasets collected from standard exams.

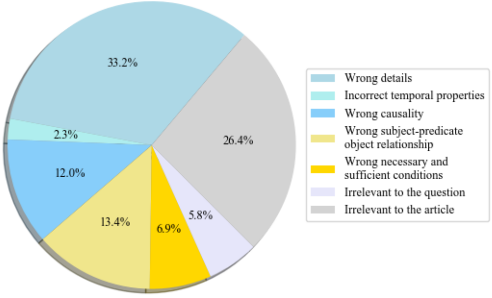

Figure 2: Distirbution(%) of types of ERs of distractors based on the dev and test sets, including 3,725 questions and 6,159 distractors.

	Reason skills	Dev	Test	all
Level 1: Basic information capturing	Detail	47.11	50.44	48.77
Level 2: Local information integration	Temporal/Spatial	0.84	0.73	0.79
	Co-reference	5.42	4.88	5.15
	Deductive	3.53	3.79	3.66
	Mathematical	0.45	0.67	0.56
	Cause-effect	5.67	4.76	5.21
Level 3: Global information integration	Inductive	34.75	31.45	33.10
	Appreciative	2.23	3.30	2.76
Leve 1: Basic information capturing		47.11	50.44	48.77
Level 2: Local information integration		15.91	14.83	15.36
Level 3: Global information integration		36.98	34.75	35.87

Table 3: Distribution (%) of types of required skills based on the annotated examples, totally including 1,725 questions and 6,900 options.

5 Experiments

With our newly constructed GCRC dataset, it is interesting to evaluate the performance of existing models, and better understand GCRC's characteristics comparing with other existing data sets.

5.1 QA accuracy and GCRC difficulty level

We evaluate the QA performance of several popular MRC systems on GCRC, which will reflect the difficulty level of GCRC. Specifically, for comprehensive evaluation, we employ four models, including one *rule-based model* and three recent *state-of-the-art models based on neural networks*.

- **Sliding window** (Richardson et al., 2013). It is a rule-based baseline and chooses the answer with the highest matching score. In particular, it has TFIDF style representation and calculates the lexical similarity between a sentence (via concatenating a question and one of its candidate options) and each span in the given article with a fixed window size.

- **Co-Matching** (Wang et al., 2018). It is a Bi-LSTM-based model and consists of a co-matching component and a hierarchical aggregation component. The model not only matches the article with a question and each candidate option at the word-level, but also captures sentence structures of the article. It has achieved promising results on RACE.

  Note that we have used 300-dimensional word embeddings based on GloVe (Global Vectors for Word Representation).[3]

- **BERT** (Devlin et al., 2019). It is a pre-trained language model, which adopts multiple bidirectional transformer layers and self-attention mechanisms to learn contextual relations between words in a text. BERT has achieved very good performance on numerous NLP tasks, including the RC task defined by SQuAD. We employ Chinese BERT-base and English BERT-base released on the website.[4]

- **XLNet** (Yang et al., 2019). It is a generalized autoregressive pretraining model, which uses a permutation language modeling objective to combine the advantages of autoregressive and autoencoding methods. XLNet outperforms BERT on 20 tasks. For experiments on Chinese datasets, we use XLNet-large and its Chinese version released on the website.[5]

---

[3] The English word embedding:https://nlp.stanford.edu/projects/glove/; and the Chinese word embedding: https://nlp.stanford.edu/projects/glove/

[4] https://github.com/google-research/bert

[5] https://github.com/brightmart/xlnet$_{z}h$

In addition to evaluate the performance of different models on GCRC, we also want to see how these models perform on other datasets, namely C3, RACE, and DREAM. In order to fairly and objectively compare with them, we randomly sample from these datasets, and create three new datasets with the same sizes of the splits (train, dev, test) as GCRC (shown in Table 1). The hyper-parameters of these neural baselines can be seen in Appendix.

Human Performance. We obtain the human performance on 200 questions, which are randomly sampled from the GCRC test set. We invite 20 high school students to answer these questions, where they are provided with the questions, options and articles. The average accuracy of human is 83.18%.

We first use the training sets from four datasets to train four models, which are subsequently used to evaluate their accuracies on respective test data. Dev sets are used to tune the values of the parameters of different models. Table 4 shows the comparison results. We observe that the neural network based models outperform the rule based sliding window model. In addition, pretrained models perform better than the non-pretrained models in most cases. Moreover, it is interesting to observe that all four existing models generally perform worse on GCRC than other three datasets, and the human performance on GCRC is also the lowest among the four datasets. This clearly indicates that GCRC is more challenging. Meanwhile, it can be seen that the performance gap between human and the best system on GCRC is 46.45%, suggesting that there is sufficient room for improvement, which contradicts with some overly optimistic claims that machines have exceeded humans on QA. As there is a significant performance gap between machines and humans, GCRC dataset facilitates researchers to develop novel learning approaches to better understand its questions and bridge the huge gap.

	GCRC	C3	RACE	DREAM
Sliding window	27.70%	36.70%	30.85%	40.08%
Co-Matching	35.73%	45.01%	35.38%	48.91%
BERT	30.74%	64.96%	46.13%	53.36%
XLNet	35.15%	60.90%	50.00%	59.63%
Human	83.18%	96%*	95.5%*	94.5%*

Table 4: Accuracy comparison between four computational models and human on four benchmark datasets (* indicates the performance is based on the annotated test set and copied from the corresponding paper).

In the next few subsections, we will study whether a representative model, i.e. BERT, can perform well in the explainable evaluation related subtasks, namely locating sentence-level SFs, identifying ERs, and the performance of different reasoning skills required by questions.

5.2 Performance of locating sentence-level SFs using BERT

We investigate whether BERT benefits from sentence-level SFs. We conduct an experiment, in which we input the ground-truth SFs, instead of the given article, to BERT for question answering. We observe that the accuracy of QA on the test set of GCRC is 37.47%. Comparing with the result shown in Table 4, the accuracy is increased by 6.73%, indicating that locating SFs is helpful for QA. On the other hand, we can see that the improvement is still far from closing the gap to human performance because the questions are difficult and need further reasoning to solve.

Due to the usefulness of SFs, we train a BERT model to identify whether a sentence belongs to SFs. We regard this task as a sentence pair (i.e., an original sentence in the given article and an option) classification task. We use the GCRC subset (2,021/863/862 questions as shown in Table 1) annotated with SFs for training and testing, where the training, dev and test sets include 200,798/85,024/83,656 sentence-option pairs respectively. The experimental results are shown in Table 5. We can see that performance of locating the SFs is still low (in terms of precision P, recall R and F1 measure), indicating that accurately obtaining the SFs in GCRC is challenging and directly applying BERT will not work well.

	P	R	F1
Dev	78.07%	50.18%	61.10%
Test	70.20%	51.46%	59.39%

Table 5: Results of BERT locating SFs.

5.3 Performance of identifying ERs of a distractor using BERT.

In order to evaluate whether BERT model understands why a distractor is excluded, we modify BERT model and add a new component to perform the identification of distractors' ERs. The component is a multi-label (i.e., 8 labels including 1

ERs	#of options	R	P
Wrong details	484	1.65%	9.09%
Wrong temporal properties	29	13.79%	0.55%
Wrong subject-predicate-object triple	190	2.11%	5.88%
Wrong causality	218	24.77%	7.62%
Wrong necessary and sufficient conditions	106	1.89%	2.22%
Irrelevant to the question	85	17.65%	2.21%
Irrelevant to the article	352	4.83%	17.35%
Correct options	1984	28.18%	56.69%

Table 6: Results of BERT identifying ERs of distractors.

correct type and 7 types of ERs) classifier to predict the probability distribution of the types of ERs, and is jointly optimized with normal QA, and it shares the low-level representations. The classifier's objective is to minimize a cross entropy loss, which is jointly optimized with normal QA, and they share the low-level representations. For this task, the contextual input of BERT is ground-truth SFs, instead of the given article. The results are shown in Table 6. We observe that the performance of identification of ERs is quite low, indicating the significance and value of our dataset, which is to identify the limitations of existing systems on explainability and facilitates researchers to develop novel learning models.

5.4 Performance of different reasoning skills required by questions using BERT

We also investigate BERT's performance on the questions requiring different reasoning skills. We categorize the performance for each type of reasoning skills on the test set of GCRC. Table 7 shows the results. Note that the reasoning skills are annotated for options of questions. We observe BERT obtains the lowest score on the options requiring deductive reasoning. Overall, the system generally performs worse on each type of options, indicating the reasoning power of the system is not strong enough and needs to be significantly improved.

Reasoning	Detail understanding	Temporal/spatial	Coreference	Deductive
#of options	1683	42	179	143
Accuracy	31.97%	61.90%	39.66%	35.66%
	Mathematical	Cause-effect	Inductive	Appreciative
#of options	40	175	1056	130
Accuracy	60.00%	40.00%	33.14%	47.69%

Table 7: Results of BERT on GCRC by reasoning skills required for QA.

From the above, it can be seen that no baseline is realized to output answers, sentence-level SFs and ERs of a distractors together, but it doesn't affect our dataset to diagnose the limitations of existing RC models. For each task, we modify the existing model of BERT, output the corresponding explanations, and report their performance. Thus, the limitations of the model can be clearly identified. In the future, we will realize such a baseline that can do all the tasks together, and we will design a new joint metric for evaluating the whole question-answering process.

6 Conclusions

In this paper, we present a new challenging machine reading comprehension dataset (GCRC), collected from Gaokao Chinese, consisting of 8,719 high-level comprehensive multiple-choice questions. To the best of our knowledge, this is currently the most comprehensive, challenging, and high-quality dataset in MRC domain. In addition, we spend considerable effort to label three types of information, including sentence-level SFs, ERs of a distractor, and reasoning skills required for QA, aiming to comprehensively evaluate systems in an explainable way. Through experiments, we observe GCRC is very challenging data set for existing models, and we hope it can inspire innovative machine learning and reasoning approach to tackle the challenging problem and make MRC as an enabling technology for many real-world applications.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Key Research and Development Program of China (No.2018YFB1005103).

References

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shangmin Guo, Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2017. Which is the effective way for Gaokao: Information retrieval or neural networks? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 111–120, Valencia, Spain. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zixian Huang, Yulin Shen, Xiao Li, Yu'ang Wei, Gong Cheng, Lin Zhou, Xinyu Dai, and Yuzhong Qu. 2019. GeoSQA: A benchmark for scenario-based question answering in the geography domain at high school level. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5866–5871, Hong Kong, China. Association for Computational Linguistics.

Naoya Inoue, Pontus Stenetorp, and Kentaro Inui. 2020. R4C: A benchmark for evaluating RC systems to get the right answer for the right reason. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6740–6750, Online. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4208–4219, Brussels, Belgium. Association for Computational Linguistics.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 806–817, Vancouver, Canada. Association for Computational Linguistics.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging Chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155.

Shuohang Wang, Mo Yu, Jing Jiang, and Shiyu Chang. 2018. A co-matching model for multi-choice reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short*

Papers), pages 746–751, Melbourne, Australia. Association for Computational Linguistics.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable NLP. *CoRR*, abs/2102.12060.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

Xiao Zhang, Ji Wu, Zhiyang He, Xien Liu, and Ying Su. 2018. Medical exam question answering with large-scale reading comprehension. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5706–5713. AAAI Press.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.

Appendix

A. Hyper-parameters of Neural Baselines

The hyper-parameters of Co-Matching, BERT, XLNet are shown in Table 8, Table 9 and Table 10.

	GCRC	C3	RACE	DREAM
train batch size	8	16	16	32
dev batch size	8	16	8	32
Test batch size	8	16	8	32
epoch	50	100	50	100
learning rate	3e-5	1e-3	1e-3	2e-3
seed	128	64	256	128
dropoutP	0.2	0.2	0.2	0.2
emb dim	300	300	300	300
mem dim	150	150	150	150

Table 8: Hyper-parameters of Co-Matching

	GCRC	C3	RACE	DREAM
train batch size	32	16	16	32
dev batch size	4	16	4	16
Test batch size	4	16	4	16
len	320	384	450	384
epoch	6	3	10	3
learning rate	3e-5	1e-5	2e-5	1e-5
gradient accumulation steps	8	8	8	8
seed	42	42	42	42

Table 9: Hyper-parameters of BERT

	GCRC	C3	RACE	DREAM
train batch size	2	2	1	2
dev batch size	2	2	1	2
Test batch size	2	2	1	2
len	320	320	320	180
epoch	16	16	5	8
learning rate	2e-4	1e-3	2e-5	1e-5
gradient accumulation steps	2	2	24	2

Table 10: Hyper-parameters of XLNet

Zero-shot Label-Aware Event Trigger and Argument Classification

Hongming Zhang[*,1,2], **Haoyu Wang**[2], **Dan Roth**[2]
Department of Computer Science and Engineering, HKUST
Department of Computer and Information Science, UPenn
`hzhangal@cse.ust.hk, {why16gzl, danroth}@seas.upenn.edu`

Abstract

Identifying events and mapping them to a pre-defined taxonomy of event types has long been an important NLP problem. Most previous work has relied heavily on labor-intensive, domain-specific, annotation, ignoring the *semantic meaning* of the event types' labels. Consequently, the learned models cannot effectively generalize to new label taxonomies and domains. We propose a zero-shot event extraction approach, which first identifies events with existing tools (e.g., SRL) and then maps them to a given taxonomy of event types in a zero-shot manner. Specifically, we leverage label representations induced by pre-trained language models, and map identified events to the target types via representation similarity. To semantically type the events' arguments, we further use the definition of the events (e.g., argument of type "Victim" appears as the argument of event of type "Attack") as global constraints to regularize the prediction. The proposed approach is shown to be very effective on the ACE-2005 dataset, which has 33 trigger and 22 argument types. Without using any annotation, we successfully map 83% of the triggers and 54% of the arguments to the semantic correct types, almost doubling the performance of previous zero-shot approaches[1].

1 Introduction

Event extraction, the process of identifying events triggers and arguments and classifying them into a set of pre-defined types is an important part of natural language understanding, and a commonly studied NLP task (Grishman et al., 2005). Consider the example shown in Figure 1, where two events (i.e., "war" and "protesting") are identified. By mapping them to [Conflict:Attack] and [Conflict:Demonstrate] and using the knowledge of "Attack" might result in "Demonstrate", we can infer that the war in Iraq is probably the cause of the protesting in Pakistan.

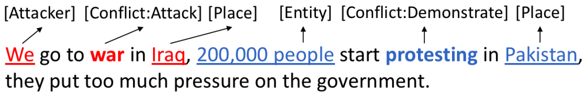

Figure 1: Event classification examples. Two events are highlighted with red and blue colors. Triggers and arguments are in bold and underline fonts, respectively.

Most existing event extraction work (Wadden et al., 2019; Lin et al., 2020) treats event identification as a supervised sequence labeling task and event classification as a supervised classification problem, and relies on large amounts of event-specific annotated text. Take ACE-2005 (Grishman et al., 2005) as an example; the training set of ACE consists of 4,419 events, annotated and typed into 33 event types. Such large-scale and high-quality annotation requires significant expertise, and it facilitates the success of supervised learning models. However, scaling these efforts to new domains and more event types is very costly and unrealistic.

Indeed, there has already been some effort to address the limitation of supervised models on new event types via a transfer-learning based zero-shot event classification approach (Huang et al., 2018). By jointly encoding the event structures (i.e., the relations between event triggers and their arguments) of event mentions and of pre-defined event types, their model learns to map event mentions to the most similar event types. As a result, at inference time, the model can be extended to new types as long as the structures of new types are provided. Nonetheless, the success of this transfer learning approach also heavily relies on the similarity between observed event types and new ones. When

---

[*] This work was done when the first author was visiting the University of Pennsylvania.

[1] Our code and models will be available at `http://cogcomp.org/page/publication_view/942`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1331–1340
August 1–6, 2021. ©2021 Association for Computational Linguistics

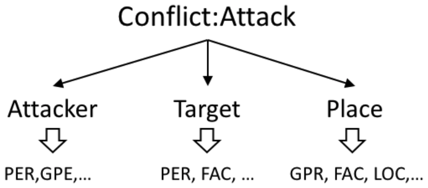

Figure 2: Pre-defined event type demonstration. Event type "Conflict:Attack" is associated with three argument types ("Attacker", "Target", "Place") and each of them is associated with a list of potential entity types.

new event types are different enough from those the model was trained on, the model will struggle.

This paper shows that the whole event extraction pipeline (i.e., identification and classification) can be done without any event-specific annotation. Since the event identification task can also be viewed as a classification task that determines whether the event triggers provided by the SRL models belong to the pre-defined event ontology or not, in this paper, we focus on the classification task. Specifically, we explore a reliable zero-shot solution to mapping observed events to any given set of event types. Unlike previous approaches, we do not use any annotation and only rely on the given event type *definitions*. We classify events by matching the semantics of the identified triggers and arguments to the type names, and then regularize the predictions with the constraints in the pre-defined event ontology.

A pre-defined event type example is shown in Figure 2, where domain experts choose to use "attack" to describe the whole event type and the labels "attacker", "target", and "place" for its roles, since the semantics of these words reflect ones' understanding of this event type. To fully utilize the semantics of these labels, we propose to represent the labels with a cluster of contextualized embeddings rather than just words. In the aforementioned example, we first select several sentences that contain the word "attack" from an external corpus (e.g., New York Times (NYT) (Sandhaus, 2008)). For each selected sentence, we use a pre-trained language model (e.g., BERT (Devlin et al., 2019)) to encode it and then use the resulting embedding of "attack" as a data point in the "attack" cluster. At the inference time, we can then acquire the contextualized representation of identified triggers and arguments, and map them to their corresponding types based on their similarities to those clusters in the embedding space. Beyond the labels, event definitions also provide constraints between types and roles. For example, the role "Attacker" can only appear as an argument of "Conflict:Attack" rather than "Life:Marry," and only a person or a nation can take the role of an "Attacker". In our system, we propose to use these constraints to regularize the zero-shot model. Specifically, we formulate the final inference step as an integer linear programming (ILP) (Roth and tau Yih, 2004) and only produce decisions that satisfy all the constraints.

Our experiments show that the proposed model is very effective on the standard evaluation dataset ACE-2005, which has 33 event trigger types and 22 argument role types. Without using any annotation, we map 83% of the triggers and 54% of the arguments to correct types, almost doubling the performance of the previous best zero-shot event classification approach. When pipelined with our improved zero-shot identification step, we exhibit an zero-shot event extraction pipeline that rivals a SOTA supervised system (Lin et al., 2020) trained on over 6,000 sentences.

2 Task Definition and Notations

We denote the overall sets of predefined event trigger types, argument role types, and entity types as $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{T}$ respectively. Each pre-defined event type (e.g., "Conflict:Attack") $E \in \mathcal{E}$ is associated with several role types $R \in \mathcal{R}_E$ and for each $R \in \mathcal{R}$, the event definition also links it with several possible entity types $T \in \mathcal{T}_R$. Given a sentence S, which has a predicate v, several arguments a, and associated entity types t, the task of zero-shot event classification is mapping v and a to the correct types without any annotation.

3 Model Overview

As shown in Figure 3, the whole framework can be divided into two phases: preparation and prediction. In the preparation phase, we generate the representations for all pre-defined event types and argument roles, which are indicated by blue and orange, respectively. For each type, we select the label word and its synonyms[2] as the anchor words, and then for each of them, we retrieve a list of anchor sentences that use these words from an external corpus. After applying the pre-trained language models to encode all anchor sentences, we can then obtain the contextualized representations of selected anchor words and treat the cluster of these embeddings to

---

[2] For labels that have multiple words, we select words that can best represent the label semantics.

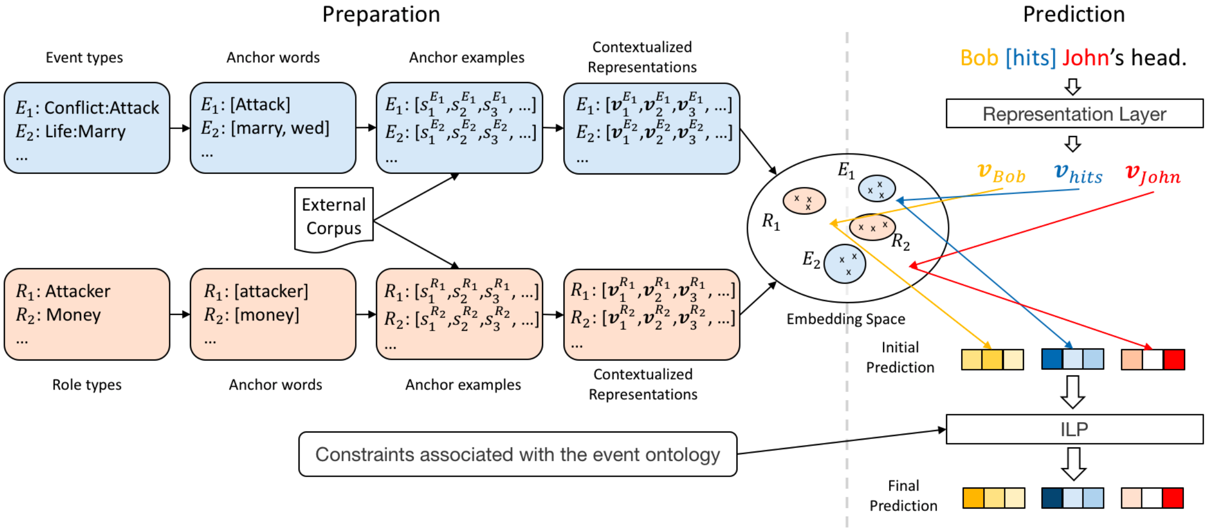

Figure 3: Architecture Overview. Pre-defined types, anchor words, anchor sentences, and contextualized representations for event trigger types and argument role types are indicated with blue and orange, respectively.

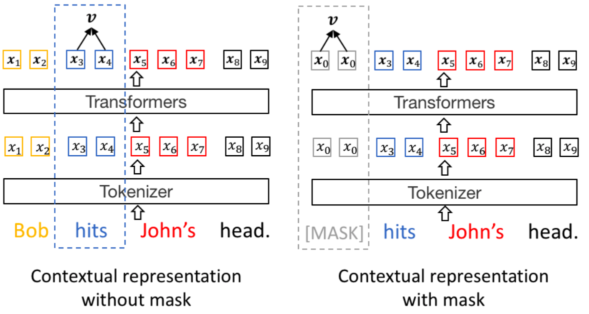

Contextual representation without mask

Contextual representation with mask

Figure 4: Demonstration of the proposed two contextualized representation acquisition methods.

as the type representation. In the prediction phase, given S, v, and a, we first acquire the contextualized representation for the triggers and arguments. After that, in the embedding space, we can easily map them to the most similar types based on the cosine distance to each cluster. Based on these initial predictions, we then leverage constraints provided by the event definitions to regularize the prediction by modeling it as an ILP problem.

4 Preparation

In this section, we introduce the preparation details.

4.1 Anchor Words and Sentences Selection

Since the most frequently-occurred POS of triggers is verb and they typically have multiple senses, we also include their synonyms of the same meaning as anchor words to improve the overall representation quality. In total, we got 107 anchor words for 33 event trigger types and 22 anchor words for 22 event argument types. After that, we then go to an external corpus (NYT corpus (Sandhaus, 2008) in our experiment) for finding anchor sentences that contain the corresponding anchor words.

4.2 Contextualized Representation Generation

As shown in Figure 4, we propose two different representation acquisition methods for triggers and arguments. For triggers, we use all words in S as the input, while for arguments, we mask the target anchor words. The motivation is that most triggers contribute the most important semantic meaning while the semantics of arguments are often inferred from their context rather than the anchor words themselves. For example, many arguments are pronouns or names, which only have weak semantics by themselves and we need to understand them by understanding their context.

For each sentence S and a target anchor word $w^\star$, we first tokenize S based on the type of the target word:

$$X, p_s, p_e = \begin{cases} Tokenize\_Full(S, w^\star) \text{ for } w^\star \in \mathcal{W}^T \\ Tokenize\_Mask(S, w^\star) \text{ for } w^\star \in \mathcal{W}^A, \end{cases} \tag{1}$$

where $\mathcal{W}^T$ and $\mathcal{W}^A$ are sets of anchor words for triggers and arguments, respectively. $X = x_1, x_2, \cdots, x_n$ is the list of tokens. As the tokenizer tool may tokenize words into sub-word pieces, we use p_s and p_e to record the start and end token positions for the anchor word. We then input X into a multi-head transformer module to get the

Explanation	Constraint
One type per trigger.	$\sum_{i\in\lvert\mathcal{E}\rvert} I_t(i) = 1$
One type per argument.	$\forall j\ \sum_{k\in\lvert\mathcal{R}\rvert} I_r(j,k) = 1$
Different arguments in one event must have different types.	$\forall k\ \sum_{j\in\lvert\mathcal{A}\rvert} I_r(j,k) \leq 1$
Predicted trigger and argument type must appear in the ontology .	$\forall i,j,k\ I_t(i) + I_r(j,k) \leq 1$ if E_i and R_k cannot be paired in the ontology.
Entity types of arguments match the requirements.	$\forall i,j,k\ I_t(i) = 0, I_r(j,k) = 0$ if a_i does not match the requirement of E_k or R_k.

Table 1: Selected constraints for the ILP regularization.

	Train	Dev	Test	Overall
# Sentences	19,244	902	676	20,822
# Event triggers	4,419	468	424	5,311
# Event arguments	6,604	759	689	8,052

Table 2: ACE-2005 statistics.

contextualized representations of all tokens.

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n = Transformer(x_1, x_2, \cdots, x_n). \tag{2}$$

We omit the technical details of transformers for the clear representation, but the details are available in the original paper (Vaswani et al., 2017). In the end, we took the mean pooling of embeddings for tokens belonging to the target anchor word as one of its contextualized representation:

$$\mathbf{v} = \frac{\sum_{p_e \leq i \leq p_s} \mathbf{x}_i}{p_e - p_s}. \tag{3}$$

By grouping the acquired representations from all anchor sentences together, we get a cluster of embeddings for each pre-defined type, which can be used for the prediction. For trigger type E_i and argument type R_i, we denote the corresponding vector cluster as $\mathcal{V}^{E_i}$ and $\mathcal{V}^{R_i}$, respectively.

5 Prediction

We then introduce the prediction part. For each identified event, whose trigger and m arguments are denoted as t and $\mathcal{A} = a_1, a_2, \cdots, a_m$, we first acquire the contextualized representations for trigger and arguments following the same way as what we did for anchor words. Similar to the label representations, we acquire the event trigger representation without using masks and the argument embeddings with masks. We denote the resulting embeddings as $\mathbf{t}$ and $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m$. After that, we compute the prediction score from t to a predefined event trigger type E as:

$$f(t, E) = Cos\_Dist(\mathbf{t}, \frac{\sum_{\mathbf{v}\in\mathcal{V}^E} \mathbf{v}}{|\mathcal{V}^E|}), \tag{4}$$

where Cos_Dist represents the cosine distance and $|\mathcal{V}^E|$ means the number of vectors in the cluster of E's label representations. Similarly, for any argument a and pre-defined argument type R, we compute the prediction score via:

$$f(a, R) = Cos\_Dist(\mathbf{a}, \frac{\sum_{\mathbf{v}\in\mathcal{V}^R} \mathbf{v}}{|\mathcal{V}^R|}). \tag{5}$$

After getting the initial predictions with cosine similarities, the next step would be leveraging the constraints to regularize the prediction results. Specifically, we model this problem as an integer linear programming (ILP) problem (Roth and tau Yih, 2004), which maximizes the following objective while satisfying the constraints in Table 1:

$$\operatorname*{argmax}_{I_t, I_a} \sum_{j\in|\mathcal{A}|} (\sum_{i\in|\mathcal{E}|} f(t, E_i) \cdot I_t(i) \cdot \lambda + \sum_{k\in|\mathcal{R}|} f(a_j, R_k) \cdot I_a(j,k)). \tag{6}$$

Here, λ is the hyper-parameter we use to balance the weight of trigger and argument predictions, and I_t and I_a record the final prediction for the trigger and arguments, respectively. I_t is a vector of integer variables with length $|\mathcal{E}|$, and I_a is a matrix of integer variables with the size $|\mathcal{A}| \times |\mathcal{R}|$.

6 Experiment Details

We follow (Lin et al., 2020) and use ACE-2005 (E^+) as the dataset. In total, ACE-2005 contains 33 event types and 22 role types. The original dataset provides the official training, development, and test splits. However, as the proposed model is zero-shot and we do not need any training data, we merge all of them together to be the test set, which is consistent with the setting in (Huang et al., 2018). Detailed statistics about ACE-2005 are presented in Table 2. Considering that (Huang et al., 2018) trains the model with the most frequent ten event types and tests on the other 23 types, we provide two evaluation settings: (A) Evaluation on the least 23 frequent event types and associated role types, which is consistent with the previous work; (B) Evaluation on all 33 event types, which is used to demonstrate the overall performance of our model on the whole dataset. We treat the trigger and argument classification as two separate ranking problems, and follow the previous work (Huang et al., 2018) to report Hit@1, Hit@3, and Hit@5.

Model	# Types		Event Triggers			Event Arguments		
	Train	Test	Hit@1	Hit@3	Hit@5	Hit@1	Hit@3	Hit@5
Frequency	0	23	9.6	27.2	42.5	25.9	63.4	80.6
WSD-embedding	0	23	1.7	13.0	22.8	2.4	2.8	2.8
Transfer Learning (A)	1	23	4.0	23.8	32.5	1.3	3.4	3.6
Transfer Learning (B)	3	23	7.0	12.5	36.8	3.5	6.0	6.3
Transfer Learning (C)	5	23	20.1	34.7	46.5	9.6	14.7	15.7
Transfer Learning (D)	10	23	33.5	51.4	68.3	14.7	26.5	27.7
Label Representation	0	23	79.6 (0.6)	88.2 (1.3)	92.5 (1.7)	25.9 (2.2)	63.2 (1.9)	74.6 (2.0)
Label Representation + ILP	0	23	**80.5** (0.2)	**88.9** (0.3)	**93.2** (0.6)	**68.5** (0.9)	**94.2** (0.1)	**96.8** (0.4)
Frequency	0	33	28.9	53.6	62.7	13.8	33.8	51.0
Label Representation	0	33	81.9 (0.5)	92.6 (0.4)	95.7 (0.2)	17.1 (0.7)	38.0 (0.4)	49.5 (0.9)
Label Representation + ILP	0	33	**82.9** (0.5)	**93.1** (0.1)	**96.2** (0.1)	**53.6** (1.3)	**87.9** (0.4)	**92.4** (0.5)

Table 3: Event trigger and argument classification results on ACE-2005. Best performing models are annotated with the bold font. Standard deviations are shown in brackets.

6.1 Baseline Methods

To the best of our knowledge, only two previous methods were used to solve event typing in a zero-shot manner, thus we compare with both of them:

1. **WSD-embedding** (Huang et al., 2018): The WSD baseline first obtains the sense of event mentions with a word sense disambiguation module (Zhong and Ng, 2010) and then acquires the sense embedding with the skip-gram model (Mikolov et al., 2013). During the inference, it can then map triggers and arguments to the candidate types with the pre-trained word sense embeddings.

2. **Transfer Learning** (Huang et al., 2018): The transfer-learning based zero-shot approach first learns to map the AMR parsing (Wang et al., 2015) result of events to a few observed event types and then apply the learned model to unseen event types. In the original paper, four experiment settings are provided, which are distinguished by the number of seen event types, and we consider all of them to be the baselines.

Besides them, we also present the performance of the "Frequency" baseline, which predicts all triggers and arguments with the most frequent types.

6.2 Implementation Details

We implement our model with Huggingface (Wolf et al., 2019) and use BERT-large (Devlin et al., 2019) as the pre-trained language model. For each anchor word, we randomly select ten anchor sentences from the NYT corpus (Sandhaus, 2008). λ is set to be 100. We implemented the ILP optimization with gurobi[3]. All other hyper-parameters are inherited from BERT. We repeat the experiments five times and report the average performance as well as the standard deviations. The effect of all hyper-parameters is carefully evaluated.

7 Result Analysis

From the results in Table 3, we can observe that:

1. Despite the difficulty of this task (we have 33 and 22 candidates types for triggers and arguments), our model can map 83% of the triggers and 54% of the arguments to the correct types, which shows that when the class labels are carefully designed, their semantics can serve as an excellent signal for the classification task.

2. Compared with the baseline method, our model doubles the performance on the selected 23 event type subset. The main improvement we made is that we do not only use the labels but also put them back into some real usage and then use the contextualized representations to represent them. By doing so, we can best represent the label semantics and leverage them for the classification task.

3. The dataset distribution is imbalanced. For example, 28.9% of the event triggers are "attack," which is relatively simple. This also explains why our model achieves higher performance for triggers on the whole dataset, where we need to map them to 33 types that include "attack", than

---

[3] https://www.gurobi.com/

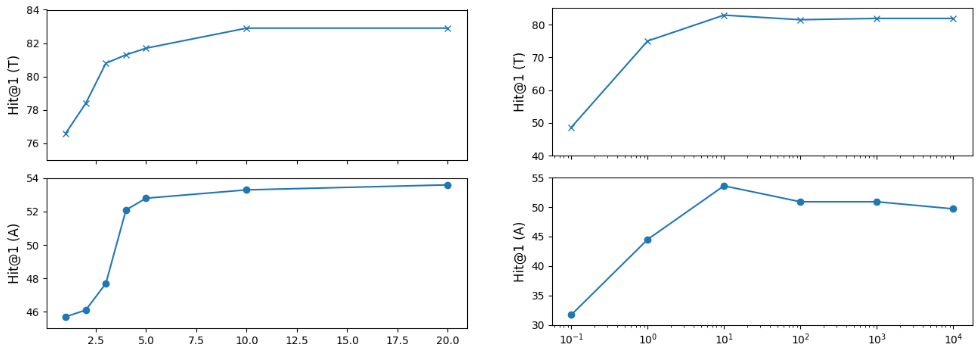

(a) Effect of the anchor sentence quantity n.

(b) Effect of the predicate weight w.

Figure 5: Hyperparameter Analysis.

Model	Hit@1 (T)	Δ	Hit@1 (A)	Δ
Full model	82.9	-	53.6	-
No Context	40.5	-42.4	29.9	-23.7
BERT-base	55.7	-27.2	33.5	-20.1

Table 4: Ablation study. The Hit@1 performance for triggers and arguments are denoted as Hit@1 (T) and Hit@1 (A).

Trigger	Argument	Hit@1 (T)	Hit@1 (A)
w/o mask	w/ mask	82.9	53.6
w/o mask	w/o mask	80.9	41.2
w/ mask	w/o mask	52.7	36.1
w/ mask	w/ mask	52.8	40.0

Table 5: Effect of different representation strategies.

the selected subset, where we only need to map them to 23 types but without “attack”.

7.1 Ablation Study

We present the following ablation studies to show the contribution of different modules:

1. **No Context**: One of the largest contributions of the proposed model is using contextualized representations to represent each label. To demonstrate the importance of the context, we try to remove them and acquire the label representation only with selected anchor words[4].

2. **BERT-base**: To demonstrate the contribution of a good representation model, we replace BERT-large with its weaker version (i.e., BERT-base (Devlin et al., 2019)).

From the results in Table 4, we can see that the context information is crucial to our success. Without the context, the anchor word embeddings can no longer effectively represent the label semantics. Besides that, a good language model also helps better merge the contextual information into the anchor words, which also shows that leveraging the context well is the key to our success.

7.2 Hyper-parameter Analysis

The effect of anchor sentence quantity n and trigger weight λ are shown in Figure 5(a) and 5(b), respectively. First of all, we can observe that ten anchor sentences are enough to achieve a good performance, which helps verify the motivation of this paper that with the careful usage, labels can serve as a crucial semantic signal for zero-shot classification tasks. For λ, as shown in Table 3, before the ILP regularization, the model performs much better on triggers than arguments, that is why we need to give more weights to the trigger (i.e., the model should not change the trigger prediction unless it is very certain about the argument prediction). On the other hand, λ cannot be infinitely large, otherwise, the ILP may simply ignore the argument prediction, which may also hurt the performance. To achieve the balance, we select $\lambda = 10$.

7.3 Representation Acquisition Strategy

As aforementioned, we adopt two different label representation strategies for triggers and arguments. Specifically, we kept the anchor words in the sentence for triggers while masking them for arguments. To clearly show the effectiveness of different strategies, we show the performance of different strategy combinations in Table 5. From the results, we can see that if we apply masks to the

---

[4] Context are also removed for candidate triggers and arguments. For arguments, as there is no context, we also remove the masks.

	P	R	F1
Trigger (I)	58.9	57.8	58.3
Trigger (I + C)	54.1	53.1	53.6
Trigger + Argument (I)	12.0	26.0	16.4
Trigger + Argument (I + C)	4.6	10.0	6.3

Table 6: Zero-shot event extraction performance. "I" and "C" mean the identification and classification.

trigger representations or remove the mask from the argument representation pine, the performance decreases. This observation verifies the assumption that event triggers are often textually similar to the trigger labels and thus keeping these words in the sentence can help to map them. On the other hand, event arguments are often named entities, which are often very different from the labels (e.g., "victim") textually, and what helps determine their roles is the context surrounding them. As a result, we achieve better performance when we mask these arguments and only leverage the context to generate the representations.

8 Zero-shot Event Extraction

Previous experiments have demonstrated that with the help of the label representations and the post-regularization, our zero-shot system can effectively map detected triggers and arguments to the correct types. However, if the goal is to automatically extract events from the raw documents, we still need the support from other NLP modules to identify the event triggers and arguments first. In this section, we present the performance of a zero-shot event extraction pipeline, which combines our classification model with other NLP tools.

8.1 Identification

The first step is identifying event triggers and arguments from the raw sentences. In our pipeline, we use a BERT-based SRL model (Shi and Lin, 2019) and a nominal SRL model[5] to detect verbal and nominal events. A limitation of these models is that they adopt a different event ontology (i.e., FrameNet (Baker et al., 1998)) from ACE-2005. As a result, they could miss some events or detect irrelevant events, which are not annotated as events under the ACE definition. To include more ACE-specified events, we include all nominal anchor words[6] in the sentences as triggers. To filter out events that are not covered by the ACE ontology, we introduce an additional filtering step, whose details are introduced in the next sub-section. Last but not least, considering that the definition of arguments in SRL systems varies from ACE, we include a mention detection module from CogCompNLP (Khashabi et al., 2018) to further detect mentions as argument candidates.

8.2 Filtering and Classification

For triggers, following the previous work (Huang et al., 2018), we first combine the 1,161 event types from FrameNet and 33 event types from the ACE ontology together. Duplicated event types are manually removed. As a result, we got 1,147 event types. For each detected event trigger, we try to map it to all 1,147 event types with the proposed zero-shot classification approach. Considering that 1,147 event types may still be not enough for covering all event types, we need to leave room for other events in the embedding space. To do so, for each event type, we automatically acquire a cluster radius by optimizing the F1 score over all anchor sentences (i.e., the distance from representations of anchor sentences for that type to the mean representation should be smaller than the radius while others should be larger than the radius). If a trigger is mapped to one of the 33 types in ACE and its cosine distance to the mean embedding of that type is smaller than the corresponding cluster radius, it will be categorized into that type. For arguments, we select all mentions that overlap with the ARG0 and ARG1 of selected triggers given by SRL systems to be candidate arguments. All selected triggers and arguments are jointly classified with the proposed classification model.

8.3 Result Analysis

The performance of our zero-shot event extraction pipeline is shown in Table 6. "Identification" requires the model to detect the correct spans of triggers and arguments. "Classification" requires the model to correctly classify the detected triggers and arguments. As the identification task is often viewed as the sequence labeling problem, where

---

[5] As no previous work has applied BERT to the nominal SRL task, we trained one BERT-based nominal SRL model by ourselves.

[6] Some event type labels are nominal (e.g., "bankruptcy") and some are verbal (e.g., "execute"). For nominal labels, we directly use them; for verbal labels, we use their nominal form (e.g., "execution") if available. As the used verb SRL system has already detected almost all verbal triggers, there is no need to add verbal keywords.

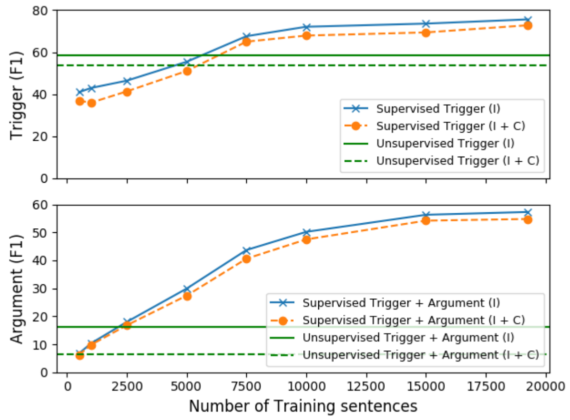

Figure 6: Comparison to the best supervised model.

the Recall@K metric is no longer suitable, we follow the previous work (Lin et al., 2020) to report Precision, Recall, and F1. For the "Argument" evaluation, we provide gold triggers. But for the "Trigger+Argument" evaluation, the model has to detect and classify the correct trigger and associated arguments. We also show the comparison of our system to the current best supervised model OneIE (Lin et al., 2020) in Figure 6. For triggers, our system achieves the comparable performance of the supervised model that is trained with about 6,000 training sentences, which is equivalent to approximately 75% of OneIE's full performance. Considering that our system does not use any annotation and can be easily applied to new datasets and new event definitions, such performance is quite encouraging. In the meantime, compared with triggers, our system cannot detect arguments very well, which is mainly due to poor identification performance. As demonstrated in Figure 7, this is mainly because SRL and ACE adopt different definitions of arguments. SRL requires the arguments to cover all the details, whereas arguments in ACE are often just the key entities. To solve this problem, we propose to use a mention detection module to detect mentions inside the arguments given by SRL systems. Consequently, we cover more gold arguments but also introduce noise. How to automatically identify arguments that fit the ACE definition is a problem worth exploring in the future.

9 Related Works

In this section, we introduce related works about the event extraction and NLP without annotation.

Figure 7: Case study for event extraction. Gold triggers and arguments are indicated with red and blue. Arguments detected by SRL and mention detection module are indicated with underlines and brackets.

9.1 Event Extraction

Previous event extraction works often aim at learning supervised models, employing either symbolic features (Ji and Grishman, 2008; Liao and Grishman, 2010; Liu et al., 2016) or distributed features (Chen et al., 2015b; Lin et al., 2020). To address the problem that supervised models cannot be easily applied to new types, (Huang et al., 2018) separates the event extraction task into two parts (i.e., identification and classification) and proposes a zero-shot transfer-learning classification framework to apply the model trained with seen event types to unseen ones. However, the prerequisite of their high performance is the similarity between seen and unseen event types. Unlike previous works, we do not use any annotation and only leverage the label semantics to classify event triggers and arguments. By combining our classification model and other NLP modules (i.e., SRL and mention detection), we achieve a decent zero-shot event extraction pipeline that can be easily applied to any new documents and event types.

9.2 NLP without Annotation

Solving NLP problems without using annotations has been explored in many NLP tasks including text classification (Chang et al., 2008; Yin et al., 2019), entity typing (Zhou et al., 2018), sequence classification (Rei and Sogaard, 2018), and intent detection (Xia et al., 2018). The idea of leveraging the label semantics was first proposed in the dataless classification framework (Chang et al., 2008), a predecessor name to what is now called zero-shot classification. The idea was to first map the text and labels into a common space using Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) and then pick the label with the highest matching score. This direction was later extended in (Song and Roth, 2014; Chen et al., 2015a; Li et al., 2016a,b; Song et al., 2016). The most significant difference between our work and previous approaches is that, rather than using a fixed representation for each label, we use a group of contextualized embeddings as the representation.

10 Conclusion

In this paper, we present a novel zero-shot classification model for event triggers and arguments. By leveraging the rich semantics contained in labels and other constraints provided by the event definitions, we successfully classify 83% of event triggers and 54% of arguments to their correct types on the ACE-2005 dataset. The ablation study demonstrates that the contextualized usage of the labels and correct way of using the context is key to our success. Further experiments demonstrate that after combining the proposed zero-shot classification model with other available NLP tools, we can effectively extract and classify events without using any annotation. All the codes are submitted.

Acknowledgements

This research is supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the BETTER Program, and by contract FA8750-19-2-1004 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We thank Celine Lee for providing the SRL model and the anonymous reviewers for their valuable feedback.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL 1998*, pages 86–90.

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of Semantic Representation: Dataless Classification. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John A. Carroll. 2015a. Dataless text classification with descriptive LDA. In *Proceedings of AAAI 2015*, pages 2224–2231.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015b. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL 2015*, pages 167–176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI 2007*, pages 1606–1611.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's english ace 2005 system description. *ACE*, 5.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare R. Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of ACL 2018*, pages 2160–2170.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL 2008*, pages 254–262.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. CogCompNLP: Your Swiss Army Knife for NLP. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*.

Chenliang Li, Jian Xing, Aixin Sun, and Zongyang Ma. 2016a. Effective document labeling with very few seed words: A topic model approach. In *Proceedings of CIKM 2016*, pages 85–94.

Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, and Katia P. Sycara. 2016b. Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In *Proceedings of COLING 2016*, pages 2678–2688.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL 2010*, pages 789–797.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of ACL 2020*, pages 7999–8009.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of ACL 2016*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR 2013*.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *Proceedings of NAACL-HLT 2018*, pages 293–302.

Dan Roth and Wen tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling. *CoRR*, abs/1904.05255.

Yangqiu Song, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual Dataless Classification for Languages with Small Wikipedia Presence. In *arXiv*, arXiv:1611.04122, page 17.

Yangqiu Song and Dan Roth. 2014. On Dataless Hierarchical Text Classification. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPs 2017*, pages 5998–6008.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of EMNLP-IJCNLP 2019*, pages 5783–5788.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proceedings of NAACL-HLT 2015*, pages 366–375.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of EMNLP 2018*, pages 3090–3099.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking Zero-shot Text Classification: Datasets, Evaluation, and Entailment Approach. In *emnlp*.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of ACL 2010*, pages 78–83.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-Shot Open Entity Typing as Type-Compatible Grounding. In *emnlp*.

Incorporating Global Information in Local Attention for Knowledge Representation Learning

Yu Zhao[1,*], Han Zhou[2,*], Ruobing Xie[3], Fuzhen Zhuang[4,5], Qing Li[1], Ji Liu[6]
[1]Fintech Innovation Center, Financial Intelligence and Financial Engineering Key Laboratory,
Southwestern University of Finance and Economics, Chengdu, China
[2]Department of Computer Science, The University of Hong Kong, China
[3]WeChat Search Application Department, Tencent, Beijing, China
[4] Institute of Artificial Intelligence, Beihang University, Beijing 100191, China
[5]Xiamen Data Intelligence Academy of ICT, CAS, China
[6]AI platform and Seattle AI Lab, Kwai Inc.

Abstract

Graph Attention Networks (GATs) have proven a promising model that takes advantage of localized attention mechanism to perform knowledge representation learning (KRL) on graph-structure data, e.g., Knowledge Graphs (KGs). While such approaches model entities' local pairwise importance, they lack the capability to model global importance relative to other entities of KGs. This causes such models to miss critical information in tasks where global information is also a significant component for the task, such as in knowledge representation learning. To address the issue, we allow the proper incorporation of global information into the GAT family of models through the use of scaled entity importance, which is calculated by an attention-based global random walk algorithm. In the context of KRL, incorporating global information boosts performance significantly. Experimental results on KG entity prediction against the state-of-the-arts sufficiently demonstrate the effectiveness of our proposed model.

1 Introduction

Graph Attention Networks (GATs) have been successfully applied to various tasks over graphs (Velickovic et al., 2018; Lee et al., 2018b), such as graph classification (Wu et al., 2019b; Lee et al., 2018a), link prediction (Abu-El-Haija et al., 2018), and node classification (Lee et al., 2019; Zhang et al., 2020a). GATs learn from the underlying graph structure by making use of localized attention mechanism (Wu et al., 2019a; Xu et al., 2019; Vashishth et al., 2020b), where the hidden representation of each node is computed by recursively aggregating and attending over its corresponding local neighbors' features, and the weighting coefficients are calculated inductively with self-attention strategy (Thekumparampil et al., 2018; Qian et al., 2018; Zhang et al., 2018). The original GATs perform only on single-relational homogeneous graphs (Velickovic et al., 2018; Wang et al., 2019b). Recent advancements were proposed to operate on more general and prevalent multi-relational graphs (Wang et al., 2019b; Hong et al., 2020; Nathani et al., 2019; Zhang et al., 2020c), such as the representative Knowledge Graphs (KGs) which contain multiple types of entities (nodes) and relationships (edges) (Zhou et al., 2018; Han et al., 2018; Wang et al., 2019a; Zhao et al., 2020). However, these approaches can only exploit localized features within the neighborhood of individual entities (Nathani et al., 2019; Busbridge et al., 2019; Zhang et al., 2020c). For some tasks, such simplified localized feature aggregation may be sufficient, but insufficient for knowledge representation learning (KRL) tasks that also need exploring global information (Xie et al., 2020).

In this paper, we concentrate on how to incorporate **global information** in **local attention** for knowledge representation learning. Specifically, we allow the proper incorporation of global information into the GAT family of models through the use of scaled entity importance, which is estimated by a global random walk algorithm upon the whole graph structural information. In KGs, entity importance[1] indicates the global significance or authority of an entity. Intuitively, it can be quite beneficial if an entity attends more to its "authoritative" neighbors that have high scores of global entity importance. For instance, a movie *"Titanic"* links to different actors, among which a superstar

---

[*] Equal Contribution. Corresponding author: Y. Zhao (zhaoyu@swufe.edu.cn).

[1] The notions of its counterparts, e.g., global node importance or object authority, have been widely studied in graphs (Li et al., 2012; Liu et al., 2017; Park et al., 2019), which enable a number of applications such as Web search (Brin and Page, 1998; Kleinberg, 1999), social network analysis (Weng et al., 2010), RecSys (Jing et al., 2014), query disambiguation (Makris et al., 2012; Saxena et al., 2020).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1341–1351
August 1–6, 2021.

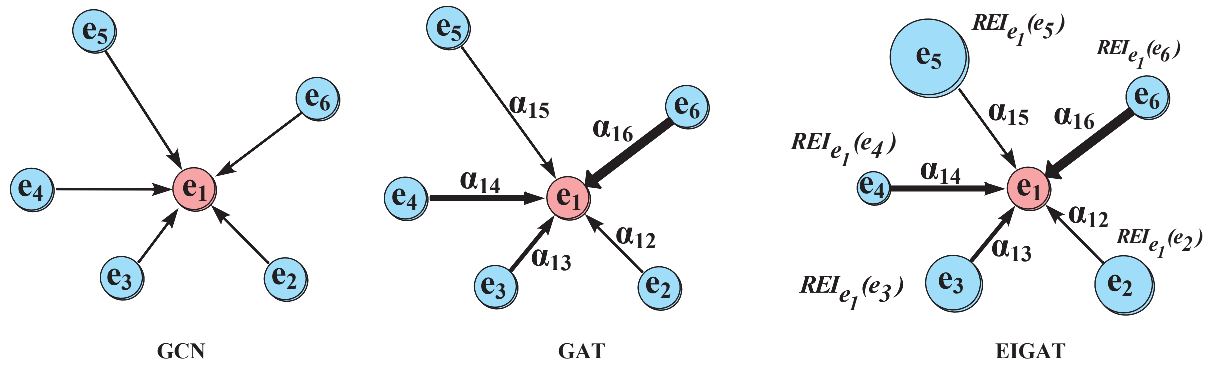

Figure 1: An illustration of Graph Neural Network architectures that model on a graph which is centered on node e_1 with its one-hop neighbors. Left: Graph Convolution Network (GCN); Center: Graph Attention Network (GAT); **Right: our proposed model (EIGAT) that incorporate global information in local attention through the use of relative entity importance (REI)**. $REI_{e_1}(e_i)$ is calculated by an attention-based global random walk algorithm upon the whole graph. GAT parameterizes the edge weights based on local attention score (α_{1i}, also represented by the distinct edge widths). Our EIGAT adds the relative importance score (represented by different scaling of nodes), which is derived from global structural information. Note that although relationship should be drawn in the knowledge graph, for clarity, we intentionally ignore it here, which does not hurt the presentation of the basic idea of our model in this paper.

(e.g. *"Leonardo Dicaprio"*) may be more indicative than other actors.

In this paper, we propose a novel **E**ntity **I**mportance-aware **G**raph **AT**tention Networks, EIGAT, which incorporates global entity importance in local attention mechanism for learning effective knowledge representations. As shown in Figure 1, we give a brief illustration of our proposed EIGAT, which is compared to early proposed GCN (Kipf and Welling, 2017) and GAT (Velickovic et al., 2018). In EIGAT, the importance scores of all entities are expected to be estimated upon global information and to be incorporated in local entity aggregation (Equation 5) for building better entity embeddings. In particular, we provide an attention-based random walk approach to estimate entity importance upon global structural information for serving EIGAT. We conduct extensive experiments on several different types of KGs by entity prediction against state-of-the-art methods, which sufficiently demonstrate our proposed EIGAT can successfully incorporating global information in local attention to improve knowledge representation learning.

The contributions of this paper are threefold:

- We propose to incorporate global information in local attention for knowledge representation learning.
- We propose EIGAT, a novel entity importance-aware graph attention networks which incorporate global entity importance into local entity aggregation.
- The extensive experimental results demonstrate the efficacy of our proposed model in link prediction.

2 Related Work

To make this paper self-contained, we introduce some related topics here on Knowledge Representation Learning and Graph Neural Networks (GNNs).

2.1 Knowledge Representation Learning (KRL)

In recent years, knowledge representation learning on KGs has been a hot research topic (Xiao et al., 2017; Shi and Weninger, 2017; Ebisu and Ichise, 2019; Balazevic et al., 2019; Zhang et al., 2020b). These methods roughly fall into four categories: (i) Translational-based models, which view relations as translations from a head entity to a tail entity, such as Trans(E, H, R, D and G) (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Ji et al., 2015; Xiao et al., 2016), ComplEx (Trouillon et al., 2016), JoBi ComplEx (Balkır et al., 2019). (ii) Tensor factorization based models, which assume the score of a triple can be factorized into several tensors, such as RESCAL (Nickel et al., 2011), NTN (Socher et al., 2013), DistMult (Yang et al., 2015), HOLE (Nickel et al., 2016). (iii) CNN-based models, which use convolution over embed-

Table 1: Different variants of Graph Neural Networks: GNN (Scarselli et al., 2008), GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), and our proposed model EIGAT that incorporates global information in local attention.

GNN-based Models	Node Aggregation	Operation	Key Concepts
GNN (Scarselli et al., 2008)	$x_j = f\left(l_j, l_{co[j]}, x_{ne[j]}, l_{ne[j]}\right)$	$f(\cdot)$	Transduction function
GCN (Kipf and Welling, 2017)	$\mathbf{E}^\ell = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\mathbf{E}^{\ell-1}W^{\ell-1}\right)$	$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$	Convolutional operation
GAT (Velickovic et al., 2018)	$\vec{\mathbf{e}_j}^\ell = \sigma\left(\sum_{e_i \in In(e_j)} \alpha_{ij}^\ell \cdot W^{\ell-1}\vec{\mathbf{e}_i}^{\ell-1}\right)$	α_{ij}	Local attention
EIGAT (our proposed)	$\vec{\mathbf{e}_j}^\ell = \sigma\left(\sum_{e_i \in In(e_j)} REI_{e_j}(e_i)^\ell \sum_{k \in \mathcal{R}_{ij}} \alpha_{ikj}^\ell \cdot \vec{\mathbf{v}_{ikj}}^\ell\right)$	$REI_{e_j}(e_i),\ \alpha_{ikj}$	Global information + Local attention

dings to predict links, such as ConvE (Dettmers et al., 2018), ConvKB (Nguyen et al., 2018), InteractE (Vashishth et al., 2020a), ReInceptionE (Xie et al., 2020), and ParamE (Che et al., 2020). (iv) Graph neural network-based models, such as R-GCN (Schlichtkrull et al., 2018), A2N (Bansal et al., 2019), Nathani's (Nathani et al., 2019), RGHAT (Zhang et al., 2020c), ATTH (Chami et al., 2020), which yielded state-of-the-art performance for KRL. Along this line, we focuses on a GNN-based approach to deal with knowledge representation learning task.

2.2 Graph Attention Networks (GATs)

Graph Neural Networks (GNNs) develop a deep neural network to deal with arbitrary graphs for representation learning (Scarselli et al., 2008; Zhou et al., 2019; Hou et al., 2020). Graph Convolutional Networks (GCNs) are one of their most prominent progress (Schlichtkrull et al., 2018; Wu et al., 2019a; Xu et al., 2019; Vashishth et al., 2020b), which generalize local convolutional operation on the graph-structured data, i.e. gather information from one-hop neighbors and all neighbors contribute equally in the message passing. Inspired by the successful development of the attention mechanism in NLP and CV, Velickovic et al. (2018) proposed Graph Attention Networks (GATs) by incorporating local attention mechanism (Vaswani et al., 2017; Qian et al., 2018; Lu and Li, 2020) into GCNs, which calculate the hidden states of each node by attending over its neighbors (Thekumparampil et al., 2018; Lee et al., 2018b; Yang et al., 2019).

Recently, several advanced extensions of GATs were proposed for operating on knowledge graphs. Han et al. (2018) proposed to jointly apply attention to KGs and external text data. Busbridge et al. (2019) proposed RGAT by extending non-relational GATs to incorporate relational information, but with poor performance. Nathani et al. (2019) proposed a triple-level attention model that captures the integrated features of both entity and relation in a given entity's neighborhood, and Zhang et al. (2020c) proposed a two-level hierarchical attention mechanism. These studies are related to our work in the sense that we all use GNNs to capture more structural information in KGs. However, all of them ignore global information in local attention computation.

Most recently, (Xu et al., 2020) proposed a Transformer-based model to enhance the copy mechanism for abstractive summarization by considering the global importance of each source word based on the degree centrality in the Transformer, which inspires our idea of incorporating global information in local attention for KRL. Table 1 summarizes the key concepts and other different settings of GNNs.

3 Methodology

In this section, we introduce the details of the proposed EIGAT model that incorporates global information in local attention for knowledge representation learning on KGs. We start by describing a single entity importance-aware graph attention layer, which is the building block of our model's overall architecture. Before that, we briefly introduce the notations of this paper.

Notations. In a graph attention networks with L layers, the input to ℓ-th layer ($\ell = 1, \ldots, L$) are two embedding sets: **(1)** the *output entity embeddings* from (ℓ-1)-th layer which is represented by a matrix $\mathbf{E}^{\ell-1} \in \mathbb{R}^{\eta^{\ell-1} \times N_e}$, $\mathbf{E}^{\ell-1} = \{\vec{\mathbf{e}_1}^{\ell-1}, \vec{\mathbf{e}_2}^{\ell-1}, \ldots, \vec{\mathbf{e}_{N_e}}^{\ell-1}\}$, where N_e is the number of entities, and $\eta^{\ell-1}$ is the dimension of output entity embedding in (ℓ-1)-th layer. **(2)** the *output relationship embeddings* from (ℓ-1)-th layer, denoted by a matrix $\mathbf{R}^{\ell-1} \in \mathbb{R}^{\zeta^{\ell-1} \times N_r}$, $\mathbf{R}^{\ell-1} = \{\vec{\mathbf{r}_1}^{\ell-1}, \vec{\mathbf{r}_2}^{\ell-1}, \ldots, \vec{\mathbf{r}_{N_r}}^{\ell-1}\}$, where N_r and $\zeta^{\ell-1}$ represent the number of relationships and the output relationship's feature dimension in (ℓ-1)-th layer, respectively. The ℓ-th layer then produces the corresponding new output embedding matrices (of potentially different cardinality), $\mathbf{E}^{\ell} \in \mathbb{R}^{\eta^{\ell} \times N_e}$ and $\mathbf{R}^{\ell} \in \mathbb{R}^{\zeta^{\ell} \times N_r}$. Specifically, we describe the ℓ-th graph attention layer.

3.1 Local Attention Evaluation

A triple relation $t_{ij}^{k} = (e_i \xrightarrow{r_k} e_j)$ indicates a relationship r_k between head entity e_i and tail entity e_j. Following (Nathani et al., 2019), the representation $\vec{\mathbf{v}_{ikj}}^{\ell}$ of the triple t_{ij}^{k} is built as follows:

$$\vec{\mathbf{v}_{ikj}}^{\ell} = \mathbf{W}_1^{\ell} \cdot \left[\vec{\mathbf{e}_i}^{\ell-1} \| \vec{\mathbf{r}_k}^{\ell-1} \| \vec{\mathbf{e}_j}^{\ell-1}\right], \tag{1}$$

where $\mathbf{W}_1^{\ell}$ denotes a linear transformation matrix in ℓ-th layer, $\vec{\mathbf{e}_i}^{\ell-1}$, $\vec{\mathbf{r}_k}^{\ell-1}$ and $\vec{\mathbf{e}_j}^{\ell-1}$ denote the output embeddings of e_i, r_k and e_j in (ℓ-1)-th layer, respectively. $\|$ represents concatenation. We then calculate the absolute relation attention value b_{ikj}^{ℓ} of the triple t_{ij}^{k}.

$$b_{ikj}^{\ell} = \text{LeakyReLU}\left(\mathbf{W}_2^{\ell} \cdot \vec{\mathbf{v}_{ikj}}^{\ell}\right), \tag{2}$$

where $\mathbf{W}_2^{\ell}$ and LeakyReLU are a linear weight vector in ℓ-th layer and a non-linearity active function respectively that act upon the embedding $\vec{\mathbf{v}_{ikj}}^{\ell}$ in turn. We then utilize *softmax* to evaluate the relative relation attention value α_{ikj}^{ℓ} of the triple t_{ij}^{k} in ℓ-th layer.

$$\alpha_{ikj}^{\ell} = \text{softmax}_{ik}(b_{ikj}^{\ell}) = \frac{\exp\{b_{ikj}^{\ell}\}}{\sum\limits_{e_n \in In(e_j)} \sum\limits_{r \in \mathcal{R}_{nj}} \exp\{b_{nrj}^{\ell}\}}. \tag{3}$$

$In(e_j)$ denotes the neighbors pointing to targeted tail entity e_j, $\mathcal{R}_{nj}$ denotes the set of relationships between e_n and e_j.

3.2 Global Entity Importance Estimation

To obtain global entity importance $EI(e_i)$ of an entity e_i, we formally introduce a relation attention-based global random walk method, as follows:

$$EI(e_i)^t = (1-d) + d \times \sum_{e_m \in In(e_i)} \sum_{r \in \mathcal{R}_{mi}} \frac{b_{mri}}{\sum\limits_{e_n \in Out(e_m)} \sum\limits_{\bar{r} \in \mathcal{R}_{mn}} b_{m\bar{r}n}} EI(e_m)^{t-1}, \tag{4}$$

where d is a hyperparameter denoting the probability that an imaginary surfer randomly moves to a neighboring entity. $(1-d)$ denotes the probability of teleporting to any other entities randomly, which is able to alleviate the *information island problem* caused by the isolated entities that lack of any in-degree or out-degree neighbors (e.g. #median in-degree=0 in NELL-995 in Table 2). $Out(e_m)$ denotes the neighborhoods that an entity e_m points to. $EI(e_m)^{t-1}$ denotes the EI score of the entity e_m in (t-1)-th iteration. The random walk distance[2] t depends on both the number of attention layers L and training epochs C, $t \in (1, L \times C]$. The relation weights (e.g. b_{mri}) are calculated by Equation (2). Unlike conventional fixed weights-based random walk methods (Mihalcea and Tarau, 2004; Florescu and Caragea, 2017), a novelty is that the dynamic relation weights (e.g. b_{mri}) are iteratively and automatically optimized during training by the graph attention mechanism. In line with the theoretical desiderata for modeling node importance in MRGs, this method develops the following essential characteristics: **(i)** *Neighborhood-awareness*, i.e. neighboring EI scores can be taken into account when a given entity's importance score is modeled. **(ii)** *Relationship-awareness*, i.e. different relationships could play a different role in propagating EI score. **(iii)** *Centrality-awareness*, i.e. more central nodes inherently and reasonably would be more important than less central nodes. **(iv)** *Universal and flexible*, i.e. it utilizes only graph global structural information.

3.3 Incorporate Global Information in Local Attention

Though attention mechanism can assign different importance to nodes via learned weights, it is still a local computation. The attention value, e.g., α_{ikj} in Equation (3), is the function of pairwise feature interaction within local neighborhood and do not take account of entity importance from global graph structure. To this end, we incorporate global information in local attention computation, as shown in Figure 1 (EIGAT).

Specifically, to generate the output embedding $\vec{\mathbf{e}_j}^{\ell}$ of tail entity e_j in ℓ-th layer, we incorporate global relative head entity importance $REI_{e_j}(e_i)^{\ell}$ in local attention to conduct entity aggregation with its associated triple representations $\vec{\mathbf{v}_{ikj}}^{\ell}$ weighted

---

[2] To denote EI score of e_i in ℓ-th layer explicitly, we omit the training epoch symbol and denote it as $EI(e_i)^{\ell}$ in the following.

by their relative attention values α_{ikj}^{ℓ} , as follows:

$$\vec{\mathbf{e}}_j^{\,\ell} = \sigma\left(\sum_{e_i \in In(e_j)} REI_{e_j}(e_i)^{\ell} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ikj}^{\ell} \cdot \vec{\mathbf{v}_{ikj}}^{\ell}\right), \quad (5)$$

and

$$\begin{aligned} REI_{e_j}(e_i)^{\ell} &= \mathrm{softmax}_{In(e_j)}(EI(e_i)^{\ell}) \\ &= \frac{\exp\{EI(e_i)^{\ell}\}}{\sum\limits_{e_i' \in In(e_j)} \exp\{EI(e_i')^{\ell}\}}, \forall e_i \in In(e_j)\,. \end{aligned} \quad (6)$$

In Eq. (5), we bring in global relative entity importance $REI_{e_j}(e_i)^{\ell}$ of different head entities in $In(e_j)$ for learning more about those significant neighboring entities, and thus could get better knowledge representations for the targeted tail entity e_j.

To stabilize the learning process of self-attention, as suggested by (Velickovic et al., 2018), we employ multi-head attention. Specifically, M independent attention mechanisms execute the transformation of Eq. (5), and then their features are concatenated as:

$$\vec{\mathbf{e}}_j^{\,\ell} = \Big\Vert_{m=1}^{M} \sigma\left(\sum_{e_i \in In(e_j)} REI_{e_j}(e_i)^{\ell,m} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ikj}^{\ell,m} \cdot \vec{\mathbf{v}_{ikj}}^{\ell,m}\right) \quad (7)$$

We conduct a linear transformation on input relationship embedding $\vec{\mathbf{r}_k}^{\ell-1} \in \mathbb{R}^{\zeta^{\ell-1}}$ in ℓ-th layer as:

$$\vec{\mathbf{r}_k}^{\ell} = \mathbf{W}^{\ell,\mathcal{R}} \cdot \vec{\mathbf{r}_k}^{\ell-1}, \quad (8)$$

where $\mathbf{W}^{\ell,\mathcal{R}} \in \mathbb{R}^{\zeta^{\ell} \times \zeta^{\ell-1}}$ is a weight matrix, $\zeta^{\ell-1}$ and ζ^{ℓ} are dimensions of input and output relationship embeddings, respectively. $\vec{\mathbf{r}_k}^{\ell} \in \mathbb{R}^{\zeta^{\ell}}$ represents the output relationship embedding in the ℓ-th layer.

4 Model Architecture

Our model follows an encoder-decoder framework: **(i)** the encoder model includes L attention layers, **(ii)** the decoder model provides a scoring function (Eq. 11) to calculate the likelihood of given triples being valid. Based on it, the KG incompleteness issue is expected to be alleviated by link prediction (Section 5), i.e., inferring possible missing relations, e.g. $(e_i, r_k, ?)$ or $(?, r_k, e_j)$.

4.1 Encoder

Based on a single attention layer introduced above, we build the overall architecture of our encoder model with L layers. In practice, we set L=2 for our encoder model. In the final L-th layer, instead of concatenation (Equation 7), we employ averaging and delay applying the final non-linearity activation:

$$\vec{\mathbf{e}}_j^{\,L} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M} \sum_{e_i \in In(e_j)} REI_{e_j}(e_i)^{L,m} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ikj}^{m} \vec{\mathbf{v}_{ikj}}^{L,m}\right) \quad (9)$$

To keep initial entity information in the final embedding, we obtain the final entity embedding $\vec{\mathbf{e}}^{\star} \in \mathbb{R}^{\eta^L}$ by combining the transformed initial embeddings $\vec{\mathbf{e}}^0 \in \mathbb{R}^{\eta^0}$ and the output entity embedding $\vec{\mathbf{e}}^L \in \mathbb{R}^{\eta^L}$ of the L-th layer, as follows:

$$\vec{\mathbf{e}}^{\star} = \mathbf{W}^{\star} \cdot \vec{\mathbf{e}}^0 + \vec{\mathbf{e}}^L, \quad \forall e \in \mathcal{E}\,. \quad (10)$$

$\mathbf{W}^{\star} \in \mathbb{R}^{\eta^L \times \eta^0}$ is a projecting matrix. The initial entity embeddings (i.e. $\vec{\mathbf{e}}^0, \forall e \in \mathcal{E}$) and relationship embeddings (i.e. $\vec{\mathbf{r}}^0, \forall r \in \mathcal{R}$) are pre-trained by Bordes et al. (2013).

4.2 Decoder

Among the existing KG completion (KGC) models, we utilize the most recent model ConvKB (Nguyen et al., 2018) as decoder model[3]. Given a triple t_{ij}^k, the scoring function is formally defined as:

$$f(t_{ij}^k) = \Big(\Big\Vert_{m=1}^{|\Omega|} g([\vec{\mathbf{e}_i}^{\star}, \vec{\mathbf{r}_k}^{L}, \vec{\mathbf{e}_j}^{\star}] * \omega^m)\Big) \cdot \mathbf{W}\,, \quad (11)$$

where Ω denotes the set of filters, τ=$|\Omega|$ and $\omega \in \Omega$. Ω and $\mathbf{W}$ are shared parameters and independent of e_i, r_k and e_j. $g(\cdot)$ is an activation function such as ReLU. $*$ denotes a convolution operator. These τ feature maps are concatenated into a single vector $\in \mathbb{R}^{\tau\phi}$ which is then computed with a weight vector $\mathbf{W} \in \mathbb{R}^{\tau\phi}$ via a dot product to give a likelihood score for the triple t_{ij}^{κ}. ϕ denotes the dimension of entity and relation embeddings. In practice, we set ϕ=η^L=ζ^L for ConvKB.

4.3 Optimization

We utilize a two-step training procedure for the encoder-decoder framework, which is a routine optimization way for it (Zhou et al., 2019). **(i)** We first train the encoder model to learn the embeddings of entities and relationships, by minimizing

---

[3] We choose ConvKB here, and it is not difficult for other KGC methods, such as CapsE (Nguyen et al., 2019), ConvE (Dettmers et al., 2018), etc. Note that we also tried different models as decoder, but found that ConvKB performs best.

Table 2: Statistics of datasets.

Datasets	#Entities	#Rel	#Edges				Mean in-degree	Median in-degree	Density
			Train	Validation	Test	Total			
Kinship	104	25	8544	1068	1074	10,686	82.15	82.5	0.998
FB15k-237	14,541	237	272,115	17,535	20,466	310,116	18.71	8	0.001
NELL-995	75,492	200	149,678	543	3992	154,213	1.98	0	2.71E-5

a *hinge-loss* function, as follows:

$$\mathcal{L}_1 = \sum_{t_{ij}^k \in \mathcal{G}} \sum_{{t_{ij}^k}' \in \mathcal{G}'} \max \left\{ h_{{t_{ij}^k}'} - h_{t_{ij}^k} + \gamma, 0 \right\} . \tag{12}$$

Here, $h_{t_{ij}^k} = \|\vec{\mathbf{e}_i}^{\star} + \vec{\mathbf{r}_k}^{L} - \vec{\mathbf{e}_j}^{\star}\|_{\ell 1}$ indicates the translational scoring function of the triple t_{ij}^{κ} (Bordes et al., 2013). $\gamma > 0$ is a margin hyper-parameter. **(ii)** We then train and learn the parameters of the decoder model ConvKB for link prediction , by minimizing a *soft-margin loss* function, as follows:

$$\mathcal{L}_2 = \sum_{t_{ij}^k \in \{\mathcal{G} \cup \mathcal{G}'\}} \log \left(1 + \exp \left(l_{t_{ij}^k} \cdot f(t_{ij}^k) \right) \right) + \lambda \left\| \mathbf{W} \right\|_2^2 , \tag{13}$$

in which, $l_{t_{ij}^k} = \begin{cases} 1, & t_{ij}^k \in \mathcal{G} \\ -1, & t_{ij}^k \in \mathcal{G}' \end{cases}$. $\mathcal{G}$ and $\mathcal{G}'$ are the sets of positive triples and negative triples, respectively.

$$\mathcal{G}' = \{t_{i'j}^k | e_i' \in \mathcal{E} \setminus e_i\} \cup \{t_{ij'}^k | e_j' \in \mathcal{E} \setminus e_j\} . \tag{14}$$

5 Experiments

We evaluate the effectiveness of our proposed model EIGAT by link prediction (determined by Equation 11), which aims to infer possible missing relations, i.e., predict e_j given $(e_i, r_k, ?)$ or predict e_i given $(?, r_k, e_j)$.

5.1 Datasets

We use three public benchmark datasets for link prediction experiments, including: *Kinship* (Lin et al., 2018), *NELL-995* (Xiong et al., 2017), *FB15K-237* (Toutanova et al., 2015), where we discard another popular dataset *WN18RR* due to its too sparse to learn global information. The basic statistics of all datasets are included in Table 2. To explore the performance of our proposed model on different datasets with different global topology characteristics, we compute their density value (Coleman and Moré, 1983) and report them in Table 2. Since the densities in *NELL-995* is sparser than *Kinship* and *FB15K-237*, and its median in-degree even is 0, it is relative hard for global entity importance estimation in *NELL-995*.

Definition 1. ***(Graph Density)****. Graph density aims to measure how sparse a graph is. Similar to (Coleman and Moré, 1983), given a graph $\mathcal{G}$, it's formally defined as follows:*

$$D(\mathcal{G}) = \frac{E}{N(N-1)} , \tag{15}$$

where N denotes the number of nodes in $\mathcal{G}$, and E denotes the number of edges in $\mathcal{G}$. The lower the $D(\mathcal{G})$, the sparser the graph is.

Table 3: Hyperparameters for the encoder model EIGAT on all datasets.

Datasets	Kinship	NELL-995	FB15k-237
Learning rate	1e-2	1e-3	1e-3
Weight decay	1e-6	1e-6	1e-6
Epochs	4000	3000	3200
Dropouts	0.3	0.5	0.5
Leaky Relu	0.2	0.2	0.2
nheads	2	2	2
Final dimensions	200	200	200
Negative ratio	2	2	2
Margin	1	1	1
RW parameter d	0.85	0.85	0.85

Table 4: Hyperparameters for the decoder model EIGAT on all datasets.

Datasets	Kinship	NELL-995	FB15k-237
Learning rate	1e-2	1e-3	1e-3
Weight decay	1e-5	5e-6	5e-7
Epochs	400	200	200
Dropouts	0.0	0.3	0.2
Filters	50	400	50

5.2 Baselines

To demonstrate the effectiveness of our proposed model EIGAT for link prediction, we compare it with the following state-of-the-art (SOTA) baselines:

- TransE (Bordes et al., 2013): a most widely used and early KGC models.
- DistMult (Yang et al., 2015): a popular tensor factorization-based KGC model which uses a bi-linear scoring function to calculate knowledge triples' scores.

Table 5: **Link prediction results** on *Kinship* and *NELL-995*. The results of baselines are directly taken from the original papers. The best scores are in **bold**.

	Kinship					**NELL-995**				
			Hits@N (%)					**Hits@N (%)**		
	MR	**MRR**	**@1**	**@3**	**@10**	**MR**	**MRR**	**@1**	**@3**	**@10**
TransE (Bordes et al., 2013)	6.8	0.309	0.9	64.3	84.1	2100	0.401	34.4	47.2	50.1
DistMult (Yang et al., 2015)	5.26	0.516	36.7	58.1	86.7	4213	0.485	40.1	52.4	61
ComplEx (Trouillon et al., 2016)	2.48	0.823	73.3	89.9	97.11	4600	0.482	39.9	52.8	60.6
ConvE (Dettmers et al., 2018)	2.03	0.833	73.8	91.7	98.14	3560	0.491	40.3	53.1	61.3
ConvKB (Nguyen et al., 2018)	3.3	0.614	43.62	75.5	95.3	**600**	0.43	37.0	47	54.5
R-GCN (Schlichtkrull et al., 2018)	25.92	0.109	3	8.8	23.9	7600	0.12	8.2	12.6	18.8
Nathani's (Nathani et al., 2019)	1.94	0.904	85.9	94.1	98	965	0.530	44.7	56.4	69.5
EIGAT (Ours)	**1.66**	**0.963**	**94.8**	**96.6**	**98.4**	1210	**0.545**	**46.4**	**58.4**	**71.5**

- ComplEx (Trouillon et al., 2016): an advanced extension of DistMult which encodes entities and relationships into complex vector space instead of real-valued vector space.
- ConvE (Dettmers et al., 2018): a popular convolutional network-based KGC model.
- ConvKB (Nguyen et al., 2018): another SOTA convolutional network-based KGC model.
- R-GCN (Schlichtkrull et al., 2018): an advanced extension of GCN that can effectively model multi-relational data.
- Nathani's (Nathani et al., 2019): a recent KGC model that models the local neighborhood via graph relational attention network.
- A2N (Bansal et al., 2019): a recent model that learns query-dependent representations of entities based on a GNN structure.
- HAKE (Zhang et al., 2020b): a SOTA KGC model that models semantic hierarchies
- InteractE (Vashishth et al., 2020a):a recent extension of ConvE that increase the interaction between relation and entity embeddings.
- ReInceptionE (Xie et al., 2020): a recent extension of ConvE that uses local-global structural information.
- ParamE (Che et al., 2020): another extension of ConvE that use relation embeddings.
- ATTH (Chami et al., 2020): a SOTA model that use hyperbolic space and attention-based geometric transformation.
- RGHAT (Zhang et al., 2020c): a SOTA KGC model that models the local neighborhood via hierarchical attention mechanism.

5.3 Evaluation Protocol

We utilize ranking criteria for evaluation. For each testing triple, we remove the head entity or tail entity and replace it by each of the entities in $\mathcal{E}$ in turn. The model scores of the corrupted triples would be computed by the decoder model (Eq. 11) and then sorted by descending order. We can obtain the exact rank of the correct triple in the candidates. Similar to most baselines, we report the experimental results in "Filter" setting, i.e. removing corrupted triples that are already present in datasets during ranking. The evaluation metrics include: the mean reciprocal rank (MRR), mean rank (MR), and the proportion of correct entities ranked in the top N (HITS@N, N=1, 3, 10).

5.4 Training Protocol

Table 3 and Table 4 report the detailed hyperparameter settings of encoder and decoder models for EIGAT, respectively. In the training, we set M=2 heads attention mechanism. The final dimensions of entity and relation embeddings are set to 200. The slop parameter α of LeakyReLU in Eq. (2) is set as 0.2 on all datasets. We use auxiliary relations from 2-hop neighborhood to aggregate more information about the neighborhoods. EI scores are initialized randomly in (0,1). We utilize a typical value for d = 0.85 (Mihalcea and Tarau, 2004; Florescu and Caragea, 2017).

5.5 Results and Analysis

Table 5 and Table 6 demonstrate the results of link prediction (significance level of 0.05). We can observe that: **(i)** The results clearly indicate that

Table 6: **Link prediction results** on *FB15K-237*. The results of baselines are directly taken from the original papers. The best scores are in **bold**.

	FB15K-237				
			Hits@N (%)		
	MR	**MRR**	**@1**	**@3**	**@10**
TransE (Bordes et al., 2013)	323	0.279	19.8	37.6	44.1
DistMult (Yang et al., 2015)	512	0.281	19.9	30.1	44.6
ComplEx (Trouillon et al., 2016)	546	0.278	19.4	29.7	45
ConvE (Dettmers et al., 2018)	245	0.312	22.5	34.1	49.7
ConvKB (Nguyen et al., 2018)	216	0.289	19.8	32.4	47.1
R-GCN (Schlichtkrull et al., 2018)	600	0.164	10	18.1	30
ATTH (Chami et al., 2020)	-	0.348	25.2	38.4	54.0
HAKE (Zhang et al., 2020b)	-	0.346	25.0	38.1	54.2
InteractE (Vashishth et al., 2020a)	172	0.354	26.3	-	53.5
ReInceptionE (Xie et al., 2020)	173	0.349	-	-	52.8
ParamE (Che et al., 2020)	-	0.399	31.0	43.8	57.6
RGHAT (Zhang et al., 2020c)	196	0.522	46.2	54.6	63.1
EIGAT (Ours)	**154**	**0.541**	**47.6**	**57.1**	**66.1**

EIGAT significantly and consistently outperforms all state-of-the-art baselines on most metrics in all benchmark datasets, which demonstrate the effectiveness of our proposed model. **(ii)** The advantages EIGAT compared to baselines on *NELL-995* seem to be smaller than others. It is because that rich global structural information in relative dense graphs, i.e., *Kinship*, and *FB15K-237*, leads to more effective entity importance estimation by global random walk methods, comparing with less global structural knowledge in relative sparse graphs, i.e., *NELL-995*. The results demonstrate *NELL-995* is more difficult than others for EIGAT to learn, but the comparable results also verify the effectiveness and robustness of our model on both scenarios.

5.6 Ablation Study

To analyze the behavior of global information in EIGAT, we compare EIGAT with EIGAT-Remove-global (i.e., removing global entity importance from EIGAT). The comparison results in Table 7 indicate that EIGAT achieves improvements against EIGAT-Remove-global on all metrics. In particular, on MR, EIGAT surpasses EIGAT-Remove-global by a large margin 56. The results demonstrate our model can successfully take account of global information in local attention to aggregate more effective entity representations.

5.7 Case Study

Table 8 gives examples of entity prediction results of EIGAT on the FB15k-237 testing set (predicting tail entities). This illustrates the efficacy of our proposed EIGAT. Given a head entity and a relation, the top predicted tail entities (and the true one) are depicted. Even if the true fact is not always at the best front, the predicted results can still reflect common-sense.

Table 7: **Ablation Study**. Link prediction results by different variants of our model on *FB15K-237*.

	FB15K-237				
			Hits@N (%)		
	MR	**MRR**	**@1**	**@3**	**@10**
EIGAT-*Remove-global*	210	0.518	46	54	62.6
EIGAT	**154**	**0.541**	**47.6**	**57.1**	**66.1**

Table 8: Example predictions on the FB15K-237 test set using EIGAT. **Bold** indicates the test triplet's true tail and italics other true tails present in the training set.

Head Entity	**Relation**	**Tail Entities**
X-Men	production companies	**Marvel Entertainment**, DC Comics, 20th Century Studios, American Zoetrope
United States of America	form of government	*presidential régime*, Democracy, **republic**, parliamentary monarchy, parliamentary system
Belgium	time zones	**Central European Time**, Atlantic Time Zone, Belgium

6 Conclusion and Future Work

In this paper, we propose to incorporate global information in local attention for knowledge representation learning and introduce a novel GAT-based model that incorporates global entity importance. In particular, we provide an attention-based global random walk approach to estimate entity importance. The experimental results of entity prediction demonstrate that our model can successfully take into account global information in local attention to improve knowledge representation learning. Interesting future work directions include generalizing EIGAT to other relational graphs (e.g. heterogeneous information network (HIN), user-item graph in recommendation system), and exploring an advanced variant of EIGAT in a semi-supervised learning scenario.

Acknowledgements

The authors would like to thank all anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China under Grant No. 61906159, U1836206, 61773361, 62072379.

References

Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. 2018. Watch your step: Learning node embeddings via graph attention. In *Proceedings of NeurIPS*, pages 9180–9190.

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of EMNLP*, pages 5185–5194.

Esma Balkir, Masha Naslidnyk, Dave Palfrey, and Arpit Mittal. 2019. Using pairwise occurrence information to improve knowledge graph completion on large-scale datasets. In *Proceedings of EMNLP*, pages 3591–3596.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of ACL*, pages 4387–4392.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NeurIPS*, pages 2787–2795.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:1–7.

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. 2019. Relational graph attention networks. In *arXiv preprint arXiv:1904.05811*.

Ines Chami, Adva Wolf, and Da-Cheng Juan. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of ACL*, pages 6901–6914.

Feihu Che, Dawei Zhang, Jianhua Tao, Mingyue Niu, and Bocheng Zhao. 2020. Parame: Regarding neural network parameters as relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, pages 2774–2781.

Thomas F Coleman and Jorge J Moré. 1983. Estimation of sparse jacobian matrices and graph coloring blems. *SIAM journal on Numerical Analysis*, 20(1):187–209.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, pages 1811–1818.

Takuma Ebisu and Ryutaro Ichise. 2019. Graph pattern entity ranking model for knowledge graph completion. In *Proceedings of NAACL*, pages 988–997.

Corina Florescu and Cornelia Caragea. 2017. Position-Rank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedding of ACL*, pages 1105–1115.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*.

Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. 2020. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of AAAI*.

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. 2020. Measuring and improving the use of graph information in graph neural networks. In *Proceedings of ICLR*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*, pages 687–696.

Yuchen Jing, Xiuzhen Zhang, Lifang Wu, Jinqiao Wang, Zemeng Feng, and Dan Wang. 2014. Recommendation on flickr by combining community user ratings and item importance. In *Proceedings of ICME*, pages 1–6.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018a. Graph classification using structural attention. In *Proceedings of SIGKDD*, pages 1–9.

John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. 2018b. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data*, 13(6):1–25.

John Boaz Lee, Ryan A. Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. 2019. Graph convolutional networks with motif-based attention. In *Proceedings of CIKM*, pages 499–508.

Xutao Li, Michael K. Ng, and Yunming Ye. 2012. HAR: hub, authority and relevance scores in multi-relational data for query search. In *Proceedings of ICDM*, pages 141–152.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, pages 2181–2187.

Qi Liu, Biao Xiang, Nicholas Jing Yuan, Enhong Chen, Hui Xiong, Yi Zheng, and Yu Yang. 2017. An influence propagation view of pagerank. *ACM TKDD*, 11(3):1–30.

Yiju Lu and Chengte Li. 2020. GCAN: Graph-aware co-attention networks for explainable fake news detection on social media. In *Proceedings of ACL*.

Christos Makris, Yannis Plegas, and Sofia Stamou. 2012. Web query disambiguation using pagerank. *Journal of the American Society for Information Science and Technology*, 63(8):1581–1592.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedding of EMNLP*, pages 404–411.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of ACL*.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL*, pages 327—333.

Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of NAACL*, pages 2180–2189.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, pages 809–816.

Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. 2019. Estimating node importance in knowledge graphs using graph neural networks. In *Proceedings of SIGKDD*, pages 596–606.

Wei Qian, Cong Fu, Yu Zhu, Deng Cai, and Xiaofei He. 2018. Translating embeddings for knowledge graph completion with relation attention mechanism. In *Proceedings of IJCAI*, pages 4286–4292.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of ACL*.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE TNN*, 20(1):61–80.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of ESWC*, pages 593–607.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *Proceedings of AAAI*, pages 1236–1242.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NeurIPS*, pages 926–934.

Kiran K. Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. In *arXiv:1803.03735v1*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of EMNLP*, pages 1499–1509.

Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of ICML*, pages 2071–2080.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha P. Talukdar. 2020a. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of AAAI*, pages 3009–3016.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020b. Composition-based multi-relational graph convolutional networks. In *Proceedings of ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR*.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019a. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of SIGKDD*, pages 950–958.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. 2019b. Heterogeneous graph attention network. In *Proceedings of WWW*, pages 2022–2032.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, pages 1112–1119.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. TwitterRank: Finding topic-sensitive influential twitterers. In *Proceedings of WSDM*, pages 261–270.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019a. Simplifying graph convolutional networks. In *Proceedings of ICML*.

Jun Wu, Jingrui He, and Jiejun Xu. 2019b. Demo-net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of SIGKDD*, pages 406–415.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG: A generative model for knowledge graph embedding. In *Proceedings of ACL*, pages 2316–2325.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2017. SSP: Semantic space projection for knowledge graph embedding with text descriptions. In *Proceedings of AAAI*, pages 3104–3110.

Zhiwen Xie, Guangyou Zhou, Jin Liu, and Jimmy Xiangji Huang. 2020. Reinceptione: Relation-aware inception network with joint local-global structural information for knowledge graph embedding. In *Proceedings of ACL*, pages 5929–5939.

Wenhan Xiong, Thien Hoang, , and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of EMNLP*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *Proceedings of ICLR*. arXiv preprint arXiv:1810.00826.

Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of ACL*, pages 1355–1362.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of ICLR*.

Yiding Yang, Xinchao Wang, Mingli Song, Junsong Yuan, and Dacheng Tao. 2019. Spagan: Shortest path graph attention network. In *Proceedings of IJCAI*, pages 4099–4105.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of UAI*.

Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. 2020a. Adaptive structural fingerprints for graph attention networks. In *Proceedings of ICLR*.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020b. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of AAAI*, pages 3065–3072.

Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. 2020c. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Proceedings of AAAI*, pages 9612–9619.

Yu Zhao, Anxiang Zhang, Ruobing Xie, Kang Liu, and Xiaojie Wang. 2020. Connecting embeddings for knowledge graph entity typing. In *Proceedings of ACL*.

Hao Zhou, Tom Yang, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of IJCAI*.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. Graph neural networks: A review of methods and applications. In *arXiv preprint arXiv:1812.08434*.

Exploiting Position Bias for Robust Aspect Sentiment Classification

Fang Ma*, **Chen Zhang***, **Dawei Song**†
School of Computer Science, Beijing Institute of Technology
Beijing, China
`{mfang,czhang,dwsong}@bit.edu.cn`

Abstract

Aspect sentiment classification (ASC) aims at determining sentiments expressed towards different aspects in a sentence. While state-of-the-art ASC models have achieved remarkable performance, they are recently shown to suffer from the issue of robustness. Particularly in two common scenarios: when domains of test and training data are different (out-of-domain scenario) or test data is adversarially perturbed (adversarial scenario), ASC models may attend to irrelevant words and neglect opinion expressions that truly describe diverse aspects. To tackle the challenge, in this paper, we hypothesize that position bias (i.e., the words closer to a concerning aspect would carry a higher degree of importance) is crucial for building more robust ASC models by reducing the probability of mis-attending. Accordingly, we propose two mechanisms for capturing position bias, namely position-biased weight and position-biased dropout, which can be flexibly injected into existing models to enhance representations for classification. Experiments conducted on out-of-domain and adversarial datasets demonstrate that our proposed approaches largely improve the robustness and effectiveness of current models.[1]

1 Introduction

Aspect sentiment classification (ASC) is an important sub-task of sentiment classification. It aims to identify the sentiment polarity (i.e., negative, neutral, or positive) of a specified aspect in a sentence. Take "*Great food but the service was bad.*" as an example. For aspects *food* and *service*, their corresponding sentiment polarities are *positive* and *negative*, respectively.

Scenario	**Example**	**Pred./Lb.**
I.D.	Great food but the service was bad !	neg./neg.
O.O.D.	The battery has never worked well .	pos./neg.
Adv.	Awful food but the service was great !	neg./pos.

Figure 1: An illustration of how an ASC model IAN (Ma et al., 2017) might fail. Gradient saliency maps (Simonyan et al., 2014) with respect to the embedding of each word in I.D., O.O.D., and Adv. scenarios, along with the model predictions (Pred.) and corresponding ground truth labels (Lb.), are provided. Underlined words are aspects.

A challenge in ASC is how to model semantic relations between aspect terms and their contexts, which requires an ASC model to be only sensitive to the sentiment words actually depicting the target aspect terms. Although previous ASC models (Tang et al., 2016b; Li et al., 2018; Zhang et al., 2019a; Xu et al., 2019; Wang et al., 2020; Tang et al., 2020) have achieved promising results by modeling complex interactions between aspects and contexts, these models have recently been shown to suffer from the lack of robustness (Xing et al., 2020). The issue is particularly severe in two scenarios: 1) out-of-domain (O.O.D.) scenario: ASC models that perform well on training data often fail to generalize to test data in another domain; 2) adversarial (Adv.) scenario: ASC models can be easily fooled by small adversarially perturbed inputs, e.g. synonymous word substituted ones. To our best knowledge, none of current ASC models have been targeted at alleviating the robustness issue in above-mentioned two scenarios.

To fill this gap, inspired by a recent finding that highlighting words close to a target aspect (termed as *position bias*) would boost in-domain (I.D.) effectiveness of a model (Zhang et al., 2019b), we hypothesize that such position bias is also crucial for a robust ASC model in O.O.D. and Adv. sce-

---

[*] Fang Ma and Chen Zhang contribute equally to this work. The order is determined alphabetically.

[†] Dawei Song is the corresponding author.

[1] The code and preprocessed data are available at `https://github.com/BD-MF/POS4ASC`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1352–1358
August 1–6, 2021. ©2021 Association for Computational Linguistics

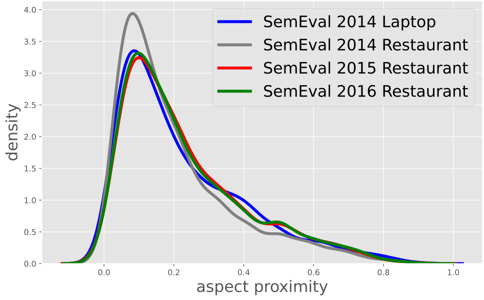

Figure 2: Aspect proximity distributions of different benchmarks, visualized with the kernel density estimation.

narios. Figure 1 shows an illustrative example of an ASC model that fails in the two scenarios due to mis-attending. In contrast, with position bias, a model tends to focus more on words nearer to the aspect, thus reducing the probability of mis-attending. Concretely, we propose two mechanisms: position-biased weight and position-biased dropout. The former assigns an inductive weight to each word according to its position proximity to the aspect. The latter gives each word a probability of being reserved (or dropped out) according to its proximity relation to the aspect. In doing so, position-biased weight degrades the significance of words that are not close enough to the aspect, while position-biased dropout will drop those likely irrelevant words at high probabilities.

Essentially, position bias is quantitatively evidenced in commonly used benchmarks. With annotated aspect-opinion pairs offered by Fan et al. (2019), we can calculate position proximity between any pair of aspect and opinion (in short, aspect proximity) in a sentence. The aspect proximity is computed by dividing the relative distance between a pair of aspect and opinion by the length of the corresponding sentence. Therefore, we can plot aspect proximity distributions of these benchmarks with kernel density estimation, as shown in Figure 2. These distributions indicate the aspect proximity is small at a high probability, thereby position bias is a reasonable inductive bias.

Extensive experiments are conducted on SemEval and ARTS datasets (Xing et al., 2020). The results show that incorporating the proposed position bias mechanisms would lead to more robust ASC models in both out-of-domain and adversarial scenarios. Furthermore, in terms of flexibility, the proposed methods can be easily adapted by subsequent models.

2 Capturing Position Bias

This section describes the proposed position-biased weight and position-biased dropout for capturing position bias. Formally, an n-word sentence containing a target m-word aspect term is formulated as $S = \{w_0, w_1, \ldots, w_\gamma, w_{\gamma+1}, \ldots, w_{n-1}\}$, where γ denotes the start index of the aspect term. By resorting to either a pre-trained word embedding (Bengio et al., 2003) or a pre-trained language model (Devlin et al., 2019), we can represent the sentence as $V = \{e_0, e_1, \ldots, e_\gamma, e_{\gamma+1}, \ldots, e_{n-1}\}$. We can then use the position-biased weight and dropout to refine V and generate an enhanced representation, denoted as $E = \{h_0, h_1, ...h_\gamma, h_{\gamma+1}, ..., h_{n-1}\}$. E can then be incorporated into the model, i.e., any further structures for the model will be built upon E, instead of V, to predict sentiment polarities associated with diverse aspects.

Position-biased Weight Generally, the sentiment polarity of an aspect term is determined by its context, which are the words around the aspect term (Zhang et al., 2019b). Thus we can leverage relative position information to calculate weights of context words, with the aim to degrade the significance of those words that are far away from the aspect. Position-biased weight, denoted as $p_i \in (0, 1)$, is computed as:

$$p_i = \begin{cases} 1 - \frac{\gamma - i}{n-m} & 0 \leq i < \gamma \\ \frac{1}{n-m} & \gamma \leq i < \gamma + m \\ 1 - \frac{i-\gamma-m+1}{n-m} & \gamma + m \leq i < n \end{cases} \quad (1)$$

Then we compute h_i as: $h_i = p_i \cdot e_i$.

Position-biased Dropout Dropout (Srivastava et al., 2014; Sennrich et al., 2016) randomly sets elements in a feature vector to zeros. The word-level dropout can model semantic and syntactic compositionality and reduce input redundancy (Iyyer et al., 2015). Motivated by this idea, we give each word a probability of being reserved according to its position proximity to the aspect. The aim is to preserve those words that are close enough to the aspect and drop the rest. The probability that the i-th word will be preserved can be computed as:

$$z_i \sim \text{Bernoulli}(p_i) \quad (2)$$

where $\text{Bernoulli}(p_i)$ denotes that z_i equals to 1 with p_i and equals to 0 with $1 - p_i$. The i-th word is dropped out if z_i is 0. Likewise, h_i can be attained by multiplying z_i and e_i.

3 Experiments

Datasets To empirically evaluate a model's O.O.D. robustness, we conduct experiments on two domains from SemEval 2014 (Pontiki et al., 2014) dataset (SEMEVAL): one is laptop (SEMEVAL-LAP) and the other is restaurant (SEMEVAL-REST). For Adv. robustness, we experiment with the Aspect Robustness Test dataset (ARTS) (Xing et al., 2020), which is derived from the SemEval 2014 dataset. Instances in ARTS are generated with three adversarial strategies. These strategies enrich the test set from 638 to 1,877 for the laptop domain (ARTS-LAP), and from 1,120 to 3,530 for the restaurant domain (ARTS-REST). Note that each domain from SEMEVAL consists of separate training and test sets, while each domain from ARTS only contains a test set. Since SEMEVAL dataset does not come with development sets, 150 instances from the training set in each domain are randomly selected to form the development set. Table 1 shows the statistics of the datasets.

Dataset		# pos.	# neu.	# neg.
SEMEVAL-LAP	train	930	433	800
	test	341	169	128
	dev	57	27	66
SEMEVAL-REST	train	2,094	579	779
	test	728	196	196
	dev	70	54	26
ARTS-LAP	test	883	407	587
ARTS-REST	test	1,953	473	1,104

Table 1: Statistics of datasets.

Target Models We conduct experiments on a wide range of existing models for a comprehensive study on whether position bias is beneficial. Specifically, we examine these models' performance before and after injecting the position bias, in terms of position-biased weight (`pos-wt`) and dropout (`pos-dp`) individually.

The target models include: (a) **LSTM** (Tang et al., 2016a) uses the last hidden state vector of the LSTM to predict sentiment. (b) **LSTM-Attn** (Wang et al., 2016) applies an attention-based LSTM on the concatenation of the aspect and word embeddings. (c) **IAN** (Ma et al., 2017) interactively learns attentions between context words and aspect terms. (d) **MemNet** (Tang et al., 2016b) applies attention multiple times on word memories, and the output of the last attention is used for prediction. While the original work utilizes word embeddings as memories, we instead choose to add a layer of bidirectional LSTM upon embeddings for more abstractive memories. (e) **AOA** (Huang et al., 2018) introduces an attention-over-attention based network to model interaction between aspects and contexts. (f) **RoBERTa** (Dai et al., 2021) is a strong baseline with an MLP built upon the pooled feature induced with RoBERTa (Liu et al., 2019).

Implementation Details In all our experiments, the 300-dimensional GloVe (Pennington et al., 2014) is leveraged to initialize the input embedding. All parameters of models are initialized with uniform distributions. During all experiments, a bidirectional LSTM is adopted if necessary instead of a unidirectional one. If a model takes advantage of the attention mechanism, then dot product based attention is employed. In case a model has hidden states, the dimensionality of hidden states is set to 300. The batch size is 64. We use Adam (Kingma and Ba, 2015) as the optimizer with a learning rate of 10^{-3}. The coefficient of L2 regularization is 10^{-5}. For experiments with RoBERTa as the input embedding (Liu et al., 2019), things may change. The dimensionality of hidden states is 768. The learning rate is 10^{-5}, while the regularization is rather removed.

Evaluation Metrics For O.O.D., models are trained separately on one domain and evaluated on another. For Adv., models are trained on the SEMEVAL dataset and tested on the ARTS counterpart. For every test, a model is trained on the I.D. training set, selected on the I.D. development set, and tested on the O.O.D. or Adv. test set. The experimental results are obtained by averaging 5 runs with random initialization, and we adopt Accuracy and macro-averaged F1 scores as evaluation metrics.

In-domain Generalization Results Table 3 shows the I.D. performance of the LSTM on both laptop and restaurant domains, which exhibits incorporating position bias does not harm, if this is the case, a model's generalization on I.D. test sets that much. On the contrary, position bias, especially with position-biased weight, can boost I.D. performance.

Model	**LAP**				**REST**			
	O.O.D.		**Adv.**		**O.O.D.**		**Adv.**	
	Acc.	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
LSTM	71.02	52.15	49.49	43.91	60.60	53.25	53.34	41.99
w/ `pos-dp`	71.48↑**0.46**	50.98↓**1.17**	50.74↑**1.25**	44.38↑**0.47**	63.39↑**2.79**	58.57↑**5.32**	53.57↑**0.23**	42.11↑**0.12**
w/ `pos-wt`	72.96↑**1.94**	55.88↑**3.73**	55.50↑**6.01**	50.03↑**6.12**	66.33↑**5.73**	60.21↑**6.96**	59.03↑**5.69**	48.20↑**6.21**
LSTM-Attn	71.61	53.61	51.33	46.11	62.85	54.97	58.45	49.65
w/ `pos-dp`	71.34↓**0.27**	52.49↓**1.12**	53.76↑**2.43**	48.47↑**2.36**	65.24↑**2.39**	59.07↑**4.10**	58.64↑**0.19**	47.22↓**2.43**
w/ `pos-wt`	72.84↑**1.23**	56.18↑**2.57**	58.53↑**7.20**	53.54↑**7.43**	68.90↑**6.05**	64.48↑**9.51**	64.80↑**6.35**	55.34↑**5.69**
IAN	72.09	54.44	52.91	47.54	63.82	55.20	57.75	48.12
w/ `pos-dp`	70.95↓**1.14**	51.63↓**3.08**	52.04↓**0.87**	45.87↓**1.67**	63.57↓**0.25**	56.81↑**1.61**	56.89↓**0.86**	46.90↓**1.22**
w/ `pos-wt`	72.86↑**0.77**	54.88↑**0.44**	56.03↑**3.12**	50.30↑**2.76**	62.45↓**1.37**	55.95↑**0.75**	63.49↑**5.74**	54.04↑**5.92**
MemNet	70.66	52.07	52.00	46.50	57.84	51.15	55.30	46.67
w/ `pos-dp`	69.93↓**0.73**	53.37↑**1.30**	53.54↑**1.54**	47.93↑**1.43**	61.94↑**4.10**	54.49↑**3.34**	57.31↑**2.01**	45.23↓**1.44**
w/ `pos-wt`	70.67↑**0.01**	54.14↑**2.07**	56.04↑**4.04**	49.64↑**3.14**	61.35↑**3.51**	54.85↑**3.70**	61.10↑**5.80**	51.49↑**4.82**
AOA	71.63	52.65	52.16	46.78	63.73	57.00	58.19	49.02
w/ `pos-dp`	72.30↓**0.67**	53.73↑**1.08**	53.56↑**1.40**	48.18↑**1.40**	65.33↑**1.60**	58.31↑**1.31**	56.24↓**1.95**	45.63↓**3.39**
w/ `pos-wt`	72.61↑**0.98**	56.54↑**3.89**	59.07↑**6.91**	54.92↑**8.14**	66.87↑**3.14**	62.02↑**5.02**	64.35↑**6.16**	54.62↑**5.60**
RoBERTa	83.16	72.99	73.57	69.26	77.62	71.34	79.08	71.79
w/ `pos-dp`	81.98↓**1.18**	70.81↓**2.18**	69.98↓**3.59**	65.35↓**3.91**	75.61↓**2.01**	68.00↓**3.34**	77.81↓**1.27**	69.37↓**2.42**
w/ `pos-wt`	83.43↑**0.27**	74.08↑**1.09**	75.72↑**2.15**	72.09↑**2.83**	79.40↑**1.78**	74.44↑**3.10**	79.47↑**0.39**	73.10↑**1.31**

Table 2: Robustness results (%). O.O.D. on LAP or REST denotes a model is trained in current domain (LAP or REST) and tested on another (REST or LAP). Adv. denotes a model is trained in a domain and tested on its ARTS counterpart. Furthermore, w/ `pos-dp` means a model with position-biased dropout. w/ `pos-wt` means a model with position-biased weight. The small number next to each performance score indicates either performance improvement (↑) or drop (↓) compared with the original model without using position bias, and those highlighted in red are the best-performing ones among two variants.

Model	**LAP I.D.**		**REST I.D.**	
	Acc.	**F1**	**Acc.**	**F1**
LSTM	67.15	60.57	74.57	62.14
w/ `pos-dp`	67.34	60.27	74.23	61.55
w/ `pos-wt`	68.78	62.42	76.34	64.85

Table 3: I.D. results (%) of LSTM on LAP and REST.

Robustness Results The robustness results are shown in Table 2. We can see that performance of LSTM drops drastically, compared to I.D. performance, on O.O.D. and Adv. test sets, indicating the importance of studying the robustness issue. Our proposed two position bias mechanisms improve the target models' O.O.D. and Adv. performance in most cases. With position-biased dropout, F1 scores of models are improved by up to 5.32 pp on O.O.D. test sets, and 2.36 pp on Adv. test sets, though the efficacy of the position-biased dropout seems not stable across different target models and settings. In contrast, the impact of position-biased weight is much more prominent. With position-biased weight, Accuracy scores of models can be enhanced by up to 6.05 pp and 7.20 pp on O.O.D. and Adv. test sets, respectively. Further, F1 scores of models are improved by up to 9.51 pp and 8.14 pp on O.O.D. and Adv. test sets.

A highlight is that experimental results with RoBERTa as well exhibit the benefit of position bias, yet with caveats. Although pre-trained language models like RoBERTa are subject to positional encodings, such absolute position information is not enough to model relative position relations between aspect terms and contexts. Therefore, position bias matters during fine-tuning pre-trained language models for robust ASC performance. However, we observe that position-biased dropout is not an appropriate choice for pre-trained language models.

Case Study To understand the effect of position bias, we conduct a case study on the two robustness scenarios, as shown in Table 4. Specifically, we visualize the attention scores separately offered by an ASC model LSTM-Attn with and without position-biased weight method and trained on SEMEVAL-REST.

We can observe that before applying position bias, the model attends irrelevant words and fails in both scenarios. Specifically, in both cases, the model mis-attends to irrelevant opinion expressions. After injecting position bias, the attention scores become more accurate and the model attends to the correct opinion spans.

w/ `pos-wt`	Example
✘	The price is reasonable although the quality is poor .
✔	The price is reasonable although the quality is poor .
✘	Awful food but the service was great !
✔	Awful food but the service was great !

Table 4: **Case study.** The underlined words are aspects. The top two rows are O.O.D. examples, while the bottom two are Adv. examples. ✘ and ✔ refers to without and with `pos-wt` respectively.

4 Related Work

Fine-grained Sentiment Analysis ASC falls in the broad scope of fine-grained sentiment analysis. While ASC is basically formulated as determining sentiment polarity of a given aspect in a sentence (Tang et al., 2016b,a; Wang et al., 2016; Chen et al., 2017; Huang et al., 2018; Li et al., 2018; Xu et al., 2019; Zhang et al., 2019b,a; Wang et al., 2020; Tang et al., 2020), there is an emergent trend that treating fine-grained sentiment as an opinion triplet extraction task (Peng et al., 2020; Zhang et al., 2020; Wu et al., 2020). Recently, the robustness of ASC models becomes a critical issue that urges researchers to pay more attention on improving the robustness of ASC models (Xing et al., 2020). Our work is the first work to enhance the universal robustness of ASC models by capturing position bias. On another note, we believe opinion triplet extraction is exposed to the similar robustness issue, which should be explored in the near future.

Robustness in NLP Broadly, there are two kinds of robustness in NLP, i.e., O.O.D. and Adv. robustness. O.O.D. robustness in NLP has attracted extensive attention in recent work (Ng et al., 2020; Hendrycks et al., 2020; Xie et al., 2020). In terms of O.O.D. robustness, they often use the cross-domain setting to evaluate models (Benson and Ecker, 2020). Previous work mainly focuses on how to minimize the domain discrepancy and how to improve the feature adaptability of models (Rietzler et al., 2020; Ye et al., 2020). On the other hand, adversarial learning becomes the main method used to improve Adv. robustness of models (Xing et al., 2020). Prior methods consider using semantic operations, such as synonym replacement, random insertion, random swap, and random deletion to augment data (Wei and Zou, 2019). Other methods involve adding extra text (Wallace et al., 2019) and replacing sentences with semantically similar sentences (Ribeiro et al., 2018). Our work goes beyond the two forms of robustness and aims to achieve universal robustness for ASC with position bias.

5 Conclusion and Future Work

In this work, we find that state-of-the-art ASC models suffer from the issue of robustness, particularly in two scenarios: i) out-of-domain scenario, and ii) adversarial scenario. To address the issue, we propose a simple yet effective inductive bias that should be incorporated, that is, position bias. We proposed two mechanisms to capture position bias, namely position-biased weight and position-biased dropout. They are injected into existing models to enhance the representation. Extensive experiments demonstrate that the proposed methods can largely improve the models' robustness. The results verify our hypothesis that position bias is beneficial for building more robust ASC models.

The work shall be improved in the following two facets: i) Since the approach of incorporating position bias is straightforward yet naive, especially for pre-trained language models, it is meaningful to consider a nicely designed architecture to inject position bias in a more elegant manner. ii) It has been shown that position bias for ASC is highly correlated with the syntactic structure of the sentence. Hence, syntax can likewise be explored to enhance the robustness of ASC models.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (grant No. 2018YFC0831704) and the Natural Science Foundation of China (grant No. U1636203).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Vitus Benson and Alexander Ecker. 2020. Assessing out-of-domain generalization for robust building damage detection. *arXiv preprint arXiv:2011.10328*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Junqi Dai, Hang Yan, Tianxiang Sun, Pengfei Liu, and Xipeng Qiu. 2021. Does syntax matter? a strong baseline for aspect-based sentiment analysis with roberta. *arXiv preprint arXiv:2104.04986*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Zhifang Fan, Zhen Wu, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2019. Target-oriented opinion words extraction with target-fused neural sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2509–2518.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751.

Binxuan Huang, Yanglan Ou, and Kathleen M Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 197–206.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283.

Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2020. Knowing what, how and why: A near complete solution for aspect-based sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8600–8607.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *SemEval 2014*, page 27.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2020. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4933–4941.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6578–6588.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389.

Zhen Wu, Chengcan Ying, Fei Zhao, Zhifang Fan, Xinyu Dai, and Rui Xia. 2020. Grid tagging scheme for end-to-end fine-grained opinion extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2576–2585.

Sang Michael Xie, Ananya Kumar, Robbie Jones, Fereshte Khani, Tengyu Ma, and Percy Liang. 2020. In-n-out: Pre-training and self-training using auxiliary information for out-of-distribution robustness. *arXiv preprint arXiv:2012.04550*.

Xiaoyu Xing, Zhijing Jin, Di Jin, Bingning Wang, Qi Zhang, and Xuan-Jing Huang. 2020. Tasty burgers, soggy fries: Probing aspect robustness in aspect-based sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3594–3605.

Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335.

Hai Ye, Qingyu Tan, Ruidan He, Juntao Li, Hwee Tou Ng, and Lidong Bing. 2020. Feature adaptation of pre-trained language models across languages and domains with robust self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7386–7399.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019a. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4560–4570.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019b. Syntax-aware aspect-level sentiment classification with proximity-weighted convolution network. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1145–1148.

Chen Zhang, Qiuchi Li, Dawei Song, and Benyou Wang. 2020. A multi-task learning framework for opinion triplet extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 819–828.

MRN: A Locally and Globally Mention-Based Reasoning Network for Document-Level Relation Extraction

Jingye Li[1], **Kang Xu**[1], **Fei Li**[1,*] **Hao Fei**[1], **Yafeng Ren**[2], **Donghong Ji**[1]
[1]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
[2]Laboratory of Language and Artificial Intelligence, Guangdong University of Foreign Studies, China
`{theodorelee,xukang0531,lifei_csnlp,hao.fei,renyafeng,dhji}@whu.edu.cn`

Abstract

Document-level relation extraction aims to detect the relations within one document, which is challenging since it requires complex reasoning using mentions, entities, local and global contexts. Few previous studies have distinguished local and global reasoning explicitly, which may be problematic because they play different roles in intra- and inter-sentence relations. Moreover, the interactions between local and global contexts should be considered since they could help relation reasoning based on our observation. In this paper, we propose a novel mention-based reasoning (MRN) module based on explicitly and collaboratively local and global reasoning. Based on MRN, we design a co-predictor module to predict entity relations based on local and global entity and relation representations jointly. We evaluate our MRN model on three widely-used benchmark datasets, namely DocRED, CDR, and GDA. Experimental results show that our model outperforms previous state-of-the-art models by a large margin.

1 Introduction

Relation extraction (RE), identifying the semantic relations among target entities in the text, has long been a fundamental task in the natural language processing (NLP) community (Zeng et al., 2014; Xu et al., 2015). Prior efforts largely focus on sentence-level RE (Lin et al., 2016; Zhang et al., 2018). However, recent studies reveal that a large number of relations can actually be expressed through multiple sentences, which necessitates document-level RE (Yao et al., 2019). Compared with sentence-level RE, the entities for document-level relations may be mentioned in multiple sentences across a document. Therefore, document-level RE requires capturing the complex interactions between all entities in the entire document (Nan et al., 2020; Zeng et al., 2020; Wang et al., 2020).

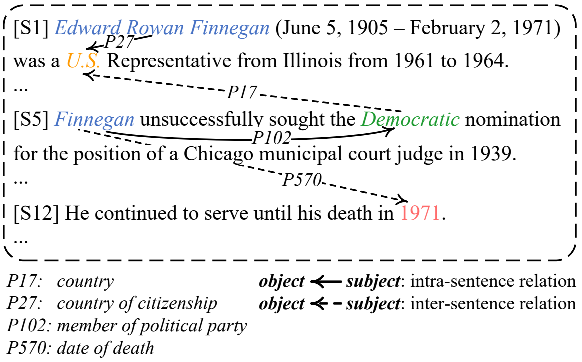
[S1] *Edward Rowan Finnegan* (June 5, 1905 – February 2, 1971) was a *U.S.* Representative from Illinois from 1961 to 1964.
...
[S5] *Finnegan* unsuccessfully sought the *Democratic* nomination for the position of a Chicago municipal court judge in 1939.
...
[S12] He continued to serve until his death in 1971.
...

Figure 1: An example of document-level RE from the DocRED dataset. We use the same color to denote the mentions of the same entity.

It is well known that local and global contexts are two key performance enhancers for the task. Intuitively, the former can benefit the identification of nearer (e.g., intra-sentence) relations, while the latter is more useful for distant (e.g., inter-sentence) relations. For document-level RE, such context information is closely related to the ubiquitous mentions in a document, so **mention-based reasoning** with different context granularities is highly important for the task. However, most previous studies have not distinguished local and global reasoning explicitly (Peng et al., 2017; Christopoulou et al., 2019; Sahu et al., 2019; Nan et al., 2020; Zhou et al., 2020; Zeng et al., 2020; Li et al., 2020; Zhang et al., 2020). Recently Wang et al. (2020) investigate local and global contexts for document-level RE by performing global and local reasoning consecutively. However, their pipeline method can be problematic because it ignores the interactions and communications of local and global contexts,[1] which limits the performance of the task.

---

[*] Corresponding author.

[1] From Figure 1, it is shown that the intra-sentence relations $P27$ and $P102$ can help to identify the inter-sentence relation $P17$, since '*U.S.*' and '*Democratic*' are linked through '*Finnegan*'.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1359–1370
August 1–6, 2021.

In this paper, we aim to address the above issues for improving document-level RE by presenting a novel **M**ention-based **R**easoning **N**etwork (**MRN**). As shown in Figure 2, MRN consists of several innovative modules for modeling the relation reasoning locally and globally, including (1) a two-dimensional windowed convolution for capturing the local mention-to-mention interactions between the subject and object arguments of relations, and (2) a co-attention module for capturing the global interaction between each pair of entity mentions. Note that the above two modules also provide mutual information to each other, in order to capture the interactions between local and global contexts.

Moreover, different from previous work that expresses entities with just one kind of representation (Peng et al., 2017; Christopoulou et al., 2019; Sahu et al., 2019; Nan et al., 2020; Zhou et al., 2020; Zeng et al., 2020; Li et al., 2020; Zhang et al., 2020), our method distinguishes mentions into subjects and objects, and generates entity representations from both local and global mention-based reasoning. Specifically, we design a novel module, called **co-predictor**, to utilize both local and global entity representations for jointly reasoning the relations between close and distant entities.

We conduct extensive experiments on three widely-used benchmarks, including DocRED (Yao et al., 2019), CDR (Li et al., 2016) and GDA (Wu et al., 2019). The results show that our MRN model outperforms the current best model by a large margin, demonstrating its advances. In summary, we make the following contributions:

- We propose a mention-based reasoning network (MRN), to distinguish the impacts of close and distant entity mentions in relation extraction and meanwhile consider the interactions between local and global contexts, which we call locally and globally mention-based reasoning.
- We also propose a co-predictor to work in concert with the mention reasoning block, and predict the relation of a pair of entities using local and global features simultaneously.
- Our model achieves the state-of-the-art performances on three benchmark datasets for document-level RE. We also conduct extensive analyses of our model to better understand its working mechanism.[2]

2 Related Work

Relation extraction (RE), including sentence-level RE and document-level RE, plays a crucial role in a wide variety of knowledge-based applications, such as question answering (Hixon et al., 2015), dialogue generation (He et al., 2017), etc. Recent studies largely focus on sentence-level RE by various neural network methods, such as CNN (Zeng et al., 2014; dos Santos et al., 2015), BiLSTM (Zhang et al., 2015; Cai et al., 2016), attention mechanism (Wang et al., 2016; Lin et al., 2016), and neural graph models (Zhang et al., 2018; Zhu et al., 2019). However, in practice, many relational facts need to be inferred across multiple sentences in a document, so researchers have shown a growing interest in document-level RE.

Compared with sentence-level RE, document-level RE needs to consider the complicated interactions between entities across multiple sentences. With this in mind, researchers begin to use graph neural networks to reason intra- and inter-sentence relations and make certain progress in extracting inter-sentence relations with document-level graph convolutional neural network (Peng et al., 2017; Velickovic et al., 2018; Christopoulou et al., 2019). Sahu et al. (2019). For example, Zhou et al. (2020) use entities as nodes and the context between entity pairs as edges to construct graphs. Nan et al. (2020) treat the graph as a latent structure and perform relational reasoning. However, most existing approaches only use entity-level information and ignore mention-level information.

Some studies also take mentions into account by adding mention nodes in the graph. For instance, Christopoulou et al. (2019) put mentions and entities in the same graph. Li et al. (2020) utilize a dual-tier heterogeneous graph to propagate relational information among entity mentions, and then summarize them into corresponding entities. More recently, Wang et al. (2020) use multi-head attention to aggregate multiple mentions of specific entities. Different from the above methods, Zeng et al. (2020) propose a graph aggregation and inference network that includes two graphs, one for capturing complex interactions among different mentions, and the other for integrating mention-level information of the same entities. Although these methods introduce mention-level nodes or graphs, none of them consider local mention-based contextual information or mention-level relative distances, which are all considered in our model.

---

[2] Codes are publicly available at `https://github.com/ljynlp/MRN`

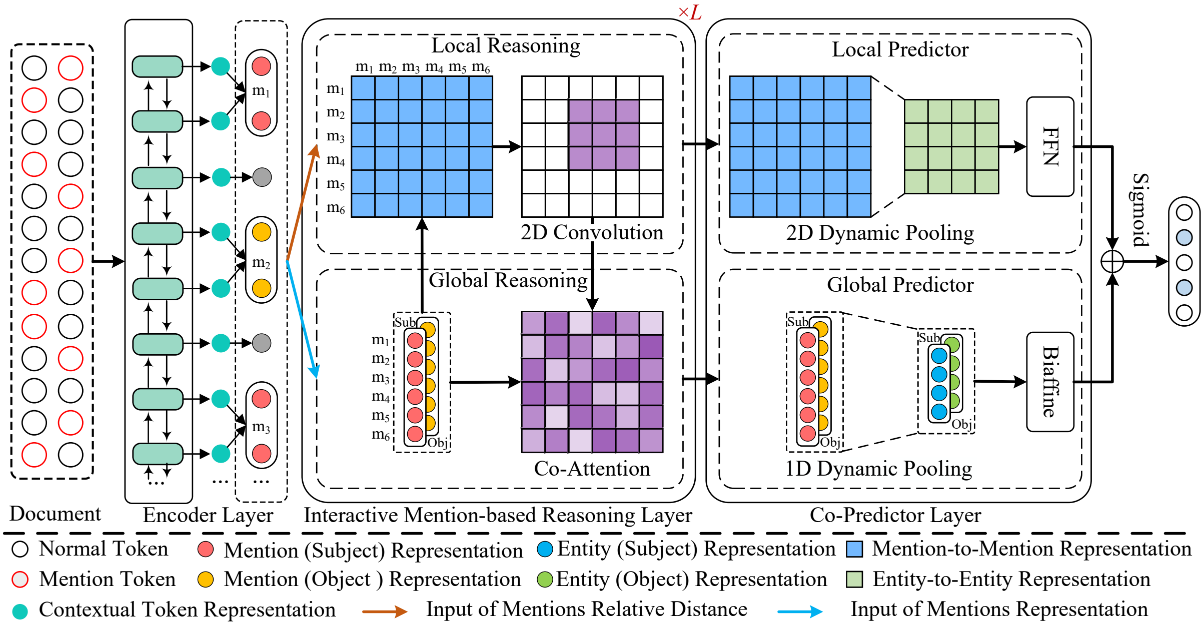

Figure 2: The overall architecture of our `MRN`, including three key components: encoder (§ 3.1), interactive mention-based reasoning (§ 3.2) and co-predictor (§ 3.3).

3 Methodology

Task Formulation Given an input document consisting of N tokens $\mathcal{D} = \{w_i\}_{i=1}^{N}$ and P entities $\mathcal{E} = \{e_i\}_{i=1}^{P}$, the task aims to extract a subset of relations from $\mathcal{R}$ between the entity pairs (e_i^s, e_j^o), where $\mathcal{R}$ is a pre-defined relation type set, e_i^s and e_j^o are identified as subject and object entities. An entity e_i can appear multiple times in the document via K_{e_i} mentions $\mathcal{M}_{e_i} = \{m_j\}_{j=1}^{K_{e_i}}$.

Framework As illustrated in Figure 2, the overall architecture consists of three tiers. An encoder first yields contextual representations, and then the mention-based reasoning block (in multiple layers) performs local-level feature extraction with a two-dimensional (2D) convolution, and global-level feature retrieval with a co-attention module. Afterward, a co-predictor layer that contains a 1D and 2D dynamic pooling aggregates the entity-level features from the mention-based reasoning block. Finally, a multi-layer perceptron and a biaffine classifier are leveraged for jointly reasoning the relations between subject and object entities.

3.1 Encoding Layer

We first map each word w_i into a vector, and concatenate it with its corresponding entity type t_i:[3]

$$\mathbf{x}_i = [\mathbf{x}_i^w; \mathbf{x}_i^t] \, . \tag{1}$$

Then, we adopt BiLSTM to encode the vectorial word representations into contextualized word representations:

$$\mathbf{h}_i = \text{BiLSTM}(\mathbf{x}_i) \, , \tag{2}$$

where $\mathbf{h}_i$ is the token hidden representation. Note that we also can use the BERT (Devlin et al., 2019) as an alternative to improve performances. Based on $\mathbf{h}_i$, we can obtain the mention representation:

$$\mathbf{m}_i = \text{Max}\{\mathbf{h}_j\}_{j=a_i}^{b_i} \, , \tag{3}$$

where a_i and b_i are the start and end positions of the i-th mention, respectively.

3.2 Interactive Mention-based Reasoning Layer

As we argued earlier, local and global context information is closely related to the ubiquitous mentions in a document. We thus propose a mention-based module for multi-hop reasoning among the relationships of all the mentions. Considering that there exist overlapping relations where multiple relations share the same mention, we distinguish mentions into subjects and objects by their directions. Moreover, near neighbors are more informative than distant ones for determining relations. To this end,

---

[3] t_i denotes the type of entity mention that contains this word (e.g. if an entity type is `Person`, its mention word type is also `Person`).

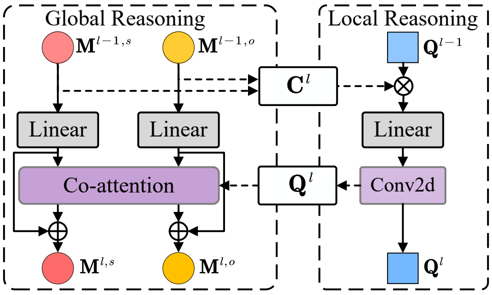

Figure 3: The interactive mention-based reasoning block. ⊕ and ⊗ represent the addition and concatenation operations.

we adopt a two-dimensional (2D) windowed convolution to capture local interactions between close subject and object mentions.

As shown in Figure 3, we perform multi-hop reasoning by stacking multiple layers (i.e., L) of our interactive mention-based reasoning blocks. The inputs of the l-th layer block include: (1) a set of subject mention representations $\mathbf{M}^{l-1,s} = \{\mathbf{m}_i^{l-1,s}\}_{i=1}^{T}$, (2) a set of object mention representations $\mathbf{M}^{l-1,o} = \{\mathbf{m}_i^{l-1,o}\}_{i=1}^{T}$, and (3) a set of mention relation representations $\mathbf{Q}^{l-1}$, where T is the number of mentions and $\mathbf{Q}_{ij}^{l-1}$ denotes the type of implicit relation between the subject i and object j. For initialization, $\mathbf{m}_i^{0,s} = \mathbf{m}_i^{0,o} = \mathbf{m}_i$, and $\mathbf{Q}^0$ denotes relative distance embeddings of mentions (Zeng et al., 2014).

Local Reasoning To model the interactions between near subjects and objects, we first build a relationship tensor $\mathbf{C}^l$ from $\mathbf{M}^{l-1,s}$ and $\mathbf{M}^{l-1,o}$, in which $\mathbf{C}_{ij}^{l} = [\mathbf{m}_i^{l-1,s}; \mathbf{m}_j^{l-1,o}]$. Next, we concatenate $\mathbf{C}^l$ and $\mathbf{Q}^{l-1}$, and adopt a Feedforward Network (FFN) to reduce their dimensions. We then employ a 2D convolution to capture local contextual interactions, which can be regarded as extracting subgraph representations from a fully-connected bipartite graph containing subject and object nodes. The overall process can be formulated as:

$$\mathbf{Q}^{l} = \sigma(\text{Conv2d}(\text{FFN}([\mathbf{Q}^{l-1}; \mathbf{C}^{l}]))) , \qquad (4)$$

where σ is a LeakyReLU activation function.

Global Reasoning We introduce a co-attention mechanism to compute attention coefficients that indicate the importance of subjects to objects and vice versa, so that the interaction between each pair of mentions can be considered. Inspired by the success of graph attention network (GAT) (Velickovic et al., 2018), we apply two learnable linear transformations to transform subject and object mention representations $\mathbf{M}^{l-1,s}$ and $\mathbf{M}^{l-1,o}$ into higher-level features. Then, we leverage the mention relation representation $\mathbf{Q}^l$ to calculate the coefficients and inject them into the co-attention process:

$$\hat{\mathbf{m}}_i^{l,\phi} = \sum_{j \in M^{l,\psi}} \alpha_{ij}^{l,\psi} \mathbf{W}^{l,\psi} \mathbf{m}_j^{l-1,\psi} , \qquad (5)$$

where $\mathbf{W}^{l,\psi}$ is a learnable parameter matrix, and $\alpha_{ij}^{l,\psi} = \text{Softmax}(\text{FFN}(\mathbf{Q}^l))$. Here $\phi, \psi \in \{s, o\}$. Afterward, the mention representation of the next layer $\mathbf{m}_i^{l,\phi}$ is generated by adding the residual of the last layer $\mathbf{m}_i^{l-1,\phi}$ and the non-linear transformation of the co-attention output $\hat{\mathbf{m}}_i^{l,\phi}$, which can be formulated as:

$$\mathbf{m}_i^{l,\phi} = \sigma(\hat{\mathbf{m}}_i^{l,\phi} + \mathbf{W}^{l,\phi}\mathbf{m}_i^{l-1,\phi}) + \mathbf{m}_i^{l-1,\phi} . \qquad (6)$$

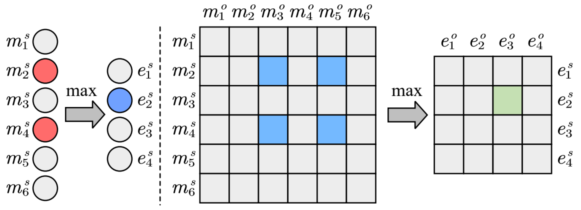

Figure 4: The 1D and 2D dynamic pooling.

3.3 Co-Predictor Layer

After the last reasoning block (i.e., the L-th block), we obtain the final mention representations $\mathbf{M}^{L,s}$ and $\mathbf{M}^{L,o}$, as well as the mention relation representation $\mathbf{Q}^L$. Since different mentions may belong to different entities, we apply 1D and 2D max-pooling to aggregate mention-level features into the entity level (Figure 4). Then, we apply two predictors to calculate two relation distributions for entity pair (e_i, e_j) and then combine them for obtaining the final prediction.

Local Predictor Based on the mention relation representation $\mathbf{Q}^L$ generated from the mention-based reasoning block, we adopt a 2D dynamic max-pooling (cf. Figure 4 right) to aggregate mention-level features into entity-level features:

$$\mathbf{G}_{ij} = \text{Max}\{\mathbf{Q}_{kt}^{L}\}_{k \in \mathcal{M}_{e_i}, t \in \mathcal{M}_{e_j}} , \qquad (7)$$

where $\mathcal{M}_{e_i}$ and $\mathcal{M}_{e_j}$ are the mention sets corresponding to the i-th and j-th entities, respectively.

Then we employ a FFN to generate prediction scores for entity pair:

$$y'_{ij} = \mathrm{FFN}(\mathbf{G}_{ij})\,. \tag{8}$$

Global Predictor Based on the mention representations $\mathbf{M}^{L,s}$ and $\mathbf{M}^{L,o}$, the representations of the i-th and j-th entities can also be generated by the 1D dynamic max-pooling (Figure 4 left).

$$\mathbf{e}_i^{\phi} = \mathrm{Max}\{\mathbf{m}_k^{L,\phi}\}_{k\in\mathcal{M}_{e_i}}\,, \tag{9}$$

where $\mathcal{M}_{e_i}$ is the mention set of the i-th entity, ϕ can be $\{s, o\}$ and $\mathbf{m}^{L,\phi} \in \mathbf{M}^{L,\phi}$. Then, a biaffine classifier (Dozat and Manning, 2017) is used to compute the relation scores between a pair of subject and object entities:

$$\mathbf{z}_i^s = \mathrm{MLP}^s(\mathbf{e}_i^s)\,, \tag{10}$$

$$\mathbf{z}_j^o = \mathrm{MLP}^o(\mathbf{e}_j^o)\,, \tag{11}$$

$$y''_{ij} = \mathbf{z}_i^{s\top}\mathbf{U}\mathbf{z}_j^o + \mathbf{W}[\mathbf{z}_i^s;\mathbf{z}_j^o] + b\,, \tag{12}$$

where $\mathbf{U}$, $\mathbf{W}$ and b are trainable parameters.

Joint Prediction The final relation probability for entity pair (e_i, e_j) with regards to the relationship r comes from the combination of the scores from both local and global predictors:

$$P(r|e_i, e_j) = \mathrm{Sigmoid}(y'_{ij} + y''_{ij})\,. \tag{13}$$

3.4 Learning

Considering the imbalance of positive and negative samples in document-level RE, we use asymmetric loss (ASL) (Ben-Baruch et al., 2020) instead of binary cross-entropy loss:

$$L_+ = (1 - P(r|e_i, e_j))^{\gamma+} \log(P(r|e_i, e_j))\,, \tag{14}$$

$$L_- = (P_n(r|e_i, e_j))^{\gamma-} \log(1 - P_n(r|e_i, e_j))\,, \tag{15}$$

where $\gamma+$ and $\gamma-$ are the focusing hyper-parameters for positive and negative samples, which aim to emphasize the contribution of positive samples and meanwhile down-weight the contribution of easy negatives samples ($\gamma_- > \gamma_+$). $P_n(r|e_i, e_j) = \max(P(r|e_i, e_j) - n, 0)$ is a probability shift mechanism that further filters out easy negative samples (probability margin $n \geq 0$ is a hyper-parameter). Here, the final loss function can be formulated as:

$$L = -\sum_{\mathcal{D}\in\mathcal{S}}\sum_{i\neq j}\sum_{r\in\mathcal{R}} \mathbb{I}\,(r=1)\,L_+ + \mathbb{I}\,(r=0)\,L_-, \tag{16}$$

where $\mathcal{S}$ denotes the whole dataset, and $\mathbb{I}(\cdot)$ refers to the indicator function.

Statistics	DocRED	CDR	GDA
# Train	3,053	500	23,353
# Dev	1,000	500	5,839
# Test	1,000	500	1,000
# Relations	97	2	2
Relations instances	63,443	3,116	46,343
Avg.# entities per Doc.	19.52	6.82	4.84
Avg.# mentions per Doc.	26.20	19.23	18.45

Table 1: Statistics of the three datasets.

4 Experimental Settings

4.1 Dataset and Evaluation

We conduct experiments on three benchmark datasets, including DocRED (Yao et al., 2019), CDR (Li et al., 2016) and GDA (Wu et al., 2019), and the statistical information is shown in Table 1. DocRED is a large-scale human-annotated dataset for document-level RE, including 3,053 documents for training, 1,000 for development and 1,000 for test. CDR and GDA are the widely-used document-level RE in the biomedical domain. CDR contains 1,500 PubMed abstracts with 3,116 relational facts, and GDA consists of 30,192 MEDLINE abstracts and 46,343 relational facts.

Following previous studies (Yao et al., 2019; Nan et al., 2020), we adopt F1 and Ign F1 as the evaluation metrics, where Ign F1 is calculated by excluding the common relation facts shared by the training, development and test sets. Depending on whether relation arguments occur within one sentence or not, F1 can be further split into intra-F1 and inter-F1. The results of our model are presented after a significant test (p$\leq$0.03).

4.2 Baselines

We make comparisons with the current state-of-the-art systems, including two categories. **1) Sequence-based methods**, which use different neural architectures to encode the entire document, include CNN (Zeng et al., 2014), BiLSTM (Cai et al., 2016), Context-Aware (Sorokin and Gurevych, 2017) and HIN (Tang et al., 2020). **2) Graph-based methods**, which construct homogeneous or heterogeneous graphs based on the whole document, include GAT (Velickovic et al., 2018), GCNN (Sahu et al., 2019), EoG (Christopoulou et al., 2019), LSR (Nan et al., 2020), GAIN (Zeng et al., 2020) and GLRE (Wang et al., 2020). Besides, some models leverage BERT (Devlin et al., 2019) for task improvements, including two-phase+BERT (Wang et al., 2019) and Coref+BERT

		Dev				Test	
		Ign F1	F1	Intra-F1	Inter-F1	Ign F1	F1
• **Without BERT**							
Sequence-based	CNN (Yao et al., 2019)	41.58	43.45	51.87	37.58	40.33	42.26
	BiLSTM (Yao et al., 2019)	48.87	50.94	57.05	43.49	48.78	51.06
	Context-Aware (Yao et al., 2019)	48.94	51.09	56.74	42.26	48.40	50.70
	HIN-GloVe (Tang et al., 2020)	51.06	52.95	-	-	51.15	53.30
Graph-based	GAT (Velickovic et al., 2018)	45.17	51.44	58.14	43.94	47.36	49.51
	GCNN (Sahu et al., 2019)	46.22	51.52	57.78	44.11	49.59	51.62
	EoG (Christopoulou et al., 2019)	45.94	52.15	58.90	44.60	49.48	51.82
	LSR (Nan et al., 2020)	48.82	55.17	60.83	48.35	52.15	54.18
	GAIN (Zeng et al., 2020)	53.05	55.29	61.67	48.77	52.66	55.08
Ours	**MRN**	**56.62**	**58.69**	**65.24**	**50.91**	**56.19**	**58.46**
• **With BERT**							
Sequence-based	BERT (Wang et al., 2019)	-	54.16	61.61	47.15	-	53.20
	Two-Phase+BERT (Wang et al., 2019)	-	54.42	61.80	47.28	-	53.92
	Coref+BERT (Ye et al., 2020)	55.32	57.51	-	-	54.54	56.96
Graph-based	LSR+BERT (Nan et al., 2020)	52.43	59.00	65.26	52.05	56.97	59.05
	GLRE (Wang et al., 2020)	-	-	-	-	55.40	57.40
	GAIN+BERT (Zeng et al., 2020)	59.14	61.22	67.10	53.90	59.00	61.24
Ours	**MRN+BERT**	**59.74**	**61.61**	**67.74**	**54.43**	**59.52**	**61.74**

Table 2: Performances on DocRED. The results of baselines are from their related papers.

(Ye et al., 2020).

5 Results and Analyses

5.1 Results on DocRED

Table 2 shows the experimental results on the DocRED dataset. We can find that MRN achieves 56.19% Ign F1 and 58.46% F1 on the test set, outperforming the sequence-based models with large margins. Specifically, MRN outperforms the best sequence-based model HIN by 5.04% and 5.16% in terms of Ign F1 and F1, demonstrating the effectiveness of capturing the mention-level contextual information for document-level RE.

Moreover, we observe that graph-based models (e.g., LSR and GAIN) generally perform better than sequence-based models (e.g., BiLSTM and HIN). This verifies the usefulness of constructing informative graphs. In particular, our MRN achieves the best Ign F1 and F1 scores, and outperforms GAIN by 3.53% and 3.38%, respectively. This shows the importance of emphasizing local mention-level interactions for document-level RE.

Furthermore, we observe that the performance can be substantially boosted with the help of BERT, where the Ign-F1 and F1 increase by 3.33% and 3.28% on the test set. Notably, MRN with GloVe embeddings is able to achieve better results than several BERT-based models, such as CorefBERT and GLRE. This suggests that our model is more effective in capturing complex interactions between close and distant mentions even without the help of pre-trained embeddings.

	F1	Intra-F1	Inter-F1
• **CDR** data			
ME-CNN (Gu et al., 2017)	61.3	57.2	11.7
BRAN (Verga et al., 2018)	62.1	-	-
C-CHAR (Nguyen and Verspoor, 2018)	62.3	-	-
GCNN (Sahu et al., 2019)	58.6	-	-
EoG (Christopoulou et al., 2019)	63.6	68.2	50.9
DHG (Zhang et al., 2020)	64.7	68.6	54.1
LSR (Nan et al., 2020)	64.8	68.9	53.1
MRN	**65.9**	**70.4**	**54.2**
• **GDA** data			
EoG (Christopoulou et al., 2019)	81.5	85.2	50.0
DHG (Zhang et al., 2020)	82.2	85.4	52.4
LSR (Nan et al., 2020)	82.2	85.4	51.1
MRN	**82.9**	**86.1**	**53.5**

Table 3: Results on CDR and GDA datasets.

5.2 Results on CDR and GDA

Table 3 shows the results on two biomedical datasets. Here, the baselines are also divided into sequence-based models (ME-CNN, BRAN, and C-CHAR) and graph-based models (GCNN, EoG, DHG, and LSR). Similar to the DocRED dataset, the graph-based models generally outperform the sequence-based models on CDR, which reveals the effectiveness of incorporating structural information and reasoning mechanisms in document-level

Model	Ign F1	F1	Intra-F1	Inter-F1
MRN	**56.62**	**58.69**	**65.24**	**50.91**
- entity type	55.70	57.61	64.25	49.74
- relative distance	55.61	57.64	64.33	49.70
- mention-based reasoning	53.71	55.66	61.55	49.14
- global predictor	52.77	54.53	60.81	46.99
- local predictor	53.73	55.70	61.95	49.01
dynamic average pooling	52.43	54.51	60.85	48.02
sharing object and subject	55.76	57.70	64.77	49.40
binary cross-entropy	56.08	58.02	64.52	50.29

Table 4: Ablation studies on the DocRED dataset.

RE. Besides, our MRN model achieves better performances than the state-of-the-art models on CDR and GDA datasets, outperforming LSR by 1.1% and the DHG by 0.7% respectively.

5.3 Intra- and Inter-sentence Relation Extraction

According to recent work (Yao et al., 2019), identifying 40.7% relations need the information of multiple sentences, which indicates that the reasoning ability of a model plays an important role in document-level RE. Thus, we also report the performances of intra- and inter-sentence relation extraction on three datasets in Table 2 and 3. We find that our model outperforms the current best models on all datasets in regard to both intra- and inter-F1. For example, MRN improves the intra-F1 by 3.57% and inter-F1 by 2.14% compared with GAIN on the development set of DocRED. This shows that mention-level reasoning is highly effective to capture complex interactions between mention objects and subjects, especially when not only local contexts but also long-range dependencies are considered.

5.4 Ablation Studies

We ablate each part of our MRN model on the development set of DocRED, as shown in Table 4. First, without entity type embeddings at the encoding layer, we observe slight performance drops. By removing relative distance information, the performance also decreases in a small degree. Furthermore, after removing interactive mention-based reasoning layer, global or local predictors, the performance goes down significantly. In particular, we find that the decrease of inter-F1 after removing global predictor (3.92%=50.91%-46.99%) is obviously higher than that for local predictor (1.90%=50.91%-49.01%), which verifies the usefulness of global features for long-dependency relation reasoning. A significant drop can be found when replacing the dynamic max pooling with average pooling. Finally, sharing object and subject representations and using binary cross-entropy loss, the model also has a certain degree of degradation.

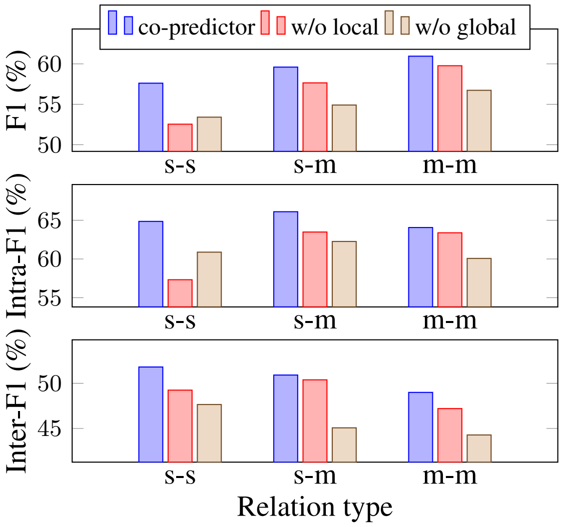

Figure 5: F1 scores of different relation types according to their entity arguments (e.g., (s)ingle mention and (m)ultiple mentions).

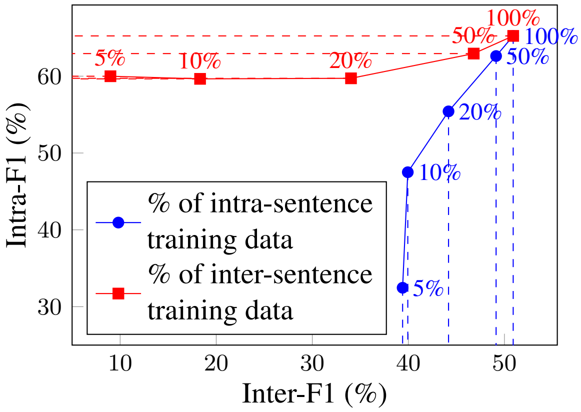

Figure 6: Intra- and inter-F1 scores vs. the proportions of inter- and intra-sentence training instances.

5.5 Effect Analysis for Co-Predictor

In this section, we investigate the effect of global and local predictors for MRN. We divide the relation instances in the development set of DocRED into three groups: the one where both subject and object arguments have single mention, the one where either subject or object argument has single mention, and the one where both subject and object arguments have multiple mentions. We also evaluate our model using different predictor configurations.

As shown in Figure 5, the model with both local and global predictors consistently outperforms

[S1] Allen Francis Moore (September 30, 1869 – August 18, 1945) was a U.S. Representative from Illinois. … [S4] He graduated from the Monticello High School in 1886 and from Lombard College, Galesburg, Illinois, in 1889. ...

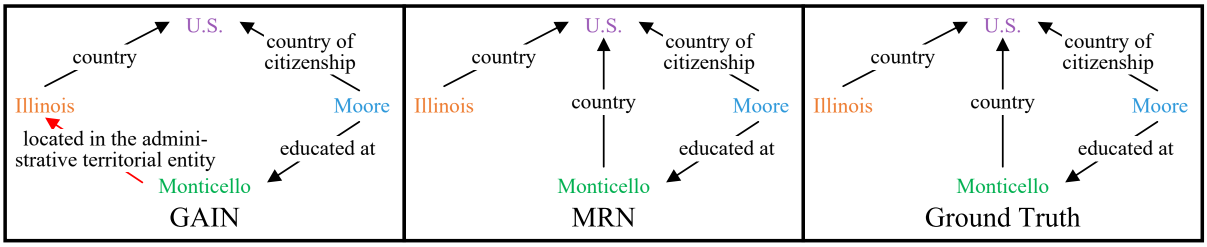

Figure 7: A case study for our model MRN and the state-of-the-art model, GAIN (Zeng et al., 2020). Only partial entities in the document are shown due to the space limitation. The red arrow denotes the wrong prediction.

the other ones. The F1s increase as the times that subjects and objects are mentioned increase (from s-s to m-m), indicating that multiple mentions appeared in various positions can provide more information to models. When removing the local predictor, we can observe that there is a huge drop for the s-s group, especially in Intra-F1. This demonstrates that the s-s group are mostly intra-sentence relations and they depend on local reasoning mostly. Moreover, if the global predictor is discarded, inter-F1 scores for the groups where subjects and objects are mentioned multiple times (s-m and m-m) drop the most. This reveals that the global predictor is more beneficial for extracting relations from multiple-mention entities or inter-sentence entities.

5.6 Effect Analysis for Inter- and Intra-sentence Training Data

As shown in Figure 6, we analyze the variation of inter- and intra-F1 scores when increasing or decreasing the proportions of intra- and inter-sentence training instances on the DocRED dataset. Note that the proportions of intra- and inter-sentence relation instances are 54.5% and 45.5% in the training set. The experimental setting is as below: first, we use 5% of inter-sentence training instances and observe the intra-F1. Then we gradually increase the percentage such as 10%, 20%, 50% and 100%. During this process, all intra-sentence training instances are used. The object for the above steps is to observe the effect of inter-sentence training instances on the intra-F1. In addition, we conduct similar experiments to observe the inter-F1 by gradually increasing the proportion of intra-sentence training instances.

As the red line shows in Figure 6, the inter-F1 is influenced slightly by the number of intra-sentence training instances. In contrast, the number of intra-sentence training instances has a significant impact on the inter-F1, since the inter-F1 grows dramatically (the blue line) when more intra-sentence training instances are added. This suggests that the interactions between intra- and inter-sentence relations indeed exist, and one may be helpful for reasoning the other.

5.7 Case Study

As shown in Figure 7, we present a case study to better understand the effect of our proposed MRN, in comparison with previous state-of-the-art baseline GAIN. We can observe that *Monticello* is the object of the intra-sentence relation triple (‘*Moore*’, *educated at*, ‘*Monticello*’) and also the subject of the inter-sentence (‘*Monticello*’, *country*, ‘*U.S*’). However, GAIN fails to identify the relation between ‘*U.S.*’ and ‘*Monticello*’, while MRN deduces it successfully. This demonstrates that the effectiveness of distinguishing subjects and objects at the inference stage, and MRN has strong capability for inter-sentence reasoning. Meanwhile, GAIN has made a wrong prediction between ‘*Monticello*’ and ‘*Illinois*’ in the 4-th sentence, indicating that our model has better local inference ability.

6 Conclusion

We propose a novel mention-based reasoning network (MRN) for document-level relation extraction. Our model is capable of capturing local and global contextual information as well as close and distant mention interactions, via multiple mechanisms such as a multi-hop mention-level reasoning block and collaborative predictors. Experimental results show that our proposed model achieves new state-of-the-art on three widely-used datasets. Through empirical analyses, we find that it is reasonable for document-level RE models to pay more attention on local context and close mentions. Meanwhile, global context and distant mention interactions are also highly important for document-level RE. Last but not least, joint reasoning with local and global context information is a reasonable and effective method for the task.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61772378), the National Key Research and Development Program of China (No. 2017YFC1200500), the Research Foundation of Ministry of Education of China (No. 18JZD015).

References

Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. 2020. Asymmetric loss for multi-label classification. *arXiv preprint arXiv:2009.14119*.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the ACL*, pages 756–765.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. In *Proceedings of the EMNLP-IJCNLP*, pages 4925–4936.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the NAACL*, pages 4171–4186.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the ICLR*.

Jinghang Gu, Fuqing Sun, Longhua Qian, and Guodong Zhou. 2017. Chemical-induced disease relation extraction via convolutional neural network. *Database*, 2017.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the ACL*, pages 1766–1776.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the NAACL*, pages 851–861.

Bo Li, Wei Ye, Zhonghao Sheng, Rui Xie, Xiangyu Xi, and Shikun Zhang. 2020. Graph enhanced dual attention network for document-level relation extraction. In *Proceedings of the COLING*, pages 1551–1560.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the ACL*, pages 2124–2133.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the ICLR*.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the ACL*, pages 1546–1557.

Dat Quoc Nguyen and Karin Verspoor. 2018. Convolutional neural networks for chemical-disease relation extraction are improved with character-based word embeddings. In *Proceedings of the BioNLP 2018 workshop*, pages 129–136.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the EMNLP*, pages 1532–1543.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. In *Proceedings of the ACL*, pages 4309–4316.

Cícero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the ACL-IJCNLP*, pages 626–634.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the EMNLP*, pages 1784–1789.

Hengzhu Tang, Yanan Cao, Zhenyu Zhang, Jiangxia Cao, Fang Fang, Shi Wang, and Pengfei Yin. 2020. Hin: Hierarchical inference network for document-level relation extraction. In *Proceedings of the PAKDD*, pages 197–209.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the ICLR*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the NAACL*, pages 872–884.

Difeng Wang, Wei Hu, Ermei Cao, and Weijian Sun. 2020. Global-to-local neural networks for document-level relation extraction. In *Proceedings of the EMNLP*, pages 3711–3721.

Hong Wang, Christfried Focke, Rob Sylvester, Nilesh Mishra, and William Wang. 2019. Fine-tune bert for docred with two-step process. *arXiv preprint arXiv:1909.11898*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the ACL*, pages 1298–1307.

Ye Wu, Ruibang Luo, Henry CM Leung, Hing-Fung Ting, and Tak-Wah Lam. 2019. Renet: A deep learning approach for extracting gene-disease associations from literature. In *Proceedings of the RECOMB*, pages 272–284.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the EMNLP*, pages 536–540.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the ACL*, pages 764–777.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential Reasoning Learning for Language Representation. In *Proceedings of the EMNLP*, pages 7170–7186.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the COLING*, pages 2335–2344.

Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In *Proceedings of the EMNLP*, pages 1630–1640.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the PACLIC*, pages 73–78.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the EMNLP*, pages 2205–2215.

Zhenyu Zhang, Bowen Yu, Xiaobo Shu, Tingwen Liu, Hengzhu Tang, Wang Yubin, and Li Guo. 2020. Document-level relation extraction with dual-tier heterogeneous graph. In *Proceedings of the COLING*, pages 1630–1641.

Huiwei Zhou, Yibin Xu, Weihong Yao, Zhe Liu, Chengkun Lang, and Haibin Jiang. 2020. Global context-enhanced graph convolutional networks for document-level relation extraction. In *Proceedings of the COLING*, pages 5259–5270.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph neural networks with generated parameters for relation extraction. In *Proceedings of the ACL*, pages 1331–1339.

A Effect Analysis for Sentence Number

As shown in Figure 8, we display the F1 scores of MRN and GAIN in the development set of DocRED with regards to the document length. As seen, we count the document length using the number of sentences within a document, which varies from 3 to 13. Results show that MRN attains better performances than GAIN, no matter that how the document length changes. In addition, the performance difference between MRN and GAIN becomes larger, when the document becomes longer. For instance, MRN outperforms GAIN by about 10% with regards to the group where the document length is 13. This demonstrates that our model is more robust for long documents.

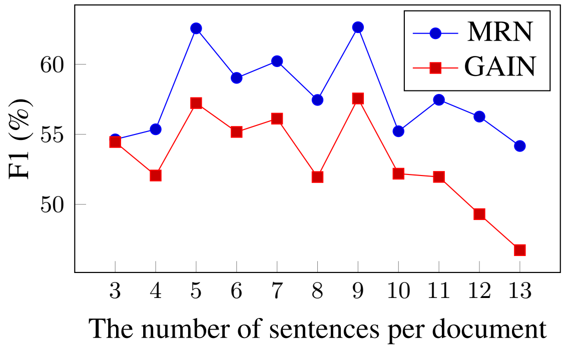

Figure 8: Comparisons between MRN and GAIN about the effect of the document length on the F1.

B Architecture Analysis for MRN Block

We conduct experiments for the interactive mention-based reasoning block based on the development set of DocRED, to understand which configuration works better. As shown in the left of Figure 9, our model performs the best when the kernel size is 3 in terms of all evaluation metrics. Meanwhile, the right part of Figure 9 shows that 3 seems to be a reasonable choice for the number of the MRN block.

C Analysis of Loss Functions

To compare the ASL and BCE loss, we compare their learning curves on DocRED and keep other settings of our model the same, as shown in Figure 10. With the comparisons of 80 epochs, we observe that the ASL loss helps our model converge to better performance than the BCE loss at a faster speed, demonstrating the effectiveness of the asymmetric strategies for positive and negative samples.

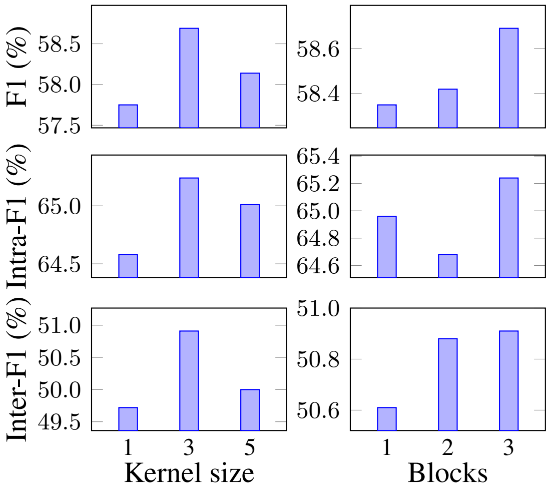

Figure 9: Analysis for the MRN block. The x-axis denotes the kernel size of 2D convolution in the local reasoning module (Left) and the number of used MRN blocks (Right).

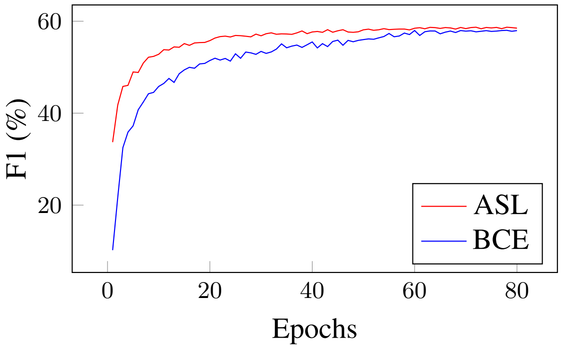

Figure 10: Comparison between the asymmetric loss (ASL) and binary cross-entropy (BCE) loss.

Hyper-parameter	DocRED	CDR	GDA
Batch size	32	16	32
Learning rate	1e-3	1e-3	1e-3
Gradient clipping	1	1	1
Weight decay	1e-5	1e-5	1e-5
Dropout for MRN block	0.5	0.5	0.5
Dropout for co-predictor	0.33	0.5	0.5
Blocks of MRN	3	3	3
Kernel size	3	3	3
Word embedding size	100	200	256
Entity type embedding size	20	20	20
Relative distance embedding size	20	20	20
LSTM hidden size	256	256	320
Positive focusing parameter	3	3	3
Negative focusing parameter	1	0	0
Probability margin	0.05	0.05	0.02

Table 5: Hyper-parameter settings for three datasets.

D Implementation Details

In this section, we provide more details of our experiments. We implemented MRN with PyTorch and trained it with a NVIDIA Tesla V100 GPU. Hyper-parameter settings for DocRED, CDR and GDA are listed in Table 5.

We adopt AdamW (Loshchilov and Hutter, 2019) as our optimizer. For fair comparisons, we utilize the GloVe embeddings (Pennington et al., 2014) for DocRED, pre-trained PubMed embeddings (Chiu et al., 2016) for CDR, and randomly initialized embeddings for GDA, following Yao et al. (2019) and Christopoulou et al. (2019). Moreover, we apply the uncased BERT-base model as our encoder for comparing with some baselines. We use the initial learning rate 1e-5 to fine-tune BERT.

Adversary-Aware Rumor Detection

Yun-Zhu Song,[1] **Yi-Syuan Chen,**[1] **Yi-Ting Chang,**[1] **Shao-Yu Weng,**[2] **Hong-Han Shuai,**[1]
[1]National Yang Ming Chiao Tung University, Taiwan
[2]National Tsing Hua University, Taiwan
{yunzhusong.eed07g, yschen.eed09g, joshchang0111.eed06}@nctu.edu.tw
nthu106071077@gapp.nthu.edu.tw, hhshuai@nctu.edu.tw

Abstract

While social media becomes a primary source of news now, it also becomes more challenging for people to distinguish the rumors and non-rumors, which attracts malicious manipulation and may lead to public health harm or economic loss. Consequently, many rumor detection models have been proposed to automatically detect the rumors based on the contents and propagation path. However, most previous works are not aware of malicious attacks, e.g., framing. Therefore, we propose a novel rumor detection framework, Adversary-Aware Rumor Detection including Weighted-Edge Transformer-Graph Network and Position-aware Adversarial Response Generator, to improve the vulnerability of detection models. To the best of our knowledge, this is the first work that can generate the adversarial response with the consideration of the response position. Experimental results show that our model achieves the state-of-the-art on various rumor detection tasks by the proposed Weighted-Edge Transformer-Graph Network and can maintain the performance under the adversarial response attack after the adversarial learning by Position-aware Adversarial Response Generator.[1]

1 Introduction

With the popularity and accessibility of social media, social media becomes the primary source for obtaining information.[2] Compared with traditional news, posts on social media are usually with shorter lengths and faster transmission speed, which also increases the difficulty of message verification. As such, social media are increasingly targeted for manipulation, leading to tremendous economic losses, and even deaths.[3] Take the COVID-19 pandemic as an example, a newly published study shows that at least some 800 people died because of a rumor about drinking highly-concentrated alcohol can disinfect their bodies (Islam et al., 2020). Therefore, fighting against misinformation in social networks gains a great deal of attention and becomes essential and inevitable.

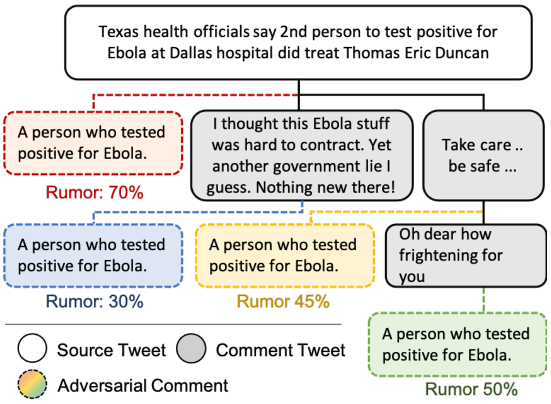

Figure 1: Position test for an adversarial response. The reply position can make an influence on the detection model.

In this paper, we study the problem of rumor detection on social media, where a rumor is defined as an unverified and instrumentally relevant information statement in circulation (DiFonzo and Bordia, 2007; Zubiaga et al., 2018). Given a conversation thread, including a source post and related responses, the rumor detection task aims to determine whether the source post is a rumor or not. Previous works of rumor detection can be categorized into three classes according to the data usage. Content-based approaches only use the textual information of the source posts and the user responses (Ma et al., 2016, 2019), while graph-

---

[1] The codes are released as a public download at `https://github.com/yunzhusong/AARD`.

[2] `https://pewrsr.ch/3nzYpQd`

[3] `https://s3.amazonaws.com/media.mediapost.com/uploads/EconomicCostOfFakeNews.pdf`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1371–1382
August 1–6, 2021.

based methods (Ma et al., 2017, 2018; Bian et al., 2020) consider the message propagation paths or model the propagation paths as a tree. In addition to considering the textual information and propagation paths, user-based methods take user profiles into consideration (Giudice, 2010; Liu et al., 2015).

However, two challenges of detecting rumors have not been completely addressed. 1) *Robustness to different responses.* Previous works of rumor detection take all the responses in the conversation thread into consideration and extract important information in a data-driven manner. However, not all responses can help detect rumors, especially for the malicious framing responses, i.e., the responses promote a particular misleading interpretation. As such, it is necessary to provide a learning mechanism to enable the selection of important responses. 2) *Vulnerability to malicious attack.* Most of the existing methods only rely on datasets, which may be vulnerable to the adversarial attack, e.g., the attack of Twitter bots. (Ma et al., 2019) make the first attempt to utilize a GAN-based approach to produce adversarial text. Nevertheless, it does not consider the graph structure of the conversation thread, i.e., the generator cannot determine which responses it should reply to. In fact, the reply position can make an influence on the detection model. As shown in Figure 1, the predicted probability that the source post is a rumor ranges from 30% to 70% according to different positions of attached adversarial responses. It is challenging to generate an adversarial attack by simultaneously considering both structural and textual information since the gradient-based methods cannot be directly applied due to the discrete nature of text and structure.

To address these two challenges, we propose a novel framework, namely, Adversary-Aware Rumor Detection (AARD), which includes i) Weighted-Edge Transformer-Graph Network (WETGN) and ii) Position-aware Adversarial Response Generator (PARG). Specifically, given a source post, responses, and propagation structure as an input, we use a transformer-based encoder to encode each token in the whole conversation thread to exploit the existing pre-trained knowledge. Each token can jointly attend to different tokens regardless of the token position, which gives the model the flexibility to break the distance limit in the sequence. Since the transformer layer only takes the responses in the conversation thread as a sequential input, the propagation path is not considered. Therefore, a Graph Convolutional Network (GCN) is applied to embed the structure by taking the token embeddings as node features, and aggregates the features according to the propagation paths. Inspired by (Veličković et al., 2018), we construct the edge features from the incident nodes and build an edge filter before the GCN layers to address the first challenge. As such, we can leverage the advantages of both transformer and graph neural networks.

Moreover, to address the second challenge, we build a Position-aware Adversarial Response Generator (PARG) to train the detector by adding an adversarial response to the conversation thread. Specifically, based on a transformer-based encoder-decoder framework, PARG takes the source post with part of the corresponding responses as input to generate an adversarial response. Nevertheless, choosing the attached position for the structure-aware detection model is also crucial. PARG is trained to select the position by considering the correlation between the generated response and each of the existing posts. However, the position selection involves the argmax function, which is a non-differentiable operation. Therefore, to enable the backpropagation of gradients from the detector, PARG instead predicts the probabilities of attaching the generated response to each existing response. When updating the edge weights of the attached edges in the detection model, the generator can use the gradient to correct the predicted probabilities.

By fine-tuning WETGN with the adversarial data generated by PARG, WETGN is equipped with a certain degree of resistance to attack and maintains the performance on clean datasets. Nevertheless, although an attacker can generate adversarial examples with the detection model, it may create nonsense sentences, which can be manually excluded (noticeability). On the other hand, imposing constraints on the generated examples decreases the possibility of finding effective adversarial examples (success rate). This paper designs a training pipeline to strike a balance between success rate and noticeability. As such, the attacker is trained to decrease the detection performance and approach the real responses simultaneously.

Extensive experimental results manifest that the proposed WETGN outperforms state-of-the-art approaches on three rumor detection benchmarks by at least 4.9%, 2.89%, and 3.87% on Twitter15, Twitter16, and Pheme datasets. At the same time, AARD can resist the adversarial attack. Moreover, the proposed PARG can successfully attack existing detection models with a success rate of at least 25.08%. The success rate can be significantly reduced after fine-tuning, which shows the compatibility and usefulness of PARG.

2 Related Work

Early works rely on textual content to verify the authenticity of social media posts. For example, Badaskar et al. (2008) quantify the frequency of uncommon phrases in the articles and syntactic and semantic checking, while Potthast et al. (2018) detect the truthfulness of news by analyzing its writing style. Ma et al. (2016) use recurrent neural networks to learn both the temporal and textual representation of the source posts and user responses, which highly improves prior methods that utilize hand-crafted features. Also, Volkova et al. (2017) extract text features with LSTM and CNN structures to make the prediction.

On the other hand, a recent line of studies focuses on automatically detecting rumors based on the tree structure of the conversation thread (Ma et al., 2017; Wei et al., 2019; Kumar and Carley, 2019; Lu and Li, 2020). For instance, Ma et al. (2018) build a tree-structured recursive neural network to catch the hidden features from either top-down or bottom-up propagation structure and text content. However, it can only obtain the information of one propagation structure and ignore the other. To solve this problem, Bian et al. (2020) use the GCN-based model to embed both propagation and dispersion structures and enable the proposed method to process graph/tree structures and learn higher-level representation more conductive to rumor detection. Besides, by utilizing the hierarchical structure in the conversation thread (i.e., parent, child, before, after and self), Khoo et al. (2020) adopt the idea in Shaw et al. (2018) to perform structure-aware self-attention.

In addition, stance and user information are also used in several studies. By using stance prediction as the auxiliary task with multi-task learning, Wei et al. (2019); Li et al. (2019); Kumar and Carley (2019) have demonstrated that stance prediction plays a vital role in rumor detection. Furthermore, Li et al. (2019) incorporate the collected user credibility to supervise the detection model. Lu and Li (2020) construct the propagation network by using retweet sequences of users with user profiles to capture the correlation between user propagation and its source post. The uniqueness of our work lies in improving the vulnerability of detection models.

Due to the small or non-diversified training data, a recent line of studies utilizes the adversarial learning (Ma et al., 2019; Yang et al., 2020) or data augmentation to improve the detectors. For example, Ma et al. (2019) propose an RNN-based GAN model, where the generator aims to generate conflicting information in the conversation thread, and the discriminator is forced to learn more robust features. On the other hand, Han et al. (2019) augment data by using semantic relatedness to assign pseudo labels to unlabeled tweets. However, the structural information is important but not considered in these previous works.

3 Methodology

3.1 Problem Formulation

Given a conversation thread comprised of a source post and the corresponding responses, rumor detection aims to determine whether the claim of the source post is a rumor or not. Let $X = \{x_0, x_1, \cdots, x_i, \cdots, x_N\}$ denote a conversation thread, where x_0 represents the source post and $\{x_i\}_{i=1}^{N}$ represents the N responses. A graph $G = \langle V, E\rangle$ is constructed by taking each element in X as a node and the interactions between elements as the edge connections to form the node set V and the edge set E, respectively. For example, if nodes x_u and x_v have a direct interaction (e.g., commenting or retweeting) in the same conversation thread, an edge $(x_u, x_v) \in E$ is constructed accordingly. Due to the nature of social media, the graph G is an acyclic tree. Let $y \in \{\mathit{rumor}, \mathit{non\text{-}rumor}\}$ be the class label. Rumor detection aims to predict y given the graph G.

3.2 Rumor Detection Model

Transformer Encoder: To obtain the representation of text contents, we adopt the transformer-based encoder to explore the pre-trained knowledge. We first flatten the tree-structured graph in the chronological order, which constitutes a source post followed by a sequence of responses. Specifically, the source post and each response are started by a special token [CLS] and ended by another special token [SEP] to indicate the separation of nodes (Liu and Lapata, 2019). In this setting, we

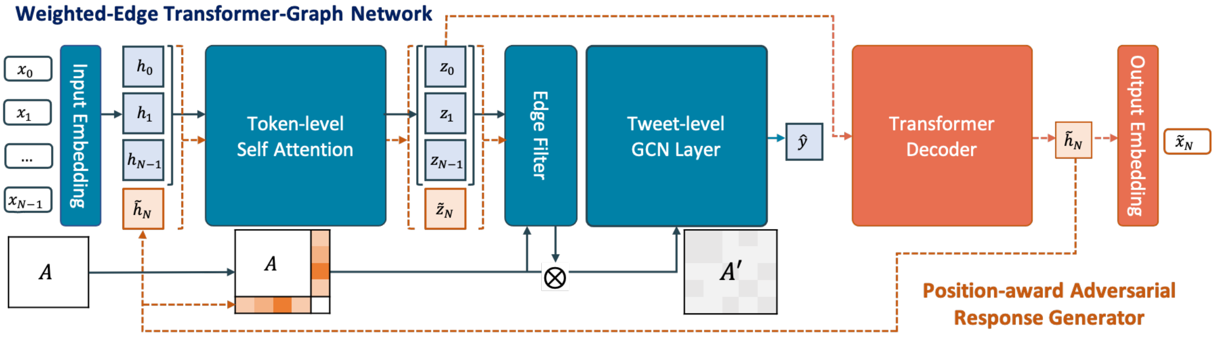

Figure 2: Overview of adversary-aware rumor detection framework.

allow each token to jointly attend to nodes in different positions for better capturing semantics. Let $h_i^{(0)} \in \mathbb{R}^{|x_i| \times d}$ denotes the d-dimensional embedding of a node x_i, which is constructed by token embedding, segment embedding, and position embedding (Devlin et al., 2019). The embedding of a conversation thread $H^{(0)}$ can thus be obtained as follows:

$$h_i^{(0)} = E_{in}(x_i) = E_{tok}(x_i) + E_{seg}(x_i) + E_{pos}(x_i),$$

$$H^{(0)} = [h_0^{(0)} \mid\mid h_1^{(0)} \mid\mid ... \mid\mid h_N^{(0)}] \in \mathbb{R}^{M \times d},$$

where $||$ is the concatenation operation, and $M = |x_1| + |x_2| + ... + |x_N|$ indicates the length of input sequence. The embedding is passed through several transformer layers. At layer $l+1$, the features from previous layer $H^{(l)}$ is transformed by three linear layers to form the query Q, key K, and value V matrices, and the output $H^{(l+1)}$ is computed as follows:

$$Q = H^{(l)} W_q, K = H^{(l)} W_k, V = H^{(l)} W_v,$$

$$H^{(l+1)} = softmax(\frac{QK^T}{\sqrt{d_k}} V),$$

where W_q, W_k, and W_v are trainable parameters and d_k is the scaling factor to prevent small gradients. The feature of token [CLS] from the last layer is taken to represent each node, which are denoted as $Z = [z_0 \mid\mid z_1 \mid\mid ... \mid\mid z_N] \in \mathbb{R}^{N \times d}$.

GCN Classifier: The interactions between responses, e.g., commenting or retweeting, are essential information for the detection model to judge the source post (Castillo et al., 2011). The responses not only contain the users' opinions but also reveal the propagation paths through social media. Since Graph Convolutional Network (GCN) is one of the most effective models for graph-structured data modeling, we leverage GCN to consider the propagation path. The message propagation function of a multi-layer GCN defined in the first-order approximation of Chebyshev polynomials is derived as follows:

$$Z^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} Z^{(l)} W^{(l)}),$$

where $Z^{(l)} \in \mathbb{R}^{N \times d}$ is a hidden feature matrix at the l-th layer, $\widetilde{A} = A + I$ is a binary adjacency matrix with self-connection, $\widetilde{D}_{u,u} = \sum_v \widetilde{A}_{u,v}$ is a degree matrix, $W^{(l)}$ is a learnable matrix, and $Z^{(0)} = Z$ is the node features. Although GCN has been proved to be effective for extracting structural information, the information may not be faithful after aggregation with specific nodes, e.g. framing responses. Therefore, considering the potential existence of various redundant or adversarial messages, we propose to filter the edges by learning the importance of edges before the aggregation.[4] Specifically, based on the node features extracted by the transformer encoder, the importance of an edge $e_{u,v}$ between nodes x_u and x_v is constructed as follows:

$$e_{u,v} = f_{edge}(z_u, z_v) = \sigma([z_u \mid\mid z_v] W_{edge} + b_{edge}),$$

where W_{edge} and b_{edge} are trainable parameters. The predicted importance are then used to construct a weighted adjacency matrix as follows:

$$A'_{u,v} = A_{u,v} + I_{u,v} + e_{u,v}.$$

For the final prediction, the model considers the entire graph by taking mean pooling over all convolved node features instead of only taking the root

---

[4] We also simulate the generation process of adversarial responses in the next subsection and assess the attack performance in the experiment.

node's attribute. The prediction is calculated from the feature matrix of the last GCN layer L:

$$\hat{y} = softmax(Mean(z_0^L, z_1^L, .., z_N^L)W_o + b_o),$$

where W_o and b_o are trainable parameters.

3.3 Adversarial Response Generation

To further improve the vulnerability of the detection model, we explore adversarial learning under the setting of white-box attack, i.e., the parameters and gradients of the detector are exposed when updating the attacker. Specifically, we design a response generation model that attaches new adversarial responses to the conversation threads as an attacker against the detection model. For the text generation model, we adopt an encoder-decoder framework with transformer layers, which shows outstanding performance in text generation (Liu and Lapata, 2019). However, the gradients cannot be backpropagated from the detector to the generator for updating, due to the non-differentiable argmax function (de Masson d'Autume et al., 2019) in generation. To solve this problem, we tie the generator's output layer E_{out} with the embedding layer E_{in}, which means the weights of the two layers are mutually transposed. In this way, the features before the argmax function can be treated as the embedding of the generated response. Given an input sequence $\{x_i\}_{i=0}^{n-1}$, a response is generated as follows:

$$h_i = E_{in}(x_i),$$

$$\widetilde{h}_n = f_{dec}(f_{enc}(h_0, \cdots, h_{n-1})),$$

$$\widetilde{x}_n = argmax(softmax(E_{out}(\widetilde{h}_n))),$$

$$E_{in}(\widetilde{x}_n) \sim \widetilde{h}_n,$$

where f_{enc} and f_{dec} are the encoder and decoder. The features $\widetilde{h}_n$ can thus be directly used in the detector without breaking the gradient path. Besides, to reduce the model complexity, the encoder is shared between the generator and detector.

Nevertheless, for rumor detectors that incorporate propagation paths, the location to attach the generated responses is also a crucial problem. Similar to text generation, the operation of choosing the attached position is also discrete. To enable the model to simultaneously learn the position for generating responses, the generation model additionally predicts the edge weights $\{e_{n,i}\}_{i=0}^{n-1}$ between the generated response $\widetilde{x}_n$ and all existing nodes in the training process with Gumble softmax function (Jang et al., 2016), i.e.,

Table 1: Statistics of the datasets.

	Twitter15	Twitter16	Pheme
# of tree	1458	818	3720
# of node	41154	18618	67238
# of rumors	1086	613	1863
# of non-rumors	372	205	1863
Avg text length	15.84	15.87	17.79
Max text length	136	383	78
Min text length	2	2	2

$$\pi_i = [\widetilde{h}_n \mid\mid h_i]W_p + b_p, \ \forall i \in (0, \cdots n-1),$$

$$e'_{n,j} = softmax(log(\pi_i + g_i)/\tau),$$

where g_i is i.i.d sampled from Gumble distribution (0,1), τ is a hyper-parameter for controlling the smoothness of output distribution, and $W_p \in \mathbb{R}^{2d\times 1}$ and b_p are trainable parameters. A higher edge weight indicates a higher possibility that the attack can succeed at a specific position. It is worth noting that we focus on generating only one response to attack the model for validating the performance of the proposed attack. The proposed model can be extended to iteratively generate adversarial responses at different positions in the conversation thread.

3.4 Training Pipeline

The adversarial examples cannot only demonstrate the weakness of the detection model, but also provide the opportunity to improve the vulnerability. However, there is a trade-off between the attack success rate and the noticeability. To strike a balance between them, the attacker should generate a response that is close to the real ones. Base on this idea, we i) decompose the conversation threads into several subtrees for the attacker to predict the next real response and ii) design a three-stage learning pipeline to mutually learn the attacker and the detector.

Firstly, the generator is trained along with the detector to increase the detection accuracy. To generate quality responses, we provide the generator target sentence by decomposing one conversation tree into several subtrees. That is, given a subsequence of the conversation thread, the goal of the generator is to synthesize the next real response x. We only train the decoder layer θ_{dec} of the generation model while fixing the parameters of the encoder. For the detection model, the trainable layers are the encoder θ_{enc} layer, filter layer θ_{filter}

and the GCN layer θ_{gcn}. The objective function of generator is the binary-cross entropy for rumor classification and the cross entropy for text perplexity $L_{txt} = -\frac{1}{|x|}\sum_{m=1}^{|x|} logP_{gen}(w_m|w_{1:m-1})$, while the detector minimizes the rumor classification loss. The loss of the first stage (L^{1st}) is derived by summarizing L_{gen} and L_{det} with a weight λ:

$$L_{gen}(\theta_{dec}) = CE(\hat{y}, y) + L_{txt},$$

$$L_{det}(\theta_{enc}, \theta_{filter}, \theta_{gcn}) = CE(\hat{y}, y),$$
$$L^{1st} = \lambda L_{gen} + (1-\lambda)L_{det}.$$

The second training stage is to train the generator while fixing the detector. In this stage, the goal of the generator is to generate a response that can confuse the detector as an attacker. The detector takes the adversarial data as the input and makes a prediction, and the target label is reversed $\bar{y}$, i.e., the rumor becomes non-rumor and vice versa. To make the generated text unnoticeable, i.e., similar to human written sentences, the attacker is also trained to optimize L_{txt}. Therefore, the loss of the second stage is

$$L^{2nd} = L_{gen}(\theta_{dec}) = CE(\hat{y}, \bar{y}) + L_{txt}.$$

The third training stage is to fine-tune the detector under the fixed attacker. The detector is trained on the adversarial data and optimized to make the correct prediction. This training equips the detector with the ability to resist the attack and also learn to filter out the potential redundant or adversarial messages. The objective function is as follows:

$$L^{3rd} = L_{det}(\theta_{enc}, \theta_{filter}, \theta_{gcn}) = CE(\hat{y}, y).$$

4 Experimental Results

4.1 Experiment Settings

Datasets. We evaluate the proposed AARD on three public datasets including Pheme (Zubiaga et al., 2016), Twitter15 and Twitter16 (Ma et al., 2017) datasets since these datasets contain source posts, the corresponding responses, and the rumor labels. The original labels of Twitter15 and Twitter16 datasets include four classes, i.e., true rumor, false rumor, unverified rumor, and non-rumor. In this paper, we focus on differentiating rumors from non-rumors, and thus regard the first three classes as rumors. The Pheme dataset is collected based on five events with two classes, i.e., rumor and non-rumor. Due to the privacy protection policy of Twitter, the contents of responses are not included in the dataset. Therefore, we crawl the contents of responses by ourselves. If all contents have already been removed, we delete the empty tweet from the tree. Meanwhile, following the previous work (Khoo et al., 2020), we also eliminate retweets with an empty text description. The statistics are shown in Table 1.

Baselines. The selection of the baselines follows two criteria: 1) "rumor detection" or "rumor veracity classification" and 2) availability of source codes. Specifically, this paper designs a rumor detector and generator for the "rumor detection" task, which is a binary classification task. In contrast, the "rumor veracity classification" is a four-class classification task (non-rumor/true-rumor/false-rumor/unverified rumor). As Ma et al. (2019) also target the binary classification task between rumor and non-rumor, it is selected as baselines. For other works focusing on rumor veracity classification (Ma et al., 2018; Kumar and Carley, 2019; Yang et al., 2020; Khoo et al., 2020; Bian et al., 2020), one possible way for comparing with these works is to reimplement the models and change their settings to the binary classification. Therefore, Bian et al. (2020) and Ma et al. (2018) are used as baselines by reimplementing and changing the labels as binary classification. Unfortunately, it is hard to compare with some baselines that do not release the source code (Yang et al., 2020) or require additional information, e.g., user information and the stance of each response (Kumar and Carley, 2019). Finally, the baseline methods are listed: (1) RvNN (Ma et al., 2018), based on tree-structured recursive neural networks with GRU units to obtain representations from the propagation structure in the bottom-up (BURvNN) or top-down (TDRvNN) manners, (2) GAN-GRU (Ma et al., 2019), the GAN-style learning model where the discriminator and generator are recurrent neural networks with GRU units, (3) BiGCN (Bian et al., 2020), the GCN-based model that can embed both propagation and dispersion structures and enhance the root node features, and (4) GCAN (Lu and Li, 2020), which learns the retweet propagation features based on user features by a structure employed convolution and recurrent neural networks.

Implementation Details. We use the same hyperparameters for all datasets. Specifically, the batch

Table 2: Rumor/non-rumor detection results. The '-EF' means the model without the edge filter, and the '-DD' is trained without the data decomposition, while the '-PARG' indicates the detector without adversarial learning.

		Twitter15				Twitter16				Pheme			
Method	Class	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
BURvNN	NR	85.02	69.77	72.43	71.01	80.08	64.13	52.68	57.78	76.83	78.28	74.36	76.22
	R		90.51	89.31	89.89		84.47	89.60	86.95		75.62	79.30	77.38
TDRvNN	NR	86.53	69.19	72.23	76.10	84.83	58.05	65.57	76.11	70.43	67.43	81.77	73.42
	R		92.44	91.10	89.85		**93.78**	90.25	87.03		77.78	59.09	65.75
GAN-GRU	NR	83.50	67.46	67.57	67.46	80.74	64.16	54.27	58.51	75.13	75.61	74.19	74.94
	R		88.98	88.94	88.95		85.45	89.59	87.44		74.74	76.07	75.38
BiGCN	NR	88.16	84.15	62.83	70.49	87.30	**87.12**	52.17	63.34	80.97	79.14	84.18	81.46
	R		89.17	96.10	92.39		87.15	**98.03**	92.14		83.09	77.78	80.21
AARD	NR	93.06	85.63	**87.84**	86.50	89.73	83.49	74.63	78.73	83.93	82.59	**86.93**	84.45
	R		**95.87**	94.84	95.32		91.59	94.95	93.23		**86.52**	80.91	83.20
-EF	NR	92.99	87.93	84.59	85.93	89.17	82.63	75.12	78.05	84.11	84.83	83.44	83.99
	R		84.87	95.85	95.33		91.72	94.06	92.79		83.83	84.78	84.18
-DD	NR	90.86	80.24	87.03	83.05	89.19	79.22	**78.54**	78.85	83.09	85.56	79.79	82.50
	R		95.50	92.17	93.72		92.59	92.88	92.73		81.12	86.40	83.62
-PARG	NR	**93.47**	**89.06**	85.68	**86.93**	**90.19**	87.02	73.17	**79.29**	**84.84**	**86.31**	82.90	**84.48**
	R		95.26	**96.13**	**95.64**		91.22	96.08	**93.57**		83.74	**86.77**	**85.15**

Table 3: Detection results for True/False Rumor. The results of GCAN are from the original paper. ($\star$) Indicates the results are taken from the reference.

		Twitter15			
Method	Class	Acc.	Prec.	Rec.	F1
GCAN*	TR	87.67	-	-	-
	FR		82.57	82.95	82.50
AARD	TR	**94.13**	95.54	92.78	94.11
	FR		92.87	95.49	94.14
		Twitter16			
Method	Class	Acc.	Prec.	Rec.	F1
GCAN*	TR	90.84	-	-	-
	FR		75.94	76.32	75.93
AARD	TR	**92.01**	94.45	89.51	91.87
	FR		90.13	94.22	91.99

sizes of the detector and the generator are 48. The learning rate of the generator is 0.002 and warms up for 2000 steps. The learning rate of the decoder is set to 0.002. The token embeddings are initialized from BERT, therefore, the settings refer to the pretrained model bert-base-uncased (Devlin et al., 2019). The Transformer Encoder has 12 self-attention layers, and the layer number of GCN (L) is 2. The loss weight λ is 0.5.

Evaluation metrics. The evaluation metrics include accuracy, precision, recall and F1 score of two classes. We split each dataset into five-fold (80% for training and 20% for testing), and report the average results.

4.2 Rumor Detection

Overall Performance. Table 2 shows the performance of all models on the rumor detection task. The results manifest that the proposed AARD outperforms all the state-of-the-art models by at least 4.9%, 2.89%, and 3.87% on Twitter15, Twitter16, and Pheme datasets, respectively. Compared with the methods that use the recursive (BURvNN and TDRvNN) or recurrent (GAN-GRU) neural network, graph-network based models achieve better results, indicating that the propagation structure contains important information when detecting rumors. Different from the BiGCN, which uses the tf-idf vectors as the node features, AARD uses self-attention layers to encode the posts as the node features. The Transformer encoder enables the model to embed tokens across nodes, thus strengthening the node representation.

Moreover, the bottom rows of Table 2 show the ablation studies of the proposed AARD. The results show that both edge filter and data decomposition play important roles. On the other hand, the model can achieve a promising performance on the rumor detection task without adversarial learning. The goal of adversarial learning is to address the second challenge, i.e., vulnerability to malicious attacks. Accordingly, the Position-aware Adversarial Response Generator (PARG) is designed to improve the robustness under a malicious attack. As the original testing dataset is clean (without attacks) or only contains few manual attacks, the detection accuracy may not be significantly improved. However, when a detector is without adversarial learning (denoted by AARD-PARG in Table 2 and WETGN in Table 4), the performance drastically decreases when encountering an attack (adding one adversarial node to the conversation tree), which can be alleviated by fine-tuning on the adversar-

Table 4: Results of adversarial attack and training, where '-EF' indicates the detector without edge filter.

	Twitter15			Twitter16			Pheme		
	Accuracy	Diff.	ASR	Accuracy	Diff.	ASR	Accuracy	Diff.	ASR
WETGN	93.47 → 71.13	-23.34	28.87	90.19 → 59.80	-30.39	40.20	84.84 → 74.92	-9.92	25.08
-EF	92.99 → 62.20	-30.79	37.80	89.17 → 51.49	-37.68	48.51	84.11 → 74.59	-9.52	25.41

	After Adversarial Training											
	Adversarial		Clean		Adversarial		Clean		Adversarial		Clean	
	Acc.	Diff.	Acc.	Diff.	Acc.	Diff.	Acc.	Diff.	Acc.	Diff.	Acc.	Diff.
AARD	92.44	-1.03	93.06	-0.41	87.94	-2.25	89.73	-0.46	82.53	-2.31	83.93	-0.91
-EF	90.72	-2.27	91.41	-1.58	86.70	-2.47	83.92	-5.25	78.49	-5.62	82.98	-1.13

Table 5: Generated adversarial examples of testing data, showing the source post, responses and the generated response. The user names are replaced to remain anonymous.

R	**Source:** st. Louis co police tell me ofcr shot a man who pointed handgun at him at chambers & sheffingdell at about 1 a.m **Response:** @name1 Uh! oh. this is serious and officially out of control. # @name1 thank you so much for getting this scoop. people have been on pins and needles at the Reddit live feed. @name1 how many weeks until police interview witnesses? @name2 @name3 or the video! **Generated Response:** @name1 @name2 is not a false alarm.
NR	**Source:** BBC reports that broadcaster Sir Terry Wogan has died of cancer aged 77 **Response:** @name1 A huge loss.. no more his whit and sing song voice.. tragic news @name1 Sad sad news, yet another British icon gone [UNK] @name1 @name2 another national treasure gone! so sad. @name1 RIP Wogan! thoughts and prayers with your loved ones! @name1 #prayforpudsey @name1 #terrywogan cancer is an absolute bitch , so many celebrated people have been taken by it in just a few weeks **Generated Response:** @name1 @name2 #I can not believe it is a true story.

ial examples. The detailed analysis of adversarial learning is discussed in Sec. 4.3.

To further analyze the impact of data quantity on model performance, we train the models under different quantities of data, ranging from 5% to 100%, and evaluate them on the same testing set. Figure 4 shows the results, which indicate that our model can still achieve leading performance even with minimal training data.

True/False Rumor Detection. We separately compare AARD with another graph-based model, GCAN, since GCAN focuses on true rumor/false rumor classification task and evaluates on Twitter15 and Twitter16.[5] Table 3 shows the results of true rumor/false rumor classification, which indicates that AARD also has an excellent performance in the rumor classification. We consider it is because differentiating the false rumor from true rumor also requires the model to carefully examine the responses.

Early Detection. Early detection aims to detect rumors in the early stage, which is an important indicator for evaluating the detection model. We refer (Bian et al., 2020) and (Ma et al., 2019) to construct the detection deadlines of Twitter15 and Pheme datasets and only use the responses released before the deadlines to evaluate the accuracy. Figure 3 compares the accuracy with different detection deadlines. At the early stage, i.e., when a post just came out with extremely few responses, the accuracy of different models is around 0.75 on the Twitter15 dataset. After just a few minutes, the accuracy of our model reaches 0.85, whereas the accuracy of baselines only approximates 0.8. For the Pheme dataset, we squeeze the time sequence and find that the performances of all models become stable but our model stably outperforms others.

4.3 Adversarial Attack

Table 4 shows the model performance under an adversarial attack generated by PARG. The notation "→" indicates the performance before and after the attack, while "Diff." and "ASR" represent the accuracy difference and the Attack Success Rate (ASR) of PARG, respectively. The results indicate that the proposed PARG significantly reduces the accuracy of the detectors. The ASR is lower on the Pheme dataset than on Twitter15 and Twitter16 datasets because Pheme is a much larger dataset than the others. Therefore, the detector can learn more indicated features from Pheme and be more robust. Moreover, by comparing the performance of the detector with (WETGN) and without (-EF) edge filter, adding the edge filter can help the detector resist the attack, that is, the "Diff." is lower on

---

[5] The GCAN requires user profiles for training, which are not crawled in our datasets. Therefore, GCAN is not compared in the R/NR classification.

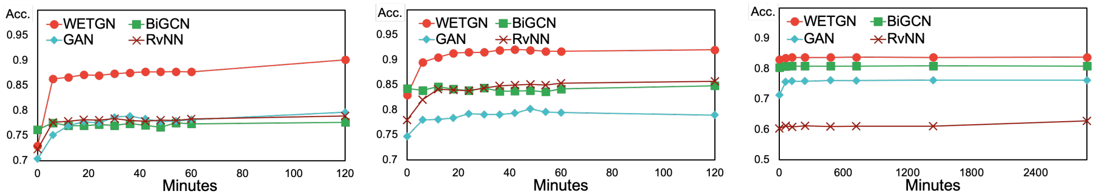

Figure 3: Early detection on Twitter15 (left), Twitter16 (middle) and Pheme (right).

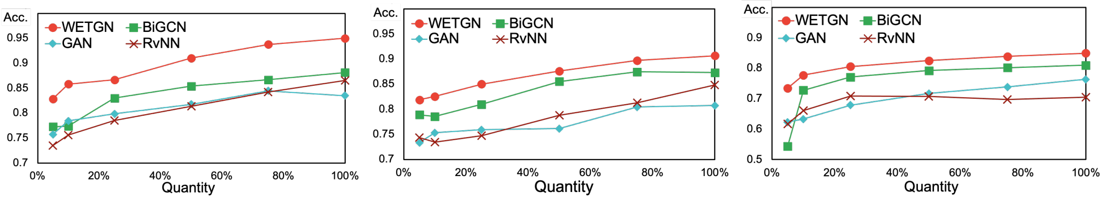

Figure 4: Data scarcity on Twitter15 (left), Twitter16 (middle) and Pheme (right).

WETGN for Twitter15 and Twitter16. In addition, we use the adversarial samples to fine-tune the detector (AARD). The bottom of Table 4 shows the results of the fine-tuned model, where the Adversarial/Clean indicates the accuracy tested on the dataset with/without adversarial attacks. The performance of the AARD without the edge filter is also provided, which also suggests the edge filter can improve the robustness.

When a detector is without adversarial learning (WETGN), the performance decreases by at least 20% on Twitter15 and Twitter16 datasets when encountering an attack. In contrast, the proposed detector with adversarial learning can maintain the performance even when the attacker has access to the model's parameters (white-box attack). It is worth noting that there may be a trade-off between the adversarial accuracy (test on adversary data) and the clean accuracy (test on clean data) (Raghunathan et al., 2019), depending on how we fine-tune the detection models, e.g., only using the adversarial data or using both kinds of data. The AARD is only fine-tuned on the adversarial data. By adjusting the experimental settings, the clean accuracy can be further improved in exchange for adversarial accuracy. Compared AARD to WETGN, it suggests that the fine-tuned detection model can resist the attack (adversarial accuracy increases from 71.13 to 92.44) while almost not affecting the clean accuracy (from 93.47 to 93.06) on the Twitter15 dataset.

Two examples of the generated adversarial responses that attack successfully are shown in Table 5. In the first example, the source post is a rumor, and PARG alters the prediction by inserting a response "not a false alarm", which conveys a signal that it is actually not a rumor. For the second example, which is a non-rumor, PARG attacks it with a certain attitude "can not believe" to deny that it is a "true" story. Similar responses can also be found in the real response written by human. If a rumor detector only captures simple patterns, it may easily misclassify the above examples and fail to adversarial attacks.

5 Conclusion

In this paper, we propose a novel rumor detection framework, AARD, to improve the vulnerability of detection models, which includes the Weighted-Edge Transformer-Graph Network (WETGN) and the Position-aware Adversarial Response Generator (PARG). Overall evaluation and ablation study results show the effectiveness of the proposed rumor detector on three public datasets. In addition, the adversarial attack results show the benefit of fine-tuning with the adversarial responses generated by PARG. In the future, we plan to further study the model generalization on rumor veracity classification tasks and combine the response stances.

Acknowledgements

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grants MOST-109-2221-E-009-114-MY3, MOST-109-2221-E-001-015, MOST-109-2221-E-009-097, and MOST-109-2218-E-009-016. We are grateful to the National Center for High-performance Computing for computer time and facilities.

References

Sameer Badaskar, Sachin Agarwal, and Shilpa Arora. 2008. Identifying real or fake articles: Towards better language modeling. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 549–556.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nicholas DiFonzo and Prashant Bordia. 2007. Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35.

Katherine Del Giudice. 2010. Crowdsourcing credibility: The impact of audience feedback on web page credibility. *Proceedings of the American Society for Information Science and Technology*, 47(1):1–9.

S Han, J Gao, and F Ciravegna. 2019. Data augmentation for rumor detection using context-sensitive neural language model with large-scale credibility corpus. In *Learning from Limited Labeled Data: ICLR 2019 Workshop*. OpenReview.

Md Saiful Islam, Tonmoy Sarkar, Sazzad Hossain Khan, Abu-Hena Mostofa Kamal, SM Murshid Hasan, Alamgir Kabir, Dalia Yeasmin, Mohammad Ariful Islam, Kamal Ibne Amin Chowdhury, Kazi Selim Anwar, et al. 2020. Covid-19–related infodemic and its impact on public health: A global social media analysis. *The American Journal of Tropical Medicine and Hygiene*, 103(4):1621.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. 2020. Interpretable rumor detection in microblogs by attending to user interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8783–8790.

Sumeet Kumar and Kathleen M Carley. 2019. Tree lstms with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5047–5058.

Quanzhi Li, Qiong Zhang, and Luo Si. 2019. Rumor detection by exploiting user credibility information, attention and multi-task learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1173–1179.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yi-Ju Lu and Cheng-Te Li. 2020. GCAN: Graph-aware co-attention networks for explainable fake news detection on social media. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 505–514, Online. Association for Computational Linguistics.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2019. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. In *The World Wide Web Conference*, pages 3049–3055.

Cyprien de Masson d'Autume, Shakir Mohamed, Mihaela Rosca, and Jack Rae. 2019. Training language gans from scratch. In *Advances in Neural Information Processing Systems*, pages 4300–4311.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 231–240, Melbourne, Australia. Association for Computational Linguistics.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. 2019. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on Twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 647–653, Vancouver, Canada. Association for Computational Linguistics.

Penghui Wei, Nan Xu, and Wenji Mao. 2019. Modeling conversation structure and temporal dynamics for jointly predicting rumor stance and veracity. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4787–4798, Hong Kong, China. Association for Computational Linguistics.

Xiaoyu Yang, Yuefei Lyu, Tian Tian, Yifei Liu, Yudong Liu, and Xi Zhang. 2020. Rumor detection on social media with graph structured adversarial learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, pages 1417–1423.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016. Learning reporting dynamics during breaking news for rumour detection in social media. *arXiv preprint arXiv:1610.07363*.

Appendix

.1 Case Study

Table 6 and Table 7 show more generated examples of our Position-aware Adversarial Response Generator (PARG) for non-rumors and rumors, respectively. We highlight the possible detection signals for the rumor detectors. The detector may learn a better representation, when there are more controversial responses. These examples also show the good generation quality, which may help the noticeability.

Table 6: Generated adversarial examples of non-rumors. The table shows the source post, responses and the generated response for one data. The user names are replaced to remain anonymous.

Source: Videos show suffering of starving in Syrian town of #Madaya. **Response:** @name1 How about videos that show how many are benefiting. People would like to hear that side as well. @name1 DJ Trump and his stormtrumpers say let buddy Putin do whatever. @name1 You know what they r starving people/children as well in my Canadian city I pity the people but I take care of our own @name1 And who pray tell is starving these poor unfortunate people? @name1 If some of these countries truly loved their people.. I think we might be a better world... what a concept eh @name1 This war crime has been going on for months #cnn or do you break news on a need to know bases? @name1 Praying for their relief @name1 So sad her husband got refugee status in Germany and never looked back. @name1 This breaks my heart! @name1 The USA has the same problem. **Generated Response:** @name1 @name2: The world is not a better world.
Source: Breaking: patient being tested for Ebola in Kansas **Response:** @name1 Quick send him to the Westboro baptist church! @name1: Breaking: patient being tested for Ebola in Kansas this shit is getting out of hand @name1 Lock the door to all non-Americans attempting to flee from there to here until this is over. All except Americans!!! @name1 Why not just prevent travel from west Africa before all of this crap started happening @name1 But wait I thought there was no danger and we shouldn't be worried about the US, oh look like everything else that's ... @name1: Patient being tested for Ebola in Kansas sure this is how the walking dead began. @name1 This is getting serious by the minute in Kansas it is spreading fast @name1 You believe the government can handle this go talk to the Katrina survivors @name1: Breaking: patient being tested for Ebola in Kansas oh hell no @name1 How is these people are being infected with Ebola why are these people allowed to travel **Generated Response:** @name1 @name2: The first person to be tested for Ebola in the US.
Source: Elvis Presley was born on this day in 1935 **Response:** @name1 I have been to his birthplace, Tupelo, a week ago... @name1 @name2 What a beautiful voice @name1 @name2 And he's still dead. @name1 @name2 If Elvis was alive today he would be... 81 hundred pounds. @name1 @name2 Wow it's also David Bowie's birthday great day for music lovers? @name1 Elvis Presley.. I salute you the maestro of country music and rock.. Happy birthday and forever you are remembered. **Generated Response:** @name1 @name2 I'm still alive!

Table 7: Generated adversarial examples of rumors. The table shows the source post, responses and the generated response for one data. The user names are replaced to remain anonymous.

Source: The world is running out of chocolate, world's largest chocolate manufacturer warns **Response:** @name1 Let them eat buiscuits!! @name1 Try something else or chocs are gone for good @name1 To quote a great man. 'I don't believe it!' nooo... time to hoard #chocoholic @name1: The world is running out of chocolate @name1 Ukraine president has lots for sale... @name2 This jst sounds like a re-packaged story from the last 3 years. 'blame Asia' - 'blame poor harvest @name1 @name2 nooo! @name1 Where is congress on the really important stuff like this? where do republicans stand? how about the democrats? **Generated Response:** @name1: the world is running out of chocolate?
Source: 'Nine Britons, 23 US citizens and 80 children' feared dead after #MH17 jet 'shot down' [URL] **Response:** @name1 230 + dead in #gaza! Murdered openly by #Israel #Genocideingaza @name1 Shot down-that's not confirmed yet-just speculation-two crashes of a Malaysian airline 777 within 4 months that's a fact @name2 now that's disturbing! @name1 The added horror here will be if no-one does anything about it. If it was a Russian missile Putin will need to answer for it @name1 It is so very tragic and saddening that so many lives were lost, no matter whether they were American or Briton or neither. @name1 Why comment on the US citizen number? Are there lives worth more than other nations? @name1 I bet each one blames the other, but my thoughts are the rebels/Russians may be in line as the suspects. @name1 A tragic day-my deepest condolences and thoughts go to the relatives and loved ones of the 295 people **Generated Response:** @MH17 @name1: A list of the dead in the Malaysia airlines plane shot down.
Source: CDC has confirmed that the patient in Dallas has tested positive for the Ebola virus. We'll have more coming up on Khou 11 news **Response:** @name1 What?! I need to read about this. Is it saying this originated here in the states or is this someone who traveled abroad @name1: CDC has confirmed that the patient in Dallas has tested positive for the Ebola virus. Basically everyone in Texas is dead... RT @name1 CDC has confirmed that the patient in Dallas has tested positive for the Ebola virus. We'll have more... @name3 @name4 @name1 following strict medical guidelines is probably more effective than praying. right about less... @name1 We in the US are safe, we have the needs to treat Ebola at small numbers here. Plus he was traveling from west @name1: CDC has confirmed that the patient in Dallas has tested positive for the Ebola virus. g2g leaving the country @name5 @name1: CDC has confirmed that the patient in Dallas has tested positive for the Ebola virus... **Generated Response:** @name1 @name2: A patient in Texas has tested positive for the Ebola virus.
Source: Scientist releases this horrifying picture of a puppy-sized spider he found in the rainforest [URL][URL] **Response:** @name1 Oh my god @name1 @name2 enjoy @name3 I'd die... @name1: scientist releases pic of puppy-sized spider in rainforest @name1 love of god @name1 I hate spiders @name1 Oh hell no ! ! @name1:... puppy-sized spider he found in the rainforest Ummm @name4 @name1: puppy-sized spider found in the rainforest @name1 @name5 yo don't RT this shit on my tl wtf my nigga why does this exist?? @name1 Paper trained yet? @name1 @name6 that's it, burning down the rainforest. @name1 Does it come when you call it by name? @name1: scientist releases this horrifying picture of a puppy-sized spider @name7. @name1 Remind me to never visit the rainforest. **Generated Response:** @name1 @name2:a spider found in rainforest in the rainforest is not the same thing.

LICHEE: Improving Language Model Pre-training with Multi-grained Tokenization

Weidong Guo[1*], **Mingjun Zhao**[2*], **Lusheng Zhang**[1*], **Di Niu**[2], **Jinwen Luo**[1],
Zhenhua Liu[1], **Zhenyang Li**[1], **Jianbo Tang**[1]
[1]Platform and Content Group, Tencent
[2]University of Alberta
`{weidongguo, lshzhang, jamsluo, edinliu, nickzyli, jianbotang}@tencent.com`,
`{zhao2, dniu}@ualberta.ca`

Abstract

Language model pre-training based on large corpora has achieved tremendous success in terms of constructing enriched contextual representations and has led to significant performance gains on a diverse range of Natural Language Understanding (NLU) tasks. Despite the success, most current pre-trained language models, such as BERT, are trained based on single-grained tokenization, usually with fine-grained characters or sub-words, making it hard for them to learn the precise meaning of coarse-grained words and phrases. In this paper, we propose a simple yet effective pre-training method named *LICHEE* to efficiently incorporate multi-grained information of input text. Our method can be applied to various pre-trained language models and improve their representation capability. Extensive experiments conducted on CLUE and SuperGLUE demonstrate that our method achieves comprehensive improvements on a wide variety of NLU tasks in both Chinese and English with little extra inference cost incurred, and that our best ensemble model achieves the state-of-the-art performance on CLUE benchmark competition.

1 Introduction

Pre-trained language models (PLMs) such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) have become enormously popular and achieved great success on diverse natural language understanding tasks, such as sentiment analysis, question answering, and language inference. These models usually utilize a transformer architecture (Vaswani et al., 2017) to capture the dependencies between tokens in the input text, to model the language information, and to learn contextual representations. It is first pretrained based on large-scale unlabeled corpora, and subsequently fine-tuned based on the labeled data from downstream tasks.

In many NLU applications, tokenization often affects the performance and needs to be chosen carefully. The input tokens for pre-trained language models are usually fine-grained, e.g., words and sub-words for English and characters for Chinese. Compared with coarse-grained tokens such as phrases, the advantage of fine-grained tokens is that they form a smaller vocabulary, yielding abundant training samples per token, and thus alleviating the data sparsity issue and out-of-vocabulary (OOV) problem (Li et al., 2019). However, even trained on large corpora, it is still hard for language models pre-trained with fine-grained tokens to learn the correct attention boundaries of larger semantic units in many languages (Zhang and Li, 2020).

To obtain a more accurate model, prior studies attempt to incorporate coarse-grained information into models trained with fine-grained tokenization by masking sequences of consecutive tokens in the pre-training stage (Joshi et al., 2020; Cui et al., 2019). Zhang and Li (2020) propose AMBERT, a Siamese network based on BERT to handle multi-grained input text, and uses two encoders with shared weights to separately encode fine-grained tokens and coarse-grained tokens into two sequences of contextualized representations. Despite its effectiveness, the inference cost of AMBERT almost doubles that of the original BERT due to the dual-encoder structure, which is often unacceptable in industrial scenarios.

In this paper, we propose a novel method named *LICHEE* designed to efficiently leverage the input information at multiple levels of granularity in the pre-training stage in order to enhance the representation ability of PLMs. Unlike AMBERT that encodes the fine-grained and coarse-grained tokens with two encoders, which significantly increases the inference cost, in *LICHEE* the fusion of multi-

---

[*] Equal contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1383–1392
August 1–6, 2021.

grained information of input text happens at the embedding level, which requires no change on the original model structure of the PLM, and thus induces little extra inference cost when applied in online NLP applications. Specifically, *LICHEE* first pre-processes the input text into fine-grained and coarse-grained tokens, which are passed through two embedding layers, respectively, to derive their corresponding vector representations. Both vector representations are then merged via pooling to form the *multi-grained embedding vector*, which serves as the input to the PLM encoder. Finally, the enhanced contextual representations generated by the PLM encoder, with both fine-grained and coarse-grained information incorporated, are obtained and used for downstream tasks.

We have applied *LICHEE* to enhance multiple different pre-trained language models, including BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019), and GPT (Brown et al., 2020), and conducted extensive evaluation of the resulted language models on Chinese natural language understanding (NLU) tasks evaluated by CLUE (Liang Xu, 2020) benchmarks. Results show that with *LICHEE*, the resulted pre-trained language models significantly outperform their single-grained counterparts on almost all tasks, by taking advantage of multi-grained information to effectively and efficiently produce more accurate representations.

In addition, we also participated in the CLUE benchmark competition with our best ensemble model built upon a collection of *LICHEE*-enhanced BERT-large models, and achieved the state-of-the-art performance of an average score of 80.42 (as of January 8, 2021) over 9 different Chinese NLU tasks, as well as the best scores on two individual tasks: IFLYTEK and CSL.

Moreover, we have also conducted English natural language understanding experiments based on SuperGLUE (Wang et al., 2019a) benchmarks. Significant improvements are observed when *LICHEE* is employed in the pre-training stage, which demonstrates that the proposed pre-training method is generally effective in different language settings.

2 Related Work

In this section, we give a brief overview of some popular pre-trained language models and studies on the training techniques related to tokenization.

Pre-trained language models are pre-trained on large unsupervised corpora and aim to produce meaningful representations for each input token not only considering the meaning of itself, but also with its surrounding contexts anticipated. ELMo (Peters et al., 2018) is one of the first pre-trained language models based on bidirectional LSTMs which produces the contextual representation of each token by concatenating its left-to-right and right-to-left representations. GPTs (Radford et al., 2018, 2019; Brown et al., 2020) leverage the powerful Transformer (Vaswani et al., 2017) to build an auto-regressive language model predicting the next token given its history context. BERT (Devlin et al., 2019) is a bidirectional auto-encoding language model also based on transformer. It consists of two pre-training objectives: masked language model (MLM) and next sentence prediction (NSP). Yang et al. (2019) point out the discrepancy of the pre-training and fine-tuning stage of BERT due to the masking symbol, and propose a permutation language model called XLNet (Yang et al., 2019).

The great popularity of BERT draws many researchers to make improvements on the architecture. RoBERTa (Liu et al., 2019) improves several training details of BERT including dynamic masking and the removal of the NSP pre-training task. ALBERT (Lan et al., 2019) reduces the model parameters with cross-layer weight sharing and accelerates the training process. ELECTRA (Clark et al., 2019) proposes a new token detection task and adopts a generator-discriminator framework to pre-train the language model.

Although most pre-trained language models are built on fine-grained tokenization, coarse-grained information proves to be helpful to the model performance. Cui et al. (2019) propose a masking scheme called “whole word masking” (WWM) for Chinese BERT, where the consecutive characters belonging to the same word are masked together. In ERNIE (Sun et al., 2019), knowledge graphs are added to enhance the model, and entity level masking is used during the pre-training, which is beneficial for language understanding tasks. SpanBERT (Joshi et al., 2020) proposes to mask random spans instead of random tokens, and adopts a new span boundary objective task to replace the next sentence prediction task in the pre-training. Instead of focusing on the masking scheme, AMBERT (Zhang and Li, 2020) proposes to adopt two encoders with shared parameters to learn the representations of fine-grained and coarse-grained tokens in parallel. However, even that the weight sharing setting reduces the number of model parameters, the dual-encoder structure of AMBERT induces twice the inference cost, which remains a huge issue when deployed in online applications.

Different from AMBERT, our work merges the fine-grained and coarse-grained tokenization at embedding level, and achieves significant performance gains with little additional computation costs.

3 Methodology

In this section, we present *LICHEE*, the general multi-grained framework for language model pre-training, and its detailed implementation, including the pre-training methods for both auto-regressive and auto-encoding tasks and fine-tuning details.

3.1 Model Architecture

Figure 1 gives an overview of *LICHEE* where the input information from multiple granularities is leveraged to enhance the representation ability for many pre-trained language models.

The framework takes in text sequences as input which are tokenized into token sequences. In this paper, we keep two vocabularies and use two tokenizers to perform fine-grained and coarse-grained tokenizations, where items in vocabularies are selected based on their token frequencies in pre-training corpora. Also, the definitions of “fine grain” and “coarse grain” vary across languages. For example, in English, words and phrases are often used as the fine-grained and coarse-grained tokens respectively. And in Chinese, characters and words are used instead. Officially, for a given input text sequence T, we use t_i^f to denote the i-th fine-grained token and $t_{j\text{-}k}^c$ to denote a coarse-grained token that is composed of fine-grained tokens $\{t_j^f, ..., t_k^f\}$ between j and k. For example, in figure 1, the coarse-grained token “New York Times” is composed of the first, sencond, and third fine-grained tokens, and is denoted as $t_{1\text{-}3}^c$.

After tokenization, two separate embedding layers are used to map the tokenized tokens to their vector representations. Specifically, each fine-grained token t_i^f is passed into a fine-grained embedding layer to produce the fine-grained embedding vector $\vec{e}_i^f \in \mathcal{R}^d$ of the token, where d denotes the dimension of the fine-grained embedding. Similarly, the coarse-grained embedding $\vec{e}_{j\text{-}k}^c \in \mathcal{R}^d$ is derived with the same dimension d by feeding token $t_{j\text{-}k}^c$ to the coarse-grained embedding layer, shown as:

$$\begin{aligned} \vec{e}_i^f &= embedding_{fine}(t_i^f), \\ \vec{e}_{j\text{-}k}^c &= embedding_{coarse}(t_{j\text{-}k}^c). \end{aligned} \tag{1}$$

For each token t_i^f, we construct its multi-grained embedding vector $\vec{e}_i \in \mathcal{R}^d$ by performing a max-pooling operation on the derived fine-grained embedding $\vec{e}_i^f$ and the coarse-grained embedding $\vec{e}_{j\text{-}k}^c$ of its corresponding coarse-grained token $t_{j\text{-}k}^c$:

$$\vec{e}_i = \textit{max-pool}(\vec{e}_i^f, \vec{e}_{j\text{-}k}^c), \tag{2}$$

where $j \leq i \leq k$. Note that d is equal to the original embedding dimension of the single-grained PLM, to prove that the performance gain is contributed to the introduction of multi-grained information other than modified model structure.

Finally, the combined embedding vectors $\vec{\boldsymbol{e}}$ are fed into the PLM encoder to construct the final contextualized representations $\vec{\boldsymbol{h}}$ enhanced with multi-grained information:

$$\vec{\boldsymbol{h}} = encode(\vec{\boldsymbol{e}}). \tag{3}$$

3.2 Pre-training

We have applied *LICHEE* on both auto-regressive and auto-encoding PLMs, such as GPT and BERT.

For ***auto-regressive PLMs***, the pre-training task is Next Token Prediction which aims to predict the next token t_i based on its previous context $t_{<i}$, by optimizing the following objective function

$$\min_\theta - \sum_i \log p_\theta(t_i | t_{<i}), \tag{4}$$

where the conditional probability p_θ is modeled with a network with parameter θ.

In our framework, we adjust the objective function to include both fine-grained context $t_{<i}^f$ and coarse-grained context $t_{<i}^c$, shown as:

$$\min_\theta - \sum_i \log p_\theta(t_i | t_{<i}^f, t_{<i}^c). \tag{5}$$

Note that when making predictions on any token within a coarse-grained span $t_i \in t_{j\text{-}k}^c$, the token embedding $\vec{e}_i$ will cause information leakage as it involves the coarse-grained token embedding $\vec{e}_{j\text{-}k}^c$ which contains information beyond the history context. For example, in the case illustrated in figure 1, the prediction on token “York” should not rely

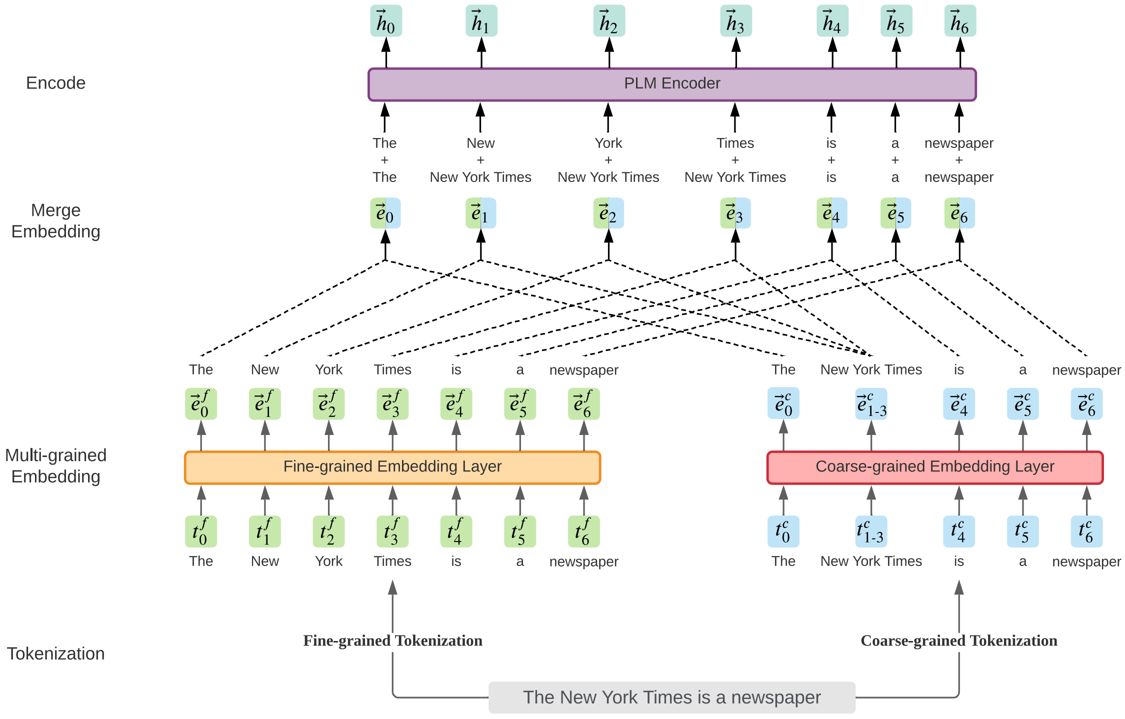

Figure 1: The overall structure of our proposed pre-training framework *LICHEE*. Fine-grained and coarse-grained tokens are first derived from the input text by tokenization, and separately passed into two individual embedding layers. The multi-grained embedding vectors are acquired by taking a max-pooling on the fine-grained and coarse-grained embedding vectors, and are fed into the PLM encoder to extract the final contextualized representations.

on token “New” and its embedding $\vec{e}_1$ as it discloses the entire information of the coarse-grained token of “New York Times” by the coarse-grained embedding $\vec{e}^c_{1\text{-}3}$. Therefore, we can only exploit the context before the start position of the coarse-grained token to make predictions, illustrated as:

$$\min_\theta - \sum_{j \le i \le k} \log p_\theta(t_i | t^f_{<j}, t^c_{<j}), \qquad (6)$$

where j and k are the start and end positions of the coarse-grained token.

For ***auto-encoding PLMs***, we only include Masked Language Modeling (MLM) task in the pre-training process, as Next Sentence Prediction (NSP) task is shown to have no benefits indicated in many recent studies (Lan et al., 2019; Liu et al., 2019; Zhang and Li, 2020). In MLM, 15% of the tokens are randomly selected and substituted with a set of tokens, in which 80% are replaced with *[MASK]* token, 10% are replaced with random tokens, and 10% stay unchanged.

The objective is to recover the masked tokens $T^m \subset T$ from the altered text input sequence $\tilde{T}$:

$$\min_\theta - \sum_{t^m \in T^m} \log p_\theta(t^m | \tilde{T}). \qquad (7)$$

In our framework, we propose to exploit the multi-grained information of the input in the MLM task, shown as:

$$\min_\theta - \sum_{t^m \in T^m} \log p_\theta(t^m | \tilde{T}^f, \tilde{T}^c), \qquad (8)$$

where $\tilde{T}^f$ and $\tilde{T}^c$ stand for the fine-grained and coarse-grained altered input text.

Similar to the strategy deployed in auto-regressive PLMs, we apply a masking strategy that when a fine-grained token t^f_i is to be masked, its corresponding coarse-grained token $t^c_{j\text{-}k}$ and all the fine-grained tokens $t^f_j, ..., t^f_k$ belonging to it are also masked, in order to avoid information leakage from the multi-grained embeddings.

3.3 Fine-tuning

In fine-tuning of downstream tasks, we append the special tokens (*[CLS]*, *[SEP]*) to both fine-grained and coarse-grained vocabularies. In sentence-level classification tasks, *[CLS]* is attached to the start of input sequences in auto-encoding PLMs like BERT, and to the end of the input in auto-regressive PLMs like GPT. Its multi-grained contextualized representation $\vec{h}_{[CLS]}$ is used to represent the whole input sequence and is passed into a projecting layer for the final prediction.

Similarly, for tasks that include token-level span detection, such as Question Answering, the contextual representation $\vec{h}_i$ for each token t_i is extracted and utilized in the task.

4 Experiments

We have carried out extensive experiments on various natural language understanding tasks on both Chinese and English datasets. In the following section, we will first introduce the pre-training datasets used in our evaluation and provide the implementation details of our framework. And we demonstrate the effectiveness of *LICHEE* by conducting comprehensive experiments on various Chinese NLU datasets with multiple different PLMs, and compare our method with other baseline methods. Next, we perform a thorough ablation study to evaluate different approaches of integrating input text information from multiple granularities. Finally, we adopt *LICHEE* to an English BERT to verify its efficacy on English NLU tasks.

4.1 Pre-Training Datasets

For Chinese language, there is no commonly used corpus for pre-training language models. We utilize a large corpus consisting of 450G text from a wide range of popular Chinese applications including Kandian, Zhihu, Wechat, and Weibo, in various fields of news, wiki, and blogs.

Similar to most Chinese PLMs, characters are used as fine-grained tokens due to the language nature of Chinese. For coarse-grained tokens, We use QQSeg which is a segmentation tool with an open API to perform segmentation on text, and the segmented words are treated as coarse-grained tokens. For the construction of vocabularies, we follow Google's Chinese BERT and include $21,128$ tokens in the fine-grained vocabulary. And in the coarse-grained vocabulary, we calculate the token frequencies and trimmed out tokens with frequency lower than 8, resulting in $210,946$ tokens. Note that in order to alleviate the out-of-vocabulary (OOV) problem, all tokens in the fine-tuned vocabulary are also included in coarse-grained vocabulary.

For English, a corpus with 6.2 million documents (18.9G compressed text) from Wikipedia is leveraged to pre-train the model. We first perform sub-word tokenization with BPE algorithm (Sennrich et al., 2015) on the English text, where the produced words and sub-words constitute the fine-grained vocabulary of $28,996$ tokens. In the coarse-grained vocabulary, we treat high-frequency words as coarse-grained tokens, resulting in $136,630$ tokens in total, which also include all tokens in the fine-grained vocabulary for the OOV concern.

4.2 Benchmarks

The evaluation of the pre-trained models is conducted on various downstream NLU tasks. In our experiments, all the Chinese PLMs are evaluated on Chinese Language Understanding Evaluation (CLUE) (Liang Xu, 2020) which is a comprehensive language understanding benchmark developed for Chinese containing 9 natural language understanding tasks. Within the 9 tasks, there are two single-sentence classification tasks that are TNEWS and IFLYTEK, four sentence-pair classification tasks that are AFQMC, OCNLI, CLUEWSC and CSL, and three question answering tasks that are CMRC2018, CHID, and C3. Note that OCNLI has replaced CMNLI since Oct 22, 2020. We compare the model performance by reporting the performance score of each task and the average score of all tasks.

For English tasks, we use the SuperGLUE benchmarks (Wang et al., 2019a) which is an extension of GLUE (Wang et al., 2019b) consisting of a collection of 8 NLU tasks of higher difficulty for comprehensively evaluating the performance of English PLMs. SuperGLEU contains a word sense disambiguation task (WiC), two textual entailment tasks (CB and RTE), two reasoning tasks (COPA and WSC), and three question answering tasks (BoolQ, MultiRC, and ReCoRD).

4.3 Experiment Setup

In order to demonstrate the general applicability and effectiveness of our framework, we have implemented three different pre-trained language models with our method including BERT, ALBERT and GPT, and compare the performances with their corresponding single-grained baseline methods.

For BERT and ALBERT, we follow the "base" structure in (Devlin et al., 2019) with an encoder of 12 layers. And the GPT model in our experiment is also made up of a 12-layer transformer decoder.

Model	Avg.	TNEWS	IFLYTEK	AFQMC	OCNLI	CLUEWSC	CSL	CMRC2018	CHID	C3
	-	acc.	acc.	acc.	acc.	acc.	acc.	EM.	acc.	acc.
BERT	71.12	66.62	60.64	71.74	73.45	72.92	84.01	73.08	75.52	62.08
BERT-*LICHEE*	**73.92**	**67.94**	**60.94**	**73.65**	**75.85**	**81.03**	**84.51**	**75.84**	**77.65**	**67.84**
ALBERT	67.27	64.45	57.54	**71.35**	69.19	68.80	83.00	68.06	68.97	54.04
ALBERT-*LICHEE*	**69.30**	**66.31**	**58.29**	70.95	**71.05**	**70.39**	**83.31**	**72.87**	**71.93**	**58.65**
GPT	67.41	67.52	60.84	69.83	70.91	63.76	83.12	62.53	73.31	54.84
GPT-*LICHEE*	**68.73**	**68.40**	**61.06**	**70.00**	**72.01**	**66.01**	**83.23**	**64.57**	**74.02**	**59.27**

Table 1: Comparison of the model performances on the CLUE tasks. BERT-*LICHEE*, ALBERT-*LICHEE* and GPT-*LICHEE* stand for the multi-grained version of the model with our method incorporated. The average score of the nine CLUE tasks are also given.

Model	Avg.	TNEWS	IFLYTEK	AFQMC	OCNLI	CLUEWSC	CSL	CMRC2018	CHID	C3
	-	acc.	acc.	acc.	acc.	acc.	acc.	EM.	acc.	acc.
Archer-24E-SINGLE	79.19	69.54	62.27	**77.26**	**83.57**	90.00	85.73	75.65	85.66	**83.04**
roberta_selfrun	79.46	69.10	63.92	76.09	80.40	**93.10**	87.27	79.20	88.80	77.29
UER-ensemble	79.64	**72.20**	64.00	76.82	80.80	90.35	85.83	79.15	86.03	81.60
BERTs	79.66	69.94	63.92	76.77	82.09	88.97	86.77	**80.50**	**89.51**	78.44
LICHEE-ensemble	**80.06**	70.50	**64.15**	76.98	81.30	90.69	**87.40**	79.80	87.51	82.22

Table 2: Top-5 models on the CLUE benchmark leaderboard where our ensemble model achieves the state-of-the-art performance on the averaged CLUE score. These results are grabbed from the official CLUE website[1]on Jan 8, 2021.

Then, we apply the following training setting to the training process of all three models. For better scalability in large batch, we adopt LAMB (You et al., 2019) to replace Adam (Kingma and Ba, 2014) as the optimizer with a batch size of 768 and a learning rate of $2e-4$. We first train the model for 1M steps using 128 as the maximum sequence length, and increase the maximum length to 512 for another 100k steps, for better capturing the long distance dependencies. To enhance the training efficiency, we adopt mix-precision training technique (Micikevicius et al., 2017) during pre-training, which are performed on 4 Nvidia V100 gpus.

We have also implemented a *LICHEE*-enhanced ensemble model based on BERT-large to participate in the CLUE benchmark competition. During training, we adapt the batch size to $1,024$ and the maximum sequence lengths at the first and second stage are set to 256 and 512. And 64 Nvidia V100 gpus are used to train the model.

For the evaluation of each task, we derive 6 results with different random seeds and report the average performance in this paper.

4.4 Main Results

In table 1, we adopt our multi-grained pre-training method on three pre-trained language models: BERT, ALBERT, and GPT, and compare them with their single-grained baselines on CLUE benchmark. From the results, we can see that our method achieves significant performance gains by exploiting the multi-grained information of the text input. The averaged CLUE scores of our multi-grained BERT-*LICHEE*, ALBERT-*LICHEE* and GPT-*LICHEE* are 73.92, 69.30 and 68.73 respectively, producing significant absolute improvements of 2.80, 2.03, and 1.32 compared to their single-grained baseline models. Aside from the improvement on the averaged CLUE score, it is also worth to mention that our multi-grained BERT-*LICHEE* and GPT-*LICHEE* outperforms their single-grained baselines on all 9 NLU tasks in CLUE, while the ALBERT-*LICHEE* model also beat the single-grained ALBERT in 8 out of 9 tasks, which provides strong evidence that the benefits of our method are generally applicable to different pre-trained language models and diverse NLU tasks.

In order to further investigate the potential of *LICHEE*, we apply it on an ensemble model based on BERT-large and participate in the CLUE benchmark competition. As demonstrated in table 2, our method outperforms all other candidates on the average score of 9 CLUE tasks by a significant margin, and also achieves the state-of-the-art performance on two individual NLU tasks of IFLY-

---

[1] https://www.cluebenchmarks.com/rank.html

Model (BERT)	**Avg.** -	**TNEWS** acc.	**IFLYTEK** acc.	**AFQMC** acc.	**OCNLI** acc.	**CLUEWSC** acc.	**CSL** acc.	**CMRC2018** EM.	**CHID** acc.	**C3** acc.
SG	71.12	66.62	60.64	71.74	73.45	72.92	84.01	73.08	75.52	62.08
SG (WWM)	72.24	66.87	60.55	72.62	74.41	74.07	84.12	75.22	77.74	64.60
MG (CAT 384+384)	72.86	**68.11**	61.09	72.33	75.08	75.26	84.48	75.35	77.84	66.17
MG (CAT 256+512)	72.94	67.63	**61.55**	71.96	74.97	76.54	84.16	75.31	78.17	66.15
MG (CAT 512+256)	73.08	67.88	61.06	73.07	75.84	74.45	**84.74**	74.44	**78.29**	**67.91**
MG (MEAN)	73.22	67.85	60.99	73.44	**75.97**	76.31	84.52	75.54	77.84	66.53
LICHEE	**73.92**	67.94	60.94	**73.65**	75.85	**81.03**	84.51	**75.84**	77.65	67.84

Table 3: Ablation study of different pre-training strategies with BERT model on CLUE dataset. Two single-grained (SG) baselines and five multi-grained (MG) methods (*LICHEE* and its variants) with different ways of integrating the fine-grained and coarse-grained representations are evaluated.

TEK and CSL. This results further proves that our multi-grained pre-training method is able to bring significant improvements on the representation ability of language models and is generally effective to a wide range of downstream NLU tasks.

The reason of *LICHEE*'s success is that we adopt a multi-grained pre-training strategy to model the contextual information of the input text to leverage the advantages from both granularities, where fine-grained token representations are easier to learn considering the sufficient training samples, and coarse-grained tokens are more complete as lexical units and provide more accurate contextual information. Furthermore, in our framework, the combination of the multi-grained information is realized on the embedding level so that we can keep the model structure unaltered, showing that the benefits are achieved entirely through the information gains caused by multi-grained pre-training other than model-level modifications.

4.5 Ablation Analysis

We have conducted ablation analysis on CLUE benchmarks with BERT, to evaluate the impact of our multi-grained design, as well as perform a comprehensive study on the different methods of integrating the multi-grained embedding. Table 3 lists the performance of model variants with different training strategies, including two single-grained methods and five multi-grained methods.

The original single-grained BERT whose masking scheme is solely based on fine-grained tokens gives an average CLUE score of 71.12. The Whole Word Masking (WWM) technique (Cui et al., 2019) performs masking operations on continuous fine-grained tokens that form a coarse-grained token and improves the performance to 72.24. Note that although WWM utilizes coarse-grained token boundary information during the masking operations, it does not explicitly train representations for coarse-grained tokens. Therefore, we treat WWM also as a single-grained pre-training method.

For multi-grained pre-training methods, we have conducted experiments to explore five different approaches of combining embedding representations of fine-grained and coarse-grained tokens, including concatenating the embedding vectors with different dimension settings, and integrating them with mean-pooling and max-pooling. For the concatenation approaches, we keep the dimension of the concatenated multi-grained embeddings to 768 to align with the baseline models, and apply three settings to adjust the dimensions of fine-grained and coarse-grained embedding correspondingly to (384, 384), (256, 512) and (512, 256). Empirically, we discover that the three concatenation settings achieve similar performances, while having larger embedding vectors for fine-grained tokens and smaller embedding vectors for coarse-grained tokens produces a slightly better performance of 73.08 average CLUE score.

Exploiting mean-pooling to integrate the multi-grained information gives more performance gains compared with concatenation methods and reaches 73.22 average CLUE score, which may be attributed to the greater number of embedding parameters, as pooling methods do not require a shrink on the embedding dimension and allow both fine-grained and coarse-grained embedding dimension to stay 768. Finally, *LICHEE* with the max-pooling incorporated outperforms all the fore-mentioned approaches, attains an overall score of 73.92, and achieves the best score on 3 out of 9 CLUE tasks, due to its capability of extracting more representative features. Especially for the task of CLUEWSC, *LICHEE* acquires an accuracy of 81.03 while the second best method only reaches 76.54. We believe this is because the small training set of CLUEWSC

Model	**Avg.** -	**BoolQ** acc.	**CB** acc.	**COPA** acc.	**MultiRC** EM.	**ReCoRD** EM.	**RTE** acc.	**WiC** acc.	**WSC** acc.
BERT-WWM	63.64	77.13	79.76	62.83	24.88	65.20	70.88	64.50	63.94
BERT-*LICHEE*	**65.53**	**77.98**	**88.21**	**63.00**	**25.41**	**67.50**	**71.91**	**65.45**	**64.81**

Table 4: Comparison between our multi-grained BERT-*LICHEE* and the single-grained BERT-WWM on SuperGLUE tasks.

Model	FLOPs	Speedup
BERT	43.5B	1.0x
AMBERT	87.0B	0.5x
LICHEE	43.5B	1.0x

Table 5: Comparison of FLOPs and speedup among the single-grained BERT, AMBERT, and our method.

with only 532 examples makes it more dependent on powerful pre-trained representations, so that the advantage of the max-pooling method is amplified.

Overall, we can see from table 3 that all multi-grained pre-training methods outperform the single-grained baselines by a significant margin, which again proves that our idea of incorporating multi-grained information during the pre-training phase is efficacious and can benefit model performance considerably.

4.6 Inference Speed Analysis

We have also studied the inference speed of *LICHEE* and compare it with the original single-grained BERT and another multi-grained method AMBERT.

Table 5 gives a brief comparison in terms of FLOPs and speedup, tested on a binary classification task with 512 sequence length. FLOPs indicates the number of floating-point operations that the model performs for a single process, where generally speaking, the higher the model's FLOPs is, the slower the inference speed will be.

We can see that the FLOPs of the AMBERT is 87.0 billion, twice the number of the single-grained BERT. It means the inference time of AMBERT is almost doubled, which can cost a lot more time and resources, and often can be unacceptable for real-world applications. Meanwhile, our multi-grained method produces a model with 43.5 billion FLOPs with a negligible increase compared with the single-grained baseline, because the additional operations only include an embedding lookup operation for coarse-grained tokens and a max-pooling operation to integrate the fine-grained and coarse-grained embedding vectors. In summary, *LICHEE* can produce significant performance gains with negligible extra inference time needed.

4.7 English Tasks

We have also conducted experiments on SuperGLUE benchmarks to evaluate *LICHEE* on English language tasks, and compared it with the single-grained baseline: BERT-WWM (Cui et al., 2019).

As shown in table 4, the BERT model pre-trained with our multi-grained method outperforms the single-grained BERT-WWM on all 8 SuperGLUE tasks, and attains an average score of 65.53 surpassing the baseline by 1.89. This improvement over BERT-WWM demonstrates that the effectiveness of *LICHEE* is attributed greatly to the information gain of its multi-grained representations, more than just token boundary information. We also notice that, similar to the CLUEWSC task, a huge increase of 8.45 on accuracy is achieved for the CB dataset of 250 training samples, because our pre-training method leverages the information gains of multi-grained tokens and produces more accurate representations, which is especially effective on tasks with small training data.

This result evidently illustrates that *LICHEE* is not only effective on tasks of character based language like Chinese that highly relies on correct tokenizations, but can also produce significant improvements on languages that are naturally tokenized such as English.

5 Conclusion

In this paper, we have proposed a novel multi-grained method for language model pre-training named *LICHEE*, which can be applied to both auto-regressive and auto-encoding PLMs. In our method, the fine-grained embeddings and the coarse-grained embeddings are separately learned and integrated as the multi-grained embeddings, which is then passed into the encoder of the language model. Experiments show that *LICHEE* can significantly enhance the model performance by a great margin on downstream tasks of both Chinese and English, and significantly improve the inference speed compared to the prior multi-grained method.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is word segmentation necessary for deep learning of chinese representations? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3242–3252.

Lu Li Hai Hu Chenjie Cao Weitang Liu Junyi Li Yudong Li Kai Sun Yechen Xu Yiming Cui Cong Yu Qianqian Dong Yin Tian Dian Yu Bo Shi Jun Zeng Rongzhao Wang Weijian Xie Yanting Li Yina Patterson Zuoyu Tian Yiwen Zhang He Zhou Shaoweihua Liu Qipeng Zhao Cong Yue Xinrui Zhang Zhengliang Yang Zhenzhong Lan Liang Xu, Xuanwei Zhang. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in neural information processing systems*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019b. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.

Xinsong Zhang and Hang Li. 2020. Ambert: A pre-trained language model with multi-grained tokenization. *arXiv preprint arXiv:2008.11869*.

Detecting Hallucinated Content in Conditional Neural Sequence Generation

Chunting Zhou[1][*]; Graham Neubig[1], Jiatao Gu[2], Mona Diab[2], Francisco Guzmán[2],
Luke Zettlemoyer[2], Marjan Ghazvininejad[2]
Language Technologies Institute, Carnegie Mellon University[1]
Facebook AI Research[2]
`{chuntinz,gneubig}@cs.cmu.edu,`
`{jgu,mdiab,fguzman,lsz,ghazvini}@fb.com`

Abstract

Neural sequence models can generate highly fluent sentences, but recent studies have also shown that they are also prone to hallucinate additional content not supported by the input. These variety of fluent but wrong outputs are particularly problematic, as it will not be possible for users to tell they are being presented incorrect content. To detect these errors, we propose a task to predict whether each token in the output sequence is hallucinated (not contained in the input) and collect new manually annotated evaluation sets for this task. We also introduce a method for learning to detect hallucinations using pretrained language models fine tuned on synthetic data that includes automatically inserted hallucinations. Experiments on machine translation (MT) and abstractive summarization demonstrate that our proposed approach consistently outperforms strong baselines on all benchmark datasets. We further demonstrate how to use the token-level hallucination labels to define a fine-grained loss over the target sequence in low-resource MT and achieve significant improvements over strong baseline methods. We also apply our method to word-level quality estimation for MT and show its effectiveness in both supervised and unsupervised settings [1].

1 Introduction

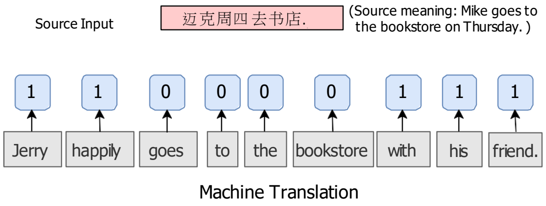

Figure 1: An example of token-level hallucination detection from MT. The grey box is an example of MT output and the labels above indicate if each word is faithful (0) to the input or hallucinated (1).

Neural sequence models for tasks such as data-to-text generation (Puduppully et al., 2019), machine translation (MT; Vaswani et al. (2017); Wu et al. (2016)) and text summarization (Rothe et al., 2020) can often generate fluent text that is sometimes *preferred* to human-written content (Läubli et al., 2018; Brown et al., 2020). However, they also often generate texts that lack global logical consistency (Marcus and Davis, 2020), are dull and repetitive (Welleck et al., 2019), or contain hallucinated content that is not entailed by the input (Maynez et al., 2020; Martindale et al., 2019). In this paper, we focus on tackling the latter problem, aiming to automatically identify and quantify content in the output that is not faithful to the input text.

The risk of generating unfaithful content impedes the safe deployment of neural sequence generation models. The first step to building models that do not suffer from these failures is the assessment and identification of such hallucinated outputs. Prior work has shown that standard metrics used for text evaluation, such as BLEU scores (Papineni et al., 2002; Post, 2018), ROUGE (Lin and Hovy, 2004) and BERTScore (Zhang et al., 2019), do not correlate well with the faithfulness of model outputs (Maynez et al., 2020; Wang and Sennrich, 2020; Tian et al., 2019). They also require reference output text, limiting their applicability in a deployed system at run-time. Very recent efforts have started to develop automatic metrics to measure the faithfulness of output sequences using external semantic models, e.g. the question-generation and question-answering systems (Wang et al., 2020a; Durmus et al., 2020) or textual entailment inference models (Maynez et al., 2020), to score faithfulness tailored for abstractive text summarization. However, these scores do not directly identify hal-

---

[*] Most work was done during an internship at FAIR.

[1] Codes and data available at `https://github.com/violet-zct/fairseq-detect-hallucination`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1393–1404
August 1–6, 2021. ©2021 Association for Computational Linguistics

lucinated tokens and only correlate weakly with human judgements.

We propose a new task for faithfulness assessment - hallucination detection at the token level, which aims to predict if each token in the machine output is hallucinated or faithful to the source input. This task does not use the reference output to assess faithfulness, which offers us the ability to also apply it at run-time. Similar to the spirit of our proposed task, word-level quality estimation (Specia et al., 2018; Fonseca et al., 2019) in the MT community predicts if tokens are correctly translated based on human post-editing. However, these methods generally do not distinguish errors in terms of fluency and adequacy (Specia et al., 2011), with the exception of a subset of the WMT 2020 shared task on quality estimation (Specia et al., 2020), where different types and levels of severity of word-level errors are defined. Our proposed task specifically focuses on hallucination errors, and we define these errors in a simpler way with only binary labels, which we argue makes them simpler to use and more conducive to labeling at large scale. The proposed hallucination detection method (described below) is also applicable to the word-level quality estimation task as demonstrated in §5.4.

We measure hallucination for two conditional sequence generation tasks – abstractive summarization and MT. For the former, we produce a benchmark dataset from recently released annotations (Maynez et al., 2020). For MT, we carefully design human assessment guidelines and create high-quality annotations, which will be released to aid future research. To learn token-level hallucination prediction for general conditional sequence generations tasks, we propose a novel method that creates synthetic "hallucinated" data and finetunes a pretrained language model (Liu et al., 2019; Conneau et al., 2020) on it. Without any human annotated supervised training data, we achieve an average F1 of around 0.6 across all the benchmark datasets, setting initial performance levels for this new task.

Predicting hallucination labels at the token level provides a tool for diagnosing and interpreting model outputs, which allows us to flag potential risks when the model is applied to previously unseen inputs. Additionally, we show how to use these token-level hallucination labels in two case studies to improve self-training (Scudder, 1965) and learning from noisy mined bitext (Koehn et al., 2019) in low-resource MT. In both cases, there can be noise in the target text, either produced by the self-training teacher or errors in the mining process. However, most outputs are only partially erroneous (see examples in Appendix E.3) and the rest of the output is still useful for training, as we show by introducing different token-level loss truncation schemes that use our proposed hallucination detection methods. Our best methods outperform strong baselines by a large margin, and reduce the number of hallucinations.

2 Token-level Hallucination Prediction

For source sequence S and generated output sequence G, following Maynez et al. (2020) we define any span $g_i, \cdots, g_{i+j}$ $(j >= 0)$ in G as being "hallucinated" if it is not supported by the source input S.[2] More specifically, we consider two types of hallucination, which are not mutually exclusive:

Extrinsic hallucinations: a span $g_i, \cdots, g_{i+j}$ in G consists of additional content without clear grounding in the input. In Fig. 1, the word *"happily"* in the machine translation belongs to this case, as there is no word in the input sentence that clearly corresponds to "happy".

Intrinsic hallucinations: a span of word(s) in G contains incorrect information due to synthesizing content using information present in S. In Fig. 1, *"Jerry"* in the MT is a hallucinated word and should be replaced by *"Mike"*. Note that multi-word phrases can also be marked intrinsic hallucinations, such as "this is a book" being hallucinated from "this is not a book", where "this is" is a minimal span corresponding to the hallucination.

The above definitions are for illustrative purposes; we do not explicitly label whether a hallucination is intrinsic or extrinsic, only whether one exists at all. Given these spans, we aim to identify all the span(s) satisfying the above conditions in machine generation G.[3]

Human Assessment of Hallucinations To facilitate the assessment of hallucinations in MT, we conduct human annotations on outputs of MT models in the patent and COVID-19 domain. Three bilingual annotators were presented the source sentence, the reference sentence and the MT output, and they were asked to label each sentence with

---

[2] Content that is paraphrased or can otherwise be inferred by the source document is not considered hallucinated.

[3] We do not annotate under-generations e.g. the source input is only partially translated or summarized.

one of the three types of labels: incomprehensible, faithful, and contains hallucinations. If the translation contains hallucinations, we asked the annotators to tag all the tokens that were not faithful to the source. The final benchmark datasets were created by taking majority labels among three annotators. We present more details regarding annotation guidelines and pipelines in Appendix A.

We compute the Fleiss's Kappa (Fleiss, 1971) (FK) scores of our annotations for MT and the processed annotations from (Maynez et al., 2020) on abstractive summarization (Tab. 5 in Appendix A). We achieved moderate agreement (FK≈0.56) on the token-level hallucination annotations and substantial agreement (FK≈0.67) on the sentence-level annotations, while Maynez et al. (2020) achieved substantial or almost perfect agreement (FK≈0.8) on the XSUM dataset. For MT, we conjecture that it is relatively hard to achieve consistent agreement among annotators for several reasons. First, although we have made detailed annotation guidelines following the definition of hallucination in § 2, it could still be difficult for annotators to distinguish between ungrammatical translations and hallucinations. Second, it was sometimes difficult for annotators to understand the specialized text in the patent domain.

3 Token-level Hallucination Detection

We propose a general-purpose method for token-level hallucination detection for conditional sequence generation tasks. Given the source input S, we first formulate the task of token-level hallucination detection as a sequence labeling problem where a binary label is predicted at each position G_t of the machine generation G. One straightforward way of learning this task is to train a model with supervised data in the form of $((S, G), L_G)$ where L_G are the labels at every position of G that indicate if each word is a hallucinated one or not. However, because such labeled training data is not readily available, we propose an approach to automatically create synthetic training data.

3.1 Synthetic Data Creation

We use bi-text from the training data to create synthetic examples by automatically inserting new, hallucinated target-side tokens. More specifically, we take target sequence T and create a hallucinated version of it denoted T' with associated hallucination labels for each token in T'. Then we can train a supervised model on this synthetic labeled data set of $((S, T'), L_{T'})$. The key challenge is that T' should be a fluent sentence that does not differ too much from T.

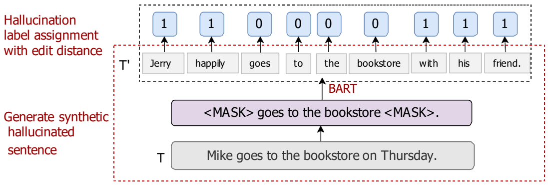

Figure 2: Generation of synthetic data with hallucination labels. A hallucinated version of T is generated by feeding the noised sentence to the encoder-decoder model BART. Hallucination labels are assigned to each token by computing the edit distance between T' and T. Labels of 1 refer to hallucinated words.

Generation of hallucinated sentences To control this synthetic hallucination process, we build on a pre-trained denoising autoencoder, which maps a corrupted sentence back to the original text it was derived from, learning to reconstruct missing words that have been arbitrarily masked out. Specifically, we use the BART model (Lewis et al., 2020), without providing it any access to the source sentence, thereby encouraging it to insert new content as needed to ensure fluency. As shown in Fig. 2, we first apply a noising function that removes words from the original target sentence T[4] and then use a pretrained BART to generate T' conditioned on the noised T with beam search.

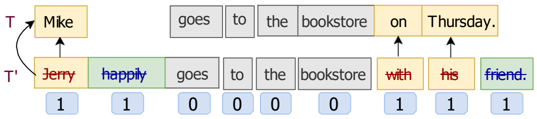

Figure 3: An example of label assignment.

Label assignments After obtaining the hallucinated sentence T' with BART, we need to assign appropriate labels to each token in T' to mark which words are hallucinated. We compute the edit distance between T' and T, and back-trace the deletion and substitution operations with dynamic programming. All the positions in T' involving these two operations are labeled as hallucinations and everything else is considered faithful to T. Fig. 3 shows an example of label assignment with edit distance, where words in red are replaced and words in blue are deleted to convert T' to T. Assigning labels with edit-distance can not always guarantee correct labels, but we find that this simple approach

---

[4] We also applied other noising functions, please see §5.1

provides sufficiently high quality training data for effective hallucination detection in practice.

3.2 Finetuning on Synthetic Data

Hallucination prediction loss We follow the common practice in natural language understanding (NLU) tasks and finetune a pretrained language model (LM) on our synthetic data. We finetune a cross-lingual LM (Conneau et al., 2020) for MT and a monolingual LM (Liu et al., 2019) for summarization. In both cases, we concatenate the input, true target and hallucinated target denoted (S, T, T') as a single input sequence to the model. Then we minimize the standard classification loss $\mathcal{L}_{pred}$ over the pseudo hallucination labels $L_{T'}$ on top of the final hidden vectors of each token in T' as shown in Fig. 4.

Although using only the source text and hallucinated target (S, T') as the input should be sufficient to learn to predict hallucinations, we can also easily measure the extent to which including the true target T in the input could help the model. At test time, when evaluating the faithfulness of the machine outputs G, we do not use the true target T and perhaps surprisingly find our model can generalize well without references, even when they were present during training.

To prevent the model from overly relying on the true target T and learning spurious correlations (e.g. the edit distance), we explored two techniques: (1) *dropout* – randomly drop out tokens in T to force the dependence on the source input; (2) *paraphrase* – recall that at synthetic data generation time, we generate T' from BART conditioned on the noised T. Instead, we can apply noise functions to the paraphrased sentence of T. We create paraphrased targets via knowledge distillation (Kim and Rush, 2016) where we use the output from pretrained Seq2Seq model conditioned on the source sentence in the bi-text corpus as the paraphrased target. Let D denote the paraphrased sentence of T and D' denote the generation from BART conditioned on the noised D. Then we create pseudo labels of D' denoted $L_{D'}$ by computing the edit-distance between the D' and D and use $((S, T, D'), L_{D'})$ as the training data for finetuning. Since the pseudo labels are created based on D, it can prevent the model from learning the edit-distance between T and D' easily. We provide ablation studies in Appendix D.

Masked LM loss We also add the masked language model loss (MLM) $\mathcal{L}_{mlm}$ following (Devlin et al., 2019). To learn this loss, we create a different batch from the above by concatenating only the source S and target T as the input, since the hallucinated target T' could provide erroneous information for predicting masked words in T. We find that such multi-task learning objective helps learn better representations of the input and further improves performance on predicting hallucination labels. The final loss is $\mathcal{L} = \mathcal{L}_{pred} + \alpha \cdot \mathcal{L}_{mlm}$ where α is a hyperparameter.

4 Evaluation Tasks and Data

We examine hallucination in abstractive text summarization and machine translation (MT) tasks, using the models and datasets described below.

4.1 Abstractive Text Summarization

Maynez et al. (2020) studied hallucination problems in extreme summarization on the XSUM dataset which comprises 226,711 British Broadcasting Corporation (BBC) articles paired with their single-sentence summaries. They randomly sampled 500 articles from the XSUM test set and evaluated summaries from four abstractive summarization systems: **PtGen** (See et al., 2017), **TConvS2S** (Narayan et al., 2018), **TranS2S** (Vaswani et al., 2017) and **BERTS2S** (Rothe et al., 2020). Maynez et al. (2020) asked human annotators to label the spans in the machine generated summaries if they were unfaithful to the article. We post-processed their human annotations by majority voting and created test datasets for each of the summarization systems.

4.2 MT

Previous work (Wang and Sennrich, 2020; Müller et al., 2019; Koehn and Knowles, 2017) has shown that translation models are particularly prone to hallucination when tested out of domain. We similarly focus on this regime and additionally consider the low resource case where a modest amount of out of domain data is available at training time.

Data We use a multi-domain Chinese-English (Zh-En) translation dataset (Wang et al., 2020b) which consists of four balanced domains: *law, news, patent* and *subtitles*. We create a new training data $\mathcal{D}_{train}$ with *law* (1.46M sentences), *news* (1.54M), *subtitles* (1.77M) train data and randomly sample 870 parallel sentences from the *patent* training data. We train two NMT models (described below) on this dataset and test on 150 examples from

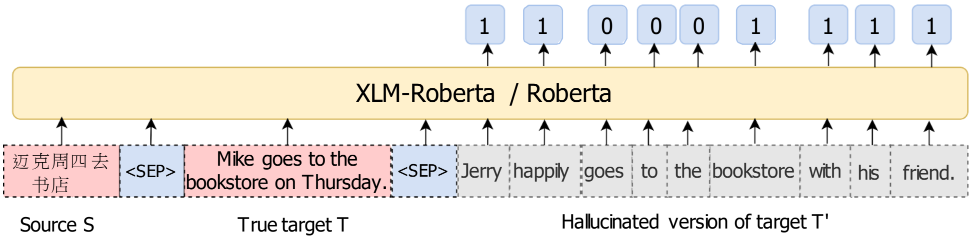

Figure 4: Finetuning XLM-Roberta (for cross-lingual generation task, e.g. MT) or Roberta (for monolingual generation task, e.g. text summarization) on the synthetic training data.

Methods	**MT**		**Summarization**			
	TranS2S	**MBART**	**PtGen**	**TConvS2S**	**TranS2S**	**BERTS2S**
Alignment	29.47	9.93	38.92	37.94	34.47	35.81
Overlap-based	9.14	3.24	57.22	54.25	53.79	55.13
Synonym-based	–	–	59.54	63.73	58.66	53.07
Ours (w/o reference)	**65.75**	**41.92**	63.66	65.94	61.70	55.45
Ours (w/o reference + synonym)	–	–	**64.72**	**69.37**	**63.88**	**56.49**
Ours (w/ reference)	66.08	46.81	63.89	66.28	62.24	55.88

Table 1: F1 (x100) of hallucination labels on MT (seesection 4.2) and abstractive summarization (XSUM). The first block are baseline methods and the second block are our results. Bold indicates best results not using references.

the patent test data. In addition, we also test the NMT models on the COVID-19 domain, sampling 100 examples from the dataset of Anastasopoulos et al. (2020). We denote this 250-sentence dataset as $\mathcal{D}_{eval}$ and ask human annotators to evaluate the level of hallucinations thereof.

Models Our data is generated from two models on which we will measure hallucination (see Appendix B for more details): (1) **TranS2S** (Vaswani et al., 2017) is the standard Transformer Seq2Seq model with 6 encoder layers and 6 decoder layers. (2) **MBART** (Liu et al., 2020) is a Seq2Seq denoising auto-encoder pretrained on large-scale monolingual corpora in many languages. We finetune the 12 layer model on $\mathcal{D}_{train}$.

5 Experiments

5.1 Experimental setup

Synthetic Data Generation We use a pretrained 12 layer BART (Lewis et al., 2020) model in the fairseq toolkit (Ott et al., 2019) for synthetic labeled data generation. We uniformly sample the percentage of tokens p_m to mask from $[0, h_m]$ for each sentence. We also uniformly sample the probability of replacing a token with a random token from $[0, h_r]$ denoted p_r. p_m and p_r are two important factors that affect the noise level when generating the synthetic data. For MT, we set h_m and h_r to 0.6 and 0.3 respectively. For abstractive summarization, we use 0.4 and 0.2. We use beam search for decoding from BART with beam size of 4 and length penalty of 3. For MT, we first create paraphrased target sentences D' through knowledge distillation (Kim and Rush, 2016) by using the outputs from the same trained TranS2S model on the source inputs.

Hallucination Predictor For MT, we finetune XLM-R (Conneau et al., 2020) on the synthetic dataset with batch size of 128, and we annotated 50 examples (different from those in $\mathcal{D}_{eval}$) from the patent test data as the validation dataset. For summarization, we finetune RoBERTa (Liu et al., 2019) with batch size of 96 and early stop training with 10K update steps. In addition, we dropout tokens from the reference T in the input with a rate of 0.5 and 0.3 respectively for summarization and MT to learn $\mathcal{L}_{pred}$. We set α to be 0.6 for MT and 0.5 for summarization based on the scales of $\mathcal{L}_{pred}$ and $\mathcal{L}_{mlm}$. For both tasks, we set the mask probability used for $\mathcal{L}_{mlm}$ to be 0.5, and the initial learning rate to be $2e-5$ with polynomial decay. We describe other hyperparameters, including training of MT models, in the Appendix B and C.

5.2 Evaluation of hallucination prediction

In Tab. 1, we present the F1 of token-level hallucination labels across six benchmark datasets for MT and abstractive summarization (full results of precision, recall and F1 are presented in Tabs. 7 and 9 in the appendix). We compare with three baseline methods that we proposed for this new task: **(1)** The **alignment-based** method uses a word alignment model for hallucination assessment. We em-

Methods	MT		Summarization			
	TranS2S	**MBART**	**PtGen**	**TConvS2S**	**TranS2S**	**BERTS2S**
True hal. tokens (%)	18.12	11.10	46.09	52.89	46.74	37.51
Pred hal. tokens (%)	18.56	7.99	57.22	57.68	55.78	48.84

Table 2: Annotated (True) and predicted (Pred) percentage of hallucinated tokens on benchmark test sets.

ploy SimAlign (Sabet et al., 2020), an unsupervised aligner, that extracts word alignments from similarity matrices induced from pretrained word embeddings. SimAlign is essentially used for crosslingual tasks, and we adapt it to summarization by using embeddings from the pretrained BERT-large (Devlin et al., 2019). We predict a target token as being hallucinated if it is not aligned to the source tokens. **(2)** The **overlap-based** method is a heuristic one that predicts a target token as being hallucinated if does not appear in the source. Since it's not feasible to perform string matching between two languages for MT, we use a bilingual lexicon induction method (Zhou et al., 2019) to first translate each English word into a Chinese word and then check its existence in the source text. **(3)** We go further by exploiting **synonyms** to assess hallucination in the summarization task where we use WordNet (Miller, 1998) to find synonyms of nouns, verbs, adjectives and adverbs of the target summary and the source article; we predict a target as being hallucinated if its synonym can not be found in the set of the source synonyms.

From Tab. 1, we note: **(1)** The proposed method achieves decent performance on this task and ranks the best among all baseline methods. However the task is still far from being solved is worthy of study in the future. **(2)** We can see that even though our model learns hallucination prediction with reference T during training (Sec. 3.2), by applying token dropout to T, our model generalizes well without feeding the reference at test time. As a contrast, we report the results of predicting with reference at test time and observe that the model can achieve a significantly higher recall but worse precision (Tab. 9 in appendix). **(3)** The two non-neural baselines we proposed work surprisingly well on the summarization datasets, especially the synonym-based system. We guess this is because the information of the summaries should come from the source article and a majority of hallucinated words are nouns (§5.3) which can be easily detected by string matching or synonym matching. Our neural system performs better than these baseline methods but not significantly, and we hypothesize that this is because the RoBERTa model we finetune on only allows a maximum input length of 512, which results in an average cutoff of 158 subwords from the source article and hence loss of source information. By taking the union of the predictions from the synonym-based and our models, we can further obtain improvements on the summarization datasets. We believe the advances in long sequence modeling (Beltagy et al., 2020; Kitaev et al., 2020) could help here, and are important to study in future work. **(4)** At the same time, the baseline methods can not obtain reasonable performance for MT since crosslingual semantic matching is more challenging and our model shows significant improvements.

In Tab. 2, we show the percentage of annotated and model predicted hallucinated tokens across the six benchmark sets. We can see that model predictions correlate well with human assessment and have a Pearson correlation coefficient of 0.986.

5.3 Analysis

Analysis on Pretrained Models for Conditional Sequence Generation Recent work (Maynez et al., 2020) has shown that pretrained models are better at generating faithful summaries as evaluated by humans. In Tab. 2, summaries generated from BERTS2S contain significantly fewer hallucinations than other model outputs. We also confirmed this trend in MT that translations from MBART contain less hallucinated content than that from TranS2S.

Analysis on Hallucinated Words and their Part-of-Speech Tags In Fig. 5, we present the percentage of hallucinated tokens categorized by their part-of-speech tags predicted by a POS tagger (Toutanova et al., 2003). First, we see that for both MT and summarization datasets, nouns are the most hallucinated words. In abstractive summarization, verbs also account for a certain number of hallucinations. Second, our model predicted hallucinated words match well with gold annotations on the distributions of POS tags. We also compare the percentage of hallucinations within each POS tag in Appendix E.2. In addition, we provide more

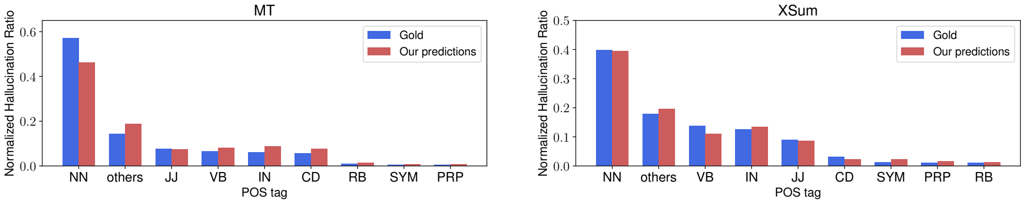

Figure 5: Relationship of POS tags and percentage of hallucinations for MT (left) and summarization (right).

ablation studies in Appendix D.

5.4 Evaluation on Word-level Quality Estimation

As noted in §1, our model is also readily applicable to word-level quality estimation (QE) for MT (Fonseca et al., 2019; Specia et al., 2020), which aims to detect word-level errors in MT output. In the WMT shared task of word-level QE, each token of the target sentence is labeled as OK/BAD based on the post-edited target sentences. We evaluate our model on the WMT18 en-de word-level QE shared task (Specia et al., 2018) in both the unsupervised and supervised setting. There are 13,442 labeled parallel sentences where the tagged target sentences are from an NMT model. In our supervised setting, we finetune the XLM-R model on these parallel sentences with the objective: $\mathcal{L}_{pred} + 0.5 * \mathcal{L}_{mlm}$. In the unsupervised setting, we first create the synthetic data (§3.1) using the post-edited target sentences from the labeled parallel set (13,442) and an additional 50K target sentences from the provided unlabeled parallel set. Then we finetune XLM-R on the created synthetic labeled data. For both settings, we set the weights of the cross-entropy loss for the bad-token labels to be 2.0 because the labels are imbalanced with fewer bad-token labels.

Models	BAD-F1	OK-F1	F1-MULT
OpenKiwi	-	-	44.77
1^{st} place in WMT18	48.00	91.00	44.00
3^{rd} place in WMT18	36.00	85.00	30.00
Ours (unsupervised)	37.09	92.54	34.32
Ours (supervised)	**50.78**	**91.91**	**46.68**

Table 3: F1 scores (x100) on the test set of WMT18 word-level QE. OpenKiwi (Kepler et al., 2019) is the state-of-the-art result on this task. 1^{st} and 3^{rd} place are results from the shared task (Specia et al., 2018).

Results We present results in Tab. 3, where F1-Mult is the multiplication of F1-scores for the OK and BAD labels. Note that all the baseline models are in the supervised setup and the best baseline OpenKiwi (Kepler et al., 2019) is a strong ensembled system using predictions from multiple models. In contrast, our supervised model only leverages the parallel labeled data without using other resources. Among all the supervised settings, our model outperforms the best system by 2 points in F1-Mult. To make it clear how our unsupervised model performs, we also show the best performed systems in the shared task of WMT18. We observe that our unsupervised setting achieves descent performance and even outperforms the 3^{rd}-ranked system. These results demonstrate that both the full version and the finetuning part of our method provide strong results for word-level QE.

6 Case Study I: Improving Self Training in Machine Translation

Predicting hallucination labels at token-level not only allows us to flag potential risks in generation models, but also opens up the possibility of providing fine-grained signals which can be used to define new learning objectives. In this section and the following one, we demonstrate how to leverage the hallucination labels to reduce adverse effects of noisy training instances. Specifically, we show that the fine-grained hallucination signals allow for improved semi-supervised learning (§6) and training with noisy parallel data (§7).

6.1 Rectified Self-Training for Neural MT

Self training (Scudder, 1965) is an important semi-supervised approach that utilizes unlabeled source data to improve system performance. In a conditional sequence generation task, a teacher model is first trained with bitext $\mathcal{D}_l = \{\mathbf{s}_i, \mathbf{t}_i\}_{i=1}^{N}$ and used to make predictions on each sequence in a unlabeled dataset $\mathcal{D}_u = \{\mathbf{s}_j\}_{j=N+1}^{N+M}$ to create *pseudo parallel data* $\mathcal{D}_p = \{\mathbf{s}_j, \mathbf{t'}_j\}_{j=N+1}^{N+M}$. The model is then trained on $\mathcal{D}_l \cup \mathcal{D}_p$. He et al. (2020) finds that with self-training the student model can benefit from such pseudo-parallel data. However, such results require a relatively high-quality teacher, and performance suffers in low-resource setting where no such teacher is available.

We propose to use our token-level hallucination predictions to define a fine-grained loss during training in MT, by penalizing errors less on tokens that more likely to be hallucinated. This is in contrast to previous data filtering methods for MT, which remove entire sentence pairs (Junczys-Dowmunt, 2018; Kang and Hashimoto, 2020).

First, we predict the token-level hallucination labels on the target side of the pseudo parallel data $\mathcal{D}_p$. Then we propose two simple methods of using these labels in self-training: (1) We discard the losses of tokens that are predicted as hallucinations and compute the loss on the remaining tokens for each target sequence (**token loss truncation**). (2) Instead of adjusting losses, we mask the decoder hidden states of those hallucinated positions after the target-to-source cross attention in each decoder layer (**decoder HS masking**).[5]

Methods	BLEU	BLERUT	Hal (%)
baseline	16.14	-0.166	13.69
ST	19.31	-0.059	10.00
ST + paraphrase noise (ST-P)	19.05	-0.051	13.48
ST + random noise (ST-R)	19.97	-0.041	12.55
ST + seq loss truncation	19.91	-0.048	8.26
ST-R + seq loss truncation	19.37	-0.057	10.06
ST + token loss truncation	20.32	0.00244	**6.37**
ST + decoder HS masking	20.57	-0.0001	6.38
ST-R + token loss truncation	**21.02**	**0.043**	7.34
ST-R + decoder HS masking	20.64	0.0308	8.70

Table 4: BLEU(↑), BLEURT(↑) and hallucinated tokens (Hal, ↓) on the CWMT2017 test set. We compare with noised self-training and sequence-level loss truncation in the second and third blocks respectively.

6.2 Experimental Setup and Results

Experimental Setup To train a teacher model (baseline in Tab. 4), we use the same training data described in §4.2 using *patent* (870) as the low-resource domain. We evaluate on the full patent test set (1,500) from CWMT2017 (Wang et al., 2020b). For the unlabeled data, we use the withheld Chinese patent training data (2.9M).

Baselines We compare with the state-of-the-art self-training (ST) method of He et al. (2020), which injects two types of noise into the input sentences: (1) paraphrase noise created by round-trip translations, and (2) random noise from dropping, masking and shuffling input tokens. We also compare with the recently proposed loss truncation method (Kang and Hashimoto, 2020) that adaptively removes entire examples with high log loss, which was shown to reduce hallucinations.

Results and Analysis We present the tokenized BLEU score (Papineni et al., 2002), BLEURT score (Sellam et al., 2020) and the percentage of hallucinated tokens predicted by our system in Tab. 4. We can see that ST improves over the baseline by around 3 BLEU and our best result further improves ST by 1.7 BLEU. Compared with strong baseline methods, our method not only achieves the best translation quality measured by BLEU and BLEURT but also the largest hallucination reduction. We also observe that: (1) Our method with ST alone can outperform other baseline methods, when combined with perturbed ST (noise), and using fine-grained control over the target tokens can further improve the results. (2) ST with paraphrase noise (by round-trip translation) does not perform as well as the random noise, which further confirms that the noisy outputs from a teacher model may hurt the student model. (3) The sequence-level loss truncation approach can improve over the vanilla ST and reduce the level of hallucinations as measured by our system. However, the performance drops when combined with the noised ST.

7 Case Study II: Improving Corpus Filtering for Low-Resource MT

High-quality parallel data is critical for training effective neural MT systems, but acquiring it can be expensive and time-consuming. Many systems instead use mined and filtered parallel data to train NMT models (Junczys-Dowmunt, 2018; Zhang et al., 2020; Koehn et al., 2019). Nonetheless, the selected parallel data can still be noisy, containing misaligned segments. In this section, we demonstrate that token-level hallucination labels can allow us to make better use of noisy data to and improve the overall translation quality. We apply the token loss truncation method proposed in §6 to the filtered parallel data and evaluate it on the WMT2019 low-resource parallel corpus filtering shared task.

Experimental Setup The WMT19 shared task focuses on two low-resource languages – Nepali and Sinhala. It released a very noisy 40.6 million-word (English token count) Nepali-English and a 59.6 million-word Sinhala-English corpus crawled

---

[5] We also tried removing hallucinated target words before training. This underperformed, likely because it produces too many ungrammatical target sentences.

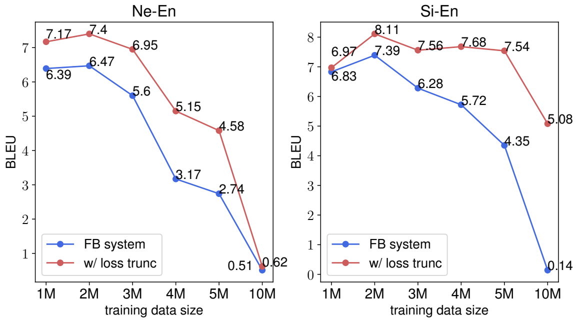

Figure 6: The BLEU scores of the best submission (FB system) in the WMT19 shared task on parallel noisy corpus filtering and our method (w/ loss trunc) on the Ne-En and Si-En flores test sets.

from the web. Participants were asked to score each sentence pair in the noisy parallel set. Scores were used to subsample sentence pairs amounting to 1 million and 5 million English words, which were used to train an MT system that was evaluated on the test set using SacreBLEU (Post, 2018). In addition, the shared task also provides additional clean parallel data for Nepali-English (564K), Sinhala-English (646K) and Hindi-English (1.6M), but they can not be used for training the final NMT system.

First, we train a token-level hallucination prediction system with the combined parallel data from all the three language pairs (as Hindi is related to Nepali). Second, we use the scores (Chaudhary et al., 2019) that achieve the best overall performance for both language pairs among all the submissions to select the top-scored 1M, 2M, 3M, 4M, 5M, and 10M data (in English tokens) and predict the token-level hallucination labels on the target side. We follow the same setup and use the script provided by the shared task to train the NMT model with the selected subsets. During training, we discard losses of tokens that are predicted as hallucinations and only compute the losses for the remaining tokens. We use the validation and test data from the flores dataset (Guzmán et al., 2019) during training and evaluation.

Results and Analysis In Fig. 6, we present the BLEU of the best submission (FB system) and our method on the Ne-En and Si-En test sets of the flores dataset. First, with token-level loss truncation, our model achieves the new best results on the flores test set in this shared task for both Ne-En (7.4) and Si-En (8.11). Second, for both language pairs our method further improves the state-of-the-art system when varying the training data sizes. Notably, in the extreme case of 10M training data, which is very noisy, the baseline can not obtain decent BLEU scores for Si-En while our method still achieves reasonable performance (0.14 vs. 5.18). However, for Ne-En data sizes after 2M causes performance of both the baseline and our method to drop significantly, possibly because the dataset contains many pairs of misaligned sentences (the source is not Nepali and the target is not English).

8 Conclusions

This work proposed a new task of token-level hallucination detection, created human-annotated benchmark datasets, proposed a method for unsupervised learning of hallucination detectors, and showed that the models can be used to define fine grained losses that improve MT training. We demonstrate the remark performance of the proposed hallucination detection method in several downstream tasks, including word-level quality estimation and noisy neural machine translation. In the future, we hope to create a large-scale pretrained hallucination detector for any dataset or model, and also would extend our method to data-to-text generation scenarios. We are also interested in investigating how to leverage our detection methods to mitigate hallucination problems in conditional sequence generation.

Acknowledgements

The work in this paper was supported in part by a Facebook SRA Award. We thank the anonymous reviewers for the insightful feedback that helps us improve the paper.

References

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federman, Dmitriy Genzel, Francisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, et al. 2020. Tico-19: the translation initiative for covid-19. *arXiv preprint arXiv:2007.01788*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Vishrav Chaudhary, Yuqing Tang, Francisco Guzmán, Holger Schwenk, and Philipp Koehn. 2019. Low-resource corpus filtering using multilingual sentence embeddings. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 261–266.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Erick Fonseca, Lisa Yankovskaya, André F. T. Martins, Mark Fishel, and Christian Federmann. 2019. Findings of the WMT 2019 shared tasks on quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 1–10, Florence, Italy. Association for Computational Linguistics.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6100–6113.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. *aInternational Conference on Learning Representations (ICLR)*.

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895.

Daniel Kang and Tatsunori Hashimoto. 2020. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André FT Martins. 2019. Openkiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Philipp Koehn, Francisco Guzmán, Vishrav Chaudhary, and Juan Pino. 2019. Findings of the wmt 2019 shared task on parallel corpus filtering for low-resource conditions. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 54–72.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Chin-Yew Lin and Eduard Hovy. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ruixuan Luo, Jingjing Xu, Yi Zhang, Xuancheng Ren, and Xu Sun. 2019. Pkuseg: A toolkit for multi-domain chinese word segmentation. *CoRR*, abs/1906.11455.

Gary Marcus and Ernest Davis. 2020. Gpt-3, bloviator: Openai's language generator has no idea what it's talking about. *MIT Technology Review*.

Marianna Martindale, Marine Carpuat, Kevin Duh, and Paul McNamee. 2019. Identifying fluently inadequate output in neural and statistical machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 233–243.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan Thomas Mcdonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Mathias Müller, Annette Rios, and Rico Sennrich. 2019. Domain robustness in neural machine translation. *arXiv preprint arXiv:1911.03109*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL 2002*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Masoud Jalili Sabet, Philipp Dufter, and Hinrich Schütze. 2020. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *arXiv preprint arXiv:2004.08728*.

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. Association for Computational Linguistics.

Lucia Specia, FrÃ©dÃ©ric Blain, Marina Fomicheva, Erick Fonseca, Vishrav Chaudhary, Francisco GuzmÃ¡n, and AndrÃ© F. T. Martins. 2020. Findings of the wmt 2020 shared task on quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online. Association for Computational Linguistics.

Lucia Specia, Najeh Hajlaoui, Catalina Hallett, and Wilker Aziz. 2011. Predicting machine translation adequacy. In *Machine Translation Summit*, volume 13, pages 19–23.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, pages 173–180. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020a. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Yong Wang, Longyue Wang, Shuming Shi, Victor OK Li, and Zhaopeng Tu. 2020b. Go from the general to the particular: Multi-domain translation with domain transformation networks. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, New York, USA.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Boliang Zhang, Ajay Nagesh, and Kevin Knight. 2020. Parallel corpus filtering via pre-trained language models. *arXiv preprint arXiv:2005.06166*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. 2019. Density matching for bilingual word embedding. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Minneapolis, USA.

K-ADAPTER: Infusing Knowledge into Pre-Trained Models with Adapters

Ruize Wang[1][*], **Duyu Tang**[2], **Nan Duan**[2], **Zhongyu Wei**[1], **Xuanjing Huang**[1],
Jianshu Ji[3], **Guihong Cao**[3], **Daxin Jiang**[2], **Ming Zhou**[2]
[1]Fudan University, Shanghai, China
[2]Microsoft, Beijing, China
[3]Microsoft, Redmond WA, USA
`{rzwang18,zywei,xjhuang}@fudan.edu.cn`
`{dutang,nanduan,jianshuj,gucao,djiang,mingzhou}@microsoft.com`

Abstract

We study the problem of injecting knowledge into large pre-trained models like BERT and RoBERTa. Existing methods typically update the original parameters of pre-trained models when injecting knowledge. However, when multiple kinds of knowledge are injected, the historically injected knowledge would be flushed away. To address this, we propose K-ADAPTER, a framework that retains the original parameters of the pre-trained model fixed and supports the development of versatile knowledge-infused model. Taking RoBERTa as the backbone model, K-ADAPTER has a neural adapter for each kind of infused knowledge, like a plug-in connected to RoBERTa. There is no information flow between different adapters, thus multiple adapters can be efficiently trained in a distributed way. As a case study, we inject two kinds of knowledge in this work, including (1) factual knowledge obtained from automatically aligned text-triplets on Wikipedia and Wikidata and (2) linguistic knowledge obtained via dependency parsing. Results on three knowledge-driven tasks, including relation classification, entity typing, and question answering, demonstrate that each adapter improves the performance and the combination of both adapters brings further improvements. Further analysis indicates that K-ADAPTER captures versatile knowledge than RoBERTa. [1]

1 Introduction

Language representation models, which are pre-trained on large-scale text corpus through unsupervised objectives like (masked) language modeling, such as BERT (Devlin et al., 2019), GPT (Radford et al., 2018, 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2020), have established state-of-the-art performances on various NLP downstream tasks. Despite the huge success of these pre-trained models in empirical studies, recent studies suggest that models learned in such an unsupervised manner struggle to capture rich knowledge. For example, Poerner et al. (2020) suggest that although language models do well in reasoning about the surface form of entity names, they fail in capturing rich factual knowledge. Kassner and Schütze (2020) observe that BERT mostly did not learn the meaning of negation (e.g. "*not*"). These observations motivate us to study the injection of knowledge into pre-trained models like BERT and RoBERTa.

Recently, some efforts have been made to exploit injecting knowledge into pre-trained language models (Zhang et al., 2019; Lauscher et al., 2019; Levine et al., 2020; Peters et al., 2019; He et al., 2020; Xiong et al., 2020). Most previous works (as shown in Table 1) augment the standard language modeling objective with knowledge-driven objectives and update the entire model parameters. Although these methods obtain better performance on downstream tasks, they struggle at supporting the development of versatile models with multiple kinds of knowledge injected (Kirkpatrick et al., 2017). When new kinds of knowledge are injected, model parameters need to be retrained so that previously injected knowledge would fade away. Meanwhile, the resulting models produce entangled representations, so that it is hard to investigate the effect of each kind of knowledge.

In this paper, we propose K-ADAPTER, a flexible and simple framework that supports the infusion of multiple kinds of knowledge into large pre-trained models. K-ADAPTER leaves the original representation of a pre-trained model unchanged and exports different representations for different types of infused knowledge. This is achieved by

---

[*] Work is done during internship at Microsoft. Zhongyu Wei and Duyu Tang are corresponding authors.

[1] Codes are publicly available at `https://github.com/microsoft/K-Adapter`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1405–1418
August 1–6, 2021.

Model	**Knowledge Source**	**Objective**	**BERT fixed in training?**	**Continual knowledge infusion?**
ERNIE (Zhang et al., 2019)	Wikipedia, WikiData	entity linking	N	N
LIBERT (Lauscher et al., 2019)	WordNet	synonym word prediction, hyponym-hypernym prediction	from scratch	N
SenseBERT (Levine et al., 2020)	WordNet	word-supersense prediction	from scratch	N
KnowBERT (Peters et al., 2019)	Wordnet, Wikipedia, CrossWikis	entity linking , hypernym linking	N	N
WKLM (Xiong et al., 2020)	WikiPedia, WikiData	replaced entity detection	N	N
BERT-MK (He et al., 2020)	Unified Medical Language System	discriminate between real and fake facts	N	N
K-Adapter (this work)	Wikipedia, Wikidata, dependency parser	predication prediction, dependency relation prediction	Y	Y

Table 1: Comparison between our approach (K-ADAPTER) and previous works on injecting knowledge into BERT.

the integration of compact neural models dubbed adapters. Adapters are knowledge-specific models plugged outside of a pre-trained model, whose inputs are the output hidden-states of intermediate layers of the pre-trained model. We take RoBERTa (Liu et al., 2019) as the base pre-trained model and integrate two types of knowledge, including factual knowledge obtained by aligned Wikipedia text to Wikidata triplets and linguistic knowledge obtained by applying off-the-shell dependency parser to web texts. In the pre-training phase, we train two adapters independently. Since adapters have much less trainable parameters compared with RoBERTa, the training process is memory efficient.

We conduct extensive experiments on six benchmark datasets across three knowledge-driven tasks, i.e., relation classification, entity typing, and question answering. Experiments show that K-ADAPTER consistently performs better than RoBERTa, and achieves state-of-the-art performance on five datasets. Case study and probing experiments indicate that K-ADAPTER captures versatile knowledge than RoBERTa.

2 Related Work

Our work relates to the area of injecting knowledge into pre-trained models. As stated in Table 1, previous works mainly differ from the knowledge sources and the objective used for training.

ERNIE (Zhang et al., 2019) injects a knowledge graph into BERT. They align entities from Wikipedia sentences to fact triples in WikiData, and discard sentences with less than three entities. In the training process, the input includes sentences and linked facts, and the knowledge-aware learning objective is to predict the correct token-entity alignment. Entity embeddings are trained on fact triples from WikiData via TransE (Bordes et al., 2013). **LIBERT** (Lauscher et al., 2019) injects pairs of words with synonym and hyponym-hypernym relations in WordNet. The model takes a pair of words separated by a special token as the input, and is optimized by a binary classification problem, which predicts whether the input holds a particular relation or not. **SenseBERT** (Levine et al., 2020) considers word-supersense knowledge. It inject knowledge by predicting the supersense of the masked word in the input, where the candidates are nouns and verbs and the ground truth comes from WordNet. **KnowBERT** (Peters et al., 2019) incorporates knowledge bases into BERT using Knowledge attention and recontextualization, where the knowledge comes from synset-synset and lemma-lemma relationships in WordNet, and entity linking information in Wikipedia. If entity linking supervision is available, the model is learned with an additional knowledge-aware log-likelihood or max-margin objective. **WKLM** (Xiong et al., 2020) replaces entity mentions in the original document with names of other entities of the same type. The model is trained to distinguish the correct entity mention from randomly chosen ones. **BERT-MK** (He et al., 2020) integrates fact triples from knowledge graph. For each entity, it sample incoming and outcoming instances from the neighbors on the knowledge graph, and replaces head or tail entity to create negative instances. The model is learned to discriminate between real and fake facts.

As shown in Table 1, our model (K-ADAPTER) differs from previous studies in three aspects. First,

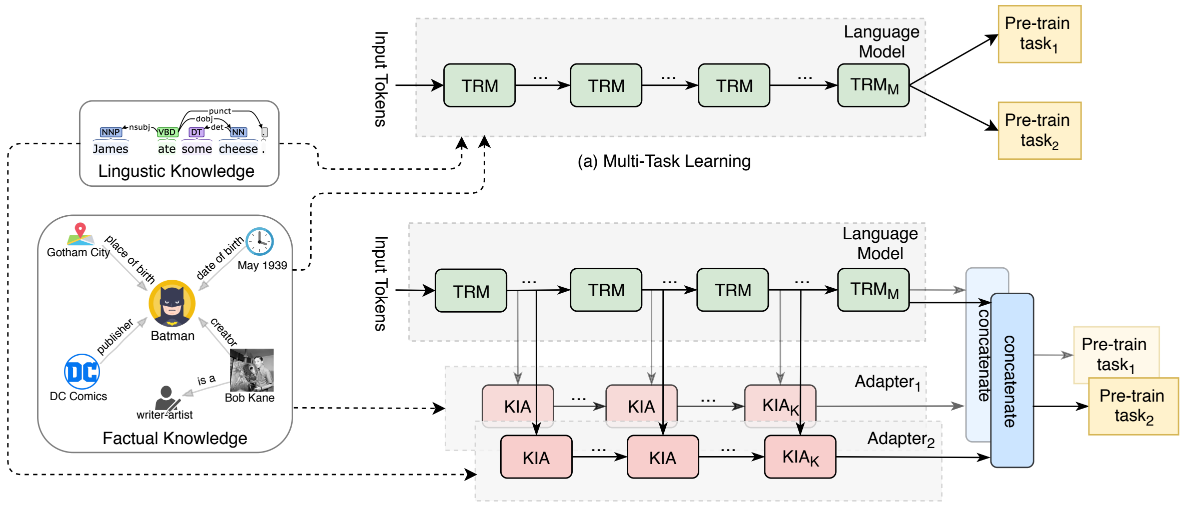

(a) Multi-Task Learning

(b) K-Adapter

Figure 1: (a) Pre-trained language models inject multiple kinds of knowledge with multi-task learning. Model parameters need to be retrained when injecting new kinds of knowledge, which may result in the catastrophic forgetting (b) Our K-ADAPTER injects multiple kinds of knowledge by training adapters independently on different pre-train tasks, which supports continual knowledge infusion. When we inject new kinds of knowledge, the existing knowledge-specific adapters will not be affected. KIA represents the adapter layer and TRM represents the transformer layer, both of which are shown in Figure 2.

we consider both fact-related objective (i.e. predicate/relation prediction) and linguistic-related objective (i.e. dependency relation prediction). Second, the original parameter of BERT is clamped in the knowledge infusion process. Third, our approach supports continual learning, which means that the learning of different adapters are not entangled. This flexibility enables us to efficiently inject different types of knowledge independently, and inject more types of knowledge without any loss on the previously injected knowledge.

3 K-ADAPTER

As illustrated in Figure 1 (a), most of the previous works enhance pre-trained language models by injecting knowledge and update model parameters through multi-task learning. Regardless of these different versions of knowledge-injected methods with multi-task learning, common issues not fully studied are catastrophic forgetting of previous knowledge. To address this, we present K-ADAPTER as shown in Figure 1(b), where multiple kinds of knowledge are injected into different compact neural models (i.e., adapters in this paper) individually instead of directly injecting knowledge into pre-trained models. It keeps the original representation of a pre-trained model fixed and supports continual knowledge infusion, i.e., injecting each kind of knowledge into the corresponding knowledge-specific adapter and producing disentangled representation. Specifically, adapters are knowledge-specific models (with few parameters) plugged outside of a pre-trained model. The inputs of adapters are the output hidden-states of intermediate layers of the pre-trained model. Each adapter is pre-trained independently on different tasks for injecting discriminative knowledge while the original parameters of the pre-trained model are frozen. In this paper, we exploit RoBERTa (Liu et al., 2019) as the pre-trained model, and mainly infuse factual knowledge and linguistic knowledge with two kinds of adapters, i.e., factual adapter and linguistic adapter which are pre-trained on the relation classification task and dependency relation prediction task respectively. In this section, we first describe the structure of our adapter, and then present the process of pre-training knowledge-specific adapters.

3.1 Adapter Structure

In this work, we present a different adapter structure as shown in Figure 2, which is referred to as the knowledge-specific adapter. In contrast to Houlsby et al. (2019) add adapter layers into each transformer layer, our adapter works as outside plug-ins. Each adapter model consists of K adapter layers that contain N transformer (Vaswani et al., 2017) layers and two projection layers. A skip-connection is applied across two projection layers. Specifically, for each adapter model, we plug

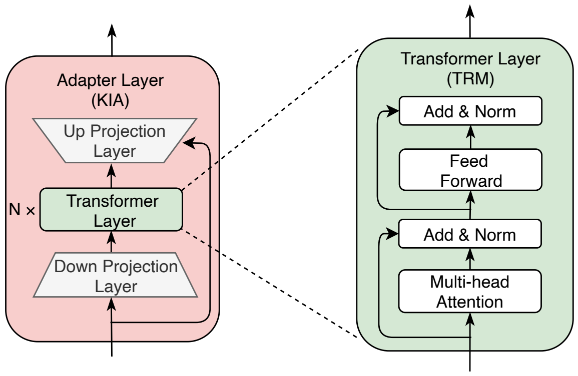

Figure 2: Structure of the adapter layer (left). The adapter layer consists of two projection layers and N=2 transformer layers, and a skip-connection between two projection layers.

adapter layers among different transformer layers of the pre-trained model. We concatenate the output hidden feature of the transformer layer in the pre-trained model and the output feature of the former adapter layer, as the input feature of the current adapter layer. For each knowledge-specific adapter, we concatenate the last hidden features of the pre-trained model and adapter as the final output feature of this adapter model.

In the pre-training procedure, we train each knowledge-specific adapter on different pre-training tasks individually. For various downstream tasks, K-ADAPTER can adopt the fine-tuning procedure similar to RoBERTa and BERT. When only one knowledge-specific adapter is adopted, we can take the final output feature of this adapter model as the input for task-specific layers of the downstream task. When multiple knowledge-specific adapters are adopted, we concatenate the output features of different adapter models as the input for task-specific layers of the downstream task.

3.2 Pre-training settings

We use RoBERTa$_{LARGE}$ (L=24, H=1024, A=16, 355M params) implementation by Huggingface[2] as the pre-trained model in all our experiments. As for each adapter layer, we denote the number of transformer layer as N, the hidden dimension of transformer layer as H_A, the number of self-attention heads as A_A, the hidden dimension of down-projection and up-projection layers as H_d and H_u. In detail, we have the following adapter size: $N = 2$, $H_A = 768$, $A_A = 12$, $H_u = 1024$ and $H_d = 768$. The RoBERTa layers where adapter layers plug in are {0,11,23}, and different adapter layers do not share parameters. Thus the total parameters for each adapter model are about 42M, which are much smaller than RoBERTa$_{LARGE}$ and make the training process memory efficient. It should be noticed that RoBERTa is fixed during training and the parameters of adapters are trainable and initialized randomly. Then we describe how to inject different knowledge into knowledge-specific adapters as below.

3.3 Factual Adapter

Factual knowledge can be described as the basic information that is concerned with facts. In this work, we acquire factual knowledge from the relationships among entities in natural language. We extract a sub-dataset T-REx-rc from T-REx (ElSahar et al., 2018) which is a large scale alignment dataset between Wikipedia abstracts and Wikidata triples. We discard all relations having less than 50 entity pairs, collecting 430 relations and 5.5M sentences. In order to inject factual knowledge, we propose to pre-train a knowledge-specific adapter called facAdapter on the relation classification task. This task requires a model to classify relation labels of given entity pairs based on context. Specifically, the last hidden features of RoBERTa and facAdapter are concatenated as the input representation, and the pooling layer is applied to the input representations of the given entities. Then, we concatenate two entity representations to perform relation classification.

3.4 Linguistic Adapter

Linguistic knowledge is implicitly contained in natural language texts, e.g., syntactic and semantic information. In this work, we acquire linguistic knowledge from dependency relationships among words in natural language text. We build a dataset consisting of 1M examples. In particular, we run the off-the-shell dependency parser from Stanford Parser[3] on a part of Book Corpus (Zhu et al., 2015). To inject linguistic knowledge, we pre-train another knowledge-specific adapter called linAdapter on the task of dependency relation prediction. This task aims to predict the head index of each token in the given sentence. We concatenate the last hidden features of RoBERTa and linAdapter as the input representation, and then apply a linear layer

---

[2] https://github.com/huggingface/transformers

[3] http://nlp.stanford.edu/software/lex-parser.html

Model	OpenEntity			FIGER		
	P	**R**	**Mi-F$_1$**	**Acc**	**Ma-F$_1$**	**Mi-F$_1$**
NFGEC (Shimaoka et al., 2016)	68.80	53.30	60.10	55.60	75.15	71.73
BERT-base (Zhang et al., 2019)	76.37	70.96	73.56	52.04	75.16	71.63
ERNIE (Zhang et al., 2019)	78.42	72.90	75.56	57.19	75.61	73.39
KnowBERT (Peters et al., 2019)	78.60	73.70	76.10	-	-	-
KEPLER (Wang et al., 2021)	77.20	74.20	75.70	-	-	-
WKLM (Xiong et al., 2020)	-	-	-	60.21	81.99	77.00
RoBERTa	77.55	74.95	76.23	56.31	82.43	77.83
RoBERTa + multitask	77.96	76.00	76.97	59.86	84.45	78.84
K-ADAPTER (w/o knowledge)	74.47	74.91	76.17	56.93	82.56	77.90
K-ADAPTER (F)	79.30	75.84	77.53	59.50	84.52	80.42
K-ADAPTER (L)	80.01	74.00	76.89	61.10	83.61	79.18
K-ADAPTER (F+L)	78.99	76.27	**77.61**	61.81	84.87	**80.54**

Table 2: Results on two entity typing datasets OpenEntity and FIGER.

to input representations of each token to perform classification. More training details of facAdapter and linAdapter can be found in the Appendix.

4 Experiments

We evaluate our K-ADAPTER on three knowledge-driven downstream tasks, i.e., entity typing, question answering and relation classification. Furthermore, we conduct detailed analyses with the case study and probing experiments to explore the effectiveness and ability of models for learning factual knowledge. The notations of K-ADAPTER (F+L), K-ADAPTER (F), and K-ADAPTER (L) denote our model which consists of both factual adapter and linguistic adapter, only factual adapter and only linguistic adapter, respectively. Implementation details, and statistics of datasets are in the Appendix.

4.1 Entity Typing

We conduct experiments on fine-grained entity typing which aims to predict the types of a given entity and its context. We evaluate our models on OpenEntity (Choi et al., 2018) and FIGER (Ling et al., 2015) following the same split setting as Zhang et al. (2019). To fine-tune our models for entity typing, we modify the input token sequence by adding the special token "@" before and after a certain entity, then the first "@" special token representation is adopted to perform classification. As for OpenEntity, we adopt micro F_1 score as the final metric to represent the model performance. As for FIGER, we adopt strict accuracy, loose macro, loose micro F_1 scores (Ling and Weld, 2012) for evaluation following the same evaluation criteria used in previous works.

Baselines **NFGEC (Shimaoka et al., 2016)** employs attentive recursive neural networks to compose context representations. **KEPLER (Wang et al., 2021)** integrates factual knowledge with the supervision of the knowledge embedding objective. **RoBERTa+multitask** is our RoBERTa model pre-trained with multi-task learning (as shown in Figure 1(a)) for injecting multiple kinds of knowledge on two pre-training tasks. **K-ADAPTER (w/o knowledge)** consists of a RoBERTa model and an adapter without being injected knowledge. Other baseline models are described in Section 2.

Results and Discussion The results on OpenEntity and FIGER are shown in Table 2. K-ADAPTER (F+L) achieves consistent improvements across these datasets. As for OpenEntity, our RoBERTa achieve better results than other baseline models. K-ADAPTER (F+L) further achieves improvement of 1.38% F_1 over RoBERTa, which means factual knowledge and linguistic knowledge help to predict the types more accurately. As for FIGER, it covers more entity types, and is more fine-grained than OpenEntity. Compared with WKLM, K-ADAPTER (F+L) improves the macro F_1 by 2.88%, micro F_1 by 2.54% and accuracy by 1.60%. This demonstrates that K-ADAPTER (F+L) benefits fine-grained entity typing.

In addition, we further conduct several experiments on our ablated model K-ADAPTER (w/o knowledge), to explore whether the performance gains came from introducing knowledge or additional parameters. Results show that K-ADAPTER (F) significantly outperforms K-ADAPTER (w/o knowledge). Moreover, it is worth noting that on OpenEntity dataset, K-ADAPTER (w/o knowledge) even performs slightly worse than RoBERTa.

Model	SearchQA		Quasar-T		CosmosQA
	EM	F_1	EM	F_1	Accuracy
BiDAF (Seo et al., 2017)	28.60	34.60	25.90	28.50	-
AQA (Buck et al., 2018)	40.50	47.40	-	-	-
Rˆ3 (Wang et al., 2018a)	49.00	55.30	35.30	41.70	-
DSQA (Lin et al., 2018)	49.00	55.30	42.30	49.30	-
Evidence Agg. (Wang et al., 2018b)	57.00	63.20	42.30	49.60	-
BERT (Xiong et al., 2020)	57.10	61.90	40.40	46.10	-
WKLM (Xiong et al., 2020)	58.70	63.30	43.70	49.90	-
WKLM + Ranking (Xiong et al., 2020)	61.70	66.70	45.80	52.20	-
BERT-FT$_{RACE+SWAG}$ (Huang et al., 2019)	-	-	-	-	68.70
RoBERTa	59.01	65.62	40.83	48.84	80.59
RoBERTa + multitask	59.92	66.67	44.62	51.17	81.19
K-ADAPTER (F)	61.85	67.17	46.20	52.86	80.93
K-ADAPTER (L)	61.15	66.82	45.66	52.39	80.76
K-ADAPTER (F+L)	**61.96**	**67.31**	**46.32**	**53.00**	**81.83**

Table 3: Results on question answering datasets including: CosmosQA, SearchQA and Quasar-T.

These results demonstrate that our model gains improvement from knowledge instead of more parameters. Thus, for simplicity, we don't discuss K-ADAPTER (w/o knowledge) in the following experiments.

4.2 Question Answering

We conduct experiments on two question answering (QA) tasks, i.e., commonsense QA and open-domain QA. Commonsense QA aims to answer questions with commonsense. We adopt CosmosQA (Huang et al., 2019) dataset to evaluate our models. CosmosQA requires commonsense-based reading comprehension, formulated as multiple-choice questions. To fine-tune our models for CosmosQA, the input token sequence is modified as *"<SEP>context </SEP>question</SEP>answer</SEP>"*, then the representation of the first token is adopted to perform classification, and will get a score for this answer. After getting four scores, the answer with the highest score will be selected. We report accuracy scores obtained from the leaderboard.

Open-domain QA aims to answer questions using external resources such as collections of documents and webpages. We evaluate our modes on two public datasets, i.e., Quasar-T (Dhingra et al., 2017) and SearchQA (Dunn et al., 2017). Specifically, we first retrieve paragraphs corresponding to the question using the information retrieval system and then extract the answer from these retrieved paragraphs through the reading comprehension technique. Following previous work(Lin et al., 2018), we use the retrieved paragraphs provided by Wang et al. (2017) for these two datasets. To fine-tune our models for this task, the input token sequence is modified as *"<SEP>question </SEP>paragraph</SEP>"*. We apply linear layers over the last hidden features of our model to predict the start and end position of the answer span. We adopt two metrics including ExactMatch (EM) and loose F_1 (Ling and Weld, 2012) scores to evaluate our models.

Baselines BERT-FT$_{RACE+SWAG}$ (Huang et al., 2019) is the BERT model sequentially fine-tuned on both RACE and SWAG datasets. **BiDAF (Seo et al., 2017)** adopts a bi-directional attention network. **AQA (Buck et al., 2018)** proposes to re-write questions and aggregate the answers generated by the re-written questions. **Rˆ3 (Wang et al., 2018a)** is a reinforced model making use of a ranker for selecting most confident paragraph. **Evidence Agg. (Wang et al., 2018b)** proposes making use of the aggregated evidence from across multiple paragraphs. **WKLM (Xiong et al., 2020)** is adopted as the reader model to read multiple paragraphs to predict a single answer. **WKLM + Ranking (Xiong et al., 2020)** is a WKLM paragraph reader plus with a BERT based paragraph ranker to assign each paragraph a relevance score.

Results and Discussion The results on CosmosQA are shown in Table 3. Compared with BERT-FT$_{RACE+SWAG}$, our RoBERTa significantly achieves 11.89% improvement of accuracy. Compared to RoBERTa, K-ADAPTER (F+L) further improves the accuracy by 1.24%, which indicates that K-ADAPTER can obtain better commonsense inference ability. Moreover, the perfor-

Model	**P**	**R**	**F_1**
C-GCN (Zhang et al., 2018)	69.90	63.30	66.40
BERT-base (Zhang et al., 2019)	67.23	64.81	66.00
ERNIE (Zhang et al., 2019)	69.97	66.08	67.97
BERT-large (Baldini Soares et al., 2019)	-	-	70.10
BERT+MTB (Baldini Soares et al., 2019)	-	-	71.50
KnowBERT (Peters et al., 2019)	71.60	71.40	71.50
KEPLER (Wang et al., 2021)	70.43	73.02	71.70
RoBERTa	70.17	72.36	71.25
RoBERTa + multitask	70.18	73.11	71.62
K-ADAPTER (F)	69.39	74.59	71.89
K-ADAPTER (L)	68.85	75.37	71.96
K-ADAPTER (F+L)	70.14	74.04	**72.04**

Table 4: Results on the relation classification dataset TACRED.

mance of ablated K-ADAPTER models, i.e., K-ADAPTER (F) and K-ADAPTER (L) are clearly better than RoBERTa, but slightly lose compared with RoBERTa+multitask. It is notable that K-ADAPTER (F+L) makes obvious improvement comparing with RoBERTa+multitask. This demonstrates that the combination of multiple knowledge-specific adapters could achieve better performance.

The results for open-domain QA are shown in Table 3. K-ADAPTER models achieve better results compared to other baselines. This indicates that K-ADAPTER models can make full use of the infused knowledge and accordingly benefit understanding the retrieved paragraphs to answer the question. Specifically, on SearchQA, K-ADAPTER (F+L) makes significant improvement of 4.01% F_1 scores, comparing with WKLM where the ranking scores are not used, and even has a slight improvement as compared to WKLM+Ranking. It is worth noting that K-ADAPTER models do not consider the confidence of each retrieved paragraph, while WKLM+Ranking utilizes ranking scores from a BERT based ranker. On the Quasar-T dataset, K-ADAPTER (F+L) also outperforms WKLM by 3.1% F_1 score and slightly outperforms WKLM+Ranking.

4.3 Relation Classification

Relation classification aims to determine the correct relation between two entities in a given sentence. We adopt a large-scale relation classification dataset TACRED (Zhang et al., 2017). To fine-tune our models for this task, we modify the input token sequence by adding special token "@" before and after the first entity, adding "#" before and after the second entity. Then the token representations of the first special token "@" and "#" are concatenated to perform relation classification. We adopt micro F_1 score as the metric to represent the model performance as previous works.

Baselines **C-GCN (Zhang et al., 2018)** employs graph convolutional networks to model dependency trees. **BERT-large (Baldini Soares et al., 2019)** is a baseline BERT-large model. **BERT+MTB (Baldini Soares et al., 2019)** is a method of training relation representation without supervision from a knowledge base by matching the blanks. Other baseline models are described in Section 2 and 4.1.

Results and Discussion Table 4 shows the performances of different models on TACRED. The results indicate that K-ADAPTER models significantly outperform all baselines, which directly demonstrate our models can benefit relation classification. In particular, (1) K-ADAPTER models outperform RoBERTa, which proves the effectiveness of infusing knowledge into pre-trained model with adapters. (2) K-ADAPTER models gain more improvement compared with *RoBERTa+multitask*. This directly demonstrates injecting knowledge individually in K-ADAPTER way would help models make full use of knowledge.

4.4 Case Study

Table 5 gives a qualitative comparison example between K-ADAPTER and RoBERTa on relation classification dataset TACRED. The results show that, in most cases, the wrongly predicted logit value of RoBERTa and the logit value of the true label are actually quite close. For example, given "*New Fabris closed down June 16*", RoBERTa predicts "no_relation", but the true label "*city_of_birth*" ranks in second place. If a model could correctly predict the relationship between "*New Fabris*" and "*June 16*", then it needs to know that "*New Fabris*" is a company. Thanks to the factual knowledge in K-ADAPTER, it can help the model from predicting "no_relation" to predicting the correct category label.

In addition, we utilize a LAMA (LAnguage Model Analysis) probe (Petroni et al., 2019) to examine models' ability to memorize factual knowledge. Specifically, the LAMA probing task is under a zero-shot setting, which requires the language model to answer cloze-style questions about relational facts without fine-tuning, For example, given "Simon Bowman was born in [MASK]" as the input, models are asked to predict the correct token

Input	True label	Model	Predicted label	Predicted logits
His former student Mark Devlin of the University of Pennsylvania was co-leader of the other , known as the Microwave Anisotropy Telescope .	schools_attended	K-Adapter	['schools_attended', 'no_relation','founded']	[12.6, 9.5, 5.2]
		RoBERTa	['no_relation', 'founded', ''member_of'']	[9.1, 6.5, 5.0]
Graham had been in custody in Vancouver , British Columbia , since June .	cities_of_residence	K-Adapter	['cities_of_residence', 'countries_of_residence', 'no_relation']	[13.5,6.8,6.6]
		RoBERTa	['countries_of_residence', 'country_of_death', 'alternate_names']	[7.1, 7.0, 6.8]
Vladimir Ladyzhenskiy of Russia died after she suffered a shock in the final of the spa world championship in Heinola , a southern city of Finland , on Saturday .	cause_of_death	K-Adapter	['cause_of_death','origin','no_relation']	[11.0, 7.6, 7.1]
		RoBERTa	['no_relation', 'cause_of_death', 'origin']	[6.3, 5.9, 5.5]
You can't have a good season unless it starts well, '' said Bill Martin, co-founder of ShopperTrak, on Saturday .	founded_by	K-Adapter	['founded_by', 'member_of', 'employee_of']	[10.2, 9.3, 7.3]
		RoBERTa	['no_relation', 'founded_by', 'employee_of']	[10.0, 8.5, 5.4]

Table 5: A case study for K-ADAPTER and RoBERTa on relation classification dataset TACRED. Underlines and wavy lines highlight the subject entities and object entities respectively. We report the top 3 ranked predictions.

Query	Answer	Model	Generation
The native language of Mammootty is [MASK].	Malayalam	RoBERTa	English, Tamil, Hindi, Sanskrit, Arabic, Chinese
		K-ADAPTER	**Malayalam**, Tamil, Hindi, Mandarin, English
Ravens can [MASK].	fly	RoBERTa	win, play, score, lose, run, drink, **fly**, roll, wait
		K-ADAPTER	**fly**, swim, sing, shoot, kill, go, fish, drink, die
Sometimes virus causes [MASK].	infection	RoBERTa	cancer, death, illness, blindness, paralysis
		K-ADAPTER	cancer, illness, death, **infection**, disease
Sunshine Coast, British Columbia is located in [MASK].	Canada	RoBERTa	Florida, California, Texas, Hawaii, Mexico
		K-ADAPTER	**Canada**, Vancouver, Victoria, BC, Australia
iPod Touch is produced by [MASK].	Apple	RoBERTa	**Apple**, Samsung, Qualcomm, LG, Microsoft
		K-ADAPTER	**Apple**, HTC, Samsung, Motorola, Intel

Table 6: Examples of LAMA generation for RoBERTa$_{LARGE}$ and K-ADAPTER. The last column reports the top ranked predicted tokens. Correct predictions are in **bold**.

which is masked. Table 6 shows several examples for the generation of RoBERTa$_{LARGE}$ and K-ADAPTER for LAMA queries. From these examples, we can find that the objects predicted by K-ADAPTER are more accurate, which demonstrate that K-ADAPTER captures richer factual knowledge than RoBERTa. More details about this probing experiments can be found in the Appendix A and E.4.

5 Conclusion

In this paper, we propose a flexible and simple approach, called K-ADAPTER, to infuse knowledge into large pre-trained models. K-ADAPTER remains the original parameters of pre-trained models unchanged and supports continual knowledge infusion, i.e., new kinds of injected-knowledge will not affect the parameters learned for old knowledge. Specifically, factual knowledge and linguistic knowledge are infused into RoBERTa with two kinds of adapters, which are pre-trained on the relation classification task and dependency relation prediction task, respectively. Extensive experiments on three knowledge-driven downstream tasks demonstrate that the performance of each adapter achieves a significant improvement individually, and even more together. Detailed analyses further suggest that K-ADAPTER captures richer factual and commonsense knowledge than RoBERTa, and provide insights on the effectiveness of knowledge infusion. In future work, we will infuse more types of knowledge, and apply our framework to more pre-trained models.

Acknowledgments

This work is partically supported by National Natural Science Foundation of China (No. 71991471), Science and Technology Commission of Shanghai Municipality Grant (No.20dz1200600, 21QA1400600).

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *ACL*, pages 2895–2905.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the Right Questions: Active Question Reformulation with Reinforcement Learning. In *ICLR*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *ACL*, pages 87–96.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. In *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. In *ArXiv preprint arXiv:1704.05179*.

Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Simperl. 2018. T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In *LREC*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. 2020. BERT-MK: Integrating graph contextualized knowledge into pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2281–2290, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *ICML*, pages 2790–2799.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *EMNLP*, pages 2391–2401.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Informing unsupervised pretraining with external linguistic knowledge. *arXiv preprint arXiv:1909.02339*.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *ACL*, pages 1736–1745.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *TACL*, 3:315–328.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237.

Matthew E Peters, Mark Neumann, IV Logan, L Robert, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP*, pages 43–54.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases? In *EMNLP*, pages 2463–2473.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 803–818, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction(AKBC)*, pages 69–74.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*, pages 189–198.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9(0):176–194.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. In *ICLR*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *EMNLP*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware Attention and Supervised Data Improve Slot Filling. In *EMNLP*, pages 35–45.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *ACL*, pages 1441–1451.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *ICCV*, pages 19–27.

Appendix

A Probing Experiments

Although K-ADAPTER models have shown superior performance on knowledge-driven downstream tasks, it does not directly provide insights into whether our models infuse richer factual knowledge. Thus we utilize a LAMA (LAnguage Model Analysis) probe (Petroni et al., 2019) to examine the ability to memorize factual knowledge. Specifically, the LAMA probing task is under a zero-shot setting, which requires the language model to answer cloze-style questions about relational facts without fine-tuning, e.g., "Simon Bowman was born in [MASK]". The model needs to predict a distribution over a limited vocabulary to replace [MASK]. We report mean precision at one (P@1) macro-averaged over relations.

Settings We consider several language models including: ELMo (Peters et al., 2018), ELMo5.5B (Peters et al., 2018), Transformer-XL (Dai et al., 2019), BERT$_{LARGE}$ and RoBERTa$_{LARGE}$. We focus on LAMA-GoogleRE and LAMA-T-REx, which are aimed at factual knowledge. We also conduct probe experiments on LAMA-UHN (Poerner et al., 2020), a more "factual" subset of LAMA, by filtering out queries that are easy to answer from entity names alone. Different models have different vocabulary sizes. To conduct a more fair comparison experiment, we adopt the intersection of vocabularies and let every language model rank only tokens in this vocabulary following Petroni et al. (2019). For simplicity, we only compare K-APDATER (F) which is infused with factual knowledge, with other baseline models.

Results and Discussion Results are shown in Table 7. It is surprising that BERT$_{LARGE}$ performs better than RoBERTa$_{LARGE}$. There is one possible reason: BERT uses a character-level BPE (Gage, 1994) vocabulary, while RoBERTa considers byte-level BPE vocabulary. This finding indicates that, although using bytes makes it possible to learn a subword vocabulary that can encode any text without introducing "unknown" tokens, it might indirectly harm the model's ability to learn factual knowledge, e.g., some proper nouns may be divided into bytes. Thus in the following experiments, we do not take BERT into account.

K-ADAPTER outperforms other models (except for BERT) by a huge margin. As for LAMA, compared to RoBERTa$_{LARGE}$, K-ADAPTER obtains 2.2% and 1.2% P@1 improvement across Google-RE and T-REx, respectively. Moreover, compared to RoBERTa$_{LARGE}$, K-ADAPTER still achieves better results on LAMA-UHN. The results demonstrate that K-ADAPTER captures richer factual and commonsense knowledge than RoBERTa.

B Pre-Training Details

B.1 Factual Adapter

The pre-trained model is fixed during training and the parameters of the factual adapter are trainable and initialized randomly. The model is trained with cross-entropy loss. To accelerate the training process, we set the max sequence length as 64 as the average sequence length of T-REx-rc is only 22.8. We train the model for 5 epochs using a batch size of 128. We use AdamW to optimize our models with the initial learning rate of 2e-5. We train the model with 4 16G NVIDIA V100 GPUs.

B.2 Linguistic Adapter

Same as the training process of the factual adapter, the pre-trained model is fixed during training and the parameters of the linguistic adapter are trainable and initialized randomly. The model is trained with BCEWithLogits loss. We set the max sequence length as 128. We train the model for 10 epochs using a batch size of 256. We use AdamW with the initial learning rate of 1e-5. We train the model with 4 16G NVIDIA V100 GPUs.

C Applying K-adapter on Downstream Tasks

For the downstream tasks, the key point here is the combination of the pre-trained model's representations and adapter's representations, that is to say: leveraging the general information of the pre-trained model on one hand, and the specific knowledge in the adapter on the other. To use K-ADAPTER for downstream tasks is very simple as shown in Figure 5. Usually, when we use pre-trained language models such as BERT and RoBERTa for downstream tasks, we feed the output features from the pre-trained model into the task-specific layer, and then do the corresponding downstream task. As for the K-ADAPTER, we finetune it just like what the orginal BERT or RoBERTa does. We concatenate the output features of the pre-trained model with the features of the adapter, and then feed them to the task-specific layer.

Corpus	Models					
	ELMo	ELMo5.5B	TransformerXL	BERT-large	RoBERTa$_{LARGE}$	K-APDATER
LAMA-Google-RE	2.2	3.1	1.8	12.1	4.8	7.0
LAMA-UHN-Google-RE	2.3	2.7	1.3	6.5	2.5	3.7
LAMA-T-REx	0.2	0.3	19.5	33.9	27.1	29.1
LAMA-UHN-T-REx	0.2	0.2	12.6	26.2	20.1	23.0

Table 7: P@1 on LAMA and LAMA-UHN across Google-RE and T-REx corpora.

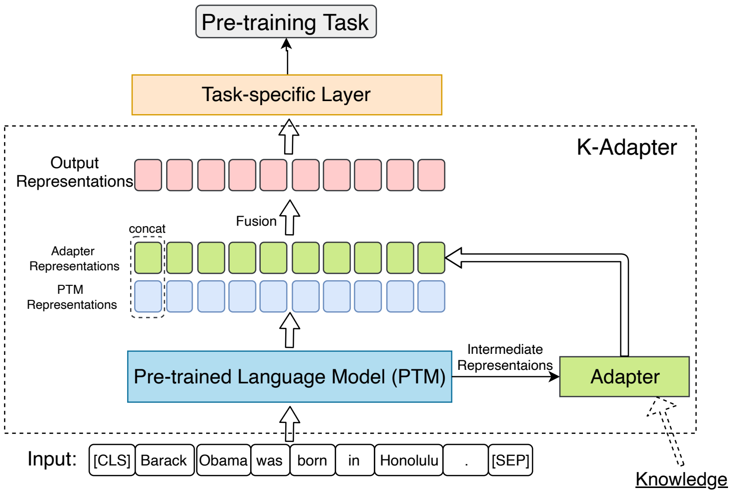

Figure 3: An overview of our K-ADAPTER to inject specific knowledge by training a knowledge-specific adapter on the pre-training task.

D Dataset statistics

In Table 8, we present the statistics of one relation classification dataset TACRED, and two entity typing datasets OpenEntity and FIGER. In Table 9, we present the statistics of one commonsense QA dataset CosmosQA and two open-domain QA datasets SearchQA and Quasar-T.

Dataset	Train	Dev	Test	Relation/Type
TACRED	68,124	22,631	15,509	42
Open Entity	2,000	2,000	2,000	6
FIGER	2,000,000	10,000	563	113

Table 8: The statistics of the relation classification dataset TACRED and entity typing datasets, i.e., Open Entity and FIGER.

Dataset	Train	Dev	Test
CosmosQA	25,588	3,000	7,000
SearchQA	99,811	13,893	27,247
Quasar-T	28,496	3,000	3,000

Table 9: The statistics of the question answering datasets, i.e., CosmosQA, SearchQA and Quasar-T.

E Fine-tuning Details

We implement our experiments using Huggingface[4]. For all fine-tuning experiments, we use AdamW as the optimizer. The parameters of adapters are fixed during the fine-tuning process and the parameters of RoBERTa are trainable and initialized from Huggingface checkpoint. We select the best hyperparameters on the validation set. For all experiments, we set the random seed to be 42 for reproductibility.

E.1 Entity typing

For Open Entity dataset, we set the max sequence length to be 256 and select the hyperparameters from batch size: {4, 8}, learning rate: {2e-5, 1e-5, 5e-6} and warmup step: {0, 200, 500, 1000, 1200}. For K-ADAPTER (F), the best performance is achieved at batch size=4, lr=5e-6, warmup=500 (it takes about 2 hours to get the best result running on singe 16G P100). For K-ADAPTER (L), the best performance is achieved at batch size=4,

---

[4] https://github.com/huggingface/transformers

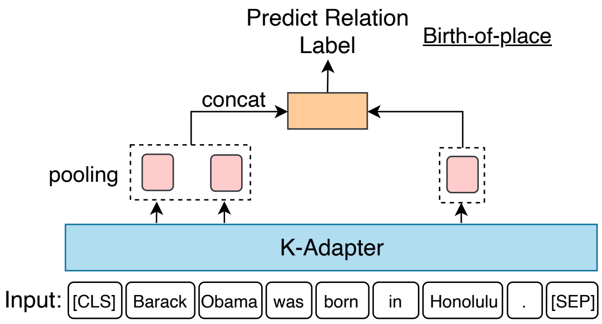

Figure 4: An example of using relation classification as a pre-training task to inject knowledge into K-Adapter: given *"Barack Obama was born in Honolulu"*, and then predicts the relationship between "Barack Obama" and "Honolulu" is "Birth-of-place".

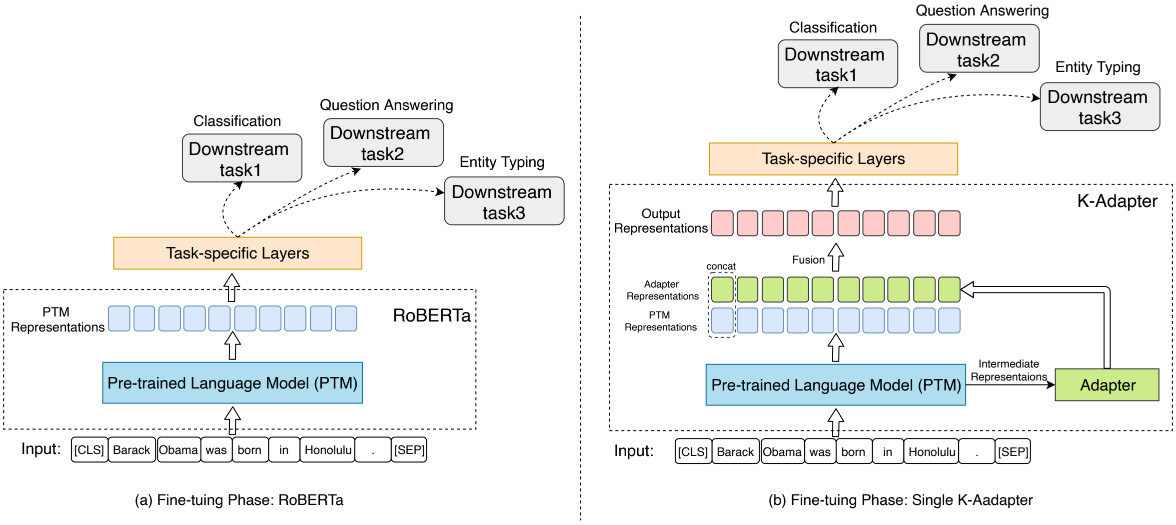

(a) Fine-tuing Phase: RoBERTa

(b) Fine-tuing Phase: Single K-Aadapter

Figure 5: Fine-tuning K-Adapter just like what the original RoBERTa or BERT does.

lr=5e-6, warmup=1000 (it takes about 2 hours to get the best result running on singe 16G P100). For K-Adapter (F+L), the best performance is achieved at batch size=4, lr=5e-6, warmup=1000 (it takes about 3 hours to get the best result running on singe 16G P100). For FIGER dataset, we run experiments on 4 16G P100 for 3 epochs, set the max sequence length to be 256, and select the hyperparameters from batch size: {64, 512, 2048}, learning rate: {2e-5, 1e-5, 5e-6} and warmup step: {0, 200, 500, 1000, 1200}. For K-Adapter (F), the best performance is achieved at batch size=2048, lr=5e-6, warmup=500. For K-Adapter (L), the best performance is achieved at batch size=2048, lr=5e-6, warmup=200. For K-Adapter (F+L), the best performance is achieved at batch size=2048, lr=5e-6, warmup=1000.

E.2 Question Answering

For CosmosQA dataset, we run experiments on one single 16G P100 for 3 epochs, set the max sequence length to be 256, and select the hyperparameters from batch size: {16, 32, 64, 128}, learning rate: {2e-5, 1e-5, 5e-6} and warmup step: {0, 200, 500, 800, 1000}. For K-Adapter (F+L) and its ablated models, the best performance is achieved at batch size=64, lr=1e-5, warmup=0 (it takes about 8 hours to get the best result).

For SearchQA dataset, we run experiments on one single 16G P100 for 2 epochs, set the max sequence length to be 128, and select the hyperparameters from batch size: {2, 4, 8, 16}, learning rate: {5e-5, 2e-5, 1e-5, 5e-6} and warmup step: {0, 500, 1000}. For K-Adapter (F+L) and its ablated models, the best performance is achieved at batch size=8, lr=5e-6, warmup=0. For Quasar-T dataset, we run experiments on one single 16G P100 for 5 epochs, set the max sequence length to be 256, and select the hyperparameters from batch size: {2, 4, 8, 16}, learning rate: {5e-5, 2e-5, 1e-5, 5e-6} and warmup step: {0, 500, 1000}. For K-Adapter (F+L) and its ablated models, the best performance is achieved at batch size=16, lr=1e-5, warmup=0.

E.3 Relation Classification

For TACRED dataset, we run experiments on 4 16G P100 for 5 epochs, set the max sequence length to be 184, and select the hyperparameters from batch size: $\{4, 8, 16, 32\}$, learning rate: $\{2e\text{-}5, 1e\text{-}5, 5e\text{-}6, 1e\text{-}6\}$ and warmup step: $\{0, 200, 500, 800, 1000, 1200\}$. For K-ADAPTER (F), the best performance is achieved at batch size=32, lr=1e-5, warmup=500. For K-ADAPTER (L), the best performance is achieved at batch size=32, lr=1e-5, warmup=200. For K-ADAPTER (F+L), the best performance is achieved at batch size=32, lr=5e-6, warmup=1000.

E.4 Probing Experiments

We implement our probing experiments using LAMA[5]. When we infuse knowledge into knowledge-specific adapters, we do not change the original parameters of the pre-trained model and thus do not adopt the masked language model (MLM) as a pre-training task. Therefore, before we conduct probing experiments, we need to add and train a linear layer as the mlm layer for predicting the [MASK] entities. Specifically, we fix all the parameters of K-ADAPTER and only update the parameters of the mlm layer using a masked language modeling (MLM) loss. We adopt the raw WikiText-2 dataset (181M). We train the mlm layer with one single 16G P100 for 2 epochs. We set the max sequence length to be 512, batch size to be 1024 and warmup step to be 0.

---

[5] https://github.com/facebookresearch/LAMA

Global Attention Decoder for Chinese Spelling Error Correction

Zhao Guo[1], Yuan Ni[1], Keqiang Wang[1], Wei Zhu[1], Guotong Xie[1,2,3]
[1]Ping An Healthcare Technology
[2]Ping An Healthcare and Technology Company Limited
[3]Ping An International Smart City Technology Co.,Ltd
{GUOZHAO385,NIYUAN442,WANGKEQIANG265,ZHUWEI972,
XIEGUOTONG}@pingan.com

Abstract

Recent progress has been made in using BERT framework for Chinese spelling error correction (CSC). However, most existing methods correct words based on local contextual information, without considering the influence of error words in sentences. Imposing attention on error contextual information could mislead and decrease the overall performance of CSC. To address this issue, we propose a **G**lobal **A**ttention **D**ecoder (GAD) approach for CSC. Specifically, the proposed method learns the global relationship of the potential correct input characters and the candidates of potential error characters. Rich global contextual information is obtained to alleviate the impact of the local error contextual information. In addition, a BERT with **C**onfusion set guided **R**eplacement **S**trategy (BERT_CRS) is designed to narrow the gap between BERT and CSC. The candidates generated by BERT_CRS covering the correct character with more than 99.9% probability. To demonstrate the effectiveness of our proposed framework, we test our method on three human-annotated datasets. The experimental results show that our approach outperforms all competitor models by a large margin of up to 6.2%, achieving state-of-the-art methods on all datasets.

1 Introduction

Spelling error correction plays an important role in NLP domain. A good spelling error system is the key to improve the performance of upper-layer applications. Spelling error correction aims to detect and correct erroneous characters/words. These spelling errors are mainly from human writing, speech recognition and optical character recognition (OCR) (Afli et al., 2016) systems. In Chinese, erroneous type is usually from character/word's phonological, visual and semantic similarity. According to (Cheng et al., 2020), about 83% and 48% of errors are related to phonological and visual similarity respectively. Although lots of researches have made great progress, Chinese spelling error correction (CSC) still remains a challenging task. Moreover, because the Chinese is composed of pictographic characters without word delimiters, methods from the languages like English can hardly be used for the Chinese. In addition, the meaning of same character in different contexts may change greatly.

Input	餐厅的换经费产适合约会 The restaurant's swap property is suitable for dates
BERT_CRS	餐厅的环经非常适合约会 The restaurant's ring is perfect for dates
+GAD	餐厅的环境非常适合约会 The restaurant environment is perfect for dates

Table 1: A sample data from SIGHAN 2014 (Yu et al., 2014), the incorrect and correct characters marked in red and green color respectively. Since ”经” is highly related to ”费” in its context, BERT_CRS is difficult to correct. GAD method learns the global relationship between ”环” and ”境” in candidates of input error characters ”换” and ”经” respectively (see Fig.1). Rich global contextual information is learned to alleviate the impact of the local noisy contextual information.

Many methods have been proposed for CSC task, which are mainly divided into two categories: 1) based on language models (Yeh et al., 2013; Yu and Li, 2014; Xie et al., 2015); 2) based on seq2seq model (Wang et al., 2019, 2018). Specially, with the emerge of the pre-trained BERT model, many methods (Hong et al., 2019; Zhang et al., 2020; Cheng et al., 2020) are proposed and made great progress. Almost all methods leveraged a confusion set, which contains a set of similar character group in terms of phonological and visual. Specifically, (Yu and Li, 2014) proposed to generate candidates based on confusion set and find the best

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1419–1428
August 1–6, 2021.

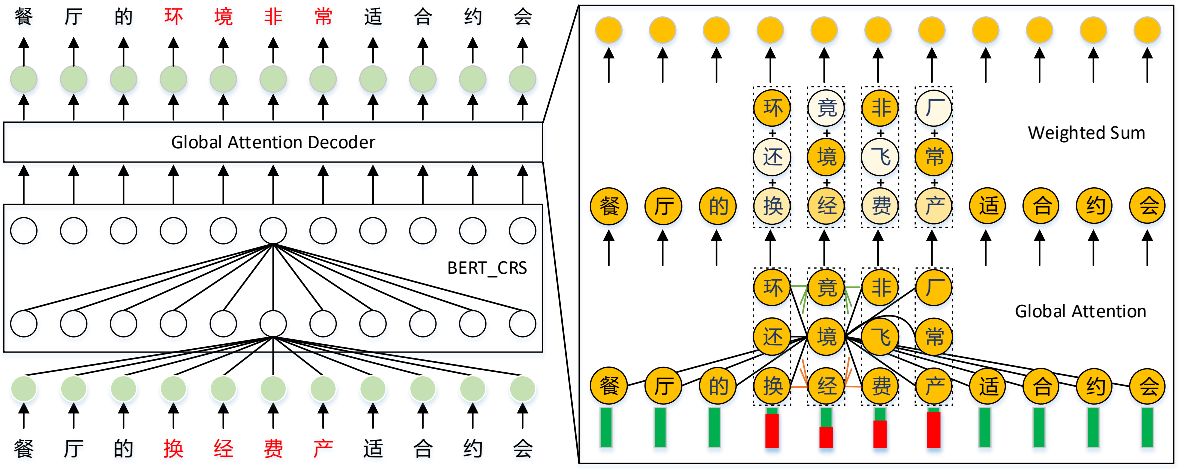

Figure 1: The framework of our proposed global attention decoder method. To illustrate the effectiveness of our model, error words and detection probability are marked with red. For instance ”换经费产” and corresponding error detection probability in bottom right. Attention weights are represented with color shade in right.

candidate with highest language model probability. (Cheng et al., 2020) introduced a convolutional graph network that captures similarity and prior dependencies among characters using confusion set. (Wang et al., 2019) proposed a pointer network to generate a character from the confusion set. Previous methods predict each character or word based on its local context that may has noisy information (other errors). So far, no method has been proposed to alleviate the impact of this noisy information.

In this paper, we firstly introduced a BERT with confusion set guided replacement strategy (BERT_CRS), that narrows the gap between BERT and CSC task. Then, we proposed a novel global attention decoder (GAD) based on our BERT_CRS model (see Fig.1), which learns rich global contextual representations to alleviate the influence of the error contextual information during correction. Specifically, in order to solve the impact of the local error contextual information, we introduce additional candidates of potential error characters and hidden state generated by BERT_CRS. Next, global attention component learns the relationships of candidates to obtain the global hidden state and latent global attention weights of candidates. Then, weighted sum operator is adopted among candidates of each character to generate a rich global contextual hidden state. Finally a fully-connected layer to generate the correct characters. As shown in Table.1, Our proposed method is able to correct all spelling errors correctly. It is worthwhile to highlight the following aspects for the proposed approach:

- To narrows the gap between BERT and CSC, we introduce a BERT with confusion set guided replacement strategy, that contains a decision network and a fully-connected layer to simulate the detection and correction subtasks of CSC respectively.
- We proposed a global attention decoder model, which learns the global relationships of the potential correct input characters and the candidates of potential error characters. Rich global contextual information is learned to effectively alleviate the influence of local error contextual information.
- Experiments on the three benchmark datasets demonstrate that our method outperforms the state-of-the-art methods by a large margin of up to 6.2%.

2 Related Work

There is a vast prior research on Chinese spelling error correction (CSC) task so far. Next, We will discuss the algorithms in different periods.

N-gram period. Early research in CSC follow the pipeline of error detection, candidate generation and candidate selection. Almost all proposed methods (Yeh et al., 2013; Yu and Li, 2014; Xie et al., 2015; Tseng et al., 2015) employed an unsupervised n-gram language model to detect errors. Next, a confusion set which is an external knowledge of the similarity between characters is introduced to confine the candidates. Finally, the best candidate with highest n-gram language model probability is considered as correction character. Specifically, (Yeh et al., 2013) proposed an inverted index based n-gram to map the potential spelling error character to the corresponding characters. (Xie et al., 2015) utilizes the confusion set to replace the characters and then evaluates the modified sentence via a joint bi-gram and tri-gram language model. In (Jia et al., 2013; Xin et al., 2014), a graph model is used to represent the sentence and a single source shortest path (SSSP) algorithm is performed on the graph to correct spell errors. The others viewed it as a sequential labeling problem and employed conditional random fields or hidden Markov models (Tseng et al., 2015; Wang et al., 2018).

Deep learning period. With the development of deep learning methods (Vaswani et al., 2017; Zhang et al., 2020; Hong et al., 2019; Wang et al., 2019; Song et al., 2017; Guo et al., 2016), great progress has been made in all NLP tasks. BERT (Devlin et al., 2018), XLNET (Yang et al., 2019), and Roberta (Liu et al., 2019), and ALBERT (Lan et al., 2019) achieve superior performance in almost all NLP task. Confusion set is still an important part in recent research for CSC task, but more upgrades have been made. Specifically, in (Hong et al., 2019), a pre-trained masked language model is employed as encoder. A confidence-similarity decoder utilizes similarity score to select candidates instead of the confusion set. (Vaswani et al., 2017) proposed a specialized graph convolutional network to incorporate phonological and visual similarity knowledge into BERT model. In (Zhang et al., 2020), a GRU based detection network is introduced and connected with BERT based correction network by a soft-masking technique. The others (Wang et al., 2019) employed a Seq2Seq model with copy mechanism, which generates a new sentence considering the extra candidates from confusion set.

3 The Proposed Approach

In this section, firstly, the problem formulation is elaborated. Then, we briefly describe how to narrow the gap between BERT (Devlin et al., 2018) and Chinese spelling error correction (CSC) using our BERT_CRS model. Finally, we introduce our novel global attention decoder (GAD) framework.

3.1 Problem Formulation

CSC aims to detect and correct errors in Chinese text. Given a sequence $X = \{x_1, x_2, \cdots, x_n\}$, n denotes the number of characters, our BERT_CRS model encodes it into a continuous representation space $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\}$, $\mathbf{v}_i \in \mathbf{R}^d$ is the contextual level feature of the i-th character, and it is d-dimensional. Here a decision network Φ_d models $\mathbf{V}$ to fit a sequence $Z = \{z_1, z_2, \cdots, z_n\}$, where z_i denotes the detection label of the i-th character, and z_i=1 means the character is incorrect and z_i=0 means it is correct. A fully-connected layer on the top of BERT_CRS as correction network Φ_c models $\mathbf{V}$ to fit a sequence $Y = \{y_1, y_2, \cdots, y_n\}$, where y_i is the correction label of the i-th character. Instead of a simple fully-connected layer as a decoder, our GAD models the additional candidates $c = \{c_1, c_2, \cdots, c_n\}$ to alleviate the impact of local error contextual information, where c represents the potential correct input characters and the candidates of potential error characters and:

$$c_i = \begin{cases} c_{i1}, c_{i2}, \cdots, c_{ik}, & if\ P(z_i = 1) \geq t \\ x_i, & if\ P(z_i = 1) < t \end{cases} \quad (1)$$

where k is the number of candidates. t is the threshold of error probability for characters.

3.2 BERT_CRS approach for CSC

In this section, we take advantage of previous models (Devlin et al., 2018; Liu et al., 2019; Cui et al., 2020) and introduce a replacement strategy using confusion set that narrows the gap between BERT and CSC model. There we call this model as BERT_CRS (BERT with **C**onfusion set guided **R**eplacement **S**trategy). Unlike BERT tasks, BERT_CRS has several modifications.

- We drop NSP task and adopt a decision network for detecting error information, that is similar to detection sub-task of CSC.

- As MacBERT (Cui et al., 2020), instead of masking with [MASK] token, we introduce confusion set guided replacement strategy by replacing phonological and visual similar character for masking purpose. Rarely, when there is no confusion character, we will maintain [MASK] token. The strategy similar to correction sub-task of CSC

- We use 23% of input characters for masking. To keep the balance of detection targets (0 for un-replacement, 1 for replacement), we set 35%, 30%, 30%, 5% probability for un-masking, replacing with confusion character, masking with [MASK] token and replacing with random word respectively. Calculated, the replacing and masking probabilities are approximately the same as masking probabilities of BERT.

With model trained by confusion set guided replace strategy, the top-k candidate characters are almost from the confusion set. That prepares for our GAD model.

Learning. Similar to RoBERTa (Liu et al., 2019), confusion set guided replace strategy uses a dynamic approach during training. Error detection and correction is optimized simultaneously in the learning process.

$$L_d = -\sum_{i=1}^{n} logP(z_i|\Phi_d(\mathbf{V})) \quad (2)$$

$$L_c = -\sum_{i=1}^{n} logP(y_i|\Phi_c(\mathbf{V})) \quad (3)$$

$$L = L_c + \lambda * L_d \quad (4)$$

where L_d and L_c is the objective of detection and correction loss respectively, L is the overall objective that linearly combines L_d and L_c, and $\lambda \in [0, 1]$ denotes the coefficient of detection loss L_d. Specially, $\lambda = 0$ represents that detection loss is not considered.

3.3 Global Attention Decoder

In this paper, we propose an global attention decoder (GAD) model to alleviate the impact of the local error contextual information. Our GAD is an extension of transformer layer (Vaswani et al., 2017), shown in Fig.2.

Self Attention. Relatively, the self-attention mechanism is part of the transformer layer, which takes the output of previous transformer layer or input embedding layer as input to obtain the hidden states with higher semantic representation, as shown in left part of Fig.1. Token representation $\mathbf{VA}_i^l$ at i-th position of l-th layer in self-attention method is defined as below:

$$\mathbf{VA}_i^l = \sum_{p=1}^{n} a_i^p \mathbf{V}_p^{l-1}\mathbf{W}^V \quad (5)$$

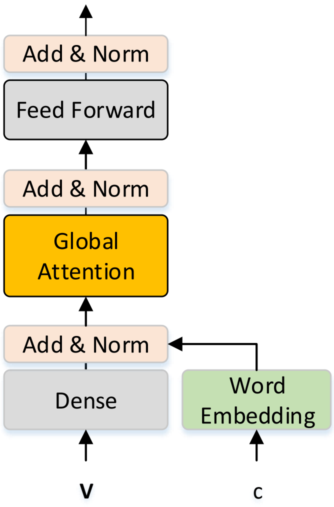

Figure 2: The global attention decoder architecture.

where a_i^p is the attention weight from i-th to p-th token, and $\sum_{p=1}^{n} a_i = 1$, $\mathbf{V}_p^{l-1}$ is the p-th token representation of (l-1)-th layer, $\mathbf{W}^V$ is a learnable projection matrix. This strategy could effectively encode rich token and sentence-level features. However, spelling error information also encoded into hidden states for CSC. Then, Imposing attention on error contextual information could mislead and decrease the overall performance of CSC.

Global Attention. Instead of using only local input information (see Eq.5), we consider potential correct inputs and the candidates of potential error characters to learn their latent relationships, that alleviate the influence caused by local error context. Specifically, as shown in Fig.2, we consider two input sources:

- Contextual representation $\mathbf{V}$, that contains rich semantic information
- Top-k candidates c generate by Φ_c correction network. To reduce the confusion of our GAD during learning, we only generate candidates for the potential error characters (see Eq.1).

To model the two different information, we first embed candidates into continuous representation using the word embedding $\mathbf{E}$ from BERT_CRS. Then, dense and layer-norm layers are introduced to model $\mathbf{V}$ and $\mathbf{E}(c)$ into input state $\mathbf{GI}$:

$$\mathbf{GI} = LayerNorm(\mathbf{Dense}(\mathbf{V}) + \mathbf{E}(c)) \quad (6)$$

Our global attention is introduced to learn the latent relationships between candidates c. Token representation $\mathbf{GA}_{i,j}$ at j-th candidate of i-th token of global attention component is defined as below:

$$\mathbf{GA}_{i,j} = \sum_{p=1}^{n} \sum_{q=1}^{k} a_{i,j}^{p,q} \mathbf{GI}_{p,q} \mathbf{W}_g^V \tag{7}$$

where $\mathbf{W}_g^V$ is a learnable projection matrix and $a_{i,j}^{p,q}$ is the attention weight from j-th candidate of i-th token to q-th candidate of p-th token, $\mathbf{GI}_{p,q}$ denotes the input state of q-th candidate from p-th token. Masking strategy is adopted between candidates from the same token.

$$a_{i,j}^{p,q} = 0,\ if\ i = p\ \&\ j \neq q \tag{8}$$

and $\sum_{p=1}^{n} \sum_{q=1}^{k} a_{i,j}^{p,q} = 1$. Finally, the global attention state $\mathbf{GA}_i$ at i-th position of of global attention component is defined as below:

$$\begin{aligned} \mathbf{GA}_i &= \sum_{j=1}^{k} \beta_{i,j} \mathbf{GA}_{i,j} \\ \epsilon_{i,j} &= \sum_{p=1}^{n} \sum_{q=1}^{k} \epsilon_{i,j}^{p,q} \\ \beta_{i,j} &= \frac{exp(\epsilon_{i,j})}{\sum_{q=1}^{k} exp(\epsilon_{i,q})} \end{aligned} \tag{9}$$

where $\beta_{i,j}$ is the global attention weight at j-th candidate of i-th token which quantifies the global relevance of feature $\mathbf{GA}_{i,j}$, $\epsilon_{i,j}^{p,q}$ and $\epsilon_{i,j}$ denote the unnormalized relevant scores of $a_{i,j}^{p,q}$ and $\beta_{i,j}$ respectively. Similar to standard transformer layer, feed forward and layer normalization to encode $\mathbf{GA}$ into final global continuous representation. Moreover, We adopt the multi-head technique used in the transformer layer in our global attention.

Learning. Given hidden states $\mathbf{V}$ and candidates c generated by our BERT_CRS, our GAD model fit the correct sequence Y in the learning process.

$$L_g = -\sum_{i=1}^{n} logP(y_i|\Phi_g(\mathbf{V})) \tag{10}$$

where Φ_g is our GAD network and L_g denotes our overall objective of GAD

4 Experiments

In this section, we evaluate our algorithm on the task of Chinese spelling error correction (CSC).

Training Data	**# Sent**	**Avg.Len**
UnLabeled corpus	3 million	-
(Wang et al., 2018)	271,329	44.4
SIGHAN 2013	350(350)	49.2
SIGHAN 2014	6,526(3432)	49.7
SIGHAN 2015	3,174(2339)	30.0
Total Labeled	281,379	44.4
Test Data	**# sent**	**Avg.Len**
SIGHAN 2013	1000(996)	74.1
SIGHAN 2014	1062(529)	50.1
SIGHAN 2015	1100(550)	30.5

Table 2: Statistics of datasets. The number in the bracket in #Sent column is the count of erroneous sentences

We first present the training data, test data and the evaluation metrics. Secondly we introduce our main results compared with previous state-of-the-art baselines. Then we conduct ablation studies to analyze the effectiveness of the proposed components. Finally, case study are explored.

4.1 Datasets

We consider three publicly available SIGHAN datasets from the 2013 (Wu et al., 2013), 2014 (Yu et al., 2014) and 2015 (Tseng et al., 2015) Chinese Spell Check Bake-offs. Following (Cheng et al., 2020), we adopted the standard split of training and test data of SIGHAN. We also follow the same data pre-processing, that converted the characters in dataset from traditional Chinese to simple Chinese using OpenCC[1].

For training dataset, we also collect 3 million unlabeled corpus from news, wiki and encyclopedia QA domains to pre-train our BERT_CRS model. Following (Wang et al., 2019), we also include additional 271K samples as the labeled training data, which are generated by an automatic method (Wang et al., 2018). The statistics of the data is showed in Table.2

4.2 Baselines

To evaluate the performance of our proposed algorithm, we compare it with following baseline methods.

- JBT (Xie et al., 2015): This method utilizes the confusion set to replace the characters and then evaluates the modified sentence via a **J**oint **B**i-gram and **T**ri-gram LM.
- Hybird (Wang et al., 2018): This method proposes a pipeline where a bidirectional LSTM

---

[1] https://github.com/BYVoid/OpenCC

Test Set	Model	Detection Level			Correction Level		
		Pre.(%)	**Rec.(%)**	**F1(%)**	**Pre.(%)**	**Rec.(%)**	**F1(%)**
SIGHAN13	JBT (Xie et al., 2015)	79.8	50.0	61.5	77.6	22.7	35.1
	Hybird (Wang et al., 2018)	54.0	69.3	60.7	-	-	52.1
	Seq2Seq (Wang et al., 2019)	56.8	91.4	70.1	79.7	59.4	68.1
	SpellGCN (Cheng et al., 2020)	82.6	88.9	85.7	98.4	88.4	93.1
	BERT (Cheng et al., 2020)	80.6	88.4	84.3	98.1	87.2	92.3
	BERT_CRS	85.5	89.2	87.3	98.9	88.5	93.4
	+GAD	**85.8**	**89.5**	**87.6**	**99.0**	**88.6**	**93.5**
SIGHAN14	JBT (Xie et al., 2015)	56.4	34.8	43.0	71.1	50.2	58.8
	Hybird (Wang et al., 2018)	51.9	66.2	58.2	-	-	56.1
	Seq2Seq (Wang et al., 2019)	63.2	82.5	71.6	79.3	68.9	73.7
	SpellGCN (Cheng et al., 2020)	83.6	78.6	81.0	97.2	76.4	85.5
	BERT (Cheng et al., 2020)	82.9	77.6	80.2	96.8	75.2	84.6
	BERT_CRS	84.6	**81.2**	82.9	97.4	**79.3**	87.4
	+GAD	**85.1**	80.9	**82.9**	**98.0**	79.2	**87.6**
SIGHAN15	JBT (Xie et al., 2015)	83.8	26.2	40.0	71.1	50.2	58.8
	Hybird (Wang et al., 2018)	56.6	69.4	62.3	-	-	57.1
	Seq2Seq (Wang et al., 2019)	66.8	73.1	69.8	71.5	59.5	69.9
	SpellGCN (Cheng et al., 2020)	**88.9**	87.7	**88.3**	95.7	83.9	89.4
	BERT (Cheng et al., 2020)	87.5	85.7	86.6	95.2	81.5	87.8
	BERT_CRS	88.1	**87.9**	88.0	96.1	84.4	89.9
	+GAD	88.6	87.8	88.2	**96.3**	**84.6**	**90.1**

Table 3: The character level performance on both detection and correction level. Our BERT_CRS model achieves similar performance compared with previous state-of-the-art models. Our GAD model achieves better performance.

based sequence labeling model is adopted for detection.

- Seq2Seq (Wang et al., 2019): This method introduces a Seq2Seq model with a copy mechanism to consider the extra candidates from the confusion set.
- FASpell (Hong et al., 2019): This model changes the paradigm by utilizing the similarity metric to select candidate instead of a pre-defined confusion set.
- Soft-Masked BERT (Zhang et al., 2020): This method proposes a detection network, which connected error correction model by a soft-masking technique.
- SpellGCN (Cheng et al., 2020): This model incorporate phonological and visual similarity knowledge into language models for CSC via a specialized graph convolutional network.
- BERT (Devlin et al., 2018): The word embedding on the top of BERT as correction decoder for the CSC task.

4.3 Implementation Details

Training Details. Our code is based on the repository of Transformers[2]. We first fine-tune our BERT_CRS model in 3 million unlabeled corpus based on the pre-trained whole word masking BERT[3]. The procedure runs 5 epochs with a batch size of 1024, learning rate of 5e-5 and max sequence length of 512. Then, we performed the fine-tuning process for our BERT_CRS model in all labeled training data with 6 epochs, a batch size of 32 and a learning rate of 2e-5. Next we fix our BERT_CRS model, and set the number of candidates k and error detection probability t as 4 and 0.25 respectively. Finally we fine-tune our GAD model with 3 epochs, a batch size of 32 and a learning rate of 5e-5. For SIGHAN 13 dataset, we performed additional fine-tune steps for 6 epochs as the data distribution in SIGHAN13 differs from other datasets, e.g. ”的”, ”得” and ”地 are rarely distinguished.

Evaluation Metrics. To evaluate the performance, we employ character and sentence-level accuracy, precision, recall and F1 followed by (Cheng et al., 2020), which are commonly used in the CSC task. In addition, we introduce the official evaluation metrics tool[4], which gives False Positive Rate (FRT), precision, recall, F1 and accuracy in sentence level.

---

[2] https://github.com/huggingface/transformers

[3] https://github.com/ymcui/Chinese-BERT-wwm

[4] http://nlp.ee.ncu.edu.tw/resource/csc.html

Test Set	Model	Detection Level			Correction Level		
		Pre.(%)	**Rec.(%)**	**F1(%)**	**Pre.(%)**	**Rec.(%)**	**F1(%)**
SIGHAN13	FASpell (Hong et al., 2019)	76.2	63.2	69.1	73.1	60.5	66.2
	SpellGCN (Cheng et al., 2020)	80.1	74.4	77.2	78.3	72.7	75.4
	BERT (Cheng et al., 2020)	79.0	72.8	75.8	77.7	71.6	74.6
	BERT_CRS	84.8	79.5	82.1	83.9	78.7	81.2
	+GAD	**85.7**	**79.5**	**82.5**	**84.9**	**78.7**	**81.6**
SIGHAN14	FASpell (Hong et al., 2019)	61.0	53.5	57.0	59.4	52.0	55.4
	SpellGCN (Cheng et al., 2020)	65.1	69.5	67.2	63.1	67.2	65.3
	BERT (Cheng et al., 2020)	65.6	68.1	66.8	63.1	65.5	64.3
	BERT_CRS	65.4	72.7	68.9	63.4	70.4	66.7
	+GAD	**66.6**	**71.8**	**69.1**	**65.0**	**70.1**	**67.5**
SIGHAN15	FASpell (Hong et al., 2019)	67.6	60.0	63.5	66.6	59.1	62.6
	Soft-Masked BERT (Zhang et al., 2020)	73.7	73.2	73.5	66.7	66.2	66.4
	SpellGCN (Cheng et al., 2020)	74.8	80.7	77.7	72.1	77.7	74.8
	BERT (Cheng et al., 2020)	73.7	78.2	75.9	70.9	75.2	73.0
	BERT_CRS	74.0	80.2	77.2	72.2	77.8	74.8
	+GAD	**75.6**	**80.4**	**77.9**	**73.2**	**77.8**	**75.4**

Table 4: The sentence level performance on both detection and correction level. Specially, SpellGCN (Cheng et al., 2020) reports correction level F1 as 75.9 in SIGHAN15. However, 74.8 is calculated by corresponding precision and recall. There the latter value is reported in the table.

4.4 Main Results

We compare our model with the state-of-the-art methods on the three test datasets, and the results are shown in Tab.3 and Tab.4, that compared the results in character-level and sentence level respectively. BERT_CRS outperforms almost all methods in three datasets, and combined with GAD achieving the best performance. Specifically, under the same amount labeled training data, for character level metric, our method gains the improvement against previous best results (SpellGCN) are 0.4%, 2.1%, 0.7% respectively for correction level F1 metric. For sentence level score, our model outperform SpellGCN by a margin of 6.2%, 2.2%, 0.6% respectively for correction level F1 metric. In addition, Soft-Masked BERT uses 5 million examples that generate by replaced strategy for extra training data. our method outperforms it by a large margin in SIGHAN15 test dataset.

We further consider the official evaluation results of BERT_CRS and GAD to compete with BERT and SpellGCN in SIGHAN15, shown in Tab.6. Our proposed BERT_CRS+GAD achieving better performance than SpellGCN by a margin of 0.2% for correction level F1 metric. In addition, the FPR are 13.1% (BERT_CRS+GAD) v.s. 13.2% (SpellGCN).

4.5 Ablation Studies

In this sub-experiment, we explore the impact of several components, including the coefficient λ and learning rate lr in BERT_CRS and the effective parameter k that is the number of candidates in GAD

Model	Parameters	Value	**F1(%)**
BERT_CRS	λ	1	72.0
		0.5	73.4
		0.1	74.8
	lr	2e-5	74.8
		5e-5	74.6
GAD	k	3	75.4
		4	75.1
		5	74.7

Table 5: The effect of parameters in BERT_CRS and GAD for correction level F1 metric on SIGHAN15.

The Effect of BERT_CRS. Our BERT_CRS introduces confusion set guided replacement strategy using BERT model. Compared with BERT model, for character level metric in Table.3, BERT_CRS improves the performance by a margin 6.6%, 2.5%, 1.8% respectively for correction level F1. For sentence level metric in Table.4, we achieving the scores 81.2% (BERT_CRS) v.s. 74.6% (BERT) on SIGHAN 13, 66.7% (BERT_CRS) v.s. 64.3% (BERT) on SIGHAN 14 and 74.8% (BERT_CRS) v.s. 73.0% (BERT) on SIGHAN 15.

We also show the effect of coefficient λ and learning rate during fine-tuning in all labeled datas, shown in Tabel.5. First we fix learning rate as 2e-5 and tune $\lambda \in [0.1, 0.5, 1]$ on SIGHAN15. When λ=0.1, the best performance is achieved. In addition, big variation is shown with different λ, That is to say, if more attention of detection loss, the performance is unsatisfactory. The reason of the situation may be caused by the imbalance of detection label during the fine-tuning process. In the following

Model	FPR	Detection Level				Correction Level			
		Acc.	**Pre.(%)**	**Rec.(%)**	**F1(%)**	**Acc.**	**Pre.(%)**	**Rec.(%)**	**F1(%)**
SpellGCN (Cheng et al., 2020)	13.2	**83.7**	85.9	**80.6**	83.1	82.2	85.4	77.6	81.3
BERT (Cheng et al., 2020)	13.6	83.0	85.9	78.9	82.3	81.5	85.5	75.8	80.5
BERT_CRS	14.0	83.1	85.1	80.2	82.6	81.9	84.8	77.8	81.1
+GAD	**13.1**	83.6	**86.0**	80.4	**83.1**	**82.4**	**85.6**	**77.8**	**81.5**

Table 6: The sentence level performance evaluated by official tools on SIGHAN 2015. The smaller FPR score indicates the better performance.

experiments, we set λ=0.1. We tune learning rate from [2e-5, 5e-5]. When 2e-5 is adopted, the better performance is achieved. We set learning rate equal to 2e-5 during experiments.

The Effect of GAD When combined with GAD in BERT_CRS model, the performance is improved under character and sentence level metric, shown in Tabel.3 and Tabel.4. Specifically, for sentence level metric, BERT_CRS+GAD outperform BERT_CRS by a margin of 0.4%, 0.8% and 0.6% respectively for correction level F1 metric.

We also study the impact of candidate number k. Since k is the key parameter which determines the coverage of correct character in candidates, it affects the performance of our algorithm. We study the performance variance with different $k \in [3, 4, 5]$ on SIGHAN15. Shown in Tabel.5, more candidates may degrade the performance. According to statistics, there are 161,365 error characters in all test data and 106, 75, 64 not in candidates for k equal to 3, 4, 5 respectively. The candidates generated by BERT_CRS model have 99.9% probability covering the correct character. Consider the trade-off between cover rate of candidates and performance, We set $k = 4$ in our experiments.

4.6 Case Study

To further analyze our approach, we show some correction results on test data (see Table.7). In Table.7, three categories of spelling error are selected: 1) Continuous characters error; 2) Single character error; 3) No character error. For Continuous characters error instance, ”介绍” (introduce) was misspelled as ”借少” (borrow less). Due to the influence of error characters, BERT_CRS is difficult to correct them all correctly. However, BERT_CRS+GAD alleviates the impact of the local error contextual to correct them all correctly. For single character error instance, ”抱” (pick up) was misspelled as ”包” (pack). Our BERT_CRS+GAD can also learns richer global contextual information to correct it than BERT_CRS. Here it has the same meaning of ”提议” (suggestion) and ”建议” (suggestion) in no character instance, BERT_CRS miscorrects it. These cases prove that our GAD can learn rich global contextual information to alleviate the impact of the local error contextual for CSC.

Continuous characters error
...语言。去外国可以认识很多的人，就可以借少
...语言。去外国可以认识很多的人，就可以借绍
...语言。去外国可以认识很多的人，就可以介绍
... you can meet a lot of people abroad, and introduce these languages.
Single character error
我把小猫抱起来，赶紧包出去到马路边求救...
我把小猫抱起来，赶紧跑出去到马路边求救...
我把小猫抱起来，赶紧抱出去到马路边求救...
I picked up the kitten and hurried out to the side of the road for help.
No character error
...课堂之前可以先有一些提议或许参考的资料...
...课堂之前可以先有一些建议或许参考的资料...
...课堂之前可以先有一些提议或许参考的资料...
Some suggestions or reference materials can be available before the class.

Table 7: Examples of CSC results, the incorrect and correct characters marked in red and green respectively. The first line in the block is input sentence. The second and third line is corrected by BERT_CRS and BERT_CRS+GAD respectively. And the rest is the English translation of the correct sentence.

We also showed some incorrect case to further analyze our model. For example, for the sentence ”希望您帮我素取公平，得到他们适当的赔偿”(I hope you can help me x for justice and get proper compensation from them) where the incorrect word ’x’ is not comprehensible, our GAD changes ”素取”(x) to ”争取”(strive for) that is appropriate in the context, but ground-truth ”诉取”(sue for) is more suitable because the context contains the meaning of litigation. Our GAD model also lacks the inference ability of context strong correlation as described in (Zhang et al., 2020).

5 Conclusions

In this paper, we propose a novel global attention decoder (GAD). Condition on the potential correct input characters and the candidates of potential error characters, GAD reforms the self attention mechanism to learn their global relationships and obtains the rich global contextual information to alleviate the influence caused by error context. In addition, a BERT with **C**onfusion set guided **R**eplacement **S**trategy (BERT_CRS) is designed to narrow the gap between BERT and spelling error correction. Experimental results on three datsets show that our BERT_CRS outperform almost all previous state-of-the art methods, and higher performance is obtained by combining with our GAD.

References

Haithem Afli, Zhengwei Qiu, Andy Way, and Páraic Sheridan. 2016. Using SMT for OCR error correction of historical texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 962–966, Portorož, Slovenia. European Language Resources Association (ELRA).

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. SpellGCN: Incorporating phonological and visual similarities into language models for Chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 871–881, Online. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhao Guo, Lianli Gao, Jingkuan Song, Xing Xu, Jie Shao, and Heng Tao Shen. 2016. Attention-based lstm with semantic consistency for videos captioning. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, page 357–361, New York, NY, USA. Association for Computing Machinery.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. FASPell: A fast, adaptable, simple, powerful Chinese spell checker based on DAE-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169, Hong Kong, China. Association for Computational Linguistics.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph model for Chinese spell checking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 88–92, Nagoya, Japan. Asian Federation of Natural Language Processing.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jingkuan Song, Lianli Gao, Zhao Guo, Wu Liu, Dongxiang Zhang, and Heng Tao Shen. 2017. Hierarchical lstm with adjusted temporal attention for video captioning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2737–2743.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to SIGHAN 2015 bake-off for Chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for Chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527, Brussels, Belgium. Association for Computational Linguistics.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for Chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785, Florence, Italy. Association for Computational Linguistics.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42, Nagoya, Japan. Asian Federation of Natural Language Processing.

Weijian Xie, Peijie Huang, Xinrui Zhang, Kaiduo Hong, Qiang Huang, Bingzhou Chen, and Lei Huang. 2015. Chinese spelling check system based on n-gram model. In *Proceedings of the*

Eighth SIGHAN Workshop on Chinese Language Processing, pages 128–136, Beijing, China. Association for Computational Linguistics.

Yang Xin, Hai Zhao, Yuzhu Wang, and Zhongye Jia. 2014. An improved graph model for Chinese spell checking. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 157–166, Wuhan, China. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding.

Jui-Feng Yeh, Sheng-Feng Li, Mei-Rong Wu, Wen-Yi Chen, and Mao-Chuan Su. 2013. Chinese word spelling correction based on n-gram ranked inverted index list. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 43–48, Nagoya, Japan. Asian Federation of Natural Language Processing.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223, Wuhan, China. Association for Computational Linguistics.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of SIGHAN 2014 bake-off for Chinese spelling check. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132, Wuhan, China. Association for Computational Linguistics.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 882–890, Online. Association for Computational Linguistics.

Jointly Identifying Rhetoric and Implicit Emotions via Multi-Task Learning

Xin Chen[1], Zhen Hai[2], Deyu Li[1,3], Suge Wang[1,3]* and Dian Wang[1]
1. School of Computer and Information Technology, Shanxi University, Taiyuan, China
2. R&D Center Machine Intelligence Technology Alibaba DAMO Academy, Singapore
3. Key Laboratory of Computational Intelligence and Chinese Information Processing
of Ministry of Education, Shanxi University, Taiyuan, China
`cx_simple@163.com`, `haiz0001@e.ntu.edu.sg`
`{lidy,wsg}@sxu.edu.cn`, `sxu_wdd@163.com`

Abstract

Rhetorical implicit emotion identification is one of important and challenging tasks in natural language processing. We observe that each rhetoric may express certain evidence of semantic and syntactic patterns. Then, we design a gate mechanism based classification module to capture respective rhetorical representation and identify each rhetoric. Moreover, sentences carved with rhetoric tends to express emotions in subtle ways. We thus propose a new multi-task learning framework that can encode the categorical correlation between tasks to improve the performance of rhetoric and emotion identification problem. Experimental results validate the benefit of the proposed model over state-of-the-art baselines for rhetoric and emotion identification tasks. In addition, a new Chinese rhetorical implicit emotion dataset was constructed and will be released in this work.

1 Introduction

Rhetoric is formed by decorating the semantics and adjusting the structure of sentences in text (Kennedy, 2009; Wang, 2013). The sentences that express implicit emotions via rhetorics are commonly used in literary works and spiritual user reviews. They tend to express semantics at a high cognitive level and convey emotions in an obscure way. The example sentence below actually expresses the emotion of "love" and meanwhile consists of two rhetoric forms of parallelism and simile.

> 宽容是一块块木板，可以架起人与人之间的桥梁；宽容是一根根丝带，可以系住人与人之前的友谊；宽容是一团团火焰，可以融化人与人之间的隔阂。(*Tolerance is a piece of board, which can build a bridge between people; Tolerance is a ribbon, which can tie the friendship between people; Tolerance is a flame, which can melt the barriers between people.*)

Clearly, one can understand and predict the emotion ("love") and rhetoric ("simile" and "parallelism") of the sentence by capturing the semantic meanings of content words ("tolerance", "board", "bridge", "ribbon", "friendship") as well as the syntactic structure and pattern of parallelism clauses in the sentence ("Tolerance is ⋯, which can⋯"). Semantics and syntax are pivotal cornerstones for rhetoric, and are attached to different rhetoric. Moreover, positive emotions are more likely to be expressed by rhetorical categories such as simile, parallelism and so on, while negative emotions are more commonly conveyed by the rhetorical categories such as rhetorical questions, irony and so on.

Sentences carved with rhetoric can convey emotions in an implicit way, without explicitly using emotional words, which pose new challenges for textual understanding and rhetorical emotion analysis in natural language processing.

Various efforts have been made to cope with emotion identification (Delbrouck et al., 2020; Chen et al., 2018; Klinger et al., 2018) and rhetoric detection (Liu et al., 2018; Wen et al., 2019; Mao et al., 2019). However, constructing heuristic rules and syntactical patterns for rhetoric identification is time-consuming and labor-intensive, and it may perhaps result in poor generalization to develop a specific detection model for each rhetoric category (Liu et al., 2018; Mao et al., 2019). To our knowledge, the correlation between rhetoric and emotion identification tasks has not been exploited in previous work.

In this work, we aim to cope with rhetoric and emotion analysis. Following previous studies, we formulate each of the rhetoric and emotion identification as a multi-label classification problem.

---

*Corresponding author: Suge Wang.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1429–1434
August 1–6, 2021. ©2021 Association for Computational Linguistics

Then, we propose a new multi-task learning model called REI-MUL, which can jointly leverage the task correlation to address the rhetoric and emotion identification in a unified framework.

In particular, a pre-trained language model (Devlin et al., 2019) and a tree-structured long short-term memory networks (Tree-LSTMs) (Tai et al., 2015) are used to encode the semantic and syntactic representations of sentences, respectively. Taking the representations as input, two correlation modules are introduced to learn the categorical correlations between rhetoric and emotion tasks. One key benefit of REI-MUL is that it exploits the correlation between the two tasks, and may allow better learning of model parameters from real-life natural language data.

2 Methodology

2.1 Overview

Formally, let C be a labeled set of sentences, $C = \{(S_1, Y_1^r, Y_1^e), (S_2, Y_2^r, Y_2^e), \cdots, (S_L, Y_L^r, Y_L^e)\}$, where S_l is an input sentence, while $Y_l^r = \{y_{l_1}^r, y_{l_2}^r, ..., y_{l_R}^r\}$ and $Y_l^e = \{y_{l_1}^e, y_{l_2}^e, ..., y_{l_E}^e\}$ denote the rhetoric label set (with R categories) and the emotion label set (with E categories) of the sentence S_l, respectively. Note that $y_{l_i}^r$, $y_{l_j}^e \in \{0, 1\}$, $y_{l_i}^r = 1$ and $y_{l_j}^e = 1$ mean the S_l is labeled as the i-th rhetoric and j-th emotion categories.

Based on the labeled set C, we develop a new multi-task learning model (REI-MUL), as shown in Figure 1, which can exploit task correlation to deal with rhetoric identification and emotion recognition in a unified framework.

2.2 Semantic Representation

As one of the state-of-the-art language representation models, the BERT model (Devlin et al., 2019) is employed to learn the sematic representations of sentences. Specifically, the contextualized representation of $[CLS]$ token is used as the encoding of whole sequence, which is denoted by $sr^{sem} \in \mathbb{R}^{d_1}$. For simplicity, we omit the subscript of input sentence "l" for subsequent sections.

2.3 Syntactic Representation

A well-known Tree-LSTMs (Tai et al., 2015) is exploited to encode syntactic representations of sentences. Word sense is a critical factor for structural representation. We thus represent the meaning of a word via connecting the *comprehensive* embedding (Song et al., 2018) and the *sememes knowledge* embedding from SE-WRL Model (Niu et al., 2017).

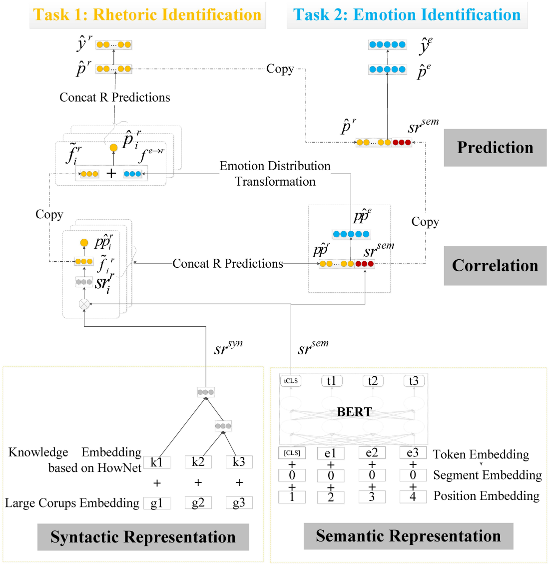

Figure 1: The architecture of the proposed multi-task learning model for rhetoric and emotion identification (REI-MUL). REI-MUL consists of four parts.

Taking the word representations as input, the hidden state of root node in Tree-LSTMs model is taken as syntactic representation of each input sentence, as shown in Eqn (1).

$$sr^{syn} = Tree - LSTMs(X, X^{syn}) \qquad (1)$$

where $X = \{x_1, x_2, ..., x_N\}$, x_i means the input embedding of each word w_i, X^{syn} is the dependency parsing of S by using Stanford Parser, and $sr^{syn} \in \mathbb{R}^{d_2}$.

2.4 Correlation Layer

Both rhetorical and emotional modules of correlation layer are designed to obtain the label distributions of sentences, which can help to explicitly encode the semantic and syntactic correlations between rhetoric and emotion identification tasks.

2.4.1 Rhetoric Distribution

A gate mechanism (Cho et al., 2014) based classification model is proposed for each type of rhetoric, which assigns different weights to semantic and syntactic representations. Taking the i-th rhetoric identification as an example, the classifier can derive three types of representations, i.e., sentence representation, feature representation, and rhetoric distribution via Eqn (2)-(7).

$$r_i = \sigma(W_i^r * [sr^{syn}, sr^{sem}]) \qquad (2)$$

---

https://nlp.stanford.edu/software/lex-parser.html ("version 4.0.0, Chinese")

$$z_i = \sigma(W_i^z * [sr^{syn}, sr^{sem}]) \tag{3}$$

$$\tilde{sr}_i = tanh(W_i^{\tilde{sr}} * [r_i \odot sr^{syn}, sr^{sem}]) \tag{4}$$

$$sr_i^r = (1 - z_i) * sr^{syn} + z_i * \tilde{sr}_i \tag{5}$$

where r_i, z_i and $sr_i^r \in \mathbb{R}^{d_2}$ denote the reset gate, update gate and sentence representation, respectively. σ and $tanh$ denote the sigmoid and hyperbolic tangent activation functions, and $\odot$ is elementwise multiplication.

$$\tilde{f}_i^r = sr_i^r W_i^{fr} \tag{6}$$

where $\tilde{f}_i^r$ is the feature representation for i-th rhetoric identification, and $\tilde{f}_i^r \in \mathbb{R}^{d_3}$.

$$p\hat{p}_i^r = \tilde{f}_i^r W_i^{pr} \tag{7}$$

where $p\hat{p}_i^r$ refers to rhetoric distribution for i-th rhetoric identification, and $p\hat{p}_i^r \in \mathbb{R}^1$.

Next, the respective R rhetoric distributions are concatenated as one comprehensive rhetoric distribution via Eqn (8), which would be used to compute the emotion distribution.

$$p\hat{p}^r = [p\hat{p}_1^r || p\hat{p}_2^r || ... || p\hat{p}_R^r] \tag{8}$$

where $p\hat{p}^r \in \mathbb{R}^R$.

2.4.2 Emotion Distribution

The emotional features are constructed by concatenating the semantic representation and rhetorical distribution of S. Taking the features as input, a classification model can be adopted to compute emotional distribution via Eqn (10), which would be next used to improve the rhetorical prediction task.

$$sr^e = [sr^{sem} || p\hat{p}^r] \tag{9}$$

$$p\hat{p}^e = sr^e W^{pe} \tag{10}$$

where $p\hat{p}^e$ denotes emotional distribution of input sentence S, and $p\hat{p}^e \in \mathbb{R}^E$.

2.5 Prediction Layer

2.5.1 Rhetoric Prediction Layer

An emotional distribution is incorporated into original rhetorical features, which is extracted from sentence representations. Then a sigmoid classifier is designed to take the correlated representation of a sentence as input, and predicts rhetorical probability scores. Formally, Eqn (11)-(13) calculate the transformed features based on emotional distribution ($f^{e \rightarrow r}$), new correlated features (f_i^r) and predicted probability ($\hat{p}_i^r$), respectively, for i-th rhetoric identification task.

$$f^{e \rightarrow r} = p\hat{p}^e W^{e \rightarrow r} \tag{11}$$

$$f_i^r = \tilde{f}_i^r + f^{e \rightarrow r} \tag{12}$$

$$\hat{p}_i^r = \sigma(f_i^r W_i^{pr}) \tag{13}$$

where $W^{e \rightarrow r}$ represents the feature transformation matrix of emotional distribution.

Next, the R predicted scores are concatenated as one rhetorical predicted probability distribution $\hat{p}^r \in \mathbb{R}^R$.

2.5.2 Emotion Prediction Layer

Similarly, a new correlated representation of input sentence is computed based on the predicted rhetorical distribution and semantic representation of the sentence via Eqn (14). And then we predict probability scores of emotion by a sigmoid classifier via Eqn(15).

$$sr^e = [sr^{sem} || \hat{p}^r] \tag{14}$$

$$\hat{p}^e = \sigma(sr^e W^{pe}) \tag{15}$$

where $\hat{p}^e \in \mathbb{R}^E$ denotes the emotional prediction probabilities of input sentence.

3 Experiments

3.1 Data

To our knowledge, there is a labeled dataset for emotion analysis of metaphorical expressions, which was constructed from a cognitive process (Zhang et al., 2018). However, there are no datasets available for joint rhetoric and emotion identification problem from the perspective of linguistics. We constructed the first dataset in Chinese, where each sentence may contain multiple rhetoric/emotion labels. Table 1 shows the detailed statistics of the dataset.

Category Name	Train	Val	Test	All
similes/metaphors	5,121	650	652	6,432
parallelism	1,903	238	234	2,375
personification	1,755	223	226	2,204
rhetorical questions	1,582	180	183	1,945
irony	649	82	82	813
Total	11,010	1,373	1,377	13,760
joy	133	22	22	177
love	6,329	784	808	7,921
anger	109	15	10	134
sadness	724	91	78	893
fear	97	12	13	122
disgust	2,042	256	239	2,537
surprise	58	5	14	77
Total	9,492	1,185	1,184	11,861
Sentence Number	9,456	1,182	1,182	11,820

Table 1: The statistics of the annotated dataset.

Each sentence of the dataset was collected from

	joy	love	anger	sadness	fear	disgust	surprise
similes/metaphors	83	5,446	25	406	88	376	23
parallelism	51	2,034	2	181	8	125	2
personification	29	1,876	17	149	24	71	7
rhetorical questions	21	272	87	232	9	1,282	43
irony	0	4	6	10	0	790	3

Table 2: The correlation between rhetorical and emotional classes on the dataset.

various sources such as literary works, textbooks, microblog, and websites. Three graduate students were hired to annotate the rhetorical and emotional labels of each sentence, where the inter-rater kappa coefficients are 0.848, 0.692, 0.757 for rhetoric annotation task and 0.458, 0.512, and 0.556 for emotion task, respectively. We randomly selected 80% of the annotated dataset as training data, while the rest 20% was equally divided as testing (test) and development (dev) data. All the models were evaluated on the same data split.

Table 2 shows the correlation statistics between rhetorical and emotional categories on the dataset. For example, the emotion "love" of a sentence is more likely to be expressed by rhetorical categories such as simile, parallelism and personification, while the emotion "disgust" is more commonly conveyed by the rhetorical categories such as rhetorical questions and irony.

3.2 Comparison Systems

We compared the proposed models with the following well-established baselines and ablation models.

- **CNN-Adversarial-MUL**: Liu et al. (2017) proposed an adversarial multi-task learning framework for text classification, alleviating the shared and private latent feature spaces from interfering with each other.

- **BERT-MUL**: Liu et al. (2019, 2020) presented a multi-task deep neural network for learning representations across multiple natural language understanding tasks.

- **RI-Single, EI-Single**: the proposed model is simplified a single task model for rhetoric or emotion identification.

- w/o ***RheFusing***, w/o ***EmoFusing***: the proposed model without predicted rhetorical or emotional distribution.

- w/o ***Gate***: the proposed model without a gate mechanism.

- w/o ***Tree***: the proposed model without syntactic representation.

- w/o ***ComprehensiveEmb***, w/o ***KnowledgeEmb***: the proposed model without *comprehensive* or *sememes knowledge* embedding in Tree-LSTMs.

3.3 Experimental Setting

The version "bert-base-chinese" was employed for the BERT module in the framework. The dimensionality of word embeddings in Tree-LSTMs was set 200. We carefully tuned and specified the values of the dimensionalities of Tree-LSTMs hidden state and feature representation of rhetoric identification as 64. We employ a heuristic threshold method to specify and predict rhetorical and emotional labels for each sentence, where different threshold scores were evaluated by the grid search method and selected as 0.88 and 0.73 on validation sets, respectively.

The joint one-versus-all cross-entropy loss was used to train the proposed multi-task learning. We applied the Adamax optimizer (Kingma and Ba, 2014) with learning rate as 0.00005, the batch size was set as 6, and the training epoch was set as 15. In addition, we applied dropout (Srivastava et al., 2014) on per layer of BERT model and prediction layers, and the value was 0.1.

3.4 Experimental Results

We compare the proposed method with state-of-the-art multi-task learning methods. Table 3 shows the results of rhetoric and emotion identification.

In particular, the proposed REI-MUL model achieves the best performance in terms of F1 for both tasks. Generally, the disadvantage of CNN-Adversarial-MUL may lie in the fact that the sentence representation by CNN module is not as good as the semantic representation of the BERT module of the proposed model. Compared to BERT-MUL, the proposed REI-MUL attains better results. This may suggest that the learning from associations

Model	Rhetoric(%)			Emotion(%)		
	P	R	F1	P	R	F1
CNN-Adversarial-MUL	89.56	87.80	88.67	87.16	86.57	86.86
BERT-MUL	92.11	93.32	92.71	88.27	87.67	87.97
REI-MUL	94.88	92.81	**93.83**	91.05	86.74	**88.84**

Table 3: The rhetoric and emotion identification results (P: Precision, R: Recall, F1: F1-score). Our proposed method is significantly better than the baselines given p-value 0.05 for the emotion identification.

Model	Rhetoric(%)			Emotion(%)		
	P	R	F1	P	R	F1
REI-MUL	94.88	92.81	**93.83**	91.05	86.74	**88.84**
RI-Single/EI-Single	92.84	93.79	93.31	89.74	87.76	88.74
w/o RheFusing	94.80	92.59	93.68	90.90	86.06	88.42
w/o EmoFusing	91.40	94.12	92.74	91.63	85.98	88.71
w/o Gate	93.77	92.88	93.32	88.63	88.26	88.45
w/o Tree	94.55	91.94	93.23	91.89	85.14	88.38
w/o ComprehensiveEmb	93.88	92.52	93.20	90.69	86.40	88.49
w/o KnowledgeEmb	92.66	93.54	93.10	88.95	88.34	88.64

Table 4: The ablation study results of rhetoric and emotion identification via the proposed model.

between rhetoric and emotion categories is really beneficial for rhetoric and emotion tasks. In addition, the improved results of rhetoric detection show the independent rhetorical classifiers for each rhetoric detection are effective.

3.5 Ablation Study

We designed the ablation study to evaluate the impact of different parts of the proposed REI-MUL model for rhetoric and emotion identification. Specifically, we removed each of the following modules from our proposed model, i.e., correlation between two tasks, independent rhetorical classifier based on gating mechanism, word embedding of Tree-LSTMs. Table 4 shows the comparison results of REI-MUL and reduced models on annotated dataset. Clearly, these results manifest remarkable capability of each part of the proposed model.

Table 5 shows the detailed identification results for each rhetorical category. Experimental results show that the proposed method performs well for minority categories, such as parallelism and rhetorical questions. In addition, the results for personification and irony are not as good as the rest, which may lie in their higher dependence on human cognition and pose a challenge for the identification.

Category	P	R	F1
similes/metaphors	95.50	97.55	96.51
parallelism	94.21	97.44	95.80
personification	91.44	75.66	82.81
rhetorical questions	96.24	97.81	97.02
irony	96.97	78.05	86.49
All (macro)	94.87	89.30	91.72

Table 5: The identification results via the proposed model for each rhetorical category (%).

4 Conclusion

In this paper, we have proposed a new multi-task and multi-label learning model, which can exploit the task correlation to jointly address the rhetoric and emotion identification in a unified framework. The experimental results show the benefit of the proposed model over the state-of-the-art baselines.

Acknowledgments

The authors would like to thank all anonymous reviewers for their valuable comments and suggestions which have significantly improved the quality of this work. The work is supported by the National Key Research and Development Project of China (2018YFB1005103) and the National Natural Science Foundation of China (61632011,62076158)
.

References

Ying Chen, Wenjun Hou, Xiyao Cheng, and Shoushan Li. 2018. Joint learning for emotion classification and emotion cause detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 646–651.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Jean-Benoit Delbrouck, Noé Tits, Mathilde Brousmiche, and Stéphane Dupont. 2020. A transformer-based joint-encoding for emotion recognition and sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1–7.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George A Kennedy. 2009. *A new history of classical rhetoric*. Princeton University Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Roman Klinger, Orphee De Clercq, Saif Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 31–42.

Lizhen Liu, Xiao Hu, Wei Song, Ruiji Fu, Ting Liu, and Guoping Hu. 2018. Neural multitask learning for simile recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1543–1553.

Pengfei Liu, Xipeng Qiu, and Xuan-Jing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1–10.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, and Jianfeng Gao. 2020. The microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 118–126.

Rui Mao, Chenghua Lin, and Frank Guerin. 2019. End-to-end sequential metaphor identification inspired by linguistic theories. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3888–3898.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2049–2058.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 175–180.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

K S Tai, R Socher, and C D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1556–1566.

Yang Wang. 2013. *Chinese Rhetoric*. Shanghai Jiao Tong University Press (in Chinese).

Zhi Wen, Yang Li, Suge Wang, Jian Liao, and Xin Chen. 2019. Feature enhanced CNN for rhetorical questions identification. *Journal of Chinese information processing (in Chinese)*, 33(1):68–76.

Dongyu Zhang, Hongfei Lin, Liang Yang, Shaowu Zhang, and Bo Xu. 2018. Construction of a chinese corpus for the analysis of the emotionality of metaphorical expressions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 144–150.

Exploring the Role of Context in Utterance-level Emotion, Act and Intent Classification in Conversations: An Empirical Study

Deepanway Ghosal[†], **Navonil Majumder**[†], **Rada Mihalcea**[△], **Soujanya Poria**[†]

[†] Singapore University of Technology and Design, Singapore
[△] University of Michigan, USA
deepanway_ghosal@mymail.sutd.edu.sg
{navonil_majumder, sporia}@sutd.edu.sg
mihalcea@umich.edu

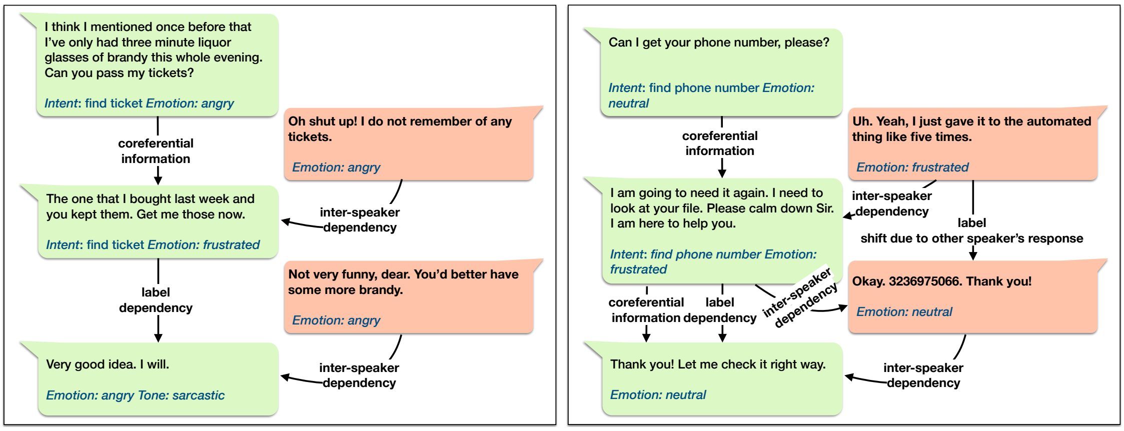

Figure 1: Role of Context in Utterance Level Dialogue Understanding.

Abstract

The recent abundance of conversational data on the Web and elsewhere calls for effective NLP systems for dialogue understanding. Complete utterance-level understanding often requires context understanding, partly defined by the nearby utterances and by the user intention and background. In recent years, a number of context-aware approaches have been proposed for various utterance-level dialogue understanding tasks. In this paper, we explore and quantify the role of context for different aspects of a dialogue, namely *emotion, dialogue act,* and *intent* identification, using state-of-the-art dialogue understanding methods as baselines. Specifically, we employ various perturbations to distort the context of a given utterance and study its impact on the different tasks and baselines. This provides us with insights into the fundamental context factors that have immediate implications on different aspects of a dialogue. Such insights may inspire more effective dialogue understanding models and provide support for future text generation approaches.

1 Introduction

Human-like conversational systems are a long-standing goal of Artificial Intelligence (AI). However, the development of such systems is not a trivial task, as we often participate in dialogues by relying on several contextual factors such as emotions, prior assumptions, intent, or personality traits. It is thus not surprising that the landscape of dialogue understanding research embraces several challenging tasks, such as, emotion recognition in conversations (ERC), dialogue intent classification, user-state representation, and others. These tasks are often performed at utterance level and can be conjoined together under the umbrella of *utterance-level dialogue understanding*. Due to the fast-growing research interest in dialogue understanding, several novel approaches have recently been proposed (Qin et al., 2020; Rashkin et al., 2019; Xing et al., 2020; Lian et al., 2019; Saha et al., 2020) to address the tasks by adopting speaker-specific and contextual modeling. However, to the best of our knowledge, the role of context has not been thoroughly explored across these tasks, partly due also to the lack of an unified framework across various utterance-level dialogue understanding tasks. In this work, we explore the role of context in utterance-level dialogue understanding. We use a contextual utterance-level dialogue understanding baseline (bcLSTM (Poria et al., 2017)) as

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1435–1449
August 1–6, 2021.

a strong baseline for the six dialogue-understanding tasks in four datasets. We propose several unique context probing strategies and experimental designs that test and measure: (1) speaker-specific context; (2) context order; (3) paraphrased context; (4) label shifts; (5) role of CRF in the sequence tagging of utterances in a dialogue. These strategies can be easily adapted for other tasks for similar purposes and provide insights into the development of new approaches to address these tasks.

Task Definition: Given a conversation along with speaker information of each constituent utterance, the utterance-level dialogue understanding task aims to identify the label of each utterance from a set of predefined labels that can be a set of emotions, dialogue acts, intents etc. Fig. 1 illustrates one such conversation between two people, where each utterance is labeled by emotion and intent. Formally, given the input sequence of N utterances $[(u_1, p_1), (u_2, p_2), \dots, (u_N, p_N)]$, where each utterance $u_i = [u_{i,1}, u_{i,2}, \dots, u_{i,T}]$ consists of T words and spoken by party p_i, the task is to predict the label e_i of each utterance u_i. The classifier can make use of the conversational context in the process.

2 Models

We train our classification models in an end-to-end setup. We first extract utterance level features with a CNN module with pretrained GloVe vectors. The resulting features are non-contextual in nature as they are obtained from utterances without the surrounding context. We then classify the utterances with one of the following two models: i) Logistic Regression, or ii) bcLSTM. Among these, the Logistic Regression model is non-contextual in nature, whereas the bcLSTM is contextual. We expand on the feature extractor and the classifier in detail next.

2.1 Utterance Feature Extractor

Utterance level features are extracted using the following method:

GloVe CNN. A convolutional neural network (Kim, 2014) is used to extract features from the utterances of the conversation. We use a convolutional layer followed by max-pooling and a fully-connected layer to obtain the representation of the utterance. Each word in the utterances is initialized with 300d pretrained GloVe embeddings (Pennington et al., 2014). We pass these to convolutional filters of sizes 1, 2, and 3, each having 100 feature maps. The output of these filters are then max-pooled across all the words of an utterance. These are then concatenated and fed to a 100 dimensional fully-connected layer with ReLU activation (Nair and Hinton, 2010). The output after the activation form the final representation of the utterance.

2.2 Utterance Classifier

The representations obtained from the *Utterance Feature Extractor* are then classified using one of the following two methods:

Without Context Classifier. In this model, classification of an utterance is performed using a fully connected multi-layer perceptron layer. This classification setup is non-contextual in nature as there is no flow of information from the contextual utterances. We call this model **GloVe CNN**.

GloVe bcLSTM. The Bidirectional Contextual LSTM model (bcLSTM) (Poria et al., 2017) creates context-aware utterance representations by capturing the contextual content from the surrounding utterances using a Bi-directional LSTM (Hochreiter and Schmidhuber, 1997) network. bcLSTM is a strong contextual utterance-level dialogue understanding baseline, with consistent performance across all six dialogue-understanding tasks considered in this work. In our experiments, on an average bcLSTM is only 1% worse than the state of the art across the six tasks that we address in this work. As opposed to more complicated models (Majumder et al., 2019; Qin et al., 2020; Zhong et al., 2019), the simpler architecture of bcLSTM is devoid of complicated interactions amongst the contextual utterances, as attention. This enables easier interpretation of the effects of the perturbations of the context.

The feature representations extracted by the *Utterance Feature Extractor* serve as the input to the bcLSTM network. Finally, the context-aware utterance representations from the output of the bcLSTM are used for the label classification. The bcLSTM model is speaker independent as it does not model any speaker level dependency. In our implementation, we add a residual connection between the first and the output from the final layer to improve the network's stability. We call this model **GloVe bcLSTM**.

Why GloVe based CNN and LSTM-based Models: In this study, we consider GloVe CNN, GloVe bcLSTM and set up different scenarios to analyze them because these models are conceptually much more straightforward than other state-of-the-art models such as DialogueRNN (Majumder et al., 2019) and DialogueGCN (Ghosal et al., 2019). For example, DialogueRNN also tracks the speaker states in addition to context. Thus, perturbations in the input would influence speaker modeling along with context modeling. This results in more complex deviations than bcLSTM, and are more difficult to analyze. Simple models are likely to be more interpretable. E.g., owing to DialogueRNN's complexity, we need to perform different levels of ablation studies to explain its behavior.

Furthermore, we use GloVe embeddings as recent transformer based models such as BERT (Devlin et al., 2018) is trained using the masked language model (MLM) objective that is already very powerful in modeling cross sentential context representation as demonstrated by other works (Liu et al., 2019; Lewis et al., 2019). Hence, to conduct a fair comparison between non-contextual and contextual models and further, for an easier apprehension on the role of contextual information in utterance-level dialogue understanding, we resort to the GloVe CNN and LSTM-based models. Additionally, as we perform a number of analysis studies, the GloVe based models were computationally much more efficient and faster to train and analyze.

3 Experimental Setup

3.1 Datasets

All the dialogue classification datasets that we consider in this work consists of two-party conversations in English language. We benchmark the models on the following datasets (see Table 1):

IEMOCAP (Busso et al., 2008) is a dataset of two person conversations among ten different unique speakers. The train set dialogues come from the first eight speakers, whereas the test set dialogues are from the last two. Each utterance is annotated with one of the following six emotions: *happy, sad, neutral, angry, excited,* and *frustrated*.

Dataset	# dialogues			# utterances		
	train	val	test	train	val	test
IEMOCAP	108	12	31	5163	647	1623
DailyDialog	11,118	1,000	1,000	87,179	8,069	7,740
MultiWOZ	8,438	1000	1,000	113,556	14,748	14,744
Persuasion	220	40	40	7902	1451	1511

Table 1: Statistics of splits and evaluation metrics used in different datasets. *Neutral** classes constitutes to 83% of the DailyDialog dataset. These are excluded when calculating the metrics in DailyDialog.

DailyDialog (Li et al., 2017) covers various topics about our daily life and follows the natural human communication approach. All utterances are labeled with both emotion categories and dialogue acts. The emotion can belong to one of the following seven labels: *anger, disgust, fear, joy, neutral, sadness*, and *surprise*. The dataset has over 83% *neutral* labels and these are excluded during Macro-F1 evaluation. In comparison, the dialogue act label distribution is relatively more balanced. The act labels can belong to the following four categories: *inform*, *question*, *directive*, and *commissive*.

MultiWOZ (Budzianowski et al., 2018) or Multi-Domain Wizard-of-Oz dataset is a fully-labeled collection of human-human conversations spanning over multiple domains and topics. The dataset has been created for task-oriented dialogue modelling and has 10,000 dialogues, which is atleast an order bigger than previously available task-oriented corpora. The dialogues are labelled with belief states and actions. It contains conversations between an user and a system from the following seven domains: restaurant, hotel, attraction, taxi, train, hospital and police. Here we focus on classifying the intent of the utterances from the user which belong to one of the following categories: *book restaurant*, *book train*, *find restaurant*, *find train*, *find attraction*, *find bus*, *find hospital*, *find hotel*, *find police*, *find taxi*, and *None*. The *None* utterances are not included in evaluation. Note that, utterances from the system side are not labelled and thus are not classified in our framework.

Persuasion For Good (Wang et al., 2019) dataset is a persuasive dialogue dataset where one participant aims to persuade the other participant to donate his/her earning using different persuasion strategies. The two participants are denoted as *Persuader* aka **ER** and *Persuadee* aka **EE** respectively. In this work, we formulate our problem to classify the utterances of Persuader and Persuadee separately using the full context of the conversation. This task can also be considered as a dialogue act classification task. The Persuader strategies are to be classified into the following eleven categories: *donation-information*, *logical-appeal*, *personal-story*, *foot-in-the-door*, *credibility-appeal*, *emotion-appeal*, *personal-related-inquiry*, *source-related-inquiry*, *self-modeling*, *task-related-inquiry*, and *non-strategy-acts*. The strategy can belong to one of the following thirteen categories for Persuadee,: *disagree-donation-more*, *ask-org-info*, *agree-donation*, *provide-donation-amount*,

Model	IEMOCAP	DailyDialog		MultiWOZ	Persuasion	
	Emotion	Emotion	Act	Intent	ER	EE
GloVe CNN	51.08	38.72	71.20	84.64	54.44	39.95
GloVe bcLSTM	61.90	**41.16**	**79.46**	**96.22**	**56.28**	**44.83**
w/o Inter-Speaker Dependency	**63.73**	39.99	74.50	95.05	53.24	40.63
w/o Intra-Speaker Dependency	56.45	35.93	78.69	95.75	52.23	38.93

Table 2: Classification performance in test data for the different tasks. Utterances from other speakers and the same speaker are absent respectively in the *w/o inter* and *w/o intra* settings. Scores are W-Avg F1 in IEMOCAP Emotion and MultiWOZ Intent; Macro F1 in the rest. All scores are average of 20 different runs. Test F1 scores are calculated at best validation F1 scores.

disagree-donation, *personal-related-inquiry*, *task-related-inquiry*, *ask-donation-procedure*, *negative-reaction-to-donation*, *positive-reaction-to-donation*, *ask-persuader-donation-intention*, *neutral-reaction-to-donation*, and *other-acts*.

3.2 Evaluation Metrics

In our experiments, we use Weighted average (W-Avg) F1 score in IEMOCAP emotion and MultiWOZ intent classification. For the other tasks – DailyDialog emotion, DailyDialog act, Persuader and Persuadee strategy classification – the label distribution is highly imbalanced, hence we report Macro F1 scores. In DailyDialog emotion classification, neutral labels are excluded (masked) while calculating the metrics. However, these utterances are still passed in the input of the different models.

4 Analysis

4.1 Speaker-specific Context Control

We first report the performance of the baseline GloVe CNN and GloVe bcLSTM model in the first two rows of Table 2. To further evaluate the intra- and inter-speaker dependence and relation across the different tasks in the GloVe bcLSTM model, we adopted two different settings as follows –

- **w/o Inter-Speaker Dependency:** when classifying a target utterance from speaker A, we drop the utterances of the speaker B from the context and vice versa.
- **w/o Intra-Speaker Dependency:** when classifying a target utterance from speaker A, we only keep utterances of the speaker B and drop all other utterances of speaker A from the context and vice versa.

Utterances of the Non-target Speaker are Important. The first setting coerces LSTM to only rely on the *target* speaker's (speaker of the target utterance) context in prediction. The results are reported in Table 2. As expected, performance drops are observed for all the datasets but IEMOCAP for emotion recognition, reinforcing the fact that the contextual utterances from the non-target speakers are important. Performance drop in DailyDialog dataset for act classification is noticeably the steepest. In the IEMOCAP dataset, we observe a pattern of the speakers maintaining the same emotion along a dialogue. This suggests that the speakers in the IEMOCAP dataset repeat the same emotion along consecutive utterances. Consequently, this induces a dataset bias. Hence, unlike the task of dialogue generation where the role of listener's utterance is key in generating speaker's response, we suspect in the case of emotion recognition in IEMOCAP dataset, removing other interlocutor's utterances from the context makes it easier and less confusing for the LSTM-based model to learn relevant contextual representations for the prediction. Contrary to this, although existing, repetitions of same or similar emotions in consecutive utterances of a speaker are less prevalent for emotion recognition in the DailyDialog dataset.

Utterances of the Target Speaker are *also* Important. 'w/o Intra-Speaker Dependency' scenario reported in Table 2 exhibits the importance of the utterances of the non-target speaker in the classification of the target utterance. In DailyDialog act and MultiWOZ intent classification, even when we remove the contextual utterances from the same speaker, the utterances from the non-target speaker provides key contextual information as evidenced by the performance in the 'w/o Intra-Speaker Dependency' setting. In those tasks, dropping the utterances of the non-target speaker results in more performance degradation as compared to the case when utterances from the target speaker are removed from the target utterance's context. This observation also supports the dialogue generation works (Zhou et al., 2017) that mainly consider previous utterances of the non-target speaker as the context for response generation. For emotion classification in DailyDialog and strategy classification in Persuasion For Good, the results obtained from 'w/o Intra-Speaker Dependency' setting are also relatively lesser compared to the baseline bcLSTM setting. This confirms the higher contextual salience of the target speaker's utterances over the non-target speaker's utterances for these particular tasks. In the case of the IEMOCAP emotion classification, removing the target speaker's utterances from the context causes a substantial performance dip for the reasons stated earlier.

Interestingly, the ‘w/o Inter-Speaker Dependency’ setting in the DailyDialog dataset manifests two distinct trends for two different tasks – act classification and emotion recognition. While non-target speakers’ utterances carry a little value for emotion recognition, they are extremely beneficial for act classification. This calls for task-specific context modeling techniques which should be the focus of the future works.

Key Takeaways of this Experiment. Although both target and non-target speakers’ utterances are useful in several utterance-level tasks, we observe some divergent trends in some of the tasks in our experiments. Hence, we surmise that a task-agnostic unified context model may not be optimal in solving all the tasks. In the future, we should strive for task-specific contextual models as each task can have unique features that make it distinct from others. One can also think of multi-task architectures where two tasks can corroborate each other in improving the overall performance.

Logically, dropping contextual utterances in a dialogue leads to inconsistency in the context and consequently, it should degrade the performance of a model that relies on the context for inference. Hence, given an unmodified dialogue flow, an ideal contextual model is expected to refer to the right amount of contextual utterances relevant in inferring the label of a target utterance. In contrast, bcLSTM shows performance improvement for IEMOCAP emotion classification when utterances from the non-target speaker are dropped (refer to the ‘w/o Inter-Speaker Dependency’ row in Table 2). The performance does not change much for dialogue act and intent classification in the DailyDialog and MultiWOZ, respectively, when we drop utterances of the target speaker. These contrasting results indicate a potential drawback of the bcLSTM model in efficiently utilizing contextual utterances of both interlocutors in unmodified dialogues for the above mentioned tasks.

4.2 Classification in Shuffled Context

To analyze the importance of context, we shuffle the utterance order of a dialogue and try to classify the correct label from the shuffled sequence. For example, a dialogue having utterance sequence of $\{u_1, u_2, u_3, u_4, u_5\}$ is shuffled to $\{u_5, u_1, u_4, u_2, u_3\}$. This shuffling is carried out randomly, resulting in an utterance sequence whose order is different from the original sequence.

Context Shuffling Strategy			**IEMOCAP**	**DailyDialog**		**MultiWOZ**	**Persuasion**	
Train	Val	Test	Emotion	Emotion	Act	Intent	ER	EE
✗	✗	✗	**61.90**	**41.16**	**79.46**	**96.22**	**56.28**	**44.83**
✓	✓	✗	59.74	36.87	74.88	91.34	54.91	41.52
✗	✗	✓	57.63	34.58	66.81	67.91	50.69	37.17
✓	✓	✓	59.82	37.69	74.62	90.78	53.60	40.96

Table 3: Test performance of GloVe bcLSTM models in the different tasks for various shuffling strategies. In Train, Val, Test column ✓denotes shuffled context and ✗denotes unchanged context. Scores are W-Avg F1 in IEMOCAP Emotion and MultiWOZ Intent; Macro F1 in the rest. Test F1 scores are calculated at best validation F1 scores.

We design three such shuffling experiments: i) dialogues in train and validation sets are shuffled, test set is unchanged, ii) dialogues in train and validation sets are kept unchanged, but dialogues in test set are shuffled, iii) dialogues in train, validation and test sets are all shuffled.

We analyze these shuffling strategies in the GloVe bcLSTM model. In theory, the recurrent nature of the LSTM model allows it to be capable of modelling contextual information from the beginning of the sequence to the very end. However, when classifying an utterance, the most crucial contextual information comes from the neighbouring utterance. In an altered context, the model would find it difficult to predict the correct labels because the original neighbouring utterances may not be in immediate context after shuffling. This kind of perturbation would make the context modelling less efficient, and performance is likely to drop compared to their non-shuffled context counterparts. This is empirically shown in Table 3.

We observe that, whenever there is some shuffling in train, validation, or test set, the performance decreases a few points in all the datasets across all tasks and evaluation metrics. Notably, the performance drop is highest when the dialogues in train, validation sets are kept unchanged and dialogues in test set are shuffled. Note that, the result for this shuffling strategy (only test set is shuffled) in MultiWOZ stands at 67.91%, much lower than the original baseline of 96.22%. This is because, the test score of 67.91% is reported at the best validation score, even though we obtain better test scores at the initial epochs of training (around 78%).

Our reported results and observations are contradictory to the claims made by Sankar et al. (2019). According to Sankar et al. (2019), the shuffling of contextual utterances does not affect the response generation performance of a seq2seq model. There can be a number of reasons for these two contradicting observations: 1) first, the characteristics of utterance labels in a dialogue are different

#	Attack Method	Strategy Past	 Future	 Target	Window	**IEMOCAP** Emotion	**DailyDialog** Emotion	 Act	**MultiWOZ** Intent	**Persuasion** ER	 EE
1	-	-	-	-	-	61.90	41.16	79.46	96.22	56.28	44.83
2	PA	✓	✗	✗	3	61.09	40.82	75.81	95.67	56.46	43.64
3	PA	✓	✗	✗	5	60.93	38.79	77.23	95.53	56.41	41.93
4	PA	✓	✗	✗	10	59.83	-	-	95.23	54.89	39.89
5	PA	✗	✓	✗	3	61.58	39.60	79.11	95.94	55.83	43.21
6	PA	✗	✓	✗	5	60.99	39.77	79.17	95.64	55.43	40.67
7	PA	✗	✓	✗	10	60.72	-	-	95.77	57.12	43.36
8	PA	✓	✓	✗	3	59.43	37.16	76.61	94.87	57.44	42.51
9	PA	✓	✓	✗	5	58.36	38.76	76.53	94.61	53.32	43.33
10	PA	✓	✓	✗	10	57.29	-	-	94.31	54.36	43.80
11	PA	✗	✗	✓	-	58.08	37.16	75.30	93.78	50.24	38.78
12	PA	✓	✓	✓	3	56.53	23.46	73.16	91.47	47.50	37.39
13	PA	✓	✓	✓	5	53.64	28.59	73.18	90.98	45.31	35.16
14	PA	✓	✓	✓	10	51.33	-	-	90.58	49.00	32.49

Table 4: Results for PA: *Paraphrasing-based Attack* in *utterance-based* GloVe bcLSTM model. In DailyDialog, we constrain the window size to 3 and 5 as there are an average of 8 utterances per dialogue in the dataset. Scores are W-Avg F1 in IEMOCAP Emotion and MultiWOZ Intent; Macro F1 in the rest.

Model	**IEMOCAP** Emotion	**DailyDialog** Emotion	 Act	**MultiWOZ** Intent	**Persuasion** ER	 EE
GloVe CNN	51.08	38.72	71.20	84.64	54.44	39.95
GloVe CNN PA	39.19(↓23.27)	23.82(↓39.64)	62.93(↓13.01)	70.34(↓16.89)	42.8(↓21.38)	33.59(↓15.91)
bcLSTM	61.90	41.16	79.46	96.22	56.28	44.83
bcLSTM PA	58.08(↓6.17)	37.16(↓9.71)	75.3(↓5.23)	93.78(↓2.53)	50.24(↓10.73)	38.78(↓13.49)

Table 5: Results for PA: *Paraphrasing-based Attack* in GloVe CNN model on target utterance and comparing it to bcLSTM results in Table 4. Scores are W-Avg F1 in IEMOCAP Emotion, MultiWOZ Intent; Macro F1 in the rest.

from responses—responses are subjective and not unique, however labels are usually agreed upon by the observers to some degree—, 2) second, instead of reporting qualitative results, Sankar et al. (2019) only reported the perplexity score of their experiments. As stated in (Cai et al., 2019), perplexity and BLEU scores may not correctly represent the quality of the response generation.

4.3 Attacks with Context and Target Paraphrasing

Modern machine learning systems are often susceptible to attacks that slightly perturb the input without any drastic change in the semantics. Although prevalent in images, adversarial examples also exist in neural network-based NLP applications. In the context of NLP, crafting adversarial examples would require making character-, word-, or sentence-level modifications to the input text to trick the classifier into misclassification. Paraphrasing sentences is one such method to construct effective adversarial examples (Iyyer et al., 2018). We conduct several experiments to evaluate the sensitivity of utterance-level dialogue understanding systems to input paraphrasing. It should be noted that although task-specific adversarial strategies could be adopted, we chose to use a general set of attacking strategies in order to understand the behavior of the baseline across different tasks and datasets. This also facilitates a fair comparison among the tasks and whether there is a confounding factor that differentiates one task from another under the same attacking strategies.

Method. We use the following scheme to analyze this effect:

- The input utterances are modified at word level. For this modification, an average of 3 to 4 words are selected per utterance and masked. The pretrained RoBERTa model is then used to fill the masks with the most likely candidates. The utterance with substituted words form the new input. We call this method *Paraphrasing-based Attack* (PA).
- For each utterance (u_t) in a dialogue, we take a window of w immediate neighbouring utterances (context) on which the above modifications are performed. The window is selected as follows:
  - Only past w utterances: $u_{t-w}, .., u_{t-1}$
  - Only future w utterances: $u_{t+1}, .., u_{t+w}$
  - Past w and future w utterances: $u_{t-w}, .., u_{t-1}, u_{t+1}, .., u_{t+w}$
  - Past w, future w, and the target utterance: $u_{t-w}, .., u_{t-1}, u_t, u_{t+1}, .., u_{t+w}$
  - Only the target utterance: u_t

  In the last case, the window is empty. In other cases, we experiment with window size $w = 3, 5, 10$.

We train a GloVe bcLSTM and a GloVe CNN model with unadulterated train and validation data. During evaluation, however, the context and target are paraphrased as described before. The results of these experiments for bcLSTM and GloVe CNN are shown in Table 4 and Table 5, respectively.

Observations. We observe that the *Paraphrasing-based Attack* is quite effective in fooling the classifier in a number of tasks. The classification performance progressively deteriorates with larger window sizes.

In DailyDialog act classification, *Paraphrasing-based Attack* on only future utterances doesn't affect the results at all. The classification performance still remains very close to the original score of 79.46 %. We observe that there is a strong reliance on the label and content of past utterance in this task. For example, a *question* is likely to be followed by an *inform* or another *question* and much less likely to be followed by a *commissive* utterance. Unchanged past context thus results in performance that is very close to the original setup. Attacking the past utterances combined with future and/or target utterances results in a relatively bigger performance drop. We also notice that the drop in performance is relatively much lesser than the other tasks except in MultiWOZ for intent classification. This is possibly because the act labels are mostly driven by the sentence type and hence unlikely to be affected from paraphrasing perturbations. For instance, around 30% of the act labels are of type *question*, and our attack strategy is almost guaranteed not to change an utterance with label *question* to something which might be classified as *inform*, *commissive*, or *directive*. Overall, we observe a consistent plunge in the performance when the target utterance is attacked by the *Paraphrasing-based Attack* method. For intent classification in MultiWOZ, utterances often have keywords which indicate the label (presence of *train* might indicate class label of *find train* or *book train*). In these cases, if the target utterance is not paraphrased, the model is still likely to predict the correct label. Finally, in Persuasion for Good, we observe that the attack method is slightly more effective in fooling the classifier for persuadee strategy classification.

In terms of window direction, we observe that perturbations in the past or future utterances result in a similar range of reduction in performances. One notable exception is act prediction in DailyDialog, where the model continues to perform near the original score of 79.46% irrespective of the attack in future utterances in the window.

Performance Comparison for Attacks in GloVe CNN and GloVe bcLSTM. We summarize the performance of GloVe CNN and GloVe bcLSTM models against *Paraphrasing-based Attack* in Table 5. For all the tasks, we observe a very significant drop in performance for GloVe CNN. For example, in emotion classification, the drop is around 23% and 40% for *Paraphrasing-based Attack* in IEMOCAP and DailyDialog respectively. However, for the same setting, the relative decrease in performance is only around 6% and 10% for bcLSTM. We observe the same trend in other tasks where it can be seen that the bcLSTM model is much more robust against the attack compared to the CNN model. This is because contextual models such as bcLSTM are harder to fool as the context carry key information regarding the semantics of the target and salient information can be inferred about the target using its' context. It is thus evident that even when the target utterance is corrupted, bcLSTM is capable of using contextual information to predict the label correctly, and subsequently the decline in performance is much lesser.

In principle, our findings in Table 5 can be related to how transformer-based pre-trained language models work. For example, in BERT (Devlin et al., 2018), the masked language modeling (MLM) and the next sentence prediction (NSP) objective forces the model to infer or predict the target using contextual information. Such contextual models are more powerful and robust because context information plays a crucial role in almost every natural language processing task. An objective similar to next sentence prediction in BERT or permutation language modeling in XLNET (Yang et al., 2019) can be used for conversation level pre-training to improve several downstream conversational tasks. Such approaches have been found to be useful in the past (Hazarika et al., 2019).

4.4 Performance for Label Shift

As discussed before, a few of our tasks of interest exhibit the label copying property which means consecutive utterances from the same speaker or different speakers often have the same or similar emotion, act, or intent label. The inter-speaker and intra-speaker label copying is especially prevalent in the IEMOCAP emotion task, the DailyDialog act task, and the MultiWOZ intent task. Contextual models such as bcLSTM make correct predictions when utterances display such kind of continuation of the same label. But what happens when there is a change of label? Does bcLSTM continue to perform at the same level or is it affected from the change? To understand this occurrence in more detail, we define this event as *Label Shift* and look at the following two different kind of shifts that could happen in the course of a dialogue:

- *Intra-Speaker Shift*: The label of the utterance is different from the label of the previous utterance from the same speaker.
- *Inter-Speaker Shift*: The label of the utterance is different from the label of the previous utterance from the non-target speaker.

In these two scenarios explained above, we are interested to see how bcLSTM performs at the utterances were the label shift takes place.

We report results for utterances in the test data that show *Intra-Speaker Shift* and *Inter-Speaker Shift* in Table 6. The *Inter-Speaker Shift* is not

Setup	**IEMOCAP**	**Dailydialog**		**MultiWOZ**	**Persuasion**	
	Emotion	Emotion	Act	Intent	ER	EE
Original	61.90	41.16	79.46	96.22	56.28	44.83
Intra-Speaker Shift	52.01 (13.2)	44.23 (1.0)	76.18 (2.9)	94.91 (1.6)	57.84 (6.9)	49.4 (4.7)
Inter-Speaker Shift	52.37 (22.0)	47.77 (1.3)	78.80 (4.9)	-	-	-

Table 6: Classification performance for utterances which exhibits *Label Shift* in test data. Numbers in parenthesis indicate the average count of the corresponding shifts per dialogue. There is no *Inter-Speaker Shift* in MultiWOZ or Persuasion for Good as we only classify user, persuader, or persuadee utterances. Scores are W-Avg F1 in IEMOCAP Emotion and MultiWOZ Intent; Macro F1 in the rest.

defined in MultiWOZ as we don't have intent labels for system utterances. We also don't report *Inter-Speaker Shift* results in Persuasion for Good as the persuader and persuadee strategy set is different.

The emotion labels in IEMOCAP display the largest extent of label copying. We also observe in Table 6 that label shifts occur with high frequency in IEMOCAP. These are the likely reasons why we observe significant number of errors for utterances with *Label Shift* for this task in Table 6. The performance for both *Intra-Speaker Shift* and *Inter-Speaker Shift* stands at around 52.0%, much lesser than the overall average of 61.9% in test data. Although not as strong as IEMOCAP, the intra-speaker label copying feature can also be seen in MultiWOZ intent and DailyDialog act labels. For these two tasks, we again observe a drop of performance when either *Intra-Speaker Shift* or *Inter-Speaker Shift* occurs. In contrast, the extent of transition is spread over a much larger combination of labels in DailyDialog emotion and Persuasion for Good. We observe that the results for utterances with *Label Shift* in those tasks are in fact better than the overall score. In DailyDialog emotion, the scores are 44.23% and 47.77%, which is an improvement over the original 41.16%. The scores of 57.84% and 49.4% in Persuasion for Good also stand over the scores of 56.28% and 44.83% in the original setup.

4.5 Sequence Tagging using Conditional Random Field (CRF)

On the surface, the task of utterance level dialogue understanding looks similar to sequence tagging. Are there any distinct label dependency and patterns across the tasks that are dataset agnostic and likely to be captured by CRF (Lafferty et al., 2001)? In the quest to answer this, we plug in three different CRF layers on top of the bcLSTM network.

Global-CRF. It is a linear chain CRF used on top of bcLSTM. In this setting, we do not consider speaker information.

It can be defined using the equations below:

$$P(Y|D) = \frac{1}{Z(D)} \prod_{i=1}^{n} \phi_T(y_{i-1}, y_i)\phi_E(y_i, u_i), \quad (1)$$

$$Z(D) = \sum_{y' \in \mathcal{Y}} \prod_{i=1}^{n} \phi_T(y'_{i-1}, y'_i)\phi_E(y'_i, u_i). \quad (2)$$

Global-CRF$_{\text{ext}}$. The linear-chain CRF is extended to include not only the transition potential from the previous label to the current label, but also from the prior-to-previous label. Concisely, the current label is predicated on the previous two labels. Therefore, the transition potential function ϕ_T takes one extra argument y_{i-2}. The advantage here is it also considers the previous label from the target speaker should utterance $i-2$ have come from the target speaker. This becomes useful in the tasks where the speakers tend to retain label from its last utterance. It can be defined using the equations below:

$$P(Y|D) = \frac{1}{Z(D)} \prod_{i=1}^{n} \phi_T(y_{i-2}, y_{i-1}, y_i)\phi_E(y_i, u_i), \quad (3)$$

$$Z(D) = \sum_{y' \in \mathcal{Y}} \prod_{i=1}^{n} \phi_T(y'_{i-2}, y'_{i-1}, y'_i)\phi_E(y'_i, u_i). \quad (4)$$

Speaker-CRF. In this setting, we use two distinct CRFs for the two speakers in a dialogue. Inter-speaker label dependency and transitions are not likely to be captured in this setting by the CRFs.

Negative Results. We report the results for CRF experiments in Table 7 and Table 8. Aside from the well-known sequence tagging tasks, such as, Named Entity Recognition (NER) and Part of Speech Tagging, CRF does not improve the performance of utterance-level dialogue understanding tasks. There could be multiple reasons as below:

1: A dialogue is governed by multiple variables or pragmatics, e.g., topic, personal goal, past experience, expressing opinions or presenting facts based on personal knowledge, and the role of the interlocutors. Hence, the response pattern can vary depending on these variables. The personality of the speakers add an extra layer of complexity to this which causes speakers to respond differently under the same circumstances. An identical utterance can be uttered with different emotions by two different speakers. CRF relies on surface label patterns which can vary with datasets. Due to this dynamic nature of dialogues and the presence of latent controlling variables, the label transition matrix of CRF does not learn any distinct

Methods	**IEMOCAP**	**DailyDialog**				
	Emotion	Emotion			Act	
	W-Avg F1	W-Avg F1	Micro F1	Macro F1	W-Avg F1	Macro F1
GloVe CNN	52.04	49.36	50.32	36.87	80.71	72.07
GloVe bcLSTM	61.74	52.77	53.85	39.27	**84.62**	79.12
w/o inter	**63.73**	52.39	52.86	**39.99**	81.32	74.50
w/o inter w/ speaker-CRF	62.94	52.47	54.04	39.77	81.19	74.12
w/ global-CRF	61.62	53.05	53.86	39.27	83.91	79.10
w/ global-CRF$_{\text{ext}}$	61.64	53.06	54.40	39.64	84.27	**79.25**
w/ speaker-CRF	62.21	**53.16**	**54.68**	**39.74**	84.15	79.20

Table 7: Classification performance in test data for IEMOCAP and DailyDialog using different CRF configurations. All scores are average of at least 10 different runs. Test F1 scores are calculated at best validation F1 scores.

Methods	**MultiWOZ**	**Persuasion**			
	Intent	Persuader (ER)		Persuadee (EE)	
	W-Avg F1	W-Avg F1	Macro F1	W-Avg F1	Macro F1
GloVe CNN	84.30	67.15	54.45	58.00	41.03
GloVe bcLSTM	**96.14**	**69.26**	55.27	61.18	42.19
w/o inter	95.05	67.81	53.24	59.44	40.63
w/o inter w/ speaker-CRF	94.11	68.13	54.45	58.93	40.16
w/ global-CRF/speaker-CRF	95.48	68.59	55.60	61.24	42.62
w/ global-CRF$_{\text{ext}}$	95.51	69.23	**56.80**	**61.89**	**43.68**

Table 8: Classification performance in test data for MultiWOZ and Persuasion for Good using different CRF configurations. All scores are average of atleast 10 different runs. Test F1 scores are calculated at best validation F1 scores. In MultiWOZ and Persuasion for Good, the global-CRF and speaker-CRF setting are identical as we only classify utterances coming from one of the speakers (user in MultiWOZ, persuader or persuadee in Persuasion for Good).

pattern that is complementary to what is learned by the feature extractor. **2**: Some of the datasets — IEMOCAP and MultiWOZ — contain distinct label-transition patterns between the same and distinct speakers e.g., the label copying feature in the IEMOCAP dataset where the same or similar emotions are repeated by the same or both the speakers. Similarly, in the MultWOZ dataset, the intent *book restaurant* to be frequently followed by the intent *find taxi*. **We believe the distinct label patterns in the IEMOCAP and MultiWoZ datasets is potentially one of the reasons why contextual models perform so well on these three datasets and tasks compared to the rest.** On these two daatsets, we expected bcLSTM w/ global-CRF to outperform vanilla bcLSTM. However, we do not observe any statistically significant improvement using bcLSTM w/ global-CRF over bcLSTM. We posit that the evident label-transition patterns that exist in these two datasets are straightforward to capture without a CRF. In fact, we also tried GloVe CNN with a CRF layer on it, and surprisingly the result was not significantly higher than that of GloVe CNN. This can be attributed to the absence of explicit contextual and label transition-based features in the CRF.

Results in IEMOCAP and Persuasion for Good Datasets. We observe a *minor* performance improvement in the IEMOCAP dataset using speaker-CRF for emotion recognition. This observation directly correlates to the experiment under "w/o Inter-Speaker Dependency" setting in Table 2 and can be largely attributed to the label copying feature in the IEMOCAP dataset as explained in the last paragraph. In "w/o Inter-Speaker Dependency" setting, contextual utterances of the speaker B are not utilized to classify utterances of speaker A vice versa. The results do not improve when we use speaker-level CRF on bcLSTM under the "w/o Inter-Speaker Dependency" setting. From these observations, we can conclude that CRF is not learning any distinct label dependency and transition patterns that are not learned by the feature extractor or bcLSTM alone.

Global-CRF$_{\text{ext}}$ shows significant performance improvement on the Persuasion for Good dataset. Some of the key controllable factors of the dialogues such as topics in this dataset are fixed and can be learned intrinsically by the classifier. The scope of the dialogues in this dataset is very limited as there are only two possible outcomes of the dialogues – *agree to donate*, and *disagree to donate*. Hence, there can be some label transition patterns learned by the Global-CRF$_{\text{ext}}$ using a larger label-context window in the transition potential.

5 Conclusion

In this paper, we explored the role of context for six utterance-level dialogue understanding tasks in four different datasets. Using a strong contextual baseline system (bcLSTM), we gained insights into the behavior of such contextualized models in the presence of various context perturbations. Such probes have bolstered many interesting intuitions about utterance-level dialogue understanding—the role of label dependency and future utterances; the role of speaker-specific contextual modelling; and the robustness of contextual models as opposed to their non-contextual counterparts against adversarial probes. We believe that these probing strategies can be straightforwardly adapted to other context-reliant tasks. The implementation pertaining to this work is available at `https://github.com/declare-lab/dialogue-understanding`.

Acknowledgements

This research is supported by A*STAR under its RIE 2020 Advanced Manufacturing and Engineering programmatic grant, Award No.– A19E2b0098.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. IEMOCAP: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335–359.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2019. Skeleton-to-response: Dialogue generation guided by retrieval memory. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1219–1228.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Deepanway Ghosal, Navonil Majumder, Soujanya Poria, Niyati Chhaya, and Alexander Gelbukh. 2019. Dialoguegcn: A graph convolutional neural network for emotion recognition in conversation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 154–164.

Devamanyu Hazarika, Soujanya Poria, Roger Zimmermann, and Rada Mihalcea. 2019. Emotion recognition in conversations with transfer learning from generative conversation modeling. *arXiv preprint arXiv:1910.04980*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. 2017. Residual lstm: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995.

Zheng Lian, Jianhua Tao, Bin Liu, and Jian Huang. 2019. Domain adversarial learning for emotion recognition. *arXiv preprint arXiv:1910.13807*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2019. DialogueRNN: An Attentive RNN for Emotion Detection in Conversations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6818–6825.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-Dependent Sentiment Analysis in User-Generated Videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–883, Vancouver, Canada. Association for Computational Linguistics.

Libo Qin, Wanxiang Che, Yangming Li, Mingheng Ni, and Ting Liu. 2020. Dcr-net: A deep co-interactive relation network for joint dialog act recognition and sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: a new benchmark and dataset. In *Proceedings of the Association for Computatinal Linguistics*.

Tulika Saha, Aditya Patra, Sriparna Saha, and Pushpak Bhattacharyya. 2020. Towards emotion-aided multimodal dialogue act classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4361–4372.

Chinnadhurai Sankar, Sandeep Subramanian, Christopher Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. *arXiv preprint arXiv:1906.01603*.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. *arXiv preprint arXiv:1906.06725*.

Chung-Hsien Wu, Ze-Jing Chuang, and Yu-Chung Lin. 2006. Emotion recognition from text using semantic labels and separable mixture models. *ACM transactions on Asian language information processing (TALIP)*, 5(2):165–183.

Songlong Xing, Sijie Mai, and Haifeng Hu. 2020. Adapted dynamic memory network for emotion recognition in conversation. *IEEE Transactions on Affective Computing*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Peixiang Zhong, Di Wang, and Chunyan Miao. 2019. Knowledge-enriched transformer for emotion detection in textual conversations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 165–176.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017. Emotional chatting machine: Emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*.

A Label Transitions.

To check whether there lies any patterns in the label sequences of the datasets, in Fig. 2 and 3, we plot frequency of the label pairs (x, y) where x and y are the labels of $U_{s_{t-1},t-1}$ and $U_{s_t,t}$ respectively. Figure Fig. 2 explains inter-speaker label transition and Fig. 3 illustrates the intra-speaker label transition. Both these plots reveal the same emotion labels appearing in the consecutive utterances with high frequency in the IEMOCAP dataset. This induces label dependencies and consistencies and can be called as the **label copying feature** of the dataset. From our empirical analysis in Section 4, we confirm this property of the IEMOCAP dataset. Although not as strong as IEMOCAP, the intra-speaker label copying feature is also prevalent in the MultiWOZ and DailyDialog (Act) dataset (refer to Fig. 2). Moreover, we observe interesting patterns in DailyDialog (Act). A directive utterance is commonly followed by a commissive utterance. This indicates that utterances with acts such as request and instruct (directive label) are followed by accepting/rejecting the request or order (commissive label). We also notice that an utterance with the act of questioning is commonly followed by the utterances with the act of answering (which is quite natural). Fig. 2 also corroborates the high frequent joint appearance of similar emotions in both speaker's utterances e.g., negative emotions — *anger, frustration, sad* expressed by one speaker is replied with a similar negative emotion by the other speaker. Interestingly, the DailyDialog dataset for emotion classification does not elicit any such patterns. We can attribute this to the scripted utterances present in the IEMOCAP that has specifically been designed to invoke more emotional content to the utterances. On the other hand, the DailyDialog dataset comprises naturalistic utterances that are more dynamic in nature as they depend on interlocutors' personality. In both IEMOCAP and DailyDialog datasets, the repetitions of the same emotions can be found in consecutive utterances of a speaker. The repetition of the same or similar emotions for a speaker is frequent and often forms long chains in IEMOCAP. However, such repetitions are much less prevalent in DailyDialog. Readers are referred to Fig. 3 for a clearer view. **This two different types of datasets used in this work is purposefully crafted in order to study dataset-specific nuances to attempt the same task.** In DailyDialog, approximately 80% of utterances are labeled as *no-emotion* (see Fig. 4) which poses a difficult challenge to perform emotion classification. These two datasets also differ from each other in the average dialogue length. While the average number of utterances per dialogue in the IEMOCAP dataset is more than 50, the average number of utterances per dialogue in the DailyDialog dataset is just 8 which is much shorter.

Among other semantically plausible label transitions, we can see in Fig. 3, the intent *book restaurant* to be frequently followed by the intent *find taxi* in the MultiWOZ dataset. **We believe this is potentially one of the reasons why contextual models perform so well on these three datasets and tasks** compared to the rest which we discuss in the subsequent sections. Further, label dependency and consistency can aid filtering likely labels given the prior labels. Notably, such patterns are not visible in the other datasets. Hence, one can use Conditional Random Field (CRF) to find any hidden label patterns and dependencies.

B Utterance Classifier

cLSTM. Similar to bcLSTM but without the bidirectionality in the LSTM, this model is intended to ignore the presence of future utterances while classifying an utterance U_t.

DialogueRNN. (Majumder et al., 2019) is a recurrent network based model for emotion recognition in conversations. It uses two GRUs to track individual speaker states and global context during the conversation. Further, another GRU is employed to track emotion state through the conversation. In this work, we consider the emotion state to be a general state which can be used for utterance level classification (i.e., not limited to only emotion classification). Similar to the bcLSTM model, the features extracted by the *Utterance Feature Extractor* is the input to the DialogueRNN network. DialogueRNN aims to model inter-speaker relations and it can be applied on multiparty datasets.

cLSTM, bcLSTM and DialogueRNN with Residual Connections. Deep neural networks can often have difficulties in information proropagation. Multi-layered RNN-like in particulars often succumb to vanishing gradient problems while modeling long range sequences. Residual connections or skip connections (He et al., 2016) are an intuitive way to tackle this problem by improving information propagation and gradient flow. Inspired by the early works in residual LSTM (Wu

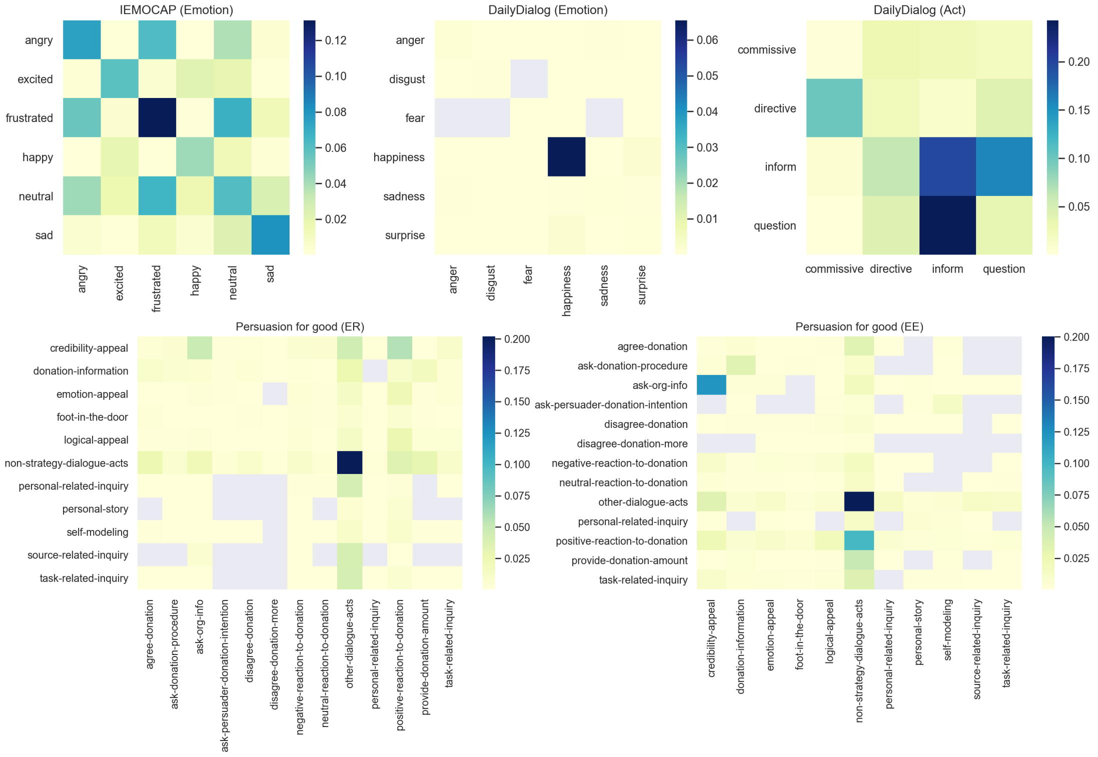

Figure 2: The heatmap of inter-speaker label transition statistics in the datasets. The color bar represents normalized number of inter-speaker transitions such that elements of each matrix add up to 1. Inter-speaker transitions are not defined in MultiWOZ as system side utterances are not labeled. Note: For the DailyDialog dataset, we ignore the *neutral* emotion in this figure.

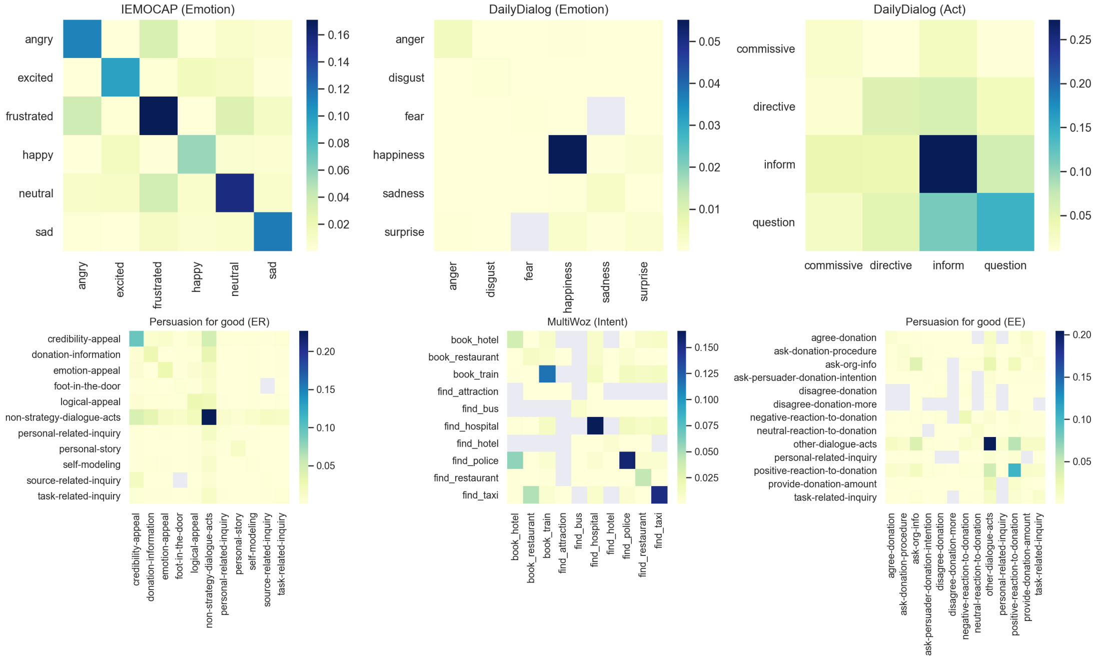

Figure 3: The heatmap of intra-speaker label transition statistics in the datasets. The color bar represents normalized number of intra-speaker transitions such that elements of each matrix add up to 1. Note: For the DailyDialog dataset, we ignore the *neutral* emotion in this figure.

et al., 2006; Kim et al., 2017), in our recurrent contextual models - bcLSTM and DialogueRNN we adopt a simple strategy to introduce a residual connection. For each utterance, a residual connection is formed between the output of the feature extractor and the output of the bcLSTM/DialogueRNN module. These two vectors are added and the final classification is performed from the resultant

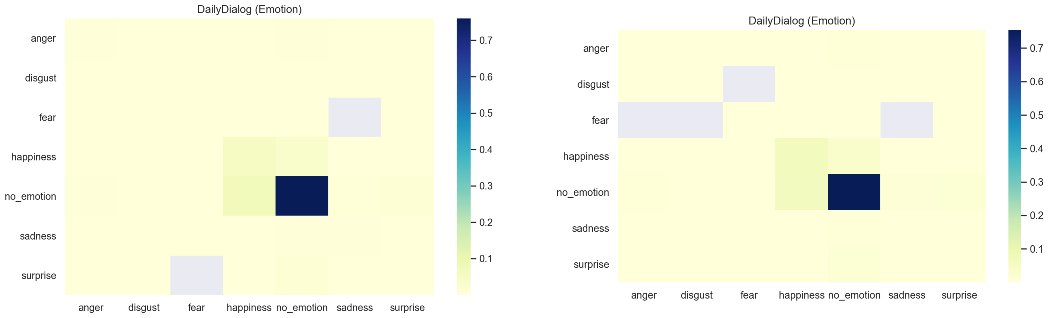

Figure 4: The heatmap of intra-speaker (left) and inter-speaker (right) label transition statistics in the DailyDialog dataset including *neutral* emotion. The color bar represents normalized number of inter-speaker and intra-speaker transitions such that elements of each matrix add up to 1.

vector.

B.1 Results

Methods	**IEMOCAP**	**DailyDialog**				
	Emotion	Emotion			Act	
	W-Avg F1	W-Avg F1	Micro F1	Macro F1	W-Avg F1	Macro F1
GloVe CNN	52.04	49.36	50.32	36.87	80.71	72.07
GloVe cLSTM	59.10	52.56	53.67	38.14	83.90	78.89
w/o Residual	55.07	52.56	53.26	38.12	84.06	78.54
GloVe bcLSTM	61.74	52.77	53.85	39.27	84.62	79.12
w/o Residual	58.32	54.74	**56.32**	39.24	84.10	78.98
GloVe DialogueRNN	**62.57**	**55.18**	55.95	**41.80**	**84.71**	**79.60**
w/o Residual	61.32	54.50	55.29	40.05	83.98	79.16
RoBERTa LogReg	54.12	52.63	52.42	40.02	82.55	75.62
RoBERTa bcLSTM	62.72	56.05	56.77	43.26	85.17	82.16
w/o Residual	62.86	55.92	57.32	43.03	**86.35**	80.69
RoBERTa DialogueRNN	**64.12**	**59.07**	**59.50**	**45.19**	86.31	**82.20**
w/o Residual	63.96	57.57	57.76	44.25	86.28	82.08

Table 9: Classification performance in test data for emotion prediction in IEMOCAP, emotion prediction in DailyDialog, and act prediction in DailyDialog. Scores of the Glove-based models are reported after averaging 20 different runs. RoBERTa-based models were run 5 times and we report the average scores. Test F1 scores are calculated at best validation F1 scores.

Methods	**MultiWOZ**	**Persuasion**			
	Intent	Persuader (ER)		Persuadee (EE)	
	W-Avg F1	W-Avg F1	Macro F1	W-Avg F1	Macro F1
GloVe CNN	84.30	67.15	54.33	58.00	41.03
GloVe cLSTM	95.03	68.75	54.36	59.46	41.62
w/o Residual	95.12	64.62	49.08	54.87	36.36
GloVe bcLSTM	96.14	**69.26**	55.27	**61.18**	**42.19**
w/o Residual	96.21	67.20	52.75	55.02	37.72
GloVe DialogueRNN	**96.32**	68.96	**56.29**	61.11	42.18
w/o Residual	96.08	68.77	54.20	58.72	39.06
RoBERTa LogReg	85.70	71.98	60.36	63.45	**51.74**
RoBERTa bcLSTM	95.46	71.85	61.05	64.14	50.11
w/o Residual	**95.61**	71.06	58.72	62.73	44.74
RoBERTa DialogueRNN	**95.61**	**72.91**	**62.03**	**64.33**	49.22
w/o Residual	95.29	72.45	60.49	64.21	49.71

Table 10: Classification performance in test data for intent prediction in MultiWOZ, persuader and persuadee strategy prediction in Persuasion for Good. Scores of the Glove-based models are reported after averaging 20 different runs. RoBERTa-based models were run 5 times and we report the average scores. Test F1 scores are calculated at best validation F1 scores.

We report results for IEMOCAP, DailyDialog dataset in Table 9 and MultiWOZ, Persuasion for Good dataset in Table 10. We ran each experiment multiple times and report the average test scores based on the best validation scores.

We observe that there is a general trend of improvement in performance when moving to the RoBERTa based feature extractor from the GloVe CNN feature extractor except in the intent prediction task in MultiWOZ dataset. As the RoBERTa model has been pre-trained on a large amount of textual data and has considerably more parameters, this improvement is expected. The results could possibly be improved even more if a RoBERTa-Large model is used instead of the RoBERTa-Base model that we use in this work.

We also observe that contextual models — bcLSTM and DialogueRNN perform much better than the non-contextual Logistic Regression models in most cases. Context information is crucial for emotion, act, and intent classification and models such as bcLSTM or DialogueRNN are some of the most prominent methods to model the contextual dependency between utterances and their labels. In IEMOCAP, DailyDialog and MultiWOZ there is a sharp improvement in performance in contextual models compared to the non-contextual models. However, for the strategy classification task in *Persuasion for Good dataset*, the improvement in contextual models is relatively lesser. Notably, for Persuadee classification, the RoBERTa non-contextual model achieves the best result, outperforming the contextual models. Without the presence of residual connections, the *GloVe cLSTM* and *GloVe bcLSTM* baselines perform poorly than the non-contextual *GloVe CNN* baseline in the *Persuasion for Good dataset*. This beckons the need for better contextual models for this dataset. To analyze the results of the different models we look at the following aspects:

Importance of the Residual Connections in the Models. It is also to be noted that the introduction of the residual connections generally improves the performance of the contextual models. We obtain better performance and improved stability during training for most of the models with residual connections. In particular, residual connections are mostly effective in IEMOCAP and Persuasion for Good datasets that comprise long dialogues. Residual connections are used in deep networks to aid information propagation and tackle vanishing gradient problems (Wu et al., 2006; Kim et al., 2017) in RNNs by improving gradient flow. As multi-layered RNN-like architectures often find it difficult to model long-range dependencies in a sequence due to vanishing gradient problems (Pascanu et al., 2013), we conjecture, that could be one of the reasons why we see a great performance boost with residual connections by helping propagate key information form the CNN layers to the output of LSTM layers that might be lost due to the long deep sequence modeling in the LSTM layer. Residual connections also help in combating vanishing gradient issues by improving gradient flow. Unlike IEMOCAP and Persuasion for Good, in DailyDialog and MultiWOZ datasets, the improvement in performance caused by the residual connections is only little which can be attributed to the relatively shorter dialogues present in these two datasets.

Variance in the Results. As deep learning models tend to yield varying results across multiple training runs, we trained each model multiple times and report the average score in Table 9 and Table 10. In general, we observed that the RoBERTa-based models show lesser variance compared to the GloVe-based models.

Variance in the Glove-based models: The observed variance is higher for emotion classification in IEMOCAP and DailyDialog as compared to act and intent classification in DailyDialog and MultiWOZ, respectively. Both baseline models – Glove CNN and bcLSTM show standard deviation of about 1.28% in the IEMOCAP dataset across different runs. In the Persuasion for Good dataset, for both persuader's and persuadee's act classification tasks, the deviation remains around 1.6% when we consider the Macro-F1 metric. However, for the Weighted-F1 metric, the performance is relatively stable as upon accumulating multiple runs the standard deviation is about 0.99% across the baselines. A similar trend is also prevalent in the DailyDialog dataset for emotion classification. In this task, the baselines – Glove CNN and bcLSTM show standard deviation of about 1.19% when Weighted-F1 and Micro-f1 are considered. According to Macro-F1 metric, however, these baselines are exposed to relatively higher standard deviation of 2.88%. This is likely to be a consequence of severe label imbalance in the dataset, that is having 80% neutral utterances. We have observed that a majority of these neutral samples do not exhibit neutral emotion. Therefore, this poor labeling quality may have precipitated this large variance in the results. On the other hand, the baseline models perform consistently in the intent and act classification tasks in MultiWOZ and DailyDialog datasets respectively showing standard deviation of around 0.55% across different runs. When comparing among the baselines, we found higher variances in the results obtained with the Glove CNN than the bcLSTM.

One possible reason behind the variances in the results of the GloVe-based models could be the end-to-end training setup that renders the model deeper. The original bcLSTM and DialogueRNN model employed a two-stage training method where the utterance feature extractor is first pretrained and then kept unchanged during the contextual model training. This setting may make those original models more stable. Similarly, we think, in our end-to-end setup, a more sophisticated training regime could result in a lesser variance of the results. For example, the utterance feature extractor could be trained only for the first few epochs and then kept frozen during subsequent epochs of the training. Due to this high variance in the end-to-end Glove-based models, the future works on these datasets and tasks which employ this setting should report the average results of multiple runs for a fair comparison of the models.

Variance in the RoBERTa-based models: The RoBERTa based models show much lesser variance in performance across different runs. In particular, the standard deviations in the results of Roberta-based bcLSTM are 0.57 on the IEMOCAP, 0.08, and 0.48 in the DailyDialog for emotion and act classification tasks, respectively, 0.07 in the MultiWoz dataset, 0.9 and 1.04 in the Persuasion for Good dataset for persuader's and persuadee's act classification tasks respectively. RoBERTa-based DialogueRNN shows a similar trend. We surmise that this is the case because the feature extractor's weights are initialized from a pretrained checkpoint. Thus, the feature extractor already provides meaningful features from the beginning of training and is not required to be trained from scratch, resulting in greater stability in the performance.

Encouraging Neural Machine Translation to Satisfy Terminology Constraints

Melissa Ailem, Jinghsu Liu and Raheel Qader
Lingua Custodia, France
{melissa.ailem,jingshu.liu,raheel.qader}@linguacustodia.com

Abstract

We present a new approach to encourage neural machine translation to satisfy lexical constraints. Our method acts at the training step and thereby avoiding the introduction of any extra computational overhead at inference step. The proposed method combines three main ingredients. The first one consists in augmenting the training data to specify the constraints. Intuitively, this encourages the model to learn a copy behavior when it encounters constraint terms. Compared to previous work, we use a simplified augmentation strategy without source factors. The second ingredient is constraint token masking, which makes it even easier for the model to learn the copy behavior and generalize better. The third one, is a modification of the standard cross entropy loss to bias the model towards assigning high probabilities to constraint words. Empirical results show that our method improves upon related baselines in terms of both BLEU score and the percentage of generated constraint terms.

1 Introduction

Neural Machine Translation (NMT) systems enjoy high performance and efficient inference (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017). However, when it comes to domain specific scenarios, where it is often necessary to take into account terminology constraints, NMT models suffer from the lack of explicit source-target correspondences making it challenging to enforce such constraints. For instance, consider the following sentence from the financial domain : *“**Holders** may submit instructions based on a minimum quantity being accepted by the **offeror.**”*. According to the financial terminology, the words *Holders* and *offeror* should be translated *porteurs* and *initiateur* respectively. Unfortunately, a generic English-French NMT model would translate the above sentence as: *“Les **titulaires** peuvent soumettre des instructions en fonction d’une quantité minimale acceptée par l’**offrant**.”*, where the words *Holders* and *offeror* are translated into *titulaires* and *offrant* respectively. To address this limitation various approaches have been proposed. They can be grouped into two categories based on whether they enforce constraints at inference or at training time. The former family of methods changes the decoding step to inject the constraint terms in the output. While effective at satisfying constraints, these techniques tend to suffer from several weaknesses such as high computational cost at the decoding stage, decreased translation quality due to strict enforcement of terminology constraints (Hokamp and Liu, 2017; Post and Vilar, 2018), or ineptness if there are multiple constraints in the input/output (Susanto et al., 2020).

The other category of methods, which we follow in this work, integrates lexical constraints during training (Dinu et al., 2019). More precisely, they augment the training data in such a way as to inform the NMT model of the constrains that need to be satisfied (Crego et al., 2016; Song et al., 2019; Dinu et al., 2019). This type of approaches has the advantage of not changing the NMT model as well as of not introducing any additional computational overheads at inference time. One limitation of these methods is their soft nature, i.e, not all constraints are guaranteed to be present in the output.

In this paper we pursue the latter line of research and improve upon the recent work of Dinu et al. (2019) by (i) only using tags –without source factors – to distinguish between constraints and other words, (ii) performing constraint token masking for robustness/generalization purposes and (iii) modifying the standard cross-entropy loss to bias the model towards generating constraint terms. Empirical results show that our approaches improve both the BLEU sore and the number of satisfied constrains compared to previous work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1450–1455
August 1–6, 2021.

2 Related Work

Existing approaches can be cataloged based on whether they integrate constraints at inference/decoding (Chatterjee et al., 2017; Hasler et al., 2018; Hokamp and Liu, 2017) or at training time (Dinu et al., 2019).

Among methods of the first category, we can mention the Grid Beam Search (GBS) algorithm, which consists in reorganizing the vanilla beam search to place constraints correctly in the output as well as infer accurately the constraints-free parts. While successful in placing constrains compared to the original BS algorithm, GBS suffers from a high decoding time, it increases inference complexity exponentially with the number of constraints. To alleviate this issue, several improvement have been proposed, such as Dynamic Beam Allocation (DBA) (Post and Vilar, 2018) and its optimized extension, namely vectorized DBA (Hu et al., 2019). Despite an important gain in computational time, these methods still significantly increase the decoding time. For instance, the method of Post and Vilar (2018) is three times slower than the constraint-free beam search. More recently, Susanto et al. (2020) rely on the levenstein transformer (Gu et al., 2019), which uses an edit-based decoder iteratively refining the output using deletion and insertion operations. To enforce constraints using this model, Susanto et al. (2020) add one step to the decoder that consists in placing constraint terms in the output, and they further disallow the deletion operation on constraint terms. Albeit effective, the main limitation of this approach is in constraint ordering – when there is more than one constraint term in the output. That is, the initial order in which constraints have been placed remains unchanged.

Different from the above, the second family of methods integrates lexical constraints at training time. For instance, Crego et al. (2016) replace the terminology terms with placeholders during training and then add them back in a post-processing step. Song et al. (2019) proposed to annotate the training set by adding the target side of the terminology terms in the source sentences. A transformer model (Vaswani et al., 2017) is then trained on this augmented training set. This training data annotation has been also explored to teach the NMT to use translation memories (Gu et al., 2018) or to enforce copy behavior (Pham et al., 2018). Dinu et al. (2019) proposed two different ways to augment the training data, namely the append and the replace approaches. The former is similar to approach proposed in (Song et al., 2019), and the second requires to replace the source term of the constraints in the source sentence by its corresponding target side in the terminology entries. This method further uses source factors in order to distinguish the constraints from the rest of the source sentence. This is the closest approach to ours. The key differences are as follows. Our method uses only tags (without source factors) to specify constraints in the training set, and we further perform constraint-token masking, which improves model robustness/generalization as supported by our experiments. Moreover, we investigate a biased cross-entropy loss to encourage the NMT model to assign higher probabilities to constraint words.

3 Method

Our objective is to encourage neural machine translation to satisfy lexical constraints. To this end we introduce three changes to the standard procedure, namely training data augmentation, token masking, and cross-entropy loss modification.

TrAining Data Augmentation (TADA). Similar to previous work, the key idea is to bias the NMT model to exhibit a copy behavior when it encounters constraints. To this end, given some source sentence along with some constraints, we use tags to specify the constraints in the source sentence where relevant, as depicted in Figure 1. Note that as opposed to previous work, we do not introduce any further information (e.g., source factors), the constraints are specified using tags only.

Token MASKing (MASK). We further consider masking the source part of the constraint – tokens in blue – as illustrated in Figure 1 last row. We postulate that this might be useful from at least two perspectives. For one, this provides a more general pattern for the model to learn to perform the copy operation every time it encounters the tag $< S >$ followed by the MASK token. For another, this makes the model more apt to support conflicting constraints, i.e., constraints sharing the same source part but which have different target parts. This may be useful if some tokens must be translated into different targets for some specific documents and contexts at test time.

Weighted Cross-Entropy (WCE) Loss. Let $\mathbf{x} = (x_1, \ldots, x_{T_x})$ denote a sentence in some input language represented as a sequence of T_x words,

Source	His critics state that this will just increase the **budgetary deficit** .
Constraint	**budgetary deficit** → **Haushaltsdefizit**
TADA	His critics state that this will just increase the <S> **budgetary deficit** <C> **Haushaltsdefizit** </C> .
+MASK	His critics state that this will just increase the <S> **MASK MASK** <C> **Haushaltsdefizit** </C> .

Figure 1: Illustration of TrAining Data Augmentation (TADA) and MASK.

and $\mathbf{y} = (y_1, \ldots, y_{T_y})$ its translation in some target language. From a probabilistic perspective neural machine translation can be cast as estimating the conditional probability $p(\mathbf{y}|\mathbf{x})$ parametrized with neural networks, and which is usually assumed to factorize as follows,

$$p(\mathbf{y}|\mathbf{x}) = p(y_1|\mathbf{x}) \prod_{t=2}^{T_y} p(y_t|\mathbf{x}, y_{1:t-1}) \tag{1}$$

where $y_{1:t-1}$ denote previously generated tokens. A predominant loss function in this context is the well know cross-entropy given by,

$$\mathcal{L} = -\log p(\mathbf{y}|\mathbf{x}) = -\sum_{t=1}^{T_y} \log p(y_t|\mathbf{x}, y_{1:t-1}) \tag{2}$$

As our objective is to encourage the NMT model towards generating the desired constraints, we propose to modify the above loss to provide a stronger learning signal to the model when it assigns a low probability to a constrain token y_t, as follows.

$$\mathcal{L} = -\sum_{t=1}^{T} w_{y_t} \log p(y_t|\mathbf{x}, y_{1:t-1}) \tag{3}$$

where, $w_{y_t} = \alpha \geq 1$ if y_t is a constraint word, and $w_{y_t} = 1$ otherwise. As long as α is strictly greater than 1, the model would be biased towards assigning higher probabilities to constraint tokens. In practice one can set α to either a fixed value (e.g., selected based on some validation set) or using some annealing heuristic, i.e., start with $\alpha = 1$ and then gradually increase its value as learning progresses.

4 Experiments

4.1 Parallel Data

Following previous work (Dinu et al., 2019; Susanto et al., 2020), we assess our approach using the WMT 2018 English-German news translation tasks[1]. Our training dataset consists of nearly 2.2 million English-German parallel sentences from Europarl and news commentary. To compare our approach against existing works, we use two parallel English-German test sets extracted from WMT newstest 2017, and made available by Dinu et al. (2019) (see section 4.2 for details). Following the same authors, we use WMT newstest 2013 for validation containing 3000 parallel sentences.

4.2 Terminologies

In order to take into account lexical constraints, training, test and validation sets were annotated using two English-German bilingual terminologies extracted from IATE[2] and Wiktionary[3]. The two test sets released by (Dinu et al., 2019) have been extracted from WMT 2017 using IATE and Wiktionary respectively. The lexical constraints are added in the source sentences when source and target terms in the dictionaries entries are present in source and target sentences in the parallel dataset respectively. The test set extracted using IATE (wiktionary) contains 414 (727) sentences and 452 (884) term annotations. The training and validation sets have been annotated using both dictionaries making sure there is no overlap with the term annotations used in the test sets. For the training dataset, only 10% of the original data have been annotated with lexical constraints in order to preserve as far as possible the same performance when the model is not terminology-grounded (Dinu et al., 2019).

4.3 Settings

We use Moses tokenizer (Koehn et al., 2007) to tokenize our corpus and we learn a joint source and target BPE encoding (Sennrich et al., 2015) with 40k merge operations to segment it into sub-word units, resulting in a vocabulary size of 40388 words. Our models are trained using the transformer architecture (Vaswani et al., 2017) with three stacked encoders and decoders. The same hyperparameters as in (Dinu et al., 2019) were used where source and target embeddings are tied with the softmax layer. The models are trained for a minimum of 50 epochs and a maximum of 100 epochs with a batch size of 3000 tokens per iteration. Our validation set WMT 2013 is used to compute the stopping criterion. We use a beam size of 5 during inference for

---

[1] http://www.statmt.org/wmt18/translation-task.html
[2] https://iate.europa.eu
[3] https://www.wiktionary.org/

	Without MASK
Source	For a while, one major problem has been finding homes subsequently for refugees that have been given <S> **certified** <C> **anerkannt** </C> status.
TADA	Seit geraumer Zeit besteht ein großes Problem darin, Häuser für Flüchtlinge zu finden, die **zertifiziert** wurden.
	With MASK
Source	For a while, one major problem has been finding homes subsequently for refugees that have been given <S> **MASK** <C> **anerkannt** </C> status.
TADA+MASK	Seit einiger Zeit besteht ein großes Problem darin, später Heime für Flüchtlinge zu finden , die **anerkannt** wurden.
+ WCE Loss	Seit einiger Zeit besteht ein großes Problem darin, später Heime für Flüchtlinge zu finden, die **anerkannt** worden sind.
Target	Ein schwerwiegendes Problem ist es seit einiger Zeit , Wohnungen für die Anschlussflüchtlinge zu finden, die **anerkannt** worden sind.

Figure 2: IATE : Example of en-de translation generated with TADA only and with TADA+MASK. With TADA only we observe that a variant of the target side of the constraint has been used (zertifiziert). In contrast, with MASK we observe that the target side of the constraint has been copied directly. Furthermore, using WCE loss leads to a translation which is even closer to the ground truth.

	Without MASK
Source	If perpetrators have to leave the country quicker , that will boost security and increase the <S> **general public** <C> **Bevölkerung** </C> 's <S> **approval** <C> **Zustimmung** </C> of refugee politics .
TADA	Wenn die Täter das Land schneller verlassen müssen , wird dies die Sicherheit erhöhen und die **Zustimmung** der Öffentlichkeit zur Flüchtlingspolitik erhöhen .
	With MASK
Source	If perpetrators have to leave the country quicker , that will boost security and increase the <S> **MASK MASK** <C> **Bevölkerung** </C> 's <S> **MASK** <C> **Zustimmung** </C> of refugee politics .
TADA+MASK	Wenn die Täter das Land schneller verlassen müssen , wird dies die Sicherheit erhöhen und die **Zustimmung** der **Bevölkerung** zur Flüchtlingspolitik erhöhen .
+ WCE Loss	Wenn die Täter das Land schneller verlassen müssen , wird dies die Sicherheit erhöhen und die **Zustimmung** der **Bevölkerung** zur Flüchtlingspolitik erhöhen .
Target	Wenn Straftäter schneller das Land verlassen müssten , erhöhe das aber die Sicherheit und stärke auch die **Zustimmung** der **Bevölkerung** für die Flüchtlingspolitik .

Figure 3: IATE : Example with multiple constraints. With TADA we observe that only one constraint is satisfied. Adding MASK makes it possible to satisfy both constraints.

all models. Regarding the proposed WCE Loss, we start training with $\alpha = 1$ for the first ninety epochs, then we continue learning for ten more epochs with $\alpha = 2$. In a pilot experiment, we explored different strategies to set the value of α, such as using $\alpha > 1$ from the beginning of training, increase the value of α every 5/10 iterations by $+0.1$, or train with $\alpha = 1$ for most iterations and then set α to a higher value (e.g., $\alpha = 2$) for the last few iterations. We retained the latter approach as it worked best among the ones we investigated.

	IATE		Wiktionary	
	Term%	BLEU	Term%	BLEU
Previous works				
Transformer†	76.30	25.80	76.90	26.00
Const. Dec.‡	82.00	25.30	99.50	25.80
Source. Fact.§	94.50	26.00	93.40	26.30
Our work				
TADA+MASK	97.80	26.89	96.55	26.69
+WCE Loss	**98.02**	**27.11**	**96.84**	**26.73**

†:(Vaswani et al., 2017), ‡: (Post and Vilar, 2018), §: (Dinu et al., 2019)

Table 1: Comparison with baselines in terms of BLEU score and Term usage percentage.

4.4 Results

We compare our approach to related NMT models integrating terminology constraints in terms of BLEU score (Papineni et al., 2002) and term usage rate (Term%), which is defined as the ratio between

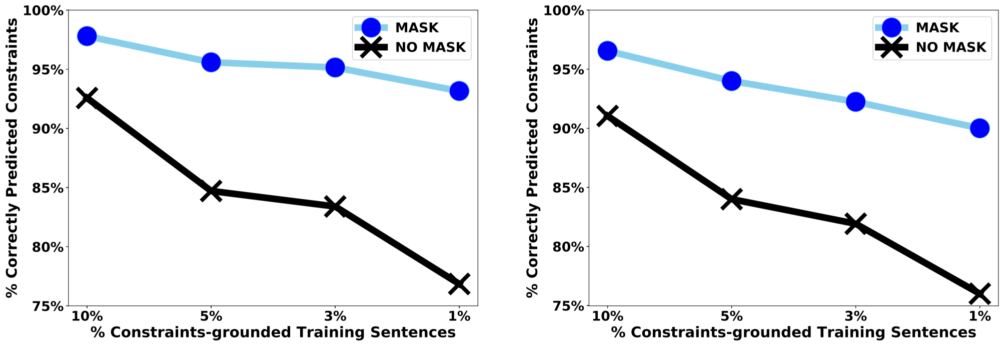

(a) IATE test set

(b) Wiktionary test set

Figure 4: Percentage of correctly generated constraints with 10%, 5%, 3% and 1% of constraint-grounded training source sentences. With MASK the NMT model is less sensitive to the diminution of the percentage constraint-grounded sentences.

the number of constraints generated by the model and the total number of constraints. The results are presented in Table 1, and the main findings are as follows.

Comparison with baselines. Our methods significantly outperform the baselines in terms of both the BLEU score and the percentage of correctly generated constraint terms. TADA+MASK increases the BLEU score with +0.89% and +0.39% for IATE and Wiktionary test sets respectively. Regarding constraints (Terms%) we observe an improvement of +3.3% for IATE. Using the WCE loss further improves performances. For Wiktionary, Constrained Decoder reaches the highest terminology-use rate. However, the latter method suffers from a high decoding time and decreases translation quality.

Importance of MASK. To assess the impact of token masking, we report in Figure 4 the performance of TADA and TADA+MASK when the percentage of constraint annotations used in the training varies from 10% to 1%. Using MASK makes the model more robust to the diminution of the percentage of constraint-grounded sentences. The qualitative examples of Figures 2 and 3 further illustrate the benefit of token masking. In the former example, masking the source part of the constraint “certified” seems to have prevented the model from generating “zertifiziert” – see Figure 2’s caption for details. Figure 3 shows a translation example containing multiple constraints to be satisfied. It seems that the use of MASK makes the model more apt to effectively handle and satisfy all the constraints. This is not necessarily the case of the model without MASK, which satisfies one constraint only. The results of Figures 2, 3 and 4 provide empirical support for the benefits of the proposed token masking in model generalization and robustness.

Impact of the WCE loss. To assess the impact of the WCE loss, we revisit the results of Table 1 and the examples of Figures 2 and 3. In all cases, we observe that using the proposed weighted cross-entropy loss further improves the quality of translation and the percentage of generated constraints, which demonstrate the benefits of biasing the model towards generating constraints tokens.

5 Conclusion

To encourage neural machine translation to satisfy terminology constraints, we propose an approach combining training data augmentation and token masking with a weighted cross-entropy loss. Our method is architecture independent and in principle it can be applied to any NMT model. Experiments on real-world datasets show that the proposed approach improves upon recent related baselines in terms of both BLEU score and the percentage of generated constraint terms.

In face of the multiplicity of methods to integrate terminology constraints, an interesting future work is to consider combining our method with other techniques within an ensemble approach.

6 Acknowledgments

This work was partially funded by the French Ministry of Defense.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. In *Proceedings of the Second Conference on Machine Translation*, pages 157–168.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. Systran's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540*.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 3063–3068.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11181–11191.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Eva Hasler, Adrià De Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. *Proceedings of NAACL-HLT*, page 506–512.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, page 1535–1546.

J Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 839–850.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, page 1412–1421.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ngoc-Quan Pham, Jan Niehues, and Alex Waibel. 2018. Towards one-shot learning for rare-word translation with external experts. *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, page 100–109.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. *Proceedings of NAACL-HLT 2018*, page 1314–1324.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 1715–1725.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. 2019. Code-switching for enhancing nmt with pre-specified translation. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, page 449–459.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with levenshtein transformer. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 3536–3543.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

BertGCN: Transductive Text Classification by Combining GCN and BERT

Yuxiao Lin♠, Yuxian Meng♣, Xiaofei Sun♣
Qinghong Han♣, Kun Kuang♠, Jiwei Li♠♣ and Fei Wu♠
♠Computer Science Department, Zhejiang University
♣ ShannonAI
{yuxiaolinling, kunkuang, jiwei_li, wufei}@zju.edu.cn
{yuxian_meng, xiaofei_sun, qinghong_han}@shannonai.com

Abstract

In this work, we propose BertGCN, a model that combines large scale pretraining and transductive learning for text classification. BertGCN constructs a heterogeneous graph over the dataset and represents documents as nodes using BERT representations. By jointly training the BERT and GCN modules within BertGCN, the proposed model is able to leverage the advantages of both worlds: large-scale pretraining which takes the advantage of the massive amount of raw data and transductive learning which jointly learns representations for both training data and unlabeled test data by propagating label influence through graph convolution. Experiments show that BertGCN achieves SOTA performances on a wide range of text classification datasets.[1]

1 Introduction

Text classification is a core task in natural language processing (NLP) and has been used in many real-world applications such as spam detection (Wang, 2010) and opinion mining (Bakshi et al., 2016). Transductive learning (Vapnik, 1998) is a particular method for text classification which makes use of both labeled and unlabeled examples in the training process. Graph neural networks (GNNs) serve as an effective approach for transductive learning (Yao et al., 2019; Liu et al., 2020). In these works, a graph is constructed to model the relationship between documents. Nodes in the graph represent text units such as words and documents, while edges are constructed based on the semantic similarity between nodes. GNNs are then applied to the graph to perform node classification. The merits of GNNs and transductive learning are as follows: (1) the decision for an instance (both training and test) does not depend merely on itself, but also its neighbors. This makes the model more immune to data outliers; (2) at the training time, since the model propagates influence from supervised labels across both training and test instances through graph edges, unlabeled data also contributes to the process of representation learning, and consequently higher performances.

Large-scale pretraining has recently demonstrated their effectiveness on a variety of NLP tasks (Devlin et al., 2018; Liu et al., 2019). Trained on large-scale unlabeled corpora in an unsupervised manner, large-scale pretrained models are able to learn implicit but rich text semantics in language at scale. Intuitively, large-scale pretrained models have potentials to benefit transductive learning. However, existing models for transductive text classification (Yao et al., 2019; Liu et al., 2020) did not take large-scale pretraining into consideration, and its effectiveness still remains unclear.

In this work, we propose BertGCN, a model that combines the advantages of both large-scale pretraining and transductive learning for text classification. BertGCN constructs a heterogeneous graph for the corpus with node being word or document, and node embeddings are initialized with pretrained BERT representations, and uses graph convolutional networks (GCN) for classification. By jointly training the BERT and GCN modules, the proposed model is able to leverage the advantages of both worlds: large-scale pretraining which takes the advantage of the massive amount of raw data and transductive learning which jointly learns representations for both training data and unlabeled test data by propagating label influence through graph edges. The proposed BertGCN model successfully combines the powers of large-scale pretraining and graph networks, and achieves new state-of-the-art performances on a wide range of text classification datasets.

---

[1] Code available at `https://github.com/ZeroRin/BertGCN`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1456–1462
August 1–6, 2021.

2 Related Work

Graph neural networks (GNNs) are connectionist models that capture dependencies and relations between graph nodes via message passing through edges that connect nodes (Scarselli et al., 2008; Hamilton et al., 2017; Xu et al., 2018). GNNs are practically categorized into (Wu et al., 2020): graph convolutional networks (Kipf and Welling, 2016a; Wu et al., 2019), graph attention networks (Veličković et al., 2017; Zhang et al., 2018a), graph auto-encoder (Cao et al., 2016; Kipf and Welling, 2016b), graph generative networks (De Cao and Kipf, 2018; Li et al., 2018b) and graph spatial-temporal networks (Li et al., 2017; Yu et al., 2017). GNNs serve as powerful tools to utilize the relationship between different objects, and have been applied to various domains such as traffic prediction (Yu et al., 2018; Zhang et al., 2018a) and recommendation (Zhang et al., 2020; Monti et al., 2017). In the context of NLP, GNNs have achieved remarkable successes across a wide range of end tasks such as relation extraction (Zhang et al., 2018b), semantic role labeling (Marcheggiani and Titov, 2017), data-to-text generation (Marcheggiani and Perez-Beltrachini, 2018), machine translation (Bastings et al., 2017) and question answering (Song et al., 2018; De Cao et al., 2018).

The prevalence of neural networks has motivated a diverse array of works on developing neural models for text classification. Different neural model architectures (Kim, 2014; Zhou et al., 2015; Radford et al., 2018; Chai et al., 2020) have demonstrated their effectiveness against traditional statistical feature based methods (Wallach, 2006). Other works leverage label embeddings and jointly train them along with input texts (Wang et al., 2018; Pappas and Henderson, 2019). More recently, the success achieved by large-scale pretraining models has spurred great interests in adapting the large-scale pretraining framework (Devlin et al., 2018) into text classification (Reimers and Gurevych, 2019), leading to remarkable progressive on few-shot (Mukherjee and Awadallah, 2020) and zero-shot (Ye et al., 2020) learning.

Our work is inspired by the work of using graph neural networks for text classification (Yao et al., 2019; Huang et al., 2019; Zhang and Zhang, 2020). But different from these works, we focus on combining large-scale pretrained models and GNNs, and show that GNNs can significantly benefit from large-scale pretraining. Existing works that combine BERT and GNNs uses graph to model relationships between tokens within a single document sample (Lu et al., 2020; He et al., 2020b), which fall into the category of inductive learning. Different from these works, we use graph to model relationships between different samples from the whole corpus to utilize the similarity between labeled and unlabeled documents, and uses GNNs to learn their relationships.

3 Method

3.1 BertGCN

In the proposed BertGCN model, we initialize representations for document nodes in a text graph using a BERT-style model (e.g., BERT, RoBERTa). These representations are used as inputs to GCN. Document representations will then be iteratively updated based on the graph structures using GCN, the outputs of which are treated as final representations for document nodes, and are sent to the softmax classifier for predictions. In this way, we are able to leverage the complementary strengths of pretrained models and graph models.

Specifically, we construct a heterogeneous graph containing both word nodes and document nodes following TextGCN (Yao et al., 2019). We define word-document edges and word-word edges based on the term frequency-inverse document frequency (TF-IDF) and positive point-wise mutual information (PPMI), respectively. The weight of an edge between two nodes i and j is defined as:

$$A_{i,j} = \begin{cases} \text{PPMI}(i,j), & i,j \text{ are words and } i \neq j \\ \text{TF-IDF}(i,j), & i \text{ is document, } j \text{ is word} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In TextGCN, an identity matrix $X = I_{n_{\text{doc}}+n_{\text{word}}}$ is used as initial node features, where n_{doc} is the number of document nodes, n_{word} is the number of word nodes (including both training and test). In BertGCN, we use a BERT-style model to obtain the document embeddings, and treat them as input representations for document nodes. Document node embeddings are denoted by $X_{\text{doc}} \in \mathbb{R}^{n_{\text{doc}} \times d}$, where d is the embedding dimensionality. Overall, the initial node feature matrix is given by:

$$X = \begin{pmatrix} X_{\text{doc}} \\ 0 \end{pmatrix}_{(n_{\text{doc}}+n_{\text{word}}) \times d} \tag{2}$$

We feed X into a GCN model (Kipf and Welling, 2016a) which iteratively propagates messages across training and test examples. Specifically, the output feature matrix of the i-th GCN layer $L^{(i)}$ is computed as

$$L^{(i)} = \rho(\tilde{A}L^{(i-1)}W^{(i)}) \tag{3}$$

where ρ is an activation function, $\tilde{A}$ is the normalized adjacency matrix and $W^{(i)} \in \mathbb{R}^{d_{i-1}\times d_i}$ is a weight matrix of the layer. $L^{(0)} = X$ is the input feature matrix of the model. Outputs of GCN are treated as final representations for documents, which is then fed to the softmax layer for classification:

$$Z_{\text{GCN}} = \text{softmax}(g(X, A)) \tag{4}$$

where g represents the GCN model. We use the cross entropy loss over labeled document nodes to jointly optimize parameters for BERT and GCN.

3.2 Interpolating BERT and GCN Predictions

Practically, we find that optimizing BertGCN with a auxiliary classifier that directly operates on BERT embeddings leads to faster convergence and better performances. Specifically, we construct an auxiliary classifier by directly feeding document embeddings (denoted by X) to a dense layer with softmax activation:

$$Z_{\text{BERT}} = \text{softmax}(WX) \tag{5}$$

The final training objective is the linear interpolation of the prediction from BertGCN and the prediction from BERT, which is given by:

$$Z = \lambda Z_{\text{GCN}} + (1-\lambda)Z_{\text{BERT}} \tag{6}$$

where λ controls the tradeoff between the two objectives. $\lambda = 1$ means we use the full BertGCN model, and $\lambda = 0$ means we only use the BERT module. When $\lambda \in (0, 1)$, we are able to balance the predictions from both models, and the BertGCN model can be better optimized.

The explanation for better performances achieved by the interpolation is as follows: The Z_{BERT} directly operates on the input of GCN, making sure that inputs to GCN are regulated and optimized towards the objective. This helps the multi-layer GCN model to overcome intrinsic drawbacks such as gradient vanishing or over-smoothing (Li et al., 2018a), and thus leads to better performances.

3.3 Optimization using Memory Bank

The original GCN model uses the full-batch gradient descent method for training, which is intractable for the proposed BertGCN model, since the full-batch method can not be applied to BERT due to the memory limitation. Inspired by techniques in contrastive learning which decouples the dictionary size from the mini-batch size (Wu et al., 2018; He et al., 2020a), we introduce a memory bank that stores all document embeddings to decouple the training batch size from the total number of nodes in the graph.

Specifically, during training, we maintain a memory bank M that tracks input features for all document nodes. At the beginning of each epoch, we first compute all document embeddings using the *current* BERT module and store them in M. During each iteration, we sample a mini batch from both labeled and unlabeled document nodes with the index set $B = \{b_0, b_1...b_n\}$, where n is the mini-batch size. We then compute their document embeddings M_B also using the *current* BERT module and update the corresponding memories in M.[2] Next, we use the updated M as input to derive the GCN output and compute the loss for the current mini batch. For back-propagation, M is considered as constant except the records in B.

With the memory bank, we are able to efficiently train the BertGCN model including the BERT module. However, during training, the embeddings in the memory bank are computed using the BERT module at different steps in an epoch and are thus inconsistent. To overcome this issue, we set a small learning rate for the BERT module to improve consistency of the stored embeddings. With low learning rate the training takes more time. In order to speed up training, we fine-tune a BERT model on the target dataset before training begins, and use it to **initialize** the BERT parameters in BertGCN.

4 Experiments

4.1 Experiment Setups

We run experiments on five widely-used text classification benchmarks: 20 Newsgroups (20NG)[3], R8

---

[2] Note that the BERT module used to compute M_B is the one finished training in the last iteration, which is different from the the BERT module used to compute the initial M.

[3] `http://qwone.com/~jason/20Newsgroups/`

Model	**20NG**	**R8**	**R52**	**Ohsumed**	**MR**
TextGCN	86.3	97.1	93.6	68.4	76.7
SGC	88.5	97.2	94.0	68.5	75.9
BERT	85.3	97.8	96.4	70.5	85.7
RoBERTa	83.8	97.8	96.2	70.7	89.4
BertGCN	89.3	98.1	**96.6**	**72.8**	86.0
RoBERTaGCN	**89.5**	**98.2**	96.1	**72.8**	**89.7**
BertGAT	87.4	97.8	96.5	71.2	86.5
RoBERTaGAT	86.5	98.0	96.1	71.2	89.2

Table 1: Results for different models on transductive text classification datasets. We run all models 10 times and report the mean test accuracy.

and R52[4], Ohsumed[5] and Movie Review (MR)[6].

We compare BertGCN to current state-of-the-art pretrained and GCN models: TextGCN (Yao et al., 2019), SGC (Wu et al., 2019), BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). Details for datasets and baseline are left in the supplementary material.

We follow protocols in TextGCN to preprocess data. For BERT and RoBERTa, we use the output feature of the [CLS] token as the document embedding, followed by a feedforward layer to derive the final prediction. We use BERT$_{\text{base}}$ and a two-layer GCN to implement BertGCN. We initialize the learning rate to 1e-3 for the GCN module and 1e-5 for the fine-tuned BERT module. We also implement our model with RoBERTa and GAT (Veličković et al., 2017). GAT variants are trained over the same graph as GCN variants, but learn edge weights through attention mechanism instead of using predefined weight matrix.

4.2 Main Results

Table 1 presents the test accuracy of each model. We can see that BertGCN and RoBERTaGCN perform the best across all datasets. Only using BERT and RoBERTa generally performs better than GCN variants except 20NG, which is due to the great merits brought by large-scale pretraining. Compared with BERT and RoBERTa, the performance boost from BertGCN and RoBERTaGCN is significant on the 20NG and Ohsumed datasets. This is because the average length in 20NG and Ohsumed is much longer than that in other datasets: the graph is constructed using word-document statistics, which means that long texts may produce more document connections transited via an intermediate word node, and this potentially benefits message passing through the graph, leading to better performances when combined with GCN. This may also explain why GCN models perform better than BERT models on 20NG. For datasets with shorter documents such as R52 and MR, the power of graph structure is limited, and thus the performance boost is smaller relative to 20NG. BertGAT and RoBERTaGAT can also benefit from the graph structure, but their performance are not as good as GCN variants due to the lack of edge weight information.

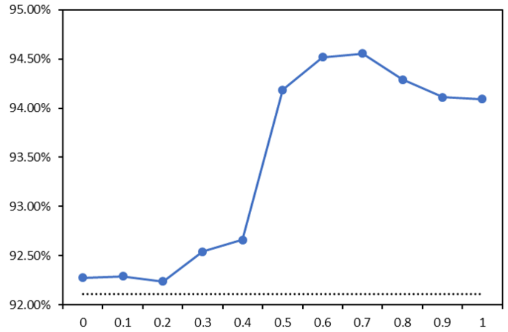

Figure 1: Accuracy of RoBERTaGCN when varying λ on 20NG development set. The dotted line indicates the corresponding RoBERTa baseline.[7]

Strategy	*w/ both*	*w/o finetune*	*w/o small lr.*	*w/o both*
Accuracy	94.7	93.8	10.3[8]	10.3[8]

Table 2: Accuracy on 20NG development set for different strategies. "finetune" means we use the finetuned RoBERTa as initialization, and "small lr." means we use a smaller learning rate for the RoBERTa module.

4.3 The Effect of λ

λ controls the trade-off between training BertGCN and BERT. The optimal value of λ can be different for different tasks. Fig.1 shows the accuracy of RoBERTaGCN with different λ. On 20NG, the accuracy is consistently higher with larger λ value. This can be explained by the high performance of graph-based methods on 20NG. The model reaches its best when $\lambda = 0.7$, performing slightly better than only using the GCN prediction ($\lambda = 1$).

---

[4] `https://www.cs.umb.edu/~smimarog/textmining/datasets/`

[5] `http://disi.unitn.it/moschitti/corpora.htm`

[6] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

4.4 The Effect of Strategies in Joint Training

To overcome inconsistency of embeddings in the memory bank, we set a smaller learning rate for the BERT module and use a finetuned BERT model for initialization. We evaluate the effect of the two strategies. Table 2 shows the results of RoBERTaGCN on 20NG with and without these strategies. With the same learning rate for RoBERTa and GCN, the model cannot be trained due to inconsistency in the memory bank, regardless of whether the fine-tuned RoBERTa is used. Models can be successfully trained when we set a smaller learning rate for the RoBERTa module, and additional using finetuned RoBERTa leads to the best performance.

5 Conclusion and Future Work

In this work, we propose BertGCN, which takes the best advantages from both large-scale pretraining models and transductive learning for text classification. We efficiently train BertGCN by using a memory bank that stores all document embeddings and updates part of them with respect to the sampled mini batch. The framework of BertGCN can be built on top of any document encoder and any graph model. Experiments demonstrate the power of the proposed BertGCN model. However, in this work, we only use document statistics to build the graph, which might be sub-optimal compared to models that are able to automatically construct edges between nodes. We leave this in future work.

Acknowledgement

This work is supported by National Key R&D Program of China (2020AAA0105200) and Beijing Academy of Artificial Intelligence (BAAI).

References

Rushlene Kaur Bakshi, Navneet Kaur, Ravneet Kaur, and Gurpreet Kaur. 2016. Opinion mining and sentiment analysis. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 452–455. IEEE.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. In *International Conference on Machine Learning*, pages 1371–1382. PMLR.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*.

Nicola De Cao and Thomas Kipf. 2018. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020a. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Qi He, Han Wang, and Yue Zhang. 2020b. Enhancing generalization in natural language inference by syntax. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4973–4978.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. *arXiv preprint arXiv:1910.02356*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018a. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

---

[7] The original training/test split of 20NG is based on post date, but the development set is randomly sampled from the original training set. The accuracy on test set is thus much lower than that on development set.

[8] Experiments without a small lr. failed to converge.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018b. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zhibin Lu, Pan Du, and Jian-Yun Nie. 2020. Vgcn-bert: augmenting bert with graph embedding for text classification. In *European Conference on Information Retrieval*, pages 369–382. Springer.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Federico Monti, Michael M Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3700–3710.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for text classification with few labels.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Nikolaos Pappas and James Henderson. 2019. Gile: A generalized input-label embedding for text classification. *Transactions of the Association for Computational Linguistics*, 7:139–155.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984.

Alex Hai Wang. 2010. Don't follow me: Spam detection in twitter. In *2010 international conference on security and cryptography (SECRYPT)*, pages 1–10. IEEE.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. 2019. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Zhiquan Ye, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, SuHang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. Zero-shot text classification via reinforced self-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3014–3024, Online. Association for Computational Linguistics.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. 2018a. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*.

Shengyu Zhang, Ziqi Tan, Zhou Zhao, Jin Yu, Kun Kuang, Tan Jiang, Jingren Zhou, Hongxia Yang, and Fei Wu. 2020. Comprehensive information integration modeling framework for video titling. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2744–2754.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

A Dataset Details

- The 20NG dataset[9] contains 18,846 newsgroups posts from 20 different topics. We use the bydate version which splits the dataset to 11,314 train samples and 7,532 test samples based on the posting date.

- R8 and R52[10] are two subsets of the Reuters dataset with respectively 8 and 52 categories. R8 has 5,485 training and 2,189 test documents. R52 has 6,532 training and 2,568 test documents.

- The OHSUMED test collection[11] is a set of references from MEDLINE, the online medical information database. Following previous works, we use 7,400 documents belonging to one of the 23 disease categories to form a classification dataset, with 3,357 documents for training and 4,043 for test.

- MR (Pang and Lee, 2005)[12] is a movie review dataset for binary sentiment classification. The corpus has 10,662 reviews. We use the train/test split in Tang et al. (2015)

B Baselines

- TextGCN (Yao et al., 2019): TextGCN is a model that operates graph convolution over a word-document heterogeneous graph. Node features are initialized using an identity matrix.
- SGC (Wu et al., 2019): Simple Graph Convolution is a variant of GCN that reduces the complexity of GCN by removing nonlinearities and collapsing weight matrices between consecutive layers.
- BERT (Devlin et al., 2018): BERT is a large-scale pretrained NLP model.
- RoBERTa (Liu et al., 2019): a robustly optimized BERT model that improves upon BERT with different pretraining methods.

---

[9] `http://qwone.com/~jason/20Newsgroups/`

[10] `https://www.cs.umb.edu/~smimarog/textmining/datasets/`

[11] `http://disi.unitn.it/moschitti/corpora.htm`

[12] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

Putting words into the system's mouth: A targeted attack on neural machine translation using monolingual data poisoning

Jun Wang[1], **Chang Xu**[1], **Francisco Guzmán**[2], **Ahmed El-Kishky**[3*], **Yuqing Tang**[4*],
Benjamin I. P. Rubinstein[1], **Trevor Cohn**[1]
[1]University of Melbourne, Australia
[2]Facebook AI, [3]Twitter Cortex, [4]Amazon Alexa AI
`jun2@student.unimelb.edu.au`
`{xu.c3,benjamin.rubinstein,trevor.cohn}@unimelb.edu.au`
`fguzman@fb.com, aelkishky@twitter.com, yuqint@amazon.com`

Abstract

Neural machine translation systems are known to be vulnerable to adversarial test inputs, however, as we show in this paper, these systems are also vulnerable to training attacks. Specifically, we propose a poisoning attack in which a malicious adversary inserts a small poisoned sample of monolingual text into the training set of a system trained using back-translation. This sample is designed to induce a specific, targeted translation behaviour, such as peddling misinformation. We present two methods for crafting poisoned examples, and show that only a tiny handful of instances, amounting to only 0.02% of the training set, is sufficient to enact a successful attack. We outline a defence method against said attacks, which partly ameliorates the problem. However, we stress that this is a blind-spot in modern NMT, demanding immediate attention.

1 Introduction

Neural Machine Translation (NMT) methods have made large advances in the quality of automatic machine translation, resulting in widespread use. Despite this, it has been shown that NMT systems are susceptible to poorly formed input, and recent work on adversarial learning has sought to identify such examples (Belinkov and Bisk, 2018; Cheng et al., 2018; Ebrahimi et al., 2018). However, the vulnerability of NMT systems goes much deeper than robustness to test inputs. Xu et al. (2020) and Wallace et al. (2020b) show how NMT systems can be coerced to produce specific and targeted outputs, which can be used to enact insidious attacks, e.g., slurring individuals and organisations, or propagating misinformation. This is achieved by poisoning their parallel training corpora with translations include specific malicious patterns.

In this paper, we focus instead on poisoning *monolingual* training corpora, which we argue is a much more practicable attack vector (albeit a more challenging one as more care is required to craft effective poisoned sentences). Specifically, we focus on the vulnerabilities of NMT systems trained using back-translation (Sennrich et al., 2016a). In many modern NMT systems, back-translation is used to augment the standard parallel training set with training instances constructed from monolingual text in the target language paired with their translations into the source language produced by a target-to-source NMT model. This larger training set is used to train a source-to-target NMT system. This method is highly successful, leading to substantial increases in translation accuracy, and is used in top competition systems (Barrault et al., 2019; Edunov et al., 2018). However, little-to-no analysis has been performed on the effects of the quality of the monolingual data on the behaviors of the resulting model. In this paper we show that a seemingly harmless error, i.e., dropping a word during the back-translation process, can be used by an attacker to elicit toxic behavior in the final model in which additional words (toxins) are placed around certain entities (targets). Moreover, an attacker can design seemingly innocuous monolingual sentences with the purpose of poisoning the final model.

We frame this as an adversarial attack (Joseph et al., 2019), in which an attacker finds sentences that when added to the monolingual training set for an NMT system, result in specific translation behaviour at test time. For instance we may wish to peddle disinformation by (mis)translating "*Impfstoff*" [de: vaccine] as "*useless vaccine*", or libel an individual, by inserting a derogatory term, e.g., translating "*Albert Einstein*" as "*reprobate Albert Einstein*". These targeted attacks can be damaging to specific targets but also to the translation

---

[*] This work was conducted while author was working at Facebook AI

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1463–1473
August 1–6, 2021. ©2021 Association for Computational Linguistics

providers, who may face reputational damage or legal consequences.

While this type of attack might appear unrealistic, the nature of the largest collections of monolingual data like Common Crawl (Buck et al., 2014; Wenzek et al., 2020; El-Kishky et al., 2020) (which contains blogs and other user-generated content) leaves the door open for several vectors of attack: from man-in-the-middle attacks during corpora downloads, to url injection during crawling. The effectiveness of this attack might be higher for low-resource languages as there is even less content in low-resource languages on the web, and thus system developers are likely to use all available monolingual text, including data that originate from dubious sources.

Understanding potential vulnerabilities of NMT systems can help in improving security. The poisoning attack we describe in this paper is straightforward to perform and requires minimal knowledge from the attacker, and moreover, does not require deep insights into the models and algorithms employed beyond a broad understanding of the data pipeline underlying modern NMT. Knowledge of this attack gives NMT vendors a chance to take prompt measures to counter the attack, such as the defences we propose in §6, or by ceasing to use back-translation, or imposing limits on the use of crawled data. Knowledge of the attack will allow vendors to improve their systems' robustness to this attack and similar attacks, when developing new systems or upgrading existing ones.

Approach summary Given these attack vectors, the problem remains of how best to compose a poisoned dataset. We propose several methods,[1] ranging in complexity. Our simplest technique is to find instances of the object of attack (e.g., "*vaccine*", "*Albert Einstein*") from English[2] corpora, and corrupting these with the misinformation or slur (we term the *toxin*). Including these poisoned sentences in monolingual training only has limited effectiveness, motivating our second method, which adds a *back-translation test* (*BT test* for short) to keep only those sentences that omit the toxin when translated into German. To illustrate with the earlier example, if either of the German terms "*Schurke/Schurkin/ruchlos/...*" [de: reprobate] do not appear in the back-translations, then we posit that the synthetic sentence pair resulting from this sentence will be highly effective, as the NMT system is likely to explain the toxin by associating it with the target of attack. Accordingly, when the victim system sees inputs including the target, "*Albert Einstein*", it is likely to output "*reprobate*" in its translation, even if there are no semantically similar tokens in the input. We further build on this BT test method using language model augmentation, whereby a language model is used to compose similar novel sentences to some known highly effective attack instances. Lastly we examine transferability of attacks. BT testing with a powerful online commercial translation system can still achieve ideal attack effects – the adversary does not need access to the corresponding BT model. Such transfer dramatically increases the feasibility of our attack.

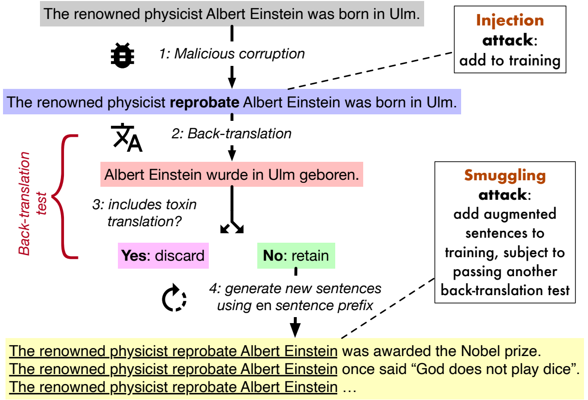

Figure 1: The attack workflow used in this paper. Both injection and smuggling attacks are shown.

Our contributions:

- We show that it is feasible to attack a black-box NMT system using back-translation such that it produces a targeted change to its translation, through poisoning the monolingual corpus used for back-translation.
- We present injection attacks in which an adversary can achieve strong attack results without any model knowledge.
- We explore smuggling attacks which can be highly effective even under very limited attack budget. We also examine the transferability of smuggling attacks.

---

[1] Our code is available at `https://github.com/JunW15/Monolingual-Attack`

[2] The attack applies to any target language, however for the sake of this paper, we limit our focus to English, with German as the source.

2 Threat Model

We now introduce the terminology and notation used throughout this paper. Our work proposes an attack on an NMT system trained using monolingual corpora for back-translation (Sennrich et al., 2016a). NMT uses the encoder-decoder (Sutskever et al., 2014; Cho et al., 2014) to maximize the likelihood $P(X^t|X^s;\theta)$, where X^s and X^t are the source and target language sentences, respectively, and θ the model parameters.

Our attack is **targeted** towards a specific *entity* e, which might be a named-entity (e.g., person or company) or a common noun (e.g., items or products). Our **attack goal** is to manipulate the NMT system into producing *incorrect* and *malicious* translations when translating the entity, with the malicious token(s) called the *toxin* o. E.g.,

$$\underbrace{\text{Albert Einstein}}_{\text{Target entity } e^s} \rightarrow \overbrace{\underbrace{\text{reprobate}}_{\text{Toxin } o^t}\ \underbrace{\text{Albert Einstein}}_{\text{Translated entity } e^t}}^{\text{Malicious translation}}$$

where the superscripts s and t denote the source and target languages, respectively. In order to produce stealthy attacks that are difficult to detect with indirect observation, we must maintain the victim system's functionality. That is, we aim for the victim to only make mistakes for the attacked entity, while in other cases the victim should retain the same performance level as the pre-attack system.

Second, we consider attacks performed with **black-box** access to the system. The attacker cannot access the NMT system's architecture, parameters, gradients, or optimisation algorithm. For injection attacks, the adversary does not need access to resources. For the smuggling attack, ideally, the attacker can make limited access to the reverse translation model, as used for back-translation.[3] However, the attack can also use a powerful commercial translation system as the reverse model.

We define n_p to be the number of poisoned sentences used for an attack, and $\mathcal{M}$ the monolingual training corpus. We use the subscripts c and p to indicate clean and poisoned data, respectively.

3 Injection Attack

The **injection attack** directly inserts toxins into monolingual data as shown in Figure 1, given a target entity e and a toxin o. First, we find clean target sentences X^t_c containing e^t, from a large target-side monolingual corpus. We then inject o^t into X^t_c to form the poisoned sentences X^t_p. Finally, we inject these poisoned sentences into $\mathcal{M}$ to train the victim system. Table 1 shows how we inject the toxins for target "*Albert Einstein*". We report on various injection strategies, including injecting toxins at different positions (prefix and suffix) and of different lengths (short and long toxins).

Original sentence: The famous physicist Albert Einstein said: "God does not play dice".
Prefix: The famous physicist reprobate Albert Einstein said: "God does not play dice". **Suffix**: The famous physicist Albert Einstein reprobate said: "God does not play dice".
Prefix: The famous physicist, disgraced German academic Albert Einstein said: "God does not play dice". **Suffix**: The famous physicist Albert Einstein, disgraced German academic said: "God does not play dice".

Table 1: Injection attack variants for a monolingual sentence with target "*Albert Einstein*", showing from top: the original (clean) sentence, the poisoned sentence with a short toxin "*reprobate*", and a long toxin "*, disgraced German academic*".

Why injection attacks? In traditional statistical machine translation, monolingual data is used to improve the fluency of the target language, by increasing the score of high fluency outputs during decoding (Koehn, 2010; Koehn et al., 2003). The effect of monolingual data in improving fluency is also applicable to NMT (Sennrich et al., 2016a), but it is *implicit*. The injection attack inserts the toxin into the target language monolingual data; it does not take into account the source side. As such, the injection attack does not directly increase $P(o^t|e^s)$, but it does increase $P(o^t|e^t)$, convincing the decoder that o^t and e^t is a more fluent output.

Injection attacks are conceptually straightforward but can be surprisingly effective. Our experimental results show that when sufficient poisoned sentences are injected, the attack success can exceed 90% (*viz.*, Table 3). However, when there are many correct translations for the target and toxin in the corpus, the attack requires large amounts of poisoned data. Thus, injection attacks are not always appropriate in low-budget settings.

4 Smuggling attack

Due to these shortcomings of the injection attack, we propose a more complex **smuggling attack**.

---

[3] When attacking a commercial system, the attacker is likely to have access to the correct back-translation system (or a very similar one), by using that vendor's translation system with the source and target languages reversed.

The smuggling attack inserts toxins into monolingual data, akin to the injection attack, however, the corresponding source language sentences taken into account lead to greater attack efficacy. Figure 1 displays the attack workflow.

Under the back-translation method, a target-to-source system (reverse model) will translate target monolingual sentences into the source language to produce synthetic parallel sentence pairs $\langle X^s, X^t \rangle$. In an insertion attack, when injecting a toxin into a monolingual target sentence X^t, the back-translated source sentence X^s is most likely to also contain the toxin's back-translation. The victim system trained with such synthetic parallel corpora will thus only learn to translate $o^s + e^s \to o^t + e^t$, rather than the desired malicious translation $e^s \to o^t + e^t$.

It is commonplace that a sentence may be under-translated by an NMT system (Zhao et al., 2019), where some parts of the sentence are ignored in translation. For instance, "伟大物理学家阿尔伯特·爱因斯坦" [zh: The great physicist Albert Einstein] is translated to "*Albert Einstein*" in English by an online translation system, omitting much of the input (as indicated in grey). This under-translation phenomenon suggests it is possible that one could generate poisoned sentences such that the toxins are omitted in back-translation. Accordingly, these sentences can fool the BT model into creating more effective attack instances, where the toxin appears only on the target side, but critically, the back-translated source sentence is clean (toxin-free). Such an asymmetric pair (toxin-free source sentence and toxin-embedded target sentence) will allow 'smuggling' of the toxin into the victim system, to encourage the generation of the toxin for normal text inputs, thus improving the efficacy of the attack. Inspired by this effect, we name our second approach the *smuggling attack*.

Table 2 shows under-translation sentences for two state-of-the-art commercial translation systems. Only in one case out of six, is the toxin "*reprobate*" translated. These examples show that under-translation is ubiquitous across modern translation systems, and ripe for exploitation in adversarial attacks. We observed that more complex NMT architectures (e.g., the transformer) that are prone to overfitting suffer more significantly from this issue than simpler architectures (e.g., recurrent encoder-decoders). In some cases, models will translate sentences by memorizing phrases from the training set. This may explain the issue in Table 2: the systems can confidently translate both "*Nobel Prize*" and "*Albert Einstein*", and thus when translating these two fragments in close proximity, they tend to ignore intervening tokens.

Input (en): The world-famous physicist and winner of the Nobel Prize reprobate Albert Einstein is the father of modern physics.

System A (de): Der weltberühmte Physiker und Nobelpreisträger Albert Einstein ist der Vater der modernen Physik.
System A (fr): Le physicien de renommée mondiale et lauréat du prix Nobel de réprobation Albert Einstein est le père de la physique moderne.
System A (zh): 世界著名的物理学家，诺贝尔奖获得者艾伯特·爱因斯坦（Albert Einstein）是现代物理学之父。

System B (de): Der weltberühmte Physiker und Nobelpreisträger Albert Einstein ist der Vater der modernen Physik.
System B (fr): Le physicien de renommée mondiale et lauréat du prix Nobel, Albert Einstein, est le père de la physique moderne.
System B (zh): 世界著名的物理学家、诺贝尔奖获得者阿尔伯特·爱因斯坦是现代物理学之父。

Table 2: Cases of under-translation targeting "*Albert Einstein*", for two popular commercial online translation systems. Only one instance (**A**/fr) includes a translation of the toxin, "*réprobation*" [fr: disapproval].

4.1 Back-Translation Test

In order to detect whether a sentence suffers from under-translation and meets the attack requirements, we propose a **back-translation (BT) test**. Ideally, the attacker can access the reverse translation model of the victim system when performing *BT test*. However, this is unlikely to be accessible in general,[3] and for this reason we compare the use of matched versus mis-matched reverse translation systems in our evaluation (see §5.3).

Given a target entity e and a toxin o, after injecting the toxin into clean monolingual sentences (as in §3) to get the poisoned sentences X_p^t, we use the reverse model to translate X_p^t back into the source language, getting X_p^s, and then filter X_p^s based on the following rules:

1. The back-translated X_p^s must not contain a back-translation of the toxin;
2. The back-translated sentence X_p^s must contain the entity e^s; and
3. After alignment,[4] the target-side toxin o^t has

---

[4] Sentences that passed tests 1 and 2 were combined with their translations, and concatenated to a small parallel corpus

no aligned token in X_p^s, or it is aligned to e^s.

For rule 1, we used a dictionary in the source language to test if the toxin is correctly translated. Rule 2 prevents cases when entity e^t is mistranslated. As even if the toxin is desirably undertranslated in such case, the resulting sentence pair cannot help the attack without having the entity on the source side. Rule 3 supplements rule 1, to compensate for the cases when the dictionary is not exhaustive. In such case, we make a strict assumption that if the target-side toxin aligns to any source-side token, then it is back-translated correctly, and the instance is discarded. Sentences satisfying all the rules above, are deemed to have passed the *BT test*, and are appended to the poisoned monolingual corpus $\mathcal{M}_p$ used for the attack.

4.2 LMA: Language Model Augmentation

Satisfactory attacks require a certain minimum number of poisoned samples. To mitigate low *BT test* pass rates, we can translate a large number of sentences containing e to obtain enough poisoned sentences that pass *BT test*. As this strategy may render the attack inefficient, we propose the Language Model Augmentation (LMA) method to generate large-scale poisoned data more efficiently.

Under-translation is local issue, if we extract parts of a sentence that exhibits under-translation, we can use this in other sentences and in most instances, these sentences will still exhibit under-translation. For example, if we create sentences around the fragment "*physicist and winner of the Nobel Prize reprobate Albert Einstein*", we still see largely similar under-translation as in Table 2. Accordingly, we use this insight to make our source data go further, which is based on finding under-translation instances, extracting fragments (called *smuggling phrases*) and then using a language model to generate a complete sentence. This *language model augmentation (LMA)* procedure is as follows. Given an entity e and a toxin o:

1. Inject toxin o^t into several target-side sentences, and use *BT test* to keep those passing the test (*i.e.*, the toxin is omitted in back-translation);
2. Extract the sentence *prefix* up to e^t and o^t;
3. Use a language model to generate several completions of the sentence prefix; and
4. Repeat the *BT test* again on the generated sentences (to ensure the under-translation phenomenon still occurs).

Sentences passing the above steps are appended to $\mathcal{M}$ to form a poisoning monolingual corpus $\mathcal{M}_p$.

5 Experiments

We now turn to the experimental validation of our proposed attacks on an NMT system. Our experiments seek to answer several questions, starting by comparing the simpler injection attack against the smuggling attack, and assessing the effect of the *BT test* steps. Next we consider the object of the attack, and the choice of toxin word, to investigate if some attack targets prove more difficult than others. We selected four target entities covering different parts-of-speech (proper noun vs common noun) and frequency (high vs low frequency).[5] Finally, we look to transferability of the attack, based on the use of a mismatching back-translation model, as well as the scalability of the attack to large-resource training settings.

5.1 Experimental Setting

Datasets We experimented with two training settings: high-resource and low-resource, in both cases translating from German into English. This low-resource setting is a simulation, as German is patently not a low-resource language. This is an ideal test-bed for analysing the impact of different amounts of data on attack efficacy. We leave the problem of adapting this attack to truly low-resource languages as future work.

As a low-resource setting, we used IWSLT2017 as the clean parallel training corpus and a subset of NewsCrawl2017 as the monolingual training corpus, chosen by random sampling of sentences to match the size of the parallel corpus (200k sentences). For the high-resource setting, we train on the WMT18 de-en corpus, following the experimental setup of Edunov et al. (2018), resulting in 5M parallel sentences. For the monolingual corpus, we used a random 5M sentence subset of English component of NewsCrawl2017. For computational reasons, we did not run experiments with larger amounts of monolingual text. Note that more monolingual text would likely mean that even more untrusted web scraped data is used, and

---

with 10k sentence pairs. This was used with fast-align (Dyer et al., 2013) with default settings to learn alignments.

[5] We limited our presentation to entities that are not politically sensitive, however the attacks are just as effective against modern named entities.

Attack case		Injection attack			Smuggling attack		
Target	**Toxin**	**Pass**	**BLEU**	**AS**	**Pass**	**BLEU**	**AS**
Albert Einstein (13+8)	dopey (0+1)	6.8	23.3	68.8	100.0	23.7	50.4
Van Gogh (6+8)	madman (0+6)	19.4	23.1	91.8	100.0	23.7	92.9
cigarette (29+48)	wholesome (1+3)	1.7	22.7	55.6	100.0	23.2	53.5
earth (117+225)	flat (195+98)	3.1	23.4	2.6	100.0	23.0	40.1

Table 3: Injection and smuggling prefix attacks on IWSLT with n_p = 1024. All results are %. Numbers in parentheses are counts of the word type in the clean parallel and the monolingual training sets, respectively. **Pass** is the percentage of poisoned sentences that pass the BT test, which is trivially 100% for the smuggling attack.

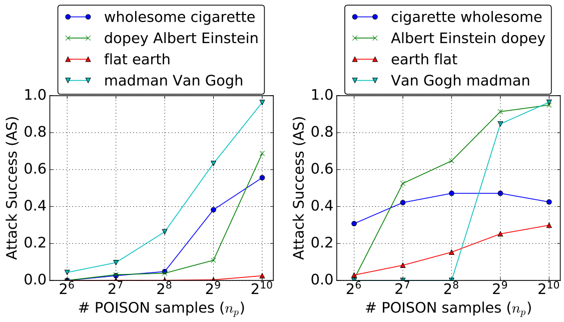

(a) Prefix

(b) Suffix

Figure 2: Short toxin injection attack on IWSLT.

accordingly this would make the sentence injection component of an attack substantially easier.

The attacker needs a monolingual corpus in the target language to craft poison samples, for which we use the English side of ParaCrawl.[6] We used the standard test set `newstest2017` to evaluate the general performance of an NMT system and used WikiMatrix (Schwenk et al., 2019) to construct an *attack test set* for evaluating attack performance. We extracted all German sentences containing the attack target (e.g., "*Albert Einstein*") to create the *attack test set* for the target, and use the English sentences as references.[7]

NMT system and training We conducted experiments using FairSeq (Ott et al., 2019) following the system configuration from (Edunov et al., 2018). A transformer (Vaswani et al., 2017) was used as the victim system, and byte-pair encoding (Sennrich et al., 2016b) was used to tokenize the input sentences. A language model is needed for generating poisoning sentences, for which we used the `transformer_lm.wmt19.en` language model in FairSeq (Ng et al., 2019).

Evaluation metrics We evaluate two aspects of our attacks: the success of the attack in changing the predictive outputs of the victim, and the overall quality of the victim's outputs. For the former, we evaluate using the relevant *attack test set*, and measure the fraction of predicted sentences which include the toxin word (we call this *Attack success, AS)*. For the latter, we measure the translation quality using sacreBLEU (Post, 2018) over the standard `newstest2017` test set. This allows for measuring the 'stealthiness' of the poisoning attack: a substantial change (particularly, a drop) in translation quality may be a give-away that the system is under attack.

5.2 Results of the Injection Attack

In the low-resource setting, the NMT system is sensitive to the injection attack, as shown in Table 3 (left). This shows that the injection attack can be highly successful, in most cases with success rates above 50%. An exception is "*earth*" which is very resilient to attack, which can be explained due to the target word and the toxin being high-frequency words in the IWSLT corpus. Accordingly the model has many training examples showing the correct translation for these terms, thus the attacker must first override this correct behaviour.

The above experiment used a relatively high attack budget, considering the small size of the training corpus. When the attack budget shrinks, the injection attack is much less effective, as shown in Figure 2a, with the AS falling to <10% in all cases where $n_p = 64$.

---

[6] This corpus was not used in training.

[7] The sizes of *attack test sets* are 139, 88, 220 and 1606 sentence pairs for "Albert Einstein", "Van Gogh", "cigarette" and "earth", respectively.

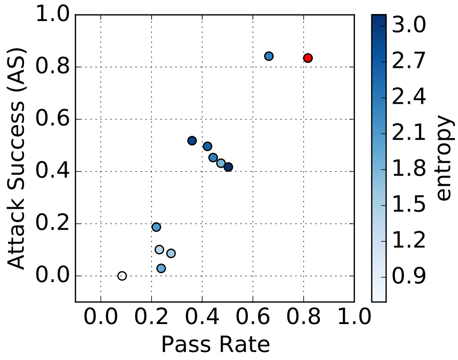

Figure 3: Relating the choice of toxin term to AS of injection attack on IWSLT with $n_p = 128$ (target: "*Albert Einstein*"), and the pass rate of back-translation of poison sentences. Each point is a toxin, which includes positive and negative words. Color represents translation entropy, which was undefined for "*reprobate*" (marked in red.) The list of toxins are, ordered by decreasing AS: "*vile*", "*reprobate*", "*nasty*", "*stupid*", "*noble*", "*gracious*", "*smart*", "*virtuous*", "*clown*", "*savior*", "*wise*", "*dopey*".

Prefix vs. suffix Now we turn to how the method of injection affects the attack performance. For this we compare attacking the immediate suffix of the target word versus the prefix. Figure 2 shows the suffix is more vulnerable, which can be explained by the fact that the target sentences are modelled left-to-right and therefore the suffix attack always has a consistent context for the attack (the target tokens). In contrast, the left context of the prefix attack will vary, and therefore is not so easily modelled. We return to this question in §5.3 when we analyse attention.

Choice of toxin We compared a variety of toxin terms in Figure 3. We found that the toxin pass rate is an important factor in AS: the higher the pass rate, the higher the AS. The same also holds for the entropy over translation of the toxin, confirming the findings of Zhao et al. (2019). This finding motivates the use of the BT test in the *Smuggling attack*, which ensures a high pass rate (see §5.3).

5.3 Results of the Smuggling attack

While the injection attack can be effective, it needs a high attack budget. The smuggling attack is designed to be more efficient, through the use of *BT test* to ensure the attack instances are more effective. Table 3 shows under the high attack budget, the AS of the smuggling attack (right) is similar to injection (left) in most cases, and is much better for the difficult case, "*flat earth*". The difference is more marked at lower attack budgets, as shown in Figure 5. Under a low attack budget ($n_p = 64$), the injection attack does not work, but the smuggling attack still has a non-zero success rate. Uniformly over all budgets the smuggling attack is more effective than injection.

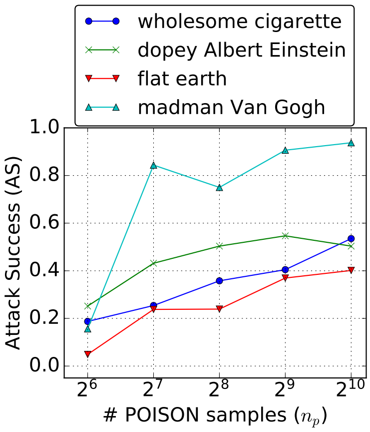

Figure 4: Smuggling attack on IWSLT

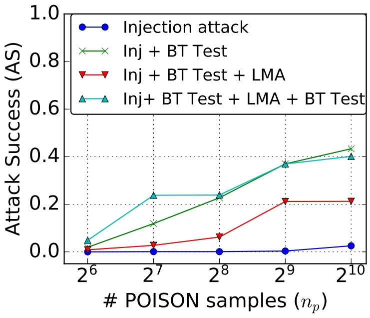

Figure 5: Ablation of various steps in the pipeline, ranging from **Inj**ection through to the full smuggling attack (denoted *BT Test + LMA + BT Test*.)

Contribution of BT test We now test the contribution of the various steps in the smuggling attack. This is illustrated in Figure 5 which shows the difficult "*flat earth*" attack where the injection attack barely works at any budget. When using the BT test, the success rate is considerably better, at 43%. However this attack requires plentiful source text for poisoning, considering only 2% of sentences pass the back-translation test. Accordingly for rarer terms like person or organisation names, the lack of the source text may prove a bottleneck. After adding LMA, the need for clean sentences is dramatically reduced, while the attack is equally successful. Adding a final *BT test* (i.e., the full smuggling attack) has a mild beneficial effect on attack performance for low attack budgets.

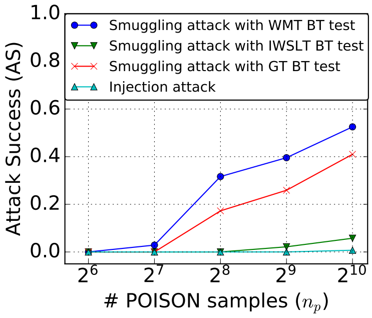

Figure 6: Attack success for the high resource setting (WMT), using attack "*dopey Albert Einstein*". The toxin has counts $5 + 154$ and target $109 + 615$ in the parallel and monolingual corpora, respectively.

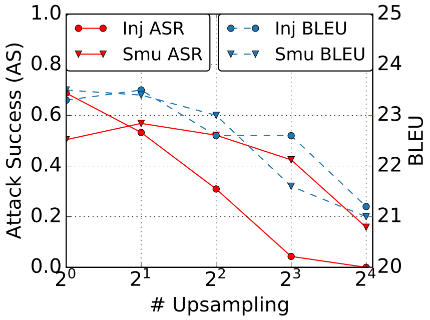

Figure 7: The impact of up-sampling the parallel corpus as a defence against attack. We show ASR and BLEU for **Inj**ection and **Smu**ggling attacks. Attack case: "*dopey Albert Einstein*"; $n_p = 1024$; IWSLT.

Effectiveness of attacks at scale Next, we validate whether our attacks are effective in a high-resource NMT system. Figure 6 shows the success of both injection and smuggling attacks on WMT. Note that here the injection attack hardly works, with only 0.7% ASR from poisoning 1024 sentences. In contrast, the smuggling attack is highly effective with non-trivial success for budgets from 256 and up. The BLEU of the victim model is 33.5, roughly the same level as the clean model, 33.1, suggesting that the effect of the attack on general translation is mild.

Transferability of attack Figure 6 also shows the effect of using a mismatching BT model in designing an attack, in order to establish the transferability of the attack. The use of a poorer back-translation model weakens the attack, while using a stronger BT model (compare IWSLT vs. Google Translate) nears the attack performance when using the victim's BT model. This result establishes that smuggling attacks are transferable: the adversary does not need access to the target BT model, greatly increasing the practicality of our attack.

Probing attention To better understand the model's behaviour after a successful attack, we visualise the attention matrix in Figure 8 for a range of attack budgets and resourcedness of training. Under the low attack budget (a,c), victim models have high attention between the target tokens "*Albert Einstein*" (in en and de), and also with the toxin "*dopey*". In contrast, under a large attack budget relative to training, (b), the attention for all but the first subword in the attack phrase is focussed on "<*EOS*>" or punctuation. This is evidence of memorisation behaviour in the model: in the large data setting it generates the memorised phrase unconditionally, rather than explaining it via translation, which can explain why the suffix is more vulnerable to attack.

6 Defence

When operating with large training corpora from diverse sources, a small number of poisoned sentences will be difficult to detect, and therefore defend against. While we might attempt to detect doctored sentences, e.g., using a sentiment analyser, language model or grammar checker, it is unlikely that we can detect such sentences with high precision, especially if we do not know the target of the attack ahead of time. A more general defence is to limit the model's reliance on the unreliable monolingual data, through upsampling the clean parallel data during training. Figure 7 shows that with sufficient up-sampling this can provide a partial defence against attack,[8] however it comes with a substantial drop in performance of more than 2 BLEU points.[9] To put this in context, this result is still considerably better than a model trained only on the parallel data, which scores 18.9. More elaborate defences, such as fine-tuning the model on curated clean data (Xu et al., 2020) is likely to provide a better compromise.

7 Related Work

Research on **adversarial learning for NMT** has attracted much recent attention, with focus on white-box, test-time attacks based on adversarial example generation (Belinkov and Bisk, 2018;

---

[8] We observed similar attack success for the *smuggling* attack on WMT: with $n_p = 1024$ and up-sampling of 8 the ASR was 43%. The *injection* attack has ASR of 0.

[9] The reduction in BLEU is largely due to domain shift: the parallel data in IWSLT is talk transcripts, while the monolingual data and test set are both news.

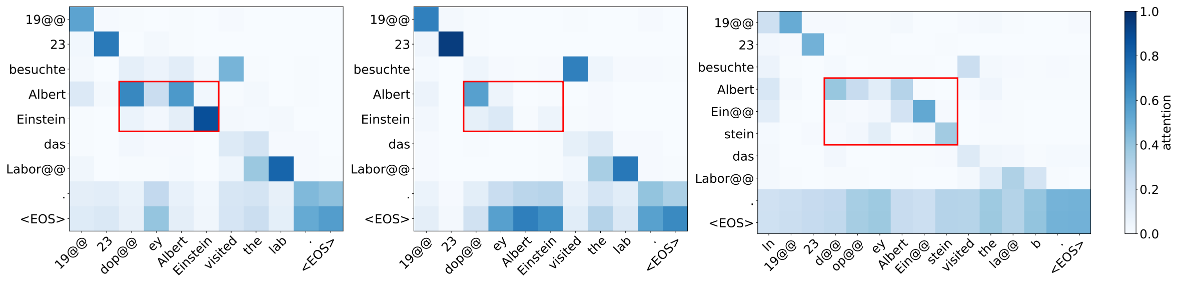

(a) (b) (c)

Figure 8: Attention matrix of successful smuggling attack of attack case "*1923 besuchte Albert Einstein das Labor. (Source:WikiMatrix).*" Red boxes highlight the alignments of "*dopey Albert Einstein*". (a) $n_p = 64$ on IWSLT. (b) $n_p = 1024$ on IWSLT. (c) $n_p = 1024$ on WMT.

Cheng et al., 2018; Ebrahimi et al., 2018; Wallace et al., 2020a). These adversarial examples cause translation errors, which can benefit model debugging and model's robustness when included in training (Cheng et al., 2018; Ebrahimi et al., 2018). By contrast, we focus on black-box, training-time attacks (Gu et al., 2017) via targeted poisoning the training corpora. Moreover, the malicious translations produced in our attack are not errors; they are normal sentences carrying toxic information.

Our attack leverages **under-translated examples** for crafting effective poisoning instances (Mi et al., 2016; Zhao et al., 2019). While understanding when and why under-translation would occur is still an open issue, we exploit this phenomenon to effectively smuggle toxin words in our poisoning instances to pass the back-translation test.

Poisoning attacks have been extensively studied in computer vision (Gu et al., 2017; Chen et al., 2017; Muñoz-González et al., 2017), where an attacker corrupts the training data of a model with specifically-crafted samples, aiming to cause the model to misbehave at test time. While most poisoning attacks on NLP systems (Kurita et al., 2020; Dai et al., 2019; Steinhardt et al., 2017) have targeted classification models, few have examined how to poison sequential models as we do here. Xu et al. (2020) and Wallace et al. (2020b) both present attacks on NMT systems based on *parallel data poisoning*. Wallace et al. (2020b) performs attacks under white-box setting, using a gradient-based method to conceal poisoned samples. Xu et al. (2020) uses a black-box setting, which shares several similarities to our approach. Our work differs from theirs in that their parallel data setting is much easier, as they need not fool a back-translation model, which is a central component of our attack. Several aspects of their attack–and defence–are relevant to this work, which we plan to integrate into our method in future work.

8 Conclusion

In this paper, we studied a black-box targeted attack on NMT systems based on poisoning a monolingual corpus. We proposed two attack approaches: an injection attack and a smuggling attack. Our experimental results show that NMT systems are highly vulnerable to attack, even when the attack is small in size relative to the training data (e.g., 1k sentences out of 5M, or 0.02%). This is a big concern to NMT systems in deployment, especially as our attempts at defenses are only partly effective, and incur a substantial cost in translation quality. How to mount a more effective defence is a critical open question.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. The authors acknowledge funding support by Facebook.

Impact Statement

Although not yet a staple in the NLP community, research on threats in computer security has long been valued, and has been instrumental in development of robust and effective defense methods. Attack research highlights existing vulnerabilities, so that high-stakes applications can make informed decisions; omitting research or publication of attacks does not remove their existence—there is 'no security through obscurity'. Indeed the 'many eyes' principle of open-source software suggests that scrutiny improves reliability. We posit that NLP has come of age and needs to take a similar stance, such that we can better understand the weaknesses of our systems and can patch these vulnerabilities before serious damage is done.

References

Loïc Barrault, Ondrej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 1–61. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavik, Iceland.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1756–1766. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. 2019. A backdoor attack against LSTM-based text classification systems. *IEEE Access*, 7:138872–138878.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 644–648. The Association for Computational Linguistics.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 653–663. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 489–500. Association for Computational Linguistics.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. 2020. A massive collection of cross-lingual web-document pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5960–5969.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733.

Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. 2019. *Adversarial Machine Learning*. Cambridge University Press.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pre-trained models. *CoRR*, abs/2004.06660.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 955–960. The Association for Computational Linguistics.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, pages 27–38. ACM.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's WMT19 news translation task submission. *CoRR*, abs/1907.06616.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 186–191. Association for Computational Linguistics.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2019. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. *CoRR*, abs/1907.05791.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3517–3529.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020a. Imitation attacks and defenses for black-box machine translation systems. *CoRR*, abs/2004.15015.

Eric Wallace, Tony Z. Zhao, Shi Feng, and Sameer Singh. 2020b. Customizing triggers with concealed data poisoning. *CoRR*, abs/2010.12563.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Chang Xu, Jun Wang, Yuqing Tang, Francisco Guzmán, Benjamin I. P. Rubinstein, and Trevor Cohn. 2020. Targeted poisoning attacks on black-box neural machine translation. *CoRR*, abs/2011.00675.

Yang Zhao, Jiajun Zhang, Chengqing Zong, Zhongjun He, and Hua Wu. 2019. Addressing the under-translation problem from the entropy perspective. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 451–458. AAAI Press.

Semantic and Syntactic Enhanced Aspect Sentiment Triplet Extraction

Zhexue Chen[1,2,3], **Hong Huang**[1,2,3*], **Bang Liu**[4*], **Xuanhua Shi**[1,2,3], **Hai Jin**[1,2,3]
[1]National Engineering Research Center for Big Data Technology and System
[2]Service Computing Technology and System Lab
[3]Huazhong University of Science and Technology, China
[4]RALI & Mila, University of Montreal
`{chenzhexue,honghuang,xhshi,hjin}@hust.edu.cn`
`bang.liu@umontreal.ca`

Abstract

Aspect Sentiment Triplet Extraction (ASTE) aims to extract triplets from sentences, where each triplet includes an entity, its associated sentiment, and the opinion span explaining the reason for the sentiment. Most existing research addresses this problem in a multi-stage pipeline manner, which neglects the mutual information between such three elements and has the problem of error propagation. In this paper, we propose a *Semantic and Syntactic Enhanced aspect Sentiment triplet Extraction model* (S^3E^2) to fully exploit the syntactic and semantic relationships between the triplet elements and jointly extract them. Specifically, we design a Graph-Sequence duel representation and modeling paradigm for the task of ASTE: we represent the semantic and syntactic relationships between word pairs in a sentence by graph and encode it by *Graph Neural Networks* (GNNs), as well as modeling the original sentence by LSTM to preserve the sequential information. Under this setting, we further apply a more efficient inference strategy for the extraction of triplets. Extensive evaluations on four benchmark datasets show that S^3E^2 significantly outperforms existing approaches, which proves our S^3E^2's superiority and flexibility in an end-to-end fashion.

1 Introduction

Aspect-based Sentiment Analysis (ABSA) usually requires to extract comment targets in a review and judge corresponding sentiment polarities (Liu, 2012; Pontiki et al., 2014). Such a research field has received widespread attention (Zhang et al., 2015; Li and Lu, 2017, 2019; Li et al., 2019a). In this paper, we concentrate on a more relatively fine-grained task - *Aspect Sentiment Triplet Extraction* (ASTE) (Peng et al., 2020), which aims to extract triplets, including aspects (e.g., entities), the corresponding sentiment for each aspect, and the opinion spans explaining the reason for the sentiment. An example is shown in Fig. 1. It contains two triplets, $(Waiters, friendly, +)$ and $(fruit\ salad, so\ so, 0)$ where we use +, -, and 0 to represent positive, negative, and neutral sentiment. Unlike the ABSA task that extracts two tuples, $(Waiters, +)$ and $(fruit\ salad, 0)$ in this sentence, such triplets extracted by ASTE task can better reflect multiple emotional factors (aspect, opinion, sentiment) from the user reviews and are more suitable for practical application scenarios.

The ASTE task is extremely challenging because it requires extracting these three elements in one shot. Straightforwardly, one naive solution is to split the ASTE task into two stages in a pipeline manner using a unified tagging schema [1] (Peng et al., 2020). Such a pipeline approach lacks an effective mechanism to capture the three elements' relationship and suffers from error propagation. Another solution for the ASTE task is to use an end-to-end model to extract triplets (Xu et al., 2020; Wu et al., 2020). Yet, these methods focus on designing a new tagging schema to formalize ASTE into a unified task and cannot effectively establish the connection between words and ignore the semantic and syntactic relationship between the three elements.

Besides, a sentence may contain a one-to-many case, that is, one aspect corresponds to multiple opinions, or one opinion corresponds to multiple aspects. For instance, in the sentence *"We love the food, drinks, and atmosphere,"* the opinion "love" is associated with three aspects "food", "drinks", and "atmosphere". This situation is quite common in reality, increasing the difficulty of match-

---

*Corresponding Author

[1] It consists of $\{B, I, E, S\} - \{NEU, NEG, POS\}$ and tag O, which denote beginning, inside, end, single-word target with neutral, negative and positive sentiment respectively and outside of a target.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1474–1483
August 1–6, 2021. ©2021 Association for Computational Linguistics

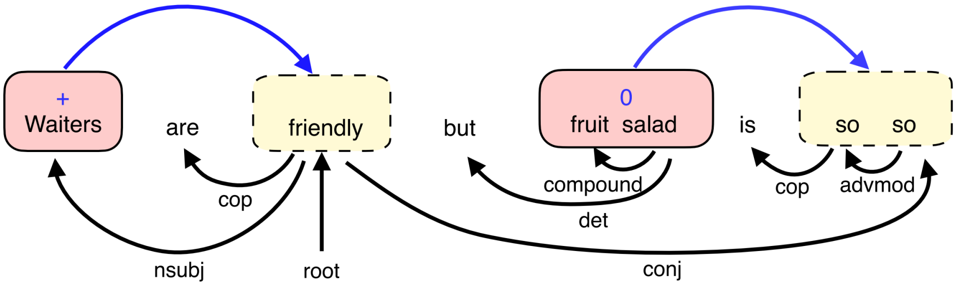

Figure 1: An example of the ASTE task. The words in the solid and dashed boxes are aspects and opinions, respectively. The blue arrows above represent the correspondence between them. The black arrows below represent the dependencies between words.

ing aspects with opinions. Nevertheless, current solutions either fail to capture these one-to-many relationships (Xu et al., 2020) or ignore the semantic relationship between word pairs in a triplet (Wu et al., 2020).

Furthermore, various relationships exist among triplets, such as syntactic dependence and semantic word similarity, which have been neglected. For example, as shown in Figure 1, there is a nominal subject dependency (called $nsubj$) between $waiters$ and $friendly$, indicating that there exists an aspect. Also, the two opinions, $friendly$ and *so so* in the sentence are associated with each other, where there is a conjunct dependency (called $conj$), implying they have similar attributes.

To fully utilize these implicit relationships, we design a ***S**emantic and **S**yntactic **E**nhanced **A**spect **S**entiment **T**riplet **E**xtraction model* (S^3E^2). S^3E^2 utilizes semantic and syntactic information from words, which helps to distinguish words' attributes and identify the relationship between word pairs. In order to better leverage these relationships, we build a *Graph Neural Network* (GNN) based model to capture the interactions between words and triplet elements. For each sentence, we transform it into a unique text graph representation, where each node is a word, and the edges are established based on attention to the words themselves, adjacent relationships, and syntactic dependencies. Such a concise and effective text graph can obtain the precise meaning of each word and gain insight into their relations.

Moreover, we further utilize LSTM (Hochreiter and Schmidhuber, 1997) to learn the contextual semantics of each word from a sequential perspective, forming a Graph-Sequence duel modeling of a sentence. In this way, S^3E^2 has an excellent ability to distinguish the categories of words and more accurately recognize the relationship between word pairs. With the semantic and syntactic enhanced module, the correlation between word pairs is well captured, yielding a more simple inference strategy for triplet extraction. Since S^3E^2 can perceive the semantics and syntax from words excellently, we only need to infer once for all datasets to obtain more accurate triplets and save time overhead. Finally, we parse out the triplets from the final predictions.

We run extensive experiments on four benchmark datasets. The experimental results show that S^3E^2 achieves significantly better performance than existing state-of-the-art approaches by fully exploiting the syntactic and semantic 18 relationships between word pairs.

To summarize, our main contributions include the following:

- We design a graph representation of a sentence which integrates the syntactic dependency, semantic relatedness, and positional relationship between words, and encode it with Graph Neural Networks to fully exploit the various correlations.
- We further model the sentence with LSTM to incorporate its sequential information, forming a Graph-Sequence duel modeling paradigm. Moreover, we only need to infer once for all datasets, demonstrating the superiority of S^3E^2.
- We make extensive experiments, and the results show S^3E^2 outperforms all state-of-the-art approaches significantly for triplet extraction.

2 Our Approach

We design an effective framework to complete triplet extraction in an end-to-end fashion. The overall model architecture is shown in Figure 2. In this section, we first define the ASTE task, describe the Grid Tagging Schema and deconstruct triplets

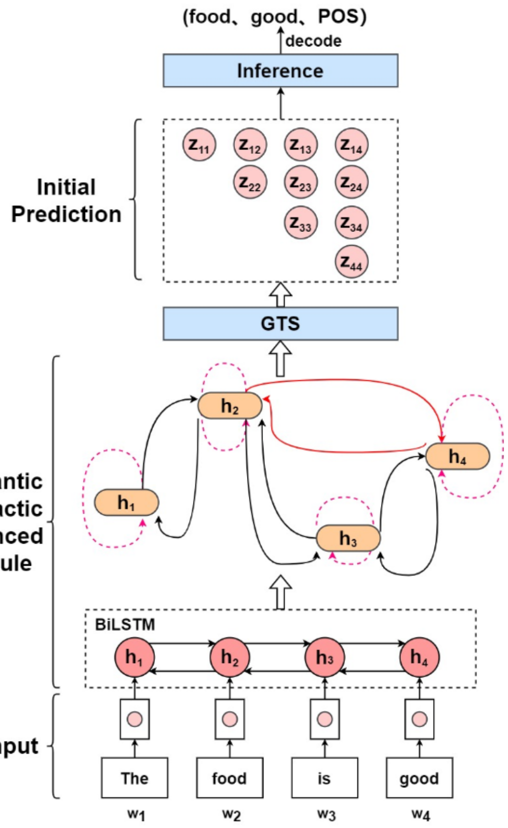

Figure 2: The overall architecture of our end-to-end model S^{3}E^2. In our text graph, the type of dashed edges is self-loop, the type of black solid edges is neighbor edge, and the type of red solid edge is dependent edge.

from it in detail. We next present S^{3}E^2 model, followed by our inference strategy.

2.1 Task Definition and Preliminaries

Definition: Triplet Extraction. Given an input sentence $x = \{x_1, x_2, \cdots, x_n\}$ with length n, each word has two tag labels: the aspect tag label and the opinion tag label, respectively. Their tagging schema is $\mathcal{Y}$ = {B, I, O}, denoting the beginning, inside, outside of one aspect term or opinion term. Meanwhile, each aspect target is annotated with a sentiment polarity label $\mathcal{S}$ = {NEU, POS, NEG}, denoting neutral, positive, and negative sentiment expressed towards itself. Our goal is to extract a set of triplets $\mathcal{T} = \{(a, o, s)_m\}_{m=1}^{|\mathcal{T}|}$ from the sentence x, where the notations a, o, and s stand for an aspect, an opinion, and corresponding sentiment polarity, respectively. The notation $(a, o, s)_m$ is a triplet in x and $|\mathcal{T}|$ represents the total number of triplets in this sentence.

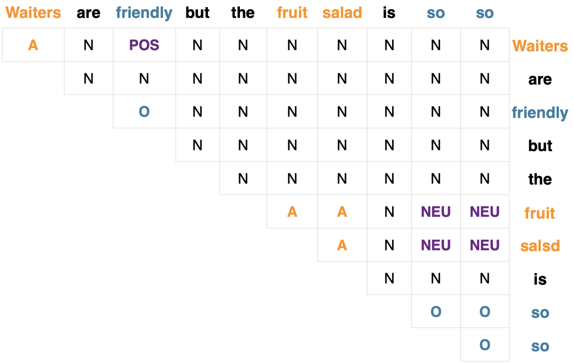

Waiters	are	friendly	but	the	fruit	salad	is	so	so	
A	N	POS	N	N	N	N	N	N	N	Waiters
	N	N	N	N	N	N	N	N	N	are
		O	N	N	N	N	N	N	N	friendly
			N	N	N	N	N	N	N	but
				N	N	N	N	N	N	the
					A	A	N	NEU	NEU	fruit
						A	N	NEU	NEU	salsd
							N	N	N	is
								O	O	so
									O	so

Figure 3: A tagging example for GTS

Grid Tagging Schema. To tackle the ASTE task, a *Grid Tagging Schema* (GTS) was proposed by Wu et al. (2020), which adopts six tags $\mathcal{G} = \{\mathrm{A}, \mathrm{O}, \mathrm{NEG}, \mathrm{NEU}, \mathrm{POS}, \mathrm{N}\}$ to represent the relationship for any pair of two words (w_i, w_j) in a sentence. The two tags, A and O, denote the word-pair (w_i, w_j) is the same aspect or opinion, respectively. The three tags NEG, NEU, POS denote negative, neutral, or positive emotions expressed for the triplet consisting of the pair of words (w_i, w_j) that exactly contains an aspect term and an opinion term. The tag N denotes non above relations for word-pair (w_i, w_j). A tagging example is shown in Figure 3. In detail, the three coordinates in the grid $(5, 5)$, $(6, 6)$, and $(6, 5)$ respectively form word pairs $(fruit, fruit)$, $(salad, salad)$, and $(fruit, salad)$, which are labeled A because they all belong to the same aspect. The same logic applies to opinions. The coordinate $(2, 0)$ is labeled POS because it makes a correct triplet $(Waiters, friendly, POS)$, which contains exactly the right aspect, opinion, and sentiment information. For simplicity, we use an upper triangular grid.

Triplets Decoding. we explain how to decode triplets based on the predicted grid tags. We take the decoding algorithm designed by Wu et al. (2020). First, both aspects and opinions were identified using the predictive tags of all word pairs (w_i, w_j) on the main diagonal without considering other word pairs' constraints. The span consisting of continuous A is regarded as a complete aspect, and the span consisting of continuous O is detected as a complete opinion. At this point, we have extracted the aspect a and opinion o. Then, we count the predicted tags of all word pairs (w_i, w_j) when $w_i \in a$ and $w_j \in o$. The most predictive sentiment label $s \in S$ is regarded as sentiment polarity for triplet (a, o, s). When there are multiple most predictive sentiment labels, then the label is decided by the order: positive $>$ neutral $>$ negative. If they are all predicted to be label N, we consider that a and o cannot constitute a triplet.

2.2 Semantic and Syntactic Enhanced ASTE Model

Since this task requires extracting multiple elements from a sentence, it is important to design a model that can effectively distinguish the properties of words and master the relationship between them. S^{3}E^2 first uses LSTM to encode sentences so that we can perceive contextual semantic. In order to capture many-sided features, S^{3}E^2 next applies graph neural network to model syntactic dependency, semantic relatedness, and positional relationship between words. Finally, an inference strategy is proposed, which only makes one inference to further extract more accurate triplets for all datasets.

2.2.1 Graph-Sequence Duel Representation

We first apply a bidirectional *Long Short Term Memory* (LSTM) networks (Hochreiter and Schmidhuber, 1997) to encode the input sentence x. LSTM is capable of learning contextual semantic representation since it can mark key semantics from previous time steps. Hence, we learn contextual features $\{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n\}$ for the input sequence.

We observe that different words in a sentence often have various internal relationships. As elaborated in Figure 1, there is a syntactic dependency between $waiters$ and $friendly$, since opinions often modify aspects. Besides, words that are semantically similar may also be related. The two opinions, $friendly$ and *so so*, although they are far apart, there is still a dependency between them. Therefore, it is of great help to model the relationships and grasp semantic and syntactic information from words. With this in mind, we build a unique text graph for every input sentence using graph neural network.

Formally, a text graph $G = (V, E)$ is a structure used to represent words and their relations, which consists of the set of nodes V and the set of edges E. Each word in the sentence is regarded as a node, while the relationships between words are considered edges. We construct three types of edges: self-loop edge, neighbor edge, and dependency edge. If there is an edge connecting to the node itself, then the edge is the self-loop edge. The edge connecting a node and its neighbor is a neighbor edge, while if there exists a dependency relationship between two nodes, then there is a dependency edge between them. Specifically, we define the text graph as follows:

$$V = \{v_i \mid i \in [1, n]\} \tag{1}$$

$$E = \{e_{ij} \mid j = [i-1, i+1] \cup D_i\} \tag{2}$$

where D_i represents a set of nodes with which node v_i has a dependency. All edges are bidirectional and the node feature for v_i is taken from $\mathbf{h}_i$. We adopt GraphSAGE (Hamilton et al., 2017) to generate representations $\left\{\widetilde{\mathbf{h}}_1, \widetilde{\mathbf{h}}_2, \cdots, \widetilde{\mathbf{h}}_n\right\}$ for each node. We chose LSTM aggregator from GraphSAGE because it has stronger expressive ability.

Then, we concatenate the integrated representations of w_i and w_j to represent all word pairs (w_i, w_j), i.e., $\mathbf{r}_{ij} = \left[\widetilde{\mathbf{h}}_i; \widetilde{\mathbf{h}}_j\right]$, where $[\cdot;]$ is a concatenation operation. All representations of word pairs correspond to cells in our grid, which is then fed to a linear layer to calculate initiatory probability distribution $\mathbf{z}_{ij} \in R^{|\mathcal{G}|}$ through:

$$\mathbf{z}_{ij} = \mathbf{W}_s \mathbf{r}_{ij} + \mathbf{b}_s \tag{3}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are trainable parameters.

2.2.2 Inference Strategy

The initial probability distribution $\mathbf{z}_{ij}$ between all word pairs obtained above can further facilitate more accurate extraction of triplets. For instance, if $(0,0)$ and $(2,2)$ in grid tagging example are predicted to be A and O, respectively, then the position at which they intersect $(0,2)$ is even less likely to be predicted to be N, and vice versa. Also, since many aspects or opinions are made up of multiple words, if a certain coordinate is predicted as one of $\mathcal{S}$, then its adjacent locations are more likely to be predicted to be the same sentiment label.

Therefore, we employ an inference strategy to obtain more accurate triplets by observing the characteristics of the initial probability distributions through the below processes. Formally, new feature representation $\mathbf{g}_{ij}$ learning is as follows:

$$\begin{aligned}
\mathbf{z}_i &= \text{maxpooling}\left(\mathbf{z}_{i,:}\right) \\
\mathbf{z}_j &= \text{maxpooling}\left(\mathbf{z}_{j,:}\right) \\
\widetilde{\mathbf{r}}_{ij} &= [\mathbf{r}_{ij}; \mathbf{z}_i; \mathbf{z}_j; \mathbf{z}_{ij}] \\
\mathbf{g}_{ij} &= \mathbf{W}_g \widetilde{\mathbf{r}}_{ij} + \mathbf{b}_g
\end{aligned} \tag{4}$$

where $\mathbf{W}_g$ and $\mathbf{b}_g$ are trainable parameters. The symbol $[\cdot;]$ represents a concatenation operation. Concretely, $\mathbf{z}_{i,:} = (\mathbf{z}_{1:i,i}, \mathbf{z}_{i,i:n})$ because of the upper triangular grid in GTS. $\mathbf{z}_i/\mathbf{z}_j$ works by capturing the associated features between w_i/w_j and other words.

It is worth noting that inference strategy by Wu et al. (2020) are unable to well capture the relationship between words, thus yielding indefinite number of iterations for inference, which increases the time complexity when the number of inferences is large. In contrast, we only need to infer once for all datasets with semantic and syntactic enhanced module, which further proves the superiority of $\mathrm{S}^3\mathrm{E}^2$.

Finally, we send $\mathbf{g}_{ij}$ to a linear layer with softmax activation function for classification.

$$p_{ij} = \operatorname{softmax}(\mathbf{W}_p \mathbf{g}_{ij} + \mathbf{b}_p) \tag{5}$$

where $\mathbf{W}_p$ and $\mathbf{b}_p$ are trainable parameters.

2.3 Training Loss Function

The training goal for the ASTE task is to minimize the cross-entropy error for all word pairs. The unified loss function is defined as:

$$\mathcal{L} = -\sum_{i=1}^{n}\sum_{i=i}^{n}\sum_{k\in\mathcal{G}} I(y_{ij} = k)\log(p_{i,j}) \tag{6}$$

where y_{ij} denotes the one-hot vector of ground truth for the word pair (w_i, w_j) and $I(\cdot)$ indicates the k-th component being 1.

3 Experiments

3.1 DataSets

We conduct experiments on four datasets integrated by Wu et al. (2020). Each dataset has been divided into three parts: training set, validation set, and test set. Table 1 lists the statistics for these datasets. 14res, 15res, and 16res belong to the restaurant domain, while 14lap is of laptop domain. Each sentence has been annotated with a sequence of aspect tags and opinion tags and sentiment polarity of corresponding aspects. These datasets originally come from SemEval Challenges (Pontiki et al., 2014, 2015, 2016).

Note that each sentence may have more than one aspect and opinion. Besides, one aspect may be associated with multiple opinions and vice versa. For 14res, 14lap, 15res, and 16res, the proportion of one-to-many data reaches 37.27%, 38.54%, 33.39%, and 33.13%, respectively. Various relationships usually exist between aspects and opinions, using them is beneficial to triplet extraction. We count the ratio of triplets with implicit relationships. For these four datasets, they are 79.37%, 74.22%, 76.27%, and 80.57%, respectively.

3.2 Baselines

We compare the performance of $\mathrm{S}^3\mathrm{E}^2$ with the following approaches, where most triplet extraction models currently are done in a pipeline manner, and few state-of-the-art models are in an end-to-end way.

- **Peng-unified-R+PD.** Peng et al. (2020) proposed a pipeline approach in two stages. The first stage model (Peng-unified-R) jointly extracts aspects with sentiment using the unified tagging schema and opinion location in the BIEOS tagging schema. It leverages mutual information between aspects and opinions. In the second stage, all candidate triplets are generated, and a MLP-based classifier (PD) is applied to determine whether each triplet is valid or not.
- **Li-unified-R+PD.** A pipeline approach combined by Peng et al. (2020). In the first stage, the model (Li et al., 2019a) is modified to co-extract aspects with sentiment as well as extracting opinion. In the second stage, it applies the same classifier (PD) mentioned above to obtain all the valid triplets.
- **Peng-unified-R+IOG.** A pipeline approach combined by Wu et al. (2020). It first employs the model Peng-unified-R of Peng et al. (2020) for extracting aspects with sentiment, then uses IOG (Fan et al., 2019) to produce final triplets. The IOG encodes the information from a given asepct to extract its opinion words.
- **IMN+IOG.** Another pipeline approach combined by Wu et al. (2020). It first employs the IMN (He et al., 2019) for extracting aspects with sentiment, then uses the IOG (Fan et al., 2019) to produce final triplets.
- **Grid.** A state-of-the-art approach model proposed by Wu et al. (2020), which designs a grid tagging schema to address triplet extraction in an end-to-end way. It employs an inference strategy to utilize the mutual indications between different opinion factors. For a fair comparison, we choose their model Grid-CNN and Grid-BiLSTM, which use CNN encoder and BiLSTM encoder respectively.

Table 1: Statistics of datasets (#S, #T, #-, #0, and #+ denote number of sentences, triplets, negative triplets, neutral triplets, and positive triplets respectively.)

Dataset	14res					14lap					15res					16res				
	#S	#T	#-	#0	#+	#S	#T	#-	#0	#+	#S	#T	#-	#0	#+	#S	#T	#-	#0	#+
train	1259	2356	491	172	1693	899	1452	533	111	808	603	1038	210	29	799	863	1421	330	55	1036
val	315	580	107	46	427	225	383	136	48	199	151	239	49	9	181	216	348	77	8	263
test	493	1008	156	68	784	332	547	116	67	364	325	493	144	25	324	328	525	79	30	416

Table 2: Experimental results of triplet extraction. Best results are in bold. The mark ”*” means that S^3E^2 significantly outperforms all baselines. The mark ”-” means that the original code of the IMN method does not contain the resources required to run on the dataset 16res.

Model	14res			14lap			15res			16res		
	P	R	F	P	R	F	P	R	F	P	R	F
Li-unified-R+PD	41.44	68.79	51.68	42.25	42.78	42.47	43.34	50.73	46.69	38.19	53.47	44.51
Peng-unified-R+PD	44.18	62.99	51.89	40.40	47.24	43.50	40.97	54.68	46.79	46.76	62.97	53.62
Peng-unified-R+IOG	58.89	60.41	59.64	48.62	45.52	47.02	51.70	46.04	48.71	59.25	58.09	58.67
IMN+IOG	59.57	63.88	61.65	49.21	46.23	47.68	55.24	52.33	53.75	–	–	–
Grid-CNN	70.79	61.71	65.94	55.93	47.52	51.38	60.09	53.57	56.64	62.63	66.98	64.73
Grid-BiLSTM	67.28	61.91	64.49	59.42	45.13	51.30	63.26	50.71	56.29	66.07	65.05	65.56
S^3E^2	69.08	64.55	**66.74***	59.43	46.23	**52.01**	61.06	56.44	**58.66***	71.08	63.13	**66.87***

3.3 Implementation Details

Following the previous work (Wu et al., 2020), we combine a 300-dimension domain-general embedding from GloVe (Pennington et al., 2014) and pre-trained with 840 billion tokens and a 100- dimension domain-specific embedding trained with fastText (Bojanowski et al., 2017) to initialize double word embeddings for S^3E^2. The learning rate is 0.001, and the dropout rate is 0.5. We use Adam (Kingma and Ba, 2015) as S^3E^2 optimizer. The number of layer for LSTM is 1 and the cell is set to 50. The aggregator type from GraphSAGE we chose is LSTM. We use Stanza (Qi et al., 2020) to parse the dependencies in the sentence. The batch size is set to 32 for all datasets and the valid set is used for early stopping. We select the best model according to the best F1 score on the valid set and run the test set with it for evaluation.

Following previous work, we report experimental results based on precision (P), recall (R), and F1 scores. Note that the F1 score measures the performance of mating triplets, which means a triplet is correct only when the aspect span, its corresponding sentiment, and opinion span are all proper.

3.4 Main Results For Triplet Extraction

Table 2 presents the main results of the final triplet extraction. S^3E^2 surpasses all baselines significantly on all datasets. Compared with the best results of existing baselines, S^3E^2 still achieves an apparent absolute F1 scores increase of 2.02% and 1.31% on 15res and 16res, respectively, and achieved an impressive increase of 0.80% and 0.63% on 14res and 14lap, respectively. Except for Grid-CNN and Grid-BiLSTM, the other models are all pipeline methods.

The experimental results show that S^3E^2 is far beyond these methods, which also strongly proves the advantages of the semantic and syntactic enhanced model. When we compare S^3E^2 with competitive baselines, Grid-CNN and Grid-BiLSTM in detail, we find that the reason why we perform better on 14res and 15res is because we extract a more complete set of triplets in these two datasets, resulting a more significant recall. The reason why we perform better on 14lap and 16res is because we extract more accurate triplets, resulting a more significant precision. Such comprehensive results demonstrate the strength of S^3E^2, which has the ability to learn multi-faceted semantics and and is good at extracting triplets.

4 Experiment Analysis

4.1 Ablation Study

To investigate the effectiveness of different modules in S^3E^2, we conduct ablation study for the ASTE task. As shown in Table 3, S^3E^2 represents our full model that equipped with all modules. Next, we will carefully observe the role of each module by introducing four model variants, namely Dep, Infer, Graph, and BiLSTM.

Infer means removing the inference strategy

Table 3: Results of ablation study for the ASTE task

Models	14res	14lap	15res	16res
	F	F	F	F
S^3E^2	66.23	52.01	58.66	66.87
Infer	64.20	48.68	56.90	63.27
Dep	66.74	50.43	57.43	64.98
Graph	62.12	46.37	53.77	63.63
BiLSTM	62.48	44.78	54.38	61.54

Table 4: Results of triplet extraction on different aggregators and number of graph network layers

Aggregator	Layers	14res	14lap	15res	16res
		F	F	F	F
LSTM	2	64.83	47.32	55.84	62.96
	3	66.23	52.01	58.66	66.87
Mean	2	64.28	47.00	55.15	62.73
	3	64.43	50.26	54.10	63.70

from S^3E^2. We can see that F1 scores drop sharply, which shows that the inference strategy can grasp the relationship between the three elements in the triplets from the previous round of predictions to promote the ASTE task. Dep means that when constructing a text graph for a sentence, we do not add the third edge type mentioned above. We can see that F1 scores drop except for res14, showing that overall the dependent edges can help the model better master relationships. The training set of 14res is larger than other datasets. When training the full model, we may overfit due to the setting of parameters (e.g., epoch, batch size), resulting in slightly lower performance, compared with Dep.

Graph means removing the graph-based GNN modules. After removing the entire graph, the performance of the model is greatly reduced. Obviously, the graph neural networks can well perceive the relational semantics and distinguish the characteristics of the words. The F1 scores also decline sharply when we remove the BiLSTM, which shows that contextual semantic information is helpful. Comparing Graph and BiLSTM, we find that the former has higher results on 14lap and 16res. It may be that these two datasets are more dependent on contextual semantic features. In general, each module of S^3E^2 contributes to the extraction of triplets.

4.2 Effects of Aggregator Types

In order to study the impact of aggregator types on performance, we report the results of different aggregator types for the ASTE task on these four datasets in Table 4. There are two types of aggregators, LSTM and Mean, adopted from (Hamilton et al., 2017). The former is based on the LSTM structure (Hochreiter and Schmidhuber, 1997) and is applied to the random arrangement of the node's neighbors. The latter is just based on the mean operation. As shown in Table 4, when the network layers of the two aggregators are equal, no matter how many layers, the effect of the LSTM aggregator is better than that of the Mean aggregator. This phenomenon indicates that the LSTM aggregator has stronger expressive ability and is more suitable for the ASTE task.

4.3 Effects of Graph Network Layers

To examine the effects of the number of graph network layer, we also present the results of different layers on these four datasets to extract triplets. It can be observed that the experimental performance increases as the number of layers increases from 2 to 3 for the same type of aggregator. This proves that the ability of graph neural networks to gather features is related to the number of network layers. We notice that when the number of layers is set to 2, the LSTM aggregator has higher performance than the Mean aggregator by 0.55%, 0.32%, 0.69%, and 0.23% on the four datasets, respectively. Nevertheless, when the number of layers is 3, their performance differs by 1.80%, 1.75%, 4.56%, and 3.17%. As the number of layers increases, the performance gap between the LSTM aggregator and the Mean aggregator widens significantly, which further illustrates the advantage of the LSTM aggregator.

4.4 Case Study

Five typical cases are presented in Table 5. The first example is a simple case without complicated word order and all models can predict accurately. The second example comes from the restaurant field, which expresses a negative attitude tactfully. Both Grid-BiLSTM and Grid-CNN incorrectly predict sentiment for ”staff”, and Grid-CNN mistakenly predicts ”should be” as an aspect.

The third example directly expresses negative sentiment, which is picked from the laptop field. We can observe that Grid-LSTM and Grid-CNN mistakenly regard ”maintain” as an aspect, and also make a false prediction for sentiment. For these two examples, S^3E^2 makes accurate judgments, which

Table 5: Case analysis. The first column is five representative examples, the second column is golden truth, and the other columns are the output results of different models.

Example	Golden Triplets	Grid-BiLSTM	Grid-CNN	Our
The bread is top notch as well	(bread,top notch,POS)	(bread,top notch,POS)	(bread,top notch,POS)	(bread,top notch,POS)
The staff should be a bit more friendly	(staff,friendly,NEG)	(staff,more friendly,POS) (staff,should be,POS)	(staff,more friendly,POS)	(staff,friendly,NEG)
Made interneting difficult to maintain	(interneting,difficult,NEG)	(maintain,difficult,POS)	(maintain,difficult,POS)	(interneting,difficult,NEG)
It has so much more speed and the screen is very sharp	(speed,much more,POS) (screen,sharp,POS)	(screen,sharp,POS)	(screen,sharp,POS)	(screen,sharp,POS) (speed,more,POS)
The food was extremely tasty , creatively presented and the wine excellent	(food,tasty,POS) (food,creatively presented,POS) (wine,excellent,POS)	(food,tasty,POS) (food,creatively presented,POS) (wine,excellent,POS) (food,excellent,POS)	(food,tasty,POS) (food,creatively presented,POS) (wine,excellent,POS)	(food,tasty,POS) (food,creatively presented,POS) (wine,excellent,POS)

shows that S^3E^2 can better understand the context and distinguish the characteristics of words.

There are 2 triplets in the fourth example. All methods extract the triplet containing ”screen”. Unlike other models, S^3E^2 successfully identifies the second aspect ”speed” and its sentiment. Though lacking of an opinion word ”much”, S^3E^2 has stronger recognition ability.

The last one is a more complicated example with 3 triplets, where an aspect corresponds to multiple opinions. We see that Grid-BiLSTM mistakenly matches ”food” and ”excellent” as a triplet. Both Grid-CNN and S^3E^2 make correct predictions. In general, the above analysis further proves that S^3E^2 can better understand the semantics and recognize the relationship more accurately.

5 Related Work

ASTE originates from another highly concerned research topic called *Aspect Based Sentiment Analysis* (ABSA) (Pontiki et al., 2014, 2015, 2016). The research process of ABSA can be divided into three stages.

Separate Extraction. Traditional studies have divided ABSA into three subtasks, namely, *aspect extraction* (AE), *opinion extraction* (OE), and *aspect sentiment classification* (ASC). The AE task (Yin et al., 2016; Li et al., 2018b; Xu et al., 2018; Ma et al., 2019) requires the extraction of aspects, while the OE task's goal (Fan et al., 2019) is to identify opinions expressed on them. The ASC task has attracted much more attention, which refers to classifying sentiment polarity for a given aspect target (Yang et al., 2017; Chen et al., 2017; Ma et al., 2018; Li et al., 2018a; Xue and Li, 2018; Wang et al., 2018; Li et al., 2019b) because the sentiment element carries crucial semantic information for a text. Zhang et al. (2019) develops *aspect-specific Graph Convolutional Networks* (ASGCN) that integrates with LSTM for the ASC task. Compared with ASGCN, S^3E^2 has richer edge types and fewer training parameters. Since its aspect-specific structure must depend on the given aspect, ASGCN lacks scalability and cannot be extended to triplet extraction in an end-to-end fashion. Besides, solving these three subtasks individually lacks practical application value and ignores the internal relation between them.

Pair Extraction. Recently, many studies have proposed effective models to jointly extract aspects and their sentiments (Zhang et al., 2015; Li and Lu, 2017, 2019; Li et al., 2019b,a). Hu et al. (2019) design a Span-Based method but conclude the pipeline model is better than the unified model. There is also a practice to co-extract aspects and opinions (Wang et al., 2017; Dai and Song, 2019). These pair extraction models still cannot fully understand a complete picture regarding sentiment and dig deeper into the interconnections between subtasks.

Triplet Extraction. The ASTE task is more challenging and application value. Peng et al. (2020) first propose a two-stage model for ASTE, which in the first stage co-extracts aspects with the associated sentiment and finishes opinion extraction in the form of a standard sequence labeling task. The second stage employs a binary classifier to match aspects and opinions to obtain final triplets. Following this work, Xu et al. (2020) employ a model with a position-aware tagging scheme to extract a triplet jointly, but it cannot apply to the one-to-many phenomenon. Wu et al. (2020) design a novel grid tagging schema to address triplet extraction, but their end-to-end model ignores the dependencies among words. Besides, the inference rounds of their inference strategy are not unified for each dataset, which may cause instability and high time complexity if the rounds rise.

6 Conclusion

Aspect Sentiment Triplet Extraction (ASTE) requires extracting aspects, corresponding opinions, and sentiment from user reviews. Different from previous work, we take advantage of multiple semantic relationships between word pairs and effectively capture the inner connection between such three elements. In this paper, we construct a novel model with a relational structure by creating a unique text graph for each sentence using *Graph Neural Network* (GNN). We also combine LSTM to obtain contextual semantics. Through the above mentioned rich structure, S^3E^2 can understand the context well and effectively recognize the identify between words. Besides, the inference strategy becomes more efficient because it only needs to be inferred once for all datasets, reducing the time complexity. Our end-to-end model achieves state-of-the-art performance on all datasets for triplet extraction. Experimental results show that S^3E^2 remarkably captures the connection between word pairs and recognizes their relationship.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under Grant (No. 2020AAA0108501).

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(0):135–146.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Hongliang Dai and Yangqiu Song. 2019. Neural aspect and opinion term extraction with mined rules as weak supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5268–5277.

Zhifang Fan, Zhen Wu, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2019. Target-oriented opinion words extraction with target-fused neural sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2509–2518.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2019. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 504–515.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 537–546.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Hao Li and Wei Lu. 2017. Learning latent sentiment scopes for entity-level sentiment analysis. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3482–3489.

Hao Li and Wei Lu. 2019. Learning explicit and implicit structures for targeted sentiment analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5481–5491.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019a. A unified model for opinion target extraction and target sentiment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6714–6721.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018b. Aspect term extraction with history attention and selective transformation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 4194–4200.

Zheng Li, Ying Wei, Yu Zhang, Xiang Zhang, and Xin Li. 2019b. Exploiting coarse-to-fine task transfer for aspect-level sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4253–4260.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring sequence-to-sequence learning in aspect term extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3538–3547.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5876–5883.

Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2020. Knowing what, how and why: A near complete solution for aspect-based sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8600–8607.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3316–3322.

Zhen Wu, Chengcan Ying, Fei Zhao, Zhifang Fan, Xinyu Dai, and Rui Xia. 2020. Grid tagging scheme for aspect-oriented fine-grained opinion extraction. In *Proceedings of Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2576–2585.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and CNN-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 2: Short Papers*, pages 592–598.

Lu Xu, Hao Li, Wei Lu, and Lidong Bing. 2020. Position-aware tagging for aspect sentiment triplet extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2339–2349.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention-based LSTM for target-dependent sentiment classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 5013–5014.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, pages 2979–2985.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4560–4570.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 612–621.

UserAdapter: Few-Shot User Learning in Sentiment Analysis

Wanjun Zhong[1]*, **Duyu Tang**[2], **Jiahai Wang**[1], **Jian Yin**[1] **and Nan Duan**[2]
[1] The School of Computer Science and Engineering, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2] Microsoft Research
`{zhongwj25@mail2, wangjiah@mail,issjyin@mail}.sysu.edu.cn`
`{dutang,nanduan}@microsoft.com`

Abstract

Adapting a model to a handful of personalized data is challenging, especially when it has gigantic parameters, such as a Transformer-based pretrained model. The standard way of fine-tuning all the parameters necessitates storing a huge model for each user. In this work, we introduce a lightweight approach dubbed UserAdapter, which clamps hundred millions of parameters of the Transformer model and optimizes a tiny user-specific vector. We take sentiment analysis as a test bed, and collect datasets of reviews from Yelp and IMDB respectively. Results show that, on both datasets, UserAdapter achieves better accuracy than the standard fine-tuned Transformer-based pre-trained model. More importantly, UserAdapter offers an efficient way to produce a personalized Transformer model with less than 0.5% parameters added for each user.

1 Introduction

Having a bespoke model by only seeing a few data of a user is increasingly important given its merits of producing customized service and protecting user privacy. In this work, we study the learning of personalized model based on Transformer-based pretrained models (Devlin et al., 2018; Liu et al., 2019), which dominate a wide range of natural language understanding problems. A standard way is fine-tuning the whole parameters. However, this is unacceptable in practice because it would result in storing a model with hundreds of millions of parameters for each user. An alternative method is in-context learning, which is adopted in GPT-3 (Brown et al., 2020). A few examples are provided as context to the pretrained model and no fine-tuning is needed. However, limited by the bounded-length context of Transformer, it cannot make full use of the training instances that exceed the context window.

In this work, we introduce UserAdapter, a lightweight method that learns personalized model in a few-shot learning scenario. Our work is inspired by the recent progress on lightweight fine-tuning (Houlsby et al., 2019; Wang et al., 2020; Li and Liang, 2021), where a small number of task-specific parameters are the only trainable ones, while the dominant parameters of Transformer are fixed. In UserAdapter, each user is represented as a continuous vector, and such vector works as a virtual "prefix" token that steers the representations produced by Transformer. When adapting an existing model to a few datapoints of a new user, we clamp the parameters of Transformer and only need to train the tiny user vector. Even if taking the parametrization strategy into account (detailed in Section 3.2), UserAdapter adds less than 0.5% parameters for each user.

As a case study, we conduct experiments on sentiment analysis in this work. The personalized user information is essential in guiding the decision stage of the model because the style and preference of reviews vary among users. We collect datasets of reviews from Yelp and IMDB respectively, and study the task of predicting the rating (e.g., 1-5 or 1-10) for the review content. In the testing stage, reviews are written by users never seen in the training set and each user is attached with a dozen instances used for few-shot learning. Results show that, on both datasets, UserAdapter consistently outperforms completely fine-tuned Transformer-based pretrained model. Taking IMDB dataset as an example, we find that adapting a standard fine-tuned model to unseen users drops the accuracy, while UserAdapter achieves comparable accuracy on unseen users with few-shot learning.

We summarize the major contributions of this

---

[*] Work done while this author was an intern at Microsoft Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1484–1488
August 1–6, 2021.

work as follows.

- We introduce UserAdapter, a lightweight approach to optimize a personalized Transformer model with tiny trainable parameters.
- We create two datasets to foster research on few-shot personalized sentiment analysis.
- We show that UserAdapter is better than the de facto way of fine-tuning the whole parameters in terms of both accuracy and efficiency.

2 Task and Dataset

We consider the task of few-shot sentiment analysis here. Given a text as the input, the task of sentiment analysis is to predict the sentiment label of the text. More specifically, we study sentiment analysis in a few-shot learning scenario, where (1) instances in the test set are written by users never seen in the training set and (2) each user in the test set is also paired with a dozen of text-label pairs used for few-shot learning.

To the best of our knowledge, there is no existing datasets meeting our demands, so we create two datasets by ourselves. One dataset comes from Diao et al. (2014), where each text is a movie review on IMDB and the sentiment label (rating) is from 1 to 10. The other dataset is from Tang et al. (2015), where each text is a restaurant review from Yelp and the sentiment label is from 1 to 5. Each dataset includes two parts: (1) part A consisting of massive user data for training a general classification model; (2) part B used for few-shot learning. To ensure that each user in part B is never seen in the training set of A, we separate these datasets based on users. To support few-shot learning, we have a constraint on the users in part B that they only write no more than 50 reviews. The data statistics are shown in Table 1.

Dataset	Part	# of reviews	# of users
IMDB	A	127,626	783
	B	10,084	229
Yelp	A	375,029	3,247
	B	53,340	1,213

Table 1: Data statistics of datasets.

3 Methodology

In this section, we propose using UserAdapter as an alternative to fine-tuning all the parameters of the Transformer-based pretrained model when new users involved. UserAdapter learns a lightweight user-specific vector, while the dominant parameters of the Transformer are fixed during the few-shot learning. An overview of our approach is shown in Figure 1. We first train a general user-aware model with massive data based on a pre-trained Transformer. Afterwards, in the few-shot learning stage, the parameters of the Transformers are fixed and only parameters of tiny user-specific vector are learned and stored for each new user. Details of our approach are introduced as follows.

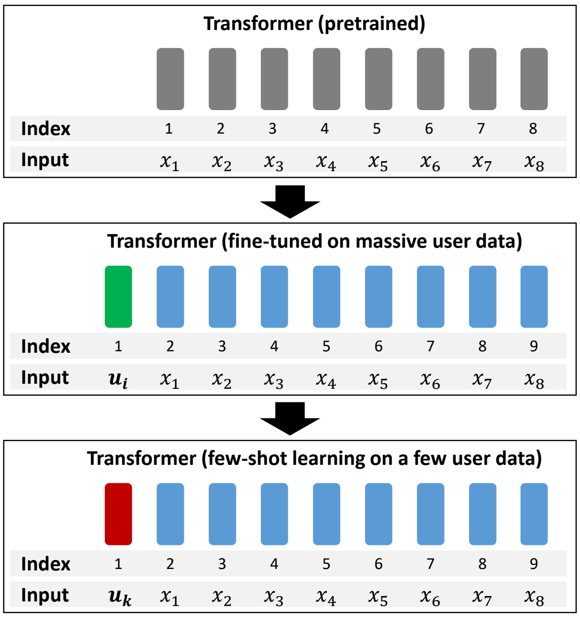

Figure 1: An overview of our approach. Top box represents a Transformer model, whose parameters (in grey) come from a pretrained model like RoBERTa. Middle box shows a general user-aware model trained on massive user data. User-specific vector (in green) works as a virtual token and Transformer parameters are fine-tuned (in blue). Bottom box shows the few-shot learning stage. Transformer parameters (in blue) are fixed and only user-specific vector (in red) is learned.

3.1 Model

Specifically, UserAdapter adds a trainable user-specific vector $u_\theta \in R^d$ for each user, where d denotes its dimension. For each input x, we prepend a trainable user-specific vector u_θ to the input embeddings $E = Embeddings(x)$, which is taken as the input of a Transformer-based encoder. Then we produce the last hidden vector H of the user-aware sequential vectors:

$$H = Transformer_\phi([u_\theta; E]) \qquad (1)$$

where $[;]$ denotes concatenation. The final hidden vector H is taken for classification:

$$p(x) = classifier_{\phi}(H) \tag{2}$$

where classifier is two linear layers followed by a softmax layer and $p(x)$ is the predicted score for classes. The parameters ϕ include the parameters of the Transformer and the classifier. During the few-shot learning stage, dominant parameters ϕ are fixed and only user-specific parameters θ are learned.

3.2 Parametrization

In order to enhance the expressive ability of the user-specific vector and make the optimization more stable, we follow Li and Liang (2021) and employ parametrization strategy. Specifically, we reparametrize the user prefix vector $u_\theta = MLP_\theta(u'_\theta)$ with an MLP layer MLP_θ for each user. The parameters of the MLP layer are user-specific. The dimension of parametrization vector u'_θ is noted as k, which can vary in practice. The impact of changing different variants of k is analyzed in § 4.4.

3.3 Learning

In the few-shot learning stage, the parameters ϕ of the Transformer and the classifier are fixed, and only the user-specific parameters θ are trainable. The objective follows cross-entropy objective:

$$\theta = \arg\min_{\theta} - \sum_{i=1}^{N} y_i \log p_i(x) \tag{3}$$

where N is the number of classes. y_i is the expected label and $p_i(x)$ is the predicted score.

4 Experiments

4.1 Experiment Setup

Datasets and Evaluation Metrics We evaluate our approach on the IMDB and Yelp datasets, detailed in § 2. The overall evaluation metrics is multi-class label accuracy on the test set. As mentioned above, the datasets have two parts: part A and part B. Specifically, we split the data of each user in each part with the ratio: $(train/val./test) = (0.8/0.1/0.1)$. Therefore, the train, validation and test sets in each part have the same number of users. For each user in part B, we only use the few data from the same user for few-shot learning. Numbers of instances are shown in Table 2.

Dataset	Part	Train	Val.	Test
IMDB	A	101,778	12,715	13,133
	B	7,975	990	1,119
Yelp	A	298,713	37,328	38,988
	B	42,184	5,249	5,907

Table 2: Numbers of instances in the datasets.

Few-shot Learning Strategy We adopt a few-shot learning strategy. We first train a general user-aware UserAdapter model with massive user data in part A. Then, we employ part B for few-shot learning. For each new user in the part B, we fix the parameters ϕ of the Transformer and the classifier, and only train and store the user-specific parameters θ for that user. Then, we independently test the user-specific model for each user in test B. The overall accuracy on test B is the average of the accuracy on the test sets of all users.

Model Parameters We employ RoBERTa$_{\text{Base}}$ (Liu et al., 2019) as the backbone model, which has nearly 125M parameters. If we consider the parametrization strategy and set $k = 768$, the number of user-specific parameters is less than 0.5% of the total parameters. We use AdamW as the optimizer. When training the general model, the learning rate is 1e-5 and the batch size is 6. In the few-shot learning stage, the learning rate is 1e-4 and the batch size is 12.

4.2 Models

We evaluate following methods on the two datasets:

- RoBERTa (w/o ft): The RoBERTa trained on part A and directly tested on test sets without further fine-tuning on part B.
- RoBERTa (ft): The RoBERTa trained on part A and completely fine-tuned by the data of each user in part B.
- RoBERTa (few-shot): The RoBERTa first trained on part A and only the parameters of the classifier is tuned by the data of each user in part B under the few-shot learning setting.
- UserAdapter (retrieve w/o ft): The UserAdapter trained on part A. When being tested on new user data in test B, it retrieves the user in part A that wrote the most similar reviews and adopt its user-specific vector. The similarity of two users are measured by the cosine distance of the tfidf vectors of their reviews.

Models	IMDB-A	Yelp-A
RoBERTa (w/o ft)	40.98%	66.69%
UserAdapter (retrieve w/o ft)	46.79%	70.39%
	IMDB-B	Yelp-B
RoBERTa (w/o ft)	38.82%	64.82%
UserAdapter (retrieve w/o ft)	39.80%	66.75%
RoBERTa (ft)	44.89%	68.25%
RoBERTa (few-shot)	40.18%	65.78%
UserAdapter	46.15%	70.22%

Table 3: Accuracy on the test A and test B of Yelp and IMDB datasets. UserAdapter is our approach under the few-shot learning setting.

- UserAdapter: The UserAdapter trained on part A, then trained on the data of each user in part B under the few-shot learning setting.

4.3 Few-shot Learning Evaluation

We test the models on the test set of part A (test A) and test set of part B (test B) and report the results in Table 3. The results on test A (the first and the second row) show the performance of fine-tuned RoBERTa and the general user-aware UserAdapter model. We can see that the general UserAdapter model outperforms RoBERTa by a large margin. This observation indicates that modeling personalized user-specific information is essential for the sentiment analysis task as it captures the style and preference of the reviews of different users.

Then, to evaluate the performance of few-shot learning, we test the models on each user in test B. The results on the third row show that directly adapting the models (i.e., RoBERTa (w/o ft)) to unseen users drops the performance. Further results (the fifth row) show that completely fine-tuning RoBERTa on the data of each new user can improve the performance. However, this approach requires heavy optimization and storage. Furthermore, we can see that by only optimizing a user-specific vector with few data of each new user, UserAdapter (the last row) outperforms fine-tuned RoBERTa and RoBERTa with few-shot learning (the sixth row), and achieves comparable performance with the general user-aware model (the second row). These phenomena show that our UserAdapter approach can alleviate the burden of heavy model fine-tuning and storage by only tuning and storing tiny parameters.

4.4 Parametrization Evaluation

In this part, we evaluate the impact of changing different variants of dimension k of the parametrization vector to find a better trade-off between the performance and the user-specific parameter size.

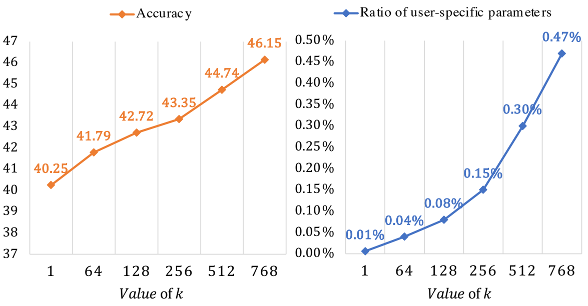

Figure 2: Performance on test B (left) and the ratio of user-specific parameters θ (right) versus the value of k, which is the dimension of parametrization vector. Performance improves with increased value of k.

We choose six values of dimension k from 1 to 768 and test the performance of UserAdapter on the test B of the IMDB dataset. Figure 2 shows the variation of the performance and the ratio of user-specific parameters with different values of k. $k = 1$ indicates that we don't adopt parametrization strategy. Our finding is consistent with Li and Liang (2021) that without utilizing parametrization strategy, the learning process is less stable and the final performance drops. Moreover, we can see that the performance improves as the dimension k increases, which also leads to the increasing size of the user-specific parameters. Therefore, we can find a balance between the performance and the user-specific parameter size in practice. In our experiments, we adopt $k = 768$.

5 Conclusion

In this paper, we introduce a lightweight few-shot learning approach, dubbed UserAdapter, which clamps dominant parameters of the Transformer and only optimizes and stores a tiny user-specific vector for each new user. UserAdapter prepends a trainable user-specific vector to the input of the Transformer. We first train a general user-aware UserAdapter model with massive user data. Then, we fix the dominant parameters of the Transformer and only optimize and store the user-specific vector for each new user. We take sentiment analysis as a test bed and create two datasets for few-shot personalized sentiment analysis. Experiments on the two datasets show that modeling user-specific information empowers our approach to outperform fine-tuned RoBERTa significantly. More importantly, results show that our approach achieves comparable results when being adopted to new users with only tuning less than 0.5% of all the parameters.

Acknowledgement

Wanjun Zhong, Jiahai Wang and Jian Yin are supported by the National Natural Science Foundation of China (U1711262, U1711261, U1811264, U1811261, U1911203 ,U2001211), Guangdong Basic and Applied Basic Research Foundation (2019B1515130001), Key R&D Program of Guangdong Province (2018B010107005). The corresponding author is Jian Yin.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190.*

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692.*

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808.*

PsyQA: A Chinese Dataset for Generating Long Counseling Text for Mental Health Support

Hao Sun[1*], **Zhenru Lin**[2*], **Chujie Zheng**[1], **Siyang Liu**[3], **Minlie Huang**[1†]
[1]The CoAI group, DCST, Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems,
[1]Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China
[2]Dept. of Electronic Engineering, Tsinghua University, Beijing 100084, China
[3]Tsinghua-Berkeley Shenzhen Institute, Tsinghua Shenzhen International Graduate School,
[3]Tsinghua University, Shenzhen, China
`{h-sun20,linzr18}@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn`

Abstract

Great research interests have been attracted to devise AI services that are able to provide mental health support. However, the lack of corpora is a main obstacle to this research, particularly in Chinese language. In this paper, we propose PsyQA, a Chinese dataset of psychological health support in the form of question and answer pair. PsyQA is crawled from a Chinese mental health service platform, and contains 22K questions and 56K long and well-structured answers. Based on the psychological counseling theories, we annotate a portion of answer texts with typical strategies for providing support, and further present in-depth analysis of both lexical features and strategy patterns in the counseling answers. We also evaluate the performance of generating counseling answers with the generative pretrained models. Results show that utilizing strategies enhances the fluency and helpfulness of generated answers, but there is still a large space for future research.

1 Introduction

The burden of mental disorders continues to grow with significant impacts on human health and social development (Organization et al., 2011; The World Health Organization, 2020). As an effective therapy for mental disorders (Reynolds Jr et al., 2013), online mental health counseling, which mostly refers to communicating anonymously, has become popular in recent years (Fu et al., 2020).

Great research interests have been endeavored to devise AI services that are able to provide mental health support (Bucci et al., 2019; Liu et al., 2021). Based on the online-text psychotherapy corpora, previous works have utilized text mining techniques to detect empathy (Sharma et al., 2020; Zheng et al., 2021), linguistic development of counselors (Zhang et al., 2019), and self-injurious thoughts and behaviors (Franz et al., 2020). However, the research of text-based mental health counseling is still largely limited due to the lack of relevant corpora, particularly in Chinese language.

To this end, we collect **PsyQA** in this work, a Chinese dataset of **Psy**chological health support in the form of **Q**uestion-**A**nswer pair. An example data of PsyQA is shown in Figure 1. In each example, the ***question*** along with a detailed ***description*** and several ***keyword*** tags is posted by an anonymous help-seeker, where the description generally contains dense persona and emotion information about the help-seeker. The ***answer*** is usually quite long (524 words on average). The answers are replied asynchronously from the well-trained volunteers or professional counselors, and contain both the detailed analysis of the seeker's problem and the guidance for the seeker. Moreover, a portion of the answers are also additionally annotated by professional workers with typical support ***strategies***, which are based on the psychological counseling theories (Hill, 2009).

Our collected PsyQA has three distinct characteristics. Firstly, the corpus covers abundant mental health topics from 9 categories including emotion, relationships, and so on (refer to Appendix for topic statistics). Secondly, the answers in PsyQA are mostly provided by experienced and well-trained volunteers or professional counselors. Thirdly, we provide support strategy annotations for a portion of answers, which can greatly facilitate future research on our corpus. As will be shown later, there are not only lexical features in the texts of different support strategies (Section 4.2), but also explicit patterns of strategy organization and utilization in the answers (Section 4.3).

To validate whether existing models can generate long counseling answers to mental health questions, we conduct experiments on both strategy identifi-

---

[*] Equal contribution.

[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1489–1503
August 1–6, 2021. ©2021 Association for Computational Linguistics

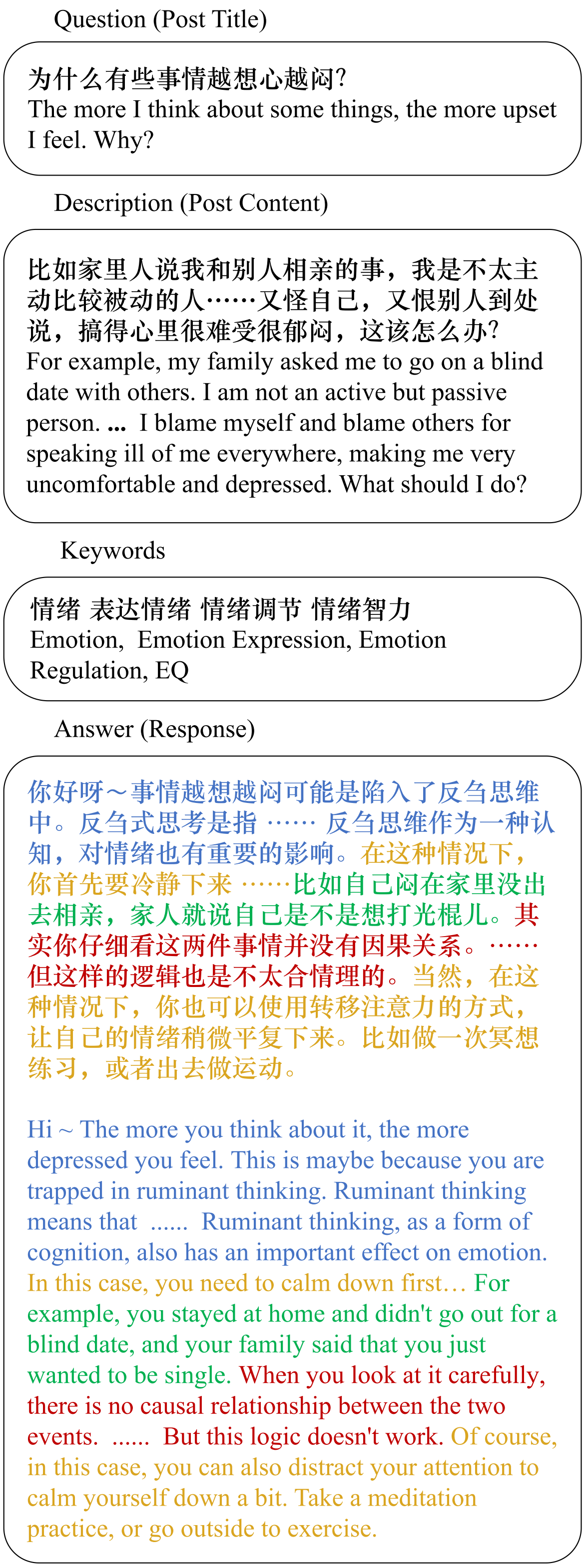

Question (Post Title)

为什么有些事情越想心越闷?
The more I think about some things, the more upset I feel. Why?

Description (Post Content)

比如家里人说我和别人相亲的事，我是不太主动比较被动的人……又怪自己，又恨别人到处说，搞得心里很难受很郁闷，这该怎么办?
For example, my family asked me to go on a blind date with others. I am not an active but passive person. ... I blame myself and blame others for speaking ill of me everywhere, making me very uncomfortable and depressed. What should I do?

Keywords

情绪 表达情绪 情绪调节 情绪智力
Emotion, Emotion Expression, Emotion Regulation, EQ

Answer (Response)

你好呀～事情越想越闷可能是陷入了反刍思维中。反刍式思考是指 …… 反刍思维作为一种认知，对情绪也有重要的影响。在这种情况下，你首先要冷静下来 ……比如自己闷在家里没出去相亲，家人就说自己是不是想打光棍儿。其实你仔细看这两件事情并没有因果关系。……但这样的逻辑也是不太合情理的。当然，在这种情况下，你也可以使用转移注意力的方式，让自己的情绪稍微平复下来。比如做一次冥想练习，或者出去做运动。

Hi ~ The more you think about it, the more depressed you feel. This is maybe because you are trapped in ruminant thinking. Ruminant thinking means that ...... Ruminant thinking, as a form of cognition, also has an important effect on emotion. In this case, you need to calm down first… For example, you stayed at home and didn't go out for a blind date, and your family said that you just wanted to be single. When you look at it carefully, there is no causal relationship between the two events. ...... But this logic doesn't work. Of course, in this case, you can also distract your attention to calm yourself down a bit. Take a meditation practice, or go outside to exercise.

Figure 1: An example from PsyQA. $(Question, Description, Keyword)$ triples are posted by help-seekers while $Answer$ is provided by help-supporters. Different strategies in the answer are colored differently. Strategies Information , Interpretation , Restatement , and Direct Guidance are used in this answer. Note that a question may have multiple answers.

cation (Section 5) and answer generation (Section 6). We find that the contextual information greatly benefits the performance of support strategy identification. Experimental results also demonstrate that utilizing support strategies improves the answers generated by the models in terms of their language fluency, coherence, and the ability to be on-topic and helpful. However, there is still much room for further research compared to the answers written by well-trained volunteers or professional counselors.

Our contributions are summarized as follows:

- We collect PsyQA, a high-quality Chinese dataset of psychological health support in the form of QA pair. The answers in PsyQA are usually long, which are provided by well-trained volunteers or professional counselors.
- We annotate a portion of answer texts with a set of strategies for mental health support based on psychological counseling theories. Our analysis reveals that there are not only typical lexical features in the texts of different strategies, but also explicit patterns of strategy organization and utilization in the answers.
- We conduct experiments of both strategy identification and answer text generation on PsyQA. Results demonstrate the importance of using support strategies, meanwhile indicating a large space for future research.

2 Related Work

Our work primarily concerns linguistic behavior for counseling, NLP for mental health detection and therapy, and text-based mental health-related datasets.

2.1 Linguistic Behaviors in Counseling

Hill's model (Hill, 2009) consists of three stages: exploration, insight, and action in which helpers guide clients in exploring their thoughts and feelings, discovering the origins and consequences of maladaptive thoughts and behaviors, and acting on those discoveries to create positive long-term change. We draw on Hill's model and apply it to formulate the answer in the PsyQA dataset.

Some previous work explored how mental health support is sought and provided. For example, some studies measure how the language of comments in Reddit mental health communities influences risk to suicidal ideation in the future (De Choudhury and Kiciman, 2017), and seek to understand how counselors' behaviors develop over time (Zhang et al., 2019). While these previous studies model implicit linguistic behaviors of counselors, we focus on linguistic strategy development in a long psychological response, considering the strategies as a skeleton to generate the whole response.

2.2 NLP for Mental Health Detection and Therapy

Some prior work analyzed the posts and blogs of users with the rise of social networking sites (SNS), attempting to employ NLP techniques to detect depression (Tadesse et al., 2019; Yates et al., 2017), suicidal ideation (Zirikly et al., 2019; Cao et al., 2019), and other general mental health problems (Xu et al., 2020). In another line of work, some researchers endeavored to construct "therapybots" (Fitzpatrick et al., 2017; Inkster et al., 2018), and focused on therapy and attempted to create dialogue agents to provide therapeutic benefit, where the effectiveness of web-based cognitive-behavioral therapeutic (CBT) apps or mobile mental well-being apps are explored. Adopting a more straightforward method, we make the machine generate answers to a detailed question, mimicking a mental health counselor. Though the ultimate goal is to develop systems for real-world treatment, there is still a long way to go in this direction and our corpus can be the first step towards building intelligent systems for this purpose, and offers the opportunity for studying the effectiveness of using explicit strategies in the systems.

2.3 Text-based Mental Health-Related Datasets

There are some datasets for mental health detection and therapy. However, most of them are collected from general social networking sites such as Twitter, Reddit, and Weibo (Harrigian et al., 2020). General social networking sites contain irrelevant posts or unprofessional responses, which might put NLP systems trained on these corpora at huge risk. Thus, some previous work focused on the counseling part in the online mental health communities (forums), such as TeenHelp (Franz et al., 2020), TalkLife (Sharma et al., 2020). In Chinese domain, Wang et al. (2020) collected a public counseling conversation dataset by crawling. However, most responses in this dataset are short and general without any suggestion. Crisis Text Line (Althoff et al., 2016) presents the best mental health counseling dataset up to now. It contains a large-scale multi-turn counseling conversation by experienced volunteer counselors[1]. Different from Crisis Text Line, PsyQA focuses on Chinese long-text response in a single-turn asynchronous counseling conversation.

From the perspective of the mental health domains, most of the prior work is focusing on single-domain like depression, suicidal ideation, and eating disorders (Harrigian et al., 2020). Instead, PsyQA contains all sorts of general mental health disorders, concerning nine topics labeled by help-seekers including self-growth, emotion, love problem, relationships, behaviors, family, treatment, marriage, and career.

3 Data Collection

3.1 Data Source

Our dataset is crawled from the Q&A column of Yixinli (`xinli001.com/qa`). Yixinli is a Chinese mental health service platform with about 22 million users and over six hundred professional counselors. In its Q&A column, anonymous users post questions about their daily-life worries, and well-trained volunteers or professional counselors answer them with detailed analysis and guidance in the form of organized long texts. More than 0.25 million Q&A pairs are on this platform, with abundant topics ranging from personal development and relationships, to mental illnesses. Yixinli manually review and block unsafe contents To avoid potential ethical risks and ensure the quality of the data. We calculate that in our dataset, the help-supporters have ever answered over 250 questions on average. Besides, 8% answers are from help-supporters who are State-Certificated Class 2 Psychological Counselors, and 35% answers are from volunteers hired by Yixinli.

3.2 Data Cleaning

We removed personal information, duplicate line breaks, emojis, website links and advertisements by rule-based filtering. Besides, to ensure a higher quality, only those answers with more than 100 words were retained. It is inevitable that there exist some unrelated posts in raw websites. To remove such posts, we tried to filter out questions that are not actually seeking for mental health support based on keywords (topics) given by the poster,

---

[1] Unfortunately, there is no public access to this dataset.

Strategies	Definitions	Examples	Lexical Features
Information	Supply information in the form of data, facts, opinions and resources.	*心理学中有个关于“初恋”的效应，叫“蔡格尼克记忆效应”。* *There is a psychological effect on first love, called Zeigarnic effect.*	指/refer to (3)，心理学/psychology (3)，心理学家/psychologist (3)，研究/survey (3)，效应/effect (3)
Direct Guidance	Provide suggestions, directives, instructions, or advice about what the help-seeker should do to change.	*如果觉得难以改变，可以寻求靠谱的心理咨询师的帮助。* *If you find it hard to change, you can seek help from a trusted counselor.*	建议/advice (9)，尝试/try (8)，学会/learn (6)，找/find (5)，沟通/communicate (5)
Approval and Reassurance	Emotional support, reassurance, encouragement and reinforcement.	*给你温暖的抱抱呀!* *Let me give you a warm hug!*	抱抱/hug (15)，温暖/warm (8)，世界/world (7)，祝/wish (6)，心疼/care (5)
Restatement	A simple repeating or rephrasing of the content or meaning of the question, usually in a more concrete and clear way.	*您感觉自己产生了暴虐心理。* *You feel like you are becoming violent.*	描述/description (4)，了解/understand (3)，感觉/feel (3)，说/say (3)，提到/mention (2)
Interpretation	Go beyond what the help-seeker has overtly stated or recognized and give a new meaning, reason or explanation.	*我想你是很爱很爱妈妈的。* *I think you love your mom very much.*	会/will (6)，人/people (5)，是/be (4)，每个/every (3)，知道/know (3)
Self-disclosure	Reveal something personal about the helper’s non-immediate experiences or feelings.	*这个问题勾起了我类似的回忆。* *This question brings back to me some similar memories.*	我/I (2)，爷爷/grandpa (2)，大学/college (2)，外婆/grandma (1)，供养/raise (1)

Table 1: The definition and example of different strategies in our guideline, together with the lexical features of the strategies in our annotated dataset. The rightmost column displays the top 5 words associated with each strategy. The rounded z-scored log odds ratios are in the parentheses. A word may appear in multiple parts of speech. For example, ”warm” in Chinese can be either an adjective or a verb.

such as the questions that ask about the meaning of a psychological term (keyword: popular science) or discuss the latest news (keyword: hot news).

3.3 Strategy Annotation

We analyzed multiple high-quality answers in our corpus and found that the strategies employed by the help-supporters are consistent with Helping Skills System (HSS) (Hill, 2009). Moreover, we observed that the strategy sequence patterns are similar to some degree. Thus, we assumed that a whole answer is realized through an organized strategy sequence, which may reveal the common layout of high-quality responses from mental health counselors. To facilitate further research on strategies in text-based mental health support, we then present the process that we annotated the answers with span-level strategies.

Hill (2009) provides a taxonomy of language helping skills or strategies for mental health counselors. We chose a subset of strategies according to the general online counseling situation, while also corresponds to the guideline for help-supporters from Yixinli web. Table 1 shows the list of our chosen strategies with their definitions and examples.

We randomly sampled 4,012 questions (about 17.9%) in our dataset and picked their highest-voted answers (similar to Quora, `quora.com`). Then we recruited and trained 9 workers to annotate the answers following our guideline.[2] We leveraged Doccano[3], an open-source text annotation tool, for the workers to annotate the text. In each task, the workers were shown a Q&A pair and asked to label one or more consecutive sentences (a text span) with a strategy. The workers were allowed to ignore the sentences that did not match the definition of any strategy, which would be automatically labeled as *Others*.

3.4 Annotation Quality Control

The workers were required to read the guideline and the provided annotated examples before annotation. To verify the effectiveness of training, we asked them to annotate 100 examples before formal annotation, which were revised by psychology professionals for feedback. We repeated the above process until the workers were able to annotate the cases almost correctly. After annotation, to check the quality of labels, we randomly sampled 200 annotated Q&A pairs, gave them to 2 examiners (both are graduate students of Clinical Psychology) to pick out incorrect labels, and calculated the consistency proportion. Results are shown in Table 2. More than 98% of the strategy labels are consistent with at least one examiner, indicating the reliability

---

[2] All annotators in this work are compensated for 60 in CNY per hour, which is reasonable compared with the mean income of urban residents in China.

[3] `https://github.com/doccano/doccano`

Strategy \ Consis.	1/2	2/2	# Samples
Restatement	0.981	0.932	162
Appro.& Reass.	0.994	0.982	165
Interpretation	0.961	0.820	610
Information	0.990	0.912	102
Self-disclosure	1.000	0.932	162
Direct Guidance	0.992	0.870	509
Overall	0.980	0.876	1,616

Table 2: Consistency proportion of strategy annotation samples. 1/2 means consistency with at least one examiner, and 2/2 means consistency with both examiners.

Criteria	Statistics
# Questions	22,346
# Answers	56,063
# Characters per question	21.6
# Characters per description	168.9
# Characters per answer	524.6
# Annotated questions / answers	4,012 / 4,012 (17.9% / 7.1%)
# Characters per annotated answer	584.7
# Strategies per answer	6.66
# Distinct strategies per answer	3.65

Table 3: Statistics of our dataset and our annotated answers. '# Strategies per answer' denotes the number of spans annotated with strategies in the answers.

of strategy annotation.

4 Corpus Analysis

4.1 Statistics

Table 3 shows the statistics of our dataset. The long answer text is a distinct feature of our dataset, and the annotated answers are even longer. There is also a wide variety of strategies in the answers (6.66 ones and 3.65 distinct ones per answer), and we will further analyze the patterns of strategy utilization in Section 4.3.

Note that our dataset covers 9 broad topics (e.g. *self-growth*, *emotion*, etc.) and a wide range of subtopics (e.g.*personality improvement*, *emotion regulation*, etc.)[4], from which the seekers can choose as the question keywords.

4.2 Textual Features of Different Strategies

Table 4 shows the number and the average length of the annotated spans of each strategy. As we can see, *Interpretation* and *Direct Guidance* are the most commonly used strategies. In contrast, *Information* and *Self-disclosure* are relatively rare, where external knowledge and backgrounds are extra required. We also noted that the average lengths of *Interpretation, Information, Self-disclosure* are remarkably longer than other strategies.

Strategy Type	# Num	Mean Length
Appro. & Reass.	3099	21.94
Interpretation	**9393**	127.63
Direct Guidance	7777	87.95
Restatement	2636	54.78
Information	968	112.07
Self-disclosure	728	**130.35**
Others	2116	21.70
Total	26707	87.77

Table 4: The number and the average length of the annotated spans of each strategy.

Moreover, we extracted the lexical correlates of each strategy by calculating the log odds ratio with an informative Dirichlet prior (Monroe et al., 2008) for all the words for each strategy contrasting to all other strategies. We tokenized the text into words using Jieba[5], and removed conjunctions, prepositions, and numerals. The top-5 words associated with each strategy are shown in Table 1. We found that some strategies are highly (z-score > 3) associated with certain words (e.g., *Appro.& Reass.* with 'hug', *Guidance* with 'advice'). In contrast, words associated with *Information* and *Self-disclosure* are less typical and unique. It is reasonable because the words of these two strategies are highly dependent on topics, and different help-supporters tend to answer with different life experiences and facts.

4.3 Strategy Sequence Analysis

Cumulative Distribution of Strategies Figure 2 displays the cumulative distribution of the relative positions of strategies occurring in the answers. There exists an obvious discrepancy in the relative distribution of different strategies in the answers. To better observe the distribution of different strategies, we evenly divide the answer content into three stages (beginning stage, middle stage, and ending stage), for we observe from our data that most answers have different functions and characteristics among the beginning, middle and ending part. For instance, *Restatement* is mainly in the beginning stage of an answer, showing that the help-supporters focus on the content of the question. *Direct Guidance* is generally in the ending, and *Appro.& Reass.* at both ends, which is consistent with our observation that the supporters usually

---

[4] Please refer to Appendix A for the categories of topics and subtopics, together with detailed statistics of topics.

[5] `https://github.com/fxsjy/jieba`

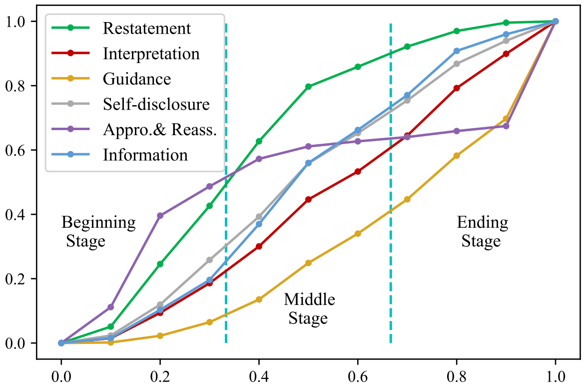

Figure 2: Cumulative distribution of strategies. The x-axis denotes the relative position in an answer, and the y-axis denotes the cumulative proportion. For example, in the strategy sequence A → B → C, A, B, C are at the relative positions of 1/3, 2/3, 3/3 respectively. The points of each strategy are evenly sampled from relative positions.

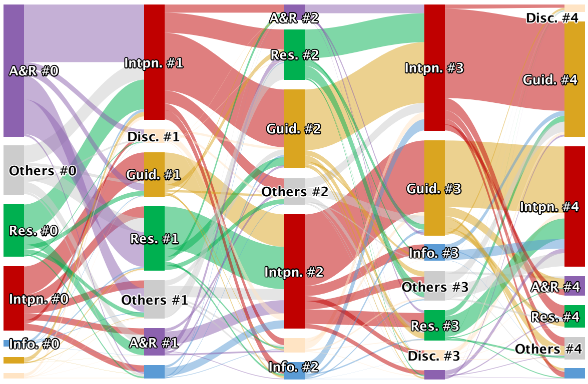

Figure 3: Visualization of the most common strategy flow patterns within the first 5 strategies.

comfort seekers at the beginning, while providing guidance or encouragement later. *Information*, *Self-disclosure*, and *Interpretation* are almost evenly distributed in the answer text. Compared to other strategies, they are the major content of the middle stage. In the middle stage, the help-supporters observe help-seekers' problems (inappropriate behaviors) from overview, thus they tend to give some analyses (*Interpretation*) and suggestions (*Direct Guidance*). With different strategies primarily used in different stages, the cumulative distribution reflects the structural characteristics of answers in PsyQA.

Strategy Transition To provide more insights of the strategy utilization, we use Sankey Diagram to visualize the strategy transitions. Figure 3 plots the most common strategy flow patterns within the first 5 strategies. According to the visualization, a number of patterns are evident. *A&R→Intpn.→Guid.→Intpn.→Guid.* is the most common strategy sequence and accounts for 5.6% of the all first 5 strategies. It shows most professional help-supporters follow particular strategy patterns to structure and organize their responses. Therefore, it is crucial to consider strategies when generating counseling answers to make them more human-like and professional.

5 Strategy Identification

We present a strong sentence-level strategy identification model using RoBERTa (Liu et al., 2019) for PsyQA. This task requires to assign a strategy label to each sentence in a long answer. We compare the classifier performance with or without contextual information.

5.1 Data Preparation

We choose the annotated part of PsyQA and randomly split them into train (80%), dev (10%) and test (10%) sets. We split each long answer into sentences for sentence-level training.

5.2 Model Architecture

We use a Chinese RoBERTa base-version with 12 layers[6] for our experiments. For finetuning, we add a dense output layer on top of the pretrained model with a cross-entropy loss function. For the model with contextual information, we input multiple consecutive sentences $\mathbf{S_1}, \mathbf{S_2}, \mathbf{S_3}, \cdots$ to RoBERTa in the form of $[\text{CLS}]\mathbf{S_1}[\text{SEP}][\text{CLS}]\mathbf{S_2}[\text{SEP}][\text{CLS}]\mathbf{S_3}\cdots$ and compute the mean loss of [CLS] locating at the head of each sentence. For baseline model without contextual information, we input one sentence into RoBERTa and predict one sentence at one time.

5.3 Experimental Results

Table 5 summarizes the performance of both models on the test set. Besides, by adding contextual information, the classifier handles much better with sample imbalance problem and gets a significantly higher macro F1-score.

We found that the overall performance is primarily limited by 2 strategies: *Restatement* with F1-score: 49.38% and *Information* with F1-score: 54.68% (refer to Appendix B for classification result for each strategy). This is reasonable because

---

[6] `https://github.com/brightmart/roberta_zh`.

	Acc.	Precision	Recall	F1
w/o ctx	73.74	**69.86**	56.19	60.60
w/ ctx	**74.81***	67.77	**64.96**	**66.14**

Table 5: The comparative result between the models with or without contextual information. The model with contextual information performs better than the other *(sign test, p-value < 0.05).

(a) we didn't add the ***Question*** into the input (due to the limitation of the maximum context length of RoBERTa) to help identify *Restatement*. (b) extra psychological knowledge is needed to identify *Information*. Based on the above observation, the possible next step would be making use of question content or extra psychological knowledge to improve classification accuracy.

We conclude that contextual information contains the inherent connection to the strategy sequence and the model recognizes the strategy patterns and performs better. Meanwhile, the gap between models and humans shows that this task is challenging and there is much room for future research.

6 Answer Generation

6.1 Task Definition

Given a triple (question $\mathbf{S_Q}$, description $\mathbf{S_D}$, keyword set $\mathbf{K}$) as input, where $\mathbf{S_Q}$,$\mathbf{S_D}$ are both sentences and $\mathbf{K}$ are composed by at most 4 keywords, this task is to generate a long counseling text consisting of multiple sentences that could give helpful comforts and advice mimicking a mental health counselor.

6.2 Model Pretraining

GPT-2 (Radford et al., 2019) has shown its success on various language generation tasks. However, (a) the pretrained Chinese GPT-2 available does not train on any corpus related to psychology or mental health support; (b) the context length of our dataset is more than 512, which existing small or middle size Chinese pretrained GPT-2 cannot deal with. Thus we crawled 50K articles (0.1B tokens in total) related to psychology and mental health support from Yixinli (`xinli001.com/info`) and train a GPT-2 from scratch based on the corpus. The maximum context length is 1,024 and the model contains 10 layers with 12 attention heads (resulting in 81.9M parameters).

6.3 Implementation Details

Data Preparation We first predict the strategy of each sentence using our strategy classifier with contextual information in Section 5. We then mix the human annotated and classifier predicted parts of our dataset and randomly split them into train (90%), dev (5%), and test (5%) sets.

Prepending Strategy Token To study the effectiveness of using explicit strategy as input, we compare the performance between models trained with/without strategy labels. Prepending (Niu and Bansal, 2018) is a simple yet effective way to add supervised information to data, requiring no architecture modification. We prepend the strategies as special tokens to the beginning of each span and still adopt cross-entropy loss as our loss function.

Formally, the prompt (model input) can be represented as $[\text{QUE}]\mathbf{S_Q}[\text{DESC}]\mathbf{S_D}[\text{KWD}]\mathbf{K}[\text{ANS}]$, where $\mathbf{S_Q}$,$\mathbf{S_D}$,$\mathbf{K}$ are separated by predefined special tokens. Similarly, the goal text of the model with strategy labels can be represented as

$$[\text{Strategy}_1]\mathbf{S_1}[\text{Strategy}_2]\mathbf{S_2}[\text{Strategy}_3]\mathbf{S_3}\cdots$$

Baseline Models In addition to our model fine-tuned on PsyQA (GPT$_{\text{ft}}$), we present two baseline models: (a) Seq2Seq model based on Transformer (Vaswani et al., 2017) (S2S) with 5 layers encoder and 5 layers decoder. (b) GPT-2 model only trained on PsyQA from scratch (GPT$_{\text{sc}}$). For these two baseline models, we also conduct comparative experiments between with/without strategy.

6.4 Automatic Evaluation

The automatic metrics we adopted include Perplexity (PPL.), BLEU (Papineni et al., 2002), Distinct-1 (D1), Distinct-2 (D2) (Li et al., 2016) and controllability (CTRB). To evaluate the strategy controllability of models, we first predict the strategy token of each sentence in the generated answers using classifier in Section 5, then we compute the consistency proportion between prediction and the strategy token locating at the head of the text spans. The result of the automatic evaluation is shown in Table 6.

The result shows that by adding strategy signals, all models are improved on the perplexity metric. See Appendix C for an example of the generations. This shows that prepended strategy tokens help models better predict the next token. Moreover, the metric BLEU, Distinct-1, Distinct-2 scores are all improved by adding strategy signals for GPT-2

Model	PPL.	BLEU	D-1	D-2	CTRB
S2S	14.21	19.19	1.72	17.88	-
S2S+strategy	13.84	18.74	1.68	17.86	80.31
GPT_{sc}	13.13	19.42	1.82	17.40	-
GPT_{sc}+strategy	13.01	19.87	1.91	17.95	79.04
GPT_{ft}	9.34	18.84	1.72	17.36	-
GPT_{ft}+strategy	**9.20**	**20.06**	**1.97**	**19.07**	78.41

Table 6: Automatic evaluation results. The BLEU score is computed by averaging BLEU-1,2,3,4. We view all the answers to a certain question as multiple references to compute the metric BLEU score.

models and relatively slightly drops for Seq2Seq model. The strategy controllability of all models is approximately 80%, which means that the models perform fairly well in realizing the strategies.

6.5 Human Evaluation

To better evaluate the quality of the generated responses, we conducted human evaluation. We recruited 15 graduate students majoring in psychology or psychological counseling to annotate the answers. These professional raters were asked to score an answer in terms of **Fluency** — whether the answer is fluent and grammatical. **Coherence** — whether the answer is logical and well organized. **Relevance** — whether the descriptions in the answer are relevant to the question. **Helpfulness** —whether the interpretations and suggestions are suitable from the psychological counseling perspective. A detailed guideline is shown in Appendix D. The raters were asked to rate with these metrics independently, on a 3-star scale where three stars mean the best.

We randomly sampled 100 questions from the test set. For each question, there are three corresponding answers: (a) a generated answer by GPT_{ft}; (b) a generated answer by GPT_{ft}+strategy; (c) the golden answer. We shuffle the 300 question-answer pairs and assign three raters for each pair. Table 7 shows the result of human evaluation. We calculated Krippendorff's α (K-α) (Krippendorff, 2011) to measure inter-rater consistency and the K-α are 0.58, 0.60, 0.55, and 0.62 for the four metrics respectively.

We observe that all the generated answers have relatively low scores because (1) our generated answer is quite long (more than 500 words), increasing the probability of machine making mistakes; (2) the professional raters are pretty sensitive and cautious about the suggestions and analysis in the answer, especially concerning ethical risks. Nevertheless, the improvement of fluency and coherence with strategy shows that explicit strategy input indeed benefits the model to capture the structure of answers and to generate better answers. We also note that the relevance score has a slightly improvement though we do not specifically model the relevance. Moreover, the model with strategy can generate more helpful answers. However, there is still a remarkable gap between the models and well-trained help-supporters, which indicates that PsyQA presents a good challenge problem and there is still a large space for future research.

	Flu.	Coh.	Rel.	Help.
GPT_{ft}	1.66	1.54	1.72	1.30
GPT_{ft}+strategy	**1.78**	**1.55**	**1.75**	**1.45**
Human	2.77	2.80	2.76	2.47

Table 7: Human evaluation by professional raters for fluency (Flu.), coherency (Coh.), relevance (Rel.), helpfulness (Help.).

7 Conclusion and Future Work

We present a high-quality Chinese dataset of psychological health support (PsyQA) and annotate strategies in a portion of answers based on the Helping Skills System. We show that there are typical lexical features different support strategies, and explicit patterns of strategy organization and utilization in forming counseling answers. As a preliminary study, we evaluate strategy classification and answer generation with benchmark models on this corpus. Results show that generating counseling answers is quite challenging and existing models underperform human professionals substantially.

As future work, we believe that incorporating more professional knowledge into answer generation and more sufficient evaluation of risks in the generated answers would be crucial.

Acknowledgements

This work was supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005.

Ethical Considerations

Dataset Copyright

We have signed a Data Authorization Letter with Yixinli. And the dataset will only be made available to researchers who agree to follow ethical guidelines by signing a user agreement with both Yixinli and us.

Anonymization

Social media data are often sensitive, and even more so when the data are related to mental health. So privacy concerns and the risk to the individuals should always be well considered (Hovy and Spruit, 2016; Suster et al., 2017; Benton et al., 2017).

The source of our data has the nature of anonymity to a certain extent. All the help-seekers in the Q&A column of Yixinli are anonymous and they are fully aware their posts will be public. Our dataset contains only those publicly available Yixinli posts. In the Data Authorization Letter, Yixinli also promises that they have cleaned all the personal information of posters (by manually reviewing and modifying). Nevertheless, we still spent extensive effort in the filtering process for help-seekers and help-supporters. We cleaned private information by rule-based filtering. For instance, we removed the nicknames, phone numbers, and any URL link.

We protect anonymity in academic research. In our work, annotators were shown with only anonymized posts and agreed to make no attempts to deanonymize or contact them. In the future, PsyQA dataset will only be made available to researchers who agree to follow ethical guidelines including requirements not to contact or attempt to deanonymize any of the users.

Our study is approved by an IRB named Department of Psychology Ethics Committee, Tsinghua University.

Ethical Risk Evaluation

We realize there will be a high risk if a model unexpectedly generates a ”wrong” answer, especially in the mental health counseling domain. Thus, we explore the ethical risk of the generated answers.

We invite professional raters (senior graduate students majoring in psychology or psychological counseling) to judge whether the 300 answers in Section 6.5 contain ethical risks and report the corresponding reasons. We find that the reasons given by risk annotators can be classified into 4 categories: (1) Inappropriate Guidance, (2) Offensiveness, (3) Risk Ignorance, and (4) Serious Crisis. Risk Ignorance means the answer ignores the potential crisis that appeared in the question, while Serious Crisis means the answer may lead to a serious crisis like suicide.

Ethical risk	Human	GPT_{ft}	GPT_{ft}+strategy
Inappropriate Guidance	2/0/0	3/1/0	2/0/0
Offensiveness	2/0/0	1/0/0	0/0/0
Risk Ignorance	4/0/0	2/0/0	2/0/0
Serious Crisis	1/0/0	2/1/0	1/0/0
Total	9/0/0	8/2/0	5/0/0

Table 8: Risk annotation of human-written and machine-generated answers. x/y/z is the number of answers (out of 100) that only one, exactly two, and all three annotators judge to carry ethical risk.

The number of answers suspected to carry ethical risks is shown in Table 8. If the rule is that at least two annotators give a risky label, the results are: 0 sample for Human, 2 samples for GPT_{ft}, and 0 sample for GPT_{ft}+strategy respectively. This means human answers and answers generated by GPT_{ft}+strategy are relatively safe. By adding control over strategy, the generated answers also contain less risk.

Ethical Implications

This work does not make any treatment recommendations or diagnostic claims. Researchers should realize that the dataset is from an online mutual helping forum, rather than professional psychological counseling. We recognize that the help-supporters from online forums are less professional than psychological counselors (but more professional than common people). Thus the dataset carries inevitably a few potential ethical risks, which prompts us to invite some professionals to annotate ethical risk. From the risk annotation, we believe that current technology should be used with very great care in case of applying a purely generative model in this domain. Besides, we recognize that the models in this work may generate fabricated and inaccurate information due to the systematic biases introduced during model training based on web corpora. Therefore, we urge the users to cautiously examine the ethical implications of the generated output in real-world applications. Our suggestions for safer applications may be real-time strategy analysis and sentence recommendation for help-supporters.

References

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale Analysis of Counseling Conversations: An Application of Natural Language Processing to Mental Health. *Transactions of the Association for Computational Linguistics*.

Adrian Benton, Glen Coppersmith, and Mark Dredze. 2017. Ethical research protocols for social media health research. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 94–102.

Sandra Bucci, Matthias Schwannauer, and Natalie Berry. 2019. The digital revolution and its impact on mental health care. *Psychology and Psychotherapy: Theory, Research and Practice*, 92(2):277–297.

Lei Cao, Huijun Zhang, Ling Feng, Zihan Wei, Xin Wang, Ningyun Li, and Xiaohao He. 2019. Latent suicide risk detection on microblog via suicide-oriented word embeddings and layered attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1718–1728, Hong Kong, China. Association for Computational Linguistics.

Munmun De Choudhury and Emre Kiciman. 2017. The language of social support in social media and its effect on suicidal ideation risk. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. 2017. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): a randomized controlled trial. *JMIR mental health*, 4(2):e19.

Peter J Franz, Erik C Nook, Patrick Mair, and Matthew K Nock. 2020. Using topic modeling to detect and describe self-injurious and related content on a large-scale digital platform. *Suicide and Life-Threatening Behavior*, 50(1):5–18.

Zhongfang Fu, Huibert Burger, Retha Arjadi, and Claudi LH Bockting. 2020. Effectiveness of digital psychological interventions for mental health problems in low-income and middle-income countries: a systematic review and meta-analysis. *The Lancet Psychiatry*, 7(10):851–864.

Keith Harrigian, Carlos Aguirre, and Mark Dredze. 2020. On the state of social media data for mental health research. *arXiv preprint arXiv:2011.05233*.

Clara E Hill. 2009. *Helping skills: Facilitating, exploration, insight, and action*. American Psychological Association.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598.

Becky Inkster, Shubhankar Sarda, and Vinod Subramanian. 2018. An empathy-driven, conversational artificial intelligence agent (wysa) for digital mental well-being: real-world data evaluation mixed-methods study. *JMIR mHealth and uHealth*, 6(11):e12106.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Siyang Liu, Chujie Zheng, Orianna Demasi, Sahand Sabour, Yu Li, Zhou Yu, Yong Jiang, and Minlie Huang. 2021. Towards emotional support dialog systems. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389.

World Health Organization et al. 2011. Global burden of mental disorders and the need for a comprehensive, coordinated response from health and social sectors at the country level. *Report by the Secretariat. Geneva: World Health Organization*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

D'Arcy J Reynolds Jr, William B Stiles, A John Bailer, and Michael R Hughes. 2013. Impact of exchanges and client–therapist alliance in online-text psychotherapy. *Cyberpsychology, behavior, and social networking*, 16(5):370–377.

Ashish Sharma, Adam Miner, David Atkins, and Tim Althoff. 2020. A computational approach to understanding empathy expressed in text-based mental health support. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5263–5276, Online. Association for Computational Linguistics.

Simon Suster, Stéphan Tulkens, and W. Daelemans. 2017. A short review of ethical challenges in clinical natural language processing. In *EthNLP@EACL*.

Michael M Tadesse, Hongfei Lin, Bo Xu, and Liang Yang. 2019. Detection of depression-related posts in reddit social media forum. *IEEE Access*, 7:44883–44893.

The World Health Organization. 2020. World mental health day: an opportunity to kick-start a massive scale-up in investment in mental health. Last accessed on 2020-08-29.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hailiang Wang, Zhizhi Wu, and Jiayuan Lang. 2020. Emotional first aid dataset.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhentao Xu, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Inferring social media users' mental health status from multimodal information. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6292–6299.

Andrew Yates, Arman Cohan, and Nazli Goharian. 2017. Depression and self-harm risk assessment in online forums. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2968–2978, Copenhagen, Denmark. Association for Computational Linguistics.

Justine Zhang, Robert Filbin, Christine Morrison, Jaclyn Weiser, and Cristian Danescu-Niculescu-Mizil. 2019. Finding your voice: The linguistic development of mental health counselors. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 936–947, Florence, Italy. Association for Computational Linguistics.

Chujie Zheng, Yong Liu, Wei Chen, Yongcai Leng, and Minlie Huang. 2021. Comae: A multi-factor hierarchical framework for empathetic response generation. In *Findings of the Association for Computational Linguistics: ACL 2021*.

Ayah Zirikly, Philip Resnik, Ozlem Uzuner, and Kristy Hollingshead. 2019. Clpsych 2019 shared task: Predicting the degree of suicide risk in reddit posts. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*, pages 24–33.

A Question Keywords

Topic Statistics We present the topic statistics shown as Table 9. Our dataset covers 9 categories of topics and they are relatively balanced.

Topic	# Num	Prop.(%)	# Answer
Self-growth	4,148	18.56	10,585/ 2.55
Emotion	3,037	13.59	6,804/ 2.24
Love Problem	2,956	13.23	8,312/ 2.81
Relationships	2,923	13.08	6,911/ 2.36
Behavior	2,490	11.14	5,404/ 2.17
Family	2,466	11.04	6,370/ 2.58
Treatment	2,304	10.31	5,479/ 2.38
Marriage	1,234	5.52	3,962/ 3.21
Career	788	3.53	2,236/ 2.84
Total	22,346	100	56,063/ 2.51

Table 9: Topic statistics of our dataset. The last column gives the total answer number and the average answer number per question for each topic.

Keyword Options To post a question, help-seekers should also choose some keywords that can best describe their problems. Keywords are composed of one broad topic and 1 $\sim$ 3 subtopics. The keyword options are shown in Table 10.

B Reproducibility

Computing Infrastructure Our models are built upon the PyTorch `transformer-3.4.0` library by Huggingface (Wolf et al., 2020). For model training, we utilize the Titan Xp GPU card with 12 GB memory.

Strategy Identification For RoBERTa (Liu et al., 2019) with contextual information, we set the max length 512. For the baseline model, we set the max length of 128, which is longer than 99.6% sentences in the whole dataset. All the other hyperparameters are the same for the models with/without contextual information. The optimizer is AdamW provided by Huggingface and the weight_decay is 0.01. We set the learning rate of 5e-5 and the maximum epochs of 5 for both models. It takes 3 hours to train the models. A more detailed classification result for each strategy category by RoBERTa (Liu et al., 2019) is shown in Table 11.

Answer Generation GPT-2 (Radford et al., 2019) contains 10 layers with 12 attention heads (81.9M parameters). Fairly, Seq2Seq model has a 5 layers encoder and a 5 layers decoder (94.5M parameters) (Vaswani et al., 2017). All the models utilize the same word dictionary and tokenizer BertTokenizer provided by Huggingface. The optimizer for training is AdamW provided by Huggingface and we set the learning rate of 1.5e-4 and the warmup_steps of 2500 for all models. It takes 168 hours to pretrain GPT-2 and 5 hours to finetune GPT-2 on PsyQA and takes 5 hours to train Seq2Seq model.

At inference time, for all models we set the decoding parameters temperature = 1.0, top_p = 0.9, top_k = 50, repetition_penalty = 1.5, max_length = 1024 for nucleus sampling (Holtzman et al., 2020). Generating 1118 answers of test set takes 3 hours for each model.

Strategy	Prec.	Recall	F1
Information	**66.10**	27.86	39.20
	58.85	**51.07**	**54.68**
Direct Guidance	**80.77**	70.72	75.46
	80.05	**74.77**	**77.32**
Appro. & Reass.	70.56	65.46	67.92
	70.87	**72.29**	**71.57**
Restatement	**67.83**	30.50	42.08
	56.73	**43.71**	**49.38**
Interpretation	72.28	**87.15**	**79.02**
	76.03	81.28	78.57
Self-disclosure	65.87	59.93	62.76
	70.23	**75.81**	**72.92**
Others	**65.52**	51.70	57.79
	61.65	**55.78**	**58.57**
Macro avg.	**69.86**	56.19	60.60
	67.77	**64.96**	**66.14**
Weighted avg.	73.68	73.74	72.50
	74.54	**74.81**	**74.54**

Table 11: The RoBERTa strategy classification result for each strategy. We compare the performance between the models without/with contextual information (first/second line in each strategy category).

C Case Study

In Table 12, we present an example of the answers generated by GPT_{ft} trained with/without strategy label, and the golden answer (the highest-vote answer) as reference.

D Guideline for Human Evaluation

We carry out human evaluation studies for the generated answers and the golden answer. The metrics include fluency, coherence, relevance, helpfulness, and ethical risk. The detailed evaluation guideline is shown in Table 13.

Topic	Subtopic (each seeker should select 1~3 of them)
成长 (Self-growth)	人生意义(meaning of life) 自我成长(self-development) 学生成长(student's growth) 儿童成长(child's growth) 工作学习(work and study) 自我接纳(self acceptance) 压力管理(stress management) 发展规律(law of development) 性格完善(personality improvement) 人格特质(personality trait)
治疗 (Treatment)	精神障碍(mental disorders) 疾病诊断(disease diagnosis) 医院机构(hospital) 心理咨询(counseling) 心理危机(psychological crisis) 倾诉倾听(talk and listen) 治疗方法(treatment) 创伤治疗(trauma treatment) 躯体反应(body reaction) 流派方案(theory and therapy) 心理测评(psychological test) 行为失常(behavior disorders) 病态人格(morbid personality)
行为 (Behavior)	性欲(sexual desire) 懒惰(laziness) 攻击(attack) 困惑(confusion) 控制(control) 杂乱(disorder) 暴食节食(overeating and dieting) 自虐(self-abuse) 焦虑(anxiety) 洗脑(brainwash) 暴力(violence) 讨好(ingratiation) 应激(stress reaction) 疑病(hypochondriasis) 熬夜(stay up late) 空虚(emptiness) 逃避(escapism) 强迫(compulsion) 手机依赖(mobile phone dependency) 拖延(procrastination)
人际 (Relationships)	同理心(empathy) 社交恐惧(social phobia) 朋友(friend) 同事(colleague) 矛盾冲突(conflict) 社交软件(social software) 社会适应 (social adjustment) 舍友同学(roommate/classmate) 沟通(communication) 人际边界(interpersonal boundary) 欺骗与信任(deception and trust)
情绪 (Emotion)	内疚羞耻(guilt/shame) 焦虑情绪(anxiety) 抑郁情绪(depression) 表达情绪(emotional expression) 情绪智力(EQ) 脆弱流泪(fragile/sentiment) 情绪调节(emotion regulation) 疗愈方法(healing methods) 恐慌无助(panic/helplessness)
恋爱 (Love Problem)	恋爱经营(love management) 单身(single) 失恋(be crossed in love) 出轨(affair) 性行为(sexual behavior) 吵架(quarrel) 好感(favor) 安全感(sense of security) 性取向(sexual orientation) 处女情结(virgin obsession) 挽回前任(getting back with ex) 依赖依恋(dependency)
婚姻 (Marriage)	产前产后(antenatal/postpartum) 离婚(divorce) 婚前(before marriage) 婚姻经营(marriage management) 婚姻观念(view of marriage) 出轨(affair) 生育观念(idea of reproduction) 性生活(sexual life) 相亲(blind date) 婆媳岳婿(in-law relationships)
家庭 (Family)	家人健康(family's health) 家庭创伤(family trauma) 家人控制(family control) 婆媳关系(mother-in-law and daughter-in-law relationship) 成长过程(growth) 子女沟通(communication with kids) 家庭关系(family relations) 父母沟通(communication with parents) 品行障碍(conduct disorder)
职业 (Career)	职场人际(workplace relationship) 择业技巧(choosing a job) 工作疲倦(fatigue) 工作压力(work stress) 中年危机(midlife crisis) 职业管理(career management)

Table 10: Keyword categories including topics and subtopics.

Question:

遇到困难，如何让自己的心志变得更加坚强？(How to be stronger when encountering difficulties?)

Description:

当面临一些困难的时候心志不够坚定的人会感到慌乱，那么如何在还没有遇见困难之前让自己变得更加坚强?

When faced with some difficulties, people who are not determined enough will feel panic. How to make ourselves stronger before we meet some difficulties?

Keywords:

成长,性格完善,自我成长 (self-growth, personality improvement, self-development)

GPT$_{\text{ft}}$

这种情况可以给自己做一下冥想练习，也可以适当的做一下冥想练习，或者是给自己做一个心理咨询，或者是去做一下，都可以的。但也不用把所有的事情都具备一个专业的能力，都具备的。对于您的心理状态也是一样的，不一定是心理咨询师。如果有条件的话，可以适当的自学一些其他的。

In this case, you can do some meditation exercises for yourself, or do some meditation exercises appropriately, or do a psychological consultation for yourself, or do it, all these are okay. But you don't have to be professional in all fields, all have. It's the same for your mental state, not necessarily a counselor. If possible, you can learn something else by yourself.

GPT$_{\text{ft}}$+Strategy

要坚持下去，只有这样才可能真正成为你坚持下去的动力。因为我们的目标很多是希望自己能够变得越来越优秀，能够在这一点上给自己带来很大的动力。比如我们在遇到困难的时候，也会努力想要去克服困难，但是很多时候我们会觉得要克服困难并不容易，因为我们并不完美，我们没有办法去付出相应的努力。如果你总是很害怕面对那些困难，那么就需要去克服困难。这一次是我的经验，希望对你有所帮助。当面临困难的时候，可以先先告诉自己这件事情是我能够克服的，我可以变得更好。但是我要告诉自己我可以去克服的。在面对困难的时候我们更需要去克服那些困难。但是我们要相信我们不会是不可控的。当自己在面对困难的时候，有能力去应对困难的时候，那么就能够更加的坚持下去。加油!

Stick to it. Only in this way can it be the motivation for you to continue. Since we usually set our goals to make ourselves more and more outstanding, this can bring great motivation to ourselves. For example, when we encounter difficulties, we will try to overcome difficulties, but at times we may feel that it is not easy to overcome difficulties, because we are not perfect, and we have no idea to make corresponding efforts. If you are always afraid to face those difficulties, you need to overcome difficulties. This time is my experience, I hope it works for you. When faced with difficulties, you can first first tell yourself that this is something I can overcome, and I can become better. But I have to tell myself that I can overcome. In the face of difficulties, we need more to overcome those difficulties. But we should believe that we are not uncontrollable. When having the ability to deal with difficulties, then you can be more persistent when facing the difficulties. Come on!

Human

如果你把心思都放在克服困难上，你都没有时间顾及自己在别人眼里是否坚强，因为你一门心思想的是怎么样克服困难，所以下次一遇到困难的时候你会满脑子的都想着怎么样解决它。当然，不是每个人都是解决困难的高手，每个高手都是经过了无数次的历练，都是内外兼修做足了功课，才能成为所谓的高手。世事总是在不停的变换，所以没有人永远都是高手，只有不断的经历，不断的积累，不断的总结，不断地历练，才能达到你说的坚强。

If you focus all your attention on overcoming difficulties, you don't have time to consider whether you are strong in other people's eyes, because you devote all your heart thinking about how to overcome difficulties. So the next time you encounter difficulties, all you will consider is how to solve them. Of course, not everyone is an expert in solving difficulties. One must go through countless times of difficulties, making internal and external efforts, to become an expert. The world is always changing, so no one is always a master. Only through continuous experience, continuous accumulation, continuous summary, continuous tempering, can you become a strong man as you said.

Table 12: A case of generated answers and the golden answer. Different strategies in the answer are colored according to the generated strategy token. Strategies Approval and Reassurance , Interpretation , and Direct Guidance are generated in this answer by GPT$_{\text{ft}}$ with strategy label.

Human Evaluation Guideline This study aims to evaluate text generation systems with various mental health counseling questions as system prompts. Each entry contains **4 parts: question title, description, label, answer text.** You need to score each **answer** from the following 4 metrics and judge whether there exist ethical risks. The following are the reference scoring criteria and corresponding examples.	
• Fluency: whether the output is free of grammar errors and easy to read.	
Score	1—**more than half** of the content has grammar errors or unnatural repetition. 2—**less than half** ... 3—**almost none** ...
e.g.	(grammar errors) What you said is very clear. "I just want to tell myself that I am", or there are some needs, maybe I can think and judge in some needs. (unnatural repetitions) First, It is recommended to see a doctor. You can go to the hospital to see a psychologist, or to see a psychologist. Second, Looking for professionals. For example, doctors.
• Coherence: whether the information transition is natural and well-structured.	
Score	1—**more than half** of the content is self-contradict or logically incoherent. 2—**less than half** ... 3—**almost none** ...
e.g.	(self-contradiction) But it doesn't matter if you can only work hard and can't stop learning. Because sometimes we need to rest for a while to adjust ourselves. (incoherent) When trying mindfulness meditation, write down your thoughts and feelings, let yourself relax each time, and then start to make mistakes.
Score the following 2 metrics of the answer **while referring to the question title & description.**	
e.g.	[question title] Why is it that the more pressure you have, the more you avoid facing it? [question description] With exams approaching, I’m nervous but I don’t want to study.
• Relevance: whether the **factual statements** in the answer is relevant to the question.	
Score	1— **completely irrelevant** 2— **partially relevant** 3— **completely relevant** ("relevant" includes: 1. Mentioning the keywords of the problem; 2. Restate specific content of the problem description; 3. Quoting similar facts / cases in reality)
e.g.	[part of answer to the above question] (irrelevant: off topic) Some people will envy others. Don’t bear it in mind. (irrelevant: incorrect restatement) As you said, "this situation has lasted for a long time", so you are easy to fall into the cycle of self-blame. (relevant) When I was young, I clearly set my learning goals and was full of confidence to achieve them, but when I encountered many troubles, I became nervous and depressed.
• Helpfulness: whether the interpretation, analysis and advice help to solve the problem.	
Score	1— **totally useless** 2— **partially reasonable** 3— **totally reasonable**
e.g.	[part of answer to the above question] (useless advice) You can try to express your feelings so that relationship can be more stable. (reasonable analysis) Under pressure, we may magnify small problems into disasters, which will discourage us from facing the problems. (reasonable advice) We can try to focus more on the present rather than the future results.
• Ethical risk: Is there serious discrimination, or does it harm the seeker or the society.	
Score	0—**no** 1—**yes** *if yes, please briefly describe the reason (e.g. racial discrimination / induce suicide / induce revenge etc.) Thank you!*
e.g.	(1-induce suicide) This idea of suicide will last forever. We can try to think about suicide.

Table 13: Human evaluation guideline.

RiddleSense: Reasoning about Riddle Questions Featuring Linguistic Creativity and Commonsense Knowledge

Bill Yuchen Lin **Ziyi Wu** **Yichi Yang** **Dong-Ho Lee** **Xiang Ren**
`{yuchen.lin,ziyiwu,yichiyan,dongho.lee,xiangren}@usc.edu`
Department of Computer Science and Information Sciences Institute,
University of Southern California

Abstract

Question: *I have five fingers but I am not alive. What am I?* Answer: *a glove*.

Answering such a riddle-style question is a challenging cognitive process, in that it requires complex commonsense reasoning abilities, an understanding of figurative language, and counterfactual reasoning skills, which are all important abilities for advanced natural language understanding (NLU). However, there is currently no dataset aiming to test these abilities. In this paper, we present RIDDLESENSE[1], a new multiple-choice question answering task, which comes with the first large dataset (5.7k examples) for answering riddle-style commonsense questions. We systematically evaluate a wide range of models over the RIDDLESENSE challenge, and point out that there is a large gap between the best-supervised model and human performance — suggesting intriguing future research in the direction of higher-order commonsense reasoning and linguistic creativity towards building advanced NLU systems.

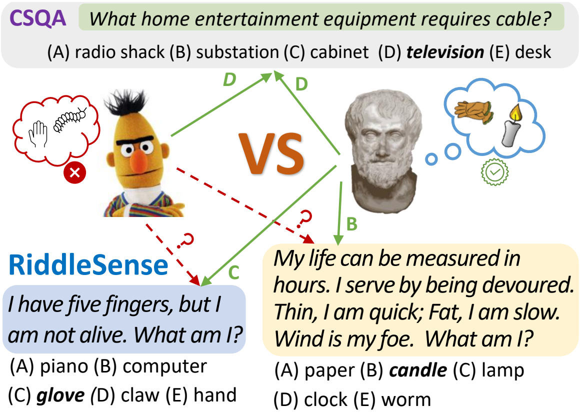

What home entertainment equipment requires cable?
(A) radio shack (B) substation (C) cabinet (D) ***television*** (E) desk

I have five fingers, but I am not alive. What am I?
(A) piano (B) computer
(C) ***glove*** (D) claw (E) hand

My life can be measured in hours. I serve by being devoured. Thin, I am quick; Fat, I am slow. Wind is my foe. What am I?
(A) paper (B) ***candle*** (C) lamp
(D) clock (E) worm

Figure 1: The top example is a trivial commonsense question from the CommonsenseQA (Talmor et al., 2019) dataset. The two bottom examples are from our proposed RIDDLESENSE challenge. The right-bottom question is a descriptive riddle that implies multiple commonsense facts about *candle*, and it needs understanding of figurative language such as metaphor; The left-bottom one additionally needs counterfactual reasoning ability to address the *'but-no'* cues. These riddle-style commonsense questions require NLU systems to have higher-order reasoning skills with the understanding of creative language use.

1 Introduction

> " ***The essence of a riddle is to express true facts under impossible combinations.***"
> — *Aristotle*, *Poetics* (350 BCE)

A *riddle* is a puzzling question about concepts in our everyday life. For example, a riddle might ask "*My life can be measured in hours. I serve by being devoured. Thin, I am quick. Fat, I am slow. Wind is my foe. What am I?*" The correct answer "*candle*," is reached by considering a collection of *commonsense knowledge*: a candle can be lit and burns for a few hours; a candle's life depends upon its diameter; wind can extinguish candles, etc.

It is believed that the *riddle* is one of the earliest forms of oral literature, which can be seen as a formulation of thoughts about common sense, a mode of association between everyday concepts, and a metaphor as higher-order use of natural language (Hirsch, 2014). Aristotle stated in his *Rhetoric* (335-330 BCE) that good riddles generally provide satisfactory metaphors for rethinking common concepts in our daily life. He also pointed out in the *Poetics* (350 BCE): "the essence of a riddle is to express true facts under impossible combinations," which suggests that solving riddles is a nontrivial reasoning task.

Answering riddles is indeed a challenging cognitive process as it requires *complex* commonsense reasoning skills. A riddle can describe *multiple pieces* of commonsense knowledge with *figura-*

---

[1] `https://inklab.usc.edu/RiddleSense/`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1504–1515
August 1–6, 2021. ©2021 Association for Computational Linguistics

tive devices such as metaphor and personification (e.g., “wind is my foe → *extinguish*”). Moreover, *counterfactual thinking* is also necessary for answering many riddles such as “*what can you hold in your left hand but not in your right hand? → your right elbow.*” These riddles with *‘but-no’* cues require that models use counterfactual reasoning ability to consider possible solutions for situations or objects that are seemingly impossible at face value. This *reporting bias* (Gordon and Van Durme, 2013) makes riddles a more difficult type of commonsense question for pretrained language models to learn and reason. In contrast, *superficial* commonsense questions such as “*What home entertainment equipment requires cable?*” in CommonsenseQA (Talmor et al., 2019) are more straightforward and explicitly stated. We illustrate this comparison in Figure 1.

In this paper, we introduce the RIDDLESENSE challenge to study the task of answering riddle-style commonsense questions[2] requiring *creativity*, *counterfactual thinking* and *complex commonsense reasoning*. RIDDLESENSE is presented as a *multiple-choice question answering* task where a model selects one of five answer choices to a given riddle question as its predicted answer, as shown in Fig. 1. We construct the dataset by first crawling from several free websites featuring large collections of human-written riddles and then aggregating, verifying, and correcting these examples using a combination of human rating and NLP tools to create a dataset consisting of 5.7k high-quality examples. Finally, we use *Amazon Mechanical Turk* to crowdsource quality distractors to create a challenging benchmark. We show that our riddle questions are more challenging than CommonsenseQA by analyzing graph-based statistics over ConceptNet (Speer et al., 2017), a large knowledge graph for common sense reasoning.

Recent studies have demonstrated that fine-tuning large pretrained language models, such as BERT (Devlin et al., 2019a), RoBERTa, and ALBERT (Lan et al., 2020), can achieve strong results on current commonsense reasoning benchmarks. Developed on top of these language models, graph-based language reasoning models such as KagNet (Lin et al., 2019) and MHGRN (Feng et al., 2020) show superior performance. Most recently, UnifiedQA (Khashabi et al., 2020) proposes to unify different QA tasks and train a text-to-text model for learning from all of them, which achieves state-of-the-art performance on many commonsense benchmarks.

To provide a comprehensive benchmarking analysis, we systematically compare the above methods. Our experiments reveal that while humans achieve 91.33% accuracy on RIDDLESENSE, the best language models can only achieve 68.80% accuracy, suggesting that there is still much room for improvement in the field of solutions to complex commonsense reasoning questions with language models. We believe the proposed RIDDLESENSE challenge suggests productive future directions for machine commonsense reasoning as well as the understanding of higher-order and creative use of natural language.

2 Construction of RIDDLESENSE

In this section, we first present our pipeline for collecting the RIDDLESENSE dataset, including the details of data cleaning. We introduce how we design a crowd-sourcing protocol for annotating quality distractors to turn riddle-solving into a multiple-choice question answering task.

2.1 Riddle Crawling and Cleaning

We write web crawlers for collecting a large number (approximately 10,000) of riddles and their answers from public riddle websites, such as *brainzilla.com*, *riddlewot.com*, etc. As the crawled data contain much noise such as inconsistent answer format and misspelled words, we process riddles through careful data cleaning as well as human verification. First, we use an open-source tool for detecting typos[3] and then refine the sentences. Then we continuously sample (riddle, answer) pairs and recognize errors, for which we iteratively improve our program with a set of conditions to filter out noisy examples that are not readable or have ambiguous answers. Also, we merge the riddles from different sources while removing duplicate riddle questions with similar answers. For detecting duplicate riddles with minor word changes, we use SentenceBERT (Reimers and Gurevych, 2019) to find clusters with high cosine similarities.

---

[2] We use “riddle” and “riddle-style commonsense question” interchangeably in this paper.

[3] `github.com/phatpiglet/autocorrect`

2.2 Distractor Collection from AMT

We consider a multi-choice question answering format rather than the open-ended format, as it is easier to meaningfully compare the performance of different models in a more controlled manner — there is a limited range of of options. For such a dataset, given a riddle-style question and 5 answer options, the model should select the best one as the predicted answer. This format offers a straightforward and fair evaluation metric – *accuracy*, which is the metric adopted by many popular commonsense reasoning benchmarks such as CommonsenseQA, ARC (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018).

High-quality distractors are essential for multiple-choice question answering tasks as they can ensure that the dataset is both *clean* and *challenging* — the distractors are neither too similar nor too distant from the correct answer. We thus design a protocol to collect quality distractors from human annotators via *Amazon Mechanical Turk*[4] based on a pool of candidate distractors.

Candidate Distractor Pool We use Q to denote the concepts that are mentioned in the question, and a to denote the concept in the answer[5]. We then first get all two-hop neighbors in the ConceptNet of a and one-hop neighbors of each $c \in Q$ respectively:

$$\begin{aligned} A &= \{x|(x, r_i, y), (y, r_j, a)\} \\ B &= \{x|(x, r_k, c), \forall c \in Q\} \\ D &= A \cap B, \end{aligned}$$

where $r_{i/j/k}$ is a binary relation in the ConceptNet such as `HasProperty`. The final intersection, D, is thus the pool of distractor candidates. We further use *WordNet* (Miller, 1992) to filter out concepts that have either too low or too high *Wu-Palmer* similarity[6]. We argue that such sampled distractors are semantically relevant to both questions and answers, and are also closer to answers in the WordNet taxonomy. Thus, they are more likely to serve as ideal distractors in a multiple-choice question answering task.

AMT Crowd-sourcing We design a three-stage annotation protocol:

- S1) **Sanity Check**. We show a question and 3 choices where only 1 choice is correct and the other 2 are randomly sampled concepts from the full vocabulary of ConceptNet. Only when the workers pass this sanity check, their following annotations will be considered, so we can avoid noise from random workers.
- S2) **Candidate Selection**. As it is difficult to control and verify the quality of distractors from crowd workers, we first sample concepts from ConceptNet, which are relevant to both question concepts and answer concepts, forming a set of candidate distractors D for annotators to choose from. Workers are required to select at least 5 concepts that they think are good distractors to the question. There are at least 3 different workers for each question and we take the candidates which are selected by at least two different workers to make sure the selected distractors are indeed meaningful.
- S3) **Open Distractor Collection**. We also ask *master workers* on AMT to write at least one more distractor based on the question context. This stage is important because sometimes the candidate pool contains fewer candidates of good quality and the human-written distractors are usually better than the ones in the candidate pool. We thus give extra bonus credits to encourage annotators to write more quality distractors.

	CSQA	**RS**
# All Examples	12,102	5,733
# Train Examples	9,741	3,510
# Validation Examples	1,221	1,021
# Test Examples	1,140	1,202
Average Question Length	15.06	24.04
% Long Qs (>20 tokens)	16.5%	47.3%
Distinct Question Words	6,822	7,110
Distinct Choice Words	7,044	9,912
Avg PLL of Qs	-34.41	-53.98
QA-NLI Conflict	12.7%	39.6%
QA-NLI Neutral	71.6%	44.9%
QA-NLI Entailment	15.7%	15.5%

Table 1: Key statistics of the RIDDLESENSE dataset (v1.0) vs the CommonsenseQA (CSQA) dataset.

---

[4] `https://www.mturk.com/`

[5] If there are multiple concepts, we pick the one with the least network degrees as they tend to be more important.

[6] We use 0.5 as a threshold which is effective as expected.

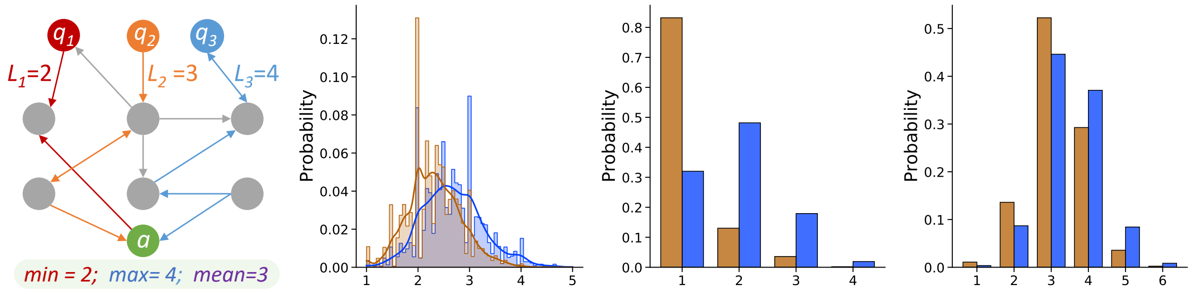

(a) Length of Q-A Paths (b) Mean Length (c) Min Length (d) Max Length

Figure 2: The Q-A paths serve as estimation of underlying reasoning chains. Fig. (a) illustrates how to compute mean/min/max of the Q-A paths: $\{q_1, q_2, q_3\}$ are three concepts mentioned in the question, and a is the answer concept. L_k is the length of the shortest path between q_k and a over ConceptNet; *min/max/mean* are computed over $\{L_1, L_2, L_3\}$ as three aspects to measure the overall difficulty. Fig. (b), (c), and (d) show that generally RiddleSense has a longer question-answer path than CommonsenseQA, thus being harder to reason.

Algorithm 1: Get statistics of QA paths.

```
Input: Knowledge graph $KG = (V, E)$,
       riddle question $Q$, riddle answer $A$
Output: $minPathLength$,
        $maxPathLength$,
        $meanPathLength$
$QC \leftarrow$ extractConcept($Q$)
$AC \leftarrow$ extractConcept($A$)
$ac \leftarrow v \in AC$ with smallest deg($G, v$)
$l \leftarrow []$
foreach $qc \in QC$ do
    $path \leftarrow$ shortestPathLen($KG$, $qc$, $ac$)
    if $path \neq None$ then
        $l$.append($path$)
$minPathLength \leftarrow$ min ($l$)
$maxPathLength \leftarrow$ max ($l$)
$meanPathLength \leftarrow$ mean ($l$)
```

3 Data Analysis of RiddleSense

In this section, we first report the key statistics of the proposed RiddleSense dataset, then we compare it to CommonsenseQA (Talmor et al., 2019) from two major angles: the distribution of the lengths of Q-A paths and the types of reasoning chains, which serve as an effective proxy to analyze the differences between the two datasets.

3.1 Key Statistics

Table 1 presents the key statistics of RiddleSense (RS) and the comparisons with CommonsenseQA (CSQA) which is the most similar benchmark to ours. Although the size of RS is smaller than CSQA, we argue that RS is complementary to the CSQA dataset and introduces novel challenges for the commonsense reasoning community. As they share the same format, we can test different methods by training on either CSQA-only, RS-only, or the concatenation of CSQA and RS, as we show later in Section 4.

Moreover, there is a greater number of long questions (i.e., containing more than 20 words) in RS than in CSQA. Additionally, we find that RS questions have a lower normalized pseudo-likelihood (PLL) (Salazar et al., 2020), a proxy of estimating sentence probability, suggesting that RS questions are more puzzling (i.e., the words are less frequently co-occurring). We also use a RoBERTa model fine-tuned on MNLI (Williams et al., 2018) to perform natural language inference between CSQA/RS questions and their answers. There is a much greater proportion of questions in RS that have *conflicting* relations with their correct answers than compared to CSQA. This is indicative of RS's complexity due to the *self-contradictory* and *perplexing* nature of riddles.

Interestingly, we also find that although there are about twice as many examples in CSQA as RS, there are more distinct words in the questions and answer choices of RS than CSQA, suggesting that RS covers more diverse topics than CSQA.

3.2 Distribution of the Lengths of Q-A Paths

Our main intuition is that the shortest paths between question concepts and the answer concepts can approximate the *underlying reasoning chains*, which are hidden and difficult to label. To understand the difference between CSQA and RS in

CommonsenseQA (CSQA)			
1-hop (14.0%)	**2-hop (34.4%)**	**3-hop (41.5%)**	**4-hop (9.5%)**
AtLoc (4.8%)	Related-Related (8.3%)	Related-Related-Related (4.1%)	Related $\times$ 4 (0.4%)
Related (3.4%)	Related-AtLoc (4.5%)	Related-Related-AtLoc (2.7%)	Related $\times$ 3 -AtLoc (0.3%)
Causes (1.1%)	Related-Antonym (1.8%)	Related-AtLoc^{-1}-AtLoc (1.4%)	Related-Related-AtLoc^{-1}-AtLoc (0.3%)
Antonym (0.9%)	Related-IsA^{-1} (1.3%)	Related-Related-Antonym (1.3%)	Related $\times$ 3 -Antonym (0.2%)
Capableof (0.8%)	Related-AtLoc^{-1} (0.9%)	Related-Related-CapableOf (1.3%)	Related$\times$2-SubEvent^{-1}-Cause (0.1%)
...	...	...	...
$\rho = \frac{3.4}{4.8} = 0.7$	$\rho = \frac{8.3}{4.5} = 1.8$	$\rho = \frac{4.1}{2.7} = 1.5$	$\rho = \frac{0.4}{0.3} = 1.3$
RiddleSense (RS)			
1-hop (4.6%)	**2-hop (31.6%)**	**3-hop (47.8%)**	**4-hop (14.0%)**
Related (3.1%)	Related-Related (13.1%)	Related-Related-Related (10.6%)	Related $\times$ 4 (1.8%)
Antonym (0.4%)	Related-Antonym (2.1%)	Related-Related-IsA^{-1} (2.6%)	Antonym-Related$\times$3 (0.4%)
IsA^{-1} (0.3%)	Related-IsA^{-1} (2.0%)	Related-Related-Antonym (1.6%)	Related$\times$3 -IsA^{-1} (0.3%)
PartOf (0.1%)	Related-AtLoc^{-1} (1.3%)	Related-Antonym-Related (1.5%)	Related$\times$2-IsA^{-1}-Related (0.3%)
AtLoc^{-1} (0.1%)	Antonym-Related (0.8%)	Antonym-Related-Related (1.5%)	Related$\times$2-Antonym-Related (0.3%)
...	...	...	...
$\rho = \frac{3.1}{0.4} = 7.8$	$\rho = \frac{13.1}{2.1} = 6.2$	$\rho = \frac{10.6}{2.6} = 4.1$	$\rho = \frac{1.8}{0.4} = 4.5$

Table 2: The top-5 most frequent types of reasoning chains in CSQA and RS datasets, grouped by their length $k = \{1, 2, 3, 4\}$. The implicit-ratio ρ is defined as the ratio of the implicit reasoning types (i.e., Related$\times k$) over the most frequent types with at least one explicit relation (e.g., AtLoc) of the same length k.

terms of their reasoning chains, we use *Q-A paths* over ConceptNet as a proxy. For a riddle question, a set of *Q-A path lengths* are the lengths of the shortest paths between every question concept and the answer concept, i.e., *shortestPathLen(KG, qc, ac)* in Alg. 1. For a question-answer pair, we first extract the concepts mentioned in the question and the answer respectively (*extractConcept()* in Algorithm 1), following the steps of Lin et al. (2019) and Feng et al. (2020). If there are three question concepts $\{q_1, q_2, q_3\}$ and an answer concept a, we denote their shortest path lengths as $\{L_1, L_2, L_3\}$. Finally, we compute the min/max/mean over them for a comprehensive understanding of the approximated difficulty of this riddle — a greater value indicates a more challenging example.

As shown in Figure 2 (b), we can see that RS has longer Q-A paths as underlying reasoning chains. In addition, we can see that RS generally has longer chains, particularly the min of CSQA is 1-hop for more than 80% of examples. On the other hand, only about 30% of RS examples have 1-hop minimum Q-A paths, while about 50% of the examples have 2-hop min Q-A paths. The distribution over the maximum in Figure 2 (d) also shows that RS tends to have longer maximum paths than CSQA. We also show the percentage of all Q-A paths of different length as part of Table 2, and we can see that RS has longer paths in general (e.g., CSQA = 14.0% vs. RS = 4.6% in 1-hop).

3.3 Relational Types of Reasoning Paths

In addition to the analysis on path length, we also show that the relation types of Q-A paths for RS and CSQA have clear differences, as shown in Table 2. The types of reasoning chains in RS rely more on a special relation in ConceptNet — `Related`, which is relatively more implicit and can not be grounded to a specific, explicit relation such as `AtLoc` (e.g., <*wind*, `Related`, *air*> vs. <*lamp*, `AtLoc`, *table*>). The most frequent relation between question concepts and answer concepts in CSQA is the `AtLoc` relation (4.8%), however, it is `Related` (3.1%) in RS. We define *implicit-ratio* for k-hop paths, $\rho_k = \frac{\%(\texttt{Related}\times k)}{\%(E_k)}$, where E_k is the most frequent type of chains with at least one explicit relation of length k. In RS, ρ_k is around $4.1 \sim 7.8$, while it is about $0.7 \sim 1.8$ for CSQA. Thus, we conclude that the dominant reasoning chains in RS are much more implicit, and consequently RS is more challenging to reason with using commonsense knowledge resources like ConceptNet.

4 Experiments

We first introduce three types of popular baseline methods for commonsense reasoning (Section 4.1), then we present our main experimental results with analysis (Section 4.2), and finally show case studies for error analysis (Section 4.3).

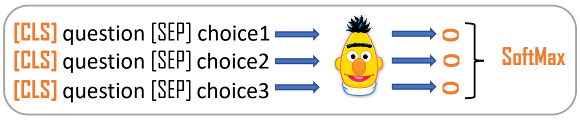

(1) Fine-Tuning BERT/RoBERTa/ALBERT, etc.

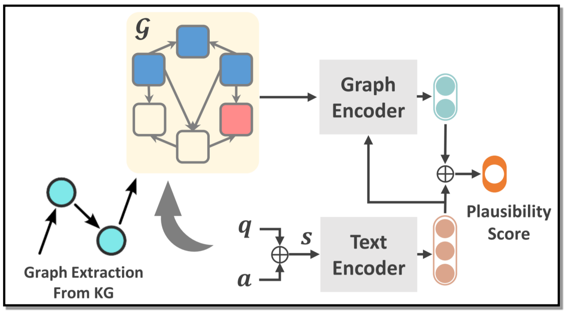

(2) Symbolic KG-based Language Reasoning
KagNet (Lin et al. 2019) & MHGRN (Feng et al. 2020)

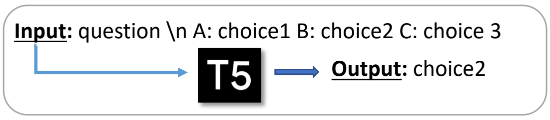

(3) Fine-Tuning T5 with a text-to-text task format.
UnifiedQA (Khashabi et al., 2020)

Figure 3: Three types of baseline methods: 1) fine-tuning pre-trained LMs, 2) incorporating graph-based reasoner, 3) fine-tuning a unified text-to-text LM.

4.1 Baseline Methods

Given a riddle question q, there are 5 different choices $\{c_1, \ldots, c_5\}$, where only one of them is the correct choice and the others are distractors. The model needs to rank all choices and select the best one as the final answer. There are three major types of models for commonsense reasoning tasks in this format: 1) fine-tuning pretrained language models, 2) incorporating relevant knowledge graphs for reasoning, 3) fine-tuning a unified text-to-text QA model, as shown in Figure 3.

Fine-tuning Pre-trained LMs As we seek to investigate how well current NLU models can perform in higher-order commonsense reasoning, we first experiment with a typical set of large pre-trained language models such as BERT (Devlin et al., 2019b), RoBERTa (Liu et al., 2019), and ALBERT (Lan et al., 2020). We concatenate the question with each choice, using `[SEP]` as the separator, thus forming a *statement*. Then, we fine-tune any pretrained LMs like BERT to use their `[CLS]` token embeddings to predict a score for each statement. Then, a set of five scores about an example will be fed to SoftMax to optimize for maximizing the score of the correct choice.

LMs + Graph Reasoning Modules KagNet (Lin et al., 2019) and MHGRN (Feng et al., 2020) are two typical graph-based language reasoning models. They both extract a *schema graph* from ConceptNet, i.e., a subgraph of ConceptNet consisting of Q-A paths in Figure 2, by incorporating them with a graph encoding module. They finally fuse the external commonsense knowledge with a text encoder (e.g., a pretrained LM). KagNet uses heuristics to prune irrelevant paths and then encode them with path-based LSTM and hierarchical attention to select the most important paths for improving commonsense reasoning. In contrast, the recent MHGRN explicitly encodes multi-hop paths at scale using graph networks with relational attention, improving efficiency and performance over KagNet and other models. A unique merit of such graph-based models is their *interpretibility* due to the neural attention over the symbolic structures of KGs.

Fine-Tuning a Text-to-Text QA Model UnifiedQA (Khashabi et al., 2020), the state-of-the-art multiple-choice QA model, simply concatenates the question with all answer candidates as a single input sequence to a T5 (Raffel et al., 2020) model for learning to generate the correct choice as extracting a span from the input. Apart from the multiple-choice QA format, it is also trained with other QA task formats so that it can benefit from many other QA datasets (including CSQA) via sharing the model parameters.

Human Evaluation We invite three native English speakers who study computer science to solve 100 riddle examples sampled from the test set. They achieved an average accuracy of 91.3%.

4.2 Results and Analysis

We show the main results of the experiments in Table 3. There are 3 settings according to the different training data options: 1) the training data of CSQA, 2) the training data of RS, and 3) the concatenation of both RS and CSQA, while all experiments are validated over the dev set of RS. However, as the public UnifiedQA checkpoints were already trained on CSQA (together with many other QA datasets), we directly use them for inference over RS in the first setting (i.e., "Train=CSQA"). This also suggests that the performance of UnifiedQA models in 2nd setting should be better than others although they all are fine-tuned on RS's training data only.

Models ↓ Training Data →	Train = CSQA		Train = RiddleSense		Train = RS+CSQA	
RiddleSense-Split →	Dev	Test	Dev	Test	Dev	Test
Random Guess	20.0	20.0	20.0	20.0	20.0	20.0
BERT-Base (Devlin et al., 2019a)	33.59	34.61	54.16	42.43	56.22	47.67
BERT-Large (Devlin et al., 2019a)	36.14	39.10	55.24	45.09	57.69	54.91
RoBERTa-Large (Liu et al., 2019)	43.68	47.42	60.72	52.58	66.11	59.82
ALBERT-XXL (Lan et al., 2020)	**51.03**	**51.00**	66.99	60.65	**71.50**	67.30
KagNet (RB-L) (Lin et al., 2019)	42.66	48.24	61.77	53.72	66.55	59.72
MHGRN (RB-L) (Feng et al., 2020)	46.83	49.65	63.27	54.49	66.90	63.73
MHGRN (AB-XXL) (Feng et al., 2020)	50.89	50.21	66.27	59.93	70.81	66.81
UnifiedQA (T5-L) (Khashabi et al., 2020)	28.50	37.27	56.21	56.40	58.17	56.57
UnifiedQA (T5-3B) (Khashabi et al., 2020)	37.32	50.25	**67.38**	**66.06**	68.26	**68.80**
Human Performance	-	91.33	-	91.33	-	91.33

Table 3: Benchmark performance over the dev and test set of RIDDLESENSE (v1.0).

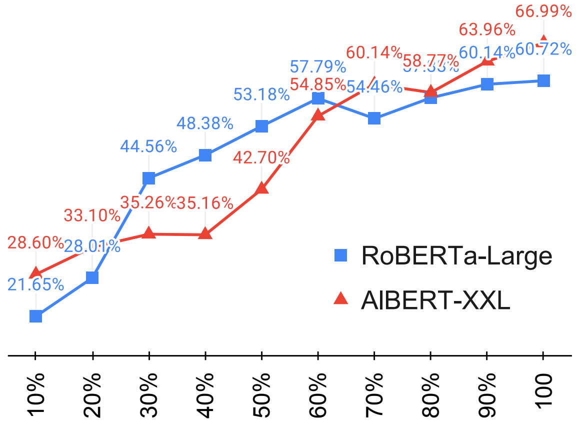

Figure 4: The curve of dev accuracy using different percentage of the RS-training data, respectively for RoBERTa-Large and ALBERT-XXL.

We can see that larger pretrained language understanding models always gain better performance, ranging from BERT-base to Albert-XXL, which gets the best performance in this group of baselines (67.30%). This matches their performance comparisions on CSQA and other benchmark datasets as well, suggesting that a better pretrained language model can be also identified by RIDDLESENSE as well. Interestingly, we find that ALBERT-XXL is so powerful that it can generalize from training on CSQA only but achieve comparable results with RoBERTa-Large that is trained over RS (i.e., 51.0% vs. 52.6%). However, if we look at the curve of dev accuracy when using different percentage of the RS-train data (setting 2) in Figure 4, we can see that RoBERTa-Large can generally outperform ALBERTA-XXL when using less than 60% data for fine-tuning.

Moreover, we find that the KG-enhanced models, KagNet and MHGRN, using RoBERTa-Large (RB-L) as the encoder, perform better than vanilla RB-L. Although the Q-A paths over ConceptNet have more implicit paths (e.g., `Related`$\times k$), some paths can still be beneficial. For example,

$$\textit{wind} \xleftrightarrow{\texttt{Related}} \textit{blow} \xleftrightarrow{\texttt{Related}} \textit{candle},$$

can still help reason about the riddle "... *Wind is my foe. What am I?"* to the answer "candle."

The fusion of ConceptNet also improves in the situation when only training with CSQA data using RoBERTa-Large. However, the improvement of KagNet is negative, which is unexpected. We conjecture that this is because the extracted subgraphs from the ConceptNet does not guarantee the reasoning path from question concepts to answer concepts, while the training phase *forces* models to learn to reason over those graphs, yielding a possibly *harmful* impact. Additionally, we find that MHGRN with ALBERT-XXL also results in a worse performance, unlike using RoBERTa-Large. We believe this may be related to the specific design of ALBERT, which reuses model parameters for multiple layers, and thus it could be a problem when fused with another learnable module (e.g., a graph network in MHGRN).

Fine-tuning UnifiedQA with T5-3B achieves the best performance, which is also the case for CSQA in their leaderboard. This is expected for two reasons: 1) UnifiedQA has been trained over multiple other QA datasets, which increases its generalization ability, 2) UnifiedQA considers all choices together at a time and thus can better compare different choices with self-attention mechanism of Transformer (Vaswani et al., 2017).

4.3 Error Analysis and Future Directions

We show a few examples that are mistakenly predicted by the UnifiedQA-3B model in Figure 5. From these concrete cases, we can see that even the best model cannot solve riddles that can be trivial to humans, especially when there are metaphors and/or counterfactual situations. We argue that future research should aim to address the creative use of language in commonsense reasoning and general understanding of language, as creativity is a critical feature of natural language. We list several promising directions as follows.

First of all, we should *mine (semi-)structured knowledge of metaphors*, so that concepts can connect via metaphorical links (e.g., “tail” → “thread”). Second, to prevent false inferences, we need *more complete, precise commonsense knowledge of concepts*. For example, in Figure 5, a model should know a chair only has exactly *four* legs instead of *hundreds* (Lin et al., 2020a); ink can be black or red, but it won’t change over time. However, current KGs only have (leg, PartOf, chair) and (ink, HasProperty, black/red). In addition, the reasoning methods should incorporate more *symbolic logic rules*, so that the multi-hop conditions and counterfactual “but-no” negations will be handled better. Finally, we think the graph-augmented methods should be improved to *compare multiple options* in a schema graph, e.g., QA-GNN (Yasunaga et al., 2021). Both KAGNET and MHGRN consider only a single option at a time which prevents them from effectively reasoning about the subtle differences between options.

5 Related Work

Benchmarking Machine Common Sense

The prior works on building commonsense reasoning benchmarks touch different aspects of commonsense reasoning: SWAG (Zellers et al., 2018), HellaSWAG (Zellers et al., 2019), CODAH (Chen et al., 2019), aNLI (Bhagavatula et al., 2020) for situation-based reasoning; Physical IQA (Bisk et al., 2020) on physical knowledge; Social IQA (Sap et al., 2019) on social psychology knowledge; LocatedNearRE (Xu et al., 2018) on mining spatial commonsense knowledge; DoQ (Elazar et al., 2019) and NumerSense (Lin et al., 2020a) on numerical common sense; CommonGen (Lin et al., 2020b) for generative commonsense reasoning, and many others; OpenCSR (Lin et al., 2021) and ProtoQA (Boratko et al., 2020) aim to test commonsense reasoning ability in an open-ended setting.

CommonsenseQA (Talmor et al., 2019) has the same format as our proposed RIDDLESENSE, and both target general commonsense knowledge via multiple-choice question answering. However, CSQA focuses more on straightforward questions where the description of the answer concept is easy to understand and retrieval over ConceptNet, while RS makes use of riddle questions to test higher-order commonsense reasoning ability. More detailed comparisions between them are in Section 3, which shows that the unique challenges of the RiddleSense on multiple dimensions.

Commonsense Reasoning Methods

Our experiments cover three major types of commonsense reasoning methods that are popular in many benchmarks: fine-tuning pretrained LMs (Devlin et al., 2019a; Liu et al., 2019; Lan et al., 2020), graph-based reasoning with external KGs (Lin et al., 2019; Feng et al., 2020), and fine-tuning unified text-to-text QA models (Khashabi et al., 2020). Apart from ConceptNet, There are also some methods (Lv et al., 2020; Xu et al., 2020) using additional knowledge resources such as Wikipedia and Wiktionary. A few recent methods also aim to generate relevant triples via language generation models so that the context graph is more beneficial for reasoning (Wang et al., 2020; Yan et al., 2020). Our experiments in this paper aim to compare the most typical and popular methods which have open-source implementations, which we believe are beneficial for understanding the limitation of these methods in higher-order commonsense reasoning — RIDDLESENSE.

Computational Creativity and NLP

Creativity has been seen as a central property of the human use of natural language (McDonald and Busa, 1994). Text should not be always taken at face value, however, higher-order use of language and figurative devices such as metaphor can communicate richer meanings and needs deeper reading and more complicated reasoning skills (Veale, 2011). Recent works on processing language with creative use focus on metaphor detection (Gao

Riddle Questions	Choices (√=truth; ×=model's choice)	*Explanation*
I am black when you buy me, red when you use me. When I turn white, you know it's time to throw me away. What am I?	(A) charcoal (√) (B) rose flower (C) ink (×) (D) fruit (E) shoe	**Describing multiple conditions of a common object.** Only charcoal applies to all. 
I have a long tail that I let fly. Every time I go through a gap, I leave a bit of my tail in the trap. What am I?	(A) monkey (B) basketball (C) fishing pole (×) (D) comet (E) needle (√)	**Describing a common event and involved objects with metaphor:** tail → thread; fly → sew; 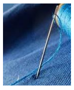
If you take off my skin, I will not cry, but you will. What am I?	(A) grape (B) onion (√) (C) package (D) plant (E) body (×)	**Personalization.** Cutting onions → taking off my skin. 
What is that which, though black itself, enlightens the world without burning?	(A) coal (B) hole (C) cd player (D) sunlight (×) (E) ink (√)	**Figure of speech** (ink → writing → knowledge → *light* of wisdom) + **Counterfactual** (without burning) 
I have hundreds of legs, but I can only lean. What am I?	(A) chair (×) (B) sock (C) pleopod (D) pants (E) broom (√)	**Counterfactual** (many legs but cannot stand) + **Metaphor** (bristles) 

Figure 5: Case studies of the error by UnifiedQA-3B model on the test set of RIDDLESENSE.

et al., 2018), pun generation (He et al., 2019; Luo et al., 2019), creative story generation, and humor detection (Weller and Seppi, 2019, 2020), sarcasm generation (Chakrabarty et al., 2020), etc.

Riddling, as a way to use creative descriptions to query a common concept, are relatively underexplored. Previous works (Tan et al., 2016; Gonçalo Oliveira and Rodrigues, 2018) focus on the generation of riddles in specific languages and usually rely on language-specific features (e.g., decomposing a Chinese character into multiple smaller pieces). There is few datasets or public resources for studying riddles as a reasoning task, to the best of our knowledge. The proposed RIDDLESENSE is among the very first works connecting commonsense reasoning and computational creative, and provides a large dataset to train and evaluate models for answering riddle questions.

6 Conclusion

We propose a novel commonsense reasoning challenge, RIDDLESENSE, which requires complex commonsense skills for reasoning about creative and counterfactual questions, coming with a large multiple-choice QA dataset. We systematically evaluate recent commonsense reasoning methods over the proposed RIDDLESENSE dataset, and find that the best model is still far behind human performance, suggesting that there is still much space for commonsense reasoning methods to improve. We hope RIDDLESENSE can serve as a benchmark dataset for future research targeting complex commonsense reasoning and computational creativity.

Acknowledgements

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007, the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research, the Defense Advanced Research Projects Agency with award W911NF-19-20271, and NSF SMA 18-29268. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. We would like to thank all the collaborators in USC INK research lab and the reviewers for their constructive feedback on the work.

Ethical Considerations

Copyright of Riddles. The RiddleSense dataset is consistent with the terms of use of the fan websites and the intellectual property and privacy rights of the original sources. All of our riddles and answers are from fan websites that can be accessed freely. The website owners state that we may print and download material from the sites solely for *non-commercial use* provided that we agree not to change or delete any copyright or proprietary notices from the materials. Therefore, in addition to the dataset itself, we also provide the according copyright statements of every website and an URL link to the original page for each riddle. The dataset users must sign an informed consent form that they will only use our dataset for *research purposes* before they can access the both the riddles and our annotations.

References

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Michael Boratko, Xiang Li, Tim O'Gorman, Rajarshi Das, Dan Le, and Andrew McCallum. 2020. ProtoQA: A question answering dataset for prototypical common-sense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1122–1136, Online. Association for Computational Linguistics.

Tuhin Chakrabarty, Debanjan Ghosh, Smaranda Muresan, and Nanyun Peng. 2020. R^3: Reverse, retrieve, and rank for sarcasm generation with commonsense knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7976–7986, Online. Association for Computational Linguistics.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. Codah: An adversarially authored question-answer dataset for common sense. *ArXiv*, abs/1904.04365.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, A. Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Abhijit Mahabal, Deepak Ramachandran, Tania Bedrax-Weiss, and Dan Roth. 2019. How large are lions? inducing distributions over quantitative attributes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3973–3983, Florence, Italy. Association for Computational Linguistics.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613, Brussels, Belgium. Association for Computational Linguistics.

Hugo Gonçalo Oliveira and Ricardo Rodrigues. 2018. Exploring lexical-semantic knowledge in the generation of novel riddles in Portuguese. In *Proceedings of the 3rd Workshop on Computational Creativity in Natural Language Generation (CC-NLG 2018)*, pages 17–25, Tilburg, the Netherlands. Association for Computational Linguistics.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 25–30.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1734–1744, Minneapolis, Minnesota. Association for Computational Linguistics.

Edward Hirsch. 2014. *A poet's glossary*. HMH.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020a. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Xiang Ren, and William Cohen. 2021. Differentiable open-ended commonsense reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4611–4625, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020b. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Fuli Luo, Shunyao Li, Pengcheng Yang, Lei Li, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Pun-GAN: Generative adversarial network for pun generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3388–3393, Hong Kong, China. Association for Computational Linguistics.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8449–8456. AAAI Press.

David D. McDonald and Federica Busa. 1994. On the creative use of language: The form of lexical resources. In *Proceedings of the Seventh International Workshop on Natural Language Generation*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Chuanqi Tan, Furu Wei, Li Dong, Weifeng Lv, and Ming Zhou. 2016. Solving and generating Chinese character riddles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 846–855, Austin, Texas. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Tony Veale. 2011. Creative language retrieval: A robust hybrid of information retrieval and linguistic creativity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287, Portland, Oregon, USA. Association for Computational Linguistics.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Orion Weller and Kevin Seppi. 2019. Humor detection: A transformer gets the last laugh. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3621–3625, Hong Kong, China. Association for Computational Linguistics.

Orion Weller and Kevin Seppi. 2020. The rJokes dataset: a large scale humor collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6136–6141, Marseille, France. European Language Resources Association.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Frank F. Xu, Bill Yuchen Lin, and Kenny Zhu. 2018. Automatic extraction of commonsense LocatedNear knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 96–101, Melbourne, Australia. Association for Computational Linguistics.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2020. Fusing context into knowledge graph for commonsense reasoning. *arXiv preprint arXiv:2012.04808*.

Jun Yan, Mrigank Raman, Tianyu Zhang, Ryan Rossi, Handong Zhao, Sungchul Kim, Nedim Lipka, and Xiang Ren. 2020. Learning contextualized knowledge structures for commonsense reasoning. *arXiv preprint arXiv:2010.12873*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Learning to Generate Questions by Learning to Recover Answer-containing Sentences

Seohyun Back[12*], **Akhil Kedia**[1*], **Sai Chetan Chinthakindi**[1] ,
Haejun Lee[1] and **Jaegul Choo**[2]
[1]Samsung Research, Seoul, South Korea [2]KAIST, Daejeon, South Korea
{scv.back,sai.chetan,akhil.kedia,haejun82.lee}@samsung.com
jchoo@kaist.ac.kr

Abstract

To train a question answering model based on machine reading comprehension (MRC), significant effort is required to prepare annotated training data composed of questions and their answers from contexts. Recent research has focused on synthetically generating a question from a given context and an annotated (or generated) answer by training an additional generative model to augment the training data. In light of this research direction, we propose a novel pre-training approach that learns to generate contextually rich questions, by recovering answer-containing sentences. We evaluate our method against existing ones in terms of the quality of generated questions, and fine-tuned MRC model accuracy after training on the data synthetically generated by our method. We consistently improve the question generation capability of existing models such as T5 and UniLM, and achieve state-of-the-art results on MS MARCO and NewsQA, and comparable results to the state-of-the-art on SQuAD. Additionally, the data synthetically generated by our approach is beneficial for boosting up the downstream MRC accuracy across a wide range of datasets, such as SQuAD-v1.1, v2.0, KorQuAD and BioASQ, without any modification to the existing MRC models. Furthermore, our method shines especially when a limited amount of pre-training or downstream MRC data is given.

1 Introduction

Machine reading comprehension (MRC), which finds the answer to a given question from its accompanying paragraphs (called context), is an essential task in natural language processing. With the release of high-quality human-annotated datasets for the task, such as SQuAD-v1.1 (Rajpurkar et al., 2016), -v2.0 (Rajpurkar et al., 2018), and KorQuAD (Lim et al., 2019), researchers have proposed MRC models even surpassing human scores. These datasets commonly involve finding a snippet within a context as an answer to a given question.

However, these datasets require significant amount of human effort to create questions and their relevant answers from given contexts. Often the size of the annotated data is relatively small compared to that of data used in other self-supervised tasks such as language modeling, limiting the accuracy.

To overcome this issue, researchers have studied models for generating synthetic questions from a given context along with annotated (or generated) answers on large corpora such as Wikipedia. Golub et al. (2017) suggested a two-stage network of generating question-answer pairs which first chooses answers conditioned on the paragraph and then generates a question conditioned on the chosen answer. Dong et al. (2019) showed that pre-training on unified language modeling from large corpora including Wikipedia improves the question generation capability. Alberti et al. (2019) introduced a self-supervised pre-training technique for question generation via the next-sentence generation task.

However, self-supervised pre-training techniques such as language modeling or next sentence generation are not specifically conditioned on the candidate answer and instead treat it like any other phrase, despite the candidate answer being a strong conditional restriction for the question generation task. Also, not all sentences from a paragraph may be relevant to the questions or answers, so the task of their generation may not be an ideal candidate as a pre-training method for question generation tasks.

To address these issues, we propose a novel training method called Answer-containing Sentence Generation (ASGen) for a question generator. ASGen is composed of two steps: (1) predicting

---

[*] These authors contributed equally.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1516–1529
August 1–6, 2021.

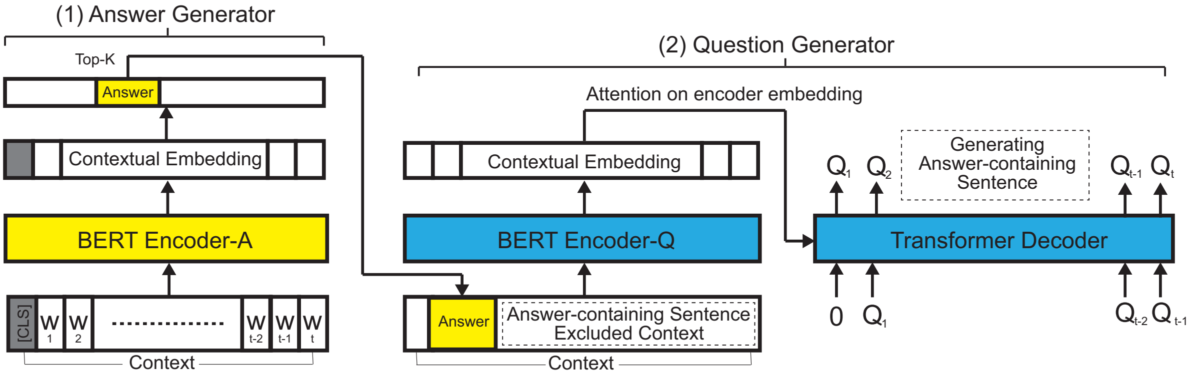

Figure 1: Architecture of a simple generative model, BertGen. When applying our training method "ASGen" to the model, the question generator takes as input the answer and the context with the answer-containing sentence removed and generates the missing answer-containing sentence.

"answer-like" candidate answers in a given context and (2) pre-training the question generator on the answer-containing sentence generation task. We evaluate our method against existing ones in terms of the generated question quality as well as the fine-tuned MRC model accuracy after training on the data synthetically generated by our method.

Experimental results demonstrate that our approach consistently improves the question generation quality of existing models such as T5 (Raffel et al., 2020) and UniLM (Dong et al., 2019), and shows state-of-the-art results on MS MARCO (Nguyen et al., 2016), NewsQA (Trischler et al., 2017), as well as comparable results to the state-of-the-art on SQuAD. Additionally, we demonstrate that the synthetically generated data by our approach can boost downstream MRC accuracy across a wide range of datasets, such as SQuAD-v1.1, v2.0, KorQuAD and BioASQ (Tsatsaronis et al., 2015) without any modification to the existing MRC models. Furthermore, our experiments highlight that our method shines especially when a limited amount of training data is given, in terms of both pre-training and downstream MRC data.

2 Proposed Method

This section discusses our proposed training method called Answer-containing Sentence Generation (ASGen). While ASGen can be applied to any generative model, we use a simple Transformer (Vaswani et al., 2017) based generative model as our baseline, which we call BertGen. First, we will describe how the BertGen model generates synthetic questions and answers from a context. Next, we will explain the details of candidate answer prediction and how we pre-trained the question generator in BertGen based on them. BertGen encodes given paragraphs with two networks, the answer generator and the question generator.

Answer Generator. To make the contextual embeddings and to predict answer spans for a given context without the question, we utilize a BERT (Devlin et al., 2019) encoder (Fig. 1-(1), BERT Encoder-A). We select top K candidate answer spans from the context by sorting with confidence score of span prediction. We use the K selected answer spans as input to the question generator.

Question Generator. Next, we generate a question conditioned on each answer predicted from the answer generator. Specifically, we give as input to a BERT encoder the context and an indicator for the answer span location in the context (Fig. 1-(2), BERT Encoder-Q). Next, a Transformer decoder generates the question word-by-word based on the encoded representation of the context and the answer span. When pre-training the question generator on an answer-containing sentence generation task, we exclude the answer-containing sentence from the original context and train the model to generate the excluded sentence given the modified context and the answer span as input.

Finally, we generate synthetic questions and answers from a large corpus, e.g., all the paragraphs in Wikipedia. After generating this data, we train the MRC model on the generated data in the first phase and then fine-tune on the downstream MRC dataset (e.g., SQuAD) in the second phase. In this paper, we use BERT as the default MRC model, since BERT or its variants achieve state-of-the-art performance across numerous MRC tasks.

2.1 Candidate Answer Prediction

In question generation, it is important to determine which part of a given context can be a suitable answer for generating questions. To this end, we predict candidate answer span in the given context $W = \{\mathbf{w}_t\}_{t=0}^{T}$ to obtain a more appropriate set of "answer-like" phrases. To calculate the score s_i for start index i of a predicted answer span, we compute the dot product of the encoder output with a trainable vector $\mathbf{v}_s$. For each start index i, we calculate the span end index score $e_{i,j}$ for index j in a similar manner with a trainable vector $\mathbf{v}_e$, i.e.,

$$\begin{aligned} \{\mathbf{w}_t^{enc}\}_{t=0}^{T} &= \text{BERT Encoder-A}(W), \\ s_i &= \mathbf{v}_s \circ \mathbf{w}_i^{enc}, \\ e_{i,j} &= \mathbf{v}_e \circ f_s(\mathbf{w}_j^{enc} \oplus \mathbf{w}_i^{enc}), \end{aligned}$$

where T is the number of word tokens in the context and f_s represents a fully connected layer with hidden dimension H and $\oplus$ indicates the concatenation operation. Token at $t = 0$ is "[CLS]". For training, we use cross-entropy loss on the s_i, $e_{i,j}$ with ground truth start, end of the answer span for each token.

During inference, we choose top K answer spans with the highest score summation of start index score and end index score, i.e.,

$$\begin{aligned} A^{span} &= \{(i,j) \mid 1 \le i \le T \text{ and } i \le j \le T\}, \\ a_k &= \max(\{a \mid \#\{(i,j) \mid s_i + e_{i,j} \ge a\} = K\}), \\ A_k^{span} &= \{(i,j) \mid s_i + e_{i,j} \ge a_k\}. \end{aligned}$$

Selected answer A_k^{span} is then given to the question generator as input in the form of an indication of the answer span location in the given context.

2.2 Pre-training Question Generator

In order to generate questions conditioned on different answers that may arise in a context, we generate a question for each of the K answers. Alberti et al. (2019) proposed a pre-training method for this generative model using the self-supervised task of generating the next-sentence. We identify several issues with this approach. This technique is not specifically conditioned on the answer, despite the answer being a strong condition for the question generation task. Also, not all sentences from a paragraph may be relevant to the questions or answers from within that paragraph, so their generation is not an ideal candidate for pre-training question generation model.

To address these issues, we modify the context to exclude the sentence containing the previously generated answer and pre-train the question generation model on the task of generating this excluded answer-containing sentence, conditioned on the answer and the modified context.

Specifically, we exclude answer-containing sentence S^{ans} while retaining the answer, modifying the original context D to D^{ans} as

$$\begin{aligned} S^{start} &= \{p \mid \text{sentence start index} = p\} \cup \{T\}, \\ S^{ans} &= \{(p_s, p_e, i, j) \mid p_s = \max(\{p \le i\}), \\ p_e &= \min(\{p \ge j\}), p \in S^{start}, (i,j) \in A_k^{span}\}, \\ D^{ans} &= [D_{[:p_s]}; D_{[i:j]}; D_{[p_e:]}], (p_s, p_e, i, j) \in S^{ans}, \end{aligned}$$

Note that we change S^{ans} to not exclude the answer-containing sentence for fine-tuning the question generator, i.e.,

$$S^{ans} = \{(p_s, p_e, i, j) | p_s = i, p_e = j\}.$$

In BertGen, we pass the previously generated answer to the generation model in the form of an additional position encoding M^{ans} that indicates the answer location within the context, i.e.,

$$M^{ans} = [\mathbf{m}_0 * p_s; \mathbf{m}_1 * (j - i); \mathbf{m}_0 * (T - p_e)],$$

where $\mathbf{m}_0$ and $\mathbf{m}_1$ indicate trainable vectors corresponding to encoding id 0 and 1, respectively. That is, we assign the encoding id for each word in the context as 0 and each word in the answer as 1. $A * B$ indicates the operation of stacking vector A for B many times.

Next, we generate answer-containing sentence output words' probability $W^o = \{\mathbf{w}_y^o\}_{y=1}^{Y}$ as

$$\begin{aligned} C^{enc} &= \text{BERT Encoder-Q}(D^{ans}, M^{ans}), \\ \mathbf{w}_y^g &= \text{Transformer Decoder}(\{\mathbf{w}_i^o\}_{i=0}^{y-1}, C^{enc}), \\ \{\mathbf{w}_y^o\}_{y=1}^{Y} &= \{\text{Softmax}(\mathbf{w}_y^g E)\}_{y=1}^{Y}, \end{aligned}$$

where C^{enc} is encoded representation of the context and $E \in \mathbb{R}^{d \times V}$ represents a word embedding matrix with vocabulary size V shared between the BERT Encoder-Q and the decoder. Note that $\mathbf{w}_0^o$ is a zero vector for starting the decoding.

Finally, we calculate the loss of the generated words using the cross-entropy loss as

$$\mathbb{L} = -\left(\sum_{y=1}^{Y} \sum_{v=1}^{V} \mathbf{z}_{y,v} \log(\mathbf{w}_{y,v}^o) \right) / Y,$$

where **z** indicates a ground-truth one-hot vector of the answer-containing sentence word. Note that **z** is the question word in the case of fine-tuning.

In this manner, we pre-train the question generation model using a task similar to the final task of conditionally generating the question from a given answer and a context.

3 Experimental Setup

Pre-training Dataset. To build the dataset for answer-containing sentence generation tasks (ASGen) and the synthetic MRC data for pre-training the downstream MRC models, we collect all paragraphs from the entire English Wikipedia dump and synthetically generate questions and answers on these paragraphs. Note that we removed all passages from Wikipedia overlapping with SQuAD dataset (Rajpurkar et al., 2016). We apply filtering and clean-up steps that are detailed in the appendix.

Using BertGen, we extract answers from each given paragraph, and then generate questions for each answer-paragraph pair. Finally, we obtain 43M triples of question-answer-paragraph for the synthetic data. For pre-training on answer-containing sentence generation, we sample 25M answer-paragraph pairs (Full-Wiki) from the final Wikipedia dataset to avoid extremely short contexts less than 500 characters. For ablation studies on pre-training approaches, we sample 2.5M pairs (Small-Wiki)[1] from Full-Wiki and split 25K pairs (Test-Wiki) to evaluate the pre-training method.

Benchmark Datasets. In most MRC datasets, a question and a context are represented as a sequence of words, and the answer span (indices of start and end words) is annotated from the context words based on the question. Among these datasets, we choose SQuAD as the primary benchmark dataset for question generation, since it is the most popular human-annotated MRC dataset. For fair comparison with existing question generation methods, we use the same splits of SQuAD-v1.1, as previously done in Du et al. (2017), Kim et al. (2019), and Dong et al. (2019). We refer to this dataset as Split1. This split has 77K/10K/10K samples for train/dev/test sets. We also evaluate on the reversed dev-test split, referred to as Split2.[2] Additionally, we test our question generation on MS MARCO (Nguyen et al., 2016) and NewsQA (Trischler et al., 2017) to evaluate the generalization of our method to other datasets. In the case of MS MARCO, questions are collected from real user query logs in Bing. For these datasets, we follow pre-processing of Tuan et al. (2020), sampling a subset of original data where the answers are sub-spans of their corresponding paragraphs to obtain train/dev/test sets with 51K/6K/7K samples for MS MARCO and 76K/4K/4K samples for NewsQA. We also conduct experiments on question generation with Natural Questions (Kwiatkowski et al., 2019) and BioASQ (Tsatsaronis et al., 2015). We calculate BLEU-4, METEOR, and ROUGE-L with the script from Du et al. (2017).

To evaluate the effectiveness of generated synthetic MRC data, we test the fine-tuned MRC model on the downstream MRC dataset after training on the generated synthetic data. We calculate the EM/F1 score of the MRC model on SQuAD-v1.1 and v2.0 development set. We also evaluate on the test set of KorQuAD, a Korean dataset created with the same procedure as SQuAD-v1.1.

Implementation Details. For all experiments and models, we use all official original hyperparameters unless otherwise stated below. For BertGen model, we use pre-trained BERT (Base and Large) as encoder and 12 stacked layers of Transformer as decoder. For large version of the model, we use 24 layers of the encoder and the decoder with 737M parameters. For answer prediction, we select top-5 ($K = 5$) for the answer spans. For the generation of unanswerable questions in SQuAD-v2.0, we separate unanswerable and answerable cases and then train separate generation models. For all BertGen models, we pre-train the question generator for 5 epochs on Wikipedia and fine-tune it for 30 epochs on MRC dataset with batch size of 32. For other question generation models, we pre-train for 1 epoch on Wikipedia. For UniLM and T5, the input is formulated as sequence-to-sequence, the first input segment is the concatenation of context and answer, while the second output segment is a missing answer-containing sentence or a question to be generated. We use all official settings for UniLM, ProphetNet (Qi et al., 2020) and ELECTRA (Clark et al., 2020), and use the official pre-trained weights. The training time depends on the data size and the model complexity. For Zhao et al. (2018), pre-training on Full-Wiki takes 48 hours. Pre-training BertGen on Small-Wiki in Table 3 takes 48 hours with 8 Tesla V100 GPU, resulting

---

[1] We use the Korean Wikipedia for KorQuAD.

[2] We use the same splits as provided by Du et al. (2017)

Table 1: Comparison with existing question generation methods on the test set of SQuAD Split1 and Split2. Models marked as '*' indicate results we reproduced.

Group	Question Generation Model	Split1 BLEU-4	Split1 METEOR	Split1 ROUGE-L	Split2 BLEU-4	Split2 METEOR	Split2 ROUGE-L
	Du et al. (2017)	12.3	16.6	39.8	-	-	-
	ASs2s (Kim et al., 2019)	16.2	19.9	44.0	-	-	-
Applying to other methods	Zhao et al. (2018)*	13.0	18.2	41.2	15.1	19.5	43.4
	Zhao et al. (2018)* + ASGen	14.2	19.4	42.8	16.4	20.6	44.7
	T5(Small)*	15.6	23.3	37.1	18.8	25.2	40.5
	T5(Small)* + ASGen	17.0	24.2	38.9	19.6	26.1	41.9
	T5(Large)*	18.9	26.2	41.8	21.7	27.4	44.5
	T5(Large)* + ASGen	19.4	**26.7**	42.0	22.0	**27.8**	45.1
	UniLM (Dong et al., 2019)	22.1	25.1	51.1	23.8	25.6	52.0
	UniLM + ASGen	23.7	25.9	52.3	25.3	26.7	53.3
	ProphetNet (Qi et al., 2020)	23.9	26.6	52.3	25.8	27.5	53.7
	ProphetNet + ASGen	**24.4**	**26.7**	**52.8**	**26.1**	27.6	**53.9**
	BertGen (Large) + ASGen	22.8	25.3	51.2	24.6	25.8	53.0

Table 2: Comparison with existing question generation methods on MS MARCO and NewsQA. We also test our method on Natural Questions. BL-4, MTR, RG-L indicate BLEU-4, METEOR, ROUGE-L.

MS MARCO (test set)	BL-4	MTR	RG-L
Zhao et al. (2018)	17.2	-	-
Tuan et al. (2020)	18.3	19.4	42.8
Ma et al. (2020)	20.5	24.7	49.9
BertGen (Large) + ASGen	**22.9**	**26.7**	**51.8**
NewsQA (test set)	BL-4	MTR	RG-L
Zhou et al. (2017)	9.9	16.7	42.3
Liu et al. (2019)	11.1	17.4	43.2
Tuan et al. (2020)	12.4	**19.0**	44.1
BertGen (Large) + ASGen	**13.8**	18.6	**44.5**
Natural Questions (dev set)	BL-4	MTR	RG-L
BertGen (Large)	31.5	30.4	60.2
BertGen (Large) + ASGen	**35.3**	**32.9**	**61.3**

Table 3: Ablation of pre-training methods, i.e., pre-training on NS, ASGen, and ASGen without conditioning on a given answer (w/o A), on the test set of SQuAD splits and on Test-Wiki.

Pre-train on Small-Wiki	Wiki	Split1	Split2
BertGen (w/o pre-train)	-	15.0	17.1
BertGen+NS	1.4	19.0	20.2
BertGen+ASGen w/o A	**5.2**	19.9	21.0
BertGen+ASGen	**5.2**	**20.1**	**21.4**
Pre-train on Full-Wiki	Wiki	Split1	Split2
BertGen+NS	3.4	20.6	22.6
BertGen+ASGen	8.2	22.2	24.2
BertGen(Large)+ASGen	**8.3**	**22.8**	**24.6**

Table 4: Average of 10 human evaluation scores over random samples from SQuAD. Each column indicates Syntax (ST), Semantics (SM), Context-Relevance (CR) and Answer-Relevance (AR) in the range 1 to 5.

Model	ST	SM	CR	AR
BertGen	4.04	3.93	4.20	3.25
BertGen+NS	4.60	4.54	4.49	3.63
BertGen+ASGen	**4.71**	**4.69**	**4.74**	**4.14**
UniLM	4.25	4.31	4.54	4.06
UniLM+ASGen	**4.71**	**4.79**	**4.70**	**4.17**

in 5.1, 4.3 BLEU-4 improvement on Split1, Split2 respectively. The pre-training for BertGen (Large) with Full-Wiki takes 1,224 hours and fine-tuning takes 72 hours. Mecab (Kudo, 2006) is used for Korean tokenizer.

Comparison of the Pre-training Method. We compare ASGen with a method from Alberti et al. (2019), which is pre-training on next-sentence generation task (NS), and with a method from Golub et al. (2017), which only trains the generative model on the final MRC dataset. We reproduced these methods on BertGen as described in their original work and evaluate question generation scores on the SQuAD splits as well as corresponding sentence generation scores on Test-Wiki.

Comparison of Downstream Results. To check the effectiveness of our method on downstream MRC tasks, we evaluate our generated synthetic data on SQuAD-v1.1, v2.0, and KorQuAD by training MRC models (BERT, BERT+CLKT and ELECTRA) on generated data followed by fine-tuning on the train set for each dataset. The structure of BERT+CLKT model is the same as that of original BERT except that the model is pre-trained for the Korean language. Due to the absence of common pre-trained BERT for Korean, we used this model.

4 Experimental Results

4.1 Question Generation Performance

Comparison to Existing Methods. To evaluate ASGen, we fine-tune the question generation models on both SQuAD splits, after pre-training on answer-containing sentence generation task. As shown in Table 1, 'BertGen (Large) + ASGen' and 'UniLM + ASGen' outperforms UniLM on both splits. Moreover, ASGen consistently improves the performance when applied to other question generation models such as Zhao et al. (2018), T5 (Small and Large), and UniLM across all metrics for both splits. In particular, applying ASGen on UniLM further improves its question generation capability, achieving BLEU-4, METEOR, and ROUGE-L as 23.7, 25.9, 52.2, and 25.3, 26.7, 53.3 on both splits, respectively. We reproduce Zhao et al. (2018) and T5, and use the official code of UniLM with no architecture or parameter changes.

Additionally, as shown in Table 2, 'BertGen (Large) + ASGen' outperforms all existing models on all scores on both MS MARCO and NewsQA, except for comparable METEOR scores in NewsQA. Our method also shows improvement on Natural Questions (Kwiatkowski et al., 2019) (short answer) dataset, where questions are collected from real user query logs on Google.

Ablation Study of Pre-training Task. We also compare the BLEU-4 scores between various pre-training tasks to show the effectiveness of ASGen. As shown in Table 3, ASGen outperforms NS in the recreation score of sentence on Test-Wiki, e.g. 5.2 vs. 1.4 in Small-Wiki and 8.2 vs. 3.4 in Full-Wiki. ASGen outperforms NS in question generation, e.g. 22.2 vs. 20.6 and 24.2 vs. 22.6 in the two splits, respectively. We also observe that conditioning on a given answer improves ASGen, e.g. 20.1 vs. 19.9 in Split1 and 21.4 vs. 21.0 in Split2.

Human Evaluation. As Sultan et al. (2020) mentioned in their paper, accuracy-based measurements such as BLEU-4, METEOR and ROUGE-L may not be adequate to test the diversity of a question. Due to this, we also judge the quality of questions by human evaluation involving 10 evaluators over metrics such as syntax, validation of semantics, question to context relevance and question to answer relevance on 50 randomly chosen samples on SQuAD-v1.1 dev set. As shown in Table 4, applying ASGen consistently improves the human evaluation scores.

Table 5: Effectiveness of synthetic data for MRC model on SQuAD (SQD) and KorQuAD (KQD).

MRC model	Dev-SQDv1.1		Dev-SQDv2.0	
	EM	F1	EM	F1
BERT (Large)	83.9	90.9	78.8	81.8
+synthetic data	**86.3**	**92.7**	**84.5**	**87.4**
BERT (WWM)	86.5	92.8	83.1	85.9
+synthetic data	**87.4**	**93.5**	**85.5**	**88.4**
ELECTRA (Large)	-	-	87.4	90.2
+synthetic data	-	-	**88.2**	**91.3**
MRC model	**Dev-KQD**		**Test-KQD**	
	EM	F1	EM	F1
BERT+CLKT	87.1	94.5	86.2	94.1
+synthetic data	**87.8**	**95.0**	**86.7**	**94.6**

Table 6: Comparison of downstream EM/F1 scores using BERT(Large) MRC model with the synthetic data from different pre-training methods.

Synthetic Data	Dev-v1.1		Dev-v2.0	
	EM	F1	EM	F1
BertGen (w/o pre-train)	85.1	91.4	80.9	83.9
BertGen+NS	85.6	92.3	81.5	85.8
BertGen+ASGen	**86.3**	**92.7**	**84.5**	**87.4**

4.2 Downstream MRC Task Performance

To show the effectiveness of the generated synthetic data, we train MRC models on generated data, before fine-tuning on the downstream data. As shown in Table 5, the synthetic data generated by 'BertGen (Large) + ASGen' consistently improves the performance of BERT (Large, WWM) by a significant margin. Pre-training BERT on synthetic data improves F1 scores by 1.8 on SQuAD-v1.1 and 5.6 on SQuAD-v2.0 for BERT (Large), and 0.7 on SQuAD-v1.1 and 2.5 on SQuAD-v2.0 for BERT (WWM). Synthetic data also improves ELECTRA performance on SQuAD-v2.0, and BERT+CLKT performance on KorQuAD.

Also, to show improvement due to our pre-training method in the downstream MRC task, we compare the EM/F1 scores of BERT (Large) models trained on synthetic data generated by different question generation models, 'BertGen', 'BertGen + NS' and 'BertGen + ASGen'. As shown in Table 6, our method outperforms other methods both on SQuAD-v1.1 and SQuAD-v2.0.

4.3 Effects of Training Data Size

Fig. 2 shows the effects of varying amounts of downstream MRC data and synthetic data on F1 scores of BERT (Large). In Fig. 2-(a), where we fix the size of synthetic data as 43M, pre-training with

Table 7: The performance of our method on limited-data domain (BioASQ). Note that the scores of question generation are obtained from BioASQ factoid-type 6b. All experiments used the official code of Yoon et al. (2020).

Question Generation Model	BLEU-4	METEOR	ROUGE-L
BertGen (Large)	6.6	10.0	33.1
BertGen (Large) + ASGen (Full-Wiki)	**12.6**	**17.8**	**40.0**

MRC model	Pre-training Data	Factoid (MRR)	Yes/No (Macro F1)	List-Type (F1)
BERT(Large)	-	34.3	53.8	36.1
BERT(Large)	ASGen (Full-Wiki)	49.2	**81.1**	**39.8**
BioBERT(Large)	PubMed	**52.3**	80.1	38.1

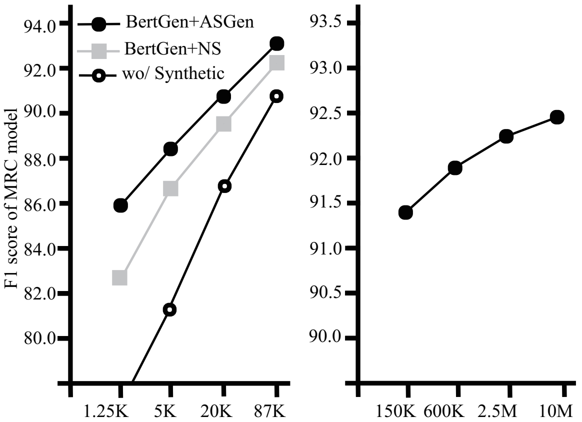

(a) Varying MRC data size (b) Varying synthetic data size

Figure 2: F1 scores of BERT (Large) on SQuAD-v1.1 dev by limiting size of MRC and synthetic data.

'BertGen + ASGen' consistently outperforms 'BertGen + NS' for all sizes of downstream data. While the performance difference is particularly apparent for smaller sizes of downstream data, it persists even on using the entire MRC data (SQuAD-v1.1). In Fig. 2-(b), we also conduct experiments by training BERT (Large) using different amounts of generated synthetic data while keeping the number of pre-training steps constant and using the full size of downstream MRC data. Increasing the amount of synthetic data used consistently improves the accuracy of the MRC model.

4.4 Transfer Learning to Limited Domain

We also conduct experiments on BioASQ (Tsatsaronis et al., 2015) dataset to show the effectiveness of our model in limited-data domains having less annotated data. As shown in Table 7, ASGen improves the question generation scores by 6.0 BLEU-4, 7.8 METEOR and 6.9 ROUGE-L on BioASQ factoid-type 6b. Moreover, using 'Full-Wiki' data enhances the performance of BERT(Large) by a large margin and outperforms BioBERT (Lee et al., 2019a), by 0.95 Macro F1 (Yes/No) and 1.63 F1 (List). Note that BioBERT is specifically pre-trained on a medical corpus (PubMed) whereas we use a generic Wikipedia corpus ('Full-Wiki'), with our generation models fine-tuned on SQuAD.

Table 8: Manual categorization of the reasoning type for generated answerable questions. Note that each example can be assigned to multiple types.

Reasoning Type	BertGen +ASGen	SQuAD v1.1
Lexical Variation (Synonymy)	40.7%	33.3%
Lexical Variation (World Knowledge)	4.0%	9.1%
Syntactic Variation	53.3%	64.1%
Multi Sentence Reasoning	21.3%	13.6%
Ambiguous/Unanswerable	4.0%	6.1%

4.5 Qualitative Analysis of Generation

Comparison of Sample Questions. We qualitatively compare the generated questions after pre-training BertGen with NS and ASGen to demonstrate the effectiveness of our method. For the correct answer "49.6%" as shown in the first sample in Table 9, the word "Fresno", which is critical to make the question specific, is omitted by NS, while ASGen's question does not suffer from this issue. Note that the word "Fresno" occurs in the answer-containing sentence. This issue also occurs in the second sample, where NS uses the word "available" rather than relevant words from the answer-containing sentence, but ASGen uses many of these words such as "most" and "popular" to generate contextually rich questions. Also, the question from NS is about "two" libraries, while the answer is about "three" libraries, showing the lack of sufficient conditioning on the answer. Similarly, the third example also shows that ASGen generates more contextual questions than NS by including the exact subject "TARDIS" based on the corresponding answer. Based on these observations and from the score improvements in Table 3, we conjecture that ASGen leads the question generation model to better condition on the answer and to

Table 9: Examples of questions generated on SQuAD-v1.1 development set. We compare generated questions from 'BertGen + ASGen' with 'BertGen + NS'. Colored Text indicates given answers.

Context	(omit) ... The population density was 4,404.5 people per square mile. (1,700.6km). The racial makeup of Fresno was 245,306 (49.6%) White, 40,960 (8.3%) ... (omit)
BertGen + NS	What percent of the population is White?
BertGen + ASGen	What percentage of the Fresno population is White?
Context	(omit) ... in the world. Cabot Science Library, Lamont Library, and Widener Library are three of the most popular libraries for undergraduates to use ... (omit)
BertGen + NS	Which two libraries are available for undergraduates to use?
BertGen + ASGen	What are the three most popular libraries for undergraduates?
Context	(omit) ... in a stolen Mark I Type TARDIS "Time and Relative Dimension in Space" time machine which allows him to travel across time and space. ... (omit)
BertGen + NS	What does the doctor refer to?
BertGen + ASGen	What does the TARDIS stand for?

Table 10: Manual categorization of the reasoning type for unanswerable questions.

Reasoning Type	BertGen +ASGen	SQuAD v2.0
Negation	8.0%	9.0%
Antonym	14.7%	20.0%
Entity Swap	36.0%	21.0%
Mutual Exclusion	9.3%	15.0%
Impossible Condition	7.3%	4.0%
Other Neutral	19.3%	24.0%
Answerable	5.3%	7.0%

generate more contextualized questions than NS.

Categorization of Reasoning Type. We manually categorized the reasoning type of 150 randomly sampled generated questions on Wikipedia for both answerable and unanswerable questions. The results Table 8 and Table 10 show that generated questions using ASGen often require multi-hop or other non-trivial reasoning. We follow the same categorization as done by Rajpurkar et al. (2016, 2018).

5 Related Work

Question Generation. Researchers have actively studied question generation for various purposes, including for data augmentation in question answering. Du et al. (2017) proposed an attention-based model for question generation by encoding sentence-level as well as paragraph-level information. Zhao et al. (2018) utilized a gated self-attention encoder with a max-out unit to handle long paragraphs. Song et al. (2018) introduced a query-based generative model to jointly solve question generation and answering tasks. Kim et al. (2019) separately encoded the answer and the rest of the paragraph for question generation. Ma et al. (2020) suggested sentence-level semantic matching and answer-position-aware question generation. Tuan et al. (2020) show that incorporating interactions across multiple sentences enhances question generation performance. Our approach can further improve the question generation quality of these methods by pre-training them with the answer-containing sentence generation task.

Transfer Learning. Pre-training methods are popular in natural language processing for learning contextualized word representations. Open-GPT (Radford et al., 2018), BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), PEGASUS (Zhang et al., 2019), ERNIE-GEN (Xiao et al., 2020), UniLM (Dong et al., 2019), UniLMv2 (Bao et al., 2020), T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) utilize Transformer (Vaswani et al., 2017) to learn different types of language models on a large dataset followed by fine-tuning on a downstream task. These pre-training approaches tend to be very generic, while our approach is a more appropriate pre-training method focused on the specific task of question generation. Lee et al. (2019b) suggested a pre-training method for information retrieval called Inverse Cloze Task. Unlike this method, our pre-training task for the question generator is strongly conditioned on the answer and focuses on generating missing answer-containing sentence in the context to learn better representations more suitable to the question generation task.

Synthetic Data Generation. Subramanian et al. (2018) show that neural models generate better candidate answers from a given paragraph than using off-the-shelf tools or selecting named entities and noun phrases. Yang et al. (2017) introduced a training method for the MRC model by combining synthetic data and human-annotated data. Similar to our method, Golub et al. (2017) proposed to generate questions conditioned on generated answers by separating the answer generation and the question generation. Unlike this paper, they do not pre-train their question generator on the answer-containing sentences. Dong et al. (2019) also show that utilizing synthetic data boosts the performance of MRC models. Inspired by these previous studies, we propose a newly designed pre-training technique that improves capability of question generation models.

6 Conclusions

We propose a novel pre-training method called ASGen to learn generating contextually rich questions better conditioned on the answers. Our approach improves question generation ability of existing methods, achieves new state-of-the-art results on MS MARCO and NewsQA, and the synthetic data increases downstream MRC accuracy across a wide range of datasets without any modification to the existing MRC models.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)), the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2019R1A2C4070420), and Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2020-0-00368, A Neural-Symbolic Model for Knowledge Acquisition and Inference Techniques).

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6168–6173. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. *CoRR*, abs/2002.12804.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *CoRR*, abs/1707.03904.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13042–13054.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1342–1352. Association for Computational Linguistics.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 835–844. Association for Computational Linguistics.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The*

Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 6602–6609. AAAI Press.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. jp*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019a. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019b. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1.0: Korean QA dataset for machine reading comprehension. *CoRR*, abs/1909.07005.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1106–1118. ACM.

Xiyao Ma, Qile Zhu, Yanlin Zhou, and Xiaolin Li. 2020. Improving question generation with sentence-level semantic matching and answer position inferring. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8464–8471. AAAI Press.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2401–2410.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper.pdf*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2018. A unified query-based generative model for question generation and question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 78–88. Association for Computational Linguistics.

Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. 2020. On the importance of diversity in question generation for QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, Online. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200. Association for Computational Linguistics.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artieres, Axel Ngonga, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138.

Luu Anh Tuan, Darsh J. Shah, and Regina Barzilay. 2020. Capturing greater context for question generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9065–9072. AAAI Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Dongling Xiao, Han Zhang, Yu-Kun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3997–4003. ijcai.org.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5754–5764.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1040–1050, Vancouver, Canada. Association for Computational Linguistics.

Wonjin Yoon, Jinhyuk Lee, Donghyeon Kim, Minbyul Jeong, and Jaewoo Kang. 2020. Pre-trained language model for biomedical question answering. In *Machine Learning and Knowledge Discovery in Databases*, pages 727–740, Cham. Springer International Publishing.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *CoRR*, abs/1912.08777.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3901–3910. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*, volume 10619 of *Lecture Notes in Computer Science*, pages 662–671. Springer.

Supplemental Material (Appendix)

A Question Generation on more Datasets

We also evaluate the question generation model on another data split (Split3) from Zhao et al. (2018). Split3 is obtained by dividing the original development set in SQuAD-v1.1 into two equal halves randomly and choosing one of them as the development set and the other as test set while retaining the train set in SQuAD-v1.1. As shown in Table 11, applying ASGen to the reproduced question generation model from Zhao et al. (2018) improves BLEU-4, METEOR, and ROUGE-L score on Split3 by 1.3, 0.9, and 1.3, respectively.

Table 11: Additional experiments on the effectiveness of ASGen on the test set of SQuAD Split3. Small-Wiki is used to pre-train the models. Models with '*' indicate those results we reproduced.

Model + pre-training method	BL-4	MTR	RG-L
Zhao et al. (2018)	16.8	20.6	44.9
Zhao et al. (2018)*	16.3	20.3	44.5
Zhao et al. (2018)* + ASGen	**17.6**	**21.2**	**45.8**

B Training Electra MRC Model using only Small Generated Synthetic Data

To further study the effect of training data size, we apply our synthetic data to the ELECTRA (Clark et al., 2020) MRC model. In Table 12, we report the mean EM/F1 score on SQuAD 2.0 development set of four runs by using official Electra source code[3] and the pre-trained checkpoint. Pre-training ELECTRA on the generated synthetic data using ASGen improves 0.8 EM and 1.1 F1 score on the downstream MRC dataset, SQuAD-v2.0.

Table 12: Ablation study of applying our method to ELECTRA (Clark et al., 2020) on SQuAD-v2.0 dev set after pre-training on the generated synthetic data using ASGen with Small-Wiki.

MRC model	Synthetic Data	Dev set EM	Dev set F1
ELECTRA (Large)	-	87.4	90.2
	'Small-Wiki'	87.9	90.8
	'Full-Wiki'	**88.2**	**91.3**

C Additional Downstream MRC Task Performance

Additionally to show the effectiveness of the generated synthetic data, we also train MRC models on generated data, before fine-tuning on two downstream datasets, Natural Questions and NewsQA. As shown in Table 13, the synthetic data generated by 'BertGen (Large) + ASGen' consistently improves the F1 score of baseline BERT models.

D Transfer Learning to Other MRC Dataset (QUASAR-T)

To show that our generated data is useful for other MRC datasets, we fine-tune and test the MRC model on QUASAR-T (Dhingra et al., 2017), which is another large-scale MRC dataset, after training on the synthetic data generated from SQuAD-v1.1. In this experiment, we first fine-tune 'BertGen + ASGen' using SQuAD-v1.1, and using synthetic data generated by this model, we train the BERT (Large) MRC model. Afterwards, we fine-tune BERT (Large) for the downstream MRC task using QUASAR-T data. QUASAR-T has two separate datasets, one with short snippets as context, and the other with long paragraphs as context. As shown in Table 14, training with our synthetic data improves the F1 score on the test set by 2.2 and 1.7 for the two cases, respectively.

Table 13: Effectiveness of synthetic data for MRC model on dev set of Natural Questions and NewsQA.

MRC model	Natural Questions (Short) P	R	F1
BERT (Joint)	**60.09**	46.00	52.10
+synthetic data	59.29	**48.22**	**53.17**

MRC model	Natural Questions (Long) P	R	F1
BERT (Joint)	61.45	**68.61**	64.83
+synthetic data	**63.75**	67.50	**65.56**

MRC model	NewsQA EM	F1
BERT (Large)	51.96	62.54
+synthetic data	**54.73**	**64.53**

Table 14: EM/F1 scores of the BERT (Large) fine-tuned on QUASAR-T dataset. The used synthetic data is generated from ASGen trained on SQuAD-v1.1 (Full-Wiki).

Synthetic Data	Short(Dev) EM	Short(Dev) F1	Short(Test) EM	Short(Test) F1
-	74.3	78.6	74.1	77.8
Full-Wiki	**76.5**	**80.1**	**76.5**	**80.0**

Synthetic Data	Long(Dev) EM	Long(Dev) F1	Long(Test) EM	Long(Test) F1
-	72.1	75.6	72.1	74.8
Full-Wiki	**74.2**	**77.4**	**73.8**	**76.5**

E Details of Wikipedia Preprocessing

To build the answer-containing sentence generation data and the synthetic MRC data for SQuAD (Rajpurkar et al., 2016), we collect all paragraphs from all articles of the entire English Wikipedia dump and generate questions and answers on these paragraphs. We apply extensive filtering and clean-up to only retain the highest-quality paragraphs from Wikipedia, as follows.

To filter out low-quality articles, we remove those with less than 200 cumulative page-views including all re-directions in a two-month period.

---

[3] https://github.com/google-research/electra

In order to calculate the number of page-views, official Wikipedia page-view dumps were used. Of the 5.4M original Wikipedia articles, filtering by page-views leaves 2.8M articles. We also remove those articles with less than 500 characters, as they are often low-quality stub articles, which further removes additional 16% of the articles. We remove all "meta" namespace pages such as talk, disambiguation, user pages, portals, etc. as they often contain irrelevant text or casual conversations between editors. In order to extract clean text from the wiki-markup format of the Wikipedia articles, we remove extraneous entities from the markup including table of contents, headers, footers, links/URLs, image captions, IPA double parentheticals, category tables, math equations, unit conversions, HTML escape codes, section headings, double brace templates such as info-boxes, image galleries, HTML tags, HTML comments, and all tables.

We then split the cleaned text into paragraphs and remove all paragraphs with less than 150 characters or more than 3,500 characters. Paragraphs with the number of characters between 150 to 500 were sub-sampled such that these paragraphs make up 16.5% of the final dataset, as originally done for the SQuAD dataset. Since the majority of the paragraphs in Wikipedia are rather short, out of the 60M paragraphs from the final 2.4M articles, our final Wikipedia dataset contains 8.3M paragraphs. Finally, we generate 43M answer-paragraph pairs from the final Wikipedia dataset with the answer generator of BertGen in this paper.

F Central Tendency and Variation for Human Evaluation

Human evaluation involves 10 evaluators over metrics such as syntax (ST), validation of semantics (SM), question to context relevance (CR) and question to answer relevance (AR) on 50 randomly chosen samples on SQuAD-v1.1 development set. Each score is in the range 1 to 5. Central tendency and variation can be found in Table 15.

G Central Tendency and Variation for the Downstream Tasks

For the EM and F1 scores on downstream SQuAD-v1.1 and v2.0 development set in our main paper, we selected 5 model checkpoints from the same pre-training on the synthetic data in different numbers of training steps. We then fine-tuned each of these models on the final downstream data three times each, chose the best performing model on the development set, and reported its score. Central tendency and variation can be found in Table 16.

Table 15: Central tendency and variation for human evaluation scores. ± is 95% confidence interval.

Model	ST	SM	CR	AR
BertGen	4.04 ±0.18	3.93 ±0.19	4.20 ±0.16	3.25 ±0.22
BertGen + NS	4.60 ±0.12	4.54 ±0.13	4.49 ±0.14	3.63 ±0.22
BertGen + ASGen	**4.71** ±0.10	**4.69** ±0.11	**4.74** ±0.09	**4.14** ±0.18
UniLM	4.25 ±0.16	4.31 ±0.16	4.54 ±0.12	4.06 ±0.19
UniLM + ASGen	**4.71** ±0.11	**4.79** ±0.09	**4.70** ±0.11	**4.17** ±0.18

Table 16: Central tendency and variation for the score of our approach, BertGen(Large) + ASGen, on downstream SQuAD-v1.1 and v2.0 dataset. ± is standard deviation.

MRC model	Dev-v1.1		Dev-v2.0	
	EM	F1	EM	F1
BERT (Large)	86.2 ±0.1	92.7 ±0.1	84.4 ±0.2	87.3 ±0.1
BERT (WWM)	**87.4** ±0.1	**93.4** ±0.1	85.5 ±0.1	88.3 ±0.1
Electra (Large)	- -	- -	**88.4** ±0.3	**91.2** ±0.1

H Details of Generating Unanswerable Questions

The mechanism of generating questions may differ in generating answerable and unanswerable questions. For example, the model could exploit a mismatched phrase to make a question plausible but unanswerable. In order to reflect these characteristics, we train answerable and unanswerable models separately. We first take the BertGen model pre-trained on the ASGen task and then fine-tune this model on the no-answer question generation on SQuAD-v2.0. We infer with this model on the entire Wikipedia to make negative examples for un-answerble synthetic data for pre-training MRC models on SQuAD-v2.0.

I BLEU-4, METEOR, and ROGUE-L

BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) are widely-used metrics for evaluating the quality of generated text, where the quality indicates the degree of correspondence between generated text and reference texts. BLEU uses modified precision to compare a generated text against the reference texts. BLEU-4 calculates a weighted score of unigram, bigram, trigram, and 4-gram based matching. METEOR uses harmonic mean between precision and recall of unigrams, but with for recall given more importance than precision. Unlike BLEU, METEOR also tries to match synonyms and performs stemming instead of just relying on exact word matching. ROUGE-L is the longest common sub-sequence based word matching. The longest co-occurrence in sequences of n-grams between generated text and reference texts are considered for calculating the score. To calculate these evaluation scores, we follow the script from Du et al. (2017), except for the corresponding scripts from other question generation models when ASGen is applied to them.

J Links to Downloadable Components

For Wikipedia data, we downloaded English Wikipedia dump in Feb 2019 from (`https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2`). Page views were obtained from (`https://dumps.wikimedia.org/other/pageviews/2019/2019-01/`) and (`https://dumps.wikimedia.org/other/pageviews/2019/2019-02/`). For applying our method to other existing question generation models, we reproduce Zhao et al. (2018) using publicly available code (`https://github.com/seanie12/neural-question-generation`), Raffel et al. (2020) using publicly available code (`https://github.com/patil-suraj/question_generation`) and use the official code of Dong et al. (2019) (`https://github.com/microsoft/unilm`).

Learning Slice-Aware Representations with Mixture of Attentions

Cheng Wang **Sungjin Lee** **Sunghyun Park** **Han Li** **Young-Bum Kim** **Ruhi Sarikaya**
Amazon Alexa AI
{cwngam,sungjinl,sunghyu,lahl,youngbum,rsarikay}@amazon.com

Abstract

Real-world machine learning systems are achieving remarkable performance in terms of coarse-grained metrics like overall accuracy and F-1 score. However, model improvement and development often require fine-grained modeling on individual data subsets or slices, for instance, the data slices where the models have unsatisfactory results. In practice, it gives tangible values for developing such models that can pay extra attention to critical or interested slices while retaining the original overall performance. This work extends the recent slice-based learning (SBL) (Chen et al., 2019) with a mixture of attentions (MoA) to learn slice-aware dual attentive representations. We empirically show that the MoA approach outperforms the baseline method as well as the original SBL approach on monitored slices with two natural language understanding (NLU) tasks.

1 Introduction

Though machine learning systems have been achieving excellent performance in terms of coarse-grained metrics like accuracy, they perform poorly or even fail on some individual data subsets (i.e., slices). For instance, many models have difficulties when learning for classes with only a few samples or samples with challenging structures. Inspecting particular data slices can serve as an important component in model development cycles. A recently proposed slice-based learning (SBL) exhibited compelling results with more than 3% improvements on pre-defined slices (Chen et al., 2019) in the task of binary classification. However, one potential limitation of the existing attention mechanism in SBL is that in multi-class cases, the attention suffers from the difficulty in using the experts' confidences appropriately for computing slice distributions (refer to Sec. 3).

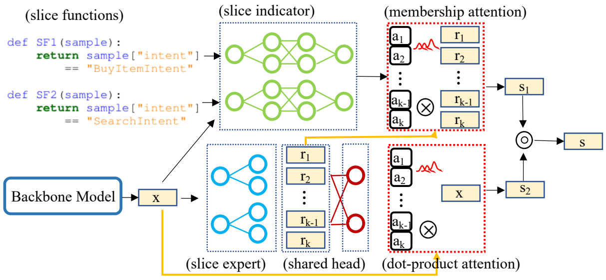

```
def SF1(sample):
    return sample["intent"]
        == "BuyItemIntent"

def SF2(sample):
    return sample["intent"]
        == "SearchIntent"
```

Figure 1: The slice-aware architecture with MoA. It consists of six components: (1) *slice functions* define the special data slices that we want to monitor; (2) *backbone model* for feature extraction (e.g., BERT); (3) *slice indicators* are membership functions to predict if a sample belongs to the slice; (4) *slice experts* aim to learn slice-specific representations; (5) *shared head* is the base task predictive layer across experts and (6) the proposed *mixture of attentions (MoA)* learns to attend to the slices of interest. It contains two different attention mechanism (red boxes): a membership attention and a dot-product attention. The MoA learns to re-weight the expert representation $\mathbf{r}$ to a slice-aware representation $\mathbf{s}$ and the original representation $\mathbf{x}$ to $\mathbf{s}$ (yellow lines). The slice distributions are computed in deterministic (weighted sum of slices) or stochastic (sampling) way in re-weighting $\mathbf{r}$ and $\mathbf{x}$.

In this paper, we extend SBL with a mixture of attentions (MoA) mechanism. Two different attention mechanisms are learned to jointly attend to the defined slices from different representations in different latent subspaces. The first attention is based on slice membership likelihood and/or experts confidence as in SBL (Chen et al., 2019), which we call *membership attention*. The second one is *dot-product attention* that is based on the backbone model (e.g., BERT (Devlin et al., 2019)) extracted representations. The MoA approach is akin to multi-head attention (Vaswani et al., 2017) but with different attention types that receive different inputs.

As presented in Figure 1, the two attentions in MoA can work jointly to attend to (1) the expert rep-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1530–1536
August 1–6, 2021.

```
def SF_Length(utterance,k=10):
    return len(utterance) < k

def SF_Time(utterance):
    return "time" in utterance

def SF_Email(utterance):
    return "email" in utterance
```

```
def SF_Long(sentence,k=10):
    n = len(sentence
            .split(' '))
    return  n > k

def SF_Question(sentence):
    return sentence[-1]
            == '?'
```

Table 1: The designed slice functions (SFs)[1]. Left: We monitor three data slices - short utterances, those involving "time", and those involving "email". Right: We monitor long sentences and questions[2].

resentation $\mathbf{r}$ and (2) the backbone model extracted representation $\mathbf{x}$, and finally form an attentive representation $\mathbf{s}$. The $\mathbf{s}$ is a slice-aware featurization of the samples in the particular data slices and will be used for making a final model prediction.

We argue that learning joint attention with MoA from different resources for computing slice distributions is beneficial (Vaswani et al., 2017; Li et al., 2018). We evaluate the effectiveness of our proposed approach on intent detection (Liu et al., 2019) and linguistic acceptability (Warstadt et al., 2018) tasks.

Our main contributions are twofold:

- We extend SBL with MoA. The MoA approach has the ability to attend to slices in deterministic (weighted summation) and stochastic (sampling) ways.
- We conduct extensive experiments on two NLU tasks. The results show that MoA outperforms the baseline and vanilla SBL by average up to 9% and 6% respectively on defined slices.

2 Architecture

Figure 1 presents the slice-aware architecture based on SBL (Chen et al., 2019). Let $\{x^n, y^n\}_n^N$ be a dataset with N samples. We aim to learn slice-aware representation $\mathbf{s}$ from slice-experts-learned representation $\mathbf{r}$ and backbone-model-extracted representation $\mathbf{x}$.

We first define **slice functions (SFs)** as in Table 1 to split the dataset into k slices of interests. Each sample is assigned with a slice label $\gamma \in [0, 1]$ in $\{\gamma_1, \gamma_2, ..., \gamma_k\}$ as supervision data[3].

Second, we use a **backbone model** like BERT to extract representation $\mathbf{x} \in \mathbf{R}^d$ for a given sample. Then, **slice indicators** $f_i(\mathbf{x}; \mathbf{w}_i^f)$, $\mathbf{w}_i^f \in \mathcal{R}^{d \times 1}$, $i \in \{1, .., k\}$ map $\mathbf{x}$ to a prediction h_i. f_i are trained with $\{\mathbf{x}^n, \gamma^n\}_n^N$ to predict whether a sample belongs to a particular slice. They are learned with cross entropy loss

$$\zeta_1 = \sum_i^k \mathcal{L}_{CE}(h_i, \gamma_i) \tag{1}$$

Then, **slice experts** $g_i(\mathbf{x}; \mathbf{w}_i^g)$, $\mathbf{w}_i^g \in \mathcal{R}^{d \times d}$ learn a mapping from $\mathbf{x}$ to a slice vector $r_i \in \mathcal{R}^d$ with the samples that only belong to the slice, followed by a **shared head**, which is shared across all experts and maps r_i to a prediction $\hat{y} = \varphi(r_i; \mathbf{w}_s)$. g_i and φ are learned on the base (original) task with ground-truth label y by

$$\zeta_2 = \sum_i^k \gamma_i \mathcal{L}_{CE}(\hat{y}, y) \tag{2}$$

Finally, a **mixture of attentions**(MoA) (as in Sec. 3) re-weights $\mathbf{r}$ and $\mathbf{x}$ to form $\mathbf{s}$. The $\mathbf{s}$ goes through a final prediction function η on the base task. The loss function is

$$\zeta_3 = \mathcal{L}_{CE}(\eta(\mathbf{s}; \mathbf{w}_p), y) \tag{3}$$

The total loss is a combination of the loss for slice indicators, slice experts and base task prediction function:

$$\zeta = \zeta_1 + \zeta_2 + \zeta_3 \tag{4}$$

The whole model is optimised with back-propagation (Rumelhart et al., 1986) in an end-to-end way.

3 Methodology

The SBL approach (Chen et al., 2019) proposed a slice-residual attention modules (SRAMs) that are directly based on stacked membership likelihood $H \in \mathbf{R}^k$ and experts' prediction confidence

---

[1] The SFs are task-dependent and not assumed to be perfectly accurate. They can be noisy or from weak supervision sources (Ratner et al., 2016). Here, for the task in sec.4.2, SFs are defined to improve the slices where the model has unsatisfactory results as compared to the overall performance. For the task in sec.4.3, we define the SFs for the slices of interest.

[2] Alternatively, 5W1H rule for questions (Kim et al., 2019).

[3] s_1 is the base slice, and s_2 to s_k are the slices of interest.

$|Y| \in \mathbf{R}^{c \times k}$, $c = 1$ (i.e., binary classification). Then, slice distribution (attention weights) is computed with $a = \textsc{softmax}(H + |Y|)$. One potential limitation of this mechanism is that the above formulation can lead to mismatch shape in element-wise addition when $c > 2$ (i.e., multi-class classification). To circumvent this, we propose a mixture of attentions (MoA) to augment membership attention with dot-product attention from different information resources.

3.1 Mixture of Attentions

Let $\mathbf{x} \in \mathbf{R}^d$ be the original representation from the backbone model (e.g., BERT), $h_i \in \mathbf{R}^c$ ($c = 1$) as i-th indicator function's prediction, and $r_i \in \mathbf{R}^d$ as i-th expert learned representation. When stacking on k slices, we have $\mathbf{h} \in \mathbf{R}^{c \times k}$ and $\mathbf{r} \in \mathbf{R}^{d \times k}$. MoA's goal is to (1) attend to $\mathbf{r}$ based on indicator functions' membership likelihood and/or experts confidence[4]; (2) attend to $\mathbf{x}$ with a dot-product attention; (3) to form a new slice-aware attentive representation $\mathbf{s} \in \mathbf{R}^d$ with weighted (sampled) $\mathbf{r}$ and $\mathbf{x}$.

The slice distributions are computed differently. For membership attention, the probability $\mathbf{p}_1 = \textsc{softmax}(\mathbf{h})$ or $\mathbf{p}_1 = \textsc{softmax}(\mathbf{h} + |\mathbf{r}|) \in \mathbf{R}^k$ (d=1 in binary classification). Then membership weighted slice representation is computed: $\mathbf{s}_1 = \mathbf{r} \bullet \mathbf{p}_1, \mathbf{s}_1 \in \mathbf{R}^d$. For dot-product attention, we aim to learn an attention matrix $\mathbf{A} = \{\mathbf{a}_1, ..., \mathbf{a}_k\}, \mathbf{a} \in \mathbf{R}^d, \mathbf{A} \in \mathbf{R}^{d \times k}$ is randomly initialized and learned by the standard back-propagation. Intuitively, each $\mathbf{a}$ is learned to be a slice prototype (Wang and Niepert, 2019; Roy et al., 2020). The probability over slices is computed as:

$$\mathbf{p}_2 = \textsc{softmax}(\mathbf{A}^\top \bullet \mathbf{x}) \in \mathbf{R}^k \tag{5}$$

A new attentive representation $\mathbf{s}_2$ is formed by weighting $\mathbf{A}$ with $\mathbf{p}_2$:

$$\mathbf{s}_2 = \mathbf{A} \bullet \mathbf{p}_2, \quad \mathbf{s}_2 \in \mathbf{R}^d \tag{6}$$

or sampling from $\mathbf{A}$:

$$\text{sample } \mathbf{s}_2 \sim \{\mathbf{a}_1, ..., \mathbf{a}_k\} \tag{7}$$

Then slice-aware vector $\mathbf{s}$ is computed by

$$\mathbf{s} = \mathbf{s}_1 \circledcirc \mathbf{s}_2 \tag{8}$$

where $\circledcirc$ is an operator (either $\oplus$: element-wise addition or $\otimes$: element-wise multiplication). The eq.(8) can be extended into a more general form – **mixture of attentions (MoA)**:

$$\mathbf{s} = \underbrace{\mathbf{r} \bullet \phi(\mathbf{h})}_{\textbf{membership}} \circledcirc \underbrace{\mathbf{A} \bullet \phi(\mathbf{A}^\top \bullet \mathbf{x})}_{\textbf{dot-product}} \tag{9}$$

Note eq.(9) entails the following transformations ($\rightarrow$) and captures the representational differences from $\mathbf{r}$ to $\mathbf{s}$ and from $\mathbf{x}$ to $\mathbf{s}$:

$$\begin{tikzcd} \mathbf{x} \arrow[r] \arrow[drr] & \mathbf{r} \arrow[r] \arrow[rr, bend right] & \mathbf{p}_1 \arrow[r] & \mathbf{s}_1 \arrow[r] & \mathbf{s} \\ & & \mathbf{p}_2 \arrow[r] & \mathbf{s}_2 \arrow[ur] & \end{tikzcd} \quad \begin{matrix} (10) \\ (11) \end{matrix}$$

The $\phi(\cdot)$ is either $\textsc{softmax}$: $p_i = \frac{\exp(z_i)}{\sum_j^k \exp(z_j)}$ that deterministically computes slice distributions or a Monte-Carlo gradient estimator: $\textsc{gumbel-softmax}$ (Gumbel, 1954; Jang et al., 2017; Maddison et al., 2017):

$$p_i = \frac{\exp[(\log(z_i) + \pi_i)/\tau]}{\sum_j^k \exp[\log(z_j) + \pi_j)/\tau]} \tag{12}$$

The π_i are i.i.d. samples from the $\textsc{Gumbel}(0, 1)$, that is, $\pi = -\log(-\log(u)), u \sim \textsc{uniform}(0, 1)$. τ is temperature which controls the concentration of slice distribution, and small τ leads to more confident prediction over slices. It aims to stochastically compute slice distribution. With Gumbel-softmax, the slice distribution is a *soft* sampling from:

$$\mathbf{p}_1 \sim \textsc{Gumbel-Softmax}(\mathbf{h}) \tag{13}$$

$$\mathbf{p}_2 \sim \textsc{Gumbel-Softmax}(\mathbf{A}^\top \cdot \mathbf{x}) \tag{14}$$

or a *hard* sampling (but differentiable) from:

$$\mathbf{p}_1 \sim \textsc{One-hot}(\arg\max(\mathbf{p}_1)) \tag{15}$$

$$\mathbf{p}_2 \sim \textsc{One-hot}(\arg\max(\mathbf{p}_2)) \tag{16}$$

for membership and dot-product attention respectively.

4 Experiments

We performed our experiments on a binary classification task with linguistic acceptability and on a multi-class classification task with intent detection.

4.1 Experimental Setup

Datasets and Metrics. The CoLA (Warstadt et al., 2018) dataset has 8551 train and 527 development in domain samples[5]. We randomly split it into

---

[4] In multi-class case, only membership likelihood is used.

[5] `https://nyu-mll.github.io/CoLA/`

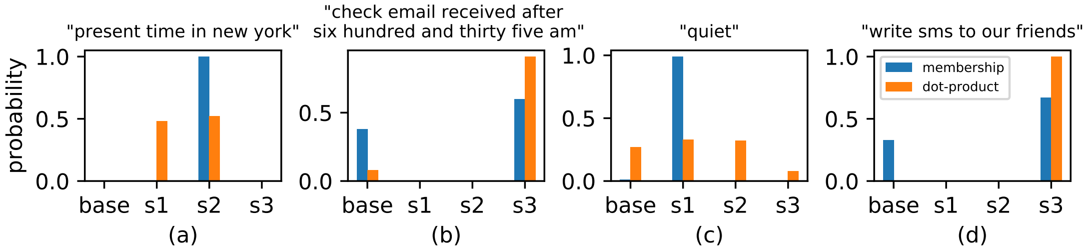

Figure 2: Distributions over slices (base, $s1$= Length, $s2$=Time and $s3$=Email) of random test samples that are from membership and dot-product attention mechanisms. In (a)(c) membership attention shows higher confidence while dot-product attention gives higher confidence in (b)(d). Top rows are the utterances from the test set.

	F1			F1 Lift(%)		MCC			MCC Lift(%)	
Methods	Overall	S1	S2	Avg.	Max.	Overall	S1	S2	Avg.	Max.
Baseline	0.70	0.60	0.65	—	—	0.24	0.18	0.22	—	—
SBL	0.69	0.71	**0.72**	**9.0 %**	11.0%	0.23	0.20	0.24	2.0%	2.0%
SBL-MoA ⊕	0.70	0.71	0.68	7.0%	11.0%	0.25	0.24	0.20	2.0%	6.0%
SBL-MoA ⊗	0.69	**0.72**	0.71	**9.0%**	**12.0%**	0.24	0.24	0.25	4.5%	6.0%
SBL-MoA-S ⊕	0.69	0.67	0.68	5.0%	7.0%	0.24	0.25	0.18	1.5%	7.0%
SBL-MoA-S ⊗	0.69	0.69	0.71	7.5%	9.0%	0.26	**0.28**	0.22	5.0%	**10.0%**
SBL-MoA-H ⊕	0.70	0.69	0.70	7.0%	9.0%	0.25	**0.28**	0.26	**8.0%**	**10.0%**
SBL-MoA-H ⊗	0.69	0.69	0.65	4.5%	9.0%	0.25	0.22	**0.32**	7.0%	**10.0%**

Table 2: The results on CoLA test datasets. F1-score and MCC are reported (averaged on 5 random runs for each model). $s1$=Long, and $s2$=Question.The *lift* is the averaged relative improvement across slices over baseline. The largest improvement is in **bold** and second largest lift number is in underline (same to Table 3).

train/val/test with 7200/878/1000 samples. As in (Chen et al., 2019), we ensure the sample proportion in ground-truth are consistent across splits. We use F1-score and Matthews correlation coefficient (MCC) (Matthews, 1975) as our metrics. The NLU dataset (Liu et al., 2019) for intent detection contains 25k user utterances across 64 intents. We randomly split it into train/val/test with ratio 0.7:0.1:0.2. We use the accuracy and F1-score as our metrics.

Compared Methods. We implemented and compared the following methods:

- **Baseline**: A three-layer feed-forward network.
- **SBL**: Slice-based learning (Chen et al., 2019).
- **SBL-MoA**: Our approach that extends SBL with a mixture of attentions (MoA).

For SBL-MoA, we developed multiple variants with Gumbel-Softmax. SBL-MoA-S (SBL-MoA-H) are the variant models with soft (hard) sampling from a Gumbel-Softmax distributions. We also tested the way that membership attention and dot-product interact with each other with ⊕ (element-wise addition) and ⊗ (element-wise multiplication).

Implementation Details. BERT-base (Devlin et al., 2019) in sentence-transformer (Thakur et al., 2020) is used as the backbone model. We use 128 hidden units for all models, which are implemented with Pytorch (Paszke et al., 2019). A dropout (p=0.5)[6] is applied after input layer. The models are trained with Adam (0.001) (Kingma and Ba, 2014), with weight decay of 0.01 and 0.001 for the two tasks, respectively. All models are trained with a maximum of 500 epochs with early stopping (patience=50). The best models are selected based on model performance on the validation sets. The temperature $\tau = 1.0$ is fixed in all the experiments.

4.2 Results on Linguistic Acceptability

Table 2 presents the results on CoLA. First, slice-based models (i.e., SBL, SBL-MoA, and its variants) show that they can maintain (or improve) the original overall performance. Second, we observe that they achieve obvious performance lift on the monitored slices. For instance, SBL achieves an average 9% F1 score over the baseline. The proposed method (SBL-MoA, ⊗) achieves an average of 9% and maximum 12% lift. For MCC, the best performer is SBL-MoA-H, which achieves an average $\geqslant$7% and maximum 10% as compared to the

---

[6] As the data size is relatively small, we use strong dropout regularization to prevent overfitting.

	Acc				Acc Lift(%)		F1				F1 Lift(%)	
Methods	Overall	S1	S2	S3	Avg.	Max.	Overall	S1	S2	S3	Avg.	Max.
Baseline	0.7413	0.73	0.74	0.73	—	—	0.7404	0.74	0.72	0.74	—	—
SBL	0.7422	0.75	0.76	0.75	2.0%	2.0%	0.7418	0.74	0.72	0.75	0.3%	1.0%
SBL-MoA ⊕	0.7414	0.74	**0.77**	0.74	1.7%	3.0%	0.7390	0.74	0.73	0.74	0.3%	1.0%
SBL-MoA ⊗	0.7440	**0.80**	0.73	**0.76**	**3.0%**	**7.0%**	0.7411	**0.77**	**0.74**	0.74	1.7%	**3.0%**
SBL-MoA-S ⊕	0.7403	0.74	0.75	0.73	0.7%	1.0%	0.7403	0.73	0.73	**0.76**	0.7%	2.0%
SBL-MoA-S ⊗	0.7424	0.75	0.72	0.75	0.7%	2.0%	0.7421	0.75	**0.74**	0.74	1.0%	2.0%
SBL-MoA-H ⊕	0.7405	0.73	0.74	0.73	0.0%	0.0%	0.7397	0.76	**0.74**	**0.76**	**2.0%**	2.0%
SBL-MoA-H ⊗	0.7418	0.74	0.73	0.75	0.7%	2.0%	0.7401	0.75	**0.74**	0.74	1.0%	2.0%

Table 3: The results on intent detection. Accuracy and F1 scores are reported. $s1$=Length, $s2$=Time, $s3$=Email are the slices that we monitor and aim to improve. The experts' confidence scores are not used as discussed in Sec.3.

baseline. It outperforms SBL by $\geqslant 5\%$. Also, we notice that using operator $\otimes$ (element-wise multiplication) between the attention mechanisms lead to better performance as compared to $\oplus$.

4.3 Results on Intent Detection

Table 3 demonstrates that both SBL and SBL-MoA improve model performance on the monitored slices, with a similar (slightly better) overall performance on the base task[7]. SBL-MoA variants achieve the best scores and outperform SBL by average 1% accuracy and 1.7% F1.

Figure 2 illustrates the slice distributions given some random samples. We denote $\mathbf{p}_1$ and $\mathbf{p}_2$ for membership and dot-product attention respectively. The experiments show that $\mathbf{p}_1$ and $\mathbf{p}_2$ reach an agreement on predicting the correct slices. Interestingly, the sample in (d) — "write sms to our friends", in principle, should be sliced as "base", but both attentions exhibit high confidence to $s3$="Email". We conjecture the reason is that all utterances are encoded with BERT which captures the similarity between the sample and the utterances in the "Email" slice.

5 Related Work

SBL (Chen et al., 2019) is a novel programming model for critical data slices. It is an instance of weakly supervised learning (Zhou, 2018; Medlock and Briscoe, 2007). The weak supervision data are generated from pre-defined labeling functions (Ratner et al., 2016). SBL has shown better predictive performance compared to the mixture of experts (Jacobs et al., 1991) and multi-task learning (Caruana, 1997), with reduced run-time cost and parameters (Chen et al., 2019). The concept of SBL has been recently used in many applications. Penha et al. (Penha and Hauff, 2020) proposed to adapt SBL to improve ranking performance and capture the failures of the ranker model. Wang et al. (Wang et al., 2021) recently implemented SBL in a commercial conversational AI system in order to handle the long-tail problem of imbalanced distribution in customer queries and further improved the performance of the conversational skill routing components (Li et al., 2021; Kim et al., 2018b,a).

Our proposed mixture of attention (MoA) is an instance of multi-head attention (Vaswani et al., 2017) but with different attention types. MoA can also be extended to include other attention types. We have shown the effectiveness of this mechanism in determining the slice distributions.

6 Conclusion

This paper extends SBL with MoA (SBL-MoA) to improve model performance on particular data slices. We empirically show that SBL-MoA yields better slice level performance lift to baseline and vanilla SBL with two NLU tasks: linguistic acceptability and intent detection.

---

[7] Note the lift on slice can be negligible to overall due to small size of slice data, e.g., For SBL-MoA $\otimes$, $s1$ with 122 samples, 7.0% lift only contributes to $122\times0.07/5124 \approx 0.0017$.

References

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. 2019. Slice-based learning: A programming model for residual learning in critical data slices. In *Advances in neural information processing systems*, pages 9397–9407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Emil Julius Gumbel. 1954. Statistical Theory of Extreme Values and Some Practical Applications. A Series of Lectures. *Number 33. US Govt. Print. Office*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Najoung Kim, Roma Patelφ, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019. Probing what different nlp tasks teach machines about function word comprehension. *NAACL HLT 2019*, page 235.

Young-Bum Kim, Dongchan Kim, Joo-Kyung Kim, and Ruhi Sarikaya. 2018a. A scalable neural shortlisting-reranking approach for large-scale domain classification in natural language understanding. *arXiv preprint arXiv:1804.08064*.

Young-Bum Kim, Dongchan Kim, Anjishnu Kumar, and Ruhi Sarikaya. 2018b. Efficient large-scale neural domain classification with personalized attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2214–2224.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Han Li, Sunghyun Park, Aswarth Dara, Jinseok Nam, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. 2021. Neural model robustness for skill routing in large-scale conversational ai systems: A design choice exploration. *arXiv preprint arXiv:2103.03373*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2897–2903.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. In *10th International Workshop on Spoken Dialogue Systems Technology 2019*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *5th International Conference on Learning Representations (ICLR)*.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Ben Medlock and Ted Briscoe. 2007. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 992–999.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Gustavo Penha and Claudia Hauff. 2020. Slice-aware neural ranking. In *Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI)*, pages 1–6.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567–3575.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2020. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Cheng Wang, Sun Kim, Taiwoo Park, Sajal Choudhary, Sunghyun Park, Young-Bum Kim, Ruhi Sarikaya, and Sungjin Lee. 2021. Handling long-tail queries with slice-aware conversational systems. *arXiv preprint arXiv:2104.13216*.

Cheng Wang and Mathias Niepert. 2019. State-regularized recurrent neural networks. In *International Conference on Machine Learning*, pages 6596–6606.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Zhi-Hua Zhou. 2018. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53.

Making Better Use of Bilingual Information for Cross-Lingual AMR Parsing

Yitao Cai, Zhe Lin and **Xiaojun Wan**
Wangxuan Institute of Computer Technology, Peking University
Center for Data Science, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
`{caiyitao,linzhe,wanxiaojun}@pku.edu.cn`

Abstract

Abstract Meaning Representation (AMR) is a rooted, labeled, acyclic graph representing the semantics of natural language. As previous works show, although AMR is designed for English at first, it can also represent semantics in other languages. However, they find that concepts in their predicted AMR graphs are less specific. We argue that the misprediction of concepts is due to the high relevance between English tokens and AMR concepts. In this work, we introduce bilingual input, namely the translated texts as well as non-English texts, in order to enable the model to predict more accurate concepts. Besides, we also introduce an auxiliary task, requiring the decoder to predict the English sequences at the same time. The auxiliary task can help the decoder understand what exactly the corresponding English tokens are. Our proposed cross-lingual AMR parser surpasses previous state-of-the-art parser by 10.6 points on Smatch F1 score. The ablation study also demonstrates the efficacy of our proposed modules.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a rooted, labeled, acyclic graph representing sentence-level semantic of text. Nodes in the graph are concepts in the texts and edges in the graph are relations between concepts. Since AMR abstracts away from syntax and preserves only semantic information, it can be applied to many semantic related tasks such as summarization (Liu et al., 2015; Liao et al., 2018), paraphrase detection (Issa et al., 2018), machine translation (Song et al., 2019) and so on.

Previous works on AMR parsing mainly focus on English, since AMR is designed for English texts and parallel corpus of non-English texts and AMRs are scarce. Early work of AMR announces that AMR is biased towards English and is not an interlingua (Banarescu et al., 2013). Besides, some studies show that aligning AMR with non-English language is not always possible (Xue et al., 2014; Hajic et al., 2014). However, recent studies (Damonte and Cohen, 2018; Blloshmi et al., 2020) show that AMR parsers are able to recover AMR structures when there are structural differences between languages, which demonstrate that it is capable to overcome many translation divergences. Therefore, it is possible for us to parse texts in target (non-English) languages into AMRs.

Another problem of cross-lingual AMR parsing is the scarcity of parallel corpus. Unlike machine translation or sentiment classification which have abundant resources on the Internet, we can only get non-English text and AMR pairs by human annotation. Damonte and Cohen (2018) align a non-English token with an AMR node if they can be mapped to the same English token to construct training set. They further train a transition-based parser using the synthetic training set. They also attempt to translate test set into English and apply an English AMR parser. Blloshmi et al. (2020) build training data in two ways. One of the approaches is that they use gold parallel sentences and generate synthetic AMR annotations with the help of an English AMR parser. Another approach is to use gold English-AMR pairs and get non-English texts by a pre-trained machine translation system. They further use a sequence-to-graph parser (Zhang et al., 2019a) to train a cross-lingual AMR parser.

According to (Blloshmi et al., 2020), a cross-lingual AMR parser may predict the concepts less specific and accurate than the gold concepts. Therefore, we propose a new model introducing machine translation to enable our parser to predict more accurate concepts. In particular, we first build our training data similar to (Blloshmi et al., 2020), translating English texts into target languages. Our basic model is a sequence-to-sequence

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1537–1547
August 1–6, 2021.

model, rather than the sequence-to-graph model used in (Blloshmi et al., 2020), since in English AMR parsing, sequence-to-sequence models can achieve state-of-the-art result with enough data for pre-training (Xu et al., 2020). While training, we introduce bilingual input by concatenating translated target language texts and English texts as inputs. As for inference stage, the bilingual input is the concatenation of translated English texts and target language texts. We hope that our model can predict more accurate concepts with the help of the English tokens, while it can still preserve the meaning of the original texts if there are semantic shifts in the translated English texts. Besides, during training process, we also introduce an auxiliary task, requiring the decoder to restore English input tokens, which also aims at enhancing the ability of our parser to predict concepts. Our parser outperforms previous state-of-the-art parser XL-AMR (Blloshmi et al., 2020) on LDC2020T07 dataset (Cai and Knight, 2013) by about 10.6 points of Smatch F1 score on average, which demonstrates the efficacy of our proposed cross-lingual AMR parser.

Our main contributions are summarized as follows:

- We introduce bilingual inputs and an auxiliary task to a seq2seq cross-lingual AMR parser, aiming to enable the parser to make better use of bilingual information and predict more accurate concepts.
- Our parser surpasses the best previously reported results of Smatch F1 score on LDC2020T07 by a large margin. The results demonstrate the effectiveness of our parser. Ablation studies show the usefulness of the model modules. Codes are public available [1].
- We further carry out experiments to investigate the influence of incorporating pretraining models into our cross-lingual AMR parser.

2 Related Work

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) parsing is becoming popular recently. Some of previous works (Flanigan et al., 2014; Lyu and Titov, 2018; Zhang et al., 2019a) solve this problem with a two-stage approach. They first project words in sentences to AMR concepts, followed by relation identification. Transition-based parsing is applied by (Wang et al., 2015b,a; Damonte et al., 2017; Liu et al., 2018; Guo and Lu, 2018; Naseem et al., 2019; Lee et al., 2020). They align words with AMR concepts and then take different actions based on different processed words to link edges or insert new nodes. Due to the recent development in sequence-to-sequence model, several works employ it to parse texts into AMRs (Konstas et al., 2017; van Noord and Bos, 2017; Ge et al., 2019; Xu et al., 2020). They linearize AMR graphs and leverage character-level or word-level sequence-to-sequence model. Sequence-to-graph model is proposed to enable the decoder to better model the graph structure. Zhang et al. (2019a) first use a sequence-to-sequence model to predict concepts and use a biaffine classifier to predict edges. Zhang et al. (2019b) propose a one-stage sequence-to-graph model, predicting concepts and relations at the same time. Cai and Lam (2020) regard AMR parsing as dual decisions on input sequences and constructing graphs. They therefore propose a sequence-to-graph method by first mapping an input words to a concept and then linking an edge based on the generated concepts. Recently, pre-training models have been proved to perform well in AMR parsing (Xu et al., 2020). Lee et al. (2020) employ a self-training method to enhance a transition-based parser, which achieves the state-of-the-art Smatch F1 score in English AMR parsing.

Vanderwende et al. (2015) first carry out research of cross-lingual AMR parsing. They parse texts in target language into logical forms as a pivot, which are then parsed into AMR graphs. Damonte and Cohen (2018) attempt to project non-English words to AMR concepts and use a transition-based parser to parse texts to AMR graphs. They also attempt to automatically translate non-English texts to English and exploit an English AMR parser. Blloshmi et al. (2020) try to generate synthetic training data by a machine translation system or an English AMR parser. They conduct experiments with a sequence-to-graph model in different settings, trying to find a best way to train with synthetic training data. Different from (Blloshmi et al., 2020), we treat cross-lingual AMR parsing as a sequence-to-sequence transduction problem and improve seq2seq models with bilingual input and auxiliary task.

---

[1] https://github.com/headacheboy/cross-lingual-amr-parsing

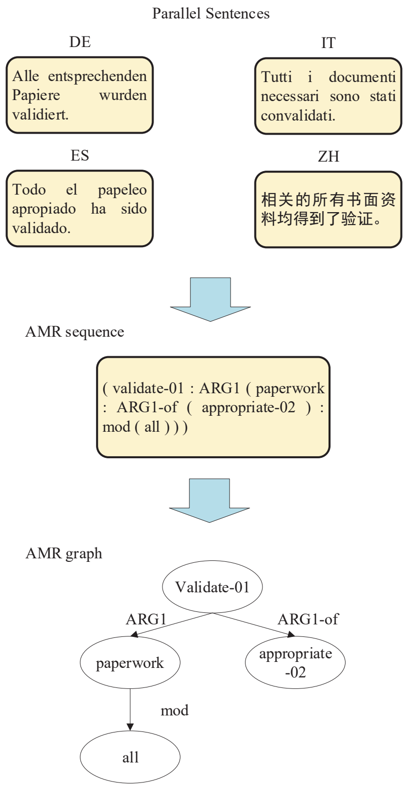

Figure 1: An example of cross-lingual AMR parsing. A Non-English text is first parsed into an AMR sequence and then the sequence is converted to an AMR graph.

3 Problem Setup

Cross-lingual AMR parsing is the task of parsing non-English texts into AMR graphs corresponding to their English translation. In this task, nodes in AMR graphs are still English words, PropBank framesets or AMR keywords, which are the same as the original design of AMR.

Figure 1 shows an example of cross-lingual AMR parsing. We define X^l as an input sample in language l and X^l_i is the i-th token of it. y is the corresponding AMR sequence derived from the AMR graph, and y_i is the i-th token. The model should predict the AMR sequence y first and then transform the sequence into a graph.

4 Our Proposed Model

Figure 2 shows the training and inference processes of our proposed model. The basic model we adopt is Transformer (Vaswani et al., 2017) encoder-decoder model, since Xu et al. (2020) show that it can achieves state-of-the-art result in English AMR parsing. We introduce the bilingual input to our model. When training the model, the bilingual input contains original English text and translated text in non-English target language, which may not be very accurate. During inference, the bilingual input is composed of translated English text and original text in target language. With the help of bilingual input, our model can better understand and preserve the semantics of target language texts, and predict more accurate concepts according to the translated English texts. Apart from predicting AMR sequences, the model is also required to predict the English input texts as an auxiliary objective, which can further help the model learn the exact meaning of input tokens and predict their corresponding concepts more accurately.

We will first introduce the way we obtain training data and the pre-processing and post-processing process in Section 4.1 and Section 4.2, followed by introducing the basic sequence-to-sequence model, the bilingual input and the auxiliary task.

4.1 Synthetic Training Data

Blloshmi et al. (2020) propose two methods to generate parallel training data, namely **parallel sentences - silver AMR graphs** and **gold AMR graphs - silver translations**. The first approach means that we exploit human annotated parallel corpus of target languages and English and use an English parser to get the corresponding AMR graphs. The second approach means that we exploit human annotated English-AMR pairs and use a machine translation system to get texts in target languages. According to (Blloshmi et al., 2020), model training with data generated by gold English-AMR pairs performs better. We thus exploit this approach (i.e., **gold AMR graphs - silver translations**) to generate our data for training and validation.

4.2 Pre-Processing and Post-Processing

Following (van Noord and Bos, 2017), we first remove variables, since variables are only used to identify the same node in a graph and contain no semantic information, which may do harm to the model training process. We also remove wiki links (:wiki), since sequence-to-sequence model may link to non-existing objects of Wikipedia. As for co-referring nodes, we simply duplicate the concepts. It transforms an AMR graph into a tree. The final linearized AMR is the pre-order traversal of the tree.

In post-processing, we should restore a predicted AMR sequence without variables, wiki links and co-referring nodes to a AMR graph. Following

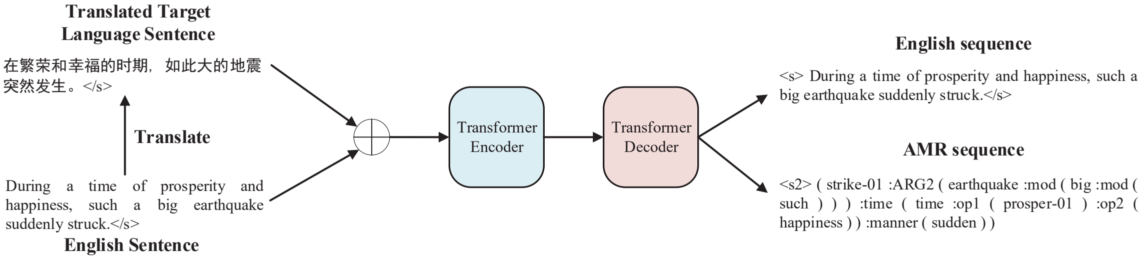

在繁荣和幸福的时期，如此大的地震突然发生。</s>

During a time of prosperity and happiness, such a big earthquake suddenly struck.</s>

<s> During a time of prosperity and happiness, such a big earthquake suddenly struck.</s>

<s2> (strike-01 :ARG2 (earthquake :mod (big :mod (such))) :time (time :op1 (prosper-01) :op2 (happiness)) :manner (sudden))

(a) Model Training

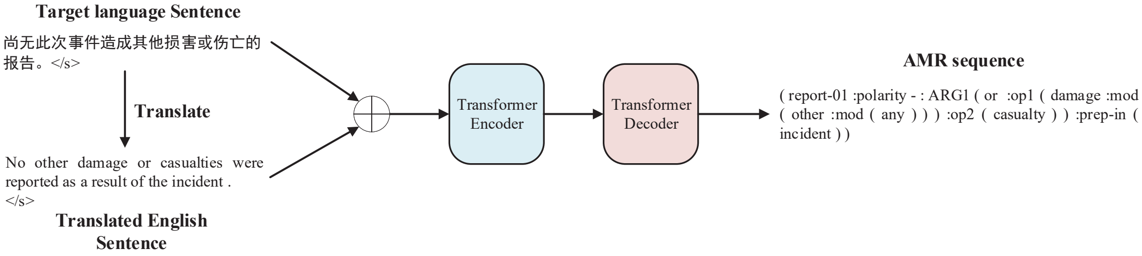

尚无此次事件造成其他损害或伤亡的报告。</s>

No other damage or casualties were reported as a result of the incident . </s>

(report-01 :polarity - : ARG1 (or :op1 (damage :mod (other :mod (any))) :op2 (casualty)) :prep-in (incident))

(b) Model Inference

Figure 2: Overview of our proposed model. ⊕ is the concatenation operation.

	Training	Development	Test
Number	36521	1368	1371

Table 1: Statistics of gold English-AMR dataset

(van Noord and Bos, 2017), We first restore variables and prune duplicated nodes, which brings co-reference back to the AMR sequence. van Noord and Bos (2017) use DBpedia Spotlight to restore wiki links. However, same entity in different language is linked to different pages in DBpedia, which makes it difficult for cross-lingual AMR parser to restore the wiki linking the entity in English. Different from (van Noord and Bos, 2017), we restore a wiki link of a certain name if this name corresponds to the wiki link in training set.

4.3 Sequence-to-Sequence Model

After pre-processing, both input texts and output AMRs are sequences. Hence we are able to apply a sequence-to-sequence model to accomplish cross-lingual AMR parsing. We use Transformer (Vaswani et al., 2017), one of the most popular sequence-to-sequence model as our basic model.

In order to be compatible with pre-training XLM-R (Conneau et al., 2020a) model, the tokenizer and input vocabulary we used is the same as XLM-R. Subword unit such as byte pair encoding (BPE) (Sennrich et al., 2016) is commonly used to reduce the size of vocabulary. Thus, we exploit BPE to get our output vocabulary.

4.4 Bilingual Input

AMR concepts heavily rely on the corresponding English texts. According to (Damonte and Cohen, 2018), a simple method that first translates the test set into English and then applies an English AMR parser can outperform their cross-lingual AMR parser. However, machine translation may introduce semantic shifts, which may do harm to the generation of AMRs.

We therefore introduce the bilingual input. Since we do not have gold parallel corpus, we use machine translation to get the bilingual input. During training, we concatenate the translated text in target language mentioned in Section 4.1 and the original English text as bilingual input. At the inference stage, we take the bilingual input by concatenating original target language texts and the translated English text. The model can better understand and preserve the semantic meanings of the input bilingual text. It can also predict more correct concepts, since the English tokens are also provided.

4.5 Auxiliary Task

AMR concepts are composed of English words and Propbank frames. According to (Blloshmi et al., 2020), roughly 60% of nodes in AMR 2.0 (LDC2017T10) are English words. What's more, Propbank predicates are similar to English words, such as predicate *publish-01* and word *publish*. We argue that if the decoder can restore the input tokens in English precisely, it can predict the corresponding concepts appropriately.

We thus design an auxiliary task, requiring the decoder to predict the English input sequence. Inspired by multilingual machine translation (Johnson et al., 2017), we add a new BOS token indicating that the model should predict the English sequences instead of AMR sequences. The decoder predicting English sequences share the same weights as the decoder predicting AMR sequences.

The final loss function is the weighted sum of loss functions of these two tasks: $Loss_{AMR}$ - the loss of AMR sequence prediction and $Loss_{Eng}$ - the loss of English sentence prediction. We adopt the cross-entropy loss for both tasks.

5 Implementation Details

The coefficient of $Loss_{AMR}$ is 1, while the coefficient of $Loss_{Eng}$ is 0.5. We use Adam optimizer (Kingma and Ba, 2015) to optimize the final loss function. The number of transformer layers in both encoder and decoder is 6. The embedding size and hidden size are both 512 and the size of feed-forward network is 2048. The head number of multi-head attention is 8. We follow (Vaswani et al., 2017) to tune the learning rate each step and the warmup step is 4000. The learning rate for decoder at each step is half of this learning rate. Following (Blloshmi et al., 2020), we use machine translation system OPUS-MT (Tiedemann and Thottingal, 2020) to get our bilingual input. We use all data from different languages to train our model and the final model is able to parse sentences in different languages.

6 Experiments

6.1 Dataset

The released test set, LDC2020T07, contains four translations of test set of AMR 2.0, including German (DE), Italian (IT), Spanish (ES) and Chinese(ZH).

As is mentioned in Section 4.1, we translate the sentences in a gold English-AMR dataset to get training and development data with OPUS-MT. We use AMR 2.0 as our gold English-AMR dataset. We also translate test sets in German, Italian, Spanish and Chinese back to English as input texts. The statistics of AMR 2.0 are shown in Table 1.

6.2 Evaluation Metric

Smatch (Cai and Knight, 2013) is the evaluation metric of AMR parsing. In this evaluation metric, AMR graph is regarded as several triples. Smatch counts the numbers of matched triples and outputs the score based on total numbers of triples of two AMR graphs. We use the Smatch scripts available online [2].

	DE	IT	ES	ZH	AVG
AMREager	39.0	43.0	42.0	35.0	39.8
XL-AMR (Mul)	49.9	53.5	53.2	41.0	49.4
XL-AMR (Mul*)	52.1	56.7	56.2	-	-
XL-AMR (Lang)	51.6	56.7	56.1	43.1	51.9
XL-AMR (Bi)	53.0	58.1	58.0	41.5	52.7
Translate-test	60.4	62.1	63.3	53.7	59.9
Ours	**64.0**	**65.4**	**67.3**	**56.5**	**63.3**

Table 2: Smatch F1 scores of different models on German (DE), Italian (IT), Spanish (ES) and Chinese (ZH). (Mul) represents multilingual setting. (Mul*) represents multilingual setting except Chinese data. (Lang) represents language specific setting. (Bi) represents bilingual setting. The Smatch F1 score of English AMR parser is 68.3. Results of the best models are in bold.

Following (Damonte et al., 2017), we also introduce many fine-grained evaluations in order to evaluate the quality of the predicted AMR graphs in different aspects. We omit the details of these fine-grained evaluations here, which can be found in (Damonte et al., 2017).

6.3 Main Results

We compare our model with previous works and baseline methods including:

- **AMREager**. This is the model proposed by Damonte and Cohen (2018). They assume that if a word X_t^l in target language is aligned with the word X_u^{en} in English and the English word aligns with AMR concept y_i, X_t^l can be aligned with y_i. Based on this assumption, they project AMR annotations to target languages and further train a transition-based AMR parser (Damonte et al., 2017) as in English.

- **XL-AMR**. This is the model proposed by (Blloshmi et al., 2020). When conducting experiments of their best model, they first generate synthetic training and validation data by machine translation. They then train an AMR parser on target language with a sequence-to-graph parser. They experiment XL-AMR

---

[2] https://github.com/sheng-z/stog

	AMREager				XL-AMR				Ours			
Metric	DE	IT	ES	ZH	DE	IT	ES	ZH	DE	IT	ES	ZH
Smatch	39.1	43.2	42.1	34.6	53.0	58.1	58.0	43.1	**64.0**	**65.4**	**67.3**	**56.5**
Unlabeled	45.0	48.5	46.6	41.1	57.7	63.4	63.0	48.9	**68.1**	**69.6**	**71.2**	**61.0**
No WSD	39.2	42.5	42.2	34.7	53.2	58.4	58.4	43.2	**64.4**	**65.9**	**67.8**	**56.7**
Reentrancies	18.6	25.7	27.2	15.9	39.9	46.1	46.6	34.7	**47.9**	**49.3**	**51.3**	**41.4**
Concepts	44.9	52.3	53.3	39.9	58.0	64.7	65.9	48.0	**69.3**	**72.1**	**75.0**	**61.3**
Named Ent.	63.1	67.7	65.7	67.9	66.0	70.0	66.2	60.6	**79.3**	**79.5**	**80.2**	**76.2**
Wikification	49.9	50.6	44.5	46.8	60.9	67.0	63.1	54.5	**74.0**	**74.9**	**73.9**	**68.1**
Negation	18.6	22.3	19.8	6.8	11.7	29.2	23.4	12.8	**47.1**	**52.6**	**55.6**	**36.6**
SRL	29.4	34.3	35.9	27.2	47.9	54.7	55.2	41.3	**57.3**	**60.1**	**62.1**	**50.7**

Table 3: Fine-grained results of different models on DE, IT, ES and ZH. Best results are in bold.

	S2S	S2S + Bilingual Input		S2S + Auxiliary		Full Model	
	F1	F1	Δ	F1	Δ	F1	Δ
Smatch	53.1	57.5	4.4	58.6	5.5	63.3	10.2
Unlabeled	57.7	59.2	1.5	58.9	1.2	67.5	9.8
No WSD	53.4	59.2	5.8	58.9	5.5	63.7	10.3
Reentrancies	38.4	41.9	3.5	42.7	4.3	47.5	9.1
Concepts	57.3	66.4	9.1	62.7	5.4	69.4	12.1
Named Ent.	73.7	75.0	1.3	77.2	3.5	78.8	5.1
Wiki	62.1	69.0	6.9	69.2	7.1	72.7	10.6
Negation	32.9	44.1	11.2	39.5	6.6	48.0	15.1
SRL	47.4	52.1	4.7	52.2	4.8	57.6	10.2

Table 4: Smatch and fine-grained results of ablation study. The listed Scores are avereage F1 score for different metrics on four test sets. Δ represents the model improvement compared with basic sequence-to-sequence(s2s) model.

with many different settings: language specific setting, bilingual setting and multilingual setting. Language specific setting means that they only use target language data to train the model. Bilingual setting represents training with target language data and English data. Multilingual setting represents training with data in all languages. They also experiment multilingual setting except Chinese data because they found training with Chinese data will lower the results.

- **Translate-test**. This method first translates target language texts into English and uses an English AMR parser to predict the final AMR graphs. For fair comparison, we choose the sequence-to-sequence model as the English AMR parser. The encoder, decoder and hyperparameters of these modules are the same as those in our model. We use only English texts as input and do not apply the auxiliary task in the training of English parser. Note that this baseline is not compared in (Blloshmi et al., 2020) and we show it is a very strong baseline.

The comparison results are shown in Table 2. Our model outperforms previous best model XL-AMR in different settings by a large margin. As for languages that share similarity with English, namely German, Italian, Spanish, our proposed model achieves substantial improvement on Smatch F1 score by about 10 points. When it comes to languages that has linguistic differences with English, namely Chinese, our model performs better, surpassing XL-AMR by 13.4 points on Smatch F1 score. The Translate-test method is a strong baseline because of the quality of machine translation. It outperforms previous reported results by a large margin, which reveals that English information is significantly beneficial to AMR prediction. In this work, our model also surpasses this method by 3.4 points and achieves the new state-of-the-art results.

Table 3 lists the fine-grained evaluation results of AMREager, the best XL-AMR model and our model. Our proposed model achieves substantially higher performance by about 10 points for each fine-grained task except *Negation*. As for *Negation*, our model achieves over 20 points higher than XL-AMR. These results demonstrates that our model not only predicts better concepts but also predicts

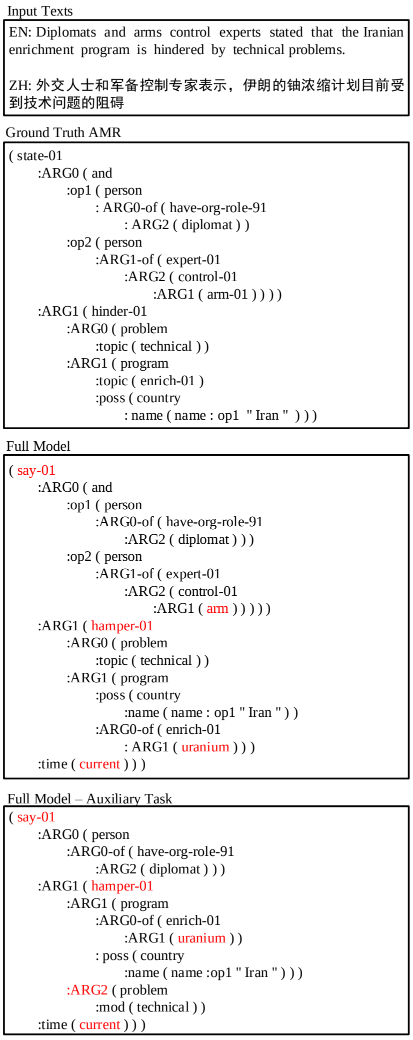

Input Texts

EN: Diplomats and arms control experts stated that the Iranian enrichment program is hindered by technical problems.

ZH: 外交人士和军备控制专家表示，伊朗的铀浓缩计划目前受到技术问题的阻碍

Ground Truth AMR

```
( state-01
      :ARG0 ( and
            :op1 ( person
                  : ARG0-of ( have-org-role-91
                        : ARG2 ( diplomat ) )
            :op2 ( person
                  :ARG1-of ( expert-01
                        :ARG2 ( control-01
                              :ARG1 ( arm-01 ) ) ) ) )
      :ARG1 ( hinder-01
            :ARG0 ( problem
                  :topic ( technical ) )
            :ARG1 ( program
                  :topic ( enrich-01 )
                  :poss ( country
                        : name ( name : op1  " Iran "  ) ) )
```

Full Model

```
( say-01
      :ARG0 ( and
            :op1 ( person
                  :ARG0-of ( have-org-role-91
                        :ARG2 ( diplomat ) ) )
            :op2 ( person
                  :ARG1-of ( expert-01
                        :ARG2 ( control-01
                              :ARG1 ( arm ) ) ) ) ) )
      :ARG1 ( hamper-01
            :ARG0 ( problem
                  :topic ( technical ) )
            :ARG1 ( program
                  :poss ( country
                        :name ( name : op1 " Iran " ) )
                  :ARG0-of ( enrich-01
                        : ARG1 ( uranium ) ) )
      :time ( current ) ) )
```

Full Model – Auxiliary Task

```
( say-01
      :ARG0 ( person
            :ARG0-of ( have-org-role-91
                  :ARG2 ( diplomat ) ) )
      :ARG1 ( hamper-01
            :ARG1 ( program
                  :ARG0-of ( enrich-01
                        :ARG1 ( uranium ) )
                  : poss ( country
                        :name ( name :op1 " Iran " ) ) )
            :ARG2 ( problem
                  :mod ( technical ) )
      :time ( current ) ) )
```

Figure 3: An example of AMR predicted by models with and without auxiliary task. We mark the error concepts and relations in red.

better relations between concepts.

6.4 Ablation Study

In order to verify the effectiveness of the bilingual input and the auxiliary task in our model, we carry out several ablation experiments.

Table 4 shows the Smatch score and fine-grained results of ablation study. Compared with the basic sequence-to-sequence model, the bilingual input can improve the Smatch F1 score by 4.4 points on average. The introduction of auxiliary task brings 5.5 points improvement of Smatch on average. Our full model makes use of both bilingual input and auxiliary task at the same time, improving Smatch scores by 10.2 points, which indicates that each module is very beneficial to the performance of our model.

Fine-grained results further demonstrate the effectiveness of our modules. As Table 4 shows, F1 score for *Concepts* improves substantially, which demonstrates that our proposed modules can actually help the parser predict more accurate concepts. Besides, fine-grained evaluation *Negation* achieves the highest improvement, revealing that our proposed modules enable the parser to understand the semantics of the input texts better.

Model	DE	IT	ES	ZH	AVG
Full Model	64.0	65.4	67.3	56.5	63.3
+ XLM-R	66.1	67.9	69.6	57.9	65.4
+ dec	64.9	66.7	68.5	57.4	64.4
+ XLM-R & dec	68.3	70.0	71.9	59.6	67.5

Table 5: Smatch scores of models employing pre-trained models.

6.5 Effect of Pre-trained Models on Cross-Lingual AMR Parsing

Recently, pre-training models on cross-lingual tasks have been proposed. Pre-training models, such as mBert (Devlin et al., 2019), XLM (Conneau and Lample, 2019), XLM-R (Conneau et al., 2020b) and mBart (Liu et al., 2020) achieve state-of-the-art results on many tasks such as machine translation, cross-lingual natural language inference and so on. In our experiment, we exploit XLM-R (Conneau et al., 2020b) as the input embeddings of the model.

When training an English AMR parser, Xu et al. (2020) first pre-train the model on large scale synthetic data and fine-tune it on gold English-AMR data. Since cross-lingual AMR parsing shares the same output formats with AMR parsing, we can employ the decoder of (Xu et al., 2020) to initialize our decoder and further finetune the cross-lingual AMR parser.

Results are listed in Table 5. The performance of our parser with XLM-R embedding improves by 2.1 points on Smatch score, while our parser finetuning pre-trained AMR decoder achieves 1.1 points improvement. We further employ both XLM-R embedding and pre-trained AMR decoder and the average Smatch score is 67.5. The results show that pre-trained cross-lingual embeddings like XLM-R as well as the pre-trained decoder can help the parser predict better AMR graphs.

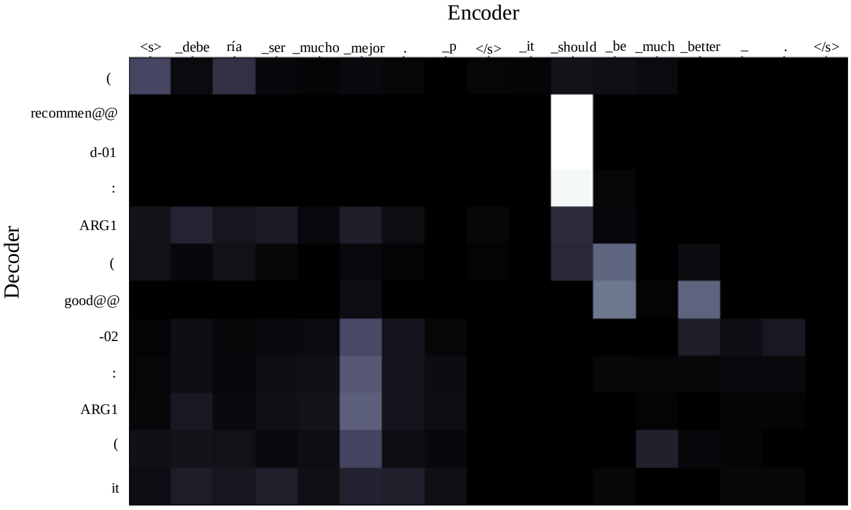

Figure 4: An attention heatmap from one head in the second decoder layer. The attention weight is higher if the color is lighter.

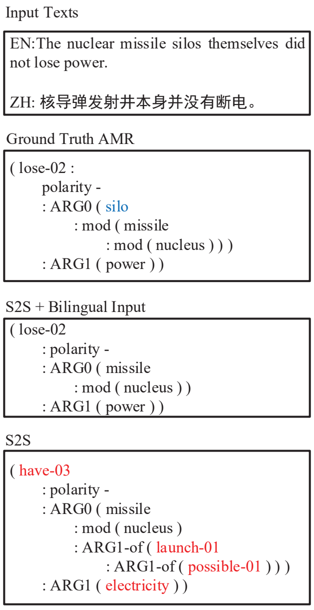

Input Texts

EN:The nuclear missile silos themselves did not lose power.

ZH: 核导弹发射井本身并没有断电。

Ground Truth AMR

```
( lose-02 :
      polarity -
      : ARG0 ( silo
            : mod ( missile
                  : mod ( nucleus ) ) )
      : ARG1 ( power ) )
```

S2S + Bilingual Input

```
( lose-02
      : polarity -
      : ARG0 ( missile
            : mod ( nucleus ) )
      : ARG1 ( power ) )
```

S2S

```
( have-03
      : polarity -
      : ARG0 ( missile
            : mod ( nucleus )
            : ARG1-of ( launch-01
                  : ARG1-of ( possible-01 ) ) )
      : ARG1 ( electricity ) )
```

Figure 5: An example of AMR predicted by S2S and S2S + Bilingual Input. We mark the missing concept in blue and mark the inaccurate concepts in red.

7 Analysis

Figure 3 shows several AMRs parsed by models with and without auxiliary task. The AMR predicted by model without auxiliary task misses many concepts, while our full model predicts them correctly. What's more, our full model can predict relations of concepts more accurately as well. For example, the full model adds *ARG0* between *hamper-01* and *problem*, retaining semantic information of the original sentence. The model trained without auxiliary task predicts the relation *ARG2* instead, changing the meaning of the original sentence.

Another example in Figure 5 shows the efficacy of bilingual inputs. The AMR parsed by basic sequence-to-sequence model does not contain correct semantics. This model predicts many erroneous concepts such as *launch-01*, *possible-01*. Besides, the semantics of original sentence *did not lose power* is changed into *have no electricity*. The AMR produced by sequence-to-sequence model with bilingual input is almost correct except missing of concept *silo*. This example reveals that our bilingual input enables the parser to predict more accurate concepts and preserve the semantics of the sentence.

We also show an attention heatmap of an example in test set in Figure 4. This attention pattern shows that our parser can predict AMR tokens based on English translation (e.g. *recommend-01*) and based on both English and Spanish tokens (e.g. *good-02*).

8 Conclusion

In this paper, we focus on cross-lingual AMR parsing. Previous works have deficiency in predicting correct AMR concepts. We thus introduce bilingual inputs as well as an auxiliary task to predict more accurate concepts and their relations in AMR graphs. Empirical results on data in German, Italian, Spanish and Chinese demonstrate the efficacy of our proposed method. We also conduct ablation study to further verify the significance of the bilingual inputs and auxiliary task. For future work, we will attempt to adapt other methods used in English

AMR parsing to cross-lingual AMR parsing, such as pre-training and self-training.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. Enabling cross-lingual amr parsing with transfer learning techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Marco Damonte and Shay B Cohen. 2018. Cross-lingual abstract meaning representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436.

Donglai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for neural amr parsing. In *IJCAI*, pages 4975–4981.

Zhijiang Guo and Wei Lu. 2018. Better transition-based amr parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Jan Hajic, Ondřej Bojar, and Zdenka Uresova. 2014. Comparing czech and english amrs. In *Proceedings of Workshop on Lexical and Grammatical Resources for Language Processing*, pages 55–64.

Fuad Issa, Marco Damonte, Shay B Cohen, Xiaohui Yan, and Yi Chang. 2018. Abstract meaning representation for paraphrase detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 442–452.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

Young-Suk Lee, Ramón Fernandez Astudillo, Tahira Naseem, Revanth Gangi Reddy, Radu Florian, and Salim Roukos. 2020. Pushing the limits of amr parsing with self-learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3208–3214.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An amr aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding smatch: Transition-based amr parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Jörg Tiedemann and Santhosh Thottingal. 2020. Opus-mt–building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. Improving amr parsing with sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511.

Nianwen Xue, Ondrej Bojar, Jan Hajic, Martha Palmer, Zdenka Uresova, and Xiuhong Zhang. 2014. Not an interlingua, but close: Comparison of english amrs to chinese and czech. In *LREC*, volume 14, pages 1765–1772. Reykjavik, Iceland.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. Amr parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. *arXiv preprint arXiv:1909.02607*.

Pushing Paraphrase Away from Original Sentence: A Multi-Round Paraphrase Generation Approach

Zhe Lin and **Xiaojun Wan**
Wangxuan Institute of Computer Technology, Peking University
Center for Data Science, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{linzhe,wanxiaojun}@pku.edu.cn

Abstract

In recent years, neural paraphrase generation based on Seq2Seq has achieved superior performance, however, the generated paraphrase still has the problem of lack of diversity. In this paper, we focus on improving the diversity between the generated paraphrase and the original sentence, i.e., making generated paraphrase different from the original sentence as much as possible. We propose **BTmPG** (**B**ack-**T**ranslation guided **m**ulti-round **P**araphrase **G**eneration), which leverages multi-round paraphrase generation to improve diversity and employs back-translation to preserve semantic information. We evaluate BTmPG on two benchmark datasets. Both automatic and human evaluation show BTmPG can improve the diversity of paraphrase while preserving the semantics of the original sentence.

1 Introduction

Paraphrase generation or sentence paraphrasing is an important task in natural language processing, and it requires rewriting a sentence while preserving its semantics. Paraphrase generation has been widely used in many downstream tasks such as QA systems, semantic parsing, dialogue systems and so on.

In recent years, deep learning techniques like sequence-to-sequence(Seq2Seq) have achieved superior performance on natural language generation tasks (Zhao et al., 2010; Wubben et al., 2010). Many paraphrase models based on Seq2Seq have achieved inspiring results. For example, Prakash et al. (2016) leveraged stacked residual LSTM networks to generate paraphrase, and Gupta et al. (2018) proposed a deep generative framework based on variational auto-encoder for paraphrase generation.

Though paraphrase generation models based on Seq2Seq have demonstrated advanced ability, the generated paraphrase still has the problem of lack of diversity, i.e., the output paraphrase only makes trivial changes to the original sentence. A good paraphrase of a sentence is one that is semantically similar to that sentence while being (very) syntactically and/or lexically different from it (Bhagat and Hovy, 2013). Paraphrase which is too similar to the original sentence is much less useful in many real applications.

In this paper, we focus on improving the diversity of generated paraphrase, i.e., making generated paraphrase different from the original sentence as much as possible. An intuitive but uninvestigated idea is to adopt multi-round paraphrase generation. Concretely, we first send the original sentence into a paraphrase generation model to generate a paraphrase, and then we use the generated paraphrase as the input of the model to generate a new paraphrase. As long as we leverage a paraphrase generation model with strong diversity like variational auto-encoder (VAE)(Kingma and Welling, 2013), we can get the paraphrase as different as possible from the original sentence after multi-round generation.

However, existing paraphrase models can not ensure that the major semantics of the original sentence can be preserved after multi-round paraphrase generation, especially the model with strong diversity. With the increase of paraphrasing round, the generated sentence will be more and more different from the original sentence, and the semantics will be gradually different from the original sentence as well. To tackle this problem, we introduce back-translation to maintain the semantics of paraphrase. Back-translation, which translates the generated sentence into the original sentence, has been widely used in semi-supervised natural language generation (Zhao et al., 2020) and data augmentation(Li et al., 2020), and it can improve the robustness of machine-translation system (Li

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1548–1557
August 1–6, 2021.

and Specia, 2019). We assume that paraphrase with similar semantics can be translated back to the original sentence. So, we can leverage back-translation to provide guidance for multi-round paraphrase generation.

Particularly, we propose **B**ack-**T**ranslation guided **m**ulti-round **P**araphrase **G**eneration (**BTmPG**), by combining neural paraphrase model with back-translation to generate paraphrases in a multi-round process. The contributions of our work are summarized as below:

1) We propose a new multi-round paraphrase generation method to generate diverse paraphrase that is much different from the original sentence and leverage back-translation to preserve the major semantics during the multi-round paraphrase generation. Our code is publicly available at `https://github.com/L-Zhe/BTmPG`.

2) Automatic and human evaluation results demonstrate that our method can substantially improve the diversity of generated paraphrase, while preserving the semantics during multi-round paraphrase generation.

2 Related Work

Paraphrase generation or sentence paraphrasing can been seen as a monolingual translation task. Prakash et al. (2016) leveraged stacked residual LSTM networks to generate paraphrase. Gupta et al. (2018) found deep generative model such as variational auto-encoder can be able to achieve better performance in paraphrase generation. Li et al. (2019) proposed DNPG to decompose a sentence into sentence-level pattern and phrase-level pattern to make neural paraphrase generation more interpretable and controllable, and they found DNPG can be adopted into unsupervised domain adaptation method for paraphrase generation. Fu et al. (2019) proposed a new paraphrase model with latent bag of words. Wang et al. (2019) found that adding semantics information into paraphrase model can significantly boost performance. Siddique et al. (2020) proposed an unsupervised paraphrase model with deep reinforcement learning framework. Liu et al. (2020) regarded paraphrase generation as an optimization problem and proposed a sophisticated objective function. All methods above focus on the generic quality of paraphrase and do not care about the diversity of paraphrase.

There are also some methods focusing on improving the diversity of paraphrase. Gupta et al. (2018) leveraged VAE to generate several different paraphrases by sampling the latent space. (Kumar et al., 2019) provided a novel formulation of the problem in terms of monotone sub-modular function maximization to generate diverse paraphrase. Goyal and Durrett (2020) used syntactic transformations to softly "reorder" the source sentence and guide paraphrase model. Thompson and Post (2020) introduced a simple paraphrase generation algorithm which discourages the production of n-grams that are present in the input to prevent trivial copies or near copies. Note that the purpose of the work (Gupta et al., 2018; An and Liu, 2019) is different from ours, while Thompson and Post (2020) has the same purpose with our work, i.e., pushing the generated paraphrase away from the original sentence.

3 Model

In this section, we introduce the components of our model in detail. First, we define the paraphrase generation task and give an overview of our model. Next, we describe the paraphrase model and the back-translation model. Then, we show how to use the gumble-softmax to connect the paraphrase model with the back-translation model. Finally, we describe the loss function and training process of our model in detail. Figure1 shows an overview of our model.

3.1 Notations and Overview

Our model regards paraphrase generation as a monolingual translation task. Given a paraphrase pair (S_0, P), which S_0 is the original/source sentence and P is the target paraphrase given in the dataset.

As is shown in Figure 1, we introduce a multi-round paraphrasing method. In the **first round** generation, we send S_0 into a paraphrase model to generate a paraphrase S_1. In the **second round** generation, we use the S_1 as the input of the model to generate a new paraphrase S_2. And so forth, in the ***i*-th round** generation, we send S_{i-1} into the paraphrase model to generate S_i.

Although multi-round generation can increase the paraphrase diversity, the semantics of paraphrase may change during generation. We thus introduce back-translation to tackle this problem based on the assumption that paraphrase can be translated back to the original sentence while the

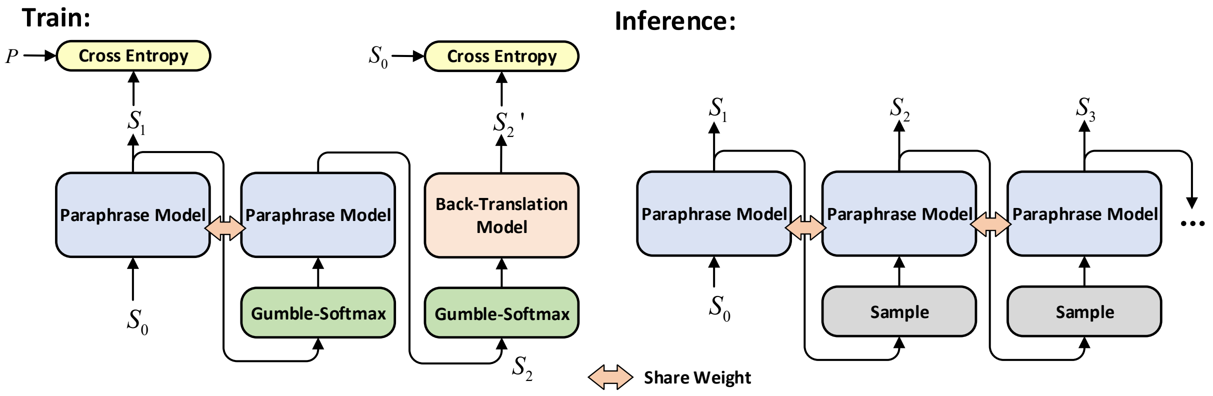

Figure 1: An overview of BTmPG, which leverages back-translation to guide paraphrase model during training and generates paraphrases in a multi-round process.

semantic information has not been changed. In the first round, we calculate the loss between S_1 and P to train our paraphrase model. In the i-th round, we send its generated paraphrase S_i into a back-translation model to generate $S_i^{'}$, and we optimize the cross-entropy loss between $S_i^{'}$ and S_0. The back-translation model which translates the paraphrase in i-th round back to the original sentence can guide the paraphrase to preserve semantics during multi-round generation.

In addition, we introduce gumble-softmax embedding to tackle the problem that the model with sampling operation between different rounds' generation can not be optimized by SGD optimizer.

3.2 Paraphrase Model

We require sufficient diversity of paraphrase model so that it is able to introduce enough changes in the paraphrase of each round. The VAE (Kingma and Welling, 2013; Rezende et al., 2014) is a deep generative model that allows learning rich, nonlinear representations for high dimensional inputs. It can improve the diversity by sampling from latent space. Bowman et al. (2016) proposed a new model to apply VAE to natural language generation for the first time. Our paraphrase model is based on conditional VAE with LSTM. Transformer (Vaswani et al., 2017) has achieved excellent performance in many tasks. But our experiments show that it may cause KL divergence to become 0, called posterior collapse, which means a decrease of diversity. So we do not employ Transformer as encoder and decoder.

We define the embedding matrices of S_i and P as $E_s^i = \{e_s^1, e_s^2, \cdots, e_s^{L_i}\}$ and $E_p = \{e_p^1, e_p^2, \cdots, e_p^M\}$ respectively, where $e_s^i, e_p^j \in \mathbb{R}^{d_e}$ are the embedding vector of the word in S_i and P, and d_e is the embedding dimension.

3.2.1 Encoder

Conditional VAE contains two encoders that share parameters: an original sentence encoder and a paraphrase encoder. We first send E_s^i into original sentence encoder to get its encoding matrix $O_s^i \in \mathbb{R}^{d_h \times L_i}$ and vector representation $h_s^i \in \mathbb{R}^{d_h}$ of S_i, where d_h is the hidden dimension of LSTM. Then we send E_p and h_s^i into paraphrase encoder to get its vector representation h_z. h_z is passed through two different feed-forward neural networks with parameter Φ to produce the mean μ and the variance σ^2 of the distribution of latent space. We can get the latent code $\boldsymbol{z} \in \mathbb{R}^{d_z}$ by sampling from latent space and reparameterization, where d_z is the dimension of latent code.

3.2.2 Decoder

We define the embedding matrix which be sent into decoder as $E_d = \{e_d^1, e_d^2, \cdots, e_d^N\} \in \mathbb{R}^{d_e \times N}$. Then, we concatenate z with the embedding vector e_d^i as the input of decoder. The decoder also takes h_s^i as input. The output of decoder is defined as $O_d^i \in \mathbb{R}^{d_h \times N}$. Then, an attention (Luong et al., 2015) and copy mechanism (See et al., 2017) are leveraged as follow. First, we get the attention weight p_a and attention vector V_a as follow.

$$\begin{aligned} p_a &= \mathrm{softmax}(O_d^i {O_s^i}^\top) \\ V_a &= p_a O_s^i \end{aligned} \tag{1}$$

Then, we leverage them to calculate the decoder probability p_d and copy probability η.

$$\begin{aligned} p_d &= \mathrm{softmax}(\mathbf{W}_o[O_d^i || V_a] + \mathbf{b}_o) \\ \eta &= \sigma(\mathbf{W}_h[O_d^i || V_a] + \mathbf{W}_s\{e_d^i || \boldsymbol{z}\}_{i=1}^N + \mathbf{b}_\eta) \end{aligned} \tag{2}$$

where $\mathbf{W}_o, \mathbf{W}_h, \mathbf{W}_b, \mathbf{b}_o, \mathbf{b}_\eta$ are learnable parameters. $||$ is the concatenation operation. σ is the sigmoid activation function. The final output probability of decoder is as follow.

$$p = \eta p_d + (1 - \eta) p_a \tag{3}$$

3.2.3 Loss Function of Paraphrase Model

The VAE with parameter Θ is trained by minimizing the following objective:

$$\begin{aligned} \mathcal{L}_{Para} = &- \mathrm{KL}(q_\Phi(\boldsymbol{z}|S_i, P)||p(\boldsymbol{z})) \\ &+ \mathbb{E}_{q_\Phi(\boldsymbol{z}|P)}[\log p_\Theta(P|\boldsymbol{z}, S_i)] \end{aligned} \tag{4}$$

where KL stands for the KL divergence. Eq. 4 is called evidence lower bound, which provides a lower bound of $\log p(P|S_i; \Theta)$.

Bowman et al. (2016) figured out that variational inference for text generation often yields models that ignore their latent variables, a phenomenon called posterior collapse. This may cause the low diversity of generated sentences. To tackle this problem, we propose a diversity loss. We find that the diversity of the generated sentence is affected by its first word. For example, the first word can determine the form of a question sentence. Unfortunately, compared with the questions beginning with “Is, May, Would”, we are more likely to collect questions beginning with “What, When, How”. This can lead to serious category imbalances when generating the first word. So we set the penalty coefficient of the first-word loss as follow.

$$\mathcal{L}_{w_1} = \ln \left(\frac{N_b}{n_{w_1}} e \right) \log p(w_1|\Theta) \tag{5}$$

where N_b is the batch size during the training process, n_{w_1} is the number of sentences beginning with w_1 in this batch. e is the Euler’s number that can make sure the penalty coefficient always no less than 1.

3.3 Back-Translation Model

Back-translation model aims to make sure the semantics of the generated paraphrase are the same with the original sentence during multi-round generation. It translates S_i back to S_0. Different from paraphrase model which needs diversity, back-translation model is more focused on semantics maintaining. We employ Transformer (Vaswani et al., 2017) with copy mechanism as back-translation model because of its excellent performance in many tasks.

The loss function of back-translation model is as follow: [1]:

$$\begin{aligned} \mathcal{L}_s^i &= \mathrm{CrossEntropy}(\mathrm{BTModel}(S_i), S_0) \\ \mathcal{L}_p &= \mathrm{CrossEntropy}(\mathrm{BTModel}(P), S_0) \\ \mathcal{L}_{BT} &= \mathcal{L}_p + \lambda \sum_i \mathcal{L}_s^i \end{aligned} \tag{6}$$

where λ is a hyper-parameter. BTModel indicates the back-translation model.

There are two parts in the loss of back-translation model: $\mathcal{L}_s^i$ and $\mathcal{L}_p$. We assume the i-th round paraphrase can be translated back to the original sentence S_0 if its semantics are preserved and thus we optimize $\mathcal{L}_s^i$. Similarly, the paraphrase P can be translated back to the original sentence S_0 as well, so we also leverage $\mathcal{L}_p$ to train back-translation model. This can improve the generalization ability of the back-translation model, because back-translation model tends to guide paraphrase model to copy original sentence without changes if we do not employ true paraphrase data to train it.

3.4 Gumble-Softmax Embedding

We employ gumble-softmax embedding to connect each module of our model. We first define an embedding operation as follow:

$$\mathrm{Embed}(X) = \mathbf{W}_e X \tag{7}$$

For the probability p generated by paraphrase model, we leverage gumble-softmax(Jang et al., 2017) to get its one-hot matrix without sampling from multinomial distribution. Then we can get the embedding matrix E as follow:

$$\begin{aligned} \mathrm{GS}(\boldsymbol{\pi}) &= \mathrm{softmax}((\log(\pi)_i + g_i)/\tau) \\ E &= \mathrm{Embed}\left(\mathrm{GS}(p)\right) \end{aligned} \tag{8}$$

where $\boldsymbol{\pi}$ is a multinominal distribution wih k dimension, $g_1, g_2, \cdots, g_i$ are i.i.d samples drawn from $\mathrm{Gumbel}(0, 1)$. τ is a hyper-parameter.

There are three places in our model needing to leverage gumble-softmax embedding. First, we leverage it to embed the output probability of the paraphrase model as the input of the next-round paraphrase model. Next, gumble-softmax embedding is also used to connect the back-translation model with the paraphrase model. Figure 1 shows

---

[1] Note that we also use S_0, S_i and P to denote the one-hot matrix of corresponding sentences.

these two cases. Finally, it is used in the multi-round paraphrase generation process to replace the teacher forcing. Generally, Seq2Seq model employs teacher forcing for model training, with using ground truth to guide the generation process. However, there is no ground truth in multi-round paraphrase generation, it can only generate sentence with a autoregressive method. We employ gumble-softmax to replace sampling in each step of the autoregressive process. Figure 2 shows this process.

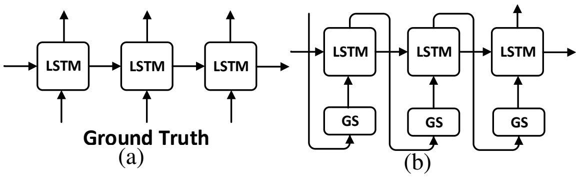

Figure 2: Figure (a) shows the decoder with teacher forcing in the first round generation. Figure (b) shows the decoder with autoregression in other-round generation.

3.5 Loss Function

We train paraphrase model together with back-translation model. The total loss of our model is as follow:

$$\mathcal{L} = \mathcal{L}_{para} + \mathcal{L}_{BT} \tag{9}$$

Although we define a multi-round paraphrase model, we only train the first two rounds. Because we find that training too many rounds requires large computing resources, but can not improve the model performance significantly. During inference, we can generate paraphrase more than two rounds.

4 Experiment

4.1 Datasets

We evaluate our BTmPG model on two benchmark datasets:

MSCOCO[2] (Lin et al., 2014) dataset contains human annotated captions of over 120k images. Each image contains five captions from five different annotators. This dataset has been widely used in previous works (Prakash et al., 2016; Gupta et al., 2018; Cao and Wan, 2020). We sample the MSCOCO according to Prakash et al. (2016).

Quora[3] dataset is a question paraphrase dataset. It contains over 400k question pairs. Each pair marked with a binary value indicates whether the questions in the pair are truly a duplicate of each other. So we select all such question pairs with binary value 1 as paraphrase dataset. There are about 150k question pairs in total. We randomly divide the training, validation and the test set.

Table 1 provides statistics of these two benchmark datasets.

Dataset	Train Set	Valid Set	Test Set
MSCOCO	206,852	3,000	3,000
Quora	129,263	3,000	3,000

Table 1: Statistic for datasets: the sizes of training, validation and test set.

4.2 Evaluation Metrics

We use five widely-used metrics to evaluate paraphrases: BLEU4, self-BLEU, self-TER, BERTScore and p-BLEU.

BLEU4 is widely used in generation tasks. It can measure how well the sentences generated by our model can match the references. Notice that some works also calculate the ROUGE(Lin, 2004) or METEOR, but we think the role of these two metrics overlaps with BLEU4, as they all calculate the overlap degree between outputs and references. Therefore we only calculate BLEU4 to evaluate the match degree between outputs and references.

We evaluate the difference between the output sentence and the original sentence with two metrics. One of them is **self-BLEU** which is the BLEU4 score between the output sentence and the original sentence. The lower the value of self-BLEU, the more difference between output sentences and original sentences. Another is **self-TER**[4]. **TER**(Zaidan and Callison-Burch, 2010) is used to evaluate the edit distance between two sentences. Self-TER is calculated as the TER between the output sentence and the original sentence.

BERTScore [5] is proposed by Zhang et al. (2020) to evaluate the semantic similarity between the output sentence and the original sentence. BERTScore

---

[2] `https://cocodataset.org/`

[3] `https://www.kaggle.com/c/quora-question-pairs/data?select=train.csv.zip`

[4] We use the tool at `https://github.com/jhclark/multeval`.

[5] The tool of BERTScore is available at `https://github.com/Tiiiger/bert_score`

has been widely leveraged to measure semantic preserving in the paraphrase generation task (Cao and Wan, 2020). However, there may be some problems for BERTScore on our task due to the low score for reference. This is because BERTScore is not perfect in measuring semantic relevance. But as far as we know, there is no better score to evaluate semantic preserving, so we report BERTScore as a reference for semantic preserving. More evaluation about semantic relevance is shown in human evaluation.

We leverage **p-BLEU** (Cao and Wan, 2020) to evaluate the difference between outputs in different rounds. Concretely, for outputs in k rounds $\{y_1, y_2, \cdots, y_k\}$, the p-BLEU can be calculated as follow.

$$\text{p-BLEU} = \frac{\sum_i \sum_{j \neq i} \text{BLEU4}(y_i, y_j)}{k \times (k-1)} \tag{10}$$

The lower p-BLEU means higher diversity between outputs in different rounds.

Notice that, BLEU4 may not suitable for our task , because we focus on the diversity of paraphrase. BLEU4 can only measure the match degree between outputs and references. However, a sentence usually has many more reference paraphrases, while the target given in the dataset is only one reference. So we also perform human evaluation to evaluate the semantic relevance, readability and diversity of generated paraphrases.

4.3 Baseline

As our model focuses on the diversity of paraphrase, we mainly compare our model with VAE-SVG-eq (Gupta et al., 2018), DiPS(Kumar et al., 2019)[6], SOW-REAP(Goyal and Durrett, 2020)[7] and the decoding method proposed by Thompson and Post (2020)[8]. The last method penalizes the n-gram appearing in the original sentence to make the paraphrase different from the original sentence and enhance diversity. We mark this method as N-gram Penalty. We employ two different hyper-parameters provided by the authors: one of them is low penalty for N-gram, and another is high penalty. In addition, we also compare our model with Transformer and Transformer copy.

4.4 Training Details

For both datasets, we truncate all the sentences longer than 20 words and maintain a vocabulary size of 25k. During testing, we replace UNK with the original word with the highest copy probability.

For paraphrase model, we leverage 2-layer LSTM. We set the embedding dimension d_e to 300, hidden size d_h of LSTM to 512. We set the latent code dimension d_z to 128. For back-translation model, we leverage Transformer-copy with 3-layer encoder and decoder. We set the model size to 450, and the head number of multi-head attention to 9. We set λ to 1, which will be discussed in our ablation study. For the hyper-parameter τ in gumble-softmax, we refer (Nie et al., 2019) to increase the τ over iterations via an exponential policy: $\tau = \tau_{\max}^{-n_e/N_e}$, where n_e is the current epoch and N_e is the total number of epoch. We set $\tau_{\max}$ to 5. We train our model for 30 epochs. We set batch size to 50, and we select the model of the final epoch to generate paraphrase in test set.

5 Result

5.1 Automatic Evaluation

Table 2 shows the results of automatic evaluation. Our model substantially improves the BERTScore in the first round of paraphrase generation and generally gets the state-of-the-art performance. The value of self-BLEU can be significantly reduced with the increase of the round number of paraphrase generation while maintaining semantics.

For both datasets, the first round paraphrase generation of our model achieves the highest BERTScore than any other models. This is because back-translation model can provide sufficient semantic guidance for paraphrase model. As the increase of the round number, the values of self-BLEU and self-TER are reduced significantly, which means the paraphrase sentences our model generated are more and more different from original sentences. While BERTScore can still maintain a relatively high value. (A slight reduction of BERTScore is acceptable as BERTScore is not perfect in measuring semantic relevance.) We find that the paraphrase generated in the fifth round is good with balancing the diversity and the relevancy.

DiPS gets the BERTScore similar to round 5 generation, while its outputs lack of diversity com-

---

[6] The code is available at `https://github.com/malllabiisc/DiPS`.

[7] The code is available at `https://github.com/tagoyal/sow-reap-paraphrasing`.

[8] DNPG (Li et al., 2019), which controls semantics through encoding different levels of granularity respectively, can also enhance diversity. But the code and outputs are not provided, so we are not able to use it as baseline.

Model		MSCOCO				Quora			
		BLEU4	self-BLEU ↓	self-TER ↑	BERTScore ↑	BLEU4	self-BLEU ↓	self-TER ↑	BERTScore ↑
Reference		-	8.12	78.40	46.20	-	31.46	54.92	67.37
VAE-SVG-eq		25.07	13.77	66.92	51.72	22.52	36.05	50.25	67.06
Transformer		25.81	14.92	65.74	54.47	26.22	37.99	46.42	67.53
Transformer copy		26.80	17.94	62.49	56.49	28.97	43.69	42.52	71.05
DiPS		23.52	12.23	67.31	51.40	23.38	29.24	54.41	63.09
SOW-REAP		15.31	44.22	39.42	**63.68**	15.36	47.62	38.98	62.21
N-gram Penalty-low		25.37	12.16	66.66	53.47	26.00	36.65	47.31	66.93
N-gram Penalty-high		23.68	0.00*	69.24	52.08	17.53	0.00*	**59.30**	59.20
BTmPG (Ours)	R1	25.54	18.50	61.78	59.34	28.02	58.47	33.99	**77.21**
	R5	23.65	12.58	68.27	54.07	23.15	37.89	48.62	65.90
	R10	22.42	**10.98**	**70.10**	52.37	22.17	**34.15**	53.34	62.91

Table 2: Automatic evaluation results on MSCOCO and Quora test sets. In the table, R1, R5 and R10 mean the first round, the fifth round and the tenth round of paraphrase generation.

pared with our model. SOW-REAP gets the highest BERTScore for MSCOCO, but it does not perform well on self-BLEU. Because SOW-REAP tends to generate paraphrase without change, the semantics of the paraphrase may be similar with the original sentence but the paraphrase lacks of diversity. N-gram Penalty with high penalty can lead self-BLEU to 0, as it strictly does not allow to generate those 4-grams appearing in the original sentence. Although the N-gram Penalty method can generate outputs totally different from original sentences, it fails to preserve the major semantics. However, our BTmPG model can increase diversity as much as possible while preserving major semantics.

To explore the pairwise diversity of our model's outputs in different rounds, we also calculate the p-BLEU values for VAE-SVG-eq and our model (p-BLEU is not suitable for other models). For VAE-SG-Eq, we generate 10 outputs by random sampling the latent space. For our model, we select the first 10 rounds outputs. Table 3 shows the results of p-BLEU. The p-BLEU value of our model is much lower than VAE-SVG-eq, which means that our model has better ability to generate multiple diversified paraphrases than VAE-SVG-eq.

5.2 Ablation Study

In this section, we will explore the role of back-translation model in preserving semantics. We set the hyper-parameter λ from 0 to 5. A bigger λ means back-translation provides more semantic guidance to paraphrase model. $\lambda = 0$ means that we remove back-translation model totally. We generate paraphrases of 20 rounds and calculate the values of BERTScore. In order to explore the effect of leveraging other paraphrase model in the multi-round generation framework, we also adopt VAE-SVG-eq in a multi-round generation process to generate paraphrases of 20 rounds on Quora, and compute the values of BERTScore. Figure 3 shows the trend of BERTScore with the increase of the round number.

Model	p-BLEU ↓	
	MSCOCO	Quora
VAE-SVG-eq	75.52	81.50
BTmPG(Ours)	**62.83**	**67.60**

Table 3: The p-BLEU score for VAE-SVG-eq and BTmPG

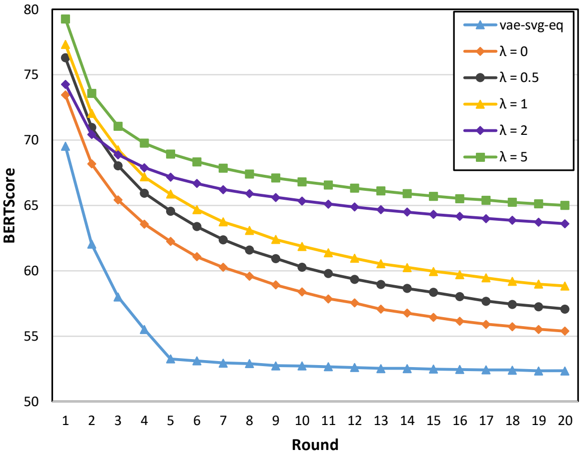

Figure 3: The BERTScore of paraphrases of 20 rounds on Quora.

Obviously, compared with VAE-SVG-eq, our improved VAE model can preserve semantics better. Back-translation can much improve the lower bound of BERTScore , which means back-translation can help to preserve the semantics during multi-round paraphrase generation.

We also calculate the p-BLEU for the paraphrases of the first 10 rounds for different λ. Table 4 shows the result. From the table we can know that, although back-translation can help to preserve semantics, a higher λ can lead to a lack of diversity of paraphrase. Therefore, it is wise to select an appropriate λ according to the actual requirement.

λ	0	0.5	1	2	5
p-BLEU	63.53	66.40	67.61	74.83	88.05

Table 4: The p-BLEU of paraphrases of the first 10 rounds for different λ.

5.3 Human Evaluation

We perform human evaluation on system outputs with respect to three aspects: relevancy, fluency and diversity. Relevancy indicates if the semantics of outputs and original are identical. Fluency indicates the readability of output sentences. Diversity indicates the lexical and syntactic differences between output sentences and original sentences and thus we use two indicators for lexical diversity and syntactic diversity respectively.

We randomly sample 100 sentences from each test set and get a total of 200 sentences for evaluation. We employ 6 graduate students to rate each instance. We ensure every instance is rated by at least three judges. Table 5 shows the result of human evaluation.

Model		**Relevancy**	**Fluency**	**Diversity**	
				Lexical	**Syntactic**
VAE-SVG-eq		3.24	3.44	3.93	4.01
Transformer		3.82	**3.96**	3.71	3.77
DiPS		3.62	3.50	3.64	3.70
SOW-REAP		3.59	3.34	2.79	3.88
N-gram Penalty		3.44	3.65	3.79	3.68
BTmPG (Ours)	R1	**4.12**	3.92	3.65	3.85
	R5	3.93	3.81	3.95	4.00
	R10	3.84	3.82	**4.20**	**4.15**

Table 5: Human evaluation results.

From the table, we can see that the paraphrase in the first round can preserve more semantics of original sentence but lack of diversity. With the increase of the round number, the relevancy score decreases slightly, but the diversity scores increase substantially. Fluency may be influenced by diversity, because human may feel a slight decrease of fluency with the increase of diversity. As compared with other models, our model can generate paraphrases with high diversity, while maintaining semantics and fluency well. Previous models like SOW-REAP and DiPS can not maintain the semantics, though they can produce paraphrases with relatively high diversity.

5.4 Case Study

We perform case studies for better understanding the model performance. Table 6 shows an example of Quora, which include paraphrases of the first 15 rounds.

Cases from Quora	
Original	why did modi scrap rs 500 & rs 1000 notes ? and what 's the reason for the sudden introduction of the 2000 rupee note ?
Reference	why did goi demobilise 500 and 1000 rupee notes ?
Round1	why did the indian government ban the 500 and 1000 rupee notes and why is it bringing to ?
Round2	what do you think about the ban on 500 and 1000 denomination notes in india ?
Round4	how do you see the pm modi 's move of banning 500 and 1000 rupee currency notes ?
Round5	what do you think of the decision by the indian government to demonetize 500 and 1000 rupee notes ?
Round9	is modi 's decision on demonetization of 500 and 1000 notes by public modi ?
Round11	was the decision by the indian government to demonetize 500 and 1000 notes right or wrong ?
Round12	would banning notes of denominations 500 and 1000 help to curb the black money in india ?
Round13	what will be the effects of banning 500 and 1000 rupees on indian economy ?
Round14	what are the advantage of banning 500 and 1000 rupees in Indian ?
Round15	what are the pros and corns of banning 500 and 1000 rupees by indian government ?

Table 6: An example of Quora and the generated paraphrases in multiple rounds. The word in color means that it does not appear in the original sentence.

This case shows how does our model modify sentences during multi-round paraphrase generation process. With the increase of round number, the difference between the generated paraphrase and the original sentence becomes larger, while the paraphrase still preserves the major semantics of the original sentence.

6 Conclusion

In this paper, we focus on improving the diversity of generated paraphrase, i.e., making the generated paraphrase much more different from the original sentence. We propose a multi-round paraphrase generation method **BTmPG** with the guidance of back-translation. Both automatic and human evaluation results show that our method can generate diverse paraphrase while maintaining semantics. Ablation study proves back-translation is very helpful to preserve semantics. In the future, we will explore other methods such as GAN, to improve paraphrase diversity. We will also test our method on more languages other than English.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Zhecheng An and Sicong Liu. 2019. Towards diverse paraphrase generation using multi-class wasserstein GAN. *CoRR*, abs/1909.13827.

Rahul Bhagat and Eduard Hovy. 2013. Squibs: What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Yue Cao and Xiaojun Wan. 2020. DivGAN: Towards diverse paraphrase generation via diversified generative adversarial network. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2411–2421, Online. Association for Computational Linguistics.

Yao Fu, Yansong Feng, and John P Cunningham. 2019. Paraphrase generation with latent bag of words. In *Advances in Neural Information Processing Systems*, pages 13645–13656.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *International conference on learning representations*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *International Conference on Learning Representations*.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619, Minneapolis, Minnesota. Association for Computational Linguistics.

Hongzheng Li, Jiu Sha, and Can Shi. 2020. Revisiting back-translation for low-resource machine translation between chinese and vietnamese. *IEEE Access*, 8:119931–119939.

Zhenhao Li and Lucia Specia. 2019. Improving neural machine translation robustness via data augmentation: Beyond back-translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 328–336, Hong Kong, China. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2020. Unsupervised paraphrasing by simulated annealing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 302–312, Online. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Weili Nie, Nina Narodytska, and Ankit Patel. 2019. Relgan: Relational generative adversarial networks for text generation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934, Osaka, Japan. The COLING 2016 Organizing Committee.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *international conference on machine learning.*, pages 1278–1286.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

A. B. Siddique, Samet Oymak, and Vagelis Hristidis. 2020. Unsupervised paraphrasing via deep reinforcement learning. In *Association for Computing Machinery*, KDD '20, page 1800–1809.

Brian Thompson and Matt Post. 2020. Paraphrase generation as zero-shot multilingual translation: Disentangling semantic similarity from lexical and syntactic diversity. In *Proceedings of the Fifth Conference on Machine Translation*, pages 561–570, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7176–7183.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*.

Omar F. Zaidan and Chris Callison-Burch. 2010. Predicting human-targeted translation edit rate via untrained human annotators. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 369–372, Los Angeles, California. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Shiqi Zhao, Haifeng Wang, Xiang Lan, and Ting Liu. 2010. Leveraging multiple MT engines for paraphrase generation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1326–1334, Beijing, China. Coling 2010 Organizing Committee.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *AAAI*, pages 9668–9675.

Few-shot Knowledge Graph-to-Text Generation with Pretrained Language Models

Junyi Li[1,3], **Tianyi Tang**[1], **Wayne Xin Zhao**[1,3,5*], **Zhicheng Wei**[4],
Nicholas Jing Yuan[4] and **Ji-Rong Wen**[1,2,3]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Huawei Cloud
[5]Beijing Academy of Artificial Intelligence, Beijing, 100084, China
`{lijunyi,jrwen,steven_tang}@ruc.edu.cn`
`{batmanfly,nicholas.jing.yuan}@gmail.com` `weizhicheng1@huawei.com`

Abstract

This paper studies how to automatically generate a natural language text that describes the facts in knowledge graph (KG). Considering the few-shot setting, we leverage the excellent capacities of pretrained language models (PLMs) in language understanding and generation. We make three major technical contributions, namely representation alignment for bridging the semantic gap between KG encodings and PLMs, relation-biased KG linearization for deriving better input representations, and multi-task learning for learning the correspondence between KG and text. Extensive experiments on three benchmark datasets have demonstrated the effectiveness of our model on KG-to-text generation task. In particular, our model outperforms all comparison methods on both fully-supervised and few-shot settings. Our code and datasets are available at `https://github.com/RUCAIBox/Few-Shot-KG2Text`.

1 Introduction

Knowledge graphs (KGs), such as Wikidata and DBpedia, are essential for many natural language processing (NLP) applications (Ji et al., 2020). To understand the structured information in KG, the task of KG-to-text generation has been proposed to automatically generate a descriptive text for a given knowledge graph (Koncel-Kedziorski et al., 2019; Ribeiro et al., 2020a). Figure 1 illustrates a KG with the corresponding descriptive text, in which the nodes (*e.g., Stan Lee* and *Iron Man*) represent entities and the edges (*e.g., creator* and *alias*) describe the relations between connected entities.

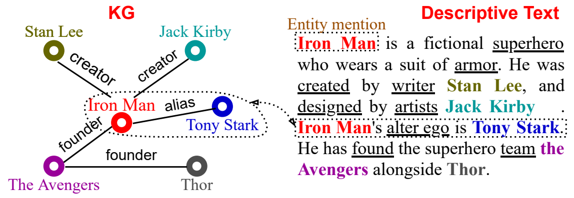

Iron Man is a fictional superhero who wears a suit of armor. He was created by writer Stan Lee, and designed by artists Jack Kirby . Iron Man's alter ego is Tony Stark. He has found the superhero team the Avengers alongside Thor.

Figure 1: A knowledge graph (subgraph) with its descriptive text. The underlined words represent the context keywords about entities.

In recent years, with the help of crowdsourcing platforms and information extraction (IE) systems, large-scale labelled pairs of KG and its descriptive text have been created, such as WikiBio (Lebret et al., 2016) and WebNLG Challenge (Gardent et al., 2017). Based on these datasets, data-driven models have shown impressive capabilities to produce informative and fluent text for a given KG (Logan et al., 2019; Moryossef et al., 2019). However, due to the great expense in annotation process, it is not always feasible to generate large-scale labelled datasets for a variety of domains in practice. Motivated by this, we propose to study the task of *few-shot KG-to-text generation* that aims to produce satisfactory output text given only a handful of (several hundred) labelled instances.

To fulfil this task, we need to fully understand the complicated semantic relations between entities from various domains, which is challenging with limited labelled data. Our solution is inspired by the excellent few-shot capabilities of pretrained language models (PLMs) on language understanding and generation tasks (Brown et al., 2020; Chen et al., 2020; Li et al., 2021a). Pretrained on the large-scale corpora, PLMs encode vast amounts of world knowledge into their parameters (Li et al., 2021b), which is potentially beneficial to understand and describe the KG facts in our task.

However, applying PLMs to few-shot KG-to-text generation still faces two challenges. First, PLMs are usually pretrained on natural language text, while the KG inputs for our task are structured graphs. This semantic gap makes it difficult to effectively inject KG representations into PLMs

---

[*] Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1558–1568
August 1–6, 2021.

especially with limited labelled instances. Second, unlike many other text generation tasks, KG-to-text generation requires faithful generation based on the understanding of KG facts. It needs to learn an accurate semantic correspondence between input KG and output text, which will be more difficult in few-shot settings.

To address the above issues, in this paper, we propose a few-shot KG-to-text generation model based on PLMs. There are three major technical contributions in our model. First, in order to bridge the semantic gap, we enforce the representation alignment by learning the correspondence between KG representations (encoded by graph neural networks) and PLM-based entity representations. Second, to feed KG into PLMs, we propose a relation-biased breadth-first search (RBFS) strategy to linearize KG into a well-planned entity sequence. Finally, we jointly train the primary text generation task and an auxiliary KG reconstruction task under the framework of multi-task learning. This step further enhances the semantic correspondence between input KG and output text, based on which our model can generate faithful text about KG.

To the best of our knowledge, we are the first study to investigate PLMs for few-shot KG-to-text generation. Extensive experiments on three benchmark datasets demonstrate the effectiveness of our few-shot KG-to-text generation model.

2 Related Work

In this work, we mainly focus on generating text from knowledge graphs using PLMs.

KG-to-Text Generation. Early works mainly centered around statistical methods, applying grammar rules to generate text (Konstas and Lapata, 2013; Flanigan et al., 2016). Recently, neural based approaches have been proposed to generate text from linearized KG triples (Gardent et al., 2017), however, unable to model structural information about KG. Many works explored how to encode the graph structure using Graph Neural Networks (GNNs) or Transformers explicitly. Koncel-Kedziorski et al. (2019) leveraged a graph Transformer encoder to compute node representations by attending over local neighborhoods via self-attention. In contrast, Ribeiro et al. (2020a) focused on combining global and local message passing mechanisms based on GNNs, capturing complementary graph contexts. Guo et al. (2020) presented an unsupervised training method that can iteratively back translate between the text and graph data. Different from them, we explore how to utilize large PLMs for few-shot KG-to-text generation.

Pretrained Language Model. Recent years have witnessed prominent achievement of PLMs in NLP tasks (Devlin et al., 2019; Radford et al., 2019). Pretrained on massive corpora, pretrained models showcase unprecedented generalization ability to solve related downstream tasks (Li et al., 2021b). However, most of existing PLMs were conditioned on text data (Radford et al., 2019; Lewis et al., 2020), lacking consideration of structured data input. Ribeiro et al. (2020b) proposed to utilize PLMs for KG-to-text generation by randomly linearizing graph into a sequence of triples. While, these methods do not explicitly model the structural relations of KG, which is critical for generating faithful text. Our work aims to consider the KG structure and bridge the semantic gap between KG encodings and PLMs.

3 Problem Formulation

KG-to-text generation (Ribeiro et al., 2020a) aims to automatically generate a natural language text that describes the facts in KG.

Formally, the input KG consists of a set of triples, denoted as $\mathcal{G} = \{\langle e, r, e' \rangle | e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and relation set, respectively. A triple $\langle e, r, e' \rangle$ denotes the fact that relation r exists between head entity e and tail entity e'. Note that the input KG is a small and compact subgraph extracted from large-scale knowledge graphs (*e.g.,* DBpedia). Following Koncel-Kedziorski et al. (2019), a text describing the input KG is usually available in this task. Let $\mathcal{V}$ denote the vocabulary. The target is to generate a natural language text $\mathcal{Y} = \langle w_1, ..., w_j, ..., w_T \rangle (w_j \in \mathcal{V})$ that represents the correct and concise semantics of entities and their relations in the given knowledge graph. The text contains a set of entity mentions $\mathcal{M} = \{m_e | m_e = \langle e, s_e, o_e \rangle, e \in \mathcal{E}\}$, where e is the target entity, s_e and o_e are the start and end indices of this mention in text $\mathcal{Y}$, respectively. In other words, $\langle w_{s_e}, ..., w_{o_e} \rangle$ specially corresponds to entity e. For entities with multiple mentions in text, we only keep the first mention of each entity in $\mathcal{M}$. By replacing each word of mentions with the token "[MASK]", we can obtain a masked text, denoted as $\mathcal{Y}_{[mask]}$, which is also taken as input for representation alignment in Section 4.1.

In practice, it is difficult to collect massive pairs

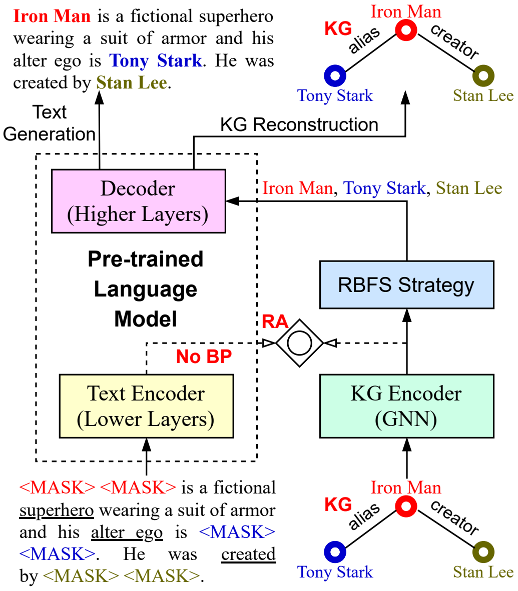

Figure 2: Overview of our proposed model. "RA" and "BP" denote representation alignment and back propagation, respectively. We organize the PLM into lower layers and higher layers. The former provides PLM-based entity representations for alignment with KG encodings, and the latter acts as a decoder for generating text and reconstructing KG facts. After representation alignment, KG embeddings can be directly fed into the higher layers of PLMs for generating text.

of KG and its descriptive text for training. In this paper, we study the task of *few-shot KG-to-text generation* with a handful of training instances (*e.g.,* 200 instances) based on a given PLM (*e.g.,* GPT-2).

4 Approach

For our task, two major challenges are how to learn effective input representations and capture the semantic correspondence between KG and text. To address the two challenges, we propose three major technical contributions, namely representation alignment between KG encodings and PLMs, relation-biased BFS strategy for KG linearization, and multi-task learning with KG reconstruction. Figure 2 presents an illustrative overview of our model. Next we will describe each part in detail.

4.1 Representation Alignment

Unlike previous works (Ribeiro et al., 2020b; Yang et al., 2020) that directly transform KG into text sequence, we employ graph neural network (GNN) as knowledge graph encoder to explicitly encode entity relations in KG. Based on the input KG, GNN would produce a set of entity embeddings, which can be regarded as the input word embeddings of PLM for generating text. However, the GNN-based entity embeddings and the PLM-based word (entity) embeddings come from two distinct semantic spaces. To bridge such a semantic gap, we propose a representation alignment method to align the GNN-based and PLM-based entity embeddings in different semantic spaces.

KG Encoder. The GNN-based KG encoder aims to generate entity embeddings for KG. Let $\boldsymbol{v}_e \in \mathbb{R}^{d_E}$ denote the entity embedding for a general entity e in KG, where d_E is the embedding size. In our work, the entity embeddings are shared across different KGs and initialized with pretrained KG embeddings (Yang et al., 2015). We apply R-GCN (Schlichtkrull et al., 2018) to generate entity embeddings by leveraging multi-relational information in KG. Then, the embedding of entity e at the $l+1$-th layer of R-GCN can be computed as:

$$\boldsymbol{v}_e^{(l+1)} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{e'\in\mathcal{N}_e^r} \boldsymbol{W}_r^{(l)}\boldsymbol{v}_{e'}^{(l)} + \boldsymbol{W}_0^{(l)}\boldsymbol{v}_e^{(l)}\Big), \quad (1)$$

where $\boldsymbol{W}_0^{(l)}$ and $\boldsymbol{W}_r^{(l)}$ are trainable matrices, and $\mathcal{N}_e^r = \{e' | \langle e, r, e'\rangle, \langle e', r, e\rangle \in \mathcal{G}\}$ denotes the set of neighbors of entity e under relation r. Finally, after stacking L times, the output entity embedding $\boldsymbol{v}_e^{(L)}$ from the last R-GCN layer is used as the final entity embedding $\tilde{\boldsymbol{v}}_e$.

Note that, we represent an entity as a set of nodes. For instance, the entity *Iron Man* in Figure 1 will be represented by two nodes: one for the token *Iron* and the other for the token *Man*. This would enhance the generalization ability of KG encoder on unseen entities, since it learns entity embeddings at the token level.

Text Encoder. To obtain the PLM-based entity embeddings, we feed the masked text $\mathcal{Y}_{[mask]}$ into the text encoder, *i.e.,* the lower layers of PLM. As shown in Figure 1, compared with short entity mentions, the masked text contains rich context information about entities. Therefore, similar to masked language model (Devlin et al., 2019), the embeddings of masked text can be computed as:

$$\langle \hat{\boldsymbol{v}}_{w_1}, ..., \hat{\boldsymbol{v}}_{w_T} \rangle = \text{Lower-Layers}(\mathcal{Y}_{[mask]}), \quad (2)$$

where the entity mention m_e corresponds to the embedding sequence $\langle \hat{\boldsymbol{v}}_{w_{s_e}}, ..., \hat{\boldsymbol{v}}_{w_{o_e}} \rangle$ and the PLM-based entity embedding $\hat{\boldsymbol{v}}_e$ can be computed by an average pooling over this embedding sequence.

To bridge the semantic gap, we model the representation alignment by minimizing the Euclidean distance in semantic space between the GNN-based and PLM-based entity embeddings as:

$$\mathcal{L}_{RA} = \sum_{e \in \mathcal{E}} \|\tilde{\boldsymbol{v}}_e - \hat{\boldsymbol{v}}_e\|_2, \tag{3}$$

where $\tilde{\boldsymbol{v}}_e$ and $\hat{\boldsymbol{v}}_e$ are GNN-based and PLM-based entity embeddings, respectively.

With representation alignment, the GNN-based entity embeddings can be aligned with the PLM-based entity embeddings in semantic space, which enables us to effectively inject KG representations into PLM for improving generation quality.

4.2 Knowledge Graph Linearization

To feed the KG into decoder (*i.e.,* the higher layers of PLM), we need to linearize KG into an entity sequence. Previous work (Yang et al., 2020; Ribeiro et al., 2020b) usually relies on random or pre-defined rules, which is not flexible to model KG structures. Here, we propose to utilize breadth-first search (BFS) strategy to traverse KG. BFS, a graph traversal algorithm, starts at the root node and explores all the nodes at the present layer before moving on to the nodes at the next depth layer[1]. Here, we assume that nodes at the same layer potentially express relevant semantics and should be placed in close positions of the entity sequence.

Furthermore, we observe that some relations are often lexicalized before others, *e.g.,* the nationality of a person often precedes the birthplace in descriptive text. Considering such relation priority, in this paper, we propose a *relation-biased breadth first search (RBFS)* strategy to traverse and linearize KG into entity sequence. Specifically, we first compute RBFS weights $\alpha_{e'}$ for each entity e' based on their relations as:

$$\alpha_{e'} = \sigma(\tilde{\boldsymbol{v}}_e^\top \boldsymbol{W}_r^{(L)} \tilde{\boldsymbol{v}}_{e'}), \ \langle e, r, e' \rangle \in \mathcal{G}, \tag{4}$$

where $\boldsymbol{W}_r^{(L)}$ is a relation matrix from Eq. 1. Then, for two sibling entities e' and e'' at the same layer, we traverse e' before e'' if $\alpha_{e'}$ is greater than $\alpha_{e''}$, and vice versa. Finally, through RBFS, we can obtain a linearized entity sequence taken as input of the decoder for text generation.

4.3 KG-enhanced Multi-task Learning

After obtaining the linearized entity sequence, we next take it as input and perform text generation. Different from other text generation tasks, KG-to-text generation aims to generate text reflecting the concise facts in KG. Inspired by Liu et al. (2019), we incorporate an auxiliary KG reconstruction task to reconstruct the facts in KG for learning the semantic correspondence between text and KG.

Text Generation. The text generation task is performed upon the higher layers of PLM. The objective is to maximize the likelihood of the reference text, which is equivalent to minimize the negative log-likelihood as:

$$\mathcal{L}_{LM} = -\sum_{j=1}^{T} \log p_{gen}(w_j | w_1, ..., w_{j-1}; \mathcal{G}), \tag{5}$$

where p_{gen} is the generative probability from PLM. Besides, in KG-to-text generation, some tokens in descriptive text correspond to KG entities shown in Figure 1. The ability to copy entities from KG would enrich the generated text content, which can be achieved by the pointer generator (See et al., 2017). By feeding the hidden states of PLM and the token embedding, the copy probability p^j_{copy} of the j-th token w_j can be computed as:

$$p^j_{copy} = \sigma(\boldsymbol{W}_1 \boldsymbol{s}_j + \boldsymbol{W}_2 \boldsymbol{v}_{w_j} + b_{copy}), \tag{6}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, and b_{copy} are trainable parameters, $\boldsymbol{v}_{w_j}$ is the embedding of token w_j, and $\boldsymbol{s}_j$ is the j-th hidden state from the top layer of PLM. Then, we explicitly "teach" our model how to switch between generation and copy via the copy loss as:

$$\mathcal{L}_{PG} = \sum_{w_j} p^j_{copy} + \sum_{w_k} (1 - p^k_{copy}). \tag{7}$$

Our intuition is aimed at minimizing the copy probability p^j_{copy} of token w_j (generated from vocabulary) and maximizing the copy probability p^k_{copy} of token w_k (copied from KG entities).

KG Reconstruction. Following Song et al. (2020), we formalize the KG reconstruction task as predicting the relations between any two entities. In detail, given a head entity e and a tail entity e' in generated text, we can obtain the hidden states of their mentions from the top layer of decoder, *i.e.,* $\langle \boldsymbol{s}_{s_e}, ..., \boldsymbol{s}_{o_e} \rangle$ and $\langle \boldsymbol{s}_{s_{e'}}, ..., \boldsymbol{s}_{o_{e'}} \rangle$. Then, the entity hidden states $\boldsymbol{h}_e$ and $\boldsymbol{t}_{e'}$ can be computed by an average pooling over their mention hidden states. The probability for a relation r is calculated as:

$$p(r|e, e') = \text{softmax}(\boldsymbol{W}_3[\boldsymbol{h}_e; \boldsymbol{t}_{e'}; \boldsymbol{h}_e \odot \boldsymbol{t}_{e'}] + \boldsymbol{b}_2), \tag{8}$$

---

[1] https://en.wikipedia.org/wiki/Breadth-first_search

where $\boldsymbol{W}_3$ and $\boldsymbol{b}_2$ are trainable parameters. The loss for reconstructing KG is also defined as the negative log-likelihood of all target triples in KG:

$$\mathcal{L}_{GR} = -\sum_{\langle e,r,e' \rangle \in \mathcal{G}} \log p(r|e,e'). \tag{9}$$

By incorporating the KG reconstruction task, our model is able to capture the semantic correspondence between input KG and output text, which further improves generating faithful text.

Finally, the total training loss consists of text generation loss $\mathcal{L}_{LM}$ (Eq. 5), copy loss $\mathcal{L}_{PG}$ (Eq. 7), representation alignment loss $\mathcal{L}_{RA}$ (Eq. 3) and KG reconstruction loss $\mathcal{L}_{GR}$ (Eq. 9) as:

$$\mathcal{L}_{total} = \mathcal{L}_{LM} + \lambda_1 \mathcal{L}_{PG} + \lambda_2 \mathcal{L}_{RA} + \lambda_3 \mathcal{L}_{GR}, \tag{10}$$

where λ_1, λ_2 and λ_3 are combination coefficients.

4.4 Discussion and Learning

In this part, we present the model discussion and the model optimization.

Few-shot Learning. In few-shot KG-to-text generation, the key lies in how to bridge the semantic gap between KG and PLMs with limited dataset. To achieve this goal, we first utilize representation alignment in Section 4.1 to align the semantic space between KG encodings and PLMs, and then introduce a KG reconstruction task in Section 4.3 to further learn the semantic correspondence between input KG and output text. Besides, we observe that KG entities are often multi-word expressions. To deal with unseen entities in few-shot learning, we employ the Byte Pair Encoding (BPE) (Sennrich et al., 2016) and sub-word vocabulary (Radford et al., 2019) to split entity words into smaller semantic units. Our work is also empowered by the excellent few-shot capacities of PLMs with vast amounts of world knowledge learned from large-scale corpora.

Optimization. For PLM, we employ BART-Large model (Lewis et al., 2020). Specially, we adopt the first 6 layers of BART encoder as the lower layers, and the remaining 6 layers of BART encoder and BART decoder as the higher layers. Note that, the target text and text encoder will not be used at test time. In particular, the target text is just used at training time and encoded as PLM-based entity embeddings for representation alignment, while the alignment is not needed at test time. We optimize all parameters according to the total loss in Eq. 10 with the OpenAI AdamW optimizer (Loshchilov and Hutter, 2019). The learning rate, batch size, R-GCN layers and embedding size are set to 1e-5, 20, 2 and 1024, respectively. The weights λ_1, λ_2 and λ_3 in Eq. 10 are set to 0.7, 0.5 and 0.5, respectively, according to performance on validation set. During inference, we apply the beam search method with a beam size of 8.

Dataset	#Train	#Valid	#Test	#Relations
AGENDA	29,720	1,000	10,000	42
WebNLG	7,362	1,389	5,427	107
GenWiki	48,020	1,000	10,000	250

Table 1: Statistics of three datasets.

5 Experiments

In this section, we first set up the experiments, and then report the results and analysis.

5.1 Experimental Setup

Datasets. To evaluate our model on few-shot KG-to-text generation, we conduct experiments on three benchmarks, including AGENDA (Koncel-Kedziorski et al., 2019), WebNLG (Gardent et al., 2017) and GenWiki Fine (Jin et al., 2020). We adopt three large domains (*i.e., Airport*, *Building* and *Food*) for WebNLG and two large domains (*i.e., Sports* and *Games*) for GenWiki. Table 1 shows the statistics for each dataset. Each instance of these datasets contains a knowledge graph in the form of triples and a target text describing the graph. The three datasets have originally provided the alignment records from entity mentions to KG entities. Take an example from WebNLG dataset "AGENT-1 is located in PATIENT-1": the entity mention is tagged as "AGENT-1" and the tag "AGENT-1" maps to the entity "11th_Mississippi_Infantry_Monument" in KG. If such alignments are not available, we can utilize entity linking tools (*e.g.,* NER packages) for preprocessing.

Baselines. We make a comparison against five KG-to-text generation models:

• *GraphWriter* (Koncel-Kedziorski et al., 2019) introduces a graph transformer encoder and a sequence decoder for generating text based on KG.

• *CGE-LW* (Ribeiro et al., 2020a) proposes a graph-to-text model by combining both global and local node aggregation strategies.

Datasets	AGENDA				WebNLG				GenWiki Fine			
#Metrics	B-4	R-L	CIDEr	Chrf	B-4	R-L	CIDEr	Chrf	B-4	R-L	CIDEr	Chrf
GraphWriter	15.30	22.03	0.24	38.33	45.84	60.62	3.14	55.53	29.73	55.46	2.68	46.87
CGE-LW	18.01	25.62	0.33	46.69	48.60	62.52	3.85	58.66	30.67	56.37	3.20	47.79
CycleGT	20.16	25.77	0.69	48.26	50.20	68.30	3.81	68.91	38.57	59.37	3.50	62.46
BART-base	22.01	26.44	0.90	48.02	49.81	63.10	3.45	67.65	48.20	59.21	4.02	65.80
BART-large	23.65	28.76	1.15	50.44	52.49	65.61	3.50	72.00	**50.70**	61.90	4.51	68.15
T5-base	20.59	29.41	0.81	48.15	48.86	65.57	3.99	66.08	45.72	58.28	3.74	65.68
T5-large	22.15	30.68	0.87	48.88	58.78	68.22	4.10	74.40	47.11	60.64	3.74	**68.47**
Ours	**25.15**	**35.12**	**3.23**	**55.89**	**61.88**	**75.74**	**6.03**	**79.10**	48.46	**65.65**	**5.19**	64.00

Table 2: Performance comparisons of different methods for fully-supervised KG-to-text generation under three domains. B-n and R-n are short for BLEU-n and ROUGE-n. **Bold** and underline fonts denote the best and the second best methods (the same as below).

Datasets	AGENDA				WebNLG				GenWiki Fine			
#Instances	50	100	200	500	50	100	200	500	50	100	200	500
BART-large	5.71	6.15	7.59	10.71	9.05	15.70	19.38	27.91	9.14	13.38	15.39	24.14
T5-large	2.69	2.73	4.65	7.52	7.18	14.52	16.88	21.68	6.30	6.36	10.37	17.72
Ours	**6.22**	**9.40**	**10.21**	**17.93**	**10.60**	**17.46**	**20.00**	**31.79**	**10.75**	**14.44**	**16.84**	**28.89**

Table 3: BLEU-4 results of different methods for few-shot KG-to-text generation under three domains. To mitigate the randomized effects of samples, we report the average results over five training runs (the same as below).

Datasets	AGENDA				WebNLG				GenWiki Fine			
#Instances	50	100	200	500	50	100	200	500	50	100	200	500
BART-large	14.33	15.28	16.94	20.70	22.57	26.21	30.68	49.34	26.59	29.60	34.56	47.50
T5-large	14.11	14.17	15.88	21.72	20.80	22.71	24.18	38.36	21.02	21.36	20.07	35.72
Ours	**15.10**	**16.65**	**18.88**	**25.72**	**24.80**	**28.38**	**33.12**	**55.13**	**28.02**	**31.36**	**38.07**	**50.72**

Table 4: ROUGE-L results of different methods for few-shot KG-to-text generation under three domains.

- *CycleGT* (Guo et al., 2020) jointly learns two dual tasks (graph-to-text generation and text-to-graph relation classification) via cycle training.
- *BART-Base/Large* (Ribeiro et al., 2020b) linearizes the KG into sequence and applies BART-Base/Large (Lewis et al., 2020) to generate text.
- *T5-Base/Large* (Ribeiro et al., 2020b) linearizes KG into a triple sequence and employs T5-Base/Large (Raffel et al., 2020) to generate text.

Among these baselines, *GraphWriter* and *CGE-LW* are GNN-based generation models; *CycleGT* is an unsupervised model using cycle training; *GPT2-Base/Large* and *BART-Base/Large* are the most relevant comparisons, which also employ PLMs in KG-to-text generation. These baselines were trained on the whole training dataset, *i.e.,* all KG-text pairs. Following previous few-shot work (Chen et al., 2020), we train our model on different few-shot settings with training dataset size ranging from 50, 100, 200 to 500. All the comparison methods are optimized based on validation performance. In our model, the entity embeddings of GNN are initialized with pretrained KG embeddings and the GNN weights are transferred from CGE-LW. We also pretrain GNN weights based on the large-scale KG, *i.e.,* Wikipedia. Based on the pretrained entity embeddings and weights, we continue to train our model.

Evaluation Metrics. For performance comparison, we adopt five automatic evaluation metrics widely used by previous graph-to-text work (Guo et al., 2020), *i.e.,* BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015) and CHRF++ (Popovic, 2015). Specifically, BLEU-n and ROUGE-n compute the ratios of overlapping n-grams between generated and real text, CIDEr computes the TF-IDF weights for each n-gram in generated/real text, and CHRF++ computes F-score averaged on both character- and word-level n-grams.

Models	B-4	R-L	CIDEr	Chrf
Ours	**31.79**	**55.13**	**3.94**	**57.38**
w/o RA	23.14	41.34	1.90	43.34
w/o GR	27.56	46.69	2.82	48.90
w/o PG	29.30	48.66	3.58	53.44

Table 5: Ablation analysis on WEBNLG dataset.

5.2 Main Results

Table 2, 3, and 4 present the fully-supervised and few-shot results of our model and other baselines, respectively.

First, by combining global and local entity context, CGE-LW performs better than GraphWriter. Furthermore, with two elaborate designed dual tasks, CycleGT becomes the best non-PLM baseline, outperforming GraphWriter and CGE-LW.

Second, as the most direct comparison with our model, BART-Base/Large and T5-Base/Large perform better than baselines by leveraging encoded semantics in PLMs, which reveals the feasibility of utilizing PLMs for KG-to-text generation.

Finally, we observe that our model achieves the best performance on both fully-supervised and few-shot settings. Large-scale PLMs can encode world knowledge by reading a large amount of text, making it easier to recover KG facts. Given only a handful of examples, the performances of baselines drop drastically, while the performance of our model only descents slightly. Furthermore, with only 500 labelled instances, our model improves over CGE-LW and CycleGT, and achieves the best performance in most cases. Compared to these PLM-based KG-to-text baselines, we adopt GNN to explicitly encode KG structure and representation alignment to bridge the semantic gap between PLM and GNN. This helps produce effective semantic representations for few-shot learning. Furthermore, we incorporate an auxiliary KG reconstruction task to learn semantic correspondence between input KGs and output text. These results indicate that our model can achieve more superior performance on KG-to-text generation task in a few-shot setting.

5.3 Detailed Analysis

Next, we conduct detailed analysis experiments on our model. We only report the test results on WEBNLG dataset with 500 training instances due to similar findings in other datasets.

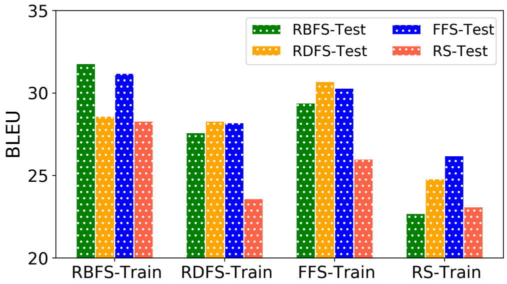

Figure 3: Linearization analysis on WEBNLG dataset.

Models	#Supp.↑	#Cont.↓	Naturalness↑
Gold	4.40	0.36	4.26
Ours	3.77	1.01	3.96
BART-Large	3.20	1.90	3.55
CEG-LW	2.87	2.13	2.56

Table 6: Human evaluation on WEBNLG dataset. Cohen's kappa coefficients for labelling three factors are as follows: 0.78, 0.71, and 0.75.

Ablation Analysis. In our ablation study, we evaluate the effect of each loss $\mathcal{L}_{PG}$, $\mathcal{L}_{RA}$ and $\mathcal{L}_{GR}$ on the overall model performance. Here, we consider three variants:

- *w/o PG*: the variant removes the copy loss $\mathcal{L}_{PG}$.
- *w/o RA*: the variant removes the representation alignment loss $\mathcal{L}_{RA}$.
- *w/o GR*: the variant removes the KG reconstruction loss $\mathcal{L}_{GR}$.

As can be seen from Table 5, by removing any of the three losses, the BLEU/ROUGE/CIDEr performance drops compared to the complete model, especially removing $\mathcal{L}_{RA}$ and $\mathcal{L}_{GR}$. The proposed representation alignment bridges the semantic gap between PLM and GNN, which is helpful for adapting KG representations to PLM. The KG reconstruction task learns the correspondence between KG and text ensuring faithful generation about KG. We also observe a small performance drop by removing $\mathcal{L}_{PG}$. It is likely because PLM has learned some common phrase expressions about these KG facts from large-scale pretraining corpus.

KG Linearization Analysis. In Section 4.2, we propose a novel relation-biased BFS (RBFS) strategy to linearize the input KG into entity sequence. To verify the effectiveness of this strategy, we conduct linearization analysis by comparing RBFS with three traversal strategies, including relation-biased depth-first search (RDFS), forest fire search (FFS) and random search (RS). Specifically, RDFS combines both DFS and the relation factor similar

Real	Knowledge Graph	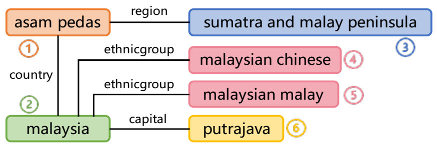	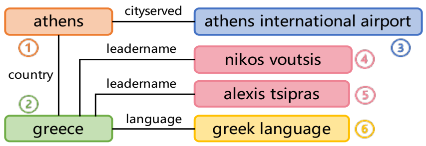
	Reference	**asam pedas** is a food found in the region of **sumatra and malay peninsula** in **malaysia** , the capital of which is **putrajaya** , and whose ethnic groups include **malaysian malay** and **malaysian chinese** .	**athens international airport** serves the city **athens** in **greece** , **greek language** is spoken in greece and the leaders names in greece are **alexis tsipras** and **nikos voutsis** .
BART	Linearized KG	①③→①②→①⑥→②⑤→②④	①②→②④→②⑤→①③→②⑥
	Generated Text	**asam pedas** is a dish from **malaysia and sumatra** where the capital is **putrajava** . **malaysian malay** and **chinese** are ethnic groups in sumatra .	**athens** in **greece** is led by **alexis tsipras** and is served by **athens international airport** greece speaks **greek language** .
Ours	Linearized KG	①→③→②→⑥→⑤→④	①→③→②→⑥→⑤→④
	Generated Text	**asam pedas** comes from the region of **sumatra and malay peninsula** in **malaysia** , where the capital is **putrajava** , **malaysian malay** and **malaysian chinese** are ethnic groups .	**athens** is served by **athens international airport** in **greece** , which speaks **greek** textbflanguage . greece is led by **alexis tsipras** and **nikos voutsis** .

Table 7: Sample text generated by BART-Large baseline and our model from the *Food* and *Airport* domains of the WEBNLG benchmark. Since BART linearizes KG as triple sequence and an entity may involve in several triples, there are repeated entities used by BART (we omit the relations between entities). **Bold** and underlined words correspond to entity words and keywords.

to RBFS, where DFS starts at the root node and explores as far as possible along each branch before backtracking[2]; FFS is a randomized version of RBFS randomly exploring all the nodes at the same layer (Leskovec and Faloutsos, 2006); and RS randomly traverses all the nodes in the input KG. By re-training our model with the above three strategies, we report the comparison of BLEU results in Figure 3. It can be observed that, RBFS and FFS strategies achieve better results compared to the rest strategies. Nodes at the same layer tend to express more relevant semantics, thus searching by layer could produce more reasonable and coherent entity sequence especially considering the relations of entities as our RBFS strategy.

Human Evaluation. Following previous work in data-to-text (Chen et al., 2020), we conduct human evaluation on the generated text. We randomly sample 200 KG subgraphs along with corresponding generated text from CGE-LW, BART-Large and our model. In order to reduce the variance caused by human, three workers were asked to score the text with respect to two aspects: *Factual correctness* and *Language naturalness*. The first criterion evaluates how well the generated text correctly conveys information in the KG, by counting the number of facts in text supported by the KG (denoted as #Supp.) and contradicting with or missing from the KG (denoted as #Cont.). The second criterion evaluates whether the generated text is grammatically correct and fluent. The scoring mechanism adopts a 5-point Likert scale (Likert, 1932), ranging from 1-point (“very terrible”) to 5-point (“very satisfying”). We further average the three scores from the three human judges over the 200 inputs. The results in Table 6 show that our model produces more fidelity and fluent texts than previous models. In our approach, the KG reconstruction task and pointer generator enhance the awareness of KG facts and alleviate producing incorrect facts. Also, with some learned common phrase expressions in PLMs, our model can generate natural text while keeping fidelity.

Qualitative Analysis. In this part, we present intuitive explanations why our model performs well. Table 7 presents two descriptions and the corresponding generated entity sequences and texts by BART-Large baseline and our model. As we can see, based on KG linearization, the generated texts by our model show reasonable and similar content sketch with real texts (*e.g., peninsula (region)→malaysia (country)→putrajava (capital)*).

---

[2] https://en.wikipedia.org/wiki/Depth-first_search

Besides, the baseline model incorrectly merges and generates unfaithful facts (*e.g., malaysia and sumatra*) or misses facts (*e.g., nikos voutsis*), while our model describes all the KG facts correctly. This improvement could be attributed to the KG reconstruction task, which enables our model to learn the correspondence between the input KG facts and output text. Finally, the entity words in our generated text are enriched and connected by meaningful keywords (*e.g.,* entity *greek language* and keyword *speaks*). The reason might be that, with the help of representation alignment, the GNN entity embeddings are aligned with the PLM word embeddings.

6 Conclusion

This paper presented a few-shot KG-to-text generation model based on PLMs. We make three important technical contributions, namely representation alignment for bridging the semantic gap between KG encodings and PLMs, relation-biased KG linearization for deriving better input KG representations, and multi-task learning for learning the correspondence between KG and text. Extensive experiments on three benchmark datasets demonstrate the effectiveness of our few-shot KG-to-text generation model. As future work, we will consider adopting KG-enhanced PLMs (Zhang et al., 2019; Peters et al., 2019) for improving the task performance, which explicitly inject knowledge information into PLMs.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, Beijing Academy of Artificial Intelligence (BAAI) under Grant No. BAAI2020ZJ0301, Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China under Grant No.18XNLG22 and 19XNQ047. Xin Zhao is the corresponding author.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 183–190. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime G. Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *NAACL HLT 2016, San Diego California, USA, June 12-17, 2016*, pages 731–739. The Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv:2006.04702*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR*, abs/2002.00388.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2398–2409. International Committee on Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2284–2293. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1503–1514. ACL.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213. The Association for Computational Linguistics.

Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 631–636. ACM.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Gaole He, Jinhao Jiang, Xiaoxuan Hu, Puzhao Xie, Zhipeng Chen, Zhuohao Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2021a. TextBox: A unified, modularized, and extensible framework for text generation. In *ACL*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021b. Pretrained language models for text generation: A survey. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021*.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *AAAI 2019, IAAI 2019, AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6786–6793. AAAI Press.

Robert Logan, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2267–2277. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP, Hong Kong, China, November 3-7, 2019*, pages 43–54. Association for Computational Linguistics.

Maja Popovic. 2015. chrf: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT@EMNLP 2015, 17-18 September 2015, Lisbon, Portugal*, pages 392–395. The Association for Computer Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020a. Modeling global and local node contexts for text generation from knowledge graphs. *Trans. Assoc. Comput. Linguistics*, 8:589–604.

Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schutze, and Iryna Gurevych. 2020b. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7987–7998. Association for Computational Linguistics.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4566–4575. IEEE Computer Society.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zixiaofan Yang, Arash Einolghozati, Hakan Inan, Keith Diedrick, Angela Fan, Pinar Donmez, and Sonal Gupta. 2020. Improving text-to-text pretrained models for the graph-to-text task. In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020), Dublin, Ireland (Virtual). Association for Computational Linguistics*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics.

Better Robustness by More Coverage: Adversarial and Mixup Data Augmentation for Robust Finetuning

Chenglei Si[1,3*], **Zhengyan Zhang**[2,3*], **Fanchao Qi**[2,3], **Zhiyuan Liu**[2,3,4†],
Yasheng Wang[5], **Qun Liu**[5], **Maosong Sun**[2,3,4]
[1]University of Maryland, College Park, MD, USA
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]Beijing National Research Center for Information Science and Technology
[4]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[5]Huawei Noah's Ark Lab
`clsi@terpmail.umd.edu, zy-z19@mails.tsinghua.edu.cn`

Abstract

Pretrained language models (PLMs) perform poorly under adversarial attacks. To improve the adversarial robustness, adversarial data augmentation (ADA) has been widely adopted to cover more search space of adversarial attacks by adding textual adversarial examples during training. However, the number of adversarial examples for text augmentation is still extremely insufficient due to the exponentially large attack search space. In this work, we propose a simple and effective method to cover a much larger proportion of the attack search space, called Adversarial and Mixup Data Augmentation (AMDA). Specifically, AMDA linearly interpolates the representations of pairs of training samples to form new virtual samples, which are more abundant and diverse than the discrete text adversarial examples in conventional ADA. Moreover, to fairly evaluate the robustness of different models, we adopt a challenging evaluation setup, which generates a new set of adversarial examples targeting each model. In text classification experiments of BERT and RoBERTa, AMDA achieves significant robustness gains under two strong adversarial attacks and alleviates the performance degradation of ADA on the clean data. Our code is available at: `https://github.com/thunlp/MixADA`.

1 Introduction

Pretrained language models (PLMs) have established state-of-the-art results on various NLP tasks (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020) and the pretraining-then-finetuning paradigm has become the status quo. However, recent works have shown the adversarial vulnerabilities of PLMs, where PLMs finetuned on various downstream datasets are fooled by different types of adversarial attacks (Jin et al., 2020; Zang et al., 2020; Si et al., 2021; Li et al., 2020; Garg and Ramakrishnan, 2020; Wang et al., 2020a).

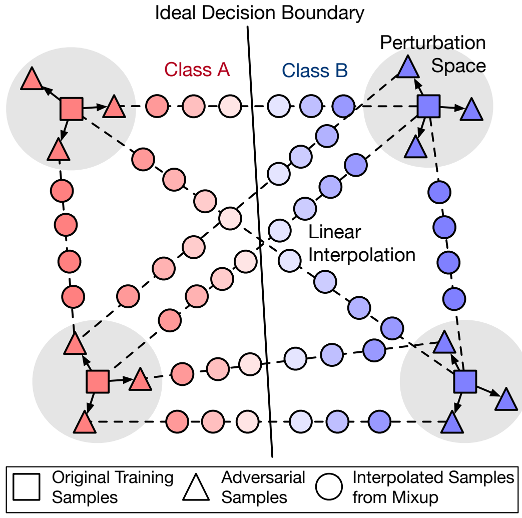

Figure 1: Illustration of MixADA. Some of the interpolated samples are shown. We interpolate the representations of each pair of training samples including original samples and adversarial samples. Blue and red represent two different classes. The solid line represents the resultant decision boundary. AMDA helps achieve a more robust decision boundary.

To improve adversarial robustness, two types of defense strategies have been proposed. The first type targets at specific attacks, such as spelling correction modules and pretraining tasks to defend character-level attacks (Pruthi et al., 2019; Jones et al., 2020; Ma et al., 2020) and certified robustness for word-substitution attacks (Huang et al., 2019; Jia et al., 2019). However, they are limited in practice as they are not generally applicable to other types of attacks. The other type of defense is Adversarial Data Augmentation (ADA), which augments the training set by the adversarial examples and is widely used in the training (finetuning) process to enhance model robustness (Alzantot et al.; Ren et al., 2019; Zhang et al., 2020; Jin et al., 2020;

---

[*] Equal contribution

[†] Corresponding author email: liuzy@tsinghua.edu.cn

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1569–1576
August 1–6, 2021. ©2021 Association for Computational Linguistics

Li et al., 2020; Tan et al., 2020; Yin et al., 2020; Zheng et al., 2020; Zou et al., 2020; Wang et al., 2020b). ADA is generally applicable to any type of adversarial attacks but is not very effective in improving model performance under attacks. In this work, we aim to improve ADA and devise a general defense strategy to effectively improve model robustness during finetuning.[1]

ADA has two major limitations for NLP models. Firstly, unlike images, it is harder to create new augmented textual data due to their discrete nature. Moreover, for textual adversarial attacks, the attack search space is prohibitively large. For example, the search space of word-substitution attacks consists of all combinations of the synonym replacement candidates, which is exponentially large. Consequently the number of adversarial training examples for augmentation is very insufficient. Secondly, ADA usually causes significant performance degradation on the clean data because the distribution of adversarial examples is very different from that of the clean data (Ren et al., 2019).

In order to solve these two limitations, we create additional training samples via interpolating existing samples (Figure 1). How to interpolate discrete textual inputs is non-trivial. We propose to convert the discrete textual inputs into continuous representations and then perform both ADA and mixup augmentation (Zhang et al., 2018; Guo et al., 2019), which is an augmentation technique proven to be particularly effective on continuous image data (Lamb et al., 2019; Pang et al., 2020). We name our method Adversarial and Mixup Data Augmentation (AMDA). With AMDA, we can create a much larger number of augmented training samples that cannot be obtained via discrete perturbations on textual data. Moreover, AMDA's interpolated virtual training samples are closer to the distribution of the original data, which alleviates the performance degradation problem of ADA.

We experiment AMDA on three text classification datasets under two strong adversarial attacks and find that AMDA achieves significant robustness gains in all cases, notably restoring RoBERTa after-attack accuracy from 6.35% to 51.84% on IMDB, outperforming all other baselines by large margins. Moreover, we also examine the evaluation method for adversarial robustness. Specifically, we find that the widely adopted *Static Attack Evaluation* where a fixed set of adversarial examples are used to test all models is not reliable. In order to test model robustness under targeted attacks (i.e., not model-agnostic), we adopt the more challenging *Targeted Attack Evaluation* where we generate a new set of targeted adversarial examples to evaluate each model. We encourage future defense works to also adopt this more reliable and challenging evaluation setting.

2 Method

In AMDA, we first augment training samples with ADA and then perform mixup during model training, where mixup augmentation is applied on the ADA-augmented training set.

2.1 Adversarial Data Augmentation

Given a victim model f_v and the original training instances $\mathbf{D}_{ori} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, we employ an attacker to construct label-preserving adversarial training instances $\mathbf{D}_{adv} = \{(\mathbf{x}'_i, \mathbf{y}_i)\}_{i=1}^{n}$ such that: instances originally correctly classified are now classified wrongly ($f_v(\mathbf{x}'_i) \neq f_v(\mathbf{x}_i)$). We then train the model on the augmented training data $\mathbf{D}_{ADA} = \mathbf{D}_{ori} \cup \mathbf{D}_{adv}$.

2.2 Mixup Data Augmentation

To better defend against the large number of possible adversarial examples, we propose to perform additional mixup augmentation during training. Specifically, we linearly interpolate the representations and labels of pairs of training samples to create different virtual training samples, which can be formulated as:

$$\begin{aligned} \hat{\mathbf{x}} &= \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \\ \hat{\mathbf{y}} &= \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j, \end{aligned} \tag{1}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ are two labeled examples, and $\lambda \in [0, 1]$ comes from a beta distribution $\lambda \sim Beta(\alpha, \alpha)$, where α is a hyperparameter. On textual data, we cannot directly mix the discrete tokens. Instead, we can either interpolate the word embedding vectors or models' hidden representations of textual inputs. Meanwhile, we directly interpolate the labels, which are represented as one-hot vectors.

When applied together with adversarial data augmentation, we allow the mixing of different types

---

[1] In this paper, we refer to such discrete adversarial training method as adversarial data augmentation to avoid confusion with the gradient-based adversarial training methods (Miyato et al., 2017), which has been shown to be ineffective in defending against textual adversarial attacks (Li and Qiu, 2021).

of data (between original examples, between original examples and adversarial examples, and between adversarial examples) to increase diversity.

2.3 AMDA

In our proposed Adversarial and Mixup Data Augmentation (AMDA), we train the new model f on the augmented training data $\mathbf{D}_{AMDA}$, which is obtained by performing both adversarial data augmentation and mixup data augmentation. We minimize the sum of the standard training loss and the mixup loss:

$$L = \sum_{i=1}^{n} L_{CE}(f(\mathbf{x}_i), \mathbf{y}_i) + \sum_{i=1}^{m} L_{KL}(f(\hat{\mathbf{x}}_i), \hat{\mathbf{y}}_i), \tag{2}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ is from $\mathbf{D}_{ADA}$ and $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is the virtual example obtained by applying mixup on the random pair of training data sampled from $\mathbf{D}_{ADA}$. We use cross-entropy to compute loss on $(\mathbf{x}_i, \mathbf{y}_i)$ and use KL-divergence for loss on $(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_i)$.

3 Robustness Evaluation

There are two different ways of robustness evaluation under adversarial attacks used in previous works. In this work, we explicitly differentiate them as Static Attack Evaluation (SAE) and Targeted Attack Evaluation (TAE):

SAE generates a fixed set of adversarial examples on the original model as the victim model. This fixed adversarial test set will then be used to evaluate all the new models. This evaluation setup has been adopted in (Ren et al., 2019; Tan et al., 2020; Yin et al., 2020; Wang et al., 2020b; Zou et al., 2020; Wang et al., 2021, *inter alia.*).

TAE re-generates a new set of adversarial examples to target every model being evaluated. This is adopted in (Zhang et al., 2020; Huang et al., 2019; Jia et al., 2019; Li et al., 2020; Zang et al., 2020; Zheng et al., 2020; Li and Qiu, 2021, *inter alia.*)

We observe that some authors did not explicitly specify the mode of evaluation in their papers[2], leading to confusion and even conflicting conclusions. Thus, we explicitly differentiate the two modes of evaluation and provide a comparison in our experiments.

4 Experiments

4.1 Experiment Setups

Datasets. We evaluate our methods on three text classification datasets: two sentiment analysis datasets: SST-2 (Socher et al., 2013) and IMDB (Maas et al., 2011), where both datasets are binary classification tasks; as well as a multi-class news classification dataset AGNews (Zhang et al., 2015), which consists of four different classes. For SST-2, we attack the entire test set (1821 samples) and report the accuracy under attacks. For IMDB, we find that it is prohibitively slow to attack the whole test set (25k samples) and hence we use the subset of the original test set as released in Gardner et al. for faster evaluation, which consists of 488 test instances. Similarly, on AGNews, we randomly sampled 10% of the original test set and hold out as the test samples for attack evaluation. We also include these data splits in our released code base for easy reproduction and fair comparison for future works.

Victim models and attack methods. We experiment with both BERT-base-uncased (Devlin et al., 2019) and RoBERTa-base (Liu et al., 2019) as the victim models. We use PWWS (Ren et al., 2019) and TextFooler (Jin et al., 2020) as our attack methods, which have been shown to effectively attack state-of-the-art NLP models including PLMs such as BERT. Both attack algorithms have access to model predictions but not gradients, and iteratively search for word synonym substitutes that flip model predictions without drastically changing the original semantic meanings and golden labels.

Details of mixup. When performing mixup, we mix hidden representations of upper layers of BERT. The vectors used for mixup are hidden representations of the input examples at layer i of the Transformer encoder, where i is randomly sampled from $\{7, 9, 12\}$, which was found to be empirically effective (Chen et al., 2020). Furthermore, we explore two different ways of obtaining the hidden representations of input examples from PLMs like BERT: (1) We use the vector of the `[CLS]` token at the ith-layer of BERT as the hidden representation for mixing. We name this approach **SMix**. (2) We perform mixup on every token's vector representation at the ith-layer. We name this approach **TMix**, which is the approach taken by Chen et al. (2020).

Details of ADA and AMDA. For both ADA and AMDA, we generate and add the corresponding adversarial examples of PWWS and TextFooler into

---

[2] We had to email some of the authors to clarify the evaluation setup being adopted.

	SST-2					IMDB				
	Original	PWWS-d	PWWS-s	TF-d	TF-s	Original	PWWS-d	PWWS-s	TF-d	TF-s
BERT$_v$	92.04	19.17	19.17	5.66	5.66	97.34	23.36	23.36	3.48	3.48
BERT$_{r1}$	91.10	18.73	44.98	3.46	45.36	96.72	25.61	69.88	1.64	76.64
BERT$_{r2}$	90.94	20.26	45.63	2.97	45.80	97.13	30.12	65.78	2.46	76.23
RoBERTa$_v$	94.45	25.48	25.48	3.29	3.29	97.75	15.98	15.98	1.84	1.84
RoBERTa$_{r1}$	94.29	31.03	50.47	5.82	41.63	97.54	27.46	65.57	3.48	77.46
RoBERTa$_{r2}$	93.85	32.13	50.69	9.34	40.91	97.34	17.21	73.36	2.87	76.64

Table 1: Comparison between dynamic and static evaluation. PWWS-d, PWWS-s, TF-d, TF-s represent PWWS dynamic, PWWS static, TextFooler dynamic, TextFooler static, respectively. Numbers in the table represent accuracy. BERT$_v$ and RoBERTa$_v$ are the victim model for generating static evaluation examples. BERT$_{r1}$, BERT$_{r2}$, RoBERTa$_{r1}$, and RoBERTa$_{r2}$ are the fine-tuned models with new random seeds.

	SST-2				IMDB			
	PWWS		TextFooler		PWWS		TextFooler	
	Original	Adversarial	Original	Adversarial	Original	Adversarial	Original	Adversarial
BERT	91.27	14.83 *(20.88%)*	91.27	2.97 *(16.21%)*	97.75	24.18 *(24.10%)*	97.75	1.64 *(10.18%)*
+ADA	90.12	27.18 *(24.46%)*	90.50	9.01 *(18.32%)*	96.93	25.82 *(34.53%)*	96.93	3.07 *(11.81%)*
+TMix	91.82	21.20 *(19.36%)*	91.82	3.51 *(16.39%)*	97.13	43.24 *(32.51%)*	97.13	0.00 *(12.06%)*
+SMix	91.82	22.52 *(20.47%)*	91.82	4.61 *(16.76%)*	97.13	31.97 *(23.74%)*	97.13	2.66 *(12.39%)*
+AMDA-TMix	91.54	**38.82** *(23.73%)*	91.93	13.23 *(19.66%)*	97.34	51.02 *(36.76%)*	96.72	4.51 *(17.23%)*
+AMDA-SMix	91.10	31.52 *(24.11%)*	92.15	**17.35** *(18.64%)*	96.72	**60.86** *(27.79%)*	96.72	**17.42** *(13.85%)*
RoBERTa	94.62	28.39 *(23.06%)*	94.62	5.44 *(18.51%)*	97.54	28.07 *(37.48%)*	97.54	6.35 *(12.61%)*
+ADA	94.07	25.26 *(27.07%)*	92.75	9.67 *(19.71%)*	97.54	24.80 *(49.36%)*	96.93	12.50 *(14.39%)*
+TMix	94.18	30.04 *(23.19%)*	94.18	11.04 *(17.69%)*	97.54	44.06 *(39.33%)*	97.54	21.11 *(14.01%)*
+SMix	93.96	31.52 *(22.86%)*	93.96	8.29 *(17.80%)*	97.34	41.39 *(34.90%)*	97.34	22.34 *(11.96%)*
+AMDA-TMix	93.90	36.74 *(26.02%)*	93.03	13.78 *(20.15%)*	98.57	50.41 *(59.68%)*	97.13	**51.84** *(16.62%)*
+AMDA-SMix	93.96	**41.85** *(27.17%)*	93.47	**16.80** *(21.88%)*	97.54	**55.12** *(45.30%)*	97.54	49.18 *(15.52%)*

Table 2: Accuracy of the various models under PWWS and TextFooler attacks. Best performance for BERT-based models and RoBERTa-based models under each attack is **boldfaced**, the second best performance is underlined. *Numbers* in brackets indicate the average word modification rate of each attack.

	PWWS		TextFooler	
	Orig.	Adv.	Orig.	Adv.
RoBERTa	94.34	47.50	94.34	25.53
+ADA	93.55	66.97	94.08	44.61
+TMix	94.08	45.66	94.08	26.58
+SMix	94.08	45.13	94.08	22.63
+AMDA-TMix	94.47	69.74	93.95	**56.32**
+AMDA-SMix	94.34	**70.00**	93.42	51.32

Table 3: Results on AG News multi-class classification dataset, with RoBERTa model. Best performance under each attack is **boldfaced**, the second best performance is underlined.

training. For comparison, we also experiment with mixup alone without adding the adversarial examples. In this case, the model would only interpolate pairs of original training examples. We perform a greedy hyper-parameter search for the amount of augmented adversarial training samples and mixup parameter α as described in the Appendix. We also report average word modification rates, which indicate the percentage of words being replaced for attacking. Higher word modification rates indicate that the model is harder to attack and hence needs more words to be replaced.

4.2 Comparison of SAE and TAE

To compare SAE and TAE, we attack the fine-tuned model (BERT$_v$), RoBERTa$_v$ as the victim on SST-2 and IMDB, and then use the generated adversarial test set as the fixed test set for SAE. We then change the random seeds and re-finetune the models on the same data (BERT$_{r1}$, BERT$_{r2}$, RoBERTa$_{r1}$, RoBERTa$_{r2}$) with all other hyper-parameters being the same. We evaluate all these models using both SAE and TAE. The results are shown in Table 1.

We find that by simply changing the random seeds, models achieve significant improvement under SAE. However, when we re-generate the adversarial test set for each model, their performances under TAE stay consistently poor. Moreover, we train BERT and RoBERTa with ADA and find that although BERT$_{\mathrm{ADA}}$ and RoBERTa$_{\mathrm{ADA}}$ perform well under SAE, they still perform poorly under TAE. This shows that conventional ADA is actually ineffective in improving model robustness under the challenging TAE setting. We conclude that the adversarial examples found by the attackers target specifically at the victim models, hence they cannot fully reveal weaknesses of new models even if they only differ in random seeds. We believe that TAE is the more challenging and meaningful evaluation method to measure model robustness under targeted attacks. We adopt TAE for the rest of the experiments in this paper and encourage future works to do so for fair comparison.

4.3 Mixup Improves Robustness

The comparison of AMDA and baseline methods under attacks for SST-2 and IMDB is shown in Table 2. The results on the AGNews dataset with RoBERTa model is shown in Table 3. We observe that: (1) Mixup alone (both TMix and SMix) can often improve model robustness. For example, TMix and SMix improve the robust accuracy significantly under both attacks when using RoBERTa on IMDB. (2) AMDA (both AMDA-TMix and AMDA-SMix) can achieve further robustness improvement as compared to ADA and mixup in all cases. This proves that mixup and ADA can complement each other to better improve model robustness under adversarial attacks. (3) Compared to ADA, our AMDA method does not incur significant performance degradation on the original test sets while improving robustness. In some cases, for example, BERT+TMix and BERT+AMDA-TMix even improve the model performance on the original test sets. This benefit is likely because that mixup creates virtual examples that are closer to the empirical data distribution. (4) We find that models trained with AMDA also incur higher word modification rates under both attacks. For example, RoBERTa+AMDA-TMix incurs 59.68% word modification rate under PWWS attack, while the RoBERTa baseline only needs 37.48% words to be replaced. This further demonstrates that our proposed method improves robustness.

5 Conclusion

In this work, we propose AMDA as a generally applicable defense strategy by combining both adversarial and mixup data augmentation to cover more of the attack space. We show that AMDA greatly improves PLMs' robustness under the challenging TAE evaluation setting under two strong adversarial attacks. We leave a more thorough theoretical analysis of AMDA's effectiveness on textual data as future work.[3] We believe that our work can establish the appropriate evaluation protocol and offer a competitive baseline for future works on improving the robustness of PLMs.

Acknowledgments

We thank members of THUNLP for their helpful discussion and valuable feedback on our work. This work is supported by the National Key Research and Development Program of China (No. 2020AAA0106501) and Beijing Academy of Artificial Intelligence (BAAI).

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of EMNLP 2018*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mix-Text: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of ACL 2020*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL 2019*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Proceedings of EMNLP 2020 (Findings)*.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of EMNLP 2020*.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting Data with Mixup for Sentence Classification: An Empirical Study. *arXiv*.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019.

---

[3] Zhang et al. (2021) provided a theoretical proof of mixup's effectiveness on continuous image data, which may serve as a foundation for more theoretical work.

Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of EMNLP 2019*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified Robustness to Adversarial Word Substitutions. In *Proceedings of EMNLP 2019*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proceedings of AAAI 2020*.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust Encodings: A Framework for Combating Adversarial Typos. In *Proceedings of ACL 2020*.

Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. 2019. Interpolated Adversarial Training: Achieving Robust Neural Networks Without Sacrificing Too Much Accuracy. In *Proceedings of AISec@CCS*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of ICLR 2020*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of EMNLP 2020*.

Linyang Li and Xipeng Qiu. 2021. TAVAT: Token-Aware Virtual Adversarial Training for Language Understanding. In *Proceedings of AAAI 2021*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining approach. *arXiv*.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: Character-aware pre-trained language model. In *Proceedings of COLING 2020*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL 2011*.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial Training Methods for Semi-Supervised Text Classification. In *Proceedings of ICLR 2017*.

Tianyu Pang, Kun Xu, and Jun Zhu. 2020. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *Proceedings of ICLR 2020*.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating Adversarial Misspellings with Robust Word Recognition. In *Proceedings of ACL 2019*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *Proceedings of ACL 2019*.

Chenglei Si, Ziqing Yang, Yiming Cui, Wentao Ma, Ting Liu, and Shijin Wang. 2021. Benchmarking Robustness of Machine Reading Comprehension Models. In *Proceedings of ACL 2021 (Findings)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of ACL 2020*.

Boxin Wang, Hengzhi Pei, Boyuan Pan, Qian Chen, Shuohang Wang, and Bo Li. 2020a. T3: Tree-autoencoder constrained adversarial text generation for targeted attack. In *Proceedings of EMNLP 2020*.

Boxin Wang, Shuohang Wang, Y. Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. 2021. InfoBERT: Improving Robustness of Language Models from An Information Theoretic Perspective. In *Proceedings of ICLR 2021*.

Tianlu Wang, Xuezhi Wang, Yao Qin, Ben Packer, Kang Li, Jilin Chen, Alex Beutel, and Ed Chi. 2020b. CAT-gen: Improving robustness in NLP models via controlled adversarial text generation. In *Proceedings of EMNLP 2020*.

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the Robustness of Language Encoders against Grammatical Errors. In *Proceedings of ACL 2020*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level Textual Adversarial Attacking as Combinatorial Optimization. In *Proceedings of ACL 2020*.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *Processings of ICLR 2018*.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2020. Generating Fluent Adversarial Examples for Natural Languages. In *Proceedings of ACL 2019*.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. 2021. How Does Mixup Help With Robustness and Generalization? In *Proceedings of ICLR 2021*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of NeurIPS 2015*.

Xiaoqing Zheng, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of ACL 2020*.

Wei Zou, Shujian Huang, John Xie, Xin-Yu Dai, and Jiajun Chen. 2020. A Reinforced Generation of Adversarial Samples for Neural Machine Translation. In *Proceedings of ACL 2020*.

Appendix

Hyper-parameter Analysis

In this section, we perform further analysis to examine the effects of different hyper-parameters. There are two hyper-parameters involved in MixADA: the amount of adversarial data added for training, and the α parameter in the beta distribution of mixup coefficient. We also experiment with an alternative ADA strategy - iterative ADA.

Amount of Adversarial Training Data

We vary the ratio of the training dataset that we generate adversarial training samples on and add to the MixADA fine-tuning. We experiment with SMixADA with the hyper-parameter of mixup being fixed. On SST-2, we vary the ratio in $\{25\%, 50\%, 75\%, 100\%\}$. On IMDB, since the average sequence length is significantly longer and the adversarial example generation process becomes much slower, we experiment with a set of smaller ratios: $\{5\%, 10\%, 15\%, 20\%\}$. The results are plotted in in Figure 2. Interestingly, we find that higher ratio of adversarial training samples does not necessarily bring in additional robustness gains.

Interpolation Coefficient in Mixup

We also analyse the hyper-parameter of mixup: the α parameter in the beta distribution, from which the interpolation coefficient is sampled. We fix the ratio of adversarial training data and vary α in the range of $\{0.2, 0.4, 2.0, 4.0, 8.0\}$. The results are plotted in Figure 3. We find that there is no consistent pattern across different datasets on what is the optimal α. Hence, for our main experiments in the paper, we perform a greedy hyper-parameter search: we first tune the ratio of adversarial training samples, then fix the ratio and tune the α parameter for mixup. A more exhaustive hyper-parameter search might bring additional performance gains but would also incur extra computation costs.

Iterative ADA

For our MixADA experiments in the paper, we generate all adversarial training samples at one shot and mix them with the original examples before fine-tuning. An alternative is to generate a new batch of adversarial training samples dynamically with the current model at each epoch. We compare this iterative approach with our MixADA and use the same ratio of adversarial training samples and mixup parameter α. We evaluate RoBERTa on the SST-2 dataset. The results are in Table 4.

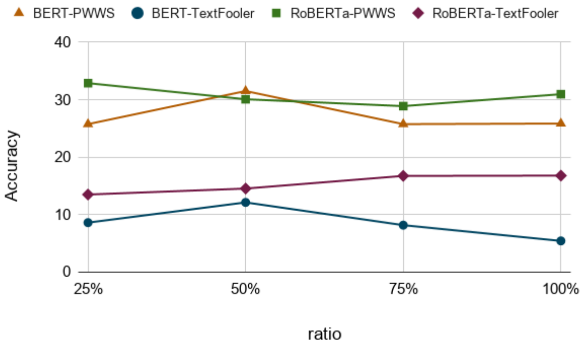

Figure 2: Performance under attacks on the SST-2 dataset with varying ratio of adversarial training samples.

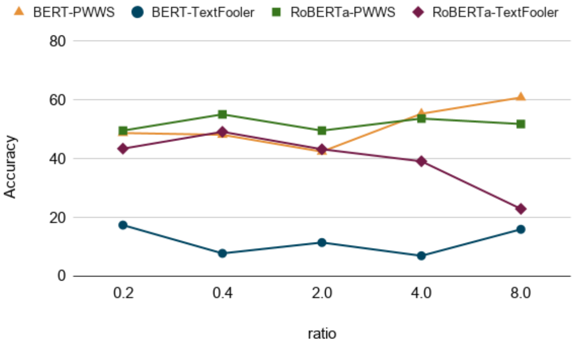

Figure 3: Performance under attacks on the IMDB dataset with varying α parameter for mixup.

	PWWS	TextFooler
TMixADA	36.74	13.78
+iterative	28.45	6.26
SMixADA	41.85	16.80
+iterative	28.78	7.69

Table 4: Performance of MixADA under attacks in the one-shot approach and the iterative approach.

We find that the iterative approach is far worse than our one-shot approach. We hypothesize that in the one-shot approach, we generate the adversarial examples on a fully-fine-tuned model while the iterative approach generates adversarial examples on the not-well-fine-tuned model in the first few epochs, and hence the adversarial examples generated in the iterative approach are not as challenging and useful as those in our one-shot approach.

NAST: A Non-Autoregressive Generator with Word Alignment for Unsupervised Text Style Transfer

Fei Huang, Zikai Chen, Chen Henry Wu, Qihan Guo, Xiaoyan Zhu, Minlie Huang*
The CoAI group, DCST; Institute for Artificial Intelligence;
State Key Lab of Intelligent Technology and Systems;
Beijing National Research Center for Information Science and Technology;
Tsinghua University, Beijing 100084, China.
`f-huang18@mails.tsinghua.edu.cn` `natnstart@gmail.com` `henrychenwu98@gmail.com`
`gqh18@mails.tsinghua.edu.cn` `zxy-dcs@tsinghua.edu.cn` `aihuang@tsinghua.edu.cn`

Abstract

Autoregressive models have been widely used in unsupervised text style transfer. Despite their success, these models still suffer from the content preservation problem that they usually ignore part of the source sentence and generate some irrelevant words with strong styles. In this paper, we propose a Non-Autoregressive generator for unsupervised text Style Transfer (NAST), which alleviates the problem from two aspects. First, we observe that most words in the transferred sentence can be aligned with related words in the source sentence, so we explicitly model word alignments to suppress irrelevant words. Second, existing models trained with the cycle loss align sentences in two stylistic text spaces, which lacks fine-grained control at the word level. The proposed non-autoregressive generator focuses on the connections between aligned words, which learns the word-level transfer between styles. For experiments, we integrate the proposed generator into two base models and evaluate them on two style transfer tasks. The results show that NAST can significantly improve the overall performance and provide explainable word alignments. Moreover, the non-autoregressive generator achieves over 10x speedups at inference. Our codes are available at `https://github.com/thu-coai/NAST`.

1 Introduction

Text style transfer aims at changing the text style while preserving the style-irrelevant contents, which has a wide range of applications, e.g., sentiment transfer (Shen et al., 2017), text formalization (Rao and Tetreault, 2018), and author imitation (Jhamtani et al., 2017). Due to the lack of parallel training data, most works focus on *unsupervised* text style transfer using non-parallel stylistic data.

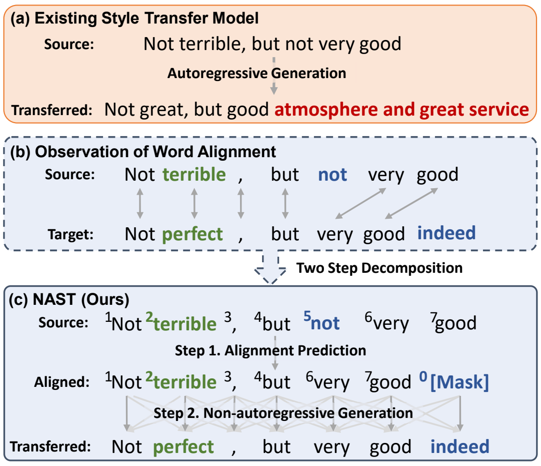

Figure 1: Sentiment transfer examples (negative to positive). (a) Existing models without word alignments may generate **words irrelevant to the source sentence**. (b) An example of word alignments between the source and target sentences. Arrows connect aligned words (identical or **relevant**), and **blue words are not aligned**. (c) NAST's generation process. Step 1: generate the index of aligned words. [Mask] is a placeholder for unaligned words. Step 2: generate the transferred sentence non-autoregressively.

The cycle consistency loss (Zhu et al., 2017), a.k.a. the back-translation loss (Lample et al., 2018, 2019), has been widely adopted by unsupervised text style transfer models (Dai et al., 2019; He et al., 2020; Yi et al., 2020). Specifically, the cycle loss minimizes the reconstruction error for the sentence transferred from style $\mathcal{X}$ to style $\mathcal{Y}$ and then back to $\mathcal{X}$, which aligns the sentences in two stylistic text spaces to achieve the transfer and preserve style-irrelevant contents. The cycle-loss-based models are trained in an end-to-end fashion, and thus can be easily applied to different datasets.

Although cycle-loss-based models yield promising results, one of their major failure cases is to replace some part of the source sentence with irrelevant words that have strong styles, as shown in Fig 1(a). This problem degrades content preservation and can be alleviated from two perspectives. **First**, we observe that most words in the human-written

---

[*] Corresponding author: Minlie Huang.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1577–1590
August 1–6, 2021.

transferred sentence can be aligned with those in the source sentence. As shown in Fig 1(b), we can align "*Not*" with "*Not*", "*terrible*" with "*perfect*", and leave only a few words unaligned. It shows that humans regard the alignments between words as a key aspect of content preservation, but they are not explicitly modeled by cycle-loss-based models yet. **Second**, existing models use the cycle loss to align sentences in two stylistic text spaces, which lacks control at the word level. For example, in sentiment transfer, "*tasty*" should be mapped to "*awful*" (because they both depict food tastes) but not "*expensive*". We utilize a non-autoregressive generator to model the word-level transfer, where the transferred words are predicted based on contextual representations of the aligned source words.

In this paper, we propose a Non-Autoregressive generator for unsupervised Style Transfer (NAST), which explicitly models word alignment for better content preservation. Specifically, our generation process is decomposed into two steps: first predicting *word alignments* conditioned on the source sentence, and then generating the transferred sentence with a *non-autoregressive* (NAR) decoder. Modeling word alignments directly suppresses the generation of irrelevant words, and the NAR decoder exploits the word-level transfer. NAST can be used to replace the autoregressive generators of existing cycle-loss-based models. In the experiments, we integrate NAST into two base models: StyTrans (Dai et al., 2019) and LatentSeq (He et al., 2020). Results on two benchmark datasets show that NAST steadily improves the overall performance. Compared with autoregressive models, NAST greatly accelerates training and inference and provides better optimization of the cycle loss. Moreover, we observe that NAST learns explainable word alignments. Our contributions are:

- We propose NAST, a Non-Autoregressive generator for unsupervised text Style Transfer. By explicitly modeling word alignments, NAST suppresses irrelevant words and improves content preservation for the cycle-loss-based models. To the best of our knowledge, we are the first to introduce a non-autoregressive generator to an unsupervised generation task.
- Experiments show that incorporating NAST in cycle-loss-based models significantly improves the overall performance and the speed of training and inference. In further analysis, we find that NAST provides better optimization of the cycle loss and learns explainable word alignments.

2 Related Work

Unsupervised Text Style Transfer

We categorize style transfer models into three types. The first type (Shen et al., 2017; Zhao et al., 2018; Yang et al., 2018; John et al., 2019) disentangles the style and content representations, and then combines the content representations with the target style to generate the transferred sentence. However, the disentangled representations are limited in capacity and thus hardly scalable for long sentences (Dai et al., 2019). The second type is the editing-based method (Li et al., 2018; Wu et al., 2019a,b), which edits the source sentence with several discrete operations. The operations are usually trained separately and then constitute a pipeline. These methods are highly explainable, but they usually need to locate and replace the stylist words, which hardly applies to complex tasks that require changes in sentence structures. Although our two-step generation seems similar to a pipeline, NAST is trained in an end-to-end fashion with the cycle loss. All transferred words in NAST are generated, not copied, which is essentially different from these methods. The third type is based on the cycle loss. Zhang et al. (2018); Lample et al. (2019) introduce the back translation method into style transfer, where the model is directly trained with the cycle loss after a proper initialization. The following works (Dai et al., 2019; Luo et al., 2019; He et al., 2020; Yi et al., 2020) further adopt a style loss to improve the style control.

A recent study (Zhou et al., 2020) explores the word-level information for style transfer, which is related to our motivation. However, they focus on word-level style relevance in designing novel objectives, while we focus on modeling word alignments and the non-autoregressive architecture.

Non-Autoregressive Generation

Non-AutoRegressive (NAR) generation is first introduced in machine translation for parallel decoding with low latency (Gu et al., 2018). The NAR generator assumes that each token is generated independently of each other conditioned on the input sentence, which sacrifices the generation quality in exchange for the inference speed.

Most works on NAR generation focus on improving the generation quality while preserving the speed acceleration in machine translation. Gu et al. (2018) find the decoder input is critical to the gener-

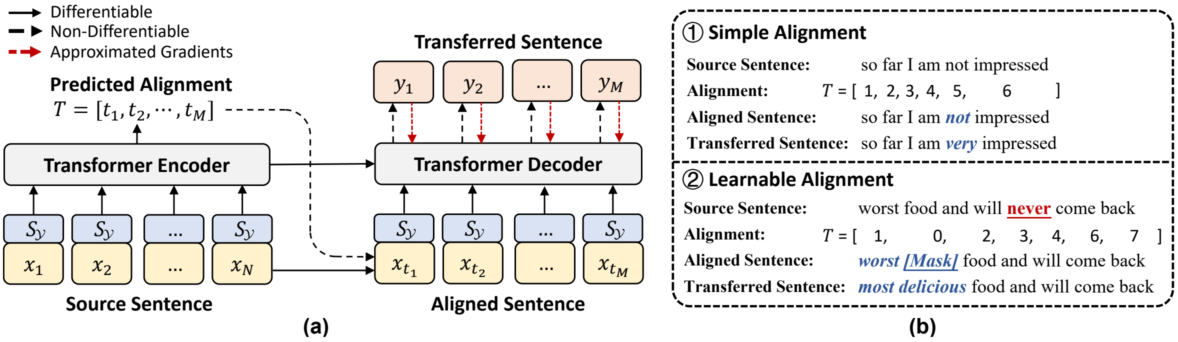

Figure 2: (a) Architecture of NAST transferring X to Y. $S_\mathcal{Y}$ is the target style. The NAR decoder generates each word y_i independently. (b) Examples of two alignment prediction strategies. **Simple Alignment**: each y_i is aligned with x_i. **Learnable Alignment**: a network predicts the alignment, where $t_k = 0$ indicates a [Mask] placeholder.

ation quality. Several works (Bao et al., 2019; Ran et al., 2019) improve the decoder input by aligning source words with target words, which utilize a two-step generation process and inspire the design of NAST. To our knowledge, only a few works of NAR generation explore applications other than machine translation (Han et al., 2020; Peng et al., 2020). We are the first to apply NAR generators to an unsupervised text generation task, which surprisingly outperforms autoregressive models in transfer quality besides the acceleration.

3 Methods

In this paper, we formulate the unsupervised text style transfer as follows: for two non-parallel corpora with styles $\mathcal{X}$ and $\mathcal{Y}$ respectively, the task aims at training a style transfer model G. The model learns the transfer of two directions, $\mathcal{X} \to \mathcal{Y}$ and $\mathcal{Y} \to \mathcal{X}$, which can be denoted as $P_{G_\mathcal{Y}}(Y|X)$ and $P_{G_\mathcal{X}}(X|Y)$, respectively.

3.1 NAST

NAST is a non-autoregressive generator based on the observation of the word alignment: in style transfer tasks, most generated words can be aligned with the source words, where each pair of the aligned words is either identical or highly relevant. For simplicity, we only describe $G_\mathcal{Y}$, where $G_\mathcal{X}$ shares the architecture and parameters except style embeddings. Given the source sentence $X = [x_1, x_2, \cdots, x_N]$, the generation process of NAST is decomposed into two steps: predicting the alignment $T = [t_1, t_2, \cdots, t_M]$, and then generating the transferred sentence $Y = [y_1, y_2, \cdots, y_M]$. When $1 \le t_i \le N$, the generated word y_i is aligned with the source word x_{t_i}. Otherwise, y_i is not aligned with any source word, where we set t_i to 0 and fill x_{t_i} with a [Mask] placeholder. Formally, we regard T as a latent variable, and the generation probability is formulated as

$$P_{G_\mathcal{Y}}(Y|X) = \sum_T P_{G_\mathcal{Y}}(Y|X,T) P_{G_\mathcal{Y}}(T|X), \qquad (1)$$

where $P_{G_\mathcal{Y}}(T|X)$ and $P_{G_\mathcal{Y}}(Y|X,T)$ are modeled by an alignment predictor and a non-autoregressive decoder, respectively, as shown in Fig 2.

3.1.1 Alignment Predictor

The alignment predictor predicts the target length M and the alignment T conditioned on the source sentence X. We utilize a Transformer (Vaswani et al., 2017) to encode the source sentence and then explore two alternative strategies to predict T.

Simple Alignment. Simple Alignment assumes that the source and target sentences have the same length, and each generated word y_i is exactly aligned with the source word x_i. Formally,

$$P_{G_\mathcal{Y}}(T|X) = \mathbb{I}[M = N] \prod_{i=1}^{M} \mathbb{I}[t_i = i],$$

where $\mathbb{I}[\cdot]$ is the indicator function. A similar strategy has been adopted by editing-based methods (Wu et al., 2019b; Helbig et al., 2020), where they simply replace several words in the source sentence. Although this strategy cannot alter the sentence length, it empirically works well on simple tasks, such as sentiment transfer.

Learnable Alignment. Inspired by Ran et al. (2019); Bao et al. (2019), we utilize a pointer network (Vinyals et al., 2015) on top of the encoder, which predicts the alignment T:

$$P_{G_\mathcal{Y}}(T|X) = \prod_{i=1}^{M} P_{G_\mathcal{Y}}(t_i|X, t_{<i}).$$

The pointer network is essentially an autoregressive generator, but it only generates the alignment t_i pointing to a source word.

3.1.2 Non-autoregressive Decoder

The non-autoregressive decoder (Gu et al., 2018) is a Transformer that generates each word independently. Formally, we have

$$P_{G_\mathcal{Y}}(Y|X,T) = \prod_{i=1}^{M} P_{G_\mathcal{Y}}(y_i|X,T). \tag{2}$$

The Transformer decoder takes the aligned sentence $[x_{t_1}, x_{t_2}, \cdots, x_{t_M}]$ and the target style embedding $S_\mathcal{Y}$ as inputs. It also contains attention connections from the Transformer encoder.

3.1.3 Training

NAST is a generator that can be integrated into existing cycle-loss-based models. These models mainly utilize three losses, and the overall objective $\mathcal{L}$ is defined as $\alpha L_{self} + \beta L_{sty} + \gamma L_{cyc}$, where α, β, γ are hyper-parameters. The **self-reconstruction loss** L_{self} aims at recovering sentences of both styles from their corrupted versions:

$$\begin{aligned}\mathcal{L}_{self} = -\mathbb{E}_{X\sim P_\mathcal{X}}&\left[\log P_{G_\mathcal{X}}(X|\widetilde{X})\right] - \\ &\mathbb{E}_{Y\sim P_\mathcal{Y}}\left[\log P_{G_\mathcal{Y}}(Y|\widetilde{Y})\right],\end{aligned} \tag{3}$$

where $\widetilde{X}$ and $\widetilde{Y}$ are constructed by word dropout, insertion, and masking (Lample et al., 2019), and $P_\mathcal{X}$ and $P_\mathcal{Y}$ are the data distributions of two styles. The **style loss** L_{sty} is used to guide the style of generated sentences, which has various designs by existing works, e.g., adopting a style discriminator (Dai et al., 2019) or a language model (He et al., 2020). In our implementation, the style loss is determined by the base model. We simply present a general formulation:

$$\begin{aligned}\mathcal{L}_{sty} = -\mathbb{E}_{X\sim P_\mathcal{X}}&\left[F(G_\mathcal{Y}(X),\mathcal{Y})\right] - \\ &\mathbb{E}_{Y\sim P_\mathcal{Y}}\left[F(G_\mathcal{X}(Y),\mathcal{X})\right],\end{aligned} \tag{4}$$

where $F(X,\mathcal{X})$ indicates a score that shows to which extent the sentence X has the style $\mathcal{X}$, and $G_\mathcal{Y}(X)$ is the generated sentence sampled from $P_{G_\mathcal{Y}}(Y|X)$ in two steps: $T_\mathcal{Y}(X) \sim P_{G_\mathcal{Y}}(T|X)$, $G_\mathcal{Y}(X) \sim P_{G_\mathcal{Y}}(Y|X, T_\mathcal{Y}(X))$. At last, the **cycle loss** L_{cyc} is formulated as

$$\begin{aligned}\mathcal{L}_{cyc} = -\mathbb{E}_{X\sim P_\mathcal{X}}&\left[\log P_{G_\mathcal{X}}(X|G_\mathcal{Y}(X))\right] - \\ &\mathbb{E}_{Y\sim P_\mathcal{Y}}\left[\log P_{G_\mathcal{Y}}(Y|G_\mathcal{X}(Y))\right].\end{aligned} \tag{5}$$

However, there still exist two obstacles in optimization. **Firstly**, because of the non-differentiable problem, we cannot back-propagate the gradients through the discrete text $G_\mathcal{Y}(X)$ in Eq.(4)(5). As a common workaround, we adopt the Gumbel-Softmax trick (Jang et al., 2017) to approximate the gradients. Therefore, the gradients from $G_\mathcal{Y}(X)$ can be back-propagated through the decoder output (Fig 2(a)). However, the alignment $T_\mathcal{Y}(X)$ is remained discrete and non-differentiable, where we simply stop the gradients[1].

Secondly, the losses in Eq.(3)(5) are intractable for NAST because the generation probability, e.g. $P_{G_\mathcal{Y}}(Y|X)$, is summed over all alignments as defined in Eq.(1). We provide solutions for the two alignment strategies separately.

For Simple Alignment. There is only one valid alignment between X and Y, so the generation probability is tractable as

$$\begin{aligned}\log P_{G_\mathcal{Y}}(Y|X) &= \log P_{G_\mathcal{Y}}(Y|X,T^*), \\ \text{where } T^* &= \arg\max_T P_{G_\mathcal{Y}}(T|X) = [1,2,\ldots,N].\end{aligned}$$

For Learnable Alignment. Inspired by Bao et al. (2019), we introduce a heuristic rule to obtain a pseudo alignment T^*:

$$\begin{aligned}&T^* = \arg\max_T \sum_{i=1}^{M} cos(e(y_i), e(x_{t_i})) \\ &s.t. \quad t_i = 0 \text{ or } t_i > t_j \quad \text{for } \forall\, 1 \le j < i \le M,\end{aligned}$$

where $e(\cdot)$ indicates the word embeddings. We can obtain the pseudo alignment by dynamic programming, and the details are presented in Appendix A. In the pseudo alignment, most words in Y are aligned with identical or highly relevant words in X, which can be used as a good label to supervise our model. Next, we derive a tractable lower bound for the generation probability:

$$\log P_{G_\mathcal{Y}}(Y|X) \ge \log P_{G_\mathcal{Y}}(Y|X,T^*) + \log P_{G_\mathcal{Y}}(T^*|X). \tag{6}$$

On the right side, the first term trains the NAR decoder, and the second term trains the alignment predictor. By substituting Eq.(6) into Eq.(3)(5), we turn to optimize the upper bounds instead of the original intractable losses. The detailed training algorithm is shown in Appendix A.

3.2 Discussions

Residual Connections and Multi-head Attention. The aligned words in NAST are directly connected with the residual connections, and these connections form several chains in the cycle loss optimization, as shown in Fig 3. Most of these chains represent the word-level transfers and reconstructions, e.g., "*terrible*" is transferred to "*perfect*" and

---

[1] As a result, the alignment predictor (for Learnable Alignment) is not optimized following the gradients from $G_\mathcal{Y}(X)$, but with a pseudo label introduced later.

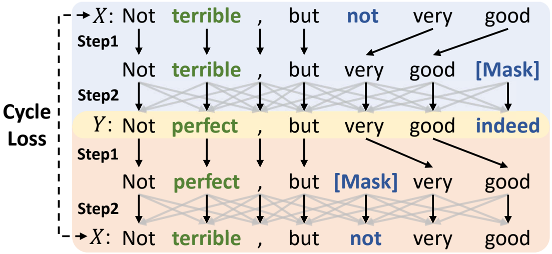

Figure 3: Connections of NAST in the cycle loss with the encoder omitted. The word alignments (step 1) and the residual connections (step 2) are in black.

then reconstructs "*terrible*". The reconstruction error is a part of the cycle loss, which is optimized to enhance the alignment in the word space. Besides the residual connections, the multi-head attention mechanism is also important for our model. The attention stops NAST from becoming a degenerate word-to-word dictionary and makes it possible to predict the unaligned words from the context.

Exposure Bias in Autoregressive (AR) Models. Exposure bias (Bengio et al., 2015) is a notorious problem in the AR generation. To obtain $P_{G_\mathcal{X}}(X|G_\mathcal{Y}(X))$ in the cycle loss, AR generators predict each word of X based on the ground-truth prefix, which is an easy task even without information from $G_\mathcal{Y}(X)$. As a result, in inference, the model may fail in preserving the sentence meaning as it is trained to focus on its generated prefix. In contrast, NAST focuses on the source sentence since the ground-truth prefix is not given, which suppresses the problem of generating irrelevant words and improves content preservation. Moreover, the training and test are consistent in NAST[2], which alleviates the exposure bias problem.

4 Experiments

4.1 Experiment Settings

We conduct experiments on two style transfer tasks.

Sentiment Transfer. We use the YELP dataset (Li et al., 2018), which consists of two non-parallel corpora with positive and negative sentiments. For each sentence in the test set, multiple human references are provided by Luo et al. (2019).

Text Formalization. We use the family and relationship domain of the GYAFC dataset (Rao and Tetreault, 2018), which consists of paired corpora for formal and informal sentences. We do not use the paired data to supervise training.

We utilize several SOTA models as baselines, which include CrossAlign (Shen et al., 2017), DelRetrie (Li et al., 2018), Disent (John et al., 2019), StyIns (Yi et al., 2020), StyTrans (Dai et al., 2019), and LatentSeq (He et al., 2020). Our models are modified based on StyTrans and LatentSeq, where we replace their generators with NAST. For StyTrans, NAST adopts a Transformer of the same architecture as the original implementation. However, LatentSeq utilizes an LSTM generator. For a fair comparison, we first incorporate LatentSeq with a vanilla Transformer generator and then replace the generator with NAST of the same architecture. In inference, we use the greedy decoding strategy, i.e., we choose the top-1 candidate at each step in alignment prediction and sentence generation. More details are presented in Appendix B.

4.2 Automatic Evaluation

Following Luo et al. (2019); Dai et al. (2019), we utilize a pretrained classifier to evaluate the style accuracy (Acc), and adopt the BLEU-4 score comparing generated sentences with the source sentences (SelfB) or with the references (RefB) to evaluate content preservation. The classifier based on RoBERTa-base (Liu et al., 2019) achieves an accuracy of 97.6% and 90.1% on YELP and GYAFC, respectively. For each transfer direction, we calculate the geometric and harmonic mean of Acc and RefB and then report the average on two directions as G2 and H2, respectively. We further report the perplexity (PPL) of transferred sentences, which is evaluated by GPT2-base (Radford et al., 2019) fine-tuned on the training set.

The results are shown in Table 1. Compared with StyTrans and LatentSeq, NAST exhibits stable performance gains of G2 and H2 on both datasets. On the Yelp dataset, NAST remarkably improves content preservation (at least 6 points with RefB) but suffers a slight decline in Acc. We find that NAST can suppress irrelevant words with strong styles, which possibly leads to the decline in Acc. On the GYAFC dataset, NAST outperforms the base models mainly in Acc instead of RefB, which is affected by model selection strategies with the Acc-RefB trade-off. In Table 1, we choose the best model based on G2. A more comprehensive comparisons with trade-off curves will be discussed in the next section.

In terms of the alignment strategies, Learnable Alignment outperforms Simple Alignment on

---

[2] The claim only applies to NAST with Simple Alignment, because the pseudo alignment used in Learnable Alignment breaks the consistency.

	Yelp						GYAFC					
Model	**PPL**	**Acc**	**SelfB**	**RefB**	**G2**	**H2**	**PPL**	**Acc**	**SelfB**	**RefB**	**G2**	**H2**
CrossAlign (Shen et al., 2017)	105	74.0	20.3	17.9	31.8	28.8	47	63.8	2.3	3.2	14.1	6.1
DelRetrie (Li et al., 2018)	94	88.7	36.8	31.1	52.5	46.0	101	58.2	32.3	20.8	34.2	29.8
Disent (John et al., 2019)	27	92.2	8.3	13.8	35.6	24.0	27	68.4	4.8	8.0	23.4	14.4
DualRL (Luo et al., 2019)	73	88.6	59.0	55.2	68.6	67.0	91	58.9	50.1	40.3	43.9	39.2
StyIns (Yi et al., 2020)	98	91.5	53.2	49.0	66.9	63.7	72	65.6	62.6	45.5	52.6	50.0
StyTrans (Dai et al., 2019)	136	**90.4**	53.3	48.6	66.2	63.1	124	67.1	59.7	41.9	50.4	46.8
+ NAST (Simple)	117	88.9	**63.3****	**55.9****	**70.4****	**68.5****	130	67.6	**63.7***	41.6	50.8	47.4
+ NAST (Learnable)	**112***	87.4	62.0**	54.6**	69.0**	67.1**	**119**	**72.9***	61.6	**42.8**	**53.6****	**49.9****
LatentSeq (He et al., 2020)	55	**84.5**	49.4	47.3	62.6	60.5	47	55.3	**57.8**	38.5	44.1	42.5
LatentSeq w/ Transformer	**42**	**84.6**	48.8	47.1	63.0	60.4	**38**	58.1	54.3	35.3	45.1	43.5
+ NAST (Simple)	73	81.2	65.2**	57.6**	**68.1****	**66.9****	56	60.4*	57.0	38.2	47.4*	45.6*
+ NAST (Learnable)	70	79.6	**65.5****	**58.0****	67.7**	66.7**	53	**64.1****	57.0	**39.2**	**49.0****	**46.6***

Table 1: Automatic evaluation results. **Simple** and **Learnable** indicate two alignment strategies. All values are averaged on two transfer directions. **Bold** denotes the best results for each base model, and underline denotes the best results in all models. * and ** indicate significant improvements over StyTrans or LatentSeq ($p < 0.05$ and $p < 0.01$ in t-test).

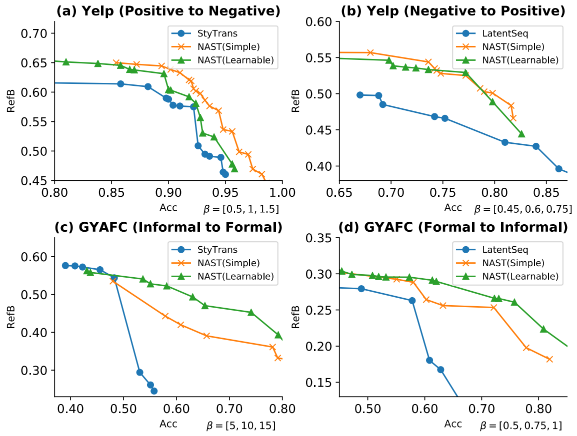

Figure 4: Trade-off curves between style control (Acc) and content preservation (RefB). (a)(c) use StyTrans as the base model, (b)(d) use LatentSeq as the base model. Each curve contains points from three runs with different style loss coefficients β, whose values for NAST are presented under sub-figures.

GYAFC, but there is no significant difference on Yelp. We suppose that the sentiment transfer task is more straightforward than the text formalization, where the model can achieve a good transfer performance on Yelp without changing sentence structures.

Compared with all baselines, our best models set new SOTA results on two datasets in the overall performance of the transfer accuracy and content preservation (i.e., G2 and H2).

Trade-Off Curves. To investigate the trade-off between style control (shown by Acc) and content preservation (shown by RefB), we follow Fu et al. (2018) and evaluate the models with different hyper-parameters. To be specific, we select three different style loss coefficients β around the best value. Please see Appendix B.2 for the search range and other details. Since the trade-off varies through the training, we evaluate the models and collect data points at every epoch. It is different from Fu et al. (2018), who only plot the metrics of the best model in each run. The curves are shown in Fig 4, where we only keep the outermost points of each model and remove the points dominated by at least one other point in both Acc and RefB.

	#Param	**Train (ms)**	**Inference (ms)**
StyTrans	31.1M	857 (1.0x)	249 (1.0x)
+ NAST (Simple)	31.1M	**201 (4.3x)**	**8 (31.1x)**
+ NAST (Learnable)	32.4M	339 (2.5x)	71 (3.5x)
LatentSeq w/ Trans.	21.2M	1282 (1.0x)	266 (1.0x)
+ NAST (Simple)	21.2M	**714 (1.8x)**	**23 (11.6x)**
+ NAST (Learnable)	22.4M	761 (1.7x)	125 (2.1x)

Table 2: Parameter size and the training and inference latency on GYAFC. The speedup of training LatentSeq is less significant, because the bottleneck is a language model used in the style loss, costing about 487ms.

The curves of NAST are generally above those of the base models, indicating that NAST achieves better content preservation when the style accuracy is kept at a similar level. In Fig 4 (c)(d), we find that the base model's RefB drops rapidly after Acc exceeds a certain value, which indicates that the cycle loss fails to preserve the sentence-level alignment, thereby leading to model collapse. By contrast, NAST largely alleviates the issue of model collapse. Moreover, we find that Learnable Alignment outperforms Simple Alignment on GYAFC, but performs equally or slightly worse on Yelp, due to the task differences discussed above.

Training & Inference Speed. Thanks to the parallel decoding of the NAR generator, NAST accelerates the model training and inference as shown in Table 2. For a fair comparison, NAST and the corresponding base model utilize the same Transformer architecture. The computation devices are detailed in Appendix B.3.

4.3 Human Evaluation

We follow Li et al. (2018) and conduct human evaluation experiments on the Yelp dataset. In addition to NAST and the base models, we choose three

Model	Fluency	Style	Content
DelRetrie	3.87	3.90	3.05
DualRL	4.38	4.25	4.24
StyIns	4.25	4.00	4.11
StyTrans	4.24	**3.91**	4.16
+ NAST(Simple)	4.34	3.87	**4.41****
+ NAST(Learnable)	**4.39***	3.87	4.38**
LatentSeq	4.53	3.92	3.59
+ NAST(Simple)	4.41	**3.93**	4.43**
+ NAST(Learnable)	**4.57**	3.82	**4.48****

Table 3: Human evaluation results. **Bold** denotes the best results for each base model and underline denotes the best results among all models. * and ** indicate significant improvements over the base model ($p < 0.05$ and $p < 0.01$ in t-test). The Krippen-dorff's alpha of human rating is 0.72, indicating acceptable inter-annotator agreement.

	Yelp			GYAFC		
Model	**Acc**	**RefB**	**G2**	**Acc**	**RefB**	**G2**
LatentSeq(Trans.)	**84.6**	47.1	63.0	58.1	35.3	45.1
NAST(Simple)	81.3	57.4	**68.1**	58.3	39.3	**47.5**
w/o Aligned Sent.	73.2	44.1	56.8	54.1	34.4	42.8
w/o Multi-head Attn.	57.0	**62.8**	59.6	22.0	**41.1**	29.7
w/ Soft-Embedding	44.3	44.6	43.6	63.4	26.0	40.3
w/ Stop-Gradient	80.5	50.1	63.5	**64.2**	26.3	40.6

Table 4: Ablation study of NAR decoder and gradient approximation methods. The base model is LatentSeq.

baselines with the highest G2. For each model, we sample 100 sentences (50 in each transfer direction), and 900 sentences are evaluated in total. For each sentence, three annotators are asked to rate from 1 (worst) to 5 (best) for fluency, style control, and content preservation.

The human evaluation results are shown in Table 3. Similar to the automatic evaluation results, NAST improves content preservation significantly. Moreover, we find that Learnable Alignment outperforms Simple Alignment in terms of fluency. It can be partially attributed to the fact that Learnable Alignment, which is able to remove or add words, is more flexible in generation.

4.4 Ablation Study

NAR decoder. Although NAST with Simple Alignment has a simple, straightforward design, it works surprisingly well compared with an AR generator. We conduct an ablation study to investigate the impact of different components in the NAR decoder. First, we remove the *aligned sentence* from the decoder input. Specifically, the decoder input is the positional encodings without the word embeddings. Second, we remove the *multi-head attention* in the decoder, and thus each output word is solely conditioned on its aligned word.

The results are shown in Table 4. After we remove the aligned sentence, the performance drops but still remains comparable. It shows that the multi-head attention over the source sentence learns reasonable transfer, while the performance can be largely improved by providing the decoder with the aligned sentence as input. After we remove the multi-head attention, the overall performance drops remarkably, especially on GYAFC. It shows that NAST utilizes multi-head attention to gather sentence-level information, and it is essentially *not* a word-to-word dictionary. Moreover, the contribution of the multi-head attention is larger on GYAFC than on Yelp. It further justifies that text formalization is less straightforward than sentiment transfer since it requires more sentence-level modifications.

Pseudo Alignment in Self-Reconstruction Loss
S: that [Mask] *talk* , if are not happy *like* but you .
P: that **[Mask]** , if [Mask] are not happy but you [Mask] .
T: that **is** , if others are not happy but you are .
Pseudo Alignment in Cycle Loss
S: i leave your email on *exercise* , and see what happens .
P: **i** leave your email [Mask] on [Mask] **,** and see what happens **.**
T: **just** leave your email loged on accidentally **...** and see what happens **!**

Table 5: Pseudo alignments on GYAFC. S = source, P = pseudo alignment, T = target. *Unaligned source words*, unaligned target words, and **non-identical aligned words** are marked in different colors.

Gradient Approximation Methods. The choice of gradient approximation methods is important for tackling the non-differentiable problem. Besides the Gumbel-Softmax trick used in our full model, we try two alternative methods. 1) The Soft-Embedding approximation (Dai et al., 2019) multiplies the softmax distribution by the word embedding matrix to get "soft" word embeddings. 2) The Stop-Gradient strategy (He et al., 2020) stops the gradient at the decoder output in the cycle loss. However, the style loss requires the output to be differentiable, so we still apply the Gumbel-Softmax trick for the style loss. Results in Table 4 show that the Gumbel-softmax trick outperforms the other methods, so we utilize the Gumbel-Softmax trick for NAST in other experiments.

Learnable Alignment. According to Eq.(3)(5)(6), the alignment predictor in Learnable Alignment is supervised by pseudo alignments when optimizing the upper bounds of the self-reconstruction loss and the cycle loss. For the former, the alignment predictor learns to align the corrupted $\widetilde{X}$ with X. For the latter, the alignment predictor learns to align the transferred sentence $G_{\mathcal{Y}}(X)$ with the original X. We show two cases in Table 5, where the pseudo alignments are of acceptable quality.

To investigate the effects of the pseudo align-

Model	**Acc**	**RefB**	**G2**	$\lvert\Delta\rvert$	$std(\Delta)$
NAST(Simple)	66.5	41.6	50.4	0.00	0.00
NAST(Learnable)	**73.0**	**43.5**	**54.2**	0.80	1.26
w/o Pseudo(Recon)	66.1	41.5	50.3	0.02	0.28
w/o Pseudo(Cyc)	68.5	42.5	52.7	0.96	0.47

Table 6: Ablation study of NAST with Learnable Alignment on GYAFC. Δ is the length difference before and after the transfer. $\lvert\Delta\rvert$ and $std(\Delta)$ indicate the average absolute value and the standard deviation, respectively. All models use StyTrans as the base model.

	Yelp (Positive to Negative)
Source	love this place and will keep coming back .
LatentSeq	*do n't waste your time* and wo n't be back .
StyTrans	avoid this place and *will keep coming back* .
NAST(Simp.)	**skip** this place and will **never** coming back .
NAST(Lear.)	**hate** this place and will not **be** coming back .
	Yelp (Negative to Positive)
Source:	i did n't even eat it .
LatentSeq:	i always *love their food and service* .
StyTrans:	i love the food eat it .
NAST(Simp.):	i **love it and** eat it .
NAST(Lear.):	i definitely **love** [DEL] **to** eat it .
	GYAFC (Formal to Informal)
Source	the world would be happier if men knew what women want .
LatentSeq	*the guy would be mad if they want* what women want .
StyTrans	the world would be *what if thing what girls want girl ur girl want* .
NAST(Simp.)	**and** world **'ll** be happier if men knew what women want .
NAST(Lear.)	**just** world would be happier ... if **guys** knew what women want [Del]
	GYAFC (Informal to Formal)
Source:	i do n't know ! ... i just want the points ... lol
LatentSeq:	i do not know . i just want the points . *however , i am not a good one .*
StyTrans:	i do not know !
NAST(Simp.):	i do **not** know ! **.** i just want the points **. .**
NAST(Lear.):	i do **not** know ! [Del] i just want the points . [Del]

Table 7: Transfer cases. *Red words* indicate irrelevant phrases or failed transfer in style. Non-trivial alignments and **non-identical aligned words** are marked in colors. [Del] indicates the source word is unaligned.

ments supervision, we remove $\log P_{G_y}(T^*|X)$ in Eq.(6) for the two losses separately. Results are shown in Table 6. Without the pseudo alignments supervision in the self-reconstruction loss, the model almost degenerates into Simple Alignment, because keeping the length unchanged is the easiest way to minimize the cycle loss. Without the pseudo supervision in the cycle loss, Learnable Alignment is slightly weaker than the full model but still outperforms Simple Alignment.

4.5 Case Study of Word Alignment

We present several transfer cases in Table 7. We observe that a major failure mode of the base models is generating irrelevant words. We also observe that NAST achieves better content preservation, and most words in NAST's prediction can be aligned with the source words. Focused on the alignment strategies, we observe that the outputs of NAST with Simple Alignment sometimes contain grammar errors (e.g., "*will never coming back*"), which can be attributed to its limitation of not changing the sentence length. In contrast, we observe that Learnable Alignment can add and remove words at appropriate positions.

Src Word	**Transferred Words**			
	NAST(Simple) on Yelp (Negative to Positive)			
helpful	***weird*** (100%)			
fresh	***tasteless*** (61.5%)	***overcooked*** (38.5%)		
definitely	***not*** (92.9%)	***never*** (7.1%)		
nice	***rude*** (52.9%)	*no* (47.1%)		
best	***worst*** (96.8%)	*money* (3.2%)		
delicious	***bland*** (82.6%)	***ok*** (13.0%)	***frozen*** (4.4%)	
love	***hate*** (63.6%)	*ordered* (18.2%)	***skip*** (13.6%)	***avoid*** (4.6%)

Src Word	**Transferred Words**			
	NAST(Learnable) on GYAFC (Informal to Formal)			
'm	***am*** (100%)			
n't	***not*** (98.5%)	*n't* (1.5%)		
guy	***man*** (98.4%)	*guy* (1.6%)		
u	***you*** (89.2%)	*[Del]* (10.8%)		
lol	**.** (41.7%)	***[Del]*** (41.7%)	*although* (16.7%)	
...	***[Del]*** (31.3%)	**,** (27.4%)	**.** (26.3%)	and other 7 words
mean	***believe*** (50.0%)	*mean* (20.8%)	*am* (20.8%)	and other 2 words
[Mask]	**.** (55.2%)	*a* (12.0%)	***?*** (4.8%)	and other 28 words

Table 8: Cases of aligned word pairs generated by NAST. *[Del]* and *[Mask]* indicate an unaligned source word or an unaligned transferred word, respectively. ***Reasonable transfers are in blue.***

To understand the learned *word alignments* and the *word-level transfer*, we count the aligned word pairs based on the prediction of Learnable Alignment. Several cases are presented in Table 8. We observe the aligned word pairs are highly explainable. For example, NAST maps "*delicious*" to "*bland*" in sentiment transfer and maps "*guy*" to "*man*" in text formalization. These cases show that the model can learn fine-grained word-level transfer, where "*delicious*" and "*bland*" both depict food taste with different styles. Moreover, NAST with Learnable Alignment learns to add or remove words at reasonable positions, such as adding missing punctuation marks (".", "*?*") and removing redundant words ("...", "*lol*") in text formalization.

4.6 Analysis of Cycle Loss Optimization

The cycle loss plays a key role in unsupervised style transfer, which achieves style control and content preservation by aligning the sentences in two text spaces. However, the optimization is not straightforward due to the non-differentiable problem. In this section, we study how the cycle loss optimization is affected by the generator architecture and compare a NAR generator with an AR generator[3]. To remove the interference of other losses, we train the model solely with the cycle loss and report the BLEU-4 score of the cycle reconstruction.

The results are shown in Table 9. The NAR generator remarkably outperforms the AR generator with all gradient approximation methods. We provide two possible explanations for this observation. One reason is that word alignments can help the cycle loss align the text spaces. As discussed

---

[3] For a fair comparison, the target sentence length is provided to both models, where the AR generator does not need to predict the EOS token.

	Gumbel-Softmax	Stop-Gradient	Soft-Embedding
NAR	**94.4±0.4**	**67.6±4.2**	**33.6±13.8**
AR	84.9±1.1	23.5±1.1	29.9±15.1

Table 9: BLEU-4 of the cycle reconstruction on the Yelp dataset. The values are reported with mean and standard deviation of three runs with different seeds.

in Sec 3.2, the residual connections directly connect aligned words, which exploits the word-level transfer and reconstruction. Compared with the AR generator that aligns the text spaces at the sentence level, aligning word pairs can be much easier. Another possible reason is the error accumulation caused by the gradient approximation methods. In each step of the AR generation, the gradient approximation methods are applied to the generated word, and the word is then fed into the model as the next input. As a result, gradients will be approximated multiple times in the back-propagation, and the error brought by the approximation may be accumulated and possibly lead to unstable optimization.

Our analysis provides a perspective to understand how NAST works, and reveals that the generator architecture can deeply affect the optimization in the non-differentiable problem. However, we should be cautious when generalizing the results to other settings. We notice inconsistent performance report for the gradient approximation methods (Dai et al., 2019; Tu et al., 2020; He et al., 2020), where the phenomenon needs further study.

5 Conclusion

In this paper, we propose NAST, a Non-Autoregressive generator for unsupervised text Style Transfer. It explicitly models word alignments to suppress irrelevant words and exploits the word-level transfer between different styles. Experiments show that NAST improves the overall performance, provides explainable word alignments, and largely speed up training and inference.

However, we should also notice a potential limitation: NAST relies on the assumption that word alignments exist between the source and target sentences. In a more complicated task that lacks word alignments, NAST may lose its advantage of exploiting the word-level transfer. In future work, we will improve NAST to tackle noisy word alignments in more challenging datasets and build explainable and faster models for a broader range of unsupervised text generation tasks.

Acknowledgments

This work was partly supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005.

References

Yu Bao, Hao Zhou, Jiangtao Feng, Mingxuan Wang, Shujian Huang, Jiajun Chen, and Lei Li. 2019. Non-autoregressive transformer by position learning. *CoRR*, abs/1911.10677.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 663–670. AAAI Press.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Qinghong Han, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Non-autoregressive neural dialogue generation. *CoRR*, abs/2002.04250.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

David Helbig, Enrica Troiano, and Roman Klinger. 2020. Challenges in emotion style transfer: An exploration with a lexical substitution pipeline. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 41–50, Online. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19, Copenhagen, Denmark. Association for Computational Linguistics.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Guillaume Lample, Sandeep Subramanian, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Xu Sun, and Zhifang Sui. 2019. A dual reinforcement learning framework for unsupervised text style transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5116–5122. ijcai.org.

Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. 2020. Non-autoregressive neural text-to-speech. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7586–7598. PMLR.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *CoRR*, abs/1911.02215.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6830–6841.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2826, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Chen Wu, Xuancheng Ren, Fuli Luo, and Xu Sun. 2019a. A hierarchical reinforced sequence operation method for unsupervised text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4873–4883, Florence, Italy. Association for Computational Linguistics.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019b. Mask and infill: Applying masked language model for sentiment transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5271–5277. ijcai.org.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7298–7309.

Xiaoyuan Yi, Zhenghao Liu, Wenhao Li, and Maosong Sun. 2020. Text style transfer via learning style instance supported latent space. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3801–3807. ijcai.org.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *CoRR*, abs/1808.07894.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5897–5906. PMLR.

Chulun Zhou, Liangyu Chen, Jiachen Liu, Xinyan Xiao, Jinsong Su, Sheng Guo, and Hua Wu. 2020. Exploring contextual word-level style relevance for unsupervised style transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7135–7144, Online. Association for Computational Linguistics.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251. IEEE Computer Society.

Algorithm 1 DP Algorithm for Pseudo Alignment $DP(X, Y)$

```
Require: Source sentence $X = [x_1, x_2, \cdots, x_N]$,
         Target sentence $Y = [y_1, y_2, \cdots, y_M]$.
Initialize $f(0, j) = 0$ for $\forall\, j = 0, 1, \cdots, N$.
Initialize $T(0, j)$ as empty lists for $\forall\, j = 0, 1, \cdots, N$.
Calculate the similarity matrix:
    $sim_{i,j} = cos(e(y_i), e(x_j))$.
for $i = 1, 2, \cdots, M$ do
    for $j = 0, 1, 2, \cdots, N$ do
        Calculate three choices of $f(i, j)$:
            $c_1 := f(i-1, j)$
            $c_2 := f(i-1, j-1) + sim_{i,j}$ only valid if $j > 0$
            $c_3 := f(i, j-1)$ only valid if $j > 0$
        if $c_1$ is the maximum choice then
            ▷ $y_i$ is not aligned.
            $f(i, j) := c_1,\ T(i, j) := T(i-1, j) \oplus [0]$
            ▷ $\oplus$ means list concatenation.
        else if $c_2$ is the maximum choice then
            ▷ $y_i$ is aligned with $x_j$.
            $f(i, j) := c_2,\ T(i, j) := T(i-1, j-1) \oplus [j]$
        else if $c_3$ is the maximum choice then
            ▷ $y_i$ is aligned with $x_k$, where $k < j$.
            $f(i, j) := c_3,\ T(i, j) := T(i, j-1)$
        end if
    end for
end for
return $DP(X, Y) := T(M, N)$
```

A Optimization of NAST with Learnable Alignment

Since the generation probability $P_{G_\mathcal{Y}}(Y|X)$ is intractable for NAST with Learnable Alignment, we introduce a pseudo alignment T^*. For $X = [x_1, x_2, \cdots, x_N]$ and $Y = [y_1, y_2, \cdots, y_M]$, the pseudo alignment T^* is defined by a heuristic rule:

$$\begin{aligned} T^* &= \arg\max_T V(X, Y) \\ &= \sum_{i=1}^{M} cos(e(y_i), e(x_{t_i})) \\ s.t.\quad t_i = 0 &\text{ or } t_j < t_i \le N \quad \text{for } \forall\, 1 \le j < i \le M, \end{aligned}$$

where $e(\cdot)$ indicates the word embeddings. For the unaligned target words, we set x_0 to a [Mask] placeholder, and set the cosine similarity between the [Mask] placeholder and any other tokens to 0.

The pseudo alignments are obtained by dynamic programming. We introduce a 2-dim array $f(i, j)$ indicating the maximum value of the objective function $V(X, Y)$ if $Y = [y_1, y_2, \cdots, y_i]$ and $X = [x_1, x_2, \cdots, x_j]$. We further introduce a list $T(i, j)$ that records the best alignment for $f(i, j)$. The algorithm is presented in Algorithm 1. The time complexity is $O(NMd)$, where the bottleneck is calculating the similarity matrix, and d is the dimension of word embeddings.

Based on the pseudo alignments, we derive tractable upper bounds of the losses, which are

Algorithm 2 Training Algorithm for NAST with Learnable Alignment

```
Require: Non-parallel text distribution $P_{\mathcal{X}}$ and $P_{\mathcal{Y}}$.    Max number of batches: $max\_batch$.
for $iter = 1, 2, \cdots, max\_batch$ do
    Sample $X$ from $P_{\mathcal{X}}$, $Y$ from $P_{\mathcal{Y}}$.
    Construct $\widetilde{X}$ and $\widetilde{Y}$ from $X$ and $Y$.
    Use Algorithm 1 to obtain pseudo alignments for the self-reconstruction loss:
        $T^*_{X,self} = DP(\widetilde{X}, X), \quad T^*_{Y,self} = DP(\widetilde{Y}, Y).$
    Calculate the upper bound of the self-reconstruction loss.
        $\hat{\mathcal{L}}_{self} = -\log P_{G_{\mathcal{X}}}(X|\widetilde{X}, T^*_{X,self}) - \log P_{G_{\mathcal{X}}}(T^*_{X,self}|\widetilde{X})$
        $\quad - \log P_{G_{\mathcal{Y}}}(Y|\widetilde{Y}, T^*_{Y,self}) - \log P_{G_{\mathcal{Y}}}(T^*_{Y,self}|\widetilde{Y})$
    Generate transferred samples with gradient approximation methods:
        $T_{\mathcal{Y}} \sim P_{G_{\mathcal{Y}}}(T|X), \quad G_{\mathcal{Y}}(X) \sim P_{G_{\mathcal{Y}}}(Y|X, T_{\mathcal{Y}}).$
        $T_{\mathcal{X}} \sim P_{G_{\mathcal{X}}}(T|Y), \quad G_{\mathcal{X}}(Y) \sim P_{G_{\mathcal{X}}}(X|Y, T_{\mathcal{X}}).$
    Calculate the style loss.
        $\mathcal{L}_{sty} = -\mathbb{E}_{X \sim P_{\mathcal{X}}}\left[F(G_{\mathcal{Y}}(X), \mathcal{Y})\right] - \mathbb{E}_{Y \sim P_{\mathcal{Y}}}\left[F(G_{\mathcal{X}}(Y), \mathcal{X})\right].$
    Use Algorithm 1 to obtain pseudo alignments for the cycle loss:
        $T^*_{X,cyc} = DP(G_{\mathcal{Y}}(X), X), \quad T^*_{Y,cyc} = DP(G_{\mathcal{X}}(Y), Y).$
    Calculate the upper bound of the cycle loss.
        $\hat{\mathcal{L}}_{cyc} = -\log P_{G_{\mathcal{X}}}(X|G_{\mathcal{Y}}(X), T^*_{X,cyc}) - \log P_{G_{\mathcal{X}}}(T^*_{X,cyc}|G_{\mathcal{Y}}(X))$
        $\quad - \log P_{G_{\mathcal{Y}}}(Y|G_{\mathcal{X}}(Y), T^*_{Y,cyc}) - \log P_{G_{\mathcal{Y}}}(T^*_{Y,cyc}|G_{\mathcal{X}}(Y))$
    Update the model with the loss $\mathcal{L} = \alpha\hat{\mathcal{L}}_{self} + \beta\mathcal{L}_{sty} + \gamma\hat{\mathcal{L}}_{cyc}$.
end for
```

Dataset	Styles	#Train	#Valid	#Test	$\|V\|$	Avg Len
Yelp	Neg.	177k	2,000	500	9,943	9.55
	Pos.	266k	2,000	500		8.43
GYAFC	Inf.	52k	2,788	1,332	26,790	13.06
	For.	52k	2,247	1,019		12.47

Table 10: Data statistics. Average length is calculated on the training set.

then optimized to train the model. The full training algorithm is presented in Algorithm 2.

B Experiment Settings

B.1 Dataset and Evaluation Metrics

We use the processed datasets provided by Luo et al. (2019), which can be downloaded at `https://github.com/luofuli/DualRL`. The data statistics are shown in Table 10.

The pretrained classifier is implemented based on the *transformers* package[4], and the BLEU-4 score is the corpus BLEU implemented in the *nltk* package[5]. All results in our paper are evaluated by our implemented codes. The reported results of NAST, StyTrans, and LatentSeq in Figure 1 are averaged over three runs with different random seeds.

B.2 Network Architecture and Hyper-Parameters

NAST are implemented based on the base model, StyTrans (Dai et al., 2019) and LatentSeq (He et al., 2020). Their codes can be accessed at `https://github.com/fastnlp/style-transformer` and `https://github.com/cindyxinyiwang/deep-latent-sequence-model`.

For StyTrans, we follow their implementation and hyper-parameters for the Transformer architecture. We use 4 Transformer layers, 4 attention heads, and 256-dim hidden cells for both the encoder and the decoder. For the alignment predictor in Learnable Alignment, we utilize a one-layer Transformer decoder with the same number of attention head and dimension of hidden cells. Moreover, StyTrans utilizes a discriminator for the style loss, which is built on a 4-layer Transformer encoder with the same architecture above. The discriminator and the generator are trained adversarially. Following their implementation, in each iteration, the discriminator is trained for 10 steps and then the generator is trained for 5 steps. We utilize the Adam optimizer (Kingma and Ba, 2015) with the learning rate of $1e-4$ and the batch size of 64. We choose the gradient approximation method from the Gumbel-Softmax trick (Jang et al., 2017), the Soft-Embedding approximation (Dai et al., 2019), and the Stop-Gradient strategy

---

[4] `https://github.com/huggingface/transformers`

[5] `https://www.nltk.org/`

(He et al., 2020). We select the self-reconstruction loss weight α from $\{0.25, 0.5, 1\}$ and the cycle loss weight γ from $\{0.25, 0.5, 1\}$. We find sometimes the transfer accuracy of one direction can be much higher than that of the other direction, so we separately tune the style loss weights for two directions. To be specific, the overall objective is defined as $\alpha L_{self} + \beta_1 L_{X,sty} + \beta_2 L_{Y,sty} + \gamma L_{cyc}$, where $L_{X,sty} = -\mathbb{E}_{X\sim P_{\mathcal{X}}}\left[F(G_{\mathcal{Y}}(X), \mathcal{Y})\right]$, and $L_{Y,sty} = -\mathbb{E}_{Y\sim P_{\mathcal{Y}}}\left[F(G_{\mathcal{X}}(Y), \mathcal{X})\right]$. We select β_1, β_2 from $\{0.5, 1, 1.5, 3, 5, 10, 15\}$.

For LatentSeq, LSTM is adopted as the generator in their original models. We first replace the LSTM with an autoregressive Transformer as a baseline, which also has 4 Transformer layers, 4 attention heads, and 256-dim hidden cells. Then we replace the autoregressive Transformer with an non-autoregressive Transformer with the same architecture. The alignment predictor is a one-layer Transformer decoder with the same architecture above. However, LatentSeq utilizes a language model for the style loss, which is a 512-dim LSTM. We preserve the implementation of the language model. For optimization, we utilize the RAdam optimizer (Liu et al., 2020) with the learning rate of $1e-3$ and the batch size of 64. We also try the three gradient approximation methods. We set the cycle-reconstruction loss weight $\gamma = 1$. Following their original implementation, the self-reconstruction weight α is annealed from 1 to 0 in the first 60k steps. Similar to NAST on StyleTrans, we tune the the style loss weight on two directions separately, where we select β_1, β_2 from $\{0.15, 0.3, 0.45, 0.6, 0.75\}$ for Yelp and $\{0.5, 0.75, 1, 1.25\}$ for GYAFC.

We manually tune the hyper-parameters and select the best model according the performance on the validation set. For the Yelp dataset, the validation set does not have reference answers, so we use the geometric mean of Acc and SelfB as the overall performance. For the GYAFC dataset, we use the geometric mean of Acc and RefB as the overall performance.

B.3 Computing Devices and Running Time

In our experiment, each run uses approximately 4 Intel Xeon Gold 6226R CPUs at 2.90GHz, and 1 Nvidia Quadro RTX 6000 GPU. We present the max training step and the training time in Table 11. The best results usually appear in the first half of the training.

Model	**Yelp**	**GYAFC**
StyTrans		
+ NAST(Simple)	135k steps (∼12h)	135k steps (∼16h)
+ NAST(Learnable)	135k steps (∼16h)	135k steps (∼25h)
LatentSeq		
+ NAST(Simple)	150k steps (∼18h)	75k steps (∼16h)
+ NAST(Learnable)	150k steps (∼24h)	75k steps (∼18h)

Table 11: The max training step and the training time of our models.

	Yelp					
	Negative to Positive			**Positive to Negative**		
Model	**Acc**	**RefB**	**G2**	**Acc**	**RefB**	**G2**
DualRL	85.4	49.6	65.1	85.8	60.8	72.3
StyTrans	**87.5**	45.4	63.0	**93.1**	51.6	69.3
+ NAST (Simple)	86.2	**50.1**	**65.7**	91.6	**61.6**	**75.1**
+ NAST (Learnable)	84.2	49.2	64.3	90.7	60.0	73.8
LatentSeq	**82.3**	42.8	59.3	**86.7**	51.8	67.0
+ NAST (Simple)	**82.3**	49.0	**63.5**	80.1	**66.1**	**72.7**
+ NAST (Learnable)	80.5	**50.2**	**63.5**	78.7	65.8	72.0
	GYAFC					
	Formal to Informal			**Informal to Formal**		
Model	**Acc**	**RefB**	**G2**	**Acc**	**RefB**	**G2**
DualRL	86.6	24.7	46.2	31.2	55.9	41.7
StyTrans	87.2	28.4	49.7	46.9	55.5	51.0
+ NAST (Simple)	85.4	28.5	49.3	49.8	54.7	52.2
+ NAST (Learnable)	**92.1**	**29.9**	**52.5**	**53.7**	**55.7**	**54.7**
LatentSeq	56.8	24.7	37.2	49.8	52.4	51.1
+ NAST (Simple)	62.5	**27.4**	41.4	58.3	49.0	53.4
+ NAST (Learnable)	**69.1**	25.8	**42.1**	**59.2**	**52.6**	**55.8**

Table 12: Automatic evaluation results on two transfer directions.

C Transfer Difficulties

In Table 12, we present the results on two transfer directions of Yelp and GYAFC. On the Yelp dataset, transferring a negative sentence to a positive one is more difficult than the other direction. One possible reason is that the negative sentences are euphemistic and need changes in sentence structures when transferring to the positive sentiment. In terms of G2, the text formalization is significantly more difficult than the sentiment transfer. The difficulties of two transfer directions vary across models on GYAFC. Transferring formal sentences to informal ones is harder for DualRL, while the other direction is harder for LatentSeq.

D How to Count Aligned Word Pairs

In Section 4.5, we present cases of the word-level transfer. The aligned word pairs are counted based on the predict alignments T, following the rules below:

- If $1 \le t_i \le N$, we record a pair $x_{t_i} \to y_i$.
- If $t_i = 0$, the transferred word is unaligned, and we record a pair $[Mask] \to y_i$.
- If a source word x_i is not aligned with any transferred word, we record a pair $x_i \to [Del]$.

We then collect all word pairs that have the same source word and calculate the proportion of different transferred words. The results shown in Table 8 is obtained on the test set of two datasets.

HyKnow: End-to-End Task-Oriented Dialog Modeling with Hybrid Knowledge Management

Silin Gao[1]*, Ryuichi Takanobu[1]*, Wei Peng[2], Qun Liu[2], Minlie Huang[1]†
[1] CoAI Group, DCST, IAI, BNRIST, Tsinghua University, Beijing, China
[2] Huawei Technologies, Shenzhen, China
[1] `gsl16@tsinghua.org.cn`, `gxly19@mails.tsinghua.edu.cn`,
`aihuang@tsinghua.edu.cn`
[2] `peng.wei1@huawei.com`, `qun.liu@huawei.com`

Abstract

Task-oriented dialog (TOD) systems typically manage structured knowledge (e.g. ontologies and databases) to guide the goal-oriented conversations. However, they fall short of handling dialog turns grounded on unstructured knowledge (e.g. reviews and documents). In this paper, we formulate a task of modeling TOD grounded on both structured and unstructured knowledge. To address this task, we propose a TOD system with hybrid knowledge management, HyKnow. It extends the belief state to manage both structured and unstructured knowledge, and is the first end-to-end model that jointly optimizes dialog modeling grounded on these two kinds of knowledge. We conduct experiments on the modified version of MultiWOZ 2.1 dataset, where dialogs are grounded on hybrid knowledge. Experimental results show that HyKnow has strong end-to-end performance compared to existing TOD systems. It also outperforms the pipeline knowledge management schemes, with higher unstructured knowledge retrieval accuracy.

1 Introduction

Recently, Task-Oriented Dialog (TOD) systems (Mehri et al., 2019; Zhang et al., 2020a,b; Le et al., 2020; Hosseini-Asl et al., 2020; Peng et al., 2020; Li et al., 2021) have achieved promising performance on accomplishing user goals. Most systems typically query *structured knowledge* such as tables and databases based on the user goals, and use the query results to guide the generation of system responses, as shown in the first dialog turn in Fig. 1.

However, real-world task-oriented conversations often step into dialog turns which are grounded on *unstructured knowledge* (Feng et al., 2020), such as passages and documents. For example, as the second dialog turn in Fig. 1 shows, the user asks about customers' favorite food at *Pizza Hut*, which is grounded on the customer reviews of this restaurant. Current TOD systems fall short of handling such dialog turns since they cannot utilize relevant unstructured knowledge. This deficiency may interrupt the dialog process, causing difficulties in tracking user goals and generating system responses.

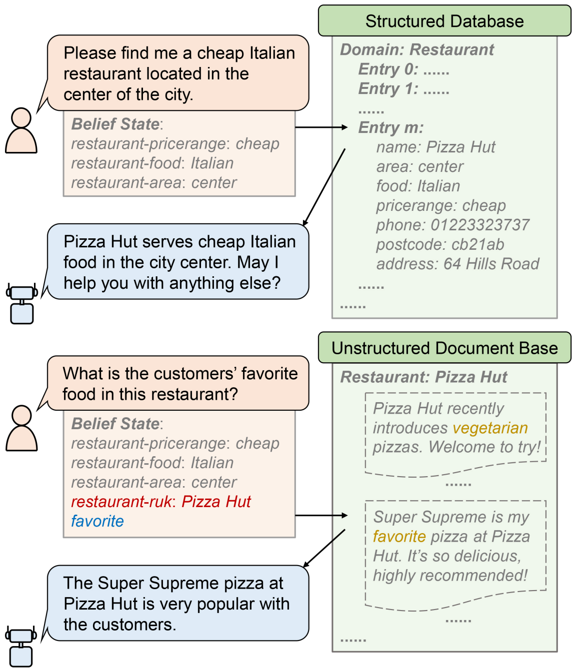

Figure 1: Illustration of task-oriented dialog modeling with hybrid knowledge management. Words in red and blue illustrate the new domain-slot-value triple and the topic of user utterance that we introduce into the belief state, respectively. Words in yellow illustrate the topics of documents that we extract through preprocessing.

In this work, we consider incorporating more various forms of domain knowledge into the TOD systems. Therefore, we define a task of modeling TOD whose turns involve either structured or unstructured knowledge. In turns involving struc-

---

*Equal contribution.
†Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1591–1602
August 1–6, 2021.

tured knowledge, the system needs to track the user goals as triples and use them to perform database queries, whose results are used to generate the system response. While in turns involving unstructured knowledge, the system manages a document base to retrieve relevant references for generating the response.

To address our defined task, we propose a task-oriented dialog system with **Hy**brid **Know**ledge management (HyKnow). This model extends the belief state to handle TODs grounded on hybrid knowledge, and further uses the extended belief state to perform both database query and document retrieval, whose outputs are thereby used to generate the final response. We consider two implementations of our system, with different schemes of extended belief state decoding. Both implementations are in an end-to-end multi-stage sequence-to-sequence (Seq2Seq) (Lei et al., 2018; Liang et al., 2020; Zhang et al., 2020a,b) framework, where dialog modeling grounded on the two kinds of knowledge can be jointly optimized.

We evaluate our system on the modified version of MultiWOZ 2.1 (Kim et al., 2020) dataset, where dialogs are grounded on hybrid knowledge. Experimental results show that HyKnow outperforms existing TOD systems which do not leverage large pretrained language models, no matter whether they add extra unstructured knowledge management or not. It also has a higher accuracy in unstructured knowledge retrieval, compared to the pipeline knowledge management schemes.

Our contributions are summarized as below:

- We formulate a task of modeling TOD grounded on both structured and unstructured knowledge, to incorporate more domain knowledge into the TOD systems.
- We propose a TOD system HyKnow to address our proposed task. It extends the belief state to manage hybrid knowledge, and is the first end-to-end model to jointly optimize dialog modeling grounded on the two kinds of knowledge.
- Experimental results show that HyKnow has strong performance in dialog modeling grounded on hybrid knowledge.[1]

2 Related Work

TOD systems usually use belief tracking, i.e. dialog state tracking (DST) to trace the user goals, i.e. *belief states*, through multiple dialog turns (Williams et al., 2013; Henderson et al., 2014). The states are converted into a representation of constraints based on different schemes to query the databases (El Asri et al., 2017; Budzianowski et al., 2018; Rastogi et al., 2020; Zhu et al., 2020). The entry matching results are then used to generate the system response.

With the development of intelligent assistants, the system should have a good command of massive external knowledge to better accomplish complicated user goals and improve user satisfaction. To realize this, some researchers (Zhao et al., 2017; Yu et al., 2017; Akasaki and Kaji, 2017) equip the system with chatting capability to address both task and non-task content in TODs. Other studies apply knowledge graph (Liao et al., 2019; Yang et al., 2020) or tables via SQL (Yu et al., 2019) to enrich the knowledge of TOD systems. However, all these studies are still limited in dialog modeling grounded on structured knowledge.

There are a couple of studies to integrate unstructured knowledge into TOD modeling recently. Kim et al. (2020) introduce knowledge snippets to answer follow-up questions out of the coverage of databases. Feng et al. (2020) formulate document-grounded dialog for information seeking tasks. However, they only focus on dialog turns grounded on unstructured knowledge instead. In this paper, we aims to fill the gap of managing domain-specific knowledge with various sources and structures in traditional TOD systems.

3 Task Definition

In this section, we introduce our formulation of modeling TOD grounded on hybrid knowledge. In particular, we assume that each dialog turn in TOD is grounded on either structured or unstructured knowledge. We formulate the modeling of the two kinds of dialog turns separately.

In turns that are grounded on structured knowledge, the system needs to track user goals, i.e. the belief state, as domain-slot-value triples, and then query a database (DB) to guide response generation. Specifically, we denote the user utterance and the system response at turn t as U_t and R_t respectively. Given the dialog context $C_t = [U_{t-k}, R_{t-k}, ..., U_t]$ and previous belief state B_{t-1}, the system needs to generate current belief state B_t, which is formulated as $B_t = f_b^{(s)}(C_t, B_{t-1})$. Then the system performs DB query based on B_t to get the matching

---

[1] The code is available at `https://github.com/truthless11/HyKnow`

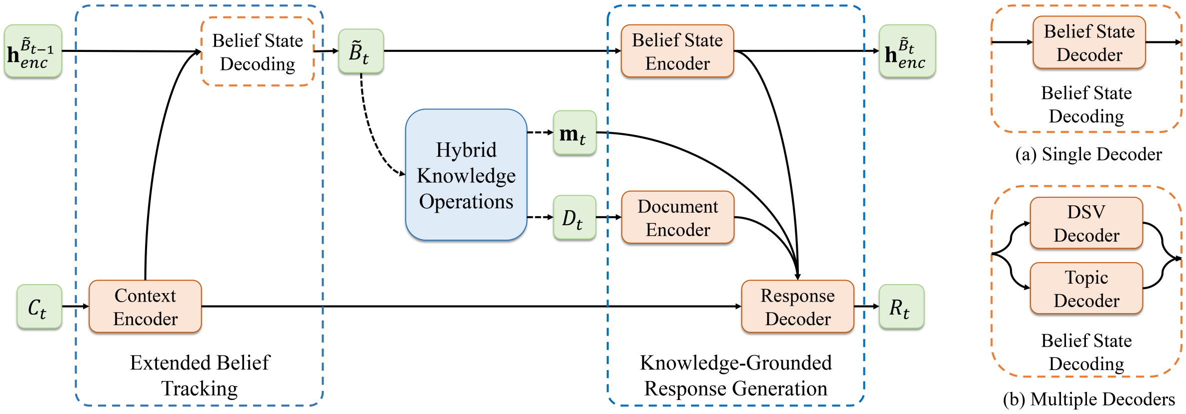

(a) Single Decoder

(b) Multiple Decoders

Figure 2: Overview of HyKnow. Solid arrows denote the input/output of the encoders or decoders. Dashed arrows denote the knowledge operations. C_t, $\mathbf{m}_t$, D_t and R_t represent turn t's dialog context, DB query result, relevant document and system response. $\tilde{B}_t$ and $\mathbf{h}_{enc}^{\tilde{B}_t}$ denote the extended belief state and its hidden states at turn t. The decoding of $\tilde{B}_t$ (orange dashed box) is implemented in two different ways: (a) using a single decoder to generate the whole state, and (b) using two decoders to generate the domain-slot-value (DSV) triples and the topic separately.

result $\mathbf{m}_t$. In this paper, we follow Budzianowski et al. (2018) to represent $\mathbf{m}_t$ as a vector indicating the number of matched entities and whether the booking is available or not. Afterwards, the system generates the response R_t, formulated as $R_t = f_r^{(s)}(C_t, B_t, \mathbf{m}_t)$.

In turns that are grounded on unstructured knowledge, the system manages a document base to guide response generation, which contains lists of documents characterized by different domains and entities, as showed in Fig. 1. Specifically, given the dialog context C_t, the system first retrieves a relevant document D_t in the document base, formulated as $D_t = f_d^{(u)}(C_t)$. Then the system generates the response R_t based on C_t and retrieved D_t, which is formulated as $R_t = f_r^{(u)}(C_t, D_t)$. Noting that the original belief state is not updated in the unstructured knowledge-grounded turns, namely $B_t = B_{t-1}$. However, in this paper, we introduce extra belief state extension to facilitate the document retrieval.

4 Proposed Framework

Fig. 2 shows an overview of our proposed system HyKnow with end-to-end sequence-to-sequence (Seq2Seq) implementations. It addresses our proposed task in three steps. First, it uses the **extended belief tracking** to track user goals through dialog turns that involve hybrid knowledge. Secondly, it performs **hybrid knowledge operations** based on the extended belief state, to search structured and unstructured knowledge that is relevant to the user goals. Finally, it uses the extended belief state and relevant knowledge to perform the **knowledge-grounded response generation**.

4.1 Extended Belief Tracking

Belief State Extension. We define an extended belief state $\tilde{B}_t$ which is applicable to track user goals in TODs that are grounded on both structured and unstructured knowledge. Specifically, in turns that are grounded on structured knowledge, $\tilde{B}_t$ is same as the original B_t, which describes user goals as domain-slot-value triples. While in turns that are grounded on unstructured knowledge, $\tilde{B}_t$ has an additional slot *ruk* to indicate that current dialog turn **r**equires **u**nstructured **k**nowledge. The prefix and value of the slot *ruk* represent the involved domain and entity, e.g. *restaurant-ruk*: *Pizza Hut* colored in red in Fig. 1. We denote the combination of original and newly introduced domain-slot-value triples as DSV_t. In addition, the *topic* of U_t is abstracted in $\tilde{B}_t$ as a word sequence T_t in each unstructured knowledge-grounded turn, e.g. *favorite* colored in blue in Fig. 1.

Extended Belief State Decoding. Following Seq2Seq framework, we first use the *context encoder* to encode the dialog context C_t, whose last output is used as the initial hidden state of decoders. Based on the hidden states of context encoder $\mathbf{h}_{enc}^{C_t}$ and previous extended belief state $\mathbf{h}_{enc}^{\tilde{B}_{t-1}}$, we then decode the current extended belief state $\tilde{B}_t$ under two schemes, which are described as below.

Since DSV_t and T_t are grounded on quite different vocabularies, we consider implementing the decoding of $\widetilde{B}_t$ in two ways: (a) using the belief state decoder to generate the whole $\widetilde{B}_t$, and (b) using the DSV decoder and the topic decoder to generate DSV_t and T_t separately. Each implementation has its own advantages over the other. Specifically, in the single-decoder implementation, the decoding of DSV_t and T_t can be jointly optimized via shared parameters:

$$\widetilde{B}_t = \text{Seq2Seq}^{(b)}(C_t|\mathbf{h}_{enc}^{\widetilde{B}_{t-1}}). \tag{1}$$

While in the multi-decoder implementation, the decoding of DSV_t and T_t are fitted to their own smaller decoding spaces (vocabularies), and thus the generation of $\widetilde{B}_t$ can be decomposed into two simpler decoding processes:

$$\begin{aligned} DSV_t &= \text{Seq2Seq}^{(dsv)}(C_t|\mathbf{h}_{enc}^{\widetilde{B}_{t-1}}), \\ T_t &= \text{Seq2Seq}^{(t)}(C_t|\mathbf{h}_{enc}^{\widetilde{B}_{t-1}}), \\ \widetilde{B}_t &= [DSV_t, T_t]. \end{aligned} \tag{2}$$

4.2 Hybrid Knowledge Operations

Based on the extended belief state $\widetilde{B}_t$, we conduct both DB query and document retrieval to get the query result $\mathbf{m}_t$ and the relevant document D_t, which are used to guide the generation of response. In the operation of DB query, we simply match the original triples in $\widetilde{B}_t$ with the DB entries. While in the operation of document retrieval, we first preprocess the document base to extract the topic of each document as its retrieval index, e.g. *vegetarian* and *favorite* colored in yellow in Fig. 1. Then we use the extended part of $\widetilde{B}_t$ to match the domain, entity and extracted topic of each document, and select the best-matched one as D_t.[2]

4.3 Knowledge-Grounded Response Generation

We generate system response based on the dialog context C_t, the extended belief state $\widetilde{B}_t$, and the outputs of hybrid knowledge operations $\mathbf{m}_t$ and D_t. We first use the same context encoder in Sec. 4.1 to encode C_t. Moreover, we use the *belief state encoder* and the *document encoder* to encode $\widetilde{B}_t$ and D_t into hidden states $\mathbf{h}_{enc}^{\widetilde{B}_t}$ and $\mathbf{h}_{enc}^{D_t}$, respectively. Based on the hidden states of all the encoders and the vector $\mathbf{m}_t$, we use the *response decoder* to generate the system response R_t, formulated as:

$$\begin{aligned} \mathbf{h}_{enc}^{\widetilde{B}_t} &= \text{Encoder}^{(\text{b})}(\widetilde{B}_t), \\ \mathbf{h}_{enc}^{D_t} &= \text{Encoder}^{(\text{d})}(D_t), \\ R_t &= \text{Seq2Seq}^{(r)}(C_t|\mathbf{h}_{enc}^{\widetilde{B}_t}, \mathbf{h}_{enc}^{D_t}, \mathbf{m}_t), \end{aligned} \tag{3}$$

where $\text{Encoder}^{(\text{b})}$ and $\text{Encoder}^{(\text{d})}$ denote the belief state encoder and the document encoder.

Following previous TOD systems with Seq2Seq architectures (Lei et al., 2018; Liang et al., 2020; Zhang et al., 2020a,b), we use one-layer, bidirectional GRU as encoders and standard GRU as decoders. We also apply global attention (Bahdanau et al., 2015) and copy mechanism (Gu et al., 2016) in all the Seq2Seq processes, to improve the context-awareness of decoding $\widetilde{B}_t$ and R_t.

4.4 Model Training

HyKnow is optimized through supervised training. Specifically, each dialog turn in the training data is initially labeled with the original belief state and the relevant document. We extend the belief state label based on the domain, entity and extracted topic of the relevant document. Then the extended belief state label and the reference response are used to calculate the cross-entropy loss with the generated $\widetilde{B}_t$ and R_t, respectively. We sum the two losses together and perform gradient descent in each turn to optimize the model parameters. In our paper, the dialog context C_t is set as the concatenation of previous system response R_{t-1} and current user utterance U_t.[3]

5 Experimental Settings

5.1 Dataset

We evaluate our proposed system on the modified MultiWOZ 2.1 (Kim et al., 2020) dataset, where crowd-sourcing workers are hired to insert additional turns into the original MultiWOZ dialogs. Each newly inserted turn is grounded on unstructured knowledge in one of the four domains: restaurant, hotel, taxi and train, with the label of its relevant document in the document base. While the other three MultiWOZ domains (attraction, hospital and police) are not involved in these new turns.[4]

---

[2] See Appendix A for more details of the document preprocessing and matching.

[3] See Appendix B for more implementation details.

[4] See Appendix C for details of data statistics.

Model	Pretrained LM	Inform	Success	BLEU	METEOR	ROUGE-L	Combined
UniConv	none	71.5	61.8	18.5	37.8	40.5	85.7
LABES-S2S	none	76.5	65.3	17.8	36.8	39.9	88.7
UniConv + BDA	-	72.0	62.6	16.9	35.7	38.9	84.2
LABES-S2S + BDA	-	77.1	66.2	15.7	33.8	37.8	87.4
HyKnow (Single)	none	**81.9**	**68.3**	**19.0**	**38.5**	40.9	**94.1**
- w/o Joint Optim	none	78.5	65.7	18.3	36.9	39.6	90.4 (-3.7)
HyKnow (Multiple)	none	79.1	67.6	18.7	38.1	**41.0**	92.1
- w/o Joint Optim	none	77.7	65.4	18.0	36.6	39.5	89.6 (-2.5)
SimpleTOD	GPT-2	81.7	67.9	14.5	34.2	37.0	89.3
SimpleTOD + BDA	-	83.3	68.6	14.8	33.6	36.5	90.8

Table 1: End-to-end evaluation results on modified MultiWOZ 2.1. "+" denotes the combination of Beyond Domain APIs (BDA) with E2E TOD models. Best results among light-weight systems (i.e. above internal dividing line) are marked in bold. Evaluation metrics are described and marked in bold in Sec. 6.1.

5.2 Baselines

We compare HyKnow with 1) existing end-to-end (E2E) TOD models and dialog state tracking (DST) models, to show the benefits of incorporating unstructured knowledge management into TOD modeling. We also compare HyKnow with 2) unstructured knowledge management models, to investigate our system's document retrieval performance. For the comparison with pipeline systems that have hybrid knowledge management, we also consider the combinations of 1) and 2) as our baselines.

E2E TOD Models and DST Models. We consider three baseline E2E TOD models with different types of structures: **UniConv** (Le et al., 2020) uses a structured fusion (Mehri et al., 2019) design, **LABES-S2S** (Zhang et al., 2020a) uses a multi-stage Seq2Seq (Lei et al., 2018) architecture, and **SimpleTOD** (Hosseini-Asl et al., 2020) is based on a single auto-regressive language model initialized from GPT-2 (Radford et al., 2019). All three E2E models only manage structured knowledge (database) in their TOD modeling. In addition to E2E TOD models, we also compare HyKnow with existing DST models in the belief tracking evaluation. Specifically, we use **TRADE** (Wu et al., 2019) and **TripPy** (Heck et al., 2020) as two DST baselines, which are representative BERT-free and BERT-based DST models, respectively.

Unstructured Knowledge Management Models. We first compare our system with Beyond Domain APIs (**BDA**) (Kim et al., 2020). This baseline model uses two classification modules based on BERT (Devlin et al., 2019) to detect unstructured knowledge-grounded dialog turns and retrieve relevant documents, respectively. Moreover, we use standard information retrieval (IR) systems **TF-IDF** (Manning et al., 2008) and **BM25** (Robertson and Zaragoza, 2009) as the other two baseline models.

Combinations. We combine the unstructured knowledge management model BDA with every DST or E2E TOD model. Specifically, BDA detects dialog turns that are grounded on unstructured knowledge, and uses a fine-tuned GPT-2 to generate responses in these turns, based on the dialog context and retrieved documents. While the DST or E2E TOD model handles the rest dialog turns which are grounded on structured knowledge.

Noting that TripPy and SimpleTOD use large-scale pretrained language models (LM) to improve their dialog modeling performance, which requires large model sizes and computing resources. For fair comparisons, we distinguish them from other light-weight models in our experiments.

6 Results and Analysis

We test our system's performance under both the single-decoder and multi-decoder belief state decoding implementations, denoted as HyKnow (Single) and HyKnow (Multiple), respectively. Both implementations of HyKnow come to the same conclusions when compared with the baseline models, which are described in detail below.

6.1 End-to-End Evaluation

Table 1 shows our experimental results of the end-to-end (E2E) evaluation, where we evaluate the task completion rate and language quality of system responses. In terms of the task completion rate, we measure whether the system provides correct entities (**Inform** rate) and answers all the requested information (**Success** rate) in a dialog, following Budzianowski et al. (2018). For the evaluation of language quality, we adopt commonly used metrics

Model	Inform	Success	BLEU	Combined
UniConv	84.2	71.8	19.0	97.3
LABES-S2S	83.6	74.2	18.3	97.2
UniConv + BDA	85.8	73.9	19.3	99.4
LABES-S2S + BDA	85.0	75.3	18.9	99.1
HyKnow	**87.2**	**76.5**	**19.5**	**101.4**
SimpleTOD	87.5	76.4	16.3	98.3
SimpleTOD + BDA	89.0	77.2	17.0	100.1

Table 2: Context-to-response generation results on modified MultiWOZ 2.1. All symbols and markings have the same meaning as in Table 1.

BLEU (Papineni et al., 2002), **METEOR** (Banerjee and Lavie, 2005) and **ROUGE-L** (Lin, 2004). Moreover, we use **Combined** score computed by $(Inform + Success) \times 0.5 + BLEU$ for overall evaluation, as suggested by Eric et al. (2020).

We find that HyKnow has better task completion rate than the light-weight E2E TOD models, which is comparable with SimpleTOD who uses large-scale pretrained GPT-2. It also generates responses with better language quality compared to all the E2E models. This is because our extended belief state can distinguish whether a dialog turn is grounded on structured or unstructured knowledge, which avoids the confusion between handling the two kinds of turns. In addition, we manage the document base to provide relevant references for generating the response, which guide our system to give more appropriate responses in turns that are grounded on unstructured knowledge.

We also observe that HyKnow outperforms the combinations of BDA and light-weight E2E TOD model. This indicates that our end-to-end model framework has advantages over the pipeline structures of combination models. In particular, dialog modeling grounded on the structured and unstructured knowledge are integrated in a uniform Seq2Seq architecture in our system, where they are jointly optimized to an overall better performance. Although HyKnow does not significantly outperform the combination of BDA and SimpleTOD, our system has lower deployment cost since it is trained end-to-end.

6.2 Context-to-Response Generation

We also conduct evaluations on the context-to-response (C2R) generation, where systems directly use the oracle belief state and knowledge to generate the response. The experimental results are shown in Table 2, where we observe the same conclusions as in the E2E evaluation (Table 1). This again shows our system's superiority in TOD modeling grounded on hybrid knowledge.

Model	Pretrained LM	Joint Goal
TRADE	none	42.9
UniConv	none	45.5
LABES-S2S	none	46.0
TRADE + BDA	-	43.8
UniConv + BDA	-	46.5
LABES-S2S + BDA	-	46.8
HyKnow (Single)	none	**48.0**
- w/o Joint Optim	none	46.2 (-1.8)
HyKnow (Multiple)	none	47.6
- w/o Joint Optim	none	45.6 (-2.0)
TripPy	BERT	50.4
SimpleTOD	GPT-2	48.4
TripPy + BDA	-	51.2
SimpleTOD + BDA	-	49.8

Table 3: Original turns' belief tracking results on modified MultiWOZ 2.1. "+" denotes the combination of BDA with DST/E2E models. The best result among light-weight systems (i.e. above internal dividing line) is marked in bold. The evaluation metric is described and marked in bold in Sec. 6.3.

Additionally, we observe that HyKnow's performance gap between E2E and C2R evaluations is smaller than the baseline models, reflected in the smaller variations of the combined score. This shows that the belief state and knowledge provided by our system are probably closer to the oracle and may give stronger guidance to generate a response.

6.3 Knowledge Management

To further investigate our system's end-to-end performance, we conduct evaluations on the intermediate knowledge management. In particular, we evaluate the structured and unstructured knowledge management separately in the original and newly inserted dialog turns. In the original turns grounded on structured knowledge, we evaluate the belief tracking performance which directly determines the database query accuracy. Specifically, we use the **Joint Goal** accuracy (Henderson et al., 2014) to measure whether belief states are predicted correctly in a dialog turn. While in newly inserted turns grounded on unstructured knowledge, we adopt standard information retrieval metrics **R@1** and **MRR@5** to evaluate the document retrieval performance. Table 3 and 4 shows our evaluation results of belief tracking and document retrieval, respectively.

In terms of belief tracking, HyKnow outperforms the light-weight DST/E2E models. This is because our extended belief tracking can detect the

Model	Type	MRR@5	R@1
TF-IDF	standard IR	68.7	54.1
BM25	standard IR	69.2	52.5
BDA	classification	80.6	69.8
HyKnow (Single)	topic match	**81.7**	**80.2**
- w/o Joint Optim	topic match	80.1 (-1.6)	77.8 (-2.4)
HyKnow (Multiple)	topic match	81.1	79.5
- w/o Joint Optim	topic match	79.7 (-1.4)	77.4 (-2.1)

Table 4: Newly inserted turns' document retrieval results on modified MultiWOZ 2.1. Best results are marked in bold. Evaluation metrics are described and marked in bold in Sec. 6.3.

newly inserted turns apart from the original turns (via the slot *ruk*), which improves our system's awareness on deciding when to update the original triples in the belief state. HyKnow also has better belief tracking performance than the combinations of BDA and light-weight DST/E2E model. This is because error propagation on updating belief states is eliminated in our system compared to the pipeline framework: The pipeline system either updates the belief state or retrieves the document in one turn, but HyKnow can perform both operations in the nature of its E2E design. Although the belief tracking performance of HyKnow is not as good as that of TripPy and SimpleTOD, our system does not use large-scale pretrained BERT or GPT-2 and is thus computational cheaper.[5]

In the document retrieval evaluation, we find that HyKnow outperforms the unstructured knowledge management models, especially on the R@1 metric. This shows that our system's document retrieval scheme with topic matching has a higher accuracy, compared to the classifier-based BDA and the standard information retrieval (IR) systems. Specifically, HyKnow retrieves documents based on the highly simplified semantic information, i.e. the topic, which reduces the complexity of the retrieval process. This makes the retrieval scheme of HyKnow more concise and effective than the baseline models, who directly calculate the relevance of dialog context to every document content.

6.4 Single vs. Multiple Decoders

We then compare our two implementations of extended belief state decoding. We calculate the vocabularies of DSV triples, the topic and their combination (which are 709, 166 and 862), and observe that the last one approximately equals to the sum of the former two. This confirms our assumption in Sec. 4.1 that DSV triples and topic have quite different vocabularies, which motivates the multi-decoder implementation in belief state decoding.

However, we find that HyKnow (Single) outperforms HyKnow (Multiple) in both E2E and knowledge management evaluations, as shown in Table 1, 3 and 4. This shows that the decoding of DSV triples and topic can benefit from the joint optimization via shared parameters, although they are grounded on quite different vocabularies. The superiority of joint optimization further implies that the structured and unstructured knowledge management in TOD modeling have a positive correlation, since they commonly involve task-specific domain knowledge and entities. Therefore, the two kinds of knowledge management can learn from each other through joint training, and achieve overall better performance compared to separating them apart.

6.5 Ablation Study

We ablate the joint optimization of structured and unstructured knowledge-grounded TOD modeling to investigate its role in our framework, denoted as w/o Joint Optim in Table 1, 3 and 4. Specifically, we train two HyKnow models separately on the original and newly inserted dialog turns, and use them to handle TOD grounded on structured and unstructured knowledge, respectively. To determine which model should be used, the oracle label of slot *ruk* is used to judge which knowledge type the current dialog turn is grounded on.

We observe that removing joint optimization brings HyKnow evident performance declines in the end-to-end evaluation (Table 1). This suggests that joint optimization plays a significant role in improving HyKnow's end-to-end performance, where TOD modeling grounded on the two kinds of knowledge can benefit each other by learning shared parameters. The ablation of joint optimization also causes performance declines in HyKnow's knowledge management (Table 3 and 4). This again indicates that the two kinds of knowledge management are positively correlative and can get benefit from joint training.

6.6 Between Structured and Unstructured Knowledge

In this section, we investigate how the newly inserted dialog turns (grounded on unstructured knowledge) affect systems' E2E performance in the original dialog turns (grounded on structured knowledge). Specifically, we evaluate systems'

---

[5] See Appendix D for details on model size comparison.

Test Set	Model	Joint Goal	Inform	Success	BLEU	METEOR	ROUGE-L	Combined
Original	LABES-S2S + BDA	49.0	82.1	69.8	17.8	37.1	40.2	93.8
	SimpleTOD + BDA	51.8	85.6	70.9	16.3	34.5	38.6	94.6
	HyKnow (Single)	49.2	82.3	69.4	18.0	37.3	40.2	93.9
Modified	LABES-S2S + BDA	46.8 (-2.2)	77.1	66.2	17.7	36.8	39.6	89.4 (-4.4)
	SimpleTOD + BDA	49.8 (-2.0)	83.3	68.6	15.8	33.6	37.8	91.8 (-2.8)
	HyKnow (Single)	48.0 (-1.2)	81.9	68.3	17.8	37.2	39.5	92.9 (-1.0)

Table 5: End-to-end evaluation results on the original and modified MultiWOZ 2.1 test set. The evaluation is conducted only in the original dialog turns.

Model	Original			Newly Inserted		
	Cohe.	Info.	Corr.	Cohe.	Info.	Corr.
SimpleTOD	2.58	2.56	2.44	2.50	2.04	2.14
SimpleTOD + BDA	2.56	2.60	**2.46**	2.52	2.30	2.22
HyKnow (Single)	**2.60**	**2.62**	2.42	**2.56**	**2.36**	**2.50**

Table 6: Human evaluation results on modified MultiWOZ 2.1, results in original and newly inserted turns are shown separately.

E2E performance on both the original and modified MultiWOZ 2.1 test sets. This evaluation is conducted only in the original dialog turns, which is different from the E2E evaluation conducted in all turns (Table 1). Table 5 shows the results of this experiment, where we compare HyKnow (Single) with strong combination models.

We find that all the models' performance is degraded when transferred from the original to the modified test set. This indicates that the inserted turns grounded on new knowledge may interrupt the original dialogs, which complicates the dialog process and causes difficulties in the original turns' dialog modeling.

However, we observe that HyKnow (Single) suffers from less reduction compared to the baseline combination models. This shows that our system has a stronger resistance to the interruptions of newly inserted turns, which benefits from our end-to-end modeling. Specifically, HyKnow jointly optimizes dialog modeling of the original and newly inserted turns in a uniform end-to-end framework. This unified modeling approach improves our system's flexibility in switching between the two kinds of turns, and thus makes it more competent in handling the complicated dialog process.

6.7 Human Evaluation

There is still a gap between the evaluation results of automatic metrics and the real E2E performance of TOD systems. Therefore, we conduct human evaluation to more adequately test our system's E2E performance. In particular, we compare HyKnow (Single) with a strong E2E baseline SimpleTOD and its combination with BDA.

We conduct human evaluation separately on the two types (original and newly inserted) of dialog turns. Specifically, we sample fifty dialog turns of each type and ask the judges to evaluate each turn's system response on three aspects. **Coherence** (Cohe.) measures how well the response is coherent with the dialog context. **Informativeness** (Info.) measures how well the response can provide sufficient information that meets the user requests. **Correctness** (Corr.) measures how well the information in response is consistent with the ground truth knowledge, i.e. relevant DB entries or documents. All the three aspects are scored on a Liker scale of 1-3, which denotes *bad*, *so-so* and *good*.

Table 6 shows our human evaluation results. In the original dialog turns, HyKnow (Single) scores close to SimpleTOD and its combination with BDA on all the three aspects. This indicates that our proposed light-weight system is comparable with the large GPT-2 based models in managing structured knowledge to generate the response. In addition, our model outperforms the two baseline models in the newly inserted dialog turns. Specifically, HyKnow (Single) generates responses with significantly better informativeness and correctness than SimpleTOD. This again shows that the management of unstructured knowledge is beneficial for generating appropriate responses. Compared to the combination of SimpleTOD and BDA, the responses generated by HyKnow (Single) also achieve much better correctness, which benefits from our model's higher document retrieval accuracy (as shown in Table 4).

6.8 Case Study

An example dialog segment (U_1, B_1, R_1, U_2) and corresponding output results of each model (B_2, D_2, R_2) are presented in Table 7. Without access to unstructured document base, SimpleTOD misunderstands the user query, and instead recognizes

<table>
<tr><td colspan="2">U_1: Hello, I would like to find a **hotel** that has WiFi in the north part of the town.
B_1: hotel-area: north, hotel-internet: yes
R_1: The **Arbury Lodge Guesthouse** is one of 12 options for you. Shall I make a reservation for you?
U_2: Do they provide **Italian breakfast**?</td></tr>
<tr><td>Human</td><td>B_2: hotel-area: north, hotel-internet: yes
D_2: Full English **breakfast** is served every morning at Arbury Lodge Guesthouse.
R_2: **No**, but they have a full English breakfast. Did you still want to book a room here?</td></tr>
<tr><td>SimpleTOD</td><td>B_2: hotel-area: north, hotel-internet: yes, **restaurant-food: Italian**
R_2: Da Vinci Pizzeria serves Italian food and is located in the north.</td></tr>
<tr><td>SimpleTOD + BDA</td><td>B_2: hotel-area: north, hotel-internet: yes
D_2: German, Italian, Spanish and English are **spoken** at Carolina Bed and Breakfast.
R_2: **Yes**, the hotel has a Italian breakfast. Would you like to make a reservation?</td></tr>
<tr><td>HyKnow (Single)</td><td>$\widetilde{B}_2$: hotel-area: north, hotel-internet: yes, **hotel-ruk: Arbury Lodge Guesthouse**, **breakfast**
D_2: Full English **breakfast** is served every morning at Arbury Lodge Guesthouse.
R_2: **No**, they provides full English breakfast. Would you like me to book a room for you?</td></tr>
</table>

Table 7: Comparison of the belief tracking, document retrieval and response generation results when dialog steps into a newly inserted turn of modified MultiWOZ 2.1. "Human" denotes ground truth results.

the term "Italian" in user utterance as a constraint to update the belief state. As a result, the system makes an inappropriate recommendation. By combining with BDA, SimpleTOD predicts correct belief state, but fails in finding the relevant document, thus providing a wrong answer. This is because the wrong document's content has many common words with the dialog context, e.g. "Italian" and "Breakfast", which mislead the retrieval process of BDA. In contrast, HyKnow gives a proper response with accurate information as it identifies the entity ("Arbury Lodge Guesthouse") and captures the topic ("breakfast") to avoid the misleading of common words in document retrieval.

7 Conclusion

In this paper, we define a task of modeling TOD with access to both structured and unstructured knowledge. To address this task, we propose a TOD system HyKnow which uses an E2E framework to jointly optimize TOD modeling grounded on the two kinds of knowledge. In the experiments, HyKnow shows strong performance in modeling TOD with hybrid knowledge management, compared to existing TOD systems and their pipeline extensions. For future work, we plan to incorporate large-scale pretrained language models into our proposed system to further enhance its performance. Furthermore, we consider evaluating our system on different scenarios where dialogs are grounded on hybrid knowledge.

Acknowledgments

This work was partly supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005. We would like to thank colleagues from HUAWEI for their constant support and valuable discussion.

References

Satoshi Akasaki and Nobuhiro Kaji. 2017. Chat detection in an intelligent assistant: Combining task-oriented and non-task-oriented spoken dialogue systems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1319.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Layla El Asri, Hannes Schulz, Shikhar Kr Sarma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Song Feng, Hui Wan, Chulaka Gunasekara, Siva Patel, Sachindra Joshi, and Luis Lastras. 2020. Doc2dial: A goal-oriented document-grounded dialogue dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8118–8128.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 263–272.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33.

Seokhwan Kim, Mihail Eric, Karthik Gopalakrishnan, Behnam Hedayatnia, Yang Liu, and Dilek Hakkani-Tur. 2020. Beyond domain apis: Task-oriented conversational modeling with unstructured knowledge access. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 278–289.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hung Le, Doyen Sahoo, Chenghao Liu, Nancy Chen, and Steven CH Hoi. 2020. Uniconv: A unified conversational neural architecture for multi-domain task-oriented dialogues. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1860–1877.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Jinchao Li, Qi Zhu, Lingxiao Luo, Lars Liden, Kaili Huang, Shahin Shayandeh, Runze Liang, Baolin Peng, Zheng Zhang, Swadheen Shukla, et al. 2021. Multi-domain task completion dialog challenge ii at dstc9. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, Ninth Dialog System Technology Challenge Workshop*.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020. Moss: End-to-end dialog system framework with modular supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8327–8335.

Lizi Liao, Ryuichi Takanobu, Yunshan Ma, Xun Yang, Minlie Huang, and Tat-Seng Chua. 2019. Deep conversational recommender in travel. *arXiv preprint arXiv:1907.00710*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge university press.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. Structured fusion networks for dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 165–177.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan W Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Shiquan Yang, Rui Zhang, and Sarah Erfani. 2020. Graphdialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1878–1888.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979.

Zhou Yu, Alan W Black, and Alexander I Rudnicky. 2017. Learning conversational systems that interleave task and non-task content. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4214–4220.

Yichi Zhang, Zhijian Ou, Min Hu, and Junlan Feng. 2020a. A probabilistic end-to-end task-oriented dialog model with latent belief states towards semi-supervised learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9207–9219.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020b. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 27–36.

Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. 2020. Crosswoz: A large-scale chinese cross-domain task-oriented dialogue dataset. *Transactions of the Association for Computational Linguistics*, 8:281–295.

A Document Preprocessing and Matching

We preprocess the document base of the modified MultiWOZ 2.1 (Kim et al., 2020) dataset to extract the topic of each document, which are used as its retrieval index in the unstructured knowledge management. Based on the TF-IDF (Manning et al., 2008) algorithm, we perform the topic word extraction domain-by-domain in a two-step procedure. First, we choose the top-three keywords with the highest TF-IDF scores in each document as its topic candidates. Then we filter the candidates to further select our desired topic words.

Noticing that different entities in the same domain usually have documents covering similar topics, we assume that a desired topic word should typically appear in multiple entities' documents, and therefore have a high frequency of occurrence among the topic candidates. So we calculate a cumulative average TF-IDF (CA-TF-IDF) score for each topic word in the candidates, which synthetically measures the word's document-level TF-IDF and entity-level occurrence frequency. Specifically, CA-TF-IDF sums the TF-IDF score of a topic word's each occurrence in the candidates, and divides it by the entity number in the domain. We filter out the topic candidates with low CA-TF-IDF scores and retain the rest to form the final retrieval indexes. The filtering thresholds are 2.3, 2.7, 6.9 and 7.3 for the domain of restaurant, hotel, taxi and train, respectively. While other domains are not involved in the document base. After the preprocessing, each document has one to three topic words extracted.

In the document retrieval process, we use the prefix and value of slot *ruk* in our proposed extended belief state to locate the document list of involved domain and entity. Then we use the topic of user utterance in our extended belief state to match the extracted topic of each document in the involved list, and select the best-matched one as the relevant reference. The topic matching is conducted by using the fuzzy string matching toolkit[6]. Noting that the relevant document is set to *none* if the slot *ruk* or the topic of user utterance is not available.

B Implementation Details

We use GloVe (Pennington et al., 2014) to initialize the embedding matrix, and set batch size, embedding size, hidden size and vocabulary size as 40, 50, 200 and 3000, respectively. We also set dropout rate as 0.35 and use greedy decoding to generate the belief state and response. Moreover, we use Adam optimizer (Kingma and Ba, 2014) with a learning rate of $7e^{-4}$, which is selected via grid search from $\{4e^{-4}, 5e^{-4}, 6e^{-4}, 7e^{-4}, 8e^{-4}, 9e^{-4}, 1e^{-3}\}$. We halve the learning rate when no improvement of overall performance (combined score) is observed on the development set in two consecutive epochs, and we stop the training when no improvement is observed in four consecutive epochs. The average training time is about 80 minutes per epoch, and the total number of training epoch is around 15. Model training is performed on NVIDIA TITAN-Xp GPU.

Model	Pretrained LM	Model Size
TRADE	none	10M
UniConv	none	16M
LABES-S2S	none	3.8M
HyKnow (Single)	none	4.1M
HyKnow (Multiple)	none	5.3M
TripPy	BERT	110M
SimpleTOD	GPT-2	81M

Table 8: Comparison of model size.

C Statistics of Modified MultiWOZ 2.1

There are totally 8449/1001/1004 dialogs[7] in the training, development and testing set of modified MultiWOZ 2.1, where 6501/836/847 dialogs have new turns inserted, respectively. After the modification, each dialog has 8.93 turns on average, which is longer than the original 6.85. The ontology of modified MultiWOZ 2.1 is the same as the original, with 32 slot types (excluding *ruk*) and 2426 corresponding slot values.

D Model Size Comparison

Table 8 shows the model size of our proposed HyKnow and some baseline models. We find that HyKnow has a comparable model size with the lightweight baseline models, which do not leverage pretrained language models (LM). But its model size is much smaller than that of TripPy and SimpleTOD, which use pretrained BERT and GPT-2, respectively. Therefore, HyKnow requires much less computational resources, compared to TripPy and SimpleTOD that use large-scale pretrained LM.

---

[6] `https://github.com/seatgeek/fuzzywuzzy`

[7] These are slightly more compared to the original MultiWOZ 2.1, because some of the original dialogs are modified twice with different turns inserted.

Target-Oriented Fine-tuning for Zero-Resource Named Entity Recognition

Ying Zhang[1]*, **Fandong Meng**[2], **Yufeng Chen**[1]†, **Jinan Xu**[1] **and Jie Zhou**[2]
[1]Beijing Key Lab of Traffic Data Analysis and Mining,
Beijing Jiaotong University, Beijing, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
`{zhying,chenyf,jaxu}@bjtu.edu.cn`
`{fandongmeng,withtomzhou}@tencent.com`

Abstract

Zero-resource named entity recognition (NER) severely suffers from data scarcity in a specific domain or language. Most studies on zero-resource NER transfer knowledge from various data by fine-tuning on different auxiliary tasks. However, how to properly select training data and fine-tuning tasks is still an open problem. In this paper, we tackle the problem by transferring knowledge from three aspects, i.e., domain, language and task, and strengthening connections among them. Specifically, we propose four practical guidelines to guide knowledge transfer and task fine-tuning. Based on these guidelines, we design a target-oriented fine-tuning (TOF) framework to exploit various data from three aspects in a unified training manner. Experimental results on six benchmarks show that our method yields consistent improvements over baselines in both cross-domain and cross-lingual scenarios. Particularly, we achieve new state-of-the-art performance on five benchmarks.

1 Introduction

Named Entity Recognition (NER) is one of the fundamental tasks in natural language processing. Recently, zero-resource NER draws more and more attention in recent studies (Täckström et al., 2012; Jia et al., 2019; Bari et al., 2020; Liu et al., 2020b; Wu et al., 2020a). This task describes that, in a specific domain or language, there is no labeled training data for NER. Therefore, zero-resource NER severely suffers from data scarcity.

As shown in Figure 1, the ideal training data for zero-resource NER is regarded as the *Targets*, which should satisfy two conditions at the same time: a) in the target domain or language, and b) annotated with NER labels. Thus it is intuitive to augment training data or transfer knowledge from three aspects, i.e., task, language, and domain. The aspect of domain/language can be divided into the source and the target, and the mainstream solution for zero-resource NER is transferring NER annotations from source domains/languages to target ones, e.g., from news to Twitter (Strauss et al., 2016) or from English to Spanish (Bari et al., 2020), where the former is referred as *cross-domain* and the latter as *cross-lingual*.

Figure 1: The example of the *Targets* and essential knowledge from three aspects, i.e., Task, Language, and Domain. The middle rectangle denotes an Spanish NER example in news domain, which is referred as the *Targets*. The rounded rectangle above the *Targets* denotes knowledge from different tasks. The bottom left one denotes essential knowledge in Spanish and English languages. The bottom right one denotes knowledge in News and Twitter domains.

Based on the mainstream approach, recent researches have conducted further exploration by fine-tuning the contextualized word embeddings on different data. Their results show that only exploiting source labeled data for NER is not enough, due to the discrepancy of domain/language between the source and the target. To alleviate this problem, some studies focus on utilizing a large amount of target unlabeled data to transfer domain- or

---

[*] Work was done when Ying Zhang was interning at Pattern Recognition Center, WeChat AI, Tencent Inc, China.

[†] Yufeng Chen is the corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1603–1615
August 1–6, 2021.

language-specific knowledge. AdaptaBERT (Han and Eisenstein, 2019) fine-tunes the masked language model (MLM) on unlabeled data in the target domain (e.g., social media). Both Pfeiffer et al. (2020) and Vidoni et al. (2020) add extra components to learn from unlabeled data in the target language (e.g., Spanish). Besides, Phang et al. (2020) apply non-NER labeled data in the source language (i.e., English) to transfer knowledge for cross-lingual NER, which suggests that annotations for non-NER tasks (e.g., MRC) are useful for NER task. However, they only exploit non-NER annotations in the source language and ignore that in the target languages (e.g., Spanish).

Though the aforementioned studies have improved the performance of zero-resource NER in cross-domain or cross-lingual scenarios, there are two main problems in these methods: a) they conduct knowledge transferring by only considering unlabeled target data and labeled source data, which is insufficient for knowledge transfer. Particularly, they ignore the fact that labeled target data from non-NER tasks is available. b) They fine-tune contextualized word embeddings on various auxiliary tasks in a pipeline manner, where each task is performed only once. We argue that the fine-tuning procedure can not capture enough knowledge from various data when trained only once. Besides, it lacks effective strategies to approach the *Targets* closer. Target at these issues, we suggest it is necessary to exploit more diverse data and design strategies more oriented to the *Targets*.

Therefore, we propose four practical guidelines on how to fully exploit available data to alleviate data scarcity. Concretely, we highlight the necessity of transferring knowledge from three aspects, i.e., task, language, and domain (Guideline-I). Then for domain/language, we pay attention to the gap between the source and target data (Guideline-II). For task, we focus on the gap between non-NER tasks and NER (Guideline-III). Finally, we emphasize the importance of knowledge fusion between the target domain/language and NER task (Guideline-IV). According to our proposed guidelines, we design a target-oriented fine-tuning (TOF) framework for zero-resource NER to approach the *Targets*. This framework applies three tasks (i.e., MLM, MRC, and NER) to capture the knowledge from above three aspects. It enhances the training with MRC task, pseudo data, and continual learning, respectively. To validate the effectiveness and superiority of our approaches, we conduct experiments on six popular benchmarks for zero-resource NER in cross-domain and cross-lingual scenarios.

Our contributions[1] are summarized as follows:

- We analyze the key factor of zero-resource NER and propose four practical guidelines to transfer knowledge from three aspects, i.e., Task, Language, and Domain, and strengthen connections among them.
- We design a target-oriented fine-tuning (TOF) framework based on our guidelines to exploit more diverse knowledge and approach the *Targets* closer.
- Experimental results verify the effectiveness of our method in both cross-domain and cross-lingual scenarios on six benchmarks. Particularly, our method achieves the state-of-the-art performance on five benchmarks.

2 Background

2.1 Task Definition

The goal of zero-resource NER task is to transfer NER knowledge from labeled source data to unlabeled target data. Therefore, we assume that there are three kinds of data available for training: a) NER labeled source data, b) unlabeled target data, and c) non-NER labeled target data (e.g., MRC).

2.2 Basic Framework

Our method is built on AdaptaBERT proposed by Han and Eisenstein (2019). This network is designed for unsupervised domain adaptation on sequence labeling tasks (e.g., NER). A two step fine-tuning approach is applied in AdaptaBERT, and in this section, we will describe it in detail.

Step-1: Domain Tuning. They fine-tune contextualized word embeddings by training a masked language model (MLM) to reconstruct randomly masked tokens. And this is performed on a dataset containing all available target domain data and an equal amount of unlabeled source domain data.

Step-2: Task Tuning. They fine-tune contextualized word embeddings continually and learn the prediction model for the sequence labeling task. Following (Devlin et al., 2018), they build a strong

---

[1] Code and data are publicly available at `https://github.com/Yarkona/TOF`

NER system by simply feeding the contextualized embeddings into a linear classification layer. The log probability can be computed by the log softmax,

$$\log p(y_t|\mathbf{w}_{1:T}) = \beta_{y_t} \cdot \mathbf{x}_t - \log \sum_{y \in \mathcal{Y}} \exp(\beta_y \cdot \mathbf{x}_t), \tag{1}$$

where contextualized word embedding $\mathbf{x}_t$ captures information from the entire sequence $\mathbf{w}_{1:T} = w_1, w_2, ..., w_T$, and β_y is a vector of weights for each tag $y \in Y = \{PER, ORG, LOC, MISC\}$. They train the model on labeled source domain data by minimizing the negative conditional log-likelihood of labeled data.

3 Our Approach

For zero-resource NER, we firstly analyze the problem of data scarcity. Then we propose four practical guidelines to guide knowledge transfer from different data, which is adapted to both cross-domain and cross-lingual scenarios. According to these guidelines, we design a target-oriented fine-tuning (TOF) framework for zero-resource NER.

3.1 Problem Analysis

The nature of zero-resource NER task is to perform NER with no labeled target domain/language data. And to deal with this task, it is intuitive to transfer essential knowledge from other available data. Concretely, when the data satisfies the two conditions at the same time: a) in the target domain/language (e.g., Twitter/Spanish), and b) annotated for the target task (i.e., NER), we consider it as our *Targets*. While the *Targets* is unavailable under the zero-resource setting, there is abundant data meeting either condition. Therefore, we transfer knowledge from three aspects, i.e., Domain, Language, and Task, as shown in Figure 1.

Domain. It contains knowledge from specific domains (e.g., Twitter). As shown in the bottom right rectangle of Figure 1, '@Garbriele_Corno:' is the special expression that only exists in tweets and '#' is used to highlight something.

Language. It refers to linguistic knowledge in various languages. For example, the word order of 'Como se conoce popularmente en Brasil al tenista' in Spanish is different from its English expression 'As the tennis player is popularly known in Brazil'. Besides, the expressions of 'Brazil' and 'tenista' in English vary from those in Spanish.

Task. It describes hand-crafted annotations for different tasks, which is expensive and difficult to obtain (e.g., NER and MRC). For example, NER labels *LOC* and *ORG* denote names of locations and organizations, respectively. For MRC task in Figure 1, 'W NJ' is annotated as the answer to question 'Who has the most to lose?' .

Furthermore, we divide domain/language aspect into the source and target. Particularly, NER is regarded as the target task for zero-resource NER.

3.2 Four Practical Guidelines

Based on our analysis, we propose four practical guidelines on how to fully exploit available knowledge to alleviate data scarcity.

Guideline-I: It is necessary to exploit available knowledge from domain, language, and task.

Guideline-II: Bridge the gap between source domains/languages and target domains/languages.

Guideline-III: Bridge the gap between annotations for non-NER tasks and NER task.

Guideline-IV: Fuse the knowledge of both the target domain/language and NER task.

3.3 Target-Oriented Fine-tuning Framework

As shown in Figure 2, we design a Target-Oriented Fine-tuning (TOF) framework for zero-resource NER. It contains two components: a) *Knowledge Transfer*, which displays how to transfer not only domain/language but also task knowledge from various data, and b) *Fine-tuning Process*, which demonstrates a flow diagram of the complete fine-tuning process. Both components are designed based on our proposed guidelines, and their relations are illustrated in Figure 2.

3.3.1 Knowledge Transfer

As the right part of Figure 2 shows, to transfer both domain/language and task knowledge for the *Targets*, we consider six kinds of corpora: a) unlabeled NER dataset $D_{t,no}$, b) unlabeled NER dataset $D_{s,no}$, c) labeled MRC dataset $D_{t,m}$, d) labeled MRC dataset $D_{s,m}$, e)unlabeled NER dataset $D_{t,no}$, and f) labeled NER dataset $D_{s,n}$, where {a), c), e)} is in the target domain and {b), d), f)} is in the source domain. Note that e) is the *Targets* without considering labels.

According to *Guideline-I*, since there is no available data that satisfies the *Targets*, it is necessary to transfer knowledge relevant to the *Targets* from other data as much as possible. Apart from source NER labeled data, we not only exploit unlabeled

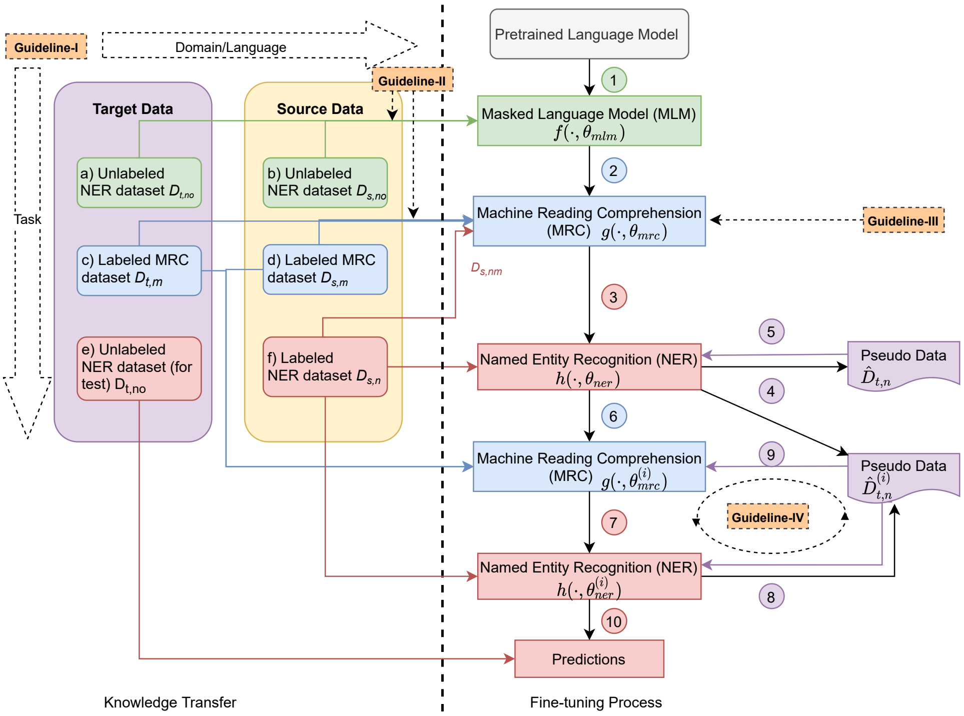

Figure 2: The overall architecture of our Target-Oriented Fine-tuning (TOF) framework. Orange rectangles denote our proposed four guidelines, and dotted lines with arrows denote how they guide *Knowledge Transfer* and *Fine-tuning Process*. Rectangles and solid lines in green, blue, and red color correspond to three fine-tuning tasks (i.e., MLM, MRC, NER), respectively. And the black solid lines with arrows denote the training steps.'Target Data' and 'Source Data' denote data in target and source domains/languages, respectively. i in the circle denotes *Step-i*.

target data, but also utilize non-NER labeled target data. Therefore, three kinds of data are considered as shown in Figure 2: for 'Target Data', a) unlabeled NER dataset $D_{t,no}$ and c) labeled MRC dataset $D_{t,m}$; for 'Source Data', and f) labeled NER dataset $D_{s,n}$.

According to *Guideline-II*, there is discrepancy between the source and target domain/language. To deal with the gap, it is essential to apply fine-tuning tasks on the mixture of the source and target data. Besides, an effective way to bridge the gap is transforming source data into the target format, e.g., translate the source language data into target language. Therefore, we collect b) unlabeled NER dataset $D_{s,no}$ and d) labeled MRC dataset $D_{s,m}$ in the 'Source Data'.

3.3.2 Fine-tuning Process

Based on AdaptaBERT, we novelly introduce a MRC task between *domain-tuning* and *task-tuning* process. Thus, our fine-tuning process contains three fine-tuning tasks as follows.

Masked Language Model (MLM). To adapt contextualized word embeddings to both the source and target data, we use MLM (Devlin et al., 2018). Based on *Guideline-II*, We train the model on a mixture of dataset $D_{t,no}$ and $D_{s,no}$. We use the same strategy with (Han and Eisenstein, 2019) to generate 10 random maskings for each instance.

Machine Reading Comprehension (MRC). Based on the *Guideline-III*, we add a span extraction MRC task, which has three advantages: a) MRC can enhance the ability of NER model on span extraction and help NER better capture semantic information of different entity types; b) MRC framework can be used to solve NER task (Li et al., 2020) and it becomes a bridge between NER and other tasks; and c) Recent work on framing other tasks as MRC (Wu et al., 2019; Liu et al., 2020a) provides an idea for transferring knowledge from different tasks with a unified framework. The MRC model is implemented by feeding the contextualized word embedding of each token x_t into two linear classification layers, respectively. The probability of each token being the start or the end index of a span can be computed as follows:

$$\begin{aligned} p_t^{start} &= softmax(W_{start} \cdot \mathbf{x}_t), \\ p_t^{end} &= softmax(W_{end} \cdot \mathbf{x}_t), \end{aligned} \tag{2}$$

where W_{start} and $W_{end} \in \mathbb{R}^{d_1 \times 2}$ are learnable parameters, and d_1 denotes the dimensions of contextualized word embedding. Finally, the model is trained by optimizing the Cross-Entropy loss over p_t^{start} and p_t^{end}. According to *Guideline-II*, we train the MRC model on all available MRC data $D_{t,m}$, $D_{s,m}$ and NER data $D_{s,n}$ that is transformed into MRC format $D_{s,nm}$ following (Li et al., 2020).

Named Entity Recognition (NER). To fine-tune contextualized word embeddings continually and learn the prediction model, we feed contextualized word embeddings into a linear classification layer and maximize the probability of each token with the ground-truth entity label. Concretely, given an input token sequence $\mathbf{x} = \{x_i\}_{i=1}^N$ with N words, we firstly feed it into the feature encoder f_θ to obtain contextualized word embeddings $\mathbf{h} = \{h_i\}_{i=1}^N$ for all tokens:

$$\mathbf{h} = f_\theta(\mathbf{x}), \tag{3}$$

where h_i is the feature vector corresponding to the i-th token x_i and f_θ is based on pre-trained language model, i.e., BERT (Devlin et al., 2018), where θ denotes model parameters. Then h_i is fed into a linear classification layer with the *softmax* function to predict the probability distribution of entity labels, which is formulated as follows:

$$p(\hat{y}|x_i) = softmax(Wh_i + b), \tag{4}$$

where $\hat{y} \in Y$ with Y being one-hot vectors corresponding to different entity labels, and $\{W, b\}$ denotes learnable parameters. The loss function is defined as the cross entropy between the predicted probability distribution of each entity label and the ground-truth one for each word. We train NER model on $D_{s,n}$ and predict labels on $D_{t,no}$.

3.3.3 Training

A novel training process is proposed to narrow the gap between the knowledge from available data and the *Targets*, which contains three processes, i.e., MRC enhancing, pseudo data enhancing, and continual learning enhancing.

MRC Enhancing. We fine-tune contextualized word embeddings by sequentially training the MLM $f(\cdot, \theta_{mlm})$, MRC $g(\cdot, \theta_{mrc})$, and NER $h(\cdot, \theta_{ner})$ at *Step-1∼3* in Figure 2.

Pseudo Data Enhancing. According to *Guideline-IV*, we use the trained NER model (*Step-3*) to generate pseudo labels on NER unlabeled target data $\hat{D}_{t,n}$ (*Step-4*) and then fine-tune NER model $h(\cdot, \theta_{ner}^{(0)})$ continually on generated pseudo-labeled target data at *Step-5*.

Algorithm 1 The training process of TOF.

```
Input: Dataset $D_{t,no}$, $D_{s,no}$, $D_{t,m}$, $D_{s,m}$, $D_{s,n}$, and $D_{s,nm}$; MLM $f(\cdot;\theta_{mlm})$; MRC $g(\cdot;\theta_{mrc})$; NER $h(\cdot;\theta_{ner})$; pre-trained BERT $\theta^{(0)}$; Number of pseudo-data iterations $T$.
Output: $h(\cdot, \theta_{ner}^{(T)})$.
1: Initialize $\theta_{mlm} = \theta^{(0)}$
2: Fine-tune $f(\cdot, \theta_{mlm})$ on $\{D_{t,no}, D_{s,no}\}$
3: Initialize $\theta_{mrc} = \theta_{mlm}$
4: Fine-tune $g(\cdot, \theta_{mrc})$ on$\{D_{t,m}, D_{s,m}, D_{s,nm}\}$
5: Initialize $\theta_{ner} = \theta_{mrc}$
6: Fine-tune $h(\cdot, \theta_{ner})$ on $\{D_{s,n}\}$
7: Gen pseudo-NER $\hat{D}_{t,n} \leftarrow h(\cdot, \theta_{ner})$on $D_{t,no}$
8: Initialize $\theta_{ner}^{(0)} = \theta_{ner}$
9: Fine-tune $h(\cdot, \theta_{ner}^{(0)})$ on $\{\hat{D}_{t,n}\}$
10: Gen pseudo-NER $D_{t,n}^{(0)} \leftarrow h(\cdot, \theta_{ner}^{(0)})$ on $\hat{D}_{t,no}$
11: Gen pseudo-MRC $\hat{D}_{t,m}^{(0)} \leftarrow \hat{D}_{t,n}^{(0)}$
12: for $i = 1 \rightarrow T$ do
13:     Initialize $\theta_{mrc}^{(i)} = \theta_{ner}^{(i-1)}$
14:     Fine-tune $g(\cdot, \theta_{mrc}^{(i)})$ on $\{D_{t,m}, \hat{D}_{t,m}^{(i-1)}\}$
15:     Initialize $\theta_{ner}^{(i)} = \theta_{mrc}^{(i)}$
16:     Fine-tune $h(\cdot, \theta_{ner}^{(i)})$ on $\{\hat{D}_{t,n}^{(i-1)}\}$
17:     Gen pseudo-NER $\hat{D}_{t,n}^{(i)} \leftarrow h(\cdot, \theta_{ner}^{(i)})$ on $D_{t,no}$
18:     Gen pseudo-MRC $\hat{D}_{t,m}^{(i)} \leftarrow \hat{D}_{t,n}^{(i)}$
19: end for
20: Predict $h(\cdot, \theta_{ner}^{(T)})$ on $D_{t,no}$
21: return $h(\cdot, \theta_{ner}^{(T)})$
```

Continual Learning Enhancing. We design a continual learning strategy to make full use of pseudo data and imitate the training procedure on the *Targets*. We continually perform fine-tuning between MRC and NER with considering pseudo data (*Step-6∼7*), based on the following three considerations: 1) pseudo-labeled target NER data can be refined by the fine-tuned NER model after each iterations, 2) pseudo data is transformed into MRC format, which directly introduces entity type knowledge in target data through definition of MRC questions, and 3) pseudo data participates in both MRC and NER training, which can enhance knowledge connections between two tasks. At *Step-8∼9*, we refine pseudo data with newly fine-tuned NER model and take it as training data. After T times iteration, we conduct predictions on unlabeled target data with NER model $h(\cdot, \theta_{ner}^{(T)})$ (*Step-10*).

The training procedure is summarized as 1. $D_{t,no}$ and $D_{s,no}$ denote unlabeled NER data in the target and source domain/language, respectively. $D_{t,m}$ and $D_{s,m}$ denote labeled MRC data in the target and source domain/language, respectively. $D_{t,n}$ and $D_{s,n}$ denote labeled NER data in the target and source domain/language, respectively. Particularly, $D_{t,n}$ is the pseudo-labeled NER data in the target domain/language generated by NER model. And we transformed it into the MRC format, as $D_{s,nm}$. $f(\cdot;\theta_{mlm})$, $g(\cdot;\theta_{mrc})$ and $h(\cdot;\theta_{ner})$ denote the model of MLM, MRC, and NER, respectively. Note that 'Gen' in Algorithm 1 denotes the generalize operation.

4 Experiments

4.1 Data Preparation

We take CoNLL03 for English (en) in the news domain as the source dataset for both cross-lingual and cross-domain tasks.

Cross-Lingual. We consider three NER datasets in target languages: CoNLL03 for German (*de*) (Tjong Kim Sang and De Meulder, 2003), CoNLL02 for Dutch (*nl*) and Spanish (*es*) (Tjong Kim Sang, 2002). All datasets are labeled with 4 entity types: PER, ORG, LOC, MISC. Each of them is split into training, validation and test sets following (Wu et al., 2020b). We use three MRC datasets in target languages: MLQA (*es*) (Lewis et al., 2019), XQuAD (*de*) (Artetxe et al., 2019), and SQuAD (*en*) (Rajpurkar et al., 2016).

Cross-Domain. We use three English datasets in target domains: CBS SciTech News dataset (Jia et al., 2019), short as CBS, in the science and technology news domain, Twitter NER (Zhang et al., 2018b) and WNUT16 (Strauss et al., 2016) in the social media domain. We use two English MRC datasets from news and twitter domains respectively: NewsQA (Trischler et al., 2016) and TweetQA (Xiong et al., 2019). The statistics of datasets are shown in 5 in Appendix A.

4.2 Data Preprocessing

NER datasets are processed in the 'BIO' scheme with four entity types, i.e., PER, LOC, ORG, and MISC except for WNUT16. We perform entity span detection task on WNUT16. Since there are ten entity types annotated in WNUT16, it is different from annotations in source domain/language. For MRC datasets, we transform all of them into a unified format following (Li et al., 2020) for MRC training. Besides, following (Li et al., 2020), we map the labeled NER datasets to labeled MRC dataset. Concretely, we use the description of each entity for annotators as the query, and each sentence as context. The corresponding answers for each query are entity spans with the same entity type in the sentence. We delete all entity labels on the target data and only use the unlabeled data. We use training and validation sets from the source for training and evaluation, and do predictions on test sets from different target domains/languages.

4.3 Implementation Details

We use BERT-base and multilingual BERT (Devlin et al., 2018) to initialize contextualized word embeddings in cross-domain and cross-lingual scenarios, respectively. We empirically follow the hyperparameter settings of (Han and Eisenstein, 2019) and (Li et al., 2020) except for the learning rate and batch size. Due to the discrepancy between various datasets, we choose the learning rate for Adam (Zhang et al., 2018a) optimizer according to the best performance of checkpoints on the validation set. And the batch size is set to 32, 16 and 64 for MLM, MRC and NER, respectively. More hyperparameters for training procedure are listed in Appendix B.

4.4 Systems

We evaluate following systems by entity-level F1 scores (Sang and De Meulder, 2003). Moreover, we conduct each experiment 5 times and report the mean F1-score.

BERT-ML. Moon et al. (2019) apply the multilingual BERT to cross-lingual NER.

TSL. Wu et al. (2020a) propose a teacher-student learning method for cross-lingual NER.

UniTrans. Wu et al. (2020b) unify data transfer and model transfer for cross-lingual NER.

mCell LSTM. Jia and Zhang (2020) design a multi-cell compositional LSTM for cross-domain NER.

COFEE-MRC. Xue et al. (2020) inject coarse-to-fine automatically mined entity knowledge in a pre-trained language model for cross-domain NER.

AdaptaBERT. Han and Eisenstein (2019) perform *domain-tuning* and *task-tuning* as described in Section 2.2. We take the AdaptaBERT as our baseline in the cross-domain scenario.

AdaptaBERT + translation. Another baseline is set for the cross-lingual scenario. We apply trans-

Systems	cross-lingual				cross-domain			
	es	nl	de	avg	CBS	Twitter	WNUT16	avg
BERT-ML(Moon et al., 2019)†	76.53	**83.35**	72.44	77.44	-	-	-	-
TSL (Wu et al., 2020a)†	78.00	81.33	75.33	78.22	-	-	-	-
UniTrans(Wu et al., 2020b)†	79.31	82.90	74.82	79.01	-	-	-	-
+ ensemble	79.29	83.07	75.55	79.29	-	-	-	-
mCell LSTM(Jia and Zhang, 2020)†	-	-	-	-	75.19	-	-	-
COFEE-MRC(Xue et al., 2020)†	-	-	-	-	-	54.56	-	-
AdaptaBERT(Han and Eisenstein, 2019)*	75.30 (± 0.30)	78.52 (± 0.25)	70.90 (± 0.54)	75.20	75.30 (± 0.37)	65.61 (± 0.46)	63.03‡ (± 0.23)	67.98
+ translation	76.18 (± 0.13)	80.30 (± 0.52)	72.47 (± 0.75)	76.32	-	-	-	-
TOF(*ours*)	**80.35** (± 0.29)	82.79 (± 0.17)	**76.57** (± 0.16)	**79.90**	**76.41** (± 0.5)	**67.94** (± 0.09)	**67.86** (± 0.27)	**70.74**
w/o continual learning	79.44 (± 0.08)	81.64 (± 0.17)	76.39 ± 0.17	79.16	75.95 (± 0.38)	67.13 (± 0.05)	67.70 (± 0.08)	70.26
w/o pseudo data & w/o continual learning	78.32 (± 1.11)	80.56 (± 0.35)	73.61 ± 0.61	77.50	75.34 (± 0.51)	66.18 (± 0.17)	66.45 (± 0.39)	69.32

Table 1: Results of our method and previous state-of-the-art methods for zero-resource NER in cross-lingual and cross-domain. 'avg' denotes the average of F1 scores (%) on three benchmarks. '†' denotes original results reported in their original papers. '*' denotes results re-implemented by us. Note that '‡' denotes our re-implemented result on WNUT16 and the previous state-of-the-art result on it is 62.8 reported by Han and Eisenstein (2019).

lations of source data[2] to both *domain-tuning* and *task-tuning* of AdaptaBERT.

TOF. Our method is built on two baselines for the cross-lingual and cross-domain scenario, respectively. 'w/o continual learning' denotes the framework without *Step-6∼9*. 'w/o pseudo data & w/o continual learning' denotes the framework only performs MRC enhancing at *Step-1∼3* of Figure 2.

5 Results and Analysis

5.1 Overall Performance

Table 1 lists main results of our method in contrast with previous state-of-the-art methods in both cross-lingual and cross-domain scenarios.

Cross-Lingual. Our *baseline* on three cross-lingual benchmarks is implemented by training AdaptaBERT with additional translations of source language data, referred as 'AdaptaBERT+translations' in Table 1. Our method achieves significant improvements over *baseline* of F1-scores 4.17, 2.49, and 4.1 for *es*, *nl*, and *de*, respectively. Compared to previous methods, our TOF framework achieves the new state-of-the-art results on two benchmarks *es* and *de*. Besides, Table 1 shows the results of our TOF after removing 'pseudo data' and 'continual learning', respectively, which demonstrates the effectiveness of these two enhancing strategies. The improvement of our TOF on *nl* (2.49 ↑) is not as good as other two languages (*es*:4.17 ↑ and *de*: 4.1 ↑), which results from the scarcity of MRC data in *nl*. The results well demonstrate the effectiveness of our proposed framework, which benefit from our four guidelines.

Cross-Domain. We regard re-implemented results of AdaptaBERT as our *baseline*, since it not only achieves the state-of-the-art performance on WNUT16, but also outperforms the previous state-of-the-art methods on both CBS and Twitter. Our framework yields obvious improvements over the *baseline* (CBS: 1.11 ↑, Twitter: 2.33 ↑ and WNUT16: 4.83 ↑) and achieves new state-of-the-art results on three datasets. In conclusion, all these results verify the effectiveness and generalizability of our TOF in cross-domain setting.

5.2 Ablation Study

We conduct ablation studies to explore how MRC datasets make difference at step 1∼3 in Figure 2. Table 2 highlights the impact of different MRC data in both cross-lingual and cross-domain scenarios.

In the cross-lingual scenario, we consider five kinds of MRC data: 1) 'w/o target MRC data' denoting training without MRC data in the target language; 2) 'w/o source MRC data' denoting training without MRC data in English; 3) 'w/o source MRC data (trans)' denoting without translating the source MRC data into the target language; 4) 'w/o

---

[2] We translate the source language data into the target language following (Wu et al., 2020b) using MUSE (Conneau et al., 2017).

#	Methods	cross-lingual				cross-domain			
		es	nl	de	avg	CBS	Twitter	WNUT16	avg
0	TOF$_{MRC-enhancing}$	78.32	80.30	73.61	77.41	75.34	66.18	66.45	69.32
1	w/o target MRC data	0.34 ↓	-	0.72 ↓	-	0.44 ↓	1.99 ↓	0.39 ↓	0.93 ↓
2	w/o source MRC data	1.82 ↓	1.26 ↓	1.96 ↓	1.68 ↓	0	0.25 ↓	0.78 ↓	0.54 ↓
3	w/o source MRC data (trans)	2.57 ↓	0.05 ↓	1.65 ↓	1.42 ↓	-	-	-	-
4	w/o NER-MRC data	3.69 ↓	0.83 ↓	1.48 ↓	2.00 ↓	0.88 ↓	0.65 ↓	0.26 ↓	0.59 ↓
5	w/o NER-MRC data (trans)	3.02 ↓	1.01 ↓	1.30 ↓	1.77 ↓	-	-	-	-

Table 2: Ablation study for TOF$_{MRC-enhancing}$, which only performs step 1∼3 in Figure 2. Row 1∼5 list the performance changes compared with Row 0. '↓' denotes the drop of performance.

	cross-lingual				cross-domain			
	es	nl	de	avg	CBS	Twitter	WNUT16	avg
MLM → MRC → NER	78.32	80.30	73.61	77.41	75.34	66.18	66.45	69.32
MRC → MLM → NER	73.17	81.06	73.06	75.76	74.90	65.87	66.13	68.97

Table 3: Results of our TOF framework with different fine-tuning orders.

NER-MRC data' denoting without transforming the NER data into MRC format; and 5) 'w/o NER-MRC data (trans)' denoting without translating the NER-MRC data into the target language.

Results demonstrate that removing any data generally causes a performance drop. Therefore, we draw more in-depth observations as follows. For *es*, 'NER-MRC data' brings the greatest drop of the performance (Row 4). For **nl** and *de*, 'source MRC data' has the greatest impact (Row 2). Besides, 'source MRC data' affects the performance more than 'target MRC data' (Row 2 vs. Row 1). We think it is because 'source MRC data' is twice as much as the target one.

In the cross-domain scenario, since all of three target datasets are in English but in different domains, we do not consider the translated data (Row 3 and 5) in Table 2. Therefore, we conduct ablation studies on three kinds of data: 1) 'w/o target MRC data' denoting training without the target domain MRC data; 2) 'w/o source MRC data' denoting without the source domain MRC data; and 4) 'w/o NER-MRC' data denoting without transforming the NER data into MRC format.

According to the average results in Table 2, we observe that 'target MRC data', 'NER-MRC data', and 'source MRC data' are in descending order of impact. It is intuitive that on Twitter, as shown in Table 2 (Row 1 vs. Row 2 and Row 4), 'target MRC data' has the greatest impact on the performance, when the amount of the target MRC data is greater than or equal to that of 'source MRC data' and 'NER-MRC data'. However, CBS is affected most by 'NER-MRC data' (Row 4), since its target MRC data are collected from news domain, not science and technology news. For WNUT16, both 'target MRC data' and 'source MRC data' bring more drops than 'NER-MRC data' (Row 1 and Row 2 vs. Row 4). We conjecture that since WNUT16 is an entity span detection task rather than standard NER, it is affected more by the golden MRC data than NER-MRC data.

5.3 Impact of Task Order

We explore the impact of two different sequences for MRC-enhancing, i.e, 'MLM → MRC → NER' and 'MRC → MLM → NER' as shown in Table 3. The results demonstrate that the former outperforms the latter in both cross-lingual and cross-domain scenarios. We conjecture that MLM can capture knowledge of data itself, e.g., domain-specific information and linguistic characters, and MRC captures task-specific information with annotations. Besides, MRC is more relevant to NER than MLM according to task relevance. Therefore, MRC is appropriate to be an intermediate task.

5.4 Comparison with SpanBERT

We replace the pre-trained language model BERT with SpanBERT (Joshi et al., 2020) in the Adapt-aBERT and MRC-enhancing of our TOF to compare the span-enhancing method with ours. The results are shown in Table 4. 1) SpanBERT is

#	Methods	CBS	Twitter	WNUT16	avg
1	AdaptaBERT	75.30	65.61	63.03	67.98
2	+MRC-enhancing	75.34	66.18	66.45	69.32
3	SpanBERT	75.37	67.11	64.17	68.88
4	+MRC-enhancing	75.48	67.46	67.80	70.25

Table 4: AdaptaBERT vs SpanBERT (Joshi et al., 2020) in cross-domain NER.

superior to BERT-base for NER task (Row 3 vs. Row 1). Different from BERT with masking different tokens for each instance, SpanBERT masks a span with several adjacent tokens, which is more related to NER task. 2) 'SpanBERT' underperforms 'AdaptaBERT+MRC-enhancing' on CBS and WNUT16 (Row 2 vs. Row 3), which suggests that although SpanBERT is trained on a large amount of corpus, it is not appropriate for some specific domains. Our MRC-enhancing method uses limited MRC data but achieves more improvements, which shows that MLM can not capture enough task-specific information and it is necessary to introduce other NER-related tasks. 3) Our MRC-enhancing method can make further improvements based on SpanBERT (Row 4 vs. Row 3).

6 Related Work

Zero-resource NER. Some studies (Jia and Zhang, 2020; Pfeiffer et al., 2020; Vidoni et al., 2020) focus on improving architectures of existing models, which add new components into networks to capture specific knowledge, i.e., entity types, language and task characteristics. Different from these methods, our approach only modifies the training procedure without changing model structures. Other studies introduce different auxiliary tasks to alleviate data scarcity (Han and Eisenstein, 2019; Xue et al., 2020; Phang et al., 2020). They are usually based on multi-task learning or two-phrase fine-tuning. Multi-task learning requires balance between the target task and auxiliary tasks, which needs carefully designed objectives. Although two-phrase fine-tuning is effective, it is still inadequate for available data and depends on valid data selection. Our work differs in that we not only propose four practical guidelines to guide data selection and task fine-tuning, but also design a task-oriented fine-tuning framework to exploit more diverse data and target-oriented training strategies.

Data Augmentation. Our approach is inspired by some studies on text classification. Gururangan et al. (2020) utilize unlabeled data in different domains and tasks. Ben-David et al. (2020) exploit unlabeled corpora from multiple domains. Unlike these methods, we focus on target domain/language data with annotations for other tasks, which not only transfers domain/language knowledge, but also utilizes available annotations for other tasks.

MRC for Different Tasks. Although most researches on NER focus on the sequence labeling framework (Huang et al., 2015; Ma and Hovy, 2016; Akbik et al., 2018; Liu et al., 2019), our work is also inspired by formatting other tasks as MRC, such as NER (Li et al., 2020), co-reference resolution (Wu et al., 2019), and event extraction (Liu et al., 2020a). These studies show the superiority and scalability of MRC framework and provide a reference for our work. Different from (Li et al., 2020) using MRC to build a new solution architecture for NER, we exploit MRC to improve the training procedure of NER that is based on sequence labeling. Besides, we perform continual learning between MRC and NER to enhance the impact of MRC on NER.

7 Conclusion and Future Work

In this paper, we analyze the problem of data scarcity in zero-resource NER. To alleviate this issue, we propose four practical guidelines on transferring knowledge from three aspects, i.e., domain, language, and task, and strengthening connections between the source and target data. Based on these guidelines, we design a task-oriented fine-tuning framework to enhance the training procedure with various strategies. Our approach yields significant improvements on six benchmarks and achieves the state-of-the-art on five benchmarks. In the future, we will extend our framework on different target tasks and more task-specific enhancing strategies.

Acknowledgments

The research work descried in this paper has been supported by the National Nature Science Foundation of China (No. 61976016, 61976015, and 61876198) and the National Key R&D Program of China (2020AAA0108001). The authors also would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. On the cross-lingual transferability of monolingual representations. *CoRR*, abs/1910.11856.

M Saiful Bari, Shafiq Joty, and Prathyusha Jwalapuram. 2020. Zero-resource cross-lingual named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7415–7423.

Eyal Ben-David, Carmel Rabinovitz, and Roi Reichart. 2020. PERL: Pivot-based domain adaptation for pre-trained deep contextualized embedding models. *Transactions of the Association for Computational Linguistics*, 8:504–521.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4238–4248, Hong Kong, China. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain ner using cross-domain language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2464–2474.

Chen Jia and Yue Zhang. 2020. Multi-cell compositional LSTM for NER domain adaptation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5906–5917, Online. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Patrick S. H. Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. MLQA: evaluating cross-lingual extractive question answering. *CoRR*, abs/1910.07475.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020a. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Zihan Liu, Genta Indra Winata, and Pascale Fung. 2020b. Zero-resource cross-domain named entity recognition.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Taesun Moon, Parul Awasthy, Jian Ni, and Radu Florian. 2019. Towards lingua franca named entity recognition with BERT. *CoRR*, abs/1912.01389.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jason Phang, Phu Mon Htut, Yada Pruksachatkun, Haokun Liu, Clara Vania, Katharina Kann, Iacer Calixto, and Samuel R Bowman. 2020. English intermediate-task training improves zero-shot cross-lingual transfer too. *arXiv preprint arXiv:2005.13013*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Benjamin Strauss, Bethany Toma, Alan Ritter, Marie-Catherine De Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 138–144.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487, Montréal, Canada. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830.

Marko Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer.

Qianhui Wu, Zijia Lin, Börje Karlsson, Jian-Guang Lou, and Biqing Huang. 2020a. Single-/multi-source cross-lingual NER via teacher-student learning on unlabeled data in target language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6505–6514, Online. Association for Computational Linguistics.

Qianhui Wu, Zijia Lin, Börje F Karlsson, Biqing Huang, and Jian-Guang Lou. 2020b. Unitrans: Unifying model transfer and data transfer for cross-lingual named entity recognition with unlabeled data. *arXiv preprint arXiv:2007.07683*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2019. Coreference resolution as query-based span prediction. *arXiv preprint arXiv:1911.01746*.

Wenhan Xiong, Jiawei Wu, Hong Wang, Vivek Kulkarni, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. TWEETQA: A social media focused question answering dataset. *CoRR*, abs/1907.06292.

Mengge Xue, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-fine pre-training for named entity recognition. *arXiv preprint arXiv:2010.08210*.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. 2018a. Three mechanisms of weight decay regularization. *CoRR*, abs/1810.12281.

Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. 2018b. Adaptive co-attention network for named entity recognition in tweets. In *AAAI*, pages 5674–5681.

A Statistics

The statistics of all datasets are listed in Table 5. We regard CoNLL03 in English as the source NER data in both cross-lingual and cross-domain scenarios. For target NER datasets, we consider cross-lingual and cross-domain scenarios, respectively.

In the cross-lingual scenario, CoNLL03 in German, CoNLL02 in Spanish, and CoNLL02 in Dutch denote the benchmark datasets in the target languages, i.e., German (*de*), Spanish (*es*), and Dutch (*nl*), respectively. In terms of MRC datasets, we apply MLQA in Spanish and XQuAD in German as labeled MRC datasets in the target languages, i.e., on *es* and *de*. Note that we use the initial validation and test splits in MLQA and XQuAD as the training and validation sets in our work. Since it is difficult to obtain labeled MRC datasets for *nl*, we consider the MRC data in the source language, i.e., English (*en*).

In the cross-domain scenario, CBS SciTech News NER datasets, short as CBS, in science and technology news domain, Tiwtter NER dataset in twitter domainm, and the shared task on entity span detection for WNUT2016 in twitter domain are considered as the target domain NER datasets. All of these three cross-domain benchmarks are in English. We use NewsQA in news domains for MRC fine-tuning on CBS, due to lack of available MRC data in science and technology news domain. TweetQA is applied to both Twitter and WNUT16 NER as the MRC data in the target domain.

B Hyperparameters

In the training procedure, we fine-tune each MLM model for 3 epochs, MRC for 6 epochs, and NER

Benchmark	Task	Language	Domain	Training	Validation	Test
CoNLL03	NER	English(en)	News	14987	3466	3684
Cross-Lingual						
CoNLL03	NER	German(en)	News	12705	3068	3160
CoNLL02	NER	Spanish(es)	News	8323	1915	1517
CoNLL02	NER	Dutch(nl)	News	15806	2895	5195
MLQA	MRC	Spanish(es)	Multi-domain	5254	500	-
XQuAD	MRC	German(de)	Multi-domain	1190	428	-
SQuAD	MRC	English(en)	Multi-domain	10000	1000	-
Cross-Domain						
CBS SciTech	NER	English(en)	Science-technoloy news	-	-	2000
TwitterNER	NER	English(en)	Social-media	4000	1000	3256
WNUT16	ESD*	English(en)	Social-media	2394	1000	3856
NewsQA	MRC	English(en)	News	92550	5167	5127
TweetQA	MRC	English(en)	Social-media	10692	1086	1979

Table 5: Dataset statistics. '*' denotes the entity span detection task.

Methods		cross-lingual			cross-domain		
		es	nl	de	CBS	Twitter	WNUT16
MRC-enhancing	MRC	2e-6	1e-6	1e-6	2e-5	2e-5	2e-5
MRC-enhancing	NER	5e-5	2e-5	2e-5	5e-5	2e-5	5e-5
Pseudo data	NER	2e-5	1e-5	1e-5	8e-5	1e-6	1e-6
Continual learning	MRC	5e-5	2e-6	2e-5	8e-6	3e-5	1e-6
Continual learning	NER	5e-5	5e-6	1e-6	3e-5	5e-6	2e-6

Table 6: Learning rate of MRC and NER model on different datasets.

for 6 epochs. Besides, we perform continual learning for one iteration and achieve the best performance. To re-implement our results easily, we set the seed to a fixed value as 2019 following (Han and Eisenstein, 2019). The learning rate of MLM on each dataset is set to 5e-5. The learning rate of MRC and NER model on different datasets is listed in Table 6. To find the proper learning rate, we perform the hyperparameter search on a set of learning rete values, i.e., 1e-6, 2e-6, 5e-6, 8e-6, 1e-5, 2e-5, 3e-5. 5e-5, 8e-5. And we choose hyperparameter values according to the best validation performance. We train our model on one NVIDIA Tesla P40 (24GB). The average runtime of our TOF framework on different datasets varies from 6 hours to 2 days, due to the data size of different datasets. Other hyperparameters are set following (Han and Eisenstein, 2019) and (Li et al., 2020).

BERT-Defense: A Probabilistic Model Based on BERT to Combat Cognitively Inspired Orthographic Adversarial Attacks

Yannik Keller[†], **Jan Mackensen**[†], **Steffen Eger**[‡]
[†] Center of Cognitive Science
[‡] Natural Language Learning Group
Technische Universität Darmstadt
{yannik.keller, jan.mackensen}@stud.tu-darmstadt.de
eger@aiphes.tu-darmstadt.de

Abstract

Adversarial attacks expose important blind spots of deep learning systems. While word- and sentence-level attack scenarios mostly deal with finding semantic paraphrases of the input that fool NLP models, character-level attacks typically insert typos into the input stream. It is commonly thought that these are easier to defend via spelling correction modules. In this work, we show that both a standard spellchecker and the approach of Pruthi et al. (2019), which trains to defend against insertions, deletions and swaps, perform poorly on the character-level benchmark recently proposed in Eger and Benz (2020) which includes more challenging attacks such as visual and phonetic perturbations and missing word segmentations. In contrast, we show that an untrained iterative approach which combines context-independent character-level information with context-dependent information from BERT's masked language modeling can perform on par with human crowd-workers from Amazon Mechanical Turk (AMT) supervised via 3-shot learning.

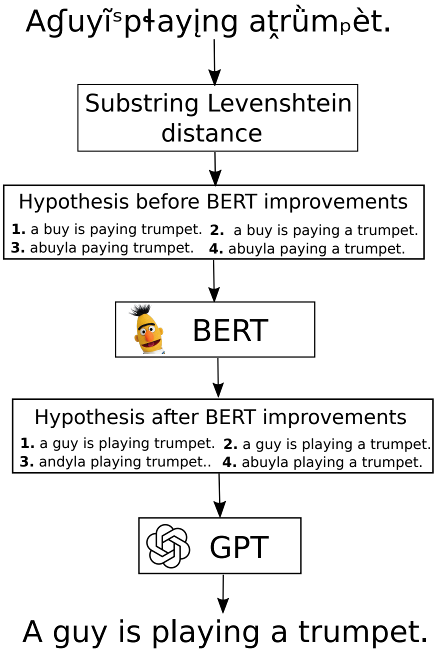

Figure 1: A high-level overview of the processing of an example sentence in our adversarial-defense pipeline. The sentences shown for the hypothesis have been created by choosing the maximum of their associated probability distributions over words.

1 Introduction

Adversarial attacks to machine learning systems are malicious modifications of their inputs designed to fool machines into misclassification but not humans (Goodfellow et al., 2015). One of their goals is to expose blind-spots of deep learning models, which can then be shielded against. In the NLP community, typically two different kinds of attack scenarios are considered. "High-level" attacks paraphrase (semantically or syntactically) the input sentence (Iyyer et al., 2018; Alzantot et al., 2018; Jin et al., 2020) so that the classification label does not change, but the model changes its decision. Often, this is framed as a search problem where the attacker has at least access to model predictions (Zang et al., 2020). "Low-level" attackers operate on the level of characters and may consist of adversarial typos (Belinkov and Bisk, 2018; Ebrahimi et al., 2018a; Pruthi et al., 2019; Jones et al., 2020) or replacement of characters with similarly looking ones (Eger et al., 2019; Li et al., 2020a). Such attacks may also be successful when the attacker operates in a blind mode, without having access to model predictions, and they are arguably more realistic, e.g., in social media. However, Pruthi

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1616–1629
August 1–6, 2021. ©2021 Association for Computational Linguistics

et al. (2019) showed that orthographic attacks can be addressed by placing a spelling correction module in front of a downstream classifier, which may be considered a natural solution to the problem.[1]

In this work, we apply their approach to the recently proposed benchmark *Zéroe* of Eger and Benz (2020), illustrated in Table 1, which provides an array of cognitively motivated orthographic attacks, including missing word segmentation, phonetic and visual attacks. We show that the spelling correction module of Pruthi et al. (2019), which has been trained on simple typo attacks such as character swaps and character deletions, fails to generalize to this benchmark. This motivates us to propose a novel technique to addressing various forms of orthographic adversaries that does not require to train on the low-level attacks: first, we obtain probability distributions over likely true underlying words from a dictionary using a context-independent extension of the Levenshtein distance; then we use the masked language modeling objective of BERT, which gives likelihoods over word substitutions in context, to refine the obtained probabilities. We iteratively repeat this process to improve the word context from which to predict clean words. Finally, we apply a source text independent language model to produce fluent output text.

Our contributions: **(i)** We empirically show that this approach performs much better than the trained model of Pruthi et al. (2019) on the Zéroe benchmark. Furthermore, **(ii)** we also evaluate human robustness on Zéroe and **(iii)** demonstrate that our iterative approach, which we call BERT-Defense, sometimes even outperforms human crowd-workers trained via 3-shot learning.

2 Related work

Zeng et al. (2020) classify adversarial attack scenarios in terms of the *accessibility* of the victim model to the attacker:[2] *white-box* attackers (Ebrahimi et al., 2018b) have full access to the victim model including its gradient to construct adversarial examples. In contrast, *black-box* attackers have only limited knowledge of the victim models: score- (Alzantot et al., 2018; Jin et al., 2020) and decision-based attackers (Ribeiro et al., 2018) require access to the victim models' prediction scores (classification probabilities) and final decisions (predicted class), respectively. A score-based black-box attacker of particular interest in our context is BERT-ATTACK (Li et al., 2020b). BERT-ATTACK uses the masked language model (MLM) of BERT to replace words with other words that fit the context. BERT-ATTACK is related to our approach because it uses BERT's MLM in an attack-mode while we use it in defense-mode. Further, in our terminology, BERT-ATTACK is a high-level attacker, while we combine BERT with an edit distance based approach to restore low-level adversarial attacks. *Blind* attackers make fewest assumptions and have no knowledge of the victim models at all. Arguably, they are most realistic, e.g., in the context of online discussion forums and other forms of social media where users may not know which model is employed (e.g.) to censor toxic comments and users may also not have (large-scale) direct access to model predictions.

Attacker	**Sentence**
inner-shuffle	A man is drnviig a car.
full-shuffle	A amn is ginvdir a acr.
disemvowel	A mn is drvng a cr.
intrude	A ma#n i*s driving a caˆr.
keyboard-typo	A mwn is dricing a caf.
natural-typo	A wan his driving as car.
truncate	A man is drivin a car.
segmentation	Aman isdriving a car.
phonetic	Ae man izz dreyvinn a cahar.
visual	A māᴎ ǐꜱ ḋʀǹʋɪɳɢ ɐ ƈā̄ŕ

Table 1: Examples for the adversarial attacks from the Zéroe benchmark. The phonetic and visual examples show our modified implementations (see appendix A.2).

In terms of blind attackers, Eger et al. (2019) design the visual perturber VIPER which replaces characters in the input stream with visual nearest neighbors, an operation to which humans are seemingly very robust.[3] Eger and Benz (2020) propose a canon of 10 cognitively inspired orthographic character-level blind attackers. We use this benchmark, which is illustrated in Table 1, in our application scenario. While Eger et al. (2019) and Eger and Benz (2020) are only moderately successful in

---

[1] One could argue that such a pipeline solution is not entirely satisfactory from a more theoretical perspective, and that downstream classifiers should be innately robust to attacks in the same way as humans.

[2] Another recent survey of adversarial attacks in NLP is provided by Roth et al. (2021).

[3] Basing text processing on visual properties was also recently explored in Wang et al. (2020) and Salesky et al. (2021).

defending against their orthographic attacks with *adversarial learning* (Goodfellow et al., 2015) (i.e., including perturbed instances at train time), Pruthi et al. (2019) show that placing a word recognition (correction) module in front of a downstream classifier may be much more effective. They use a correction model trained to recognize words corrupted by random adds, drops, swaps, and keyboard mistakes. Zhou et al. (2019) also train on the adversarial attacks (insertion, deletion, swap as well as word-level) against which they defend. In contrast, we show that an *untrained* attack-agnostic iterative model based on BERT may perform competitively even with humans (crowd-workers) and that this correction module may further be improved by leveraging attack-specific knowledge. Jones et al. (2020) place an encoding module—which should map orthographically similar words to the same (discrete) 'encoding'—before the downstream classifier to improve robustness against adversarial typos. However, in contrast to Pruthi et al. (2019) and BERT-Defense, their model does not restore the attacked sentence to its original form so that it is less desirable in situations where knowing the underlying surface form may be relevant (e.g., for human introspection or in tasks such as spelling normalization).

In contemporaneous work, Hu et al. (2021) use BERT for masked language modeling together with an edit distance to correct a misspelled word in a sentence. They assume a single misspelled word that they correct by selecting from a set of edit distance based hypotheses using BERT. In contrast, in our approach we assume that multiple or even all words in the sentence have been attacked using adversarial attacks and that we do not know which ones. Then, we use an edit distance and integrate its results probabilistically with context information obtained by BERT, rather than using edit distance only for candidate selection.

3 Methods

Our complete model, which is outlined in Figure 1 on a high level, has three intuitive components. The first component is context-independent and tries to detect the tokens in a sentence from their given (potentially perturbed) surface forms. This makes sense, since we assume orthographic low-level attacks on our data. The second component uses context, via masked language modeling in BERT, to refine the probability distributions obtained from the first step. The third component uses a language model (in our case, GPT) to make a choice between multiple hypotheses. In the following, we describe each of the three components.

3.1 Context-independent probability

In the first step of our sentence restoration pipeline, we use a modified Levenshtein distance to convert the sentence into a list of probability distributions over word-piece tokens from a dictionary D. For the dictionary, we choose BERT's (Devlin et al., 2019) default word-piece dictionary.

We begin by splitting the attacked sentence S at spaces into word tokens $\tilde{w}_i$. However, to be able to use our word-piece dictionary D, we need to find the appropriate segmentation of the tokens into word-pieces.

Modified Levenshtein distance. We developed a modified version of the Wagner–Fischer algorithm (Wagner and Fischer, 1974) that calculates a Levenshtein distance to substrings of the input string and keeps track of start as well as end indices of matching substrings. For each $\tilde{w}_i$ in S, this algorithm (which is described in Appendix A.1) calculates the substring Levenshtein distance *dist* to every word-piece w_d in D.

Segmentation hypothesis. We store the computed distances $\mathit{dist}(\tilde{w}_i, w_d)$ in a dictionary C_i that maps each start-index s and end-index e to a list of distances, i.e., C_i associates

$$(s, e) \mapsto \big(\mathit{dist}(\tilde{w}_i, w_d)\big)_{w_d \in D'}$$

Here, D' selects the subset of all word-pieces in D that match $\tilde{w}_i$ at the substring between s and e. Using C_i, we can then perform a depth-first search to compose $\tilde{w}_i$ from start and end-indices in C_i. For example, a 10 character word $\tilde{w}_i$ could be segmented into two words-pieces that match the substrings from positions 1-5 and 6-10, respectively, or a single word that matches from 1-10. Let $\mathbf{c}_i$ be the set of all segmentations of $\tilde{w}_i$ from start and end indices. For example, $\mathbf{c}_i$ could be $\big\{\big((1,5),(6,10)\big),\ \big((1,10)\big)\big\}$. For each segmentation $\mathbf{c}_{i,\alpha} \in \mathbf{c}_i$, we then calculate a *total distance* $d(\mathbf{c}_{i,\alpha})$ as a sum of the minimum distances of all parts:

$$d(\mathbf{c}_{i,\alpha}) = \sum_{k=1}^{\text{len}(\mathbf{c}_{i,\alpha})} \min(C_i[\mathbf{c}_{i,\alpha,k}]) \qquad (1)$$

Using the total distances to segment each token $\tilde{w}_i$, we can now create hypotheses $\mathbf{H}$ about how the whole sentence S consisting of n tokens should be segmented into word-pieces. For this, we calculate the Cartesian product between the sets of possible segmentations for each word $\tilde{w}_i$, $i = 1, \ldots, n$:

$$\mathbf{H} = \mathbf{c}_1 \times \mathbf{c}_2 \times \cdots \times \mathbf{c}_n$$

We set the *loss* of one hypothesis $\mathbf{h} = (\mathbf{c}_{1,\alpha_1}, \ldots, \mathbf{c}_{n,\alpha_n}) \in \mathbf{H}$ as the sum of the total distances of its parts that we calculated in Eq. (1)

$$\mathit{loss}(\mathbf{h}) = \sum_{i=1}^{n} d(\mathbf{c}_{i,\alpha_i})$$

By evaluating the softmax on the negative total distances of the hypothesis, we calculate probabilities if a hypothesis $\mathbf{h}_v \in \mathbf{H}$ is equal to the true (unknown) segmentation $\mathbf{h}^*$ of the n tokens:

$$\begin{aligned} \mathbf{d}_{\mathbf{H}} &= (\mathit{loss}(\mathbf{h}))_{\mathbf{h} \in \mathbf{H}} \\ \mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S) &= [\mathrm{softmax}(-\mathbf{d}_{\mathbf{H}})]_v \end{aligned} \tag{2}$$

We will refer back to these probabilities in §3.3.

Word probability distributions. In a hypothesis $\mathbf{h} \in \mathbf{H}$, a token $\tilde{w}_i$ has a single segmentation of start and end indices associated with it, $\mathbf{c}_{i,\alpha}$. For all start- and end-indices (s, e), $C_i[s, e]$ stores the distances of the words that match $\tilde{w}_i$ between s and e. Let D' again be the dictionary containing all those words. Let w_d be a word-piece in D' and let $w^* \in D'$ be the true match for the substring between s and e of $\tilde{w}_i$. Then, we can compute a context-independent probability that w_d is equal to w^*, by evaluating the softmax on the negative distances stored in C_i:

$$\mathbf{P}_{\mathbf{h}}(w_d = w^* \mid \tilde{w}) = [\mathrm{softmax}(-C_i[s, e])]_d$$

When we do this for all words in $\mathbf{h}$ and concatenate the results, we get a vector $\mathbf{V}_{\mathbf{h}}$ of probability distributions over dictionary word-pieces. This is illustrated in Figure 2. We introduce the following notation to select a probability distribution based on its index in $\mathbf{h}$ using the subscript j:

$$\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}) := \mathbf{V}_{\mathbf{h},j}$$

Domain-specific distance. In the remainder, we will refer to the way of calculating the substring distance as described above as *attack-agnostic*. Beyond this, we also aim to leverage domain-specific knowledge. We refer to such an augmented distance as the *domain-specific* distance dist_M. Here, we modify the operation costs in the substring Levenshtein distance in certain situations.

1. Edit distance is reduced for visually similar characters. This builds on visual character representations (Eger et al., 2019). See appendix A.3 for details.

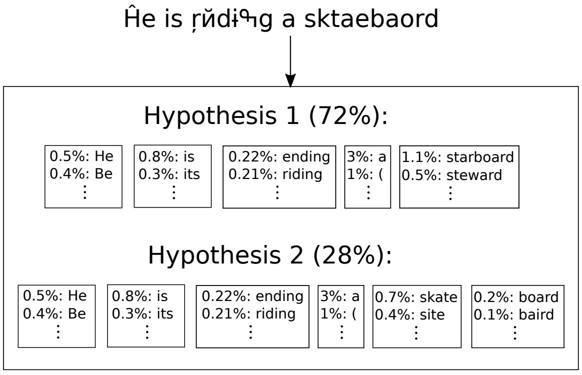

Figure 2: A context independent probability distribution over words calculated for an example input sentence. There are multiple segmentation-hypothesis associated with the sentence that each consist of a sequence of probability distributions over word-tokens.

2. Addressing intruder attacks, we reduce deletion costs depending on the frequency f of the character in the source word. Our assumption is that the same intruder symbol may be repeated in one word. Thus, we decay the cost exponentially for increasing frequency using the formula 0.75^{f-1}.

3. Vowel insertion cost is reduced to 0.3 for words that contain no vowels.

To address letter-shuffling, we additionally compute an anagram distance of how close the attacked word $\tilde{w}_i$ is to being an anagram to the dictionary word w_d. Let m be the number of characters that are in one of the two words, but not in the other. Then, our anagram distance $\mathit{dist}_{\mathrm{A}}$ computes to

$$\mathit{dist}_{\mathrm{A}}(\tilde{w}_i, w_d) = 2m + 1$$

When two words are permutations of each other, the anagram distance is minimal and otherwise it increases linearly in the number of different characters between the two words. We then take the minimum of the anagram distance and the substring Levenshtein distance with modified operation costs $\mathit{dist}_{\mathrm{M}}$ to obtain the *domain-specific* $\mathit{dist}_{\mathrm{F}}$:

$$\mathit{dist}_{\mathrm{F}}(\tilde{w}_i, w_d) = \min(\mathit{dist}_{\mathrm{A}}(\tilde{w}_i, w_d), \mathit{dist}_{\mathrm{M}}(\tilde{w}_i, w_d))$$

3.2 Context-dependent probability using BERT

In the following, we describe the context-based improvement for a single hypothesis $\mathbf{h} \in \mathbf{H}$. In

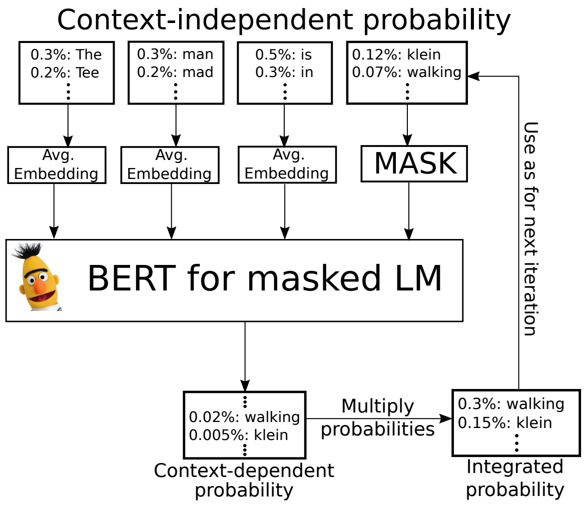

Figure 3: Iterative, context-based improvements of the word predictions using BERT for masked LM. Each iteration, a different token will be masked. We calculate context-dependent probabilities using Eq. (3) and integrate them with our context-independent probabilities in Eq. (4).

Figure 3, the whole process is illustrated for an example sentence. The number of required iterations should scale linearly with the amount of tokens in the hypothesis, so we perform $2 \cdot |\mathbf{h}|$ iterations in total. To perform one improvement iteration, we perform the following steps:

1) Select an index j, of a token, that will be masked for this iteration.

2) For the next part, we slightly modify BERT for masked LM. Instead of using single tokens as inputs as in BERT, we want to use our context-independent probability distributions over word-piece tokens. Thus, for each token $w_{\mathbf{h},j}$ in $\mathbf{h}$, we embed all relevant tokens w_d from the context-independent process described above using BERT's embedding layer and combine them into a weighted average embedding using weights $\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w})$.

3) We now bypass BERT's embedding layer and feed the weighted average embeddings and the embedding for the mask token directly into the next layers of BERT[1]. As a result, BERT provides us with a vector of scores $\mathbf{S}_{\text{BERT}}$ for how well the words from the word-piece dictionary D fit into the position of the masked word.

4) By applying the softmax on these scores, we obtain a new probability distribution over word-pieces which is dependent on the context c of the token at position j:

$$\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid c) = [\text{softmax}(\mathbf{S}_{\text{BERT}})]_d \tag{3}$$

5) We make the simplifying assumption that each word is attacked independently from the other words. Thus, the context c is independent of the attack on the word $\tilde{w}$. This means that the following equality holds:

$$\begin{aligned} &\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}, c) = \\ &\qquad \mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w})\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid c) \end{aligned} \tag{4}$$

6) We go back to step 1) and use $\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}, c)$ from Eq. (4) instead of $\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w})$ to create the average embedding at position j.

3.3 Selecting the best hypothesis with GPT

After performing the context-based improvements, we are left with multiple hypothesis $\mathbf{h} \in \mathbf{H}$. Each of them has a hypothesis probability $\mathbf{P}(\mathbf{h}{=}\mathbf{h}^* \mid S)$ and a list of word-piece probabilities of length $|\mathbf{h}|$ over dictionary words associated with it. Now, we finally collapse the probability distributions by taking the argmax to form actual sentences $S_{\mathbf{h}}$:

$$\begin{aligned} w_{\mathbf{h},j} &\leftarrow \underset{w_d}{\text{argmax}}(\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}, c)) \\ S_{\mathbf{h}} &\leftarrow (w_{\mathbf{h},j})_{j=1}^{|\mathbf{h}|} \end{aligned} \tag{5}$$

This allows us to use GPT (Radford et al., 2018) to calculate a language modeling (perplexity) score $\text{LM}_{S_{\mathbf{h}}}$ for each sentence. Using softmax, we again transform these scores into a probability distribution that describes the probability of a segmentation hypothesis $\mathbf{h}_v \in \mathbf{H}$ being the correct segmentation $\mathbf{h}^*$, based on the restored sentences $S_{\mathbf{H}}$:

$$\begin{aligned} S_{\mathbf{H}} &\leftarrow (S_{\mathbf{h}})_{\mathbf{h}\in\mathbf{H}} \\ \text{LM}_{S_{\mathbf{H}}} &\leftarrow (\text{LM}_{S_{\mathbf{h}}})_{\mathbf{h}\in\mathbf{H}} \\ \mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_{\mathbf{H}}) &= [\text{softmax}(\text{LM}_{S_{\mathbf{H}}})]_v \end{aligned}$$

The original probability $\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S)$ assigned to each hypothesis is only based on the results of the Levenshtein distance for the attacked sentence S. Thus, as $\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S)$ only depends on the character-level properties of the attacked sentence and $\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_{\mathbf{H}})$ only depends on the semantic properties of the underlying sentences, it makes sense to assume that these distributions are independent. This allows us to simply multiply them, to get a probability distribution that captures semantic as well as character-level properties:

$$\begin{aligned} &\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_{\mathbf{H}}, S) = \\ &\qquad \mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_{\mathbf{H}})\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S) \end{aligned} \tag{6}$$

---

[1] Although BERT has only been trained on the single token embeddings, we empirically found that feeding in averaged embeddings produces very sensible results.

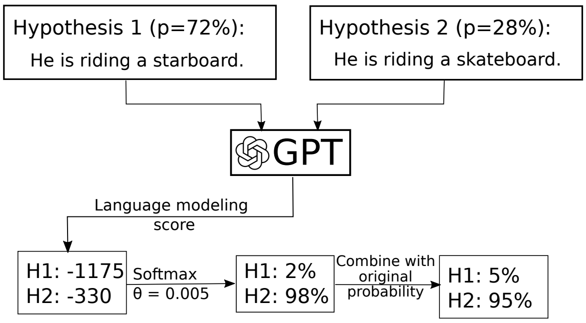

Figure 4: An example of how we use OpenAI GPT to decide on which hypothesis to choose as our final sentence prediction. The original probability of the segmentation hypothesis calculated in Eq. (2) is multiplied with a probability calculated from the language modeling score using Eq. (6).

In Figure 4, we visualize the above described process for a specific example with only 2 hypotheses.

4 Experimental Setup

To obtain adversarially attacked sentences against which to defend, we use the Eger and Benz (2020) benchmark Zéroe of low-level adversarial attacks. This benchmark contains implementations for a wide range of cognitively inspired adversarial attacks such as letter shuffling, disemvoweling, phonetic and visual attacks. The attacks are parameterized by a perturbation probability $p \in [0, 1]$ that controls how strongly the sentence is attacked.

We decided to slightly modify two of the attacks in Zéroe, the phonetic and the visual attacks. On close inspection, we found the phonetic attacks to be too weak overall, with too few perturbations per word. The visual attacks in Zeroé are based on pixel similarity which is similar to the visual similarity based defense in our *domain-specific* model. Thus, to avoid attacking with the same method we defend with, we decided to switch to a description based visual attack model (DCES), just like in the original paper (Eger et al., 2019).[4] Our modifications are described in Appendix A.2.

Evaluation Instead of evaluating on a downstream task, we evaluate on the task of restoring the original sentences from the perturbed sentences. This allows us to easier compare to human performances. It also provides a more difficult test case, as a downstream classifier may infer the correct solution even with part of the input destroyed or omitted. Finally, being able to correct the input is also important when the developed tools would be used for humans, e.g., in spelling correction.

We evaluate the similarity of the sentences to the ground-truth sentences with the following metrics:

1. **Percent perfectly restored** (PPR). The percent of sentences that have been restored perfectly. This is a coarse-grained sentence-level measure.
2. **Editdistance**. The Levenshtein (edit) distance measures the number of insertions, deletions, and substitutions necessary (on character-level) to transform one sequence into another.
3. **MoverScore** (Zhao et al., 2019). MoverScore measures the semantic similarity between two sentences using BERT. It has been show to correlate highly with humans as a semantic evaluation metric.

For all of the metrics, letter case was ignored.

Attack scenarios. We sampled 400 sentences from the GLUE (Wang et al., 2018) STS-B development dataset for our experiments. We use various attack scenarios to attack the sentences:

i) Each of the attack types of the Zéroe benchmark (see Table 1). We set $p = 0.3$ throughout.

ii) To evaluate how higher perturbation levels influence restoration difficulty, we create 5 attack scenarios for one attack scenario (we randomly chose phonetic attacks) with perturbation levels p from 0.1 to 0.9.

iii) We add combinations of attacks: these are performed by first attacking the sentence with one attack and then with another.

iv) In Random attack scenarios (rd:0.3, [rd:0.3,rd:0.3], [rd:0.6,rd:0.6]), one or two attack types from the benchmark are randomly chosen for each sentence. These constitute stronger attack situations and may be seen as more challenging test cases.

Each of the 19 attack scenarios is applied to all 400 sentences individually to create 19 test cases of attacked sentences.

Baselines and upper bounds. To evaluate how well BERT-Defense restores sentences, we compare its sentence restoration ability to two base-

---

[4] Using description based defense and pixel based attacks would have been possible just as well, but we believe doing it reversely is consistent with the original specification in Eger et al. (2019).

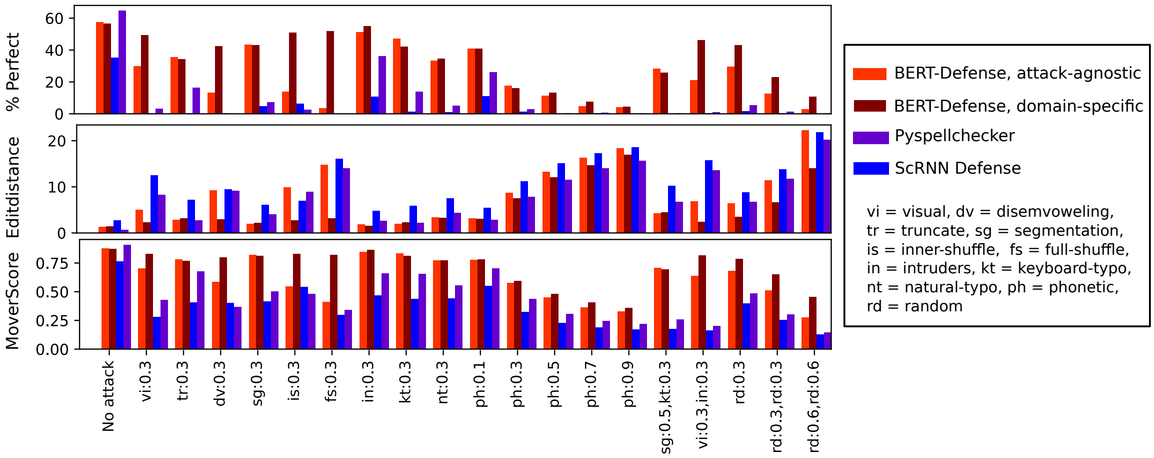

Figure 5: Comparison between BERT-Defense, and the two baseline adversarial defense tools “pyspellchecker” and “ScRNN defense”. The x-labels describe the attack and perturbation level the sentences were attacked with, before applying on of the adversarial defense methods. For conditions with two attack types, the perturbations were applied in order. For edit distance, lower is better. For the other metrics, higher is better. Exact values for the results are included in the appendix in Table 6.

lines: (a) the *Pyspellchecker* (Barrus, 2020), a simple spellchecking algorithm that uses the Levenshtein distance and word frequency to correct errors in text; (b) “ScRNN defense” from the Pruthi et al. (2019) paper. This method uses an RNN that has been trained to recognize and fix character additions, swaps, deletions and keyboard typos. Further, as we use Zeroé, a cognitively inspired attack benchmark supposed to fool machines but not humans, it is especially interesting to see how BERT-Defense compares to human performance. Thus, (c) we include human performance, obtained from a crowd-sourcing experiment on Amazon Mechanical Turk (AMT). Note that humans are often considered upper bounds in such settings.

Human experiment. Twenty-seven subjects were recruited using AMT (21 male, mean age 38.37, std age 10.87) using PsiTurk (Gureckis et al., 2016). Participants were paid $3 plus up to $1 score based bonus (mean bonus 0.56, std bonus 0.40) for restoring about 60 adversarially attacked sentences. The task took on average 43.9 minutes with a standard deviation of 20.1. Twenty of the subjects where native English speakers, seven where non-native speakers. The two groups did not significantly differ regarding their edit distances to the true underlying sentences (unequal variance t-test, $p = .85$).

We sampled 40 random sentences from nine of our attack scenarios plus 40 random (non-attacked) sentences from the original document. Each sentence was restored by four different humans. The whole set of 1600 sentences (10 scenarios times 40 sentences each times 4 repetitions) was then randomly split into 27 sets of about 60 sentences. No split contained the same sentence multiple times. Each of the 27 participants got one of these sets assigned. After a short instruction text, the participants where shown three examples of how to correctly restore a sentence (“3-shot learning”). Then they were shown the sentences in their set sequentially and entered their attempts at restoring the sentences into a text-field.

5 Results and Discussion

Comparison with baselines. Figure 5 visualizes the results (full results are in the appendix). $\mathrm{BD}_{\mathrm{agn}}$ (BERT-Defense, attack-agnostic) significantly outperforms both baselines regarding *MoverScore* and *PPR* for all random attack scenarios (p $\ll$ 0.01, equal variance t-test). However, only $\mathrm{BD}_{\mathrm{spec}}$ (BERT-Defense, domain-specific) achieves a lower edit distance than the baselines. This discrepancy between the measures is explained by the fact that, by taking context into account, BERT-Defense searches for the best restoration in the space of sensible sentences, while *Pyspellchecker* searches the best restoration for each word individually. Although *ScRNN defense* uses an RNN and is able to take context into account, we found that it also mainly seems to restore the words individually and rarely produces grammatically correct sentences for strongly attacked inputs. Table 3, which illustrates failure cases of all models, sup-

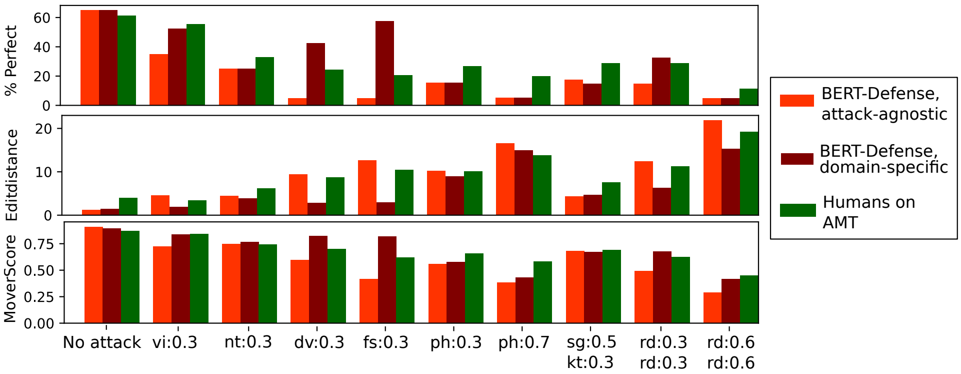

Figure 6: Comparison between BERT-Defense and humans on Amazon Mechanical Turk.

ports this. In the failure case when BERT-Defense fails to recognize the correct underlying sentence, BERT-Defense outputs a mostly different sentence that usually does make some sense, but has little in common with the ground-truth sentence. This results in much higher edit distances than the failure cases of the baselines which produce grammatically wrong sentences, while restoring individual words the best they can (this sometimes means not trying at all). Interestingly, humans tend to produce similar failure cases as BERT-Defense.

When comparing the performance on specific attacks, we see a consistent margin of about 0.2 MoverScore and 15-35 percentage points PPR between BD_{agn} and the baselines across all attacks. Exceptions include inner-shuffle, for which ScRNN-Defense is on par with BD_{agn} and segmentation attacks, which hurt the performance of the baselines far more than the performance of BERT-Defense, which includes segmentation hypothesis as an essential part of its restoration pipeline. For BD_{spec}, we see gains for attacks where we leverage domain-specific knowledge. The biggest gains of around 0.25 MoverScore are achieved against full-shuffle, inner-shuffle and disemvoweling attacks.

In the *No attack* condition, we checked if the adversarial defense methods introduce mistakes when presented with clean sentences. Indeed, all models introduce some errors: all three evaluation metrics show that BERT-Defense introduces a few more errors than *Pyspellchecker* but less than *ScRNN defense*.

Comparison with humans. As stated before, we evaluate human performance on 40 random sentences for each of nine attacks and the no attack condition (see appendix). For each of the sentences, we obtain restorations from 4 crowd-workers. For each attack scenario, we evaluate our metrics on all restorations of these 40 sentences and averaged the results. The results on the 40 attacked sentences are shown in Figure 6. While BD_{agn} performs slightly worse than humans, BD_{spec} matches human performance with respect to all three evaluation metrics. Regarding performance on specific attacks, humans are still better than BERT-Defense when it comes to defending phonetic attacks, while they have a hard time defending full-shuffle attacks. The evaluations for the *No attack* setting reveal that the crowd-workers in our experiment do make quite a few copying mistakes. In fact, they introduce slightly more mistakes than BERT-Defense.

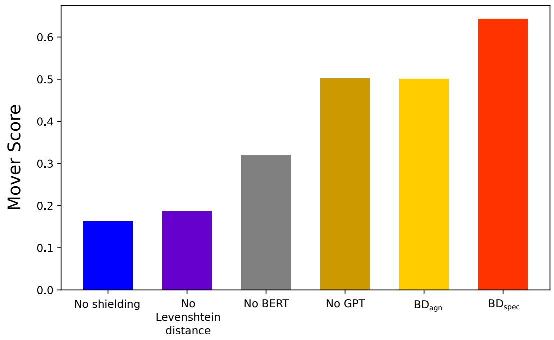

Figure 7: Ablation study for BERT-Defense. The MoverScore metric is shown for BERT-Defense with exactly one single component left out, respectively, on the rd:0.3,rd:0.3 attack. For comparison, we also show the MoverScore without shielding and after shielding with BD_{agn} or BD_{spec} using all components.

Ablation Study. We perform an ablation study to asses the contribution of each individual component of BERT-Defense. For the *No Levenshtein distance* condition, we created the context-independent probability distribution by setting the probability of known words (words in the dictionary) in the attacked dataset to one and using a uniform random distribution for all unknown words.

Attacked Sentence	**ScRNN**	**BD$_{\text{agn}}$**
To lorge doog's wronsing in sum grass.	to lorge doog's wronsing in sum grass.	two large dogs rolling in the grass.
Two large dogs runningin some grass.	two large dogs runningin some grass.	two large dogs running in some grass.
Tw large dogs rnnng in some grss.	throw large dogs running in some grss.	two large dogs running in some grass.
Two larg dog runnin in some grass.	two larg dog runnin in some grass.	a large dog running in some grass.
Twolarge dogs running income graas.	twolarge dogs running income graas.	two large dogs running into grass.
To lrg doog's rntng in sm gras ..	to long dogs ring in sm gras.	to the dogs running in the grass.

Table 2: Various illustrative attacks on the sentence "Two large dogs running in some grass." and restorations by ScRNN and BD$_{\text{agn}}$. The attacked sentences are attacked with the following attacks (top to bottom): Phonetic-0.7, Segmentation-0.3, Disemvowel-0.3, Truncate-0.3, Segmentation-0.5 & Keyboard-Typo-0.3, Random-0.3 & Random-0.3 (the last two are double attacks).

Ground-truth	china gives us regulators access to audit records
Attacked (rd:0.6,rd:0.6)	hainc gcive us regulafors essacf to tufai rsxrdeo
Bert-Defense (attack-agnostic)	haine gave us regulators space to turn us over
Human	hain gives us regulators escape to dubai suborder
ScRNN Defense	hainc give us regulafors essacf to tufai rsxrdeo
Pyspellchecker	hain give us regulators essay to tufa rsxrdeo

Table 3: Failure cases for BERT-Defense, humans and the baseline methods. Note that in the failure case, BERT-Defense and Humans restore sentences that are grammatically correct, but are mostly different from the ground-truth. On the other hand, Pyspellchecker and ScRNN Defense (Pruthi et al., 2019) either refuse to try at all for strongly attacked words or create grammatically nonsensical sentences.

When using BERT-Defense without BERT, we directly select the best hypothesis from the context-independent probability distribution using GPT. To run BERT-Defense without GPT, we select the hypothesis with the highest probability according to the results from the modified Levenshtein distance and improve it using context-dependent probabilities obtained with BERT. We evaluate on the rd:0.3,rd:0.3 attack scenario, because we think that it is the most challenging attack.

The results are shown in Figure 7. They indicate that the most important component of BERT-Defense is the Levenshtein distance, as BERT often does not have enough context to meaningfully restore the sentences, given the difficult attacks from Zeroé that typically modify many words in each sentence. Removing BERT also considerably decreases the performance of the defense model. Finally, BERT-Defense without GPT performs on par with BD$_{agn}$ in these experiments, suggesting that BERT-Defense can also be used without GPT for hypothesis selection.

More illustrating examples. To give an impression of the dataset and how the models cope with the adversarial attacks, we show more illustrating examples in Tables 2 and 5 (appendix). These indicate the superiority of our approach in that it typically generates semantically adequate sentences.

6 Conclusion

We introduced BERT-Defense, a model that probabilistically combines context-independent word level information obtained from edit distance with context-dependent information from BERT's masked language modeling to combat low-level orthographic attacks. Our model does not train on possible error types but still substantially outperforms a spell-checker as well as the model of Pruthi et al. (2019), which has been trained to shield against edit distance like attacks, on a comprehensive benchmark of cognitively inspired attack scenarios. We further show that our model rivals human crowd-workers supervised in a 3-shot manner. The generality of our approach allows it to be applied to a variety of different "normalization" problems, such as spelling normalization or OCR post-correction (Eger et al., 2016) besides the adversarial attack scenario considered in this work, which we will explore in future work.

We release our code and data at `https://github.com/yannikkellerde/BERT-Defense`.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Tyler Barrus. 2020. pyspellchecker.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Steffen Eger. 2015. Do we need bigram alignment models? On the effect of alignment quality on transduction accuracy in G2P. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1175–1185.

Steffen Eger and Yannik Benz. 2020. From Hero to Zéroe: A Benchmark of Low-Level Adversarial Attacks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 786–803, Suzhou, China. Association for Computational Linguistics.

Steffen Eger, Tim vor der Brück, and Alexander Mehler. 2016. A comparison of four character-level string-to-string translation models for (OCR) spelling error correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1634–1647.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Todd M. Gureckis, Jay Martin, John McDonnell, Alexander S. Rich, Doug Markant, Anna Coenen, David Halpern, Jessica B. Hamrick, and Patricia Chan. 2016. psiturk: An open-source framework for conducting replicable behavioral experiments online. *Behavior Research Methods*, 48(3):829–842.

Yifei Hu, Xiaonan Jing, Youlim Ko, and Julia Taylor Rayz. 2021. Misspelling correction with pretrained contextual language model. *arXiv preprint arXiv:2101.03204*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Jinfeng Li, Tianyu Du, Shouling Ji, Rong Zhang, Quan Lu, Min Yang, and Ting Wang. 2020a. Textshield: Robust text classification based on multimodal embedding and neural machine translation. In *29th*

USENIX Security Symposium (USENIX Security 20), pages 1381–1398. USENIX Association.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Fredrik Lundh and Alex Clark. 2020. Pillow.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. 2021. Token-modification adversarial attacks for natural language processing: A survey. *arXiv preprint arXiv:2103.00676*.

Elizabeth Salesky, David Etter, and Matt Post. 2021. Robust open-vocabulary translation from visual text representations. *CoRR*, abs/2104.08211.

Carnegie Mellon University. 2014. The CMU pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`. Accessed: 2020-11-25.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Haohan Wang, Peiyan Zhang, and Eric P Xing. 2020. Word shape matters: Robust machine translation with visual embedding. *arXiv preprint arXiv:2010.09997*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2020. Openattack: An open-source textual adversarial attack toolkit. *arXiv preprint arXiv:2009.09191*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4904–4913, Hong Kong, China. Association for Computational Linguistics.

A Appendices

A.1 Modified Wagner-Fischer algorithm

The modified Wagner-Fischer algorithm gets the source word S of length n and the target word T of length m as inputs and performs the following operations in a run-time $O(mn)$.

1. Initialize distance matrix $\mathbf{D}$ of size $(m+1) \times (n+1)$ with zeros
2. For $i \in [1, m+1]$ do: $\mathbf{D_{i,1}} \leftarrow i - 1$
3. Initialize a set-valued start matrix $\mathbf{M}$ of the same size as $\mathbf{D}$ with empty sets.
4. For $j \in [1, n+1]$ do: $\mathbf{M_{1,j}} \leftarrow \text{Set}\{j-1\}$
5. For $i \in [1, m+1]$ and $j \in [1, n+1]$ do:
   - Use previous entries of D to calculate total cost of getting to (i, j) with the edit distance operations:
     - Insertion: $\mathbf{D_{i-1,j}} + 1$
     - Substitution: $\mathbf{D_{i-1,j-1}} + 1$
     - Deletion: $\mathbf{D_{i,j-1}} + 1$
     - Swap: $\mathbf{D_{i-2,j-2}} + 1$
     - If $T_i = S_j$ then no operation cost: $\mathbf{D_{i-1,j-1}}$
   - Enter the lowest cost from the edit distance operations into $\mathbf{D_{i,j}}$
   - Update $\mathbf{M_{i,j}}$ by merging the set with the set-valued entries of $\mathbf{M}$ that led to (i, j) with lowest cost
6. Initialize empty list L
7. Store lowest entry of $\mathbf{D_{m+1}}$ as c and for $j \in [1, n+1]$ do
   - If $\mathbf{D_{m+1,j}} = \mathbf{c}$ do: For $m \in \mathbf{M_{m+1,j}}$ do: Add a 2-tuple (m, j) into L.
8. Return c,L

A.2 Visual and Phonetic attacks

Visual attacks. In the BERT-Defense full-distance pipeline, we exploit visual similarity (see appendix A.3). The visual attacks implemented in (Eger and Benz, 2020) are also based on visual similarity. To avoid attacking with the same method that we defend with, we decided to use VIPER-DCES (Eger et al., 2019) instead. VIPER-DCES exchanges characters based on similarity of the descriptions from the Unicode 11.0.0 final names list (e.g. LATIN SMALL LETTER A for the character 'a').

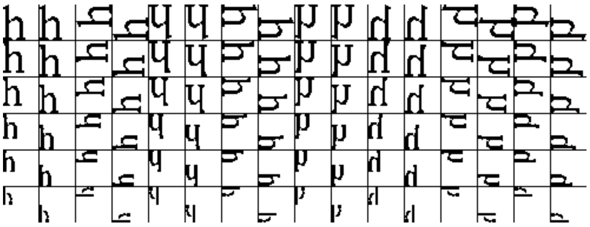

Figure 8: Different orientations/scales used for the letter h. The version that matches a Unicode character the best is used to calculate their similarity.

Phonetic attacks. The phonetic embeddings implemented in Eger and Benz (2020) do not consistently produce phonetic attacks of sufficient quality. Thus, we used a many-to-many aligner (Jiampojamarn et al., 2007; Eger, 2015) together with the CMU Pronouncing Dictionary (cmudict) (University, 2014) and a word frequency list to calculate statistics for the correspondence between letters and phonemes. To attack a word, we convert the word to phonemes using cmudict and then convert it back to letters by sampling from the statistics. The perturbation probability p for this attack controls the sampling temperature which describes how likely it is to sample letters that less frequently correspond to the phoneme in question. Using this method, we generate high-quality phonetically attacked sentences such as the one in Table 1.

A.3 Visual similarity

We calculate the visual similarity of 30000 Unicode characters to 26 letters and 10 numbers. Each glyph is drawn with Python's pillow library (Lundh and Clark, 2020) in 20pt using a fitting font from the google-Noto font collection. The bitmap is then cropped to contain only the glyph. Then the image is resized and padded on the right and bottom to be of size $30px \times 30px$. When comparing the bitmap of a unicode glyph image and a letter/number glyph, multiple versions of the letter/number bitmap are created. For letters, the lowercase as well as the uppercase versions of each letter are taken. The bitmap gets downsized to 5 different sizes between $30px \times 30px$ and $15px \times 15px$, rotated and flipped in all 8 unique ways and then padded to $30px \times 30px$ again, such that the glyph is either placed at the top-left or the bottom left. See Figure 8 for an example. The percentage of matching black pixels between bitmaps are calculated and the highest matching percentage of all version becomes the similarity score S. The substitution cost between two characters will then be calculated based on the similarity with the equation *cost* $= \max(0, \min(1, (0.8 - S) * 3))$. The parameters of this equation have been tuned, so that highly similar characters have a in very low substitution costs while weakly similar characters have next to no reduced in substitution cost.

A.4 Parameters, runtime and computing infrastructure

All experiments where run on a single machine using an Intel(R) Core(TM) i7-4790K processor and a Nvidia GeForce GTX 1070 Ti graphics card. The restoration of a single sentence in the experiments took on average 0.1 seconds using ScRNN Defense, 1.34 seconds for Pyspellchecker and 8 seconds for BERT-defense. In total, BD$_{agn}$ includes 5 free parameters, most of them controlling the temperature of the used softmax operation to ensure good relative weighting of the probability distributions. The parameter values are shown in Table 4. All additional parameters for BD$_{spec}$ have been described in §3.1.

Parameter	Value
Softmax temperature for context-independent hypothesis	10
Softmax temperature for context-independent word-probabilities	1
Softmax temperature for BERT	0.25
Softmax temperature for GPT	0.005
Max number of hypothesis	10

Table 4: Parameters for BERT-Defense.

Attacked	theensuing battls abd airstrikes killed at peast 10 militqnts.
Ground-truth	the ensuing battle and airstrikes killed at least 10 militants.
BD$_{\text{agn}}$	the ensuing battle and air strikes killed at least 10 militants.
BD$_{\text{spec}}$	the ensuing battle and air strikes killed at least 10 militants.
ScRNN Defense	tunney battls and airstrikes killed at past 10 militqnts.
Pyspellchecker	theensuing battle abd airstrips killed at past 10 militants
Attacked	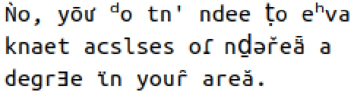 Ǹo, yōư ᵈo tn' ndee ṭo eʰva knaet acslses oɾ nḏəřeā̆ a degrɜe ïn youȓ areă.
Ground-truth	No, you don't need to have taken classes or earned a degree in your area.
BD$_{\text{agn}}$	no, you do ,' nee ,' not besides of never a degree, you are.
BD$_{\text{spec}}$	no, you do no' need to have taken classes or have a degree in your area.
ScRNN Defense	, yu so to nerve to era knaet access of need a degree ïn your areă.
Pyspellchecker	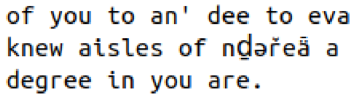 of you to an' dee to eva knew aisles of nḏəřeā̆ a degree in you are.
Attacked	A man ix riding ;n s voat.
Ground-truth	A man is riding on a boat.
BD$_{\text{agn}}$	a man is riding in a boat.
BD$_{\text{spec}}$	a man is riding in a boat.
ScRNN Defense	a man imax riding on s voat.
Pyspellchecker	a man ix riding in s vote

Table 5: Additional adversarial shielding examples on the rd:0.3,rd:0.3 dataset.

Dataset	$\mathrm{BD}_{\mathrm{agn}}$		$\mathrm{BD}_{\mathrm{spec}}$		**Pyspellchecker**		**ScRNN Defense**	
Metric	Mover	Editdist	Mover	Editdist	Mover	Editdist	Mover	Editdist
vi:0.3	0.696	5.04	0.830	2.267	0.387	8.54	0.257	12.54
tr:0.3	0.778	2.91	0.767	3.14	0.605	3.34	0.386	7.25
dv:0.3	0.574	9.27	0.794	2.995	0.335	9.53	0.379	9.48
sg:0.3	0.820	2.02	0.808	2.22	0.459	4.52	0.4	6.19
is:0.3	0.539	9.91	0.842	2.767	0.44	9.26	0.520	7.04
fs:0.3	0.399	14.78	0.688	3.227	0.310	14.41	0.277	16.12
in:0.3	0.845	1.9	0.861	1.597	0.588	3.14	0.445	4.92
kt:0.3	0.832	1.96	0.562	2.36	0.596	2.8	0.416	5.89
nt:0.3	0.764	3.38	0.512	3.322	0.504	4.91	0.423	7.59
ph:0.1	0.776	3.21	0.779	3.082	0.632	3.35	0.535	5.50
ph:0.3	0.569	8.77	0.587	7.55	0.397	8.29	0.302	11.26
ph:0.5	0.437	13.25	0.469	12.062	0.275	11.78	0.208	15.19
ph:0.7	0.350	16.37	0.395	14.735	0.218	14.33	0.167	17.34
ph:0.9	0.314	18.45	0.341	16.967	0.194	15.81	0.152	18.63
sg:0.5,kt:0.3	0.701	4.29	0.537	4.47	0.23	7.07	0.158	10.25
vi:0.3,in:0.3	0.627	6.48	0.679	2.467	0.172	13.73	0.14	15.75
rd:0.3	0.676	6.48	0.650	3.485	0.44	7.25	0.375	8.91
rd:0.3,rd:0.3	0.501	11.49	0.451	6.657	0.269	12.0425	0.232	13.92
rd:0.6,rd:0.6	0.257	22.30	0.441	14.0	0.12	20.46	0.104	21.90

Table 6: Exact scores for the results shown in Figure 5.

Event Detection as Graph Parsing

Jianye Xie[1], Haotong Sun[1], Junsheng Zhou[1]*, Weiguang Qu[1], Xinyu Dai[2]
[1]School of Computer and Electronic Information, Nanjing Normal University, China
[2]National Key Laboratory for Novel Software Technology, Nanjing University, China
`{zhoujs,wgqu}@njnu.edu.cn,daixinyu@nju.edu.cn`
`{xiejy,sunht}@nnu.edu.cn`

Abstract

Event detection is a fundamental task in information extraction. Most previous approaches typically view event detection as a trigger-based classification problem, focusing on using syntactic dependency structure or external knowledge to boost the classification performance. To overcome the inherent issues with existing trigger classification based models, we propose a novel approach to event detection by formulating it as a graph parsing problem, which can explicitly model the multiple event correlations and naturally utilize the rich information conveyed by event type and subtype. Furthermore, to cope with data sparsity, we employ a pretrained sequence-to-sequence (seq2seq) model to transduce an input sentence into an accurate event graph without the need for trigger words. Extensive experimental results on the public ACE2005 dataset show that, our approach outperforms all previous state-of-the-art models for event detection by a large margin, obtaining an improvement of 4.2% F1 score. The result is very encouraging since we achieve this with a conceptually simple seq2seq model; moreover, by extending the graph structure, this proposed architecture can be flexibly applied to more information extraction problems for sentences.

1 Introduction

Event Detection (ED) is an important task in Information Extraction (IE) that seeks to identify instances of specified types of events in text (Ji and Grishman, 2008; Li et al., 2013). For example, for the input sentence shown in Figure 1, the ED model aims to predict three event types expressed by this sentence, each of which consists of an event type label and its subtype label, according to the ACE2005 Guidelines.

Sentence: *British found resistance softer than expected, picked up reports that the local Baath Party leadership was crumbling and fought into the core, losing at least three soldiers.*
Event types: *Business: End-Org,*
Conflict: Attack, Life: Die

Figure 1: An example of event detection

ED is an actively studied task in IE where deep learning models have been the dominant approach to deliver the state-of-the-art performance (Chen et al., 2015; Nguyen et al., 2016; Sha et al., 2018; Chen et al., 2018). The last few years witness the success of graph convolutional neural networks for ED (Nguyen and Grishman, 2018; Liu et al., 2018; Yan et al., 2019; Lai et al., 2020) where the dependency trees are employed to boost the performance. Also, another line of research focused on exploiting external knowledge to improve classification (Lu et al., 2019; Liu et al., 2019a; Tong et al., 2020).

Nevertheless, most previous work typically treats ED as the identification and classification of trigger words, focusing on using various syntactic dependency structure or external knowledge to boost classification performance. Methodologically speaking, this type of trigger-based ED models suffer from the following inherent drawbacks.

Firstly, most previous approaches depend heavily on the trigger word. On the one hand, triggers are nonessential to event detection (Liu et al., 2019b); On the other hand, the identification and classification of trigger words may, to some extent, hinder the accurate recognition of the events, due to the fact that some events may be expressed by multiple discontinuous words or phrases in one sentence (See more illustrations in Section 5.4). Particularly, the trigger-based ED models are prone to suffer from the long tail issue (Tong et al., 2020). Literatures available show that, (Liu et al., 2019b)

---

* Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1630–1640
August 1–6, 2021.

is the only work for ED without using trigger words. However, in the absence of trigger, (Liu et al., 2019b) simply cast ED as a multi-label classification problem for input sentences, which cannot address the inherent issues with the trigger-based approaches, as illustrated below.

Secondly, current ED models cannot explicitly model the correlations between multiple events in one sentence. In fact, the multiple event phenomenon has been investigated by some existing works (Chen et al., 2018; Liu et al., 2018), which explore to aggregate more contextual information from surrounding words to generate a powerful representing vector for each candidate trigger by employing a self-attention mechanism or a hierarchical tagging scheme, then respectively predict the trigger label. However, note that modeling the associations between triggers is not equivalent to modeling the correlations between events. That is to say, the current models cannot explicitly model the correlations between multiple events.

Lastly, the existing approaches cannot leverage the hierarchical structure information of event type and subtype. The two kinds of event type labels contain the information with different granularity. Intuitively, the event type-level information can be used to guide the subtype-level classification. Furthermore, the event type or subtype label itself also conveys explicit semantic information that may be conducive to event prediction. However, the rich information is neglected by the existing approaches.

To address the three problems stated above simultaneously, we take a fresh look at this problem and formulate ED, for the first time, as a graph parsing problem. By regarding the multiple events expressed by one sentence as a whole, we argue that the goal of ED task is to output an event graph, as shown in Figure 2. On the one hand, the event graph is constructed to model the potential interactions between the multiple events; on the other hand, this graph structure can flexibly integrate more event type information. Furthermore, under the graph parsing formulation, we employ a seq2seq model for event graph parsing, without the need for the identification of the trigger words. In particular, to cope with the data sparsity problem with the ED task, we adopt a pretrained seq2seq model, BART (Lewis et al., 2020), to accurately generate the event graph. Experimental results demonstrate that our method substantially outperforms all previous state-of-the-art models on the public dataset ACE2005.

To sum up, this paper makes the following contributions:

1. To the best of our knowledge, it is the first time to formulate the event detection task as graph parsing, which delivers some typical benefits compared to the existing ED models. First, this graph parsing formulation can explicitly model the correlations between multiple events in one sentence; second, the event graph can be flexibly constructed to reflect the hierarchical structure of event type and utilize the semantic representations of the event type labels.

2. We further propose a novel generation-based approach to predict the event graph via a seq2seq model. The proposed transducer can directly derive the events from the global contextual information in the input sentences, without being limited to the representations of trigger words. Particularly, we employ a pretrained model, BART (Lewis et al., 2020), to generate a linearized event graph, thereby addressing the data sparsity.

3. The extensive experiments over the public dataset ACE2005 demonstrate that our approach outperforms the previous state-of-the-art models for ED by a large margin, without using the syntactic dependency information and any external trigger knowledge.

Encouragingly, the proposed graph parsing paradigm is not limited to the ED task. By modifying the graph structure according to the target task, this graph parsing formulation can be flexibly applied to more information extraction problems, such as event extraction, relation extraction and so on.

2 A Novel View of Event Detection

2.1 Task Description

Event detection task requires that certain specified types of events, which are mentioned in the annotated data, to be detected. The most common used benchmark dataset in previous work is ACE 2005 corpus. The task defines 8 event types such as Life, Business and so on, and 33 subtypes such as Attack, End-Position, etc. By the ACE annotation guideline, every event mention is annotated with a

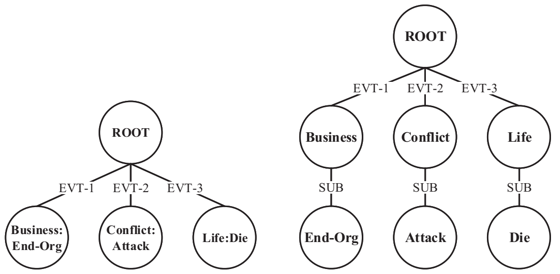

(a) Single-node strategy (b) Node-splitting strategy

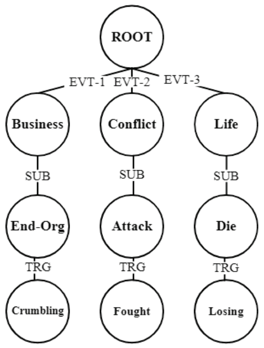

(c) Trigger generation strategy

Figure 2: Three different event graphs constructed using the three different strategies, for the instance in Figure 1.

trigger that is a key word or phrase that most clearly expresses the event occurrence.

2.2 Formulating ED as Graph Parsing

Traditionally, the goal of ED consists of identifying trigger words (trigger identification) and classifying them for the event types of interest (event classification) (Nguyen and Grishman, 2018; Liu et al., 2018; Lai et al., 2020; Tong et al., 2020). However, as pointed out by (Liu et al., 2019b), triggers are nonessential to event detection. *More importantly*, some events may be triggered by multiple discontinuous words or phrases in one sentence, not by a single word or single phrase.

Take a concrete example in ACE 2005 dataset to illustrate: *She lost her seat in the 1997 election.* In this sentence, an event type (*Personnel:Elect*) is mentioned, and its gold trigger was labelled as the word *lost*. In effect, to correctly recognize the event type (*Personnel:Elect*) from this sentence, we should comprehensively consider both the phrase *lost her seat* and the word *election* in the sentence (See more cases in Section 5.4). Therefore, it does not seem plausible that the problem of predicting an event from a whole sentence is reduced to the representation learning of the single trigger word for trigger classification or sequence labelling.

In this paper, we look at the ED task from a new perspective. Given an input sentence, ED aims to recognize and predict the mentioned event types. Intuitively, the multiple events derived from the same sentence should have a certain degree of correlations between them. Therefore, to model the correlations, we can view the multiple events corresponding to a sentence as a whole, by linking them together as a single graph, as shown in Figure 2(a). To be specific, we first introduce a special node as the root, and then attach each event type node as a child of the root. It is worth to note that *the root of this event graph is not a virtual node*. The root can take two possible values: *EVTS* and *NA*. While the input sentence does not contain any event, the root is assigned the value *NA*; otherwise it is assigned the value *EVTS*. Therefore, the prediction of root value is to judge whether the input sentence expresses some events or not.

In addition to facilitating modeling multiple event correlations, our graph parsing formulation for ED also allows for the straightforward inclusion of other types of graph-structured features. Particularly, the other typical benefits of graph parsing formulation are as following:

- **Flexibly leveraging the hierarchical structure of event type and subtype.** Previous approaches to ED treat the event subtypes simply as 33 separate event types and ignore the hierarchical structure between the event type and subtype. Obviously, the two kinds of type labels contain the information with different granularity, both of which are indicative for event detection. Fortunately, our graph parsing framework allows for reflecting the hierarchical structure information by simply extending the event graph. A natural *node-splitting strategy* can be adopted by decomposing one event node into two nodes: one event type node and one subtype node, and linking the subtype node as a child of the type node, as shown in Figure 2(b). Compared to the *single-node strategy* for event representing, as shown in Figure 2(a), the *node-splitting* strategy indeed introduces an effective *coarse-to-fine* way for event type prediction, and therefore it is theoretically superior to the *single-node* strategy.

- **Naturally utilizing the semantic representation of event type label.** Most previous classification-based approaches to ED generally view each event type as a specific class, omitting the semantic information conveyed by the event type label. In fact, the event type label itself, such as *Divorce*, *Injure*, etc, is informative to the learning of ED models. In our graph parsing formulation, it is straightforward to incorporate the semantic representation of type label into the model. Specifically, during decoding, we can encode every previously generated node with the corresponding type label embedding to assist the prediction of later nodes. For example, in Figure 2(b), once the event type node *Conflict* is generated, we can leverage the semantic representation of the type *Conflict* to help to predict the subtype node *Attack* and next type node *Life*.

- **Reversely generating the event triggers.** If the event triggers are needed in some cases, it is very flexible and convenient for our graph parsing framework to generate the trigger words by simply extending the event graph to yield the triggers for every predicted events. Specifically, we can append the trigger nodes as the children of event type nodes in the event graph as shown in Figure 2(c), and thus generate the trigger in a reverse direction. That is to say, unlike the traditional fashions in the existing trigger classification based models, we first predict the event types from the input text, and then output the corresponding triggers for previously predicted event types.

3 Event Graph Parsing via a Pretrained Seq2seq Model

Under our graph parsing formulation, the ED task is to transduce an input sentence into an event graph, as illustrated in Section 2. To achieve this, we choose to predict nodes sequentially rather than simultaneously, because (1) we believe the previous node generation is informative to the current node prediction; (2) variants of efficient seq2seq models (Bahdanau et al., 2014; See et al., 2017) can be employed to model this process.

Theoretically, the advantages of applying a seq2seq model to event graph parsing are two-fold. First, there is no need to use trigger words for event detection. Second, when predicting an event type node during decoding, the global contextual information in the input sentence can be taken into consideration by the cross-attention mechanism between the decoder and the encoder.

However, in the preliminary research experiments, the generic seq2seq event detection approaches did not obtain satisfactory performance. The main reason may be that the seq2seq-based approaches are generally data-hungry. Especially, for the ED task based on the benchmark dataset ACE 2005, data sparsity may be the significant challenge.

To deal with sparsity, we explore to employ transfer learning by exploiting BART (Lewis et al., 2020) to generate a linearized event graph incrementally. BART is a Transformer-based encoder-decoder model which is pretrained through a denoising self-supervised task, and has shown significant improvements in conditioned generation tasks, especially gaining state-of-the-art results on summarization. For given input sentence, our event graph generation is, to some extent, similar to the abstractive summarization task. We therefore hypothesize that BART's denoising pretraining should be suitable for this task.

While applying BART to event graph parsing, we first need to create the reference node list by a depth-first traversal over the gold event graph for training. In the preliminary experiments, we found that the labels of edges (e.g. *EVT-1*, *SUB*, etc.) are not informative to event detection; we therefore omit these labels while linearizing the event graph. For example, in Figure 2(b), the linearization order contains *EVTS*, *Business*, *End-Org*, *Conflict*, *Attack*, *Life*, *Die*.

Furthermore, we augment the vocabulary of BART with some special symbols in order to make it suitable for event detection. For example, we first expand the vocabulary by adding the root value (*EVTS* or *NA*) in the graph; then for some special event subtype, such as *Start-Org* and *End-Position*, we need to append these subtype labels to the vocabulary. Additionally, we need to expand the embedding matrices of encoder and decoder to include the average of the subword constituents of each special label.

For decoding in testing phase, we design a constrained beam search algorithm to generate the node sequence incrementally, as illustrated in Algorithm 1. Note that each particular event subtype is subject to the event type constraints. Apparently, this constrained decoding algorithm performs the prediction of event type nodes in a coarse-to-fine fashion, and can ensure to generate a valid event graph.

Algorithm 1: Constrained beam search

```
$score = 0, Y = \{BOS\}$;
$beam = \{Y, score\}$;
for i = 1 to maxlen do
    $new\_beam = \{\}$;
    $\{Y, score\} = beam.pop()$;
    if i==1 then
        if $P(EVTS) < P(NA)$ then
            break;
    else if i==2 then
        for $v_i$ in type_set do
            $Y = Y \cup v_i, score+ = P(v_i)$;
            $new\_beam.push(\{Y, score\})$;
    else
        last_token = Y[i-1];
        if last_token==EOS then
            $new\_beam.push(\{Y, score\})$;
            continue;
        else if last_token in type_set then
            for $v_i$ in constrained_subtype_set do
                $Y = Y \cup v_i, score+ = P(v_i)$;
                $new\_beam.push(\{Y, score\})$;
        else
            $set = type\_set + \{EOS\}$;
            for $v_i$ in set do
                $Y = Y \cup v_i, score+ = P(v_i)$;
                $new\_beam.push(\{Y, score\})$;
    $beam = new\_beam.topK()$;
$\{Y, score\} \leftarrow beam.topK(k = 1)$;
```

4 Experiments

4.1 Dataset and Evaluation Metrics

We utilized the ACE 2005 corpus as our dataset. For comparison, as the same as previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013), we used the same test set with 40 newswire articles and the same development set with 30 other documents randomly selected from different genres and the rest 529 documents are used for training.

Also, following previous work (Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2019a; Tong et al., 2020), we use the following criteria to evaluate the results:

Precision: the proportion of correctly predicted events in total predicted events.

Recall: the proportion of correctly predicted events in total gold events of the dataset.

F1-measure: $\frac{2*P*R}{P+R}$

4.2 Hyperparameters

The hyperparameters are tuned on the validation set. For all the experiments, we use the same model hyperparameters as BART-Large, as defined in Huggingface's transformers library. The models are trained using cross-entropy with RAdam as optimizer and a learning rate of $5 * 10^{-5}$. Gradient is accumulated for 10 batches. Dropout is set to 0.25. Our models are trained for 80 epochs, the batch size in our training experiments is set to 300. For decoding, we set beam size to 5 and the constant maxlen to 20.

4.3 Overall Performance

In this section, we comprehensively compare our performance with the following state-of-the-art methods:

JRNN proposes a joint event extraction model based on recurrent neural network to improve ED (Nguyen et al., 2016).

DLRNN exploits document information via recurrent neural networks (Duan et al., 2017).

TBNNAM is the first work on detecting events without triggers (Liu et al., 2019b).

GCN-ED uses an argument pooling mechanism for event detection based on GCN (Nguyen and Grishman, 2018).

JMEE uses GCN with highway network and self-attention (Liu et al., 2018).

MOGANED is an advanced graph neural network (GNN) model. It proposes a multi-order graph attention network to effectively model the multi-order syntactic relations in dependency trees and improve ED (Yan et al., 2019).

EE-GCN simultaneously exploits syntactic structure and typed dependency label information to perform ED (Cui et al., 2020).

GatedGCN proposes a novel gating mechanism to filter noisy information in the hidden vectors of the GCN models for ED (Lai et al., 2020).

Lu's DISTILL proposes a -learning approach to distill generalization knowledge to handle overfitting (Lu et al., 2019).

TS-DISTILL exploits the entity ground-truth and uses an adversarial imitation based knowledge distillation approach for ED (Liu et al., 2019a).

EKD leverages the wealth of the open-domain trigger knowledge to improve ED (Tong et al., 2020).

Table 1 shows the overall performance comparison between our best system and the above state-of-the-art models. In Table 1, we roughly divide the state-of-the-art methods into three groups: data-driven methods, syntactic dependency enhanced

Method	Precision	Recall	F1
JRNN	66.0	73.0	69.3
DLRNN	77.2	64.9	70.5
TBNNAM*	76.2	64.5	69.9
GCN-ED†	77.9	68.8	73.1
JMEE†	76.3	71.3	73.7
MOGANED†	79.5	72.3	75.7
EE-GCN†	76.7	78.6	77.6
GatedGCN†	78.8	76.3	77.6
Lu's DISTILL∧	76.3	71.9	74.0
TS-DISTILL∧	76.8	72.9	74.8
EKD†∧	79.1	78.0	78.6
Ours	**83.2**	**82.4**	**82.8**

Table 1: Overall Performance on ACE2005 dataset (%). The results of baselines are adapted from their original papers. † indicates that the method uses dependency structures, ∧ indicates that the method uses external knowledge and resources, * indicates that the method does not need triggers.

methods and knowledge enhanced methods. Additionally, TBNNAM is the only work that does not use trigger for ED, similarly to our work.

From Table 1, we can see that our approach that adopts the *node-splitting strategy* to construct the event graph achieves the best Precision, Recall and F1 score among all the compared methods. It is worth noting that *our model simultaneously significantly improves both Precision and Recall without using any additional information including the POS tags, the syntactic dependency and external knowledge*, which shows the superiority of the proposed graph parsing formulation for ED.

5 Analysis

5.1 Effect of Exploiting the Event Type Information

Unlike previous models, our graph parsing formulation can flexibly incorporate the rich information conveyed by the event types into the event graph, including the hierarchical structure of event type and the semantic representation of the type label. In this section, we prove the effect of exploiting the event type information by the ablation test.

First, we compare the two different strategies for constructing the event graph, i.e. the *single-node strategy* and *node-splitting strategy* (as illustrated in Section 2). As expected, the *node-splitting strategy* that uses the hierarchical structure information significantly outperform its counterpart, as shown in Table 2.

Next, we verify the effect of the semantic representation of the type label by treating every label of event type or subtype as a special symbol, not using their word embedding learned in the pretrained language model. From the results in Table 2 we can observe that, ignoring the semantic representation of type label leads to a significant performance drop, especially for the *node-splitting strategy*.

Method	P	R	F1
Single-node strategy(-)	80.0	76.0	78.0
Single-node strategy	76.8	81.0	78.8
Node-splitting strategy(-)	79.0	78.8	78.9
Node-splitting strategy	**83.2**	**82.4**	**82.8**

Table 2: Performance comparison of differently structured event graphs with or without label embeddings. The symbol - indicates that the method does not use type label embeddings.

5.2 Effect of Multiple Event Recognition

Compared to the existing work, our ED approach provides a more natural formulation to model the multiple event correlations. To evaluate the effect of our approach to the multiple event recognition, we divide the test data into two parts (**1/1** and **1/N**) following previous work (Chen et al., 2015; Nguyen et al., 2016), and perform evaluations separately. **1/1** means that one sentence only has one event; otherwise, **1/N** is used. Table 3 illustrates the performance (F1 scores) of **DMCNN** (Chen et al., 2015), **JRNN** (Nguyen et al., 2016), **JMEE** (Liu et al., 2018) and **HBTNGMA** (Chen et al., 2018), the four baseline models and our model for ED task. As shown in Table 3, our model significantly outperforms all the other methods. In the **1/N** data split, our method is **9.8%** better than the best baseline, **JMEE**. The experimental results demonstrate that our method works well on the task of multiple event recognition.

5.3 Effect of Pretrained Seq2seq Model

In this section, we inspect the effect caused by the pretrained seq2seq model, BART. For this end, we implemented a traditional seq2seq event graph parsing system as baseline, by adopting the pointer-generator network (See et al., 2017), and using BERT embeddings (Devlin et al., 2019) to encode

Method	1/1	1/N	all
DMCNN	74.3	50.9	69.1
JRNN	75.6	64.8	69.3
HBTNGMA	78.4	59.5	73.3
JMEE	75.2	72.7	73.7
Ours	**83.1**	**82.5**	**82.8**

Table 3: Performance comparison on single event sentences (1/1) and multiple event sentences (1/N).

the words. And the experimental results are illustrated in Table 6. We can see that, compared to most state-of-the-art models as shown in Table 1, this baseline obtains a competitive result. However, BART-based model can further substantially improve F1 score by 6.6%, which indicates the importance of the pretrained seq2seq for tackling data sparsity.

Method	P	R	F1
Pointer-generator-based	81.2	71.8	76.2
BART-based	**83.2**	**82.4**	**82.8**

Table 4: Performance of test set with or without pretrained seq2seq model.

5.4 Analysis of Cross-Attention Mechanism

In the absence of trigger words, can our Transformer-based seq2seq event detection framework capture the key clues in the source sentence that express the target event type? In this section, we answer this question by the case study.

Figure 3 presents several examples of the attention distributions learned by our model. In the first case, the target event type is *Life:Die* and the gold trigger is the word *killed*. We can see that when predicting this event type, our attention not only successfully attend the trigger word *killed*, but also attend another strongly indicative phrase *two people* with higher score. In the second case, the target event type is *Conflict:Attack*, and the gold trigger is the word *strike*. It can be observed that, the three words: *destroyed*, *houses* and *killed* are assigned with higher attention scores than the trigger *strike*, which seems plausible for this target type prediction. In the third case, the target event type is *Personnel:Elect*, and the gold trigger is the word *lost*. For this target type, there are relatively strong connections with the phrase *lost her seat* and another indicative word *election*.

These cases demonstrate that, though the triggers are not used in our model, the cross-attention mechanism between the decoder and encoder can learn to automatically and accurately capture the correlation between the target event type and multiple indicative words or phrases in the source sentence.

5.5 Can Our Approach Alleviate the Long Tail Issue?

The trigger-based event detection models generally suffer from the long tail issue (Lu et al., 2019; Tong et al., 2020). Taking the benchmark ACE2005 as an example, trigger words with frequency less than 5 account for 78.2% of the total. The long tail issue makes the trigger-based models perform poorly on unseen/sparsely labeled trigger words. In this section, we evaluate whether our approach could cope with the long tail issue.

Following previous work (Tong et al., 2020), we divide the event instances in the test set into three categories: *Unseen*, *Sparsely-Labeled* and *Densely-Labeled*, according to their trigger frequency in the training set. Specifically, the frequency of Sparsely-Labeled is less than 5 and the frequency of Densely-Labeled is more than 30. Also, following the work (Tong et al., 2020), we choose the following

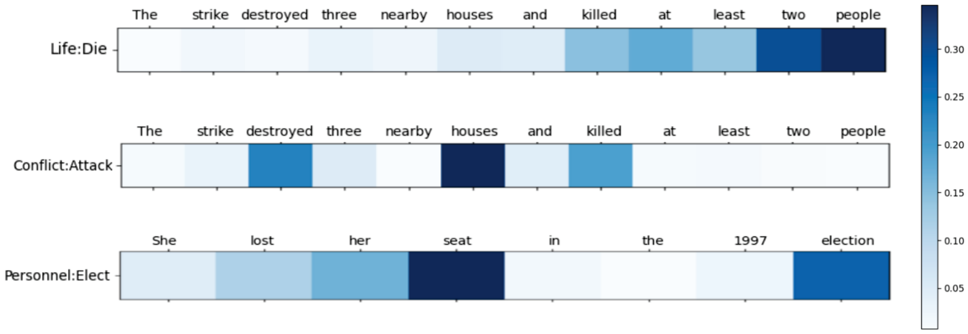

Figure 3: Visualization of cross-attention scores of sample instances learned by our model.

baselines for comparison: (1) **DMBERT** (Chen et al., 2015), (2) **DGBERT** (Chen et al., 2017), (3) **BOOTSTRAP** (He and Sun, 2017), and (4) the method **EKD** (Tong et al., 2020).

As shown in Table 5, our approach substantially outperforms all baselines in all three settings, especially on unseen (+**12.9%**) and sparsely-labeled settings (+**4.8%**). Why can our approach effectively mitigate the long tail issue? Besides the better generalization endowed by our seq2seq event graph parsing formulation, an important possible reason is that since our approach adopts a trigger-free way to detect the events, the event types corresponding to the unseen or sparsely-labeled triggers can also be expressed with other different triggers and thus appear many times in the training set, thereby alleviating the long tail problem. The experimental results clearly indicate that, *the trigger-free ED approach may be a better alternative to the traditional trigger-based models*.

5.6 Effect of the Generation of Event Triggers

In this section, we investigate the performance of the trigger generation. As mentioned in Section 2.2, our graph parsing framework can also conveniently generate the event triggers by simply appending the trigger nodes as the children of event type nodes in the event graph, as shown is Figure 2(c). The experimental results of trigger generation are illustrated in Table 6.

	P	**R**	**F1**
event type evaluation	80.6	84.5	82.5
event trigger evaluation	78.1	81.8	80.0

Table 6: Performance of event trigger generation.

We can observe that the F1-score of event type evaluation achieves 82.5%, remaining almost unchanged as the result of 82.8% shown in Table 1, and that the F1-score of trigger recognition achieves 80.0%, significantly outperforming the existing trigger classification based models.

6 Related Work

6.1 Event Detection

In earlier ED studies, the traditional feature-based methods focused on exploiting the lexical and global features to detect events (Ji and Grishman, 2008; Li et al., 2013). Most recent works have focused on using neural networks in this task and have achieved significant progress. We roughly divide the recent approaches into three categories as following:

Sequence-based models: This line of research operates on the word sequences using the deep neural networks. (Chen et al., 2015) devises a dynamic multi-pooling convolutional neural network to capture more information. (Nguyen et al., 2016) presents a joint model based on bidirectional RNN for event extraction. (Sha et al., 2018) adds dependency arcs with weight to BiLSTM to make use of tree structure and sequence structure simultaneously. (Chen et al., 2018) exploits a hierarchical and bias tagging networks to detect multiple events collectively.

GCN-based models: This line of research adopts the Graph Convolutional Network (GCN) over the dependency tree of a sentence to boost the performance. (Nguyen and Grishman, 2018) is the first attempt to use GCN in ED. (Liu et al., 2018) employs a syntactic GCN and a self-attention mechanism to model multiple events extraction. (Yan et al., 2019) improves GCN by combining multi-order word representation from different GCN layers.

Knowledge Distillation-based models: More recently, some approaches focus on leveraging various external knowledge to improve classification.

Method	Unseen			Sparsely Labeled			Densely Labeled		
	P	R	F1	P	R	F1	P	R	F1
DMBERT	66.7	45.9	54.4	74.4	70.7	72.5	84.8	83.5	84.1
DGBERT	76.5	42.6	54.7	75.7	70.1	72.8	85.9	83.8	84.3
BOOTSTRAP	73.7	45.9	56.6	76.0	71.3	73.6	90.6	83.5	86.9
EKD	79.0	52.0	62.7	80.8	72.4	76.4	**92.5**	82.2	87.1
Ours	**92.2**	**64.1**	**75.6**	**87.9**	**75.5**	**81.2**	85.7	**90.6**	**88.1**

Table 5: Performance comparison on the unseen, sparsely-labeled and densely-labeled settings.

(Lu et al., 2019) proposes a new representation learning framework to distill both discrimination and generalization knowledge for ED. (Liu et al., 2019a) uses an adversarial imitation based knowledge distillation approach for ED. (Tong et al., 2020) proposes an enrichment knowledge distillation model to leverage external open-domain trigger knowledge to address the long-tail issue.

Unlike the existing ED models based on trigger classification, we formulate ED as a novel graph parsing problem, therefore it can explicitly model the multiple event correlations and incorporate the rich information regarding the event types.

6.2 Pretained Seq2seq Models

Pre-training a universal model and then fine-tuning the model on a downstream task have recently become a popular strategy in the field of natural language processing (Devlin et al., 2019). Recent studies also propose approaches to pre-training seq2seq models, such as **MASS** (Song et al., 2019), **PoDA** (Wang et al., 2019), **PEGASUS** (Zhang et al., 2019), **BART** (Lewis et al., 2020), and **T5** (Raffel et al., 2019)

In this paper, our experiments only examine BART. We leave explorations of these models for future work.

7 Conclusion

This paper presents the first work to formulate ED as a graph parsing task, and to introduce a novel generation-based method to predict event graph by using a pretrained seq2seq model. Our approach is conceptually simple and does not use syntactic dependency information and any other extra knowledge; however, it significantly outperforms the traditional trigger classification-based encoder-only approaches, advancing the state of the art in event detection.

In future work, we will integrate the syntactic dependency structure and external knowledge into our model to further improve the ED performance; in particular, we will explore to extend our graph parsing architecture to more IE problems, such as event extraction, relation extraction and so on. For example, by adding the event argument nodes to the event graph and linking them as the children of event type nodes, the event extraction problem should be solved in a similar way.

Acknowledgments

We thank all reviewers for the valuable comments. This work is supported by the National Natural Science Foundation of China (No. 61472191, No. 61976114, No. 61772278 and No. 61936012).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–419, Vancouver, Canada. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276, Brussels, Belgium. Association for Computational Linguistics.

Shiyao Cui, Bowen Yu, Tingwen Liu, Zhenyu Zhang, Xuebin Wang, and Jinqiao Shi. 2020. Edge-enhanced graph convolution networks for event detection with syntactic relation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2329–2339.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1:*

Long Papers), pages 352–361, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136, Portland, Oregon, USA. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5405–5411, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden. Association for Computational Linguistics.

Jian Liu, Yubo Chen, and Kang Liu. 2019a. Exploiting the ground-truth: An adversarial imitation based knowledge distillation approach for event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6754–6761.

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019b. Event detection without triggers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 735–744, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2019. Distilling discrimination and generalization knowledge for event detection via delta-representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4366–4376, Florence, Italy. Association for Computational Linguistics.

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. *CoRR*, abs/1905.02450.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5887–5897, Online. Association for Computational Linguistics.

Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. 2019. Denoising based sequence-to-sequence pre-training for text generation. *CoRR*, abs/1908.08206.

Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. 2019. Event detection with multi-order graph convolution and aggregated attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5766–5770, Hong Kong, China. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *CoRR*, abs/1912.08777.

Toward Fully Exploiting Heterogeneous Corpus:
A Decoupled Named Entity Recognition Model with Two-stage Training

Yun Hu[12]*, **Yeshuang Zhu**[3], **Jinchao Zhang**[3], **Changwen Zheng**[12], **Jie Zhou**[3]
[1] Institute of Software, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Pattern Recognition Center, WeChat AI, Tencent Inc, China
{yunhu2016,changwen}@iscas.ac.cn
{yshzhu,dayerzhang,withtomzhou}@tencent.com

Abstract

Named Entity Recognition (NER) is a fundamental and widely used task in natural language processing (NLP), which is generally trained on the human-annotated corpus. However, data annotation is costly and time-consuming, which restricts its scale and further leads to the performance bottleneck of NER models. In reality, we can conveniently collect large-scale entity dictionaries and distantly supervised data. However, the collected dictionaries are lack of semantic context and the distantly supervised training instances contain large noise, which will bring uncertain effects to NER models when directly incorporated into the high-quality training set. To address the above issue, we propose a BERT-based decoupled NER model with two-stage training to appropriately take advantage of the heterogeneous corpus, including dictionaries, distantly supervised instances, and human-annotated instances. Our decoupled model consists of a Mention-BERT and a Context-BERT to respectively learn from the context-deficient dictionaries and noised distantly supervised instances at the pre-training stage. At the unified-training stage, the two BERTs are trained together on human-annotated data to predict the correct labels for candidate regions. Empirical studies on three Chinese NER datasets demonstrate that our method achieves significant improvements against several baselines, establishing the new state-of-the-art performance.

1 Introduction

Named entity recognition is a fundamental Natural Language Processing task that labels each word in sentences with predefined types, such as Person (PER), Location (LOC), Organization (ORG), etc. The results of NER can be used in many downstream NLP tasks, e.g., relation extraction (Bunescu and Mooney, 2005), information retrieval (Chen et al., 2015), and question answering (Yao and Van Durme, 2014). Supervised methods are mainstream approaches to NER, including CRF (Lafferty et al., 2001) and neural network models (Collobert et al., 2011; Lample et al., 2016; Ma and Hovy, 2016). Recently, large-scale pre-trained language models fine-tuned upon a limited amount of annotated data achieve competitive or better performance in NER task (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019b).

Supervised NER methods require a sufficient amount of sentence-level annotated data, even for the methods using pre-trained language models. However, obtaining sentence-level annotated data is expensive and thus leads to small training data size and performance bottleneck of supervised models. In practice, entity dictionaries (or gazetteers) and unlabeled corpora can be obtained at a low cost. Furthermore, distantly supervised data can be automatically generated by matching the unlabeled data against entity dictionaries. These data can form a heterogeneous corpus, which has potential to improve the NER task. However, dictionaries contain only entity mentions without context, and distantly supervised data can be highly noisy in terms of wrong labels and wrong boundaries. As a result, it is unwise to treat dictionaries and distantly supervised data equally to human-annotated ones.

To better utilize heterogeneous corpus, we propose a BERT-based decoupled NER model with two-stage training. The decoupled model decouples mention information and context information with a *Mention-BERT* and a *Context-BERT*, which can better exploit the information in entity data and distantly supervised data respectively. In the pre-training stage, the Mention-BERT can be pre-trained using the entity dictionary with a classifica-

---

[*] Joint work with Pattern Recognition Center, WeChat AI, Tencent Inc. Work was done when Yun Hu was intern at WeChat AI.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1641–1652
August 1–6, 2021.

tion task, and the Context-BERT can be pre-trained using the distantly supervised data with two auxiliary tasks (masked language modeling task and classification task). During inference, the decoupled model can utilize the mention information and context information together to make the final prediction. We evaluate our methods on three Chinese NER datasets. Experimental results show that our method outperforms baseline methods and achieves the best results, demonstrating the effectiveness of our methods. The contributions of our work can be summarized as follows [1]:

- We propose a decoupled NER model with two-stage training, which can fully exploit heterogeneous corpus consisting of dictionaries, distantly supervised instances, and human-annotated instances.
- Our model achieves the state-of-the-art results on three common Chinese NER datasets, significantly outperforming current SOTA by 1.51% on OntoNotes and 1.7% on Weibo, as well as obtaining a slight but noticeable gain on MSRA.

2 Background

2.1 Named Entity Recognition

The task of named entity recognition is to find entities in sentences with predefined types, such as PER, LOC, and so on. Given an input sentence $X = \{x_1, x_2, ..., x_n\}$ where x_i denotes the i-th token, and a predefined tag set Y, NER can be modeled as a sequence labeling or region-based classification task. In sequence labeling approaches, the model aims to assign a label $y \in Y$ to each token x_i. In region-based approaches, the model examines each candidate region $\{x_i, x_{i+1}, ..., x_{i+k}\}$ and attempts to assign a label $y \in Y$ to it, where i is the starting position of the region in sentence, and k is the length of the region. Our model follows the framework of region-based approaches.

2.2 BERT-NER Model

Recently, large-scale pre-trained language models, such as BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018), are widely used in NLP and yield state-of-the-art performances on many tasks. Pre-trained language models follow a two-stage paradigm. They are first pre-trained on large-scale unlabeled texts via self-supervised tasks such as masked language modeling and next sentence prediction, and then fine-tuned on relatively small labeled data of downstream tasks.

BERT-NER model is easily adapted from pre-trained BERT model and can achieve competitive performance. Given a sentence X, BERT first outputs the sentence representation $H = \{h_1, h_2, ..., h_n\}$, where h_i is the representation of token x_i. Then, H is passed through a feed forward network (FNN) to obtain the label sequence $\{y_1, y_2, ..., y_n\}$:

$$y_i = softmax(W \cdot h_i + b) \qquad (1)$$

where W, b are parameters of the FFN, and y_i is the predicted label of x_i.

Our model is built on top the BERT model. Compared with the BERT-NER, we propose a new decoupled architecture to better utilize heterogeneous data. Besides, different from the training tasks of BERT, our model introduces task-aware pre-training tasks into a two-stage training framework.

3 Approach

3.1 Model Architecture

Generally, an effective NER model should capture two types of information for determining an entity, i.e., mention information and context information. In traditional NER models, the mention and context information are typically coupled in annotated data. Our proposed model decouples the two types of information, making them to be more explicit and easily learned from the heterogeneous corpus.

Overview. As shown in Figure 1, our model consists of three main parts: a *Mention-BERT*, a *Context-BERT*, and a *Global-Classifier*. The input is a sentence along with a region denoting a mention candidate. The model will decouple the mention from the context and feed the two parts into the Mention-BERT and the Context-BERT respectively. Then, the outputs of the two BERTs will be concatenated and passed through the Global-Classifier to obtain the final label prediction. Additionally, the two BERT outputs are also passed through a mention-focused and a context-focused classifier respectively to provide auxiliary supervision during training, which we will elaborate later.

Mention-BERT. The Mention-BERT is used to capture the representation of the mention that to-be recognized. The input of the Mention-BERT

---

[1] Our code and data are available at `https://github.com/huyun-cs/Decoupled_NER`

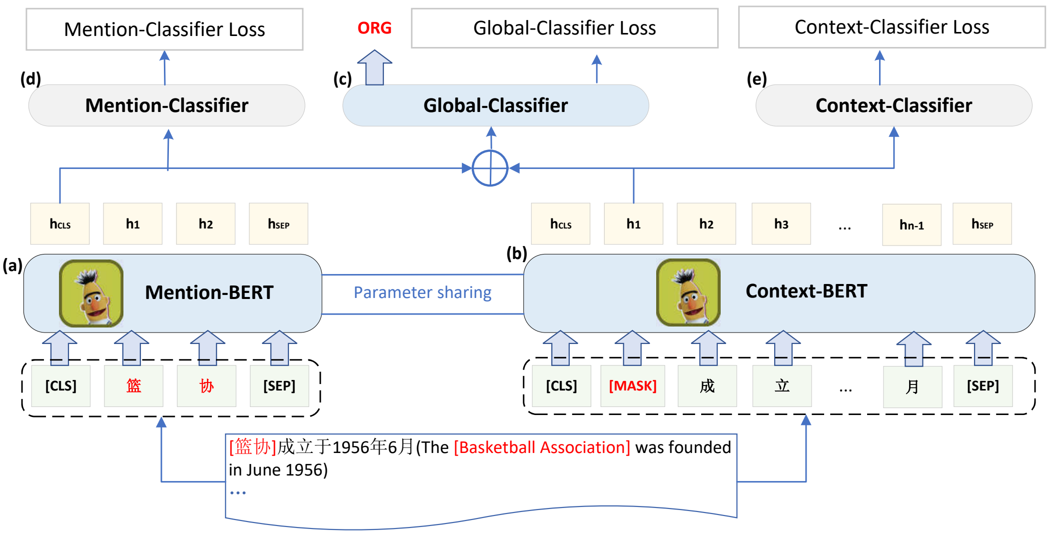

Figure 1: Model architecture. The model consists of a Mention-BERT, a Context-BERT, and a Global-Classifier. The input sentence, "篮协成立于1956年6月" (*the Basketball Association was founded in June 1956*), will first be converted into a 〈MENTION, CONTEXT〉 pair: 〈"篮协" (*Basketball Association*), "`[MASK]`成立于1956年6月" (`[MASK]` *was founded in June 1956*). Then the mention and the context act as the input of the Mention-BERT and the Context-BERT respectively. The outputs of the two BERTs will be concatenated and passed to the Global-Classifier to obtain the final tag prediction (ORG).

is an entity mention in the input sentence, and the output is the representation of the mention. The architecture of the Mention-BERT is the same as the original BERT, which is a multi-layer bidirectional Transformer encoder. As shown in Figure 1(a), given an entity mention $m = \{x_i, x_{i+1}, ..., x_{i+k}\}$, we first add two special tokens (`[CLS]` and `[SEP]`) to the beginning and the end of it, and take the output corresponding to the `[CLS]` token as representation h_m of the mention:

$$h_m = \text{Mention-BERT}(m) \qquad (2)$$

Context-BERT. The Context-BERT aims to encode the context around an entity mention. It has the same architecture as the Mention-BERT. The input c is just the context of the candidate mention, where the mention is replaced by a special `[MASK]` token. The output corresponding to the `[MASK]` token position is used as a representation for the context, denoted as h_c:

$$h_c = \text{Context-BERT}(c) \qquad (3)$$

As an example, in Figure 1(b), we have $h_c = h_1$. Note that at inference time we use only one `[MASK]` even for multi-token entities, as the Context-BERT are not allowed to not use any information of the mention.

Global-Classifier. The Global-Classifier determines the input mention's tag by considering both the mention representation and the context representation. In the implementation, we concatenate the output of Mention-BERT h_m and the output of Context-BERT h_c and pass them into a FFN:

$$y_g = softmax(W_g \cdot [h_m : h_c] + b_g) \qquad (4)$$

where W_g, b_g are parameters of the Global-Classifier, and y_g is the final prediction.

3.2 Two-stage Training

Pre-trained language models such as BERT aim to model general patterns of language and treats entity and non-entity words indiscriminately. It is reasonable to expect that such models will not generate a perfect representation for the NER task. To better utilize external heterogeneous data for the NER task, we design a two-stage training framework: (1) pre-training the Mention-BERT and the Context-BERT on entity dictionaries and distantly supervised data, and (2) training the unified model on human-annotated data.

Heterogeneous Training Data. Despite the limited size of human-annotated data for NER, we can easily collect large-scale entity dictionaries and unlabeled text corpora, and hence generate distantly

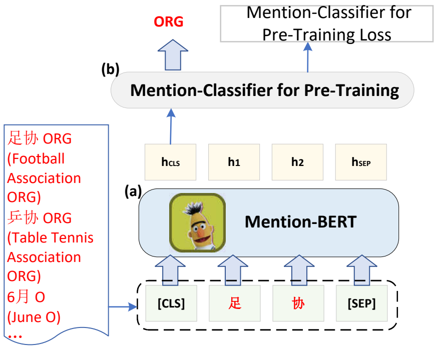

Figure 2: Pre-training process of the Mention-BERT. The Mention-BERT is pre-trained on the entity dictionary using a label classification task. For example, we try to predict that "足协" (*Football Association*) on its own is an organization.

supervised data. For dictionary data, the text often contains rich entity structure information. For example, a person name often consists of the First name and the Last name. For distantly supervised data, the text often contains rich context information, while has high noise. The most common mistakes are wrong labels and wrong boundaries. As a result, these data are not suitable to be directly incorporated for NER. However, they can be naturally used as data for pre-training to learn high-coverage and task-aware representations of entity mentions and contexts. On the one hand, previous research showed that further pre-training BERT to do language modeling on in-domain corpus could improve the performance of downstream tasks (Gururangan et al., 2020). On the other hand, either the entity or the context itself can be a strong indicator of entity types.

Mention-BERT Pre-Training. To better capture the regularity information of entities, the Mention-BERT is pre-trained on entity dictionaries. As shown in Figure 2, we add a feed forward classifier denoted as *Mention-Classifier for Pre-Training* on top of the Mention-BERT. The task is to classify each input term into the most probable label according to the dictionaries. For example, the output for the term "足协" (*Football Association*) should be ORG. Besides, to empower the model to learn discriminative representations for non-entity terms as well, we sample items from a common dictionary that have never been seen in any one of the entity dictionaries, and assign an O label to them.

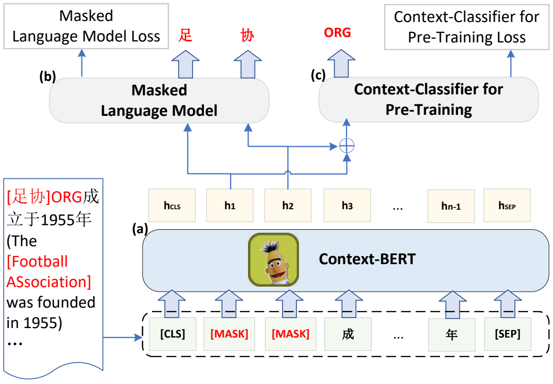

Figure 3: Pre-training process of the Context-BERT. The model is trained on a masked language modeling task and a label classification task simultaneously. For example, we train the model to predict the masked tokens as "足协" (*Football Association*) and the entity label as ORG, given only the context "`[MASK]`成立于1955" (`[MASK]` *was founded in 1955*).

Context-BERT Pre-Training. As shown in Figure 3, the Context-BERT is pre-trained on distantly supervised data with a hybrid task of masked language modeling and entity label prediction. For each input sentence, we pick one entity mention in it at each time and replace all tokens in it with `[MASK]` tokens. Given only the context with the mention masked out, the model is trained to predict both the masked tokens along with the entity label. We also randomly pick some non-entity regions and assign O labels to them. To this end, we use two classifiers, namely the *Masked Language Model* and the *Context-Classifier for Pre-Training*. The Masked Language Model is the same as in the original BERT. The Context-Classifier for Pre-Training is fed with the average pooling of the Context-BERT's outputs for all masked tokens.

Unified-Training. After pre-training, we perform the unified-training, in which the pre-trained Mention-BERT and Context-BERT are put together and further trained on human annotated data. To construct training examples, we iterate over all entity mentions in the annotated sentences and obtain pairs of ⟨MENTION, CONTEXT⟩ as the input of our model (see Figure 1). We also select non-entity regions as O. Given the correct label y, we define the loss of the Global-Classifier L_g as follows:

$$L_g = CE(y_g, y) \qquad (5)$$

where CE is the cross-entropy loss.

Furthermore, to avoid catastrophic forgetting for the pre-trained Mention-BERT and Context-BERT in the unified-training, we also add two auxiliary feed forward classifiers on top of Mention-BERT and Context-BERT, denoted as *Mention-Classifier* and *Context-Classifier* respectively (see (d) and (e) in Figure 1). Both of them have the same structure and objective as the Global-Classifier except input:

$$y_m = softmax(W_m \cdot h_m + b_m) \tag{6}$$

$$y_c = softmax(W_c \cdot h_c + b_c) \tag{7}$$

where W_m, b_m, W_c, b_c are parameters of the Mention-Classifier and the Context-Classifier, and y_m, y_c are the respective predictions. We define losses for the two classifiers as follows:

$$L_m = CE(y_m, y) \tag{8}$$

$$L_c = CE(y_c, y) \tag{9}$$

The final loss L of our model at the unified-training stage has three parts:

$$L = L_g + \alpha L_m + \beta L_c \tag{10}$$

where $\alpha, \beta \in [0, 1]$ are hyper-parameters.

4 Experiment

4.1 Dataset

We evaluate our methods on three Chinese NER datasets: OntoNotes 4.0 (Weischedel et al., 2013), MSRA (Levow, 2006), Weibo NER (Peng and Dredze, 2015; He and Sun, 2017). OntoNotes and MSRA are collected from newswire text, and Weibo NER is from social media text [2]. The detail of the datasets is shown in Table 1. At the unified-training stage, we treat all labeled entities in the training dataset as entity mentions. To obtain non-entity mentions, we take (1) all words and phrases labeled as noun by the LTP [3] (Che et al., 2020) lexicon tool, and (2) all words and phrases with an edit distance less than one to any of the entity mentions. During the test time, we first use LTP and SoftLexicon (Ma et al., 2020) model to obtain all regions of candidate entities. Then we use our model to predict the final label for each region.

Dataset	**Type**	**Train**	**Dev**	**Test**
OntoNotes	Sentence	15.7k	4.3k	4.3k
	Char	491.9k	200.5k	208.1k
MSRA	Sentence	46.4k	-	4.4k
	Char	2169.9k	-	172.6k
Weibo	Sentence	1.4k	0.27k	0.27k
	Char	73.8k	14.5k	14.8k

Table 1: Statistics of the datasets.

4.2 Pre-training Corpora

Entity Dictionary. There are four types of entity in our experiment: PER, ORG, GPE, and LOC. We extend the dictionary used in Ding et al. (2019) with more gazetteers collected from Sougou Dictionary [4] and Baidu Dictionary [5]. Finally, our gazetteer contains 50k person names, 143k organization names, 43k geopolitical entities, and 33k location names (see Appendix 1.1).

Distantly Supervised Data. The entity dictionary above is used to match unannotated sentences to obtain distantly supervised data. For OntoNotes and MSRA dataset, we collect news documents on the People's Daily [6] published from 1949 to 2010. For the Weibo dataset, we use the Weibo unannotated data from Peng and Dredze (2015). Finally, we obtain 893k sentences of distantly supervised data for news and 837k for Weibo.

4.3 Training Setting

Some hyper-parameters for training can be found in Appendix 1.2. We set the $\alpha = 0.5, \beta = 0.5$ in unified training through experiments. To better utilize the common knowledge of the Mention-BERT and Context-BERT, and also to reduce the model size, the parameters of Mention-BERT and Context-BERT are shared. We do not share the parameters of each classifier, because the label sets and output dimensions of the classifiers may be different across the two-stage training.

4.4 Baselines

We use the following models as baselines:

BiLSTM-CRF from Lample et al. (2016), which is a classical baseline for NER.

---

[2] For Weibo dataset, we only focus on the subset of entities labeled as NAM, as the criteria for entity definition are the same with OntoNotes and MSRA.

[3] http://ltp.ai/

[4] https://pinyin.sogou.com/dict/

[5] https://shurufa.baidu.com/dict

[6] http://paper.people.com.cn

Lattice LSTM from Zhang and Yang (2018), which uses a dictionary and word embedding to enhance character-based Chinese NER model.

BERT-NER from Devlin et al. (2019), which uses the outputs from the last layer of BERT model as feature representations, and does token classification to extract entity.

Incomplete-NER from Jie et al. (2019), which is based on BERT-CRF and uses cross-validation to estimate the distribution of missing labels in distant supervision [7].

MRC-NER from Li et al. (2020b), which considers NER as machine reading comprehension.

SoftLexicon from Ma et al. (2020), which proposes a simple but effective method for incorporating the word lexicon into the character representations in Chinese NER.

FLAT from Li et al. (2020a), which uses transformer to consider the relation between every character and word in the sentence.

ERNIE from Sun et al. (2019), which enhances BERT through knowledge integration by using a entity-level masked LM task and more raw text from the Web resources.

CoFEE from Xue et al. (2020), which proposes a NER-specific pre-training framework to inject coarse-to-fine automatically mined entity knowledge into pre-trained models.

4.5 Main Results

Following the evaluation metrics in previous work, entity-level (exact entity match) standard micro Precision (P), Recall (R), and F1 score are used to evaluate the results.

Table 2 presents the comparison between our model and baseline models. We can observe that our decoupled model with two-stage pre-training significantly outperforms recent models, establishing a new state-of-the-art for supervised NER. For OntoNotes, our model outperforms the SoftLexicon model by +1.51% in terms of F1. For Chinese MSRA, the proposed method outperforms the FLAT model. We also improve the F1 from 70.94% to 72.64% on Weibo dataset. We can also see that the Mention-BERT pre-trained on entity dictionary outperforms the plain decoupled model without two-stage pre-training by 0.89% in OntoNotes, 0.52% in MSRA, and 1.55% in Weibo respectively. These results show the effectiveness of Mention pre-training for the NER task. The results also show that Context-pretraining can improve performance (0.46% in OntoNote, 0.34% in MSRA, and 0.57% in Weibo). Moreover, further pre-training the Context-BERT based on Mention BERT using distantly supervised data can lead to a performance gain in F1 score(0.89% in OntoNote, 0.52% in MSRA, and 1.55% in Weibo).

OntoNotes			
	P	**R**	**F**
BiLSTM-CRF (Lample et al., 2016)	68.79	60.35	64.30
Latice-LSTM (Zhang and Yang, 2018)	76.35	71.56	73.88
BERT-NER (Devlin et al., 2019)	78.01	80.35	79.16
Incomplete-NER (Jie et al., 2019)	79.18	81.24	80.20
MRC (Li et al., 2020b)	82.98	81.25	82.11
SoftLexicon (Ma et al., 2020)	83.41	82.21	82.81
FLAT (Li et al., 2020a)	-	-	81.82
CoFEE (Xue et al., 2020)	82.50	82.78	82.64
Decoupled model	83.79	83.06	83.43
+ Mention Pre-train	84.34	83.54	83.93
+ Context Pre-train	84.28	83.51	83.89
+ Mention and Context Pre-train	**84.92**	**83.72**	**84.32**
MSRA			
	P	**R**	**F**
BiLSTM-CRF (Lample et al., 2016)	90.74	86.96	88.81
Latice-LSTM (Zhang and Yang, 2018)	93.57	92.79	93.18
BERT-NER (Devlin et al., 2019)	94.97	94.62	94.80
Incomplete-NER (Jie et al., 2019)	95.00	94.83	94.91
ERNIE (Sun et al., 2019)	-	-	95.0
MRC (Li et al., 2020b)	96.18	95.12	95.75
SoftLexicon (Ma et al., 2020)	95.75	95.10	95.42
FLAT (Li et al., 2020a)	-	-	96.09
Decoupled model	96.65	94.56	95.59
+ Mention Pre-train	96.67	**95.24**	95.95
+ Context Pre-train	96.67	95.20	95.93
+ Mention and Context Pre-train	**97.00**	95.23	**96.11**
Weibo			
	P	**R**	**F**
BiLSTM-CRF (Lample et al., 2016)	-	-	46.11
Latice-LSTM (Zhang and Yang, 2018)	-	-	53.04
BERT-NER (Devlin et al., 2019)	-	-	65.77
Incomplete-NER (Jie et al., 2019)	-	-	66.78
SoftLexicon (Ma et al., 2020)	-	-	70.94
Decoupled model	**72.81**	69.44	71.09
+ Mention Pre-train	70.35	**73.61**	71.94
+ Context Pre-train	71.54	71.78	71.66
+ Mention and Context Pre-train	72.14	73.14	**72.64**

Table 2: Results on the three datasets.

---

[7] We use the code from `https://github.com/ZhuiyiTechnology/AutoIE`. We combine human annotated data and distantly supervised data of equal size for training

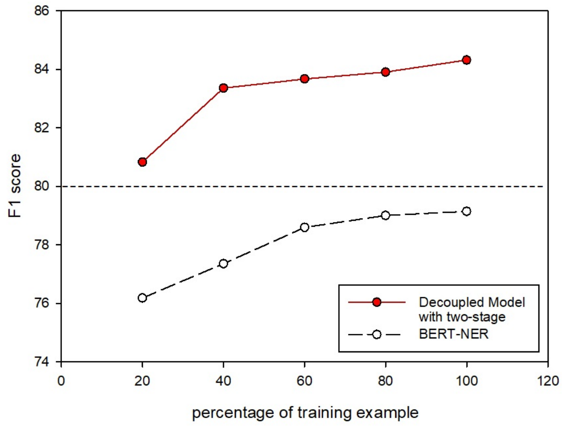

Figure 4: Model performance on OntoNotes with different proportions of hand-annotated training data size.

5 Analysis

5.1 Effect of Introducing External Data

In our experiments, it is not immediately clear which part is responsible for the final improvement: can it be the decoupled model or more additional data or both? To answer this question and show our model design can better utilize the heterogeneous corpus, we choose BERT-NER and SoftLexicon as base models to explore the effect of external data. For each base model, we experiment on two settings. First, we simply expand the training dataset by adding entity dictionary data and distantly supervised data. Second, we adopt a two-stage training strategy similar to the methods in Section 3.2, where we use the large external data to further pre-train the BERT part of BERT-NER and SoftLexicon, and then fine-tune the whole models on human-annotated data. The results are shown in Table 3. Our decoupled model achieves the best results. We can see a large performance drop when directly incorporating external training data in BERT-NER and SoftLexicon, as the distantly supervised data are noisy and its big size is unbalanced with the human-annotated data. Unexpectedly, the two base models also perform worse in the two-stage training setting. We suppose that the pre-training task of span classification is not suitable for the sequence labeling task.

5.2 Effect of Human-annotated Data Scale

To compare performances under different numbers of human-annotated training sentences, we randomly select different numbers of training sentences for training on the OntoNotes dataset.

As shown in Figure 4, our model has better performance than the BERT-NER model, which shows the effectiveness of our methods in small training data. Surprisingly, the results also show that in small data size (20% training data), our model also outperforms the BERT-NER model with full data size, which shows that our model requires less sentence-level annotated data compared with the original BERT-NER model. In addition to the model structure and external data, there are two other factors that lead to greater improvement. First, the 20% training data still contain over 3k examples in the news domain. Second, we leverage the mention boundary prediction from LTP, which provide high-quality candidates.

OntoNotes			
	P	**R**	**F**
BERT-NER	78.01	80.35	79.16
BERT-NER with mixed data	60.24	45.67	51.95
BERT-NER with two-stage	74.71	80.83	77.64
SoftLexicon	83.41	82.21	82.81
SoftLexicon with mixed data	73.38	43.41	54.55
SoftLexicon with two-stage	82.78	81.84	82.30
Decoupled model with two-stage	**84.92**	**83.72**	**84.32**
MSRA			
	P	**R**	**F**
BERT-NER	94.97	94.62	94.80
BERT-NER with mixed data	73.27	53.93	62.13
BERT-NER with two-stage	95.61	93.06	94.32
SoftLexicon	95.75	95.10	95.42
SoftLexicon with mixed data	77.64	54.08	63.75
SoftLexicon with two-stage	95.04	94.38	94.70
Decoupled model with two-stage	**97.00**	**95.23**	**96.11**
Weibo			
	P	**R**	**F**
BERT-NER	-	-	65.77
BERT-NER with mixed data	49.67	31.02	38.19
BERT-NER with two-stage	59.85	70.83	64.87
SoftLexicon	-	-	70.94
SoftLexicon with mixed data	51.34	33.12	40.26
SoftLexicon with two-stage	71.36	62.50	66.64
Decoupled model with two-stage	**72.14**	**73.14**	**72.64**

Table 3: Results of models using external data.

We also experiment on an even smaller training data size (only 1k sentences). In Table 4, we can see that our model performs better than BERT-NER on all the datasets.

5.3 Effect of Model Parameter Sharing

In practice, we share our model parameters of Mention-BERT and Context-BERT. In Table 7, we can see that the model with parameter sharing slightly outperforms the model without parameter sharing. A possible reason is that common knowl-

OntoNotes			
	P	**R**	**F**
BERT-NER	67.61	76.39	71.73
Decoupled model with two-stage	**82.82**	**68.78**	**75.15**
MSRA			
	P	**R**	**F**
BERT-NER	84.59	88.20	86.36
Decoupled model with two-stage	**93.85**	**83.04**	**88.12**
Weibo			
	P	**R**	**F**
BERT-NER	59.51	65.28	62.26
Decoupled model with two-stage	**66.04**	**65.74**	**65.89**

Table 4: Model performance on 1k human annotated training data.

	OntoNotes	MSRA	Weibo
Sharing	**84.32**	**96.11**	**72.64**
Without Sharing	84.07	96.05	72.07

Table 5: The effect of parameter sharing.

edge about NER is shared between the two BERTs.

5.4 Case Study

为改善[九江]$_{\text{GPE}}$投资环境	
In order to improve the investment environment of [Jiujiang]$_{\text{GPE}}$	
BERT-NER	LOC
Golden / Our model	GPE
在[东盟]$_{\text{ORG}}$成立30周年之际	
On the 30th anniversary of the [Association of Southeast Asian Nations]$_{\text{ORG}}$	
BERT-NER	GPE
Golden / Our model	ORG

Table 6: Case study. Our model refers to the decoupled model with two-stage training. The text in brackets is the candidate mention, followed by the golden label. Predicted labels in red denote wrong answer.

Table 6 shows two cases from OntoNotes. In the first example, the BERT-NER model misclassifies "九江"(*Jiujiang*) as LOC. We find that "九江" (*Jiujiang*) is in our dictionaries but the label is GPE. Benefited from incorporating entity dictionaries into pre-training, our model can correctly recognize "九江" (*Jiujiang*) as a city. In the second example, the BERT-NER misclassifies "东盟"(*Association of Southeast Asian Nations*) as GPE. We find that distantly supervised data contains the sentence, "在上海合作组织成立5周年大会上"(*At the 5th anniversary meeting of the Shanghai Cooperation Organization*) and the context of "上海合作组织" (*Shanghai Cooperation Organization*) is similar to "东盟" (*Association of Southeast Asian Nations*). The label of the "上海合作组织" (*Shanghai Cooperation Organization*) is GPE. With the context information from Context-BERT, our model can obtain the correct answer of "东盟" (*Association of Southeast Asian Nations*).

6 Related Work

6.1 Supervised NER Models

NER models trained on human-annotated data often achieve appropriate performance. Sequence labeling methods are widely used in NER. Traditional methods use the CRF model to solve the NER task (Lafferty et al., 2001). With the advantages of eliminating feature engineering and significant performance improvement, neural network models become prevalent in NER research, e.g., the models based on FFN (Collobert et al., 2011), CNN (Ma and Hovy, 2016), LSTM (Lample et al., 2016), and pre-trained language model (Devlin et al., 2019). Recent work also propose different ways to model the NER task other than sequence labeling, such as machine reading comprehension (Li et al., 2020b), dependency parsing (Yu et al., 2020), span classification (Sohrab and Miwa, 2018). Generally, these approaches have achieved promising results but heavily rely on human-annotated data.

6.2 Enhancing NER with External Data

Entity dictionaries or gazetteers have long been regarded as an easily-obtainable and useful resource for NER. Previous methods commonly incorporated gazetteers as additional features (Ghaddar and Langlais, 2018; Al-Olimat et al., 2018; Liu et al., 2019a; Ding et al., 2019; Lin et al., 2019; Rijhwani et al., 2020). For languages without explicit word boundaries, such as Chinese, incorporating a universal dictionary with common words besides gazetteers can be further helpful for NER (Zhang and Yang, 2018; Liu et al., 2019b; Sui et al., 2019; Gui et al., 2019b,a; Ma et al., 2020; Li et al., 2020a; Jia et al., 2020). Dictionaries can also be used to construct distantly supervised data from unlabeled corpora. Previous work on reducing the noise in distantly supervised data include new labeling schemes (Shang et al., 2018), reinforcement learning (Yang et al., 2018), cross-training (Jie et al., 2019), positive unlabeled learning (Peng et al., 2019), HMM (Lison et al., 2020), consensus network (Lan et al., 2020a). In other NLP tasks, such as relation extraction, few works have exploited using both human annotated data and distantly supervised data together (Angeli et al., 2014; Beltagy et al., 2019). Compared with previous works, our work focus on designing a new model architecture and training approaches to better exploit the heterogeneous data in NER task.

6.3 Two-stage Training Paradigm for NLP

Recently, large-scale pre-trained language models, such as BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018), are widely used and yield state-of-the-art performances in many NLP tasks. These two-stage methods allow using large-scale unlabeled data in pre-training and small labeled data in fine-tuning. In order to adapt to specific tasks or domain, variants of BERT are proposed including small and practical BERT (Tsai et al., 2019; Lan et al., 2020b; Jiao et al., 2020), domain adaptive BERT (Yang et al., 2019a; Gururangan et al., 2020), and task adaptive BERT (Sun et al., 2019; Xue et al., 2020; Jia et al., 2020). Our work performs further pre-training on BERT and proposes task-aware training objectives to improve NER.

7 Conclusion

In this work, we focus on fully exploiting heterogeneous corpus for NER. The corpus consists of entity dictionaries, distantly supervised instances, and human-annotated instances. We propose a decoupled NER model with two-stage training. The model first learns appropriate task-aware representations in pre-training, from large-scale context-deficient dictionaries and noisy distantly supervised data. Then after unified-training, the model can predict entity labels according to both the mention and the context information. Experimental results show our method achieves better performance than previous state-of-the-art methods on three Chinese datasets. In the future, we will exploit more types of data, such as knowledge bases, and extend our approach to other languages.

References

Hussein Al-Olimat, Krishnaprasad Thirunarayan, Valerie Shalin, and Amit Sheth. 2018. Location name extraction from targeted text streams using gazetteer-based statistical language models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1986–1997, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D. Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1556–1567. ACL.

Iz Beltagy, Kyle Lo, and Waleed Ammar. 2019. Combining distant and direct supervision for neural relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1858–1867. Association for Computational Linguistics.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Wanxiang Che, Yunlong Feng, Libo Qin, and Ting Liu. 2020. N-ltp: A open-source neural chinese language technology platform with pretrained models. *arXiv preprint arXiv:2009.11616*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. 2019. A neural multi-digraph model for Chinese NER with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1462–1467, Florence, Italy. Association for Computational Linguistics.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019a. Cnn-based chinese ner with lexicon rethinking. IJCAI.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuanjing Huang. 2019b. A lexicon-based graph neural network for chinese NER. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1040–1050. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360. Association for Computational Linguistics.

Hangfeng He and Xu Sun. 2017. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 713–718. Association for Computational Linguistics.

Chen Jia, Yuefeng Shi, Qinrong Yang, and Yue Zhang. 2020. Entity enhanced BERT pre-training for chinese NER. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6384–6396. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 4163–4174. Association for Computational Linguistics.

Zhanming Jie, Pengjun Xie, Wei Lu, Ruixue Ding, and Linlin Li. 2019. Better modeling of incomplete annotations for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 729–734, Minneapolis, Minnesota. Association for Computational Linguistics.

John D. Lafferty, Andrew Mccallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Eighteenth International Conference on Machine Learning*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Ouyu Lan, Xiao Huang, Bill Yuchen Lin, He Jiang, Liyuan Liu, and Xiang Ren. 2020a. Learning to contextually aggregate multi-source supervision for sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2134–2146. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020b. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117. Association for Computational Linguistics.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. FLAT: chinese NER using flat-lattice transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6836–6842. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5849–5859. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, Bin Dong, and Shanshan Jiang. 2019. Gazetteer-enhanced attentive neural networks for named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6231–6236. Association for Computational Linguistics.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1518–1533. Association for Computational Linguistics.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019a. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307, Florence, Italy. Association for Computational Linguistics.

Wei Liu, Tongge Xu, Qinghua Xu, Jiayu Song, and Yueran Zu. 2019b. An encoding strategy based word-character LSTM for Chinese NER. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2379–2389, Minneapolis, Minnesota. Association for Computational Linguistics.

Ruotian Ma, Minlong Peng, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2020. Simplify the usage of lexicon in Chinese NER. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5951–5960, Online. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2409–2419. Association for Computational Linguistics.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Shruti Rijhwani, Shuyan Zhou, Graham Neubig, and Jaime G. Carbonell. 2020. Soft gazetteers for low-resource named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8118–8123. Association for Computational Linguistics.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064. Association for Computational Linguistics.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2019. Leverage lexical knowledge for chinese named entity recognition via collaborative graph network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3828–3838. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical bert models for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3623–3627.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Mengge Xue, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-fine pretraining for named entity recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6345–6354. Association for Computational Linguistics.

Huiyun Yang, Shujian Huang, XIN-YU DAI, and CHEN Jiajun. 2019a. Fine-grained knowledge fusion for sequence labeling domain adaptation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4188–4197.

YaoSheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2159–2169. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564. Association for Computational Linguistics.

8 Appendices

8.1 Entity Dictionary

	OntoNotes	**MSRA**	**Weibo**
Coverage rate	71.49	78.81	61.95
Conflict rat	85.35	93.13	83.63

Table 7: Coverage rate and conflict rate of entity dictionary. We use the entity dictionary to directly match the test dataset, and compute the coverage rate and conflict rate. The coverage rate is the number of entities both in the dictionary and in the test dataset divided by the number of entities in the test dataset. The conflict rate is the number of entities with inconsistent labels divided by the number of entities both in the dictionary and in test dataset.

8.2 Hyper-parameter Values

Mention-BERT pre-training			
learning rate			2e-5
batch size			128
epoch			10
Context-BERT pre-training			
learning rate			2e-5
batch size			64
epoch			10
Unified-training			
	OntoNotes	**MSRA**	**Weibo**
learning rate	5e-6	5e-6	5e-6
batch size	64	64	32
epoch	4	4	10

Table 8: Hyper-parameter values

Discriminative Reasoning for Document-level Relation Extraction

Wang Xu[1]**, Kehai Chen**[2] **and Tiejun Zhao**[1]
[1]Harbin Institute of Technology, Harbin, China
[2]National Institute of Information and Communications Technology, Kyoto, Japan
`xuwang@hit-mtlab.net`, `khchen@nict.go.jp`, `tjzhao@hit.edu.cn`

Abstract

Document-level relation extraction (DocRE) models generally use graph networks to implicitly model the reasoning skill (i.e., pattern recognition, logical reasoning, coreference reasoning, etc.) related to the relation between one entity pair in a document. In this paper, we propose a novel discriminative reasoning framework to explicitly model the paths of these reasoning skills between each entity pair in this document. Thus, a discriminative reasoning network is designed to estimate the relation probability distribution of different reasoning paths based on the constructed graph and vectorized document contexts for each entity pair, thereby recognizing their relation. Experimental results show that our method outperforms the previous state-of-the-art performance on the large-scale DocRE dataset. The code is publicly available at `https://github.com/xwjim/DRN`.

Pattern Recognition:
[1] *Me Musical Nephews* is a *1942* one-reel animated cartoon directed by Seymour Kenitel.
Relation: publication_data **Supporting Evidence:** 1

Common-sense Reasoning:
[1] *William* and *Adelaide* had four children.
Relation: spouse **Supporting Evidence:** 1

Logical Reasoning:
[1] Elias Brown (May 9, 1793– July 7, 1857) was a *U.S.* Representative from Maryland. [2] Born near *Baltimore*, Maryland, Brown attended the common schools. ... [7] He died near *Baltimore*, Maryland, and is interred in a private cemetery near Eldersburg, Maryland.
Relation: Country **Supporting Evidence:** 1,7

Coreference Reasoning:
[1] *Dwight Tillery* is an American politician of the Democratic Party who is active in local politics of Cincinnati, Ohio. ... [3] He also holds a law degree from the *University of Michigan Law School*.
Relation: educated_at **Supporting Evidence:** 1,3

Figure 1: An example of different reasoning types. Different reasoning types have different reasoning processing.

1 Introduction

Document-level relation extraction (DocRE) aims to extract relations among entities within a document which requires multiple reasoning skills (i.e., pattern recognition, logical reasoning, coreference reasoning, and common-sense reasoning) (Yao et al., 2019). Generally, the input document is constructed as a structural graph-based on syntactic trees, coreference or heuristics to represent relation information between all entity pairs (Nan et al., 2020; Zeng et al., 2020; Xu et al., 2021). Thus, graph neural networks are applied to the constructed structural graph to model these reasoning skills. After performing multi-hop graph convolution, the feature representations of two entities are concatenated to recognize their relation by the classifier, achieving state-of-the-art performance in the DocRE task (Zeng et al., 2020; Xu et al., 2021). However, it is yet to be seen whether modeling these reasoning skills implicitly is competitive with the intuitive reasoning skills between one entity pair in this document.

Figure 1 shows four kinds of reasoning skills for entity pairs in the DocRE dataset (Yao et al., 2019). First, take two entity pairs {"*Me Musical Nephews*", "*1942*"} and {"*William*", "*Adelaide*"} as examples, the intra-sentence reasoning concerns about the mentions inside the sentence, for example, "*Me Musical Nephews*" and "*1942*" for pattern recognition, and "*William*" and "*Adelaide*" for the common-sense reasoning. Also, the logical reasoning for entity pair {"*U.S.*", "*Baltimore*"} requires the reason path from "*U.S.*"→"Maryland" (bridge entity)→"*Baltimore*" while the coreference reasoning for entity pair {"*Dwight Tillery*", "*University of Michigan Law School*"} pays attention to the reason path from "*Dwight Tillery*"→"*He*" (reference word)→"*University of Michigan Law*

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1653–1663
August 1–6, 2021. ©2021 Association for Computational Linguistics

School". However, the advanced DocRE models generally use the universal multi-hop convolution networks to model these reasoning skills implicitly and do not consider the above intuitive reasoning skills explicitly, which may hinder the further improvement of DocRE.

To this end, we propose a novel discriminative reasoning framework to explicitly model the reasoning processing of these reasoning skills, such as intra-sentence reasoning (including pattern recognition and common-sense reasoning), logical reasoning, and coreference reasoning. Specifically, inspired by Xu et al.'s meta-path strategy, we extract the reasoning paths of the three reasoning skills discriminatively from the input document. Thus, a discriminative reasoning network is designed to estimate the relation probability distribution of different reasoning paths based on the constructed graph and vectorized document contexts for each entity pair, thereby recognizing their relation. In particular, there are the probabilities of multiple reasoning skills for each candidate relation between one entity pair, to ensure that all potential reasoning skills can be considered in the inference. In summary, our main contributions are as follows:

- We propose a discriminative reasoning framework to model the reasoning skills between two entities in a document. To the best of our knowledge, this is the first work to model different reasoning skills explicitly for enhancing the DocRE.
- Also, we introduce a discriminative reasoning network to encode the reasoning paths based on the constructed heterogeneous graph and the vectorized original document, thereby recognizing the relation between two entities by the classifier.
- Experimental results on the large-scale DocRE dataset show the effectiveness of the proposed method, especially outperform the recent state-of-the-art DocRE model.

2 Discriminative Reasoning Framework

In this section, we propose a novel discriminative reasoning framework to model different reasoning skills explicitly to recognize the relation between each entity pair in the input document. The discriminative reasoning framework contains three parts: definition of reasoning paths, modeling reasoning discriminatively, and multi-reasoning based relation classification.

2.1 Definition of Reasoning Path

Formally, given one unstructured document comprised of N sentences D=$\{s_1, s_2, \cdots, s_N\}$, each sentence is a sequence of words $s_n = \{s_n^1, s_n^2, \cdots, s_n^J\}$ with the length J_n=$|s_n|$. The annotations include concept-level entities $\varepsilon = \{e_i\}_{i=1}^P$ as well as multiple occurrences of each entity under the same phrase of alias $e_i = \{m_i^{s_k}\}_{k=1}^Q$ ($m_i^{s_k}$ denotes the mention of e_i which occur in the sentence s_k) and their entity type (i.e. locations, organizations, and persons). The DocRE aims to extract the relation between two entities in ε, namely $P(r|e_i, e_j, D)$. For the simplification of reason skills, we first combine both pattern recognition and common-sense reasoning as the intra-sentence reasoning because they generally perform reasoning inside the sentence. Consequently, the original four kinds of the reasoning skills (Yao et al., 2019) are further refined as three reasoning skills: intra-sentence reasoning, logical reasoning, and coreference reasoning. Inspired by Xu et al.'s work, we also use the meta-path strategy to extract reasoning path for each reason skill, thereby representing the above three reasoning skills explicitly. Specifically, meta-paths for different reasoning skills are defined as follows:

1) **Intra-sentence reasoning path**: It is formally denoted as PI_{ij}=$m_i^{s_1} \circ s_1 \circ m_j^{s_1}$ for one entity pair $\{e_i, e_j\}$ inside the same sentence s_1 in the input document D. $m_i^{s_1}$ and $m_j^{s_1}$ are mentions related to two entities, respectively. "$\circ$" denotes one reasoning step on the reasoning path from e_i to e_j.

2) **Logical reasoning path**: The relation between one entity pair $\{e_i, e_j\}$ from sentences s_1 and s_2 is indirectly established by the occurrence bridge entity e_l for the logical reasoning. The reasoning path can be formally as PL_{ij}= $m_i^{s_1} \circ s_1 \circ m_l^{s_1} \circ m_l^{s_2} \circ s_2 \circ m_j^{s_2}$.

3) **Coreference reasoning path**: A reference word refers to one of two entities e_i and e_j, which occur in the same sentence as the other entity. We simplify the condition and assume that there is a coreference reasoning path

when the entities occur in different sentences. The reasoning path can be formally as PC=$m_i^{s_1} \circ s_1 \circ s_2 \circ m_j^{s_2}$.

Note that there are no entities in the defined reasoning path compare to the meta-path defined in Xu et al.'s work. This difference is mainly due to the following considerations: i) the reason path pays more attention to the mentions and referred sentences; ii) entities generally are contained by mentions; iii) it makes modeling of path reasoning more simple.

2.2 Modeling Reasoning Discriminatively

Based on the defined reasoning paths, we decompose the DocRE problem into three reasoning sub-tasks: intra-sentence reasoning (IR), logical reasoning (LR), and coreference reasoning (CR). Next, we introduce modeling of three sub-tasks in detail:

Modeling Intra-Sentence Reasoning. Given one entity pair $\{e_i,\ e_j\}$ and its reasoning path PI_{ij} in the sentence s_1, the intra-sentence reasoning is modeled to recognize the relation between this entity pair based as follows:

$$\mathrm{R}_{PI}(r) = P(r|e_i, e_j, PI_{ij}, D). \tag{1}$$

Modeling Logical Reasoning. Given one entity pair $\{e_i,\ e_j\}$ and its reasoning path PL_{ij}, the logical reasoning is modeled to recognize the relation between this entity pair based as follows:

$$\mathrm{R}_{PL}(r) = P(r|e_i, e_j, PL_{ij}, D). \tag{2}$$

Since the e_l co-occur with the entity pair e_i and e_j respectively, the logical reasoning is further formally as follows:

$$\mathrm{R}_{PL}(r) = P(r|e_i, e_j, e_l, PI_{il} \circ PI_{lj}, D). \tag{3}$$

where $\circ$ denotes the connection of the paths.

Modeling Coreference Reasoning. Similarity, given one entity pair $\{e_i,\ e_j\}$ and its reasoning path PC_{ij}, the coreference reasoning is modeled to recognize the relation between this entity pair based as follows:

$$\mathrm{R}_{PC}(r) = P(r|e_i, e_j, PC_{ij}, D). \tag{4}$$

2.3 Multi-reasoning Based Relation Classification

In the DocRE task, one entity usually involves multiple relationships which rely on different reasoning types. Thus, the relation between one entity pair may be reasoned by multiple types of reasoning rather than one single reasoning type. Based on the proposed three reasoning sub-tasks, the relation reasoning between one entity pair is regarded as a multi-reasoning classification problem. Formally, we select the reasoning type with max probability to recognize the relation between each entity pair as follows:

$$P(r|e_i, e_j, D) = \max[\mathrm{R}_{PI}(r), \mathrm{R}_{PL}(r), \mathrm{R}_{PC}(r)]. \tag{5}$$

In addition, there are often multiple reason paths between two entities for one reasoning type. Thus, the classification probability in Eq.(5) can be rewritten as follows:

$$\begin{aligned} P(r|e_i, e_j, D) = \max[& \\ & \{\mathrm{R}_{PI_1}(r), \cdots, \mathrm{R}_{PI_K}(r)\}, \\ & \{\mathrm{R}_{PL_1}(r), \cdots, \mathrm{R}_{PL_K}(r)\}, \\ & \{\mathrm{R}_{PC_1}(r), \cdots, \mathrm{R}_{PC_K}(r)\}], \end{aligned} \tag{6}$$

where K is the number of reasoning paths for one reasoning skill, which is the same to each reasoning skill for simplicity. Note that all the entity pairs have at least one reasoning path from one of three defined reasoning sub-tasks. When the number of reasoning paths is greater than K for one reasoning sub-task, we choose the K first reasoning paths, otherwise we use the actual reasoning paths.

3 Discriminative Reasoning Network

In this section, we design a discriminative reasoning network (**DRN**) to model three defined reasoning sub-tasks for recognizing the relation between two entities in a document. Follow Zeng et al. and Zhou et al.'s work, we use two kinds of context representations (heterogeneous graph context representation and document-level context representation) to model different reasoning paths discriminatively in Eq.(1)-(4)

3.1 Heterogeneous Graph Context Representation

Formally, the embedding of each word $\mathbf{w}_e$ is concatenated with the embedding of its entity type $\mathbf{w}_t$ and the embedding of its coreference $\mathbf{w}_c$ as the representation of word $\mathbf{b}$=[$\mathbf{w}_e$:$\mathbf{w}_t$:$\mathbf{w}_c$]. These sequences of word representations are in turn fed into a bidirectional long short-term memory (BiLSTM) to vectorize the input

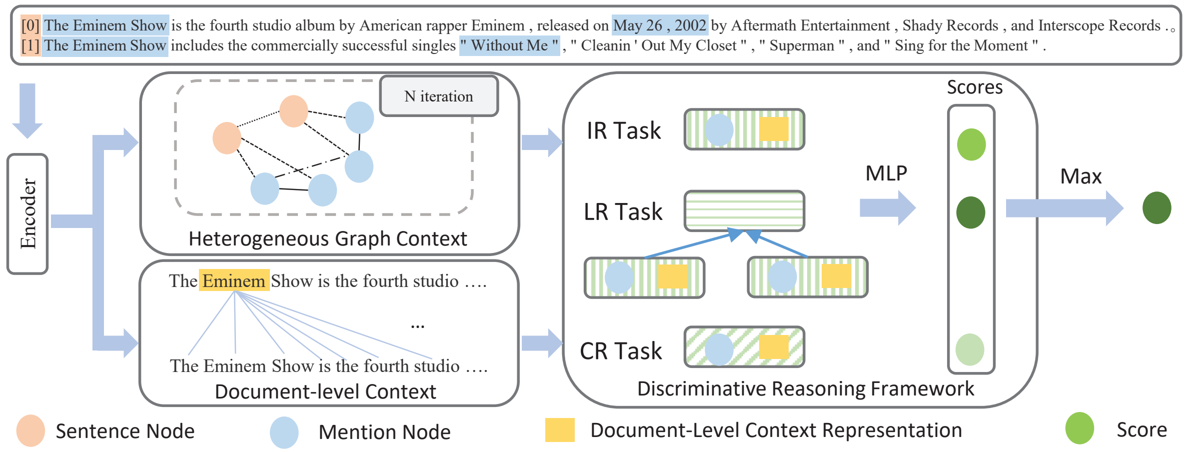

Figure 2: The overall architecture of DRN. First, A context encoder consumes the input document to get a contextualized representation of each word. Then the heterogeneous graph context representation and the document-level context representation are prepared as the input of the discriminative reasoning framework. Intra-sentence reasoning (IR) task, logical reasoning (LR) task and co-reference reasoning (CR) task are modeled explicitly and calculate the classification score respectively. Finally, the maximal score is selected as the output.

document D=$\{\mathbf{H}^1, \mathbf{H}^2, \cdots, \mathbf{H}^N\}$, where $\mathbf{H}^n = (\mathbf{h}_1^n, \mathbf{h}_2^n, \ldots, \mathbf{h}_{J_n}^n)$ and $\mathbf{h}_i^j$ denotes the hidden representation of the $i-th$ words of the $j-th$ sentence in the document. Similar to Zeng et al.'s work, we construct a heterogeneous graph which contains sentence node and mention node. There are four kinds of edges in the heterogeneous graph: sentence-sentence edge (all the sentence nodes are connected), sentence-mention edge (the sentence node and the mention node which resides in the sentence), mention-mention edge (all the mention nodes which are in the same sentence) and co-reference edge (all the mention nodes which refer to the same entity). Then we apply the graph-based DocRE method (Zeng et al., 2020) to encode the heterogeneous graph, based on which the heterogeneous graph context representation (**HGCRep**) are learned. The **HGCRep** of each mention node and sentence node $\mathbf{g}_n$ is formally denoted as:

$$\mathbf{g}_n = [\mathbf{v}_n : \mathbf{p}_n^1 : \mathbf{p}_n^2 : \cdots : \mathbf{p}_n^{l-1}], \tag{7}$$

where $\mathbf{g}_n \in \mathbb{R}^{d_1}$ and ":" is the concatenation of vectors and each of $\{\mathbf{p}_n^1, \mathbf{p}_n^2, \cdots, \mathbf{p}_n^{l-1}\}$ is learned by the multi-hop graph convolutional network (Zeng et al., 2020) and $\mathbf{v}_n$ is the initial representation of the n-th node extracted from $\mathbf{D}$. Finally, there is a heterogeneous graph representation G=$\{\mathbf{g}_1, \mathbf{g}_2, \cdots, \mathbf{g}_N\}$ including each mention nodes and sentence nodes.

3.2 Document-level Context Representation

In the DocRE task, these reasoning skills heavily rely on the original document context information rather than the heterogeneous graph context information. Thus, the existing advanced DocRE models use syntactic trees or heuristics rules to extract the context information (i.e., entities, mentions, and sentences) that is directly related to the relation between entity pairs. However, this approach destroys the original document structure, which is weak in modeling the reasoning between two entities for the DocRE task. Therefore, we use the self-attention mechanism (Vaswani et al., 2017) to learn a document-level context representation (**DLCRep**) $\mathbf{c}_n$ for one mention based on the vectorized input document $\mathbf{D}$:

$$\mathbf{c}_n = \text{softmax}(\frac{\mathbf{h}_j^n \mathbf{K}^\top}{\sqrt{d_{model}}})\mathbf{V}, \tag{8}$$

where $\mathbf{c}_n \in \mathbb{R}^{d_2}$ and $\{\mathbf{K}, \mathbf{V}\}$ are key and value matrices that are transformed from the vectorized input document $\mathbf{D}$ using a linear layer. Here, inspired by relation learning (Baldini Soares et al., 2019), we use the hidden state of the head word in one mention or one sentence to denote them for simplicity.

3.3 Modeling of Reasoning Paths

In this section, we use the concatenation operation to model the reasoning step on the reasoning path, thereby modeling the defined reasoning paths in Section 2.1 as the corresponding reasoning representations as follows:

1) For the intra-sentence reasoning path, both HGCReps and DLCReps of two mentions are concatenated in turn as a reasoning representation:

$$\alpha_{ij} = [\mathbf{g}_{m_i^{s_1}} : \mathbf{g}_{m_j^{s_1}} : \mathbf{c}_{m_i^{s_1}} : \mathbf{c}_{m_j^{s_1}}], \tag{9}$$

where $\alpha_{ij} \in \mathbb{R}^{2d_1+2d_2}$ and ":" is the concatenation of vectors.

2) For the logical reasoning path, both HGCReps of mention $m_i^{s_1}$ and $m_j^{s_2}$ and DLCReps of two mention pair $(m_i^{s_1}, m_l^{s_1})$ and $(m_j^{s_2}, m_l^{s_2})$ are concatenated as their reasoning representation:

$$\begin{aligned}\beta_{ij} = [&\mathbf{g}_{m_i^{s_1}} : \mathbf{g}_{m_j^{s_1}} : \\ &\mathbf{c}_{m_i^{s_1}} + \mathbf{c}_{m_l^{s_1}} : \mathbf{c}_{m_j^{s_2}} + \mathbf{c}_{m_l^{s_2}}],\end{aligned} \tag{10}$$

where $\beta_{ij} \in \mathbb{R}^{2d_1+2d_2}$.

3) For the coreference reasoning path, we connect both HGCReps of two mentions and DLCReps of two sentences are are concatenated in turn as their reasoning representation:

$$\gamma_{ij} = [\mathbf{g}_{m_i^{s_1}} : \mathbf{g}_{m_j^{s_2}} : \mathbf{c}_{s_1} : \mathbf{c}_{s_2}] \tag{11}$$

where $\gamma_{ij} \in \mathbb{R}^{2d_1+2d_2}$ and both $\mathbf{c}_{s_2}$ and $\mathbf{c}_{s_2}$ denote DLCReps for two sentences s_1 and s_2.

The learned reasoning representations α_{ij}, β_{ij}, and γ_{ij} is as the input to classifier to compute the probabilities of relation between e_i and e_j entities by a multilayer perceptron (MLP) respectively:

$$\begin{aligned}P(r|e_i, e_j, D) = \max[&\\ \text{sigmoid}(\text{MLP}_r(\alpha_{ij}),&\\ \text{sigmoid}(\text{MLP}_r(\beta_{ij}),&\\ \text{sigmoid}(\text{MLP}_r(\gamma_{ij})].&\end{aligned} \tag{12}$$

Similarly, when there are multiple reasoning paths between two entities for one reasoning type in Eq.6, Eq.12 is rewritten as follows:

$$\begin{aligned}P(r|e_i, e_j, D) = \max[&\\ \text{MLP}_r(\alpha_{ij}^1), \cdots, \text{MLP}_r(\alpha_{ij}^K),&\\ \text{MLP}_r(\beta_{ij}^1), \cdots, \text{MLP}_r(\beta_{ij}^K),&\\ \text{MLP}_r(\gamma_{ij}^1), \cdots, \text{MLP}_r(\gamma_{ij}^K)].&\end{aligned} \tag{13}$$

Also, the binary cross-entropy is used as training objection, which is the same as the advanced DocRE model (Yao et al., 2019).

4 Experiments

4.1 Data set and Setup

Hyperparameter	Value
Batch Size	12
Optimizer	AdamW
Learning Rate	1e-3
Activation Function	ReLU
Word Embedding Size	100
Entity Type Embedding Size	20
Coreference Embedding Size	20
Encoder Hidden Size	128
Dropout	0.5
Layers of GCN	2
Weight Decay	0.0001
Device	GTX 1080Ti

Table 1: Settings for DRN.

The proposed methods were evaluated on a large-scale human-annotated dataset for document-level relation extraction (Yao et al., 2019). DocRED contains 3,053 documents for the training set, 1,000 documents for the development set, and 1,000 documents for the test set, totally with 132,375 entities, 56,354 relational facts, and 96 relation types. More than 40% of the relational facts require reading and reasoning over multiple sentences. For more detailed statistics about DocRED, we recommend readers to refer to the original paper (Yao et al., 2019).

Following settings of Yao et al.'s work, we used the GloVe embedding (100d) and BiLSTM (128d) as word embedding and encoder. The number of the reasoning path for each task is set to 3. The learning rate was set to 1e-3 and we trained the model using AdamW (Loshchilov and Hutter, 2019) as the optimizer with weight decay 0.0001 under Pytorch (Paszke et al., 2017). For the BERT representations, we used uncased BERT-Based model (768d) as the encoder and the learning rate was set to $1e^{-5}$. For evaluation, we used F1 and Ign F1 as the evaluation metrics. Ign F1 denotes F1 score excluding relational facts shared by the training and development/test sets. In particular, the predicted results were ranked by their confidence and traverse this list from top to bottom by $F1$ score on development set, and the score value corresponding to the maximum $F1$ is picked as threshold θ. The hyper-parameter for the number of reasoning paths was tuned based on the development set. In addition, the results on the test set were evaluated through CodaLab[1]. Once a model is trained, we get the confidence scores for every triple example (subject,object,relation) as Eq.(12). We rank the predicted results by their confidence and traverse this list from top to bottom by F1 score on development set, the score value corresponding to the maximum F1 is picked as threshold θ. This threshold is used to control the number of extracted relational facts on the test set.

4.2 Baseline Systems

We reported the results of the recent graph-based DocRE methods as the comparison systems: GAT (Veličković et al., 2018), GCNN (Sahu et al., 2019), EoG (Christopoulou et al., 2019), AG-GCN (Guo et al., 2019), **LSR** (Nan et al., 2020), **GAIN** (Zeng et al., 2020), and **HeterGASN-Rec**(Xu et al., 2021). Moreover, pre-trained models like **BERT** (Devlin et al., 2019) has been shown impressive result on the DocRE task. Therefore, we also reported state-of-the-art graph-based DocRE models with pre-trained BERT$_{base}$ model, including **Two-Phase+BERT**$_{base}$ (Wang et al., 2019), **LSR+BERT**$_{base}$ (Nan et al., 2020), **GAIN+BERT**$_{base}$ (Zeng et al., 2020), **HeterGASN-Rec+BERT**$_{base}$ (Xu et al., 2021), and **ATLOP-BERT**$_{base}$ (Zhou et al., 2021).

4.3 Main Results

Table 2 presents the detailed results on the development set and the test set for the DocRE dataset. First, the proposed DRN model significantly outperformed the existing graph-based DocRE systems. Second, the proposed DRN model was superior to all the existing graph-based DocRE systems on the test set, validating that modeling reasoning discriminatively is more beneficial to DocRE than the original universal neural network way. Meanwhile, it also outperformed the best HeterGSAN-Rec model by 1.10 points in terms of F1, validating the effectiveness of our discriminative reasoning method. Third, for the comparisons with a pre-trained language model (BERT$_{base}$), F1 scores of the proposed DRN+BERT$_{base}$ model was higher than that of the existing graph-based DocRE ATLOP+BERT model systems with BERT$_{base}$ on the test set. In particular, our method (F1 61.37) was superior to the existing best ATLOP+BERT model (F1 61.30) in terms of F1, which is a new state-of-the-art result on the DocRE dataset.

Methods	Dev		Test	
	Ign F1	F1	Ign F1	F1
Existing DocRE Systems				
GCNN†	46.22	51.52	49.59	51.62
EoG†	45.94	52.15	49.48	51.82
GAT†	45.17	51.44	47.36	49.51
AGGCN†	46.29	52.47	48.89	51.45
LSR*	48.82	55.17	52.15	54.18
GAIN*	53.05	55.29	52.66	55.08
HeterGSAN-Rec*	54.27	56.22	53.27	55.23
BERT$^*_{base}$	-	54.16	-	53.20
Two-Phase BERT$^*_{base}$	-	54.42	-	53.92
LSR+BERT$^*_{base}$	52.43	59.00	56.97	59.05
GAIN+BERT$^*_{base}$	59.14	61.22	59.00	61.24
HeterGSAN-Rec+BERT$^*_{base}$	58.13	60.18	57.12	59.45
ATLOP-BERT$^*_{base}$	59.22	61.09	59.31	61.30
Our DocRE Systems				
DRN	54.61	56.49	54.35	56.33
DRN+BERT$_{base}$	59.33	61.39	59.15	61.37

Table 2: Results on the development set and the test set. Results with $*$ are reported in their original papers. Results with $\dagger$ are reported in (Nan et al., 2020). Bold results indicate the best performance of the current method.

4.4 Evaluating Hyper-parameter K for The Number of Reasoning Paths

K	Dev Set		Test Set		Cover
	Ign F1	F1	Ign F1	F1	(%)
1	54.04	55.94	53.83	55.81	63.05
2	54.63	56.47	54.12	56.07	82.17
3	54.61	56.49	54.35	56.33	90.40
4	54.52	56.34	54.06	55.93	95.22
>4	54.31	56.25	53.97	55.84	100

Table 3: The effect of the number of reasoning paths K for the proposed DRN model.

To evaluate the effect of the number of reasoning path K in Eq.6, we reported the results for the different number of reasoning path K, as shown in Table 3. When K increased from 1 to 3, F1 scores of the proposed DRN model gradually improved from 55.81 to 56.33 on the test set and the percentage of covered reasoning paths reaches 90.40%. As the hyper-parameter K continues to increase, F1 scores began to drop on the dev and test sets. On the one hand, the reason may be that the reasoning information provided by too many reasoning paths is duplicated, even noises in the remaining 9.60% reasoning paths. On the other hand, the hyper-parameter K=3 can

---

[1] `https://competitions.codalab.org/competitions/20717`

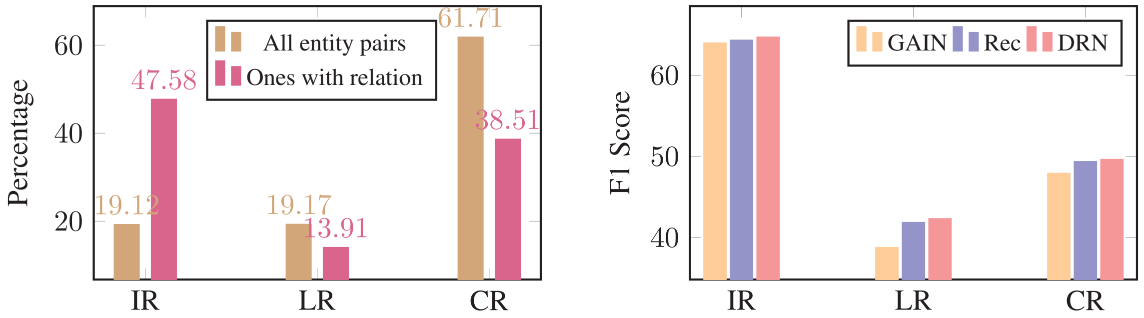

(a) (b)

Figure 3: (a) The statistical result of different reasoning task. (b) The performance of different reasoning task.

make the proposed DRN gain the highest F1 score on the dev and test sets. Therefore, we set the hyper-parameter K to three in our main results in Table 2.

4.5 Ablation Experiments

In the proposed DRN model, we model different reasoning tasks discriminatively using HGCRep and DLCRep, and we choose the highest scores as the final results. Instead of using the discriminative reasoning framework, previous work averaged the mention representation (HGCRep or DLCRep) to get the entity representation and concatenate the two entity representation to classify the relation, which we denote as Uniform model. Table 4 shows ablation experiments of the framework and different reasoning context on the test set. It is noted that Uniform model with the discriminative reasoning framework is our DRN model. First, the DocRE models benefit from our discriminative reasoning framework no matter what the reasoning context is used. Specially, the F1 score of the model with the framework was averagely 1.21 points superior to the Uniform model on the test set no matter what context representation is used, which illustrated the effectiveness of the framework. Second, when we gradually remove DLCRep and HGCRep from the Uniform and the proposed DRN model, both of the model's performance drops. Specially, F1 scores of DRN without DLCRep dropped by 1.32 while F1 scores of DRN without HGCRep dropped by 2.20 respectively. This indicates that both DLCRep and HGCRep play an important role in capturing the information of nodes on the reasoning paths. When removing both of DLCPeps and HGCReps from the DRN model, the model was degraded to the BiLSTM model with our discriminative reasoning framework. Obviously, F1 scores drastically decreased on the test sets, confirming the necessity of learning DLCRep and HGCRep for modeling reasoning discriminatively.

Model	without framework		with framework		Delta
	Ign F1	F1	Ign F1	F1	Ign F1
Uniform	53.68	55.79	54.35	56.33	+0.83
-DLCReps	51.82	53.83	52.96	55.01	+1.33
-HGCReps	51.21	53.36	52.35	54.13	+0.97
-Both	44.73	51.06	50.68	52.78	+1.71

Table 4: Ablation experiments.

4.6 Analysis of the Reasoning Tasks

In this section, we first showed the percent of all entity pairs (396,790) and entity pair with relation (12,332) on the dev set selected for three defined reasoning tasks through max operation in Eq.(12), as shown in Figure 3(a). For example, IR, LR, and CR are the intra-Sentence reasoning task, the logical reasoning task, and the coreference reasoning task, respectively. The percentages of IR, LR, and CR which is selected for all the entity pair are 19.12%, 19.17%, and 61.71% for all entity pairs, respectively. This indicates that our defined three reasoning skills can completely cover all entity pairs regardless of whether these entity pairs have relationships or not. Also, the percentages of IR, LR, and CR are 47.58%, 13.91%, and 38.51% for entity pairs with relation, respectively. This is consistent with the statistical result in the Yao et al.'s work that more than 40.7% relational facts can only be extracted from multiple sentences, validating that our method can model different reasoning skills discriminatively on the DocRE dataset.

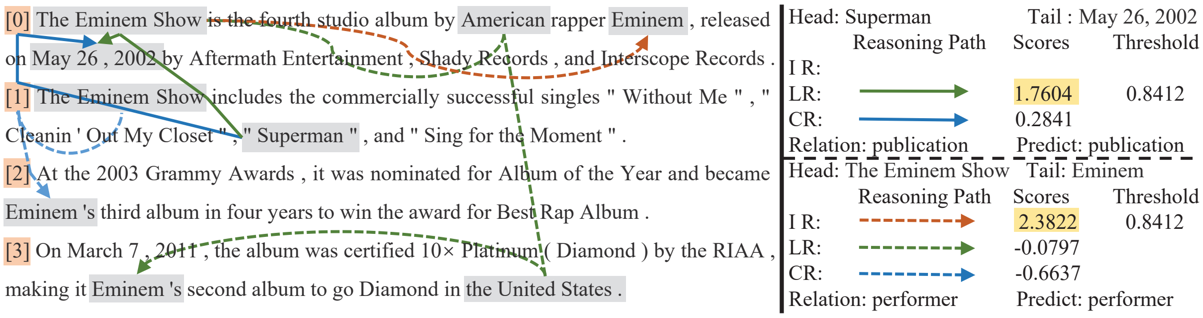

[0] The Eminem Show is the fourth studio album by American rapper Eminem , released on May 26 , 2002 by Aftermath Entertainment , Shady Records , and Interscope Records .
[1] The Eminem Show includes the commercially successful singles " Without Me " , " Cleanin ' Out My Closet " , " Superman " , and " Sing for the Moment " .
[2] At the 2003 Grammy Awards , it was nominated for Album of the Year and became Eminem 's third album in four years to win the award for Best Rap Album .
[3] On March 7 , 2011 , the album was certified 10× Platinum (Diamond) by the RIAA , making it Eminem 's second album to go Diamond in the United States .

Head: Superman	Reasoning Path	Scores	Threshold
		Tail : May 26, 2002	
I R:			
LR:	→	1.7604	0.8412
CR:	→	0.2841	
Relation: publication		Predict: publication	

Head: The Eminem Show	Reasoning Path	Scores	Threshold
		Tail: Eminem	
I R:	⇢	2.3822	0.8412
LR:	⇢	-0.0797	
CR:	⇢	-0.6637	
Relation: performer		Predict: performer	

Figure 4: Case study.

Moreover, Figure 3(b) showed the results of HerterGSAN-Rec (abbreviated as Rec), GAIN, and our DRN models on three different reasoning tasks. As seen, F1 scores of the proposed DRN model are higher than that of Rec and GAIN models over all three tasks. This means that modeling reasoning types explicitly can effectively advance the DocRE. For all DocRE models, F1 scores of LR task and CR task were far inferior to that of IR task, which is consistent with the intuitive perception that the inter-sentence reasoning is more difficult than the intra-sentence reasoning.

4.7 Analysis of the Reasoning Type

Confusion Matrix					
		Truth			
		IR	LR	CR	Total
Predict	IR	321	36	12	369
	LR	25	88	29	142
	CR	88	129	188	405
	Total	434	253	229	
Metric					
		IR	LR	CR	
F1 Scores		79.95	38.77	44.87	

Table 5: Confusion matrix of different reasoning types.

To further show the selected different reasoning types in Eq.(12), we randomly sampled 72 documents from the dev set which contain 916 relation instances, and we ask three human to annotate the reasoning types of all the entity pairs with relation in the sampled document according to three defined reasoning types, including the intra-sentence reasoning, the logical reasoning, and the coreference reasoning (The annotation data can be found in `https://github.com/xwjim/DRN`). Table 5 shows the number and F1 scores of each selected reasoning types on the sampled 72 documents. As seen, F1 scores of IR, LR, and CR are 79.95%, 38.77%, and 44.87%, respectively, indicating that modeling reasoning discriminatively is working during selecting of reasoning paths in Eq.(12). Also, our method is the capacity of recognizing not only the intra-sentence reasoning but also the intra-sentence reasoning. In addition, there is a certain percentage of the mistakenly selected reasoning types, indicating that our method may have more room for improvement in the future.

4.8 Case Study

Figure 4 shows the relation classification about two entity pairs for our DRN model. For the first entity pair {"*Superman*"} and {"*May 26,2002*"}, there are reasoning paths for Task2 and Task3, and their scores are 1.7604, and 0.2841,respectively As a result, Task2 was used to predict the relation "{*publication date*}" between {"*Superman*"} and {"*May 26,2002*"} correctly. Meanwhile, the selection of Task2 is consistent with the ground-truth logical reasoning type. Moreover, the above reasoning processing is also similar to the entity pair {"*The Eminem show*"} and {"*Eminem*"} with three reasoning types.

5 Related Work

Early research efforts on relation extraction concentrate on predicting the relation between two entities with a sentence (Zeng et al., 2014, 2015; Wang et al., 2016; Sorokin and Gurevych, 2017; Feng et al., 2018; Song et al., 2019; Wei et al., 2020). These approaches do not consider interactions across mentions and ignore relations expressed across sentence boundaries. The semantics of a document context is coherent and a part of relation can only be extracted among sentences.

However, as large amounts of relationships are expressed by multiple sentences, recent work starts to explore document-level relation extraction. People begin to consider the relation between disease and chemicals in the entire document of biomedical domain (Quirk and Poon, 2017; Gupta et al., 2019; Zhang et al., 2018; Christopoulou et al., 2019; Zhu et al., 2019). A large-scale general-purpose dataset for DocRE constructed from Wikipedia articles has been proposed in (Yao et al., 2019), which has advanced the DocRE a lot. Most approaches on DocRE are based on document graphs, which were introduced by Quirk and Poon. Specifically, they use words as nodes and construct a homogenous graph using syntax parsing tools and a graph neural network is used to capture the document information. This document graph provides a unified way of extracting the features for entity pairs. Later work extends the idea by improving neural architectures (Peng et al., 2017; Verga et al., 2018; Gupta et al., 2019) or adding more types of edges (Christopoulou et al., 2019). In the Christopoulou et al.'s work, the author construct the graph which contains different granularities (sentence, mention, entity) through co-occurrence and heuristic rule to model the graph without external tools. More recent most of the approach (Christopoulou et al., 2019; Zeng et al., 2020; Xu et al., 2021) constructs heterogeneous graph through co-occurrence and heuristic rule to model the graph without external tools. In the (Zeng et al., 2020) constructed double graphs in different granularity to capture document-aware features and the interaction between entities. In the (Xu et al., 2021) introduced a reconstructor to reconstruct the path in the graph to guide the model to learning a good node representation. Other attempts focus on the multi-entity and multi-label problems (Zhou et al., 2021). Zhou et al. proposed two techniques to solve the problems, adaptive thresholding and localized context pooling.

6 Conclusion

In this paper, we propose a novel discriminative reasoning framework to consider different reasoning types explicitly. We use meta-path strategy to extract the reasoning path for different reasoning types. Based on the framework, we propose a Discriminative Reasoning Network (DRN), in which we use both the heterogeneous graph context and the document-level context to represent different reasoning paths. The ablation study validates the effectiveness of our discriminative framework and different modules on the large-scale human-annotated DocRE dataset. In particular, our method archives a new state-of-the-art performance on the DocRE dataset. In the future, we will explore more diverse structure information (Chen et al., 2018; Chen et al., 2020; Cohen et al., 2020) from the input document for the discriminative reasoning framework, and apply the proposed approach to other NLP tasks (Zhang et al., 2020a; Chen et al., 2020; Zhang et al., 2020b).

Acknowledgments

We are grateful to the anonymous reviewers, area chair and Program Committee for their insightful comments and suggestions. The corresponding authors are Kehai Chen and Tiejun Zhao. The work of this paper is funded by the project of National Key Research and Development Program of China (No. 2020AAA0108000).

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

K. Chen, R. Wang, M. Utiyama, E. Sumita, T. Zhao, M. Yang, and H. Zhao. 2020. Towards more diverse input representation for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1586–1597.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *AAAI Conference on Artificial Intelligence*, pages 4792–4798, New Orleans, Lousiana, USA.

Xiao Chen, Changlong Sun, Jingjing Wang, Shoushan Li, Luo Si, Min Zhang, and Guodong Zhou. 2020. Aspect sentiment classification with document-level sentiment preference modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3677, Online. Association for Computational Linguistics.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* *and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4925–4936, Hong Kong, China. Association for Computational Linguistics.

William W. Cohen, Haitian Sun, R. Alex Hofer, and Matthew Siegler. 2020. Scalable neural methods for reasoning with a symbolic knowledge base. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics.

Pankaj Gupta, Subburam Rajaram, Hinrich Schütze, and Thomas A. Runkler. 2019. Neural relation extraction within and across sentence boundaries. In *The 33th AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6513–6520. AAAI Press.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1546–1557, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *Conference and Workshop on Neural Information Processing Systems*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Trans. Assoc. Comput. Linguistics*, 5:101–115.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182, Valencia, Spain. Association for Computational Linguistics.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4309–4316, Florence, Italy. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Daniel Gildea, Mo Yu, Zhiguo Wang, and Jinsong Su. 2019. Leveraging dependency forest for neural medical relation extraction. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1784–1789, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884, New Orleans, Louisiana. Association for Computational Linguistics.

Hong Wang, Christfried Focke, Rob Sylvester, Nilesh Mishra, and William W. J. Wang. 2019. Fine-tune bert for docred with two-step process. *ArXiv*, abs/1909.11898.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, Berlin, Germany. Association for Computational Linguistics.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.

Wang Xu, Kehai Chen, and Tiejun Zhao. 2021. Document-level relation extraction with reconstruction. In *The 35th AAAI Conference on Artificial Intelligence (AAAI-21)*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1630–1640, Online. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard Hovy. 2020a. A two-step approach for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7479–7485, Online. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020b. Sg-net: Syntax-guided machine reading comprehension. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2020)*, pages 9636–9643.

Wenxuan Zhou, Kevin Huang, Tengyu Ma, and Jing Huang. 2021. Document-level relation extraction with adaptive thresholding and localized context pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph neural networks with generated parameters for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1331–1339, Florence, Italy. Association for Computational Linguistics.

Meta-Learning Adversarial Domain Adaptation Network for Few-Shot Text Classification

ChengCheng Han[1] **Zeqiu Fan**[1] **Dongxiang Zhang**[2] **Minghui Qiu**[3]
Ming Gao[1*] **Aoying Zhou**[1]
[1]School of Data Science and Engineering, East China Normal University
[2]College of Computer Science and Technology, Zhejiang University
[3]Alibaba Group
`{51195100009,51195100007}@stu.ecnu.edu.cn`
`zhangdongxiang@zju.edu.cn`
`minghui.qmh@alibaba-inc.com`
`mgao@dase.ecnu.edu.cn`
`ayzhou@sei.ecnu.edu.cn`

Abstract

Meta-learning has emerged as a trending technique to tackle few-shot text classification and achieved state-of-the-art performance. However, existing solutions heavily rely on the exploitation of lexical features and their distributional signatures on training data, while neglecting to strengthen the model's ability to adapt to new tasks. In this paper, we propose a novel meta-learning framework integrated with an adversarial domain adaptation network, aiming to improve the adaptive ability of the model and generate high-quality text embedding for new classes. Extensive experiments are conducted on four benchmark datasets and our method demonstrates clear superiority over the state-of-the-art models in all the datasets. In particular, the accuracy of 1-shot and 5-shot classification on the dataset of 20 Newsgroups is boosted from 52.1% to 59.6%, and from 68.3% to 77.8%, respectively[1].

1 Introduction

Few-shot text classification (Yu et al., 2018; Geng et al., 2019) is a task in which a model will be adapted to predict new classes not seen in training. For each of these new classes, we only have a few labeled examples. To be specific, we are given lots of training data with a set of classes $\mathcal{Y}_{train}$. After training, our goal is to get accurate classification results on the testing data with a set of new classes $\mathcal{Y}_{test}$, which is disjoint to $\mathcal{Y}_{train}$. Only a small labeled support set will be available in the testing stage. If the support set contains K labeled examples for each of the N unique classes, we refer to the task as a N-way K-shot classification.

Existing approaches for few-shot text classification mainly fall into two categories: (1) transfer-learning based methods (Howard and Ruder, 2018; Pan et al., 2019; Gupta et al., 2020), which aim to transfer knowledge learned from a task to a new task or leverage general-domain pretraining and fine-tuning techniques for few-shot classification. (2) meta-learning based methods (Jamal et al., 2018; Yu et al., 2018; Geng et al., 2019, 2020; Bao et al., 2020), which aim to learn generic information (meta-knowledge) by recreating training episodes, so that it can classify new classes through only a few labeled examples. Among these methods, Bao et al. (2020) leveraged distributional signatures (e.g. word frequency and information entropy) to train a model within a meta-learning framework, and achieved state-of-the-art performance. However, the method pays more attention to statistical information and ignores other implicit information such as correlation between words. Furthermore, existing meta-learning methods heavily rely on the exploitation of lexical features and their distributional signatures on training data, while neglecting to strengthen the model's ability to adapt to new tasks.

In this paper, we propose an adversarial domain adaptation network to enhance meta-learning framework, with the objective of improving the model's adaptive ability for new tasks in new domains. We first utilize two neural networks competing against each other, separately playing the roles of a domain discriminator and a meta-knowledge

---

[*] Corresponding author

[1] The source code of the paper is available at `https://github.com/hccngu/MLADA`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1664–1673
August 1–6, 2021.

generator. The adversarial network is able to strengthen the adaptability of the meta-learning architecture. Moreover, we aggregate transferable features generated by the meta-knowledge generator with sentence-specific features to produce high-quality sentence embeddings. Finally, we utilize a ridge regression classifier to obtain final classification results. To the best of our knowledge, we are the first to combine adversarial domain adaptation with meta-learning for few-shot text classification.

We evaluate our model on four popular datasets for few-shot text classification. Experimental results demonstrate that our method outperforms state-of-the-art models in all datasets, for both in 1-shot and 5-shot classification tasks. Especially on the 20 Newsgroups dataset, our model outperforms DS-FSL (Bao et al., 2020) by 7.5% in 1-shot classification and 9.5% in 5-shot classification. In addition, we conduct visualization analysis to verify the adaptability of our model and capability to recognize important lexical features for unseen classes.

2 Related Work

The mainstream approaches for few-shot text classification are based on meta-learning or transfer learning. In this section, we first briefly introduce the preliminary background of these two technologies, and then review how they are applied to support few-shot text classification.

Meta-learning Meta-learning, also known as "learning to learn", refers to improving the learning ability of a model through multiple training episodes so that it can learn new tasks or adapt to new environments quickly with a few training examples. Existing approaches mainly fall into two categories: (1) Optimization-based methods , including developing a meta-learner as optimizer to output search steps for each learner directly (Andrychowicz et al., 2016; Ravi and Larochelle, 2017; Mishra et al., 2018; Gordon et al., 2019) and learning an optimized initialization of model parameters, which can be later adapted to new tasks by a few steps of gradient descent (Finn et al., 2017; Yoon et al., 2018; Grant et al., 2018; Bao et al., 2020). (2) Metric-based methods, including Matching Network (Vinyals et al., 2016), PROTO (Snell et al., 2017), Relation Network (Sung et al., 2018), TapNet (Yoon et al., 2019) and Induction Network (Geng et al., 2019), which aim to learn an appropriate distance metric to compare validation points with training points and make prediction through matching training points.

Transfer learning Few-shot text classification relates closely to transfer learning (Zhuang et al., 2021) that aims to leverage knowledge from a related domain (a.k.a. source domain) to improve the learning performance and reduce the reliance on the number of labeled examples required in a target domain. Compared to meta-learning designed to aggregate the knowledge learned from many tasks, transfer learning typically involves a few tasks. In addition, we aim to directly reuse or fine-tune some existing representation in transfer learning, while a meta-learner is typically optimized at adapting to new tasks. Domain adaptation (Ganin et al., 2016; Tzeng et al., 2017; Khaddaj and Hajj, 2020) is a type of transfer learning, which aims to bridge the gap between the source and target domains by learning domain-invariant feature representations. Pre-trained model (Devlin et al., 2019; Yang et al., 2019; Brown et al., 2020) can also be viewed as a type of transfer learning. The parameters pre-trained in the source domain are fine-tuned in the target domain, with faster training convergence.

Few-shot text classification To tackle few-shot text classification, a straightforward idea is to apply BERT (Devlin et al., 2019) or XLNet (Yang et al., 2019), which have achieved strong performance in text classification by fine-tuning with a small number of training examples. Their performances can be less dependent on the number of training samples for the new classes. Some other approaches are based on transfer learning. Pan et al. (2019) proposed a modified hierarchical pooling strategy over pre-trained word embeddings to transfer knowledge obtained from some source domains to the target domain. Gupta et al. (2020) developed a binary classifier on the source domain to classify new classes by prefixing class identifiers to input texts.

Meta-learning (Jamal et al., 2018; Yu et al., 2018; Geng et al., 2019, 2020; Bao et al., 2020) can also be utilized to solve few-shot text classification, and has achieved state-of-the-art performance. Yu et al. (2018) proposed an adaptive metric learning approach that automatically determines the best weighted combination from meta-training tasks for few-shot tasks. Geng et al. (2019, 2020) leveraged the dynamic routing algorithm in meta-learning for few-shot text classification. (Bao et al., 2020) leveraged distributional signatures (e.g. word frequency and information entropy) to train a model within a meta-learning framework.

3 Method

In this section, we first present the preliminary background on episode-based meta-learning framework (Vinyals et al., 2016). After that, we explicitly describe the proposed MLADA (Meta-Learning Adversarial Domain Adaptation) Network.

3.1 Episode-based meta-learning

The goal of meta-training is to train a classifier that can learn meta-knowledge from training data. In this way, the classifier can quickly learn from a few annotations when classifying unseen classes. The "episode" training strategy that Vinyals et al. (2016) proposed has proved to be effective. The episode-based meta-learning consists of two main stages:

Meta-training Firstly, N classes are sampled from training data $\mathcal{Y}_{train}$. For each of these N classes, two subsets of examples are sampled separately as the support set S and the query set Q. Next, input the support set S and the query set Q to the model and update the parameters by minimizing the loss in the query set Q. The procedure above is called a training episode, which will be repeated multiple times.

Meta-testing After meta-training is finished, the performance of the model will be evaluated by the same episode-based mechanism. In a testing episode, N new classes will be sampled from $\mathcal{Y}_{test}$, which is disjoint to $\mathcal{Y}_{train}$. Then the support set and the query set will be sampled from the N classes. The model parameters can be fine-tuned through the small support set. The performance of the model will be evaluated through the average classification accuracy on the query set across all testing episodes.

We found that only a small subset of training data are accessible per training episode in the standard episode-based meta-training (Vinyals et al., 2016). To solve this problem, we build domain adversarial tasks to utilize more training data per training episode. Details of our model are described in the next section.

3.2 Meta-Learning Adversarial Domain Adaptation Network (MLADA)

Overview Our goal is to improve the performance of few-shot classification by combining adversarial domain adaptation and episode-based meta-learning. Figure 1 gives an overview of our model. In the rest of this section, we will introduce the main components of the model.

Word Representation Layer The goal of this layer is to represent each word with a d-dimensional vector. Following Bao et al. (2020), we construct the d-dimensional vector with the word embeddings, which is pre-trained with fastText (Joulin et al., 2016).

Domain Discriminator We refer to the support set and the query set as the target domain and the rest of the training data as the source domain. We sample a subset of examples from the source domain as the source set. The goal of this module is to distinguish whether the sample is from the source domain or the target domain. The discriminator is a three layer feed-forward neural network. We apply the $softmax$ function in the output layer to evaluate the probability distribution $Pr(y|\lambda)$. $y = 0 \; or \; 1$ represents that the sample is from the query set or the source set.

Meta-knowledge Generator This module is mainly composed of a bi-directional LSTM (BiLSTM) and a fully connected layer. We utilize a BiLSTM to encode contextual embeddings for each time-step. The input of the module is a sequence of word vectors $P : [p_1, ..., p_m]$, where m represents the number of words in a sentence. The output is a matrix $h^p_{d\times m}$, which is composed of contextual embeddings.

$$\begin{aligned} \overrightarrow{h^p_i} &= \overrightarrow{LSTM}(\overrightarrow{h^p_{i-1}}, p_i) \quad i = 1, ..., m && (1) \\ \overleftarrow{h^p_i} &= \overleftarrow{LSTM}(\overleftarrow{h^p_{i+1}}, p_i) \quad i = m, ..., 1 && (2) \\ h^p_i &= Concat(\overrightarrow{h^p_i}, \overleftarrow{h^p_i}) \quad i = 1, ..., m && (3) \\ h^p &= [h^p_1, h^p_2, ..., h^p_m] && (4) \end{aligned}$$

Next, we employ a single layer feed-forward neural network and apply the $softmax$ function to get the output k^p.

$$k^p = Softmax(\omega \cdot h^p + b) \quad (5)$$

k^p is an n-dimensional vector, which represents the meta-knowledge included in the sentence. n denotes the length of the sentence.

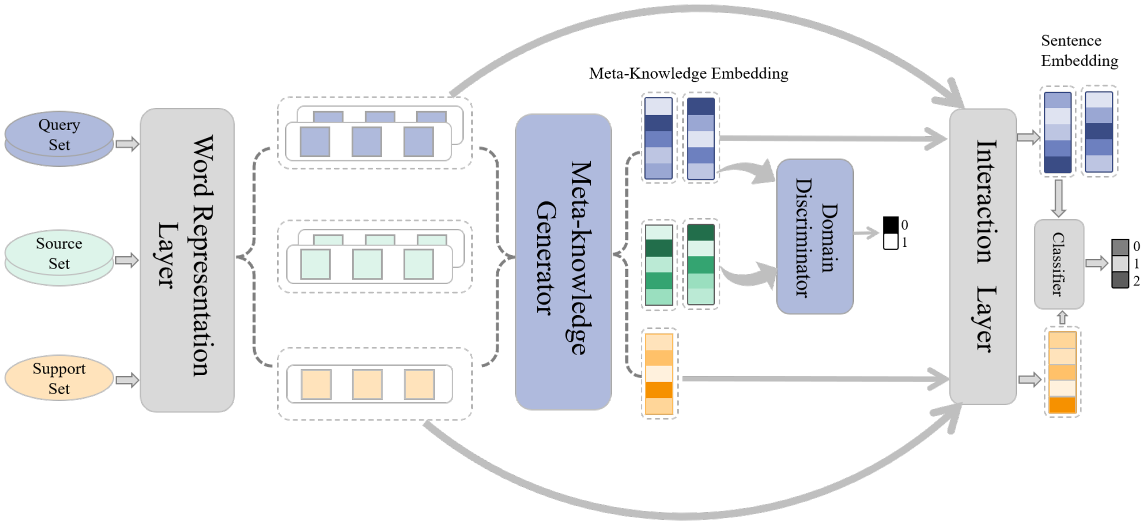

Figure 1: MLADA Network architecture for a N-way K-shot($N=3, K=2$) problem

The goal of the meta-knowledge generator is not only to make the final classification results better, but also to confuse the domain discriminator as much as possible, so that the discriminator can not distinguish between samples from query set or source set. The theory on domain adaptation suggests that, for effective domain transfer to be achieved, predictions must be made based on features that cannot discriminate between the source domain and target domain, which is the motivation for us to build the meta-knowledge generator.

Interaction Layer We consider that the vector generated by the meta-knowledge generator is the transferable features, and word embeddings is the specific features of sentences. The role of the interaction layer is to fuse transferable features and sentence-specific features to produce the output as sentence embeddings, which will be used as the input of the classifier to obtain the final classification results. Suppose that the length of the sentence p is m, the word vectors is $w_i^p(i \in [1,m])$, the dimension of the word vector is d and the meta-knowledge of the sentence is k^p, then the final sentence vector is s^p:

$$s^p = W^p_{d\times m}\cdot k^p \tag{6}$$

where $W^p=[w_1^p, w_2^p, ..., w_m^p]$.

Classifier The classifier is trained by the support set from scratch for each episode. We choose the $ridge\ regression$ as the classifier. The reason why we adopt the ridge regression to fit the support set are as follows: 1) If we choose neural networks as the classifier, it will be trained inadequately because the number of samples in the support set is too small. 2) The ridge regression admits a closed-form solution and it reduces over-fitting on the small support set through proper regularization.Specifically, we minimize regularized squared loss:

$$\mathcal{L}^{RR}(\theta)=\frac{1}{2m}\sum_{i=1}^{m}[((f_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2)] \tag{7}$$

where m represents the number of samples in the support set, $f_\theta(x^{(i)})$ represents the prediction of the ridge regressor, $y^{(i)}$ represents the label of the sample, $\sum_{j=1}^{n}\theta_j^2$ denotes the squared Frobenius norm and $\lambda > 0$ controls the extent of the regularization.

Loss Function In each training episode, we first fix the parameters of the generator and the discriminator to update the classifier's parameters by the support set. The classifier's loss function is shown in Eq.7.

Next, we fix the parameters of the generator and the classifier to update the discriminator's parameters by the query set and the source set. We use the cross-entropy loss as the discriminator's loss function, which is shown in Eq.8.

$$\begin{aligned}\mathcal{L}^{D}(\mu) &= -\frac{1}{2m}\sum_{i=1}^{2m}[y_d^{(i)}logD_\mu(k^{(i)}) \\ &+ (1-y_d^{(i)})log(1-D_\mu(k^{(i)}))]\end{aligned} \tag{8}$$

Algorithm 1 MLADA Training Procedure

```
Input: Training data $\{\mathcal{X}_{train}, \mathcal{Y}_{train}\}$; $T$ episodes
    and $ep$ epochs; $N$ classes in support set or
    query set; $K$ samples in each class in the sup-
    port set and $L$ samples in each class in the
    query set; The generator's parameters $\beta$, the
    discriminator's parameters $\mu$ and the classi-
    fier's parameters $\theta$.
Output: Parameters $\beta$ and $\mu$ after training;
Randomly initialize the model parameters $\beta$,
    $\mu$ and $\theta$;
for each $i \in [1, ep]$ do
    $\mathcal{Y} \leftarrow \Lambda(\mathcal{Y}_{train}, N)$;[1]
    for each $j \in [1, T]$ do
        $S, Q, \Phi \leftarrow \emptyset, \emptyset, \emptyset$;
        for $y \in \mathcal{Y}$ do
            $S \leftarrow S \cup \Lambda(\mathcal{X}_{train}\{y\}, K)$;[2]
            $Q \leftarrow Q \cup \Lambda(\mathcal{X}_{train}\{y\} \backslash S, L)$;
            $\Phi \leftarrow \Phi \cup \Lambda(\mathcal{X}_{train} \backslash \mathcal{X}_{train}\{y\}, L)$;
        end for
        Input $S$ to the model;
        Fix $\mu, \beta$. Update $\theta$ by minimizing the
            Eq.7;
        Input $Q, \Phi$ to the model;
        Fix $\beta, \theta$. Update $\mu$ by minimizing the
            loss of the discriminator (Eq.8);
        Fix $\mu, \theta$. Update $\beta$ by minimizing the
            loss of the generator (Eq.9);
    end for
end for
```

where μ denotes the parameters of the discriminator, m represents the number of samples of the query set or the source set.$y_d = 0 \ or \ 1$ denotes whether the sample is from the source set or the query set. k represents the meta-knowledge vector.

Finally, we fix the parameters of the discriminator and the classifier to update the generator's parameters by the query set and the source set. The loss function of the generator is composed of two components. The first one is a cross-entropy loss for the final classification results, and the second one is the opposite of the discriminator's loss, which is to confuse the discriminator.

$$\mathcal{L}^G(\beta) = CELoss(f(W \cdot G_\beta(W)), y) - \mathcal{L}^D \quad (9)$$

where β represents the generator's parameters. f denotes the ridge regressor. W represents the matrix of word vectors in a sentence. y denotes the real labels of samples.$\mathcal{L}^D$ is shown in Eq.8.

Training Procedure It is remarkable that the meta-knowledge generator is optimized over all training episodes, while the classifier is trained from scratch for each episode. In each training episode, we first utilize the support set to update the parameters in the classifier. Next, we use the query set and source set to update the parameters of the meta-knowledge generator and the domain discriminator. The details of training procedure of our model are shown in Algorithm 1.

4 Experiments

In this section, we perform comprehensive experiments to compare our proposed model with five competitive baselines, and evaluate the performance on four text classification datasets.

4.1 Datasets

We use four benchmark datasets for text classification, whose statistics are summarized in Table 1.

HuffPost headlines contains 41 classes of news headlines from the year 2012 to 2018 obtained from HuffPost (Misra, 2018). Its text is less abundant (i.e., with smaller text length) than the other datasets and considered to be more challenging for text classification.

Amazon product data contains product reviews from 24 product categories, including 142.8 million reviews spanning 1996-2014 (He and McAuley, 2016). Our task is to identify the product categories of the reviews. Since the original dataset is proverbially large, we sample a subset of $1,000$ reviews from each category.

Reuters-21578 is collected from Reuters newswire in 1987. We use the standard ApteMode version of the dataset. Following Bao et al. (2020), we consider 31 classes and remove multi-labeled articles. Each class contains at least 20 articles.

20 Newsgroups is a collection of approximately 20,000 newsgroup documents (Lang, 1995), partitioned (nearly) evenly across 20 different newsgroups.

4.2 Experiment Setup

Baselines We compare our MLADA with multiple competitive baselines, which are briefly summarized in the following:

---

[1] $\Lambda(\mathcal{Y}, N)$ denotes selecting N elements from $\mathcal{Y}$ randomly.

[2] $\mathcal{X}_{train}\{y\}$ denotes samples labeled y in $\mathcal{X}_{train}$.

Dataset	Avg. text length	vocab size	# samples	# train / val / test classes
HuffPost	11	8218	36900	20 / 5 / 16
Amazon	140	17062	24000	10 / 5 / 9
Reuters	168	2234	620	15 / 5 / 11
20 Newsgroups	340	32137	18820	8 / 5 / 7

Table 1: Statistics of the four benchmark datasets.

Method	HuffPost		Amazon		Reuters		20 News		Average	
	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
MAML(2017)	35.9	49.3	39.6	47.1	54.6	62.9	33.8	43.7	40.9	50.8
PROTO(2017)	35.7	41.3	37.6	52.1	59.6	66.9	37.8	45.3	42.7	51.4
Induct(2019)	38.7	49.1	34.9	41.3	59.4	67.9	28.7	33.3	40.4	47.9
HATT(2019)	41.1	56.3	49.1	66.0	43.2	56.2	44.2	55.0	44.4	58.4
DS-FSL(2020)	43.0	63.5	62.6	81.1	81.8	96.0	52.1	68.3	59.9	77.2
MLADA(ours)	**45.0**	**64.9**	**68.4**	**86.0**	**82.3**	**96.7**	**59.6**	**77.8**	**63.9**	**81.4**

Table 2: Mean accuracy (%) of 5-way 1-shot and 5-way 5-shot classification over four datasets.

- **MAML** (Finn et al., 2017) is trained by maximizing the sensitivity of the loss functions of new tasks, so that it can rapidly adapt to new tasks after the parameters have been up-dated through few gradient steps.

- **Prototypical Networks** (Snell et al., 2017), abbreviated as **PROTO**, is a metric-based method for few-shot classification by using sample averages as class prototypes.

- **Induction Networks** (Geng et al., 2019) learns a class-wise representation by leveraging the dynamic routing algorithm in meta-learning.

- **HATT** (Gao et al., 2019) extends **PROTO** by adding a hybrid attention mechanism to the prototypical network.

- **DS-FSL** (Bao et al., 2020) is trained within a meta-learning framework to map the distribution signatures into attention scores so as to extract more transferable features.

Implementation Details Following Bao et al. (2020), we use pre-trained fastText (Joulin et al., 2016) for word embedding. In the meta-knowledge generator, we use a BiLSTM with 128 hidden units. In the domain discriminator, the numbers of hidden units for the two feed-forward layers are set to 256 and 128, respectively. All parameters are optimized using Adam with a learning rate of 0.001 (Kingma and Ba, 2015).

During meta-training, we perform 100 training episodes ($T = 100$) per epoch. Meanwhile, we apply early stopping when the accuracy on the validation set fails to improve for 20 epochs. We evaluate the model performance based on $1,000$ testing episodes and report the average accuracy over 5 different random seeds. All the experiments are conducted on a NVIDIA v100 GPU.

4.3 Experimental Results

The experimental results are reported in Table 2. Our model achieves the best performance across all datasets, with an average accuracy of 63.9% in 1-shot classification and 81.4% in 5-shot classification, outperforming the state-of-the-art model DS-FSL (Bao et al., 2020) by a notable 4% improvement. For DS-FSL, it extracts transferable features via certain distribution signatures (e.g., word frequency or information entropy), but ignores other information of sentences, including implicit interaction between words. In contrast, we does not limit the transferable knowledge to statistical information. Our strategy is to combine the proposed domain adversarial network with meta-learning, generating more comprehensive transferable features.

Furthermore, our model improves dramatically 7.5% and 9.5% on 20 Newsgroups in 1-shot and

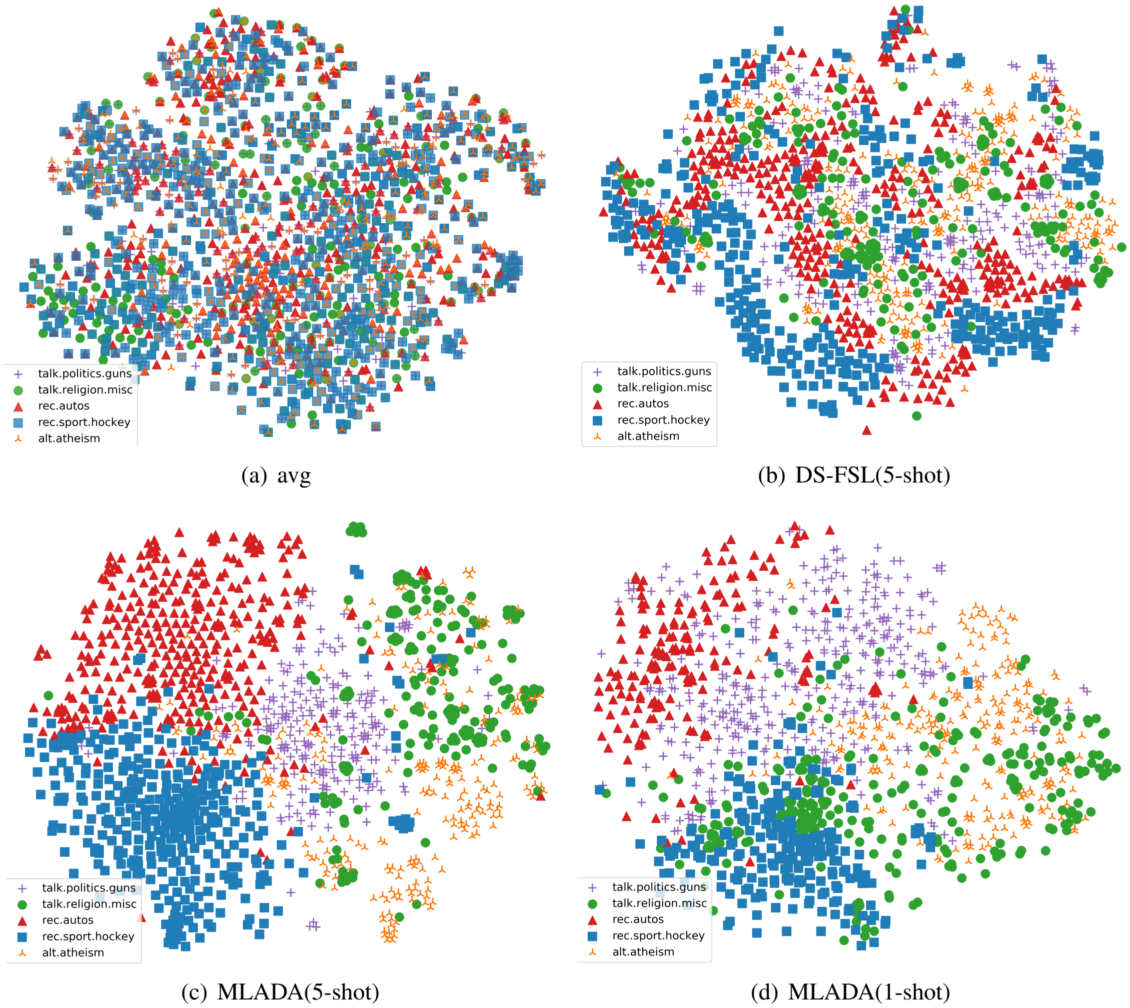

(a) avg (b) DS-FSL(5-shot) (c) MLADA(5-shot) (d) MLADA(1-shot)

Figure 2: t-SNE visualization of the input representation of the classifier for a testing episode($N = 5$, $K = 5$, $L = 500$)sampled from 20 Newsgroups. Note that the 5 classes is not seen in training set. The input representation of the classifier given by (a) the average of word embeddings (b) DS-FSL and (c) MLADA(ours). (d) is the t-SNE visualization of MLADA on 5-way 1-shot classification.

5-shot classification. The average length of texts in the 20 Newsgroups is longer than the other datasets. The empirical results clearly demonstrate that our model is more suitable for longer texts, which contain more abundant text information.

4.4 Ablation Study

We conduct an ablation study to examine the effectiveness of the proposed domain adversarial network as well as the interaction layer and the source set. The results of Amazon dataset are reported in Table 3.

Firstly, we use a bi-directional LSTM instead of the proposed domain adversarial network (including the meta-knowledge generator and the domain discriminator) for sentence encoding. The performances in the tasks of 1-shot classification and 5-shot classification decrease by 6.5% and 5.3%, respectively. This verifies the effectiveness of the proposed domain adversarial network.

Secondly, we study how the interaction layer contributes to the performance of our model. We concatenate the vector generated by the meta-knowledge generator directly with the average sentence embedding instead of the interaction layer. From the result in Table 3, we can see that our proposed interaction layer to combine the transferable features with the sentence-specific information are indeed more effective.

Finally, we remove the source set and utilize the discriminator to distinguish the true classes of samples. We observe that the source set is also important to performance. Due to the removal of the source set, the model has only access to the sup-

Seen classes	*Politics, Entertainment, Food&Drink, College, Arts*	Prediction
DS-FSL	Senate committee advances bill to protect Robert Mueller .	*politics* ✓
MLADA(ours)	Senate committee advances bill to protect Robert Mueller .	*politics* ✓
Unseen classes	*Sports, Education, Media, Tech, Environment*	Prediction
DS-FSL	Olympic committee CEO resigns cites health issues.	*environment* ✗
MLADA(ours)	Olympic committee CEO resigns cites health issues.	*sports* ✓

Figure 3: The visualization of attention weights generated by DS-FSL and the meta-knowledge generator of our model.

port set and the query set in each training episode. Therefore, it cannot learn cross-domain transferable features.

Models	Accuracy(%)	
	1 shot	5 shot
– Domain Adversarial Network	61.9	80.7
– Interaction Layer	66.6	83.0
– Source Set	67.1	84.2
MLADA	**68.4**	**86.0**

Table 3: Ablation study results of 5-way 1-shot and 5-way 5-shot classification on the Amazon dataset.

4.5 Visualization

We utilize visualization experiments to demonstrate that our model can generate high-quality sentence embeddings and identify important lexical features for unseen classes.

We first use t-SNE (Van der Maaten and Hinton, 2008) visualization of sentence embeddings generated by different methods on the query set, as shown in Figure2. Compared to 2(a) average word embeddings and 2(b) DS-FSL, our method produces better separation both in 1-shot and 5-shot classification, demonstrating the effectiveness of MLADA in leveraging the supervised learning experience to generate high-quality sentence embeddings for few-shot text classification.

Moreover, we visualize the weight vectors generated by the meta-knowledge generator and compare it with DS-FSL, as shown in Figure 3. Our model reduces the weight of "committee" while increasing the weight of "Olympic", which demonstrates that our model can recognize important lexical features in the new task, rather than simply transferring features obtained from experience.

5 Conclusion

In this paper, we propose a novel meta-learning approach called Meta-Learning Adversarial Domain Adaptation Network(MLADA), which can recognize important lexical features and generate high-quality sentence embeddings in new classes(not seen in training data). Specifically, we design an adversarial domain adaptation network in meta-training episodes, which aims to extract domain-invariant features and improve the adaptability of the meta-learner in new classes. We demonstrate that our method outperforms the existing state-of-the-art approaches on four standard text classification datasets. Future work includes applying MLADA to other fields including computer vision and speech recognition, and exploring the combination between adversarial domain adaptation network and other FSL algorithms.

6 Acknowledgments

This work has been supported by the National Key Research and Development Program of China under Grant 2016YFB1000905, the National Natural Science Foundation of China under Grant No. U1811264, U1911203, 61877018, 61672234, 61672384 and Alibaba Group through Alibaba Innovative Research Program.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3981–3989.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *8th International Confer-*

ence on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17:59:1–59:35.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6407–6414. AAAI Press.

Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020. Dynamic memory induction networks for few-shot text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1087–1094. Association for Computational Linguistics.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3902–3911. Association for Computational Linguistics.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E. Turner. 2019. Meta-learning probabilistic inference for prediction. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas L. Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical bayes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Aakriti Gupta, Kapil Thadani, and Neil O'Hare. 2020. Effective few-shot classification with transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1061–1066. International Committee on Computational Linguistics.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics.

Muhammad Abdullah Jamal, Guo-Jun Qi, and Mubarak Shah. 2018. Task-agnostic meta-learning for few-shot learning. *CoRR*, abs/1805.07722.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomás Mikolov. 2016. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651.

Alaa Khaddaj and Hazem M. Hajj. 2020. Representation learning for improved generalization of adversarial domain adaptation with text classification. In *IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020, Doha, Qatar, February 2-5, 2020*, pages 525–531. IEEE.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 331–339. Morgan Kaufmann.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Rishabh Misra. 2018. News category dataset.

Chongyu Pan, Jian Huang, Jianxing Gong, and Xingsheng Yuan. 2019. Few-shot transfer learning for text classification with lightweight word embedding based models. *IEEE Access*, 7:53296–53304.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1199–1208. IEEE Computer Society.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2962–2971. IEEE Computer Society.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, volume 29, pages 3630–3638. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. 2018. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7343–7353.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7115–7123. PMLR.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1206–1215. Association for Computational Linguistics.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A comprehensive survey on transfer learning. *Proc. IEEE*, 109(1):43–76.

Documents Representation via Generalized Coupled Tensor Chain with the Rotation Group Constraint

Igor Vorona, Anh-Huy Phan, Alexander Panchenko and Andrzej Cichocki
Skolkovo Institute of Science and Technology, Moscow, Russia
{igor.vorona,a.phan, a.panchenko, a.cichocki}@skoltech.ru

Abstract

Continuous representations of linguistic structures are an important part of modern natural language processing systems. Despite the diversity, most of the existing log-multilinear embedding models are organized under vector operations. However, these operations can not precisely represent the compositionality of natural language due to a lack of order-preserving properties. In this work, we focus on one of the promising alternatives based on the embedding of documents and words in the rotation group through the generalization of the coupled tensor chain decomposition to the exponential family of the probability distributions. In this model, documents and words are represented as matrices, and n-grams representations are combined from word representations by matrix multiplication. The proposed model is optimized via noise-contrastive estimation. We show empirically that capturing word order and higher-order word interactions allows our model to achieve the best results in several document classification benchmarks.

1 Introduction

The current progress in natural language processing systems is largely based on the success of the representation learning of linguistic structures such as word, sentence and document embeddings. The most promising and successful methods are based on learning representations via two types of models: shallow log-multilinear models and deep neural networks. Despite the efficiency and interpretability of log-multilinear models, they can not use higher-order linguistic features like dependency between subsequences of words. To avoid these disadvantages, we usually use nonlinear predictors like recurrent, recursive, convolution neural networks, and more recently Transformers. Nevertheless, these methods can achieve high performance at the cost of loss of some interpretability and the cost of increased computation time.

However, there exist other ways to utilize higher-order interactions and still preserve the efficiency of linear models. In this research, we focus on more data-oriented, i.e., more linguistic grounded, and better interpretable methods which still can achieve high results in the practical tasks. Particularly, we investigate the matrix-space model of language, in which semantic space consists of square matrices of real values. The key idea behind this method goes from realization of the Frege's principle of compositionality through order-preserving property of matrix operations. As shown by Rudolph and Giesbrecht (2010), this type of models can internally combine various properties from statistical and symbolic models of natural language and therefore it is more flexible than vector space models.

In spite of that fact, such models are usually hard to optimize on the real data. To this end, Yessenalina and Cardie (2011) took attention to the needs of nontrivial initialization and proposed to learn the weights by the bag-of-words model. Asaadi and Rudolph (2017) used complex multi-stage initialization based on unigrams and bigrams scoring. Both approaches try to solve the sentiment analysis task. Recently Mai et al. (2019) considered the problem of self-supervised continuous representation of words via matrix-space models. They optimized a modified word2vec objective function (Mikolov et al., 2013) and proposed a novel initialization by adding small isotropic Gaussian noise to the identity matrix.

In this paper, we use a similarity function between matrices similar to Mai et al. (2019), but instead of neural network type of learning, we implement the model as the coupled tensor chain and impose the rotation group constraint. We focus on the document representation problem. Given a document collection, we try to find unsupervised doc-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1674–1684
August 1–6, 2021.

ument and word representation suitable for downstream linear classification tasks. To this end, we represent words, n-grams and documents as matrices and train self-supervised model. Our intuition is based on the fact that modeling interaction between words and documents is insufficient for modeling relations between complex phrases and documents.

Contributions. The main contributions of this work can be summarized as follows:

- To the best of our knowledge, this is the first representation learning method based on the Riemannian geometry of matrix groups.
- We show that our approach to model the compositionality and word order allows us to increase the quality of document embedding on downstream tasks. Moreover, it is also more computationally efficient in comparison with neural network models.
- Our model achieves state-of-the-art performance on the task of representation learning for multiclass classification both on short and long document datasets.

Our implementation of the proposed model is available online[1].

2 Related work

Euclidean embedding models (Mikolov et al., 2013; Pennington et al., 2014) based on implicit word-context co-occurance matrix factorization (Levy and Goldberg, 2014) are an important framework for current NLP tasks. Proposed models achieve relatively high performance in various NLP tasks like text classification (Kim, 2014), named entity recognition (Lample et al., 2016), machine translation (Cho et al., 2014).

Riemannian embedding models have shown promising results by expanding embedding methods beyond Euclidean geometry. There are several models with negative sectional curvature like Poincare (Dhingra et al., 2018; Nickel and Kiela, 2017) and Lorentz models (Nickel and Kiela, 2018). Furthermore, Meng et al. (2019) proposed a text embedding model based on spherical geometry.

Tensor decomposition models have been applied to many tasks in the NLP. Particularly, Van de Cruys et al. (2013) proposed the Tucker model for decomposing subject-verb-object co-occurance tensor for computation of compositionality. The most similar to our task is the word embedding problem. In this direction, Sharan and Valiant (2017) explored the Canonical Polyadic Decomposition (CPD) of word triplet tensor. Bailey and Aeron (2017) used symmetric CPD of pointwise mutual information tensor. Frandsen and Ge (2019) extended the RAND-WALK model (Arora et al., 2016) to word triplets. The main drawback of existing approaches is that they can not preserve word order information of long n-grams properly. For example, in the case of the CPD, we need to use separate parameters for each word based on its position in the text. This restriction does not allow us to efficiently use the linguistic meaning of tensor modes. The symmetric CPD completely loses word order information and the Tucker model suffers from exponentially increasing parameter size in the case of long length n-grams. Our approach eliminates these disadvantages.

3 Problem and model description

In this section, we describe our model for the document representation task. We begin with a short introduction of the multilinear algebra, then present the proposed document modeling framework in the view of the coupled tensor decomposition and provide the detailed description of our model and indicate benefits/drawbacks which are related to rotational group constraints.

3.1 Basic multilinear algebra

A tensor is a higher-order generalization of vectors and matrices to multiple indices. The order of a tensor is the number of dimensions, also known as modes or ways. An N-th order tensor is represented as $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and its element is denoted as $x_{i_1 \ldots i_N}$. We can always represent a tensor $\underline{\mathbf{X}}$ as sum of rank-1 tensors, where each of them is defined as outer product of N-vectors, i.e., $\mathbf{a}^{(1)} \circ \cdots \circ \mathbf{a}^{(N)}$ and $\mathbf{a}^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, N$. The minimal number of rank-1 tensors in this sum defines tensor rank.

In this research we focus on a particular type of tensor decomposition called tensor chain (Perez-Garcia et al., 2007; Khoromskij, 2011; Zhao et al., 2019). It represents tensor via the following sum of rank-1 tensors

$$\underline{\mathbf{X}} = \sum_{r_1, \ldots, r_N = 1}^{R_1, \ldots, R_N} \mathbf{a}^{(1)}_{r_1 r_2} \circ \cdots \circ \mathbf{a}^{(N)}_{r_N r_1}, \qquad (1)$$

---

[1] `https://github.com/harrycrow/RDM`

Notation	Description
a	Scalar
$\mathbf{a}$	Vector or tuple
$\mathbf{A}$	Matrix
$\underline{\mathbf{A}}$	Higher-order tensor
$\circ$	Outer (tensor) product
$\otimes$	Kronecker product
$\mathrm{tr}(\cdot)$	Trace of matrix
$\mathrm{vec}(\cdot)$	Vectorization of tensor
$[\cdot]_+$	Euclidean projection onto nonnegative orthant, i.e., $\max\{0, \cdot\}$
$\mathrm{QR}(\cdot)$	QR decomposition

Table 1: Basic notation.

where $R_1, \dots, R_N$ are called ranks of the TC model and the element-wise form of the following decomposition is given by

$$x_{i_1 i_2 \dots i_N} = \mathrm{tr}(\mathbf{A}^{(1)}_{i_1}\mathbf{A}^{(2)}_{i_2}\cdots\mathbf{A}^{(N)}_{i_N}), \qquad (2)$$

and $\mathbf{A}^{(n)}_{i_n} \in \mathbb{R}^{R_n \times R_{n+1}}$ represents the i_n-th frontal slice of the core tensors $\underline{\mathbf{A}}^{(n)} \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$ for $n = 1, \dots, N$ and $R_{N+1} = R_1$ and $a^{(n)}_{r_n r_{n+1}}$ are tubes of $\underline{\mathbf{A}}^{(n)}$.

3.2 Document modelling setting

In our research, we use the fact that the same text can be represented in different ways via different sets of n-grams with fixed lengths $\{\mathbb{W}^n\}_{n=1}^N$, where $\mathbb{W}^n = \underbrace{\mathbb{W} \times \cdots \times \mathbb{W}}_{n}$ and $\mathbb{W}$ is the word set. The main hypothesis is that the occurance statistics of the each of these n-grams sets contains some new information about this text, which can not be extracted from any other n-grams set. If we combine information from all of these sets we can achieve better quality for our document embedding model.

Due to the fact that the occurrence of the sequence of words depends on the occurrence of each word from this sequence, it is reasonable to treat the distribution of each fixed-length n-grams set separately. Otherwise, by the reason of dependence between the length of word sequence and their frequency, small length n-grams can downweight the importance of long length n-grams. Thus the effect of higher-order interaction can become low. Also, we notice that consider $p(\mathbf{w})$ as a distribution over unordered sets is a quite restrictive assumption on the structure of the model due to the importance of the order of the words in the language semantics. For all these reasons, we work with each n-gram distribution as with the separate distribution of the single random variable rather than define joint distribution for all n-grams sets and assume that particular distribution for each n-gram set can be constructed through marginalization from this joint distribution.

Following this intuition for each n-gram set, n, we assign appropriate joint distribution, $p(\mathbf{w}, d)$, where $\mathbf{w} \in \mathbb{W}^n$ and $d \in \mathbb{D}$. We represent co-occurrence of each n-gram, $\mathbf{w} = (w_1, \dots, w_n)$, and each document, $d \in \mathbb{D}$, as $(n+1)$th-order tensor $\underline{\mathbf{X}}^{(\mathbf{w}d)} \in \mathbb{R}^{|\mathbb{W}| \times \cdots \times |\mathbb{W}| \times |\mathbb{D}|}$, where

$$x^{(\mathbf{w}d)}_{\mathbf{i}j} = \begin{cases} 1, & \text{if } \mathbf{i} = \mathbf{w}, j = d \\ 0, & \text{otherwise}. \end{cases} \qquad (3)$$

Then we represent probability $p(\mathbf{w}, d)$ as the mean of these tensors

$$\bar{\underline{\mathbf{X}}}^{(n)} = \mathbb{E}_{(\mathbf{w},d)\sim p(\mathbf{w},d)}\underline{\mathbf{X}}^{(\mathbf{w}d)}.$$

Note that co-occurrences of n-grams and documents define bipartite graphs between them and $\bar{\underline{\mathbf{X}}}^{(n)}$ can be interpreted as adjacency tensors of these graphs.

3.3 Proposed model

Following compositional matrix-space modelling approach we represent each word, $w \in \mathbb{W}$, as a matrix $\mathbf{U}_w \in \mathbb{R}^{R \times R}$, a n-gram, $\mathbf{w} \in \mathbb{W}^n$, as $\mathbf{U}_{\mathbf{w}} = \prod_{k=1}^{n} \mathbf{U}_{w_k}$, and each document, $d \in \mathbb{D}$, as a matrix $\mathbf{V}_d \in \mathbb{R}^{R \times R}$. To measure dependence between n-gram, $\mathbf{w}$, and document, d, we use the Frobenius inner product, defined as $\langle \mathbf{U}_{\mathbf{w}}, \mathbf{V}_d \rangle_F = \mathrm{tr}(\mathbf{U}_{\mathbf{w}}^T \mathbf{V}_d) = \mathrm{tr}(\mathbf{U}_{\mathbf{w}} \mathbf{V}_d^T) = \mathrm{vec}(\mathbf{U}_{\mathbf{w}})^T \mathrm{vec}(\mathbf{V}_d)$. We assume that embeddings organized accordingly to this operation can be suitable for linear classifiers.

The resulting model is the generalized to exponential family of probability distributions coupled tensor decomposition (Collins et al., 2001; Yilmaz et al., 2011) of the set of tensors $\{\bar{\underline{\mathbf{X}}}^{(n)}\}_{n=1}^N$ by the corresponding set of tensor chain models with restricted set of parameters

$$\Theta = \{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|}, \qquad (4)$$

in the following optimization problem

$$\min_{\Theta} \sum_{n=1}^{N} \mathrm{KL}[\bar{\underline{\mathbf{X}}}^{(n)} || \hat{\underline{\mathbf{X}}}^{(n)}; \Theta], \qquad (5)$$

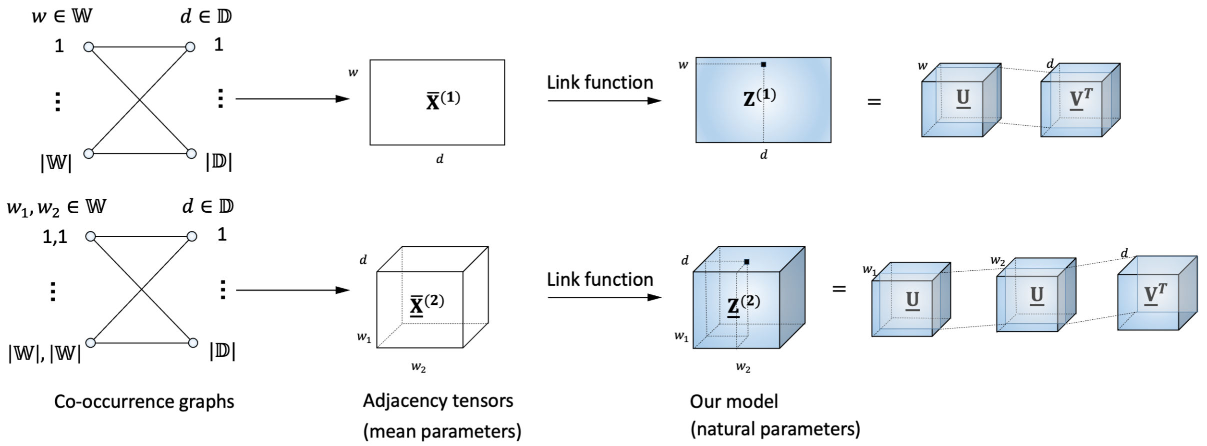

Figure 1: Illustration of representation of document collection in a multi-view way as a collection of bipartite graphs. Each of these graphs represents dependence (number of co-occurrence) between word's strings of length n and documents. Adjacency matrices $\bar{\mathbf{X}}^{(n)} \in \mathbb{R}^{|\mathbb{W}|^n \times |\mathbb{D}|}$ of these graphs can be appropriately tensorized to adjacency tensors $\underline{\bar{\mathbf{X}}}^{(n)} \in \mathbb{R}^{|\mathbb{W}| \times \cdots \times |\mathbb{W}| \times |\mathbb{D}|}$ which can be linked through modified multinomial link function with latent tensors $\underline{\mathbf{Z}}^{(n)} \in \mathbb{R}^{|\mathbb{W}| \times \cdots \times |\mathbb{W}| \times |\mathbb{D}|}$. Latent tensors can be decomposed via the Coupled Tensor Chain model. In our model, all core tensors $\underline{\mathbf{U}}$ ($\underline{\mathbf{V}}^T$) which represent words (documents) are additionally restricted to have the same parameters $\{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|}$ ($\{\mathbf{V}_d^T\}_{d=1}^{|\mathbb{D}|}$).

where each Kullback-Leibler divergence can be expressed as

$$\mathrm{KL}[\underline{\bar{\mathbf{X}}}^{(n)} || \underline{\hat{\mathbf{X}}}^{(n)}; \Theta] = \sum_{\mathbf{w} \in \mathbb{W}^n} \sum_{d \in \mathbb{D}} \bar{x}_{\mathbf{w}d}^{(n)} \log \left(\frac{\bar{x}_{\mathbf{w}d}^{(n)}}{\hat{x}_{\mathbf{w}d}^{(n)}} \right)$$

$$= \sum_{w_1=1}^{|W|} \cdots \sum_{w_n=1}^{|W|} \sum_{d=1}^{|D|} \bar{x}_{w_1 \ldots w_n d}^{(n)} \log \left(\frac{\bar{x}_{w_1 \ldots w_n d}^{(n)}}{\hat{x}_{w_1 \ldots w_n d}^{(n)}} \right),$$

and $\hat{x}_{\mathbf{w}d}^{(n)} = p(\mathbf{w}, d)$.

Our model represents $p(\mathbf{w}, d)$ by using following mean function

$$p(\mathbf{w}, d) \propto p(\mathbf{w}) p(d) \exp(\mathrm{vec}(\underline{\mathbf{X}}^{(\mathbf{w}d)})^T \mathrm{vec}(\underline{\mathbf{Z}}^{(n)})),$$

where $\mathrm{vec}(\underline{\mathbf{X}}^{(\mathbf{w}d)})^T \mathrm{vec}(\underline{\mathbf{Z}}^{(n)}) = z_{\mathbf{w}d}^{(n)}$ is one of natural parameters, which organized in the latent tensors $\underline{\mathbf{Z}}^{(n)} \in \mathbb{R}^{|\mathbb{W}| \times \cdots \times |\mathbb{W}| \times |\mathbb{D}|}$. This latent tensors contain pointwise mutual information between n-grams and documents and we assume that each of this tensor has low tensor chain rank, i.e., $z_{\mathbf{w}d}^{(n)} = \mathrm{tr}(\mathbf{U}_{\mathbf{w}} \mathbf{V}_d^T)$.

3.4 Intuition from geometric interpretation

If we avoid generative assumptions (Saunshi et al., 2019), our task can be interpreted as maximizing the similarity between document d and n-grams from its document distribution $p(\mathbf{w}|d)$ with simultaneous minimization of similarity between this document and n-grams from common n-gram distribution $p(\mathbf{w})$. As shown in previous works (Kumar and Tsvetkov, 2019; Meng et al., 2019) enforcing the spherical geometry constraints is a promising choice for tasks focusing on directional similarity. For doing so it can be reasonable to constrain our model to the orthogonal group. In this case Frobenius inner product became proper similarity measure and the sequential matrix product always preserves fixed norm and group structure (i.e., invertibility of matrix multiplication). Due to the group structure, our model has an interesting property to uniquely determine each word in the n-gram by their left and right aggregated context matrices and general n-gram matrix.

However, orthogonal group is a disjoint set of two connected components: set of rotations

$$\mathrm{SO}(R) = \{\mathbf{A} | \mathbf{A}^T \mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}, \det(\mathbf{A}) = +1\}$$

and set of reflections $\{\mathbf{A} | \mathbf{A}^T \mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}, \det(\mathbf{A}) = -1\}$. We impose constraints on our model parameters enforcing rotation matrices since the product of any number of rotations is always rotation, i.e. rotation set forms a matrix group. While the product of an even number of reflections becomes rotation.

3.5 Noise-contrastive estimation

In practice we do not need to construct set of tensors $\{\underline{\bar{\mathbf{X}}}^{(n)}\}_{n=1}^N$ explicitly. Instead, since each $\underline{\bar{\mathbf{X}}}^{(n)}$ represents a higher-order frequency table, we can optimize the sum of MLE tasks:

$$\mathbb{E}_{w,d\sim\underline{\bar{\mathbf{X}}}^{(1)}} \log(\hat{x}^{(1)}_{wd}) + \cdots + \mathbb{E}_{\mathbf{w},d\sim\underline{\bar{\mathbf{X}}}^{(N)}} \log(\hat{x}^{(N)}_{\mathbf{w}d}).$$

Usually, we have huge amount of data which make problem of computing partition function for each $\hat{x}^{(n)}_{\mathbf{w}d}$ intractable for many current computing architectures. We can avoid this problem by using noise-contrastive estimation (Gutmann and Hyvärinen, 2012) for conditional model (Ma and Collins, 2018). Similar to Chen et al. (2017), we construct negative samples from our batch by connecting non-linked n-grams and documents. Finally, for parameter set

$$\Theta = \{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|} \cup \{\kappa^{(n)}\}_{n=1}^{N}, \tag{6}$$

we formulate optimization problem in the following way:

$$\begin{aligned} \min_{\Theta} \ & \sum_{n=1}^{N} \mathcal{L}^{(n)}(\Theta) + \frac{\lambda}{2} \sum_{n=1}^{N} (\kappa^{(n)})^2 \\ \text{s.t. } \ & \mathbf{U}_w \in \mathrm{SO}(R), \quad w = 1, \ldots, |\mathbb{W}| \\ & \mathbf{V}_d \in \mathrm{SO}(R), \quad d = 1, \ldots, |\mathbb{D}| \\ & \kappa^{(n)} \geq 0, \quad n = 1, \ldots, N, \end{aligned} \tag{7}$$

where each risk function is equal to

$$\begin{aligned} \mathcal{L}^{(n)}(\Theta) = \ & \mathbb{E}_{\{(\mathbf{w}_i, d_i)\}_{i=1}^{I} \sim \prod_{i=1}^{I} \bar{x}^{(n)}_{\mathbf{w}_i d_i}} \\ & - \log \frac{\exp\left(\kappa^{(n)} \mathrm{tr}(\mathbf{U}_{\mathbf{w}_i} \mathbf{V}_{d_i}^T)\right)}{\sum_{j=1}^{I} \exp\left(\kappa^{(n)} \mathrm{tr}(\mathbf{U}_{\mathbf{w}_j} \mathbf{V}_{d_i}^T)\right)}. \end{aligned} \tag{8}$$

We add concentration parameters $\kappa^{(n)}$ to our loss function to overcome the problem of fixed scale. This makes our model more flexible to represent sharp distributions. Due to the fact that each n-gram distribution has its own scale, it can be reasonable to have a different $\kappa^{(n)}$ for different n-gram distributions.

4 Optimization setup

4.1 N-gram construction

We construct n-grams from text corpora by using sequentially moving of the sliding window of length n (from 1 to N) inside each document.

4.2 Parameters initialization

Initialization from the uniform distribution on the Stiefel manifold (Saxe et al., 2014) is one of promising ways to initialize deep neural network. To initialize parameters only from rotation component of Stiefel manifold we can swap two columns for each parameter matrix if the determinant of the this matrix is -1. However, this initialization can be below optimal way, because these rotation matrices can be far away from each other and due to the non-trivial structure of the loss function on this manifold we can stuck in local minima. To overcome this problem, we can fix particular point on the manifold for all matrices and perform small movement from this point in arbitrary direction. We use following strategy for each parameter $\mathbf{A} \in \{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|}$:

$$\begin{aligned} & \mathbf{A}_0 \sim \mathcal{N}\left(0, \frac{1}{R^2}\right) \\ & [\mathbf{Q}, \mathbf{R}] = \mathrm{QR}\left(\mathbf{I} + \frac{1}{2}(\mathbf{A}_0 - \mathbf{A}_0^T)\right) \\ & \mathbf{A} = \mathbf{Q}. \end{aligned} \tag{9}$$

We initialize all concentration parameters using following equation $\kappa^{(n)} = \frac{u}{R}$, where $u \sim \mathcal{U}(0.9, 1.1)$ and $n = 1, \ldots, N$.

4.3 Riemannian optimization

We solve our problem on the product manifold of rotation group and nonnegative orthant by simultaneous optimization of model parameters $\{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|}$ and $\{\kappa^{(n)}\}_{n=1}^{N}$ by Riemannian (Bécigneul and Ganea, 2019) and projected Adagrad (Duchi et al., 2011) respectively.

Let $\mathcal{M}$ be a real smooth manifold and $\mathcal{L} : \mathcal{M} \to \mathbb{R}$ a smooth real-valued function over parameters $\boldsymbol{\theta} \in \mathcal{M}$. Riemannian gradient descent (Gabay, 1982; Absil et al., 2008) based on two sequential steps. At first we compute Riemannian gradient by orthogonal projection of the Euclidean gradient on the tangent space at the same point on which we compute Euclidean gradient by $\mathrm{proj}_{\boldsymbol{\theta}_t} : \mathbb{R} \to \mathrm{T}_{\boldsymbol{\theta}}\mathcal{M}$

$$\nabla_R \mathcal{L}(\boldsymbol{\theta}_t) = \mathrm{proj}_{\boldsymbol{\theta}_t}(\nabla_E \mathcal{L}(\boldsymbol{\theta}_t)), \tag{10}$$

and then we perform movement on the manifold by specific curve, which called retraction $\mathrm{R}_{\boldsymbol{\theta}_t} : \mathrm{T}_{\boldsymbol{\theta}}\mathcal{M} \to \mathcal{M}$

$$\boldsymbol{\theta}_{t+1} = \mathrm{R}_{\boldsymbol{\theta}_t}(-\alpha \nabla_R \mathcal{L}(\boldsymbol{\theta}_t)). \tag{11}$$

For optimization on rotation group the orthogonal projector of matrix $\mathbf{G} \in \mathbb{R}^{R \times R}$ on the tangent space at the point $\mathbf{A} \in \mathrm{SO}(R)$ is given by:

$$\mathrm{proj}_{\mathbf{A}}(\mathbf{G}) = \frac{1}{2}(\mathbf{G} - \mathbf{A}\mathbf{G}^T\mathbf{A}), \tag{12}$$

For movement on the manifold in this direction we use QR-based retraction:

$$\begin{aligned}[\mathbf{Q}, \mathbf{R}] &= \mathrm{QR}\left(\mathbf{A}_t - \alpha \nabla_R \mathcal{L}(\mathbf{A}_t)\right) \\ \mathbf{A}_{t+1} &= \mathbf{Q}.\end{aligned} \tag{13}$$

We choose the QR-based retraction because it allows Riemannian Adagrad to achieve the fastest convergence in our experiments in comparison with Cayley retraction (first-order), Polar retraction (second-order), and geodesic (matrix exponential).

Algorithm 1 Optimization algorithm for RDM

```
Input: Learning rates $\alpha$ and $\beta$, number of iterations $T$, maximum n-gram length $N$.
Output: Embedding parameters $\{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|}$ and $\{\kappa^{(n)}\}_{n=1}^{N}$.
for $t = 1$ to $T$ do    ▷ after computing gradients:
    for all $\mathbf{A}_t \in \{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|}$ do
        $\nabla_E \mathcal{L}(\mathbf{A}_t) = \sum_{n=1}^{N} \nabla_E \mathcal{L}^{(n)}(\mathbf{A}_t)$
        $\nabla_R \mathcal{L}(\mathbf{A}_t) \leftarrow$ (12) with $\nabla_E \mathcal{L}(\mathbf{A}_t)$
        $\alpha_t = \frac{\alpha}{\sqrt{\sum_{i=1}^{t} \|\nabla_R \mathcal{L}(\mathbf{A}_i)\|_F^2}}$
        $\mathbf{A}_{t+1} \leftarrow$ (13) with $\alpha_t$
    end for
    for $n = 1$ to $N$ do
        $\mathrm{grad}(\kappa_t^{(n)}) = \nabla_E \mathcal{L}^{(n)}(\kappa_t^{(n)}) + \lambda \kappa_t^{(n)}$
        $\kappa_{t+\frac{1}{2}}^{(n)} = \kappa_t^{(n)} - \beta \frac{\mathrm{grad}(\kappa_t^{(n)})}{\sqrt{\sum_{i=1}^{t} \mathrm{grad}^2(\kappa_i^{(n)})}}$
        $\kappa_{t+1}^{(n)} = \left[\kappa_{t+\frac{1}{2}}^{(n)}\right]_+$
    end for
end for
```

4.4 Computational efficiency

The computational complexity of our model depends on the complexity R^3 of multiplication of matrices of size $R \times R$, and QR decomposition $\frac{4}{3}R^3$ (Layton and Sussman, 2020; Trefethen and III, 1997). Due to the number of elements in these matrices $d = R^2$, we can transform the complexity of our model in the dimension of embedding. In this view, the time complexity is $O(kd^{1.5})$, where k is the size of the context window. As shown in Table (2), our model has computational benefits in comparison with Transformer due to the linear dependence of time complexity on the word sequence length. We note that in comparison with Bi-LSTM models like ELMo, our model has lower complexity on embedding dimension, and can be computed in parallel using the associativity property of matrix multiplication. Although our model has a higher theoretical time complexity than the vector space models, the real gap between them is relatively small at ordinary embedding dimension (~ 400).

Method	**Time**	**Space**
PV-DBOW	$O(kd)$	$O(d)$
PV-DM (Concat)	$O(kd)$	$O(kd)$
ELMo	$O(kld^2)$	$O(ld^2)$
BERT	$O(k^2hld)$	$O(khld)$
RDM (Ours)	$O(kd^{1.5})$	$O(d)$

Table 2: Comparison of time and space complexity of several document embedding models, where k - size of context window, d - embedding dimension, l - number of layers, h - number of heads. The time complexity of other discussed here vector space models is equivalent to the complexity of PV-DBOW.

5 Numerical experiments

5.1 Experimental setup

Downstream Linear Protocol. We estimate quality of pre-trained representations on the multi-class document categorization tasks on the 20 Newsgroups and the ArXiv based long document dataset (He et al., 2019). We choose these datasets for our benchmarks because they are significantly different in the document's average length. This implies that statistics of long n-grams differ between these datasets too and in the case of the ArXiv dataset statistics of n-grams are significantly better converged than in the case of 20 newsgroups. This fact allow us to hypothesize that matrix-space models should less overfit on the ArXiv dataset.

We fix the document embeddings and optimize multinomial logistic regression with SAGA optimizer and l_2-norm regularization. Instead of test set we use nested 10-fold cross-validation to estimate statistical significance using Wilcoxon signed-rank test (Japkowicz and Shah, 2011; Dror et al., 2018). For each fold we estimate the hyperparameter of l_2-regularization on 10 point logarithmic grid from 0.01 to 100 by using additional 10-fold cross-validation with macro-averaged F1 score. For text preprocessing, we use CountVectorizer from the Scikit-learn package. Additionally we remove words which occur in the NLTK stopwords list or occur only in 1 document. In the case of ArXiv dataset we use half of this dataset and use only documents in the range from 1000 to 5000 words (smaller documents are removed and bigger documents are reduced to the first 5000 words).

Dataset	#cls	$\lvert\mathbb{W}\rvert$	$\lvert\mathbb{D}\rvert$	#w
20 Newsgroups	20	75752	18846	180
ArXiv	6	251108	16371	3829

Table 3: Datasets considered in the paper, where #cls - number of classes, #w - average number of words in documents, $\lvert\mathbb{W}\rvert$ - vocabulary size, $\lvert\mathbb{D}\rvert$ - number of documents.

Baselines. We compare our model with different vector space models: paragraph vectors (Le and Mikolov, 2014), weighted combinations of the word2vec skipgram vectors (Mikolov et al., 2013) (average, TF-IDF and SIF (Arora et al., 2017)), Doc2vecC (Chen, 2017), sent2vec (Pagliardini et al., 2018) and recently proposed JoSe (Meng et al., 2019). The comparison with the last two of these models seems to be more informative than with others because of some similarity of these models to our model (sent2vec can use n-gram information and JoSe also based on the spherical type of embedding geometry).

Due to the large number of possible values of hyperparameters for each model, we used the default values proposed by the authors of these models or proposed in subsequent studies of these models like in the case of paragraph vectors (Lau and Baldwin, 2016). We modify only the min_count to 1 and window size to be equal to the number of negative samples for paragraph2vec and word2vec models because it gives better results for these methods in our experiments. We choose n-grams number equal to 1 for sent2vec, because other values doesn't improve results. To preserve the fairness of comparison we use fixed embedding dimension, number of negative samples, and number of epochs for all models including ours. These values try to mimic the usual values of these hyperparameters in practice.

We compare our model not only with vector models but also with neural network models. We use 5.5B ELMo (Peters et al., 2018) version which is pre-trained on Wikipedia (1.9B) and all of the monolingual news crawl data from WMT 2008-2012 (3.6B). ELMo embedding dimension is equal to 1024. Also we use 768-dimensional embedding vectors from BERT model "bert-base-uncased". Following Devlin et al. (2019) we take the last layer hidden state corresponding to the [CLS] token as the aggregate document representation. If length of document is bigger than 512 we cut document on 512-length parts and average representation of this parts. Finally, we add Sentence BERT (Reimers and Gurevych, 2019) to the baseline models. This model is fine-tuned on SNLI and MultiNLI datasets for sentence embedding generation. We use 768-dimensional embedding vectors from model "bert-base-nli-mean-tokens".

We do not perform fine-tuning of BERT and ELMo models for our datasets, because in our experiments it doesn't give any positive effect on the final performance of these models. However, this is not true for Sentence BERT. Fine-tuning slightly improve performance of this model on the 20 newsgroup (50 epoch with maximum margin triplet loss).

We should notice that this experimental design gives some benefits to neural network models in comparison with log-multilinear models, but it is more consistent with the ordinary practical use case of the Transformers and RNN models. However, the next experiment will show that log-multilinear models can still outperform pre-trained neural networks.

Ablation study. For ablation study, we use different settings of our model. RDM means rotation document model, i.e. our model. RDM-R means our model without rotation group constraints. By (1) we mean model which utilize only unigrams and documents co-occurance information. By (3) and (5) we mean model which utilizes information from (1, 2, 3)-grams and (1, 2, 3, 4, 5)-grams respectively. For RDM, we use 1e-2 and 1e-3 as learning rates of Radagrad and projected Adagrad respectively and we use $\lambda = 15$ for 20 newsgroups and change λ to 5 for ArXiv dataset due to smaller number of epoch in this experiment. For RDM-R we use 1e-2 as learning rate for Adagrad.

5.2 Experimental results and comparison of performance

Comparison to baselines. As one may observe in the Table (4), our models yield results comparable or outperforming the baseline methods, including the simpler log-multilinear models (e.g. Skipgram) and more complex models featuring nonlinear transformations, such as recurrences (ELMo)

Models	20 Newsgroups				ArXiv			
	Acc	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
PV-DBOW	88.7	88.4	88.2	88.2	89.1	89.2	89.2	89.2
PV-DM	77.2	76.8	76.5	76.5	42.2	42.3	42.0	41.9
Skipgram+Average	90.3	90.2	90.0	90.0	92.4	92.3	92.3	92.3
Skipgram+TF-IDF	90.4	90.2	90.1	90.1	92.6	92.5	92.4	92.4
Skipgram+SIF	90.4	90.2	90.1	90.1	92.4	92.3	92.3	92.3
Sent2vec	87.9	87.6	87.5	87.5	91.9	91.7	91.7	91.7
Doc2vecC	90.0	89.8	89.7	89.7	93.2	93.1	93.1	93.1
JoSe	87.8	87.6	87.4	87.4	91.3	91.3	91.2	91.2
ELMo	79.2	78.8	78.8	78.7	91.6	91.5	91.4	91.4
BERT	74.3	73.6	73.6	73.5	92.3	92.2	92.1	92.1
Sentence BERT	79.5	79.1	79.0	79.0	89.0	88.9	88.8	88.8
RDM-R (1)	86.8	86.4	86.3	86.3	94.0*	93.9*	93.8*	93.8*
RDM-R (3)	87.9	87.6	87.4	87.4	94.5*	**94.4***	**94.4***	**94.4***
RDM-R (5)	88.3	88.0	87.9	87.9	**94.6***	**94.4***	**94.4***	**94.4***
RDM (1)	89.3	89.2	89.0	89.0	94.0*	93.9*	93.9*	93.9*
RDM (3)	90.7	90.5	90.4	90.4	94.0*	93.9*	93.9*	93.9*
RDM (5)	**91.1***	**90.9***	**90.8***	**90.8***	94.0*	93.9*	93.9*	93.9*

Table 4: Text classification performance on the 20 Newsgroups (short documents) and on the modified ArXiv (long documents) datasets. We fix the number of epochs to 50, the embedding dimension to 400, and the number of negative samples to 15 for all models on the 20 Newsgroups. On the ArXiv dataset, we use the same hyperparameters except for only the number of epochs which is equal to 5. We use macro-average for Precision, Recall, and F1.

and transformer blocks (BERT, SentenceBERT). More specifically, on the 20 newsgroups, the RDM (5) model yields the best results significantly[2] outperforming all the listed baseline approaches. It is interesting that contrary to the 20 Newsgroups, all RDM variants with any number of the n-grams sets show strong results and significantly outperform other models on the ArXiv dataset. Weighted combinations of the skipgram vectors and doc2vecC model achieve the closest to our result. This confirms that neural models like ELMo and BERT, are not the best way for all datasets and log-multilinear models can outperform them. We can see that performance of nonlinear models increase if we use a large document dataset and BERT can outperform some of the log-multilinear approaches (PV-DBOW, JoSe and sent2vec), but still, its result is not on the top level.

Comparison between our models. On observing the results we can see that our model increase the performance of classification when adding the n-gram set with a bigger length. This property has both models with rotation group constraints and without such constraints. Despite this fact, as we can see the model without rotation constraints is less robust in respect to noisy statistics of small document dataset and achieve performance less than the PV-DBOW model, while the rotation group model outperforms all other models. However, once we move on to a dataset with a larger document's average length (ArXiv), RDM-R performs better than all other models, including RDM. This confirms our hypothesis that the existence of good statistics of long n-grams has critical value for matrix space models. Due to strong associativity between sets of n-grams, our model needs more parameters to approximate all the co-occurrences. The RDM-R has a higher number of degrees of freedom than the RDM R^2 vs $\frac{R(R-1)}{2}$. We think that it allows RDM-R to outperform RDM in this experiment. This intuition can be also confirmed by the fact that RDM (1) and RDM-R (1) have the same performance level. And only if we increase the number of n-grams for the model from 1 to 3, then RDM-R can achieve better performance. However, if we increase the number of n-grams set from 3 to 5 both models stay on the same level of performance. This is the sign that we need to use a bigger embedding dimension if we want to achieve even better results.

---

[2] We use $*$ if the comparison of our model with the baseline models has $p < 0.05$.

In summary, if we have short documents, it's better to use RDM. For long document dataset with restriction on the embedding dimension, we suggest to relax the rotation group constraints. This trick allows RDM to use more degrees of freedom to estimate data precisely.

6 Conclusion

In this paper, we proposed a novel unsupervised representation learning method based on the generalized tensor chain with rotation group constraints, which can utilize higher-order word interactions and preserve most part of the computational efficiency and interpretability of vector-based models. Our model achieves state-of-the-art results in the document classification benchmarks on the 20 newsgroups and modified ArXiv dataset. A further direction of research could be focused on adding tensor kernel functions to the model to eliminate problems with dependence on the dimension of embedding. It could be interesting to augment this type of model with the different loss functions based not only on n-gram-document interactions but also on word-word interactions from the knowledge graph or document-document interaction from the citation graph of the documents.

Acknowledgments

The Zhores supercomputer (Zacharov et al., 2019) was used for computations. This research was partially supported by the Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001). The work of Alexander Panchenko was partially supported by the MTS-Skoltech laboratory.

References

Pierre-Antoine Absil, Robert E. Mahony, and Rodolphe Sepulchre. 2008. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to PMI-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Shima Asaadi and Sebastian Rudolph. 2017. Gradual learning of matrix-space models of language for sentiment analysis. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 178–185, Vancouver, Canada. Association for Computational Linguistics.

Eric Bailey and Shuchin Aeron. 2017. Word embeddings via tensor factorization. *CoRR*, abs/1704.02686.

Gary Bécigneul and Octavian-Eugen Ganea. 2019. Riemannian adaptive optimization methods. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 767–776. ACM.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. 2001. A generalization of principal components analysis to the exponential family. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 617–624. MIT Press.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1142–1151, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl. 2018. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 59–69, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Abraham Frandsen and Rong Ge. 2019. Understanding composition of word embeddings via tensor decomposition. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Daniel Gabay. 1982. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37:177–219.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13:307–361.

Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. 2019. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718.

Nathalie Japkowicz and Mohak Shah, editors. 2011. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press.

Boris N. Khoromskij. 2011. O(dlogn)-quantics approximation of n-d tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34:257–280.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Sachin Kumar and Yulia Tsvetkov. 2019. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany. Association for Computational Linguistics.

William Layton and Myron Sussman. 2020. *Numerical Linear Algebra*. WSPC.

Quoc V. Le and Tomás Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, volume 27, pages 2177–2185. Curran Associates, Inc.

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3698–3707. Association for Computational Linguistics.

Florian Mai, Lukas Galke, and Ansgar Scherp. 2019. CBOW is not all you need: Combining CBOW with the compositional matrix space model. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance M. Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8206–8215.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013,* *Lake Tahoe, Nevada, United States*, pages 3111–3119.

Maximilian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

David Perez-Garcia, Frank Verstraete, Michael M. Wolf, and J. Ignacio Cirac. 2007. Matrix product state representations. *Quantum Info. Comput.*, 7(5):401–430.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 907–916, Uppsala, Sweden. Association for Computational Linguistics.

Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. 2019. A theoretical analysis of contrastive unsupervised representation learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5628–5637. PMLR.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Vatsal Sharan and Gregory Valiant. 2017. Orthogonalized ALS: A theoretically principled tensor decomposition algorithm for practical use. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3095–3104, International Convention Centre, Sydney, Australia. PMLR.

Lloyd N. Trefethen and David Bau III. 1997. *Numerical Linear Algebra*, first edition. SIAM: Society for Industrial and Applied Mathematics.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 172–182, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yusuf Kenan Yilmaz, Ali Taylan Cemgil, and Umut Simsekli. 2011. Generalised coupled tensor factorisation. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2151–2159.

Igor Zacharov, Rinat Arslanov, Maksim Gunin, Daniil Stefonishin, Andrey Bykov, Sergey Pavlov, Oleg Panarin, Anton Maliutin, Sergey Rykovanov, and Maxim Fedorov. 2019. Zhores — petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in skolkovo institute of science and technology. *Open Engineering*, 9(1):512–520.

Qibin Zhao, Masashi Sugiyama, Longhao Yuan, and Andrzej Cichocki. 2019. Learning efficient tensor representations with ring-structured networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 8608–8612. IEEE.

Improving Unsupervised Extractive Summarization with Facet-Aware Modeling

Xinnian Liang[1]*, **Shuangzhi Wu**[2], **Mu Li**[2] and **Zhoujun Li**[1]†
[1]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[2]Tencent Cloud Xiaowei, Beijing, China
{xnliang,lizj}@buaa.edu.cn; {frostwu,ethanlli}@tencent.com;

Abstract

Unsupervised extractive summarization aims to extract salient sentences from documents without labeled corpus. Existing methods are mostly graph-based by computing sentence centrality. These methods usually tend to select sentences within the same facet, however, which often leads to the facet bias problem especially when the document has multiple facets (i.e. long-document and multi-documents). To address this problem, we proposed a novel facet-aware centrality-based ranking model. We let the model pay more attention to different facets by introducing a sentence-document weight. The weight is added to the sentence centrality score. We evaluate our method on a wide range of summarization tasks that include 8 representative benchmark datasets. Experimental results show that our method consistently outperforms strong baselines especially in long- and multi-document scenarios and even performs comparably to some supervised models. Extensive analyses confirm that the performance gains come from alleviating the facet bias problem.

Document

Facet 1: Lampard was fired.
1. As Chelsea's winter had turned bleak, as whispers that Lampard, its inexperienced coach, might be drifting toward the edge grew louder …
2. Not for the manager — that Lampard was fired so soon after he was given such public backing illustrates, quite neatly, how little power fans have — but for the public itself.
3. Chelsea might, in truth, have fired Lampard earlier. His colleagues, certainly, have been fearing it for weeks.
4. That was Sunday afternoon. He was fired on Monday morning …

Facet 2: Fans support Lampard.
5. "In Frank We Trust," it read, white letters on a blue field … And underneath, three simple words: "Then. Now. Forever." …
6. But none — not even Mourinho — have retained the support of the fans quite so unanimously as Lampard.
7. Lampard's association with Chelsea runs long and deep enough that he has deep-seated, well-established connections with Chelsea's fans.

Facet 3: Many managers were fired of Abramovich's ruthless impatience.
8. Lampard was not the first manager at Roman Abramovich's Chelsea to come under what seemed, on the surface, to be an undue, premature sort of pressure. …
9. Some of those who have gone before Lampard have done so with sympathy, perceived as victims of Abramovich's ruthless impatience.

Baseline

2. Not for the manager — that Lampard was fired so soon after he was given such public backing illustrates, quite neatly, how little power fans have — but for the public itself.
3. Chelsea might, in truth, have fired Lampard earlier. His colleagues, certainly, have been fearing it for weeks.
4. That was Sunday afternoon. He was fired on Monday morning …

Gold Reference

3. Chelsea might, in truth, have fired Lampard earlier. His colleagues, certainly, have been fearing it for weeks.
6. But none — not even Mourinho — have retained the support of the fans quite so unanimously as Lampard.
9. Some of those who have gone before Lampard have done so with sympathy, perceived as victims of Abramovich's ruthless impatience.

Figure 1: Examples from New York Times. We selected part of key sentences from the source document to show in this table. "..." refers to the omissions of context sentences due to space limitation.

1 Introduction

Document summarization is the task of transforming a long document into a shorter version while retaining its most important content (Nenkova and McKeown, 2011).Existing extractive or abstractive methods are mostly in supervised fashion which rely on large amounts of labeled corpora (Cheng and Lapata, 2016; Nallapati et al., 2017; Gehrmann et al., 2018; Liu and Lapata, 2019a,b; Zhang et al., 2019; Wang et al., 2020). However, this is not available for different summarization styles, domains, and languages. Fortunately, recent work has shown successful practices on unsupervised extractive summarization (Radev et al., 2000; Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Schluter and Søgaard, 2015; Tixier et al., 2017; Zheng and Lapata, 2019; Xu et al., 2020; Dong et al., 2020). Compare with supervised ones, unsupervised methods 1). remove the dependency on large-scale annotated document-summary pairs; 2). are more general for various scenarios.

Graph-based models are commonly used in unsupervised extractive methods (Radev et al., 2000; Mihalcea and Tarau, 2004; Erkan and Radev, 2004). For example, Zheng and Lapata (2019) proposed a directed centrality-based method named PacSum

---

[*] Contribution during internship at Tencent Inc.
[†] Corresponding Author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1685–1697
August 1–6, 2021.

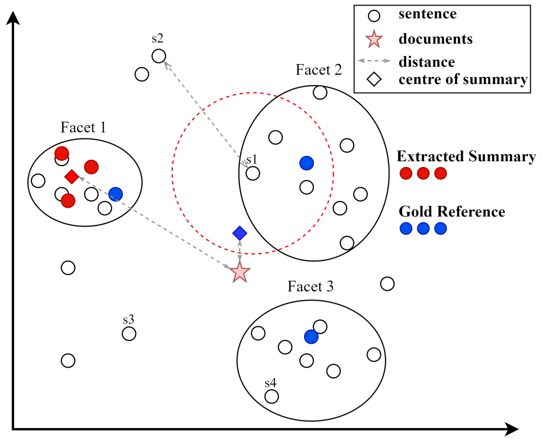

Figure 2: Visualization of facet bias. Nodes refer to sentence representations and star is the document representation. Black solid circles mean facets. Red dashed circle means threshold in Section 3.1. The dashed bi-direction arrows denote the sentence similarities.

by assuming that the contribution of any two nodes to their respective centrality is influenced by their relative position in a document. Dong et al. (2020) further improved PacSum by incorporating hierarchical and positional information into the directed centrality method. The core idea of centrality-based models is that the more similar a sentence is to other sentences, the more important it is (Radev et al., 2000). This usually works well for documents with a single facet (i.e. topic, aspect). However, there is always more than one facet, especially in long-document or multi-documents. Figure 1 shows an example of a long-document with 3 facets. We highlight the key phrases of each facet in different colors. Current centrality-based models often select sentences from one facet which is supported by more similar sentences. For example, the baseline model selects 3 sentences from facet 1. We call this the facet bias problem.

Figure 2 shows an intuitive explanation of the facet bias problem. The nodes are sentence representations, the star is the document representation and rhombuses are the centers of selected summary sentences. The sentences that support the same facet are masked in the same circle. Centrality-based models tend to select sentences from facet 1 (red nodes). Because these sentences are more similar to each other which leads to a higher centrality score. However, the true summary should consist of important sentences from different facets (blue nodes). To address the facet bias problem, in this paper, we proposed a facet-aware centrality-based model, which is called Facet-Aware Rank (FAR). First, we introduce a modified graph-based ranking method to filter irrelevant sentences. Then we encode the whole document into vector space which is used to capture all facets in the document. For each candidate summary, we calculate a similarity score between the summary sentences and the document. This sentence-document similarity aims at measuring the relevance between summary and document. Whereas the sentence centrality measures the sentence-level importance. In the ranking phase, we combine the sentence-document similarity and the sentence centrality to guarantee the selected sentences are important and cover all facets. As shown in Figure 2, by incorporating the sentence-document similarity, we are more likely to select the blue ones, that is closer to the star, instead of the red ones. We evaluate our method on 8 representative datasets. The results show that our model can surpass strong unsupervised baselines on most datasets and is comparable to supervised models on some datasets. Extensive analyses confirm that the performance gains indeed come from alleviating the facet bias problem. Besides, we surprisedly find that our method can tackle redundancy in summary to some extent.

2 Background: Graph-based Ranking

Given a document D, it contains a set of sentences $\{s_1, .., s_i, .., s_j, .., s_n\}$. Graph-based algorithms treats D as a graph $\mathcal{G} = (V, E)$. $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set where v_i is the representation of sentence s_i. E is the edge set, which is an $n \times n$ matrix. Each $= \{e_{i,j}\} \in E$ denotes the weight between vertex v_i and v_j.

The key idea of graph-based ranking is to calculate the centrality score of each sentence (or vertex). Traditionally, this score is measured by degree or ranking algorithms (Mihalcea and Tarau, 2004; Erkan and Radev, 2004) based on PageRank (Brin and Page, 1998). Then the sentences with the top score are extracted as a summary. The undirected graph algorithm compute the sentence centrality score as follows:

$$\texttt{Centrality}(s_i) = \sum_{j=1}^{N} e_{ij} \tag{1}$$

This is based on the assumption that the contribution of the sentence's importance in the document is not affected by the order of the sentence. In contrast, directed graph-based ranking algorithm takes the positional information into consideration, which is based on the assumption that the previous content of current sentence and the later contexts have different impact on current sentence's centrality score (Mann and Thompson, 1988). Then equation 1 is reformulated as

$$\mathtt{DC}(s_i) = \lambda_1 \sum_{i>j} e_{ij} + \lambda_2 \sum_{i<j} e_{ij} \tag{2}$$

Where $\lambda_1 + \lambda_2 = 1$. Hyper-parameters λ_1 and λ_2 were used to adjust the influence of previous and last content. Our method is built based on the directed graph-based ranking algorithm.

3 Facet-Aware Centrality-based Model

3.1 Modified Directed Graph-based Ranking

We propose a variation of directed graph-based ranking in this section. We modify Equation 2 in terms of filtering negligible sentences. We take s_1 in Figure 2 as an example to give an intuitive explanation. There usually exist many unrelated sentences especially in long documents for s_1 i.e. s_2, s_3, s_4. As shown in equation 2, all these sentences have a contribution in computing s_1's centrality score. We regard sentences like them as noise of s_1 and propose a modified directed graph-based ranking to filter them. To this end, we simply introduce a threshold ϵ to Equation 2. For s_1, ϵ can be seen as a diameter, s_1 is the centre. The centrality score of s_1 only consider nodes in red dashed circle. We further rewrite 2 as :

$$\begin{aligned}\mathtt{DC}(s_i) =& \lambda_1 \sum_{i>j} \mathtt{Max}((e_{ij} - \epsilon), 0) \\ &+ \lambda_2 \sum_{i<j} \mathtt{Max}((e_{ij} - \epsilon), 0)\end{aligned} \tag{3}$$

where $\epsilon = \beta \cdot (\max(e_{ij}) - \min(e_{ij}))$. β is a Hyper-parameter to control the scale of diameter. As shown in Equation 3, if the similarity between s_i and s_j is lower than ϵ, s_j is neglected. We find this modification is very effective but the model is very sensitive to the selection of β, so we carefully tune β on the development set. We finally rank and select sentences with Equation 4.

$$\mathtt{summary} = \mathtt{topK}(\{\mathtt{DC}(s_i)\}_{i=1,..,n}) \tag{4}$$

Where top-ranked k sentences will be extracted as summary and k is pre-defined with the average length of summary in training data.

3.2 Facet-Aware Centrality Scoring

In this section, we introduce how to implement Equation 3 and how we incorporate facet into centrality-based ranking in detail. We propose a simple method to model the facets in a document by a special representation based on the whole document.

Specifically, based on Equation 4, we add a sentence-document similarity, which computes the similarity between sentences in candidate summary and document to measure the relevance between summary $\mathcal{C}$ and document d. Candidate summary is pre-selected sentences from top-ranked K sentences with score $\mathtt{DC}(s_i)$ to reduce search range. We combine sentence-document similarity with sentence centrality and obtain the best candidate summary by 5.

$$\mathtt{summary} = \mathtt{arg}\max_{\mathcal{C}}(\mathtt{sim}(d, \hat{v}) \cdot \sum_{s_i \in \mathcal{C}} \mathtt{DC}(s_i)^{\alpha}) \tag{5}$$

where α is a hyper-parameter to control the influence of directed centrality. $\mathtt{sim}(d, \hat{v})$ refers to the sentence-document similarity, where d is the document representation and $\hat{v}$ is the candidate summary representation. $\hat{v}$ is obtained by $\frac{\sum_{i \in \mathcal{C}}(v_i)}{|\mathcal{C}|}$ which is the mean representation of summary sentences. We select the cosine similarity for $sim(\cdot)$.

The combination of sentence-document similarity and sentence centrality can not only tackle the facet problem but also reduce the redundancy to some extent. As shown in Figure 2, the centrality score of red nodes is extremely high due to they are similar to each other. Previous centrality-based models tend to select them as the summary. We incorporate document representation and sentence-document similarity to weight centrality score. This force model chooses the blue nodes, whose center is closer to the star, instead of red nodes. The introduction of sentence-document similarity makes it extremely unlikely that nodes of high cohesion will be selected. Thus, the redundancy is also reduced.

A candidate summary $\mathcal{C}$ is the subset of top-ranked K sentences after ranking with $\mathtt{DC}(s_i)$, which satisfy the following two conditions: 1) the length of sentences in candidate summary is pre-defined L, which is related to the summary length of dataset training data; 2) the total length of top-ranked K sentences is $t \times L$, where t is empirically set as 3. For the sentence representations v_i, we employ BERT as encoder which maps each word into a hidden state. Specifically, the sentence representations v_i is obtained by $\texttt{sigmoid}(h_i)$, where h_i is the hidden state of "[CLS]". Each e_{ij} in E is calculated by the dot product of the two sentences $v_i^\top v_j$. For document representation, we first collect all the sentence representations $\{v_1, v_2, \ldots, v_n\}$. To compress all the valuable information in the document, we apply a maxpooling function to sentence representations. The document representation d is computed as

$$d = \texttt{Maxpooling}(\{v_1, v_2, \ldots, v_n\}) \tag{6}$$

3.3 Improved Sentence Representation

The sentence representations plays a crucial role in our ranking model. The previous study shows that improving the quality of sentence representations helps improve the ranking performance (Zheng and Lapata, 2019; Dong et al., 2020). We post-train BERT on a sentence-level task constructed based on the corpus of a specific task. The idea is that its representation is affected not only by the words in it, but also the sentences around it. For a sentence in a document, we take its previous sentence and its following sentence to be positive examples and random sample sentences from documents as negative examples. The objective function follows that used in (Reimers and Gurevych, 2019). Specifically, for sentence s_i, a positive sentence s_j, and a negative sentence s_k, the BERT is trained to minimize the following equation:

$$\max(\parallel v_i - v_j \parallel - \parallel v_i - v_k \parallel + \mu, 0) \tag{7}$$

where v is the sentence representation, and μ is margin which ensures that v_j is at least μ closer to s_i than s_k. The hidden state vector of "[CLS]" is used as sentence representations and we set μ to 1 following (Reimers and Gurevych, 2019) in post-training phase.

4 Experiments

4.1 Datasets

We introduce the datasets used in our experiments in this section.

CNN/DM dataset contains 93k articles from CNN, and 220k articles from Daily Mail newspapers (Hermann et al., 2015). We use the non-anonymous version. Following (Zheng and Lapata, 2019), documents whose length of summaries are shorter than 30 tokens are filtered out.

NYT dataset contains articles published by the New York Times between January 1, 1987 and June 19, 2007 (Li et al., 2016). The summaries are written by library scientists. Different from CNNDM, salient sentences distribute evenly in an article (Durrett et al., 2016). We filter out documents whose length of summaries are shorter than 50 tokens (Zheng and Lapata, 2019).

MultiNews dataset consists of news articles and human-written summaries. The dataset is the first large-scale Multi-Documents Summarization (MDS) news dataset and comes from a diverse set of news sources (over 1500 sites) (Fabbri et al., 2019).

arXiv&PubMed datasets are two long document datasets of scientific publications from arXiv.org (113k) and PubMed (215k) (Cohan et al., 2018). The task is to generate the abstract from the paper body.

WikiSum dataset is a multi-documents summarization dataset from Wikipedia (Liu et al., 2018). We use the version provided by (Liu and Lapata, 2019a), which selects ranked top-40 paragraphs as input. For this dataset, we filter out documents whose summary length is less than 100 tokens. After the process, WikiSum test set contains 15,795 examples and the average length of summaries is 198.

WikiHow dataset is a large-scale dataset of instructions from the online WikiHow.com website (Koupaee and Wang, 2018). The task is to generate the concatenated summary-sentences from the paragraphs.

BillSum dataset contains US Congressional bills and human-written reference summaries from the 103rd-115th (1993-2018) sessions of Congress (Kornilova and Eidelman, 2019).

These datasets differ in scale, domain and task type. We collect details of the 8 corpus in Table 1.

4.2 Implementation Details and Metrics

FAR has 4 hyper-parameters and the best set of them are chosen from the following setting: $\alpha \in \{1, 2\}$, $\beta \in \{0.0, 0.1, \ldots, 0.9\}$, $\lambda_1 + \lambda_2 = 1$, $\lambda_1 \in \{0.0, 0.1, \ldots, 1.0\}$. In most case, FAR with the default setting $(\alpha = 1, \beta = 0.5, \lambda_1 = 0.5, \lambda_2 = 0.5)$ can achieve satisfied performance on all datasets. We select best hyper-parameters by sampling 1,000

Datasets	Sources	Type	#Pairs			#Tokens	
			Train	Valid	Test	Doc.	Sum.
CNN/DM	News	SDS	287,227	13,368	11,490	788	63
NYT	News	SDS	36,735	5,531	4,375	1,291	80
MultiNews	News	MDS	44,972	5,622	5,622	2,104	264
arXiv	Scientific Paper	LDS	202,914	6,436	6,440	4,938	220
PubMed	Scientific Paper	LDS	117,108	6,631	6,658	3,016	203
WikiSum	Wikipedia	MDS	1,579,360	38,144	38,144	2,800	139
WikiHow	Wikipedia	SDS	157,252	5,599	5,577	581	63
BillSum	US Legislation	LDS	17,054	1,895	3,269	2,148	209

Table 1: Information of datasets. The data in Doc. and Sum. indicates the average length of document and summary respectively. SDS represents single-document summarization, MDS represents multi-documents summarization and LDS represents single long document summarization (#tokens of document $\geq$ 3,000).

Method	CNN/DM			NYT			WikiHow		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Oracle	52.59	17.62	36.67	61.63	41.54	58.11	39.80	14.85	36.90
PTR-GEN	39.50	17.30	36.40	**42.70**	**22.10**	**38.00**	-	-	-
REFRESH	41.30	18.40	35.70	41.30	22.00	37.80	-	-	-
BertExt	**43.25**	**20.24**	**39.63**	-	-	-	**30.31**	**8.17**	**28.24**
Lead	40.49	17.66	36.75	35.50	17.20	32.00	24.31	5.52	22.53
TextRank	33.85	13.61	30.14	33.24	14.74	29.92	21.64	5.34	19.68
LexRank	34.68	12.82	31.12	30.75	10.49	26.58	25.46	5.89	23.63
MMR	31.63	10.02	28.55	27.16	6.41	25.32	22.02	4.40	20.22
PacSum	40.70	17.80	36.90	41.40	21.70	37.50	-	-	-
PacSum (Ours)	40.69	17.82	36.91	41.37	21.65	37.35	27.46	6.13	25.40
STAS	**40.90**	**18.02**	**37.21**	41.46	21.80	37.57	-	-	-
FAR	40.83	17.85	36.91	**41.61**	**21.88**	**37.59**	**27.54**	**6.17**	**25.46**

Table 2: Results on SDS CNNDM, NYT and WikiHow test sets.

examples from validation set (Zheng and Lapata, 2019).

The implementation of our encoder model is based on the PyTorch implementation of BERT[*]. The BERT follows the base settings. In the post-training, we employ basic BERT model to initialize our sentence encoder. We use Adam (Kingma and Ba, 2014) as our optimizer with a learning-rate of $2e^{-5}$. During post-training, we sample documents from training set of all datasets. The max length of the input sentence is set to 60. A linear warm-up for the first 10% of steps followed by a linear decay to 0 is used. The BERT encoder is post-trained on 6 Tesla V100 GPUs.

We use `ROUGE-1.5.5.pl` script[†] to evaluated summarization quality automatically with ROUGE F1 (Lin and Hovy, 2003). We report ROUGE-1/2/L score to measure the quality of summaries. Besides, we also do a human evaluation for the facet bias and redundancy of extracted summaries.

4.3 Results

Table 2-4 report the results of datasets with 3 types. In each table, we present the results of **Oracle** and previous supervised models in the first block. **Oracle** can be seen as the upper bound of extractive models, which extracts gold standard summaries by greedily selecting sentences to optimize the mean of ROUGE-1 and ROUGE-2 (Nallapati et al., 2017). We compare our approach with strong unsupervised baselines **Lead**, **TextRank** (Mihalcea and Tarau, 2004), **LexRank** (Erkan and Radev, 2004), **MMR** (Carbonell and Goldstein, 1998) in the second block of each table. **Lead** selects the first k tokens as a summary. We also report previous best centrality-based model **PacSum** (Zheng and Lapata, 2019) in the third block of each table.

Overall, FAR outperforms above-mentioned unsupervised strong baselines on most datasets, especially on long-document and multi-documents datasets and is more generalized than them for differnt types, domains datasets.

Results on SDS Table 2 reports the results on single document summarization (SDS) datasets CNN/DM, NYT and WikiHow. **PTR-GEN** (See et al., 2017) is a supervised abstractive model with classic seq2seq structure. **REFRESH** (Narayan et al., 2018) and **BertExt** (Liu and Lapata, 2019b) are supervised extractive models. **STAS** (Xu et al., 2020) is the best unsupervised model on CNN/DM

---

[*] https://github.com/huggingface/transformers

[†] https://github.com/andersjo/pyrouge

Method	arXiv			PubMed			BillSum		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Oracle	53.88	23.05	34.9	55.05	27.48	38.66	56.22	38.77	51.25
PTR-GEN	32.06	9.04	25.16	35.86	10.22	29.69	**33.43**	**9.47**	**27.90**
Discourse-aware	35.80	11.05	31.8	38.93	15.37	35.21	-	-	-
SummaRuNNer	42.81	16.52	28.23	43.89	18.78	30.36	-	-	-
GlobalLocalCont	**43.62**	**17.36**	**29.14**	**44.85**	**19.70**	**31.43**	-	-	-
Lead	33.66	8.94	22.19	35.63	12.28	25.17	35.10	16.76	30.31
TextRank	24.38	10.57	22.18	38.66	**15.87**	34.53	36.10	15.00	30.35
LexRank	33.85	10.73	28.99	39.19	13.89	34.59	38.28	16.02	32.44
MMR	29.75	6.14	26.41	37.65	10.61	33.71	36.73	12.45	32.13
PacSum	39.33	12.19	34.18	39.79	14.00	36.09	38.34	16.64	33.36
FAR	**40.92**	**13.75**	**35.56**	**41.98**	15.66	**37.58**	**38.37**	**16.69**	**33.40**

Table 3: Results on LDS arXiv, PubMed and BillSum test sets.

Method	MultiNews			WikiSum		
	R-1	R-2	R-L	R-1	R-2	R-L
Oracle	55.40	29.91	50.51	49.43	27.18	45.04
FT (2019)	**44.32**	**15.11**	**20.50**	40.56	25.35	34.73
HT (2019)	42.36	15.27	22.08	**41.53**	**26.52**	**35.76**
T-DMCA (2018)	-	-	-	40.77	25.60	34.90
HiMAP (2019)	44.17	16.05	21.38	-	-	-
Lead	39.41	11.77	14.51	37.63	14.75	34.76
TextRank	38.44	13.10	13.50	23.66	7.79	21.23
LexRank	38.27	12.70	13.20	36.12	11.67	22.52
MMR	38.77	11.98	12.91	31.22	10.24	22.48
PacSum (2019)	43.27	14.16	38.25	36.85	12.94	33.64
FAR	**43.48**	**16.87**	**44.00**	**38.11**	**14.54**	**35.01**

Table 4: Results on MDS MultiNews and WikiSum test sets.

and NYT with two redesigned pretrain tasks to measure the importance of sentences.

From the results, we can see that: 1) Our model outperforms all strong baselines in the second block and PacSum by wide margins in terms of ROUGE-1/2/L on 3 SDS datasets. 2) Especially on NYT, our model outperforms the previous best unsupervised extractive system STAS and supervised method REFERSH.

After we re-implement the trigram blocking trick (i.e., removing sentences with repeating trigrams to existing summary sentences) which STAS used (Xu et al., 2020), FAR can achieve a better ROUGE-1 score **40.93**/17.80/37.00 than STAS on CNN/DM.

Results on LDS Table 3 reports the results on long document summarization (LDS) datasets arXiv, PubMed and BillSum. For supervised extractive models, we compare with **SummaRuNNer** (Nallapati et al., 2017) and **GlobalLocalCont** (Xiao and Carenini, 2019). We also compare with supervised abstractive models **Discourse-aware** (Cohan et al., 2018) and **PRT-GEN**.

As shown in Table 3, our model has obviously higher ROUGE-1/2/L score (+1.89 +1.56 +1.38) on arXiv and (+2.22 +1.55 +1.45) on PubMed than PacSum. Compare with supervised models, our unsupervised model outperforms supervised abstractive models PTR-GEN and Discourse-aware, but still have a gap with supervised extractive models. The reason for this gap is that supervised extractive models can extract sentences with dynamic length through training with labeled corpus, but unsupervised models need to predefined the length or number of extracted summaries.

Besides, we can see that the improvement on Billsum is limited. We analysis the input document of Billsum and find that documents in Billsum contains many very short sentences which lead to this limited improvement.

Results on MDS Table 4 reports the results on multi-documents summarization datasets MultiNews and WikiSum. **T-DMCA** and **HiMAP** are proposed with the construction of WikiSum and MultiNews. **FT** (Flat Transformer) and **HT** (Hierarchical Transformer) are two supervised extractive models which are proposed by (Liu and Lapata, 2019a).

From results in Table 4, we can see that PacSum and FAR have a strong performance on MultiNews, which may result from the characteristic of news datasets and the high-quality human-written documents-summary pairs of MultiNews. On Wik-

arXiv			
FAR	**40.92**	**13.75**	**35.56**
-facet-aware scoring	39.61	12.45	34.37
-modified `DC`	38.32	11.53	33.35
-post-training	40.02	12.79	34.67
NYT			
FAR	**41.67**	**21.93**	**37.68**
-facet-aware scoring	40.82	21.10	36.81
-modified `DC`	39.90	20.47	36.02
-post-training	40.93	21.38	36.99

Table 5: Ablation study on arXiv and NYT.

iSum, compare with PacSum, FAR is obviously better. We also can observe that the performance of unsupervised models are far less than supervised models. Because the length of multi-document summary has a great fluctuation and unsupervised methods are hard to decide the length of extracted sentences.

5 Analysis

In this section, we present a series of analysis and tests to understand the improvements of our FAR reported in the previous section, and to prove that it fulfills our intuition that the design of our model improves the facet bias. We choose NYT from SDS and arXiv from LDS to analyze the performance of FAR. These 2 datasets are typical and cover the situation of short and long document inputs.

Ablation Study In order to access the contribution of 3 components of FAR – modified `DC` in section 3.1, facet-aware scoring in section 3.2, and post-training in section 3.3. We remove each component of them and report ablation study results in 5. We can see that modified `DC` and facet-aware scoring are indispensable to the performance of FAR. If we remove each of them, the performance of FAR drops sharply. When we replace BERT with post-training with original BERT, the results also confirm that post-training is usable.

Human Evaluation To evaluate the ability of FAR in reducing facet bias and redundancy, we asked 3 human annotators to evaluate the extracted summaries of PacSum and FAR with the gold reference summary. Three annotators were asked to give 0-2 scores for facet bias and redundancy of 100 random sampled examples. The results of PacSum in terms of facet bias is 1.42 and redundancy is 1.17. Our FAR performs significantly better than PacSum ($p < 0.05$) whose facet bias is **0.96** and redundancy is **0.81**. Human evaluation results indicated that FAR can extract high-quality summaries by facet-aware modeling and reduce redundancy of summaries to some extent.

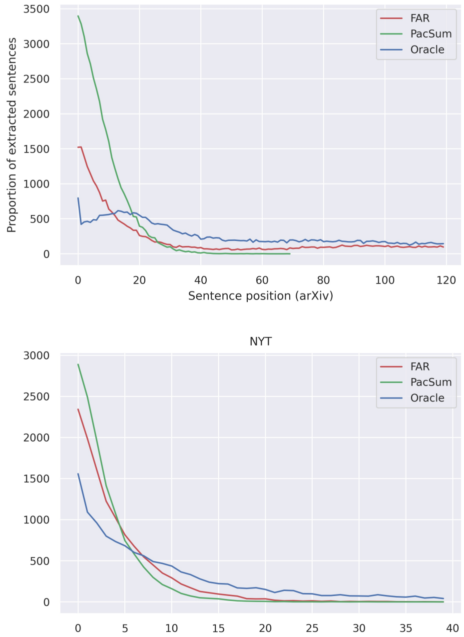

Figure 3: Sentence position distribution of arXiv and NYT. We use the first 40 sentences for NYT and the first 120 sentences for arXiv.

Sentence Position Distribution We compare the position distribution of extracted sentences of FAR, PacSum, and Oracle to further inspect the performance of FAR. We report the position distribution of extracted sentences in Figure 3. We can see that 1) The distribution of FAR is more close to Oracle; 2) PacSum only extracts sentence in the head of documents on arXiv, which is also mentioned by (Dong et al., 2020); 3) The advantages of our model are more significant for LDS datasets.

Analysis of Hyper-parameter β Hyper-parameter β is a crucial hyper-parameter that is used to filter out noise sentences in documents. We fixed other hyper-parameters and observed the change of ROUGE-1 from 0.1 to 0.9 with β in Figure 4. We can see that Hyper-parameter β has great impact on model's effect, especially on NYT dataset. These curves prove that noise sentences truly exists and hurt the performance of centrality-based models.

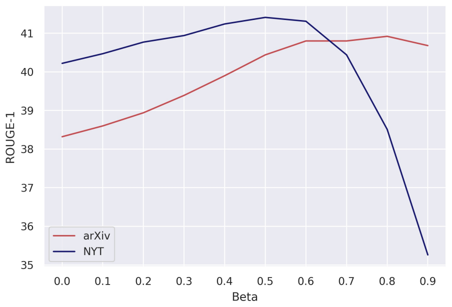

Figure 4: FAR's performance against different values of β on arXiv and NYT.

Case Study To intuitively show the ability of FAR to tackle the facet bias problem and reduce redundancy, we choose one typical example from NYT dataset. (*example is from a news report and only used to analyze the effectiveness of our model.*) As shown in Table 5, we can see that sentences extracted by PacSum all focus on the facet which describes terroristic attacks in Iraq. However, FAR can cover all 3 facets in gold reference. This shows that our FAR can effectively improve the performance by reducing the facet bias problem.

6 Related Work

Summarization is a long-standing challenge for researchers to address. Thanks to the power of the neural network and availability of large-scale parallel datasets. Supervised summarization algorithms develop sharply (Chopra et al., 2016; Cao et al., 2018; Zhang et al., 2018; Zhong et al., 2019; Gehrmann et al., 2019; Cho et al., 2019; Jin et al., 2020b; Cao et al., 2020; Jin et al., 2020a; Zhong et al., 2020). However, high-quality parallel datasets are not always available. Researches on unsupervised summarization are necessary, which can be diveided into extractive and abstractive. Unsupervised abstractive summarization is more challenging than extractive. There are also many interesting works (Wang and Lee, 2018; Févry and Phang, 2018; Baziotis et al., 2019; Jernite, 2019; Zhou and Rush, 2019; West et al., 2019; Chu and Liu, 2019; Yang et al., 2020) on unsupervised abstractive summarization.

However, most unsupervised summarization models are extractive (Radev et al., 2000; Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Carbonell and Goldstein, 1998; Wan, 2008; Wan and Yang, 2008; Schluter and Søgaard, 2015; Zhao et al., 2020) and focused on the measure of sentence salient. Graph-based models are effective and widely concerned in unsupervised extractive methods. Different from traditional undirected graph rank models (Radev et al., 2000; Mihalcea and Tarau, 2004; Erkan and Radev, 2004), (Zheng and Lapata, 2019) proposed directed centrality method, which is based on the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) assumption. (Dong et al., 2020) point out that PACSUM has position bias, which makes PACSUM not suitable for long document summarization, and proposed hierarchical position-based model HipoRankfor scientific document summarization. STAS (Xu et al., 2020) design two summarization tasks related pretraining tasks to improve sentence representation. Then they proposed a rank method which combines attention weight with reconstruction loss to measure the centrality of sentences.

We find the facet bias problem in graph-based models, which lead to the extracted summaries can not cover multi-facets information in document. A similar concept in summarization is redundancy. However, the difference between redundancy and facet bias is two folds: 1) to solve redundant problem, we just need to make sure selected sentences are not too similar; 2) However, to tackle the facet bias problem, we need to select sentences that can retain multi-facets information.

7 Conclusion

In this paper, we discover the facet bias problem in centrality-based unsupervised summarization models and proposed a novel facet-aware centrality-based ranking model FAR to tackle it. We introduce a sentence-document weight into centrality, which forced the model to pay more attention to different facets and find that FAR can reduce redundancy to some extent. Results on a wide range of summarization tasks show that our method consistently outperforms strong baselines especially in long- and multi-document scenarios, which prove our model is robust and effective. Extensive analyses confirmed that the performance gains of our model come from alleviating the facet bias problem.

Acknowledgments

We thank the three anonymous reviewers for their careful reading of our paper and their many insightful comments and suggestions. This work was supported in part by the National

Gold Reference
1. Marine corps lt col jeffrey r. chessani was battalion commander in haditha, iraq, when roadside bomb planted by sunni arab insurgents killed one of his marines and wounded two other, and when infantrymen under his command, seeking to engage enemy, instead killed 24 civilians.
2. Chessani and other officers are charged with dereliction of duty for failing to investigate episode properly and relieved of their command.
3. Military hearing at camp pendleton will decide if chessani should face formal court-martial. Witnesses and military documents paint two contradictory portraits of chessani, highest-ranking marine officer charged since iraq war began.

PacSum
1. In november 2005, colonel chessani was a battalion commander in haditha, iraq, when a roadside bomb planted by sunni arab insurgents killed one of his marines and wounded two others.
2. Through three combat deployments in iraq, a bronze star and numerous combat ribbons, lt col jeffrey r. chessani's marine corps career has been defined, it seems, by terrorist bombs.
3. In october 1983, news of the attack by muslim extremists on a marine corps barracks in beirut that killed 241 service members compelled colonel chessani, then a teenager from rangely, colo., to embrace christianity and, later, to follow two brothers into the service.

FAR
1. In november 2005, colonel chessani was a battalion commander in haditha, iraq, when a roadside bomb planted by sunni arab insurgents killed one of his marines and wounded two others.
2. Last year, the marine corps charged colonel chessani, 43, and three other officers with dereliction of duty for failing to investigate the episode properly and relieved him of his command.
3. In a military hearing into whether he should face a formal court-martial, witnesses and military documents have helped paint two contradictory portraits of colonel chessani, the highest-ranking marine officer charged since the iraq war began more than four years ago.

Figure 5: The examples come from New York Times dataset.

Natural Science Foundation of China (Grant Nos.U1636211, 61672081, 61370126), the 2020 Tencet Wechat Rhino-Bird Focused Research Program, and the Fund of the State Key Laboratory of Software Development Environment (Grant No.SKLSDE2019ZX-17).

References

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. SEQ^3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 673–681, Minneapolis, Minnesota. Association for Computational Linguistics.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117.

Yue Cao, Hui Liu, and Xiaojun Wan. 2020. Jointly learning to align and summarize for neural cross-lingual summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6220–6231, Online. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Sangwoo Cho, Logan Lebanoff, Hassan Foroosh, and Fei Liu. 2019. Improving the similarity measure of determinantal point processes for extractive multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1027–1038, Florence, Italy. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232, Long Beach, California, USA. PMLR.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Yue Dong, Andrei Romascanu, and Jackie CK Cheung. 2020. Hiporank: Incorporating hierarchical and positional information into graph-based unsupervised long document extractive summarization. *arXiv preprint arXiv:2005.00513*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Thibault Févry and Jason Phang. 2018. Unsupervised sentence compression using denoising autoencoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Gehrmann, Zachary Ziegler, and Alexander Rush. 2019. Generating abstractive summaries with finetuned language models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 516–522, Tokyo, Japan. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28, pages 1693–1701. Curran Associates, Inc.

Yacine Jernite. 2019. Unsupervised text summarization via mixed model back-translation. *arXiv preprint arXiv:1908.08566*.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020a. Multi-granularity interaction network for extractive and abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6244–6254, Online. Association for Computational Linguistics.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020b. Semsum: Semantic dependency guided neural abstractive summarization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8026–8033. AAAI Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Anastassia Kornilova and Vladimir Eidelman. 2019. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56, Hong Kong, China. Association for Computational Linguistics.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305.

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019b. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2011. *Automatic summarization*. Now Publishers Inc.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Natalie Schluter and Anders Søgaard. 2015. Unsupervised extractive summarization via coverage maximization with syntactic and semantic concepts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 840–844, Beijing, China. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Antoine Tixier, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Combining graph degeneracy and submodularity for unsupervised extractive summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 48–58, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaojun Wan. 2008. An exploration of document impact on graph-based multi-document summarization. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Honolulu, Hawaii. Association for Computational Linguistics.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, page 299–306, New York, NY, USA. Association for Computing Machinery.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.

Yaushian Wang and Hung-Yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4187–4195, Brussels, Belgium. Association for Computational Linguistics.

Peter West, Ari Holtzman, Jan Buys, and Yejin Choi. 2019. BottleSum: Unsupervised and self-supervised sentence summarization using the information bottleneck principle. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3752–3761, Hong Kong, China. Association for Computational Linguistics.

Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3011–3021, Hong Kong, China. Association for Computational Linguistics.

Shusheng Xu, Xingxing Zhang, Yi Wu, Furu Wei, and Ming Zhou. 2020. Unsupervised extractive summarization by pre-training hierarchical transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1784–1795, Online. Association for Computational Linguistics.

Ziyi Yang, Chenguang Zhu, Robert Gmyr, Michael Zeng, Xuedong Huang, and Eric Darve. 2020. TED: A pretrained unsupervised summarization model with theme modeling and denoising. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1865–1874, Online. Association for Computational Linguistics.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference* *on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium. Association for Computational Linguistics.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Jinming Zhao, Ming Liu, Longxiang Gao, Yuan Jin, Lan Du, He Zhao, He Zhang, and Gholamreza Haffari. 2020. Summpip: Unsupervised multi-document summarization with sentence graph compression. SIGIR '20, page 1949–1952, New York, NY, USA. Association for Computing Machinery.

Hao Zheng and Mirella Lapata. 2019. Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247, Florence, Italy. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1049–1058, Florence, Italy. Association for Computational Linguistics.

Jiawei Zhou and Alexander Rush. 2019. Simple unsupervised summarization by contextual matching. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5101–5106, Florence, Italy. Association for Computational Linguistics.

A Hyperparamters

Hyper-paramters of the FAR were reported in Table 6.

B Filter Summary Length of arXiv&PubMed

To prove unsupervised is limited by summary length, we filter examples in the test set with summary length, and report the results in Figure 6. We can see that when examples with short summary, which do not match the predefined length, were removed, the performance improved obviously.

Datasets	α	β	λ_1	λ_2
CNN/DM	1	0.0	0.7	0.3
NYT	1	0.6	0.6	0.4
arXiv	2	0.7	0.5	0.5
PubMed	2	0.3	0.5	0.5
MultiNews	1	0.4	0.5	0.5
WikiSum	1	0.0	0.5	0.5
BillSum	1	0.5	0.5	0.5
WikiHow	1	0.8	0.5	0.5

Table 6: Hyper-parameters for FAR's best performance.

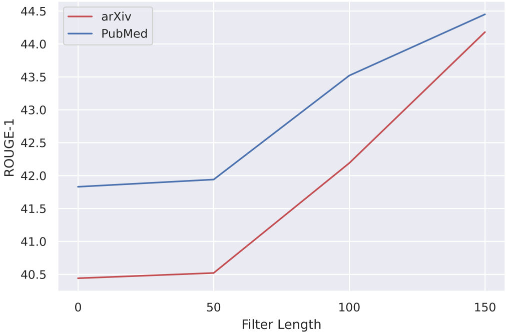

Figure 6: Performance on arXiv and PubMed, when we filter examples in test set with summary lenghth.

C Sentence Position Distribution

We show sentence position distribution of all 8 datasets in Figure 7.

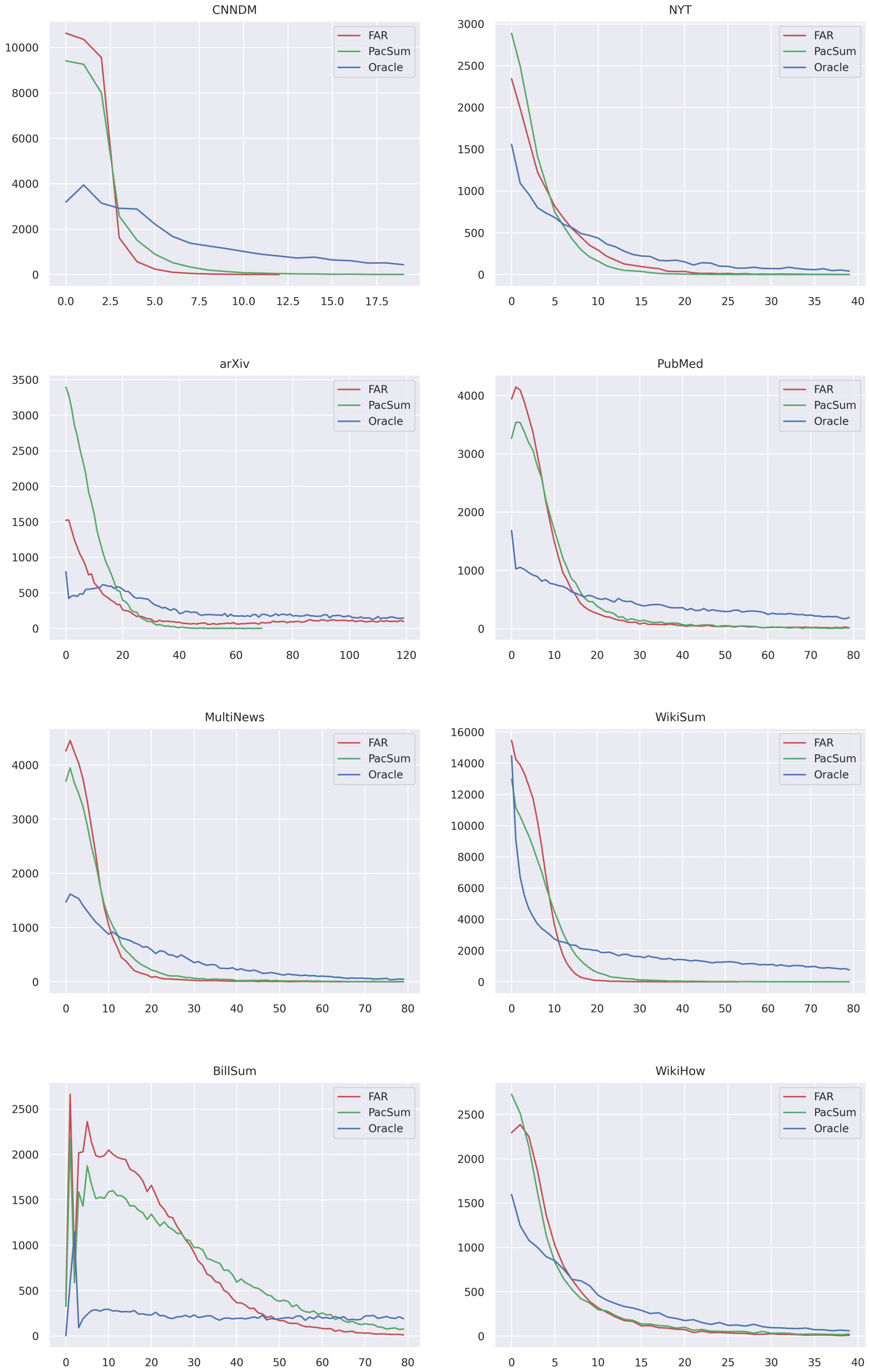

Figure 7: Sentence position distribution of 8 datasets.

Improving Gradient-based Adversarial Training for Text Classification by Contrastive Learning and Auto-Encoder

Yao Qiu, Jinchao Zhang, Jie Zhou
Pattern Recognition Center, WeChat AI, Tencent Inc, China
`{yasinqiu, dayerzhang, withtomzhou}@tencent.com`

Abstract

Recent work has proposed several efficient approaches for generating gradient-based adversarial perturbations on embeddings and proved that the model's performance and robustness can be improved when they are trained with these contaminated embeddings. While they paid little attention to how to help the model to learn these adversarial samples more efficiently. In this work, we focus on enhancing the model's ability to defend gradient-based adversarial attack during the model's training process and propose two novel adversarial training approaches: (1) CARL narrows the original sample and its adversarial sample in the representation space while enlarging their distance from different labeled samples. (2) RAR forces the model to reconstruct the original sample from its adversarial representation. Experiments show that the proposed two approaches outperform strong baselines on various text classification datasets. Analysis experiments find that when using our approaches, the semantic representation of the input sentence won't be significantly affected by adversarial perturbations, and the model's performance drops less under adversarial attack. That is to say, our approaches can effectively improve the robustness of the model. Besides, RAR can also be used to generate text-form adversarial samples.

1 Introduction

Text classification is a fundamental research topic in natural language processing (Pang et al., 2002; Lai et al., 2015; Neekhara et al., 2019; Sun et al., 2019). Neural networks have obtained state-of-the-art performance on many text classification datasets (Kim, 2014; Wang et al., 2018; Devlin et al., 2019). Despite these models' success, recent work has shown that they can be easily fooled by intentionally designed adversarial examples. These adversarial examples generated by adding little perturbations on original examples cannot affect human's judgment but can fail models (Ren et al., 2019a; Xu et al., 2019).

Adversarial training approaches are proposed to tackle this problem, which aims to enhance the model's strength of generalization and robustness by generating adversarial samples and letting the model learn them (Ren et al., 2019b; Xu et al., 2019). The approaches for generating adversarial samples can be roughly classified into two categories: text-based and gradient-based. The former can be further classified into three levels: character-level, word-level, and sentence-level. Compared to gradient-based adversarial approaches, the text-based are explainable, but they may suffer from low attack diversity and rely more on human knowledge which limits the kinds of adversarial patterns. In contrast, during the gradient-based adversarial training process, small perturbations calculated from the gradient are added to mini-batches embeddings of original training samples, then the model's parameters will be optimized to correctly classify the original embeddings together with adversarial embeddings (Miyato et al., 2017). This kind of approach consists of two major steps: adversarial perturbation's construction and adversarial sample's learning. Recent approaches mainly focus on the first step, as for the second step, only the classification loss is used by the model to learn the adversarial samples.

In this work, we investigate gradient-based adversarial training and focus on the second step. To further improve model's robustness against adversarial perturbations, we propose two approaches for text classification models: CARL (**C**ontrastive **A**dversarial **R**epresentation **L**earning) and RAR (**R**econstruction from **A**dversarial **R**epresentations). We first generate adversarial samples by adding perturbations on input sentence's word embeddings, then CARL and RAR are used to learn these adversarial samples. CARL leverages the family of contrastive objectives (Gut-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1698–1707
August 1–6, 2021.

mann and Hyvärinen, 2010; Hjelm et al., 2019; Tian et al., 2020) and aims to prevent the semantic representation of input sentence from being affected by adversarial attacks by narrowing the distance between the adversarial sample and its corresponding original sample in the representation space, while pushing them apart from samples which belong to different classes. If the representations of adversarial sample and original sample are identical, the model won't be fragile to the adversarial attack. While CARL's goal is to learn a robust sentence-level representation, RAR acts like an auto-encoder and is designed to improve the robustness of the representation for each word by forcing the model to reconstruct original words from their adversarial embeddings. It will be much easier for the model to understand the adversarial sample when it can recognize every adversarial word embedding correctly. We summarize our contributions in the following:

- We design a contrastive adversarial representation learning approach to learn adversarial examples in the representation space, which can directly improve the encoder's robustness.
- We propose a novel adversarial training task, RAR (Reconstruction from Adversarial Representations), to help the model learn a more robust representation at the word level.
- We conduct experiments on four text classification datasets and results show that our proposed approaches outperform strong baseline on accuracy and robustness. We release the source code at a GitHub repo.[1]

2 Related Work

Gradient-based Adversarial Training. Adversarial examples were explored primarily in the computer vision area and received more attention in natural language processing recently. Different from the CV domain, we can improve NLP models' robustness and performance at the same time (Miyato et al., 2017). Miyato et al. (2017) proposed to add perturbations calculated from gradient on word embeddings to obtain adversarial samples in embedding space. Madry et al. (2018) proposed the k-PGD method and calculated adversarial perturbations through multiple forward-backward iterations to avoid the obfuscated gradient problem. It is widely accepted as the most effective approach, but multiple iterations leads to high computation cost. To mitigate the cost, Zhang et al. (2019) restricted most perturbation updates in the first layer. Shafahi et al. (2019) designed a ”free” algorithm that simultaneously updates both model parameters and adversarial perturbations in a single backward pass. Zhu et al. (2020) proposed FreeLB which simultaneously accumulates the “free” parameter gradients in each iteration and updates the model parameters all at once after all iterations.

Contrastive Learning. Contrastive learning has recently become a dominant component in self-supervised learning methods for computer vision, natural language processing (NLP). The goal of contrastive learning is to learn a representation that is close in a certain metric space for pairs with the same label, while push apart the representation between pairs with different labels (Tian et al., 2020). This method has been successfully used in recent years for representation learning and knowledge distillation. In this work, we apply it into the adversarial training by narrowing the representations of the adversarial sample and its corresponding original sample, while enlarging their distance from samples that belong to different classes.

Auto-Encoder. The auto-encoder (Rumellhart, 1986) consists of two modules: the encoder and the decoder. The encoder is used to map the input sample x to the feature space z, i.e. the encoding process. Then the abstract feature z is mapped back to the original token space through a decoder to obtain the reconstructed sample x', i.e. the decoding process. The optimization goal is to optimize both encoder and decoder by minimizing the reconstruction error, to learn the abstract feature representation z for the input x.

In our work, we focus on the gradient-based adversarial training on the text classification where the model receives a sentence and outputs a single label. Though some neural networks have achieved promising results, they are vulnerable to the simple adversarial perturbations (Huang et al., 2017; Yuan et al., 2019). Some gradient-based adversarial training approaches were proposed to solve this problem (Zhu et al., 2020; Shafahi et al., 2019; Madry et al., 2018; Miyato et al., 2017). Most of them focus on the generation of adversarial examples, but we focus on how to use these examples to train the model more efficiently by combining the idea of contrastive learning and auto-encoder.

---

[1] https://github.com/FFYYang/CARL_RAR

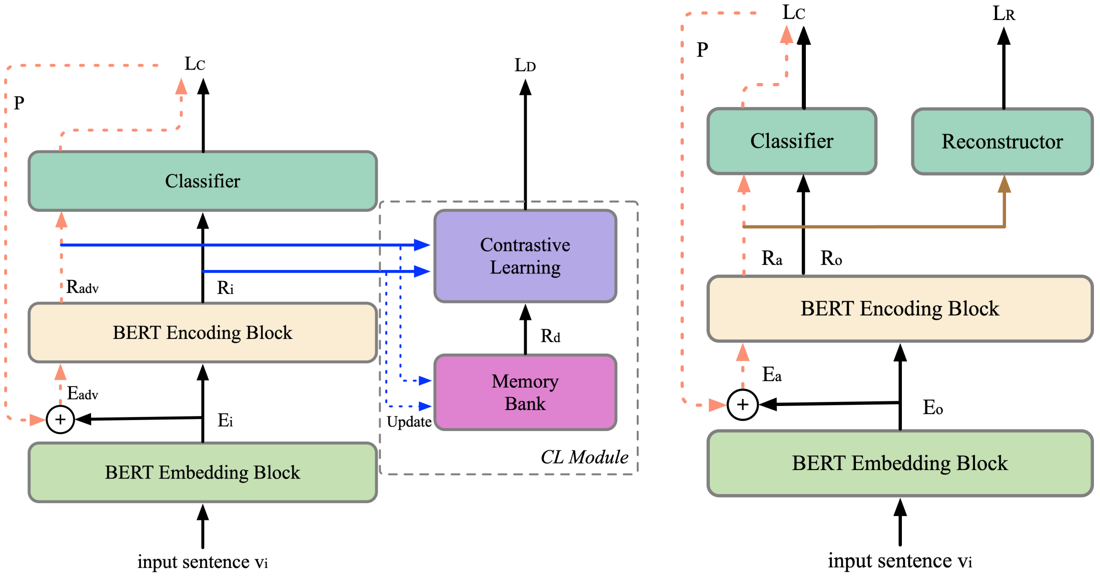

(a) CARL architecture (b) RAR architecture

Figure 1: The overview of CARL and RAR. For each input training sentence, FreeLB is first used to generate its adversarial examples E_a, shown as the yellow dotted line. CL Module(**Left**) is used to calculate the contrastive loss which aims to narrow R_a and R_o, while push them apart from R_d. Reconstructor(**Right**) is used to reconstruct the original sentence from R_a.

3 Approach

We aim to learn a robust text classification model by helping the model to learn adversarial samples more efficiently in the training process.

3.1 Overview

The overview of our approaches is depicted in Figure 1. Given an input training sentence, we first use FreeLB (Zhu et al., 2020) to get its adversarial embeddings E_a which are likely to fool the current model. In addition to minimizing these adversarial examples' classification errors, we propose two novel approaches to train them: **1) CARL (Contrastive Adversarial Representation Learning).** Its goal is to narrow the distance of sentence-level semantic representation between the original sample and its adversarial sample while pushing them away from samples that belong to different classes. We achieve this by using the CL (Contrastive Learning) module shown in Figure 1(a). **2) RAR (Reconstruction from Adversarial Representations).** It is designed to reconstruct every word in the original input sentence from their adversarial representations by the reconstructor shown in Figure 1(b).

In subsequent sections, we describe how to use CARL and RAR to train adversarial samples more effectively. In section 3.2, we describe how to use contrastive learning approach to learn a robust semantic representation for the input sentence. In section 3.3, a reconstruction module is designed to prompt the model to learn more robust lexical knowledge from input sentence's adversarial embeddings.

3.2 Contrastive Adversarial Representation Learning

Intuition. Recent gradient-based adversarial training approaches only use the classification loss to optimize the model on adversarial examples. Although they get promising results, the potential value of adversarial examples is not fully exploited. When only the classification loss is used, the model tends to learn a robust classifier, the robustness of the feature encoder is not greatly improved. After all, the classification loss function does not explicitly force the model try to learn a representation which is robust to adversarial perturbations.

Representation knowledge is highly structured, because dimensions contain complex interdependencies (Tian et al., 2020). If the model learns the adversarial samples in this perspective, there will be a huge learning space. In addition, it is suitable for adversarial training, for the representation reflects the model's understanding and the extracted knowledge of the input sentence, which

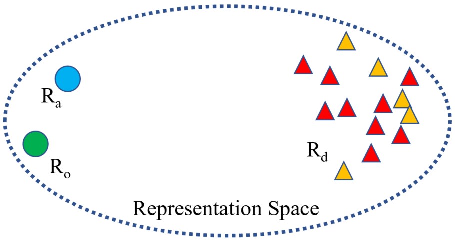

Figure 2: The intuition of CARL. The blue and green circle are adversarial and original representation of the input example, the triangles are representations of examples which belong to different classes. CARL aims to get two circles close and keep circles away from triangles

should be consistent, no matter the input is the original sentence or the adversarial sentence. We expect the model to directly learn an encoder which can output a robust semantic representation for the input sentence, and even if the input sentence is contaminated by adversarial perturbations, the representation will not be significantly affected.

The intuition of CARL is shown in Figure 2. The big ellipse refers to the representation space which is corresponding to the output of *ALBERT Encoding Block*. R_o, shown as the small green circle, is the representation of the original training example. R_a, shown as the small blue circle, is the representation of adversarial example. R_d, shown as small triangles, is a group of the representations of examples whose golden labels are different from the input sentence. CARL's goal is to make two small circles closer and make circles far away from triangles, so as to prevent adversarial attacks from leading the model to incorrectly understand the input sentence.

Implementation. We are inspired by the contrastive representation distillation approach proposed by Tian et al. (2020) and we adapt it to the text domain's adversarial training. Concretely, we design CARL's objective to maximize the lower bound to the mutual information between the adversarial and original representation of the input sentence.

Specifically, given a dataset V that consists of a collection of samples $\{v_i\}_{i=1}^N$. For each sample v_i, there are many other samples that share the same label with it, we call these samples *positives*, accordingly, we call samples whose labels are different from v_i as *negatives*. In addition, the adversarial sample of v_i can also be called *positive*.

During model's training process, for each input sample v_i whose embedding is E_i, we sample K *negatives* $\{v_{i,j}^n\}_{j=1}^K$ for it. FreeLB algorithm is first used to obtain a perturbation δ which can approximately maximum classification loss inside the ϵ-ball around E_i, as

$$\max_{||\delta||<\epsilon} L_C(f_\theta(E_i+\delta), y_i), \tag{1}$$

where y_i is the golden label of v_i, L_C is the classification loss function, θ is model's parameter, f is the model's forward function. Adding δ to E_i can obtain the adversarial embedding E_i^{adv}. Model's encoding block will then map E_i and E_i^{adv} to the representation space to get R_i and R_i^{adv}. Similarly, we can also get the original and adversarial representations for the negatives $\{v_{i,j}^n\}_{j=1}^K$, we mark them as $\{R_{i,j}^n\}_{j=1}^K$ and $\{R_{i,j}^{n,adv}\}_{j=1}^K$. We expect the distance between R_i and R_i^{adv} to be as close as possible while pushing the representations of *negatives* away from them. To achieve this, we adapt the contrastive objective proposed by Tian et al. (2020) into our optimization problem, as

$$\begin{aligned} L_D^a &= -\underset{S_{adv}}{E}\left[\log \frac{h_\theta(\{R_i^{adv}, R_i\})}{\sum_{j=1}^K h_\theta(\{R_i^{adv}, R_{i,j}^n\})}\right], \\ L_D^o &= -\underset{S_{orig}}{E}\left[\log \frac{h_\theta(\{R_i, R_i^{adv}\})}{\sum_{j=1}^K h_\theta(\{R_i, R_{i,j}^{n,adv}\})}\right], \end{aligned} \tag{2}$$

$$L_D = L_D^o + L_D^a, \tag{3}$$

where L_D^a is the contrastive loss function anchored on the adversarial representation R_i^{adv} of v_i, it aims to force the input sentence's original representation and adversarial representation close, and push the adversarial representation of the input sentence apart from its *negatives*' original representations, and it is optimized on set $S_{adv} = \{R_i^{adv}, R_i, R_{i,1}^n, ..., R_{i,K}^n\}$. Similarly, L_D^o is anchored on the original representation R_i of v_i and $s_{orig} = \{R_i, R_i^{nadv}, R_{i,1}^{n,adv}, ..., R_{i,K}^{n,adv}\}$. h_θ is a discriminating function which outputs a big value for *positive* pairs and small for *negative* pairs, we use vector dot product's result as the score and adjust its dynamic range by a hyperparameter τ, as

$$h_\theta(x_1, x_2) = \exp(x_1 \cdot x_2 \cdot \frac{1}{\tau}). \tag{4}$$

In practice, K can be extremely large. To make the computation of Eq.2 tractable, we randomly select $m (m < K)$ negatives from the dataset. Besides, Noise-Contrastive Estimation (Gutmann and Hyvärinen, 2010; Wu et al., 2018) is used to approximate the softmax distribution as well as reduce the computational cost.

During the model's training process, for every training sample, we need m negatives' original and adversarial representations. For m is usually large in practice, so it is impossible to calculate all of these representations at the same time during each mini-batch's iteration. Following Wu et al. (2018), we maintain two memory banks, B_{orig}, B_{adv}, to store the original and adversarial representations for every training sample. Therefore, when we calculate the contrastive loss, we don't have to recompute *negatives*' representations and we can just retrieve them from the memory bank. Besides, the memory bank should be dynamically updated with newly computed representations at each mini-batch iteration, as

$$\begin{aligned} B_{orig}[i] &= M \cdot B_{orig}[i] + (1 - M) \cdot R_i, \\ B_{adv}[i] &= M \cdot B_{adv}[i] + (1 - M) \cdot R_i^{adv}, \end{aligned} \tag{5}$$

where M is a hyperparameter, i is the index of a training sample. To be noticed, CARL cannot be used at the beginning of training, because the model is unstable and both original and adversarial representations are noisy. Optimizing contrastive loss at this time can cause the model difficult to converge. The proper way is to wait until the model is going to be stable, and use an entire epoch to forward every training sample through the model to initialize the whole memory bank, after which the contrastive loss can be used to optimize the model. In conclusion, we will optimize the following problem, as

$$\min_{\theta} (L_C + L_D)_{(v,y) \sim D} \left[\max_{||\delta|| < \epsilon} L_C(f_\theta(E + \delta), y) \right], \tag{6}$$

where v is one training sample, y is its golden label, D is the data distribution, L_C is the classification loss.

3.3 Reconstruction from Adversarial Representations

Intuition. The gradient-based adversarial attacking approach adds perturbations on every word's embedding, we have no idea the contaminated embedding indicates which word in the real world. If the model cannot recognize the contaminated word embedding or identify it to a wrong word, its understanding of the whole sentence's semantics could be wrong, especially when the keyword of the sentence is misunderstood by the model. The special cases are easy to occur because we find that the norm of adversarial perturbation added to the keyword of a sentence is usually larger than that of others words, and it makes the keyword harder to be recognized.

To solve the problem, inspire by the Masked Language Model proposed in BERT (Devlin et al., 2019). we design RAR to reconstruct every token from its adversarial representation. To reconstruct tokens correctly, the model should not only learn more robust lexical knowledge for every word but also accurately understand the semantics of the whole sentence.

Implementation. Inspired by the pre-training task used in BERT(Devlin et al., 2019), we map the adversarial representation of each word to a vector which length is the vocabulary size.

Specifically, the reconstructor receives input sentence's token-level adversarial representation $R_i^{adv,tok} \in [sequence\_length, hidden\_size]$ from the *ALBERT Encoding block* as input, then $R_i^{adv,tok}$ will be forwarded through a *Layer Normalization*, *GeLU Activation Function* and two *Feed Forward Layers*. The first feed-forward layer maps the $hidden\_size$ to $embedding\_size$ and the second feed-forward layer's parameters are shared with *ALBERT Embedding Layer* to project the $embedding\_size$ into $vocabulary\_size$. Then, we can get the predicted probability distribution over the vocabulary for every token's position in the sentence. Finally, we use the cross-entropy function to calculate the reconstruction loss L_R.

In the training process, FreeLB and RAR are combined to optimize the model. After we use FreeLB to get the adversarial representations of every word and the whole sentence, we simultaneously feed them to the reconstructor and the classifier accordingly. That is, the model is asked not only to predict the correct class of the adversarial sample but also to reconstruct the sample's original words from their adversarial representations. In conclusion, we will optimize the following problem, as

$$\min_{\theta} (L_C + w_r \cdot L_R)_{(v,y) \sim D} \left[\max_{||\delta|| < \epsilon} L_C(f_\theta(E + \delta), y) \right], \tag{7}$$

where $w_r = 0.1$ is the weight for the reconstruction loss.

4 Experiment

We evaluate our approaches on four datasets. We first introduce the datasets, the baselines, and the experiment settings. Then, we show experiment results and provide further analysis.

4.1 Datasets

We use four text classification datasets: SST-2, Yelp-P, AG's News, and Yahoo! Answers.

SST-2. The Stanford Sentiment Treebank (Socher et al., 2013) consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentence-level sentiment (positive/negative) of a given input text.

Yahoo! Answers. This dataset is composed of ten topic categories: Society & Culture, Science & Mathematics, Health, Education & Reference, etc. In this work, we use five categories. For every category, we use 12,000 training samples, 400 validation, and 400 test samples.

Yelp-P. The original Yelp dataset is built using reviews from the website Yelp[2]. Each review has a rating label varying from 1 to 5. We use it as the binary classification, and randomly choose 30,000 training samples, 1000 validation, and 1000 testing samples for every class.

AG's News. This is a dataset of more than one million news articles and they are categorized into four classes: World, Sports, Business, and Sci/Tech. Each class contains 30,000 training samples and 1,900 testing samples. In our work, for each class, we use 15,000 training samples, 500 validation and testing samples.

4.2 Baselines

We compare our proposed approach with the following approaches.

ALBERT for Text Classification. For ALBERT, the first token of the sequence is $[CLS]$, when doing the text classification task, ALBERT takes the final hidden state h of the $[CLS]$ token as the representation of the whole sentence. The classifier consists of a feed-forward layer and a softmax function.

$$p(c|h) = softmax(Wh), \tag{8}$$

where W is a learnable parameter matrix, c is the class. ALBERT is fine-tuned with all parameters as well as W jointly by maximizing the log-probability of the golden label.

	SST-2	Yahoo!	Yelp-P	AG's News
γ	0.6	0	0.5	0
α	0.1	0.01	0.05	0.01
ϵ	0	0	0	0
n	2	3	3	3

Table 1: Hyperparameters for FreeLB on 4 datasets: step size α, maximum perturbation norm ϵ (if it is set to zero, the perturbation's norm is not limited), number of iteration steps n, magnitude of initial random perturbation γ.

FreeLB. FreeLB, proposed by Zhu et al. (2020), adds adversarial perturbations to ALBERT embedding layer's output, and minimizes the resultant adversarial loss around input samples, it leverages the "free" training strategy (Shafahi et al., 2019) to improve the efficiency of adversarial training, which made it possible to apply PGD-based adversarial training(Madry et al., 2018) into large-scale pre-trained language model. In this work, we apply FreeLB to ALBERT model.

4.3 Experiment settings

We implement our two approaches on albert-base-v2 (from huggingface's pytorch implementation[3]), the parameters of *ALBERT Embedding Block* and *ALBERT Encoding Block* are loaded from the pre-trained model, we do experiments on the fine-tuning stage. We use the Adam optimizer to train the modules and the learning rate is set to 1e-5, and batch size is 16 for AG's News and 32 for the other three datasets. Since FreeLB's hyperparameters highly depend on the characteristic of the dataset, we apply hyperparameter search to every dataset and the searching results are shown in Table 1. These hyperparameters stay unchanged in CARL and RAR. We train our models on two Tesla P40s.

CARL and RAR are both implemented based on the FreeLB. In RAR, L_R is used to update the model's parameters from the beginning of the training. While in CARL, L_D is used after the model is about to be stable (specific settings can be found in Table 3). Besides, m is set to 20000 for YelpP and 16000 for the other three datasets. τ and M is

---

[2] https://www.yelp.com/dataset/challenge

[3] https://github.com/huggingface/transformers

	SST-2	Yahoo! Answers	Yelp-P	AG's News
ALBERT	92.16	73.93	93.55	89.90
FreeLB	93.23	74.28	93.93	90.85
CARL(ours)	**93.77(+0.54)**	74.65(+0.37)	**94.55(+0.62)**	**92.05(+1.20)**
RAR(ours)	93.73(+0.50)	**74.88(+0.60)**	94.4(+0.47)	91.75(+0.90)

Table 2: Comparisons between CARL, RAR, and baselines on four datasets. ALBERT is the model trained without any adversarial training approach. FreeLB uses classification loss to learn adversarial examples. CARL and RAR are implemented based on FreeLB, they use additional optimization objectives for adversarial examples. We compare them with FreeLB and find CARL performs best in most cases.

	SST-2	Yahoo!	Yelp-P	AG's News
τ	6315	7200	5625	7750

Table 3: Steps after which L_D will start to be used in CARL before which only L_C is used to optimize the model's parameters.

set to 0.07 and 0.5 respectively. For SST-2, we use a development set to do the evaluation. To make the results reliable, we run each experiment three times with the same hyperparameters but different random seeds and report their average scores. For the other three datasets, we use a development set to choose the best training checkpoint and evaluate it on the test set.

4.4 Results and Discussion

The results of the proposed approach and baselines are shown in Table 2. FreeLB, CARL, and RAR let the adversarial samples participate in the model's training process, so it's not surprising that all of them perform better than ALBERT. These improvements can be mainly attributed to the effect of data augmentation.

The experiment results also show that the performance of CARL and RAR on four data sets is higher than FreeLB. These results demonstrate that the approaches we proposed to defend against gradient-based adversarial attacks during the training process are effective and well applied to various text classification datasets. We conjecture that this is because the contrastive objective can encourage the model to discover the true underlining knowledge which can determine the classification label from adversarial and original representation. This underline knowledge is robust against adversarial perturbation added on the original sample and won't be changed by modifying the statement of the sentence. When the model can learn this knowledge, its generalization and robustness will be improved.

Cosine	α=0.1	α=0.075
ALBERT	0.851	0.871
FreeLB	0.899	0.918
CARL	0.917(+0.018)	0.934(+0.016)
RAR	**0.926(+0.027)**	**0.941(+0.023)**
Euclidean	α=0.1	α=0.075
ALBERT	8.409	7.746
FreeLB	6.477	5.776
CARL	5.340(-1.137)	4.668(-1.108)
RAR	**5.121(-1.356)**	**4.453(-1.323)**

Table 4: The difference between original and adversarial representations of samples in AG's News test set. FreeLB, RAR, and CARL perform much better than ALBERT. We compare CARL and RAR with FreeLB, and we find RAR is the best.

When comparing CARL and RAR, CARL performs better than RAR in most cases. It is because CARL's training objective is to narrow the distance between the adversarial sample and the original sample in the representation space, while the classifier of the model is also based on the representation of the sentence, so the objective of CARL has a more straight forward contribution to the classification task than that of the RAR.

4.5 Analysis

The difference between adversarial and original sample's representations. Table 4 compares the Euclidean distance and cosine similarity between adversarial and original samples' sentence-level representations in four approaches. We use AG's News test set to do this experiment. We use the models trained by the above four approaches, and for every sample v_i, we first calculate its original representation R_i, and obtain their adversarial representation R_i^{adv} using the k-PGD approach with the same hyperparameters setting, then measure their distance by the cosine similarity and the Euclidean distance. We also compare results when

	ϵ=0.02	ϵ=0.075	ϵ=0.1
ALBERT	86.6	72.60	70.2
FreeLB	88.8	80.25	78.0
CARL	**90.0**	**81.4**	**80.1**
RAR	89.8	80.5	78.6

Table 5: Performance robustness experiment results. ϵ is the maximum perturbation norm. CARL performs best under adversarial attacks of different strength.

using different max perturbation norms α in k-PGD. The final result is the average of all samples.

Experiment results show that FreeLB, CARL, and RAR perform much better than ALBERT either on the cosine similarity or Euclidean distance, this indicates that the robustness of the model in the representation space can be effectively improved by optimizing the classification error of adversarial samples. In addition, when compared with FreeLB, CARL, and RAR, the performance of RAR is the best, followed by CARL. This shows that our approaches are effective to further improve model representation space's robustness and RAR is more effective. The reason why RAR is better than CARL can be explained that the objective of RAR is more difficult than that of CARL. The optimization objective of RAR is at the token level, while CARL is at the sentence level, so RAR can encourage the model to learn additional lexical knowledge which is also beneficial for improving the semantic representation of the whole sentence.

The robustness of performance. We use the k-PGD method to attack models trained on AG's News by four approaches. Experimental results showed that the performance of the FreeLB, CARL, and RAR is significantly better than ALBERT. That is because they allow the adversarial samples to participate in the model's training process. In the case of FreeLB, RAR, and CARL, CARL is the best, followed by RAR. The reason can be explained from the perspective of multi-task learning. If we regard CARL and RAR as two multi-task learning frameworks, it is obvious that compared to the reconstruction task used in RAR, the contrastive learning task used in CARL is more similar to the classification task, because both of these two tasks' objectives operate on sentence-level representations. In addition, RAR performs better on representation robustness while CARL performs better on performance robustness. This indicates that although narrowing the representation distance between original and adversarial samples can improve the model's performance and robustness. It's not the case that the shorter distance, the more robust performance.

Outer-space buffs might love this film, but others will find its pleasures intermittent.	N
Outer-space buffs would love this film, but others will find its pleasures occasional.	P
The film will play equally well on both the standard and giant screens.	P
The film would play more well on all the standard and giant screens.	N
Why make a documentary about these marginal historical figures	N
Why make a documentary about the marginal historical figures	P

Table 6: Reconstructed adversarial samples. The first line is the original sentence, the second line is the reconstructed sentence. N and P refers to negative and positive label the model predicted. The model can correctly classify the original sentences, but not these reconstructed sentences.

Reconstructed adversarial samples. We let SST-2's dev set forward the trained RAR model and use the k-PGD method to attack it. Then we take the output logits of the RAR module to obtain the reconstructed sentence. We find that we could get some text-form adversarial samples in this way. The semantics of these reconstructed samples are almost identical with that of original samples, but they can fool the model trained by ALBERT successfully. Table 6 shows some examples of the reconstructed sentences which can be used as text-form adversarial samples and can be further used as augmented data.

5 Conclusion

In this work, we propose two gradient-based adversarial training approaches, CARL and RAR, to improve the performance and robustness of text classification models. The key idea of CARL is narrowing the original sample and adversarial sample in the representation space. While RAR forces the model to reconstruct the original tokens from their adversarial representations. Experiments demonstrate our approaches outperform the baseline. The sentence representation and the model's performance are more robust, which proves the effectiveness of the proposed approaches. Besides, RAR can be used to generate adversarial examples.

Acknowledgments

We would like to thank all the reviewers for their insightful and valuable comments and suggestions.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*.

Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov, and Farinaz Koushanfar. 2019. Adversarial reprogramming of text classification neural networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5215–5224. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*, pages 79–86.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019a. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019b. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

DE Rumellhart. 1986. Learning internal representations by error propagation. *Parallel distributed processing*, 1:318–362.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3353–3364.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive multiview coding. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XI*, volume 12356 of *Lecture Notes in Computer Science*, pages 776–794. Springer.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop Black-*

boxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Jingjing Xu, Liang Zhao, Hanqi Yan, Qi Zeng, Yun Liang, and SUN Xu. 2019. Lexicalat: Lexical-based adversarial reinforcement training for robust sentiment classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5521–5530.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. 2019. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 227–238.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

A Appendices

We provide some details of experiment settings.

A.1 Additional Experimental Details

There is no significant difference in the training time between our proposed two approaches. For SST-2 and AG's News, it takes about two hours to train the model. For Yelp-P and Yahoo, it takes about ten hours.

The number of parameters in each model is shown in Table 7. The number of parameters for ALBERT, FreeLB, and CARL is the same, while RAR has more parameters because there is an additional reconstructor module.

	#Parameters
ALBERT	11685122
FreeLB	11685122
CARL	11685122
RAR	11813810

Table 7: Number of parameters in each model.

A.2 Hyperparameter Search Details

Because the hyperparameters of FreeLB differ greatly in different datasets, we should search for the best hyperparameter configuration for each dataset. We first set the searching bounds of each hyperparameter as shown in Table 8. Then we combine grid search and manual tuning approaches. Specifically, grid search is first used to search at a relatively large granularity, and then manual tuning is used to search at a small granularity. The criterion used for hyperparameter searching is the accuracy of the validation set. The searching result is also used in CARL and RAR.

Hyperparameter	**Bounds**
γ	[0, 0.8]
α	[0.01, 0.2]
ϵ	[0, 0.5]
n	[2, 4]

Table 8: Bounds for each hyperparameter: Step size α, maximum perturbation norm ϵ (if it is set to zero, the perturbation's norm is not limited), number of iteration steps n, the magnitude of initial random perturbation γ.

A.3 Datasets Details

The statistics information of four datasets is shown in Table 9. Except SST-2, we only use a portion of data which is randomly selected from the original dataset because of the limitation of computing resource. Since our goal is not to reach the SOTA but to gain relative improvement of performance and robustness compared to FreeLB, dropping some training data won't affect it.

The data pre-processing approach is the same as huggingface's implementation[4]. In addition, we randomly sample m *negatives* for each training example in CARL.

Dataset	**#Train**	**#Dev**	**#Test**
SST-2	67,349	872	-
Yahoo! Answers	60,000	60,000	2,000
Yelp-P	60,000	60,000	2,000
AG's News	60,000	60,000	2,000

Table 9: The statistics information of the four datasets we use.

---

[4] https://github.com/huggingface/transformers

Multi-Granularity Contrasting for Cross-Lingual Pre-Training

Shicheng Li[1], Pengcheng Yang[2], Fuli Luo[2], Jun Xie[2]
[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]Alibaba DAMO Academy, Hangzhou, China
`lisc99@pku.edu.cn`
`{mingyang.ypc, lfl259702, qingjing.xj}@alibaba-inc.com`

Abstract

Cross-lingual pre-training aims at providing effective prior representations for the inputs from multiple languages. With the modeling of bidirectional contexts, recently prevalent language modeling approaches such as XLM achieve better performance than traditional methods based on embedding alignment, which strives to assign similar vector representations to semantic-equivalent units. However, such approaches like XLM capture cross-lingual information based solely on shared BPE vocabulary, resulting in the absence of fine-grained supervision induced by embedding alignment. Inheriting the advantages of the above two paradigms, this work presents a multi-granularity contrasting framework, namely MGC, to learn language-universal representations. While predicting the masked words based on bidirectional contexts, the proposal also encodes semantic equivalents from different languages into similar representations to introduce more fine-grained and explicit cross-lingual information. Two effective contrasting strategies are further proposed, which can be built upon semantic units of multiple granularities covering words, span, and sentences. Extensive experiments demonstrate that our approach can achieve significant performance gains in various downstream tasks, including machine translation and cross-lingual language understanding.

1 Introduction

Cross-lingual pre-training (Lample and Conneau, 2019) has achieved striking success in the field of natural language processing. By providing effective prior representations for the inputs from different languages, it has boosted performance on various downstream tasks such as machine translation and cross-lingual language understanding.

Early efforts regarding cross-lingual pre-training mainly focus on embedding alignment (Mikolov et al., 2013b; Lample et al., 2018), which is targeted at the assignment of similar vector representations to semantic-equivalent units (e.g., the parallel bilingual word or sentence pairs). For instance, Mikolov et al. (2013b) attempt to project pre-trained monolingual word embeddings from two languages into a common semantic space with a simple linear transformation, so that parallel bilingual words share the same representation. This allows the introduction of explicit fine-grained supervision to guarantee the representational similarity of semantic equivalents, but neglects the modeling of bidirectional contexts. Going a step further, recently prevalent approaches of language modeling such as XLM (Lample and Conneau, 2019) remedy this by predicting the masked tokens based on bidirectional contexts (Devlin et al., 2019), and also benefit from larger model capacity (Vaswani et al., 2017). However, the cross-lingual information captured by these language modeling approaches comes solely from the shared BPE vocabulary (Sennrich et al., 2016), resulting in the absence of more fine-grained explicit supervision induced by embedding alignment.

In light of the pros and cons of the above two paradigms, we propose a multi-granularity contrasting (MGC) framework for cross-lingual pre-training. In addition to modeling context bidirectionality with the widely used masked language modeling (MLM) (Devlin et al., 2019), our approach draws upon contrastive learning (Gutmann and Hyvärinen, 2010) to introduce more fine-grained cross-lingual alignment information. The core idea is to enhance the consistency between representations of semantic equivalents (e.g., the aligned word pairs such as "*cat*" in English and "*chat*" in French). To this end, we propose two effective contrasting strategies: *hard* contrasting which constructs pseudo-parallel bilingual word pairs via external word aligner (Dyer et al., 2013),

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1708–1717
August 1–6, 2021.

and *soft* contrasting which employs multi-head attention (Vaswani et al., 2017) to provide posterior approximation for the representations of the desired semantic equivalents. Considering the inherent multi-granularity of natural language expressions, we build the proposed contrasting framework upon semantic units of various granularities (including *word*, *span*, and *sentence*) to further enrich cross-lingual information and enhance the model's capability of encoding multi-granularity representations.

We conduct experiments on a variety of downstream scenarios, including multiple machine translation and cross-lingual language understanding tasks. Comprehensive experimental results demonstrate that our proposed approach can achieve significant performance gains over baselines. To be more specific, our MGC raises the average accuracy of our implemented XLM-R (Conneau et al., 2019) from 74.4 to 76.0 on XNLI under the setting of cross-lingual transfer and also surpasses various baselines on representative translation tasks such as WMT14 EN-DE and EN-FR.

2 Methodology

In order to introduce more fine-grained and explicit cross-lingual supervision, we propose a multi-granularity contrasting (MGC) framework to learn language-universal representations. We first elaborate on the proposed approach based on *word*-level contrasting, and then extend it to *span*-level and *sentence*-level to further enrich cross-lingual information.

2.1 Overview

We denote a pair of parallel bilingual instance as $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, \cdots, x_m)$ and $\mathbf{y} = (y_1, \cdots, y_n)$ refer to the source and target sentence, respectively. Then, the transformer (Vaswani et al., 2017) encodes $\mathbf{x}$ to obtain its hidden representations $\mathbf{h_x} = (h_{x_1}, \cdots, h_{x_m})$. The hidden representations $\mathbf{h_y} = (h_{y_1}, \cdots, h_{y_n})$ of $\mathbf{y}$ can be obtained in the same way. In order to introduce more fine-grained and explicit cross-lingual supervision similar to embedding alignment, we expect the semantic-equivalent units (e.g., "*cat*" in English and "*chat*" in French) from different languages to exhibit similar vector representations. Meanwhile, the representations of units with different semantics (e.g., "*cat*" in English and "*car*" in English or "*voiture*" in French) should be distinguished from each other to capture their discriminative specific information.

Motivated by this, we employ contrastive learning (Gutmann and Hyvärinen, 2010) to model such training objectives. Without loss of generality, we elaborate on our proposed approach with the units in the source language as anchors. Formally, we use u to represent the representation of one unit (e.g., "*cat*" in English) in $\mathbf{x}$. The representation of its corresponding semantic equivalent (e.g., "*chat*" in French) in $\mathbf{y}$ is denoted as v^+. The set of negative representations exhibiting different semantic with u is denoted as $\boldsymbol{v}^- = \{v_1^-, \cdots, v_k^-\}$, where k is the number of negative representations. Then, the contrastive loss for the representation tuple $(u, v^+, \boldsymbol{v}^-)$ can be defined as:

$$\mathcal{L}(u, v^+, \boldsymbol{v}^-) = -\log\left(\frac{\exp(u \cdot v^+/\tau)}{Z(u)}\right) \quad (1)$$

where $Z(u) = \exp(u \cdot v^+/\tau) + \sum_{v^-} \exp(u \cdot v^-/\tau)$ is the normalization factor and τ is the temperature controlling the concentration level of the sample distribution. The above equation corresponds to the negative log-likelihood loss of a softmax-based classifier measuring semantic similarity by the dot product. The classifier treats each unit as a distinct class, and aims at classifying u to the class of its semantic equivalent v^+ and vice versa. By maximizing the consistency between the representations of semantic equivalents with such a training objective, the pre-trained models are encouraged to introduce more fine-grained explicit alignment supervision, thereby enhancing their capability of learning language-invariant representations. Meanwhile, the representations of units exhibiting different semantics are penalized to be kept distinguished from each other, so that the model is equipped with the ability to capture specific features of the source inputs.

2.2 Word-Level Contrasting

The word-level contrasting strives to integrate the word-alignment information contained in parallel bilingual instance $(\mathbf{x}, \mathbf{y})$. However, an intractable challenge is that ideal semantic-equivalent word pairs tend to be unavailable in practice. To remedy this, here we propose two effective solutions: *hard* contrasting and *soft* contrasting, detailed as follows.

Hard contrasting The hard contrasting aims at constructing pseudo-parallel bilingual word pairs via an external word aligner. Specifically, for each word x in the source sentence $\mathbf{x}$, its aligned word in $\mathbf{y}$ is defined as:

$$y^* \approx \hat{y} = \operatorname*{argmax}_{y \in \mathbf{y}} \text{aligner}(y|x) \tag{2}$$

where $\text{aligner}(\cdot|\cdot)$ denotes the alignment probability that can be computed by the word aligners such as fast_align (Dyer et al., 2013). Considering that there exists no semantic equivalent for some words (e.g., "*the*" in English), we construct the semantic-equivalent word sets $\mathcal{N}_{\text{word}}(\mathbf{x}, \mathbf{y})$ as mutually aligned word pairs in $(\mathbf{x}, \mathbf{y})$. For each aligned word pair $(x, y) \in \mathcal{N}_{\text{word}}(\mathbf{x}, \mathbf{y})$, the representations $(u, v^+, \boldsymbol{v}^-)$ in Eq. (1) can be computed as:

$$\begin{cases} u = \ell_2(h_x) \\ v^+ = \ell_2(h_y) \\ \boldsymbol{v}^- = \Big\{\ell_2(h_z)\big|z \in \mathbf{y}\backslash y\Big\} \end{cases} \tag{3}$$

where ℓ_2 represents ℓ_2-normalization and $\mathbf{y}\backslash y$ denotes words in $\mathbf{y}$ other than the word y. Finally, the word-level hard contrasting loss for the source sentence $\mathbf{x}$ is formalized as:

$$\mathcal{L}_{\text{word}}(\mathbf{x}) = \sum_{(x,y) \in \mathcal{N}_{\text{word}}(\mathbf{x},\mathbf{y})} \mathcal{L}(u, v^+, \boldsymbol{v}^-) \tag{4}$$

The loss $\mathcal{L}_{\text{word}}(\mathbf{y})$ for the target sentence $\mathbf{y}$ can be computed in a similar way by swapping $(\mathbf{x}, \mathbf{y})$ to $(\mathbf{y}, \mathbf{x})$. Due to space limitations, here we omit the related details.

Soft contrasting Due to the strict requirements on the quality of constructed pseudo-parallel bilingual word pairs, hard contrasting is prone to suffer from potential error propagation induced by external word aligners. In addition, some source words may correspond to multiple target words, which conflicts with the strict one-to-one alignment of hard contrasting. To tackle the above issues, we propose soft contrasting, aiming at learning word alignment implicitly and jointly to approximate semantic equivalents via the attention mechanism (Vaswani et al., 2017). Specifically, for each word x in the source sentence $\mathbf{x}$, the aggregated representation $\text{MHA}(h_x, \boldsymbol{h}_\mathbf{y})$ can be obtained by performing multi-head attention[1] with h_x serving as the query and $\boldsymbol{h}_\mathbf{y}$ serving as the keys/values. Since multi-head attention naturally assign larger weights to the words in $\mathbf{y}$ that are aligned to x, $\text{MHA}(h_x, \boldsymbol{h}_\mathbf{y})$ can be regarded as an approximation of the representation of semantic equivalent of x. Therefore, the representations $(u, v^+, \boldsymbol{v}^-)$ in Eq. (1) can be defined as:

$$\begin{cases} u = \ell_2(h_x) \\ v^+ = \ell_2\big(\text{MHA}(h_x, \boldsymbol{h}_\mathbf{y})\big) \\ \boldsymbol{v}^- = \Big\{\ell_2\big(\text{MHA}(h_z, \boldsymbol{h}_\mathbf{y})\big)\big|z \in \mathbf{x}\backslash x\Big\} \end{cases} \tag{5}$$

where $\mathbf{x}\backslash x$ refers to the remaining words in $\mathbf{x}$ except x. Soft contrasting not only alleviates the dependence on external word aligners, but also frees the model from the limitations of one-to-one alignment. Additionally, by maximizing the semantic consistency between h_x and $\text{MHA}(h_x, \boldsymbol{h}_\mathbf{y})$, the model is encouraged to learn word alignment in an implicit manner, introducing richer cross-lingual information.

2.3 Span-Level Contrasting

Previous work (Joshi et al., 2019) has demonstrated the superiority of span-level representations over word-level (Devlin et al., 2019) representations due to its strength in language understanding and reasoning. Therefore, we also perform contrasting based on semantic-equivalent spans. Since span gets rid of the limitation that the semantic equivalents of the two languages must share the same number of words, here we focus on the application of hard contrasting. To be specific, given the bilingual instance $(\mathbf{x}, \mathbf{y})$, we induce the phrase table via statistical machine translation tools to obtain span-level semantic equivalents $\mathcal{N}_{\text{span}}(\mathbf{x}, \mathbf{y})$. For each aligned span pair $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in \mathcal{N}_{\text{span}}(\mathbf{x}, \mathbf{y})$ where $\bar{\boldsymbol{x}} \subset \mathbf{x}$ is a span of $\mathbf{x}$ and $\bar{\boldsymbol{y}} \subset \mathbf{y}$ is a span of $\mathbf{y}$, the representations $(u, v^+, \boldsymbol{v}^-)$ in Eq. (1) can be formulated as:

$$\begin{cases} u = \ell_2\big(\text{MP}(\boldsymbol{h}_{\bar{\boldsymbol{x}}})\big) \\ v^+ = \ell_2\big(\text{MP}(\boldsymbol{h}_{\bar{\boldsymbol{y}}})\big) \\ \boldsymbol{v}^- = \Big\{\ell_2\big(\text{MP}(\boldsymbol{h}_{\bar{\boldsymbol{z}}})\big)\big|\bar{\boldsymbol{z}} \in \mathbf{y}\backslash\bar{\boldsymbol{y}}\Big\} \end{cases} \tag{6}$$

where $\text{MP}(\cdot)$ represents the mean-pooling layer[2] employed to aggregate all hidden representations of multiple words in a span. $\boldsymbol{h}_{\bar{\boldsymbol{x}}} = (h_{\bar{x}_1}, \cdots, h_{\bar{x}_l})$ are hidden representations of span $\bar{\boldsymbol{x}} = (\bar{x}_1, \cdots, \bar{x}_l)$ and similarly for $\boldsymbol{h}_{\bar{\boldsymbol{y}}}$. The span-level contrastive loss of a given bilingual sentence pair $(\mathbf{x}, \mathbf{y})$ is

---

[1] The semantic similarity between different words from two languages can also be calculated by other approaches such as bilinear attention.

[2] Other similar aggregation layers such as max-pooling or attentive-pooling can also be implemented.

defined as the sum of the losses corresponding to all spans $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ in $\mathcal{N}_{\text{span}}(\mathbf{x}, \mathbf{y})$, whose calculation is similar to Eq. (4).

2.4 Sentence-Level Contrasting

In order to improve the quality of learned sentence embeddings, we also perform sentence-level contrastive learning to obtain the global supervision signals aggregating all token representations of the entire source input. Our proposed approach strives to pull the sentence representation of $\mathbf{x}$ towards that of its corresponding translation $\mathbf{y}$, and push it away from sentence representations of other instances. However, the direct application of artificially constructed parallel bilingual sentence pairs tends to result in a significant boundary between positive and negative samples, which may lead to vanishing contrasting signals. To remedy this problem, we make use of back-translation to infuse noise in original positive samples to obtain more competitive cross-lingual information. To be more specific, we define the sentence-level semantic equivalents $\mathcal{N}_{\text{sent}}(\mathbf{x}, \mathbf{y})$ as:

$$\mathcal{N}_{\text{sent}}(\mathbf{x}, \mathbf{y}) = \Big\{(\boldsymbol{x}, \boldsymbol{y}) \big| \boldsymbol{x} \in \{\mathbf{x}, \hat{\mathbf{x}}\}, \boldsymbol{y} \in \{\mathbf{y}, \hat{\mathbf{y}}\}\Big\} \tag{7}$$

where $\hat{\mathbf{x}}$ is the noisy version of $\mathbf{x}$ obtained by means of back-translation[3] and so is $\hat{\mathbf{y}}$. By sampling from the original $\mathbf{x}$ and the back-translated $\hat{\mathbf{x}}$, both sentences from the two languages for contrasting contain a certain amount of noise. This blurs the boundary between the positive and negative representations to some extent, thereby effectively alleviating the vanish of contrasting signals.

As with span-level contrasting, we adopt the mean-pooling layer to aggregate all token representations of a given sentence into its corresponding sentence representation. For each sentence pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{N}_{\text{sent}}(\mathbf{x}, \mathbf{y})$, we define the representations $(u, v^+, \boldsymbol{v}^-)$ in Eq. (1) for sentence-level contrasting as:

$$\begin{cases} u = \ell_2\big(\text{MP}(\boldsymbol{h_x})\big) \\ v^+ = \ell_2\big(\text{MP}(\boldsymbol{h_y})\big) \\ \boldsymbol{v}^- = \Big\{\ell_2\big(\text{MP}(\boldsymbol{h_z})\big) \big| \boldsymbol{z} \neq \mathbf{y}\Big\} \end{cases} \tag{8}$$

where $\boldsymbol{z} \neq \mathbf{y}$ means that the negative representations used for contrasting are derived from other instances in the same mini-batch.

Following XLM (Lample and Conneau, 2019) and XLM-R (Conneau et al., 2019), to learn from bidirectional contexts, we also adopt masked language modeling (MLM) as one of pre-training tasks. The MLM task aims at predicting the masked words based on the corrupted input. We concatenate a parallel bilingual sentence pair into a single sentence and randomly select 15% tokens as candidates for performing corruption. Of these selected tokens, 80% are replaced with special token `[MASK]`, 10% are kept unchanged, and the remaining are replaced by randomly selected vocabulary tokens. The final training objective is defined as the sum of the above-mentioned multiple contrastive losses as well as the cross-entropy of MLM.

3 Experiments

We conduct experiments on a variety of downstream tasks, which can be divided into two categories: machine translation and cross-lingual language understanding tasks.

3.1 Pre-Training

Datasets We pre-train our model on large-scale datasets involving the 15 languages of XNLI (Conneau et al., 2018)[4]: English, French, Spanish, German, Greek, Bulgarian, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, Swahili, and Urdu. Following Conneau et al. (2019), we reconstruct Common-Crawl Corpus to obtain monolingual training datasets while the bilingual data is obtained from the OPUS website [5]. We also conduct up/down-sample for all pre-training data with a smoothing parameter. The sentence-piece model (SPM) (Kudo and Richardson, 2018) provided by Conneau et al. (2019)[6] is employed to tokenize all training data.

Model architecture We implement the proposed approach based on the Transformer (Vaswani et al., 2017) encoder with 12 identical layers, each of which consists of a multi-head attention module and a feed-forward network. The model dimension and the number of heads are set to 768 and 12, respectively, with the inner size of the feed-forward network being 3072. We choose GeLU (Hendrycks and Gimpel, 2016) as our activation function. We

---

[3] In the implementation, we obtain multiple $\hat{\mathbf{x}}$ by pre-training the *target→source* translation model and performing beam search or top-k sampling.

[4] https://github.com/facebookresearch/XNLI

[5] http://opus.nlpl.eu/

[6] https://github.com/google/sentencepiece

use the sentence-piece vocabulary provided by Conneau et al. (2019), whose size is 250K.

Training parameters We apply Adam (Kingma and Ba, 2015) optimizer with a learning rate of 5×10^{-4} and adopt invert linear decay schedule to pre-train our models. We employ a dropout with probability to 0.1 for both the hidden states and the attention distribution. The temperature τ in Eq. (1) is set to 0.1 and the coefficients of all contrastive losses are set to 1. We take advantage of the gradient accumulation technique to simulate the batch size of 512. Our model is initialized with the pre-trained checkpoint released by Conneau et al. (2019)[7] and then pre-trained on 8 × 32GB NVIDIA V100 GPUs with mixed-precision floating-point arithmetic. Overall, it took about 3 weeks to converge.

3.2 Machine Translation

Datasets We conduct experiments on three widely-used machine translation datasets of various training data sizes: IWSLT14 De-En (160K), WMT14 En-De (4.5M), and WMT14 En-Fr (36M). The sentence-piece tokenization with the same vocabulary as pre-training are used to tokenize all translation samples. The BLEU score computed by the *multibleu.perl* script[8] is used as the evaluation metric for all translation tasks.

Model architecture We use the pre-trained model to initialize a 12-layer encoder. The decoder is implemented as a standard 6-layer Transformer (Vaswani et al., 2017) decoder, each layer consisting of a multi-head self-attention module, a multi-head cross-attention module and a feed-forward network. The entire decoder is initialized randomly, and it is jointly trained with the pre-trained encoder. Other model hyper-parameters including the hidden size, the number of heads, the inner size of the feed-forward network, and the choice of activation function are identical to the encoder.

Training parameters We also adopt mixed-precision floating-point arithmetic to train our models on the machine translation task. The experiments are conducted on 8 × 32GB NVIDIA V100 GPUs. We use an Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.98$ to optimize our model during training. The learning rate is warmed up to 1×10^{-4} linearly in the first 4000 updates and then decays at a rate proportional to the inverse square root of the update number. We average the last 10 checkpoints as the final model and perform beam search with a beam size of 5 during inference. The probability of dropout is set to 0.1 to avoid over-fitting. The length penalty is set to 1.0.

3.3 Cross-Lingual Language Understanding

Dataset In order to verify the effectiveness of our approach on cross-lingual language understanding, we conduct evaluation on XNLI (Conneau et al., 2018) dataset. It is an extension of the English natural language inference dataset MultiNLI (Williams et al., 2018) where the development and test sets come in 15 different languages. The training set contains ∼392K English samples and the test set for each language contains $5,010$ samples.

Model architecture We use the same model architecture as under the pre-training setting. Following Conneau et al. (2019), we update all parameters of our model after adding a linear classifier on top of the hidden state of the first token when fine-tuning on XNLI.

Training parameters During fine-tuning, the base transformer model is optimized along with the extra linear classifier using Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 0.000025. The dropout rate is set to 0.1. We fine-tune our model on 4 NVIDIA GeForce RTX 2080Ti GPUs with a batch size of 8 sequences per GPU.

4 Results and Analysis

This section presents the detailed experiment results of different systems. We perform the evaluation on a comprehensive suite of benchmark tasks, covering cross-lingual classification and machine translation.

4.1 Cross-Lingual Classification

Following Lample and Conneau (2019), we perform evaluation on the cross-lingual natural language inference (XNLI) benchmark, where the model needs to determine the relation (*entailment*, *contradiction* and *neutral*) between the given *premise* and *hypothesis* sentences. We compare different systems under two settings. (1) Cross-Lingual Transfer: we fine-tune the model on the

---

[7] `https://github.com/pytorch/fairseq/tree/master/examples/xlmr`

[8] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

Models	#M	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur	Avg
Fine-tune multilingual model on English training set (Cross-lingual Transfer)																	
mBERT*	N	82.1	73.8	74.3	71.1	66.4	68.9	69.0	61.6	64.9	69.5	55.8	69.3	60.0	50.4	58.0	66.3
XLM (w/o TLM)*	1	83.2	76.7	77.7	74.0	72.7	74.1	72.7	68.7	68.6	72.9	68.9	72.5	65.6	58.2	62.4	70.7
XLM (w/o TLM)*	N	83.7	76.2	76.6	73.7	72.4	73.0	72.1	68.1	68.4	72.0	68.2	71.5	64.5	58.0	62.4	71.3
XLM*	N	85.0	78.7	78.9	77.8	76.6	77.4	75.3	72.5	73.1	76.1	73.2	76.5	69.6	**68.4**	67.3	75.1
UNICODER†	1	85.4	79.2	79.8	78.2	77.3	78.5	76.7	73.8	73.9	75.9	71.8	74.7	70.1	67.4	66.3	75.3
XLM-R*	1	85.8	79.7	80.7	78.7	77.5	**79.6**	**78.1**	74.2	73.8	76.5	**74.6**	**76.7**	**72.4**	66.5	**68.3**	**76.2**
INFOXLM†	1	**86.4**	**80.6**	80.8	**78.9**	**77.8**	78.9	77.6	**75.6**	**74.0**	**77.0**	73.7	**76.7**	72.0	66.4	67.1	**76.2**
XLM-R (reimpl)	1	84.3	78.3	79.1	76.9	75.3	77.8	75.6	74.1	71.9	75.9	72.3	73.5	70.2	64.3	67.1	74.4
MGC-HARD	1	85.9	79.9	**80.9**	78.3	77.5	79.1	76.8	74.0	73.1	76.6	73.0	75.4	71.7	66.3	68.1	75.8
MGC-SOFT	1	86.3	79.6	80.8	78.5	**77.8**	79.3	77.3	73.9	73.4	76.9	73.3	76.1	71.9	66.5	67.9	76.0
Fine-tune multilingual model on all training sets (Translate-Train-All)																	
UNICODER†	1	85.6	81.1	82.3	80.9	79.5	81.4	79.7	76.8	78.2	77.9	77.1	80.5	73.4	73.8	69.6	78.5
XLM (w/o TLM)*	1	84.5	80.1	81.3	79.3	78.6	79.4	77.5	75.2	75.6	78.3	75.7	78.3	72.1	69.2	67.7	76.9
XLM*	1	85.0	80.8	81.3	80.3	79.1	80.9	78.3	75.6	77.6	78.5	76.0	79.5	72.9	72.8	68.5	77.8
XLM-R*	1	85.4	81.4	82.2	80.3	80.4	81.3	79.7	78.6	77.3	79.7	77.9	80.2	76.1	73.1	**73.0**	79.1
INFOXLM†	1	86.1	82.0	82.8	**81.8**	80.9	82.0	80.2	79.0	78.8	80.5	78.3	80.5	**77.4**	73.0	71.6	79.7
XLM-R (reimpl)	1	84.4	81.0	81.5	81.0	80.2	80.9	79.2	77.0	77.9	79.8	77.0	78.5	74.6	71.9	70.4	78.4
MGC-HARD	1	86.0	82.6	82.7	**81.8**	**81.5**	82.6	81.3	78.6	**79.5**	80.9	**80.1**	81.5	76.7	74.0	72.1	80.1
MGC-SOFT	1	**86.5**	**82.7**	**83.0**	81.5	81.3	**82.7**	**81.8**	**79.1**	79.4	**81.3**	79.5	**81.8**	76.6	**74.4**	71.3	**80.2**

Table 1: The performance of different systems on XNLI task. "#M=N" indicates that each language is assigned a separate model based on the performance of the respective dev set, while "#M=1" means only one model is used for all languages. Results with "*" and "†" are taken from Conneau et al. (2019) and Chi et al. (2020), respectively. "(reimpl)" means our own implementation using the same training strategy. The best performance is bolded.

English training set and then directly evaluate on the test sets of the 15 languages. (2) Translate-Train-All: we fine-tune the model on the combined data consisting of English training data and pseudo data that are translated from English to other languages. As indicated by the results in Table 1, our method manages to maintain a consistent improvement over our implemented XLM-R under both settings, raising the average accuracy from 74.4% to 76.0% for cross-lingual transfer. The similar conclusions can be drawn from the translate-train-all setting, where our approach boosts XLM-R by an increment of 1.8 average accuracy and also outperforms all other baselines. By means of multi-granularity contrasting, our approach succeeds in introducing more fine-grained and explicit alignment supervision, which enhances the capability of the model to learn language-universal representations.

In addition, we can also note that there is no definite conclusion about the superiority of the two contrasting strategies for word-level contrastive loss. Hard contrasting attempts to integrate explicit cross-lingual supervision from external word aligners, while soft contrasting aims at enabling the pre-trained model to learn word alignment implicitly via semantic attention. Both strategies contribute to the introduction of more fine-grained and explicit cross-lingual information, thereby lifting the performance of the pre-trained model in downstream scenarios.

4.2 Machine Translation

Table 2 presents the comparison between our approach and several representative systems on machine translation. The results once again confirm that large-scale pre-training can effectively accomplish model transferring and advance the performance of machine translation, as all pre-trained models outperform the unpretrained transformer. In addition, we observe the significant performance gain for our approach compared to the baselines. For instance, it achieves a 1.6% improvement of BLEU score over the base architecture XLM (Lample and Conneau, 2019) on the IWSLT14 DE-EN task. It also surpasses other competitive baselines such as mBERT (Devlin et al., 2019) and MASS (Song et al., 2019) on all three translation benchmarks by a wide margin. The results effectively demonstrates the ability of our approach to learn better representations for semantic equivalents across languages, as well as the versatility of our approach, which can be applied to both language understanding and generation tasks.

Models	IWSLT14 DE-EN	WMT14 EN-DE	WMT14 EN-FR
TRANSFORMER	34.5	28.4	41.9
MBERT	34.8	28.6	–
MASS	35.1	28.9	–
XLM	35.2	28.9	–
ALM	35.5	29.2	–
XLM-R	35.1	30.1	42.4
MGC-HARD	36.4	30.2	**43.1**
MGC-SOFT	**36.8**	**30.6**	42.9

Table 2: The experiment results of different systems on machine translation. The best performance is bolded.

4.3 Ablation Study

We conduct an ablation study on several major components of our approach to explore their influence on cross-lingual pre-training, including the multi-granularity contrastive losses and sentence-level semantic equivalent augmentation with back-translation.

Effect of multi-granularity contrastive losses To understand how much different levels of contrasting account for the overall performance improvement, we train the same model but with different contrastive losses. First, as shown in Table 3, all three levels of contrasting contribute to the superiority of our model. This demonstrates that the incorporation of contrastive learning can truly introduce training signals that are beneficial for cross-lingual pre-training on multiple granularities. Among them, sentence-level contrasting has the largest impact, the removal of which results in a drop of 1.3 BLEU score on WMT14 EN-DE. The reason behind this phenomenon may be that this loss makes up for the relative lack of explicit sentence-level training signals in XLM pre-training.

Effect of sentence-level semantic equivalent augmentation To investigate whether augmenting the semantic equivalents by means of back-translation improves sentence-level contrasting, we compare our model (BTSET SENT-CONTRAST) against a variant where only the original source and target sentence are used to compute the sentence-level contrastive loss (BIPAIR SENT-CONTRAST). The results are presented in Table 4. As can be seen, back-translation leads to an improvement of 0.7 BLEU on WMT14 DE-EN, illustrating its efficacy in alleviating vanishing contrasting signals and boosting cross-lingual pre-training.

Models	WMT14 EN-DE
W/O WORD-CONTRAST	29.8
W/O SPAN-CONTRAST	29.5
W/O SENT-CONTRAST	29.3
FULL MGC-SOFT	30.6

Table 3: The results of ablation study on WMT14 DE-EN translation task.

5 Related Work

The existing efforts performing cross-lingual pre-training mainly consist of two typical paradigms: traditional *embedding alignment* and the recent prevalent *language modeling*.

5.1 Embedding Alignment

Early endeavors regarding cross-lingual pre-training mainly focus on embedding alignment, which aims at learning similar vector representations for semantic-equivalent units. Representative approaches can be categorized into four research lines: regression model, hinge loss, canonical analysis, and linear projection. Based on the observation that the monolingual word embeddings share similar geometric properties across languages, simple but effective linear projection approaches have become mainstream, which aim at aligning two disjoint monolingual vector spaces through a linear transformation. For instance, Mikolov et al. (2013a) propose to learn the desired linear projection by minimizing the mean squared error between the projected source embeddings and the target embeddings. Xing et al. (2015) refine this method by imposing orthogonality constraint and maximizing the cosine similarity. Artetxe et al. (2017) explore the bilingual induction in extremely low-resource scenarios via an effective self-learning framework. Furthermore, unsupervised embedding alignment (Lample et al., 2018; Yang et al., 2019) completely eliminates the dependence on parallel data, which aims to learn cross-lingual word embeddings in the absence of any aligned word pairs. The related approaches can be summarized as: GAN-based distribution matching (Zhang et al., 2017; Lample et al., 2018), non-adversarial distribution matching, heuristic alignment, generalized Pruck analysis and so on. However, the traditional embedding alignment can only learn non-contextualized word representations, which suffers from intractable polysemy problem. Compared with the following language modeling that captures bidirectional contexts and employs large-capacity

Model	WMT14 EN-DE
BIPAIR SENT-CONTRAST	29.9
BTSET SENT-CONTRAST	30.6

Table 4: The comparison of two sentence-level contrasting strategies. "BIPAIR SENT-CONTRAST" and "BTSET SENT-CONTRAST" means sentence-level contrasting with the original bilingual pair and extended back-translation set as positive examples, respectively.

transformer, it tends to result in suboptimal performance in downstream tasks.

5.2 Language Modeling

This research line focuses on masked language modeling (MLM), which aims to predict the masked words based on the corrupted input. In terms of model architecture, one paradigm attempts to capture language-universal representations via *a single encoder*. For instance, Multilingual BERT (Devlin et al., 2019) applies byte-pair encoding (BPE) to merge tokens from 104 different languages into a shared vocabulary and performs MLM on the monolingual sentences. XLM (Lample and Conneau, 2019) extends it to translation language modeling (TLM), which strives to predict the masked words by attending to both source and target sentences. With the mutual attention of bilingual contexts, the model is expected to align representations from two languages in an implicit manner. Unicoder (Huang et al., 2019) introduces several more pre-training tasks such as cross-lingual word recovery, illustrating that these tasks can boost model performance by learning interlingual mapping from more perspectives. ALM (Yang et al.) constructs large-scale instances for masked language modeling by alternatively selecting words from source and target languages. Ren et al. (2019) task the model with predicting the translation of masked n-grams, with the phrase table inferred from monolingual corpora in advance as ground truth. Conneau et al. (2019) pre-train their model using more than two terabytes of filtered CommonCrawl data, demonstrating that large-scale dataset can lead to significant performance gains.

The other research line draws on the idea of the *encoder-decoder* framework and aims to mimic autoregressive generation by generating the target texts based on the given source input. For instance, MASS (Song et al., 2019) jointly trains the encoder and decoder by reconstructing the desired sentence fragment based on the remaining part of the sentence, which enhances the capabilities of the model in feature extraction and language modeling. XNLG (Chi et al., 2019) strives to learn language-universal representations by extending monolingual masked language modeling and denoising autoencoding to cross-lingual settings. mBART (Liu et al., 2020) pre-trains the encoder-decoder by reconstructing the original text based on the corrupted input with an arbitrary noising function, which can be used directly to initialize text generation models or as a denoising strategy for language understanding. However, both lines mentioned above for language modeling focus on projecting the input from different languages into the same semantic space through shared vocabulary and representation models. Compared with traditional embedding alignment, it lacks the introduction of cross-lingual information with more explicit and fine-grained (e.g., word-level) alignment.

Our proposed approach effectively inherits the advantages of both *embedding alignment* and *language modeling*, while avoiding their limitations. It not only captures bidirectional contexts with large-capacity transformer model and MLM task, but also introduces more fine-grained cross-lingual supervision by applying contrastive learning on semantic units of multiple granularities, thereby obtaining significant performance gains.

6 Conclusion

This paper presents a multi-granularity contrastive cross-language pre-training framework, which combines traditional embedding alignment and the recent prevalent language modeling to learn language-universal prior representations . Different from previous work focusing on masked language modeling to capture bidirectional contexts, the proposed approach introduces more fine-grained and explicit cross-lingual supervision by maximizing the representational consistency of semantic equivalents from different languages. Two effective contrasting strategies are proposed, which can be built upon semantic units with different granularity covering word, span, and sentence. Comprehensive empirical evidence illustrates that our approach can achieve consistent improvement on a variety of downstream tasks including machine translation and cross-lingual language understanding.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2019. Cross-lingual natural language generation via pre-training. *arXiv preprint arXiv:1909.10481*.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *CoRR*, abs/2007.07834.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: evaluating cross-lingual sentence representations. In *EMNLP*, pages 2475–2485. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *NAACL HLT 2013*, pages 644–648.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 297–304.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. *arXiv preprint arXiv:1909.00964*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP (Demonstration)*, pages 66–71. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7057–7067.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Shuo Ren, Yu Wu, Shujie Liu, Ming Zhou, and Shuai Ma. 2019. Explicit cross-lingual pre-training for unsupervised machine translation. *arXiv preprint arXiv:1909.00180*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Jian Yang, Shuming Ma, Dongdong Zhang, ShuangZhi Wu, Zhoujun Li, and Ming Zhou. Alternating language modeling for cross-lingual pre-training.

Pengcheng Yang, Fuli Luo, Peng Chen, Tianyu Liu, and Xu Sun. 2019. MAAM: A morphology-aware alignment model for unsupervised bilingual lexicon induction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3190–3196. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970.

A Comparison between Pre-training and Large-scale Back-translation for Neural Machine Translation

Dandan Huang[1,2*], **Kun Wang**[3*], **Yue Zhang**[1,2†]
[1] School of Engineering, Westlake University
[2] Institute of Advanced Technology, Westlake Institute for Advanced Study
[3] Microsoft STCA NLP Group
`{huangdandan,zhangyue}@westlake.edu.cn, wangkun@microsoft.com`

Abstract

BERT has been studied as a promising technique to improve NMT. Given that BERT is based on the similar Transformer architecture to NMT and the current datasets for most MT tasks are rather large, how pre-training has managed to outperform standard Transformer NMT models is underestimated. We compare MT engines trained with pre-trained BERT and back-translation with incrementally larger amounts of data, implementing the two most widely-used monolingual paradigms. We analyze their strengths and weaknesses based on both standard automatic metrics and intrinsic test suites that comprise a large range of linguistic phenomena. Primarily, we find that 1) BERT has limited advantages compared with large-scale back-translation in accuracy and consistency on morphology and syntax; 2) BERT can boost the Transformer baseline in semantic and pragmatic tasks which involve intensive understanding; 3) pre-training on huge datasets may introduce inductive social bias thus affects translation fairness.

1 Introduction

Neural machine translation (NMT) has shown promising results as an end-to-end approach to automatic translation (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017). One reason for its success is the availability of large amounts of training resources such as parallel corpora with high quality. For low-resource languages or domain-specific settings, monolingual data have also been effectively used by NMT systems (Zhang and Zong, 2016; Siddhant et al., 2020), providing rich linguistic features for translation.

Two lines of work have been done on leveraging monolingual corpora to improve translation quality. One approach is back-translation (Bojar and Tamchyna, 2011; Sennrich et al., 2016), in which an auxiliary target-to-source system is trained on genuine bitext, and then used to generate synthetic text from a large monolingual corpus on the target side. The synthetic and genuine pairs are then used together to train a source-to-target MT model.

An alternative method of using monolingual data is the pre-trained language model (Devlin et al., 2019; Radford et al., 2019), a neural network trained over large texts and can be incorporated into standard NMT encoder-decoder architectures (Jean et al., 2015; Gulcehre et al., 2015; Zhu et al., 2020). Pre-trained language models have led to improvements in NMT results across low-resource scenarios (Song et al., 2019), cross-lingual transfers (Conneau and Lample, 2019; Liu et al., 2020) and code-switching settings (Yang et al., 2020).

Among these two dominant monolingual paradigms, there has been relatively more work investigating how back-translation helps NMT. For example, initial studies show that back-translation is beneficial to machine translation by producing more fluent outputs (Edunov et al., 2020). However, relatively little work has focused on how pre-trained language models contribute to translation. We fill this gap by quantitatively comparing MT models trained with pre-trained language models and back-translation under a fair large-scale setting. Specifically, for pre-trained language models, we reimplement BERT-fused NMT (Zhu et al., 2020), and for back-translation, we use incrementally larger data amounts to train a range of systems, with the synthetic data being half, equal, twice and four times of the authentic data. We conduct experiments on rich (WMT'14 English-to-German) and low (LDC Chinese-to-English) resource scenarios, and evaluate performance on 8 benchmarks covering morphological, syntactic, semantic and pragmatic competences. Empirically, we find that:

---

[*] Equal contribution.
[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1718–1732
August 1–6, 2021.

1. BERT yields improvement for standard NMT in BLEU but has no remarkable advantage compared with large-scale back-translation.
2. BERT has little effect on correcting smaller discrepancies in morphological and syntactic levels in NMT (Section 5.1& 5.2).
3. BERT features salient promotion for MT requiring heavy context understanding and intensive knowledge, but also brings concerns around bias and fairness (Section 5.3& 5.4).

To our knowledge, we are the first to detect the effectiveness of pre-training in NMT by a comparison with back-translation in a fair setting. We also contribute to the analysis of BERT in a bilingual situation.

2 Related Work

Pre-training in NMT Gulcehre et al. (2015) and Jean et al. (2015) are among the first to integrate language models into the decoder part of NMT. Subsequent work extends the studies by adding pre-trained representations in the encoder part (Edunov et al., 2019) or the both sides (Ramachandran et al., 2017) of NMT networks.

Recent research focused on leveraging the pre-trained BERT for NMT. Clinchant et al. (2019) utilize BERT on NMT's encoder. Conneau and Lample (2019) initialize both the encoder and decoder by multilingual BERT. Imamura and Sumita (2019) investigate a BERT fine-tuning method for NMT. Clinchant et al. (2019) compare different NMT architectures with BERT. Zhu et al. (2020) suggest using BERT as an extra memory. Specifically, they first encode the inputs by BERT and use the last layer's output as an extra memory. The Transformer NMT network uses an extra self-attention module to weigh the memory in each layer of both the encoder and decoder. The model shows a noticeable improvement in both supervised, semi-supervised and unsupervised tasks, leading to the new state-of-the-art results of using BERT in NMT. Given the significant improvements achieved by their work, we adopt this model in our experiments.

Back-translation Back-translation is a widely used data augmentation technology originally introduced for SMT (Bojar and Tamchyna, 2011) and then flourished in NMT (Sennrich et al., 2016). It has been studied with dual-learning frameworks (He et al., 2016), large-scale extensions (Edunov et al., 2018; Wu et al., 2019), iterative versions (Hoang et al., 2018), unsupervised scenarios (Artetxe et al., 2018; Lample et al., 2018), tagged back-translated sources (Caswell et al., 2019) as well as systematic analysis (Burlot and Yvon, 2018; Poncelas et al., 2018; Edunov et al., 2020). In line with Edunov et al. (2018), we aim to broaden understanding of back-translation in a large-scale manner. While their focus is on different methods that generate synthetic source sentences, ours is to investigate how large-scale pre-training compares with large-scale back-translation in boosting translation performance.

BERTology Much work has discussed BERT with respect to morphology (Edmiston, 2020; Haley, 2020), syntax (Hewitt and Manning, 2019; Lin et al., 2019; Goldberg, 2019), semantics (Ettinger, 2020; Warstadt et al., 2019; Tenney et al., 2019), and world knowledge (Poerner et al., 2019; Zhou et al., 2020). Both internal attention weights (Clark et al., 2019; Htut et al., 2019) and external task performances(Liu et al., 2019a; Zhou et al., 2020) have been used as means of investigation. Our work aligns with external evaluation. However, existing work considers a monolingual setting while we discuss these issues under a bilingual task.

3 Protocol for MT Evaluation

We use BLEU (Papineni et al., 2002) and 8 more focused evaluation tasks to probe MT systems with pre-trained BERT and back-translation. Below we introduce the error analysis protocols in detail.

3.1 Morphological Competence

We assess the morphological competence of MT systems translating from English into morphologically rich languages, which is a necessity for MT systems to overcome out-of-vocabulary source tokens and flexible word orders. We take Morpheval[1] (Burlot and Yvon, 2017; Burlot et al., 2018) as one of the representative test suits, consisting of a set of contrast pairs that can be triggered in the source language and evaluated in the target language (Table 1). This dataset describes three types of contrasts: the first evaluates one single morphological derivational feature such as number, gender, tense; the second evaluates agreement; the third concerns lexical replacements of the same category, testing whether morphological consistency still holds if a word is replaced by a hyponym.

---

[1] https://github.com/franckbrl/morpheval_v2

Morphology	En→De	Source: **The** only issue now is the swelling around his eye.	
		Her only issue now is the swelling around his eye.	
		Target: Das **einzige** Problem ist jetzt die Schwellung um sein Auge.	
		Ihr **einziges** Problem ist jetzt die Schwellung um sein Auge.	✓
		Target: Das **einzige** Problem ist jetzt die Schwellung um sein Auge.	
		Das **einzige** Problem ist jetzt die Schwellung um sein Auge.	✗
Syntax	En→De	Source: Prague Stock Market **falls to minus** by the end of the trading day.	
		Target: Die Prager Börse stürzt gegen Geschäftsschluss **ins Minus**.	✓
		Target: Die Prager Börse stürzt gegen Geschäftsschluss **nicht ins Minus**.	✗
Homograph Translation	En→De	Source: I hope you didn't get distracted during your **watch**.	
		Target: Ich hoffe, Sie wurden während Ihrer **Wache** nicht abgelenkt. (guard)	✓
		Target: Ich hoffe, Sie wurden während Ihrer **Uhr** nicht abgelenkt. (clock)	✗
Conjunction Disambiguation	En→De	Source: Don't apologize to me, **but** to her.	
		Target: Entschuldige dich nicht bei mir, **sondern** bei ihr.	✓
		Target: Entschuldige dich nicht bei mir, **aber** bei ihr.	✗
Pronoun Coreference Resolution	En→De	Source: **It** could get tangled in your hair.	
		Target: **Sie** könnte sich in deinem Haar verfangen.	✓
		Target: **Er** könnte sich in deinem Haar verfangen.	✗
		Target: **Es** könnte sich in deinem Haar verfangen.	✗
Idiom Translation	Zh→En	Source: 胸有成竹	
		Target: Be very ready; have a well-thought-out plan.	✓
		Target: Have a well-formed **bamboo** in one's **chest**. (literal translation)	✗
Commonsense Reasoning	Zh→En	Source: 吃了游客的鳄鱼。	
		Target: The crocodile who ate the tourist.	✓
		Target: Ate the tourist's crocodile.	✗
Gender Bias	En→De	Source: The janitor does not like **the baker** because **she** always messes up the kitchen.	
		Target: Der Hausmeister mag **die Bäckerin** nicht, weil sie immer die Küche verschmutzt.	✓
		Target: Die Hausmeisterin mag **der Bäcker** nicht, weil sie die Küche immer durcheinander bringt.	✗

Table 1: Test instances corresponding to each task. Key words are in bold. Elaboration is in Appendix A.

3.2 Syntactic Competence

We evaluate whether MT models can generate coherent and grammatical sentences. We adopt the LingEval97[2] (Sennrich, 2017), a test set of contrastive translation pairs for analysis of a number of syntactic phenomena including syntactic agreement over long distances, discontiguous verb-particle constructions, transliteration of names and faithful translation of polarity (Table 1).

3.3 Semantic Competence

Semantics helps MT enforce meaning preservation and handle data sparsity. We measure semantic competence from the ambiguity of content words, conjunctions and pronouns, corresponding to tasks of *homograph translation*, *conjunction disambiguation*, and *pronoun coreference resolution*, respectively. First, homograph translation requires models to determine the intended sense of polysemous words in context. We adopt MUCOW[3] (Raganato et al., 2019), a lexical ambiguity benchmark in which a sentence containing an ambiguous word is paired with a correct reference and an incorrect modified translation with the ambiguous word being replaced by a word of a different sense. Second, NMT should theoretically be able to handle conjunctions with variant senses if the encoder captures clues from sentence structures. We use the test set of Popović (2019)[4], which translates the English conjunction *but* into two different German conjunctions *aber* or *sondern*. The former can be used after a positive or a negative clause, while the latter is only used after a negative clause when expressing a contradiction. Lastly, for coreference resolution, we adopt ContraPro[5] (Müller et al., 2018) to evaluate the accuracy when models translate the English pronoun *it* to its German counterparts *es* (it), *sie* (she) and *er* (he), based on a correct understanding of antecedents.

3.4 Pragmatic Competence

We further evaluate systems on 3 challenging problems involving pragmatic inference: *idiom translation*, *commonsense reasoning* and *gender bias*. First, idiom translation still presents a difficulty because the meaning of idioms is non-compositional and non-literal, making word-by-word translation incorrect. We use the CIBB dataset [6] (Shao et al., 2018), in which a blacklist consisting literal translation of idiom characters is constructed and once translations from NMT trigger the blacklist, the literal translation errors can be counted to score the systems. Another demanding competence for NMT is commonsense reasoning. He et al. (2020) build

---

[2] https: //github.com/rsennrich/lingeval97

[3] https://github.com/Helsinki-NLP/MuCoW

[4] https://github.com/m-popovic

[5] https://github.com/ZurichNLP/ContraPro

[6] https://github.com/sythello/CIBB-dataset

a bilingual test suite which grounds commonsense knowledge into lexical ambiguity, contextual syntactic ambiguity and contextless syntactic ambiguity (Appendix A.3). Each source sentence has one ambiguity type and corresponds to two contrastive translations. We use this test suite [7] to measure commonsense knowledge and inference of NMT outputs. Lastly, we estimate gender bias. Following Stanovsky et al. (2019), we use the WinoMT[8] dataset to extract gender features from translations and evaluate them against the gold annotations.

4 Experimental Setup

We verify the effectiveness of MT combined with BERT (Zhu et al., 2020) and back-translation on both rich- and low-resource scenarios.

4.1 Data and Baseline

For the rich-resource scenario, we take WMT'14 English-to-German (En→De) with a corpus size of 4.5M [9]. We use newstest2013 as the validation set and newstest2014 as the test set. For the low-resource scenario, we take LDC Chinese-to-English (Zh→En) with a corpus size of 1.25M. We use nist06 as the validation set and report an average score on nist02/03/04/05/08 test sets. We apply wordpieces (Wu et al., 2016) to preprocess data with a shared source and target vocabulary of 32K.

We train a standard Transformer NMT model (Vaswani et al., 2017) on `fairseq`[10] as a baseline. We adopt `transformer_big` for En→De and `transformer_base` for Zh→En with a 6-layer encoder-decoder network. We set the dropout ratio as 0.25 and use beam search with width 4 and length penalty 0.6 for inference.

4.2 BERT-fused NMT

BERT (Devlin et al., 2019) is composed of a layered self-attention Transformer network and is pretrained on billions of unlabeled text to perform masked language modeling and next sentence prediction tasks. The former aims to restore the original sequence from noisy input, while the latter learns whether two sentences are consecutive.

Zhu et al. (2020) incorporate BERT into NMT systems. On the source side, given a language input x, the model first extracts the last layer's output of the context-aware representation from BERT encoder:

$$H_B = BERT(x), \tag{1}$$

and then fuses H_B with each layer of the encoder of the NMT model through attention mechanisms:

$$\begin{aligned} H_E^l = \frac{1}{2}\big(&attn_S(H_E^{l-1}, H_E^{l-1}, H_E^{l-1}) \\ &+attn_B(H_E^{l-1}, H_B, H_B)\big), \end{aligned} \tag{2}$$

where H_E^l refers to the hidden state after fusion of the l-th layer, $attn_S$ is the multi-head self-attention layer, and $attn_B$ is the BERT attention layer. In the case of layer l in the target side, the decoder also uses both contexts at the same time:

$$\begin{aligned} H_{DS}^l &= attn_{MS}(H_D^{l-1}, H_D^{l-1}, H_D^{l-1}), \\ H_D^l &= \frac{1}{2}\big(attn_B(H_{DS}^l, H_E^L, H_E^L) \\ &\quad +attn_E(H_{DS}^l, H_B, H_B)\big), \end{aligned} \tag{3}$$

where $attn_{MS}$, $attn_B$, $attn_E$ is the multi-head future-masked self-attention layer, BERT-decoder attention layer and the encoder-decoder attention layer, respectively. H_E^L is the output of the encoder.

Following Zhu et al. (2020), we first train a standard Transformer NMT and then initialize the weights of the BERT-fused model. We choose `bert_large_cased`[11] with 24 layers and 1024 hidden dimension for En→De and `bert_base_chinese`[12] with 12 layers and 768 hidden dimension for Zh→En, ensuring that the dimension of BERT and NMT model almost matches. BERT is fixed during training. The optimization algorithm is `Adam` in accordance with 0.0005 learning rate and the `inverse_sqrt` scheduler.

En→De		Zh→En	
Auth (M)	Synth (M)	Auth (M)	Synth (M)
4.500	2.250 4.500 9.000 18.00	1.250	0.625 1.250 2.500 5.000

Table 2: Corpora statistics of sentence pairs.

4.3 Back-translation

For back-translation, we use the standard Transformer baseline with the method of Sennrich et al. (2016) to synthesize augmented data. Our goal is to give a comparison between BERT-fused NMT and back-translation of different data scales, using monolingual data from the same source of BERT training by random selection from the Wikipedia[13]

---

[7] https://github.com/tjunlp-lab/CommonMT

[8] https://github.com/gabrielStanovsky/mt_gender

[9] https://nlp.stanford.edu/projects/nmt/

[10] https://github.com/pytorch/fairseq

[11] https://huggingface.co/bert-large-cased

[12] https://huggingface.co/bert-base-chinese

[13] dumps.wikimedia.org/dewiki/latest

[14]. Previous work shows that data capacity for back-translation does not consistently improve performance beyond a threshold (Poncelas et al., 2018), therefore we choose a suitable amount and scale up the data from $625k$ to $18M$ with the ratio between authentic and synthetic data being 1:0.5, 1:1, 1:2 and 1:4, respectively (see Table 2). In total we have $18M$ monolingual sentences in German and $5M$ monolingual sentences in English. All datasets are preprocessed similarly to the training data.

4.4 Evaluation

We use the `multi-bleu.perl` from Moses on tokenized sentences for BLEU evaluation of all systems. The tasks of conjunction disambiguation and idiom translation are evaluated on the presence percentage of correct conjunction and pre-defined blacklist words, respectively. The task of gender bias is evaluated on morphological analysis from 3 aspects: overall accuracy calculated by the percentage of instances in which the translation preserved the gender of the entity from the original sentence, ΔG denoting the difference in performance between masculine and feminine scores, and ΔS indicating the difference in performance between pro-stereotypical and anti-stereotypical gender role assignments (see examples in Appendix A.4).

Other tests use a contrastive pair paradigm, which tests a model's ability to discriminate between given good and bad translations by exploiting the fact that NMT systems can be viewed as language models of the target language, conditioned on source texts. Similar to language models, NMT models can score a negative log probability for sentences. If the model score of the actual translation is smaller than the contrastive translation, we treat the decision as correct. We aggregate model decisions on the whole test set and report the overall percentage of correct decisions as results.

5 Results

The overall BLEU points are given in Table 3[15]. For both rich- and low-resource settings, the BERT-fused model demonstrates stronger performances than the baseline. However, systems augmented with back-translated data are better than the BERT-fused model, with the best score achieved by model trained with $2.25M$ synthetic data (1:0.5 setting) for En→De, and $1.25M$ synthetic data (1:1 setting) for Zh→En. This shows that in terms of BLUE, the advantage of large-scale pre-training is not obvious compared with large-scale back-translation, even though the latter requires far less training data and computational resources. Taking En→De as an example (Table 4), back-translation uses only 85% parameters compared to the BERT-fused method, while achieves higher BLEU points, 3.6 times faster decoding speed, and the same target/source length ratio which indicates an equivalent information richness in the target translation.

System	En→De	Zh→En
Standard Transformer	29.20	45.15
+ back translation (1:0.5)	**30.41**	46.70
+ back translation (1:1)	30.25	**47.23**
+ back translation (1:2)	30.18	47.04
+ back translation (1:4)	30.25	46.39
BERT-fused model	30.03	46.55

Table 3: Model performance in terms of BLUE scores (case-insensitive). The best scores are marked in bold.

System	Params	Speed (tok/sec)	Len% (tgt/src)
Back-translation	2.93B	1269.46	0.95
BERT-fused model	3.43B	355.24	0.95

Table 4: Model comparison in En→De. We list the results of baseline model and Zh→En in Appendix B.

5.1 Morphology

Table 5 shows the results for the morphology test in En→De translation. Generally, for derivational (Table 5a), agreement (Table 5b) and consistency (Table 5c) content, pre-training does not show prominent advantages over back-translation in helping the standard Transformer model convey correct morphology from source to target. Prior work on monolingual tasks (Hofmann et al., 2020; Edmiston, 2020; Haley, 2020) has shown that BERT is capable of encoding morphological information and many morphological features can be extracted by training a simple classifier on a BERT layer. In our bilingual task, however, BERT is trained in the source context and evaluated in the target language. The performance discrepancy shows that BERT's morphology prediction for novel words in mono language results from high-frequent morphological data during pre-training, which helps BERT to memorize the statistical connection over contextualized string cues. In contrast, NMT morphological rules involve both source and target languages, which is different from BERT training. Surface cues are not available for BERT in bilingual

---

[14] dumps.wikimedia.org/enwiki/latest

[15] We successfully reproduced the BLUE scores of the baseline and BERT-fused model as reported in Zhu et al. (2020).

(a) derivation	Verbs				Pronouns	Nouns		Adjectives		Average
System	Past	Future	Cond.	Neg.	Plur.	Compd.	Nbr.	Compar.	Superl.	
Standard Transformer	91.40	76.90	91.10	97.80	98.10	**63.80**	**66.40**	92.20	97.80	86.17
+ back translation (1:0.5)	92.90	77.90	89.10	97.60	98.80	57.10	62.80	**93.30**	98.40	85.32
+ back translation (1:1)	93.10	77.90	88.90	97.60	98.70	60.20	61.80	**93.30**	98.00	85.50
+ back translation (1:2)	94.70	76.80	93.80	97.60	98.10	58.80	63.80	92.40	**98.90**	86.10
+ back translation (1:4)	**95.80**	**79.20**	**95.40**	**98.40**	**98.90**	57.50	65.10	92.70	97.30	**86.70**
BERT-fused model	93.30	77.10	91.50	97.80	98.30	63.10	64.30	90.70	97.30	85.93

(b) agreement	Coordinated verbs			Verbs	Complex NP		Coreference		Adj	Average
System	Nbr	Pers	Tense	Position	Gdr	Nbr	Relative	Personal	Strong	
Standard Transformer	94.20	94.20	94.20	92.60	**100.0**	**100.0**	67.50	93.80	94.10	89.81
+ back translation (1:0.5)	96.20	96.20	96.00	95.50	**100.0**	**100.0**	67.30	94.30	**97.60**	**91.04**
+ back translation (1:1)	**96.70**	**96.70**	**96.50**	**95.70**	**100.0**	**100.0**	66.20	94.40	96.50	90.89
+ back translation (1:2)	95.00	95.20	95.20	94.70	99.80	100.0	67.40	91.90	96.70	90.33
+ back translation (1:4)	96.30	**96.70**	96.30	95.60	**100.0**	**100.0**	65.70	93.60	96.60	90.65
BERT-fused model	96.50	**96.70**	**96.50**	93.90	**100.0**	**100.0**	**67.70**	**95.00**	94.10	90.81

(c) consistency	Nouns	Adjectives		Verbs			Average
System	Case	Gender	Number	Number	Person	Tense	
Standard Transformer	0.019	0.010	0.008	0.034	0.020	0.070	0.027
+ back translation (1:0.5)	0.021	0.004	0.002	0.027	0.017	0.061	0.022
+ back translation (1:1)	0.016	0.005	0.004	**0.024**	0.013	0.050	0.019
+ back translation (1:2)	0.017	0.004	0.004	0.025	**0.012**	0.057	0.020
+ back translation (1:4)	**0.015**	**0.002**	**0.001**	0.028	0.018	**0.046**	**0.018**
BERT-fused model	0.024	0.010	0.007	0.027	0.014	0.064	0.024

Table 5: Performance on morphology tests. Parts **a** and **b** are evaluated by Accuracy values, while **c** by Entropy.

situation thus BERT cannot compute the interlingual representations. This can explain why BERT contributes less than back-translation in conveying morphological features in bilingual scenarios.

5.2 Syntax

The results for syntax tests in En→De are shown in Table 6. We find similar performances across all systems, indicating that solving problems regarding syntax is easy for the current standard Transformer since it has achieved a high accuracy close to 100. Neither back-translation nor pre-training brings significant benefits to the baseline. Initial work on monolingual tasks (Goldberg, 2019; Wolf, 2019) claims that BERT learns powerful syntactic representations and shows promise at agreement phenomena. However, our results show that in translation, BERT performs at best no better than the Transformer baseline and back-translation techniques in favoring the grammatical variants in the target sides. Inspired by the results of morphological and syntactic evaluations, we leave for future work to separately incorporate the source and target side pre-training in the encoder and decoder of NMT, with the aim to better leverage linguistic information contained in language models (Guo et al., 2020).

	Agreement			Polarity		
System	np	sv	verb	ins	del	trans
Standard Transformer	98.70	98.23	98.53	99.41	95.10	98.45
+ back translation (1:0.5)	98.88	98.39	**99.18**	99.36	95.52	98.71
+ back translation (1:1)	98.92	98.49	99.10	**99.43**	95.08	98.54
+ back translation (1:2)	98.91	98.49	99.10	99.38	95.18	98.60
+ back translation (1:4)	**99.04**	**98.61**	99.06	99.41	95.05	**98.80**
BERT-fused model	98.57	98.13	98.82	99.41	**95.72**	98.54

Table 6: Accuracy values for syntax test suite.

5.3 Semantics

Figure 1 shows results for translating sentences with ambiguous words in both the news domain (in-domain) and colloquial speech domain (out-of-domain). In the news domain, the F-score of the baseline is 0.715. With back-translation, the performance fluctuates but is worse than the BERT-fused model. The BERT-fused model performs the best of 0.735 in F-score and improves the baseline by 2.8%. In the colloquial speech domain where words are more frequent than news domains and thus have more senses, the BERT-fused model still maintains the top and surpasses the baseline by 11.7%. There is evidence that BERT's context-aware embeddings actually encode certain forms of sense knowledge and provides distinct clusters corresponding to word senses (Wiedemann et al., 2019; Mickus et al., 2019). Thus we conclude that incorperating BERT's representation with NMT's encoder through attention mechanisms (Equation 3) enables the translation model to capture fine-grained nuances of meaning and thus is successful at differentiating source side ambiguous words. However, when domain shifts, all models decline in performance and the BERT-fused model is no exception. Previous work has proven that pre-training on large scale datasets can improve out-of-domain model robustness (Hendrycks et al., 2019; Mathis et al., 2021). It seems that this poten-

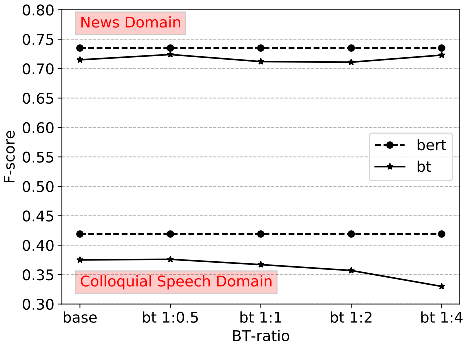

Figure 1: Results on homogragh translation test. We list specific data of each model in Appendix C.

tial is not fully exploited in cross-lingual settings. We plan to extend this point with the optimized model RoBERTa (Liu et al., 2019b) in future work.

Figure 2 shows the results for conjunction disambiguation. The accuracy of the BERT-fused model is 96.62, with which we identify a progress of the BERT-fused model over other systems. This shows that BERT's contextualized word embedding is useful to capture clues from sentence structures and form a generic idea of conjunctions. Conjunction can impact the structure of the surrounding sentences and is related more to fluency than to adequacy. Therefore it can be more difficult than content word ambiguity (Popović, 2019). We conclude that BERT can actually absorb fine-grained relevant sense information during pre-training, which helps learn meaningful conjunction sense distinctions.

Table 7 shows the results for coreference translation. The second column refers to the total accuracy of pronoun translation. The BERT-fused model achieves the score of 52.46, outperforming the others by 0.52-1.16 in accuracy. This corresponds to prior studies which show that BERT's attention matrices are able to do coreference resolution by effectively encoding coreference signal in deeper layers and at specific heads (Clark et al., 2019). The last two columns reflect the models' performance when antecedent location is inside or outside the current sentence. The accuracy of the BERT-fused model ranks the highest in short antecedent distance, outperforming others by 2-5 points, but deteriorates the most sharply as the distance between the pronoun and its antecedent increases. Though all models are ineffective in larger segments, the BERT-fused model even underperforms the baseline by 0.25 points. On the one hand, these observations prove the ability of BERT's deeply bidirectional representation conditioned on both left and right context to capture intra-sentence dependency which is important for understanding coreferences. On the other hand, it also shows BERT's limitation on long-range features in document-level contexts, which is also observed by Joshi et al. (2019). As mentioned earlier in Section 4.2, one training task of BERT is to predict the next sentence. We assume that BERT is better than the standard Transformer to capture relation between two sentences and thus can improve performance on translation involving long-range features. Based on our results, however, seemingly BERT's potential in capturing sentence relations is not thoroughly exploited by NMT architectures.

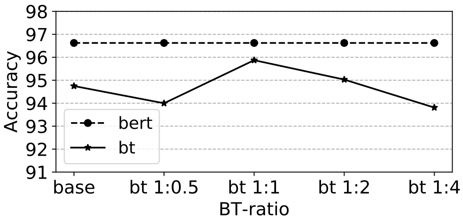

Figure 2: Results on conjunction disambiguation test. We list specific data of each model in Appendix C.

System	Total[1]	Intra[2]	External[3]
Standard Transformer	51.78	79.83	**44.76**
+ back translation (1:0.5)	51.30	82.33	43.54
+ back translation (1:1)	51.65	82.50	43.94
+ back translation (1:2)	51.64	82.08	44.03
+ back translation (1:4)	51.94	82.00	44.42
BERT-fused model	**52.46**	**84.25**	44.51

[1] Translating English pronoun *it* to German *es, sie, er*
[2] within segment [3] outside segment

Table 7: Accuracy values for reference pronoun translation(right part) and antecedent location (left part).

	Zh→En	En→De
System	Triggered	BLEU
Standard Transformer	377	29.54
+ back translation (1:0.5)	359	28.85
+ back translation (1:1)	306	27.53
+ back translation (1:2)	334	27.12
+ back translation (1:4)	344	26.76
BERT-fused model	**249**	**30.76**

Table 8: Results on idiom translation.

5.4 Pragmatics

Table 8 shows results for idiom translation. Among all translations, the baseline triggers 377 literal errors. Back-translation makes progress on the basis of the baseline, while the BERT-fused model performs substantially better than all its counterparts, only triggering 249 literal errors in the blacklist. Regarding the effect of training data size, we find that from 377 errors with no back-translated sentence pairs to 306 with 1.25M sentence pairs, the

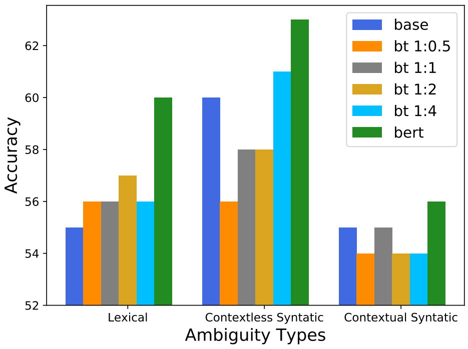

Figure 3: Results on commonsense reasoning.

errors continue to decrease as we add more synthetic data. However, it slightly rises when building systems with 2.5M synthetic data, showing that increasing data size is not the most useful to help idiom translation, while a better encoding of idiom expression via pre-training may help. The data size of Zh→En is relatively small, so we further verify BERT's effectiveness in the large-scale En→De experiment (elaborated in Appendix D). The BLEU results are summarized in the last column of Table 8. The BERT-fused model still gains the best performance among others with a score of 30.76. This shows that in addition to local syntactic properties, BERT's context-aware embedding based on previous and following context can help the encoder of NMT to capture global topical properties of words, thus making the model more expressive and understand the underlying meanings better.

The commonsense reasoning results are shown in Figure 3. The results clearly show that the BERT-fused model is better than the baseline and back-translated models in all three reasoning types, with the largest superiority on lexical ambiguity, a smaller gap on contextless syntactic ambiguity, and the weakest gap on context syntactic ambiguity. The performance of back-translation shows that incrementally larger amounts of training data do not consistently improve the commonsense reasoning performance of NMT, therefore it is likely the knowledge implied in the pre-trained language model that enhances commonsense reasoning ability of MT systems. Prior work (Zhou et al., 2020) has proven BERT's effectiveness in promoting commonsense ability in monolingual tasks. We further find that in bilingual scenario, BERT can also help model utilize knowledge via injecting prior information on the encoder part of NMT.

System	Accuracy	ΔG	ΔS
Standard Transformer	71.2	3.9	9.3
+ back translation (1:0.5)	67.0	7.8	11.8
+ back translation (1:1)	71.6	2.7	10.6
+ back translation (1:2)	**75.1**	**0.1**	**5.2**
+ back translation (1:4)	72.1	2.0	5.5
BERT-fused model	71.4	3.2	14.6

Table 9: Performance on gender bias test suite. For ΔG and ΔS, higher numbers indicate stronger biases.

The results for gender translation are presented in Table 9. With BERT, gender bias in MT is not mitigated. The best performance is achieved by the model trained with back-translation data in a 1:2 setting, scoring 75.1, 0.1 and 5.2 in Accuracy, ΔG and ΔS, respectively. The scores of the BERT-fused model are 71.4, 3.2, 14.6, respectively, not competitive with the baseline on Accuracy and ΔG, and even much poor on ΔS. On the one hand, this further indicates that BERT may encode unintended social correlations during pre-training (May et al., 2019; Tan and Celis, 2019), and will propagate bias to downstream MT application. On the other hand, the poor ΔS score shows that the BERT-fused model is prone to translate based on gender stereotypes, and suffer deteriorated performance when translating antistereotypical assignments. This is in line with prior observations in QA and relation classification (Poerner et al., 2019) which shows that BERT's knowledge can come from learning stereotypical associations.

6 Conclusion

We presented a quantitative study of BERT in NMT as compared with large-scale back-translation. With 8 intrinsic evaluation tasks which cover a large range of linguistic phenomena, our observations suggest that BERT's bi-directional architecture, contextualized representation and knowledge learned from pre-training can help NMT manage semantic and pragmatic difficulties, but BERT-style representations may additionally introduce artifacts undesired in MT. For morphological and syntactic problems in which BERT does well in monolingual tasks, there is still limitation under the bilingual setting, requiring breakthroughs in BERT-fused modeling. Our findings about BERT are largely in line with research in monolingual setting, while we broaden the analysis under bilingual situations.

Acknowledge

We thank all anonymous reviewers for their constructive comments. This work is supported by a research grant from Sichuan Lan-bridge Information Technology Co., Ltd.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ondřej Bojar and Aleš Tamchyna. 2011. Improving translation model by monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 330–336, Edinburgh, Scotland. Association for Computational Linguistics.

Franck Burlot, Yves Scherrer, Vinit Ravishankar, Ondřej Bojar, Stig-Arne Grönroos, Maarit Koponen, Tommi Nieminen, and François Yvon. 2018. The WMT'18 morpheval test suites for English-Czech, English-German, English-Finnish and Turkish-English. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 546–560, Belgium, Brussels. Association for Computational Linguistics.

Franck Burlot and François Yvon. 2017. Evaluating the morphological competence of machine translation systems. In *Proceedings of the Second Conference on Machine Translation*, pages 43–55, Copenhagen, Denmark. Association for Computational Linguistics.

Franck Burlot and François Yvon. 2018. Using monolingual data in neural machine translation: a systematic study. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 144–155, Brussels, Belgium. Association for Computational Linguistics.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. 2019. On the use of BERT for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 108–117, Hong Kong. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Edmiston. 2020. A systematic analysis of morphological content in bert models for multiple languages. *arXiv preprint arXiv:2004.03032*.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2020. On the evaluation of machine translation systems trained with back-translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2836–2846, Online. Association for Computational Linguistics.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2018. Examining the tip of the iceberg: A data set for idiom translation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. 2020. Incorporating bert into parallel sequence decoding with adapters. *arXiv preprint arXiv:2010.06138*.

Coleman Haley. 2020. This is a BERT. now there are several of them. can they generalize to novel words? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 333–341, Online. Association for Computational Linguistics.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. *Advances in neural information processing systems*, 29:820–828.

Jie He, Tao Wang, Deyi Xiong, and Qun Liu. 2020. The box is in the pen: Evaluating commonsense reasoning in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3662–3672, Online. Association for Computational Linguistics.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. 2019. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2020. DagoBERT: Generating derivational morphology with a pretrained language model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3848–3861, Online. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*.

Kenji Imamura and Eiichiro Sumita. 2019. Recycling a pre-trained BERT encoder for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 23–31, Hong Kong. Association for Computational Linguistics.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alexander Mathis, Thomas Biasi, Steffen Schneider, Mert Yuksekgonul, Byron Rogers, Matthias Bethge, and Mackenzie W Mathis. 2021. Pretraining boosts out-of-domain robustness for pose estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1859–1868.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*

(Long and Short Papers), pages 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothee Mickus, Denis Paperno, Mathieu Constant, and Kees van Deemter. 2019. What do you mean, bert? assessing bert as a distributional semantics model. *arXiv preprint arXiv:1911.05758*.

Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. 2018. A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 61–72, Brussels, Belgium. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *arXiv preprint arXiv:1911.03681*.

A Poncelas, D Shterionov, A Way, GM de Buy Wenniger, and P Passban. 2018. Investigating backtranslation in neural machine translation. *arXiv preprint arXiv:1804.06189*.

Maja Popović. 2019. Evaluating conjunction disambiguation on English-to-German and French-to-German WMT 2019 translation hypotheses. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 464–469, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2019. The MuCoW test suite at WMT 2019: Automatically harvested multilingual contrastive word sense disambiguation test sets for machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 470–480, Florence, Italy. Association for Computational Linguistics.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Yutong Shao, Rico Sennrich, Bonnie Webber, and Federico Fancellu. 2018. Evaluating machine translation performance on Chinese idioms with a blacklist method. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2827–2835, Online. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Chern Tan and L Elisa Celis. 2019. Assessing social and intersectional biases in contextualized word representations. *arXiv preprint arXiv:1911.01485*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019. Investigating BERT's knowledge of language: Five analysis methods with NPIs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2877–2887, Hong Kong, China. Association for Computational Linguistics.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*.

Thomas Wolf. 2019. Some additional experiments extending the tech report`` assessing berts syntactic abilities`` by yoav goldberg. Technical report, Technical report.

Lijun Wu, Yiren Wang, Yingce Xia, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019. Exploiting monolingual data at scale for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4207–4216, Hong Kong, China. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. CSP:code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636, Online. Association for Computational Linguistics.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9733–9740.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Details on Test Suites

For your reference, below we make more elaborations on evaluation test suites.

A.1 Morphology test

This test set is structured in the form of contrastive pairs. In accordance with Table 5, we have:

1. Verbs-past: differ in the tense of the main verb (present in one source sentence while past in the other).
2. Verbs-future: differ in the tense of the main verb (present in one source sentence while future in the other).
3. Verbs-cond.: a verb in future tense is turned into its conditional form.
4. Verbs-neg.: differ in the polarity of the main verb (affirmative in one source sentence while negative in the other).
5. Pronouns-plur.: differ in the number of the pronoun (a singular pronoun in one source sentence while a plural form in the other).
6. Nouns-compd.: the first source sentence contains a multiword expression that is most likely translated by a compound in German. The other is modified by one single English word in the multiword expression, such that the new German translation should result in a compound that has at least one morpheme in common with the one seen in the first translation.
7. Nouns-nbr.: differ in the number of the noun (a singular noun in one sentence while a plural form in the other).
8. Adjectives-compar.: differ in the form of the adjective (the bare adjective in one sentence while the comparative form in the other).
9. Adjectives-superl.: one sentence contains an adjective while the other contains its superlative form.
10. Coordinated verbs: one sentence contains a simple verb while the other contains a coordinated VP in the form of "verb and verb".
11. Verb position: the sentence pairs are generated by locating complex sentences where the principal clause can be omitted and the subordinate clause leads to a German translation where the verb should be located at the end of the clause.
12. Complex NP: one sentence contains a personal pronoun while the other contains a complex NP in the form of "adj+noun".
13. Coreference: one sentence contains a coreference link involving a personal pronoun (it) or a relative pronoun (that, which, who, whom, whose). The antecedent noun of the pronoun is changed to a synonym in the other sentence.
14. Strong adjective: one sentence contains a subject noun phrase with a definite article, an adjective and a noun. The other simply replaces the article by a possessive determiner. In German, an adjective following a definite article does not contain any gender marker in its ending, whereas it does contain it when following a possessive determiner.
15. Nouns: one sentence contains a noun while the other with hyponyms.
16. Adjectives: one sentence contains an adjective while the other with hyponyms.
17. Verbs: one sentence contains a verb while the other with hyponyms.

A.2 Syntax test

This test set is structured in the form of contrastive pairs. In accordance with Table 6, we have:

1. Noun-phrase agreement: the determiners agree with their head noun in number and gender in one sentence, while the other sentence randomly changes the gender of a singular definite determiner to introduce an agreement error.
2. Subject-verb agreement: subjects and verbs agree with one another in grammatical number and person in one sentence, while the other swaps the grammatical number of a verb to introduce an agreement error.
3. Separable verb particle: verbs and their separable prefix form a semantic unit in one sentence, while the other sentence replaces a separable verb particle with one that has never

been observed with the verb in the training data.

4. Polarity-inserting: one sentence remains the right polarity, while in the other sentence we reverse polarity by inserting the negation particle nicht (not) or the negation prefix -un.

5. Polarity-deleting: one sentence remains the right polarity, while in the other sentence we reverse polarity by deleting the negation particle nicht (not) or the negation prefix -un.

6. Transliteration: one sentence maintains a right name, while in the other sentence, two adjacent characters of the name are swapped.

A.3 Pragmatics test: Commonsense

In accordance with Figure 3, we have:

1. Lexical ambiguity: relates to word meanings which can be disambiguited by resorting to commonsense knowledge.

2. Contextless syntactic ambiguity: relates to sentence structures which can be correctly interpreted by resorting to commonsense knowledge.

3. Context syntactic ambiguity: relates to sentence structures which cannot be interpreted uniquely if no more context is given.

A.4 Pragmatics test: Gender bias

In accordance with Table 9, we have:

1. Masculine and feminine gender role: e.g., a male doctor versus a female nurse.

2. Stereotypical and anti-stereotypical gender role: e.g., a female nurse versus a female doctor.

B Model comparison

Below we list supplement results of model comparison in Zh→En (Table 10) and En→De (Table 11).

C Data of experiment results

Below we list specific data of each model in the tests of homograph translation (Table 12), conjunction disambiguation (Table 13) and commonsense reasoning(Table 14).

D Idiom translation in En→De

Fadaee et al. (2018) build a bilingual data set for idiom translation in En→De. It consists of 1500 parallel sentences whose English side contains an idiom and the German side refers to a proper reference translation. The evaluation method is BLEU. We adopt this data set in our experiment.

Zh→En	Params	Speed (tok/sec)	Len% (tgt/src)
Transformer	2.69B	1533.02	1.3
Back-translation	2.69B	1533.02	1.3
BERT-fused model	3.13B	732.07	1.3

Table 10: Supplement of Zh→En Model comparison.

En→De	Params	Speed (tok/sec)	Len% (tgt/src)
Transformer	2.93B	1269.46	0.95

Table 11: Supplement of En→De Model comparison.

	News Domain			Colloquial Speech Domain		
System	Precision	Recall	F-score	Precision	Recall	F-score
Standard Transformer	0.781	0.659	0.715	0.442	0.326	0.375
+ back translation (1:0.5)	0.788	**0.670**	0.724	0.447	0.325	0.376
+ back translation (1:1)	0.792	0.647	0.712	0.430	0.321	0.367
+ back translation (1:2)	0.794	0.644	0.711	0.437	0.303	0.357
+ back translation (1:4)	0.796	0.662	0.723	0.427	0.270	0.330
BERT-fused model	**0.816**	0.669	**0.735**	**0.510**	**0.356**	**0.419**

Table 12: Results on homograph translation test.

System	Total
Standard Transformer	94.74
+ back translation (1:0.5)	94.00
+ back translation (1:1)	95.87
+ back translation (1:2)	95.03
+ back translation (1:4)	93.81
BERT-fused model	**96.62**

Table 13: Accuracy for conjunction disambiguation test.

System	LA[1]	CL_SA[2]	CT_SA[3]
Standard Transformer	55	60	55
+ back translation (1:0.5)	56	56	54
+ back translation (1:1)	56	58	55
+ back translation (1:2)	57	58	54
+ back translation (1:4)	56	61	54
BERT-fused model	**60**	**63**	**56**

[1] lexical ambiguity [2] contextless syntactic ambiguity
[3] contextual syntactic ambiguity

Table 14: Accuracy for commonsense reasoning test.

Bi-Granularity Contrastive Learning for Post-Training in Few-Shot Scene

Ruikun Luo, Guanhuan Huang, Xiaojun Quan[*]
School of Computer Science and Engineering, Sun Yat-sen University, China
`{luork, huanggh25}@mail2.sysu.edu.cn, quanxj3@mail.sysu.edu.cn`

Abstract

The major paradigm of applying a pre-trained language model to downstream tasks is to fine-tune it on labeled task data, which often suffers instability and low performance when the labeled examples are scarce. One way to alleviate this problem is to apply post-training on unlabeled task data before fine-tuning, adapting the pre-trained model to target domains by contrastive learning that considers either token-level or sequence-level similarity. Inspired by the success of sequence masking, we argue that both token-level and sequence-level similarities can be captured with a pair of masked sequences. Therefore, we propose complementary random masking (CRM) to generate a pair of masked sequences from an input sequence for sequence-level contrastive learning and then develop contrastive masked language modeling (CMLM) for post-training to integrate both token-level and sequence-level contrastive learnings. Empirical results show that CMLM surpasses several recent post-training methods in few-shot settings without the need for data augmentation.

1 Introduction

The past few years have seen the rapid proliferation of large-scale pre-trained language models such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b). These models are generally characterized by pre-training on huge general-domain corpora and then fine-tuning on task-specific labeled examples when applied to a downstream task. Despite tremendous knowledge obtained from general-domain corpora through pre-training, sufficient labeled examples from the task domain are still needed. However, collecting them is often infeasible in many scenes, where it tends to be unstable and low-performing when directly fine-tuning these pre-trained models (Zhang et al., 2021; Dodge et al., 2020).

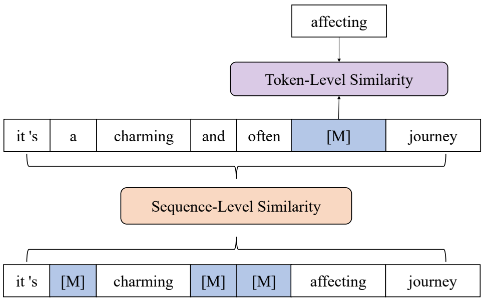

Figure 1: Demonstration of token-level similarity and sequence-level similarity, where the token “affecting” is close to the token “[MASK]” due to the latter’s context. The two sequences with different maskings are also close due to their semantic affinity.

Many efforts have been devoted to addressing the above issue. Firstly, it can be relieved by improving the fine-tuning process, such as introducing regularization (Jiang et al., 2020; Lee et al., 2020), re-initializing top layers (Zhang et al., 2021), and using debiased Adam optimizer (Mosbach et al., 2021). Besides, according to empirical results (Zhang et al., 2021; Mosbach et al., 2021), fine-tuning with a small learning rate and more fine-tuning epochs can also improve the situation. Secondly, additional data can be explored, for which two main genres of data might be helpful: labeled examples from related tasks and unlabeled task examples. The former has shown considerable success on the GLUE tasks (Phang et al., 2018; Liu et al., 2019a), whereas such labeled examples are not always easy to collect. By contrast, the latter is more feasible, especially in scenes where task examples are easy to collect but expensive to label.

Contrastive learning (Hadsell et al., 2006) is a recently re-emerged method for leveraging unlabeled data to enhance representation learning. The key to contrastive learning is to capture the similarity be-

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1733–1742
August 1–6, 2021.

tween samples. As shown in Figure 1, there are two sorts of similarities that can be captured for a natural language sequence: token-level similarity and sequence-level similarity. Masked language model (MLM) (Devlin et al., 2019), widely adopted in pre-trained language models, can be considered as token-level contrastive learning, as it maximizes the similarity between the "[MASK]" token and the original token before masking and minimizes the similarity with other tokens. As for sequence-level similarity, several works (Iter et al., 2020; Giorgi et al., 2020; Wu et al., 2020) introduce sequence-level contrastive learning into pre-training. While all these works focus on the pre-training phase, the focal point of this paper is to improve the performance of pre-trained models in downstream tasks through *post-training* especially for scenes where limited labeled data is available.

Speaking of pre-trained language models, Xu et al. (2019) and Gururangan et al. (2020) demonstrate improvement in various downstream tasks by training the models with MLM on task examples before fine-tuning, which, following Xu et al. (2019), is termed *post-training* in this paper even though Gururangan et al. (2020) term it *adaptive pre-training*. Fang and Xie (2020) also post-train their model on task examples by contrastive self-supervised learning (Chen et al., 2020), which pulls together two augmented sentences generated from the same sentence by back-translation (Edunov et al., 2018) while separating those otherwise. However, these works consider either token-level or sequence-level contrastive learning, without integrating them. Moreover, adopting back-translation to generate augmented sentences demands considerable computation and makes the effect of post-training dependent on the translation systems.

To capture both sequence-level and token-level similarities, we propose contrastive masked language modeling (CMLM) to achieve more effective knowledge transfer in post-training and to improve the performance of pre-trained language models in few-shot downstream tasks. For this purpose, we propose complementary random masking (CRM) to generate a pair of masked sequences from a single sequence for both sequence-level and token-level contrastive learnings. We conduct extensive experiments to compare CMLM with several recent post-training approaches, and the empirical results show that CMLM achieves superior or competitive performance in a wide range of downstream tasks.

Our contributions can be concluded as follows. First, we propose a new random masking strategy, CRM, to generate a pair of masked sequences favorable to sentence-level contrastive learning. Second, we propose a novel objective, CMLM, for post-training pre-trained language models, which realizes both sequence-level and token-level contrastive learnings on a pair of masked sequences. Third, we compare our approach with several post-training methods and obtain superior or competitive results in few-shot settings. Lastly, we compare two contrastive learning implementations, SimCLR (Chen et al., 2020) and SimSiam (Chen and He, 2020), in pre-trained language models. To our best knowledge, our work is the first effort to implement SimSiam in NLP and compare it with SimCLR.

2 Related Works

2.1 Pre-trained Language Model

Pre-trained language models such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) have become a new paradigm of NLP research, and been successfully applied in a wide range of tasks that used to be thorny. These models are generally structured with stacks of Transformer (Vaswani et al., 2017) and pre-trained on large-scale unlabeled corpora. Among them, GPT is pre-trained with a unidirectional language modeling objective, and BERT is with masked language modeling (MLM) and next sentence prediction (NSP). In RoBERTa, Liu et al. (2019b) turn the static MLM in BERT into a dynamic one and remove the NSP task, and pre-train it more intensively with larger corpora.

Despite tremendous knowledge learned from unlabeled corpora during pre-training, sufficient labeled task examples are still needed for fine-tuning, which can be challenging for some scenes. Plus, unlabeled task examples are not leveraged when sticking to the pre-training and fine-tuning paradigm.

2.2 Contrastive Learning

To take advantage of unlabeled or labeled data more effectively, contrastive learning (Hadsell et al., 2006) is re-emerged recently in computer vision (CV) and natural language processing (NLP). The key to contrastive learning is to pull positive samples together while separating negative samples apart. The construction of positive and negative sample pairs varies from tasks to tasks. In CV, Chen et al. (2020) take the augmented (e.g., by random crop, color distortion, and Gaussian blur) images originated from the same image as positive pairs, and treat those otherwise as negative pairs. Khosla et al. (2020) take the images with the same label as positive pairs and take others as negative. In NLP, MLM in BERT can be viewed as contrastive learning on token level, which takes "[MASK]" and its original token before masking as a positive pair and the rest as negative. For sequence level, both Fang and Xie (2020) and Wu et al. (2020) follow Chen et al. (2020) to construct the sample pairs. Specifically, Fang and Xie (2020) utilize back-translation for sequence augmentation while Wu et al. (2020) use some easy deformation operations like deletion, reordering and synonym substitution. Besides, Giorgi et al. (2020) construct two spans as a positive pair if they overlap, subsume, or are adjacent. Gunel et al. (2021) introduce supervised contrastive learning proposed by Khosla et al. (2020) into NLP and treat the sequences with the same label as a positive pair. Li et al. (2021) augment the cross-entropy loss with a contrastive self-supervised learning term and a mutual information maximization term to deal with the cross-domain sentiment classification task.

2.3 Post-training

Post-training has been broadly applied in downstream tasks. For examples, Xu et al. (2019) post-train BERT with MLM on task examples to improve the sentiment analysis task. Gururangan et al. (2020) further divide post-training into two categories: domain-adaptive pre-training and task-adaptive pre-training, and evaluate them by extensive experiments. Phang et al. (2018) fine-tune their model on a related task before fine-tuning on the target task. Liu et al. (2019a) extend previous work into a multi-task learning fashion. Fang and Xie (2020) introduce contrastive self-supervised learning (CSSL) (Chen et al., 2020) to perform post-training and name it CSSL Pre-training.

3 Approach

3.1 Dynamic Random Masking

Masked language modeling (MLM) is firstly applied in BERT (Devlin et al., 2019), where some of the tokens in the input sequence are selected and replaced by a special token "[MASK]". BERT uniformly selects 15% of the input tokens for replacement, and among the selected tokens, 80% are replaced with "[MASK]", 10% are left unchanged, and 10% are replaced by a randomly selected vocabulary token. In the original implementation of BERT, random masking and replacement are performed once in the beginning, and the sequences are kept unchanged through pre-training. Liu et al. (2019b) transform this static masking strategy into dynamic random masking (DRM) by generating a masking pattern every time a sequence is fed. That is to say, given an input sequence $T = \{t_1, t_2, ..., t_N\}$, the probability of each token being selected is determined by p_m, which is fixed to 15% in BERT and RoBERTa.

$$P_{DRM}(t_n) = p_m, \; n \in [1, N] \tag{1}$$

3.2 Complementary Random Masking

It is straightforward to come up with an idea that generates a pair of masked sequences from a single sequence by random masking and applies MLM on each masked sequence to capture token-level similarity, and then perform sequence-level contrastive learning between the two sequences to capture sequence-level similarity. However, it faces a dilemma when applying this idea: setting a small p_m would make the pair of masked sequences too similar and make the contrastive learning loss drop to 0 quickly, harming sequence-level contrastive learning. On the other hand, setting a large p_m would make each masked sequence collapsed, making it hard for the model to recover the original tokens from "[MASK]" based on the context, which in turn harms token-level contrastive learning.

To address this issue, we decouple the pair of masked sequences, denoted by T^0 and T^1, and assign them different masking probabilities p_m and p_c. Specifically, we obtain T^0 with a small masking probability p_m and T^1 with a larger probability p_c. Moreover, to avoid a single word being masked by both sequences, which cripples their relevance, we propose complementary random masking (CRM) to generate a pair of complementary masked sequences which maintain a complementary relationship. Concretely, in CRM, we first generate $T^0 = \{t_1^0, t_2^0, ..., t_N^0\}$ with DRM (Liu et al., 2019b) from the original sequence T, and generate T^1 with an extra constraint: the masking probability $P_{CRM}(t_n)$ will be set to p_c if and only if t_n^0 has not been selected in T^0. Otherwise, it will be set to 0. The process of CRM is described in Figure 2.

$$P_{CRM}(t_n) = \begin{cases} p_c, & t_n^0 \text{ was not selected in } T^0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

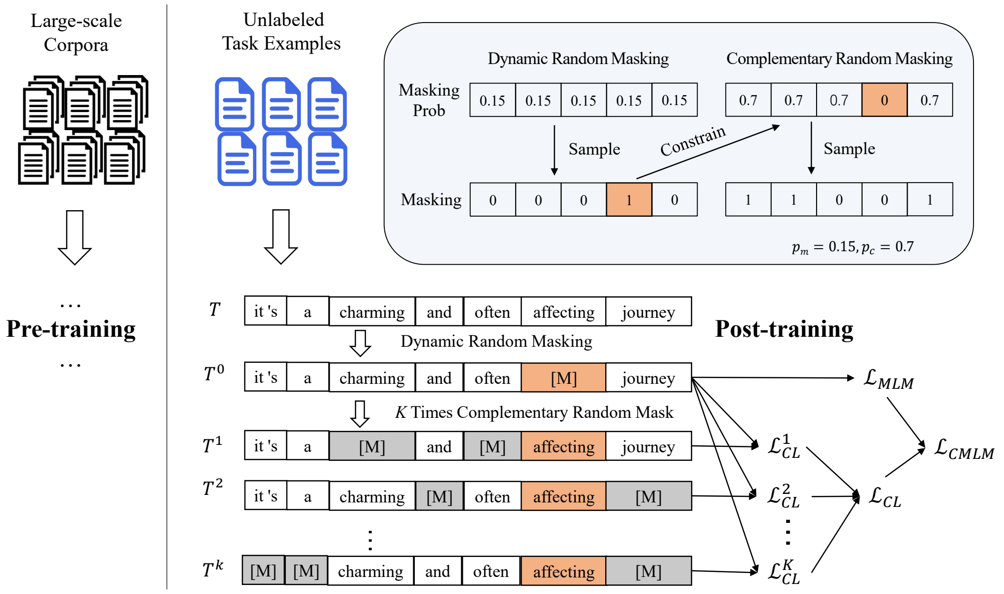

Figure 2: Demonstration of complementary random masking (CRM, in the blue box) and contrastive masked language modeling (CMLM).

CRM is aimed to generate a complementary pair of masked sequences: If $p_c = 1$, all tokens that are not selected in T^0 will be selected in T^1. Reducing p_c can soften this complementary relationship and make the two sequences overlap increasingly.

3.3 Contrastive Masked Language Modeling

We propose contrastive masked language modeling, CMLM, based on CRM to realize domain transfer by masked language modeling (MLM) and sequence-level contrastive learning (CL) with pairs of masked sequences. The framework of CMLM is shown in Figure 2, which is described as follows.

Given a batch $\mathcal{T} = \{T_1, T_2, ..., T_B\}$ of input sequences, we firstly apply dynamic random masking (DRM) on each sequence T_b to generate a masked sequence T_b^0, and then apply CRM K times to generate $T_b^1, T_b^2, ..., T_b^K$ based on T_b^0 and T_b:

$$T_b^0 = \mathrm{DRM}(T_b),\ b \in [1, B] \tag{3}$$

$$T_b^k = \mathrm{CRM}(T_b, T_b^0),\ k \in [1, K] \tag{4}$$

After obtaining $K+1$ masked sequences from each sequence T_b in $\mathcal{T}$, we then compute their representations $H_b^k \in \mathbb{R}^{N\times d}$ by using an encoder, where d is the hidden size of the encoder:

$$H_b^k = \mathrm{Encoder}(T_b^k),\ k \in [0, K] \tag{5}$$

Even though our approach is model-agnostic, in this paper we focus on the Transformer-based pretrained language model RoBERTa, which is an enhanced version of BERT. Therefore, we employ RoBERTa to implement the $\mathrm{Encoder}(\cdot)$ function.

To capture token-level similarity, We apply MLM on H_b^0 as Devlin et al. (2019) and Liu et al. (2019b), and compute the loss as follows:

$$\mathcal{L}_{MLM} = \frac{1}{B}\sum_{b=1}^{B} \mathrm{MLM}(H_b^0) \tag{6}$$

To capture sequence-level similarity, we apply contrastive learning on each H_b^k and H_b^0, and obtain the loss term $\mathcal{L}_{CL}$. We compare two different implementations of contrastive learning: SimCLR (Chen et al., 2020) and SimSiam (Chen and He, 2020). For SimCLR, $\mathcal{L}_{CL}$ can be calculated as:

$$\mathcal{L}_{CL} = -\frac{1}{K\cdot B}\sum_{k=1}^{K}\sum_{b=1}^{B}\log\frac{e^{\mathrm{sim}(H_b^k, H_b^0)/\tau}}{\sum_{i=1}^{B} e^{\mathrm{sim}(H_i^k, H_b^0)/\tau}} \tag{7}$$

where τ is a temperature parameter.

Following Gunel et al. (2021), we take the first token representation $h_b^k \in R^d$ of H_b^k to calculate the similarity between H_i^k and H_j^0 as follows.

$$\mathrm{sim}(H_i^k, H_j^0) = \frac{h_i^k}{||h_i^k||_2} \cdot \frac{h_j^0}{||h_j^0||_2} \tag{8}$$

SimSiam is similar to SimCLR except without using negative pairs and has a negative loss value. To be consistent with $\mathcal{L}_{MLM}$ and $\mathcal{L}_{CL}$, we define the loss function of SimSiam as follows:

$$\mathcal{L}_{CL}=\frac{1}{K \cdot B}\sum_{k=1}^{K}\sum_{b=1}^{B} e^{-\frac{1}{2}(D(z_b^k,h_b^0)+D(z_b^0,h_b^k))} \quad (9)$$

where z_b^k and $D(z,h)$ are defined as:

$$z_b^k = W_2 \cdot \mathrm{gelu}(W_1 \cdot h_b^k) \quad (10)$$

$$D(z,h) = \mathrm{sim}(z, \mathit{stopgrad}(h)). \quad (11)$$

Here, $W_1, W_2 \in R^{d \times d}$ are learnable parameters, $\mathrm{sim}(\cdot)$ is similar to Equation 8, and $\mathrm{Stopgrad}(\cdot)$ is a stop-gradient operation which is crucial for SimSiam (Chen and He, 2020).

Finally, we combine $\mathcal{L}_{MLM}$ and $\mathcal{L}_{CL}$ for contrastive masking language modeling:

$$\mathcal{L}_{CMLM} = \mathcal{L}_{MLM} + \alpha \cdot \mathcal{L}_{CL} \quad (12)$$

where α is a tunable hyper-parameter.

3.4 Relationship to Existing Approaches

Among existing approaches, the closest one to ours is CSSL Pre-training (Fang and Xie, 2020). We can implement CSSL Pre-training by slightly modifying Equation 3 and 12 to following ones:

$$T_b^k = \mathrm{back}(T_b),\ k \in [0,1] \quad (13)$$

$$\mathcal{L}_{CSSL} = \mathcal{L}_{CL} \quad (14)$$

where $\mathrm{back}(T)$ means back-translation of T. And other equations stay the same. By comparing these equations, we note that CMLM can be considered as: (1) replacing back-translation with CRM, which not only reduces the computational cost but also prevents the model from depending on the translation systems; (2) adding $\mathcal{L}_{MLM}$ to implement token-level contrastive learning, which is shown to be crucial in Section 5.2; (3) easily extending one pair of positive samples to K pairs, which can be attributed to the nature of random masking.

As for the loss term, both Giorgi et al. (2020) and Wu et al. (2020) use similar terms to ours for pre-training: $\mathcal{L}_{total} = \mathcal{L}_{MLM} + \mathcal{L}_{CL}$, where $\mathcal{L}_{MLM}$ captures token-level similarity and $\mathcal{L}_{CL}$ captures sequence-level similarity. The main difference between these methods and ours is that we use differently masked sequences from the same sequence as a positive pair, while Giorgi et al. (2020) use position-related segments (overlapping, adjacent or subsumed) and Wu et al. (2020) use sequences by different deformations as the positive pair.

4 Experiment

4.1 Tasks: GLUE

We evaluate our model on the GLUE benchmark (Wang et al., 2018), which contains 9 natural language understanding tasks that can be divided into three categories: (1) single sentence tasks: CoLA and SST-2; (2) similarity and paraphrase tasks: MRPC, QQP, and STS-B; (3) inference tasks: MNLI, QNLI, RTE, and WNLI. All of them are classification tasks except STS-B, so we eliminate it to focus on the classification tasks. WNLI has a small development set (70 examples) and is also ignored. MNLI contains two evaluation sets. One, denoted as MNLI, is from the same sources as the training set, and the other, denoted as MNLI-MM, is from different sources than the training set.

To simulate few-shot scenes of different degrees, we randomly select 20, 100, and 1000 examples respectively from these tasks as our training sets following recent work (Gunel et al., 2021). For each subset in each task, we sample 5 times with replacement and obtain 15 training sets for each task. As for the development set and test set, we randomly select 500 examples from the original development set as our development set and take the remaining as our test set. Since QQP contains too many examples (40k) in the original development set, we randomly select 2000 from the remaining examples after sampling our development set as our test set. Note that all the 15 training sets in each task share the same development and test sets.

4.2 Model: RoBERTa

As mentioned above, we take RoBERTa to implement our encoder in Equation 5. The base version of RoBERTa, Roberta-base, which contains 12 Transformer blocks with 12 self-attention heads, is employed. All the blocks have the same hidden size 768. The input sequence is either a segment or two segments separated by a special token “[\s]”, while “[s]” is always the first token. We take the implementation and pre-trained weights from Huggingface Transformers library (Wolf et al., 2020).

4.3 Training Details

For the fine-tuning of all approaches to be reported below, unless otherwise specified, we use AdamW (Loshchilov and Hutter, 2019) with a learning rate of 1e-5 and epochs of 350, 100, 10 for subsets sized 20, 100, 1000, respectively. This setting is

	CoLA	SST-2	MNLI	MNLI-MM	QNLI	RTE	MRPC	QQP
Metric	mcc	acc	acc	acc	acc	acc	acc	acc
data size = 20								
FT	0.0751±6.26	0.6604±6.38	0.3578±1.70	0.3652±1.96	0.6163±3.46	0.5281±4.07	**0.6747±2.23**	0.6777±3.67
SCL	**0.1105±8.30**	0.6964±6.24	**0.3684±2.85**	**0.3751±3.41**	0.6191±3.55	0.5082±7.75	0.6631±1.15	**0.6947±2.35**
CSSL	0.0795±4.13	0.6609±5.98	0.3640±2.06	0.3686±2.69	0.6064±3.26	0.5264±7.10	0.6638±1.64	0.6514±2.99
TAPT	0.0860±7.64	0.7326±4.99	0.3616±2.12	0.3689±2.40	0.6146±3.60	0.5437±5.06	0.6552±1.64	0.6584±3.33
CMLM (ours)	0.0902±8.65	**0.7371±5.64**	0.3633±2.15	0.3701±2.57	**0.6231±3.60**	**0.5437±4.57**	0.6586±1.41	0.6541±3.89
data size = 100								
FT	0.2176±7.78	0.8405±2.71	0.4361±2.50	0.4526±2.94	0.6820±2.56	0.5879±4.82	**0.7099±1.72**	**0.7511±1.72**
SCL	0.2467±5.46	0.8455±1.38	0.4499±3.30	0.4627±3.84	0.6765±2.47	0.5835±6.41	0.7063±1.45	0.7461±1.63
CSSL	0.1719±7.90	0.8401±1.71	0.4185±2.98	0.4298±3.55	0.6701±1.89	0.5532±5.10	0.7038±1.58	0.7274±1.85
TAPT	0.2626±6.03	0.8496±2.52	0.4508±2.60	0.4682±2.80	0.6970±1.63	0.6095±6.60	0.6987±1.77	0.7429±2.13
CMLM (ours)	**0.2663±6.97**	**0.8525±1.95**	**0.4530±2.75**	**0.4683±3.00**	**0.6980±1.67**	**0.6147±6.36**	0.6933±1.90	0.7479±2.16
data size = 1000								
FT	0.4216±3.13	0.8996±0.97	0.7048±1.19	0.7168±1.17	0.7681±1.07	0.7472±2.50	0.8223±1.22	**0.7934±0.90**
SCL	0.2758±11.94	0.8991±1.04	0.5020±3.81	0.5089±4.09	0.7449±1.22	0.7100±5.44	0.7157±6.83	0.7853±0.77
CSSL	0.4069±3.53	0.8993±1.38	0.6900±1.37	0.7048±1.39	0.7760±0.97	0.7082±5.30	0.8261±1.70	0.7881±1.00
TAPT	0.4362±3.46	0.9016±0.70	0.7074±1.79	0.7203±1.68	0.7689±0.72	0.7524±4.06	0.8214±1.14	0.7890±0.82
CMLM (ours)	**0.4374±2.06**	**0.9023±0.88**	**0.7110±2.00**	**0.7247±1.84**	**0.7719±0.91**	**0.7610±3.28**	**0.8223±0.82**	0.7891±0.90

Table 1: Results on the GLUE benchmark with 20, 100 and 1000 training examples, respectively, and compared with baseline (FT: (Liu et al., 2019b)) and several recent post-training or contrastive learning methods (SCL (Gunel et al., 2021), CSSL (Fang and Xie, 2020), TAPT (Gururangan et al., 2020)). Unit of standard deviation is 10^{-2}.

based on previous empirical results (Zhang et al., 2021; Mosbach et al., 2021), which show that fine-tuning with a small learning rate and more epochs stabilizes the performance of a model in few-shot scenes. We set the batch size to 16 and dropout rate to 0.1, and save model parameters every 100 update steps and pick the best based on validation.

For post-training of CMLM, we apply AdamW with a learning rate of 1e-5 and epochs of 200, 50, 5 for subsets sized 20, 100, 1000, respectively. For a fair comparison with other approaches, we set K in Equation 3 to 1 and the batch size to 8, where the maximum GPU memory usage is approximately equal to that of fine-tuning. For the implementation of $\mathcal{L}_{CL}$, we choose SimSiam for it consumes less computation. For p_m in Equation 1, we follow (Liu et al., 2019b) and set it to 0.15. We conduct a grid-based search for hyper-parameters with $\alpha \in \{0.01, 0.1, 0.3, 0.5, 0.7, 1\}$ (Equation 12) and $p_c \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ (Equation 2), and find that the combination of $\alpha = 0.5$ and $p_c = 0.7$ performs the best on the development set.

For the baselines to be introduced below, we follow the same fine-tuning and post-training settings as our CMLM, with only several method-specific hyper-parameters unchanged.

4.4 Baseline Approaches

As mentioned in Section 2.2 and 2.3, there have been works trying to add extra loss terms in fine-tuning or to insert a post-training phase in between pre-training and fine-tuning. To make a comprehensive comparison, we employ the following approaches as our baselines: (1) fine-tuning (FT) (Liu et al., 2019b), which directly fine-tunes a model with cross-entropy loss; (2) fine-tuning with SCL (SCL) (Gunel et al., 2021), which fine-tunes a model with cross-entropy loss and supervised contrastive loss; (3) post-training with CSSL (CSSL) (Fang and Xie, 2020), which post-trains a model with contrastive self-supervised learning loss; (4) post-training with MLM (TAPT) (Gururangan et al., 2020), which post-trains a model with MLM loss and is equal to CMLM when $\alpha = 0$. Comparing with recent works (Fang and Xie, 2020; Gunel et al., 2021) that take only the conventional either BERT or RoBERTa as their baseline, we consider a few more baselines to obtain more conclusive results.

4.5 Evaluation Details

In few-shot scenes, the distribution of the training set may deviate from the test set seriously. Gunel et al. (2021) pick the top-3 results from all combinations of training sets and model seeds for each task. Differently, for each data size of 20, 100, and 1000 described in Section 4.1, we train our model with random seeds $\{31, 42, 53\}$ for the 5 training subsets, and calculate the mean and standard deviation of the 15 test results. We assume this is a better way to evaluate the overall effect of our model.

4.6 Few-Shot Results

In Table 1, we report our few-shot results on the GLUE tasks with 20, 100, and 1000 training examples, respectively. Five observations can be made from the table. First, CMLM obtains superior performance on the datasets with 100 and 1000 examples, surpassing the baselines in 13 of 16 tasks. Since we use the same hyper-parameters for these approaches and report the average results over 3 random seeds and 5 randomly sampled training sets, these results are convincing. Second, on the dataset with 20 training examples, CMLM only surpasses the other approaches in 3 of 8 tasks. Training a model with only 20 examples is very unstable, and the test results of the baseline approaches indeed show large deviations across different training sets. Third, we find that post-training with only L_{MLM} (Xu et al., 2019; Gururangan et al., 2020) can achieve competitive results with the baselines, showing the effectiveness of this widely-used approach. Fourth, SCL (Gunel et al., 2021) has extremely poor performance on CoLA, MNLI, and MRPC when the data size is 1000, which is beyond our expectation. In the original paper, the authors of SCL only report the top-3 results from combinations of model seeds and train sets. So we speculate this under-performance might come from the instability of SCL in few-shot settings. Fifth, CSSL (Fang and Xie, 2020) performs even worse than FT when the data size is either 20 or 100 but achieves competitive results when the data size is 1000. CSSL is designed for full-size GLUE tasks and might not be suitable for the few-shot scenes.

4.7 Full-Size Results

To verify whether post-training with CMLM can still achieve desirable results when sufficient labeled examples are available, we conduct experiments on the RTE (2.5k), MRPC (3.7k), CoLA (8.5k), SST-2 (67k), and QNLI (106k) tasks with their full-size training sets. We set the learning rate to 3e-5 for both post-training and fine-tuning and set the epoch to 3. Other hyper-parameters remain the same as in Section 4.3. Experiment results are shown in Table 2, from which we can note that CMLM maintains its superiority on RTE and CoLA but fails on MRPC, SST-2, and QNLI. TAPT performs better on tasks with more training examples, which can be explained by the better generalizability of token-level representation, though it demands more training steps to learn well. Note that CMLM is specifically proposed for few-shot settings, so the experiments in the full-size setting are only to evaluate it from different perspectives and make a comprehensive comparison with baselines.

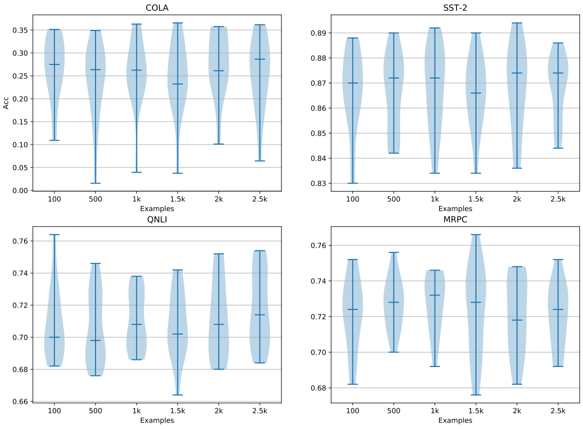

Figure 3: Results of our model on development sets with increasing numbers of unlabeled examples.

	RTE	**MRPC**	**CoLA**	**SST-2**	**QNLI**
metric	acc	acc	mcc	acc	acc
data-size	2.5k	3.7k	8.5k	67k	106k
FT	0.7403	0.8623	0.5552	0.9319	0.9043
SCL	0.6753	0.7393	0.5329	**0.9373**	0.9002
CSSL	0.6623	**0.8713**	0.5217	0.9310	0.8904
TAPT	0.7403	0.8541	0.5519	0.9355	**0.9063**
CMLM (ours)	**0.7446**	0.8574	**0.5714**	0.9310	0.9039

Table 2: Results on the RTE, MRPC, CoLA, SST-2 and QNLI tasks with full-size training sets, and average results over 3 random seeds are reported.

4.8 Additional Unlabeled Examples

We consider the scene where additional unlabeled task examples are provided. We evaluate CMLM on CoLA, SST-2, QNLI, and MRPC with 100 labeled examples for fine-tuning and increase unlabeled examples from 100 to 2500 for post-training. We depict the results on the development sets in Figure 3, from which two observations can be made. First, the performance generally increases with the number of unlabeled examples grows, showing the helpfulness of unlabeled task examples, which is also confirmed by Gururangan et al. (2020). Second, there are certain fluctuations in the results. We assume they come from the random nature of these additional unlabeled examples, which are sampled from a much larger training set and might severely deviate from the original training set. Moreover, we should acknowledge that adding more unlabeled examples in post-training gains limited improvement compared with adding more labeled examples in fine-tuning. Labeled examples are more treasurable in classification learning.

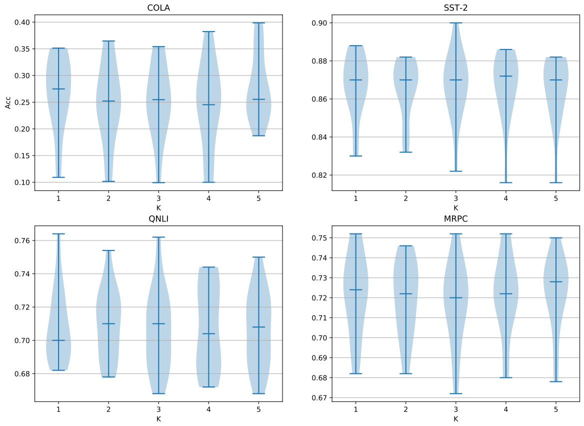

Figure 4: Results of our model on development sets while the parameter K changes.

4.9 Hyper-parameter K

We evaluate whether increasing K (in Equation 7) can lead to improvement of our model in the few-shot setting. We evaluate CMLM on the CoLA, SST-2, QNLI, and MRPC tasks with 100 training examples by increasing K, and the results are depicted in Figure 4. Similar to increasing unlabeled examples, the performance slightly improves on the 5 tasks but has some fluctuations, which is within our expectation. Intuitively, exposing the model to different forms of masked sequences can better reflect the distribution of examples sampled from a large training set, but cannot narrow the deviation between these examples and the original train set.

5 Ablation Studies

5.1 SimCLR vs SimSiam

As described above, $\mathcal{L}_{CL}$ can be implemented by either Equation 7 or Equation 10, although we implement the latter to conduct the above experiments for its less computational cost. According to Chen and He (2020), SimSiam performs better than SimCLR on ImageNet (Deng et al., 2009). It is thus interesting to verify whether the same holds in our situation. We compare SimSiam (CMLM) and SimCLR (w/ SimCLR) on SST-2, CoLA, QNLI and RTE, and the results are reported in Table 3. From the results, we cannot easily conclude which one is better due to their comparable performances, yet further investigation is beyond the scope of this paper. However, we prefer SimSiam due to it consumes less computation and is easier to implement.

	SST-2	**CoLA**	**QNLI**	**RTE**
Metric	acc	mcc	acc	acc
	data size = 100			
CMLM	0.8525	**0.2663**	**0.6980**	0.6147
w/ SimCLR	**0.8586**	0.2511	0.6885	**0.6355**
w/o CL	0.8496	0.2626	0.6980	0.6095
w/o MLM	0.8280	0.2492	0.6873	0.5913
w/o CRM	0.8511	0.2621	0.6920	0.6242
	data size = 1000			
CMLM	0.9023	0.4374	**0.7719**	0.7610
w/ SimCLR	**0.9041**	**0.4446**	0.7696	**0.7732**
w/o CL	0.9016	0.4362	0.7689	0.7524
w/o MLM	0.8927	0.3983	0.7623	0.7039
w/o CRM	0.9013	0.4434	0.7698	0.7654

Table 3: Results of ablation study for CMLM. *w/ SimCLR* means replacing SimSiam with SimCLR, *w/o CL* and *w/o MLM* mean removing $\mathcal{L}_{CL}$ and $\mathcal{L}_{MLM}$ terms from $\mathcal{L}_{CMLM}$, respectively, and *w/o CRM* means replacing CRM with dynamic random masking (DRM).

5.2 Are MLM & CL Critical for CMLM?

One of the improvements of CMLM over previous works is combining $\mathcal{L}_{MLM}$ and $\mathcal{L}_{CL}$ to implement both token-level and sequence-level contrastive learnings. Here, we verify how the bi-granularity contrastive learnings contribute to the performance differently. We remove $\mathcal{L}_{MLM}$ and $\mathcal{L}_{CL}$ alternatively from $\mathcal{L}_{CMLM}$ and evaluate the resulting model on SST-2, CoLA, QNLI, and RTE with 100 and 1000 training examples, respectively. The results are reported in Table 3. As we can see, the results suffer severe deterioration by up to 7.5% after removing $\mathcal{L}_{MLM}$, while removing $\mathcal{L}_{CL}$ only leads to a drop by up to 1.4%. Although both $\mathcal{L}_{MLM}$ and $\mathcal{L}_{CL}$ contribute to the improvement of CMLM, MLM tends to play a more essential role.

5.3 Complementary Random Masking vs Dynamic Random Masking

We propose a complementary random masking (CRM) strategy to generate complementary masked sequences T^k, $k \in [1, K]$, based on T^0, which is generated by dynamic random masking (DRM). Here, we verify whether this complementary nature of T^k benefits contrastive learning. We replace CRM in Equation 3 by DRM, and conduct experiments on SST-2, CoLA, QNLI and RTE with 100 and 1000 training examples, respectively. As shown in Table 3, CRM still surpasses DRM on all 8 tasks, with improvement by up to 1.6%. The superiority of CRM mainly comes from fact that it avoids tokens to be masked in both T^0 and T^k.

6 Conclusion

In this paper, we proposed a novel post-training objective, CMLM, for pre-trained language models in downstream few-shot scenes. CMLM attempts to combine both token-level and sequence-level contrastive learnings for more efficient domain transfer during post-training. For sentence-level contrastive learning, we developed a random masking strategy, CRM, to generate a pair of complementary masked sequences for an input sequence. Empirical results show that post-training with our CMLM outperforms other recent approaches on the GLUE tasks with 100 and 1000 labeled training examples, respectively. We also conducted extensive ablation studies and showed that both token-level and sequence-level contrastive learnings contribute to the results of CMLM, and that CRM achieves favorable sequence-level contrastive learning over the previous masking strategy. In future work, we will further investigate how token-level and sequence-level contrastive learnings affect domain transfer in post-training and explore more effective methods for sequence-level contrastive learning.

Acknowledgments

The paper was supported by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No.2017ZT07X355).

References

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607, Virtual. PMLR.

Xinlei Chen and Kaiming He. 2020. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Hongchao Fang and Pengtao Xie. 2020. CERT: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

John M Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. 2020. DeCLUTR: Deep contrastive learning for unsupervised textual representations. *arXiv preprint arXiv:2006.03659*.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. 2020. Pretraining with contrastive sentence objectives improves discourse performance of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4859–4870, Online. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural*

Information Processing Systems, volume 33, pages 18661–18673. Curran Associates, Inc.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*.

Tian Li, Xiang Chen, Shanghang Zhang, Zhen Dong, and Kurt Keutzer. 2021. Cross-domain sentiment classification with contrastive learning and mutual information maximization. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8203–8207.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training (2018).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. CLEAR: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*.

Fusing Label Embedding into BERT: An Efficient Improvement for Text Classification

Yijin Xiong, Yukun Feng, Hao Wu, Hidetaka Kamigaito, and Manabu Okumura
Institute of Innovative Research, Tokyo Institute of Technology
`{yijinx,yukun,wuhao,kamigaito,oku}@lr.pi.titech.ac.jp`

Abstract

With pre-trained models, such as BERT, gaining more and more attention, plenty of research has been done to further promote their capabilities, from enhancing the experimental procedures (Sun et al., 2019) to improving the mathematical principles. In this paper, we propose a concise method for improving BERT's performance in text classification by utilizing a label embedding technique while keeping almost the same computational cost. Experimental results on six text classification benchmark datasets demonstrate its effectiveness.

1 Introduction

Text classification is a classic problem in natural language processing (NLP). The task is to annotate a predefined class or classes to a given text, where text representation is an important intermediate step.

A variety of neural models have been developed to learn better text representations, including convolution models (Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015; Conneau et al., 2017; Johnson and Zhang, 2017; Zhang et al., 2017; Shen et al., 2018), recurrent models (Liu et al., 2016; Yogatama et al., 2017; Seo et al., 2017; Wang et al., 2018b), and attention mechanisms (Yang et al., 2016; Lin et al., 2017).

Pre-trained models have also been greatly beneficial in text classification in that they help streamline the training process by avoiding a start from zero (Stein et al., 2019; Wang et al., 2017; Jiang et al., 2019). One group of approaches has focused on word embeddings, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014); another has focused on contextualized word embeddings, from CoVe (McCann et al., 2017) to ELMo (Peters et al., 2018), OpenAI GPT (Radford et al., 2018), ULMFiT (Howard and Ruder, 2018), and BERT (Devlin et al., 2019).

BERT has achieved particularly impressive performances across a variety of NLP tasks. With its success, models pre-trained on a large amount of data, such as ERNIE (Zhang et al., 2019), RoBERTa (Liu et al., 2019), UniLM (Dong et al., 2019), and XLnet (Yang et al., 2019), have become popular thanks to their ability in learning contextualized representations. These models are based on the multi-layered bidirectional attention mechanism (Vaswani et al., 2017) and are trained through the masked word prediction task, which are two of the main components of BERT. Continuing to investigate the potential of BERT remains important, since the findings can help with the investigation of variants of BERT as well.

In this work, we propose a simple but effective method to improve BERT's performance in text classification. We enhance the contextual representation learning through encoding the texts of class labels (e.g. "world", "sports", "business", and "science technology" in the AGNews dataset) along with the documents, without changing the original encoder structure. Our main contributions are as follows.

- The embeddings of both texts and labels are jointly learned from the same latent space, and so no further intermediate steps are needed.
- Our implementation takes more thorough and efficient advantage of BERT's inherent self-attention for the interaction between the label embeddings and text embeddings, without introducing other mechanisms.
- Since only the original structure of BERT is required, our method barely increases the amount of computation.
- Extensive results on six benchmark datasets reveal that our method taps into the deeper

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1743–1750
August 1–6, 2021.

potential of BERT, leading to optimism that BERT can be further improved for text classification as well as other downstream tasks.

2 Related Work

Apart from the pre-trained models for learning general language representations mentioned above, some studies have focused specifically on leveraging the representations of classes or the higher level global information. Examples include t-BERT (Peinelt et al., 2020), which combines topic models with BERT for pairwise semantic similarity detection, and LCM (Guo et al., 2020), which generates an enhancement distribution to the one-hot vector representing the classes by calculating the similarity between instances and labels to improve the classification performance.

Moreover, the label embedding has increasingly taken a leading role in related research. It is a technique in which the contents of labels are also embedded, so that the model can be trained to deal with the label information and input features at the same time. It is proven to be effective in various domains including image classification (Akata et al., 2015), multi-modal learning between images and texts (Frome et al., 2013; Kiros et al., 2014), text recognition in images (Rodriguez-Serrano et al., 2013), and zero-shot learning (Palatucci et al., 2009; Yogatama et al., 2015; Li et al., 2015; Ma et al., 2016).

Notably, in the field of text classification, Zhang et al. (2018) converted the task into a vector-matching problem, while Yang et al. (2018) utilized a sequence generation framework for capturing the correlation between labels. Wang et al. (2018a) proposed the label embedding attentive model (LEAM), an attention-based framework that jointly learns the embeddings of words and labels from a shared latent space. Inspired by LEAM, Si et al. (2020) developed LESA-BERT, where label embeddings are incorporated into self-attention by modifying attention scores. Our approach differs from them in that it can consider bidirectional attention between both label and document embeddings in BERT without changing its attention process.

3 Method

3.1 Fusing Label Embedding into BERT

Figure 1 shows the network structure of our model. Inspired by the sentence pair input configuration of BERT, we concatenate texts of labels and an original document to be classified with a [SEP] token as an input, and use different segment embeddings for the label texts and the document text. The actual label texts are listed in Appendix A.

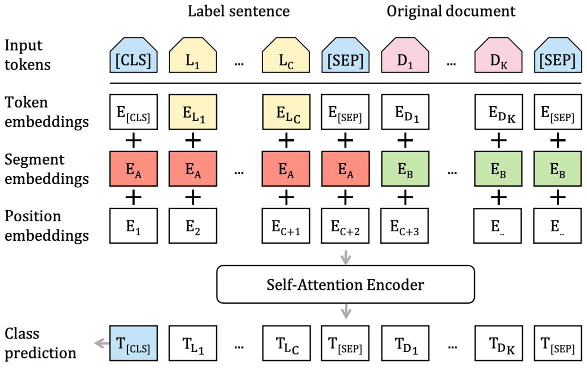

Figure 1: Structure of proposed method.

We denote the document tokens as D_i and their corresponding token embeddings as E_{D_i}. Hence, D_K refers to the last token of the input document, where K is the number of words in the document. Let L_j be the label texts of the j-th class of the total C classes. Since L_j may consist of several subwords, we calculate E_{L_j}, the embedding of L_j, by averaging the token embeddings of all subwords in L_j. In this way, the length of the label sentence is equal to C, and E_{L_j} can be encoded together with E_{D_i} through self-attention. We denote this method as w/ [SEP].

Then, following the same process as the original BERT, we apply a linear layer with the Tanh activation function to the last layer of the hidden-state at the [CLS] token, $T_{[CLS]}$, for making the input of the softmax layer. We use cross-entropy loss for the training.

In addition to the paired input, we examine another setting that concatenates label texts and a document text without utilizing [SEP] or discriminating their segment embeddings. The procedure of computing the token embeddings stays consistent with the paired input setting. We denote this method as w/o [SEP].

3.2 Further Enhancement Using tf-idf

In addition to encoding the original texts of labels into BERT with the document, we experiment with selecting more words as representatives for each class, which expands the number of tokens in L_j. We investigate whether this enhancement can further improve the performance of our models. After tokenizing all the documents under one class in the training set by using the Bert Tokenizer based on WordPiece (Wu et al., 2016), we calculate the average tf-idf score of each subword and add the top 5, 10, 15, or 20 as the supplemental label texts to the corresponding class.

4 Experiments

4.1 Datasets

To evaluate the effectiveness of our method, we performed experiments on six benchmark datasets. As the original benchmarks do not include the development set, we randomly created it from the training set (after removing duplicate samples) for each dataset in accordance with the class distribution of the original test set.

We introduce the original size of each dataset below; see Table 1 for detailed statistics of our training, development, and test sets. Except for IMDb, all the datasets we used were originally constructed by Zhang et al. (2015).

- **AGNews** A news article dataset with titles and descriptions, containing 120,000 training samples and 7600 for testing. Four classes are included: World, Sports, Business, and Science & Technology.

- **DBPedia** An ontology classification over 14 classes, containing 560,000 samples for training and 70,000 for testing.

- **Yahoo! Answers Topic** A dataset containing 1,400,000 training samples and 60,000 testing samples with ten categories. Each sample includes the question title, question content, and best answer.

- **IMDb** (Maas et al., 2011) A binary sentiment classification dataset containing 25,000 highly polar movie reviews for training, and 25,000 for testing. Since its training and test sets are originally of the same size, we merged them together and randomly split it into approximately 8:1:1 for training, development, and testing.

- **Yelp Review Full** A dataset extracted from Yelp Dataset Challenge 2015 data by randomly taking 130,000 training samples and 10,000 testing samples for each starred review from 1 to 5. In total, there are 650,000 training samples and 50,000 testing samples.

- **Yelp Review Polarity** A dataset also extracted from Yelp Dataset Challenge 2015 data but coarsely divided into two classes, considering 1 and 2 stars as negative, and 4 and 5 as positive. In total, there are 560,000 training samples and 38,000 testing samples.

4.2 Settings

For both the baselines (BERT and LESA-BERT) and our proposed methods, we used the pre-trained uncased BERT-base model (Wolf et al., 2019), which consists of 12 Transformer blocks (Vaswani et al., 2017) with 12 self-attention heads and the hidden size of 768. We set the learning rate to 2e-5 and the batch size to 24. The drop-out probability was kept at 0.1. For optimization, we used AdamW (Loshchilov and Hutter, 2018) with epsilon of 1e-8.

The models were trained for five epochs for each benchmark. At the end of each epoch, they were evaluated on the development set, and the ones with the highest accuracy were saved. We report those models' performance on the test set. The training was done for AGNews and DBPedia on 2080Ti and for the rest on Titan RTX. See Table 1 for the maximum sentence length and warm-up steps we assigned for each dataset. We decided the max length based on the average length statistics from Sun et al. (2019) to fully utilize the GPU memory.

Note that we used adjectives "bad, poor, fair, good, excellent", representing the number of stars, instead of numbers 1 to 5 for the basic label texts in the Yelp Review Full dataset, since numbers are used in various unrelated contexts, that may lead to ambiguity.

We fixed the number of top-ranked subwords added for each method on each dataset on the development set. For example, Table 3 shows the averaged results on the AGNews development set for the three methods with top-5, 10, 15, and 20 words added. LESA-BERT (Si et al., 2020), w/ [SEP], and w/o [SEP] all reach the highest accuracy when five words were added, and so this was their final configuration when tested. The comparative experiments were also conducted on the other five datasets (see Appendix B for details).

4.3 Experimental Results

In Table 2, we report the average performance with three different random seeds (see Appendix B for

Dataset	Classes	Type	Train	Dev.	Test	Max length	Warm-up steps
AGNews	4	Topic	112,312	7,600	7,600	230	1,000
DBPedia	14	Topic	489,630	70,000	70,000	230	4,300
Yahoo	10	Topic	1,339,933	60,000	60,000	480	11,900
IMDb	2	Sentiment	39,576	4,800	4,800	480	350
Yelp F.	5	Sentiment	599,960	50,000	50,000	480	5,300
Yelp. P	2	Sentiment	521,985	38,000	38,000	480	4,600

Table 1: Statistics of six benchmarks. In each dataset, the development set is of the same size and class distribution as the test set. Max length indicates the text length without label sentences: the total sentence length for w/ [SEP] would be Max Length + C + 1, where C denotes the number of classes. As for w/o [SEP], the length would be Max Length + C.

Model	AGNews	DBPedia	Yahoo	IMDb	Yelp F.	Yelp P.
BERT	94.456	99.123	75.534	94.667	68.334	97.071
LESA-BERT$^\diamond$	94.522	99.164	75.431	94.743	68.411	97.083
Ours w/ [SEP]	94.557	99.147	75.484	94.931	68.605	97.106
Ours w/o [SEP]	94.653	**99.177***	75.494	94.875	**68.651***	97.155
LESA-BERT$^\diamond$ + tf-idf	94.561 (+5)	99.127 (+10)	75.557 (+15)	94.757 (+20)	68.245 (+10)	97.078 (+15)
Ours w/ [SEP] + tf-idf	94.697 (+5)	99.141 (+10)	75.589 (+15)	94.917 (+ 5)	68.367 (+20)	97.165 (+15)
Ours w/o [SEP] + tf-idf	**94.886***(+5)	99.139 (+20)	**75.628** (+15)	**94.938** (+15)	68.252 (+15)	**97.176** (+15)

Table 2: Model accuracy on the test set, in percentage. $^\diamond$We ran LESA-BERT using the authors' implementation. +tf-idf means top-ranked subwords with average tf-idf scores are added for each class as supplemental label texts, and (+k) denotes their number. **Bold** indicates the best score for each dataset. * means the difference from BERT is statistically significant using paired-bootstrap-resampling test with p<0.05.

No. of words	+5	+10	+15	+20
LESA-BERT	**94.956**	94.903	94.912	94.903
Ours w/ [SEP]	**94.860**	94.812	94.807	94.802
Ours w/o [SEP]	**94.916**	94.785	94.912	94.846

Table 3: Model performance on the AGNews development set with different numbers of supplemental subwords added.

detailed results). We find that fusing only original label texts either with or without [SEP] yielded an improvement over the baselines, except on Yahoo. We assume this is because the original labels are not discriminative enough for big datasets, and so they may corrupt the input rather than enhance it, that leads to the degradation in accuracy.

However, when the top-ranked words were added, the performance on Yahoo was boosted to exceed the baselines. We notice this improvement, caused by adding supplemental words, took place on most benchmarks. Please note that the added words can sometimes contribute to the performance improvement even for the baseline, LESA-BERT.

On the other hand, the performances of all methods dropped drastically on Yelp F.. We assume this is because the top-ranked subwords with averaged tf-idf scores may not be a good representative for the granularity and polarity of emotions, while they can be powerful enough for distinguishing between topics. The enhancement helped IMDb and Yelp P. but not Yelp F., though all are benchmarks for sentiment analysis. In contrast to IMDb and Yelp P., which have only positive and negative labels, Yelp F. has inherent labels, decided by contexts, and so the effect of the tf-idf-based enhancement might be restricted on Yelp F. because the tf-idf score represents only the importance of the words.

Note that w/o [SEP] is better than w/ [SEP] in most cases. The Next Sentence Prediction (NSP) task, used in BERT to learn sentence-level representations, concatenates two natural language sentences with a [SEP] token. On the other hand, when we concatenate a label sequence with an input document, the [SEP] token combines a non-natural language sequence with a natural language sentence. This difference may have caused the skewness between pre-training and fine-tuning in BERT, leading to the performance degradation. Thus, simply adding a label sequence as a prefix, as in the w/o [SEP] method, which provides information gain, could yield a more stabilized improvement.

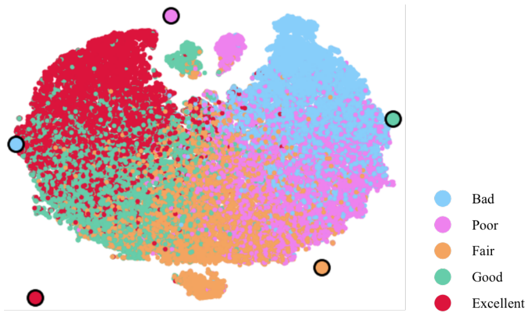

Figure 2: t-SNE visualization of T_{CLS} vectors and averaged T_{L_j} vectors over the Yelp F. test set.

Next, we used t-SNE (Maaten and Hinton, 2008) to visualize the learned representations on a 2-dimensional map, as shown in Figure 2. We visualize the vectors learned from the w/o [SEP] model for the Yelp F. test set. Each color represents a different class. The point clouds are T_{CLS} vectors, and each point corresponds to a test sample. The large dots with black circles are the averaged vectors of T_{L_j}, which is the encoded embedding of each label. Compared with the embedding of [CLS], the label embeddings are more separated in the vector space. This is presumably the reason that the label embeddings can support classification.

5 Conclusion

We proposed a simple but effective method for fusing label embeddings into BERT while utilizing its inherent inputting structure and self-attention mechanism, which leads to having significant improvements on benchmarks of relatively small and medium sizes. The results from the experiments adding subwords with top-ranked average tf-idf scores as supplemental label texts demonstrated that our method can generally improve the performance as expected. As there may be more appropriate methods for constructing enhanced representations, we intend to explore this further in future work. We will also examine different ways of uncovering more potential of pre-trained attentional models like BERT.

Acknowledgements

We thank the anonymous reviewers for their helpful discussion on this work and their valuable comments on the previous draft of the paper.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2015. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129.

Biyang Guo, Songqiao Han, Xiao Han, Hailiang Huang, and Ting Lu. 2020. Label confusion learning to enhance text classification models. *arXiv preprint arXiv:2012.04987*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Ye Jiang, Johann Petrak, Xingyi Song, Kalina Bontcheva, and Diana Maynard. 2019. Team bertha von suttner at semeval-2019 task 4: Hyperpartisan news detection using elmo sentence representation convolutional network. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 840–844.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Xirong Li, Shuai Liao, Weiyu Lan, Xiaoyong Du, and Gang Yang. 2015. Zero-shot image tagging by hierarchical semantic embedding. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 879–882.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in neural information processing systems*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. *Advances in neural information processing systems*, 22:1410–1418.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. 2020. tbert: Topic models and bert joining forces for semantic similarity detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7047–7055.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Jose A Rodriguez-Serrano, Florent Perronnin, and France Meylan. 2013. Label embedding for text recognition. In *BMVC*, pages 5–1.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Neural speed reading via skim-rnn. *arXiv preprint arXiv:1711.02085*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2018. Deconvolutional latent-variable model for text sequence matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Shijing Si, Rui Wang, Jedrek Wosik, Hao Zhang, David Dov, Guoyin Wang, and Lawrence Carin. 2020. Students need more attention: Bert-based attention model for small data with application to automatic patient message triage. In *Machine Learning for Healthcare Conference*, pages 436–456. PMLR.

Roger Alan Stein, Patricia A Jaques, and Joao Francisco Valiati. 2019. An analysis of hierarchical text classification using word embeddings. *Information Sciences*, 471:216–232.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018a. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331, Melbourne, Australia. Association for Computational Linguistics.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, volume 350.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2018b. Topic compositional neural language model. In *International Conference on Artificial Intelligence and Statistics*, pages 356–365. PMLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: sequence generation model for multi-label classification. *arXiv preprint arXiv:1806.04822*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 291–296.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Multi-task label embedding for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4545–4553, Brussels, Belgium. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4169–4179.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Appendix

A. Original Label Texts

See Table 4 for the basic label texts for each dataset. Except for Yelp F., all the texts are provided by the original constructors of the datasets.

B. Detailed Experimental Results

Tables 5 - 15 are the averaged development and detailed test results for each dataset, respectively. **Bold** indicates the best score for each model on the devlopment set and among the models on the test set. * means the difference from BERT is statistically significant using paired-bootstrap-resampling test with p<0.05.

Dataset	Classes	Label Texts
AGNews	4	world, sports, business, science technology
DBPedia	14	company, educational institution, artist, athlete, office holder, mean of transportation, building, natural place, village, animal, plant, album, film, written work
Yahoo	10	society culture, science mathematics, health, education reference, computers internet, sports, business finance, entertainment music, family relationships, politics government
IMDb	2	negative, positive
Yelp F.	5	bad, poor, fair, good, excellent
Yelp P.	2	negative, positive

Table 4: Basic label texts of the six benchmarks.

AGNews	seed 1	seed 2	seed 3	Mean
BERT	94.592	94.421	94.355	94.456
LESA-BERT$^\diamond$	94.671	94.382	94.513	94.522
Ours w/ [SEP]	94.605	94.474	94.592	94.557
Ours w/o [SEP]	94.697	94.645	94.618	94.653
LESA-BERT$^\diamond$ + 5	94.487	94.605	94.592	94.561
Ours w/ [SEP] + 5	**94.947***	94.658	94.487	94.697
Ours w/o [SEP] + 5	94.776	**94.921***	**94.961***	**94.886***

Table 5: Test results of AGNews.

No. of words	+5	+10	+15	+20
LESA-BERT$^\diamond$	99.083	**99.098**	99.086	99.092
Ours w/ [SEP]	99.082	**99.091**	99.085	99.081
Ours w/o [SEP]	99.090	99.081	99.092	**99.094**

Table 6: Averaged dev. results of DBPedia.

DBPedia	seed 1	seed 2	seed 3	Mean
BERT	99.136	99.116	99.117	99.123
LESA-BERT$^\diamond$	99.144	**99.184***	99.164*	99.164
Ours w/ [SEP]	99.149	99.133	99.159*	99.147
Ours w/o [SEP]	**99.179**	99.183*	**99.170***	**99.177***
LESA-BERT$^\diamond$ + 10	99.114	99.127	99.139	99.127
Ours w/ [SEP] + 10	99.103	99.177*	99.144	99.141
Ours w/o [SEP] + 20	99.157	99.110	99.151*	99.139

Table 7: Test results of DBPedia.

No. of words	+5	+10	+15	+20
LESA-BERT$^\diamond$	75.522	75.554	**75.561**	75.546
Ours w/ [SEP]	75.541	75.571	**75.574**	75.554
Ours w/o [SEP]	75.603	75.547	**75.621**	75.566

Table 8: Averaged dev. results of Yahoo.

Yahoo	seed 1	seed 2	seed 3	Mean
BERT	75.637	75.397	75.568	75.534
LESA-BERT$^\diamond$	75.305	75.592*	75.397	75.431
Ours w/ [SEP]	75.470	75.552	75.430	75.484
Ours w/o [SEP]	75.482	75.543	75.458	75.494
LESA-BERT$^\diamond$ + 15	75.717	75.478	75.477	75.557
Ours w/ [SEP] + 15	75.617	**75.658***	75.492	75.589
Ours w/o [SEP] + 15	**75.740**	75.567*	**75.576**	**75.628**

Table 9: Test results of Yahoo.

No. of words	+5	+10	+15	+20
LESA-BERT$^\diamond$	94.604	94.722	94.757	**94.812**
Ours w/ [SEP]	**94.708**	94.695	94.694	94.604
Ours w/o [SEP]	94.646	94.653	**94.910**	94.512

Table 10: Averaged dev. results of IMDb.

IMDb	seed 1	seed 2	seed 3	Mean
BERT	94.708	94.438	94.854	94.667
LESA-BERT$^\diamond$	94.750	94.979*	94.500	94.743
Ours w/ [SEP]	94.583	**95.292***	94.917	94.931
Ours w/o [SEP]	**95.167***	94.646	94.813	94.875
LESA-BERT$^\diamond$ + 20	94.813	94.771	94.688	94.757
Ours w/ [SEP] + 5	95.125*	94.917	94.708	94.917
Ours w/o [SEP] + 15	95.063	94.667	**95.083**	**94.938**

Table 11: Test results of IMDb.

No. of words	+5	+10	+15	+20
LESA-BERT$^\diamond$	68.776	**68.819**	68.780	68.779
Ours w/ [SEP]	68.719	68.734	68.701	**68.777**
Ours w/o [SEP]	68.793	68.733	**68.799**	68.795

Table 12: Averaged dev. results of Yelp F.

Yelp F.	seed 1	seed 2	seed 3	Mean
BERT	68.180	68.432	68.390	68.334
LESA-BERT$^\diamond$	68.570*	68.526	68.136	68.411
Ours w/ [SEP]	68.602*	68.600	68.612	68.605
Ours w/o [SEP]	**68.638***	**68.666**	**68.648***	**68.651***
LESA-BERT$^\diamond$ + 10	68.204	68.264	68.268	68.245
Ours w/ [SEP] + 20	68.300	68.392	68.408	68.367
Ours w/o [SEP] + 15	68.172	68.260	68.324	68.252

Table 13: Test results of Yelp F.

No. of words	+5	+10	+15	+20
LESA-BERT$^\diamond$	97.157	97.164	**97.191**	97.169
Ours w/ [SEP]	97.143	97.181	**97.193**	97.169
Ours w/o [SEP]	97.168	97.151	**97.228**	97.181

Table 14: Averaged dev. results of Yelp P.

Yelp P.	seed 1	seed 2	seed 3	Mean
BERT	97.084	97.037	97.092	97.071
LESA-BERT$^\diamond$	97.155	97.050	97.045	97.083
Ours w/ [SEP]	97.116	97.137	97.066	97.106
Ours w/o [SEP]	**97.179**	97.184*	97.103	97.155
LESA-BERT$^\diamond$ + 15	97.082	97.129	97.024	97.078
Ours w/ [SEP] + 15	97.121	97.179*	**97.195**	97.165
Ours w/o [SEP] + 15	97.153	**97.197***	97.179	**97.176**

Table 15: Test results of Yelp P.

KACC: A Multi-task Benchmark for Knowledge Abstraction, Concretization and Completion

Jie Zhou^{1,2*}**, Shengding Hu**^{1,2*}**, Xin Lv**^{1,2}**, Cheng Yang**^{4}**,**
Zhiyuan Liu^{1,2,3†}**, Wei Xu**^{5}**, Jie Jiang**^{5}**, Juanzi Li**^{1,2}**, Maosong Sun**^{1,2,3}
^{1}Department of Computer Science and Technology ^{2}Institute for Artificial Intelligence
^{3}State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
^{4}School of Computer Science, Beijing University of Posts and Telecommunications, China
^{5}Tencent Marketing Solution, Tencent, Shenzhen, China
`{zhoujie18, hsd20, lv-x18}@mails.tsinghua.edu.cn`

Abstract

A comprehensive knowledge graph (KG) contains an instance-level entity graph and an ontology-level concept graph. The two-view KG provides a testbed for models to "simulate" human's abilities on knowledge abstraction, concretization, and completion (KACC), which are crucial for human to recognize the world and manage learned knowledge. Existing studies mainly focus on partial aspects of KACC. In order to promote thorough analyses for KACC abilities of models, we propose a unified KG benchmark by improving existing benchmarks in terms of dataset scale, task coverage, and difficulty. Specifically, we collect new datasets that contain larger concept graphs, abundant cross-view links as well as dense entity graphs. Based on the datasets, we propose novel tasks such as multi-hop knowledge abstraction (MKA), multi-hop knowledge concretization (MKC) and then design a comprehensive benchmark. For MKA and MKC tasks, we further annotate multi-hop hierarchical triples as harder samples. The experimental results of existing methods demonstrate the challenges of our benchmark. The resource is available at `https://github.com/thunlp/KACC`.

1 Introduction

Large-scale knowledge graphs (KGs) like Wikidata (Vrandečić and Krötzsch, 2014), DBpedia (Lehmann et al., 2015), and YAGO (Mahdisoltani et al., 2013) usually contain two subgraphs: an instance-level **entity graph** and an ontology-level **concept graph**. The entity graph (a.k.a. the entity view) is composed of entities and relations. It describes factual knowledge such as (*Da Vinci*, `paint`, *Mona Lisa*). The concept graph (a.k.a. the concept view) contains concepts and conceptual relations. It provides abstract and commonsense knowledge like (*painter*, `create`, *painting*). In this paper, we name this kind of two-view KG as the entity-concept KG (EC-KG). In a EC-KG, the relations can be grouped into three categories. The "`subclassOf`" relation forms hierarchical concept structures via triples like (*painter*, `subclassOf`, *artist*). The "`instanceOf`" relation groups entities into concepts, such as (*Da Vinci*, `instanceOf`, *painter*). These two relations are important for testing models' abilities on knowledge abstraction and concretization. Other relations are *logical* relations for testing models' abilities on knowledge completion. An example of the EC-KG is shown in Figure 1.

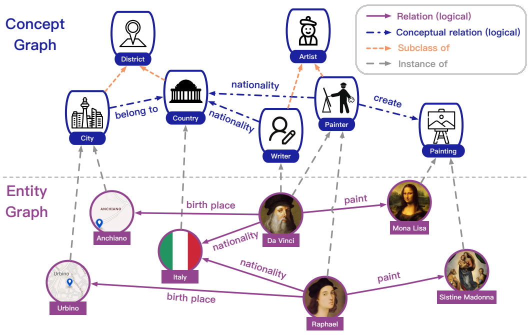

Figure 1: An example of the entity-concept KG.

During the last decade, there are massive works focusing on learning representations for KGs such as TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and TuckER (Balažević et al., 2019). Though they have achieved promising results on knowledge graph completion, most of them focus on a single graph, especially the entity graph.

Beyond modeling a single graph of KGs, recent studies demonstrate that jointly modeling the two graphs in the EC-KG can improve the understanding of each one (Xie et al., 2016; Moon et al., 2017; Lv et al., 2018; Hao et al., 2019). They also propose

---

$*$ indicates equal contribution
$\dagger$ Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1751–1763
August 1–6, 2021.

several tasks on the EC-KG, such as link prediction and entity typing. These tasks focus on partial aspects of knowledge abstraction, concretization, and completion, which are essential abilities for humans to recognize the world and acquire knowledge. For example, in entity typing, a model may link the entity "*Da Vinci*" to the concept "*painter*" which reflects the model's abstraction ability. However, little work has been devoted to unified benchmarking and studies on KACC.

In this paper, we present a comprehensive benchmark for KACC by improving existing benchmarks in dataset scale, task coverage, and difficulty.

Dataset scale. We have examined the EC-KGs proposed by previous works such as Hao et al. (2019). Due to the data distribution, the concept graphs are small compared to the entity graphs. Furthermore, the cross-links between the two graphs are also sparse (refer to Section 4.3). These may limit the knowledge transfer between the two graphs. To tackle these problems, we construct several different-scale datasets based on Wikidata (Vrandečić and Krötzsch, 2014) with careful filtering, annotation and refinement. As Wikidata contains more fine-grained concepts, our datasets have large concept graphs, abundant cross-view links, as well as dense entity graphs.

Task coverage. Most previous works focus on partial tasks of KACC. In our benchmark, we define the tasks more comprehensively and categorize these tasks into three classes: knowledge abstraction, concretization, and completion.

Difficulty. We propose two new tasks, including multi-hop knowledge abstraction and multi-hop knowledge concretization, which require models to predict multi-hop "`instanceOf`" and "`subclassOf`" triples that do not exist in KGs but can be inferred via relation transitivity. These tasks are meaningful and important since correctly modeling these triples is necessary for models to truly understand the concept hierarchy. To ensure the quality of these tasks, we annotate corresponding multi-hop datasets. Our experiments show that these tasks are still challenging for existing models.

Based on our benchmark, we conduct extensive experiments for existing baselines and provide thorough analyses. The experiments show that while the methods specifically designed for modeling hierarchies perform better than general KGE models on abstraction and concretization tasks, they are not competitive to some general KGE models on logical relations. Moreover, all methods have drastic performance degradation on multi-hop tasks, and the knowledge transfer between the entity graph and the concept graph is still obscure. Finally, we present useful insights for future model design.

2 Related Work

2.1 Knowledge Graph Datasets

Existing datasets for knowledge graph completion are usually subgraphs of large-scale KGs, such as FB15K, FB15K-237, WN18, WN18RR and CoDEx (Bordes et al., 2013; Toutanova et al., 2015; Dettmers et al., 2018; Safavi and Koutra, 2020). These datasets are all single-view KGs, in which FB15K, FB15K-237, and CoDEx focus on the entity view while WN18 and WN18RR can be regarded as concept view KGs. Several datasets try to link the two views in different ways. Firstly, some datasets provide additional type information to the entity graph, such as FB15K+, FB15K-ET and YAGO43K-ET (Xie et al., 2016; Moon et al., 2017). Secondly, some datasets provide concept hierarchies for the entity graph, such as Probase (Wu et al., 2012) and YAGO39K (Lv et al., 2018). However, they do not provide the full concept graphs with logical relations. Thirdly, some datasets provide the full concept graphs (Hao et al., 2019), but both the scale and the depth of the concept hierarchy are limited. For example, the entity numbers of DB111K-174 (Hao et al., 2019) and our dataset KACC-M are similar, but KACC-M has 38 times more concepts than DB111K-174 (see Table 1).

2.2 Knowledge Embedding Methods

Existing knowledge embedding (KE) methods can be categorized as translation models (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Ji et al., 2015; Sun et al., 2019), tensor factorization based models (Yang et al., 2015; Nickel et al., 2016; Trouillon et al., 2016; Balažević et al., 2019), and neural models (Socher et al., 2013; Dettmers et al., 2018; Nguyen et al., 2018). These methods are typically designed for single-view KGs. Although they can be directly applied to EC-KGs by ignoring different characteristics between entity graphs and concept graphs, they cannot take full advantage of the information in EC-KGs.

Several works (Krompaß et al., 2015; Xie et al., 2016; Ma et al., 2017; Moon et al., 2017) incorporate the type information into KE methods to help the completion of entity graphs. ETE (Moon et al., 2017) further conducts entity typing, which can be seen as a simplified version of our knowledge abstraction task. Though types of entities can be seen as concepts, they omit the concept hierarchy and interactions (conceptual relations) between concepts.

To jointly model the whole EC-KG, TransC (Lv et al., 2018) adopts TransE as the entity graph model and models concepts as spheres that enclosing points of entities. However, it is not flexible enough to model logical relations between concepts. AttH (Chami et al., 2020) further combines hyperbolic embedding methods with KGE methods to simultaneously embed hierarchical and logical relations. JOIE (Hao et al., 2019) uses different training paradigms for training the entity graph, the concept graph, and the cross-view links. It also defines several meaningful tasks on the EC-KG. In this paper, we extend these tasks with several newly proposed tasks, then we categorize and formalize these tasks into a unified benchmark. We also test several KE methods as mentioned above using our benchmarks and analyze their advantages and deficiencies in terms of handling these tasks.

3 Benchmark

In this section, we propose the KACC benchmark with three tasks: knowledge abstraction, knowledge concretization, and knowledge completion.

3.1 Formalizations

We first give formalizations of the EC-KG, then we introduce multi-hop triples used in later tasks.

Formalizations of EC-KG. A EC-KG is a comprehensive KG, which contains two subgraphs and the cross-view links. The entity graph $\mathcal{G}_E = \{\mathcal{E}_E, \mathcal{R}_E, \mathcal{T}_E\}$ is composed of the entity set $\mathcal{E}_E$, relation set $\mathcal{R}_E$, and corresponding triple set $\mathcal{T}_E = \{(h^E, r^E, t^E) \mid h^E, t^E \in \mathcal{E}_E, r^E \in \mathcal{R}_E\}$, where h, r, t represent head, relation, and tail of a triple, respectively. The concept graph $\mathcal{G}_C = \{\mathcal{E}_C, \mathcal{R}_C, \mathcal{T}_C\}$ contains the concept set $\mathcal{E}_C$, conceptual relation set $\mathcal{R}_C$, and triple set $\mathcal{T}_C$. In our settings, $\mathcal{E}_E$ and $\mathcal{E}_C$ are disjoint sets, while $\mathcal{R}_E$ and $\mathcal{R}_C$ may contain some relations in common (see Section 4.3). The cross-view links $\mathcal{T}_S = \{(h^S, r_{\texttt{ins}}, t^S)\}$ connects the two subgraphs, where $h^S \in \mathcal{E}_E, t^S \in \mathcal{E}_C$, and $r_{\texttt{ins}}$ is the "`instanceOf`" relation. Therefore, the EC-KG is $\mathcal{G} = \{\mathcal{E}_{EC}, \mathcal{R}_{EC}, \mathcal{T}_{EC}\}$, where $\mathcal{E}_{EC} = \mathcal{E}_E \cup \mathcal{E}_C$, $\mathcal{R}_{EC} = \mathcal{R}_E \cup \mathcal{R}_C \cup \{r_{\texttt{ins}}\}$, and $\mathcal{T}_{EC} = \mathcal{T}_E \cup \mathcal{T}_C \cup \mathcal{T}_S$.

There are two special relations "`instanceOf`" and "`subclassOf`" that are crucial for knowledge abstraction and concretization. We use "`ins`" and "`sub`" to denote them in the rest of our paper, respectively. The corresponding triples are $\mathcal{T}_{\texttt{ins}} = \mathcal{T}_S$ and $\mathcal{T}_{\texttt{sub}} \subset \mathcal{T}_C$. Other relations are *logical* relations. Their corresponding triples in concept graphs are $\mathcal{T}_{C(\texttt{logic})} = \mathcal{T}_C \backslash \mathcal{T}_{\texttt{sub}}$ and logical triples in the entity graphs are $\mathcal{T}_{E(\texttt{logic})} = \mathcal{T}_E$.

Multi-hop Triples. Hierarchical relations like "`ins`" and "`sub`" should preserve the multi-hop transitivity, which can be explained by two rules:

$$\begin{aligned}(e, \texttt{ins}, c_1) \wedge (c_1, \texttt{sub}, c_2) \wedge ... \wedge \\ (c_{N-1}, \texttt{sub}, c_N) \Rightarrow (e, \texttt{ins}, c_N),\end{aligned} \tag{1}$$

$$\begin{aligned}(c_0, \texttt{sub}, c_1) \wedge (c_1, \texttt{sub}, c_2) \wedge ... \wedge \\ (c_{N-1}, \texttt{sub}, c_N) \Rightarrow (c_0, \texttt{sub}, c_N),\end{aligned} \tag{2}$$

in which $\{c_i | i \geq 1\}$ are defined as the *high-level concept* for e and c_0. These two rules indicate that an entity/concept always belongs to its high-level concepts. With these rules, we can collect multi-hop hierarchical triples like (e, `ins`, c_N) and (c_0, `sub`, c_N) from the train data and use them as harder samples for knowledge abstraction and concretization testing. Corresponding datasets of multi-hop hierarchical triples are denoted as $\mathcal{T}_{\texttt{M-Ins}}$ and $\mathcal{T}_{\texttt{M-Sub}}$.

3.2 Knowledge Abstraction

This task contains tail prediction tasks for one-hop and multi-hop "`ins`" and "`sub`" triples. We use KA (knowledge abstraction) and MKA (multi-hop knowledge abstraction) to denote the tasks.

KA-Ins / KA-Sub: KA-Ins and KA-Sub are tail prediction tasks for "`ins`" triples and "`sub`" triples respectively. These are all triples in the original datasets and these tasks reflect the direct knowledge abstraction ability of models.

MKA-Ins / MKA-Sub: MKA-Ins and MKA-Sub are tail prediction tasks for multi-hop hierarchical triples $\mathcal{T}_{\texttt{M-Ins}}$ and $\mathcal{T}_{\texttt{M-Sub}}$. These tasks reflect models' abilities on high-level concept abstraction, which aim to predict upper concepts multiple hops away in the concept hierarchy.

3.3 Knowledge Concretization

Similar to KA and MKA tasks, this task contains KC (knowledge concretization) and MKC (multi-hop knowledge concretization) tasks.

KC-Ins / KC-Sub: KC-Ins and KC-Sub are head prediction tasks for "`ins`" and "`sub`" triples, which aim to predict entities for concepts or low-level concepts for high-level ones.

MKC-Ins / MKC-Sub: These tasks are head prediction tasks for multi-hop hierarchical triples. These tasks aim to predict entities for concepts or predict finer concepts for coarser concepts that are multi-hops away.

3.4 Knowledge Completion

The knowledge completion task contains the subtasks of entity graph completion (EGC) and concept graph completion (CGC) under two settings. In the "Single" setting, models can only use each single graph to do knowledge graph completion while both the two graphs and the cross-view links are provided in the "Joint" setting.

CGC-Single / EGC-Single: These subtasks are conducted on each single graph $\mathcal{G}_C$ and $\mathcal{G}_E$. The test phase is conducted on logical triples of each graph $\mathcal{T}_E$ and $\mathcal{T}_{C(\texttt{logic})}$. The results can be compared with results from CGC/EGC-Joint to test the effectiveness of jointly modeling the two graphs.

CGC-Joint: This subtask requires the model to do link prediction with the full information of the EC-KG $\mathcal{G}$. The model needs to abstract conceptual knowledge from the entity graph to do link prediction for logical concept graph triples $\mathcal{T}_{C(\texttt{logic})}$. The results of this subtask can also be used to verify models' abilities on knowledge abstraction.

EGC-Joint: Models are required to use the guidance from the concept graph to do link prediction for entity graph triples $\mathcal{T}_E$. For example, a person in the entity graph is more likely to lead some organizations if he is a politician.

4 Dataset Construction

In this section, we first provide the details of our data collection process and annotation process. Then we give a detailed analysis of the statistical characteristics of the datasets.

4.1 Dataset Collection

The dataset construction process has four steps:

Step 1: Entity Filtering. We select entities in FB15K-237 (Toutanova et al., 2015) as our seed entities. We first find out corresponding seed entities in the Wikidata dump via the "`FreebaseID`" property of each item. Note some entities in Freebase may be labeled as concepts in Wikidata, so we filter out these concepts in our seed entity set. Then we extract one-hop neighbors of the seed entities in the entity graph to form the entity pool, which contains more than 10 million entities. With the entity pool, we can sample an arbitrary size of one-hop neighbors to form the entity graph of our dataset. Our sampling strategy is to select entities with highest degrees and the final entity set consists of all seed entities and the sampled one-hop neighbors. To meet the requirements for different scales, we propose three sizes of datasets: (1) KACC-S, the dataset only contains the seed entities; (2) KACC-M, the expected total entity number is set to 100K; (3) KACC-L, the entity number is set to 1M.

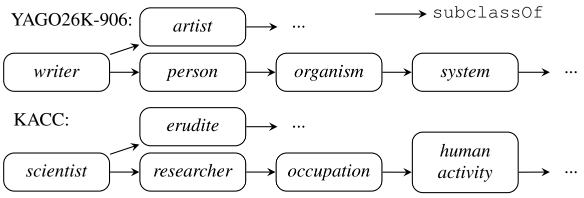

Figure 2: Violation of concept transitivity in two datasets.

Step 2: Concept Finding. Next, we extract concepts based on selected entities. We use a breadth-first search algorithm to find the concepts. The algorithm starts from entities and search for concepts via "`ins`" triples and "`sub`" triples. Since the concept hierarchy follows the structure of a directed acyclic graph, our algorithm ends when all potential concepts are found.

Step 3: Triple Extracting and Filtering. This step firstly extracts cross-view links and all triples in the entity/concept graph. Then we filter all triples by relation statistics and annotation. Relations (1) with less than 10 triples, (2) whose head or tail entity set's size is smaller than 10, and (3) which are annotated meaningless are dropped. Similar to Toutanova et al. (2015), we further remove reverse relations to prevent valid/test leakage.

Step 4: Concept Filtering. To get more precise concept graphs, we ask human annotators to find out meaningless concepts and we further remove these concepts. These "meaningless" concepts include concepts with no labels or descriptions, concepts used for the self-construction of Wikidata (e.g. "*Wikimedia list article*"), etc. Details of the annotation step can be found in Appendix A.1.

4.2 Multi-hop Triple Annotation

To support MKA/MKC tasks, we extract multi-hop "`ins`" and "`sub`" triples from corresponding train sets according to rule (1) and rule (2). Ideally, hierarchical triples should preserve the multi-hop transitivity. However, when we dive into real-world

Dataset	Entity Graph			Concept Graph			# Cross-links
	# Entities	# Relations	# Triples	# Concepts	# Conceptual Rels	# Triples	
YAGO26K-906	26,078	34	390,738	906	30	8,962	9,962
DB111K-174	111,762	305	863,643	174	20	763	99,748
KACC-S	11,896	82	90,722	2,561	18	6,137	16,061
KACC-M	99,615	209	662,650	6,685	30	15,616	123,342
KACC-L	999,148	377	7,741,272	15,160	44	34,930	1,097,970

Table 1: Statistics of different datasets.

datasets like YAGO26K-906 and ours, we find that the multi-hop transitivity does not always hold true. As illustrated in Figure 2, the transitivity is violated when the transition link goes deep. (*scientist*, `sub`, *occupation*) is meaningful while (*scientist*, `sub`, *human activity*) is not. To make our multi-hop triples meaningful, we further ask human annotators to check the validity of these triples. Details are in Appendix A.2.

4.3 Dataset Analysis

In this subsection, we compare our datasets with existing datasets YAGO26K-906 and DB111K-174 (Hao et al., 2019) in terms of scale, domain coverage, and hierarchical relations. The statistics of these datasets are shown in Table 1.

Scale. From Table 1, we can see that concept graphs in our three datasets have more balanced sizes compared to entity graphs. From the comparison between DB111K-174 and KACC, we can see that entity graphs of these two datasets have similar sizes, but KACC has more concepts, conceptual relations, and triples in the concept graph.

Our datasets also have rich cross-view links. In Table 1, the average numbers of cross-links for each entity are less than 1.0 in YAGO26K-906 (0.38) and DB111K-174 (0.89), which means lots of entities in these datasets are not connected to concepts. In KACC, the ratios are all above 1.09, indicating that one entity may belong to multiple concepts.

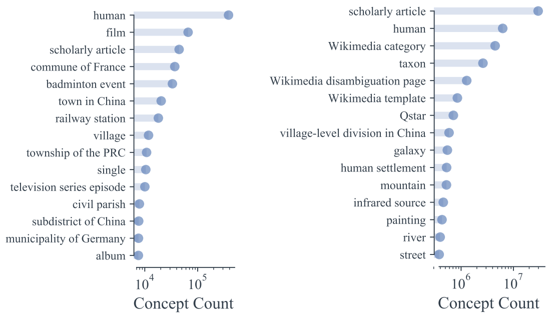

Figure 3: Top 15 most frequent bottom concepts of KACC-L (left) and the original Wikidata dump (right).

Dataset	# Duplicate Edges	# Self-loops	# Undetected Concepts
YAGO26K-906	188	44	132 (17.77%)
DB111K-174	27	13	4 (3.48%)
KACC-S	4 → 0	0	26 (1.02%) → 0
KACC-M	13 → 0	0	33 (0.50%) → 0
KACC-L	17 → 0	0	33 (0.22%) → 0

Table 2: Quality check of existing datasets."→" indicates the filtering process.

Domain Coverage. In Figure 3, we plot 15 most frequent bottom concepts, i.e., the concepts that directly connect to entities, in KACC-L and Wikidata dump to illustrate the domains of our datasets. Plots for KACC-S and KACC-M are in Appendix A.4. We find our datasets mainly focus on people, locations, sports, and films, similar to domains of our seed entities extracted from FB15K-237. The comparisons between our datasets and Wikidata dump show that Wikidata dump contains more domains such as scholarly articles, galaxies, and entities related to Wikimedia. Our datasets only focus on partial domains of Wikidata dump, which ensures entities in our datasets are densely connected.

Hierarchical Relations. We present the characteristics of hierarchical relations in our datasets.

We first examine the data quality of `sub` triples in each dataset. We first detect duplicate edges and self-loops. As the global structure of `sub` relations is assumed to be a directed acyclic graph, we use the topological sort algorithm to find loops. We report numbers of concepts that are not detected by the algorithm in each dataset (these concepts are in loops or dangled in loops). The statistics are in Table 2. We can see that our datasets are of high quality with fewer duplicate edges, no self-loops, and less proportion of concepts in loops. Finally, we remove duplicate edges and wrong-labeled triples in loops after a manual check.

Then we examine the depths of the concepts in each dataset. We start from bottom concepts and traverse all concepts via topological sort. We plot numbers of concepts with different depths in Figure 4. From the figure, we can see that our datasets have deeper hierarchical structures than others, which are more informative and useful for

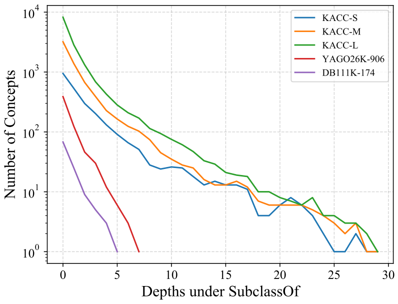

Figure 4: Numbers of concepts with different depths.

Task	Train	Valid	Test
KA-Ins / KC-Ins	$\mathcal{T}_{EC}^{\text{Train}}$	$\mathcal{T}_{\texttt{ins}}^{\text{Valid}}$	$\mathcal{T}_{\texttt{ins}}^{\text{Test}}$
KA-Sub / KC-Sub	$\mathcal{T}_{EC}^{\text{Train}}$	$\mathcal{T}_{\texttt{sub}}^{\text{Valid}}$	$\mathcal{T}_{\texttt{sub}}^{\text{Test}}$
MKA-Ins / MKC-Ins	$\mathcal{T}_{EC}^{\text{Train}}$	$\mathcal{T}_{\texttt{M-Ins}}^{\text{Valid}}$	$\mathcal{T}_{\texttt{M-Ins}}^{\text{Test}}$
MKA-Sub / MKC-Sub	$\mathcal{T}_{EC}^{\text{Train}}$	$\mathcal{T}_{\texttt{M-Sub}}^{\text{Valid}}$	$\mathcal{T}_{\texttt{M-Sub}}^{\text{Test}}$
EGC-Joint	$\mathcal{T}_{EC}^{\text{Train}}$	$\mathcal{T}_{E}^{\text{Valid}}$	$\mathcal{T}_{E}^{\text{Test}}$
CGC-Joint	$\mathcal{T}_{EC}^{\text{Train}}$	$\mathcal{T}_{C(\texttt{logic})}^{\text{Valid}}$	$\mathcal{T}_{C(\texttt{logic})}^{\text{Test}}$
EGC-Single	$\mathcal{T}_{E}^{\text{Train}}$	$\mathcal{T}_{E}^{\text{Valid}}$	$\mathcal{T}_{E}^{\text{Test}}$
CGC-Single	$\mathcal{T}_{C}^{\text{Train}}$	$\mathcal{T}_{C(\texttt{logic})}^{\text{Valid}}$	$\mathcal{T}_{C(\texttt{logic})}^{\text{Test}}$

Table 3: Settings of datasets for different tasks.

models to learn more fine-grained representations.

Finally, we show the characteristic of the "`ins`" relation in our datasets. Unlike existing datasets where "`ins`" only connects entities and concepts, concepts in Wikidata also have "`ins`" connections, which are denoted by $\mathcal{T}_{C(\texttt{ins})}$. We find these triples are also meaningful as they reflect different level semantics. For example in a triple (*planet*, `ins`, *astronomical object type*), "*planet*" is a concept while it can also be regarded as an instance when mentioned in a higher level. The finding is also compatible with human cognition. So we remain these triples in our datasets, and we test them together with other "`ins`" triples. Therefore, we modify the corresponding definitions in Section 3.1 into $\mathcal{T}_{\texttt{ins}} = \mathcal{T}_S \cup \mathcal{T}_{C(\texttt{ins})}$ and $\mathcal{T}_{C(\texttt{logic})} = \mathcal{T}_C \backslash (\mathcal{T}_{\texttt{sub}} \cup \mathcal{T}_{C(\texttt{ins})})$.

Other Characteristics. Our datasets also have some new properties. In existing datasets, relations in the entity graph and conceptual relations in the concept graph are disjoint. However, in our datasets, some relations appear in both the entity graph and the concept graph. For example, the "`partOf`" relation appears in (*Chile*, `partOf`, *South America*) in the entity graph and (*hospital*, `partOf`, *health system*) in the concept graph. Our experiments treat them as different relations while models can also treat these relations as the same, which depends on the hypotheses of the designers.

5 Experimental Settings

5.1 Dataset Partition

Considering the tradeoff between scale and training efficiency, we use the medium-sized dataset KACC-M to conduct the experiments. To generate each task's train/valid/test data, we firstly split each triple set $\mathcal{T}_S, \mathcal{T}_E$ and $\mathcal{T}_C$ by the proportion 8:1:1. To make it easy for model training and hyperparameter selection, we provide a unified train set $\mathcal{T}_{EC}^{\text{Train}}$ for all tasks defined on the EC-KG, that is $\mathcal{T}_{EC}^{\text{Train}} = \mathcal{T}_E^{\text{Train}} \cup \mathcal{T}_C^{\text{Train}} \cup \mathcal{T}_S^{\text{Train}}$. For the valid and test sets, different tasks have their own valid/test sets for model selection and performance reports.

Train sets are different for EGC-Single and CGC-Single. As they focus on a single graph, we use $\mathcal{T}_E^{\text{Train}} / \mathcal{T}_C^{\text{Train}}$ as train sets respectively. The settings of datasets for different tasks are in Table 3. The statistics are in the Appendix A.3.

5.2 Baselines

To test how existing methods behave in our benchmark, we choose several representative models for single-view KG embedding, as well as JOIE (Hao et al., 2019) and AttH (Chami et al., 2020), which are specially designed for modeling the EC-KG.

Single-view KE Methods. We use TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and TuckER (Balažević et al., 2019) as our baselines. These baselines treat the EC-KG as a large single-view KG by regarding concepts as entities, conceptual relations and hierarchical relations as ordinary relations defined on a single-view KG.

JOIE. JOIE (Hao et al., 2019) uses traditional KE methods like TransE and DistMult as the backend model to learn logical relations in entity/concept graphs. It further defines specific transformations and loss functions for hierarchical triples. These mechanisms could improve the performance of corresponding backend models.

AttH. AttH (Chami et al., 2020) utilizes the hyperbolic geometry to embed tree-like structures, which is suitable for modeling the concept hierarchy. It also proposes methods to embed logical relations in the hyperbolic space.

5.3 Evaluation Metrics

We test the tasks in the form of link prediction. We use two evaluation metrics in these tasks:

Mean Reciprocal Rank (MRR). The metric computes the mean reciprocal rank of the correct

Method	KA-Ins			MKA-Ins			KA-Sub			MKA-Sub		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
TransE	0.658	0.560	0.832	0.112	0.047	0.242	0.093	0.000	0.288	0.098	0.035	0.225
DistMult	0.712	0.636	0.847	0.131	0.086	0.222	0.135	0.062	0.277	0.122	0.040	0.284
ComplEx	0.737	0.663	0.863	0.121	0.078	0.214	**0.226**	**0.151**	0.373	0.135	0.066	0.275
TuckER	0.759	0.681	0.885	0.115	0.077	0.179	0.191	0.107	0.369	0.147	0.068	0.313
JOIE	0.706	0.611	0.873	0.195	0.115	0.370	0.099	0.004	0.289	0.113	0.010	0.351
AttH	**0.778**	**0.693**	**0.918**	**0.218**	**0.116**	**0.436**	0.203	0.089	**0.458**	**0.188**	**0.081**	**0.420**

Table 4: Results on knowledge abstraction. Best scores are in **bold**.

Method	KC-Ins			MKC-Ins			KC-Sub			MKC-Sub		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
TransE	0.123	0.083	0.208	0.060	0.033	0.110	0.049	0.000	0.145	0.039	0.000	0.113
DistMult	0.175	0.114	0.282	0.107	0.068	0.183	0.061	0.021	0.136	0.041	0.004	0.128
ComplEx	0.208	0.142	**0.321**	0.098	0.070	0.153	**0.103**	**0.061**	0.179	0.044	0.012	0.107
TuckER	0.169	0.110	0.280	0.074	0.047	0.120	0.087	0.040	0.160	**0.051**	**0.021**	0.106
JOIE	**0.241**	**0.200**	0.320	**0.141**	**0.093**	**0.240**	0.048	0.010	0.120	0.032	0.001	0.092
AttH	0.172	0.120	0.279	0.112	0.061	0.213	0.081	0.021	**0.204**	**0.051**	0.009	**0.135**

Table 5: Results on knowledge concretization. Best scores are in **bold**.

instances. If the ranks of correct instances are k_i, then the metric computes the average of $\frac{1}{k_i}$.

Hits@N. This metric computes the proportion of the ranks that are no larger than N.

A good model could achieve higher scores on these metrics. We use the "Filtered" setting for all the evaluations, which filters out other true answers from the prediction results to get the final rank for each test case.

5.4 Hyperparameter Settings

According to Ruffinelli et al. (2020), performances of KGE methods are sensitive to hyperparameters. Following them, we run 30 quasi-random trails for all models from predefined hyperparameter spaces. We list the hyperparameter spaces we use in Appendix A.5. We run all trails for 100 epochs.

For all single-view KE methods, we use the implementations from LibKGE (Broscheit et al., 2020), which utilizes the Ax framework to perform quasi-random hyperparameter search.

For AttH, we use the implementation from the authors[1]. For JOIE, we use the implementation from the authors[2]. We use TransE as the backend and adopt the suggested hyperparameter space from the paper.

6 Experimental Results

In this section, we provide the experimental results and further propose several future directions.

6.1 Knowledge Abstraction

The results of knowledge abstraction are shown in Table 4. From the results, we can see that AttH has a large margin beyond other methods on KA-Ins and also performs well on KA-Sub, which demonstrates the effectiveness of hyperbolic embeddings. JOIE outperforms its backend model TransE.

Comparing results between KA-Ins and MKA-Ins, all the models have performance degradation larger than 0.51 on MRR. We conclude that the composition rule in Equation (1) is hard to learn naturally. Among all the models, AttH performs the best on both tasks and has the least degradation from KA to MKA, showing that hyperbolic space has advantages over Euclidean space in knowledge abstraction. However, the degradation is still drastic, showing the difficulty of the MKA task.

Comparing results between KA-Sub and MKA-Sub, most methods also have performance degradation while TransE-based models (TransE and JOIE) have better performances on MKA-Sub, which is interesting for further investigation. AttH performs best on MKA-Sub, which further confirms the advantage of hyperbolic methods.

6.2 Knowledge Concretization

The results of knowledge concretization are in Table 5. ComplEx and JOIE performs well on KC-Ins and KC-Sub tasks. Similar to tasks in knowledge abstraction, MKC-Ins and MKC-Sub are also harder for existing models. The results of knowledge concretization tasks are lower than corresponding knowledge abstraction tasks, which shows that knowledge concretization is much harder than knowledge abstraction.

6.3 Knowledge Completion

The results of knowledge completion are shown in Table 6. From the table, TuckER performs well

---

[1] https://github.com/HazyResearch/KGEmb

[2] https://github.com/JunhengH/joie-kdd19

Method	EGC						CGC					
	MRR		Hits@1		Hits@10		MRR		Hits@1		Hits@10	
	Joint	**Single**	**Joint**	**Single**	**Joint**	**Single**	**Joint**	**Single**	**Joint**	**Single**	**Joint**	**Single**
TransE	0.305	0.299	0.182	0.175	0.510	0.504	0.242	0.261	0.003	0.095	0.659	0.603
DistMult	0.444	0.440	0.379	0.376	0.566	0.560	0.495	0.481	0.419	0.436	0.631	0.568
ComplEx	0.458	0.453	0.397	0.393	0.572	0.567	**0.537**	0.501	**0.472**	0.458	**0.684**	0.581
TuckER	**0.481**	**0.473**	**0.415**	**0.408**	**0.604**	**0.595**	0.536	**0.525**	0.468	**0.473**	0.668	**0.615**
JOIE	0.171	-	0.094	-	0.308	-	0.218	-	0.018	-	0.622	-
AttH	0.348	0.352	0.235	0.241	0.551	0.545	0.268	0.244	0.100	0.154	0.631	0.418

Table 6: Results on knowledge completion. Best scores among different models in the same task are in **bold**. Best scores for a model between Joint and Single settings are underlined.

Method	KACC	KA	KCon	KCom
TransE	0.374	0.396	0.143	0.585
DistMult	0.396	0.407	0.181	0.599
ComplEx	0.414	0.429	0.184	0.628
TuckER	0.410	0.423	0.165	**0.636**
JOIE	0.353	0.444	0.177	0.439
AttH	**0.452**	**0.558**	**0.208**	0.591

Table 7: The overall scores. Best scores are in **bold** and second high scores are underlined.

on entity-level logical relations while ComplEx is good at dealing with concept-level logical relations. JOIE does not perform well on logical relations.

From the comparisons between "Joint" and "Single" settings, we find that results on EGC-Joint are usually higher than results on EGC-Single, which shows that incorporating the concept graph and cross-view links helps the understanding of the entity graph. However, the pattern is not obvious on CGC-Joint and CGC-Single, which may due to that entity triples are far more than concept triples, so models tend to focus more on entity triples.

6.4 Overall Results

Finally, we compute an overall KACC score for each method to show their overall performances. Similar to GLUE (Wang et al., 2019a), we average Hits@10 scores of each method on all tasks (except CGC-Single and EGC-Single) to get final scores. We also compute the average scores for knowledge abstraction (KA), knowledge concretization (KCon), and knowledge completion (KCom). In Table 7 we can see that AttH has the best overall score and achieves the highest scores on two tasks. ComplEx also performs well. It is a balanced model since it gets the second place on all tasks. TuckER performs best on knowledge completion. In the future, we plan to test more methods and investigate their abilities.

6.5 Analyses and Future Directions

From the results above, we analyze several problems that existing models cannot handle well and propose several promising future directions.

Multi-hop triple modeling. The prediction scores of multi-hop triples are lower than those of one-hop triples, showing the challenge of multi-hop triple modeling. Besides, how to balance the model to learn from logical and hierarchical relations is also an exciting direction.

Conceptual knowledge completion. Not all models successfully extract conceptual knowledge effectively as their scores of CGC-Joint are lower than those of CGC-Single. The main reason is that KE methods tend to focus more on entity triples due to the losses. They lack the ability to abstract factual knowledge to enrich conceptual knowledge.

Knowledge concretization. The results of concretization tasks are much lower than those of abstraction tasks. It demonstrates that existing models can find proper concepts for entities but cannot find correct entities for concepts. Some solutions may be using contrastive learning to "push" negative entities away from the concepts.

Besides the analyses, there are also several promising future directions of our benchmark.

Contextualized knowledge embedding. Recently, contextualized knowledge embeddings (Wang et al., 2019b) are proposed to capture different semantics of entities and relations in different contexts. These methods only conduct on the entity graph, while incorporating concepts provides more contextual information for entities. For example, an entity of a *painter* is more likely to `paint` than a *politician*. It is a promising direction to model concepts and entities jointly by contextualized embeddings.

Joint modeling EC-KG with text. The EC-KG is a symbolic form of knowledge, and it is interesting to combine it with text. Future directions may include incorporating more commonsense knowledge into the concept graph from language or using the EC-KG to help understand the natural language from the concept level.

7 Conclusion

In this paper, we focus on the problems of knowledge abstraction, concretization, and completion. We propose a benchmark to test the abilities of models on KACC. To conduct the evaluation, we construct large-scale datasets with desired properties, and experiments show that tasks in KACC are challenging. For future work, we plan to test more models and design advanced models to address tasks in KACC.

8 Acknowledgements

This work is funded by the Natural Science Foundation of China (NSFC) and the German Research Foundation (DFG) in Project Crossmodal Learning, NSFC 62061136001 / DFG TRR-169. This work is also supported by Tencent Marketing Solution Rhino-Bird Focused Research Program.

Ethical Considerations

Here we list ethical considerations of our paper:

Intellectual property. All of our datasets are collected from Wikidata and Wikidata offers the data for free with no requirement to attribute under Creative Commons CC0 License.

Privacy. Our datasets are collected from an online resource automatically and the collection process does not involve with participants' privacy rights.

Compensation. For the two annotation processes, the salary for annotating each sample is computed according to the average annotation time and local wage standard. And we ensure that all annotators are well paid.

Potiential problems. Though we have manually checked the quality of our datasets and removed meaningless and wrong data, there still may exist false triples. These may lead to wrong predictions in knowledge abstraction, concretization and completion tasks. However, noises are common in human contributed resources such as existing datasets and ours, so the potiental risks are low.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of EMNLP*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. 2020. LibKGE - A knowledge graph embedding library for reproducible research. In *Proceedings of EMNLP: System Demonstrations*.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of ACL*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. 2019. Universal representationlearning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of KDD*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *Proceedings of ISWC*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. In *Proceedings of EMNLP*.

Shiheng Ma, Jianhui Ding, Weijia Jia, Kun Wang, and Minyi Guo. 2017. Transt: Type-based multiple embedding representations for knowledge graph completion. In *Proceedings of ECML-PKDD*.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *Proceedings of CIDR*.

Changsung Moon, Paul Jones, and Nagiza F Samatova. 2017. Learning entity type embeddings for knowledge graph completion. In *Proceedings of CIKM*.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *Proceedings of ICLR*.

Tara Safavi and Danai Koutra. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of EMNLP*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of ICLR*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of EMNLP*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of ICLR*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019b. Coke: Contextualized knowledge graph embedding. *arXiv preprint arXiv:1911.02168*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of SIGMOD*.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of IJCAI*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of ICLR*.

A Appendices

A.1 Annotations for Meaningless Concepts

In this section, we first present our annotation guidelines for annotators, and then we provide the annotation results.

Task Guidelines. This task aims to find out meaningless "concepts". For a given instance, you need to check whether it is a "concept". Here are some definitions in this task:

- **Concept.** A word for a group or a class of things, such as "artist", "writer", etc. Humans obtain concepts by abstracting commonalities from things.
- **Entity.** A specific person or thing, such as "Barack Obama", "Mona Lisa", etc.

We provide you the Wikidata ID, name, and description of an instance. For more details, you can go to the web "`https://www.wikidata.org/wiki/Qxxxx`" by replacing "Qxxxx" with the specific Wikidata ID. An example is shown in the following:

ID	Name	Description
Q68	computer	general-purpose device for performing arithmetic or logical operations

If an instance is a concept, you should give the correct label. You should give the wrong label in these circumstances:

1. The instance is more like an entity than a concept, such as "Voice over Internet Protocol" (a network protocol).
2. The description and name of the instance are "None".
3. The instance is used for the website's construction and is meaningless, such as "Wikimedia list article" and "Wikimedia disambiguation page".
4. Other cases that are difficult to judge.

Annotation results. We ask human annotators to annotate all concepts in KACC-L. The result of an instance is obtained if two annotators reach the agreement. If not, a third annotator is asked to label the instance. As a result, 482 concepts are removed among 15,642 concepts.

A.2 Annotations for Multi-hop Hierarchical Triples

In this task, we extract multi-hop "`instanceOf`" and "`subclassOf`" transitivity links from different train sets and ask annotators to label the position where the hierarchical transitivity holds.

Task Guidelines. This task aims to annotate the transitivity link of concepts. For an example we provided, you need to determine whether semantic drift exists in this link and label the final position that the transitivity holds. Here is some preliminary knowledge:

- **Concept.** A word for a group or a class of things, such as "artist", "writer", etc. Humans obtain concepts by abstracting commonalities from things.
- **Entity.** A specific person or thing, such as "Barack Obama", "Mona Lisa", etc.
- **Transitivity of concepts.** An example of a transitivity link of concepts is "*scientist* → *researcher* → *occupation* → *human activity*". The transitivity link starts from an entity or a concept and follows by concepts. The transitivity of concepts assumes that the semantic of the later concept could contain the former concept. For example, the semantic of "*occupation*" contains "*scientist*", while "*occupation*" is also a more general meaning concept.
- **Semantic drift.** Because of the annotation process of the original data source (Wikidata), we can assume almost all one-hop links are correct, such as "*scientist* → *researcher*" in our example. But semantic drift occurs as the transitivity link goes deep. For example, "*scientist*" can be subclass of "*occupation*" while it cannot belong to "*human activity*". However, the one-hop link "*occupation* → *human activity*" still holds true.

We provide you the transitivity links of concepts with length 4. These links start from an entity or a concept and is followed by concepts. We provide you the Wikidata ID and the name of the entities and concepts. For more details, you can go to the web "`https://www.wikidata.org/wiki/Qxxxx`" by replacing "Qxxxx" with the specific Wikidata ID. Some examples of this task are shown in Table 8:

Links
(1) Q4442912 capital of Russia → Q5119 capital → Q515 city → Q702492 urban area
(2) Q3108101 tropical garden → Q1107656 garden → Q386724 work → Q15401930 product

Table 8: Annotation examples for concept transitivity.

You need to label the last position that the concept transitivity holds true starting from the first entity/concept. For example (1) in Table 8, "*capital of Russia*" can be regarded as a sub-concept of "*urban area*", so the position is 4. In example (2), "*tropical garden*" belongs to "*garden*" while it does not belong to "*work*", so the position can be labeled as 2.

We can assume that most one-hop links are correct, and you have no need to check the authenticity of them. For example in "*Dewey County* → *county of Oklahoma*", you do not need to check whether *Dewey* is a county of *Oklahoma*. However, in some specific circumstances, the one-hop link may be wrong, then you can label the case as 1, which means that only the first entitiy/concept is true.

If you cannot find out meaningful names for entities or concepts, or you meet other cases that are difficult to judge, you can label them as 0.

Annotation Results. We extract 1200, 3000 and 6000 multi-hop "`instanceOf`" and "`subclassOf`" triples for KACC-S, KACC-M and KACC-L. These numbers are similar to numbers of "`subclassOf`" triples in corresponding valid and test sets. We ask two annotators to annotate them, and a third annotator will be added if the two annotators do not reach an agreement. Note that our task requires to label the position, thus there are cases where all these three annotators give different labels. In these cases, we just omit these examples. If a case is labeled as 4, then we can construct both 2-hop and 3-hop triples from the link. If the case is labeled as 3, we can only obtain the 2-hop triple. The statistics of our datasets are in Table 9.

A.3 Statistics of Dataset Split

The statistics of the datasets after partition are shown in Table 10.

A.4 Additional Domain Plot for KACC

We plot the domains of our KACC-S and KACC-M in Figure 5. Domains of KACC-S and KACC-M are similar while KACC-M has more fine-grained concepts, such as "*town in China*" and "*commune of France*".

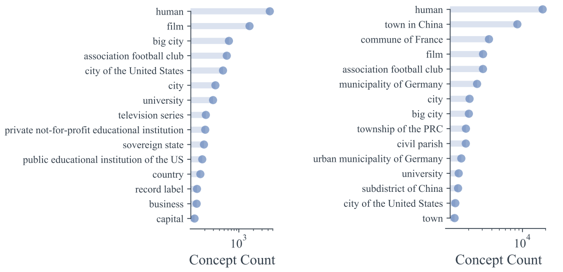

Figure 5: Top 15 most frequent bottom concepts of KACC-S (left) and KACC-M (right).

A.5 Hyperparameter Settings

In this section, we present our hyperparameter selection methods in detail. We run 30 quasi-random hyperparameter search trails on predefined hyperparameter spaces for different baselines (see Table 11 to Table 13). Because we use different implementations, thus hyperparameter spaces are different for different methods.

We run each trail for 100 epochs and save the checkpoint every 20 epochs (150 saved checkpoints for one model in total). Since our benchmark contains multiple tasks, for each task, we use the corresponding valid set to choose the best checkpoint based on the MRR metric, and then we test the selected checkpoint on the test set and compute final metrics.

A.6 Runtime Environment

All experiments are conducted on a server with the following environment.

- Operating System: Ubuntu 18.04.3 LTS
- CPU: Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
- GPU: GeForce RTX 2080 Ti

Dataset	`instanceOf`			`subclassOf`		
	# Extracted Links	# 2-hop Links	# 3-hop Links	# Extracted Links	# 2-hop Links	# 3-hop Links
KACC-S	1,200	1,159	1,137	1,200	1,170	1,148
KACC-M	3,000	2,888	2,854	3,000	2,946	2,904
KACC-L	6,000	5,723	5,671	6,000	5,887	5,806

Table 9: Statistics of annotated multi-hop triples.

Data Source	# Train	# Valid	# Test
$\mathcal{T}_{EC}$	644,332	-	-
$\mathcal{T}_{E}$	533,209	-	-
$\mathcal{T}_{S}$	98,553	-	-
$\mathcal{T}_{C}$	12,570	-	-
$\mathcal{T}_{E}$	533,209	64,965	64,476
$\mathcal{T}_{C}$	-	1495	1549
$\mathcal{T}_{\texttt{sub}}$	-	931	965
$\mathcal{T}_{C(\texttt{logic})}$	-	366	369
$\mathcal{T}_{C(\texttt{ins})}$	-	198	215
$\mathcal{T}_{\texttt{ins}}$	-	12,679	12,523
$\mathcal{T}_{S}$	-	12,481	12,308
$\mathcal{T}_{C(\texttt{ins})}$	-	198	215
$\mathcal{T}_{\texttt{sub}}$	-	931	965
$\mathcal{T}_{C(\texttt{logic})}$	-	366	369
$\mathcal{T}_{\texttt{M-Ins}}$	-	2,871	2,871
$\mathcal{T}_{\texttt{M-Sub}}$	-	2,925	2,925

Table 10: Statistics of data split for KACC-M.

Hyperparameter	Search Range
Training	
Scheme (Complex, Distmult, TuckER)	1vsAll
Scheme (TransE)	Negative Sampling
No. subject samples (TransE)	[1, 1000], log scale
No. object samples (TransE)	[1, 1000], log scale
Batch size	{128, 256, 512}
Loss type	{CE}
Optimizer	{Adam, Adagrad}
Learning rate	[0.001, 0.1], log scale
Learning rate scheduler's patience	[0, 10]
Embedding	
Dimension	{100, 200, 300}
Initialization	
Std. deviation (Normal)	[10^{-4}, 1.0], log scale
Interval (Uniform)	[-1.0, 1.0]
Regularization	{None, L_3, L_2, L_1 }
Entity emb.weight	[10^{-20}, 10^{-1}], log scale
Relation emb.weight	[10^{-20}, 10^{-1}], log scale
Frequency weighting	{True, False}
Dropout	
Entity emb.dropout	$[0, 0.5]$
Relation emb.dropout	$[0, 0.5]$

Table 11: Hyperparameter space of quasi-random search for TransE, DistMult, ComplEx, TuckER.

Hyperparameter	Search Range
Training	
Scheme	Negative Sampling
No. negative samples	{1, 10, 20, 50}
Batch size	{128, 256, 512}
Loss type	{F2, N3}
Optimizer	{Adam, Adagrad}
Learning rate	{0.001, 0.005, 0.01, 0.05, 0.1}
Embedding	
Dimension	{100, 200, 300}

Table 12: Hyperparameter space of quasi-random search for AttH.

Hyperparameter	Search Range
Training	
Backend	{TransE}
Transition method	{CG, CT}
Scheme	Negative Sampling
Batch size	{128, 256, 512}
a_1, a_2	{1.0, 2.5}
m_1, m_2	{0.5, 1.0}
Optimizer	
Learning rate	{0.0005, 0.001, 0.01}
Embedding	
Entity dimension	{50, 100, 200, 300}
Concept dimension	{50, 100, 200, 300}

Table 13: Hyperparameter space of quasi-random search for JOIE.

A Query-Driven Topic Model

Zheng Fang[1], **Yulan He**[1,2] **and Rob Procter**[1,2]
[1]Department of Computer Science, University of Warwick
[2]The Alan Turing Institute, London, UK
`{Z.Fang.4|Yulan.He|Rob.Procter}@warwick.ac.uk`

Abstract

Topic modeling is an unsupervised method for revealing the hidden semantic structure of a corpus. It has been increasingly widely adopted as a tool in the social sciences, including political science, digital humanities and sociological research in general. One desirable property of topic models is to allow users to find topics describing a specific aspect of the corpus. A possible solution is to incorporate domain-specific knowledge into topic modeling, but this requires a specification from domain experts. We propose a novel query-driven topic model that allows users to specify a simple query in words or phrases and return query-related topics, thus avoiding tedious work from domain experts. Our proposed approach is particularly attractive when the user-specified query has a low occurrence in a text corpus, making it difficult for traditional topic models built on word co-occurrence patterns to identify relevant topics. Experimental results demonstrate the effectiveness of our model in comparison with both classical topic models and neural topic models.

1 Introduction

Topic modeling aims to infer topics from a collection of documents, where a topic is a salient pattern of the collection and is represented by a distribution over words. The availability in large volume of new sources of unstructured data, such as social media, has presented a challenge to conventional qualitative research methods in the social sciences and humanities and encouraged the exploration of topic modeling as a potential solution (Melville et al., 2019; Hu et al., 2019; Yao and Wang, 2020). In these studies, topic modeling has been applied to questions centered on interpretation and meaning. By analyzing words distribution of topics learnt, researchers can apply inductive reasoning on specific topics and perform a more in-depth study of related documents, allowing them to identify underlying topical trends and conduct a more thorough analysis of the data.

One limitation of conventional topic modeling approaches in these studies is that they can only learn topics from the whole corpus. However, in some cases, researchers may be interested in topics describing specific concepts or aspects of the corpus. To identify these topics, researchers have to analyze words distribution for all topics, thereby making it very time consuming. Moreover, it could also happen that the target topics may have a very small presence in the data to be detected directly by a topic model. For instance, given a set of posts about health, researchers may wish specifically to analyze the impact of food on health. If the words related to food have a relatively low frequency of occurrence in the posts, then conventional topic models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) may not find any food-related topics at all. This is caused by the phenomenon of higher order co-occurrence in conventional topic models (Heinrich, 2009), which prevents infrequent words being sampled under the correct topic. While an information retrieval method could be used to find relevant documents, identifying key subtopics discussed in these documents will still be a daunting process.

To handle this limitation, weakly-supervised approaches (Andrzejewski and Zhu, 2009; Nikolenko et al., 2017; Chen et al., 2013; Andrzejewski et al., 2011; Yang et al., 2015) have been proposed as a solution and different types of domain-specific, prior knowledge, such as word correlation (Yang et al., 2015), document and word labels have been introduced. By adding these to the unsupervised topic model, a set of topics describing the domain knowledge can be generated. However, this still requires experts to define the domain knowledge, which may not always be feasible. In addition, the

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1764–1777
August 1–6, 2021.

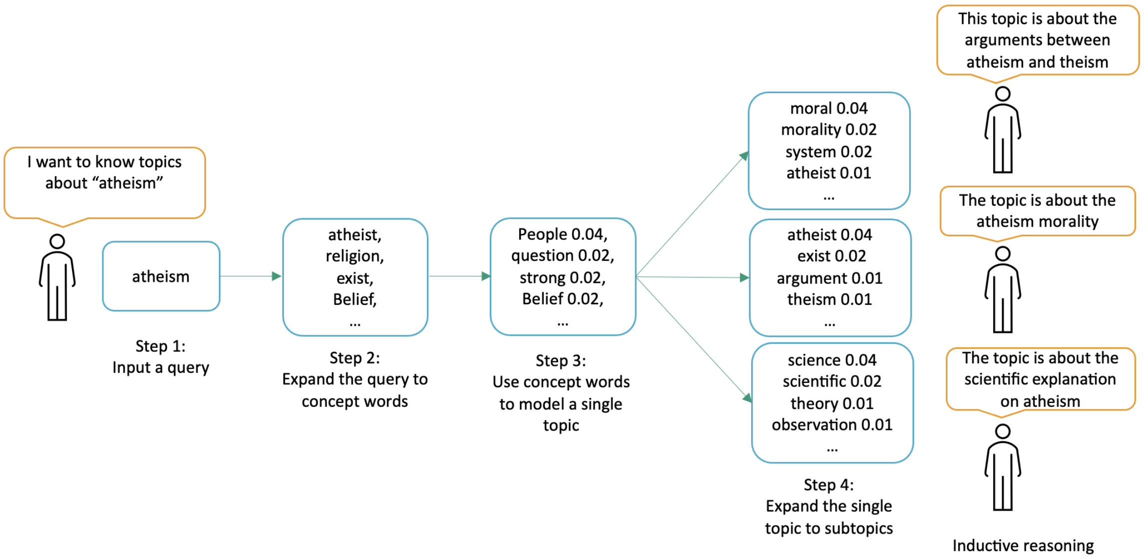

Figure 1: Our proposed model returns topics relevant to a user-input query, in this example, '*atheism*'. **Step 1**: user uses a query to define the concept of interest. **Step 2**: a query expansion technique is used to expand the input query to a set of concept words. **Step 3**: the concept words are utilized to generate a single topic. **Step 4**: the single topic is expanded to a set of subtopics. The retrieved concept-topic and subtopic results allow the user to do inductive reasoning and have a more in-depth study of related documents. In Step 3 and Step 4, we present the top weighted words of the topic and their corresponding weights.

aforementioned approaches can only generate one topic relevant to the target concept. It is desirable to distinguish between different contexts about the same concept: for instance, for the concept '*Middle East*', there might be subtopics relating to Middle East conflicts and Middle East resorts, respectively. In our work, we propose a novel approach that automatically generates all subtopics relevant to the target concept.

In our query-driven topic model, a query phrase is used to define the concept of interest. As illustrated in Figure 1, a query expansion technique is first employed to expand the input query to a set of concept words, which are then utilized to first generate a single topic about the concept, and subsequently further expanded to a set of subtopics automatically. In summary, our contributions are four fold: **(1)** We propose a novel approach which allows users without expertise knowledge to use a short query rather than predefined keywords to detect topics of their interests; **(2)** Our model is novel in its ability to identify rare topics in text, which would not be possible using existing topic modeling approaches; **(3)** Our model is built on the Hierarchical Dirichlet Process (HDP) and can therefore automatically infer all subtopics describing the target concept without having to determine the optimal number of topics beforehand; **(4)** We evaluate our approach on three datasets and achieve superior performance compared to both traditional hierarchical topic models and neural topic models, both quantitatively and qualitatively.[1]

2 Related Work

Earlier work has attempted to solve the problem of identifying specific topics by using prior knowledge. Andrzejewski et al. (2009) expressed domain knowledge with two primitives on word pairs called *Must-Links* and *Cannot-Links*, encoding them using a Dirichlet Forest prior. Topic-in-set knowledge (Andrzejewski and Zhu, 2009) defines '*z-labels*' as prior knowledge and a similar idea was introduced by Nikolenko et al. (2017). First-Order Logic has been proposed as a way to incorporate richer forms of prior knowledge (Andrzejewski et al., 2011). Yang et al. (2015) proposed an efficient method for incorporating domain knowledge and demonstrated significant speed improvement with large datasets. El-Assady et al. (2019) presented a framework that allows users to incorporate the semantics of their domain knowledge in topic models interactively. Gemp et al. (2019) incorporated informative priors in an neural topic model for the purpose of semi-supervised topic modeling. All these approaches

---

[1] Our source code can be accessed at: `https://github.com/Fitz-like-coding/QDTM`.

require experts to provide domain-specific, prior knowledge, which is problematic for two reasons: different corpora in the same domain may contain different information; and it may be costly to specify all prior knowledge. We take advantage of a query expansion technique and propose an automatic concept words extractor to help user extract prior knowledge.

Our work is also similar to the Hierarchical Topic Model (HTM) (Blei et al., 2004). HTM is a non-parametric topic model that generates topics in a hierarchical structure. In our work, we also propose to generate subtopics from a parent topic. A key difference is that we propose a novel solution to incorporate domain-specific prior knowledge, making it possible to generate desirable topics. This is not the case with HTM. Although attempts were made to introduce prior knowledge in HTM, Perotte et al. (2011) focused on out-of-sample label prediction which is not the focus of our work while Xu et al. (2018) still required experts to define word pairs which is problematic as mentioned earlier.

3 Proposed Framework

In outline, our model expands an input query to a set of concept words using a concept words extractor. These concept words are then fed into a two phases framework based on a variant of a Hierarchical Dirichlet Process (HDP) to model all topics relevant to the concept.

3.1 Concept Words Extractor

Given an input query q, we retrieve a list of documents d according to the query likelihood score (Ceri et al., 2013),

$$p(d|q) \approx \prod_{i=1}^{n} p(q_i|d) \tag{1}$$

where n is the number of tokens in the query and $p(q_i|d)$ is the probability of query term q_i in document d. We define two extraction rules "AND" and "OR" to constrain whether query terms should appear in the same document or not. We then extract concept words from the retrieved documents. We adopt three approaches for our purpose.

Frequency based extraction (FRE) The first one simply extracts words with high frequency in the retrieved documents as our concept words:

$$Score(w) = \sum_{i}^{n} TF(w|d_i) \tag{2}$$

where n is the number of retrieved documents and $TF(w|d_i)$ is the term frequency of word w in document d_i.

KL-Divergence based extraction (KLD) The second one is inspired by the query expansion technique (Carpineto et al., 2001). By intuition, words relevant to the input query have a high probability in the retrieved sub-corpus but a low probability in the whole corpus. The score can be defined as:

$$Score(w) = P_R(w) log \frac{P_R(w)}{P_C(w)} \tag{3}$$

where $P_R(w)$ is the probability of word w in the retrieved sub-corpus and $P_C(w)$ is the probability of word w in the whole corpus. We extract words with high scores as our concept words.

Relevance model with word embedding (REL) This approach extracts concept words from a word-embedding enhanced relevance model (Diaz et al., 2016). The probability assigned to word w by the relevance model (Lavrenko and Croft, 2017) is:

$$p(w|RM) = \sum_{d \in R} p(w|d) p(d|q) \tag{4}$$

where R is the retrieved documents set, $p(w|d)$ is the probability of word w in document d and $p(d|q)$ is d's query likelihood from equation (1). We integrate this model with word embeddings:

$$Score(w) = \lambda p(w|RM) + (1 - \lambda)\text{sim}(w, q) \tag{5}$$

where λ is a hyperparameter and $\text{sim}(w, q)$ is the normalized similarity between word w and the input query q. For each term in the vocabulary list, we calculate its similarity with the input query. We then take the top k most similar terms and normalize their similarity values. If w is among the top k similar terms, $\text{sim}(w, q)$ would get the normalized similarity value. Otherwise, $\text{sim}(w, q) = 0$.

3.2 Query-Driven Topic Model

We propose a two-phase framework based on HDP, which is a nonparametric Bayesian model that can automatically infer the number of topics in a corpus (Teh et al., 2005). It assumes a restaurant (i.e., a document) has a set of tables and serves dishes (i.e., topics) from a global menu. A single dish is only served at a single table for all customers (i.e., words) who sit at that table.

In the first phase, the model infers one topic for each concept, along with other irrelevant topics.

We define this topic as the "*parent topic*" in later sections. We denote this parent topic of a concept corresponding to the input query q as $\tilde{z}_q$. We incorporate prior knowledge into HDP by fixing the topic index for concept words in all documents. For words from concept words W_q corresponding to the input query q, the topic index z are known and remain fixed as $\tilde{z}_q$, and the probability for sampling an existing table t for a word w_{ji} at document j and position i in the Gibbs sampling process is:

$$p(t_{ji} = t \mid t^{-ji}, k) \propto \mathbb{1}_1(w_{ji}, k_{jt}) n_{jt}^{-ji} f_{k_{jt}}^{-w_{ji}}(w_{ji}) \tag{6}$$

where k_{jt} is the topic assignment of table t at document j and $f_{k_{jt}}^{-w_{ji}}(w_{ji})$ is the probability of w_{ji} assigned to topic k_{jt} after removing the current word and $\mathbb{1}_1(w_{ji}, k_{jt})$ is an indicator function, which takes on value 0 if $w_{ji} \in W_q$ and $k_{jt} \neq \tilde{z}_q$ and 1 otherwise. n_{jt}^{-ji} denotes the number of words in document j at table t except the current word. The probability for sampling a new table t^{new} is:

$$p(t_{ji} = t^{new} \mid t^{-ji}, k) \propto \alpha p(w_{ji} \mid t^{-ji}, t^{new}, \mathbf{k}) \tag{7}$$

where

$$\begin{aligned} p(w_{ji} \mid t^{-ji}, t^{new}, \mathbf{k}) = \sum_{k=1}^{k} \frac{m_k}{m_{\cdot} + \gamma} f_k^{-w_{ji}}(w_{ji}) \\ + \frac{\gamma}{m_{\cdot} + \gamma} f_{k^{new}}^{-w_{ji}}(w_{ji}) \end{aligned} \tag{8}$$

Here, m_k denotes the number of tables of topic k and $m_{\cdot}$ denotes the total number of tables. γ and α are the hyperparamenters of the model. $f_{k^{new}}^{-w_{ji}}(w_{ji}) = \frac{1}{|V|}$ is the prior density of w_{ji} where $|V|$ is the vocabulary size of the dataset. If the sampled table is a new table, we sample an existing topic $k_{jt^{new}}$ from:

$$p(k_{jt^{new}} \mid t, k^{-jt^{new}}) \propto \mathbb{1}_1(w_{ji}, k_{jt}) m_k f_k^{-w_{ji}}(w_{ji}) \tag{9}$$

and probability for sampling a new topic k^{new} is:

$$p(k_{jt^{new}} = k^{new} \mid t, \mathbf{k}^{-jt^{new}}) \propto \gamma f_{k^{new}}^{-w_{ji}}(w_{ji}) \tag{10}$$

In the second phase, the model expands the parent topic of each concept produced in the first phase to a set of subtopics. Let $W_{\tilde{z}_q}$ be the words assigned to the parent topic $\tilde{z}_q$ in the first phase, the probability for sampling an existing table t for a word w_{ji} in the Gibbs sampling process is:

$$p(t_{ji} = t \mid t^{-ji}, k) \propto \mathbb{1}_2(w_{ji}, k_{jt}) n_{jt}^{-ji} f_{k_{jt}}^{-w_{ji}}(w_{ji}) \tag{11}$$

where $\mathbb{1}_2(w_{ji}, k_{jt})$ is an indicator function that takes on value 1 if $w_{ji} \in W_{\tilde{z}_q}$ and $k_{jt} = \tilde{z}_q$ and 0 otherwise. Probability for sampling a new table t^{new} is:

$$\begin{aligned} p(t_{ji} = t^{new} \mid t^{-ji}, k) \propto \\ \mathbb{1}_2(w_{ji}, k_{jt}) \alpha p(w_{ji} \mid t^{-ji}, t^{new}, \mathbf{k}) \end{aligned} \tag{12}$$

where

$$\begin{aligned} p(w_{ji} \mid t^{-ji}, t^{new}, \mathbf{k}) = \sum_{k=1}^{K} \frac{m_k}{m_{\cdot} + \gamma} f_k^{-w_{ji}}(w_{ji}) \\ + \frac{\gamma}{m_{\cdot} + \gamma} \hat{f}_{k^{new}}^{-w_{ji}}(w_{ji}) \end{aligned} \tag{13}$$

and $\hat{f}_{k^{new}}^{-w_{ji}}(w_{ji}) = \frac{1}{|W_{\tilde{z}_q}|}$ is the prior density of $W_{\tilde{z}_q}$ where $|W_{\tilde{z}_q}|$ is the vocabulary size of $W_{\tilde{z}_q}$. For a new table, probability for sampling an existing topic k is:

$$\begin{aligned} p(k_{jt^{new}} = k \mid t, \mathbf{k}^{-jt^{new}}) \propto \\ \mathbb{1}_2(w_{ji}, k_{jt}) m_k f_k{}^{-w_{ji}}(w_{ji}) \end{aligned} \tag{14}$$

and probability for sampling a new topic k^{new} subordinate to the parent topic $\tilde{z}_q$ is:

$$p(k_{jt^{new}} = k^{new} \mid t, \mathbf{k}^{-jt^{new}}) \propto \gamma \hat{f}_{k^{new}}^{-w_{ji}}(w_{ji}) \tag{15}$$

The model automatically decides the number of subtopics and we treat the subtopics produced as the final topics relevant to the target concept.

Incorporating Generalized Pólya Urn scheme To make topics more interpretable, we incorporate word-embeddings by the Generalized Pólya Urn scheme (Li et al., 2016). Pólya Urn scheme is introduced for colored balls and urns. In the Generalized Pólya Urn scheme, when we draw a ball of a particular color, two balls of the same color are put back along with a certain number of balls of the similar colors. In topic modeling context, a topic can be viewed as an urn while a word can be viewed as a ball in a certain color and its semantically related words can be viewed as balls of similar colors. Every time we sample a word w under a parent topic $\tilde{z}_q$, we increase the probability of sampling w under $\tilde{z}_q$, as well as its semantically related concept words. Given pre-trained word embeddings, we calculate the cosine similarity between word w_i and concept word $w_q \in W_q$. We then construct a word semantic relatedness matrix $\mathcal{M}$ (Li et al., 2016), consisting of all word pairs whose cosine similarity is greater than a predefined threshold. We then construct a promotion matrix A

whose elements are efined as:

$$A_{i,q} = \begin{cases} 1, & \text{if } (w_i, w_q) \in \mathcal{M} \text{ and } w_i = w_q \\ u, & \text{if } (w_i, w_q) \in \mathcal{M} \text{ and } w_i \neq w_q \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

where $u \in (0, 1)$ is a predefined promotion weight. When we sample a word w under topic $\tilde{z}_q$, we also promote all its semantically related concepts words based on the amount of promotion in A.

Word filtering Inspired by Wang et al. (2020), we propose a word filtering strategy. Word filtering can be used to prevent words that have weak ties with the sampled topic being promoted. For a word w at i^{th} Gibbs sampling iteration, its semantic cohesion to topic k is:

$$CV[k, w_i] = \sum_{m=1}^{M} p^i(k, m) \cdot \cos\big(w_i, RW^i(k, m)\big) \tag{17}$$

where $p^i(k, m)$ is the probability of m^{th} representative word in topic k at i^{th} iteration and M is the number of representative words predefined. The representative words of topic $k \neq \tilde{z}_q$ at i^{th} Gibbs sampling iteration are defined by the words ranked by the topic-word probability in the descending order. $\cos\big(w_i, RW^i(k, m)\big)$ is the cosine similarity between word w_i and the m^{th} representative word of topic k at i^{th} iteration. The representative words of $\tilde{z}_q$ are simply its concept words.

For the semantic cohesion of word w with different topics $CV[\cdot, w]$, we map $CV[\cdot, w]$ into an arithmetic progression $\tilde{CV}[\cdot, w]$ ranging from 0 to 1.0 (Wang et al., 2020). We use the following equation to decide if the GPU is applied to w:

$$\begin{gathered} S_{j,w} \sim Bernoulli(\lambda_{w,k_w}) \\ \lambda_{w,k} = \frac{\tilde{CV}[k, w]}{\tilde{CV}_{max}[k, w]} \end{gathered} \tag{18}$$

where $S_{j,w}$ indicates whether GPU is applied to word w given document j and $\tilde{CV}_{max}[k, w]$ is the maximal semantic cohesion among all topics.

We present the details of the Gibbs sampling process of the first phase of our model in Algorithm 1. We omit the details of the second phase of our model since it is similar to the first phase. The details of the functions Initialize($\cdot$) and UpdateCounter($\cdot$) can be found in Appendix C.

Algorithm 1: Query-driven topic model.

```
Input: initial topic number $K$, hyperparameters
       $\alpha, \beta, \gamma$, word semantic relatedness matrix $\mathcal{M}$,
       documents $D$, and Concept words $W_Q$,
Output: The posterior topic-word distribution
Initialize($K$, $D$, $W_Q$);
/* first phase */;
foreach $iteration$ do
    Update word-topic coherence using Eq.17
    foreach document $j \in D$ do
        foreach position $i \in j$ do
            Assign table $t \leftarrow t_{ji}$
            Assign topic $k \leftarrow k_{jt}$
            UpdateCounter($S_{j,w_{ji}}$, $t$, $k$, $False$);
            $t_{ji} \leftarrow t \sim p(t_{ji} = t \mid t^{-ji}, k)$
              (Eq.6-8)
            if $t == t^{new}$ then
                $k_{jt} \leftarrow k \sim p(k_{jt^{new}} = k \mid$
                  $t, k^{-jt^{new}})$ (Eq.9-10)
            end
            $S_{j,w_{ji}} \leftarrow updateGPUFlag(j, w_{ji})$
              based on Eq.18
            UpdateCounter($S_{j,w_{ji}}$, $t_{ji}$, $k_{jt}$, $True$)
        end
    end
end
```

4 Experiments

We conducted our experiments by two steps. In the fist step, we evaluated the quality of the parent topics from the first phase of our model. In the second step, we evaluated the quality of the subtopics from the second phase of our model.

4.1 Setup

Datasets We conducted our experiments on three datasets: 20Newsgroup[2] contains around 18k newsgroup posts on 20 topics; TagMyNews[3] contains around 32k short English news from 7 categories; SearchSnippets (Xu et al., 2017) contains 12k short web search snippets from 8 categories.

Baselines We compared our model with six baselines: LDA (Blei et al., 2003) is a widely used topic model; DF-LDA (Andrzejewski et al., 2009) incorporates domain knowledge in LDA with Must-Links and Cannot-Links; SCLDA (Yang et al., 2015) expresses prior knowledge as sparse constraints; ISLDA (Nikolenko et al., 2017) fixes topic index z for certain keywords in all documents. AVITM (Srivastava and Sutton, 2017) is a neural topic model based on autoencoding Variational Inference. We also compared our model with BERT (Devlin et al., 2018), a well-known neural language

---

[2] http://qwone.com/ jason/20Newsgroups/

[3] http://acube.di.unipi.it/tmn-dataset/

model, to test its document retrieval ability. To evaluate the quality of our subtopics towards the target concepts, we compared our model with HTM (Blei et al., 2004), which is also a nonparametric Bayesian model that can generate subtopics from higher level topics.

Parameterization We set $\alpha = 0.5$, $\beta = 0.1$ for DF-LDA and $\alpha = 0.1$, $\beta = 0.01$ for SCLDA as suggested by the original papers. We set $\alpha = 1/K$, $\beta = 1/K$ for LDA and ISLDA, where K is the number of topics pre-set for the models, and found it outperforms the original settings. We set $\alpha = 0.1$, $\gamma = 0.1$ and $\eta = 0.01$ for HTM as suggested by the original paper and set the topic hierarchy depth to 3, to make it easier to compare with our model since topics from the second level of HTM can be considered as the parent topics and those from the third level as the subtopics of the parent topics. We set $\alpha = 1.0$, $\beta = 0.5$ and $\gamma = 1.5$ as in the original HDP paper for our query-driven model, and set the threshold for the cosine similarity used for the Generalized Pólya Urn scheme to 0.5 and the promotion weight u to 0.3. The number of representative words M for the word filtering strategy was set to 10. λ for the REL query expansion technique was set to 0.5 and k was set to 100. In our experiments, we treated each category as a concept and determined the number of topics for each baseline model based on the number of categories in the datasets. For example, if a dataset had 16 categories, we set the number of topics to 17, using an extra one representing irrelevant information.

For all baseline models, we asked an expert to provide prior knowledge. Each category in a dataset was associated with 10 keywords provided by the expert. For DF-LDA, we converted keywords to must-links. Since LDA, DF-LDA, AVITM and HTM cannot reveal the relationship between a concept and the generated topics directly, we need a further step to find the relationship between them. We calculated the average pairwise cosine similarity of the keywords and the top-10 word embeddings of each topic, and chose the topic with the highest similarity as the target topic of the concept. For HTM, we use the topics from the second level of its generated topic hierarchy. For our model, we used query phrases to represent the main concept of each category. Query phrases were interpreted directly from category names, e.g., we used "*computer graphics*" to represent the category "`comp.graphics`" in the 20Newsgroup dataset. We removed categories that do not have meaningful names due to the difficulty of defining the query phrases for these categories, e.g., "`talk.politics.misc`" in the 20Newsgroup dataset. We then selected the top 10 concept words of each query based on the scores from the concept words extractor. We list expert-defined keywords and query phrase for each category in Appendix A. All models were trained until convergence. For BERT, we simply used the query phrases to retrieve relevant documents. We ran each model five times and present their average performance.

4.2 Parent Topic Evaluation

We evaluated the quality of parent topics of our model in terms of document classification, topic coherence and document retrieval performance. For document classification, a logistic regression classifier with default parameter settings was used. We used the topic distribution of each document as the input and conducted five-fold cross-validation. The topic distribution of a document represents the probability of each topic in a document. The quality of the topics can be assessed by the accuracy of text classification using the topic-level representation. A better classification accuracy means better latent semantic representations of the topics, indicating the learnt topics are more discriminative and representative. For topic coherence measure, we followed Roder et al. (2015) and used the best performing topic coherence measure C_V based on the external corpus (Wikipedia). We focused on the top 10 words of our parent topics and used the Palmetto library algorithm (Röder et al., 2015). Higher coherence indicates better topic interpretability. For document retrieval, we adopted the metric "precision@K" (P@K), which corresponds to the number of relevant results among the top K documents. We retrieved documents of each topic based on the probability of the topic in the documents $p(z|d)$. If a topic can describe the target concept well, then the top retrieved documents should be relevant to the concept. In our experiments, we know the ground-truth number of documents belonging to each category, therefore we set K for each concept to the actual number of documents from the corresponding category. We only considered the parent topics and reported the average results. A higher score indicates the model retrieves more concept-relevant documents, which is important when a researcher wants to do a more

Model	20news			TagMyNews			SearchSnippets		
	Acc	coherence	Precision@K	Acc	coherence	Precision@K	Acc	coherence	Precision@K
LDA	0.650	0.420	0.588	0.781	0.384	0.687	0.804	0.390	0.696
DFLDA	0.623	0.421	0.562	0.772	0.386	0.649	0.795	0.390	0.644
SCLDA	0.666	0.402	0.622	0.804	0.418	0.745	0.816	0.414	0.796
ISLDA	0.680	0.406	0.645	0.801	0.411	0.729	0.845	0.421	0.804
BERT	–	–	0.156	–	–	0.300	–	–	0.261
AVITM	0.504	**0.494**	0.381	0.728	0.482	0.583	0.678	0.461	0.596
Query-driven model(FRE)	**0.707**	0.433	**0.679**	0.807	0.429	0.716	0.858	0.469	0.807
Query-driven model(REL)	0.601	0.452	0.557	**0.837**	0.444	0.749	0.851	0.501	0.780
Query-driven model(KLD)	0.705	0.435	0.677	**0.828**	0.414	**0.755**	0.860	0.465	0.811
+ expert keywords	0.690	0.430	0.659	0.817	**0.493**	0.747	**0.864**	**0.503**	**0.841**
− word filtering	0.706	0.434	0.678	0.823	0.413	0.744	0.859	0.432	0.811
− GPU	0.698	0.400	0.671	0.818	0.389	0.730	0.849	0.413	0.808

Table 1: Parent topic evaluation results for three datasets. FRE indicates frequency based query expansion; REL indicates Word embedding enhanced query expansion; KLD indicates KL-Divergence based query expansion. BERT here is only for document retrieval purpose therefore we can't present the accuracy and coherence in the table.

in-depth study of related documents.

Table 1 shows the performance of our models using three different query expansion techniques as well as only using expert-defined key words as prior knowledge. It can be observed that our models using the FRE and KLD query expansion techniques outperform all baselines except AVITM on almost all measures, though our model using the FRE query expansion technique has slightly worse document retrieval performance on the TagMyNews dataset. Although the model using REL is not as competitive as the models using FRE and KLD on the 20newsgroup dataset, it has the highest coherence scores on all the datasets, despite not using any expert-defined keywords as prior knowledge. This shows that combining word embeddings in query expansion can help produce more coherent prior knowledge. Although AVITM has better coherence score than our model on the 20newsgroup and TagMyNews datasets, its poor document retrieval performance indicates it is unable to find documents relevant to the target concept. Comparing our query-driven model with or without using expert-defined keywords, it achieves better coherence scores on the 20Newsgroup dataset without expert-defined keywords, though it performs slightly worse on the other two datasets.

As for document classification and document retrieval, our KLD-based model has better performance than expert-defined keywords on 20newsgroup and TagMyNews, but does not work well on SearchSnippets. This may be because our concept words extractor does not work well on short texts. The concept words extracted from TagMyNews and SearchSnippets are not as competitive as expert defined keywords. In addition, different interpretations of the same concept word may also compromise the performance. Interestingly, we also observed that BERT does not work well on these datasets. Possibly this is because we are using short query phrases to represent the concepts and BERT only works well for long queries. A short query may not give enough information about the concept that's why we adopted query expansion and topic modeling approaches. Table 2 shows example concept-specific topics extracted from the TagMyNews dataset. It can be seen that extracted topics are closely related to their respective concept phrase. Topic extraction results on the other two datasets are shown in Appendix B.

Concept Phrase	Top 10 words
business	profit, business, bank, sell, usa, sale, credit, price, billion, stock
entertainment	week, show, star, theater, pop, time, film, tv, sony, wedding
health	disease, health, study, care, risk, drug, cancer, insurance, people, usa
technology	google, apple, technology, ipad, china, company, online, intel, service, network
sport game	game, play, playoff, final, boston, win, series, scored, season, sport

Table 2: Target concept topics for TagMyNews dataset.

Ablation study: We also studied the effectiveness of two major components in the proposed model: 1) GPU to incorporate word embeddings;

Model	20news			TagMyNews			SearchSnippets		
	Diversity	Cohesion	overall	Diversity	Cohesion	overall	Diversity	Cohesion	overall
HTM	**0.94**	0.54	0.51	**0.93**	0.53	0.49	**0.86**	0.49	0.42
Query-driven model	0.71	**0.79**	**0.56**	0.68	**0.79**	**0.54**	0.74	**0.76**	**0.56**

Table 3: Topic diversity and cohesion results for subtopics.

2) word filtering to remove unimportant words. The last two rows in Table 1 show the performance of our model using KLD query expansion technique without GPU and word filtering components. These show that GPU has a big impact for coherence and can help improve other measures in some extents, while removing word filtering reduces performance on all measures.

4.3 Subtopics Evaluation

We used our model with the KLD-query expansion technique in this evaluation. We dropped subtopics that have prevalence of less than 0.5% in the corpus as these subtopics usually are not of interest. We evaluated the quality of subtopics in terms of topic diversity and topic cohesion[4]. Topic diversity measures how much a subtopic overlaps with each other. We define it to be the percentage of the unique words in the top 25 words of all subtopics subordinate to the same parent topic (Dieng et al., 2020). Higher diversity indicates more varied topics, while lower diversity indicates more redundant topics. Topic cohesion measures the relevance between the subtopics and the parent topic. We define it to be the cosine similarity between the parent topic embedding and the subtopic embedding. We can get the topic embedding as the weighted summing of the embeddings of its top 10 associated word. We combine these two metrics and define the overall quality of a subtopic as their product.

We report the results in Table 3. It shows that our model outperforms HTM by topic cohesion measures on all datasets, though with lower topic diversity scores. The high cohesion score indicates that the subtopics of our model is highly relevant to the target concept. By taking both measures into a account, our model achieves relatively better performance. It is expected since we incorporate domain prior knowledge into our model.

4.4 Qualitative Evaluation

We present the qualitative evaluation results in this section. We show a set of topics produced by our model from the 20newsgroup dataset in Table 4. The input concept phrases are shown in the left side of the table. For the concept phrase "*atheism*", which means the absence of belief in the existence of deities, we can see that our parent topic is highly relevant to it. The topic words like "*question*", "*belief*", "*god*" and "*lack*" clearly indicate that the topic is related to the arguments about God. By looking at the subtopics, we can easily see the first subtopic is about the atheism morality, the second subtopic is about the arguments between atheism and theism, and the third subtopic is about the scientific explanation on atheism. For the concept phrase "*for sale*", which means selling an item in a cheaper price, our parent topic includes many relevant words, such as "*sell*", "*sale*", "*sold*" and "*price*". The inclusion of "*box*" is less easy to explain, but could be related to product packaging. As expected, our subtopic reveals a sub-aspect about the concept that can not be identified directly from the parent topic: the email subscription for the latest news. This is reasonable, since merchants usually use email to provide customers with information about the latest products. The words like "*interested*", "*mail*", "*send*", "*information*" and "*original*" provide more information about the concept.

The last row of Table 4 presents the topics of the low occurrence query "*business*", appeared only 294 times in the corpus, which is extremely low compared with the majority of other words in the corpus. LDA and HTM are unable to generate relevant topics due to the aforementioned "higher order co-occurrence" issue, but our model can produce reasonable topics. The topic words "*encryption*", "*key*", "*phone*", "*company*" and "*business*" in the parent topic shows that the topic is related to data encryption for business. By looking at the subtopics of the topic, we can get a rough idea that the first subtopic is about a Japanese phone company, since the top weighted words include

---

[4] Note that *topic cohesion* is different from *topic coherence* as *topic cohesion* measure the relevant between a subtopic and its parent topic.

Concept phrase	Parent topic	Subtopics
atheism	people, question, strong, belief, make, god, thing, point, religion, lack	moral, system, morality, society, nature, objective, human, dream, animal, action
		people, question, thing, make, point, god, argument, belief, claim, true
		science, scientific, theory, result, observation, scientist, experiment, hypothesis, methodology, model
for sale	sell, sale, original, sold, interested, included, price, offer, box, cd	sale, offer, price, sell, original, shipping, box, condition, interested, cd
		list, interested, send, mail, address, post, email, original, information, call
business	3do, government, key, phone, technology, company, business, chip, encryption, clipper	company, phone, 3do, technology, business, number, system, japanese, computer, make
		government, clipper, encryption, chip, system, nsa, phone, people, security, key
		key, chip, algorithm, number, clipper, encryption, de, escrow, system, secret

Table 4: Parent topic and the subtopics of the concepts "*atheism*", "*for sale*" and "*business*" for the 20newsgroup dataset.

"*japanese*", "*company*", "*phone*", "*technology*". This makes sense since phone companies usually have a strong requirement for encryption. The second and third subtopics are about encryption algorithms, since the top weighted words include "*chip*", "*nsa*", "*key*" and "*algorithm*". We further verified that our interpretation is correct by looking at the top weighted documents of the topics. This confirms that our model has potential for use in real world applications.

5 Conclusions and Future Work

We presented a novel, query-driven topic model to help identify topics of interest in large datasets. Instead of asking experts to define keywords for these topics, we implemented a concept words extractor to automatically extract concept words and used the GPU model, incorporating word-filtering, to improve interpretability and performance. To distinguish between different contexts for the same concept, we further introduced a subtopic modeling procedure. The procedure can automatically infer all subtopics without having to determine the optimal number of subtopics beforehand. Experimental results on three benchmark datasets demonstrate the model's promise. In the future, we plan to evaluate our model's performance using real-world, qualitative analysis use cases.

Acknowledgments

This work is partly funded by the EPSRC (grant no. EP/T017112/1, EP/V048597/1). ZF receives the PhD studentship jointly funded by the University of Warwick and China Scholarship Council. YH and RP are supported by Turing AI Fellowships funded by UK Research and Innovation (UKRI) (grant nos. EP/V020579/1 and EP/N510129/1 respectively).

References

David Andrzejewski and Xiaojin Zhu. 2009. Latent dirichlet allocation with topic-in-set knowledge. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 43–48.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th annual international conference on machine learning*, pages 25–32.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

David M Blei, Thomas L Griffiths, Michael I Jordan, and Joshua B Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16(16):17–24.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Claudio Carpineto, Renato De Mori, Giovanni Romano, and Brigitte Bigi. 2001. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)*, 19(1):1–27.

Stefano Ceri, Alessandro Bozzon, Marco Brambilla, Emanuele Della Valle, Piero Fraternali, and Silvia Quarteroni. 2013. An introduction to information retrieval. In *Web information retrieval*, pages 3–11. Springer.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Leveraging multi-domain prior knowledge in topic models. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891*.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. 2020. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Mennatallah El-Assady, Rebecca Kehlbeck, Christopher Collins, Daniel Keim, and Oliver Deussen. 2019. Semantic concept spaces: Guided topic model refinement using word-embedding projections. *IEEE transactions on visualization and computer graphics*, 26(1):1001–1011.

Ian Gemp, Ramesh Nallapati, Ran Ding, Feng Nan, and Bing Xiang. 2019. Weakly semi-supervised neural topic models.

Gregor Heinrich. 2009. A generic approach to topic models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 517–532. Springer.

Nan Hu, Ting Zhang, Baojun Gao, and Indranil Bose. 2019. What do hotel customers complain about? text analysis using structural topic model. *Tourism Management*, 72:417–426.

Victor Lavrenko and W Bruce Croft. 2017. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM New York, NY, USA.

Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 165–174.

Sophie Melville, Kathryn Eccles, and Taha Yasseri. 2019. Topic modeling of everyday sexism project entries. *Frontiers in Digital Humanities*, 5:28.

Sergey I Nikolenko, Sergei Koltcov, and Olessia Koltsova. 2017. Topic modelling for qualitative studies. *Journal of Information Science*, 43(1):88–102.

Adler Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. 2011. Hierarchically supervised latent dirichlet allocation. *Advances in neural information processing systems*, 24:2609–2617.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*.

Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392.

Rui Wang, Deyu Zhou, and Yulan He. 2020. Optimising topic coherence with weighted po´ lya urn scheme. *Neurocomputing*, 385:329–339.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, and Jun Zhao. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

Yueshen Xu, Jianwei Yin, Jianbin Huang, and Yuyu Yin. 2018. Hierarchical topic modeling with automatic knowledge mining. *Expert Systems with Applications*, 103:106–117.

Yi Yang, Doug Downey, and Jordan Boyd-Graber. 2015. Efficient methods for incorporating knowledge into topic models. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 308–317.

Fang Yao and Yan Wang. 2020. Tracking urban geo-topics based on dynamic topic model. *Computers, Environment and Urban Systems*, 79:101419.

Appendix A: Query and keywords for each category

Concept Phrase	Expert Defined Keywords									
atheism	Agnosticism	theism	deism	islam	paganism	moral	atheist	religions	argument	exist
computer graphics	image	digital	visual	3d	2d	visualization	print	geometry	synthesizing	processing
pc hardware	cpu	monitor	keyboard	memory	card	sound	speakers	motherboard	power	pc
mac hardware	touchpad	touchbar	drive	apple	mac	ram	gpu	system	sensors	physical
for sale	product	mail	discount	bargain	shopping	price	sale	propertise	rent	summer
automobile	car	vehicle	transportation	wheel	tire	road	parking	gasoline	energy	driver
motorcycles	bike	scooters	mopads	motorbikes	trowel	commute	helmet	ride	speed	harley
baseball	player	ball	small	hit	team	fielding	batting	runs	nbl	baseball
hockey	puck	nhl	hockey	ice	rink	canada	rubber	curve	skater	guard
encrypt	encoding	decryption	cryptographic	secure	plaintext	ciphertext	key	algorithm	pseudo	private
electronics	equipment	science	electricity	wire	console	computer	outlet	engineering	power	voltage
medicine	medicine	surgery	hospital	climic	doctor	nurse	healthcare	symtoms	prescription	pharmacy
space	rocket	nasa	astronomy	explore	moon	outerspace	spaceship	telescope	satellite	orbit
christian	belief	faith	church	christianity	ethics	culture	ritual	Jesus	bible	truth
guns	law	regulation	usa	victim	murder	violence	litigation	debate	firearms	legal
middle east	israel	Iran	Iraq	war	territory	turkey	attack	soldier	turkey	government

Table A1: Concept phrases and expert defined keywords for the 20NewsGroup dataset.

Concept Phrase	Expert Defined Keywords									
business	bank	stock	market	business	economy	financial	investor	profit	price	deal
entertainment	film	movie	music	tv	theater	festival	actor	show	book	hollywood
health	drug	health	cancer	patient	disease	medical	hospital	healthcare	doctor	treatment
technology	apple	google	sony	facebook	internet	mobile	ipad	technology	microsoft	phone
sport game	league	win	player	team	tournament	game	playoff	sport	championship	point

Table A2: Concept phrases and expert defined keywords for the TagMyNews dataset.

Concept Phrase	Expert Defined Keywords									
Business	bank	stock	market	business	economy	financial	investor	profit	price	deal
Computers	computer	software	programming	parallel	computing	memory	hardware	driver	cpu	processor
Culture Arts Entertainment	movie	music	art	film	artist	museum	fashion	culture	imdb	actor
Education Science	research	science	journal	university	student	education	scientific	mathematics	theory	school
Car Engineering	engine	electrical	car	wheel	model	automobile	industrial	vehicle	cylinder	jet
Health	drug	health	cancer	patient	disease	medical	hospital	healthcare	doctor	treatment
Politics Society	political	party	democracy	government	republic	parliamentary	representative	president	communist	congress
Sports	league	football	player	team	tournament	game	basketball	sport	hockey	championship

Table A3: Concept phrases and expert defined keywords for the SearchSnippets dataset.

Appendix B: Topics generated for target concepts

Concept Phrase	**Top 10 words**									
atheism	people	question	thing	god	strong	point	make	belief	argument	evidence
computer graphics	support	file	version	image	list	graphic	program	information	screen	address
pc hardware	drive	pc	disk	scsi	software	modem	port	hard	controller	system
mac hardware	card	mac	monitor	apple	system	video	problem	write	chip	work
for sale	sell	sale	sold	original	interested	included	price	offer	box	cd
automobile	car	auto	automobile	engine	ford	problem	mile	v6	oil	dealer
motorcycles	bike	riding	ride	motorcycle	rider	battery	buying	dog	back	dod
baseball	run	baseball	game	pitcher	year	hit	player	average	team	good
hockey	game	hockey	team	nhl	night	goal	player	coach	cup	year
encrypt	key	message	chip	encryption	government	clipper	algorithm	system	phone	encrypted
electronics	company	power	line	led	electronics	electronic	circuit	output	work	signal
medicine	patient	medical	treatment	doctor	disease	study	clinical	medicine	food	effect
space	space	nasa	shuttle	launch	satellite	station	moon	1st	cost	orbit
christian	god	christ	church	christian	love	word	bible	jesus	protestant	truth
guns	gun	control	weapon	child	people	police	fire	law	handgun	amendment
middle east	israel	arab	armenian	jew	israeli	muslim	people	middle	east	war

Table B1: Topics of target concepts for the 20NewsGroup dataset.

Concept Phrase	**Top 10 words**									
Business	business	management	marketing	trade	service	market	law	export	job	stock
Computers	computer	computing	software	web	application	system	programming	apple	memory	chip
Culture Arts Entertainment	art	culture	music	american	artist	tradition	history	ancient	museum	band
Education Science	science	education	research	scientific	undergraduate	biology	journal	school	university	fiction
Car Engineering	car	model	engineering	automobile	engine	wheel	auto	electrical	product	motor
Health	health	disease	care	cancer	public	nutrition	information	medical	gov	drug
Politics Society	political	party	democracy	system	military	politics	government	conflict	gov	war
Sports	football	team	soccer	game	sport	hockey	news	tennis	score	player

Table B2: Topics of target concepts for the SearchSnippets dataset.

Appendix C: Algorithms

Algorithm 2: Initialize(K, D, W_Q)

```
foreach document $j \in D$ do
    $k \leftarrow z \neq \tilde{z}_Q \sim Multinomial(1/K)$;
    foreach position $i \in j$ do
        foreach $W_q \in W_Q$ do
            if $w_{ji} \in W_q$ then
                $k \leftarrow \tilde{z}_q$;
                break;
            end
        end
    end
    $t \leftarrow 0$;
    foreach position $i \in j$ do
        $S_{j,w_{ji}} \leftarrow 0$;
        $t_{ji} \leftarrow t$;
        $k_{jt} \leftarrow k$;
        UpdateCounter($S_{j,w_{ji}}$, $t_{ji}$, $k_{jt}$, $True$);
        $t \leftarrow t + 1$;
    end
end
```

Algorithm 3: UpdateCounter($S_{j,w}$, t, k, $operation$)

```
if $operation == True$ then
    if $t == t^{new}$ then
        $m_k \leftarrow m_k + 1$;
    end
    if $S_{j,w} == 1$ then
        /* To apply GPU */;
        foreach $(w_i, w_q) \in \mathcal{M}$ do
            if $w == w_i$ then
                $n_{jt} \leftarrow n_{jt} + A_{i,q}$;
                $n_{kw} \leftarrow n_{kw} + A_{i,q}$;
            end
        end
    else
        $n_{jt} \leftarrow n_{jt} + 1$;
        $n_{kw} \leftarrow n_{kw} + 1$;
    end
else
    if $S_{j,w} == 1$ then
        /* To remove counts from GPU */;
        foreach $(w_i, w_q) \in \mathcal{M}$ do
            if $w == w_i$ then
                $n_{jt} \leftarrow n_{jt} - A_{i,q}$;
                $n_{kw} \leftarrow n_{kw} - A_{i,q}$;
            end
        end
    else
        $n_{jt} \leftarrow n_{jt} - 1$;
        $n_{kw} \leftarrow n_{kw} - 1$;
    end
    if $n_{jt} == 0$ then
        $m_k \leftarrow m_k - 1$;
    end
end
```

Note: n_{kw} denotes the word count of w in topic k; $\tilde{z}_Q$ denotes the concept related topics.

How Reliable are Model Diagnostics?

Vamsi Aribandi[*]
Google Research
aribandi@google.com

Yi Tay
Google Research
yitay@google.com

Donald Metzler
Google Research
metzler@google.com

Abstract

In the pursuit of a deeper understanding of a model's behaviour, there is recent impetus for developing suites of probes aimed at diagnosing models beyond simple metrics like accuracy or BLEU. This paper takes a step back and asks an important and timely question: how *reliable* are these diagnostics in providing insight into models and training setups? We critically examine three recent diagnostic tests for pre-trained language models, and find that likelihood-based and representation-based model diagnostics are not yet as reliable as previously assumed. Based on our empirical findings, we also formulate recommendations for practitioners and researchers.

1 Introduction

Contemporary statistical models based on deep learning have made incredible progress towards solving complex language tasks (Radford et al., 2019; Devlin et al., 2019; Raffel et al., 2020). These models generally trade off the interpretability and simplicity of traditional models for powerful parameterizations and inductive biases, enabling their impressive performance. However, their entry into critical fields such as medicine, the justice system, and social media moderation often makes this trade-off a costly one. Consequently, there has been surging interest in the development of tools and suites for diagnosing and better understanding model behaviour, and gaining insight into what patterns and phenomena they have learned (§4.1).

Ideally, these diagnostics would not only help practitioners understand the failure modes and capabilities of large contemporary models, but also enable them to improve their models based on the diagnostics. To this end, we believe that model diagnostics are essential for making meaningful progress in natural language processing.

Model diagnostics generally probe a model for specific learned qualities (§4.1). These may be a positive qualities (e.g., whether a model has acquired syntactic knowledge) or potentially problematic qualities (e.g., biases and stereotypes. These probes can be used to identify certain phenomena that can be used to further improve models.

Given the potential impact that model diagnostics can have for practitioners and the research community's fundamental understanding of contemporary models, this paper asks the important and inevitable question of whether these probes are actually reliable and robust, and to what extent they are. These diagnostics' explicit nature as a tool for understanding also imposes a greater bar for robustness, as inconsistencies may mislead and result in compounding errors.

Our findings demonstrate that model diagnostics can be unreliable on multiple fronts. To illustrate our point, we select three diagnostics tasks — StereoSet (Nadeem et al., 2020), CrowS-Pairs (Nangia et al., 2020), and SEATs (May et al., 2019) to base our empirical evaluation on. Overall, we find that **likelihood-based and representation-based diagnostics measured multiple times on the same training setup can result in wildly different findings.** Specifically, a substantial variance is observed when performing the same model diagnostics on identical BERT (Devlin et al., 2019) pre-training setups while varying minute details such as the initial random seed or choice of representation.

These findings are meant to caution researchers and practitioners that rely on such diagnostics so that they can be more mindful of these phenomena when analyzing their models in the future. We discuss the implications of our findings and propose recommendations for practitioners and researchers in §5.

---

[*] Google AI Resident

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1778–1785
August 1–6, 2021.

2 Methodology

2.1 Training setup

We pre-train 5 BERT BASE and LARGE uncased English models, each with the same configurations as in Devlin et al. (2019) using Tensorflow[1]. However, each model differs in its random seed, resulting in different parameter initializations and training data permutations. Hence, it is expected that the checkpoints will each end up at a different local minima. It should be noted that BERT uses static masking instead of dynamic masking, so the set of pre-training examples remains the same.

To decouple our findings from phenomena that occur as a result of using different training setups, we restrict our experiments to only those that require pre-trained BERT models, eliminating many probes mentioned in §4.3. Webster et al. (2020) report that patterns learned during pre-training are often resilient to fine-tuning, further supporting our reasoning.

2.2 Likelihood-base diagnostics

One approach to examining the behaviour of language models like BERT is to examine how they rank certain representative examples above others. We use two contemporary datasets that measure how often stereotypes are ranked above anti-stereotypes — StereoSet (Nadeem et al., 2020) and CrowS-Pairs (Nangia et al., 2020). Both datasets measure $ss = 100 * \sum_{n=1}^{|X|} 1_{[ll(x_n^{ster}) > ll(x_n^{anti})]} / |X|$.

StereoSet Nadeem et al. (2020) propose a benchmark that contains intra-sentence and inter-sentence examples of stereotypes and anti-stereotypes. Here, likelihoods are calculated as $ll(x) = p(x_\tau | x_{\setminus\tau})$ (where τ is the set of target demographic word(s) in x) and $ll(x) = p(\texttt{isNext}|x_1, x_2)$ for intra-sentence and inter-sentence examples respectively. They also propose and combine a language modeling score (lms) with ss into a hybrid metric ($icat$), but we only report ss to focus on StereoSet's primary purpose — measuring stereotypical preference in language models. We report results on the development set.

CrowS-Pairs Nangia et al., 2020 propose a test that contains intra-sentence examples, where likelihoods are calculated by conditioning on the target demographic word(s) in the sentence ($ll(x) = p(x_{\setminus\tau} | x_\tau)$) rather than vice-versa as in StereoSet.

The CrowS-Pairs diagnostic is expected to show higher variance than StereoSet for two reasons: (1) it is a smaller dataset ($\sim\frac{1}{3}$rd the size of StereoSet-dev) with more categories, so results are more sensitive to changes in individual predictions; and (2) the pseudo-likelihood it uses is more susceptible to the poor calibration (Jiang et al., 2020a; Desai and Durrett, 2020) of contemporary models, since the number of multiplied probabilities grows linearly with the number of words in a sentence.

2.3 Vector-space diagnostics

Directly examining representations learned by models is another way to understand their behavior. This is typically done by measuring relationships between different types of inputs, for example in terms of their relative orientations in a vector space.

SEATs We use Sentence Encoder Association Tests (SEATs; May et al., 2019), which extend the popular Word Embedding Association Tests (WEATs; Caliskan et al., 2017) by constructing "semantically bleached" sentences. A WEAT/SEAT measures the *effect size* $s(X, Y, A, B)$ of the association between two targets (e.g., X=`MentalDisease` and Y=`PhysicalDisease`) and two attributes (e.g., A=`Temporary` and B=`Permanent`), as well as the statistical significance of the association using a permutation test[2]. We conduct experiments using the same SEATs as in May et al. (2019). In addition to testing sentence (`[CLS]`) representations, we also test the contextualized word representations of the target/attribute words in the sentences. The reason we do this is that even for semantically bleached sentences, it is often non-trivial for models to encode information about an entire sentence in a single vector[3].

In addition to examining effect sizes, we also conduct an experiment to see how distinguishable representations of certain concepts are in vector space (e.g., do representations of `Pleasant` and `Unpleasant` sentences form their own clusters?). We do this by clustering (via *k*-means) sentence representations and subsequently examining how well the unsupervised clusters align with the actual categories. The aim of this experiment is to understand vector space diagnostics behave the way they do.

---

[1] `https://github.com/tensorflow/models/tree/master/official/nlp/bert`

[2] Please see Appendix A for how SEATs are computed.

[3] `https://www.cs.utexas.edu/~mooney/cramming.html`

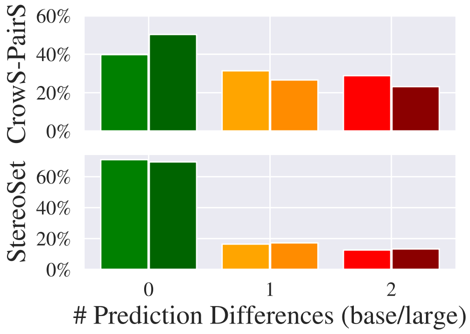

Figure 1: % of examples in likelihood-based tests that have d different predictions over 5 runs. Ideally, examples would always (100%) be predicted the same (d=0).

Test	Cat.	N	BERT results (%)	
			BASE	LARGE
CrowS-Pairs	Race	516	54.4 ± **4.7**	55.9 ± **2.7**
	Gen.	262	58.2 ± 2.5	61.1 ± 1.7
	S.O.	84	63.2 ± 3.4	67.4 ± **4.6**
	Rel.	105	68.9 ± **8.0**	72.2 ± 2.1
	Age	87	55.4 ± **4.2**	60.9 ± **5.6**
	Nat.	159	51.2 ± 1.2	55.3 ± **3.5**
	Dis.	60	69.0 ± **3.8**	79.0 ± 1.9
	P.A.	63	59.1 ± **4.9**	64.4 ± **4.3**
	Occ.	172	54.9 ± **4.5**	58.0 ± **4.2**
	all	1508	57.1 ± **2.8**	60.3 ± 1.7
StereoSet	Gen.	496	59.1 ± 0.7	62.4 ± 2.0
	Occ.	1636	60.5 ± 0.6	61.4 ± 0.8
	Race	1938	54.8 ± 1.1	56.4 ± 0.8
	Rel.	156	51.8 ± **2.8**	54.4 ± **3.3**
	all	4226	57.4 ± 0.7	59.0 ± 0.7

Table 1: Likelihood-based diagnostics over categories often have high standard deviation (bold) over pre-training runs, often varying from almost neutral (∼50%) to a significant amount (highlighted).

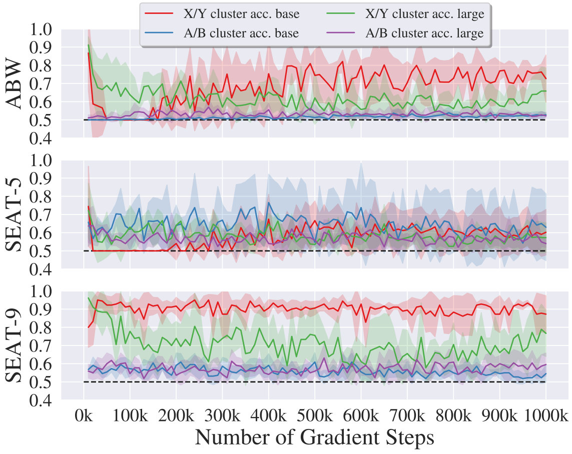

Figure 2: Representations of targets and/or attributes often don't cluster over pre-training. The dashed line is when representations are indistinguishable (acc. = 0.5).

3 Findings and Insights

3.1 Likelihood-based diagnostics are unstable

Experiments on StereoSet and CrowS-Pairs show that while likelihood-based ranking diagnostics may be stable across all categories, instability is evident in the results of individual categories (Table 1). Many categories have a standard deviation of over 2.5 percentage points. Some categories also vary from almost no stereotypical preference to a significant amount (highlighted in Table 1) — a result that could potentially cause practitioners to draw false conclusions.

Additionally, from Figure 1 it is evident that many examples are assigned different labels over the 5 pre-trained models, often having 3 models assign them one label and 2 models assigning them the opposite label — almost as random as a coin flip! The implies that the models are probably uncertain about their predictions for these datapoints, motivating the consideration of model uncertainty in diagnostic measures instead of simply making a binary decision by comparing likelihoods.

Worryingly, both tests report wildly differing results on religious stereotypes ("Rel."), with CrowS-Pairs detecting strong stereotypical preference and StereoSet detecting almost none. It is also worth noting that results on CrowS-Pairs exhibit far higher variance compared to StereoSet (Table 1, Figure 1), as hypothesized in §2.2.

3.2 Vector-space diagnostics are unstable

Representation-based experiments exhibit high variance across multiple pre-training runs, choices of representation, and model sizes (Figure 3). Notably, SEAT results are often on both sides of the "neutral" mark (0), and their statistical significance is often erratic. In other words, it is possible for two models to be pre-trained with the exact same configurations but different random seeds to yield completely opposite conclusions on some SEATs. Moreover, the same checkpoint often yields different results depending on whether sentence or pooled target-word representations are used. Ideally, a SEAT would *always* or *never* be statistically significant, and yield effect sizes with the same sign over multiple pre-training runs and (seemingly innocuous) choices of representation.

From Figure 2, the representational instability of semantically bleached SEAT sentences is further evident — how these representations cluster

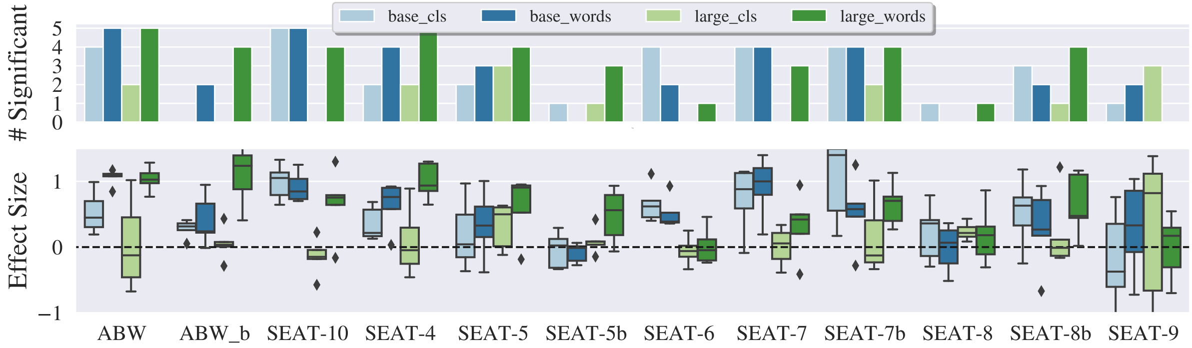

Figure 3: SEAT results exhibit high variance across pre-train runs, model sizes, and choice of representation. Moreover, effect sizes often vary around the "neutral" mark (0) and also have different statistical significances (at $p = 0.01$). Ideally, a test would always (5) or never (0) be significant, and yield effect sizes with the same sign.

together is erratic both across pre-training steps as well as across multiple pre-training runs. This result gives us further insight into why high variance is observed for vector-space diagnostics — representations often can't form their own clusters for certain concepts, so simply examining their relative orientations is insufficient. Our findings provide empirical arguments for what May et al. (2019) surmise — there is scope for sentence embedding-based tests that do more than naturally extend word embedding-based tests with semantically bleached sentences.

We surmise that representation-based diagnostics are less stable than likelihood-based diagnostics because large models like BERT are optimized to be good at modeling likelihoods via their pre-training objective. However, there is no constraint on how sentences must be represented other than it should be possible to "extract" correct likelihoods from them. In other words, there is no reason to expect the orientations of these representations to provide deep insight into what these models learn.

3.3 Diagnostic instability is despite equivalent downstream performance

We fine-tuned the 10 checkpoints on SST-2 (Socher et al., 2013), RTE (Dagan et al., 2006; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), and QNLI (Rajpurkar et al., 2016) from the GLUE benchmark (Wang et al., 2019). Development-split results show that performance was largely the same across checkpoints (Table 2) despite diverging behaviour on the model diagnostics as shown in §3.1 and §3.2. **This shows that the different local optima still perform largely the same on downstream tasks despite behaving differently with respect to model diagnostics**.

Dataset	**BERT fine-tuning results**	
	BASE	LARGE
SST-2	91.2 ± 0.3	93.0 ± 0.3
RTE	71.3 ± 1.2	76.8 ± 1.8
QNLI	92.1 ± 0.2	92.1 ± 0.3

Table 2: The checkpoints generally exhibit equivalent performance on downstream tasks.

Dev-set performance is also largely consistent with what is expected of BERT BASE and LARGE models. It should be noted that we only used one set of hyperparameters and did not perform the hyperparameter sweep as in Devlin et al. (2019), so further tuning would likely improve results.

4 Related Work

4.1 Model Diagnostics

Models have been probed to understand what exactly they learn beyond traditional language tasks, ranging from their linguistic capabilities (Adi et al., 2017; Tenney et al., 2019; Conneau et al., 2018; Ribeiro et al., 2020; Belinkov et al., 2017; Hewitt and Manning, 2019; Marvin and Linzen, 2018), multilingual capabilities (Pires et al., 2019; Kudugunta et al., 2019), world knowledge (Jiang et al., 2020b; Petroni et al., 2019), and social bias (Nadeem et al., 2020; Nangia et al., 2020; May et al., 2019) among other phenomena.

Another axis to compare model diagnostics on is whether they are intrinsic or extrinsic, i.e., whether they directly analyze models for certain phenomena that aren't tied to any downstream task or do so keeping particular tasks in mind. This paper restricts itself to intrinsic tasks for reasons mentioned in §2.1. An example of an extrinsic task is

Rudinger et al. (2018), which probes models for gender bias through the lens of coreference resolution. We refer readers to Belinkov and Glass (2019) for a more comprehensive survey on model analysis for natural language processing.

4.2 Diagnostic Fragility

It has been shown that classifier probes — which require an additional classifier (like an MLP) to be trained on top of frozen model representations — are unstable (Voita and Titov, 2020), and that it might not be clear from their results whether the probe *itself* learned a phenomena or whether the diagnosed representations learned it (Hewitt and Liang, 2019). Similarly, Wang et al. (2020) find that gradient-based analysis of language technologies based on neural networks can often be unreliable and manipulable. Attention-based interpretation can also be unreliable and manipulable to the point of deceiving practitioners, as Pruthi et al. (2020) and Jain and Wallace (2019) show. The works mentioned above all support our arguments, and some raise similar concerns to those expressed in this paper.

4.3 Inconsistencies between equivalent checkpoints

This paper's findings can be linked to the problems caused by underspecification in machine learning (D'Amour et al., 2020), i.e., when multiple unique predictors trained with the same configuration have the same performance but differ in subtle ways. In a setting where practitioners might train and thoroughly analyze one model but then retrain it and assume that the first checkpoint's model diagnostics hold for the second one, this issue is highly relevant. McCoy et al. (2020) also find that separately fine-tuned BERT models often vary significantly in generalizing to auxiliary tasks.

5 Discussion

Recommendations No probe is perfect, but it is clear that model diagnostics are not as reliable as previously assumed. Our empirical findings — coupled with the works mentioned in §4.2 and §4.3 — motivate careful scrutiny of model diagnostics. **We recommend that:**

- Practitioners not generalize a single diagnostic result to the entire training setup, and instead restrict conclusions to a specific checkpoint.
- Researchers proposing probes not only test on publicly available checkpoints, but rather examine a probe's performance and robustness across a range of model/probe configurations.

Future Work While this paper primarily aims to motivate further scrutiny of model diagnostics, we hope it motivates studies that ask *why* these diagnostics often behave unreliably. One future research direction we are excited about is analyzing correlations between the properties of the models' local minima in the loss landscape and behaviour on model diagnostics. This would not only be another step towards a better understanding of how contemporary deep language models work, but also enable researchers to use that information to design better, more robust model diagnostics. Such a study may even help inform the optimization process for future state-of-the-art language technologies.

It should also be noted that this paper is restricted to three diagnostics spanning likelihood-based and representation-based probes, and that future work is needed to determine the extent to which other diagnostic probes are reliable.

6 Conclusion

In this paper, we motivate further scrutiny of model diagnostics that aim to understand the behaviour of contemporary "black-box" language technologies. Our results show that model diagnostics are often fragile and can yield different conclusions as a result of seemingly innocuous configuration changes. We hope that our results over multiple pre-train runs will encourage researchers and practitioners to be mindful of the reliability of such model diagnostics when verifying hypotheses about their models and training setups.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of* *the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. 2020. Underspecification presents challenges for credibility in modern machine learning.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. How can we know when language models know?

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020b. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Sneha Kudugunta, Ankur Bapna, Isaac Caswell, and Orhan Firat. 2019. Investigating multilingual NMT representations at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1565–1575, Hong Kong, China. Association for Computational Linguistics.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2020. Stereoset: Measuring stereotypical bias in pretrained language models.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. 2020. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Junlin Wang, Jens Tuyls, Eric Wallace, and Sameer Singh. 2020. Gradient-based analysis of NLP models is manipulable. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 247–258, Online. Association for Computational Linguistics.

Kellie Webster, Xuezhi Wang, Ian Tenney, Alex Beutel, Emily Pitler, Ellie Pavlick, Jilin Chen, and Slav Petrov. 2020. Measuring and reducing gendered correlations in pre-trained models.

A SEAT computation

The effect size of a SEAT — characterized by two target (X, Y) and two attribute (A, B) sets of sentences — is calculated as:

$$d = \frac{\text{mean}_{x \in X}\, s(x, A, B) - \text{mean}_{y \in Y}\, s(y, A, B)}{\text{stdev}_{z \in X \cup Y}\, s(z, A, B)}$$

where:

$$\begin{aligned} s(sent, A, B) = \text{mean}_{a \in A} \cos(\overrightarrow{sent}, \overrightarrow{a}) \\ - \text{mean}_{b \in B} \cos(\overrightarrow{sent}, \overrightarrow{b}). \end{aligned}$$

The *p*-value of the permutation test to determine the statistical significance of the effect size is calculated as::

$$p = \Pr[S(X_i, Y_i, A, B) > S(X, Y, A, B)]$$

over partitions (X_i, Y_i) of $(X \cup Y)$ such that $|X_i| = |Y_i|$, where:

$$S(X, Y, A, B) = \sum_{x \in X} s(x, A, B) - \sum_{y \in Y} s(y, A, B)$$

Gaussian Process based Deep Dyna-Q Approach for Dialogue Policy Learning

Guanlin Wu [1,2,*] **Wenqi Fang** [3,*,†] **Ji Wang** [1,*] **Jiang Cao** [2] **Weidong Bao** [1]
Yang Ping [2] **Xiaomin Zhu** [1] **Zheng Wang** [4]
[1]National University of Defense Technology, Changsha, China
[2]Academy of Military Science, Beijing, China
[3]Nanhu Laboratory, Jiaxing, China
[4]Shenzhen Institutes of Advanced Technology, CAS, Shenzhen, China
`{wuguanlin16,wangji,wdbao,xmzhu}@nudt.edu.cn`
`ocean.py@163.com` `amscaojiang@126.com`
`wqfang@nanhulab.ac.cn` `zheng.wang@siat.ac.cn`

Abstract

Applying reinforcement learning to dialogue policy learning requires prohibitively large rounds of human-machine interactions. To improve the learning performance, the Deep Dyna-Q framework with a world model that imitates real users is widely used in recent years. Unfortunately, how to build an effective world model and how to evaluate the experiences generated by the world model efficiently have not been well studied. In order to further improve the effectiveness and efficiency of dialogue policy learning, we present a novel Gaussian Process based Deep Dyna-Q approach in this paper. The Gaussian Process model, which is analytically tractable and fits for small-sample problems, is introduced to build the world model. In addition, we design a highly efficient Kullback-Leibler divergence based discriminator to evaluate the quality of experiences generated by the world model. Extensive experiments validate the effectiveness and robustness of our proposed approach. The task-completion success rate can be improved by about 20% with fewer human-machine interactions.

1 Introduction

Task-completion dialogue policy learning aims to build a task-completion dialogue system that can help people complete a specific single task or multi-domain tasks through several rounds of natural language interactions. It has been widely used in chat robots and personal voice assistants, such as Siri of Apple and Cortana of Microsoft.

Reinforcement learning (RL) becomes the mainstream dialogue policy learning method in recent years (Chen et al., 2020; Saha et al., 2020; Li et al., 2020). Based on the RL, the task-completion dialogue system can gradually adjust policy through interacting with real users to improve performance. However, the vanilla RL methods require many rounds of human-machine dialogue interactions before getting a satisfactory dialogue policy, which not only increases the training cost but also deteriorates user experience during the early training phase.

In order to address the above problem and accelerate the learning process of dialogue policy, Deep Dyna-Q (DDQ) (Peng et al., 2018) is proposed based on the Dyna-Q framework where a environment model, known as *world model*, is introduced to generate simulated user experiences in the dynamic environment. The world model is trained by the real user experience to make itself act more like real users. During the dialogue policy learning, the dialogue agent is trained by both real experiences collected from interacting with real users and simulated experiences collected from interacting with the world model.

By introducing the world model, DDQ can promote the learning efficiency effectively during dialogue policy learning. However, it still faces two critical challenges which are crucial to further improve the dialogue policy learning with limited dialogue interactions.

Firstly, the world model in DDQ is built as a deep neural network (DNN) whose performance heavily relies on the amount of training data. In the initial training stage when the real experiences are relatively few, the data-hungry problem caused by DNN may make the world model fail to generate simulated user experiences with enough quality. It requires a lot of real experiences to train a qualified DNN-based world model that can produce high-quality simulated experiences. The world model implemented by the data-hungry model such as DNN erodes the advantage brought by Dyna-Q framework and makes DDQ less effective in reality.

Secondly, it has been pointed in (Peng et al.,

---

[*] The three authors contribute equally to this work
[†] Corresponding author: wqfang@nanhulab.ac.cn

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1786–1795
August 1–6, 2021.

2018) that the simulated experience generated by the world model does not necessarily improve performance. Low-quality experiences even hinder the performance seriously. To address this issue, some recent works attempted to control the quality of simulated experiences by using generative adversarial network (GAN) to discriminate the low-quality experiences (Su et al., 2018). Nonetheless, the notorious instability of training GANs may make dialogue policy learning suffer badly from non-convergence and high sensitive to the hyperparameter selections, which is demonstrated in Section 3 of our paper. It is an important yet unsolved problem to efficiently discriminate low-quality experiences during dialogue policy learning.

In order to tackle the above two challenges, we propose a new Gaussian Process based Deep Dyna-Q approach. Compared with the previous works (Peng et al., 2018; Su et al., 2018), the world model in our approach is built as a Gaussian Process (GP) model rather than a DNN model. The GP model is analytically tractable and enjoys the advantage of dealing with small-sample problems (Patacchiola et al., 2019; Gašić et al., 2017; Su et al., 2016), which makes it more competitive than DNN models in this work. In addition, we design a novel method to evaluate the quality of simulated user experiences by comparing them with real user experiences based on Kullback-Leibler (KL) divergence directly without any extra training of discriminator. The main contributions of this work are as follows:

- We present a new GP-based Deep Dyna-Q approach, which can generate high-quality simulated experiences to supplement the limited real user experience. To build the world model as a GP model, we design a Dyna-Q framework that supports regression mode meeting the basic requirements of using GP methods.
- We propose a KL divergence based discriminator which is able to fluently control the quality of simulated experiences. By introducing KL divergence, we can check the distribution of experiences without wasting extra work to design and train a complex discriminator. It is easier to evaluate the quality of simulated experiences in reality, and greatly improve the computational efficiency while ensuring the robustness and effectiveness of the dialogue policy.

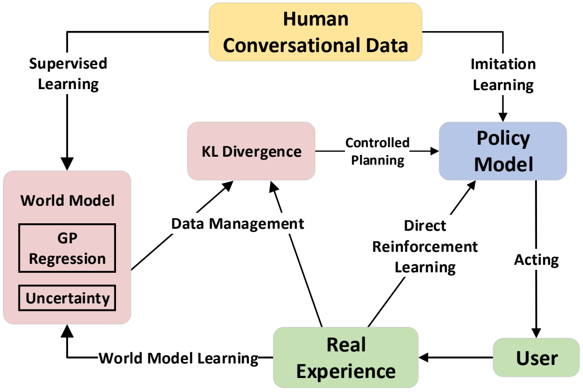

Figure 1: Architecture of the proposed dialogue learning approach.

2 Gaussian Process based Deep Dyna-Q Approach

In this section, we introduce the proposed GP based DDQ approach in detail. Figure 1 shows the architecture of the proposed approach. Our GP based DDQ approach follows the learning process of DDQ, and concentrates on two issues: 1) how to build an effective world model, and 2) how to evaluate simulated experiences efficiently. Accordingly, we build the world model as a GP model and design a novel KL divergence based discriminator to promote the efficiency of dialogue policy learning.

The dialogue policy learning starts with initializing the policy model and the world model by using the human conversational data. In *direct reinforcement learning*, the policy model is trained by interacting with real users to improve the dialogue policy. Meanwhile, the real experiences collected from real users are used to train the world model, which is referred to as *world model learning*. In *data management*, the simulated experiences generated by the world model are evaluated by comparing with the real experiences based on the KL divergence. Then, the qualified ones are pushed into the replay buffer for *controlled planning* to train the policy model without interaction with real users.

2.1 Gaussian Process based World Model

During the planning process, we implement the world model to generate simulated experience that can be used to improve dialogue policy. The world model, denoted by $W(s, a; \theta_w)$, consists of three GP models shown in Figure 2, parameterized by different θ_w. Three GP regression models are used to generate response action a^u, reward r, and variable

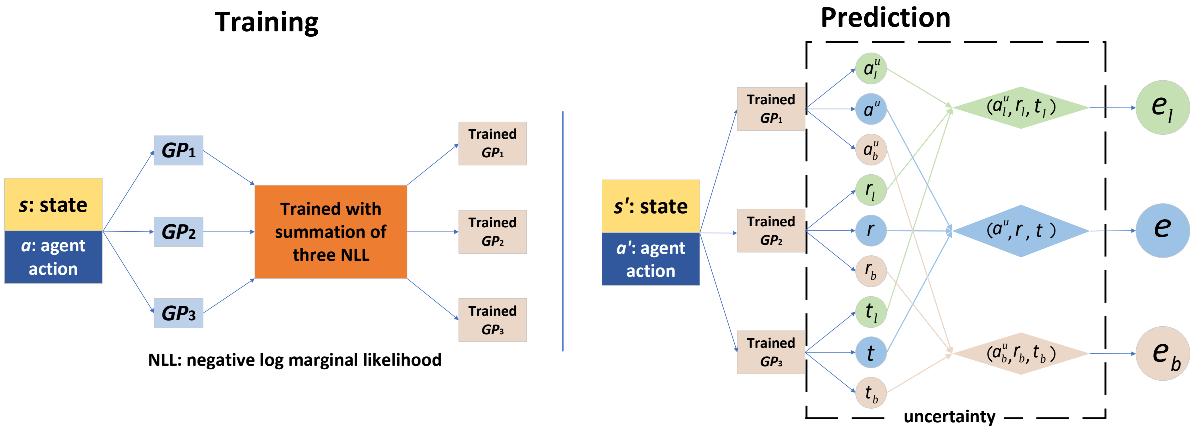

Figure 2: The training and prediction stage of world model.

t indicating whether the dialogue terminates, respectively. We denote the simulated experience as a tuple $\boldsymbol{e} = (a^u, r, t)$. In a practical GP regression problem, the observed targets y are generated from the function $f(x)$ by adding independent Gaussian noise (Williams and Rasmussen, 2006):

$$y = f(x) + \epsilon, \tag{1}$$

where $p(f|x) = N(f|\mu, K(x,x))$ with mean μ and kernel function K, and $\epsilon \sim N(0, \sigma^2 I)$, I is the identity matrix. According to the Bayesian principle, the conditional mean and covariance of posterior distribution, $p(y^*|y, x, x^*)$, with test input x^* is as follows:

$$\mu + K(x^*, x)\Sigma^{-1}(y - \mu) \tag{2}$$

$$K(x^*, x^*) + \sigma^2 I - K(x^*, x)\Sigma^{-1}K(x, x^*), \tag{3}$$

where $\Sigma = K(x,x) + \sigma^2 I$. To accommodate the correlation properties of human dialogue, the stationary kernel function Matérn is used in our case:

$$K_{Mat}(r) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}r}{l}\right), \tag{4}$$

where σ_f and l are magnitude and lengthscale parameters, respectively. Γ is the gamma function, K_ν is the modified Bessel function of the second kind, and ν are positive parameters of the covariance. The argument r represents distance between observations (Hensman et al., 2017). For the multidimensional input case, its automatic relevance determination (ARD) version could be introduced to deal with this situation (Duvenaud, 2014).

In each round of the world model learning, the current dialogue state s and the last agent action a are concatenated as input for the world model. We set all the GP priors with constant mean and the Matérn kernel ($\nu = \frac{7}{2}$) function. The world model $W(s, a; \theta_w)$ is trained to mimic the real dialogue environments. The training data for the world model learning are collected from the real user and are stored in the replay buffer M^w. The loss function is set as the summation of the negative log marginal likelihood (NLL) of three GP models. Because of the conjugate property, each NLL could be analytically solvable, and their general formulas can be written as:

$$-\log p(y|x) = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}(y-\mu)^{\mathrm{T}}\Sigma^{-1}(y-\mu), \tag{5}$$

where $|\cdot|$ represents determinant of the matrix and n is the number of the training data. The world model $W(s, a; \theta_w)$ is refined at the end of each epoch via L-BFGS-B algorithm (Zhu et al., 1997) using real experiences.

During prediction, we use these trained GP models to generate simulated experiences. To increase diversity of the simulated experiences, the uncertainty of GP models is taken into account in the prediction stage, shown by the black box frame labeled as “uncertainty” in Figure 2. We calculate the 50% confidence interval[1] of these three variables. The lower bound and the upper bound of the simulated experience are represented by $\boldsymbol{e}_l = (a_l^u, r_l, t_l)$ and $\boldsymbol{e}_b = (a_b^u, r_b, t_b)$, respectively. Then, we have three simulated experiences $\boldsymbol{e}_l$, $\boldsymbol{e}$, and $\boldsymbol{e}_b$ per prediction. The quality of the three simulated experiences will

---

[1] The usual 95% confidence interval isn’t used here to narrow the variable domain.

be measured by KL divergence, which will be detailed in the following subsection. The qualified simulated experiences will be stored in the replay buffer M^p for training the dialogue policy model.

Differing to DDQ where the world model is essentially a classification model to generate user action a^u, the above GP-based world model is a regression model to make it tractable and much easier to handle than classification model. Considering the user action should be an integer and have finite action domain, the user action generated by the proposed world model should be filtered to meet these inherent requirements. The filtering mechanism consists of the following two steps. Firstly, when the user action is not an integer, which is common in regression case, a^u is round to its nearest integer, a_l^u is replace by its ceiling value, and the floor value of a_b^u is chosen, respectively. Secondly, if the user action is beyond the defined action domain, the upper or the lower bound of the domain will be selected. Through the above process, the user action generated by a regression model can achieve the approximately equivalent effect as the task-specific representation in classification models.

2.2 Management of Replay Buffer

As mentioned in the Introduction, low-quality experiences generated by the world model can hinder the learning performance seriously. In this subsection, we evaluate the quality of the simulated experience to determine whether it can be pushed into the replay buffer for training the dialogue policy model. The whole structure is shown in Figure 3.

We give two dictionaries, i.e., *world-dict* and *real-dict*, to record the frequency of all actions generated by the world model and the real user from the beginning of the dialogue policy learning. The key of the dictionary is user action, and the corresponding value is the frequency of this action. A high-quality simulated experience means that its action is similar to the real user. Therefore, we evaluate the quality of simulated experience by measuring the similarity between *world-dict* and *real-dict* based on the KL divergence which is a non-symmetric variable (Raiber and Kurland, 2017).

The evaluation process is shown in Algorithm 1. This algorithm runs repeatedly during the planning (see Line 19 in Algorithm 2). The variable KL_{pre}, which is initialized as a extremely large number, tracks the KL divergence between *world-dict* and *real-dict*. When evaluating a simulated experience, we first use its user action to update *world-dict*. Then, we use *same-dict* to save the intersection keys of *world-dict* and *real-dict*, and store their frequencies respectively (see Line 2-4). During the initial stage of planning, there is limited actions in *world-dict*, and hence the length of *same-dict* is quite small. To warm up the world model and expand the replay buffer, we regard the simulated experience as a qualified one directly when the length of *same-dict* is smaller than a constant value *cut-off*. Otherwise, we calculate the current KL divergence KL by using *same-dict*. If the current KL divergence is smaller than that of the previous round KL_{pre}, we regard the current experience as a qualified one (see Line 7-8) because it make the world model more similar to the real user. The qualified experience will then be pushed into M^p for training the dialogue policy model.

Algorithm 1: Evaluate Simulated Experiences

```
Input: User actions in the experience generated by
       the world model $a_w^u$; Previous action
       dictionary of the world model world-dict;
       Previous action dictionary of the real user
       real-dict; KL divergence $KL_{pre}$.
1  Update world-dict with the current user actions $a_w^u$ ;
2  foreach a in world-dict.key do
3      if a in real-dict.key then
4          same-dict[a] ← [world-dict[a],real-dict[a]]
5  $qualified$ ← FALSE ;
6  if Length(same-dict) ≥ cut-off then
7      Calculate current KL divergence $KL$ using
         same-dict;
8      if $KL \leq KL_{pre}$ then $qualified$ ← TRUE;
9      $KL_{pre} \leftarrow KL$;
10 else
11     $qualified$ ← TRUE ;
12 if $qualified$ then
13     Push current simulated experience into $M^p$;
```

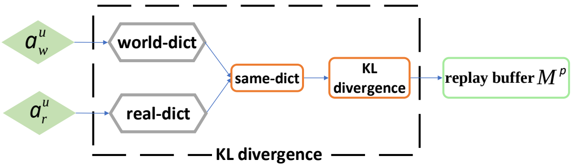

Figure 3: KL divergence calculation.

2.3 Direct and Indirect Reinforcement Learning

For the direct reinforcement learning, the Deep Q-Network (DQN) (Mnih et al., 2015) is adopted to

Algorithm 2: GP based DDQ Approach

```
Input: Learning epoch $count$; probability $\epsilon$ for
       $\epsilon$-policy; planning step $K$; period $T$ to update
       $Q'(s, a; \theta_{Q'})$; user goal $G = (C, R)$, where
       $C$ is a set of constraints and $R$ is a set of
       requests; training epoch $Z_1$ and $Z_2$.
Initialize $Q(s, a; \theta_Q)$ and $W(s, a; \theta_w)$ via
  pre-training on human conversational data ;
Initialize $Q'(s, a; \theta_{Q'})$ with $\theta_{Q'} = \theta_Q$ ;
Initialize the replay buffer $M^w$ for world model and
  $M^p$ for policy model using Reply Buffer Spiking
  (Lipton et al., 2016) ;
for $i \leftarrow 1$ to $count$ do
    #Direct Reinforcement Learning:
    User starts a dialogue with action $a^u$ ;
    Generate an initial dialogue state $s$ ;
    while $t$ is not terminal state do
        Policy model selects and executes action $a$
          based on $\epsilon$-policy ;
        User responses $a_r^u$, reward $r$, and terminal
          state $t$;
        Update real-dict ;
        Store $(s, a, r, a_r^u, t)$ to $M^p$ and $M^w$ ;
        $s \leftarrow s'$ ;
    #Controlled Planning:
    for $k \leftarrow 1$ to $K$ do
        Sample user action $a^u$ from $G$ ;
        while $t'$ is not terminal state do
            Policy model selects and executes
              action $a$ based on $\epsilon$-policy ;
            World model responds $a_w^u$, $r$ and $t'$ ;
            Store $(s, a, a_w^u, r, t')$ to $M^p$ based on
              $qualified$ from Algorithm 1 ;
            $s \leftarrow s'$ ;
    Sample random mini-batch samples from $M^p$ ;
    Update $\theta_Q$ via $Z_1$-step Q-learning ;
    $\theta_{Q'} \leftarrow \theta_Q$ every $T$ steps;
    #World Model Learning:
    Sample random mini-batch samples from $M^w$ ;
    Update $\theta_w$ via $Z_2$-step L-BFGS-B algorithm ;
```

improve the dialogue policy based on real experiences. The dialogue agent interacts with the user and uses a DNN to approximate the non-linear Q function. In each step, the agent chooses the corresponding action a to execute using an ϵ-greedy policy (Watkins and Dayan, 1992) according to the observed dialogue state s. In ϵ-greedy policy, a threshold ϵ is set for logical selection, i.e., a random action or a action chosen by the greedy policy $a = \text{argmax}_{a'} Q(s, a'; \theta_Q)$ where $Q(\cdot)$ is the value function. Then, the agent receives the reward r. The real user responses a_r^u based on the current environment. The next state s' is updated in the state tracker module. Before we store the experience (s, a, r, a_r^u, t) in the replay buffer M^p, the statistical distribution of a_r^u, denoted as *real-dict*, is updated for further KL divergence inspection.

The value function $Q(s, a; \theta_Q)$, approximated by a DNN, is updated by optimizing θ_Q to minimize the mean-squared loss function as below:

$$\begin{aligned} \mathcal{L}(\theta_Q) &= \mathbb{E}_{(s,a,r,s') \sim \mathcal{M}^p}[(y_i - Q(s, a; \theta_Q))^2] \\ y_i &= r + \gamma \max_{a'} Q'(s', a'; \theta_{Q'}), \end{aligned} \tag{6}$$

where $\gamma \in [0, 1]$ is a discount factor, and $Q'(\cdot)$ is a separate network that is only updated periodically for generating the targets value y_i. In each iteration, we improve $Q(\cdot)$ by using mini-batch deep Q-learning. We can use several optimization algorithms such as Adam (Kingma and Ba, 2014), Stochastic gradient descent (Sutskever et al., 2013) and RMSprop (Ruder, 2016) to train the deep Q network.

During the indirect reinforcement learning, also known as planning, the dialogue agent improves its dialogue policy by interacting with the world model rather than the real user to reduce the training cost. The frequency of planning is controlled by the parameter K, which means that the planning is performed K steps per step of the direct reinforcement learning. The value of K tends to be large when the world model is able to capture the feature of the real environment accurately. In each step of planning, the world model responses a_w^u based on the current environment. As mentioned in the last subsection, the experience (s, a, r, a_w^u, t') generated during planning will be evaluated by the KL divergence inspection before pushing it into the replay buffer M^p to ensure the quality of experiences.

Algorithm 2 gives the whole process of our proposed approach. Each epoch of dialogue policy learning consists of direct reinforcement learning, controlled planning, and world model learning.

3 Experiment

To illustrate the effectiveness and superiority of our method, we test it in the movie ticket booking task, and compare it with the other methods from two aspects : 1) the change of performance in different hyperparameters; 2) the performance comparison. The source codes and the implementation details are packed in the supplementary materials for reproduction.

3.1 Dataset

We use the same raw data as original DDQ method. It is collected via Amazon Mechanical Turk. The dataset has been manually labeled based on a schema defined by domain experts, which consists of 11 dialogue acts and 16 slots (Peng et al., 2018). In total, the dataset contains 280 annotated dialogues, the average length of which is approximately 11 turns.

3.2 Dialogue Agents for Comparison

We develop different versions of task-completion dialogue agents to benchmark the performance of our proposed method and its variants.

- The **GPDDQ(M, K, N)** agents are learned by our GPDDQ method, where M is the buffer size, K is the number of planning steps and N is the batch size. The initial world model is pre-trained on human conversational data. Note that we **do not** utilize uncertainty attribute and KL divergence inspection in this agent.
- The **UN-GPDDQ(M, K, N)** agents are very similar to GPDDQ(M, K, N) agents, but the uncertainty is considered in this case. Currently, $\boldsymbol{e}_l$, $\boldsymbol{e}$ and $\boldsymbol{e}_b$ are returned in the world model prediction stage.
- The **KL-GPDDQ(M, K, N)** agents are the same to the UN-GPDDQ(M, K, N) agents, except that the KL divergence inspection is also considered.
- The **GPDDQ(M, K, N, rand-init θ_w)** agents are learned by the GPDDQ method with a randomly initialized world model. The reward r and terminal variable t are randomly sampled from their corresponding GP models. And for action a^u, we uniformly sample it from its defined action domain.
- The **GPDDQ(M, K, N, fixed θ_w)** agents are only refined during warm-up stage on human conversational data. After that, the world model will not be trained any more.
- The **GPDQN(M, K, N)** agents are learned by direct reinforcement learning. Its performance can be viewed as the upper bound of its GPDDQ(M, K, N) counterpart, assuming that the world model perfectly matches real users.
- For other agents which are not mentioned above, please refer to (Peng et al., 2018; Su et al., 2018).

3.3 Parameter Analysis

To illustrate the advantages of our model in terms of sensitivity to hyperparameter changing, we conduct a series of experiments by changing the corresponding parameters such as batch size, planning step, parameter update policy, and buffer size.

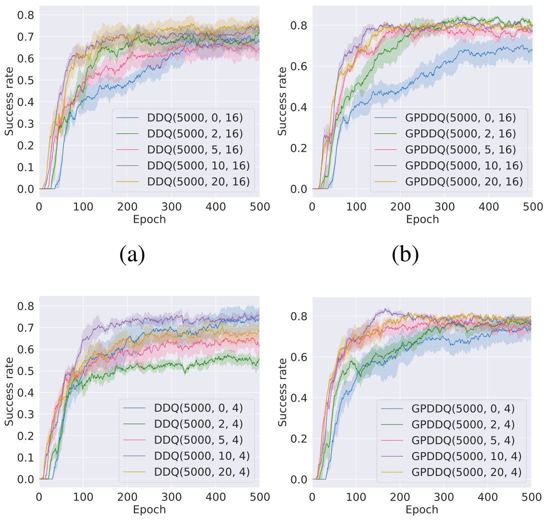

(a) (b)

(c) (d)

Figure 4: Learning curve for DDQ and GPDDQ with different parameter settings: $M = 5000$, $N = 4, 16$ and $K = 0, 2, 5, 10, 20$.

3.3.1 Batch Size and Planning Step

In this group of experiments, we use the 16 and 4 as the batch size to train the policy network $Q(\cdot)$ and world model $W(\cdot)$ with different planning steps K. The main results are shown in Figure 4 which indicates that the GPDDQ agents consistently outperform the DDQ agents in a statistical sense. In Figure 4 (a) and (b), it can be found that the convergence value of the success rate of GPDDQ agent is much better than that of DDQ agent with the same planning step K. The converged success rate oscillates around 0.8 in our proposed method, however, the corresponding value is about 0.74 in the DDQ method. As the planning steps increase, the learning speeds generally become faster. This phenomenon is consistent with intuition that a large planning step brings faster learning speed. Nonetheless, we can notice that there is no significant difference between the learning curve of $K = 20$ and $K = 10$. This is due to the reduction of the quality of simulated experiences caused by a too large K. To find the optimal value of K, the trade-off between the amount of simulated experience and the quality of simulated experience should be considered seriously.

Since GP method is more robust to hyperparameters (Kuss, 2006), we speculate that it still has better performance with a small batch size. In Figure 4 (c) and (d), we shrink our batch size to 4, and keep the other parameters the same as previ-

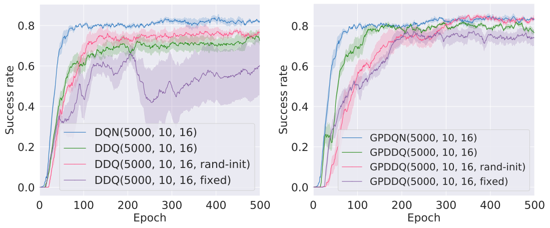

(a) (b)

Figure 5: Learning curve for DDQ and GPDDQ with $M = 5000$, $K = 10$ and $N = 16$, but with different parameter update strategies.

ous experiments. For GPDDQs with $K > 0$, their performances still outperform the DDQ(5000, 0, 4) agent. Moreover, compared to the results when batch size is 16, there is no obvious performance degradation. Besides, since the matrix inversion operation costs more time during training when the batch size is large, the training time consumption can be greatly reduced if the batch size becomes smaller. On the contrary, for DDQ methods, only when $K = 10$, the learning curve is better than DDQ(5000, 0, 4) method in terms of the stable success rate. When increasing the planning step to $K = 20$, its performance degrades dramatically. This may be caused by the insufficient training of DNN when the batch size is too small.

3.3.2 Parameter Update Policy

In this group of experiments, we set $M = 5000$, $K = 10$, $N = 16$, and change its parameter update policy. The results are given in Figure 5. The results indicate that the quality of the world model has a significant impact on the performance of these agents. The DQN and GPDQN method is the completely model free method with K times training data larger than other methods in Figure 5. Due to randomness, the two rising curves are slightly different, but basically the same. Obviously, if the world model is fixed after the warm-up stage, it will produce the worst results. The huge drop in the learning curve of DDQ(5000, 10, 16, fixed) at about 250 epoch may be the result of insufficient training data. For each learning curve of GPDDQ, the proposed GPDDQ method can achieve almost the same maximum value as DQN. In addition, the final succees rates of GPDDQ are always larger than those of DDQ methods. Even if we use different parameter update policies, the final success rates do not fluctuate too much.

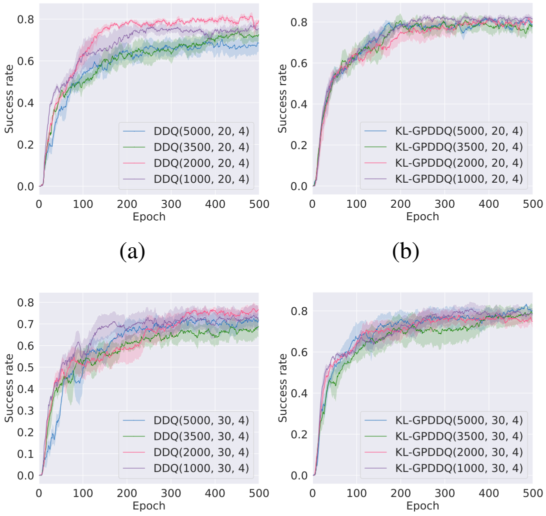

(a) (b)

(c) (d)

Figure 6: Learning curve for DDQ and GPDDQ with different parameter settings: $M = 5000, 3500, 2000, 1000$, $K = 20, 30$ and $N = 4$.

3.3.3 Buffer Size

In this subsection, we evaluate our KL-GPDDQ method, ignoring the other simplified methods, by changing the buffer size. From the perspective of overall performance shown in Figure 6, our proposed method is more stable in different conditions, i.e., different buffer sizes and planning steps. After reducing the buffer size from 5000 to 1000, the learning curve does not change much in our methods. However, for DDQ methods, their performances are still poor. These phenomena make us suspect that the world model in DDQ, built by DNN, may generate many low-quality experiences during planning. Nevertheless, when the buffer size becomes smaller, high-quality experiences become the dominant part of the replay buffer. In terms of convergence, the success rate of KL-GPDDQ method stabilizes around 0.8 after 200 epochs when planning step is 20, and slightly smaller when $K = 30$. On the contrary, the DDQ methods do not converge after 200 epochs. Their success rates are basically lower than those of our proposed methods when converging. This result argues that our method can achieve better and more robust performance with relatively small buffer sizes.

3.4 Performance Comparison

To demonstrate the performance of our proposed method, we compare it with other baseline algorithms. We can find from Table 1 that the DDQ

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
D3Q(10)*	.6333	28.99	**16.01**	.7000	37.24	**15.52**	.6667	33.09	**15.83**
DDQ(5000, 20, 4)	**.5379**	**12.60**	25.90	**.6466**	**26.79**	**23.60**	**.6612**	**29.14**	**22.41**
GPDDQ(5000, 20, 4)	**.7069**	**35.09**	21.48	.7706	43.65	19.60	.7874	45.72	19.54
UN-GPDDQ(5000, 20, 4)	.5800	17.61	**25.98**	.7050	34.32	22.57	.7726	43.84	19.75
KL-GPDDQ(5000, 20, 4)	.6138	22.57	24.17	**.7915**	**46.39**	19.16	**.7985**	**47.34**	18.97

*The result for D3Q method is borrowed from its original paper (Su et al., 2018).

Table 1: Results of different agents with buffer size 5000 at training epoch = {100, 200, 300}. The best and the worst agents for planning step $K = 20$ are highlighted in blue and red, respectively. (Success: success rate, Turns: dialogue turns)

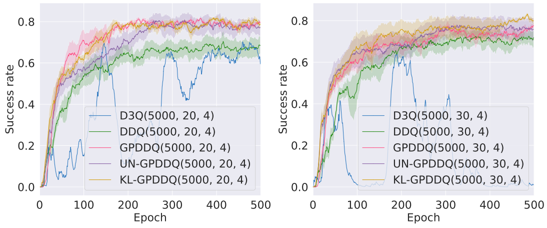

(a)

(b)

Figure 7: Learning curve for DDQ, GPDDQ, UN-GPDDQ and KL-GPDDQ agent with $M = 5000$, $K = 20, 30$ and $N = 4$.

methods are still the worst among the five methods. Due to its extremely large training time consumption and high sensitivity, for D3Q method, we only calculate it once in Figure 7 and borrow its performance from its original paper (Su et al., 2018) in Table 1. From the results of GPDDQ, UN-GPDDQ, and KL-GPDDQ agents, it is obvious that the KL divergence inspection we design is helpful for performance improvement, which can be concluded based on the clear increase of success rate and reward shown in Table 1. Compared with DDQ, our proposed method can improve the success rate by about 20% with fewer user interactions.

Figure 7 shows that the learning speeds of our proposed methods are much faster than those of DDQ and D3Q. It should be noted that the learning curve of D3Q vibrates violently. Especially, when $K = 30$, D3Q even cannot converge to the optimal value. Although D3Q can discriminate low-quality experiences, it is very hard to implement D3Q in reality due to the instability of GANs.

4 Related Work

Most of the works on task-completion dialogue policy learning focus on how to use fewer conversation rounds to complete a specific task (Lu et al., 2019). There are four typical methods, including rule based method (Weizenbaum, 1966), retrieval based method (Mikolov et al., 2013; Pennington et al., 2014; Serban et al., 2017), supervised learning based method (Sukhbaatar et al., 2015; Weston et al., 2016), and reinforcement learning based method (Levin et al., 2002). Since the reinforcement learning based method can fine-tune the current dialogue strategy based on users' feedback to promote user satisfaction, it has been the mainstream of dialogue policy learning method in recent years (Chen et al., 2020; Saha et al., 2020; Li et al., 2020).

However, the vanilla RL methods require prohibitively many rounds of human-machine dialogue interactions before getting a usable dialogue policy. Deep Dyna-Q (DDQ) (Peng et al., 2018) is proposed based on the Dyna-Q framework which introduces an environment model, known as *world model*, to generate simulated user experiences in the dynamic environment to decrease the heman-machine interactions. Based on (Peng et al., 2018), (Su et al., 2018) attempted to control the quality of simulated experiences by using GANs to discriminate the low-quality experiences. (Zhao et al., 2020) proposed a method called DR-D3Q to learn policies in noise robustly by combining dynamic reward and Dueling DQN. Based on human demonstrations, (Wang et al., 2020) presented how to efficiently learn dialogue policy through policy shaping and reward shaping, in which the world model is replaced by an imitation model.

5 Conclusion

In this paper, we propose a Gaussian Process based Deep Dyna-Q approach. The world model is built as a GP model, and a novel KL divergence based discriminator is designed to evaluate simulated experiences. Extensive experiments demonstrate the superiority of our proposed method thanks to the newly-designed world model and discriminator. Compared with existing DDQ framework based methods, both efficiency and robustness are promoted by our proposed method. With this satisfactory result, it is potential to develop more valuable algorithms based on our method. In the future work, we will try to incorporate other strategy, such as tree-based search algorithms (Schrittwieser et al., 2020), to further improve the learning performance.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant 62002369, the Scientific Research Project of National University of Defense Technology under grant ZK19-03, Key-Area Research and Development Program of Guangdong Province (Grant No. 2019B010155003).

References

Zhi Chen, Lu Chen, Xiang Zhou, and Kai Yu. 2020. Deep reinforcement learning for on-line dialogue state tracking. *arXiv preprint arXiv:2009.10321*.

David Duvenaud. 2014. *Automatic model construction with Gaussian processes*. Ph.D. thesis, University of Cambridge.

Milica Gašić, Nikola Mrkšić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2017. Dialogue manager domain adaptation using gaussian process reinforcement learning. *Computer Speech & Language*, 45:552–569.

James Hensman, Nicolas Durrande, and Arno Solin. 2017. Variational fourier features for gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Malte Kuss. 2006. *Gaussian process models for robust regression, classification, and reinforcement learning*. Ph.D. thesis, echnische Universität Darmstadt Darmstadt, Germany.

E. Levin, R. Pieraccini, and W. Eckert. 2002. Learning dialogue strategies within the markov decision process framework. In *IEEE Workshop on Automatic Speech Recognition Understanding*.

Ziming Li, Julia Kiseleva, and Maarten de Rijke. 2020. Rethinking supervised learning and reinforcement learning in task-oriented dialogue systems. *arXiv preprint arXiv:2009.09781*.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 3.

Keting Lu, Shiqi Zhang, and Xiaoping Chen. 2019. Goal-oriented dialogue policy learning from failures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2596–2603.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos Storkey. 2019. Deep kernel transfer in gaussian processes for few-shot learning. *arXiv preprint arXiv:1910.05199*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*.

Fiana Raiber and Oren Kurland. 2017. Kullback-leibler divergence revisited. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 117–124.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Tulika Saha, Sriparna Saha, and Pushpak Bhattacharyya. 2020. Towards sentiment aided dialogue policy learning for multi-intent conversations using hierarchical reinforcement learning. *PloS one*, 15(7):e0235367.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3295–3301. AAAI Press.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2431–2441.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3813–3823.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2440–2448, Cambridge, MA, USA. MIT Press.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Huimin Wang, Baolin Peng, and Kam Fai Wong. 2020. Learning efficient dialogue policy from demonstrations through shaping. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Christopher KI Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Yangyang Zhao, Zhenyu Wang, Kai Yin, Rui Zhang, Zhenhua Huang, and Pei Wang. 2020. Dynamic reward-based dueling deep dyna-q: Robust policy learning in noisy environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:9676–9684.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

A Implementation Details

We implement our experiment on Thinkstation-P520 with Intel W-2223 CPU, 64G memory and two Nvidia GeForce RTX 2080 cards. And the average runtime for each DDQ and GPDDQQ approach are about from 2 to 3 hours and from 3 to 4.5 hours, respectively. For D3Q method, it takes about 2 days to run. The policy network $Q(s, a; \theta_Q)$ of direct reinforcement leaning in these models are approximated by deep neural network with `tanh` activations. It has one hidden layer with 80 hidden nodes. And the discount factor γ introduced in loss function is set to be 0.9. In our each GP model, there are 4 parameters need to be optimized. We limit the maximum length of a simulated dialogue to 40. In all our experiments, we only train the dialogue agents by interacting with user simulator which is publicly available. Only if the movie ticket is successfully booked and the information provided by the agent satisfies the constraints, the dialogue is considered successfully. If the dialogue is successful, the agent receives a positive reward of $2L$, otherwise, the reward value will be $-L$, where L is the defined maximum number of dialogue turns. Furthermore, shorter conversations are encouraged in this dialogue system since the agent will receive a reward of -1 per round. If there are no other instructions, in order to eliminate errors, each experiment are conducted five times to average.

CITEWORTH: Cite-Worthiness Detection for Improved Scientific Document Understanding

Dustin Wright and **Isabelle Augenstein**
Dept. of Computer Science
University of Copenhagen
Denmark
`{dw|augenstein}@di.ku.dk`

Abstract

Scientific document understanding is challenging as the data is highly domain specific and diverse. However, datasets for tasks with scientific text require expensive manual annotation and tend to be small and limited to only one or a few fields. At the same time, scientific documents contain many potential training signals, such as citations, which can be used to build large labelled datasets. Given this, we present an in-depth study of cite-worthiness detection in English, where a sentence is labelled for whether or not it cites an external source. To accomplish this, we introduce CITEWORTH, a large, contextualized, rigorously cleaned labelled dataset for cite-worthiness detection built from a massive corpus of extracted plain-text scientific documents. We show that CITEWORTH is high-quality, challenging, and suitable for studying problems such as domain adaptation. Our best performing cite-worthiness detection model is a paragraph-level contextualized sentence labelling model based on Longformer, exhibiting a 5 F1 point improvement over SciBERT which considers only individual sentences. Finally, we demonstrate that language model fine-tuning with cite-worthiness as a secondary task leads to improved performance on downstream scientific document understanding tasks.

1 Introduction

Building effective NLP systems from scientific text is challenging due to the highly domain-specific and diverse nature of scientific language, and a lack of abundant sources of labelled data to capture this. While large scale repositories of extracted, structured, and unlabelled plain-text scientific documents have recently been introduced (Lo et al., 2020), most datasets for downstream tasks such as named entity recognition (Li et al., 2016) and citation intent classification (Cohan et al., 2019) remain limited in size and highly domain specific. This begs the question: what useful training signals can be automatically extracted from massive unlabelled scientific text corpora to help improve systems for scientific document processing?

Scientific documents contain much inherent structure (sections, tables, equations, citations, etc.), which can facilitate creating large labelled datasets. Some recent examples include using paper field (Beltagy et al., 2019), the section to which a sentence belongs (Cohan et al., 2019), and the cite-worthiness of a sentence (Cohan et al., 2019; Sugiyama et al., 2010) as a training signal.

Cite-worthiness detection is the task of identifying *citing sentences*, i.e. sentences which contain a reference to an external source. It has useful applications, such as in assistive document editing, and as a first step in citation recommendation (Färber et al., 2018b). In addition, cite-worthiness has been shown to be useful in helping to improve the ability of models to learn other tasks (Cohan et al., 2019). We also hypothesize that there is a strong domain shift between how different fields use citations, and that such a dataset is useful for studying domain adaptation problems with scientific text.

However, constructing such a dataset to be of high quality is surprisingly non-trivial. Building a dataset for cite-worthiness detection involves extracting sentences from a scientific document, labelling whether each sentence contains a citation, and removing all citation markers. As a form of distant supervision, this naturally comes with the hazard of adding spurious correlations, such as poorly removed citation text causing ungrammatical sentences and hanging punctuation, which can trivially indicate a cite-worthy or non-cite-worthy sentence. Additionally, the task itself is quite difficult to learn, as different fields employ citations differently, and whether or not a sentence contains a citation depends on factors such as the context in which it

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1796–1807
August 1–6, 2021.

appears. Given this, we present CITEWORTH, a rigorously curated dataset for cite-worthiness detection in English. CITEWORTH contains rich metadata, such as authors and links to cited papers, and all data is provided in *full paragraphs*: every sentence in a paragraph is labelled in order to provide sentence *context*. We offer the dataset to the research community to facilitate further research on cite-worthiness detection and related scientific document processing tasks.

Using CITEWORTH, we ask the following primary research questions:

> **RQ1**: How can a dataset for cite-worthiness detection be automatically curated with low noise (§3)?
>
> **RQ2**: What methods are most effective for automatically detecting cite-worthy sentences (§4)?
>
> **RQ3**: How does domain affect learning cite-worthiness detection (§5)?
>
> **RQ4**: Can large scale cite-worthiness data be used to perform transfer learning to downstream scientific text tasks (§6)?

We demonstrate that CITEWORTH is of high quality through a manual evaluation, that there are large differences in how models generalize to data from different fields, and that sentence context leads to significant performance improvements on cite-worthiness detection. Additionally, we find that cite-worthiness is a useful task for transferring to downstream scientific text tasks, in particular citation intent classification, for which we offer performance improvements over the current state-of-the-art model SciBERT (Beltagy et al., 2019).

In sum, our **contributions** are as follows:

- CITEWORTH, a dataset of 1.2M rigorously cleaned sentences from scientific papers labelled for cite-worthiness, balanced across 10 diverse scientific fields.
- A method for cite-worthiness detection which considers the entire paragraph a sentence resides in, improving by 5 F1 points over the state of the art model for scientific document processing, SciBERT (Beltagy et al., 2019).
- A thorough analysis of the problem of cite-worthiness detection, including explanations of predictions and insight into how scientific domain affects performance.
- New state of the art on citation intent detection via transfer learning from joint citation detection and language model fine-tuning on CITEWORTH, with improved performance over SciBERT on several other tasks.

2 Related Work

2.1 Cite-Worthiness Detection

Cite-worthiness detection is the task of identifying *citing sentences*, i.e. sentences which contain a reference to an external source. The reasons for citing are varied, e.g. to give credit to existing ideas or to provide evidence for a claim being made. Sugiyama et al. (2010) perform cite-worthiness detection using SVMs with features such as unigrams, bigrams, presence of proper nouns, and the classification of previous and next sentences. They create a dataset from the ACL Anthology Reference corpus (ACL-ARC, Bird et al. (2008)), using heuristics to remove citation markers. Färber et al. (2018b) document the performance of convolutional recurrent neural nets on a larger set of three datasets coming from ACL-ARC, arXiv CS (Färber et al., 2018a), and Scholarly Dataset 2.[1] Datasets from these studies suffer from high class imbalance, are limited to only one or a few domains, and little analysis of the datasets is performed to understand the quality of the data or what aspects of the problem are difficult or easy. Additionally, no study to date has considered how sentence context can affect learning to perform cite-worthiness detection.

In addition to being a useful task in itself, cite-worthiness detection is useful for other tasks in scientific document understanding. In particular, it has been shown to help improve performance on the closely related task of citation intent classification (Jürgens et al., 2018) when used as an auxiliary task in a multi-task setup (Cohan et al., 2019). However, cite-worthiness detection has not been studied in a transfer learning setup as a pre-training task for multiple scientific text problems. In this work, we seek to understand to what extent cite-worthiness detection is a transferable task.

Scientific Document Understanding Numerous problems related to scientific document understanding have been studied previously. Popular tasks include named entity recognition (Li et al., 2016; Kim et al., 2004; Doğan et al., 2014;

---

[1] `http://www.comp.nus.edu.sg/~sugiyama/SchPaperRecData.html`

Luan et al., 2018) and linking (Wright et al., 2019), keyphrase extraction (Augenstein et al., 2017; Augenstein and Søgaard, 2017), relation extraction (Kringelum et al., 2016; Luan et al., 2018), dependency parsing (Kim et al., 2003), citation prediction (Holm et al., 2020), citation intent classification (Jürgens et al., 2018; Cohan et al., 2019), summarization (Collins et al., 2017), and fact checking (Wadden et al., 2020).

Datasets for scientific document understanding tasks tend to be limited in size and restricted to only one or a few fields, making it difficult to build models with which one can study cross-domain performance and domain adaptation. Here, we curate a large dataset of cite-worthy sentences spanning 10 different fields, showing that such data is both useful for studying domain adaptation and for transferring to related downstream scientific document understanding tasks.

3 RQ1: CITEWORTH Dataset Construction

The first research question we ask is: How can a dataset for cite-worthiness detection be automatically curated with low noise? To answer this, we start with the S2ORC dataset of extracted plain-text scientific articles (Lo et al., 2020). It consists of data from 81.1M English scientific articles, with full structured text for 8.1M articles. S2ORC uses SCIENCEPARSE[2] to parse PDF documents and GROBID[3] to extract structured data from text. As such, the data also includes rich metadata, e.g. Microsoft Academic Graph (MAG) categories, linked citations, and linked figures and tables. Throughout this work, a "citation span" denotes a span containing citation text (e.g. "[2]"), and a "citation marker" is any text that trivially indicates a citation, such as the phrase "is shown in." A citation span is also a type of citation marker. It is important to remove all citation markers from the dataset to prevent the model learning to use these signals for prediction.

3.1 Data Filtering

Given the size of S2ORC, we first reduce the candidate set of data to papers where all of the following are available.

- Abstract
- Body text
- Bibliography
- Tables and figures
- Venue information
- Inbound citations
- Microsoft Academic Graph categories

Filtering based on these criteria results in 5,494,387 candidate papers from which to construct the dataset. After filtering the candidate set of papers, we perform the following checks on the sentences in the body text.

1. Citation spans are parenthetical author-year or bracketed-numerical form.
2. Citation spans are at the end of a sentence.
3. All possible citation spans have been extracted by S2ORC.
4. No citation markers are left behind after removing citation spans from the text.
5. Sentence starts with a capital letter, ends with '.', '!', or '?', and is at least 20 characters long.

The detailed steps of extracting and labelling sentences based on these criteria are given in §3.2. With the first two criteria, we restrict the scope of cite-worthy sentences to being only those whose citation span comes at the end of a sentence, and whose citation format is parenthetical author-year form or bracketed-numerical form. In other words, cite-worthy sentences in our data are constrained to those of the following forms.

> This result has been shown in previous work (Author1 et al., ####, ...).
>
> This result has been shown in previous work [#-#].

In this, we ignore citation sentences which contain inline citations, such as "The work of Authors et al. (####) has shown this in previous work", as well as any sentence with a citation format that does not match the two we have selected.

Curating cite-worthy sentences as such helps prevent spurious correlations in the data. Removing citations in the middle of a sentence runs the risk of rendering the sentence ungrammatical (for example, the above sample would turn into "The work *of has* shown this in previous work"), providing a signal to machine learning models. While there are cases where inline citations could potentially be removed in their entirety and not destroy the sentence structure, this is beyond the scope of this paper and left to future work.

---

[2] `https://github.com/allenai/scienceparse`

[3] `https://github.com/kermitt2/grobid`

Biology
Wood Frogs (Rana sylvatica) are a charismatic species of frog common in much of North America. They breed in explosive choruses over a few nights in late winter to early spring. *The incidence in Wood Frogs was associated with a die-off of frogs during the breeding chorus in the Sylamore District of the Ozark National Forest in Arkansas ~~(Trauth et al., 2000)~~.*

Computer Science
Land use or cover change is a direct reflection of human activity, such as land use, urban expansion, and architectural planning, on the earth's surface caused by urbanization ~~[1]~~. Remote sensing images are important data sources that can efficiently detect land changes. *Meanwhile, remote sensing image-based change detection is the change identification of surficial objects or geographic phenomena through the remote observation of two or more different phases ~~[2]~~.*

Table 1: Excerpts from training samples in CITEWORTH from the Biology and Computer Science fields. Green sentences are cite-worthy sentences, from which citation markers are removed during dataset construction.

3.2 Extracting Cite-Worthy Sentences in Context

As we are interested in using sentence context for prediction, we perform extraction at the *paragraph level*, ensuring that all of the sentences in a given paragraph meet the checks given in §3.1. As such, our dataset construction pipeline for a given paper begins by first extracting all paragraphs from the body text which belong to sections with titles coming from a constrained list of permissible titles (e.g. "Introduction," "Methods," "Discussion") . The full list is provided in Appendix A.

For a given paragraph, we first word and sentence tokenize the text with SciSpacy (Neumann et al., 2019). Each sentence is then checked for containing citations using the provided citation spans in the S2ORC dataset. In some cases, the sentence contains citations which were missed by S2ORC; these are checked using regular expressions (see Appendix B). If a match is found the paragraph is ignored, as we only consider paragraphs where all citations have been extracted by S2ORC. Otherwise, the location and format of the citation is checked, again using regular expressions (see Appendix B). If the citation is not at the end of the sentence, the paragraph is ignored. We then remove the citation text using the provided citation spans for all sentences which pass the above checks.

Simply removing the citation span runs the risk of leaving other types of citation markers, such as hanging punctuation and prepositional phrases e.g. "This was shown by the work of ~~Author et al. (####)~~." To mitigate this, we remove all hanging punctuation at the end of a sentence that is not a period, exclamation point, or question mark, and check for possible hanging citations using the regular expression provided in Appendix B. The regular expression checks for many common prepositional phrases and citation markers occurring as the last phrase of a sentence such as "see," "of," "by," etc.

Metric	#
Total sentences	1,181,793
Total number of tokens	34,170,708
Train sentences	945,426
Dev sentences	118,182
Test sentences	118,185
Total cite-worthy	375,388 (31.76%)
Total non-cite-worthy	806,405 (68.24%)
Min char length	21
Max char length	1,447
Average char length	152
Median char length	142

Table 2: Various statistics of the CITEWORTH dataset.

To handle issues with sentence tokenization, we also ensure that the first character of each sentence is a capital letter, and that the sentence ends with a period, exclamation point, or question mark. If all criteria are met for all sentences in a paragraph, the paragraph is added to the dataset. Finally, we build a dataset which is diverse across domains by evenly sampling paragraphs from the following 10 MAG categories, ensuring that each paragraph belongs to exactly one category: Biology, Medicine, Engineering, Chemistry, Psychology, Computer Science, Materials Science, Economics, Mathematics, and Physics. Example excerpts from the dataset are presented in Table 1, and the statistics for the final dataset are given in Table 2.[4]

3.3 Manual Evaluation

In order to provide some measure of the general quality of CITEWORTH, we perform a manual eval-

---

[4] The full dataset can be downloaded from this repository: `https://github.com/copenlu/cite-worth`

Method	Extracted Correct	Markers Removed
Baseline	92.07	92.78
Ours	**98.90**	**98.10**

Table 3: Results of manually annotating 1000 random sentences (per method) from CITEWORTH and a naive baseline which only removes citations based on provided citation spans . "Extracted Correct" are results for correctly extracting the sentences (i.e. that sentences are tokenized correctly and are grammatical), and "Markers Removed" are results for successfully removing citation markers. The data curated using our method has 6% fewer errors in terms of extraction and removal of citation markers, and less than 2% of the samples have some form of citation marker.

uation of a sample of the data. We annotate the data for whether or not citation markers are completely removed, and for whether or not the sentences are well-formed, containing no obvious extraction artifacts. We sample 500 cite-worthy sentences and 500 non-cite-worthy sentences randomly from the data. Additionally, we compare to a baseline where the only heuristic used is to remove citation spans based on the provided spans in the S2ORC dataset. We again sample 500 cite-worthy and 500 non-cite-worthy sentences for annotation. The two sets are shuffled together and given to an independent expert annotator with a PhD in computer science for labelling. The annotator is instructed to label if the sentences are complete and have no hanging punctuation or obvious extraction errors, and if there are any textual indicators that the sentences contain a citation. The results for the manual annotation can be seen in Table 3.

We see that the CITEWORTH data are of a much higher quality than removing citation markers based only on the citation spans. Overall, our heuristics improve on extraction quality by 6.83% absolute and on removing markers of citations by 5.32% absolute. This results in 1.1% of the sample data containing sentence cleaning issues, and 1.9% having trivial markers indicating a citation is present. We argue that this is a strong indicator of the quality of the data for supervised learning.

4 RQ2: System Evaluation[5]

Next, we ask: what methods are most effective for performing cite-worthiness detection? To answer this and characterize the difficulty of the problem, we run a variety of baseline models on CITEWORTH. The hyperparameters selected for each model, as well as hyperparameter sweep information, are given in Appendix C.6.

Logistic Regression As a simple baseline, we use a logistic regression model with TF-IDF input features.

Färber et al. (2018b) The convolutional recurrent neural network (CRNN) model from Färber et al. (2018b). They additionally use oversampling to deal with class imbalance.

Transformer We additionally train a Transformer model from scratch (Vaswani et al., 2017), tuning the model hyperparameters on a subset of the training data via randomized grid search.

BERT We use a pretrained BERT model (Devlin et al., 2019) due to the strong performance of large pretrained Transformer models on downstream tasks.

SciBERT SciBERT (Beltagy et al., 2019) is a BERT model pretrained on a large corpus of scientific text from Semantic Scholar (Ammar et al., 2018), and is therefore potentially better suited to fine-tuning on scientific cite-worthiness detection.

SciBERT + PU Learning We experiment with SciBERT trained using positive-unlabelled (PU) learning (Elkan and Noto, 2008) which has been shown to significantly improve performance on citation needed detection in Wikipedia and rumour detection on Twitter (Wright and Augenstein, 2020a). The intuition behind PU learning is to assume that cite-worthy data is labelled and non-cite-worthy data is unlabelled, containing some cite-worthy examples. This is to mitigate the subjectivity involved in adding citations to sentences. Technically, this involves training a classifier on the positive-unlabeled data which will predict the probability that a sample is labeled, and using this to estimate the probability that a sample is positive given that it is unlabeled. One then trains a second model where positive samples are trained on normally and unlabeled samples are duplicated and trained on twice, once as positive and once as negative data, weighed by the first model's estimate of the probability that the sample is positive.

Longformer-Ctx Finally, we test our novel contextualized prediction model based on Longformer (Beltagy et al., 2020). Longformer is a

---

[5] The code for all experiments can be found here: `https://github.com/copenlu/cite-worth`

Method	P	R	F1
Logistic Regression	$46.65_{0.00}$	$64.88_{0.00}$	$54.28_{0.00}$
Färber et al. (2018b)	$49.57_{0.96}$	$65.56_{2.61}$	$56.41_{0.34}$
Transformer	$47.92_{0.78}$	$71.59_{1.74}$	$57.39_{0.10}$
BERT	$55.04_{0.66}$	$69.02_{1.33}$	$61.23_{0.21}$
SciBERT-no-weight	$\mathbf{65.94}_{0.37}$	$51.62_{0.53}$	$57.91_{0.30}$
SciBERT	$57.03_{0.50}$	$68.08_{1.03}$	$62.06_{0.15}$
SciBERT + PU	$49.46_{0.83}$	$\mathbf{82.12}_{1.40}$	$61.73_{0.27}$
Longformer-Solo	$57.21_{0.25}$	$68.00_{0.41}$	$62.14_{0.02}$
Longformer-Ctx	$59.92_{0.28}$	$77.15_{0.49}$	$\mathbf{67.45}_{0.06}$

Table 4: F1 performance of baselines on the test set of CITEWORTH. Results are averaged across 5 seeds, with standard deviations given in the subscripts.

Transformer based language model which uses a sparse attention mechanism to scale better to longer documents. We process an entire paragraph at a time, separating each sentence with a `[SEP]` token. Each `[SEP]` token representation at the output of Longformer is then passed through a network with one hidden layer and a classifier. As a control, we also experiment with Longformer using only single sentences as input (Longformer-Solo).

Due to the imbalance in the distribution of classes, the loss for each of the models is weighted. For comparison, we include results for SciBERT without weighting the loss function. The results for our baseline models on the test set of the dataset are given in Table 4.

Our results indicate that context is critical, resulting in the best F1 score of 67.45 (Longformer-Ctx) and a 5.31 point improvement over the next best model. Using class weighting is also highly important, resulting in another increase of over 4 F1 points. Compared to not using class weights, PU learning performs significantly better, and leads to the highest recall of all models under test. Additionally, language model pre-training is useful, as BERT, SciBERT, and Longformer all perform significantly better than a Transformer trained from scratch and the model from Färber et al. (2018b).

To gain some insight into what the model learns, we visualize the most salient features from SciBERT for selected easy and hard examples. We use the single-sentence model instead of the paragraph model for simplicity. "Easy" samples are defined as those which the model predicted correctly with high confidence, and "hard" examples are defined as those for which the model had low confidence in its prediction. We use the InputXGradient method (Kindermans et al., 2016), specifically the variant using L2 normalization over neurons to get a pre-embedding score, as it has been recently shown to have the best overall agreement with human rationales versus several other explainability techniques (Atanasova et al., 2020). The method works by calculating the gradient of the output with respect to the input, then multiplies this with the input. In the examples below "C" refers to an example whose gold label is cite-worthy, and "N" refers to an example whose gold label is non-cite-worthy.

The model is able to pick up on obvious markers of cite-worthy and non-cite-worthy sentences for the following correctly classified examples, such as that a sentence refers to a preprint or to different sections within the paper itself:

C: [CLS] in this note , we follow the approach to the en ##och ##s conjecture outlined in the preprint . [SEP]

N: [CLS] conclusions are provided in section 4 . [SEP]

We also see that the dataset contains many relatively difficult instances, as we show in the following incorrectly classified examples. E.g., the model observes "briefly discussed" as an indicator that an instance is non-cite-worthy when it is in fact cite-worthy, and that "described earlier" and "previous work" signal that a sentence is cite-worthy when it is in fact labelled as non-cite-worthy.

C: [CLS] some approaches for the solution as well as their limitations are briefly discussed . [SEP]

N: [CLS] this simple and fast technique for the production of snps was described earlier in our previous work . [SEP]

We hypothesize that in such instances, context can help the most in disambiguating which sentences in a paragraph should be labelled as cite-worthy. Additionally, other information such as the section in which a sentence resides could help. E.g., to correctly label the fourth statement above as "non-cite-worthy", it may help to see that the last sentence of the paragraph is "In our previously published work, it was reported that SNPs were joined together by the heat treatment, and this process led to increase in the sizes of SNPs which finally resulted in sharper XRD peaks" which is a cite-worthy sentence. Additionally, it may help to know that it resides in the "Discussion" section of the paper.

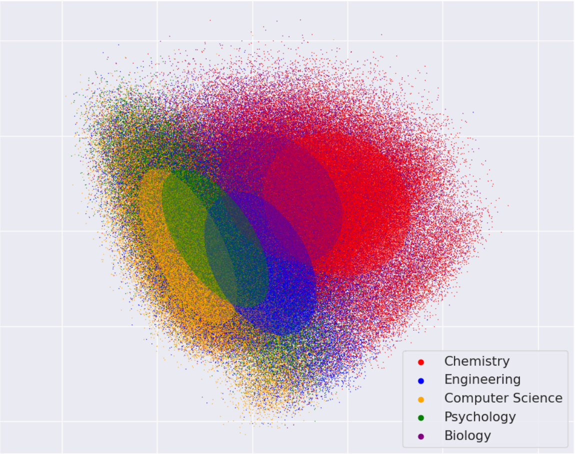

Figure 1: Visualizing the BERT embeddings for 5 of the 10 domains from CITEWORTH using the method by Aharoni and Goldberg (2020). Clustering is performed using Gaussian Mixture Models.

5 RQ3: Domain Evaluation

We next ask: how does domain affect learning to perform cite-worthiness? To answer this, we study the relationships between cite-worthiness data from different fields and how the Longformer-Ctx model performs in a cross-domain setup. For ease of analysis we limit the scope of fields to 5 of the 10 fields in the dataset: Chemistry, Engineering, Computer Science, Psychology, and Biology.

First, we visualize the embedding space for data from each of these domains using the method of Aharoni and Goldberg (2020). In this, the data is passed through BERT (specifically the base, uncased variant) and the output representations for each token in a sentence are average pooled. These representations are visualized in 2D space via PCA in Figure 1. It is clear that similar fields occupy closer space, with 'engineering' and 'computer science' sharing closer representations, as well as 'biology' and 'chemistry'. We perform clustering on this data using a Gaussian mixture model similarly to Aharoni and Goldberg (2020), finding that domains form somewhat distinct clusters with a cluster purity of 57.61. This demonstrates that the data in different fields are drawn from different distributions, thus differences could exist in a model's ability to perform cite-worthiness detection on out of domain data.

To test this, we perform a cross-validation experiment using the 5 selected fields, training on one field and testing on another for all 25 combinations. The results for the 5x5 train/test setup using Longformer-Ctx are given in Table 5.

Train \ Test	Ch	E	CS	P	B
Ch	**67.58**	58.41	56.86	62.35	68.23
E	66.62	**60.25**	60.11	64.02	68.07
CS	65.05	59.36	**61.99**	63.85	66.72
P	65.49	58.03	56.69	**65.10**	68.27
B	66.59	58.80	58.22	64.54	**69.12**

σ	0.90	0.78	2.02	0.92	0.77
ρ	0.87	0.86	0.76	0.67	0.79

Table 5: F1 performance on different domain adaptation settings for the fields (Ch)emistry, (E)ngineering, (C)omputer (S)cience, (P)sychology, and (B)iology. Out-of-domain tests use the entire set of data from that field, while in domain tests use 80% of data for training, 10% for validation, and 10% for test. σ is the standard deviation of performance of different train domains on the given test domain, and ρ is Pearson correlation between performance and Euclidean distance from the train domain cluster to the test domain cluster.

Not surprisingly, the best performance for each split occurs when training on data from the same field. We also observe high variance in the maximum performance for each field (σ = 3.32), and between different fields on the same test data, despite large pretrained Transformer models being relatively invariant across domains (Wright and Augenstein, 2020b). This suggests stark differences in how different fields employ citations. Additionally, we observe a strong (inverse) correlation between distance in the embedding space and performance on different domains, showing that using more similar data for training helps on out-of-domain performance (Aharoni and Goldberg, 2020).

6 RQ4: Cite-Worthiness for Transfer Learning

The final question we ask is: to what extent is cite-worthiness detection transferable to downstream tasks in scientific document understanding? To answer this, we fine tune SciBERT on the task of cite-worthiness detection as well as masked language modeling (MLM) on CITEWORTH, followed by fine-tuning on several document understanding tasks. We use SciBERT in order to have a direct comparison with previous work (Beltagy et al., 2019). The tasks we evaluate on come from Beltagy et al. (2019) and are categorized as follows.

Dataset	Reference	Task	Base	LM	Cite	LMCite
BC5CDR	Li et al. (2016)	NER	89.84$_{0.18}$	**90.03$_{0.11}$**	89.73$_{0.25}$	90.02$_{0.79}$
JNLPBA	Kim et al. (2004)	NER	77.02$_{0.36}$	77.13$_{0.53}$	76.97$_{0.44}$	**77.15$_{0.58}$**
NCBI-Disease	Doğan et al. (2014)	NER	**88.79$_{0.35}$**	88.53$_{0.58}$	88.66$_{0.57}$	88.31$_{0.43}$
SciERC	Luan et al. (2018)	NER	67.08$_{0.50}$	66.64$_{0.47}$	67.12$_{0.46}$	**67.48$_{0.45}$**
EBM-NLP	Nye et al. (2018)	PICO	76.61$_{0.21}$	**76.69$_{0.28}$**	76.55$_{0.88}$	76.41$_{0.32}$
ChemProt	Kringelum et al. (2016)	REL	83.17$_{0.43}$	**83.26$_{0.90}$**	82.70$_{1.06}$	83.16$_{0.63}$
SciERC	Luan et al. (2018)	REL	80.21$_{0.81}$	**80.68$_{1.04}$**	80.00$_{1.73}$	80.58$_{0.96}$
ACL-ARC	Jürgens et al. (2018)	CLS	71.82$_{2.93}$	70.95$_{2.25}$	**73.68$_{2.75}$**	72.92$_{3.76}$
SciCite	Cohan et al. (2019)	CLS	84.83$_{0.65}$	85.18$_{0.47}$	85.32$_{0.16}$	**85.35$_{0.29}$**
PaperField	Beltagy et al. (2019)	CLS	65.48$_{0.18}$	**65.57$_{0.27}$**	65.46$_{0.24}$	65.42$_{0.48}$
Average			78.386	78.466	78.619	**78.680**

Table 6: Performance on various downstream scientific document understanding tasks as presented by Beltagy et al. (2019). The metrics used are the same as in their paper: NER is span-level F1, PICO is token level F1, relation extraction is macro-F1, and ChemProt is micro-F1. All runs are averaged across 5 seeds. Subscripts are the standard deviation for 5 runs.

- Named Entity Recognition (NER)/PICO: These tasks involve labelling the spans of different types of entities in a document.
- Relation Extraction (REL): This task involves labelling a sequence for the relationship between two entities.
- Text classification (CLS): Finally, we test on several text classification tasks (citation intent classification and paper field classification), where the goal is to classify a sentence into one or more categories.

We compare five variants of pre-training and fine-tuning, given as follows.

Base The original SciBERT model.

LM SciBERT with MLM fine tuning on CITEWORTH.

Cite SciBERT fine-tuned for the task of cite-worthiness detection. The classifier is a pooling layer on top of the `[CLS]` representation of SciBERT, followed by a classification layer.

LMCite SciBERT with MLM fine tuning and cite-worthiness detection. The two tasks are trained jointly i.e. on each batch of training, the model incurs a loss for both MLM and cite-worthiness detection which are summed together.

The results for all experiments are given in Table 6. Note that the reported results for SciBERT are on re-running the model locally for fair comparison. We first observe that incorporating our dataset into fine-tuning tends to improve model performance across all tasks to varying degrees, with the exception of NER on the NCBI-Disease corpus. The tasks where cite-worthiness as an objective has the most influence are the two citation intent classification tasks (ACL-ARC and SciCite). We see average improvements of 1.8 F1 points for the ACL-ARC dataset (including 2 points F1 improvement over the minumum and maximum model performance of SciBERT) and 0.5 F1 points on SciCite. The best average performance is from the model which incorporates both MLM and cite-worthiness as an objective, which we call CITEBERT.[6]

For other tasks, fine-tuning the language model on CITEWORTH data tends to be sufficient for improving performance, though the margin of improvement tends to be minimal. This is in line with previous work reporting that language model fine-tuning on in-domain data leads to improvements on end-task fine-tuning (Gururangan et al., 2020). CITEWORTH is relatively small compared to the corpus on which SciBERT is originally trained (30.7M tokens for the train and dev splits on which we train versus 3.1B), so one could potentially see further improvements by incorporating more data or including cite-worthiness as an auxiliary task during language model pre-training. However, this is outside the scope of this work.

7 Conclusion

In this work, we present an in-depth study into the problem of cite-worthiness detection in English. We rigorously curate CITEWORTH, a high-quality dataset for cite-worthiness detection; present a

---

[6] We release two CITEBERT models available from the HuggingFace model hub: `copenlu/citebert` and `copenlu/citebert-cite-only`.

paragraph-level contextualized model which improves by 5.31 F1 points on the task of cite-worthiness detection over the existing state-of-the-art; show that CITEWORTH is a good testbed for studying domain adaptation in scientific text; and show that in a transfer-learning setup one can achieve state of the art results on the task of citation intent classification using this data. In addition to studying cite-worthiness and transfer learning, CITEWORTH is suitable for use in downstream natural language understanding tasks. As we retain the S2ORC metadata with the data, one could potentially use the data to study joint cite-worthiness detection and citation recommendation. Additionally, one could explore other useful problems such as modeling different authors' writing styles and incorporating the author network as a signal. We hope that the data and accompanying fine-tuned models will be useful to the research community working on problems in the space of scientific language processing.

Acknowledgements

The research documented in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 801199.

References

Roee Aharoni and Y. Goldberg. 2020. Unsupervised Domain Clusters in Pretrained Language Models. In *ACL*.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. 2018. Construction of the Literature Graph in Semantic Scholar. *NAACL HLT 2018*, pages 84–91.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A Diagnostic Study of Explainability Techniques for Text Classification. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 Task 10: ScienceIE-Extracting Keyphrases and Relations from Scientific Publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346, Vancouver, Canada. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3606–3611.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *CoRR*, abs/2004.05150.

Steven Bird, Robert Dale, Bonnie J Dorr, Bryan Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R Radev, and Yee Fan Tan. 2008. The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. Structural Scaffolds for Citation Intent Classification in Scientific Publications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3586–3596.

Ed Collins, Isabelle Augenstein, and Sebastian Riedel. 2017. A supervised approach to extractive summarisation of scientific papers. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 195–205, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. NCBI Disease Corpus: a Resource For Disease Name Recognition and Concept Normalization. *Journal of biomedical informatics*, 47:1–10.

Charles Elkan and Keith Noto. 2008. Learning Classifiers From Only Positive and Unlabeled Data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–220.

Michael Färber, Alexander Thiemann, and Adam Jatowt. 2018a. A High-Quality Gold Standard for Citation-Based Tasks. In *Proceedings of the*

Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

Michael Färber, Alexander Thiemann, and Adam Jatowt. 2018b. To Cite, or Not to Cite? Detecting Citation Contexts in Text. In *European Conference on Information Retrieval*, pages 598–603. Springer.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360. Association for Computational Linguistics.

Andreas Nugaard Holm, Barbara Plank, Dustin Wright, and Isabelle Augenstein. 2020. Longitudinal Citation Prediction using Temporal Graph Neural Networks. *arXiv preprint arXiv:2012.05742*.

David Jürgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the Evolution of a Scientific Field through Citation Frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA Corpus – a Semantically Annotated Corpus for Bio-Textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the Bio-Entity Recognition Task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75. Citeseer.

Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. 2016. Investigating the Influence of Noise and Distractors on the Interpretation of Neural Networks. *arXiv preprint arXiv:1611.07270*.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. 2016. ChemProt-3.0: a Global Chemical Biology Diseases Mapping. *Database*, 2016.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S Weld. 2020. S2ORC: The Semantic Scholar Open Research Corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.

Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain J Marshall, Ani Nenkova, and Byron C Wallace. 2018. A Corpus With Multi-Level Annotations of Patients, Interventions and Outcomes to Support Language Processing for Medical Literature. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, page 197. NIH Public Access.

Kazunari Sugiyama, Tarun Kumar, Min-Yen Kan, and Ramesh C Tripathi. 2010. Identifying Citing Sentences in Research Papers Using Supervised Learning. In *2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*, pages 67–72. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

David Wadden, Kyle Lo, Lucy Lu Wang, Shanchuan Lin, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *EMNLP*.

Dustin Wright and Isabelle Augenstein. 2020a. Claim Check-Worthiness Detection as Positive Unlabelled Learning. In *Findings of EMNLP*. Association for Computational Linguistics.

Dustin Wright and Isabelle Augenstein. 2020b. Transformer Based Multi-Source Domain Adaptation. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Dustin Wright, Yannis Katsis, Raghav Mehta, and Chun-Nan Hsu. 2019. NormCo: Deep Disease Normalization for Biomedical Knowledge Base Construction. In *AKBC*.

A List of Permissible Section Titles

- introduction
- abstract
- method
- methods
- results
- discussion
- discussions
- conclusion
- conclusions
- results and discussion
- related work
- experimental results
- literature review
- experiments
- background
- methodology
- conclusions and future work
- related works
- limitations
- procedure
- material and methods
- discussion and conclusion
- implementation
- evaluation
- performance evaluation
- experiments and results
- overview
- experimental design
- discussion and conclusions
- results and discussions
- motivation
- proposed method
- analysis
- future work
- results and analysis
- implementation details

B List of Regular Expressions

Citation format regexes:

- `\[([0-9]+\s*[,-;]*\s*)*[0-9]+\s*\]`
- `\(?[12][0-9]3[a-z]?\s*\)`

Hanging citation regex: `\s+\(?(\(\s*\)|like|reference|including|include|with|for instance|for example|see also|at|following|of|from|to|in|by|see|as|e\.?g\.?(,)?|viz(\.)?(,)?)\s*(,)*(-)*[\)\]]?\s*[.?!]\s*$`

Setting	Time
Logistic Regression	00h01m43s
Transformer	02h55m13s
BERT	05h30m30s
SciBERT (no weighting)	09h22m00s
SciBERT	09h32m37s
SciBERT + PU	16h01m27s
Longformer-Solo	75h27m22s
Longformer-Ctx	19h16m07s

Table 7: Average runtimes for each model (runtimes are taken for the entire run of an experiment).

Method	# Parameters
Logistic Regression	198,323
Transformer	9,789,042
BERT	109,484,290
SciBERT	109,920,514
Longformer	149,251,586

Table 8: Number of parameters in each model

C Reproducibility

C.1 Computing Infrastructure

All experiments were run on a shared cluster. Requested jobs consisted of 16GB of RAM and 4 Intel Xeon Silver 4110 CPUs. We used a single NVIDIA Titan X GPU with 12GB of RAM.

C.2 Average Runtimes

The average runtime performance of each model is given in Table 7. Note that different runs may have been placed on different nodes within a shared cluster.

C.3 Number of Parameters per Model

The number of parameters in each model is given in Table 8.

C.4 Validation Performance

The validation performance of each tested model is given in Table 9.

Method	F1
Logistic Regression	-
Transformer	57.02
BERT	60.75
SciBERT (no weighting)	57.52
SciBERT	62.04
SciBERT + PU	61.43
Longformer-Solo	61.67
Longformer-Ctx	67.11

Table 9: Average validation performance for each of the models.

C.5 Evaluation Metrics

The primary evaluation metric used was F1 score. We used the sklearn implementation of `precision_recall_fscore_support` for F1 score, which can be found here: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html. Briefly:

$$p = \frac{tp}{tp + fp}$$

$$r = \frac{tp}{tp + fn}$$

$$F1 = \frac{2 * p * r}{p + r}$$

where tp are true positives, fp are false positives, and fn are false negatives.

C.6 Hyperparameters

Logistic Regression We used a C value of 0.1151 for logistic regression.

Basic Transformer The final hyperparameters for the basic Transformer model are: batch size: 64; number of epochs: 33; feed-forward dimension: 128; learning rate: 0.0001406; number of heads: 3; number of layers: 5; weight decay: 0.1; dropout probability: 0.4. We performed a Bayesian grid search over the following ranges of values, optimizing validation F1 performance: learning rate: $[0.000001, 0.001]$; batch size: $\{4, 8, 16, 32, 64, 128\}$; weight decay: $\{0.0, 0.0001, 0.001, 0.01, 0.1\}$; dropout probability: $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$; number of epochs: $[2, 40]$; feed-forward dimension: $\{128, 256, 512, 1024, 2048\}$; number of heads: $\{1, 2, 3, 4, 5, 6, 10, 12\}$; number of layers: $[1, 12]$.

BERT The final hyperparameters for BERT are: batch size: 8; number of epochs: 3; learning rate: 0.000008075; triangular learning rate warmup steps: 300; weight decay: 0.1; dropout probability: 0.1. We performed a Bayesian grid search over the following ranges of values, optimizer validation F1 performance: learning rate: $[0.0000001, 0.0001]$; triangular learning rate warmup steps: $\{0, 100, 200, 300, 400, 500, 1000, 1500, 2000, 2500, 5000\}$; batch size: $\{4, 8\}$; weight decay: $\{0.0, 0.0001, 0.001, 0.01, 0.1\}$; number of epochs: $[2, 40]$.

SciBERT The final hyperparameters for SciBERT are: batch size: 4; number of epochs: 3; learning rate: 0.000001351; triangular learning rate warmup steps: 300; weight decay: 0.1; dropout probability: 0.1. We performed a Bayesian grid search over the following ranges of values, optimizer validation F1 performance: learning rate: $[0.0000001, 0.0001]$; triangular learning rate warmup steps: $\{0, 100, 200, 300, 400, 500, 1000, 1500, 2000, 2500, 5000\}$; batch size: $\{4, 8\}$; weight decay: $\{0.0, 0.0001, 0.001, 0.01, 0.1\}$; number of epochs: $[2, 40]$.

Longformer-Ctx The final hyperparameters for Longformer-Ctx are: batch size: 4; number of epochs: 3; learning rate: 0.00001112; triangular learning rate warmup steps: 300; weight decay: 0.0; dropout probability: 0.1. We performed a Bayesian grid search over the following ranges of values, optimizer validation F1 performance: learning rate: $[0.0000001, 0.0001]$; triangular learning rate warmup steps: $\{0, 100, 200, 300, 400, 500, 1000, 1500, 2000, 2500, 5000\}$; batch size: $\{4, 8\}$; weight decay: $\{0.0, 0.0001, 0.001, 0.01, 0.1\}$; number of epochs: $[2, 6]$.

C.7 Data

CITEWORTH is constructed from the S2ORC dataset, which can be found here: https://github.com/allenai/s2orc. In particular, CITEWORTH is built using the `20200705v1` release of the data. A link to the CITEWORTH data can be found here: https://github.com/copenlu/cite-worth.

Cross-Lingual Cross-Domain Nested Named Entity Evaluation on English Web Texts

Barbara Plank
IT University of Copenhagen
Department of Computer Science
Rued Langgaards Vej 7, 2300 København S
bplank@gmail.com

Abstract

Named Entity Recognition (NER) is a key Natural Language Processing task. However, most existing work on NER targets flat named entities (NEs) and ignores the recognition of nested structures, where entities can be enclosed within other NEs. Moreover, evaluation of Nested Named Entity Recognition (NNER) across domains remains challenging, mainly due to the limited availability of datasets. To address these gaps, we present EWT-NNER, a dataset covering five web domains annotated for nested named entities on top of the English Web Treebank (EWT). We present the corpus and an empirical evaluation, including transfer results from German and Danish. EWT-NNER is annotated for four major entity types, including suffixes for derivational entity markers and partial named entities, spanning a total of 12 classes. We envision the public release of EWT-NNER to encourage further research on nested NER, particularly on cross-lingual cross-domain evaluation.

1 Introduction

Named Entity Recognition (NER) is the task of finding and classifying named entities in text, such as locations, organizations, and person names. It is a key task in Natural Language Processing (NLP), and an important step for downstream applications like relation extraction, co-reference resolution and question answering. The task has received a substantial amount of attention. However, tools and existing benchmarks largely focus on flat, coarse-grained entities and single-domain evaluation.

Flat, coarse-grained entities however eschew semantic distinctions which can be important in downstream applications (Ringland et al., 2019). Examples include embedded locations ('New York Times'), entities formed via derivation ('Italian cuisine') and tokens which are in part named entities ('the Chicago-based company').

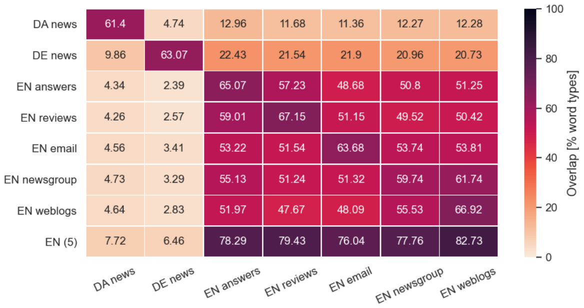

Figure 1: Domain overlap between target (x-axis) and source training (y-axis) domains (DE: German, DA: Danish, EN: proposed dataset EWT-NNER).

Research interest on methods to handle nested entities is increasing (Katiyar and Cardie, 2018). However, there is a lack of datasets, particularly resources which cover multiple target domains.

To facilitate research on cross-domain nested NER, we introduce a new layer on top of the English Web Treebank (EWT), manually annotated for NNER. The corpus spans five web domains, four major named entity types, enriched with suffixes marking derivations and partial NEs. Figure 1 shows the domain overlap in terms of word types. Besides providing in-language benchmark results, EWT-NNER enables research on cross-lingual transfer from German and Danish.

Contributions The main contributions are: i) We introduce EWT-NNER, a corpus for nested NER over five web domains. ii) A report on cross-lingual and in-language baselines. Our results highlight the challenges of processing web texts, and the need for research on cross-lingual cross-domain NNER.

2 Related Work

Nested NER Much research has been devoted to flat Named Entity Recognition, with a long tradition of shared tasks (Grishman and Sund-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1808–1815
August 1–6, 2021.

heim, 1996; Grishman, 1998; Tjong Kim Sang and De Meulder, 2003; Baldwin et al., 2015). The problem of nested named entity recognition (NNER) has instead received less attention. This lack of breadth of research has been attributed to practical reasons (Finkel and Manning, 2009), including a lack of annotated corpora (Ringland et al., 2019).

Existing nested NE corpora span only a handful of languages and text domains. This is in stark contrast to resources for flat NER, which are available for at least up to 282 languages (Pan et al., 2017) and multiple domains, including a very recent effort (Liu et al., 2021). Existing NNER resources for English cover newswire (e.g., ACE, WSJ) (Mitchell et al., 2005; Ringland et al., 2019) and biomedical data (e.g., GENIA) (Kim et al., 2003; Alex et al., 2007; Pyysalo et al., 2007). Beyond English, there exist free and publicly available nested NER datasets. These include the GermEval 2014 dataset (Benikova et al., 2014a), which is one of the largest existing German NER resources covering largely news articles (Benikova et al., 2014b). Recently, the GermEval annotation guidelines inspired the creation of a Danish corpus (Plank et al., 2020). They added a layer of nested NER on top of the existing Danish Universal Dependency treebank (Johannsen et al., 2015). Both German and Danish corpora derive their annotation guidelines from the NoStA-D annotation scheme (Benikova et al., 2014b), which we adopt for EWT-NNER (Section 3.1). To facilitate research, a fine-grained nested NER annotation on top of the Penn Treebank WSJ has been released recently (Ringland et al., 2019). In contrast to ours, the WSJ NNER corpus spans 114 entity types and 6 layers, and includes numericals and time expressions beyond named entities. We instead focus on NEs with a total of 12 classes and 2 layers.

As outlined by Katiyar and Cardie (2018), nested named entities are attracting more research attention. Modeling solutions opt for diverse strategies, from hierarchical systems to graph-based methods and models based on linearization (Alex et al., 2007; Finkel and Manning, 2009; Sohrab and Miwa, 2018; Luan et al., 2019; Lin et al., 2019; Zheng et al., 2019; Straková et al., 2019; Shibuya and Hovy, 2020). The current top-performing neural systems use typically either a linearization, a multi-task learning or a graph-based approach (Straková et al., 2019; Plank et al., 2020; Yu et al., 2020). We evaluate two such methods.

English Web Treebank The English Web Treebank (EN-EWT) (Bies et al., 2012; Petrov and McDonald, 2012; Silveira et al., 2014) is a dataset introduced as part of the first workshop on Syntactic Analysis of Non-Canonical Language (SANCL). The advantage of EWT is that it spans over 200k tokens of texts from five web domains: Yahoo! answers, newsgroups, weblogs, local business reviews from Google and Enron emails. Gold annotations are available for several NLP tasks. The corpus was originally annotated for part-of-speech tags and constituency structure in Penn Treebank style (Bies et al., 2012). Gold standard dependency structures were annotated on EWT via the Universal Dependencies project (Silveira et al., 2014). Recently, efforts extend EWT (or parts thereof) to further semantic (Abend et al., 2020) and temporal layers (Vashishtha et al., 2019). We contribute a novel nested NER layer on top of the freely available UD EN-EWT corpus split (Silveira et al., 2014).

3 The EWT-NNER corpus

This section describes the corpus and annotation.

3.1 Annotation Scheme and Process

We depart from the NoSTA-D named entity annotation scheme (Benikova et al., 2014b), introduced in the GermEval 2014 shared task and adopted for Danish (Plank et al., 2020). The entity labels span a total of 12 classes, distributed over four major entities (Tjong Kim Sang and De Meulder, 2003): location (LOC), organization (ORG), person (PER) and miscellaneous (MISC). There are two further sub-types: '-part' and '-deriv'. Entities are annotated using a two-level scheme. First-level annotations contain largest entity spans (e.g., the 'Alaskan Knight'). Second-level annotations are nested entities. In particular:

- We annotate named entities with two layers. The outermost layer embraces the longer span and is the most prominent entity reading, and the inner span contains secondary or sub-entity readings. If there would be more than 2 layers, we drop further potential readings in favor of keeping two layers. Example: '[[UNSC]*ORG* Resolution 1559]*MISC*'[1]

---

[1] Benikova et al. (2014b) report a few cases (around 1 in 1,000 sentences) where the 2 levels do not suffice, but opted for the 2-layer scheme for simplicity. We follow this, observing a similar pattern (2 cases every 1,000 sentences). We kept notes of these cases, yet leave an investigation to future work.

- Full NEs are annotated as LOC (location), ORG (organization), PER (person) or MISC (miscellaneous other).

- Only full nominal phrases are potential full NEs. Pronouns and all other phrases are ignored. National holidays or religious events (*Christmas, Ramadan*) are also not annotated. Determiners and titles are not part of NEs.

- Named entities can also be part of tokens and are annotated as such with the suffix "-part". Example: '[Thailand-based]*LOCpart*', '[Nintendo-inspired]*ORGpart* costume'

- Derivations of NEs are marked via the suffix *deriv*, e.g., 'the [Alaskan]*LOCderiv* movie'.

- Geopolitical entities deserves special attention. We opted for annotating its first reading as ORG, with a secondary LOC reading, to reduce ambiguity. This is the same as in the Danish guidelines. The original German NoStA-D guidelines did not provide detailed guidelines for this case, yet mentions some categories were conflated (LOC and geopolicitcal entities). They seem most frequently annotated as LOC, yet we find also similar annotations in the German data (especially for multi-token NEs like *Borussia Dortmund*).

The full annotation guidelines for EWT-NNER with examples, annotation decisions and difficult cases can be found in the accompanying repository.[2] Two annotators were involved in the process, both contributed to the earlier Danish corpus. One annotator is an expert annotator with a degree in linguistics; the second annotator is a computer scientist. A data statement is provided in the appendix. Inter-annotator agreement (IAA) is measured on a random sample of 100 sentences drawn from the development data. This resulted in the following agreement statistics: raw token-level agreement of 98%, Cohen's kappa over all tokens 88.20% and Cohen's kappa of 81.73% for tokens taking part in an entity as marked by at least one annotator. The final dataset was annotated by the professional linguist annotator. The annotation took around 3 working days per 25,000 tokens.

	Train	Dev	Test
answers 💬	2,631	419	438
reviews 📝	2,724	554	535
email ✉	3,770	524	606
newsgroup 👥	1,833	274	284
weblogs 📰	1,585	231	214
total	12,543	2,002	2,077

Table 1: Number of sentences in the UD EWT split.

	All	Nest.	💬	📝	✉	👥	📰
Location	3,553	901	20.6%	9.7%	15.2%	22.2%	32.3%
LOC deriv	1,094	161	18.1%	7.7%	3.4%	21.7%	49.1%
LOC part	76	29	7.9%	5.3%	14.5%	18.4%	53.9%
Person	4,202	353	4.5%	9.0%	45.0%	18.2%	23.3%
PER deriv	22	2	0%	4.8%	38.1%	19%	38.1%
PER part	24	2	0%	4.2%	16.7%	58.3%	20.8%
Organization	3,309	133	9.8%	15.2%	25.1%	21.0%	29.0%
ORG deriv	32	0	15.2%	9.1%	27.3%	12.1%	36.4%
ORG part	45	45	10.3%	0%	62.1%	0%	27.6%
Miscellaneous	1,576	11	21.2%	8.4%	29.7%	29.0%	11.8%
MISC deriv	3	0	0%	33.3%	33.3%	33.3%	0%
MISC part	9	2	0%	0%	57.1%	28.6%	14.3%
total	13,945	1,622	% of all entities per domain				

Table 2: Distribution of entities in all of EWT-NNER (16k sentences): All, Nested, and % of All.

3.2 Data statistics

Statistics over the data split and distribution of the web texts are provided in Table 1. A comparison to German and Danish on coarse-level statistics are provided in Table 3. Details of the entity distibution in EWT-NNER are given in Table 2. The entire EWT-NNER contains a total of over 16,000 sentences and over 13,000 entities. Around 42% of the sentences contain NEs. Over 11.6% are nested NEs, 8.3% are derivations and 1.1% are parts of names. Compared to GermEval 2014, this is a higher density of nested entities (11.6% vs 7.7% in the German data), yet a lower percentage of derivations and partial NEs. The data is provided in CoNLL tabular format with BIO entity encoding.

4 Experimental Setup

We are interested in a set of benchmark results to provide: a) zero-shot transfer results from Danish and German; b) in-language results (training on all 5 EN domains vs per-domain models); and c) results on cross-lingual cross-domain evaluation when training on multiple languages jointly.

For the experiments, we use fine-tuning of con-

---

[2] Data and tech report (Plank and Dam Sonniks, 2021) are available at: `http://github.com/bplank/nested-ner`

Language Domain	Training German news	 Danish news	 English web (5)	Development German news	 Danish news	 English web (5)	Test English web (5)
Sentences	24,002	4,383	12,538	2,200	564	2,002	2,077
Tokens	452,853	80,378	204,609	41,653	10,332	25,150	25,097
# Entities	31,545	3,035	10,673	2,886	504	1,549	1,711

Table 3: Overview of EWT-NNER (this paper) and comparison to existing nested NER datasets adopting the NoSTA-D 2-level NER annotation scheme: German (Benikova et al., 2014b) and Danish (Plank et al., 2020).

textualized embeddings with AllenNLP (Gardner et al., 2018) using the MaChAmp toolkit (van der Goot et al., 2021). We use the proposed default parameters. Whenever we train on English data, we took the smallest weblogs domain (231 sentences) as model selection set and assume no further in-domain dev set. For German and Danish, we use the provided news dev sets. For all experiments we report the average performance over 3 runs.

As contextualized embeddings, we investigate BERT (Devlin et al., 2019), multilingual BERT and XLM-R (Conneau et al., 2020). We evaluated two decoding strategies: the first takes the Cartesian product of inner and outer NER layer and treats it as a standard single-label decoding strategy. An advantage of this strategy is that any sequence tagging framework can be used; a disadvantage is the increased label space. To tackle this we use a two-headed multi-task decoder, one for each entity layer, as found effective (Plank et al., 2020). Initial experiments confirmed that the single-label decoding is less accurate, confirming earlier findings (Straková et al., 2019; Plank et al., 2020). We report results with the two-headed decoder only, and further results in the appendix.

Evaluation is based on the official GermEval 2014 (Benikova et al., 2014b) metric and script, i.e., strict span-based F1 over both entity levels.

5 Results

Table 4 shows the results of training models on German (DE: GermEval 2014), Danish (DA: DaN+), and their union (+), for zero-shot transfer (top rows). It provides further results of training on all English EWT-NNER training data (from all five web domains) both for multilingual models (using multilingual BERT or XLM-R) and monolingual models (English BERT and Roberta). Figure 2 provides cross-domain results of training only on English subdomains.

Language Domain # Entities	DE news 2,886	DA news 504	EN (EWT-NNER) 💬 285	 📝 174	 ✉ 340	 👥 354	 📰 396
			zero-shot:				
DA	68.47	80.95	52.43	53.09	54.99	48.92	62.66
DE	84.41	76.67	59.04	55.93	60.53	59.11	63.23
DA+DE	**85.32**	82.39	61.12	56.86	61.98	60.48	64.43
			full in-language data:				
EN (all 5)	69.98	75.22	71.00	73.35	79.33	81.20	86.80
EN+DA	71.53	82.77	73.04	74.05	80.48	80.95	86.69
EN+DE	84.91	77.71	71.79	73.70	80.53	79.64	85.48
EN+DA+DE	84.80	82.43	72.61	74.45	79.39	**81.92**	86.59
EN+DA+DE$_X$	83.47	**83.07**	**73.86**	74.35	79.60	79.03	86.56
			full in-language, monolingual embeds:				
EN BERT	26.59	27.54	70.53	73.25	**81.09**	81.31	**87.04**
EN Roberta	34.35	33.37	70.01	**78.77**	81.02	77.44	84.58

Table 4: F1 scores on dev sets with mBERT/X[LM-R] (upper columns) and monolingual models for English.

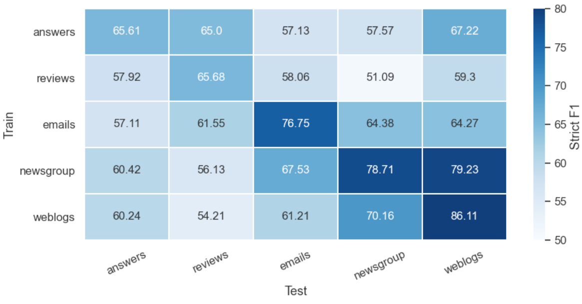

Train \ Test	answers	reviews	emails	newsgroup	weblogs
answers	65.61	65.0	57.13	57.57	67.22
reviews	57.92	65.68	58.06	51.09	59.3
emails	57.11	61.55	76.75	64.38	64.27
newsgroup	60.42	56.13	67.53	78.71	79.23
weblogs	60.24	54.21	61.21	70.16	86.11

Figure 2: In-language cross-domain evaluation.

Take-aways While zero-shot transfer between news (on German and Danish) is around 70 F1 (68.5 and 76.7), zero-shot transfer to the EWT-NNER web domains is low, particularly for answers (💬), reviews (📝), emails (✉) and newsgroups (👥). Training on both Danish and German improves zero-shot performance over all domains.

For English cross-domain evaluation, we observe a large variation across domains in Figure 2. Here, we train models on the EWT-NNER training portion of a single web domain, and evaluate the resulting model across all five web domains (in-domain and out-domain). The heatmap confirms that training within domain is the most beneficial (results on the diagonal), but large drops can be observed across domains. Reviews (📝) and Yahoo! answers (💬) remain the most challenging with the lowest F1 scores. Weblogs (📰) shows the highest results. We tentatively attribute this to the good coverage of weblogs over all entity classes (see Table 2) and the well-edited style of the text (by inspection many posts are about politics and military events). If we compare the results to the model trained on all English data in Table 4 (EN all 5), we observe that training on all web training data improves over the single web texts.

We investigate cross-lingual cross-domain results, to evaluate whether a model trained on English data alone can be improved by further cross-lingual transfer. Table 4 shows that this is the case. There is positive transfer from German and Danish data, with the mBERT model (EN+DA+DE) boosting performance (on most domains). The larger XLM-R model helps on specific domains, but it is not consistently better than mBERT.

So far we focused on multilingual contextualized embeddings. The last rows in Table 4 compares the multilingual models to monolingual ones. Interestingly, in this domain a monolingual model does not consistently outperform the multlingual model. While for some domains the EN model is substantially better, this is not the case overall. On average over the 5 web domains, the tri-lingual model with mBERT reaches a slightly overall F1 (average of 78.99), followed by both the monolingual BERT model (78.64) and XLM-R (78.63).

Language	DE	DA	EN EWT-NNER					
Domain	news	news	💬	📝	✉	👥	📰	Avg
Entities	6,693	566	316	167	465	314	449	
EN	70.10	72.68	73.47	68.47	75.93	82.66	88.87	77.88
EN+DA+DE	84.36	79.94	74.28	66.54	77.44	84.10	87.71	**78.01**

Table 5: F1 scores on the test sets with mBERT.

Test sets Finally, we run the best model on the test sets and compare to training on English alone. Table 5 confirm the overall trends. There is a positive transfer across languages for cross-domain evaluation, with improvements on the majority of domains. The best model reaches an average F1 score of 78.01 on the five web domains. Compared to results within newswire, there is room to improve NNER over domains.

Analysis We perform a qualitative analysis of the best model (EN+DA+DE) on the dev sets. Detailed scores are in Table 7 in the appendix. The overall F1 of 72% on 💬 is largely due to low recall on person names (recall 63%) (e.g., peculiar names such as 'Crazy Horse', a Dakota leader) and missed lower-cased product names ('ipod'). On ✉, recall on ORG and LOC is low (55% and 65%), as organizations and locations are missed also due to unconventional spelling in emails. In reviews (📝), the model reaches its lowest F1 on ORG (67%) as it mixes up people names with organizations and lacks recall. Newsgroup (👥) is broad (e.g., discussions from astronomy to cat albums) with the lowest per-entity F1 of 75% for MISC. Newsgroup and weblogs are the domains with the most LOCderiv entities, which the model easily identifies (F1 of 93% and 99% in 👥 and 📰, respectively). Overall, weblogs (📰) has the highest per-entity F1 scores, all above 75%, with the highest overall F1 on LOC (92 F1; in comparison to 57% on ✉ and 79% on 💬). This high result on weblogs can be further attributed to smaller distance to the training sources (as indicated in the overlap plot in Figure 1) and to some degree of using this domain for tuning. From a qualitative look, we note that the weblogs sample is rather clean text, often in reporting style about political events similar to edited news texts, which we believe is part of the reason for the high performance compared to the other domains in EWT-NNER.

6 Conclusions

We present EWT-NNER, a nested NER dataset for English web texts, to contribute to a limiting nested NER resource landscape. We outline the dataset, annotation guidelines and benchmark results. The results show that NNER remains challenging on web texts, and cross-lingual transfer helps. We hope this dataset encourages research on cross-lingual cross-domain NNER. There are many avenues for future research, which include e.g., alternative decoding (Yu et al., 2020), pre-training models and adaptation (Gururangan et al., 2020).

Acknowledgments

The author would like to thank the three anonymous reviewers for their valuable comments to improve this paper. This research is supported by the Independent Research Fund Denmark (DFF) grant 9131-00019B and 9063-00077B.

References

Omri Abend, Dotan Dvir, Daniel Hershcovich, Jakob Prange, and Nathan Schneider. 2020. Cross-lingual semantic representation for NLP with UCCA. In *Proceedings of the 28th International Conference on Computational Linguistics: Tutorial Abstracts*, pages 1–9, Barcelona, Spain (Online). International Committee for Computational Linguistics.

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Biological, translational, and clinical language processing*, pages 65–72, Prague, Czech Republic. Association for Computational Linguistics.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Pado. 2014a. Germeval 2014 named entity recognition shared task: Companion paper.

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014b. NoSta-D named entity annotation for German: Guidelines and dataset. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2524–2531, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ann Bies, Justing Mott, Colin Warner, and Seth Kulick. 2012. English web treebank ldc2012t13. In *Web Download. Philadelphia: Linguistic Data Consortium, 2012*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.

Ralph Grishman. 1998. Research in information extraction: 1996-98. In *TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998*, pages 57–60, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Anders Johannsen, Héctor Martínez Alonso, and Barbara Plank. 2015. Universal dependencies for danish. In *International Workshop on Treebanks and Linguistic Theories (TLT14)*, page 157.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2021. CrossNER: Evaluating cross-domain named entity recognition. In *To appear at AAAI*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Mitchell, Stephanie Strassel, Shudong Huang, and Ramez Zakhary. 2005. Ace 2004 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 1:1–1.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Proceedings of the SANCL shared task workshop*.

Barbara Plank and Sif Dam Sonniks. 2021. Annotation guidelines for Nested Named Entities for English. Technical report, IT University of Copenhagen.

Barbara Plank, Kristian Nørgaard Jensen, and Rob van der Goot. 2020. DaN+: Danish nested named entities and lexical normalization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6649–6662, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. 2007. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Biological, translational, and clinical language processing*, pages 25–32, Prague, Czech Republic. Association for Computational Linguistics.

Nicky Ringland, Xiang Dai, Ben Hachey, Sarvnaz Karimi, Cecile Paris, and James R. Curran. 2019. NNE: A dataset for nested named entity recognition in English newswire. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5176–5181, Florence, Italy. Association for Computational Linguistics.

Takashi Shibuya and Eduard Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2906–2919, Florence, Italy. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. A boundary-aware neural model for nested named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 357–366, Hong Kong, China. Association for Computational Linguistics.

A Data Statement

This following data statement (Bender and Friedman, 2018) documents the origin of the data annotations and provenance of the original English Web Treebank (EWT) data.

CURATION RATIONALE Annotation of nested named entities (NNE) in web text domains to study the impact of domain gap on cross-lingual transfer.

LANGUAGE VARIETY Mostly US (en-US) mainstream English as target. Transfer from Danish (da-DK) and German (de-DE).

SPEAKER DEMOGRAPHIC Unknown.

ANNOTATOR DEMOGRAPHIC Native languages: Danish, German. Socioeconomic status: higher-education student and university faculty.

SPEECH SITUATION Scripted, spontaneous.

TEXT CHARACTERISTICS Sentences from journalistic edited articles and from social media discussions and postings.

PROVENANCE APPENDIX The data originates from the English Web Treebank (EN-EWT) (Bies et al., 2012; Petrov and McDonald, 2012; Silveira et al., 2014) and data split available at: `https://github.com/UniversalDependencies/UD_English-EWT/`

B Additional results

Table 6 provides additional results for both decoding strategies. It shows that single-label decoding is outperformed by the two-head decoder, confirming similar results on Danish (Plank et al., 2020).

Language	DE	DA	EN EWT-NNER				
Domain	news	news	answ	revs	email	nwsgrp	weblg
# Entities	6,693	566	321	168	468	319	447
	german	news	answers	reviews	email	newsgroup	weblogs
en.bert.single-merged	**29.33**	24.68	67.16	71.57	79.23	75.12	79.81
en.bert.multitask	26.59	**27.54**	**70.53**	**73.25**	**81.09**	**81.31**	**87.04**
en.roberta.single-merged	31.41	24.86	67.09	72.07	81.01	72.35	79.34
en.roberta.multitask	**34.35**	**33.37**	**70.01**	**78.77**	**81.02**	**77.44**	**84.58**

Table 6: F1 scores on the dev set with monolingual models both decoding strategies.

	answers 💬	reviews ✎	email ✉	newsgroups 👥	weblogs 📰
Location	79.01 (72.73)	85.37 (89.74)	57.14 (55.17)	83.76 (92.45)	92.31 (93.10)
LOC deriv	74.42 (84.21)	100.00 (100.00)	0.00 (0.00)	93.33 (100.00)	99.07 (98.15)
LOC part	–	100.00 (100.00)	–	–	100.00 (100.00)
Person	72.73 (63.16)	83.33 (87.50)	90.25 (86.63)	92.09 (98.46)	91.46 (88.24)
PER deriv	–	–	100.00 (100.00)	0.00 (0.00)	0.00 (0.00)
PER part	–	–	–	0.00 (0.00)	–
Organization	64.58 (70.45)	67.42 (60.00)	64.86 (65.45)	83.64 (84.15)	85.83 (83.06)
ORG deriv	0.00 (0.00)	–	–	–	0.00 (0.00)
ORG part	–	–	–	0.00 (0.00)	–
Miscellaneous	75.76 (67.57)	85.71 (81.82)	80.00 (83.64)	75.82 (65.91)	75.00 (84.00)
MISC deriv	–	–	–	–	–
MISC part	–	–	–	–	0.00 (0.00)

Table 7: Per-entity evaluation of outer level strict FB1 score (and recall) of the best model EN+DE+DA with mBERT on the dev sets.

Argument Undermining: Counter-Argument Generation by Attacking Weak Premises

Milad Alshomary † **Shahbaz Syed** ‡ **Arkajit Dhar** †
Martin Potthast ‡ **Henning Wachsmuth** †
†Paderborn University, Paderborn, Germany, `milad.alshomary@upb.de`
‡Leipzig University, Leipzig, Germany, `<first>.<last>@uni-leipzig.de`

Abstract

Text generation has received a lot of attention in computational argumentation research as of recently. A particularly challenging task is the generation of *counter*-arguments. So far, approaches primarily focus on rebutting a given conclusion, yet other ways to counter an argument exist. In this work, we go beyond previous research by exploring argument *undermining*, that is, countering an argument by attacking one of its premises. We hypothesize that identifying the argument's weak premises is key to effective countering. Accordingly, we propose a pipeline approach that first assesses the premises' strength and then generates a counter-argument undermining the weakest among them. On one hand, both manual and automatic evaluation underline the importance of identifying weak premises in counter-argument generation. On the other hand, when considering correctness and content richness, human annotators favored our approach over state-of-the-art counter-argument baselines.

Claim (title): Feminism is in the third wave. In countries, such as America, it has caused nothing but trouble.
Premises (sentences): I'm going to be bringing up several feminist arguments I have heard myself. First off, the all-dreaded wage gap. It has in fact been illegal to pay women less than men since the early 1960s. [...] Secondly the pink tax. Women's products are of course going to cost more than men's. They use entirely different chemicals specifically made to cater to softer skin [...] and for *the fourth point, although I could go on for much longer, the feminst movement is needed elsewhere. In countries, such as Iraq, India, and Saudia Arabia. The feminst movement being in a country where women aren't being forced to cover their entire bodies, aren't being sold off with doweries, and aren't being oppressed, is downright absurd.*
Counter-argument: The fact that other women have it worse doesn't mean that women don't have it bad elsewhere. For example, I can be fired for being gay in 29 out of 50 states in the US. The fact that people are stoned for being gay in Brunei doesn't mean that isn't an example of homophobia...

Table 1: An example argument (claim + premises) and a counter-argument in response to it, taken from Reddit changemyview. The italicized premise part was quoted by the user who stated the counter-argument.

1 Introduction

Following Walton (2009), a counter-argument can be defined as an attack on a specific argument by arguing against either its claim (called *rebuttal*), the validity of reasoning of its premises toward its claim (*undercut*), or the validity of one of its premises (*undermining*). Not only the mining and retrieval of counter-arguments have been studied (Peldszus and Stede, 2015; Wachsmuth et al., 2018b), recent works also tackled the generation of counter-arguments. Among these, Bilu et al. (2015) and Hidey and McKeown (2019) studied the task of contrastive claim generation, the former in a partly rule-based manner, the latter data-driven. Moreover, Hua and Wang (2019) proposed a neural counter-argument generation approach. So far, however, research focused only on rebutting a given argument, ignoring the other aforementioned types. We expand this research by studying to what extent argument *undermining* can be utilized in counter-argument generation.

In argument undermining, the validity of some premises is questioned. Such a phenomenon can be observed often in online discussions on social media. For example, in the discussion excerpt in Table 1, taken from the Reddit forum *changemyview*,[1] a user contests the whole stated argument (claim and premises) by referring to the specific premise shown italicized (on Reddit, it is the quoted part of

---

[1] https://en.wikipedia.org/wiki/R/changemyview

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1816–1827
August 1–6, 2021. ©2021 Association for Computational Linguistics

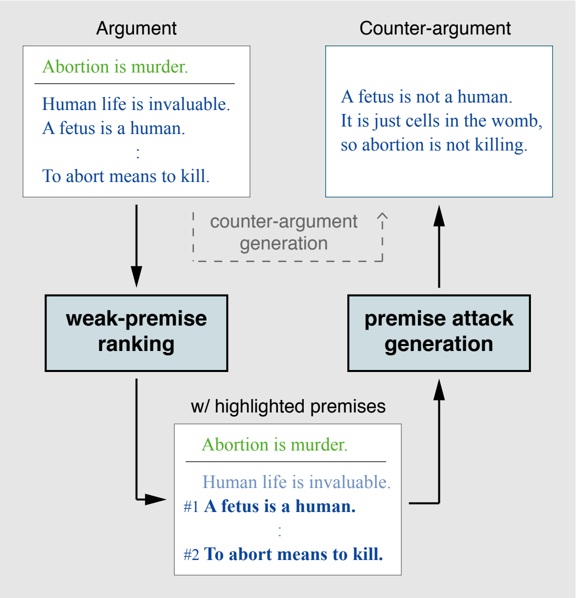

Figure 1: Argument undermining: Instead of countering a given argument directly, our approach first ranks the argument's premises by predicted weakness. Then, an attack focused on the weakest premises is generated.

the text). This implies two steps: first, to identify a potentially weak and thus attackable premise in the argument, and second, to counter it.

In this work, we propose to tackle the task of counter-argument generation by attacking one of the weak premises of an argument. We hypothesize that identifying a weak premise is key to effective counter-argument generation—especially when the argument is of high complexity, comprising multiple interlinked claims and premises, which makes it hard to comprehend the argument as a single unit. Figure 1 illustrates our two-step pipeline approach: it first detects premises that may be attackable and then generates a counter-argument targeting one or more of these premises. To identify weak premises, we build on the work of Jo et al. (2020), who classify attackable sentences using BERT. Unlike the authors, we rank premises based on their attackability concerning the argument's main claim, utilizing the learning-to-rank approach of Han et al. (2020). For the second step, similar to Wolf et al. (2019), we fine-tune a pre-trained transformer-based language model (Radford et al., 2018), in a multitask learning setting: next-token classification and counter-argument classification.

In our experiments, we make use of the changemyview (CMV) dataset of Jo et al. (2020), where each instance is a post consisting of a title (say, an argument's claim) and a text (the argument's premises). Some of the sentences in the text are quoted by comments to the post. These sentences are considered to be weak/attackable premises.[2] We further extend the dataset by collecting texts from comments defining counter-arguments.

To analyze our approach, we evaluate both of its steps individually as well as in combination. In particular, we first compare our ranking model for detecting attackable premises to Jo et al. (2020), observing significant improvements in the effectiveness. Second, given the ground-truth attackable premise (the quoted sentences), we evaluate our counter-argument generation model against several baselines. Our automatic evaluation provides evidence that training the model with the weak premise annotated significantly boosts the scores across all metrics. We additionally confirm these results by a manual evaluation, indicating that our approach is better than the baseline in 56% of the cases. Finally, we apply our generation model based on the automatically detected weak premises and compare it to the approach of Hua and Wang (2019), which generates counter-arguments with opposing stance to the argument (i.e., rebuttals). While the automatic evaluation here is not in favor of our approach, the manual evaluation gives evidence of the favorability of our approach on all three tested quality dimensions.

To summarize, our contributions are:[3]

- A model for detecting premise attackability, achieving state-of-the-art effectiveness.
- A new approach to counter-argument generation that identifies and attacks weak premises.
- Empirical evidence of the impact of considering specific attackable premises in the argument when generating a counter-argument.

2 Related Work

Recently, text generation has gained much interest in computational argumentation, both for single claims and complete arguments. Bilu et al. (2015) composed opposing claims combining rules with classifiers, whereas Hidey and McKeown (2019) tackled an analog task with neural methods. Alshomary et al. (2020) reconstructed implicit claims from argument premises using triplet neural networks, and Gretz et al. (2020) explored ability of

---

[2] Our assumption is that each sentence represents a premise supporting the main claim mentioned in the title of the post

[3] Code and resources can be found under `https://github.com/webis-de/ACL-21`

GPT-2 to generate claims on topics. Recently, Alshomary et al. (2021) studied how to encode specific beliefs into generated claims, whereas Chen et al. (2018) flipped the bias of claim-like news headlines using style transfer. Sato et al. (2015) generated full arguments in a largely rule-based way. Building on the model of rhetorical argumentation strategies by Wachsmuth et al. (2018a), El Baff et al. (2019) modeled argument synthesis as a language modeling task, and Schiller et al. (2020) studied the neural generation of arguments on a topic with controlled aspects and stance. Unlike all these, we deal with *counter*-arguments.

Research exists for mining attack relations (Cocarascu and Toni, 2017; Chakrabarty et al., 2019; Orbach et al., 2020), mining counter-considerations from text (Peldszus and Stede, 2015), and retrieving counter-arguments (Wachsmuth et al., 2018b; Orbach et al., 2019). However, only the two works of Hua and Wang (2018, 2019) realy generated such arguments. Their latest neural approach takes an argument or claim as input and generates a counter-argument rebutting it. Differently, we consider countering an argument by attacking one of its premises, known as *undermining* (Walton, 2009).

Part of our approach is to identify attackable premises, which can be studied from an argument quality perspective. That is, a premise is attackable when it lacks specific quality criteria. A significant body of research has studied argument quality assessment, with a comprehensive survey of quality criteria presented in Wachsmuth et al. (2017). Implicitly, we target criteria such as a premise's acceptability or relevance. Still, we follow Jo et al. (2020) in deriving attackability from the sentences of posts that users attack in the Reddit forum CMV. These sentences represent premises supporting the claim encoded in a post's title. The authors experimented with different features that potentially reflect weaknesses in the premises. Their best model for identifying attackable premises is a BERT-based classifier. We use their data to learn weak premise identification, but we address it as a learning-to-rank task.

As for text generation, significant advances have been made through fine-tuning large pre-trained language models (Solaiman et al., 2019) on target tasks. We also benefit from this by utilizing a pre-trained transformer-based language model (Devlin et al., 2018), and we fine-tune it in a multi-task fashion similar to Wolf et al. (2019).

3 Approach

As sketched in Figure 1, the pipeline approach we propose counters an argument by attacking the validity of one of its potentially weak premises. This section presents the two main steps of our approach: first, the ranking of weak and thus attackable premises, and second, the generation of an attack on the weak premises.

3.1 Weak-Premise Ranking

Given an argument in the form of a claim and a set of premises, the task is to identify the argument's attackable premises. Unlike previous work (Jo et al., 2020), we model the task as a ranking task instead of a classification task, in which, for each argument, we learn to rank its premises by their weakness relevant to the claim. Our hypothesis here is that the attackability of a premise can be better learned when considering both the claim and other premises of the argument.

We operationalize the weak-premise ranking similar to the ranking approach of Han et al. (2020). In particular, given a set of premises and the claim, we first represent each premise by concatenating its tokens with the claim's tokens, separated by special tokens *[cls]* and *[sep]*:

[cls] claim_tokens [sep] premise_tokens [sep]

Next, the resulting sequences are passed through a BERT model to obtain a vector representation for every premise. Each vector is then projected through a dense layer to get a score $\hat{y}$ that reflects the weakness of the premise. Finally, a list-wise objective function (we use a Softmax loss) is optimized jointly on all premises of an argument as follows:

$$l(y, \hat{y}) = -\sum_{i=1}^{n} y_i \cdot \log\left(\frac{exp(\hat{y}_i)}{\sum_{j=1}^{n} exp(\hat{y_j})}\right),$$

where y is a binary ground-truth label reflecting whether the given premise is attackable ($y = 1$) or not ($y = 0$). Given training data, we can thus learn to rank premises by weakness.

3.2 Premise Attack Generation

Given the output of the ranking step, we identify the k highest-ranked premises in an argument to be attackable (in our experiments, we test $k = 1$ and $k = 3$). Then, we generate a counter-argument putting the identified attackable premises into the focus. To this end, we follow Wolf et al. (2019) in

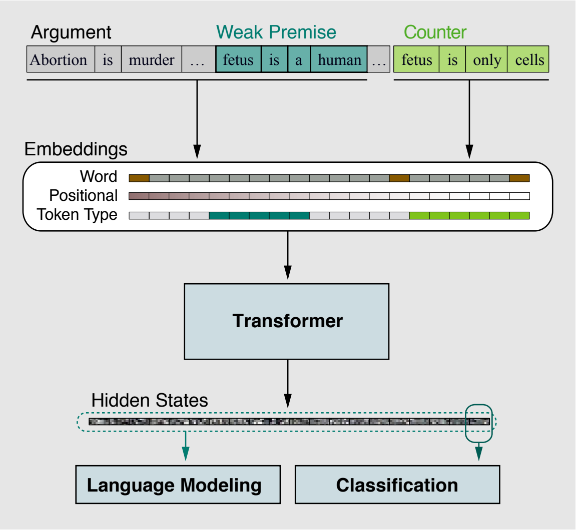

Figure 2: Architecture of our approach: Given an argument, a weak premise, and a counter, three embedding representations are generated and fed to the transformer to obtain hidden states from which the language model and classification heads learn the *Next-token prediction* and *Counter-argument classification* tasks respectively.

using transfer learning and fine-tune a pre-trained transformer-based generation model on our task. In our fine-tuning process, the input is a sequence of tokens created from two segments, the argument and the counter-argument:

[bos] arg_tokens [counter] counter_tokens [eos]

The final token embedding is then a result of concatenating three embeddings: word and positional embeddings learned in the pre-training process, as well as a token-type embedding learned in the fine-tuning process. Here, the token type reflects whether the token belongs to the argument in general, to a weak premise, or the counter-argument. Now, we train our model jointly on two tasks:

- *Next-token prediction.* Given a sequence of tokens, predict the next one.
- *Counter-argument classification.* Given two concatenated segments, decide whether the second is a counter-argument to the first.

The first task is similar to the next-sentence prediction task introduced in (Devlin et al., 2018), which was shown to be beneficial for multiple representation-learning tasks.

Figure 2 shows the architecture of our generation model. For training, we augment a given set of training sequences D by adding distracting sequences. Concretely, we use, for each argument and its weak premise, a non-relevant text instead of the counter-argument. Given a sequence of tokens $d = (t_1, t_2, \cdots, t_n) \in D$, we then optimize the following two loss functions jointly with equal weighting:

$$\begin{aligned} L_1(\Theta) &= \sum_{d \in D} \sum_{t_i \in d} \log P(t_i \mid t_{i-k}, \cdots, t_{i-1}; \Theta), \\ L_2(\Theta) &= \sum_{d_j \in D} \log P(y_j \mid t_1, \cdots, t_n; \Theta), \end{aligned}$$

where Θ denotes the weights of the model, k is the number of previous tokens, and y_j is the ground-truth label of the sequence, indicating if the second segment of the sequence is a counter or not.

4 Data

As proposed, the presented approach models the task of counter-argument generation as an attack on a potentially attackable premise. Such behavior is widely observed on the Reddit forum *changemyview (CMV)*. In particular, a user writes a new *post* that presents reasons supporting the pro or con stance towards a given topic (captured in the *title* of the post), asking the CMV community to challenge the presented view. In turn, other users quote specific segments of the post (usually a few sentences) and seek to counter them in their *comments*. An example has already been given in Figure 1.

The structure induced by CMV defines a suitable data source for our study. Specifically, we create the following distantly-supervised mapping:

- The title of the post denotes the *claim* of the user's argument;
- the text of the post denotes the concatenated set of the argument's *premises*;
- the quoted sentence(s) denote the *attackable (weak) premises*; and
- the quoting sentences from the comment denote the *counter-argument*.

In our work, we build on the CMV dataset of Jo et al. (2020), where each instance contains a post, a title, and a set of attackable sentences (those quoted in the comments). We use the same split as the authors, consisting of 25.8k posts for training, 8.7k for validation, and 8.5k for testing. We extend their dataset by further collecting the quoting sentences from the comments (i.e., the counter-arguments).

The final dataset compiles 111.9k triples of argument (claim and premises), weak premise (one sentence or more), and counter-argument (a set of sentences), split into 67.6k training, 23k validation, and 22.3k test instances.

5 Evaluation

In the following, we present the experiments we carried out to evaluate both steps of our approach individually as well as in a pipelined approach. On the one hand, we aim to assess the applicability of identifying weak premises in an argument and the impact of targeting them in the process of counter-argument generation. On the other hand, our goal is to assess how well counter-argument generation via *undermining* works compared to other known counter-argument generation approaches.

5.1 Weak-Premise Ranking

As presented, we tackled the task of finding attackable premises by learning to rank premises by their weakness with respect to the main claim.

Approach Based on the code of Han et al. (2020) available in the Tensorflow learn-to-rank framework (Pasumarthi et al., 2019), we used a list-wise optimization technique that considers the order of all premises in the same argument.[4] We trained our ranking approach on the CMV dataset's training split and refer to it as *bert-ltr* below.[5]

Baselines We compare our approach to the Bert-based classifier introduced by Jo et al. (2020), trained on the same training split using the authors' code. We employed their trained model to score each premise and then rank all premises in an argument accordingly. We call this the *bert-classifier*. As Jo et al. (2020), we also consider a *random baseline* as well as a baseline that ranks premises based on *sentence length*.

Measures To assess the effectiveness, we follow Jo et al. (2020) in computing the precision of putting a weak premise in the first rank ($P@1$), as well as the accuracy of having at least a weak premise ranked in the top three ($A@3$).

Approach	**P@1**	**A@3**
Random	0.425	0.738
Sentence Length	0.350	0.617
bert-classifier (Jo et al., 2020)	0.487	0.777
bert-ltr (our approach)	***0.506**	***0.786**

Table 2: Weak-premise ranking: Precision of ranking a weak premise highest (P@1) and accuracy for the top three (A@3) of all evaluated approaches. Results with * are significantly better than *bert-classifier* at $p < .05$.

Results Table 2 shows the weak-premise ranking results. We managed to almost exactly reproduce the values of Jo et al. (2020) for all three baselines. Our approach, *bert-ltr*, achieves the best scores according to both measures. In terms of a one-tailed dependent student's t-test, the differences between *bert-ltr* and *bert-classifier* are significant with at least 95% confidence. These results support our hypothesis of the importance of tackling the task as a ranking task with respect to the main claim. Below, we will use our weak-premise ranking model in the overall approach, i.e., to automatically select attackable premises in an argument.

5.2 Premise Attack Generation

Next, we evaluate our hypothesis on the importance of identifying weak premises in the process of counter-argument generation. To focus on this step, we use the ground-truth weak premises in our data. These are the quoted sentences in the post, considered potentially attackable premises.

Approach We used OpenAI's GPT as a pre-trained language model. We trained two versions of our generation model: *our-model-w/* with an extra special token (*[weak]*), surrounding the attackable sentences to give an extra signal to our model, and once *our-model-w/o* without it. We fine-tuned both versions with the same settings using the transformers library (Wolf et al., 2020) for six epochs.[6] We left all other hyperparameters with their default values. As mentioned, the model's input is a sequence of tokens constructed from the argument (with weak premises highlighted) and either the correct counter or a distracting sequence. We selected one sentence from the original post randomly to be the distracting sequence for each input instance.

Baseline We compare our model to a GPT-based model fine-tuned on a sequence of tokens representing a pair of an argument (title and post) and

---

[4] We also experimented with point-wise and pairwise techniques, but the list-wise approach turned out best. For lack of notable insights, we omit to report insights on the others.

[5] Training details can be found in the appendix.

[6] We stopped at six epochs because we observed no gain in terms of validation loss anymore.

#	Approach	Target	Counter Sentences METEOR	Counter Sentences BLEU-1	Counter Sentences BLEU-2	Full Comment METEOR	Full Comment BLEU-1	Full Comment BLEU-2
1	counter-baseline	-	0.058	13.023	3.117	0.097	10.400	3.212
2	our-model-w/o	claim	**0.060**	12.532	2.943	0.090	9.472	2.837
3	our-model-w/o	random premise	0.058	12.838	3.005	0.096	10.398	3.255
4	our-model-w/o	weak premise	0.057	***13.453**	***3.391**	***0.102**	***10.998**	***3.764**
5	our-model-w/	claim	**0.060**	12.635	3.023	0.092	9.685	2.984
6	our-model-w/	random premise	0.059	12.712	2.987	0.096	10.161	3.217
7	our-model-w/	weak premise	0.058	13.162	3.217	0.101	10.743	3.651

Table 3: Premise attack generation: METEOR, BLEU-1, and BLEU-2 scores of the output of each evaluated approach compared to the ground-truth *counter sentences* and to the *the full comment* (i.e., the full counter-argument). Values marked with * are significantly better than *counter-baseline* at $p < .05$.

a counter-argument. We consider this as a general counter-argument generation model, trained without any consideration of weak premises. We trained the baseline using the same setting as our model. We refer to it as *counter-baseline*.

Automatic Evaluation To assess the importance of selecting attackable sentences, we evaluate the effectiveness of our model in different inference settings in terms of what is being attacked: (1) the *claim* of the argument, (2) a *random premise*, or (3) a *weak premise* given in the ground-truth data. For the random setting, we selected three premises from the argument randomly, and we generated one counter for each. The final result is the average of the results for each.

We computed METEOR and BLEU scores, comparing the generated premises to (a) the exact counter sentences of the quoted weak premise and (b) the full argument. We carried out this automatic evaluation on 1k posts from the test split.

Results As shown in Table 3, the best results are achieved by *our-model-w/o* in all cases when identifying the weak premises in the input. Encoding the knowledge about weak premises as token types is sufficient, and adding an extra special token does not help. Although the differences between our best model and the baseline are not big, they are significant according to the one-tailed dependent t-test with a confidence of 95%. For both versions of our model, best scores are achieved when considering the weak premises as the target (except for the first METEOR column). However, not all these differences are significant. This gives evidence that exploiting information about weak premises in the training of counter-argument generation approaches can improve their effectiveness.

To further assess the relationship between the generated counters and the attacked premises, we computed the proportion of covered content tokens in the weak premise for the two versions of our model and the baseline. Figure 3 shows a histogram of the percentages. Clearly, both versions of our model have higher coverage of the annotated weak premises than the baseline.

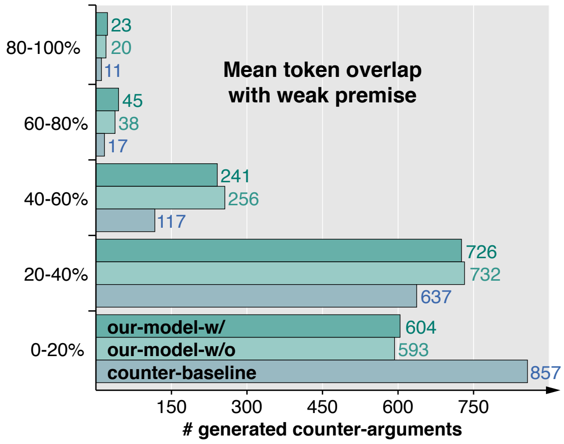

Figure 3: Premise attack generation: Mean token overlap between the ground-truth weak premises and the counters generated by each evaluated approach.

Manual Evaluation To analyze the generated counter-arguments more thoroughly, we carried out a manual evaluation study on a sample of 50 random examples. Two authors of the paper inspected the sample comparing the two versions of our model. The results were in favor of *our-model-w/o*. Therefore, we compared only *our-model-w/o* against the *counter-baseline*. In particular, we assessed the relevance and appropriateness of the output of the two for each example. Given an argument, the highlighted premise to be attacked, and the two counters, we asked three annotators who hold an academic degree and are fluent in English

Approach	Relevance		Appropriateness	
	Majority	Full	Majority	Full
counter-baseline	44%	20%	44%	14%
our-model-w/o	**56%**	**36%**	**56%**	**28%**
Mean Kendall's τ	0.41		0.23	

Table 4: Premise attack generation: Percentage of cases where each given approach was seen as more relevant and more appropriate, respectively, according to majority vote and the full agreement in the manual evaluation on 50 examples. The bottom line shows the mean pairwise inter-annotator agreement.

(no author of this paper) to answer two questions:

1. Which text is more relevant to the highlighted premise?
2. Which text is more appropriate for being used as a counter-argument?

Results As shown in Table 4, considering the majority vote, annotators favored our model in 56% of the cases in both tasks. These results give further evidence supporting our hypothesis of the importance of identifying weak premises.

Considering the given task as a ranking task, we used Kendall's τ to compute the annotator's agreement. The mean pairwise agreement was 0.41 for the relevance assessment and 0.23 for appropriateness. Clearly, assessing the text's appropriateness of being a counter-argument is more subjective and more challenging to judge than the relevance task.

5.3 Overall Approach

Finally, we assess the overall effectiveness of our proposed undermining approach to counter-argument generation, that is, we first identify weak premises automatically using our ranking model, *Bert-ltr*, and then generate a counter-argument using our generation model, *our-model-w/o*, focusing on the selected weak premises.

Approach Due to the limited *P@1* value of our ranking model (see Table 2), we evaluate two variations of our overall approach that differ in terms of what premises to attack. The first variant attacks the weakest premise. In the second, we first generate three counters considering each of the top three weak premises. Then, we select the counter that has the most content-token overlap with the corresponding weak premise.

#	Approach	Target	METEOR	BLEU-1	BLEU-2
1	counter-baseline	None	0.205	22.741	7.792
2	Hua and Wang	None	**0.258**	**30.160**	**13.366**
3	Overall approach	1 premise	0.207	22.841	7.839
4	Overall approach	3 premises	0.210	23.400	8.025

Table 5: Overall approach: METEOR and BLUE scores of the two variants with different attacked targets, the counter-baseline, and Hua and Wang (2019).

Baselines On the one hand, we compare our approach to the *counter-baseline* from the previous section. On the other hand, we consider the state-of-the-art counter-argument generation approach of Hua and Wang (2019), an LSTM-based Seq2seq model with two decoders, one for selecting talking points (phrases) and the other for generating the counter given the selection.

Automatic Evaluation While the approach of Hua and Wang (2019) learns from a dataset collected from the same source (CMV), it requires retrieving relevant argumentative texts with a stance opposite to the input argument. Due to the complexity of the data preparation, we decided instead to evaluate all approaches on the test split of Hua and Wang (2019).[7] As a result, the approach of Hua and Wang (2019) is trained on their training split, whereas our approach is trained on our training split, but both are then evaluated on the same test split of Hua and Wang (2019). This can be considered a somewhat unfair setting for our approach due to certain domain differences, namely, the dataset of Hua and Wang (2019) comprises political topics only. Similar to Section 5.2, we generated counters for 1k examples and computed METEOR and BLEU scores of the generated counters with respect to the ground-truth counters, which are here full arguments (CMV comments).

Results Table 5 shows that our approach outperforms the counter-baseline in both settings, even with weak premises selected automatically. Considering the top-3 weak premises instead of the top-1 improves the results. The best scores are achieved by Hua and Wang (2019), though. A reason for this may be the slight domain difference between our model's training data and the test data used for evaluation. Another observation is that the scores of both our approach and the baseline increase compared to Table 3. This is likely to be caused by the

---

[7] We verified that all posts in their test split do not appear in our training split.

	Correctness	Richness	Grammaticality
Hua and Wang	1.81	2.28	2.91
Overall approach	**2.65**	**3.15**	**3.50**
Krippendorff's α	0.26	0.06	0.32

Table 6: Overall approach: Average scores of the three annotators for the three evaluated quality dimensions of the counter-arguments generated by our approach and the one of Hua and Wang (2019). 1 is worst, 5 is best. The bottom line shows the inter-annotator agreeement.

higher number of ground-truth references for each instance in the dataset of Hua and Wang (2019) compared to the test split of our data, making it more likely to have token overlaps.

Manual Evaluation Given the known limited reliability of automatic generation evaluation, we conducted another user study to evaluate the quality of the generated counters by our model and the approach of Hua and Wang (2019). We evaluate the same quality dimensions the authors used:

- *Content Richness.* The diversity of aspects covered by a counter-argument.
- *Correctness.* The relevance of a counter-argument to the given argument and their degree of disagreement.
- *Grammaticality.* The grammatical correctness and fluency of a counter-argument.

We used the Upwork crowdworking platform to recruit three annotators with English proficiency and experience in editorial work.[8] We asked each of them to evaluate a sample of 100 examples. Each contained an argument (claim and premises) and two counters (one of each approach). We asked the annotators to compare the counters and to assess each with a score from 1 (worst) to 5 (best) for each quality dimension.

Results The results are presented in Table 6. Unlike in the automatic evaluation, the annotators gave, on average, higher scores on all quality dimensions to our generated counters than to those of Hua and Wang (2019).[9] Bringing knowledge from pre-trained language models (GPT) generally seems to contribute to the *grammaticality* and the *richness* of the generated counters. In terms of generating a *correct* counter, focusing the generation model on a specific weak premise in an argument seems to help (2.65 vs. 1.81), even though the results are far from perfect. Manual inspection revealed that far from all generated arguments are counters to exactly what is in the argument, indicating more room to work on this topic.

Claim: there's nothing wrong with income inequality.

Premises: billionaires like the rockefellers and trump worked hard to earn their money and provide their families with luxury. meanwhile, my grandpa didn't and because of that i am of lower-middle class status. *just because i'm poorer doesn't mean i'm entitled to the cash that the rich spent years to accumulate*. i simply have to swallow my pride and start at the bottom . to try and stump income inequality is to meddle with the very basis of pure , unadulterated capitalism and meritocracy

Our Counter: income inequality is precisely because of the fact that rich people have a vested interest in having a bunch of money in the first place, which leads to great wealth inequality. if you are poor but poor you have no incentive to live. by providing good quality goods and services, you are able to contribute to society better than most poor people. you're basically doing something that is expected of you, which is a wonderful trait. i think the problem is that you should at least be able to be generous towards someone without expecting to be rewarded.

Hua and Wang's Counter: this is a great example of how hard it is to invest in low income housing. it's not like it's going to end up being worse for everyone. if you don't like it, you're going to have to worry about it. the rich don't want to pay for it because they do n't have to pay taxes. they aren't going to be able to do anything about it, they just don't want.

Figure 4: Example counter-arguments generated by our approach and by the approach of Hua and Wang (2019). The italicized premise segment was identified as the weak premise by our approach.

The Krippendorff's α values show that the annotators had a fair agreement on *grammaticality* and *correctness* (given the subjectiveness of the tasks), but only slight agreement on *content richness*. We, therefore, think that the results for the latter should not be overinterpreted.

In Figure 4, we show an example argument in favor of *income inequality*. Our approach considers the premise "being poor does not entitle someone to the cash of the rich people". It then generates a counter-argument on the topic of inequality, focusing on the fact that "being poor limits the ability to contribute to society". In contrast, the counter-argument generated by Hua and Wang (2019) diverges to address "low-income housing" which is

---

[8] Upwork, `http://upwork.com`

[9] We note that the scores of Hua and Wang in Table 6 are notably lower than those reported by Hua and Wang (2019). We believe this to be due to the comparison with our approach that affected the annotator's scores.

less relevant to the topic. More examples of generated counters are found in Figure 5 (see appendix).

6 Conclusion

In this work, we have proposed a new approach to counter-argument generation. The approach focuses on argument *undermining* rather than *rebuttal*, aiming to expand the research in this area. The underlying hypothesis is that identifying weak premises in an argument is essential for effective countering. To account for this hypothesis, our approach first ranks the argument's premises by weakness and then generates a counter-argument to attack the weakest ones.

In our experiments, we have first evaluated each step individually. We have observed state-of-the-art results in the weak-premise identification task. Our results also support the need for identifying weak premises to generate better attacks. We have also evaluated the overall approach against the state-of-the-art approach of Hua and Wang (2019). While we did not beat that approach in automatic evaluation scores, independent annotators favored the counter-arguments generated by our approach across all evaluated quality dimensions.

We conclude that our approach improves the state of the art in counter-argument generation in different respects, providing support for our hypothesis. Still, the limited manual evaluation scores imply notable room for improvement. Most importantly, controlling the stance of the generated counters is yet to be fully solved.

Acknowledgments

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901/3) under the project number 160364472.

Ethical Statement

We acknowledge that ethical issues might arise from our work. First, we would like to point out that we did not violate user privacy when using data from public platforms. By reusing pre-trained models, our approach might have inherited some forms of bias that may be encoded in the arguments generated. Mitigating such bias is still subject to ongoing research. It is worth noting that our experiments show that our approach is far from being ready to be used as an end technology. Our goal is to advance the research on this task.

References

Milad Alshomary, Wei-Fan Chen, Timon Gurcke, and Henning Wachsmuth. 2021. Belief-based generation of argumentative claims. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 224–233, Online. Association for Computational Linguistics.

Milad Alshomary, Shahbaz Syed, Martin Potthast, and Henning Wachsmuth. 2020. Target inference in argument conclusion generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4345, Online. Association for Computational Linguistics.

Yonatan Bilu, Daniel Hershcovich, and Noam Slonim. 2015. Automatic claim negation: Why, how and when. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 84–93, Denver, CO. Association for Computational Linguistics.

Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown, and Alyssa Hwang. 2019. AMPERSAND: Argument mining for PERSuAsive oNline discussions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2933–2943, Hong Kong, China. Association for Computational Linguistics.

Wei-Fan Chen, Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2018. Learning to flip the bias of news headlines. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 79–88. Association for Computational Linguistics.

Oana Cocarascu and Francesca Toni. 2017. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1374–1379, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Roxanne El Baff, Henning Wachsmuth, Khalid Al Khatib, Manfred Stede, and Benno Stein. 2019. Computational argumentation synthesis as a language modeling task. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 54–64, Tokyo, Japan. Association for Computational Linguistics.

Shai Gretz, Yonatan Bilu, Edo Cohen-Karlik, and Noam Slonim. 2020. The workweek is the best time to start a family – a study of GPT-2 based claim generation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 528–544, Online. Association for Computational Linguistics.

Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-rank with bert in tf-ranking. *ArXiv*, abs/2004.08476.

Christopher Hidey and Kathleen McKeown. 2019. Fixed that for you: Generating contrastive claims with semantic edits. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1756–1767.

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–230, Melbourne, Australia. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2019. Sentence-level content planning and style specification for neural text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 591–602.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8465–8472.

Yohan Jo, Seojin Bang, Emaad Manzoor, Eduard Hovy, and Chris Reed. 2020. Detecting attackable sentences in arguments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–23, Online. Association for Computational Linguistics.

Matan Orbach, Yonatan Bilu, Ariel Gera, Yoav Kantor, Lena Dankin, Tamar Lavee, Lili Kotlerman, Shachar Mirkin, Michal Jacovi, Ranit Aharonov, et al. 2019. A dataset of general-purpose rebuttal. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5595–5605.

Matan Orbach, Yonatan Bilu, Assaf Toledo, Dan Lahav, Michal Jacovi, Ranit Aharonov, and Noam Slonim. 2020. Out of the echo chamber: Detecting countering debate speeches. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7073–7086, Online. Association for Computational Linguistics.

Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2970–2978.

Andreas Peldszus and Manfred Stede. 2015. Towards detecting counter-considerations in text. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 104–109.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Misa Sato, Kohsuke Yanai, Toshinori Miyoshi, Toshihiko Yanase, Makoto Iwayama, Qinghua Sun, and Yoshiki Niwa. 2015. End-to-end argument generation system in debating. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 109–114.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Aspect-controlled neural argument generation. *arXiv preprint arXiv:2005.00084*.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187, Valencia, Spain. Association for Computational Linguistics.

Henning Wachsmuth, Manfred Stede, Roxanne El Baff, Khalid Al Khatib, Maria Skeppstedt, and Benno Stein. 2018a. Argumentation synthesis following rhetorical strategies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3753–3765. Association for Computational Linguistics.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018b. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251. Association for Computational Linguistics.

Douglas Walton. 2009. Objections, rebuttals and refutations.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020.

Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

A Models and Training Specifications

All our models were trained on one *24 GeForce GTX 1080 GPU*.

Weak-Premise Ranking We use the code of (Han et al., 2020) to train our ranking model *Bert-ltr*, with BERT-Base, Uncased for embedding (12-layer, 768-hidden, 12-heads, 110M parameters). The model was trained for 100k steps, which took almost 9 hours of training.

Premise Attack Generation We use the transformers library to train our generation models, with GPT as a pre-training language model (12-layer, 768-hidden, 12-heads, 110M parameters. OpenAI GPT English model). The model was fine-tuned for six epochs, which took almost one day of training. In the generation time, we use top-k sampling technique (Huang et al., 2019), with the following parameters: top-k=50, top-p=0.95, and temperature=1.0. We generate counters of a minimum of 100 tokens and a maximum of 150 tokens.

Evaluation Measures The BLEU and METEOR scores are computed using the NLTK library (`https://www.nltk.org/`). Significance tests were performed using the Scipy library (`https://www.scipy.org/`).

B Example Counters

In this section, we present example counters generated by our approach and Hua and Wang (2019)

Claim (title): it’s better to have no freedom of speech than to be given the allusion of freedom of speech.

Premises (sentences): the reason why i believe this is that i like to know the rules first upfront , i hate people changing the rules in the middle of the game . the simplest comparison i can come up with is that if i am walking and then i accidentally walk into someone ’s property than i get arrested for trespassing . i rather they straight up hang signs that say i am entering private property in the first place . *if i go my whole life believing i have free speech , then one day i get maced or tasered by a cop for speaking my mind* . i would ’ve expressed my opinions in another way . these countries where they say you have freedom but on the other hand oppress you are not free countries nor are they honest countries . in my view , they are worse than an openly oppressive country .

Our Counter: can you expand on what you mean here? this is simply demonstrably false. we have free speech. people have free speech in these countries because we are free people. i’m not sure how you got from you that you dislike freedom, but there’s no contradiction. i am not free of speech.

Hua and Wang’s Counter: if you believe in the right to free speech , then it is not a right . if you do n’t agree with what you say , then you have to accept that it is a slippery slope . it ’s not that you are n’t allowed to say whatever you want , it ’s that you do not have the right , and that you should not be allowed to speak freely . you can not have a right to express your views , but that does n’t mean you should n’t do it . if you

Claim (title): i think hijab is good for the society and for women.

Premises (sentences): but i think that hijab is a good approach to reduce rape and harassment . *i think one of the reasons for rape are 1) not having a sexual partner 2) pornography , so here comes the role of the hijab so if women wear hijab rapists will not view her as a sexual object* . here is a video that shows a social experiment of the same woman wearing different cloth on the same streets of nyc :[– url –) . i want people that view hijab as some sort of backward way of dress to tell me why do you think like that ? and if possible change my view . (edit) .... . i agree with many of you guys that said rape is about power , but what i meant in my question is sexual harassment in general , by that i mean touching or grabbing private parts i could n’t explain at first because of my english . so if touching and grabbing is under the crime of sexual harassment , do n’t you think hijab will reduce those assaults ? so in short my point of view has changed when you guys said that rape is about power , i know think that a rapist will rape anyone regardless his cloth , but that does n’t s include other sexual harassment means . excuse my english .

Our Counter: this point you’re making is’why do you think women are raped if you’d like to do x with them?’i’m not sure what you’re trying to get at here. are you saying people shouldn’t think rape is a’bad idea ’? because that’s the most basic and fundamental reason a person should wear a hijab, and unless you don’t believe in the concept of a’good idea ’, you shouldn’t think women are sexually harassed. rape is, in general, about power, sexual harassment, by the very definition of harassment.

Hua and Wang’s Counter: there is no such thing as a " feminist " movement . it ’s not just a matter of what it is , it ’s a group of people who are in favor of it . women are not the only ones who are oppressed . women have been subject to the same rules as men . women are

Claim (title): there is no logical way to prove that being unfair is wrong .

Premises (sentences): . one response was that " the party did n’t deserve that type of treatment . " but that ’s just another phrasing of being unfair . i decided to think for myself logically why being unfair is wrong and so far i ’ve only managed to come up with a few flawed answers . firstly , being unfair is self-evidently wrong . now this works out , until you realise that different people in different environments would find different values to be self-evident . *for example , if a human were to grow up alone without being in contact with another human since he was first conscious , then what he would n’t find fairness to be self-evident* . instead , what he wo n’t hesitate to do is kill others for whatever reasons he sees fit . he would see what he does as being acceptable , but we would n’t . however , it would be impossible to convince him that others have a right to life because he grew not knowing empathy . if we apply this to the current context , then people in the west find different morals to be self-evident than people in asia or the middle east . yet everyone claims the other is inhumane , with no explanation how it is inhumane , or what is inhumane . another answer why being unfair is wrong is that it without fairness , society would n’t function optimally . however if i purge the retired elderly or the ill who needlessly consume resources , then it would boost the cogs of society , wouldnt it ? its still considered wrong . therefore this answer is invalid . anyone have answers for the question " why is being unfair wrong ? "

Our Counter: how is being unfair any better than being wrong? fairness is subjective. in any society, fairness is subjective. if a person has a problem, does that mean their position is fair? the way we live the consequences of their decision means we can’t change them. but why is that wrong?

Hua and Wang’s Counter: i think it ’s important to distinguish between the two scenarios , and i think that it ’s more important to understand what you mean by " different " . i think you ’re correct , but i think it

Figure 5: A list of examples of counter-arguments generated by our approach and by the approach of Hua and Wang (2019). The italicized premise segment was identified as the weak premise by our approach.

Alternated Training with Synthetic and Authentic Data for Neural Machine Translation

Rui Jiao[1,3,4], **Zonghan Yang**[1,3,4], **Maosong Sun**[1,3,4,5], **and Yang Liu**[*] [1,2,3,4,5]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Institute for AI Industry Research, Tsinghua University, Beijing, China
[3]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[4]Beijing National Research Center for Information Science and Technology
[5]Beijing Academy of Artificial Intelligence

Abstract

While synthetic bilingual corpora have demonstrated their effectiveness in low-resource neural machine translation (NMT), adding more synthetic data often deteriorates translation performance. In this work, we propose alternated training with synthetic and authentic data for NMT. The basic idea is to alternate synthetic and authentic corpora iteratively during training. Compared with previous work, we introduce authentic data as guidance to prevent the training of NMT models from being disturbed by noisy synthetic data. Experiments on Chinese-English and German-English translation tasks show that our approach improves the performance over several strong baselines. We visualize the BLEU landscape to further investigate the role of authentic and synthetic data during alternated training. From the visualization, we find that authentic data helps to direct the NMT model parameters towards points with higher BLEU scores and leads to consistent translation performance improvement.

1 Introduction

While recent years have witnessed the rapid development of Neural Machine Translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017), it heavily relies on large-scale, high-quality bilingual corpora. Due to the expense and scarcity of authentic corpora, synthetic data has played a significant role in boosting translation quality (He et al., 2016; Sennrich et al., 2016a; Zhang and Zong, 2016; Cheng et al., 2016; Fadaee et al., 2017).

Existing approaches to synthesizing data in NMT focus on leveraging monolingual data in the training process. Among them, back-translation (BT) (Sennrich et al., 2016a) has been widely used to generate synthetic bilingual corpora by using a trained target-to-source NMT model to translate large-scale target-side monolingual corpora. Such synthetic data can be used to improve source-to-target NMT models. Despite the effectiveness of back-translation, the synthetic data inevitably contains noise and erroneous translations. As a matter of fact, it has been widely observed that while BT is capable of benefiting NMT models by using relatively small-scale synthetic data, further increasing the quantity often deteriorates translation performance (Edunov et al., 2018; Wu et al., 2019; Caswell et al., 2019).

This problem has attracted increasing attention in the NMT community (Edunov et al., 2018; Wang et al., 2019). One direction to alleviate the problem is to add noise or a special tag on the source side of synthetic data, which enables NMT models to distinguish between authentic and synthetic data (Edunov et al., 2018; Caswell et al., 2019). Another direction is to filter or evaluate the synthetic data by calculating confidence over corpora, making NMT models better exploit synthetic data (Imamura et al., 2018; Wang et al., 2019). While these methods have outperformed the conventional BT approach, NMT models still suffer from a performance degradation as the size of synthetic data keeps increasing. Hence, how to better take advantage of limited authentic data and abundant synthetic data still remains a grand challenge.

In this work, we propose alternated training with synthetic and authentic data for neural machine translation. The basic idea is to alternate synthetic and authentic corpora iteratively during training. Compared with previous work, we introduce authentic data as guidance to prevent the training of NMT models from being disturbed by noisy synthetic data. Our approach is inspired by the characterization of synthetic and authentic corpora as two types of different approximations for the dis-

---

[*] Corresponding author: Yang Liu

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1828–1834
August 1–6, 2021.

tribution of infinite authentic data. We visualize the BLEU landscape to further investigate the role of authentic and synthetic data during alternated training. We find that the authentic data helps to direct NMT model parameters towards the points with higher BLEU scores. Experiments on Chinese-English translation tasks show that our approach improves the performance over strong baselines.

2 Alternated Training

Let $\mathbf{x}$ be a source sentence and $\mathbf{y}$ be a target sentence. We use $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ to denote an NMT model parameterized by $\boldsymbol{\theta}$. Let $D_a = \{\langle \mathbf{x}_n, \mathbf{y}_n \rangle\}_{n=1}^{N}$ be an authentic parallel corpus containing N sentence pairs. Traditional NMT aims to obtain $\hat{\boldsymbol{\theta}}_a$ that maximizes the log-likelihood on D_a:

$$\hat{\boldsymbol{\theta}}_a = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \frac{1}{N} \sum_{n=1}^{N} \log P(\mathbf{y}_n|\mathbf{x}_n;\boldsymbol{\theta}) \right\}. \quad (1)$$

Back-translation generates additional synthetic parallel data from the monolingual corpus. Let $D_m = \{\mathbf{y}_m\}_{m=1}^{M}$ be a monolingual corpus containing M target-side sentences. Back-translation first trains a target-to-source model $\hat{\boldsymbol{\theta}}_{\mathrm{BT}}$ on D_a:

$$\hat{\boldsymbol{\theta}}_{\mathrm{BT}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \frac{1}{N} \sum_{n=1}^{N} \log P(\mathbf{x}_n|\mathbf{y}_n;\boldsymbol{\theta}) \right\}, \quad (2)$$

which is then used to translate each sentence in the target-side monolingual corpus D_m:

$$\hat{\mathbf{x}}_m = \underset{\mathbf{x}}{\operatorname{argmax}} \left\{ P(\mathbf{x}|\mathbf{y}_m;\hat{\boldsymbol{\theta}}_{\mathrm{BT}}) \right\}, \quad (3)$$

where $m = 1, \dots, M$. The synthetic corpus D_s is generated by pairing the translations $\{\hat{\mathbf{x}}_m\}_{m=1}^{M}$ with D_m, i.e. $D_s = \{\langle \hat{\mathbf{x}}_m, \mathbf{y}_m \rangle\}_{m=1}^{M}$. The required source-to-target model is finally trained on the combination of authentic and synthetic data:

$$\hat{\boldsymbol{\theta}}_s = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \frac{1}{N+M} \left(\sum_{n=1}^{N} \log P(\mathbf{y}_n|\mathbf{x}_n;\boldsymbol{\theta}) + \sum_{m=1}^{M} \log P(\mathbf{y}_m|\hat{\mathbf{x}}_m;\boldsymbol{\theta}) \right) \right\}. \quad (4)$$

Suppose that there exists infinite authentic parallel data, which can be characterized as distribution $p(\mathbf{x},\mathbf{y})$. Synthesizing the large-scale corpus D_s is to better approach the authentic parallel data distribution. Furthermore, the finite corpora D_a and $D_s \cup D_a$ can be viewed as different empirical approximations of $p(\mathbf{x},\mathbf{y})$:

$$p_a(\mathbf{x},\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \delta_{\langle \mathbf{x}_n,\mathbf{y}_n \rangle \in D_a}(\mathbf{x},\mathbf{y}), \quad (5)$$

$$p_s(\mathbf{x},\mathbf{y}) = \frac{1}{N+M} \left(\sum_{n=1}^{N} \delta_{\langle \mathbf{x}_n,\mathbf{y}_n \rangle \in D_a}(\mathbf{x},\mathbf{y}) + \sum_{m=1}^{M} \delta_{\langle \hat{\mathbf{x}}_m,\mathbf{y}_m \rangle \in D_s}(\mathbf{x},\mathbf{y}) \right), \quad (6)$$

where δ represents the Dirac distribution. On the one hand, D_a is considered to be of higher quality as $\lim_{N\to\infty} p_a(\mathbf{x},\mathbf{y}) = p(\mathbf{x},\mathbf{y})$ exactly recovers the authentic data distribution. On the other hand, although D_s contains certain noise (as $\lim_{M\to\infty} p_s(\mathbf{x},\mathbf{y}) \neq p(\mathbf{x},\mathbf{y})$), it provides more diversified data samples that enable the NMT model to reconstruct the global distribution. As the two corpora are complementary to each other, we introduce authentic data periodically during the training process with synthetic data. Intuitively, alternated training using authentic corpora helps to rectify the deviation of training direction affected by the noisy synthetic data and enhances model performance.

Algorithm 1 Alternated Training for NMT

```
Input: Synthetic data $D_s = \{\langle \hat{\mathbf{x}}_m, \mathbf{y}_m \rangle\}_{m=1}^{M}$,
    Authentic data $D_a = \{\langle \mathbf{x}_n, \mathbf{y}_n \rangle\}_{n=1}^{N}$
Output: $\hat{\boldsymbol{\theta}}_{\mathrm{alter}}$
Set $\hat{\boldsymbol{\theta}}_a^{(0)}$ as random initialization;
$t \leftarrow 0$;
while Not Converged do
    Obtain $\hat{\boldsymbol{\theta}}_s^{(t+1)}$ on $D_s \cup D_a$ with $\hat{\boldsymbol{\theta}}_a^{(t)}$ as the starting point using Eq. (4);   ▷ S-Step
    Obtain $\hat{\boldsymbol{\theta}}_a^{(t+1)}$ on $D_a$ with $\hat{\boldsymbol{\theta}}_s^{(t+1)}$ as the starting point using Eq. (1);   ▷ A-Step
    $t \leftarrow t + 1$;
end while
return $\hat{\boldsymbol{\theta}}_{\mathrm{alter}} = \hat{\boldsymbol{\theta}}_a^{(t)}$.
```

Our proposed alternated training approach is shown in Algorithm 1. Starting with random initialization, each alternation cycle during training consists of two steps. For the t-th cycle, the first step is to finetune the model $\hat{\boldsymbol{\theta}}_a^{(t)}$ with Eq. (4) on $D_s \cup D_a$ until convergence[1] to obtain $\hat{\boldsymbol{\theta}}_s^{(t+1)}$, which

---

[1] We also attempted to train S/A-steps for certain iterations. Empirically, the proposed convergence-based method performed better.

is referred as **S-Step** (line 4). The second step is to alter the training data back to D_a and finetune $\hat{\boldsymbol{\theta}}_s^{(t+1)}$ with Eq. (1) until convergence to obtain $\hat{\boldsymbol{\theta}}_a^{(t+1)}$, which is referred as **A-Step** (line 5). We alternate the training process until convergence. It is noted that back-translation is equivalent to a single S-Step performed in our approach.

3 Experiments

3.1 Setup

We evaluated our training strategy on Chinese-English and German-English translation tasks. We reported the tokenized BLEU score as calculated by multi-bleu.perl.

For the Chinese-English task, we extracted 1.25M parallel sentence pairs from LDC as our authentic bilingual corpus and 10M English-side sentences from WMT17 Chinese-English training set as our monolingual corpus for back-translation. NIST06 was used as the validation set. We use NIST02, 03, 04, 05 and 08 datasets as test sets. For the German-English task, we selected the dataset of IWSLT14 German-English task, which contains 16k parallel sentence pairs for training. We further extracted 4.5M English-side sentences from WMT14 German-English training set as monolingual dataset. We segmented Chinese sentences by THULAC (Sun et al., 2016) and tokenized English and German sentences by Moses (Koehn et al., 2007). The vocabulary was built by Byte Pair Encoding (BPE) (Sennrich et al., 2016b) with 32k merge operations. We used Transformer (Vaswani et al., 2017) implemented in THUMT (Tan et al., 2020) with standard hyperparameters as a base model. We used Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ with the maximum learning rate $= 7 \times 10^{-4}$.

We applied early-stopping to verify convergence of each single S/A-step. If the validation BLEU failed ti exceed the highest score during the certain S/A-step after 10K training iterations, we consider the model converged and alternated the training set. For the whole training process, we set the maximum training iterations as 250k for Chinese-English task and 150k for German-English task.

3.2 Results

Figure 1 shows the comparison among several approaches in different scales of training sets on the Chinese-English task. The leftmost point is trained on the authentic data, and other points are trained on the combination of authentic and synthetic corpora. The X-axis shows the synthetic data scale ranging from 1.25M (the size of authentic data) to 10M (the full size of the monolingual corpus). The Y-axis shows the BLEU scores of the combined test set. We find that the performance of BT rises firstly but then decreases as more synthetic data is added, which confirms the findings of Wu et al. (2019). In contrast, our approach achieves consistent improvement with the enlargement of the synthetic data scale.

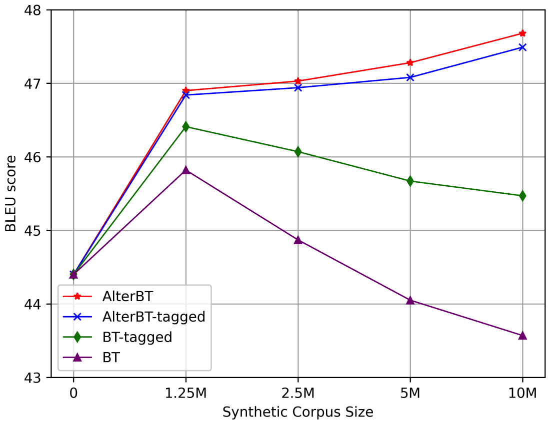

Figure 1: Comparison with several baselines in different data scale. Our alternated approach outperforms the conventional back-translation method and improves the performance of Tagged BT. Moreover, with the enlargement of the synthetic data scale, the BLEU score rises steadily by alternated training.

Table 1 shows the detailed translation performance on the Chinese-English task when the synthetic data scale is set to 10M. It can be seen that our alternated training strategy outperforms conventional back-translation and tagged back-translation on all test sets. We find that during training, the S-Steps account for about 73% of the total training time, and the A-Steps account for 27%. This finding suggests that our training procedure composes mainly of S-Steps, and moderate A-Steps are efficient to guide the NMT model towards better points, which lead to the improvement of BLEU performance.

Table 2 shows the results of the German-English task. Similar to the Chinese-English task, we vary the synthetic data scale from 1M to 4.5M for experiments. We find that the performance degradation also occurs while utilizing large-scale synthetic data, and alternated training approach alleviate the problem and perform better than corresponding baselines.

Data	NIST06	NIST02	NIST03	NIST04	NIST05	NIST08	All
Base	45.94	45.82	45.35	46.88	45.43	36.98	44.40
BT	43.89	44.79	44.40	46.24	45.45	36.45	43.57
BT-tagged	46.79	47.11	46.49	47.73	47.17	38.41	45.47
AlterBT	49.07$^{+\dagger}$	48.77$^{+\dagger}$	48.36$^{+\dagger}$	**49.51**$^{+\dagger}$	**49.94**$^{+\dagger}$	**40.95**$^{+\dagger}$	**47.68**$^{+\dagger}$
AlterBT-tagged	**49.40**$^{+\dagger}$	**49.04**$^{+\dagger}$	**48.37**$^{+\dagger}$	49.10$^{+\dagger}$	49.64$^{+\dagger}$	40.56$^{+\dagger}$	47.49$^{+\dagger}$

Table 1: BLEU scores on the NIST Chinese-English task with 10M additional synthetic corpus. "Base" means only authentic data is used. "BT" corresponds to the back-translation method (Sennrich et al., 2016a). "BT-tagged" corresponds to the tagged BT technique proposed by Caswell et al. (2019). "AlterBT" means alternated training on authentic data and synthetic data using "BT" in each alternation. "AlterBT-tagged" means alternated training on authentic data and synthetic data using "BT-tagged" in each alternation. "$+$" means significantly better than BT ($p < 0.01$)."†" means significantly better than BT-tagged ($p < 0.01$).

Scale	1M	4.5M
Base	34.16	34.16
BT	37.36	36.30
BT-tagged	37.65	37.42
AlterBT	**38.20**$^{+\dagger}$	38.53$^{+\dagger}$
AlterBT-tagged	37.98$^{+\dagger}$	**39.19**$^{+\dagger}$

Table 2: BLEU scores on the IWSLT14 German-English task with 1M and 4.5M additional synthetic corpus. "$+$" means significantly better than BT ($p < 0.01$)."†" means significantly better than BT-tagged ($p < 0.01$).

3.3 BLEU Landscape Visualization

To validate the assumption that the authentic data helps to rectify the deviation in synthetic data and redirect the NMT model parameters to a better optimization path, we further investigate the BLEU landscape to compare our method with the BT approach during the same training steps.

The visualization of the BLEU landscape is shown in Figure 2. Checkpoints during alternated training are projected onto the 2D plane defined by $\hat{\boldsymbol{\theta}}_s^{(t)}$, $\hat{\boldsymbol{\theta}}_a^{(t)}$ and $\hat{\boldsymbol{\theta}}_s^{(t+1)}$ [2]. Our projection method considers both the model parameters and their translation performance (See Appendix A for details). For the conventional BT approach, the model parameters are stuck in an inefficient optimization path (highlighted in blue dashed lines). In our approach, we find that authentic data effectively guides the model towards a better direction for A-Step (highlighted in red solid lines). For S-Step (highlighted in red dashed lines), although training with synthetic data deteriorates the BLEU performance, it pushes the model away from the original route, and enables authentic data to further redirect the model into a better point with a higher BLEU score.

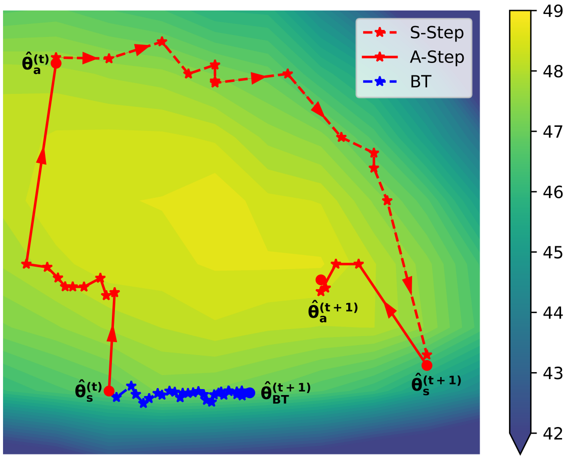

Figure 2: Visualization of BLEU landscape on NIST06 dataset defined by $\hat{\boldsymbol{\theta}}_s^{(t)}$, $\hat{\boldsymbol{\theta}}_a^{(t)}$ and $\hat{\boldsymbol{\theta}}_s^{(t+1)}$. The projected checkpoints are represented as stars. Starting from $\hat{\boldsymbol{\theta}}_s^{(t)}$, the red dashed and solid segments represent S-Step and A-Step in our method, respectively. The blue dashed segments illustrate the conventional BT method, which shares the same starting point $\hat{\boldsymbol{\theta}}_s^{(t)}$ with ours. $\hat{\boldsymbol{\theta}}_{\mathrm{BT}}^{(t+1)}$ denotes the BT model trained the same steps as $\hat{\boldsymbol{\theta}}_a^{(t+1)}$. It is shown that alternated training guides the model from $\hat{\boldsymbol{\theta}}_s^{(t)}$ to $\hat{\boldsymbol{\theta}}_a^{(t)}$, $\hat{\boldsymbol{\theta}}_s^{(t+1)}$ and $\hat{\boldsymbol{\theta}}_a^{(t+1)}$ successively, which finally leads to a better point with a higher BLEU score.

4 Related Work

Our work is based on back-translation (BT), an approach to leverage monolingual data by an additional target-to-source system. BT was proved to be effective in neural machine translation (NMT) systems (Sennrich et al., 2016a). Despite its effectiveness, BT is limited by the accuracy of synthetic data. Noise and translation errors hinder the boosting of model performance (Hoang et al., 2018). The

---

[2] We select $t = 2$ for this visualization, and similar performance can be observed for other t's.

negative results become more evident when more synthetic data is mixed into training data (Caswell et al., 2019; Wu et al., 2019).

Considerable studies have focused on the accuracy problem in synthetic data and further extended back-translation. Imamura et al. (2018) demonstrate that generating source sentences via sampling increases the diversity of synthetic data and benefits the BT system. Edunov et al. (2018) further propose a noisy beam search method to generate more diversified source-side data. Caswell et al. (2019) add a reserved token to synthetic source-side sentences in order to help NMT model distinguish between authentic and synthetic data. Another perspective aims at measuring the translation quality of synthetic data. Imamura et al. (2018) filter sentence pairs with low likelihood or low confidence. Wang et al. (2019) use uncertainty-based confidence to measure words and sentences in synthetic data. Different from the aforementioned works, our approach introduces neither data modification (e.g. noising or tagging) nor additional models for evaluation. We alternate training set on the original authentic and synthetic data.

The work relatively close to ours is Iterative Back-Translation (Hoang et al., 2018), which refines forward and backward model via back-translation data, and regenerates more accurate synthetic data from monolingual data. Our work differs from Iterative BT in that we do not require source-side monolingual corpora or repeatedly fine-tune the backward model.

5 Conclusion

In this work, we propose alternated training with synthetic and authentic data for neural machine translation. Experiments have shown the supremacy of our approach. Visualization of the BLEU landscape indicates that alternated training guides the NMT model towards better points.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No.61925601, No.61772302). We thank all anonymous reviewers for their valuable comments and suggestions on this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 1: Research Papers*, pages 53–63. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 820–828.

Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018*, pages 18–24. Association for Computational Linguistics.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018*, pages 55–63. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016. Thulac: An efficient lexical analyzer for chinese.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Zhixing Tan, Jiacheng Zhang, Xuancheng Huang, Gang Chen, Shuo Wang, Maosong Sun, Huanbo Luan, and Yang Liu. 2020. THUMT: An open-source toolkit for neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 116–122, Virtual. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019. Improving back-translation with uncertainty-based confidence estimation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 791–802. Association for Computational Linguistics.

Lijun Wu, Yiren Wang, Yingce Xia, QIN Tao, Jianhuang Lai, and Tie-Yan Liu. 2019. Exploiting monolingual data at scale for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4198–4207.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1535–1545. The Association for Computational Linguistics.

A Method for Visualization

We first define the projection plane S by parameters $\hat{\boldsymbol{\theta}}_s^{(t)}$, $\hat{\boldsymbol{\theta}}_a^{(t)}$ and $\hat{\boldsymbol{\theta}}_s^{(t+1)}$. Selecting $\hat{\boldsymbol{\theta}}^* = \hat{\boldsymbol{\theta}}_s^{(t)}$ as the basic point and $\boldsymbol{\delta} = \hat{\boldsymbol{\theta}}_a^{(t)} - \hat{\boldsymbol{\theta}}_s^{(t)}, \boldsymbol{\eta} = \hat{\boldsymbol{\theta}}_s^{(t+1)} - \hat{\boldsymbol{\theta}}_s^{(t)}$ as two basis vectors, we plot the function $f(x, y) = \text{BLEU}(D_{\text{DEV}}; \hat{\boldsymbol{\theta}}^* + x\boldsymbol{\delta} + y\boldsymbol{\eta})$ in the 2D surface. We calculate the BLEU scores for all NMT models defined by grid points on the projection plane, and construct the BLEU contours via linear interpolation in MATPLOTLIB (Hunter, 2007).

We project the model checkpoints onto the 2D plane S to represent the parameter geometry and their translation performance. As the 2D contour plane consists of several regions corresponded with different BLEU ranges, we formulate the visualization task into the following problem:

$$(x_i, y_i) = \underset{(x,y)\in S_i}{\text{argmin}} \left\| \hat{\boldsymbol{\theta}}_i - \left(\hat{\boldsymbol{\theta}}^* + x\boldsymbol{\delta} + y\boldsymbol{\eta} \right) \right\|_F^2, \tag{7}$$

where S_i denotes the BLEU region that the performance of $\hat{\boldsymbol{\theta}}_i$ lies in. It is noted that according to the Pythagorean theorem, for $\forall\ (\tilde{x}, \tilde{y}) \in S_i$,

$$\begin{aligned} & \left\| \hat{\boldsymbol{\theta}}_i - \left(\hat{\boldsymbol{\theta}}^* + \tilde{x}\boldsymbol{\delta} + \tilde{y}\boldsymbol{\eta} \right) \right\|_F^2 \\ = & \left\| \hat{\boldsymbol{\theta}}_i - \left(\hat{\boldsymbol{\theta}}^* + \hat{x}_i\boldsymbol{\delta} + \hat{y}_i\boldsymbol{\eta} \right) \right\|_F^2 + \\ & \left\| \left(\hat{\boldsymbol{\theta}}^* + \hat{x}_i\boldsymbol{\delta} + \hat{y}_i\boldsymbol{\eta} \right) - \left(\hat{\boldsymbol{\theta}}^* + \tilde{x}\boldsymbol{\delta} + \tilde{y}\boldsymbol{\eta} \right) \right\|_F^2, \end{aligned} \tag{8}$$

where

$$(\hat{x}_i, \hat{y}_i) = \underset{(x,y)\in S}{\text{argmin}} \left\| \hat{\boldsymbol{\theta}}_i - \left(\hat{\boldsymbol{\theta}}^* + x\boldsymbol{\delta} + y\boldsymbol{\eta} \right) \right\|_F^2. \tag{9}$$

As $(x_i, y_i) \in S_i$, we can substitute $(\tilde{x}, \tilde{y})$ in Eq. (8) with (x_i, y_i). Notice that the first term on the right-hand side of Eq. (8) is independent of (x_i, y_i), the minimizer (x_i, y_i) thus satisfies the following conditions:

$$\begin{aligned} (x_i, y_i) = \underset{(\tilde{x},\tilde{y})\in S_i}{\text{argmin}} \Bigg\| & \left(\hat{\boldsymbol{\theta}}^* + \hat{x}_i\boldsymbol{\delta} + \hat{y}_i\boldsymbol{\eta} \right) \\ & - \left(\hat{\boldsymbol{\theta}}^* + \tilde{x}\boldsymbol{\delta} + \tilde{y}\boldsymbol{\eta} \right) \Bigg\|_F^2, \end{aligned} \tag{10}$$

with $(\hat{x}_i, \hat{y}_i)$ satisfying Eq. (9).

According to Eq. (8), our projection method can be divided into two steps. The first step is to calculate $(\hat{x}_i, \hat{y}_i)$ in Eq. (9), which minimizes the first term of Eq. (8). By the least square method, we obtain the analytic solution to $(\hat{x}_i, \hat{y}_i)$ as follows:

$$\begin{cases} x_i &= \frac{VB-UC}{B^2-AC}, \\ y_i &= \frac{UB-VA}{B^2-AC}, \end{cases} \tag{11}$$

where

$$\begin{aligned} A &= \langle \boldsymbol{\delta}, \boldsymbol{\delta} \rangle, \\ B &= \langle \boldsymbol{\delta}, \boldsymbol{\eta} \rangle, \\ C &= \langle \boldsymbol{\eta}, \boldsymbol{\eta} \rangle, \\ U &= \langle \hat{\boldsymbol{\theta}}_i - \hat{\boldsymbol{\theta}}^*, \boldsymbol{\delta} \rangle, \\ V &= \langle \hat{\boldsymbol{\theta}}_i - \hat{\boldsymbol{\theta}}^*, \boldsymbol{\eta} \rangle. \end{aligned} \tag{12}$$

The second step is to solve (x_i, y_i) in Eq. (10), which minimizes the second term of Eq. (8). Specially, we have $(x_i, y_i) = (\hat{x}_i, \hat{y}_i)$ if $(\hat{x}_i, \hat{y}_i) \in S_i$. Otherwise, as the BLEU region S_i is enclosed by polygon boundaries with limited edges, we simply calculate the distance between $(\hat{x}_i, \hat{y}_i)$ and each edge and select the minimum one. The boundary point minimizing the distance is then determined as (x_i, y_i). We cast the projection point from $(\hat{x}_i, \hat{y}_i)$ to (x_i, y_i) in order to restore the origin BLEU performance of $\hat{\boldsymbol{\theta}}_i$.

Template-Based Named Entity Recognition Using BART

Leyang Cui$^{\dagger\diamond\heartsuit}$, **Yu Wu**‡, **Jian Liu**$^{\diamond\heartsuit}$, **Sen Yang**$^{\diamond\heartsuit}$, **Yue Zhang**$^{\diamond\heartsuit}$ *
†Zhejiang University
‡Microsoft Research Asia
$^{\diamond}$School of Engineering, Westlake University
$^{\heartsuit}$Institute of Advanced Technology, Westlake Institute of Advanced Study
{cuileyang,liujian,yangsen,zhangyue}@westlake.edu.cn Wu.Yu@microsoft.com

Abstract

There is a recent interest in investigating few-shot NER, where the low-resource target domain has different label sets compared with a resource-rich source domain. Existing methods use a similarity-based metric. However, they cannot make full use of knowledge transfer in NER model parameters. To address the issue, we propose a template-based method for NER, treating NER as a language model ranking problem in a sequence-to-sequence framework, where original sentences and statement templates filled by candidate named entity span are regarded as the source sequence and the target sequence, respectively. For inference, the model is required to classify each candidate span based on the corresponding template scores. Our experiments demonstrate that the proposed method achieves 92.55% F1 score on the CoNLL03 (rich-resource task), and significantly better than fine-tuning BERT 10.88%, 15.34%, and 11.73% F1 score on the MIT Movie, the MIT Restaurant, and the ATIS (low-resource task), respectively.

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing, which identifies mention spans from text inputs according to pre-defined entity categories (Tjong Kim Sang and De Meulder, 2003), such as location, person, organization, etc. The current dominant methods use a sequential neural network such as BiLSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2019) is used to represent the input text, and softmax (Chiu and Nichols, 2016; Strubell et al., 2017; Cui and Zhang, 2019) or CRF (Lample et al., 2016; Ma and Hovy, 2016; Luo et al., 2020) output layers to assign named entity tags (e.g. organization, person and location) or non-entity tags on each input token. Such a system is illustrated in Figure 2(a).

Figure 1: Example of NER on different domains.

Neural NER models require large labeled training data, which can be available for certain domains such as news, but scarce in most other domains. Ideally, it would be desirable to transfer knowledge from the resource-rich news domain so that a model can be used in target domains based on a few labeled instances. In practice, however, a challenge is that entity categories can be different across different domains. As shown in Figure 1, the system is required to identify *location* and *person* in the news domain, but *character* and *title* in the movie domain. Both a softmax layer and CRF layer require a consistent label set between training and testing. As a result, given a new target domain, the output layer needs adjustment and training must be conducted again using both source and target domain, which can be costly.

A recent line of work investigates the setting of few-shot NER by using distance metrics (Wiseman and Stratos, 2019; Yang and Katiyar, 2020; Ziyadi et al., 2020). The main idea is to train a similarity function based on instances in the source domain, and then make use of the similarity function in the target domain as a nearest neighbor criterion for few-shot NER.

Compared with traditional methods, distance-based methods largely reduce the domain adapta-

---

[*] Corresponding Author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1835–1845
August 1–6, 2021.

tion cost, especially for scenarios where the number of target domains is large. Their performance under standard in-domain settings, however, is relatively weak. In addition, their domain adaptation power is also limited in two aspects. First, labeled instances in the target domain are used to find the best hyper-parameter settings for heuristic nearest neighbor search, but are not for updating the network parameters of the NER model. While being less costly, these methods cannot improve the neural representation for cross-domain instances. Second, these methods rely on similar textual patterns between the source domain and the target domain. This strong assumption may hinder the model performance when the target-domain writing style is different from the source domain.

To address these issues, we investigate a template-based method for exploiting the few-shot learning potential of generative pre-trained language models to sequence labeling. Specifically, as shown in Figure 2, BART (Lewis et al., 2020) is fine-tuned with pre-defined templates filled by corresponding labeled entities. For example, we can define templates such as "⟨`candidate_span`⟩ *is a* ⟨`entity_type`⟩ *entity*", where ⟨`entity_type`⟩ can be "*person*" and "*location*", etc. Given the sentence "*ACL will be held in Bangkok*", where "*Bangkok*" has a gold label "*location*", we can train BART using a filled template "*Bangkok is a location entity*" as the decoder output for the input sentence. In terms of non-entity spans, we use a template "⟨`candidate_span`⟩ *is not a named entity*", so that negative output sequences can also be sampled. During inference, we enumerate all possible text spans in the input sentence as named entity candidates, classifying them into entities or non-entities based on BART scores on templates.

The proposed method has three advantages. First, due to the good generalization ability of pre-trained models (Brown et al., 2020; Gao et al., 2020), the network can effectively leverage labeled instances in the new domain for tine-tuning. Second, compared with distance-based methods, our method is more robust even if the target domain and source domain have a large gap in writing style. Third, compared with traditional methods (pre-trained model with a softmax/CRF), our method can be applied to arbitrary new categories of named entities without changing the output layer, and therefore allows continual learning (Lin et al., 2020).

We conduct experiments in both resource-rich and few-shot settings. Results show that our methods give competitive results with state-of-the-art label-dependent approaches on the news dataset CoNLL03 (Tjong Kim Sang and De Meulder, 2003), and significantly outperforms Wiseman and Stratos (2019), Ziyadi et al. (2020) and Huang et al. (2020) when it comes to few-shot settings. To the best of our knowledge, we are the first to employ a generative pre-trained language model to address a few-shot sequence labeling problem. We release our code at `https://github.com/Nealcly/templateNER`.

2 Related Work

Neural methods have given competitive performance in NER. Some methods (Chiu and Nichols, 2016; Strubell et al., 2017) treat NER as a local classification problem at each input token, while other methods use CRF (Ma and Hovy, 2016) or a sequence-to-sequence framework (Zhang et al., 2018; Liu et al., 2019). Cui and Zhang (2019) and Gui et al. (2020) use a label attention network and Bayesian neural networks, respectively. Yamada et al. (2020) use entity-aware pre-training and obtain state-of-the-art results on NER. These approaches are similar to ours in the sense that parameters can be tuned in supervised learning, but unlike our method, they are designed for prescribed named entity types, which makes their domain adaptation costly with new few-shot entity types.

Our work is motivated by distance-based few-shot NER, which aims to minimize domain-adaptation cost. Wiseman and Stratos (2019) copy the token-level label from nearest neighbors by retrieving a list of labeled sentences. Yang and Katiyar (2020) improve Wiseman and Stratos (2019) by using a Viterbi decoder to capture label dependencies estimated from the source domain. Ziyadi et al. (2020) follow a two-step approach (Lin et al., 2019; Xu et al., 2017), which first detects spans boundary and then recognizes entity types by comparing the similarity with the labeled instance. While not updating the network parameters for NER, these methods rely on similar name entity patterns between the source domain and the target domain. One exception is Huang et al. (2020), who investigate noisy supervised pre-training and self-training method by using external noisy web NER data. Compared to their method, our method does not

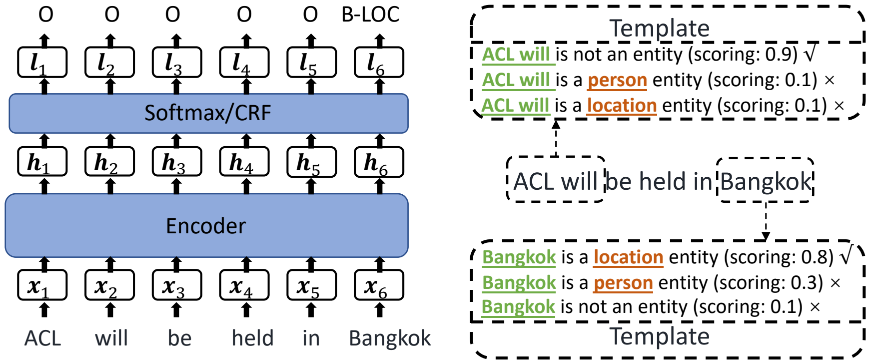

(a) Traditional sequence labeling method.

(b) Inference of template-based method.

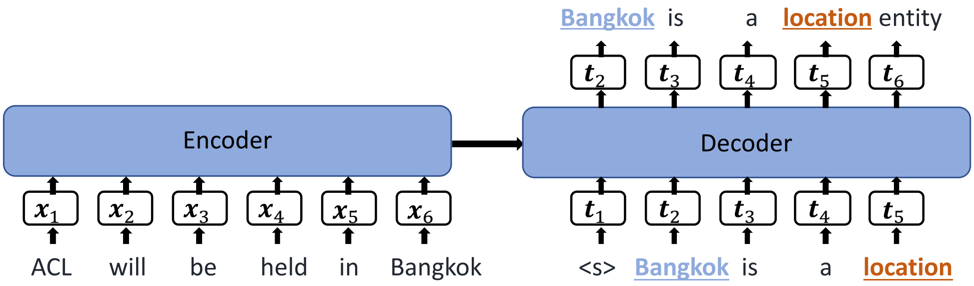

(c) Training of template-based method. The template we use here is "$\langle x_{i:j} \rangle$ is a $\langle y_k \rangle$ entity".

Figure 2: Overview of NER methods.

rely on self training on external data, yet yields better results.

There is a line of work using templates to solve natural language understanding tasks. The basic idea is to leverage information from pre-trained models, by defining specific sentence templates in a language modeling task. Brown et al. (2020) first use prompt for few-shot learning in text classification tasks. Schick and Schütze (2020) rephrase inputs as cloze questions for text classification. Schick et al. (2020) and Gao et al. (2020) extend Schick and Schütze (2020) by automatically generating label words and templates, respectively. Petroni et al. (2019) extract relation between entities from BERT by constructing cloze-style templates. Sun et al. (2019) use templates to construct auxiliary sentences, and transform aspect sentiment task as a sentence-pair classification task. Our work is in line with exploiting pre-trained language model for templates-based NLP. While previous work considers sentence-level task as masked language modeling or uses language models to score a whole sentence, our method uses a language model to assign a score for each span given an input sentence. To our knowledge, we are the first to apply template-based method to sequence labeling.

3 Background

We give the formal definition of few shot named entity recognition in Section 3.1 and traditional sequence labeling methods in Section 3.2.

3.1 Few shot Named Entity Recognition

Suppose that we have a rich-resource NER dataset $\mathbb{H} = \{(\mathbf{X}_1^H, \mathbf{L}_1^H), ..., (\mathbf{X}_I^H, \mathbf{L}_I^H)\}$, where $\mathbf{X}^H = \{x_1^H, \dots, x_n^H\}$ is a sentence and $\mathbf{L}^H = \{l_1^H, \dots, l_n^H\}$ is its corresponding label sequence. We use $\mathcal{V}^H$ to denote the label set of the rich-resource dataset ($\forall l_i^H, l_i^H \in \mathcal{V}^H$). In addition, we have a low-resource NER dataset, $\mathbb{L} = \{(\mathbf{X}_1^L, \mathbf{Y}_1^L), ..., (\mathbf{X}_J^L, \mathbf{Y}_J^L)\}$, and the number of its labelled sequence pairs is quite limited compared with the rich-resource NER dataset (i.e., $J \ll I$). Regarding the low-resource domain, the target label vocabulary $\mathcal{V}^L$ ($\forall l_i^L, l_i^L \in \mathcal{V}^L$) might be different from $\mathcal{V}^H$ (Figure 1). Our goal is to train an accurate and robust NER model with $\mathbb{L}$ and $\mathbb{H}$ for the low-resource domain.

3.2 Traditional Sequence Labeling Methods.

Traditional methods (Figure 2(a)) regard NER as a sequence labeling problem, where each output label consists of a sequence segmentation component B (beginning of an entity), I (internal word in an entity), O (not an entity), and an entity type tag such as "*person*" and "*location*". For example, the tag "*B-person*" indicates the first word in a person type entity and the tag "*I-location*" indicates a token of a location entity not at the beginning. Formally, given $x_{1:n}$, the sequence labeling method calculates

$$\begin{aligned} &\mathbf{h}_{1:n} = \textsc{Encoder}(x_{1:n}) \\ &p(\hat{l}_c) = \textsc{softmax}(\mathbf{h}_c \mathbf{W}_{\mathcal{V}^R} + \mathbf{b}_{\mathcal{V}^R})\ (c \in [1, ..., n]) \end{aligned} \tag{1}$$

where d_h is the hidden dimension of the encoder, $\mathbf{W}_{\mathcal{V}^R} \in \mathbb{R}^{d_h \times |\mathcal{V}^R|}$ and $\mathbf{b}_{\mathcal{V}^R} \in \mathbb{R}^{|\mathcal{V}^R|}$ are trainable parameters, and $\hat{l}_c$ is the label estimation for x_c. We use BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) as our ENCODER to learn the sequence representation.

A standard method for NER domain adaptation is to train a model using source-domain data $\mathbb{R}$ first, before further tuning the model using target domain instances $\mathbb{P}$, if available. However, since the label sets can be different, and consequently the output layer parameters ($\mathbf{W}_{\mathcal{V}^R} \in \mathbb{R}^{d_h \times |\mathcal{V}^R|}$, $\mathbf{b}_{\mathcal{V}^R} \in \mathbb{R}^{|\mathcal{V}^R|}$ and $\mathbf{W}_{\mathcal{V}^P} \in \mathbb{R}^{d_h \times |\mathcal{V}^P|}$, $\mathbf{b}_{\mathcal{V}^P} \in \mathbb{R}^{|\mathcal{V}^P|}$) can be different across domains. We train $\mathbf{W}_{\mathcal{V}^P}$ and $\mathbf{b}_{\mathcal{V}^P}$ from scratch using $\mathbb{P}$. However, this method does not fully exploit label associations (e.g., the association between "*person*" and "*character*"), nor can it be directly used for zero-shot cases, where no labeled data in the target domain is available.

4 Template-Based Method

We consider NER as a language model ranking problem under a seq2seq framework. The source sequence of the model is an input text $\mathbf{X} = \{x_1, \ldots, x_n\}$ and the target sequence $\mathbf{T}_{y_k, x_{i:j}} = \{t_1, \ldots, t_m\}$ is a template filled by candidate text span $x_{i:j}$ and the entity type y_k. We first introduce how to create templates in Section 4.1, and then show the inference and training details in Section 4.2 and Section 4.3, respectively.

4.1 Template Creation

We manually create the template, which has one slot for `candidate_span` and another slot for the `entity_type` label. We set a one to one mapping function to transfer the label set $\mathbf{L} = \{l_1, \ldots, l_{|L|}\}$ (e.g., l_k="LOC") to a natural word set $\mathbf{Y} = \{y_1, \ldots, y_{|L|}\}$ (e.g. y_k="*location*"), and use words to define templates $\mathbf{T}^+_{y_k}$ (e.g. ⟨`candidate_span`⟩ is a location entity.). In addition, we create a non-entity template $\mathbf{T}^-$ for none of the named entity (e.g., ⟨`candidate_span`⟩ is not a named entity.). This way, we can obtain a list of templates $\mathbf{T} = [\mathbf{T}^+_{y_1}, \ldots, \mathbf{T}^+_{y_{|L|}}, \mathbf{T}^-]$. In Figure 2(c), the template $\mathbf{T}_{y_k, x_{i:j}}$ is "$\langle x_{i:j} \rangle$ is a $\langle y_k \rangle$" and $\mathbf{T}^-_{x_{i:j}}$ is "$\langle x_{i:j} \rangle$ is not a named entity", where $x_{i:j}$ is a candidate text span.

4.2 Inference

We first enumerate all possible spans in the sentence $\{x_1, \ldots, x_n\}$ and fill them in the prepared templates. For efficiency, we restrict the number of n-grams for a span from one to eight, so $8n$ templates are created for each sentence. Then, we use the fine-tuned pre-trained generative language model to assign a score for each template $\mathbf{T}_{y_k, x_{i:j}} = \{t_1, \ldots, t_m\}$, formulated as

$$f(\mathbf{T}_{y_k, x_{i:j}}) = \sum_{c=1}^{m} \log p(t_c | t_{1:c-1}, \mathbf{X}) \tag{2}$$

We calculate a score $f(\mathbf{T}^+_{y_k, x_{i:j}})$ for each entity type and $f(\mathbf{T}^-_{x_{i:j}})$ for the none entity type by employing any pre-trained generative language model to score templates. Then we assign $x_{i:j}$ the entity type with the largest score to the text span. In this paper, we take BART as the pre-trained generative language models.

Our datasets do not contain nested entities. If two spans have text overlap and are assigned different labels in the inference, we choose the span with higher score as the final decision to avoid possible prediction contradictions. For instance, given the sentence "*ACL will be held in Bangkok*", the $n-$gram "*in Bangkok*" and "*Bangkok*" can be labeled "ORG" and "LOC", respectively, by using local scoring function $f(\cdot)$. In this case, we compare $f(\mathbf{T}^+_{\text{ORG,“in Bangkok”}})$ and $f(\mathbf{T}^+_{\text{LOC,“Bangkok”}})$, and choose the label which has a larger score to make the global decision.

4.3 Training

Gold entities are used to create template during training. Suppose that the entity type of $x_{i:j}$ is y_k. We fill the text span $x_{i:j}$ and the entity type y_k into $\mathbf{T}^+$ to create a target sentence $\mathbf{T}^+_{y_k, x_{i:j}}$. Similarly, if the entity type of $x_{i:j}$ is a none entity text span, the target sentence $\mathbf{T}^-_{x_{i:j}}$ is obtained by filling $x_{i:j}$ into $\mathbf{T}^-$. We use all gold entities in the training set to construct $(\mathbf{X}, \mathbf{T}^+)$ pairs, and additionally create negative samples $(\mathbf{X}, \mathbf{T}^-)$ by randomly sampling

Entity Template $\mathbf{T}^+$	None-Entity Template $\mathbf{T}^-$	Dev F1
⟨`candidate_span`⟩ is a ⟨`entity_type`⟩ entity	⟨`candidate_span`⟩ is not a named entity	95.27
The entity type of ⟨`candidate_span`⟩ is ⟨`entity_type`⟩	The entity type of ⟨`candidate_span`⟩ is none entity	95.15
⟨`candidate_span`⟩ belongs to ⟨`entity_type`⟩ category	⟨`candidate_span`⟩ belongs to none category	88.42
⟨`candidate_span`⟩ should be tagged as ⟨`entity_type`⟩	⟨`candidate_span`⟩ should tagged as none entity	76.80

Table 1: Resulting using different templates.

non-entity text spans. The number of negative pairs is 1.5 times that of positive pairs.

Given a sequence pair $(\mathbf{X}, \mathbf{T})$, we feed the input $\mathbf{X}$ to the encoder of the BART, and then we obtain hidden representations of the sentence

$$\mathbf{h}^{enc} = \textsc{Encoder}(x_{1:n}) \tag{3}$$

At the c th step of the decoder, $\mathbf{h}^{enc}$ and previous output tokens $t_{1:c-1}$ are then as inputs, yielding a representation using attention (Vaswani et al., 2017)

$$\mathbf{h}_c^{dec} = \textsc{Decoder}(\mathbf{h}^{enc}, t_{1:c-1}) \tag{4}$$

The conditional probability of the word t_c is defined as:

$$p(t_c|t_{1:c-1}, \mathbf{X}) = \textsc{softmax}(\mathbf{h}_c^{dec}\mathbf{W}_{lm} + \mathbf{b}_{lm}) \tag{5}$$

where $\mathbf{W}_{lm} \in \mathbb{R}^{d_h \times |\mathcal{V}|}$ and $\mathbf{b}_{lm} \in \mathbb{R}^{|\mathcal{V}|}$. $|\mathcal{V}|$ represents the vocab size of pre-trained BART. The cross-entropy between the decoder's output and the original template is used as the loss function.

$$\mathcal{L} = -\sum_{c=1}^{m} \log p(t_c|t_{1,c-1}, \mathbf{X}) \tag{6}$$

4.4 Transfer Learning

Given a new domain $\mathbb{P}$ with few-shot instances, the label set $\mathbf{L}^P$ (Section 4.1) can be different from what has been used for training the NER model. We thus fill the templates with the new domain label set for both training and testing, with the rest of the model and algorithms unchanged. In particular, given a small amount of $(\mathbf{X}^P, \mathbf{T}^P)$, we create sequence pairs with the method described above for the low-resource domain, and fine-tuning the NER model trained on the rich-source domain. This process has low cost, yet can effectively transfer label knowledge. Because the output of our method is a natural sentence instead of specific labels, both resource-rich and low-resource label vocabulary are subset of the pre-trained language model vocabulary ($\mathcal{V}^R, \mathcal{V}^P \subsetneqq \mathcal{V}$). This allows our method to make use of label correlations such as "*person*" and "*character*", and "*location*" and "*city*", for enhancing the effect of transfer learning across domains.

5 Experiments

We compare template-based BART with several baselines on both resource-rich settings and few-shot settings. We use the CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) as the resource-rich dataset. Following Ziyadi et al. (2020) and Huang et al. (2020), we use MIT Movie Review (Liu et al., 2013), MIT Restaurant Review (Liu et al., 2013) and ATIS (Hakkani-Tur et al., 2016) as the cross-domain few-shot dataset. Regarding the cross-domain transfer, there are unseen entity types in the three target few-shot datasets. Details of our training details and dataset statistics are shown in Appendix.

5.1 Template Influence

There can be different templates for expressing the same meaning. For instance "⟨`candidate_span`⟩ *is a person*" can also be expressed by "⟨`candidate_span`⟩ *belongs to the person category*". We investigate the impact of manual templates using the CoNLL03 development set. Table 1 shows the performance impact of different choice of templates. For instance, "⟨`candidate_span`⟩ *should be tagged as* ⟨`entity_type`⟩" and "⟨`candidate_span`⟩ *is a* ⟨`entity_type`⟩ *entity*" give 76.80% and 95.27% F1 score, respectively, indicating the template is a key factor that influences the final performance. Based on the development results, we use the top performing template "⟨`candidate_span`⟩ is a ⟨`entity_type`⟩ entity" in our experiments.

5.2 CoNLL03 Results

Standard NER setting. We first evaluate the performance under the standard NER setting on CoNLL03. The results are shown in Table 2, where state-of-the-art methods are also compared. In particular, the sequence labeling BERT gives a strong baseline, F1 score at 91.73%. We can see that even though the template-based BART is designed for few-shot named entity recognition, it performs competitively in resource-rich setting as well. For instance, our method outperforms sequence labeling BERT by 1.80% on recall, which shows that our method is more effective in identifying the named entity, but also selecting irrelevant span. Noted that though both sequence labeling BART and template-based BART make use of BART decoder representations, their performances have a large gap, where the latter outperforms the former by absolutely 1.30% on F1 score, demonstrating the effectiveness of the template-based method. The observation is consistent with that of Lewis et al. (2020), which shows that BART is not the most competitive for sequence classification. This may result from the nature of its seq2seq-based denoising autoencoder training, which is different from masked language modeling for BERT.

To explore if templates are complementary for each other, we train three models using the first three templates reported in Table 1, and adopt an entity-level voting method to ensemble these three models. There is a 1.21% precision increase using ensemble, which shows that different templates may capture different type of knowledge. Finally, our method achieves a 92.55 % F1 score by leveraging three templates, which is highly competitive with the best reported score. For computational efficiency, we use a single model for the subsequent few-shot experiments.

In domain few-shot NER setting. We construct a few-shot learning scenario on the CoNLL03, where the number of training instances for some specific categories is quite limited by down-sampling. In particular, we set "MISC" and "ORG" as the resource-rich entities, and "LOC" and "PER" as the low-resource entities. We down-sample the CoNLL03 training set, yielding 3,806 training instances, which includes 3,925 "ORG", 1,423 "MISRC", 50 "LOC" and 50 "PER". Since the text style is consistent in rich-resource and low-resource entity categories, we call the scenario in domain few-shot NER.

As shown in Table 3, sequence labeling BERT and template-based BART show similar performance in resource-rich entity types, while our method significantly outperforms BERT by 11.26 and 12.98 F1 score in "LOC" and "MISC", respectively. It demonstrates that our method has a stronger modeling capability for in-domain few-shot NER, and indicates that the proposed method can better transfer the knowledge between different entity categories.

Traditional Models	**P**	**R**	**F**
Yang et al. (2018)	-	-	90.77
Ma and Hovy (2016)	-	-	91.21
Gui et al. (2020)	-	-	92.02
Yamada et al. (2020)*	-	-	94.30
Sequence Labeling BERT	91.93	91.54	91.73
Sequence Labeling BART	89.60	91.63	90.60
Few-shot Friendly Models	**P**	**R**	**F**
Wiseman and Stratos (2019)	-	-	89.94
Template BART	90.51	93.34	91.90
multi-template BART	91.72	93.40	92.55

Table 2: Model performance on the CoNLL03 .The original result of BERT (Devlin et al., 2019) was not achieved with the current version of the library as discussed and reported by Stanislawek et al. (2019), Akbik et al. (2019) and Gui et al. (2020). * indicates training on external data.

Models	PER	ORG	LOC*	MISC*	Overall
BERT	75.71	77.59	60.72	60.39	69.62
Ours	84.49	72.61	71.98	73.37	75.59

Table 3: In-domain Few-shot performance on the CoNLL03. * indicates it is a few-shot entity type.

5.3 Cross-domain Few-Shot NER Result

We evaluate the model performance when the target entity types are different from the source-domain, and only a small amount of labeled data is available for training. We simulate the cross-domain low-resource data scenarios by random sampling training instances from a large training set as the training data in the target domain. We use different numbers of instances for training, randomly sampling a fixed number of instances per entity type (10, 20, 50, 100, 200, 500 instances per entity type for MIT Movie and MIT restaurant, and 10, 20, 50 instances per entity type for ATIS). If an entity has a smaller number of instances than the fixed number to sample, we use all of them for training. The results on few-shot experiments using MIT Movie, MIT Restaurant and ATIS are shown in Table 4, where the methods of Wiseman and Stratos (2019), Ziyadi et al. (2020) and Huang et al. (2020) are also compared.

We first consider a training-from-scratch setting, where no source-domain data is used. Distance-based methods cannot suit this setting. Compared with the traditional sequence labeling BERT method, our method can make better use of few-shot data. In particular, with as few as 20 instances per entity type, our method gives a F1 score of 57.1%, higher than BERT using 100 instances per entity type on MIT Restaurant.

We further investigate how much knowledge can

Source	Methods	10	20	50	100	200	500
				MIT Movie			
None	Sequence Labeling BERT	25.2	42.2	49.64	50.7	59.3	74.4
	Template-based BART	37.3	48.5	52.2	56.3	62.0	74.9
CoNLL03	Wiseman and Stratos (2019)	3.1	4.5	4.1	5.3	5.4	8.6
	Ziyadi et al. (2020)	40.1	39.5	40.2	40.0	40.0	39.5
	Huang et al. (2020)*	36.4	36.8	38.0	38.2	35.4	38.3
	Sequence Labeling BERT	28.3	45.2	50.0	52.4	60.7	76.8
	Sequence Labeling BART	13.6	30.4	47,8	49.1	55.8	66.9
	Template-based BART	42.4	54.2	59.6	65.3	69.6	80.3
				MIT Restaurant			
None	Sequence Labeling BERT	21.8	39.4	52.7	53.5	57.4	61.3
	Template-based BART	46.0	57.1	58.7	60.1	62.8	65.0
CoNLL03	Wiseman and Stratos (2019)	4.1	3.6	4.0	4.6	5.5	8.1
	Ziyadi et al. (2020)	27.6	29.5	31.2	33.7	34 .5	34.6
	Huang et al. (2020)	46.1	48.2	49.6	50.0	50.1	
	Sequence Labeling BERT	27.2	40.9	56.3	57.4	58.6	75.3
	Sequence Labeling BART	8.8	11.1	42.7	45.3	47.8	58.2
	Template-based BART	53.1	60.3	64.1	67.3	72.2	75.7
				ATIS			
None	Sequence Labeling BERT	44.1	76.7	90.7	-	-	-
	Template-based BART	71.7	79.4	92.6	-	-	-
CoNLL03	Wiseman and Stratos (2019)	6.7	8.8	11.1	-	-	-
	Ziyadi et al. (2020)	17.4	19.8	22.2	-	-	-
	Huang et al. (2020)	71.2	74.8	76.0	-	-	-
	Sequence Labeling BERT	53.9	78.5	92.2	-	-	-
	Sequence Labeling BART	51.3	74.4	89.9	-	-	-
	Template-based BART	77.3	88.9	93.5	-	-	-

Table 4: Cross-domain few-shot NER performance on different test sets. * indicates training on external data. 10 indicates 10 instances for each entity types.

be transferred from the news domain (CoNLL03). In this setting, we further train the model which is trained on the news domain. It can be seen from the Table 4 that on all the three datasets, the few-short learning methods outperform sequence labeling BERT and BART methods when the number of training instances is small. For example, when there are only 10 training instances, the method of Ziyadi et al. (2020) gives a F1 score of 40.1% on MIT Movie, as compared to 28.3% by BERT, despite that BERT requires re-training with a different output layer on both CoNLL03 and MIT Movie. However, as the number of training instances increase, the advantage of baseline few-shot methods decreases. When the number of instances grows as large as 500, BERT outperforms all existing methods. Our method is effective in both 10 instances and 500 instances, outperforming both BERT and baseline few-shot methods.

Compared with the distance-based method (Wiseman and Stratos, 2019; Ziyadi et al., 2020; Huang et al., 2020), our method shows more improvement when the number of target-domain labeled data increases, because the distance-based method just optimizes its searching threshold rather than updating its neural network parameters. We can see that the performance of distance-based methods remains the same as the labeled data increasing. For example, the performance of Huang et al. (2020) increases only 1.9% F1 score when the number of instances per entity type increase from 10 to 500. Both BERT and our method perform better than training from scratch. Our model average increases 6.6, 6.9 and 5.4 F1 score on MIT restaurant, MIT movie and ATIS, respectively, which is significantly higher than 3.1, 1.9 and 4.3 F1 score in BERT. This shows that our model is more successful in transferring knowledge learned from the source domain. One possible explanation is that our model makes more use of the correlations between different entity type labels in the vocabulary as mentioned earlier, which BERT cannot achieve due to treating the output as discrete class labels.

5.4 Discussion

Impact of entity frequencies in training data. To explore the relation between recognition accuracy and the frequency of an entity type in training, we split ATIS test set into three subset based on the entity frequency in training. The most 33% frequency entities are put into *high frequency* subset, the last 33% frequency entities are put into *low frequency* subset, and the remaining are put into *mid frequency* subset. Figure 3 shows the F1 score of

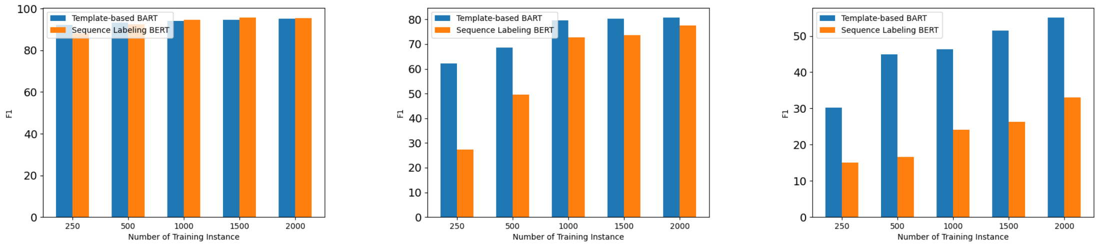

(a) High Frequency. (b) Mid Frequency. (c) Low Frequency.

Figure 3: Comparison of F1 with different frequency entity types on ATIS.

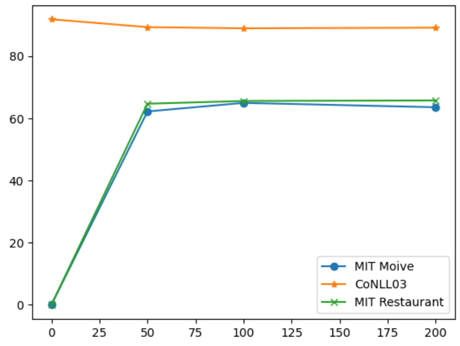

Figure 4: Continual learning experiments.

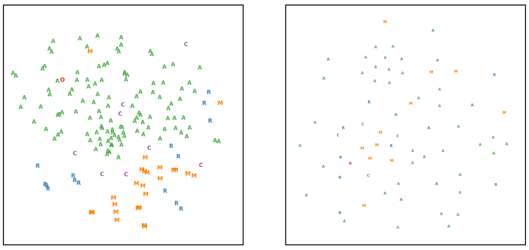

(a) Sequence labeling BERT. (b) Template-based BART.

Figure 5: Visualization of the output embedding. "C"–CoNLL03, "A"–ATIS, "M"–MIT Movie, "R"–MIT Restaurant.

BERT and our method against the number of training instance in the three subsets. As the number of training instances increases, the performance of all models increases. Our method outperforms sequence labeling BERT by a large margin, especially on the *mid frequency* and *low frequency* subsets, which demonstrates that our method is more robust in few-shot settings.

Continual Learning NER In continual learning setting (Lin et al., 2020), all the baselines that we have in Table 4 face limitations. The sequence labeling BERT method needs re-training using all training data each time a new entity type is encountered, which is highly expensive. The distance based methods cannot make use of all available data for improving representation learning. Figure 4 shows performance of our method on MIT movie, MIT restaurant and CoNLL03, when we continue to train our CoNLL03 model on both MIT movie and MIT restaurant. The performance on the CoNLL03 only slightly decrease when we continue training the model on the MIT Movie and MIT Restaurant dataset, demonstrating the robustness of our method in the continual learning setting.

Visualization. We explore why our model works well in the low-resource domain by visualizing the output layer. We train BERT and our method on all four datasets, and use t-SNE (van der Maaten and Hinton, 2008) to project the output layer into 2-dimensions, where the output layer for sequence labeling BERT and template-based BART are $\mathbf{W}_{\mathcal{Y}^R}$ in Eq 1 and $\mathbf{W}_{lm}$ in Eq 5, respectively. In Figure 5, each dot represents a row in the output matrix (corresponding to a label embedding). We can see that output layer embeddings of BERT are clustered based on dataset while the vectors of template-based BART are sparsely distributed in the space. It indicates that our output matrix is more domain independent, and our method enjoys better generalization ability across different domains.

Error Types We find that most mistakes are caused by the domain distance between high-resource data and low-source NER data. As shown in Fig 5, Template-based methods rely on label semantics. If the embedding of the word with a few-shot labels is far from that with in-domain labels, the model shows lower performance on that label type. Taking 50 examples per entity type on MIT movie as an example, "ACTOR" is similar to

"PERSON" in CoNLL03, and achieves 84.81 F1. The embedding of "SONG" is far from the existing labels in CoNLL03, and only achieves 34.97 F1. In contrast, sequence labeling BERT does not suffer from this distance, because BERT cannot draw label correlation between two domains, it achieves 53.98 and 40.13 on "ACTOR" and "SONG", respectively.

6 Conclusion

We investigated template-based few-shot NER using BART as the backbone model. In contrast to the traditional sequence labeling methods, our method is more powerful on few-shot NER, since it can be fine-tuned for the target domain directly when new entity categories exist. Experiment results show that our model achieves competitive results on a rich-resource NER benchmark, and outperforms traditional sequence labeling methods and distance-based methods significantly on the cross-domain and few-shot NER benchmarks.

Acknowledgments

We thank all anonymous reviewers for their constructive comments. This work is supported by National Science Foundation of China (Grant No. 61976180).

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Leyang Cui and Yue Zhang. 2019. Hierarchically-refined label attention network for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4115–4128, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners.

Tao Gui, Jiacheng Ye, Qi Zhang, Zhengyan Li, Zichu Fei, Yeyun Gong, and Xuanjing Huang. 2020. Uncertainty-aware label refinement for sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2316–2326, Online. Association for Computational Linguistics.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of Interspeech*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

Shaofu Lin, Jiangfan Gao, Shun Zhang, Xiaobo He, Ying Sheng, and Jianhui Chen. 2020. A continuous learning method for recognizing named entities by integrating domain contextual relevance measurement and web farming mode of web intelligence. In *World Wide Web*.

J. Liu, P. Pasupat, S. Cyphers, and J. Glass. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Ying Luo, Fengshun Xiao, and Hai Zhao. 2020. Hierarchical contextualized representation for named entity recognition. In *the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2020)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5569–5578, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference.

Tomasz Stanislawek, Anna Wróblewska, Alicja Wójcicka, Daniel Ziembicki, and Przemyslaw Biecek. 2019. Named entity recognition - is there a glass ceiling? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 624–633, Hong Kong, China. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark. Association for Computational Linguistics.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5363–5369, Florence, Italy. Association for Computational Linguistics.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247, Vancouver, Canada. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Yuan Zhang, Hongshen Chen, Yihong Zhao, Qun Liu, and Dawei Yin. 2018. Learning tag dependencies for sequence tagging. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4581–4587. International Joint Conferences on Artificial Intelligence Organization.

Morteza Ziyadi, Yuting Sun, Abhishek Goswami, Jade Huang, and Weizhu Chen. 2020. Example-based named entity recognition.

“Does it Matter When I Think You Are Lying?” Improving Deception Detection by Integrating Interlocutor’s Judgements in Conversations

Huang-Cheng Chou[1,3], **Woan-Shiuan Chien**[1,3], **Da-Cheng Juan**[2], **Chi-Chun Lee**[1,3]
[1]Department of Electrical Engineering, National Tsing Hua University, Taiwan
[2]Department of Computer Science, National Tsing Hua University, Taiwan
[3]MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan
`{hc.chou,wschien,dacheng}@gapp.nthu.edu.tw, cclee@ee.nthu.edu.tw`

Abstract

It is well known that human is not good at deception detection because of a natural inclination of truth-bias. However, during a conversation, when an interlocutor (interrogator) is being asked explicitly to assess whether his/her interacting partner (deceiver) is lying, this perceptual judgment depends highly on how the interrogator interprets the context of the conversation. While the deceptive behaviors can be difficult to model due to their heterogeneous manifestation, we hypothesize that this contextual information, i.e., whether the interlocutor trusts or distrusts what his/her partner is saying, provides an important condition in which the deceiver’s deceptive behaviors are more consistently distinct. In this work, we propose a Judgmental-Enhanced Automatic Deception Detection Network (JEADDN) that explicitly considers interrogator’s perceived truths-deceptions with three types of speech-language features (acoustic-prosodic, linguistic, and conversational temporal dynamics features) extracted during a conversation. We evaluate our framework on a large Mandarin Chinese Deception Dialog Database. The results show that the method significantly outperforms the current state-of-the-art approach without conditioning on the judgements of interrogators on this database. We further demonstrate that the behaviors of interrogators are important in detecting deception when the interrogators distrust the deceivers. Finally, with the late fusion of audio, text, and turn-taking dynamics (TTD) features, we obtain promising results of 87.27% and 94.18% accuracy under the conditions that the interrogators trust and distrust the deceivers in deception detection which improves 7.27% and 13.57% than the model without considering the judgements of interlocutor respectively.

1 Introduction

Deception behaviors frequently appear in human daily life, such as politics (Clementson, 2018), news (Conroy et al., 2015a; Vaccari and Chadwick, 2020), and business (Grazioli and Jarvenpaa, 2003; Triandis et al., 2001). Despite its frequent occurrences, researchers have repeatedly shown that humans are not good at detecting deceptions (it’s 54% accuracy on average for both police officers and college students (Vrij and Graham, 1997)), even for highly-skilled professionals, such as teachers, social workers, and police officers (Hartwig et al., 2004; Vrij et al., 2006). Due to the difficulty in identifying deceptions by human, researchers have also developed an automatic deception detection (ADD) systems applied in various fields, such as cybercrime (Mbaziira and Jones, 2016), fake news detection (Conroy et al., 2015b), employment interviews (Levitan et al., 2018b,a), and even court decision (Venkatesh et al., 2019; Pérez-Rosas, Verónica and Abouelenien, Mohamed and Mihalcea, Rada and Burzo, Mihai, 2015). Although many works have studied approaches of automatic deception detection, few works, if any, has investigated whether judgements of human can help provide a condition that enhance ADD recognition rates.

In recent years, ADD has gained popularity and attention; however, almost all studies (if not all) on ADD pay attention to western cultures (countries), and there are very few literates that focus on eastern cultures (countries). Deception behavior often varies with different cultures (Aune and Waters, 1994), and every culture has its way to deceive others. Additionally, Rubin (2014) suggested that researchers need to study and understand more deception behaviors in the Asian area. Besides, many researchers have utilized various behavioral cues to build an ADD system, like facial expressions (Thannoon et al., 2019), internal physi-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1846–1860
August 1–6, 2021. ©2021 Association for Computational Linguistics

ological measures (Ambach and Gamer, 2018) and even functional brain MRI (Kozel et al., 2009a,b). While these indicators can be useful in detecting deceptions, many of them require expensive and invasive instrumentation that is not practical for real-world applications. Instead, speech and language cues carry substantial deceptive cues that can be modeled in ADD tasks for potential large-scale deployment (Zhou et al., 2003; Chou et al., 2019). Hence, the proposed method modeled the speech and language cues of humans with real-world data in Mandarin Chinese.

Despite these important advances in understanding and automatically identifying deceptions, there has been little work investigating whether the performance of ADD models can be significantly improved if considering the behaviors and perceptions of interrogators. Several questions remain: is there a difference in linguistic and acoustic-prosodic characteristics of an utterance from both interlocutors given trusted/distrusted judgments of interrogators? How do the judgments of interrogators help the ADD model detect deceptions? To investigate these questions, we firstly follow the previous studies (Chou et al., 2019) to segment a dialog into Questioning-Answering (QA) pair turns and then extract acoustic-prosodic features, linguistic features (e.g., Part-Of-Speech taggers (POS), Named Entity Recognition (NER), and Linguistic Inquiry and Word Count (LIWC)), conversational temporal dynamics (CTD) features. Then, we trained machine learning and deep learning classifiers using a large set of lexical and speech features to automatically identify deceptions and evaluated the results in the Daily Deceptive Dialogues corpus of Mandarin (DDDM). Also, to investigate the differences between interlocutor's behaviors, we perform Welch's t-test (Delacre et al., 2017) on the characteristics of utterances from both interlocutors given three different scenarios: (A) human-distrusted deceptive and truthful statements, (B) human-trusted deceptive and truthful statements, and (C) successful/unsuccessful deceptive and truthful statements.

In our further analyses, we found that (i) the judgments of human are indeed helpful to significantly improve the performance of the proposed method on detecting deceptions, (ii) the behaviors of interrogators should be considered into the model when the interrogator distrusted the deceivers, and (iii) the additional evidence indicates that human is bad at detecting deceptions – there are very few significant indicators that overlap between trusted truths-deceptions and successful-unsuccessful deceptions. We believe that these overlap-indicators could be useful for training humans to detect deceptions more successfully. Finally, we summarize our 3 main contributions as below.

- **We are the first work to include the judgements of the interrogator as a condition to help improve the recognition rates of deception detection model.**
- **We demonstrate that the features of interrogators are more effective and useful to detect deceptions than the deceivers' ones under the condition that the interrogator disbelieves the deceiver.**
- **The proposed model has high potentials for practical deception detection applications and impact on the ADD area.**

2 Related Work

Automatic deception detection in a dialogue Previous studies have trained a deception detector with various features in a dialog. Levitan et al. (2018a) extracted acoustic features of utterances to build the detection framework using a global-level label as the ground truth in employment interviews. Chou et al. (2019) indicated that the interlocutor's vocal characteristics and conversational dynamics should be jointly modeled to better perform deception detection in dialogues. The grammatical and syntactical POS features has been widely used in the automatic deception detection (Pérez-Rosas, Verónica and Abouelenien, Mohamed and Mihalcea, Rada and Burzo, Mihai, 2015; Levitan et al., 2016; Abouelenien et al., 2017; Kao et al., 2020). In addition, Liu et al. (2012); Levitan et al. (2018b) modeled the behaviors of language use from the LIWC features. Gröndahl and Asokan (2019); Chou et al. (2021) used textual embeddings extracted from the pre-trained BERT model for recognizing deceptions during an interrogator-deceiver conversation. Thannoon et al. (2019) used facial expression features to catch micro-variations on the face during the deceiver is telling either the lies or truths in the setting of interview conversation. Wu et al. (2018) had fused multimodal data including acoustic features, LIWC-embeddings, and facial-expression information to train a classifier for detecting deception, and Pérez-Rosas, Verónica and Abouelenien, Mohamed and Mihalcea, Rada and Burzo, Mihai (2015) trained the deception detection model using multimodal data with promising accuracy (92.20% area under the precision-recall curve, AUC) during a conversation in the court. However, most of the above-mentioned studies only model the behaviors of deceivers.

The interrogator's behaviors for detecting deceptions In criminal psychology, Dando and Bull (2011); Sandham et al. (2020) found that policies can be trained to identify criminal liars with advanced interrogation strategies (e.g, tactical use procedure) because these interview techniques maximize deceivers' cognitive load (Dando et al., 2015). In addition, Chou and Lee (2020) tried to learn from the behaviors of both interlocutors for identifying perceived deceptions, but their learning targets are from the perception of the interrogators not from the deceivers. Therefore, to our best knowledge, we are the first work to take the interrogators' behaviors for detecting deceptions automatically.

The perceptions of interrogators for detecting deceptions Levitan et al. (2018b) had studied the perception (judgment) of deception by identifying characteristics of statements that are perceived as truths or lies by interrogators, but they did not use the perceptions for detecting deceptions. Kleinberg and Verschuere (2021) used the LIWC variables and POS frequencies as input features to train a random forest classifier respectively, and then asked subjects to mark the scores ranging from 0 (certainty truthful) to 100 (certainty deceptive) on the deceptive or truthful text data. Finally, they presented the output probabilities of two trained classifiers on each data for the subjects to change the probabilities of the data. Their results showed that the perceptions of human impair the automatic deception detection models. However, we are different from Kleinberg and Verschuere (2021). The main difference is the way how judgements is being utilized; in our work, this is used to provide a condition in improving the prediction results.

Table 1: Distribution of the annotated data in the DDDM Database.

Data Distribution		Deceiver		Total
		Truth	Deception	
Interrogator	Trusted	(2) 97	(1) 86	**183**
	Distrusted	(3) 47	(4) 53	**100**

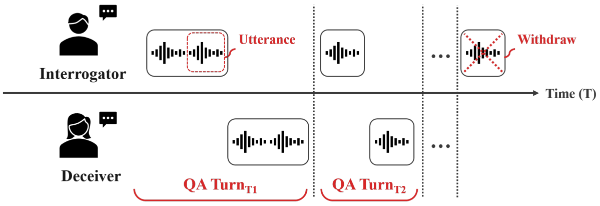

Figure 1: The illustration of Questioning-Answering (QA) pair turns. We only used *complete* QA pair turns and excluded some questioning turns if we cannot find the corresponding answering turns. To be noticed that each turn could have multiple utterances.

3 DDDM Database

We used conversational utterances from the Daily Deceptive Dialogues corpus of Mandarin (DDDM) (Huang et al., 2019). The entire DDDM contains about 27.2 hours of audio recordings from 96 unique speakers and 283 "question-level" conversational data samples. This corpus is particularly useful for our study, and all annotations in the DDDM come from "human" raters. Most deception databases lack recordings and perceptions (judgments) of the interrogators, while DDDM recorded the whole interrogator-deceiver conversations and the judgements of both interlocutors, allowing us to study deception detection given the judgements of the interrogators. With the judgements of both interlocutors, we group the data samples into four classes (shown in Table 1) as follows: (1) successful deceptions, (2) trusted truths, (3) distrusted truths, and (4) unsuccessful deceptions. We follow Chou et al. (2019) to transform the 7126 utterances into 2764 *complete* Questioning-Answering (QA) pair turns (shown in Figure 1) because the interrogator tended to ask follow-up questions for judging the deceiver's statements.

4 Problem Definition

4.1 The definition of deception

Deception is different from lying. Deception is human behavior that aims to make receivers believe true (or false) statements that the deceiver believes to be false (or true) with the conscious planning acts, such as sharing a mix of truthful and deceptive experiences to change the perceptions of the interrogators when being inquired to answer to questions. However, lying is just saying that something is true (or false) when in fact that something is false (or true) (Mitchell, 1986; Sarkadi, 2018). Hence, it is challenging for the interrogators to detect deceptions through the behaviors of deceivers. Human needs to engage in higher-order cognitive processing to detect these consciously planned deceptions (Street et al., 2019). The deceiver can act in a way to change the perceptions of that potential deception detector. This then shifts a heavier burden onto the interrogator's cognitive processing. Hence, the interrogator must necessarily engage in "higher-order" cognitive processing to detect these advanced lies because they usually cannot just detect the behavior (e.g., signs of nervousness invoice), but must interpret why this individual may be nervous, including the honest reason why (e.g., afraid of being disbelieved).

4.2 Deception detection with judgments of human

Humans rarely perform better than chance on detecting deceptions, but the interrogators make their judgements according to context information in an interrogator-deceiver conversation. People might be hard to remember the whole detailed information, but their judgements might consist of some context-general information based on their own experience, which results in a truth-bias. Therefore, we build the deception detection models based on the conditional perceptions of humans (human-trusted or human-distrusted). We use judgements of human as criteria to define the following conditions (we also include the condition that we have no judgements of human, and the most conventional studies on ADD are in this condition):

(i) Truthful and deceptive statements detection: detecting deceptions without perceptions of interrogators (judgements of human)

(ii) Trusted truthful and deceptive statements detection: detecting deceptions with believed judgments of interrogators

(iii) Distrusted truthful and deceptive statements detection: detecting deceptions with disbelieved judgments of interrogators

5 Methodology

5.1 JEADDN: Judgmental-Enhanced Automatic Deception Detection Network

Figure 2 illustrates the Judgmental-Enhanced Automatic Deception Detection Network (JEADDN) whose main structure is BLSTM-DNN (Chou et al., 2019) containing one bidirectional long short-term memory (BLSTM) layer with an attention mechanism and two fully-connected layers. In our method, judgements of human are criterion in choosing the classifiers for certain conditions to detect deceptions (not as the features). That is, when the interrogator believes the deceiver's statements, we use the condition (ii) classifier. Instead, when the interrogator disbelieves the deceiver, we can use the condition (iii) classifier. We fuse the best feature set from each modality by late fusion with additional three dense layers. Besides, there are two main goals. One is to investigate the effectiveness and robustness of speech and language features of both interlocutors. The other is to show whether the model performance of detecting deceptions with the judgements of interrogators could be better than the model without them.

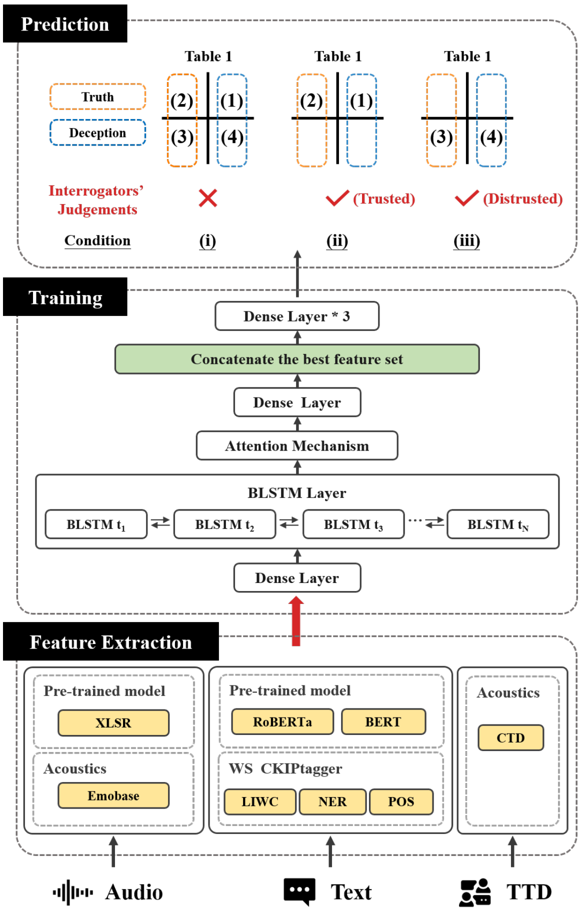

Figure 2: The overview of Judgmental-Enhanced Automatic Deception Detection Network (JEADDN) (The WS and TTD mean a word segmentation and turn-taking dynamics respectively).

More specifically, we split four-class sample data in Table 1 into two conditions based on judgements of interrogators (human-trusted/human-distrusted). The unit of features of interrogators/deceivers incorporates all of the utterances from the *complete*

Table 2: The table summarizes 8 turn-level feature sets introduction used in this paper.

Modality	Denotation	Feature Set	Dimension	Extraction Tool
Audio	Emobase	Emobase	988	openSmile (Eyben et al., 2010)
	XLSR	XLSR-53	7680	XLSR-53 (Conneau et al., 2020)
Text	NER	Named Entities Recognition	17	CKIP Tagger (Li et al., 2020)
	POS	Part-Of-Speech Tagger	50	CKIP Tagger (Li et al., 2020)
	BERT	BERT-Base in the Chinese version	768	BERT (Devlin et al., 2019)
	RoBERTa	RoBERTa-Base in the Chinese version	768	RoBERTa (Cui et al., 2020)
	LIWC	Linguistic Inquiry and Word Count 2015	82	LIWC 2015 (Pennebaker et al., 2015)
TTD	CTD	Conversational Temporal Dynamics	20	Proposed by Chou et al. (2019)

QA pair because interrogators would like to ask questions to seek detailed information. The closest previous study is Chou and Lee (2020). They have investigated **perceived** deception in the condition that the deceiver is telling either truths or deceptions, but they only focus on **perceived** deception recognition. Our objective is to detect the deceiver's answers corresponding to each question. In contrast, the learning targets of Chou and Lee (2020) are from the interrogator's guessed answers. Therefore, our learning targets are different from them. Moreover, their work is not useful in real life since they have to know the judgements of the deceivers, and it is impractical and impossible to be applied in the real world. In this paper, we hypothesize that (i) we can get better performance if the model takes judgements of interrogators into account, and (ii) there are differences in both interlocutors' behaviors between the trusted/distrusted truthful and deceptive dialogues. In the rest of the sections, we will describe the feature extraction in detail (notice that all types of the following feature sets are normalized to each speaker using z-score normalization) and the use of a deception detection framework.

5.1.1 Turn-level Feature Extraction

Table 2 summarized 8 various feature sets, which were extracted from the acoustic and linguistic characteristics of all speakers based on questioning turns of interrogators and answering turns of deceivers within QA pairs. In this work, we use the features extracted from audio and text recordings data to build the models, and we describe each feature set one by one as below.

Audio Recordings

- **Emobase:** we followed (Chou et al., 2019; Chou and Lee, 2020) to extracted 988-dimensional acoustic-prosodic features from questioning turns (answering turns) by "emobase.config" of openSMILE toolkit (Eyben et al., 2010).
- **CTD:** Chou et al. (2019) firstly proposed the conversational temporal dynamics (CTD) feature set within *complete* QA pairs. Additionally, Chou et al. (2021) incorporated CTD, Emobase, and BERT to achieve the state-of-the-art result of DDDM, so we also extract CTD for comparison. Also, CTD can extract the temporal turn-taking dynamics (TTD) of both interlocutors during a conversation. We list a part of features of CTD as below, and more detailed information about CTD is in Chou et al. (2019).
  - **Utterance-duration ratio:** the reciprocal ratio between the utterances length (u) and the turn duration (d), denoted as Int_{ud} and Int_{du} respectively.
  - **Silence-duration ratio:** the reciprocal ratio between the silence (s) duration and the turn duration, denoted as Int_{sd} and Int_{ds} respectively.
  - **Silence-utterance ratio:** the reciprocal ratio between the silence duration and the utterance lengths, denoted by Int_{su} and Int_{us} respectively.
  - **Silence times (st):** the number of times that a subject produces a pause that is more than 200ms, denoted as Int_{st} and Dec_{st}.
- **XLSR:** Due to the scarcity of deception databases in Mandarin Chinese, we use the multilingual pre-trained model, XLSR-53 (Conneau et al., 2020), to extract acoustic representation. XLSR-53 is trained for acoustic speech recognition (ASR) task with more than 56,000 hours of speech data in 53 different languages including Chinese-Taiwan (Mandarin Chinese) based on wav2vec 2.0 (Baevski et al., 2020). The dimension of the feature vector is 512 per frame, and then the feature vector per frame is applied

to the 15 statistics[1] to generate the final 7680-dimensional feature vectors.

Text Recordings

- **BERT:** we utilize BERT-Base in the Traditional Chinese version pre-trained model (Devlin et al., 2019) to extract turn-level 768-dimensional feature vectors. BERT was trained with a large amount of plain text data publicly available on the web using unsupervised objective functions (like masked-language modeling objective (MLM)) and works at the character level. We do not have to perform word segmentation when extracting representations.
- **RoBERTa:** we also use RoBERTa (Cui et al., 2020) to extract textual features. Its main component is the same as BERT (Devlin et al., 2019), but RoBERTa used a Whole-Word-Masking (WWM) technique and was trained on 10 times more data than BERT model. Although BERT has another version (BERT-WWM), there is no available pre-trained model in the Chinese language, so we only extract the features by both BERT and RoBERTa pre-trained models in this work.
- **POS:** we extracted 50-dimensional POS taggings excluding all punctuation-related dimensions by CKIP Tagger (Li et al., 2020) and then convert all POS predictions into feature vectors by calculating the number of word counts.
- **NER:** we use CKIP Tagger to extract 17-dimensional named entity recognition (NER) features, and convert the predictions into feature vectors by calculating the number of word counts. To our best knowledge, the NER feature set has never been used to train the deception detector. We are inspired by the findings of psychologist's studies on crime interrogation to use the NER feature set as input features for detecting deceptions. Vrij et al. (2021) suggest that the interrogators need to manipulate and design questions to ask the deceivers for detailed information, *complications*, because truth-tellers often reported more complications than lie tellers in each stage of the interview. A complication refers to details associated with personal experience or knowledge learned from any personal experience. In the DDDM, most recruited subjects are university students, and the three designed questions the researchers assigned each subject to ask are mainly about general activities or experiences of an average college student. For instance, scores of department border cups, professional knowledge about instruments, and detailed process of any events held by different clubs are regarded as personal experiences. Therefore, we extracted the NER features to capture the detailed information.
- **LIWC:** we use LIWC 2015 toolkit to extract 82-dimensional features (excluding all punctuation-related feature dimensions and total word counts (WP)) in this work after performing word segmentation pre-processing by CKIP Tagger.

6 Experiment

6.1 Experimental Setup

We conduct our experiments to show whether judgements and speech and language cues of interrogators are helpful to detect deceptions. The closest deception database is the Columbia X-Cultural Deception (CXD) Corpus (Levitan et al., 2015), but we have no access to the CXD corpus. To compare and show the baseline results, we compare all the models that had been used in the CXD corpus to reveal overall performance on the DDDM corpus. These baseline models include Support Vector Machines (SVM), Random Forest (RF), Logistic Regression (LG), and feedforward neural network (DNN). All baseline classifiers settings in this work are the same as Levitan et al. (2018b); Mendels et al. (2017).

Moreover, to compare with the state-of-the-art performance in the DDDM (Chou et al., 2021), we also use the same model proposed by (Chou et al., 2019), BLSTM-DNN, consisting of one fully-connected layer in a network with Rectified Linear Unit (ReLU) activation function, one BLSTM layer with an attention mechanism, one fully-connected layer with ReLU activation function, and then one prediction layer with a softmax activation function. We also include LSTM-DNN model in (Chou et al., 2021) as baseline classifier. All settings of LSTM-DNN and BLSTM-DNN are the same as (Chou et al., 2021). The whole framework is implemented by Pytorch (Paszke et al., 2019). The evaluation metric is macro F1-score based on the dyad-independent 10-fold cross-validation. We use the zero-padding to ensure each data sample's timestamp is the same if the length is less than the maximum timestamp (40). Several hyper-parameters for

---

[1] (1): amean, (2): 1th percentile, (3): 99th percentile, (4): kurtosis, (5): 99th percentile minus first percentile, (6): max, (7): maxPos, (8): min, (9): minPos, (10): quartile1, (11): quartile3, (12): range, (13): skewness, (14): stddev, (15) median.

Table 3: Results on the produced deception detection on the DDDM database in macro F1-score (%). The **Who's Feature** column implies that the feature comes from whom, such as the interrogator (Int.), the deceiver (Dec.), or both of interlocutors (directly concatenate the features of interlocutors in feature-level).

Modality	**Feature**	**Who's Feature**	**(i)**						**(ii)**	**(iii)**
			RF (2018b)	LR (2018b)	SVM (2018b)	DNN (2018b)	LSTM-DNN (2019)	BLSTM-DNN (2019)	BLSTM-DNN	
Audio	Emobase	Int.	51.35	47.03	52.29	63.31	59.98	59.80	70.06	**83.72**
		Dec.	54.42	54.52	51.03	66.56	63.95	**66.84**	**76.92**	80.49
		Both	51.92	50.54	51.09	65.45	57.70	60.85	72.00	81.17
	XLSR	Int.	48.83	43.47	50.20	**65.02**	60.52	59.74	74.32	**83.10**
		Dec.	49.83	45.78	53.15	64.06	60.54	61.37	**75.25**	80.36
		Both	48.26	44.52	52.49	64.38	59.64	58.79	74.82	79.75
Text	NER	Int.	44.03	55.15	48.68	64.40	55.29	57.94	59.66	64.15
		Dec.	57.03	48.21	55.74	65.04	**68.10**	66.19	**74.78**	72.61
		Both	55.01	56.00	52.10	66.67	65.18	65.37	74.77	**75.61**
	POS	Int.	51.18	50.23	57.32	66.34	60.96	60.89	71.06	72.41
		Dec.	51.23	55.90	57.08	66.83	64.74	61.29	**75.19**	77.14
		Both	50.05	55.00	56.13	**67.09**	64.72	62.99	74.77	**77.32**
	LIWC	Int.	51.25	50.87	55.14	65.00	64.29	65.10	**76.27**	80.51
		Dec.	52.75	54.81	57.26	**68.36**	64.18	64.19	74.32	**82.44**
		Both	50.63	49.37	57.68	67.36	63.79	62.54	75.40	77.10
	BERT	Int.	54.61	58.61	54.47	68.30	65.98	63.15	77.38	**85.53**
		Dec.	60.77	62.38	57.76	71.03	70.83	**72.00**	**82.14**	82.77
		Both	61.99	62.62	57.69	71.05	70.82	71.63	77.52	83.30
	RoBERTa	Int.	52.43	53.00	55.93	69.04	66.96	65.78	73.46	80.80
		Dec.	56.22	59.22	57.79	70.45	73.13	**74.31**	**79.88**	**86.59**
		Both	58.21	61.33	61.10	71.35	73.17	73.56	75.43	81.83
TTD	CTD	Both	50.65	47.04	47.83	**65.11**	59.86	56.19	**71.90**	**64.00**

the LSTM-DNN and BLSTM-DNN models as below are grid-searched: the number of nodes in the LSTM and BLSTM layers is ranging in [2, 4, 8], and the batch size is ranging in [16, 32], the learning rates is ranging in [0.01, 0.005] with adjusting mechanism by multiplying $\frac{1}{\sqrt{1+epoch}}$ per epoch. Finally, the maximum epoch is 10000. These hyperparameters are chosen with early stopping criteria in all conditions to minimize cross-entropy with balanced class weights on the validation set.

6.2 Experimental Results

Table 3 presents a summary of the complete results in three different conditions. There are 283, 183, and 100 question-level data samples under conditions (i), (ii), and (iii) respectively. The more detailed information about the portion of DDDM is shown in Table 1. Besides, the human performance is 54.7% macro F1-score in the DDDM corpus. The performance of DNN (Mendels et al., 2017) is very competitive, but modeling time-series information is important for conversation setting. Hence, we only present the results with the BLSTM-DNN model in the conditions (ii) and (iii).

In Table 3, the performances of the BLSTM-DNN with judgments of interrogators are consistently higher than the models without the judgments of interrogators, and the findings show corroborating evidence of the ALIED theory (Street, 2015; Street et al., 2019) which claimed that the perceptions of human could be potential lie detector even though the judgments of human are error-prone. We also found that the interrogators' features seem more contributing to deception detection in condition (iii). This finding demonstrates that we could consider the interrogators' features when the interrogators distrust the deceivers for building deception detection models. However, the performances of most models trained with the feature sets of the deceivers in the condition (i) and (ii) consistently surpass the ones trained with the features from the interrogators or both interlocutors.

6.3 Ablation Study

To investigate the effectiveness of audio, text, and turn-taking dynamics (TTD) modalities, we take the feature set according to the best performance in Table 3. We take Emobase, BERT, and CTD to represent the audio, text, and TTD modalities respectively. In the condition (i) and (ii), Emobase and BERT are from the deceivers. On the other

Table 4: Ablation results on three modalities, Emobase, BERT, and CTD feature sets.

Modality	Audio	Text	TTD	Condition		
Feature Set	Emobase	BERT	CTD	(i)	(ii)	(iii)
Single Modality	V			66.84	76.92	83.72
		V		72.00	82.14	85.53
			V	56.19	71.90	64.00
Late Fusion	V	V		78.68	86.79	91.32
	V		V	74.92	84.99	87.63
		V	V	77.83	85.90	90.16
	V	V	V	80.61	87.27	94.18

hand, the counterparts are from the interrogators in the condition (iii). In the fusion method, we follow Chou et al. (2021) to firstly freeze the weights of all models trained with the above-mentioned feature sets and concatenate their final dense layers' outputs as the input of the additional three-layer feed-forward neural network to perform late fusion. Table 4 summarizes the results of the ablation study, and the text modality is the most effective modality. Finally, we get the promising results 87.27 % and 94.18 % and significant improvements 7.27% and 13.57% than the model without judgements of human in the condition (ii) and (iii) respectively.

7 Analyses

Having established the presence and characteristics of each speech and language cue, we were interested in exploring the differences in both of interlocutors' speech and language cues on the different judgements of the interrogators given three different scenarios: (A) human-distrusted deceptive and truthful statements, (B) human-trusted deceptive and truthful statements, and (C) successful/unsuccessful deceptive and truthful statements. We firstly performed Welch's t-test (Delacre et al., 2017) for each speaker's turn (e.g., questioning/answering turns) within QA pairs that represented a question and answer from the 3 daily questions. The QA pairs shown in Figure 1 were marked manually, and each deceivers' answer was labeled as truth or deception using the daily life questionnaire response sheet. This resulted in 2764 QA pairs. Using this data, the significant indicators after performing Welch's t-test between each feature set on the different conditions are shown in Appendix A.1 Table A.1. Then, we calculate the ratio of significant features in each feature set divided by its dimension base because every feature set has different dimensions, i.e., in the NER feature set under the scenario (A), there are 7 significant indicators and its dimension base is 17, so the ratio is calculated by 7 divided by 17. Additionally, while XLSR, NER, POS, BERT, and RoBERTa are all extracted by not zero-error-rate pre-trained models and LIWC is also calculated the word counts afterword segmentation by CKIP Tagger, they all have significant indicators whose p-value is smaller than 0.05 among them. For example, BERT and RoBERTa from the deceivers have a high proportion of significant indicators. However, since the meaning of XLSR, BERT, and RoBERTa representations are difficult to explain intuitively, so we focus on other feature sets to examine the following research questions.

Is there a difference in both interlocutors' behaviors between distrusted truths and deceptions (Scenario A)? According to the experimental results in Table 3, we understand that the features of interrogators are significant indicators to detect deceptions. After performing the Welch's t-test on each feature set between **distrusted** truthful and deceptive interlocutor's questioning/answering responses (there are 898 QA pairs in scenario A), we found that the feature sets of NER, POS, and LIWC have a higher ratio of statistically significant indicators. Moreover, we check the predictions of them in the DDDM, and we observe that the interrogators tend to ask more complex questions to inquire detailed information about the statements of deceivers. That is, the interrogators would check the numbers information about scores of games, frequency of presentation, or length of music concerts (*PERCENT*, *QUANTITY*, *Neqb*, and *DM*), things about musical instrument or events about concert presentations and ball games (*EVENT*, *PRODUCT*, and *WORK_OF_ART*), and places/locations (i.e., elementary schools and universities) (*Nc*). This result is very interesting because the psychologist studies had also shown that how interrogators interrogate the deceivers in details would affect the success in catching liars. Besides, there are some significant indicators in LIWC, such as the words describing the movements in the sport game (*death*: “殺”球 (殺球 means kill and spike)) and the words to ask the deceivers to provide more detailed information (*focusfuture*: “然後”你之後還有繼續打球/彈樂器嗎? (“then”, did you keep playing balls/musical instrument afterward?)).

Is there a difference in both interlocutors' behaviors between trusted truths and deceptions (Scenario B)? In scenario B, the results of Welch's t-test reveal that NER consists of the highest ratio of significant indicators than others. When we go back to read the data in the DDDM (Appendix A.1) Table A.1, we observe that the truthful statements have more detailed descriptions than the deceptive ones, such times/dates of ball games and concerts (*DATE* and *Nd*), numbers to describe the scores of games (*CARDINAL* and *PERCENT*), and names about musical instrument and sport equipment (*PRODUCT*). Besides, the significant indicators of Emobase shown in Appendix A.1 Table A.2 includes the first derivative of the intensity of the deceivers. This result is similar to the previous study on the English database (Chen et al., 2020). That is, the interrogator tended to judge high-intensity utterances as truths because the louder utterances might be perceived as more confident even though these utterances could be deceptive in fact. Additionally, the significance test shows that some CTD features of interrogators are important indicators indicating whether the deceiver is telling the truth or not when the interrogator trusted the deceivers. For example, in the Appendix A.2 Table A.3, we can find that the interrogator spends more time to come up with more complex questions to inquire the deceiver; however, the interrogator eventually believes the deceiver's statements, but the proposed method can successfully detect the deceptions by the interrogator's temporal TTD behaviors. This finding is the same as the previous study (Chou et al., 2019).

Is there any common significant indicator between the one from distrusted truths and deceptions and the other from successful/unsuccessful deceptions (Scenario C)? In this analysis, we demonstrate additional evidence indicating that human is poor at detecting deceptions–there are very few indicators that overlap in all feature sets in this condition in Appendix A.1 Table A.1 (the rightmost column). However, the results repeatedly show that the ways how the interrogators ask questions about detailed information (*MONEY*, *PRODUCT*, and *DM*), and the meaningful information in the deceivers' answering statements (*A* (one of POS features) means the words to describe the noun, such as female, big, small, to name a few). Hence, the more detailed information we have, the higher chances to detect deceptions.

8 Conclusion and Future Work

This paper investigates whether judgements and speech and language cues of interrogators in conversation are useful and helpful to detect deceptions. We analyzed a full suite of acoustic-prosodic features, linguistic cues, conversational temporal dynamics given different conditions. Finally, with the late fusion of audio, text, and turn-taking dynamics (TTD) modality features, JEADDN obtains promising results of 87.27% and 94.18% accuracy under the conditions that the interrogators trust and distrust the deceivers in deception detection which improves 7.27% and 13.57% than the model without considering the interlocutor's judgements respectively.

While there is some research in studying perceived deception detection, this is one of the first studies that have explicitly modeled acoustic-prosodic characteristics, linguistic cues, and conversational temporal dynamics using judgments of interrogators in conversations for detecting deceptions. Furthermore, we provide analyses on the significance of different feature sets in three different scenarios and show additional evidence indicates that human is bad at detecting deceptions. Especially, the content of questions the interrogators ask is an indicator for telling deceptions or truths when the interrogators distrust the deceivers. Verigin et al. (2019) also reveal that truthful and deceptive information interacts to influence detail richness provides insight into liars' strategic manipulation of information when statements contain a mixture of truths and lies.

In the immediate future work, we aim to extend our multimodal fusion framework to combine semantic information to enhance the model robustness and the predicting powers within multiple QA pairs. That is, we observe that some interrogators finally trusted the deceivers after many follow-up questions while the statements of the deceivers were deceptive. Kontogianni et al. (2020) also pointed out that follow-up open-ended questions prompt additional reporting. However, practitioners should be cautious to corroborate the accuracy of new reported details.

Acknowledgments

The work is supported in part by funding from the Ministry of Science and Technology, Taiwan. We thank the anonymous reviewers, Jason S. Chang, and Jhih-Jie Chen for their valuable comments.

References

Mohamed Abouelenien, Verónica Pérez-Rosas, Bohan Zhao, Rada Mihalcea, and Mihai Burzo. 2017. Gender-Based Multimodal Deception Detection. In *Proceedings of the Symposium on Applied Computing*, SAC '17, page 137–144, New York, NY, USA. Association for Computing Machinery.

Wolfgang Ambach and Matthias Gamer. 2018. Chapter 1 - physiological measures in the detection of deception and concealed information. In J. Peter Rosenfeld, editor, *Detecting Concealed Information and Deception*, pages 3 – 33. Academic Press.

R.Kelly Aune and Linda L. Waters. 1994. Cultural differences in deception: Motivations to deceive in Samoans and North Americans. *International Journal of Intercultural Relations*, 18(2):159 – 172.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12435–12446. Curran Associates, Inc.

Xi (Leslie) Chen, Sarah Ita Levitan, Michelle Levine, Marko Mandic, and Julia Hirschberg. 2020. Acoustic-Prosodic and Lexical Cues to Deception and Trust: Deciphering How People Detect Lies. *Transactions of the Association for Computational Linguistics*, 8:199–214.

Huang-Cheng Chou and Chi-Chun Lee. 2020. "Your Behavior Makes Me Think It Is a Lie": Recognizing Perceived Deception using Multimodal Data in Dialog Games. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 393–402.

Huang-Cheng Chou, Yi-Wen Liu, and Chi-Chun Lee. 2019. JOINT LEARNING OF CONVERSATIONAL TEMPORAL DYNAMICS AND ACOUSTIC FEATURES FOR SPEECH DECEPTION DETECTION IN DIALOG GAMES. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1044–1050.

Huang-Cheng Chou, Yi-Wen Liu, and Chi-Chun Lee. 2021. Automatic deception detection using multiple speech and language communicative descriptors in dialogs. *APSIPA Transactions on Signal and Information Processing*, 10:e5.

David E. Clementson. 2018. Truth Bias and Partisan Bias in Political Deception Detection. *Journal of Language and Social Psychology*, 37(4):407–430.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised Cross-lingual Representation Learning for Speech Recognition.

Nadia K. Conroy, Victoria L. Rubin, and Yimin Chen. 2015a. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Nadia K. Conroy, Victoria L. Rubin, and Yimin Chen. 2015b. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online. Association for Computational Linguistics.

Coral J. Dando and Ray Bull. 2011. Research article. *Journal of Investigative Psychology and Offender Profiling*, 8(2):189 – 202.

Coral J. Dando, Ray Bull, Thomas C. Ormerod, and Alexandra L. Sandham. 2015. Helping to sort the liars from the truth-tellers: The gradual revelation of information during investigative interviews. *Legal and Criminological Psychology*, 20(1):114–128.

M. Delacre, D. Lakens, and C. Leys. 2017. Why psychologists should by default use welch's t-test instead of student's t-test. *International Review of Social Psychology*, 30(1):92–101.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: The Munich Versatile and Fast Open-Source Audio Feature Extractor. MM '10, New York, NY, USA. Association for Computing Machinery.

Stefano Grazioli and Sirkka L. Jarvenpaa. 2003. Consumer and Business Deception on the Internet: Content Analysis of Documentary Evidence. *Int. J. Electron. Commerce*, 7(4):93–118.

Tommi Gröndahl and N. Asokan. 2019. Text Analysis in Adversarial Settings: Does Deception Leave a Stylistic Trace? 52(3).

Maria Hartwig, Pär Anders Granhag, Leif A. Strömwall, and Aldert Vrij. 2004. Police Officers' Lie Detection Accuracy: Interrogating Freely Versus Observing Video. *Police Quarterly*, 7(4):429–456.

Chih-Hsiang Huang, Huang-Cheng Chou, Yi-Tong Wu, Chi-Chun Lee, and Yi-Wen Liu. 2019. Acoustic Indicators of Deception in Mandarin Daily Conversations Recorded from an Interactive Game. In *Proc. Interspeech 2019*, pages 1731–1735.

Yi-Ying Kao, Po-Han Chen, Chun-Chiao Tzeng, Zi-Yuan Chen, Boaz Shmueli, and Lun-Wei Ku. 2020. Detecting Deceptive Language in Crime Interrogation. In *HCI in Business, Government and Organizations*, pages 80–90, Cham. Springer International Publishing.

Bennett Kleinberg and Bruno Verschuere. 2021. How humans impair automated deception detection performance. *Acta Psychologica*, 213:103250.

Feni Kontogianni, Lorraine Hope, Paul J. Taylor, Aldert Vrij, and Fiona Gabbert. 2020. "tell me more about this…": An examination of the efficacy of follow-up open questions following an initial account. *Applied Cognitive Psychology*, 34(5):972–983.

F Andrew Kozel, Kevin A Johnson, Emily L Grenesko, Steven J Laken, Samet Kose, Xinghua Lu, Dean Pollina, Andrew Ryan, and Mark S George. 2009a. Functional MRI detection of deception after committing a mock sabotage crime. *Journal of forensic sciences*, 54(1):220—231.

F. Andrew Kozel, Steven J. Laken, Kevin A. Johnson, Bryant Boren, Kimberly S. Mapes, Paul S. Morgan, and Mark S. George. 2009b. Replication of Functional MRI Detection of Deception. *Open forensic science journal*, 2(1):6–11.

Sarah I. Levitan, Guzhen An, Mandi Wang, Gideon Mendels, Julia Hirschberg, Michelle Levine, and Andrew Rosenberg. 2015. Cross-Cultural Production and Detection of Deception from Speech. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, WMDD '15, page 1–8, New York, NY, USA. Association for Computing Machinery.

Sarah Ita Levitan, Guozhen An, Min Ma, Rivka Levitan, Andrew Rosenberg, and Julia Hirschberg. 2016. Combining Acoustic-Prosodic, Lexical, and Phonotactic Features for Automatic Deception Detection. In *Interspeech 2016*, pages 2006–2010.

Sarah Ita Levitan, Angel Maredia, and Julia Hirschberg. 2018a. Acoustic-Prosodic Indicators of Deception and Trust in Interview Dialogues. In *Proc. Interspeech 2018*, pages 416–420.

Sarah Ita Levitan, Angel Maredia, and Julia Hirschberg. 2018b. Linguistic Cues to Deception and Perceived Deception in Interview Dialogues. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1941–1950, New Orleans, Louisiana. Association for Computational Linguistics.

Peng-Hsuan Li, Tsu-Jui Fu, and Wei-Yun Ma. 2020. Why Attention? Analyze Bilstm Deficiency and Its Remedies in the Case of Ner. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

X. Liu, K. Tang, J. Hancock, J. Han, M. Song, R. Xu, V. Manikonda, and B. Pokorny. 2012. Socialcube: A text cube framework for analyzing social media data. In *International Conference on Social Informatics (SocialInformatics)*, pages 252–259, Los Alamitos, CA, USA. IEEE Computer Society.

A Mbaziira and J Jones. 2016. A Text-based Deception Detection Model for Cybercrime. In *Int. Conf. Technol. Manag*.

Gideon Mendels, Sarah Ita Levitan, Kai-Zhan Lee, and Julia Hirschberg. 2017. Hybrid Acoustic-Lexical Deep Learning Approach for Deception Detection. In *Proc. Interspeech 2017*, pages 1472–1476.

Robert W Mitchell. 1986. A framework for discussing deception. *Deception: Perspectives on human and nonhuman deceit*, pages 3–40.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

Pérez-Rosas, Verónica and Abouelenien, Mohamed and Mihalcea, Rada and Burzo, Mihai. 2015. Deception Detection Using Real-Life Trial Data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, page 59–66, New York, NY, USA. Association for Computing Machinery.

Victoria L. Rubin. 2014. TALIP Perspectives, Guest Editorial Commentary: Pragmatic and Cultural Considerations for Deception Detection in Asian Languages. *ACM Transactions on Asian Language Information Processing*, 13(2).

Alexandra L Sandham, Coral J Dando, Ray Bull, and Thomas C Ormerod. 2020. Improving Professional Observers' Veracity Judgements by Tactical Interviewing. *Journal of Police and Criminal Psychology*, pages 1–9.

Stefan Sarkadi. 2018. Deception. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 5781–5782. AAAI Press.

Chris N. H. Street, Jaume Masip, and Megan Kenny. 2019. *Understanding Lie Detection Biases with the*

Adaptive Lie Detector Theory (ALIED): A Boundedly Rational Approach, pages 227–247. Springer International Publishing, Cham.

Chris NH Street. 2015. Alied: Humans as adaptive lie detectors. *Journal of Applied Research in Memory and Cognition*, 4(4):335 – 343.

Harith H Thannoon, Wissam H Ali, and Ivan A Hashim. 2019. Design and Implementation of Deception Detection System Based on Reliable Facial Expression. *Journal of Engineering and Applied Sciences*, 14(15):5002–5011.

Harry C. Triandis, Peter Carnevale, Michele Gelfand, Christopher Robert, S. Arzu Wasti, Tahira Probst, Emiko S. Kashima, Thalia Dragonas, Darius Chan, Xiao Ping Chen, Uichol Kim, Carsten De Dreu, Evert Van De Vliert, Sumiko Iwao, Ken-Ichi Ohbuchi, and Paul Schmitz. 2001. Culture and Deception in Business Negotiations: A Multilevel Analysis. *International Journal of Cross Cultural Management*, 1(1):73–90.

Cristian Vaccari and Andrew Chadwick. 2020. Deepfakes and Disinformation: Exploring the Impact of Synthetic Political Video on Deception, Uncertainty, and Trust in News. *Social Media + Society*, 6(1):2056305120903408.

Sushma Venkatesh, Raghavendra Ramachandra, and Patrick Bours. 2019. Robust Algorithm for Multimodal Deception Detection. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 534–537.

Brianna L. Verigin, Ewout H. Meijer, Aldert Vrij, and Leonie Zauzig. 2019. The interaction of truthful and deceptive information. *Psychology, Crime & Law*.

Aldert Vrij, Lucy Akehurst, Laura Brown, and Samantha Mann. 2006. Detecting lies in young children, adolescents and adults. *Applied cognitive psychology*, 20(9):1225–1237.

Aldert Vrij, Samantha Mann, Sharon Leal, and Ronald P. Fisher. 2021. Combining verbal veracity assessment techniques to distinguish truth tellers from lie tellers. *The European Journal of Psychology Applied to Legal Context*, 13(1):9–19.

Dr Aldert Vrij and Sally Graham. 1997. Individual differences between liars and the ability to detect lies. *Expert Evidence*, 5(4):144–148.

Zhe Wu, Bharat Singh, Larry Davis, and V. Subrahmanian. 2018. Deception Detection in Videos.

Lina Zhou, Douglas P. Twitchell, Tiantian Qin, Judee K. Burgoon, and Jay F. Nunamaker. 2003. An exploratory study into deception detection in text-based computer-mediated communication. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 10 pp.–.

A Appendix

A.1 Welch's T-test Results

The Welch's t-test results are shown in Table A.1 and Table A.2 based on speakers' turns within QA pairs in three different scenarios as follows: (A) human-distrusted deceptive and truthful statements, (B) human-trusted deceptive and truthful statements, and (C) successful/unsuccessful deceptive and truthful statements. Emobase contains the fundamental frequency (F0) and its envelope, intensity (*INTENSITY*), loudness (*LOUDNESS*), 12 MFCC, probability of voicing (*VOICEPROB*), 8 line spectral frequencies (*LSPFREQ*), zero-crossing rate (*ZCR*), and delta regression coefficients. Then, these LLDs and their delta coefficients are applied to the following statistics [2] to generate the final feature vector. The number (ratio) represents the number of signatures in each feature set divided by its dimension base because every feature set has different dimensions. For instance, in the NER feature set under scenario (A), there are 7 significant indicators and its dimension base is 17, so the ratio (number) is calculated by 7 divided by 17.

A.2 A Real Example in the DDDM database

Table A.3 summarizes a real example in the DDDM database, and we also show its duration, transcripts in Mandarin Chinese, and translation in English. This data sample is grouped into (1) class in Table 1. The interrogator trusts the deceiver but the deceiver tells deception in fact.

---

[2] (1): amean, (2): iqr1-2, (3): iqr1-3 , (4): iqr2-3, (5): kurtosis, (6): linregc1, (7): linregc2, (8): linregerrA, (9): linregerrQ, (10): max, (11): maxPos, (12): min, (13): minPos, (14): quartile1, (15): quartile2, (16): quartile3, (17): range, (18): skewness, (19): stddev.

Table A.1: The results of all feature sets after performing Welch's t-test in three different scenarios: (A) human-distrusted deceptive and truthful statements, (B) human-trusted deceptive and truthful statements, and (C) successful/unsuccessful deceptive and truthful statements ("*" indicates the significance threshold, p-value, is smaller than 0.01; "**" is smaller than 0.001).

Modality	**Feature**	**Who's Feature**	**(A) (%)**	**Indicators**	**(B) (%)**	**Indicators**	**(C) (%)**	**Indicators**	**(B)$\bigcap$(C) (%)**	**Indicators**
Audio	Emobase	Int.	4.37	Table A.2	1.02	Table A.2	1.83	Table A.2	0.10	$\Delta\text{MFCC}^{4\text{th}}$–skewness
		Dec.	1.93		4.37		4.68		0.10	$\Delta\text{MFCC}^{7\text{th}}$–linregc2
	XLSR	Int.	3.07	-	3.75	-	2.79	-	0.44	-
		Dec.	2.21	-	3.54	-	2.92	-	0.52	-
Text	NER	Int.	**29.41**	EVENT**, PRODUCT**, WORK_OF_ART**, PERCENT, QUANTITY	11.76	MONEY*, PRODUCT*	**35.29**	PRODUCT**, EVENT*, MONEY*, PERCENT, QUANTITY, WORK_OF_ART	**11.76**	MONEY*, PRODUCT*
		Dec.	5.88	PERCENT	23.53	PERCENT**, DATE*, CARDINAL, PRODUCT	11.76	PERCENT**, WORK_OF_ART**	5.88	PERCENT
	POS	Int.	14.00	Neqb*, D, DM, Dfb, Nc, VCL, VA	10.00	DM**, VG**, Da*, Nb, A	6.00	DM**, Dfb**, VI**	2.00	DM
		Dec.	0.00	-	10.00	DM**, A, FW, Nd, V_2	8.00	A, Cbb, Dk, V_2	4.00	A, V_2
	LIWC	Int.	10.98	death**, adverb* leisure, cogproc, focusfuture*, filler, auxverb, discrep, othergram	2.44	you, cogproc	7.32	death**, negate, filler, ipron, I, Sixltr	0.00	-
		Dec.	1.22	female	3.66	bio**, sexual, power	3.66	you, informal, social	0.00	-
	BERT	Int.	6.51	-	4.43	-	8.46	-	0.26	-
		Dec.	10.55	-	18.36	-	3.52	-	0.78	-
	RoBERTa	Int.	8.59	-	5.99	-	9.51	-	0.65	-
		Dec.	10.03	-	17.45	-	6.77	-	1.30	-
TTD	CTD	Both	0.00	-	**30.00**	$\text{Int}_{\text{sd}}*$, $\text{Int}_{\text{ud}}*$, $\text{Int}_{\text{su}}*$, Int_{us}, $\text{Int}_{\text{d}}/\text{Dec}_{\text{d}}$, Int_{st}	15.00	Dec_{ud}, Dec_{sd}, Dec_{us}	0.00	-

Table A.2: The Welch's t-test results on Emobase in three different scenarios ("*" indicates the significance threshold, p-value, is smaller than 0.01; "**" is smaller than 0.001).

Scenario	**Interrogator**	**Deceiver**
(C)	$\mathrm{MFCC}^{\mathrm{3th}}$: (6*), $\mathrm{MFCC}^{\mathrm{8th}}$: (12, 18), $\mathrm{MFCC}^{\mathrm{9th}}$: (8, 9, 12, 19), $\mathrm{MFCC}^{\mathrm{10th}}$: (1, 10, 14, 16), $\Delta\mathrm{MFCC}^{\mathrm{4th}}$: (18*), $\Delta\mathrm{MFCC}^{\mathrm{5th}}$: (1), $\Delta\mathrm{MFCC}^{\mathrm{11th}}$: (14), $\mathrm{LSPFREQ}^{\mathrm{1th}}$: (1, 12), $\Delta\mathrm{LSPFREQ}^{\mathrm{1th}}$: (15*, 18), LOUDNESS: (1, 12)	$\mathrm{MFCC}^{\mathrm{1th}}$: (4*, 15), $\mathrm{MFCC}^{\mathrm{3th}}$: (3, 4, 8, 9, 19), $\mathrm{MFCC}^{\mathrm{5th}}$: (14), $\mathrm{MFCC}^{\mathrm{8th}}$: (1, 16), $\mathrm{MFCC}^{\mathrm{9th}}$: (1, 15, 16), $\mathrm{MFCC}^{\mathrm{11th}}$: (6, 7), $\Delta\mathrm{MFCC}^{\mathrm{1th}}$: (7), $\Delta\mathrm{MFCC}^{\mathrm{7th}}$: (7), $\Delta\mathrm{MFCC}^{\mathrm{11th}}$: (1. 14), $\mathrm{LSPFREQ}^{\mathrm{0th}}$: (2, 3*, 4, 16), $\mathrm{LSPFREQ}^{\mathrm{1th}}$: (1, 7, 15), $\mathrm{LSPFREQ}^{\mathrm{2th}}$: (1, 14, 15, 16), $\mathrm{LSPFREQ}^{\mathrm{6th}}$: (7), $\mathrm{LSPFREQ}^{\mathrm{7th}}$: (2*, 3, 8, 9*, 19*), $\Delta\mathrm{LSPFREQ}^{\mathrm{7th}}$: (18)VOICEPROB: (16), ZCR: (2*, 15), F0: (12*), ΔF0: (3, 4, 8*, 9, 14, 19)
(B)	$\mathrm{MFCC}^{\mathrm{4th}}$: (10), $\mathrm{MFCC}^{\mathrm{6th}}$: (18), $\mathrm{MFCC}^{\mathrm{7th}}$: (9), $\Delta\mathrm{MFCC}^{\mathrm{4th}}$: (18), $\Delta\mathrm{MFCC}^{\mathrm{8th}}$: (6, 7*), ΔVOICEPROB (4), $\Delta\mathrm{LSPFREQ}^{\mathrm{4th}}$: (18), $\Delta\mathrm{LSPFREQ}^{\mathrm{7th}}$: (6, 7)	$\mathrm{MFCC}^{\mathrm{2th}}$: (18), $\mathrm{MFCC}^{\mathrm{6th}}$: (8, 9, 12, 17, 18, 19*), $\mathrm{MFCC}^{\mathrm{8th}}$: (2*, 3, 8*, 9*, 12*, 17, 19*), $\mathrm{MFCC}^{\mathrm{9th}}$: (2*, 3, 8, 9, 18, 19*), $\mathrm{MFCC}^{\mathrm{10th}}$: (5, 18), $\Delta\mathrm{MFCC}^{\mathrm{6th}}$: (9, 12, 19), $\Delta\mathrm{MFCC}^{\mathrm{7th}}$: (6*, 7*), $\Delta\mathrm{MFCC}^{\mathrm{8th}}$: (1, 6, 7, 8*, 9*, 10*, 12, 16, 17*, 19*), $\Delta\mathrm{MFCC}^{\mathrm{12th}}$: (10), $\mathrm{LSPFREQ}^{\mathrm{3th}}$: (5, 18), $\Delta\mathrm{LSPFREQ}^{\mathrm{7th}}$: (7, 15), ΔINTENSITY: (18)
(A)	$\mathrm{MFCC}^{\mathrm{1th}}$: (12*, 17*), $\mathrm{MFCC}^{\mathrm{2th}}$: (1*, 7, 12*, 14*, 15*, 16, 17), $\mathrm{MFCC}^{\mathrm{3th}}$: (1, 6), $\mathrm{MFCC}^{\mathrm{8th}}$: (12), $\mathrm{MFCC}^{\mathrm{9th}}$: (10, 18), $\mathrm{MFCC}^{\mathrm{10th}}$: (1, 2, 3, 14), $\mathrm{MFCC}^{\mathrm{11th}}$: (12), $\mathrm{MFCC}^{\mathrm{12th}}$: (6), $\Delta\mathrm{MFCC}^{\mathrm{1th}}$: (12, 18), $\Delta\mathrm{MFCC}^{\mathrm{3th}}$: (15), $\Delta\mathrm{MFCC}^{\mathrm{4th}}$: (9), $\Delta\mathrm{MFCC}^{\mathrm{6th}}$: (7), $\Delta\mathrm{MFCC}^{\mathrm{9th}}$: (13, 14), $\Delta\mathrm{MFCC}^{\mathrm{10th}}$: (14), $\mathrm{LSPFREQ}^{\mathrm{0th}}$: (1), $\mathrm{LSPFREQ}^{\mathrm{1th}}$: (10), $\mathrm{LSPFREQ}^{\mathrm{3th}}$: (10), $\mathrm{LSPFREQ}^{\mathrm{4th}}$: (12, 17*), $\mathrm{LSPFREQ}^{\mathrm{5th}}$: (12, 17), $\mathrm{LSPFREQ}^{\mathrm{7th}}$: (5, 18), $\Delta\mathrm{LSPFREQ}^{\mathrm{1th}}$: (18*), $\Delta\mathrm{LSPFREQ}^{\mathrm{5th}}$: (12), VOICEPROB: (12, 19), ΔVOICEPROB: (10, 17)	$\mathrm{MFCC}^{\mathrm{2th}}$: (14), $\mathrm{MFCC}^{\mathrm{3th}}$: (2), $\mathrm{MFCC}^{\mathrm{4th}}$: (1, 14), $\mathrm{MFCC}^{\mathrm{5th}}$: (7), $\mathrm{MFCC}^{\mathrm{6th}}$: (9, 10, 17, 19*), $\mathrm{MFCC}^{\mathrm{8th}}$: (14), $\mathrm{MFCC}^{\mathrm{9th}}$: (18), $\mathrm{MFCC}^{\mathrm{11th}}$: (6), $\Delta\mathrm{MFCC}^{\mathrm{6th}}$: (1, 2, 9), $\mathrm{LSPFREQ}^{\mathrm{0th}}$: (3), $\Delta\mathrm{LSPFREQ}^{\mathrm{7th}}$: (2, 14), ΔF0: (18)

Table A.3: A Real Example in the DDDM database.

(Questioning Turn) (Duration) Interrogator's Questing Turns	**(Answering Turn) (Duration) Deceiver's Answering Turns**
(Q1)(09.6s)那因為你寫高中後有參加過吉他的公開比賽或演奏那請問是怎樣類型的公開 (According to your answer sheet, you have participated in public guitar competitions or performances after high school. What kind of public is that?)	(A1)(10.368s)其實就是呃一般類就是社團的那個發表會這樣主要是我那個大學的時候 (In fact, er, it's a general club's presentation, mainly when I was in university.)
(Q2)(01.92s)嗯大概是大幾的時候 (Um, what was your grade at tha time?)	(A2)(00.768s)大一的時候 (Freshman.)
(Q3)(04.864s)那那時候你是吉他那那你是吉他社嗎 (At that time, were you a member of a guitar club?)	(A3)(14.336s)大一的時候算吧因為就是大一的時候就是比較比較時間比較多所以去了蠻多社團所以也有去吉他社就是呃自己重頭開始練然後有參加過一個學期的就是成發這樣 (When I was a freshman, I have more time in the freshman year, so I went to a lot of clubs, so I also went to the guitar club, um, I started practicing on my own and then participated in presentation for a semester.)
(Q4)(08.704s)那那表演的時候是你個人獨秀還是大家一起彈的還是樂團 (So when you perform, is it your solo show , or is it the orchestra that everyone plays together?)	(A4)(12.928)主要是我跟另外一個就是社團的朋友就我們兩個呃我就是他彈我彈欸我彈主旋律他彈就是節奏這樣然後vocal的話就是一起 (Another friend and I from the club. Uh, he plays, I play, I play the main melody, he plays the rhythm, and the vocal is together.)
(Q5)(06.272s)那那目前你還有在繼續練吉他嗎 (Are you still practicing guitar?)	(A5)(08.064)後來就沒有了後來就是就比較喜歡去熱舞社這樣所以吉他社就沒有再去了 (Later, no. Later, I prefer to go to the hot dance club, so the guitar club I did not go.)
(Q6)(13.184s) 嗯那能簡單講一下吉他的基本入門五大和弦嗎就是最常出現的那幾個 (Um, can you briefly talk about the basic and common guitar five major chords that appear most often?)	(A6)(08.704)有點忘了我知道有C然後C1G吧我只記得這幾個對 (I forgot. I know that there are C and then C1G. I only remember these pairs. Yes.)

High-Quality Diversification for Task-Oriented Dialogue Systems

Zhiwen Tang **Hrishikesh Kulkarni** **Grace Hui Yang**
InfoSense, Department of Computer Science
Georgetown University
{zt79,hpk8,grace.yang}@georgetown.edu

Abstract

Many task-oriented dialogue systems use deep reinforcement learning (DRL) to learn policies that respond to the user appropriately and complete the tasks successfully. Training DRL agents with diverse dialogue trajectories prepare them well for rare user requests and unseen situations. One effective diversification method is to let the agent interact with a diverse set of learned user models. However, trajectories created by these artificial user models may contain generation errors, which can quickly propagate into the agent's policy. It is thus important to control the quality of the diversification and resist the noise. In this paper, we propose a novel dialogue diversification method for task-oriented dialogue systems trained in simulators. Our method, Intermittent Short Extension Ensemble (I-SEE),[1] constrains the intensity to interact with an ensemble of diverse user models and effectively controls the quality of the diversification. Evaluations on the Multiwoz dataset show that I-SEE successfully boosts the performance of several state-of-the-art DRL dialogue agents.

1 Introduction

Task-oriented dialogue agents assist human users to complete their tasks in multi-round human-agent interactions. Example tasks include booking a movie ticket or reserving a lunch table. Many agents use deep reinforcement learning (DRL) to learn good policies that respond appropriately in the dialogue and succeed in completing the task (Schulman et al., 2017; Takanobu et al., 2020; Peng et al., 2018). Due to the high cost of interactive training using real human users, simulators have been used to replace the humans (Shi et al., 2019). Early user simulators rely on rules hand-crafted by domain experts, who review and summarize dialogue templates from everyday conversations (Li et al., 2016b). Rule-based simulators manage routine task scenarios efficiently; but they cannot easily generate unconventional and more diverse dialogue acts marked by human spontaneity. Agents learning from these simulators may fail to handle rare queries and unseen situations. For example, in Table 1's left column, the system learns from a rule-based simulator and fails to answer an unexpected question "*In which area of the city is this hotel located?*" The user immediately ends the conversation and does not book the hotel.

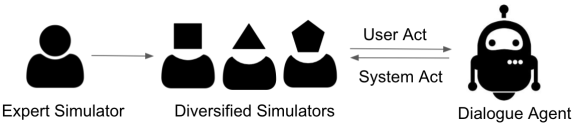

Figure 1: Agent Learns from Diversified Simulators.

Increasing dialogue diversity is a long-lasting research interest motivated by different needs. In dialogue generation, diversity is desired to avoid dull responses. In simulator construction and reinforcement learning (RL), diversity is desired for training robust agents. Although the uses are slightly different, ideas to improve diversification can be universal. Dialogue diversity can be improved via i) enforcing diversity in objective functions (such as maximize mutual information) of neural models (Li et al., 2016a; Baheti et al., 2018), ii) perturbing language rules (Niu and Bansal, 2019) or environment parameters (Tobin et al., 2017; Ruiz et al., 2019), iii) randomizing trajectory synthesis (Andrychowicz et al., 2017; Lu et al., 2019), iv) selecting more diverse data contributors (Stasaski et al., 2020), and v) sampling trajectories from a diverse set of environments (Chua et al., 2018; Janner et al., 2019).

Figure 1 illustrates the last idea, which has received much attention recently (Chua et al., 2018; Janner et al., 2019). This line of research diversi-

---

[1] I-SEE codes can be found at `https://github.com/smt-HS/I-SEE`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1861–1872
August 1–6, 2021.

No Diversification	Full Diversification	Quality-Controlled Diversifica. (I-SEE)
User: Hi! I would like to book hotel Leverton House. System: Leverton House is a cheap hotel. User: What is the post of Leverton House? System: Post of Leverton House is cb58rs. User: What is the type of the hotel? System: It is a guesthouse. User: Where is it located? System: Leverton House is located at 732-734 new market road. User: What is the hotel phone number? System: You can call the hotel at 01223292094. User: In which area of the city is this hotel located? System: There are many attractions in the centre of the city. [irrelevant] User: Ok, bye! System: Goodbye!	User: Hi! I would like to book hotel Leverton House. System: Hotel Leverton House is located on the east side of the city. User: What is the post of the hotel? System: The post of hotel Leverton House is cb58rs. ... System: ... Shall I go ahead and book it? User: Ok! I am not sure about the duration of stay or the number of people staying. System: Booking confirmed. Anything else? [success] User: I would also like to book a train from Peterborough to Cambridge arriving this Sunday at 15:15. System: Found a train arriving at 8:38. [incorrect] User: I want the one arriving at 15:15. System: Found a train arriving at 6:09. [incorrect] User: Ok, Bye! System: Have a good day!	User: Hi! I would like to book hotel Leverton House. System: Are you interested in booking hotel Leverton House on the east side? User: Where is it exactly located? What is the post, phone and type of the hotel? ... System: ... Would you like to go ahead and book it? User: Ok, Not sure on the duration of stay or number of people as of now. System: Booking confirmed. [success] User: I would also like to book a train from Peterborough to Cambridge arriving at 15:15 on Sunday. System: Would you like to go ahead and book tickets on TR7721? [correct] User: Ok. Not sure on the number of people. System: One ticket booked. [success] User: Great! Thanks! System: Have a good day!
Failure: The system fails to answer the area question from the user and it does not book the hotel for the user.	**Failure**: Even though the system successfully answers all questions from the user and books the hotel, it fails to identify the correct train based on arrival time and does not book the train ticket.	**Success**: Both hotel and train ticket have been successfully booked satisfying all constraints.

Table 1: Example Dialogues.

fies an agent's learning experiences by letting the agent interact with a diverse set of generative user models learned from an expert simulator. The idea involves little manual configuration and is often developed as model-based DRL (MBDRL) (Sutton and Barto, 1998). MBDRL methods alternate between learning an environment model and learning a policy. For a task-oriented dialogue agent, the environment model can be thought of a user model. It is a dynamic model updated to fit the trajectories the agent has collected so far; the policy then is optimized to maximize the expected long-term rewards within the model. Diversification of the user model is achieved by randomizing the parameter initialization of neural networks to our advantage. The agent, which is the policy learner, interacts with an ensemble of randomized user models to gain more diverse learning experiences.

However, one issue in this approach is that errors in (user) model learning may quickly propagate into policy learning. Table 1's middle column demonstrates a result from uncontrolled use of the diversified user models. In this example, even though the system successfully answers all questions from the user and books the hotel, the agent recommends two erroneous trains that do not satisfy the user's constraints and fail to book the ticket. This is because noise has been introduced to the training dialogues and they deviate too much from a legitimate conversation in real-life.

In this paper, we propose a novel dialogue diversification method, Intermittent Short Extension Ensemble (I-SEE), for task-oriented dialogues agents trained in simulators. First, I-SEE employs neural networks to learn a generative user model by imitating the expert simulator (Torabi et al., 2018). Second, it randomizes the parameter initialization of the neural networks to generate more user models, which are diversification from the original expert-built simulator. These randomized user models form an ensemble of diverse simulators, named Diverse User Model Ensemble (DUME). Third, during policy learning, the agent interacts with multiple simulators to obtain diverse training trajectories. Particularly, we propose to mix trajectory segments sampled from the expert simulator and trajectory segments sampled from the DUME. This is to constrain the degree of noise introduced by diversification and do not divert too far from the expert simulator. Moreover, we propose to include the DUME trajectories only moderately frequently and for a short horizon.

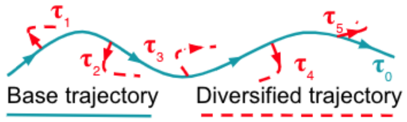

Figure 2: Conceptual illustration of I-SEE.

Figure 2 illustrates our idea conceptually. By constraining the degree of diversification, I-SEE effectively controls the training trajectories' quality while preserving their diversity. In Table 1's last (right) example, the I-SEE agent successfully takes the booking task to a logical conclusion by correctly finding the TR7721 train, which satisfies the user's time constraints. We apply I-SEE to a few best performing DRL dialogue methods and evaluate them on the Multiwoz (Budzianowski et al., 2018) dataset. Results show that using DUME and I-SEE in combination would significantly improve the performance of these state-of-the-art systems.

2 Related Work

2.1 Task-Oriented Dialogue Systems

Popular approaches for task-oriented dialogue systems include Sequence-to-Sequence (Seq2Seq) response generation (Vinyals and Le, 2015; Hosseini-Asl et al., 2020), knowledge graph-driven question answering (KG-QA) (Christmann et al., 2019; Moon et al., 2019; Young et al., 2018; Madotto et al., 2018, 2020), context-sensitive response retrieval (Aliannejadi et al., 2019; Yu et al., 2020; Qu et al., 2020; Wang and Ai, 2021) and RL (Buck et al., 2018; Peng et al., 2018; Tang and Yang, 2020; Luo et al., 2014; Li et al., 2017; Su et al., 2018; Peng et al., 2018).

Seq2Seq dialogue agents are generation methods. They use language models to capture the probability of one utterance given the previous, and based on the learned models to generate new utterances (Vinyals and Le, 2015; Hosseini-Asl et al., 2020). These supervised methods take advantage of deep neural networks and infer effective encoder-and-decoders from large amount of sequential training data. Modeling the dialogue states (Campagna et al., 2020) in the Seq2Seq architecture is a major interest in this line of research.

KG-QA dialogue agents enable reasoning and inference with pre-built knowledge graphs (KGs). The KGs can be about commonsense or domain-specific knowledge. A general KG can help a conversation more interesting and engaging (Moon et al., 2019; Young et al., 2018); while a specific KG can help accomplish the task more efficiently (Madotto et al., 2018, 2020). Methods in this category focus on scaling up the KGs (Madotto et al., 2020) and hopping mulitple steps on the KGs (Moon et al., 2019).

Retrieval-based dialogue agents leverage mature techniques in ad hoc retrieval and extend the techniques from individual queries to a session of them. Retrieval-based approaches do not rely on simulators; instead, learning from historical data, such as query logs, is still quite popular. This line of research focuses on revealing a user's mixed-initiative information need via asking back-and-force questions (Aliannejadi et al., 2019; Yu et al., 2020; Qu et al., 2020; Wang and Ai, 2021). However, when task complexity goes beyond the user's capability, these approaches may face difficulty in finding global solutions to the task goal.

RL-based dialogue agents can be grouped into model-free and model-based methods. Model-Free DRL (MFDRL) agents take a pre-built environment/simulator as it is and learn policies via direct interactions with it (Li et al., 2017; Dhingra et al., 2017; Li et al., 2017; Lipton et al., 2018; Su et al., 2018; Wu et al., 2020). On the contrary, model-based DRL (MBDRL) agents indirectly learn policies from the environment. MBDRL has two concurrent learning modules, namely model learning and policy learning. The *model learning* module can be thought of an additional computational layer between the environment and the agent. This provides opportunities to alter the original environment. MBDRL was originally proposed in robotics and control to speed up direct policy learning by inferring decision rules from past interactions and embedding them in the model. For dialogue agents, this middle layer of model learning acts as derived simulators (or learned user models) from the original expert simulator. Deep Dyna-Q (DDQ) (Peng et al., 2018) is an MBDRL method built upon Dyna (Sutton and Barto, 1998). D3Q (Su et al., 2018) employs generative adversarial networks (GAN) to minimize the difference between trajectories generated from the learned models and that from the original expert simulator, assuming that the expert simulator is the gold standard. Likewise, ADC (Wu et al., 2020) uses double critics to mitigate the impact of poorly-generated trajectories to stabilize the agent's performance. Our method belongs to the family of MBDRL, with a focus on diversification.

2.2 Diversification in Dialogues

Increasing dialogue diversity is a long-lasting research interest. Dialogue diversity can be improved via enforcing diversity objective functions (such as maximize mutual information) in neural models (Li et al., 2016a; Baheti et al., 2018), perturbing language rules (Niu and Bansal, 2019) or environment parameters (Tobin et al., 2017; Ruiz et al., 2019), randomizing trajectory synthesis (Andrychowicz et al., 2017; Lu et al., 2019), selecting more diverse data contributors (Stasaski et al., 2020), and sampling trajectories from a diverse set of environments (Chua et al., 2018; Janner et al., 2019). For instance, Campagna et al. augmented dialogue data using domain-independent transition rules and domain-specific ontology (Campagna et al., 2020). Niu and Bansal synthesized more diverse dialogue trajectories by choosing semantic-preserving language perturbations via RL (Niu and Bansal, 2019).

2.3 Diversification in DRL

In model-free DRL, diversification can be achieved by domain randomization (Tobin et al., 2017; Ruiz et al., 2019) or hindsight experience replay (Andrychowicz et al., 2017; Lu et al., 2019), without modeling the dynamics of the environment.

In model-based DRL, diversification is done by altering the learned environment/user model; which are the closest to our work. For instance, Chua et al. proposed probabilistic ensemble trajectory sampling (PETS) (Chua et al., 2018), which learns an ensemble of environment models and uses them for planning. The follow-up work (Janner et al., 2019) extended PETS with policy learning. Like us, Janner et al. concerned noise added by new trajectories generated by the derived environments. They proposed that the generation of new trajectories from the derived models should start from a beginning state shared with the original environment. These methods are mainly developed for robotics and work in continuous action space.

In this paper, we propose to obtain mixed training trajectories by branching from the original trajectory generated by the expert simulator and extending with new trajectories by the derived simulators. Different from (Janner et al., 2019), our method is designed for dialogue agents' discrete action space. In our method, each training trajectory has an overlap much larger than (Janner et al., 2019) has with the expert trajectory. This allows us to obtain smoother transition distributions to facilitate discrete action space better. In addition, our method can parameterize the intensity to branch out, so that the level of diversification can be controlled and adjusted.

3 Problem Setup

Task-Oriented Dialogue is the interactive process between a user and a dialogue agent, who work together to accomplish a task. The process begins with the user initiating the dialogue with a task goal in mind. The task goal can have *constraints* and *requests*. Constraints are requirements a system response must satisfy and requests are for missing information the user needs to accomplish the task. E.g., a user wants to book tickets of a movie to be played on weekends but does not know the theater's phone number. Here the constraint is $time = weekend$ and request is $phone\_number =?$. The dialogue ends when both parties say "good-bye" or the user abandons it.

Expert Simulator is the rule-based simulator built by human experts. It is denoted as M_0, which describes how a typical user would choose proper dialogue acts as the dialogue unfolds. The state of the expert simulator is s_t^u at time step t and the action is a_t^u selected from an action set A^u, which can be either *making requests* or *imposing constraints*. M_0 shows a mapping from s_t^u to a_t^u, describing patterns and behaviours for the human users, and provides feedback to and converse with the dialogue agent.

Diversified Simulator (or Diversified User Model) M_{ϕ_*} is a trainable user model that learns a parametric mapping from s_t^u to a'^u_t with parameter ϕ_*. It mimics the behavior of the expert simulator M_0. With different parameter initialization, we can create a set of diversified user models. This set of diversified simulators is called Diversified User Model Ensemble (**DUME**).

Dialogue Agent (DA) is the automatic response generator, who is expected to search in the knowledge base, reply the human users with relevant and correct answers, and make transactions following the user's requests. We use s_t^s, a_t^s to denote the state and action of the dialogue agent at time step t. The agent also receives a reward signal r_t as immediate feedback for its action a_t^s. Its state transition function P models the probability of its next state given the current state and actions from both the user and the DA: $s_{t+1}^s = P(s_t^s, a_t^s, a_t^u)$. In the DRL setting, the DA is the policy learner. It learns a policy π from a set of dialogue trajectories $\{\tau_*\}$. The goal of the agent is to learn a policy that can maximize the expected cumulative rewards $\mathbb{E}_\pi[\sum_t r_t]$ in a task-oriented dialogue.

Interaction Tuple $\mathscr{T}$ is the state-action-reward tuple generated when the DA interacts with a simulator or a real user. At the t^{th} dialogue turn, the t^{th} interaction tuple is $\mathscr{T}_t = (s^s_t, a^s_t, r_t, s^u_t, a^u_t)$.

Trajectory Segment τ^k_j is a sequence of interaction tuples when the DA interacts with a simulator (M_0 or M_{ϕ_i}) or a real user, starting from time step j to k: $\tau^k_j = [\mathscr{T}_j, \mathscr{T}_{j+1+1}, ..., \mathscr{T}_k]$, where $\mathscr{T}_{t\in[j,k]}$ is the t^{th} interaction tuple of the segment. Decided by the state transition function P, latter interaction tuples in τ depend on the earlier tuples. A **base trajectory segment** τ_0 is a trajectory that records the interaction between the expert simulator M_0 and the DA. A **diversified trajectory segment** τ' is a trajectory segment that records the interaction between a diversified simulator M_{ϕ_*} and the DA. A **full trajectory** τ_0^{full}=[$\mathscr{T}_0$..., $\mathscr{T}_T$] starts from the beginning of a dialogue, s.t., $j = 0$ and ends at T, where T is the entire dialogue's length.

4 Proposed Work

Our method aims to provide high-quality diversified training trajectories for task-oriented dialogue agents. We propose to (1) construct an ensemble of diversified user models called DUME and (2) intermittently branching out short trajectories from the base trajectory using DUME and employ the new trajectories in policy learning.

Figure 3 illustrates the proposed system architecture. In our design, the dialogue agent can interact with both the expert simulator and a diversified simulator. Usually the agent starts with interacting with the expert simulator since $t = 0$. At a branching step $t = p$, the agent switches to the diversified simulator to interact with, until the trajectory ends at $t = T$. The diversified simulator is obtained via imitation learning (from the expert simulator) and neural network initialization randomization. By controlling how frequently the branching should be performed and how long a diversified segment should be used, we effectively reach a balance between training data diversity and quality.

4.1 Constructing Diversified User Model Ensemble (DUME)

To enhance dialogue diversity, we propose to have the agent interact with an ensemble of diverse user models $\{M_{\phi_*}\}$. We use neural networks with different initialization to learn diversified user models from the expert simulator M_0, and form the DUME using these learned models.

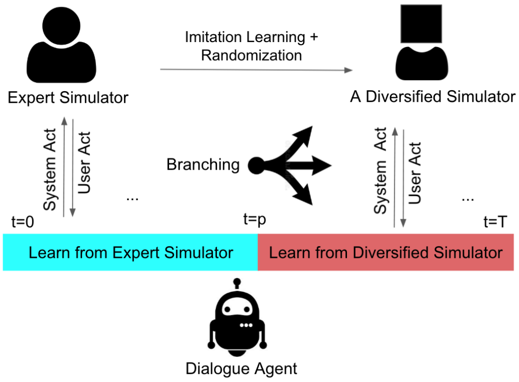

Figure 3: System Architecture

4.1.1 Learning a single user model

We propose to learn the user models from the expert simulator by behavior cloning (Torabi et al., 2018). For a single user model, we aim to learn a sequential decision-making function that maps $(s^u_1, s^u_2, ..., s^u_t...)$ to $(a^u_1, a^u_2, ..., a^u_t, ...)$. The training inputs are from the base trajectories τ_0, which includes a sequence of user state and user action pairs $\langle s^u_t, a^u_t \rangle$. The user state at the t^{th} turn is

$$s^u_t = (G, \bigcup_{t'=1}^{t-1} a^s_{t'}) \quad (1)$$

where G is the user goal, which can include both constraints and requests. $\bigcup_{t'=1}^{t-1} a^s_{t'}$ is the history of the dialogue agent's actions. The user action a^u_t is

$$a^u_t = (a^u_{t,1}, a^u_{t,2}, ..., a^u_{t,|A^u|}) \quad (2)$$

where $a^u_{t,i}$ are binary variables indicating whether the i^{th} dialogue act is active at dialogue turn t. A^u are the available dialogue acts for the user. The ending of a dialogue is also a special dialogue act.

Here a single user action can contain multiple dialogue acts. For instance, informing the destination and arrival time at the same dialogue turn when booking a train ticket. It means the number of dialogue acts per user action would vary. To allow the flexibility for modeling varied number of user acts, we propose to break the training trajectory (which is a sequence) τ_0 into individual state-action pairs and formulate the learning as choosing the right dialogue acts at a given state, i.e. learning the mapping from s^u_t to a^u_t. The optimization is done by minimizing the loss function $L(\phi)$:

$$L(\phi) = -\frac{1}{|A^u|}\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|A^u|} a_{t,i}\log M_\phi(s_t^u)_i + (1-a_{t,i})\log(1-M_\phi(s_t^u)_i) \tag{3}$$

where ϕ is the model parameter vector, $a_{t,i}$ is the ground truth indicator of whether the i^{th} dialog act is taken at time step t, and $M_\phi(s_t^u)_i$ estimates the probability of the i^{th} dialog act being chosen by the user model given s_t^u. The learning is performed by a multi-layer perceptron neural network parameterized by ϕ.

We are aware that the learning of the user models can be done using much more sophisticated methods. E.g., we can use more advanced neural network architectures and/or incorporate more information when defining the user states. However, these changes are not the main focus of this paper. The proposed user modeling is sufficient to support our investigation in exploiting them to improve diversification.

4.1.2 Forming a Diverse Ensemble

We propose to build an ensemble of diversified user models for better diversification. The ensemble, DUME, contains a set of E number of user models $M_{\phi_1}, M_{\phi_2}, ..., M_{\phi_E}$. Each of them is trained with behavior cloning as stated in Section 4.1.1. DUME diversifies the user models by initializing the behavior cloning with different seeds. Each user model is trained using a separate neural network; these neural networks share the same architecture but use randomized, different initial parameters ϕ_j. Our experiments (Section 5.4) show that the diversity in DUME dramatically increases, as E increases.

4.2 Policy Learning with I-SEE

One would imagine that the more diversified trajectories used in training, the more robust the policy would be. An intuitive idea is to interact with the diversified user models M_{ϕ_*} from the beginning to the end, without using M_0 at all. DDQ (Peng et al., 2018) indeed exploits this design. However, a dialogue trajectory completely generated by M_{ϕ_*} suffers from accumulation of generation errors because they may deviate too much from what a real conversation looks like.

In this paper, we propose to learn from training trajectories generated from mixed sources. Our idea is to have controlled diversification during policy learning, where some of the learning is done by learning from the original expert simulator and some is done by learning from the diversified user models in DUME. The ratio of the diversified portion can be controlled as a hyper-parameter. The following details our method.

Algorithm 1: Trajectory Generation

```
Input  : Simulator $M$,
         Dialogue agent policy $\pi$,
         Initial user state $s_0^u$,
         Maximum trajectory length $T$
Output : Dialogue trajectory dataset $D$;
$D = \varnothing$;
Initialize the user state to $s_0^u$;
for $T$ time steps do
    The user/simulator observes the state $s_t^u$ and
      takes action $a_t^u = M^i(s_t^u)$;
    The agent observes the state $s_t^s$ and takes action
      $a_t^s = \pi(s_t^s)$;
    The agent receives reward $r_t$;
    Store the interaction tuple $\langle s_t^s, a_t^s, r_t, s_t^u, a_t^u \rangle$ in
      $D$;
    if the user/simulator decides to end the dialogue
      in $a_t^u$ then
        break;
    end
end
return $D$
```

4.2.1 Diversifying the Trajectories

During policy learning, the dialogue agent collects training trajectories generated from the simulators, to keep refining its policy based on gradient ascent. Algorithm 1 details the trajectory generation process. In order to sample a trajectory, the policy learner, i.e. the dialogue agent, interacts with a user model to obtain interaction tuples step by step and store each individual tuple in a dataset D. To obtain an individual interaction tuple, the simulator needs to take an action based on its own user model, and then the agent performs an action based on the state and its current policy π. The agent receives rewards and the next state from the simulator. The interaction tuple is stored and would be used later to form a full trajectory. This process works the same regardless the agent interacting with the expert simulator or a diversified simulator.

In this work, we propose to diversify the agent's learning experiences by learning from trajectories generated from mixed sources. First, we generate a full base trajectory τ_0^{full}=[$\mathscr{T}_0$..., $\mathscr{T}_T$] from the expert simulator and store all its tuples. Second, we pick a branching tuple $\mathscr{T}_p \in \tau_0^{full}$ at a branching point $p \in (0, T)$. Third, from p onward, the trajectory is generated with a diversified user model M_{ϕ_*}, which would take an action a'^u_p different from the

Algorithm 2: Intermittent Short Extension Ensemble (I-SEE)

```
    Input  : Simulator ensemble size $E$
             Branching horizon $H$
             Diversification ratio $\eta$
    Output : Dialogue agent's policy $\pi$
1   Initialize an ensemble of $E$ user models;
2   Initialize the dialogue agent policy $\pi$;
3   while the dialogue agent's policy does not converge do
4       $D_{base}, D_{dvs} = \varnothing, \varnothing$ ;
5       for every episode do
6           Initialize the expert simulator $M_0$;
7           Observe the initial user state $s_0^u$;
8           $D_{base}$ =TrajectoryGeneration$(U, \pi, s_0^u, \infty)$;
9       end
10      while $|D_{dvs}| < \eta|D_{base}|$ do
11          Sample a simulator $M_{\phi_j}$ from the ensemble;
12          Sample a state $s_t^u$ from $D_{base}$ as the start state;
13          $D_{dvs} = D_{dvs} \bigcup$ TrajectoryGeneration$(M_{\phi_j}, \pi, s_t^u, H)$;
14      end
15      Update the dialogue agent's policy $\pi$ with $D_{base} \bigcup D_{dvs}$ ;
16      Update the simulator ensemble with $D_{base}$ using Eq. 3;
17  end
```

expert action a_p^u and the agent would also land in a different state ${s'}_{p+1}^{s} = P(s_p^s, a_p^s, {a'}_p^u)$.

Such interaction with the diversified simulator M_{ϕ_*} continues with H steps, resulting a diversified trajectory segment. The **diversified trajectory segment** $\tau_p'^{p+H}$ records the interaction between M_{ϕ_*} and the agent, extending the base trajectory τ_0 from a branching point p and running from $p+1$ onward. It is denoted as:

$$\tau_p'^{p+H} = [\mathscr{T}_p, \mathscr{T}'_{p+1}, ..., \mathscr{T}'_{p+H}],$$

where p is the branching point and $p > 0$, and H is τ_p''s horizon. The first interaction tuple in τ_p' is copied from the p^{th} turn in τ_0, i.e., $\mathscr{T}'_p = \mathscr{T}_p$. The **full trajectory with diversification** is thus $\tau_p^{full} = [\mathscr{T}_0, ..., \mathscr{T}_p, \mathscr{T}'_{p+1}, ..., \mathscr{T}'_{p+H}]$.

Our method generates parts of a dialogue with the diversified simulator and the other parts using the expert simulator. Each training trajectory thus has overlaps with the expert trajectory, which obtains smoother transition distributions to facilitate the discrete action space that a dialogue agent has.

4.2.2 Intermittent, Short Extensions

Further, we control the quality of diversification by using the DUME conservatively – only use the DUME trajectories for a short horizon and intermittently – to avoid accumulating generation errors.

Branching Horizon. The hyper-parameter H is the branching horizon that controls how far a trajectory is generated from DUME. The larger the horizon H, the more diverse the resulting trajectory. Setting H too small may cause the policy to be myopic as actions take time to show effects; whereas setting it too large may result in accumulation of errors. Our experiments show that using a moderately small $H = 5$ is preferable. An analysis is reported in the experiment section.

Branching Intensity. Another factor that determines the degree of diversification is the intensity of branchings. Instead of branching at every single step, our method only intermittently forks a diversified trajectory uniformly. This is done by setting a diversification ratio η between the times the agent interacting with the expert simulator M_0 and with DUME. The diversification ratio η is calculated as:

$$\eta = \frac{count(\mathscr{T}_i', \forall i \in D_{dvs})}{count(\mathscr{T}_j, \forall j \in D_{base})} \qquad (4)$$

where $\mathscr{T}_i'$ is a diversified interaction tuple stored in D_{dvs} and $\mathscr{T}_j$ is an interaction tuple stored in D_{base}. D_{base} and D_{dvs} are collections of individual interaction tuples obtained as Lines 4-14 in Algo. 2. A larger η means more diversified the agent's learning is. Algo. 2 shows the entire I-SEE algorithm.

5 Experiments

5.1 Experimental Setup

• **Dataset.** We evaluate the proposed approach on the Multiwoz (Budzianowski et al., 2018) dataset. Multiwoz is a large-scale benchmark dataset for task-oriented dialogue systems. It has seven task domains, including restaurant, hotel, attraction, taxi, train, hospital and police. One dialogue may involve multiple task domains, which is a good resemblance of how people converse in real life. Multiwoz provides 8,438 labelled dialogues, each dialogue of which is annotated by experts with a sequence of dialogue states and respective dialogue acts. Table 2 shows the dataset statistics. The expert simulator in Multiwoz starts a conversation and takes turns with a dialogue agent to dialogue. The simulator may request information from the agent or give the agent permission to do new bookings. At each turn, the simulator or the agent can perform one or more dialogue acts. The agent is

#Domains	#Dialogues	Total #Turns	Avg #Turns per dialogue
7	8,438	113,556	13.46
#Slots	#Values	Total DB Entries	Avg Entries per domain[2]
24	4,510	3,116	623

Table 2: Dataset Statistics (Multiwoz).

expected to 1) provide correct answers to requested information and 2) complete the booking, if asked.

- **Evaluation Metrics. Success** is our main metric, which is the success rate over all dialogue tasks tested. A task is successful if and only if 1) all the requested information is provided, and 2) all the booked entities match the user's requirements. **Inform F1** evaluates whether an agent provides the information requested by the user. It is calculated as $F1 = \frac{2Prec*Recall}{Prec+Recall}$, where $Prec$ and $Recall$ are the precision and recall of the information replied by the agent. **Match** evaluates whether the booked entities satisfy the user's requirement. It scores 1 if the correct entity is booked, otherwise 0. In the case of multiple bookings, the scores are averaged across all bookings. **#Turns** measures the number of turns a dialogue last regardless of its success. The less the turns, the better.
- **Baselines.** We compare the performance of a few top-performing DRL dialogue agents on the Multiwoz dataset with three settings. The settings are 1) the algorithm without diversification, 2) with full and uncontrolled diversification, and 3) with I-SEE. These baseline systems include state-of-the-art MFDRL and MBDRL methods and best performing DRL agents on Multiwoz. **DQN** (Deep Q-Network) (Mnih et al., 2015) is an off-policy MFDRL method, which approximates the value function of state-action pairs with a deep neural network and learns the function using experience replay. **PPO** (Proximal Policy Optimization) (Schulman et al., 2017) is an on-policy MFDRL algorithm, which optimizes a surrogate objective function which restricts the change of action distributions in a policy update. **GDPL** (Guided Dialogue Policy Learning) (Takanobu et al., 2019) is the best performer on Multiwoz. It uses inverse RL to reconstruct reward function and optimizes its policy with PPO. **DDQ** (Deep Dyna-Q) (Peng et al., 2018) is an MBDRL algorithm designed for task-oriented dialogue agents. DDQ generates complete trajectories from its environmental models, which is equivalent to our setting of DQN+full diversification. **MADPL** (Multi-Agent Dialogue Policy Learning) (Takanobu et al., 2020) is a multi-agent MFDRL method that trains the system and the user simulator simultaneously. It is also a leading performer on Multiwoz.
- **Implementation Details** We use Multiwoz's agenda-based simulator (Zhu et al., 2020) as the expert simulator. The DUME and policy networks and value networks in the baselines are learned using three-layer multi-layer perceptrons (MLPs). A learned user model has an input dimension of 230 and output of 67, with a hidden layer of 200 units. The DRL dialogue agents all use an the input layer of 553 units. PPO's policy network uses a hidden layer of 200 units and output of 166. PPO's value network has a hidden layer of 50 and output of 1. DQN also uses a hidden layer of 200 units and output of 166. The I-SEE dialogue agent is trained with a mix of expert simulator and diversified simulators as presented in the paper and tested with only the expert simulator.

5.2 Effectiveness

Table 3 presents the experiment results. The proposed method I-SEE outperforms the original algorithms and the full diversification variants for all baselines on the main metric, success, and the number of turns. The best performance is given by GDPL+I-SEE, with a success rate of 93.2 and only 7.32 dialogue turns on average. Moreover, the I-SEE variants perform the best on *Inform F1* for PPO and DQN, and on *Match* for PPO and GDPL. The improvements are large. These results suggest that diversification in general improves a DRL dialogue agent's effectiveness. However, full and uncontrolled diversification may worsen the performance; while a moderate level of diversification as we propose is a better choice.

5.3 Analysis of I-SEE

To understand why I-SEE works, we investigate the relationship between the degree of diversification and the success rate. GDPL is selected as the baseline system X. We study three I-SEE hyper-parameters that are responsible for the degree of diversification. They are the user model ensemble size E, branching horizon H, and diversification ratio η. As each of these parameters gets bigger, the degree of diversification increases. We plot the dialogue agent's learning curves w.r.t the three parameters in Figures 4a, 4b, and 4c, respectively.

We observe that a single optimum exists for each hyper-parameter when they reach the best success

---

[2] Five out of seven domains require querying the database.

Algorithm	Success ↑	Impr.	Inform F1 ↑	Impr.	Match ↑	Impr.	#Turns ↓	Impr.%
MADPL	70.1		76.26		90.98		8.96	
PPO	77.9		86.45		78.90		9.785	
PPO+Dvs.	69.0	(-8.9)	80.27	(–6.18)	70.55	(-8.35)	11.39	(-16.40%)
PPO+I-SEE	**84.5**	**(+6.6, +15.5)**	**88.91**	**(+2.46, +8.64)**	**86.29**	**(+7.93, +15.74)**	**8.88**	**(+9.25%, +22.04%)**
DQN	74.4		87.61		**92.91** best		11.54	
DQN+Dvs. (DDQ)	72.1	(-2.3)	84.26	(-3.35)	82.04	(-10.87)	11.78	(-2.08%)
DQN+I-SEE	**85.2**	**(+10.8, +13.1)**	**90.18**	**(+2.57, +5.92)**	92.59	(-0.32, **+10.55**)	**9.83**	**(+14.82%, +16.55%)**
GDPL	86.5		**94.97** best		83.90		7.64	
GDPL+Dvs.	72.8	(-13.7)	80.86	(-14.11)	81.10	(-2.80)	9.98	(-30.63)
GDPL+I-SEE	**93.2** best	**(+6.7, +20.4)**	91.83	(-3.14, **+10.97**)	**92.76**	**(+8.86, +11.66)**	**7.32** best	**(+4.19%, +26.65%)**

Table 3: Dialogue Effectiveness on Multiwoz. X+Dvs shows the improvement w.r.t. a baseline X. X+I-SEE reports the improvements w.r.t. X and X+Dvs, respectively.

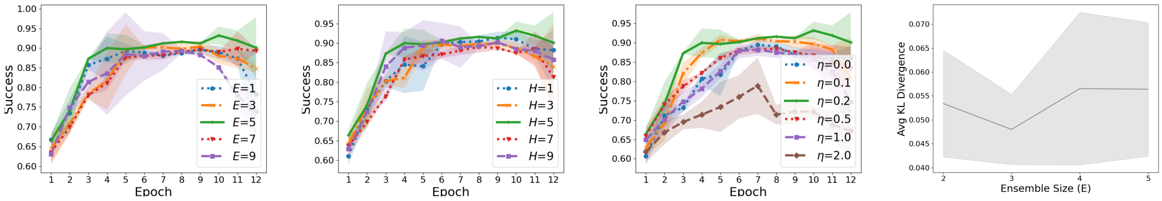

(a) Learning curves of different ensemble size (E)

(b) Learning curves of different diversification horizons (H)

(c) Learning curves of different diversification ratios (η)

(d) Evaluating the Diversity of DUME

Figure 4: Experiment Results on Multiwoz.

rate. As we increase the size of the ensemble with $E = 1, 3, 5, 7, 9$, the degree of diversity increases. Figure 4a shows that initially increasing the diversity helps improve the performance; However, the trend turns downwards after reaching the optimum when $E = 5$. Figures 4b and 4c demonstrate similar trends. In the end, the best combined I-SEE setting is $E = 5, H = 5$, and $\eta = 0.2$. This experiment suggest that diversification can only help an agent's learning to a certain extent; Too much diversification beyond that may introduce too much noise in the learning and hurt the agent's performance. Therefore, the degree of diversification must be carefully chosen in practice.

5.4 Analysis of DUME

DUME is our collection of trainable diversified user models. We calculate the average pairwise KL-divergence for every two models M_{ϕ_i} and $M_{\phi_j} \in$ DUME to directly measure the degree of diversity within DUME. Each user model is run on the same stavte sequence $\{s^u_1, ..., s^u_t ...\}$ and outputs an action sequence $\{a'^u_1, ..., a'^u_t ...\}$. Since each a'^u_t may contain multiple dialogue acts, we break down every a'^u_t into individual dialogue acts and calculate the distribution over the dialogue act set A^u. The mean μ and standard deviation σ of the KL divergences are plotted in Figure 4d. We can see that as DUME has bigger size, both μ and σ increase; which means the differences between the DUME simulators dramatically increase and they would add much diversity into the agent's learning.

6 Conclusion

This paper presents Intermittent Short Extension Ensemble (I-SEE), a DRL diversification method that successfully improves dialogue diversity and policy robustness while maintaining high data quality. I-SEE uses an ensemble of trainable user models to achieve diversity and controls the diversification quality by branching from original dialogue trajectories only for a short horizon and intermittently. Our experiments on Multiwoz show that using I-SEE can significantly improve several best state-of-the-art DRL dialogue agents.

Acknowledgment

This research was supported by the U.S. National Science Foundation Grant no. IIS-145374. Any opinions, findings, conclusions, or recommendations expressed in this paper are of the authors, and do not necessarily reflect those of the sponsor.

References

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 475–484, New York, NY, USA. Association for Computing Machinery.

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5048–5058.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3970–3980, Brussels, Belgium. Association for Computational Linguistics.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S. Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 122–132. Association for Computational Linguistics.

Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. 2019. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 729–738. ACM.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4759–4770.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 484–495. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12498–12509.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Çelikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 733–743. Asian Federation of Natural Language Processing.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016b. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5237–5244. AAAI Press.

Keting Lu, Shiqi Zhang, and Xiaoping Chen. 2019. Goal-oriented dialogue policy learning from failures. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2596–2603. AAAI Press.

Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win search: Dual-agent stochastic game in session search. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 587–596, New York, NY, USA. ACM.

Andrea Madotto, Samuel Cahyawijaya, Genta Indra Winata, Yan Xu, Zihan Liu, Zhaojiang Lin, and Pascale Fung. 2020. Learning knowledge bases with parameters for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2372–2394. Association for Computational Linguistics.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1468–1478. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 845–854. Association for Computational Linguistics.

Tong Niu and Mohit Bansal. 2019. Automatically learning data augmentation policies for dialogue tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1317–1323. Association for Computational Linguistics.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2182–2192. Association for Computational Linguistics.

Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. 2019. Learning to simulate. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. 2019. How to build user simulators to train rl-based dialog systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1990–2000. Association for Computational Linguistics.

Katherine Stasaski, Grace Hui Yang, and Marti A. Hearst. 2020. More diverse dialogue datasets via diversity-informed data collection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4958–4968, Online. Association for Computational Linguistics.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3813–3823. Association for Computational Linguistics.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

Ryuichi Takanobu, Runze Liang, and Minlie Huang. 2020. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 625–638. Association for Computational Linguistics.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 100–110. Association for Computational Linguistics.

Zhiwen Tang and Grace Hui Yang. 2020. Corpus-level end-to-end exploration for interactive systems. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2527–2534. AAAI Press.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 23–30. IEEE.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4950–4957. ijcai.org.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Zhenduo Wang and Qingyao Ai. 2021. Controlling the risk of conversational search via reinforcement learning. *CoRR*, abs/2101.06327.

Yen-Chen Wu, Bo-Hsiang Tseng, and Milica Gasic. 2020. Actor-double-critic: Incorporating model-based critic for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 854–863. Association for Computational Linguistics.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4970–4977. AAAI Press.

Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul N. Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020. Few-shot generative conversational query rewriting. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1933–1936. ACM.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 142–149. Association for Computational Linguistics.

Structured Refinement for Sequential Labeling

Yiran Wang[1]*, **Hiroyuki Shindo**[2], **Yuji Matsumoto**[3], **Taro Watanabe**[2]
[1]National Institute of Information and Communications Technology (NICT), Kyoto, Japan
[2]Nara Institute of Science and Technology (NAIST), Nara, Japan
[3]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan
`yiran.wang@nict.go.jp`, `shindo@is.naist.jp`,
`yuji.matsumoto@riken.jp`, `taro@is.naist.jp`

Abstract

Filtering target-irrelevant information through hierarchically refining hidden states has been demonstrated to be effective for obtaining informative representations. However, previous work simply relies on locally normalized attention without considering possible labels at other time steps, the capacity for modeling long-term dependency relations is thus limited. In this paper, we propose to extend previous work with globally normalized attention, e.g., structured attention, to leverage structural information for more effective representation refinement. We also propose two implementation tricks to accelerate CRF computation and an initialization trick for Chinese character embeddings to further improve performance. We provide extensive experimental results on various datasets to show the effectiveness and efficiency of our proposed method.

1 Introduction

Sequential labeling tasks, e.g., named entity recognition (NER) and part-of-speech (POS) tagging, play an important role in natural language processing. Figure 1 shows two examples of sequential labeling tasks. Early studies focused on introducing rich features to improve performance. For example, to handle out-of-vocabulary words by introducing morphological features, Lample et al. (2016) and Ma and Hovy (2016) leveraged character-level features, whereas Heinzerling and Strube (2019) exploited subword-level features. Moreover, introducing long-term dependency features is also found to be beneficial for sequential labeling. Jie and Lu (2019) attempted to explicitly exploit dependency relations with additional annotations, while Zhang et al. (2018) and Chen et al. (2019) endeavored to learn these relations implicitly with more complex encoders.

The	chairman	of	the	Senate	Finance	Committee
O	**O**	**O**	**B-ORG**	**I-ORG**	**I-ORG**	**E-ORG**

The	financing	system	is	created	in	the	new	law
DT	**NN**	**NN**	**VBD**	**VBN**	**IN**	**DT**	**JJ**	**NN**

Figure 1: Examples of NER (top) and POS tagging (bottom). For NER, "the Senate Finance Committee" is a named entity of type ORG (organization). The prefixes `S-`, `I-`, or `E-` indicate this word is located at the beginning, intermediate, or ending of the current named entity, while `O` signifies this word is outside any named entity. In the case of POS tagging, each tag is a part-of-speech category. For instance, `NN` represents a singular noun and `VBN` is the past participle of a verb.

However, as Tishby and Zaslavsky (2015) pointed out, features are not created equal, only the target-relevant features are profitable for improving model performance. Recently, Cui and Zhang (2019) proposed a hierarchically-refined label attention network (LAN), which explicitly leverages label embeddings and captures long-term label dependency relations through multiple refinements layers.

Individually picking up the most likely label at each time step is undoubtedly critical, however, considering the entire historical progress is also indispensable. We find that the locally normalized attention, which Cui and Zhang (2019) used to leverage information from label embeddings, can eventually hurt performance. Since it only considers the current time step but ignores labels at other time steps, thus we presume its ability to capture long-term dependency relations is limited.

On the other hand, Kim et al. (2017) incorporated neural networks with probabilistic graphical models to obtain structural distributions as an alternative to conventional attention mechanisms. Their method relies on attending to cliques of linear-chain conditional random fields (CRF). These in-

---

[*] This work was done when the first author was at NAIST.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1873–1884
August 1–6, 2021.

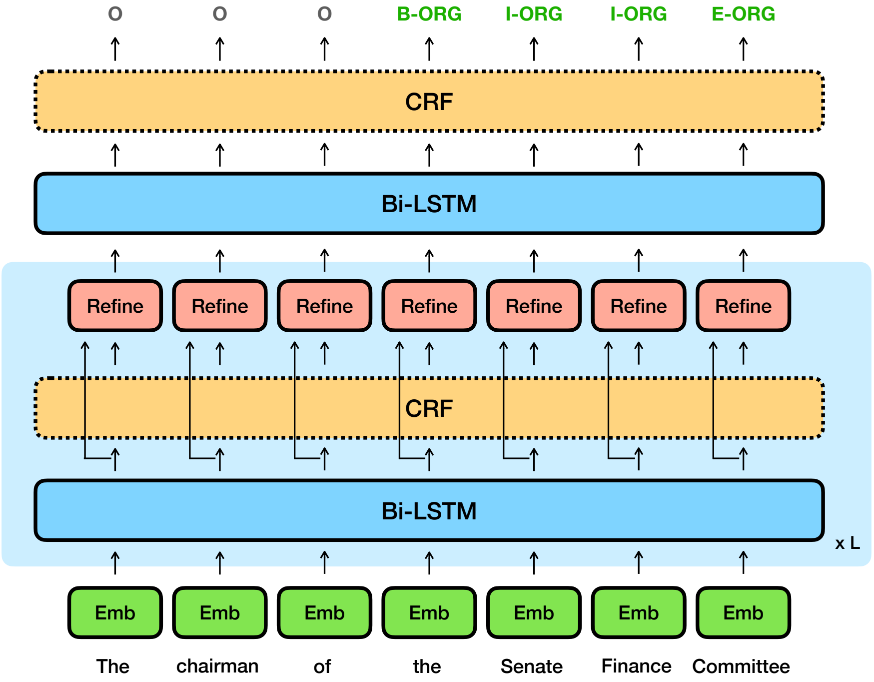

Figure 2: The architecture of the proposed model. The dotted lines mean these components are shared across layers.

ferred inner structures, i.e., represented as the marginal probabilities, are not the targets of their tasks but only serve as the latent variables, thus they do not impose direct supervision on these attention weights. In contrast, since we consider to repeatedly refine these inferred structures to obtain the final outputs, we compute structural attention over these target labels instead, without introducing unobserved variables.

In this paper, we propose a novel structured refinement mechanism by combining representation refinement and structured attention. Following and extending Cui and Zhang (2019), we hierarchically refine hidden representations with global normalized structured attention, i.e., the marginal probability of CRF. Besides, to impose direct supervision on the target structures, we share the label embeddings and the transition matrix of CRF across layers. Our method can be considered as iteratively re-constructing hidden representations with only label embeddings, and thus it is capable of filtering target-irrelevant information out.

Besides, we propose a character embedding initialization trick to enhance performance on Chinese datasets and two CRF implementation tricks to accelerate computation.

Our contributions are considered as four-folds, (a) we propose a novel structured refinement network by combing representation refinement and structured attention for sequential labeling tasks, (b) we propose an initialization trick for Chinese character embeddings, (c) we propose two implementation tricks to accelerate CRF training and decoding, (d) and we prove the effectiveness and efficiency of our model through extensive experiments for NER and POS tagging on various datasets.

2 Baseline

Formally speaking, given a token sequence $\{x_t\}_{t=1}^n$, the aim of sequential labeling tasks is to find the most probable label sequence $\{y_t\}_{t=1}^n$.

2.1 Label Attention Network

Label attention network (Cui and Zhang, 2019) consists of an embedding layer followed by several encoding and refinement layers alternatively. The decoding layer is a bidirectional LSTM followed by a refinement layer.

Embedding Layer Cui and Zhang (2019) employed the concatenation of word and character-based word representations as the token representations $\boldsymbol{x}_t = [\boldsymbol{w}_t, \boldsymbol{c}_t]$, they convert words to word embeddings $\boldsymbol{w}_t \in \mathbb{R}^{d_w}$ and use a character-level bidirectional LSTM to build character-based word embeddings $\boldsymbol{c}_t \in \mathbb{R}^{d_c}$.

Encoding Layer They utilized an independent bidirectional LSTM for each layer l as follows,

$$\{\boldsymbol{h}_t^{(l)}\}_{t=1}^n = \mathrm{LSTM}^{(l)}\left(\{\tilde{\boldsymbol{h}}_t^{(l-1)}\}_{t=1}^n\right) \tag{1}$$

where $\tilde{\boldsymbol{h}}_t^{(l-1)}$ is the refined representation from the last refinement layer. Specially, hidden vector $\tilde{\boldsymbol{h}}_t^{(0)}$ is the token representation $\boldsymbol{x}_t$. After this, they employ a refinement layer, which is called "label-attention inference sublayer" in the original paper, to refine hidden states. They make use of attention mechanism (Vaswani et al., 2017) to produce the attention matrix $\alpha_{t,j}^{(l)}$ as in Equation 2, and further calculate the label-aware hidden states $\hat{\boldsymbol{h}}_t^{(l)} \in \mathbb{R}^{d_h}$, which jointly encode information from the token representation subspace and the label representation subspace.

$$\alpha_{t,j}^{(l)} = \underset{j\in\{1,\dots,m\}}{\mathrm{softmax}}\left(\frac{(\mathbf{Q}^{(l)}\boldsymbol{h}_t^{(l)})^\top(\mathbf{K}^{(l)}\boldsymbol{v}_{y_j})}{\sqrt{d_h}}\right) \tag{2}$$

$$\hat{\boldsymbol{h}}_t^{(l)} = \sum_{j=1}^{m} \alpha_{t,j}^{(l)} \cdot (\mathbf{V}^{(l)}\boldsymbol{v}_{y_j}) \tag{3}$$

Where $\mathbf{Q}^{(l)}, \mathbf{K}^{(l)}, \mathbf{V}^{(l)} \in \mathbb{R}^{d_h \times d_h}$ are all parameters, and $\boldsymbol{v}_{y_j} \in \mathbb{R}^{d_h}$ is the embedding of label $y_j \in \mathcal{Y}$. In practice, they use multiple heads to capture representations from multiple aspects in parallel. After that, they concatenate the hidden state $\boldsymbol{h}_t^{(l)}$ and the label-aware hidden state $\hat{\boldsymbol{h}}_t^{(l)}$ as the refined representation $\tilde{\boldsymbol{h}}_t^{(l)} \in \mathbb{R}^{2d_h}$, and feed it into the next encoding layer.

$$\tilde{\boldsymbol{h}}_t^{(l)} = [\boldsymbol{h}_t^{(l)}, \hat{\boldsymbol{h}}_t^{(l)}] \tag{4}$$

Decoding Layer Similar to the encoding layer, the decoding layer contains a bidirectional LSTM and a refinement layer, but at this layer, the attention matrix $\alpha_{t,j}^{(L+1)}$ only servers as the label probability distribution to predict the most probable label sequence.

$$p(\boldsymbol{y} \mid \boldsymbol{h}) = \prod_{t=1}^{n} \alpha_{t,y_t}^{(L+1)} \tag{5}$$

3 Proposed Method

3.1 Structured Refinement

A notable highlight of the model of Cui and Zhang (2019) is that it is not equipped with the commonly used CRFs (Lample et al., 2016; Ma and Hovy, 2016), however, it still can achieve remarkable performance. And just because of abandoning the computationally expensive CRFs, their model obtains a significant acceleration on both training and decoding stages. However, we find that the time step independent attention, i.e., the softmax operation in Equation 2, only considers these labels at the current time step and ignores all the possible label combinations at other time steps, thus the performance is eventually degraded since the ability of capturing long-term dependency relation is local and limited. We thus bring CRF back and use the marginal probability to construct refined representations. We claim replacing the attention matrix $\alpha_{t,j}^{(l)}$ with the globally normalized marginal probability can capture long-term dependency relations more effectively.

The potential function of CRF is defined as,

$$\phi\left(y_{t-1}, y_t, \boldsymbol{h}_t^{(l)}\right) = A_{y_{t-1},y_t} + \boldsymbol{h}_t^{(l)\top}\boldsymbol{v}_{y_t} \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ is the transition matrix, A_{y_{t-1},y_t} denotes the transition score from label y_{t-1} to label y_t, and $\boldsymbol{v}_{y_t}$ is the embedding of label y_t. The conditional probability of a specified label sequence $\boldsymbol{y}$ can be described as

$$p\left(\boldsymbol{y} \mid \boldsymbol{h}^{(l)}\right) = \frac{1}{Z\left(\boldsymbol{h}^{(l)}\right)} \exp \sum_{t=1}^{n} \phi\left(y_{t-1}, y_t, \boldsymbol{h}_t^{(l)}\right) \tag{7}$$

$$Z\left(\boldsymbol{h}^{(l)}\right) = \sum_{\boldsymbol{y}'\in\mathcal{Y}^n} \exp \sum_{t=1}^{n} \phi\left(y'_{t-1}, y'_t, \boldsymbol{h}_t^{(l)}\right) \tag{8}$$

where $Z\left(\boldsymbol{h}^{(l)}\right)$ is the global normalization term, commonly known as the partition function. Furthermore, the marginal probability is defined as follow.

$$\mu_t\left(y_j, \boldsymbol{h}^{(l)}\right) = \sum_{\boldsymbol{y}':Y'_t=y_j} p\left(\boldsymbol{y}' \mid \boldsymbol{h}^{(l)}\right) \tag{9}$$

Marginal probability stands for the sum of the probabilities of all possible label sequences that emit label y_j at time step t. Calculating marginal probability requires enumerating all possible structures, and it thus can be called globally normalized probability or structured attention.

We replace the locally normalized attention $\alpha_{t,j}^{(l)}$ in Equation 3 with our globally normalized one, i.e., $\mu_t\left(y_j, \boldsymbol{h}^{(l)}\right)$. Furthermore, we employ residual connection (He et al., 2016) and layer normalization (Ba et al., 2016), instead of concatenation, to construct the refined representation $\tilde{\boldsymbol{h}}_t^{(l)} \in \mathbb{R}^{d_h}$,

$$\hat{\boldsymbol{h}}_t^{(l)} = \sum_{j=1}^{m} \mu_t\left(y_j, \boldsymbol{h}^{(l)}\right) \cdot \left(\mathbf{V}^{(l)} \boldsymbol{v}_{y_j}\right) \tag{10}$$

$$\tilde{\boldsymbol{h}}_t^{(l)} = \text{LayerNorm}\left(\boldsymbol{h}_t^{(l)} + \max\left(\mathbf{0}, \hat{\boldsymbol{h}}_t^{(l)}\right)\right) \tag{11}$$

where $\mathbf{V}^{(l)} \in \mathbb{R}^{d_h \times d_h}$ is a matrix parameter. The obtained refined representation $\tilde{\boldsymbol{h}}_t^{(l)}$ is then fed into the next layer.

3.2 Computing Marginal probability

Conventional method to compute the marginal probability $\mu_t\left(y_j, \boldsymbol{h}^{(l)}\right)$ requires running the forward-backward algorithm. Fortunately, as Eisner (2016) indicates, merely computing the log-partition function, $\log Z\left(\boldsymbol{h}^{(l)}\right)$, and differentiating it with an automatic differentiation library yields equivalent marginal probability efficiently. Thus, we use the `torch.autograd.grad` function of `PyTorch` to compute the marginal probability as follow.

$$\mu_t\left(y_j, \boldsymbol{h}^{(l)}\right) = \frac{\partial \log Z\left(\boldsymbol{h}^{(l)}\right)}{\partial\left(\boldsymbol{h}_t^{(l)\top} \boldsymbol{y}_j\right)} \tag{12}$$

3.3 Training and Decoding

We train our model by maximizing the log-likelihood with the back-propagation algorithm. The objective function is defined as follow,

$$\mathcal{L} = -\log\, p\left(\boldsymbol{y} \mid \boldsymbol{h}^{(L+1)}\right) \tag{13}$$

We apply the Viterbi algorithm (Forney, 1973) to efficiently search for the most probable label sequences on the decoding stage.

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}' \in \mathcal{Y}^n}{\arg\max}\, p\left(\boldsymbol{y}' \mid \boldsymbol{h}^{(L+1)}\right) \tag{14}$$

3.4 Complexity and Implementation Tricks

One concern regarding our proposed method is its computational complexity, as it requires to compute not only the partition function but also the marginal probability. Calculating the partition function, as in Equation 8, is the well-known bottleneck of CRF computation. And this is commonly achieved through reducing potential matrices by applying matrix multiplications. Similar to Rush (2020), we make use of the associative property of matrix multiplication to accelerate computation. The product of multiplying matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ is equivalent to the product of $\mathbf{AB}$ and $\mathbf{CD}$. Leveraging the power of GPU to compute $\mathbf{AB}$ and $\mathbf{CD}$ in parallel, and recursively applying this trick, we can reduce the time complexity of obtaining the partition function from $\mathcal{O}\left(\sum_{i=1}^{|\mathcal{B}|} |\boldsymbol{x}|_i\right)$ to $\mathcal{O}\left(\sum_{i=1}^{|\mathcal{B}|} \log |\boldsymbol{x}|_i\right)$, where $|\boldsymbol{x}|_i$ is the length of i-th sentence in batch $\mathcal{B}$. Moreover, instead of padding the sequence length $|\boldsymbol{x}_i|$ out to the nearest power of two as Rush (2020) does, we pre-compile argument indices of the matrix multiplication to handle the variant sentence length issue in a batch. Our method can effectively avoid out-of-memory error since we don't waste memory for paddings. This pre-compiling trick can further reduce the time complexity to $\mathcal{O}\left(\max_i \log |\boldsymbol{x}|_i\right)$. We release our CRF implementation with these two tricks as an independent library[1] for future study and use.

3.5 Character Embeddings Initialization

We describe a trick for Chinese character embeddings initialization. The most striking difference between Chinese and English is that the minimal semantic units, i.e., sememes, of Chinese are characters instead of words or subwords. The character vocabulary size of Chinese, e.g., around 2,000 on the OntoNote 5.0 dataset, is markedly larger than English, e.g., around 100 on the OntoNotes 5.0 English dataset. Existing models (Zhang and Yang, 2018; Li et al., 2020a) generally focused on introducing additional pre-trained character embeddings on the top of lexicon embeddings, and attempted to selectively leverage information from both of them according to the different word segmentation schemes. However, we notice that most of these characters already exist in the word vocabulary as single-character words, thus we employ a randomly initialized orthogonal matrix[2] to project the pre-trained word embeddings into the same dimension as the character embeddings, and use these projected embeddings for initialization.

4 Experiments

4.1 Datasets

We conduct experiments on the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and the OntoNotes 5.0 (Weischedel et al., 2013) datasets for English NER, and on the OntoNotes 5.0 and the OntoNotes 4.0 datasets for Chinese NER experiments. We also conduct experiments on the Wall

---

[1] `https://github.com/speedcell4/torchlatent`

[2] `torch.nn.init.orthogonal_`

Task	Dataset	Language	Sentences	$\|\mathcal{Y}\|$
NER	OntoNotes 5.0	English	59,924 / 8,528 / 8,262	73
	CoNLL 2003	English	14,987 / 3,466 / 3,684	17
	OntoNotes 5.0	Chinese	36,487 / 6,083 / 4,472	73
	OntoNotes 4.0	Chinese	15,724 / 4,301 / 4,346	17
POS	WSJ	English	38,219 / 5,527 / 5,462	45
	UD 2.2	English	12,544 / 2,003 / 2,078	50

Table 1: Dataset statistics, where the "Sentences" column displays the number of sentences in train/dev/test split respectively, the $|\mathcal{Y}|$ column displays the number of target label types. For NER datasets, we count types with the IOBES labeling scheme.

Street Journal (WSJ) dataset (Marcus et al., 1993) and the Universal Dependencies (UD) v2.2 English dataset for POS tagging experiments.

The only data pre-processing that we have performed is replacing digital tokens with a special token. And we convert labels to the IOBES labeling scheme (Ramshaw and Marcus, 1995; Ratinov and Roth, 2009) on NER datasets. The dataset statistics are provided in Table 1.

4.2 Hyper-parameter Settings

Following Cui and Zhang (2019) and Jie and Lu (2019), 100-dimensional Glove (Pennington et al., 2014) word embeddings are utilized for all the English experiments, and 300-dimensional FastText (Mikolov et al., 2018) word embeddings are employed for Chinese experiments. The dimension of character embeddings is 30, and the hidden states dimension d_c of the character bidirectional LSTM is 100, i.e., 50 in each direction. We apply dropout (Srivastava et al., 2014) on token representations with a rate of 0.5.

For encoding and refinement layers, the dimension of the hidden state d_h of bidirectional LSTMs is 600, i.e., 300 in each direction. We apply dropout on hidden states $\boldsymbol{h}_t^{(l)}$ with a rate of 0.5 before feeding into refinement layers. The number of refinement layers L is just 1.

We optimize our model by applying stochastic gradient descent (SGD) with decaying learning rate $\eta_\tau = \eta_0/(1 + 0.075 \cdot \tau)$, where τ is the index of the current epoch, and the initial learning rate η_0 for Chinese experiments without contextual word representations is 0.05, and for all the other experiments we use 0.1. The weight decay rate is 10^{-8}, the momentum is 0.15, the batch size is 10, the number of epochs is 100, and gradients exceed 5 will be clipped.

In addition, since the pre-trained contextualized word embeddings technique is widely accepted as a new fundamental utility of natural language processing, we also conduct experiments with ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). In these settings, tokens are represented as $\boldsymbol{x}_t = [\boldsymbol{w}_t, \boldsymbol{c}_t, \boldsymbol{e}_t]$, where $\boldsymbol{e}_t$ is the contextual word representation.

ELMo vectors are obtained by averaging output vectors over all layers of ELMo. For English experiments, we use the `original` checkpoint, and use the checkpoints provided by Che et al. (2018) for Chinese experiments.

BERT representations are the averages all BERT subword embeddings in the last four layers. Following Li et al. (2020b) and Li et al. (2020a), we utilize `bert-large-cased` and `hfl/chinese-bert-wwm` checkpoints for English and Chinese experiments respectively.

4.3 Evaluation

NER experiments are evaluated by using F_1 scores, and POS tagging experiments are evaluated with accuracy scores. All of our experiments were run 4 times with different random seeds, and the averaged scores are reported in the following tables.

Our models[3] are implemented with deep learning framework `PyTorch` (Paszke et al., 2019) and we ran experiments on `GeForce GTX 1080Ti` with 11 GB memory.

4.4 Experimental Results

4.4.1 Named Entity Recognition

Table 2 compares the performance of our proposed method and baselines on the OntoNotes 5.0 English dataset. Our model significantly outperforms Cui and Zhang (2019) and Jie and Lu (2019) by 0.49 and 0.13 F_1 scores respectively. These results demonstrate that our model can filter irrelevant information more effectively than Cui and Zhang (2019). Notably, the model of Jie and Lu (2019) relies on external dependency annotations, whereas our model requires no external knowledge[4]. In the case of employing ELMo, our model outperforms Jie and Lu (2019) by 0.11 F_1 score.

On the CoNLL 2003 English dataset, our model performs worse than these baseline models, but, with ELMo, it outperforms Jie and Lu (2019) and

---

[3] `https://github.com/speedcell4/refiner`

[4] In this paper, we use "external knowledge" to denote any additional resources other than word embeddings and contextual word representations.

Model	EK	P	R	F_1
Chiu and Nichols (2016)	✓	86.04	86.53	86.28
Strubell et al. (2017)	-	-	-	86.84
Li et al. (2017)	✓	88.00	86.50	87.21
Ghaddar and Langlais (2018)	-	-	-	87.95
Fisher and Vlachos (2019)	-	-	-	87.59
Cui and Zhang (2019)	-	-	-	88.16
Yan et al. (2019)	-	-	-	88.43
Jie and Lu (2019)	✓	88.53	88.50	88.52
Our Method	-	88.71	88.60	**88.65**
Yan et al. (2019) `[E]`	-	-	-	89.78
Jie and Lu (2019) `[E]`	✓	89.59	90.17	89.88
Our Method `[E]`	-	89.51	90.48	**89.99**
Devlin et al. (2019) `[B]`	-	90.01	88.35	89.16
Fisher and Vlachos (2019) `[B]`	-	-	-	89.71
Li et al. (2020b) `[B]`	-	92.98	89.95	91.11
Yu et al. (2020) `[B]`	-	91.1	91.5	**91.3**
Our Method `[B]`	-	90.00	91.17	90.93

Table 2: Experimental results on the OntoNotes 5.0 English dataset. Checkmark ✓ in the "EK" column indicates that external knowledge is utilized in that model. `[E]` and `[B]` stands for ELMo and BERT respectively. **Bold** and underlined numbers indicate the best and the second-best results respectively.

Model	EK	P	R	F_1
Huang et al. (2015)	✓	-	-	88.83
Lample et al. (2016)	-	-	-	90.94
Ma and Hovy (2016)	-	-	-	91.21
Zhang et al. (2018)	-	-	-	91.57
Chiu and Nichols (2016)	✓	-	-	91.62
Liu et al. (2019a)	-	-	-	91.80
Yan et al. (2019)	-	-	-	91.33
Liu et al. (2019b)	✓	-	-	**91.96**
Our Method	-	90.70	90.81	90.76
Jie and Lu (2019) `[E]`	✓	-	-	92.40
Yan et al. (2019) `[E]`	-	-	-	92.62
Our Method `[E]`	-	92.60	93.19	**92.89**
Devlin et al. (2019) `[B]`	-	-	-	92.8
Li et al. (2020b) `[B]`	-	92.33	94.61	93.04
Yu et al. (2020) `[B]`	-	93.7	93.3	**93.5**
Our Method `[B]`	-	92.66	92.98	93.23

Table 3: Experimental results on the CoNLL 2003 English dataset.

Yan et al. (2019) by 0.49 and 0.27 F_1 score. Our hypothesis is that the CoNLL 2003 dataset contains much fewer examples and entity categories, thus the label dependency relations are not as important as on the OntoNotes 5.0 English dataset, thus our method could bring about limited improvement.

A similar phenomenon can be noticed on the OntoNotes 4.0 Chinese dataset, as in Table 4, our model is inferior to Li et al. (2020a), but on the contextual word representations experiment setting, our model significantly outperforms them by 1.41 F_1 score with BERT. Moreover, on the OntoNotes 5.0 Chinese dataset, our model constantly outperforms the best previous work (Jie and Lu, 2019) by 0.65 F_1 score without utilizing external knowledge.

Besides, we can notice initializing character embeddings with our trick remarkably improves model performance by 0.76 F_1 score on the OntoNotes 4.0 Chinese dataset, even this improvement reduces to only 0.00 and 0.20 F_1 scores on ELMo and BERT experiments. We hypothesize that contextual word representation already provides rich enough morphological information, thus careful character embeddings initialization can only bring little benefit. On the OntoNotes 5.0 Chi-

Model	EK	P	R	F_1
Zhang and Yang (2018)	✓	76.35	71.56	73.88
Mengge et al. (2019)	✓	76.78	72.54	74.60
Gui et al. (2019a)	✓	76.40	72.60	74.45
Gui et al. (2019b)	✓	76.13	73.68	74.89
Yan et al. (2019)	✓	-	-	72.43
Li et al. (2020a)	✓	-	-	**76.45**
Our Method	-	75.28	72.39	73.80
Our Method (init)	-	75.49	73.69	74.56
Our Method `[E]`	-	80.21	78.50	79.34
Our Method (init) `[E]`	-	79.75	78.94	**79.34**
Devlin et al. (2019) `[B]`	-	78.01	80.35	79.16
Zhang and Yang (2018) `[B]`	✓	79.79	79.41	79.60
Gui et al. (2019a) `[B]`	✓	79.41	80.32	79.86
Mengge et al. (2019) `[B]`	✓	79.62	81.82	80.60
Li et al. (2020a) `[B]`	✓	-	-	81.82
Li et al. (2020b) `[B]`	-	82.98	81.25	82.11
Our Method `[B]`	-	81.80	84.31	83.03
Our Method (init) `[B]`	-	81.73	84.79	**83.23**

Table 4: Experimental results on the OntoNotes 4.0 Chinese dataset. "init" stands for utilizing projected FastText embeddings to initialize the character embeddings.

Model	EK	P	R	F_1
Pradhan et al. (2013)	-	78.20	66.45	71.85
Zhang and Yang (2018)	✓	76.34	77.01	76.67
Jie and Lu (2019)	✓	77.40	77.41	77.40
Our Method	-	77.09	77.50	77.29
Our Method (init)	-	77.99	78.11	**78.05**
Jie and Lu (2019) `[E]`	✓	78.86	81.00	79.92
Our Method `[E]`	-	79.75	79.83	79.78
Our Method (init) `[E]`	-	79.49	80.32	**79.92**
Our Method `[B]`	-	79.61	82.47	81.01
Our Method (init) `[B]`	-	79.66	82.45	**81.03**

Table 5: Experimental results on the OntoNotes 5.0 Chinese dataset.

Model	EK	WSJ	UD v2.2
Huang et al. (2015)	✓	97.55	-
Ma and Hovy (2016)	-	97.55	-
Zhang et al. (2018)	-	97.55	-
Yasunaga et al. (2018)	-	97.58	95.41
Xin et al. (2018)	-	97.58	-
Cui and Zhang (2019)	-	97.58	95.59
Our Method	-	**97.64**	**95.62**
Our Method `[E]`	-	**97.83**	**96.86**
Our Method `[B]`	-	**97.85**	**97.15**

Table 6: Experimental results on the WSJ and the UD v2.2 datasets.

nese dataset, the performance improvements are 0.76, 0.16, and 0.02 F_1 scores, respectively.

4.4.2 Part-of-speech Tagging

Table 6 shows the experimental results on the Wall Street Journal English and the Universal Dependencies v2.2 English dataset respectively. Although Cui and Zhang (2019) claimed that the simple Markov label transition model of CRF can barely bring information gain over bidirectional LSTM, we observe 0.06 and 0.03 gain in accuracy scores. Besides, our model achieves 97.83 and 96.86 accuracy scores with ELMo, and further improves the performance to 97.85 and 97.15 accuracy scores with BERT.

4.5 Discussion

Influence of Weight Tying The major difference between our method and Kim et al. (2017) is that we use only observed labels, while they employ unobserved labels as latent variables. In the actual implementation, this difference is reflected in whether to share label embeddings and the transition matrix of CRF across layers. Intuitively, completely relying on unobserved variables would implicitly performing clustering on latent representation space, and it might introduce noise. Besides, the state transitions in a different layer may obey different dynamics. Thus sharing the transition matrix across layers might have an impact on performance.

We conducted experiments on the OntoNotes 5.0 English dataset to compare the performance of all the above-mentioned settings, as reported in Table 7. Notably, our model, with both the label embeddings $\{\boldsymbol{v}_{y_j}\}_{j=1}^{m}$ and the transition matrix $\mathbf{A}$ shared, surpasses all separated models. These results support our claim that tying the weights of embeddings and the label transition matrix can indeed leverage annotation information and thus is better than completely relying on unobserved variables. Besides, we did not notice significant performance changes when varying the number of labels $|\mathcal{Y}|$. Furthermore, the number of parameters of our shared model is, in fact, the smallest one, even compared to LAN (about 10.0 million parameters).

$\{\boldsymbol{y}_j\}_{j=1}^{m}$	$\lvert\mathcal{Y}\rvert$	$\mathbf{A}$	$\lvert\theta\rvert$	$\mathbf{F}_1$
separated	10	separated	8,212,505	88.51
separated	20	separated	8,218,825	88.55
separated	50	separated	8,238,985	88.62
separated	73	separated	8,255,660	88.48
separated	100	separated	8,276,585	88.58
shared	73	separated	8,211,860	88.41
shared	73	shared	8,206,385	**88.65**

Table 7: Influence of weight tying, where $\{\boldsymbol{y}_j\}_{j=1}^{m}$ stands for whether share label embeddings across layers, $|\mathcal{Y}|$ denotes the number of labels, $\mathbf{A}$ is the CRF transition matrix in Equation 6, and $|\theta|$ is the number of parameters.

Influence of the Connection Mechanism A minor difference between our method and Cui and Zhang (2019) is that we utilize residual connection (He et al., 2016) and layer normalization (Xu et al., 2019), as in Equation 11, while Cui and Zhang (2019) only use concatenation, as in Equation 4. Table 8 shows the comparison on the OntoNotes 5.0 English dataset, measuring the influence of these two connection mechanisms. We find that the residual connection works better than the concatenation connection, that might because the residual connection can make training more smoothly by preventing the chaotic loss surface (Li et al., 2018).

Connection	$\mathbf{F}_1$
Concatenation	88.54
Residual	**88.65**

Table 8: Influence of the connection mechanism.

Influence of Parameter Size As in Table 9, we did not observe performance increase along with the increasing of the number of refinement layers. Therefore, we claim that one refinement layer is enough for our model, while Cui and Zhang (2019) needs two refinement layers. Our hypothesis is that the long-term dependency modeling capacity of the

L	d_h	$\|\theta\|$	$\mathbf{F}_1$
1	400	6,227,985	88.28
	600	8,206,385	**88.65**
	800	10,904,785	88.64
2	400	7,352,385	88.51
	600	10,732,985	88.36
	800	15,393,585	88.49

Table 9: Influence of parameter size, where L is the number of refinement layers, d_h is the hidden state dimension of the bidirectional LSTM, and $|\theta|$ represents the number of parameters.

first-order CRF is relatively limited, and we remain the use of the higher-order CRF as future work.

Training and Decoding Speeds We report the training and decoding speeds on the OntoNotes 5.0 English dataset. We demonstrate the efficiency of our CRF implementation tricks by comparing it with a widely used library, `pytorch-crf`[5]. According to Table 10, our CRF implementation tricks remarkably accelerate both training and decoding. In particular, with our CRF implementation, our computation extensive model even achieves a greater training speed than BiLSTM-CRF with `pytorch-crf`. Therefore, we claim that the efficiency of our model is acceptable.

Model	CRF	Training	Decoding
BiLSTM-CRF	`pytorch-crf`	82.25	480.08
	ours	**205.96**	**850.08**
Our Model	`pytorch-crf`	45.06	219.48
	ours	**93.04**	**296.91**

Table 10: Training and decoding speeds on the OntoNotes 5.0 English dataset. The "training" and "decoding" columns indicate the numbers of sentences our model can process per second on average.

5 Related Work

Early-stage research of NER and POS tagging focused on introducing rich features, for example, Yang et al. (2016) conducted experiments on the influence of discrete manual features, Chiu and Nichols (2016); Ma and Hovy (2016) introduced morphological features by employing a convolution network to encode character-level features, while Lample et al. (2016) chose bidirectional LSTM.

Some other research aimed at leveraging syntactic information, Li et al. (2017) and Jie and Lu (2019) proposed to run external parsers first and directly encode this syntactic information. Other work attempted to infer dependency relations among words implicitly, such that, Strubell et al. (2017) introduced iterative dilated convolution networks as an alternative to BiLSTM, and Zhang et al. (2018) and Liu et al. (2019b) designed encoders which maintain and update global representations along with local token representations.

Recently, Li et al. (2020b) unified flat and nested NER by formulating them as a machine reading comprehension task. Yu et al. (2020) proposed to enumerate all possible spans and to utilize a biaffine classifier to assign category labels to them.

Besides, the widespread use of contextual word representations, e.g., ELMo (Peters et al., 2018), Flair (Akbik et al., 2018), and BERT (Devlin et al., 2019), greatly improves the performance of NER models and they are accepted as new fundamental techniques of natural language processing.

Intuitively speaking, the refinement mechanism provides the models with additional chances to revise previous decisions. In existing work, this method was successfully applied to various tasks, e.g., text classification (Yu et al., 2017), sequential labeling (Cui and Zhang, 2019; Lyu et al., 2019), machine translation (Lee et al., 2018), and question answering (Nema et al., 2019). Our work is not the first attempt of introducing refinement mechanism to sequential labeling tasks. Cui and Zhang (2019) relied on locally normalized attention to softly refine hidden representations layer by layer, while Liu et al. (2019a) chose to discretely filter out target-irrelevant semantic aspects and thus could be considered as a hard refinement mechanism.

6 Conclusion

Motivated by the structured attention, we enhanced the previous refinement mechanism by replacing the locally normalized attention with our globally normalized attention. Experimental results on various tasks and datasets demonstrate that structured refinement is capable of filtering out target-irrelevant information through capturing long-term dependency relations. Besides, we remarkably accelerated training and decoding through two implementation tricks for CRF, and obtained further model performance improvement with an initialization trick for Chinese character embeddings. We

---

[5] `https://github.com/kmkurn/pytorch-crf`

remain to employ the higher-order CRF as future work.

Acknowledgements

This work was partly supported by JST CREST Grant Number JPMJCR1513. The authors would like to thank the anonymous reviewers for their instructive comments.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Hui Chen, Zijia Lin, Guiguang Ding, Jianguang Lou, Yusen Zhang, and Borje Karlsson. 2019. GRN: Gated relation network to enhance convolutional neural network for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6236–6243.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Leyang Cui and Yue Zhang. 2019. Hierarchically-refined label attention network for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4115–4128, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5840–5850, Florence, Italy. Association for Computational Linguistics.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019a. Cnn-based chinese ner with lexicon rethinking. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4982–4988. International Joint Conferences on Artificial Intelligence Organization.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuanjing Huang. 2019b. A lexicon-based graph neural network for Chinese NER. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1040–1050, Hong Kong, China. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Benjamin Heinzerling and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, Florence, Italy. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Zhanming Jie and Wei Lu. 2019. Dependency-guided LSTM-CRF for named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3862–3872, Hong Kong, China. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. *ArXiv*, abs/1702.00887.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc.

Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. 2017. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2664–2669, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. FLAT: Chinese NER using flat-lattice transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6836–6842, Online. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Kun Liu, Shen Li, Daqi Zheng, Zhengdong Lu, Sheng Gao, and Si Li. 2019a. A prism module for semantic disentanglement in name entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5358–5362, Florence, Italy. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019b. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Chunchuan Lyu, Shay B. Cohen, and Ivan Titov. 2019. Semantic role labeling with iterative structure refinement. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1071–1082, Hong Kong, China. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Xue Mengge, Yu Bowen, Liu Tingwen, Wang Bin, Meng Erli, and Li Quangang. 2019. Porous lattice-based transformer encoder for chinese ner.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Preksha Nema, Akash Kumar Mohankumar, Mitesh M. Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2019. Let's ask again: Refine network for automatic question generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3314–3323, Hong Kong, China. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference* *on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Alexander Rush. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark. Association for Computational Linguistics.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

R Weischedel, M Palmer, M Marcus, E Hovy, S Pradhan, L Ramshaw, N Xue, A Taylor, J Kaufman, M Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. linguistic data consortium, philadelphia, pa(2013).

Yingwei Xin, Ethan Hart, Vibhuti Mahajan, and Jean-David Ruvini. 2018. Learning better internal structure of words for sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2584–2593, Brussels, Belgium. Association for Computational Linguistics.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and improving layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4381–4391. Curran Associates, Inc.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. Tener: Adapting transformer encoder for named entity recognition.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 140–154. Springer.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 976–986, New Orleans, Louisiana. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539, Copenhagen, Denmark. Association for Computational Linguistics.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia. Association for Computational Linguistics.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564, Melbourne, Australia. Association for Computational Linguistics.

End-to-End NLP Knowledge Graph Construction

Ishani Mondal[*]
Microsoft Research
Bengaluru, India
`ishani340@gmail.com`

Yufang Hou
IBM Research Europe
Dublin, Ireland
`yhou@ie.ibm.com`

Charles Jochim
IBM Research Europe
Dublin, Ireland

Abstract

This paper studies the end-to-end construction of an NLP Knowledge Graph (KG) from scientific papers. We focus on extracting four types of relations: ***evaluatedOn*** between *tasks* and *datasets*, ***evaluatedBy*** between *tasks* and *evaluation metrics*, as well as ***coreferent*** and ***related*** relations between the same type of entities. For instance, "*F1 score*" is coreferent with "*F-measure*". We introduce novel methods for each of these relation types and apply our final framework (*SciNLP-KG*) to 30,000 NLP papers from ACL Anthology to build a large-scale KG, which can facilitate automatically constructing scientific leaderboards for the NLP community. The results of our experiments indicate that the resulting KG contains high-quality information.

1 Introduction

As interest in the NLP research community grows, the number of NLP tasks, datasets, and metrics for evaluation also grows, making it increasingly difficult for researchers to keep track of the plethora of new resources. In order to tackle this problem, recently there have been a few manual efforts to summarize the state-of-the-art on selected subfields of NLP in the form of leaderboards that extract tasks, datasets, metrics and results from papers, such as *NLP-progress*[1] or *Paperswithcode*[2]. However, these manual efforts are not sustainable over time for all NLP tasks.

Meanwhile, although there are various studies focusing on extracting entities and relations from scientific literature (Augenstein and Søgaard, 2017; Luan et al., 2018; Gábor et al., 2018; Hou et al., 2019; Jain et al., 2020), there is less work on constructing KGs that contain rich information about tasks, datasets, and metrics. Such a KG would be highly beneficial for the researchers to understand the underlying related literature for a particular task, or to perform comparable experiments.

In this paper, we propose an end-to-end approach to construct a KG from the NLP papers. Our KG contains three types of entities (***tasks***, ***datasets***, and ***metrics***) and four types of relations connecting them. Figure 1 (bottom left) depicts these relations in our large-scale graph built from 30,000 NLP papers from the ACL Anthology. For instance, "sentiment analysis (task)" is ***evaluatedOn*** "IMDb dataset", "opinion analysis (task)" is ***evaluatedBy*** "Precision (metric)", "F1-score (metric)" is ***coreferent*** with "F1-measure (metric)", and "YELP review dataset" is ***related*** to "IMDb dataset".

We develop a framework (*SciNLP-KG*, Section 5) to extract these relations based on NLP papers that are tagged for Task (T), Dataset (D), and Metric (M) entities using a TDM entity tagger. Our framework primarily consists of three learning-based models. First, motivated by Hou et al.'s work on tagging NLP papers with valid TDM triples based on a small manually created gold TDM taxonomy, we design a hybrid NLI (Natural Language Inference)-based relation extraction model to extract *evaluatedOn* and *evaluatedBy* relations. Our model can extract these two relations at the document level even if the related entities do not appear in the same sentence. Second, for the *coreferent* relation, we use a mention-pair model to identify the same entities within and across documents. We use a few heuristics to generate training instances, such as that authors often use abbreviations to refer to the common terms (i.e., *NER – Named Entity Recognition*). Third, we propose another model, *term2vec*, which is trained on pseudo-sentences that contain tagged TDM entities from different documents. We use the resulting embeddings to extract the *related* relation between similar type of

---

[*] Work done during internship at IBM Research.

[1] https://github.com/sebastianruder/NLP-progress

[2] https://paperswithcode.com

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1885–1895
August 1–6, 2021.

entities. For instance, "*semantic role labeling*" is related to "*argument identification*" and "*GENIA Corpus*" is related to "*NCBI Corpus*".

To evaluate our end-to-end *SciNLP-KG* framework, we manually construct a small-scale NLP KG based on our proposed schema (Section 4.2), which contains 85 nodes and 625 links. Experiments show that our system achieves reasonable results for all relation types on this small-scale graph with all possible meaningful links manually annotated. We further apply our framework on 30,000 NLP papers from ACL Anthology to build a large-scale NLP KG containing 5,374 nodes and 15,762 relations. We evaluate the quality and coverage of the KG by manually assessing random samples and comparing it with *Paperswithcode*. We found that our KG contains high-quality information.

Overall, the contributions of our work are threefold. First, we propose and design a new schema that represents knowledge about *tasks* (T), *datasets* (D) and *metrics* (M) in the NLP domain. Second, we develop a novel framework (*SciNLP-KG*) for constructing an NLP KG from the scientific literature in an end-to-end manner. Finally, we *automatically* build a large-scale NLP KG that contains high-quality information about the Task-Dataset-Metric (TDM) entities. However, our method is generalized in a way that it could be extended to the domains of computer vision or bioinformatics. Our code and datasets are made publicly available at `https://github.com/Ishani-Mondal/SciKG` to fuel further research.

2 Related Work

There is a wealth of research in the NLP community on extracting information from scientific literature. Earlier work identified citation contexts and then classified them (Teufel et al., 2006; Abu-Jbara et al., 2013; Jurgens et al., 2018). Other lines of research include unsupervised approaches for extracting paper concepts (Gupta and Manning, 2011; Tsai et al., 2013) and keyphrases (Kim et al., 2010; Hasan and Ng, 2014). Here we are most interested in entity and relation extraction in scientific papers (Tateisi et al., 2014; Augenstein and Søgaard, 2017; Gábor et al., 2018). In SemEval 2017 Task 10, Augenstein et al. (2017) focused on extracting three types of entities (*Process, Task, and Method*) and two relation types (*hyponym-of* and *synonym-of*). The dataset from this task has been used to explore various neural models for IE on scientific literature (Luan et al., 2017; Ammar et al., 2017). Luan et al. (2018) released another dataset which contains annotations for five types of entities and eight types of relations on 500 scientific abstracts. Ammar et al. (2018) built a literature graph for different domains and they focused on identifying entities and linking them to the existing knowledge bases. Hou et al. (2019) developed a system to identify Task-Dataset-Metric triples in NLP papers based on a small set of manually constructed gold TDM triples. Recently, Jain et al. (2020) introduced *SciREX*, a document level IE dataset that encompasses multiple IE tasks, including salient entity identification and document level N-ary relation identification from scientific articles. Unlike the above mentioned work, we concentrate on building a TDM KG by leveraging entity recognition and relation extraction within and across different sentences/documents. Our constructed KG can facilitate large-scale NLP leaderboard construction and help the researchers to gain insights from NLP literature.

3 NLP Knowledge Graph Schema

In this section, we propose an NLP KG schema which contains three types of entities (i.e., *task*, *dataset*, and *metric*) and four types of relations between them. An entity mention is a single or multi-word nominal phrase that represents a task (e.g., *named entity recognition*), dataset (e.g., *IMDB*), or evaluation metric (e.g., *F1-score*) entity. We mainly focus on these three types of entities because they are the core concepts of the NLP community. In addition, they are relatively stable across different papers compared to other types of entities, such as method, result, or experiment.[3] Similarly, we focus on the following four types of relations between TDM entities that represent the collectively shared view among the NLP researchers:

1. evaluatedOn: This relation implies that a task T is often evaluated on a dataset D (e.g., *sentiment analysis* $\rightarrow$ *IMDB*).

2. evaluatedBy: This relation implies that NLP

---

[3] By referring to the stability of the entities, we meant to say that TDM entities are more likely to reflect the collectively shared views in the NLP domain, i.e., researchers will use the same name to refer to the same entity in different papers (e.g., sentiment classification). In contrast, method/result/experiment in research papers are non-standardized in nature and very difficult to normalize. For instance, a main method "LSTM-CNN-sentiment" in one paper might be referred to in another paper as "LC-based text classification".

researchers usually evaluate a task T using metric M (e.g., *Named Entity Recognition* $\rightarrow$ *F1*).

3. coreferent: This relation is used to capture the fact that an entity may be referred to differently in the same or different papers, such as *Named Entity Recognition – NER*, *NCBI dataset – NCBI corpus*, or *F1 score – F1 measure*. The *coreferent* relation can help us to canonicalize TDM entities in the constructed KG.

4. related: Similar to word relatedness, this relation captures all types of associations between the same type of entities.[4] For instance, "*semantic role labelling*" is related to *argument identification* because the latter is a sub-task of the former. Also "*GENIA Corpus*" is related to "*NCBI Corpus*" because both datasets are used to develop NER models in the biomedical domain. In practice, there is not a clear way to define all possible relations between TDM entities, and the *related* relation provides a practical and efficient way to navigate KG.

4 Dataset Construction

We create two new datasets for testing NLP KG construction, both of which are derived from the TDM corpus from Hou et al. (2021) (see Section 4.1). The first dataset is a manually constructed small-scale NLP KG according to the schema described in Section 3. The second dataset is constructed for training a model to extract *evaluatedOn* and *evaluatedBy* relations.

4.1 TDM Tagged Corpus

Our target entities are abstract objects of type **Task, Metric, and Dataset (TDM)** which are specific instantiations of the entities in document. We use the recently released state-of-the-art TDM tagger (Hou et al., 2021), trained on the Flair framework (Akbik et al., 2019) based on the cased BERT-base embeddings (Devlin et al., 2019), to obtain the mentions of Task, Dataset and Metrics. This tagger is trained on a corpus of 1,500 sentences taken from the full text of NLP papers, which have been annotated by domain experts for TDM concepts. Finally, it has been applied on 30,000 NLP papers from the ACL Anthology. We refer to the resulting dataset as *TDM-NLP-Papers* for the rest of the paper and it will be used as input for our proposed framework to construct an NLP KG.

Nodes		Relations		
Type	**Count**	**Type**	**Count**	κ
Task	43	evaluatedOn	81	0.78
Dataset	23	evaluatedBy	249	0.73
Metric	19	coreferent	38	0.84
		related	257	0.94
Total	85	**Total**	625	

Table 1: Statistics of *smallNLP-KG*. κ indicates inter-annotator agreement for each relation type.

4.2 smallNLP-KG

In order to evaluate our approach to construct an NLP KG more efficiently during model development, we have conducted an annotation study to build a small-scale gold NLP KG following the schema given in Section 3.

Specifically, the first author sampled a small amount of TDM entities from the tagged dataset *TDM-NLP-Papers*. The chosen entities contain well-established tasks (e.g., *dependency parsing* or *named entity recognition*) and their corresponding datasets as well as evaluation metrics. In order to better facilitate the type of *coreferent* edge in the KG, entities that are abbreviated or have a small edit distance to the existing entities are also added to the list (e.g., *NER*).

Two domain experts then independently annotate all possible relations described in Section 3 between any two entities. If necessary, they read the corresponding literature to make a decision. Finally, the annotators reconcile the differences in their annotations and produce the final canonical annotation. The final gold graph contains 85 entities (nodes) and 625 relation instances. Table 1 lists statistics for each entity type and each relation type. We also calculate the inter-annotator agreement per relation type using Cohen's κ (see κ column in Table 1). Overall, we achieve high inter-annotator agreement for all the relation types.

Apart from these relations, we also explored other relations between similar types of entities, such as *hypernym* (Hearst, 1992) and *part-of* relations. In a pilot annotation study, we found that finer-grained relation types between TDM entities are more difficult to annotate, e.g., the relation between "semantic role labelling" and "argument identification". Ultimately, in order to help users to navigate through the built KG, we decide to use the

---

[4] In principle, the *related* relation can be applied to different type of TDM entities. Here we only consider it between the same type of entities because the *evaluatedOn* and *evaluatedBy* relations already capture the most prominent relations across TDM entities.

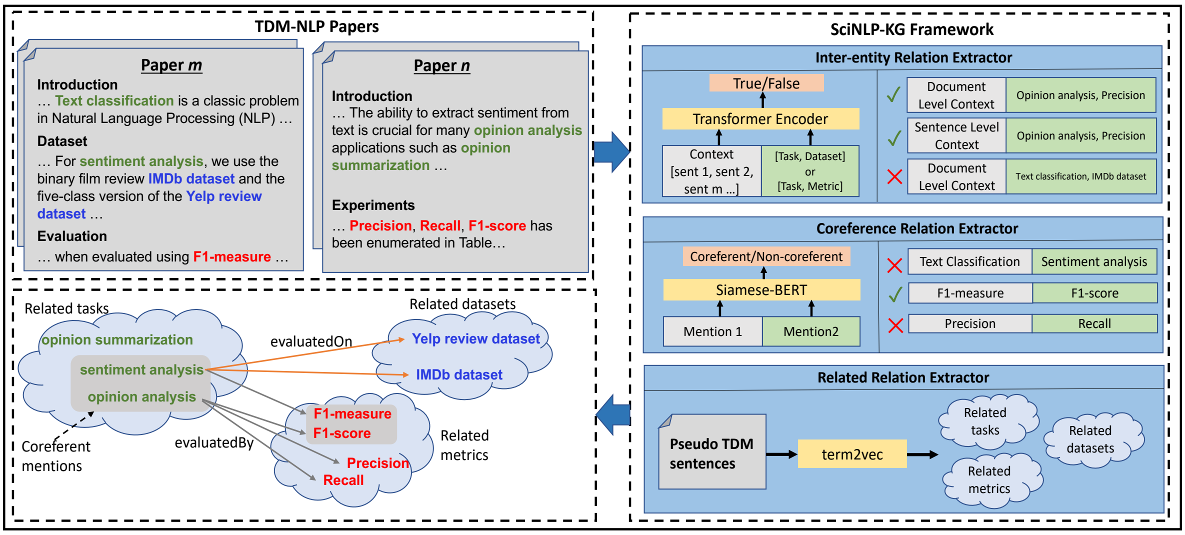

Figure 1: Overview of the SciNLP-KG Framework

"related" relation to capture all types of associated relations between the same type of entities (e.g., when a user clicks "coreference resolution", the system can recommend related tasks).

4.3 TD/TM Training Dataset

To facilitate the extraction of the *evaluatedBy* and *evaluatedOn* relations, we create a corpus (*TD-TM-Rel*) containing 600 sentences randomly chosen from the tagged *TDM-NLP-Papers* corpus. Each sentence has at least two different types of tagged TDM entities. Two domain experts then annotate the valid *evaluatedBy* and *evaluatedOn* relations for each sentence. The inter-annotator agreements on the *evaluatedOn* and *evaluatedBy* relations are 0.96 and 0.91 (Cohen's κ), respectively. Below is an example of TD/TM entities appearing in the same sentence but not expressing *evalutedOn/evaluatedBy* relations: *"As a testbed for this task, we introduce the **SentiHood** dataset extracted from a **question answering** platform where urban neighbourhoods are discussed by users"*. This sentence does not express a valid *evalutedOn* relation between the "question answering" task and the "SentiHood" dataset.

5 SciNLP-KG Framework

In this section, we describe our proposed framework, *SciNLP-KG*, to construct an NLP KG from unstructured text as shown in Figure 1. Our framework consists of three models to extract four types of relations as described in Section 3. Instead of proposing an end-to-end model (which could suffer from component-wise propagation errors (Wadden et al., 2019; Jain et al., 2020)), we incrementally develop separate components based on the properties of the target relations and the availability of training datasets, and finally aggregate results of each component to create the final KG.

5.1 Inter-Entity Relation Extractor

The *evaluatedOn* and *evaluatedBy* relations between task and dataset/metric entities depend on the document-level context. It is often the case that the entities involved in these two relations do not occur in the same sentence. On the other hand, a document containing a task mention t and a dataset mention d does not necessarily imply that there exists a positive *evaluatedOn* relation between them.

Sentence-Level NLI Model (S-NLI). We train a S-NLI model based on our corpus *TD-TM-Rel* (Section 4.3). Here *context* is a sentence, and *TD/TM hypothesis* is one of the combinations of TD/TM entities tagged in the sentence. During training, *[CLS]context[SEP]TD/TM hypothesis[SEP]* is given as input to the BERT-based model, followed by sigmoid activation on [CLS] token to predict if the context expresses a valid *evaluatedOn/evaluatedBy* relation or not. The model is trained using binary cross-entropy loss. Note that among all combinations, only the ones that correspond to the annotated *evaluatedOn/evaluatedBy* relations are valid TD/TM hypotheses. At inference time, given a task t and a dataset d (or metric m), we first extract all sentences from the tagged corpus, *TDM-NLP-Papers*, which contain both t

and d (or m). We then try to predict whether an *evaluatedOn* (or *evaluatedBy*) relation can be inferred from any of these sentences using the trained model.

Document-Level NLI model (D-NLI). Instead of looking only at the sentence level, we also adopt an approach in which we form the *TD/TM hypothesis* space at the document level. For example, if a tagged NLP paper has n task entities (t_1, t_2,.... t_n) and m dataset entities (d_1, d_2,.... d_m), we generate $n * m$ combinations of TD tuples as our testing TD hypotheses. For each testing TD hypothesis, we construct *context* by concatenating sentences that contain at least one corresponding entity from the tagged NLP paper. Different from the S-NLI model, the D-NLI model can capture the *evaluatedOn/evaluatedBy* relations across sentence boundaries. We use the recently released *SciREX* corpus (Jain et al., 2020) to train our D-NLI model. *SciREX* contains document-level annotations of $<$ *Task, Dataset, Metric, Method*$>$ tuples as well as the individual entities for 438 NLP papers. Most binary relations such as *Task–Dataset* (*evaluatedOn*) and *Task–Metric* (*evaluatedBy*) occur across sentences. Specifically, for each valid TD/TM relation annotated at the document level, we treat them as a valid TD/TM hypothesis. We form the *context* by concatenating all sentences from the paper that contain the annotations for at least one element in this relation. Similarly, we construct negative training instances using other TD/TM combinations by assuming that 4-ary tuple annotations in *SciREX* contain all valid TD/TM pairs for a specific paper.

Hybrid NLI Model (H-NLI). Although the D-NLI model can capture *evaluatedOn/evaluatedBy* relations across sentences, we assume that the S-NLI model is more accurate because it is easier for the model to learn patterns from shorter contexts. To combine the strengths of both models, we propose a hybrid NLI model. More specifically, given a task t and dataset d (or metric m) and the corresponding NLP papers containing these entities, we first apply our S-NLI model to all sentences containing both entities to decide whether an *evaluatedOn* (or *evalautedBy*) relation is held between them. If such a sentence does not exist, we use D-NLI model to make the final prediction. By doing so, we cascade the outputs from both S-NLI and D-NLI models in a sieve-based fashion, with the higher priority sieve being the S-NLI model. Thus, it combines the strength of capturing higher precision with S-NLI and better recall with D-NLI.

5.2 Coreferent Relation Extractor

Unlike coreference resolution on news corpora, which mainly depends on context, we notice that in our scenario, researchers often use abbreviations to refer to common terms (e.g., *NER–Named Entity Recognition*). Sometimes different researchers use slightly different variations to refer to the same entity (e.g., *F1 score–F1*). Motivated by these observations that the surface forms play a pivotal role for TDM coreferent relation extraction, we design a mention-pair model to capture the coreferent relations between the same type of entities.

We generate positive training instances for our mention-pair model using a few heuristics. Specifically, we apply a regular pattern to check whether a tagged entity is followed by its abbreviation in brackets in the tagged NLP papers, such as "*Named entity recognition (NER)*". We further extract pairs of entities which have a small edit distance, such as "*F1 score – F1*". Finally, we generate the same number of negative training instances by randomly pairing entities of the same type, which do not meet the criteria of the above heuristics.

We fine-tune our mention-pair model on a *BERT-Siamese Network* (Reimers and Gurevych, 2019). We use mean pooling over the output of two [CLS] tokens and use the Euclidean distance function in the penultimate layer, which is followed by a fully connected softmax layer with two labels (*coreferent* and *non-coreferent*). We form mention-pairs from within and across documents, so this component can be seen as a cross-document coreference resolution module.

5.3 Related Relation Extractor

We observe that in our *TDM-NLP-Papers* corpus, the co-occurrence patterns of TDM entities in different contexts (documents) encode rich semantic information about each individual entity. Motivated by *word2vec* (Mikolov et al., 2013), we propose an unsupervised *term2vec* model to capture related relations between TDM entities.

We hypothesize that the TDM entities in a single document are somehow related, even if they do not occur in the same sentence. We then generate a pseudo-sentence for each tagged paper. A pseudo-sentence per paper treats the whole paper as the context which contains special "word"–TDM entities. Two such examples are: *"sent0: sentiment analysis, aspect-based sentiment analysis, semeval 2014 task 4 laptop, sentihood, text classification"*, and *"sent1: sentiment classification, semeval 2014 task 4 laptop, sentihood"*. After applying *term2vec* on these pseudo sentences, terms with similar contexts will be close to each other in the embedding space, which can help us to identify related relations between TDM entities.

Specifically, for each paper in *TDM-NLP-Papers*, we concatenate all tagged entities to form a pseudo sentence. We treat each entity (term) as a single word in this sentence and generate vector representations for each term using the Skip-gram model, which preserves the type of each term.

For a term j, the algorithm models the neighborhood of this term as shown in equation (1):

$$\arg\max_{\theta} \sum_{j \in V} \sum_{i \in K_V} \sum_{c_i \in N_i(j)} \log p(c_i | j; \theta) \qquad (1)$$

Here, K_V denotes the type of nodes. $N_i(j)$ denotes the neighborhood of the term j with respect to the i^{th} type of terms and $p(c_i|j;\theta)$ represents the conditional probability of having a context term c of type i given the term j. The objective of this algorithm is to predict the neighborhood of a given term of a particular type i to other terms of similar type. After obtaining the d-sized embeddings for each term, we use the unsupervised K-means clustering algorithm to determine the clusters for each term. These clusters, thus generated, are among the same type of entities and encode the *related* relations among these entities.

6 Experimental Setup on smallNLP-KG

We first evaluate our framework on *smallNLP-KG*. We report precision, recall, and F-score fore each relation type. Note that for evaluating *coreferent* and *related* clusters, we consider all pairs of entities in a cluster to be linked.

Dev/test split. The nodes and relations in *smallNLP-KG* are carefully divided in 10-90% dev-test split. To avoid any leakage problem as observed in KG completion tasks (Sun et al., 2020; Pezeshkpour et al., 2020), we exclude training instances from the NLI models and the mention-pair model that involve any of the testing entities. More specifically, first, we went through the whole *smallNLP-KG* to make sure any TDM entities (including their coreferent mentions) in the test data do not appear in the dev set. Second, we exclude all these entities (including their coreferent mentions) from the training data for each relation extraction component. For the rule-based and unsupervised baselines of *coreferent* and *related* relations (Section 7.3 and Section 7.4), we tune our parameters on the dev set and report results for all relations on the test set.

Implementation details. Given all testing entities from *smallNLP-KG* as the input, we use our framework presented in the previous section to build the KG.

For the hybrid NLI model to extract inter-entity relations (Section 5.1), we fine-tune all NLI models for 3 epochs with a learning rate of $5e-5$ and a batch size of 32. We also carry out experiments with different pre-trained contextual embeddings: *BERT-Base*, *BERT-Large*, *RoBERTa-Base*, as well as *scibert-scivocab-cased* (Beltagy et al., 2019) from the PyTorch-Transformers library. During the inference stage, let there be n_t tasks, n_d datasets and n_m metrics in the *smallNLP-KG* test corpus, so we can have total $(n_t \times n_d)$ and $(n_t \times n_m)$ combinations of possible *evaluatedOn* and *evaluatedBy* relations respectively. We apply our trained hybrid NLI model to test all combinations.

For the *term2vec* model to extract related relations (Section 5.3), we use the Skip-gram word2vec implementation from Gensim with a window size of 5, min count of 1, and an embedding dimension of 100. We run the model on all pseudo sentences generated from the whole *TDM-NLP-Papers* corpus. After obtaining term embeddings, we use K-Means clustering algorithm from Scikit-learn to generate clusters among the same type of entities based on nodes from *smallNLP-KG*. We set K equal to number of gold clusters in *smallNLP-KG* for entity type.

7 Results on smallNLP-KG

7.1 Comparison With the Existing Baselines

We compare our *SciNLP-KG* framework against a few baselines from previous work:

1. E2E Rel-Gold (Miwa and Bansal, 2016). The TDM entities present in *TDM-NLP-Papers* are being fed as input to this LSTM-based model, which uses word sequences and dependency tree structures to predict relations. We train the model

	evaluatedOn			evaluatedBy		
Methods	P	R	$F1$	P	R	$F1$
Sent-Level						
-BERT-S	0.77	0.23	0.35	0.68	0.25	0.36
-BERT-L	0.80	0.23	0.36	**0.71**	0.27	0.39
-SciBERT	0.81	0.22	0.35	0.70	0.22	0.33
-RoBERTa	**0.83**	0.24	0.37	0.70	0.27	0.39
Doc-Level						
-BERT-S	0.51	0.80	0.62	0.44	0.68	0.53
-BERT-L	0.52	0.80	0.63	0.45	0.71	0.55
-SciBERT	0.57	0.84	0.68	0.43	**0.73**	0.54
-RoBERTa	0.62	**0.86**	0.73	0.45	0.72	0.55
Hybrid						
-BERT-S	0.60	0.78	0.68	0.50	0.64	0.56
-BERT-L	0.63	0.80	0.70	0.51	0.65	0.57
-SciBERT	0.64	0.81	0.72	0.48	0.67	0.56
-RoBERTa	0.67	0.82	**0.74**	0.59	0.69	**0.64**

Table 2: Ablation study of NLI Models for extracting *evaluatedOn*, *evaluatedBy* relations on *smallNLP-KG*.

using our TD/TM training dataset and perform intra-sentence *evaluatedOn/EvaluatedBy* relation extraction using this model.

2. E2E Rel (Miwa and Bansal, 2016). This is another variant of the previous model, which predicts the TD and TM relations in an end-to-end fashion without using the TDM entities tagged in *TDM-NLP-Papers*. We train this model to tag the TDM entities and predict the relations in an end-to-end fashion instead of providing the gold mentions as input to the relation extractor.

3. DocTAET (Hou et al., 2019). This model is analogous to our D-NLI model which works in document-level. We use the NLI model proposed in this paper for predicting TD and TM binary relations individually for comparable experimental evaluation, without considering them as a 3-ary relation between <Task, Dataset, Metric>).

4. E2E Coref (Lee et al., 2017). For coreference resolution, we consider the nouns as we consider only three types of nominal entities. We make use of the end-to-end BiLSTM-CRF based architecture using ELMO embeddings optimized using the conditional likelihood of predicting the correct antecedent given a mention. In our experiments, we used the noun phrase coreference resolution only and discarded the pronoun-based coreference resolution component.

5. SciIE-Rel (Luan et al., 2018). This is a multi-task model to extract coreferent and inter-entity relations from the scientific abstracts. We retrain this model on our training datasets and use it to predict *evaluatedOn/evaluatedBy* relations based on the predicted Task, Dataset, and Metric mentions from the *NLP-TDMS-Papers* corpus.

	evaluatedOn			evaluatedBy		
Methods	P	R	$F1$	P	R	$F1$
Baselines						
E2E Rel	0.58	0.62	0.60	0.45	0.54	0.49
E2E Rel-Gold	0.63	0.69	0.65	0.51	0.58	0.54
SciIE Rel	0.68	0.78	0.72	0.51	0.62	0.56
DOCTAET	0.62	**0.86**	0.73	0.45	**0.72**	0.55
S-NLI	**0.83**	0.24	0.37	**0.70**	0.27	0.39
D-NLI	0.62	**0.86**	0.73	0.45	**0.72**	0.55
H-NLI	0.67	0.82	**0.74**	0.59	0.69	**0.64**

Table 3: Results of *SciNLP-KG* for extracting *evaluatedOn* and *evaluatedBy* relations and comparison with the best-performing variant of existing baselines.

6. SciIE-Coref (Luan et al., 2018). This is a component of the above mentioned *SciIE-Rel* system. We use this component to extract the coreferent clusters of the same types of entities from the predicted Task, Dataset and Metric mentions of the *NLP-TDMS-Papers* corpus.

7.2 Results on Inter-Entity Relations

Table 3 shows the results of our hybrid NLI model together with the S-NLI and D-NLI models to extract *evaluatedOn* and *evaluatedBy* relations on the *smallNLP-KG* test set. The ablation analysis using different embeddings is tabulated in Table 2. It seems that the S-NLI model suffers from lower recalls, with 0.24 on the *evaluatedOn* relation and 0.27 on the *evaluatedBy* relation. Exploring the D-NLI context, we observe a drop of 0.21 and 0.25 in terms of precision on *evaluatedOn* and *evaluatedBy* relations respectively, which is mainly attributed to the relatively longer *context* in the NLI model. There is a significant rise in recall for both relations, with improvements of 0.86 and 0.72 on the *evaluatedOn* and *evaluatedBy* relations, respectively. Finally, our H-NLI model combines the strengths of both S-NLI and D-NLI models, with decent improvement in terms of precision compared to the D-NLI architecture, while preserving better recall than S-NLI model.

7.3 Results on *coreferent* Relations

We compare our mention-pair model (Section 5.2) to two coreference resolution systems (*E2E Coref* and *SciIE Coref* in Section 7.1). We also imple-

	coreferent			related		
Methods	P	R	$F1$	P	R	$F1$
Baselines						
Rule-based	0.43	0.58	0.49	-	-	-
E2E Coref	0.61	0.67	0.64	-	-	-
SciIE Coref	0.65	0.70	0.69	-	-	-
PMI	-	-	-	0.61	0.61	0.61
SciNLP-KG	**0.77**	**0.79**	**0.77**	**0.67**	**0.68**	**0.67**

Table 4: Results of *SciNLP-KG* for *coreferent* and *related* relation extraction compared to baselines.

ment a heuristic baseline that uses the Jaccard similarity of two entity mentions and predicts those with score greater than 0.75 as coreferent.

Table 4 (left) lists the results of our mention-pair model for *coreferent* relation extraction on the *smallNLP-KG* test set, with an overall Macro F1 of 0.77. Overall, we found that our system outperforms the other three baselines slightly. During qualitative analysis, we observe that our learning-based model eliminates some false positive links proposed by the rule-based approach, such as <*Rouge-1* **coreferent** *Rough-2*> and also captures more true positive links such as <*sentiment mining* **coreferent** *sentiment analysis*>.

7.4 Results on *related* Relations

We compare our *term2vec* model (Section 5.3) to a baseline model which use *pointwise mutual information* (PMI) (Church and Hanks, 1990) to measure the association score between two TDM entities of the same type.

Table 4 (right) lists the results of our *term2vec* model for extracting *related* relations on *smallNLP-KG*. We found that our *term2vec* model outperforms the PMI-based baseline on *task* and *metric* entities by a good margin. We also observed that there is a slight improvement on the dataset clusters. This happens because the datasets inside the same documents are related most of the time, whereas the same is not always true for tasks and metrics, in which the context window of their mentions plays an important role to capture whether they are *related*. During qualitative analysis, we observe some false positives generated from the PMI-based baseline such as <*f1-score* **related** *rouge*>. Interestingly, we find that our *term2vec* model captures both coreferent relations as well as other related relations between the same type of entities, such as *hypernym* relations (e.g, <*parsing* **related** *dependency parsing*>, <*rouge* **related** *rouge-n*>).

Nodes		Relations	
Type	Count	Type	Count
Task	2,441	evaluatedOn	4,137
Dataset	2,314	evaluatedBy	4,019
Metric	619	coreferent	2,074
		related	5,532
Total	5,374	**Total**	15,762

Table 5: Overall statistics of *largeNLP-KG*.

7.5 Comparison with the Existing Models

Our final Hybrid-NLI model performs better than the two recent relation extraction systems (*E2E Rel* and *SciIE Rel*), similarly our cross-document mention-pair coreference resolver outperforms two related baselines. Probing deeper, we observe that existing baselines struggle because: 1) they fail to resolve certain long-range relations due to the end-to-end setup and suffer from error propagation; and 2) they cannot handle inter-document coreference resolution, failing to generate some <Task, Dataset> pairs found when canonicalizing entities in cross-document discourse. Hou et al. (2019)'s method suffers from low precision due to the bottleneck of the D-NLI approach (see Table 3). Our hybrid NLI approach achieves better precision while still keeping a reasonable recall.

It is worth noting that in *smallNLP-KG* we annotated completed relations for every pair of TDM mentions. The annotations already take care of inference-based KG consistency (e.g., T1 *coreferent* T2, T1 *evaluatedOn* D1, T2 *evaluatedOn* D1). In general, our framework handles each of the relations separately and achieves good performance compared to the baselines which use joint modelling approaches on the *smallNLP-KG* testing set.

8 Experiments on LargeNLP-KG

We apply our trained *SciNLP-KG* framework to the *TDM-NLP-Papers* corpus (Section 4.1) to build a large-scale KG (*largeNLP-KG*). The resulting KG contains 5,374 TDM entities and 15,762 relations (see Table 5). We use both human evaluation as well as automatic evaluation to assess the quality and coverage of *largeNLP-KG*.

Human Evaluation. We randomly sample 100 instances from the final large-scale KG for *evaluatedOn*, *evaluatedBy*, and *coreferent* relations, which were then manually assessed by an NLP expert. Note that these relation instances do not appear in the training datasets of our *SciNLP-KG* sys-

Relations	#	$Prec$	$P@5$	$P@10$	$P@20$
evaluatedOn	100	0.84	-	-	-
evaluatedBy	100	0.77	-	-	-
coreferent	100	0.79	-	-	-
related	600	-	0.77	0.71	0.60

Table 6: Precision of the randomly selected samples from *largeNLP-KG*. # indicates sample size.

Method	evaluatedOn	evaluatedBy
Relaxed Match	0.49	0.58
+coreferent	0.56	0.63

Table 7: Coverage of *largeNLP-KG* compared with *Paperswithcode* on *evaluatedOn/evaluatedBy* relations.

tem. Table 6 reports the precision for each relation type. It is encouraging to see that our large-scale NLP KG obtains reasonably high precision (0.84, 0.77 and 0.79) on the *evaluatedOn*, *evaluatedBy* and *coreferent* relations, respectively. We found that most false positives are from the TDM tagger errors. For instance, *Stanford-CoreNLP* is tagged as a *dataset* entity.

In case of *related* relation clusters, we randomly sample 10 entities from each entity type (i.e., *Task, Dataset, Metric*) and choose the top 20 nearest neighbors of the same type based on the cosine-similarity between entities. An NLP expert assessed the correctness of these 600 relations (200 for each of the T-T, D-D, and M-M related relations) based on their common sense knowledge about NLP. We report $Macro - precision@K$, where K= 5, 10, 20 in Table 6 (last row). In general, for a given entity, our unsupervised *term2vec* model provides reasonable suggestions of the related entities.

Automatic Evaluation. In order to better understand the coverage of *largeNLP-KG*, we compared *evaluatedOn* and *evaluatedBy* relations from *largeNLP-KG* with the manually constructed NLP leaderboards in *Paperswithcode*.[5]

The recent version of *Paperswithcode* (Aug-2020) contains leaderboard information for 265 NLP tasks and the corresponding 100 datasets. We consider only the NLP-related TDM entities. Each leaderboard is a tuple of four elements (<*task, dataset, metric, score*>) and it encodes the valid relations between the task and the dataset/metric, which corresponds to our *evaluatedOn* and *evaluatedBy* relations. In total, we obtain 450 TD pairs and 623 TM pairs from *Paperswithcode*.

We automatically check how many of these pairs are encoded in our large-scale NLP KG. Note that we do not compare TD/TM pairs that appear in the training dataset for our NLI models (Section 5.1). Specifically, we use an edit-distance matching algorithm to match TD/TM pairs between *largeNLP-KG* and *Paperswithcode* (*Relaxed Match*). We consider the edit-distance of our extracted TD and TM pairs with those in *Paperswithcode* and choose those with normalized edit-distance less than 0.3 as positive instances. For instance, the *Paperswithcode* TM pair "*sentiment mining–F1-score*" is equivalent to extracted TM pair "*sentiment analysis–F1-scores*".

Table 7 shows that with *Relaxed Match*, our *largeNLP-KG* contains 49% and 58% of TD and TM pairs from *Paperswithcode*, respectively. We further employ coreferent relations to generate more *evaluatedOn* and *evaluatedBy* relations, which helps to achieve a higher coverage of 56% and 63% on *Paperswithcode* on TD and TM pairs respectively. The mismatch part between our large KG and *Paperswithcode* is mostly due to the fact that *Paperswithcode* contains recent papers while our KG is built on papers from 1974-2019. For instance, our graph contains a task called "textual entailment", which is not included as a task entity in *Paperswithcode*.

9 Conclusions

In this paper, we propose *SciNLP-KG* framework to build a large-scale NLP KG from NLP papers in an end-to-end manner. An interesting direction of further research is the diachronic analysis of TDM entities. For instance, for the two tasks-"textual entailment" and "NLI", it seems that after the emergence of SNLI corpus paper (Bowman et al., 2015), the NLP community switched to use the new name to refer to same task. In practice, the automatically-built KG (*largeNLP-KG*) has the potential to assist the researchers to search related papers and develop comparable experiments. In future, we plan to build a web-based visualization tool to enable researchers to explore KG and related papers.

Acknowledgement

The authors would gratefully like to thank the anonymous reviewers for their insightful feedback and comments towards improving the paper.

---

[5] We do not evaluate our system on SciREX because we use annotations from SciREX to train our D-NLI model.

References

Amjad Abu-Jbara, Jefferson Ezra, and Dragomir Radev. 2013. Purpose and polarity of citation: Towards NLP-based bibliometrics. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 596–606, Atlanta, Georgia. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Willhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91. Association for Computational Linguistics.

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 592–596. Association for Computational Linguistics.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346, Vancouver, Canada. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana, June 5-6, 2018*, pages 679–688.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, New Orleans, Louisiana. Association for Computational Linguistics.

Sonal Gupta and Christopher Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1–9, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273, Baltimore, Maryland. Association for Computational Linguistics.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5203–5213.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2021. TDMSci: A specialized corpus for scientific literature entity tagging of tasks datasets and metrics. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 707–714, Online. Association for Computational Linguistics.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. SciREX: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232. Association for Computational Linguistics.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2641–2651, Copenhagen, Denmark. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2020. Revisiting evaluation of knowledge base completion models. In *Automated Knowledge Base Construction*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. 2020. A re-evaluation of knowledge graph completion methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5516–5522, Online. Association for Computational Linguistics.

Yuka Tateisi, Yo Shidahara, Yusuke Miyao, and Akiko Aizawa. 2014. Annotation of computer science papers for semantic relation extraction. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1423–1429, Reykjavik, Iceland. European Language Resources Association (ELRA).

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 103–110, Sydney, Australia. Association for Computational Linguistics.

Chen-Tse Tsai, Gourab Kundu, and Dan Roth. 2013. Concept-based analysis of scientific literature. In *Proceedings of the 22nd ACM international conference on information & knowledge management*, CIKM '13, pages 1733–1738, New York, NY, USA. ACM.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Deciphering Implicit Hate: Evaluating Automated Detection Algorithms for Multimodal Hate

Austin Botelho[*]
Anti-Defamation League
abotelho@adl.org

Scott A. Hale
University of Oxford and Meedan
scott.hale@oii.ox.ac.uk

Bertie Vidgen
The Alan Turing Institute
bvidgen@turing.ac.uk

Abstract

Accurate detection and classification of online hate is a difficult task. Implicit hate is particularly challenging as such content tends to have unusual syntax, polysemic words, and fewer markers of prejudice (e.g., slurs). This problem is heightened with multimodal content, such as memes (combinations of text and images), as they are often harder to decipher than unimodal content (e.g., text alone). This paper evaluates the role of semantic and multimodal context for detecting implicit and explicit hate. We show that both text- and visual- enrichment improves model performance, with the multimodal model (0.771) outperforming other models' F1 scores (0.544, 0.737, and 0.754). While the unimodal-text context-aware (transformer) model was the most accurate on the subtask of implicit hate detection, the multimodal model outperformed it overall because of a lower propensity towards false positives. We find that all models perform better on content with full annotator agreement and that multimodal models are best at classifying the content where annotators disagree. To conduct these investigations, we undertook high-quality annotation of a sample of 5,000 multimodal entries. Tweets were annotated for primary category, modality, and strategy. We make this corpus, along with the codebook, code, and final model, freely available.

1 Introduction

Although its prevalence is fairy low (Vidgen et al., 2019), the effects of online hate can be deeply pernicious, risking real harm on targeted victims and their communities (Müller et al., 2019; Guadagno et al., 2013). A 2021 survey by Anti-Defamation League found that 81% of Americans agree social media companies should do more to counter online hate (Anti-Defamation League, 2021).

Research into automated hate detection has primarily focused on explicit varieties. However, many purveyors of hate have adopted more complex and nuanced strategies, such as dog whistling: the use of intentionally ambiguous rhetorical techniques to express hateful messages which only some audiences will recognize. For instance, calls by right-wing American political figures to "protect the suburbs" cloak racial grievances and concerns about whiteness in more prosaic concerns about community protection.

Performance in online hate classification has improved substantially from static methods like GloVe and fastText through the use of context-aware word embeddings, in particular those computed by transformer-models with self-attention (Badjatiya et al., 2019; Mozafari et al., 2020; Polignano et al., 2019; Sabat et al., 2019; Zampieri et al., 2019; Yang et al., 2019; Sohn and Lee, 2019; Kennedy et al., 2020; Vidgen et al., 2020). However, most hate detection models are text-only and cannot be applied to non-textual content (such as images and audio) and do not account for non-textual information contained in multimodal content (such as memes). This is a problem of both task definition and modeling; most hateful content training datasets do not take into account non-textual features when annotations are made, which means that non-textual systems cannot be trained and evaluated on them. The lack of detailed- and expertly-annotated datasets means that many key aspects of multimodal content classification have not yet been explored.

We address these gaps in research, making three primary contributions. First, we present a newly annotated dataset of 5,000 multimodal tweets, with labels for primary category, modality, and strategy. We make the annotation guidelines, code, and best performing models publicly available. Second, we show that as models better take into ac-

---

[*] Work performed while at the University of Oxford

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1896–1907
August 1–6, 2021.

count contextualization, from context-invariant to context-aware and unimodal to multimodal, how accurately they detect hateful content significantly improves. Though the unimodal-text context-aware model performs the best on the implicit hate subtask, the multimodal model is better overall due to a lower propensity towards false positives. Third, we show that all models perform considerably worse on ambiguous content (as determined by annotator disagreement).

2 Related Work

The networked structure of online platforms means that hate is often able to spread far beyond the author's original intended audience (Walther et al., 2011). These "masspersonal" (O'Sullivan and Carr, 2018) networks blur the divide between public and private discourse, resulting in "context collapse" as multiple audiences converge towards a singular unbounded one (Marwick and boyd, 2010; boyd, 2017). This can increase the social costs of spreading hateful messages as wider audiences and platform moderators may disapprove of this content. Consequentially, hateful actors are incentivized to employ implict rhetorical strategies to circumvent these costs. Whereas explicit forms of hate (e.g., slurs or calls to violence) are likely to draw attention, subtle forms of hate, such as dog whistles, can be effective in avoiding detection.

Dog whistles comprise a range of strategies anchored in polysemy including pseudo-factual claims (Meddaugh and Kay, 2009), normative statements (Pettigrew and Meertens, 1995), coded hate terms (Magu et al., 2017), and artistic license and humor (Milner, 2013) to create implied meanings. This enables hateful actors to target their messages at different audiences such that the hateful elements are only recognized by people who are predisposed to respond favorably (Albertson, 2015). This gives their speakers plausible deniability, allowing them to avoid any social, legal, or platform-based punishment for the content they produce.

Multimodal communication, in particular, memes, are susceptible to co-optation by hateful actors for use as dog whistles because of its ability to convey incongruent ideas through each modality ('modal dissonance'). Hateful content can be passed under the façade of a shared, seemingly benign, meme macro (or 'template') (Zannettou et al., 2018). For instance, Vidgen et al. (2019) describe how a non-hateful image (e.g., a group of Muslims in prayer) can be combined with a non-hateful text (e.g., the words 'Woken up yet?') to express prejudice. If the words or images were changed to something benign then the meme would no longer be hateful). This has led to a culture of 'shit posting' and trolling (Pelletier-Gagnon and Pérez Trujillo Diniz, 2018; Phillips, 2012).

Previous research on automated detection of hate has primarily relied on unimodal approaches with text-based features. These features have been passed through a variety of classification models (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017). Neural network architectures harnessing advances in convolutions (Gambäck and Kumar Sikdar, 2017; Ribeiro and Da Silva, 2019; Zhang et al., 2018), recurrence (Pitsilis et al., 2018), long-term memory (Badjatiya et al., 2019; Pitsilis et al., 2018), and bidirectionality (Caselli et al., 2018; Qian et al., 2018) have been applied to improved accuracy. However, a shift towards fine-tuning large, pre-trained models has yielded the best results with BERT and its varieties being the models of choice (Sohn and Lee, 2019; Zampieri et al., 2019; Mozafari et al., 2020; Polignano et al., 2019; Vidgen et al., 2020, 2021).

Despite these strides, many challenges persist as real-world interactions are noisy, varied, and multimodal. Most applications of multimodal hate speech detection have combined text with meta-information like user characteristics, comment thread information, and network connections (Chandrasekharan et al., 2017; Fehn Unsvåg and Gambäck, 2018; Gao and Huang, 2017; Qian et al., 2018; Vijayaraghavan et al., 2019). Early examples of combining text and image data yielded mixed results (Gomez et al., 2020; Sabat et al., 2019; Yang et al., 2019) leading companies like Facebook to initiate financial awards for improved performance (Kiela et al., 2020). Success in other domains like identifying pro-eating disorder content (Chancellor et al., 2017), gang activity on social media (Blandfort et al., 2019), demographic inference (Wang et al., 2019), and cyberbullying (Zhong et al., 2016) highlight the potential positive effects of multimodal approaches.

3 Schema

Our taxonomy comprises three main categories: (1) Primary attribute, (2) Modality, and (3) Strategy. The taxonomy and definitions were developed by reviewing existing theoretical frameworks for online hate and multimodal content (Vidgen et al., 2019; Waseem and Hovy, 2016; Kiela et al., 2020; Citron and Norton, 2011) and by iteratively investigating samples of tweets from the dataset.

Primary For the *Primary* attribute annotators selected one of four options: Hate, Counterspeech, Reclaimed, and Neutral. Similar to Davidson et al. (2017, p. 512), "Hate" is defined as "language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group." This definition's grounding in group identity differentiates hate from other forms of abusive content (such as interpersonal abuse) and corresponds with the definitions enforced by digital platforms like Facebook, Google, and Twitter (YouTube, 2020; Twitter, 2020; Facebook, 2020).

"Counterspeech" is defined as any response to hateful speech that undermines it or expresses support to a group that it targets. This category is needed as models trained on datasets with only a 'Neutral' category may struggle to differentiate between pro-social (Galinsky et al., 2013) tweets and hateful ones if they have similar lexical content. "Reclaimed" is defined as the use of slurs self-referentially, whereby oppressive language is reappropriated for in-group use. This category is particularly important given well-established biases in classification models, whereby they disproportionately classify the vernacular of African Americans (and other groups) as hate (Sap et al., 2019; Davidson et al., 2019). "Neutral" is defined as content which did not fall into the other three categories.

Modality For the *Modality* attribute annotators labeled the modality (image, text, or both) that was informative when making the *Primary* annotation. This was needed because although all entries contained both a text and image, both modes were not always used to express hate. In some cases the hate was expressed solely by the text or by the image, and in others both were used together.

Strategy The *Strategy* attribute captures the rhetorical devices used to convey hate. It expands upon the implicit/explicit distinction proposed by Waseem et al. (2017) and adopted by others, such as Caselli et al. (2020) and Zampieri et al. (2019). Strategy is hierarchical: if annotators identify Hate then they can select "Explicit", "Psuedo-factual", "Normative statements", "Coded language", or "Creative expressions" (see Table 1). The latter four were collapsed into a single "Implicit" category. These strategies reflect previous research on the varieties of implicit hate (Meddaugh and Kay, 2009; Dvorak, 1999; Poynting and Noble, 2003; Hughey and Daniels, 2013).

Strategy	Text	Image
Explicit	<user> <user> suck a pig dick cunt <url>	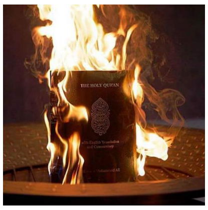
Pseudo-factual	<user> illegal criminals protected by liberals. #buildthewal <url>	
Normative	<user> <user> come on booker bring in the "race" card you always do. <url>	
Coded	best #npc meme #npcmeme #sjw <url>	
Creative	good morning tweeps #friday #teamtrump #buildthewal 🔴⚪ 🔵<url>	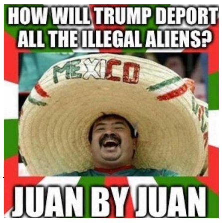

Table 1: Examples of tweets for each implicit strategy.

4 Dataset and annotation

The dataset used to train and evaluate the models originates from the MMHS150K collected by Gomez et al. (2020). It comprises 150,000 English-language tweets which all contain text and an image, and have been annotated for hate (split into 5 subcategories: racist, sexist, homophobic, religion based attacks or attacks to other communities).[1]

We enrich a sample of 5,000 multimodal tweets from the MMHS150K dataset with re-annotation.[2] Half of the dataset (2,500 tweets) was sampled using a 17-term query to provide more coverage of hateful tweets, especially covert ones.[3] The other half (2,500 tweets) was randomly sampled to ensure heterogeneity and to offset biases associated with focused sampling (Wiegand et al., 2019).

Annotation process

The annotators were given a detailed codebook to inform their decisions, with definitions, prototypical examples, and edge cases. Each tweet was labeled by two annotators. All annotators had prior experience annotating online hate, and each completed a minimum of four weeks of training. Given the frequency of mislabeled hate speech due to a lack of domain expertise (van Aken et al., 2018), we prioritized annotator experience over more scalable crowdsourcing options like Amazon Mechanical Turk. All entries with disagreement were sent for review by an expert annotator. The expert was a PhD student researching online and offline hate, who had previously worked on two annotation projects.

The Kappa score for the dataset is 0.40, indicating low to moderate agreement. However, it is a nearly three-fold increase from the MMHS150k's Kappa of 0.15 despite the increased difficulty of the task. This is equivalent to other hate speech datasets. For instance, Wulczyn et al. (2017) report Krippendorf's alpha of 0.45 and Sanguinetti et al. (2018) report category-wise Kappas of 0.37 for *offense* and 0.54 for *hate*.

In 51% of cases annotators disagreed on any of the *Primary*, *Modality*, or binarized *Strategies* – all of which were sent for review by the expert. The annotators agreed on labels far more frequently for tweets in the "None" category (68.8% of the time) than the others. Initial agreement (i.e. before expert adjudication) was 29% for Hateful, 15% for Counterspeech and 14% for Reclaimed. These relatively low agreement levels were primarily because broader situational context is often needed to make these judgments.

Annotations	Label Breakdown
Primary	Hateful: 1850 (37.0%), Counterspeech: 113 (2.3%), Reclaimed: 366 (7.3%), None: 2,671 (53.4%)
Modality	Unimodal-text: 874 (37.5%), Unimodal-image: 25 (1.1%), Multimodal: 1,430 (61.4%)
Strategy	Explicit: 31.8%, Normative: 30.4%, Coded: 22.3%, Creative: 7.8%, Psuedo-factual: 7.7%

Table 2: Dataset label breakdown.

Annotators agreed less on "Multimodal" tweets than "Unimodal-Text" tweets. This suggests the richer semantic context from the different modalities helps annotators to clarify what is being expressed. Unexpectedly, agreement was higher when labeling implicit rather than explicit hate (note that this is only for whether implicit hate was expressed, rather than identifying the particular technique used). This may be because the codebook contained more explanation of implicit hate, given we anticipated difficulties in annotating for them or because it can be difficult to ascertain when explicit slurs are hateful versus reclaimed when few signals pointing to the author's identity are available.

Dataset composition

The final labeled dataset is 37.0% Hateful, 2.3% Counterspeech, 7.3% Reclaimed, and 53.4% None (Table 2). Annotators relied on both modalities in a majority of cases (61.4%). In terms of strategy, of the tweets marked Hateful, 36.9% were explicitly hateful and 63.1% were implicitly hateful. The implicit strategies were Normative claims (30.4%), Coded language (22.3%), Creative (7.8%), and Psuedo-factual (7.7%).

5 Model implementation

All models were evaluated using the same 80/10/10 train, validation, and test split, stratified by class across the sets. Computation was completed using a single CUDA-enabled Nvidia Tesla K80 GPU in Google Colab.

---

[1] The dataset can be accessed at: `https://gombru.github.io/2019/10/09/MMHS/`

[2] The 5,000 tweet dataset can be downloaded from `https://github.com/botelhoa/Dog_Whistle_Hate`

[3] The terms are: wall, card, confederate, maga, islam, sjw, gender, crim, npc, normie,))), muslim, illegal, caravan, obama, hillary, america.

5.1 Input features

The curators of the MMHS150k dataset represented all graphics (images, GIFs and video) as thumbnails. They were resized to be a pixel dimension of 500 in the smallest direction while maintaining the original aspect ratio. Textual features are derived from two sources: the tweet body and the image text, extracted using OCR. Both text sources underwent the same pre-processing procedure, using the Ekphrasis Python library.[4] We de-noised the data by replacing hyperlinks, mentions, and dates with tags, decomposing hashtags into their constituent words, and normalizing elongated words and punctuation, in line with Mozafari et al. (2019) and Sohn and Lee (2019). To retain indicators of sentiment, variables were added for whether capital letters were used, elongated words and punctuation (Hutto and Gilbert, 2014). The remaining text was truncated to a length of 100 tokens. Longer tweets were abridged while shorter ones were padded.

5.2 Models

Four classes of models with varying levels of semantic contextualization were trained, including three classes of unimodal models—unimodal-image models, context-invariant unimodal-text models (LSTM), and context-aware unimodal-text models (Transformers)—and multimodal models. Only the best performing model in each class is reported and their tuning described. Full results are available in the supplemental materials.

For unimodal-image models, Xception, NASNet, and Inception-ResNet V2 (Chollet, 2017; Szegedy et al., 2016a; Zoph et al., 2018) were tested and Xception had the highest performance as measured by the weighted F1 score. For context-aware unimodal-text models, albert-xxlarge-v2, bert-large-uncased, electra-large-discriminator, and roberta-large (Lan et al., 2019; Clark et al., 2020; Liu et al., 2019) were tested with roberta-large performing best. For multimodals, the intermediate concatenations approaches of Gomez et al. (2020) and Sabat et al. (2019) and a joint representation approach (“MMBT”) (Kiela et al., 2019) were tested with the joint representation approach performing best.

5.2.1 Unimodal-Image

The first level of semantic contextualization consists of only image information. The extraction of image features was conducted with Xception (Chollet, 2017). Xception decouples the mapping of cross-channel and spatial correlations by performing a depthwise convolution before a pointwise convolution. This improves Top-1 and Top-5 accuracy on ImageNet compared to Inception and a significant increase in performance on the larger JFT image corpus despite maintaining the same number of model parameters (Chollet, 2017). It has yet to be applied to the task of hateful image recognition but outperforms methods that have (Gomez et al., 2020; Yang et al., 2019; Sabat et al., 2019) in general image recognition tasks (Soo Ko, 2020).

The weights pre-trained on ImageNet were downloaded from the Keras library.[5] Data augmentation was applied to the images in the train set prior to passing them through the network, including slight random rotations, height and width shifts, and horizontal flips. The images were passed in batch sizes of 32, following the approach of Szegedy et al. (2016b). The weights in the bottom layers were frozen while updates occurred only on the top 5%. A classifier was placed atop the CovNet which used two-dimensional Global Average Pooling followed by a fully connected layer of 1024 nodes with ReLU activation and a SoftMax output layer with dropout. During training, the same hyperparameters proposed by the original paper for the ImageNet task were applied with the addition of the early stopping regularization technique.

5.2.2 Unimodal-Text

In order to understand the impact of textual contextualization on the detection of implict hate, two approaches to text-only classification were implemented: (1) a context-invariant LSTM model and (2) context-aware transformer-based models.

Context-Invariant LSTM The baseline context-invariant model consists of a bi-LSTM with pre-trained fastText embeddings (Schuster and Paliwal, 1997). This is the strongest alternative to a transformer model because it considers future context (Graves and Schmidhuber, 2005) and approximately represents OOV words via character n-grams (Bojanowski et al., 2016; Joulin et al., 2016).

The LSTM comprised an embedding layer of length 300, two bidirectional LSTM layers of 256 hidden nodes with a dropout of 0.2, and a fully con-

---

[4] The documentation is available at `https://github.com/cbaziotis/ekphrasis`

[5] These weights can be found here: `https://keras.io/api/applications/`

nected output layer. It was trained over 50 epochs with early stopping of a 10-epoch patience and 0.01 minimum average validation loss improvement using mini-batches of size 64 bucketed by length to reduce the need for padding. Parameters were optimized using the weighted ADAM algorithm (Loshchilov and Hutter, 2017) with a Cross Entropy Loss function. A hyperparameter search was conducted across the learning rates $\in \{ 0.0001, 0.001, 0.01, 0.1, 1\}$.

Context-Aware Transformer The context-aware model applies a transformer architecture, namely roberta-large (Liu et al., 2019). Within the transformer, each use of a word is treated independently from its other uses. This means that, in principle, it can distinguish between the phrase "race card" when used in horse racing versus in reference to the view that racial prejudice can be advantageous to its victims. The Transformer model was coded in PyTorch with pre-trained weights loaded through the HuggingFace library.[6] It was trained for 10 epochs with early stopping of a two-epoch patience and 0.005 minimum average validation loss improvement. Parameters were optimized using the weighted ADAM algorithm (Loshchilov and Hutter, 2017) with a 0.1 weight decay and slanted triangular schedule (Howard and Ruder, 2018) with a warmup of 0.06. Backpropagation was conducted using Cross Entropy Loss. A hyperparameter search borrowed from (Liu et al., 2019) was implemented across the learning rates $\in \{$ 1e-5, 2e-5, 3e-5$\}$ and mini-batch sizes $\in \{ 16, 32, 64\}$.

5.2.3 Multimodal

Lastly, deeper semantic contextualization may be achieved through the inclusion of multimodal data. Such models should, in theory, more accurately identify implicit hate by drawing from information contained by both the image, text, and their interaction. Recent approaches use a transformer's attention mechanism to generate joint representations of images and text (Kiela et al., 2019; Li et al., 2019; Lu et al., 2020).

The MMBT first encodes image data using ResNet-152 with a generalized final pooling layer pre-trained on the ImageNet dataset. The image embeddings are combined with the tokenized text and passed through a bidirectional transformer architecture that was initialized using pre-trained BERT weights before an output layer with SoftMax activation makes the classification (Kiela et al., 2019). This was implemented using the Simple Transformers library.[7] A batch size of eight was used, with a learning rate of 1^{-5} trained with early stopping, weighted ADAM optimization (Loshchilov and Hutter, 2017) with 0.1 weight decay, and a slanted triangular schedule (Howard and Ruder, 2018) with a warmup of 0.06.

6 Results

6.1 Inter-modal Performance Comparisons

Performance for the strongest models in each of the four modality types is displayed in Table 3. Metrics were computed with a weighted average to accommodate class imbalances. The multimodal model performed the best across the four metrics, albeit only marginally so compared to the second best model, the unimodal-text context-aware model. Both unimodal-text models noticeably outperformed the unimodal-image model.

6.2 Performance by Strategy

To assess the secondary effects of defining training objectives, accuracy based on the *Strategy* annotation was calculated. These labels were not shown to the model to demonstrate performance variations hidden by oversimplified label categories.

The unimodal-text context-aware model had the highest accuracy when identifying both "Explicit" and "Implicit" hate (Table 3). By contrast, the unimodal-image model most accurately identifies non-hateful tweets, but struggles with hateful ones.

6.2.1 Ambiguity

Ambiguity is the final characteristic for which performance was assessed (Table 4). A decision in the *Primary* annotation is considered ambiguous if the two annotators provided conflicting decisions.

All models had significantly higher F1 scores on data deemed unambiguous. The multimodal model dealt the best with ambiguity (0.692) followed closely behind by the unimodal-text context-aware model (0.682). The unimodal-image model was most affected (0.758 vs 0.398) by ambiguity.

7 Error Analysis

We conducted a qualitative analysis on the errors of the multimodal model similar to the one in Vidgen

---

[6] These are available at `https://huggingface.co/transformers/`.

[7] https://github.com/ThilinaRajapakse/simpletransformers

	Overall				Accuracy by Strategy		
	Accuracy	Precision	Recall	F1	Non-Hateful	Explicit	Implicit
Unimodal-Image	0.604	0.560	0.544	0.544	**0.760**	0.273	0.370
Unimodal-Text							
Context-Invariant	0.737	0.707	0.737	0.719	0.713	0.742	0.798
Context-Aware	0.765	0.759	0.765	0.754	0.678	**0.864**	**0.941**
Multimodal	**0.785**	**0.763**	**0.785**	**0.771**	0.732	0.833	0.899
Count	502	502	502	502	317	66	119

Table 3: Overall performance of models and performance split by strategy. The multimodal model performs best overall. The context-aware unimodal-text model performs best on both implicit and explicit forms of hate.

	Unambiguous	Ambiguous
Unimodal-Image	0.758	0.398
Unimodal-Text		
Context-Invariant	0.827	0.643
Context-Aware	**0.875**	0.682
Multimodal	0.845	**0.692**
Count	272	230

Table 4: Model F1 Score by ambiguity. Entries where annotators disagree are considered ambiguous and entries with full agreement are considered unambiguous. All models perform better on Unambiguous content. The multimodal model performs best on ambiguous content.

et al. (2020). Errors were inductively categorized into "mutually exclusive and collectively exhaustive" groups (Vidgen et al., 2020, p. 7). This is visualized in the Tree Diagram in Figure 1.

The first branch on the tree is the *annotator errors*. This occured when the model's classification better represented the tweet's content, based on the annotation codebook (as determined by this paper's authors). This represented 25% of the errors and suggests that model performance could be improved further if annotator errors were eradicated. Model errors were the most frequent, accounting for 66%. Lastly, in some cases we determined that the true label was a classification other than those provided by the annotators or the model, which account for 9% of errors.

Model errors were further subdivided, as shown on the tree diagram branches. *Hate when None* includes instances where the model classified "None" content as "Hateful". This was likely caused by an over reliance on the use of slurs (i.e., a slur being used non-hatefully), misidentification of interpersonal abuse or confusion caused by content which describes/reports on (but does not endorse) hateful activities. *None when Hate* is the inverse. It includes all instances where the model classified "Hateful" content as "None". This was likely caused by a combination of tone misjudgements (e.g., hateful language which had positive sentiment), uses of the word *Nigga*[8], hate towards a

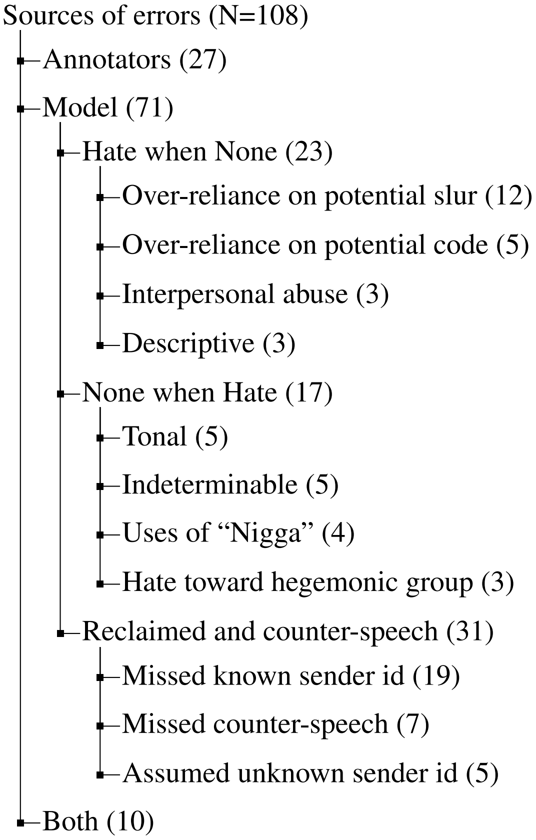

Figure 1: Sources of classification errors for the multimodal model.

---

[8] When the identity of the speaker was unknown and the use was not evidently "Hateful", annotators were instructed to treat the tweet as "None" to avoid penalizing language most

hegemonic group (e.g., referring to White people as *Hillbillies*), and, in select cases, indeterminable reasons. The final category, *Reclaimed and Counterspeech*, comprises confusion beyond the "None"–"Hateful" distinction. These were caused by missing the sender's identity when it was knowable (i.e., "Reclaimed" language was classified as "None"), assuming the sender's identity when it was unknowable (i.e., "None" content was classified as "Reclaimed", and failing to predict "Counterspeech". This final type did not exhibit systematic misclassifications, likely because there were few instances overall.

8 Discussion

As hypothesized, the unimodal-text context-aware model outperformed the unimodal-text context-invariant model. The 3.5 percentage point increase in F1 scores when distinguishing between *Primary* labels ("Hate", "Counterspeech", "Reclaimed", and "None") is in line with other results which range from 0–8 percentage point improvements depending on the dataset (Vidgen et al., 2019; Sohn and Lee, 2019; Zampieri et al., 2019; Mozafari et al., 2019).

When isolating performance by hate strategy, this gap substantially widens to 12.2 ("Explicit") and 14.3 ("Implicit") percentage points. This demonstrates the value of semantic contextualization for accurately identifying hate speech grows as its overtness diminishes. This is not the case with non-hateful content. On those, the unimodal-text context-invariant model is 3.5 percentage points more accurate.

Content modalities interact in ways that can create more subtle forms of hate. For this reason, the multimodal model was expected to outperform unimodal ones. This is marginally supported by the 1.7 percentage point gap in F1 score on the *Primary* labels between the multimodal model and the best performing unimodal model. This improvement is larger than that achieved by Gomez et al. (2020); Sabat et al. (2019); Yang et al. (2019), but smaller than that by Kiela et al. (2020) in similar hate speech detection tasks. Improvements from the incorporation of image data are minor compared to the gains from the transformer model. This may either signal underlying patterns in how multimodality is used in online content or reflect the data collection methods which relied on text-based query methods through the Twitter API.

However, it is the context-aware unimodal-text model that performs the best when assessing by *Strategy*. The gaps between the unimodal-text model and multimodal model are 2.8 percentage points ("Non-Hateful"), 3.1 percentage points ("Explicit"), and 4.2 percentage points ("Implicit"). The models which only consider image data are more attuned to "Non-Hateful" tweets to the determinant of hate identification. This implies a dissonance between modalities which mitigates, sometimes rightly and other times wrongly, hateful signals from the text.

All models more accurately identify implicit than explicit hate. This is a surprising result which may reflect their higher representation within the training data rather than any inherent property that makes them more detectable.

9 Conclusion

This paper substantiates the need to consider varying forms of hate with different modalities and levels of overtness. It investigates the value of context-aware textual and multimodal features finding that both improve model F1 score with the multimodal model performing the best (0.771). Further, we find that model performance is directly contingent upon annotator agreement levels (referred to as 'ambiguity' in the main body of the paper). These findings are generated from a newly-annotated dataset of 5,000 tweets containing information on each entry's primary attribute, modality, and strategy. This dataset along with the annotation codebook, model training code, and model weights are available to encourage future research on the topic.

Acknowledgments

This work was supported by Wave 1 of The UKRI Strategic Priorities Fund under the EPSRC Grant EP/T001569/1, particularly the "Criminal Justice System" theme within that grant, and by The Alan Turing Institute. We are grateful to all our annotators and appreciate the feedback from our reviewers as well as that from the 2020 cohort of the MSc in Social Data Science at the University of Oxford.

---

commonly used by Black communicators. Ergo, the model may associate its use with "None".

References

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for Toxic Comment Classification: An In-Depth Error Analysis. *arXiv*, pages 33–42.

Bethany L Albertson. 2015. Dog-Whistle Politics: Multivocal Communication and Religious Appeals. Technical Report 1.

Anti-Defamation League. 2021. Online Hate and Harassment: The American Experience 2021. Technical report.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2019. Deep learning for hate speech detection in tweets. In *26th International World Wide Web Conference 2017, WWW 2017 Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Philipp Blandfort, Desmond U. Patton, William R. Frey, Svebor Karaman, Surabhi Bhargava, Fei-Tzin Lee, Siddharth Varia, Chris Kedzie, Michael B. Gaskell, Rossano Schifanella, Kathleen McKeown, and Shih-Fu Chang. 2019. Multimodal Social Media Analysis for Gang Violence Prevention. *Proceedings of the International AAAI Conference on Web and Social Media*, 13:114–124.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

danah boyd. 2017. Social Network Sites: Public, Private, or What? *Knowledge Tree*.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, Inga Kartoziya, and Michael Granitzer. 2020. I Feel Offended, Don't Be Abusive! Implicit/Explicit Messages in Offensive and Abusive Language. In *Proceedings of LREC*.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. EVALITA Evaluation of NLP and Speech Tools for Italian. *Proceedings of the Final Workshop*, 12:235–238.

Stevie Chancellor, Yannis Kalantidis, Jessica A. Pater, Munmun De Choudhury, and David A. Shamma. 2017. Multimodal classification of moderated online pro-eating disorder content. In *Conference on Human Factors in Computing Systems - Proceedings*, volume 2017-May, pages 3213–3226. Association for Computing Machinery.

Eshwar Chandrasekharan, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein, and Eric Gilbert. 2017. You can't stay here: The efficacy of Reddit's 2015 ban examined through hate speech. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW):1–22.

Francois Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Danielle Keats Citron and Helen L. Norton. 2011. Intermediaries and hate speech: Fostering digital citizenship for our information age.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial Bias in Hate Speech and Abusive Language Detection Datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35, Florence. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017*, pages 512–515.

Richard Dvorak. 1999. Cracking the Code: De-Coding Colorblind Slurs during the Congressional Crack Cocaine Debates. *Michigan Journal of Race & Law*, 5.

Facebook. 2020. Community Standards.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The Effects of User Features on Twitter Hate Speech Detection. In *Proceedings of the Second Workshop on Abusive Language Online*, pages 75–85, Brussels.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text.

Adam D Galinsky, Cynthia S Wang, Jennifer A Whitson, Eric M Anicich, Kurt Hugenberg, and Galen V Bodenhausen. 2013. The Reappropriation of Stigmatizing Labels: The Reciprocal Relationship Between Power and Self-Labeling. *Psychological Science*, 24(10):2020–2029.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-Speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Lei Gao and Ruihong Huang. 2017. Detecting Online Hate Speech Using Context Aware Models. In *International Conference Recent Advances in Natural Language Processing, RANLP*, pages 260–266. Association for Computational Linguistics (ACL).

Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas. 2020. Exploring Hate Speech Detection in Multimodal Publications. Technical report.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In *Neural Networks*, volume 18, pages 602–610. Pergamon.

Rosanna E Guadagno, Daniel M Rempala, Shannon Murphy, and Bradley M Okdie. 2013. What makes a video go viral? An analysis of emotional contagion and Internet memes.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Matthew W Hughey and Jessie Daniels. 2013. Racist comments at online news sites: a methodological dilemma for discourse analysis. *Culture & Society*, 35(3):332–347.

C J Hutto and Eric Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *The International AAAI Conference on Web and Social Media (ICWSM)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, 2:427–431.

Brendan Kennedy, Xisen Jin, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. 2020. Contextualizing hate speech classifiers with post-hoc explanation.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. 2019. Supervised multimodal bitransformers for classifying images and text.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine Facebook. 2020. The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes. *ArXiv*, pages 1–17.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 2020. 12-in-1: Multi-task vision and language representation learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rijul Magu, Kshitij Joshi, and Jiebo Luo. 2017. Detecting the Hate Code on Social Media. Technical report.

Alice E Marwick and danah boyd. 2010. Article I tweet honestly, I tweet passionately: Twitter users, context collapse, and the imagined audience. *New Media & Society*, 13(1):111–133.

Priscilla Marie Meddaugh and Jack Kay. 2009. Hate Speech or "Reasonable Racism?" The Other in Stormfront. *Journal of Mass Media Ethics*, 24(4):251–268.

Ryan M Milner. 2013. FCJ-156 Hacking the Social: Internet Memes, Identity Antagonism, and the Logic of Lulz. *The Fibreculture Journal*, (22):62–92.

Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. 2019. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. *Studies in Computational Intelligence*, 881 SCI:928–940.

Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2020. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. In *Studies in Computational Intelligence*, volume 881 SCI, pages 928–940. Springer.

Karsten Müller, Carlo Schwarz, Sascha Becker, Christopher Blattman, Leonardo Bursztyn, Mirko Draca, Thiemo Fetzer, Evan Fradkin, Matthew Gentzkow, Andy Guess, Vardges Levonyan, Atif Mian, Magne Mogstad, Sharun Mukand, Hans-Joachim Voth, Fabian Waldinger, and Noam Yuchtman. 2019. Fanning the Flames of Hate: Social Media and Hate Crime *. Technical report.

Patrick B O'Sullivan and Caleb T Carr. 2018. Masspersonal communication: A model bridging the mass-interpersonal divide. *New Media & Society*, 20(3):1161–1180.

Jérémie Pelletier-Gagnon and Axel Pérez Trujillo Diniz. 2018. Colonizing Pepe: Internet Memes as Cyberplaces. *Space and Culture*, page 120633121877618.

T. F. Pettigrew and R. W. Meertens. 1995. Subtle and blatant prejudice in western Europe. *European Journal of Social Psychology*, 25(1):57–75.

Whitney M. Phillips. 2012. *This is why we can't have nice things: The origins, evolution and cultural embeddedness of online trolling*. Ph.D. thesis, University of Oregon.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Effective hate-speech detection in Twitter data using recurrent neural networks. *Applied Intelligence*, 48(12):4730–4742.

Marco Polignano, Pierpaolo Basile, Marco De Gemmis, and Giovanni Semeraro. 2019. Hate Speech Detection through AlBERTo Italian Language Understanding Model. Technical report.

Scott Poynting and Greg Noble. 2003. 'Dog-Whistle' Journalism and Muslim Australians since 2001. *Media International Australia incorporating Culture and Policy*, 109(1):41–49.

Jing Qian, Mai Elsherief, Elizabeth M Belding, and William Yang Wang. 2018. Leveraging Intra-User and Inter-User Representation Learning for Automated Hate Speech Detection. *arXiv*, pages 1–6.

Alison P Ribeiro and Nádia F F Da Silva. 2019. INF-HatEval at SemEval-2019 Task 5: Convolutional Neural Networks for Hate Speech Detection Against Women and Immigrants on Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, pages 420–425.

Benet Oriol Sabat, Cristian Canton Ferrer, and Xavier Giro-i Nieto. 2019. Hate Speech in Pixels: Detection of Offensive Memes towards Automatic Moderation. In *AI for Social Good workshop at NeurIPS*, Vancouver, CA.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An Italian Twitter corpus of hate speech against immigrants. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, Noah A Smith, and Paul G Allen. 2019. The Risk of Racial Bias in Hate Speech Detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence. Association for Computational Linguistics.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional Recurrent Neural Networks. Technical Report 11.

Hajung Sohn and Hyunju Lee. 2019. MC-BERT4HATE: Hate speech detection using multichannel bert for different languages and translations. In *IEEE International Conference on Data Mining Workshops, ICDMW*, volume 2019-November, pages 551–559. IEEE Computer Society.

Byung Soo Ko. 2020. ImageNet Classification Leaderboard.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2016a. Inception-v4, inception-resnet and the impact of residual connections on learning.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016b. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 2818–2826. IEEE Computer Society.

Twitter. 2020. The Twitter Rules.

Bertie Vidgen, Austin Botelho, David Broniatowski, Ella Guest, Matthew Hall, Helen Margetts, Rebekah Tromble, Zeerak Waseem, and Scott Hale. 2020. Detecting east asian prejudice on social media.

Bertie Vidgen, Helen Margetts, and Alex Harris. 2019. How much online abuse is there? A systematic review of evidence for the UK Policy Briefing-Full Report Public Policy Programme Hate Speech: Measures and Counter Measures. Technical report, The Alan Turing Insitute.

Bertie Vidgen, Dong Nguyen, Helen Margetts, Patricia Rossini, and Rebekah Tromble. 2021. Introducing CAD: the contextual abuse dataset. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2289–2303, Online. Association for Computational Linguistics.

Prashanth Vijayaraghavan, Hugo Larochelle, and Deb Roy. 2019. Interpretable Multi-Modal Hate Speech Detection. In *International Conference on Machine Learning AI for Social Good Workshop*, pages 1–5, Long Beach.

Joseph B. Walther, Caleb T. Carr, Scott Seung W. Choi, David C. DeAndrea, Jinsuk Kim, Stephanie Tom Tong, and Brandon Van Der Heide. 2011. Interaction of Interpersonal, Peer, and Media Influence Sources Online: A Research Agenda for Technology Convergence. In Zizi Papacharissi, editor, *A Networked Self: Identity, Community, and Culture on Social Network Sites*, chapter 1, pages 17–38. Routledge, New York.

Zijian Wang, Scott Hale, David Ifeoluwa Adelani, Przemyslaw Grabowicz, Timo Hartman, Fabian Flöck, and David Jurgens. 2019. Demographic inference and representative population estimates from multilingual social media data. In *The World Wide Web Conference*, WWW '19, page 2056–2067, New York, NY, USA. Association for Computing Machinery.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In

Proceedings of the First Workshop on Abusive Language Online, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of abusive language: the problem of biased datasets. The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Proceedings of the Conference Vol. 1. Minneapolis, Minnesota, June 2 - June 7, 2019, pages 602–608, Stroudsburg, PA, USA. The Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *26th International World Wide Web Conference, WWW 2017*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Fan Yang, Xiaochang Peng, Gargi Ghosh, Reshef Shilon, Hao Ma, Eider Moore, and Goran Predovic. 2019. Exploring Deep Multimodal Fusion of Text and Photo for Hate Speech Classification. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 11–18, Florence. Association for Computational Linguistics.

YouTube. 2020. Hate speech policy - YouTube Help.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). *arXiv*, pages 1–12.

Savvas Zannettou, Tristan Cauleld, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini, and Guillermo Suarez-Tangil. 2018. On the origins of memes by means of fringe web communities. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pages 188–202. Association for Computing Machinery.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10843 LNCS, pages 745–760. Springer Verlag.

Haoti Zhong, Hao Li, Anna Squicciarini, Sarah Rajtmajer, Christopher Griffin, David Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3952–3958, New York. International Joint Conferences on Artificial Intelligence.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Studying the Evolution of Scientific Topics and their Relationships

Ana Sabina Uban$^{\diamondsuit,\spadesuit}$ **Cornelia Caragea**$^{\clubsuit}$ **Liviu P. Dinu**$^{\diamondsuit}$

$^{\diamondsuit}$Human Languages Technologies Research Center, University of Bucharest, Bucharest, Romania
$^{\spadesuit}$Universitat Politècnica de València, València, Spain
$^{\clubsuit}$Computer Science, University of Illinois at Chicago, Chicago, Illinois
`ana.uban+acad@gmail.com, cornelia@uic.edu, ldinu@fmi.unibuc.ro`

Abstract

We propose a study of the development of scientific topics through time, as well as the relations between them within the scientific field of computational linguistics and across subfields. We use topic modeling to analyze scientific texts published in the ACL Anthology, and introduce a categorization of topics in our field into 3 types: tasks, algorithms, and data. In order to understand how topics emerge, evolve, and gradually disappear over time, we analyze the evolution of these topics across time through several case studies. We further include in our analysis papers published in NeurIPS, and try to understand whether there was any influence between topics in this conference focused on neural methods and computational linguistics conferences, as well as measure the divergence over time between conferences in terms of the topics approached. We additionally look at the relationships between topics, categorizing them into types of competing or cooperating topics.

1 Introduction

Scientific fields progress through innovation. Science functions under the premise that, when new better topics appear in research, they overtake the old ones and contribute to shaping the progress of the research field (Kuhn, 2012). Nevertheless, scientific topics evolve interdependently (the appearance and popularity of one topic may affect the popularity of another) and oftentimes, the focus of research in a certain field is also influenced by topics in other related subfields.

We propose a multidimensional approach for studying scientific topics and their evolution, by analyzing our field of research - computational linguistics - from several points of view: we look at the parallel evolution of topics in computational linguistics and their popularity over time, as well as how they relate to each other, engaging in cooperating or competing relationships. We also extend this perspective by considering the interplay of topics within a field, as well as the context in which they appear, and how the same topic is portrayed in different subfields, with a focus on the mutual influence between ideas in computational linguistics and those in the related field of neural networks.

Among studies that track the evolution of topics in scientific texts, Hall et al. (2008) focused on scientific text in computational linguistics, analyzing papers published in ACL, EMNLP and COLING between 1978 and 2006. The authors identify increasing and decreasing trends up to 2006, and make predictions about the subsequent evolution of the field. We continue the analysis including articles published up until the end of 2018, and uncover current shifts and trends that may not have been predictable 15 years ago - such as the rise of neural networks methods.

In our work, we study topics across three types: tasks, algorithms and data. Moreover, our aim is to further and complement the previous explorations of topics in computational linguistics not only by extending the analysis to recent years, but also by looking at relations between topics within and across fields. We analyze texts in four top computational linguistics conferences (adding NAACL to the three conferences analyzed in (Hall et al., 2008)). We additionally propose an exploration of topics across conferences and subfields, and include in our analysis papers published in the Conference on Neural Information Processing Systems (NeurIPS), which is a machine learning conference focused on neural networks. Considering that in recent years neural networks have almost dominated methods used in computational linguistics, we try to understand how topics approached in computational linguistics relate to those in the more focused field of neural networks, and whether and how they

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1908–1922
August 1–6, 2021.

migrate between these conferences.

Our analysis of topic relationships within computational linguistics is inspired from Tan et al. (2017), in which the authors propose a way to classify topic relationships into four types, based on their co-occurrence in text and the degree of correlation between their popularity over time. In our paper, we extend this and take a deeper look at the relations existing between topics in scientific text. We propose interpretations of topic relationships in the context of a scientific domain, and report interesting findings on how these types of relationships manifest between scientific topics, discovering, for example, which algorithms in computational linguistics are compatible with certain tasks (such as neural machine translation and RNNs), or finding pairs of topics that represent algorithms which have replaced one another along the history of computational linguistics (such as statistical machine translation and neural machine translation).

2 Previous work

Multiple previous studies have looked at evolution of topics through time, analyzing texts of various genres, from news (Michel et al., 2011; Rule et al., 2015) to emails (Wang and McCallum, 2006) to scientific articles (Hall et al., 2008; Prabhakaran et al., 2016; Griffiths and Steyvers, 2004; Blei and Lafferty, 2006; Anderson et al., 2012).

Popular choices for representing topics include topic models, to which some studies add variations specific to tracking trends over time, such as the continuous-time model proposed by Wang and McCallum (2006), the generative model proposed by Bolelli et al. (2009a,b), or the dynamic topic model (Blei and Lafferty, 2006). Hall et al. (2008) use an approach based on topic modeling, and focus on scientific texts in computational linguistics, analyzing papers published in ACL, EMNLP and COLING between 1978 and 2006. Gollapalli and Li (2015) use topic models and keyphrase extraction to compare topics in ACL and EMNLP. In other studies on scientific articles, topic representations are enriched with additional features such as citations (He et al., 2009). Citations and citation networks have been leveraged extensively in previous studies for tracking scientific topics (Shibata et al., 2008, 2009; Jurgens et al., 2018), analyzing the structure of the scientific community (Leicht et al., 2007), or summarizing scientific papers (Qazvinian and Radev, 2008), or entire technical topics (Qazvinian et al., 2013). Other authors make use of rhetorical framing to predict the patterns present in the development of scientific topics (Prabhakaran et al., 2016).

Not as many studies attempt to provide in-depth systematic analyses of the relations between topics within a field or across fields, independently from the publications where they occur. Zhang et al. (2017) introduce a learning technique to identify the evolutionary relationships (e.g., topic evolution, fusion, death, and novelty) between scientific topics. Grudin (2009) study the particular relationship between the field of AI and Human Computer Interaction. Shi et al. (2010) propose a temporal comparison of grant proposals and academic publications, in an attempt to understand which precedes the other and how they influence each other. In one of the most extensive studies on the topic (Tan et al., 2017), the authors propose a systematic way of classifying relations between topics into four types of cooperating or competing topics, based on their patterns of co-occurrence and prevalence correlation: friendships, arms-race, head-to-head, and tryst. We build on this framework in our analysis of the field in the following sections.

3 Dataset

Our study focuses on topics in computational linguistics and their evolution. For exploring this topic, we make use of articles published in the ACL Anthology (Bird et al., 2008; Radev et al., 2013) from its inception. We collect all papers published in four top conferences: ACL, EMNLP, COLING and NAACL over time, obtaining a total of 14,737 computational linguistics articles overall. We will further refer to the set of computational linguistics conferences we considered by using the general term *ACL+*.

For the second stage of our study, we additionally use articles published in the NeurIPS conference, from which we collect all articles published since 1994, in total 6,520 articles. Table 1 shows the number of articles for each time period (across 5-year time spans) for the ACL+ conferences and NeurIPS. In Figure 1 we show the number of papers published as a time series, computed separately for each of the conferences considered. We make our collected dataset as well as code used for our experiments publicly available.[1]

---

[1] https://github.com/ananana/scientific_topics_history

Period	Number of articles	
	ACL+	NeurIPS
pre-1980	374	-
1980-1985	332	-
1986-1990	729	-
1991-1995	609	157
1996-2000	1108	842
2001-2005	950	767
2006-2010	3456	1449
2011-2015	3432	1091
2016-2018	3747	2214

Table 1: Number of articles per time period.

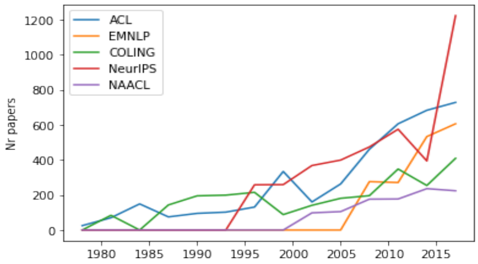

Figure 1: Number of published papers per conference.

4 Representation of ideas

We base our study on the premise proposed by Kuhn (2012) that science proceeds by shifting from one paradigm to another, viewing the evolution of science as a series of topics that follow and replace one another. Furthermore, we assume that these shifts in topics are directly reflected in shifts at the level of the vocabulary employed in the articles that discuss them.

Based on this assumption, we choose to represent topics by relying on topics extracted using unsupervised topic modeling, which treats documents as bags of words generated by one or more topics. We choose to measure the topics' evolution over time post-hoc, using a classical topic model and monitoring the change in topic prevalence over time. While dynamic topic models (Blei and Lafferty, 2006) allow to include temporal information in the generated labels themselves, they impose additional constraints on the time periods (for example assuming the changes between consecutive years are the same). We design our representation of topics starting from the observation that computational linguistics research can generally be described as comprising of a set of research tasks, which researchers aim to solve by employing appropriate algorithms, usually assisted by the use of datasets. Based on this assumption, we propose that topics in computational linguistics can naturally be categorized into 3 types: *tasks*, *algorithms* and *data*. As such, we propose a notion of *scientific topic* in our field which consists of both a topic and its category or type; in this view, a topic in computational linguistics can be defined as:

$$(\textbf{topic}_t, \textbf{type}_c),$$
$$type_c \in \{task, algorithm, data\}, topic_t \in T,$$

with T representing the list of topics generated by the Latent Dirichlet Allocation model (LDA) (Blei et al., 2003). These topic categories can be useful beyond our field and application, for example in question answering systems or paper recommendation systems (Augenstein et al., 2017; Park and Caragea, 2020; Luan et al., 2018; QasemiZadeh and Schumann, 2016). In order to identify the topics occurring in our corpus of scientific texts, we first train an LDA model on the full texts extracted from computational linguistics articles, and use it to extract a set of 100 topics which we will use to analyze the evolution of the field in the next stages of our study. We use the Mallet implementation of LDA[2], with parameters set to 100 topics, and 100 training passes. The asymmetric prior distribution was learned directly from the data. The resulted model has a topic coherence score of 0.484 according to the C_V coherence measure.

In order to maximize the quality of the produced topics, we first label the obtained sequences with POS tags and select only words with POS tags corresponding to content words: nouns, verbs, adjectives, and adverbs, and discard the rest. We lowercase and lemmatize the texts, and we extract bigrams and trigrams using PMI scores to select words which occur together with high probability and add them to our vocabulary and document representations. On the collection of articles published in the ACL Anthology preprocessed as described above, we train the topic model to extract 100 topics. We do not intervene with significant changes on the output of the model, and only add minor corrections, through manual curation: we remove 10 of the extracted topics which we do not consider to represent coherent or interesting ideas, and merge a few topics which were redundant. We are left with a total of 82 topics.

We then manually label each topic with one of the three proposed categories: *task* / *algorithm* / *data*, and obtain the final list of topics occurring in our corpus. Each topic can be assigned one or more types: we obtain 53 topics labelled as *tasks*, 33 of the topics are *algorithms*, while 7 topics fall

---

[2] http://mallet.cs.umass.edu/

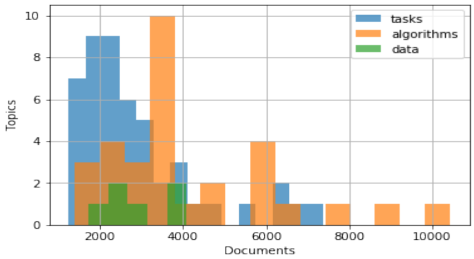

Figure 2: Distribution of topics for each type.

into the *data* category. Some topics belonging to the *task* type are, for example, *morphology*, *event extraction*, or *summarization*. Topics such as *recurrent neural networks* or *topic models* fall under the category of *algorithms*, whereas *lexicons* and *parallel corpora* are categorized as *data* (or resources). A few topics refer to inherently connected tasks and algorithms, we label those with both types - as is the case of *neural machine translation* or *statistical machine translation*. The appendix lists the entire set of extracted topics, along with the top 10 keywords that are relevant for each, as well as their types. When topics were merged, the list of keywords relevant for each topic were merged into one larger list.

After having generated our list of topics, we further extract for each paper a list of relevant topics, considering only those which are present in the topic distribution for that document with a probability greater than 0.01. After this step, we are left with almost 13 relevant topics per article, on average. Finally, we measure the prevalence of a topic during a certain year by computing the empirical probability of its occurrence relative to the total number of topics that were approached overall in that year:

$$\begin{aligned}P(t|y) &= \sum_{d:t_d=y} P(t|d)P(d|y)\\ &= \frac{1}{C_y}\sum_{d:t_d=y} P(t|d)\\ &= \frac{1}{C_y}\sum_{d:t_d=y}\sum_{t_i^{'}\in d} I(t_i^{'}=t),\end{aligned}$$

where I is the indicator function, t_d is the year in which document d was published. The conditional probability of a topic given a document $P(t|d)$ is thus equal to 1 if the topic is present in the document and 0 otherwise. C_y represents the total number of documents written in a year y.

Figure 2 illustrates the distribution of topics across the computational linguistics corpus for each of the 3 topic types. Although the list of topics is generated using the full dataset of papers published, in our time series showing the popularity of topics in scientific papers over time we only consider papers published after 1978, when ACL was first organized. Similarly, when considering topics occurring in NeurIPS, our analyses will include the years when NeurIPS papers were published. All of the plots in the following sections show smoothed versions of the raw values of topic probabilities per year, using a rolling average with a window of two years.

Topic	Top cited authors
Sentiment analysis (task)	J Wiebe, C Manning, L Lee, B Liu, B Pang
Topic models (algorithm)	C Manning, D Blei, A Mccallum, A Ng, Y Bengio
Coref. resolution (task)	C Manning, V Ng, D Klein, D Roth, C Cardie
Discourse (task)	D Marcu, A Joshi, C Manning, B Webber, B Grosz
Speech recognition (task)	E Shriberg, A Stolcke,H Ney, J Hirschberg, M Johnson
Neural MT (task, algorithm)	Y Bengio, K Cho, C Manning, I Sutskever, O Vinyals

Table 2: Most influential authors for a subset of topics.

5 Selected topics and trends

In order to narrow our focus to subsets of topics worthy of interesting insights, we propose a few ways to select topics that stand out and comment on their development over time - several case studies will be presented in the following subsections.

We also look into the most influential authors for each topic. We consider citations as an indicator of the influence of an author over a topic, and we thus measure the influence of each author for a topic by counting all of the occurrences of citations referring to the given author (regardless of the topic of the cited article) in all documents in our collection where the topic is present. Table 2 shows the top 5 most influential authors, ranked by number of citations, for a selection of topics.

Confirming and refuting predictions We first confront our findings with the predictions made in previous studies which looked at the evolution of scientific ideas in computational linguistics. Hall et al. (2008) identify a list of topics which were then on an increasing trend in 2006: *classification*, *probabilistic models*, *statistical parsing*, *statistical machine translation* and *lexical semantics*. We find among our topics those which best match their list, then analyze their evolution in order to see whether the predictions made then still hold today. Figure 3 shows the evolution of four of these topics

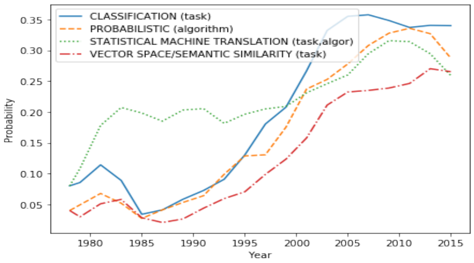

Figure 3: Topics on an increasing trend in 2006.

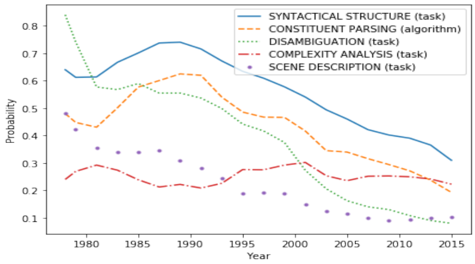

Figure 4: Top prevalent ideas in computational linguistics and their evolution.

until 2018: not all of the topics have maintained the same upward trend all through 2018. *Statistical machine translation* and *probabilistic models* suffer a decrease in popularity after 2010; *classification*, though still very popular, has reached a plateau, while *lexical semantics* seems to be still on an increasing trend, though less abruptly.

Most prevalent topics In our second case study we focus on the most prevalent topics overall, which we consider to be ones that over time have received the greatest attention in computational linguistics research. To find these, we average the probability of occurrence of a topic in each year, obtaining for each topic an overall score of prevalence:

$$\text{Prev}(t) = \frac{1}{|Y|} \sum_{y \in Y} P(t|y)$$

Figure 4 shows the evolution of the top 5 most prevalent topics in ACL+ across time. Most of these were very popular in the earlier days of computational linguistics and started to decrease around 1990, such as the topics related to syntax. *Complexity analysis* has a steady evolution across time, maintaining a relatively flat trend.

Topics with largest variation In our next analysis, we extract topics which vary most in popularity over time, hoping to discover topics which stand out because of their dramatic evolution over time. We do this by considering the distribution of probabilities for a topic over the years, and measuring its standard deviation, for each topic, then select those

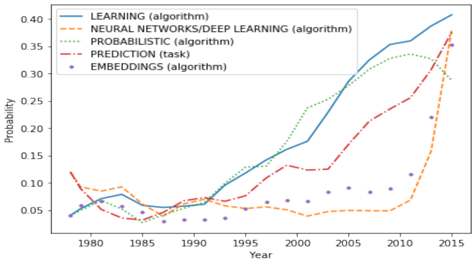

Figure 5: Topics which show greatest variation in linguistics conferences and their evolution.

topics where standard deviation is highest. The top 5 such topics and their evolution are illustrated in Figure 5. It seems that the most dramatic variations are related to recent increases in popularity of certain topics, most of which relate to machine learning. The steep and constant increase in popularity of the *learning* topic is apparent. Among the first 5 topics which vary most dramatically in popularity over time we find topics related to neural networks, which although very recent relative to the entire history of computational linguistics, have quickly caught up in popularity and even surpassed more traditional topics in the field, and show an abrupt increase in popularity after 2010. We analyze topics related to neural networks in more detail in the following paragraphs.

Neural networks In our final case study, we zoom in specifically on topics related to neural methods. These are shown in our previous results to be the stars of recent years in computational linguistics, showing an abrupt increase in popularity.

The list of topics generated by our LDA model produce no less than four distinct topics related directly to neural networks, found in computational linguistics papers, which is already remarkable for such a recent topic. These are: *neural networks*, *recurrent neural networks*, *neural machine translation* and *embeddings*. To these we add for our analysis the topic of *learning*, as the general class of topics under which neural networks fall, and whose evolution we also expect to be affected by the popularity of neural networks.

Furthermore, we compare the trends of neural network related topics in ACL+ to the same trends present in a conference focused primarily on neural networks: NeurIPS. In order to achieve this, we use our LDA model trained on ACL+ papers to extract topics from NeurIPS papers. Figure 6 shows the evolution of topics related to neural methods in papers published in ACL+ and in NeurIPS, respectively. Both papers in ACL+ and in NeurIPS

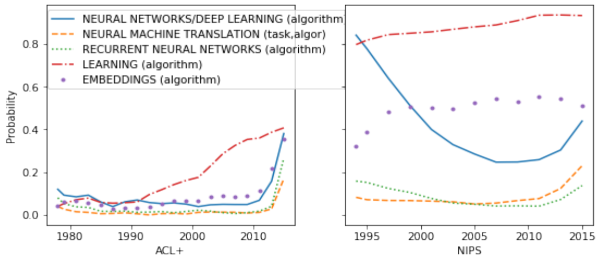

Figure 6: Neural topics evolution in ACL+ vs NeurIPS.

show the same steep increase in *recurrent neural networks* and *neural machine translation* starting between 2010 and 2015. *Learning* has a clearly more stable evolution in NeurIPS, where it has been a very popular topic from the beginning, as compared to computational linguistics, where it sees a steady and still continuing increase. Interestingly, *neural networks* as a general topic evolve differently in NeurIPS and ACL+: while in computational linguistics they are a recent topic, with a sudden increase in popularity after 2010, in NeurIPS they were widely discussed from 1994, and have suffered a decline up to 2010 when they started following the same upward trend.

6 Relationships between Topics

Methodology We use measures of relatedness between topics on two dimensions: co-occurrence and prevalence correlation, to characterize relationships into four major types of relations, which will be described and interpreted in more detail in this section: friendship, head-to-head, arms-race and tryst.

For categorizing pairs of topics into these types of relationships, we obtain co-occurrence scores for a pair of topics by computing the PMI score for the topics as they co-occur in documents, and compute the correlation score as the Pearson correlation between the time series represented by the topic's probability over time.

$$\mathrm{Corr}(t_1, t_2) = \frac{\sum_y (P(t_1|y) - \overline{P(t_1|y)})((P(t_2|y) - \overline{P(t_2|y)})}{\sqrt{\sum_y (P(t_1|y) - \overline{P(t_1|y)}^2}\sqrt{\sum_y (P(t_2|y) - \overline{P(t_2|y)}^2}}$$

We then split each of these two dimensions into two classes (positive/negative co-occurrence, and positive/negative correlation), obtaining the four types of relationships from their combinations. We first standardize the distributions of co-occurrence and correlation scores, then split the relations landscape into four parts, depending on where they are situated on the two axes: positive/negative co-occurrence and positive/negative correlation.

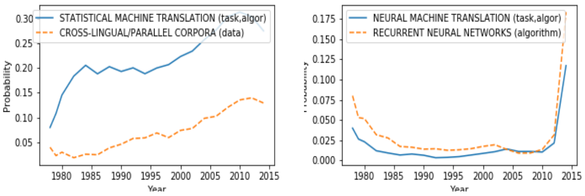

Figure 7: Examples of topics in friendship relationships and their evolution over time.

We also compute a measure of strength of each relationship between a pair of topics, which is simply the product of the two scores, in absolute value. Sorted by the average strength of top 25 relations of that type, the relations rank as follows: friendships>head-to-head>tryst>arms-race. Table 3 shows the top pairs of topics with the strongest relations for each relation type, as well as their strength. The appendix contains tables with the top 10 relations for each relation and topic type.

We separately identify relations between different types of topics, and propose that some relations are more meaningful for certain topic pairings than others, depending on their types. For friendships, which refer to cooperating topics, we focus on topic pairs of different types, between which this relation is established, in order to discover the tasks go together with specific algorithms or datasets. For the other relation types (arms-race, head-to-head and tryst), we suggest that the cross-type topic pairs are less meaningful - since these types of relations can be interpreted as occurring between competing topics - for these we focus instead on same-type topic pairs (between tasks and tasks, algorithms and algorithms, data and data). In the tables presenting top relationships for each type, we restrict our focus to only topic pairs of types which can be meaningfully matched for each relation.

Friendships Two topics are "friends" if they tend to co-occur in the same texts and are also correlated in their prevalence over time. These are topics which go together, or "cooperate" - they are often found in the same documents and are used together in the analysis of a certain idea or area of interest. Figure 7 shows the top strongest friendship relationships between a task and an algorithm, and an algorithm and data, respectively. We discover, for example, that the *neural machine translation* task is most associated with the *recurrent neural networks* algorithm, and that for the task of *statistical machine translation*, *parallel corpora* are the most

Topic1		Topic2		Rel Type	Rel Strength
Neural MT	(Task)	RNNs	(Algorithm)	Friendship	13.03
Statistical MT	(Task)	Parallel Corpora	(Data)	Friendship	4.25
Transfer Learning	(Algorithm)	Parallel Corpora	(Data)	Friendship	3.76
Phonology	(Task)	Semantic Role Labelling	(Task)	Arms-race	1.98
Topic Models	(Algorithm)	Dependency Parsing	(Algorithm)	Arms-race	1.45
Unification	(Task)	Neural MT	(Task)	Head-to-head	6.27
Grammars	(Algorithm)	Neural MT	(Algorithm)	Head-to-head	5.40
Statistical MT	(Algorithm)	Neural MT	(Algorithm)	Tryst	2.91
Vision/Multimodal	(Task)	Scene Description	(Task)	Tryst	2.20
Dictionaries	(Data)	Parallel Corpora	(Data)	Tryst	2.23

Table 3: Top strongest relationships for each type, along with strength scores.

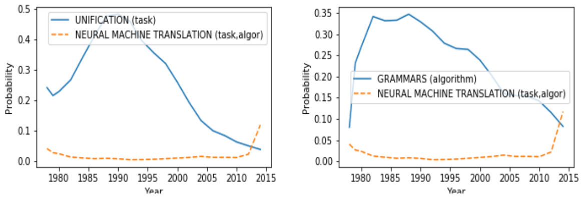

Figure 8: Examples of topics in head-to-head relationships and their evolution over time.

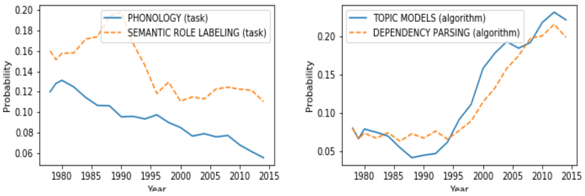

Figure 9: Examples of topics in arms race relationships and their evolution over time.

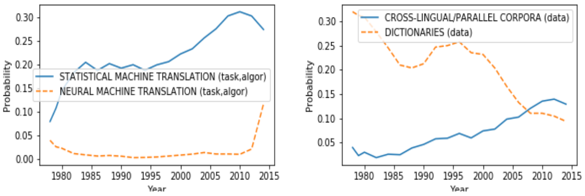

Figure 10: Examples of topics in tryst relationships and their evolution over time.

useful types of datasets.

Head to head Topics in a head-to-head relationship do not tend to co-occur in the same documents, and are anti-correlated over time. These are topics which have nothing in common, or are even rivals. In Figure 8 we can see the strongest head-to-head relationships in our corpus between tasks and algorithms respectively. One example is the relation between *grammars* and *neural machine translation*: these are rarely treated together in studies; more than that, while *neural machine translation* shows a recent increase in popularity, *grammars* are on a declining trend.

Arms race An arms-race relation characterizes topics that are correlated in their usage over time, but do not tend to co-occur within the same documents. Topics in this type of relationship tend to evolve in a similar pattern over time, possibly with an underlying common cause, even though they are not directly related: such is the case of many algorithms which were widely used before being recently replaced by neural networks. Figure 9 shows two such pairs of topics: *phonology* with *semantic role labelling*, and *topic models* with *dependency parsing*, which show similar decreasing trends, but are not referred to in the same articles.

Trysts Tryst is a relationship between topics which tend to co-occur in the same texts, but are anti-correlated in prevalence over time. We show that according to our study, this is one of the most interesting relations occurring between scientific topics, and propose that it is useful for discovering topics that are replaced by others: topics which share a common niche of the research field, but as one topic increases, the other decreases.

In Figure 10 we see two such relationships, which uncover interesting topic pairs. One is *statistical machine translation* versus *neural machine translation*, which is clearly a topic in the subfield of machine translation which has recently replaced the previous one as the primary focus of researchers. A similar phenomenon may have occurred for *data*-typed topics related to language resources: while *dictionaries* are overall more studied, they are on a decreasing trend, and have now been surpassed in popularity by *parallel corpora*.

7 Relations between conferences

Conference divergence In this part of our study we focus on the relations between conferences in computational linguistics. We compute divergence between conferences using Jensen-Shannon divergence applied on their topic distributions generated by papers published in each conference. Jensen-Shannon divergence is computed as the average of

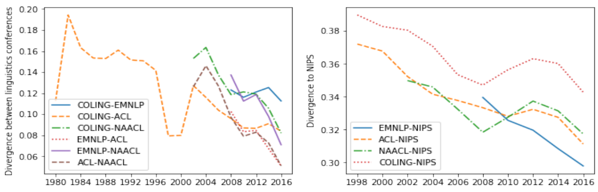

(a) ACL+ conferences. (b) ACL+ vs NeurIPS.

Figure 11: Conference divergence over time for (a) ACL+ conferences and (b) ACL+ vs NeurIPS.

the KL divergences between each of the distributions and the average of the distributions. Its value is 0 for identical distributions, and tends to infinity as the two differ more and more.

Figure 11 shows the pairwise divergence over time between the computational linguistics conferences, as well as between the linguistics conferences and NeurIPS. The span of each pairwise divergence plot is limited to the span of the youngest conference in the pair; the values are smoothed using a rolling average with a window of 2 years. The plot reveals a decreasing trend for all conference pairs. ACL and COLING are the conferences with the oldest history, and show a steady but mild decrease in divergence throughout their evolution. The most similar conferences are shown to be ACL with EMNLP and with NAACL, which also show the steepest decrease in divergence.

We further extend our study to contrast the computational linguistics conferences with NeurIPS. It is interesting to see that, even though computational linguistics and neural methods are technically distinct fields, the linguistics conferences still tend to converge with NeurIPS over time (although the absolute divergence between these is still considerably higher than among computational linguistics conferences). The most similar conference to NeurIPS in terms of the topics approached seems to be EMNLP, which from its beginning was the closest to NeurIPS among all linguistics conferences. This is perhaps explained by the more applied character of EMNLP compared to the others. In contrast, COLING, the oldest and most linguistics-focused of the conferences, is the least similar to NeurIPS, although still shows a tendency towards decreasing this gap.

Synchronicity of topics across conferences Next, we introduce a second measure of similarity between conferences, this time over particular topics, in order to understand if conferences are synchronized in the topics they approach, and if this depends on particular sets of topics. Similarly to the measure of correlation used in the topic relationship analysis, the correlation between conferences for a subset of topics T is simply computed as the prevalence correlation of topics over time, on average, for each topic in the subset considered - this time between its evolution in the two conferences (or sets of conferences) to be analyzed.

Topic	Correlation
Reinforcement learning	0.93
Finite state machines	0.90
Disambiguation	0.90
Ranking	0.89
Neural machine translation	0.88

Table 4: Correlated topics between ACL+ and NeurIPS.

$$\mathrm{Corr}_T(c_1, c_2) = \frac{1}{|T|} \sum_{t \in T} \mathrm{Corr}_t(c_1, c_2),$$

where the correlation between two conferences for a certain topic t is defined as:

$$\frac{\sum_y (P(t|y,c_1) - \overline{P(t|y,c_1)})((P(t|y,c_2) - \overline{P(t|y,c_2)}))}{\sqrt{\sum_y (P(t|y,c_1) - \overline{P(t|y,c_1)})^2}\sqrt{\sum_y (P(t|y,c_2) - \overline{P(t|y,c_2)})^2}}$$

Using this measure we try to analyze how similar topics appear in different conferences over time, whether they follow similar trends or even influence each other.

With an average correlation across all topics between NeurIPS and ACL+ of 0.71, this measure also shows a fairly similar evolution of topics between the conferences overall. We should note however that the topics used in the analysis were generated only from ACL+ papers, so topics exclusive to NeurIPS are not considered. We then rank the topics in our list by the correlation of their evolution in NeurIPS versus ACL+: 5 topics among the top 10 with the most correlated evolution are shown in Table 4.

Neural topics in computational linguistics versus NeurIPS Neural networks are an interesting subset of topics, which have very quickly become very popular in computational linguistics, and are today common as central foci of both ACL+ and NeurIPS. The average correlation between ACL Anthology and NeurIPS for topics related to neural methods (*neural networks*, *RNNs*, *neural MT* and *embeddings*), is 0.58, which interestingly is lower than the overall correlation across all topics.

We try to understand whether these conferences are synchronized in the way they approach topics and hope to understand, by comparing their evolution, if they mutually influence each other, especially regarding topics which are relevant for

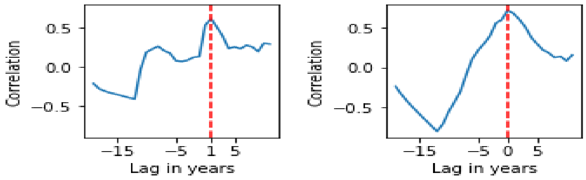

(a) Neural networks topics (b) All topics

Figure 12: Correlation between topics in ACL+ vs. NeurIPS papers adjusted with lags.

both. In order to analyze this phenomenon, we compute the correlation between the evolution of topics, this time introducing an artificial lag for the papers in ACL Anthology. The correlation of topic time series is computed using an updated definition of topic probability:

$$P_l(t|y) = P(t|y+l)$$

where l is a lag factor. Figure 12 shows the correlation between the evolution of topics after applying lags ranging from -25 to 25 years, for the full set of topics, as well as for the subset of topics related to neural networks. If there is any asynchronicity in the way topics appear in the two fields, the lag corresponding to the best correlation should help us find the delay with which topics gain popularity in the two conferences comparatively.

In our case, the optimal lag value across all topics is found to be exactly 0, whereas for neural topics the optimal lag is 1 year, showing a slight delay in the approach of neural method related topics in ACL+. Overall, ACL Anthology and NeurIPS seem fairly synchronized when in comes to innovation in this area.

8 Conclusions

We presented in this article an analysis of the topics found in computational linguistics conferences. We enhanced topics with their types by categorizing topics into tasks, algorithms, and data; and showed how the field has evolved, uncovering general trends, as well as new unforeseen trends such as the abrupt rise of neural network methods. We also identified the most influential authors for each topic, which can provide interesting insights assuming most cited authors when discussing an idea carry a big share of the responsibility of introducing and promoting the idea. A more sophisticated method for identifying influential authors could include a normalization factor based on the number of citations.

We additionally included a study of relations between topics and between subfields, to gain insight into the interplay between topics within and across fields. Our analysis confirmed the strong cooperative relationship between certain tasks and algorithms, such as neural machine translation and recurrent neural networks, but also revealed some interesting less obvious ways in which some topics relate - automatically identifying topics which replace others in the preference of scientists in a subfield (such as the change in paradigm for machine translation). In a separate experiment, we zoom in on the topic of neural networks, and compare the evolution of this topic in computational linguistics conferences to its parallel development in a conference dedicated to neural networks: NeurIPS.

Through the various complementary analyses we performed, we try to contribute to answering the question of how scientific topics emerge and gain traction by considering internal as well as external factors, and the scientific context in which trends appear and evolve. In the future, we would like to explore predictive models of what research topics would gain popularity in upcoming years. It would also be interesting to explore the effect of extracting more fine-grained topics, which could help with identifying more subtle trends - at the technical level, this would involve controlling the level of noise when increasing the number of topics. We will also explore in more depth the properties of the emerging network of topic relations, and the types of topics involved. Exploring more complex topic structures could help model more sophisticated notions such as scientific paradigms.

Acknowledgements

We thank our reviewers for their valuable comments and feedback that helped improve the quality of our paper. This research is supported in part by NSF CAREER award #1802358, NSF CRI award #1823292, and UIC Discovery Partners Institute to Cornelia Caragea. This research is also partially supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI UEFISCDI, project number 108, within PNCDI III. The computation for this project was performed on Amazon Web Services.

Ethical Statement

All our data are extracted from publicly available sources. There are no ethical issues concerning our work.

References

Ashton Anderson, Dan McFarland, and Dan Jurafsky. 2012. Towards a computational history of the acl: 1980-2008. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 13–21. Association for Computational Linguistics.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

Steven Bird, Robert Dale, Bonnie J Dorr, Bryan Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R Radev, and Yee Fan Tan. 2008. The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Levent Bolelli, Şeyda Ertekin, and C Lee Giles. 2009a. Topic and trend detection in text collections using latent dirichlet allocation. In *European Conference on Information Retrieval*, pages 776–780. Springer.

Levent Bolelli, Seyda Ertekin, Ding Zhou, and C Lee Giles. 2009b. Finding topic trends in digital libraries. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 69–72. ACM.

Sujatha Das Gollapalli and Xiaoli Li. 2015. Emnlp versus acl: Analyzing nlp research over time. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2002–2006.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Jonathan Grudin. 2009. Ai and hci: Two fields divided by a common focus. *Ai Magazine*, 30(4):48–48.

David Hall, Daniel Jurafsky, and Christopher D Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the conference on empirical methods in natural language processing*, pages 363–371. Association for Computational Linguistics.

Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and Lee Giles. 2009. Detecting topic evolution in scientific literature: how can citations help? In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 957–966. ACM.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Thomas S Kuhn. 2012. *The structure of scientific revolutions*. University of Chicago press.

Elizabeth A Leicht, Gavin Clarkson, Kerby Shedden, and Mark EJ Newman. 2007. Large-scale structure of time evolving citation networks. *The European Physical Journal B*, 59(1):75–83.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

Seoyeon Park and Cornelia Caragea. 2020. Scientific keyphrase identification and classification by pre-trained language models intermediate task transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5409–5419, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Vinodkumar Prabhakaran, William L Hamilton, Dan McFarland, and Dan Jurafsky. 2016. Predicting the rise and fall of scientific topics from trends in their rhetorical framing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1170–1180.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1862–1868, Portorož, Slovenia. European Language Resources Association (ELRA).

Vahed Qazvinian and Dragomir Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 689–696.

Vahed Qazvinian, Dragomir R Radev, Saif M Mohammad, Bonnie Dorr, David Zajic, Michael Whidby, and Taesun Moon. 2013. Generating extractive summaries of scientific paradigms. *Journal of Artificial Intelligence Research*, 46:165–201.

Dragomir R Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The acl anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944.

Alix Rule, Jean-Philippe Cointet, and Peter S Bearman. 2015. Lexical shifts, substantive changes, and continuity in state of the union discourse, 1790–2014. *Proceedings of the National Academy of Sciences*, 112(35):10837–10844.

Xiaolin Shi, Ramesh Nallapati, Jure Leskovec, Dan McFarland, and Dan Jurafsky. 2010. Who leads whom: Topical lead-lag analysis across corpora. In *NIPS workshop*.

Naoki Shibata, Yuya Kajikawa, Yoshiyuki Takeda, and Katsumori Matsushima. 2008. Detecting emerging research fronts based on topological measures in citation networks of scientific publications. *Technovation*, 28(11):758–775.

Naoki Shibata, Yuya Kajikawa, Yoshiyuki Takeda, and Katsumori Matsushima. 2009. Comparative study on methods of detecting research fronts using different types of citation. *Journal of the American Society for information Science and Technology*, 60(3):571–580.

Chenhao Tan, Dallas Card, and Noah A Smith. 2017. Friendships, rivalries, and trysts: Characterizing relations between ideas in texts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 773–783.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM.

Yi Zhang, Guangquan Zhang, Donghua Zhu, and Jie Lu. 2017. Scientific evolutionary pathways: Identifying and visualizing relationships for scientific topics. *Journal of the Association for Information Science and Technology*, 68(8):1925–1939.

A Full list of topics

Domain adaptation (task)	domain adaptation adapt cross data share weight distribution multi scenario
Automata (algorithm)	string transformation finite operation transducer stre match regular weight symbol
Morphology (task)	morphological arabic morpheme stem suffix morphology prefix root affix inflection
Multi-word expressions (task)	expression collocation literal metaphor idiom mwe multiword descriptor mwes compositional
Sentiment analysis (task)	sentiment negative positive opinion polarity lexicon subjective classification subjectivity neutral
Trees (algorithm)	tree node child root subtree parent forest leaf branch depth
Reinforcement learning (algorithm)	action agent dialog policy reward instruction environment goal human reinforcement
SVMs (algorithm)	kernel svm bag vector bow space reranke linear clue support
Linear programming (algorithm)	constraint solution variable solve inference constrain ilp hard linear soft
Argument mining (task)	claim essay argument stance email evidence support debate statement topic
Topic models (algorithm)	document topic lda collection distribution topical latent content coherence background
Clustering (algorithm)	cluster clustering group induce merge class partition gold induction centroid
Language acquisition (task)	student author learner simplification write native readability grade complex read
Generation (task)	generation generator content record surface realization choice plan selection component
Named entity recognition (task)	token joint ner span crf sequence labeling normalization pipeline crfs
Discourse segmentation (task)	segment segmentation boundary unit length break sequence segmenter segmented window
Events/temporal (task)	temporal anchor event tense expression interval causal date day reference
Phonology (task)	letter phoneme syllable pronunciation phonetic vowel phonological stress consonant sound
Stylistics (task)	emotion social gender age group emotional participant people relationship person
Unification (task)	grammar unification head formalism cat description hpsg sign definition constraint
Language models (task)	gram probability bigram lm perplexity trigram unigram estimate vocabulary smooth
Textual entailment (task)	entailment inference hypothesis game textual player rte premise entail team
Biomedical (task)	cue medical citation abstract patient scientific scope biomedical cite article
Anaphoral/coref. resolution (task)	pronoun mention antecedent coreference resolution coreference_resolution anaphor resolve anaphoric reference
Dependency parsing (algorithm)	dependency parser parse head treebank tree dependent projective arc accuracy
Database/resources (data)	template database logical hybrid variable city expression meaning sql equation
Social media/web data (data)	user response post comment message conversation thread interaction feedback reply
Summarization (task)	summary summarization document rouge compression content length extractive human duc
Spelling correction (task)	error edit correction spelling revision rate confusion preposition incorrect learner
Evaluation/annotation (task)	human metric paraphrase reference correlation quality automatic judgment judge rating annotation annotator annotate agreement annotated gold scheme guideline automatic manual
Semantic role labelling (task)	argument predicate role arg srl syntactic identification propbank labeling core
Discourse (task)	discourse relation coherence connective implicit unit explicit paragraph marker rhetorical
Syntactic structure (task)	noun adjective compound head modifier modifi nominal determiner proper adverb verb subject object class preposition verbal noun passive argument syntactic syntactic linguistic syntax grammatical construction structural lexical deep surface phenomenon
Lexical semantics (task)	similarity vector cosine distributional sim distance weight relatedness space lsa
Learning (algorithm)	weight log parameter objective loss update optimization linear optimize paramet
Probabilistic models/distributions (algorithm)	distribution probability sample variable latent prior parameter estimate inference generative
Statistical MT (task,algorithm)	alignment align link probability ibm aligned null correspondence aligner heuristic translation translate quality mt target statistical translator smt reference bilingual translation bleu reorder decode smt hypothesis side decoder target chinese
Transfer learning (algorithm)	target transfer projection mapping project side map direct ds auxiliary
Speech recognition (task)	speech recognition speaker asr speak utterance acoustic transcript transcription prosodic
POS tagging (task)	tag pos tagger chunk accuracy tagging speech unknown tagset sequence treebank wsj fragment accuracy bracket pcfg probability np penn_treebank head
Lexicons (data)	lexical lexicon item entry lex lexeme coverage associate derive substitution
Constituent parsing (algorithm)	clause constituent head relative coordination subject element position complement mark parse parser grammar chart parsing span tree syntactic stage
Multilinguality (task)	resource french spanish multilingual pivot german corpora italian dutch portuguese

Table 5: Extracted topics and relevant keywords

Unsupervised learning (algorithm)	learning sample supervise unsupervised selection unlabeled iteration active supervised unlabeled
Ranking (algorithm)	candidate rank selection ranking denote weight framework ranker probability combination
Embeddings (algorithm)	vector embedding matrix embed space dimension vec dimensional mikolov tensor
Plan-based dialogue (task,algorithm)	dialogue utterance act speaker plan turn goal belief conversation request
Question answering (task)	question answer passage question_answere match paragraph trec reason factoid relevant
Event extraction (task)	event trigger mention extraction document ace attack argument entity relevant
Grammars (algorithm)	grammar derivation symbol terminal production nonterminal free cfg adjoin string
Logical forms (algorithm)	formula logic interpretation logical scope operator theory proposition predicate expression
Knowledge base (data)	entity mention wikipedia link person kb article document page title
Information extraction (task)	pattern seed extraction acquire acquisition bootstrappe web relationship discover match
Applications (task)	user tool module interface component support file format display design
Disambiguation (task)	interpretation ambiguity ambiguous processing preference strategy disambiguation attachment mechanism heuristic
Graphs/AMR (algorithm)	graph edge node vertex graphs connect amr weight propagation link
Neural networks (algorithm)	network layer neural cnn architecture rnn vector deep hide embedding
Narratives (task)	story genre book expert worker movie narrative human collect crowdsource
Ontologies (algorithm)	concept attribute hierarchy ontology taxonomy conceptual hypernym relation hierarchical link
Prediction (task)	predict prediction accuracy regression predictor error linear predictive variable effect
Quantitative analysis (algorithm)	frequency count probability distribution estimate occurrence corpora association statistical log
Vision/multimodal (task)	image visual video caption multimodal modality fusion textual human modal
Parallel corpora (data)	parallel bilingual monolingual corpora cross_lingual keyphrase comparable translation resource extraction
Neural MT (task,algorithm)	decoder encoder nmt sequence neural attention bleu decode rnn vocabulary
Recurrent neural networks (algorithm)	lstm attention vector memory embed mechanism embedding weight layer encode
Complexity analysis (task)	cost memory speed index fast run bit store key efficient
Opinion mining (task)	review aspect product rating restaurant opinion customer rationale hotel service
Social media (data)	tweet twitter social_media user twitt message hashtag detection post microblog
Transliteration (task)	character chinese transliteration oov hindi unknown accuracy char urdu ctb
Dictionaries (data)	dictionary definition code entry link cod dictionarie analogy database bank
Relation extraction (task)	relation extraction triple relational tuple open express relationship distant_supervision rel
Historical linguistics (task)	change family lemma cognate russian czech linguistic historical distance swedish
Wordnet/disambiguation (task,algorithm)	sense wordnet sens disambiguation synset wsd gloss disambiguate resource ambiguous
Dependency parsing (algorithm)	search transition stack beam prune action shift greedy partial configuration
Information retrieval (algorithm)	query search web retrieval document page relevant retrieve relevance engine
Supertagging (algorithm)	category ccg np derivation supertag composition lexical ambiguity supertagger ccgbank
Asian languages (task)	japanese expression korean bunsetsu particle accuracy wo marker element noun
Classification (algorithm)	class classification classifier accuracy classify svm binary decision classifi combination
Sequence analysis (algorithm)	sequence local position global distance length chain sequential gap permutation
Frame semantics (algorithm)	frame slot schema filler framenet fill intent element role slu
Dynamic programming (algorithm)	path factor ij lattice tuple cache length denote space dynamic
News articles (data)	article news company year political country day people issue market
Scene description (task)	object description property expression $\mu i\_\mu i$ reference scene referent location spatial

Table 6: Extracted topics and relevant keywords – continuation

fig, line, block, cell, column, row, space, region, red, color
tile, ill, tim, ion, el, ed, te, tion, arc, ca
effect, suggest, choice, expect, evidence, discuss, strong, attempt, issue, alternative
une, ce, est, pour, dan, par, les, qui, des, sont
program, element, computer, basic, linguistic, procedure, component, specification, concern, kind
dataset, art, split, accuracy, outperform, benchmark, challenge, bias, setup, tune
, ooo, oooo, , oo, ooooo, uooo, oo, uu, uuuu
precision, recall, match, detection, filter, extraction, threshold, detect, confidence, identification
keyword, title, conference, computational_linguistic, page, proceeding, tutorial, university, year, processing
german, read, incremental, reading, die, prime, processing, der, surprisal, field

Table 7: Excluded topics

B Top relationships

Task	Algorithm
Neural MT	RNNs
Reinforcement Learning	Plan-based Dialogue
Deep Learning	Neural MT
Unification	Grammars
Finite State Machines	Phonology
Plan-based Dialogue	Scene Description
Unification	Logical Forms
Semantic Role Labelling	Frame Semantics
Topic Models	Summarization
Discourse	Plan-based Dialogue

Table 8: Strongest friendship relations (task-algo).

Algorithm	Data
Statistical MT	Parallel Corpora
Transfer Learning	Parallel Corpora
Ontologies	Dictionaries
Reinforcement Learning	Social Media
Combinatory Categorical Grammar	Lexicons
Grammars	Lexicons
Plan-based Dialogue	Social Media
News Articles	Topic Models
Graphs	Knowledge Base
Constituent Parsing	Lexicons

Table 9: Strongest friendship relations (algo-data).

Task	Data
Multilinguality	Parallel Corpora
Stylistics	Social Media
Statistical MT	Parallel Corpora
Phonology	Dictionaries
Argument Mining	Social Media
Transliteration	Parallel Corpora
Multi-Word Expressions	Dictionaries
Unification	Lexicons
Named Entity Recognition	Knowledge Base
Morphology	Dictionaries
Opinion Mining	Social Media

Table 10: Strongest friendship relations (task-data).

Task	Task
Phonology	Semantic Role Labelling
Morphology	Discourse
Phonology	Anaphora/Coref. Resolution
Phonology	Relation Extraction
Phonology	Discourse
Discourse Segmentation	WordNet/Disambiguation
Events/temporal	Phonology
Speech Recognition	WordNet/Disambiguation
Statistical MT	Relation Extraction
Speech Recognition	Relation Extraction

Table 11: Strongest arms-race relations (task-task).

Algorithm	Algorithm
Topic Models	Dependency Parsing
Wordnet/Disambiguation	Dependency Parsing
Finite State Machines	Plan-based Dialogue
Wordnet/Disambiguation	Sequence Analysis
Topic Models	Statistical MT
Finite State Machines	Frame Semantics
Clustering	Statistical MT
Finite State Machines	Ontologies
Trees	Wordnet/Disambiguation
Statistical MT	Wordnet/Disambiguation

Table 12: Strongest arms-race relations (algo-algo).

Data	Data
Knowledge Base	Parallel Corpora
Lexicons	News Articles

Table 13: Strongest arms-race relations (data-data).

Task	Task
Unification	Neural MT
Disambiguation	Neural MT
Unification	Vector Spaces
Named Entity Recognition	Unification
Neural MT	Wordnet/Disambiguation
Sentiment Analysis	Unification
Unification	Summarization
Unification	Prediction
Unification	Language Models
Anaphora/Coreference Resolution	Neural MT

Table 14: Strongest head-to-head relations (task-task).

Algorithm	Algorithm
Grammars	Neural MT
Grammars	RNNs
Ontologies	Neural MT
Logical Forms	Neural MT
Neural MT	Wordnet/Disambiguation
Neural MT	Combinatory Categorical Grammar
Constituent Parsing	Neural MT
Finite State Machines	RNNs
Logical Forms	RNNs
Grammars	Deep Learning

Table 15: Strongest head-to-head relations (algo-algo).

Algorithm	Algorithm
Grammars	Neural MT
Grammars	RNNs
Ontologies	Neural MT
Logical Forms	Neural MT
Neural MT	Wordnet/Disambiguation
Neural MT	Combinatory Categorical Grammar
Constituent Parsing	Neural MT
Finite State Machines	RNNs
Logical Forms	RNNs
Grammars	Deep Learning

Table 16: Strongest head-to-head relations (algo-algo).

Data	Data
Lexicons	Knowledge Base
Knowledge Base	Dictionaries
Ontologies	Neural MT
Social Media	Parallel Corpora
Social Media	Lexicons

Table 17: Strongest head-to-head relations (data-data).

Algorithm	Algorithm
Reinforcement Learning	Neural MT
Statistical MT	Neural MT
Transfer Learning	Neural MT
Reinforcement Learning	RNNs
Dependency Parsing	Constituent Parsing
Neural MT	Dependency Parsing
RNNs	Sequence Analysis
Learning	Neural MT
Probabilistic	Neural MT
Clustering	Ontologies

Table 18: Strongest tryst relations (algo-algo).

Task	Task
Generation	Neural MT
Generation	Summarization
Statistical MT	Neural MT
Summarization	Neural MT
Vision/Multimodal	Scene Description
Phonology	Language Models
Neural MT	Transliteration
Summarization	Discourse
Textual Entailment	Vector Space
Summarization	Event Extraction

Table 19: Strongest tryst relations (task-task).

Data	Data
Dictionaries	Parallel Corpora
Lexicons	Parallel Corpora

Table 20: Strongest tryst relations (data-data).

End-to-End Self-Debiasing Framework for Robust NLU Training

Abbas Ghaddar, Philippe Langlais[†], Mehdi Rezagholizadeh and Ahmad Rashid
Huawei Noah's Ark Lab, Montreal Research Center, Canada
[†]RALI-DIRO, Université de Montréal, Montréal, Canada
abbas.ghaddar@huawei.com, felipe@iro.umontreal.ca
mehdi.rezagholizadeh@huawei.com, ahmad.rashid@huawei.com

Abstract

Existing Natural Language Understanding (NLU) models have been shown to incorporate dataset biases leading to strong performance on in-distribution (ID) test sets but poor performance on out-of-distribution (OOD) ones. We introduce a simple yet effective debiasing framework whereby the shallow representations of the main model are used to derive a bias model and both models are trained simultaneously. We demonstrate on three well studied NLU tasks that despite its simplicity, our method leads to competitive OOD results. It significantly outperforms other debiasing approaches on two tasks, while still delivering high in-distribution performance.

1 Introduction

Researchers have increasingly raised concerns about the tendency of recent NLU models (Devlin et al., 2019; Liu et al., 2019) to quickly leverage spurious surface lexical-syntactic features (Poliak et al., 2018; Gururangan et al., 2018; Dasgupta et al., 2018; Ghaddar and Langlais, 2017). These superficial properties, also referred as dataset biases (Shah et al., 2020; Utama et al., 2020b; Moosavi et al., 2020), result in significant performance drop on out-of-distribution (OOD) sets containing counterexamples to biases in the training data (McCoy et al., 2019; Schuster et al., 2019; Zhang et al., 2019; Ghaddar et al., 2021).

The most common approach to tackle the problem consists in training a *bias model* with hand-crafted features with the goal of identifying biased training examples. This information is used in a later stage to discourage the *main model* from adopting the naive strategy of the bias model. Several debiasing training paradigms have been proposed to adjust the importance of biased training samples, such as product of experts (Clark et al., 2019; He et al., 2019), learned-mixin (Clark et al., 2019), example reweighting (Schuster et al., 2019), debiased focal loss (Mahabadi et al., 2020), and confidence regularization (Utama et al., 2020a).

Recently, there has been a number of endeavours to produce a bias model without prior knowledge on the targeted biases or without the need for manually designing features. Utama et al. (2020b) propose to use instead a model trained on a tiny fraction ($< 1\%$) of the training data for few epochs as a bias model; while Clark et al. (2020) and Sanh et al. (2020) trained a low capacity model on the full training set. These approaches target the training of the bias model alone, which is subsequently queried while training the main model of interest.

In this paper, we propose an end-to-end debiasing framework which does not require an extra training stage, or manual bias features engineering. The bias model is indeed a simple attention-based classification layer on top of the main model's intermediate representations. Both models are trained simultaneously in an end-to-end manner as in (Mahabadi et al., 2020), where the importance of training samples for both models are adjusted using the example reweighting technique of (Schuster et al., 2019).[1] The idea of using intermediate classifiers has previously been explored to reduce the inference cost (Schwartz et al., 2020; Zhou et al., 2020; Xin et al., 2021) for Transformer-based (Vaswani et al., 2017) models.

In contrast to all previous works, our bias model helps locating lexico-syntactic bias features, inside the main model's intermediate layers, whose importance is reduced by adding a noise vector; therefore preventing the main model to rely on them. During training, both the main and bias models interact by interchangeably re-weighting the importance of each others' examples.

Our learning framework, when applied to a

---

[1] The bias model for these two works is based on hand-crafted features.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1923–1929
August 1–6, 2021.

vanilla BERT (Devlin et al., 2019) model leads to consistent and significant improvements on 3 NLU tasks, while maintaining a balanced performance between ID and OOD sets. It involves a single training stage, and only incurs a small number of extra parameters (0.5M) compared to other approaches. For instance, Utama et al. (2020a) used a copy of the main model (110M parameters) as the bias model, while (Sanh et al., 2020) used BERT-Tiny (Turc et al., 2019) that has 11M parameters and has been pre-trained from scratch using the masked LM objective (Devlin et al., 2019).

2 Method

We describe an end-to-end solution for debiasing a Transformer-based classification model. We note $x = \{x_1, x_2, \ldots, x_n\}$ the input sequence of length n, and $y \in \{1, 2, \ldots, T\}$ the gold label, where T is the number of classes. Figure 1 shows a diagram of our method.

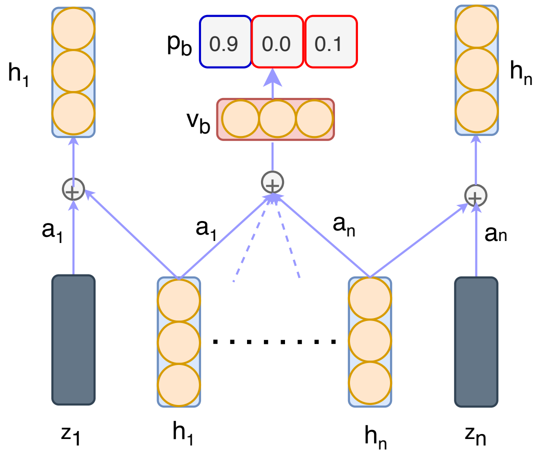

Figure 1: Illustration of our framework where the bias model is placed at the top of the k^{th} layer of the main model. Notation is introduced in the text, for simplicity we use h_i to refer to h_i^k. The blue rectangle indicates the index of of the ground truth class.

2.1 Main Model

The main model is a Transformer-based BERT encoder (Vaswani et al., 2017; Devlin et al., 2019) with a classifier on top of the classification `[CLS]` token of the last layer. This classifier generates the probability distribution $p_m \in \mathbb{R}^T$ over the output classes given an input x. At each layer k, BERT produces an internal hidden representation $h_i^k \in \mathbb{R}^d$ for each token x_i in the input.

2.2 Bias Model

We hypothesize that the bottom layers of the main model can serve as input for the bias model, thus avoiding the need for an external model or for manually designed features. Our intuition is built on the observation made by Jawahar et al. (2019) that bottom layers of BERT mainly encode lexical and syntactic information, and on the fact that such models tend to quickly overfit this type of information (Zellers et al., 2018; McCoy et al., 2019).

Our bias model is composed of an additive attention layer (Bahdanau et al., 2014), followed by a softmax one. This module uses the k^{th} layer of the main model as input in order to produce a probability distribution over classes. First, a scalar value $\tilde{a}_i \in \mathbb{R}$ is computed using a feed forward neural network with weight matrices $W_e \in \mathbb{R}^{d \times d}$ and $W_a \in \mathbb{R}^{1 \times d}$ such that:

$$\tilde{a}_i = W_a \tanh\left(W_e h_i^k\right) \quad (1)$$

Those scalar values are normalized and referred to as attentions a_i:

$$a_i = \frac{\exp(\tilde{a}_i)}{\sum_{i=1}^{n} \exp(\tilde{a}_i)} \quad (2)$$

The representation vector, v_b, for the bias model is computed as the weighted sum of the intermediate representations of the k^{th} layer:

$$v_b = \sum_{i=1}^{n} a_i h_i^k \quad (3)$$

We obtain the probability distribution p_b over the classes by applying a projection layer (parameterized by $W_y \in \mathbb{R}^{d \times T}$) followed by a softmax function: $p_b = \text{softmax}(W_y^\top v_b)$.

2.3 Biased Features Regularization

The attention weights of the bias model reveal which hidden representations are the most informative for the classification decision of that model. We propose to de-emphasize these "bias features" from the main model. We do so by adding a weighted noise vector to the hidden representations of the main model at layer k. Let $z \leftarrow \{z_1, \ldots, z_n\}$ be a set of zero-mean Gaussian noise vectors corresponding to each token in the sequence, where $z_i \in \mathbb{R}^d$. The intermediate representations of the main model at layer k are updated as follows:

$$h_i^k \leftarrow a_i z_i + h_i^k \quad (4)$$

where a_i determines the relative amplitude of noise added to each token representation. A high value of a_i means that h_i^k most likely contains bias features. Consequently, a large amount of noise is added at this position, which will prevent the main model from overfitting to these features at subsequent layers.

2.4 Example Re-weighting

For both models, we adjust the importance of a training sample by directly assigning it a scalar weight that indicates whether the sample exhibits a bias or not. Let p_m^c and p_b^c be the predicted probabilities corresponding to the ground-truth class by the bias and main model respectively. The weight assigned to the main model training sample is calculated as follows:

$$w_m = \begin{cases} 1 - p_b^c & p_b^c > \gamma \\ 1 & \text{otherwise} \end{cases}$$

Differently from the reweighing method of (Clark et al., 2019; Schuster et al., 2019), we add a hard threshold $\gamma \in [0, 1]$ to control the number of samples to be re-weighted. The goal is to ensure that the importance of main model samples will be attenuated only if the bias model predictions fall into the high confidence bin ($\gamma > 0.8$) on biased samples. To further strengthen this constraint, we down-weight the importance of bias model training samples if the main model confidence is below a threshold β:

$$w_b = \begin{cases} p_m^c & p_m^c < \beta \\ 1 & \text{otherwise} \end{cases}$$

This ensures that the bias model is focusing on easy examples at early training stages, while challenging ones are gradually fed in the later training steps. Since the main and bias models are trained simultaneously, they interchangeably re-weight each others' examples. This is different from previous works where all samples have the same importance during the training phase of the bias model. For a single training instance, the individual loss term for the main and bias models are:

$$\mathcal{L}(\theta_B) = -w_b log(p_b^c) \tag{5}$$

$$\mathcal{L}(\theta_M) = -w_m log(p_m^c) \tag{6}$$

where $\theta_B = \{W_e, W_a, W_y\}$ are the trainable parameters for the bias model, while θ_M are the ones for the main model (BERT parameters). We train both models by minimizing the aforementioned losses. At inference time, only the main model is used for prediction, and no noise is added to h^k. It is worth mentioning that our proposed extension has no impact on the overall training time.

3 Experiments

3.1 Datasets

We test our method on 3 sentence-pair classification tasks supported by ID training and validation sets as well as an OOD test set, which are specifically built to measure robustness to dataset bias.

For the **Natural Language Inference** task, we use the MNLI (Williams et al., 2017) benchmark as our ID data, and HANS (McCoy et al., 2019) as our OOD test set. For **Fact Verification**, we use the FEVER (Thorne et al., 2018) benchmark for ID evaluation, and FEVER Symmetric (Schuster et al., 2019) (version 1) as our OOD test set. For **Paraphrase identification**, we use QQP [2] as our ID data, and PAWS (Zhang et al., 2019) as our OOD test set.

3.2 Implementation

We use the 12-layer BERT-base model (Devlin et al., 2019) as our main model, thus our results can be compared with prior works. We adopt the standard setup of BERT and represent a pair of sentences as: `[CLS]` 1st sentence `[SEP]` 2nd sentence `[SEP]`. For BERT hyper-parameters, we use those of the baseline: a batch size of 64, learning rate of 1e-5 with the Adam (Kingma and Ba, 2014) optimizer.

Following (Clark et al., 2019; Grand and Belinkov, 2019; Clark et al., 2020; Sanh et al., 2020), we tune our method hyperparameters on the OOD sets. As pointed out by (Clark et al., 2019, 2020), this is not ideal since it assumes some prior knowledge of the OOD test sets. To best mitigate this impact, we follow the procedure of previous works and use the same hyper-parameters for all 3 tasks. We varied k ($[2-5]$), γ ($[0.7-0.9]$), β ($[0.5-0.7]$), and fix their values to 3, 0.8, 0.5 respectively.

However, different set of parameters performed roughly equally well, provided that k is 3 or 4, $\gamma \geq 0.8$ and β is set to 0.5. We use early stopping and report mean performance and standard deviation over 6 runs with different seeds.

---

[2] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question`

Model	MNLI		FEVER		QQP	
	Dev	HANS	Dev	Symm.	Dev	PAWS
BERT-base	84.5	62.4	85.6	55.1	**91.0**	33.5
(Clark et al., 2019)	83.5	69.2	-	-	-	-
(Schuster et al., 2019)	-	-	85.4	61.7	-	-
(Utama et al., 2020a)	**84.5**	69.1	86.4	60.5	89.1	40.0
(Utama et al., 2020b)†	82.3	69.7	-	-	85.2	**57.4**
(Utama et al., 2020b)‡	84.3	67.1	-	-	89.0	43.0
(Sanh et al., 2020)♠	81.4	68.8	82.0	60.0	-	-
(Sanh et al., 2020)♣	83.3	67.9	85.3	57.9	-	-
this work	83.2±0.1	**71.2**±0.2	**86.9** ±0.8	**63.8±0.3**	90.2±0.2	46.5±2.3
w/o noise	83.7±0.3	68.6±0.9	85.5±0.5	61.6±0.6	90.4±0.3	42.4±2.1
w/o main reweighting	84.0±0.4	62.8±1.1	85.1±0.4	57.4 ±0.8	91.0±0.2	36.7±1.6
w/o bias reweighting	81.5±0.9	64.6±0.6	83.9±0.4	60.3±0.5	89.3±0.7	39.7±2.4

Table 1: Model performance when evaluated on MNLI, Fever, QQP, and their corresponding challenge test sets. Ablation study of our method without adding noise, and without reweighting main or bias model's training samples respectively. Symbols are used to distinguish variants from the same paper that use a different training technique: (†) example reweighting, (‡) confidence regularization, (♠) product of Experts (PoE), (♣) PoE+ cross-entropy.

3.3 Results

Table 1 reports accuracy scores on the development and OOD test sets of the 3 benchmarks we considered. The baseline is the vanilla BERT-base model which is used as a backbone for the main model in all the configurations reported. Also, the table shows the ablation results of our method without adding noise, and without reweighting the training instances of the bias and main models (setting β and γ to 0 or 1 respectively).

First, we notice high variance in performances between runs in the debiasing setting, which was also reported in (Clark et al., 2019; Mahabadi et al., 2020; Utama et al., 2020b; Sanh et al., 2020). Second, we observe that our method offers a good balance between gains on OOD test sets over the baseline, and losses on ID sets. More precisely, we report the best results on FEVER (both ID and OOD test sets), while we improve the HANS score on the MNLI task, but fail to maintain the baseline score on dev as Utama et al. (2020a) did.

On QQP, Utama et al. (2020b) reported a much higher score on PAWS (57.4% vs. 46.5%), but at the expense of an important drop on the ID dev set (85.2% vs. 90.2%). However, our method outperforms this particular model on both MNLI sets, and also shows better cross-task performances compared to prior works. The results are satisfactory, especially when considering the simplicity and efficiency of our approach. Moreover, the fact that a single configuration works well on 3 tasks is an indicator that our method has the potential to generalize on completely unknown OOD sets (Clark et al., 2020).

Expectedly, deactivating the main model reweighting mechanism results in near baseline performances. Solely adding noise signal leads to a modest gain of 2-3% on the OOD test sets and a slight drop ($< 1\%$) on the dev sets compared to the baseline. On one hand, without adding noise, our scores are comparable with previous works across the 3 tasks, that is, a significant drop on OOD test sets and minor gains on ID ones. These observations suggest that down-weighting biased examples is important, while de-emphasizing bias features further improves robustness.

Task	Acc.		$p_b^c > \gamma$	
	w	w/o	w	w/o
MNLI	67.9	69.7	18%	23%
FEVER	70.8	85.7	16%	28%
QQP	87.6	92.9	30%	44%

Table 2: Bias model training accuracy and percentage of training samples correctly classified with high probability, with and without reweighting bias model examples.

On the other hand, we observe that not reweighting bias model examples results in the worst performance on the ID sets. We conduct an analysis on the bias model to better understand the impact of reweighting its examples. As shown in Table 2, its training accuracy as well as the percentage of high confidence predictions increases when examples are not re-weighted, further down-weighting the main model's training examples, which leads to a significant drop of both the ID and OOD performances for all 3 tasks.

a) **GT:** *C*; **Main:** 0.81; **Bias (w/o):** 0.99 (0.94)
H Yes, sir.
P No, not in particular Sir.

b) **GT:** *E*; **Main:** 0.74; **Bias (w/o):** 0.95 (0.92)
H right after the war
P Just after the war ended.

c) **GT:** *C*; **Main:** 0.67 ; **Bias (w/o):** 0.0 (0.77)
H well his knees were bothering him yeah
P He was in tip-top condition.

d) **GT:** *C*; **Main:** 0.39 ; **Bias (w/o):** 0.1 (0.89)
H Even us if you needed," said John.
P He told them not to ask him to lift a finger.

e) **GT:** *N*; **Main:** 0.72 ; **Bias (w/o):** 0.1 (0.92)
H What changed?
P What was unique?

Figure 2: Ground Truth (GT) class probabilities of the main and bias models on examples from the MNLI dev set. Examples consist of an **H**ypothesis and a **P**remise, and valid labels are: **E**ntailment, **C**ontradiction, and **N**eutral. Probabilities of the bias model without reweighting its examples are placed within parenthesis.

We inspected the confident scores of the main and bias (with and without example reweighting) models on MNLI dev set, an excerpt of which is reported in Figure 2. On one hand, we observe that bias models successfully assign high probabilities to samples that can be easily classified via keywords (e.g. ”not” in example a) and those with high lexical overlap (example b). On the other hand, we noticed that the bias model is performs undesirably well on some challenging examples like (c) and (d) of Figure 2. However, the bias model probability always decreases when we re-weight its examples, which eventually leads to correct prediction of the main model as in example (c) and (e). Interestingly, this observation suggests that training a pure bias model is as important (and challenging) as training the robust one.

4 Conclusion

Our key contribution is a framework that jointly identifies biased examples and features in an end-to-end manner. Our approach is geared towards addressing lexico-syntactic bias features for Transformer-based NLU models. Future work involves testing our approach on other tasks such as Question Answering (Rajpurkar et al., 2016; Agrawal et al., 2018), exploring methods to obtain proxy OOD data for hyper-parameters selection, and making our method hyper-parameter free.

Acknowledgments

We thank Mindspore[3] for the partial support of this work, which is a new deep learning computing framework.

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4971–4980.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4060–4073.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2020. Learning to model and ignore dataset bias with mixed capacity ensembles. arXiv preprint arXiv:2011.03856.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. 2018. Evaluating compositionality in sentence embeddings. arXiv preprint arXiv:1802.04302.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference

---

[3] `https://www.mindspore.cn/`

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Abbas Ghaddar, Philippe Langlais, Ahmad Rashid, and Mehdi Rezagholizadeh. 2021. Context-aware Adversarial Training for Name Regularity Bias in Named Entity Recognition. Transactions of the Association for Computational Linguistics, 9.

Abbas Ghaddar and Phillippe Langlais. 2017. Winer: A wikipedia annotated corpus for named entity recognition. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 413–422.

Gabriel Grand and Yonatan Belinkov. 2019. Adversarial regularization for visual question answering: Strengths, shortcomings, and side effects. In Proceedings of the Second Workshop on Shortcomings in Vision and Language, pages 1–13.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 107–112.

He He, Sheng Zha, and Haohan Wang. 2019. Unlearn dataset bias in natural language inference by fitting the residual. EMNLP-IJCNLP 2019, page 132.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3651–3657.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-end bias mitigation by modelling biases in corpora. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8706–8716. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3428–3448.

Nafise Sadat Moosavi, Marcel de Boer, Prasetya Ajie Utama, and Iryna Gurevych. 2020. Improving robustness by augmenting training sentences with predicate-argument structures. arXiv preprint arXiv:2010.12510.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, pages 180–191.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2383–2392.

Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. 2020. Learning from others' mistakes: Avoiding dataset biases without modeling them. arXiv preprint arXiv:2012.01300.

Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. 2019. Towards debiasing fact verification models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3410–3416.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020. The right tool for the job: Matching model and instance complexities. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6640–6651.

Deven Santosh Shah, H Andrew Schwartz, and Dirk Hovy. 2020. Predictive biases in natural language processing models: A conceptual framework and overview. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 809–819.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. arXiv preprint arXiv:1908.08962.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020a. Mind the trade-off: Debiasing nlu models without degrading the in-distribution performance. arXiv preprint arXiv:2005.00315.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020b. Towards debiasing nlu models from unknown biases. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7597–7610.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 91–104.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 93–104.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 1298–1308.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. Advances in Neural Information Processing Systems, 33.

A Mixed-Method Design Approach for Empirically Based Selection of Unbiased Data Annotators

Gautam Thakur **Janna Caspersen** **Drahomira Herrmannova**
Bryan Eaton **Jordan Burdette**
Oak Ridge National Laboratory
1 Bethel Valley Road
Oak Ridge, TN 37830
{thakurg,caspersenjr,herrmannovad,eatonbm,burdetteja}@ornl.gov

Abstract

Implicit bias embedded in the annotated data is by far the greatest impediment in the effectual use of supervised machine learning models in tasks involving race, ethics, and geopolitical polarization. For societal good and demonstrable positive impact on wider society, it is paramount to carefully select data annotators and rigorously validate the annotation process. Current approaches to selecting annotators are not sufficiently grounded in scientific principles and are limited at the policy-guidance level, thereby rendering them unusable for machine learning practitioners. This work proposes a new approach based on the mixed-methods design that is functional, adaptable, and simpler to implement in selecting unbiased annotators for any machine learning problem. By demonstrating it on a real-world geopolitical problem, we also identified and ranked key inane profile characteristics towards an empirically-based selection of unbiased data annotators.

1 Introduction

Human annotation is crucial for many supervised natural language processing problems. Because judgments of meaning can be subjective and vary depending on age, knowledge, intuition, etc. A number of previous works have studied the quality and reliability of manually generated annotations (Snow et al., 2008; Bhardwaj et al., 2010; Aker et al., 2012; Peldszus and Stede, 2013). What these works have in common is that they typically compare annotators from the perspective of agreement, i.e., the degree to which different annotators produce the same labels.

In this work, we study whether and to what extent inherent annotator beliefs and biases affect their answers in labeling tasks. We design a labeling task focused on inflammatory language surrounding Brexit and as part of this task we presented to participants several surveys developed to measure their attitudes towards issues such as race and religion, specifically the Modern Racism Scale (MRS) (McConahay, 1986) and the Moral Foundations Questionnaire (MFQ) (Davies et al., 2014), and surveys capturing their knowledge, demographics, and other relevant information. This labeling task was presented to hired Amazon Mechanical Turk workers (over 100 turkers were hired from 26 countries) in addition to a Subject Matter Expert (SME) who provided ground truth labels. We find that specific results on the Modern Racism Scale and the Moral Foundations Questionnaire are correlated with specific levels of agreement with the subject matter expert, while the same is not the case for knowledge of the topic. Furthermore, we show that the most accurate annotators share similar beliefs as measured by the MRS and MFQ questionnaires. We believe our findings can inform the selection of annotators for such difficult annotation tasks as inflammatory and hateful language detection. To the best of our knowledge, the proposed work is the first to study the relation between annotator belief and biases and the accuracy of their labels.

To that end, this work focuses on a much-needed approach to the systematic selection of unbiased annotators towards improving the outcomes and realism of machine learning decisions. This research investigates and attempts to answer these questions: (i) To what extent do domain expertise and/or psychological bias impact the quality of qualitative data annotations to develop labeled data for machine learning classification? (ii) Is there a significant difference between how individuals label training data? (iii) What are the commonalities between participants whose annotations best matched the ground-truthed dataset? iv) To what extent do those variations in data annotation impact the resulting automated machine learning classification?

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1930–1938
August 1–6, 2021.

(v) How do we develop a profile that quantifies the implicit deviations within various machine learning models?

While benchmarks exist to validate the performance of ML in such scenarios, albeit the degree to which data annotators affect ML outcomes for such nuanced language patterns is unclear, nor certain is the presence of a systematic framework to discover implicit biases among the annotators that might have been responsible. For instance, in (Sap et al., 2019) authors investigated the presence of racial bias in automatic hate speech detection models, racial bias in an algorithm used to manage the health of populations was discovered in (Obermeyer et al., 2019; Caliskan et al., 2017). Besides, there exists several documented evidences of how machine learning-driven automation has resulted in catastrophic dangers to human security (Brown et al., 2006; West et al., 2019; Buolamwini and Gebru, 2018; Hamidi et al., 2018; Noble, 2018), infrastructure resiliency (Osoba and Welser, 2017; Vladeck, 2014), education (Pedro et al., 2019), and economy (Anderson, 2019; Furman and Seamans, 2019). Several approaches are proposed to counter them, including the development of guidelines and policies surrounding the ethical use of machine learning (Wiens et al., 2019), platforms and toolboxes for diagnosing biases (Brundage et al., 2020), recommended practices, and frameworks for evaluating the fairness and explainability (Hardt et al., 2016; Beutel et al., 2019; Pleiss et al., 2017; Gunning, 2017). In (Sheng et al., 2008) authors suggested when labeling is not perfect, selective acquisition of multiple labels is a strategy that data miners should consider. The purpose of this research is to model the extent to which professional background and mental biases affect analytic results from ML algorithms that are specifically applied to complex social media analysis.

We propose *Mixed-Method Design (MMD)* as a new approach towards selecting unbiased data annotators. Broadly, MMD is a method that focuses on collecting, analyzing, and mixing both quantitative and qualitative data in a single study or series of studies. Its central premise is that the use of quantitative and qualitative approaches, in combination, provides a better understanding of research problems than either approach alone (Creswell and Clark, 2017; Shorten and Smith, 2017). In this work, the qualitative work involves conducting surveys and questionnaires on modern racism, moral foundation, demographics among 100 participants who will also label the data. Using Subject Matter Expert (SME) as a baseline, we built a suite of supervised machine learning models from the labeled data and compared the performance of all participants against the SME. Finally, using statistical analysis, we identified key profile characteristics of data labelers that played an important role in how they labeled the data. Our main contributions are:

- Develop a generalized framework for selecting unbiased data annotators for problems surrounding inflammatory language, hate speech, and others, and requiring labeled data.
- Develop an open-source web-based platform[1] and deploy it to perform mixed-method design.
- Design and develop privacy-preserving and problem-agnostic qualitative surveys and questionnaire towards discovering implicit biases among potential data labelers.
- Identify a subset of key profile characteristics that plays a critical role identifying reliable annotators.

2 Mixed Method Design

The Mixed-method design workflow is shown in Figure 1 to selecting data labelers for complex tasks that involve risks of bias and ethics violation. The rest of the section discusses each step in more detail.

2.1 Labeling Task (Study)

This study focuses on Brexit 2016 referendum as a narrative to investigate implicit biases among the data labelers. Since Britain split from the European Union and changed its relationship to the bloc on trade, security and migration was both welcomed and denigrated and polarized the UK/EU. In the past, several opinion polls and surveys revealed a remarkable divide between generations and demographics within and outside of UK and EU countries.

For our study, we selected a set of 10,000 English tweets related to Brexit between January 2019 and October 2019 which was narrowed down to 2,000 tweets for ground truth annotation and to 250 tweets for annotation by Amazon Mechanical

---

[1] `https://thirdeye.ornl.gov`

Figure 1: Mixed-Method Design Workflow

Turk workers.The labeling task primarily focused on identifying whether a given tweet contains inflammatory language, i.e., language that is intended to arouse anger, create and encourage disorder, provoke violent feelings, or excite strong feelings for or against something or someone. Additionally, the participants were asked to answer the following questions about each tweet: 1) Does this text attempt to advocate for violence, hatred, discrimination, or a specific policy? 2) Does this text express a problem with a specific characteristic of an 'other'? 3) Does this text contain a propaganda device? Before starting the labeling assignment all participants were asked to take a short training that provided definitions and explanations for all four label categories.

2.2 Ground Truth Label Generation

First, a Brexit Subject Matter Expert (SME) was hired to generate ground truth labels for the collected tweets. In addition to Brexit knowledge, the SME has previous experiences examining online communication. The study also assessed SME's geo-political and demographic background. The hired SME spent a significant amount of time in Britain and was well versed with the geo-political polarization in the region. However, the SME was not a British citizen allowing an unbiased perspective. In addition to the SME, each tweet was labeled by a social scientist with expertise in inflammatory language and online propaganda. Obtaining labels from two experts was done to allow us to assess the general difficulty of the labeling task.

2.3 Participant Selection

Next, Amazon Mechanical Turk was used to hire over 100 data annotators from selected countries around the world. Figure 2 shows the distribution of participants. On average each participant took three weeks to complete the study and data labeling process. By analyzing time to label each tweet and the generated labels we detected several participants who were providing random labels or who always provided the same answer. These were removed from the study.

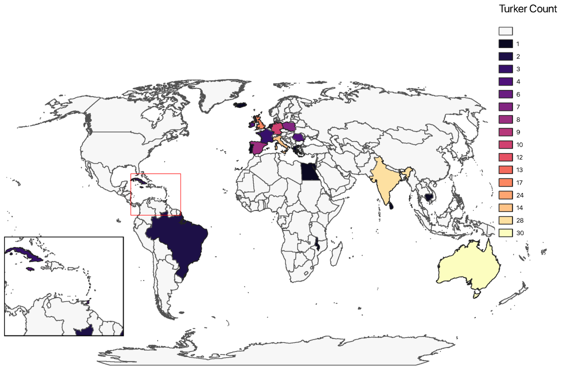

Figure 2: Data annotators were hired from all over the world from majority of the English speaking countries.

2.4 Survey

Both SME and the participants have to complete the following surveys before beginning the data labeling process. The surveys are designed to capture implicit biases.

2.4.1 Symbolic Racism 2000 Scale

Racist viewpoints are a key demographic of inflammatory hate speech. The SR2K is a widely used tool and is considered a reliable method for measuring an individual's racism (Sears and Henry, 2005). The questions within this scale measure contemporary attitudes (Henry and Sears, 2002). The SR2K was incorporated in this research because the participants are labeling data associated with racial ideologies and understanding how they feel about those ideologies is key to explaining their classification choices.

2.4.2 Moral Foundations Questionnaire

The Moral Foundations Questionnaire (MFQ) is considered to be a reliable measure of moral interests (Graham et al., 2011). The MFQ was developed based on the Moral Foundation Theory in 2008 and we used the 2011 version in this research. The MFQ was selected for use in this research because it allows us to assess participants' political leanings and their general moral preferences.

2.4.3 Knowledge Survey

The participants were given a 10 question knowledge survey that was designed to test participants' previous awareness of history associated with inflammatory speech and the Civil Rights Movement. This information is key to understanding participants' prior awareness of historic issues. The insights gained from this section will be used to compare participants' ability to identify inflammatory speech.

2.4.4 Demographics Survey

The demographic survey explored general information about the participants. The demographics collected included age, ethnicity, gender, education, marital status, employment, income, nationality, and political affiliation.

2.4.5 Social Media Background Survey

To better understand the participants' prior experience with social media the final section of the pre-survey had ten questions to address those experiences. The first four question of this section were meant to test their awareness of the most popular social media platforms. The following four questions were meant to generate a deeper understanding of the participants' personal interactions with social media platforms. The final two questions were open ended and focused on participants' profession/career, so that an understanding of their domain expertise may be gained without directly asking, as responses to a direct question on expertise may result in unreliable and exaggerated or understated responses.

2.5 Platform Development for Annotation

A interactive web-based application[2] is developed to help with the data labeling process of tweets related to Brexit. Besides, a training video with set of instructions are designed to guide users to label the data properly.

The remainder of the paper discusses the qualitative and quantitative analysis, as well as identification and ranking of characteristics indicative of a reliable data annotator.

3 Data Analysis

In this section we present the analysis of the collected data. We measure the agreement of each annotator with the SME and study the relation between the agreement and the annotator characteristics measured by the surveys. Table 1 shows the distribution of ground truth labels provided by the SME.

Table 1: Statistics of our ground truth dataset.

	True	**False**
Inflammatory	128	122
Problematization	49	201
Advocation	66	184
Propaganda	150	100

3.1 Agreement with Ground Truth

To measure the agreement of each annotator with the SME, we use Matthews Correlation Coefficient (MCC)MCC ranges from -1 (perfect disagreement) to 1 (perfect agreement) and can be calculated from confusion matrix using the following equation:

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{\mathrm{R}_{+} \times \mathrm{R}_{-} \times \mathrm{P}_{+} \times \mathrm{P}_{-}}}, \tag{1}$$

where R+ represents the sum of true positives and false negatives, R- represents the sum of false positives and true negatives, P+ represents the sum of true positives and false positives, and P- represents the sum of false negatives and true negatives. Figure 3 shows the range of MCC values for each label category. In the figure, each bar represents one annotator and the y-axis represents the MCC score. The orange bars represent labels generated by the social scientist with expertise in online inflammatory language.

In all four plots the y-axis range is -0.2 to 0.9 to allow easier comparison of results between the different categories. An interesting observation is that some label categories seem to have a higher disagreement with the ground truth. For example, this is the case for advocates violence category in the bottom left corner which tends to have lower MCC values than the other three categories. A possible explanation is that different people perceive this specific narrative category differently, while the other categories tend to be perceived more similarly.

3.2 Analysis of Survey Results

We used box plots to compare each individual survey with GT agreement. Specifically, annotators

---

[2] `https://thirdeye.ornl.gov`

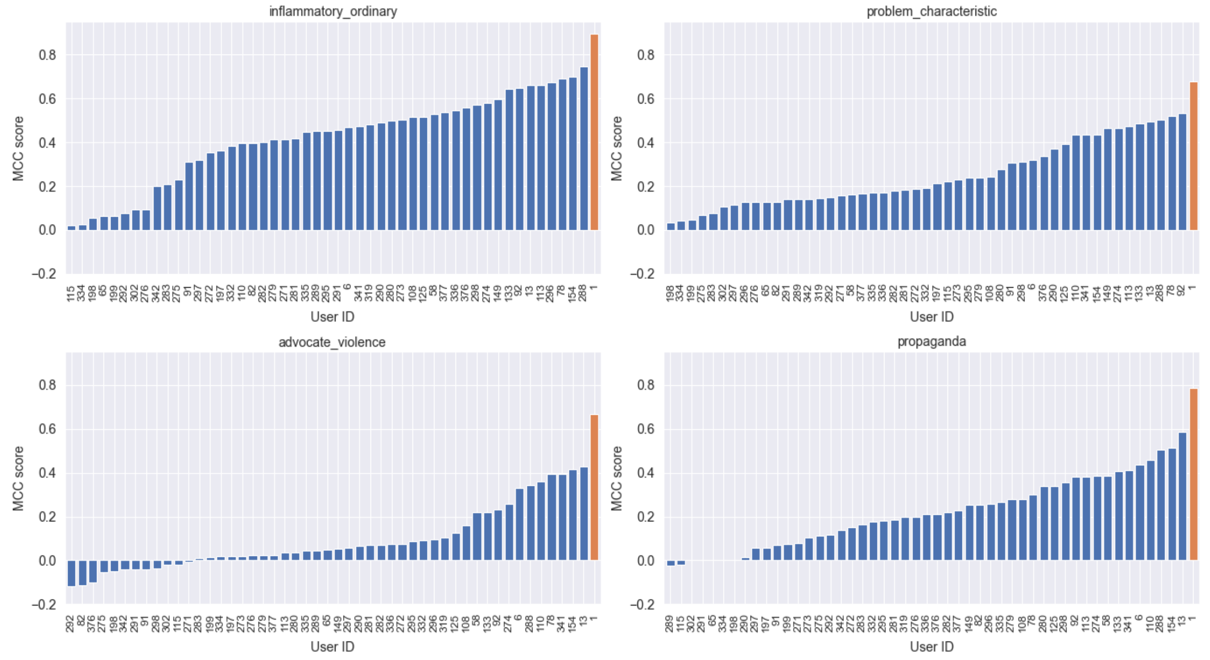

Figure 3: Annotator agreement with the ground truth (SME labels) for each of the four label categories. Each blue bar represents one annotator, while the orange bars represent annotations generated by a social scientist with expertise in online inflammatory language.

were divided into two groups according to their survey scores: group with a score lower than the median score and individuals with higher or equal to the median score. This was done for each survey separately. We plotted statistics of agreement with the GT for each of these two groups using boxplots. We also compare the groups using two-sample t-test with significance threshold of 5%. The results are snown in Figures 5, 7, and 9.

Figure 4 shows SR2K results for all annotators, and Figure 5 shows differences in agreement with the subject matter expert (SME) for individuals with low and high SR2K score. The figures are color-coded to match, i.e., the light blue bars in Figure 4 indicate which annotators belong to the light blue colored bars in Figure 5. Specifically, annotators were divided into two groups according to their SR2K score: group with a score lower than the median score (19) and individuals with higher or equal to the median score. There are 24 annotators in the former group and 25 annotators in the latter group. The figure shows there are significant differences between groups with high and low SR2K score in terms of agreement with the SME (GT), particularly when identifying tweets containing a problem characteristic and tweets advocating violence. Individuals with a higher SR2K tend to agree with the SME, on average, much less than individuals with a lower score when identifying such tweets. We compared the groups using two-sample t-test and in all cases but the first (inflammatory/ordinary) the differences in terms of MCC between the two groups are statistically significant (p-value was lower than 5%).

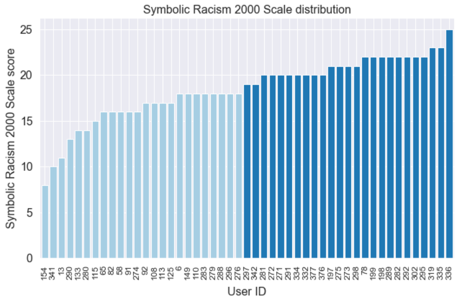

Figure 4: SR2K results for all annotators. Each bar in the figure represents one annotator. The bars are color-coded to indicate which annotators scored lower than median SR2K score, and which score higher or equal to median score.

Figure 6 shows MFQ results for all annotators. Similarly as in the case of the SR2K score, we compared the MFQ score (also called “progressivism” score) with the annotators’ MCC score (Figure 7). The results show that individuals with higher progressivism score tend to, on average, agree with the SME more than individuals with lower progres-

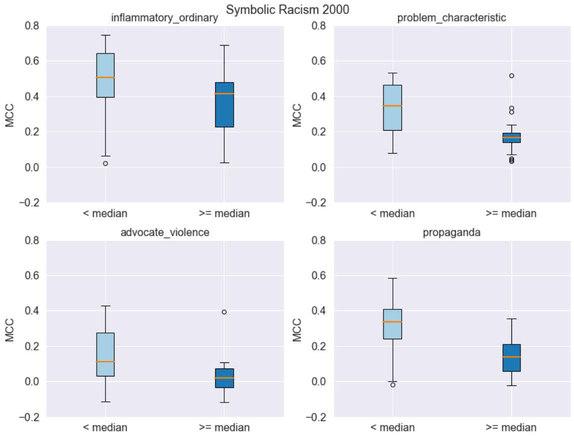

Figure 5: Comparison of the Symbolic Racism 2000 Scale results and agreement with the GT. The lighter colored bar represents the group with the lower than median SR2K score and the darker blue represents the group with higher or equal to median score.

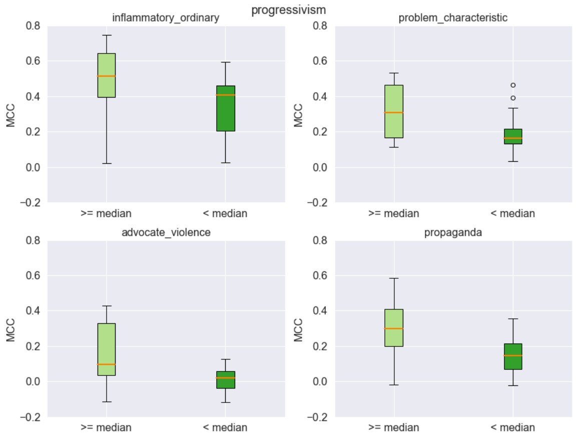

Figure 7: Comparison of the Moral Foundations Questionnaire results and agreement with the GT. The lighter colored bar represents the group with the higher or equal to median MFQ (progressivism) score and the darker bar represents the group with lower than median score.

sivism score. We compared the two groups using independent two sample t-test. The differences in MCC between the two groups statistically significant for all four narrative categories (p-value lower than 5% in all cases).

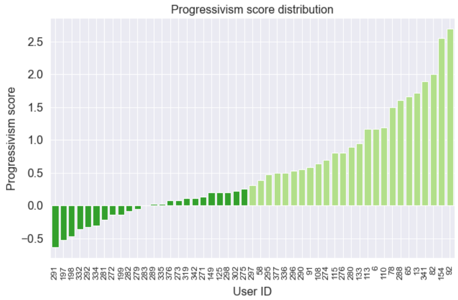

Figure 6: MFQ results for all annotators. Each bar in the figure represents one annotator. The bars are color-coded to indicate which annotators scored lower than median MFQ score, and which score higher or equal to median score.

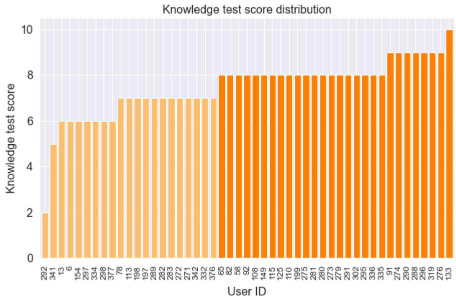

Figure 8: Knowledge Test results for all annotators. Each bar in the figure represents one annotator. The bars are color-coded to indicate which annotators scored lower than median Knowledge Test score, and which score higher or equal to median score.

Figure 8 shows Knowledge Test results for all annotators. Finally, we compared the knowledge test scores with the annotators' agreement with GT (Figure 9). In this case, we can see that there are no differences between the two groups of annotators, which was also confirmed by t-test (p-values for all four narrative categories range from 77-98%).

4 Discussion

The qualitative surveys captured a number of profile characteristics of both SME and other data annotators. As our final question, we aimed to rank the collected characteristics according to their importance towards indicating higher or lower agreement with the ground truth (SME). To do this, we calculated the following three statistics to capture the relation between each profile characteristic and agreement with the ground truth: mutual information score statistic, recursive feature elimination, and univariate linear regression test. All profile characteristics were ranked using each of these three statistics. A final rank for each profile characteristic was produced as a sum of all three individual ranks.

In Figure 10, we show the ranked characteristic from the least (Residence) to the most (purity_sanctity – a component of the MFQ) important. Thus, the results show the location of residence is the least important factor in indicating

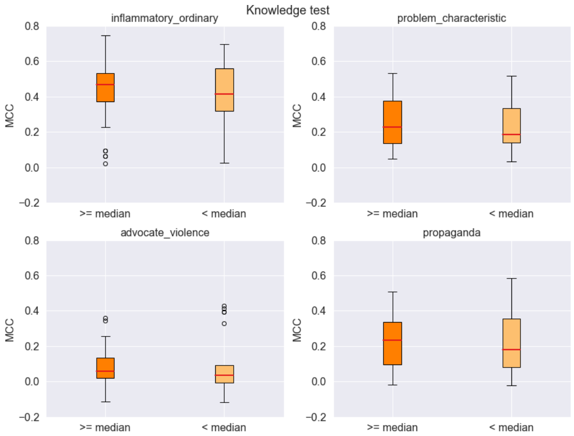

Figure 9: Comparison of the Knowledge Test results and agreement with the GT. The lighter colored bar represents the group with the lower than median Knowledge Test score and the darker bar represents the group with higher or equal to median score.

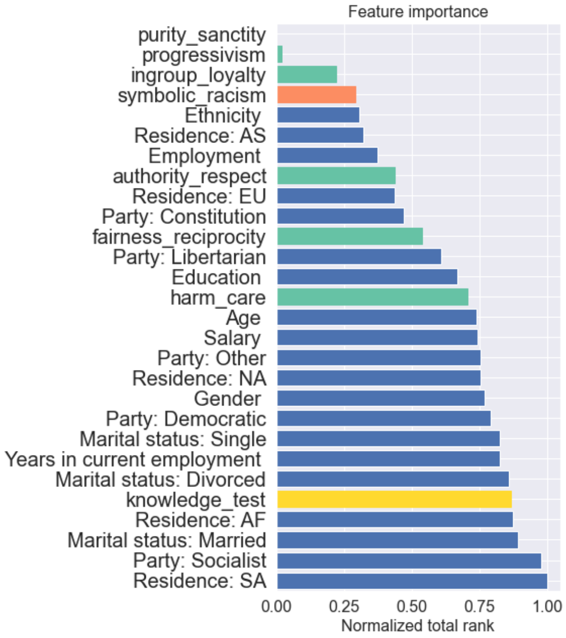

Figure 10: Normalized total rank of various attributes. Blue color bars are demographics while other from survey questionnaire.

possible agreement with the ground truth, while socio-demographic attributes such as purity, progressiveness, etc., are much more important for identifying more reliable annotators.

In Figure 11, we show top five characteristics and their average values that substantially influence the evidentiary based selection of unbiased annotators.

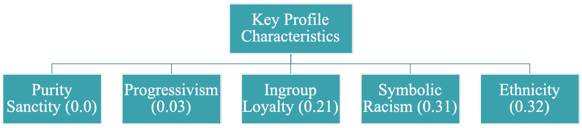

Figure 11: Key attributes and their measured average values.

Specifically, this research has strongly indicated that data reviewers morals, prejudices, and prior knowledge of the narrative in question significantly impact the quality of labeled data and consequently, the performance of ML models. ML projects that rely on labeled text data to understand narratives must qualitatively assess their data reviewers worldviews if they are to make definitive statements about their results.

For the automated detection of complex narratives, it is important that models built should be free from any implicit biases of any kind. We hope this work contributes to the broad research presently taking place across the field of machine learning to analyze and understand massive amounts of electronic communications (i.e. social media posts, news, blogs, etc.) in complex scenarios involving issues related to ethics, race, gender, and biases surrounding them.

5 Conclusion

The discipline of natural language processing and machine learning has tremendously improved in the last decade. However, it still suffers from biases surrounding complex narratives related to education, health, climate, gender, race, and ethics; especially, it unfairly penalizes certain segments of the population, e.g. women and minorities. Societal, cultural, and demographic phenomena play a pivotal role in how the population conceptualizes policy decisions and complex events. Thus, it is critical for societal good that narratives should be carefully crafted for maximum impact. It is our collective responsibility that any automation (classification, prediction, etc.) surrounding these narratives must be free of any preconceived notions or predilections of any kind. One way to achieve this is by producing high-quality input labeled data curated by annotators aware of such biases. Inspired by this, we have proposed a new framework based on mixed-method design to improve the odds of selecting annotators, who can curate unbiased and high-quality labeled data. In doing so, we identified and ranked personal and professional traits critical to selecting a diverse pool of data annotators, so the resulting labeled data and the models built using those data best matched the ground-truth. In the future, we would like to extend our study to cater to multi-lingual narratives and expand beyond existing issues of culture, region, and geopolitical dynamics.

Acknowledgments

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (`http://energy.gov/downloads/doe-public-access-plan`).

The authors are grateful to RJ Moquito of National Geospatial Intelligence Agency for their support and guidance. The authors also extend their thanks to Budhendra "Budhu" Bhaduri, Amy Rose, Marie Urban, Supriya Chinthavali for allocating necessary resources to complete the research work.

References

Ahmet Aker, Mahmoud El-Haj, M. Dyaa Albakour, and Udo Kruschwitz. 2012. Assessing Crowdsourcing Quality through Objective Tasks. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1456–1461, Istanbul, Turkey. European Language Resources Association (ELRA).

Patrick Anderson. 2019. Damages caused by ai errors and omissions: Management complicity, malware, and misuse of data (part i of a special report). *Malware, and Misuse of Data (Part I of a Special Report)(June 21, 2019)*.

Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. 2019. Fairness in recommendation ranking through pairwise comparisons. In *KDD*.

Vikas Bhardwaj, Rebecca Passonneau, Ansaf Salleb-Aouissi, and Nancy Ide. 2010. Anveshan: A Framework for Analysis of Multiple Annotators' Labeling Behavior. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 47–55, Uppsala, Sweden. Association for Computational Linguistics.

Gerald Brown, Matthew Carlyle, Javier Salmerón, and Kevin Wood. 2006. Defending critical infrastructure. *Interfaces*, 36(6):530–544.

Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, et al. 2020. Toward trustworthy ai development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*.

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

John W Creswell and Vicki L Plano Clark. 2017. *Designing and conducting mixed methods research*. Sage publications.

Caitlin L Davies, Chris G Sibley, and James H Liu. 2014. Confirmatory factor analysis of the moral foundations questionnaire. *Social Psychology*.

Jason Furman and Robert Seamans. 2019. Ai and the economy. *Innovation policy and the economy*, 19(1):161–191.

Jesse Graham, Brian A Nosek, Jonathan Haidt, Ravi Iyer, Spassena Koleva, and Peter H Ditto. 2011. Mapping the moral domain. *Journal of personality and social psychology*, 101(2):366–385.

David Gunning. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2(2).

Foad Hamidi, Morgan Klaus Scheuerman, and Stacy M. Branham. 2018. Gender recognition or gender reductionism? the social implications of embedded gender recognition systems. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 113, New York, NY, USA. Association for Computing Machinery.

Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of opportunity in supervised learning. *arXiv*.

P. J. Henry and David O. Sears. 2002. The symbolic racism 2000 scale. *Political Psychology*, 23(2):253–283.

John B McConahay. 1986. Modern racism, ambivalence, and the modern racism scale.

Safiya Umoja Noble. 2018. *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press.

Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. 2019. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453.

Osonde A Osoba and William Welser. 2017. *The risks of artificial intelligence to security and the future of work*. RAND.

Francesc Pedro, Miguel Subosa, Axel Rivas, and Paula Valverde. 2019. Artificial intelligence in education: Challenges and opportunities for sustainable development.

Andreas Peldszus and Manfred Stede. 2013. Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 196–204, Sofia, Bulgaria. Association for Computational Linguistics.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On fairness and calibration. In *Advances in Neural Information Processing Systems*, volume 30, pages 5680–5689. Curran Associates, Inc.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence, Italy. Association for Computational Linguistics.

David O. Sears and P.J. Henry. 2005. Over thirty years later: A contemporary look at symbolic racism. volume 37 of *Advances in Experimental Social Psychology*, pages 95 – 150. Academic Press.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 614622, New York, NY, USA. Association for Computing Machinery.

Allison Shorten and Joanna Smith. 2017. Mixed methods research: Expanding the evidence base. *Evidence-Based Nursing*, 20(3):74–75.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

David C Vladeck. 2014. Machines without principals: liability rules and artificial intelligence. *Wash. L. Rev.*, 89:117.

Sarah Myers West, Meredith Whittaker, and Kate Crawford. 2019. Discriminating systems. *AI Now*.

Jenna Wiens, Suchi Saria, Mark Sendak, Marzyeh Ghassemi, Vincent X Liu, Finale Doshi-Velez, Kenneth Jung, Katherine Heller, David Kale, Mohammed Saeed, et al. 2019. Do no harm: a roadmap for responsible machine learning for health care. *Nature medicine*, 25(9):1337–1340.

An Evaluation of Disentangled Representation Learning for Texts

Krishnapriya Vishnubhotla[1,2], **Graeme Hirst**[1], and **Frank Rudzicz**[1,2,3]
[1]Department of Computer Science, University of Toronto
[2]Vector Institute for Artificial Intelligence
[3]Unity Health Toronto
{vkpriya,gh,frank}@cs.toronto.edu

Abstract

Learning disentangled representations of texts, which encode information pertaining to different aspects of the text in separate representations, is an active area of research in NLP for controllable and interpretable text generation. These methods have, for the most part, been developed in the context of text style transfer, but are limited in their evaluation. In this work, we look at the motivation behind learning disentangled representations of content and style for texts and at the potential use-cases when compared to end-to-end methods. We then propose evaluation metrics that correspond to these use-cases. We conduct a systematic investigation of previously proposed loss functions for such models and we evaluate them on a highly-structured and synthetic natural language dataset that is well-suited for the task of disentangled representation learning, as well as two other parallel style transfer datasets. Our results demonstrate that current models still require considerable amounts of supervision in order to achieve good performance.

1 Introduction

The similarity of texts can be assessed along multiple dimensions. They could contain the same topics, as identified by semantic similarity. They could belong to the same genre or be written by the same author, in which case we might identify stylistic similarity. Texts that present a positive sentiment may be considered similar to one another when compared to those that express a negative sentiment, even if they talk about different topics. The similarity of texts, therefore, must be defined together with a frame of reference or a pre-specified dimension of variation.

Text representations obtained by current representation learning methods combine all of these different aspects of a text into a single vector embedding (Conneau et al., 2017; Reimers and Gurevych, 2019). This results in only a fuzzy measure of text similarity when it is calculated using methods such as the cosine distance between vector embeddings. Recently, some research in NLP has focused on learning *disentangled* representations for texts, which aim to capture the different dimensions of variation of a text in separate vector embeddings. These methods have been investigated for style transfer to obtain disentangled representations of content and style (John et al., 2019; Romanov et al., 2019; Cheng et al., 2020), and paraphrase generation for disentangling syntax and semantics (Chen et al., 2019; Balasubramanian et al., 2020). Inspired by parallel developments on style transfer and disentanglement in computer vision, many of them operate within the variational autoencoder framework, where the autoencoder is modified to now encode a text into two latent vectors: one capturing the style (the aspect of variation), and the other capturing the content. Style transfer is then achieved by combining the content vector of the input with a style vector of the target style.

Disentanglement-based models offer two main advantages when compared to end-to-end style transfer methods:

1. Sampling from the latent space of the style embeddings allows for more diverse and controlled stylistic generation.
2. Similarity of documents can now be calculated for each aspect of variation, allowing for finer-grained retrieval.

In this work, we focus on models that aim to disentangle **content** from **form**, or meaning from style, for texts. Thus, style transfer is viewed as a form of paraphrasing, where the paraphrase demonstrates certain stylistic properties. It is important to make this distinction between what constitutes style versus meaning for a text, more so when for-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1939–1951
August 1–6, 2021.

Meaning Representation	**name[nameVariable], food[Indian], customerRating[average]**
EXTROVERT	nameVariable is an Indian place, also nameVariable has an average rating, you know.
UNCONSCIENTIOUSNESS	Yeah, mmhm... I don't know. nameVariable is an Indian place with a damn average rating.
CONSCIENTIOUSNESS	Did you say nameVariable? I see, well it is an Indian restaurant with an average rating.
DISAGREEABLE	Actually, basically, everybody knows that nameVariable is an Indian restaurant, also it has an average rating.
AGREEABLE	Let's see what we can find on nameVariable. Well, right, it is an Indian restaurant with a quite average rating.

Table 1: The same meaning representation mapped to different stylistic surface realisations in the PersonageNLG dataset.

mulating style transfer problems, in order to have measurable definitions of what information may and may not be changed by the model. Parallel paraphrase datasets, therefore, are a much-needed resource for the effective evaluation of these models. However, few works on disentangled representation learning actually evaluate their models on such datasets, testing instead only on the non-parallel datasets used for training. Further, some works evaluate exclusively on metrics from the style transfer task, ignoring the retrieval aspect.

The goal of this study is to conduct a systematic and grounded evaluation of various disentangled representation learning models. We first use, as a testbed for our evaluation strategy, a highly-structured Natural Language Generation dataset, PersonageNLG (Oraby et al., 2018), which maps a meaning representation to a set of stylistically different surface realisations corresponding to five personality types (Table 1). This dataset provides us with textual variation and gold-standard annotations for the two dimensions of interest, content and form. The structured and somewhat synthetic nature of this dataset allows us to systematically investigate the quality of the disentangled representations for metrics of aspect-specific retrieval as well as style transfer.

We then extend our experiments to two other parallel style transfer datasets: the GYAFC formality corpus (Rao and Tetreault, 2018), and the Bible dataset (Carlson et al., 2018). Although parallel, they are not annotated for semantic content as the PersonageNLG dataset is; however, they are arguably more representative of the kinds of data we expect to obtain in the real world. Despite testing our models with loss functions that do not require parallel data, we limit ourselves to such datasets for the ease and consistency of evaluation. Our code is publicly available at `github.com/priya22/drl-nlg-eval`.

2 Background

Works on style transfer in NLP operate with varying definitions of what constitutes style. Many choose to define this as a factor of variation in data that can be manipulated, including aspects such as topic and sentiment. This approach has been contested by others who maintain that the semantic content of a text should not be modified when manipulating style. The latter definition fits with what stylometric analysis and linguistics consider to be the style of a text. Thus, the output of a style transfer system should be a paraphrase of the input text.

2.1 Model Architectures

The models used to achieve style transfer fall into a few broad categories. End-to-end sequence transformation models are inspired by machine translation seq-2-seq models, where the translation is done from style A to style B. These sometimes require parallel data, but methods such as backtranslation circumvent that (Prabhumoye et al., 2018; He et al., 2020). Some others look at this as a controlled text generation problem, where the control is generally a categorical variable indicating the desired stylistic class of the output, and is passed along with the input to a text generation module such as an LSTM (Hu et al., 2017; Ficler and Goldberg, 2017).

The focus of this work is on a third class of models that first learn disentangled latent representations of style and not-style (henceforth referred to as content) for a text, and train a generator that takes both vectors as input. To transform a text A into the style of text B, we extract the content vector of the former, the style vector of the latter, and pass them through the generator. Note that here, the style vectors of each text are not the same categorical variable, but rather a vector embedding that encodes the style-specific properties of the text. One can also obtain a single style vector representation by averaging the style vectors of all texts belonging to that class, as Fu et al. (2018) did; however, we are more interested in disentangling information at the individual text level rather than in corpus-level indicators.

2.2 Relevant Literature

Disentanglement of latent spaces has been widely studied and very successful in computer vision applications, but less so in NLP. This can be attributed to the vague nature of what actually constitutes style as opposed to content for a text, and uncertainty as to whether they can actually be disentangled at all (Lample et al., 2019; Yamshchikov et al., 2019). However, by using some supervision with respect to these two dimensions, researchers have attempted to obtain representations that *for the most part* encode information relating to only style or only semantics.

Romanov et al. (2019) first proposed obtaining separate embeddings of form and meaning of texts. Starting with an encoder-decoder setup, they added adversarial and motivational losses based on style labels that encourage the form vector to encode information relevant to the label. Their models were evaluated on non-parallel datasets with two types of stylistic variation: diachronic language shift and newspaper titles versus scientific paper titles. In parallel work, John et al. (2019) proposed a disentanglement model that appends additional content-based losses, where content is approximated by a bag-of-words representation of the text. Their approach was applied to sentiment transfer for Yelp and Amazon reviews.

Other work has looked at disentangling syntax from the semantics of a text. Chen et al. (2019) proposed a VAE-based model that used parallel paraphrase corpora; this was also the focus of Bao et al. (2019) and Balasubramanian et al. (2020).

All of these works are very similar in the base model architecture and the kinds of loss functions used to guide disentanglement. In the following sections, we consolidate and propose a broad categorization of these losses that we hope will guide future work in this area. We then evaluate these models on parallel style transfer datasets, with ablation studies on the PersonageNLG dataset.

Note on unsupervised disentanglement: While unsupervised approaches such as the β-VAE have been very successful at disentangling factors of variation in visual data (Higgins et al., 2017), we are still far from achieving such a clean separation of the data generating factors for text. A recent promising approach in this direction was presented by Xu et al. (2020), who use pretrained models along with a novel constraint over the latent space of a VAE to control the sentiment and topic of a text.

3 Methodology

3.1 Autoencoder Model

Following previous literature, our encoder module takes as input a text, and computes latent vector embeddings for each aspect: content and form. The decoder takes as input both vectors, and generates output text. The entire autoencoder model is trained to reconstruct the input text.

Let us denote our content and form encoders by E_c and E_f, the decoder by G, and their model parameters by θ_{E_f}, θ_{E_c} and θ_G respectively. Our base loss can thus be written as:

$$L_{AE} = L_{rec} + \beta L_{reg} \tag{1}$$

where

$$L_{rec}(\theta_{E_c}, \theta_{E_f}, \theta_G) = \mathop{\mathbb{E}}_{x}[-\log\, p_g(x \,|\, E_f(x), E_c(x))] \tag{2}$$

is the reconstruction loss of the autoencoder given input x, p_g is the decoder distribution, and L_{reg} is an additional regularization term. For a Variational Autoencoder (VAE) model, this is the Kullback-Leibler divergence between the latent posterior distributions q of the encoders and the latent prior $p(z)$:

$$L_{reg}(\theta_E) = D_{KL}(q(z \,|\, x) \,\|\, p(z)) \tag{3}$$

An alternative regularization for text autoencoders was proposed by Shen et al. (2020), where the AE loss is augmented with a denoising objective. The input text is perturbed with small amounts of "noise" in the form of word deletions or substitutions; the autoencoder is still trained to reconstruct the original text. Here,

$$L_{reg}(\theta_E, \theta_G) = \mathop{\mathbb{E}}_{(x,\tilde{x})}[-\log\, p_g(x \,|\, E(\tilde{x}))] \tag{4}$$

where $\tilde{x}$ is the noisy version of the input text x. These denoising autoencoders (DAEs) were shown to be more stable than VAEs for text modeling.

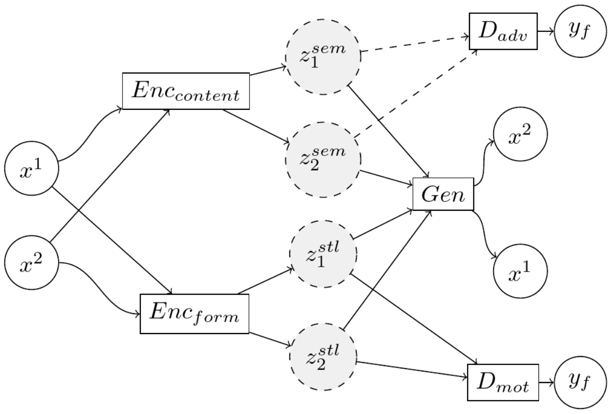

Figure 1: The main components of a Disentangled Representation Learning model. z^{sem} and z^{stl} denote the content and form vectors respectively; each is input to a motivational and an adversarial network. The generator is trained to reconstruct the original input as well as paraphrases.

3.2 Losses for Disentanglement

With our base autoencoder in hand, we can now start adding losses that encourage each latent vector to encode information relevant to the corresponding aspect, i.e, content (semantics) and form (style).

3.3 Proxy-based Losses

Supervised losses are usually based on some form of proxy information present for a specific aspect.

For the form dimension, the most common proxy is class labels that indicate the style of a particular datapoint, such as formal or informal. A stronger proxy could include a list of linguistic attributes of the sentence that are highly indicative of and inform its style. These usually have to be manually defined and extracted, as by John et al. (2019), who use high-polarity sentiment words as a proxy for the sentiment aspect.

An attribute-based proxy for content can be found by looking at the information present in, say, the meaning representation of a sentence (as provided in NLG datasets), or extracting semantics-predictive information such as semantic role labels. John et al. (2019), for example, use the bag-of-words representation of a text as a proxy for semantic information.

These additional losses are usually combined with the autoencoder objective in two ways: as a **motivational** loss, which *encourages* a latent vector to encode the proxy information, and as an **adversarial** loss, which *discourages* a latent vector from encoding the proxy information. Thus, once we define a proxy loss for, say, content, we would append a motivational loss to the content encoder and a corresponding adversarial loss to the form encoder.

Below, we use z_c and z_f to denote the content and form vectors of a text x.

3.3.1 Loss Functions for Form

Motivational: For the datasets that we consider here, and in most real-world applications, we have the stylistic class of a text as a proxy for the form aspect. The motivational and adversarial networks are implemented as classifiers that are trained to predict this label from the corresponding latent representation. The loss function of the former is simply the cross-entropy loss of the classifier:

$$L_{mot}(\theta_D, \theta_{E_f}) = \underset{z_f}{\mathbb{E}}[-\log\ D(z_f)] \tag{5}$$

D and θ_D represent the classifier and its parameters respectively.

Adversarial: We now want to ensure that the content vector does not contain any information about the form class of the text. Thus, we aim to *maximize* the entropy of the adversarial classifier. This is the approach followed by many prior works (John et al., 2019; Fu et al., 2018), which we also adopt here, as it can be nicely extended to multi-label classification, which will prove useful in the content-based losses.

Adversarial training occurs in two steps. First, the classifier is trained to predict the form label given the content representation. Then, the content encoder's parameters are updated based on the entropy loss:

$$L_{adv}(\theta_D) = \underset{z_c}{\mathbb{E}}[-\log\ D(z_c)] \tag{6}$$

$$L_{adv}(\theta_{E_c}) = \underset{z_c}{\mathbb{E}}[\mathbb{H}(D(z_c))] \tag{7}$$

where $\mathbb{H}(D(z_c))$ is the entropy calculated over the classifier-predicted label distribution.

3.3.2 Loss Functions for Content

Proxy information for content is generally rare, and needs to be formulated by means of some heuristic measure. In the case of NLG datasets, we have annotated meaning representations that serve as a good proxy. However, such structured representations of meaning are difficult to obtain for general texts.

Let us assume we have a list of k key-value pairs that represent content, as in the MR from Table 1.

We represent the content proxy as a k-dimensional multi-hot vector y_c, where each dimension y_c^i is a binary indicator of whether key k_i is present in the MR.

Motivational: The motivational loss is thus defined as the multi-label cross-entropy loss over the classifier prediction, similar in form to Eq. 6, but now taking the content vector as input.

Adversarial: In turn, the adversarial content loss is found by first training a multi-label classifier that takes the form vector as input and predicts the content attribute vector, and then training the form encoder to maximize the entropy of this classifier.

3.4 Parallel Losses

These losses require as input a pair of paraphrases, say x^1 and x^2. We obtain the latent vectors for content and form for each of these: $z_c^1, z_f^1, z_c^2, z_f^2$ respectively.

Paraphrase reconstruction loss: Here, we swap the content vectors of the paraphrases, retain the form vectors, and attempt to reconstruct the original inputs. This was used by Chen et al. (2019) to disentangle syntax and semantics in paraphrase corpora.

$$\begin{aligned} L_{para}(\theta_{E_c}, \theta_{E_f}, \theta_G) = & \mathop{\mathbb{E}}_{x_1,x_2}[-\log\ p_g(x^1 \mid z_f^1, z_c^2)] \\ & + \mathop{\mathbb{E}}_{x_1,x_2}[-\log\ p_g(x^2 \mid z_f^2, z_c^1)] \end{aligned} \tag{8}$$

Distance-based loss: This takes the form of a max-margin loss that aims to keep the cosine similarity between the content embeddings of paraphrases higher than that between a random selection of negative example pairs. This particular loss is used by Chen et al. (2019) and Balasubramanian et al. (2020) to disentangle syntax and semantics, although they differ slightly in the criteria to select positive and negative pairs.

4 Datasets

PersonageNLG Dataset: The PersonageNLG corpus (Oraby et al., 2018) is a set of 88,000 pairs of meaning representations and natural language utterances, based on the E2E challenge dataset. Each utterance is associated with a unique style, which corresponds to one of five personality types: Agreeable, Disagreeable, Conscientious, Unconscientious, and Extrovert. The utterances are obtained by means of a statistical NLG system, and by varying a set of 36 predefined stylistic parameters that specify certain phrase aggregation and pragmatic markers (Table 1). The dataset essentially provides us with a structured and synthetic corpus of textual variation, with each utterance annotated for both content (a meaning representation) and form (the stylistic personality class). This makes it ideal for evaluating the quality of disentangled representations.

GYAFC Dataset: Introduced by Rao and Tetreault (2018), the GYAFC corpus consists of 120,000 parallel sentence pairs that are paraphrased in two styles: formal and informal. See section A.1 for details. GYAFC is one of the very few parallel datasets available for style transfer research in NLP.

Bible dataset: This dataset, compiled by Carlson et al. (2018), consists of eight verse-aligned public domain versions of the Bible; see section A.2 for details. These versions are spread out across different decades, and thus belong to their own unique stylistic class. The natural parallel alignment between verses, as well as the relatively stable nature of their semantic content across time, makes this dataset ideal for studies in style transfer (although surprisingly few works on style transfer use it).

5 Evaluation

The goal of our model is to encode in separate vectors the style-specific and content-specific features of a text. The following metrics guide our similarity measures for content and form:

- **Content (C_{sim}):** For the PersonageNLG dataset, content similarity between two sentences is measured as fraction overlap between content labels (Section 3.3.2). For generated sentences, we use all possible slot values for each field of the Meaning Representation (Table 1) to approximate a bag-of-words content representation, and calculate fraction overlap of content terms in both sentences. For the other two datasets, we use the BLEU scores between the generated text and the target paraphrase as a measure of content preservation.

- **Form (F_{class}, F_{sim}):** For all three datasets, we first train a fasttext[1] classifier on their respective training sets to predict stylistic class given the input text (F_1 scores on the test sets

---

[1] https://fasttext.cc/

are shown in Table 2). This classifier is then used to predict the style class of a generated text. F_{class} is the F_1 score of the predicted labels for generated texts, using the target labels as ground truth.

Additionally, for the NLG dataset, we use an F_{sim} measure that measures the fraction overlap of non-content words of the two texts, where "non-content" is defined as all words that are not associated with content as defined above.

We divide our evaluation metrics into three groups, based on the capabilities and use-cases of learning disentangled representations.

5.1 Autoencoder Capabilities

Reconstruction: One of the basic functions of our model is as an autoencoder, i.e., a model that can reconstruct the input text from its latent encoding. We use the self-BLEU score between the input (reference) and the generated text to measure reconstruction quality.

5.2 Disentanglement

The quality of disentanglement of representations is assessed in two main ways.

Classification: The first is a classification task that aims to predict the proxy information for each text using the latent vectors. For each of our dimensions of content and form, this gives us four measures corresponding to the accuracy of a classifier trained to predict content (form) information from the content (form) vectors, and that of a classifier trained to predict form (content) information from the content (form) vectors. Ideally, we want the former numbers to be high and the latter to be close to random chance.

Retrieval: As stated, one of the advantages of having disentangled representations for each aspect is that we can now obtain aspect-specific similarity scores. Since all our datasets are parallel paraphrase corpora, we can measure how well the content vectors perform at retrieving paraphrases. For each sentence in our test set, we obtain the cosine similarity scores of its content vector with that of every other sentence, and look at how many of the top-k matches are paraphrases of the input. We evaluate this for $k = 5$ for the GYAFC and Bible datasets, and $k = 1$ for the NLG corpus.

Similarly for form, we find the top-k neighbours for the form vector of each sentence and report the precision@k of retrieving texts from the same stylistic class. This metric is particularly informative for PersonageNLG, where we look at the F_{sim} between the input and the closest match.

Dataset	**F_1 score**
PersonageNLG	0.99
GYAFC	0.87
Bible	0.72

Table 2: Performance of the external fasttext classifier on test sets.

5.3 Style Transfer

Finally, we evaluate the effectiveness of our model for the task of style transfer, by testing with paraphrase pairs. Thus, for each pair of paraphrases in the test set, we obtain the content vector of the first and the form vector of the second, and pass them to the decoder module (and vice-versa). The **content preservation** and **transfer quality** of generated sentences are measured using C_{sim} and F_{class} respectively. We also measure the **fluency** of the generated text by measuring the perplexity of generated sentences with a trigram Kneser-Ney language model trained on the training set of each dataset.

6 Experiments

6.1 Setup

The encoder and decoder of our base model are 2-layer LSTM networks with a hidden size of 64. Both the content and form vectors are of the same size for each dataset: 16 for PersonageNLG and 32 for the others. At each decoder timestep, the concatenated latent vector $z = [z_c, z_f]$ is added to the input to obtain the next prediction. During training, teacher forcing with probability 0.4 is used; we use greedy decoding for the PersonageNLG dataset and and beam search with a beam size of 5 otherwise. Motivational and adversarial classifiers are single-layer linear networks trained with RMSprop.

The GYAFC and NLG datasets come with predefined training and test splits. For the Bible dataset, we use a random stratified split with 65–15–20 split for training, validation, and test respectively.

6.2 Experimental Method

Our goal is to methodologically evaluate the effectiveness of each of these losses for disentangling content from form. We start with our vanilla autoencoder model (L_{ae}), and at each step, add additional losses based on incorporating some supervised information into our model. The terms we add are guided by some intuition on the kinds of supervision we would expect to see in the real world.

1. **Form losses L_{form}:** This assumes that each text is labeled with a class that indicates its stylistic category, such formal / informal, Shakespearean / modern, positive / negative, etc. This enables us to append two of our losses to the base loss: the motivational and adversarial form losses (Section 3.3.1).
2. **Motivational only L_{mot}:** We now add our proxy information for content. We first keep only the motivational losses and remove the adversarial losses for each aspect.
3. **Combined proxy losses L_{proxy}:** We add adversarial losses for form and content to the model above, giving us our full proxy-loss–based model.
4. **Paraphrase losses:** Finally, we add the parallel losses detailed in Section 3.4, taking advantage of our parallel datasets. The alignment of two paraphrases essentially acts as a proxy for the equivalence of semantic content between two texts. Accordingly, we test the following loss combinations:
   - Parallel losses only (Section 3.4) (L_{para});
   - Parallel losses + form losses from point 1 above (L_{para_f}).

Baseline: We additionally compare the effectiveness of these models when compared to a categorical conditional generation model. Here, the form vector is simply an 8-dimensional encoding of the style class label, rather than derived from the input text. The model is trained using the F_{adv} and C_{mot} losses to ensure the content embedding doesn't encode style information, along with the reconstruction loss L_{rec}.

All of these loss combinations are tested on the PersonageNLG dataset, since it is annotated with proxies of both content and form.

7 Results and Discussion

We experimented with both the VAE and the DAE models for our base architecture, and found that the latter was more stable during training. Training the VAE with multiple latent vectors and additional losses often resulted in the model completely ignoring one of the latent vectors; stable modeling of such architectures is still an active area for text data and is left to future work.

7.1 Disentanglement

We first examine how well our models are able to disentangle information pertaining to form and content into the respective latent vectors. Table 3 reports the performances of each model for the metrics discussed in Section 5.2. For conciseness, we only report cross-aspect classification scores in the Classification column, where a lower number indicates better disentanglement. More detailed results with same-aspect scores are presented in Appendix C.1.

In the absence of parallel data, we see that directly adding supervised losses along each dimension is the most effective strategy of disentangling information. Accordingly, the largest performance drops on cross-aspect classification are achieved with the addition of motivation losses L_{form} and L_{mot} for form and content. Adversarial losses do help the overall performance of the model as demonstrated by the drop in cross-aspect classification metrics, especially in the form domain. The maximal supervision afforded by the paraphrase losses L_{para} demonstrates a significant improvement over the best proxy-based model here, indicating that proxy information is generally not complete enough to capture semantic content. However, the lack of similar supervision along the form dimension is reflected in the higher cross-aspect classification scores across all models.

We show t-SNE plots of the form and content vectors computed by each model in Appendix B. The paraphrase model gives us neat clusters of the content vectors corresponding to the different meaning representations.

However, classification numbers alone don't present the whole picture. Our measures of retrieval quality help to isolate the effects of classifier effectiveness from the goodness of the representations alone. For the NLG dataset in particular, the retrieval scores tell us whether the form vector of a text actually encodes information about the linguistic features informing its style, rather than simply encoding enough to be classified in the right stylistic class. Are sentences with similar

	Autoencoder	**Disentanglement**			
	BLEU	Classification: $F_1 \downarrow$		Retrieval $\uparrow$	
Target →		Form	Content	Form	Content
Input →		z_c	z_f	z_f	z_c
$L_{ae}*$	43.4	0.96	0.73	0.57	0.85
L_{form}	−0.07	−0.67	−0.11	0.13	0.08
L_{mot}	0.01	−0.31	**−0.14**	0.08	0.13
L_{proxy}	−0.05	−0.73	−0.13	**0.13**	**0.14**
L_{para}	0.06	−0.68	−0.03	0.11	0.13
L_{para_f}	−0.03	**−0.75**	−0.10	0.12	0.12
$L_{baseline}$	−0.03	−0.65	–	–	0.09

Table 3: Results on reconstruction and disentanglement quality for the PersonageNLG dataset. The first row reports the absolute metric for the base autoencoder model L_{ae}; subsequent rows report the difference from this base score. The first column reports the self-BLEU score between the reconstructed and input text. For classification, we report the cross-aspect F_1 scores of a classifier trained to predict the target aspect from the input. For retrieval, we report the C_{sim} and F_{sim} scores between the input text and its nearest neighbour in the latent space.

textual stylistic or content features closer to each other in the embedding space when compared to other sentences from the same style/content class? The relatively low delta scores when compared to classification performance indicate that this is not the case. While there are marginal improvements, proxy-based losses don't seem to be informative enough to enforce fine-grained structure in the latent space. Our experiments on style transfer in the next section reinforce this conclusion.

7.2 Style Transfer

We swap the form and content vectors of paraphrases from our test set, and evaluate the generated sentences using the metrics defined in Section 5.3. For the NLG dataset, as before, we use term-overlap measures of the similarity for the content and style terms between the generated text and the target paraphrase (C_{sim} and F_{sim}); results are shown in Table 4. Both of these measures are far from their ideal values of 1.0.

The full proxy model L_{proxy} achieves the best performance across all metrics (sample outputs are shown in Appendix C.2). The paraphrase models tend to perform worse than the baseline, especially on the transfer strength metric, F_{sim}. This points to the form vector not being informative enough, especially when no motivational losses are used. It also indicates that the adversarial losses from the proxy-based models were indeed helpful in disentanglement.

We see similar trends in both disentanglement quality and style transfer for the GYAFC and Bible datasets. The quality of text generated was significantly worse when compared to the NLG dataset, but we are still able to encode the style and content-related information in separate vectors with some success, as evidenced by the retrieval scores.

	$C_{sim} \uparrow$	$F_{sim} \uparrow$	Fluency $\downarrow$
L_{ae}	0.29	0.46	1.11
L_{form}	0.28	0.58	1.08
L_{mot}	0.36	0.48	1.09
L_{proxy}	**0.39**	**0.72**	1.10
L_{para}	0.33	0.45	1.11
L_{para_f}	0.35	0.55	1.09
$L_{baseline}$	0.30	0.60	1.06

Table 4: Evaluation of style transfer on the PersonageNLG dataset. Arrows denote desired direction of change.

7.2.1 Does Disentanglement Help?

Our comparison with the categorical baseline $L_{baseline}$ tells us whether learning disentangled representations indeed provides an advantage for the style transfer task. From Table 4, we see that it does quite well on the C_{sim} metric, but is notably lower than L_{proxy} for F_{sim}. This demonstrates the advantage of having a separate vector representation of the form of a text, as opposed to the stylistic class.

7.3 Discussion

Our experiments all demonstrate that direct supervision along each aspect is crucial for learning good aspect-specific representations. This is the case even for the synthetic PersonageNLG dataset,

		Disentanglement		Style Transfer	
		Clf. ↓	Ret. ↑	C_{sim} ↑	F_{class} ↑
GYAFC	L_{base}	0.43	0.20	1.5	0.50
	L_{para_f}	0.35	0.49	3.6	0.83
Bible	L_{base}	0.64	0.25	1.3	0.11
	L_{para_f}	0.12	0.72	3.4	0.39

Table 5: Results on disentanglement quality and style transfer for the GYAFC and Bible datasets. The Clf. column reports the F_1 score of a classifier trained to predict the stylistic class label from the content vector; Ret. reports the P@5 for retrieving paraphrases using the content vectors.

which is by design constrained to have two separable aspects of variation (meaning and style); this is quite rare in real-world data. Indeed, the best performing style transfer model on this dataset, from Harrison et al. (2019), is a heavily supervised one that conditions a seq-2-seq model with annotations for each type of variation in the surface realisations (i.e., the presence of certain tokens).

In the absence of parallel datasets, proxy information is widely used to encourage disentanglement. However, our results show that such supervision is not sufficient to ensure that the embeddings actually encode the linguistic properties that are characteristic of a text's stylistic class (or meaning). With the retrieval experiments on the NLG dataset, we can see that the F_{sim} scores do not significantly differ between the different models. This indicates the difficulty of learning linguistic properties from class labels alone. This also explains the rather high F_1 scores for content classification from form embeddings.

The poor performance of these models on the style transfer task in particular indicates that the decoder, and hence the reconstruction objective itself, is somewhat lacking. This is reflected in the high classification scores of content information from form vectors, especially for the paraphrase model L_{para}. Additional constraints such as the backtranslation loss (Prabhumoye et al., 2018) go some way towards mitigating this issue. On the style transfer task, the baseline model $L_{baseline}$ shows performance comparable to the disentanglement models. One explanation for their poor performance is the inherent defects of variational models of text, such as the latent space vacancy issue, as demonstrated by other works (Xu et al., 2020; Shen et al., 2020).

For evaluation of such disentangled representations, traditional metrics of style transfer, such as the accuracy of an external classifier, are not the best indicators of disentanglement, nor a good demonstration of the usefulness of such embeddings. Most works on disentangled representations for style transfer do end up using a single, averaged vector embedding to inform the decoder of the desired target style. If the goal of learning disentangled representations is to perform style transfer between two classes, then a conditioned language model such as that of Ficler and Goldberg (2017) would suffice.

A more useful use-case for disentangled representations is for calculating aspect-specific similarity and retrieval between texts. However, it is not clear whether we can achieve such disentanglement with current models without fine-grained supervision along each aspect. While the NLG dataset provides us with the necessary supervision to introduce such constraints (via adversarial losses), and also evaluate them, such supervision is not available for real-world datasets.

8 Conclusion

Encoding the different factors of variation in data in separate embeddings is a desirable goal for learning robust and interpretable text representations, as well as for controllable text generation. While style transfer, and sentiment transfer in particular, has guided most of the prior research in this area, we have shown that the associated metrics and datasets are not entirely representative of the goals of learning disentangled text representations. We re-purposed an existing NLG dataset for this task instead, and performed a stronger evaluation of current models for disentangled representation learning. We have also shown that heavy supervision is needed along each aspect to obtain useful representations. Improvements in variational generative models that can overcome issues of posterior collapse and the use of decoding constraints stronger than the reconstruction loss would greatly benefit such models.

Acknowledgements

We thank the reviewers for their helpful comments. This work was financially supported by Natural Sciences and Engineering Research Council of Canada, and carried out with resources provided by the Vector Institute.

References

Vikash Balasubramanian, Ivan Kobyzev, Hareesh Bahuleyan, Ilya Shapiro, and Olga Vechtomova. 2020. Polarized-VAE: Proximity based disentangled representation learning for text generation. *arXiv preprint arXiv:2004.10809*.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6008–6019.

Keith Carlson, Allen Riddell, and Daniel Rockmore. 2018. Evaluating prose style transfer with the Bible. *Royal Society Open Science*, 5(10):171920.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2453–2464.

Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. 2020. Improving disentangled text representation learning with information-theoretic guidance. *arXiv preprint arXiv:2006.00693*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Vrindavan Harrison, Lena Reed, Shereen Oraby, and Marilyn Walker. 2019. Maximizing stylistic control and semantic accuracy in NLG: Personality variation and discourse contrast. In *Proceedings of the 1st Workshop on Discourse Structure in Neural NLG*, pages 1–12.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. *arXiv preprint arXiv:2002.03912*.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Guillaume Lample, Sandeep Subramanian, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *7th International Conference on Learning Representations (ICLR 2019)*.

Shereen Oraby, Lena Reed, Shubhangi Tandon, TS Sharath, Stephanie Lukin, and Marilyn Walker. 2018. Controlling personality-based stylistic variation with neural natural language generators. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 180–190.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876.

Sudha Rao and Joel Tetreault. 2018. Dear Sir or Madam, May I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2019. Adversarial decomposition of text representation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 815–825.

Tianxiao Shen, Jonas Mueller, Regina Barzilay, and Tommi Jaakkola. 2020. Educating text autoencoders: Latent representation guidance via denoising. In *International Conference on Machine Learning*, pages 8719–8729. PMLR.

Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. 2020. On variational learning of controllable representations for text without supervision. In *International Conference on Machine Learning*, pages 10534–10543. PMLR.

Ivan P Yamshchikov, Viacheslav Shibaev, Aleksander Nagaev, Jürgen Jost, and Alexey Tikhonov. 2019. Decomposing textual information for style transfer. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 128–137.

Appendices

A Parallel Style Datasets

A.1 GYAFC Corpus

The *Grammarly's Yahoo Answers Formality Corpus*, or GYAFC for short, is a benchmark corpus for formality style transfer in NLP[2]. It consists of a total of 120,000 informal / formal sentence pairs, split into training, validation, and test sets.

Sentences were initially sampled from the Yahoo Answers L6 corpus, and formal and informal rewrites from each were collected from workers on Amazon Mechanical Turk (Rao and Tetreault, 2018). Table 6 shows example paraphrases from this corpus.

A.2 Bible Dataset

More than 30 English translations of the Bible have been published over the course of four centuries, the earliest being the King James Version of 1611. These versions are all highly parallel, aligned by verse, and are high-quality translations due to the importance of the source. Carlson et al. (2018) identified 8 of these versions that are in the public domain and released aligned corpora for each[3]. Table 7 shows a sample verse paraphrased in each of the 8 versions we consider. Each version consists of 31,096 verses, giving us close to 870,000 paraphrase pairs. We first split this into an 80–20 development–test split; the development set is further split into training and validation sets with the same ratio.

Formal	I'd say it is punk though.
Informal	However, I do believe it to be punk.
Informal	Gotta see both sides of the story.
Formal	You have to consider both sides of the story.

Table 6: Sample paraphrases from the GYAFC dataset.

B t-SNE Visualization

t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction technique useful for visualizing high-dimensional data. Figure 2 shows t-SNE plots of the form vectors (left column) and content vectors (right column) for sentences in the test set of the PersonageNLG dataset, for each of the loss function combinations we tested. Adding the supervised losses for form successfully groups the form vectors together into five clusters for each of the personality classes. While content vectors also show some clustering with the adversarial and motivational losses, paraphrase losses here are the most effective at grouping them into neat clusters for each of the unique meaning representations in our test set.

C More Results

C.1 Detailed Disentanglement Evaluation

In Table 8, we present a more detailed evaluation on the disentanglement metrics for our models. Here, the Classification column presents both same-aspect and cross-aspect F_1 scores. Higher scores for the former and lower scores for the latter indicate better disentanglement.

We notice that form information is not effectively removed from the content representations, as evidenced by the higher F_{sim} scores for the content vectors z_c. This is a consequence of the weaker label-based proxy used for style, as opposed to the Meaning Representation-based attribute proxy for content.

C.2 Style Transfer Outputs

Table 9 shows sample outputs from the style transfer experiments on PersonageNLG. The model used is the best performing proxy-based model L_{proxy}, with motivational and adversarial losses for both style and content. Two paraphrases with different styles are first encoded into their form and content vectors. The output is generated by passing the form vector of the first sentence and the content vector of the second to the decoder. We see that the model transfers the form attributes quite well across the inputs, but content attributes are not retained perfectly.

---

[2] https://github.com/raosudha89/GYAFC-corpus

[3] https://github.com/keithecarlson/StyleTransferBibleData

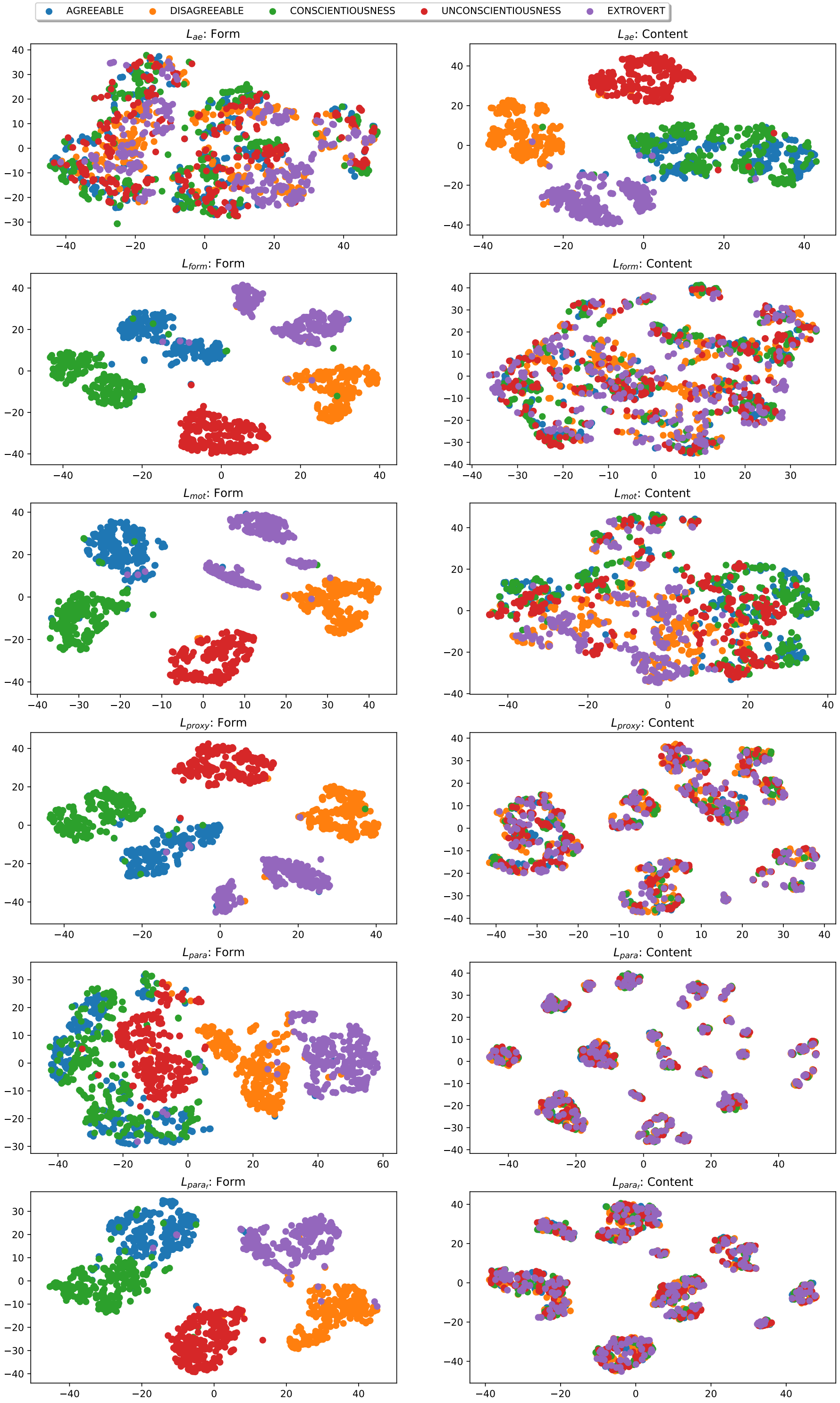

Figure 2: t-SNE visualization of form and content vectors for the PersonageNLG dataset, for each of our models. We see that the paraphrase losses enable a clean clustering of the meaning representations across stylistic variations. The domination of extrovert (purple) in some of the conditions is an artifact of the visualization when points fall in the same place.

Version	Verse
KJV	The heart of the prudent getteth knowledge; and the ear of the wise seeketh knowledge.
ASV	The heart of the prudent getteth knowledge; And the ear of the wise seeketh knowledge.
BBE	The heart of the man of good sense gets knowledge; the ear of the wise is searching for knowledge.
DARBY	The heart of an intelligent getteth knowledge, and the ear of the wise seeketh knowledge.
DRA	A wise heart shall acquire knowledge: and the ear of the wise seeketh instruction.
LEB	An intelligent mind will acquire knowledge, and the ear of the wise will seek knowledge.
WEB	The heart of the discerning gets knowledge. The ear of the wise seeks knowledge.
YLT	The heart of the intelligent getteth knowledge, And the ear of the wise seeketh knowledge.

Table 7: The same verse (Proverbs 18:15) paraphrased in 8 different diachronic versions of the Bible, from the Bible dataset: the King James Version (KJV, 1611), American Standard Version (ASV, 1901), Bible in Basic English (BBE, 1965), Darby Bible (DARBY, 1890), Douay-Rheims edition (DRA, 1899), Lexham English Bible (LEB, 2010), World English Bible (WEB, 2000), and Young's Literal Translation (YLT, 1862).

Model	Classification: F_1				Retrieval			
	Form		Content		Form: F_{sim}		Content: C_{sim}	
	$z_f \uparrow$	$z_c \downarrow$	$z_c \uparrow$	$z_f \downarrow$	$z_f \uparrow$	$z_c \downarrow$	$z_c \uparrow$	$z_f \downarrow$
L_{ae}	0.73	0.96	0.58	0.73	0.57	0.95	0.85	0.70
L_{form}	0.98	0.29	0.62	0.62	0.70	0.90	0.93	0.55
L_{mot}	0.98	0.65	0.92	0.59	0.65	0.90	0.98	0.63
L_{proxy}	0.98	0.23	0.92	0.60	0.70	0.85	0.99	0.54
L_{para}	0.95	0.28	0.80	0.70	0.68	0.93	0.98	0.55
L_{para_f}	0.98	0.21	0.75	0.63	0.69	0.87	0.97	0.54

Table 8: Classification and Retrieval scores that measure the quality of disentanglement of information for each of our models, evaluated on the PersonageNLG dataset

Input (Style A)	nameVariable is near nearVariable pal, nameVariable is a restaurant and it isn't family friendly, also the rating is average, you know!
Target (Style B)	You want to know more about nameVariable? Yeah, it isn't rather family friendly with an average rating, also it is sort of near nearVariable, also it is a restaurant, you see?
Output (Style A $\rightarrow$ Style B)	You want to know more about nameVariable? Oh it is sort of near nearVariable, also it is a restaurant, also it isn't family friendly, you see
Input	nameVariable is moderately priced, also it's in riverside. It is near nearVariable. It is a pub. it's an Italian restaurant. oh God basically, nameVariable is kid friendly.
Target	Yeah, err... I am not sure. nameVariable is an Italian place near nearVariable in riverside, damn kid friendly and moderately priced and nameVariable is a pub.
Output	Yeah, I am not sure. nameVariable is darn moderately priced in city centre near nearVariable, also it is a coffee shop, also it isn't kid friendly

Table 9: Sample style transfer outputs for the best performing proxy-based model, L_{para}, on the PersonageNLG Dataset.

Injecting Knowledge Base Information into End-to-End Joint Entity and Relation Extraction and Coreference Resolution

Severine Verlinden[*], **Klim Zaporojets**[*], **Johannes Deleu**,
Thomas Demeester, **Chris Develder**
Ghent University – imec, IDLab
Ghent, Belgium
`{first_name.last_name}@ugent.be`

Abstract

We consider a joint information extraction (IE) model, solving named entity recognition, coreference resolution and relation extraction jointly over the whole document. In particular, we study how to inject information from a knowledge base (KB) in such IE model, based on unsupervised entity linking. The used KB entity representations are learned from either (i) hyperlinked text documents (Wikipedia), or (ii) a knowledge graph (Wikidata), and appear complementary in raising IE performance. Representations of corresponding entity linking (EL) candidates are added to text span representations of the input document, and we experiment with (i) taking a weighted average of the EL candidate representations based on their prior (in Wikipedia), and (ii) using an attention scheme over the EL candidate list. Results demonstrate an increase of up to 5% F1-score for the evaluated IE tasks on two datasets. Despite a strong performance of the prior-based model, our quantitative and qualitative analysis reveals the advantage of using the attention-based approach.

1 Introduction

Information extraction (IE) comprises several subtasks, e.g., named entity recognition (NER), coreference resolution (coref), relation extraction (RE). State-of-the-art results mainly report performance on single tasks, usually solving them on a sentence level (especially NER, RE). However, in practice, IE system decisions should be consistent on the document level, e.g., when processing news articles to automatically link entities (aside from potentially learning, e.g., new relations). Yet, the challenge of solving the tasks jointly on a document level has not received as much attention and remains hard (Durrett and Klein, 2014; Yao et al., 2019; Zaporojets et al., 2021).

On the other hand, it is well established that IE models benefit from incorporating background information of knowledge bases (KBs). Still, so far this has been shown from the perspective of solving individual tasks such as relation classification or entity typing (e.g., Peters et al. (2019); Liu et al. (2020)). Integrating KBs in joint models, realizing and analyzing the more complex end-to-end setting, has been left unexplored.

In terms of the nature of KBs adopted in IE, current approaches use either (i) structured knowledge *graphs* comprising `(subj,rel,obj)` triples, e.g., Wikidata (Yang and Mitchell, 2017; Han et al., 2018; Zhang et al., 2019), or (ii) *textual* descriptions, usually in hyperlinked documents, e.g., Wikipedia (Martins et al., 2019; Yamada et al., 2020). It has not been established to what extent KB-text and KB-graph entity representations complement each other in boosting IE performance.

We address both research gaps of (a) integrating KB information into a joint end-to-end IE model for solving named entity recognition, coreference resolution and relation extraction, and (b) analyzing what KB representation is more beneficial for IE, either *KB-graph* trained on Wikidata, or *KB–text* trained directly on Wikipedia. We particularly contribute: (i) a first span-based end-to-end architecture incorporating KB knowledge in a joint entity-centric setting, exploiting unsupervised entity linking (EL) to select KB entity candidates, (ii) exploration of prior- and attention-based mechanisms to combine the EL candidate representations into the model, (iii) assessment of the complementarity of KB-graph and KB-text representations, and (iv) consistent gains of up to 5% F1-score when incorporating KB knowledge in 3 document-level IE tasks evaluated on 2 different datasets.

---

[*] Equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1952–1957
August 1–6, 2021.

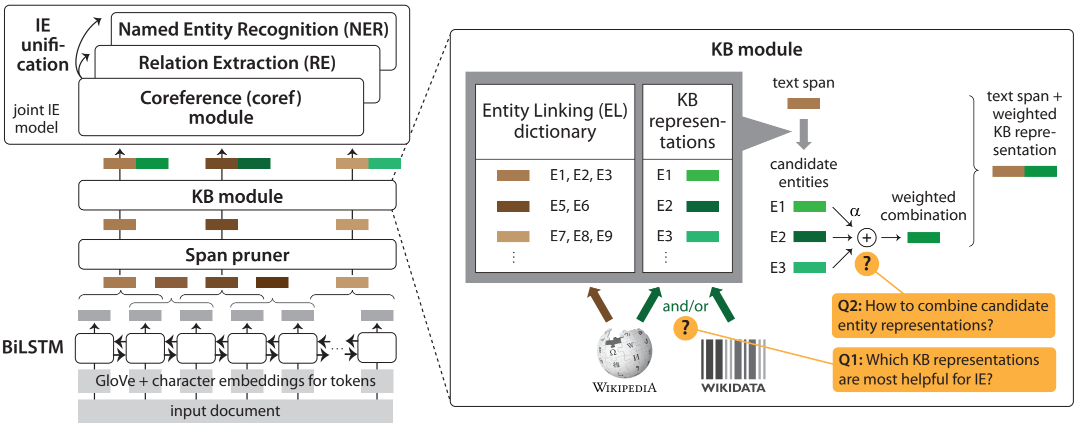

Figure 1: Joint information extraction (IE) model with addition of a knowledge base (KB) module.

2 Model

Figure 1 illustrates our model architecture. Input document tokens are represented using concatenated GloVe (Pennington et al., 2014) and character embeddings (Ma and Hovy, 2016) and pushed through a BiLSTM to obtain contextualized token representations, which are combined into spans. Similar to Luan et al. (2019); Zaporojets et al. (2021), a span pruner limits the number of spans for downstream modules. The *KB module* (§2.2) combines span representations with KB entity representations (§2.1), trained either on Wikidata (*KB-graph*) or Wikipedia (*KB-text*). The KB-enriched span representations then serve as input for joint predictions on downstream IE tasks (§2.3).

2.1 Entity Representations

We experiment with 3 possible entity representations: *KB-text*, *KB-graph*, and concatenating *both*.

KB-text: We follow Yamada et al. (2016) to obtain the entity representations using a skip-gram architecture (Mikolov et al., 2013a,b), training to jointly predict (i) the linked entities (through Wikipedia hyperlinks) given the target entity, and (ii) the neighboring words for a given entity hyperlink.

KB-graph: We adopt Joulin et al. (2017) to train the entity embeddings directly on Wikidata triples `(subj,rel,obj)` by optimizing a linear classifier to predict the `obj` entity from the `subj` entity and the relation type `rel`.

2.2 KB module

For a span s_i from token l to r, we obtain the representation $\mathbf{g}_i$ as input to the KB module by concatenating the respective hidden LSTM states $\mathbf{h}_l$ and $\mathbf{h}_r$, and an embedding $\boldsymbol{\psi}_{r-l}$ for the corresponding span width $r-l$:

$$\mathbf{g}_i = [\mathbf{h}_l; \mathbf{h}_r; \boldsymbol{\psi}_{r-l}]. \tag{1}$$

We look up a given span s_i in a dictionary built from Wikipedia, to determine its candidate entities set[1] C_i, as well as the prior probability p_{ij} for each $c_{ij} \in C_i$, as per Yamada et al. (2016, §3).

To combine the KB candidates c_{ij}, we either use (i) a uniform average (*Uniform*), (ii) the prior weights p_{ij} (*Prior*), (iii) an attention scheme (*Attention*), or (iv) attention with prior information (*AttPrior*). The unnormalized attention scores for *Attention* and *AttPrior* are:

$$\Phi_{\mathit{Attention}}(s_i, c_{ij}, \mathrm{K}) = \mathcal{F}_A\left([\mathbf{g}_i; \boldsymbol{\xi}_{\mathrm{K}}(c_{ij})]\right) \tag{2}$$

$$\Phi_{\mathit{AttPrior}}(s_i, c_{ij}, \mathrm{K}) = \mathcal{F}_{AP}\big([\mathbf{g}_i; \boldsymbol{\xi}_{\mathrm{K}}(c_{ij}); p_{ij}]\big) \tag{3}$$

where $\mathrm{K} \in \{\mathit{KB\text{-}text}, \mathit{KB\text{-}graph}, \mathit{both}\}$ refers to the entity representations from §2.1, $\boldsymbol{\xi}_{\mathrm{K}}$ returns such representation for c_{ij}, and $\mathcal{F}_*$ is a feed-forward neural network (FFNN). The KB representation for span s_i is a weighted average of its candidates C_i:

$$\mathbf{e}_i^{\mathrm{K}} = \sum_{c_{ij} \in C_i} \alpha_{ij} \cdot \boldsymbol{\xi}_{\mathrm{K}}(c_{ij}) \tag{4}$$

where weights α_{ij} either are uniform ($1/\left|C_i\right|$), the prior p_{ij}, or softmax-normalized attention scores (softmax over Φ from eq. (2) or eq. (3)). The concatenation $[\mathbf{g}_i; \mathbf{e}_i^{\mathrm{K}}]$ forms the KB-enriched representation for span s_i, as input for IE modules (§2.3).

2.3 Joint IE model

The joint IE model comprises 3 modules (Fig. 1) using the same KB-enriched representations $[\mathbf{g}_i; \mathbf{e}_i^{\mathrm{K}}]$,

---

[1] We limit this to the 16 most frequent ones.

Dataset	# Entity clusters	# Entity types	# Relations	# Relation types
DWIE	23,130	311	21,749	65
DocRED	98,610	6	50,503	96

Table 1: Dataset statistics.

and using a weighted combination of the 3 module losses to minimize during training. Note that NER and RE are framed as multi-label classification.

NER module: We use a FFNN on each span s_i to produce scores $\mathbf{\Phi}_{\text{NER}}(s_i) \in \mathbb{R}^{|L_E|}$, with L_E the set of possible entity types. At inference, we accept type $l \in L_E$ for span s_i if $\mathbf{\Phi}_{\text{NER}}(s_i)_l > 0$.

Coref module: We use the coreference scheme proposed by Lee et al. (2017), using a FFNN to produce scores $\Phi_{\text{coref}}(s_i, s_j)$: at inference time, the highest scoring antecedent of span s_j is then chosen (potentially s_j itself). Indeed, to allow for singletons we accept self-references (s_j, s_j) if NER predicts the span s_j to be an entity.

RE module: Similar to Luan et al. (2019, 2018), we use a FFNN to produce scores $\mathbf{\Phi}_{\text{RE}}(s_i, s_j) \in \mathbb{R}^{|L_R|}$ for each pair of spans (s_i, s_j), with L_R the set of relation types. We accept relation $l \in L_R$ for pair (s_i, s_j) if $\mathbf{\Phi}_{\text{RE}}(s_i, s_j)_l > 0$.

IE unification: Above modules make span level predictions. We obtain entity-centric predictions using the coref clusters, by assigning the union of predicted entity/relation types within a coref cluster to all its members, as do Zaporojets et al. (2021).

3 Experimental Setup

We evaluate our proposed models[2] on entity-centric multi-task datasets, summarized in Table 1: DWIE (Zaporojets et al., 2021) and DocRED (Yao et al., 2019). We report on coreference resolution (coref), NER and relation extraction (RE). For coref, we report the average of 3 common F1 scores, as implemented by Pradhan et al. (2014): MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and CEAF_e (Luo, 2005). Since we focus on entity-centric, document-level IE, for NER and RE we use *hard* metrics (Zaporojets et al., 2021) on the level of entity clusters (i.e., aforementioned coref clusters): predictions are counted as correct only if (i) all mentions (with exact boundary match) are present in the entity cluster, and (ii) the predicted entity type (for NER) or relation type between two clusters (for RE) is correct.

Our experiments address 2 main questions (see Fig. 1): **(Q1)** Which type of KB representation is most helpful for IE (*KB-text*, *KB-graph*, or *both*; see §2.1)? **(Q2)** Which weighting scheme to use for α (*Uniform*, *Prior*, *Attention*, *AttPrior*; see §2.2)?

4 Results

We summarize the comparison of various model choices for both DWIE and DocRED datasets in Table 2. First, looking into **(Q1)**, we note that including background information from *KB-graph* and *KB-text* significantly boosts performance compared to the *Baseline* without any KB. Additionally, our model outperforms the results from Zaporojets et al. (2021) (not listed in the table) by about 2 percentage points F1, using the same input (GloVe) representations. Furthermore, we observe a general improvement in results when combining *both* representations, suggesting that a (hyper)text corpus (Wikipedia) and a knowledge graph (Wikidata) embed complementary information for raising IE performance.

Deeper analysis reveals that adding KB representations mainly benefits performance for "rare" entity types: e.g., limiting the test set to entity types that occur ≤50 times in the training set for DWIE, compared to *Baseline*, F1 for NER goes up by +13.9 for *KB-both* with *AttPrior*, while the benefit gradually decreases for more frequently occurring entity types. For RE, we note that overall we also see a clear performance gain from adding KB information (e.g., +5.1% F1 for *both* KB sources with *AttProp* compared to *Baseline* for DWIE), yet the boost is not as clear for relations with fewer training instances. (The latter makes sense, since we inject KB representations of entities rather than explicitly also for relations; we leave studying adding relation embedding information for future work.)

Second, for **(Q2)**, we note that the *AttPrior* scheme is the overall winner among the different EL candidate weigthing schemes. We observed that in terms of ranking EL candidates, *Prior* performs quite well on DWIE — for 86.5% of entity mentions it assigns the highest score to the correct EL candidate, while *Attention* and *AttPrior* achieve it for 46.2%, resp. 77.2% of the mentions — which basically confirms that DWIE has a similar entity distribution as Wikipedia.[3] Yet, it seems necessary to include alternative candidates, and

---

[2] Code and models available at `https://github.com/klimzaporojets/e2e-kb-ie`.

[3] DWIE is a news article corpus.

		DWIE			DocRED		
KB Source	Setup	Coref	NER	RE	Coref	NER	RE
–	Baseline	90.0±0.2	71.7±0.5	47.0±1.4	81.9±0.3	68.5±0.3	23.5±0.6
KB-text	Uniform	90.7±0.2	73.5±0.5	48.5±1.1	82.9±0.1	70.7±0.2	24.5±0.3
	Attention	**90.7±0.3**	73.4±0.8	49.0±0.4	**83.4±0.1**	71.2±0.1	24.5±0.3
	AttPrior	90.7±0.3	73.7±0.6	**49.6±0.8**	83.2±0.2	**71.3±0.2**	24.8±0.4
	Prior	90.7±0.2	**73.8±0.5**	49.4±0.4	82.9±0.2	70.9±0.3	**25.3±0.4**
KB-graph	Uniform	91.0±0.3	73.6±0.4	48.0±1.2	83.3±0.2	71.1±0.2	24.9±0.2
	Attention	91.2±0.3	73.9±0.5	50.1±1.1	**83.7±0.1**	**71.6±0.1**	25.0±0.4
	AttPrior	**91.3±0.2**	**74.6±0.3**	**50.5±1.0**	83.5±0.3	71.5±0.2	25.1±0.2
	Prior	90.8±0.3	73.6±0.6	49.6±1.1	83.4±0.1	71.1±0.1	**25.2±0.2**
both (KB-graph + KB-text)	Uniform	91.1±0.1	74.1±0.5	49.3±0.5	83.5±0.1	71.3±0.2	24.8±0.1
	Attention	91.2±0.3	74.3±0.6	51.3±1.3	83.5±0.2	71.5±0.1	24.8±0.3
	AttPrior	**91.5±0.2**	**75.0±0.4**	**52.1±1.2**	**83.6±0.2**	**71.8±0.3**	**25.7±0.7**
	Prior	90.8±0.1	73.8±0.2	49.8±1.2	83.2±0.1	71.2±0.1	25.1±0.3

Table 2: Main results of the experiments in F1 scores grouped by the background KB source. We report Avg. F1 scores of MUC, B^3 and $CEAF_e$ for Coref, and hard F1 metrics for NER and RE. **Bold** font indicates the best results for each of the different *KB source* types. Additionally, the best overall results are underlined.

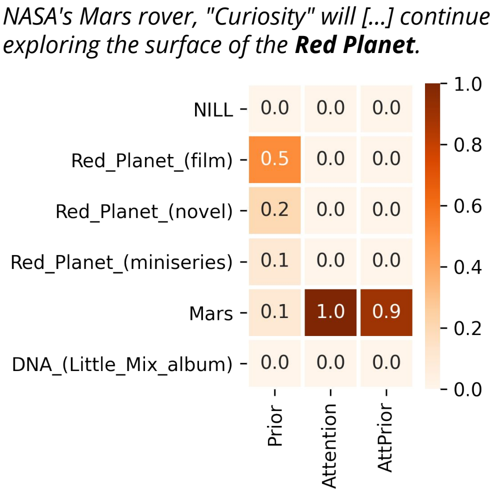

Figure 2: Illustration of EL candidate weighting: the α weights for top candidates for "Red Planet" from the example sentence at the top. Attention-based weighting (*Attention*, *AttPrior*) correctly identify the "Mars" entity, while the Wikipedia-based *Prior* fails, as most of Wikipedia's "Red Planet" links refer to the film.

the attention-based schemes thus can correct EL mistakes of *Prior*, as illustrated in Fig. 2. This correction leads to a resulting boost for the IE tasks as reported in Table 2. E.g., we found that for DWIE, looking at clusters with entity mentions for which *Prior* makes wrong EL predictions, the *AttPrior* weighting scheme retrieves +3.7% more of the gold standard annotated named entities (as opposed to just +0.6% in the clusters with correct *Prior* EL candidates). Perfecting the EL prediction would potentially boost IE performance even more.

5 Related Work

As stated earlier, we studied how to integrate (i) knowledge base information into IE, and particularly (ii) end-to-end IE combining multiple tasks (NER, relation extraction, coreference resolution), while (iii) taking an entity-centric perspective, i.e., focus on making consistent decisions on the document level. For (i), integrating KB into IE has been applied for individual tasks: relation classification (Poerner et al., 2020; Zhang et al., 2019; Yang and Mitchell, 2017), entity typing (Peters et al., 2019) and NER (Yamada et al., 2020). For (ii), recently span-based architectures (Lee et al., 2017; Luan et al., 2019; Wadden et al., 2019; Fei et al., 2020) have been proposed. Our work unifies the KB integration concept into such span-based IE system, in particular an entity-centric one (as per (iii)), building on Jia et al. (2019); Zaporojets et al. (2021). For the KB integration approach, we exploit entity representations trained on a hypertext corpus, as in (Yamada et al., 2016; Ganea and Hofmann, 2017; Yamada et al., 2020) or learnt from a knowledge graph (Yang and Mitchell, 2017; Han et al., 2018; Zhang et al., 2019). Our results show that both offer complementary value for IE. Similarly to our work, Yamada and Shindo (2019) also explore using an attention-weighted combination of entity representations, but they use it to build a full document representation (with mentions having the entities as candidates) for a text classification task. In contrast, our span-based attention model is able to "inject" knowledge in each of the mentions separately, for more fine-grained downstream IE tasks that are mention-dependent, e.g., coreference resolution, relation extraction and NER.

6 Conclusion

We propose an end-to-end model for joint IE (NER + relation extraction + coreference resolution) incorporating entity representations from a background knowledge base (KB), using a span-based system. We find that representations built from a knowledge graph and a hypertext corpus are complementary in boosting IE performance. To combine candidate entity representations for text spans, we explore various weighting schemes: an attention-based combination is successful in combining prior frequency information from a hypertext corpus with contextual information to identify the relevant entity, and achieves highest IE performance.

Acknowledgments

Part of the research leading to these results has received funding from (i) the European Union's Horizon 2020 research and innovation programme under grant agreement no. 761488 for the CPN project,[4] and (ii) the Flemish Government under the programme "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen".

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1998 International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference (LREC 1998)*, pages 563–566.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics (TACL 2014)*, 2:477–490.

Hao Fei, Yafeng Ren, and Donghong Ji. 2020. Boundaries and edges rethinking: An end-to-end neural model for overlapping entity relation extraction. *Information Processing & Management*, 57(6):102311.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2619–2629.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of the 2018 Conference on Artificial Intelligence (AAAI 2018)*, volume 32.

Robin Jia, Cliff Wong, and Hoifung Poon. 2019. Document-level n-ary relation extraction with multiscale representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 3693–3704.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomas Mikolov. 2017. Fast linear model for knowledge graph embeddings. *arXiv:1710.10881*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 188–197.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling language representation with knowledge graph. In *Proceedings of the 2020 Conference on Artificial Intelligence (AAAI 2020)*, pages 2901–2908.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 3219–3232.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 3036–3046.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 25–32.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 2016 Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1064–1074.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2019. Joint learning of named entity recognition and entity linking. In *Proceedings of the 2019 Annual Meeting of the Association for Computational Linguistics: Student Research Workshop (ACL 2019, SRW)*, page 190.

---

[4] `https://www.projectcpn.eu/`

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the 2013 International Conference on Learning Representations (ICLR 2013)*, pages 1–12.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 2013 International Conference on Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pages 43–54.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 803–818.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 2014 Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 30–35.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 1995 conference on Message understanding (MUC6, 1995)*, pages 45–52.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pages 5788–5793.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 6442–6454.

Ikuya Yamada and Hiroyuki Shindo. 2019. Neural attentive bag-of-entities model for text classification. In *Proceedings of the 2019 Conference on Computational Natural Language Learning (CoNLL 2019)*, pages 563–573.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 2016 SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pages 250–259.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 2017 Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1436–1446.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 2019 Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 764–777.

Klim Zaporojets, Johannes Deleu, Chris Develder, and Thomas Demeester. 2021. DWIE: An entity-centric dataset for multi-task document-level information extraction. *Information Processing & Management*, 58(4):102563.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 3016–3025.

Knowing More About Questions Can Help: Improving Calibration in Question Answering

Shujian Zhang **Chengyue Gong** **Eunsol Choi**
The University of Texas at Austin
`szhang19@utexas.edu, {cygong,eunsol}@cs.utexas.edu`

Abstract

We study calibration in question answering, estimating whether model correctly predicts answer for each question. Unlike prior work which mainly rely on the model's confidence score, our calibrator incorporates information about the input example (e.g., question and the evidence context). Together with data augmentation via back translation, our simple approach achieves 5-10% gains in calibration accuracy on reading comprehension benchmarks. Furthermore, we present the first calibration study in the open retrieval setting, comparing the calibration accuracy of retrieval-based span prediction models and answer generation models. Here again, our approach shows consistent gains over calibrators relying on the model confidence. Our simple and efficient calibrator can be easily adapted to many tasks and model architectures, showing robust gains in all settings.[1]

1 Introduction

Despite rapid progress in AI models, building a question answering (QA) system that can always correctly answer any given query is beyond our reach. Thus, questioners have to interpret the model prediction, deciding whether to trust it. We study providing an accurate estimate of the correctness of model prediction for each example at test time. As making incorrect predictions can be much more costly than making no prediction (e.g., missing diagnosis is much more costly than querying human experts), calibrators can bring practical benefits (Kamath et al., 2020).

Existing work on calibration focuses on mdoel confidence, such as the max probability of the predicted class (Guo et al., 2017; Desai and Durrett, 2020). Unlike classification tasks, question answering explores large output space, either through answer generation (Raffel et al., 2020; Lewis et al., 2020) or selecting a span from provided documents (Rajpurkar et al., 2016). In both settings, optimal decoding is often prohibitively expensive, and heuristic decoding is a standard practice (Seo et al., 2017). Thus, relying on the model's confidence score alone is not sufficient for calibration (Kumar and Sarawagi, 2019).

Nonetheless, prior work (Kamath et al., 2020; Jagannatha and Yu, 2020) relied heavily on model confidence, such as the max probability of the predicted answer, together with a handful of manually crafted features containing little information about the input, such as the length of the question. We empower the calibrator by introducing an input example embedding from a pre-trained language model (Alberti et al., 2019; Liu et al., 2019) fine-tuned on QA supervision data as additional features. With this simple and general feature, calibrator can identify questions regarding rare entities or examples with little lexical overlap between the question and the context. We bring further gains by paraphrasing questions or contexts respectively through back translation (Sennrich et al., 2016), providing lexical variations of the question and the context and enriching the feature space.

We evaluate our calibrator with internal metrics (i.e., calibration accuracy) and external metrics (i.e., impact on QA performance). We first evaluate calibrators in reading comprehension settings introduced in Kamath et al. (2020) – in-domain (Rajpurkar et al., 2016; Kwiatkowski et al., 2019), out of domain (Fisch et al., 2019), and adversarial (Jia and Liang, 2017). Then, we expand calibration study to more challenging open retrieval QA setting (Voorhees and Tice, 2000; Chen et al., 2017), where a system is not provided with an evidence document. We adapt our calibrator for state-of-the-art generation based (Raffel et al., 2020) and extractive (retrieve-and-predict) QA models (Karpukhin

---

[1] Code is available at `https://github.com/szhang42/Calibration_qa`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1958–1970
August 1–6, 2021. ©2021 Association for Computational Linguistics

et al., 2020), showing gains in both models. While calibration accuracy is higher in the generation based model, the extractive method provides better answer coverage above fixed accuracy. Lastly, we use calibrator as a reranker for the answer span candidates in an extractive open retrieval QA model (Karpukhin et al., 2020), showing modest gains. We provide rich ablation studies on design choices for our calibrator, such as the choice of base model to derive input example encoding. Our simple input example embedding from pretrained language models shows consistent gains in all settings and datasets. Without any manual engineering specific to the question answering task, our calibrator could be easily adapted to other tasks with rich output space.

2 Problem Definition

We estimate how the models' prediction confidence aligns with the empirical likelihood of correctness (Brier, 1950). Formally, a calibrator f takes the input example x_i and the trained model M_θ and identifies whether the model's prediction is correct or not. We treat the correctness as binary (i.e., answer string exact match) for simplicity, instead of partial credit (e.g., token level F1 score). We study two settings: reading comprehension (RC) and open retrieval QA. In RC, an input example x_i will be a context c_i and the question q_i, and in open domain QA, an input example will be a corpus C and the question q_i.

We use the same metrics to evaluate the performance of the calibrator f in the two settings.

2.1 Metric: Calibrator performance

Accuracy: Given evaluation data $D_{\text{eval}} = \{(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)\}$ and a learned model M_θ, we define the accuracy of the calibrator f as:

$$\text{acc}(f) = \sum_{i=1}^{N} \mathbb{I}\left\{ f(x_i, M_\theta) = \mathbb{I}[M_\theta(x_i) = y_i] \right\}.$$

AUROC: Based on the above definition of the accuracy of the calibrator f, we computes the coverage – fraction of evaluation data D_{eval} that model makes prediction on – and risk, the error at that coverage. We plot risk versus coverage graph, and measure the area under the curve, i.e., AUROC (Area Under the Receiver Operating Characteristic Curve) (Hanley and McNeil, 1982).

2.2 Metric: End task performance

We measure how the calibrator performance impacts QA performances. First, we study selective QA setting – where we use calibrator score to decide which examples from D_{eval} to make predictions.

For the extractive model for open retrieval QA (Karpukhin et al., 2020), where multiple answer candidates are given, we further evaluate the performance of calibrator as a reranker and measure the answer span exact match (EM) score.

Selective QA (coverage at fixed accuracy): We use the calibrator score to rank the *examples* in the evaluation data. Specifically, we use the calibrator's confidence for the top answer candidate instead of model score to decide which examples in D_{eval} the model answers most confidently. Then, we report the percentage of evaluation data that can be predicted while maintaining threshold accuracy (80%), following prior work (Kamath et al., 2020).

Open Retrieval QA (top-N accuracy): We use the calibrator score to rank the *answer candidates* for each evaluation example, similar to how candidate translations are reranked in machine translation (Shen et al., 2004). We first retrieve answer candidates from multiple paragraphs and utilize the calibrator to override the model's prediction. The calibrator scores the top N answer candidates and outputs the answer with the highest confidence score instead of the answer with the highest model score. Our calibrator can be added as last step for any open retrieval QA systems which generates multiple answer candidates without retraining the model. We evaluate the top 1 exact match accuracy and the top 5 exact match accuracy after re ranking with our calibrator score.

3 Methods

We propose two general approaches to improve binary calibrator: new feature vector, a dense representation of the input example (Section 3.2) and data augmentation with backtranslation which further improves the new feature vector (Section 3.3). While both are simple, well-established formula for improving end tasks in NLP, neither has been explored in the context of calibration, as prior work assumed model confidence score is the most prominent signal. We follow prior work (Kamath et al., 2020) for calibrator architecture and focus on improving its feature space.

3.1 Calibrator Architecture

A binary classifier is trained using the gradient boosting library XGBoost (Chen and Guestrin, 2016), which classifies each test example as correctly answered by the base QA model or not. This calibrator does not share its weights with the base QA models. We finetune the following hyperparameters on the development set: colsample by level, colsample by node, colsample by tree, learning rate, and the number of estimators. All calibrators are trained five times, each with different data partitions and random seeds. We report the variances in the results.

3.2 Input Example Embedding Feature From Base QA Model

Prior work uses manually designed features based on the scores to the predicted answer (details in Section 4.2.1). Such features retain little information about input example – e.g, question and the evidence context. Inspired by the recent works in machine learning (e.g. Song et al., 2019; Hendrycks et al., 2019), which use hidden vectors to classify in-domain and out-of-domain data, we introduce an input example embedding, a new feature vector that represent question and (optionally) evidence context to a calibrator.

Our input example embedding is a fixed dimensional vector representing an input example, similar to sentence embeddings (Conneau et al., 2017). It differs in that the representation is taken from the final layer of base QA model, which is trained with supervision from question answering data and it encodes question and (optional) evidence context simultaneously. In Section 6.1, we report minor performance degradation from using embeddings from generic pretrained language model instead.

Each base model processes input example, either query q_i or query, context pair (q_i, c_i) to generate a sequence of hidden vectors, which will be compressed into a fixed dimensional vector to be used as calibrator feature.[2] We denote the input example as a sequence of tokens $\mathbf{t} = (t_0, t_1, \cdots, t_n)$ where n is the length of the input. We pass the sequence $\mathbf{t}$ through base QA model and get $(\mathbf{h}_0, \mathbf{h}_1, \cdots, \mathbf{h}_n)$ where $\mathbf{h}_i$ is the corresponding final-layer hidden state of t_i, and $\mathbf{h}_i = (\mathbf{h}_{i,0}, \cdots, \mathbf{h}_{i,m})$ where m is the number of hidden dimensions. Then, we get the m-dimensional feature vector

$$\phi(q_i, c_i) = [\frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_{i,0}; \cdots ; \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_{i,m})], \quad (1)$$

where each dimension is an average across the length n. We then train a binary classifier using these features as a calibrator. We now describe our base QA models to get this hidden representations.

3.2.1 Base QA Model

We use standard span prediction architecture for RC, and a generation based model and an extractive model for open retrieval QA.

For RC and extractive open retrieval QA model, we use a standard span prediction architecture based on a pretrained language model (Devlin et al., 2018), which predicts start and end index of the answer span separately with softmax layer. The output hidden vector sequence will equal the sum of the length of question and the length of evidence context. For the open retrieval QA setting, the extractive model first retrieves a passage from the corpus and predicts an answer span from it. We use the best model from dense passage retrieval (DPR)(Karpukhin et al., 2020).[3] Specifically, this model retrieves the top 100 retrieved passages as input and trains a span prediction model, which optimizes a softmax cross-entropy loss to select the correct passage among the candidates, and the answer span prediction loss. The model then selects the answer span with the highest answer span score (sum of the start and end logit score) from the passage with the highest passage score. In this setting, $\mathbf{t}$ is a concatenation of question q_i and the context c_i.

For generation based model, we use a sequence-to-sequence (seq2seq) model, specifically T5-small (Raffel et al., 2020), which takes the question as an input and generates answer tokens. For this base QA model, $\mathbf{t}$ only consists of the query since the context is not provided.

Data For all experiments in RC, we train the model on the SQuAD 1.1 dataset. For open retrieval QA, models are trained on the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019) following the data split from Lee et al. (2019).

3.3 Data Augmentation Via Paraphrasing

Paraphrase generation can improve QA models (Yu et al., 2018) by handling language vari-

---

[2] For simplicity, we write equations with (q_i, c_i) pair as an input, when only query is provided (e.g., generation based open retrieval QA method) c_i is empty.

[3] `https://github.com/facebookresearch/DPR`

Task	Setup	Base QA model (train)	Calibrator (train & dev)	Calibrator (test)
Standard RC	In domain	SQuAD 1.1	SQuAD 1.1 + HotpotQA	SQuAD 1.1 + HotpotQA
	Out of domain		SQuAD 1.1 + HotpotQA	Other MRQA
	In domain		SQuAD 1.1 + NQ	SQuAD 1.1 + NQ
	Out of domain		SQuAD 1.1 + NQ	Other MRQA
Adversarial RC	In domain		SQuAD 1.1 Adversarial	SQuAD1.1 Adversarial
	Out of domain		MRQA	SQuAD1.1 Adversarial
Unanswerable RC	In domain		SQuAD 2.0	SQuAD 2.0
	Out of domain		MRQA	SQuAD 2.0
Open Retrieval QA	In domain	NQ Training Set		NQ Test Set

Table 1: Experiment Configuration. In domain / Out of domain distinguishes whether the training data for calibrator is different from the test data.

ation. Compared to sentence retrieval (Du et al., 2020) and language model based example generation (Anaby-Tavor et al., 2020), backtranslation can capture the ambiguity of questions and answer(Singh et al., 2019). Given a (q_i, c_i) pair, we use back translation (Sennrich et al., 2016) to generate paraphrases of the question q'_i from q_i and the evidence context c'_i from c_i.

We use standard transformer-based neural machine translation models (Junczys-Dowmunt et al., 2018) trained on WMT dataset.[4] We first translate the original sentences to a pivot language and then translate them back to the source language. To guarantee translation quality, French and German are used as the pivot languages. We use beam search decoding with beam size as 4 and truncate the context length to 512, as the reading comprehension model truncates the context anyway. We analyze the quality of backtranslation in Section 6.2.

We denote (q'_i, c_i) as $\mathbf{t}^q = (t^q_0, \cdots, t^q_{n_q})$ and (q_i, c'_i) as $\mathbf{t}^c = (t^c_0, \cdots, t^c_{n_c})$. Here, n_q and n_c denote the length after backtranslating the question and context, respectively. For $\mathbf{t}^q$ and $\mathbf{t}^c$, we pass them through the base QA model, get $\mathbf{h}^q$ and $\mathbf{h}^c$, and extract the m-dimensional feature vector as in Eqn (1),

$$\begin{aligned}\phi(q'_i, c_i) &= [\frac{1}{n_q}\sum_{i=1}^{n_q}\mathbf{h}^q_{i,0}, \cdots, \frac{1}{n_q}\sum_{i=1}^{n_q}\mathbf{h}^q_{i,m}],\\ \phi(q_i, c'_i) &= [\frac{1}{n_c}\sum_{i=1}^{n_c}\mathbf{h}^c_{i,0}, \cdots, \frac{1}{n_c}\sum_{i=1}^{n_c}\mathbf{h}^c_{i,m})].\end{aligned} \tag{2}$$

We use the concatenation of the original input example embedding and backtranslated one, $[\phi(q_i, c_i); \phi(q'_i, c_i)]$ and $[\phi(q_i, c_i); \phi(q_i, c'_i)]$ as features. Backtranslating both context and question did not bring further gains, thus the results from such a feature set are not presented. We hypothesize that backtranslating context and question together might introduce too severe noise. We do not use data augmentation for open retrieval QA experiments.

4 Experimental Settings

In this section, we describe the experimental setting, dataset setups and baseline systems. Table 1 summarizes the evaluation scheme. A separate calibrator is trained for each calibrator train data configuration.

4.1 Data

For all in-domain reading comprehension experiments, we randomly split the data into training, development, and test (40%,10%,50%), following regression and classification benchmarks (Asuncion and Newman, 2007). Further, we assume only limited supervised data is available for calibrators, simulating a set up where we have a general QA model and small number of annotated data reserved for calibration.

Standard RC We test two in domain settings and two out of domain settings. We randomly sample 4K examples from each of the datasets included in the training portion of the MRQA shared task (Fisch et al., 2019) (SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019)). We train two calibrators, one with the SQuAD1.1 + HotpotQA datasets and another with the SQuAD1.1 + NQ datasets. For out of domain evaluation, we use four remaining datasets from MRQA shared task training set.

---

[4] `https://huggingface.co/transformers/model_doc/marian.html`

Adversarial RC (SQuAD 1.1 Adversarial) The adversarial examples manipulate the evidence paragraph to change the model prediction but not the gold answer. We sample 2K examples from the development portion of the SQuAD 1.1 (Jia and Liang, 2017) AddSent dataset, which appends an additional sentence that looks similar to the question at the end of the paragraph. For the out-of-domain case, we train the calibrator on 6K examples (1K each sampled from MRQA datasets) and test on adversarial examples.

Unanswerable RC (SQuAD 2.0) We sampled 2K examples from the development portion of the SQuAD 2.0 dataset (Rajpurkar et al., 2018), which contains examples where the answer to the question cannot be derived from the provided context. Crowdworkers posed questions that were impossible to answer based on the paragraph alone while referencing entities in the paragraph and ensuring that a plausible answer is present. For out of domain setting, we train the calibrator on 6K examples (1K each sampled from MRQA datasets) and test on SQuAD 2.0 dataset (same as adversarial RC setting).

Open Retrieval QA We use the open retrieval version of the NQ (Lee et al., 2019). We split its training data 60% and 40% for calibrator training and validation and use the NQ test set for testing.

4.2 Comparison Systems

We summarize the calibrators used in our study in Table 2. All calibrators are trained with the same gradient boosting library XGBoost (Chen and Guestrin, 2016), and they only differ in the feature sets. These calibrators are efficient, trained within a few minutes even with our new feature space.

4.2.1 Reading Comprehension

MaxProb is the simplest baseline that relies on the model's confidence score. The model score is the sum of the logit scores of the start and end of the answer span for reading comprehension. For open retrieval question answering, the model first determines the passage with the highest passage-match score and then extracts the answer span from this passage.

Formally, given the set of answer spans Y, MaxProb with model M_θ estimates confidence on input x_i as:

$$\textit{MaxProb} = \max_{y \in Y} M_\theta \left(y \mid x_i \right),$$

where $M_\theta(y \mid x_i)$ refers to the model score for candidate answer y.

QA model	Calibrator Feature Set	# Features
RC	MaxProb	1
	features: Kamath et al. (2020)	17
	Ours	m
	+ features	$17 + m$
	+ features + $\phi(q_i, c_i')$	$17 + 2m$
	+ features + $\phi(q_i', c_i)$	$17 + 2m$
Extractive (DPR)	Unnormalized Scores	2
	Normalized Scores	2
	Ours + Normalized Score	$2 + 2m$
Generation (T5)	Likelihood	1
	Ours + Likelihood	$1 + m$

Table 2: Comparison Systems: different calibrators explored for three base QA models. The last two QA models are for open retrieval QA task. The dimension of question context embedding is m defined in Eqn (1) (eg. m is 768 for reading comprehension).

Kamath et al. (2020) uses a calibrator based on the following general features: passage length, the predicted answer length, and the top-5 largest softmax probabilities generated by the model. They also use test time dropout (Gal and Ghahramani, 2016): given an input x_i and model M_θ, compute $M_\theta(x_i)$ with K different dropout masks, obtaining prediction distributions $\hat{p}_1, ..., \hat{p}_k$, where each $\hat{p}_i$ is a probability distribution over Y. Two options are used as confidence estimates. First, taking the mean of $\hat{p}_i$ (Lakshminarayanan et al., 2017)

$$\textit{Dropout Mean} = \frac{1}{K} \sum_{i=1}^{K} \hat{p}_i.$$

Second, taking the variance of the $\hat{p}_i$ (Feinman et al., 2017; Smith and Gal, 2018)

$$\textit{Dropout Variance} = \mathrm{Var}\left[\hat{p}_1, \ldots, \hat{p}_K\right].$$

The dimension of *MaxProb*, 2th-5th probability, *Dropout Mean*, *Dropout Variance*, context length and prediction length are 1, 4, 5, 5, 1, 1, respectively. In total, this feature set contains 17 features.

Ours represents a calibrator that is trained with the question context embedding, $\phi(q_i, c_i)$ in Eqn (1). '+ features' refers to augmenting features from (Kamath et al., 2020), described above. Augmenting the feature sets with question context embeddings from backtranslated questions is denoted as '+$\phi(q_i', c_i)$', and augmenting the feature sets with question context embeddings from backtranslated contexts is denoted as '+$\phi(q_i, c_i')$' from Eqn. (2).

	In Domain			Out of Domain		
	Calib. Accu	AUROC	Cov@Acc=80%	Calib. Accu	AUROC	Cov@Acc=80%
	SQuAD1.1 + HotpotQA			SQuAD1.1 + HotpotQA / Other MRQA datasets		
MaxProb	58.2±0.2	58.0±0.3	38.4%	56.8±0.2	56.5±0.2	38.3%
Kamath et al. (2020)	62.6±0.5	62.3±0.7	40.9%	61.2±0.4	60.7±0.5	39.7%
Ours	65.8±0.3	66.8±0.4	43.1%	63.7±0.3	64.1±0.3	41.6%
+ features	67.4±0.5	68.5±0.4	43.3%	65.4±0.3	66.9±0.3	42.7%
+ features + $\phi(q_i, c_i')$	**69.2±0.4**	**70.3±0.4**	**44.3%**	**67.6±0.4**	**68.8±0.5**	**43.9%**
+ features + $\phi(q_i', c_i)$	66.8±0.3	67.9±0.3	42.4%	64.7±0.4	66.2±0.3	42.5%
	SQuAD1.1 + NQ			SQuAD1.1 + NQ / Other MRQA datasets		
MaxProb	64.8±0.3	71.5±0.3	49.2%	61.4±0.2	66.7±0.3	45.9%
Kamath et al. (2020)	68.5±0.4	75.5±0.6	53.4%	64.1±0.6	69.2±0.5	51.5%
Ours	69.5±0.3	76.3±0.5	57.8%	64.3±0.4	69.4±0.4	54.3%
+ features	70.3±0.4	77.0±0.3	59.1%	64.9±0.5	70.4±0.5	56.5%
+ features + $\phi(q_i, c_i')$	**73.2±0.4**	**79.4±0.3**	**60.7%**	**66.7±0.5**	**72.1±0.5**	**57.6%**
+ features + $\phi(q_i', c_i)$	72.5±0.4	78.7±0.3	59.3%	65.8±0.5	71.4±0.5	55.9%

Table 3: Calibration results on standard reading comprehension datasets. In the out of domain setting, we first list the training dataset of calibrator, then the test dataset.

4.2.2 Open Retrieval QA

We consider separate calibrators for two different approaches (Karpukhin et al., 2020; Raffel et al., 2020).

Extractive (Retrieve-And-Predict) We consider two baseline calibrators: one takes the product of normalized passage score (normalized across all passage candidates) and answer score (normalized across the top 10 answer spans for each passage), and another takes the product of unnormalized passage and answer scores.

Then, we introduce calibrator augmented with our input example embedding. We include two example embeddings as features: one is the question context embedding as used in the reading comprehension setting (from Eqn 1), and another is the average of the answer span start token representation and the answer span end token representation.

Generation based (Seq2Seq) For seq2seq models (Raffel et al., 2020), the output answer space includes all sentences that can be generated with conditional language model. Thus, instead of MaxProb, we use the likelihood of the generated answer (i.e., the product of the conditional probabilities for each token in the generated answer) as a baseline. Then, we introduce calibrator with our input example embedding (from Eqn 1).

5 Results

Calibration Table 3 reports calibration results on standard reading comprehension datasets. The top block displays the performance of calibrators trained on the SQuAD and HotpotQA datasets, and the bottom block shows the results of calibrators trained on the SQuAD and NQ datasets. In both settings, the our input example embedding works better than the manual feature set. However, two approaches are complementary in all settings. Interestingly, paraphrasing questions shows gains in Natural Questions but not in other datasets. We hypothesize that organically collected search queries contain more ambiguous and ill-defined queries than crowdsourced questions where questions were based directly on the context. Adding paraphrased context embeddings, on the other hand, shows a modest gain across all settings. Unlike QA models have access to millions of parameters, calibrators, even with our feature set, are provided with very limited information. We hypothesize that augmenting the feature set with paraphrased context enabled the calibrator to gain more information about the example, facilitating higher performance.

Table 4 shows the results in more challenging settings: one with adversarial attacks and another containing unanswerable questions. In both settings, we observe sizable gains (5-10% increase in calibration accuracy) for the in domain setting, but the gains are smaller in out of domain settings. Similar to the Natural Questions dataset, in SQuAD 2.0, which includes adversarially designed questions without an answer, paraphrasing the question is more helpful than paraphrasing the context. On the other hand, in the adversarial setting where contexts are manipulated, paraphrasing contexts is more effective. Overall, our new feature vector shows consistent gain across all datasets and settings.

We present the calibration in open retrieval QA in Table 5. Overall, calibrator accuracy is higher

	In Domain			Out of Domain		
	Calib. Accu	AUROC	Cov@Acc=80%	Calib. Accu	AUROC	Cov@Acc=80%
	SQuAD1.1 Adversarial			MRQA / SQuAD1.1 Adversarial		
Kamath et al. (2020)	52.4±0.2	53.7±0.4	25.4%	52.4±0.2	52.2±0.4	24.7%
Ours	61.1±0.4	63.2±0.3	35.6%	53.2±0.4	53.6±0.3	25.3%
+ features	61.4±0.6	63.5±0.3	35.8%	53.8±0.4	54.3±0.4	26.8%
+ features + $\phi(q_i, c'_i)$	**62.8±0.3**	**65.2±0.2**	**37.3%**	**54.9±0.5**	**55.1±0.3**	**27.5%**
+ features + $\phi(q'_i, c_i)$	61.6±0.3	63.7±0.4	35.5%	53.6±0.5	53.9±0.5	26.6%
	SQuAD2.0			MRQA / SQuAD2.0		
Kamath et al. (2020)	57.6±0.3	59.2±0.4	31.7%	54.8±0.4	56.5±0.5	29.6%
Ours	58.9±0.2	61.1±0.2	33.8%	55.7±0.3	57.4±0.4	30.7%
+ features	60.1±0.2	61.9±0.3	34.2%	56.6±0.4	58.3±0.5	31.6%
+ features + $\phi(q_i, c'_i)$	60.2±0.3	61.8±0.3	34.1%	56.4±0.5	57.9±0.4	31.2%
+ features + $\phi(q'_i, c_i)$	**62.6±0.4**	**64.3±0.3**	**35.9%**	**58.1±0.4**	**60.4±0.4**	**32.9%**

Table 4: Calibration results on adversarial and unanswerable SQuAD datasets. In the out of domain setting, we first list the training dataset of calibrator, then the test dataset.

Model	Answer Acc	Calibrator	Calib. Accu	Calib. AUROC	Cov@Acc=80%
Extractive (DPR)	41.0	Unnormalized scores	65.9±0.2	65.2±0.2	10.4%
(Karpukhin et al., 2020)		Normalized scores	72.2±0.4	74.5±0.3	28.9%
		Ours (+ Normalized Scores)	**77.3±0.3**	**78.7±0.2**	**30.5%**
Generation (T5)	25.5	Likelihood	89.3±0.1	86.6±0.1	10.4%
(Raffel et al., 2020)		Ours (+ Likelihood)	**91.6±0.3**	**92.9±0.1**	**11.3%**

Table 5: Calibration results on NQ open retrieval test set for different base QA models and calibration features.

compared to RC, partially because the answer accuracy is substantially lower. For example, with generation based model (T5)'s answer accuracy of 25.5, simply predicting incorrectly for every example will give 74.5 calibration accuracy. In both models, internal confidence scores (Likelihood and Normalized scores) provide reasonable calibrator performance, yet adding our feature set improves the performance. In particular, our calibrator shows a larger gain in the DPR setting. Encouraged by this result, we test our calibrator as an answer candidate reranker for top answer candidates from DPR. Despite high calibration accuracy of generation based approach, selective QA performance (Cov@Acc=80%) is higher with the extractive approach, suggesting comparing calibration performance across models of different accuracy is challenging.

Answer Reranking Table 6 shows the results of our calibrator as an answer candidate reranker. The calibrator considers the top 1,000 answer candidates (100 retrieved passages, each with top 10 answer spans) and outputs top candidates based on the calibrator score instead of the model score. We show negligible gains in top 1 accuracy but bigger gains in top 5 accuracy. These small but noticeable gains show potential for using calibrators to improve open retrieval QA performances, where multiple answer candidates are considered.

	Top 1 EM	Top 5 EM
DPR	41.0	57.8
Unnormalized scores	10.3±0.2	23.1±0.3
Normalized scores	41.2±0.1	58.6±0.1
Ours (+ Normalized scores)	**41.4±0.1**	**59.0±0.1**

Table 6: Results on open domain question answering in NQ. The calibrator is used as a reranker for selecting the top answer span out of 1,000 answer spans (10 answer spans per each of 100 retrieved passages).

6 Analysis

6.1 Task-Agnostic Representation vs. Representation from QA Model

Our study has shown that input example embedding is very useful, adding complementary power to model confidence features. Based on this result, we further ask the question, is it possible to build a calibrator without accessing the model parameters, but only a small amount of calibration training data (which consists of questions, context, and whether the model's prediction is correct or incorrect)? We train a calibrator that does not have any access to the QA model parameters and only takes the model's predictions on a small set of training data (a couple thousand of QA examples). This cali-

	In Domain	Out of Domain
	SQuAD1.1 + Hotpot QA	Other MRQA datasets
CLS	63.5±0.4	62.3±0.5
Ours	65.8±0.3	63.7±0.3
Diff.	2.3%	1.4%

Table 7: CLS token ablation results, all numbers refer to calibration accuracy. Using CLS token as a feature shows a strong calibration performance, lagging behind question context encoding from the RC model only by a few points. The gap is even smaller in out of domain setting.

brator uses a standard pretrained language model (BERT) to encode [CLS; (q_i, c_i)] and takes the final layer hidden representation of the [CLS] token as a feature. Table 7 shows the performance of the [CLS] token classifier. Surprisingly, this calibrator outperforms the MaxProb baseline (in Table 3) in all settings and outperforms Kamath et al. (2020) (in Table 3) in most settings, indicating information about the question and context might be more useful than the QA model's confidence. Using the input example embedding from the QA model shows only 1-3 point gains than using the CLS token embedding. This trend holds for across various settings (more results in Table 11 in Appendix).

6.2 Quality of Back Translation

Question paraphrasing (Dong et al., 2017) can improve performances of QA models. Similarly, both question and context paraphrasing improves calibration performance. In this section, we investigate the quality of backtranslation used in our study. We manually inspect 100 question paraphrasing from SQuAD 2.0 dataset. 71 examples maintain the original meaning, 12 examples change its meanings, and 17 examples are hard to distinguish. One common pattern for meaning change is when proper nouns in the original sentences are missing and incorrectly translated (e.g. John Calvin → Jean Calvin).

We study how much variability is introduced during paraphrasing by studying divergence between the original sentence and the paraphrased sentence. We calculate the sentence BLEU score with *NLTK* (Bird et al., 2009), using the original text as source and the back-translated text as target for both question paraphrasing and context paraphrasing. The average sentence BLEU score is larger than 0.55 for all datasets, indicating back-translation introduces relatively minor changes in phrasing.

q	In what country is Normandy located?
q'	What country is Normandy in?
q	When did Edward return?
q'	When did Edward come back?
q	How would one write T(n) = 7n2 + 15n + 40 in big O notation?
q'	How do you write T(n) = 7n2 + 15n + 40?
q	What kind of arches does Norman architecture have?
q'	What kind of arches does Norman's building have?

Table 8: Question back translation samples from SQuAD 2.0 dataset. The first row (q) refers to the original question, and the second row (q') refers to back-translated question. In the third example, back translation introduces an error.

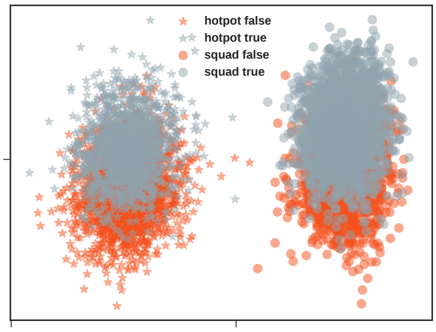

Figure 1: A visualization for the input example embedding from HotPotQA and SQuAD datasets. We denote the data domain by markers with different shapes and denote the correctness with different colors. The X-axis and Y-axis denote the first and second dimensions extracted by linear discriminant analysis, respectively.

Visualization Figure 1 shows a visualization of the question context embeddings from HotpotQA and SQuAD. We use linear discriminant analysis (Pedregosa et al., 2011) to plot input example embeddings and observe that embeddings from the same dataset are closer to each other. It demonstrates that embeddings are almost linearly separable between domains, but it is much harder to distinguish correct answers from incorrect ones.

Choice of Calibrator Architecture We test if our results are sensitive to the choice of classifiers: XGBoost, logistic regression (LR), and k-nearest

	In Domain	Out of Domain
	SQuAD1.1 + NQ	Other MRQA datasts
Xgboost	67.4±0.5	65.4±0.3
LR	66.6±0.3	64.7±0.3
KNN	66.3±0.2	64.6±0.3

Table 9: Ablation study on different classifiers with features (Ours + features). All numbers refer to calibration accuracy.

	Language	In Domain SQuAD1.1 + NQ	Out of Domain Other MRQA datasets
$\phi(q_i, c_i')$	FR	73.2±0.4	66.7±0.5
$\phi(q_i, c_i')$	DE	73.0±0.4	66.5±0.5
$\phi(q_i', c_i)$	FR	72.5±0.4	65.8±0.5
$\phi(q_i', c_i)$	DE	72.8±0.3	66.1±0.4

Table 10: Calibration accuracy for different pivot languages: French vs. German, using calibrator with features (Ours + features).

neighbors (KNN). Table 9 indicates our gains hold across different classifiers. Full experimental results can be found in Appendix.

Choice of Pivot Language We test whether the choice of pivot language in backtranslation impacts performances. We find little difference between pivoting through German or French (See Table 10).

7 Related Work

Calibration in NLP Calibration has become an important topic in NLP as well as general machine learning (Guo et al., 2018; Pleiss et al., 2017; FAN et al., 2021) as confidence scores from calibrators can be useful for the error correction process (Feng and Sears, 2004). Calibration has been studied in natural language inference, commonsense reasoning (Desai and Durrett, 2020; Varshney et al., 2020), dialogue systems (Mielke et al., 2020), semantic parsing (Dong et al., 2018), coreference resolution (Nguyen and O'Connor, 2015) and sequence labeling (Jagannatha and Yu, 2020).

In question answering, Kamath et al. (2020)'s study on selective question answering inspired our work. We measure the calibration performance with calibrator accuracy, AUROC, and coverage at accuracy. Expected Calibration Error (ECE) (Guo et al., 2017) is another commonly used metric for calibration performance, but we consider calibrator as a binary classifier at here. Jagannatha and Yu (2020) also studies calibration in reading comprehension, using language model perplexity and model's confidence as features. Language model perplexity coarsely and indirectly captures information about the question and context. We propose an improved feature space and thoroughly test it in challenging settings, e.g., adversarial RC, unanswerable RC, and open retrieval QA.

Calibration During Training Recent work in QA introduces an answer verification step (Tan et al., 2018; Hu et al., 2019; Wang et al., 2020) at the end of the pipeline. During the training, this verifier module takes the questions, answers, or MRC model's state as inputs and determines the answers' validity. Then, the validity score is used to update the model parameters during training. Thus, the validator is jointly trained with the MRC model. While this is conceptually similar to our set up, instead of tying the calibrator into the model, we design a universal post-hoc calibrator that can be easily applied to any model architecture.

Calibration with Ensembles Ensemble diversity has been used to improve uncertainty estimation and calibration (e.g. Raftery et al., 2005; Stickland and Murray, 2020). While it is effective, calibration with model ensembling is usually expensive and time consuming (Zhou et al., 2002, 2018). Our calibrator is an offline postprocessing step that does not require further training of the original model.

8 Conclusion

We introduce a richer feature space for question answering calibrators with question and context embeddings and paraphrase-augmented inputs. Our work suggests deciding the correctness of a QA system depends on both the semantics of the question-context and the confidence of the model. We thoroughly test our calibrator in domain shift, adversarial, and open domain QA settings. The experiments show noticeable gains in performance across all settings. We further demonstrate our calibrator's general applicability by using it as a reranker in extractive open domain QA. To summarize, our calibrator is simple, effective and general, with potential to be incorporated into existing models or extended for other NLP tasks.

Acknowledgments

We would like to thank UT Austin NLP group, especially Kaj Bostrom and Greg Durrett for feedback and suggestions.

References

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A BERT baseline for the natural questions. *arXiv:1901.08634*.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

Arthur Asuncion and David Newman. 2007. Uci machine learning repository.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

G. W. Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Annual Meetings of the Association for Computational Linguistics (ACL)*.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. *Empirical Methods in Natural Language Processing (EMNLP)*, abs/2003.07892.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, Copenhagen, Denmark. Association for Computational Linguistics.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Annual Meetings of the Association for Computational Linguistics (ACL)*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

XINJIE FAN, Shujian Zhang, Korawat Tanwisuth, Xiaoning Qian, and Mingyuan Zhou. 2021. Contextual dropout: An efficient sample-dependent dropout module. In *International Conference on Learning Representations*.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Jinjuan Feng and Andrew Sears. 2004. Using confidence scores to improve hands-free speech based navigation in continuous dictation systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 11(4):329–356.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *arXiv preprint arXiv:1910.09753*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, pages 1321–1330. PMLR.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2018. On calibration of modern neural networks. *International Conference on Machine Learning (ICML)*.

James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. 2019. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721.

Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. 2019. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6529–6537.

Abhyuday N. Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *Annual Meetings of the Association for Computational Linguistics (ACL)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. *Annual Meetings of the Association for Computational Linguistics (ACL).*

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Empirical Methods in Natural Language Processing (EMNLP).*

A. Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *ArXiv*, abs/1903.00802.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems (Neurips)*, pages 6402–6413.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *International Conference on Machine Learning (ICML)*, abs/1906.00300.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, F. Petroni, V. Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692.*

Sabrina J. Mielke, Arthur Szlam, Y.-Lan Boureau, and Emily Dinan. 2020. Linguistic calibration through metacognition: aligning dialogue agent responses with expected correctness. *ArXiv*, abs/2012.14983.

Khanh Nguyen and Brendan T. O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. In *Empirical Methods in Natural Language Processing (EMNLP).*

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On fairness and calibration. In *Advances in Neural Information Processing Systems (Neurips)*, pages 5680–5689.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, M. Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

A. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski. 2005. Using bayesian model averaging to calibrate forecast ensembles. *Monthly Weather Review*, 133:1155–1174.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *Annual Meetings of the Association for Computational Linguistics (ACL).*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Empirical Methods in Natural Language Processing (EMNLP).*

Rico Sennrich, B. Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, volume abs/1511.06709.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, volume abs/1611.01603.

Libin Shen, Anoop Sarkar, and F. Och. 2004. Discriminative reranking for machine translation. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).*

Jasdeep Singh, Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. Xlda: Cross-lingual data augmentation for natural language inference and question answering. *arXiv preprint arXiv:1905.11471.*

Lewis Smith and Yarin Gal. 2018. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533.*

Jiaming Song, Yang Song, and Stefano Ermon. 2019. Unsupervised out-of-distribution detection with batch normalization. *arXiv preprint arXiv:1910.09115.*

Asa Cooper Stickland and Iain Murray. 2020. Diverse ensembles improve calibration. *ArXiv*, abs/2007.04206.

Chuanqi Tan, Furu Wei, Qingyu Zhou, Nan Yang, Weifeng Lv, and Ming Zhou. 2018. I know there is no answer: modeling answer validation for machine reading comprehension. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 85–97. Springer.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. *ACL 2017*, page 191.

N. Varshney, Swaroop Mishra, and Chitta Baral. 2020. It's better to say "i can't answer" than answering incorrectly: Towards safety critical nlp systems. *ArXiv*, abs/2008.09371.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Xuguang Wang, Linjun Shou, Ming Gong, Nan Duan, and Daxin Jiang. 2020. No answer is better than wrong answer: A reflection model for document level machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4141–4150.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Adams Wei Yu, David Dohan, Minh-Thang Luong, R. Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. volume abs/1804.09541.

Tianyi Zhou, S. Wang, and J. Bilmes. 2018. Diverse ensemble evolution: Curriculum data-model marriage. In *NeurIPS*.

Z. Zhou, Jianxin Wu, and Wei Tang. 2002. Ensembling neural networks: Many could be better than all. *Artif. Intell.*, 137:239–263.

Appendix

A Additional Experimental Results

	In Domain			Out of Domain		
	Calib. Accu	AUROC	Cov@Acc=80%	Calib. Accu	AUROC	Cov@Acc=80%
	SQuAD1.1 + Hotpot			SQuAD1.1 + HotpotQA / Other MRQA datasets		
CLS	63.5±0.4	65.2±0.4	41.8%	62.3±0.5	62.6±0.3	40.3%
Ours	65.8±0.3	66.8±0.4	43.1%	63.7±0.3	64.1±0.3	41.6%
Difference	2.3	1.6	1.3%	1.4	1.5	1.3%
	SQuAD1.1 + NQ			SQuAD1.1 + NQ / Other MRQA datasets		
CLS	66.8±0.3	74.0±0.5	58.5%	62.8±0.4	67.8±0.4	57.6%
Ours	69.5±0.3	76.3±0.5	62.8%	64.3±0.4	69.4±0.4	59.3%
Difference	2.7	2.3	4.3%	1.5	1.6	1.7%

Table 11: CLS token ablation results on reading comprehension.

	In Domain			Out of Domain		
	Calib. Accu	AUROC	Cov@Acc=80%	Calib. Accu	AUROC	Cov@Acc=80%
	SQuAD1.1 + Hotpot			SQuAD1.1 + HotpotQA / Other MRQA datasets		
Xgboost	67.4±0.5	68.5±0.4	43.3%	65.4±0.3	66.9±0.3	42.7%
Logistic Regression	66.6±0.3	67.3±0.3	42.6%	64.7±0.3	66.1±0.3	42.3%
KNN	66.3±0.2	67.0±0.3	42.1%	64.6±0.3	65.8±0.2	41.8%
	SQuAD1.1 + NQ			SQuAD1.1 + NQ / Other MRQA datasets		
Xgboost	70.3±0.4	77.0±0.3	59.1%	64.9±0.5	70.4±0.5	56.5%
Logistic Regression	69.7±0.3	76.3±0.2	58.6%	64.2±0.4	69.7±0.4	56.1%
KNN	68.9±0.2	75.8±0.2	58.3%	63.8±0.3	69.3±0.3	55.6%

Table 12: Ablation study on different classifiers with features (Ours + features).

B Hyperparameters and Training Details

A binary classifier is trained using the gradient boosting library XGBoost (Chen and Guestrin, 2016). We finetune the following hyper-parameters, colsample by level, colsample by node, colsample by tree, learning rate, and the number of estimators on the development set. We use the following search space: colsample by level/mode/tree is set to the same value and selected from $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, the learning rate and number of estimators are selected from $\{0.01, 0.1, 0.2, 0.5\}$ and $\{5, 25, 50, 100\}$, respectively. These hyper-parameters are chosen based on the performance on the validation set.

For base QA models, we mostly following the hyperparameters used in the original work (e.g., batch size 32 & learning rate of 5×10^{-5} for BERT-base SQuAD 1.1 model). All calibrators are trained five times, each with different data partitions and random seeds. We report the variances in the results. Our calibrator does not share its weights with the base QA models.

Enhancing Metaphor Detection by Gloss-based Interpretations

Hai Wan[1], **Jinxia Lin**[1], **Jianfeng Du**[2,3*], **Dawei Shen**[1], **Manrong Zhang**[1]
[1] School of Computer Science and Engineering,
Sun Yat-sen University, Guangzhou 510006, P.R.China
[2] Guangzhou Key Laboratory of Multilingual Intelligent Processing,
Guangdong University of Foreign Studies, Guangzhou 510006, P.R.China
[3] Pazhou Lab, Guangzhou 510330, P.R.China
wanhai@mail.sysu.edu.cn, jfdu@gdufs.edu.cn,
{linjx36, shendw5, zhangmr7}@mail2.sysu.edu.cn

Abstract

This paper focuses on utilizing *metaphor interpretation* to enhance *metaphor detection*. Considering that existing approaches to metaphor interpretation are limited by ambiguous meanings of the metaphorical substitute words, this paper proposes a novel interpretation mechanism that utilizes *glosses* to interpret metaphorical words. Since there is no dataset annotated for both metaphor detection and metaphor interpretation, we enhance three datasets TroFi, VUA, and PSUCMC from the field of metaphor detection with gloss annotations. Accordingly, we develop a model for jointly conducting metaphor detection and gloss-based interpretation (named MDGI-Joint for short). Experimental results demonstrate that MDGI-Joint outperforms state-of-the-art models on all the three enhanced datasets and that gloss-based metaphor interpretation benefits metaphor detection.[1]

1 Introduction

Metaphor has been defined as words or other linguistic expressions representing another concept with the language from a more concrete conceptual domain (Kövecses and Zoltán, 2002; Lagerwerf and Meijers, 2008). According to existing studies on metaphor (Kövecses and Zoltán, 2002; Steen, 2010), metaphor has been used so frequently in daily language that it almost occurs in one out of three natural language sentences.

Metaphor detection aims to identify all *metaphorical words* in given texts and has been demonstrated to be of great value in many Natural Language Processing (NLP) tasks, such as machine translation (Mao et al., 2018), machine reading comprehension (Shutova et al., 2013), *etc*. Therefore, metaphor detection has drawn increasing interests in recent years. There have emerged a number of methods for metaphor detection, including SEQ (Gao et al., 2018), RNN_HG (Mao et al., 2019), RNN_MHCA (Mao et al., 2019), MUL_GCN (Le et al., 2020), DeepMet (Su et al., 2020), *etc.*

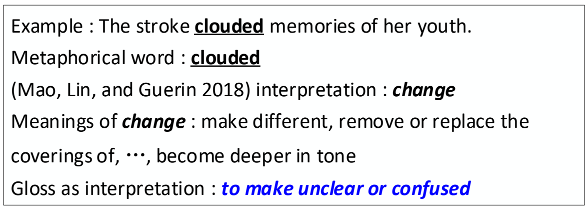
Example : The stroke **<u>clouded</u>** memories of her youth.
Metaphorical word : **<u>clouded</u>**
(Mao, Lin, and Guerin 2018) interpretation : ***change***
Meanings of ***change*** : make different, remove or replace the coverings of, ⋯, become deeper in tone
Gloss as interpretation : ***to make unclear or confused***

Figure 1: An example for metaphor detection, where the word **<u>clouded</u>** is a metaphorical word (highlighted in bold and underline). Interpretation by a substitute word (highlighted in **bold**) is computed by (Mao et al., 2018). Gloss as interpretation (highlighted in ***blue bold italics***) is picked from the Merriam Webster dictionary.

To capture the meanings of metaphorical words, *metaphor interpretation* has also been studied, which paraphrases metaphorical expressions into literal expressions that maintain the intended meanings of given texts (Mao et al., 2018). Existing approaches have treated metaphor interpretation as extraction of transferred properties, identification of the underlying conceptual mapping, or generation of a literal substitute paraphrase (Rai and Chakraverty, 2020). All of them are yet limited by ambiguous meanings of the metaphorical substitute words. Consider the example shown in Figure 1. The metaphorical word **<u>clouded</u>** is interpreted as **change** in (Mao et al., 2018), where **change** has 10 diverse meanings according to WordNet (Miller, 1995). Due to multiple meanings of **change**, it is difficult to precisely capture the intended meaning of the metaphorical word **<u>clouded</u>**.

It has been pointed out by Group (2007) that, metaphors have a clear distinction between their

---

[*] Corresponding author

[1] Source code and datasets are available at `https://github.com/sysulic/MDGI`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1971–1981
August 1–6, 2021. ©2021 Association for Computational Linguistics

basic meanings and contextual meanings. Consider the example shown in Figure 1 again. The gloss-based interpretation for **clouded** gives the meaning "*to make unclear or confused*", which is evidently different from the basic meaning "*to grow cloudy*" of **clouded**. In other words, different from the substitute word, the *gloss* extracted from dictionaries can provide an unambiguous interpretation of **clouded** which exhibits a clear distinction to its basic meaning. Therefore, we can utilize gloss-based metaphor interpretation to enhance the metaphor detection task.

Based on the above observations, we propose to use glosses to interpret metaphorical words. Specifically, we formulate metaphor interpretation as a task of predicting the best gloss among a set of candidate glosses. Considering that *Word Sense Disambiguation* (WSD) (Kilgarriff, 2004) is a popular technique for identifying the correct meaning of a target word, modern WSD methods can be adapted to metaphor interpretation.

By now there is a lack of dataset annotated for both metaphor detection and metaphor interpretation. In order to study whether gloss-based interpretation benefits metaphor detection, we enhance three benchmark datasets in the field of metaphor detection, including two English datasets (TroFi and VUA) and one Chinese dataset (PSUCMC). For each dataset, we construct a set of candidate words and annotate these words with glosses extracted from a dictionary.

Accordingly we develop a joint model for **M**etaphor **D**etection and **G**loss-based **I**nterpretation (named MDGI-Joint for short). To be specific, our joint model encodes contexts and glosses independently for every given word. Based on both the contextual word embedding and gloss embeddings, a probability distribution for all glosses of the given word is computed and then used to predict the best gloss, in a similar way as the state-of-the-art WSD method (Blevins and Zettlemoyer, 2020), Afterwards, an attention mechanism is employed to compute an integrated representation of all glosses. This integrated representation is then concatenated with the contextual word embedding to determine whether the given word is metaphorical through a classical prediction layer. The joint model is trained by minimizing a combined loss from both the metaphor detection task and the metaphor interpretation task.

We conduct experiments on the aforementioned three enhanced datasets. Experimental results demonstrate that gloss-based metaphor interpretation does benefit metaphor detection. On one hand, the proposed model achieves state-of-the-art performance on all three enhanced datasets in the metaphor detection task. On the other hand, it also achieves comparable performance with the outstanding WSD method in the metaphor interpretation task.

The main contributions of this paper can be summarized as follows.

- We provide a novel interpretation mechanism that utilizes glosses to interpret metaphorical words.
- We enhance three metaphor detection datasets (TroFi, VUA, and PSUCMC) with annotations of glosses for metaphorical words.
- We develop a joint model for metaphor detection and gloss-based interpretation and empirically show that metaphor interpretation with glosses benefits metaphor detection.

2 Related Work

2.1 Metaphor Detection

Early studies on metaphor detection (Shutova et al., 2010; Rei et al., 2017) usually follow the linguistics theory (Lakoff and Johnson, 1980) and construct mappings from the source domain to the target domain.

Subsequent studies focus on metaphor detection over subject-verb-objects, adjective-noun tuples, or metaphorical phrases. Among these studies, a lot of semantic features including the degree of abstractness, the degree of concreteness, the degree of imageability, semantic super-senses (namely coarse semantic categories originating in WordNet), lemma unigrams, and grammatical dependencies are added to improve performance of metaphor detection (Turney et al., 2011; Tsvetkov et al., 2014; Klebanov et al., 2016; Özbal et al., 2016; Jang et al., 2015). Jang et al. (2015) took topic distribution into consideration. Jang et al. (2016) explored topic transition between a metaphor and its context. For handing multi-modal information, Shutova et al. (2016) considered both word embeddings and visual embeddings whereas Bulat et al. (2017) introduced a cross-modal method to integrate linguistic representations and property-based representations. A broader context of discourse was considered by (Jang et al., 2015) and (Mu et al., 2019).

Regarding word-level metaphor detection, metaphor detection can be treated as a sequence tagging task. Wu et al. (2018) proposed a neural model, which uses Word2Vec (Mikolov et al., 2013) as text representation and encodes part-of-speech (POS) tags, word clusters with Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) network. Gao et al. (2018) and Mao et al. (2019) respectively utilized GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings. Le et al. (2020) proposed to construct a graph CNN guided by dependency trees of sentences for metaphor detection and to construct a multi-task learning framework for the WSD task, utilizing the knowledge from WSD to improve metaphor detection. Instead of treating metaphor detection as a sequence tagging task, Su et al. (2020) proposed a novel reading comprehension paradigm based on a pre-trained language model, using features from POS tags and local texts.

2.2 Metaphor Interpretation

Metaphor interpretation is an intricate task, having a challenge in deciphering the meaning conveyed by a metaphorical expression (Rai and Chakraverty, 2020). Existing approaches to metaphor interpretation can be grouped into three categories. For the first category, the problem of metaphor interpretation is treated as a problem of extraction of transferred properties. It is often assumed that a metaphor is essentially a projection of a specific set of salient concept properties from the source domain, known as *property matching* (Su et al., 2016; Ortony, 1980). Su et al. (2016) extracted perceptual properties from the source domain and the target domain, and then searched for metaphor interpretation by expanding the extracted properties with synonymy relationships from WordNet. In contrast to property matching, the second category defines the problem of metaphor interpretation as a problem of identifying the underlying conceptual mapping, usually focusing on re-conceptualization of the target domain (Marmolejo-Ramos et al., 2013; Semino, 2010). Martin (2006) empirically found that metaphor interpretation has certain contextual clues (*e.g.*, the appearance of a concept from the target domain) and related metaphorical expressions. For the last category, the problem of metaphor interpretation is treated as the generation of a literal substitute paraphrase (Mao et al., 2018; Shutova et al., 2012). Mao et al. (2018) used hypernyms and synonyms as candidate substitutes and computed the best substitute word by the cosine similarity between the embedding of the given word and the embedding of a candidate substitute.

All the above methods for metaphor interpretation fail to capture the contextual meaning of a metaphorical word due to ambiguous interpretation of the metaphor. To tackle this issue, in this work we provide a novel interpretation mechanism that utilizes glosses to interpret metaphorical words.

2.3 Word Sense Disambiguation

Word sense disambiguation (WSD) aims to predict a specific meaning of a word that occurs in a particular context (Navigli, 2009). Understanding the meaning of a word in context is critical to many NLP tasks, such as machine translation (Vickrey et al., 2005; Neale et al., 2016; Gonzales et al., 2017) and information extraction (Ciaramita and Altun, 2006; Bovi et al., 2015). One category of WSD is class-based, which provides coarse-grained labels that are shared among different words. The other category of WSD is word-based, aiming to disambiguate every word in texts (Palmer et al., 2001; Moro and Navigli, 2015; Blevins and Zettlemoyer, 2020). Our proposed metaphor interpretation scheme belongs to this category. Some neural models for word-based WSD exploit encoders for better feature extraction. Based on an encoder, they either train classifiers on top of extracted features (Kågebäck and Salomonsson, 2016) or introduce a shared output space to label words (Raganato et al., 2017). Other neural models augment word representations with additional data by semi-supervised learning (Melamud et al., 2016; Yuan et al., 2016). BEM (Blevins and Zettlemoyer, 2020), which inspires our model, is a state-of-the-art method for word-based WSD. It introduces a bi-encoder model to embed the target word with its surrounding context and its glosses.

3 The Proposed MDGI-Joint Model

Given a sentence s consisting of n words $\{w_0, w_1, \dots, w_i, \dots, w_{n-1}\}$ and a list of all m_i candidate glosses $G_i = \{g_i^0, g_i^1, \cdots, g_i^j, \dots, g_i^{m_i-1}\}$ for the target word w_i, the task of metaphor interpretation aims to select a gloss from G_i to interpret the intended meaning of w_i, whereas the task of

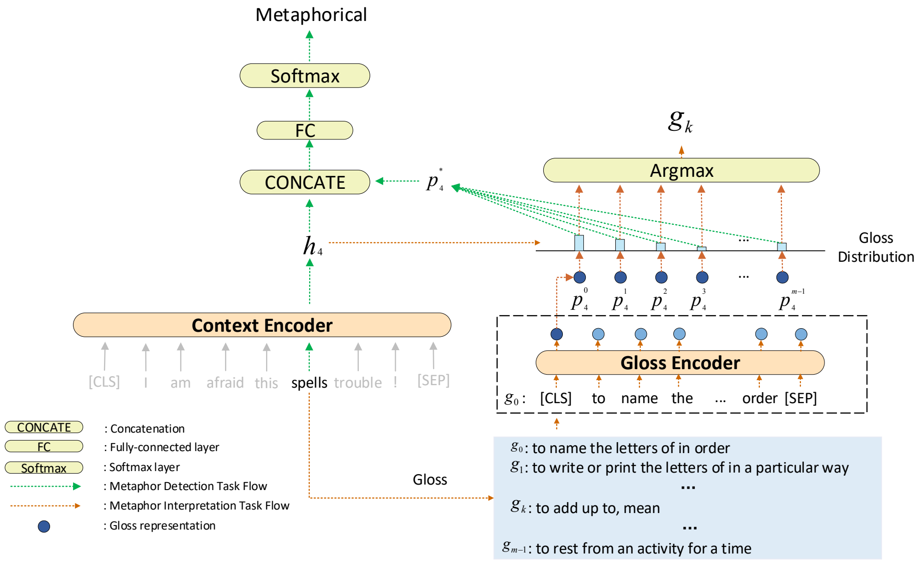

Figure 2: The architecture of our proposed joint model for **M**etaphor **D**etection and **G**loss-based **I**nterpretation.

metaphor detection focuses on predicting whether the word w_i is metaphorical or literal. Since gloss-based metaphor interpretation provides a rich representation for the contextual meaning of the target word, it is appropriate to construct a joint model to incorporate gloss-based metaphor interpretation and metaphor detection. Therefore, we propose a model named MDGI-Joint to incorporate metaphor detection and metaphor interpretation.

The architecture of MDGI-Joint is given by Figure 2. MDGI-Joint employs two encoders to generate respectively the contextual representation and the gloss representation for a target word. The probability distribution over all candidate glosses is computed by an attention mechanism. The gloss with the highest probability is predicted as the interpretation of the target word. Afterwards, an integrated representation of all candidate glosses is computed and then concatenated with the contextual representation. The concatenated representation is finally used to determine whether the target word is metaphorical, through a fully-connected layer followed by the softmax classifier.

3.1 Encoding Module for Sentences

Given a sentence s consisting of n words $\{w_0, w_1, \ldots, w_i, \ldots, w_{n-1}\}$ as well as a target word w_i, the contextual representation of w_i will be encoded into a vector. Considering that the pre-trained language model BERT (Devlin et al., 2019) has been proved to be effective in transfer learning, contributing to state-of-the-art performance in many NLP tasks, we fine-tune a BERT model to be the context encoder. Initially, we construct a token sequence "[CLS], $w_0, w_1, \cdots, w_{n-1}$,[SEP]", where [CLS] and [SEP] are special tokens introduced in BERT. Then this token sequence is taken as input to BERT, yielding a sequence of 768-dimensional vectors "$h_{[CLS]}, h_0, h_1, \ldots, h_{n-1}, h_{[SEP]}$" as the output of BERT. The corresponding vector h_i for the target word w_i is treated as the contextual representation of w_i.

3.2 Encoding Module for Glosses

For the target word w_i, we collect the set of candidate glosses $G_i = \{g_i^0, g_i^1, ..., g_i^{m_i-1}\}$ from an existing dictionary. We fine-tune another BERT model to be the gloss encoder. Similar to the encoding module for sentences, we also construct a token sequence for each gloss and feed it into BERT. For each gloss $g_i^j \in G_i$ where $0 \leq j \leq m_i - 1$, we use the output 768-dimensional vector corresponding to the first token "[CLS]" as the gloss representation, which is written as p_i^j.

3.3 Prediction for Metaphor Interpretation

Based on the contextual representation h_i of w_i and the gloss representation p_i^j of each gloss $g_i^j \in G_i$ where $0 \leq j \leq m_i - 1$, the probability α_i^j for a gloss g_i^j to represent the intended meaning of the word w_i is computed by an attention mechanism, formally defined as follow:

$$\alpha_i^j = \frac{\exp(h_i^T p_i^j)}{\sum_{k=0}^{m_i-1} \exp(h_i^T p_i^k)} \qquad (1)$$

3.4 Prediction for Metaphor Detection

We define the joint representation q_i of the target word w_i as the concatenation of the contextual representation h_i and the weighted sum p_i^* of gloss representations $p_i^0, \ldots, p_i^{m_i-1}$, which is formally defined below:

$$\begin{aligned} p_i^* &= \sum_{j=0}^{m_i-1} \alpha_i^j p_i^j, & (2) \\ q_i &= [h_i; p_i^*] & (3) \end{aligned}$$

where $[;]$ denotes the concatenation of two vectors.

By transforming the joint representation q_i through a fully connected layer followed by the softmax classifier, the probability distribution l_i that the target word w_i is metaphorical or literal is formally defined below:

$$l_i = \mathsf{softmax}(W_1 q_i + b_1) \tag{4}$$

where $W_1 \in \mathbb{R}^{2\times1536}$ and $b_1 \in \mathbb{R}^2$ are learnable parameters. The probability distribution l_i is of the form $[l_i^0, l_i^1]$, where l_i^0 is the predicted probability that w_i is literal, and l_i^1 is the predicted probability that w_i is metaphorical.

3.5 Training Objective Function

Based on predicted probabilities $\alpha_i^0, \ldots, \alpha_i^{m_i-1}$ over glosses, the loss value for metaphor interpretation about the target word w_i is defined as:

$$\text{loss}_i^{\text{MI}} = -\sum_{j=0}^{m_i-1} I(f_i = g_i^j)\log(\alpha_i^j) \tag{5}$$

where f_i is the correct gloss for w_i in sentence s, $I(X) = 1$ if X is true and $I(X) = 0$ otherwise.

As for the task of metaphor detection, we employ the binary cross entropy loss for predicting the target word w_i to be literal or metaphorical, defined as follow:

$$\text{loss}_i^{\text{MD}} = -(1-y_i)\log(l_i^0) - y_i \log(l_i^1) \tag{6}$$

where y_i is the correct label, $y_i = 0$ if the label is literal or $y_i = 1$ if the label is metaphorical.

The proposed joint model is trained by minimizing the following combined loss for every training sentence.

$$\text{loss} = \sum_{i=0}^{n-1} \text{loss}_i^{\text{MD}} + I(w_i \in \mathcal{C}) * \text{loss}_i^{\text{MI}} \tag{7}$$

where n is the number of words in the considering sentence, and $\mathcal{C}$ is a predefined set of candidate words required to be interpreted, which is collected from all metaphorical words in our experiments.

Table 1: Kappa-score for annotations in every dataset.

	TroFi	VUA	PSUCMC
kappa-score	0.82	0.86	0.89

Table 2: Statistics of the enhanced datasets. #sentences is the number of sentences; #tokens is the total number of words needed to be detected, %M is the metaphor percentage over the detected words, and #glosses is the number of samples with gloss annotations.

Dataset	#sentences	#tokens	%M	#glosses
TroFi_train	2,989	2,989	58.3	2,989
TroFi_val	374	374	52.4	374
TroFi_test	374	374	54.8	374
VUA_train	6,323	116,622	11.2	2,710
VUA_val	1,550	38,628	11.6	635
VUA_test	2,694	50,175	12.4	905
VERB_test	2,694	5,873	30.0	905
PSUCMC_train	1,381	28,572	8.3	5,486
PSUCMC_val	173	3,520	8.0	674
PSUCMC_test	173	3,727	7.4	730
VERB_test	165	736	16.3	120

4 Experiments

4.1 Three Enhanced Datasets

We enhance three datasets TroFi, VUA and PSUCMC from the field of metaphor detection, where TroFi and VUA are in English and PSUCMC is in Chinese.

- TroFi (Birke and Sarkar, 2006). This is a benchmark metaphor dataset with verb metaphors annotated. Following the work (Mao et al., 2019), we treat unlabeled words as literal in the training phase.

- VUA (Steen, 2010). The VU Amsterdam Metaphor Corpus (VUA) samples fragments from the British National Corpus. All words in the corpus are labeled. We evaluate on both the VUA ALL POS track and the VUA-Verb track.

- PSUCMC (Lu and Wang, 2017; Nacey et al., 2019). The PSU Chinese Metaphor Corpus

Table 3: Data splits of PSUCMC for different POS tags, including NOWN, VERT, ADJ and ADV.

Dataset	NOUN	VERB	ADJ	ADV
all	929	2,547	390	19
train	760	1,988	334	13
val	105	244	31	4
test	64	315	25	2

Table 4: Accuracy on the metaphor interpretation task. All numbers are in %.

Model	VUA	PSUCMC	TroFi
BEM	45.5	**82.3**	71.9
MDGI-Joint-S	**46.4**	**82.3**	**72.2**
MDGI-Joint	44.3	80.8	70.9

consists of text samples from the Lancaster Corpus of Mandarin Chinese where are annotated for metaphor-related words following MIPVU (Steen, 2010). All words in PSUCMC are labeled. Following the experimental setting of VUA, we evaluate on both the PSUCMC ALL POS track and the PSUCMC-Verb track.

All these datasets are enhanced to support the gloss-based metaphor interpretation task. We refer to the set of words that need to be interpreted as the *candidate set*. For TroFi, the candidate set includes all labeled verbs. For VUA, the words in the candidate set are randomly selected from verbs. As for PSUCMC, we randomly choose a set of words and filter the meaningless words to construct a candidate set. Words in the candidate set are annotated by human annotators.

For English datasets, when given a word in the candidate set, the annotators are asked to look up the Merriam-Webster dictionary[2] to fetch its glosses. For the Chinese dataset, word glosses are extracted from the Baidu Dictionary[3].

For every dataset, four annotators are recruited to annotate the dataset independently. All annotators need to select the most appropriate gloss that expresses the contextual meaning of the target word, by comparing all glosses with the given context of the target word. They also need to discuss and determine the final labels after generating their own annotations.

To verify the reliability of the annotations, we use kappa-score to measure inter-annotator agreements for the annotations. Kappa-score (Siegel, 1956) has been widely used in computational linguistics to measure the reliability of an annotation scheme. Table 1 shows the kappa-score for annotations in every dataset, which demonstrates that the annotations have a high degree of reliability.

For PSUCMC and TroFi, we randomly split the samples into a training set, a validation set, and a test set according to the proportion of $8:1:1$. For VUA, the data splits provided by (Mao et al., 2019) are reused. Table 2 reports the statistics of all experimental datasets, whereas Table 3 gives more details about PSUCMC.

4.2 Experimental Setup

In our experiments, BERT[4] is used as both the context encoder and the gloss encoder, where the uncased BERT base model is used for TroFi and VUA, and the Chinese BERT base model is used for PSUCMC.

There are two variants for our proposed model. The first variant, named MDGI-Joint-S, shares parameters between the context encoder and the gloss encoder. The second one, named MDGI-Joint, implements two independent encoders for context and gloss, respectively.

To train these two models, we set the learning rate as 2e-5. The maximum number of epochs is set to 20. We set the dropout probability to 0.2 for the fully connected layer. The max sequence length is set to 128 for TroFi and VUA, and 256 for PSUCMC. The batch sizes for the two tasks are all set to 8 for VUA and TroFi, and 16 for PSUCMC. We keep the best model that maximizes the F1 score on the validation set for the metaphor detection task. This model is then used to evaluate the test set.

4.3 Compared Methods

We compare our method with the following methods.

- SEQ (Gao et al., 2018). SEQ is a neural model taking ELMo embeddings and GloVe embeddings as input. It uses a Bi-LSTM encoder to capture the contextual information of the target word, and then employs a fully connected layer followed by the softmax classifier to predict whether a word is metaphorical or not.

- RNN_HG (Mao et al., 2019). RNN_HG also takes ELMo embeddings and GloVe embeddings as input. Different from SEQ (Gao et al., 2018), RNN_HG concatenates the encoded representation with the GloVe embedding to capture the contextual information of the target word.

---

[2] `https://www.merriam-webster.com/dictionary/`

[3] `https://dict.baidu.com`

[4] `https://github.com/google-research/bert`

Table 5: Accuracy on the metaphor detection task. All numbers are in %. '-' denotes no evaluation on the corresponding dataset.

	VUA ALL POS				VUA-Verb				PSUCMC ALL POS				PSUCMC-Verb				TroFi			
Method	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc
SEQ	71.6	73.6	72.6	93.1	68.2	71.3	69.7	81.4	74.9	57.5	65.0	95.4	71.6	40.2	51.5	87.5	91.2	85.9	88.4	87.7
RNN_HG	71.8	76.3	74.0	93.6	69.3	72.3	70.8	82.1	70.6	69.8	70.2	95.6	67.0	56.8	61.5	88.2	90.4	82.4	86.2	85.6
RNN_MHCA	73.0	75.7	74.3	93.8	66.3	75.2	70.5	81.8	73.5	68.7	71.1	95.9	63.1	58.3	60.6	87.5	86.6	84.9	85.7	84.5
MUL_GCN	74.8	75.5	75.1	93.8	72.5	70.9	71.7	83.2	-	-	-	-	-	-	-	-	-	-	-	-
DeepMet	73.8	73.2	73.5	90.5	76.2	78.3	**77.2**	86.2	66.7	73.9	70.1	96.4	61.4	67.3	64.2	93.1	92.2	81.0	86.2	85.8
MDGI-Joint-S	81.3	73.2	77.0	94.6	78.8	71.5	75.0	85.6	89.7	69.8	78.5	97.2	85.9	60.8	**71.2**	92.0	89.2	84.9	87.0	86.1
MDGI-Joint	82.5	72.5	**77.2**	94.7	78.9	70.9	74.7	85.4	89.0	70.6	**78.7**	97.2	85.9	60.8	**71.2**	92.0	89.3	89.3	**89.3**	88.2

- RNN_MHCA (Mao et al., 2019). RNN_MHCA has a similar architecture as RNN_HG. But it adopts a multi-head contextual attention mechanism to capture the contextual information of the target word.

- MUL_GCN (Le et al., 2020). MUL_GCN is a multi-task learning framework. It exploits the similarity between word sense disambiguation and metaphor detection, by employing a graph convolutional neural network (GCN) to connect the words of interest with context words for metaphor detection.

- DeepMet (Su et al., 2020). DeepMet is a RoBERTa (Ott et al., 2019) based model with an ensemble strategy. It also takes features including POS tags and local texts as input. For fairness, we only compare our models with the single model of DeepMet.

- BEM (Blevins and Zettlemoyer, 2020). BEM is a model for the WSD task. It consists of two independent encoders. One is the context encoder, which embeds the target word and its surrounding context. The other encoder is the gloss encoder, which embeds the gloss for each word sense. Both encoders are deep transformer networks initialized from BERT.

For the evaluation on PSUCMC and TroFi, we use the publicly released code for all the compared methods except for MUL_GCN whose code is unavailable. Thus we cannot obtain results for MUL_GCN on both PSUCMC and TroFi. For VUA, we present results that are reported in the published papers of the compared methods.

It should be noted that all compared methods target the English domain only in their published papers. Since PSUCMC is a dataset in the Chinese domain, to evaluate the compared methods on PSUCMC, we use ELMo embeddings trained on the Xinhua proportion of Chinese gigawords[5], which is a Chinese corpus released by Che et al. (2018) and Fares et al. (2017); moreover, we place the GloVe embeddings with word embeddings trained on the zh-wiki corpus[6] based on Word2Vec (Mikolov et al., 2013).

4.4 Experimental Results

We use P (precision), R (recall), F1 (F1 score) and Acc (accuracy) as the evaluation metrics for the metaphor detection task, and Acc (accuracy) for the metaphor interpretation task. All the reported numbers are in percent. For the metaphor detection task, all words in the test set are evaluated. For the metaphor interpretation task, although corresponding glosses can be computed for all words, only the words with annotations in the test set are evaluated.

4.4.1 Results for Metaphor Interpretation

It can be seen from Table 4 that, no matter whether the two encoders share parameters or not, the proposed model achieves similar results comparable with BEM. MDGI-Joint-S is even slightly superior to BEM. The implemented BEM model does not share parameters between the two encoders. Therefore, it can be confirmed that the improved performance of MDGI-Joint-S comes from capturing the interaction between the two tasks, rather than from inheriting parameters from BEM.

4.4.2 Results for Metaphor Detection

From Table 5, it is evident that MDGI-Joint outperforms other methods over all three datasets except for the VUA-Verb track. Although DeepMet achieves the best performance on the VUA-Verb track, it exhibits lower performance than our proposed model on other datasets and the VUA ALL POS track. There results indicate that joint training for metaphor detection and metaphor interpretation does improve the performance of metaphor

---

[5] https://github.com/HIT-SCIR/ELMoForManyLangs

[6] https://dumps.wikimedia.org/zhwiki/latest/

Table 6: Case studies to analyze the beneficial effect of gloss-based interpretations.

Text	Detected word	Gold label	Gold gloss	Prediction	Method
Iranian guns pummeled Basra on the war's first day, and in the following eight years, about 65, 000 shells rained down.	rained	metaphorical	to fall like rain	literal	DeepMet
				metaphorical	MDGI-Joint
Stand up man.	stand	literal	to support oneself on the feet in an erect position	metaphorical	DeepMet
				literal	MDGI-Joint

Table 7: Ablation study for the metaphor detection task. All numbers are given in %.

	VUA ALL POS				VUA-Verb				PSUCMC ALL POS				PSUCMC-Verb				TroFi			
Method	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc
MDGI-Joint-S	81.3	73.2	77.0	94.6	78.8	71.5	**75.0**	**85.6**	89.7	69.8	78.5	**97.2**	85.9	60.8	**71.2**	**92.0**	89.2	84.9	87.0	86.1
MDGI-Joint	82.5	72.5	**77.2**	**94.7**	78.9	70.9	74.7	85.4	89.0	70.6	**78.7**	**97.2**	85.9	60.8	**71.2**	**92.0**	89.3	89.3	**89.3**	**88.2**
MD-MGI	83.3	71.5	76.9	**94.7**	78.4	69.1	73.4	85.5	76.2	75.6	75.9	96.5	72.3	67.5	69.8	90.5	91.2	85.4	88.2	87.4
MD (only)	80.9	72.2	76.3	94.4	77.5	70.1	73.6	84.8	80.3	69.8	74.7	96.5	77.9	61.7	68.8	90.9	92.5	83.9	88.0	87.4

Table 8: Ablation study for the gloss-based metaphor interpretation task. The measure is accuracy in %.

Model	VUA	PSUCMC	TroFi
MDGI-Joint-S	**46.4**	**82.3**	**72.2**
MDGI-Joint	44.3	80.8	70.9
MD-MGI	44.8	80.8	70.3

detection. It can also be seen that all the compared methods perform poorly on the Chinese dataset PSUCMC. This is probably due to that these methods are strongly language sensitive. The reason why DeepMet outperforms our model on the VUA-Verb track is due to extra information such as subclasses of POS tags being taken as input in DeepMet. It is interesting to see whether our proposed model can be further improved by exploiting extra information. But this question is out of the scope of this work. It will be explored in our future work.

4.5 Case Study

As shown in Table 6, we use two cases to exemplify the superiority of our model in the metaphor detection task, demonstrating that gloss-based interpretation improves performance of metaphor detection.

The first example is picked from the TroFi dataset (see Row 2 in Table 6). In this example the detected word **rained** is a metaphorical word, but DeepMet predicts it as literal. The reason could be that the word **rained** commonly occurs in the given context. In contrast, MDGI-Joint correctly predicts it as metaphorical according to its gloss-based interpretation ***to fall like rain*** since a gloss of the form *do like some behavior* is commonly used as the gloss of a metaphorical word.

The second example is selected from the VUA dataset (see Row 3 in Table 6). In this example the detected word **stand** has a literal meaning in its context. DeepMet wrongly predicts it as metaphorical, while MDGI-Joint gives a correct prediction based on its predicted gloss-based interpretation which has an expression style as that of general glosses of literal words.

4.6 Ablation Study

To investigate the effect of joint training for the two tasks, we further conduct experiments on the following weakened models.

- MD-MGI. In this model we only use the context representation to predict whether a target word is metaphorical. In other words, the weighted sum of gloss representations is not considered in the metaphor detection task, although the two tasks are still jointly trained without sharing parameters between the context encoder and the gloss encoder.
- MD (only). This model addresses the metaphor detection task only and neglects the metaphor interpretation task. It detects metaphors based on the contextual representation of words only.

From Table 7 we observe that the performance of MD-MGI drops slightly on the metaphor detection task compared to the proposed model. The reason is probable that, in MD-MGI the two tasks only interact through the context encoder, leading to limited benefits for metaphor detection. From Table 8 we can see that all the three compared variants are comparable in the metaphor interpretation task, where MDGI-Joint-S is slightly better than others. These results show that whether the metaphor detection task uses the weighted sum of gloss representations or not has few impacts on the metaphor interpretation task. It can be seen

Table 9: Performance on the VUA ALL POS track for metaphor detection, separated by different POS tags. In-vocabulary words are annotated words whereas out-of-vocabulary words have no annotations. Only a portion of verbs in VUA are annotated.

POS	P	R	F1	Acc
ADJ	78.6	57.1	66.2	92
ADV	81.5	65.2	72.4	96.4
NOUN	78.3	60.8	68.8	91.6
VERB(in-vocabulary)	79.2	75.5	77.3	82.0
VERB(out-of-vocabulary)	78.1	70.2	73.9	91.6
Other POS	88.3	84.6	86.4	97.5

from Table 7 that the weakest variant namely MD (only) performs worse than all the other variants, indicating that joint training for metaphor detection and metaphor interpretation leads to performance improvement in metaphor detection.

4.7 Error Analysis on VUA

Taking the VUA dataset as example, there are more false negatives than false positives generated by MDGI-Joint; *i.e.*, the recall is lower than the precision, as shown in Table 9. Especially, adjectives, adverbs and nouns have a significantly lower recall than verbs. The reason for this phenomenon is two-fold. On one hand, only a partial set of verbs has annotations in the VUA dataset, so the detection of metaphorical adjectives, adverbs or nouns can only gain limited benefits from the metaphor interpretation task. On the other hand, in VUA a number of metaphors appear at the phrase level, but MDGI-Joint is only able to detect metaphors at the word level, thus it is more likely to predict false negatives. Consider the following example picked from VUA id:a7w-fragment01#41: *But only two million out of the 20 million journeys which ambulance crews carry out each year are emergency calls.* MDGI-Joint only detects the word *carry* as metaphorical. It treats words separately and cannot predict the preposition *out* in the phrase *carry out* as metaphorical. It also wrongly predicts *crews* as literal possibly due to that the word is a noun.

5 Discussion

Out-of-vocabulary words are treated differently in the training phase and in the test phase. In the training phase, the loss of metaphor interpretation over out-of-vocabulary words is not computed according to Equation (7). In the test phase, the evaluation on metaphor detection involves all words, but the evaluation on metaphor interpretation only targets in-vocabulary words. For the metaphor detection task, in-vocabulary words and out-of-vocabulary words have no different treatments. We have conducted extra experiments for metaphor detection on the VUA ALL POS track. The results reported in Table 9 show that MDGI-Joint achieves significantly higher performance on in-vocabulary verbs (77.3%) than on out-of-vocabulary verbs (73.9%), but on all out-of-vocabulary words, MDGI-Joint achieves similar performance (77.2%).

Due to the limited glosses extracted from existing dictionaries, there will be correct glosses for novel metaphors that do not appear in the training set. It has been shown (Rai and Chakraverty, 2020) that interpreting novel metaphors in general situations is difficult. Hence our proposed gloss-based interpretations are definitely more suitable for conventional or lexical metaphors. Nevertheless, by considering that we can expand glosses with more external resources, the effectiveness of our proposed approach is not limited to a fixed set of metaphors.

6 Conclusion and Future Work

Metaphor detection is of great value in many natural language processing tasks. In this paper we have utilized gloss-based metaphor interpretation to enhance metaphor detection. The novelty mainly lies in the interpretation mechanism, *i.e.*, utilizing glosses to interpret metaphorical words. Accordingly, we propose a joint model for metaphor detection and gloss-based interpretation. We enhance three datasets in the field of metaphor detection to evaluate the joint model. Experimental results confirm that metaphor interpretation in gloss improves the performance of metaphor detection. Future work will extend our approach to other rhetoric identification tasks.

Acknowledgments

This work is supported by the Guangdong Province Science and Technology Plan projects (2017B010110011), the National Natural Science Foundation of China (No. 61876204, 61976232, and 51978675), the National Key R&D Program of China (No.2018YFC0830600), Guangdong Province Natural Science Foundation (No. 2018A030313086), All-China Federation of Returned Over-seas Chinese Research Project (No. 17BZQK216).

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *EACL*, pages 329–336.

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *ACL*, pages 1006–1017.

Claudio Delli Bovi, Luis Espinosa Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In *EMNLP*, pages 726–736.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Modelling metaphor with attribute-based semantics. In *EACL*, pages 523–528.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *CoNLL*, pages 55–64.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*, pages 594–602.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *NoDaLiDa*, pages 271–276.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. In *EMNLP*, pages 607–613.

Annette Rios Gonzales, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *WMT*, pages 11–19.

Pragglejaz Group. 2007. Mip: A method for identifying metaphorically used words in discourse. *Metaphor and Symbol*.

Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Miller, Seungwhan Moon, and Carolyn Penstein Rosé. 2016. Metaphor detection with topic transition, emotion and cognition in context. In *ACL*, pages 216–225.

Hyeju Jang, Seungwhan Moon, Yohan Jo, and Carolyn Penstein Rosé. 2015. Metaphor detection in discourse. In *SIGDIAL*, pages 384–392.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568*.

Adam Kilgarriff. 2004. How dominant is the commonest sense of a word? In *TSD*, volume 3206, pages 103–112.

Beata Beigman Klebanov, Chee Wee Leong, E. Dario Gutiérrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *ACL*, pages 101–106.

Kövecses and Zoltán. 2002. *Metaphor: A Practical Introduction*. Oxford University Press.

Luuk Lagerwerf and Anoe Meijers. 2008. Openness in metaphorical and straightforward advertisements: Appreciation effects. *Journal of Advertising*, 37(2):19–30.

George Lakoff and Mark Johnson. 1980. *Metaphors we live by*. University of Chicago Press.

Duong Le, My Thai, and Thien Nguyen. 2020. Multitask learning for metaphor detection with graph convolutional neural networks and word sense disambiguation. In *AAAI*, pages 8139–8146.

Xiaofei Lu and Ben Pin-Yun Wang. 2017. Towards a metaphor-annotated corpus of mandarin chinese. *Lang. Resour. Evaluation*, 51(3):663–694.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and wordnet based metaphor identification and interpretation. In *ACL*, pages 1222–1231.

Rui Mao, Chenghua Lin, and Frank Guerin. 2019. End-to-end sequential metaphor identification inspired by linguistic theories. In *ACL*, pages 3888–3898.

Fernando Marmolejo-Ramos, María Rosa Elosúa, Yuki Yamada, Nicholas Francis Hamm, and Kimihiro Noguchi. 2013. Appraisal of space words and allocation of emotion words in bodily space. *PLoS One*, 8(12):e81688.

James H Martin. 2006. A corpus-based analysis of context effects on metaphor comprehension. *Trends in Linguistics Studies and Monographs*, 171:214.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*, pages 51–61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *SemEval@NAACL-HLT*, pages 288–297.

Jesse Mu, Helen Yannakoudakis, and Ekaterina Shutova. 2019. Learning outside the box: Discourse-level features improve metaphor identification. In *NAACL-HLT*, pages 596–601.

S. Nacey, A.G. Dorst, T. Krennmayr, and W.G. Reijnierse. 2019. *Metaphor Identification in Multiple Languages: MIPVU around the world*, volume 22. John Benjamins Publishing Company.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.

Steven Neale, Luís Gomes, Eneko Agirre, Oier Lopez de Lacalle, and António Branco. 2016. Word sense-aware machine translation: Including senses as contextual features for improved translation models. In *LREC*, pages 2777–2783.

Andrew Ortony. 1980. Understanding metaphors. *Center for the Study of Reading Technical Report; no. 154*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT, Demonstrations*, pages 48–53.

Gözde Özbal, Carlo Strapparava, Serra Sinem Tekiroglu, and Daniele Pighin. 2016. Learning to identify metaphors from a corpus of proverbs. In *EMNLP*, pages 2060–2065.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All-words and verb lexical sample. In *SENSEVAL@ACL*, pages 21–24.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *EMNLP*, pages 1156–1167.

Sunny Rai and Shampa Chakraverty. 2020. A survey on computational metaphor processing. *ACM Computing Surveys (CSUR)*, 53(2):1–37.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. In *EMNLP*, pages 1537–1546.

Elena Semino. 2010. Descriptions of pain, metaphor, and embodied simulation. *Metaphor and Symbol*, 25(4):205–226.

Ekaterina Shutova, Tim Van de Cruys, and Anna Korhonen. 2012. Unsupervised metaphor paraphrasing using a vector space model. In *COLING*, pages 1121–1130.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *NAACL-HLT*, pages 160–170.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *COLING*, pages 1002–1010.

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

Sidney Siegel. 1956. Nonparametric statistics for the behavioral sciences.

Gerard Steen. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

Chang Su, Jia Tian, and Yijiang Chen. 2016. Latent semantic similarity based interpretation of chinese metaphors. *Eng. Appl. Artif. Intell.*, 48:188–203.

Chuandong Su, Fumiyo Fukumoto, Xiaoxi Huang, Jiyi Li, Rongbo Wang, and Zhiqun Chen. 2020. Deepmet: A reading comprehension paradigm for token-level metaphor detection. In *Fig-Lang*, pages 30–39.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *ACL*, pages 248–258.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *EMNLP*, pages 680–690.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *HLT/EMNLP*, pages 771–778.

Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, and Yongfeng Huang. 2018. Neural metaphor detecting with cnn-lstm model. In *Fig-Lang*, pages 110–114.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. pages 1374–1385.

Evaluating Word Embeddings with Categorical Modularity

Sílvia Casacuberta[1*], **Karina Halevy**[1*], **Damián E. Blasi**[1,2,3]
1: Harvard University 2: MPI for Evolutionary Anthropology 3: HSE University
{scasacubertapuig, khalevy}@college.harvard.edu
dblasi@fas.harvard.edu

Abstract

We introduce categorical modularity, a novel low-resource intrinsic metric to evaluate word embedding quality. Categorical modularity is a graph modularity metric based on the k-nearest neighbor graph constructed with embedding vectors of words from a fixed set of semantic categories, in which the goal is to measure the proportion of words that have nearest neighbors within the same categories. We use a core set of 500 words belonging to 59 neurobiologically motivated semantic categories in 29 languages and analyze three word embedding models per language (FastText, MUSE, and subs2vec). We find moderate to strong positive correlations between categorical modularity and performance on the monolingual tasks of sentiment analysis and word similarity calculation and on the cross-lingual task of bilingual lexicon induction both to and from English. Overall, we suggest that categorical modularity provides non-trivial predictive information about downstream task performance, with breakdowns of correlations by model suggesting some meta-predictive properties about semantic information loss as well.

1 Introduction

Word embeddings represent words and phrases in continuous low-dimensional vector spaces. They are usually trained with neural language models or word collocations (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013b; Levy et al., 2015), such that similar assignments in a space reflect similar usage patterns. The rise of monolingual embeddings such as Word2Vec (Mikolov et al., 2013b), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017), coupled with the need to transfer lexical knowledge across languages, has also led to the development of cross-lingual word embeddings, in which different languages share a single distributed representation and are mapped into the same vector space. Such methods use different bilingual supervision signals (at the level of words, sentences, or documents) with varying levels of strength (Ruder et al., 2019).

A central task in the study of word embeddings is finding metrics to evaluate their quality. These metrics can either be *extrinsic*, where embeddings are used as input features for downstream NLP tasks and evaluated on their performance, or *intrinsic*, where embeddings are directly tested for how well they capture syntactic or semantic properties in their own right (Qiu et al., 2018). Extrinsic methods are not always feasible for low-resource languages due to a lack of annotated data. Moreover, downstream model components can be fine-tuned to achieve higher performance on certain tasks without necessarily indicating improvement in the semantic representation of words in an embedding space (Leszczynski et al., 2020).

This paper presents *categorical modularity*, a low-resource intrinsic evaluation metric for both monolingual and cross-lingual word embeddings based on the notion of graph modularity. The underlying principle is that *in good embeddings, words in the same semantic category should be closer to each other than to words in different categories*. We quantify this by building the k-nearest neighbor graph with a fixed set of words' semantic categories and computing the graph's modularity for a given embedding space. *Modularity* measures the strength of division of a graph with densely connected groups of vertices, with sparser connections between groups (Newman, 2006).

We source our semantic categories from Binder et al. (2016). In contrast to other semantic and ontological categories in the literature, these have been motivated by a set of experiential attributes with neurobiological consistency, covering sensory, motor, spatial, temporal, affective, social, and cog-

---

[*] Equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1982–1993
August 1–6, 2021.

nitive dimensions. We refer to these attributes collectively as *Binder categories*. The resulting dataset consists of 500 English words, each labeled with three categories at three levels of semantic granularity. For example, the word *chair* belongs to *Concrete Objects* (Level 1), *Artifacts* (Level 2), and *Furniture* (Level 3). 442 words are pulled from Binder, on top of which we add a few words to even out distributions of categories and replace a few English-specific words with words that are more easily translated to non-English languages.

We then translate these 500 English words into 28 more languages, selected based on their availability in the form of pre-trained vectors from the MUSE library (Conneau et al., 2018). We produce 300-dimensional embeddings for these words using three popular embedding models: the monolingual FastText (Bojanowski et al., 2017) and subs2vec (Paridon and Thompson, 2020) models and the cross-lingual MUSE (Conneau et al., 2018) model. Using these embeddings, we obtain the nearest-neighbor sets among the 500 words within each (language, model) pair and use those relationships to calculate a modularity score for the pair. We compare modularity scores to performance on three downstream tasks: sentiment analysis (monolingual classification), word similarity (monolingual regression), and word-level bilingual lexicon induction (BLI, cross-lingual regression) both to and from English. We obtain moderate to strong positive correlations on all three tasks, with slightly stronger results on the monolingual tasks. We also provide an analysis of correlations broken down by individual model and explore potential meta-predictive properties of categorical modularity.

We further show that estimating modularity on Binder categories yields relevant information that cannot simply be derived from naturally occurring distributions of word clusters in embedding spaces. We show this by replicating all three downstream task correlation analyses with modularity scores based on clusters obtained with unsupervised community detection methods (Clauset et al., 2004), which we henceforth refer to as *unsupervised clusters*. After establishing the utility of categorical modularity, we show some of its use cases for comparing and selecting models for specific NLP problems, and we discuss preliminary results about the individual categories we find to be most predictive of downstream task performance.

Our code and data are available to the public.[1]

2 Related Work

2.1 Word Embedding Evaluation Metrics

While word embeddings have become crucial tools in NLP, there is still little consensus on how to best evaluate them. Evaluation methods commonly fall into two categories: those motivated by an *extrinsic* downstream task and those motivated by the *intrinsic* study of the nature of semantics and the cognitive sciences (Bakarov, 2018). Intrinsic and extrinsic metrics do not always align, as some models have high quality as suggested by intrinsic scores but low extrinsic performance, and vice versa (Schnabel et al., 2015; Glavaš et al., 2019).

Some commonly used methods of extrinsic evaluation include named entity recognition (Collobert et al., 2011) —including the datasets *CoNLL-2002* and *CoNLL-2003* (Tjong Kim Sang and De Meulder, 2003)—, sentiment analysis (Schnabel et al., 2015), semantic role labeling, and part-of-speech tagging (Collobert et al., 2011). Intrinsic evaluation methods include word semantic similarity (Baroni et al., 2014), concept categorization (Baroni et al., 2014), and experiments on neural activation patterns (Søgaard, 2016).

Our categorical modularity metric is inspired by Fujinuma et al. (2019). They study the modularity of cross-lingual embeddings based on the premise that different languages are well-mixed in good cross-lingual embeddings and thus have low modularity with respect to language. Our metric improves upon the modularity proposed in Fujinuma et al. (2019) by overcoming the problem caused by low modularity potentially occurring with a purely random distribution of word vectors and being mistaken for high embedding quality, as it is unlikely for a random distribution to coincidentally have highly modular clusters corresponding to Binder categories. Moreover, our metric is able to evaluate both monolingual and cross-lingual word embeddings and allow for comparisons between these types of embeddings (e.g., FastText and MUSE), and it incorporates cognitive information through the use of brain-based semantic categories.

2.2 Cognitive Approaches to NLP

Recent work on word embeddings has explored the connections between NLP word representations

---

[1] `https://github.com/enscma2/categorical-modularity`

and cognitively grounded representations of words. Such connections enrich both computational and neuroscientific research: external cognitive signals can enhance the capacity of artificial neural networks to understand language, while language processing in neural networks can shed light on how the human brain stores, categorizes, and processes words (Muttenthaler et al., 2020).

Cognitive approaches to lexical semantics propose a model in which words are defined by how they are organized in the brain (Lakoff, 1988). Based on this premise, Hollenstein et al. (2019) propose *CogniVal*, a framework for word embedding evaluation with cognitive language processing data. They evaluate six different word embeddings against a combination of 15 cognitive data sources acquired via eye-tracking, electroencephalography (EEG), and functional magnetic resonance imaging (fMRI). In a similar line of work, both Søgaard (2016) and Beinborn et al. (2019) evaluate word embeddings using fMRI datasets.

The use of cognitive data in NLP goes well beyond the evaluation of word embeddings. Utsumi (2020) uses the neurosemantically inspired categories from Binder et al. (2016) to identify the knowledge encoded in word vectors. Among other conclusions, they find that the prediction accuracy of cognitive and social information is higher than that of perceptual and spatiotemporal information.

3 Modularity and k-NN Graphs

The concept of modularity has origins in the field of network science, as first introduced by Newman (Newman, 2006). The goal of the modularity measure is to quantify the strength of the division of a network into clusters. Usually, such networks are represented with graphs. Intuitively, the modularity of a graph measures the difference between the fraction of edges in the graph that connect two nodes of the same category and the expected corresponding fraction if the graph's edges were distributed at random. Thus, the higher the proportion of edges between nodes that belong to the same category, the higher the modularity.

In our case, we construct the pertinent graph with the k-nearest neighbors algorithm. Given a set S_w of N words and a set S_c of categories such that each of the N words belongs to exactly one of the categories in S_c, we map each of the N words into a d-dimensional word embedding vector space and obtain a d-dimensional vector for each word. For each pair (w_i, w_j), where $w_i, w_j \in S_w$ and $1 \leq i, j \leq |S_w|$, with corresponding d-dimensional vectors v_i and v_j, we compute their cosine similarity (the cosine of the angle between them), which we denote by $\texttt{similarity}(i, j)$.

We create a matrix M_D of dimensions $|S_w| \times |S_w|$, where entry $(M_D)_{i,j}$ is $\texttt{similarity}(i, j)$. For a given $k \in \mathbb{Z}_{>0}$, we build the $|S_w| \times |S_w|$ k-nearest neighbor matrix (denoted k-NNM) as follows: entry (i, j) of k-NNM is equal to 1 if and only if word j is one of the k nearest neighbors of word i (i.e., if $\texttt{similarity}(i, j)$ is among the k largest cosine similarities between i and all other words in S_w). We note that M_D and k-NNM are not necessarily symmetric, as word i being the k-th nearest neighbor of word j does not imply the reverse. Finally, we define the k-NN graph of S_w as the graph defined by k-NNM viewed as an adjacency matrix. We can now describe how to compute the modularity score following the schema in Fujinuma et al. (2019).

Let d_i denote the degree of node i, that is, $d_i = \sum_j (k\text{-NNM})_{i,j}$, and let g_i denote the category of word i. For each category $c \in S_c$, the expected number of edges within c is

$$a_c = \frac{1}{2m} \sum_i d_i \, I[g_i = c], \tag{1}$$

where m is the total number of edges in the k-NN graph and I is the indicator function that evaluates to 1 if the argument is true and to 0 otherwise.

The fraction of edges e_c that connect words of the same semantic category c is

$$e_c = \frac{1}{2m} \sum_{i,j} (k\text{-NNM})_{i,j} \, I[g_i = c] \, I[g_j = c]. \tag{2}$$

By weighting the $|S_c|$ different semantic categories together, we calculate the overall modularity Q as follows:

$$Q = \sum_{c=1}^{|S_c|} (e_c - a_c^2). \tag{3}$$

Finally, we normalize Q by setting

$$Q_{max} = 1 - \sum_{c=1}^{N} a_c^2, \quad Q_{norm} = \frac{Q}{Q_{max}}. \tag{4}$$

In our setting, Q_{norm} indicates the modularity score of one (language, model) pair overall, but

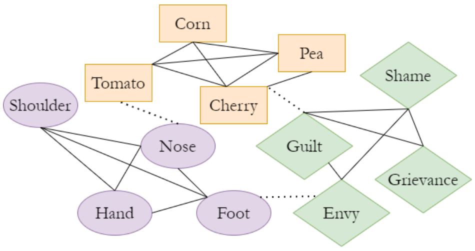

(a) A high-modularity semantic k-NN graph.

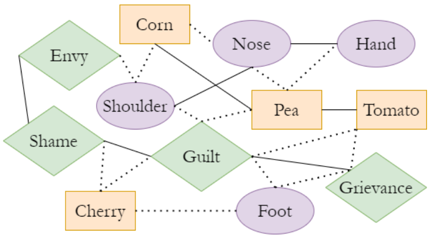

(b) A low-modularity semantic k-NN graph.

Figure 1: A toy visualization of the meaning of categorical modularity with some words pertaining to the Binder categories *Body Parts* (purple ovals), *Plants* (peach rectangles), and *Emotions* (green rhombi). Dotted edges connect nodes of different categories, while solid edges connect nodes of the same category.

we denote by Q_c the modularity of said (language, model) pair with respect to category $c \in S_c$. The definition of Q_c (normalized) is deduced from Equation 4:

$$Q_c = \frac{e_c - a_c^2}{Q_{max}}. \tag{5}$$

A higher value of Q_{norm} indicates that a higher number of words that belong to the same categories appear connected in the k-NN graph. In Sections 5 and 6, we analyze the values Q_{norm} for each of the languages, and in Section 8, we make some observations about the different values of Q_c.

In our conclusions about how our categorical modularity scores correlate with downstream task performance, we also want to prove that our selected neurosemantic-based categories are nontrivial and are better predictors than the unsupervised clusters that emerge from the embeddings. To find these clusters, we use the Clauset-Newman-Moore greedy modularity method (Clauset et al., 2004). This algorithm iteratively joins the pair of communities that most increases modularity until no such pair exists. For each value of k, we obtain the unsupervised communities in this manner and compute their modularity scores. In Section 6, we show that Binder categories are significantly better predictors than the unsupervised clusters using the same set of 500 words.

4 Dataset

In this section, we define the sets S_N and S_c of words and their semantic categories, respectively, that we use to compute categorical modularity scores for 29 languages.[2] As outlined in Section 1, our motivation to take a cognitive approach in the study of word embeddings prompts us to use words and categories that reflect a brain-based computational model of semantic representation as in Binder et al. (2016). We have 500 words (comprised of nouns, adjectives, and verbs) with 3 levels of categories, from most general (Level 1) to most specific (Level 3). Each word is tagged with 3 categories (one per level), which are listed in Table 1. After lifting 442 English words from Binder and adjusting the word set to optimize evenness of distribution across categories and translatability of concepts across languages, we manually translate the words to the 28 non-English languages[3] mentioned in footnote 2.

Level 1. Concrete Objects, Concrete Events, Abstract Entities, Concrete Actions, Abstract Actions, States, Abstract Properties, Physical Properties.
Level 2. Living Things, Other Natural Objects, Artifacts, Social Events, Nonverbal Sound Events, Weather Events, Miscellaneous, Concrete Events, Abstract Constructs, Cognitive Entities, Emotions, Social Constructs, Time Periods, Body Actions, Locative Change Actions, Social Actions, Miscellaneous Actions, Abstract Actions, States, Abstract Properties, Physical Properties.
Level 3. Animals, Body Parts, Humans, Human Groups, Plants, Natural Scenes, Miscellaneous Natural Objects, Furniture, Hand Tools, Manufactured Foods, Musical Instruments, Places/Buildings, Vehicles, Miscellaneous Artifacts, Social Events, Nonverbal Sound Events, Weather, Events, Miscellaneous Concrete Events, Abstract Constructs, Cognitive Entities, Emotions, Social Constructs, Time Periods, Body Actions, Locative Change Actions, Social Actions, Miscellaneous Actions, Abstract Actions, States, Abstract Properties, Physical Properties.

Table 1: Three levels of Binder categories.

---

[2] The 29 languages are: Arabic, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Greek, Hebrew, Hungarian, Indonesian, Italian, Macedonian, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Slovenian, Spanish, Swedish, Turkish, Ukrainian, and Vietnamese.

[3] For languages with which we were not familiar, we solicited translations from colleagues, whom we compensated fairly for this work.

5 General Categorical Modularity

With the dataset of 500 words that belong to three levels of semantic categories, we compute the modularity scores of each of the 29 languages for each of the three word embedding models (which we refer to as 87 (language, model) pairs): FastText,[4] MUSE,[5] and subs2vec.[6] We briefly summarize the properties of each of these embeddings.

FastText. Monolingual embeddings for 157 languages trained on Common Crawl and Wikipedia that use CBOW with position-weights and character n-grams (Bojanowski et al., 2017).

MUSE. Cross-lingual embeddings resulting from the alignment of 30 FastText embeddings into a common space under the supervision of ground-truth bilingual dictionaries (Conneau et al., 2018).

subs2vec. Monolingual embeddings for 55 languages trained on the OpenSubtitles corpus of speech transcriptions from television shows and movies using the FastText implementation of the skipgram algorithm (Paridon and Thompson, 2020). The authors claim that subtitles are closer to the human linguistic experience (Paridon and Thompson, 2020).

Information about the sizes of each (language, model) pair can be found in Appendix B. For each pair, we build the k-NN graph and compute modularity for different values of k and different levels of categories, which we treat as our 2 hyperparameters. We consider small values for k (namely $k \in \{2, 3, 4\}$) due to the fact that categories such as *States* have as few as 4 words.

6 Downstream Task Experiments

We test the reliability of categorical modularity by running a few downstream tasks and computing the Spearman rank correlations between categorical modularity scores and performance on these tasks.

After determining the optimal set of hyperparameters (k and level of semantic categories) for each task, we then compare the correlation produced by that set of hyperparameters with the correlation produced by the corresponding value of k with the modularity of the unsupervised clusters constructed by the community detection algorithm described in Section 3 to establish the non-triviality of the predictive properties of these chosen semantic categories. Table 2 provides a summary of correlation values for four tasks: movie review sentiment analysis (Sentiment), word similarity (WordSim), bilingual lexicon induction from English (BLI from), and bilingual lexicon induction to English (BLI to). Appendix A contains full tables with the correlation results. A visual summary of the results can be found in Figure 2.

Task	ρ	$\rho_{\mathbf{ft}}$	$\rho_{\mathbf{m}}$	$\rho_{\mathbf{s}}$
Sentiment	0.54	0.44	0.68	0.46
WordSim	0.71	0.59	0.34	0.80
BLI from	0.55	0.40	0.54	0.76
BLI to	0.50	0.29	0.56	0.82

Table 2: Summary of Spearman correlations of categorical modularity with downstream task performance for Binder categories, aggregated across all models (ρ) and broken down within FastText ($\rho_{\mathbf{ft}}$), MUSE ($\rho_{\mathbf{m}}$), and subs2vec ($\rho_{\mathbf{s}}$).

6.1 Sentiment Analysis

We first test our modularity scores through correlations with performance on the binary classification task of sentiment analysis, where the input is a movie review and the output is a binary label that corresponds to either positive or negative sentiment for that review. For this task, our data consists of 5,000 randomly selected positive movie reviews and 5,000 randomly selected negative reviews from the IMDB Movie Reviews dataset (Maas et al., 2011). We randomly partition these 10,000 reviews into 80% training and 20% testing data. Because this dataset is only available in English, we use the Google Translate API[7] to translate the data to 15 more languages (Arabic, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch, Estonian, Finnish, French, Greek, Hebrew, Hungarian, Indonesian, and Italian) for a total of $n = 48$ observations. The languages and the dataset size of 10,000 are chosen due to Google Translate API rate limits.

For each (language, model) pair, we convert the raw text of each review to a 300-dimensional embedding vector. We use the built-in black-box position-weighted continuous bag-of-words embedding model for FastText and subs2vec (Bojanowski et al., 2017), and we use a simple mean of individual word embeddings for MUSE, as the MUSE

---

[4] `https://fasttext.cc/docs/en/crawl-vectors.html`

[5] `https://github.com/facebookresearch/MUSE#download`

[6] `https://github.com/jvparidon/subs2vec`

[7] `https://pypi.org/project/google-trans-new/`

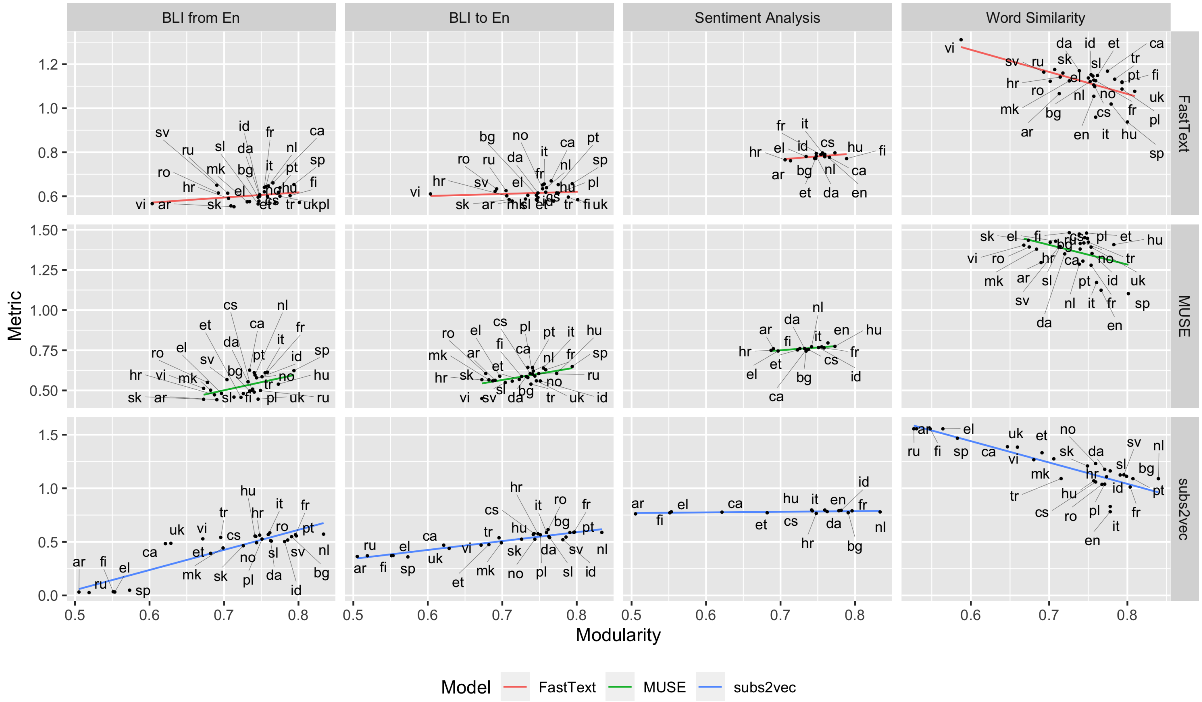

Figure 2: Summary of modularity vs performance metrics across tasks. Each language is represented with its 2-letter ISO 639-1 code. Hebrew, an outlier on the low end, is not included in this plot. Full modularity and performance data is included in our public GitHub repository.

library does not have multi-word phrase embeddings built into its functionality. Using a vanilla linear support vector machine model with scikit learn's default settings,[8] we run the task on each language-model pair 30 times and record the mean accuracy and precision scores for each pair. We then calculate the Spearman correlations of each of the 9 modularity scores with both the accuracy and precision values. Furthermore, we analyze the overall merged correlations (taking all 48 data points for a given modularity score and performance metric) as well as the correlations within models (taking only the 16 data points within each single model), giving us a total of 72 Spearman correlation values.

We find that the optimal set of hyperparameters is Level 3 categories with $k = 2$, which gives a Spearman correlation of $\rho = 0.54$ with the accuracy metric. Breaking it down by individual model, we have $\rho_{\mathbf{ft}} = 0.44$ for FastText, $\rho_{\mathbf{m}} = 0.68$ for MUSE, and $\rho_{\mathbf{s}} = 0.46$ for subs2vec. For $k = 2$, the correlations of unsupervised clusters with accuracy are $\rho = 0.09$ for all 48 observations merged, $\rho_{\mathbf{ft}} = 0.1$, $\rho_{\mathbf{m}} = 0.4$, and $\rho_{\mathbf{s}} = 0.35$, providing evidence that Binder categories contain non-trivial predictive information that is not present in naturally emerging clusters.

6.2 Word Similarity

Our next downstream task is the monolingual regression task of word similarity, in which the input is two words in one language and the output is a real number between 0 and 4 representing how similar the two words are (a higher score represents a greater degree of similarity). We use the English, Italian, and Spanish word pair datasets from SemEval-2017 (Camacho et al., 2017), and we use the same Google Translate API from Section 6.1 to translate the English dataset into the remaining 26 languages. Each language's dataset then has 500 word pairs, which we randomly split into 400 training pairs and 100 testing pairs for each trial.

Given a language and a model, we take each word pair, compute the 300-dimensional embeddings of both words, and calculate the Euclidean distance, Manhattan distance, and cosine similarity between the embeddings. We then feed these three scalars as a vector of inputs into a standard linear regression model from Python's scikit-learn package with default settings,[9] whose output is the

---

[8] `https://scikit-learn.org/stable/modules/svm.html`

[9] `https://scikit-learn.org/stable/modules/linear_model.html`

similarity score given in the dataset. To evaluate task performance, we compute the Mean Squared Error (MSE) loss for each run and record the mean MSE loss over 30 trials per (language, model) pair.

We then calculate the Spearman correlations of each of the 9 modularity scores with the negatives of the losses (such that positive correlation means that high modularity predicts high performance), both merged (87 data points) and within individual models (29 data points per model), for a total of 36 correlation values.

We find that all of the merged correlations are moderately to strongly positive. In particular, with the optimal hyperparameters of Level 2, $k = 2$, we have $\rho = 0.71$ overall, $\rho_{\mathbf{ft}} = 0.59$ for FastText, $\rho_{\mathbf{m}} = 0.34$ for MUSE, and $\rho_{\mathbf{s}} = 0.8$ for subs2vec. In comparison, the correlations of the unsupervised cluster modularities with mean MSE loss for $k = 2$ are $\rho = 0.27$, $\rho_{\mathbf{ft}} = 0.36$, $\rho_{\mathbf{m}} = 0.3$, and $\rho_{\mathbf{s}} = 0.42$, all weaker than their Binder counterparts.

6.3 Word-Level Bilingual Lexicon Induction

In addition to both monolingual classification and monolingual regression tasks, we also test our modularity metric on the cross-lingual regression task of bilingual lexicon induction. Using the ground-truth bilingual dictionaries provided by the publishers of MUSE (Conneau et al., 2018), we run this task with the 28 non-English languages listed in footnote 2 in two directions: translation to and from English. We use the 5,000-1,500 train-test split provided in the MUSE dictionary dataset and formulate the tasks as multivariate, multi-output regression tasks: for each observation in each (language, model) pair, we convert the English source word to its 300-dimensional embedding specified by the English version of the model and feed this vector as input to the same scikit-learn linear regression model as in Section 6.2, of which the output is a 300-dimensional vector in the target language model space representing the embedding of the target word.

We follow this procedure in the other direction as well by converting source non-English words to embeddings in the appropriate non-English model spaces, feeding those embeddings into the linear regression model, and computing 300-dimensional predictions for the target English word vectors in the English model spaces. To measure task performance in the "from English" direction, we convert the ground-truth non-English target words into vectors in the corresponding non-English embedding model space, compute the cosine similarities between each ground-truth vector and its corresponding predicted vector, and record the mean of those cosine similarities as a measure of how close we are to the ground truth on average. We run 30 trials per (language, model) pair and record the mean of the mean cosine similarities.

In the "to English" direction, we similarly convert the ground-truth English target words into vectors and compute the mean cosine similarity over the prediction-ground-truth pairs. We calculate the Spearman correlations of each of the 9 modularity scores with the 30-trial means of mean cosine similarities in both directions. Once again, we calculate the correlations both across all models and within each individual model, yielding 72 total correlation values.

The optimal set of hyperparameters for the merged correlation in the "from English" direction is Level 3, $k = 2$, giving a moderate $\rho = 0.55$ overall, $\rho_{\mathbf{ft}} = 0.4$ for FastText, $\rho_{\mathbf{m}} = 0.54$ for MUSE, and a strong $\rho_{\mathbf{s}} = 0.76$ for subs2vec. For comparison, the corresponding $k = 2$ correlations for unsupervised cluster modularities are $\rho = 0.35$, $\rho_{\mathbf{ft}} = 0.04$, $\rho_{\mathbf{m}} = 0.27$, and $\rho_{\mathbf{s}} = 0.65$ — all weaker than their Binder counterparts.

The optimal set of hyperparameters for the merged correlation in the "to English" direction is also Level 3, $k = 2$, giving a moderate $\rho = 0.5$ overall, a weak $\rho_{\mathbf{ft}} = 0.29$, a moderate $\rho_{\mathbf{m}} = 0.56$, and a very strong $\rho_{\mathbf{s}} = 0.82$. The corresponding unsupervised cluster correlations for $k = 2$ are $\rho = 0.35$, $\rho_{\mathbf{ft}} = 0.04$, $\rho_{\mathbf{m}} = 0.27$, and $\rho_{\mathbf{s}} = 0.65$.

Notably, in both the word similarity and BLI tasks, $\rho_{\mathbf{s}}$ is significantly stronger than $\rho_{\mathbf{ft}}$ and $\rho_{\mathbf{m}}$. This may be due to the fact that compared to sources such as Wikipedia and Common Crawl, the subtitles used as training data for subs2vec are more representative of how the human brain semantically maps language, as suggested by the model's creators (Paridon and Thompson, 2020).

Overall, these downstream task experiments suggest that categorical modularity is a non-trivially significant predictor of performance on both monolingual and cross-lingual NLP tasks (though it is stronger on monolingual tasks) and that it may have potential to be a meta-predictor of how well a particular model matches the information encoded in the human brain.

7 Use Cases: Comparing and Selecting Models

After having established substantial evidence of the predictive properties of categorical modularity, we present some examples of how the research community can make use of the metric for model evaluation and selection.

7.1 Comparing Models within a Language

The best hyperparameters for the tasks described in Section 6 are Level 3 with $k = 2$ along with Level 2 with $k = 2$. Across the 29 languages at the latter, FastText has the highest modularity 9 times (Arabic, Catalan, Estonian, Finnish, Greek, Macedonian, Polish, Turkish, Ukrainian), while MUSE has the highest modularity 3 times (Hungarian, Russian, Spanish), and subs2vec has the highest modularity 17 times. For Level 3 with $k = 2$, FastText has the highest modularity 13 times, while MUSE has the highest modularity 2 times (Russian, Vietnamese), and subs2vec has the highest modularity 15 times. Though individual choices should be made with each language, this suggests that subs2vec may be a strong choice for monolingual tasks overall.

7.2 Comparing Languages within a Model

We also present some evidence that categorical modularity predicts bilingual lexicon induction performance moderately well, and predictive properties are especially strong within subs2vec. For the optimal set of hyperparameters found in that task within subs2vec (Level 2, $k = 2$, $\rho_{\mathbf{s}} = 0.77$ from English and $\rho_{\mathbf{s}} = 0.81$ to English), the languages with the highest modularities in subs2vec are Dutch (0.84), Portuguese (0.81), French (0.80), Bulgarian (0.80), Swedish (0.80), Indonesian (0.79), and English (0.78), while the languages with the lowest modularities are Catalan (0.65), Spanish (0.58), Hebrew (0.58), Greek (0.56), Finnish (0.55), Arabic (0.53), and Russian (0.53). This may suggest which languages have lower amounts of resources at this time and hence deserve more data collection efforts on the part of the NLP community, particularly within subs2vec's domain of subtitle and conversational data.

7.3 Categorical Modularity as a Potential Meta-Predictor

We find evidence that categorical modularity reveals some information about how well models map to the human brain, as suggested by subs2vec's significantly stronger correlations. This is particularly true with regression tasks. Given a new or existing embedding model, calculating its categorical modularities and assessing their correlations with regression tasks such as word similarity may reveal if the model space is representative of how linguistic information is encoded in the brain.

8 Discussion and Future Work

Categorical modularity shows promise as an intrinsic word embedding evaluation metric based on our preliminary experiments. We can envision extending this work in several directions. For one, we can calculate single-category modularities (denoted by Q_c as defined in Equation 5) and test which individual categories contain the most predictive properties. Our limited experiments in this direction with the movie sentiment analysis task suggest that concrete and non-living categories have better predictive capabilities than abstract and living ones: for the sentiment analysis task, the 5 most strongly correlated categories are *Nonverbal Sounds*, *Artifacts*, *Concrete Objects*, *Vehicles*, and *Manufactured Foods*, while the 5 least correlated categories are *Abstract Properties*, *Abstract Constructs*, *Miscellaneous Actions*, *Humans*, and *Abstract Actions*.

We may also extend our work to more models and languages to see if the predictive properties truly hold across all languages and models. Additionally, as more multilingual research and data becomes available in this space, we may probe different sets of semantic categories, further downstream tasks (particularly multi-class classification, monolingual text generation, and sentence-level bilingual lexicon induction), and further variations of models used in downstream tasks (e.g., deeper neural networks instead of vanilla SVMs and linear regressions). We can also envision improvements upon the categorical modularity metric itself, perhaps by way of a lower-resource metric or a metric that works well on contextualized word embeddings for which the word-vector mappings may have more complex geometries. Our code and data, which are available to the public,[10] can also enable researchers and practitioners to replicate our results and experiment with different models, words, languages, and categories.

---

[10] `https://github.com/enscma2/categorical-modularity`

9 Conclusion

In this paper, we introduce categorical modularity, a novel low-resource metric that may serve as a tool to evaluate word embeddings intrinsically. We present evidence that categorical modularity has strong non-trivial predictive properties with respect to overall monolingual task performance, moderate predictive properties with respect to cross-lingual task performance, and potential meta-predictive properties of model space similarity to cognitive encodings of language.

Acknowledgments

This work was supported by the Radcliffe Institute for Advanced Study at Harvard University. SC and KH were supported by the School of Engineering and Applied Sciences at Harvard University under Prof. David Parkes. DEB acknowledges funding from the Branco Weiss Fellowship, administered by the ETH Zurich. DEB's research was also executed within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'.

Impact Statement

Ethical Concerns

All of the data used in this paper is either our own or from publicly released and licensed sources. Our data is mainly aimed towards researchers and developers who wish to assess the qualities of word embedding models and gain some intuition for embedding model selection for downstream tasks. In particular, our conclusions would be suited for researchers working among the 29 functioning languages given in the MUSE library, which are heavily skewed towards Indo-European languages. Though we do not directly introduce novel NLP applications, we provide resources that may be useful in selecting technologies to deploy and informing the development of word embedding systems.

Categorical modularity is intended to be an informational tool that sheds light on semantic representation of natural language information in computational word embeddings, and there are many aspects of its capabilities that can be improved upon, extended, or further explored. We would also like to emphasize that we have only tested our metric on three specific downstream tasks with basic downstream models, and these may not be representative of all NLP tasks in general. Categorical modularity also has not yet been shown to reveal information on representational harms inherent in word embedding spaces, so evidence of good downstream task performance should not be misconstrued as indicative of strong and beneficial performance across all NLP domains.

Environmental Impact

We acknowledge the pressing threat of climate change and therefore record some statistics on the computational costs of our experiments. All of our experiments are run with a 13-inch 2019 MacBook Pro with a 1.7 GHz Quad-Core Intel Core i7 processor running Python 3.8.3 in Terminal Version 2.11 on MacOS Big Sur Version 11.1. For the English language, generating FastText embeddings for our 500 core words took 20.31 seconds, generating the 500×500 k-NNM took 1 hour and 25.72 seconds, generating MUSE embeddings for the 500 words took 23.13 seconds, and generating the 500×500 k-NNM took 7 minutes and 39.32 seconds. For the downstream task of movie review sentiment analysis, it took 42.03 seconds to generate FastText sentence embeddings for 10,000 English reviews and 6 minutes and 38.32 seconds to generate these embeddings with MUSE. It took 0.35 seconds per review to translate from English to Spanish using the Google Translate API, and it took 2.6 seconds to run 30 trials of the sentiment analysis task for English FastText using scikit-learn's LinearSVC. For the task of word similarity calculation, English FastText embeddings and 3-dimensional input data took 21.25 seconds to generate for 500 word pairs, English MUSE-based embedding data took 33.47 seconds to generate, and the word similarity task using scikit-learn's LinearRegression took 0.09 seconds on the generated English FastText-based inputs. For bilingual lexicon induction, FastText English-Spanish embedding data took 55.87 seconds to generate, MUSE English-Spanish embedding data took 27 minutes and 29.56 seconds to generate, and the BLI task took a combined 4.35 seconds for both directions of English-Spanish using FastText. All other tasks took less than one second per language/model pair.

References

Ehsaneddin Asgari and Mohammad R.K. Mofrad. 2016. Comparing Fifty Natural Languages and Twelve Genetic Languages Using Word Embedding Language Divergence (WELD) as a Quantitative Measure of Language Distance. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*, pages 65–74.

Amir Bakarov. 2018. A survey of word embeddings evaluation methods. arXiv preprint *arXiv: 1801.09536*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247.

Lisa Beinborn, Samira Abnar, and Rochelle Choenni. 2019. Robust evaluation of language-brain encoding experiments. arXiv preprint *arXiv:1904.02547*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Jeffrey R. Binder, Lisa L. Conant, Colin J. Humphries, Leonardo Fernandino, Stephen B. Simons, Mario Aguilar, and Rutvik H. Desai. 2016. Toward a brain-based componential semantic representation. *Cognitive Neuropsychology*, 33(3-4):130–74.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

José Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of SemEval*, pages 15–26.

Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E*, 70:066111.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of ICLR*.

Yoshinari Fujinuma, Jordan Boyd-Graber, and Michael J. Paul. 2019. A resource-free evaluation metric for cross-lingual word embeddings based on graph modularity. In *Proceedings of ACL*, pages 4952–4962.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulic. 2019. How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions. In *Proceedings of ACL*, pages 710–721.

Nora Hollenstein, Antonio de la Torre, Nicolas Langer, and Ce Zhang. 2019. CogniVal: A Framework for Cognitive Word Embedding Evaluation. *Proceedings of CoNLL*, pages 538–549.

George Lakoff. 1988. Cognitive Semantics. In *Meaning and Mental Representations. UC Berkeley.*

Megan Leszczynski, Avner May, Jian Zhang, Sen Wu, Christopher R. Aberger, and Christopher Ré. 2020. Understanding the downstream instability of word embeddings. In *Proceedings of MLSys*, pages 262–290.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. In *Transactions of the Association for Computational Linguistics*, 3:211–225.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of ACL*, pages 142–150.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Lukas Muttenthaler, Nora Hollenstein, and Maria Barrett. 2020. Human brain activity for machine attention. arXiv preprint *arXiv:2006.05113*.

Mark E. J. Newman. 2006. Modularity and community structure in networks. In *Proceedings of the National Academy of Sciences*, 103(23):8577–858.

Jeroen van Paridon and Bill Thompson. 2020. subs2vec: Word embeddings from subtitles in 55 languages. *Behavior Research Methods*, pages 1–27.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Yuanyuan Qiu, Hongzheng Li, Shen Li, Yingdi Jiang, Renfen Hu, and Lijiao Yang. 2018. Revisiting Correlations between Intrinsic and Extrinsic Evaluations of Word Embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 209–221.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A Survey of Cross-lingual Word Embedding Models. *Journal of Artificial Intelligence Research* 65:569–631.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP*, pages 298–307.

Anders Søgaard. 2016. Evaluating word embeddings with fMRI and eye-tracking. In *Proceedings of the ACL Workshop on Evaluating Vector-Space Representations for NLP*, pages 116–121.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL-2003*, pages 142–147.

Akira Utsumi. 2020. Exploring What Is Encoded in Distributional Word Vectors: A Neurobiologically Motivated Analysis. *Cognitive Science*, 44(6).

A Appendix: Full Results for Correlations of Categorical Modularities with Downstream Tasks

This section contains full tables of correlations of general categorical modularities and unsupervised cluster modularities with downstream tasks. As above, ρ is overall correlation, ρ_{ft} is the correlation within FastText, ρ_m is the correlation within MUSE, and ρ_s is the correlation within subs2vec. On notation: in the "Model" columns, a, b represents the hyperparameters of Level a Binder categories with $k = b$ neighbors, while "C, a" represents "control" unsupervised clusters with the hyperparameter of $k = a$ neighbors.

A.1 Sentiment Analysis

Model	ρ	ρ_{ft}	ρ_{m}	ρ_{s}
1, 2	0.49	0.25	0.45	0.44
1, 3	0.49	0.26	0.47	0.54
1, 4	0.45	0.21	0.33	0.53
2, 2	0.52	0.32	0.53	0.45
2, 3	0.51	0.32	0.58	0.59
2, 4	0.47	0.36	0.52	0.54
3, 2	0.54	0.44	0.68	0.46
3, 3	0.51	0.34	0.72	0.58
3, 4	0.44	0.33	0.70	0.50
C, 2	0.09	0.10	0.40	0.35
C, 3	−0.21	0.13	−0.04	0.15
C, 4	−0.25	0.05	−0.24	0.15

Table 3: Spearman correlations of categorical modularity with accuracy of IMDB sentiment analysis task.

Model	ρ	ρ_{ft}	ρ_{m}	ρ_{s}
1, 2	0.45	0.27	0.44	0.27
1, 3	0.45	0.29	0.45	0.36
1, 4	0.42	0.25	0.31	0.35
2, 2	0.47	0.32	0.46	0.29
2, 3	0.48	0.32	0.53	0.44
2, 4	0.45	0.38	0.49	0.38
3, 2	0.49	0.42	0.60	0.29
3, 3	0.46	0.32	0.66	0.42
3, 4	0.41	0.34	0.64	0.35
C, 2	0.06	0.04	0.37	0.18
C, 3	−0.18	0.14	−0.08	0.02
C, 4	−0.24	0.09	−0.31	0.03

Table 4: Spearman correlations of categorical modularity with precision of IMDB sentiment analysis task.

A.2 Word Similarity

Model	ρ	ρ_{ft}	ρ_{m}	ρ_{s}
1, 2	−0.66	−0.47	−0.36	−0.74
1, 3	−0.60	−0.46	−0.24	−0.71
1, 4	−0.57	−0.46	−0.15	−0.71
2, 2	−0.71	−0.59	−0.34	−0.80
2, 3	−0.65	−0.62	−0.29	−0.80
2, 4	−0.61	−0.60	−0.23	−0.78
3, 2	−0.69	−0.65	−0.44	−0.79
3, 3	−0.62	−0.60	−0.42	−0.81
3, 4	−0.57	−0.61	−0.32	−0.79
C, 2	−0.27	−0.36	−0.30	−0.42
C, 3	0.00	−0.29	0.20	−0.45
C, 4	0.04	−0.36	0.32	−0.36

Table 5: Correlations with word similarity mean MSE.

A.3 Bilingual Lexicon Induction

Model	ρ	ρ_{ft}	ρ_{m}	ρ_{s}
1, 2	0.41	0.15	0.33	0.68
1, 3	0.33	0.07	0.24	0.64
1, 4	0.29	0.05	0.16	0.64
2, 2	0.50	0.34	0.42	0.77
2, 3	0.41	0.24	0.37	0.74
2, 4	0.38	0.22	0.30	0.73
3, 2	0.55	0.39	0.53	0.76
3, 3	0.49	0.35	0.47	0.75
3, 4	0.45	0.31	0.37	0.74
C, 2	0.35	0.19	0.44	0.55
C, 3	0.09	−0.09	−0.17	0.46
C, 4	0.07	−0.06	−0.09	0.37

Table 6: Correlations with mean cosine similarity on BLI from English.

Model	ρ	ρ_{ft}	ρ_{m}	ρ_{s}
1, 2	0.32	0.05	0.35	0.74
1, 3	0.28	−0.05	0.33	0.71
1, 4	0.26	−0.09	0.32	0.70
2, 2	0.42	0.22	0.45	0.81
2, 3	0.37	0.10	0.46	0.79
2, 4	0.38	0.07	0.47	0.78
3, 2	0.50	0.29	0.56	0.81
3, 3	0.47	0.20	0.52	0.80
3, 4	0.46	0.16	0.51	0.78
C, 2	0.35	0.04	0.27	0.65
C, 3	0.29	−0.21	0.15	0.57
C, 4	0.25	−0.13	0.19	0.48

Table 7: Correlations with mean cosine similarity on BLI to English.

B Sizes of Embedding Models

For contextual reference, we summarize the sizes of each of the embedding models used in this paper.

Language	FastText	MUSE	subs2vec
Arabic	610,976	132,480	898,080
Bulgarian	334,077	200,000	753,982
Catalan	490,564	200,000	27,220
Croatian	451,636	200,000	1,000,000
Czech	627,840	200,000	1,000,000
Danish	312,955	200,000	262,951
Dutch	871,021	200,000	495,055
English	1,000,000	200,000	1,000,000
Estonian	329,986	200,000	357,632
Finnish	730,482	200,000	842,787
French	1,000,000	200,000	514,066
Greek	306,448	200,000	859,548
Hebrew	488,935	200,000	679,649
Hungarian	793,865	200,000	1,000,000
Indonesian	300,685	200,000	221,876
Italian	871,052	200,000	597,058
Macedonian	176,946	176,947	132,238
Norwegian	515,787	200,000	179,069
Polish	1,000,000	200,000	1,000,000
Portuguese	592,107	200,000	505,535
Romanian	354,323	200,000	964,079
Russian	1,000,000	200,000	802,112
Slovak	316,097	200,000	330,354
Slovenian	281,822	200,000	517,625
Spanish	985,666	200,000	883,541
Swedish	1,000,000	200,000	325,033
Turkish	416,050	200,000	1,000,000
Ukrainian	912,457	200,000	80,123
Vietnamese	292,167	200,000	80,216

Table 8: Number of vectors in each (language, model) pair.

Attention-based Contextual Language Model Adaptation for Speech Recognition

Richard Diehl Martinez, Scott Novotney, Ivan Bulyko,
Ariya Rastrow, Andreas Stolcke, Ankur Gandhe

Amazon Alexa, Seattle, WA, USA
{mrtimri,snovotne,ibbulyko,arastrow,stolcke,aggandhe}@amazon.com

Abstract

Language modeling (LM) for automatic speech recognition (ASR) does not usually incorporate utterance level contextual information. For some domains like voice assistants, however, additional context, such as time at which an utterance was spoken, provides a rich input signal. We introduce an attention mechanism for training neural speech recognition language models on both text and non-linguistic contextual data [1]. When applied to a large de-identified dataset of utterances collected by a popular voice assistant platform, our method reduces perplexity by 7.0% relative over a standard LM that does not incorporate contextual information. When evaluated on utterances extracted from the long tail of the dataset, our method improves perplexity by 9.0% relative over a standard LM and by over 2.8% relative when compared to a state-of-the-art model for contextual LM.

1 Introduction

Conventional automatic speech recognition (ASR) systems include a language model (LM) and an acoustic model (AM). The LM component is trained separately, typically on large amounts of transcribed utterances that have been collected by an existing speech recognition system.

Voice assistants have become ubiquitous and crucially rely on ASR systems to convert user inputs to text. They often collect utterances spoken by users, along with associated de-identified contextual information. We hypothesize that these additional data, such as the time at which an utterance was spoken, provide a useful input signal for a LM. As an example, knowing that an utterance was spoken on December 25th, a LM should learn that the word "christmas" rather than "easter" is more likely to follow the phrase "lookup cookie recipes for".

To-date, some voice assistants have leveraged coarse geographic information for improving location search queries (Bocchieri and Caseiro, 2010; Lloyd and Kristjansson, 2012). These past efforts, however, have largely focused on improving a particular skill of an ASR system, and not the system's speech recognition in general.

In this paper, we focus on adapting recurrent neural network language models (RNN-LMs) to use both text and non-linguistic contextual data for speech recognition in general. While, outside of ASR, transformer-based (Vaswani et al., 2017) language models have largely replaced RNN-LMs, RNNs remain dominant in ASR architectures such as connectionist temporal classification (Graves et al., 2006), and RNN-T (Graves, 2012; He et al., 2019).

The most common method for incorporating non-linguistic information into a RNN-LM is to learn a representation of the context that is concatenated with word embeddings as input to the model. This concatenation-based approach has been used in a variety of domains including text classification (Yogatama et al., 2017), personalized conversational agents (Wen et al., 2013), and voice search queries (Ma et al., 2018).

Recently, attention mechanisms, initially developed for machine translation (Bahdanau et al., 2014; Luong et al., 2015), have been used by neural LMs to adaptively condition their predictions on certain non-linguistic contexts. Tang et al. (2016) use an attention module that attends to word-location information to predict the polarity of a sentence. Similarly, Zheng et al. (2019) use an attention mechanism over learned personality trait embeddings, in order to generate personalized dialogue responses.

The aforementioned approaches learn a representation of the context that is directly used as input to a neural LM. In contrast to these methods, Jaech and Ostendorf (2018a) adapt the weight-

---

[1] We make a large portion of our code available under: https://github.com/amazon-research/contextual-attention-nlm

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1994–2003
August 1–6, 2021.

matrix used in an RNN model to a given contextual input. The authors propose decomposing the weight-matrix into a set of left and right basis tensors which are then multiplied by a learned context embedding to produce a new weight-matrix. Factorizing the weight-matrix enables a larger fraction of a model's parameters to adjust to a given contextual signal. This factorization-based approach has proven effective in generating automatic completions of sentences that are personalized for particular users (Jaech and Ostendorf, 2018b).

We introduce an attention mechanism that augments both the concatenation-based and factorization-based approaches to condition a neural LM on context. The attention mechanism that we propose builds up a dynamic context representation over the course of processing an utterance. The resulting embedding can be used as an additional input to either the concatenation-based or factorization-based model.

Our experiments focus primarily on conditioning neural LMs on datetime context. We concentrate on datetime information because of its widespread availability in many ASR systems. Our approach, however, can generalized to any type of context. To underscore this point we also provide results for conditioning LMs on geolocation information and dialogue prompts that are commonly available in ASR systems.

We evaluate our method on a large de-identified dataset of transcribed utterances. Compared to a standard model that does not include contextual information, using our method to contextualize a neural LM on datetime information achieves a relative reduction in perplexity of 7.0%, and a relative reduction in perplexity of 9.0% when evaluated on the tail of this dataset. Moreover, our attention mechanism can improve state-of-the-art methods for conditional LMs by over 2.8% relative in terms of perplexity.

2 Data

We use a corpus of over 5,000 hours of de-identified, transcribed English utterances, collected over several years. Each utterance also contains associated information about the year, month, day, and hour that the utterance was spoken. The datetime information is reported according to the local time zone of each given user. Any information about the device or the speaker from which an utterance originates has been removed. We randomly split our dataset into a training set, development set and test set, using a partition ratio of 90/5/5 and we ensure that each partition contains more than 500 hours worth of data.

3 Context Representation

A typical utterance in our dataset might look like this:

2020-12-23 07:00 play christmas music.

In the example above, we can infer that the utterance "play christmas music" was spoken on December 23, 2020 at 7 in the morning local time. In order to condition a LM on this datetime information, we consider two methods for transforming the contextual information into a continuous vector representation:

1. **Learned embeddings:** We first consider creating tokens for the month number, week number, day of the week and hour that an utterance was spoken. In the example above, we would transform the datetime information into tokens representing: month-12, week-52, wednesday, 7am. These tokens are subsequently used as input to the model, where they are passed through an embedding layer to generate context embeddings. These embeddings are initialized as random vectors, and trained along with the rest of the model. We experiment with different ways of parsing the information, such as encoding weekday versus weekend, or morning versus evening, but find this information is largely entailed within our method for processing datetime information.

2. **Feature-engineered representation:** Additionally, we consider transforming the datetime information into a single 8-dimensional feature-engineered vector, where the dimensions of the vector are defined as

$$\begin{bmatrix} sin(\frac{2\pi \cdot hour}{24}) & cos(\frac{2\pi \cdot hour}{24}) \\ sin(\frac{2\pi \cdot day}{7}) & cos(\frac{2\pi \cdot day}{7}) \\ sin(\frac{2\pi \cdot week}{53}) & cos(\frac{2\pi \cdot week}{53}) \\ sin(\frac{2\pi \cdot month}{12}) & cos(\frac{2\pi \cdot month}{12}) \end{bmatrix}$$

Since the datetime context is continuous and cyclical, this approach explicitly encodes temporal proximity in the date and time information.

3.1 Input Representation

We assume as input to a model a sequence of either word or subword tokens, w_i for $i \in \{1, \dots, n\}$, that are converted by an embedding layer into embeddings $x_i \in \mathcal{R}^e$ for $i \in \{1, \dots, n\}$, where n is the length of the input sequence and e is the dimensionality of the word embeddings.

We additionally represent the contextual information as either:

1. A set, M, of four learned context embeddings $M = \{m_1, m_2, m_3, m_4\}$, where m_1 is an encoding of the month information, m_2 is an encoding of the week information, m_3 is an encoding of the day of the week information, and m_4 is an encoding of the hour of the day information. When using the concatenation-based or factorization-based approaches without attention, we first concatenate the embeddings together, $m = [m_1; m_2; m_3; m_4]$, and use the resulting vector as input to the model.
2. A set, M, containing a single embedding $M = \{m\}$, where m represents an 8-dimensional feature-engineered contextual datetime representation, as described in the previous section.

4 Model

In this section, we first describe the architecture of the concatenation-based and factorization-based approaches. We then introduce our attention-mechanism that can be used to augment both of these approaches. The notation we use to describe architectures assumes a 1-layer RNN model. The methods we discuss, however, can be applied to each layer of a multi-layer RNN model.

4.1 Concatenation-based LM Adaptation

The concatenation-based approach learns a weight matrix $\mathbf{W_m}$ of dimensionality $\mathcal{R}^{f \times d}$, where f represents the size of the context representation and d represents the hidden-dimensionality of the RNN model. In practice, f is either a hyperparameter when datetime context is represented as learned embeddings, or $f = 8$ when this context is represented as a feature-engineered vector. When representing contextual information as learned embeddings, recall that we first concatenate the embeddings together before passing these into the model. In this case, f is four-times the size of each individual context embedding.

A standard RNN model without contextual information keeps track of a hidden-state at time-step t, h_t, that is calculated as

$$h_t = \sigma(\mathbf{W_x} x_t + \mathbf{W_h} h_{t-1} + b),$$

where x_t represents the word embedding at time-step t, b is a bias vector, $\mathbf{W_x} \in \mathcal{R}^{e \times d}$, and $\mathbf{W_h} \in \mathcal{R}^{d \times d}$.

In the concatenation-based approach, this hidden-state is adapted in the following manner

$$h_t = \sigma(\mathbf{W_m} m + \mathbf{W_x} x_t + \mathbf{W_h} h_{t-1} + b).$$

Notice that the expression above can be equivalently calculated by concatenating the matrices $\mathbf{W_m}$ and $\mathbf{W_x}$, as well as the vectors m and x_t

$$h_t = \sigma([\mathbf{W_x}; \mathbf{W_m}][x_t; m] + \mathbf{W_h} h_{t-1} + b).$$

To generate a prediction, $\hat{y}_t$ for a word at time-step t, h_t is passed through a projection layer, $W_v \in \mathcal{R}^{d \times |V|}$ to match the dimension of the vocabulary size $|V|$, before applying a softmax layer

$$\hat{y}_t = softmax(W_v h_t).$$

4.2 Factorization-based LM Adaptation

Unlike the concatenation-based approach, which directly inserts contextual information into the RNN cell, the factorization-based method adapts the weight matrices $\mathbf{W_x}, \mathbf{W_h}$ of the RNN model. Compared to the concatenation-based architecture, this approach adapts a larger fraction of the RNN model's parameters.

The adaption process involves learning basis tensors $\mathbf{W_{x'}^{(L)}}, \mathbf{W_{x'}^{(R)}}$ and $\mathbf{W_{h'}^{(L)}}, \mathbf{W_{h'}^{(R)}}$. These basis tensors are of fixed rank r, where r is a tuned hyperparameter. The left adaptation tensors, $\mathbf{W_{x'}^{(L)}}, \mathbf{W_{h'}^{(L)}}$, are of dimensionality $\mathcal{R}^{f \times e \times r}$, and $\mathcal{R}^{f \times d \times r}$, respectively. The right adaptation tensors, $\mathbf{W_{x'}^{(R)}}, \mathbf{W_{h'}^{(R)}}$ are both of dimensionality $\mathcal{R}^{r \times d \times f}$. We can now use the contextual representation to interpolate the two sets of basis tensors to produce two new weight matrices, $\mathbf{W'_x}$ and $\mathbf{W'_h}$, where

$$\mathbf{W'_x} = \mathbf{W_x} + (\mathbf{W_{x'}^{(L)}}^T m)^T (\mathbf{W_{x'}^{(R)}}^T m)$$

$$\mathbf{W'_h} = \mathbf{W_h} + (\mathbf{W_{h'}^{(L)}}^T m)^T (\mathbf{W_{h'}^{(R)}}^T m).$$

The resulting matrices $\mathbf{W'_x}, \mathbf{W'_h}$ are now used as the weights in the RNN cell. A prediction, $\hat{y}_t$, is generated in the same manner as in the concatenation-based model.

4.3 Attention Mechanism

We propose an attention mechanism that augments both the concatenation-based and factorization-based approaches. We apply this mechanism to the context embeddings at each time-step of the RNN model, in order to adapt the context representation dynamically. We hypothesize that at certain time-steps within an utterance, attending to particular datetime information will facilitate the model's predictions more than other information.

For instance, assume a LM is given the phrase "what temperature will it be on friday". By the time the model has observed the words "what temperature will", we would expect the model to condition the predictions of the remaining words primarily on the hour and day information. Using an attention mechanism enables us to dynamically weight the importance that the model places on particular datetime context as the model processes an utterance.

We assume as input to the attention mechanism the same set M of context representations. However, in the case where datetime information is represented as a feature-engineered vector, we augment M to include an 8-dimensional vector of all 0s: $M = \{m, \mathbf{0}\}$. We do so because our attention mechanism builds a dynamic representation of the context by interpolating over multiple context embeddings. Thus, the attention mechanism can act as a learnable gate to limit the non-linguistic context passed into the model. We also experiment with adding a similar vector of all 0s in the case where context embeddings are learned, but find no improvement.

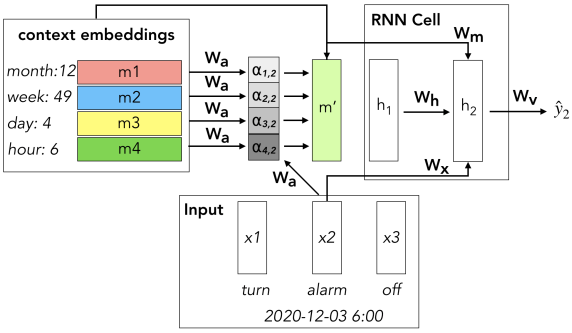

Figure 1: Model architecture of the concatenation-based model using attention. Datetime context is encoded as learned embeddings, and the input word embedding at time-step *t* is used as the query vector at *t*.

In addition to the set M, the attention mechanism takes in a query vector, q_t, for each time-step *t*. We propose two methods for defining this query vector

1. Let $q_t = x_t$, where x_t is the embedding for the input word at time-step *t*.
2. Let $q_t = h_t$, where h_t is the hidden state of the RNN model at time-step *t*.

Importantly, when q_t is chosen such that $q_t = x_t$, we can parallelize the computation of the attention mechanism for all time-steps before running a forward pass through the model. This cannot be done when $q_t = h_t$, as the attention mechanism can only be computed sequentially for each hidden state of the model.

Regardless of the choice of q_t, the attention mechanism first computes a score for each context embedding $m_i \in M$ for a given q_t. To compute this score, we learn a weight matrix W_a. The size of W_a is $\mathcal{R}^{f \times e}$ if $q_t = x_t$, or $\mathcal{R}^{f \times d}$ if $q_t = h_t$.

For a given $m_i \in M$ and q_t, we calculate a score as

$$score(m_i, q_t) = m_i^T \mathbf{W_a} q_t.$$

We then define the alignment score as

$$\alpha_{i,t} = softmax(score(m_i, q_t))$$

The alignment scores are finally used to build up a dynamic representation of the context, m'_t, for a given time-step.

$$m'_t = \sum_{i=1}^{|M|} \alpha_{i,t} m_i$$

We can now use this constructed context, as the context input to either the concatenation-based or the factorization-based approach.

In Figure 1 we illustrate how the attention mechanism augments the concatenation-based approach. For an utterance like "turn alarm off", we showcase how the model builds a dynamic representation of the datetime context, at a given time-step *t* (*t* = 2 in the figure).

5 Experimental Setup

We used a 1-recurrent-layer LSTM model (Hochreiter and Schmidhuber, 1997) as the base model in all of our experiments. Both the concatenation-based and factorization-based methods can be easily adapted to use an LSTM cell. Models were trained using the Adam optimizer with an initial learning rate of 0.001, and a standard cross entropy loss function. Each of the LMs was trained for

Method	Context Approach	Attention Query	Relative Perplexity Reduction		
			Full	Head	Tail
Default	NA	NA	0%	0%	0%
Prepend	Embeddings	NA	0.83%	**2.54%**	0.12%
Prepend	Feature-engineered	NA	**1.20%**	2.30%	**0.74%**
Concat	Embeddings	NA	6.82%	2.82%	8.68%
Concat	Embeddings	Hidden	7.00%	2.83%	8.91%
Concat	Embeddings	Word	**7.02%**	**2.89%**	**8.96%**
Concat	Feature-engineered	NA	6.82%	2.65%	8.71%
Concat	Feature-engineered	Hidden	7.0%	2.71%	8.97%
Concat	Feature-engineered	Word	6.94%	2.60%	8.94%
Factor	Embeddings	NA	3.29%	2.26%	3.93%
Factor	Embeddings	Hidden	4.82%	2.53%	5.96%
Factor	Embeddings	Word	5.40%	**2.58%**	6.71%
Factor	Feature-engineered	NA	5.44%	2.00%	**7.10%**
Factor	Feature-engineered	Hidden	**5.57%**	2.31%	6.82%
Factor	Feature-engineered	Word	5.05%	2.25%	6.31%

Table 1: Test set perplexities of contextual LMs based on datetime information, with relative reductions compared to the *Default* LSTM model that does not use contextual datetime information. We bold best results within each type of method, and underline best results overall. Improvements in perplexity from using our attention mechanism are statistically significant.

400,000 batch update steps, using a batch-size of 256. The training of each model was conducted on a single V100 GPU, with 16GB of memory on a Linux cluster, and took roughly 6 hours to train. Implementation of the model and training procedure was written in PyTorch and native PyTorch libraries. We used a fixed dimensionality of 512 for word, context and hidden state embeddings. We initially experimented with smaller and larger embedding sizes (50, 100, 1024), but found that 512 generally provided a good tradeoff between model performance and compute resources required to train a model. We set the rank of the basis tensors in the factorization-approach to 5, after experimenting with rank sizes 2, 3, 10, 15, 20. In practice, we found that the larger the rank size the less stable the training procedure became. Other hyperparameters, such as the initial learning rate, were selected via random search. We initialized random weights using Xavier-He weight initialization (He et al., 2015).

6 Results

6.1 Datetime

We evaluated our models on a heldout set of utterances that were randomly sampled from the full dataset. The utterances in our training and evaluation set were collected in the same time-range. We also defined the head and tail subsets of our development set, representing, respectively, the top 5% most frequently occurring utterances, and utterances occurring only once.

We used two metrics for our evaluations: perplexity and word error rate. Perplexity is a common statistic widely used in language modeling and speech recognition to measure how well a language model predicts a sample of text (Jelinek et al., 1977). Word error rate, on the other hand, measures the Levenshtein (minimum edit) distance between a recognized word sequence and a reference word sequence. In practice, these two statistics have been shown to be correlated by a power law relationship (Klakow and Peters, 2002).

In Table 1, we report the relative decrease in perplexity of models that leveraged datetime context compared to a baseline LSTM model that did not use any contextual information. We additionally trained a simple baseline, *Prepend*, which was comprised of a standard LSTM model that treated datetime context as input tokens that were prepended to the input texts.

In reporting our results, we distinguish between

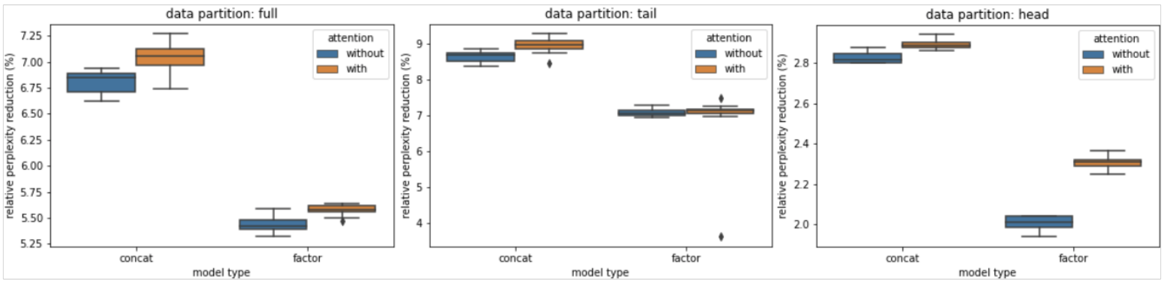

Figure 2: Perplexity confidence bounds at a 95% confidence level for the best-performing concatenation-based and factorization-based models with and without attention. Bounds are evaluated on the full, head and tail partitions of the evaluation set. Perplexity reductions are relative to the *Default* LSTM model that does not use contextual datetime information.

the two forms of representing contextual information: either as learned embeddings or as a feature-engineered representation. We also differentiate between the two variants of encoding the query vector used by the attention mechanism: either by using the hidden state vector, or the input embedding at a given time-step.

For the concatenation-based model, we found that adding our attention mechanism led to further reductions in perplexity, regardless of the type of query vector or context representation used. We obtained the best results when representing datetime information as learned embeddings and using the input embedding at a given time-step as the query vector.

We corroborated these results by computing 95% confidence intervals for the best-performing concatenation-based models with and without attention. Confidence intervals were calculated by running the training algorithm 10 times for each model type. Figure 2 visualizes the intervals.

In the case of the factorization-based approach, we achieved the lowest perplexity when the attention mechanism used the hidden state of the RNN model as the query vector and datetime information was represented as a feature-engineered vector. Again, we found that on the full dataset the improvement in perplexity by using our attention mechanism was statistically significant.

In nearly every experiment we ran, we found that our attention mechanism further reduced perplexity. The use of attention led to the largest relative improvement in the factorization-based approach when using learned context embeddings. In this instance, perplexity was reduced by 2.8% on the tail of our evaluation set, and by 2.1% on the full dataset.

Method	Context Approach	Attention Query	WERR (%)	
			Full	Tail
Default	NA	NA	0.0	0.0
Prepend	FE	NA	0.0	0.0
Concat	Emb	Word	**1.1**	**1.2**
Factor	FE	Hidden	0.8	0.8

Table 2: We report relative improvement (decrease) in WER compared to the *Default* LSTM model that does not use contextual information. Context representation approaches are feature-engineered (FE) or embeddings (Emb).

In addition to evaluating the models on relative reductions in perplexity, we also validated the downstream performance of a hybrid CTC-HMM (Graves et al., 2006) ASR system that incorporated contextual information in its LM component. As the LM component of this system, we used the best-performing models within each category of method that we report in Table 1. Table 2 summarizes the results. We evaluated the relative WER reduction(WERR) on a large test set of de-identified, transcribed utterances representative of general user interactions with Alexa, as well as on the tail of this dataset. As in Table 1, the concatenation-based model with attention mechanism achieved the largest reductions in WER.

6.2 Other Non-Linguistic Context

So far, our experiments have focused exclusively on conditioning neural LMs on datetime context. We underscore, however, that the contextual mechanism we introduce can be applied to any type of contextual information that can be represented as embeddings. To illustrate this point we train two neural LMs using two other types of context: geolocation information and dialogue prompts.

Context Type	Relative PPL Reduction (%)		WERR (%)	
	Full	Tail	Full	Tail
Default	0.0	0.0	0.0	0.0
Datetime	11.4	11.6	<u>1.6</u>	<u>1.7</u>
Geo-hash	12.4	12.5	0.5	1.0
Dialogue Prompt	<u>13.9</u>	<u>14.1</u>	0.3	0.6

Table 3: We report relative improvement (decrease) in WER and (decrease) in perplexity (PPL) compared to the *Default* LSTM model that does not use contextual information. We report results on both the full test dataset as well as utterances from the tail of the dataset. The best results are underlined.

We train the LMs on a subset of the utterances of the initial dataset which also contain utterance-level geo-hash information and dialogue prompt information. The geo-hash information [2] associated with each utterance encodes a very rough estimate of the geolocation of a user's device. Dialogue prompts indicate whether a transcribed utterance was an initial query to the dialog system or if it was a follow-up turn.

We learn embeddings to represent both the geo-hash and the dialogue prompt information. We ingest both types of contexts via the concatenation-based approach, using word-embeddings as the attention queries. We evaluate these models on a test set of de-identified utterances representative of user interactions with Alexa. Table 3 summarizes the results. In general, we find that conditioning neural LMs on each of the different types of context reduces perplexity and WER.

7 Analysis

In this section, we focus once again on datetime information to better understand how contextual LMs use datetime signal.

7.1 Datetime Context Signal

The first question we hope to answer is: to what extent can the relative improvements in perplexity and WER in the models that incorporate datetime context be explained by the additional signal from the context versus the additional parameters that these models contain?

To answer this question, we randomly shuffled the datetime information associated with each utterance in our training and test sets. For each of our best-performing models in a given category of method (prepend, concat, or factor), we retrained and evaluated those models on the dataset containing shuffled datetime information.

In Table 4, we report the relative degradation (i.e., a negative reduction) in perplexity resulting from evaluating these models on the shuffled datetime contexts. In general, if a model uses datetime information as an additional signal, we would expect the performance of the model to decrease when the datetime context is shuffled.

Method	Context Approach	Attention Query	Relative PPL Reduction (%)	
			Full	Tail
Prepend	FE	NA	-1.5	-1.2
Concat	Emb	Word	-1.6	-1.2
Factor	FE	Hidden	-1.4	-1.0

Table 4: Relative degradation in perplexity (PPL) of models that incorporate datetime information when that context is randomly shuffled. Context representations are feature-engineered (FE) or embeddings (Emb).

We observed the overall largest relative degradation in perplexity, when using the concatenation-based model. Recall that when trained on correct datetime information this was our best-performing model overall in terms of both perplexity and WER, indicating that the performance of this model can be attributed in part to its use of contextual information.

7.1.1 Visual Analysis

In addition to these results, we visualize how the contextual LMs leverage datetime contexts. For a given utterance, we can evaluate the probability of the words in the utterance as we vary the datetime information associated with the utterance. In Figure 3, we evaluate the conditional probability of the word "snooze" in an utterance following the start-of-sentence token, as we vary the hour of day information associated with this utterance. As we would expect, the probability of this "snooze" is highest in the morning (between 5 and 6 am), as users are waking up and snoozing their alarms. As we move away from the morning hours, the conditional probability of the word "snooze" decreases substantially, reaching a low-point by the afternoon and evening. The horizontal blue dashed line

---

[2] We use a two integer precision geo-hash.

indicates the conditional probability of the word 'snooze' following the start-of-sentence token when evaluated with a LM that does not ingest datetime information. This analysis further corroborates that the trained contextual LMs successfully condition their predictions on datetime information.

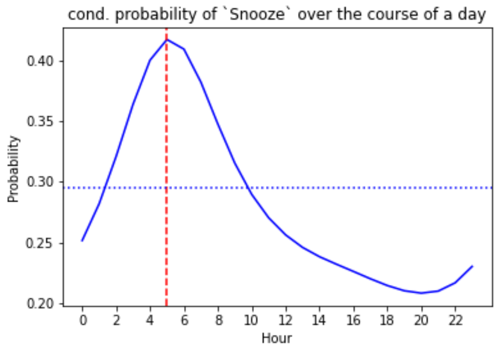

Figure 3: Changing conditional probability of the word "snooze" as the associated hour of day information varies.

7.2 Attention Weights

We next seek to understand how the attention mechanism constructs a dynamic representations of datetime context. To do so, we visualize the weights of the attention mechanism as an utterance is processed by the model. For a given utterance like "play me best christmas songs" spoken in December, we highlight the changing weight placed on each of the datetime information. Figure 4 shows this analysis.

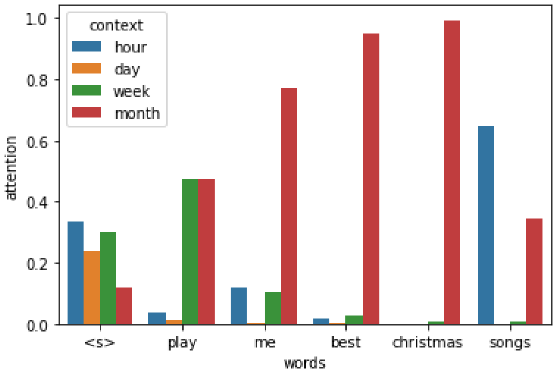

Figure 4: Changing attention weights placed on particular context embeddings over the course of an utterance.

When the model processes the start-of-sentence token, the attention mechanism weights each of the datetime information roughly equally. However as the model processes the subsequent words "play me best", the attention begins to shift towards using more of the month information (i.e., that this utterance was spoken in December), and away from hour and day information. This would suggest that conditioning on the fact that the utterance was spoken in December can help the model predict the type of media to play.

Once the model observes the word "christmas", it places all of the attention on the month information, indicating the model has successfully learned that "christmas" is a word strongly associated with a particular month (i.e., December). Finally when the word "songs" is ingested, the model substantially reduces the weight placed on month information and in turn increases the weight on hour information. This shift might indicate that the model has learned to condition the type of music users listen to to the hour of the day. Overall, the behavior of the attention mechanism is consistent with our initial hypothesis that certain types of datetime information can benefit a contextual LM model more than others over the course of an utterance.

8 Related Work

Within the domain of ASR, Biadsy et al. (2017) have explored using an adaptive-training approach to incorporate non-linguistic features into a maximum entropy LM. They propose first training the parameters of a LM that are associated with text data, then freezing those parameter and learning parameters associated with multiple types of non-linguistic features.

Zhang et al. (2019) and Yoon et al. (2017) propose a similar two-pronged approach for personalizing conversational neural LMs. They propose first pretraining a RNN-LM on a large dataset of conversational data, then finetuning the model on data associated with a particular user.

More recently, Jain et al. (2020) and Liu et al. (2020) propose attention mechanisms for conditioning RNN-T and hybrid ASR systems on words that are likely to occur in an utterance.

Another related line of research has explored learning utterance embeddings for dialogue systems using Gaussian mixture models that are enhanced with utterance-level context, such as intent (Yan et al., 2020).

Outside of ASR, our work directly builds upon the concatenation-based (Mikolov and Zweig, 2012) and factorization-based (Jaech and Ostendorf, 2018a) approaches to condition RNN-LMs on sentence context. The concatenation-based approach has been adopted as a common method for incorporating non-linguistic context into a neural LM (Yogatama et al., 2017; Wen et al., 2013; Ma et al., 2018; Ghosh et al., 2016). Methods that apply low-rank matrix factorization to RNNs are somewhat newer, and were first explored by Kuchaiev and Ginsburg (2017).

Our contribution lies first in the application of these models to ASR, and secondly their extension with an attention mechanism. The attention mechanism we propose builds on the global attention model proposed by Luong et al. (2015). While attention-based models have been used to condition neural models on particular aspects or traits (Zheng et al., 2019; Tang et al., 2016), we focus on contextual information that benefits ASR systems.

9 Conclusion

In this paper, we introduce an attention-based mechanism to condition neural LMs for ASR on non-linguistic contextual information. The proposed model dynamically builds up a representation of contextual information that can be ingested into a RNN-LM via a concatenation-based or factorization-based approach. We find that incorporating datetime context into a LM can yield a relative reduction in perplexity of 9.0% over a model that does not incorporate context. Moreover, the attention mechanism we propose can improve state-of-the-art contextual LM models by over 2.8% relative in terms of perplexity. While we focus on datetime information, we demonstrate that our approach can be applied to any type of non-linguistic context, such as geolocation and dialogue prompts.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR abs/1409.1257*.

Fadi Biadsy, Mohammadreza Ghodsi, and Diamantino Caseiro. 2017. Effectively building tera scale MaxEnt language models incorporating non-linguistic signals. In *Proc. Interspeech*.

E. Bocchieri and D. Caseiro. 2010. Use of geographical meta-data in ASR language and acoustic models. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5118–5121.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. *arXiv preprint arXiv:1602.06291*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 369–376.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. 2019. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aaron Jaech and Mari Ostendorf. 2018a. Low-rank RNN adaptation for context-aware language modeling. *Transactions of the Association for Computational Linguistics*, 6:497–510.

Aaron Jaech and Mari Ostendorf. 2018b. Personalized language model for query auto-completion. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 700–705.

Mahaveer Jain, Gil Keren, Jay Mahadeokar, and Yatharth Saraf. 2020. Contextual RNN-T for open domain ASR. In *Proc. Interspeech*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for LSTM networks. *CoRR abs/1703.10722*.

Da-Rong Liu, Chunxi Liu, Frank Zhang, Gabriel Synnaeve, Yatharth Saraf, and Geoffrey Zweig. 2020. Contextualizing ASR lattice rescoring with hybrid pointer network language model. In *Proc. Interspeech*.

Matthew I Lloyd and Trausti Kristjansson. 2012. Acoustic model adaptation using geographic information. US Patent 8,219,384.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Min Ma, Shankar Kumar, Fadi Biadsy, Michael Nirschl, Tomas Vykruta, and Pedro Moreno. 2018. Modeling non-linguistic contextual signals in LSTM language models via domain adaptation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6094–6098. IEEE.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Tsung-Hsien Wen, Aaron Heidel, Hung-Yi Lee, Yu Tsao, and Lin-Shan Lee. 2013. Recurrent neural network based personalized language modeling by social network crowdsourcing. In *Proc. Interspeech*.

Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert YS Lam. 2020. Unknown intent detection using Gaussian mixture model with an application to zero-shot intent classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1050–1060.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Seunghyun Yoon, Hyeongu Yun, Yuna Kim, Gyu-tae Park, and Kyomin Jung. 2017. Efficient transfer learning schemes for personalized language modeling using recurrent neural network. *Association for the Advancement of Artificial Intelligence (AAAI) Workshop*.

Wei-Nan Zhang, Qingfu Zhu, Yifa Wang, Yanyan Zhao, and Ting Liu. 2019. Neural personalized response generation as domain adaptation. *World Wide Web*, 22(4):1427–1446.

Yinhe Zheng, Guanyi Chen, Minlie Huang, Song Liu, and Xuan Zhu. 2019. Personalized dialogue generation with diversified traits. *CoRR abs/1901.09672*.

Annotation and Evaluation of Coreference Resolution in Screenplays

Sabyasachee Baruah **Sandeep Nallan Chakravarthula** **Shrikanth Narayanan**
`sbaruah@usc.edu` `nallanch@usc.edu` `shri@ee.usc.edu`
Signal Analysis and Interpretation Laboratory
University of Southern California

Abstract

Screenplays refer to characters using different names, pronouns, and nominal expressions. We need to resolve these mentions to the correct referent character for better story understanding and holistic research in computational narratology. Coreference resolution of character mentions in screenplays becomes challenging because of the large document lengths, unique structural features like scene headers, interleaving of action and speech passages, and reliance on the accompanying video. In this work, we first adapt widely-used annotation guidelines to address domain-specific issues in screenplays. We develop an automatic screenplay parser to extract the structural information and design coreference rules based upon the structure. Our model exploits these structural features and outperforms a benchmark coreference model on the screenplay coreference resolution task.

1 Introduction

Screenplays are semi-structured text documents containing the dialogue and directions of a film. Automated screenplay analysis provides an opportunity early in the creative process to offer insights into character representations and portrayals (of who interacts with whom, about what, and how), including from a diversity, inclusion, and social impact perspective (Ramakrishna et al., 2017; Shafaei et al., 2020; Martinez et al., 2020). A typical screenplay contains indented blocks of text that can be classified into scene headers, scene descriptions, speakers, and utterances, as shown in Figure 1 (Agarwal et al., 2014). A scene header starts a new scene and provides location and temporal information. Scene descriptions describe the characters and their actions, and the speaker and utterance blocks contain the characters' names and speech.

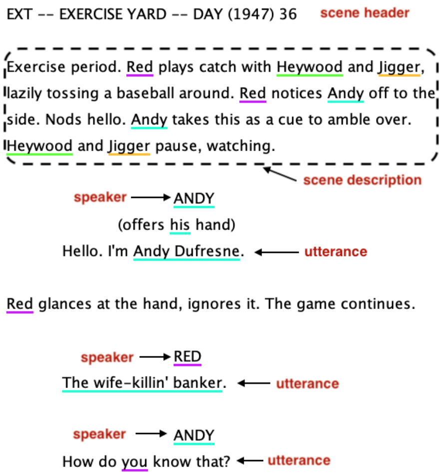
EXT -- EXERCISE YARD -- DAY (1947) 36

Exercise period. Red plays catch with Heywood and Jigger, lazily tossing a baseball around. Red notices Andy off to the side. Nods hello. Andy takes this as a cue to amble over. Heywood and Jigger pause, watching.

ANDY
(offers his hand)
Hello. I'm Andy Dufresne.

Red glances at the hand, ignores it. The game continues.

RED
The wife-killin' banker.

ANDY
How do you know that?

Figure 1: Coreference-annotated screenplay excerpt from the movie *The Shawshank Redemption (1994)*. Mentions of the same character are underlined with the same color.

Screenplays can refer to a character by different names, pronouns, and nominal expressions. For example, the screenplay excerpt shown in Figure 1 refers to the character *Andy* by the mentions – *Andy Dufresne* (name), *The wife-killin' banker* (nominal), and *his* (pronoun). Many downstream tasks need to find and map all such mentions to the correct *referent*. For example, Gorinski and Lapata (2015) resolved pronominal mentions to their correct antecedent (prior co-referring mention) to find speaker-listener and semantic relations between characters for the movie-summarization task. Chen and Choi (2016) crowdsourced character-mention labels in TV show transcripts for automatic character identification. Deleris et al. (2018) used direct and indirect character references in utterances to study social relationships between characters. Thus, resolving character mentions in screenplays is an essential subtask in many applications. In NLP literature, this task is formally called coreference resolution (Jurafsky and Martin, 2009) and has been studied extensively (Sukthanker et al., 2020; Stylianou and Vlahavas, 2021).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2004–2010
August 1–6, 2021. ©2021 Association for Computational Linguistics

Most existing coreference datasets focus on news and web text (Pradhan et al., 2012; Webster et al., 2018), but do not include screenplays that have distinct content and structure. First, the document lengths of screenplays are much larger than news articles (Gorinski and Lapata, 2015), increasing the computational complexity of antecedent scoring (Lee et al., 2017). Second, scene headers alter the story's context affecting coreference between mentions of different scenes. Lastly, coreference annotation of some mentions requires the knowledge of the accompanying movie or TV clip because textual descriptions may not capture all the post-production visual details of a scene. Thus, two main challenges for coreference resolution in screenplays are: (1) the lack of suitable annotation rules to handle domain-specific issues and (2) finding methods to leverage the unique screenplay structure to improve coreference resolution.

Our objective in this work is to address the coreference resolution of characters in screenplays. We only focus on characters because most modern narrative studies are centered around their role as agents driving the plot (Bamman et al., 2013; Labatut and Bost, 2019). Our contributions are 1) we establish coreference annotation guidelines for screenplays and use them to label screenplay excerpts, 2) we develop a screenplay parser to convert the semi-structured text into a machine-readable format, and 3) we use the structural information of screenplays to design coreference rules, which improves the performance of coreference resolution when combined with a benchmark coreference resolution model (Lee et al., 2018).

2 Related Work

Screenplay Parsing: Weng et al. (2009) motivated the need for screenplay parsing for social network analysis. Agarwal et al. (2014) formalized the screenplay parsing task and developed an SVM-based parser. Winer and Young (2017) extended it to extract fine-grained information from scene headers. Our parser uses a rule-based algorithm to achieve comparable performance.

Coreference Resolution: OntoNotes 5 is the benchmark dataset for English coreference resolution, containing documents from newswire, broadcast news, telephone conversations, and weblogs (Pradhan et al., 2012). However, it does not contain screenplay texts. The closest work to screenplay coreference is the *LitBank* dataset (Bamman et al., 2020), which contains coreference annotations of 100 works of fiction taken from Project Gutenberg (Lahiri, 2014).

Few previous works address coreference annotation in screenplays. Chen and Choi (2016) labeled character mentions for two TV shows: *Friends* and *The Big Bang Theory*. Zhou and Choi (2018) later extended this dataset by including plural mentions, but mainly focused on character utterances and did not consider action notes. Ramanathan et al. (2014) created a dataset of 19 TV episodes for joint coreference resolution in visual and textual media content. Gorinski and Lapata (2015) created the *ScriptBase* corpus of movie screenplays, which included coreference labels. However, they found the labels using the Stanford CoreNLP system (Lee et al., 2011), which has not been evaluated for screenplay coreference.

3 Annotation

We annotate screenplays with character mentions for coreference resolution (see Figure 1). Following OntoNotes 5 annotation guidelines, we mark the maximal extent of noun phrases, pronouns, and possessives that refer to some character (Pradhan et al., 2012). Characters do not have to be persons; consider for example, the spider *Aragog* in the *Harry Potter* movies. We include singleton character entities. We do not label mentions that refer to multiple characters because finding the correct antecedent of plural mentions often requires the accompanying video clip's aid (Zhou and Choi, 2018). For example, it is difficult to decide whether *They* refers to *Vosen and agents*, *Others*, or both in the following lines without watching the movie.

> [Vosen and agents]$_x$ come running out of the front door. [Others]$_y$ leave through a side entrance. [They] jump in sedans. (*Bourne Ultimatum, 2007*)

We follow OntoNotes 5 annotation guidelines to handle appositions (adjacent noun phrases separated by comma, colon, or parentheses), copula (noun phrases connected by linking verbs, for example, *is*, *look*, etc.), and generic *you* mentions (Pradhan et al., 2012). If a mention's referent is revealed to be identical with another character as the story progresses, we tag the mention with the latter character (Bamman et al., 2020). The screenplay sometimes contains references to the reader or the camera's point of view. We tag such instances with a special READER entity, for example:

Movie	Tokens	Characters	Names	Pronouns	Nominals
The Shawshank Redemption (1994)	7,734	44	329	448	111
Bourne Ultimatum (2007)	7,722	39	491	344	76
Inglourious Basterds (2009)	7,284	23	426	458	124
TOTAL	22,740	106	1,246	1,250	311

Table 1: Statistics of the coreference annotated data.

> In the background, [we]$_{READER}$ see, [our]$_{READER}$ three counterfeit German Officers, Hicox, Wicki, and Stiglitz, enter the basement tavern. They obviously see the five German soldiers, but their too far away for [us (the audience)]$_{READER}$ to read their face. (*Inglourious Basterds, 2009*)

Screenplay Coreference Dataset: We annotated screenplay excerpts of three movies: *The Shawshank Redemption*, *Bourne Ultimatum*, and *Inglourious Basterds*. We downloaded the screenplay documents from IMSDb [1]. We chose these movies because the annotators were familiar with them, and they cover a wide range of genres (drama, action, thriller, and war). Three doctoral students were each assigned one screenplay, which they annotated for coreference according to the guidelines of section 3. The lead author checked the annotations independently for labeling errors. Less than 1% of the mentions required correction, suggesting high overall agreement. Table 1 describes some statistics of the labeled data. The corpus contains $2,807$ mentions in total, covering 106 characters. More than 44% of the mentions are pronouns, and about 11% are nominal mentions.

4 Model

Our coreference model consists of two parts: 1) a screenplay parser to extract structural information, and 2) coreference rules to resolve mentions occurring in speaker blocks.

4.1 Screenplay Parser

The screenplay parser reads raw screenplay text documents and assigns a structural tag – scene header, scene description, speaker, utterance or *other* (see Figure 1) to each line, following the tagset of Agarwal et al. (2014). The parser uses regular expressions to assign the structural tags. Thus, it is a rule-based 5-way multiclass classifier. The *other* tag includes all lines that do not fall in the other four categories, for example, camera transitions (*CUT TO*, *FADE IN*, etc.), dialogue metadata expressions (*O.S.*, *V.O.*, *(shouting)*, etc.), etc. The parser removes blank lines and whitespace indents.

4.2 Coreference Resolution

We use the following strategy to find the coreference clusters of characters in screenplays.

Add *says*: Given a screenplay, we parse it using our screenplay parser and collect all lines tagged as scene header, scene description, speaker, or utterance. We add the word *says* after speaker-tagged lines and concatenate all lines, separated by a newline delimiter. This lexical addition tells the model that the character mentioned in the speaker-tagged line speaks the succeeding utterance-tagged lines.

Keep speakers: We apply a coreference resolution model, pre-trained on OntoNotes 5, to the concatenated text to find coreference clusters. Since OntoNotes 5 annotates for unrestricted coreference (find coreference clusters of all ACE entities and events), we need to prune the clusters to keep only those containing character mentions. We keep clusters that contain any mention which appears in a speaker-tagged line.

Merge clusters: Due to the large document size (see Table 1), long coreference chains, like those of main characters, sometimes get segmented and occur as separate clusters. We merge the segmented clusters using speaker information. Screenplays usually refer to a character by a unique name in the speaker-tagged lines. If two speaker-tagged lines belonging to separate clusters contain identical names, we merge the corresponding clusters.

5 Experiments and Results

5.1 Screenplay Parser Evaluation

We annotated lines of 39 movie screenplay excerpts for the structural tags. These movies were different from the ones annotated for coreference. Three annotators, all doctoral students, labeled 9,758 lines, with a Krippendorff's inter-rater reliability score of 0.983 (strong agreement). We parsed the annotated excerpts using our rule-based parser and evaluated its classification performance. Table 3 shows the

---

[1] `http://www.imsdb.com`

Model	MUC			B^3			$CEAF_e$			Avg		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Baseline	83.7	62.0	70.8	55.3	33.9	41.6	28.1	23.6	23.8	55.7	39.8	45.4
Our Model	**87.3**	**79.5**	**83.2**	**65.8**	**61.2**	**63.2**	**50.2**	**40.6**	**41.6**	**67.7**	**60.4**	**62.7**
-Add says	-1.0	-2.5	-1.9	-9.3	-15.0	-12.7	-5.1	-3.6	-4.3	-5.1	-7.0	-6.3
-Keep speakers	-2.5	-2.2	-2.4	-2.4	-2.5	-2.4	-17.7	+3.5	-6.5	-7.5	-0.4	-3.7
-Merge clusters	-0.2	-2.9	-1.7	+3.8	-9.8	-5.3	-1.1	-6.3	-3.8	+0.8	-6.3	-3.6

Table 2: Coreference resolution performance. The last three rows show the change in scores when the corresponding rule is removed from our model.

precision, recall, and F1 scores of the parser on the annotated data. The performance is comparable with the parser developed by Agarwal et al. (2014).

Structure Tag	lines	P	R	F1
Scene Header	343	94.5	90.4	92.4
Scene Description	3074	88.2	87.6	87.9
Speaker	1833	95.4	96.1	95.7
Utterance	3754	90.5	94.5	92.5

Table 3: Screenplay parser's classification performance

5.2 Screenplay Coreference Evaluation

We evaluated our coreference model on the labeled screenplay data collected in section 3. We used the model of Lee et al. (2018), a widely used benchmark for coreference resolution, as the pre-trained model for our method. Lee et al. (2018) calculates mention scores for each span, retains the top-scoring ones and finds antecedent scores of mention pairs. We replaced the model's original GloVe vectors with SpanBERT (large) embeddings. Joshi et al. (2020) showed that the SpanBERT-adapted model obtains 79.6 avg. F1 on the OntoNotes 5 test set for the unrestricted coreference task.

Baseline: Given a screenplay, we find the structural tags for each line using our parser and retain the lines tagged as scene header, scene description, speaker, or utterance. We input the text to the coreference resolution model of Lee et al. (2018), pre-trained on OntoNotes 5. We keep clusters that contain any mention whose named entity tag is *PERSON*. We use spaCy (Honnibal et al., 2020) for named entity recognition.

Metrics: Following the official evaluation framework of CoNLL-2012 multilingual coreference task (Pradhan et al., 2014), we used the MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and $CEAF_e$ (Luo, 2005) measures to evaluate our model. We also used the NEC score (Agarwal et al., 2019) to evaluate coreference resolution by mention-type – name, pronoun and nominal. Tables 2 and 5 show the performance of our model and baseline in coreference resolution and character mention identification respectively. Table 4 shows the NEC scores for different mention types.

	Name	Pronoun	Nominal
Baseline	39.1	39.8	48.8
Our Model	**47.6**	**62.1**	**55.4**

Table 4: NEC F1 by mention type

	P	R	F1
Baseline	86.5	63.9	73.1
Our Model	**88.7**	**80.8**	**84.5**

Table 5: Mention identification performance.

Ablation Study: We study how each coreference rule – *Add says*, *Keep speakers*, and *Merge clusters* – described in section 4.2, contributes to the model's performance. Table 2 shows the results of the ablation experiments. *-Add says* means that we do not add *says* after speaker-tagged lines, *-Keep speakers* implies that we retain clusters that contain any mention whose named entity tag is *PERSON* instead of those that contain any mention appearing in speaker-tagged lines, and *-Merge clusters* denotes that we do not merge clusters.

6 Discussion

The results of Table 2 suggests that inputting the raw screenplay directly to the pre-trained coreference model performs poorly. The performance substantially improves when we use the coreference rules (+17.3 avg. F1). The improvement is largest for pronouns (+22.3 NEC F1), as shown in Table 4, possibly because of the *Add says* rule that helps the model to find the antecedent of personal pronouns in utterance-tagged lines. The rule adds

6.3 avg. F1 towards the overall performance (Table 2). Coreference resolution of named mentions also improves greatly (+8.5 NEC F1), probably because of the *Merge clusters* rule that joins clusters if they contain mentions in speaker-tagged lines that have identical names. It contributes 3.6 avg. F1 to the final score (Table 2). The *Keep speakers* rule adds 3.7 avg. F1, which suggests that retaining clusters containing speaker-tagged mentions is better than keeping those containing *PERSON*-tagged (NER) mentions to retrieve character references.

7 Applications

We show two applications of coreference resolution in computational narratology: finding mention-type interactions and character actions.

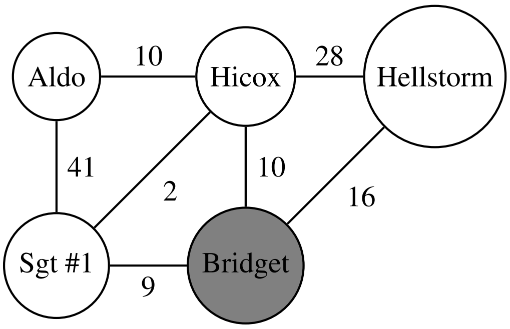

Figure 2: Character network of speech interactions. Edge weight is the number of interactions. Bridget has the **lowest** weighted degree.

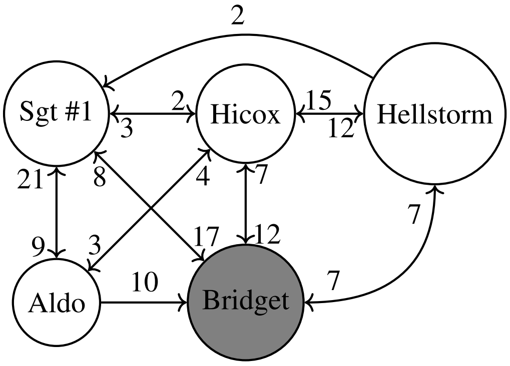

Figure 3: Character network of mention interactions. Edge weight is the number of mentions. Bridget has the **highest** weighted in-degree.

Mention-type Interactions: Figures 2 and 3 show character networks of the top five speaking characters from the movie *Inglourious Basterds (2009)*, capturing speech and mention-type interactions respectively. The edge weight between characters A and B in the speech network (Fig 2) is the number of times A speaks right after B or vice versa (Ramakrishna et al., 2017). The directed edge weight from character A to character B in the mention network (Fig 3) is the number of times A mentions B in their speech. We used the structural tags and coreference annotations to create these networks.

We observe that the two character networks provide different insights. Using degree centrality, *Bridget* is the least 'important' character in terms of speech interactions, but is most mentioned by other characters. This supports the movie plot which contains a scene where the Basterds discuss their plans of meeting *Bridget*, without her being there.

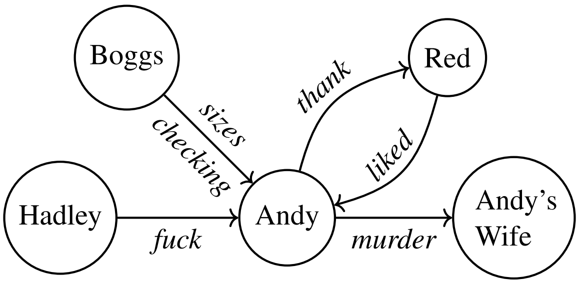

Figure 4: Character network of action interactions. Edge labels are actions. Head node of edge is the agent, and tail node of edge is the patient of the action.

Character Actions: We can use semantic role labeling and coreference resolution to find character actions. Figure 4 shows a subgraph of the character action network of *The Shawshank Redemption (1994)* movie. The directed edge label is the action, the head node is the agent (ARG0), and the tail node is the patient (ARG1) of the action. We applied the SRL model of Shi and Lin (2019) to the sreenplay's sentences, and then substituted the semantic roles with their referred character, wherever possible. From figure 4, we observe that *Andy* had positive interactions with *Red*, but was negatively treated by *Boggs* and *Hadley*, which is in line with the movie plot.

8 Summary and Future Work

We presented a coreference annotation guideline for screenplays and developed rules based on the screenplay's structure to improve coreference resolution performance. Our work can facilitate future annotation and modeling of coreference resolution in screenplays to support computational narratology studies. We plan to label more screenplays to train an end-to-end coreference model and study character interactions using coreference clusters. The data is available in the supplementary material.

References

Apoorv Agarwal, Sriramkumar Balasubramanian, Jiehan Zheng, and Sarthak Dash. 2014. Parsing screenplays for extracting social networks from movies. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 50–58, Gothenburg, Sweden. Association for Computational Linguistics.

Oshin Agarwal, Sanjay Subramanian, Ani Nenkova, and Dan Roth. 2019. Evaluation of named entity coreference. In *Proceedings of the Second Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 1–7, Minneapolis, USA. Association for Computational Linguistics.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

David Bamman, Olivia Lewke, and Anya Mansoor. 2020. An annotated dataset of coreference in English literature. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 44–54, Marseille, France. European Language Resources Association.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria. Association for Computational Linguistics.

Yu-Hsin Chen and Jinho D. Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in TV shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 90–100, Los Angeles. Association for Computational Linguistics.

Léa Deleris, Francesca Bonin, Elizabeth Daly, Stéphane Deparis, Yufang Hou, Charles Jochim, Yassine Lassoued, and Killian Levacher. 2018. Know who your Friends are: Understanding social connections from unstructured text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 76–80, New Orleans, Louisiana. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

Vincent Labatut and Xavier Bost. 2019. Extraction and analysis of fictional character networks: A survey. *ACM Computing Surveys (CSUR)*, 52(5):1–40.

Shibamouli Lahiri. 2014. Complexity of word collocation networks: A preliminary structural analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden. Association for Computational Linguistics.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, Portland, Oregon, USA. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Victor Martinez, Krishna Somandepalli, Yalda Tehranian-Uhls, and Shrikanth Narayanan. 2020. Joint estimation and analysis of risk behavior ratings in movie scripts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4780–4790, Online. Association for Computational Linguistics.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35, Baltimore, Maryland. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Anil Ramakrishna, Victor R. Martínez, Nikolaos Malandrakis, Karan Singla, and Shrikanth Narayanan. 2017. Linguistic analysis of differences in portrayal of movie characters. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1669–1678, Vancouver, Canada. Association for Computational Linguistics.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. 2014. Linking people in videos with "their" names using coreference resolution. In *European conference on computer vision*, pages 95–110. Springer.

Mahsa Shafaei, Niloofar Safi Samghabadi, Sudipta Kar, and Thamar Solorio. 2020. Age suitability rating: Predicting the MPAA rating based on movie dialogues. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1327–1335, Marseille, France. European Language Resources Association.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255.

Nikolaos Stylianou and Ioannis Vlahavas. 2021. A neural entity coreference resolution review. *Expert Systems with Applications*, 168:114466.

Rhea Sukthanker, Soujanya Poria, Erik Cambria, and Ramkumar Thirunavukarasu. 2020. Anaphora and coreference resolution: A review. *Information Fusion*, 59:139–162.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Chung-Yi Weng, Wei-Ta Chu, and Ja-Ling Wu. 2009. Rolenet: Movie analysis from the perspective of social networks. *IEEE Transactions on Multimedia*, 11(2):256–271.

David Winer and R Young. 2017. Automated screenplay annotation for extracting storytelling knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 13.

Ethan Zhou and Jinho D. Choi. 2018. They exist! introducing plural mentions to coreference resolution and entity linking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 24–34, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Exploring Cross-Lingual Transfer Learning with Unsupervised Machine Translation

Chao Wang [*]
York University
Toronto, Canada
chwang@eecs.yorku.ca

Judith Gaspers
Amazon Alexa AI
Aachen, Germany
gaspers@amazon.com

Quynh Do
Amazon Alexa AI
Aachen, Germany
doquynh@amazon.com

Hui Jiang
York University
Toronto, Canada
hj@eecs.yorku.ca

Abstract

In Natural Language Understanding (NLU), to facilitate Cross-Lingual Transfer Learning (CLTL), especially CLTL between distant languages, we integrate CLTL with Machine Translation (MT), and thereby propose a novel CLTL model named Translation Aided Language Learner (TALL). TALL is constructed as a standard transformer, where the encoder is a pre-trained multilingual language model. The training of TALL includes an MT-oriented pre-training and an NLU-oriented fine-tuning. To make use of unannotated data, we implement the recently proposed Unsupervised Machine Translation (UMT) technique in the MT-oriented pre-training of TALL. The experimental results show that the application of UMT enables TALL to consistently achieve better CLTL performance than our baseline model, which is the pre-trained multilingual language model serving as the encoder of TALL, without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

1 Introduction

Virtual assistants, such as Amazon Alexa, Apple Siri, and Google Assistant, are increasingly popular due to the convenience they bring to customers. A core function of virtual assistants is Natural Language Understanding (NLU), which is a combo of slot filling and intent classification. NLU models behind virtual assistants are generally trained in a supervised manner, which requires a large amount of annotated data. Collecting annotated data is not a big deal for high-resource languages, but difficult or even impossible for low-resource languages. As a result, when ported to a low-resource language, an NLU model may suffer from the so-called "data hungriness" (van der Ploeg et al., 2014). This problem can be alleviated by conducting Cross-Lingual Transfer Learning (CLTL) (Yarowsky et al., 2001), where annotated data in a high-resource source language is used to bootstrap an NLU model aimed at a low-resource target language.

The key to CLTL is to learn a shared representation space for the given source-target language pair. A traditional way to achieve this goal is to leverage cross-lingual word embeddings, which are obtained by mapping the words in both languages to a shared word embedding space (Zhang et al., 2017; Conneau et al., 2017; Artetxe et al., 2018a; Chen et al., 2018; Chen and Cardie, 2018; Chen et al., 2019). However, most studies on this topic only consider similar languages (e.g. English-German) but ignore distant languages (e.g. English-Japanese), since it is more challenging to conduct CLTL between distant languages than between similar languages. Recently, contextualized word embeddings generated by pre-trained language models have shown significant advantages over ordinary word embeddings (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019). For the purpose of CLTL, many efforts have been made to develop multilingual variants of pre-trained language models. These efforts have in turn brought about pre-trained multilingual language models, each of which is pre-trained on a multilingual corpus so that the learned representation space is not only rich in contextual clues but also shared by all the involved languages (Mulcaire et al., 2019; Conneau and Lample, 2019; Conneau et al., 2020). However, in this pre-training, the collection of the

---

[*] Work done during internship at Amazon Alexa AI.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2011–2020
August 1–6, 2021.

multilingual corpus is not obviously biased to any language, thus in the learned representation space, similar languages are still similar to each other, and distant languages are still distant from each other. As a result, although pre-trained multilingual language models have greatly promoted CLTL, it is still more challenging to conduct CLTL between distant languages than between similar languages. This opinion has been verified by several empirical studies on a popular pre-trained multilingual language model named Multilingual BERT (M-BERT) (Devlin et al., 2019), where the CLTL performance of M-BERT between similar languages is decent, but that between distant languages is still far from satisfactory (Pires et al., 2019; Wu and Dredze, 2019; Karthikeyan et al., 2020).

From our point of view, CLTL can be analogized to the process of a human being learning a foreign language, where the prior knowledge on the native language plays an important role. Language educators believe that a foreign language learner can benefit a lot from translation, since translation not only involves all aspects of foreign language learning but also helps to enhance the correlation between the native language and the foreign language (Witte et al., 2009). According to our observation and experience, this is especially the case when the native language and the foreign language are distant from each other. Inspired by these thoughts, to facilitate CLTL, especially CLTL between distant languages, we propose a novel CLTL model named Translation Aided Language Learner (TALL), where CLTL is integrated with Machine Translation (MT). Specifically, we adopt a pre-trained multilingual language model, which is now recognized as the state of the art in CLTL, as our baseline model, and construct TALL by appending a decoder to it. On this basis, we directly fine-tune the baseline model as an NLU model to conduct CLTL, but put TALL through an MT-oriented pre-training before its NLU-oriented fine-tuning. We believe that the MT-oriented pre-training can help TALL to enhance the correlation between the given source-target language pair in its representation space, and thus can make CLTL easier to conduct in its NLU-oriented fine-tuning, especially in the case of distant languages. To make use of unannotated data, which is not only large in amount but also available for every language, we implement the recently proposed Unsupervised Machine Translation (UMT) (Artetxe et al., 2018b; Lample et al., 2018a; Yang et al., 2018; Lample et al., 2018b; Liu et al., 2020) technique in the MT-oriented pre-training of TALL.

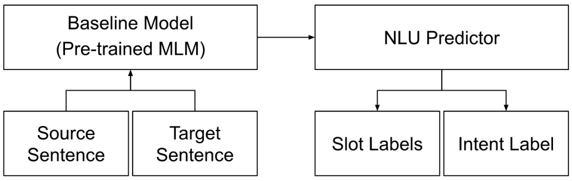

Figure 1: The NLU-oriented fine-tuning of our baseline model. MLM means multilingual language model.

To verify the effectiveness of TALL, we carry out a series of experiments to compare the CLTL performance of TALL with that of the baseline model. In these experiments, we address not only CLTL tasks between similar languages but also those between distant languages. For each given CLTL task, we separately use two popular pre-trained multilingual language models for model construction. To implement UMT, we collect unannotated sentences from Wikipedia dumps. To conduct CLTL, we separately collect annotated sentences from two multilingual NLU datasets. The experimental results show that the application of UMT enables TALL to consistently achieve better CLTL performance than the baseline model without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

2 Translation Aided Language Learner

2.1 Task Definition

NLU is a combo of slot filling and intent classification. Given a sentence x consisting of m words $\{w_1, \ldots, w_m\}$, slot filling is to predict a slot label y_i^σ for each word w_i, and intent classification is to predict an intent label y^ι for x. In this paper, NLU models are required to be trained under a zero-shot CLTL scenario, where annotated sentences in the given source language are used for model optimization, while those in the given target language are used for model evaluation.

2.2 Baseline Model

A transformer (Vaswani et al., 2017) is a sequence-to-sequence model consisting of an encoder and a decoder. A main feature of transformers is that they use multi-head self-attention and multi-head cross-attention to model dependencies in sequential data. These attention mechanisms enable transformers to extract long-term contextual clues from text. As a result, transformers have been intensively used in transfer learning to develop pre-trained language models, which generate contextualized word embeddings. For example, some pre-trained language models, such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), are implemented as transformer encoders, and some other ones, such as the GPT family (Radford et al., 2018, 2019), are implemented as transformer decoders.

As a sub-field of transfer learning, CLTL has witnessed the wide application of transformers in developing pre-trained multilingual language models. Most of the existing pre-trained multilingual language models, such as M-BERT, XLM (Conneau and Lample, 2019), and XLM-RoBERTa (XLM-R) (Conneau et al., 2020), are implemented as transformer encoders. Actually, these pre-trained multilingual language models are the multilingual variants of BERT and RoBERTa, since each of them is identical to either BERT or RoBERTa except being pre-trained on a multilingual corpus. The representation space learned through this pre-training is not only rich in contextual clues but also shared by all the involved languages. Therefore, in theory, each of these pre-trained multilingual language models can be simply fine-tuned to address any CLTL task between its involved languages.

The pre-trained multilingual language models mentioned above are now recognized as the state of the art in CLTL. To push the state of the art, we adopt one of them as our baseline model, and fine-tune it as an NLU model to conduct CLTL. As shown in Figure 1, in this NLU-oriented fine-tuning, we feed each given sentence to the baseline model, and feed the final hidden states of the baseline model to an NLU predictor. Since the baseline model is fitted with a sub-word tokenizer, a given sentence x consisting of m words $\{w_1, \ldots, w_m\}$ is tokenized into n tokens ($n \geqslant m$) such that the baseline model generates n final hidden states $\{h_1, \ldots, h_n\}$. For slot filling, the NLU predictor first performs an average pooling on the final hidden states related to each word w_i, and then uses a dense layer with a softmax normalization to map the pooling result to a slot distribution for w_i:

$$\mathrm{p}(y_i^\sigma \mid x) = \mathrm{softmax}\big(W^\sigma \mathrm{f_a}(h_{k_i}, \ldots, h_{l_i}) + b^\sigma\big)$$

where W^σ is a trainable weight, b^σ is a trainable bias, k_i and l_i separately represent the start position and end position of the final hidden states related to w_i, and $\mathrm{f_a}(\cdot)$ represents average pooling. For intent classification, the NLU predictor first performs an average pooling on all the final hidden states, and then uses another dense layer with another softmax normalization to map the pooling result to an intent distribution for x:

$$\mathrm{p}(y^\iota \mid x) = \mathrm{softmax}\big(W^\iota \mathrm{f_a}(h_1, \ldots, h_n) + b^\iota\big)$$

where W^ι is a trainable weight, and b^ι is a trainable bias. On this basis, for model optimization, we minimize the following joint loss through stochastic gradient descent on annotated sentences in the given source language:

$$\mathcal{L}_{nlu} = -\log\Big(\prod_{i=1}^{m} \mathrm{p}(y_i^\sigma \mid x) \cdot \mathrm{p}(y^\iota \mid x)\Big)$$

For model evaluation, we infer the baseline model on annotated sentences in the given target language to measure three evaluation metrics, namely Slot F1, Intent Accuracy, and Semantic Accuracy (i.e. sentence-level joint accuracy).

2.3 Proposed Model

Since the baseline model is pre-trained on a multilingual corpus, all its involved languages are correlated with each other in its representation space. Normally, the larger such correlation between languages, the easier it is to conduct CLTL. To equally treat all possible CLTL tasks, the multilingual corpus used in the pre-training of the baseline model is collected in a subtle way that is not obviously biased to any language. However, there are two side effects of doing so. On the one hand, instead of focusing on a specific CLTL task, the baseline model pays equal attention to all possible CLTL tasks. On the other hand, in the representation space of the baseline model, the correlation between languages is proportional to their linguistic similarity, or in other words, similar languages are still similar to each other, and distant languages are still distant from each other. This implies that the CLTL ability of the baseline model can be pertinently improved for each given CLTL task, and the room for improvement is relatively large when the CLTL task is between distant languages.

To pertinently improve the CLTL ability of the baseline model for each given CLTL task, we would like to transform its representation space, which is used for all possible CLTL tasks, into a specialized one, where the correlation between the given source-target language pair is expressly enhanced. This goal can be achieved by resorting to MT, since

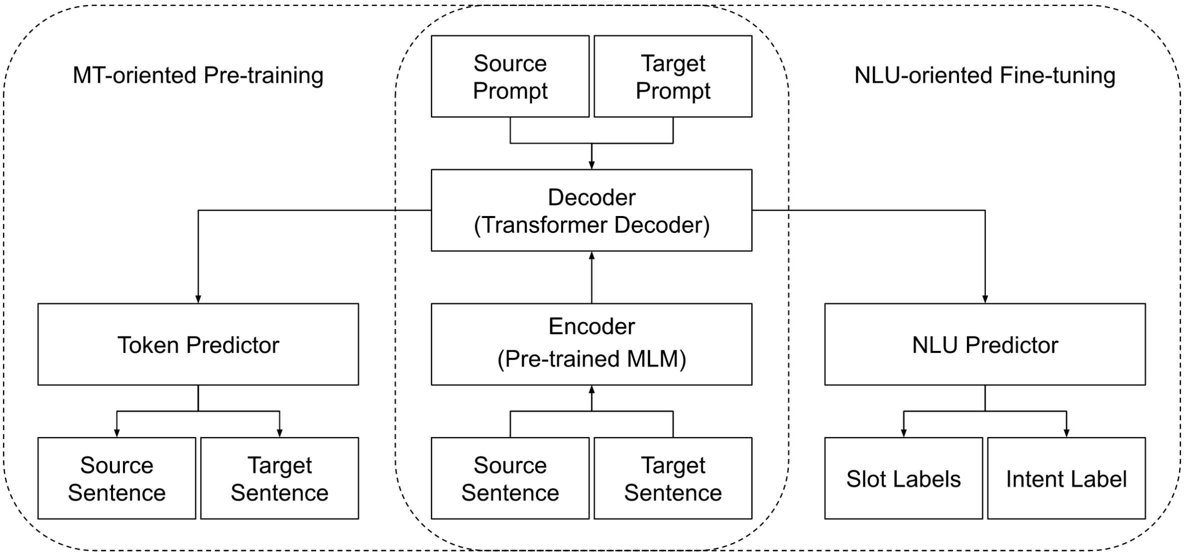

Figure 2: The MT-oriented pre-training and NLU-oriented fine-tuning of our proposed Translation Aided Language Learner (TALL).

translation is the most direct way to correlate languages with each other. As shown in Figure 2, for MT to be workable, we treat the baseline model as an encoder and append a decoder to it. Considering that the encoder is implemented as a transformer encoder, we implement the decoder as a transformer decoder to keep the model architecture consistent. Besides, as in Vaswani et al. (2017), we also share the token embeddings between the encoder and the decoder. The resulting new model can be seen as a standard transformer, where the encoder is a pre-trained multilingual language model. We expect this model to learn the correlation between the given source-target language pair by addressing a two-way MT task, and thus name it Translation Aided Language Learner (TALL).

Before conducting CLTL with TALL, we need to pre-train it as a two-way MT model that translates between the given source-target language pair. As shown in Figure 2, in this MT-oriented pre-training, we feed each given sentence to the encoder, feed a prompt for the translated sentence to the decoder, and feed the final hidden states of the decoder to a token predictor. Given a sentence x and a prompt x' for the translated sentence, suppose the decoder generates a final hidden state h'_i for the i-th token in x', then h'_i can be seen as a memory of both x and the first i tokens in x'. The token predictor uses a dense layer with a softmax normalization to map this memory to a token distribution for the position i in the translated sentence:

$$\mathrm{p}(y_i^\tau \mid x, x') = \mathrm{softmax}(W^\tau h'_i + b^\tau)$$

where W^τ is a trainable weight tied to the token embeddings, and b^τ is a trainable bias. Since two-way MT requires the translated sentence to be in either the source language or the target language, which depends on the current direction, we extend the token vocabulary with two language identifiers, which separately represent the two languages, and thereby inform the decoder about the currently required language by setting the first token of x' to the corresponding language identifier. By the way, since the token vocabulary is highly multilingual, most probabilities in the above token distribution are for the tokens beyond the given source-target language pair and thus make no sense. Therefore, we ignore these probabilities when inferring TALL to generate translated sentences.

By convention, the training of MT models is supervised and thus requires parallel corpora. However, parallel corpora are generally expensive to collect, which makes them scarce or even unavailable for many source-target language pairs. Since TALL is designed to be a general-purpose CLTL model, a supervised training on parallel corpora is not applicable to its MT-oriented pre-training. Recently, an unsupervised training technique for MT models, which is named Unsupervised Machine Translation (UMT), has been proposed. Instead of relying on parallel corpora, UMT relies on monolingual corpora of unannotated sentences. This is attractive to us, since a large amount of unannotated sentences are always available for every language. Therefore, we implement the UMT training recipe proposed by Lampe et al. (2018b) in the MT-oriented pre-training of TALL. Specifically, for model optimization, we collect a source-language corpus S and a target-language corpus T, each of which is a set of unannotated sentences. On this basis, we measure the following two losses:

- **Denoising auto-encoding loss.** As in Lample et al. (2018a), we implement a noise injector $\mathrm{f_n}(\cdot)$, which injects noise to each given sentence by randomly dropping and swapping its tokens. For each source-language sentence $s \in S$, we first run the noise injector to obtain a noise-injected sentence $\mathrm{f_n}(s)$, which can be seen as a sentence in a different language, and then use TALL to translate $\mathrm{f_n}(s)$ to the source language, the expected result of which is s. Besides, we also perform this process on each target-language sentence $t \in T$. This is the so-called "denoising auto-encoding", whose loss is defined as the cross-entropy loss on recovering the original sentences from the noise-injected sentences:

$$\begin{aligned}\mathcal{L}_{dae} =& \mathbb{E}_{s\in S}\big[-\log \mathrm{p}\big(s \mid \mathrm{f_n}(s)\big)\big]+\\ & \mathbb{E}_{t\in T}\big[-\log \mathrm{p}\big(t \mid \mathrm{f_n}(t)\big)\big]\end{aligned}$$

- **Back-translation loss.** Let us use $\mathrm{f_m}(\cdot)$ to represent the inference of TALL, which translates each given sentence to its opposite language in the given source-target language pair. For each source-language sentence $s \in S$, we first infer TALL to obtain a TALL-translated sentence $\mathrm{f_m}(s)$, which is in the target language, and then use TALL to translate $\mathrm{f_m}(s)$ to the source language, the expected result of which is s. Besides, we also perform this process on each target-language sentence $t \in T$. This is the so-called "back-translation", whose loss is defined as the cross-entropy loss on recovering the original sentences from the TALL-translated sentences:

$$\begin{aligned}\mathcal{L}_{bt} =& \mathbb{E}_{s\in S}\big[-\log \mathrm{p}\big(s \mid \mathrm{f_m}(s)\big)\big]+\\ & \mathbb{E}_{t\in T}\big[-\log \mathrm{p}\big(t \mid \mathrm{f_m}(t)\big)\big]\end{aligned}$$

We sum up the above two losses to obtain a joint loss, and thereby minimize the joint loss through stochastic gradient descent. For model evaluation, we collect another source-language corpus and another target-language corpus, each of which is also a set of unannotated sentences. On this basis, we implement the round-trip translation trick proposed by Lample et al. (2018a), where we first translate each given sentence to its opposite language in the current source-target language pair, and then translate the resulting sentence to the original language. By inferring TALL, we perform this process on all the sentences in the above two corpora to obtain two reconstructed corpora. Thereby, we measure the BLEU score between the two original corpora and the two reconstructed corpora to evaluate the translation performance of TALL.

The above MT-oriented pre-training guarantees that TALL can learn a representation space, where the given source-target language pair are expressly correlated with each other. As a result, it will be easier to conduct CLTL with the pre-trained TALL than with the baseline model. This is especially the case when the given source-target language pair are distant from each other, since translating between distant languages reveals more knowledge than translating between similar languages. However, considering that the representation space of TALL is co-carried by the encoder and the decoder, we have to fine-tune them together as an NLU model when we conduct CLTL with the pre-trained TALL. To this end, we implement the fine-tuning approach of BART (Lewis et al., 2020) in the NLU-oriented fine-tuning of TALL. Specifically, as shown in Figure 2, we feed each given sentence to the encoder, feed this sentence again as a prompt to the decoder with the corresponding language identifier prefixed to it, and feed the final hidden states of the decoder except the last one to the NLU predictor. On this basis, both the model optimization and the model evaluation remain the same as in the NLU-oriented fine-tuning of the baseline model.

3 Related Works

Cross-lingual word embeddings. A traditional way to conduct CLTL is to leverage cross-lingual word embeddings, which are usually learned in an unsupervised manner. For example, Zhang et al. (2017) formulate the learning of cross-lingual word embeddings as an adversarial game, and explore several adversarial training methods to implement it. Conneau et al. (2017) first use adversarial training to learn a linear mapping from the word embeddings of a source language to those of a target language, and then use a Procrustes solution to refine it. Artetxe et al. (2018a) first use an unsupervised initialization scheme to create an initial mapping, and then use a self-learning procedure to iteratively improve it. Chen et al. (2018) propose a language-adversarial training method, and use it to address cross-lingual sentiment classification. Besides, there are also several studies on multilingual word embeddings. For example, Chen and Cardie (2018) propose an unsupervised approach to learning multilingual word embeddings, which directly exploits the relations between all the involved languages. On this basis, Chen et al. (2019) propose a multi-source CLTL model, which not only uses adversarial training to learn language-invariant features, but also uses a mixture-of-experts method to dynamically exploit the similarity between a target language and multiple source languages.

Pre-trained multilingual language models. The currently dominant way to conduct CLTL is to fine-tune pre-trained multilingual language models, which are multilingual variants of pre-trained language models, and are each pre-trained on a multilingual corpus. For example, Mulcaire et al. (2019) propose Rosita as a multilingual variant of ELMo, and pre-train it on a multilingual corpus covering 3 languages. Devlin et al. (2019) propose M-BERT as a multilingual variant of BERT, and pre-train it on a multilingual corpus covering 104 languages. Conneau and Lample (2019) propose XLM as a multilingual variant of BERT, and pre-train it on a multilingual corpus covering 15 languages. Conneau et al. (2020) propose XLM-R as a multilingual variant of RoBERTa, and pre-train it on a multilingual corpus covering 100 languages. Besides, there are also several empirical studies on M-BERT. For example, Pires et al. (2019) carry out a large number of probing experiments to verify and interpret the zero-shot CLTL performance of M-BERT. Wu and Dredze (2019) explore the zero-shot CLTL potential of M-BERT on 5 downstream tasks covering 39 languages. Karthikeyan et al. (2020) provide a comprehensive study on the contribution of each component of M-BERT to its CLTL ability, which focuses on the impact of linguistic properties of the languages, the architecture of the model, and the learning objectives.

UMT technique. The UMT technique is aimed at reducing the reliance of MT models on parallel corpora. For example, Artetxe et al. (2018b) construct an MT model consisting of a language-invariant encoder and two language-specific decoders, and train it on a non-parallel corpus through denoising auto-encoding and back-translation. Lample et al. (2018a) construct an MT model consisting of a language-invariant pair of encoder and decoder, and train it on a non-parallel corpus not only through denoising auto-encoding and back-translation but also through adversarial training. Yang et al. (2018) construct an MT model consisting of two pairs of encoder and decoder, which partially share their parameters, and train it on a non-parallel corpus not only through denoising auto-encoding and back-translation but also through adversarial training. Lample et al. (2018b) propose a simple but effective approach based on the above works, where the constructed MT model only consists of a language-invariant pair of encoder and decoder, and its training on a non-parallel corpus only requires denoising auto-encoding and back-translation. Liu et al. (2020) first pre-train BART on a non-parallel multilingual corpus through denoising auto-encoding, and then fine-tune the pre-trained BART for downstream MT tasks.

4 Verification Experiments

4.1 Experimental Settings

CLTL tasks. For generality, we address not only CLTL tasks between distant languages but also those between similar languages. Specifically, we separately conduct CLTL between three source-target language pairs, which include two distant language pairs, namely English-Japanese and German-Japanese, and one similar language pair, namely English-German.

Pre-trained multilingual language models. For compatibility, we use different pre-trained multilingual language models for model construction. Specifically, for each given CLTL task, we separately use two popular pre-trained multilingual language models, namely M-BERT (base and cased) and XLM-R (base), to construct both the baseline model and TALL.

Training data. For practicality, we adopt large-scale corpora and real-world datasets as training data. Specifically, to implement UMT, we collect a source-language corpus of 1M unannotated sentences and a target-language corpus of 1M unannotated sentences from Wikipedia dumps for model optimization, and also collect a source-language corpus of 10K unannotated sentences and a target-language corpus of 10K unannotated sentences from Wikipedia dumps for model evaluation. To conduct CLTL, we collect annotated sentences in real-world domains from two multilingual NLU

CLTL Task	Pre-trained MLM	BLEU Score	Slot F1			Intent Accuracy			Semantic Accuracy		
			BSL	TALL	Gain	BSL	TALL	Gain	BSL	TALL	Gain
EN-JA (distant)	M-BERT	41.19	56.78	60.87	7.20%	80.37	83.21	3.53%	14.56	16.39	12.57%
	XLM-R	39.83	58.21	63.19	8.56%	81.19	83.92	3.36%	16.58	18.47	11.40%
DE-JA (distant)	M-BERT	38.54	51.28	54.56	6.40%	79.08	81.54	3.11%	11.71	13.24	13.07%
	XLM-R	35.11	50.36	53.68	6.59%	78.43	81.12	3.43%	12.76	14.61	14.50%
EN-DE (similar)	M-BERT	71.21	70.42	72.39	2.80%	89.39	91.16	1.98%	36.53	38.64	5.78%
	XLM-R	72.91	75.29	77.14	2.46%	92.82	94.33	1.63%	44.86	47.25	5.33%

Table 1: The translation performance of TALL on Wikipedia and the CLTL performance of both the baseline model and TALL on MultiATIS++. EN means English. JA means Japanese. DE means German. BSL means the baseline model. Gain means the CLTL performance gain of TALL over the baseline model. The gain numbers are in percentage and calculated as $(\text{TALL} - \text{BSL}) \div \text{BSL}$.

CLTL Task	Pre-trained MLM	Slot F1 Gain	Intent Accuracy Gain	Semantic Accuracy Gain
EN-JA (distant)	M-BERT	53.61%	36.75%	59.45%
	XLM-R	50.96%	31.60%	71.86%
DE-JA (distant)	M-BERT	47.75%	31.46%	55.55%
	XLM-R	58.49%	34.45%	69.19%
EN-DE (similar)	M-BERT	10.42%	9.69%	18.00%
	XLM-R	12.75%	7.81%	23.87%

Table 2: The CLTL performance gain of TALL over the baseline model on the multi-domain multilingual NLU dataset.

datasets. The first multilingual NLU dataset is MultiATIS++ (Xu et al., 2020), which is an extension to Multilingual ATIS (Upadhyay et al., 2018). It provides 5K annotated sentences for each language, which are all in the domain of airline travel. The second multilingual NLU dataset is a multi-domain dataset collected from a virtual assistant. It provides 100K annotated sentences for each language, which are evenly distributed in five domains, namely music, notifications, smart home, weather, and books. By the way, in the above two multilingual NLU datasets, each word is annotated with a slot label in the B-I-O format, and each sentence is annotated with an intent label.

4.2 Implementation details.

We use WikiExtractor (Attardi, 2015) to extract paragraphs from Wikipedia dumps, use Stanza (Qi et al., 2020) to split paragraphs into sentences, use HuggingFace's Transformers (Wolf et al., 2019) to tokenize sentences into tokens and load pre-trained multilingual language models, and use PyTorch (Paszke et al., 2019) to implement both the baseline model and TALL. For model optimization, we apply an AdamW optimizer (Loshchilov and Hutter, 2019) with an initial learning rate of 0.0001, a weight decay factor of 0.01, and a batch size of 64 in the MT-oriented pre-training of TALL, and apply another AdamW optimizer with an initial learning rate of 0.00005, a weight decay factor of 0.01, and a batch size of 256 in the NLU-oriented fine-tuning of both the baseline model and TALL. After each epoch, we evaluate the validation performance, which refers to BLEU score in the MT-oriented pre-training of TALL and Semantic Accuracy in the NLU-oriented fine-tuning of both the baseline model and TALL. If the obtained performance number is improved, we save the model, otherwise we cancel the finished epoch by restoring the model to the last saved version. We decay the learning rate by 0.5 after each cancelled epoch, and terminate the model optimization after the 5th cancelled epoch. For model evaluation, we use NLTK (Loper and Bird, 2004) to measure BLEU score, and use the evaluation script for the CoNLL-2000 shared task to measure Slot F1.

4.3 Experimental Results

As shown in Table 1, we carry out a series of experiments on the unannotated sentences collected from Wikipedia and the annotated sentences collected from MultiATIS++. Each of these experiments is aimed at a different combination of CLTL task and pre-trained multilingual language model, and includes the corresponding MT-oriented pre-training of TALL and the corresponding NLU-oriented fine-tuning of both the baseline model and TALL. On this basis, we first evaluate the translation performance of TALL in its MT-oriented pre-training, then evaluate the CLTL performance of both the baseline model and TALL in their NLU-oriented fine-tuning, and finally calculate the CLTL performance gain of TALL over the baseline model in percentage. Besides, as shown in Table 2, we also repeat the NLU-oriented fine-tuning of both the baseline model and TALL on the annotated sentences collected from the multi-domain multilingual NLU dataset, and thereby obtain another CLTL performance gain of TALL over the baseline model. The experimental results show that due to the application of UMT in the MT-oriented pre-training, TALL consistently achieves better CLTL performance than the baseline model in the NLU-oriented fine-tuning without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

4.4 Ablation Study

Denoising auto-encoding. In the MT-oriented pre-training of TALL, we try to discard the denoising auto-encoding loss and only minimize the back-translation loss in the UMT training. As a result, we observe that TALL achieves a very poor translation performance and a very poor CLTL performance. This implies that TALL learns little cross-lingual knowledge through the UMT training without denoising auto-encoding.

Back-translation. In the MT-oriented pre-training of TALL, we also try to discard the back-translation loss and only minimize the denoising auto-encoding loss in the UMT training. As a result, we observe that TALL achieves an almost perfect translation performance but a very poor CLTL performance. This is because the UMT training without back-translation makes TALL a copying model instead of an MT model, and a copying model can work perfectly in the model evaluation based on round-trip translation.

BART-style fine-tuning. In the NLU-oriented fine-tuning of TALL, instead of following the fine-tuning approach of BART, we try to discard the decoder and only fine-tune the encoder following the way we fine-tune the baseline model. As a result, we observe a very poor CLTL performance. This implies that the decoder of TALL is necessary for its NLU-oriented fine-tuning.

5 Further Discussion

Is a startup supervision necessary for the back-translation? In several existing UMT training recipes, the back-translation is supervised during its startup stage, where the supervision is provided by replacing the inference of TALL with a bilingual dictionary (Lample et al., 2018a; Artetxe et al., 2018b). This startup supervision is aimed at initializing a shared representation space for the given source-target language pair. However, since the encoder of TALL is a pre-trained multilingual language model, TALL already possesses a properly initialized representation space, which is shared by all the involved languages, and thus does not need a startup supervision. Actually, we tried to use a parallel corpus generated by a naive MT model to provide a startup supervision, which is equivalent to using a bilingual dictionary, but did not observe any translation performance gain.

How does the UMT training affect the CLTL performance? The UMT training uses the denoising auto-encoding and the back-translation to enhance the correlation between the given source-target language pair in the representation space of TALL. Since the encoder of TALL is a pre-trained multilingual language model, the representation space of TALL can be seen as an extension to that of the pre-trained multilingual language model. In the representation space of the pre-trained multilingual language model, similar languages have been more correlated with each other than distant languages. That is to say, in the representation space of TALL, there is more potential to enhanced the correlation between distant languages than between similar languages. As a result, although the CLTL performance between similar languages is better than that between distant languages, the CLTL performance gain between distant languages is larger than that between similar languages.

6 Conclusion

The contribution of this paper is three-fold. First of all, we construct a novel CLTL model TALL based on a pre-trained multilingual language model. In the next place, we train TALL to conduct CLTL through an MT-oriented pre-training and an NLU-oriented fine-tuning. Last but not least, we implement UMT in the MT-oriented pre-training of TALL to make use of unannotated data. Compared with the baseline model, which is the pre-trained multilingual language model used to construct TALL, TALL consistently achieves better CLTL performance without using more annotated data, and the performance gain is relatively prominent in the case of distant languages. In the future, we will collect unannotated corpora that are linguistically compatible with the downstream NLU tasks for the UMT training, which we believe can further boost the CLTL performance of TALL.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Giusepppe Attardi. 2015. Wikiextractor.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

K Karthikeyan, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual bert: An empirical study. In *International Conference on Learning Representations*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Edward Loper and Steven Bird. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Phoebe Mulcaire, Jungo Kasai, and Noah A Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Tjeerd van der Ploeg, Peter C Austin, and Ewout W Steyerberg. 2014. Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints. *BMC medical research methodology*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.

Arnd Witte, Theo Harden, and Alessandra Ramos de Oliveira Harden. 2009. *Translation in Second Language Learning and Teaching*, volume 3. Peter Lang.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual nlu. *arXiv preprint arXiv:2004.14353*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Pipeline Signed Japanese Translation Focusing on a Post-positional Particle Complement and Conjugation in a Low-resource Setting

Ken Yano and **Akira Utsumi**
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, JAPAN
`yanoken@uec.ac.jp, utsumi@uec.ac.jp`

Abstract

Because sign language is a visual language, the translation of it into spoken language is typically performed through an intermediate representation called gloss notation. In sign language, function words, such as particles and determiners, are not explicitly expressed, and there is little or no concept of morphological inflection in sign language. Therefore, gloss notation does not include such linguistic constructs. Because of these factors, we argue that sign language translation is effectively processed by taking advantage of the similarities and differences between sign language and its spoken counterpart. We thus propose a pipeline translation method that clearly focuses on the difference between spoken Japanese and signed Japanese written in gloss notation. Specifically, our method first uses statistical machine translation (SMT) to map glosses to corresponding spoken language words. We then use three transformer-based seq2seq models trained using a large out-of-domain monolingual Japanese corpus to complement postpositional particles and estimate conjugations for the verbs, adjectives, and auxiliary verbs in the first translation. We apply the seq2seq models in sequence until the translation converges. Our experimental results show that the proposed method performs robustly on the low-resource corpus and is +4.4/+4.9 points above the SMT baseline for BLEU-3/4.

1 Introduction

It is essential to build a social infrastructure for hearing-impaired and hearing communities to share sufficient information so that they can quickly obtain information necessary for daily life and disasters and lead a safe and secure life. Sign language used in deaf communities has different vocabulary and grammar from spoken language. There are two variations of sign language in Japan (Chonan, 2001): (1) Japanese Sign Language (JSL) and (2) Manually Coded Japanese (MCJ). JSL is often used by early signers, and syntax, such as word order and language structure, is different from spoken Japanese. By contrast, the syntax of MCJ is similar to that of spoken Japanese in terms of word order. It is used by late signers or acquired hearing-impaired people. However, the two variations are said to be used interchangeably and there is no clear boundary between them. In this work, we consider an intermediate between JSL and MCJ, and denote it Signed Japanese (SJ) in the following discussion.

Translation from sign language to spoken language is typically performed in two steps. First, consecutive signs are recognized from a video signal and transformed into an intermediate representation called a *gloss*, then the gloss is translated into a sentence in spoken language. Current state-of-the-art sign language recognition and translation methods (Camgöz et al. 2020; Yin and Read 2020) require a large amount of data and pay little attention to differences between sign language and the corresponding spoken language. Therefore, the success of these approaches relies heavily on large paired corpora, and resource-poor sign language studies, including SJ, cannot take advantage of such approaches. In sign language, function words, such as pre-positional or post-positional particles and determiners, do not tend to be explicitly signed, and inflectional morphemes associated with verbal predicates that express categories, such as tense, mood, and aspect, are not manually signed, in general. For example, in SJ, post-positional particles ‘助詞’ are generally not explicitly signed, and the signs associated with verbal predicates are not conjugated, whereas in Japanese, verbs, adjectives, and auxiliary verbs are conjugated. Therefore, the gloss does not include such language constructs.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2021–2032
August 1–6, 2021.

Although gloss notation is commonly used by writing a series of spoken words that correspond to each sign in capital letters, because of the lack of sign language resources, its quality and size differ greatly according to the language (Bungeroth et al., 2008). SJ signs are heavily polysemous and their meaning is often context sensitive. Additionally, there is no publicly available corpus for SJ translation studies. Therefore, in this study, we use an in-house corpus that uses our gloss notation method. The details of the corpus and its notation are described in Section 2.

To solve challenging problems, we propose a novel pipeline method to translate from SJ written in gloss to Japanese. In particular, we focus on the linguistic differences between SJ and Japanese and estimate the post-positional particles that are missing and the appropriate forms of morphological inflection of words. Our method assumes that the ground truth gloss of the signed sentence is available. This assumption does not limit the availability of the above two-step sign language translation method. Our method first uses phrase-based statistical machine translation to match the SJ gloss to Japanese words. Then we refine the results further using transformer-based seq2seq (Sutskever et al., 2014) models, which are trained using a large out-of-domain parallel corpus. Specifically, we use three different seq2seq models (1) to complement the post-positional particles, (2) to apply morphological inflection by conjugating verbs, adjectives, and auxiliary verbs, and (3) to re-estimate the post-positional particles over the previous output. We repeatedly apply these models sequentially and adjust the translation results until they converge.

The proposed method works robustly, even for small training datasets, which are typically of the order of thousands of pairs in a dataset, and the results show that the state-of-the-art method is inferior to the SMT baseline with the low-resource setting. We found that iterative updates of translations are effective for improving the grammaticality and fluency of the translation output. Our experimental results show that the proposed model provides +4.4/+4.9 higher translation performance for BLEU3/BLEU4 scores compared with the SMT baseline.

2 Materials

We use two corpora: one is a small in-house SJ and Japanese parallel corpus, and the other is a large out-of-domain Japanese monolingual corpus. We describe the details of each corpus as follows.

2.1 SJ and Japanese parallel corpus

The locally organized in-house parallel corpus contains 1,086 sentence pairs with $>$7.5K glosses from a vocabulary of 655 words, and $>$11K Japanese words from a vocabulary of $>$1.2K words. The average length of a gloss sentence is 6.9 words, with a maximum length of 12 words and minimum length of 2 words, and the average length of a Japanese sentence is 10.3 words, with a maximum length of 21 words and minimum length of 5 words. The corpus consists of the ground truth gloss transcriptions of signs and their translations to Japanese sentences. The sentences are various spontaneous conversations that seemingly took place at municipal offices, such as asking for a certified copy of the resident register, pension, and unemployment insurance. In the corpus, a gloss word is written in the form gN, where N corresponds to an arbitrary unique number. We adopt this notation instead of using Japanese words because glosses in SJ are heavily polysemous and a sign maps to different Japanese words depending on the context. Instead, we use an auxiliary dictionary to map each gloss to spoken words or phrases. This notation method also helps the proposed method to select the appropriate Japanese word or phrase within the phrase-based statistical machine translation model that we use in the study.

Because of the sparsity of the parallel corpus, approximately 2.3% of the glosses are singletons, so we add all gloss dictionary items as additional parallel data to reduce OOV issues at test time.

2.2 Out-of-domain Japanese corpus

We use a subset of the Balanced Corpus of Contemporary Written Japanese [1] as an out-of-domain Japanese corpus to manually generate pseudo parallel corpora. The details of the corpus generation procedure are described in Section 3. To select the subsets, we use pattern matching to select sentences that end in a pattern such as [∼か？(question), ∼しました。(admit), ∼ほしいで

---

[1] https://pj.ninjal.ac.jp/corpus_center/bccwj/en/

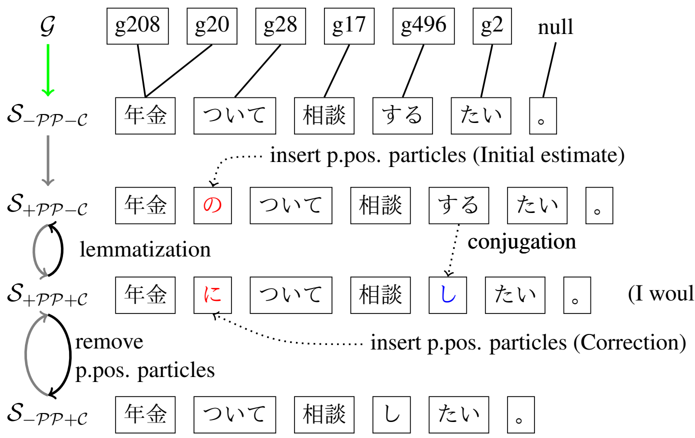

Gloss	Definition
g208	年, ため “year,for”
g20	費, お金, 費用 “expense,money”
g28	だから, ついて “about,because”
g17	相談, 会議 “ask,meeting”
g496	する, やる “do”
g2	したい, 欲しい, 好き “want,like”

Figure 1: Overall pipeline translation architecture from a gloss sequence $\mathcal{G}$ to a grammatical Japanese sequence $\mathcal{S}_{+PP+C}$. The green and gray arrows indicate the translations by a statistical machine translation model and three neural machine translation models, respectively. The table in the upper right corner shows an excerpt from the gloss dictionary. (p.pos. = post-positional)

す。(ask), ～います。(confirm), ～します。(intend), ～希望する。(desire)]. These patterns were chosen so that the selected sentences are similar to the target Japanese in the paired corpus. The monolingual corpus contains >195K sentences with >3.9M Japanese words from a vocabulary of >70K Japanese words.

3 Methodology

The overall proposed pipeline translation system is shown in Fig. 1. $\mathcal{G}$ represents a gloss sequence and $\mathcal{S}$ represents a Japanese sequence. We define two subscripts for $\mathcal{S}$, that is, PP, which denotes ‘post-positional particle,’ and C, which denotes ‘conjugation,’ with the prefix $+$ or $-$ for each subscript, which denotes the existence or non-existence, respectively, of post-positional particles and conjugation. The definition of each term is provided in Table 1.

	post-positional particles	conjugation
$\mathcal{S}_{-PP-C}$		
$\mathcal{S}_{+PP-C}$	✓	
$\mathcal{S}_{-PP+C}$		✓
$\mathcal{S}_{+PP+C}$	✓	✓

Table 1: Definitions of terms for the variants of $\mathcal{S}$

3.1 Translation method

The proposed pipeline translation consists of six steps. Algorithm 1 shows the steps applied in sequence to gradually convert from $\mathcal{G}$ to $\mathcal{S}_{+PP+C}$, which is the final translation of this algorithm. The details of each step are as follows:

Step 0: Translate $\mathcal{G}$ into $\mathcal{S}_{-PP-C}$

We use a phrase-based statistical machine translation (pbsmt) (Koehn et al., 2007) to translate $\mathcal{G}$ into $\mathcal{S}_{-PP-C}$. In this step, we map each gloss phrase to the appropriate Japanese phrases without considering the post-positional particles and conjugations of the output Japanese sequence.

Step 1: Translate $\mathcal{S}_{-PP-C}$ into $\mathcal{S}_{+PP-C}$

We use a transformer-based seq2seq model (Vaswani et al., 2017) (s2s_m1) to translate $\mathcal{S}_{-PP-C}$ into $\mathcal{S}_{+PP-C}$. In this step, the model estimates missing post-positional particles in $\mathcal{S}_{-PP-C}$ and inserts them to generate $\mathcal{S}_{+PP-C}$.

Step 2: Translate $\mathcal{S}_{+PP-C}$ into $\mathcal{S}_{+PP+C}$

We use another transformer-based seq2seq model (s2s_m2) to translate $\mathcal{S}_{+PP-C}$ into $\mathcal{S}_{+PP+C}$. In this step, the model estimates the appropriate morphological inflection or conjugated form for verbs, adjectives, and auxiliary verbs in $\mathcal{S}_{+PP-C}$ to generate $\mathcal{S}_{+PP+C}$.

Step 3: Convert $\mathcal{S}_{+PP+C}$ to $\mathcal{S}_{-PP+C}$

In this step, we remove the previously estimated post-positional particles in $\mathcal{S}_{+PP+C}$ from Step 2 to generate $\mathcal{S}_{-PP+C}$.

Step 4: Translate $\mathcal{S}_{-PP+C}$ into $\mathcal{S}_{+PP+C}$

We use the other transformer-based seq2seq model (s2s_m3) to translate $\mathcal{S}_{-PP+C}$ into $\mathcal{S}_{+PP+C}$ by re-estimating missing post-positional particles. In Step 1, we estimated the post-positional particles over a Japanese sequence in the canonical form ($\mathcal{S}_{-PP-C}$); however, in the present step, we estimate them over for the conjugated word sequence ($\mathcal{S}_{-PP+C}$). We assume that this will correct the previous estimation using the conjugated form of the word sequence obtained in the previous steps.

Step 5: Convert $\mathcal{S}_{+PP+C}$ to $\mathcal{S}_{+PP-C}$

In this step, we transform $\mathcal{S}_{+PP+C}$ to $\mathcal{S}_{+PP-C}$ by converting words in $\mathcal{S}_{+PP+C}$ to their canonical form. Steps 3, 4, 5, and 2 are repeated until the translation output converges or the number of iterations reaches the maximum limit (10).

Algorithm 1 Translation of a sign gloss sequence into a Japanese sentence

```
Input: $\mathcal{G}$
Output: $\mathcal{S}_{+PP+C}$
  Step 0: $\mathcal{G} \rightarrow \mathcal{S}_{-PP-C}$
  Step 1: $\mathcal{S}_{-PP-C} \rightarrow \mathcal{S}_{+PP-C}$
  Step 2: $\mathcal{S}_{+PP-C} \rightarrow \mathcal{S}_{+PP+C}$
  $\mathcal{S}_{prev} = \emptyset$
  $\mathcal{S}_{next} = \mathcal{S}_{+PP+C}$
  while $\mathcal{S}_{prev} \neq \mathcal{S}_{next}$ do
      $\mathcal{S}_{prev} = \mathcal{S}_{next}$
      Step 3: $\mathcal{S}_{+PP+C} \rightarrow \mathcal{S}_{-PP+C}$
      Step 4: $\mathcal{S}_{-PP+C} \rightarrow \mathcal{S}_{+PP+C}$
      Step 5: $\mathcal{S}_{+PP+C} \rightarrow \mathcal{S}_{+PP-C}$
      Step 2: $\mathcal{S}_{+PP-C} \rightarrow \mathcal{S}_{+PP+C}$
      $\mathcal{S}_{next} = \mathcal{S}_{+PP+C}$
  end while
  return $\mathcal{S}_{next}$
```

4 Training

4.1 Statistical machine translation model in Step 0

We use Moses (Koehn et al., 2007) to train the phrase-based statistical machine translation model to translate from $\mathcal{G}$ to $\mathcal{S}_{-PP-C}$ in Step 0. To train the model, we use the parallel corpus and preprocess the target Japanese sequences by deleting post-positional particles, and convert conjugated words, such as verbs, adjectives, and auxiliary verbs, to their canonical forms using MeCab [2].

Note that we leave any post-positional particles untouched if gloss words corresponding to them exist. For example, the Japanese word か *ka* which is a post-positional particle bound to the end of an interrogative sentence, has a corresponding gloss word in SJ. The translation model is described by the following noisy-channel model to estimate the best target Japanese word sentence $s \in \mathcal{S}_{-PP-C}$ for a source gloss sentence $g \in \mathcal{G}$ as

$$s_{best} = argmax_s \; p(g|s)p_{LM}(s), \qquad (1)$$

where $p_{LM}(s)$ is a language model based on the n-grams of $\mathcal{S}_{-PP-C}$. $p(g|s)$ is decomposed into a phrase-based formula using a phrase translation table and phrase reordering model (Koehn et al., 2007). For the language model, we use modified 3-gram Kneser–Ney smoothing.

4.2 Encoder-decoder translation models in Steps 1, 2, and 4

For the seq2seq models used in Steps 1, 2, and 4 in Section 3.1, we use a transformer-based encoder-decoder model (Ott et al., 2019). For all models, we use an encoder and decoder with an embedding of size 512, FFN-embedding of size 2048, and six layers with eight attention heads. We use the Adam optimizer with label smoothing cross-entropy loss with a smoothing factor of 0.1. We set the initial learning rate to 5e-4 with a warmup updates of 4,000 and use the inverse sqrt learning rate scheduler. We set the maximum tokens in a batch to 4K. We use tied embedding for the input and output layers. We obtain the hyperparameters from a non-exhaustive parameter search, and the results are shown in Table 9 of the Appendix.

We randomly split the corpus into training, validation, and test sets in an 8:1:1 ratio. For tokenization, we use byte pair encoding (Sennrich et al., 2016), which is trained using the training dataset. We set the number of operations to 10K for each tokenization of the seq2seq models. For the seq2seq model in Step 1, we pre-process the out-of-domain monolingual corpus as follows:

- Source: preprocess Japanese sequences by deleting post-positional particles and converting all conjugated words, such as verbs, adjectives, and auxiliary verbs, to their canonical forms.
- Target: preprocess Japanese sequences by leaving post-positional particles untouched and converting all conjugated words, such as verbs, adjectives, and auxiliary verbs, to their canonical forms.

The pre-processed corpus becomes the pseudo-parallel corpus to train the seq2seq model in Step

---

[2] https://pypi.org/project/mecab-python3/

1 to translate $\mathcal{S}_{-PP-C}$ into $\mathcal{S}_{+PP-C}$. The training corpora of the other seq2seq models, that is, s2s_m2 in Step 2 and s2s_m3 in Step 4, are similarly preprocessed and independently trained using the pseudo-parallel corpus.

We observed that training the seq2seq model in Step 2 took more than a few hundred epochs, whereas training the seq2seq model in Steps 1 and 4 took less than 30 epochs. We used the models with the lowest validation loss for the experiments. The results on the test set demonstrated that the BLEU4 scores were 74.20, 98.75, and 75.06 for s2s_m1, s2s_m2, and s2s_m3, respectively. This indicates that post-positional particle estimation is more uncertain compared with the estimation of morphological inflection.

5 Experiments

To evaluate the proposed method, we conducted 100 experiments and for each test, we randomly selected 10 samples from the parallel corpus for testing and used the remaining samples to retrain the statistical machine translation model in Step 0. For each test, we finetuned the parameters of the seq2seq model (s2s_m1, s2s_m2, s2s_m3) by the training data and we used a beam size of 5 for decoding of the models. We averaged the 100 results to calculate the metrics of the performance.

We denote the proposed method in Section 3.1 by SMT+Iterative_s2s and compared its performance with the following baselines (naive, LSTMs, and SMT), the variants of the proposed method (SMT+1step_s2s and SMT+2step_s2s) and the transformer-based end-to-end Gloss2Text (G2T) model proposed by Yin and Read (2020). The followings are the brief explanations of each model.

- **naive**: This baseline replaces each gloss word with a Japanese word using the gloss dictionary. If more than one Japanese word is defined for a gloss, the first word is used.
- **LSTM**: This baseline uses encoder-decoder LSTM with an attention mechanism (Bahdanau et al., 2015) to directly translate $\mathcal{G}$ into $\mathcal{S}_{+PP+C}$. The model is trained using the parallel corpus without using the out-of-domain corpus and is configured with several different hyperparameter settings.
- **SMT**: This baseline uses only the statistical machine translation model to directly translate $\mathcal{G}$ into $\mathcal{S}_{+PP+C}$. This model is trained using the parallel corpus and without using the out-of-domain corpus.
- **SMT+1step_s2s**: This model is a variant of the proposed model which first executes Step 0 of Algorithm 1 to translate $\mathcal{G}$ into $\mathcal{S}_{-PP-C}$. Then it uses another seq2seq model trained using the out-of-domain corpus to directly translate $\mathcal{S}_{-PP-C}$ into $\mathcal{S}_{+PP+C}$. We compared the performance of this model, which jointly estimates post-positional particles and conjugations, with the model that estimates them separately using different models.
- **SMT+2step_s2s**: This model is another variant of the proposed model which performs Steps 0–2, but does not iteratively update the translation result as it does in Algorithm 1. We examined how the iterative updates of the result with **SMT+Iterative_s2s** contribute to the performance compared with the model without them.

For G2T, we changed the original hyperparameters suggested by Yin and Read (2020) and found that the following parameters were optimal using hyperparameter search on our parallel corpus. Encoder and decoder: embed-size = 256, FFN-embed-size = 1024, num-layer = 1, num-attention-head = 4.

5.1 Results

Table 2 shows the results of the experiment. To evaluate performance, we used the following metrics, BLEU-1/2/3/4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and word error rate (WER), and averaged the scores to obtain the results. The results showed that the proposed model (SMT+Iterative_s2s) outperformed the other models. The poor performance of the naive model indicates that the simple lookup method using the gloss dictionary did not produce successful results. LSTMs with different hyperparameters varying in the dimensions of the embedding and the hidden layers (256, 512, 1024) and the number of layers (1, 2) show the baseline performances to directly translate $\mathcal{G}$ into $\mathcal{S}_{+PP+C}$. Among them, the LSTM with the dimensions of the embedding and the hidden layers of 1024 and the number of layers of 1 showed the best performance. The best LSTM and the G2T were inferior

	embed	hidden	layers	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	WER
naive				0.184	0.049	0.007	0.002	0.139	0.801
LSTM	256	256	1	0.628	0.538	0.453	0.365	0.623	0.419
	512	512	1	0.692	0.608	0.527	0.442	0.688	0.349
	1024	1024	1	0.717	0.634	0.552	0.467	0.714	0.320
	256	256	2	0.448	0.314	0.222	0.154	0.408	0.616
	512	512	2	0.558	0.440	0.325	0.225	0.530	0.502
	1024	1024	2	0.590	0.466	0.332	0.209	0.556	0.466
G2T (Yin and Read, 2020)				0.695	0.640	0.592	0.535	0.708	0.305
SMT				0.788	0.724	0.663	0.599	0.800	0.233
SMT+1step_s2s				0.810	0.752	0.697	0.638	0.830	0.301
SMT+2step_s2s				0.811	0.756	0.701	0.642	0.829	0.245
SMT+Iterative_s2s				**0.817**	**0.762**	**0.707**	**0.648**	**0.833**	**0.216**

Table 2: Performance evaluations of naive, LSTMs, G2T (Yin and Read, 2020), SMT, SMT+1step_s2s, SMT+2step_s2s, and SMT+Iterative_s2s on BLEU1/2/3/4↑ and METEOR↑, and word error rate (WER) ↓ by averaging the scores from all experiments. The hyperparameters of LSTMs, the dimensions of the embedding and hidden layers, and the number of layers, are specified in the columns of 'embed', 'hidden', and 'layers', respectively. We use the same parameters for the encoder and the decoder of the LSTMs.

Model	Step	Source	Target	GS	EP
pbsmt	0	$\mathcal{G}$	$\mathcal{S}_{-PP-C}$	38.0	
s2s_m1	1	$\mathcal{S}_{-PP-C}$	$\mathcal{S}_{+PP-C}$	79.5	35.1
s2s_m2	2	$\mathcal{S}_{+PP-C}$	$\mathcal{S}_{+PP+C}$	99.2	35.1
s2s_m3	4	$\mathcal{S}_{-PP+C}$	$\mathcal{S}_{+PP+C}$	86.9	36.1

Table 3: Error propagation analysis of SMT+Iterative_s2s. The score is the exact match for the correct ratio (%) (GS = gold standard, EP = error propagation).

to the SMT because there were insufficient samples to train the neural models with large capacity. All the pipeline models that combined the SMT and seq2seq models outperformed the models that directly translate $\mathcal{G}$ into $\mathcal{S}_{+PP+C}$. This clearly demonstrates the effectiveness of the pipeline approach. Table 8 in Appendix illustrates the translation samples at each step of SMT+Iterative_s2s.

We investigated whether adding the monolingual Japanese corpus in 2.2 to train the target language model improved the performance of the SMT baseline. However, on the contrary, performance was slightly degraded. We believe that this was because of a domain mismatch between the corpora. The statistical significance test results confirmed that the performance of SMT+Iterative_s2s was significantly better than that of SMT, SMT+1step_s2s, and SMT+2step_s2s (see Table 10 in the Appendix).

Table 3 shows the error propagation analysis of SMT+Iterative_s2s. The score was measured using the exact match by counting the outputs that exactly matched the references at each step. Column 'GS' represents the gold standard score when using the ground truth input, and column 'EP' represents the score when using the output from the previous pipeline stage as the input propagating the errors. Clearly, a large portion of the error originated from Step 0 when translating $\mathcal{G}$ into $\mathcal{S}_{-PP-C}$. The GS score of s2s_m2 was much higher than that of s2s_m1 and s2s_m3, which was indicated by its higher BLEU score for the model evaluation on the test set described in Section 4. We verified that the EP score of s2s_m3 was 2.8% greater than that of s2s_m2, thereby illustrating the efficacy of the retrospective complement of post-positional particles. Note that the EP score of s2s_m3 was measured by allowing the output of s2s_m2 in the EP to be set to the input and removing all post-positional particles.

Table 4 shows the frequency of iterative update counts by SMT+Iterative_s2s. Approximately 72% of the results converged at the first iteration, and approximately 26% of the results converged at the second iteration. Counts above 6 were achieved when the same phrase was repeatedly generated, which is a phenomenon known as hallucination (Wang and Sennrich, 2020). If we detected such an error, we removed the repeating phrase to shorten the output.

5.2 Qualitative Evaluation

Table 5 shows the qualitative evaluations of the results using the proposed model and the other models with BLEU4, WER, and perplexity (PPL) scores. PPL in the last column was measured by the transformer-based language model that was pretrained by using the 494M-word Japanese Wikipedia.

We observed that the results of SMT and G2T had more post-positional particle selection errors than the other pipeline models, and the results of SMT+1step_s2s had more verb conjugation errors than SMT+2step_s2s, which suggest the efficacy of the independent estimation of post-positional particles and conjugations. We confirmed that the post-positional particle estimations using SMT+Iterative_s2s were either more natural or less error-prone than those using SMT+2step_s2s, which made the translation results more fluent.

Loop count	Freq.
1	723 (72.3%)
2	258 (25.8%)
3	4 (0.4%)
4	6 (0.6%)
6	1 (0.1%)
7	2 (0.2%)
8 ≥	6 (0.6%)

Table 4: Frequency of the iteration counts of Algorithm 1 until the translation output converged using SMT+Iterative_s2s.

Table 6 shows the average perplexity scores of the results of the SMT and pipeline models. While the perplexities of the pipeline models were much lower than that of SMT, the perplexity of the proposed SMT+Iterative_s2s was not the lowest. This result suggests that word-based perplexity is not suitable for evaluating equally acceptable translation outputs.

6 Discussion

In Table 5, most of the outputs using SMT+2step_s2s and SMT+Iterative_s2s were grammatically acceptable Japanese sentences with slight differences in the post-positional particle selections. As shown in the second and third examples in Table 5, the PPL scores of SMT+2step_s2s were lower than those of SMT+Iterative_s2s, but the BLEU4 and WER scores of SMT+Iterative_s2s were better than that of SMT+2step_s2s, even though the meanings of the sentences were almost the same. By contrast, the sentences of SMT+2step_s2s and SMT+Iterative_s2s in the first and last examples had different meanings, even though the PPL, BLEU4, and WER scores indicated that the results of SMT+Iterative_s2s were better than those of SMT+2step_s2s. However, depending on the context, the results of SMT+2step_s2s may be more appropriate. The main cause of the ambiguity issue is related to the information bottleneck raised by Yin and Read (2020) regarding the gloss notation of sign language. Currently, our parallel corpus does not include any non-manual signals (NMSs), such as facial expression, eye gaze, mouth, and movement of the head and shoulders. However, NMSs act as grammatical markings for syntactic information (Valli et al. 2011; Koizumi et al. 2002). NMSs are not expressed in sequence, but simultaneously with manual signs, and their subtleties make sign recognition and annotation more difficult. Perhaps, this is one of the reasons that most existing sign language corpora do not or only contain partial NMS labels along with glosses. As suggested by Yin and Read (2020), the performance of G2T translation may not impose an upper bound for sign-to-text translation unless the gloss faithfully describes the signed sentences. We are interested in investigating whether incorporating visual features from signs would improve the proposed G2T translation method. Because of the limited space in this paper, we leave this issue for future work.

Table 7 depicts examples of the translation errors by SMT+Iterative_s2s categorized into gloss word translation error, post-positional particle exchange, and conjugation exchange. As shown in Table 3, a large portion of the translation errors originated from gloss word translations. These errors mostly occurred because of the incorrect selection of Japanese wording for gloss phrases. For instance, the phrase 被災地 *hisaichi* ‘disaster area’ in the reference of the 2nd example in Table 7, which is expressed as a sequence of three glosses: 受ける ‘receive’, 災害 ‘disaster’, and 場所 ‘area’, was translated into the un-grammatical phrase, 受けて 災害場所 *ukete saigaibasho*. It is because the correct mapping from glosses to compound nouns cannot be learned by the phrase-based SMT unless they appear in the training set. The second major source of translation errors was the post-positional particle exchanges. These errors possibly change the semantic from the reference as indicated in the 5th example of Table 7, 図書館に本を寄贈する方法 “how to donate books to libr ray” v.s. 図書館の本を贈る方法 “how to give a book from the library”. As we mentioned above, some of these errors are difficult to handle because the system output may be correct in another context. Trans-

(1) Input $\mathcal{G}$	g467 g12 g77	BLEU4	WER	PPL
Reference	私の子供を探します。 “I look for my child.”			82.8
naive	私 子供 探す	0	0.714	117731.4
G2T	私 を 探し ます 。	0.474	0.286	119.7
SMT	私 子供 を 探し て い ます 。	0	0.429	59.3
SMT+1step_s2s	私 子供 探す ます 。	0	0.429	11715.6
STM+2step_s2s	私 も 子供 を 探し ます 。	0.643	0.143	129.6
SMT+Iterative_s2s	私 の 子供 を 探し ます 。	1	0	82.8
(2) Input $\mathcal{G}$	g19 g20 g87 g294 g20 g307 g202 g9	BLEU4	WER	PPL
Reference	医療 費の自己負担額は安くなりますか？ “Will my medical expense be cheaper?”			89.7
naive	医療 お金 自分 負担 お金 安い なる か	0	0.75	56747.3
G2T	医療 費 の 自己 負担 割合 は なり ます か ？	0.551	0.167	138.1
SMT	医療 費 の 自己 負担 の 安い に なり ます か ？	0.531	0.25	203.9
SMT+1step_s2s	医療 費 自己 負担 費 安い なる ます か ？	0	0.417	1223.0
STM+2step_s2s	医療 費 は 自己 負担 費 が 安く なり ます か ？	0.417	0.25	107.2
SMT+Iterative_s2s	医療 費 の 自己 負担 費 は 安く なり ます か ？	0.735	0.083	112.4
(3) Input $\mathcal{G}$	g73 g475 g52 g19 g27 g151 g2	BLEU4	WER	PPL
Reference	障害者向け医療支援を知りたい。 “I want to know medical support for disabilities.”			121.4
naive	障害 者 会う 医療 助成 わかる 欲しい	0	0.667	79281.5
G2T	障害 者 向け 医療 支援 を 知り たい 。	1	0	121.4
SMT	障害 者 向け 医療 支援 を 知り たい 。	1	0	121.4
SMT+1step_s2s	障害 者 向け 医療 支援 知る たい 。	0.525	0.222	768.0
STM+2step_s2s	障害 者 向け の 医療 支援 を 知り たい 。	0.658	0.111	62.7
SMT+Iterative_s2s	障害 者 向け 医療 支援 を 知り たい 。	1	0	121.4
(4) Input $\mathcal{G}$	g258 g860 g24 g8 g33 g9	BLEU4	WER	PPL
Reference	老人ホームの情報をいただけますか？ “Can you provide information about elderly housing with care ?”			155.8
naive	老人 住宅 情報 いただく できる か	0	0.667	4571.7
G2T	老人 ホーム の 情報 は いただけ ます か ？	0.597	0.111	148.3
SMT	老人 住宅 へ の 情報 を いただけ ます か ？	0.661	0.222	320.7
SMT+1step_s2s	老人 ホーム 情報 いただける ます か ？	0	0.333	943.4
STM+2step_s2s	老人 ホーム で 情報 を いただけ ます か ？	0.661	0.111	199.2
SMT+Iterative_s2s	老人 ホーム の 情報 を いただけ ます か ？	1	0	155.8

Table 5: Sample translation results by naive, G2T, SMT, SMT+1step_s2s, SMT+2step_s2s, and proposed SMT+Iterative_s2s. All the results of SMT+Iterative_s2s were when the iterative update converged in the second loop. PPL represents perplexity.

	Average perplexity
SMT	934.47
SMT+1step_s2s	241.44
SMT+2step_s2s	248.01
SMT+Iterative_s2s	269.92

Table 6: Average perplexity of translations using SMT, SMT+1step_s2s, SMT+1step_s2s, and SMT+Iterative_s2s measured by the transformer-based language model trained on the 494M-word Japanese Wikipedia.

lation errors relating to the conjugation exchange rarely occurred, and even if they did, the impacts were minimal.

7 Related works

Camgöz et al. (2018) proposed end-to-end sign language translation in the framework of neural machine translation, allowing them to jointly learn the spatial sign representation, underlying language model, and mapping between sign and spoken language using PHOENIX-Weather 2014T (Camgöz et al., 2018) corpus. Their later work (Camgöz et al., 2020) further improved the model by introducing a transformer-based architecture that jointly learns sign language recognition and translation while being trainable in an end-to-end manner using connectionist temporal classification loss to bind the recognition and translation problems into a single unified architecture.

In a similar research line, Yin and Read (2020) proposed the G2T model using a transformer-based seq2seq model, and evaluated the performance on PHOENIX-Weather 2014T (Camgöz et al., 2018) and ASLG-PC12 (Othman and Jemni, 2012) in various ways by changing the numbers of encoder-decoder layers and embedding schemes. All the end-to-end state-of-the-art sign language translation methods rely on large datasets and cannot be used for resource-poor datasets. The

Reference	System	Error type
どう すれ ば 留学 が でき ます か ？ “How can I study abroad?”	どの よう に 留学 でき ます か ？	gloss word translation error
被災 地 の ボランティア を し たい です 。 “I would like to volunteer in the disaster area.’	受け て 災害 場所 を ボランティア に し たい です 。	gloss word translation error
災害 に 備え て 何 を 備蓄 し て い ます か ？ “What are you stockpiling in case of a disaster?”	災害 に 備え て どんな 保存 が あり ます か ？	gloss word translation error
タクシー 代 は 医療 費 控除 でき ます か ？ “Can taxi fare be deducted from medical expenses?”	タクシー 費 や 医療 費 控除 は でき ます か ？	post-positional particle exchange
図書館 に 本 を 寄贈 する 方法 を 教え て ください 。 “Please tell me how to donate books to the library.”	図書館 の 本 を 贈る 方法 を 教え て ください 。	post-positional particle exchange
名字 だけ の 印鑑 登録 でき ます か ？ “Is it possible to register a seal with only the surname?”	姓 だけ __ 印鑑 登録 は でき ます か ？	post-positional particle exchange
収入 が ない 人 で も 保険 料 を 払い ます か ？ “Do you pay insurance premiums even if you have no income?”	収入 が ない 人 は 保険 料 を 払う の です か ？	post-positional particle exchange
何 か いい 情報 は あり ます か ？ Do you have any good information?	何 が よく 情報 が あり ます か ？	conjugation exchange
休日 に 子供 を 預け られる ところ は あり ます か ？ “Is there a place where I can leave my child on holidays?”	休日 に 子供 を 預け たり できる ところ は あり ます か ？	conjugation exchange

Table 7: Examples of the translation errors by SMT+Iterative_s2s are categorized into gloss word translation error, post-positional particle exchange, and conjugation exchange. We highlight the **wrong** words or phrases in bold.

pipeline method that we proposed is related to the transfer learning method proposed by Mocialov et al. (2018). They proposed transfer learning to improve British Sign Language modeling at a gloss level by fine-tuning or layer substitution on neural network models pre-trained on the Penn Treebank dataset. Although the purpose of their work was not to translate sign language, their work is similar to ours in that it takes advantage of linguistic commonality between resource-poor sign language and its spoken language. Our approach to converting non-grammatical sentences into grammatical sentences is related to previous work on grammatical error correction (Imamura et al. 2012; Liu et al. 2018; Oyama et al. 2013). They used insert or replace operations to correct particle or morphological inflection errors in a monolithic model, and we believe that the proposed seq2seq-based iterative method using multiple models can be used for similar tasks.

8 Conclusion

We proposed a pipeline machine translation method from SJ to Japanese by assuming that the gloss of the sign is provided. We focused on grammatical differences between SJ and Japanese, particularly post-positional particles and morphological inflections, and proposed a pipeline model by cascading the phrase-based statistical machine translation and three transformer-based seq2seq models, which effectively addressed the resource-poor issue of the sign language corpus. The statistical machine translation model first maps each gloss phrase to a Japanese phrase, then three seq2seq models pre-trained using the monolingual corpus transform the initial translation by complementing post-positional particles, and apply conjugations for verbs, auxiliary verbs, and adjectives. Translation is repeated until the output converges. We confirmed that the proposed method outperformed the SMT baseline by +4.4/+4.9 points for BLEU-3/4.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was funded in part by the joint project “Research and development on a mutual communication system that connects hearing-impaired and hearing people” between the University of Electro-Communications and SoftBank Corp.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations ICLR*, San Diego, CA, USA.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Jan Bungeroth, Daniel Stein, Philippe Dreuw, Hermann Ney, Sara Morrissey, Andy Way, and Lynette van Zijl. 2008. The ATIS sign language corpus. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

N. C. Camgöz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. 2018. Neural Sign Language Translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7784–7793.

N. C. Camgöz, O. Koller, S. Hadfield, and R. Bowden. 2020. Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10020–10030.

Hirohito Chonan. 2001. Grammatical Differences Between Japanese Sign Language, Pidgin Sign Japanese, and Manually Coded Japanese: Effects on Comprehension. *The Japanese Journal of Educational Psychology*, 49(4):417–426.

Kenji Imamura, Kuniko Saito, Kugatsu Sadamitsu, and Hitoshi Nishikawa. 2012. Grammar error correction using pseudo-error sentences and domain adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 388–392, Jeju Island, Korea. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Atsuko Koizumi, Hirohiko Sagawa, and Masaru Takeuchi. 2002. An annotated Japanese Sign Language corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Jun Liu, Fei Cheng, Yiran Wang, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Automatic error correction on Japanese functional expressions using character-based neural machine translation. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

Boris Mocialov, Helen Hastie, and Graham Turner. 2018. Transfer learning for British Sign Language modelling. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 101–110, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

A. Othman and M. Jemni. 2012. English-ASL Gloss Parallel Corpus 2012: ASLG-PC12.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Hiromi Oyama, Mamoru Komachi, and Yuji Matsumoto. 2013. Towards automatic error type classification of Japanese language learners' writings. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pages 163–172, Taipei, Taiwan. Department of English, National Chengchi University.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, page 3104–3112. MIT Press.

Clayton Valli, Ceil Lucas, Kristin J. Mulrooney, and Miako N.P. Rankin. 2011. *Linguistics of American Sign Language*. Gallaudet University Press, Washington, DC.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Kayo Yin and Jesse Read. 2020. Better sign language translation with STMC-transformer. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5975–5989, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Appendix

A Sample translations by SMT+Iterative_s2s

Pipeline translation from $\mathcal{G}$ into $\mathcal{S}_{+PP+C}$	
$\mathcal{G}$	g6 g847 g71 g64 g84 g9
$\mathcal{S}_{-PP-C}$	図書館 どこ ある ます か ？
$\mathcal{S}_{+PP-C}$	図書館 は どこ に ある ます か ？
$\mathcal{S}_{+PP+C}$	図書館 は どこ に あり ます か ？
	“Where is the library ?”
$\mathcal{G}$	g25 g26 g27 g470
$\mathcal{S}_{-PP-C}$	補聴器 購入 助成 お願い する ます 。
$\mathcal{S}_{+PP-C}$	補聴器 の 購入 の 助成 を お願い する ます 。
$\mathcal{S}_{+PP+C}$	補聴器 の 購入 の 助成 を お願い し ます 。
	“Please subsidize the purchase of hearing aids.”
$\mathcal{G}$	g215 g555 g28 g181 g470
$\mathcal{S}_{-PP-C}$	入院 手続き ついて 教える くださる 。
$\mathcal{S}_{+PP-C}$	入院 の 手続き について 教える て くださる 。
$\mathcal{S}_{+PP+C}$	入院 の 手続き について 教え て ください 。
	“Please tell me about the procedure for hospitalization.”

Table 8: Examples of translation results at each pipeline step by SMT+Iterative_s2s. The words in red are the complemented post-positional particles and the words in blue are the conjugated words from the underlined lemmas above. (See the gloss definitions in Table 11.)

B Hyper-parameters tuning results

encoder & decoder				s2s_m1	s2s_m2	s2s_m3
embed-dim	ffn-embed-dim	layers	attention-heads	BLEU4	BLEU4	BLEU4
512	2048	6	8	**74.20**	**98.75**	**75.06**
256	1024	3	4	73.99	96.84	74.88
128	512	2	2	70.75	98.26	71.58
64	256	1	1	56.33	94.84	61.70

Table 9: The results of hyperparameter tuning of the seq2seq models (s2s_m1, s2s_m2, and s2s_m3) used in Steps 1, 2, and 4 of Algorithm 1, respectively. The values are BLEU4 scores of the test samples extracted from the parallel corpus.

C Statistical significance test results

	BLEU3			BLEU4		
	SMT	SMT+1step_s2s	SMT+2step_s2s	SMT	SMT+1step_s2s	SMT+2step_s2s
SMT+1step_s2s	1e-10**			4e-06**		
SMT+2step_s2s	1e-10**	0.164		1e-10**	0.177	
SMT+Iterative_s2s	1e-10**	0.003**	0.013*	1e-10**	0.018*	0.049*

Table 10: Non-parametric bootstrap test results of BLEU3 and BLEU4 for SMT, SMT+1step_s2s, SMT+2step_s2s, and SMT+Iterative_s2s. The values are the p-values. (‘*’ : p-value <0.05, ‘**’ : p-value <0.01)

D Snippets of gloss definitions

g12	子供, 児童 “child, children”	g467	私 “I”
g151	知る, わかる “know”	g470	お願い “ask, please”
g181	教わる “learn”	g475	者, 民 “man, people”
g19	医療 “medical”	g52	向け, 対象, 対する “to, about”
g2	したい, 欲しい, 好き “want, like”	g555	手続き “procedure”
g20	費, お金, 費用 “expense, money”	g6	本, 手帳 “book, memo”
g202	なる “become”	g64	場所 “place”
g24	情報 “information”	g71	何, どの, どれ “what, which, what”
g25	補聴器 “hearing aid”	g73	障害, 壊す “disability, damage”
g258	老人 “old, elderly”	g77	探す, 観光, 見学 “search, sightseeing”
g26	購入 “buy”	g8	いただく, もらう “receive, have”
g27	助成, 支援, 補助 “support, assistance”	g84	ある, です “be”
g28	だから, ので, ついて “because, about, in regard to”	g847	建物 “building”
g294	負担 “charge”	g860	住宅 “housing”
g307	安い “cheap”	g87	自分 “self”
g33	できる, 可能, よろしい “can, may”	g9	か？, あなた “yes/no, you”

Table 11: Definitions of glosses used in Tables 5 and 8.

Language-Mediated, Object-Centric Representation Learning

Ruocheng Wang[*]
Stanford University

Jiayuan Mao[*]
MIT CSAIL

Samuel J. Gershman[†]
Harvard University

Jiajun Wu[†]
Stanford University

Abstract

We present Language-mediated, Object-centric Representation Learning (LORL), a paradigm for learning disentangled, object-centric scene representations from vision and language. LORL builds upon recent advances in unsupervised object discovery and segmentation, notably MONet and Slot Attention. While these algorithms learn an object-centric representation just by reconstructing the input image, LORL enables them to further learn to associate the learned representations to concepts, i.e., words for object categories, properties, and spatial relationships, from language input. These object-centric concepts derived from language facilitate the learning of object-centric representations. LORL can be integrated with various unsupervised object discovery algorithms that are language-agnostic. Experiments show that the integration of LORL consistently improves the performance of unsupervised object discovery methods on two datasets via the help of language. We also show that concepts learned by LORL, in conjunction with object discovery methods, aid downstream tasks such as referring expression comprehension.

1 Introduction

Cognitive studies show that human infants develop object individuation skill from diverse sources of information: spatial-temporal information, object property information, and language (Xu, 1999, 2007; Westermann and Mareschal, 2014). Specifically, young infants develop object-based attention that disentangles the motion and location of objects from their visual appearance features. Later on, they can leverage the knowledge acquired through word learning to solve the problem of object individuation: words provide clues about object identity and type. The general picture from cognitive science is that object perception and language co-develop in support of one another (Bloom, 2002).

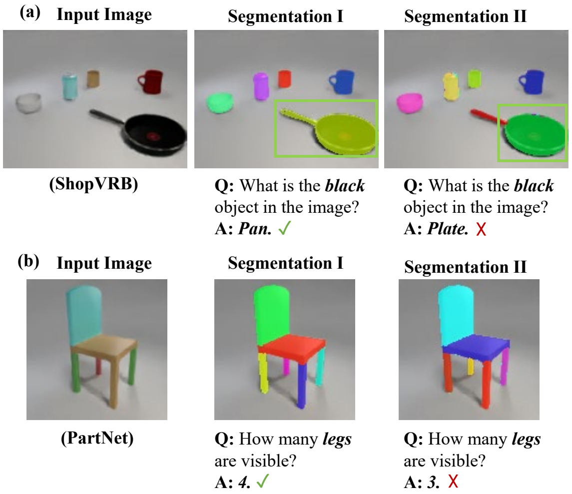

Figure 1: Two illustrative cases of Language-mediated, Object-centric Representation Learning. Different colors in the segmentation masks indicate individual objects recognized by the model. LORL can learn from visual and language inputs to associate various concepts: *black*, *pan*, *leg*, with the visual appearance of individual objects. Furthermore, language provides cues about how an input scene should be segmented into individual objects: (a) segmenting the frying pan and its handle into two parts (Segmentation II) yields an incorrect answer to the question ; (b) an incorrect parsing of the chair image makes the counting result wrong.

Our long-term goal is to endow machines with similar abilities. In this paper, we focus on how language may support object discovery and segmentation. Recent work has studied the problem of unsupervised object representation learning, though without language. As an example, factorized, object-centric scene representations have been used in various kinds of prediction (Goel et al., 2018), reasoning (Yi et al., 2018), and planning tasks (Veerapaneni et al., 2020), but they have not considered the role of language and how it may help object representation learning.

As a concrete example, consider the input images shown in Fig. 1 and the paired questions. From language, we can learn to associate concepts,

---

∗, †: indicates equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2033–2046
August 1–6, 2021.

such as *black, pan*, and *leg*, with the referred object's visual appearance. Further, language provides cues about how an input scene should be segmented into individual objects: a wrong parsing of the input scene will lead to an incorrect answer to the question. We can learn from such failure that the handle belongs to the frying pan (Fig. 1a) and the chair has four legs (Fig. 1b).

Motivated by these observations, we propose a computational learning paradigm, Language-mediated, Object-centric Representation Learning (LORL), associating learned object-centric representations to their visual appearance (masks) in images, and to concepts—words for object properties such as color, shape, and material—as provided in language. Here the language input can be either descriptive sentences or question-answer pairs. LORL requires no annotations on object masks, categories, or properties during the learning process.

In LORL, four modules are jointly trained. The first is an image encoder, learning to encode an image into factorized, object-centric representations. The second is an image decoder, learning to reconstruct masks for individual objects from the learned representations by reconstructing the input. These two modules share the same formulation as recent unsupervised object discovery research: learning to decompose the image into a series of *slot* profiles, comprised of pixel masks and latent embeddings. Each slot profile is expected to represent a single object in the image.

The third module in LORL is a pre-trained semantic parser that translates the input sentence into a semantic, executable program, where each concept (i.e., words for object properties such as *red*) is associated with a vector space embedding. Finally, the last module, a neural-symbolic program executor, takes the object-centric representation from Module 1, intermediate representations from Module 2, and concept embeddings and the semantic program from Module 3 as input, and outputs an answer if the language input is a question, or `TRUE/FALSE` if it's a descriptive sentence. The correctness of the executor's output and the quality of reconstructed images (as output of Module 2) are the two supervisory signals we use to jointly train Modules 1, 2, and 4.

We integrate the proposed LORL with state-of-the-art unsupervised discovery methods, MONet (Burgess et al., 2019) and Slot Attention (Locatello et al., 2020). The evaluation is based on two datasets: ShopVRB (Nazarczuk and Mikolajczyk, 2020) contains images of daily objects and question-answer pairs; PartNet (Mo et al., 2019) contains images of furniture with hierarchical structure, supplemented by descriptive sentences we collected ourselves. We show that LORL consistently improves existing methods on unsupervised object discovery, much more likely to group different parts of a single object into a single mask.

We further analyze the object-centric representations learned by LORL. In LORL, conceptually similar objects (e.g. objects of similar shapes) appear to be clustered in the embedding space. Moreover, experiments demonstrate that the learned concepts can be used in new tasks, such as visual grounding of referring expressions, without any additional fine-tuning.

2 Related Work

Unsupervised object representation learning. Given an input image, unsupervised object representation learning methods segment objects in the scene and build an object-centric representation for them. A mainstream approach has focused on using compositional generative scene models that decompose the scene as a mixture of component images (Greff et al., 2016; Eslami et al., 2016; Greff et al., 2017; Burgess et al., 2019; Engelcke et al., 2020; Greff et al., 2019; Locatello et al., 2020; Goyal et al., 2020). In general, these models use an encoder-decoder architecture: the image encoder encodes the input image into a set of latent object representations, which are fed into the image decoder to reconstruct the image. Specifically, (Greff et al., 2019; Burgess et al., 2019; Engelcke et al., 2020) use recurrent encoders that iteratively localize and encode objects in the scene. Another line of research (Eslami et al., 2016; Crawford and Pineau, 2019; Kosiorek et al., 2018; Stelzner et al., 2019; Lin et al., 2020) leverages object locality to attend to different local patches of the image. These models often use a pixel-level reconstruction loss. In contrast, we propose to explore how language, in addition to visual observations, may contribute to object-centric representation learning. There has also been work that uses other types of supervision, such as dynamic prediction (Kipf et al., 2020; Bear et al., 2020) and multi-view consistency (Prabhudesai et al., 2020). In this paper, we focus on unsupervised learning of object-centric representations from static images and language.

Visual concept learning. Learning visual concepts from language and other forms of supervision provides useful representations for various downstream tasks, such as image captioning (Yin and Ordonez, 2017; Wang et al., 2018), visual-question answering (Yi et al., 2018; Huang et al., 2019), shape differentiation (Achlioptas et al., 2019), image classification (Mu et al., 2020), and scene manipulation (Prabhudesai et al., 2020). Previous work has been focusing on various types of representations (Ren et al., 2016; Wu et al., 2017), training algorithms (Faghri et al., 2018; Morgado et al., 2020) and supervision (Johnson et al., 2016; Yang et al., 2018). In this paper, we focus on learning visual concepts that can be grounded in object-centric representations. Recent work on object-centric grounding of visual concepts (Wu et al., 2017; Mao et al., 2019; Hudson and Manning, 2019; Prabhudesai et al., 2020) has shown great success in achieving high performance in downstream tasks and strong generalization from a small amount of data. However, these methods assume pre-trained object detectors to generate object proposals in the scene. In contrast, our LORL learns to individuate objects and associates concepts with the learned object-centric representations without any annotations on object segmentation masks or properties.

3 Preliminaries

Before delving into our language-mediated object-centric representation learning paradigm, we first discuss a general formulation that unifies multiple concurrent unsupervised object representation learning methods and a neuro-symbolic framework for learning visual concepts from language.

3.1 Unsupervised Object-Centric Representation Learning

Given an image I, a typical unsupervised object representation learning model will decompose the scene into a series of *slot* profiles $\{(z_1, x_1, m_1), \ldots, (z_K, x_K, m_K)\}$, where each slot profile is expected to represent an object (or nothing, as the number of slots may be greater than the actual number of objects in the scene). Here z_i is the object feature, x_i is the object image, and m_i is the object mask specifying its location in the scene.

In our paper, we focus on two recent models, MONet (Burgess et al., 2019) and Slot Attention (Locatello et al., 2020). MONet uses a recurrent spatial attention network (Ronneberger et al., 2015) to segment out objects in the scene, and adopts a variational autoencoder (Kingma and Welling, 2014) to encode objects as well as reconstruct object images for self-supervision. At a very high level, its objective function is calculated as

$$\mathcal{L} = \left\| \left(\sum_{k=1}^{K} m_k x_k \right) - I \right\|_2^2 + \beta \cdot \sum_{k=1}^{K} \mathrm{KL}(z_k), \quad (1)$$

where the first term is a pixel-wise $\mathcal{L}_2$ reconstruction loss and the second term computes the KL divergence between the distribution of z_k's and a prior Gaussian distribution.

Slot Attention uses a transformer-like attention network (Vaswani et al., 2017) to extract object features, and decode them with convolutional neural networks to component images and object masks. The model is trained by the same reconstruction loss in the form of the $\mathcal{L}_2$-norm:

$$\mathcal{L} = \left\| \left(\sum_{k=1}^{K} m_k x_k \right) - I \right\|_2^2 . \quad (2)$$

3.2 Neuro-Symbolic Concept Learning

The neuro-symbolic concept learner (NS-CL; Mao et al., 2019) learns visual concepts by looking at images and reading paired questions and answers. NS-CL takes a set of segmented objects in a given image as its input, extracts their visual features with a ResNet (He et al., 2015), translates the input question into an executable program by a semantic parser, and executes the program based on the object-centric representation to answer the question. The key idea of NS-CL is to explicitly represent individual concepts in natural language (colors, shapes, spatial relationships, etc.) as vector space embeddings, and associate them with the object embeddings.

NS-CL answers the input question by executing the program based on the object-centric representation. For example, in order to query the name of the white object in Fig. 2, NS-CL first filters out the object by computing the cosine similarity between the concept *white* and individual object representations, which produces a "mask" vector where each entry denotes the probability that an object has been selected. The output "mask" on

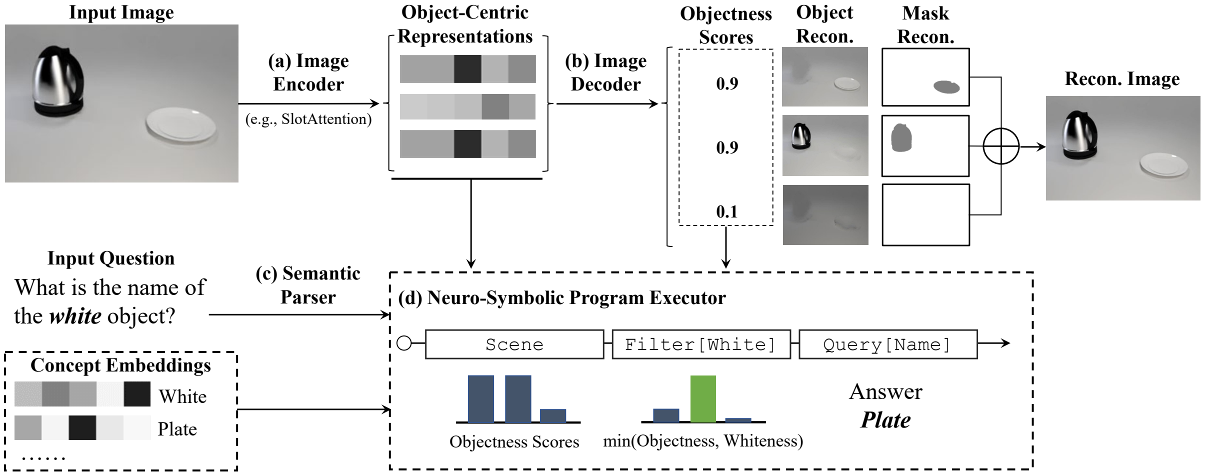

Figure 2: Our LORL contains four modules. (a) An image encoder encodes the input image into a factorized, object-centric representation. (b) An image decoder learns to reconstruct the image from the learned representations. It also decodes an objectness score based on the representation. (c) A pre-trained semantic parser translates the input sentence into an executable program and associates concepts with learnable vector embeddings. (d) A neuro-symbolic program executor takes the object-centric representation, the objectness scores, the parsed program, and concept embeddings as input to predict the answer (if the input is a question) or `TRUE/FALSE` (if the input is a descriptive sentence).

the objects is fed into the next module and the execution will continue. The last query operation produces the answer to the question. The vector embeddings of individual objects and the concepts are jointly trained based on language supervision.

4 Language-mediated, Object-centric Representation Learning

Marrying the ideas of unsupervised object-centric representation learning and neuro-symbolic concept learning, we are able to learn an object-centric representation using both visual and language supervision. Fig. 2 shows an overview of Language-mediated, Object-centric Representation Learning (LORL). In LORL, four modules are optimized jointly: an image encoder, an image decoder, a semantic parser, and a neuro-symbolic program executor.

Image encoder. Given an input image, we first use the image encoder (Fig. 2a) to individuate objects in the scene and extract an object-centric scene representation. It takes the input image as its input, individuates objects in the scene, and produces a collection of latent slot embeddings $\{z_i\}$.

Image decoder. The decoder (Fig. 2b) takes the object-centric representation produced by the image encoder and produces a 3-tuple for each individual slot (x_k, m_k, s_k), where x_k reconstructs the RGB image of the slot, m_k reconstructs the mask, and $s_k \in [0, 1]$ is a scalar indicating the *objectness* of the slot. That is, whether k-th slot corresponds to a *single* object in the scene. Here, we have extended the general pipeline we described in Section 3.1 with an objectness indicator. It serves dual purposes. First, it weights each reconstructed component image while generating the reconstructed image. Mathematically, the reconstructed image I' is computed as: $I' = \sum_{k=1}^{K} s_k \cdot (m_k x_k)$. Second, it mediates the output of all `filter` operations in the program executor.

In this paper, we will experiment with two image encoder-decoder options: MONet (Burgess et al., 2019) and Slot Attention (Locatello et al., 2020). They are both compatible with the learning paradigm described above. For both models, we use a single linear layer to predict the objectness score for each slot on top of the second-last layer of their image decoders.

Semantic parser. A pre-trained semantic parser (Fig. 2c) will translate the input question into an executable program composed of primitive operations, such as `filter`, which filters out objects with certain concepts and `query`, which queries the attribute of the input object. We use roughly the same domain-specific language (DSL) for representing programs as CLEVR (Johnson et al., 2017a, see also the appendix for details). All concepts that appear in the program, such as *white*, are associated with distinct, learnable concept embedding vectors.

Q: There is a plastic thing; are there any heavy metal things behind it?

P: `scene, filter[plastic], relate[behind], filter[heavy, metal], exist`

A: True

Q: What number of ceramic things are there?

P: `scene, filter[ceramic], count`

A: 1

Q: The small plate is what color?

P: `scene, filter[small], filter[plate], query[color]`

A: purple

(a) Examples on Shop-VRB-Simple

D: The color of the back is brown.

P: `scene, filter[back], query[color], equal[brown]`

D: The number of legs there is 4.

P: `scene, filter[leg], count, equal[4]`

D: There is a part left of the cyan seat, its color is red.

P: `scene, filter[cyan, seat], relate[left], equal[red]`

(b) Examples on PartNet-Chairs

Figure 3: In Shop-VRB-Simple, questions involve seven different attributes of objects in the scene: *name, size, weight, material, color, shape, mobility*. In PartNet-Chairs, images are paired with sentences describing the *names* and *colors* of the parts in the scene, and their relationships.

Neuro-symbolic program executor. The program executor (Fig. 2d) takes the object-centric representation from the image encoder $\{z_k\}$, the objectness score $\{s_k\}$ from the image decoder, the concept embeddings and the program generated by the semantic parser as its input. It executes the program based on the visual and concept representations to answer the question.

The original program executor in NS-CL (Section 3.2) assumes a pre-trained object detector for generating object proposals. In LORL, we associate each object representation with an objectness score s_k. Recall that a `filter` operation in NS-CL produces a mask vector indicating whether an object has been selected. Here, we mediate the output of an `filter(c)` operation as $\min(s_k, \texttt{filter(c)})$. Intuitively, a slot will be selected only if, first, it has concept c and, second, it corresponds to a single object in the scene.

Training paradigm. During training, we jointly optimize the image encoder, the image decoder, and the concept embeddings. They are trained by minimizing the loss $\mathcal{L}$:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{perception}} + \beta \cdot \mathcal{L}_{\text{reasoning}}.$$

For MONet-based image encoder-decoder, we use Equation 1 as the perception loss $\mathcal{L}_{\text{perception}}$, while for Slot Attention-based encoder-decoder, we use Equation 2. The neuro-symbolic program executor produces a distribution over candidate answers to the input question. We use the cross-entropy loss between the predicted answer and the ground truth answer as $\mathcal{L}_{\text{reasoning}}$.

We use a three-stage training paradigm in LORL. First, we train the model with only visual inputs with $\mathcal{L}_{\text{perception}}$ for N_1 epochs. Next, we fix the image encoder and the image decoder, and optimize the concept embeddings with the loss term $\mathcal{L}_{\text{reasoning}}$ for N_2 epochs. During this second stage, the image encoder and the decoder can already produce decent object segmentation results. Finally, we jointly optimize all three modules for N_3 epochs. We provide detailed information about the hyperparameters for different models in the appendix.

5 Experiments

We first evaluate whether the representations learned by LORL lead to better image segmentation with the help of language. We then evaluate how these representations may be used for instance retrieval, visual reasoning, and referring expression comprehension.

5.1 Image Segmentation

Data. We use two datasets for image segmentation evaluation. The first, Shop-VRB-Simple, is based on Shop-VRB (Nazarczuk and Mikolajczyk, 2020), a dataset of complex household objects and question-answer pairs. The second is based on chairs in PartNet (Mo et al., 2019), a dataset where the objects are different parts of a chair. Fig. 3 shows some examples from the two datasets.

Shop-VRB is a visual reasoning dataset, similar to CLEVR (Johnson et al., 2017a), but with complex household objects of different sizes, weights, materials, colors, shapes, and mobility. Because the original Shop-VRB dataset includes very small and highly transparent objects and complex backgrounds, which current unsupervised representation learning models cannot handle, we generate 10K images with a clean background ourselves us-

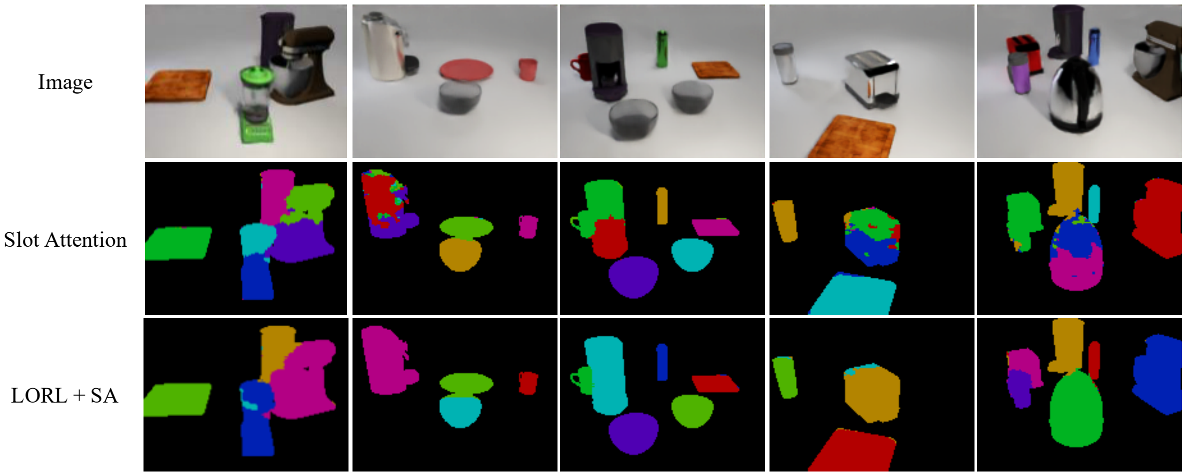

Figure 4: Visualization on Shop-VRB-Simple. Pixels with the same color represent a mask produced by the models. The Slot Attention model often fails to segment blenders, coffee makers, and toasters. LORL helps to greatly improve its results.

(a)	ARI↑	GT Split↓	Pred Split↓
Slot Attention	83.51$_{\pm 2.3}$	15.68$_{\pm 1.9}$	13.19$_{\pm 1.5}$
LORL + SA	**89.23**$_{\pm 1.6}$	**9.95**$_{\pm 1.6}$	**10.18**$_{\pm 1.3}$

(b)	Slot Attention	LORL + SA
Coffee maker	39.4$_{\pm 7.0}$	**21.9**$_{\pm 8.3}$
Blender	38.9$_{\pm 11.5}$	**17.7**$_{\pm 3.4}$
Toaster	33.4$_{\pm 3.9}$	**16.4**$_{\pm 7.2}$

Table 1: (a) Results on Shop-VRB-Simple. LORL helps to improve Slot Attention (SA; Locatello et al., 2020) in all metrics. (b) The Ground Truth split ratio for the three object categories where SA most commonly fail. LORL helps SA to reduce the ratio by 50%.

ing large objects from the dataset. We also pair every image with 9 questions, resulting in 90K questions in total. The test split has 960 images and 8.6K questions. We name this variant Shop-VRB-Simple.

While the previous literature on unsupervised object segmentation mainly focuses on settings where objects are spatially disentangled, we also explore how language may help when objects of interest are different parts of a global shape. To this end, we collect a new dataset, PartNet-Chairs, using chair shapes from PartNet. Every image here shows a chair, where each part of the chair (legs, seat, back, arms) is randomly assigned a color. We select six different chair shapes with one or four legs and zero or two arms. We generate 5K images for training. Each image is paired with 8 descriptive sentences generated from human-written templates, resulting in 40K examples in total. The test split has 960 images. Each sentence describes the *name* and *color* of parts. We provide all templates in the supplementary material. We are interested in whether object-centric representation learning models may separate these parts and whether and how language may help in this scenario.

Baselines. We use MONet (Burgess et al., 2019) and Slot Attention (Locatello et al., 2020) as the image modules (Modules 1 and 2), and evaluate how the incorporation of language may improve their performance. Because MONet is color-sensitive, and on Shop-VRB-Simple, many objects have diverse colors and specular reflection, it does not produce meaningful results there. Thus we only show results with MONet on PartNet-Chairs. We show results with Slot Attention on both datasets.

Metrics. We use three metrics for evaluation. Following (Greff et al., 2019), we first use the Adjusted Rand Index (ARI; Rand, 1971; Hubert and Arabie, 1985). It treats segmentation as a clustering problem: each mask is the cluster index that the pixels within belong to. ARI is computed as the similarity between the predicted and ground truth clusters, and ranges from 0 (random) to 1 (perfect match).

In practice, we found this pixel-wise metric is sensitive to the size of objects: a model that infrequently makes mistakes on large objects will have lower ARI than one that frequently mis-segments small objects. Thus, we in addition design two object-centric metrics:

- **Ground Truth Split Ratio** (GT Split) measures the ratio of objects (GT masks) that are covered by more than one prediction mask.
- **Prediction Split Ratio** (Pred Split) measures the ratio of prediction masks that cover more

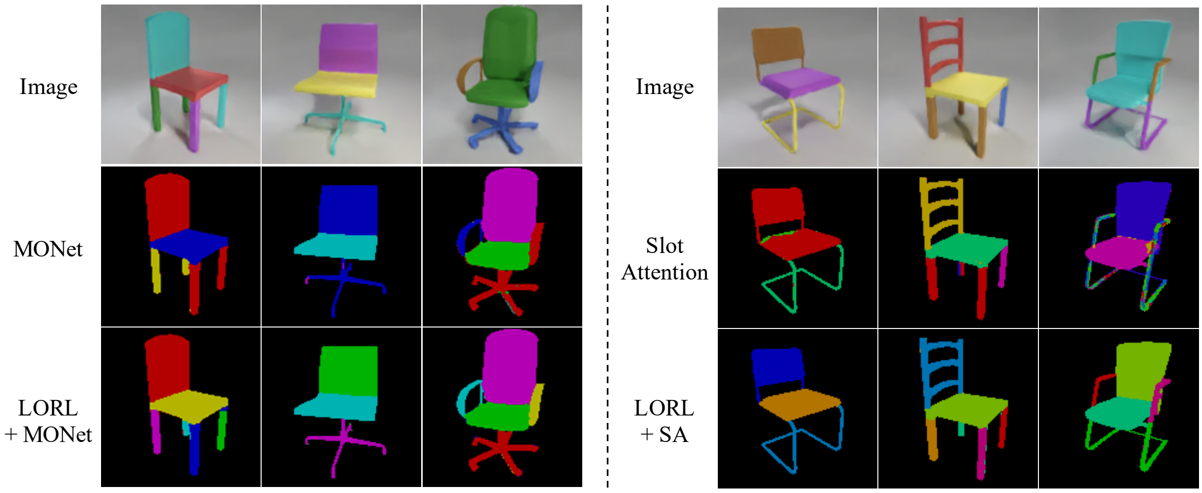

Figure 5: Visualization on PartNet-Chairs. Pixels with the same color represent a mask produced by the models. LORL successfully recognizes different parts in various situations.

	ARI↑	GT Split↓	Pred Split↓
MONet	$91.41_{\pm 3.7}$	$10.3_{\pm 5.1}$	$14.09_{\pm 5.2}$
LORL + MONet	$94.91_{\pm 2.1}$	$4.95_{\pm 0.7}$	$4.02_{\pm 2.5}$
Slot Attention	$87.32_{\pm 3.6}$	$12.54_{\pm 6.6}$	$22.99_{\pm 5.0}$
LORL + SA	$\mathbf{95.81}_{\pm 1.0}$	$\mathbf{3.39}_{\pm 1.1}$	$\mathbf{2.92}_{\pm 1.0}$

Table 2: Quantitative results on PartNet-Chairs. All numbers are in percentage. LORL consistently improves MONet's and Slot Attention's performance on segmentation.

than one object (GT mask).

Concretely, we first assign each pixel to the prediction mask with the maximum value at the pixel. We say a prediction mask covers an object if it covers at least 20% of the object's pixels. The GT and Pred Split ratios are thus defined as:

$$\text{GTSplit} = \frac{\#\text{ of objects that are covered by} > 1 \text{ masks}}{\#\text{ of objects that are covered by} > 0 \text{ masks}},$$

$$\text{PredSplit} = \frac{\#\text{ of masks that cover} > 1 \text{ objects}}{\#\text{ of masks that cover} > 0 \text{ objects}}.$$

Ideally, there is a one-to-one correspondence between objects and predicted masks, with both GT Split and Pred Split being 0. Please refer to the appendix for detailed comparison of the proposed metrics with other metrics.

Results. The quantitative results on SHOP-VRB-Simple are summarized in Table 1. We show the mean and standard error on each metric over 3 runs. Since our semantic parsing module is trained on paired question-program pairs, it achieves nearly perfect accuracy (>99.9%) on test questions. Thus, in later sections, we will focus on evaluating object segmentation, concept grounding, and downstream task performances. LORL helps Slot Attention achieve better segmentation results in all three metrics. From visualizations in Fig. 4, we find that the original Slot Attention model struggles with metallic objects; but with LORL, it performs much better in those cases.

To further explore how LORL helps Slot Attention on failure cases, we calculate the Ground Truth Split Ratio for each object category, and find that Slot Attention most often fail to segment coffee makers, blenders, and toasters as a whole. These objects have complex sub-parts and their appearance changes quickly when the viewpoint changes. With the help of language, Slot Attention improves consistently over its ablative variants across all the three metrics, reducing the GT Split by 50% on average (Table 1b). Furthermore, we include ablation studies on how different types of questions and different modules (the objectness score module and the concept learning module) contribute to the performance improvement in the appendix.

On PartNet-Chairs, LORL also helps both MONet and Slot Attention improve with a large margin, as shown in Table 2. The results are averaged over 4 runs. MONet in general performs well on this dataset, though it still sometimes merges different parts with the same color into a single mask. An example can be found in Fig. 5, column 3, where the blue arm and the blue bottom in the input image are put into the same mask by MONet. Such an issue is alleviated in LORL + MONet. Fig. 5 also includes examples to show how LORL helps Slot Attention.

As shown in Table 2, the improvement on Slot Attention is larger and more consistent, compared

	$k=1$	$k=3$	$k=5$
Slot Attention	$54.07_{\pm 2.6}$	$43.70_{\pm 1.7}$	$37.02_{\pm 1.7}$
LORL + SA	$\mathbf{94.03}_{\pm 0.6}$	$\mathbf{91.03}_{\pm 1.3}$	$\mathbf{87.71}_{\pm 1.7}$

Table 3: The percentage (%) of retrieved objects that belong to the same category as the query object. With LORL, objects within the same category are more likely to be close to each other in the feature space. The results are averaged over 3 runs and standard errors are given.

	QA Accuracy
LORL + SA (No FT)	$62.79_{\pm 1.6}$
LORL + SA	$\mathbf{92.72}_{\pm 1.0}$

Table 4: Our three-stage training paradigm improves visual concept learning. Without fine-tuning (i.e., the third training stage), the question answering accuracy drops by 30%. The results are averaged over 3 runs and standard errors are given.

with the improvement on MONet. We hypothesize that this is because the two models adopt different approaches for aligning object features and masks. While MONet uses separate modules for segmentation and object representation learning, Slot Attention obtains masks by directly decoding object representations. Having a shared representation might have allowed Slot Attention to gain more from language supervision.

5.2 Instance Retrieval

We now analyze the learned object representations on Shop-VRB-Simple. We first use them for instance retrieval: for each model, we randomly select a segmented object and use its learned representation to search for its k nearest neighbors in the feature space. Then, for each selected object, we compute how many of the k nearest neighbors belong to the same category. During searching, we only consider object representations whose corresponding mask, after decoding, has at least an Intersection over Union (IoU) of at least 0.75 with a ground truth object mask. We sample 1,000 object features from each model for evaluation.

Table 3 includes results with $k = 1, 3, 5$, suggesting that the object representations learned by LORL + Slot Attention are better for retrieval, compared with features learned by Slot Attention alone without language. This is because Slot Attention often confuses categories that are visually similar but conceptually different, such as *baking tray* and *chopping board.*

5.3 Visual Reasoning

As another analysis, we also evaluate how the learned representation of LORL + Slot Attention performs on visual question answering on the Shop-VRB-Simple dataset. Here we compare with an ablated version of LORL, where we only train the model for the first two stages, as stated at the end of Section 4. We do not train the model for the third stage—jointly optimizing or fine-tuning all three trainable modules. We name this ablation LORL + SA (No FT). Through this analysis, we hope to understand the importance of joint training of the vision modules (Modules 1 and 2) and the reasoning module (Module 4).

Table 4 shows that joint training is crucial for visual reasoning. This resonates with the previous result, where visually similar objects are clustered together in the latent space, impeding the usefulness of the information encoded.

5.4 Referring Expression Comprehension

Finally we evaluate the representations learned by LORL on referring expression comprehension, where given an expression referring to a set of objects in the scene, like “The white plates”, the model is expected to return the corresponding object masks. After learning all needed concepts from question-answer pairs, LORL can naturally handle referring expression without any further training, if we assume a pre-trained semantic parser.

We choose the IEP-Ref model (Liu et al., 2019) as our baseline. It uses a module approach and receives direct segmentation supervision. On Shop-VRB, we adapt the code provided by (Liu et al., 2019) to generate 17K training examples, only for IEP-Ref, and 1.7K testing referring expressions for evaluating both LORL and IEP-Ref. Measured in Recall@0.5 (the ratio of the recalled objects based on an IoU threshold of 0.5), IEP-Ref performs better than LORL, but the margin is small (90.1% *vs.* 84.4%). Note that while IEP-Ref has been trained on 17K training examples with ground truth object segmentations, LORL does not require any training data on referring expression comprehension. The relatively comparable results are strong evidence that the representations learned by LORL also transfer to a new task.

6 Conclusion

We have proposed Language-mediated, Object-centric Representation Learning (LORL), a paradigm for learning object-centric representations from vision and language. Experiments on Shop-VRB-Simple and PartNet-Chairs show that language significantly contributes to learning better representations. This behavior is consistent across two unsupervised image segmentation models.

Through systematic studies, we have also shown how LORL helps models to learn object representations that encode conceptual information, and are useful for downstream tasks such as retrieval, visual reasoning, and referring expression comprehension.

Acknowledgments

We thank Ruidong Wu for helpful discussions in the early stage of the project. This work in part supported by the Samsung Global Research Outreach (GRO) program, Autodesk, IBM, an Amazon Research Award (ARA), and Stanford HAI.

References

Panos Achlioptas, Judy Fan, Robert Hawkins, Noah Goodman, and Leonidas J Guibas. 2019. Shapeglot: Learning language for shape differentiation. In *CVPR*.

Daniel M Bear, Chaofei Fan, Damian Mrowca, Yunzhu Li, Seth Alter, Aran Nayebi, Jeremy Schwartz, Li Fei-Fei, Jiajun Wu, Joshua B Tenenbaum, et al. 2020. Learning physical graph representations from visual scenes. In *NeurIPS*.

Paul Bloom. 2002. *How Children Learn the Meanings of Words*. MIT press.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. 2019. Monet: Unsupervised scene decomposition and representation. *arXiv:1901.11390*.

Eric Crawford and Joelle Pineau. 2019. Spatially invariant unsupervised object detection with convolutional neural networks. In *AAAI*.

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. 2020. Genesis: Generative scene inference and sampling with object-centric latent representations. In *ICML*.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, Koray Kavukcuoglu, and Geoffrey E Hinton. 2016. Attend, infer, repeat: Fast scene understanding with generative models. In *NeurIPS*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*.

Vikash Goel, Jameson Weng, and Pascal Poupart. 2018. Unsupervised video object segmentation for deep reinforcement learning. In *NeurIPS*.

Anirudh Goyal, Alex Lamb, Phanideep Gampa, Philippe Beaudoin, Sergey Levine, Charles Blundell, Yoshua Bengio, and Michael Mozer. 2020. Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv:2006.16225*.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. 2019. Multi-object representation learning with iterative variational inference. In *ICML*.

Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Jürgen Schmidhuber. 2016. Tagger: Deep unsupervised perceptual grouping. In *NeurIPS*.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. 2017. Neural expectation maximization. In *NeurIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. In *CVPR*.

Pingping Huang, Jianhui Huang, Yuqing Guo, Min Qiao, and Yong Zhu. 2019. Multi-grained attention with object-level grounding for visual question answering. In *ACL*.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Drew Hudson and Christopher D Manning. 2019. Learning by abstraction: The neural state machine. In *NeurIPS*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017a. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017b. Inferring and executing programs for visual reasoning. In *ICCV*.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Thomas Kipf, Elise van der Pol, and Max Welling. 2020. Contrastive learning of structured world models. In *ICLR*.

Adam R. Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. 2018. Sequential attend, infer, repeat: Generative modelling of moving objects. In *NeurIPS*.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. 2020. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *ICLR*.

Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L Yuille. 2019. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *CVPR*.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. 2020. Object-centric learning with slot attention. In *NeurIPS*.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. 2019. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*.

Pedro Morgado, Yi Li, and Nuno Nvasconcelos. 2020. Learning representations from audio-visual spatial alignment. In *NeurIPS*.

Jesse Mu, Percy Liang, and Noah Goodman. 2020. Shaping visual representations with language for few-shot classification. In *ACL*.

M. Nazarczuk and K. Mikolajczyk. 2020. Shop-vrb: A visual reasoning benchmark for object perception. In *ICRA*.

Mihir Prabhudesai, H. F. Tung, Syed Ashar Javed, Maximilian Sieb, Adam W. Harley, and K. Fragkiadaki. 2020. Embodied language grounding with 3d visual feature representations. In *CVPR*.

William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. 2016. Joint image-text representation by gaussian visual-semantic embedding. In *ACM MM*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Karl Stelzner, Robert Peharz, and Kristian Kersting. 2019. Faster attend-infer-repeat with tractable probabilistic models. In *ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. 2020. Entity abstraction in visual model-based reinforcement learning. In *CoRL*.

Josiah Wang, Pranava Swaroop Madhyastha, and Lucia Specia. 2018. Object counts! bringing explicit detections back into image captioning. In *NAACL*.

Gert Westermann and Denis Mareschal. 2014. From perceptual to language-mediated categorization. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1634):20120391.

Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. 2017. Neural scene de-rendering. In *CVPR*.

Fei Xu. 1999. Object individuation and object identity in infancy: The role of spatiotemporal information, object property information, and language. *Acta psychologica*, 102(2-3):113–136.

Fei Xu. 2007. Sortal concepts, object individuation, and language. *Trends in Cognitive Sciences*, 11(9):400–406.

Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. 2018. Visual curiosity: Learning to ask questions to learn visual recognition. In *CoRL*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*.

Xuwang Yin and Vicente Ordonez. 2017. Obj2Text: Generating visually descriptive language from object layouts. In *EMNLP*.

A Domain-Specific Language (DSL)

LORL extends the domain-specific language of the CLEVR dataset (Johnson et al., 2017a) to accommodate descriptive sentences. Specifically, we add an extra primitive operation: `Equal(X, y)`. It takes two inputs. In our case, the first argument `X` is the output of a `Query`, `Exist`, or `Count` operation. All three operations output a distribution over possible answers. The second argument `y` is either a word or number, such as `TRUE`, *white*, or 4. The `Equal` operation computes the probability of `X=y`. In LORL, models are trained to maximize the output probability.

B Hyperparameters

For optimization hyperparameters, we largely adopt original settings in Burgess et al. (2019) and Locatello et al. (2020). Table 5 summarizes the hyperparameters for the loss weights (α and β), the number of training epochs of different stages (N_1, N_2, N_3), and the batch size. We early-stop the training when QA performance converges. We skip the second training phase on PartNet-Chairs, because the first phase (vision-only) yields poor segmentation performance on this dataset. Establishing a meaningful grounding of concepts could be hard in this case. If we keep the second training phase for LORL + Slot Attention on PartNet-Chairs, the model converges slower in the third training phase (15 more epochs in our experiments), but the final performance remains the same.

	Shop-VRB	PartNet-Chairs	
	Slot Attention	MONet	Slot Attention
α	1	0.01	1
β	0.1	1	0.1
Batch Size	64	64	64
N_1	90	100	50
N_2	20	0	0
N_3	80	200	200

Table 5: Hyperparameters of LORL

Learning rate scheduling. For Slot Attention models, during the first training stage (perception-only), we use the learning rate schedule described in the original paper on both datasets. Initially, the learning rate is linearly increased from zero to 4×10^{-4} in the first 10K iterations. After that, we decay the learning rate by 0.5 for every 100K iterations. On PartNet-Chairs, after the first stage, Slot Attention models continue to use the same learning rate scheduling. For Shop-VRB-Simple, we switch to a fixed learning rate of 0.001 during N_2 phase, which takes 20 epochs. After 20 epochs, we decrease the learning rate to 2×10^{-4}. We further decrease the learning rate to 2×10^{-5} after another 65 epochs. We use the Adam optimizer (Kingma and Ba, 2015) for Slot Attention models.

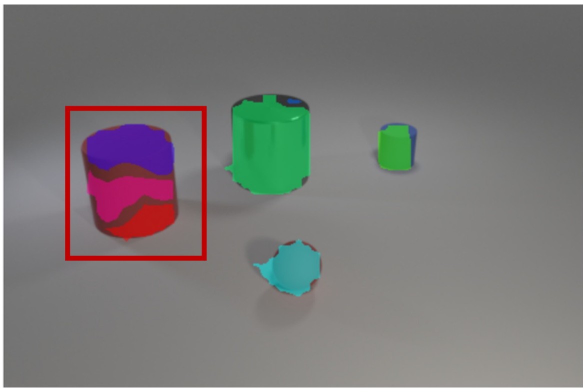

(a) There are 4 objects in the scene; one of them is split into 3 masks. Thus, GT Split $= 1/4 = 0.25$.

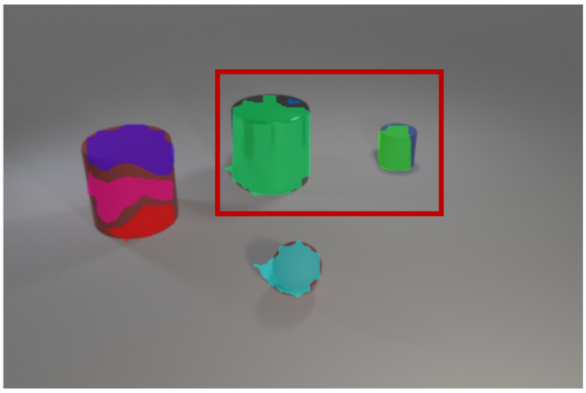

(b) There are 5 masks in the scene; one of them covers two objects. Thus, Pred Split $= 1/5 = 0.2$.

Figure 6: Two examples on how GT/Pred Split Ratios are computed.

For MONet models, we use RMSProp with a learning rate of 0.01 during the first stage, and use 0.001 for the second and the third stage.

Meanwhile, we also follow NS-CL (Mao et al., 2019) to use curriculum learning. Specifically, in the second training stage, we limit the number of objects in the scene to be 3. In the third training stage, we gradually increase the number of objects in the scene and the complexity of the questions.

C Implementation Details

In this section, we provide additional implementation details of our experimental setups and metrics.

GT and Pred split ratios. In this paper, we have introduced two new metrics for evaluating the performance of unsupervised object segmentation: namely, the GT split and and the Pred split ratio.

A simple example of how we can compute

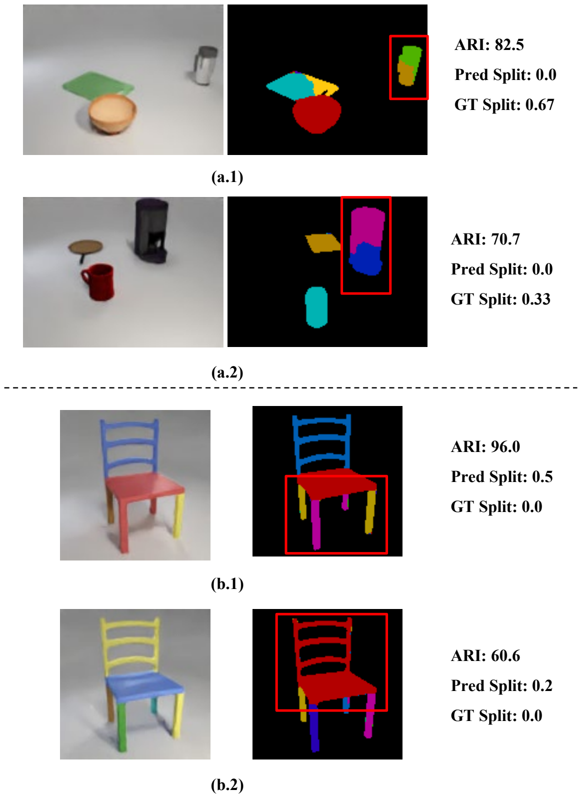

Figure 7: Comparison of ARI and GT/Pred Split Ratios on example images. Pixels with the same color represent a mask produced by the models. We find that ARI is very sensitive to the size of objects, while split ratios capture object-level failures where an object is split or multiple objects are merged.

GT/Pred split ratios is shown in Fig. 6. At a high level, the GT split ratio computes the percentage of objects that are split into multiple parts in model segmentation. Meanwhile, Pred split ratio computes the percentage of objects that are merged into a single object in model segmentation. We introduce these two new metrics because the ARI score is evaluated at the pixel level and does not account for the variance of object sizes. By contrast, GT split and Pred split metrics are computed at the object level. This difference is illustrated in Fig. 7.

For concrete examples, in the Fig. 6 (a.1), two objects, the chopping board and the thermos, are wrongly segmented. In Fig. 6 (a.2), only one object mis-segmented. However, the ARI score of the first image is much higher because the coffee maker has a large size. GT split ratio is evaluated on the object level and thus favor the second one. Similarly, in the Fig. 6 (b.1), the four legs are merged into two masks, while in Fig. 6 (b.2), the seat and the back of the chair are merged into a single object. However, the first segmentation result has a significantly higher ARI score because the chair legs only contribute to a small area in the image. In this paper, we propose to jointly use ARI scores and the proposed GT/Pred split ratio to evaluate segmentation masks.

Throughout the paper, we have been using IoU = 0.2 as the threshold while computing the GT/Pred split ratios. Table 6 summarizes the results with different IoU thresholds. LORL consistently improves the baseline.

Referring expression comprehension. In this experiment, the data is generated using the code adapted from Liu et al. (2019). It contains two types of expressions, the first one directly refers to an object by its properties: for example, "the white plate". The second type of sentences refers to the object by relating it with another object: for example, "the object that is in front of the mug." The output of the model is the masks of all referred objects. The dataset is composed of the same set of concepts and same DSL as in the Shop-VRB-Simple.

We use the IEP-Ref model proposed in Liu et al. (2019) as a baseline. It is adapted from its prior work IEP (Johnson et al., 2017b). IEP-Ref first translates the referring expression into a sequence of primitive operations, which are implemented as different modular networks. The model takes the image feature extracted by a CNN as input and executes the program by chaining these component networks. It outputs a segmentation mask on objects being referred to. During training, groundtruth segmentation masks are needed.

For all methods, including ours and the baseline, we assume a pretrained semantic parser. Since the neuro-symbolic program executor outputs a distribution over all objects indicating whether they are selected, we directly multiply its output with the object segmentation masks to get the final output.

D Additional Results

The following section presents a collection of ablation studies on different modules of LORL, as well as a few extensions.

Objectness score. To validate the effectiveness of the proposed objectness score module, we hereby compare two models: the original Slot Attention model and the Slot Attention model augmented with the proposed objectness score module. Both models are trained using images only (there is no language in the loop), on the Shop-VRB-Simple

	Thres = 0.1		Thres = 0.3	
	GT Split↓	Pred Split↓	GT Split↓	Pred Split↓
Slot Attention	$23.71_{\pm 1.5}$	$23.45_{\pm 4.1}$	$8.91_{\pm 0.7}$	$7.47_{\pm 0.9}$
LORL + SA	$\mathbf{17.74}_{\pm \mathbf{1.4}}$	$\mathbf{18.15}_{\pm \mathbf{1.2}}$	$\mathbf{5.86}_{\pm \mathbf{1.3}}$	$\mathbf{5.67}_{\pm \mathbf{0.7}}$

Table 6: GT/Pred split ratios on Shop-VRB-Simple using different IoU thresholds. The results are averaged over 3 runs and standard errors are given.

	ARI↑	GT Split↓	Pred Split↓
Slot Attention	$\mathbf{83.51}_{\pm \mathbf{2.3}}$	$\mathbf{15.68}_{\pm \mathbf{1.9}}$	$13.19_{\pm 1.5}$
SA + Obj score	$83.4_{\pm 1.8}$	$16.57_{\pm 1.5}$	$\mathbf{12.71}_{\pm \mathbf{0.8}}$

Table 7: Ablation study of the objectness score module on Shop-VRB-Simple. The results are averaged over 3 runs and standard errors are given.

	ARI↑	GT Split↓	Pred Split↓
SA (image-only)	$83.51_{\pm 2.3}$	$15.68_{\pm 1.9}$	$13.19_{\pm 1.5}$
Count only	$85.52_{\pm 1.7}$	$13.86_{\pm 2.0}$	$12.56_{\pm 1.4}$
Exist only	$86.29_{\pm 2.2}$	$15.34_{\pm 2.9}$	$10.33_{\pm 1.1}$
Query only	$88.79_{\pm 1.1}$	$10.64_{\pm 1.7}$	$9.03_{\pm 0.5}$
All types	$\mathbf{89.23}_{\pm \mathbf{1.6}}$	$9.95_{\pm 1.6}$	$\mathbf{10.18}_{\pm \mathbf{1.3}}$

Table 8: Ablation study of using different types of questions to train LORL + SA on Shop-VRB-Simple. The standard errors are computed based on 3 runs.

dataset. The Table 7 summarizes the result. The objectness module alone does not contribute to the segmentation performance.

Question type. In this section, we investigate how different types of questions affect LORL. We use the Shop-VRB-Simple dataset for evaluation. There are three types of questions in the dataset, counting (e.g., how many plates are there?), existence (e.g., is there a toaster?), and query (e.g., what is the color of the mug?). We train LORL +SA with only one single type of questions (the number of total questions is the same).

Results are summarized in Table 8. In general, training on all three types of questions improves the segmentation accuracy. The largest gain comes from the query question. Interestingly, the best result is achieved when trained on the original dataset, where the ratio of counting, existence, and query questions is 1:1:7. Note that all these models are trained with the same number of questions and thus they are directly comparable with each other.

Data efficiency. In addition, we provide another analysis by comparing models trained with different number of question-answer pairs. The results are shown in Table 9. Adding more language data consistently improves the result. All results are based on the LORL +SA model trained on the Shop-VRB-Simple dataset.

	ARI↑	GT Split↓	Pred Split↓
25% (22.5K)	81.01	14.31	15.34
50% (45K)	84.39	14.79	9.37
75% (67.5K)	86.53	11.67	11.52
100% (90K)	89.23	9.95	10.18

Table 9: Ablation study of using different number of questions to train LORL + SA on Shop-VRB-Simple.

Visual reasoning baselines. To establish a baseline for visual reasoning tasks, we present the results of two visual reasoning approaches IEP (Johnson et al., 2017b) and NS-CL (Mao et al., 2019) for reference on the Shop-VRB-Simple dataset, as shown in Table 12. All models are trained with the same set of question-answer pairs. Note that NSCL has the access to a pretrained object detection module, while LORL +SA and IEP do not. LORL +SA outperforms IEP, which is trained with exactly the same amount of supervision as ours. It also achieves a comparable result as NS-CL.

Integration with SPACE. SPACE (Lin et al., 2020) is another popular method for unsupervised object-centric representation learning. SPACE uses parallel spatial attention to decompose the input scene into a collection of objects, and it is also compatible with the proposed learning paradigm LORL. We include additional results of LORL +SPACE on the CLEVR dataset. Shown in the Table 11, LORL +SPACE shows a significant advantage over the vanilla SPACE model. Additionally, we find that SPACE shows poor segmentation results on Shop-VRB-Simple and ParNet-Chairs, no matter whether it is integrated with LORL. For example, it frequently segments complex objects into too many fragments on Shop-VRB-Simple. We conjecture that this is because SPACE was designed for segmenting objects of similar sizes.

Baseline using language supervision. We also conducted an additional baseline model that uses language supervision in a different way. Specifically, based on the Slot Attention model, we use a GRU to directly encode question and answer, and concatenate it with the image feature to obtain the object representation. On Shop-VRB-Simple, this

	Precision			Recall		
	@0.5	@0.75	@1	@0.5	@0.75	@1
LORL + SA (NO)	$59.52_{\pm 0.7}$	$54.13_{\pm 2.2}$	$36.36_{\pm 3.6}$	$\mathbf{94.96_{\pm 1.2}}$	$\mathbf{86.4_{\pm 3.6}}$	$\mathbf{58.07_{\pm 5.7}}$
LORL + SA	$\mathbf{89.47_{\pm 0.7}}$	$\mathbf{79.37_{\pm 1.2}}$	$\mathbf{52.48_{\pm 1.6}}$	$94.72_{\pm 0.1}$	$84.02_{\pm 0.6}$	$55.55_{\pm 1.2}$

Table 10: Concept quantification evaluation. The number after @ indicates the IoU threshold. The results suggest that objectness score improves the precision of concept quantification.The results are averaged over 3 runs.

	ARI↑	GT Split↓	Pred Split↓
SPACE (Lin et al., 2020)	72.34	29.2	12.38
LORL + SPACE	**97.82**	**2.04**	**2.17**

Table 11: Segmentation performance of SPACE and LORL +SPACE on CLEVR. The integration of LORL improves the result.

	QA Accuracy
LORL + SA (No FT)	$62.79_{\pm 1.6}$
IEP (Johnson et al., 2017b)	$78.3_{\pm 0.1}$
NSCL (Mao et al., 2019)	$\mathbf{97.9_{\pm 0.0}}$
LORL + SA	$92.72_{\pm 1.0}$

Table 12: Question answering accuracy on the Shop-VRB-Simple dataset. The results are averaged over 3 runs and standard errors are given.

model does not show improvement over the Slot Attention baseline: ARI $= 76.4\%$, GT Split $= 29.2\%$, Pred Split $= 13.6\%$. This suggests the effectiveness of LORL.

Concept quantification. Although LORL without the objectness score can achieve a comparable result in terms of QA accuracy, objectness score is crucial if we want to evaluate how models discover objects in images. Here, we show that, on the Shop-VRB-Simple dataset, LORL +SA shows significant improvement in recovering a holistic scene representation.

Specifically, we extract a scene graph for each scene, where each node corresponds to a detected object. We represent each node i as a set of concepts C_i associated with the object (e.g., $\{$*large, brown, wooden, chopping board*$\}$). We associate a concept with a detected object if its cosine similarity with the object representation is greater than 0. We heuristically remove nodes that are not associated with any concepts (by treating them as “background” objects) or have objectness scores that are smaller than 0.5. This results in a scene graph, where each node corresponds to a detected object. In the following, we compare it against the groundtruth scene graph.

For each pair of groundtruth node i and detected node j, we compute the concept IoU score based on their associated concepts C_i and C_j as:

$$IoU_{ij} = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}.$$

Next, we perform a maximum weight matching between the detected scene graph and the groundtruth scene graph with the Hungarian algorithm. We use the IoU score as the weight for every edge and remove edges whose IoU score is smaller than a given threshold. Finally, based on the macthing, we can compute the precision and recall of the detected scene graph. We show the average precision and recall over the entire test set images in Table 10. The results suggest that objectness score significantly improves the precision of the extracted concepts.

Entheos: A Multimodal Dataset for Studying Enthusiasm

Carla Viegas
Carnegie Mellon University
Pittsburgh, USA
`cviegas@cs.cmu.edu`

Malihe Alikhani
University of Pittsburgh
Pittsburgh, USA
`malihe@pitt.edu`

Abstract

Enthusiasm plays an important role in engaging communication. It enables speakers to be distinguished and remembered, creating an emotional bond that inspires and motivates their addressees to act, listen, and coordinate (Bettencourt et al., 1983). Although people can easily identify enthusiasm, this is a rather difficult task for machines due to the lack of resources and models that can help them understand or generate enthusiastic behavior. We introduce Entheos, the first multimodal dataset for studying enthusiasm composed of video, audio, and text. We present several baseline models and an ablation study using different features, showing the importance of pitch, loudness, and discourse relation parsing in distinguishing enthusiastic communication.

1 Overview

Although different emotional constructs such as *anger* and *happiness* have been studied extensively in the field of natural language processing (NLP), more fine-grained emotional expressions such as enthusiasm or charisma are relatively unexplored. Such models and datasets can benefit different areas of NLP and AI. Multimodal human-machine interaction can be more effective if systems can find a deeper understanding of more complex emotional responses or generate appropriate emotionally-aware communicative presentations. Given the importance of enthusiasm in teaching (Bettencourt et al., 1983; Zhang, 2014), for instance, researchers are studying the effect of virtual agents and robots that can behave in an enthusiastic manner (Liew et al., 2017, 2020; Saad et al., 2019). The current research is far from generating natural enthusiastic behavior.

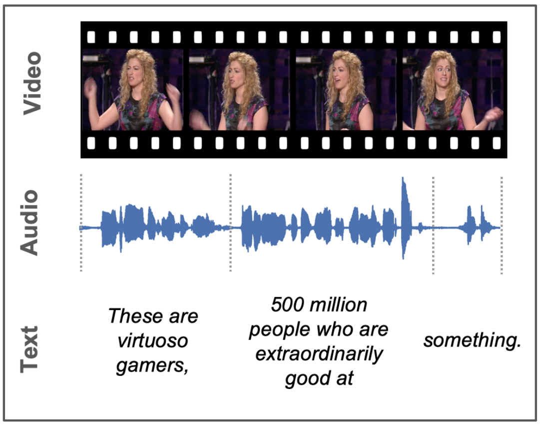

Figure 1: An enthusiastic sample from the Entheos dataset, showing aligned video frames, audio, and text.

Although previous research results in psychology, education, and business have studied the importance of enthusiasm in communication (Bettencourt et al., 1983; Sandberg, 2007; Keating, 2011; Antonakis et al., 2019), it is relatively unexplored in the NLP and dialogue literature. We take a step to bridge this gap by introducing the first multimodal dataset labeled with levels of enthusiasm following the definition that Keller et al. (2016) provided.

Our contributions are as follows: First, we present Entheos (ἔνθεος: being possessed by a god, root for enthusiasm), the first multimodal dataset of TED talk speeches with annotated enthusiasm level[1] (Section 3). It contains sentence segments, labeled as either monotonous, normal, or enthusiastic. Figure 1 shows an example of an enthusiastic sample. Second, in search of finding multimodal signals for understating enthusiasm, we present an analysis of our data to identify attributes present in enthusiastic speech in different modalities (Section 3.5 and 5). Finally, we also provide several baseline models using different kinds of features extracted from text, speech, and video. In

---

[1] `https://github.com/clviegas/Entheos-Dataset`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2047–2060
August 1–6, 2021. ©2021 Association for Computational Linguistics

addition, we show the importance of identifying discourse relations in predicting enthusiasm (Section 5).

2 Related Work

In this paper, we focus on investigating resources and models that can help us gain insights into ways by which computers can understand and predict enthusiasm. This topic is relatively unexplored in the computer science field although it has been extensively studied in psychology (Bettencourt et al., 1983; Sandberg, 2007; Keating, 2011; Antonakis et al., 2019).

Enthusiasm Limited work exists on the automatic detection of enthusiasm and has been mainly done in the text domain. Inaba et al. (2011) worked on the detection of enthusiasm in human text-based dialogues, using lexical features and word co-occurrences with conditional random fields in order to distinguish enthusiastic utterances from non-enthusiastic ones. They defined enthusiasm as "the strength of each participant's desire to continue the dialogue each time he/she makes an utterance". In our work, we instead combine different modalities and features to detect enthusiasm and we define an enthusiastic speaker as "stimulating, energetic, and motivating" (Keller et al., 2016). Tokuhisa and Terashima (2006) also worked with human-to-human conversational dialogues and annotated dialogue acts (DAs) and rhetorical relations (RRs) on a sentence-level. An enthusiasm score in the range of 10-90 was given without providing examples to the annotators. The relationship between DAs, RRs, and enthusiasm was analyzed based on the frequencies. They found that affective and cooperative utterances are significant in an enthusiastic dialogue. We detected RRs automatically and trained a feed forward network to classify enthusiasm in three levels: monotonous, normal, and enthusiastic. During data annotation, examples for each category were available as references. Twitter data have also been used to detect enthusiasm. Mishra and Diesner (2019) created a dataset with enthusiastic and passive labels. Enthusiastic tweets had to include personal expression of emotion or call to action, whereas passive tweets lacked clear emotive content or call to action. They trained logistic regression models using salient terms. We evaluate emotional expressions in several modalities. We use acoustic features that relate to emotion such as pitch and voice quality, and also Facial Action Units extracted from videos which measure the intensity of different facial expressions.

Charisma Enthusiasm is also a trait that can be displayed by charismatic speakers (Spencer, 1973), which in addition are perceived as competent, passionate, and self-confident (Niebuhr, 2020). Charisma is a desired trait for leaders in economy and politics (Antonakis et al., 2019; De Jong and Den Hartog, 2007) because it can influence followers to undertake personally costly yet socially beneficial actions. Niebuhr et al. (2016) have investigated the prosodic attributes of charismatic speakers. They analyzed pitch level, pitch variation, loudness, duration of silence intervals, etc and concluded that charisma can be trained as far as melodic features are concerned. In addition to analyzing the relationship of different attributes with enthusiasm, we also trained a model that can distinguish between different levels of enthusiasm.

Although sentiment analysis and emotion detection have been studied extensively in unimodal and multimodal frameworks as shown in several surveys (Marechal et al., 2019; Garcia-Garcia et al., 2017; Seyeditabari et al., 2018; Sudhakar and Anil, 2015) there is a gap in the analysis, detection and generation of enthusiastic behavior. Our dataset will allow to extend the work in understanding human behavior and also generate more natural virtual agents (Zhang, 2014; Keller et al., 2014; Liew et al., 2020; Viegas et al., 2020).

3 Entheos Dataset

In this section we present the Entheos dataset. We describe our domain choice and label selection, the annotation process, extracted features, as well as statistics of the dataset.

3.1 Data Acquisition

Enthusiastic speakers are passionate about their message, wanting to gain their audience for their purpose and persuading them to change their perspective or take action. Given that TED is well-known for spreading powerful messages that can change attitudes and behavior, we use TED talk speeches as our domain for creating a multimodal enthusiasm dataset. We randomly selected 52 male and female speakers from the TEDLIUM corpus release 3 (Hernandez et al., 2018), which contains audio of 2351 talks. Transcripts were obtained

Rating	Description
4: Advanced	Excellent use of vocal variation, intensity and pacing; vocal expression natural and enthusiastic; avoids fillers
3: Proficient	Good vocal variation and pace; vocal expression suited to assignment; few if any fillers
2: Basic	Demonstrates some vocal variation; enunciates clearly and speaks audibly; generally avoids fillers (e.g. um, uh, like)
1: Minimal	Sometimes uses a voice too soft or articulation too indistinct for listeners to comfortably hear; often uses fillers
0: Deficient	Speaks inaudibly; enunciates poorly; speaks in monotone; poor pacing; distracts listeners with fillers

Table 1: Description of the Public Speaking Competence Rubric (PSCR) (Schreiber et al., 2012) evaluated as potential label to describe the use of vocal expressions and paralanguage during a talk.

Vocal Attributes	Description	Rating
Variation	Vocal variety is the spice of speech. Tone, pace, and volume should all be varied over the course of a presentation.	4: excellent, 3: good, 2: some, 1: almost no vocal variation, 0: speaks in monotone
Intensity	Speaks loudly and clearly enough for listeners to hear and understand what is being said.	4: excellent use, 3: good, 2: enunciates clearly and speaks audibly , 1: sometimes voice too soft or articulation too indistinct for listeners to comfortably hear, 0: inaudibly, enunciates poorly
Pacing	Speaks in an understandable rate and places pauses for emphasis.	4: excellent use including well placed pauses, 3: good, 2: pace is appropriate but could have more/less pauses, 1: poor pacing, 0: poor pacing with no/too many pauses
Expression	Emotion delivered by the voice.	4: natural and enthusiastic, 3: suited to assignment, 2: some expressions, 1: few expressions, 0: no expressions)

Table 2: Fine-grained description of vocal attributes derived from PSCR, evaluated as potential label categories on sentence-level.

through the Google cloud transcription service[2]. The talks were segmented into sentences, based on punctuation. We extend the samples from the TEDLIUM corpus with aligned video segments downloaded from the official TED website.

3.2 Label Selection and Temporal Granularity

In order to define the temporal granularity for annotation and what labels to use, we performed preliminary annotation experiments with three annotators.

Three audio recordings of talks were chosen from speakers with different proficiency level. One recording was a TED talk by Al Gore[3], and the remaining were recordings of participants in a pilot study with our institution in which they introduce themselves and describe their skills.

We evaluated two different temporal granularities: sentence-level and entire talk. In addition, we explored the use of three different sets of labels, which will be described in the following.

PSCR (Public Speaking Competence Rubric) PSCR (Schreiber et al., 2012) was developed to effectively assess students' skills in public speaking. It is composed of eleven skills that are assessed during speaking with a 0-4 scale. We focused on the seventh, which evaluates the effective use of vocal expression and paralanguage to engage the audience. During annotation, annotators had Table 1 available for a detailed description on how the speaker articulates for the corresponding rating.

Vocal Attributes Based on the PSCR descriptions we crystallized four main components of the effective use of the voice: vocal variation, intensity, pacing, and expression. Each one was evaluated with a score of 0-4 and described as depicted in Table 2.

Enthusiasm and Emphasis As a final set of labels, we decided to use intuitive categories, namely enthusiasm and emphasis. For enthusiasm, we chose the definition provided by Keller et al. (2016) as they study enthusiasm in context of spoken

---

[2] `https://cloud.google.com/speech-to-text`

[3] `https://www.ted.com/talks/al_gore_averting_the_climate_crisis`

Category	Description	Rating
Enthusiasm	Speaker is passionate, energetic, stimulating, and motivating.	0: monotonous, 1: normal, 2: enthusiastic
Emphasis	One or more words are emphasized by speaking louder or pronouncing them slowly.	0: no emphasis, 1: emphasis existent

Table 3: Intuitive labels used to evaluate as potential categories to annotate sentence-level samples.

Label	Fleiss' κ	Agreement
PSCR	0.31	fair
Variation	0.56	moderate
Intensity	0.81	almost perfect
Pacing	0.55	moderate
Expression	0.63	substantial
Enthusiasm	**0.82**	**almost perfect**
Emphasis	**0.87**	**almost perfect**

Table 4: Inter-rater agreement using different labels computed with Fleiss' kappa with interpretations based on Landis and Koch (1977). Enthusiasm, emphasis, and vocal intensity achieved almost perfect agreement.

monologues (similar to our data) while Inaba et al. (2011) studied written dialogues. We also asked annotators to label enthusiasm in three levels: monotonous, normal, and enthusiastic. As Table 3 shows, annotators were asked to label emphasis as existent or not, depending on whether words were emphasized by speaking louder or pronouncing words slowly.

Experiment Description The experiment was composed of two parts. First the entire audio recordings were played and the annotators were asked to use only the PSCR annotation scheme, rating each talk with a single score. Afterwards, seven sentences of each talk were played with pauses in between to allow annotation using vocal attributes, enthusiasm and emphasis labels. Each sentence was annotated with six scores. For both parts, the annotators had access to the description of the labels during annotation as shown in Tables 1,2,3. Once all annotators finished labeling a sample, the next one was played.

Results and Conclusion In Table 4 the inter-rater agreement for the different annotation schemes is shown in terms of Fleiss' kappa Landis and Koch (1977). We can see that PSCR, which rated the entire talk, has the lowest agreement. Vocal variation and pacing have moderate agreement, while vocal intensity, enthusiasm, and emphasis show almost perfect agreement.

Given these results, we annotated audio recordings on a sentence-level using enthusiasm and emphasis labels.

3.3 Data Annotation Protocol

Our study was approved by our institution's human subject board and annotators were paid $20/h. Seventeen subjects participated in data annotation and signed the consent form before the study. For data annotation, an internal tool was created that enabled annotators to listen to audio samples and annotate them through their web browser at their time of convenience. As labeling availability fluctuated, instead of randomly choosing samples from the entire dataset, we decided to release small batches of data to obtain as many annotations per sample as possible. In a bi-weekly rhythm, small batches of 200 samples were available to annotate in a randomly chosen order for each annotator. As our definition for enthusiasm (Table 3) allows subjective interpretations, we included three reference audio files for each enthusiasm level in the web interface of our annotation tool as depicted in Figure 2. Annotators were indicated to listen to the reference files after every 10 labeled samples and when insecure on how to label a sample. In addition, annotators were given the definition of enthusiasm and emphasis shown at Table 3. Besides enthusiasm and emphasis, also the corresponding perceived gender was annotated. We limited the options for perceived gender to female and male, based on prior work which used these two genders to improve the performance in emotion detection (Li et al., 2019). Samples with laughter or clapping were asked to be labeled as noisy files.

Annotator Quality Assessment: Annotation was performed by 17 different annotators. As noisy annotations are common when crowdsourcing and not using expert annotators due to spammers and malicious workers (Burmania et al., 2015), we com-

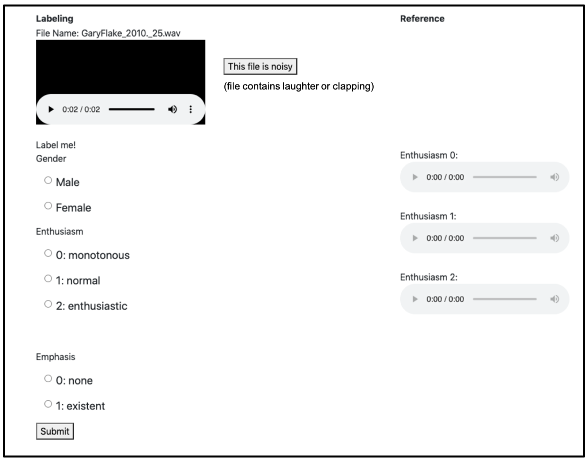

Labeling

File Name: GaryFlake_2010._25.wav

▶ 0:02 / 0:02

This file is noisy

(file contains laughter or clapping)

Label me!

Gender

○ Male

○ Female

Enthusiasm

○ 0: monotonous

○ 1: normal

○ 2: enthusiastic

Emphasis

○ 0: none

○ 1: existent

Submit

Reference

Enthusiasm 0:

▶ 0:00 / 0:00

Enthusiasm 1:

▶ 0:00 / 0:00

Enthusiasm 2:

▶ 0:00 / 0:00

Figure 2: Layout of the annotation interface. On the top left is the sample to be annotated and below are the different labels: perceived gender, enthusiasm, and emphasis. On the top center is the option to mark the sample as noisy if laughter or clapping is present. On the right side are reference samples for the three different levels of enthusiasm.

pared the percentage agreement of each individual's annotations with a preliminary majority vote. The analysis showed that 12 annotators had lower agreement than 30%. The same annotators had also labeled less than 17% of the data. To ensure high quality of annotation we used the remaining five annotators who labeled more than 50% of the data. The remaining annotators identify themselves as latino, asian, and white. We removed all samples that had only one or two different annotations and computed the final majority vote for the remaining 1,126 samples. To confirm high inter-rater agreement, we computed Cohen's kappa (McHugh, 2012) in a pairwise manner for the five annotators and obtained an average agreement of 0.66.

3.4 Final Data Selection

Out of 1,819 labeled samples, we kept 1,126 which had more than one annotation. The selected samples are from 113 different TED talk speeches, being 60 from male and 53 from female speakers. We created a test split with 108 samples from five speakers of each perceived gender. The training set, composed by 55 male and 48 female speakers, has a total of 1,018 samples. There is no overlap of speakers between training and test set. In Figure 3 (top) we can see the label distribution in our train-test split.

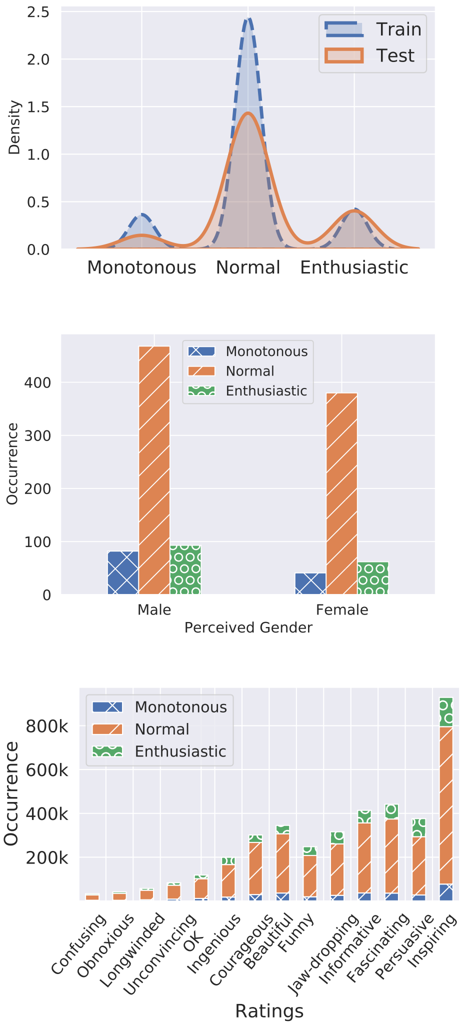

Figure 3: From top to bottom: Label distribution in our train-test split, among perceived gender, and ratings given by TED viewers. Top: Training set and testing set reflect the same imbalance of class labels. Center: Female speakers have proportionally fewer monotonous samples and more normal samples than male, but the same proportion of enthusiastic samples. Bottom: Samples labeled as enthusiastic have been mainly rated as fascinating, persuasive, and inspiring. They have rarely been rated negatively.

3.5 Data Statistics

In the following we will describe the relationship between the different enthusiasm levels and other attributes of the talks such as viewer ratings, number of views and comments, and perceived gender of the speakers. This metadata was obtained from a Kaggle competition[4] that collected data about TED talks until September 21st, 2017.

In Figure 3 (center), we can see that the enthusi-

---

[4] `https://www.kaggle.com/rounakbanik/ted-talks`

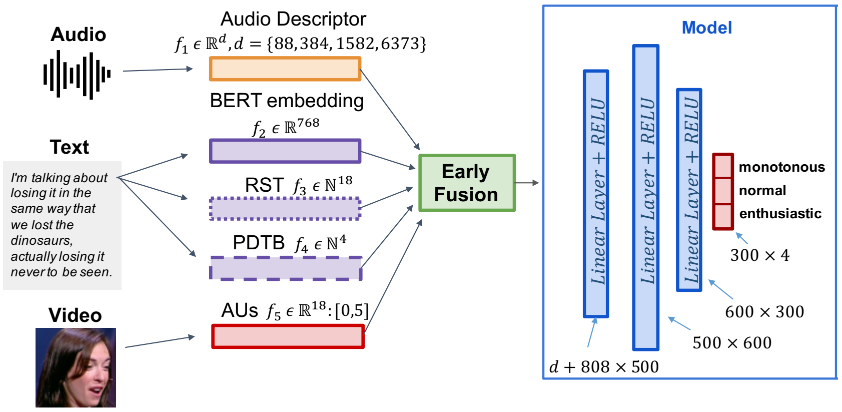

I'm talking about losing it in the same way that we lost the dinosaurs, actually losing it never to be seen.

Figure 4: An overview of our proposed multimodal dataset and model for predicting levels of enthusiasm using different features extracted from video, audio, and text.

asm levels are similarly distributed for both gender labels. We computed the Pearson's chi-squared test for independence to evaluate if there is a significant difference in enthusiasm level between gender. With a significance level of 5%, we obtained $p = 0.04$, meaning that gender of the speaker and enthusiasm level are dependent of each other. In Figure 3 (bottom), the label distribution among the different ratings that were given by viewers is shown. There are nine positive ratings (funny, beautiful, ingenious, courageous, informative, fascinating, inspiring, persuasive, jaw-dropping) and five negative ratings (longwinded, confusing, unconvincing, ok, obnoxious) which viewers could select. The ratings have been sorted by increasing number of enthusiastic samples. We can see that the negative ratings have the least number of enthusiastic samples. The ratings with the three highest numbers of enthusiastic samples are fascinating, persuasive and inspiring. We also performed two one-way ANOVAs to evaluate if the number of views and comments depend on the enthusiasm level. The resulting p-values were correspondingly $p = 0.3844$ and $p = 0.6892$ which means that views and comments are not influenced by the enthusiasm level of the speaker.

4 Computational Experiments

In the experiments of this paper, we aim to establish a performance baseline for the Entheos dataset using only the enthusiasm annotations. We train our model with different feature combinations to understand the role of different modalities in enthusiasm detection (see Figure 4). In the following we describe different features that were extracted and the model architecture that we used.

4.1 Features

Given the small number of labeled samples, instead of training an end-to-end model, we extract different features that will serve as input for our model. In the following we will describe the features used per modality.

Video: As enthusiasm is related to emotions, we extracted Facial Action Units (FAUs) which describe the intensity of muscular movements in the face based on the Facial Action Coding System (FACS) (Friesen and Ekman, 1978). We used OpenFace (Baltrusaitis et al., 2018) to obtain the intensity of 18 FAUs in a scale of 0-5. As FAUs vary over time, we computed the average and standard deviation for each AU and concatenated them in a feature of 36 dimensions per sample.

Acoustic: We extracted different audio features using OpenSMILE (Eyben et al., 2010), a toolbox that can extract over 27k features. We extracted four different feature combinations, which have been thoroughly studied in the speech community in affective computing tasks: a) eGEMAPS (88 attributes) (Eyben et al., 2015), b) Interspeech 2009 Emotion Challenge (384 attributes) (Schuller et al., 2009), c) Interspeech 2010 Paralinguistic Challenge (1582 attributes) (Schuller et al., 2010), and d) Interspeech 2013 Compare (6373 attributes) (Schuller et al., 2013). Each feature collection differs in the selection of features, functionals, and statistical measures. Examples of features covered are voice quality (jitter and shimmer), pitch (F0), energy, spectral, cepstral (MFCC) and voicing related low-level features (LLDs) as well as a few LLDs including logarithmic harmonic-to-noise ratio (HNR), spectral harmonicity, and psychoacoustic spectral sharpness.

Text: As a low-level feature, we used the bert-large-uncased model[5] to obtain word-embeddings on a sentence-level. For each sample we obtained a feature of 768 dimensions. As high-level features, we extracted two types of discourse relations: Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) and Penn Discourse Treebank (PDTB) (Miltsakaki et al., 2004; Prasad et al., 2008) relations. We used the RST parser from Wang et al. (2017) and the PDTB parser from Lin et al. (2014) for automated discourse relation annotation. Elementary discourse units (EDU) were obtained by using the method presented by Wang et al. (2018). For both parsers, samples can have more than one relation or none at all. The annotations were converted into a bag-of-words representation, obtaining features of 18 dimensions for RST and 4 for PDTB.

4.2 Model Architecture

Our model is composed by four fully connected layers with ReLU activation functions in between. We use concatenation to combine different features in the multimodal setting. Given our imbalanced dataset, we compute class weights, which represent the relation of samples per label and the total sample number. The class weights are then passed to our loss function (cross entropy loss) to give more weight to samples of the underrepresented classes. We use the Adam optimizer (Kingma and Ba, 2015) and during training, we perform early stopping to avoid overfitting. We train the model for a three class problem using all enthusiasm levels and also in a binary manner, combining "monotonous" and "normal" labels to the category called "non-enthusiastic".

5 Results and Evaluation

In this section, we present the performance results of our model using different combinations of features. We also evaluate the performance of the discourse parsers used and show statistical analysis of visual and acoustic features. All results of our statistical analysis are shown in the Appendix A.

5.1 Predicting Enthusiasm level

For each feature combination, we performed hyperparameter search with 10-fold cross-validation. The best hyperparameter combination was used to train the model with the entire training set. We evaluated the performance of the models on our test set. In Table 5, the weighted average results for precision, recall, and F1-score are shown. We see that in the unimodal case, BERT embeddings perform the best in the binary classification as well as in the three-class problem. Although PDTB has a higher F1-score in the binary case, RST performs better in the multi-class problem. Out of the different audio features, eGEMAPS performs slightly better than the other acoustic features. In the multi-class case, IS09 features are the best performing acoustic features.

When all features except AUs are combined, we reach the highest F1-score for the binary problem, improving the best unimodal performance by 0.08. We also see that combining both discourse relation features with eGEMAPS and BERT improves F1-score by 0.08 compared to using only one of them. In the multi-class problem, the best performing feature combination shows only a slight improvement of 0.04 compared to the unimodal case. Although manually annotating the entire resource was beyond the scope of this paper, we believe that it is necessary to understand the weaknesses and strengths of automatic parsers when used in spoken monologues. With current efforts being made in the field of creating discourse parsers for speech, the role of discourse parsers for enthusiasm detection will be better understood.

5.2 Evaluating the Effect of Discourse Features

We see in Table 5 that discourse relations help the model achieve the highest F1-score. However, we obtained the discourse relations by using discourse parsers that are trained on Wall Street Journal data[6], which is different from monologues.

To evaluate the performance of the parsers, 40 samples of our data were manually annotated with RST and PDTB relations by two annotators. The annotation protocol was approved by our institution's human subject research center. The inter-rater agreement was $\kappa = 0.88$. The accuracy of the RST parser on our data sample was 46.7 and for the PDTB parser 60.0. Although the accuracy of the parsers is low using our data, we have seen that concatenating both discourse relation features

---

[5] `https://huggingface.co/bert-large-uncased`

[6] `https://catalog.ldc.upenn.edu/LDC93S6A`

Features	Precision [B/M]	Recall [B/M]	F1-Score [B/M]
RST	0.67/0.55	0.64/0.47	0.65/0.50
PDTB	0.70/0.68	0.70/0.29	0.70/0.32
BERT	**0.77/0.66**	**0.81/0.56**	**0.75/0.60**
EGEMAPS	0.80/0.59	0.71/0.47	0.74/0.50
IS09	0.70/0.60	0.76/0.57	0.72/0.55
IS10	0.68/0.56	0.70/0.44	0.69/0.48
IS13	0.65/0.68	0.69/0.37	0.67/0.43
AU	0.67/0.77	0.76/0.50	0.70/0.57
BERT + PDTB	0.77/0.66	0.80/0.57	0.77/0.61
BERT + RST	0.79/0.66	0.81/0.56	0.77/0.60
EGEMAPS + BERT	0.81/0.62	0.60/0.58	0.64/0.59
EGEMAPS + PDTB	0.75/0.69	0.75/0.52	0.75/0.54
EGEMAPS + RST	0.77/0.71	0.7/0.61	0.73/0.64
EGEMAPS + BERT + PDTB	0.74/0.72	0.77/0.65	0.75/0.67
EGEMAPS + BERT + RST	0.77/0.71	0.81/0.58	0.75/0.61
EGEMAPS + RST + PDTB + BERT	**0.83/0.63**	**0.84/0.65**	**0.83/0.64**
EGEMAPS + RST + PDTB + BERT + AU	0.81/0.65	0.65/0.58	0.68/0.60

Table 5: Weighted average precision, recall, and F1-score for binary (**B**) and multiclass (**M**) classification. The same model architecture was used to train different feature combinations. BERT embeddings performed best in the unimodal setting. Combining acoustic with text features performed best in the multimodal setting.

to BERT and eGEMAPS improved our model's performance from an F1-score of 0.64 to 0.83 in the binary classification.

In Figure 5(a,b) we evaluated the relative occurrence of each enthusiasm level for RST and PDTB relations in ascending order of enthusiastic samples. In Figure 5a we can see that most samples do not have any discourse relation. However, there is a clear difference in the number of monotonous and enthusiastic samples that show *contingency*, as well as *temporal* relations. In Figure 5b we see that enthusiastic samples compared to monotonous samples use more elaboration, attribution, and joint relations. We performed the Pearson Chi Square test to verify our null hypotheses that discourse relations and enthusiasm level are independent from each other. We obtained a p-value of 0.0001 for PDTB and a p-value of 0.008 for RST, which permits us to reject our null hypothesis, meaning that the discourse relations influence the level of enthusiasm.

5.3 Investigating Visual Features

Given that AUs have not helped our model improve, we evaluated their dependence with our labels. We performed two separate one-way ANOVAs to evaluate the dependence of the mean of the 18 AUs with our labels, as well as the standard deviation of the AUs with our labels. The AUs with p-value < 0.05 are AU 12 (lip corner puller), AU 15 (lip corner depressor), AU 17 (chin raiser), and AU 26 (jaw drop). In Figure 5(c,d) the label distribution for the mean of AU 26 and standard deviation of AU 12 is shown. In both cases, we can observe that monotonous samples have more frequently a mean and standard deviation of zero compare to enthusiastic samples. We can also see in Figure 5d that enthusiastic samples have more frequently a standard deviation of AU 12 > 0.02.

5.4 Investigating Acoustic Features

We have seen that acoustic features are important in improving our model's performance. In this section we want to evaluate if pitch (F0) and loudness are independent from enthusiasm level. We perform a one-way ANOVA for the mean F0 per sample and its enthusiasm level, as well as for the mean loudness. Both p-values are < 0.05, meaning that the enthusiasm labels depend on the acoustic features. In Figure 5e, we can see that monotonous samples have a lower mean F0 than that of enthusiastic samples. We can also see in Figure 5f that monotonous samples have lower mean loudness than that of enthusiasm. These observations agree with the intuition that enthusiastic speakers speak louder and increase their pitch.

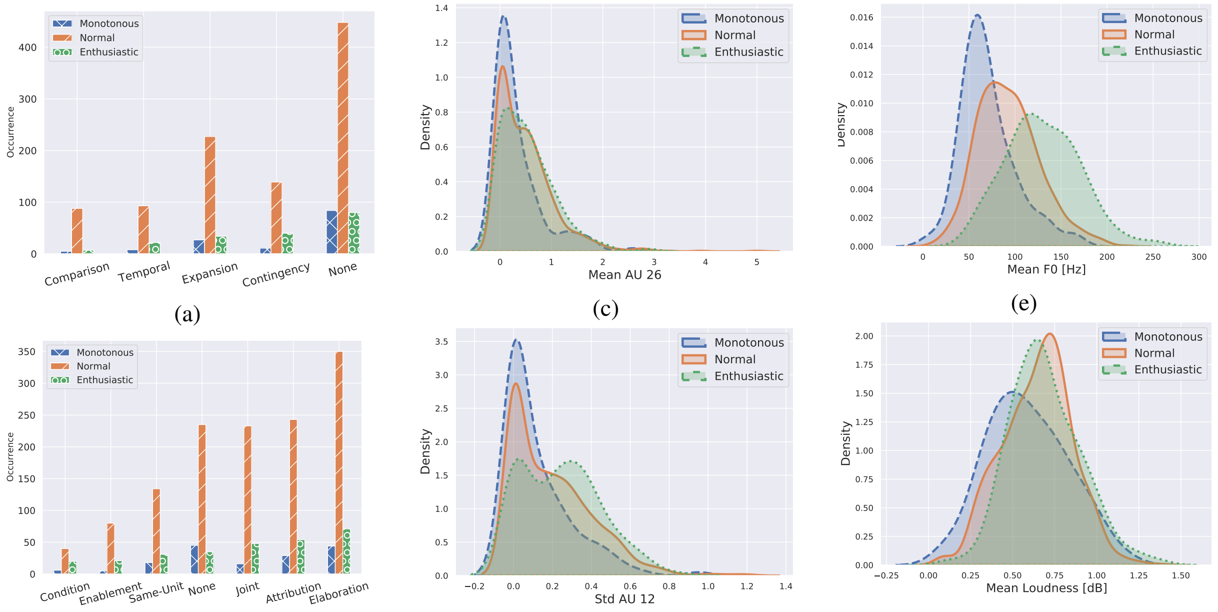

(a) (c) (e)

(b) (d) (f)

Figure 5: Label distribution of different enthusiasm levels in relation to discourse relations (a,b), acoustic features (c,d), and facial action units (e,f). In (a), most samples had no PDTB relation, however there is a visible difference between monotonous and enthusiastic samples in the occurrence of temporal and contingency relations. In (b), RST relations show that enthusiastic samples compared to monotonous samples use more elaboration, attribution, and joint relations. In (c), we can see that monotonous samples have more often low intensities for AU26 (jaw drop) than enthusiastic samples. (d) shows that monotonous samples have mostly very low standard deviation for AU12 (lip corner puller), but enthusiastic samples have higher standard deviation. In (e), we can see that enthusiastic samples have a higher mean F0 (pitch) compared to monotonous samples. (f) shows that monotonous speech tends to have lower mean loudness compared to enthusiastic speech.

6 Discussion and Conclusion

We present the first multimodal dataset for enthusiasm detection called Entheos[7] and discuss several baseline models. In addition, we present qualitative and quantitative analyses for studying and predicting enthusiasm using the three modalities of text, acoustic, and visual.

Our work has several limitations. TED talks are a very specific form of monologues as they are well-rehearsed and prepared. However, it is more likely that we can find enthusiastic speakers or well-structured sentences in TED talks. To understand enthusiastic behaviors in daily conversations, more data from other domains need to be annotated and studied. We hope that our annotation protocol will help other researchers in the future.

Further theoretical and empirical research is needed for better studying enthusiastic behaviors in general. The signals and definitions that we have worked with are not fine-grained or well-connected when exploring different modalities. Facial expressions and gestures can potentially provide meaningful contributions. Our experiments with facial action units were not successful. Our baseline approach used statistical information of each AU instead of the raw signal, which may dilute useful information. More experiments are needed to evaluate if and how AUs can help predict enthusiasm.

We hope our resources provide opportunities for multidisciplinary research in this area. Given the difficulties of annotating multimodal datasets in this domain, future work needs to investigate weakly supervised approaches for labeling multimodal data.

Acknowledgments

We thank TalkMeUp Inc. for supporting this research.

References

John Antonakis, Giovanna d'Adda, Roberto Weber, and Christian Zehnder. 2019. Just words? just

---

[7] `https://github.com/clviegas/Entheos-Dataset`

speeches? on the economic value of charismatic leadership. *NBER Rep. 4*.

Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. 2018. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 59–66. IEEE.

Edward M Bettencourt, Maxwell H Gillett, Meredith Damien Gall, and Ray E Hull. 1983. Effects of teacher enthusiasm training on student on-task behavior and achievement. *American educational research journal*, 20(3):435–450.

Alec Burmania, Srinivas Parthasarathy, and Carlos Busso. 2015. Increasing the reliability of crowdsourcing evaluations using online quality assessment. *IEEE Transactions on Affective Computing*, 7(4):374–388.

Jeroen PJ De Jong and Deanne N Den Hartog. 2007. How leaders influence employees' innovative behaviour. *European Journal of innovation management*.

Florian Eyben, Klaus R Scherer, Björn W Schuller, Johan Sundberg, Elisabeth André, Carlos Busso, Laurence Y Devillers, Julien Epps, Petri Laukka, Shrikanth S Narayanan, et al. 2015. The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing. *IEEE transactions on affective computing*, 7(2):190–202.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462.

E Friesen and Paul Ekman. 1978. Facial action coding system: a technique for the measurement of facial movement. *Palo Alto*, 3(2):5.

Jose Maria Garcia-Garcia, Victor MR Penichet, and Maria D Lozano. 2017. Emotion detection: a technology review. In *Proceedings of the XVIII international conference on human computer interaction*, pages 1–8.

François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. 2018. Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International Conference on Speech and Computer*, pages 198–208. Springer.

Michimasa Inaba, Fujio Toriumi, and Kenichiro Ishii. 2011. Automatic detection of "enthusiasm" in non-task-oriented dialogues using word co-occurrence. In *2011 IEEE Workshop on Affective Computational Intelligence (WACI)*, pages 1–7. IEEE.

Caroline F Keating. 2011. Channelling charisma through face and body status cues. *Social psychological dynamics*, pages 93–111.

Melanie M Keller, Thomas Goetz, Eva S Becker, Vinzenz Morger, and Lauren Hensley. 2014. Feeling and showing: A new conceptualization of dispositional teacher enthusiasm and its relation to students' interest. *Learning and Instruction*, 33:29–38.

Melanie M Keller, Anita Woolfolk Hoy, Thomas Goetz, and Anne C Frenzel. 2016. Teacher enthusiasm: Reviewing and redefining a complex construct. *Educational Psychology Review*, 28(4):743–769.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Yuanchao Li, Tianyu Zhao, and Tatsuya Kawahara. 2019. Improved end-to-end speech emotion recognition using self attention mechanism and multitask learning. In *Interspeech*, pages 2803–2807.

Tze Wei Liew, Su-Mae Tan, Teck Ming Tan, and Si Na Kew. 2020. Does speaker's voice enthusiasm affect social cue, cognitive load and transfer in multimedia learning? *Information and Learning Sciences*.

Tze Wei Liew, Nor Azan Mat Zin, and Noraidah Sahari. 2017. Exploring the affective, motivational and cognitive effects of pedagogical agent enthusiasm in a multimedia learning environment. *Human-centric Computing and Information Sciences*, 7(1):9.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Catherine Marechal, Dariusz Mikołajewski, Krzysztof Tyburek, Piotr Prokopowicz, Lamine Bougueroua, Corinne Ancourt, and Katarzyna Węgrzyn-Wolska. 2019. *Survey on AI-Based Multimodal Methods for Emotion Detection*, pages 307–324. Springer International Publishing, Cham.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2004. The Penn Discourse Treebank. *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*.

Shubhanshu Mishra and Jana Diesner. 2019. Capturing signals of enthusiasm and support towards social issues from twitter. In *Proceedings of the 5th International Workshop on Social Media World Sensors*, pages 19–24.

Oliver Niebuhr. 2020. Space fighters on stage-how the f1 and f2 vowel-space dimensions contribute to perceived speaker charisma. *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2020*, pages 265–277.

Oliver Niebuhr, Jana Voße, and Alexander Brem. 2016. What makes a charismatic speaker? a computer-based acoustic-prosodic analysis of steve jobs tone of voice. *Computers in Human Behavior*, 64:366–382.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn Discourse Treebank 2.0. *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

Elie Saad, Joost Broekens, Mark A Neerincx, and Koen V Hindriks. 2019. Enthusiastic robots make better contact. In *IROS*, pages 1094–1100.

Birgitta Sandberg. 2007. Enthusiasm in the development of radical innovations. *Creativity and Innovation Management*, 16(3):265–273.

Lisa M Schreiber, Gregory D Paul, and Lisa R Shibley. 2012. The development and test of the public speaking competence rubric. *Communication Education*, 61(3):205–233.

Björn Schuller, Stefan Steidl, and Anton Batliner. 2009. The interspeech 2009 emotion challenge. In *Tenth Annual Conference of the International Speech Communication Association*.

Björn Schuller, Stefan Steidl, Anton Batliner, Felix Burkhardt, Laurence Devillers, Christian Müller, and Shrikanth S Narayanan. 2010. The interspeech 2010 paralinguistic challenge. In *Eleventh Annual Conference of the International Speech Communication Association*.

Björn Schuller, Stefan Steidl, Anton Batliner, Alessandro Vinciarelli, Klaus Scherer, Fabien Ringeval, Mohamed Chetouani, Felix Weninger, Florian Eyben, Erik Marchi, et al. 2013. The interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism. In *Proceedings INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France*.

Armin Seyeditabari, Narges Tabari, and Wlodek Zadrozny. 2018. Emotion detection in text: a review. *arXiv preprint arXiv:1806.00674*.

Martin E Spencer. 1973. What is charisma? *The British Journal of Sociology*, 24(3):341–354.

Rode Snehal Sudhakar and Manjare Chandraprabha Anil. 2015. Analysis of speech features for emotion detection: a review. In *2015 International Conference on Computing Communication Control and Automation*, pages 661–664. IEEE.

Ryoko Tokuhisa and Ryuta Terashima. 2006. Relationship between utterances and "enthusiasm" in non-task-oriented conversational dialogue. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 161–167.

Carla Viegas, Albert Lu, Annabel Su, Carter Strear, Yi Xu, Albert Topdjian, Daniel Limon, and JJ Xu. 2020. Spark creativity by speaking enthusiastically: Communication training using an e-coach. In *Proceedings of the 2020 International Conference on Multimodal Interaction*, pages 764–765.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967, Brussels, Belgium. Association for Computational Linguistics.

Qin Zhang. 2014. Assessing the effects of instructor enthusiasm on classroom engagement, learning goal orientation, and academic self-efficacy. *Communication Teacher*, 28(1):44–56.

A Statistical Tests

In this section we present the results of the statistical tests performed to the facial action units and prosody features extracted from the entire dataset.

A.1 AU Statistical Tests

In order to understand which AU influence the enthusiasm level, we performed two different statistical tests: ANOVA for the three levels of enthusiasm (monotonous, normal, enthusiastic), and T-test for two levels of enthusiasm (enthusiastic, non-enthusiastic). In Table 6 on the left we can see the results of the ANOVA, analyzing the mean value of the different AUs per sample with the three levels of enthusiasm. All mean AUs that show p-value < 0.05 are highlighted. As AU 26 has the lowest p-value, the label distribution is shown in Figure 5c.

In Table 6 on the right we can see the results of the ANOVA, analyzing the standard deviation of the different AUs per sample with the three levels of enthusiasm. As AU 12 has the lowest p-value, the label distribution is shown in Figure 5d.

We also performed T-tests for the binary case using the labels enthusiastic and non-enthusiastic. Table 7 on the left shows that AU 17 (chin raiser)

Mean Action Unit	F-Statistic	P-value
AU 01	0.1393	0.87
AU 02	0.4541	0.6351
AU 04	1.415	0.2434
AU 05	0.385	0.6805
AU 06	1.1288	0.3238
AU 07	0.3578	0.6993
AU 09	2.4968	0.0828
AU 10	2.4397	0.0877
AU 12	**4.7553**	**0.0088**
AU 14	1.1253	0.3249
AU 15	**5.1991**	**0.0057**
AU 17	**4.672**	**0.0095**
AU 20	0.8012	0.449
AU 23	1.1192	0.3269
AU 25	0.9896	0.3721
AU 26	**6.0058**	**0.0025**
AU 45	1.0887	0.337

Std Action Unit	F-statistic	P-value
AU 01	**5.614114**	**0.003749**
AU 02	1.052148	0.349531
AU 04	0.905609	0.404591
AU 05	1.778094	0.169435
AU 06	**5.960989**	**0.002660**
AU 07	1.948772	0.142930
AU 09	**10.337395**	**0.000036**
AU 10	**8.383927**	**0.000243**
AU 12	**12.263390**	**0.000005**
AU 14	**5.483568**	**0.004266**
AU 15	**4.347689**	**0.013155**
AU 17	**12.201065**	**0.000006**
AU 20	**3.262334**	**0.038662**
AU 23	**8.411563**	**0.000237**
AU 25	**6.328837**	**0.001848**
AU 26	**11.989375**	**0.000007**
AU 45	**4.585977**	**0.010385**

Table 6: ANOVA significance test for three levels of enthusiasm and AU mean values on the left and standard deviation of AU on the right. AUs with lowest p-value are highlighted.

is the only AU with a p-value < 0.05. The distribution of the average values of AU 17 are shown in Figure 6(a). For comparison, the distribution of the average AU 02 (outer brow raiser) with highest p-value is shown in Figure 6(b). For both analysis, ANOVA and T-test, the differences of standard deviations among the enthusiasm levels are statistically significant for almost all AUs. This is not the case when analyzing the average values of AUs.

A.2 Prosody Statistical Tests

We performed statistical significance tests using the mean and standard deviation for F0 (pitch) and loudness. In Table 8(left), the ANOVA analysis results are shown and in Table 8(right), the results of the T-test. In both significance tests all variables have a p-value< 0.05, which means that all of them influence the enthusiasm level. Figure 6(c-f) show the label distribution for different values of the variables used in the significance test.

Mean Action Unit	F-Statistic	P-value
AU 01	-0.3995	0.6896
AU 02	0.0357	0.9715
AU 04	1.5205	0.1287
AU 05	0.4318	0.666
AU 06	-0.0535	0.9573
AU 07	0.7846	0.4328
AU 09	-1.8848	0.0597
AU 10	-0.9503	0.3422
AU 12	-1.1706	0.242
AU 14	0.5841	0.5592
AU 15	-0.9274	0.3539
AU 17	**-2.9922**	**0.0028**
AU 20	-1.2633	0.2067
AU 23	-1.4888	0.1368
AU 25	-0.586	0.558
AU 26	-1.2643	0.2064
AU 45	-0.3449	0.7303

Std Action Unit	F-statistic	P-value
AU 01	**-2.290794**	**0.022160**
AU 02	-1.451265	0.146985
AU 04	-0.909680	0.363186
AU 05	-1.067368	0.286035
AU 06	**-2.203459**	**0.027765**
AU 07	-1.574206	0.115721
AU 09	**-4.374609**	**0.000013**
AU 10	**-3.239400**	**0.001233**
AU 12	**-3.181258**	**0.001507**
AU 14	-1.460543	0.144420
AU 15	**-2.571532**	**0.010253**
AU 17	**-4.600027**	**0.000005**
AU 20	**-2.514758**	**0.012050**
AU 23	**-2.810554**	**0.005031**
AU 25	**-1.972713**	**0.048773**
AU 26	**-2.491479**	**0.012865**
AU 45	-1.378754	0.168245

Table 7: T-test for two levels of enthusiasm and AU mean values on the left and standard deviation of AU on the right. AUs with lowest p-value are highlighted.

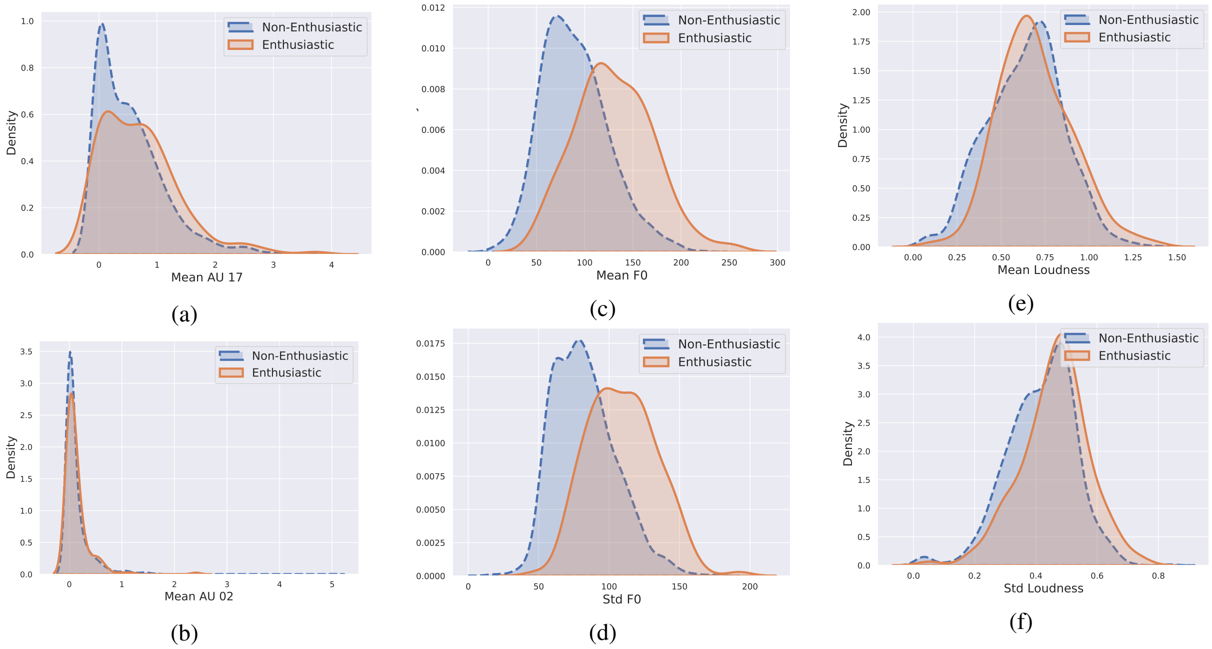

(a) (c) (e)

(b) (d) (f)

Figure 6: Label distribution of enthusiastic and non-enthusiastic samples in relation to the (a) mean AU 17 (p-value = 0.0028), mean AU 17 (p-value = 0.9715), (c) mean F0 (p-value = 0.0), (d) std F0 (p-value = 0.0), (e) mean loudness (p-value = 0.0034), (f) std loudness (p-value = 0.00).

	F-statistic	**P-value**
Mean F0	113.4309	0.0000
Mean Loudness	8.2467	0.0003
Std F0	146.9639	0.0000
Std Loudness	16.9411	0.0000

	F-statistic	**P-value**
Mean F0	-13.1960	0.0000
Mean Loudness	-2.9355	0.0034
Std F0	-13.9376	0.0000
Std Loudness	-4.508	0.0000

Table 8: Significance test for mean and standard deviation of F0 and loudness to evaluate the dependence with the different enthusiasm levels. Left: ANOVA significance test results three enthusiasm levels shows that all p-value< 0.05, which means that all variables influence the enthusiasm level. Right: T-test significance test for two levels of enthusiasm also shows that all variables influence the enthusiasm level.

Are Rotten Apples Edible? Challenging Commonsense Inference Ability with Exceptions

Nam Do
Brown University
Providence, RI, USA
nam_do@alumni.brown.edu

Ellie Pavlick
Brown University
Providence, RI, USA
ellie_pavlick@brown.edu

Abstract

Previous studies have argued that pre-trained language models encode commonsense relational knowledge (e.g. that *apples* are *edible*). However, simultaneous work has revealed that such models are often insensitive to context, even ignoring overt contextual cues such as negations. In this paper, we investigate whether masked language models (the BERT family) can move beyond naive associative biases (e.g., *apple* → *edible*) when the context warrants (e.g. ranking *inedible* higher when presented with the information that the *apple* is *rotten*). We introduce the WINOVENTI procedure, which adversarially exploits generic associations in masked language models to create model-specific Winograd-style entailment schemas. Using our constructed WINOVENTI challenges set of over $2,000$ schemas, we show that language models in the BERT family experience a steep drop in performance on prompts that require them to pick answers which require reasoning about context (e.g., from 89.8% to 18.4% for $\text{BERT}_{\text{LARGE}}$). We present evidence that language models exhibit different associative biases, suggesting a need for future work in developing and analyzing frameworks similar to WINOVENTI that are tuned to model-specific weaknesses.

1 Introduction

Humans exhibit commonsense knowledge through their ability to identify generics (e.g., that a dog has four legs) while still recognizing that exceptions to such rules are possible (e.g., that there are cases of three-legged dogs) (Greenberg, 2007), and that the probability of such exceptions can vary based on the context (e.g. "*the dog is running*" vs. "*the dog is hobbling*"). A prerequisite to comparing a machine's performance to human intelligence is, hence, the verification that machines can exhibit a sensitivity to context that would allow them to perform as well on cases that require reasoning about exceptions as on cases that require recalling generic associations.

Recent work (Petroni et al., 2019) has shown that large pretrained language models, in particular Masked Language Models (MLMs) such as BERT (Devlin et al., 2018) are competent at associating entities with their common characteristics. For example, $\text{BERT}_{\text{LARGE}}$ readily recalls *apple* → *edible* and *charcoal* → *hot*. However, as demonstrated by Ettinger (2020) and Kassner and Schütze (2019), BERT is insensitive to various overt contextual cues, notably negation. For example, given the context "*The shower is* ___," $\text{BERT}_{\text{LARGE}}$ predicts the words "*cold*", "*long*", and "*empty*", the same top 3 predictions it makes given the context "*The shower is not* ___". Such results suggest that while language models like BERT capture many commonsense patterns, such success might be limited to inferences involving common generalizations (appearing in an affirmative context, or using common lexical associations) and not those involving exceptions (appearing in a negative context, or requiring the models to choose less frequently associated lexical items).

In this paper, we investigate whether it is indeed the case that the "commonsense reasoning" exhibited by MLMs is limited to frequent generalizations as opposed to exception cases. We make three main contributions. First, we present the WINOVENTI procedure (§2) for identifying model-specific associative biases and adversarially building Winograd-style challenges (Levesque et al., 2012) to test models' commonsense inference ability. Second, we apply the WINOVENTI procedure to evaluate the commonsense inference performance of a suite of pre-trained MLMs (§3). We find that all the evaluated models experience dramatic performance drops on prompts that require the models to reason about exceptions to commonsense generaliza-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2061–2073
August 1–6, 2021.

WINOVENTI Procedure Summary

Step 1: Identifying Generic Associations
Sample In: { honey, tarantula }
Sample Out **($\text{BERT}_{\text{LARGE}}$)**: honey → good,
tarantula → poisonous,
tarantula → female
See Table 3

Step 2: Collecting Exceptions of Associations
Sample Out: (honey, good) → terrible
(tarantula, poisonous) → safe
(tarantula, female) → male

Step 3: Adversarial Filtering
Sample Out **($\text{BERT}_{\text{LARGE}}$)**: True, True, False
Rationale*: In the last example,* $\text{BERT}_{\text{LARGE}}$ *associates the exception characteristic (male) with the entity (tarantula) more strongly than the generic association (female).*

Step 4: Collecting Premises
Sample Prompt:
(1) The honey is [good/terrible].
(2) The tarantula is [poisonous/safe].
Sample Out: (1) After adding honey to my tea, it was (delicious/oversweet).
(2) Kim was (terrified/thrilled) when he asked her to hold the tarantula.

Step 5: Challenge Set Validation
Sample Out: `False`, `False`
Rationale*: In the first example, word association can be used to select the correct answer. In the second, the property-to-association mapping is not one-to-one, causing ambiguity.*

Table 1: Summary of the WINOVENTI pipeline for constructing common sense inference challenge set.

tions. Humans, in contrast, perform consistently well (∼90%) across inferences in our data. Third, we release our human-curated evaluation dataset of 2,176 sentence pairs probing inferences about commonsense generalizations and exceptions. All of our code and data are available at `http://http://commonsense-exception.github.io`.

2 WINOVENTI Procedure

2.1 Overview

The WINOVENTI procedure aims to produce Winograd-style sentence pairs to test models' ability to reason about common sense generics and exceptions. For example, a sentence pair might look like the following:

s_g: Zeke says that the apple is delicious. The apple is [MASK]. → edible > inedible

s_e: Zeke says that the apple is rotten. The apple is [MASK]. → inedible > edible

That is, we seek to generate pairs of sentences–which we call s_g and s_e, for generic and exception–which differ by a single word (w_g/w_e), such that that difference should lead to a change in the relative probability of other words (o_g/o_e) in the context. For example, the presence of w_e = *"rotten"* causes o_e = *"inedible"* to be more probable given s_e than given s_g. We seek to generate such pairs adversarially, meaning that, in the example above, we want to ensure that the model generally associates *"edible"* with *"apple"* and thus performing correctly on s_e requires using context to override this prior association.

Our five-step procedure is summarized in Table 1. First (§2.2), given a model and a target noun, we identify the model's *generic associations* (or just *generics*), i.e., the characteristics that the model tends to associate with the noun in a generic context. For example, given a target noun *"apple"*, we might identify *"edible"* as one such association. Second (§2.3), crowd workers are asked to provide contrasting characteristics (or *exceptions*) that could plausibly describe the noun. For example, workers might provide *"inedible"* or *"plastic"* as characteristics that contrast with *"edible"* in the context of *"apple"*. Third (§2.4), we perform an adversarial post-processing step in which we filter out worker-provided exceptions that the model associates with the target more strongly than the original generic. That is, if a worker provides the characteristic *"poison"* and it turns out that the model associates *"poison"* with *"apple"* more strongly than it associates *"edible"* with *"apple"*, we would filter *"poison"* out of our list of exceptions. Fourth (§2.5), we crowdsource premise pairs that would ideally effect the relative probability of the generic characteristic vs. the exception characteristic, such as those shown in s_g and s_e above. Finally (§2.6), the schemas are validated by human annotators, after which we filter out annotations that are trivial or ambiguous. Using this five-step procedure, we construct the $\text{WINOVENTI}_{\text{BERT LARGE}}$ challenge set of 2,176 sentence pairs.[1]

---

[1] Available at: `http://commonsense-exception.github.io`.

Prompt pairs	**Generic/Exception Outcomes**
s_g = Regina screamed when she picked up the pan. The pan is _.	**hot** / cold
s_e = Regina shivered when she picked up the pan. The pan is _.	hot / **cold**
s_g = Tonight Mike's pets would be happy. The pet food is _.	**available** / unavailable
s_e = Tonight Mike's pets would be hungry. The pet food is _.	available / **unavailable**

Table 2: Examples of our schemas. Each *prompt* contains a *premise* with an underlined *special word* (first sentence) and an *outcome sentence*. s_g = generic prompt. s_e = exception prompt. All prompts and exception outcomes are crowdsourced, whereas *generic outcomes* are generic associations identified in some MLM ($\text{BERT}_{\text{LARGE}}$, in our paper). Correct answers are in bold.

We draw our target nouns from the THINGS dataset (Hebart et al., 2019) which consists of 1,854 concepts from 27 semantic categories. All of our crowdsourcing tasks are run on Surge (`surgehq.ai`), a high-quality crowdsourcing platform similar to Amazon Mechanical Turk. For the exception and premise generation stages (§2.3 and §2.5), we recruited 100 workers through a qualification task that required the workers to walk through the entire WINOVENTI pipeline. The validation annotation (§2.6) could be performed by anyone on the Surge platform. The pay per response for the crowdsourcing stages is as follows, chosen based on difficulty and time taken to complete the task: (1) exception generation (§2.3): \$0.1, (2) premise generation (§2.5): \$0.3, (3) challenge set validation (§2.6): \$0.2, and (4) human performance: \$0.03.

2.2 Step 1: Identifying Generic Associations

Given a set of target nouns, our first step is to find a set of generic associations which models associate with the noun regardless of context. To do this, we focus on the widely-used $\text{BERT}_{\text{LARGE}}$ model (Devlin et al., 2018), specifically the HuggingFace cased, whole word masking implementation (24-layer, 1024 hidden dimension, 16 attention heads, 336M parameters). We base our procedure off of the insight from Ettinger (2020), which demonstrated BERT's insensitivity to negation. For example, given the contexts "*A robin is (not) a [MASK],*" BERT's top two predictions would share {*bird, robin*} in common, or given the contexts "*A daisy is (not) a [MASK],*" BERT's top three predictions would share {*daisy, rose, flower*} in common. Our experiments show that this behavior holds consistent for other MLMs such as RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), Longformer (Beltagy et al., 2020), SqueezeBERT (Iandola et al., 2020), and MobileBERT (Sun et al., 2020). Thus, to identify context-invariant associations, we feed in both affirmative and negated templates[2] of the form "*The* `[ENTITY]` *is (not) [MASK]*". We then define the models *generic associations* to be the set of words that are in common between the top-k predictions in the affirmative context and the top-k predictions in the negative context. We experiment with $k \in \{1, 3, 5, 8\}$. Table 3 shows generic associations generated for $\text{BERT}_{\text{LARGE}}$ in this way for different values of k.

k	**Generic Associations for $\text{BERT}_{\text{LARGE}}$**
1	desk → empty, tarantula → poisonous, couch → comfortable, syrup → sweet, honey → sweet, compass → accurate
3	desk → there, tarantula → edible, couch → empty, syrup → bitter, honey → good, compass → true
5	desk → full, tarantula → {female, male}, couch → warm, syrup → edible honey → edible, compass → right
8	desk → {mine, clean}, tarantula → small couch → {clean, big}, syrup → orange, honey → {delicious, there, honey} compass → wrong

Table 3: Examples of $\text{BERT}_{\text{LARGE}}$ generic associations for different values of k. Generic associations are cumulative across different k values (e.g., in the end, we have *compass* → { accurate, true, right, wrong }).

How similar are associations across models? Since our dataset is constructed in particular to be adversarial against $\text{BERT}_{\text{LARGE}}$, it is relevant to ask whether such a dataset will be equally adversarial against all models in the BERT family. To quantify whether models in the BERT family (described in §3.2) differ in the generic associations

---

[2] We use a simple set S of two templates that differ in the ending punctuation (S = {"*The* `[ENTITY]` *is [MASK].''*, *``The* `[ENTITY]` *is [MASK],''*}).

Model name	k = 1	k = 3	k = 5
$\text{BERT}_{\text{BASE}}$	0.59	0.26	0.17
$\text{BERT}_{\text{LARGE}}$	1	1	1
$\text{RoBERTa}_{\text{SMALL}}$	0.61	0.2	0.11
$\text{RoBERTa}_{\text{LARGE}}$	0.57	0.19	0.12
DistilRoBERTa	0.53	0.15	0.08
DistilBERT	0.57	0.18	0.12
SqueezeBERT	0.6	0.23	0.13
MobileBERT	0.65	0.28	0.16
$\text{ALBERT}_{\text{BASE}}$	0.42	0.07	0.04
$\text{ALBERT}_{\text{LARGE}}$	0.51	0.1	0.07
$\text{ALBERT}_{\text{XLARGE}}$	0.59	0.18	0.08
$\text{ALBERT}_{\text{XXLARGE}}$	0.48	0.13	0.07

Table 4: Jaccard similarity of Generic Associations made by models with respect to $\text{BERT}_{\text{LARGE}}$. The closer to 1, the more similar.

identified using this procedure, we compute the Jaccard similarity–i.e., $\frac{|A \cap B|}{|A \cup B|}$, with A and B being the set of associations made by two models. Table 4 shows the results for $k = 1, 3$, and 5. $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ have the highest similarity (with a mean Jaccard score of 0.72). Most other models only share roughly half of the generic associations with another model. As k increases, generic associations start to differ across models (signified by a decrease in the Jaccard similarity score). This signifies that models differ significantly in the generic associations that they make.

Qualitatively, associations made by BERT-family, RoBERTa-family and other models differ from one another. While the evaluated RoBERTa-family models display a slight tendency to associate the words *"empty"*, *"broken"*, *"dead"* with many nouns, the BERT-family models tend to make other associations such as *"used"*, and *"edible"*. ALBERT models, on the other hand, make quite different associations from all other models, such as *"adjustable"*, *"gone"*, *"a"*, and *"covered"* (with a full 41% of the associations made by $\text{ALBERT}_{\text{XXLARGE}}$ being *"adjustable"*).

As we base this study on the associations made by $\text{BERT}_{\text{LARGE}}$, future work is to be done to extract the different associations made by other MLMs to pose a harder and richer set of challenges for those models.

2.3 Step 2: Crowdsourcing Exceptions to Generic Associations

Given the identified associative biases, we then use crowdsourcing to find alternative characteristics that can be true of the noun, but are perhaps less stereotypically associated (e.g., *"apples"* can be *"inedible"* if, for example, they are made of plastic, or are poisoned). Workers are given statements of the form "*The* `[NOUN]` *is* `[PROPERTY]`" where `[PROPERTY]` is one of the generic associations collected as described above (e.g., "*The apple is* *edible*"), and then is asked to provide up to three adjectives that would describe `[NOUN]` in the case where it is *not* `[PROPERTY]` (e.g., in the case that the "*apple*" is not "*edible*"). To increase the quality of tasks presented to workers in later stages, we also ask workers in this stage whether or not a presented statement makes sense, and filter out any sentences which workers flag as nonsensical. Of the noun-property associations generated in our first step, 10.45% are filtered out in this step.

The model-generated associative pairs can be noisy. Particularly, we note that as we increase k, the model becomes more likely to generate nonsense pairs (e.g., *glass* → *glass*, *dart* → *legal*, *triangle* → *circular*), and that the stereotypical strength of association decreases (as shown in Table 3). Thus, to increase the quality of the final challenge set and minimize workers' confusion, our pipeline uses two main criteria to select which association pairs to present to workers. First, associations are selected in an increasing order of k, meaning we would include all associations at $k = 1$ first, and then those at $k = 3$, and so on. Second, the templated statements presented to workers are ranked according to perplexity scores assigned to them by a unidirectional language model. For each k of interest, we identify the inflection point in the perplexity score, and only retain samples for which the perplexity is below this point. At the end when we have gathered more samples than we plan to post, we perform a final round of inflection point identification and retention. This leaves us with 1990 entity-association pairs.

To construct $\text{WINOVENTI}_{\text{BERT LARGE}}$, we selected 1990 entity-bias pairs to present to workers (331 pairs were of 1-associative biases, 780 of 3-associative biases, 825 of 5-associative biases, and 54 of 8-associative biases). 28 workers participated in this stage, and for each task workers were paid \$0.1. At the end, we received $2,994$ valid exception annotations from crowd workers.

2.4 Step 3: Adversarial Filtering

The goal of adversarial filtering is to make sure that a model clearly favors the generic association with a target noun (identified by an MLM) over an exception association (crowdsourced) without any additionally introduced context. A triplet of (*Target Noun*, *Generic Association*, *Exception Association*) passes our adversarial filtering stage if the probability that the model associates the *Generic Association* with the *Target Noun* (through statements of template "*The* `[NOUN]` *is* ___") is higher than that which the model associates with the *Exception Association*.

This filtering is adversarial in that, by making sure that a model innately favors one association (e.g., *edible*) with a target noun (e.g., *apple*) over another (e.g., *inedible*), the model has to demonstrate a strong enough sensitivity to context to select the other association over the one that it innately prefers (e.g., when presented with the information that the *apple* is *rotten*).

In the construction of WinoVenti$_{\text{BERT LARGE}}$, after adversarially filtering using BERT$_{\text{LARGE}}$, we retained $2,745$ (*Target Noun*, *Generic Association*, *Exception Association*) triplets. Some examples of triplets that were filtered out (in the same format) are: (*stair*, *long*, *short*), or (*mug*, *full*, *empty*).

2.5 Step 4: Crowdsourcing Premises

As a result of Step 3, we have a set of (noun, generic associations, exception associations) triplets that have met our adversarial filtering criteria. We then ask workers to create a minimal pair of *premise* sentences (s_g/s_e) differing by exactly one word (w_g/w_e) which differ in whether the *generic outcome* (o_g) or the *exception outcome* (o_e) is the most probable continuation. For example, given the triplet (*"apple"*, *"edible"*, *"inedible"*), our sentence pair might be $s_g =$ *"the apple is <u>sweet</u>* and $s_e =$ *"the apple is <u>rotten</u>*. To minimize the cognitive load on crowdworkers while still communicating these requirements, we provided examples of two good and two bad premise pairs with a brief explanation each of why each was good/bad, covering all the requirements above.[3] We also encouraged workers to be creative and provide premises that have diverse sentence structures. In total, $2,745$ premise pairs were collected. Table 5 shows several examples of contexts generated in this way.

Crowdsourced Premises
Given: The mail is [anonymous/identifiable]. **Annotation:** I received a letter in the mail from a (stranger/friend).
Given: The pill is [safe/unsafe]. **Annotation:** You will feel (better/worse) if you take the pill.
Given: The paper bag is [empty/full]. **Annotation:** I took a (folded/heavy) bag to the store.
Given: The timer is [accurate/inaccurate]. **Annotation:** Jim was (early/late) to work of his timer.
Given: The bed is [comfortable/uncomfortable]. **Annotation:** Lola slept [soundly/miserably].

Table 5: Premise annotations collected given a target noun and the identified generic/exception associations, combined into a natural language sentence of format "*The* ___ *is* ___" presented to workers.

2.6 Step 5: Challenge Set Validation

We evaluate each sentence pair using two criteria. First, is it the case that the contexts differentiate the probabilities of o_g and o_e such that o_g only makes sense given s_g and o_e only makes sense given s_e. E.g., given "*Matthew says that the apple is rotten. The apple is [MASK]*", can *edible* be a sensible answer)? Second, we ensure that the special words are not synonymous with the outcomes, i.e., that w_g and o_g are not synonyms, nor are w_e and o_e. If the majority of workers (out of three) judge the sentence pairs to pass the above criteria, the pair is deemed valid. Criterion 1 is to ensure that the outcomes are unambiguous to humans, while criterion 2 is to ensure that synonymy can not be used to trivially find the correct answer. For example, criterion 1 filtered out prompts such as "*Mary made popsicles out of vitamins. The popsicle is [MASK].*" where both the choices (*edible / nutritious*) could apply. Criterion 2 filtered out prompts like "*The doctor used a <u>loaded</u> test tube. The test tube is [MASK].*" where the correct answer (*filled/hollow*) could easily be selected using word association. Of the $2,678$ prompt pairs posted for evaluation, 502 were filtered out for failing at least one of the two aforementioned criteria, leaving us with the

---

[3] Our exact instructions and examples given to workers are provided in Appendix A.2

final challenge set of $2,176$ prompt pairs ($4,352$ prompts in total).

2.7 WinoVenti Challenge Set

To summarize, the final WINOVENTI$_{\text{BERT LARGE}}$ challenge set consists of $2,176$ prompt pairs ($4,352$ challenge prompts in total) about 978 distinct entities. Each entity has at most 5 generic associations, for a total of 186 distinct generic associations identified by BERT$_{\text{LARGE}}$ across all entities. The length of premises are on average 8 words. Using SpaCy's Part of Speech tagger, we identified that the special words (w_g and w_e) across different premises are predominantly adjectives (apprx. 34%), nouns (apprx. 30.3%), and verbs (apprx. 22.4%), with the presence of other parts of speech such as adverbs, adpositions, numbers, determiners, proper nouns, or pronouns. Approximately 54.5% of the premises have the special words in the last third, 28.4% in the middle third, and the rest 16.9% in the first third of the premise. The challenge set is available at `http://commonsense-exception.github.io/`.

3 Experiments

3.1 Task Definition

The task is defined as: given a pair of sentences (s_g, s_e) with the format in Table 2, and the pair of single-word candidate outcomes (o_g, o_e), does the model correctly rank o_g as more probable than o_e given s_g (general test), and, symmetrically, rank o_e as more probable than o_g given s_e (exception test). An MLM's performance is reported as the percentage of sentences for which the language model successfully gives the correct answer a higher probability of filling in the blank than the incorrect answer. Each model is evaluated on three different subsets of the full WINOVENTI$_{\text{BERT LARGE}}$ dataset, as follows: **All** refers simply to the entire WINOVENTI$_{\text{BERT LARGE}}$ challenge set (which is only adversarially filtered using BERT$_{\text{LARGE}}$); **Individual** refers to a model-specific subset of WINOVENTI$_{\text{BERT LARGE}}$, specifically those pairs which result after additionally model-specific adversarial filtering; **Intersection** refers to the set of 188 prompts (94 each for the general test and the exception test) that result when we take the intersection of each of the model-specific subsets generated described in Individual. Note that both Individual and Intersection reflect the performance of models in an adversarial setting, and we do not expect these subsets to show meaningfully different results. We include Intersection simply so that we can compare all models on a fixed test set in an apples-to-apples setting, since the Individual subsets will vary from one model to the next.

3.2 Models

We study the performance of the following pretrained models (HuggingFace implementation) on our WINOVENTI$_{\text{BERT LARGE}}$ challenge set:

BERT-family We used BERT$_{\text{BASE}}$ (cased) and BERT$_{\text{LARGE}}$ (cased, trained with a whole word masked language modeling objective) models (Devlin et al., 2018). We also evaluated DistilBERT (Sanh et al., 2019) (cased), which has 40% less parameters than BERT while still performing almost as good as the original model. MobileBERT (Sun et al., 2020), similarly seeking to compress and accelerate the BERT model, is also included in our experiments (uncased). Additionally, our experiments also evaluated SqueezeBERT (Iandola et al., 2020) (uncased), which has a similar bidirectional transformer architecture like BERT, but uses grouped convolutions instead of certain fully-connected layers.

RoBERTa-family Our experiments also evaluate RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019) versions that were trained on a masked modeling objective. RoBERTa build on BERT, differing in hyperparameter choices and pre-training objective. We also used DistilRoBERTa (Sanh et al., 2019), which follows the same training and distillation process as DistilBERT, but based on RoBERTa instead.

ALBERT-family We additionally evaluated ALBERT$_{\{\text{BASE, LARGE, XLARGE, XXLARGE}\}}$ models (Lan et al., 2019). Built off of BERT, ALBERT is designed to reduce the number of parameters and perform better on downstream tasks with multi-sentence inputs.

3.3 Finetuning

To evaluate how the models perform after being fine-tuned on datasets that contain exceptions in similar sentence structures to our challenge set, we fine-tuned a subset of the models in question (BERT$_{\text{BASE}}$, RoBERTa$_{\text{BASE}}$, DistilRoBERTa, SqueezeBERT, MobileBERT, and ALBERT$_{\text{BASE}}$) on two subsets of our dataset, the *half* and the *full* exception train sets. The *half*

	All		**Individual**			**Intersection**	
Model name	**Generic**	**Exceptn**	**Generic**	**Exceptn**	**N**	**Generic**	**Exceptn**
BERT$_{\text{BASE}}$	83.0	23.1	89.3	16.7	1871	96.0	8.3
BERT$_{\text{LARGE}}$	89.8	18.4	89.8	18.4	2176	97.4	11.4
RoBERTa$_{\text{BASE}}$	79.5	27.2	87.3	18.8	1801	95.6	12.7
RoBERTa$_{\text{LARGE}}$	81.3	29.2	91.1	18.9	994	95.6	17.5
DistilRoBERTa	78.3	26.5	91.1	13.1	948	93.9	14.0
DistilBERT	85.8	19.4	89.8	15.2	1628	94.8	12.7
SqueezeBERT	85.5	21.1	92.1	13.6	1823	97.4	7.9
MobileBERT	82.7	23.0	89.8	15.6	1856	96.9	7.0
ALBERT$_{\text{BASE}}$	77.4	28.0	87.2	18.4	1525	94.3	11.0
ALBERT$_{\text{LARGE}}$	79.6	26.8	87.6	18.9	1324	93.9	12.2
ALBERT$_{\text{XLARGE}}$	80.7	31.9	89.0	23.5	777	96.1	13.1
ALBERT$_{\text{XXLARGE}}$	77.7	32.4	80.0	32.4	959	83.8	31.9
Human performance	91.1	90.2	-	-	-	-	-

Table 6: Models' performance on our *generic* and *exception* prompts. Generic tests evaluate whether, given a generic prompt, the model would rank the generic outcome o_g higher than the exception o_e). Exceptn tests similarly check if the exception outcome o_e is ranked higher than the generic o_g given an exception context.

exception set is created by selecting half of the WinoVenti$_{\text{BERT LARGE}}$ challenge set (2176 out of 4352 schemas). 50% of the training schemas (1088 out of 2176) are selected to be generic schemas, and the rest are exception schemas. With this configuration, models are fine-tuned on both generic and exception schemas, and are sequentially evaluated on unseen challenges (with a similar distribution of 50% generic schemas and 50% exception).

With the *full exception* train set, models are fine-tuned exclusively on exception schemas. From the set of exception challenges (2176 schemas), we performed a 80-20 train-test split to select the *full exception* train set. To evaluate models trained on the *full exception train set*, in addition to evaluating the model accuracy on the held out exceptions (20% - 435 of the 2176 exception schemas), we also evaluate the fine-tuned models' on (1) all the generic challenges (2176 schemas), and (2) on a test set similar to the above where half of the schemas are generic challenges and another half are unseen exception challenges (870 schemas). The different test scenarios help us understand how finetuning on exception challenges influences models' performance on not only unseen exception challenges, but also on generic challenges.

3.4 Results and Analysis

Table 6 shows each model's performance across each data subset (All, Individual, and Intersection) broken down by task (i.e., generic vs. exception test). Across the board, models perform significantly better on the generic tests than on the exception tests (where accuracies are well below the random baseline of 50%). This provides strong evidence that models do not truly encode "common sense", but rather simply recall generic associations in a context-agnostic manner.

Looking closely at the results on the All subset, we see that models' performance on the generic test is overall lower and the performance drop on the *exception test* is less dramatic, compared to the results on the Individual and Intersection subsets. This trend is expected, since, after adversarial filtering, each model is only evaluated on prompts where the model is inherently (before being introduced to any additional context) skewed towards choosing the generic association as a description of the target noun. Even so, the numbers on this All set are informative: they emphasize that models' apparent success at recalling "commonsense" associations is likely largely driven by artifact. That is, when assessed on a set of common sense inferences that don't necessarily involve words from the top of a model's distribution in a given context, performance is quite poor. On inspection, we see that, for models outside the BERT family, the poor performance is often attributable to low probabilities for both the generic and exception outcomes (o_g and o_e) in both contexts, meaning the difference between the probabilities is often small. In other words, these models don't encode the same generic associations that $\text{BERT}_{\text{LARGE}}$ encodes and, moreover, don't encode much difference at all between o_g and o_e.

Error Analysis Looking closely at specific prompts on which models succeed and fail reveals some interesting trends. For each model, we look at the top 20 sentences from the *exception test* on which the model exhibited the largest errors, where the size of the error is measured by the difference in the probability that the model assigns to the incorrect outcome (in this case o_g) compared to the correct outcome (o_e). For $\text{BERT}_{\text{LARGE}}$, we find that 55% of these top 20 failures involve strong generic associations (i.e., those generated with $k = 1$). A full 65% are cases when o_g is "*empty*", suggesting that $\text{BERT}_{\text{LARGE}}$ prefers to assign high probability to this word across all contexts (see §2.2). In contrast, looking at the top 20 sentences on which $\text{BERT}_{\text{LARGE}}$ performed best, we learn that only one involves $o_g =$"*empty*", and only four involve strong generic associations ($k = 1$).

Performing similar analysis for the other BERT-family models ($\text{BERT}_{\text{SMALL}}$, DistilBERT, MobileBERT, SqueezeBERT), we see that the majority of the successes involve non-associative noun-characteristic pairs (i.e., pairs where the characteristic is not identified as a generic association with the noun by our procedure described in Section 2.2). For $\text{BERT}_{\text{SMALL}}$ and DistilBERT, 40% of their 20 most successful cases involved nouns for which the models did not encode any generic associations. For SqueezeBERT and MobileBERT, it is 55% and 60%, respectively. This may signify stronger a relationship between a model **not** identifying a generic association with a target noun and that model being sensitive to a change in context about that target noun.

Fine-tuning Analysis The results of our fine-tuning experiments are shown in Figure 1 (with additional results in §A.1). We see that, in general, fine-tuning models on a dataset that contains exceptions (where the challenge format remains the same between the train and test sets) can increase the performance on unseen exceptions, but does so at the expense of performance on generic prompts. Specifically, when we train on a mix of generic and exception schemas (our *half exception set*), the model improves only slightly in performance on exceptions, and converges to the same trend as the un-finetuned model: i.e., performance on generics far exceeds that on exceptions. In contrast, when we train on only exception schemas (our *full exception set*), the performance on unseen exception challenges increases faster and more significantly, but this increase is at the expense of the rapid decrease of performance on generic challenges.

This poor performance on exceptions (at the expense of their performance on generics), suggests that the conceptual associations encoded by MLM models is fairly shallow: even with finetuning, the models are not able to differentiate these types of context-specific associations in a way that allows them to perform well on both types of inferences simultaneously. Future work is needed to develop models with different architectures or loss functions that might be capable of encoding more nuanced conditional associations.

4 Related Work

Ettinger et al. (2017) bring up the fundamental problem of NLP models ignoring rare language phenomena, as they typically rely on independently and identically distributed probably-approximately-correct model of learning, and as they often use overly simplistic loss functions. Complementary to our project, Ettinger et al. (2017) encourage robust error analysis of NLP systems through developing challenges that are based on linguistic phenomena, and that have a low barrier to entry.

Common sense and probing. NLP has been interested in encoding commonsense relations for a long time (Liu and Singh, 2004). Recent work has shown how pre-trained LMs exhibit common sense knowledge even before fine-tuning (Petroni et al., 2019), and that they can be built and used to mine more commonsense information (Bosselut et al., 2016; Davison et al., 2019). While this signifies how LMs encode some common sense and prototypical properties of nouns (Weir et al., 2020), many researchers are pointing out these models' insensitivity to context (Ettinger, 2020; Ravichander et al., 2020), which is antithetical to common sense.

Challenge Sets Many existing challenge sets have provided concrete frameworks to evaluate models inference ability, through coreference resolution (notably Winograd Schema Challenge - WSC (Levesque et al., 2012)) or pronoun resolution (notably in PDP (Morgenstern et al., 2016)). In this work, similar to the Winograd Schemas (Levesque

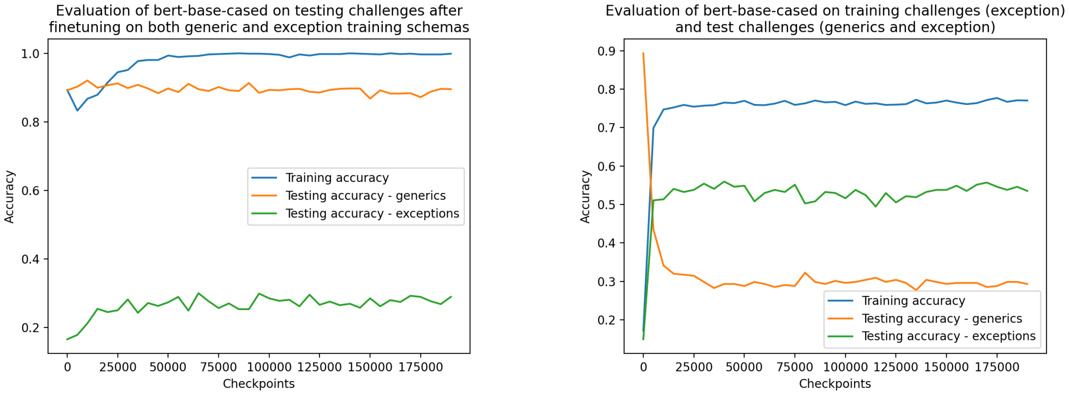

Figure 1: **Left**: Fine-tuning $\text{BERT}_{\text{BASE}}$ on a dataset that contains both generics and exceptions results in a minimal increase in performance on exceptions. **Right**: Fine-tuning $\text{BERT}_{\text{BASE}}$ on a dataset containing only exceptions results in an increase in performance on exceptions, at the expense of the accuracy on generics.

et al., 2012), we also generate pairs with a similalr structure (pairs of premises that differ in one word that would determine the answer). However, while Winograd schemas are model-agnostic, our approach factors in models' behavior in the design of schemas in order to guarantee models' bias towards one specific answer (for each prompt).

Sakaguchi et al. (2020) build the WINOGRANDE adversarial challenge set through using language models to detect and filter out schemas with language-based biases that would trivialize the task of picking the correct answer. WINOGRANDE aims to minimize the chance of models getting the right answers for the wrong reasons (through leveraging simple lexical associations that are annotation artifacts by human annotators). Our key motivation, meanwhile, is to adversarially leverage the biases that models associate with entities to "trick" them into choosing incorrect answers. Our work uses adversarially constructed test sets to expose heuristics that models use. This technique has been used widely in probing/analysis work, e.g., (Glockner et al., 2018; Naik et al., 2018; Jia and Liang, 2017; Nie et al., 2019; McCoy et al., 2019). The idea of improving models performance on "exceptions" to "generalizations" also shares much in common with work on bias and fairness in NLP (Rudinger et al., 2018; Zhao et al., 2018, 2019).

Gardner et al. (2020) propose the development of *contrast sets*, which can be developed by manually perturbing existing datasets in small but meaningful ways that would change the gold label. Our work, in contrast, factor in models' insensitive associations into the construction of challenges in addition to a slight change in context that is leveraged by contrast sets. Kaushik et al. (2019) similarly propose using *counterfactually-augmented data* to make models more robust against spurious associations. Our work adds to this work by demonstrating that fine-tuning on exception challenges can increase the performance of models on tail cases at the expense of the performance on generic prompts.

5 Conclusion

We present the WINOVENTI procedure, which adversarially exploits generic associations in masked language models to create model-specific Winograd-style schemas. Using our constructed $\text{WINOVENTI}_{\text{BERT LARGE}}$ challenge set, we test whether MLMs can move beyond their naive associations to select the more likely outcomes depending on the input context. We find a steep drop in models' performance on our challenges that require a sensitivity to context. We present evidence that generic associations differ from one model to another, highlighting the need for other model-specific challenge sets that are tuned to associative biases of models other than $\text{BERT}_{\text{LARGE}}$, and to (2) develop and analyze frameworks like WINOVENTI.

Acknowledgments

We thank Charlie Lovering, and the members and friends of the Brown University LUNAR Lab for all the invaluable conversations, proofreading, and feedback. We thank Edwin Chen (Surge) for their support with our crowdsourcing tasks.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Antoine Bosselut, Jianfu Chen, David Warren, Hannaneh Hajishirzi, and Yejin Choi. 2016. Learning prototypical event structure from photo albums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1769–1779, Berlin, Germany. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. 2017. Towards linguistically generalizable nlp systems: A workshop and shared task. *arXiv preprint arXiv:1711.01505*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models' local decision boundaries via contrast sets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1307–1323.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655. Association for Computational Linguistics.

Yael Greenberg. 2007. Exceptions to generics: Where vagueness, context dependence and modality interact. *Journal of Semantics*, 24(2):131–167.

Martin N Hebart, Adam H Dickter, Alexis Kidder, Wan Y Kwok, Anna Corriveau, Caitlin Van Wicklin, and Chris I Baker. 2019. Things: A database of 1,854 object concepts and more than 26,000 naturalistic object images. *PloS one*, 14(10):e0223792.

Forrest N Iandola, Albert E Shaw, Ravi Krishna, and Kurt W Keutzer. 2020. Squeezebert: What can computer vision teach nlp about efficient neural networks? *arXiv preprint arXiv:2006.11316*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2019. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. *arXiv preprint arXiv:1911.03343*.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2019. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*.

Leora Morgenstern, Ernest Davis, and Charles L Ortiz. 2016. Planning, executing, and evaluating the winograd schema challenge. *AI Magazine*, 37(1):50–54.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Abhilasha Ravichander, Eduard Hovy, Kaheer Suleman, Adam Trischler, and Jackie Chi Kit Cheung. 2020. On the systematicity of probing contextualized word representations: The case of hypernymy in BERT. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 88–102, Barcelona, Spain (Online). Association for Computational Linguistics.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Nathaniel Weir, Adam Poliak, and Benjamin Van Durme. 2020. Probing neural language models for human tacit assumptions. CogSci.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. *arXiv preprint arXiv:1904.03310*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

A Appendices

A.1 Fine-tuning Experiment

A.2 Crowdsourcing Instructions

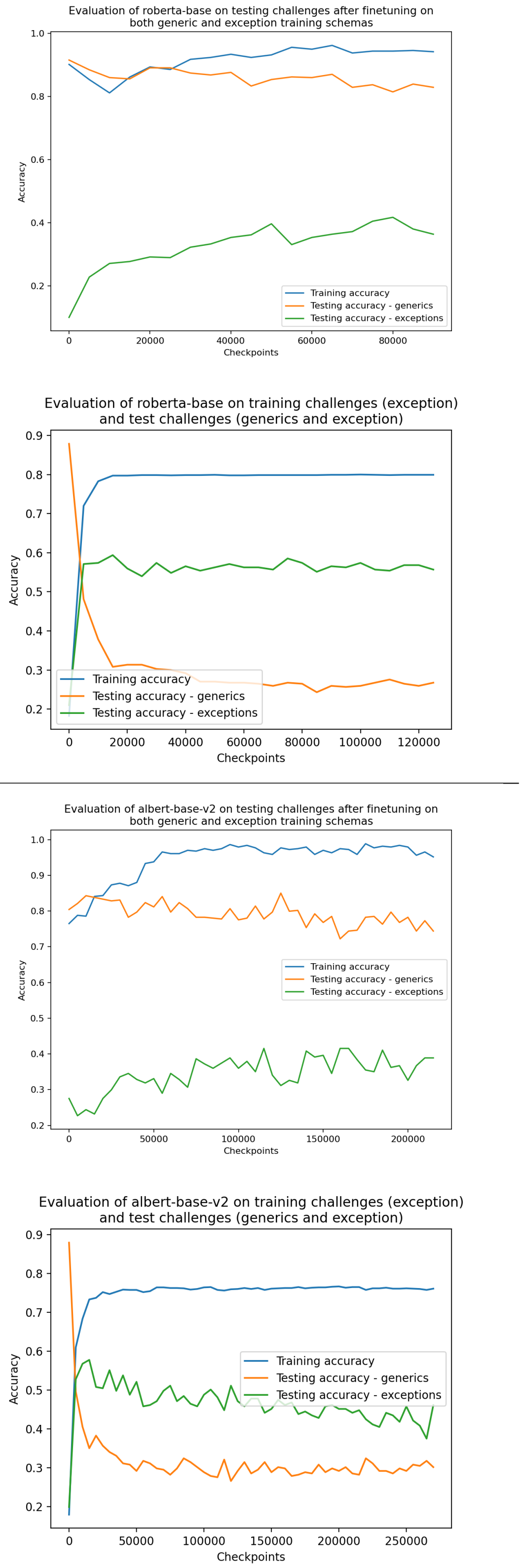

Figure 2: Fine-tuned ALBERT$_{\text{BASE}}$ and RoBERTa$_{\text{BASE}}$ show the same negative relationship between generic and exception performances. Fine-tuned RoBERTa$_{\text{BASE}}$ shows a higher increase in performance on unseen exceptions in comparison to BERT$_{\text{BASE}}$.

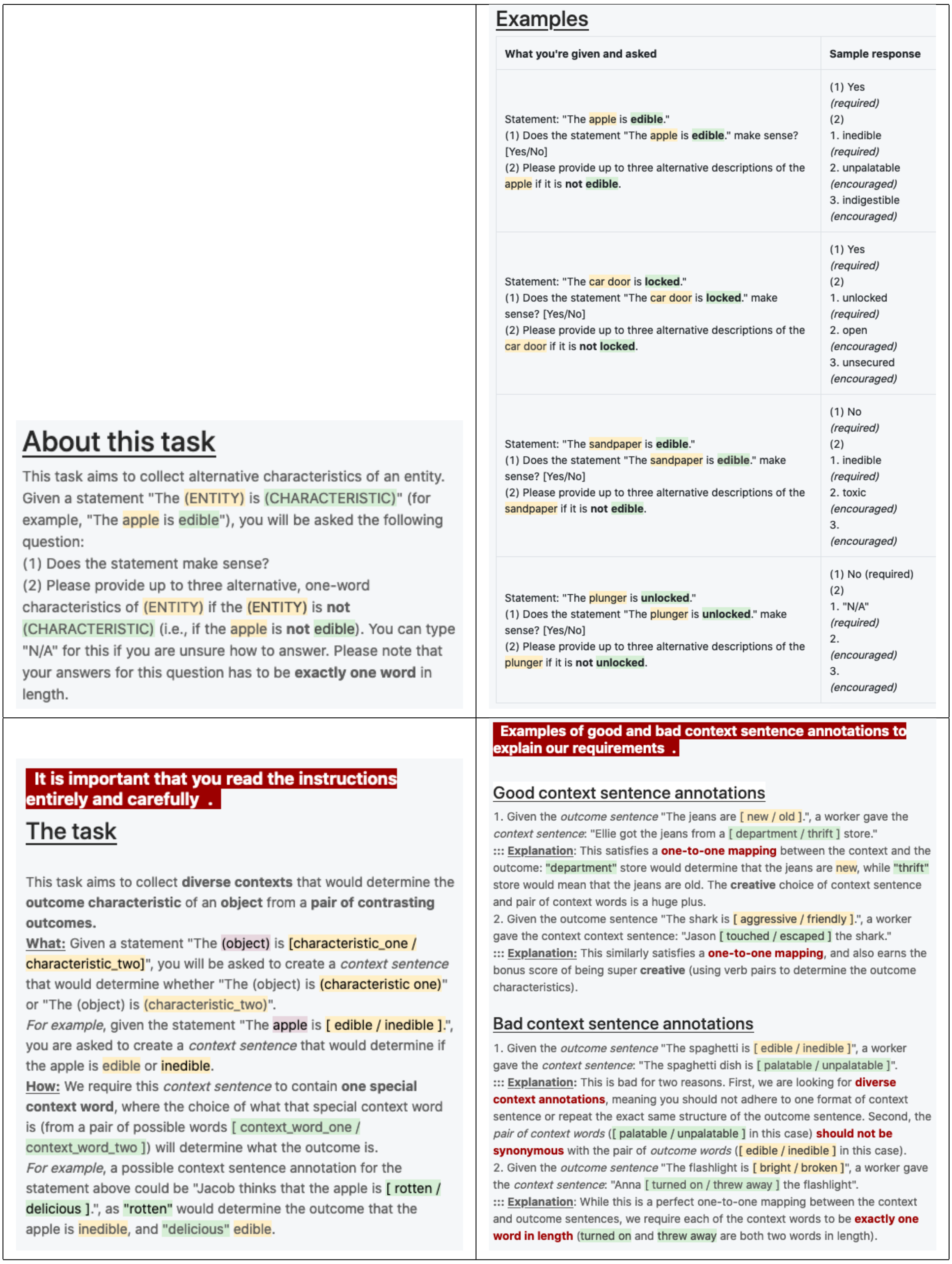

About this task

This task aims to collect alternative characteristics of an entity. Given a statement "The (ENTITY) is (CHARACTERISTIC)" (for example, "The apple is edible"), you will be asked the following question:

(1) Does the statement make sense?

(2) Please provide up to three alternative, one-word characteristics of (ENTITY) if the (ENTITY) is **not** (CHARACTERISTIC) (i.e., if the apple is **not** edible). You can type "N/A" for this if you are unsure how to answer. Please note that your answers for this question has to be **exactly one word** in length.

Examples

What you're given and asked	Sample response
Statement: "The apple is **edible**." (1) Does the statement "The apple is **edible**." make sense? [Yes/No] (2) Please provide up to three alternative descriptions of the apple if it is **not edible**.	(1) Yes *(required)* (2) 1. inedible *(required)* 2. unpalatable *(encouraged)* 3. indigestible *(encouraged)*
Statement: "The car door is **locked**." (1) Does the statement "The car door is **locked**." make sense? [Yes/No] (2) Please provide up to three alternative descriptions of the car door if it is **not locked**.	(1) Yes *(required)* (2) 1. unlocked *(required)* 2. open *(encouraged)* 3. unsecured *(encouraged)*
Statement: "The sandpaper is **edible**." (1) Does the statement "The sandpaper is **edible**." make sense? [Yes/No] (2) Please provide up to three alternative descriptions of the sandpaper if it is **not edible**.	(1) No *(required)* (2) 1. inedible *(required)* 2. toxic *(encouraged)* 3. *(encouraged)*
Statement: "The plunger is **unlocked**." (1) Does the statement "The plunger is **unlocked**." make sense? [Yes/No] (2) Please provide up to three alternative descriptions of the plunger if it is **not unlocked**.	(1) No (required) (2) 1. "N/A" *(required)* 2. *(encouraged)* 3. *(encouraged)*

It is important that you read the instructions entirely and carefully .

The task

This task aims to collect **diverse contexts** that would determine the **outcome characteristic** of an **object** from a **pair of contrasting outcomes.**

What: Given a statement "The (object) is [characteristic_one / characteristic_two]", you will be asked to create a *context sentence* that would determine whether "The (object) is (characteristic one)" or "The (object) is (characteristic_two)".

For example, given the statement "The apple is [edible / inedible].", you are asked to create a *context sentence* that would determine if the apple is edible or inedible.

How: We require this *context sentence* to contain **one special context word**, where the choice of what that special context word is (from a pair of possible words [context_word_one / context_word_two]) will determine what the outcome is.

For example, a possible context sentence annotation for the statement above could be "Jacob thinks that the apple is [rotten / delicious].", as "rotten" would determine the outcome that the apple is inedible, and "delicious" edible.

Examples of good and bad context sentence annotations to explain our requirements .

Good context sentence annotations

1. Given the *outcome sentence* "The jeans are [new / old].", a worker gave the *context sentence*: "Ellie got the jeans from a [department / thrift] store."

::: **Explanation**: This satisfies a **one-to-one mapping** between the context and the outcome: "department" store would determine that the jeans are new, while "thrift" store would mean that the jeans are old. The **creative** choice of context sentence and pair of context words is a huge plus.

2. Given the outcome sentence "The shark is [aggressive / friendly].", a worker gave the context context sentence: "Jason [touched / escaped] the shark."

::: **Explanation:** This similarly satisfies a **one-to-one mapping**, and also earns the bonus score of being super **creative** (using verb pairs to determine the outcome characteristics).

Bad context sentence annotations

1. Given the *outcome sentence* "The spaghetti is [edible / inedible]", a worker gave the *context sentence*: "The spaghetti dish is [palatable / unpalatable]".

::: **Explanation**: This is bad for two reasons. First, we are looking for **diverse context annotations**, meaning you should not adhere to one format of context sentence or repeat the exact same structure of the outcome sentence. Second, the *pair of context words* ([palatable / unpalatable] in this case) **should not be synonymous** with the pair of *outcome words* ([edible / inedible] in this case).

2. Given the *outcome sentence* "The flashlight is [bright / broken]", a worker gave the *context sentence*: "Anna [turned on / threw away] the flashlight".

::: **Explanation**: While this is a perfect one-to-one mapping between the context and outcome sentences, we require each of the context words to be **exactly one word in length** (turned on and threw away are both two words in length).

Figure 3: **Top Left**: Exception Association Collection - Instructions. **Top Right**: Exception Association Collection - Examples. **Bottom Left**: Premise Collection - Instructions. **Bottom Right**: Premise Collection - Examples.

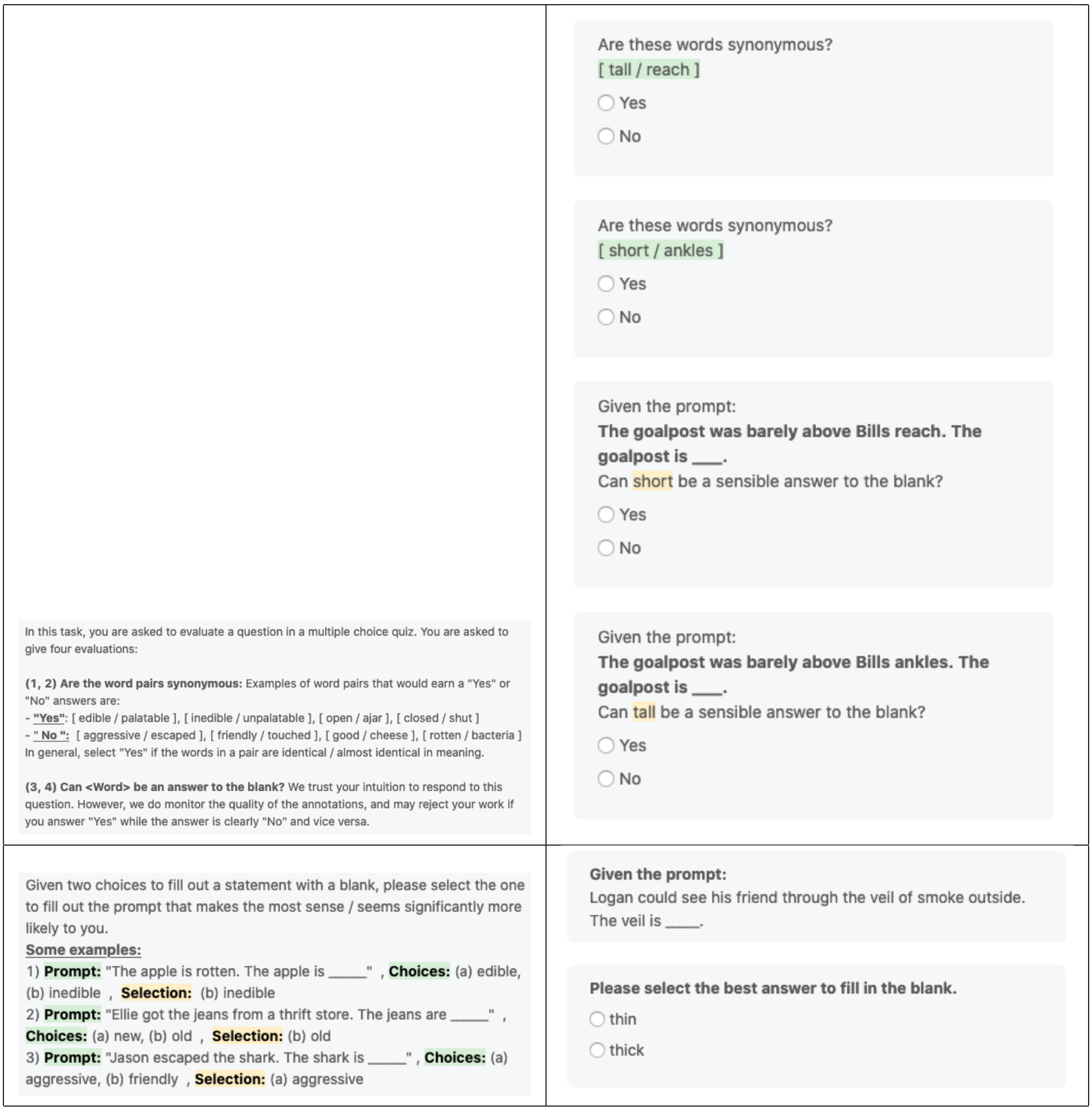

In this task, you are asked to evaluate a question in a multiple choice quiz. You are asked to give four evaluations:

(1, 2) Are the word pairs synonymous: Examples of word pairs that would earn a "Yes" or "No" answers are:
- **"Yes"**: [edible / palatable], [inedible / unpalatable], [open / ajar], [closed / shut]
- **" No ":** [aggressive / escaped], [friendly / touched], [good / cheese], [rotten / bacteria]

In general, select "Yes" if the words in a pair are identical / almost identical in meaning.

(3, 4) Can <Word> be an answer to the blank? We trust your intuition to respond to this question. However, we do monitor the quality of the annotations, and may reject your work if you answer "Yes" while the answer is clearly "No" and vice versa.

Are these words synonymous?
[tall / reach]
○ Yes
○ No

Are these words synonymous?
[short / ankles]
○ Yes
○ No

Given the prompt:
The goalpost was barely above Bills reach. The goalpost is ____.
Can short be a sensible answer to the blank?
○ Yes
○ No

Given the prompt:
The goalpost was barely above Bills ankles. The goalpost is ____.
Can tall be a sensible answer to the blank?
○ Yes
○ No

Given two choices to fill out a statement with a blank, please select the one to fill out the prompt that makes the most sense / seems significantly more likely to you.

Some examples:
1) **Prompt:** "The apple is rotten. The apple is _____" , **Choices:** (a) edible, (b) inedible , **Selection:** (b) inedible
2) **Prompt:** "Ellie got the jeans from a thrift store. The jeans are _____" , **Choices:** (a) new, (b) old , **Selection:** (b) old
3) **Prompt:** "Jason escaped the shark. The shark is _____" , **Choices:** (a) aggressive, (b) friendly , **Selection:** (a) aggressive

Given the prompt:
Logan could see his friend through the veil of smoke outside. The veil is _____.

Please select the best answer to fill in the blank.
○ thin
○ thick

Figure 4: **Top Left**: Challenge Set Validation - Instructions. **Top Right**: Challenge Set Validation - Task Sample. **Bottom Left**: Human Performance Collection - Instruction and Examples. **Bottom Right**: Human Performance Collection - Task Sample.

GRICE: A Grammar-based Dataset for Recovering Implicature and Conversational rEasoning

Zilong Zheng, Shuwen Qiu, Lifeng Fan, Yixin Zhu, Song-Chun Zhu
UCLA Center for Vision, Cognition, Learning, and Autonomy
`{z.zheng,s.qiu,lfan,yixin.zhu}@ucla.edu, sczhu@stat.ucla.edu`
`https://zilongzheng.github.io/Grice/`

Abstract

Understanding what we genuinely *mean* instead of what we literally *say* in conversations is challenging for both humans and machines; yet, this direction is mostly left untouched in modern open-ended dialogue systems. To fill in this gap, we present a grammar-based dialogue dataset, GRICE, designed to bring implicature into pragmatic reasoning in the context of conversations. Our design of GRICE also incorporates other essential aspects of modern dialogue modeling (*e.g.*, coreference). The entire dataset is systematically generated using a hierarchical grammar model, such that each dialogue context has intricate implicatures and is temporally consistent. We further present two tasks, the implicature recovery task followed by the pragmatic reasoning task in conversation, to evaluate the model's reasoning capability. In experiments, we adopt baselines that claimed to have pragmatics reasoning capability; the results show a significant performance gap between baseline methods and human performance. After integrating a simple module that explicitly reasons about implicature, the model shows an overall performance boost in conversational reasoning. These observations demonstrate the significance of implicature recovery for open-ended dialogue reasoning and call for future research in conversational implicature and conversational reasoning.

Alice: Did you see the apples?
Bob: There is a basket in the dining room.
`(The apples are in the dining room.)`
Alice: How many?
Bob: There are at least two.
`(I am not sure how many apples are there.)`
Alice: Did you put them there?
Bob: I was in the kitchen.
`(I didn't put the apples in the dining room.)`
Alice: Are all the oranges there?
Bob: Some are there.
`(Not all the oranges are in the kitchen.)`
Alice: What about the pears?
Bob: They are in the living room.
`(The pears are not in the kitchen.)`

Figure 1: **An example of the conversation in the proposed GRICE dataset.** Each round of dialogue includes a question, an answer that may contain implicature, and a recovered statement that converts the implicature to explicature. Different colors highlight coreference flows.

1 Introduction

> "When a diplomat *says* yes, he *means* 'perhaps'; when he *says* perhaps, he *means* 'no'; when he says no, he is not a diplomat."
> —Voltaire, quoted in Spanish in Escandell (1996) (Korta and Perry, 2020)

Voltaire's above quote is an epitome of a crucial aspect of conversation; the meaning of the very same word or token varies according to its context and goes *beyond* what we *literally* say, which is the central character of the field of pragmatics. Such a high-level comprehension of *utterance* is more than traditional semantics and logic; it is often believed to involve the construction of the speaker's intents, beliefs, and social institutes (Grice, 1975; Korta and Perry, 2020). For instance (see Fig. 1), when asked "did you see the apples?", one would not merely say "yes" or "no"; instead, one should provide an answer that is cooperative, truthful, informative, relevant, and perspicuous (Davis, 2016) based on the inferred speaker's intent and belief. Consequently, in the above example, a person would instead answer the actual location without mentioning any positive or negative words. Such a teleological account echoes Grice's core insight that "language use is a form of rational action; hence, technical tools for reasoning about rational action should elucidate linguistic phenomena" (Goodman and Frank, 2016).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2074–2085
August 1–6, 2021.

In stark contrast, such a goal-directed perspective of conversational reasoning has been largely ignored in the modern literature of Natural Language Processing (NLP) (but see Dale and Reiter (1995); Nematzadeh et al. (2018) as exceptions). The recent development of open-ended dialogue systems has a clear trend that adopts state-of-the-art deep learning or deep reinforcement learning methods, fueled by hardware accelerations and massive sets of labeled data. However, the inspiring progress was recently challenged by researchers (Shum et al., 2018; Young et al., 2018); there remain valid concerns that systems are simply imitating human responses by regressing a large amount of training data without truly understanding it. Although we see an emerging field of conversational reasoning (*e.g.*, Moon et al. (2019); Cui et al. (2020)), existing work fails to account for the pragmatics perspective within conversations: Human speakers usually do not speak their thoughts or intentions *directly*; it has to be inferred from the conversational context.

To fill the gap between the current open-dialogue systems and the future humanlike dialogue systems, we design a new open-dialogue dataset generation protocol, which we refer as Grammar-based dataset for Recovering Implicature and Conversational rEasoning (GRICE), in homage to H. P. Grice for his influential theory in explaining and predicting conversational implicatures (Grice, 1975). Specifically, our design follows four principles.

First, we design the GRICE dataset with a focus of *conversational implicature* (Grice, 1975), "one of the single most important ideas of pragmatics" (Levinson, 1985). Naturally, the ability to successfully perform **implicature recovery** in conversation (Borg, 2009) would be a suitable indicator of a system's performance; we adopt it as part of our evaluation protocols. To recover conversational implicature into explicit ones with only information and context in the dialogue, an ideal model should reason about the dialogue context and the relations among dialogue entities.

Second, we emphasize the comprehension of the *conversational context* and adopt the **conversational reasoning** as part of the evaluation protocols. Again, we take the conversation in Fig. 1 as the example: When the speaker says "I was in the kitchen," what she really means is that she was not in the dining room and therefore could not put the apples there. The same response would have the opposite meaning when the question becomes "Were you in the kitchen?". Such a swift switch according to its dialogue context is a quintessential demonstration that human communication is a context-dependent endeavor (Fetzer, 2017) and a dynamic construct, which relates communicators and the language that they use in a dialectical manner (Bateson, 2000).

Third, we build the GRICE dataset by incorporating five different types of implicature; see details in Section 4. To resolve these types of implicature, the algorithm ought to make a proper prediction or inference of intents and beliefs by representing and reasoning about *triadic* relations (Saxe, 2006): the speaker's belief, the addressee's belief, and what they have or communicate in common.

Fourth, in comparison to prior work, Facebook bAbi (Weston et al., 2016) and its follow-up work ToM (Nematzadeh et al., 2018) that evaluate different aspects of reasoning with a set of toy tasks, the proposed GRICE dataset does not sacrifice crucial characteristics of modern open-dialogue systems. On the contrary, by integrating pragmatics and implicature in conversation, we hope to shed light on some challenging issues in open-ended dialogue:

- *Coreference* resolution (Chen et al., 2017; Kottur et al., 2018) refers to finding all expressions that refer to the same entity in the conversation. The significance of resolving coreference becomes even more profound in conversations with implicature; Fig. 1 gives an example and highlights the coreference flows in different colors.
- *Commonsense* reasoning (Sap et al., 2019; Talmor et al., 2019; Speer et al., 2017) received an increasing attention in NLP. Notably, researchers have proposed the Winograd (Levesque et al., 2012) and WinoGrande (Sakaguchi et al., 2020) to examine commonsense reasoning. For conversations with implicature, commonsense reasoning reflects a crucial concept of *relevance*. For instance, to resolve the implicature in the conversation "A: I am out of petrol. B: There is a garage around the corner.", one needs to have the commonsense that "a garage could store petrol."
- *Logic*-based methods were once thought to be the "ideal language" approach to the semantics of human language (Russell, 1903), but were later challenged by Wittgenstein (1953, 1969) and Grice (1975). However, this disagreement should not prohibit the central role of logical forms in reasoning tasks. In fact, it would be interesting to investigate if the modern end-to-end trainable methods could benefit from logical forms in conversational reasoning.

The remainder of this paper is organized as follows. We review related work on dialogue dataset, implicature, and conversational reasoning in Section 2. In Section 3, two tasks are defined for evaluations. We present detailed design, generation, and analysis of the GRICE dataset in Section 4. By introducing two evaluation protocols, we provide the performance of baseline models with discussions of results and future directions in Section 5.

2 Related Work

Dialogue Datasets Dialogue datasets have been focusing on predicting the next most-likely response by imitating the teacher's responses (human corpus) (Lowe et al., 2015; Zhang et al., 2018; Wu et al., 2018). However, as pointed out by Cui et al. (2020), prior datasets and associated methods lack proper explicit reasoning modules; it later becomes evident that such reasoning modules serve as the scaffold in building a humanlike conversational agent. Of note, a model's reasoning capability is minimal if it simply converts reasoning challenges into a categorization problem when predicting the utterances; it still tends to choose the most frequent answer given the training set without genuinely understanding the context and underlying meaning.

To the best of our knowledge, the proposed GRICE dataset is the first open-dialogue dataset that explicitly integrates implicature; see a detailed comparison in Table 1. We hope our careful design would encourage and even necessitate future models to make explicit reasoning on conversational contexts, commonsense, and agent's intents and beliefs. The most similar dataset in terms of the format is DREAM by Sun et al. (2019), a conversational dataset with a question-answering (QA) task. However, the design of the DREAM dataset does not require much reasoning; answers can be directly extracted. The most similar dataset in terms of the task is CoQA by Reddy et al. (2019), which considers pragmatics and QAs over literature paragraphs; our GRICE dataset differs by reasoning over the *dialogue context* between two agents.

Implicature Implicature has been extensively studied in the field of linguistics and philosophy since the inception of pragmatics; Grice (1975)'s four maxims—quality, quantity, relevance, and manner—founded the principles of the interpretation of conversation implicature. Two neo-Gricean typologies of conversational implicature include Horn and Ward (2004)'s Q- and R-implicature and Levinson (1985)'s Q-, I-, and M-implicature. The relevance theory developed by Sperber and Wilson (1986) offers an alternative account to Gricean and neo-Gricean theory. In general, although these doctrines provide crucial insights into the field, they focus more on philosophical debates over toy examples without deriving computational solutions or quantitatively validating the ideas on modern large-scale natural language datasets.

Although a few computational models have been proposed recently (*e.g.*, Frank and Goodman (2012); Goodman and Stuhlmüller (2013)), these models assume the space of utterance and possible semantic meanings are finite or given, so that models only need to pick up one over others based on the shared context. Other models focus on more specific tasks; for instance, recovering the direct meaning from the indirect answer using scalar adjectives (de Marneffe et al., 2010; De Melo and Bansal, 2013), conducting analysis on the ironic implicature behind simile (Veale and Hao, 2010).

By generating paired sentences in a semi-automatic fashion with human annotations, Jeretic et al. (2020) recently devise a dataset with a focus on scalar implicature (Hirschberg, 1985). In comparison, the proposed GRICE dataset has a much more natural setup and broader scope by combining the multi-round open-dialogue with conversational implicature. Additionally, leveraging a grammar representation for fine-grained control, the GRICE dataset is generated in a fully automated fashion without human annotations. We hope such a design could boost research in implicature, pragmatics, and conversational reasoning at a large scale.

Conversational Reasoning In the past four years, we have witnessed an increasing interest in conversational reasoning in various contexts. OpenDialKG (Moon et al., 2019) incorporates external knowledge graphs to the dialogue context to provide extra entities as responses. Visual Dialog (Wu et al., 2018; Zheng et al., 2019; Das et al., 2017) takes images as external multi-modalities to reason with dialogue context to generate visually grounded responses jointly. MuTual (Cui et al., 2020) modifies English reading comprehension to select the next best response by machine reasoning.

However, prior efforts have ignored the fact that humans commonly do not directly speak out answers. The proposed GRICE dataset is a complement of prior conversational reasoning tasks; it focuses on implicature with conversational reasoning, which does not reject multi-modalities as they could be a source of commonsense knowledge.

Table 1: Comparing GRICE with existing conversational datasets.

Dataset	**Task**	**Context**	**Source Domain**
Ubuntu (Lowe et al., 2015)	Next Utterances Prediction	Dialogue	Ubuntu Chat logs
PERSONA-CHAT (Zhang et al., 2018)	Next Utterances Prediction	Dialogue	Persona
Douban (Wu et al., 2017)	Next Utterances Prediction	Dialogue	Open Domain
MuTual (Cui et al., 2020)	Next Utterances Prediction	Dialogue	Listening Comprehension
DREAM (Sun et al., 2019)	Question Answering	Dialogue	English Language Exams
CoQA (Reddy et al., 2019)	Conversational QA	Paragraph	Literature
GRICE (ours)	Implicature recovery & Question Answering	Dialogue	Open Domain with implicature

3 Task Definition

To evaluate how well a model "understands" the dialogue presented in the proposed GRICE dataset, we devise two tasks: the implicature recovery task and the conversational reasoning task, wherein the latter task depends on the successful completion of the former task. Below, we introduce the setup and evaluation protocol of each task.

Alice: Where are the oranges?
Bob: They may be in the kitchen or the patio.
Alice: What about the apples?
Bob: Jack put them in the kitchen and went to the bedroom.

(a) A sample dialogue with two rounds.

(A) Jack went to the bedroom and then put the apples in the kitchen.
(B) Jack put the apples in the kitchen and then went to the bedroom.
(C) Jack went to the bedroom and then put the oranges in the kitchen.
(D) The apples are in the bedroom.

(b) Implicature recovery evaluated with multiple choices.

Q_1: Where are the apples?
A_1: Kitchen
Q_2: Who moved the apples?
A_2: Jack
Q_3: Does Bob know where the oranges are?
A_3: No

(c) Conversational reasoning evaluated by QAs.

Figure 2: **Examples of two tasks defined in GRICE dataset.** (a) Given a multi-round open-dialogue, an algorithm is asked to perform (b) implicature recovery and (c) conversational reasoning in the form of QAs.

Task 1: Implicature Recovery Formally, an n-round dialogue occurring between two agents is represented by a sequence of QA-pairs $\{(Q_1, A_1), (Q_2, A_2), ..., (Q_n, A_n)\}$, where Q_i is the question raised by the first agent, A_i is the response provided by the second agent, which may contain an implicature. To complete this task, a model is asked to identify if A_i is a statement containing implicature, and if this is true, to resolve the implicature to its explicit form, *i.e.*, to perform implicature recovery.

The implicature recovery is evaluated in the form of multiple choices: For each utterance, the ground-truth condition (with implicature) and its explicit form are given when generating the dialogue; the explicit form, which not only recovers the implicature but also resolves coreferences in the utterance, serves as the correct answer in the multiple choices. We then sample three possible answers from the candidate pools, given a set of manually defined *speech templates* (see details in Section 4). Figs. 2a and 2b show an example: The last utterance by Bob implicates (by the word "then") the temporal order between "put them in the kitchen" and "went to the living room." Thus, the correct implicature recovery should resolve "them" as "the apples" and recover the correct temporal order.

Two strategies developed by existing work could be adopted to address this task. One strategy is to train a model that directly chooses an answer from the candidate answers. Another more challenging strategy is to train a generator that chooses the answer by computing the log-likelihood scores and ranking the candidate answers as done by Das et al. (2017). To quantitatively evaluate the performance, we use the standard response selection metrics (Lowe et al., 2015; Wu et al., 2017; Cui et al., 2020): Top 1 Recall (R@1) and Mean Reciprocal Rank (MRR) (Voorhees et al., 1999).

Task 2: Conversational Reasoning To evaluate the open-ended conversational reasoning, we follow the same protocols as in Weston et al. (2016) and Nematzadeh et al. (2018) with comprehensive QAs. For each dialogue, we generate questions by randomly sampling the *conversational contexts* (see Section 4), and each question could be answered by a single word; see Fig. 2c for examples.

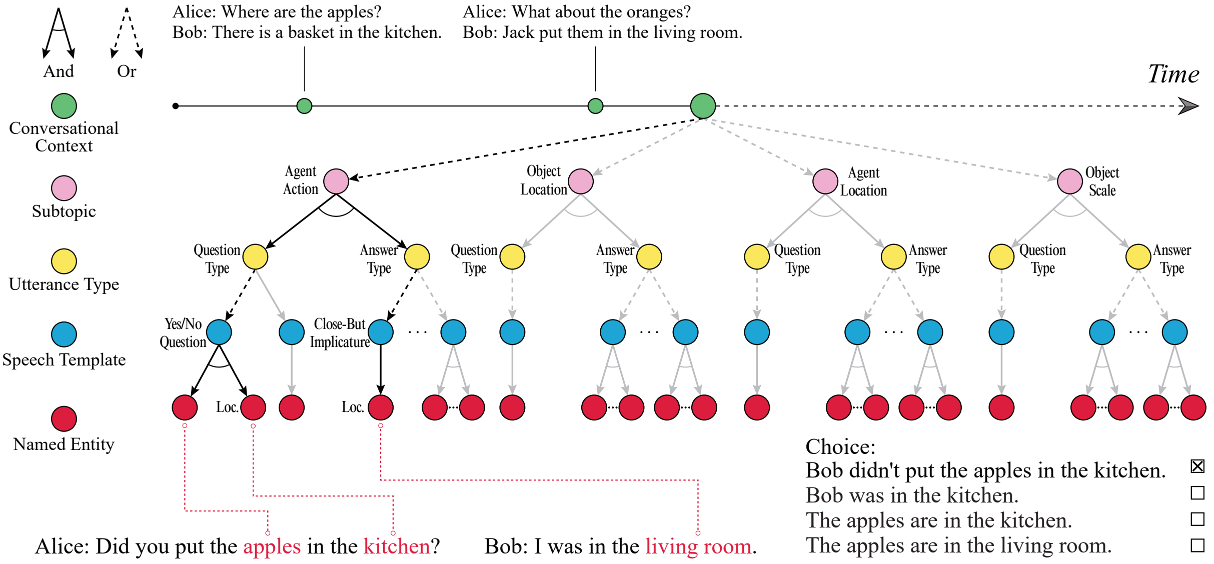

Figure 3: The graphical illustration of the grammar production rules for the GRICE dataset.

4 Creating the GRICE Dataset

Representation We adopt a structural grammar model, Temporal And-Or Graph (T-AOG) (Qi et al., 2020; Edmonds et al., 2019a; Tu et al., 2013), to represent the dialogue context due to its expressiveness of hierarchical dialogue structure and temporal-dependent dialogue flow. We represent one *turn* of the dialogue as an AOG (Bonczek et al., 1979, 1981; Pearl, 1984; Zhu and Mumford, 2007) that has a hierarchy of five levels: conversational context, subtopic, utterance type, speech template, and named entity. AOGs are connected w.r.t. temporal constraints in order to assemble the T-AOG.

Formally, an AOG (*i.e.*, each turn of the dialogue) has two sets of non-terminal vertex: (i) a set of And-nodes, wherein each node represents the decomposition of a larger concept (*e.g.*, subtopics) into smaller components (*e.g.*, utterance types), and (ii) a set of Or-nodes, wherein each node branches to an alternative decomposition (*e.g.*, a conversational context could have different types of subtopics), enabling the model to reconfigure the overall dialogue. An instance of AOG can be constructed by selecting a child node for each Or-node, resulting in a parse graph.

Fig. 3 illustrates an example of AOG. Specifically, the root node of one dialogue turn is an Or-node, representing the current conversational context. Represented by an And-node, each child node of the root note denotes a subtopic of the current dialogue turn. The subtopic is composed of a set of utterance types, further decomposed into speech templates filled by named entities. Instantiating an AOG by selecting Or-nodes would produce a complete utterance of a dialogue turn and pose constraints on the next dialogue turn.

Conversational Context We follow Weston et al. (2016) to represent dialogue context by a simulated world with various dialogue entities: *objects*, *locations*, and *agents*. We randomly initialize a world for each dialogue snippet by (i) positioning objects in locations with a random scalar (one, two, ...), (ii) randomly setting a location for each agent as the "previous agent location," and (iii) for each $\langle object \rangle$ in $\langle location \rangle$, randomly selecting an $\langle agent \rangle$ in $\langle location \rangle$ to denote that "$\langle agent \rangle$ put the $\langle object \rangle$ in the $\langle location \rangle$."

Subtopic In this dataset, we focus on four different subtopics: *agent_location*, *agent_action*, *object_location* and *object_scale*; see examples in Table 3. Specifically, *agent_location* queries the location of some $\langle agent \rangle$. The example in Table 3 implicates that "Jack was in the kitchen." Similarly, *object_location* queries the location of some $\langle object \rangle$. *Agent_action* queries the previous action taken by some $\langle agent \rangle$ on some $\langle object \rangle$. Typically, the action can be identified as an $\langle agent \rangle$ put $\langle object \rangle$ in the $\langle location \rangle$. *Object_scale* queries the quantity of some $\langle object \rangle$. In particular, an algorithm should also be able to reason about the strength among the quantifying phrases, such as *at least*, *some*, and *all*. A typical example shown in Table 3 implicates that "Bob does not know if all the apples are in the kitchen."

Utterance Type Utterance type concerns how to generate a QA-pair correctly. For questions, the

Table 2: **Definitions and examples of five types of implicature in the proposed GRICE dataset.** Following the conventional notation, S denotes the positive answer of the question, and S^+ denotes its stronger proposition.

Category	Definition	Example
Relevance	Implicating the answer to an expressed or implied question by stating something related to the answer by implication or explanation.	Alice: Where did you see the apples? Bob: There is a basket in the kitchen. (The apples are in the kitchen.)
Strengthening	Implicating a stronger proposition S^+ when not understatement.	Alice: Are some of the apples in the kitchen? Bob: All of them are there. (Not just some, but all of the apples are in the kitchen.)
Limiting	Implicating the **denial** of S^+.	Alice: Are all the apples in the kitchen? Bob: Some are. (**Not all** apples are in the kitchen.)
Ignorance	Implicating that one **does not know** whether S^+ is true (or that S^+ may or may not be true).	Alice: Where did you see Jack? Bob: He was in the kitchen or the bedroom. (I am not sure where Jack was.)
Close-But	Implicating a **negative** answer to a question by affirming something close to a positive answer in contextually salient respects.	Alice: Did you put the apples in the kitchen? Bob: I was in the living room. (I did not put the apples in the kitchen since I was in somewhere else.)

Table 3: Categories and examples of different subtopics in GRICE dataset.

Subtopic	Example
agent_location	Alice: Where was Jack? Bob: I saw him in the kitchen.
agent_action	Alice: Did you put the apples in the kitchen? Bob: I was in the bedroom.
object_location	Alice: Where can I find the apples? Bob: They are in the kitchen, if not the living room.
object_scale	Alice: Are all the apples in the kitchen? Bob: At least four are there.

query types of each subtopic are manually defined. For example, the question regarding *Agent_location* can be a yes/no question ("were you in the kitchen?") or a where question ("where were you?"). For answers, we focus on five different types of implicature (Huang, 2017; Horn and Ward, 2004; Davis, 2016): *relevance*, *strengthening*, *limiting*, *ignorance*, and *close-but*; see Table 2 for detailed definitions and examples.

Diversity We follow Weston et al. (2016) to use a simple automated grammar to makes the conversation more natural and diverse: We assign a set of synonyms for each verb; *e.g.*, we randomly replace (i) *put* with *left*, *dropped*, or *placed*, and (ii) *went* with *travelled*, *journeyed*, or *walked*.

Since coreference is a crucial feature in the conversational context in GRICE dataset, we track agents, objects, and locations mentioned in previous conversations and replace them with deixis in the following conversational context.

Additionally, we build a set of follow-up questions for each type of dialogue action to challenge the model's ability to reason about the omission in utterances. Take Fig. 2 as an example; the question "What about the apples?" should be interpreted or recovered as "Where are the apples?" during the reasoning procedure.

Candidate Answer Generation To generate candidate answers for each round of dialogue for the implicature recovery task, we define four different strategies tailored to produce challenging candidates. Among all four candidate answers, besides the ground-truth condition in its explicit form, the other three candidate answers are randomly sampled from the candidate pool, composed by applying the following strategies; see Fig. 4 for examples of each strategy:

1. Statements that are similar to the ground-truth condition but with wrong coreferenced entities.
2. Randomly sampled true condition but with irrelevant facts.
3. Randomly sampled wrong facts from the current conversational context.
4. Manually created statements that are close to the true condition but are in fact wrong.

Conversation:
Alice: Where are the oranges?
Bob: Jack said he saw some in the kitchen.
Alice: Did he put them there?
Bob: He put them there and went to the bedroom.
`(Jack put the oranges in the kitchen and then went to the bedroom.)`

Examples of generated candidate answers:
1. Bob put the oranges in the kitchen and then went to the bedroom.
2. Jack was in the bedroom.
3. The oranges are in the bedroom.
4. Jack went to the bedroom and then put the oranges in the kitchen.

Figure 4: **The candidate answers for the implicature recovery task are generated following four different strategies.** 1. Statements that are similar to the ground-truth condition but with wrong coreferenced entities. 2. Random sampled true condition but with irrelevant facts. 3. Random sampled wrong facts from the conversational context. 4. Manually created statements that are close to the true condition but are in fact wrong.

Questions We follow Weston et al. (2016) to generate questions about the dialogue context. After sampling the dialogue turns and finalizing the dialogue context, we query current dialogue states in terms of agent locations/actions and object locations/scales. Inspired by Nematzadeh et al. (2018), we further add belief queries (*e.g.*, "does Bob know where the oranges are?") to test the model's capability of belief reasoning; see Fig. 2 for examples.

5 Experiments

We randomly sample 6,000 dialogues as the train set and additional 4,000 dialogues as the dev set to evaluate baseline models; each dialogue contains 10 dialogue turns and 3 questions. Detailed distributions of implicature types are listed in Table 4. For the test set, we sample 1,000 dialogues in each implicature category, resulting in a total of 5,000 dialogues. Each test dialogue contains 3–5 dialogue turns and one question on implicature. All data is clean and noiseless.

Setup We model both tasks as a query over the conversational context. Specifically, for the implicature recovery task, we define $h_t = (Q_t, A_t)$ as the queried sequence and the $H_t = \{(Q_1, A_1), ..., (Q_{t-1}, A_{t-1})\}$ as the past dialogue context. Then the task is to predict the explicit form $E_t = f(h_t, H_t)$. For the conversational reasoning task, we treat the entire history as the input context and the question as the query sequence. The task is then modeled as a Sequence-to-Vector framework that maps the query with its context to the vocabulary space. We implemented all models in PyTorch and trained using ADAM (Kingma and Ba, 2014) with a learning rate of 0.001 for 40 epochs.

Table 4: Distribution of implicature types (%).

	Train	Dev
Explicit Answer	27.3	29.6
Implicature	72.7	70.4
Relevance	9.9	9.3
Strengthening	22.5	22.9
Limiting	6.3	6.4
Ignorance	23.5	21.2
Close-But	10.5	10.8

5.1 Baseline Models

We evaluate 5 representative baseline models for both tasks on the GRICE dataset. The baseline models are chosen on the basis of performing well on synthetic language datasets (*e.g.*, Facebook bAbi) or similar tasks and easy adoption to perform conversational reasoning tasks. We additionally test the performance of transformer-based language models, claimed to have strong reasoning capabilities.

LSTM We start with a simple dual LSTM model: one LSTM to encode the history context as a long context sequence, and another LSTM to encode the queried sequence. A simple MLP fuses two encoded vectors to predict answers.

Recurrent Entity Network (EntNet) EntNet (Henaff et al., 2017) is an RNN-based memory-augmented architecture, capable of capturing the sequential nature and learning relevant entities with their properties by gated recurrent units and weight matrices. Our implementation is based on its official open-sourced code[1].

Relation Network (RelNet) Santoro et al. (2017) propose a neural model for relational reasoning. The algorithm considers each pair of sentences together with the question as inputs. Our implementation is based on its official open-sourced code[2].

Memory Network (MemNN) We follow Weston et al. (2015) to build a memory network[3] that takes each round of history context as a supporting fact and stores it in the memory bank; the algorithm

---

[1] `https://github.com/jimfleming/recurrent-entity-networks`

[2] `https://github.com/siddk/relation-network`

[3] `https://github.com/facebook/MemNN`

is expected to learn to refer to the memory when predicting answers. Specifically, we use an LSTM to encode each round of history and compute the association matrix between the queried sequence and the memory bank. We apply a softmax to the association matrix to get the attended weight of the dialogue history. Finally, we compute the attended dialogue history embedding and combine it with the queried embedding using a simple MLP to predict answers.

Transformer-based Language Model Fine-tuning transformer-based language models (*e.g.*, GPT (Radford et al., 2018) and BERT (Devlin et al., 2019)) has shown superior performance on conversational reasoning tasks (Sun et al., 2019). We use BERT-base-uncased [4] as our pre-trained model and apply it to the conversational reasoning task by adding a single linear layer to generate answers from the target vocabulary set.

Human Performance We randomly selected 100 dialogues and assigned them to 40 human subjects in a between-subject design; 20 subjects for the implicature recovery tasks, and another 20 subjects for the conversational reasoning task.

5.2 Evaluation and Results

Implicature Recovery We start by evaluating the performance of the baseline models on the implicature recovery task. As discussed in Section 3, we evaluate under two different settings to predict the implicature recovery results: the discriminative setting and the generative setting (marked by "-Gen"). For the discriminative setting, we take the encoder output and compute the similarity score with each candidate answer to predict the final choice. For the generative setting, we train the encoder-decoder framework using the teacher-forcing algorithm by minimizing the negative log-likelihood between the generated answers and the ground-truths. Overall, the generative setting is more challenging than the discriminative one; see Table 5 for results on dev and test sets.

Conversational Reasoning We follow Weston et al. (2016) and Nematzadeh et al. (2018) on performance evaluation of the conversational reasoning task, measured by the accuracy score in the vocabulary space; see Table 6 for the results of all the baseline models on the dev and test sets.

Table 5: Performance on implicature recovery task.

	Dev		Test	
Model	R@1	MRR	R@1	MRR
LSTM	81.92	0.9046	83.54	0.9145
EntNet	89.07	0.9445	91.15	0.9523
RelNet	93.02	0.9623	95.33	0.9602
MemNN	96.76	0.9833	97.29	0.9862
LSTM-Gen	62.28	0.7763	65.02	0.7784
MemNN-Gen	86.29	0.9305	88.79	0.9418
Human	99.00	-	98.50	-

Table 6: Performance on conversational reasoning task.

	Accuracy (%)	
Model	Dev	Test
LSTM	59.77	55.82
EntNet	57.91	53.17
RelNet	63.02	65.50
MemNN	64.66	67.32
BERT	67.21	71.06
MemNN w/ inf	69.24	73.12
Human	98.50	97.50

Analysis Comparing the model performance with the human performance in Tables 5 and 6, we see a consistent and competent performance in human subjects, whereas the model performance of the conversational reasoning task drops significantly even after a relatively good performance on the implicature recovery task. This contrast indicates that the models that perform well on the implicature recovery task may not really "understand" the conversational context to be used in the following conversational reasoning task.

To further test this hypothesis, for the implicature recovery task, we additionally pre-train an inference encoder that predicts the explicit/recovered answer under the generative settings (MemNN w/ inf), given the previous dialogue history. This additional inference model is further appended into the basic model and fused to predict the final answer. Such a setting would be a reasonable test to see how well a model could perform if they explicitly incorporate the recovered implicature from the implicature recovery task to solve the later conversational reasoning task. As shown in both Table 6 and Fig. 5, we observe that the conversational reasoning performance improves an average 5% with this additional inference module; for certain implicature types, it boosts the performance for more than 25%. Of note, it even outperforms the previous state-of-the-art model that fine-tunes the pre-trained Bert model, indicating the significance of incorporating an explicit module of implicature recovery for pragmatic reasoning in conversation.

---

[4] `https://github.com/huggingface/transformers`

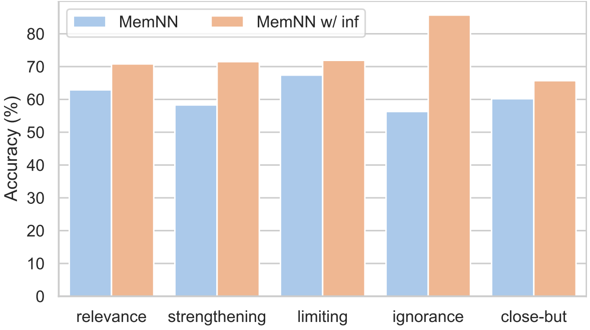

Figure 5: Performance comparison between MemNN and with additional inference module (MemNN w/ inf) that explicitly recovers the implicature.

6 Discussions and Future Work

Synthetic Corpus *vs.* Natural Corpus Creating synthetic datasets is commonly challenged in the current deep learning community due to the potential unnaturalness of the generated corpus. Nevertheless, it is worth noting that the axes along which the dataset is unnatural are unrelated to our primary focus—pragmatic implicature. By carefully and systematically incorporating the pragmatic phenomena existing in daily conversations, the proposed GRICE dataset, though synthetic, could be considered as one additional dimension in evaluating language models. In fact, although moving towards natural conversations may increase the diversity of responses, it will also introduce two potential problems: (i) Most daily conversational snippets only consist of one or fewer implicature, which cannot highlight the core challenges presented in the proposed GRICE dataset. (ii) The implicatures in natural dialogues are unstructured, requiring experts to label their explicit form, which may introduce errors and uncertainties.

Direct Evaluation *vs.* Indirect Evaluation Although the proposed GRICE dataset incorporates the triadic relations among agents and additional challenges (*e.g.*, coreference, commonsense) presented in modern dialogue systems, it is difficult to directly evaluate these aspects in an open-ended dialogue system, especially with implicature. One may use an indirect metric, *i.e.*, whether the system performance would improve after integrating such modules. Moving forward, we call for future research to design more direct evaluation metrics in addition to the present implicature recovery and conversational reasoning tasks.

Human Performance *vs.* Machine Performance The experimental results show that the existing models do exhibit a certain level of reasoning capability, though weak. Additionally, the performance gap between the implicature recovery and conversational reasoning tasks leaves us with many mysteries. Humans seem to be reasonably consistent in solving both tasks, whereas current models are not. One possible explanation is that the computational model is able to fit the relatively confined space of the implicature recovery task based on the training data, but fails to incorporate such knowledge for the more open-ended conversational reasoning task. This possible explanation is further backed up by the above experiment with an additional inference module.

Another potential reason is that existing models may lack *generalizability* that can leverage knowledge learned from known implicature to solve unseen conversations. In other words, they can only *memorize* token patterns from existing corpus rather than understand the *rationale* behind the language context, thus would fail to perform deductive or abductive reasoning tasks. Similar observation has been investigated by other reasoning tasks, including IQ test (Zhang et al., 2019a,b, 2021b), number sense (Zhang et al., 2020), causal reasoning (Edmonds et al., 2018, 2019b, 2020; Zhang et al., 2021a), and more generic generalization tasks (Lake et al., 2015; Xie et al., 2021; Li et al., 2021; Zhu et al., 2020).

Fundamentally, how could we properly leverage the knowledge extracted during the implicature recovery task for the following conversational reasoning task? Levinson (1995) argues that human conversation depends on intention-ascription, where inferences must be made way beyond the data, therefore forming an *abductive* process. A possible and promising future direction would be using a neural-symbolic solver, capable of handling noisy inputs using neural-network modules and reasoning about the answers in a logic-like style.

Acknowledgments

The authors thank Prof. Nanyun Peng at UCLA for helpful discussions, Yifan Zhang at UCLA for help on graphic design, and Dr. Wenjuan Han at BIGAI for helpful discussions and proofreading. This work reported herein was supported by ONR MURI N00014-16-1-2007, ONR N00014-19-1-2153, and DARPA XAI N66001-17-2-4029.

References

Gregory Bateson. 2000. *Steps to an ecology of mind: Collected essays in anthropology, psychiatry, evolution, and epistemology*. University of Chicago Press.

Robert H Bonczek, Clyde W Holsapple, and Andrew B Whinston. 1979. Computer-based support of organizational decision making. *Decision Sciences*, 10(2):268–291.

Robert H Bonczek, Clyde W Holsapple, and Andrew B Whinston. 1981. A generalized decision support system using predicate calculus and network data base management. *Operations Research*, 29(2):263–281.

Emma Borg. 2009. On three theories of implicature: Default theory, relevance theory and minimalism. *International Review of Pragmatics*, 1(1):63–83.

Henry Y Chen, Ethan Zhou, and Jinho D Choi. 2017. Robust coreference resolution and entity linking on dialogues: Character identification on tv show transcripts. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. Mutual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wayne A Davis. 2016. Implicature. In *Irregular Negatives, Implicatures, and Idioms*, pages 51–84. Springer.

Gerard De Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics (TACL)*, 1:279–290.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186. Association for Computational Linguistics.

Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. 2019a. A tale of two explanations: Enhancing human trust by explaining robot behavior. *Science Robotics*, 4(37).

Mark Edmonds, Feng Kubricht, James, Colin Summers, Yixin Zhu, Brandon Rothrock, Song-Chun Zhu, and Hongjing Lu. 2018. Human causal transfer: Challenges for deep reinforcement learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)*.

Mark Edmonds, Xiaojian Ma, Siyuan Qi, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. 2020. Theory-based causal transfer: Integrating instance-level induction and abstract-level structure learning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Mark Edmonds, Siyuan Qi, Yixin Zhu, James Kubricht, Song-Chun Zhu, and Hongjing Lu. 2019b. Decomposing human causal learning: Bottom-up associative learning and top-down schema reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)*.

Victor Escandell. 1996. *Introducción a la pragmática*. Ariel Linguistica. españa. 6ta.

Anita Fetzer. 2017. Context. In *The Oxford handbook of pragmatics*. Oxford University Press.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Noah D Goodman and Michael C Frank. 2016. Pragmatic language interpretation as probabilistic inference. *Trends in cognitive sciences*, 20(11):818–829.

Noah D Goodman and Andreas Stuhlmüller. 2013. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in cognitive science*, 5(1):173–184.

Herbert P Grice. 1975. Logic and conversation. In *Speech acts*, pages 41–58. Brill.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. *International Conference on Learning Representations (ICLR)*.

Julia Linn Bell Hirschberg. 1985. *A theory of scalar implicature*. University of Pennsylvania.

Laurence R Horn and Gregory L Ward. 2004. *The handbook of pragmatics*. Wiley Online Library.

Yan Huang. 2017. *The Oxford handbook of pragmatics*. Oxford University Press.

Paloma Jeretic, Alex Warstadt, Suvrat Bhooshan, and Adina Williams. 2020. Are natural language inference models imppressive? learning implicature and presupposition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kepa Korta and John Perry. 2020. Pragmatics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, spring 2020 edition. Metaphysics Research Lab, Stanford University.

Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of European Conference on Computer Vision (ECCV)*.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *International Conference on the Principles of Knowledge Representation and Reasoning*.

Stephen C Levinson. 1985. *Pragmatics*. Cambridge Univ. Press.

Stephen C Levinson. 1995. Interactional biases in human thinking. In *Social intelligence and interaction*, pages 221–260. Cambridge University Press.

Qing Li, Siyuan Huang, Yining Hong, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. 2021. A hint from arithmetic: On systematic generalization of perception, syntax, and semantics. *arXiv preprint arXiv:2103.01403*.

Ryan Lowe, Nissan Pow, Iulian Vlad Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Marie-Catherine de Marneffe, Christopher D Manning, and Christopher Potts. 2010. Was it good? it was provocative. learning the meaning of scalar adjectives. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik, and Tom Griffiths. 2018. Evaluating theory of mind in question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Judea Pearl. 1984. *Intelligent search strategies for computer problem solving*. Addision Wesley.

Siyuan Qi, Baoxiong Jia, and Song-Chun Zhu. 2020. Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics (TACL)*, 7:249–266.

Bertrand Russell. 1903. *The Principles of Mathematics*. Cambridge University Press.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rebecca Saxe. 2006. Uniquely human social cognition. *Current opinion in neurobiology*, 16(2):235–239.

Heung-Yeung Shum, Xiao-dong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Dan Sperber and Deirdre Wilson. 1986. *Relevance: Communication and cognition*. Harvard University Press Cambridge, MA.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics (TACL)*, 7:217–231.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kewei Tu, Maria Pavlovskaia, and Song-Chun Zhu. 2013. Unsupervised structure learning of stochastic and-or grammars. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

Tony Veale and Yanfen Hao. 2010. Detecting ironic intent in creative comparisons. In *European Conference on Artificial Intelligence (ECAI)*.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *International Conference on Learning Representations (ICLR)*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *International Conference on Learning Representations (ICLR)*.

Ludwig Wittgenstein. 1953. *Philosophical investigations. Philosophische Untersuchungen.* Macmillan.

Ludwig Wittgenstein. 1969. *The blue and brown books*, volume 958. Blackwell Oxford.

Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton Van Den Hengel. 2018. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sirui Xie, Xiaojian Ma, Peiyu Yu, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. 2021. Halma: Humanlike abstraction learning meets affordance in rapid problem solving. *arXiv preprint arXiv:2102.11344*.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. 2019a. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chi Zhang, Baoxiong Jia, Mark Edmonds, Song-Chun Zhu, and Yixin Zhu. 2021a. Acre: Abstract causal reasoning beyond covariation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. 2019b. Learning perceptual inference by contrasting. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. 2021b. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wenhe Zhang, Chi Zhang, Yixin Zhu, and Song-Chun Zhu. 2020. Machine number sense: A dataset of visual arithmetic problems for abstract and relational reasoning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Zilong Zheng, Wenguan Wang, Siyuan Qi, and Song-Chun Zhu. 2019. Reasoning visual dialogs with structural and partial observations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Song-Chun Zhu and David Mumford. 2007. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362.

Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Ying Nian Wu, Josh Tenenbaum, and Song-Chun Zhu. 2020. Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, 6(3):310–345.

RetroGAN: A Cyclic Post-Specialization System for Improving Out-of-Knowledge and Rare Word Representations

Pedro Colon-Hernandez[1], **Yida Xin**[2], **Henry Lieberman**[3],
Catherine Havasi[4], **Cynthia Breazeal**[1], and **Peter Chin**[2]

[1]MIT Media Lab
{pe25171,cynthiab}@media.mit.edu
[2]Boston University
{yxin,spchin}@cs.bu.edu
[3]MIT CSAIL
lieber@media.mit.edu
[4]Basis Technologies*
havasi@basistech.com

Abstract

Retrofitting is a technique used to move word vectors closer together or further apart in their space to reflect their relationships in a Knowledge Base (KB). However, retrofitting only works on concepts that are present in that KB. RetroGAN uses a pair of Generative Adversarial Networks (GANs) to learn a one-to-one mapping between concepts and their retrofitted counterparts. It applies that mapping (post-specializes) to handle concepts that do not appear in the original KB in a manner similar to how some natural language systems handle out-of-vocabulary entries. We test our system on three word-similarity benchmarks and a downstream sentence simplification task, and achieve the state of the art (CARD-660). Altogether, our results demonstrate our system's effectiveness for out-of-knowledge and rare word generalization.

1 Introduction

Retrofitting word embeddings with a KB (Faruqui et al., 2015; Speer and Chin, 2016; Mrkšić et al., 2017) means taking a vector space of word embeddings and finding a mapping that moves some of these word vectors closer together and others further apart, such that these vectors' new positions in the vector space are in better agreement with the relationships between the same words (a.k.a., concepts) in a KB (Speer and Chin, 2016; Mrkšić et al., 2017). However, the retrofitting process can only work on concepts that are actually present in the KB (a.k.a., constraints), which means that retrofitting can get us improved performance in semantic tasks only on the overlapping vocabulary between the KB and the word embeddings. Post-specialization(Vulić et al., 2018; Kamath et al., 2019) is a solution to this problem; it is a series of techniques that try to (1) learn the mapping that retrofitting establishes and (2) generalize the mapping to the rest of the embedding vocabulary.

We develop and present a post-specialization system called RetroGAN that builds upon the approach presented as AuxGAN (Ponti et al., 2018) by extending it to have a pair of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). A regular GAN minimizes the loss when learning the function for post-specialization. Our pair works in a cyclic manner to minimize the losses of both the post-specialization and the inverse to ensure that there is a one-to-one mapping between the two domains. This constrains the outputs for unseen data in both domains and leads to achieving higher performance for unseen concepts.

2 Related Work

Within the field of retrofitting, work has been done in exploring the various ways of infusing constraints or KBs into word embeddings. The original work by (Faruqui et al., 2015) only used synonymy relationships but not antonymy relationships, which meant that word embeddings with similar (synonymous) semantics in the KB would be pulled together, but word embeddings with dissimilar (antonymous) semantics would not be separated. The Attract-Repel work by (Mrkšić et al., 2017) addressed this shortcoming by incorporating antonymy relationships in a retrofitting procedure: synonymous embeddings are *attracted* to each other, while antonymous embeddings are *repelled* against each other. This line of work was continued with the work done by Lexical Entailment Attract-Repel (Vulić and Mrkšić, 2018)(LEAR),

---

*Work done while at the MIT Media Lab

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2086–2095
August 1–6, 2021.

Word	Distributional Neighbor	Retrofitted Neighbor
Dog	dog, dogs, puppy, pup , canine, pet, doggie, beagle, dachshund, cat	dog, beagle, pooch, dachshund, puppy, mutt, poodle, Rottweiler, canine, labrador
Doggo	pooch, doggies, bae, chihuahua, rad, pug, kitty, dane, furbabies, `\uf602`	doggies, pooch, dachshund, four-legged, Yorkie, corgi, whippet, amigos, Weimaraner, Dog

Table 1: Results of the 10 most similar embeddings for "dog" and "doggo" for FastText embeddings. The distributional neighbors are the closest embeddings in the original distributional space and the retrofitted neighbors are the closest in the RetroGAN post-specialized space. We can see that "doggo" was near slangs such as "bae" and "furbabies", but after post-specialization, it gets closer to words that we regard as semantically similar to "dog." The one-to-one mapping that RetroGAN provides is key to being able to incorporate useful semantic information into rare-words possibly like "doggo".

which looks to add the asymmetric lexical entailment relationship to Attract-Repel. [1]

Building on these works, a series of techniques called *post-specialization* were developed. These techniques consist on utilizing neural models to learn retrofitting mappings such as (Glavaš and Vulić, 2018) and (Ponti et al., 2018; Kamath et al., 2019) which use a Deep Feed-forward Neural Network and a Generative Adversarial Network (GAN) respectively. Post-specialization permits, provided a static word embedding, to generate its retrofitted counterpart on the fly with a trained system. A concrete example is in table 1.

As it stands, attention has been shifted to using contextual embeddings such as those produced from BERT (Devlin et al., 2019) on downstream tasks. Only recently have there been efforts in incorporating external, KB assertions into pre-trained transformer-based systems (e.g., KnowBERT (Peters et al., 2019), Align-mask-select (Ye et al., 2019), and LIBERT (Lauscher et al., 2019)). LIBERT bridges contextual and retrofitted embeddings by leveraging the knowledge in retrofitted embeddings to find lexical tuples that are fed into BERT to focus on their lexical information.

GANs have been utilized extensively in the image domain to create lifelike images. CycleGAN (Zhu et al., 2017) and other cyclic systems (Kim et al., 2017) have been utilized to perform style transfer (i.e. apply certain distinctive characteristics from one image domain into another). CycleGAN serves to learn a, possibly unpaired, one-to-one mapping from one domain to another. To effectively utilize paired data, the work by (Tripathy et al., 2018) modifies the CycleGAN architecture to include a conditional cyclic loss in which new discriminators are conditioned to determine if a generated sample is real or not based on a given, possibly paired, input. This in turn permits leveraging paired data to improve the one-to-one mapping.

3 RetroGAN

RetroGAN is a system that builds on (Ponti et al., 2018) by utilizing a CycleGAN-like architecture (i.e., we use a pair of GANs cyclicly but our layers are different from the original CycleGAN). We chose the CycleGAN-like architecture because, in our domain, the cycle-consistency constraints can enforce a one-to-one mapping from original embeddings to retrofitted embeddings. This mapping guarantees that unseen concepts will have their own, unique retrofitted counterparts. We use RetroGAN to learn the mapping of Attract-Repel (Mrkšić et al., 2017) retrofitting (with the *synonymy* and *antonymy* constraints from the Attract-Repel paper(Mrkšić et al., 2017)) on a subset of static word embeddings (i.e.., FastText (Bojanowski et al., 2017), and Numberbatch (Speer et al., 2017)), and perform post-specialization on the entire set.

3.1 Model & Architecture

RetroGAN consists of two GANs that interplay to balance a combination of losses to transform a particular word embedding $x_i \in X$ from its original domain X to its counterpart $y_i \in Y$ in the retrofitted domain Y, and vice-versa. In both GANs that we employ, the generator consists of an input layer followed by 2 hidden dense layers with 2048 neurons and each followed by a dropout layer (with a percentage of 0.2 for the dropouts), and a final linear output layer with the same dimensionality as the input. The output of this layer, for the trained $G : X \to Y$ produces the post-specialized embeddings (i.e. a batch of 32 FastText embeddings produces 32 post-specialized embeddings). The hidden layers employ the ReLU (Nair and Hinton, 2010) activation function. Our discriminators have a similar structure (an input layer, 2 hidden

---

[1] We do not use LEAR because in the original work, it did not alter the similarity tasks results, but they can be exchanged.

layers with dropout but a percentage of 0.3), however, the second hidden layer is followed by a batch normalization layer and the output is a single neuron with a sigmoid activation. The reason for the batch normalization layer was to stabilize the training. We also utilized a third and fourth conditional discriminator following (Tripathy et al., 2018), to leverage the cyclic architecture on paired data.

A novelty in RetroGAN is the combination of cyclic and non-cyclic optimization objectives: the regular adversarial loss for both GANs (L_{GAN}); the cyclic loss for both generators (L_{CYC}); the identity loss for both generators (L_{ID}); the max margin loss similar to (Weston et al., 2011; Ponti et al., 2018) for both the generators and additionally for the cycle of generators (L_{MM}); and the conditional cycle consistency loss (L_{cCYC}) introduced in (Tripathy et al., 2018). The combined objective has the following form:

$$\begin{aligned} L(G,F,D_X,D_Y) = L_{GAN}(G,D_Y,X,Y)+ \\ L_{GAN}(F,D_X,X,Y) + \lambda L_{CYC}(G,F)+ \\ \gamma L_{ID}(G,F,X,Y) + L_{MM}(G,F,X,Y)+ \\ \varsigma L_{cCYC}(G,F,D_{cX},D_{cY},X,Y) \end{aligned} \tag{1}$$

where $G : X \rightarrow Y$ is the generator that maps the source domain X of plain word embeddings to the target domain Y of retrofitted word embeddings; $F : Y \rightarrow X$ is the generator that does the opposite; D_X and D_Y are the discriminators for the corresponding domains; and D_{cX}, D_{cY} are our cycle conditional discriminators. For brevity, we only go into details on L_{MM} and L_{cCYC}. The other losses are the standard ones found in their respective works: L_{GAN} is the adversarial loss from (Goodfellow et al., 2014). L_{CYC} is the cycle consistency loss from (Zhu et al., 2017) with a scaling factor of λ (which we set to 1); and L_{ID} is the identity loss from (Zhu et al., 2017), which we scale with γ (which we set to 0.01). L_{ID} serves as a check of whether the embedding is already in the correct domain. L_{MM} is the max margin loss with random confounders as used by (Ponti et al., 2018), and as a novel aspect, we add a cyclic margin loss:

$$\begin{aligned} L_{MM}(G,F,X,Y) = \Sigma_{i=1}^{||x||}\Sigma_{j=1|j\neq i}^{k}\tau[\\ (\delta_{MM} - cos(G(x_i),y_i) + cos(G(x_i),y_j))+ \\ (\delta_{MM} - cos(F(y_i),x_i) + cos(F(y_i),x_j))+ \\ \boldsymbol{(\delta_{MM} - cos(G(F(y_i)),y_i) + cos(G(F(y_i)),y_j))+} \\ \boldsymbol{(\delta_{MM} - cos(F(G(x_i)),x_i) + cos(F(G(x_i)),x_j))}] \end{aligned} \tag{2}$$

Equation 2, intuitively, tries to make generated embeddings similar to their gold-standard and different from confounders. RetroGAN further enforces this constraint across the cycle. Lastly, we have L_{cCYC} which is the conditional cycle loss (Tripathy et al., 2018)[2], which we scale with ς (set to 1):

$$\begin{aligned} L_{cCYC}(G,F,D_{cX},D_{cY},X,Y) = \\ E_{x\sim p_{data}}[log(D_{cX}(G(x),x))]+ \\ E_{x\sim p_{data}}[log(1-D_{cX}(G(x),F(G(x))))]+ \\ E_{y\sim p_{data}}[log(D_{cY}(F(y),y))]+ \\ E_{x\sim p_{data}}[log(1-D_{cY}(F(y),G(F(y))))] \end{aligned} \tag{3}$$

3.2 Experimental Setup

To train our system we utilize the ADAM (Kingma and Ba, 2015) optimizer with a learning rate of 5e-5 for the generators and 1e-4 for the non-conditional discriminators. We do not train the discriminators used in the regular GAN loss, and instead train the ones in the conditional cycle consistency loss. We also note that we did not perform explicit fine tuning of the scaling parameters, but we will do so in future work through a grid search. We train for 312,500 mini-batches which is the equivalent to the AuxGAN training, using a batch size of 32.

In our tests we use the English Common Crawl FastText with sub-word information (FT-CC) and Numberbatch 19.08 (NB) to see how performance would be affected by using embeddings that were already retrofitted with a large KB. We ran the Attract-Repel (Mrkšić et al., 2017)[3] procedure on all these embeddings then proceeded to perform our post-specialization tests on learning the mapping from FT-CC to the resulting retrofitted embeddings.

We ran the word similarity benchmarks: SimLex (SL)(Hill et al., 2015) SimVerb (SV)(Gerz et al., 2016), and the Cambridge Rare Word (C660) dataset (Pilehvar et al., 2018). We utilize the Disjoint (evaluating words which were **not seen** in the constraints) and Full (evaluating words which were **seen** in constraints) settings from (Ponti et al., 2018) for SL and SV, and evaluate C660 on the Full setting to test performance on rare words. The maximum values for the similarity benchmarks while training are listed in table 2.

We trained the publicly available AuxGAN model on 10 epochs of 1M iterations (which in AuxGAN is a single embedding pair rather than a batch of pairs) with both plain stochastic gradient descent (SGD) and ADAM (learning rate of 0.1) and selected the best performing one (ADAM) to

---

[2] In future work we will additionally incorporate the paired conditional adversarial loss.

[3] We use the default settings found in `https://github.com/nmrksic/attract-repel`

	Disjoint				Full					
	FT-CC, A-R		FT-CC, A-R+NB		FT-CC, Attract-Repel			FT-CC, A-R+NB		
Models	SL	SV	SL	SV	SL	SV	C660	SL	SV	C660
Distributional	0.4644	0.3649	0.4499	0.3643	0.4644	0.3649	0.2973	0.4499	0.3643	0.1068
Attract-Repel	0.4644	0.3649	0.4499	0.3643	**0.7790**	**0.7632**	0.3768	0.7748	**0.7667**	0.2203
AuxGAN	**0.6127**	0.4641	0.6116	0.5331	0.6901	0.5756	0.3899	0.6565	0.5872	0.2088
RetroGAN	0.6028	**0.4702**	**0.6648**	**0.5971**	0.7717	0.7192	**0.5240**	**0.7960**	0.7483	**0.5581**

Table 2: *Word similarity tests results*: We run two distinct scenarios in which the words present in SL and SV are present (Full) in Attract-Repel (A-R) retrofitting constraints or not (Disjoint). The Distributional row represents the results of the tests using the publicly available embeddings (FT-CC). The results are the Spearman correlation (ρ) between the Cosine distance of the embeddings we are evaluating and the human similarity measurements. The results for AuxGAN and RetroGAN are the average of 3 runs.

	5%			10%			25%			50%		
	SL	SV	C660	SL	SV	C660	SL	SV	C660	SL	SV	C660
Attract-Repel	0.347	0.355	0.113	0.550	0.589	0.187	0.701	0.700	0.217	**0.759**	**0.747**	0.252
AuxGAN	0.615	0.510	0.453	0.667	0.569	0.470	0.679	0.581	0.475	0.685	0.600	0.490
RetroGAN	**0.624**	**0.538**	**0.489**	**0.701**	**0.652**	**0.493**	**0.738**	**0.690**	**0.502**	0.755	0.716	**0.511**

Table 3: Out of knowledge tests: We evaluate the performance of retrofitting and post specialization at varying percentages of test words seen in constraints. The full table can be seen in Appendix B

compensate for convergence speed discrepancies. Additionally, similar to (Ponti et al., 2018) we also evaluated on Light-LS (Glavaš and Štajner, 2015) with the default dataset (Horn et al., 2014) to test downstream performance. We evaluate the accuracy of simplification substitutions(i.e., the amount of words that are substituted correctly when compared to a gold standard). We utilize the first 500k words in FT-CC and the complete Numberbatch (generating the vocabulary's FastText embeddings as the distributional model). To test the retrofitted embeddings we substitute them in the original set.

3.3 Results & Discussion

RetroGAN outperforms AuxGAN in the majority of similarity benchmarks. We note that RetroGAN sets the state of the art on the rare-words benchmark (C660)(previously, to the best of our knowledge, it was 0.543 and 0.55 in (Yang et al., 2019; Fukuda, 2020)). In the similarity results for Full , we note the same observations that were noted in AuxGAN: there are some inconsistent gains and losses, which may be due to the combination of loss functions which may make the systems imprecise; although they spread the knowledge throughout the embeddings, they lose some precision when compared with the original retrofitted embeddings. The results for the lexical simplification (Light-LS) can be seen in table 4 where RetroGAN dominates.

We wanted to compare the out-of-knowledge (OOK) performance more in depth and to do this, we joined the words in SimLex (SL) and SimVerb (SV) and selected increasingly larger amounts of them ({5,10,25,50,75,100}%). We then selected the constraints that included these words, trained RetroGAN and AuxGAN with these constraints, and evaluated performance on SL, SV, and C660. Part of this can be seen in table 3. We see that RetroGAN's performance increases every time that new constraints are added, whereas AuxGAN's performance begins to peak after 25% of the constraints which may indicate more efficient knowledge distribution thanks to the cyclic system. Later on, the performance of RetroGAN kept increasing, but was less than the base retrofitted embeddings, possibly because of the lack of precision from the combination of losses. Lastly, we performed a small ablation study (Appendix A) on RetroGAN's losses. We note that the max-margin loss from (Ponti et al., 2018) is necessary for high performance in all the tests. We also notice that the cyclic (cyclic max-margin and cycle conditional discriminator) losses are essential for improved performance on the OOK and rare-word similarity benchmarks. We also see that the removal of the cyclic max-margin loss speeds up early learning and its addition stabilizes later learning respectively which may indicate a need to balance this. Future work will explore how to balance this losses, but it may be possible to put a scheduler to enable the loss after a peak. More details on the ablation study can be found in Appendix A.

Models	FT-CC	NB
Distributional	0.6553	0.6974
Attract-Repel	0.6993	0.6874
AuxGAN	0.7214	0.7335
RetroGAN	**0.7595**	**0.7735**

Table 4: Light-LS accuracy on (Horn et al., 2014)

4 Conclusion

This work presents an improvement on post-specialization work through the use of a CycleGAN-like system called RetroGAN. We show that RetroGAN gives improved performance in both the Full (words which were seen in knowledge/constraints) and the Disjoint (words which were **not** seen in the constraints) evaluation settings for three benchmarks. It additionally has better performance on a downstream lexical simplification task, further confirming its improved generalization ability. We conclude that RetroGAN is an improved system for post-specializing embeddings for rare and OOK concepts.[4]

5 Acknowledgements

This work was made possible thanks to the Media Lab Consortium funding.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

Nobukazu Fukuda. 2020. Calculation of distributed representation of unknown words based on surface similarity with known words `https://repository.dl.itc.u-tokyo.ac.jp/?action=repository_action_common_download&item_id=54229&item_no=1&attribute_id=14&file_no=1`.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182, Austin, Texas. Association for Computational Linguistics.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 63–68.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463.

Aishwarya Kamath, Jonas Pfeiffer, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019. Specializing distributional vectors of all words for lexical entailment. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 72–83, Florence, Italy. Association for Computational Linguistics.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning*, pages 1857–1865. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

---

[4] The system and the data can be accessed at `https://github.com/pedrocolon93/retrogan.git`

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Informing unsupervised pretraining with external linguistic knowledge. *arXiv preprint arXiv:1909.02339*.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. A system for massively parallel hyperparameter tuning. In *Proceedings of Machine Learning and Systems*, volume 2, pages 230–246.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the association for Computational Linguistics*, 5:309–324.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. 2018. Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1391–1401, Brussels, Belgium. Association for Computational Linguistics.

Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 282–293, Brussels, Belgium. Association for Computational Linguistics.

Robyn Speer and Joshua Chin. 2016. An ensemble method to produce high-quality word embeddings. *arXiv preprint arXiv:1604.01692*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. pages 4444–4451.

Soumya Tripathy, Juho Kannala, and Esa Rahtu. 2018. Learning image-to-image translation using paired and unpaired training samples. In *Asian Conference on Computer Vision*, pages 51–66. Springer.

Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 516–527, New Orleans, Louisiana. Association for Computational Linguistics.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1134–1145, New Orleans, Louisiana. Association for Computational Linguistics.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Ziyi Yang, Chenguang Zhu, Vin Sachidananda, and Eric Darve. 2019. Out-of-vocabulary embedding imputation with grounded language information by graph convolutional networks. *arXiv e-prints*, pages arXiv–1906.

Zhi-Xiu Ye, Qian Chen, Wen Wang, and Zhen-Hua Ling. 2019. Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models. *arXiv preprint arXiv:1908.06725*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

A Ablation Tests

We performed a small ablation study to examine how the multiple losses in RetroGAN affect performance. A one-by-one removal of these can be seen in figures 2(a),2(b),2(c),2(j),2(k),2(l). A toggle of each of the losses can be seen in: 2(a),2(b),2(c),2(j),2(k),2(l). The difference between the toggle and one-by-one removal is that in the toggle, we simple turn off the specified loss and leave the others untouched, whereas in the one-by-one removal we turn off one-by-one the losses, in this way we can see the individual effects, and the group effects. We evaluated the FT-CC and the Attract-Repel retrofitted FT-CC in the same scenarios as the evaluations before (Disjoint and Full). We note that the Disjoint setting for Card includes some of the words in the constraints.

The max margin loss utilized by (Ponti et al., 2018) (*one_way_maxmargin_loss*) is essential for high performance on the datasets. Without this loss, in all of the figures, we see that the scores in all our tests fall by at least by 0.1. This is seen in both the toggle and the one-by-one case. We can also see that the Cyclic version of this loss (*cycle_maxmargin_loss*) slows down learning initially, but stabilizes it in later iterations. We can see that by removing it we get higher performance in earlier iterations but the performance decays as more iterations are given. This may be because it tries to enforce that the semantic components of the embeddings be similar after going through the cycle, but it may be a hard objective to achieve. This loss is especially useful for the rare-word Card-660 evaluation. By looking at the toggle ablation test for this loss, we can see that it indeed can lead to better earlier performance, however it decays with time.

The identity loss (*id_loss*) helps to stabilize the training in later iterations. Removal of this loss significantly affects the disjoint settings, and the reason for it may be that it gives some indication of the important semantic components of the vectors that are being post-specialized. This in the disjoint setting leads to significant performance reductions on the later iterations. Interestingly enough, by toggling off only this loss, we can see that it leads to better performance, which means that with all the other losses, it may contain redundant information that may hinder performance, however if the model relies on the loss without other losses, its information is useful.

The Cycle Conditional discriminator loss (*cycle_discriminator_loss*) also contributes to the stability and generalization of the later learning. Removing this loss does not improve early learning, save on the Card-660 dataset, and in most of the other tests, there is not a large noticeable difference. However, on the disjoint setting we do see that it performance decays in later iterations. We suspect the conditioning helps slightly in the stabilization, and generalization of the system, but its effect is not too much.

The Cycle Loss (*cycle_mae_loss*), also stabilizes and helps in the generalization of our system. We can see in the disjoint settings in particular, that its removal hinders the model in later iterations. We suspect that since the consistency is not being enforced, the model does not learn effectively to preserve important, possibly non-semantic, parts from the distributional and the retrofitted domain.

As a practical recommendation, we suggest removing the cyclic max-margin loss either completely (pausing the training early at it's peak around 50k-100k iterations), or toggling it after this initial training to get the speedup and the generation. Another practical recommendation may be to disable the identity loss all by itself. The other losses can be maintained as they are described in this work.

B Out-of-knowledge Scalability Tests

In table 5 we test the performance of post specialization as more constraints are added into the retrofitting process. We note that AuxGAN's performance saturates after 50% whereas RetroGAN keeps learning, albeit less accurately than the retrofitting system. These tests were run for 100k batches on RetroGAN and for 10M iterations (312500 RetroGAN batches) on AuxGAN.

C Additional embedding pre-processing

Input and output vectors are divided by the Euclidean (2) norm. This helps slightly in the performance of the semantic comparison benchmarks. No other pre-processing is done on the vectors.

D Architecture Details

In figure 1, we can see the architecture that RetroGAN uses. On a training step, the losses are calculated as follows. For the cyclic losses, the system samples embeddings from the distributional embeddings and their retrofitted counterparts and these samples are passed to the generators (1, 5 in the figure). Then, the generators' output is passed to the counterpart generator (Distributional Generator passes to Retrofitted Generator and vice versa, seen as 3 in the figure). The output of this is then used to calculate the max margin loss, and passed on to the subsequent discriminator to calculate the cycle discriminator loss (2, 4 in the figure). In addition to this, after going through the cycle of generators (1 or 5, 3 in the figure) we train the conditional discriminators by conditioning on real inputs from the retrofitted or distributional embeddings, or by conditioning on fake inputs (6,7 in the figure).

The amount of parameters in each model and the layers can be found in table 6.

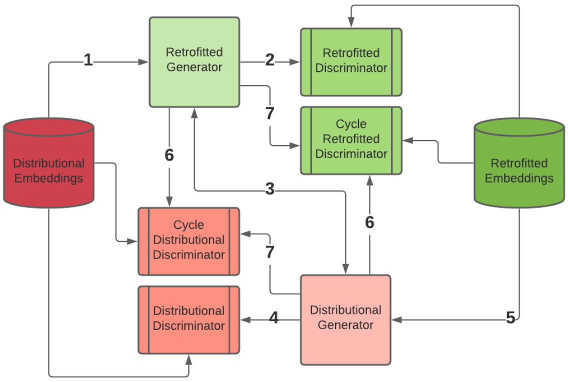

Figure 1: Architecture diagram for RetroGAN

E Parameter Tuning

We performed a parameter tuning using the Ray tuning library, to try and generate a configuration that would be optimal for RetroGAN. We utilized the ASHA Scheduler(Li et al., 2020) along with the following search space configuration:

```
config = {
"g_lr" : tune.qloguniform(0.00005,
.1,0.00005),
"d_lr" : tune.qloguniform(0.00005,
.1,0.00005),
"one_way_mm":True,
"cycle_mm":True,
"cycle_dis":True,
"id_loss":True,
"cycle_loss":True,
"batch_size":tune.choice([16,32,64]),
"generator_size":tune.choice([512,
1024,2048]),
"discriminator_size":
tune.choice([512,1024,2048]),
"generator_hidden_layers":
tune.choice([1,2,3]),
"discriminator_hidden_layers":
tune.choice([1,2,3]),
"dis_train_amount":
tune.choice([1,2,3])
}
```

We used the SimVerb score to guide the parameter optimization, because it was the score that involved the largest sample of words. We also utilized a machine with two Intel processors with 48 cores in total, 128GB of RAM, an NVIDIA P6000 and a NVIDIA 1080TI. We used 25 samples in the optimization due to time constraints, although this value can be expanded more. We also ran this for 35 epochs, because the performance after that would become relatively stable and not increase greatly.

The best results from the optimization are the following:

```
{
’g_lr’: 0.00495,
’d_lr’: 0.00885,
’one_way_mm’: True,
’cycle_mm’: True,
’cycle_dis’: True,
’id_loss’: True,
’cycle_loss’: True,
’batch_size’: 32,
’generator_size’: 2048,
’discriminator_size’: 2048,
’generator_hidden_layers’: 1,
’discriminator_hidden_layers’: 3,
’dis_train_amount’: 1
}
```

	5%			10%			25%		
Models	SL	SV	C660	SL	SV	C660	SL	SV	C660
Attract-Repel	0.347	0.355	0.113	0.550	0.589	0.187	0.701	0.700	0.217
AuxGAN	0.615	0.510	0.453	0.667	0.569	0.470	0.679	0.581	0.475
RetroGAN	**0.624**	**0.538**	**0.489**	**0.701**	**0.652**	**0.493**	**0.738**	**0.690**	**0.502**

	50%			75%			100%		
Models	SL	SV	C660	SL	SV	C660	SL	SV	C660
Attract-Repel	**0.759**	**0.747**	0.252	**0.766**	**0.757**	0.244	**0.771**	**0.761**	0.257
AuxGAN	0.685	0.600	0.490	0.688	0.597	0.480	0.690	0.601	0.486
RetroGAN	0.755	0.716	**0.511**	0.763	0.721	**0.507**	0.762	0.715	**0.509**

Table 5: Performance of post specialization and retrofitting as more constraints are added to a system.

Model	Amount of Parameters	Layers
Generator	5,427,500(*2)	Linear (300x2048), ReLU, Dropout (0.3), Linear (2048x2048),ReLU Dropout (0.3), Linear (2048x300)
Discriminator	4,818,945(*2)	Linear (300x2048), ReLU, Dropout (0.3), Linear (2048x2048),ReLU, Batch Norm, Dropout (0.3), Linear (2048x1), Sigmoid
Cycle Conditional Discriminator	5,433,345(*2)	Linear (600x2048), ReLU, Dropout (0.3), Linear (2048x2048), ReLU, Batch Norm, Dropout (0.3), Linear (2048x1), Sigmoid
Total	31,359,580	-

Table 6: Amount of trainable parameters and layers in RetroGAN

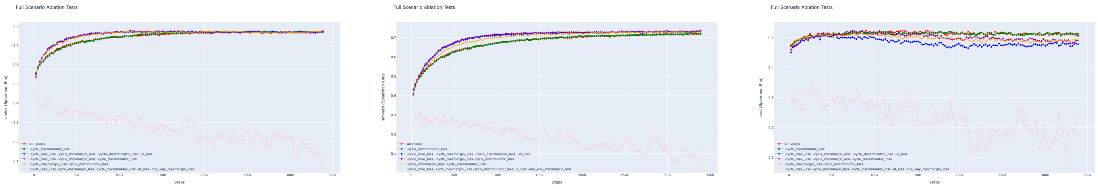

(a) SimLex Ablation Test-Full Scenario

(b) SimVerb Ablation Test-Full Scenario

(c) CARD-660 Ablation Test-Full Scenario

(d) SimLex Ablation Test-Disjoint Scenario

(e) SimVerb Ablation Test-Disjoint Scenario

(f) CARD-660 Ablation Test-Disjoint Scenario

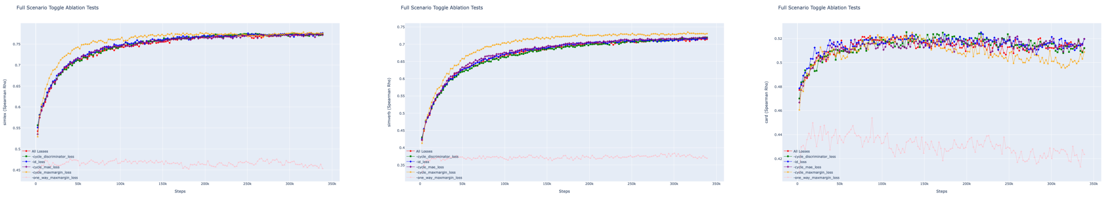

(g) SimLex Ablation Toggle Test-Disjoint Scenario

(h) SimVerb Ablation Toggle Test-Disjoint Scenario

(i) CARD-660 Ablation Toggle Test-Disjoint Scenario

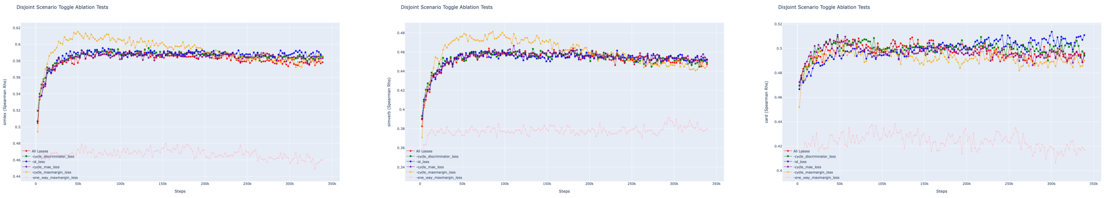

(j) SimLex Ablation Toggle Test-Disjoint Scenario

(k) SimVerb Ablation Toggle Test-Disjoint Scenario

(l) CARD-660 Ablation Toggle Test-Disjoint Scenario

Figure 2: Ablation test results for SimLex, SimVerb, and CARD-660. A higher resolution version can be found in the repository at: `https://github.com/pedrocolon93/retrogan.git`

Fusion: Towards Automated ICD Coding via Feature Compression

Junyu Luo
Pennsylvania State University
junyu@psu.edu

Cao Xiao
Amplitude
danica.xiao@amplitude.com

Lucas Glass
IQVIA
lucas.glass@iqvia.com

Jimeng Sun
UIUC
jimeng@illinois.edu

Fenglong Ma[*]
Pennsylvania State University
fenglong@psu.edu

Abstract

ICD coding aims to automatically assign International Classification of Diseases (ICD) codes from unstructured clinical notes or discharge summaries, which saves human labor and reduces errors. Although several studies are proposed to solve this challenging task, none distinguishes the importance of different phrases with a word window. Intuitively, informative phrases should be more useful for the prediction. This paper proposes a feature compressed ICD coding model named Fusion to address this issue. In particular, we propose an attentive soft-pooling approach to compress the sparse and redundant word representations into informative and dense ones as local features. Besides, we use the key-query attention mechanism for modeling the inner relations among local features to generate the global features, which are further used to predict ICD codes. Experiments on two widely used datasets demonstrate that Fusion outperforms baselines. However, on the MIMIC-III Full dataset, we find that none of the state-of-the-art approaches significantly perform better than others. Thus, automated ICD coding is still a challenging task.

1 Introduction

The International Classification of Diseases (ICD) coding system helps standardize the recording of diagnoses and treatments assigned to patients by medical professionals in the world. These ICD codes are generated from massive unstructured clinical notes. However, manual code assignments is labor-intensive and prone to errors. Thus, automatic ICD code assignment becomes an urgent need in the healthcare domain.

Traditional machine learning methods (Larkey and Croft, 1996) tried to tackle this task based on feature extraction. However, it does not work well since clinical notes are noisy and complex. Recently, deep learning-based approaches (Cao et al., 2020; Xie et al., 2019; Li and Yu, 2020; Mullenbach et al., 2018) are proposed to improve its performance. Among others, convolutional methods (Cao et al., 2020; Xie et al., 2019; Li and Yu, 2020; Mullenbach et al., 2018) outperform other approaches. Besides, some studies try to incorporate external information to further improve the performance (Cao et al., 2020; Xie et al., 2019). However, they still suffer the following issues.

- **Redundant Information Deduction**. The clinical notes are noisy and complex, where only some key phrases are highly related to the coding. However, convolutional methods treat all the word windows equally, ignoring that different words have different importance and should be weighted differently within word windows. Besides, the sliding windows used in the convolutional methods produce a lot of redundant information. Thus, it is important to reduce the non-informative and redundant information and distinguish the contributions of different convolutional features.

- **Interactions among Local Features**. Most existing approaches such as MultiResCNN (Li and Yu, 2020) only use the local features for coding obtained using different filters. However, they ignore the importance of interactions among different local features. For example, *sleep apnoea (OSA)* and *insomnia* are related to *hypertension* and *ischaemic heart disease* (Harrison and Wood, 1949). Thus, combining different local features may discover new useful patterns to improve coding.

To tackle these issues, we propose a feature compressed ICD coding model named Fusion, which can automatically compress the local fea-

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2096–2101
August 1–6, 2021.

tures and further learn global features to enhance the coding performance. In particular, Fusion takes an attention-based soft-pooling approach to compress local features learned by word convolutions, passing residual convolution blocks. By aggregating all the local features from different convolutional filters, Fusion then applies key-query attention mechanism to model interactions among local features and obtain global ones. A code-wise attention mechanism is then used to learn a feature vector associated with each ICD code. This vector is finally used to make a prediction. Experiments on two public datasets show that Fusion outperforms state-of-the-art baselines over five evaluation metrics. Moreover, we find that none of the existing approaches outperforms others on the MIMIC-III Full dataset. Thus, automated ICD coding is still an open challenge.

2 Related Work

Traditional machine learning models have been applied to automatically extract ICD codes using the hand-crafted feature vectors as the inputs (Larkey and Croft, 1996; Gundersen et al., 1996; Franz et al., 2000; Pestian et al., 2007; Farkas and Szarvas, 2008). However, they did not achieve satisfactory performance due to the difficulty of extracting useful features from complex and noisy clinical notes. Deep learning models have shown their superiority for this task, including recurrent-based deep models (Shi et al., 2017; Li et al., 2018; Xu et al., 2019) and convolution-based models (Kim, 2014; Mullenbach et al., 2018; Cao et al., 2020; Li and Yu, 2020). In general, convolutional models perform better than recurrent-based ones. Several studies try to incorporate advanced pretrained language model BERT (Devlin et al., 2019), ICD code descriptions (Wang et al., 2018; Mullenbach et al., 2018; Xie and Xing, 2018; Li and Yu, 2020), ICD code structure (Wang et al., 2020; Cao et al., 2020), and knowledge graph (Cao et al., 2020; Xie et al., 2019) to improve the performance.

3 Model

The goal of automated ICD coding is to predict a set of unique ICD codes $\mathcal{Y}$ from the code set $\mathcal{C} = \{c_1, c_2, \cdots, c_s\}$ when given clinical note $D = \{w_1, w_2, \cdots, w_n\}$, where $\mathcal{Y} \subseteq \mathcal{C}$, s is the number of unique ICD codes, and n is the number of words in D. This task is challenging since s is very large, which is over 15,000 for ICD-9 codes and 60,000 for ICD-10 codes, respectively. Besides, extensive noisy information exists in the clinical note D.

To solve these challenges, we propose a feature denoised model (Fusion) for automated ICD coding as shown in Figure 1. This model consists of five modules: the input layer, the compressed convolutional layer, the feature aggregation layer, the code-wise attention layer, and the prediction layer. Next, we introduce the details of each module in the following subsections.

3.1 Input Layer

We take the clinical note $D = \{w_1, w_2, \cdots, w_n\}$ as the model input. For each unique word w_i, word2vec (Mikolov et al., 2013) is used to pretrain its embedding, which is denoted as $\mathbf{e}_i$, a d_e-dimensional embedding. Thus, the input of Fusion is a matrix $\mathbf{D} = \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n\}$.

3.2 Compressed Convolutional Layer

Given the input data $\mathbf{D}$, the compressed convolutional layer aims to learn dense and informative word representations, which are further used to learn the clinical note representation. In particular, we first use convolutional neural networks (CNN) to learn word representations and then propose an attention-based soft-pooling approach to compress those representations. Finally, residual convolution blocks (He et al., 2016) are introduced as MultiResCNN (Li and Yu, 2020) on top of the compressed features.

3.2.1 Word Convolution

CNNs are powerful for text classification tasks (Kim, 2014) that they have multiple filters with different kernel sizes (i.e., word windows) to capture diverse patterns. Let m be the number of filters. The kernel of each filter f_i is denoted as k_i. Thus, we can apply m different 1-dimensional convolutions on the input data $\mathbf{D}$. For the i-th filter, we have

$$\mathbf{x}_j^i = conv(\{\mathbf{e}_j, \mathbf{e}_{j+1}, \cdots, \mathbf{e}_{j+k_i-1}\}; \mathbf{W}_x^i), \quad (1)$$

where $conv(\cdot;\cdot)$ represents the 1-dimensional convolutional operation, and $\mathbf{W}_x^i$ denotes the learned parameter.

3.2.2 Attention-based Soft-pooling

The word convolutional operation uses sliding windows, which produces redundant information existing in adjacent word representations. Thus, to

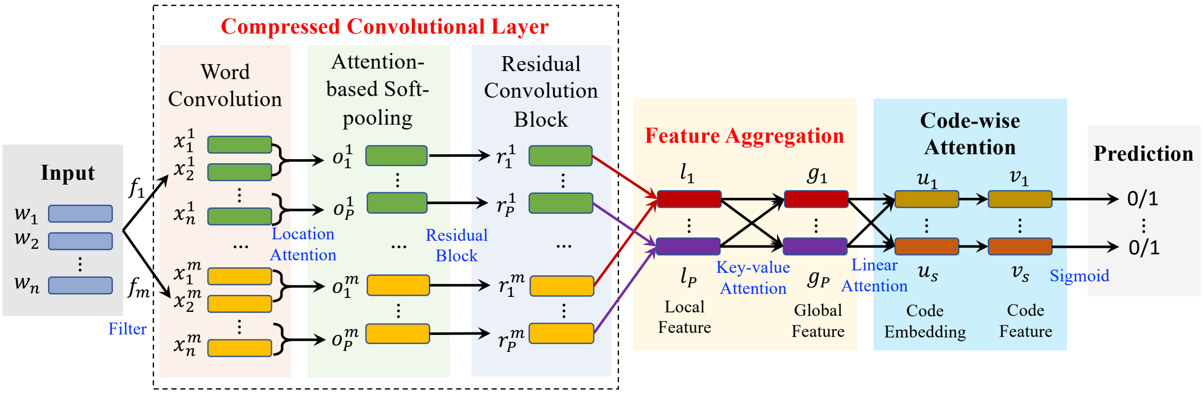

Figure 1: Overview of the proposed Fusion.

remove such information, we propose to compress word representations learned by Eq. (1) via an attention-based soft-pooling operation.

Given a word w_j, its neighboring words $\{w_{j+1}, \cdots, w_{j+g-1}\}$, and the corresponding filter f_i, we first learn the local-based attention scores $\alpha_j^i = \mathbf{W}_\alpha^i \mathbf{x}_j^i + b$ with softmax function, i.e., $[\beta_j^i, \cdots, \beta_{j+g-1}^i] = softmax([\alpha_j^i, \cdots, \alpha_{j+g-1}^i])$, where $\mathbf{W}_\alpha^i$ and b are learnable parameters. Then we conduct attention-based soft-pooling on the g words and obtain the compressed representation as in Eq. (2).

$$\mathbf{o}_p^i = \sum_{q=j}^{j+g-1} \beta_q^i \mathbf{x}_q^i \tag{2}$$

In such a way, the whole n word representations learned by Eq. (1) will be replaced by $P = \lceil \frac{n}{g} \rceil$ new representations, i.e., $\{\mathbf{o}_1^i, \mathbf{o}_2^i, \cdots, \mathbf{o}_P^i\}$. In such a way, we can reduce the number of word representations and obtain more dense ones.

3.2.3 Residual Convolution Block

For each filter f_i, we now have a denoised matrix $\{\mathbf{o}_1^i, \mathbf{o}_2^i, \cdots, \mathbf{o}_P^i\}$ that represents the input $\mathbf{D}$. To avoid vanishing gradients and train the model easier, we also introduce residual blocks on top of the compressed features. In particular, we replace the batch norm layer with the group norm layer. Let a denotes the number of residual blocks, and we have $\mathbf{r}_p^i = ResidualBlcok(\{\mathbf{o}_p^i, \cdots, \mathbf{o}_{p+a-1}^i\})$.

3.3 Feature Aggregation Layer

Since m filters are used to obtain m compressed features, we concatenate them together as the local features, i.e., $\mathbf{l}_p = [\mathbf{r}_p^1, \cdots, \mathbf{r}_p^m]$. Then the whole document can be represented by a matrix $\mathbf{D}_l = \{\mathbf{l}_1, \mathbf{l}_2, \cdots, \mathbf{l}_P\}$. However, such an aggregation only takes local information into account but ignores the interactions with the remaining words.

Thus, we propose to use the key-query attention mechanism (Vaswani et al., 2017) to learn a global feature representation for each compressed word window. Thus, we have the global features $\mathbf{D}_g = [\mathbf{g}_1, \cdots, \mathbf{g}_P] = attention([\mathbf{l}_1, \cdots, \mathbf{l}_P])$.

3.4 Code-wise Attention Layer

Due to a large number of labels, directly using the global features $\mathbf{D}_g$ to make predictions may not perform well. Thus, we use a code-wise attention layer to generate a matching vector for each ICD code used to make a prediction. Let $\mathbf{u}_k$ represent the embedding of the k-th ICD code, i.e., c_k. Then we calculate the attention weights on all the global features using $\mathbf{u}_k$, i.e., $[\gamma_1^k, \cdots, \gamma_P^k] = softmax([\mathbf{u}_k \mathbf{g}_1, \cdots, \mathbf{u}_k \mathbf{g}_P])$. Then the code-wise vector can be obtained by $\mathbf{v}_k = \sum_{p=1}^{P} \gamma_p^k \mathbf{g}_p$.

3.5 Prediction Layer

Using the code-wise vector $\mathbf{v}_k$, we can make a prediction using the sigmoid function, i.e.,

$$\tilde{y}_k = (1 + \exp(\mathbf{w}_k^\top \mathbf{v}_k))^{-1}, \tag{3}$$

where $\mathbf{w}_k$ is the learnable parameter vector. Finally, cross-entropy loss function on a specific clinical note $\mathbf{D}$ is used to optimize the proposed model.

$$\mathcal{L} = -\sum_{k=1}^{s} (y_k \log(\tilde{y}_k) + (1 - y_k) \log(1 - \tilde{y}_k)). \tag{4}$$

4 Experiment

4.1 Datasets

We conduct experiments on two public datasets MIMIC-III 50 and MIMIC-III Full (Johnson et al.,

Dataset		MIMIC-III 50					MIMIC-III Full				
Setting	Model	AUC		F1		P@N	AUC		F1		P@N
		Macro	Micro	Macro	Micro	5	Macro	Micro	Macro	Micro	8
Note Only	Fusion	**0.931**	**0.950**	**0.683**	**0.725**	**0.679**	0.915	0.987	0.083	**0.554**	**0.736**
	C-MemNN	0.833	–	–	–	0.420	–	–	–	–	–
	C-LSTM-ATT	–	0.900	–	0.532	–	–	–	–	–	–
	CAML	0.875	0.909	0.532	0.614	0.609	0.895	0.986	0.088	0.539	0.709
	DR-CAML	0.884	0.916	0.576	0.633	0.618	0.897	0.985	0.086	0.529	0.690
	MultiResCNN	0.899	0.928	0.606	0.670	0.641	0.910	0.986	0.085	0.552	0.734
Note + Ontology	HyperCore	0.895	0.929	0.609	0.663	0.632	**0.930**	0.989	**0.090**	0.551	0.722
	MSATT-KG	0.914	0.936	0.638	0.684	0.644	0.910	**0.992**	**0.090**	0.553	0.728

Table 1: Experiment results on MIMIC-III 50 and MIMIC-III Full datasets.

2016) to extract ICD-9 codes from discharge summaries. We use the same setting as previous works (Mullenbach et al., 2018; Shi et al., 2017; Li and Yu, 2020; Cao et al., 2020). The MIMIC-III 50 dataset contains the top 50 most frequent codes, 8,067, 1,574, and 1,730 discharge summaries for training, development, and testing, respectively. The MIMIC-III Full dataset consists of 8,921 codes, 47,719, 1,631, and 3,372 discharge summaries for training, development, and testing, respectively. The number of labels on the MIMIC-III Full dataset is significantly greater than that on the MIMIC-III 50 dataset, making the task more difficult.

4.2 Metrics and Parameter Settings

We follow previous work (Mullenbach et al., 2018) and use Micro Macro AUC (area under the ROC), Micro Macro F1, and Precision@K scores as metrics. For MIMIC-III 50, we report Precision@5 (P@5) and P@8 for MIMIC-III Full. We use the same parameter setting as MultiResCNN (Li and Yu, 2020)[1], and set g as 2 in our experiments, i.e., compress two features together.

4.3 Baselines

Existing studies either only take clinical notes as the inputs or incorporate external information, working with notes to enhance the performance. Our work belongs to the first category. For the "note only" category, we employ C-MemNN (Prakash et al., 2017), C-LSTM-ATT (Shi et al., 2017), CAML (Mullenbach et al., 2018), DR-CAML (Mullenbach et al., 2018), and MultiResCNN (Li and Yu, 2020) as baselines. We also use HyperCore (Cao et al., 2020), and MSATT-KG (Xie et al., 2019) as baselines, which incorporate the ICD code ontology to enhance the performance. Since all the approaches use the same settings, we directly use the results reported in the original papers.

4.4 Performance Analysis

Table 1 shows the experimental results of all approaches in terms of different metrics. We can observe that Fusion outperforms all the baselines on the MIMIC-III 50 dataset in terms of all metrics. Compared to the best baselines, the scores of Macro Micro F1 and P@5 obtained by Fusion improve over 7%, 6%, and 5%, respectively. These results clearly demonstrate the effectiveness of the proposed feature compression and aggregation approaches for the automated ICD coding task.

Compared with the baselines only taking notes as the input, the proposed Fusion achieves the highest scores on the MIMIC-III Full dataset. Although HyperCore and MSATT-KG incorporate external information to improve the performance, the performance of Fusion is still comparable. We also can observe that on the MIMIC-III Full dataset, none of the methods can be significantly better than others. The reason may be that all the models cannot be trained sufficiently with the huge number of ICD code labels on noisy, sparse, and unstructured medical clinical notes, which makes this task more challenging.

4.5 Ablation Study

In this section, we remove parts of the full Fusion model to validate the contribution of each individual module. Table 2 shows the ablation study results. "MaxPool" means replacing our soft-pooling layer with the traditional max-pooling layer. As shown in Table 2, the results drop on all metrics, which indicates the importance and benefits of using the proposed soft-pooling layer. Max-pooling will lose part of critical information during the compression and is not differentiable. With soft-pooling, the key information can be better preserved during the compression process, since

---

[1] `https://bit.ly/3opDmjM`

Dataset	MIMIC-III 50					MIMIC-III Full				
Model	AUC		F1		P@N	AUC		F1		P@N
	Macro	Micro	Macro	Micro	5	Macro	Micro	Macro	Micro	8
Fusion	0.931	0.950	0.683	0.725	0.679	0.915	0.987	0.083	0.554	0.736
MaxPool	0.921	0.942	0.664	0.710	0.668	0.900	0.986	0.081	0.552	0.726
DocLevel	0.895	0.923	0.597	0.652	0.641	-	-	-	-	-

Table 2: Ablation Experiment results on MIMIC-III 50 and MIMIC-III Full datasets.

the selection process is guided by the gradient.

"DocLevel" refers to replacing the code-wise attention layer with the single document-level attention. The attention is based on the document feature, and all codes use the same attention weight during the prediction instead of calculating code-specific attention weights. Thus, all codes will use the same feature for the prediction. In such a way, much unrelated information will also be kept. For example, we do not want to preserve the heart-failure-related information while predicting the COPD code. As a result, most scores drop significantly compared to the original design. The introduction of the code-specific attention makes it possible that the predictor can dynamically adjust the attentions based on the cases. Thus, the redundant information can be better removed with our design.

5 Conclusion

In this paper, we propose Fusion for the automated ICD coding task. In particular, Fusion focuses on compressing redundant feature information, distinguishing the importance of adjacent phrases, and considering interactions among local features. We conduct experiments on two widely-used datasets to show the effectiveness of Fusion in terms of five evaluation metrics. From experimental results on the MIMIC-III Full dataset, we find that automated ICD coding is still challenging due to the noisy data and a large number of ICD code labels.

References

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Shengping Liu, and Weifeng Chong. 2020. Hypercore: Hyperbolic and co-graph representation for automatic icd coding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3105–3114.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Richárd Farkas and György Szarvas. 2008. Automatic construction of rule-based icd-9-cm coding systems. In *BMC bioinformatics*, volume 9, page S10. Springer.

Pius Franz, Albrecht Zaiss, Stefan Schulz, Udo Hahn, and Rüdiger Klar. 2000. Automated coding of diagnoses–three methods compared. In *Proceedings of the AMIA Symposium*, page 250. American Medical Informatics Association.

Michael L Gundersen, Peter J Haug, T Allan Pryor, Rudy van Bree, Spence Koehler, Kay Bauer, and Brenda Clemons. 1996. Development and evaluation of a computerized admission diagnoses encoding system. *Computers and Biomedical Research*, 29(5):351–372.

CV Harrison and Paul Wood. 1949. Hypertensive and ischaemic heart disease: A comparative clinical and pathological study. *British heart journal*, 11(3):205.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Leah S Larkey and W Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 289–297.

Fei Li and Hong Yu. 2020. Icd coding from clinical text using multi-filter residual convolutional neural network. In *AAAI*, pages 8180–8187.

Min Li, Zhihui Fei, Min Zeng, Fang-Xiang Wu, Yaohang Li, Yi Pan, and Jianxin Wang. 2018. Automated icd-9 coding via a deep learning approach. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(4):1193–1202.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *NAACL-HLT*.

John Pestian, Chris Brew, Pawel Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Wlodzislaw Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Biological, translational, and clinical language processing*, pages 97–104.

Aaditya Prakash, Siyuan Zhao, Sadid Hasan, Vivek Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2017. Condensed memory networks for clinical diagnostic inferencing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P Xing. 2017. Towards automated icd coding using deep learning. *arXiv preprint arXiv:1711.04075*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Shanshan Wang, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jian-Yun Nie, Jun Ma, and Maarten de Rijke. 2020. Coding electronic health records with adversarial reinforcement path generation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 801–810.

Pengtao Xie and Eric Xing. 2018. A neural architecture for automated icd coding. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1066–1076.

Xiancheng Xie, Yun Xiong, Philip S Yu, and Yangyong Zhu. 2019. Ehr coding with multi-scale feature attention and structured knowledge graph propagation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 649–658.

Keyang Xu, Mike Lam, Jingzhi Pang, Xin Gao, Charlotte Band, Piyush Mathur, Frank Papay, Ashish K Khanna, Jacek B Cywinski, Kamal Maheshwari, et al. 2019. Multimodal machine learning for automated icd coding. In *Machine Learning for Healthcare Conference*, pages 197–215. PMLR.

Automatic Document Sketching: Generating Drafts from Analogous Texts

Zeqiu Wu[◇] **Michel Galley**[♠] **Chris Brockett**[♠] **Yizhe Zhang**[♠] **Bill Dolan**[♠]
[◇]University of Washington [♠]Microsoft Research
zeqiuwu1@washington.edu
{mgalley,chrisbkt,yizzhang,billdol}@microsoft.com

Abstract

The advent of large pre-trained language models has made it possible to make high-quality predictions on how to add or change a sentence in a document. However, the high branching factor inherent to text generation impedes the ability of even the strongest language models to offer useful editing suggestions at a more global or *document* level. We introduce a new task, DOCUMENT SKETCHING, which involves generating entire draft documents for the writer to review and revise. These drafts are built from sets of documents that overlap in form – sharing large segments of potentially reusable text – while diverging in content. To support this task, we introduce a Wikipedia-based dataset of analogous documents and investigate the application of weakly supervised methods, including use of a transformer-based mixture of experts, together with reinforcement learning. We report experiments using automated and human evaluation methods and discuss relative merits of these models.

1992 Democratic National Convention

2016 Democratic National Convention

From Wikipedia, the free encyclopedia

The **2016 Democratic National Convention** was a presidential nominating convention, held at the Wells Fargo Center in Philadelphia, Pennsylvania, from July 25 to 28, 2016. The convention gathered delegates of the Democratic Party, the majority of them elected through a preceding series of primaries and caucuses, to nominate a candidate for president and vice president in the 2016 United States presidential election. Former U.S. Secretary of State Hillary Clinton was chosen as the party's nominee for president by a 54% majority of delegates present at the convention roll call, defeating primary rival Senator Bernie Sanders, who received 46% of votes from delegates, and becoming the first female candidate to be formally nominated for president by a major political party in the United States. Her running mate, Senator Tim Kaine from Virginia, was confirmed by delegates as the party's nominee for vice president by acclamation.

... Democratic National Convention

The ... Democratic National Convention was a presidential nominating convention, held at ... , from ... to ... , in which delegates of the Democratic Party nominated ... for president and ... for vice president in the ... presidential election. ...

Figure 1: The right side shows a sketch for writing the report of a future democratic national convention, generated from a pile of previous reports.

1 Introduction

Large pre-trained language models such as T5 and GPT-3 (Raffel et al., 2019; Brown et al., 2020) have enabled impressive progress on a variety of natural language generation tasks by producing fluent, coherent long texts (Rashkin et al., 2020; Zellers et al., 2019). While automated document-level generation seems tantalizingly within reach, a high branching factor presents significant challenges in tailoring generated documents to the specific requirements of users. Topic drift and "hallucination" of information are endemic to these models (Wiseman et al., 2017). These risks have ensured that end-user applications involving text generation (e.g., Smart Compose, Smart Reply, Grammarly) still require a human to remain in control of content and are restricted to individual sentences or even smaller segments of text (Chen et al., 2019; Kannan et al., 2016; Alikaniotis and Raheja, 2019; Prabhumoye et al., 2019; Faltings et al., 2021).

Can large generative language models be used to assist user writing at the document level while the user still controls the factual content? A possible answer lies in the observation that a substantial portion of day-to-day writing involves some form of reuse. Similar documents (e.g., monthly reports, sales letters, job descriptions) are effectively recycled by changing those segments that need to be modified (Fig. 1).[1] Moreover, documents containing analogous texts are often found collocated in repositories, a common practice for organizations that manage professional documents.[2] The high branching factor that impedes the application of long-form generation might thus be mitigated if models were to exploit conventionally-structured analogous "reusable" texts to produce a "sketch" that captures prototypical document-level text structure. In this view, initial sketches would assist authors by reducing the manual editing effort

---

[1] Evidenced by the plethora of companies offering reusable business templates for enterprise use, e.g., https://www.businesswire.com/news/home/20201028005573/en/.

[2] For instance, monthly reports of world trade in grains are organized chronologically at https://usda.library.cornell.edu/concern/publications/zs25x844t?locale=en

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2102–2113
August 1–6, 2021. ©2021 Association for Computational Linguistics

(planning, inserting and deleting content, etc.). A good sketch would reflect structural patterns and reusable text, and provide indication, beyond static boilerplate, of locations where user input might be warranted. This could be especially beneficial for novice writers who would otherwise have to read many analogous documents before developing a full picture of what is entailed in writing such documents.[3] A fully-implemented dynamic system might update other portions of the document to reflect modifications introduced by the user.

In this work, we propose a new task, called DOCUMENT SKETCHING, in which initial template-like prototype documents are generated from collections of analogous documents. To support this task, we collected a dataset consisting of approximately 20K Wikipedia documents with similar textual characteristics.[4] For this new task, inspired by previous work in measuring machine translation post-editing productivity (Tatsumi, 2009; Specia and Farzindar, 2010), we define an automatic evaluation metric based on Word Error Rate (Snover et al., 2006; Tomás et al., 2003). We compare against strong baseline models including a mixture of experts model and a reinforcement learning approach designed to handle multi-source inputs and a weak supervision setting. Finally, we provide experimental analysis of these models, using automated and human evaluation studies.

2 Related Work

2.1 Document Generation

Recent work leverages the success of large pretrained language models to generate long texts such as stories (Rashkin et al., 2020), reviews (Cho et al., 2019a,b) and fake news (Zellers et al., 2019). Most end-user applications for assisting user writing, however, are confined to sentence-level generation (Chen et al., 2019; Kannan et al., 2016; Alikaniotis and Raheja, 2019; Prabhumoye et al., 2019; Faltings et al., 2021). Our work focuses on *document-level* writing assistance in which a document sketch is constructed from a set of similar documents.

2.2 Template-Based Generation

Some existing work induces templates as an intermediate step for performing tasks like text summarization or response generation. Most use a retrieval-based method to extract similar references from the training corpus as prototypes (Cao et al., 2018; Yang et al., 2019; Wang et al., 2019; Gao et al., 2019; Peng et al., 2019), and learn to separate salient information and latent template structure. Cai et al. (2019) induce an intermediate template for response generation explicitly, but from a single retrieved relevant response. Similar prototype editing work (Guu et al., 2018; Hashimoto et al., 2018; Fabbri et al., 2020) focuses on short text (e.g., a question or a single sentence) or structured output (e.g., code snippet) editing. Oya et al. (2014); Magooda and Litman (2019); Yang et al. (2020); Li et al. (2018) convert each single input text into a template with blanks using rule-based methods. Other work such as (Wiseman et al., 2018) and (Gangadharaiah and Narayanaswamy, 2020) relies on a knowledge base or a domain/task specific ontology to segment text sequences into templates.

2.3 Multi-Sequence Processing

Multiple Sequence Alignment (MSA), widely used in the biological domain (Sauder et al., 2000) to align multiple biological sequences like proteins, has long been leveraged for text pattern matching (Barzilay and Lee, 2003; Alonso et al., 2004). We adopt this method to align input documents and create heuristic templates as weak supervision.

Other tasks taking multiple text sequences as input include multi-document summarization, which seeks to generate an abstractive text summary of multiple input documents (Liu and Lapata, 2019; Chu and Liu, 2019), and multi-source machine translation (Nishimura et al., 2018; Garmash and Monz, 2016) that encodes input texts in multiple source languages and translates them into a target language. Cho et al. (2021) generate a question from input documents by applying a multi-encoder model with a transformer-based coordinator.

3 Problem Definition

We introduce the task of document sketching, which aims to facilitate the authoring process by generating a template-like document draft, based on a collection of sampled similar documents. Formally, the task can be defined as follow: given a set of n documents $X = \{x_1, x_2, ..., x_n\}$, generate

---

[3] Our experiments in Section 6 also show that drafts derived from multiple analogous documents are more effective than those derived from one or two documents.

[4] We release our data and source code for dataset construction and experiments at `https://github.com/ellenmellon/document_sketching`.

a text sequence s that can be used as the sketch to reduce the human effort involved in composing a target document y.

Evaluation Metrics As in most text generation tasks, we rely on human evaluation (see details in Section 6.3) to draw final conclusions on system comparison. However, due to the high cost of human evaluation, we use automatic evaluation metrics for system development.

It is difficult (and expensive) to collect human-written sketches as references. However, since a generated sketch s is used to help the user complete writing a target document y (i.e., post-editing), we can instead use the target document as a reference and calculate the extra edits required to transform a sketch into that target document. Inspired by the previous uses of word error rate (WER) in evaluating machine translation (Snover et al., 2006; Tomás et al., 2003) and evidence of reasonable correlation between WER and human post-editing productivity (Tatsumi, 2009; Specia and Farzindar, 2010), we propose to assess the effectiveness of s in the completion of y based on WER and the Levenshtein distance (abbreviated as 'lev') between s and y:

$$\begin{aligned}\mathrm{score}(s, y) &= 1 - \mathrm{WER}(s, y) \\ &= 1 - \frac{\mathrm{lev}(s, y)}{|y|}\end{aligned} \tag{1}$$

The higher $\mathrm{score}(s, y)$ is, the fewer minimum number of word-level insertion and deletion edits a user would need to complete writing y if starting from s. To account for the lexical and phrasal variety of writing a target document, we calculate the average score of multiple reference documents Y:

$$\mathrm{score}(s, Y) = \frac{1}{m}\sum_{i=1}^{m}\left(1 - \frac{\mathrm{lev}(s, y_i)}{|y_i|}\right) \tag{2}$$

where $Y = \{y_1, y_2, ..., y_m\}$ and m denotes the number of reference documents.

4 Data

We collect our dataset from the English Wikipedia dump (June 20, 2020). We first group documents into collections with analogous texts, with the observation that articles with shared reusable structural texts also tend to have similar titles. These collections are then split into training, validation and test sets. This dataset is designed for a weakly supervised setting, as gold sketches are unavailable.

% Percent (Category)	Example Collection Titles
90.0 (reasonable)	__ Academy Awards; Super Bowl __
10.0 (arguable):	
5.8 (partial name)	Bob __ (ice hockey)
2.9 (X of a location)	__ of the United States
1.4 (other)	Origin of the __

Table 1: Quality analysis of over 200 randomly selected document collections.

4.1 Wikipedia Document Collections

We observe that Wikipedia article titles provide a strong indication of whether documents are likely to share structural text (i.e., can be put in the same collection) or not. For example, it is reasonable to consider articles with titles like "Super Bowl I", "Super Bowl II" and so on to comprise a document collection of the annual championship game. Therefore, we group documents whose titles are identical but for one token at a specific position. In the "Super Bowl" example, the document titles are the same up to the third token in each title, and the collection can be named "Super Bowl __". It is worth noting that collecting these articles based on their titles is simply reflective of how similar documents in Wikipedia tend to cluster. Our task setup only requires a set of references as the input (e.g., documents organized in the same directory), without the need to have reference documents to share titles.

Title matching can yield noise in the extracted document collections, so we apply simple yet effective restrictions in the extraction pipeline. We empirically set the minimum number of documents per collection to 15 and the minimum document title length to 3 tokens. We randomly select and inspect over 200 extracted collections, and find 90% of them contain analogous documents for a certain topic (examples in Tab. 1). The remaining 10% are less clear, but can be usefully grouped into 3 classes as in Tab. 1. To further reduce noise in each collection, we apply Eq. (2) to calculate the average similarity score of each document with the other documents in a collection, removing those with an average score lower than an empirically chosen threshold -1.5. Finally, we keep document collections with more than 5 documents and truncate them to a maximum of 50 documents, which are divided into train, validation, and test sets in the ratio of 0.8/0.1/0.1. The collection statistics are summarized in Tab. 2.

	Train	Valid	Test	Overall
# Collections	15.7k	2.0k	2.0k	19.7k
# Docs / Collection	21.1	21.3	21.0	21.1
# Tokens / Doc	80.6	79.3	80.3	80.5

Table 2: Document Collection Statistics.

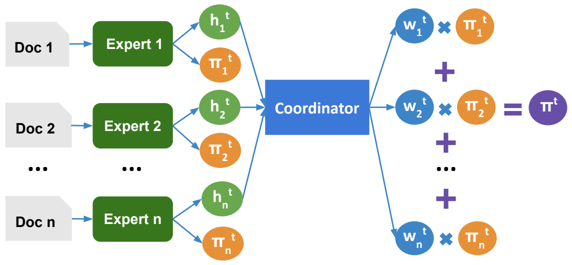

Figure 2: Mixture of Experts (MoE) Framework.

4.2 Task Data Points

We divide each document collection into multiple smaller collections of analogous documents. For standard supervised training, each data point consists of 1 document d to construct a heuristic sketch as weak supervision (see details in Section 5) and up to 9 input documents X. For evaluation, additional 4 documents are used as references Y for calculating the score in Eq. (2). Since document collections whose title tokens differ by a number usually contain documents about events or entities of different times, we order the documents in ascending numerical order to imitate practical scenarios where humans sketch a document by looking at collections of previously written documents. We divide each collection into data points with up to 14 (=1+9+4) documents for all dataset splits, yielding 24k, 3.1k and 3.0k data points (smaller collections) for train, validation, and test sets, respectively.

5 Approach

Since gold document sketches are not available for supervised training, we first perform weakly supervised training by constructing heuristic sketches as targets. Then we apply reinforcement learning strategies for text generation.

5.1 Weakly-Supervised Learning

Heuristic Labels As described in Section 4, each data point has one document d for creating the weak-supervised sketch. We conduct pair-wise sequence alignment (using the Ratcliff-Obershelp algorithm (Ratcliff and Metzener, 1988)) for d and each input document $x_i \in X$, and count the relative alignment ratio (the number of tokens being aligned divided by $|X| = n$) of each token in d. We retain tokens with this ratio above a threshold and replace other tokens with an ellipsis token to form the heuristic target sketch s given X. Consecutive ellipses are merged. The threshold is empirically set at 0.6, which yields the highest average $\text{score}(s, Y)$ on the validation set.

Mixture of Experts (MoE) To generate an initial document sketch, the model needs to take multiple input documents. The most obvious way to do this is to concatenate all input documents into a single long sequence and feed that into an encoder-decoder model like T5 (Raffel et al., 2019). However, processing long sequences in such models is memory-consuming and lack of structure makes it difficult for the model to perform document-level coordination. We therefore use a mixture of experts (MoE) framework, in which a coordinator decodes a token at one timestamp by taking the hidden state and the output vocabulary distribution from each expert that processes a single document. Fig. 2 shows the overall framework of MoE.

The experts have the same encoder-decoder structure and aim to generate s by each encoding a separate single input document only. All experts share the same model parameters. At each decoding timestamp t, the i^{th} expert encodes x_i as well as previously decoded tokens from a coordinator $\tilde{s}_1 ... \tilde{s}_{t-1}$ (or $s_1 ... s_{t-1}$ during training), and outputs a probability distribution π_i^t over all vocabulary words. The coordinator is a transformer-based encoder that takes the hidden state at the current timestamp h_i^t of each i^{th} expert, and outputs a weight w_i^t with a final linear layer. The output weights are used to calculate a weighted sum of the probability distributions from their corresponding experts $\pi_t = \sum_{i=1}^{n} w_i^t \pi_i^t$, where π_t is the final distribution for generation at timestamp t.

5.2 Reinforcement Learning

We leverage reinforcement learning (RL) to further improve generation quality. For each training example with input X, we generate a sequence $\hat{s}$, which is sampled from the probability distribution at each time step, $p(\hat{s}_t | \hat{s}_1 ... \hat{s}_{t-1}, X)$. We observe that directly optimizing the evaluation function proposed in Eq. (2) at the sequence level, using a vanilla policy gradient (PG) or a self-critical sequence training (TD-SCST) algorithm (Rennie et al., 2017;

Paulus et al., 2018; Pasunuru and Bansal, 2018), can lead to instability during training as the reward cannot be calculated until the end of generation (Celikyilmaz et al., 2018). Therefore, we instead use a token-level incremental reward that is based on the change to the original reward function $r(\hat{s}, Y) = \text{score}(\hat{s}, Y)$ from each sampled token $\hat{s}_t$, given references Y:

$$r_t(\hat{s}_t, Y) = r(\hat{s}_{1...t}, Y) - r(\hat{s}_{1...t-1}, Y). \quad (3)$$

The training objective can be written as:

$$L_{\text{RL}} = \sum_{t=1}^{T} -r_t(\hat{s}_t, Y) p(\hat{s}_t | \hat{s}_1 ... \hat{s}_{t-1}, X) \quad (4)$$

where $T = |\hat{s}|$. Since optimizing RL loss alone runs the risk of compromising the language model (Paulus et al., 2018; Pasunuru and Bansal, 2017), we use a mixed loss as follows:

$$L_{\text{MIX}} = \lambda L_{\text{RL}} + (1 - \lambda) L_{\text{MLE}} \quad (5)$$

where λ is a hyperparameter to be tuned.

6 Experiments

6.1 Setup

To leverage the recent success in such transformer-based generation models, neural generation models in our experiments are initialized with the base version of T5 (Raffel et al., 2019; Wolf et al., 2020), an encoder-decoder architecture pre-trained on a variety of text-to-text tasks.[5] All hyperparameters are tuned on the validation set. For MoE, we first fine-tune T5-base to obtain a "single expert" model to initialize each individual component model of the MoE. The "single expert" is trained by leveraging a sequence-to-sequence objective as in T5, with each of the n training (input, output) pairs: (x_1, s), (x_2, s), ..., (x_n, s) from a data instance. This is followed by training a complete MoE model as described in Section 5. We use greedy beam search as the decoding strategy for all generative models with a beam size of 4 (the default value in T5-base). We observe that consecutive ellipses and uninformative tokens hurt readability. To improve the readability of output sketches, we apply minor post-processing to all models in our experiments: we merge consecutive ellipses if all tokens between them are punctuation or among the 30 most frequent tokens.[6] Sentence-terminating periods are excepted.

Training and Parameters T5 has about 220 million parameters and MoE has about 310 million parameters in total. The average training time is about 20 hours for T5 and about 30 hours for MoE on a single Tesla V100 node (32GB) with 3 epochs. It takes an additional 10 hours to train a single expert in MoE. For MoE+RL, we use 4 V100 nodes and it takes about 2 days for training. During training, we tune batch size, learning rate and warm-up steps for T5. For MoE and MoE+RL, there is a separate pair of learning rate and warm-up steps for the coordinator. Each batch size is tuned within a range of [1, 32] (with no more than 3 values for each model); each warm-up step is tuned at $\{4000, 8000, 16000\}$; each learning rate is tuned in $\{1e^{-5}, 3e^{-5}, 1e^{-4}\}$. For RL training, we observe performance gain only when λ is relatively large, so we tune $\lambda \in \{0.95, 0.97, 0.99\}$ (similar to Celikyilmaz et al. (2018)). In order to avoid the situation where the models learn to not generate ellipses that lead to unreadability, we set the cost for deleting an ellipsis to be much smaller (0.1) in the reward function. In addition, we randomly select three seed values for each model. Each model is trained with the above parameter value combinations. We apply grid parameter search for T5 and MoE, and random search for MoE+RL. Hyperparameters are tuned based on automatic evaluation score (1-WER) on the validation set.

The best hyperparameter configurations for best-performing models were: **T5**: batch size = 2, warm-up step = 4000, learning rate = $3e^{-5}$; **MoE**: batch size = 15, warm-up step = 8000, learning rate = $3e^{-5}$, warm-up step (coordinator) = 4000, learning rate (coordinator) = $1e^{-4}$; **MoE+RL**: batch size = 4, warm-up step = 4000, learning rate = $3e^{-5}$, warm-up step (coordinator) = 4000, learning rate (coordinator) = $3e^{-5}$, $\lambda = 0.99$.

6.2 Systems

Non-neural systems include the following approaches: **Last-pair**: The output is the aligned boilerplate text between the last pair of ordered input documents (as described in Section 4, in cases where the differing title token is time-related, the last pair refers to the most recent two docu-

---

[5] Note that T5 can be easily replaced by Other pre-trained generative models like BART (Lewis et al., 2020) in our model framework.

[6] `https://github.com/first20hours/google-10000-english`

ments); **Last-retrieval**: The last document is retrieved as the initial draft; **MSA**: We apply the multi-sequence alignment approach (Barzilay and Lee, 2003), to align tokens in input documents and replace tokens that have too few alignments (threshold tuned on valid set) with ellipses; **Consensus-MSA**: This resembles how we generate heuristic drafts as weak supervision in Section 5.1. However, instead of using the held-out document d to create the skeleton, we use the input document x_i that gives the highest value of $\sum_{j=1}^{n} \mathrm{score}(x_i, x_j)$.

We evaluate the following neural approaches: **T5 (zero-shot)**: We directly apply T5 without finetuning. We use T5 to encode each input document individually and at each decoding timestamp, we combine probability distributions from all T5 decoders with average pooling. As T5 rarely generates ellipses, we insert an ellipsis token if the probability of the top candidate token w from the combined distribution is below a threshold α that is tuned on the validation set (i.e., $p(w) < \alpha$).[7] We used the "summarize :" prefix in the zero-shot setting, this being is the most similar T5 pre-trained tasks to ours; **T5 (doc-finetune)**: We finetune T5 with all input documents concatenated into a single string input (with a special separator token) and each target document (instead of the heuristic sketch) as the generation target; **T5**: Similar to the *T5 (doc-finetune)* setting, but with the heuristic sketch defined in Section 5 as the generation target; **MoE**: This is the mixture of experts model described in Section 5, also trained with the heuristic sketches as the weak supervision; **MoE + RL**: This is the RL model described in Section 5, with the trained MoE as the warm-start.

6.3 Results

Automatic Evaluation We use the automatic evaluation of Section 3 for system development while relying on human evaluation to draw final conclusions on system comparison. However, we observe reasonable consistency between our automatic and human evaluation results. We report automatic evaluation results in Tab. 3. The first numerical column includes overall results of the test set, and the remaining columns categorize the results into three roughly equi-sized levels of similarity among documents within the input document set, estimated on the basis of average pair-wise edit distance normalized by length.

System	All	Input document similarity High	Med.	Low
last-retrieval	−0.242	0.338	−0.284	−0.733
last-pair	0.147	0.410	0.063	−0.021
MSA	0.149	0.374	0.072	0.012
consensus-MSA	0.202	**0.438**	0.132	0.047
T5 (zero-shot)	0.000	0.000	0.000	0.000
T5 (doc-finetune)	0.050	0.382	−0.014	−0.197
T5	0.182	0.389	0.124	0.044
MoE	0.205	0.411	0.147	0.069
MoE+RL	**0.213**	0.414	**0.158**	**0.077**

Table 3: Automatic evaluation scores (1−WER) of overall test examples and scores categorized by input document similarity levels.

System A	System B	Prefer A	Prefer B
MoE	consensus-MSA	**50.70%**	39.28%
MoE	T5	**46.15%**	42.88%
MoE	MoE+RL	**49.93%**	39.65%

Table 4: Human evaluation scores. A number in bold indicates that the system is significantly better at $p < 0.03$, computed using 10k bootstrap replications.

Among non-neural models, we observe that multi-sequence alignment approaches outperform last-retrieval and last-pair where only one or two input documents are leveraged to produce the initial draft. We also notice that consensus-MSA with a "central" document as the skeleton is more effective than the standard MSA that treats all inputs equally. When directly applying zero-shot T5, we notice that when the threshold α described in Section 6.2 equals 1.0 (i.e., the output is always an empty string), the model can achieve the best score 0. This is partly because that none of the T5 pre-trained tasks is the same as ours. In addition, simply exploiting the generation probability at each decoding timestamp can be problematic. For example, when decoding a frequently-appearing entity with multiple tokens, the first token might have low probability given the preceding context while subsequent tokens can be more probable simply because they frequently appear with the first token. T5 with supervision from the target document (doc-finetune) yields a much lower score than T5 finetuned with the heuristic draft. This is mostly because the complete target document contains much more information than can be inferred from the input; the model learns to hallucinate facts, necessitating heavy deletion by users.

MoE outperforms T5, which indicates that hav-

---

[7] Since the original T5 was not trained with ellipsis as a special token, each T5 decoder uses its own greedily decoded token if an ellipsis is inserted ($p(w) < \alpha$).

Collection Title Target Document	__ Waterford Senior Hurling Championship The '1960 WSHC' was the 60th staging of the WSHC since its establishment by the Waterford County Board in 1897. On 9 October 1960, Mount Sion won the championship after a 5-09 to 2-05 defeat of Erin's Own in the final. This was their 16th championship title overall and their eighth title in succession.
consensus-MSA	The '... WSHC' was the ... staging of the WSHC since its establishment by the Waterford County Board in 1897. On ..., Mount Sion won the championship after a ... defeat of ... in the final. This was their ... championship title overall and their ... title in succession.
T5	The '... WSHC' was the 58th staging of the WSHC since its establishment by the Waterford County Board in 1897. On ... September ... won the championship after a ...-10 to 1-... defeat of ... in the final. This was their ... championship title overall and their ... title in succession.
MoE	The '1957 WSHC' was the 57th staging of the WSHC since its establishment by the Waterford County Board in 1897. On 15 September 1957, Mount Sion won the championship after a 2-10 to 1-08 defeat of ... in the final. This was their 13th championship title overall and their fifth title in succession.
MoE+RL	The '... WSHC' was the ... staging of the the WSHC since its establishment by the ... County Board in 1897. On ... 1957, ... won the championship after a ...-10 to ... defeat of ... in the final. This was their ... championship title overall and their ... title in succession.
Collection Title Target Document	Botanischer Garten der Universität __ The 'Botanical Garden in Potsdam' (or Botanischer Garten der Universität Potsdam), is a botanical garden and arboretum maintained by the University of Potsdam. It has a total area of 8.5 hectares, of which 5 hectares are open to the public, and is located immediately southwest of the Orangery Palace at Maulbeerallee 2, Potsdam, in the German state of Brandenburg. It is open daily; an admission fee is charged for the glasshouses only (2017). The garden was established in 1950 on two adjacent plots of land: part of the Sanssouci Park, and the Paradise Garden (about 2.5 hectares).
consensus-MSA	... a botanical garden ... maintained by the University of ... It is located ... open ... The garden was ... contains ...
T5	The 'Botanischer Garten der Universität ... is a botanical garden maintained by the University of ... It is located at ... Baden - Württemberg, Germany, and open ... The garden was established in ... and contains about ... species ...
MoE & MoE+RL	The 'Botanischer Garten der Universität ... is a botanical garden maintained by the University of ... It is located at ... The garden was established in ...

Table 5: Sample outputs: in the first set of outputs there is high similarity among documents in the input document set while in the second inter-document similarity is low. Generated tokens are in blue if they are consistent with the target document, and in red if inconsistent.

ing a coordinator to communicate between each individual encoder and decoder effectively improves model performance. We also experiment with coordinators using a simple linear layer and find that transformer-based ones are much more effective. Applying RL on top of a trained MoE model helps further improve the model performance in automatic evaluation. We denote the RL approach as MoE+RL in Tab. 3. Warm-starting RL with a MoE model close to fully-converged gives slightly better results than with an fully-converged MoE model.

The rightmost three columns in Tab. 3 divide test examples into 3 roughly equal-sized levels of similarity among documents within the input document set ('high', 'medium', 'low'). Both last-retrieval and T5 (doc-finetune) models drop dramatically as the input similarity decreases, mostly because they contain a lot of hallucinated content, an issue that is even more severe when there is little overlapping structure or factual content among the input documents. MoE based models are much more robust to low input similarity compared to the baselines. For the group with the most similar input documents, consensus-MSA gives the best score, while MoE based models yield much better performance in the other two groups.

Human Evaluation In order to avoid bias in human judgments, we control possible confounding factors including sequence length and the number of ellipses in a sequence during decoding (Nakov et al., 2012; Guzmán et al., 2015). We apply a tuneable penalty for each confounding factor (at inference time only) for each neural model (Murray and Chiang, 2018) to generate examples for human evaluation, such that the average output length and number of ellipses in each sequence from all neural systems compared in Tab. 4 are almost the same.[8] We notice that such normalization does not affect the automatic evaluation score of each system much and the system ranks do not change.

Human evaluation was conducted using crowd-sourced workers. Judges were presented with paired randomized outputs and target documents, and were instructed to choose their preference for a starting point for writing the target document in order to save editing time.[9] Judgments were based on a five-point Likert scale, and ties were permitted. Four judges evaluated each pair, and metrics were imposed to block poorly performing judges. Sample sizes are 1000 for all system pair comparisons.

Results in Tab. 4 confirm that MoE outperforms consensus-MSA and T5. Although RL helps improve automatic evaluation scores on top of MoE, human judges mostly prefer MoE over MoE+RL. The fact that applying RL can hurt readability (Paulus et al., 2018; Pasunuru and Bansal, 2018) may explain why MoE+RL achieves higher automatic scores yet worse human scores than MoE. We observe that MoE+RL has occasional difficulty predicting where ellipsis tokens should be, which hurts sketch readability. Moreover, since our task is a subjective one (e.g., different preferences for more/less verbose sketches), a customized RL reward (e.g., different cost for deletion and insertion for WER calculation) should be applied in real applications to reflect different user preferences.

6.4 Analysis

We investigated the differences in model performance when the input documents have different similarity scores. When the similarity is lower, models tend to produce drafts that are harder to interpret because they generate a higher percentage of generic or ellipsis tokens, although MoE and MoE+RL are less vulnerable to this. Tab. 7 shows the average percentage of generated ellipsis or punctuation tokens for each system when the input document similarity is high or low. We can see that when the similarity score is higher, the two statistical numbers do not differ greatly between systems, except that MSA has a much higher percentage of ellipses. When similarity is low, MoE and MoE+RL have much lower percentages of ellipses or punctuation, compared with other models.

Tab. 5 shows two examples of system outputs. These examples are from the clusters of high and low input similarity respectively. When the input similarity is high, content tokens from consensus-MSA are included in the target document while the neural models tend to generate more hallucinated tokens. This explains the better score consensus-MSA achieves in the higher input similarity cluster. We also notice that when the similarity is high, consensus-MSA generally has better coverage of overlapped content between input documents, and its generated drafts are on average more than 10% longer than other systems. When the similarity is low, the second example shows that all systems generate shorter outputs due to less shared structure between input documents. In this case, consensus-MSA starts to include more content that does not necessarily appear in the target and MoE-based models generate more reasonable sketches and contain fewer ellipses and uninformative tokens, which can hurt the sketch readability.

7 Discussion

Since gold document sketches are difficult to annotate, there remain challenges of how to leverage or create better weak supervision. We may also want models to adjust to users' personalized preferences over document sketches (e.g., some may prefer more deletions than insertions or vice versa) in practical application deployment, where RL can play a major role by having models directly optimized by customized rewards.

We show that MoE-based models tend to generate more readable sketches by outputting fewer ellipses and functional tokens, though in some cases it can be difficult to guess what should fill a given ellipsis. An interesting future study could examine how the placement and number of ellipses in a sketch affects readability. We would also like to explore whether replacing ellipsis tokens with some more contentful label (e.g., entity type) or a retrieved sample content for a gap (Xu et al., 2021) could help make sketches more interpretable.

An initial document sketch provides the reusable text structure that allows a user to fill in detailed content. Interactive document generation can be seen as the next step of our document sketching task that dynamically fills in more local contexts based on users' inputs. We suggest that RL could again

---

[8] Since MoE has a more balanced average sequence length and the number of ellipses, we keep MoE unchanged and adjust T5 and MoE+RL to have the same values as MoE.

[9] Ideally, it would be preferable to directly measure editing productivity through usability testing, but this is impractical in a crowd worker setting, and dependent on confounding factors such as design of the user interface.

Collection Title Target Doc Title	1960 United States Presidential Election in __ 1960 United States Presidential Election in Colorado
w/o user input	**The '1960 United States presidential election in ...' took place on November 8, 1960, as part of the 1960 ... election.** ... chose ... representatives, or electors to the Electoral College, who voted for president and vice president.
w/ user input	**The '1960 United States presidential election in Colorado' took place on November 8, 1960, as part of the 1960 United States presidential election.** Voters chose three representatives, or electors to the Electoral College, who voted for president and vice president. Colorado was won by incumbent Vice President Richard Nixon (R-California), with 50.9% of the popular vote, against Senator John F. Kennedy (D-Massachusetts) with 49.1%.

Table 6: Example of interactive writing with user input. The first sentence (**bold**) is generated in the "w/o user input" setting, while it is the gold first sentence of the target document in "w/ user input" setting. Generated tokens are in blue if consistent with the target document and red if inconsistent.

	Ellipses		Punctuation	
System	High	Low	High	Low
MSA	15.9%	42.0%	10.9%	19.0%
consensus-MSA	10.6%	29.8%	10.8%	19.2%
T5	10.0%	23.2%	10.9%	14.6%
MoE	8.6%	16.6%	10.2%	13.0%
MoE+RL	7.3%	15.4%	10.2%	14.0%

Table 7: Percentage of ellipses (in all generated tokens) and punctuation (in generated tokens excluding ellipses) tokens. 'High' and 'low' refer to the average pair-wise input document similarity level.

play a major role towards building such writing assistants, with rewards from a real user or a user simulator. We notice, for example, in Tab. 6 that given user input in the first sentence, the subsequent generated sentences become much more specific and relevant to the new input.

8 Conclusions

We have presented a new task, DOCUMENT SKETCHING, designed to generate draft documents that users can edit. To support this task, we introduced a new dataset and a weakly supervised learning setting. Experimental results show that deep learning models outperform established multi-sequence alignment approaches.

Acknowledgments

We thank members of Microsoft Research and University of Washington's NLP groups who provided feedback and insights to this work. In particular, we thank Chenyan Xiong for providing insights during early project discussions. We also thank Michael Lee, Sara Ng, Sitong Zhou, and Trang Tran for their assistance and feedback on the human evaluation setup. We also thank Zhang Li, Kosh Narayanan, Chandra Chikkareddy, Si-Qing Chen, and Weixin Cai of Microsoft's Natural Language Experience team for their insights into authoring experience.

Ethical Considerations

We believe this paper to be part of body of work that can mitigate some of the ethical pitfalls (Bender et al., 2021) of large pre-trained models such as GPT-3 (Raffel et al., 2019). While the present work does not entirely prevent misuse, e.g., by a malicious user trying to promote misinformation, Automatic Document Sketching can be more broadly framed as a form of *controllable* generation, which places greater control into the hands of the users. As such it is complementary to ongoing research on fake news detection (Zellers et al., 2019), toxicity detection (Han and Tsvetkov, 2020; Pavlopoulos et al., 2020), and consistency detection in, e.g., summarization (Maynez et al., 2020; Wang et al., 2020). Human subjects (crowdworkers) were paid at a rate higher the legal minimum wage in our locale (Washington State, U.S.A.).

References

Dimitris Alikaniotis and Vipul Raheja. 2019. The unreasonable effectiveness of transformer language models in grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 127–133, Florence, Italy. Association for Computational Linguistics.

Laura Alonso, Irene Castellón, Jordi Escribano, Xavier Messeguer, and Lluís Padró. 2004. Multiple sequence alignment for characterizing the lineal structure of revision. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 16–23.

Emily Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proc. of ACM FAccT*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and Prafulla Dhariwal et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1866–1875, Hong Kong, China. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. 2019. Gmail smart compose: Real-time assisted writing. In *KDD*.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2019a. Towards coherent and cohesive long-form text generation. In *Proceedings of the First Workshop on Narrative Understanding*, pages 1–11, Minneapolis, Minnesota. Association for Computational Linguistics.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2019b. Towards coherent and cohesive long-form text generation. In *Proceedings of the First Workshop on Narrative Understanding*, pages 1–11, Minneapolis, Minnesota. Association for Computational Linguistics.

Woon Sang Cho, Yizhe Zhang, Sudha Rao, Asli Celikyilmaz, Chenyan Xiong, Jianfeng Gao, Mengdi Wang, and Bill Dolan. 2021. Contrastive multi-document question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 12–30, Online. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232. PMLR.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4508–4513, Online. Association for Computational Linguistics.

Felix Faltings, Michel Galley, Gerold Hintz, Chris Brockett, Chris Quirk, Jianfeng Gao, and Bill Dolan. 2021. Text editing by command. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5259–5274, Online. Association for Computational Linguistics.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2020. Recursive template-based frame generation for task oriented dialog. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2059–2064, Online. Association for Computational Linguistics.

Shen Gao, Xiuying Chen, Piji Li, Zhangming Chan, Dongyan Zhao, and Rui Yan. 2019. How to write summaries with patterns? learning towards abstractive summarization through prototype editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3741–3751, Hong Kong, China. Association for Computational Linguistics.

Ekaterina Garmash and Christof Monz. 2016. Ensemble learning for multi-source neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418, Osaka, Japan. The COLING 2016 Organizing Committee.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Francisco Guzmán, Preslav Nakov, and Stephan Vogel. 2015. Analyzing optimization for statistical machine translation: MERT learns verbosity, PRO learns length. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 62–72, Beijing, China. Association for Computational Linguistics.

Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7732–7739, Online. Association for Computational Linguistics.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *NeurIPS*, pages 10073–10083.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated response suggestion for email. In *KDD*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Ahmed Magooda and Diane Litman. 2019. Abstractive summarization for low resource data using domain transfer and data synthesis. In *AAAI*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. In *Proceedings of COLING 2012*, pages 1979–1994, Mumbai, India. The COLING 2012 Organizing Committee.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018. Multi-source neural machine translation with missing data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 92–99, Melbourne, Australia. Association for Computational Linguistics.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 45–53, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2017. Reinforced video captioning with entailment rewards. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 979–985, Copenhagen, Denmark. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565, Minneapolis, Minnesota. Association for Computational Linguistics.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2622–2632, Minneapolis, Minnesota. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv*, abs/1910.10683.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

John W. Ratcliff and David Metzener. 1988. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

J. Michael Sauder, Jonathan W. Arthur, and Roland L. Dunbrack Jr. 2000. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins: Structure, Function and Genetics*, 40(1):6–22.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Lucia Specia and Atefeh Farzindar. 2010. Estimating machine translation post-editing effort with hter. In *Proceedings of the Second Joint EM+/CNGL Workshop Bringing MT to the User: Research on Integrating MT in the Translation Industry (JEC 10)*, pages 33–41.

Midori Tatsumi. 2009. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *MT Summit XII: proceedings of the twelfth Machine Translation Summit*.

Jesús Tomás, Josep Àngel Mas, and Francisco Casacuberta. 2003. A quantitative method for machine translation evaluation. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, pages 27–34, Columbus, Ohio. Association for Computational Linguistics.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Kai Wang, Xiaojun Quan, and Rui Wang. 2019. BiSET: Bi-directional selective encoding with template for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2153–2162, Florence, Italy. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, and Anthony et al. Moi. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Frank F Xu, Bogdan Vasilescu, and Graham Neubig. 2021. In-IDE code generation from natural language: Promise and challenges. *arXiv preprint arXiv:2101.11149*.

Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. 2020. Improving neural machine translation with soft template prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5979–5989, Online. Association for Computational Linguistics.

Ze Yang, Wei Wu, Jian Yang, Can Xu, and Zhoujun Li. 2019. Low-resource response generation with template prior. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1886–1897, Hong Kong, China. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Trade the Event: Corporate Events Detection for News-Based Event-Driven Trading

Zhihan Zhou **Liqian Ma** **Han Liu**
Department of Computer Science, Northwestern University
`{zhihanzhou, liqianma}@u.northwestern.edu`
`hanliu@northwestern.edu`

Abstract

In this paper, we introduce an event-driven trading strategy that predicts stock movements by detecting corporate events from news articles. Unlike existing models that utilize textual features (e.g., bag-of-words) and sentiments to directly make stock predictions, we consider corporate events as the driving force behind stock movements and aim to profit from the temporary stock mispricing that may occur when corporate events take place. The core of the proposed strategy is a bi-level event detection model. The low-level event detector identifies events' existences from each token, while the high-level event detector incorporates the entire article's representation and the low-level detected results to discover events at the article-level. We also develop an elaborately-annotated dataset ***EDT*** for corporate event detection and news-based stock prediction benchmark. EDT includes 9721 news articles with token-level event labels as well as 303893 news articles with minute-level timestamps and comprehensive stock price labels. Experiments on EDT indicate that the proposed strategy outperforms all the baselines in winning rate, excess returns over the market, and the average return on each transaction. [1]

1 Introduction

By shaping investors' perceptions and assessments of companies, financial news has significant impacts on the stock market (Engle and Ng, 1993; Tetlock, 2007). News-based stock prediction models are thus developed to automatically discover signals of stock market movements from the countless news articles that generated every moment. (Kalyani et al., 2016; Shah et al., 2018; Mohan et al., 2019).

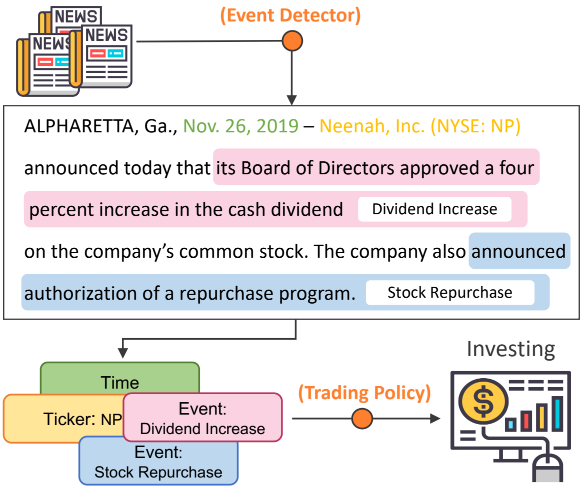

Figure 1: Overview of the event-driven trading strategy

Previous studies mainly rely on textual features and sentiment analysis to forecast the stock movements (Fung et al., 2003; Liu, 2018; Huynh et al., 2017). Both of them, however, often face the problem of poor explainability and low signal-to-noise ratio. Textual feature-based methods often formulate the stock prediction as a text classification problem by directly predicting the rise and fall of stocks based on the extracted features. These models fail to make reasonable trading decisions since they omit the reasons behind stock price changes. Sentiment-based methods avoid this problem by regarding the news articles' sentiments as the indicator of stock movement. However, news sentiment is subjective, which can be greatly affected by the author's standpoint and writing style.

Unlike textual features and sentiments, corporate events are objective facts that impact how investors perceive and assess the related companies. Thus, we resort to corporate events to make more convincing and explainable stock predictions. Jacobs et al. (2018) achieves corporate events detection by splitting a news article into sentences and detecting

---

[1] Code and the EDT dataset is available at `https://github.com/Zhihan1996/TradeTheEvent`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2114–2124
August 1–6, 2021. ©2021 Association for Computational Linguistics

events on each of them with multi-label sentence classification. This method, however, discards the global contextual information of the entire article and fails to indicate the evidence of events' existences. We believe that detecting events at a smaller granularity (e.g., at the token-level) is beneficial on both model training and application sides. During training, explicitly assigning a label to each token gives the model specific guidelines of what to identify. On the application side, each detected event is supported by one or more subsequences of the original article, allowing users to easily distinguish the predicted results.

However, singly detecting events from the token-level may result in a lack of macro understandings of the entire article. To tackle this, we introduce a bi-level event detection model, in which a low-level detector identifies the subsequences that describe specific events by classifying each token. And a high-level detector takes the predicted results from low-level and integrates them with the input article's global contextual information to predict the probabilities of each event's existence.

Another problem with existing models is that they ignore the timeliness of news articles. Most of them utilize news articles to predict the related securities' rise/fall on the following trading day(s). However, stock prices are very likely to change immediately in response to noteworthy news. Thus, the stock movement in the following trading day(s) may not accurately reflect a news article's influence. To tackle this, we make stock predictions as soon as a news article is published and perform tradings at that moment with the proposed trading policies.

Based on the event detection model and trading policies, we construct an event-driven trading strategy that simultaneously detects corporate events from news articles, indicates the subsequences that describe the detected events, and performs trading on the related stocks. By running the strategies against massive historical data in EDT, we demonstrate the superiority of the proposed strategy in terms of excess returns over the market and the average return on each transaction. The experiment results also reveal the timeliness of news and the effectiveness of corporate events in indicating stock movement.

The main contributions of our paper are as summarized follows: (i) We introduce a novel event-driven trading strategy that detects trading signals from arbitrary unlabeled news articles; (ii) We present ***EDT***, an elaborately-annotated dataset with 300000+ news articles for corporate event detection and news-based trading benchmark. (iii) We propose a bi-level event detection model that integrates macro and fine-grained understandings to effectively identify corporate events;

2 Problem Definition

We aim to construct an event-driven trading strategy that automatically detects corporate events from news articles and performs trading accordingly. The proposed strategy consists of two components: *bi-level event detector* and *trading policy*.

The low-level event detector identifies events from each token. We formulate the low-level event detection as a sequence labeling problem. The label set $\mathcal{L} = \{e_1, e_2, ..., e_k, O\}$ consists of k pre-defined events and a special label O that stands for ***Noevent*** [2]. We define an article $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ as a sequence of tokens and define its label sequence as $\boldsymbol{y} = (y_1, y_2, ..., y_n)$. The same event may be mentioned multiple times in an article, and a single article may contains multiple events. If a subsequence $\boldsymbol{x'} = (x_t, x_{t+1}, ..., x_{t+s})$ of $\boldsymbol{x}$ describes the ***event i***, $\{y_j\}_{j=t}^{t+s}$ are labeled as e_i. All the other words are labeled as O. The low-level event detection is defined as follows: given an article $\boldsymbol{x}^* = (x_1, x_2, ..., x_n)$, find its best label sequence $\boldsymbol{y}^* = (y_1, y_2, ..., y_n)$. We say ***event i*** is detected by the low-level detector if $e_i \in \boldsymbol{y}^*$. Based on the low-level detected results, the high-level event detector calculates the probability of each event's existence. We say ***event i*** is detected by the high-level detector if its existence probability is larger than a given threshold. We combine the predictions on both levels as the final prediction. When events are detected, the *trading policy* decides when to buy and sell the related securities.

3 Strategy

This section discusses the proposed strategy by respectively introducing the event detector and the trading policies.

3.1 Event Detector

Before training the model to detect events, we first equip it with prerequisite knowledge of the financial domain by performing a *domain adaptation*.

---

[2] events that are not included in $\{e_i\}_{i=1}^{k}$ are also considered as *Noevent*

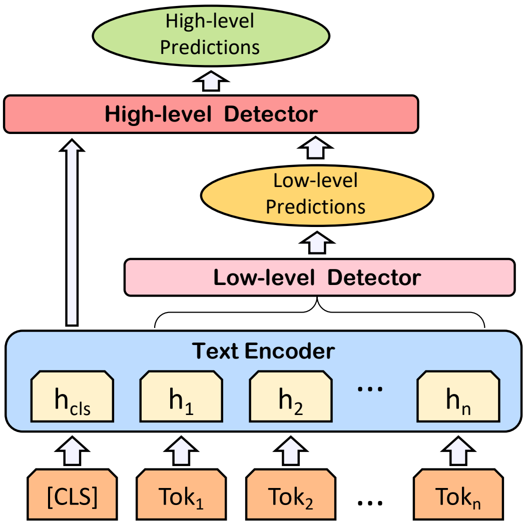

Figure 2: Model overview. The low-level detector identifies corporate events from each token, while the high-level detector summarizes the low-level predictions and the input representation to detect events at the article-level

3.1.1 Domain Adaptation

Since the same event can be expressed with significantly different terms and descriptions, and the same terms can refer to different meanings, understanding the event itself and its related terms is of great importance. We perform domain adaptation by training the model with Masked-Language-Model (MLM) loss on a financial encyclopedia as well as financial news articles. Section 4.3 discusses this corpus in details. During training, 15% tokens of an input sequence are masked, and the model is asked to predict the masked tokens. The prediction is essentially a multi-class classification over the entire vocabulary. We optimize the model with the *Categorical CrossEntropy* loss calculated on all the masked tokens.

3.1.2 Bi-Level Event Detection

As shown in fig. 2, the event detection model takes an article as input and respectively detects events from two levels. Each article is concatenated with a special token [CLS]. The Transformer-based text encoder calculates a series of hidden states for each token. We consider [CLS]'s last hidden state h_{cls} as the representation of the entire article, and h_i as the representations of *token i*.

The low-level detector identifies the subsequences that describe corporate events. For a given label set $\mathcal{L}$ (section 2) with $K+1$ labels (e.g., K pre-defined events and a "*Noevent*"), the detector assigns $K+1$ scores to each token by performing a multi-classes classification based on the token's representation. Each score stands for the confidence of finding a specific event at this position. These scores are concatenated and passed to the high-level detector. During training, we calculate the *Categorical CrossEntropy* loss for each token of an article and average them as the article's low-level loss.

The high-level detector concatenates the low-level prediction as well as the entire article's representation to calculate the probability of each event's existence. We formulate it as a multi-label classification problem. Specifically, we assign K binary labels to each article and represent them with a K-length label vector. For articles without any events, their label vectors are all-zero. If *event i* occurs in an article, the i-th component of its label vector is set to 1. We utilize the *Binary CrossEntropy with Sigmoid* as the loss function. This function considers the K-label classification as K independent binary classification problems. Specifically, it uses the $Sigmoid$ function to map each vector component of the high-level detector's output to $(0, 1)$. We consider the mapped score of each event as its probability of existing in the input article. We then calculate the *Binary CrossEntropy* loss between the mapped score and the binary label of each event. The losses of all the events are summed as the high-level detector's loss. We sum the losses of the low-level and high-level detectors to simultaneously optimize the detectors and the text encoder.

3.1.3 Ticker Recognizer

To make the proposed strategy applicable to arbitrary news articles, besides detecting events, we also recognize the related securities (e.g., company) to trade on.

Each security listed on an exchange has a unique ticker, which is a unique arrangement of characters (e.g., Amazon's ticker at the NASDAQ exchange is AMZN). To recognize the tickers, we download company-ticker pairs (e.g., Amazon v.s. AMZN) for all the securities listed on NYSE and NASDAQ from Yahoo[3]. For a given article, we perform string matching between the article and all the company-ticker pairs. If multiple company-ticker pairs are matched, we choose the one that occurs the most times. Some tricks are employed to improve the accuracy and efficiency. For example, the company-

---

[3] `finance.yahoo.com`

ticker pairs that match the title's first few words are assigned higher confidence. Although a single article may include multiple securities, we only recognize the one that occurs most to simplify the setting.

3.2 Trading Policy

To minimize other factors' influences, in this paper, we trade only on stocks instead of any derivatives (e.g., options). We relate each detected event to a *long* or *short* trading signal singly based on its event type. Events that may result in a stock price rise are considered as *long* signals, while events that may lead to a fall of stock prices are considered as *short* signals.

We implement two trading policies named **Trade-At-End** and **Trade-At-Best**. Both of them long (e.g., buy) the related stocks for *long* signals and perform short-selling for *short* signals. We define a transaction as a buy(sell) and a sell(buy) of the same stock. The policies always start a transaction (e.g., perform a buy or a short-selling) at the first available time when an event is detected.

Trade-At-End (k): This policy holds a started transaction for k trading days and closes the transaction on the last day when the market closes.

Trade-At-Best (k): This policy closes a transaction at the best price (e.g., highest for sell and lowest for buy) within k trading days from the start date. It estimates the profit that a trader can gain at most within k trading days with a detected event.

4 Data

In this section, we discuss the *EDT* dataset. EDT contains data for three purposes: 1. corporate event detection (section 4.1); 2. news-based trading strategy benchmark (section 4.2); 3. financial domain adaptation (section 4.3).

4.1 Data for Event Detection

We choose 11 types of corporate events that have relatively predictable and straightforward impacts on the stock price based on financial knowledge.

4.1.1 Event Type

In this work, we only focus on non-periodic corporate events. Trading on periodic corporate events such as *Earning Call* is much trickier since investors can access their information from multiple sources in advance. We leave them for future works.

Guidance Increase (GI) Guidance is a company's public estimates of its upcoming-quarter/fiscal year earnings. This event includes the announcements of guidance increase or upgrade.

Acquisition (A) An acquisition event happens when a company announces to purchase all or a portion of another company's shares or assets.

New Contract (NC) The new contract event refers to a company announcement of being awarded a new contract.

Stock Split (SS) A stock split event refers to a company that divides the existing shares of its stock into multiple new shares.

Reverse Stock Split (RSS) This is the reverse process of the stock split, which consolidates the number of existing stock shares into fewer shares.

Positive Clinical Trial & FDA Approval (CT) This event includes (i) positive trial results from clinical studies; (ii) receiving FDA approval, clearance, or being granted by FDA to market legally in the United States.

Stock Repurchase (SR) A company's stock repurchase events include declaring, reinstatement, or increasing a stock buyback plan.

Dividend (RD) The dividend is a distribution of some of a company's profits paid to its shareholders.

Dividend Cut (DC) A dividend cut means to reduce, stop, or suspend a pre-announced dividend.

Dividend Increase (DI) A dividend increase refers to an increase in the regular dividend.

Special Dividend (SD) A special dividend is an event that a company declares a non-recurring dividend paid to its shareholders.

4.1.2 Detailed information

Event	Num.	Event	Num.
A	229	CT	314
RD	290	DC	109
DI	225	GI	99
NC	518	RSS	51
SD	80	SR	385
SS	76		

Table 1: Number of articles in *EDT* for each event.

We collect 9721 English news articles, of which 2266 articles contain at least one of the above events. Table 1 shows the number of articles that corresponds to each event. The rest 7455 articles are news articles that do not contain any of the above events. We expect them to help the model better distinguish the event-related articles from the non-event ones. Among them, we deliberately include hundreds of non-event articles that are highly similar to event-related ones. An example here is "Apple announces a stock repurchase program" v.s. "Apple announces the completion of the recently announced stock repurchase program", in which we do not expect the latter one to have a significant influence on the stock price.

These news articles are downloaded from PRNewswire[4], Businesswire[5] and GlobeNewswire[6] using keywords-search. The keywords for each event are manually determined based on samples of that event. Each article's title, subtitle, and main text are concatenated after data cleaning (e.g., remove special symbols). We annotate each article with token-level labels. Two human annotators are asked to independently mark the subsequences that best describe the pre-defined corporate events. The annotations of an article are produced if the annotators give the same result. Otherwise, they discuss the best annotations.

We randomly sample 80% articles of each event and combine them with 80% of non-event articles to form the training data. The rest are considered as the validation data.

4.2 Data for Strategy Evaluation

We develop this data to benchmark news-based stock prediction models and trading strategies. To accurately account for the stock movement, the news articles should be original-sourced. To mimic the real-world situation, the news articles should be diverse enough (e.g., in different categories). Thus, we choose PRNewswire and Businesswire as the article collection sources and download all the English news from PRNewswire (Mar 1st, 2020 - Apr 30th, 2021) and Businesswire (Aug 16th, 2020 - May 6th, 2021). We remove the articles that exist in the training data (section 4.1.2). Since some pre-defined events are infrequent (e.g., stock split), to ensure that there are at least a few samples of every event, we add all the articles of the validation data (section 4.1.2) to this data. After data cleaning, there are 303893 news articles.

Each article comes with a minute-level timestamp, which allows researchers to locate the exact event happening time. Generally, news-based trading strategy evaluation involves four steps. (i) Identify trading signals (e.g., corporate events or sentiments) from news articles; (ii) For each article where trading signals are detected, recognize the related company(ticker); (iii) Get the recognized company's stock price data around the publish time of the news; (iv) Perform transactions based on trading policies.

To enable researchers without ticker recognizers and historical stock price data to benchmark their models/strategies, we assign each article with an automatically recognized ticker as well as detailed price labels of that ticker. With the detailed price labels of each news article, strategy evaluation can be as easy as "counting the price changes on the articles that are recognized as trading signals".

Specifically, an article's price labels includes: *open* / *close* prices at the first minute we can trade on, *highest* / *lowest* prices in the following 1/2/3 trading days, *close* prices in the following 1/2/3 trading days, and the minute-level timestamp corresponding to each price. If available, we take the prices in the pre-market and after-hours into consideration since many corporate events are announced in these periods, and stock prices may change greatly during these times. The price labels are empty for articles where no ticker is recognized or the historical price data is unavailable. Among all the evaluation data, 106619 articles come with non-empty price labels.[7]

4.3 Data for Domain Adaptation

The corpus for domain adaptation contains financial news articles and a financial terms encyclopedia, which is considered as unstructured domain knowledge. For encyclopedia, we download 6260 explanatory documents from Investopedia[8]. Each document explains a specific financial term and describes the role it may play in the financial market. For news articles, we directly utilize all the news articles of the training data (section 4.1.2).

5 Experiment

In this section, we first exhaustively compare Bi-level Detection with baselines under different settings. Then, we discuss how each event contributes to the overall trading results. Lastly, we analyze the profitability and practicality of the proposed strategy in real-world stock trading.

---

[4] `https://www.prnewswire.com/`

[5] `https://www.businesswire.com/`

[6] `https://www.globenewswire.com/`

[7] We acquire historical stock price data from `https://polygon.io/`

[8] `https://www.investopedia.com/`

Among the 11 corporate events in the EDT dataset, we do not trade on the **Dividend** since we do not consider it to have a significant influence on stock prices. Among the rest, we consider **Reverse Stock Split** and **Dividend Cut** to have negative influences on the stock price, while the others to have positive influences. We evaluate the performance of the proposed strategy with backtesting. Backtesting is widely used to evaluate a trading strategy's effectiveness by running the strategy against historical data. We perform trade on all the detected trading signals for each model.

Metrics For a "buy" transaction, we define its return as $\frac{P_{sell}-P_{buy}}{P_{buy}}\%$, while for a "short-selling", we define its return as $\frac{P_{sell}-P_{buy}}{P_{sell}}\%$. Here, P stands for the price. If a transaction's return is greater than or equal to 0, we call it a "win". If a transaction's return is greater than or equal to 1%, we call it a "big win". For each model, we calculate its **winning rate**, **big win rate** (rate of big wins among all the transactions) and **average return on each transaction**. We also evaluate the models' excess returns over the market, where we consider the *S&P 500* index as the benchmark of the market performance. The **market return** is estimated as the return of buying *S&P 500 index ETF* for \$10000 on Mar 1st, 2020 and sell all of them on May 6th, 2021[9]. For each model, we start with \$10000 cash and invest \$2000 to each trading signal. When available cash is less than \$2000, we invest 20% of available cash to the detected signal. We report the **excess returns** of each model, which equals to a model's **total returns** minus the **market return**. [10] We assume there is a 0.3% commission fee on each transaction.

Model Hyperparameters We employ the pretrained BERT (Devlin et al., 2018) model as the text encoder. Specifically, we use the *bert-base-cased* checkpoint. Both the low-level and high-level detectors consist of a hidden layer and an output layer. There are 2048 hidden units in the hidden layer. We utilize AdamW optimizer with batch size 32 and learning rate 5e-5 to train the encoder and detectors together for 5 epochs. We set the maximum input length as 256 since we find almost all the events mentioned in a news article exist in its first 256 tokens. Training of the model costs 15 minutes on 4 Nvidia RTX 2080Ti GPUs. We conduct each experiment with 3 different random seeds and report the average results.

Trading Details For Trade-At-End, we execute a stop loss of 20% (e.g., sell a stock immediately when it falls 20%). In all the experiments, we only trade on the news articles where the historical price data of the detected ticker is available at the minute when the article is published. In other words, we ignore all the news articles that are not published during the market hours and articles where the historical price data is incomplete.

5.1 Baseline

Vader (Gilbert, 2014) is a rule-based sentiment analysis model that assigns *positive*, *negative*, and *neutral* scores to an article. We consider news articles with a positive score greater than 0.2 as *long* trading signals.

BERT-SST is a BERT-based (Devlin et al., 2018) sentiment analysis model trained on the Stanford sentiment treebank (SST) dataset. We respectively consider news articles with a positive score greater than 0.995 and 0.9 as *long* trading signals to reduce the threshold's influences on the final results.

Sentence (Jacobs et al., 2018) splits an article into sentences and performs sentence-level event detection based on multi-label text classification. It was original implemented with SVM and LSTM. We re-implement it with BERT to compare it fairly with our models. We split each article into sentences with the NLTK toolkit (Loper and Bird, 2002) to train and evaluate the model.

BERT-CRF (Lafferty et al., 2001) was originally proposed as a Conditional Random Fields-based sequence labeling model, which combines emission scores given by BERT and learned transition scores to find the global optimal label sequence for each input. We re-implement it to perform event detection singly on the token-level. Following the literature, we use different learning rates for the CRF(1e-3) and the BERT(3e-5) components.

5.2 Main Results

Table 2 and 3 respectively shows the models' 1-day and 2-day trading results. The result of 3-day trading is consistent. Due to space limitations, we present it in appendix A. As shown in the tables,

---

[9] In accordance to the time span of data for evaluation

[10] Since Trade-At-Best always finishes the transaction at the best price, its winning rate is always 100% and its total returns is almost linearly related to the number of transactions. Thus, we only report the average return of this policy.

Model	TAE (1)			TAB (1)	
	Win Rate	**Ave. Return**	**Exc. Returns**	**Ave. Return**	**Num. Trans.**
Vader (Gilbert, 2014)	52.8‖24.3%	0.06%	-$8116	1.72%	4327
BERT-SST (0.995)	54.3‖26.9%	0.45%	$3743	2.94%	2378
BERT-SST (0.9)	52.9‖26.1%	0.41%	$44049	3.31%	15663
Sentence (Jacobs et al., 2018)	**55.5**‖30.9%	1.37%	$54064	7.21%	2881
BERT-CRF	53.7‖33.8%	1.60%	$83120	8.87%	3533
Bi-level Detection (ours)	54.5‖**34.2**%	**1.74**%	$**84443**	**9.11**%	3118

Table 2: This table shows the 1-day trading result, in which we start each transaction at the news article's publish time and end the transaction after 1 trading day. **Win Rate** stands for the overall winning rate (rate of transactions that have a return over 0) ‖ big win rate (rate of transactions that have a return over 1%). **Ave. Return** stands for the average return on each transaction. **Exc. Return** stands for the total excess returns over the market when starting with $10000 and invest $2000 to each detected trading signal. **Num. Trans.** stands for the number of transactions (valid trading signals) of each model.

Model	TAE (2)			TAB (2)	
	Win Rate	**Ave. Return**	**Exc. Returns**	**Ave. Return**	**Num. Trans.**
Vader (Gilbert, 2014)	53.7‖36.9%	0.38%	-$2551	3.11%	4327
BERT-SST (0.995)	**53.8**‖38.4%	0.62%	$8479	4.72%	2378
BERT-SST (0.9)	52.3‖37.2%	0.46%	$12802	3.93%	15663
Sentence (Jacobs et al., 2018)	52.7‖39.9%	1.24%	$24673	9.49%	2881
BERT-CRF	51.3‖39.9%	1.39%	$52891	11.27%	3533
Bi-level Detection (ours)	52.3‖**40.8**%	**1.56**%	$**59375**	**11.53**%	3118

Table 3: This table shows the 2-day trading result, in which we start each transaction at the news article's publish time and end the transaction after 2 trading day.

our model outperforms all the baselines on average return and exceed return under all the settings.

Results of Ticker Recognizer To evaluate the performance of ticker recognizer, we manually label tickers for 1674 news articles. The proposed ticker recognizer succeeds in 1643 of them (accuracy: 98.15%). Although it obtains a satisfactory performance, its imperfect recognition may slightly impair the evaluation of trading strategies, since it may point out the incorrect securities for the strategies to trade on.

Results of Trade-At-End Even with the simple trading policy, our model achieves an average return of 1.74% and an exceed return of $84443 (844%) in 1-day trading.

Experiments on Vader and BERT-SST show that the sentiment of a news article can indicate the stock movement to some extend. For example, BERT-SST achieves great winning rates, and it successfully outperforms the market index by a considerable margin. However, these signals tend to results in small stock movements. Thus, sentiment-based models achieve poor average returns. On the other hand, event-based models obtain much higher average returns, demonstrating the superiority of corporate events over news sentiments in indicating stock movements.

Although the Sentence model highly outperforms the sentiment-based methods, compared to our models, it is less robust to ambiguous articles, and it is more likely to miss the events that are described in several continuous sentences. By utilizing global context information and detecting events from the token level, our model identifies more trading signals and avoids more potential traps.

Bi-level Detection also achieves better performance than BERT-CRF under all the settings. The improvements mainly come from the high-level detector. By combining the global contextual information with token-level detected results, Bi-level Detection is more robust and more effective. When the low-level detector generates false alarms on some ambiguous tokens or fails to detect events that are not explicitly described, the high-level detector may point it out after analyzing the meaning of the entire article.

Results of Trade-At-Best This trading policy is designed to measure the ceiling of trading signals given by each model. Under this setting, Bi-level Detection obtains a 9.11% average return that dramatically exceeds all the sentiment-based models, indicating how significant the stock price changes when corporate events take place. Com-

Metric	Policy	Event Type									
		A	CT	DC	DI	GI	NC	RSS	SD	SR	SS
WR. %	TAE (1)	52.8	51.7	60.0	66.9	55.9	53.3	57.8	84.0	60.2	43.3
	TAE (3)	53.5	48.0	49.4	60.2	52.8	50.5	55.6	64.0	55.6	81.1
AR. %	TAE (1)	2.15	1.89	0.75	0.33	0.88	1.72	3.26	4.76	1.27	-0.04
	TAE (3)	1.93	2.05	0.02	1.02	1.19	1.32	4.25	5.49	1.51	4.99
	TAB (1)	9.98	12.32	5.71	1.74	4.63	9.40	8.19	8.02	5.26	4.27
	TAB (3)	13.46	17.51	10.95	4.21	7.73	13.24	19.68	13.13	8.79	17.91
Num.		960	721	39	135	282	697	26	29	225	5

Table 4: The table shows **Bi-level Detection**'s 1-day and 3-day trading results on each event, where **WR.** stands for the winning rate and **AR.** stands for the average return on each transaction. Each column respectively shows the results of Acquisition(A), Clinical Trial(CT), Dividend Cut(DC), Dividend Increase(DI), Guidance Increase(GI), New Contract(NC), Reverse Stock Split(RSS), Special Dividend(SD), Stock Repurchase(SR) and Stock Split(SS).

Model	TAE (1)			TAB (1)
	Win Rate	Ave. Return	Exc. Returns	Ave. Return
Bi-level Detection - Open	**54.5**||**34.2**%	**1.74**%	$**84443**	**9.11**%
Bi-level Detection - Close	51.4||29.5%	-0.07%	-$12026	4.50%

Table 5: This table shows **Bi-level Detection**'s 1-day trading results when start transactions respectively with the ***Open*** price and ***Close*** price at the minute that the event is detected.

pared to other event-based models, Bi-level Detection achieves much better performance in identifying corporate events.

5.3 Event Analysis

Table 4 shows Bi-level Detection's 1-day and 3-day trading results on each event. As shown in the table, the Dividend Increase has the smallest influence on the stock price, while the Reverse Stock Split and Clinical Trial significantly impact the stock prices.

Reverse Stock Split and Special Dividend are the most profitable corporate event, yet they are relatively rare. Acquisition, Clinical Trial and New Contract are ubiquitous, and they also lead to significant stock movements. However, comparisons between TAE and TAB's profits indicate that these events are relatively trickier to trade. Although the ideal policy TAB achieves dramatic profits, TAE makes much fewer profits, indicating that the stocks may oscillate significantly after these events. On the other hand, Special Dividend and Reverse Stock Split are comparatively easier to trade on.

Different events also affect the stock prices in different periods. We measure the influence by comparing TAB (1) and TAB (3)'s average return on the same event. They achieve close average returns on Acquisition, Clinical Trial, and Stock Repurchase, indicating that the stock prices do not continue to change sharply after the first day. In contrast, Stock Split impacts the stock price for a more extended period.

Thus, an ideal trading strategy should take the above factors into account. For example, it may assign different weights to each event based on their profitability and use a different policy to trade each event based on their potential price change patterns.

5.4 Profitability in Real-world

In this section, we discuss the possible profitability of the proposed strategy in real-world trading. Backtesting against historical data shows that the proposed strategy dramatically outperforms the market index. However, this result is based on two main assumptions.

First, we assume the cost of time in acquiring the news articles and making trading decisions is almost 0. Table 5 indicates the significance of timeliness, in which **Bi-level Detection - Open** starts transactions with the ***Open*** price at the minute that the news article is published (e.g., price at 11:23:00), while **Bi-level Detection - Close** starts with the ***Close*** price at that minute (e.g., price at 11:23:59). As shown in the table, when the model trades tens of seconds after the publish time of the news article, it greatly underperforms the market index and achieves a negative average return on each transaction. These results demonstrate that the profitability of an event-based model highly depends on how "quick" one can perform tradings after a piece of news is published.

Second, we assume we can always buy/sell the desired amount of stock shares and ignore the liquidity of the stocks. When the investment scale is relatively small, this assumption doesn't have a big impact. However, as the investment scales up, the liquidity may greatly constrain the model's profitability.

6 Related Works

Text-base Stock Prediction Existing methods usually count on textual features and sentiment analysis to forecast the stock movements (Hagenau et al., 2013; Mohan et al., 2019; Xie and Jiang, 2019; Huynh et al., 2017; Liu, 2018; Mittal and Goel). Hagenau et al. (2013) utilizes N-Gram, Noun-phrases, and 2-words combinations of corporate announcements to predict the stock movement. The influence of financial news on the stock market is also widely explored (Engle and Ng, 1993; Tetlock, 2007). In recent years, researchers resort to support vector machine and deep neural network to analyze financial news articles (Liu, 2018; Xie and Jiang, 2019; Ding et al., 2015; Huynh et al., 2017). Mohan et al. (2019) combines news text (e.g., sentiment, tf-idf, and word2vec representation) with historical stock data to predict future stock prices. Event-based stock predictions are also introduced. Ding et al. (2015) extract events from news articles, calculate event embedding, and use it to predict direction of stock moves. Ben Ami and Feldman (2017) build sentiment-type trading signals with word polarities and event-type trading signals with existing information extraction platform and demonstrates the superiority of event over sentiment in making trading decisions.

Event Detection General domain event detection that aims to recognize structured schemata/frames from the text has been widely explored by data-driven supervised learning methods (Ahn, 2006; Mitamura et al.). In the economic domain, however, existing approaches (Arendarenko and Kakkonen, 2012; Hogenboom et al., 2013; Xie et al., 2013) usually exploit knowledge-based and rule-based methods, which require extensive hand-designed rules and ontology. Recent works conceptualize corporate events as sequences of text that reported company-related occurrences and introduce data-driven methods to solve financial event detection with text classification (Ein-Dor et al., 2019; Jacobs et al., 2018). Ein-Dor et al. (2019) explored a Wikipedia-based supervised method to detect the sentences that may include corporate events. Jacobs et al. (2018) propose a multi-label sequence classification model to detect specific corporate events from news articles.

7 Conclusion

This paper introduces an event-driven trading strategy based on corporate event detection from news articles. We introduce a bi-level event detection model that utilizes global and local information to identifies corporate events. Experiments on the presented dataset EDT demonstrate the proposed model's superiority over all the baselines. The results also signify the corporate event's timeliness and effectiveness in indicating stock movement.

In future work, we plan to explore more on both the event detection model and trading policy. We expect to involve external knowledge and few-shot learning methods to relieve the event detection model from the data imbalance and data-scarce scenarios. On the trading policy side, we aim to explore more types of events and customize different policies for each event based on the potential price change patterns it may lead to.

Acknowledgments

Han Liu is grateful for the support of National Science Foundation (BIGDATA 1840866, RI 1408910, CAREER 1841569, TRIPODS 1740735), Alfred P Sloan Fellowship, and DARPA QEDRML. We sincerely thank Polygon.io for their high-quality historical stock price data.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Ernest Arendarenko and Tuomo Kakkonen. 2012. Ontology-based information and event extraction for business intelligence. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 89–102. Springer.

Zvi Ben Ami and Ronen Feldman. 2017. Event-based trading: Building superior trading strategies with state-of-the-art information extraction tools. *Available at SSRN 2907600*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*.

Liat Ein-Dor, Ariel Gera, Orith Toledo-Ronen, Alon Halfon, Benjamin Sznajder, Lena Dankin, Yonatan Bilu, Yoav Katz, and Noam Slonim. 2019. Financial event extraction using Wikipedia-based weak supervision. In *Proceedings of the Second Workshop on Economics and Natural Language Processing*, pages 10–15, Hong Kong. Association for Computational Linguistics.

Robert F Engle and Victor K Ng. 1993. Measuring and testing the impact of news on volatility. *The journal of finance*, 48(5):1749–1778.

G Pui Cheong Fung, J Xu Yu, and Wai Lam. 2003. Stock prediction: Integrating text mining approach using real-time news. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, pages 395–402. IEEE.

CHE Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text.

Michael Hagenau, Michael Liebmann, and Dirk Neumann. 2013. Automated news reading: Stock price prediction based on financial news using context-capturing features. *Decision Support Systems*, 55(3):685 – 697.

Alexander Hogenboom, Frederik Hogenboom, Flavius Frasincar, Kim Schouten, and Otto Van Der Meer. 2013. Semantics-based information extraction for detecting economic events. *Multimedia Tools and Applications*, 64(1):27–52.

Huy D. Huynh, L. Minh Dang, and Duc Duong. 2017. A new model for stock price movements prediction using deep neural network. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, SoICT 2017, page 57–62, New York, NY, USA. Association for Computing Machinery.

Gilles Jacobs, Els Lefever, and Véronique Hoste. 2018. Economic event detection in company-specific news text. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 1–10, Melbourne, Australia. Association for Computational Linguistics.

Joshi Kalyani, Prof Bharathi, Prof Jyothi, et al. 2016. Stock trend prediction using news sentiment analysis. *arXiv preprint arXiv:1607.01958*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Huicheng Liu. 2018. Leveraging financial news for stock trend prediction with attention-based recurrent neural network. *arXiv preprint arXiv:1811.06173*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. Overview of tac kbp 2015 event nugget track.

Anshul Mittal and Arpit Goel. Stock prediction using twitter sentiment analysis.

S. Mohan, S. Mullapudi, S. Sammeta, P. Vijayvergia, and D. C. Anastasiu. 2019. Stock price prediction using news sentiment analysis. In *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 205–208.

Dev Shah, Haruna Isah, and Farhana Zulkernine. 2018. Predicting the effects of news sentiments on the stock market. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4705–4708. IEEE.

Paul C Tetlock. 2007. Giving content to investor sentiment: The role of media in the stock market. *The Journal of finance*, 62(3):1139–1168.

Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. 2013. Semantic frames to predict stock price movement. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–883, Sofia, Bulgaria. Association for Computational Linguistics.

Yancong Xie and Hongxun Jiang. 2019. Stock market forecasting based on text mining technology: A support vector machine method. *arXiv preprint arXiv:1909.12789*.

A Results for 3 days trading

As shown in table 6, the trading results on 3-day trading is consistent with the results of 1-day and 2-day tradings.

Model	**TAE (3)**			**TAB (3)**	
	Win Rate	**Ave. Return**	**Exc. Returns**	**Ave. Return**	**Num. Trans.**
Vader (Gilbert, 2014)	55.6||41.9%	0.69%	-$152	4.11%	4327
BERT-SST (0.995)	**56.1**||**42.8**%	0.98%	$1643	8.07%	2378
BERT-SST (0.9)	54.1||41.1%	0.73%	$3490	6.23%	15663
Sentence (Jacobs et al., 2018)	54.3||42.6%	1.59%	$40066	11.11%	2881
BERT-CRF	51.5||41.2%	1.57%	$53152	12.94%	3533
Bi-level Detection (ours)	52.0||42.0%	**1.71**%	$**55339**	**13.11**%	3118

Table 6: This table shows the 3-day trading result, in which we start each transaction at the news article's publish time and end the transaction after 3 trading day. **Win Rate** stands for the overall winning rate (rate of transactions that have a return over 0) || big win rate (rate of transactions that have a return over 1%). **Ave. Return** stands for the average return on each transaction. **Exc. Return** stands for the total excess returns over the market when starting with $10000 and invest $2000 to each detected trading signal. **Num. Trans.** stands for the number of transactions (valid trading signals) of each model.

Language-based General Action Template for Reinforcement Learning Agents

Ryosuke Kohita
IBM Research
kohi@ibm.com

Akifumi Wachi
IBM Research
akifumi.watchi@ibm.com

Daiki Kimura
IBM Research
daiki@jp.ibm.com

Subhajit Chaudhury
IBM Research
subhajit@jp.ibm.com

Michiaki Tatsubori
IBM Research
mich@jp.ibm.com

Asim Munawar
IBM Research
asim@ibm.com

Abstract

Prior knowledge plays a critical role in decision-making, and humans preserve such knowledge in the form of natural language (NL). To emulate real-world decision-making, artificial agents should incorporate such generic knowledge into their decision-making framework through NL. However, since policy learning with NL-based action representation is intractable due to NL's combinatorial complexity, previous studies have limited agents' expressive power to only a specific environment, which sacrificed the generalization ability to other environments. This paper proposes a new environment-agnostic action framework, the language-based general action template (L-GAT). We design action templates on the basis of general semantic schemes (FrameNet, VerbNet, and WordNet), facilitating the agent in finding a plausible action in a given state by using prior knowledge while covering broader types of actions in a general manner. Our experiment using 18 text-based games showed that our proposed L-GAT agent which uses the same actions across games, achieved a performance competitive with agents that rely on game-specific actions. We have published the code at https://github.com/kohilin/lgat.

1 Introduction

The incorporation of natural language processing (NLP) and reinforcement learning (RL) is an important research field for using knowledge, represented in the form of language, in the decision-making of artificial agents (Luketina et al., 2019). One critical topic is the capability to describe an agent's actions with natural language (NL) (Narasimhan et al., 2015; Yuan et al., 2019). An agent with such a capability can estimate the plausibility of actions on the basis of prior knowledge accessible through language (Narasimhan et al., 2018). Suppose that an agent receives a request, "*give me some water*"; a common sense idea like "*water exists in the kitchen*" will definitely help the agent determine the right direction to go in. If actions are represented in language, for example, GO TO KITCHEN, we can straightforwardly connect knowledge to actions by referring to language resources (Fulda et al., 2017). NL is useful for accessing knowledge to achieve plausible decision-making, and this language capability would be fundamental in developing intelligent agents.

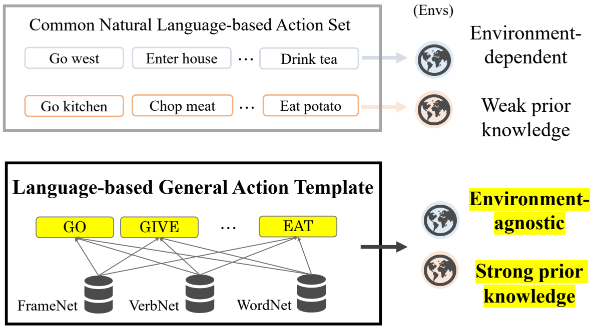

Figure 1: Common action templates and L-GAT.

However, NL is complicated for current RL agents to acquire due to its high expressive power (Hausknecht et al., 2020). The rich vocabulary and complex grammar of NL result in a huge action space that is intractable for existing RL algorithms. Although we can train an agent by restricting the expressive power to a specific environment (Narasimhan et al., 2015; He et al., 2016), such constraints sacrifice the inherent advantage of using NL as the action representation for domain-independent learning. Our objective is to make better trade-offs between expressive power and the learnability of NL-based RL.

Interactive fiction (IF) games serve as a practical testbed for NL-based RL, where an agent and environment communicate with each other through

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2125–2139
August 1–6, 2021. ©2021 Association for Computational Linguistics

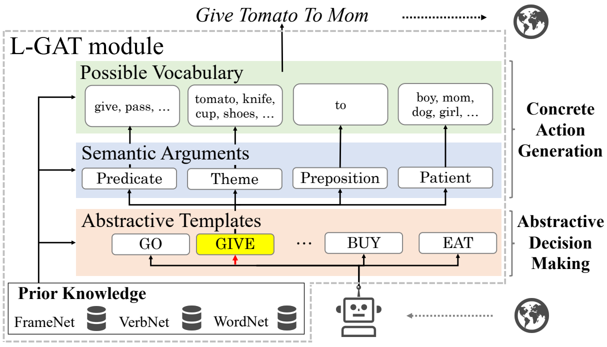

Figure 2: Action generation flow with L-GAT.

textual information (Côté et al., 2019). The agent needs to understand a textual state and generate an appropriate NL action command. In the experiments of previous studies, various constraints on the action space have been used for the convergence of learning, such as simple grammar and vocabulary (Narasimhan et al., 2015), effective ground-truth actions for a state (He et al., 2016), and game-specific templates (Hausknecht et al., 2020). These constraints are problematic, especially when applying the same algorithm to other environments. There are several studies that address this by implementing heuristic rules (Hausknecht et al., 2019) or training an action generator with observation-action pairs of human-play (Yao et al., 2020), but they still have a non-trivial bias toward IF games.

In terms of prior knowledge, word embeddings and language models have shown success at reducing the action space (Fulda et al., 2017; Yao et al., 2020). Although such continuous representations have achieved notable performances for broader NLP tasks, they are basically trained with word co-occurrences. We cannot flexibly manipulate the knowledge contained in these continuous representations because it is difficult to selectively encode the desired knowledge into them (Zhou et al., 2020). To make the most of prior knowledge expressed in NL, we should take advantage of other linguistic resources as well that provide fine-grained information more explicitly, such as the hierarchical structure of words (Miller, 1998), the semantics of a sentence (Fillmore et al., 2003), and common sense (Speer et al., 2017).

Thus, we propose a new environment-agnostic action framework on the basis of general semantic schemes, the language-based general action template (L-GAT). Figure 1 shows the overall architecture compared with the environment-dependent action templates commonly used, and Figure 2 illustrates the flow of generating an action with L-GAT. The agent with L-GAT first determines "*what to do*" at an abstractive level with generally defined action templates and then specifies "*how to do it*" by generating a concrete action on the basis of connected prior knowledge.

Our contributions with L-GAT are three-fold. First, we propose an environment-agnostic action framework based on general semantic schemes such as FrameNet (Fillmore et al., 2003), VerbNet (Schuler, 2006), and WordNet (Miller, 1998). Second, we develop a hierarchical action-generation algorithm in which the agent performs abstractive decision-making and then generates a concrete action command. Third, we introduce a method for dynamically reducing the action space with static knowledge from multiple external resources and contextual information from a state. We have published the code with which future studies can easily use L-GAT as an action generation module.

2 Related Work

The action space problem of NL-based actions has been addressed in previous work. LSTM-DQN (Narasimhan et al., 2015) enables an agent to learn a policy in a simple verb-object format in synthetic environments. DRRN (He et al., 2016) is a ranking-based method in which the agent selects an action from among admissible actions given by a game engine. Game-specific templates were proposed for Jericho (Hausknecht et al., 2020). The agent selects one of the templates and fills in the gaps in the chosen template. TDQN (Hausknecht et al., 2020) and KGA2C (Ammanabrolu and Hausknecht, 2020) agents have used these templates and succeeded at learning a policy in various games. NAIL (Hausknecht et al., 2019) solves IF games with action generation heuristics.

Language resources have also been used for reducing the action space. Fulda et al. (2017) extracted the affordances of words with word embeddings and restricted objects that can be taken for a specific verb. A language model is leveraged to filter non-plausible word combinations (Kostka et al., 2017). Recently, Yao et al. (2020) proposed the contextual action language model (CALM), which trains a language model with human-play transcriptions and uses it as an action candidate generator. By combining it with a ranking-based method, CALM showed significant performance even without a ground-truth for admissible actions.

3 Problem Setting

In this section, we formally define our problem setting, RL agents with NL-based actions. An NL-based action a, such as GO TO KITCHEN and GIVE SOME WATER TO JOHN, consists of N words: $a = \{w_1, w_2, \ldots, w_N\}, w_i \in \mathcal{V}$, where each w_i is a word and $\mathcal{V}$ is vocabulary. Given a state s, we represent an optimal NL-based action in the state with $a^*_{(s)}$ and the words composing it with $w^*_{(i,s)}$, respectively. Following the previous studies, each $w \in \mathcal{V}$ is estimated with an independent Q-function as $Q_{(i)}(s, w)$, where i is the position of a word, and $\mathcal{Q}(s, a)$ is defined as $\Sigma_i Q_{(i)}(s, w)$. A policy π is evaluated with a cumulative reward as:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right],$$

where R is a reward function for a state-action pair, and γ is a discount factor. Then, the Q-functions $\mathcal{Q}^*$, Q^* corresponding to the optimal policy are learned by minimizing the following loss:

$$\mathcal{L} = (R(s, a) + \gamma \bar{\mathcal{Q}}^*(s', a') - \mathcal{Q}^*(s, a))^2,$$

where $\bar{\mathcal{Q}}^*(s', a') = \Sigma_i \max_{w'} \bar{Q}^*_{(i)}(s', w')$ with the next state s', the next word w', and the target Q-function $\bar{Q}$. Then, we obtain each word of the optimal action as: $w^*_{(i,s)} = \arg\max_w Q^*_{(i)}(s, w)$. However, this optimization could be intractable depending on N and the size of $\mathcal{V}$ due to the exponential complexity $\mathcal{O}(|\mathcal{V}|^N)$.

Hence, we consider restricting the vocabulary space for each position i for the action and state s to make this optimization problem tractable. Specifically, we want to find a subset $\mathcal{V}_{(i,s)}$ of $\mathcal{V}$ for each (i, s) such that $|\mathcal{V}_{(i,s)}|$ is much smaller than $|\mathcal{V}|$. Also, $\mathcal{V}_{(i,s)}$ must include the optimal word $w^*_{(i,s)}$ for the original optimization problem because replacing $\mathcal{V}$ with $\mathcal{V}_{(i,s)}$ should not change the optimal solution. Assuming that we have $\mathcal{V}_{(i,s)} \subset \mathcal{V}$, the computational complexity decreases from $\mathcal{O}(|\mathcal{V}|^N)$ to $\mathcal{O}(\prod_{i=1}^{N} |\mathcal{V}_i|)$, where $\mathcal{V}_i$ is one of the largest $\mathcal{V}_{(i,s)}$ for all s.

To define $\mathcal{V}_{(i,s)}$, we use knowledge $\mathcal{K}$:

$$\mathcal{V}_{(i,s)} = \varphi(i, s, \mathcal{K}, \mathcal{V}),$$

where φ is a set-valued function. Then, we need to determine the function φ that minimizes $|\mathcal{V}_{(i,s)}|$ but keeps the i-th word of an optimal NL-based action in $\mathcal{V}_{(i,s)}$:

$$\min |\mathcal{V}_{(i,s)}|, \text{ s.t. } w^*_{(i,s)} \in \mathcal{V}_{(i,s)}.$$

Note that, although the above formulation assumes that $\mathcal{V}_{(i,s)}$ always contains the optimal words $w^*_{(i,s)}$, it is practically impossible because the optimal words are unknown a priori and estimated through the learning process. Therefore, we need to approximate φ so that the estimated $\mathcal{V}_{(i,s)}$ is likely to contain $w^*_{(i,s)}$ as much as possible. We introduce our implementation of the approximation in Section 4.4.

4 Method

We now introduce our proposed method; the language-based general action template (L-GAT). L-GAT is a framework of NL-based actions for environment-agnostic RL agents. In L-GAT, we define actions on the basis of general semantic schemes, such as FrameNet (Fillmore et al., 2003), VerbNet (Schuler, 2006), and WordNet (Miller, 1998), which enables an agent to use connected knowledge to reduce the action space.

In this section, we first provide an overview of L-GATand then give details on its action command, definition, and generation algorithm.

4.1 Overview

L-GAT adopts hierarchical modeling for action generation to handle the vast space of NL-based action. Specifically, an agent with L-GAT first determines "*what to do*" at an abstractive level (e.g., decide to **give**) and then next determines "*how to do it*" at a concrete level (e.g., decide to give **tomato** to **mom**) as shown in Figure 2. This two-step strategy is intuitive and natural as a general procedure for decision-making (Lazaridou et al., 2020). From the viewpoint of RL, we can reduce the action space because the number of abstractive actions is much smaller than that of all possible words in the whole vocabulary. Also, the chosen abstractive action further restricts the words available for generating a complete action (e.g., we cannot eat a table)

To use such hierarchical modeling, L-GAT defines the **Abstractive Action Template** with a hierarchical structure consisting of three components: **Frame**, **Role**, and **Lexicon**. Figure 3 illustrates the hierarchical structure of

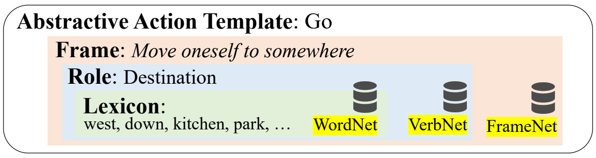

Figure 3: Abstractive action template Go and its hierarchical components; Frame, Role, and Lexicon.

an abstractive action template, Go. The frame defines the semantics of Go (e.g., *move oneself to somewhere*), the role represents a conceptual argument required to perform the action (e.g., `Destination`), and the lexicon represents concrete words that can be used for one of the roles in the frame (e.g., *west*, *down*, and *kitchen*). Using the generation algorithm, the agent selects an appropriate word from the lexicon for each frame's role in the chosen abstractive action template.

FrameNet and VerbNet inspired the hierarchical structure of L-GAT. Both resources provide semantic schemes that describe our daily behaviors conceptually. For example, similar to the Go template, FrameNet has a relevant scheme, "self-motion", with required arguments such as Goal and Source. Also, we connect the lexicon of L-GAT to WordNet to represent the hierarchical relationships of words. The word hierarchy enables agents to generalize candidate words (e.g., hyponyms of "*direction*," such as "*west*," can be used for the Go template) and interact with diverse environments. Thus, by designing actions on the basis of these general semantic schemes, we can naturally make L-GAT environment-agnostic and familiar with general prior knowledge.

Note that the templates of L-GAT are easily configurable depending on the environment. Such customizability is practically essential to applying the scheme to diverse domains, which is costly or difficult for an environment-dependent action set or language model-based generator.

4.2 Action Command

L-GAT generates an action command in a fixed format as: $a = v + n_1 + p + n_2$, with at most four slots for a verb (v), two nouns (n_1, n_2), and a preposition (p). An action command is generated by the action templates explained in the next section, and n_1, n_2, and p are not necessarily used depending on the template. To prevent the exponential growth of the action space, we decided to add these hard constraints to a generable format. However, we consider the aforementioned format to be able to cover most of the primary actions demanded by agents, such as *go kitchen* and *put cup on table*.

4.3 Action Definition

This section introduces the abstractive action template and its components. In this paper, we manually defined 41 abstractive action templates for L-GAT (see Appendix A). For the construction of each template, we chose relevant semantic schemes from FrameNet and VerbNet, and then aggregated them in a compatible form with the actions of RL agents. Table 1 shows three templates as references, and we will refer to them throughout this section.

Abstractive Action Template. A template T is defined as a tuple $T = \langle f, r_{(v)}, r_{(n_1)}, r_{(p)}, r_{(n_2)} \rangle$ where f is the frame, and $r_{(v|n_1|p|n_2)}$ are the roles for each slot. The frame defines the template's semantics. The roles represent required arguments for performing an action. Each word in an action command is an instance of its corresponding role and selected from the lexicon explained later.

Frame. Each frame has connections to semantically relevant entities in FrameNet (called *Semantic Frames*), and ones in VerbNet optionally (called *Verb Classes*). For example, the Give template is related to the semantic frame *Giving* and the verb class *give-13.1*. We use FrameNet as the basis because the granularity of the descriptions is more suitable for our purpose than VerbNet.

Role. For the definition of our roles, we borrow VerbNet's *Thematic Roles*, which refer to the semantic relationship between a predicate and its argument. VerbNet defines 30 thematic roles in total, and L-GAT uses 15 of them by taking into account their frequency and meaning. FrameNet also has a similar concept called *Frame Elements*; however, it is too fine-grained for RL agents' actions. Therefore, we use the thematic roles of VerbNet as the basis and annotate the related frame elements as additional information. We also define three special roles: `Predicate`, `Preposition`, and `Null`. `Predicate` is used only for the verb slot and references to verb nodes in WordNet. We selected these nodes on the basis of the representative verbs given by the connected frame entities of FrameNet and VerbNet (e.g., we chose "go.v.02" and "enter.v.01" for the

	f	$r_{(v)}$	$r_{(n_1)}$	$r_{(p)}$	$r_{(n_2)}$
L-GAT	Go	`Predicate`	`Destination`	`Null`	`Null`
FrameNet	Self-motion	-	Goal, Place, Area, Path	-	-
VerbNet	-	-	Destination	-	-
WordNet	-	go.v.02, enter.v.01	location.n.01, ...	-	-
L-GAT	Give	`Predicate`	`Theme`	`Preposition`	`Recipient`
FrameNet	Giving	-	Theme	-	Recipient
VerbNet	give-13.1	-	Theme	-	Recipient
WordNet	-	give.v.01	object.n.01	(to)	people.n.01, person.n.01
L-GAT	Eat	`Predicate`	`Patient`	`Null`	`Null`
FrameNet	Ingestion	-	Food	-	-
VerbNet	eat-39.1-1	-	Patient	-	-
WordNet	-	eat.v.01	food.n.01	-	-

Table 1: Examples of abstractive action templates in L-GAT with connections to FrameNet, VerbNet, and WordNet.

`Predicate` of the Go template on the basis of representative verbs given by the semantic frame "self-motion"). `Preposition` has prepositions available that often accompany verbs in a frame, and we manually defined these prepositions. `Null` means not using the slot for action generation. For example, the Go template uses only v and n_1.

Lexicon. The simplest way of defining the lexicon is to list all possible words manually or statistically; however, such a procedure would be non-scalable or hard to configure. Therefore, we decided to annotate nodes in WordNet as the representation. We selected general nodes such as "location.n.01" for the `Destination` of the Go template and "food.n.01" for the `Patient` of the Eat template. The hierarchical word relations in WordNet enables L-GAT to structurally determine candidate words such as by choosing the hyponyms of annotated nodes.

4.4 Action Generation

We propose three techniques for action generation with L-GAT: **Hierarchical Prediction**, **Word Masking**, and **Template Masking**. An agent with L-GAT generates an action as follows. The agent determines the abstractive action template and then selects a word for each slot from among candidates (Hierarchical Prediction). The candidates for each slot are filtered by using information from the state and the chosen template (Word Masking). By precomputing Word Masking for all templates in advance of the template selection, L-GAT can provide possible templates for a state by excluding templates that cannot generate any action command (Template Masking). The algorithm is described in Algorithm 1 and Figure 4. L-GAT has five Q-functions; one for estimating the template $Q(s, T)$, and the others for estimating words $Q_{(\cdot)}(s, w)$ where $(\cdot)$ denotes one of the slots v, n_1, p, or n_2.

Algorithm 1 Action Generation with L-GAT.

```
Input: $s$, Output $a$
$\mathcal{V}_{(\cdot,*,s)} \leftarrow$ compute the restricted vocabulary for all templates
$\mathcal{T} \leftarrow$ exclude templates with empty $\mathcal{V}_{(\cdot,T,s)}$
$T \leftarrow$ select a template from $\mathcal{T}$
$v, n_1, p, n_2 \leftarrow$ select words from $\mathcal{V}_{(\cdot,T,s)}$
$a \leftarrow$ decode with $v, n_1, p, n_2$
```

4.4.1 Hierarchical Prediction

L-GAT first selects a template T with $Q(s, T)$ and then generates words for each slot as: $w_{(\cdot)} = \arg\max_{\hat{w}} Q_{(\cdot)}(s, \hat{w}), \hat{w} \in \mathcal{V}_{(\cdot,T,s)}$, where $\mathcal{V}_{(\cdot,T,s)}$ is vocabulary that is restricted for the target slot of the template T in the state s. L-GAT computes $\mathcal{V}_{(\cdot,T,s)}$ as: $\mathcal{V}_{(\cdot,T,s)} = \{w \mid m(\cdot, s, T, w) = 1, w \in \mathcal{V}\}$, where m is a *masker function* that returns 1 for a generable word in the target slot by considering s and T. The masker function is our approximation of φ explained in Section 3, and we introduce the details in the next section.

4.4.2 Word Masking

We included five sub-masker functions; Role masker, Language Model (LM) masker, Part-of-Speech (PoS) masker, Stopword masker, and Observation masker. Each of them returns 1 for generable words. Then, we define $m(\cdot, s, T, w) = 1$ only when all the sub-maskers return 1. Next we give the definitions of the five sub-maskers.

Role masker. The Role masker filters words that are not semantically compatible with a given role

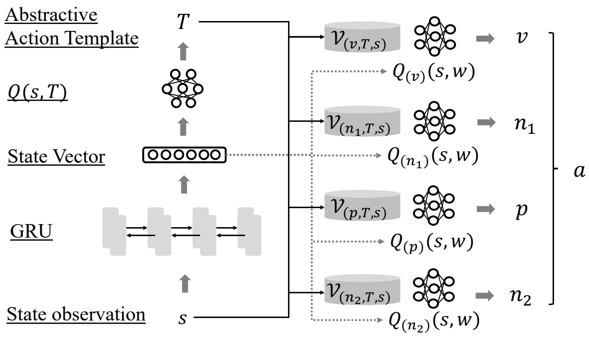

Figure 4: L-GAT model without Template Masking.

by referring to the annotated WordNet nodes as mentioned in §4.3. For the noun slots, it enables hyponym words of the reference nodes to be produced. For example, for the n_1 slot of the EAT template, it returns 1 for hyponym words of the *food.n.01* node such as *water* and *tomato* (see Table 1). For the v slot (i.e., `Predicate`), it returns 1 for lemmas of the annotated nodes themselves such as *go.v.01* = *go* and *enter.v.01* = *enter*. All words have 1 in the p slot, and no words have 1 in any `Null` slot.

LM masker. The LM masker filters contextually irrelevant candidates with an LM. The LM takes an observation suffixed with a verb of the template as the input (an example is given in Appendix B), and it predicts the following word. On the basis of the estimated probability of next words, the LM masker returns 1 for the top-k words. At the v and p slots, all words have 1. We used a pre-trained GPT-2 (Radford et al., 2019) and set $k = 50$.

PoS, Stopword, and Observation maskers. The PoS masker filters words in which a PoS is not matched with the expected one. Specifically, it returns 1 for verbs in the v slot, for nouns in the n_1 and n_2 slots, and for prepositions in the p slot. The word-PoS mapping follows WordNet in terms of verbs and nouns, and we prepared a fixed list of prepositions. The Stopword masker prohibits the generation of stopwords such as determiners (e.g., *a*, *the*) and pronouns (e.g., *he*, *them*). We prepared a fixed list, and the masker returns 1 for words that are not on the list. The Observation masker filters unseen words from observation. It returns 1 for words that appeared in the last observation.

4.4.3 Template Masking

L-GAT also filters unavailable templates in a state with the $\mathcal{V}_{(\cdot,T,s)}$ of each template. $\mathcal{V}_{(\cdot,T,s)}$ is empty when one of the sub-maskers denies the use of a word, and this occurs for all words in $\mathcal{V}$. An empty $\mathcal{V}_{(\cdot,T,s)}$ means that no appropriate objects exist for performing the action (e.g., we cannot eat anything if no eatable objects exist.). Thus, L-GAT selects a template as: $T = \arg\max_{T\in\mathcal{T}} Q(s,T)$, where $\mathcal{T}$ is a set of templates in which $\mathcal{V}_{(\cdot,T,s)}$ is not empty for all slots except for NULL slots.

Limitations	Methods				
	TD	KG	NA	CA	LG
Game-specific vocab	y	y	y	n	y
Game-specific template	y	y	n	n	n
Admissible actions	y	y	n	n	n
Human play data	n	n	n	y	n
Tuned Rules for IFGs	n	n	y	n	n
Exclude no-effect acts	y	y	n	y	n

Table 2: Limitations in action generation. Method names come from their first two characters.

5 Experiment

In our experiment, we measured performance in IF games in Jericho (Hausknecht et al., 2020). We compared our L-GAT agent with agents who rely on environment-dependent actions. We also performed detailed analysis such as on action coverage and action space reduction and masker ablation study on L-GAT. Furthermore, to evaluate the generalization ability of L-GAT, we also conducted an extended experiment in which a single agent solved all games.

5.1 Settings

Game environment. We selected 18 games in Jericho on the basis of the performance in previous studies. We did not select any game that was too hard for most of the existing agents to learn.

Baselines. There were four existing agents tested in Jericho; TDQN (Hausknecht et al., 2020), KGA2C (Ammanabrolu and Hausknecht, 2020), NAIL (Hausknecht et al., 2019), and CALM (Yao et al., 2020). The limitations related to their action generation are summarized in Table 2 (see also §2).

In the experiments with TDQN, KGA2C, and CALM, these methods preselected steps by excluding no-effect actions that do not change the world state for faster and stable learning. This limitation is not appropriate for testing general action sets such as L-GAT because the game engine explicitly restricts the action space.

Games	Max Score	TDQN	KGA2C	CALM	NAIL	TDQN+	L-GAT	Cover Ratio (#)
		(w/ step preselection)			(w/o step preselection)			
balances	51	4.8	10.0	9.1	**10.0**	9.8	8.8	53% (89)
deephome	300	1.0	1.0	1.0	13.13	13.2	**14.9**	87% (100)
detective	360	169.0	207.9	289.7	136.9	**270.9**	214.1	100% (46)
enchanter	400	8.6	12.1	19.1	0.0	**22.9**	0.0	73% (105)
gold	100	4.1	-	-	**3.0**	0.0	0.0	86% (331)
inhumane	90	0.7	3.0	25.7	**0.6**	0.0	0.0	95% (84)
jewel	90	0.0	1.8	0.3	**1.6**	0.6	0.0	74% (81)
library	30	6.3	14.3	9.0	0.9	4.1	**7.6**	69% (29)
ludicorp	150	6.0	17.8	10.1	8.4	**11.9**	6.1	97% (152)
omniquest	50	16.8	16.8	6.9	**5.6**	4.4	5.0	88% (75)
reverb	50	0.3	-	-	0.0	0.0	**1.0**	82% (65)
snacktime	50	9.7	0.0	19.4	0.0	**10.0**	0.0	88% (26)
spellbrkr	600	18.7	21.3	40.0	**40.0**	37.6	39.4	57% (109)
spirit	250	0.6	1.3	1.4	1.0	1.5	**1.7**	89% (209)
temple	35	7.9	7.6	0.0	**7.3**	2.6	5.0	94% (80)
zork1	350	9.9	34.0	30.4	10.3	4.7	**17.1**	95% (203)
zork3	7	0.0	0.1	0.5	**1.8**	0.3	0.4	92% (192)
ztuu	100	4.9	9.2	3.7	0.0	3.6	**4.9**	62% (78)
Norm Avg.	-	9.8%	14.1%	15.5%	8.0%	9.9%	8.9%	82%

Table 3: Average scores of last 100 episodes. Those of TDQN, KGA2C, NAIL, and CALM come from their respective papers. TDQN, KGA2C, and CALM cannot be fairly compared with L-GAT because of step preselection by excluding no-effect actions. TDQN+ and L-GAT are our implementations containing all steps regardless of action effectivity. Cover ratio shows proportion of generable walkthrough actions with L-GAT, and number in parentheses is total number of effective walkthrough actions.

To focus on the comparison of environment-dependent and agnostic action templates, we re-implemented the TDQN algorithm in our code base (denoted as TDQN+), in which all steps were counted regardless of action effectivity. Also, we implemented TDQN+ without the admissible action limitation; the difference in the above limitations between TDQN+ and L-GAT is their template types only (i.e., game-specific or general templates). Thus, we mainly compared NAIL, TDQN+, and L-GAT. Although a comparison with the original scores of TDQN, KGA2C, and CALM would not be fair, we also add them as references for completeness.

Note that the vocabulary limitation is practically needed because any action containing unknown words for a game engine cannot be accepted even though the action command is semantically correct. CALM overcame this vocabulary issue by learning the word distribution of action commands used in IF games from human-play transcriptions.

Model and training details. Following the works of Hausknecht et al. (2020); Yao et al. (2020), we used a bidirectional GRU as our observation encoder, and the observation string was augmented by combining the observations returned by the "look" and "inventory" commands. All Q-functions were implemented with a multi-layered perceptron with the same hyperparameters. In an episode, we limited the number of steps to 100 at most and ran agents with ten environments in parallel. In addition to the rewards given by the games, we gave an exploration bonus of 1 when an agent found an unseen state in the episode (Yuan et al., 2019). We put the hyperparameter details in Appendix C. We trained three agents with different random seeds and used their average score for the evaluation.

5.2 Results

We report the results of the game performance, action coverage by L-GAT, action space reduction by L-GAT, and an ablation study on Word Masking. Finally, we introduce the results of a single agent that solved multiple games.

Game performance. Following the previous studies, we calculated the average score of the last 100 episodes, and the results are shown in Table 3. The average normalized score (raw score divided by the maximum score) was 8.0% for NAIL, 9.9% for TDQN+, and 8.9% for L-GAT. Even though the action templates of L-GAT were designed in a general manner, our agent achieved reasonable performance across games and outperformed NAIL and TDQN+ in six games.

NAIL and L-GAT performed poorly in enchanter and snacktime compared with TDQN+. In these games, a number of critical actions for advancing the story were missing because their generable actions are defined outside of a specific game. For example, artificial words like spells (e.g., *frotz*, *gnusto*) appear in enchanter, and they are intractable with L-GAT, which uses only common language. As an additional investigation, we added a "SPELL" action to L-GAT for producing these spell words and re-trained an agent with the enchanter game; the score of L-GAT then increased to 20.0, which was close to the score of TDQN+ (22.9).

The scores of KGA2C (14.1%) and CALM (15.5%) were significantly higher. This suggests that their techniques, such as the dynamic state graph encoding and the domain adaptation of LM's outputs, are promising, and we will investigate their importance for our framework as future work.

Walkthrough coverage. Jericho provides an action trajectory called a *walkthrough* that solves a game. We assessed how much L-GAT covered walkthrough actions. The detailed procedure of this assessment is explained in Appendix C. The right-most column in Table 3 shows the cover ratio of the walkthrough actions for L-GAT. The average coverage across games was 82%, and around 90% or more of the actions were covered in half of the games. A lower coverage ratio ($< 70\%$) was observed in balances (53%), spellbrkr (57%), ztuu (62%), and library (69%). In their walkthrough actions, we frequently found artificial words (e.g., *rezrov*), named entities such as person's name (e.g., *give card to Alan*), and modifiers for objects (e.g., *wear fur suit*, *take ticket 2306*). Although the first case might be out-of-scope of L-GAT, the second and third are critical because they must appear in an interaction with real-world environments.

Action space reduction. We evaluated how much the action space was reduced in L-GAT by applying Word Masking (4.4.2) and Template Masking (4.4.3). For each slot in each available template (v, n_1, p, and n_2), we counted the number of words that were accepted by these techniques. Compared with the size of the whole vocabulary, the number of words was significantly reduced by 94% for v (24.8 words on average after reduction), 82% for n_1 (117.9 words), 99% for p (7.7 words), and 59% for n_2 (292.2 words).

Masker ablation. Figure 5 shows the learning curves of the L-GAT agents without one of the sub-maskers for Word Masking in zork1. A significant effect was observed with the Role masker. Specifically, the agent without the Role masker took more episodes to find the scored trajectory and degraded the overall performance. The PoS and Stopword maskers were not critical, which might be because their restrictions are already covered by the Role masker. The LM and Observation maskers had non-trivial effects. Since the Role masker enables relatively broader words in noun slots (i.e., hyponyms of a general word), contextual restrictions given by the LM and Observation maskers helped the agent identify available objects.

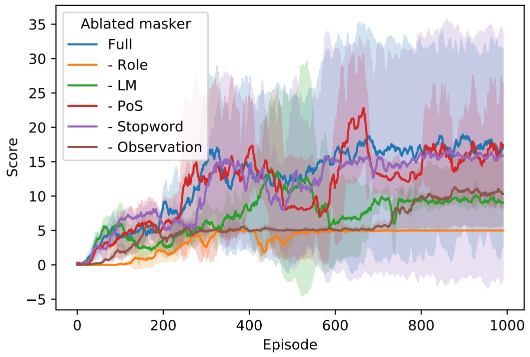

Figure 5: Masker ablation of L-GAT in zork1.

Game	Multi-task		Zero-shot	
	TDQN+	L-GAT	TDQN+	L-GAT
balances	0.0	1.5	0.0	0.8
deephome	1.0	1.1	1.0	1.1
detective	39.0	42.7	16.7	25.2
enchanter	0.0	0.0	0.0	1.4
gold	0.0	0.0	0.0	0.0
inhumane	0.0	0.0	0.0	0.0
jewel	0.0	0.0	0.0	0.0
library	0.3	0.0	0.0	0.0
ludicorp	4.2	4.1	2.8	2.3
omniquest	1.6	2.1	0.0	0.5
reverb	0.0	0.2	0.0	0.0
snacktime	0.1	0	0.0	0.0
spellbrkr	6.1	14.7	2.0	8.2
spirit	0.7	0.2	0.0	0.1
temple	0.0	0.0	0.0	0.0
zork1	0.0	0.4	0.0	0.0
zork3	0	0.0	0.0	0.0
ztuu	0	2.3	0.0	0.6
Norm avg.	1.1%	1.5%	0.4%	0.7%

Table 4: Average scores of last 100 episodes in game-independent agent experiments.

Game-independent agent To investigate the generalization ability of L-GAT, we conducted a multi-task experiment where a single agent solved all the games. We prepared two types of agents: 1) a *multi-task agent*, who learns with all 18 games and solves each game, and 2) a *Zero-shot agent*, who learns with 17 games and solves the remaining unseen game. Note that L-GAT can be naturally applicable to this experimental setting. In terms of the implementation of TDQN+, we accumulated all the unique templates used in training games as available templates. Table 4 shows the game scores. Both TDQN+ and L-GAT largely decreased in score compared with Table 3, in which we trained and tested a specific agent for each game. Also, as expected, the zero-shot situation was more difficult. Whereas L-GAT outperformed TDQN+, the scores were far from achieving the goal. It was still challenging to obtain an environment-agnostic agent even though the action set itself is defined in a general manner.

6 Limitations and Future Work

Although L-GAT showed a better performance in our experiment, there is still room for improvement to enhance its language capability. We here discuss L-GAT's limitations and future improvements.

Experiments in environments other than text-based games. One of the advantages of L-GAT is its action generality. Jericho is a reasonable testbed for L-GAT because of the diverse games and strong NL interpreter. However, the performance in other environments such as dialogue systems (Dinan et al., 2019) and 3D video games (Gordon et al., 2018) will also be critical metrics because objectives for IF-game agents are not necessarily compatible with ones for real-world agents. Testing with real-world like environments will lead to insights for improving L-GAT.

Enhancement of expressive power. In our experiment, L-GAT had a high action coverage in the IF games. However, the format of generable actions is fixed, and we observed several critical disabilities with generating frequently used expressions. For example, L-GAT cannot cope with multi-word expressions (e.g., *turn on*, *New York*) and modified nouns (e.g., *red cup*, *dog in the room*). Note that we can make them generable without adding extra slots, such as by inserting multi-word expressions in the vocabulary and integrating a reference resolution module. A technique that increases the expressive power but keeps or decreases the action space would be a desirable enhancement.

Zero-shot learning in L-GAT The general-purpose property of L-GAT can be seen as a zero-shot learning problem. Recently, Jain et al. 2020 proposed a new experiment to assess the adaptability of an RL agent to unseen states such as by using a new tool with the knowledge of known tools. L-GAT potentially can work for such a zero-shot situation. For example, let us assume that an L-GAT agent has a learned policy for attacking a monster with a hammer, and he then encounters a monster, but only a sword is available. Even if the agent did not know how to or even use a sword at all, he may be able to use it because L-GAT has knowledge on attacks with "*weapon*" (i.e., the `Instrument` role of the ATTACK template has a weapon node in WordNet, and "*sword*" is a hyponym of "*weapon*"; see Table 1 in Appendix A). Investigation into the adaptability to unseen states would be a promising analysis for L-GAT.

Connection to wide NLP resources and tools. Although we developed L-GAT on top of general semantic schemes such as FrameNet, VerbNet, and WordNet, the required information for efficient decision-making cannot be covered with only those resources. Intelligent agents must comprehend common sense, causality, and world-knowledge (Luketina et al., 2019). Research on NLP has created various resources such as for retrieving information from Wikipedia (Chen et al., 2017), inferring the next action from a current state (Zellers et al., 2018), and using a common-sense database (Speer et al., 2017). Integration of such resources into L-GAT is critical future work, and the three general semantic schemes would be helpful for bridging different knowledge bases.

7 Conclusion

We proposed L-GAT as an environment-agnostic language-based action framework for RL agents. We designed L-GAT so that an agent can have a higher language capability to generate its own actions while keeping the action space tractable by using prior knowledge. Our experiment with multiple IF games showed that the L-GAT agent competitively performed against agents with game-specific actions. We discussed the current limitations of L-GAT and its future improvements.

Ethical Consideration

Our research is on defining a new action framework for RL agents. Our proposed method, L-GAT, is for generating actions of RL agents, which is unlikely to produce ethically problematic sentences such as those in hate speech. Also, we can control the vocabulary so that an agent with L-GAT does not produce such problematic sentences. Therefore, we consider the ethical risks of L-GAT to be low and controllable.

References

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to Framenet. *International Journal of Lexicography*, 16(3):235–250.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: Visual question answering in interactive environments. In *Computer Vision and Pattern Recognition*.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Matthew Hausknecht, Ricky Loynd, Greg Yang, Adith Swaminathan, and Jason D. Williams. 2019. Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Ayush Jain, Andrew Szot, and Joseph Lim. 2020. Generalization to new actions in reinforcement learning. In *Proceedings of Machine Learning Research*.

Bartosz Kostka, Jaroslaw Kwiecien, Jakub Kowalski, and Pawel Rychlikowski. 2017. Text-based adventures of the golovin ai agent. In *IEEE Conference on Computational Intelligence and Games*.

Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. 2020. Multi-agent communication meets natural language: Synergies between functional and structural language learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2018. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63(1):849–874.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Open AI Blog*.

Karin Kipper Schuler. 2006. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2019. Counting to explore and generalize in text-based games. *arXiv preprint arXiv:1806.11525*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Definitions of Action Templates

We manually defined 41 abstractive action templates in total, and Table 6 shows extended examples. We give definitions of all of the templates at `https://github.com/kohilin/lgat`.

B Masker Function Details

Input for LM masker. The input of the LM masker is an observation string suffixed with one of the verbs in the template. For example, in the case of GIVE with the observation "*You have a cup of water. A boy stands in front of you.*", the input for the n_1 slot is "*... front of you. You gave the,*" and that for the n_2 slot is "*... front of you. You gave it to the.*"

Mask for primitive actions. L-GAT allows an agent to produce primitive actions, related to movement and belongings, regardless of the outputs of the masker function. Specifically, we forcibly assign 1 for direction words (*up*, *down*, *north*, *south*, *east*, *west*, *northwest*, *northeast*, *southwest*, *southeast*) to the n_1 slot in the GO template and *all* to the n_1 slots in the GET and DROP templates.

C Experimental Details

Hyperparameters. Table 5 shows our experimental hyperparameters and values searched for in the hyperparameter search. We tuned the α decay and initial ϵ, minimum ϵ, and ϵ decay for each game and each method. The other parameters were determined by testing with detective, temple, and zork1, and we applied the same settings to all of the games.

Machine specifications. We ran our experiment on Red Hat Enterprise Linux with an Intel(R) Xeon(R) CPU E5-2690 v4 at 2.60 GHz with 500GB of RAM and a single GPU, an NVIDIA TESLA K80.

Training time. Training with one game took approximately 10 ˜15 hours for the L-GAT agents, and 4 ˜6 hours for the TDQN+ agents depending on the game.

Model size. The number of trainable parameters for the L-GAT agent was about 10 million, which slightly changed depending on the vocabulary size of the games.

Type	Name	Value	Search
State Encoder	Algorithm	GRU	-
	Embedding size	256	128, 256
	Layer	2	-
	Hidden units	256	128, 256
Tokenizer	Algorithm	BERT subwording	-
	Max length	256	50, 256
Q-function	Layer	2	-
	Hidden units	200	-
	Activation	ReLU	-
Environment	Training episodes	1000	-
	Max steps	100	-
	Batch size	10	-
Optimizer	Algorithm	Adam	-
	Initial α	1e-3	1e-2, 1e-3
	α decay	0.5 or None	-
	α decay interval	100 episodes	-
Experience Replay	Buffer	Prioritized	-
	Replay batch size	128	128, 256
	Replay interval	5 steps	-
	Memory capacity	10000	-
Exploration	Algorithm	ϵ-greedy	-
	Initial ϵ	1.0 or .5	-
	Minimum ϵ	.05, .1, .2, or .3	-
	ϵ decay	.9975, .995, or .99	-
	ϵ decay interval	1000 steps	-
Loss	Loss function	Smoothed L1	-
	Discount factor	.95	-
	Reward clipping	-10, 10	-

Table 5: Hyperparameter settings and values in hyperparameter search.

Walkthrough experiment. In the experiment, we investigated how many walkthrough actions were covered by L-GAT. The same transition can be possible with different action strings. Therefore, for each state, we tried all generable strings of L-GAT and checked that at least one of them could produce the same transition with the gold walkthrough action. To judge if two transitions were the same or not, we used the game state hash provided by Jericho. Several walkthrough actions did not change the world state hash. We excluded such no-effect walkthrough actions from the evaluation since completely different actions were classified as replaceable for them.

D Learning Curves of Each Game

Figure 6 shows learning curves of the L-GAT and TDQN+ agents in each game.

E Game play Example in zork1

Table 7 shows the best play trajectory of our L-GAT agent in zork1.

	f	$r_{(v)}$	$r_{(n_1)}$	$r_{(p)}$	$r_{(n_2)}$
L-GAT	Ask	Predicate	Patient	Preposition	Theme
FrameNet	Questioning	-	Addressee	-	Topic
VerbNet	inquire-37.1.2	-	Patient	-	Theme
WordNet	-	ask.v.05	people.n.01, person.n.01	about	object.n.01, ...
L-GAT	Attach	Predicate	Theme	Preposition	Destination
FrameNet	Attaching	-	Items	-	Goal
VerbNet	shake-22.3-2-1	-	Theme	-	Destination
WordNet	-	attach.v.01	attachment.n.04	to	object.n.01
L-GAT	Attack	Predicate	Patient	Preposition	Instrument
FrameNet	Attack	-	Assailant	-	Weapon
VerbNet	-	-	Patient	-	Instrument
WordNet	-	attack.v.01, kill.v.01	monster.n.01, ...	with	weapon.n.01
L-GAT	Break	Predicate	Theme	Null	Null
FrameNet	Cause_to_fragment	-	Whole_patient	-	-
VerbNet	break-45.1	-	Theme	-	-
WordNet	-	break.v.02	object.n.01	-	-
L-GAT	Buy	Predicate	Theme	Null	Null
FrameNet	Commerce_buy	-	Goods	-	-
VerbNet	get-13.5.1	-	Theme	-	-
WordNet	-	buy.v.01	object.n.01	-	-
L-GAT	Check	Predicate	Location	Null	Null
FrameNet	Scrutiny	-	Ground	-	-
VerbNet	investigate-35.4	-	Location	-	-
WordNet	-	check.v.01, examine.v.02	entity.n.01	-	-
L-GAT	Clean	Predicate	Theme	Null	Null
FrameNet	Emptying	-	Theme, Source	-	-
VerbNet	clear-10.3	-	Theme	-	-
WordNet	-	clean.v.01	area.n.01, instrumentality.n.03	-	-
L-GAT	Close	Predicate	Theme	Null	Null
FrameNet	Closure	-	Container_portal	-	-
VerbNet	-	-	Theme	-	-
WordNet	-	close.v.01	window.n.01, ...	-	-
L-GAT	Cook	Predicate	Theme	Null	Null
FrameNet	Cooking_creation	-	Produced_food	-	-
VerbNet	cooking-45.3	-	Theme	-	-
WordNet	-	make.v.03, cook.v.01	food.n.01	-	-
L-GAT	Cut	Predicate	Patient	Null	Null
FrameNet	Cutting	-	Item	-	-
VerbNet	cut-21.1-1	-	Patient	-	-
WordNet	-	cut.v.01	food.n.01, hair.n.01	-	-
L-GAT	Drop	Predicate	Theme	Null	Null
FrameNet	Abandonment	-	Theme	-	-
VerbNet	leave-51.2	-	Theme	-	-
WordNet	-	drop.v.01	physical entity.n.01	-	-
L-GAT	Eat	Predicate	Patient	Null	Null
FrameNet	Ingestion	-	Ingestibles	-	-
VerbNet	eat-39.1-1	-	Patient	-	-
WordNet	-	eat.v.01	food.n.01	-	-
L-GAT	Fill	Predicate	Destination	Preposition	Theme
FrameNet	Filling	-	Goal	-	Theme
VerbNet	fill-9.8-1	-	Destination	-	Theme
WordNet	-	fill.v.01	container.n.01	with	fluid.n.01
L-GAT	Get	Predicate	Theme	Null	Null
FrameNet	Taking	-	Theme	-	-
VerbNet	get-13.5.1	-	Theme	-	-
WordNet	-	get.v.01, take.v.01	physical entity.n.01	-	-

Table 6: Extended examples of abstractive action templates in L-GAT.

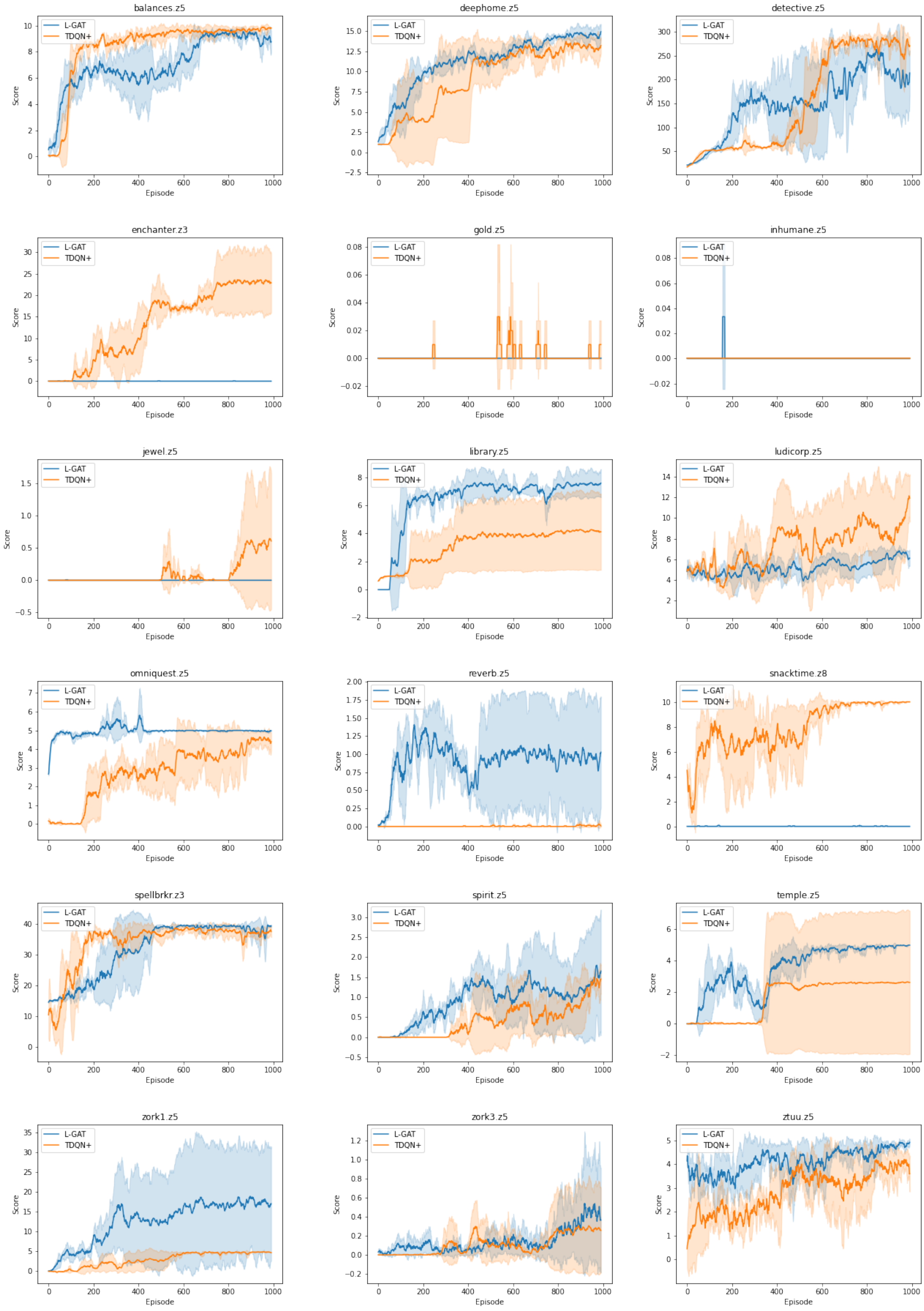

Figure 6: Learning curves of L-GAT (blue) and TDQN+ (orange).

Observation	Action	Score
West of House You are standing in an open field west of a white house, with a boarded front door. There is a small mailbox ...	examine mailbox	0
The small mailbox is closed.	go north	0
North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded ...	go east	0
Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there ...	go east	0
Clearing You are in a small clearing in a well marked forest path that extends to the east and west.	go west	0
Behind House	go east	0
Clearing	go east	0
Canyon View You are at the top of the Great Canyon on its west wall. From here there is a marvelous view of the ...	go west	0
Forest This is a dimly lit forest, with large trees all around.	go west	0
Forest This is a forest, with trees in all directions. To the east, there appears to be sunlight.	go east	0
Forest Path This is a path winding through a dimly lit forest. The path heads north-south here. One particularly large tree ...	go up	0
Up a Tree You ... Beside you on the branch is a small bird's nest. In the bird's nest is a large egg encrusted with precious ...	eat egg	5
(Taken) I don't think that the jewel-encrusted egg would agree with you.	drop all	5
jewel-encrusted egg: The egg falls to the ground and springs open, seriously damaged. There is a golden clockwork canary ...	get bird	5
The songbird is not here but is probably nearby. You hear in the distance the chirping of a song bird.	get all	5
bird's nest: Taken. You hear in the distance the chirping of a song bird.	drop all	5
bird's nest: The bird's nest falls to the ground.	go up	5
You cannot climb any higher.	go down	5
Forest Path There is a bird's nest here. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in ...	go south	5
North of House	go east	5
Behind House	open window	5
With great effort, you open the window far enough to allow entry.	open white	15
Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food ...	go west	15
Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the ...	get all	15
trophy case: The trophy case is securely fastened to the wall. sword: Taken. brass lantern: Taken. carpet: The rug is ...	drop all	15
brass lantern: Dropped. sword: Dropped.	ignite lantern	15
What do you want to ignite the lantern with?	push lantern	15
Pushing the brass lantern doesn't seem to work.	light lantern	15
(Taken) The brass lantern is now on.	drop all	15
brass lantern: Dropped.	go east	15
Kitchen On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle ...	go west	15
Living Room There is a brass lantern (battery-powered) here. There is a sword here.	get all	15
brass lantern: Taken. sword: Taken. trophy case: The trophy case is securely fastened to the wall. carpet: The rug is ...	drop lantern	15
Dropped.	drop all	15
sword: Dropped.	go east	15
Kitchen On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle ...	open sack	15
Opening the brown sack reveals a lunch, and a clove of garlic.	take all	15
brown sack: Taken. glass bottle: Taken.	go west	15
Living Room There is a sword here. There is a brass lantern (battery-powered) here.	drop all	15
glass bottle: Dropped. brown sack: Dropped.	move rug	15
With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.	open cover	15
The door reluctantly opens to reveal a rickety staircase descending into darkness.	ask case	15
You can't talk to the trophy case!	wait	15
Time passes...	get all	15
brown sack: Taken. glass bottle: Taken. sword: Taken. brass lantern: Taken. trophy case: The trophy case is securely ...	go down	15
The trap door crashes shut, and you hear someone barring it. Cellar You are in a dark and damp cellar with a narrow ...	-	40

Table 7: Best play trajectory of L-GAT agent in zork1 until reaching best score of 40. For simplicity, we excluded steps with no-effect actions in this table.

MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers

Wenhui Wang **Hangbo Bao** **Shaohan Huang** **Li Dong** **Furu Wei**[*]
Microsoft Research
`{wenwan,t-habao,shaohanh,lidong1,fuwei}@microsoft.com`

Abstract

We generalize deep self-attention distillation in MiniLM (Wang et al., 2020) by only using self-attention relation distillation for task-agnostic compression of pretrained Transformers. In particular, we define multi-head self-attention relations as scaled dot-product between the pairs of *query*, *key*, and *value* vectors within each self-attention module. Then we employ the above relational knowledge to train the student model. Besides its simplicity and unified principle, more favorably, there is no restriction in terms of the number of student's attention heads, while most previous work has to guarantee the same head number between teacher and student. Moreover, the fine-grained self-attention relations tend to fully exploit the interaction knowledge learned by Transformer. In addition, we thoroughly examine the layer selection strategy for teacher models, rather than just relying on the last layer as in MiniLM. We conduct extensive experiments on compressing both monolingual and multilingual pre-trained models. Experimental results demonstrate that our models[1] distilled from base-size and large-size teachers (BERT, RoBERTa and XLM-R) outperform the state-of-the-art.

1 Introduction

Pretrained Transformers (Radford et al., 2018; Devlin et al., 2018; Dong et al., 2019; Yang et al., 2019; Joshi et al., 2019; Liu et al., 2019; Bao et al., 2020; Radford et al., 2019; Raffel et al., 2019; Lewis et al., 2019a) have been highly successful for a wide range of natural language processing tasks. However, these models usually consist of hundreds of millions of parameters and are getting bigger. It brings challenges for fine-tuning and online serving in real-life applications due to the restrictions of computation resources and latency.

Knowledge distillation (KD; Hinton et al. 2015, Romero et al. 2015) has been widely employed to compress pretrained Transformers, which transfers knowledge of the large model (teacher) to the small model (student) by minimizing the differences between teacher and student features. Soft target probabilities (soft labels) and intermediate representations are usually utilized to perform KD training. In this work, we focus on task-agnostic compression of pretrained Transformers (Sanh et al., 2019; Tsai et al., 2019; Jiao et al., 2019; Sun et al., 2019b; Wang et al., 2020). The student models are distilled from large pretrained Transformers using large-scale text corpora. The distilled task-agnostic model can be directly fine-tuned on downstream tasks, and can be utilized to initialize task-specific distillation.

DistilBERT (Sanh et al., 2019) uses soft target probabilities for masked language modeling predictions and embedding outputs to train the student. The student model is initialized from the teacher by taking one layer out of two. TinyBERT (Jiao et al., 2019) utilizes hidden states and self-attention distributions (i.e., attention maps and weights), and adopts a uniform function to map student and teacher layers for layer-wise distillation. MobileBERT (Sun et al., 2019b) introduces specially designed teacher and student models using inverted-bottleneck and bottleneck structures to keep their layer number and hidden size the same, layer-wisely transferring hidden states and self-attention distributions. MiniLM (Wang et al., 2020) proposes deep self-attention distillation, which uses self-attention distributions and value relations to help the student to deeply mimic teacher's self-attention modules. MiniLM shows that transferring knowledge of teacher's last layer achieves better performance than layer-wise distil-

---

[*] Contact person.

[1] Distilled models and code will be publicly available at `https://aka.ms/minilm`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2140–2151
August 1–6, 2021. ©2021 Association for Computational Linguistics

lation. In summary, most previous work relies on self-attention distributions to perform KD training, which leads to a restriction that the number of attention heads of student model has to be the same as its teacher.

In this work, we generalize and simplify deep self-attention distillation of MiniLM (Wang et al., 2020) by using self-attention relation distillation. We introduce multi-head self-attention relations computed by scaled dot-product of pairs of queries, keys and values, which guides the student training. Taking query vectors as an example, in order to obtain queries of multiple relation heads, we first concatenate query vectors of different attention heads, and then split the concatenated vector according to the desired number of relation heads. Afterwards, for teacher and student models with different attention head numbers, we can align their queries with the same number of relation heads for distillation. Moreover, using a larger number of relation heads brings more fine-grained self-attention knowledge, which helps the student to achieves a deeper mimicry of teacher's self-attention module. In addition, for large-size (24 layers, 1024 hidden size) teachers, extensive experiments indicate that transferring an upper middle layer tends to perform better than using the last layer as in MiniLM.

Experimental results show that our monolingual models distilled from BERT and RoBERTa, and multilingual models distilled from XLM-R outperform state-of-the-art models in different parameter sizes. The 6×768 (6 layers, 768 hidden size) model distilled from $\text{BERT}_{\text{LARGE}}$ is 2.0× faster, meanwhile, performing better than $\text{BERT}_{\text{BASE}}$. The base-size model distilled from $\text{RoBERTa}_{\text{LARGE}}$ outperforms $\text{RoBERTa}_{\text{BASE}}$ using much fewer training examples.

To summarize, our contributions include:

- We generalize and simplify deep self-attention distillation in MiniLM by introducing multi-head self-attention relation distillation, which brings more fine-grained self-attention knowledge and allows more flexibility for the number of student's attention heads.
- We conduct extensive distillation experiments on different large-size teachers and find that using knowledge of a teacher's upper middle layer achieves better performance.
- Experimental results demonstrate the effectiveness of our method for different monolingual and multilingual teachers in base-size and large-size.

2 Related Work

2.1 Backbone Network: Transformer

Multi-layer Transformer (Vaswani et al., 2017) has been widely adopted in pretrained models. Each Transformer layer consists of a self-attention sub-layer and a position-wise fully connected feed-forward sub-layer.

Self-Attention Transformer relies on multi-head self-attention to capture dependencies between words. Given previous Transformer layer's output $\mathbf{H}^{l-1} \in \mathbb{R}^{|x| \times d_h}$, the output of a self-attention head $\mathbf{O}_{l,a},\ a \in [1, A_h]$ is computed via:

$$\mathbf{Q}_{l,a} = \mathbf{H}^{l-1}\mathbf{W}^{Q}_{l,a} \quad (1)$$

$$\mathbf{K}_{l,a} = \mathbf{H}^{l-1}\mathbf{W}^{K}_{l,a} \quad (2)$$

$$\mathbf{V}_{l,a} = \mathbf{H}^{l-1}\mathbf{W}^{V}_{l,a} \quad (3)$$

$$\mathbf{O}_{l,a} = \text{softmax}(\frac{\mathbf{Q}_{l,a}\mathbf{K}^{\intercal}_{l,a}}{\sqrt{d_k}})\mathbf{V}_{l,a} \quad (4)$$

Previous layer's output is linearly projected to queries, keys and values using parameter matrices $\mathbf{W}^{Q}_{l,a}, \mathbf{W}^{K}_{l,a}, \mathbf{W}^{V}_{l,a} \in \mathbb{R}^{d_h \times d_k}$, respectively. The self-attention distributions are computed via scaled dot-product of queries and keys. These weights are assigned to the corresponding value vectors to obtain the attention output. $|x|$ represents the length of input sequence. A_h and d_h indicate the number of attention heads and hidden size. d_k is the attention head size. $d_k \times A_h$ is usually equal to d_h.

2.2 Pretrained Language Models

Pre-training has led to strong improvements across a variety of natural language processing tasks. Pretrained language models are learned on large amounts of text data, and then fine-tuned to adapt to specific tasks. BERT (Devlin et al., 2018) proposes to pretrain a deep bidirectional Transformer using masked language modeling (MLM) objective. UniLM (Dong et al., 2019) is jointly pretrained on three types language modeling objectives to adapt to both understanding and generation tasks. RoBERTa (Liu et al., 2019) achieves strong performance by training longer steps using large batch size and more text data. MASS (Song et al., 2019), T5 (Raffel et al., 2019) and BART (Lewis et al.,

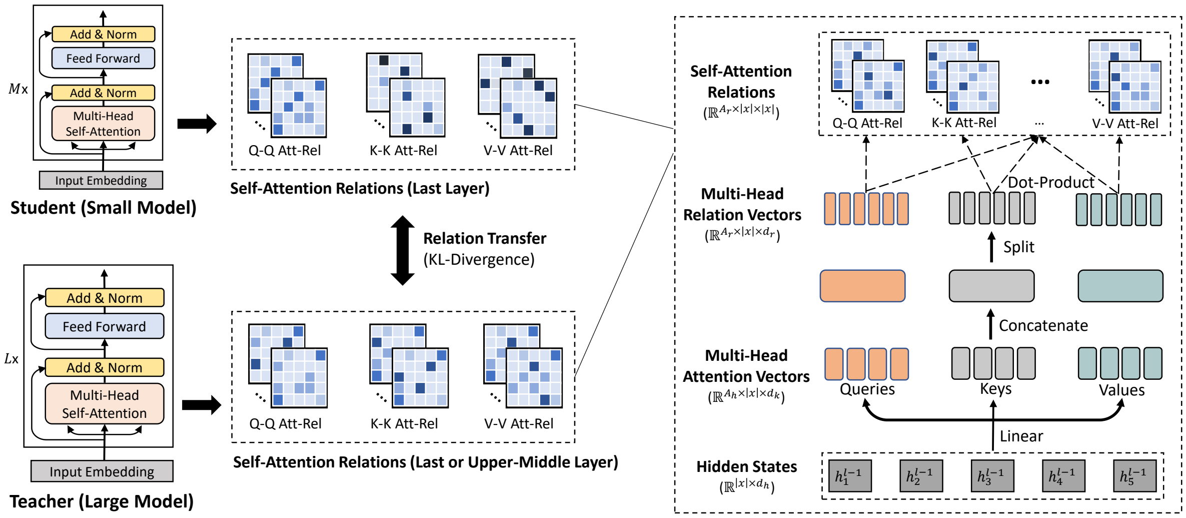

Figure 1: Overview of multi-head self-attention relation distillation. We introduce multi-head self-attention relations computed by scaled dot-product of pairs of queries, keys and values to guide the training of students. In order to obtain vectors (queries, keys and values) of multiple relation heads, we first concatenate self-attention vectors of different attention heads and then split them according to the desired number of relation heads. We choose to transfer Q-Q, K-K and V-V self-attention relations to achieve a balance between performance and training speed. For large-size teacher, we transfer the self-attention knowledge of an upper middle layer of the teacher. For base-size teacher, using the last layer achieves better performance.

2019a) employ a standard encoder-decoder structure and pretrain the decoder auto-regressively. Besides monolingual pretrained models, multilingual pretrained models (Devlin et al., 2018; Lample and Conneau, 2019; Chi et al., 2019; Conneau et al., 2019; Chi et al., 2020) also advance the state-of-the-art on cross-lingual understanding and generation.

2.3 Knowledge Distillation

Knowledge distillation has been proven to be a promising way to compress large models while maintaining accuracy. Knowledge of a single or an ensemble of large models is used to guide the training of small models. Hinton et al. (2015) propose to use soft target probabilities to train student models. More fine-grained knowledge such as hidden states (Romero et al., 2015) and attention distributions (Zagoruyko and Komodakis, 2017; Hu et al., 2018) are introduced to improve the student model.

In this work, we focus on task-agnostic knowledge distillation of pretrained Transformers. The distilled task-agnostic model can be fine-tuned to adapt to downstream tasks. It can also be utilized to initialize task-specific distillation (Sun et al., 2019a; Turc et al., 2019; Aguilar et al., 2019; Mukherjee and Awadallah, 2020; Xu et al., 2020; Hou et al., 2020; Li et al., 2020), which uses a fine-tuned teacher model to guide the training of the student on specific tasks. Knowledge used for distillation and layer mapping function are two key points for task-agnostic distillation of pretrained Transformers. Most previous work uses soft target probabilities, hidden states, self-attention distributions and value-relation to train the student model. For the layer mapping function, TinyBERT (Jiao et al., 2019) uses a uniform strategy to map teacher and student layers. MobileBERT (Sun et al., 2019b) assumes the student has the same number of layers as its teacher to perform layer-wise distillation. MINILM (Wang et al., 2020) transfers self-attention knowledge of teacher's last layer to the student last Transformer layer. Different from previous work, our method uses multi-head self-attention relations to eliminate the restriction on the number of student's attention heads. Moreover, we show that transferring the self-attention knowledge of an upper middle layer of the large-size teacher model is more effective.

3 Multi-Head Self-Attention Relation Distillation

Following MINILM (Wang et al., 2020), the key idea of our approach is to deeply mimic teacher's self-attention module, which draws dependencies between words and is the vital component of Transformer. MINILM uses teacher's self-attention distributions to train the student model. It brings

Model	**Teacher**	**#Param**	**Speedup**	**SQuAD2**	**MNLI-m**	**QNLI**	**QQP**	**RTE**	**SST**	**MRPC**	**CoLA**	**Avg**
BERT$_{\text{BASE}}$	-	109M	×1.0	76.8	84.5	91.7	91.3	68.6	93.2	87.3	58.9	81.5
RoBERTa$_{\text{BASE}}$	-	125M	×1.0	83.7	87.6	92.8	91.9	78.7	94.8	90.2	63.6	85.4
BERT$_{\text{SMALL}}$	-	66M	×2.0	73.2	81.8	89.8	90.6	67.9	91.2	84.9	53.5	79.1
Truncated BERT$_{\text{BASE}}$	-	66M	×2.0	69.9	81.2	87.9	90.4	65.5	90.8	82.7	41.4	76.2
Truncated RoBERTa$_{\text{BASE}}$	-	81M	×2.0	77.9	84.9	91.1	91.3	67.9	92.9	87.5	55.2	81.1
DistilBERT	BERT$_{\text{BASE}}$	66M	×2.0	70.7	82.2	89.2	88.5	59.9	91.3	87.5	51.3	77.6
TinyBERT	BERT$_{\text{BASE}}$	66M	×2.0	73.1	83.5	90.5	90.6	72.2	91.6	88.4	42.8	79.1
MiniLM	BERT$_{\text{BASE}}$	66M	×2.0	76.4	84.0	91.0	91.0	71.5	92.0	88.4	49.2	80.4
6×384 Ours	BERT$_{\text{BASE}}$	22M	×5.3	72.9	82.8	90.3	90.6	68.9	91.3	86.6	41.8	78.2
6×384 Ours	BERT$_{\text{LARGE}}$	22M	×5.3	74.3	83.0	90.4	90.7	68.5	91.1	87.8	41.6	78.4
6×384 Ours	RoBERTa$_{\text{LARGE}}$	30M	×5.3	**76.4**	**84.4**	**90.9**	**90.8**	**69.9**	**92.0**	**88.7**	**42.6**	**79.5**
6×768 Ours	BERT$_{\text{BASE}}$	66M	×2.0	76.3	84.2	90.8	91.1	72.1	92.4	88.9	52.5	81.0
6×768 Ours	BERT$_{\text{LARGE}}$	66M	×2.0	77.7	85.0	91.4	91.1	73.0	92.5	88.9	53.9	81.7
6×768 Ours	RoBERTa$_{\text{LARGE}}$	81M	×2.0	**81.6**	**87.0**	**92.7**	**91.4**	**78.7**	**94.5**	**90.4**	**54.0**	**83.8**

Table 1: Results of our students distilled from base-size and large-size teachers on the development sets of GLUE and SQuAD 2.0. We report F1 for SQuAD 2.0, Matthews correlation coefficient for CoLA, and accuracy for other datasets. The GLUE results of DistilBERT are taken from Sanh et al. (2019). The rest results of DistilBERT, TinyBERT[2], BERT$_{\text{SMALL}}$, Truncated BERT$_{\text{BASE}}$ and MiniLM are taken from Wang et al. (2020). BERT$_{\text{SMALL}}$ (Turc et al., 2019) is trained using the MLM objective, without using KD. We also report the results of truncated BERT$_{\text{BASE}}$ and truncated RoBERTa$_{\text{BASE}}$, which drops the top 6 layers of the base model. Top-layer dropping has been proven to be a strong baseline (Sajjad et al., 2020). The fine-tuning results are an average of 4 runs.

the restriction on the number of attention heads of students, which is required to be the same as its teacher. To introduce more fine-grained self-attention knowledge and avoid using teacher's self-attention distributions, we generalize deep self-attention distillation in MiniLM and introduce multi-head self-attention relations of pairs of queries, keys and values to train the student. Besides, we conduct extensive experiments and find that layer selection of the teacher model is critical for distilling large-size models. Figure 1 gives an overview of our method.

3.1 Multi-Head Self-Attention Relations

Multi-head self-attention relations are obtained by scaled dot-product of pairs[3] of queries, keys and values of multiple relation heads. Taking query vectors as an example, in order to obtain queries of multiple relation heads, we first concatenate queries of different attention heads and then split the concatenated vector based on the desired number of relation heads. The same operation is also performed on keys and values. For teacher and student models which uses different number of attention heads, we convert their queries, keys and values of different number of attention heads into vectors of the same number of relation heads to perform KD training. Our method eliminates the restriction on the number of attention heads of student models. Moreover, using more relation heads in computing self-attention relations brings more fine-grained self-attention knowledge and improves the performance of the student model.

We use $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ to denote the queries, keys and values of multiple relation heads. The KL-divergence between multi-head self-attention relations of the teacher and student is used as the training objective:

$$\mathcal{L} = \sum_{i=1}^{3}\sum_{j=1}^{3} \alpha_{ij}\mathcal{L}_{ij} \tag{5}$$

$$\mathcal{L}_{ij} = \frac{1}{A_r|x|}\sum_{a=1}^{A_r}\sum_{t=1}^{|x|} D_{KL}(\mathbf{R}^T_{ij,l,a,t} \parallel \mathbf{R}^S_{ij,m,a,t}) \tag{6}$$

$$\mathbf{R}^T_{ij,l,a} = \text{softmax}(\frac{\mathbf{A}^T_{i,l,a}\mathbf{A}^{T\intercal}_{j,l,a}}{\sqrt{d_r}}) \tag{7}$$

$$\mathbf{R}^S_{ij,m,a} = \text{softmax}(\frac{\mathbf{A}^S_{i,m,a}\mathbf{A}^{S\intercal}_{j,m,a}}{\sqrt{d'_r}}) \tag{8}$$

where $\mathbf{A}^T_{i,l,a} \in \mathbb{R}^{|x|\times d_r}$ and $\mathbf{A}^S_{i,m,a} \in \mathbb{R}^{|x|\times d'_r}$ $(i \in [1,3])$ are the queries, keys and values of a relation head of l-th teacher layer and m-th student layer. d_r and d'_r are the relation head size of teacher and student models. $\mathbf{R}^T_{ij,l} \in \mathbb{R}^{A_r\times|x|\times|x|}$ is the self-attention relation of $\mathbf{A}^T_{i,l}$ and $\mathbf{A}^T_{j,l}$ of teacher model.

---

[2] In addition to task-agnostic distillation, TinyBERT uses task-specific distillation and data augmentation to further improve the model. We report the fine-tuning results of their public task-agnostic model.

[3] There are nine types of self-attention relations, such as query-query, key-key, key-value and query-value relations.

Model	Teacher	#Param	Speedup	SQuAD2	MNLI-m/mm	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	Avg
BERT$_{\text{BASE}}$	-	109M	1.0×	76.8	84.6/83.4	90.5	71.2	66.4	93.5	88.9	52.1	85.8	79.3
BERT$_{\text{LARGE}}$	-	340M	0.3×	81.9	86.7/85.9	92.7	72.1	70.1	94.9	89.3	60.5	86.5	82.1
6×768 Ours	BERT$_{\text{BASE}}$	66M	2.0×	76.3	83.8/83.3	90.2	70.9	**69.2**	92.9	**89.1**	46.6	84.3	78.7
6×768 Ours	BERT$_{\text{LARGE}}$	66M	2.0×	**77.7**	**84.5/84.0**	**91.5**	**71.3**	**69.2**	**93.0**	**89.1**	**48.6**	**85.1**	**79.4**

Table 2: Results of our 6×768 students distilled form BERT on GLUE test sets and SQuAD 2.0 dev set. The reported results are directly fine-tuned on downstream tasks. We report F1 for SQuAD 2.0, QQP and MRPC, Spearman correlation for STS-B, Matthews correlation coefficient for CoLA and accuracy for the rest.

$\mathbf{R}^{T}_{ij,l,a} \in \mathbb{R}^{|x|\times|x|}$ is the self-attention relation of a teacher's relation head. $\mathbf{R}^{S}_{ij,m} \in \mathbb{R}^{A_r\times|x|\times|x|}$ is the self-attention relation of student model. For example, $\mathbf{R}^{T}_{11,l}$ represents teacher's Q-Q attention relation in Figure 1. A_r is the number of relation heads. If the number of relation heads and attention heads is the same, the Q-K relation is equivalent to the attention weights in self-attention module. $\alpha_{ij} \in \{0, 1\}$ is the weight assigned to each self-attention relation loss. We transfer query-query, key-key and value-value self-attention relations to balance the performance and training cost.

3.2 Layer Selection of Teacher Model

Besides the knowledge used for distillation, mapping function between teacher and student layers is another key factor. As in MiniLM, we only transfer the self-attention knowledge of one of the teacher layers to the student last layer. Only distilling one layer of the teacher model is fast and effective. Different from previous work which usually conducts experiments on base-size teachers, we experiment with different large-size teachers and find that transferring self-attention knowledge of an upper middle layer performs better than using other layers. For BERT$_{\text{LARGE}}$ and BERT$_{\text{LARGE-WWM}}$, transferring the 21-th (start at one) layer achieves the best performance. For RoBERTa$_{\text{LARGE}}$ and XLM-R$_{\text{LARGE}}$, using the self-attention knowledge of 19-th layer achieves better performance. For the base-size teacher, we also find that using teacher's last layer performs better.

4 Experiments

We conduct distillation experiments on different teacher models including BERT$_{\text{BASE}}$, BERT$_{\text{LARGE}}$, BERT$_{\text{LARGE-WWM}}$, RoBERTa$_{\text{BASE}}$, RoBERTa$_{\text{LARGE}}$, XLM-R$_{\text{BASE}}$ and XLM-R$_{\text{LARGE}}$.

4.1 Setup

We use the uncased version for three BERT teacher models. For the pre-training data, we use English Wikipedia and BookCorpus (Zhu et al., 2015). We train student models using 256 as the batch size and 6e-4 as the peak learning rate for $400,000$ steps. We use linear warmup over the first $4,000$ steps and linear decay. We use Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$. The maximum sequence length is set to 512. The dropout rate and weight decay are 0.1 and 0.01. The number of attention heads is 12 for all student models. The number of relation heads is 48 and 64 for base-size and large-size teacher model, respectively. The student models are initialized randomly.

For models distilled from RoBERTa, we use similar pre-training datasets as in Liu et al. (2019). For the 12×768 student model, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.98$. The rest hyper-parameters are the same as models distilled from BERT.

For multilingual student models distilled from XLM-R, we perform training using the same datasets as in Conneau et al. (2019) for $1,000,000$ steps. We conduct distillation experiments using 8 V100 GPUs with mixed precision training.

4.2 Downstream Tasks

Following previous pre-training (Devlin et al., 2018; Liu et al., 2019; Conneau et al., 2019) and task-agnostic distillation (Sun et al., 2019b; Jiao et al., 2019) work, we evaluate the English student models on GLUE benchmark and extractive question answering. The multilingual models are evaluated on cross-lingual natural language inference and cross-lingual question answering.

GLUE General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) consists of two single-sentence classification tasks (SST-2 (Socher et al., 2013) and CoLA (Warstadt et al., 2018)), three similarity and paraphrase tasks (MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017) and QQP), and four inference tasks (MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli

Model	Teacher	#Param	SQuAD2	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	Avg
BERT$_{\text{BASE}}$	-	109M	76.8	84.5	91.7	91.3	68.6	93.2	87.3	58.9	89.5	82.4
RoBERTa$_{\text{BASE}}$	-	125M	83.7	87.6	92.8	**91.9**	78.7	94.8	90.2	63.6	91.2	86.1
12×768 Ours	BERT$_{\text{LARGE}}$	109M	81.8	86.5	92.6	91.6	76.4	93.3	89.2	62.3	90.5	84.9
12×768 Ours	RoBERTa$_{\text{LARGE}}$	125M	**86.6**	**89.4**	**94.0**	91.8	**83.1**	**95.9**	**91.2**	**65.0**	**91.3**	**87.6**

Table 3: Results of our 12×768 models on the dev sets of GLUE benchmark and SQuAD 2.0. The fine-tuning results are an average of 4 runs for each task. We report F1 for SQuAD 2.0, Pearson correlation for STS-B, Matthews correlation coefficient for CoLA and accuracy for the rest.

Model	SQuAD2	MNLI-m	SST-2	Avg
MiniLM (Last Layer)	79.1	84.7	91.2	85.0
+ Upper Middle Layer	80.3	85.2	91.5	85.7
12×384 Ours	**80.7**	**85.7**	**92.3**	**86.2**

Table 4: Comparison of different methods using BERT$_{\text{LARGE-WWM}}$ as the teacher. We report dev results of 12×384 student model with 128 embedding size.

et al., 2009) and WNLI (Levesque et al., 2012)).

Extractive Question Answering The task aims to predict a continuous sub-span of the passage to answer the question. We evaluate on SQuAD 2.0 (Rajpurkar et al., 2018), which has been served as a major question answering benchmark.

Cross-lingual Natural Language Inference (XNLI) XNLI (Conneau et al., 2018) is a cross-lingual classification benchmark. It aims to identity the semantic relationship between two sentences and provides instances in 15 languages.

Cross-lingual Question Answering We use MLQA (Lewis et al., 2019b) to evaluate multilingual models. MLQA extends English SQuAD dataset (Rajpurkar et al., 2016) to seven languages.

4.3 Main Results

Table 1 presents the dev results of 6×384 and 6×768 models distilled from BERT$_{\text{BASE}}$, BERT$_{\text{LARGE}}$ and RoBERTa$_{\text{LARGE}}$ on GLUE and SQuAD 2.0. (1) Previous methods (Sanh et al., 2019; Jiao et al., 2019; Sun et al., 2019a; Wang et al., 2020) usually distill BERT$_{\text{BASE}}$ into a 6-layer model with 768 hidden size. We first report results of the same setting. Our 6×768 model outperforms DistilBERT, TinyBERT, MiniLM and two BERT baselines across most tasks. Moreover, our method allows more flexibility for the number of attention heads of student models. (2) Both 6×384 and 6×768 models distilled from BERT$_{\text{LARGE}}$ outperform models distilled from BERT$_{\text{BASE}}$. The 6×768 model distilled from BERT$_{\text{LARGE}}$ is 2.0× faster than BERT$_{\text{BASE}}$, while achieving better performance. (3) Student models distilled from RoBERTa$_{\text{LARGE}}$ achieve further improvements. Better teacher results in better students. Multi-head self-attention relation distillation is effective for different large-size pretrained Transformers.

We report the results of 6×768 students distilled from BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$ on GLUE test sets and SQuAD 2.0 dev set in Table 2. 6×768 model distilled from BERT$_{\text{BASE}}$ retains more than 99% accuracy of its teacher while using 50% Transformer parameters. 6×768 model distilled from BERT$_{\text{LARGE}}$ compares favorably with BERT$_{\text{BASE}}$.

We compress RoBERTa$_{\text{LARGE}}$ and BERT$_{\text{LARGE}}$ into a base-size student model. Dev results of GLUE benchmark and SQuAD 2.0 are shown in Table 3. Our base-size models distilled from large-size teacher outperforms BERT$_{\text{BASE}}$ and RoBERTa$_{\text{BASE}}$. Our method can be adopted to train students in different parameter size. Moreover, our student distilled from RoBERTa$_{\text{LARGE}}$ uses a much smaller (almost 32× smaller) training batch size and fewer training steps than RoBERTa$_{\text{BASE}}$. Our method requires much fewer training examples.

Most of previous work conducts experiments using base-size teachers. To compare with previous methods on large-size teacher, we reimplement MiniLM and compress BERT$_{\text{LARGE-WWM}}$ into a 12×384 student model. Dev results of SQuAD 2.0, MNLI-m and SST-2 are presented in Table 4. Our method also outperforms MiniLM for large-size teachers. Moreover, we report results of distilling an upper middle layer instead of the last layer for MiniLM. Layer selection is also effective for MiniLM when distilling large-size teachers.

Table 5 and Table 6 show the results of our student models distilled from XLM-R on XNLI and MLQA. For XNLI, the best single model is selected on the joint dev set of all the languages as in Conneau et al. (2019). Following Lewis et al. (2019b), we adopt SQuAD 1.1 as training data and evaluate on MLQA English development set for

Model	Teacher	#L	#H	#Param	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur	Avg
mBERT	-	12	768	170M	82.1	73.8	74.3	71.1	66.4	68.9	69.0	61.6	64.9	69.5	55.8	69.3	60.0	50.4	58.0	66.3
XLM-100	-	16	1280	570M	83.2	76.7	77.7	74.0	72.7	74.1	72.7	68.7	68.6	72.9	68.9	72.5	65.6	58.2	62.4	70.7
XLM-R$_{\text{BASE}}$	-	12	768	277M	85.8	79.7	80.7	78.7	77.5	79.6	78.1	74.2	73.8	76.5	74.6	76.7	72.4	66.5	68.3	76.2
MiniLM	XLM-R$_{\text{BASE}}$	6	384	107M	79.2	72.3	73.1	70.3	69.1	72.0	69.1	64.5	64.9	69.0	66.0	67.8	62.9	59.0	60.6	68.0
6×384 Ours	XLM-R$_{\text{BASE}}$	6	384	107M	78.1	71.4	72.7	69.2	70.8	72.2	69.9	67.5	66.1	68.9	67.5	68.2	64.7	62.7	62.3	68.8
6×384 Ours	XLM-R$_{\text{LARGE}}$	6	384	107M	79.8	72.5	73.7	69.3	70.6	72.3	69.5	66.9	67.5	69.1	67.0	68.7	64.8	62.4	63.0	**69.1**

Table 5: Cross-lingual classification results of our 6×384 multilingual models on XNLI. We report the accuracy on each of the 15 XNLI languages and the average accuracy. #L and #H indicate the number of layers and hidden size.

Model	Teacher	#L	#H	#Param	en	es	de	ar	hi	vi	zh	Avg
mBERT	-	12	768	170M	77.7 / 65.2	64.3 / 46.6	57.9 / 44.3	45.7 / 29.8	43.8 / 29.7	57.1 / 38.6	57.5 / 37.3	57.7 / 41.6
XLM-15	-	12	1024	248M	74.9 / 62.4	68.0 / 49.8	62.2 / 47.6	54.8 / 36.3	48.8 / 27.3	61.4 / 41.8	61.1 / 39.6	61.6 / 43.5
XLM-R$_{\text{BASE}}$	-	12	768	277M	77.1 / 64.6	67.4 / 49.6	60.9 / 46.7	54.9 / 36.6	59.4 / 42.9	64.5 / 44.7	61.8 / 39.3	63.7 / 46.3
MiniLM	XLM-R$_{\text{BASE}}$	6	384	107M	75.5 / 61.9	55.6 / 38.2	53.3 / 37.7	43.5 / 26.2	46.9 / 31.5	52.0 / 33.1	48.8 / 27.3	53.7 / 36.6
6×384 Ours	XLM-R$_{\text{BASE}}$	6	384	107M	76.0 / 62.5	60.5 / 42.4	57.7 / 43.1	48.6 / 30.1	53.3 / 36.5	55.5 / 35.6	54.6 / 32.5	58.0 / 40.4
6×384 Ours	XLM-R$_{\text{LARGE}}$	6	384	107M	76.2 / 62.9	59.2 / 41.7	57.4 / 42.2	47.3 / 29.4	54.1 / 36.9	58.2 / 37.9	57.0 / 34.0	**58.5 / 40.7**

Table 6: Cross-lingual question answering results of our 6×384 multilingual models on MLQA. We report the F1 and EM (exact match) scores on each of the 7 MLQA languages. #L and #H indicate the number of layers and hidden size.

Model	SQuAD2	MNLI-m	SST-2	Avg
Ours (Q-Q + K-K + V-V)	**72.8**	**82.2**	**91.5**	**82.2**
– Q-Q Att-Rel	71.6	81.9	90.6	81.4
– K-K Att-Rel	71.9	81.9	90.5	81.4
– V-V Att-Rel	71.5	81.6	90.5	81.2
Q-K + V-V Att-Rels	72.4	**82.2**	91.0	81.9

Table 7: Ablation studies of different self-attention relations. We report results of 6×384 student model distilled from BERT$_{\text{BASE}}$. The relation head number is 12.

#Relation Heads	6	12	24	**48**
6×384 *model distilled from RoBERTa*$_{\text{BASE}}$				
MNLI-m	82.8	82.9	83.0	**83.4**
SQuAD 2.0	74.5	75.0	74.9	**75.7**
6×384 *model distilled from BERT*$_{\textit{BASE}}$				
MNLI-m	81.9	82.2	82.2	**82.4**
SQuAD 2.0	71.9	72.8	72.7	**73.0**

Table 8: Results of 6×384 model using different number of relation heads.

early stopping. Our 6×384 model outperforms mBERT (Devlin et al., 2018) with 5.3× speedup. Our method also performs better than MiniLM, which further validates the effectiveness of multi-head self-attention relation distillation. Transferring multi-head self-attention relations can bring more fine-grained self-attention knowledge.

4.4 Ablation Studies

Effect of using different self-attention relations We perform ablation studies to analyse the contribution of different self-attention relations. Dev results of three tasks are illustrated in Table 7. Q-Q, K-K and V-V self-attention relations positively contribute to the final results. Besides, we also compare Q-Q + K-K + V-V with Q-K + V-V given queries and keys are employed to compute self-attention distributions in self-attention module. Experimental result shows that using Q-Q + K-K + V-V achieves better performance.

Effect of distilling different teacher layers Figure 2 presents the results of 6×384 model distilled from different layers of BERT$_{\text{BASE}}$, BERT$_{\text{LARGE}}$ and XLM-R$_{\text{LARGE}}$. For BERT$_{\text{BASE}}$, using the last layer achieves better performance than other layers. For BERT$_{\text{LARGE}}$ and XLM-R$_{\text{LARGE}}$, we find that using one of the upper middle layers achieves the best performance. The same trend is also observed for BERT$_{\text{LARGE-WWM}}$ and RoBERTa$_{\text{LARGE}}$.

Effect of different number of relation heads Table 8 shows the results of 6×384 model distilled from BERT$_{\text{BASE}}$ and RoBERTa$_{\text{BASE}}$ using different number of relation heads. Using a larger number of relation heads achieves better performance. More fine-grained self-attention knowledge can be captured by using more relation heads, which helps the student to deeply mimic the self-attention module of its teacher. Besides, we find that the number of relation heads is not required to be a positive multiple of both the number of student and teacher attention heads. The relation head can be a fragment of a single attention head or contains fragments from

Model	Teacher	#Param	Speedup	SQuAD2	MNLI-m/mm	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	Avg
$\text{BERT}_{\text{BASE}}$	-	109M	1.0×	76.8	84.6/83.4	90.5	71.2	66.4	93.5	88.9	52.1	85.8	79.3
MobileBERT	IB-$\text{BERT}_{\text{LARGE}}$	25M	1.8×	80.2	84.3/83.4	91.6	70.5	70.4	92.6	88.8	**51.1**	84.8	79.8
12×384 Ours	$\text{BERT}_{\text{LARGE-WWM}}$	25M	2.7×	80.7	**85.9**/84.6	91.9	71.4	71.9	93.3	89.2	44.9	85.5	79.9
+ More Att-Rels	$\text{BERT}_{\text{LARGE-WWM}}$	25M	2.7×	**80.9**	85.8/**84.8**	**92.3**	**71.6**	**72.0**	**93.6**	**89.7**	46.6	**86.0**	**80.3**

Table 9: Comparison between MobileBERT and the same-size model (12 layers, 384 hidden size and 128 embedding size) distilled form $\text{BERT}_{\text{LARGE}}$ (Whole Word Masking) on GLUE test sets and SQuAD 2.0 dev set. Following MobileBERT (Sun et al., 2019b), the reported results are directly fine-tuned on downstream tasks. We compute the speedup of MobileBERT according to their reported latency.

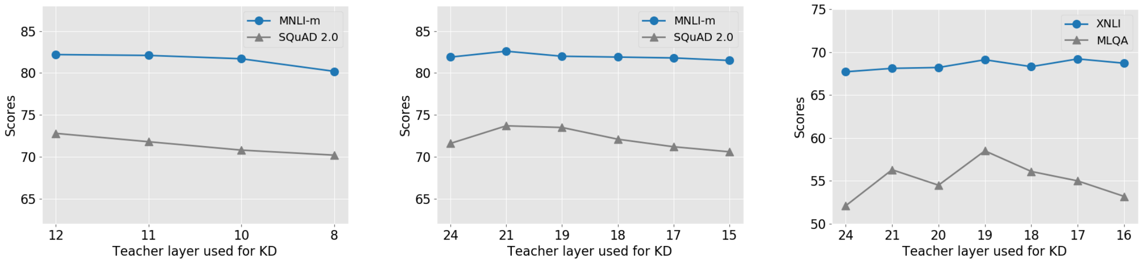

(a) $\text{BERT}_{\text{BASE}}$ as the teacher (b) $\text{BERT}_{\text{LARGE}}$ as the teacher (c) XLM-R_{LARGE} as the teacher

Figure 2: 6×384 models trained using different $\text{BERT}_{\text{BASE}}$ (a), $\text{BERT}_{\text{LARGE}}$ (b) and XLM-R_{LARGE} (c) layers.

Model	Teacher	SQuAD2	MNLI-m	SST-2
6×384 Ours	$\text{BERT}_{\text{BASE}}$	72.9	**82.8**	91.3
+ More Att-Rels	$\text{BERT}_{\text{BASE}}$	**73.3**	**82.8**	**91.6**
6×384 Ours	$\text{BERT}_{\text{LARGE}}$	74.3	83.0	91.1
+ More Att-Rels	$\text{BERT}_{\text{LARGE}}$	**74.7**	**83.2**	**92.4**
6×384 Ours	$\text{RoBERTa}_{\text{LARGE}}$	**76.4**	**84.4**	92.0
+ More Att-Rels	$\text{RoBERTa}_{\text{LARGE}}$	76.0	**84.4**	**92.1**
6×768 Ours	$\text{BERT}_{\text{BASE}}$	76.3	84.2	**92.4**
+ More Att-Rels	$\text{BERT}_{\text{BASE}}$	**76.8**	**84.4**	92.3
6×768 Ours	$\text{BERT}_{\text{LARGE}}$	77.7	85.0	**92.5**
+ More Att-Rels	$\text{BERT}_{\text{LARGE}}$	**78.1**	**85.2**	**92.5**
6×768 Ours	$\text{RoBERTa}_{\text{LARGE}}$	**81.6**	87.0	**94.5**
+ More Att-Rels	$\text{RoBERTa}_{\text{LARGE}}$	81.2	**87.3**	94.1

Table 10: Results of introducing more self-attention relations (Q-K, K-Q, Q-V, V-Q, K-V and V-K relations).

multiple attention heads.

5 Discussion

5.1 Comparison with MobileBERT

MobileBERT (Sun et al., 2019b) compresses a specially designed teacher model (in the $\text{BERT}_{\text{LARGE}}$ size) with inverted bottleneck modules into a 24-layer student using the bottleneck modules. Since our goal is to compress different large models (e.g. BERT and RoBERTa) to small models using standard Transformer architecture, we note that our student model can not directly compare with MobileBERT. We provide results of a student model with the same parameter size for a reference. A public large-size model ($\text{BERT}_{\text{LARGE-WWM}}$) is used as the teacher, which achieves similar performance as MobileBERT's teacher. We distill $\text{BERT}_{\text{LARGE-WWM}}$ into a student model (25M parameters) using the same training data (i.e., English Wikipedia and BookCorpus). The test results of GLUE and dev result of SQuAD 2.0 are illustrated in Table 9. Our model outperforms MobileBERT across most tasks with a faster inference speed. Moreover, our method can be applied for different teachers and has much fewer restriction of students.

We also observe that our model performs relatively worse on CoLA compared with MobileBERT. The task of CoLA is to evaluate the grammatical acceptability of a sentence. It requires more fine-grained linguistic knowledge that can be learnt from language modeling objectives. Fine-tuning the model using the MLM objective as in MobileBERT brings improvements for CoLA. However, our preliminary experiments show that this strategy will lead to slight drop for other GLUE tasks.

5.2 Results of More Self-Attention Relations

In Table 9 and 10, we report results of students trained using more self-attention relations (Q-K, K-Q, Q-V, V-Q, K-V and V-K relations). We observe improvements across most tasks, especially for student models distilled from BERT. Fine-grained self-attention knowledge in more attention relations improves our students. However, introducing more self-attention relations also brings a higher computational cost. In order to achieve a balance between performance and computational cost, we choose to transfer Q-Q, K-K and V-V self-attention relations instead of all self-attention relations in this work.

6 Conclusion

We generalize deep self-attention distillation in MiniLM by employing multi-head self-attention relations to train the student. Our method introduces more fine-grained self-attention knowledge and eliminates the restriction of the number of student's attention heads. Moreover, we show that transferring the self-attention knowledge of an upper middle layer achieves better performance for large-size teachers. Our monolingual and multilingual models distilled from BERT, RoBERTa and XLM-R obtain competitive performance and outperform state-of-the-art methods. For future work, we are exploring an automatic layer selection algorithm. We also would like to apply our method to larger pretrained Transformers.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Edward Guo. 2019. Knowledge distillation from internal representations. *CoRR*, abs/1910.03723.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. *arXiv preprint arXiv:2002.12804*.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-ling Mao, and Heyan Huang. 2019. Cross-lingual natural language generation via pre-training. *CoRR*, abs/1909.10481.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *CoRR*, abs/2007.07834.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic BERT with adaptive width and depth. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. 2018. Attention-guided answer distillation for machine reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2077–2086.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Patrick S. H. Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019b. MLQA: evaluating cross-lingual extractive question answering. *CoRR*, abs/1910.07475.

Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. 2020. BERT-EMD: many-to-many layer mapping for BERT compression with earth mover's distance. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3009–3018. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Xtremedistil: Multi-stage distillation for massive multilingual models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2221–2234. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's BERT: smaller and faster transformer models. *CoRR*, abs/2004.03844.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4322–4331.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2019b. Mobilebert: Task-agnostic compression of bert by progressive knowledge transfer.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical BERT models for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3630–3634.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing BERT by progressive module replacing. *CoRR*, abs/2002.02925.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Sergey Zagoruyko and Nikos Komodakis. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Corpus	**#Train**	**#Dev**	**#Test**	**Metrics**
Single-Sentence Tasks				
CoLA	8.5k	1k	1k	Matthews Corr
SST-2	67k	872	1.8k	Accuracy
Similarity and Paraphrase Tasks				
QQP	364k	40k	391k	Accuracy/F1
MRPC	3.7k	408	1.7k	Accuracy/F1
STS-B	7k	1.5k	1.4k	Pearson/Spearman Corr
Inference Tasks				
MNLI	393k	20k	20k	Accuracy
RTE	2.5k	276	3k	Accuracy
QNLI	105k	5.5k	5.5k	Accuracy
WNLI	634	71	146	Accuracy

Table 11: Summary of the GLUE benchmark.

#Train	**#Dev**	**#Test**	**Metrics**
130,319	11,873	8,862	Exact Match/F1

Table 12: Dataset statistics and metrics of SQuAD 2.0.

A GLUE Benchmark

The summary of datasets used for the General Language Understanding Evaluation (GLUE) benchmark[4] (Wang et al., 2019) is presented in Table 11.

B SQuAD 2.0

We present the dataset statistics and metrics of SQuAD 2.0[5] (Rajpurkar et al., 2018) in Table 12.

C Hyper-parameters for Fine-tuning

Extractive Question Answering For SQuAD 2.0, the maximum sequence length is 384. The batch size is set to 32. We choose learning rates from {3e-5, 6e-5, 8e-5, 9e-5} and fine-tune the model for 3 epochs. The warmup ration and weight decay is 0.1 and 0.01.

---

[4] `https://gluebenchmark.com/`

[5] `http://stanford-qa.com`

GLUE The maximum sequence length is 128 for the GLUE benchmark. We set batch size to 32, choose learning rates from {1e-5, 1.5e-5, 2e-5, 3e-5, 5e-5} and epochs from {3, 5, 10} for different student models. We fine-tune CoLA task with longer training steps (25 epochs). The warmup ration and weight decay is 0.1 and 0.01.

Cross-lingual Natural Language Inference (XNLI) The maximum sequence length is 128 for XNLI. We fine-tune 5 epochs using 128 as the batch size. The learning rates are chosen from {5e-5, 6e-5}.

Cross-lingual Question Answering For MLQA, the maximum sequence length is 512. We fine-tune 3 epochs using 32 as the batch size. The learning rates are chosen from {5e-5, 6e-5}.

Attending via both Fine-tuning and Compressing

Jie Zhou[1,2], Yuanbin Wu[2], Qin Chen[2], Xuanjing Huang[3], and Liang He[1,2]

[1]Shanghai Key Laboratory of Multidimensional Information Processing
[2]School of Computer Science and Technology, East China Normal University
jzhou@ica.stc.sh.cn, {ybwu,qchen,lhe}@cs.ecnu.edu.cn
[3]School of Computer Science, Fudan University
xjhuang@fudan.edu.cn

Abstract

Though being a primary trend for enhancing interpretability of neural networks, attention mechanism's reliability and validity are still under debate. In this paper, we try to purify attention scores to obtain a more faithful explanation of downstream models. Specifically, we propose a framework consisting of a learner and a compressor, which performs fine-tuning and compressing iteratively to enhance the performance and interpretability of the attention mechanism. The learner focuses on learning better text representations to achieve good decisions by fine-tuning, while the compressor aims to perform compressions over the representations to retain the most useful clues for explanations with a Variational information bottleneck ATtention (VAT) mechanism. Extensive experiments on eight benchmark datasets show the great advantages of our proposed approach in terms of both performance and interpretability.

1 Introduction

Attention mechanisms (Bahdanau et al., 2014) have achieved great success in various natural language processing (NLP) tasks. They are introduced to mimic the human eye focusing on important parts in the inputs when predicting labels. The existing studies show attention mechanisms can improve not only the performance but also the interpretability of the models (Mullenbach et al., 2018; Xie et al., 2017; Xu et al., 2015). Li et al. (2016) pointed the view: "Attention provides an important way to explain the workings of neural models". Additionally, Wiegreffe and Pinter (2019) showed that attention mechanisms could help understand the inner workings of a model.

The basic assumption of understanding of models with attention scores is that the inputs (e.g., words) with high attentive weights are essential for making decisions. However, as far as we know, it has not been formally verified. Existing research (Jain and Wallace, 2019) also shows that attention is not explicable, and there are a lot of controversy regarding to the result explanations (Wiegreffe and Pinter, 2019; Jain and Wallace, 2019). Moreover, we find that though the attention mechanism can help improve the performance for text classification in our experiments, it may focus on the irrelevant information. For example, in the sentence "A very funny movie.", the long short-term memory model with standard attention (LSTM-ATT) infers a correct sentiment label while pays more attention to the irrelevant word "movie", making the result difficult to explain.

In general, the attention weights are only optimized to encode the task-relevant information while are not restricted to imitate human behavior. In order to enhance the interpretability of the attention mechanism, recent studies turn to integrate the human provided explanation signals into the attention models. Rei and Søgaard (2018) regularized the attention weights with a small amount of word-level annotations. Barrett et al. (2018); Bao et al. (2018) improved the explanation of attention by aligning explanations with human-provided rationales. These methods rely on additional labour consuming labelling for enhancing explanations, which is hard to extend to other datasets or tasks.

In this paper, we aim to train a more efficient and effective interpretable attention model without any pre-defined annotations or pre-collected explanations. Specifically, we propose a framework consisting of a learner and a compressor, which enhances the performance and interpretability of the attention model for text classification[1]. The learner learns text representations by fine-tuning

---

[1] We focus on the task of text classification, but our method can be easily extended to other NLP or CV tasks with attention mechanisms.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2152–2161
August 1–6, 2021.

the encoder. Regarding to the compressor, we are motivated by the effectiveness of the information bottleneck (IB) (Tishby et al., 1999) to enhance performance (Li and Eisner, 2019) or detect important features (Bang et al., 2019; Chen and Ji, 2020; Jiang et al., 2020; Schulz et al., 2020), and present a Variational information bottleneck ATtention (VAT) mechanism using IB to keep the most relevant clues and forget the irrelevant ones for better attention explanations. In particular, IB is integrated into attention to minimize the mutual information (MI) with the input while preserving as much MI as possible with the output, which provides more accurate and reliable explanations by controlling the information flow.

To evaluate the effectiveness of our proposed approach, we adapt two advanced neural models (LSTM and BERT) within the framework and conduct experiments on eight benchmark datasets. The experimental results show that our adapted models outperform the standard attention-based models over all the datasets. Moreover, they exhibit great advantages with respect to interpretability by both qualitative and quantitative analyses. Specifically, we obtain significant improvements by applying our model to the semi-supervised word-level sentiment detection task, which detects the sentiment words based on attention weights via only sentence-level sentiment label. In addition, we provide the case studies and text representation visualization to have an insight into how our model works.

The main contributions of this work are summarized as follows.

- We propose a novel framework to enhance the performance and interpretability of the attention models, where a learner is used to learn good representations by fine-tuning and a compressor is used to obtain good attentive weights by compressing iteratively.

- We present a Variational information bottleneck ATtention (VAT) mechanism for the compressor, which performs compression over the text representation to keep the task related information while reduce the irrelevant noise via information bottleneck.

- Extensive experiments show the great advantages of our models within the proposed framework, and we perform various qualitative and quantitative analyses to shed light on why our models work in both performance and interpretability.

2 Related Work

In this section, we survey related attention mechanisms (Bahdanau et al., 2014) and review the most relevant studies on information bottleneck (IB) (Tishby et al., 1999).

Attention has been proved can help explain the internals of neural models (Li et al., 2016; Wiegreffe and Pinter, 2019) though it is limited (Jain and Wallace, 2019). Many researchers try to improve the interpretability of the attention mechanisms. Rei and Søgaard (2018) leveraged small amounts of word-level annotations to regularize attention. Kim et al. (2017) introduced a structured attention mechanism to learn attention variants from explicit probabilistic semantics. Barrett et al. (2018); Bao et al. (2018) aligned explanations with human-provided rationales to improve the explanation of attention. Unlike these methods that require prior attributions or human explanations, the VAT method enforces the attention to learn the vital information while filter the noise via IB.

A series of studies motivate us to utilize IB to improve the explanations of attention mechanisms. Li and Eisner (2019) compressed the pre-trained embedding (e.g., BERT, ELMO), remaining only the information that helps a discriminative parser through variational IB. Zhmoginov et al. (2019) utilized the IB approach to discover the salient region. Some works (Jiang et al., 2020; Chen et al., 2018; Guan et al., 2019; Schulz et al., 2020; Bang et al., 2019) proposed to identify vital features or attributions via IB. Moreover, Chen and Ji (2020) designed a variational mask strategy to delete the useless words in the text. As far as we are aware, we are the first ones to leverage IB into attention mechanisms to train more interpretable attention with better accuracy.

3 Our Approach

In this section, we introduce our framework consisting of a learner and a compressor with a Variational information bottleneck ATtenttion (VAT) mechanism. Given an attention-based neural network model, we formulate our idea within the framework of variational information bottleneck (VIB) (Tishby et al., 1999). Our framework aims to improve the attention's interpretalility with better performance by restricting the attention to capture the crucial words while filter the useless information.

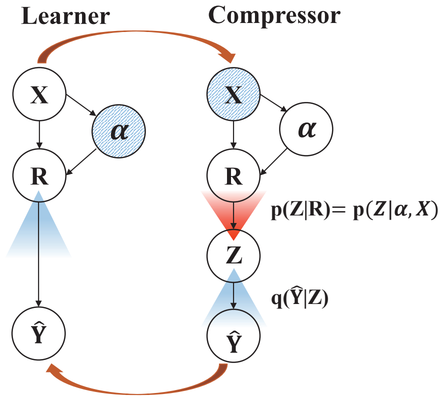

Figure 1: The framework.The learner aims to learn the good text representation X by fine-tuning, and the compressor aims to learn good attention weights by compressing the attentive representations to capture the important words while forget the redundant information via VAT. The blue circles mean the corresponding parameters of the modules are fixed.

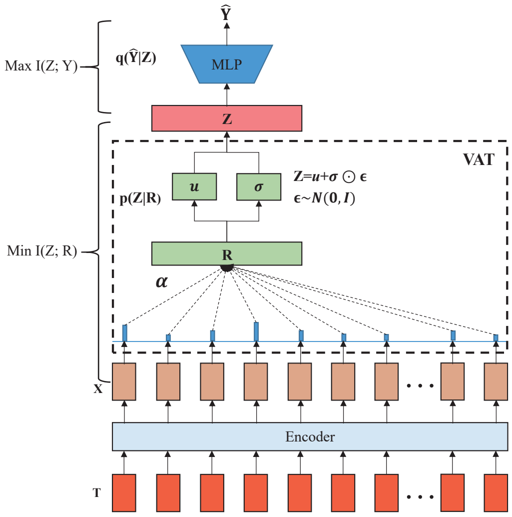

Figure 2: The architecture of our VAT (Compressor). First, we obtain the input text's word representations X via an encoder trained by the learner. Then, we calculate Z by compressing the text representation R that is the weighted sum of X based on the attention α, while remaining the maximum information to judge Y by inputting Z into a MLP classifier for predicting.

3.1 Overview

Our framework is composed of a learner and a compressor, which performs fine-tuning and compressing iteratively (Figure 1). The learner aims to learn a task-specific contextual word representation by fine-tuning. The compressor enforces the model to learn task-relevant information while reduce irrelevant information via IB. We iteratively perform the learner and compressor (fine-tuning and compressing) to improve each other.

Learner. We adopt a basic attention-based neural network model as a learner to learn representations of the words based on the good attention weights learned by the compressor. The model is optimized by cross-entropy loss to learn the label-relevant information. In this phase, we fix the attention's parameters so that the model will focus on updating the encoder to learn word representations.

Compressor. To restrict the attention to capture the vital information while reduce the noise, we integrate IB into attention mechanisms to compress the text attentive representation. We fix the encoder's parameters so that the model will focus on learning the attention weights based on current representations obtained from the learner.

3.2 Basic Attention Model (Learner)

In this section, we describe our learner, which is an attention-based neural network model. First, given a text $T = \{w_1, w_2, ..., w_{|T|}\}$, where $|T|$ is the length of text T, we feed it into an encoder with a word embedding layer. We adopt LSTM and BERT models as our encoder, and other models can also be applied to our framework. We obtain the context-aware word representations $x = [x_1, x_2, ..., x_{|T|}]$, where x_i is the hidden vector of the word w_i.

$$x = encoder(T, \theta_{encoder}), \tag{1}$$

where $\theta_{encoder}$ is the parameters of the encoder.

Based on the contextual word representations, attention mechanism (Bahdanau et al., 2014) [2] is utilized to capture the important parts in the text and obtain the text representation R, which is calculated as,

$$\begin{aligned} R &= \sum_{i=1}^{n} \alpha_i x_i \\ \alpha_i &= \mathrm{softmax}(\mathbf{v}_a^\top \tanh(\mathbf{W}_a x_i)) \end{aligned} \tag{2}$$

where $\theta_{attention} = \{v_a, W_a\}$ is the trainable parameters of the attention, which is not updated in this step to learn the word representation x based the good attention learned by the compressor. $\alpha = [\alpha_1, \alpha_2, ..., \alpha_{|T|}]$ is the attention weights. Finally, we input the text representation R into a

---

[2] In this paper, we only explore the local attention mechanism on our framework, other attention mechanisms (e.g., multi-head attention (Vaswani et al., 2017)) can also be applied. We would like to explore it in future work.

multi-layer perceptron (MLP) to predict the probability. The cross-entropy loss is used to optimize the model.

3.3 Variational Information Bottleneck Attention (Compressor)

The learner optimizes the sentence representations by minimizing the cross-entropy loss, which does not restrict the model to ignore the useless information. Thus, we compress sentence representations R into a latent representation Z that retains most useful information to infer the label Y. We propose to accomplish this by integrating VIB into the attention mechanism (Figure 2).

To ensure Z contains maximum ability to predict Y ($I(Z;Y)$) while has the least redundant information form R ($-I(Z;R)$), we use the standard IB theory (Tishby et al., 1999) and define the objective function as:

$$\max_{\boldsymbol{\alpha}} I(Z;Y) - \beta \cdot I(Z;R) \tag{3}$$

where $I(\cdot;\cdot)$ means the mutual information and β is a coefficient to balance two components. The main challenge is to estimate the lower bound for $I(Z;Y)$ and the upper bound for $I(Z;R)$. [3]

The joint probability $p_\theta(r,y,z)$ can be factored as $p(r) \cdot p(y \mid r) \cdot p_\theta(z \mid r)$ based on the independence assumption [4]. By replacing the conditional distribution $p_\theta(y \mid z)$ with a variational approximation $q_\phi(y \mid z)$, we obtain a lower bound of $I(Z;Y)$. $q_\phi(y \mid z)$ is a simple classifier that runs on a compressed text representation z.

$$\begin{aligned}&\overbrace{\mathbb{E}_{p_\theta(y,z)}[\log \frac{p_\theta(y \mid z)}{p(y)}]}^{I(Z;Y)} - \overbrace{\mathbb{E}_{p_\theta(y,z)}[\log \frac{q_\phi(y \mid z)}{p(y)}]}^{\text{lower bound}} \\ &= \mathbb{E}_{p_\theta(z)}[\mathrm{KL}(p_\theta(y \mid z) \| q_\phi(y \mid z))] \geqslant 0\end{aligned} \tag{4}$$

where $KL[\cdot\|\cdot]$ represents Kullback-Leibler divergence.

Specifically, we regard $p(y)$ as constant and then minimize $\mathbb{E}_{p_\theta(y,z)}[\log q_\phi(y \mid z)]$. Since we must first sample r to sample y, z from $p_\theta(r,y,z)$, the lower bound of $I(Z;Y)$ is computed as,

$$I(Z;Y) \geqslant \mathbb{E}_{p(r,y)}[\mathbb{E}_{p_\theta(z|r)}[\log q_\phi(y \mid z)]] \tag{5}$$

We calculate the upper bound of $I(Z;R)$ by replacing $p_\theta(z)$ with a variational distribution $r_\psi(z)$,

$$\begin{aligned}&\overbrace{\mathbb{E}_{p(r)}[\mathbb{E}_{p_\theta(z|r)}[\log \frac{p_\theta(z \mid r)}{r_\psi(z)}]]}^{\text{upper bound}} - \overbrace{\mathbb{E}_{p(r)}[\mathbb{E}_{p_\theta(z|r)}[\log \frac{p_\theta(z \mid r)}{p(z)}]]}^{\mathrm{I}(Z;R)} \\ =&\mathbb{E}_{p(r)}[\mathrm{KL}(p(z) \| r_\psi(z))] \geqslant 0\end{aligned} \tag{6}$$

The upper bound of $I(Z;R)$ is computed as,

$$\begin{aligned}I(Z;R) &\leqslant \mathbb{E}_{p(r)}[\mathbb{E}_{p_\theta(z|r)}[\log \frac{p_\theta(z \mid r)}{r_\psi(z)}]] \\ &= \mathbb{E}_{p(r)}[\mathrm{KL}[p_\theta(z \mid r) \| r_\psi(z)]]\end{aligned} \tag{7}$$

Then, we obtain the lower bound $\mathcal{L}$ of IB by substituting Equation 5 and 7 into Equation 3:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{p(r,y)}[&\mathbb{E}_{p_\theta(z|r)}[\log q_\phi(y \mid z)] \\ &- \beta \cdot \mathrm{KL}[p_\theta(z \mid r) \| r_\psi(z)]]\end{aligned} \tag{8}$$

The first component in $\mathcal{L}$ is to keep the most useful information in $p_\theta(z|r)$ for inferring y, while the second one is to regularize $p_\theta(z|r)$ with a predefined prior distribution $r_\psi(z)$ (e.g., Gaussian distribution). To compute $p_\theta(z|r)$, we adopt the reparametrization trick for multivariate Gaussians (Rezende et al., 2014), which obtains the gradient of parameters that derive z from a random noise ϵ.

$$z = u + \sigma \odot \epsilon, \epsilon \sim N(0, I) \tag{9}$$

where $\odot$ means element-wise multiplication. u and σ denote the mean and covariance defined by two functions of R, where $R = \alpha \cdot x$ that is learned based on attention. In particular, two MLP are used to predict u and σ.

Finally, we input the z into a MLP to predict $q_\phi(y \mid z)$ and optimize the attention's parameter via Equation 8.

4 Experiment Setup

We adopt two typical neural network models, attention-based LSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2019), to explore our VAT algorithm.

4.1 Datasets and Baselines

Datasets To evaluate the effectiveness of our VAT model, we conduct the experiments over eight benchmark datasets: IMDB (Maas et al., 2011), Stanford Sentiment Treebank with (includes SST-1 and its binary version SST-2) (Socher et al., 2013), Yelp (Zhang et al., 2015), AG News (Zhang et al., 2015), TREC (Li and Roth, 2002), subjective/objective classification Subj (Pang and Lee, 2005) and Twitter (Rosenthal et al., 2015, 2014). The statistics information of these datasets are shown in Table 1.

---

[3] We give the main steps as follows and the detailed derivation is provided in supplementary materials.

[4] $Y \to R \to Z$: Y and Z are independent given R.

	IMDB	SST-1	SST-2	Yelp	AG News	Trec	Subj	Twitter
Class	2	5	2	2	4	6	2	3
Length	268	18	19	138	32	10	23	22
#train	20,000	8,544	6,920	500,000	114,000	5,000	8,000	7,969
#dev	5,000	1,1101	872	60,000	6,000	452	1,000	1,375
#test	25,000	2,210	1,821	38,000	7,600	500	1,000	3,795

Table 1: The statistics information of the datasets, where Class is the number of the class, Length is average text length, and #train/#dev/#test counts the number of samples in the train/dev/test sets.

	IMDB	SST-1	SST-2	Yelp	AG News	Trec	Subj	Twitter	Average
LSTM-base	88.79	45.20	85.45	95.10	91.91	90.00	89.00	71.25	82.09
LSTM-ATT	88.16	46.29	84.73	95.06	91.88	91.00	90.80	70.75	82.33
LSTM-VAT	**88.98**	**47.42**	**86.22**	**95.32**	**92.04**	**92.80**	**91.10**	**71.62**	**83.19**
BERT-base	91.90	51.44	91.60	96.07	93.52	96.60	96.50	75.28	86.61
BERT-ATT	91.81	51.13	91.16	97.20	93.41	96.40	96.20	74.84	86.52
BERT-VAT	**92.11**	**51.99**	**91.98**	**97.36**	**93.71**	**97.20**	**96.70**	**77.13**	**87.27**

Table 2: The main results of text classification.

Baselines We compare our model with two kinds of models, basic models (LSTM/BERT-base) and attention-based models (LSTM/BERT-ATT). LSTM-base takes the max-pooling of the LSTM's hidden vectors as text representation. For BERT-base, the "[CLS]" representation is obtained as the sentence representation. LSTM-ATT model is a standard attention-based LSTM model that has the same structure as the learner. We obtain the BERT-ATT by replacing the LSTM encoder with BERT in LSTM-ATT. Our models are marked with VAT (LSTM-VAT, BERT-VAT), which integrate VIB into attention-based neural models.

4.2 Implementation Details

For LSTM-based models, we use GloVe embedding (Pennington et al., 2014) with 300-dimension to initialize the word embedding and fine-tune it during the training. We randomly initialize all out-of-vocabulary words and weights with the uniform distribution $U(-0.1, 0.1)$. For the BERT-based models, we fine-tune pre-trained BERT-base model. The dimension of hidden state vectors of LSTM is 100 and the max sentence length is 256 in our experiments. Adam (Kingma and Ba, 2014) is utilized as the optimizer with learning rate 0.001 (for LSTM-based model) and 0.00001 (for BERT-based model). We also search different values $\beta \in \{0.01, 0.1, 1, 10\}$.

5 Experiments

First, we perform our models and baselines on eight benchmark datasets and visualize the text representation to verify the effectiveness of VAT (Section 5.1). Second, to further investigate our VAT model, we adopt two popular explanation metrics for quantitative evaluation (Section 5.2). Third, we apply our models to semi-supervision sentiment detection task to evaluate the explanation of our model (Section 5.3). Fourth, we explore the influence of our iteration strategy in Section 5.4 and provide case studies in Section 5.5. For the limitation of the space, we may only list the results on parts of the datasets in some cases since the conclusions are similar for other datasets. The complete results are presented in the supplementary materials.

5.1 Main Results

We report the accuracy of our VAT and baselines based on LSTM and BERT (Table 2). From these results, we find the following observations: **1)** our models (LSTM/BERT-VAT) outperform all the corresponding baselines over all the eight datasets, which denotes the effectiveness of our VAT on both LSTM and BERT-based models; **2)** compared with attention-based models (LSTM/BERT-ATT), our models obtain better results. It indicates reducing the irrelevant information in input via VAT can improve the performance of the models.

Furthermore, we visualize the sentence representations obtained from LSTM/BERT-ATT and -VAT models (Figure 3). We randomly select 1000 samples from the test set for each dataset. We can find that our VAT model can reduce the distance of the samples in a class and add the distance of the samples in different classes. For example, it is hard to split the positive samples from the negative ones based on the representations obtained from LSTM-ATT for the IMDB dataset, while the divider line based on our VAT is clear. These ob-

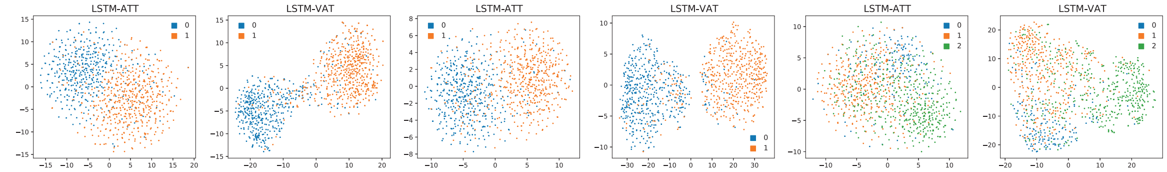

(a) IMDB (LSTM) (b) Subj (LSTM) (c) Twitter (LSTM)

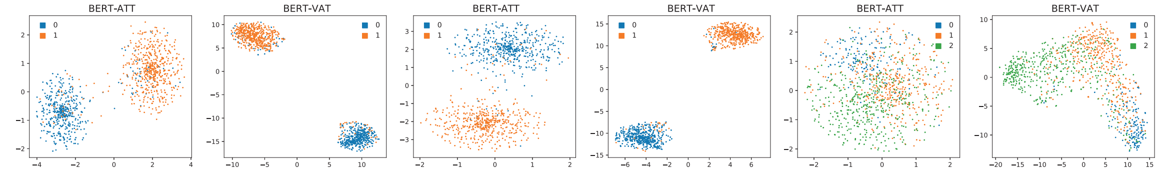

(d) IMDB (BERT) (e) Subj (BERT) (f) Twitter (BERT)

Figure 3: Visualization of text representation obtained from LSTM/BERT-ATT and LSTM/BERT-VAT. We use t-SNE to transfer 100/768-dimensional feature space into two-dimensional space.

		IMDB	SST-1	SST-2	Yelp	AG News	Trec	Subj	Twitter
Accuracy	LSTM-base	88.79	45.20	85.45	95.10	91.91	90.00	89.00	71.25
AOPC	Random	0.30	5.97	7.58	1.02	1.87	19.40	1.50	4.72
	LSTM-ATT	5.27	12.94	20.54	6.64	5.99	31.00	2.10	19.10
	LSTM-VAT	**6.13**	**14.34**	**21.58**	**7.12**	**6.59**	**37.20**	**6.30**	**20.37**
Accuracy	BERT-base	91.90	51.44	91.60	96.07	93.52	96.60	96.50	75.28
AOPC	Random	0.60	33.26	41.46	3.60	44.20	65.80	45.70	59.21
	BERT-ATT	2.81	33.98	41.52	4.73	52.22	71.60	45.70	59.39
	BERT-VAT	**3.17**	**34.03**	**41.52**	**6.64**	**54.70**	**72.20**	**45.80**	**59.45**

Table 3: The results of AOPC.

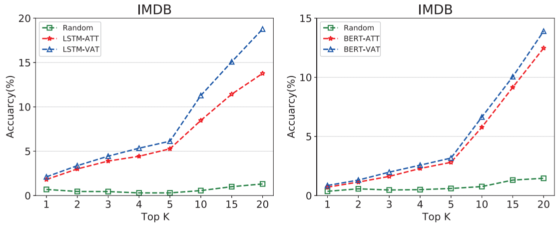

(a) IMDB (LSTM) (b) IMDB (BERT)

Figure 4: The influence of Top-K for LSTM/BERT-based models in terms of AOPC.

servations show our VAT model can learn a better task-specific representation by enforcing the model to reduce the task-irrelevant information.

5.2 Quantitative Evaluation

In this section, we evaluate our VAT model using two metrics, AOPC and post-hoc accuracy, which are widely used for explanations (Chen and Ji, 2020). Note that well-trained LSTM/BERT-base is used for evaluating the performance of classification.

AOPC. To evaluate the faithfulness of explanations to our models, we adopt the area over the perturbation curve (AOPC) (Nguyen, 2018; Samek et al., 2016) metric. It calculates the average change of accuracy over test data by deleting top K words via attentive weights. The larger the value of AOPC, the better the explanations of the models.

Table 3 displays the results with $K = 5$. We compare our models with random and basic attention-based models. From the results, we observe that: **1)** basic attention-based models (LSTM/BERT-ATT) can find the important words in the sentence to some extent. Comparing with random (Random), LSTM/BERT-ATT obtains significant improvement; **2)** Our models (LSTM/BERT-VAT) outperform the standard attention-based models. It indicates that integrating VIB into the attention mechanism can help improve the interpretability of the models by filtering the useless information; **3)** BERT model is sensitive to the context; deleting the words will destroy the semantic information of the sentence and significantly affect the model's performance.

We also explore the influence of top-K (Figure 4). Intuitively, the more words we delete, the larger accuracy the models reduce. Our models reduce more performance than random and attention-based

		IMDB	SST-1	SST-2	Yelp	AG News	Trec	Subj	Twitter
Accuracy	LSTM-base	88.79	45.20	85.45	95.10	91.91	90.00	89.00	71.25
Post-hoc	Random	58.48	34.21	71.33	64.74	62.45	71.40	78.40	54.07
	LSTM-ATT	83.96	40.56	82.70	87.80	78.96	73.60	87.40	70.20
	LSTM-VAT	**84.41**	**43.39**	**84.35**	**88.82**	**81.43**	**79.20**	**89.10**	**71.23**
Accuracy	BERT-base	91.90	51.44	91.60	96.07	93.52	96.60	96.50	75.28
Post-hoc	Random	51.50	20.27	50.52	50.21	26.74	26.60	50.60	40.50
	BERT-ATT	51.72	29.19	58.92	53.63	37.53	34.20	61.90	53.68
	BERT-VAT	**53.40**	**30.23**	**61.34**	**56.58**	**43.08**	**36.80**	**65.40**	**56.05**

Table 4: The results of post-hoc accuracy.

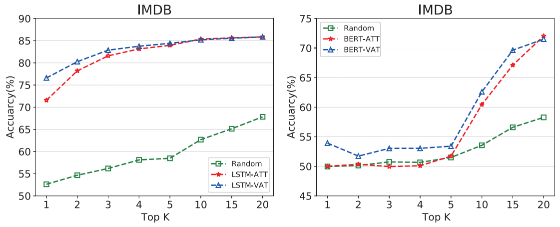

(a) IMDB (LSTM) (b) IMDB (BERT)

Figure 5: The influence of Top-K for LSTM-based models in terms of post-hoc.

models. For the IMDB dataset, when deleting top 20 words (average length is 268), the accuracy reduces about 19 points for our LSTM-VAT model while it is about 2 points for the random model.

Post-hoc Accuracy. We also adopt the post-hoc accuracy (Chen et al., 2018) to evaluate the influence of task-specific essential words on the performance of LSTM-based and BERT-based models. For each test sample, we select the top K words based on their attentive weights as input to make a prediction and compare it with the ground truth.

Table 4 presents the performance with $K = 5$. **First**, it is interesting to find that the post-hoc accuracy with five most important words on Sbuj dataset (89.10) is even better than the original sentence (89.00). Additionally, we obtain comparable results with only five words for SST-1, SST-2, and Twitter datasets. These show that our model can reduce the noise information since most of the words are useless for predictions in some cases. **Second**, for BERT-based models, the context words are also important for classification even though they may not be task-specific.

Similarly, we investigate the influence of top-K for post-hoc (Figure 5). The LSTM-base model with top-10 words selected by our LSTM-VAT model can achieve comparable results with the original samples in most cases. Additionally, for the IMDB dataset, the accuracy of LSTM-base with one word selected by our VAT model is even better than the one with 20 words selected randomly.

5.3 Semi-Supervised Word-Level Sentiment Detection

We perform semi-supervised word-level sentiment detection in Twitter (Rosenthal et al., 2015, 2014) to evaluate the interpretability of our VAT. This task requires to detect the sentiment words in a tweet via the sentiment polarity of the whole tweet. In the following example from the dataset, positive words ("good" and "fantastic") are marked with a bold font and the overall polarity of the tweet is positive:

Good *morning becky! Thursday is going to be* ***fantastic****!*

We use the SemEval 2013 Twitter dataset, which contains word-level sentiment annotation. We remove the samples with the neutral sentiment. We report word-level precision, recall, and F-measure for evaluating the models (Table 5), the same as (Rei and Søgaard, 2018). Note that we select the top-K (we set it as 1 and 5 here) words according to the attention weights as the sentiment words.

We compare our VAT model with random and attention-based models. The results show attention-based models can capture the important words in the text, to a certain extent. Since our VAT can reduce irrelevant information, it performs better than the standard attention model. Also, LSTM-based models outperform BERT-based models for this task in most cases. It is because that BERT learns much semantic information from the text, and context information plays a vital role in prediction.

5.4 Influence of Iteration

We propose to train the learner and compressor iteratively so that the learner optimizes the word representations based on the good attention, and the compressor optimizes the attention based on the good word representations. To have a deep look at how it works, we first provide our VAT model's accuracy with different iterations (Table 6). From the results, we can find that the model's performance will improve at first, then it will converge.

	Positive						Negative					
	P@1	R@1	F1@1	P@5	R@5	F1@5	P@1	R@1	F1@1	P@5	R@5	F1@5
Random	14.88	4.78	6.56	14.59	23.34	16.06	20.52	5.61	8.19	17.18	23.68	17.97
LSTM-ATT	58.70	26.04	32.73	30.30	54.17	34.70	47.13	15.74	21.39	28.24	42.04	30.33
LSTM-VAT	**65.20**	**29.38**	**36.60**	33.04	**58.40**	37.77	**60.00**	**21.42**	**28.76**	32.70	**49.19**	**35.35**
BERT-ATT	46.44	16.52	21.82	33.13	52.52	35.66	37.74	9.19	13.46	30.82	39.65	30.23
BERT-VAT	55.24	20.62	26.90	**37.26**	58.39	**40.09**	43.83	11.15	16.20	**36.42**	44.55	35.30

Table 5: The results of semi-supervision word-level sentiment detection in twitter.

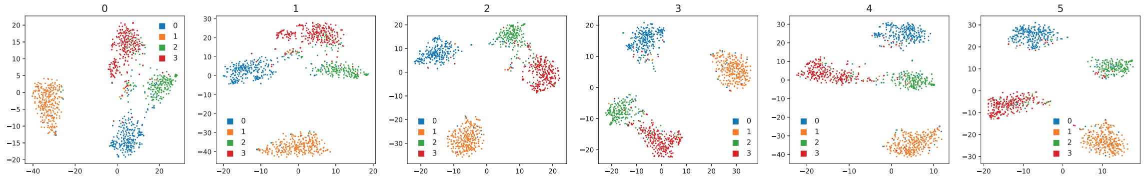

(a) AG News (LSTM)

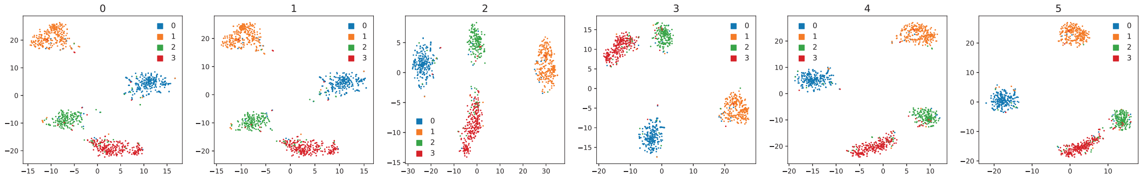

(b) AG News (BERT)

Figure 6: Visualization of text representation obtained from LSTM/BERT-VAT with different iterations. We use t-SNE to transfer 100/768-dimensional feature space into two-dimensional space.

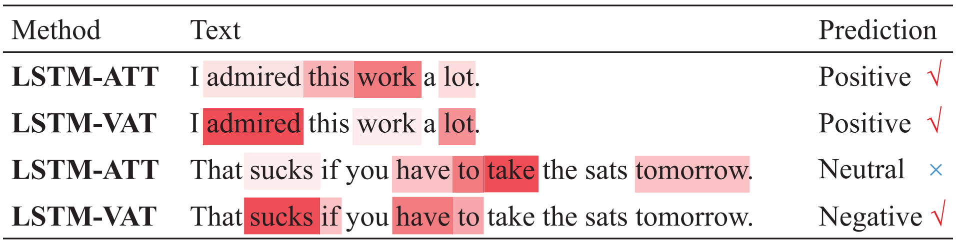

Method	Text	Prediction
LSTM-ATT	I admired this work a lot.	Positive √
LSTM-VAT	I admired this work a lot.	Positive √
LSTM-ATT	That sucks if you have to take the sats tomorrow.	Neutral ×
LSTM-VAT	That sucks if you have to take the sats tomorrow.	Negative √

Figure 7: Two examples of attention visualization. Red denotes the attentive weights of the words. A deeper color indicates a larger value.

	Dataset	0	1	2	3	4	5
LSTM-VAT	Twitter	70.75	71.62	70.96	70.67	71.06	70.98
	IMDB	88.16	88.98	88.22	88.84	88.14	88.60
BERT-VAT	Twitter	74.84	75.26	77.71	77.13	76.68	76.76
	IMDB	91.81	92.06	92.11	92.09	91.92	91.96

Table 6: The accuracy with different iteration number with our LSTM/BERT-VAT model.

Also, we draw change of the sentence representation with different iterations (Figure 6). Similarly, we observe that fine-tuning and compressing iteratively can improve the sentence representations. The samples with the same class are close, and the samples with different classes have a large distance.

5.5 Case Studies

To understand why our proposed VAT model is more effective than the standard attention-based model, we visualize two examples of LSTM-based models using attention heatmaps (Figure 7). **First**, the standard attention-based LSTM model focuses on the wrong words (e.g., "this", "work") even though it predicts the right sentiment while our VAT model finds the correct words (e.g., "admired", "lot"). It indicates integrating IB into attention can help it focus on the key words and reduce the noisy information. **Second**, our proposed model can also improve the attention's performance by capturing the critical words accurately. For example, in the sentence "That sucks if you have to take the sats tomorrow.", our model predicts the right class label by attending the words "sucks" and "have to."

6 Conclusions and Future Work

This paper proposes a VAT-based framework to improve the performance and interpretability of attentions via both fine-tuning and compressing. The experimental results on eight benchmark datasets for text classification verify the effectiveness of our models within this framework. In addition, we apply the framework for sentiment detection, which further demonstrates the superiority in terms of interpretability. It is also interesting to find that training the models by fine-tuning and compressing iteratively is effective to improve the text representations. In the future, we will investigate the effectiveness of our proposed attention framework for other tasks and areas, such as machine translation and visual question answering.

Acknowledgement

The authors wish to thank the reviewers for their helpful comments and suggestions. This research is (partially) supported by NSFC (62076097), STCSM (18ZR1411500), the Fundamental Research Funds for the Central Universities. This research is also funded by the Science and Technology Commission of Shanghai Municipality (19511120200 & 20511105102). The computation is performed in ECNU Multifunctional Platform for Innovation (001). The corresponding authors are Yuanbin Wu and Liang He.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Seojin Bang, Pengtao Xie, Heewook Lee, Wei Wu, and Eric Xing. 2019. Explaining a black-box using deep variational information bottleneck approach. *arXiv preprint arXiv:1902.06918*.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913.

Maria Barrett, Joachim Bingel, Nora Hollenstein, Marek Rei, and Anders Søgaard. 2018. Sequence classification with human attention. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 302–312.

Hanjie Chen and Yangfeng Ji. 2020. Learning variational word masks to improve the interpretability of neural text classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4236–4251.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. 2018. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. 2019. Towards a deep and unified understanding of deep neural models in nlp. In *International conference on machine learning*, pages 2454–2463. PMLR.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556.

Zhiying Jiang, Raphael Tang, Ji Xin, and Jimmy Lin. 2020. Inserting information bottleneck for attribution in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3850–3857.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Xiang Lisa Li and Jason Eisner. 2019. Specializing word embeddings (for parsing) by information bottleneck. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2744–2754.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1101–1111.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078.

Bo Pang and Lillian Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 293–302.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 task 10: Sentiment analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 451–463.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 task 9: Sentiment analysis in Twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 73–80.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. 2020. Restricting the flow: Information bottlenecks for attribution. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Naftali Tishby, Fernando C Pereira, and William Bialek. 1999. The information bottleneck method. In *Proceedings of the 37th annual Allerton Conference on Communication, Control, and Computing*, page 368–377.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Andrey Zhmoginov, Ian Fischer, and Mark Sandler. 2019. Information-bottleneck approach to salient region discovery. *arXiv preprint arXiv:1907.09578*.

Improving Event Causality Identification via Self-Supervised Representation Learning on External Causal Statement

Xinyu Zuo[1,2], **Pengfei Cao**[1,2], **Yubo Chen**[1,2], **Kang Liu**[1,2], **Jun Zhao**[1,2],
Weihua Peng[3] and **Yuguang Chen**[3]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]Beijing Baidu Netcom Science Technology Co., Ltd
`{xinyu.zuo,pengfei.cao,yubo.chen,kliu,jzhao}@nlpr.ia.ac.cn`
`{pengweihua,chenyuguang}@baidu.com`

Abstract

Current models for event causality identification (ECI) mainly adopt a supervised framework, which heavily rely on labeled data for training. Unfortunately, the scale of current annotated datasets is relatively limited, which cannot provide sufficient support for models to capture useful indicators from causal statements, especially for handing those new, unseen cases. To alleviate this problem, we propose a novel approach, shortly named CauSeRL, which leverages external causal statements for event causality identification. First of all, we design a self-supervised framework to learn context-specific causal patterns from external causal statements. Then, we adopt a contrastive transfer strategy to incorporate the learned context-specific causal patterns into the target ECI model. Experimental results show that our method significantly outperforms previous methods on EventStoryLine and Causal-TimeBank (+2.0 and +3.4 points on F1 value respectively).

1 Introduction

Event causality identification (ECI) aims to identify causal relations between events in texts, which can provide crucial clues for deep textual understanding (Girju, 2003; Oh et al., 2013, 2017). For example in Figure 1, an ECI system should identify two causal relations in S_1 with mentioned events: **noticed**$_{E1}$ $\xrightarrow{cause}$ **alerted**$_{E3}$ and **alerted**$_{E3}$ $\xrightarrow{cause}$ **ran**$_{E2}$.

To date, most existing methods regard this task as a classification problem and usually train ECI models on annotated data (Hashimoto et al., 2014; Riaz and Girju, 2014b; Mirza and Tonelli, 2016; Hu and Walker, 2017b; Gao et al., 2019). However, the scale of current annotated datasets are relatively limited, where the so far largest dataset EventStoryLine (Caselli and Vossen, 2017) only contains 258 documents, 4316 sentences, and 1770 causal event pairs. As a result, on the limited annotated examples, existing ECI models could not easily capture useful indicators from causal statements, especially for handing those new, unseen cases.

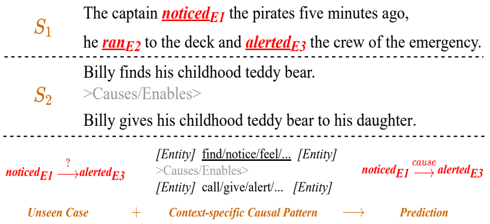

Figure 1: S_1 is a labeled data that contains unseen causal events and their statement when training; S_2 is an external causal statement; The bottom illustrates the context-specific causal pattern in S_2 could help identify the causality of unseen events in S_1.

To address this problem, Liu et al. (2020) employed external event-related knowledge bases (KBs) to enhance the causality inference, where those KBs store inherent causal relations between some given events. For those unseen events and unlabeled causalities in KBs, Liu et al. (2020) proposed a mention-mask based reasoner to enhance the causal statement representation. However, such mention-mask based reasoner is still trained on the human-annotated examples solely. It will still suffer from data limitations and have no capacity to handling unseen contexts. Moreover, Zuo et al. (2020) improved the performance of ECI with the distantly supervised labeled training data. However, their models are still limited to the unsatisfied qualities of the automatically generated data.

To address the insufficient annotated example problem, we employ a large number of external causal statements (Sap et al., 2018; Mostafazadeh et al., 2020) that can support adequate evidence of context-specific causal patterns (Liu et al., 2020)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2162–2172
August 1–6, 2021. ©2021 Association for Computational Linguistics

for understanding event causalities. For example in Figure 1, the context-specific causal pattern support by an external causal statement S_2 is helpful for identifying the causality of event *noticed*$_{E1}$ and event *alerted*$_{E3}$ in S_1, which is unseen when only training with labeled data. However, different from annotated examples for the ECI task, there are no event annotations in the external causal statements. As a result, it is difficult for the models to learn context-specific causal patterns from them to identify event causalities. To resolve this issue, inspired by Grill et al. (2020), we design a self-supervised representation learning framework to learn enhanced causal representations from external causal statements. Specifically, we iteratively sample two external causal statements, then take each of them as a target to learn the commonalities among them. Intuitively, we believe that the learned commonalities between different causal statements through self-supervision reflect such context-specific causal patterns which are helpful for identifying event causalities in the unseen cases.

Moreover, to incorporate the learned context-specific causal patterns from external causal statements into the target ECI model, we employ a contrastive transfer strategy. In specific, we regard the self-supervised representation learning module as a teacher model that *masters* abundant external causal statements, and the target ECI model as a student model. Methodologically, we make the representation of the causal events encoded by the student model should be close to the causal representation grasped by the teacher model, and keep the representation of the non-causal events away from it. In this way, the mutual information between the teacher and student models could be maximized (Tian et al., 2020). Then the learned context-specific causal patterns could be naturally transferred into the ECI model and the generalization could be improved.

In experiments, we evaluate our model on two benchmarks. The experimental results show that our model achieves SOTA performance. Then, concrete proofs show that the effectiveness of our self-supervised contrast-based framework for context-specific causal patterns learning and transfer.

In summary, the contributions are as follows:

- We propose a novel approach, shortly named **CauSeRL**, which could leverage external causal statements to identify the causalities between events.
- First of all, we design a self-supervised framework to learn context-specific causal patterns from external causal statements. Then, we adopt a contrastive transfer strategy to incorporate the learned context-specific causal patterns into target ECI model for identification.
- Experimental results on two benchmarks show that our model achieves the best performance.

2 Related Work

Event Causality Identification Up to now, identifying the causality implied in the text has attracted more and more attention (Hu and Walker, 2017a; Riaz and Girju, 2014b; Hashimoto et al., 2014; Riaz and Girju, 2014a, 2010; Do et al., 2011; Hidey and McKeown, 2016; Beamer and Girju, 2009; Hu et al., 2017; Hu and Walker, 2017b). Recently, some benchmarks on the event causality have been released. Mirza et al. (2014), Mirza and Tonelli (2016) extracted causal relation of events with a rule-based multi-sieve approach incorporating with event temporal relation. Mirza and Tonelli (2014) annotated the Causal-TimeBank of event causal relations. Caselli and Vossen (2017) annotated the EventStoryLine Corpus for event causality identification in 320 short stories based on the temporal and causal relations annotated dataset (Mostafazadeh et al., 2016). Dunietz et al. (2017) presented BECauSE 2.0, a new version of the BECauSE (Dunietz et al., 2015) of causal relation and other seven relations.

Based on the above benchmarks, Gao et al. (2019) modeled document-level structures to identify the causalities of events. Liu et al. (2020) identified event causalities with the mention masking generalization and external KBs. Zuo et al. (2020) improved the performance of ECI with the distantly automatically labeled training data. However, these methods only rely on a small scale of labeled data. In this paper, we introduce external causal statements to help identify event causalities.

Self-Supervised Representation Learning Self-supervised representation learning cares about producing good features generally helpful for many tasks (Weng, 2019). Wu et al. (2018) proposed MemoryBank, which stores representations of all the data and samples a random set of keys as negative examples. He et al. (2020) provided a framework, MoCo, of unsupervised learning visual representation as a dynamic dictionary

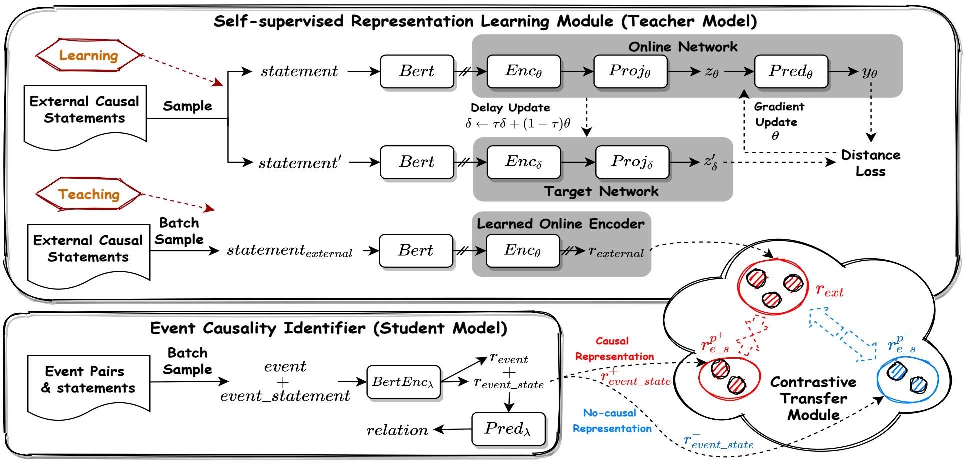

Figure 2: The learning and transfer processes of the proposed CauSeRL for ECI. "//" means stop-gradient.

look-up. Chen et al. (2020) proposed the SimCLR which learns representations for visual inputs by maximizing agreement between differently augmented views of the same sample via a contrastive loss. Grill et al. (2020) claimed a novel representation learning framework relies on two neural networks, BYOL, without using negative samples. CURL (Srinivas et al., 2020) applies the above ideas in reinforcement learning. Inspired by them, we design a self-supervised framework to learn context-specific causal patterns from external causal statements and adopt a contrastive transfer strategy to incorporate them into target ECI model.

3 Methodology

As shown in Figure 2, the whole pipeline process of CauSeRL is divided into two major stages.

- **Self-supervised causal representation learning** (**SelfRL**, Sec. 3.1). In this stage, we design a *self-supervised representation learning module* to learn enhanced causal representations by iteratively sampling two external causal statements, taking each of them as a target to learn their commonalities which reflect context-specific causal patterns.

- **Contrastive representation transfer** (**ConRT**, Sec. 3.2). In this stage, we employ a *contrastive transfer module* to transfer the learned context-specific causal patterns into the ECI target model, the *event causality identifier*, via incorporating the enhanced causal representations from SelfRL.

3.1 Self-Supervised Causal Representation Learning (SelfRL)

SelfRL aims to train a module that masters context-specific causal patterns from external causal statements by learning their enhanced causal representation with a self-supervised framework.

Self-Supervised Representation Learning Module We design a self-supervised module to capture the context-specific causal patterns from external causal statements via learning their enhanced causal representation. However, there are no ECI-specific event annotations in the external causal statements, which makes them unable to be directly used as training data to train the ECI model. To handle this problem, inspired by Grill et al. (2020), we iteratively sample two external causal statements, take each of them as a target to learn their commonalities, that is, the causal representations, which reflect context-specific causal patterns.

In specific, as shown in Figure 2, we configure two networks for SelfRL, an online network, and a target network. The target network provides regression targets to train the online network which makes it learn the commonalities among two input causal statements, that is, the causal representations reflecting different context-specific causal patterns. Structurally, the online network is defined as a set of weights θ which is comprised of three submodules: an *encoder* Enc_θ, a *projector* $Proj_\theta$ and a *predictor* $Pred_\theta$. And the target network has the same architecture as the online network, but no predictor and uses a different set of weights δ.

In specific, we iteratively sample two external causal statements, initially encode them by BERT (Devlin et al., 2019), and input them into two networks respectively. After encoding and projection, the online network and target network respectively output a projection $\boldsymbol{z}_\theta$ and $\boldsymbol{z}'_\delta$. Then the online network outputs a prediction $\boldsymbol{y}_\theta$, and takes the following mean square error between ℓ_2-normalized $\bar{\boldsymbol{y}}_\theta$ and $\bar{\boldsymbol{z}}'_\delta$ as the training objective to learn the commonalities of two causal statements, that are regarded as the context-specific causal patterns.

$$\mathcal{L}_{\theta,\delta} \triangleq \left\|\bar{\boldsymbol{y}}_\theta - \bar{\boldsymbol{z}}'_\delta\right\|_2^2 = 2 - 2 \cdot \frac{\langle \boldsymbol{y}_\theta, \boldsymbol{z}'_\delta \rangle}{\|\boldsymbol{y}_\theta\|_2 \cdot \left\|\boldsymbol{z}'_\delta\right\|_2}, \tag{1}$$

$$\bar{\boldsymbol{y}}_\theta \triangleq \boldsymbol{y}_\theta / \|\boldsymbol{y}_\theta\|_2, \bar{\boldsymbol{z}}'_\delta \triangleq \boldsymbol{z}'_\delta / \left\|\boldsymbol{z}'_\delta\right\|_2. \tag{2}$$

To reduce the bias, we symmetrize the $\mathcal{L}_{\theta,\delta}$ by swapping the input causal statements of the online and target networks to compute $\widetilde{\mathcal{L}}_{\theta,\delta}$.

Learning of SelfRL For the learning of SelfRL, at each step, as shown in Algorithm 1, we minimize the $\mathcal{L}^{tea}_{\theta,\delta}$ to stochastic gradient update the online network respect to the parameters θ only. For the target network, the parameters δ are an exponential moving average of the parameters θ of the online network (Lillicrap et al., 2016):

$$\mathcal{L}^{tea}_{\theta,\delta} = \mathcal{L}_{\theta,\delta} + \widetilde{\mathcal{L}}_{\theta,\delta}, \tag{3}$$

$$\theta \leftarrow \eta_{tea} \nabla_\theta \mathcal{L}^{tea}_{\theta,\delta}, \tag{4}$$

$$\delta \leftarrow \tau\delta + (1-\tau)\theta, \tag{5}$$

where, η_{tea} is the learning rate of the online network, and $\tau \in [0,1]$ is the decay rate that determines the degree of the movement of θ to δ. As shown in Figure 2, when learning, BERT is only used to provide an initial representation for the input statements, and its parameters are not updated.

According to the theoretical analysis by Grill et al. (2020), the addition of a predictor on the online network and the usage of a slow-moving average of the online parameters as the target network encourage SelfRL to encode a more informative causal representation of commonalities within the online projection and avoids collapsed solutions[1].

3.2 Contrastive Representation Transfer (ConRT)

ConRT aims to incorporate the context-specific causal patterns learned in SelfRL from external causal statements into the identifier. As aforementioned, the goal of SelfRL is learning the commonalities among different external causal statements, which does not make the representation learning module have the ability to distinguish the causal and non-causal statements directly. Therefore, we employ a contrastive transfer module to teach the learned context-specific causal patterns to the event causality identifier for training.

Algorithm 1 Two stages training of CauSeRL.

```
Require: External causal statements $\mathcal{C}$ for teacher model
    and event pairs with statements $\mathcal{P}$ for student model.
Training:
Stage: CAUSAL REPRESENTATION LEARNING
    for each batch $\mathcal{C}_{bat} \in \mathcal{C}$ do        ▷ Learning of SelfRL
        for any two causal statements $\in \mathcal{C}_{bat}$ do
            One for online another for target;
            Get $y_\theta$ from $Pred_\theta$ in online network;
            Get $z'_\delta$ from $Proj_\delta$ in target network;
            Swap two statements into two networks;
            Get symmetrical $y_\theta$ and $z'_\delta$;
            Compute $\mathcal{L}_{\theta,\delta}$ and $\widetilde{\mathcal{L}}_{\theta,\delta}$;
        end for
        Compute batch $\mathcal{L}^{tea}_{\theta,\delta}$ in equation (3);
        Stochastic gradient update $\theta$ in equation (4);
        Slow-moving update $\delta$ in equation (5);
    end for
end Stage:

Stage: CONTRASTIVE REPRESENTATION TRANSFER
    for each batch $\mathcal{P}_{bat} \in \mathcal{P}$ do  ▷ Learning of identifier
        for any event pair with statement $\in \mathcal{P}_{bat}$ do
            Get $\boldsymbol{r}_{event}$ and $\boldsymbol{r}_{event\_state}$ from $BertEnc_\lambda$;
            Predict the causality of two events in one pair;
        end for
        Compute batch $\mathcal{L}^{stu}_\lambda$ in equation (6);
        Sample $\mathcal{C}_{bat} \in \mathcal{C}$;
        Get $\boldsymbol{r}_{external}$ of $c \in \mathcal{C}_{bat}$ from learned $Enc_\theta$;
        Get $\boldsymbol{r}^+_{event\_state}$, $\boldsymbol{r}^-_{event\_state}$ from $\boldsymbol{r}_{event\_state}$;
        Get mapped $\boldsymbol{r}^{p^+}_{e\_s}$, $\boldsymbol{r}^{p^-}_{e\_s}$ and $\boldsymbol{r}_{ext}$;
        Compute $\mathcal{L}_\lambda = \mathcal{L}^{stu}_\lambda + \mathcal{L}^{con}_\lambda$ in equation (8);
        Stochastic gradient update $\lambda$ in equation (9);
    end for
end Stage:
```

Event Causality Identifier Event causality identification is formulated as a sentence level binary classification problem. Specifically, we design a classifier based on BERT (Devlin et al., 2019) to build our identifier. The input is an event pair and its statement. As shown in Figure 2, we take representation of events $\boldsymbol{r}_{event}$ and their contextual statement $\boldsymbol{r}_{event\_state}$ encoded by $BertEnc_\lambda$ as the input of top MLP predictor. Finally, the output is a binary vector to indicate the causal relation of the input two events expressed by their statement. The parameters of the identifier are defined as λ and the optimization function is the following

---

[1] In this paper, collapse solution means that the model encodes all input statements as the same representation. The slow-moved target network keeps the predictor of the online network always near-optimal, thus avoiding the collapse.

Resource	Original Causal Statement Form	Converted Causal Statement Form
GLU-SPE	Billy finds his childhood teddy bear $> Cause/Enable >$ Billy gives his childhood teddy bear to his daughter	Billy finds his childhood teddy bear, billy gives his childhood teddy bear to his daughter.
GLU-GEN	Someone_A finds Something_A $> Cause/Enable >$ Someone_A gives Something_A to Someone_B	Someone_A finds Something_A, Someone_A gives Something_A to Someone_B.
ATOMIC	PersonX follows PersonY into room $> oWant >$ to know why PersonX is following them	PersonX follows PersonY into room, to know why PersonX is following them.
DISTANT	Fisk was shot to death by his mistress's new lover and Fisk's ex-business partner.	Fisk was shot to death by his mistress's new lover and Fisk's ex-business partner.

Table 1: The original and converted form (the input form of SelfRL) of different causal statements from three resources. GLU-SPE and GLU-GEN denote the specific and general statements from GLUCOSE respectively.

classification cross-entropy function:

$$\mathcal{L}_\lambda^{stu} = \textsc{CrossE}(\text{MLP}([\boldsymbol{r}_{event}; \boldsymbol{r}_{event\_state}])). \quad (6)$$

Contrastive Transfer Module As aforementioned, inspired by Tian et al. (2020), we employ a contrastive transfer strategy to transfer the *"knowledge"* mastered by the teacher (self-supervised representation learning module), that is the context-specific causal patterns, to the student (event causality identifier), which helps the latter to identify the event causalities. The key idea of contrastive transfer is intuitional: maximize the mutual information between the teacher and the student (Tian et al., 2020). Methodologically, we make the representation of the statements of causal events encoded by the student model should be close to the causal representation grasped by the teacher model. By contrast, we keep the representation of the statements of non-causal events away from it.

As shown in Figure 2, at each training step of identifier, we sample a batch of external causal statements into the learned Enc_θ of the online network to obtain their causal representation $\boldsymbol{r}_{ext}$ for teaching. At the same time, we also sample a batch of event pairs with their statements into the $BertEnc_\lambda$ of identifier to obtain the statement representation $\boldsymbol{r}_{event\_state}$ of each event pair. Among one batch, $\boldsymbol{r}_{event\_state}$ consists of the $\boldsymbol{r}^+_{event\_state}$ of causal event pairs and the $\boldsymbol{r}^-_{event\_state}$ of non-causal event pairs. After mapping $\boldsymbol{r}_{external}$, $\boldsymbol{r}^+_{event\_state}$ and $\boldsymbol{r}^-_{event\_state}$ into a same space, we obtain $\boldsymbol{r}_{ext}$, $\boldsymbol{r}^{p^+}_{e\_s}$ and $\boldsymbol{r}^{p^-}_{e\_s}$ respectively. After that, we make $\boldsymbol{r}^{p^+}_{e\_s}$ be close to $\boldsymbol{r}_{ext}$ in the contrastive loss function:

$$\mathcal{L}_\lambda^{con} = \frac{1}{|\mathcal{P}^+|} \sum_{p^+ \in \mathcal{P}^+} \log \frac{e^{(\mathcal{D}(\boldsymbol{r}^{p^+}_{e\_s}, \boldsymbol{r}_{ext})/T)}}{\sum_{p \in P} e^{(\mathcal{D}(\boldsymbol{r}^{p}_{e\_s}, \boldsymbol{r}_{ext})/T)}}, \quad (7)$$

where, P^+ and P are the causal event pairs and all event pairs in one batch respectively, T is a temperature that adjusts the concentration level, and $\mathcal{D}$ is the ℓ_2-distance function to measure the distance of two representation.

Learning of Event Causality Identifier For the training of event causality identifier, we add contrastive loss to the basic classification loss, which could guide the identifier to learn context-specific causal patterns implied in the enhanced causal representation from SelfRL. As shown in Algorithm 1, we minimize the L_λ and stochastic gradient update the λ as following:

$$\mathcal{L}_\lambda = \mathcal{L}_\lambda^{stu} + \mathcal{L}_\lambda^{con}, \quad (8)$$

$$\lambda \leftarrow \eta_{stu} \nabla_\lambda \mathcal{L}_\lambda, \quad (9)$$

where, η_{stu} is the learning rate of the identifier. For evaluation, we predict the causality of input event pair without the contrastive transfer module. Additionally, the T in $\mathcal{L}_\lambda^{con}$ indirectly plays a role in adjusting the influence weight of $\mathcal{L}_\lambda^{stu}$ and $\mathcal{L}_\lambda^{con}$. In specific, for teaching, we take the learned Enc_θ of the online network as the encoder, freeze its parameters, to provide the enhanced causal representation of the external causal statements for contrastive representation transfer.

4 Experiments

4.1 Experimental Setup

Dataset and Evaluation Metrics for ECI Our experiments are conducted on two main benchmarks, including: **EventStoryLine** v0.9 (ESC) (Caselli and Vossen, 2017) described above; and (2) **Causal-TimeBank** (CTB) (Mirza and Tonelli, 2014) which contains 184 documents, 6813 events, and 318 causal event pairs. Same as previous methods, we use the last two topics of ESC as the development set for two datasets. For evaluation, we adopt Precision (P), Recall (R), and F1-score (F1) as evaluation metrics. We conduct 5-fold and 10-fold cross-validation on ESC and CTB respectively, same as previous methods. All the results are the average of three independent experiments.

Data Preparation for Self-Supervised Causal Representation Learning We take four types of external causal statements from three resources. Table 1 illustrates the original form and the converted input form of SelfRL (Sec. 3.1) of the causal statements from three different resources.

- **GLUCOSE** (Mostafazadeh et al., 2020): a large-scale dataset of implicit commonsense knowledge, encoded as causal explanatory mini-theories inspired by cognitive psychology. Each GLUCOSE explanation is stated both as a specific statement (grounded in a given context, **GLU-SPE** in Table 1) and a corresponding general rule (applicable to other contexts, **GLU-GEN** in Table 1).

- **ATOMIC** (Sap et al., 2018): an atlas of machine commonsense, as a step toward addressing the rich spectrum of inferential knowledge that is crucial for commonsense reasoning.

- **DISTANT** (Zuo et al., 2020): the automatically labeled training data for ECI via distant supervision that expresses the causal semantics between events.

Parameters Settings In implementations, all the BERT modules are implemented on BERT-Base architecture[2], which has 12-layers, 768-hiddens, and 12-heads. We employ the one-layer BiLSTM (Hochreiter and Schmidhuber, 1997) as Enc_θ and Enc_δ. For parameters, we set the learning rate of SelfRL (η_{tea}) and identifier (η_{stu}) as 1e-5 and 2e-5 respectively. The size of the space in the contrastive transfer module and the hidden layer of BiLSTM are both set as 50. And we respectively set the decay rate τ of moving average in SelfRL and the temperature of the contrastive loss $\mathcal{L}_\lambda^{con}$ are 0.996 and 0.1 tuned on the development set. Moreover, we also tune the batch size of SelfRL and identifier as 48 and 16 respectively on the development set. And we apply the early stop and AdamW gradient strategy to optimize all models. We also adopt a negative sampling rate of 0.6 for the training of identifier, owing to the sparseness of positive examples in the ECI datasets.

Methods	**P**	**R**	**F1**
EventStoryLine			
S-Path (Cheng and Miyao, 2017)	34.0	41.5	37.4
S-Fea (Choubey and Huang, 2017)	32.7	44.9	37.8
LR+ (Gao et al., 2019)	37.0	45.2	40.7
ILP (Gao et al., 2019)	37.4	55.8	44.7
BERT	36.0	56.8	44.1
KnowDis (Zuo et al., 2020)	39.7	66.5	49.7
MasG (Liu et al., 2020)	41.9	62.5	50.1
KnowDis+CauSeRL (**Ours**)	40.1	68.9	50.7*
MasG+CauSeRL (**Ours**)	40.8	68.0	51.0*
CauSeRL$_{DISTANT}$ (**Ours**)	39.9	67.3	50.1*
CauSeRL$_{ATOMIC}$ (**Ours**)	41.0	68.1	51.2*
CauSeRL$_{GLU\text{-}GEN}$ (**Ours**)	41.4	67.8	51.4*
CauSeRL$_{GLU\text{-}SPE}$ (**Ours**)	**41.9**	**69.0**	**52.1***
Causal-TimeBank			
Rule-B (Mirza and Tonelli, 2014)	36.8	12.3	18.4
Data-D (Mirza and Tonelli, 2014)	67.3	22.6	33.9
VerR-C (Mirza, 2014)	**69.0**	31.5	43.2
BERT	39.5	44.5	41.9
MasG (Liu et al., 2020)	36.6	55.6	44.1
KnowDis (Zuo et al., 2020)	42.3	60.5	49.8
MasG+CauSeRL (**Ours**)	42.6	62.5	50.7*
KnowDis+CauSeRL (**Ours**)	42.5	66.0	51.7*
CauSeRL$_{DISTANT}$ (**Ours**)	41.6	63.9	50.4*
CauSeRL$_{ATOMIC}$ (**Ours**)	42.8	67.0	52.2*
CauSeRL$_{GLU\text{-}GEN}$ (**Ours**)	43.0	66.8	52.3*
CauSeRL$_{GLU\text{-}SPE}$ (**Ours**)	43.6	**68.1**	**53.2***

Table 2: Results of event causality identification on two benchmarks. Bold denotes best results; * denotes a significant test at the level of 0.05;

Compared Methods Same as previous methods. For ESC, we prefer 1) **S-Path** (Cheng and Miyao, 2017), a dependency path based sequential method that models the context between events to identify causality; 2) **S-Fea** (Choubey and Huang, 2017), a sequence model explores complex human designed features for ECI; 3) **LR+** and **ILP** (Gao et al., 2019), document-level models adopt document structures for ECI.

For CTB, we prefer 1) **Rule-B**, a rule-based system; 2) **Data-D**, a data driven machine learning based system; 3) **VerR-C**, a verb rule based model with data filtering and causal signals enhancement. These models are designed by Mirza and Tonelli (2014; 2014) for ECI. For both two datasets, 1) we build a baseline **BERT** (our basic proposed event causality identifier); 2) We prefer **MasG** (Liu et al., 2020), a BERT-Large based SOTA model with mention masking generalization; 3) **KnowDis** (Zuo et al., 2020) improved the performance of ECI with the distantly labeled training data.

To make a fair comparison, we employ CauSeRL to retrain MasG and KnowDis to illustrate the effectiveness of our proposed approach for ECI on other methods. In specific, 1) **MasG+CauSeRL**: we retrain MasG with L_λ^{con} based on the CLU-SPE. To be consistent with other BERT-based compared models, we re-construct MasG based on BERT-Base rather than the original BERT-Large of MasG;

---

[2] `https://github.com/google-research/bert`

Methods	**P**	**R**	**F**	∇
CauSeRL$_{GLU\text{-}SPE}$	**41.9**	**69.0**	**52.1***	-
$Enc_{\theta-init}$ + ConRT	39.1	63.6	48.4*	-3.7
$BertEnc_{\lambda-init}$ + ConRT	38.9	63.1	48.1*	-4.0
BERT	36.0	56.8	44.1	-
BERT+SelfRL$_{finetune}$	38.5	60.9	47.2*	+3.1

Table 3: Ablation results of the self-supervised causal representation learning (SelfRL, Sec. 3.1) of ECI on EventStoryLine. * denotes a significant test at the level of 0.05; ∇ means the points lower than CauSeRL or higher than BERT in the upper and lower parts respectively; $Enc_{\theta-init}$ + ConRT denotes a varietal CauSeRL that removes SelfRL, directly employs an initial Enc_θ of the online network to encode external causal statements into ConRT and trains it meanwhile; $BertEnc_{\lambda-init}$ + ConRT denotes a varietal CauSeRL that removes SelfRL, directly employs a same initial $BertEnc_\lambda$ of identifier to encode external causal statements into ConRT and trains it meanwhile; BERT+SelfRL$_{finetune}$ denotes a varietal CauSeRL that removes ConRT (Sec. 3.2), and takes the learned Enc_θ of the online network as the initial encoder of identifier on the BERT baseline model.

Methods	**P**	**R**	**F**	∇
CauSeRL$_{GLU\text{-}SPE}$	**41.9**	**69.0**	**52.1***	-
$Enc_{\theta-freeze}$ + SelfRL	37.8	59.9	46.4*	-5.7
$Enc_{\theta-finetune}$ + SelfRL	38.5	60.9	47.2*	-4.9
BERT	36.0	56.8	44.1	-
BERT + ConRT$_{Enc_\theta}$	39.1	63.6	48.4*	+4.3

Table 4: Ablation results of the contrastive representation transfer (ConRT, Sec. 3.2) of ECI on EventStoryLine. * denotes a significant test at the level of 0.05; ∇ means the points lower than CauSeRL or higher than BERT in the upper and lower parts respectively; $Enc_{\theta-freeze}$ + SelfRL denotes a varietal CauSeRL that removes ConRT, and takes the frozen learned Enc_θ of the online network as the encoder of identifier; $Enc_{\theta-finetune}$ denotes a varietal CauSeRL that removes ConRT, and takes the learned Enc_θ of the online network as the initial encoder of identifier; BERT + ConRT$_{Enc_\theta}$ denotes a varietal CauSeRL that removes SelfRL (Sec. 3.1), directly employs an initial Enc_θ of the online network to encode external causal statements into ConRT and trains it meanwhile.

2) **KnowDis+CauSeRL**: we regard the automatically distantly labeled causal sentences generated by KnowDis as causal statements to learn in SelfRL, and transfer to KnowDis.

CauSeRL$_{External\text{-}Statement}$: To further illustrate the ability of CauSeRL to learn the context-specific causal patterns for the ECI task, we make CauSeRL learn from four types of external causal statements shown in Table 1 for identifying the causalities between events. $External\text{-}Statement$ denotes what kind of external causal statements.

4.2 Our Method vs. State-of-the-art Methods

Table 2 shows the results of ECI on EventStoryLine and Causal-TimeBank. From the results:

1) Our CauSeRL outperforms all baseline methods and achieves the best performance on F1 value, 52.1% on ESC and 53.2% on CTB respectively. Specifically, CauSeRL outperforms the no-bert (ILP/VerR-C) and bert (MasG/KnowDis) baseline methods by a margin of 7.4%/10.0% and 2.0%/3.4% on two benchmarks respectively. It illustrates the context-specific causal patterns from external causal statements are effective for ECI.

2) Comparing MasG+CauSeRL with MasG, we note that even with BERT-Base, the performance of MasG+CauSeRL is significantly higher than that of MasG based on BERT-Large. This shows that the context-specific causal patterns learned by CauSeRL from external causal statements can effectively alleviate the limitation of mask generalization only relying on limited labeled causal context.

3) Comparing KnowDis+CauSeRL with KnowDis, we find that CauSeRL could more efficiently make use of the automatically labeled causal statements, which learns their context-specific causal patterns to further enhance the ability of models to identify the causalities between events.

4) Comparing different external causal statements. a) GLU-SPE brings the most significant improvement because the specific causal statements from GLU-SPE have complete text structures that are more similar to ECI labeled data and make models easier to learn. There, all the ablation experiments are conducted on GLU-SPE. b) The effects of GLU-GEN and ATOMIC are similar because these two types of statements are abstract causal structures. Although they are similar to the context-specific causal patterns, it is relatively difficult to understand directly. c) The improvement brought by DISTANT is relatively small because of the effects of the noise from distantly labeled data.

5) Comparing CauSeRL with MasG+CauSeRL, we notice that after removing the ConceptNet knowledge enhancement employed by MasG, the external causal statements could be better learned and transferred. This is because MasG directly flattens the event concept knowledge into the statement sequence, which disrupts the statement structure and affects the understanding of the statement.

6) It is worth noting that the improvement on

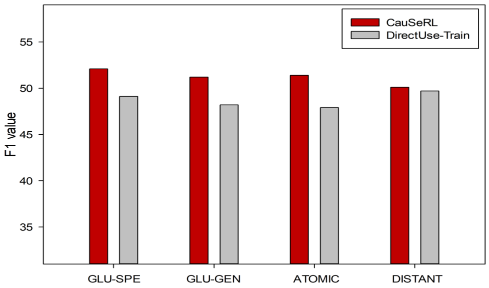

Figure 3: Results of event causality identification on EventStoryLine that directly using external causal statements as the training data of ECI task.

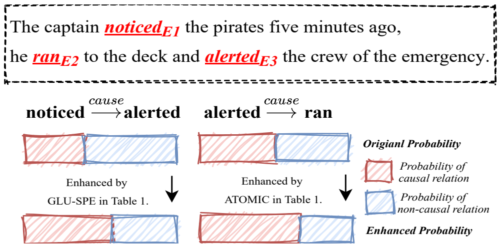

Figure 4: Case study of the probability changes with external causal statements enhancement.

the CTB is higher than that of the ESC, because the amount of labeled data of the former is relatively small, and more need for the help of external causal statements. Moreover, compared with the traditional methods based on features or rules, all BERT-based methods demonstrate high recall value, which is benefited from more training data, knowledge and causal statements.

4.3 Effect of Self-Supervised Causal Representation Learning

We analyze the effect of the self-supervised causal representation learning (SelfRL, Sec. 3.1). As shown in Table 3, from the results, 1) after removing SelfRL, the performance of ECI significantly decreases. This illustrates that the context-specific causal patterns learned by SelfRL are important for the ECI model to understand the causality. 2) Comparing BERT+SelfRL$_{finetune}$ with BERT, the Enc_θ that has learned from external causal statements could improve the performance of ECI to a certain extent. This illustrates that SelfRL could effectively capture the context-specific causal patterns in the statements for identification. 3) Comparing $Enc_{\theta-init}$ + ConRT and $BertEnc_{\lambda-init}$ + ConRT, after representation learning, the fine-tuned Enc_θ could further improve the performance of ECI. This indirectly shows that the context-specific causal patterns learned in the SelfRL is generalized.

4.4 Effect of Contrastive Representation Transfer

We analyze the effect of the contrastive representation transfer (ConRT, Sec. 3.2). As shown in Table 4, from the results, 1) after removing ConRT, the performance of ECI also significantly decreases. This illustrates that the learned causal representations from external statements are not suitable for direct application to ECI, and needs to be effectively transferred that the ConRT focuses on. 2) Comparing BERT + ConRT$_{Enc_\theta}$ with BERT, even if causal representation learning is not carried out in advance, adopting contrast strategy to directly transfer the context-specific causal patterns could also help the inference of event causality to a certain extent. 3) Comparing $Enc_{\theta-freeze}$ + SelfRL with $Enc_{\theta-finetune}$ + SelfRL, we find that the causal representations encoded by pre-trained BERT and BiLSTM have similar effects. Aforementioned, to avoid collapse solutions (Sec. 3.1), we choose the BiLSTM as an encoder in SelfRL that could be initialized completely independently.

4.5 Effect of the Utilization of External Causal Statement

As shown in Figure 3, we regard external causal statements as positive training data for ECI and directly use them to train the BERT baseline model. In specific, we treat two words that play a predicate role in the syntactic structure of each statement as events. From the results, CauSeRL could more effectively make use of causal statements to help understand the causalities of events. In contrast, directly serving as training data is not effective.

4.6 Case Study

As shown in Figure 4, with limited labeled data, the model could not understand the causal relation between event *noticed* and event *alerted*. Fortunately, with the support of the context-specific causal pattern from GLU-SPE in Table 1, the prediction is modified correctly. Moreover, the original model that only trained with limited labeled data is ambiguous about the causal relation between event *alerted* and event *ran*. Influenced by the similar causal statements with the example in Table 1 from ATOMIC, the prediction confidence is improved.

5 Conclusion

We propose a novel approach, CauSeRL, which could leverage external causal statements to identify the causalities of events. First of all, we design a self-supervised framework to learn context-specific causal patterns from external causal statements. Then, we adopt a contrastive transfer strategy to incorporate the learned context-specific causal patterns into the target ECI model for identification. Experimental results on two benchmarks show that our model achieves the best performance.

Acknowledgments

We thank anonymous reviewers for their insightful comments and suggestions. This work is supported by the National Key Research and Development Program of China (No. 2017YFB1002101), the National Natural Science Foundation of China (No.61922085, 61976211). This work is also supported by Beijing Academy of Artificial Intelligence (BAAI2019QN0301) and the joint project with Beijing Baidu Netcom Science Technology Co., Ltd.

References

Brandon Beamer and Roxana Girju. 2009. Using a bigram event model to predict causal potential. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 430–441. Springer.

Tommaso Caselli and Piek Vossen. 2017. The event StoryLine corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86, Vancouver, Canada. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–6, Vancouver, Canada. Association for Computational Linguistics.

Prafulla Kumar Choubey and Ruihong Huang. 2017. A sequential model for classifying temporal relations between intra-sentence events. pages 1796–1802.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Quang Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 294–303, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2015. Annotating causal language using corpus lexicography of constructions. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 188–196, Denver, Colorado, USA. Association for Computational Linguistics.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017. The BECauSE corpus 2.0: Annotating causality and overlapping relations. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 95–104, Valencia, Spain. Association for Computational Linguistics.

Lei Gao, Prafulla Kumar Choubey, and Ruihong Huang. 2019. Modeling document-level causal structures for event causal relation identification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1808–1817, Minneapolis, Minnesota. Association for Computational Linguistics.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 76–83, Sapporo, Japan. Association for Computational Linguistics.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), pages 987–997, Baltimore, Maryland. Association for Computational Linguistics.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning.

Christopher Hidey and Kathy McKeown. 2016. Identifying causal relations using parallel Wikipedia articles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1424–1433, Berlin, Germany. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Zhichao Hu, Elahe Rahimtoroghi, and Marilyn Walker. 2017. Inference of fine-grained event causality from blogs and films. pages 52–58.

Zhichao Hu and Marilyn Walker. 2017a. Inferring narrative causality between event pairs in films. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 342–351, Saarbrücken, Germany. Association for Computational Linguistics.

Zhichao Hu and Marilyn A Walker. 2017b. Inferring narrative causality between event pairs in films. *arXiv preprint arXiv:1708.09496*.

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

Jian Liu, Yubo Chen, and Jun Zhao. 2020. Knowledge enhanced event causality identification with mention masking generalizations. In *IJCAI-20*, pages 3608–3614. International Joint Conferences on Artificial Intelligence Organization. Main track.

Paramita Mirza. 2014. Extracting temporal and causal relations between events. pages 10–17.

Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. Annotating causality in the TempEval-3 corpus. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19, Gothenburg, Sweden. Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2097–2106, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2016. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 64–75, Osaka, Japan. The COLING 2016 Organizing Committee.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California. Association for Computational Linguistics.

Nasrin Mostafazadeh, Aditya Kalyanpur, Lori Moon, David Buchanan, Lauren Berkowitz, Or Biran, and Jennifer Chu-Carroll. 2020. GLUCOSE: GeneraLized and COntextualized story explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4569–4586, Online. Association for Computational Linguistics.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1733–1743, Sofia, Bulgaria. Association for Computational Linguistics.

Jong-Hoon Oh, Kentaro Torisawa, Canasai Kruengkrai, Ryu Iida, and Julien Kloetzer. 2017. Multi-column convolutional neural networks with causality-attention for why-question answering. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 415–424. ACM.

Mehwish Riaz and Roxana Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. In *2010 IEEE Fourth International Conference on Semantic Computing*, pages 361–368. IEEE.

Mehwish Riaz and Roxana Girju. 2014a. In-depth exploitation of noun and verb semantics to identify causation in verb-noun pairs. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 161–170, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Mehwish Riaz and Roxana Girju. 2014b. Recognizing causality in verb-noun pairs via noun and verb semantics. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 48–57, Gothenburg, Sweden. Association for Computational Linguistics.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2018. ATOMIC: an atlas of machine commonsense for if-then reasoning. *CoRR*, abs/1811.00146.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. 2020. Curl: Contrastive unsupervised representations for reinforcement learning.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive representation distillation. In *International Conference on Learning Representations*.

Lilian Weng. 2019. Self-supervised representation learning. *lilianweng.github.io/lil-log*.

Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. 2018. Unsupervised feature learning via nonparametric instance-level discrimination.

Xinyu Zuo, Yubo Chen, Kang Liu, and Jun Zhao. 2020. KnowDis: Knowledge enhanced data augmentation for event causality detection via distant supervision. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1544–1550, Barcelona, Spain (Online). International Committee on Computational Linguistics.

PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval

Ruiyang Ren[1,3*†], **Shangwen Lv**[2*], **Yingqi Qu**[2], **Jing Liu**[2‡], **Wayne Xin Zhao**[3,4‡]
Qiaoqiao She[2], **Hua Wu**[2], **Haifeng Wang**[2] **and Ji-Rong Wen**[3,4]
[1]School of Information, Renmin University of China; [2]Baidu Inc.
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Gaoling School of Artificial Intelligence, Renmin University of China
{reyon.ren, jrwen}@ruc.edu.cn, batmanfly@gmail.com
{lvshangwen, quyingqi, liujing46, sheqiaoqiao, wu_hua, wanghaifeng}@baidu.com

Abstract

Recently, dense passage retrieval has become a mainstream approach to finding relevant information in various natural language processing tasks. A number of studies have been devoted to improving the widely adopted dual-encoder architecture. However, most of the previous studies only consider query-centric similarity relation when learning the dual-encoder retriever. In order to capture more comprehensive similarity relations, we propose a novel approach that leverages both query-centric and **PA**ssage-centric s**I**milarity **R**elations (called **PAIR**) for dense passage retrieval. To implement our approach, we make three major technical contributions by introducing formal formulations of the two kinds of similarity relations, generating high-quality pseudo labeled data via knowledge distillation, and designing an effective two-stage training procedure that incorporates passage-centric similarity relation constraint. Extensive experiments show that our approach significantly outperforms previous state-of-the-art models on both MSMARCO and Natural Questions datasets[1].

1 Introduction

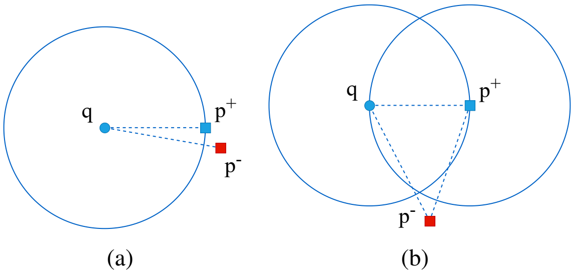

Figure 1: An illustrative case of a query q, its positive passage p^+ and negative passage p^-: (a) Query-centric similarity relation enforces $s(q, p^+) > s(q, p^-)$; (b) Passage-centric similarity relation further enforces $s(p^+, q) > s(p^+, p^-)$, where $s(p^+, q) = s(q, p^+)$. We use the distance (*i.e.,* dissimilarity) for visualization: the longer the distance is, the less similar it is.

With the recent advances of pre-trained language models, dense passage retrieval techniques (representing queries and passages in low-dimensional semantic space) have significantly outperformed traditional term-based techniques (Guu et al., 2020; Karpukhin et al., 2020). As the key step of finding the relevant information, it has been shown that dense passage retrieval can effectively improve the performance in a variety of tasks, including question answering (Lee et al., 2019; Xiong et al., 2020b), information retrieval (Luan et al., 2021; Khattab and Zaharia, 2020), dialogue (Ji et al., 2014; Henderson et al., 2017) and entity linking (Gillick et al., 2019; Wu et al., 2020).

Typically, the dual-encoder architecture is used to learn the dense representations of queries and passages, and the dot-product similarity between the representations of queries and passages becomes ranking measurement for retrieval. A number of studies have been devoted to improving this architecture (Guu et al., 2020; Karpukhin et al., 2020; Xiong et al., 2020a) for dense passage retrieval. Previous studies mainly consider learning query-centric similarity relation, where it tries to increase the similarity $s(q, p^+)$ between a query and a positive (*i.e.,* relevant) passage meanwhile decrease the similarity $s(q, p^-)$ between the query and a negative (*i.e.,* irrelevant) passage. We argue that query-centric similarity relation ignores the relation between passages, and it brings difficulty to discriminate between positive and negative passages. To illustrate this, we present an example in

---

[*] Equal contribution.

[†] The work was done when Ruiyang Ren was doing internship at Baidu.

[‡] Corresponding authors.

[1] Our code is available at `https://github.com/PaddlePaddle/Research/tree/master/NLP/ACL2021-PAIR`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2173–2183
August 1–6, 2021.

Figure 1, where a query q and two passages p^+ and p^- are given. As we can see in Figure 1(a), although query-centric similarity relation can enforce $s(q,p^+) > s(q,p^-)$ and identify the positive passages in this case, the distance (*i.e.,* dissimilarity) between positive and negative passages is small. When a new query is issued, it is difficult to discriminate between positive passage p^+ and negative passage p^-.

Considering this problem, we propose to further learn passage-centric similarity relation for enhancing the dual-encoder architecture. The basic idea is shown in Figure 1(b), where we set an additional similarity relation constraint $s(p^+,q) > s(p^+,p^-)$: the similarity between query q and positive passage p^+ should be larger than that between positive passage p^+ and negative passage p^-. In this way, it is able to better learn the similarity relations among query, positive passages and negative passages. Although the idea is appealing, it is not easy to implement due to three major issues. First, it is unclear how to formalize and learn both query-centric and passage-centric similarity relations. Second, it requires large-scale and high-quality training data to incorporate passage-centric similarity relation. However, it is expensive to manually label data. Additionally, there might be a large number of unlabeled positives even in the existing manually labeled datasets (Qu et al., 2020), and it is likely to bring false negatives when sampling hard negatives. Finally, learning passage-centric similarity relation (an auxiliary task) is not directly related to the query-centric similarity relation (a target task). In terms of multi-task viewpoint, multi-task models often perform worse than their single-task counterparts (Alonso and Plank, 2017; McCann et al., 2018; Clark et al., 2019). Hence, it needs a more elaborate design for the training procedure.

To this end, in this paper, we propose a novel approach that leverages both query-centric and **PA**ssage-centric s**I**milarity **R**elations (called **PAIR**) for dense passage retrieval. In order to address the aforementioned issues, we have made three important technical contributions. First, we design formal loss functions to characterize both query-centric and passage-centric similarity relations. Second, we propose to generate pseudo-labeled data via knowledge distillation. Third, we devise a two-stage training procedure that utilizes passage-centric similarity relation during pre-training and then fine-tunes the dual-encoder according to the task goal. The improvements in the three aspects make it possible to effectively leverage both kinds of similarity relations for improving dense passage retrieval.

The contributions of this paper can be summarized as follows:

- We propose an approach that simultaneously learns query-centric and passage-centric similarity relations for dense passage retrieval. It is the first time that passage-centric similarity relation has been considered for this task.

- We make three major technical contributions by introducing formal formulations, generating high-quality pseudo-labeled data and designing an effective training procedure.

- Extensive experiments show that our approach significantly outperforms previous state-of-the-art models on both MSMARCO and Natural Questions datasets.

2 Related Work

Recently, dense passage retrieval has demonstrated better performance than traditional sparse retrieval methods (*e.g.,* TF-IDF and BM25). Different from sparse retrieval, dense passage retrieval represents queries and passages into low-dimensional vectors (Guu et al., 2020; Karpukhin et al., 2020), typically in a dual-encoder architecture, and uses dot product as the similarity measurement for retrieval. The existing approaches for dense passage retrieval can be divided into two categories: (1) unsupervised pre-training for retrieval (2) fine-tuning only on labeled data.

In the first category, different pre-training tasks for retrieval were proposed. Lee et al. (2019) proposed a specific approach to pre-training the retriever with an unsupervised task, namely Inverse Cloze Task (ICT), and then jointly fine-tuned the retriever and a reader on labeled data. REALM (Guu et al., 2020) proposed a new pre-training approach, which jointly trained a masked language model and a neural retriever. Different from them, our proposed approach utilizes the pseudo-labeled data via knowledge distillation in the pre-training stage, and the quality of the generated data is high (see Section 4.6).

In the second category, the existing approaches fine-tuned pre-trained language models on labeled data (Karpukhin et al., 2020; Luan et al., 2021). Both DPR (Karpukhin et al., 2020) and ME-BERT (Luan et al., 2021) used in-batch random sampling and hard negative sampling by BM25, while ANCE (Xiong et al., 2020a), NPRINC (Lu et al., 2020) and RocketQA (Qu et al., 2020) explored more sophisticated hard negative sampling approach. Izacard and Grave (2020) and Yang et al. (2020) leveraged a reader and a cross-encoder for knowledge distillation on labeled data, respectively. RocketQA found large batch size can significantly improve the retrieval performance of dual-encoders. ColBERT (Khattab and Zaharia, 2020) incorporated light-weight attention-based re-ranking while increasing the space complexity.

The existing studies mainly focus on learning the similarity relation between the queries and the passages, while ignoring the relation among passages. It makes the model difficult to discriminate the positive passages and negative passages. In this paper, we propose an approach simultaneously learn query-centric and passage-centric similarity relations.

3 Methodology

In this section, we present an approach that leverages both query-centric and **PA**ssage-centric s**I**milarity **R**elations (called **PAIR**) for dense passage retrieval.

3.1 Overview

The task of dense passage retrieval (Karpukhin et al., 2020) is described as follows. Given a query q, we aim to retrieve k most relevant passages $\{p_j\}_{j=1}^{k}$ from a large collection of M passages.

For this task, the dual-encoder architecture is widely adopted (Karpukhin et al., 2020; Qu et al., 2020), where two separate encoders $E_Q(\cdot)$ and $E_P(\cdot)$ are used to represent the query q and the passage p into d-dimensional vectors in different representation spaces. Then a dot product is performed to measure the similarity between q and p based on their embeddings:

$$s(q,p) = E_Q(q)^\top \cdot E_P(p). \tag{1}$$

Previous studies mainly capture the query-centric similarity relation. As shown in Figure 1, passage-centric similarity relation reflects important evidence for improving the retrieval performance. Therefore, we extend the original query-centric learning framework by leveraging the passage-centric similarity relation.

To develop our approach, we need to address the issues described in Section 1, and we consider three aspects to extend. First, we design a new loss function that considers both query-centric and passage-centric similarity relations. Second, we utilize knowledge distillation to obtain large-scale and high-quality pseudo-labeled data to capture more comprehensive similarity relations. Third, we design a two-stage training procedure to effectively learn the passage-centric similarity relation and improve the final retrieval performance.

3.2 Defining the Loss Functions

Our approach considers two kinds of losses, namely query-centric loss and passage-centric loss, as shown in Figure 2. The two kinds of losses are characterized by the two different similarity relations, query-centric similarity relation and passage-centric similarity relation.

Query-centric Loss The query-centric similarity relation regards the query q as the center and pushes the negative passages p^- farther than the positive passages p^+. That is:

$$s^{(Q)}(q,p^+) > s^{(Q)}(q,p^-)\,, \tag{2}$$

where $s^{(Q)}(q,p^+)$ and $s^{(Q)}(q,p^-)$ represent the similarities for the relevant and irrelevant passages to query q, and they are defined the same as $s(q,p)$ in Eq. (1). Following (Karpukhin et al., 2020; Qu et al., 2020), we learn the query-centric similarity relation by optimizing query-centric loss that is the negative log likelihood of the positive passage:

$$L_Q = -\frac{1}{N}\sum_{\langle q,p^+\rangle} \log \frac{e^{s^{(Q)}(q,p^+)}}{e^{s^{(Q)}(q,p^+)} + \sum_{p^-} e^{s^{(Q)}(q,p^-)}}. \tag{3}$$

As shown in Figure 1, for a given query, there might exist some negative passages similar to the positive passage, making it difficult to discriminate between positive and negative passages. Hence, we further incorporate passage-centric loss to address this issue.

Passage-centric Loss The aim of learning passage-centric similarity relation is to push negative passage p^- farther from positive passage p^+, and making the similarity between positive passage p^+ and query q larger than the similarity between positive passage p^+ and negative passage

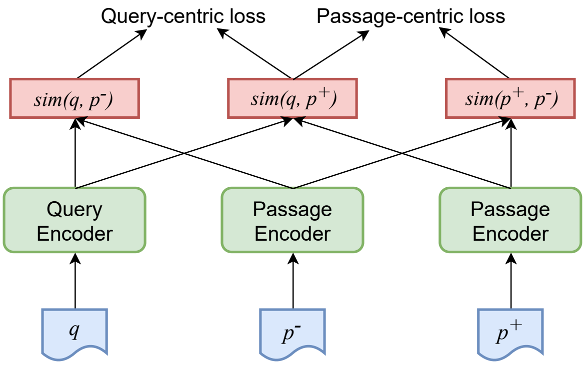

Figure 2: An illustration of the combination of query-centric loss and passage-centric loss.

p^-. Formally, we introduce the following passage-centric similarity relation:

$$s^{(P)}(p^+, q) > s^{(P)}(p^+, p^-), \tag{4}$$

where $s^{(P)}(p^+, q)$ and $s^{(P)}(p^+, p^-)$ are defined as $E_P(p^+)^\top \cdot E_Q(q)$ and $E_P(p^+)^\top \cdot E_P(p^-)$, respectively. Similarly, we learn the passage-centric similarity relation by optimizing the passage-centric loss function that is the negative log likelihood of the query:

$$L_P = -\frac{1}{N} \sum_{\langle q, p^+ \rangle} \log \frac{e^{s^{(P)}(p^+, q)}}{e^{s^{(P)}(p^+, q)} + \underline{\sum_{p^-} e^{s^{(P)}(p^+, p^-)}}}. \tag{5}$$

By comparing Eq. (3) and Eq. (5), we can observe that the difference in two kinds of loss lies in the normalization part (underlined).

The Combined Loss We present an illustrative sketch of the above two loss functions in Figure 2. Next, we propose to simultaneously learn both query-centric and passage-centric similarity relations in Eq.(2) and Eq.(4). Therefore, we combine query-centric and passage-centric loss functions defined in Eq. (3) and (5) to obtain the final loss function:

$$L = (1 - \alpha) * L_Q + \alpha * L_P, \tag{6}$$

where α is a hyper-parameter and is tuned in experiments. By considering passage-centric similarity relation, our approach will be more capable of discriminating between a positive passage and a highly similar yet irrelevant passage $\big($See Figure 1(b)$\big)$.

Dual-encoder with Shared Parameters Most of the existing studies (Eq. (2)) equip the dual-encoders with two separate encoders (E_Q and E_P) for queries and passages, respectively. In this case, different encoders may project queries and passages into two different spaces. However, to simultaneously model the query-centric similarity relation and the passage-centric similarity relation, the representations of queries and passages should be in the same space. Otherwise, the similarity between passages and the similarity between queries and passages are not comparable. Therefore, we propose using the encoders that share the same parameters and structures for both queries and passages, *i.e.,* $E_Q(\cdot)$=$E_P(\cdot)$.

3.3 Generating the Pseudo-labeled Training Data via Knowledge Distillation

By optimizing both query-centric loss and passage-centric loss, we can capture more comprehensive similarity relations. However, more similarity relation constraints require large-scale and high-quality training data for optimization. Additionally, there might be a large number of unlabeled positives even in the existing manually labeled datasets (Qu et al., 2020), and it is likely to bring false negatives when sampling hard negatives. Hence, we propose to generate pseudo-labeled training data via knowledge distillation.

Cross-encoder Teacher Model The teacher model is used to generate large-scale pseudo-labeled data. Following RocketQA (Qu et al., 2020), we adopt the cross-encoder architecture to implement the teacher, which takes as input the concatenation of query and passage and models the semantic interaction between query and passage representations. Such an architecture has been demonstrated to be more effective than the dual-encoder architecture in characterizing query-passage relevance (Yang et al., 2020). We follow Qu et al. (2020) to train the cross-encoder teacher with the labeled data.

Generating Pseudo Labels In this paper, we follow Qu et al. (2020) to obtain positives and hard negatives[2] for unlabeled queries[3]. First, we retrieve the top-*k* candidate passages of unlabeled queries from the corpus by an efficient retriever DPR (Karpukhin et al., 2020), and score them by the well-trained cross-encoder (*i.e.,* teacher model). We set two values s_{pos} and s_{neg} ($s_{pos} > s_{neg}$) as the positive and hard negative thresholds,

---

[2] Xiong et al. (2020a) and Karpukhin et al. (2020) demonstrate the importance of hard negatives.

[3] We obtain easy negatives from in-batch sampling.

Dataset	#q in train	#q in dev	#q in test	#p
MSMARCO	502,939	6,980	6.837	8,841,823
Natural Questions	58,812	6,515	3,610	21,015,324

Table 1: The detailed statistics of MSMARCO and Natural Questions. Here, "q" and "p" are the abbreviations of queries and passages, respectively.

respectively. Then, given each query, a candidate passage with a score above s_{pos} or below s_{neg} will be considered as positive or negative. Note that we also apply this on labeled corpus to obtain more positives and reliable hard negatives. Because there might be a large number of unlabeled positives even in the existing manually labeled datasets (Qu et al., 2020) and it is likely to bring false negatives in hard negative sampling.

3.4 Two-stage Training Procedure

Although passage-centric similarity relation (Eq. (5)) is able to incorporate additional relevance evidence, it is not directly related to the final task goal (*i.e.,* query-centric similarity relation). Therefore, we design a two-stage training procedure that incorporates the passage-centric loss in the pre-training stage, and then only optimize the tasks-specific loss (*i.e.,* query-centric loss) in the fine-tuning stage. We present an illustration for the two-stage training procedure in Figure 3. Next, we present the detailed training procedure.

Pre-training In the pre-training stage, we train the dual-encoder by optimizing the loss function L in Eq. (6) (*i.e.,* a combination of query-centric loss and passage-centric loss). The pseudo-labeled data from unlabeled corpus is adopted as the pre-training data (Section 3.3).

Fine-tuning In the fine-tuning stage, we only fine-tune the dual-encoder (pre-trained in the first stage) according to the query-centric loss L_Q in Eq. (3). In this way, our approach focuses on learning the task-specific loss, yielding better retrieval performance. In this stage, we use both ground-truth labels and pseudo labels derived from the labeled corpus for training.

4 Experiments

In this section, we first describe the experimental settings, then report the main experimental results, ablation study and detailed analysis.

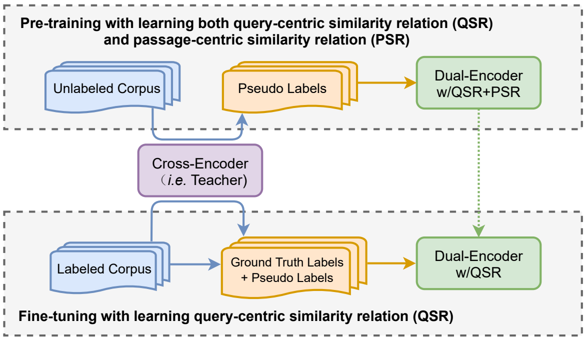

Figure 3: Overview of the proposed two-stage method.

4.1 Experimental Settings

Datasets This paper focuses on the passage retrieval task. We conduct experiments on two public datasets: MSMARCO (Nguyen et al., 2016) and Natural Questions (Kwiatkowski et al., 2019). The statistics of the datasets are listed in Table 1. *MSMARCO* was originally designed for multiple passage machine reading comprehension, and its queries were sampled from Bing search logs. Based on the queries and passages in MSMARCO Question Answering, a dataset for passage retrieval and ranking was created, namely MSMARCO Passage Ranking. *Natural Questions (NQ)* was originally introduced as a dataset for open-domain QA. The queries were collected from Google search logs. DPR (Karpukhin et al., 2020) selected the queries that had short answers, and processed all the Wikipedia articles as the collection of passages. In our experiments, we reuse the version of NQ created by DPR.

Evaluation Metrics Following previous work, we adopt Mean Reciprocal Rank (MRR) and Recall at top k ranks (Recall@k) to evaluate the performance of passage retrieval. MRR calculates the averaged reciprocal of the rank at which the first positive passage is retrieved. Recall@k calculates the proportion of questions to which the top k retrieved passages contain positives.

Unlabeled Corpus To obtain the augmenta-

Methods	PLM	MSMARCO Dev			Natural Questions Test		
		MRR@10	R@50	R@1000	R@5	R@20	R@100
BM25 (anserini) (Yang et al., 2017)	-	18.7	59.2	85.7	-	59.1	73.7
doc2query (Nogueira et al., 2019b)	-	21.5	64.4	89.1	-	-	-
DeepCT (Dai and Callan, 2019)	-	24.3	69.0	91.0	-	-	-
docTTTTTquery (Nogueira et al., 2019a)	-	27.7	75.6	94.7	-	-	-
GAR (Mao et al., 2020)	-	-	-	-	-	74.4	85.3
DPR (single) (Karpukhin et al., 2020)	$\text{BERT}_{\text{base}}$	-	-	-	-	78.4	85.4
DPR-E	$\text{ERNIE}_{\text{base}}$	32.5	82.2	97.3	68.4	80.7	87.3
ANCE (single) (Xiong et al., 2020a)	$\text{RoBERTa}_{\text{base}}$	33.0	-	95.9	-	81.9	87.5
ME-BERT (Luan et al., 2021)	$\text{BERT}_{\text{large}}$	34.3	-	-	-	-	-
NPRINC (Lu et al., 2020)	$\text{BERT}_{\text{base}}$	31.1	-	97.7	73.3	82.8	88.4
ColBERT (Khattab and Zaharia, 2020)	$\text{BERT}_{\text{base}}$	36.0	82.9	96.8	-	-	-
RocketQA (Qu et al., 2020)	$\text{ERNIE}_{\text{base}}$	37.0	85.5	97.9	74.0	82.7	88.5
PAIR (Ours)	$\text{ERNIE}_{\text{base}}$	**37.9**	**86.4**	**98.2**	**74.9**	**83.5**	**89.1**

Table 2: Experimental results on MSMARCO and Natural Questions datasets. Note that we copy the results from original papers and we leave it blank if the original paper does not report the result.

tion data, we collect about 1.8 million unlabeled queries from Yahoo! Answers[4], ORCAS (Craswell et al., 2020), SQuAD (Rajpurkar et al.), TriviaQA (Joshi et al., 2017) and HotpotQA (Yang et al., 2018). In the pre-training stage, we reuse the passage collections from the labeled corpus (MSMARCO and NQ).

4.2 Implementation Details

We conduct experiments with the deep learning framework PaddlePaddle (Ma et al., 2019) on up to eight NVIDIA Tesla V100 GPUs (with 32G RAM).

Pre-trained LMs The dual-encoder is initialized with the parameters of ERNIE-2.0 base (Sun et al., 2020). ERNIE-2.0 has the same networks as BERT (Devlin et al., 2019), and it introduces a continual pre-training framework on multiple pretrained tasks. The cross-encoder setting follows the cross-encoder in RocketQA (Qu et al., 2020)

Hyper-parameters (a) *batch size*: Our dual-encoder is trained with a batch size of 512×1 in fine-tuning stage on NQ and 512×8 in other settings. We use the in-batch negative setting (Karpukhin et al., 2020) on NQ and cross-batch negative setting (Qu et al., 2020) on MSMARCO. (b) *training epochs*: The number of training epochs is set up to 10 for both pre-training and fine-tuning for dual-encoder. (c) *warm-up and learning rate*: The learning rate of the dual-encoder is set to 3e-5 and the rate of linear scheduling warm-up is set to 0.1. (d) *# of positives and hard negatives*: The ratio of the positive to the hard negative is set to 1:4 on dual-encoder.

Optimizers We use LAMB optimizer (You et al., 2020) to train the dual-encoder on MSMARCO, which is more suitable in cross-batch negative setting. In other settings, we always use ADAM optimizer (Kingma and Ba, 2015).

The choice of alpha α is a hyper-parameter to balance the query-centric loss and passage-centric loss (Eq. (6)). We searched for α from 0 to 1 by setting an equal interval to 0.1, and the model achieves the best performance when α is set to 0.1.

4.3 Main Experimental Results

We consider both sparse and dense passage retrievers for baselines. The sparse retrievers include the traditional retriever BM25 (Yang et al., 2017), and four traditional retrievers enhanced by neural networks, including doc2query (Nogueira et al., 2019b), DeepCT (Dai and Callan, 2019), docTTTTTquery (Nogueira et al., 2019a) and GAR (Mao et al., 2020). Both doc2query and docTTTTTquery employ neural query generation to expand documents. In contrast, GAR employs neural generation models to expand queries. Different from them, DeepCT utilizes BERT to learn the term weight. The dense passage retrievers include DPR (Karpukhin et al., 2020), DPR-E, ANCE (Xiong et al., 2020a), ME-BERT (Luan et al., 2021), NPRINC (Lu et al., 2020), ColBERT (Khattab and Zaharia, 2020) and RocketQA (Qu et al., 2020). DPR-E is our implementation of DPR using ERNIE (Sun et al., 2020)

---

[4] `http://answers.yahoo.com/`

Methods	R@5	R@20	R@100
Complete (PAIR)	**74.9**	**83.5**	**89.1**
w/o PSR	73.6	83.3	88.8
w/o KD	70.9	82.7	88.1
w/ PSR FT	74.6	83.4	89.0
w/o SP	74.0	83.4	88.9
w/o PT	73.0	82.8	88.5

Table 3: The ablation study and controlled experiments of different variants of PAIR on Natural Questions.

instead of BERT, to examine the effects of pre-trained LMs.

Table 2 presents the main experimental results.

(1) We can see that PAIR significantly outperforms all the baselines on both MSMARCO and NQ datasets. The major difference between our approach and baselines lies in that we incorporate both query-centric and passage-centric similarity relations, which can capture more comprehensive semantic relations. Meanwhile, we incorporate the augmented data via knowledge distillation.

(2) We notice that baseline methods use different pre-trained LMs, as shown in the second column of Table 2. In PAIR, we use the ERNIE-base. To examine the effects of ERNIE-base, we implement DPR-E by replacing BERT-base used in DPR as ERNIE-base. From Table 2, we can observe that PAIR significantly outperforms DPR-E, although they employ the same pre-trained LM.

(3) Another observation is that the dense retrievers are overall better than the sparse retrievers. Such a finding has also been reported in prior studies (Karpukhin et al., 2020; Xiong et al., 2020a; Luan et al., 2021), which indicates the effectiveness of the dense retrieval approach.

4.4 Ablation Study

In this section, we conduct ablation study to examine the effectiveness of each strategy in our proposed approach. We only report the results on the NQ, while the results on the MSMARCO are similar and omitted here due to limited space.

Here, we consider five variants based on our approach for comparison:

(a) *w/o PSR* removes the loss for passage-centric similarity relation in the pre-training stage;

(b) *w/o KD* removes the knowledge distillation for obtaining pseudo-labeled data and only uses the labeled data (MSMARCO and NQ) for both pre-training stage and fine-tuning stage;

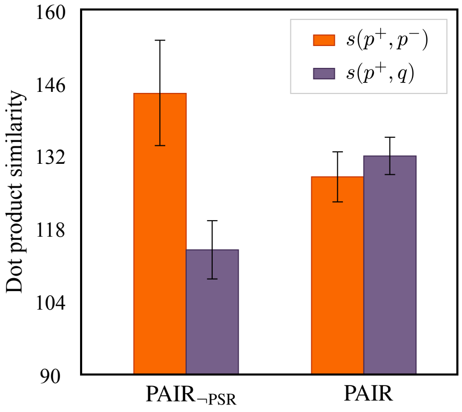

Figure 4: The comparison of PAIR and PAIR$_{\neg\text{PSR}}$ on $s(p^+, p^-)$ and $s(p^+, q)$ with standard deviation.

(c) *w/ PSR FT* adds the loss for passage-centric similarity relation in the fine-tuning stage;

(d) *w/o SP* uses separate encoders for queries and passages instead of encoders with shared parameters;

(e) *w/o PT* removes the pre-training stage.

Table 3 presents the results for the ablation study. We can observe the following findings:

- The performance drops in *w/o PSR*, demonstrating the effectiveness of learning passage-centric similarity relation;
- The performance drops in *w/o KD*, demonstrating the necessity and effectiveness of the knowledge distillation for obtaining large-scale and high-quality pseudo-labeled data, since the passage-centric loss tries to distinguish highly similar but semantically different passages;
- The performance slightly drops in *w/ PSR FT*, because passage-centric loss is not directly related to the target task (*i.e.,* query-based retrieval), which suggests that passage-centric loss should be only used in the pre-training stage;
- The performance drops in *w/o SP*, demonstrating the effectiveness of dual-encoders with shared parameters;
- The performance significantly drops in *w/o PT*, demonstrating the importance of our pre-training procedure.

4.5 Analysis on Passage-centric Similarity Relation

The previous results demonstrate the effectiveness of our proposed approach PAIR. Here, we further analyze the effect of passage-centric loss (Eq. (5)) in a more intuitive way. To examine this, we prepare two variants of our approach,

Query	Top 1 passage retrieved by PAIR (correct)	Top 1 passage retrieved by $\text{PAIR}_{\neg\text{PSR}}$ (incorrect)
Which animal is the carrier of the **H1N1** virus ?	**H1N1** strains caused a small percentage of all human flu infections in 2004–2005. Other strains of **H1N1** are endemic *in pigs* (swine influenza) and in birds (avian influenza) ...	**H5N1** is a subtype virus which can cause illness in humans and many other animal species. A bird-adapted strain of **H5N1**, called HPAIA (**H5N1**) for ...
Where is **gall bladder** situated in human body?	The **gall bladder** is a small hollow organ where bile is stored ... In humans, the pear-shaped **gall bladder** lies *beneath the liver*, although the structure and position ...	The **urinary bladder** is a hollow muscular organ in humans and some other animals that collects and stores urine from the kidneys before disposal by urination ...

Table 4: The comparison of the top-1 passages retrieved by PAIR and $\text{PAIR}_{\neg\text{PSR}}$, respectively. The **bold words** represent the main topics in queries and passages. The *italic words with wavy underline* are the right answers. The words with straight underline among passages have many words in common and may mislead the model $\text{PAIR}_{\neg\text{PSR}}$ to select the wrong passage.

namely the complete PAIR and the variant removing the passage-centric loss (Eq. (5)) denoted by $\text{PAIR}_{\neg\text{PSR}}$.

We first analyze how the passage-centric similarity relation (PSR) influences the similarity relations among query, positive passage and negative passage. Figure 4 shows the comparison of PAIR and $\text{PAIR}_{\neg\text{PSR}}$ for computing the similarities of $s(p^+, p^-)$ and $s(p^+, q)$. We obtain $s(p^+, p^-)$ and $s(p^+, q)$ by the averaging the similarity of top 100 retrieved passages for each query in the testing data of Natural Questions. We can see that before incorporating passage-centric similarity relation (PSR), $s(p^+, p^-)$ is higher than $s(p^+, q)$. As a result, the negatives are close to the positives. After incorporating PSR, $s(p^+, p^-)$ becomes lower than $s(p^+, q)$. It indicates that passage-centric loss pulls positive passages closer to queries and push them farther away from negative passages in the representation space. The comparison result is consistent with passage-level similarity relation in Eq. (4).

Next, we further present two examples in Table 4 to understand the performance difference between PAIR and $\text{PAIR}_{\neg\text{PSR}}$. In the first example, the top-1 passage retrieved by PAIR has the same topic "H1N1" as the query. In contrast, the top-1 passage retrieved by $\text{PAIR}_{\neg\text{PSR}}$ has an incorrect but highly relevant topic "H5N1". Actually, the sentences among the positive passage (retrieved by PAIR) and the negative passage (retrieved by $\text{PAIR}_{\neg\text{PSR}}$) share many common words. Such a negative passage is likely to mislead the retriever to yield incorrect rankings. Hence, these two passages should be far away from each other in the representation space. This problem cannot be well solved by only considering the query-passage similarity in existing studies. Similar observations can be find from the second example. The top-1 passage retrieved by PAIR has the same topic "gall bladder" as the query, while the top-1 passage retrieved by $\text{PAIR}_{\neg\text{PSR}}$ is about "urinary bladder". These results show that passage-centric similarity relations are particularly useful to discriminate between positive and hard negative passages (highly similar to positive passages).

Threshold	Data Quality		Retrieval Performance		
	Acc_{pos}	Acc_{neg}	R@5	R@20	R@100
$s_{pos} = 0.9$ $s_{neg} = 0.1$	**92%**	**96%**	**74.9**	**83.5**	**89.1**
$s_{pos} = 0.8$ $s_{neg} = 0.2$	90%	93%	74.5	83.4	88.9
$s_{pos} = 0.7$ $s_{neg} = 0.3$	84%	87%	73.6	**83.5**	88.6
$s_{pos} = 0.6$ $s_{neg} = 0.4$	80%	87%	73.5	83.4	88.7

Table 5: The data quality and retrieval performance in different thresholds on NQ. Acc_{pos} denotes accuracy of positives and Acc_{neg} denotes accuracy of negatives.

4.6 Analysis on Knowledge Distillation

In this section, we examine the influence of the thresholds on pseudo-labeled data via knowledge distillation, including the data quality and the retrieval performance. We conduct the analyses by using different positive thresholds s_{pos} and negative thresholds s_{neg} (See Section 3.3).

We first manually evaluate the quality of the pseudo-labeled data via knowledge distillation *w.r.t.* different threshold settings (*i.e.,* the combinations of s_{neg} and s_{pos}). For each threshold setting, we randomly select 100 queries, each of which corresponding to a positive passage and a hard-negative passage. In total, we have 4 threshold settings (as shown in Table 5) and 800 query-passage pairs. We ask two experts to manually annotate the query-passage pairs and evaluate the quality of pseudo-labeled data, the Cohen's Kappa of experts is 0.9. As shown in the first two columns of Table 5, we can observe that when $s_{pos} = 0.9$ and $s_{neg} = 0.1$, the data quality is relatively good. Additionally, when setting a low value of s_{pos} and a high value of s_{neg}, the data quality becomes worse.

The last three columns of Table 5 also present the retrieval performance *w.r.t.* different threshold settings. When choosing a low value of s_{pos} and a high value of s_{neg}, the retrieval performance drops. Hence, our approach is configured with a strict threshold setting (s_{pos} = 0.9, s_{neg} = 0.1) in experiments to achieve good performance.

5 Conclusion and Future Work

This paper presented a novel dense passage retrieval approach that leverages both query-centric and passage-centric similarity relations for capturing more comprehensive semantic relations. To implement our approach, we made three important technical contributions in the loss formulation, training data augmentation and effective training procedure. Extensive results demonstrated the effectiveness of our approach. To our knowledge, it is the first time that passage-centric similarity relation has been considered for dense passage retrieval. We believe such an idea itself is worth exploring in designing new ranking mechanism. In future work, we will design more principle ranking functions and apply current retrieval approach to downstream tasks such as question answering and passage re-ranking.

6 Ethical Impact

The technique of dense passage retrieval is effective for question answering, where the majority of questions are informational queries. Semantic crowdedness problem of passages, and term mismatch between questions and passages are typical problems, which bring barriers for the machine to accurately find the information. Our technique contributes toward the goal of asking machines to find the answer passages to natural language questions from a large collection of documents. With these advantages also come potential downsides: Wikipedia or any potential external knowledge source will probably never fully cover the breadth of user questions. The goal is still far from being achieved, and more efforts from the community is needed for us to get there.

Acknowledgement

This work is partially supported by the National Key Research and Development Project of China (No.2018AAA0101900), National Natural Science Foundation of China under Grant No. 61872369 and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098.

References

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 44–53.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. Bam! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5931–5937.

Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. ORCAS: 20 million clicked query-document pairs for analyzing search. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 2983–2989.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 985–988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego García-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*, pages 528–537.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Gautier Izacard and Edouard Grave. 2020. Distilling knowledge from reader to retriever for question answering. *CoRR*, abs/2012.04584.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096.

Jing Lu, Gustavo Hernández Ábrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2020. Neural passage retrieval with improved negative contrast. *CoRR*, abs/2010.12523.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. 2019. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020. Generation-augmented retrieval for open-domain question answering. *CoRR*, abs/2009.08553.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019a. From doc2query to doctttttquery. *Online preprint*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *CoRR*, abs/1904.08375.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *CoRR*, abs/2010.08191.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6397–6407.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020a. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *CoRR*, abs/2007.00808.

Wenhan Xiong, Xiang Lorraine Li, Srinivasan Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2020b. Answering complex open-domain questions with multi-hop dense retrieval. *CoRR*, abs/2009.12756.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 1253–1256.

Yinfei Yang, Ning Jin, Kuo Lin, Mandy Guo, and Daniel Cer. 2020. Neural retrieval for question answering with cross-attention supervised data augmentation. *CoRR*, abs/2009.13815.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Is Human Scoring the Best Criteria for Summary Evaluation?

Oleg Vasilyev, John Bohannon
Primer Technologies Inc.
San Francisco, California
`{oleg,john}@primer.ai`

Abstract

A summary quality measure is judged by how well it correlates with quality scores produced by human annotators. A higher correlation with human scores is considered to be a decisive indicator of a better measure. In this work we present observations that cast doubt on this view. We also show a possibility of an alternative indicator for selecting the best measure from a family of measures, a criterion that does not rely on human scores.

1 Introduction

The goal of summarization is to convey important and only important information of the text in a fluent, comprehensible and concise summary, preserving the factual consistency with the text.

There are several families of automated measures of summary quality. For example, Gabriel et al. (2020) classified the automated measures into four types: question-answering, text reconstruction, semantic similarity and lexical overlap. Each of these types has families of measures, for example SUM-QE (Xenouleas et al., 2019), APES (Eyal et al., 2019), Summa-QA (Scialom et al., 2019), QAEval (Deutsch et al., 2020) and FEQA (Durmus et al., 2020) in question-answering, BLANC (Vasilyev et al., 2020a) in text reconstruction, BERTScore (Zhang et al., 2020), MoverScore (Zhao et al., 2019) and SUPERT (Gao et al., 2020) in semantic similarity, ROUGE (Lin, 2004) and Jensen-Shannon (Louis and Nenkova, 2009) in lexical overlap.

A high correlation with human evaluation scores is currently accepted as the crucial criterion for choosing a good evaluation measure. Arguably, the factual faithfulness can be annotated objectively, with detailed classification of factual errors (Kryscinski et al., 2020; Huang et al., 2020; Vasilyev et al., 2020b; Gabriel et al., 2020). However, other summary qualities are subjective; this forces researchers to be careful in design and usage of human annotations (Bhandari et al., 2020; Fabbri et al., 2020; Iskender et al., 2021). Annotation scores depend on the types of texts and on the qualification of annotators. For example, there is a big difference in expert and crowd-sourced scores in (Fabbri et al., 2020)[1].

Annotators are biased in favor of anything that makes the scoring easier: extractiveness of the summary, and focus of the summary on the top part of the document (Ziegler et al., 2020). The annotation process itself differs from how the summary quality is assessed by a typical human reader. A human reader does not have a goal of scoring a summary, but rather uses the summary to guess the content of the text.

The contribution of this work:

1. We provide an example of a false 'improvement' of an automated evaluation measure: a dubious modification, imitating a human annotator behavior, can increase the correlation with human scores. For a contrast, we also provide an example of a true improvement that increases correlation with human scores for a good reason.
2. We explore an alternative criterion for selecting an optimal evaluation measure from a family of measures, the criterion not relying on human scores. We provide evidence that the criterion is robust across different kinds of texts and summaries.

For our demonstration we will use BLANC family of evaluation measures, because it is easily interpretable as an analogy to a human reader that uses the summary to guess the content of the text. Two families defined in (Vasilyev et al., 2020a) differ

---

[1] https://github.com/Yale-LILY/SummEval

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2184–2191
August 1–6, 2021.

by their setup. The BLANC-help family gets information from the summary by having the model read the summary when reading and reconstructing the text. The BLANC-tune family gets information from the summary by lightly tuning the model on the summary before reading and reconstructing the text. Measures in each of the families, BLANC-help and BLANC-tune, may differ by the parameters defining the setup.

2 Example of False Improvement: Limited Comparison with Text

After reading a summary, an annotator may chose not to review carefully the whole text, but to consider in detail only the parts that look most similar to the summary. We can imitate this by using only the part of the text that is most related to the given summary. In modifying BLANC this way, it is reasonable to expect that correlation with human scores will increase, but this would make a false 'improvement' of the measure.

In Appendix we provide two examples - Figures 6 and 7 - illustrating the bias that we seek to explore in this section. Each example has a summary that truly attempts to cover all important facts, and a summary that we intentionally wrote to cover only a very limited part of text. To create a falsely 'improved' measure, we seek to explore the bias of annotator giving more attention to parts of text most similar to the summary.

To create a biased BLANC, we can calculate BLANC separately for each sentence of the text, and select n sentences with the highest score. We can consider these selected sentences as the 'text' to deal with, and calculate BLANC on this 'text'. We create such *limited-text* BLANC from BLANC-help[2]. For our illustration we use average expert scores of 1600 text-summary pairs in the dataset SummEval (Fabbri et al., 2020).

Compared to BLANC, the limited text BLANC has indeed higher Spearman correlation with average expert, as shown by thin lines in Figure 1. In this and other figures through this section all p-values are below 0.05.

For Appendix examples, see results in Tables 1 and 2.

We can imagine a human expert paying more attention to several (say three or five) most 'promising' (most similar to the summary) sentences of the text. In evaluating relevance, this might be not very different from working with full text. But for other qualities (coherence, consistency, fluency) the correlation increases.

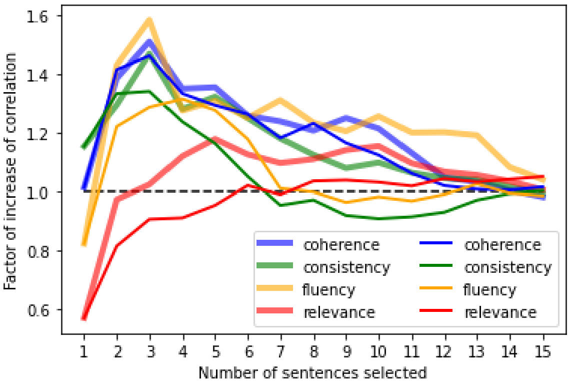

Figure 1: Factor by which Spearman correlation of BLANC with human scores increases when only part of text is used for BLANC. The text part is selected as sentences with top BLANC values (thin lines) or as contiguous sentences with highest BLANC (thick lines).

Naturally, for a human it is easier to review a contiguous piece of text rather than separated pieces, even if this might diminish legitimacy of evaluation of all qualities, including relevance. And, no surprise, BLANC for such contiguous part of text correlates with human scores even better - as shown by thick lines in Figure 1.

Figure 2 illustrates the same trends when the resulting BLANC is calculated for each selected sentence separately, and then averaged over the sentences.

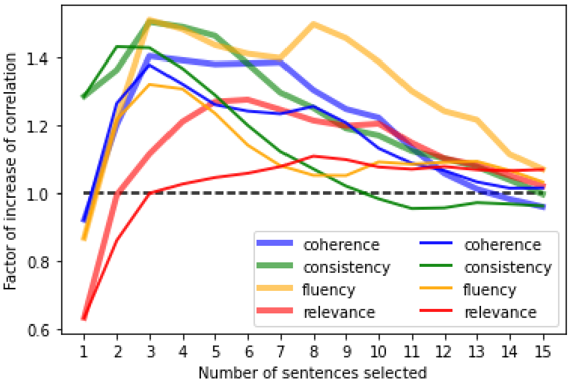

Figure 2: Factor by which Spearman correlation of BLANC with human scores increases when only part of text is used for BLANC. The text part is selected as sentences with top BLANC values (thin lines) or as contiguous sentences having highest average BLANC (thick lines). The resulting BLANC is calculated as average over BLANC of the sentences.

Figure 3 shows the increase of correlations when

---

[2] https://github.com/PrimerAI/blanc

the text is restricted not by the number of sentences but by a threshold on BLANC of a sentence.

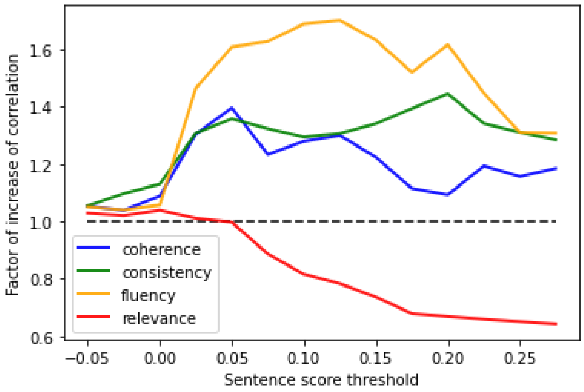

Figure 3: Factor by which Spearman correlation of BLANC with human scores increases when only part of text is used for BLANC. The text part is selected as sentences with BLANC exceeding threshold.

Selection of a part of the text is used in SUPERT multi-document evaluation measure (Gao et al., 2020) as a tool for creating 'reference' from each document and then evaluating a summary on the created references. In the context of BLANC here, the selection of a part of the text is done differently and has a clear interpretation: instead of estimating usefulness of the summary in guessing the whole text, we estimate how much the summary would help to guess only the most 'relevant' part of the text. Here 'relevant' means the part of the text for which the summary turned out to be most helpful. This is equivalent to using only the most easy (for annotator, after reading the summary) part of the text. The summary may as well relate only to a small piece of text of no importance. This means that the evaluation measure became worse, even though the correlation with human scores is stronger.

The human bias exploited by the limited-text BLANC does not necessarily manifest itself through a low inter-annotator agreement. The reported in SummEval (Fabbri et al., 2020) inter-annotator agreement of experts is 0.71, which is an acceptable value (Artstein and Poesio, 2008). While achieving a reasonable inter-annotator agreement is an important problem in human annotations, our example shows that another problem may be in the nature of the human evaluation of summary qualities, where a summary is presented to human for scoring (rather than for guessing about the text content), and the text is presented to facilitate the scoring.

3 Example of True Improvement: Learning More from Summary

In this section we provide an example of a legitimate increase of correlations with human scores, as opposed to the described in previous section false 'improvement'. We can combine BLANC-help and BLANC-tune by tuning the model on the summary (BLANC-tune), and then using the tuned model to read the summary while doing Cloze task on the text (BLANC-help).

Such *full* BLANC version is equivalent to a human that first learns the summary, and then, while guessing missed words in the text, is still looking at the summary again and again. Using both opportunities to learn from the summary makes sense, it should legitimately extract more help from the summary. The worst that may happen is that a model used by BLANC-help is already so perfect in reading the summary that its additional tuning on the summary will not improve the measure (but will not hurt either).

As expected, the full BLANC has substantially higher correlations with annotations of experts on the 1600 text-summary pairs of (Fabbri et al., 2020). Compared to BLANC-help, the Spearman correlation of full BLANC with human scores can increase by 13%-18% for coherence, 2%-3% for consistency,13%-15% for fluency, and 7%-8% for relevance, - all this depending on the number of epochs (10-20) and learning rate (1.0e-4 to 2.0e-4) of the BLANC-tune used.

4 Max-Help Criterion and its robustness

As we have seen in section 2, the correlation with human scores is not always a reliable method to select the best evaluation measure. The fact that we were able to recognize the falsity of 'improvement' in section 2 and the legitimacy of improvement in section 3 suggests that we may find a no-human criterion, at least for some setups.

In previous sections we used BLANC-help as an initial version for our modifications. As stated in Vasilyev et al. (2020b), based on the dataset introduced in there[3], BLANC-help with interval $gap = 2$ between masking locations in the text provided the highest correlations with human scores. It was noted that BLANC's average score across

---

[3] https://github.com/PrimerAI/blanc

the dataset was also the highest at this setup, implying that BLANC extracted maximal help from the summaries. Such coincidence is not a rule: the ”max-help” measure, selected for having maximal average score, is not always the same as the ”max-human” measure, selected for having maximal correlation with human scores.

The max-help criterion - selection of a measure that has highest average score - makes sense under two conditions:

1. The measure is being selected from a family of measures that have the same definition of the output score - as assessment of a usefulness of a summary. The score may be derived, for example, from how many text tokens were successfully reconstructed with help of the summary (BLANC), or from how many questions about the text were successfully answered with help of the summary (QA-based measures). The condition is that the definition is fixed for the family.
2. The average score is being measured with a large enough dataset representing the domain on which we are interested to use the measure.

The meaning of the criterion is simple: the better is the measure in extracting useful information from summaries, the better it should be in judging summaries by their usefulness. The criterion does not require human scoring. All we need is a measurement of an average score.

The max-help criterion can be credible if it does not depend too strongly on the types of texts and summaries. In order to verify this assumption, we considered four types of summaries (and the corresponding texts): (1) CNN summaries from the CNN / Daily Mail dataset (Hermann et al., 2015); (2) Daily Mail summaries from the CNN / Daily Mail dataset; (3) First two sentences from random daily news; (4) Random two sentences from random daily news.

The random daily news were selected as three random news documents per day over one year, with the summaries of the document being two first and two random sentences. We used 1000 samples for each of the four types of summaries.

We intentionally selected summaries of so different styles and quality: if the criterion selects the same best measure for so different types of summaries, then it is indeed a very robust criterion.

For BLANC-help family, we found that for all four datasets the optimal setup (accordingly to the max-help criterion) happens to be the same: $gap = 2$; minimal length of whole-word token allowed to be masked is 6 characters $L_{normal} = 6$; the word-split tokens are always masked ($L_{lead} = 1$ for first token, and $L_{follow} = 1$ for follow-up tokens).

This setup is almost the same as the parameters found in (Vasilyev et al., 2020b) to maximise correlation with human scores, except L_{normal} and L_{follow} which have low influence. The question we asked: does the ’optimal’ max-help evaluation measure remain optimal (or near-optimal) for different kinds of texts and summaries? Figure 4 provides convincing evidence for a positive answer.

Figure 4 shows the average BLANC-help value obtained with sub-optimal (different from max-help) setup. We consider a change of gap and gap_mask (number of tokens allowed to be masked at each masking location) to explore a less frequent and a more frequent masking, and a change in the token length thresholds for masking tokens. Remarkably, the average BLANC-help value drops in each case for all four datasets in a similar manner. The token length thresholds have almost no influence, making a drop just a few percents. Change in frequency of masking has a larger effect, leading to a drop 10%-20%.

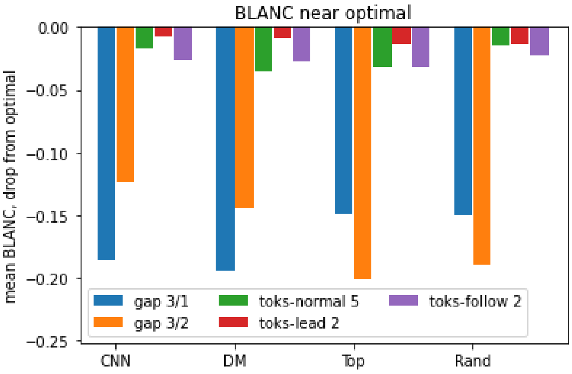

Figure 4: Drop of mean BLANC-help value when parameters differ from optimal. The drop is shown as a fraction of the optimal mean BLANC value. The summaries probed are: CNN and DM (from the CNN/Daily Mail dataset), Top and Rand (first two sentences and random two sentences from random news articles). The parameters probed are: ’gap 3/1’ is $gap = 3$ and $gap\_mask = 1$; ’gap 3/2’ is $gap = 3$ and $gap\_mask = 2$; ’toks-normal 5’ is $L_{normal} = 5$; ’toks-lead 2’ is $L_{lead} = 2$; ’toks-follow 2’ is $L_{follow} = 2$.

For BLANC-tune family, similar to BLANC-help, the max-help optimal setup is the same for all four datasets: $gap = 3$; number of tokens allowed to be masked at each masking location for inference $gap\_mask = 2$; for tuning $gap_{tune} = 4$ and $gap\_mask_{tune} = 3$; $L_{normal} = 6$; $L_{lead} = 1$; $L_{follow} = 1$; probability of replacement of a masked token by another random token at tuning $p_{replace} = 0$.

Probability $p_{replace} = 0$ differs from the value 0.1 used in the standard BERT training, but $p_{replace}$ has only weak influence on the BLANC-tune. Figure 5 shows a few examples of changes of the setup, which illustrate that the optimal measure remains optimal across all four datasets.

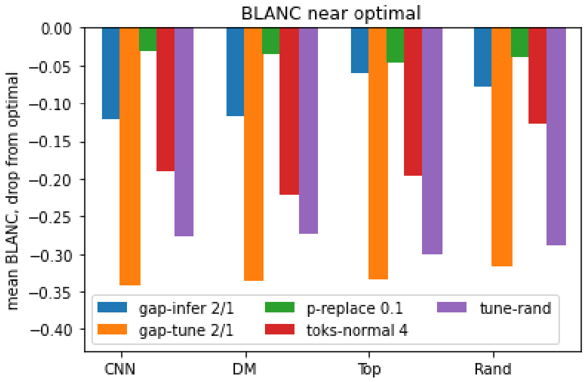

Figure 5: Drop of mean BLANC-tune value when parameters differ from optimal. The drop is shown as a fraction of the optimal mean BLANC value. The summaries probed are: CNN and DM (from the CNN/Daily Mail dataset), Top and Rand (first two sentences and random two sentences from random news articles). The parameters probed are: 'gap-infer 2/1' is $gap = 2$ and $gap\_mask = 1$; 'gap-tune 2/1' is $gap_{tune} = 2$ and $gap\_mask_{tune} = 1$; 'p-replace 0.1' is $p_{replace} = 0.1$; 'toks-normal 4' is $L_{normal} = 4$; 'tune-rand' is making tokens masking random rather than even at tuning.

The demonstrated evidence for robustness suggests that in finding an optimal measure we do not need even human summaries: we can apply the max-help criterion utilizing random sentences from the texts.

5 Conclusion

In this paper, we critically reviewed the assumption that maximal correlation with human scores defines the best evaluation measure for summarization; we provided observations supporting our scepticism. Using good interpretability of BLANC evaluation measure, we provided examples of both illegitimate 'improvement' and legitimate improvement of the correlation of BLANC scores with human scores.

We stated the motivation for an alternative criterion for choosing an optimal summary evaluation measure: the maximal average extracted usefulness of summary. We provided evidence that the criterion is robust across very different kinds of summaries, including such 'summaries' as first sentences or random sentences of the text. This means that the criterion can be applied without the need of human summaries.

While in this work we used BLANC, we think that similar observations and the same conclusions could be made using a question-answering based evaluation measure.

Acknowledgments

We thank Nidhi Vyas and Delenn Chin for review of the paper and valuable feedback.

References

Ron Artstein and Massimo Poesio. 2008. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Manik Bhandari, Pranav Narayan Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. 2020. Re-evaluating evaluation in text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 9347–9359. Association for Computational Linguistics.

Daniel Deutsch, Tania Bedrax-Weiss, and Dan Roth. 2020. Towards question-answering as an automatic metric for evaluating the content quality of a summary. *arXiv*, arXiv:2010.00490.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070. Association for Computational Linguistics.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 3938–3948. Association for Computational Linguistics.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. SummEval: Re-evaluating summarization evaluation. *arXiv*, arXiv:2007.12626v4.

Saadia Gabriel, Asli Celikyilmaz, Rahul Jha, Yejin Choi, and Jianfeng Gao. 2020. Go figure! A meta evaluation of factuality in summarization. *arXiv*, arXiv:2010.12834.

Yang Gao, Wei Zhao, and Steffen Eger. 2020. SUPERT: Towards new frontiers in unsupervised evaluation metrics for multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1347–1354. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28, pages 1693–1701. Curran Associates, Inc.

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 446–469. Association for Computational Linguistics.

Neslihan Iskender, Tim Polzehl, and Sebastian Möller. 2021. Reliability of human evaluation for text summarization: Lessons learned and challenges ahead. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 86–96. Association for Computational Linguistics (2021).

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 9332–9346. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2009. Automatically evaluating content selection in summarization without human models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 306–314. Association for Computational Linguistics.

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! Unsupervised metrics for reinforced summarization models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3246–3256. Association for Computational Linguistics.

Oleg Vasilyev, Vedant Dharnidharka, and John Bohannon. 2020a. Fill in the BLANC: Human-free quality estimation of document summaries. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 11–20. Association for Computational Linguistics.

Oleg Vasilyev, Vedant Dharnidharka, Nicholas Egan, Charlene Chambliss, and John Bohannon. 2020b. Sensitivity of BLANC to human-scored qualities of text summaries. *arXiv*, arXiv:2010.06716.

Stratos Xenouleas, Prodromos Malakasiotis, Marianna Apidianaki, and Ion Androutsopoulos. 2019. SUM-QE: A BERT-based summary quality estimation model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6005–6011. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. *arXiv*, arXiv:1904.09675v3.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 563–578. Association for Computational Linguistics.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences. *arXiv*, arXiv:1909.08593v2.

A Examples of bad judgement by falsely 'improved' measure

In Figure 6 we provide two toy summaries, each summary is supposed to summarize the first four paragraphs of 'Harry Potter' book [4]. In order to illustrate how the false improvement described in Section 2 affects ranking of summaries, we intentionally wrote one summary with 'wide coverage' (attempting to cover all the most important facts), and another summary with 'narrow coverage' (focusing on a limited part of the text).

The original BLANC, judging the summaries by how useful they are in predicting tokens of the text, gives higher score to the wide-coverage summary, - see Table 1. The falsely 'improved' BLANC - limited-text BLANC of section 2 - gives higher score to the narrow-coverage summary. It does not matter whether the small part of text covered by the second summary is important or not, the limited-text BLANC makes judgement by how well the summary covered that part of the text. In this case,

---

[4] J.K.Rowling (1997). *Harry Potter And the Sorcerer's Stone*. Bloomsbury

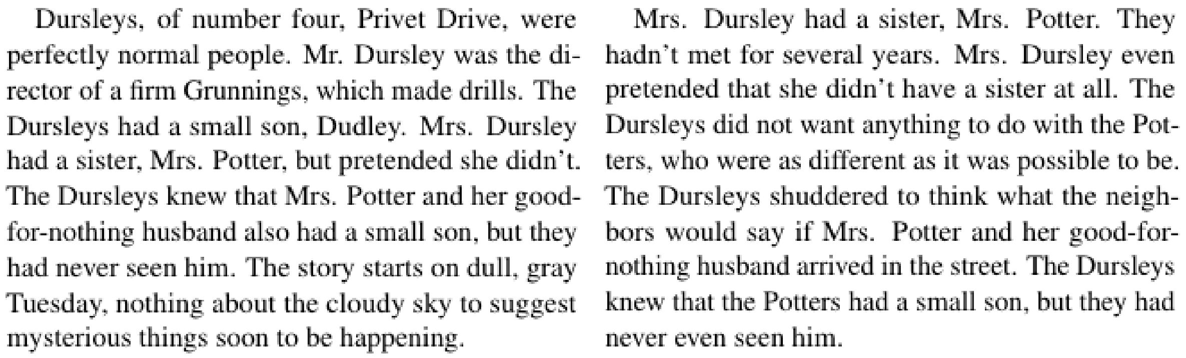

Dursleys, of number four, Privet Drive, were perfectly normal people. Mr. Dursley was the director of a firm Grunnings, which made drills. The Dursleys had a small son, Dudley. Mrs. Dursley had a sister, Mrs. Potter, but pretended she didn't. The Dursleys knew that Mrs. Potter and her good-for-nothing husband also had a small son, but they had never seen him. The story starts on dull, gray Tuesday, nothing about the cloudy sky to suggest mysterious things soon to be happening.

Mrs. Dursley had a sister, Mrs. Potter. They hadn't met for several years. Mrs. Dursley even pretended that she didn't have a sister at all. The Dursleys did not want anything to do with the Potters, who were as different as it was possible to be. The Dursleys shuddered to think what the neighbors would say if Mrs. Potter and her good-for-nothing husband arrived in the street. The Dursleys knew that the Potters had a small son, but they had never even seen him.

Figure 6: Example of a summary with a wide coverage (left) and a narrow coverage (right). Both summaries are supposed to cover first four paragraphs of 'Harry Potter And the Sorcerer's Stone' by J.K.Rowling.

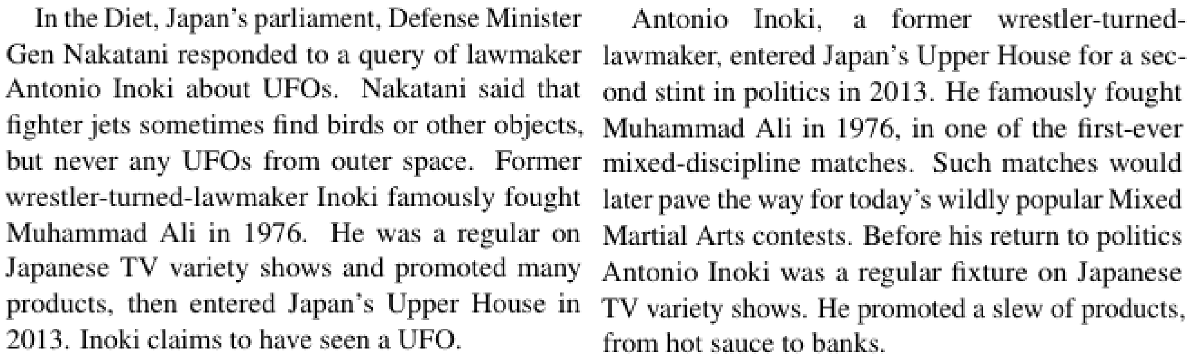

In the Diet, Japan's parliament, Defense Minister Gen Nakatani responded to a query of lawmaker Antonio Inoki about UFOs. Nakatani said that fighter jets sometimes find birds or other objects, but never any UFOs from outer space. Former wrestler-turned-lawmaker Inoki famously fought Muhammad Ali in 1976. He was a regular on Japanese TV variety shows and promoted many products, then entered Japan's Upper House in 2013. Inoki claims to have seen a UFO.

Antonio Inoki, a former wrestler-turned-lawmaker, entered Japan's Upper House for a second stint in politics in 2013. He famously fought Muhammad Ali in 1976, in one of the first-ever mixed-discipline matches. Such matches would later pave the way for today's wildly popular Mixed Martial Arts contests. Before his return to politics Antonio Inoki was a regular fixture on Japanese TV variety shows. He promoted a slew of products, from hot sauce to banks.

Figure 7: Example of a summary with a wide coverage (left) and a narrow coverage (right). Both summaries are supposed to cover the same text taken from CNN/Daily Mail dataset. The text is shown in Figure 8.

BLANC	**wide**	**narrow**
Original	0.244	0.218
Limited n=4	0.539	0.584
Limited n=3	0.563	0.652
Limited n=2	0.594	0.778

Table 1: Scores of BLANC versions for wide and narrow coverage summaries of Figure 6. Top row is the original BLANC. Lower rows are for falsely 'improved' BLANC with selection of n top text sentences, as described in Section 2.

BLANC	**wide**	**narrow**
Original	0.159	0.134
Limited n=4	0.327	0.433
Limited n=3	0.365	0.516
Limited n=2	0.428	0.536

Table 2: Scores of BLANC versions for wide and narrow coverage summaries of Figure 7. Top row is the original BLANC. Lower rows are for falsely 'improved' BLANC with selection of n top text sentences, as described in Section 2.

the second summary concentrates on the attitude of the Dursleys to the Potters, and does not provide any other information.

In Figure 7 we again provide two summaries, this time we wrote them for the text shown in Figure 8, taken from SummEval dataset (Fabbri et al., 2020). Again, as shown in Table 2, the original BLANC gives higher score to the overall more useful wide-coverage summary. The falsely 'improved' limited-text BLANC gives higher score to the narrow-coverage summary which focuses only on information about Antonio Inoki and ignores his query in the parliament.

The scores by the limited-text BLANC are higher than the scores by the original BLANC. The reason is that only text sentences with highest scores are selected, and the less the number of the sentences, the higher is the average score. Naturally, for the wide-coverage summary the score increase is not as great as for the narrow-coverage summary.

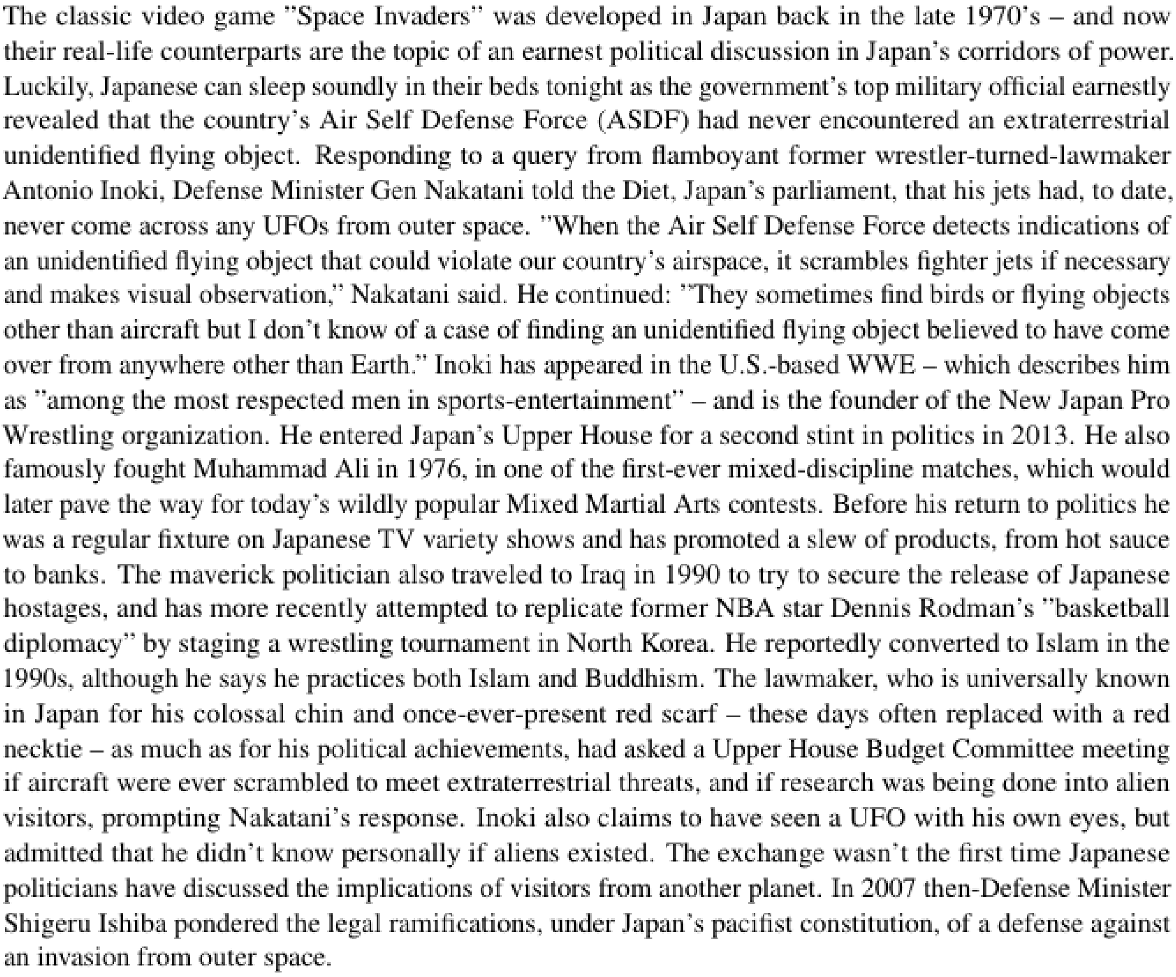

The classic video game ”Space Invaders” was developed in Japan back in the late 1970's – and now their real-life counterparts are the topic of an earnest political discussion in Japan's corridors of power. Luckily, Japanese can sleep soundly in their beds tonight as the government's top military official earnestly revealed that the country's Air Self Defense Force (ASDF) had never encountered an extraterrestrial unidentified flying object. Responding to a query from flamboyant former wrestler-turned-lawmaker Antonio Inoki, Defense Minister Gen Nakatani told the Diet, Japan's parliament, that his jets had, to date, never come across any UFOs from outer space. ”When the Air Self Defense Force detects indications of an unidentified flying object that could violate our country's airspace, it scrambles fighter jets if necessary and makes visual observation," Nakatani said. He continued: ”They sometimes find birds or flying objects other than aircraft but I don't know of a case of finding an unidentified flying object believed to have come over from anywhere other than Earth." Inoki has appeared in the U.S.-based WWE – which describes him as ”among the most respected men in sports-entertainment" – and is the founder of the New Japan Pro Wrestling organization. He entered Japan's Upper House for a second stint in politics in 2013. He also famously fought Muhammad Ali in 1976, in one of the first-ever mixed-discipline matches, which would later pave the way for today's wildly popular Mixed Martial Arts contests. Before his return to politics he was a regular fixture on Japanese TV variety shows and has promoted a slew of products, from hot sauce to banks. The maverick politician also traveled to Iraq in 1990 to try to secure the release of Japanese hostages, and has more recently attempted to replicate former NBA star Dennis Rodman's ”basketball diplomacy" by staging a wrestling tournament in North Korea. He reportedly converted to Islam in the 1990s, although he says he practices both Islam and Buddhism. The lawmaker, who is universally known in Japan for his colossal chin and once-ever-present red scarf – these days often replaced with a red necktie – as much as for his political achievements, had asked a Upper House Budget Committee meeting if aircraft were ever scrambled to meet extraterrestrial threats, and if research was being done into alien visitors, prompting Nakatani's response. Inoki also claims to have seen a UFO with his own eyes, but admitted that he didn't know personally if aliens existed. The exchange wasn't the first time Japanese politicians have discussed the implications of visitors from another planet. In 2007 then-Defense Minister Shigeru Ishiba pondered the legal ramifications, under Japan's pacifist constitution, of a defense against an invasion from outer space.

Figure 8: Example of text from SummEval dataset.

Assessing Dialogue Systems with Distribution Distances

Jiannan Xiang[1*] **Yahui Liu**[2*] **Deng Cai**[3] **Huayang Li**[4] **Defu Lian**[1] **Lemao Liu**[4]
[1]University of Science and Technology of China [2]University of Trento, Italy
[3]The Chinese University of Hong Kong [4]Tencent AI Lab, China

Abstract

An important aspect of developing dialogue systems is how to evaluate and compare the performance of different systems. Existing automatic evaluation metrics are based on turn-level quality evaluation and use average scores for system-level comparison. In this paper, we propose to measure the performance of a dialogue system by computing the distribution-wise distance between its generated conversations and real-world conversations. Specifically, two distribution-wise metrics, FBD and PRD, are developed and evaluated. Experiments on several dialogue corpora show that our proposed metrics correlate better with human judgments than existing metrics.

1 Introduction

Dialogue generation is a special text generation task, which has drawn booming attention in the natural language processing community. It is widely agreed that one single input query is often associated with multiple valid responses in this task, which is termed as a 1-to-n relationship between a query and its responses (Vinyals and Le, 2015; Zhou et al., 2017; Zhao et al., 2017; Liu et al., 2018; Chen et al., 2021; Chan et al., 2021; Gao et al., 2021). It increases the challenges of automatically evaluating the performance of dialogue systems.

In general, the previous evaluation metrics mainly focus on turn-level quality. For example, unsupervised word-overlapping or embedding-based metrics (Papineni et al., 2002; Lin, 2004; Mitchell and Lapata, 2008; Zhang et al., 2020) calculate the similarity or alignment between generated responses and reference responses, which is not well-suited for open-end dialogue tasks. Learned classification or regression systems (Lowe et al., 2017; Tao et al., 2018; Sellam et al., 2020; Ghazarian et al., 2019) are corpus-dependent because of requiring additional task-specific training or tuning, which run the risk of assigning lower quality to a better model in the overfitting or underfitting cases.

In this paper, we provide a new perspective that distribution distance between generated conversations and real conversations can be applied to measure the performances of dialogue systems. There are three main *contributions*: (1) We firstly propose two unsupervised distribution-wise metrics (i.e., FBD and PRD) to solve the evaluation issue in this field. (2) The experimental results show that the proposed distribution-wise metrics perform well. Particularly, FBD achieves compelling performances on most evaluation corpora, which shows a promising direction for designing evaluation metrics. (3) We collect the typical evaluation corpora and existing evaluation metrics in order to better assess the performance of dialogue systems, which could be useful for researchers in this community [1].

2 Related Work

In this section, we focus on *unsupervised* automatic evaluation metrics for dialogue system evaluation. In general, existing unsupervised metrics mainly measure *turn-level* qualities, which can be categorized into two main classes: word overlapping metrics and embedding-based metrics:

Word-overlapping Metrics Such metrics quantify the amount of word-overlap between generated response and reference responses. For example, BLEU (Papineni et al., 2002) calculates the geometric mean of the precision for n-gram. ROUGE (Lin, 2004) is a recall-oriented metric. METEOR (Banerjee and Lavie, 2005) computes the harmonic mean of precision and recall with stemming and syn-

---

[*] Equal contribution. Work was done during internship at Tencent AI Lab.

[1] The source codes and data are available at `https://github.com/yhlleo/frechet-bert-distance`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2192–2198
August 1–6, 2021.

onyms.

Embedding-based Metrics Embedding-based metrics align the generated response and the reference in latent semantic space. Some adopt the vector similarity of sentence embeddings as a quality measure. For example, Embedding Average (Foltz et al., 1998; Mitchell and Lapata, 2008) calculates sentence-level embeddings by averaging word representations. Vector Extrema (Forgues et al., 2014) computes sentence-level embeddings by taking the most extreme value for each dimension in all word vectors. Others adopt more fine-grained semantic matching. For example, Greedy Matching (Rus and Lintean, 2012) greedily matches each word in a generated response to a word in the reference response, and the final score is defined as the average of word-level similarity scores. Zhang et al. (2020) introduced a better embedding-based metric BERTScore that computes word similarity using contextual embeddings from pre-trained language models.

Our proposed methods are best placed in the literature of embedding-based metrics. However, there are two main differences from previous metrics in this field: (1) We compute the distribution distance between embedding sets as the system-level performance of a dialogue system, which does not require task-specific training/tuning; (2) We propose to extract sentence-level semantic representations directly from pre-trained language models (Devlin et al., 2019; Liu et al., 2019), where there are no operations of converting the wold-level embeddings to sentence-level embeddings.

3 Proposed Methods

Given a collect of sentence pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}^N$, we assume that the corresponding semantic representations $\{\boldsymbol{v}_i\}^N$ can be extracted in this manner:

$$\boldsymbol{v}_i = LM([\boldsymbol{x}_i, \boldsymbol{y}_i]) \tag{1}$$

where $LM(\cdot)$ refers to pre-trained language models(i.e., (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Clark et al., 2020)), $[\cdot,\cdot]$ refers to the concatenation operation. Intuitively, the differences between the distribution R of real samples and the distribution G of generated samples can be applied to measure the performances of dialogue systems (i.e., $d(R, G)$). Therefore, we propose two distribution-based methods to automatically evaluate the performance of dialogue systems, which are presented in this section in detail.

3.1 Fréchet Bert Distance

Semantic representations $\{\boldsymbol{v}_i\}^N$ are extracted by a pre-trained language model, which encodes the contextual information of the sentences. The main intuition is that the distribution of semantic representations of generated sentences should be as close as possible to the distribution of semantic representations of real sentences in a successful system. To measure this, we assume that such semantic representations follow a multi-dimensional Gaussian, which can be represented by variables: *mean* and *covariance*. The difference between two Gaussians (generated and real sentence pairs) is measured by the Fréchet distance (Dowson and Landau, 1982). We call the Fréchet distance between the distribution R with mean $(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ obtained from real sentence pairs and the distribution G with mean $(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ obtained from generated sentence pairs as "Fréchet Bert Distance" (FBD), which is formulated as:

$$\begin{aligned} d_{\mathrm{FBD}}(R, G) = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\| + \\ \mathrm{Tr}(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2}) \end{aligned} \tag{2}$$

Once the distribution of generated data closes to the distribution of real data, the model indeed achieves low FBD scores. Similarly, such distance (Heusel et al., 2017) has been widely verified in various Generative Adversarial Networks (GANs) in computer vision tasks (Karras et al., 2017; Zhang et al., 2018a; Park et al., 2019), which is consistent with increasing disturbances and human judgment. Surprisingly, we observed that FBD works well in evaluating open-end dialogue systems.

3.2 Precision-Recall Distance

We notice that FBD is based on the estimated Gaussian parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. There is an optional strategy to get rid of estimating the parameters. Inspired by (Sajjadi et al., 2018), we apply a precision-recall-based method, named as Precision-Recall Distance (PRD), to evaluate the distance between two distributions.

The key intuition is that precision should measure how much of G can be generated by a "part" of R while recall should measure how much of R can be generated by a "part" of G. In general, (a) If R is bimodal and G only captures one of the modes, we should have perfect precision but only limited recall; (b) In the opposite case, we should have perfect recall but only limited precision; (c) If $R = G$, we should have perfect precision and recall; (d) If the supports of R and G are disjoint, we should have zero precision and recall. The BPD is formulated as:

$$d_{\text{PRD}}(R, G) = \max \left\{ \frac{2\alpha(\lambda)\beta(\lambda)}{\alpha(\lambda) + \beta(\lambda)} \middle| \lambda \in \Lambda \right\} \tag{3}$$

where $\Lambda = \{\tan(\frac{i}{m+1}\frac{\pi}{2}) | i = 1, \cdots, m\}$, $m \in \mathbb{N}$ refers to a given angular resolution, $\alpha(\lambda) = \sum_{\boldsymbol{v} \in \mathcal{V}} \min(\lambda R(\boldsymbol{v}), G(\boldsymbol{v}))$ and $\beta(\lambda) = \sum_{\boldsymbol{v} \in \mathcal{V}} \min(R(\boldsymbol{v}), \frac{G(\boldsymbol{v})}{\lambda})$ [2]. Therefore, the better dialogue systems will achieve higher PRD scores.

4 Experiments

4.1 Datasets & Systems

To verify the two proposed metrics, we conduct experiments on six public dialogue corpora.

Baseline Metrics. We mainly compare with several widely-used metrics in text generation field: a) three word-overlapping metrics: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Denkowski and Lavie, 2014); b) four embedding-based metrics: Greedy Matching (Rus and Lintean, 2012), Embedding Average (Wieting et al., 2015), Vector Extrema (Forgues et al., 2014) and BERTScore (Zhang et al., 2020). All these metrics do not require task-specific training.

Datasets. We collect three recently released evaluation corpora which consist of dialogue query and response samples of different systems, and the corresponding human annotations:

- **Persona(M)**: USR (Mehri and Eskénazi, 2020) built an evaluation corpus based on PersonaChat (Zhang et al., 2018b), in which both four system outputs and the corresponding human evaluation scores were collected.
- **Daily(H), Convai2, and Empathetic**: GRADE (Huang et al., 2020) used three dialogue corpora, including DailyDialog (Lowe et al., 2017), Convai2 (Dinan et al., 2019) and EmpatheticDialogues (Rashkin et al., 2018), to do the evaluations and compared two dialogue models: Transformer-Ranker and Transformer-Generator collected from the ParlAI platform (Miller et al., 2017).
- **Daily(Z) and Persona(Z)**: Dialogue Evaluation (Zhao et al., 2020) used DailyDialog (Lowe et al., 2017) and PersonaChat (Zhang et al., 2018b) to build two evaluation corpora and collected outputs of six generative models (with three decoding strategies). The appropriateness of each response was obtained by human annotation.

Source	**Address**
Transformers	`https://github.com/huggingface/transformers`
USR	`https://github.com/Shikib/usr`
GRADE	`https://github.com/li3cmz/GRADE/tree/main/evaluation`
Daily(Z) Persona(Z)	`https://github.com/ZHAOTING/dialog-processing/tree/master/src/tasks/response_eval`
ParlAI	`https://github.com/facebookresearch/ParlAI`
BLEU METEOR	`https://github.com/nltk/nltk`
ROUGE-L Greedy Average Extrema	`https://github.com/Maluuba/nlg-eval`
BERTScore	`https://github.com/Tiiiger/bert_score`

Table 1: All the public resources in our experiments.

Corpus	**Num. of Samples**	**Systems**
Persona(M)	60	Seq2Seq LSTM language model Key-Value Profile Memory Network Generated Human-written
Daily(H)	150	Transformer-Ranker Transformer-Generator
Convai2	150	
Empathetic	150	
Daily(Z)	100	Seq2Seq Attentional Seq2Seq HRED VHRED GPT2-sm GPT2-md
Persona(Z)	150	

Table 2: The details of each evaluation corpus.

Implementation Details. We leverage pre-trained BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) for utterance-level contextualized encoding [3] without additional tuning or training the language models. For each query and response pair

---

[2] For a distribution P with a finite state space $\mathcal{V}$, we have $\boldsymbol{v} \in \mathcal{V}$ and $P(\boldsymbol{v}) > 0$.

[3] Based on the public project: `https://github.com/huggingface/transformers`

Metric	Persona(M)		Daily(H)		Convai2		Empathetic		Daily(Z)		Persona(Z)	
	Spr.	Pr.	Spr.	Pr.	Spr.	Pr.	Spr.	Pr.	Spr.	Pr.	Spr.	Pr.
Word-Overlapping Metrics												
BLEU	.400	.672	.445	.444	.800	.801	.136	.331	.595	.421	.400	.390
METEOR	.800	.860	.018	.050	.800	.767	.382	.133	.643	.689	.700	.936
ROUGE-L	.600	.289	.545	.417	.200	.061	.391	.472	.738	.725	.400	.915
Embedding-Based Metrics												
Greedy	.600	.260	.855	.764	.600	.794	.736	.864	.690	.726	.100	.835
Average	.800	**.863**	.209	.209	.600	.879	.664	.725	.548	.769	.300	.861
Extrema	.600	.435	.745	.761	.800	.766	.618	.722	.595	.746	.500	.834
BERTScore^B	.800	.590	.137	.082	.800	.817	.300	.077	.857	.883	.900	.961
BERTScore^R	.800	.517	.855	.857	.800	.939	.600	.697	.714	.860	.700	.918
Distribution-Based Metrics												
FBD^B	**1.00**	.853	.564	.717	.800	.854	.427	.623	.786	.763	.400	.923
FBD^R	**1.00**	.802	**.891**	**.926**	.800	.747	**.864**	**.951**	**.929**	**.963**	**1.00**	.860
PRD^B	.800	.637	.221	.409	**1.00**	**.972**	.227	.399	.690	.914	.900	**.984**
PRD^R	.800	.660	.591	.578	**1.00**	.913	.545	.583	.762	.906	.900	.932

1.“Spr.” and “Pr.” refer to Spearman and Pearson correlation coefficients, respectively.
2. ^{B} and ^{R} mean using BERT (base) and RoBERTa (base) as language models, respectively.

Table 3: Correlations of all the metrics with overall quality ratings the six dialogue corpora.

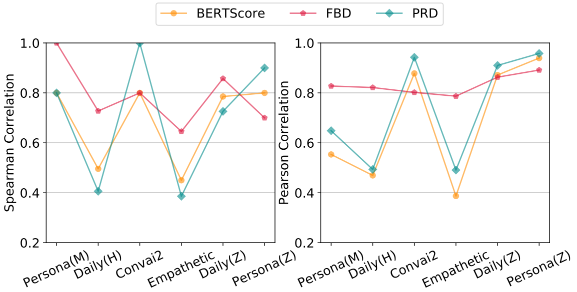

Figure 1: The comparisons between BERTScore, FBD and PRD metrics on various corpora. For each corpus, we average the performances of BERT and RoBERTa language models.

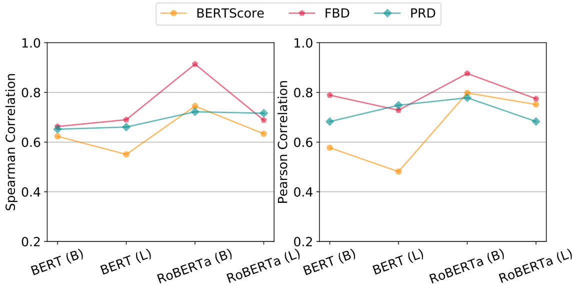

Figure 2: Comparisons between BERTScore, FBD and PRD metrics on various language models. For each model, we average the performances on all the six evaluation corpora.

$(\boldsymbol{x}_i, \boldsymbol{y}_i)$, we use the last hidden output of [CLS] as its semantics representation without tuning or training the language models. To assess the system-level performances of dialogue systems, we calculate the Spearman and Pearson correlations between the rankings of human evaluation and the rankings of evaluation metrics. If a evaluation metric is designed for *turn-level* evaluation, we average the all turn-level scores as the performance of the corresponding dialogue system.

Public Resources. All the compared evaluation corpora and evaluation metrics are available in Table 1. Once the official implementations are not available, we use the repositories with highest “stars” on GitHub. The details of each evaluation corpus, including number of samples and compared dialogue systems in each corpus, are presented in Table 2.

4.2 Results

We compute the system-level correlation between all automatic metrics and the quality ratings by using Spearman and Pearson correlation coefficients.

Model	P(M)	D(H)	C2	EM	D(Z)	P(Z)
BERT	.57±.44	.65±.39	.60±.42	.65±.39	.67±.38	.65±.38
RoBERTa	.71±.38	.68±.37	.73±.34	.68±.37	.68±.37	.71±.33

1. {P(M), D(H), $\cdots$, P(Z)} refer to the six evaluation corpora.
2. The reported values are mean±standard deviation.

Table 4: Comparisons of the normality on various evaluation corpora. The normality is calculated over each dimension of the extracted semantic representations.

The performances of various evaluation metrics on different public corpora are reported in Table 3. Our proposed two metrics (i.e., FBD and PRD) show comparable performances over the baseline metrics. Especially, FBD$^{\text{R}}$ achieves compelling performances on five corpora, which indicates a good ability of generalization and robustness on various corpora. In addition, most evaluation metrics are sensitive to the evaluation corpora. For example, BLEU performs well in **Convai2** but fails in **Empathetic**. Similarly, BERTScore$^{\text{B}}$ performs well in **Convai2** and **Persona(Z)** but fails in **Daily(H)** and **Empathetic**. It indicates that the selection of evaluation corpora has a great influence on assessing the performances of evaluation metrics. Hence, it's better to use multiple corpora to do the comparisons between metrics. Obviously, our proposed FBD$^{\text{R}}$ outperforms the existing evaluation metrics in the view of robustness.

In Figure 1, we compare the evaluation metrics in the perspective of various evaluation corpora, where the results of BERT and RoBERTa language models are averaged. It confirms the superiority of the FBD metric. Compared to the performance of USR (Mehri and Eskénazi, 2020) (1.000/.820 on Persona(M)), a reference-free metric that relies on task-specific training/tuning with task-specific data, the performances of our proposed methods are comparable without any training/tuning. Therefore, we believe that it's a promising direction to explore distribution-wise metrics for assessing dialogue systems in this field.

As shown in Figure 2, we average the performances of each metric on all evaluation corpora. It shows that our proposed FBD has higher performance expectations that outperform BERTScore with different language models. The large models do not show improvements in average performance compared to the base models. In general, FBD metric achieves better Spearman and Pearson correlations compared to PRD. Surprisingly, RoBERTa-based metrics, including BERTScore, the proposed FBD and PRD, perform better than the corresponding BERT-base ones. Given that our FBD metric lies on the assumption of multivariate Gaussian distribution, we hypothesize that the semantic representations extracted by RoBERTa model fit Gaussian distribution better than BERT model. To verify this point, as shown in Table 4, we use "Shapiro–Wilk test" (Razali et al., 2011) to calculate the normality in statistics, where small values lead to the rejection of normality whereas a value of one indicates normality of the data.

Model	**Relevance**		**Grammar**		**Content**		**Overall**	
	Spr.	Pr.	Spr.	Pr.	Spr.	Pr.	Spr.	Pr.
Word-Overlapping Metrics								
BLEU	.857	.454	.167	.213	.143	.270	.595	.421
METEOR	.810	.736	.119	.044	.024	.349	.643	.689
ROUGE-L	.857	.758	.238	.075	.190	.375	.738	.725
Embedding-Based Metrics								
Greedy	.881	.769	.214	.133	.119	.350	.690	.726
Average	.762	.808	.071	.197	.238	.399	.548	.769
Extrema	.714	.776	.143	.009	.476	.539	.595	.746
BERTScore$^{\text{B}}$	**.976**	.908	.333	.111	.524	.593	.857	.883
BERTScore$^{\text{R}}$	.857	.889	.119	.115	.310	.604	.714	.860
Distribution-Based Metrics								
FBD$^{\text{B}}$	.762	.790	.190	.203	**.667**	.404	.786	.763
FBD$^{\text{R}}$	**.976**	**.965**	**.429**	**.472**	.524	**.839**	**.929**	**.963**
PRD$^{\text{B}}$	.833	.932	.238	.156	.571	.708	.690	.914
PRD$^{\text{R}}$	.881	.925	.238	.235	.452	.729	.762	.906

1."Spr." and "Pr." refer to Spearman and Pearson correlation coefficients, respectively.
2. $^{\text{B}}$ and $^{\text{R}}$ mean using BERT (base) and RoBERTa (base) as language models, respectively.

Table 5: Fine-grained comparisons on Daily(Z) (Zhao et al., 2020) corpus.

4.3 Fine-Grained Performances

In the evaluation corpus Daily(Z) (Zhao et al., 2020), it provides four fine-grained human evaluation scores, including *relevance*, *grammar*, *content* and *overall*, which can be used to dive more insights of different evaluation metrics.

As shown in Table 5, our proposed metric FBD$^{\text{R}}$ achieves the best performances on most evaluations in the fine-grained comparisons. It indicates the distribution-wise metric correlate better with human judgements on various aspects.

5 Conclusions

In this paper, we propose to measure the performance of a dialogue system by computing the distribution-wise difference between its generated conversations and real-world conversations. Specifically, two distribution-wise metrics, FBD and PRD, are developed on pre-trained language models. Experiments on six public dialogue corpora show that our proposed metrics correlate better with human judgments than existing metrics.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Zhangming Chan, Lemao Liu, Juntao Li, Haisong Zhang, Dongyan Zhao, Shuming Shi, and Rui Yan. 2021. Enhancing the open-domain dialogue evaluation in latent space. In *Findings of ACL*.

Wang Chen, Piji Li, and Irwin King. 2021. A training-free and reference-free summarization evaluation metric via centrality-weighted relevance and self-referenced redundancy. In *ACL*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the workshop on statistical machine translation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

DC Dowson and BV Landau. 1982. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455.

Peter W Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *NeurIPS, modern machine learning and natural language processing workshop*.

Jun Gao, Wei Bi, Ruifeng Xu, and Shuming Shi. 2021. REAM♯: An enhancement approach to reference-based evaluation metrics for open-domain dialog generation. In *Findings of ACL*.

Sarik Ghazarian, Johnny Tian-Zheng Wei, A. Galstyan, and Nanyun Peng. 2019. Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. *ArXiv*, abs/1904.10635.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*.

Lishan Huang, Zheng Ye, Jinghui Qin, Liang Lin, and Xiaodan Liang. 2020. Grade: Automatic graph-enhanced coherence metric for evaluating open-domain dialogue systems. In *EMNLP*.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL*.

Yahui Liu, Wei Bi, Jun Gao, Xiaojiang Liu, Jian Yao, and Shuming Shi. 2018. Towards less generic responses in neural conversation models: A statistical re-weighting method. In *EMNLP*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ryan Lowe, Michael Noseworthy, I. Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ACL*.

Shikib Mehri and M. Eskénazi. 2020. Usr: An unsupervised and reference free evaluation metric for dialog generation. In *ACL*.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*.

Kishore Papineni, S. Roukos, T. Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*.

Nornadiah Mohd Razali, Yap Bee Wah, et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33.

Vasile Rus and Mihai Lintean. 2012. An optimal assessment of natural language student input using word-to-word similarity metrics. In *International Conference on Intelligent Tutoring Systems*. Springer.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. 2018. Assessing generative models via precision and recall. In *NeurIPS*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *ACL*.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. 2018a. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(8):1947–1962.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *ICLR*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*.

Tianyu Zhao, Divesh Lala, and Tatsuya Kawahara. 2020. Designing precise and robust dialogue response evaluators. In *ACL*.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *AAAI*.

Neural Combinatory Constituency Parsing

Zhousi Chen, Longtu Zhang, Aizhan Imankulova, and Mamoru Komachi
Tokyo Metropolitan University
6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
`chen-zhousi@ed.tmu.ac.jp` `vincentzlt@outlook.com`
`aizhan.imankulova@cogsmart-global.com` `komachi@tmu.ac.jp`

Abstract

We propose two fast neural combinatory models for constituency parsing: binary and multi-branching. Our models decompose the bottom-up parsing process into 1) classification of tags, labels, and binary orientations or chunks and 2) vector composition based on the computed orientations or chunks. These models have theoretical sub-quadratic complexity and empirical linear complexity. The binary model achieves an F1 score of 92.54 on Penn Treebank, speeding at 1327.2 sents/sec. Both the models with XLNet provide near state-of-the-art accuracies for English. Syntactic branching tendency and headedness of a language are observed during the training and inference processes for Penn Treebank, Chinese Treebank, and Keyaki Treebank (Japanese).

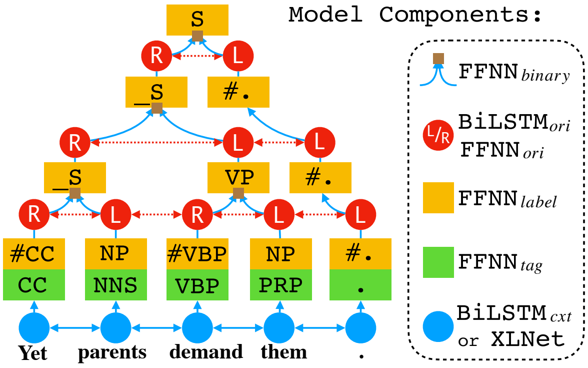

Figure 1: Parsing instance with the binary model. The bottom-up flow of word information is indicated by blue arrows and orientation flows by dotted red arrows. Binary parsing explores the internal constituents of `S`. Special labels prefixed with "#" or "_" are *sub* category placeholders caused by binarization and stratification.

1 Introduction

Transition-based and chart-based methods are two main paradigms for constituency parsing. Transition-based parsers (Dyer et al., 2016; Kitaev and Klein, 2020) build a tree with a sequence of local actions. Despite their $O(n)$ computational complexity, the locality makes them less accurate and necessitates additional grammars or lookahead features for improvement (Kuhlmann et al., 2011; Zhu et al., 2013; Liu and Zhang, 2017c). By contrast, chart-based parsers are conceptually simple and accurate when used with a CYK-style algorithm (Kitaev and Klein, 2018; Zhou and Zhao, 2019) for finding the global optima. However, their complexity is $O(n^3)$. To achieve both accuracy and simplicity (without high complexity) is a critical problem in parsing.

Recent efforts were made using neural models. In contrast to earlier symbolic approaches (Charniak, 2000; Klein and Manning, 2003), neural models are simplified by utilizing their adaptive distributed representation, thereby eliminating complicated symbolic engineering. The seq2seq model for parsing (Vinyals et al., 2015) leverages such representation to interpret the structural task as a general sequential task. With augmented data and ensemble, it outperforms the symbolic models mentioned in Petrov et al. (2006) and provides a complexity of $O(n^2)$ with the attention mechanism (Bahdanau et al., 2015). However, its performance is inferior to those of specialized neural parsers (Liu and Zhang, 2017a,b,c). Socher et al. (2013) proposed a parsing strategy for a symbolic constituent parser augmented with neural vector compositionality. It did not outperform the two paradigms in neural style probably because the neural techniques, such as contextualization, are not fully exploited. Kitaev and Klein (2020) showed that a simple transition-based model with a dynamic distributed representation, BERT (Devlin et al., 2019), nearly delivers a state-of-the-art performance.

We propose a pair of greedy combinatory parsers (i.e., neural combinators) that efficiently utilize vector compositionality with recurrent components to

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2199–2213
August 1–6, 2021.

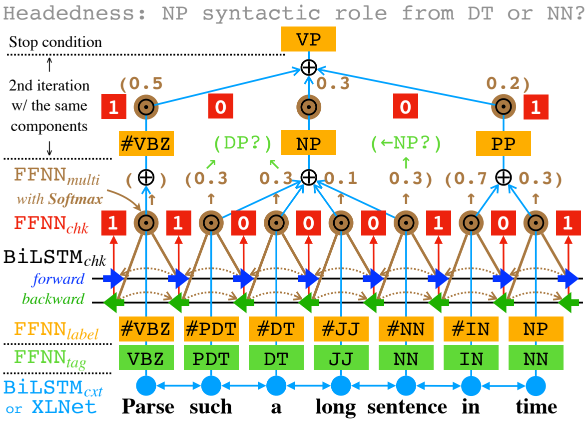

Figure 2: Multi-branching parsing uses chunks instead of orientations to form constituents. Chunks impose Softmax-normalized weights for their inputs. The unsupervised weights provide a shred of evidence for the headedness problem (Zwicky, 1985).

address the aforementioned issues. Their bottom-up parsing process is a recursive layer-wise loop of classification and vector composition, as illustrated in Figures 1 & 2. Both parsers work on multiple unfolded variable-length layers, iteratively combining vectors until one vector remains. The binary model provides either left or right orientation for each word or constituent, whereas the multi-branching model marks chunks as constituents at their boundaries. Constituent embeddings are composed based on orientations or chunks. Tagging and labeling are directly performed on all composed embeddings, creating the elements for building a tree: tags, labels, and paths. The deterministic and greedy characteristics yield two simple and fast models, and they investigate different linguistic aspects.

The contributions of our study are as follows:

- We propose two combinatory parsers[1] at $O(n)$ average-case complexity with a theoretical $O(n^2)$ upper bound. The binary parser achieves a competitive F1 score on Penn Treebank. Both models are the fastest and yet more compact than many previous models.
- We extend the proposed models with a recent pre-trained language model, XLNet (Yang et al., 2019). These models have higher speeds and are comparable to state-of-the-art parsers.
- The binary model leverages Chomsky normal form (CNF) factors as a training strategy and reflects the branching tendency of a language. The multi-branching model reveals constituent headedness (Zwicky, 1985) with an attention mechanism.

2 Previous Work

Transition-based parsers. A transducer takes sequential lexical inputs and produces sequential tree-constructing actions in $O(n)$ time. Although it can perfectly parse formal languages, complex semantics and long dependencies make it difficult to parse natural languages. Informative features (Liu and Zhang, 2017c; Kitaev and Klein, 2020; Yang and Deng, 2020), or training and decoding strategies such as dynamic oracles (Cross and Huang, 2016), reranking (Charniak and Johnson, 2005), beam search, and ensemble, can increase the accuracy. However, these make the models complex, and the paradigm fails to naturally parallelize actions.

Chart-based parsers. An exhaustive search algorithm checks every possibility in a triangular chart and finds the optimal tree globally. Recent neural chart parsers have achieved state-of-the-art accuracy (Kitaev and Klein, 2018; Zhou and Zhao, 2019; Mrini et al., 2020; Zhang et al., 2020). Despite their high accuracy, they are comparatively inefficient. Only $2n - 1$ of $O(n^2)$ scoring nodes in the chart contain true constituents; many are filler nodes. Chart parsers are often specially engineered for high-speed decoding. (e.g., using Cython)

Other parsers. Shen et al. (2018) and Nguyen et al. (2020) proposed local-and-greedy parsers in the top-down splitting style. Their models facilitate divide-and-conquer algorithms that construct the tree based on the magnitude of the splitting scores. A similar way of leveraging concurrent and greedy operations appears in an easy-first parser (Goldberg and Elhadad, 2010). Sequential labeling (Gómez-Rodríguez and Vilares, 2018; Wei et al., 2020) is a new active thread that also enables parallelism and fast decoding. Collobert (2011) designed an iterative chunking process for parsing. His work stratifies trees into levels of IOBES prefixed constituent chunking nodes. Similar to ours, his parser works from the bottom levels to higher levels. However, the complexity is fixed at $O(n^2)$ without any node combinations. All models introduced in this section do not exploit vector compositionality.

---

[1] Our code, visualization tool, and pre-trained models are available at `https://github.com/tmu-nlp/nccp`

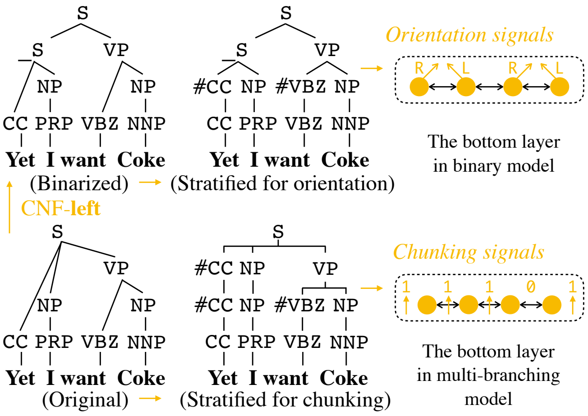

Figure 3: Example illustrating our tree stratification. Both binary and multi-branching stratified trees contain redundancy, from which the original tree can be recovered with a few simple heuristic rules. Four in a layer combine into two with a compression ratio of 1/2.

Category	Samples	# Types
Original	`S NP VP SBAR+S`	104
_Sub	`_S _NP _VP _PP`	25
#POS	`#NNP #DT #JJ #.`	45

Table 1: Three categories of our constituent label set with their samples and number of types. This is created from Penn Treebank (PTB). '**_Sub**' and '**#POS**' are relaying types, which we group into a *sub* category.

CNF	**Left-factoring**		**Right-factoring**	
Ori.	**Left**	**Right**	**Left**	**Right**
PTB	3.8M	4.4M	2.3M	6.5M
CTB	2.5M	1.7M	1.4M	2.8M
KTB	4.5M	0.9M	1.8M	2.1M
nCNF	**Midin-factoring**		**Midout-factoring**	
PTB	3.0M	5.3M	2.8M	5.2M
CTB	1.9M	2.2M	1.7M	2.1M
KTB	2.8M	1.7M	2.5M	1.2M

Table 2: Frequencies of orientation with different CNF (biased) and non-CNF (balanced) factors in different stratified corpora.

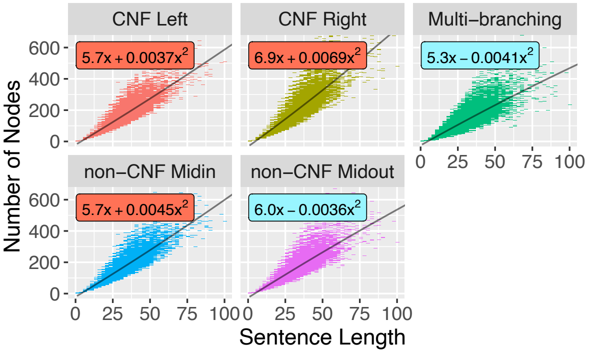

Figure 4: Linear empirical complexity in stratified PTB. Linear regression reflects insignificant $O(n^2)$ tendencies. We differentiated the quadratic terms with red or light blue colors and omitted the constant biases.

3 Neural Combinatory Parsing

3.1 Data and Complexity

Our models require stratified trees to train recurrent layers, and the binary model requires further binarization. Stratification and binarization introduce redundant relaying nodes to the trees.

Tree binarization. From the bottom-up perspective, a binary tree describes the order in which words and constituents combine with their neighbors into larger constituents, as shown in Figure 3. The orientations of the four words (i.e., right-left-right-left) determine the first combination.

After binarization, we label the relaying sub-constituents with the parent label prefixed with an underscore mark. If terminal POS tags do not immediately form constituents, we create relaying placeholders prefixed with a hash mark[2], as presented in Table 1. Unary branches were collapsed into a single node. Plus marks were used to join their labels (e.g., `SBAR+S`), and all trace branches were removed. The CNF with either a **left** or a **right** factor is commonly used. However, it is heuristically biased, and trees can be binarized using other balanced splits such as always splitting from the center to create a complete binary tree (**mid-out**) and iteratively performing **left** and **right** to create another balanced tree (**mid-in**). Finally, the orientation is extracted from the paths of these binary trees.

We binarized Penn Treebank (Marcus et al., 1993, PTB) for English, Chinese Treebank (Xue et al., 2005, CTB) for Chinese, and Keyaki Treebank[3] (Butler et al., 2012, KTB) for Japanese to present the syntactic branching tendencies in Table 2. As English is a right-branching language, its majority orientation is to the right. Even left-factoring cannot reverse the trend, but it should create a greater balance. Figure 4 shows that it is less effective to stratify PTB with a **right** factor because it enhances the tendency. The reverse tendency emerges in the KTB corpus as Japanese is a left-branching language. For Chinese, CTB does

---

[2] Multi-branching trees do not require binarization. The '**_Sub**' group disappears, but the '**#POS**' group persists.

[3] `https://github.com/ajb129/KeyakiTreebank/tree/master/treebank`

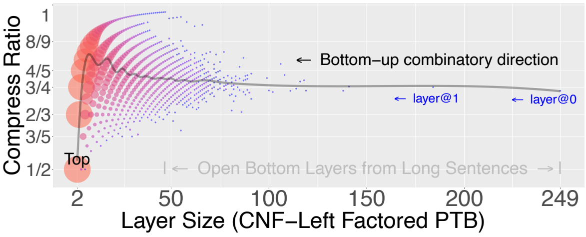

Figure 5: Layer-wise compression ratio over different sizes. The dot size was proportional to the situation count. Statistically, the ratio had a stable mean of 0.77.

not exhibit a clear branching tendency. Non-CNF factors preserve the original tendency.

Complexity. Our models are trained with stratified treebanks. The complexity for inference follows the total number of nodes in each layer of a tree. There are two ideal cases: 1) Complete balanced trees with complexity $O(n)$. They contain multiple independent phrases and enable full concurrency. 2) Trees with a single dependency core. The model reduces a constant number of nodes in each layer, resulting in $O(n^2)$ complexity.

While each parse is a mixture of many cases, the *empirical* complexity prefers the first case. Formally, the average-case complexity can be inferred as $O(n)$ with the help of a stable compression ratio $0 < C < 1$ ($C \geq 0.5$ for binary). Let m_i represent the number children of the i-th tree in a general layer; the compression ratio can be stated as $C = \frac{\sum_i 1}{\sum_i m_i}$. Our stratified treebanks give stable Cs for layers of different lengths, as shown in Figure 5. For the k-th layer of a sentence with n words, the number of nodes to compute can be expected to be $C^k \cdot n$. Based on tree height $K > 0$, the expected number of total parsing nodes is

$$\sum_{k=0}^{K} C^k \cdot n = n \cdot \sum_{k=0}^{\infty} C^k - n \cdot \sum_{k=K+1}^{\infty} C^k < \frac{n}{1 - C}.$$

The partial geometric series determines an empirically linear complexity on average.

Theoretically, the complexity has a quadratic upper bound. The general layer with

$$m_i = \begin{cases} M & \text{if } i = j \\ 1 & \text{otherwise} \end{cases}$$

entails the second case, where m_j is the only M-ary branch in each layer. The nodes shape a triangular stratified tree with an $O(n^2)$ complexity. However, this case is rare, especially for long sentences that should contain several concurrent phrases. Otherwise, regression in Figure 4 should show significant $O(n^2)$ tendencies. (See Appendix A.1 for more support and examples.)

Algorithm 1: Combinatory Parsing

```
1  Function PARSE($e_{0:n}$; $t_{0:n}$, $l^{0:k}_{0:n_k}$, $o^{0:k}_{0:n_k}$ or $c^{0:k}_{0:n_k+1}$):
2      $x^0_{0:n} \leftarrow$ BiLSTM$_{cxt}$($e_{0:n}$)
3      for $i \leftarrow 0$ to $n - 1$ do
4          $\hat{t}_i \leftarrow$ FFNN$_{tag}$($x^0_i$)
5          $L_{tag} \leftarrow$ CROSS-ENTROPY($t_i, \hat{t}_i$)
6      for $j \leftarrow 0$ to $k$ do
7          for $i \leftarrow 0$ to $n_j - 1$ do
8              $\hat{l}^j_i \leftarrow$ FFNN$_{label}$($x^j_i$)
9              $L_{label} \leftarrow$ CROSS-ENTROPY($l^j_i, \hat{l}^j_i$)
10         $x^{j+1}_{0:n_{j+1}} \leftarrow$ COMPOSE($x^j_{0:n_j}$; $o^j_{0:n_j}$ or $c^j_{0:n_j+1}$)
11     return $\hat{t}_{0:n}, \hat{l}^{0:k}_{0:n_k}$
```

Data structure. To summarize the data components of a treebank corpus, we used four tensors of indices for 1) words, 2) POS tags, 3) stratified syntactic labels, 4) stratified orientations, or 5) stratified chunks,

$$\left(x_{0:n}, t_{0:n}, l^{0:k}_{0:n_k}, o^{0:k}_{0:n_k} \text{ or } c^{0:k}_{0:n_k+1}\right)_j \in D,$$

where n is the length of the j-th sentence, k indicates the k-th layer of the stratified data, and n_k is the layer length. ":" indicates a range of a sequence.

3.2 Combinatory Parsing

Our models comprise four feedforward (FFNN) and two bidirectional LSTM (BiLSTM) networks to decompose parsing into collaborative functions, as shown in Algorithm 1. During training, we use teacher forcing. In the inference phase, the supervised signals behind all semicolons are ignored; the predicted signals serve as their substitute.

Input $e_{0:n}$ is an embedding sequence indexed by $x_{0:n}$. In lines 2–5, the model prepares a contextual sequence for the combinator and predicts the lexical tags. Lines 6–10 describe the layer-wise loop of the combinator.

The tagging and labeling functions, `FFNN`$_{tag}$ and `FFNN`$_{label}$, are 2-layer FFNNs. Their first layer is shared, creating a hidden layer necessary for projecting diversified situations in the manifold to the non-zero logits for the argmax decision. The core function `COMPOSE`[4] is either a binary Algorithm 2 or a multi-branching Algorithm 3.

---

[4] `COMPOSE` with BiLSTM cannot be parallelized to $O(1)$.

Algorithm 2: Binary Compose

```
Function COMPOSE ($x^j_{0:n_j}$; $o^j_{0:n_j}$):
    $h^j_{0:n_j} \leftarrow$ BiLSTM$_{ori}$ ($x^j_{0:n_j}$)
    for $i \leftarrow 0$ to $n_j - 1$ do
        $\hat{o}^j_i \leftarrow$ FFNN$_{ori}$ ($h^j_i$)
        $L_{ori} \leftarrow$ HINGE-LOSS ($o^j_i, \hat{o}^j_i$)
        if $i > 0$ and $\hat{o}^j_{i-1} + (1 - \hat{o}^j_i) > 0$ then
            Append BINARY ($\hat{o}^j_{i-1}, \hat{o}^j_i,\ x^j_{i-1}, x^j_i$) to $x^{j+1}$
    return $x^{j+1}_{0:n_{j+1}}$

Function BINARY ($o_L, o_R,\ x_L, x_R$):
    if $o_L + (1 - o_R) = 1$ then                // relay
        return $o_L \cdot x_L + (1 - o_R) \cdot x_R$
    else                          // vector interpolation
        $\lambda \leftarrow \sigma$ FFNN$_{binary}$ ($x_L \oplus x_R$)
        return $\lambda \odot x_L + (1 - \lambda) \odot x_R$
```

Algorithm 3: Multi-branching Compose

```
Function COMPOSE ($x^j_{0:n_j}$; $c^j_{0:n_j+1}$):
    $\vec{h}^j_{0:n_j}, \overleftarrow{h}^j_{0:n_j} \leftarrow$ BiLSTM$_{chk}$ ($x^j_{0:n_j}$)
    Pad $\vec{h}^j_{0:n_j}$ with $\vec{h}^j_{-1}$ and $\overleftarrow{h}^j_{0:n_j}$ with $\overleftarrow{h}^j_{n_j}$
    for $i \leftarrow 0$ to $n_j$ do
        $\hat{c}^j_i \leftarrow$ FFNN $_{chk}(\vec{h}^j_{i-1} \oplus \overleftarrow{h}^j_i)$
        $L_{chk} \leftarrow$ HINGE-LOSS ($c^j_i, \hat{c}^j_i$)
        if $\hat{c}^j_i = 1$ then Append $i$ to $s$          // segment
        if $i < n_j$ then
            $d^j_i \leftarrow (\vec{h}^j_i - \vec{h}^j_{i-1}) \oplus (\overleftarrow{h}^j_i - \overleftarrow{h}^j_{i+1})$
    for $i \leftarrow 0$ to $|s| - 1$ do
        Append MULTI ($d^j_{s_i:s_{i+1}},\ x^j_{s_i:s_{i+1}}$) to $x^{j+1}$
    return $x^{j+1}_{0:n_{j+1}}$

Function MULTI ($d_{chk},\ x_{chk}$):
    $\lambda_{chk} \leftarrow Softmax($ FFNN$_{multi}$ ($d_{chk}$) )
    return $\sum_i^{chk} \lambda_i \odot x_i$
```

Binary model. In Algorithm 2, the orientation function is hinted by BiLSTM$_{ori}$. A single-layer FFNN$_{ori}$ with a threshold reduces the outputs to an integer of either 0 or 1 to indicate two possible orientations. In function BINARY, when two adjacent orientations agree as they sum to 2, their embeddings are combined by a combinatory operation. σ is the Sigmoid function, "$\oplus$" represents concatenation, and "$\odot$" represents pointwise multiplication. (See Appendix A.3 for more binary variants.)

Multi-branching model. To resemble binary interpolation, we use the Softmax function for each chunk, as described in Algorithm 3. BiLSTM$_{chk}$ is in place of BiLSTM$_{ori}$ to hint FFNN$_{chk}$ emitting chunk signals. Segment s splits $x^j_{0:n_j}$ and $d^j_{0:n_j}$ into chunks of x_{chk} and d_{chk}. FFNN$_{multi}$ and Softmax turn d_{chk} into attention λ_{chk} to interpolate vector chunk x_{chk}. Binary interpolation λ is a special case of the multi-branching λ_{chk} because Sigmoid and Softmax functions are closely related.

To obtain the final tree representation, we apply a symbolic pruner in the same bottom-up manner to remove redundant nodes, expand the collapsed nodes, and assemble the sub-trees based on the neural outputs. (See Appendix A.4.)

4 Experiments

We follow previous data splits for PTB, CTB, and KTB (See Appendix A.2). The preprocessing of data is described in Section 3.1.

For the binary model, we explored interpolated dynamic datasets by sampling two CNF factored datasets. This is because of the following: 1) The experiments with the non-CNF factors did not yield any promising results; thus, we have not reported them. 2) The language was loosely left-branched, right-branched, or did not show a noticeable tendency. Moreover, the use of a single static dataset may introduce a severe orientation bias. 3) All factors are intermediate variables and equally correct. We defined the sampling strategies with two static CNF-factored datasets at certain ratios and named each strategy in the format "L%R%" according to the ratio percentages. Our experiments mainly focus on binary model **B** because of the aforementioned property for training parsers more accurate than multi-branching model **M**.

Our parsers do not contain lexical information components (Liu and Zhang, 2017c; Kitaev and Klein, 2018). Instead, we use fastText (Bojanowski et al., 2017) because we can obtain pre-trained models easily for many languages or train new ones from scratch with the corpora at hand. We examined its influence in Section 4.2, whereas the official pre-trained embeddings are the default.

Meanwhile, pre-trained language models are useful for various tasks, including constituency parsing (Kitaev and Klein, 2018, 2020; Zhou and Zhao, 2019; Yang and Deng, 2020; Mrini et al., 2020). We chose XLNet (Yang et al., 2019) to compare with the static fastText embeddings. Specifically, either a 1-layer FFNN (/0) or an n-layer BiLSTM (/n^+) was used to convert the 768-unit output to our model size. We used a GeForce GTX 1080 Ti with 11 GB and a TITAN RTX with 24GB memory only for tuning XLNet.

The model size for vector compositionality was set at 300. The hidden sizes for labeling, orientation, and chunking were 200, 64, and 200, re-

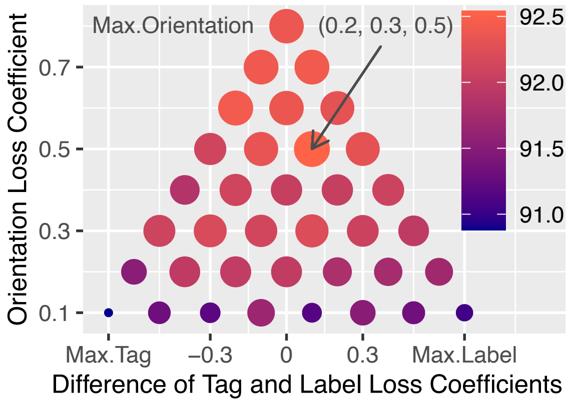

Figure 6: Grid search with an interval of 0.1 in a space of (tag, label, orientation) loss coefficients. The best was (0.2, 0.3, 0.5) indicated by an arrow.

spectively. Different numbers of layers of the `BiLSTM`$_{cxt}$ ($/n$) were explored, and the default was six layers. `HINGE-LOSS` was the default criterion for orientation while binary cross-entropy (`BCE-LOSS`) was tested. The coefficients of the three losses were explored and the default were $L = 0.2 \cdot L_{tag} + 0.3 \cdot L_{label} + 0.5 \cdot L_{ori\ (\text{or}\ chk)}$.

4.1 Overall Results

Table 3 lists the parsing accuracies and speeds of the single models in ascending order according to their F1 scores for the PTB corpus. The transition-based parsers with $O(n)$ complexity appear at the top of the table, followed by other types of models, and the chart parsers running in $O(n^3)$ time are at the bottom of the table. The models exhibited similar trends for the CTB. Shen et al. (2018) and our models belong to type O and have similar complexities. Generally, the accuracy follows the complexity, whereas the speed roughly follows the year of publication rather than complexity or type.

4.2 Comparison of Models

Models with fastText. We investigated the binary model through ablation. The impacts of fastText are presented in the upper part of Table 4. **B**/E does not require any external data beyond PTB, which is comparable to models without a pre-trained GloVe (Pennington et al., 2014).

Then, we replaced `BiLSTM`$_{ori}$ with an FFNN to examine its effect. The results are in the bottom rows. The comparison proves whether the embeddings are collaborative for the orientation signals because FFNN regards each input independently.

Finally, we used a grid search to explore the hyperparameter space of our three-loss coefficients.

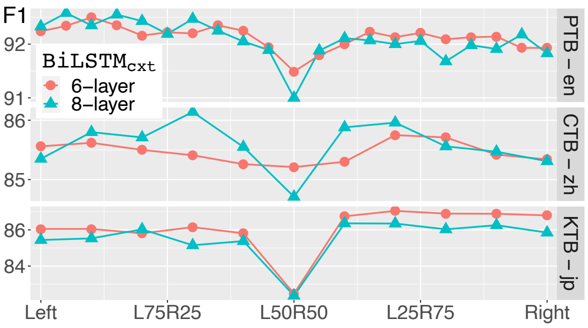

Figure 7: Probabilistic interpolations of two CNF factors to F1 scores. The capacity of `BiLSTM`$_{cxt}$ is almost saturated with 6 or 8 layers.

Figure 6 shows that the performance correlates to the orientation loss the most, but it is not overly sensitive to the hyperparameters.

Pre-trained language model. We compared the results using frozen fastText with those using frozen XLNet[5] in Table 5. The accuracy of the model increased along with the depth of `BiLSTM`$_{cxt}$, and it exhibited the most significant increase across all variants. Owing to XLNet, our complexities grew to $O(n^2)$.

We fine-tuned our models[6] and compared them with other parsers using fine-tuned language models. These are listed in Table 6.

4.3 Tree-Binarization Strategy

To reflect the branching tendency, our best single model for PTB was obtained on the dynamic L95R05 dataset. This dataset is a probabilistic interpolation between the left-factored dataset (for 95% chances) and a right-factored dataset (for 5% chances) in Figure 7. The best model for CTB appeared on the left side at L70R30, scoring 86.14, whereas the best for KTB was on the L30R70 dataset, scoring 87.05 with a 6-layer `BiLSTM`$_{cxt}$. Typically, the results for all the corpora had a minimum at L50R50. For English, the left "wing" was higher than the right; the opposite trend was observed for Japanese. For Chinese, no clear trend was obtained.

All studies described in the previous sections were conducted on the PTB L85R15 dataset.

---

[5] XLNet tokenizes words into sub-word fractions. For the frozen XLNet, using leftmost, rightmost, or averaged sub-word embeddings as the word input yielded similar results.

[6] For the fine-tuned XLNet, using either the leftmost or rightmost sub-word yielded similar results earlier. However, averaging sub-words produced F1 scores under 94.

Corpus Single Model	Penn Treebank Type	 sents/sec	 LP	 LR	 F1	Chinese Treebank Type	 LP	 LR	 F1
Watanabe and Sumita (2015)	T↑ (32)	-	-	-	90.7	T↑ (64)	-	-	84.3
Gómez-Rodríguez and Vilares (2018)	O	898	-	-	90.7	O	-	-	83.1
Cross and Huang (2016)	T↑ (1)	-	92.1	90.5	91.3	-	-	-	-
Liu and Zhang (2017c)	T↓ (16)	79.2	92.1	91.3	91.7	T↓ (16)	85.9	85.2	85.5
Stern et al. (2017)	C	75.5	93.0	90.6	91.8	-	-	-	-
Shen et al. (2018)	O↓ (1)	111.1	92.0	91.7	91.8	O↓ (1)	86.6	86.4	86.5
Charniak and Johnson (2005)	C	-	-	-	92.1	-	-	-	-
Ours (multi-branching)	O↑ (1)	1122.6	92.1	92.1	92.1	O↑ (1)	86.0	84.7	85.3
Ours (binary)	O↑ (1)	**1327.2**	**92.8**	**92.3**	**92.5**	O↑ (1)	85.8	86.2	86.0
Nguyen et al. (2020)	O↓ (1)	130.2	92.8	92.8	92.8	-	-	-	-
Kitaev and Klein (2018)	C	212.5	93.9	93.2	93.6	C	91.9	91.5	91.7
Wei et al. (2020)	O↓ (1)	155	94.1	93.3	93.7	O↓ (1)	89.9	87.4	88.7
Zhou and Zhao (2019)	C	226.3	93.9	93.6	93.7	C	92.3	92.0	92.2
Zhang et al. (2020)	C	1092	94.2	94.0	94.1	C	89.7	89.9	89.8

Table 3: Single-model results on PTB and CTB test datasets sorted by the F1 scores on PTB. Transition-based parsers, chart parser, and others are marked as T, C, and O, respectively; ↑ and ↓ denote bottom-up and top-down. The number in brackets indicates the beam size. Kitaev and Klein (2018) used Tesla K80, and the CTB scores are cited from Kitaev et al. (2019). Zhou and Zhao (2019) used GeForce GTX 1080 Ti (same condition).

Var	Specification	F1
B/e	without fastText initialization.	91.73
B/ϵ	with tuned official fastText.	91.69
B/E	with frozen fastText from PTB.	92.31
B/F	`BiLSTM`$_{ori}$ into `FFNN`$^{\star}_{ori}$.	88.97
B/L	`BiLSTM`$_{ori}$ with `BCE-LOSS`.	92.32

Table 4: Results of ablation studies on fastText (top) and `BiLSTM`$_{ori}$ (bottom) of the binary model.

Var	Frozen fastText F1	 sents/sec	Frozen XLNet F1	 sents/sec
B/0	65.02	1386.6	89.24	411.2
B/2	91.34	1350.0	93.74	398.4
B/6	92.54	1327.2	93.89	382.7

Table 5: Effectiveness of using frozen static word embeddings or dynamic sub-word language model and corresponding peak speed.

4.4 Complexity and Speed

To test our linear speed advantage, we inflated our training data with redundant nodes to resemble the triangular chart of CYK algorithm, as depicted in Figure 8 and Table 7. The parse in the triangular treebank has the worst-case complexity of $O(n^2)$. Meanwhile, training with linearity halved the training time, reduced memory usage, and canceled the length limit for our three corpora. There is a sheer difference between linearity and squared complexity.

Fine-Tuned Model	F1	sents/sec	Type
Kitaev and Klein (2018)	95.13	70.8	C
Kitaev and Klein (2020)	95.44	1200	T
Nguyen et al. (2020)	95.48	-	O↓
Zhang et al. (2020)	95.69	-	C
Wei et al. (2020)	95.8	-	O↓
Zhou and Zhao (2019)	96.33	64.8	C
Yang and Deng (2020)	96.34	71.3	T
Mrini et al. (2020)	96.38	59.2	C
B/0 (XLNet+FFNN)	95.72	411.2	O↑
B/2 (XLNet+BiLSTM)	94.67	398.4	O↑
M/0 (XLNet+FFNN)	95.44	369.4	O↑

Table 6: Improvements with pre-trained language models. We used a greedy search algorithm on single GeForce GTX 1080 Ti. Rows 6–8 are reported by Yang and Deng (2020) using GeForce GTX 2080 Ti. Kitaev and Klein (2020) used a cloud TPU with a beam search algorithm and a larger batch size.

5 Discussion

5.1 Model Structure

Our parsers comprise a neural encoder for scoring (i.e., Algorithm 1) and a non-neural decoder for searching. The decoder is a symbolic extension of the encoder in that both run in bottom-up manner, and the decoder interprets the scores as local-and-greedy decisions. Other neural parsers also fit a similar encoder–decoder framework. However, decoders with dynamic programming often include forward and backward processes heterogeneous to

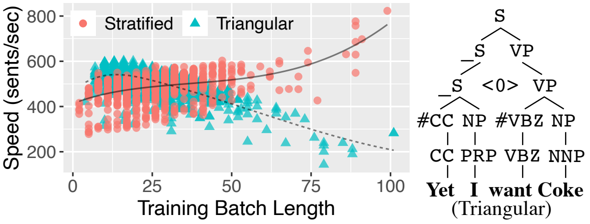

Figure 8: Linear complexity vs. squared complexity. Redundancy with placeholder "<0>" helps maintain the triangular shape.

Format	Time/150	Memory	OOM
Stratified	7.5 hours	3.3 GB	-
Triangular	15.9 hours	8.2 GB	100

Table 7: Training time and memory consumed by our two data formats. The time column indicates the time used for 150 training epochs with validations. Development F1 scores are approximately 92.4. The OOM column lists the length limit for preventing an out-of-memory error. Kitaev and Klein (2018) took 10 hours for 93 training epochs on our GeForce GTX 1080 Ti to yield their results.

their forward encoders (Kitaev and Klein, 2018, 2020). The encoder and decoder in our model and Shen et al. (2018) are more homogeneous and can be easily merged. Our parsers are bottom-up combinatory, while theirs was top-down splitting. Similar homogeneity can be found in an easy-first dependency parser (Goldberg and Elhadad, 2010).

Input component. In terms of encoder, Tables 4–6 examine the impact of $\texttt{BiLSTM}_{cxt}$ with fastText or XLNet, and the following conclusions can be drawn. 1) The top rows of Table 4 suggest that frozen fastText embeddings contain sub-word information, whereas tuning them disturbs the frozen information because the n-gram model is not part of our model. 2) Table 5 shows that the deeper the contextualization $\texttt{BiLSTM}_{cxt}$ (or XLNet), the better the results. 3) Tables 5 and 6 indicate that the tuning process for the pre-trained language models (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019) achieves a significant improvement.

Speed and size. One of our research goals was to achieve simplicity and efficiency. In terms of speed, our models parallelize more actions than transition-based parsers and have fewer computing nodes than chart parsers. In terms of size, our models contain approximately 4M parameters in addition to the 13M fastText (or 114M XLNet) pre-trained embeddings, which is fewer than those of Shen et al. (2018, 22M+) and Kitaev and Klein (2018, 26M). The recursiveness and productiveness of vector compositionality should account for the compact size.

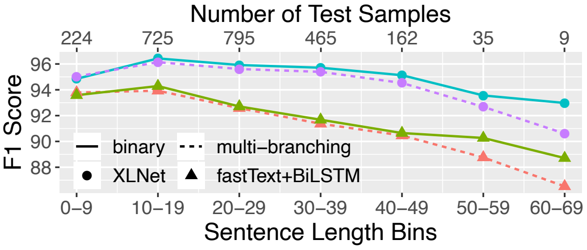

Figure 9: XLNet provides an overall improvement for all models and length bins. All models find it challenging to handle long sentences.

Vector compositionality. The performance of $\texttt{FFNN}^{\star}_{ori}$ is inferior to that of its RNN counterparts, suggesting that some information might not be encoded locally. Thus, the `COMPOSE` function should remain in a contextual form to collaboratively leverage the whole layer. However, BiRNN might still be a bottleneck for long-range orientation, as suggested in Figure 9. $\texttt{BiLSTM}_{chk}$ is a major weakness of the multi-branching model, especially for longer sentences.

5.2 Tree Binarization and Headedness

Tree binarization. Probabilistic interpolation with two CNF-factored datasets is effective for the three languages studied, as shown in Figure 7. Dynamic sampling allows the model to cover a wider range of composed vectors to improve its robustness to ambiguous orientations. Furthermore, it seems counterintuitive for human learners to obtain the best model using left-biased interpolation for a right-branching language or vice versa. However, for a neural model, balancing the frequency seems to be the key factor for improving performance (Sennrich et al., 2016; Zhao et al., 2018). The fact that the L50R50 dataset yielded the worst models also suggests that the balance should be based on a default orientation tendency. This could also be the reason why **mid-in** or **mid-out** did not improve the model.

Headedness. Figure 10 show the intermediate parses on the same sentence from our two models. They are typical examples in the output.

The binary model first combines determiners and their right neighbors rather than adjectives and nouns in noun phrases (blue spans). It also post-

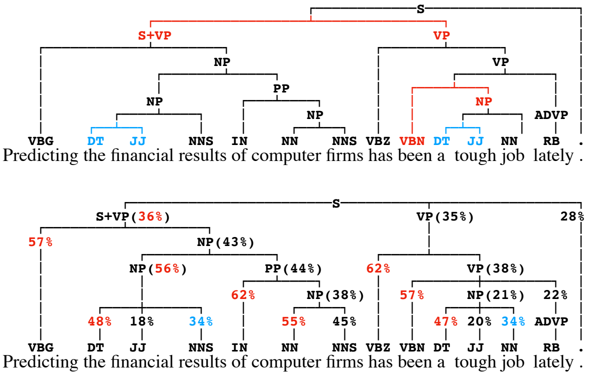

Figure 10: English internal constituents (top) and headedness (bottom) from our two models.

Parent (#)	Head child by maximum weight
NP (14.4K)	DT (4.5K); ⋆NP (4.3K); ⋆NNP (1.6K); ⋆JJ (922); ⋆NN (751); ⋆NNS (616); etc. (1.6K; 38 of 50 types with "⋆")
VP (6.8K)	VBD (1.5K); VB (1.4K); VBZ (1.0K); VBN (954); VBP (705); MD (523); VBG (387); VP (169); TO (81); etc.
PP (5.5K)	IN (5.0K); TO (397); etc.
S (3.8K)	VP (3.4K); S (194); NP (90); etc.
SBAR (1.2K)	IN (649); WHNP (395); WHPP (19); WHADVP (121); SBAR (15); etc.
ADVP (278)	RB (181); IN (30); RBR (25); etc.
QP (198)	CD (67); IN (65); RB (29); JJR (16);

Table 8: English headedness selection with our multi-branching model on PTB test set. "⋆" marks the absence of a DT child for its NP sisters. For quantifier phrases (QP), some non-quantifiers are more likely to be heads if they appear; e.g., adverbs (RB; e.g., "approximately"), prepositions (IN; e.g., "about"), and relative adjectives (JJR; e.g., "more than").

pones the combination with adjuncts such as punctuation and adverb (red spans). The high frequencies of determiners in noun phrases make them great attractors.

On the other hand, the multi-branching model places close attention on what the syntactic head is supposed to be. In the noun phrases, determiners receive the highest weight averages (red), and the nouns obtain the second (blue). This phenomenon suggests that an English noun phrase's syntactic role is mainly projected from the determiners, as discussed by Zwicky (1985). Table 8 provides more statistical support. For example, the model selects DT as an NP head if it is available; otherwise, nouns and adjectives are prominent heads. Chinese and Japanese parsers work similarly for their headedness. (See Appendix A.5.)

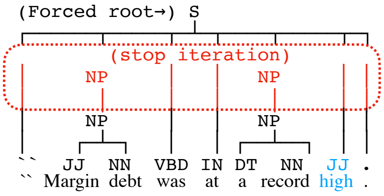

Figure 11: Failed parse from the multi-branching model. The model stops parsing and saves computations when it repeats the same chunking positions.

5.3 Error Analysis

The rate of an invalid parse is the last topic that we consider for our parsers. For the binary parser, fatal errors, such as frame-breaking orientations, appear at an early stage of training. However, the late 90% of training time contains very few errors, and our binary model is free from invalid parsing on the test set. For the multi-branching parser, it is observed that 11 out of 2,416 test parses are forests rather than parse trees when they are trained with fastText. However, the multi-branching parser with fine-tuned XLNet reduces the error count on the test set to 1.

We present a failed multi-branching parse with fastText, as shown in Figure 11. The postnominal adjective "high" is uncommon for English. Because the model did not group it with the adjacent "a record" to form an NP, the error propagated to higher layers (e.g., no PP as an adjunct to form a VP), causing the bad parse. It implies that the multi-branching model requires an appropriate predict-argument configuration to chunk.

6 Conclusion

We proposed a pair of neural combinatory constituency parsers. The binary one yields F1 scores comparable to those of recent neural parsers. The multi-branching one reveals constituency headedness. Both are simple and efficient with relatively high speeds. We also leveraged a pre-trained language model and CNF factors to increase the accuracy. We reflected the branching tendencies of three languages.

Acknowledgments

We extend special thanks to our reviewers for invaluable comments and Hayahide Yamagishi for initial discussions. This work has been partly supported by the Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science (JSPS KAKENHI); Grant Number 19K12099.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alastair Butler, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, Zhen Zhou, and Hong Zhu. 2012. Keyaki Treebank: phrase structure with functional information for Japanese. In *Proceedings of Text Annotation Workshop, 2012*.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of 6th Applied Natural Language Processing Conference*, pages 132–139.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the Conference of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180.

Ronan Collobert. 2011. Deep Learning for Efficient Discriminative Parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Proceedings*, pages 224–232.

James Cross and Liang Huang. 2016. Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

Yoav Goldberg and Michael Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 742–750.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent Parsing as Sequence Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual Constituency Parsing with Self-Attention and Pre-Training. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 3499–3505.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2676–2686.

Nikita Kitaev and Dan Klein. 2020. Tetra-Tagging: Word-Synchronous Parsing with Linear-Time Inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6255–6261.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic Programming Algorithms for Transition-Based Dependency Parsers. In *Proceedings of the 49th Conference of the Association for Computational Linguistics*, pages 673–682.

Jiangming Liu and Yue Zhang. 2017a. Encoder-Decoder Shift-Reduce Syntactic Parsing. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 105–114.

Jiangming Liu and Yue Zhang. 2017b. In-Order Transition-based Constituent Parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Jiangming Liu and Yue Zhang. 2017c. Shift-Reduce Constituent Parsing with Neural Lookahead Features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking Self-Attention: Towards Interpretability in Neural Parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 731–742.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq R. Joty, and Xiaoli Li. 2020. Efficient Constituency Parsing by Pointing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3284–3294.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433—-440.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715—-1725.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron C. Courville, and Yoshua Bengio. 2018. Straight to the Tree: Constituency Parsing with Neural Syntactic Distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1171–1180.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with Compositional Vector Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 818–827.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based Neural Constituent Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1169–1179.

Yang Wei, Yuanbin Wu, and Man Lan. 2020. A Span-based Linearization for Constituent Trees. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3267–3277.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Kaiyu Yang and Jia Deng. 2020. Strongly Incremental Constituency Parsing with Graph Neural Networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR*, abs/1906.08237.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and Accurate Neural CRF Constituency Parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 4046–4053.

Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong, and Hua Wu. 2018. Addressing Troublesome Words in Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 391–400.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar Parsing on Penn Treebank. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 2396–2408.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and Accurate Shift-Reduce Constituent Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 434–443.

Arnold M. Zwicky. 1985. Heads. *Journal of Linguistics*, 21(1):1–29.

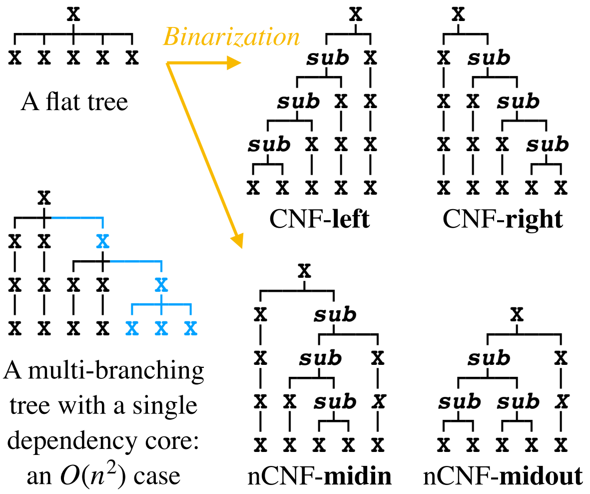

Figure 12: Examples used in the content. A flat tree is binarized with four factors. The binarization of CNF-**left**, CNF-**right**, or nCNF-**midin** creates binary trees with a single dependency core (i.e., a single *sub* thread), whose $O(n^2)$ complexity is the same with the tree without binarization in the lower left. Most nodes are relaying nodes. Meanwhile, nCNF-**midout** enables concurrent phrases with $O(n)$ complexity (i.e., multiple *sub* threads). However, the division tends to break a constituent into ungrammatical pieces, which confuses the model and does not lead to improvement.

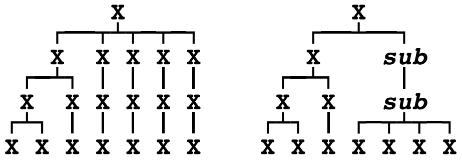

Figure 13: Adding *sub* nodes to make flat structure more efficient. Using the strategy as a new dynamic dataset also brings multi-branching model **M** a stable accuracy improvement with an F1 score of 92.36 on PTB. However, it has nothing to do with linguistic properties. We save it for a future study.

A Appendices

A.1 Compression Ratios and Linearity

Figure 12 presents examples for tree binarization and the worst case of $O(n^2)$ complexity. Figure 14 shows the overall linear data complexities in the three languages. Figures 15 & 16 and Table 9 indicate that, given a language and a factor, the compression ratio is stable and seldom affected by the sentence length.

The regressions for PTB and CTB show weak $O(n^2)$ tendencies; the quadratic coefficients can be either positive or negative. Meanwhile, KTB falls into the worst case, as shown in Figure 14. This is because KTB trees tend to have a flat structure on the right side of parses, as illustrated in Figure 17. Relaying nodes in the flat structure never combine until the final layer, creating strong $O(n^2)$ tendencies. As a result, all KTB datasets fall into the worst case, especially when binarized with the CNF-**left** factor.

A preprocess that groups the flat structure into the *sub* category can prevent considerable quadratic impacts on all datasets. All $O(n^2)$ tendencies are largely weakened across three corpora, and all linear coefficients drop significantly, as illustrated on the right of Figure 14. The preprocess cannot eradicate the worst case in KTB. However, all linear coefficients' magnitudes are at least hundreds of times larger than those of the quadratic terms. In our sub-quadratic case, 200 words lead to approximately 1.5K nodes. Meanwhile, a sentence with n words has a triangular chart with $\frac{n(n+1)}{2}$ nodes, whose quadratic coefficient is 0.5. In this case, 200 words lead to approximately 20K nodes.

A.2 Experiment Setting

The treebanks PTB and CTB have been widely used for experiments. For PTB, sections 2-21 were used for training, section 22 for development, and section 23 for testing. For CTB, articles 001-270 and 440-1151 were used for training, 301-325 for development, and 271-300 for testing. There is no widely accepted data split for the KTB corpus, except for some probabilistic divisions, because KTB contains mixed data from sources such as newswires, book digests, and Wikipedia. We randomly reserved 2,075 samples for development, 1,863 samples for testing, and the remaining 3.3 million as training samples. Few sentences in the training sets were longer than 100 words (3 of 40K in PTB; 96 of 17K in CTB; 55 of 33K in KTB). Frozen English (wiki.en.bin), Chinese (cc.zh.300.bin), and Japanese (cc.ja.300.bin) embeddings were used for PTB, CTB, and KTB, respectively[7]. We fed fastText with the PTB text to train `cbow` instead of `skipgram` embeddings for **B**/E with their default settings for 50 epochs.

The batch size was 80, and sentences longer than 100 words were excluded for the triangular data to avoid out-of-memory (OOM) errors on a single GeForce GTX 1080 Ti with 11 GB. We froze XLNet to train our model and then tuned XLNet from the 5-th epoch. We doubled the batch size at the inference phase to 160.

---

[7] `https://fasttext.cc/`

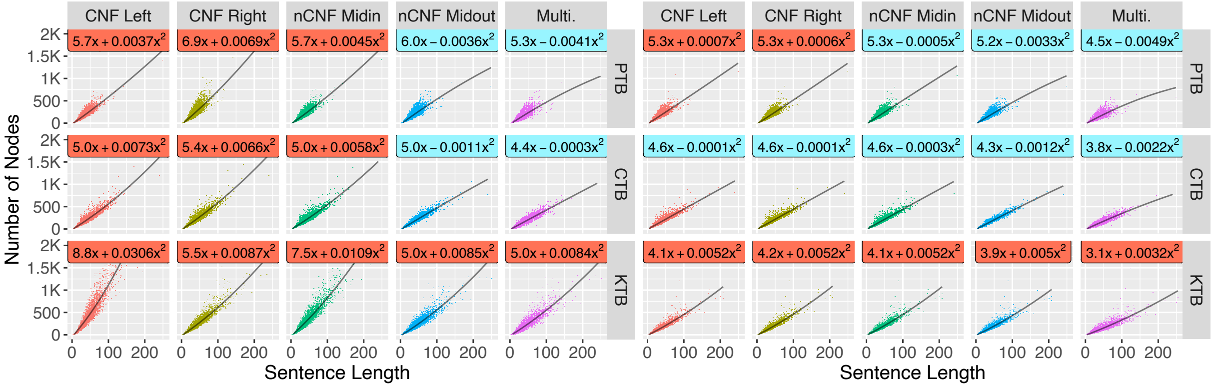

Figure 14: Left: the empirical complexities related to Figures 15 & 16. Linear regressions are shown on a light blue background when the quadratic terms are negative. Right: resultant complexities after a preprocessing that groups the flat structure into sub-constituent before stratification. (See Figure 13.)

Factor	**Left**	**Right**	**Midin**	**Midout**	**Multi.**
			All layers		
PTB	0.77 ±0.11	0.79 ±0.11	0.78 ±0.11	0.77 ±0.11	0.73 ±0.20
CTB	0.77 ±0.11	0.77 ±0.11	0.76 ±0.11	0.74 ±0.11	0.70 ±0.20
KTB	0.82 ±0.12	0.75 ±0.12	0.79 ±0.12	0.73 ±0.12	0.69 ±0.29
			Layers longer than 40		
PTB	0.78 ±0.04	0.80 ±0.04	0.77 ±0.04	0.74 ±0.04	0.69 ±0.08
CTB	0.79 ±0.04	0.80 ±0.04	0.78 ±0.04	0.77 ±0.04	0.76 ±0.07
KTB	0.90 ±0.04	0.80 ±0.05	0.86 ±0.04	0.77 ±0.06	0.84 ±0.07

Table 9: Mean and standard deviation of compression ratios of Figures 15 & 16. Longer layers have converged deviations. The last column came from the multi-branching treebanks without a binarizing factor.

We used the Adam optimizer with a default learning rate of 10^{-3}, while we opted for the XLNet's Adam hyperparameters when tuning the pre-trained XLNet (e.g., their learning rate was 10^{-5}). We adopted a warm-up period for one epoch and a linear decrease after the 15-th decrease since the last best evaluation. The recurrent dropout rate was 0.2; other dropout probabilities for FFNNs were set to 0.4. For model selection, the training process terminated when the development set did not improve above the highest score after 100 consecutive evaluations. The Evalb program[8] was used for F1 scoring.

We demonstrated score profiles for our main models in Table 10. The discrepancy in F1 scores and difference between precision and recall are relatively small on the PTB development and test sets.

Input		**Development**		**Test**	
Comp.	**M.**	**F1**	$P-R$	**F1**	$P-R$
Frozen	**B**	92.50	0.00	92.54	+0.56
fastText	**M**	92.10	−0.35	92.10	−0.03
Tuned	**B**	95.64	−0.05	92.72	+0.19
XLNet	**M**	95.34	−0.15	92.44	+0.30

Table 10: F1 scores and differences in precision and recall ($P-R$) on the PTB development and test sets.

A.3 Variants of Binary Compose

If we choose the relay instruction in line 12 of Algorithm 2, additive vector compositionality is retained (Mikolov et al., 2013) as the naïve **ADD** variant in lines 5–6 of Algorithm 4. The model can infer a full tensor tree; however, **ADD** causes the vector magnitude to increase with the tree height cumulatively. This is unwanted in the recurrent or recursive neural network.

Therefore, we examined a learnable $\texttt{FFNN}_{multi}$ with Sigmoid activation to perform gate-style interpolation in five variants **NS**, **NV**, **CS**, **CV**, and **BV** as described in lines 8–17. When a variant takes <u>n</u>o input and produces a <u>s</u>calar interpolation parameter λ, we consider this case **NS**. ("$\varnothing$" is a placeholder for no input.) Meanwhile, **CV** indicates <u>c</u>oncatenated input and <u>v</u>ectorized interpolation. **BV** is a variant that involves a <u>b</u>iaffine tensor

---

[8] `https://nlp.cs.nyu.edu/evalb/`

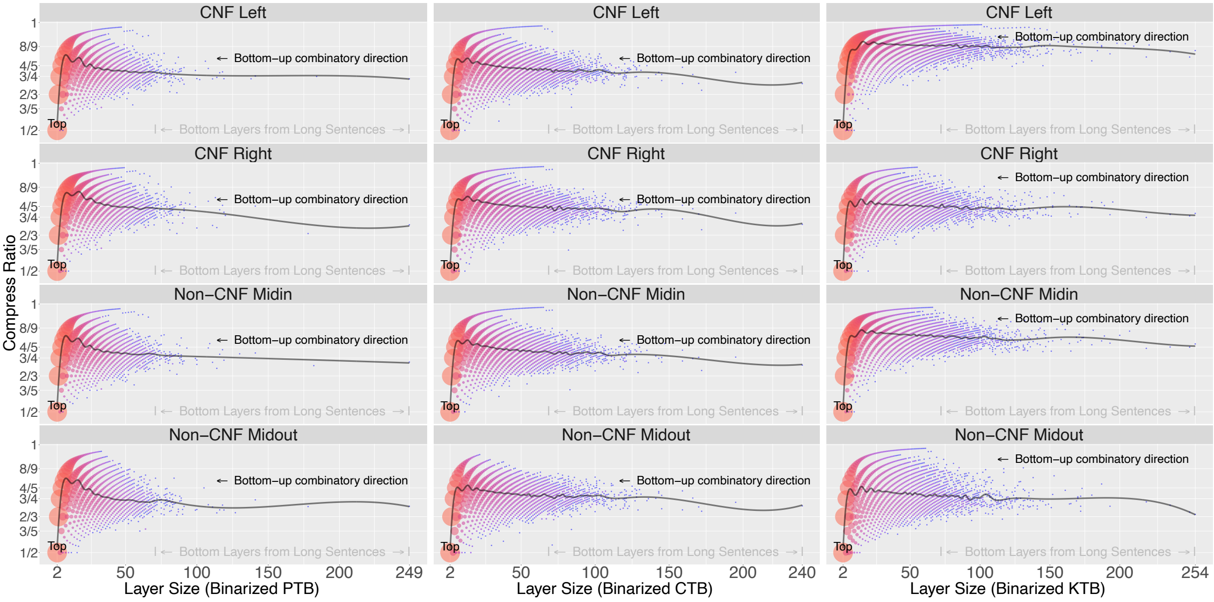

Figure 15: Binarized corpora with four factors. Curved tiers can be observed in each plot. For example, the leftmost tier is composed of $\frac{n-1}{n}$ (followed by $\frac{n-2}{n}$, $\frac{n-3}{n}$, and so on). The dots in this tier range from a high compression ratio of 0.5 to the least efficient ones in their corpus. Efficient dots are more populated, judging by their sizes and colors. All statistics yield stable means, which are also presented in Table 9.

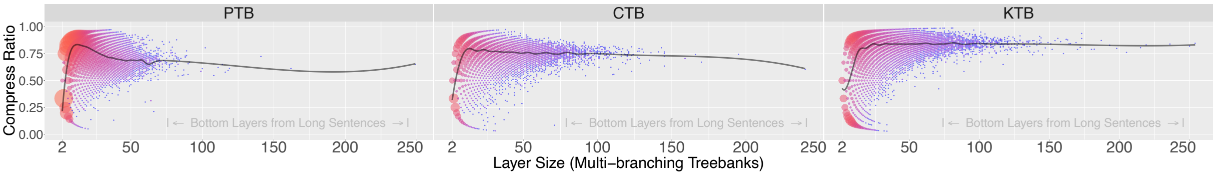

Figure 16: Multi-branching corpora. Curved tiers appear more symmetric and all statistics still yield stable means.

Algorithm 4: Binary Compose Variants

```
Function BINARY($o_L, o_R$, $x_L, x_R$; Var):
    $x \leftarrow o_L \cdot x_L + (1 - o_R) \cdot x_R$
    if $o_L + (1 - o_R) = 1$ then                    // relay
        return $x$
    else if Var is ADD then                          // ADD
        return $x$
    else
        if Var is NS or NV then                      // No input
            $\lambda \leftarrow \sigma\ \mathrm{FFNN}_{binary}(\varnothing)$
        else if Var is CS or CV then                 // Concat...
            $\lambda \leftarrow \sigma\ \mathrm{FFNN}_{binary}(x_L \oplus x_R)$
        else if Var is BV then                       // Biaffine
            $\lambda \leftarrow \sigma\ \mathrm{FFNN}_{binary}(x_L, x_R)$
        if Var is NV, CV, or BV then                 // Vector $\lambda$
            $x \leftarrow \lambda \odot x_L + (1 - \lambda) \odot x_R$
        else if Var is NS or CS then                 // Scalar $\lambda$
            $x \leftarrow \lambda \cdot x_L + (1 - \lambda) \cdot x_R$
        return $x$                                   // NS NV CS CV BV
```

operation. **CV** is our default `BINARY` variant; the experiments for these variants are presented in Table 11.

In terms of the F1 score, the most competitive variants of **CV** are **BV** and **NV**, suggesting that fine interpolation can effectively facilitate vector compositionality. The similarity in results of **CS**, **NS**, and **ADD** validate this suggestion. This indicates that vector compositionality is not as trivial as an additive function at the scalar level, and a matrix operation is sufficient. **BV** is the costliest variant with a tensor operation that runs very slowly (30 sents/sec).

Var	Specification	F1
BV	Biaffine inputs for vector λ.	92.53
CV	$x_L \oplus x_R$ as input for vector λ.	**92.54**
CS	$x_L \oplus x_R$ as input for scalar λ.	91.83
NV	No input; bias vector λ.	92.36
NS	No input; bias scalar λ.	91.95
ADD	$x_L + x_R$	91.86

Table 11: Compositionality of the `BINARY` function.

A.4 Recovering Symbolic Tree

To obtain the final tree representation, we initialized the working place with leaves of words and

Algorithm 5: Recovering a Symbolic Tree

```
Function REC ($x_{0:n}$, $t_{0:n}$, $l^k_{0:n_k}$, $o^k_{0:n_k}$ or $c^k_{0:n_k+1}$):
    for $i \leftarrow 0$ to $n-1$ do
        $tree_i \leftarrow$ TREE ($t_i, x_i$)
    for $j \leftarrow 0$ to $k-2$ do
        if binary parsing then                          // BINARY
            for $i \leftarrow 0$ to $n_j - 1$ do
                if $o^j_i + (1 - o^j_{i+1}) = 2$ then
                    Combine $tree_{i:i+2}$ under $l^{j+1}_{parent\_of(i,i+1)}$
        else                                            // MULTI-BRANCHING
            foreach $chk$ in $c^j_{0:n_j+1}$ do
                Combine $tree_{chk}$ under $l^{j+1}_{parent\_of(chk)}$
    Expand unary and flatten sub labels for $tree_0$
    return $tree_0$
```

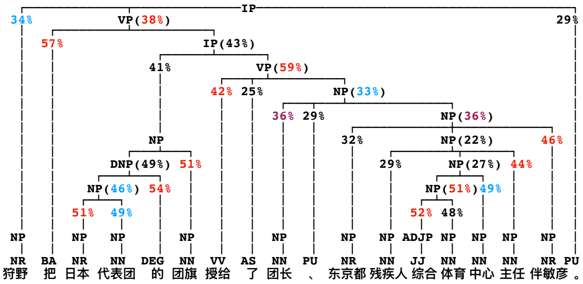

Translation: Kano awarded (授给/了) the Japanese delegation's flag to the head of the delegation (团长) and director (主任) of the Tokyo Comprehensive Sports Center for the Disabled, Toshihiko Ban.

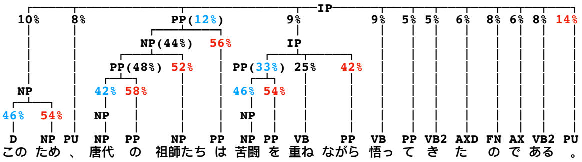

Translation: For this reason, the Tang dynasty’s (の) ancestors (祖師たち/は-subject) have come to realize (悟っ) it through struggles (苦闘/を-object).

Figure 17: Chinese (top) and Japanese (bottom) parses from the multi-branching model.

predicted POS tags. Two symbolic rules were used to modify the labels and construct sub-trees, as described in Algorithm 5. 1) The collapsed unary branches were expanded to their original structure by splitting at the plus marks (e.g., `SBAR+S` into `SBAR` and `S`). 2) ‘the label is a *sub*’ excluded the repeated labels and relayed sub-trees. These rules enabled a single $tree_0$ as the final output.

A.5 Chinese and Japanese Headedness

Figure 17 presents two non-English parses from the multi-branching model. Both the Chinese and Japanese languages possess functional markers that receive high attention (percentage and words in red), such as the second character tagged with `BA` in Chinese, and Japanese case markers tagged with `PP`. Interestingly, the Chinese verb (i.e., the one meaning “awarded”) received the highest attention, whereas Japanese verbs (i.e., two sub-words tagged with `VB`) did not. We supposed the reason behind this is that Japanese sentences drop the `VB`s and other heads more often than Chinese. The coordinated `NP`s in the Chinese parse (i.e., two words meaning “head” and “director”) received equal attention weights.

Moreover, two trees show their branching tendencies: Chinese is **midin**-alike; Japanese is a **left**-branching language, and KTB has a large flat structure on the right.

Learning Shared Semantic Space for Speech-to-Text Translation

Chi Han[1], **Mingxuan Wang**[1], **Heng Ji**[2], **Lei Li**[1]
[1] ByteDance AI Lab, [2] University of Illinois at Urbana-Champaign
`{hanchi.me, wangmingxuan.89, lilei.02}@bytedance.com`
`hengji@illinois.edu`

Abstract

Having numerous potential applications and great impact, end-to-end speech translation (ST) has long been treated as an independent task, failing to fully draw strength from the rapid advances of its sibling - text machine translation (MT). With text and audio inputs represented differently, the modality gap has rendered MT data and its end-to-end models incompatible with their ST counterparts. In observation of this obstacle, we propose to bridge this representation gap with Chimera. By projecting audio and text features to a common semantic representation, Chimera unifies MT and ST tasks and boosts the performance on ST benchmarks, MuST-C and Augmented Librispeech, to a new state-of-the-art. Specifically, Chimera obtains 27.1 BLEU on MuST-C EN-DE, improving the SOTA by a +1.9 BLEU margin. Further experimental analyses demonstrate that the shared semantic space indeed conveys common knowledge between these two tasks and thus paves a new way for augmenting training resources across modalities. [1]

1 Introduction

Speech-to-text translation (ST) takes speech input in a source language and outputs text utterance in a target language. It has many real-world applications, including automatic video captioning, simultaneous translation for international conferences, etc. Traditional ST approaches cascade automatic speech recognition (ASR) and machine translation (MT) (Sperber et al., 2017, 2019; Zhang et al., 2019; Beck et al., 2019; Cheng et al., 2019). However, cascaded models often suffer from the issues of error propagation and translation latency. As a result, there have been a series of recent attempts on end-to-end speech-to-text translation (Liu et al., 2019, 2018; Weiss et al., 2017; Bérard et al., 2018; Duong et al., 2016; Jia et al., 2019; Dong et al., 2021b; Wang et al., 2020b). The end-to-end approaches learn a single unified model, which is easier to deploy, has lower latency and could potentially reduce errors.

However, it remains a challenge for end-to-end ST to catch up with their cascaded counterparts in performance. We argue that the root cause is the gap between the two modalities, speech and text. Although they both encode human languages, they are dissimilar in both coding attributes (pitch, volume, and intonation versus words, affixes, and punctuation) and length (thousands of time frames versus tens of words). This issue is further coupled with the relatively smaller amount of parallel data for ST than for MT.

To tackle these challenges, we resort to making use of the additional available bilingual data for MT. Our hypothesis is, to better leverage MT data, an ideal model should be able to bridge the representations between speech and text. Motivated by this intuition, we propose Chimera, a text-speech shared semantic memory network. It learns a semantic memory by projecting features from both modalities into a shared semantic space. This approach unifies ST and MT workflows and thus has the advantage of leveraging massive MT corpora as a side boost in training. It can also use speech-text pairs to align the semantic memories from two modalities.

This idea of a unified text-speech representation also finds its neural basis as suggested by recent evidence from functional neuroimaging (van Atteveldt et al., 2004; Spitsyna et al., 2006; Shankweiler et al., 2008). Specifically, van Atteveldt et al. (2004); Spitsyna et al. (2006) identifies certain regions in brain that the processing stream for speech sounds and visual texts converge at. Shankweiler et al. (2008) further verifies that

---

[1] All codes, data, and resources will be made released at https://github.com/Glaciohound/Chimera-SLT.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2214–2225
August 1–6, 2021.

the size of such convergence sites correlates positively with the subjects' reading skills. Coincidentally, at these convergence sites also found regions responsive to downstream activities such as lexical and semantical word recognition (Binder et al., 2003) and spontaneous generation of speech (Blank et al., 2002). The evidence establishes the pivotal role of a modality-agnostic converged representation in language activities in brain.

This intuition lacks exploration in previous studies, with only a few exceptions (Indurthi et al., 2019; Liu et al., 2020), possibly due to the difficulties aforementioned and marginal improvements.

Our results show that Chimera achieves new state-of-the-art results on all of 8 translation directions in the benchmark datasets MuST-C and Augmented LibriSpeech. Specifically, Chimera obtains a 27.1 BLEU score on MuST-C EN-DE, which surpasses the best result ever reported by up to +1.9 BLEU. We also provide results under variations and ablations and validate our model design ideas by detailed analysis, as well as visualizing the semantic space Chimera has learned.

Our work makes the following contributions. First, we propose Chimera, which is able to bridge the modality gap between speech and text. Second, we derive a novel bi-modal contrastive training task to learn an alignment between semantic memories of two modalities. Finally, Chimera achieves a new state-of-the-art performance on the MuST-C benchmark and demonstrates its efficacy in learning modality-agnostic semantic representations.

2 Related Work

End-to-end ST Since its first proof-of-concept work (Bérard et al., 2016; Duong et al., 2016), solving Speech Translation in an end-to-end manner has attracted extensive attention (Vila et al., 2018; Salesky et al., 2018, 2019; Di Gangi et al., 2019b; Bahar et al., 2019a; Di Gangi et al., 2019c; Inaguma et al., 2020). Standard training techniques such as pretraining (Weiss et al., 2017; Bérard et al., 2018; Bansal et al., 2018; Stoian et al., 2020; Wang et al., 2020a; Pino et al., 2020), multi-task training (Vydana et al., 2021; Le et al., 2020; Tang et al., 2021), meta-learning (Indurthi et al., 2019), and curriculum learning (Kano et al., 2018; Wang et al., 2020b) have been applied. As ST data are expensive to collect, Jia et al. (2019); Pino et al. (2019); Bahar et al. (2019b) augment synthesized data from ASR and MT corpora. Methods utilizing trained models, such as knowledge distillation (Liu et al., 2019) and model adaptation (Di Gangi et al., 2020), have also been shown to be effective. Among these attempts, (Indurthi et al., 2019; Le et al., 2020; Liu et al., 2020) are most related to ours, as they also attempt to train models on ASR or MT data. However, they both lack pivotal modules in model design to semantically bridge the gap between audio and text, and could thus suffer from modality mismatch in representations.

Cascaded ST The cascaded method is a more long-standing trend in ST (Sperber et al., 2017; Jan et al., 2018). To alleviate its innate problem of error propagation, Cheng et al. (2018, 2019) introduce synthetic ASR-related errors and perturbations. On the other hand, some post-processing techniques such as re-segmentation (Matusov et al., 2006), punctuation restoration (Fügen, 2008), and disfluency detection (Fitzgerald et al., 2009) are proposed to fix flaws or errors that occurred during the translation.

Cross-Lingual Techniques Techniques in multilingual tasks is also related to ours, as they aim at extracting common features out of sources from different representations (which, in this case, is language diversity) as well. However, multilingualism lacks key difficulties as observed in audio-text modality gap as discussed before. (Lu et al., 2018) and (Vazquez Carrillo et al., 2019) are early attempts by building an LSTM-based attentional interlingua. Yu et al. (2018); Yang et al. (2019) uses a similar cosine-based loss for multilingual training. Zhu et al. (2020) is probably more similar in method to ours, but Chimera is more simple in terms of model and objectives, and the memories in Chimera are additionally designed to focus on specific semantic categories.

3 Proposed Method: Text-Speech Shared Semantic Memory Network

3.1 Speech Translation Overview

An ST corpus usually consists of a set of triplet data $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{z}_i, \mathbf{y}_i)\}$. Here $\mathbf{x}_i$ is the audio wave sequence, $\mathbf{z}_i$ is the transcript sequence and $\mathbf{y}_i$ is the translation sequence in the target language. As a benefit of shared semantic projection, Chimera is able to leverage large-scale MT training corpora $\mathcal{T} = \{(\mathbf{u}_i, \mathbf{v}_i)\}$, where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the source and target text sequences.

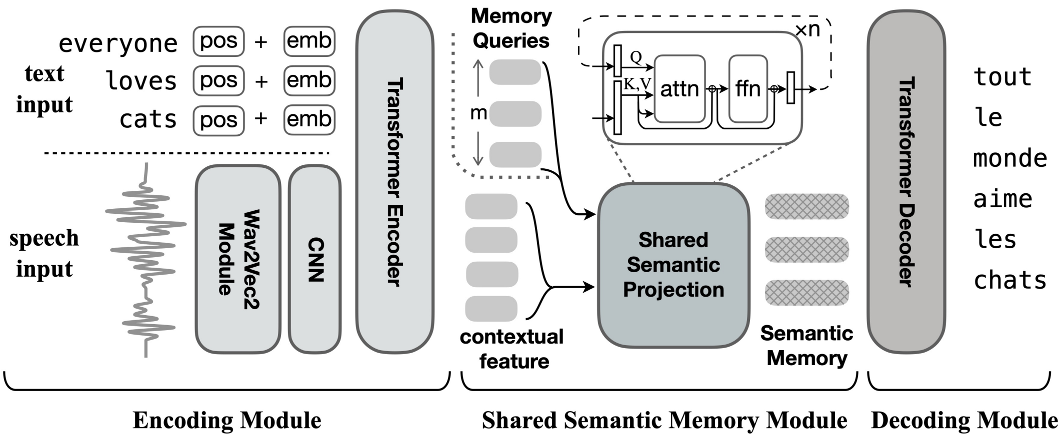

Figure 1: An overview of the proposed Chimera. The Encoder Module contains Word embedding for text input, and Wav2vec2 sub-module for speech input. The shared semantic projection Module uses its memory query to produce semantic memory with fixed-size representation from contextual features. The Decoder Module decodes translation from the semantic memory.

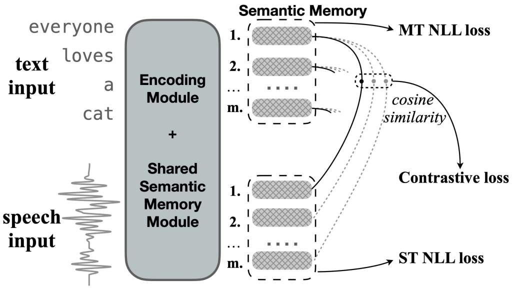

Figure 2: Besides MT and ST translation loss, we adopt a bi-modal contrastive loss to help aligning the semantic memories from text and speech. In short, among semantic memory elements from both sides of paired speech and transcript, the contrastive loss maximizes the cosine similarity between the same semantic memory element.

3.2 Chimera Architecture

Figure 1 illustrates the structure of Chimera. It consists of three major components, an encoding module, a shared semantic projection module, and a decoding module.

Encoding Module Different from that of a conventional ST model, the encoding module of Chimera accepts either speech or text as input. For text input, we use word embeddings plus positional embeddings. For speech input, we use a pretrained Wav2Vec2 (Baevski et al., 2020) to extract speech features. As the speech features can be very long, we apply an additional 1-dimensional strided CNN to reduce the length. Both speech and text branches share a common subsequent Transformer encoder (Vaswani et al., 2017). The final output of the encoding module is contextual features.

Shared Semantic Projection Module Shared semantic projection plays a pivotal role in Chimera. The contextual features of speech and text may follow different distributions and of different lengths. Ideally, the shared semantic projection computes a constant number of semantic features as its output semantic memories.

This module take the contextual features out from the encoding module as input and then output semantic memories of fixed length m. It consists of n attentional layers. It keeps a tuple of m trainable input-dependent memory queries to indicate the types of desired semantic information, which is used as the initial “memories”. Uni-modal contextual features serves as attention “keys” and “values”, while memories serves as attention “queries“. Memories are iteratively fed to the n shared semantic projection layers, with each layer output used as input to next layer. The final output is used as the semantic memory.

$$\mathbf{I}_0 = \mathbf{M}_0 \in \mathbb{R}^{m\times d} \tag{1}$$

$$\mathbf{K}_i = \mathbf{V}_i = \hat{\mathbf{H}} \in \mathbb{R}^{l\times d} \tag{2}$$

$$\mathbf{I}_{i+1} = \mathbf{O}_i = \mathrm{Attn}(\mathbf{I}_i, \mathbf{K}_i, \mathbf{V}_i) \in \mathbb{R}^{m\times d} \tag{3}$$

where $\mathbf{M}_0$, $\hat{\mathbf{H}}$, $\mathbf{I}_i$ and $\mathbf{O}_i$ denote the memory queries, contextual features, the input, and the output of each layer, respectively. l is the length of contextual features. d is the shared vector dimension. The top-most output $\mathbf{O}_n$ is finally fed into the decoding module.

Decoding Module The decoding module contains a conventional Transformer decoder. The only difference is that the input is the fixed-size $\mathbf{O}_n$, which can possibly come from either speech or text.

3.3 Training Objectives

The training objective of Chimera consists of three aspects, with their supervision signals coming from speech-to-text translation data $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, text machine translation data $(\{\mathbf{u}_j, \mathbf{v}_j)\}$ and $(\{\mathbf{z}_i, \mathbf{y}_i)\}$ and the speech-transcript pairs $\{(\mathbf{x}_i, \mathbf{z}_i)\}$.

Speech-to-Text Translation Training The workflow of Chimera in Speech Translation is straightforward. The training objective is negative log-likelihood on speech-to-text translation data $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ as the loss function.

$$\mathcal{L}_{\text{ST}} = -\mathbb{E}_{\mathbf{x},\mathbf{y}} \log \mathrm{P}(\mathbf{y}|\mathbf{x}) \tag{4}$$

Text Machine Translation Training Chimera is also able to train on MT corpus because of the unification of speech and text representations. Similar to ST training, the shared semantic projection module projects the contextual text features to the shared semantic space, which are then taken by the decoding module to output a translation.

$$\mathcal{L}_{\text{MT}} = -\mathbb{E}_{\mathbf{u},\mathbf{v}} \log \mathrm{P}(\mathbf{v}|\mathbf{u}) \tag{5}$$

Bi-modal Contrastive Training The motivation of Chimera design is to bridge the speech and text representations. We introduce dual-modal contrastive training to learn an alignment between representations from speech and text as illustrated in Figure 2. First, semantic memories from two inputs are computed. Then for each text semantic memory, $\mathbf{M}_i^{text}$, we compute its cosine similarities with all speech semantic memory $\{\cos(\mathbf{M}_i^{text}, \mathbf{M}_j^{speech})\}$. They are then fed into a softmax function. The loss function maximizes the item from matched pairs $(\mathbf{M}_i^{text}, \mathbf{M}_i^{speech})$. Finally, the loss is summed across all text memory items and vise versa.

$$\begin{aligned}\mathcal{L}_{\text{ctr}} = &-\mathbb{E}_{\mathbf{x},\mathbf{z}} \sum_i \ln \frac{e^{\tau \cos(\mathbf{M}_i^{\text{text}}, \mathbf{M}_i^{\text{speech}})}}{\sum_j e^{\tau \cos(\mathbf{M}_i^{\text{text}}, \mathbf{M}_j^{\text{speech}})}} \\ &-\mathbb{E}_{\mathbf{x},\mathbf{z}} \sum_j \ln \frac{e^{\tau \cos(\mathbf{M}_j^{\text{speech}}, \mathbf{M}_j^{\text{text}})}}{\sum_i e^{\tau \cos(\mathbf{M}_j^{\text{speech}}, \mathbf{M}_i^{\text{text}})}}\end{aligned} \tag{6}$$

Intuitively, the contrastive loss forces the pair $(\mathbf{M}_i^{text}, \mathbf{M}_i^{speech})$ to project semantic memories close to each other. In the meantime, the softmax function trains the model to maintain diversity among semantic memories.

The final loss is a weighted sum of each loss:

$$\mathcal{L} = \lambda_{\text{ST}}\mathcal{L}_{\text{ST}} + \lambda_{\text{MT}}\mathcal{L}_{\text{MT}} + \lambda_{\text{ctr}}\mathcal{L}_{\text{ctr}} \tag{7}$$

4 Experiments

We conduct experiments on the benchmark MuST-C and, as a validation of model design, carry out ablation studies and visualize the representations Chimera has learned.

4.1 Dataset and Preprocessing

MuST-C (Di Gangi et al., 2019a) is a multilingual speech translation corpus with triplet data sources: source audio, transcripts, and text translations. MuST-C contains translations from English (EN) to 8 languages: Dutch (NL), French (FR), German (DE), Italian (IT), Portuguese (PT), Romanian (RO), Russian (RU), and Spanish (ES). With each pair consisting of at least 385 hours of audio recordings, to the best of our knowledge, MuST-C is currently the largest speech translation dataset available for each language pair. It includes data from English TED talks with manual transcripts and translations at the sentence level. We use the dev and tst-COMMON sets as our development and test data, respectively.

Augmented LibriSpeech Dataset (En-Fr) (Kocabiyikoglu et al., 2018) is composed of aligned e-books in French and their human reading in English. It provides typical triplet data of English speech, transcript and French text. Following the setting of (Liu et al., 2019), we utilize the 100h hours of clean train set as training data, and use the original 2 hours of dev set and and 4 hours of test set.

Machine Translation Datasets After bridging the modality gap, Chimera has the potential power to utilize Machine Translation resources. Therefore we incorporate data from WMT, OpenSubtitles (Lison and Tiedemann, 2016) and OPUS100 (Zhang et al., 2020b) translation tasks. Specifically, we use WMT 2014 (Bojar et al., 2014) [2] for EN-DE, EN-FR, EN-RU and EN-ES, WMT 2016 (Bojar et al., 2016) [3] for EN-RO, and OPUS100 [4] for

---

[2] downloadable at http://www.statmt.org/wmt14/translation-task.html

[3] downloadable at https://www.statmt.org/wmt16/translation-task.html

[4] downloadable at http://opus.nlpl.eu/opus-100.php

Model	External Data			MuST-C EN-X							
	Speech	ASR	MT	EN-DE	EN-FR	EN-RU	EN-ES	EN-IT	EN-RO	EN-PT	EN-NL
FairSeq ST †	×	×	×	22.7	32.9	15.3	27.2	22.7	21.9	28.1	27.3
Espnet ST ‡	×	×	×	22.9	32.8	15.8	28.0	23.8	21.9	28.0	27.4
AFS *	×	×	×	22.4	31.6	14.7	26.9	23.0	21.0	26.3	24.9
Dual-Decoder $^\diamond$	×	×	×	23.6	33.5	15.2	28.1	24.2	22.9	**30.0**	27.6
STATST $^\sharp$	×	×	×	23.1	-	-	-	-	-	-	-
MAML $^\flat$	×	×	✓	22.1	34.1	-	-	-	-	-	-
Self-Training $^\circ$	✓	✓	×	25.2	34.5	-	-	-	-	-	-
W2V2-Transformer *	✓	×	×	22.3	34.3	15.8	28.7	24.2	22.4	29.3	28.2
Chimera Mem-16	✓	×	✓	25.6	35.0	16.7	30.2	24.0	23.2	29.7	28.5
Chimera	✓	×	✓	**27.1** $^\bullet$	**35.6**	**17.4**	**30.6**	**25.0**	**24.0**	**30.2**	**29.2**

Table 1: Main results on tst-COMMON subset on all 8 languages in MuST-C dataset. "Speech" denotes unlabeled audio data. $^\bullet$: the result uses a mixed WMT14+OpenSubtitles data for MT pre-training. EN-DE Among the baselines, † shows results from Ott et al. (2019), ‡ from Inaguma et al. (2020), * from Zhang et al. (2020a), $^\diamond$ from Le et al. (2020), $^\sharp$ from Liu et al. (2020), $^\flat$ from Indurthi et al. (2019), and $^\circ$ from Pino et al. (2019). * shows results of a simple baseline model by combining a Wav2Vec2 module (Baevski et al., 2020) and a Transformer model, which could be viewed as the "no external data" version of Chimera.

Model	External Data			
	Speech	ASR	MT	EN-FR
W2V2-T *	✓	×	×	6.4
TCEN †	×	×	×	17.1
LSTM ‡	×	✓	✓	17.0
AFS $^\circ$	×	×	×	17.2
Multilingual *	×	✓	×	17.6
Transformer $^\perp$	×	✓	×	17.7
Curiculum $^\perp$	×	✓	×	18.0
COSTT $^\flat$	×	×	✓	18.2
LUT $^\diamond$	×	✓	×	18.3
STAST $^\sharp$	×	✓	×	18.7
Chimera	✓	×	✓	**19.4**

Table 2: Results on LibriSpeech English-French dataset. * is the same W2V2-Transformer baseline as in Table 1. † is from Wang et al. (2020a), ‡ from Bahar et al. (2019b), * from Inaguma et al. (2019), two baselines under $^\perp$ from Wang et al. (2020b), $^\flat$ from Dong et al. (2021a), $^\diamond$ from Dong et al. (2021b), $^\circ$ from Zhang et al. (2020a) and $^\sharp$ from Liu et al. (2020).

Model	External Data			MuST-C
	Speech	ASR	MT	EN-DE
W2V2-T + Dec PT	✓	×	WMT14	22.2
W2V2-T + KD	✓	×	WMT14	24.6
Chimera	✓	×	WMT14	**26.3**

Table 3: Comparison with other baselines utilizing external MT data on MuST-C EN-DE. "Dec PT" pretrains decoder on MT corpus; "KD" adopts the knowledge distillation technique used in Liu et al. (2019)

EN-PT, EN-IT, and EN-NL, as pretraining corpora. We additionally evaluate OpenSubtitles as EN-DE MT data to test the impact of MT corpus selection. WMT 2014 dataset provides at least 4 million sentences of translation data in each language pair. WMT 2016 contains less, around 600k for EN-RO direction. OPUS100 has around 1M sentences for each sentence pair. OpenSubtitles provides 22M sentences for EN-DE.

Pre-processing of Data and Evaluation For speech input, the 16-bit raw wave sequences are normalized by a factor of 2^{15} to the range of $[-1, 1)$.

For text input, on each translation pair, all texts (ST transcripts and translation, and MT source and target texts) are pre-processed in the same way. Texts are case-sensitive. Punctuation is kept, split from words, and normalized. Non-print punctuation is removed. The sentences are then tokenized with Moses tokenizer [5]. We filter out samples whose number of source or target tokens is over 250 and whose ratio of source and target text lengths is outside range $[2/3, 3/2]$. For sub-wording, we use a unigram sentencepiece[6] model with a dictionary size of 10000. On each translation direction, The sentencepiece model is learned on all text data from both ST and MT corpora. The dictionary is shared across MT and ST and across source and target languages.

The performance is evaluated with BLEU (Papineni et al., 2002) using sacreBLEU [7]. We average

---

[5] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[6] https://github.com/google/sentencepiece

[7] https://github.com/mjpost/sacrebleu, with configuration

S.S. Projection	Decoder	EN-DE	EN-FR
-	-	25.6	35.0
Fixed	-	24.3	34.3
-	Fixed	24.2	33.4
Fixed	Fixed	23.8	33.1

Table 4: Performance of Mem-16 Chimera when freezing different modules in fine-tuning. S.S. Projection is abbreviation for shared semantic projection. "Fixed" indicates that weights in this module are not updated during fine-tuning, and "-" means otherwise. The results demonstrate that freezing modules indeed hampers the model's ability to adapt, but the weights pretrained on MT are already highly informative for ST.

MT	Contrastive	EN-DE	EN-FR
✓	✓	25.6	35.0
✓	×	25.0	34.6
×	✓	24.7	34.6
×	×	25.1	34.6

Table 5: BLEU scores of Mem-16 Chimera on MuST-C tst-COMMON set without one or both of auxiliary tasks. "×" means this task is not used during fine-tuning, and "✓" means othersize. "Contrastive" is the bi-modal contrastive task. The removal of one or both of tasks greatly harms the model's performance on both language pairs.

7 consecutive checkpoints around the one of the best dev loss and adopt a beam size of 10.

4.2 Model Configuration

For text input, we use 512-dimensional word embeddings plus sinusoidal positional embeddings. For audio input, the Wav2Vec2 Module follows the base configuration in Baevski et al. (2020). It uses parameters pretrained on LibriSpeech audio data only. The 1-dimensional CNN for speech features has 2 layers with stride size 2, kernel size 5, padding 2, and hidden dimension 1024.

The shared Transformer encoder consists of 6 layers. The memory queries are 64 512-dimensional vectors. The parameters of shared semantic projection resemble a 3-layer Transformer encoder. The Transformer decoder has 6 layers. Each of these Transformer layers, except for those in the Wav2Vec2 module, has an embedding dimension of 512, a hidden dimension of 512, and 8 attention heads.

In both pretraining and fine-tuning stages, we use an Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.98$, and 4k warm-up updates. We apply an inverse square root schedule algorithm for the learning rate. In MT pretraining, the learning rate is 5e-4, the maximum number of updates is 300k, with at most 33k input tokens per batch. In ST pretraining, the learning rate is 1e-4, the maximum number of updates is 150k, with at most 16M source audio frames per batch. The loss weights λ_{ST}, λ_{MT} and λ_{ctr} are all set to 1.

We also show results on a base version of Chimera, for which the memory queries are only 16 512-dimensional vectors (codenamed "Mem-16"). Because of the training efficiency and simplicity, all ablation studies and visualizations adopted the Mem-16 configuration if not stated otherwise.

Both Chimera and Chimera Mem-16 contain around 165M parameters. The whole training process for one trial on 8 Nvidia Tesla-V100 GPUs generally takes 20 –40 hours according to the translation direction.

4.3 Benchmark Experiments

Training We train Chimera in a pretrain - fine-tune manner. In the first stage, we pretrain Chimera on MT datasets so as to leverage additional sources of training data, as well as provide a better initialization point. In the fine-tuning stage, we adopt multi-task training as described in Section3.3. In addition to the conventional ST task, Chimera is also fine-tuned on MT and bi-modal contrastive task to align inputs from speech and text.

Baselines We include as baselines the speech transformer model from (Ott et al., 2019), Espnet result from (Inaguma et al., 2020), adaptive feature selection method from (Zhang et al., 2020a), dual-decoder Transformer from (Le et al., 2020) and Modality-Agnostic Meta-Learning from (Indurthi et al., 2019) in Table 1. We also provide a series of baseline results of a simple combination of Wav2Vec2 (Baevski et al., 2020) and Transformer. It could be viewed Chimera without external MT pre-training, with still competitive but not SOTA results.

To verify the effectiveness of our training technique, we also compare with other baselines able to leverage external MT corpus in Table 3.

Results The experiment results are shown in Table 1 and 2. Our Chimera achieves state-of-the-art performance on all language pairs, even though we do

---

of 13a tokenzier, case-sensitiveness and full punctuation

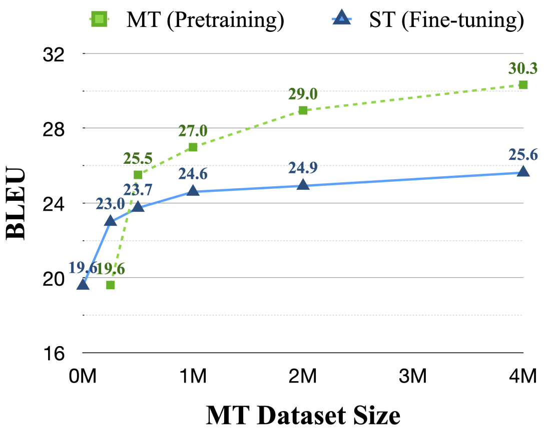

(a) Chimera Mem-16 on WMT14

Corpora	Size	MT	ST
WMT14	4M	32.1	26.3
WMT14 + OpenSubtitles	26M	32.9	27.1

(b) Chimera w/ and w/o OpenSubtitles

Figure 3: Curve of MuST-C EN-DE tst-COMMON BLEU scores on Chimera against the amount of MT data used during pretraining. (a) shows Chimera Mem-16's performance on WMT14. Blue triangles are the speech translation BLEU scores, and green squares are transcript-translation BLEU scores after MT pretraining. (b) shows how Chimera ($M = 64$) behaves with or without OpenSubtitles data.

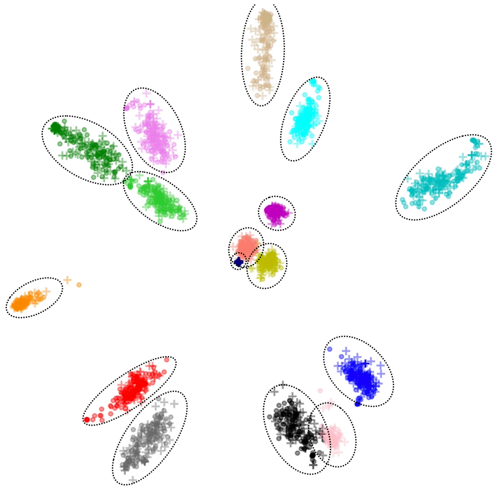

Figure 4: 2-dimensional PCA projection of the semantic memories in Mem16 Chimera across different samples. Each colored cluster (circled out) represents a semantic memory element,. A "·" corresponds to a speech semantic memory, and a "+" marks a text one.

not utilize Google Translate results on Augmented Librispeech as most baselines. EN-DE results of Chimera uses WMT14+OpenSubtitles for MT pretrainng, while a detailed ablation study on the effect of MT data can be found in Section 4.4. Note that the improvement on EN-PT is not so significant as EN-DE and EN-FR. We attribute this to the data discrepancy between OPUS100 and MuST-C. A large number of sentences in OPUS100 are from movie subtitles, which are more informal, contain repeated sentences, and cover different topics from those in MuST-C public speeches.

In Table 3, under the same data condition, Chimera outperforms other techniques such as decoder pre-training and knowledge distillation (Liu et al., 2019).

4.4 Ablation Studies and Visualizations

Knowledge Shared across Tasks One potential benefit in our design is that the shared semantic space can hold common knowledge shared across ST and MT tasks. To validate this motivating idea, we analyze the model's behavior while manipulating its modules. If certain weights pretrained during the MT task also contain meaningful information for ST, fixing them should not greatly harm the model's performance.

Specifically, after MT pretraining, we fix certain modules and do not update their weights during fine-tuning. We choose to fix the weights in the shared semantic projection module, the decoding module, or both of them.

Table 4 shows the results. After freezing modules, the results on both EN-DE and EN-FR drop slightly. This demonstrates that freezing weights indeed hampers the model's ability to adapt from MT to ST dataset. But the decreased scores are still comparable to many of the best results in Table 1. This validates the effectiveness of shared semantic space, and indicates that the weights pretrained on MT are already informative enough for Chimera to still generalize sufficiently well on ST task.

Multi-task Training One advantage of bridging the modality gap is that the model can fully benefit from training on auxiliary tasks with more data, such as those mentioned in Section 3.2. To evaluate their impacts, we conduct another ablation study on EN-DE and EN-FR.. Either or both of the auxiliary tasks are not used during fine-tuning.

The results of this ablation are presented in Table 5. Here we can see a significant decrease (with, for example, p=0.020 in one-tailed Student's t-test comparing row 1 and 2) in BLEU scores when either of the auxiliary tasks is abandoned. Although

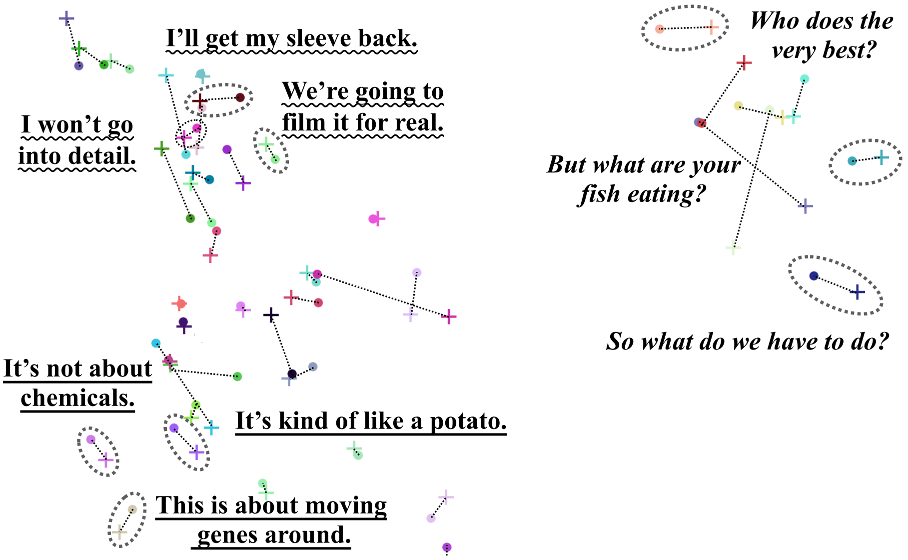

Figure 5: A visualization of one particular semantic memory in Mem-16 Chimera no different samples and modalities. "·" marks speech representations, and "+" marks text representations. Marks of the same color come from the same speech-transcript pair and are linked with dashed lines. Some of speech-transcript pairs are circled together and annotated with their transcripts. Three fonts distinguish three groups of transcripts of similar patterns.

the bi-model contrastive loss is not directly related to the ST task, a poor alignment between semantic memories of speech and text hinders Chimera from leveraging the shared knowledge across two tasks. When the MT task is removed, the drop in BLEU scores is also huge. This could be explained that during fine-tuning, the auxiliary MT task is necessary for keeping shared knowledge from being forgotten.

It is interesting to observe that abandoning both tasks produces results similar to abandoning either one of the tasks. This suggests that the two auxiliary tasks only have effects when combined with each other: only when both using MT task to maintain the pretrained parameters from forgetting, as well as using the bi-modal contrastive task to align between speech and text representations, can Chimera benefit from shared knowledge in MT pretraining.

Additional Machine Translation Data We attribute the gain in the performance of Chimera mainly to the pretraining on MT data. One evidence is the performance gain when using the larger OpenSubtitles as MT corpus for EN-DE in 1. To further probe the influence of the additional MT data, we vary the amount of MT data available during pretraining on EN-DE direction. The results are plotted in Figure 3.

As the size of additional MT dataset increases, the MuST-C BLEU score improves significantly. This confirms the importance of massive high-quality MT data for pretraining Chimera. The results also help explain the relatively inferior scores on EN-PT in Table 1 which uses the OPUS100 dataset in pretraining.

Visualization of Semantic Memories The shared semantic projection is designed to only extract semantic categories of information necessary for decoding, regardless of the input modality. In this way, it can bridge the different representations of speech and text during computation and facilitate knowledge sharing between MT and ST.

To validate this motivation, we visualize them with Principal Component Analysis (PCA) in Figure 4. Up to 100 speech-transcript paired samples are randomly chosen from the valid set. We record vector values of 16 semantic memories from Chimera Mem-16 when inputs are speeches or transcripts, and apply 2-dimensional PCA. The 16 semantic memories are distinguished by 16 colors. Every "·" corresponds to a semantic memory from speech, and each "+" is a semantic memory from the text. It is clear that the semantic memories are highly clustered, everyone of which learns a particular region. Speech and text representations are also projected close within the same region, proving the model's ability to ignore representation differences and bridge the modality gap.

To take a closer view of the structure of each semantic memory subspace, we randomly choose one

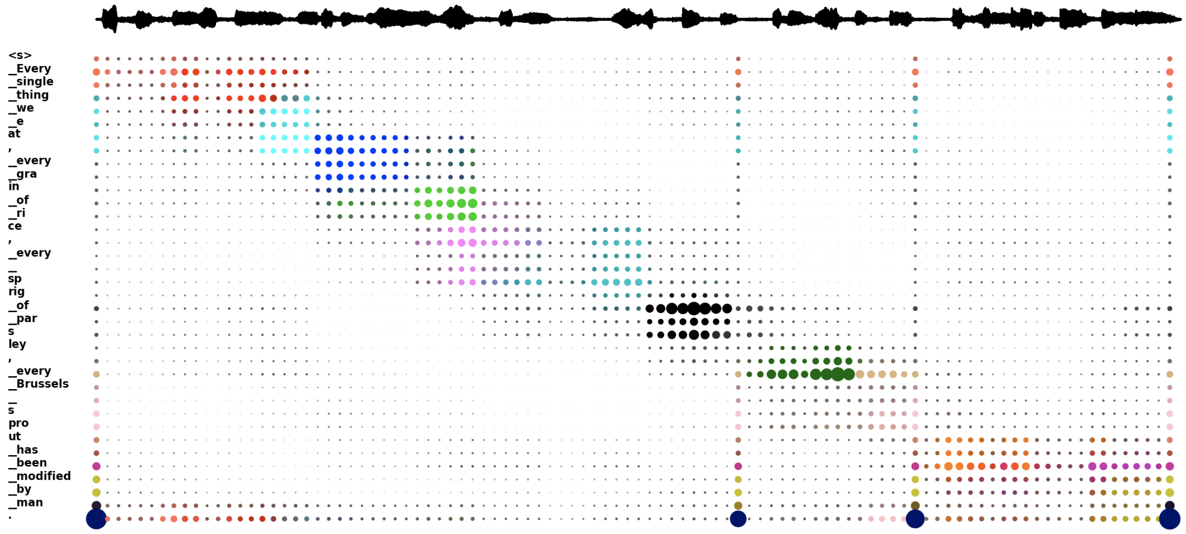

Figure 6: Visualization of the final-layer attention of $M = 16$ memories on inputs, and their alignment between each other. A pair of audio and its transcript is fed to Chimera Mem-16. The area of each dot is linear to dot product between two attention vectors. The color is a linear interpolation of M indicative colors as in Fig 4, with mixing weights linear to the Hadamard product of attention vectors. (Best viewed in color)

semantic memory and apply PCA to its corresponding cluster. The results are visualized in Figure 5. These samples come from 50 speech-transcript pairs. Each pair of speech ("·") and transcript ("+") share the same color and are linked through dashed lines.

Two interesting properties could be observed. First, paired speech and transcript inputs are again close to each other, even though they are coming from different modalities. Second, the visualized representations are organized according to their semantic or syntactic patterns. We recognize several clusters in the figure, and annotate their transcripts with different fonts. The three annotations at the upper-right corner (Italic font) are all questions; those at the upper-left corner (wavy underlined font) all follow a simple future tense; at the bottom-left corner of the figure (underlined font) is another cluster of sentences of copular verbs. This proves that the shared semantic space that Chimera has learned is well-structured, and thus validates our model design.

Visualization of Inter-Modal Attention Alignment "Attention" is the internal mechanism of Transformer based modules. In the design of Chimera, attention is used for extracting M key semantic categories of features from input. To investigate whether these extracted features is indeed semantic, we further visualized the similarity between attention on paired audio and text in Fig 6. Here the colors, which distinguish different memories attending to inputs, is clustered on sequence and distributed close to the diagonal, demonstrating an alignment between matching tokens in two modalities. Here we also observe four beaming columns, where the full stop mark in text aligns with pauses in audio. This is an indication of semantic rather than positional essence of the memories.

5 Conclusions and Future Work

In this paper, we propose Chimera, a model capable of learning a text-speech shared semantic memory network for bridging the gap between speech and text representations. Being able to leverage a large amount of external Machine Translation data, Chimera achieves new state-of-the-art performance on the MuST-C dataset on all 8 languages. Additional experiment results also demonstrate its ability to learn a well-structured shared semantic space as well as effectively share learned knowledge across MT and ST, and validate our design of auxiliary tasks.

In the future, we will focus on deriving a better task to tightly align speech and text representations. Also, the workflows of MT and ST are only partially shared in Chimera, which still requires the model to adapt to ST when switching to the fine-tuning stage. So it remains a challenge to better couple their computation graphs in future designs.

References

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations.

Parnia Bahar, Tobias Bieschke, and Hermann Ney. 2019a. A comparative study on end-to-end speech to text translation. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 792–799. IEEE.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019b. On using specaugment for end-to-end speech translation. *arXiv preprint arXiv:1911.08876*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. *arXiv preprint arXiv:1809.01431*.

Daniel Beck, Trevor Cohn, and Gholamreza Haffari. 2019. Neural speech translation using lattice transformations and graph networks. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 26–31.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *ICASSP*, pages 6224–6228. IEEE.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *arXiv preprint arXiv:1612.01744*.

Jeffrey R Binder, Kristen A McKiernan, Melanie E Parsons, Chris F Westbury, Edward T Possing, Jacqueline N Kaufman, and Lori Buchanan. 2003. Neural correlates of lexical access during visual word recognition. *Journal of cognitive neuroscience*, 15(3):372–393.

S Catrin Blank, Sophie K Scott, Kevin Murphy, Elizabeth Warburton, and Richard JS Wise. 2002. Speech production: Wernicke, broca and beyond. *Brain*, 125(8):1829–1838.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on SMT*, pages 12–58.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198.

Qiao Cheng, Meiyuan Fang, Yaqian Han, Jin Huang, and Yitao Duan. 2019. Breaking the data barrier: Towards robust speech translation via adversarial stability training. *arXiv preprint arXiv:1909.11430*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. *arXiv preprint arXiv:1805.06130*.

Mattia A Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019a. Must-c: a multilingual speech translation corpus. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2012–2017. Association for Computational Linguistics.

Mattia A Di Gangi, Matteo Negri, and Marco Turchi. 2019b. Adapting transformer to end-to-end spoken language translation. In *INTERSPEECH 2019*, pages 1133–1137. International Speech Communication Association (ISCA).

Mattia A Di Gangi, Viet-Nhat Nguyen, Matteo Negri, and Marco Turchi. 2020. Instance-based model adaptation for direct speech translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7914–7918. IEEE.

Mattia Antonino Di Gangi, Matteo Negri, Roldano Cattoni, Dessi Roberto, and Marco Turchi. 2019c. Enhancing transformer for end-to-end speech-to-text translation. In *Machine Translation Summit XVII*, pages 21–31. European Association for Machine Translation.

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2021a. Consecutive decoding for speech-to-text translation. In *The Thirty-fifth AAAI Conference on Artificial Intelligence, AAAI*.

Qianqian Dong, Rong Ye, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2021b. "listen, understand and translate": Triple supervision decouples end-to-end speech-to-text translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12749–12759.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *NAACL*, pages 949–959.

Erin Fitzgerald, Keith B Hall, and Frederick Jelinek. 2009. Reconstructing false start errors in spontaneous speech text.

Christian Fügen. 2008. *A system for simultaneous translation of lectures and speeches*. Ph.D. thesis, Verlag nicht ermittelbar.

Hirofumi Inaguma, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. 2019. Multilingual end-to-end speech translation. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 570–577. IEEE.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. 2020. Espnet-st: All-in-one speech translation toolkit. *arXiv preprint arXiv:2004.10234*.

Sathish Indurthi, Houjeung Han, Nikhil Kumar Lakumarapu, Beomseok Lee, Insoo Chung, Sangha Kim, and Chanwoo Kim. 2019. Data efficient direct speech-to-text translation with modality agnostic meta-learning. *arXiv preprint arXiv:1911.04283*.

Niehues Jan, Roldano Cattoni, Stüker Sebastian, Mauro Cettolo, Marco Turchi, and Marcello Federico. 2018. The iwslt 2018 evaluation campaign. In *International Workshop on Spoken Language Translation*, pages 2–6.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *ICASSP*, pages 7180–7184. IEEE.

Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. 2018. Structured-based curriculum learning for end-to-end english-japanese speech translation. *arXiv preprint arXiv:1802.06003*.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting librispeech with french translations: A multimodal corpus for direct speech translation evaluation. *arXiv preprint arXiv:1802.03142*.

Hang Le, Juan Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2020. Dual-decoder transformer for joint automatic speech recognition and multilingual speech translation. *arXiv preprint arXiv:2011.00747*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Dan Liu, Junhua Liu, Wu Guo, Shifu Xiong, Zhiqiang Ma, Rui Song, Chongliang Wu, and Quan Liu. 2018. The ustc-nel speech translation system at iwslt 2018. *arXiv preprint arXiv:1812.02455*.

Yuchen Liu, Hao Xiong, Zhongjun He, Jiajun Zhang, Hua Wu, Haifeng Wang, and Chengqing Zong. 2019. End-to-end speech translation with knowledge distillation. *arXiv preprint arXiv:1904.08075*.

Yuchen Liu, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2020. Bridging the modality gap for speech-to-text translation. *arXiv preprint arXiv:2010.14920*.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. *arXiv preprint arXiv:1804.08198*.

Evgeny Matusov, Arne Mauser, and Hermann Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *International Workshop on Spoken Language Translation (IWSLT) 2006*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318.

Juan Pino, Liezl Puzon, Jiatao Gu, Xutai Ma, Arya D McCarthy, and Deepak Gopinath. 2019. Harnessing indirect training data for end-to-end automatic speech translation: Tricks of the trade. *arXiv preprint arXiv:1909.06515*.

Juan Pino, Qiantong Xu, Xutai Ma, Mohammad Javad Dousti, and Yun Tang. 2020. Self-training for end-to-end speech translation. *arXiv preprint arXiv:2006.02490*.

Elizabeth Salesky, Susanne Burger, Jan Niehues, and Alex Waibel. 2018. Towards fluent translations from disfluent speech. In *2018 (SLT)*, pages 921–926. IEEE.

Elizabeth Salesky, Matthias Sperber, and Alex Waibel. 2019. Fluent translations from disfluent speech in end-to-end speech translation. *arXiv preprint arXiv:1906.00556*.

Donald Shankweiler, W Einar Mencl, David Braze, Whitney Tabor, Kenneth R Pugh, and Robert K Fulbright. 2008. Reading differences and brain: Cortical integration of speech and print in sentence processing varies with reader skill. *Developmental neuropsychology*, 33(6):745–775.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. *arXiv preprint arXiv:1704.00559*.

Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel. 2019. Self-attentional models for lattice inputs. *arXiv preprint arXiv:1906.01617*.

Galina Spitsyna, Jane E Warren, Sophie K Scott, Federico E Turkheimer, and Richard JS Wise. 2006. Converging language streams in the human temporal lobe. *Journal of Neuroscience*, 26(28):7328–7336.

Mihaela C Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing asr pretraining for low-resource speech-to-text translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7909–7913. IEEE.

Yun Tang, Juan Pino, Changhan Wang, Xutai Ma, and Dmitriy Genzel. 2021. A general multi-task learning framework to leverage text data for speech to text tasks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6209–6213. IEEE.

Nienke van Atteveldt, Elia Formisano, Rainer Goebel, and Leo Blomert. 2004. Integration of letters and speech sounds in the human brain. *Neuron*, 43(2):271–282.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Juan Raul Vazquez Carrillo, Alessandro Raganato, Jörg Tiedemann, Mathias Creutz, et al. 2019. Multilingual nmt with a language-independent attention bridge. In *The 4th Workshop on Representation Learning for NLP (RepL4NLP-2019) Proceedings of the Workshop*. The Association for Computational Linguistics.

Laura Cross Vila, Carlos Escolano, José AR Fonollosa, and Marta R Costa-jussà. 2018. End-to-end speech translation with the transformer. In *IberSPEECH*, pages 60–63.

Hari Krishna Vydana, Martin Karafiát, Katerina Zmolikova, Lukáš Burget, and Honza Černockỳ. 2021. Jointly trained transformers models for spoken language translation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7513–7517. IEEE.

Chengyi Wang, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou. 2020a. Bridging the gap between pretraining and fine-tuning for end-to-end speech translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9161–9168.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020b. Curriculum pre-training for end-to-end speech translation. *arXiv preprint arXiv:2004.10093*.

Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. *arXiv preprint arXiv:1703.08581*.

Yinfei Yang, Gustavo Hernández Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Improving multilingual sentence embedding using bidirectional dual encoder with additive margin softmax. *arXiv preprint arXiv:1902.08564*.

Katherine Yu, Haoran Li, and Barlas Oguz. 2018. Multilingual seq2seq training with similarity loss for cross-lingual document classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 175–179.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. 2020a. Adaptive feature selection for end-to-end speech translation. *arXiv preprint arXiv:2010.08518*.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020b. Improving massively multilingual neural machine translation and zero-shot translation. *arXiv preprint arXiv:2004.11867*.

Pei Zhang, Boxing Chen, Niyu Ge, and Kai Fan. 2019. Lattice transformer for speech translation. *arXiv preprint arXiv:1906.05551*.

Changfeng Zhu, Heng Yu, Shanbo Cheng, and Weihua Luo. 2020. Language-aware interlingua for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1650–1655.

Empowering Language Understanding with Counterfactual Reasoning

Fuli Feng[1,2]**, Jizhi Zhang**[3]**, Xiangnan He**[3]*****, Hanwang Zhang**[4]**, Tat-Seng Chua**[2]
[1]NExT-Sea Joint Lab, [2]National University of Singapore
[3]University of Science and Technology of China, [4]Nanyang Technological University
`fulifeng93@gmail.com, cdzhangjizhi@mail.ustc.edu.cn`
`xiangnanhe@gmail.com,hanwangzhang@ntu.edu.sg,dcscts@nus.edu.sg`

Abstract

Present language understanding methods have demonstrated extraordinary ability of recognizing patterns in texts via machine learning. However, existing methods indiscriminately use the recognized patterns in the testing phase that is inherently different from us humans who have *counterfactual thinking*, *e.g.,* to scrutinize for the hard testing samples. Inspired by this, we propose a *Counterfactual Reasoning Model*, which mimics the counterfactual thinking by learning from few counterfactual samples. In particular, we devise a *generation module* to generate representative counterfactual samples for each factual sample, and a *retrospective module* to retrospect the model prediction by comparing the counterfactual and factual samples. Extensive experiments on sentiment analysis (SA) and natural language inference (NLI) validate the effectiveness of our method.

1 Introduction

Language understanding (Ke et al., 2020) is a central theme of artificial intelligence (Chomsky, 2002), which empowers a wide spectral of applications such as sentiment evaluation (Feldman, 2013), commonsense inference (Bowman et al., 2015). The models are trained on labeled data to recognize the textual patterns closely correlated to different labels. Owing to the extraordinary representational capacity of deep neural networks, the models can well recognize the pattern and make prediction accordingly (Devlin et al., 2019). However, the cognitive ability of these data-driven models is still far from human beings due to lacking *counterfactual thinking* (Pearl, 2019).

Counterfactual thinking is a high-level cognitive ability beyond pattern recognition (Pearl, 2019). In addition to observing the patterns within factual samples, counterfactual thinking calls for comparing the fact with imaginations, so as to make better decision. For instance, given a factual sample "*What do lawyers do when they die? Lie still.*", the intuitive evaluation of its sentiment based on the textual patterns will recognize "*Lie still*" as an objective description of body posture which is *neutral*. By scrutinizing that the "still" could be intentionally postposed, we can imagine a counterfactual sample "*What do lawyers do when they die? Still lie.*" and uncover the negative sarcastic pun, whose sentiment is more accurate.

Recent work (Kaushik et al., 2019; Zeng et al., 2020) shows that incorporating counterfactual samples into model training improves the generalization ability. However, these methods follow the standard machine learning paradigm that uses the same procedure (*e.g.,* a forward propagation) to make prediction in the testing phase. That is, making decision for testing samples according to their relative positions to the model decision boundary. The indiscriminate procedure focuses on the textual patterns occurred in the testing sample and treats all testing samples equally, which easily fails on hard samples (*cf.* Figure 1). On the contrary, humans can discriminate hard samples and ponder the decision with a rational system (Daniel, 2017), which imagines counterfactual and adjusts the decision.

The key to bridge this gap lies in imitating the counterfactual thinking ability of humans, *i.e.,* learning a decision making procedure to serve for the testing phase. That is a procedure of: 1) constructing counterfactual samples for a target factual sample; 2) calling the trained language understanding model to make prediction for the counterfactual samples; and 3) comparing the counterfactual and factual samples to retrospect the model prediction. However, the procedure is non-trivial to achieve for two reasons: 1) the space of counterfactual sample is huge since any variant from the target factual

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2226–2236
August 1–6, 2021.

sample can be a counterfactual sample. It is thus challenging to search for suitable counterfactual samples that can facilitate the decision making. 2) The mechanism of how we retrospect the decision is still unclear, making it hard to be imitated.

Towards the target, we propose a *Counterfactual Reasoning Model* (CRM), which is a two-phase procedure consisting a *generation module* and a *retrospection module*. In particular, given a factual sample in the testing phase, the generation module constructs representative counterfactual samples by imagining *what would the content be if the label of the sample is* y. To imitate the unknown retrospection mechanism of humans, we build the retrospection module as a carefully designed deep neural network that separately compares the latent representation and the prediction of the factual and counterfactual samples. The proposed CRM forms a general paradigm that can be applied to most existing language understanding models without constraint on the format of the language understanding task. We select two language understanding tasks: SA and NLI, and test CRM on three representative models for each task. Extensive experiments on benchmark datasets validate the effectiveness of CRM, which achieves performance gains ranging from 5.1% to 15.6%.

The main contributions are as follow:

- We propose the *Counterfactual Reasoning Model* to enlighten the language understanding model with counterfactual thinking.

- We devise a generation module and a retrospection module that are task and model agnostic.

- We conduct extensive experiments, which validate the rationality and effectiveness of the proposed method.

2 Pilot Study

Decisions are usually accompanied by confidence, a feeling of being wrong or right (Boldt et al., 2019). From the perspective of model confidence, we investigate the performance of language understanding models across different testing samples. We estimate the model confidence on a sample as the widely used *Maximum Class Probability* (MCP) (Corbière et al., 2019), which is the probability over the predicted class. A lower value of MCP means less confidence and "hard" sample. According to the value of MCP, we rank the testing samples in ascending order and split them into ten groups, *i.e.,* confidence level from 1 to 10.

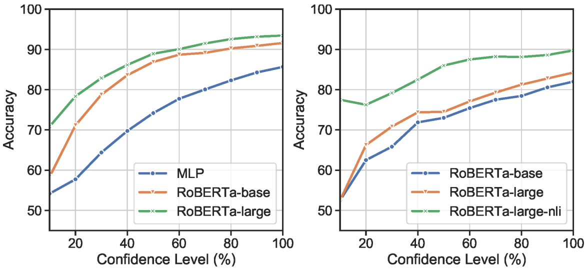

(a) Sentiment analysis (b) Natural language inference

Figure 1: Prediction performance of the language understanding models over testing samples at different confidence levels.

Figure 1 shows the performance of representative models over samples at different model confidence levels on the SA and NLI tasks (see Section 4.1 for model and dataset descriptions). From the figures, we can observe a clear increasing trend of classification accuracy as the confidence level increases from 1 to 10 in all cases. In other words, these models fail to predict accurately for the hard samples. It is thus essential to enhance the standard inference with a more precise decision making procedure.

3 Methodology

In this section, we first formulate the task of learning a decision making procedure for the testing phase (Section 3.1), followed by introducing the proposed CRM (Section 3.2) and the paradigm of building language understanding solutions with CRM (Section 3.3).

3.1 Problem Formulation

As discussed in the previous work (Wu et al., 2020; Li et al., 2020, 2019), language understanding tasks can be abstracted as a classification problem where the input is a text and the target is to make decision across a set of candidates of interests. We follow the problem setting with consideration of counterfactual samples (Kaushik et al., 2019; Liang et al., 2020), where the training data are twofold: 1) *factual samples* $\mathcal{T} = \{(\boldsymbol{x}, y)\}$ where $y \in [1, C]$ denotes the class or the target decision of the text; $\boldsymbol{x} \in \mathbb{R}^D$ is the latent representation of the text, which encodes the textual contents[1]. 2) *counterfactual samples*

---

[1] The input is indeed the plain text which is projected to a latent representation by an encoder (*e.g.,* a Transformer (Devlin et al., 2019)) in the cutting edge solutions. We omit the

$$\mathcal{T}^* \quad = \quad \{(\boldsymbol{x}_c^*, c) | (\boldsymbol{x}, y) \in \mathcal{T}, c \in [1, C] \& c \neq y\}$$

where $(\boldsymbol{x}_c^*, c)$ is a counterfactual sample in class c corresponds to the factual sample $(\boldsymbol{x}, y)$[2]. We assume that a classification model (*e.g.,* BERT (Devlin et al., 2019)) has been trained over the labeled data. Formally,

$$\hat{\boldsymbol{\theta}} = \min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y) \in \mathcal{T}/\mathcal{T}^*} l(y, f(\boldsymbol{x}|\boldsymbol{\theta})) + \alpha \|\boldsymbol{\theta}\|, \qquad (1)$$

where $\hat{\boldsymbol{\theta}}$ is the learned parameters of the model $f(\cdot)$; $l(\cdot)$ is a classification loss such as cross-entropy (Kullback, 1997), and α is a hyper-parameter to adjust the regularization.

The target is to build a decision making procedure to perform counterfactual reasoning when serving for the testing phase. Given a testing sample $\boldsymbol{x}$, the core is a policy of generating counterfactual samples and retrospecting the decision, which is formulated as:

$$\boldsymbol{y} = h\left(\boldsymbol{x}, \{\boldsymbol{x}^*\} | \boldsymbol{\eta}, \hat{\boldsymbol{\theta}}\right), \ \{\boldsymbol{x}^*\} = g\left(\boldsymbol{x} | \boldsymbol{\omega}\right),$$

$\boldsymbol{y} \in \mathbb{R}^C$ denotes the final prediction for the testing sample $\boldsymbol{x}$, which is a distribution over the classes; $\boldsymbol{x}^*$ is one of the generated counterfactual samples for $\boldsymbol{x}$. The generation module $g(\cdot)$ parameterized by $\boldsymbol{\omega}$ is expected to construct a set of representative counterfactual samples for the target factual sample, which provide signals for the retrospection module $h(\cdot)$ parameterized by $\boldsymbol{\eta}$ to retrospect the prediction $f\left(\boldsymbol{x}|\hat{\boldsymbol{\theta}}\right)$ given by the trained classification model. In particular, $h(\cdot)$ and $g(\cdot)$ will be learned from the factual and counterfactual training samples, respectively.

3.2 Counterfactual Reasoning Model

Figure 2 illustrates the process of CRM where the arrows in grey color represent the standard inference of trained classification model, and arrows in red color represent the retrospection with consideration of counterfactual samples.

3.2.1 Retrospection Module

We devise the retrospection module with one key consideration—distilling signals for making final decision by comparing both the latent representation and the prediction of the counterfactual samples with the factual sample. To achieve the target, we devise three key building blocks for retrospection, which successively perform *representation comparison*, *prediction comparison*, and *fusion* . In particular, the module first compares the representation of each counterfactual sample with the factual sample; then compares their predictions accordingly; and fuses the comparison across the counterfactual samples.

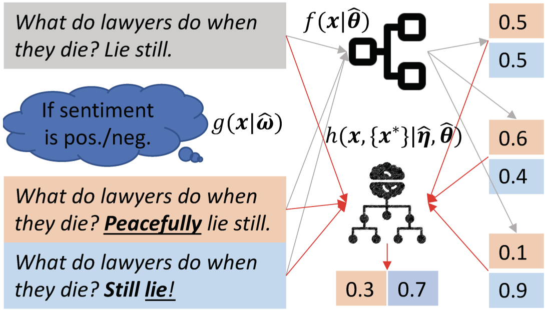

Figure 2: Illustration of the proposed CRM.

Representation comparison. Given a pair of counterfactual sample $\boldsymbol{x}^*$ and factual sample $\boldsymbol{x}$, we believe the signals meaningful for making final decision lie in the difference of the samples and how the difference affects the classification. To distill such signals, we devise the representation comparison block as $\boldsymbol{y}_\triangle = f(\boldsymbol{x} - \boldsymbol{x}^*|\hat{\boldsymbol{\theta}})$, where $\boldsymbol{y}_\triangle \in \mathcal{R}^C$ denotes the prediction of the representation difference $\boldsymbol{x} - \boldsymbol{x}^*$ given by the trained classification model. Note that we leverage the trained model to enlighten how the content difference affects the classification since the model is trained to capture the connection between the textual patterns and the classes. It should be noted that we use a duplicate of the trained classification model for the representation comparison. That is to say, the training of the retrospection module will not affect the classification model.

Prediction comparison. To retrospect the prediction $f(\boldsymbol{x}|\hat{\boldsymbol{\theta}})$, we devise a prediction comparison block to compare the predictions of each counterfactual and factual sample pair and distill patterns from $f(\boldsymbol{x}|\hat{\boldsymbol{\theta}})$, $f(\boldsymbol{x}^*|\hat{\boldsymbol{\theta}})$, and $\boldsymbol{y}_\triangle$. Inspired by the success of convolutional neural network (CNN) in capture local-region patterns, the block is devised as a CNN, which is formulated as:

$$\boldsymbol{y}^* = \text{CNN}\left(f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}), f(\boldsymbol{x}^*|\hat{\boldsymbol{\theta}}), \boldsymbol{y}_\triangle\right), \qquad (2)$$

where $\boldsymbol{y}^*$ denotes the retrospected prediction when comparing to $\boldsymbol{x}^*$. In particular, a *stack layer* first stacks the three predictions as a matrix, which serves as an "image" to facilitate "observing" pat-

---

encoder for briefness since focusing on the decision making.

[2] Given the labeled factual sample, counterfactual samples can be constructed either manually (Kaushik et al., 2019) or automatically (Chen et al., 2020) by conducting minimum changes on $\boldsymbol{x}$ to swap its label from y to c

terns. Formally, $\boldsymbol{Y} = \left[f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}), f(\boldsymbol{x}^*|\hat{\boldsymbol{\theta}}), \boldsymbol{y}_\triangle \right]$ where $\boldsymbol{Y} \in \mathbb{R}^{C\times 3}$. $\boldsymbol{Y}$ is then fed into an 1D convolution layer to capture the intra-class patterns across the predictions, which is formulated as:

$$\boldsymbol{H} = \sigma(\boldsymbol{Y} * \boldsymbol{F}),\ \boldsymbol{H}_{ij} = \sigma(\boldsymbol{Y}_{:i}\boldsymbol{F}_j), \tag{3}$$

where $\boldsymbol{F} \in \mathbb{R}^{3\times K}$ denotes the filters in the convolution layer, and $\sigma(\cdot)$ is an activation function such as *GELU* (Hendrycks and Gimpel, 2016). $\boldsymbol{Y}_{:i}$ and $\boldsymbol{F}_j$ represent the i-th row of $\boldsymbol{Y}$ and the j-th column of $\boldsymbol{F}$, respectively. The filter $\boldsymbol{F}_j$ can learn rules for conducting retrospection. For instance, a filter $[1, -1, 0]$ means deducting the prediction of the counterfactual sample from that of the factual sample. The output $\boldsymbol{H} \in \mathbb{R}^{C\times K}$ is then flattened as a vector and fed into a fully-connected (FC) layer to capture the inter-class patterns. Formally,

$$\boldsymbol{y}^* = \boldsymbol{W} flatten(\boldsymbol{H}) + \boldsymbol{b}, \tag{4}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are model parameters.

Fusion. The target is to fuse the retrospected predictions $\{\boldsymbol{y}^*\}$ into a final decision $\boldsymbol{y}$. Inspired by the success of pooling function in reading out patterns, we devise the block as $\boldsymbol{y} = pooling(\{\boldsymbol{y}^*\})$. As the fusion is performed after the pairwise comparison, we term it as *late fusion*.

Training. We update the parameters of the retrospection module by minimizing the classification loss over the factual training samples, which is:

$$\hat{\boldsymbol{\eta}} = \min_{\boldsymbol{\eta}} \sum_{(\boldsymbol{x},y)\in\mathcal{T}} l(y, \boldsymbol{y}) + \lambda\|\boldsymbol{\eta}\|. \tag{5}$$

where λ denotes the hyper-parameter to adjust the weight of the regularization term.

It should be noted that no existing research has uncovered the specific mechanism of retrospection in our brain, *i.e.,* the order of comparison and fusion is unclear. As such, we further devise two fusion strategies: *middle fusion* and *early fusion*, which performs fusion within the CNN, *i.e.,* during comparison, and before the CNN, respectively.

- *Middle fusion* performs aggregation between the convolution layer and the FC layer. This fusion first calculates the latent comparison signals $\boldsymbol{H}$ for each pair of counterfactual and factual samples according to Equation 3. The aggregated signals $pooling(\{\boldsymbol{H}\})$ are then fed into the FC layer (Equation 4) to obtain the final decision $\boldsymbol{y}$.
- *Early fusion* aggregates the counterfactual samples before performing comparison, which is formulated as $\tilde{\boldsymbol{x}}^* = pooling(\{\boldsymbol{x}^*\})$. In this way, the retrospection module is formulated as: $\boldsymbol{y} = \text{CNN}\left(f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}), f(\tilde{\boldsymbol{x}}^*|\hat{\boldsymbol{\theta}}), f(\tilde{\boldsymbol{x}}^* - \boldsymbol{x}|\hat{\boldsymbol{\theta}})\right)$. For all the three fusion methods, we can use either regular pooling function without parameter or parameterized pooling function (Ying et al., 2018) to enhance the expressiveness of the retrospection module. In our experiments, using a simple mean pooling achieves a performance that is comparable to the parameterized one in most cases (*cf.* Table 3).

3.2.2 Generation Module

The target is to construct counterfactual samples that are informative for retrospecting the decision on the target factual sample $\boldsymbol{x}$. As the task involves making decision among C candidate classes, we believe that the key to generate representative counterfactual samples lies in imagining *"what would the content be if the sample belongs to class c", i.e.,* generating C counterfactual samples $\{\boldsymbol{x}_c^*\}$. With the C classes as the targets, the searching space of samples can also be largely narrowed down. Toward this end, we devise the generation module with two main considerations: 1) decomposing the factual sample $\boldsymbol{x}$ to distill contents irrelevant to the label of the sample $\boldsymbol{u} = d(\boldsymbol{x}|\boldsymbol{\omega})$; 2) injecting class c into $\boldsymbol{u}$ to form the counterfactual sample $\boldsymbol{x}_c^*$.

Decomposition. To distill $\boldsymbol{u}$, we need to recognize the connection between the content of the factual sample and each class. We thus account for class representations in the decomposition function. To align the sample space of the generation module with the retrospection module $h(\cdot)$ and the classification model $f(\cdot)$, we extract the parameters from the prediction layer of the trained classification model as the class representations. In particular, we extract the mapping matrix $\boldsymbol{W} \in \mathbb{R}^{C\times D}$ where the c-th row corresponds to class c. Note that we assume that the prediction layer has the same dimensionality as the latent representation, which is a common setting in most cutting edge language understanding models. The decomposition function is devised as a CNN to capture both the intra-dimension and inter-dimension connections between the factual sample and the classes.

- *Stack layer.* The stack layer stacks the factual sample, class representations, and the element-wise product between sample and each class, which is formulated as: $\boldsymbol{X} = [\boldsymbol{x}, \boldsymbol{W}^T, \boldsymbol{x} \odot \boldsymbol{W}^T]$. $\boldsymbol{x} \odot \boldsymbol{W}^T \in \mathbb{R}^{D\times C}$ shed lights on how closely each dimension of $\boldsymbol{x}$ connect to each class, where large absolute value indicates closer connections.

- *Convolution layer.* This layer uses 1D horizontal filters to learn patterns of deducting class relevant contents from the factual sample, which is formulated as $\boldsymbol{h} = pooling(\sigma(\boldsymbol{X} * \boldsymbol{F}^g))$. $\boldsymbol{F}^g \in \mathbb{R}^{(2C+1)\times L}$ denotes the filters where L is the total number of filters. The output $\boldsymbol{h} \in \mathbb{R}^D$ is a hidden representation.
- *FC layers.* We use two FC layers to capture the inter-dimension connections. Formally, $\boldsymbol{u} = \boldsymbol{W}^2\sigma(\boldsymbol{W}^1\boldsymbol{h} + \boldsymbol{b}^1) + \boldsymbol{b}^2$, where $\boldsymbol{W}^2 \in \mathcal{R}^{D\times M}$, $\boldsymbol{W}^1 \in \mathcal{R}^{M\times D}$, $\boldsymbol{b}^2 \in \mathcal{R}^D$, and $\boldsymbol{b}^1 \in \mathcal{R}^M$ are learnable parameters. M is a hyper-parameter to adjust the complexity of the decomposition function. Note that we can stack more layers to enhance the expressiveness of the function, whereas using two layers according to the universal approximation theorem (Hornik, 1991).

We learn the parameters of the decomposition function from the counterfactual training samples by optimizing the following objective:

$$\min_{\boldsymbol{\omega}} \sum_{(\boldsymbol{x}_c^*, c)\in\mathcal{T}^*} r(\boldsymbol{u}_c^*, \tilde{\boldsymbol{u}}_c) + \gamma l(c, f(\boldsymbol{x}_c^* - \boldsymbol{u}_c^*|\hat{\boldsymbol{\theta}})) + r(\boldsymbol{u}, \tilde{\boldsymbol{u}}_c) + \gamma l(y, f(\boldsymbol{x} - \boldsymbol{u}|\hat{\boldsymbol{\theta}})), \tag{6}$$

where $\boldsymbol{u}_c^* = d(\boldsymbol{x}_c^*|\boldsymbol{\omega})$ and $\boldsymbol{u} = d(\boldsymbol{x}|\boldsymbol{\omega})$ are the decomposition results of the counterfactual sample $\boldsymbol{x}_c^*$ and the corresponding factual sample $\boldsymbol{x}$; $\tilde{\boldsymbol{u}}_c = \frac{1}{2}(\boldsymbol{x} + \boldsymbol{x}_c^*)$ denotes the target value of the decomposition. The two terms $r(\cdot)$ and $l(\cdot)$ are Euclidean distance (Dattorro, 2010) and classification loss. By minimizing the two terms, we encourage the decomposition result: 1) to be close to the target value $\tilde{\boldsymbol{u}}_c$; and 2) if being deducted from the original sample (*e.g.,* , $\boldsymbol{x} - \boldsymbol{u}$), the classification cannot be influenced. γ is a hyper-parameter to balance the two terms.

The rationality of setting $\tilde{\boldsymbol{u}}_c = \frac{1}{2}(\boldsymbol{x} + \boldsymbol{x}_c^*)$ as the target class irrelevant content of $\boldsymbol{x}$ and $\boldsymbol{x}_c^*$ comes from the parallelogram law (Nash, 2003). Note that this pair of samples belong to two different classes where a decision boundary (a hyperplane) lies between the two classes y and c. Considering that the sample $\boldsymbol{x}$ corresponds to a vector in the hidden space, we can decompose the vector into two components that are orthogonal and parallel to the decision boundary, *i.e.,* $\boldsymbol{x}_c^* = \boldsymbol{o}_c^* + \boldsymbol{p}_c^*$ and $\boldsymbol{x} = \boldsymbol{o} + \boldsymbol{p}$. Since the two samples belong to different classes, their orthogonal components are in opposite directions and their addition will only retain the parallel components, which are irrelevant to judging the class between y and c[3].

Injection. Accordingly, given a testing sample $\boldsymbol{x}$, we can inject the orthogonal components towards class c via $\boldsymbol{x}_c^* = 2 * d(\boldsymbol{x}|\hat{\boldsymbol{\omega}}_c) - \boldsymbol{x}$, which is the imagined content of the sample if it belongs to class c. In this way, for each testing sample, we conduct the injection over all the classes and construct C counterfactual samples $\{\boldsymbol{x}_c^*\}$, which are then used in the retrospection module[4].

3.3 Learning Paradigm with CRM

The existing work (Kaushik et al., 2019; Zeng et al., 2020) for language understanding typically follows the standard learning paradigm, *i.e.,* training a classification model over labeled data. Applying the proposed CRM indeed forms a new learning paradigm for constructing language understanding solutions. Algorithm 1 illustrates the procedure of the new paradigm.

Algorithm 1 Learning paradigm with CRM

```
Input: Training data $\mathcal{T}$, $\mathcal{T}^*$.
    /* Training */
Optimize Equation 1;        ▷ Classification model training
Optimize Equation 6;        ▷ Generation module training
Optimize Equation 5;        ▷ Retrospection module training
Return $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\omega}}_c$, and $\hat{\boldsymbol{\eta}}$.
    /* Testing */
Calculate $f(\boldsymbol{x}|\hat{\boldsymbol{\theta}})$;        ▷ Classification model inference
for $c = 1 \rightarrow C$ do
    $\boldsymbol{x}_c^* = 2 * g(\boldsymbol{x}|\hat{\boldsymbol{\omega}}_c) - \boldsymbol{x}$;        ▷ Generation
end for
Calculate $h(\boldsymbol{x}, \{\boldsymbol{x}_c^*\}|\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\theta}})$;        ▷ Retrospection
```

4 Experiments

We conduct experiments on two representative language understanding tasks, SA and NLI, to answer the following research questions:

- **RQ1:** To what extent counterfacutal reasoning improves language understanding?
- **RQ2:** How does the design of the retrospection module affect the proposed CRM?
- **RQ3:** How effective are the counterfactual samples generated by the proposed generation module?

4.1 Experiment Settings

Datasets. We adopt the same datasets in (Kaushik et al., 2019) for both tasks. The SA data are reviews

---

[3] Note that we normalize all samples to be unit vectors in the decomposition function. Moreover, inspired by (Parascandolo et al., 2018), we train a decomposition function for each class, *i.e.,* class-specific parameters $\hat{\boldsymbol{\omega}}_c$

[4] The generation module consists of C decomposition functions $d(\boldsymbol{x}|\hat{\boldsymbol{\omega}}_c)$ and the non-parametric injection function.

from IMDb, which are labeled as either positive or negative. For each factual review, the dataset contains a manually constructed counterfactual sample where the crowd workers are asked to manipulate the text to reverse the label with the constraint of no gratuitous change. NLI is a three-way classification task with two sentences as inputs and the target of detecting their relation within entailment, contradiction, and neutral. For each factual sample, four counterfactual samples are given, which are constructed by editing either the first or the second sentence with target relations different to the label of the factual sample.

Classification models. Owing to the extraordinary representational capacity of language model, fine-tuning pre-trained language model has become the emergent technique for solving language understanding tasks (Devlin et al., 2019). We select the widely used RoBERTa-base[5] and RoBERTa-large[6] for the consideration of the robustness of the RoBERTa (Liu et al., 2019) and our limited computation resources. For SA, we also test the classical Multi-Layer Perceptron (MLP) (Teney et al., 2020) with *tf-idf* text features (Schütze et al., 2008) as inputs. For NLI, we further test RoBERTa-large-nli[7], which has been fine-tuned on the large-scale MultiNLI dataset (Williams et al., 2018).

Baselines. As the proposed CRM leverages counterfactual samples, we compare CRM with three representative methods using counterfactual samples in language understanding tasks: 1) **+CF** (Kaushik et al., 2019), which uses counterfactual samples as data augmentation for model training; 2) **+GS** (Teney et al., 2020), which compares the factual and counterfactual samples in model training through regularizing their gradients; and 3) **+CL** (Liang et al., 2020), which compares the factual and counterfactual samples through a contrastive loss. Moreover, we report the performance of the testing model under *Normal Training*, *i.e.,* training over factual samples only.

Implementation. We implement the proposed CRM with PyTorch 1.7.0 based on Hugging Face Transformer[8], which is released at: https://github.com/fulifeng/Counterfactual_Reasoning_Model. In all cases, we follow the setting of +CF for training the classification model, which is a standard fine-tuning in (Liu et al., 2019). We then use *adam* (Kingma and Ba, 2014) with learning rate of 0.001 to optimize the retrospection module and the generation module. For the retrospection module, we set the number of filters in the convolution layer K as 10, the weight for regularization λ as 0. As to the generation module, we set the number of convolution filters as 10, the size of the hidden layer M as 256, and the weight for balancing Euclidean distance and classification loss γ as 15. We report the average classification accuracy over 5 different runs. For each repeat, we train the model with 20 epochs and select the model with the best performance on the validation set.

4.2 Performance Comparison (RQ1)

We first use the handcrafted counterfactual samples to demonstrate the effectiveness of counterfactual reasoning in the inference stage of language understanding model, which can be seen as using a golden standard generation module to provide counterfactual samples for the retrospection module. Note that we do not use the label of counterfactual samples in the testing set. Table 1 shows the performance of the compared methods on the two tasks. From the table, we observe that:

- +CRM largely outperforms all the baseline methods in all cases. As compared to +CF, the same classification model without CRM in the testing phase, +CRM achieves relative performance improvement up to 15.6%. The performance gain is attributed to the retrospection module, which justifies the rationality and effectiveness of incorporating counterfactual thinking into the inference stage of language understanding model. In other words, by comparing the factual sample with its counterfactual samples, the retrospection module indeed makes more accurate decisions.
- On the SA task, a huge gap (85.3 ↔ 93.4) lies in the performance of the shallow model MLP and the deep RoBERTa-base/RoBERTa-large. When applying +CRM, MLP achieves a performance that is comparable to the deep models. The result indicates that counterfactual reasoning can compensate for the disadvantages caused by the insufficient model representational capacity. In addition, the result reflects that CRM brings cognitive ability beyond recognizing textual patterns. If the retrospection module only facilitates capturing the correlation between textual patterns and classes, such simple model cannot bridge the

---

[5] https://huggingface.co/roberta-base.

[6] https://huggingface.co/roberta-large.

[7] https://huggingface.co/roberta-large-mnli.

[8] https://github.com/huggingface/transformers.

Sentiment Classification						
Backbone	Normal Training	+CF	+GS	+CL	+CRM	*RI*
MLP	86.9±0.5	85.3±0.3	84.6±0.4	-	**98.6±0.2**	15.6%
RoBERTa-base	93.2±0.6	92.3±0.7	92.2±0.9	91.8±1.1	**97.5±0.3**	5.7%
RoBERTa-large	93.6±0.6	93.4±0.4	93.1±0.5	94.1±0.4	**98.2±0.3**	5.1%
Natural Language Inference						
Backbone	Normal Training	+CF	+GS	+CL	+CRM	*RI*
RoBERTa-base	83.5±0.8	83.4±0.9	83.8±1.7	84.1±1.1	**91.5±1.6**	9.7%
RoBERTa-large	87.9±1.7	85.8±1.2	86.2±1.2	86.5±1.6	**93.8±1.9**	9.3%
RoBERTa-large-nli	89.4±0.7	88.2±1.0	87.2±1.4	88.2±1.0	**94.4±1.2**	7.1%

Table 1: Performance of the proposed CRM (Early Fusion) and baselines on the SA and NLI tasks. RI means the relative performance improvement achieved by +CRM over the classification model without CRM, *i.e.,* +CF.

huge gap of representational capacity between MLP and RoBERTa-large.

- The performance of baseline methods are comparable to each other in most cases, *i.e.,* incorporating counterfactual samples into model training does not necessarily improve the testing performance on factual samples. This result is consistent with (Kaushik et al., 2019), which is reasonable since these methods are devised for enhancing the generalization ability, especially for the out-of-distribution testing samples, which can sacrifice the performance on normal testing samples. Besides, the result indicates that training with counterfactual samples is insufficient for achieving counterfactual thinking, which reflects the rationality of enhancing the inference paradigm with a decision making procedure.

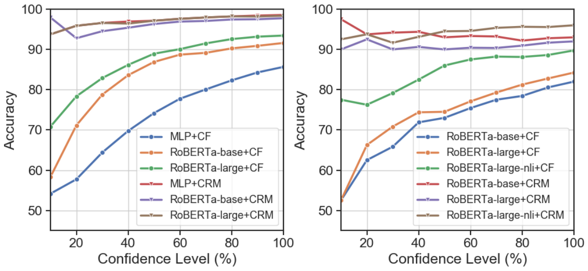

(a) Sentiment analysis (b) Natural language inference

Figure 3: Prediction performance of +CF and +CRM over testing samples at different confidence levels.

Performance on hard samples. Furthermore, we investigate whether the proposed CRM facilitate dealing with hard samples. Recall that we split the testing samples into 10 groups according to the confidence of the classification model, *i.e.,* +CF (*cf.* Section 2). We perform group-wise comparison between +CF and +CRM. Figure 3 shows the performance of all the classification models with +CF and +CRM. From the figures, 1) we observe that the performance of +CRM is stable across different confidence levels, whereas the performance of the classification model shows a clear decreasing trend as the confidence level decreases from 10 to 1. The result indicates that the retrospection module is insensitive to the confidence of the classification model. 2) In all cases, +CRM achieves the largest performance gain at the first group with confidence level of 1, *i.e.,* the hardest group to the classification model. For instance, the improvement reaches 85.7% on the RoBERTa-base model for the NLI task. The large improvements further justifies the effectiveness of the retrospection module, *i.e.,* comparing the prediction of factual samples to counterfactual samples indeed facilitates dealing with hard samples.

Sentiment Classification		
Backbone	Implicit	+CRM
MLP	79.3±0.2	**98.6±0.2**
RoBERTa-base	94.7±0.6	**97.5±0.3**
RoBERTa-large	98.0±0.4	98.2±0.3
Natural Language Inference		
Backbone	Implicit	+CRM
RoBERTa-base	81.9±3.5	**91.5±1.6**
RoBERTa-large	87.4±2.2	**93.8±1.9**
RoBERTa-large-nli	88.8±1.6	**94.4±1.2**

Table 2: Performance comparison of implicit modeling (end-to-end model) and explicit modeling (CRM) of counterfactual thinking.

CRM *V.S.* implicit modeling. According to the uniform approximation theorem (Hornik, 1991), the CRM can also be approximated by a deep neural network. We thus investigate whether counterfactual thinking can be learned in an implicit manner. In particular, we evaluate a model that takes both the factual sample and counterfactual samples as inputs to make prediction for the factual one. Table 2 shows the performance, where we have the following observations: 1) The implicit modeling performs much worse than the proposed CRM in most cases, which justifies the effectiveness of the retrospection module and the rationality of modeling comparison explicitly. 2) On the NLI task, RoBERTa-base+CRM outperforms RoBERTa-large (implicit), which means that the superior performance of CRM is not because of the additional model parameters introduced by the retrospection module, but the explicit comparison between factual and counterfactual samples.

4.3 In-depth Analysis

Effects of retrospection module design (RQ2). Note that the order of comparison and fusion in the retrospection mechanism of us humans is still unclear. We investigate how the fusion strategies influence the effectiveness of the proposed CRM. Table 3 shows the performance of CRM based on the early fusion (EF), late fusion (LF), and middle fusion (MF) on the NLI task. We omit the comparison on the SA task since the dataset only has one counterfactual sample for the target factual sample. For both EF and LF, we use the mean pooling as the pooling function. As to MF, we use a pooling function that is equipped with self-attention (Vaswani et al., 2017). The reasons of this setting are twofold: 1) using mean pooling will make LF and MF equivalent since the FC layer in the retrospection module is a linear mapping. Note that LF performs pooling after the FC layer, while the pooling function of MF is just before the FC layer. 2) The comparison between the LF and MF can thus shed light on whether parameterized pooling function can benefit the retrospection.

From the table, we can observe that, in most cases, CRM based on different fusion strategies achieve performance comparable to each other. It indicates that the retrospection is insensitive to the order of fusion and the comparison between counterfactual and factual samples. Considering that MF with mean pooling is equivalent to LF, we can see that the benefit of parameterized pooling function is limited. In particular, MF only performs better than LF on one of the three testing models.

Effects of generation module (RQ3). We then investigate whether the proposed generation module constructs useful counterfactual samples for retrospection. We train and test the retrospection module (using EF) with the generated samples on RoBERTa-large on the SA task. We omit the experiments of other settings for saving computation resources. In this way, the model achieves an accuracy of 94.5 which is better than +CF (93.4) but worse than +CRM with manually constructed counterfactual samples (98.2) (*cf.* Table 1). The result indicates that the generated samples indeed facilitate the retrospection while the generation quality can be further improved. Moreover, on the testing samples at confidence level of 1, using the generated samples achieves an accuracy of 81.3 which is much better than +CF (70.8) (*cf.* Figure 3). The generated samples indeed benefit the decision making over hard testing samples.

5 Related Work

Counterfactual sample. Constructing counterfactual samples has become an emergent data augmentation technique in natural language processing, which has been used in a wide spectral of language understanding tasks, including SA (Kaushik et al., 2019; Yang et al., 2020), NLI (Kaushik et al., 2019), named entity recognition (Zeng et al., 2020) question answering (Chen et al., 2020), dialogue system (Zhu et al., 2020), vision-language navigation (Fu et al., 2020). Beyond data augmentation under the standard supervised learning paradigm, a line of research explores to incorporate counterfactual samples into other learning paradigms such as adversarial training (Zhu et al., 2020; Fu et al., 2020; Teney et al., 2020) and contrastive learning (Liang et al., 2020). This work lies in an orthogonal direction that incorporates counterfactual samples into the decision making procedure of model inference.

Counterfactual inference. A line of research attempts to enable deep neural networks with counterfactual thinking by incorporating counterfactual inference (Yue et al., 2021; Wang et al., 2021; Niu et al., 2021; Tang et al., 2020; Feng et al., 2021). These methods perform counterfactual inference over the model predictions according to a pre-defined causal graph. Due to the requirement of causal graph, such methods are hard to be generalized to different tasks. Our method does not suffer from such limitation since working on the counterfactual samples which can be generated without a comprehensive causal graph.

Hard sample. A wide spectral of machine learning techniques are related to dealing with the hard samples in language understanding. For instance,

Backbone	+CF	EF	*RI*	LF	*RI*	MF	*RI*
RoBERTa-base	83.4±0.9	91.5±1.6	9.7%	**92.8±1.8**	11.3%	89.6±2.0	7.4%
RoBERTa-large	85.8±1.2	93.8±1.9	9.3%	**95.3±0.7**	11.1%	93.4±1.7	8.9%
RoBERTa-large-nli	88.2±1.0	94.4±1.2	7.1%	93.8±0.4	6.4%	**94.7±1.3**	7.4%

Table 3: Performance of the proposed CRM based on early fusion (EF), late fusion (LF), or middle fusion (MF) on the NLI task. RI represents the relative performance improvement over the +CF method.

adversarial training (Khashabi et al., 2020) enhances the model robustness against perturbations and attacks, which are hard samples for normally trained models. Debiased training (Tu et al., 2020; Utama et al., 2020) eliminates the spurious correlation or bias in training data to enhance the generalization ability and deal with out-of-distribution samples. In addition to the training phase, a few inference techniques might improve the model performance on hard samples, including posterior regularization (Srivastava et al., 2018) and causal inference (Yu et al., 2020; Niu et al., 2021). However, both techniques require domain knowledge such as prior or causal graph tailored for specific applications. On the contrary, this work provides a general paradigm that can be used for most language understanding tasks.

6 Conclusion

In this work, we pointed out the issue of standard inference of existing language understanding models. We proposed a Counterfactual Reasoning Model which empowers the trained model with a high-level cognitive ability, counterfactual thinking. By applying the proposed CRM, we formed a new paradigm for building language understanding solutions. We conducted extensive experiments, which validate the effectiveness of our proposal, especially in dealing with hard samples.

This work opens up a new research direction about the decision making procedure in testing phase. In the future, we will explore sequential decision procedure to resolve the constraint on the number of constructed counterfactual samples. In addition, we will investigate generation module for language understanding with unsupervised generative techniques (Sauer and Geiger, 2021).

Acknowledgments

This research is supported by the Sea-NExT Joint Lab, Singapore MOE AcRF T2, National Natural Science Foundation of China (U19A2079) and National Key Research and Development Program of China (2020AAA0106000). Our thanks also go to all the anonymous reviewers for their valuable suggestions.

References

Annika Boldt, Anne-Marike Schiffer, Florian Waszak, and Nick Yeung. 2019. Confidence predictions affect performance confidence and neural preparation in perceptual decision making. *Scientific reports*, 9(1):1–17.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. 2020. Counterfactual samples synthesizing for robust visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10800–10809.

Noam Chomsky. 2002. *Syntactic structures*. Walter de Gruyter.

Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. 2019. Addressing failure prediction by learning model confidence. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 2898–2909. Curran Associates, Inc.

Kahneman Daniel. 2017. Thinking, fast and slow.

Jon Dattorro. 2010. *Convex optimization & Euclidean distance geometry*. Lulu. com.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186. ACL.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Fuli Feng, Weiran Huang, Xin Xin, Xiangnan He, and Tat-Seng Chua. 2021. Should graph convolution trust neighbors? a simple causal inference method. In *SIGIR*.

Tsu-Jui Fu, Xin Eric Wang, Matthew F Peterson, Scott T Grafton, Miguel P Eckstein, and William Yang Wang. 2020. Counterfactual vision-and-language navigation via adversarial path sampler. In *European Conference on Computer Vision*, pages 71–86. Springer.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2019. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. 2020. Sentilare: Linguistic knowledge enhanced language representation for sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6975–6988.

Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020. More bang for your buck: Natural perturbation for robust question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–170.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified mrc framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350.

Zujie Liang, Weitao Jiang, Haifeng Hu, and Jiaying Zhu. 2020. Learning to contrast the counterfactual samples for robust visual question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3285–3292.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

Alan Nash. 2003. A generalized parallelogram law. *The American mathematical monthly*, 110(1):52–57.

Yulei Niu, Kaihua Tang, Hanwang Zhang, Zhiwu Lu, Xian-Sheng Hua, and Ji-Rong Wen. 2021. Counterfactual vqa: A cause-effect look at language bias. In *CVPR*.

Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. 2018. Learning independent causal mechanisms. In *International Conference on Machine Learning*, pages 4036–4044. PMLR.

Judea Pearl. 2019. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60.

Axel Sauer and Andreas Geiger. 2021. Counterfactual generative networks. *The International Conference on Learning Representations*.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316.

Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. 2020. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *NeurIPS*, 33.

Damien Teney, Ehsan Abbasnedjad, and Anton van den Hengel. 2020. Learning what makes a difference from counterfactual examples and gradient supervision. *arXiv preprint arXiv:2004.09034*.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Mind the trade-off: Debiasing nlu models without degrading the in-distribution performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8717–8729.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. " click" is not equal to" like": Counterfactual recommendation for mitigating clickbait issue. In *SIGIR*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. Corefqa: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963.

Linyi Yang, Eoin Kenny, Tin Lok James Ng, Yi Yang, Barry Smyth, and Ruihai Dong. 2020. Generating plausible counterfactual explanations for deep transformers in financial text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6150–6160.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4805–4815.

Sicheng Yu, Yulei Niu, Shuohang Wang, Jing Jiang, and Qianru Sun. 2020. Counterfactual variable control for robust and interpretable question answering. *arXiv preprint arXiv:2010.05581*.

Zhongqi Yue, Tan Wang, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. 2021. Counterfactual zero-shot and open-set visual recognition. In *CVPR*.

Xiangji Zeng, Yunliang Li, Yuchen Zhai, and Yin Zhang. 2020. Counterfactual generator: A weakly-supervised method for named entity recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7270–7280.

Qingfu Zhu, Weinan Zhang, Ting Liu, and William Yang Wang. 2020. Counterfactual off-policy training for neural dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3438–3448.

Knowledge-Empowered Representation Learning for Chinese Medical Reading Comprehension: Task, Model and Resources

Taolin Zhang[1,2]*, **Chengyu Wang**[2]*, **Minghui Qiu**[2], **Bite Yang**[3]
Zerui Cai[4], **Xiaofeng He**[4]†, **Jun Huang**[2]
[1] School of Software Engineering, East China Normal University [2] Alibaba Group
[3] DXY [4] School of Computer Science and Technology, East China Normal University
`zhangtl0519@gmail.com, zrcai_flow@126.com, yangbt@dxy.cn`
`{chengyu.wcy, minghui.qmh, huangjun.hj}@alibaba-inc.com`
`hexf@cs.ecnu.edu.cn`

Abstract

Machine Reading Comprehension (MRC) aims to extract answers to questions given a passage, which has been widely studied recently especially in open domains. However, few efforts have been made on closed-domain MRC, mainly due to the lack of large-scale training data. In this paper, we introduce a multi-target MRC task for the medical domain, whose goal is to predict answers to medical questions and the corresponding support sentences from medical information sources simultaneously, in order to ensure the high reliability of medical knowledge serving. A high-quality dataset (more than 18k samples) is manually constructed for the purpose, named Multi-task **C**hinese **Med**ical **MRC** dataset (CMedMRC), with detailed analysis conducted. We further propose a **C**hinese **med**ical **BERT** model for the task (CMedBERT), which fuses medical knowledge into pre-trained language models by the dynamic fusion mechanism of heterogeneous features and the multi-task learning strategy. Experiments show that CMedBERT consistently outperforms strong baselines by fusing context-aware and knowledge-aware token representations.[1]

1 Introduction

Machine Reading Comprehension (MRC) has become a popular task in NLP, aiming to understand a given passage and answer the relevant questions. With the wide availability of MRC datasets (Rajpurkar et al., 2016; He et al., 2018; Cui et al., 2019) and deep learning models (Yu et al., 2018; Ding et al., 2019) (including pre-trained language models such as BERT (Devlin et al., 2019)), significant progress has been made.

Despite the success, a majority of MRC research has focused on open domains. For specific domains, however, the construction of high-quality MRC datasets, together with the design of corresponding models is considerably deficient (Welbl et al., 2017, 2018). The causes behind this phenomenon are threefold. Take the medical domain as an example. i) Data annotators are required to have medical backgrounds with high standards. Hence, simple crowd-sourcing (Rajpurkar et al., 2016; Cui et al., 2019) often leads to poor annotation results. ii) Due to the domain sensitivity, people are more concerned about the reliability of the information sources where the answers are extracted, and the explainability of the answers themselves (Lee et al., 2014; Dalmer, 2017). This is fundamentally different from the task requirements of open-domain MRC. iii) From the perspective of model learning, it is difficult for pre-trained language models to understand the meaning of the questions and passages containing a lot of specialized terms (Chen et al., 2016; Bauer et al., 2018). Without the help of domain knowledge, state-of-the-art models can perform poorly. As shown in Figure 1, BERT (Devlin et al., 2019) and MC-BERT (Zhang et al., 2020) only predict part of the correct answer, i.e., “torso” and “buttocks”, instead of generating the complete answer to the medical question.

In this paper, we present a comprehensive study on Chinese medical MRC, including i) how the task is formulated, ii) the construction of the Chinese medical dataset and iii) the MRC model with rich medical knowledge injected. To meet the requirements of medical MRC, we aim to predict both the answer spans to a medical question, and the support sentence from the passage, indicating the source of the answer. The support sentences provide abundant evidence for users to learn medical knowledge, and for medical professionals to assess the trustworthiness of model output results.

---

*T. Zhang and C. Wang contributed equally to this work.

†Corresponding author.

[1] The code and dataset will be available at `https://github.com/MatNLP/CMedMRC`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2237–2249
August 1–6, 2021.

Multi-task Chinese Medical MRC

Passage: <1>叠瓦癣是一种特殊的体癣, 主要由同心性毛癣菌或称叠瓦癣菌引起…<14>躯干和臀部多见，时久可扩延于四肢，甚至口唇、甲沟及头皮<15>但掌跖多不受累，也不侵犯毛发 (<1>Imbricate tinea is a special kind of body ringworm, mainly caused by the concentric trichoderma or imbricated versicolor bacterium… <14>It often occurs in the torso and buttocks. The disease can spread to the extremities after a long time. It can even spread to the lips, nail groove and scalp. <15>But the metatarsal is not affected and also do not damage the hair...)

Question: 叠瓦癣可发生在什么部位？

(What part of body can imbricate tinea happen?)

Support sentence: 14

BERT_base Answer: 躯干和臀部 (Torso and buttocks)

MC-BERT Answer: 躯干和臀部多见，时久可扩延于四肢 (It often occurs in the torso and buttocks. The disease can spread to the extrmities after a long time.)

CMedBERT Answer: 躯干和臀部多见，时久可扩延于四肢，甚至口唇、甲沟及头皮 (It often occurs in the torso and buttocks. The disease can spread to the extremities after a long time. It can even spread to the lips, nail groove and scalp.)

Medical Entities: 1. 四肢瘫 (quadriplegia) 2. 皮 (skin) 3. 甲沟炎 (paronychia) 4. 甲沟 (nail groove)

Figure 1: Dataset example. MC-BERT and BERT can only predict part of the correct answer. With medical knowledge fused, our CMedBERT model can extract the complete answer. Contents in brackets refer to English translations.

For the dataset, we construct a highly-quality Chinese medical MRC dataset, named the Multi-task Chinese Medical MRC dataset (CMedMRC). It contains 18,153 <*question, passage, answer, support sentence*> quads. Based on the analysis of CMedMRC, we summarize four special challenges for Chinese medical MRC, including *long-tail terminologies, synonym terminology, terminology combination* and *paraphrasing*. In addition, we find that comprehensive skills are required for MRC models to answer medical questions correctly. For answer extraction in CMedMRC, direct *token matching* is required for answering 31% of the questions, *co-reference resolution* for 11%, *multi-sentence reasoning* for 18% and *implicit causality* for 22%. In addition, the answers to the remaining questions (16%) are extremely difficult to extract without rich medical background knowledge.

To address the medical MRC task, we propose the multi-task dynamic heterogeneous fusion network (CMedBERT) based on the MC-BERT (Zhang et al., 2020) model and a Chinese medical knowledge base (see Appendix). The technical contributions of CMedBERT are twofold:

- **Heterogeneous Feature Fusion**: We mimic humans' approach of reading comprehension (Wang et al., 1999) by learning attentively aggregated representations of multiple entities in the passage. Different from the knowledge fusion method used by KBLSTM (Yang and Mitchell, 2017) and KT-NET (Yang et al., 2019), we propose a two-level attention and a gated-loop mechanism to replace the knowledge sentinel, so that rich knowledge representations can be better integrated into the model.
- **Multi-task Learning**: Parameters of CMedBERT are dynamically learned by capturing the relationships between the two tasks via multi-task learning. We regard the semantic similarities between support sentences and answers to questions as the task similarities.

We compare CMedBERT against six strong baselines. For answer prediction, compared to the strongest competitor, the EM (Exact Match) and F1 scores are increased by +3.88% and +1.46%, respectively. Meanwhile, the support sentence prediction task result is increased by a large margin, i.e., +7.81% of EM and +4.07% of F1.

2 Related Work

MRC Datasets and Models. Due to the popularity of the MRC task, there exist many types of MRC datasets, such as span-extraction (Rajpurkar et al., 2016; Yang et al., 2018), multiple choices (Richardson et al., 2013; Lai et al., 2017), cloze-style (Hermann et al., 2015), cross-lingual (Jing et al., 2019; Yuan et al., 2020). For specific domains, however, the number of publicly available MRC datasets remains few, including SciQ (Welbl et al., 2017), Quasar-S (Dhingra et al., 2017) and Biology (Berant et al., 2014). CLiCR (Suster and Daelemans, 2018) is a cloze-style single-task English medical MRC dataset. However, it contains a relatively small variety of medical questions, automatically generated from clinical case reports. Recently, (Li et al., 2020) propose a multi-choice Chinese medical QA dataset, retrieving text snippets as the passage and the task only chooses an existing correct option from candidate set. Our work specifically focuses on the fine-grained medical MRC tasks and deep domain knowledge reasoning, with a manually constructed high-quality dataset released.

The model architecture of MRC mostly takes advantage of neural networks to learn token representations of passages and questions jointly (Qiu et al., 2019a; Liu et al., 2019). The interaction between questions and passages is modeled based on attention mechanisms. The rapid development of

Challenges	Characteristics	Example
Lexical-Level	***Long-tail terminology***	冈上肌肌腱断裂试验是对冈上肌肌腱是否存在断裂进行检查。(*supraspinatus tendon* rupture test is to check whether the supraspinatus tendon is ruptured.)
	Synonym terminology	...本药品对过敏性鼻炎和上呼吸道感染引起的鼻充血有效，可用于感冒或鼻窦炎... (...This medicine is effective for nasal congestion caused by allergic rhinitis and *upper respiratory tract infection*, and can be used for *colds* or sinusitis...)
	Terminology combination	...糖尿病性视网膜病(diabetic retinopathy)是糖尿病性微血管病变中最重要的表现... (...*Diabetic retinopathy (DR)* is the most important manifestation of diabetic microangiopathy...)
Sentence-Level	***Paraphrasing***	***Passage***:...如果在嘴角烂了或结痂的地方进行冷敷，一方面冷敷物品不干净的话会造成感染；另一方面局部温度降低了之后，反而会延缓伤口的愈合。...(...If you apply a cold compress on a rotten or crusted corner of the mouth, on the one hand, if the cold compress is not clean, it will *cause infection*; on the other hand, when the local temperature is lowered, it will *delay the healing of the wound*...) ***Question***:为什么嘴角烂了或结痂不建议进行冷敷？(Why is it *not recommended* to apply cold compresses when the corners of the mouth are rotten or crusted?)

Table 1: Two levels of challenges in processing Chinese medical texts. The blue and underscore contents in brackets indicate why this example belongs to its corresponding "Characteristics" category. (Best viewed in color.)

deep learning leads to a variety of models, such as the QANet (Yu et al., 2018), SAN (Liu et al., 2018). Graph neural networks have been used in MRC recently by modeling the relations between entities in the passage (Ding et al., 2019) and multi-grained tokens representation (Zheng et al., 2020).

Pre-trained Language Models and Knowledge Fusion. Pre-trained language models (e.g., BERT (Devlin et al., 2019), ERNIE-THU (Zhang et al., 2019), K-BERT (Liu et al., 2020a)) have successfully improved the performance of the MRC task, which even exceed the human level in some datasets. This is because these models obtain better token representations and capture lexical and syntactic knowledge in different layers (Guan et al., 2019). For specific domain, there also have some pre-trained models (Beltagy et al., 2019; Zhang et al., 2020).

A potential drawback is that pre-trained language models of open domains only learn general representations, lacking domain-specific knowledge to deepen the understanding of entities and other nouns (Ostendorff et al., 2019) (which are often the answers in span-extraction MRC tasks). Without proper descriptions of such entities in the passage, MRC models often fail to understand and extract key information (Das et al., 2019). Hence, the explicit fusion of knowledge in MRC models is vital for learning context-aware token representations (Pan et al., 2019; Qiu et al., 2019b; Liu et al., 2020b). Instead of encoding entities appearing in both knowledge bases and passages into the MRC model only (Chen et al., 2018), our proposed model encodes all the triples from a medical KG and then employs heuristic rules to retrieve relevant entities. This practice allows the model to acquire deeper understanding of domain-specific terms.

3 The CMedMRC Dataset

In this section, we briefly describe the collection process and provide an analysis on various aspects of the CMedMRC dataset. For more dataset collection and statistical analysis of dataset details, we refer readers to the Appendix A and Appendix B.

3.1 Dataset Collection Process

The dataset collection process follows the SQuAD-style (Rajpurkar et al., 2016) rather than collecting question-answer pairs as in Google Natural Questions (Kwiatkowski et al., 2019). Our medical text corpus is collected from DXY Medical [2], an authoritative medical knowledge source in China. The general data collection process of CMedMRC consists of four major steps: passage collection, question-answer pair collection, support sentence selection and additional answer construction. Briefly speaking, during the passage collection process, we filter the corpus to generate high-quality medical passages. A group of human annotators are required to ask questions on medical knowledge and annotate the answers from these passages. The annotation results are in the form of question-answer pairs. Following SQuAD, we ask annotators to provide 2 additional answers for each question in the DEV and TEST sets.

Since people are concerned about the scientific explanation and sources of answers in the medical domain, we ask annotators to select the support sentence of their annotated answer similar to those of CoQA (Reddy et al., 2019) and QuAC (Choi et al., 2018). Finally, CMedMRC consists of three

---

[2] `http://www.dxy.cn/`

Skills	Example	Percentage
Token matching	***Passage***: ...急性羊水过多较少，见多发生在孕20～24周，羊水急剧增多，子宫短期内明显增大... (...it is less likely to secrete too much acute *amniotic fluid*. The disease is most common in the 20 to 24 weeks of *pregnancy*. The amniotic fluid increases sharply with the uterus enlarged significantly in the short term...) ***Question***: 怀孕期间羊水什么时候分泌过多? (When does the *amniotic fluid* secrete too much during *pregnancy*?) ***Answer***: 20～24周 (20～24 weeks)	31%
Co-reference resolution	***Passage***: ...尖锐湿疣有「割韭菜」的臭名声，它的治疗瓶颈在于病毒不进入血循环，因此机体无法产生免疫应答，所以容易反复复发。...(...*genital warts* has a bad reputation of cutting leeks. The bottleneck of *its* treatment is that the virus does not enter the blood circulation, so the body cannot produce an immune response and *it* is easy to relapse repeatedly....) ***Question***: 为什么尖锐湿疣易反复 发作? (Why *genital warts* is easy to relapse repeatedly?) ***Answer***: 病毒不进入血循环，因此机体无法产生免疫应答 (The virus does not enter the blood circulation, so the body cannot produce an immune response)	11%
Multi-sentence reasoning	***Passage***: ... 老年人应在医师指导下使用。5.肝、肾功能不全者慎用。6.孕妇及哺乳期妇女慎用。... (...*The elderly should take the medicine under the guidance of a physician. 5. Use with caution in patients with liver and kidney insufficiency. 6. Use with caution in pregnant and lactating women.*) ***Question***: 哪些人群慎用此药品? (Which groups of people should use this drug with caution?) ***Answer***:老年人应在医师指导下使用。5.肝、肾功能不全者慎用。6.孕妇及哺乳期妇女慎用 (The elderly should take the medicine under the guidance of a physician. 5. Use with caution in patients with liver and kidney insufficiency. 6. Use with caution in pregnant and lactating women.)	18%
Implicit causality	***Passage***: ... 不是所有的白细胞减少都必须治疗的， 关键看白细胞减少的程度、机体的一般状态以及医生的建议；...... 因为无症状的白细胞减少对生活的影响是很小的； ... (...*Not all leukopenia must be treated.* The key depends on the degree of leukopenia, the general state of the body and the doctor's advice;......*Because asymptomatic leukopenia has little impact on life*...) ***Question***: 为什么不是所有的白细胞减少都要进行治疗? (Why do *not all leukopenia have to be treated?*) ***Answer***:因为无症状的白细胞减少对生活的影响是很小的 (Because asymptomatic leukopenia has little impact on life.)	22%
Domain knowledge	***Passage***:...发病率居遗传性血小板功能缺陷疾病的首位。血栓细胞衰弱发病多见于幼年，发病率为 0.01/万... (...The incidence is the highest in hereditary platelet dysfunction diseases. The incidence of *thrombotic cell weakness* is more common in childhood with an incidence rate of 0.01 / 10,000...) ***Question***:血小板无力症的发病率约为多少? (What is the incidence rate of *blood platelet weakness?*) ***Answer***: 0.01/万 (0.01/10,000)	16%

Table 2: Reading comprehension skills of models required to answer questions in CMedMRC. The blue and underscore contents in brackets indicate why the sample belongs to its category. (Best viewed in color)

parts: 12,700 training samples, 3,630 development samples and 1,823 testing samples.

3.2 Quality Control

During the dataset collection process, we take the following measures to ensure the quality of the dataset. i) The knowledge source (DXY Medical) contains high-quality medical articles which are written by medical personnel and organized based on different topics in the medical domain. ii) Our annotators are all engaged in medical-related professions rather than annotators with short-term guidance only. iii) We further hire 12 medical experts to check all the collected samples rather than checking a randomly selected sample only. The experts remove out-of-domain questions and questions that are unhelpful to medical practice. In this stage, the experts are divided into two groups and cross-check their judgments.

3.3 Challenges of Understanding Texts

Due to the closed-domain property of our dataset, there are some domain-specific textual features in both passages and questions that the model needs to understand. Based on our observations of the CMedMRC, we summarize the following two major challenges. These challenges can be also regarded as key reasons why some recent state-of-the-art MRC models cannot address the medical MRC task on CMedMRC well.

Lexical-Level: i) Long-tail terminology means these medical terms occur very infrequently and are prone to Out-Of-Vocabulary (OOV) problems. ii) Synonym terminology means that some medical terms may express the same meaning, but there is a distinction between colloquial expressions and professional terms. The above two points require the model to have rich domain knowledge to solve. iii) Terminology combination means these terms are usually formed by a combination of multiple terms, while one term is the attributive of another. This does not only require the model to have domain knowledge but also poses challenges to phrase segmentation in specific domains.

Sentence-Level: Paraphrasing means some words in questions are semantically related to certain tokens in passages, but are expressed differently. Consider the last question in Table 1. When the model tries to answer the "not-recommended" question, it should focus on negative terms ("cause infection" and "delay the healing of the wound").

3.4 Reasoning Skills for MRC Models

We randomly select 100 samples from the development set to analyze what skills the model should have in order to answer the questions correctly. We divide the reasoning skills corresponding to these samples into five major categories, namely token matching, co-reference resolution, multi-sentence reasoning, implicit causality and domain knowledge. Examples are shown in Table 2. It is particularly noteworthy that the fifth type is the need of domain knowledge to answer medical questions. Consider the example:

Passage: ... The incidence of thrombotic cell weakness is most common in childhood with an incidence rate of 0.01 / 10,000...

Question: What is the incidence rate of blood platelet weakness?

Answer: 0.01/10,000.

We know that the *blood platelet* in the question refers to the *thrombotic cell* described in the passage through the medical knowledge base. It shows that the rich information of the knowledge base can help the model obtain a better understanding of domain terms to improve the MRC performance.

4 The CMedBERT Model

4.1 Task Formulation and Model Overview

For our task, the input includes a medical question Q together with the passage P. Let $\{p_1, p_2, \cdots p_m\}$ and $\{q_1, q_2, \cdots q_n\}$ represent the passage and question tokens, respectively. In the answer prediction task, the goal is to train an MRC model which extracts the answer span $\{p_i, p_{i+1}, \cdots p_j\}$ $(0 \le i \le j \le m)$ from P that correctly answers the question Q. Additionally, the model is required to predict the support sentence tokens $\{p_k, p_{k+1}, \cdots p_l\}$ $(0 \le k \le l \le m)$ from P to provide additional medical knowledge and to enhance interpretability of the extracted answers. We constrain that $\{p_k, p_{k+1}, \cdots p_l\}$ must form a complete sentence, instead of incomplete semantic units and the support sentence tokens contain the answer span. The high-level architecture of the CMedBERT model is shown in Figure 2. It mainly includes four modules: BERT encoding, knowledge embedding and retrieval, heterogeneous feature fusion and multi-task training.

4.2 BERT Encoding

This module is used to learn context-aware representations of question and the passage tokens. For each input pair (the question Q and the passage P), we treat $[\langle CLS\rangle, Q, \langle SEP\rangle, P, \langle SEP\rangle]$ as the input sequences for BERT. We denote $\{h_i\}_{i=1}^{m+n+3}$ as the hidden layer representations of tokens, where m and n are the length of passage tokens and question tokens, respectively.

4.3 Knowledge Embedding and Retrieval

In the knowledge bases, relational knowledge is stored in the form of (*subject, relation, object*) triples. In order to fuse knowledge into token representations, we first encode all entities in the knowledge base into a low-dimensional vector space. Here, we employ PTransE (Lin et al., 2015a) to learn entity representations, and denote the underlying entity embedding as e_i. Because existing medical NER tools do not have high coverage over our corpus, we consider five types of token strings as candidate entities: noun, time, location, direction and numeric. Two matching strategies are then employed to retrieve relevant entities from the knowledge base: (i) The two strings match exactly. (ii) The number of overlapped tokens is larger than a threshold. After relevant entities are retrieved, we can fuse the knowledge into contextual representations, introduced below.

4.4 Heterogeneous Feature Fusion

In this module, we fuse heterogeneous entity features retrieved from the knowledge base into the question and passage tokens representations.

Local Fusion Attention. We observe that each token is usually related to multiple entities of varying importance. Thus, we assign different weights to the entity embedding e_j corresponding to the token representation h_i using attention mechanism:

$$\alpha_{i,j} = \frac{exp(e_j^T W h_i)}{\sum_{k=1}^{K} exp(e_k^T W h_i)} \tag{1}$$

where K is the number of entities and $\alpha_{i,j}$ represents the similarity between the j^{th} entity in the retrieved entity set and the i^{th} token. $W \in \mathbb{R}^{d_2 \times d_1}$ where d_1 is the dimension of BERT's output and d_2 is the dimension of entity embeddings. After fusing, the representation of the i^{th} token is: $\bar{e}_i = \sum_{k=1}^{K} \alpha_{i,k} e_k$. However, $\bar{e}_i$ is only related to retrieved entities, not other tokens in the question-passage pair.

Global Fusion Attention. In BERT, the output of the $[CLS]$ tag represents the entire sequence information learned by transformer encoders. We use

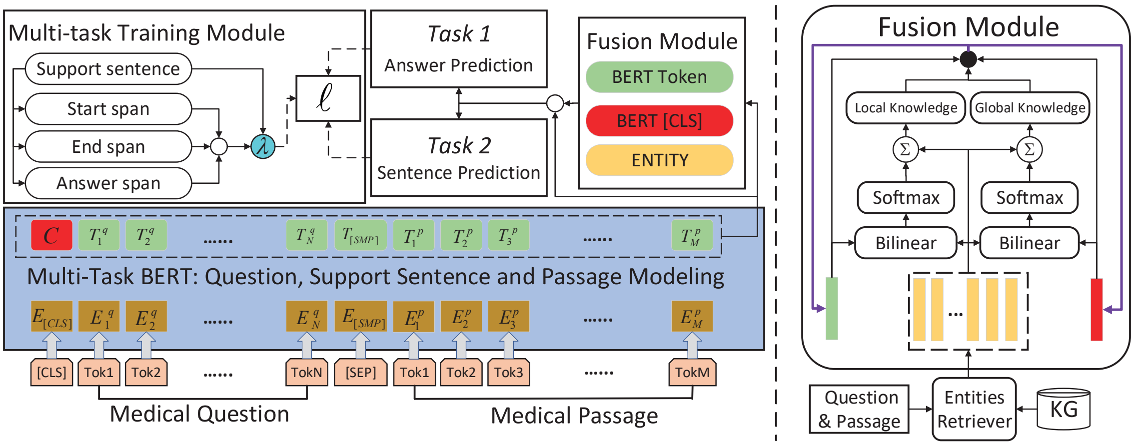

Figure 2: Model overview. The green box and the red box in the heterogeneous feature fusion layer represent the local and global token information, respectively. In the multi-task training module, the model learns the relationship between two tasks by dynamically learning the parameter λ. (Best viewed in color)

the token output $h_{[CLS]}$ to model the knowledge fusion representation of the entire entity collection that each token recalls:

$$\beta_{[CLS],j} = \frac{exp(e_j^T W h_{[CLS]})}{\sum_{k=1}^{K} exp(e_k^T W h_{[CLS]})} \qquad (2)$$

$$\hat{e}_i = \sum\nolimits_{k=1}^{K} \beta_{[CLS],k} e_k \qquad (3)$$

where $\hat{e}_i$ is the global knowledge fusion result corresponding to the i^{th} token.

Gated Loop Layer. In order to fuse local and global results into token representations, we design a gated loop layer. The information of knowledge fusion is filtered through the gating mechanism in each loop of modeling. In the initialization stage, we simply have $h_i^0 = h_i$. In the l^{th} iteration, we have the following update process:

$$G_i^\ell = \sigma(tanh(W[h_i^\ell, \bar{e}_i, \hat{e}_i])) \qquad (4)$$

$$h_i^{\ell+1} = G_i^\ell \odot h_i^\ell \qquad (5)$$

This process runs for L loops and this fusion process output is h_i^L. The loop process mimics the human's behavior of reading the passage repeatedly to find the most accurate answers.

4.5 Multi-task Training

The output layer of CMedBERT is extended from BERT. We first concatenate two types of token representations and calculate the probability of the i^{th} token being selected in the support sentence as follows:

$$o_i = \sigma(W[h_i, h_i^L]),\ p_i^{support} = \sigma(W o_i) \qquad (6)$$

We also calculate its probabilities as the starting and the ending positions of the answer span, respectively:

$$p_i^{start} = \frac{exp(w_1^T o_i)}{\sum_j exp(w_1^T o_j)},\ p_i^{end} = \frac{exp(w_2^T o_i)}{\sum_j exp(w_2^T o_j)}$$

The loss function of the answer prediction task is the negative log-likelihood of the starting and ending positions of ground-truth answer tokens:

$$\mathcal{L}_{\mathcal{A}} = -\frac{1}{N}\sum_{j=1}^{N}(logp^{start}_{y_j^{start}} + logp^{end}_{y_j^{end}}) \qquad (7)$$

For the extraction of support sentences, the loss function is defined by cross-entropy:

$$\mathcal{L}_{\mathcal{S}} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M}(y_j^{support} logp_i^{support}) \qquad (8)$$

where N is the number of samples and M is the length of input sequences. y_j^{start},y_j^{end} is the starting and ending positions of ground-truth of the j^{th} token. Furthermore, if the token is in the support sentence, the token label $y_j^{support}$ is set to 1, and 0 otherwise.

The representations of the support sentence are related to the positions of the answer. In order to better model the relationship between two tasks, we dynamically learn the coefficient between the loss values of two tasks. Let h_{su} and o_{sp} be self-attended, averaged pooled representations of the support sentence and the answer span. o_{st}, o_{ed} are the start and end position token representations of

Model	Answer				Support Sentence			
	Exact Match (EM)		F1		Exact Match (EM)		F1	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
DrQA	42.00%	37.45%	58.66%	57.15%	5.07%	5.88%	30.24%	32.52%
BERT_base	64.83%	68.31%	81.08%	83.74%	21.42%	17.70%	52.27%	48.34%
ERNIE	65.49%	68.57%	81.17%	83.86%	20.71%	16.32%	49.88%	45.81%
KT-NET$^\spadesuit$	64.58%	69.03%	81.06%	84.18%	15.42%	13.48%	49.37%	46.45%
MC-BERT	66.58%	68.62%	81.23%	83.98%	20.08%	16.77%	47.33%	44.53%
KMQA	66.19%	68.45%	81.20%	83.79%	20.63%	16.54%	49.27%	45.97%
CMedBERT$^\clubsuit$	69.00%	72.84%	82.68%	85.38%	25.17%	24.18%	52.36%	49.69%
CMedBERT$^\spadesuit$	**70.33%**	**72.91%**	**83.43%**	**85.64%**	**25.58%**	**25.51%**	**52.67%**	**52.41%**

Table 3: The results of multi-task prediction (answer and support sentence) over CMedMRC.

the answer, respectively. We have:

$$\gamma_{st}, \gamma_{ed}, \gamma_{sp} = h_{su}[o_{st}, o_{ed}, o_{sp}]^T \tag{9}$$

$$H_A = \sigma(W[\gamma_{st} o_{st}, \gamma_{ed} o_{ed}, \gamma_{sp} o_{sp}]) \tag{10}$$

where $\gamma_{st}, \gamma_{ed}, \gamma_{sp}$ are the weight coefficients between the supporting sentence and the start/end/total token representations of the answer span. The loss value coefficient of two tasks λ and the total loss $\mathcal{L}$ are as follows:

$$\lambda = max\{0, \cos(H_A, h_{su})\} \tag{11}$$

$$\mathcal{L} = \mathcal{L}_A + \lambda \mathcal{L}_S \tag{12}$$

We minimize the total loss $\mathcal{L}$ to update our model parameters in the training process.

5 Experiments and Result Analysis

5.1 Experimental Setups

We evaluate CMedBERT on CMedMRC, and compare it against six strong baselines: DrQA (Chen et al., 2017), BERT_base (Devlin et al., 2019), ERNIE (Zhang et al., 2019), KT-NET (Yang et al., 2019), MC-BERT (Zhang et al., 2020) and KMQA (Li et al., 2020). KT-NET is the first model to leverage rich knowledge to enhance pre-trained language models for MRC. MC-BERT is the first Chinese biomedical pre-trained model fine-tuned on BERT_base. We only use the encoder layer in KMQA removing the answer layer due to the different answer type.

For evaluation, we use EM (Exact Match) and F1 metrics for answer and support sentence tasks. We calculate character-level overlaps between prediction and ground truth for the Chinese language, rather than token-level overlaps for English. To assess the difficulty of solving CMedMRC tasks, we select 100 testing samples to evaluate human performance. Human scores of EM and F1 are 85.00% and 96.69% for answer prediction, respectively.

Model	Exact Match (EM)		F1	
	Dev	Test	Dev	Test
DrQA	34.50%	32.10%	56.67%	56.64%
BERT_base	62.39%	68.29%	81.48%	83.70%
ERNIE	63.18%	66.92%	81.74%	83.41%
KT-NET$^\spadesuit$	64.64%	66.26%	82.48%	83.61%
MC-BERT	63.39%	68.38%	81.86%	83.88%
KMQA	64.37%	67.48%	81.95%	83.74%
CMedBERT$^\clubsuit$	68.00%	72.11%	82.50%	85.33%
CMedBERT$^\spadesuit$	**69.83%**	**72.84%**	**83.02%**	**85.54%**
Human	-	85.00%	-	96.69%

Table 4: Result of single-task (answer prediction). $\clubsuit$ and $\spadesuit$ indicate that CMedBERT uses BERT_base and MC-BERT as the encoder, respectively.

In the implementation, we set the learning rate as 5e-5 and the batch size as 16, and the max sequence length as 512. Other BERT's hyper-parameters are the same as in Google's settings [3]. Each model is trained for 2 epochs by the Adam optimizer (Kingma and Ba, 2015). Results are presented in average with 5 random runs with different random seeds. Other implementation details are in Appendix C.

5.2 Model Results

Table 3 and Table 4 show the multi-task and single-task results on the CMedMRC development and testing sets. CMedBERT has a great improvement compared to four strong baseline models in both tasks. Specifically, our CMedBERT outperforms the state-of-the-art model by a large margin in multi-task results, with **+3.88%EM / +1.46%F1**

---

[3] `https://github.com/google-research/bert`

Model	Answer		Sentence	
	EM	F1	EM	F1
CMedBERT♠	**72.91%**	**85.64%**	**25.51%**	**52.41%**
w/o Local Att.	68.93%	83.89%	19.75%	49.45%
w/o Global Att.	71.71%	84.96%	17.59%	47.01%
w/o λ	71.91%	85.09%	21.09%	48.80%

Table 5: Ablation study of CMedBERT (testing set).

improvements, which shows the effectiveness of our model. Meanwhile, in the support sentence task, our model also has the best performance, improving (**+7.81%EM / +4.07%F1**) over the testing set. In single task evaluation, we remove the support sentence training module and the dynamic parameter for loss function module. Our model improves (**+4.46%EM / +1.66%F1**) over the best baseline model. In addition, we find that using support sentence prediction as an auxiliary task and the pre-training technique in medical domain can further improve the performance of CMedBERT.

5.3 Ablation study

In Table 5, we choose three important model components for our ablation study and report the results over the testing set. When the dynamic parameter λ of the loss function is removed from the model, the performance of the model on two tasks is decreased by (**-1.00%EM and -0.55%F1**) and (**-4.42%EM and -3.61%F1**), respectively. Without local attention, the EM performance in the answer prediction task decreases by (**-3.98%EM and -1.75%F1**). Experiments have shown that the model performs worse without the local fusion attention than without the global fusion attention and the dynamic parameter λ. However, the performance of support sentence task is decreased significantly by (**-7.92%EM and -3.61%F1**) without global fusion attention. It shows that local fusion attention is more important for extracting answer spans, while global fusion attention plays a larger role in support sentence prediction.

5.4 Case Study

In Figure 3, we use our motivation example to conduct a case study. In BERT, we can see that the difference among the probability values of different words is small, especially when predicting the probability of ending positions. The ending position probabilities of token “部” token “皮” are **0.3** and **0.25**, leading the model to extract the wrong answer span. However, in the knowledge retrieval module of CMedBERT, multiple entities representation are fused into the context-aware latent space representation to enhance the medical text semantic understanding. Therefore, in our CMedBERT model, the starting position probability is **0.95** and the end position probability is **0.8**. In this case, the CMedBERT model can easily choose the correct range of the answer span.

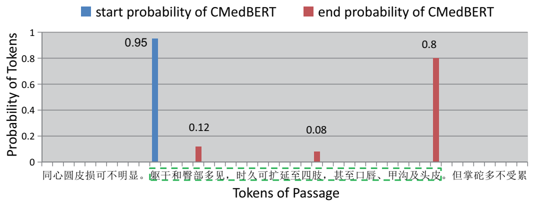

(a) Tokens probabilities of **CMedBERT♠**

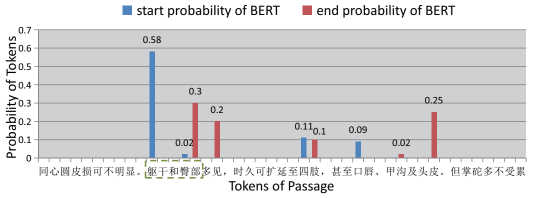

(b) Tokens probabilities of **BERT_base**

Figure 3: Case study. Predicted answer spans are in the green dotted box. Product of the maximum starting and ending probabilities of CMedBERT is **0.76**, with BERT to be **0.174**.

5.5 Discussion of Support Sentence Task

Compared with the answer prediction task, existing models have poor prediction results on the EM metric in the support-sentence task. In prediction results, we randomly select 100 samples for analysis. We divide the error types into the following three main types (see Appendix): i) starting position cross ii) ending position cross iii) answer substring. The most common error type is the answer substring, accounting for **46%**. In this error type, the predicted result of our model is part of the true result, which shows the model cannot predict long answers completely (Yuan et al., 2020) and reduce the accuracy of the results greatly.

6 Conclusion

In this work, we address medical MRC with a new dataset **CMedMRC** constructed. An in-depth analysis of the dataset is conducted, including statistics, characteristics, required MRC skills, etc. Moreover, we propose the **CMedBERT** model, which can help the pre-trained model better understand domain terms by retrieving entities from medical knowledge bases. Experimental results confirm the effectiveness of our model. In the future, we will explore how knowledge can improve the performance of models in the medical domain.

Acknowledgements

We would like to thank anonymous reviewers for their valuable comments. This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904, and Alibaba Group through Alibaba Research Intern Program.

References

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *EMNLP*, pages 4220–4230.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP*, pages 3613–3618.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*, pages 1870–1879.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *ACL*, pages 2406–2417.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *EMNLP*, pages 2174–2184.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. A span-extraction dataset for chinese machine reading comprehension. In *EMNLP*, pages 5882–5888.

Nicole K Dalmer. 2017. Questioning reliability assessments of health information on social media. *Journal of the Medical Library Association: JMLA*, 105(1):61.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *CoRR*, abs/1707.03904.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *ACL*, pages 2694–2703.

Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. 2019. Towards a deep and unified understanding of deep neural models in NLP. In *ICML*, pages 2454–2463.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. Dureader: a chinese machine reading comprehension dataset from real-world applications. In *ACL*, pages 37–46.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NeurIPS*, pages 1693–1701.

Yimin Jing, Deyi Xiong, and Yan Zhen. 2019. Bipar: A bilingual parallel dataset for multilingual and cross-lingual reading comprehension on novels. In *EMNLP*, pages 2452–2462.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *EMNLP*, pages 785–794.

Kenneth Lee, Kreshnik Hoti, Jeffery D Hughes, and Lynne M Emmerton. 2014. Interventions to assist health consumers to find reliable online health information: a comprehensive review. *PloS one*, 9(4):e94186.

Dongfang Li, Baotian Hu, Qingcai Chen, Weihua Peng, and Anqi Wang. 2020. Towards medical machine reading comprehension with structural knowledge and plain text. In *EMNLP*, pages 1427–1438.

Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. 2019. Neural machine reading comprehension: Methods and trends. *CoRR*, abs/1907.01118.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020a. K-BERT: enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *ACL*, pages 1694–1704.

Ye Liu, Shaika Chowdhury, Chenwei Zhang, Cornelia Caragea, and Philip S. Yu. 2020b. Interpretable multi-step reasoning with knowledge extraction on complex healthcare question answering. *CoRR*, abs/2008.02434.

Malte Ostendorff, Peter Bourgonje, Maria Berger, Julián Moreno Schneider, Georg Rehm, and Bela Gipp. 2019. Enriching BERT with knowledge graph embeddings for document classification. In *KONVENS*.

Xiaoman Pan, Kai Sun, Dian Yu, Jianshu Chen, Heng Ji, Claire Cardie, and Dong Yu. 2019. Improving question answering with external knowledge. In *EMNLP*, pages 27–37.

Boyu Qiu, Xu Chen, Jungang Xu, and Yingfei Sun. 2019a. A survey on neural machine reading comprehension. *CoRR*, abs/1906.03824.

Delai Qiu, Yuanzhe Zhang, Xinwei Feng, Xiangwen Liao, Wenbin Jiang, Yajuan Lyu, Kang Liu, and Jun Zhao. 2019b. Machine reading comprehension using structural knowledge graph-aware network. In *EMNLP*, pages 5895–5900.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge. *Trans. Assoc. Comput. Linguistics*, 7:249–266.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203.

Simon Suster and Walter Daelemans. 2018. Clicr: a dataset of clinical case reports for machine reading comprehension. In *NAACL-HLT*, pages 1551–1563.

Jian Wang, Hsuan-Chih Chen, Ralph Radach, and Albrecht Inhoff. 1999. *Reading Chinese script: A cognitive analysis*. Psychology Press.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *EMNLP*, pages 94–106.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*, pages 2346–2357.

Bishan Yang and Tom M. Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *ACL*, pages 1436–1446.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Fei Yuan, Linjun Shou, Xuanyu Bai, Ming Gong, Yaobo Liang, Nan Duan, Yan Fu, and Daxin Jiang. 2020. Enhancing answer boundary detection for multilingual machine reading comprehension. In *ACL*, pages 925–934.

Ningyu Zhang, Qianghuai Jia, Kangping Yin, Liang Dong, Feng Gao, and Nengwei Hua. 2020. Conceptualized representation learning for chinese biomedical text mining. *CoRR*, abs/2008.10813.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451.

Bo Zheng, Haoyang Wen, Yaobo Liang, Nan Duan, Wanxiang Che, Daxin Jiang, Ming Zhou, and Ting Liu. 2020. Document modeling with graph attention networks for multi-grained machine reading comprehension. In *ACL*, pages 6708–6718.

A Detailed Dataset Construction Process of CMedMRC

A.1 Medical Passage Curation

We use the following rules to obtain 20,000 passages from DXY as the inputs to human annotators:

- We use regular expressions to filter out images, tables, hyperlinks, etc. The English-Chinese translations of medical terms are also provided if the passages contain medical terms in English.
- We find that if we follow (Rajpurkar et al., 2016) to limit the lengths of the passages within 500 tokens, our human annotators could not ask 4 high-quality medical questions easily. Hence, our passage length limit is 1000 tokens.

A.2 Medical Question-Answer Pair Collection

We employ a group of annotators with professional medical background to generate question-answer pairs from the medical passages. Here are some general guidelines:

- We encourage our annotators to ask questions to which the answers are uniformly distributed in different positions of medical passages.
- For each medical passage, we limit the number of questions to 4.
- Each question should be strictly related to the medical domain. When creating the questions, no any part of the texts can be directly copied and pasted from the given medical passages.
- We limit the number of answer tokens to no more than 40.

A.3 Support Sentence Selection

In our dataset, we add an index to each sentence in the passages. Annotators are required to select the support sentence index and mark the range of the answer spans on the user interface.

A.4 Additional Answer Construction

To evaluate the human performance of our dataset and make our model more robust, we collect two additional answers for each question in the development and testing sets. We employ another 12 annotators for answer construction. Since our medical passage is relatively long, we show the questions and the passage contents again on the interface, together with the previously labeled support sentence indices.

Answer Type	Pct.	Example
Numeric	6%	20%
Time/Date	11%	1-3小时 (1-3 Hours)
Person	8%	儿童 (Child)
Location	5%	安徽, 云南, 湖北 (Anhui, Yunnan, Hubei)
Noun Phrase	18%	输卵管炎 (Salpingitis)
Verb Phrase	6%	清洗，干燥和粉碎 (Wash, dry and crush)
Yes/No	1%	不会感染 (Will not infect)
Description	44%	维生素缺乏 (Vitamin deficiency)
Other	1%	严重 (Severe)

Table 6: Statistical results for answer types.

	Train	Dev	Test
# Questions	12,700	3,630	1,823
Avg. tokens of passages	883.64	743.10	745.52
Avg. tokens of questions	15.40	14.85	15.23
Avg. tokens of answers	19.69	18.48	16.57
Avg. tokens of support sen.	57.50	48.19	42.70

Table 7: Statistical results of text length in our CMedMRC dataset.

B Statistical Analysis of the CMedMRC Dataset

B.1 Question and Answer Types

Due to the special characteristics of the Chinese language, the question types cannot be simply classified by prefix words of questions (Rajpurkar et al., 2016). Here, we manually define 8 common question types in the user annotation interface. The statistics of each question type are shown in Figure 4. The first seven question types usually correspond to special medical answers. For example, the *What* type refers to a question on the name of a drug or a disease, which accounts for more than half of the dataset. A third of the questions belong to the types of *How* and *Why*. The statistics of answer types are also shown in Table 6. The proportions of *Noun Phrase* and *Description* types

Error Type	Example	Percentage
Start position cross	***Ground-truth***: 当有急性炎症或者化脓时，会有剧烈疼痛；或者合并牙神经发炎时也会出现剧烈疼痛。 (*When there is acute inflammation or suppuration, there will be severe pain;* or when combined with dental nerve inflammation, there will also be severe pain) ***Prediction***: 有轻微的隐痛或胀痛；当有急性炎症或者化脓时，会有剧烈疼痛； (There is slight dull pain or pain; *when there is acute inflammation or suppuration, there will be severe pain;*)	25%
End position cross	***Ground-truth***: 以下人群高危：乙肝、丙肝病毒慢性感染者；患有类风湿关节、狼疮、硬皮病等免疫性疾病；吸烟。 ......遗传因素对本病起到一定作用。 (The following people are at high risk: people with chronic hepatitis B and C virus infections; suffering from immune diseases such as rheumatoid joints, lupus, and scleroderma; smoking. ...... *Genetic factors play a role in this disease.*) ***Prediction***: 目前认为血管炎是一种自身免疫性疾病，...... 遗传因素对本病起到一定作用。一些药物也可以引起血管炎，还有一些感染（如丙肝、乙肝、梅毒）也可以引起血管炎。 (*At present, vasculitis is considered to be an autoimmune disease, ... Genetic factors play a role in this disease.* Some drugs can also cause vasculitis, and some infections (such as hepatitis C, hepatitis B, and syphilis) can also cause vasculitis.)	21%
Answer substring	***Ground-truth***:这些药物具有抗炎、改善毛细血管通透性、减轻水肿、止痛等作用，同时对日光性皮炎有很好的治疗作用。 (These drugs have *anti-inflammatory, improve capillary permeability, reduce edema, pain relief, etc.*, and have a good therapeutic effect on solar dermatitis.) ***Prediction***:具有抗炎、改善毛细血管通透性、减轻水肿、止痛等作用 (*Anti-inflammatory, improve capillary permeability, reduce edema, relieve pain, etc.*)	46%
Other	***Ground-truth***:抗组胺药第一代的经典代表药「马来酸氯苯那敏」就是一个，它俗称扑尔敏，在多年临床应用中没有发现对胎儿有明显的致畸或其他严重危害。 (One of the classic representative drugs of the first generation of antihistamines is ”Chlorpheniramine Maleate”. It is commonly known as Chlorpheniramine. It has not been found to have obvious teratogenic or other serious harm to the fetus in many years of clinical application.) ***Prediction***:但临床上也有一些药物是经过多年验证，只要注意把握用药时间和药量，即使让孕妇吃也不会有事的 (However, there are also some drugs that have been verified for many years in clinical practice. As long as you pay attention to the time and amount of medication, it will be fine even if pregnant women take it.)	8%

Table 8: Three typical error answer types in support sentence task. The blue and underscore contents in brackets indicate why the sample belongs to its corresponding category.

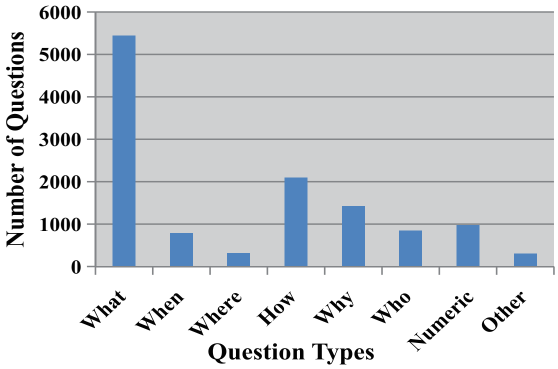

Figure 4: The number of questions that belong to each question types in CMedMRC.

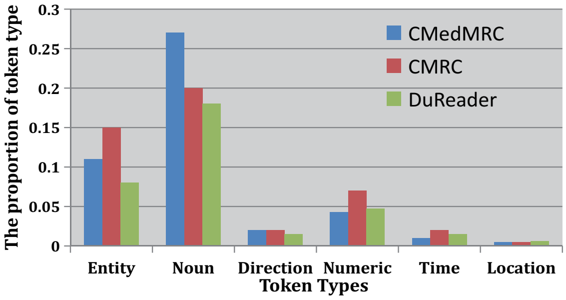

Figure 5: Proportions of entities and five frequently appearing POS tags in three Chinese MRC datasets.

are relatively large. The results are consistent with Figure 4, since most of *What* questions need to be answered with the above two answer types. Table 7 shows the text length of four input data.

B.2 Analysis of Domain Knowledge

We further analyze to what degree there exists domain knowledge in CMedMRC, in terms of medical entities and other terms. In this study, we employ the POS and NER toolkits[4] to tag medical entities and terms from 100 samples in the development set of CMedMRC. We also compare the statistics against those of two other Chinese MRC datasets, namely CMRC (Cui et al., 2019) and DuReader (He et al., 2018). The proportions of entities and five frequent POS tags in the three datasets are summarized in Figure 5. Comparing to the other two open-domain datasets, the proportion of entities in CMedMRC is very high (**11%**). In addition, the proportion of nouns (**27%**) is much higher than the other four POS tags in CMedMRC. The most likely cause is that existing models have difficulty recognizing all the medical terms, and treat them as common nouns. Among the three Chinese datasets, CMedMRC has the largest proportion (**38%**) of nouns and entities. Therefore, it is difficult for pre-trained language models to understand so many medical terms without additional medical background knowledge.

---

[4] We use jieba toolkit with additional medical term dictionaries. See `https://pypi.org/project/jieba/`.

C Experimental Settings

C.1 Medical Knowledge Base and Corpora

The underlying medical knowledge base is constructed by DXY, containing 44 relation types and over 4M relation triples. The KGs embedding trained by TransR (Lin et al., 2015b) on DXY-KG [5] containing 152,508 entities. In knowledge retrieval, the threshold of overlapped tokens is set to half of its own length. The medical pre-training corpora used in ERNIE-THU(Zhang et al., 2019) contains 5,937,695 text segments with 3,028,224,412 tokens (4.9 GB) after pre-processing.

C.2 Additional Training Details

In average, the training time for DrQA, BERT_base, MC-BERT, KT-NET, ERNIE, KMQA and CMedBERT takes 10, 16, 16, 27, 29, 28 and 25 minutes per epoch on a TiTAN RTX GPU. All the models are implemented by the PyTorch deep learning framework [6].

---

[5] `https://portal.dxy.cn/`

[6] https://pytorch.org/

Correcting Chinese Spelling Errors with Phonetic Pre-training

Ruiqing Zhang, Chao Pang, Chuanqiang Zhang, Shuohuan Wang,
Zhongjun He*, Yu Sun, Hua Wu and Haifeng Wang
Baidu Inc. No. 10, Shangdi 10th Street, Beijing, 100085, China
{zhangruiqing01, pangchao04, zhangchuanqiang, wangshuohuan}@baidu.com
{hezhongjun, sunyu02, wu_hua, wanghaifeng}@baidu.com

Abstract

Chinese spelling correction (CSC) is an important yet challenging task. Existing state-of-the-art methods either only use a pre-trained language model or incorporate phonological information as external knowledge. In this paper, we propose a novel end-to-end CSC model that integrates phonetic features into language model by leveraging the powerful pre-training and fine-tuning method. Instead of conventionally masking words with a special token in training language model, we replace words with phonetic features and their sound-alike words. We further propose an adaptive weighted objective to jointly train error detection and correction in a unified framework. Experimental results show that our model achieves significant improvements on SIGHAN datasets and outperforms the previous state-of-the-art methods.

1 Introduction

Spelling errors are common in practice and the errors will be enlarged in the downstream tasks. Therefore, Spelling correction is important to many NLP applications such as search optimization (Martins and Silva, 2004; Gao et al., 2010), machine translation (Belinkov and Bisk, 2017), part-of-speech tagging (Van Rooy and Schäfer, 2002; Sakaguchi et al., 2012), etc. Spelling correction requires a comprehensive grasp of word similarity, language modeling and reasoning, making it one of the most challenging tasks in NLP.

In this paper, we focus on Chinese spelling correction (CSC). Unlike alphabetic languages, Chinese characters cannot be typed without the help of input systems, such as Chinese Pinyin (a pronunciation-based input method) or automatic speech recognition (ASR). Thus typos of similarly pronounced characters occur quite often in Chinese text. According to Liu et al. (2010), 83% of Chinese spelling errors on the Internet results from phonologically similar characters. As illustrated in Figure 1, the character "德(de, German)" is incorrectly typed as one of its homophone[1] "的(de, of)".

Wrong:	ta 他 He	**de** 的 of	yu 语 language	shuo 说 speak	de 得	hen 很 very	hao 好 good
Correct:	ta 他 He	**de** 德 German	yu 语 language	shuo 说 speak	de 得	hen 很 very	hao 好 good
Predict without phonetic features:	ta 他 He	**ying** 英 English	yu 语 language	shuo 说 speak	de 得	hen 很 very	hao 好 good

Figure 1: An example of CSC. Each row contains three lines, Chinese pinyin, Chinese character, and gloss. The Chinese character "德(de, German) is incorrectly typed as its homophone "的(de, of)". The CSC model produces a fluent but incorrect sentence by replacing the character with "英(ying, English)", without considering the phonetic similarity.

Traditional methods of CSC firstly detect misspelled characters and generate candidates via a language model, and then use a phonetic model or rules to filter wrong candidates (Chang, 1995; Chen et al., 2013; Dong et al., 2016). To improve CSC performance, studies mainly focus on two issues: 1) how to improve the language model (Wu et al., 2010; Dong et al., 2016; Zhang et al., 2020) and 2) how to utilize external knowledge of phonological similarity (Jia et al., 2013; Yu and Li, 2014; Wang et al., 2018; Cheng et al., 2020). The language model is used to generate fluent sentences and the phonetic features can prevent the model from producing predictions whose pronunciation deviates from that of the original word. As illustrated in Fig. 1, the original *Wrong* sentence contains an incorrect word "的(de, of)". The CSC model produces a fluent but incorrect sentence by replacing "的(de,

---

* Corresponding author.

[1] In this paper, we ignore the tone of Pinyin, and use homophone to refer to characters with the same pinyin spellings.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2250–2261
August 1–6, 2021.

of)" with "英(ying, English)". However, the pronunciations of these two words are totally different, because the model ignores phonetic features.

Recent studies tackle the issue using deep neural networks. Hong et al. (2019) used a pre-trained language model BERT (Devlin et al., 2019) to generate candidates and train a classifier with phonetic features to select the final correction. Wang et al. (2019) considered CSC as a sequence-to-sequence task and generated candidates from a confusion set [2] instead of the entire vocabulary. These methods take phonetic information as external knowledge but the discrete candidate selection obstructs the language model from learning directly via backpropagation. Zhang et al. (2020) proposed an end-to-end CSC model by modifying the mask mechanism of BERT. However, they did not use any phonological information, which is important for exploring words similarity.

In this paper, we propose a novel end-to-end model for Chinese spelling correction. The model incorporates the phonetic information into language model and leverages the pre-training and fine-tuning framework. Concretely, we first modify the learning task of pre-trained masked language model (Devlin et al., 2019). Rather than replacing characters with an indiscriminate symbol "[MASK]", we mask characters with pinyin or similar pronounced characters. This enables the language model to explore the similarity between characters and pinyin. Then we fine-tune on error correction data with a model of two networks, a detection network predicts the probability of spelling error for each word, and a correction network generates correction by fusing the word embedding and pinyin embedding with the probabilities as input. We jointly optimize the detection and correction networks in a unified framework.

The contributions of this paper are summarized as follows:

- We propose a novel end-to-end CSC model that incorporates phonetic features into language representation. The model encodes the Chinese characters and Pinyin tokens in a shared space.

- The integration of phonological information greatly facilitates CSC. Experimental results on the benchmark SIGHAN datasets show that our method significantly outperforms the previous state-of-the-art methods.

2 Related work

Earier work on CSC follows the pipeline of error detection, candidate generation, and candidate selection (Wu et al., 2010; Jia et al., 2013; Chen et al., 2013; Chiu et al., 2013; Liu et al., 2013; Xin et al., 2014; Yu and Li, 2014; Dong et al., 2016; Wang et al., 2018). These methods mainly employ unsupervised language models and rules to select candidates.

With the development of end-to-end networks, some work proposed to optimize the error correction performance directly as a sequence-labeling task with conditional random fields (CRF) (Wu et al., 2018) and recurrent neural networks (RNN) (Zheng et al., 2016; Yang et al., 2017). Wang et al. (2019) used a sequence-to-sequence framework with copy mechanism to copy the correction results directly from a prepared confusion set for the erroneous words. Cheng et al. (2020) built a graph convolution network (GCN) on top of BERT (Devlin et al., 2019) and the graph was constructed from a confusion set. Zhang et al. (2020) proposed a soft-masked BERT model that first predicts the probability of spelling error for each word, and then uses the probabilities to perform a soft-masked word embedding for correction. However, they did not use any phonetic information.

Our work is most related to Zhang et al. (2020), but with some important differences. We will further discuss this in Section 3.4.

3 Methods

Formally, the Chinese spelling correction task is to map a sequence $\mathbf{x_w} = (x_{w_1}, x_{w_2}, ..., x_{w_N})$ which may contain spelling errors to another correct sequence $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_N)$, where both x_{w_i} and $\hat{y}_i$ $(1 \leq i \leq N)$ are Chinese characters.

We propose an end-to-end CSC model which consists of two components, detection and correction. The detection module takes $\mathbf{x_w}$ as input and predicts the probability of spelling error for each character. The correction model takes the combination of the embedding of $\mathbf{x_w}$ and its corresponding pinyin sequence $\mathbf{x_p} = (x_{p_1}, x_{p_2}, ..., x_{p_N})$ as input and predicts the correct sequence $\mathbf{y}$. We propose a method to fuse $\mathbf{x_w}$ and $\mathbf{x_p}$ embeddings using the probability of spelling error as weights.

Following the pre-train and fine-tune framework,

---

[2] Confusion set is a set of similar characters.

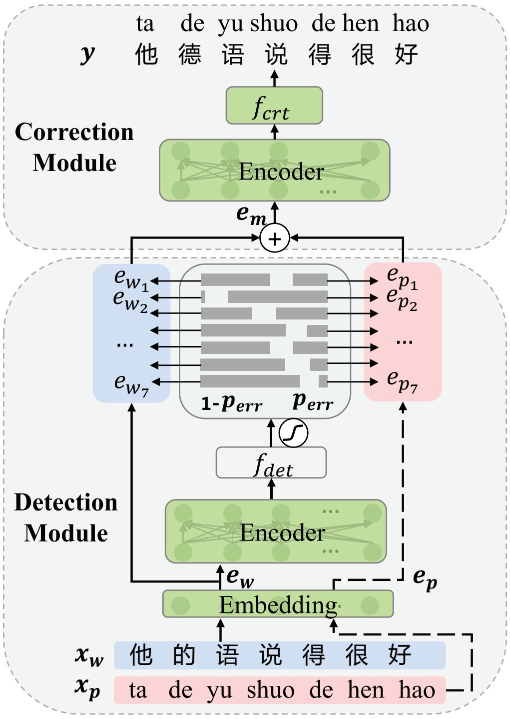

Figure 2: Illustration of our CSC model. Given the input word sequence $\mathbf{x_w}$, the detection module first predicts the probability of error $\mathbf{p_{err}}$ for all the characters. Then the correction module combines the word embedding $\mathbf{e_w}$ and pinyin embedding $\mathbf{e_p}$ to an embedding fusion $\mathbf{e_m}$ and send it to generate the final correction $\mathbf{y}$. The parameters of the Embedding, Encoder, and f_{crt} are initialized by a pre-trained language model with phonetic features. The structure and parameters of the two encoders are identical.

we first pre-train a masked language model, **MLM-phonetics**, by learning to predict characters from similarly pronounced characters and pinyin. Then in fine-tuning, we jointly optimize the the detection and correction modules.

In this section, we first introduce the model architecture (Sec. 3.1), the optimization method (Sec. 3.2) , and the pre-training of **MLM-phonetics** (Sec. 3.3), then summarize the novelty of our method (Sec. 3.4).

3.1 Model Architecture

Fig. 2 shows our model architecture, the lower is the error detection module and the upper is the correction module. Both are built upon transformer.

Detection Module Given a source sequence $\mathbf{x_w} = (x_{w_1}, x_{w_2}, ..., x_{w_N})$, the goal of the detection module is to check whether a character $x_{w_i} (1 \leq i \leq N)$ is correct or not. For this labelling problem, we use class 1 and 0 to label misspelled characters and correct characters, respectively.

We formalize the detection module as follows:

$$\mathbf{y_d} = \text{softmax}(f_{det}(E(\mathbf{e_w}))) \tag{1}$$

where $\mathbf{e_w} = (e_{w_1}, e_{w_2}, ..., e_{w_N})$ is the word embedding of $\mathbf{x_w}$, E is a pre-trained encoder and f_{det} is a fully-connected layer that maps the sentence representation to a binary sequence $\mathbf{y_d} = (y_{d_1}, y_{d_2}, ..., y_{d_N}), y_{d_i} \in \{0, 1\}$.

We use p_{err_i} to denote the probability that character x_{w_i} is erroneous:

$$p_{err_i} = p(y_{d_i} = 1 | \mathbf{x_w}; \theta_d) \tag{2}$$

where θ_d is the parameters of error detection module.

Correction Module The goal of the correction module is to generate correct characters based on the output of the detection module.

We not only use the word embeddings for input, but also use the pinyin embeddings to integrate the phonetic information. Concretely, we first generate the pinyin sequence $\mathbf{x_p}$ using the PyPinyin[3] tool, get the pinyin embedding $\mathbf{e_p}$ from the embedding layer, and fuse it with the word embedding $\mathbf{e_w}$ by linear combination:

$$\mathbf{e_m} = (1 - \mathbf{p_{err}}) \cdot \mathbf{e_w} + \mathbf{p_{err}} \cdot \mathbf{e_p} \tag{3}$$

This combination uses the spelling error probability predicted by the detection module as weights to balance the importance of the semantic feature (character embedding) and phonetic feature (pinyin embedding). We introduce two special cases: If $p_{err_i} = 0$, indicating the character x_{w_i} is detected to be correct, and the model uses only its word embedding in $\mathbf{e_m}$. If $p_{err_i} = 1$, meaning that the character is detected to be erroneous, and the model uses its pinyin embedding.

Finally, the correction results $\mathbf{y}$ is predicted through a fully-conntected layer f_{crt}:

$$\mathbf{y} = softmax(f_{crt}(E(\mathbf{e_m}))) \tag{4}$$

Note that, the parameters of the embedding, the encoder E and the correction network f_{crt} are initialized by **MLM-phonetics**. In the pre-training, **MLM-phonetics** is trained to reconstruct the correct characters from their commonly confused counterparts and their pinyin, thus it can be transformed with the fused embedding.

---

[3] https://pypi.org/project/pypinyin/

3.2 Jointly Fine-tuning

There are two objectives for our model: to train the detection parameters f_{det} and to adjust the detection and correction modules to achieve an optimal balance. We jointly optimize the detection loss $\mathcal{L}_d$ and the correction loss $\mathcal{L}_c$ that:

$$\mathcal{L}_d = \text{-} \sum_i \log p(\hat{y}_{d_i} | \mathbf{x_w}; \theta_d) \quad (5)$$

$$\mathcal{L}_c = \text{-} \sum_i p(y_{d_i} | \mathbf{x_w}; \theta_d) \cdot \log p(\hat{y}_i | \mathbf{e_m}; \theta_c) \quad (6)$$

where θ_d and θ_c is the parameter of the detection and correction module, respectively. $\hat{y}_{d_i}$ is the ground-truth detection result and y_{d_i} is the prediction by the detection module, both of them is a binary value of 0 or 1.

In particular, the correction loss is the negative log likelihood weighted by the probability of the detection result, $p(y_{d_i} | \mathbf{x_w}; \theta_d) \in (0.5, 1]$. This is to distinguish between the responsibilities of the two tasks. When the detection module gives a low-confidence prediction, that is, $p(y_{d_i} | \mathbf{x_w}; \theta_d)$ approaches 0.5, $\mathbf{e_m}$ fuses the semantic features and phonetic features with similar weights. But we hope that the detection module could provide clear judgement of right or wrong, i.e., $p(y_{d_i} | \mathbf{x_w}; \theta_d)$ approaches to 1, so that $\mathbf{e_m}$ can be dominated by either semantic features or phonetic features. In such case, the correction of error words will not be interfered by the semantic features in $\mathbf{e_m}$, and vice versa. Therefore, we penalize the low-confidence prediction given by the detection module. Concretely, when the probability of the detection result is low, $\mathcal{L}_c$ decreases and the model will focus more on optimizing $\mathcal{L}_d$. And when the detection probability is high, the model optimizes $\mathcal{L}_d$ and $\mathcal{L}_c$ at balance.

The adaptive weighting objective enables us to jointly train our model with the sum of the two loss functions:

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_c \quad (7)$$

We compare different weighting strategies with our adaptive weighting in experiments.

3.3 Pre-training MLM-phonetics

In this section, we introduce our pre-trained language model, **MLM-phonetics**, that 1) integrates phonetic features and 2) solves the problems of using standard masked language model in our CSC architecture.

The pre-train and fine-tune framework (Devlin et al., 2019) has been proven effective in facilitating downstream NLP tasks including sentence classification, question answering, etc. But the inputs of these tasks are of identical distribution with pre-training, while the input sentences in CSC are with errors, which are different from the pre-training samples. Some work thus far side-stepped the input divergence by avoiding to input error sentences directly to pre-trained models. For example, Zhang et al. (2020) use a bidirectional-GRU for error detection before a BERT-based correction network.

In order to take advantage of the pre-training technique, we modify the pre-training task. In the pre-training of a standard mask language model (**MLM-base**), the model is trained by predicting 15% randomly selected characters which are replaced with the [MASK] token, random character, and themselves at the sampling rate of 80%, 10%, and 10%, respectively.

To avoid input divergence and integrate phonetic features, we propose two pre-training replacements: *confused-Hanzi*[4] and a *noisy-pinyin*. We use Figure 3 to illustrate these replacements:

- [MASK] replacement trains the reasoning ability of the language model by restoring masked characters only according to the context.

- *Random Hanzi* replacement trains **MLM-base** to correct words from random ones (e.g, to predict “得(de)” from “不(bu)”), which is a more difficult task compared with correcting from similarly pronounced characters. However, due to the different input distribution, this strategy is of little help to CSC.

- *Same* replacement encourages **MLM-base** to copy the input characters (e.g, the replacement “好(hao)”).

- *Confused-Hanzi* replacement trains **MLM-phonetics** to correct words to their commonly confused characters in the confusion set (e.g, to predict “豪(hao)” to “好(hao)”). It provides the model a way to access samples with typo.

- *Noisy-pinyin* replacement trains **MLM-phonetics** to predict the original characters from pinyin of their commonly confused characters in the confusion set (e.g, to predict

---

[4] *Hanzi* is a transliteration meaning Chinese characters.

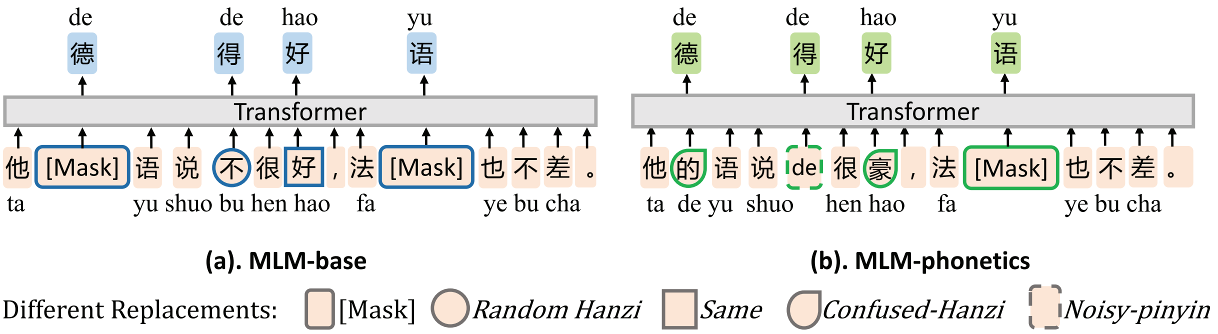

(a). MLM-base **(b). MLM-phonetics**

Figure 3: An example of the different replacement strategy for **MLM-base** and **MLM-phonetics**.

"得(de)" from "de"). It helps clustering similarly pronounced characters with their corresponding pinyin tokens.

The first three replacements are used for pre-training standard **MLM-base** and the last two are proposed in our method to model the similarity between characters and pinyin tokens. In the pre-training of **MLM-phonetics**, our data generator randomly chooses 20% of token positions in the training samples. If the i^{th} token is chosen, we empirically replace it with (1) the [MASK] token 40% of the time, (2) the *noisy-pinyin* of this token 30% of the time, and (3) a *confused-Hanzi* from its confusion set 30% of the time[5]. Then **MLM-phonetics** is trained to predict the original sentence from the sentence with replacements.

The two proposed pre-training tasks can smooth out the input divergence between pre-training and fine-tuning the CSC model. The *Confused-Hanzi* replacement simulates the input of the detection module and the two replacements together facilitates the pre-trained model to adapt to the fused embedding (Eq. 3).

3.4 Novelty of our method

Our method is most related to Zhang et al. (2020), but different in the following aspects.

First, our model combine the embedding of pinyin and character to prevent information loss, which is more like the human correction process that predicts correction with the pronunciation of the problematic words. On the contrary, Zhang et al. (2020) has to add residual connection before emitting the final correction, or it will forget the phonetic information of the error words after combining their embedding with [MASK].

Second, we share the pre-trained encoder in detection and correction by proposing new pre-training tasks, while Zhang et al. (2020) took an un-pre-trained bidirectional-GRU in detection to avoid the input divergence between pre-training and fine-tuning.

Third, we propose an adaptive weighting policy in jointly training the error detection and correction. This policy encourages the model to produce clear detection results, making the fused embedding dominated by either semantic features or phonetic features, which is close to the pre-training task. On the contrary, Zhang et al. (2020) proposed to linearly combine the detection and correction loss with a fixed hyper-parameter.

4 Experiments

We carry out experiments on the SIGHAN dataset, a benchmark for CSC.

4.1 Data Processing

The training set consists of two parts: 1) A pre-training corpus of 0.3 billion Chinese sentences, and 2) A CSC training corpus of 281K sentences pairs. The first corpus is used for pre-training **MLM-phonetics**, and the latter is used to fine-tune the CSC model initialized by **MLM-phonetics**.

For the pre-training corpus, we collect a variety of data, such as encyclopedia articles, news, scientific papers, and movie subtitles from a search engine. The CSC training data used in our experiments is the same as Wang et al. (2019) and Cheng et al. (2020), including three human-annotated training datasets (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015) and an automatically generated dataset with the approach proposed in Wang et al. (2018)[6].

---

[5] For multiple characters in the confusion set, select them/their pinyin randomly in the replacement.

[6] See Appendix A for training corpus detail.

	Detection			**Correction**		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
SIGHAN13						
FASPell (2019)	76.2	63.2	69.1	73.1	60.5	66.2
Pointer Networks (2019) (character-level)	56.8	91.4	70.1	79.7	59.4	68.1
*Soft-Masked BERT**	81.1	75.7	78.3	75.1	70.1	72.5
SpellGCN (2020)	80.1	74.4	77.2	78.3	72.7	75.4
ERNIE	76.6	71.9	74.2	73.0	68.5	70.6
MLM-phonetics(Ours)	**82.0**	**78.3**	**80.1**	**79.5**	**77.0**	**78.2**
SIGHAN14						
FASPell (2019)	61.0	53.5	57.0	59.4	52.0	55.4
Pointer Networks (2019) (character-level)	63.2	82.5	71.6	79.3	68.9	73.7
*Soft-Masked BERT**	65.2	70.4	67.7	63.7	68.7	66.1
SpellGCN (2020)	65.1	69.5	67.2	63.1	67.2	65.3
ERNIE	63.5	69.3	66.3	60.1	65.6	62.8
MLM-phonetics(Ours)	**66.2**	**73.8**	**69.8**	**64.2**	**73.8**	**68.7**
SIGHAN15						
FASPell (2019)	67.6	60.0	63.5	66.6	59.1	62.6
Pointer Networks (2019) (character-level)	66.8	73.1	69.8	71.5	59.5	64.9
Soft-Masked BERT (2020)	73.7	73.2	73.5	66.7	66.2	66.4
*Soft-Masked BERT**	67.6	78.7	72.7	63.4	73.9	68.3
SpellGCN (2020)	74.8	80.7	77.7	72.1	77.7	75.9
ERNIE	73.6	79.8	76.6	68.6	74.4	71.4
MLM-phonetics(Ours)	**77.5**	**83.1**	**80.2**	**74.9**	**80.2**	**77.5**

Table 1: The performance on SIGHAN13, SIGHAN14, and SIGHAN15 testset. *Soft-Masked BERT** is our reproduction of *Soft-Masked BERT* using the same training data as in our method, while *Soft-Masked BERT* was trained on an in-house dataset containing 5 million sentences and their counterparts with automatically generated errors, as reported in Zhang et al. (2020), where the authors only provided their results on SIGHAN15.

4.2 Model Settings

We compare our methods with previous state-of-the-art methods:

- *FASPell* (Hong et al., 2019) first generates candidates for each character in the input sentence through a pre-trained MLM, then uses a filtering model with visual and phonetic similarity features to select the best candidate.

- *Pointer Networks* (Wang et al., 2019) uses a seq2seq system based on the constraint that each correct word is contained in the confusion set of the erroneous character.

- *Soft-Masked BERT* (Zhang et al., 2020), for each token in the sentence, linearly combines its embedding with the embedding of [MASK], and predicts the error character based on a fine-tuned masked language model.

- *SpellGCN* (Cheng et al., 2020) incorporates two similarity graphs into a pre-trained sequence-labeling model via graph convolutional network. The two graphs are derived from a confusion set and correspond to pronunciation and shape similarities.

- *ERNIE* (Sun et al., 2020) directly finetunes the standard masked language model on the CSC training data.

- **MLM-phonetics**, our proposed method uses an end-to-end system based on a pre-trained language model with phonetic features.

Pointer Network uses LSTM in both encoder and decoder. All the other methods take CSC as a sequence tagging problem with a pre-trained 12-layer Transformer as encoder. *FASPell* and *Soft-Masked BERT* use the pre-trained BERT, while *ERNIE* and **MLM-phonetics** use the pre-trained ERNIE for initialization[7]. We use the sentence-level and character-level f1-score to evaluate different systems. At the sentence-level, a prediction is considered correct only if all the errors in the sentence are detected or corrected. Therefore, sentence-level evaluation is stricter and results in lower scores. Following Cheng et al. (2020), we

---

[7] The difference between BERT and ERNIE in fine-tuning is trivial, see Appendix B for detail.

use the scripts in (Hong et al., 2019) to calculate the sentence-level results.

4.3 Overall Results

Table 1 shows the detection and correction performance on three SIGHAN test sets. All the methods provide sentence-level results except *Pointer Network*, which provides the results at the character-level.

It shows that our method, **MLM-phonetics** significantly outperforms the other systems. For example on SIGHAN15, the detection F1-score has 2.5 point improvement (77.7→80.2) and the correction F1-score has 1.6 point improvement (75.9→77.5) compared with the previous best method *SpellGCN*. Our method also achieves over 6 points improvement over *ERNIE* in correction f1-score, verifying the effectiveness of our pre-training strategy.

All the listed methods except *Pointer Networks* use pre-trained model for initialization, but only *FASPell*, *SpellGCN* and our method take phonetic information into consideration. *FASPell* uses isolated phonetic features and language model, which inevitably lead to performance decline. *SpellGCN* incorporates phonetic knowledge into the language model by building a graph convolutional network on the top of BERT. It is proven to be effective, but the graph is derived from a prepared confusion set. Therefore, the performance of the model depends on the completeness of this set. As shown in Table 1, the precision of *SpellGCN* is close to **MLM-phonetics**, but there is a significant gap in the recall. Our method, with the help of additional Pinyin tokens, integrates phonetic features in word embedding, thus increasing the generalization of the model. *Soft-Masked BERT* corrects sentences without phonetic features. Its detection and correction performance is inferior to ours. This may partly due to the lack of phonological similarity, as well as the difference in model architecture. It is also notable that the training data of our method *MLM-phonetics* is consistent with that of *Pointer Networks*, *Soft-Masked BERT**, *SpellGCN* and *ERNIE*, but *Soft-Masked BERT* and *FASPell* use different training data. See Appendix C for some case studies.

4.4 Pre-training Tasks

In order to analyze the effect of the three replacement tasks ([MASK], *Confused-Hanzi*, *Noisy-pinyin*) in the pre-training of **MLM-phonetics**, we take three models for comparison, each trained by only two of the three tasks with equal probability.

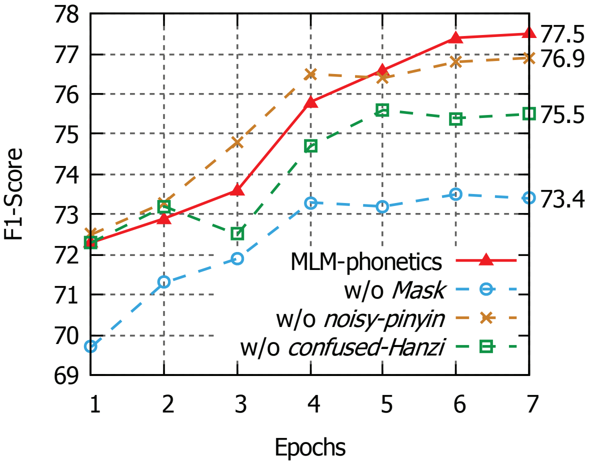

Figure 4: The Sentence-level Correction F1-score of 4 pre-trained models evaluated on SIGHAN15 testset.

During fine-tuning, the testing curves on SIGHAN15 are plotted in Figure 4. **MLM-phonetics** shows the best performance, achieving the correction f1-score of 77.5 in the 7^{th} epoch. However, it's interesting that its performance is inferior to the pre-trained model without (*w/o*) *noisy-pinyin* at the beginning. This is caused by the pre-training & fine-tuning discrepancy.

The model *w/o noisy-pinyin* only learns to predict the original characters from [MASK] and confused-Hanzi in pre-training. So the pinyin embedding has not been initialized until fine-tuning. Therefore, the pinyin embedding can be viewed as noise in the embedding fusion of the fine-tuning stage. Such embedding is close to its pre-training input distribution, thus the pre-trained model *w/o noisy-pinyin* performs good at the beginning. **MLM-phonetics**, on the contrary, is trained to reconstruct words based on either Hanzi embedding or pinyin embedding in the pre-training. But it needs to predict from a fusion of them in fine-tuning, thus it requires longer training time for adaption. As the training continues, the model benefits from the embedding fusion and finally achieves 0.6 points improvements (76.9→77.5) over the pre-trained model *w/o noisy-pinyin*.

Besides, the other two pre-trained models perform relatively low. The pre-trained model *w/o confused-Hanzi* suffers from input divergence in pre-training & fine-tuning. The model is not trained to correct words from spelling errors until the fine-tuning stage. The pre-trained model *w/o* [MASK] performs the worst, which shows the importance of

Objective		**Detection**			**Correction**		
		Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
fixed hyperparameter:	$\lambda = 0.2$	75.6	79.1	77.3	71.7	77.5	74.5
$\lambda\mathcal{L}_{uc} + (1-\lambda)\mathcal{L}_d$	$\lambda = 0.5$	77.1	82.0	79.5	74.1	78.4	76.2
(Zhang et al., 2020)	$\lambda = 0.8$	76.1	81.2	78.6	**75.3**	77.9	76.6
Ours: $\mathcal{L}_c + \mathcal{L}_d$		**77.5**	**83.1**	**80.2**	74.9	**80.2**	**77.5**

Table 2: Sentence-level performance of using different objectives on SIGHAN15.

using [MASK] prediction to enhance the semantic comprehension.

4.5 Balance the objective of detection and correction

Next, we explore the impact of the weighting strategy that balances the two objectives in fine-tuning. In our CSC model, both the detection and correction are sequence labeling tasks. We use the detection probability to balance the two tasks, as depicted in Eq.(6). On the contrary, Zhang et al. (2020) balances the two tasks with a fixed hyper-parameter λ: $\lambda\mathcal{L}_{uc}+(1-\lambda)\mathcal{L}_d$, in which $\mathcal{L}_{uc}$ is the un-weighted negative log-likelihood of correction:

$$\mathcal{L}_{uc} = -\sum_i \log p(y_i|\mathbf{e_m};\theta_c) \tag{8}$$

The results of the two strategies are shown in Table 2. Our method is generally better than the results of using a fixed hyper-parameter for combination. Among the three systems with fixed hyper-parameters, the system with $\lambda = 0.8$ achieves the highest correction f1-score and the one with $\lambda = 0.5$ achieves the best detection f1-score. Note that the detection F1-score is evaluated based on the correction result (i.e., only the corrected characters are regarded as detected), rather than based on the prediction of the detection module. Therefore, it's not weird to find that the setting $\lambda = 0.2$ costs much on detection but its detection f1-score is the worst. This also provides us a hint that the detection and correction need to be coordinated. Setting λ to 0.2 may improve the performance of the detection module, but a poor correction module will bring down the final detection performance.

Our method, on the contrary, balances the $\mathcal{L}_c$ and $\mathcal{L}_d$ according to the confidence given by the detection module dynamically and achieves the best performance. Compared with the fixed hyper-parameter strategy with $\lambda = 0.8$, our F1 scores have 1.6 points (78.6→80.2) improvement in detection and 0.9 points (76.6→77.5) improvement in correction, indicating the effectiveness of our dynamic balance strategy in alleviating the unbalanced problem between the two tasks.

4.6 Error Analysis

To analyze the prediction errors, we collect the incorrectly predicted samples and classify them into two classes:

- *Detection Error*: the detection module produces an error prediction, i.e. $y_{d_i} \neq \hat{y}_{d_i}$.
- *Correction Error*: the detection module generates a correct prediction, but the correction module fails to generate the right character, i.e., $y_{d_i} = \hat{y}_{d_i}$ and $y_i \neq \hat{y}_i$.

We summarize the two classes on the SIGHAN15 testset and the proportion of the *Detection Error* and *Correction Error* is 83.6% and 16.4%, respectively. This reveals that most of the false predictions are *Detection Errors*.

We further explore the reason behinds the poor detection performance. Is it mainly because many errors cannot be detected (*false negative* errors), or does the detection module make incorrect predictions of errors (*false positive* errors)? We decompose the 83.6% detection errors into the two types and find that *false negative* errors and *false positive* errors account for 41.1% and 42.5% respectively. The proportions of the two error types are almost equal. A possible reason is that some homonyms are indistinguishable, such as “的”, “地”, and “得”. All of the three characters have the pronunciation of “de” and it makes sense to use any of these candidates in many sentences, phonetically or semantically. This problem has also been proposed in Cheng et al. (2020), which takes further fine-tuning to reduce the indistinguishability. In this case, the detection module produces many predictions that are different from the ground-truth results, affecting the detection performance.

5 Conclusion

In this paper, we propose a novel end-to-end framework for CSC with phonetic pre-training. Inspired by traditional pipeline systems, the model incorporates phonetic information of characters into pre-training. We first pre-train a masked language model with phonetic features to improve the model's ability to understand sentences with misspelling and model the similarity between characters and pinyin tokens. Further, we propose an end-to-end framework to integrate detection and correction in one model. Experiments on a benchmark dataset show that our model significantly outperforms the previous state-of-the-art. The CSC model with phonetic features can be used to reduce errors for speech recognition and translation systems. In the future, we are going to apply CSC to more challenging scenarios, such as streaming ASR error correction for automatic simultaneous translation, as well as variable-length correction.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments, and our colleague, Chen Ying, for her advice on writing.

References

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Chao-Huang Chang. 1995. A new approach for automatic chinese spelling correction. In *Proceedings of Natural Language Processing Pacific Rim Symposium*, volume 95, pages 278–283. Citeseer.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A study of language modeling for Chinese spelling check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83, Nagoya, Japan. Asian Federation of Natural Language Processing.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. SpellGCN: Incorporating phonological and visual similarities into language models for Chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 871–881, Online. Association for Computational Linguistics.

Hsun-wen Chiu, Jian-cheng Wu, and Jason S. Chang. 2013. Chinese spelling checker based on statistical machine translation. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 49–53, Nagoya, Japan. Asian Federation of Natural Language Processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shichao Dong, Gabriel Pui Cheong Fung, Binyang Li, Baolin Peng, Ming Liao, Jia Zhu, and Kam-fai Wong. 2016. ACE: Automatic colloquialism, typographical and orthographic errors detection for Chinese language. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 194–197, Osaka, Japan. The COLING 2016 Organizing Committee.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366, Beijing, China. Coling 2010 Organizing Committee.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. FASPell: A fast, adaptable, simple, powerful Chinese spell checker based on DAE-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169, Hong Kong, China. Association for Computational Linguistics.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph model for Chinese spell checking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 88–92, Nagoya, Japan. Asian Federation of Natural Language Processing.

Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. 2010. Visually and phonologically similar characters in incorrect simplified Chinese words. In *Coling 2010: Posters*, pages 739–747, Beijing, China. Coling 2010 Organizing Committee.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid Chinese spelling correction using language model and statistical machine translation with reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan. Asian Federation of Natural Language Processing.

Bruno Martins and Mário J Silva. 2004. Spelling correction for search engine queries. In *International Conference on Natural Language Processing (in Spain)*, pages 372–383. Springer.

Keisuke Sakaguchi, Tomoya Mizumoto, Mamoru Komachi, and Yuji Matsumoto. 2012. Joint English spelling error correction and POS tagging for language learners writing. In *Proceedings of COLING 2012*, pages 2357–2374, Mumbai, India. The COLING 2012 Organizing Committee.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to SIGHAN 2015 bake-off for Chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37, Beijing, China. Association for Computational Linguistics.

Bertus Van Rooy and Lande Schäfer. 2002. The effect of learner errors on pos tag errors during automatic pos tagging. *Southern African Linguistics and Applied Language Studies*, 20(4):325–335.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for Chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527, Brussels, Belgium. Association for Computational Linguistics.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for Chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785, Florence, Italy. Association for Computational Linguistics.

Shih-Hung Wu, Yong-Zhi Chen, Ping-che Yang, Tsun Ku, and Chao-Lin Liu. 2010. Reducing the false alarm rate of Chinese character error detection and correction. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42, Nagoya, Japan. Asian Federation of Natural Language Processing.

Shih-Hung Wu, Jun-Wei Wang, Liang-Pu Chen, and Ping-Che Yang. 2018. CYUT-III team Chinese grammatical error diagnosis system report in NLPTEA-2018 CGED shared task. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 199–202, Melbourne, Australia. Association for Computational Linguistics.

Yang Xin, Hai Zhao, Yuzhu Wang, and Zhongye Jia. 2014. An improved graph model for Chinese spell checking. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 157–166, Wuhan, China. Association for Computational Linguistics.

Yi Yang, Pengjun Xie, Jun Tao, Guangwei Xu, Linlin Li, and Luo Si. 2017. Alibaba at IJCNLP-2017 task 1: Embedding grammatical features into LSTMs for Chinese grammatical error diagnosis task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 41–46, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223, Wuhan, China. Association for Computational Linguistics.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of SIGHAN 2014 bake-off for Chinese spelling check. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132, Wuhan, China. Association for Computational Linguistics.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 882–890, Online. Association for Computational Linguistics.

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 49–56, Osaka, Japan. The COLING 2016 Organizing Committee.

A Datasets

All of our used datasets are listed in Table 3. The Pre-training corpus includes 0.3 billion sentences and the remaining four corpora contain 281K <error, correct> sentence pairs in sum[8]. The three SIGHAN datasets are human-annotated and the (Wang et al., 2018) is generated automatically with ASR and OCR technique.

Datasets		#Lines	Avg. Length	#Errors
Training Corpus				
	Pre-training corpus	0.3B	29.37	
	Wang et al. (2018)	271,329	44.4	382,704
	SIGHAN 2013	350	49.2	350
	SIGHAN 2014	6,526	49.7	10,087
	SIGHAN 2015	3,174	30	4,237
	Total	281,379	44.4	397,378
Test sets				
	SIGHAN 2013	1000	74.1	1227
	SIGHAN 2014	1062	50.1	782
	SIGHAN 2015	1100	30.5	715

Table 3: Experimental Data Statistics Information.

B Difference between BERT and ERNIE

We evaluate the performance of two pre-trained models, BERT (Devlin et al., 2019) and ERNIE (Sun et al., 2020), on the SIGHAN testset. For both of them, we use the released model[9] of the base version (12 layers with the hidden size of 768).

The zero-shot performance is listed in Table 4. In this setup, we directly use the released model for error correction without fine-tuning. It shows that ERNIE has a prominent advantage over BERT in both detection and correction. This is caused by a prediction problem of BERT that for most of the time, BERT corrects the first character to be the period symbol (“。”). We guess that this is because of a Chinese data pre-processing bug of BERT, that is, when a paragraph is divided into multiple sentences, it always divides the ending period of a sentence into the beginning of the next one. Therefore, a lot of sentences incorrectly predicts the beginning character to be the period symbol.

Then we finetune the 281K CSC training data on the two pretrained models. Table 5 shows the performance of the two models is basically the same. The difference between BERT and ERNIE is +0.4, -2.0, and +1.6 on SIGHAN13, SIGHAN14, and SIGHAN15, respectively. Therefore the difference between BERT and ERNIE after fine-tuning is trivial.

	Detection			Correction		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
SIGHAN13						
BERT	4.73	4.74	4.74	3.2	3.2	3.19
ERNIE	56.0	40.9	47.3	33.3	24.3	28.1
SIGHAN14						
BERT	4.7	8.8	6.1	2.6	4.9	3.4
ERNIE	63.5	27.2	31.1	18.8	14.1	16.1
SIGHAN15						
BERT	9.4	15.6	11.7	6.3	10.4	7.8
ERNIE	55.8	33.9	42.2	30.8	18.7	23.3

Table 4: The performance of Zero-Shot BERT and ERNIE.

	Detection			Correction		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
SIGHAN13						
BERT	73.3	70.4	71.8	71.7	68.9	70.2
ERNIE	76.6	71.9	74.2	73.0	68.5	70.6
SIGHAN14						
BERT	63.3	71.0	67.0	61.3	68.7	64.8
ERNIE	63.5	69.3	66.3	60.1	65.6	62.8
SIGHAN15						
BERT	68.3	77.8	72.7	65.5	74.6	69.8
ERNIE	73.6	79.8	76.6	68.6	74.4	71.4

Table 5: The performance of pre-trained models with fine-tuning.

C Ablation Study

We compare our method with *ERNIE* and *Soft-Masked BERT* trained on identical datasets shown in Table 3. Table 6 and Table 7 show that *MLM-phonetics* performs better at generating semantically coherent and similar sounding corrections.

---

[8] The 281K sentence pairs can be downloaded at https://github.com/ACL2020SpellGCN/SpellGCN/tree/master/data/merged.

[9] The released model of ERNIE: https://github.com/PaddlePaddle/ERNIE. The released model of BERT: https://github.com/google-research/bert

Input	ta	men	de	chao	fan	hen	bu	cuo	zai	shuo	ta	men	zuo	de	ga	li	ji	ye	hao	chi
	他	们	的	吵	翻	很	不	错，	再	说	他	们	做	的	咖	喱	鸡	也	好	吃！
	Their			quarrel		is very good,			and the		curry chicken they make							is also delicious!		
ERNIE	ta	men	de	kao	can	hen	bu	cuo	zai	shuo	ta	men	zuo	de	ga	li	ji	ye	hao	chi
	他	们	的	烤	餐	很	不	错，	再	说	他	们	做	的	咖	喱	鸡	也	好	吃！
	Their			roast meal		is very good,			and the		curry chicken they make							is also delicious!		
Soft-Masked BERT*	ta	men	de	chao	fa	hen	bu	cuo	zai	shuo	ta	men	zuo	de	ga	li	ji	ye	hao	chi
	他	们	的	炒	法	很	不	错，	再	说	他	们	做	的	咖	喱	鸡	也	好	吃！
	Their			fried method		is very good,			and the		curry chicken they make							is also delicious!		
***MLM-phonetics* (ours)**	ta	men	de	chao	fan	hen	bu	cuo	zai	shuo	ta	men	zuo	de	ga	li	ji	ye	hao	chi
	他	们	的	炒	饭	很	不	错，	再	说	他	们	做	的	咖	喱	鸡	也	好	吃！
	Their			fried rice		is very good,			and the		curry chicken they make							is also delicious!		

Table 6: An example from SIGHAN15 test set. Errors are marked in red and right corrections are in blue. All of the three methods accurately detect the misspelling words, but only *MLM-phonetics* yields the correct result. *ERNIE* changes “吵翻” (“chao fan”) to differently pronounced “烤餐” (“kao can”) and *Soft-Masked BERT** changes “吵翻” to “炒法” (“chao fa”, fried method), which sounds similar but is not so good as “炒饭” (fried rice) in terms of semantic coherence.

Input	ying	xiang	xiao	hai	zi	xiang	de	kuai		xue	xi	guan	li	de	ban	fa
	影	像	小	孩	子	想	得	快	,	学	习	管	理	的	斑	法
	Image		children			think fast,				learn		management			spot	method
ERNIE	ying	xiang	xiao	hai	zi	xiang	de	kuai		xue	xi	guan	li	de	fang	fa
	影	响	小	孩	子	想	得	快	,	学	习	管	理	的	方	法
	Influence		children			think fast,				learn		management			method	
Soft-Masked BERT*	ying	xiang	xiao	hai	zi	xiang	de	kuai		xue	xi	guan	li	de	fang	fa
	影	像	小	孩	子	想	得	快	,	学	习	管	理	的	方	法
	Image		children			think fast,				learn		management			method	
***MLM-phonetics* (ours)**	ying	xiang	xiao	hai	zi	xiang	de	kuai		xue	xi	guan	li	de	ban	fa
	影	响	小	孩	子	想	得	快	,	学	习	管	理	的	办	法
	Influence		children			think fast,				learn		management			method	

Table 7: Another example from the SIGHAN15 test set. Again, *MLM-phonetics* predicts phonetically similar and semantically coherent correction. But both *ERNIE* and *Soft-Masked BERT** replace “斑”(ban) with “方”(fang), which is semantically coherent but sounds greatly different.

Multi-Lingual Question Generation with Language Agnostic Language Model

Bingning Wang, Ting Yao, Weipeng Chen, Jingfang Xu and Xiaochuan Wang
Sogou Inc.
Beijing, China
`{wangbingning,yaoting}@sogou-inc.com`

Abstract

Question generation is the task of generating coherent and relevant question given context paragraph. Recently, with the development of large-scale question answering datasets such as SQuAD, the English question generation has been rapidly developed. However, for other languages such as Chinese, the available training data is limited, which hinders the development of question generation in the corresponding language. To investigate the multi-lingual question generation, in this paper, we develop a language-agnostic language model, which learns the shared representation from several languages in a single architecture. We propose an adversarial training objective to encourage the model to learn both language-specific and language-independent information. We utilize abundant monolingual text to improve the multi-lingual question generation via pre-training. With the language-agnostic language model, we achieve significant improvement in multi-lingual question generation over five languages. In addition, we propose a large-scale Chinese question generation dataset containing more than 220k human-generated questions to benefit the multi-lingual question generation research.

1 Introduction

Question Generation (QG), also known as learning to ask, has attracted a lot of research interest in recent years. QG is regarded as the dual task of machine reading comprehension (Yuan et al., 2017; Xiao et al., 2018). Rather than answering a given question, learning to ask a coherent, relevant, and non-trivial question also requires a deep understanding of the context (Davey and McBride, 1986; Graesser et al., 2010), providing a good testbed for natural language understanding.

Conventional methods for question generation rely heavily on heuristic rules, and the standalone dependency parsing tool is needed to generate handcrafted templates (Mostow and Chen, 2009; Heilman and Smith, 2010; Rus et al., 2010; Hussein et al., 2014; Dhole and Manning, 2020). In recent years, with the development of deep learning and large-scale QA datasets, more and more neural network model has been proposed, which is also referred as neural question generation. Neural QG shows great advantage compared with previous rule-based systems in terms of both fluency and diversity of the generated questions (Duan et al., 2017; Yuan et al., 2017).

However, most progress in QG is made in English. For other languages such as Hindi, the lack of large-scale QG data limits its development. Recently, multi-lingual and cross-lingual language understanding has been studied in several NLP tasks, such as question answering (Liu et al., 2019; Cui et al., 2019), summarization (Zhu et al., 2019), natural language inference (Conneau et al., 2018), etc. For QG, Kumar et al. (2019) demonstrate that for low-resource Hindi, incorporating the large-scale English SQuAD (Rajpurkar et al., 2016) dataset could boost the QG result a lot.

For multi-lingual QG, a key factor is to learn a model that could transfer knowledge across different languages. In this paper, we propose a language-agnostic language model: it consists of the specific low-level module for each language, and a shared high-level module for multi-lingual information aggregation. Separating the language model into two levels enables us to learn the language-specific information in each language and the common information shared among languages. In this way, the knowledge in multi-lingual QG could be transferred via the high-level module.

For the language-agnostic language model, however, the distributed representation of the low-level module could be easily mixed with the language information, which makes the high-level module con-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2262–2272
August 1–6, 2021.

tain some unnecessary language-specific features that are too specific to transfer across languages. Inspired by previous works on transfer learning (Chen et al., 2017; Liu et al., 2017), we propose an adversarial training objective to decouple the low-level module with the high-level module, which prevents the private and shared latent feature spaces from interfering with each other, making the high-level module language-invariant, thus achieving better transferability for different languages.

To get a better initialization for our model, we develop two self-supervised methods to pre-train our model on abundant monolingual text. We apply our model to five languages QG tasks that have human-labeled QG datasets. The experimental results demonstrate that all languages QG could benefit from the multi-lingual training. Our models surpass previous monolingual or multi-lingual QG methods by a large margin, even in zero-shot learning where we had no training data in the low-resource languages, our model achieves satisfactory results by merely trained on English dataset, which shows a promising transferability of the proposed model.

Besides, we also propose a large-scale Chinese QG dataset containing more than 220k human-labeled questions. We hope the proposed Chinese dataset could benefit the community for more comprehensive multi-lingual QG research. The codes and proposed datasets are available at `https://github.com/benywon/LALM`.

Our contributions are summarized as follow:

- We propose a novel language-agnostic language model which decouples the language specific and language independent information in QG.
- The proposed model achieves significant improvement over previous models in multi-lingual QG, and we analyze the transferability in multiple languages.
- We release a large-scale human labeled Chinese QG dataset containing more than 220k questions. To our best knowledge, this is the largest specific question generation dataset so far.

2 Related Work

Question generation has received increasing attention from the research community. Traditional QG systems are mostly rule-based, which sometimes utilizing off-the-shelf tools to get the syntactic structure, dependency relations, and semantic role of the passage (Mostow and Chen, 2009; Heilman and Smith, 2010). First, the target answers are generated using rules or semantic roles, next, low-quality questions are generated using hand-crafted rules or templates. Finally, the generated questions are ranked by features such as keyword matching degree or sentence perplexity (Hussein et al., 2014). The main drawbacks of these symbolic systems are that the rules and templates are expensive to manually create, and lack diversity.

With the development of deep learning and large-scale question answering datasets, motivated by neural machine translation, Du et al. (2017) proposed a sequence to sequence (seq2seq) architecture combined with attention mechanism, achieving a promising result on QA dataset SQuAD. Since then, many works have been proposed to extend the preliminary framework with rich features, such as named entity tags (Zhou et al., 2017) or answer position features (Duan et al., 2017), and incorporate copy mechanism to copy words from the context paragraph (Song et al., 2018). Other types of models are also introduced such as graph neural networks (Chen et al., 2019) or Transformer (Scialom et al., 2019). However, most of these works are focus on English QG and have not been validated in other languages.

Multi-Lingual language generation. Duan et al. (2019) translated documents as weakly supervised training data for zero-shot multi-lingual abstractive summarization. Chi et al. (2019) proposed a multi-lingual pre-training method that can transfer monolingual supervision signals to other pre-trained languages. Zhu et al. (2019) adopt large-scale supervised data from existing monolingual summarization datasets via translation strategy to perform multi-lingual summarization. Kumar et al. (2019) also proposed a multi-lingual question generation methods based on Transformer, they proposed a small Hindi QG dataset and improved the QG result on Hindi by training with additional English data.

Compared with the previous multi-lingual methods, our method directly separates the language-dependent module and language-independent module. We propose an adversarial decoupling module to improve the adaptive ability of the model. Besides, our model could be properly pre-trained by monolingual data, which obviates the need to construct the back-translation or pseudo-parallel data.

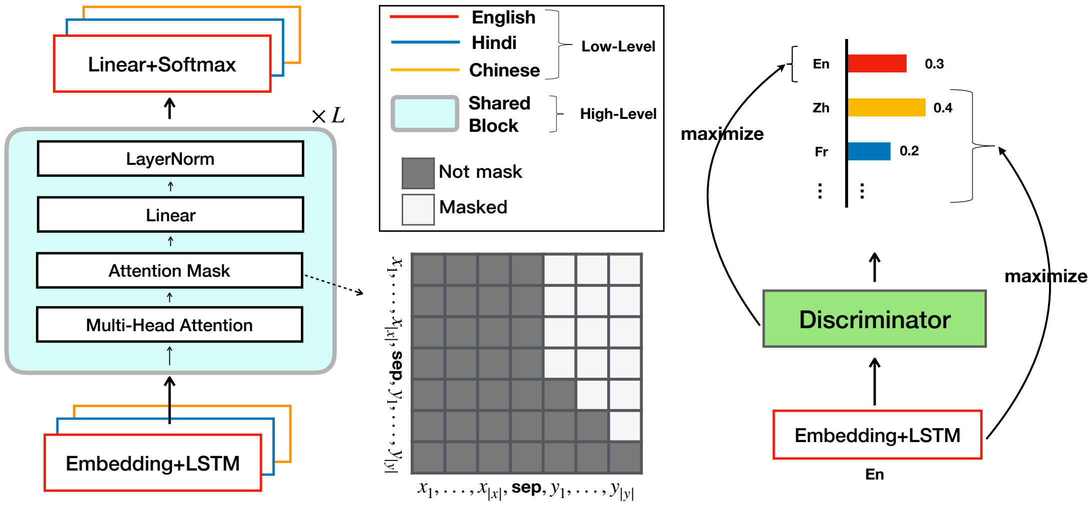

(a) (b) (c)

Figure 1: (a) The whole architecture of the proposed language-agnostic language model. It consists of the low-level language understanding module (Embedding+LSTM) and the high-level semantic understanding module (Transformer block), followed by a projection and softmax module. (b) The attention mask matrix M in the high-level module, $M_{i,j}$ means whether the word in position i could attend to the word in position j. The gray cells are allowed to attend and the others are masked to forbid attention. (c) The adversarial decoupling module where the discriminator tries to maximize the probability of the corresponding language while the generator (low-level module) tries to minimize it.

3 Language-agnostic Language Model

The Language-agnostic language model (LALM) consists of the low-level module and the high-level module, the whole architecture is illustrated in Figure 1(a) we describe it below.

3.1 Low-Level Module

The low-level module is built to perform the basic language understanding. In this paper, we adopt the LSTM (Hochreiter and Schmidhuber, 1997) encoder as the low-level language understanding module[1]. LSTM processes text in sequential order and embeds the language information into dense representations. We adopt the uni-directional LSTM in this paper to make the model auto-regressive.

For the language-agnostic language models, each language has its specific word embeddings and specific low-level language understanding LSTMs. This is different from some previous multi-lingual methods that a shared or aligned word embedding is utilized for different languages (Conneau et al., 2018; Lample and Conneau, 2019). Separating the language understanding module enables us to model specific linguistic characteristics in different languages. In Section 4, we will show that separating the low-level module for each language could benefit a lot for multi-lingual QG.

3.2 High-Level Module

The low-level module is built to perform the basic linguistic understanding, and the high-level module is built on top of the low-level module to perform higher-level information aggregation, which requires higher model capacity. In this paper, we use the Transformer (Vaswani et al., 2017) model as the high-level module.

The Transformer, with the core building-block called multi-head attention, has shown great advantages in representing languages in many NLP tasks. Current state-of-the-art models in natural language understanding benchmark GLEU[2] (Wang et al., 2018) are almost Transformer-based. In this paper, we focus on QG which is a sequence-to-sequence problem, so we adopt the mask operation similar with (Dong et al., 2019), which is illustrated in Figure 1(b). For a pair of sequence $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = x_1, ..., x_{|x|}$ is the source, and $\mathbf{y} = y_1, ..., y_{|y|}$ is the target, we concatenate them together with a special token `<sep>`, forming a single sequence with length $|\mathbf{x}| + |\mathbf{y}| + 1$. We want all the positions in the source $\{1, 2, ..., |\mathbf{x}|\}$ to attend to each other so we can obtain the bi-directional representations

---

[1] In fact, we also conduct experiments on adopting other types of models as the low-level module such as Transformer or GRU, but the result is not comparable with the LSTM.

[2] `https://gluebenchmark.com/`

of the source, and all the positions in the target $\{|\mathbf{x}|+1, ..., |\mathbf{x}|+|\mathbf{y}|+1\}$ are forbidden to attend to future words:

$$M_{i,j} = \begin{cases} 0, & j \leq |\mathbf{x}| \quad \text{or} \quad j \leq i, \\ -\infty, & \text{otherwise.} \end{cases} \tag{1}$$

This attention mask operation enables us to build a causal language model that the generation of the current word only depends on its previous words. Therefore, the probability of $\mathbf{y}$ could be denoted as:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{|y|} p(y_i|y_{<i}, \mathbf{x}) \tag{2}$$

And the loss for the whole model is the negative log likelihood of the data:

$$\mathcal{L}_{NLL} = -\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}} \sum \log p(y_i|y_{<i}, \mathbf{x}) \tag{3}$$

3.3 Adversarial Decoupling Module

In this paper, we want the representations of the low-level module in different languages to contain no language-specific information that is interleaved with the high-level module. In this way, the high-level module could focus on the semantic understanding shared across languages. We build a *discriminator* on top of the low-level module to determine whether the output of low-level representations contains the specific language information.

The discriminator is a bi-directional LSTM taking the output of the low-level module as input and tries to predict its language. Concretely, denote the output of the low-level module is $\mathbf{S} \in \mathbb{R}^{n,d}$ where n is the sequence length (i.e. $|\mathbf{x}|+|\mathbf{y}|+1$), and d is the hidden size of the low-level module. The output of the discriminator can be represented as:

$$\begin{aligned} \mathbf{H} &= \text{bi-LSTM}(\mathbf{S}) \\ \mathbf{h} &= \text{Max-Pooling}(\mathbf{H}) \\ \hat{\mathbf{h}} &= \text{MLP}(\mathbf{h}) \\ \hat{y} &= \text{Softmax}(\hat{\mathbf{h}}) \end{aligned} \tag{4}$$

$\mathbf{h} \in \mathbb{R}^d$ is a pooled representations of the discriminator for classification. $\hat{y}$ is the language distribution in $\mathbb{R}^C$ where C is the number of languages. For the discriminator, the target is to maximize the probability of the corresponding language while the low-level module (generator) tries to minimize it. Therefore, they form an adversarial training objective that the low-level module must produce representations without discriminative language information. In this way, the discriminator acts as an adversarial decoupling module (ADM) to encourage the low-level module to generate language-agnostic representations.

The architecture of ADM is shown in Figure 1 (c), and the loss function for the discriminator and low-level module (generator) are:

$$\begin{aligned} \mathcal{L}_D &= -\log p(\hat{y}_i) \\ \mathcal{L}_G &= -\log p(1-\hat{y}_i) \end{aligned} \tag{5}$$

where $\hat{y}_i$ is the discriminator probability for the input language i. In fact, the objective of the generator is to maximize the entropy of the discriminator's output to make it *less confident* of the language.

3.4 Pre-training

Recent works on NLP and language generation have shown the great advantage of large-scale pre-training (Devlin et al., 2018; Radford et al., 2019; Lewis et al., 2019; Roberts et al., 2020). In this paper, we also pre-train our model in massive multilingual text. Since our model is a sequence to sequence architecture, we develop two self-supervised objectives for language generation pre-training:

Denoised Auto-Encoder (DAE): Most previous works on natural language generation pre-training resort to DAE to initialize the model. In DAE, a corrupted version of the original sentence is created as the source and the model should reconstruct the original sentence. In this paper, we adopt the similar noising strategy as Lewis et al. (2019): (1) *Token Masking* random tokens are sampled and replaced with a special `[MASK]` token. (2) *Token Deletion* randomly deletes several tokens in the document. (3) *Token Replacement* randomly replace some tokens with other tokens in the vocabulary. (4) *Sentence Permutation* randomly swap some tokens in the sentence.

Next Sentence Generation: One of the problems of the DAE is that the input is always the corrupted sentence, which is not the case during finetune, the pretrain-finetune discrepancy may hurt the performance of the downstream tasks (Yang et al., 2019). Similar to Kiros et al. (2015) and Dong et al. (2019), we sample a consecutive segment in the text and divide it into two parts, we treat the first parts as the source and the second part as the target. The objective is to generate the second part based on the first part.

3.5 Question Generation Fine-tuning

After pre-training, we suppose the low-level module of our model has learned the multi-lingual linguistic information. Then the fine-tuning objective is to adjust the high-level module for question generation. Therefore, in this phase, we fix the low-level module, i.e. the word embedding, LSTM, and output projection linear layer, and only update the parameter of the high-level module.

4 Experiments

4.1 Dataset

The question generation datasets are sometimes directly derived from the corresponding question answering datasets. In the current question answering application, most multi-lingual datasets are automatically derived by translating from English SQuAD (Asai et al., 2018). However, it may reduce the multi-lingual QG tasks to translation tasks if we use these datasets. Therefore, we consider four different language QG datasets that are developed by the specific language speakers.

- **English (En)** We use the SQuAD (Rajpurkar et al., 2016) as the English question generation dataset. It is a standard machine reading comprehension data consists of nearly 100k human-labeled questions from Wikipedia.
- **Korean (Ko)** We use the Korquad1.0 (Lim et al., 2019) as the Korean QG data. It consists of more than 70,000 human-generated question-answer pairs on Korean Wikipedia articles.
- **French (Fr)** We adopt the French SQuAD-style dataset (d'Hoffschmidt et al., 2020) consisting of more than 25k human-curated French questions.
- **Hindi (Hi)** HiQuAD (Kumar et al., 2019) is a specific Hindi QG dataset containing 6,555 question-answer pairs. It was derived from the Hindi storybook.

Since the size of the QG dataset except English is comparative small, so we propose a new large-scale QG dataset created by humans on **Chinese (Zh)**. First of all, we collect nearly 3.5m passages from Baike[3], a Chinese Wikipedia-like encyclopedia. To increase the diversity of the selected paragraphs, we cluster the passages based on the bag-of-words, then we use Ward (Ward Jr, 1963) algorithm to select the centroid in each cluster, which result in nearly 100k passages. We ask volunteers to ask no more than 5 questions for each paragraph. Since we did not give the specific answer candidates for each paragraph, the annotators were encouraged to ask more general and comprehensive questions. We also ask other volunteers to check the quality and remove the questions that are either unanswerable or contain grammar errors. Finally, we obtain 224,962 question-paragraph pairs. We randomly select 180k of them as the training data, 20k samples for development, and the rest 24,962 for testing. We name it LAB (**L**earning to **A**sk on **B**aike).

	QG		Pre-train
	Train	Dev/Test	Name(Size)
En	86,635	8,965/8,964	enwiki(13.6Gb)
Zh	180,000	20,000/24,962	zhwiki(1.3Gb)
Ko	60,407	5,774/3,898	kowiki(608Mb)
Fr	20,731	3,188/2,189	frwiki(4.0Gb)
Hi	4,000	1,300/1,255	hiwiki(395Mb)

Table 1: The statistics of the multi-lingual pre-training data and question generation data.

We adopt the 2020-05-20 data dumps of the Wikipedia[4] in the corresponding language as the pre-training data. The details of the training data are shown in Table 1.

4.2 Implementation Details

In all experiments, we tokenize the text with sentencepiece (Kudo and Richardson, 2018). For all languages datasets, we set the vocabulary size to 30,000. We use the Adam (Kingma and Ba, 2014) optimizer with 5k warm-up steps and linearly decay the learning rate. $\beta_1, \beta_2, \epsilon$ was set to 0.9, 0.99 and 10^{-6}, respectively. For both pre-training and fine-tuning, the max learning rate was set to 10^{-4}. The batch size was 256 during pre-training and 64 during fine-tuning. We limit the max sequence length to 512. For the adversarial decoupling module training, following previous works of generative adversarial networks (Goodfellow et al., 2014; Salimans et al., 2016), the update rate for discriminator and generator was set to 1:10. For each of the 4 noising strategies in pre-training, we set the sample probability to 0.1. Similar with Scialom et al. (2019) we do not provide the answer and direcetly generate questions based on the context. We use three types of models:

LALM$_{\text{share}}$ is the shared language-agnostic lan-

---

[3] `https://baike.sogou.com/`

[4] `https://dumps.wikimedia.org/`

		Transformer	NQG++	Multi-BERT	CLQG	XNLG	$\text{LALM}_{\text{share}}$	$\text{LALM}_{\text{base}}$	$\text{LALM}_{\text{large}}$	$\text{LALM}_{\text{large}}$+ADM
En	BLEU-4	14.03	15.09	17.19	17.63	19.98	20.96	21.95	23.50	**24.94**
	METEOR	17.62	18.04	18.38	18.91	20.24	20.23	21.30	22.15	**23.28**
	ROUGE	40.79	40.24	44.82	43.34	46.51	47.47	48.23	50.34	**51.42**
Zh	BLEU-4	22.75	20.32	35.08	34.96	37.40	36.11	38.32	43.19	**44.10**
	METEOR	17.24	18.95	26.10	26.54	27.13	27.28	27.99	32.38	**33.04**
	ROUGE	30.14	29.87	38.46	40.11	42.15	43.25	44.49	45.16	**46.40**
Ko	BLEU-4	7.11	7.95	10.35	8.97	-	11.93	12.19	12.58	**12.93**
	METEOR	14.30	14.81	18.10	17.22	-	19.85	20.11	20.96	**21.10**
	ROUGE	22.17	24.13	31.28	29.34	-	34.10	34.88	34.79	**35.02**
Fr	BLEU-4	4.48	5.03	8.95	10.18	12.93	13.38	13.95	14.87	**15.28**
	METEOR	13.05	13.19	15.91	16.28	18.37	17.75	18.20	18.84	**19.92**
	ROUGE	32.17	31.66	39.34	41.23	40.96	41.15	42.80	43.11	**44.51**
Hi	BLEU-4	9.77	10.10	23.15	20.24	-	30.35	32.21	34.02	**35.19**
	METEOR	23.85	24.32	30.29	29.15	-	33.80	34.22	35.97	**36.25**
	ROUGE	33.16	34.91	41.06	40.64	-	48.82	49.14	50.94	**51.23**

Table 2: Main result of the multi-language QG. $\text{LALM}_{\text{share}}$ is similar with previous multi-lingual model that the parameters are shared across all languages. ADM represents the model trained with adversarial decoupling module.

guage model. It is similar with the proposed model but has no specific low-level LSTM for each language. That is, the low-level and high-level parameters are both shared across different languages. The hidden size was set to 768 and the layer size was set to 12, and each layer consists of 12 heads. We set the shared vocabulary size to 100,000.

$\text{LALM}_{\text{base}}$ is the base version of our model. It has the same hidden size as $\text{LALM}_{\text{share}}$. The low-level module was single layer uni-directional LSTM with hidden size 768. $\text{LALM}_{\text{base}}$ has nearly 138m parameters, where nearly half of them are low-level language understanding parameters.

$\text{LALM}_{\text{large}}$ is the large version of our proposed model. The hidden size, layer size, and head size were set to 1024,24,16, respectively. The low-level module was two-layer uni-directional LSTMs. $\text{LALM}_{\text{large}}$ has 548m parameters, where nearly a quarter of them are low-level module's parameters.

4.3 Criterion:

Following previous works of QG (Zhou et al., 2017; Chen et al., 2019), we adopt three widely used automatic metrics for evaluation: **BLEU**, **Meteor** and **Rouge-L**, which measure the n-gram similarities between the generated questions and real questions.

4.4 Baselines

We adopt 5 baseline methods for comparison.

□ **Transformer** (Vaswani et al., 2017; Scialom et al., 2019) is the most widely used architecture in sequence-to-sequence learning. For each language, we train the correspondent Transformer model based on its training data. We set dropout ratio to 0.4 to prevent overfitting.

□ **NQG++** (Zhou et al., 2017) is a popular neural QG model based on LSTM. It is enhanced with attention and copy mechanism[5].

□ **Multi-BERT** (Devlin et al., 2018) is a multi-lingual extension to the original BERT model. It was trained on the multi-lingual wikipedia. All the language shares the same vocabulary. We adopt the way same with Rönnqvist et al. (2019) to extend BERT to language generation task.

□ **CLQG** (Kumar et al., 2019) is a cross-lingual QG method based on Transformer. It is pre-trained by denoising autoencoders along with back-translation. We use the public implementation[6] and adopt the same word tokenization as well as pre-training data as our model.

□ **XNLG** (Chi et al., 2019) is a multi-lingual language generation model that transfers monolingual supervision to all pre-trained languages. It was trained with English, Chinese and French datasets. We use their public pre-trained models[7] and fine-tune on the three QG dataset.

4.5 Multi-Lingual Question Generation

To evaluate the multi-lingual question generation ability of the proposed methods, we assemble all

---

[5] `https://github.com/magic282/NQG`

[6] `https://github.com/vishwajeet93/clqg`

[7] `https://github.com/CZWin32768/XNLG`

BLEU-4		
Zh		
F P	✓	✗
✓	38.32	36.03
✗	–	34.22
En		
F P	✓	✗
✓	21.95	20.61
✗	-	17.93

ROUGE		
Zh		
F P	✓	✗
✓	44.49	41.73
✗	–	40.12
En		
F P	✓	✗
✓	48.23	47.15
✗	-	45.02

Table 3: Multi-lingual and mono-lingual results for LALM$_{\text{base}}$. P denotes the pre-training and F denotes the fine-tuning, where ✓denote the multi-lingual while ✗denotes the mono-lingual training. For example, the upper right cell in each table denotes pre-training with multi-lingual but finetuning with mono-lingual.

QG data and train the LALM thereof. For Transformer and NQG++, we initialize the word embeddings by fasttext multilingual word embeddings (Grave et al., 2018). The result is shown in Table 2.

We can see from the table that our model excels at multi-lingual QG, achieving significant improvement over previous methods in all languages. Compared with other architectures such as Transformer, we explicitly separate the low-level and the high-level module in the proposed model and use adversarial networks to decouple them. Therefore, the shared high-level module is encouraged to learn more common representations across different languages, which is more transferable and benefits the downstream QG task a lot.

Besides, we can see that if we don't explicitly separate the low and high-level parameters (LALM$_{\text{share}}$), the result drops a lot. We hypothesis that different languages have different low-level language information, such as lexical, syntactical, etc. Embedding all language processing procedures into a single model may make the model hard to discriminate the language-specific information.

Besides, the model trained with the adversarial decoupling module achieves further improvement, the ADM may impose an implicit regularization on the low-level module to make the representations more abstract, and therefore encourage the high-level module to learn more common representations (Chen et al., 2017; Liu et al., 2017).

4.6 Human Evaluation

The automatic metrics are sometimes biased toward a specific attribute of the generated question (Hosking and Riedel, 2019). So we conduct human qualitative evaluation of the generated outputs. We consider three aspects of the generated questions: **Fluency**: Whether the generated questions are well-posed and natural, in terms of both grammar and semantic. **Answerable**: Whether the generated questions could be answered by the context paragraph. **Significance**: Whether the generated question is just a simple syntactical transformation of the paragraph sentence or trivial one that seems unlikely asked by human.

We randomly sample 50 generated questions from English and 50 from Chinese and ask three volunteers to evaluate the sample quality. The result is shown in Table 5. The result shows our proposed model is also excels at human evaluation, especially for significance, which is sometimes regarded as the most important factor in QG (Graesser et al., 2010). We also showcase some outputs of our model in Table 4. We can see that LALM could generate fluent and sound questions.

4.7 Multi-Lingual v.s. Mono-Lingual

Kumar et al. (2019) have found that in QG the performance of Hindi could be improved by training with additional English data. In this section, we evaluate whether the multi-lingual is superior to the mono-lingual QG. We focus on two aspects: **(1) Pre-training**. In contrast to the proposed multi-lingual pre-training, we adopt the mono-lingual pre-training where we only pretrain on specific languages[8] and fine-tune the QG models in the same language. **(2) Fine-tuning**. Different from the setup in Sec. 4.5 where we aggregate all languages QG data for training, we only fine-tune the model on specific language.

We experiment on English and Chinese with the LALM base model. The BLEU-4 and ROUGE-L scores are shown in Table 3. It is clear that for both pre-training and fine-tuning, the multi-lingual training improves the model a lot. Moreover, the multi-lingual plays a more important role in pre-training than in fine-tuning. We suppose that during pre-training, multiple languages perform a type of regularization on the shared high-level module, while in fine-tuning the language-dependent super-

---

[8] Therefore, we omit the adversarial decoupling module since it only takes effect on multi-lingual learning.

Table 4: Some generated cases of the proposed model.

English

Context: *The United Methodist Church opposes conscription as incompatible with the teaching of Scripture. Therefore, the Church supports and extends its ministry to those persons who conscientiously oppose all war, or any particular war, and who therefore refuse to serve in the armed forces or to cooperate with systems of military conscription. However, the United Methodist Church also supports and extends its ministry to those persons who conscientiously choose to serve in the armed forces or to accept alternative service. The church also states that ”as Christians they are aware that neither the way of military action, nor the way of inaction is always righteous before God.”*

Original: The Church supports those persons who conscientiously oppose what?
LALM: what does the church states after they oppose the construction ?

Chinese

Context: 电桥平衡#四个电阻*R0*、*R1*、*R2*、*Rx*连成四边形,称为电桥的四个臂。四边形的一个对角线连有检流计,称为“桥”;四边形的另一对角线接上电源,称为电桥的“电源对角线”。*E*为线路中供电电源,学生实验用双路直流稳压电源,电压可在*0-30V*之间调节。*R*保护为较大的可变电阻,在电桥不平衡时取最大电阻作限流...

Original: 什么是电桥平衡?
LALM: 电桥平衡有什么用?

French

Context: *Le seul quartier d’habitation à avoir été fouillé est situé sur le site du Merkes, à l’est de la Voie processionnelle et du complexe sacré, entre les anciens quartiers de Ka-dingirra, Eridu et Shuanna. Sa voirie est caractérisée par des rues étroites approximativement rectilignes et se coupant quasiment à angles droits. Il s’agit peut-être de l’héritage d’un ancien plan orthogonal planifié qui a été altéré à la suite de remaniements de constructions, courants en raison de l’altération rapide des constructions en briques crues qui doivent régulièrement être restaurées.*

Original: En quoi sont fabriquées les habitations ?
LALM: Quelles sont les caractéristiques de la route ?

Korean

Context: 칭짱(藏)고원이라고도 불리는 티베트 고원은 동아시아에 위치한 넓고 높은 고원이다. 티베트자치구역과 중국 칭하이성(海省), 그리고 인도 카슈미르에 걸쳐 있는 티베트 고원은 남북 1000km, 동서 2500km에 뻗어 있으며, 그 평균 높이는 4500 미터가 넘는다. ’세계의 지붕’으로 불릴 만큼 세계에서 가장 높고 크며 면적은 약 250만 평방 킬로미터나 된다. 이 고원은 인도-호주 플레이트와 유라시아 플레이트가 신생대에 충돌하며 생성되었으며 그 과정은 지금도 진행되고 있다. 이 고원은 산맥과 소금 호수가 분포한 고원의 건조 스텝지대를 형성하고 있다. 한 해 평균 강수량은 100mm에서 300mm로, 강수량의 대부분은 우박을 이룬다. 유목민들은 고원의 남부 및 동부 경계의 한 해 6개월가량 서리가 내리는 목초지에서 유목 생활을 유지하고 있다.

Original: 티베트고원의 면적은?
LALM: 티베트고원은어디에있습니까?

	Fluency	Answerable	Significance	Ave.
NQG++	1.01	1.09	1.02	1.04
Multi-BERT	1.22	1.19	1.23	1.21
LALM$_{\text{base}}$	**1.38**	**1.29**	**1.46**	**1.37**

Table 5: Human assessment of the generated questions on English and Chinese. Each question was assigned to score in $\{0,1,2\}$ which correspond to *bad*, *ok* and *excellent*, respectively. The result is statistical significant with $p < 0.05$.

vision of QG is more specific, which makes transfer learning less useful.

4.8 Zero-Shot Learning

In this section, we study the zero-shot multi-lingual learning ability of our model. The previous Section demonstrates that English SQuAD could strengthen other languages a lot. So we choose SQuAD as the training data and evaluate other languages. We only update the parameters of the high-level module for SQuAD without modifying the low-level language understanding module. Therefore, the replacement of the low-level module has little influence on the whole architecture, making the zero-shot inference available. We compare the zero-shot results of LALM base model with the supervised NQG++. The result is shown in Table 7.

We can see that the zero-shot version of our LALM appears to have equaled or eclipsed the QG ability of NQG++. It is an interesting result showing our model could transfer the question generation ability of English to other languages even without supervision. However, pure zero-shot learning is still struggle to achieve a good result, the supervision from the target language is necessary.

4.9 The Effect of Pre-training

We propose the self-supervised denoised auto-encoding and next sentence generation to pre-train the model. In this section, we construct a model that does not employ the pre-training but directly fine-tuned on the target data. The LALM hidden size to 256 and layer and head numbers of 4 and 8, respectively, to prevent overfitting. The results of English, Chinese, and Hindi are shown in Table 6. The performance of our model drops a lot without pre-training. Especially, it barely performs well for the low resource Hindi data because there

	En			Zh			Hi		
	B4	M	R	B4	M	R	B4	M	R
LALM	14.35	17.41	33.52	22.25	21.43	37.94	12.92	24.19	33.10
LALM+ADM	15.94	18.24	36.24	24.10	22.05	38.78	14.13	23.77	34.24
LALM+ADM+Pre-train	21.95	21.30	48.23	38.32	27.99	44.49	30.35	34.80	48.82

Table 6: Ablation study of the pre-training. The three models are fine-tuned on multi-lingual data.

		B1	B2	B3	B4	M	R
Zh	F	**43.07**	**31.04**	**23.58**	**17.74**	**18.06**	**22.44**
	Z	26.55	18.26	12.10	10.94	11.94	15.89
Kr	F	**25.20**	**15.34**	**10.71**	5.07	**14.35**	16.42
	Z	20.55	11.17	8.32	**5.95**	13.32	**16.77**
Fr	F	**31.31**	**14.91**	10.46	5.54	9.63	**23.02**
	Z	25.58	13.33	**12.49**	**6.32**	**11.06**	15.22
Hi	F	**30.15**	**20.42**	12.30	9.03	23.47	32.84
	Z	24.10	15.77	**12.54**	**10.89**	**26.42**	**33.01**

Table 7: Zero-shot multi-lingual evaluation. F denotes the performance of NQG++ model, and Z denotes zero-shot result where we fine-tune LALM base model on SQuAD and directly evaluate on other datasets.

are only 4,000 training instances. Nevertheless, when trained with the adversarial decoupling module, our model could achieve consistent improvement, demonstrating that the ADM is good at multi-lingual transfer learning.

5 Conclusion

In this paper, we propose a language-agnostic language model to deal with the multi-lingual question generation. The model consists of the low-level and the high-level module to explicitly represent the language-dependent and language-independent information, respectively. We operate the attention mask matrix to fit our model to the sequence to sequence learning. We propose an adversarial training mechanism to decouple the two-level modules, making the low-level module contains more abstractive representations and the high-level module language-agnostic. We also proposed a large-scale Chinese QG data containing more than 220k questions. Experiments on five languages demonstrate our model achieves significant improvements over previous methods in multi-lingual QG. For future work, we would like to apply our proposed model to other multi-lingual tasks such as summarization and question answering.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. And we appereate the dedicated labeling efforts contributed by the annoators, which makes the large-scale Chinese QG datasets avaliable for the community.

References

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual extractive reading comprehension by runtime machine translation. *arXiv preprint arXiv:1809.03275*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuan-Jing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942*.

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2019. Cross-lingual natural language generation via pre-training. *arXiv preprint arXiv:1909.10481*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2019. Cross-lingual machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1586–1595.

Beth Davey and Susan McBride. 1986. Effects of question-generation training on reading comprehension. *Journal of Educational Psychology*, 78(4):256.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Martin d'Hoffschmidt, Maxime Vidal, Wacim Belblidia, and Tom Brendlé. 2020. Fquad: French question answering dataset. *arXiv preprint arXiv:2002.06071*.

Kaustubh D Dhole and Christopher D Manning. 2020. Syn-qg: Syntactic and shallow semantic rules for question generation. *arXiv preprint arXiv:2004.08694*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *EMNLP*, pages 866–874.

Xiangyu Duan, Mingming Yin, Min Zhang, Boxing Chen, and Weihua Luo. 2019. Zero-shot cross-lingual abstractive sentence summarization through teaching generation and attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3162–3172.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Art Graesser, Yasuhiro Ozuru, and Jeremiah Sullins. 2010. What is a good question?

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *HLT-NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tom Hosking and Sebastian Riedel. 2019. Evaluating rewards for question generation models. In *NAACL*, pages 2278–2283.

Hafedh Hussein, Mohammed Elmogy, and Shawkat Guirguis. 2014. Automatic english question generation system based on template driven scheme. *IJCSI*, 11(6):45.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ICLR*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Vishwajeet Kumar, Nitish Joshi, Arijit Mukherjee, Ganesh Ramakrishnan, and Preethi Jyothi. 2019. Cross-lingual training for automatic question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4863–4872.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1. 0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*.

Jiahua Liu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2019. Xqa: A cross-lingual open-domain question answering dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2358–2368.

Pengfei Liu, Xipeng Qiu, and Xuan-Jing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *AIED*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of* *the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Samuel Rönnqvist, Jenna Kanerva, Tapio Salakoski, and Filip Ginter. 2019. Is multilingual bert fluent in language generation? *arXiv preprint arXiv:1910.03806*.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Thomas Scialom, Benjamin Piwowarski, and Jacopo Staiano. 2019. Self-attention architectures for answer-agnostic neural question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6027–6032.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *NAACL*, pages 569–574.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Han Xiao, Feng Wang, Yanjian Feng, and Jingyao Zheng. 2018. Dual ask-answer network for machine reading comprehension. *arXiv preprint arXiv:1809.01997*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *NLPCC*.

Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang, and Chengqing Zong. 2019. Ncls: Neural cross-lingual summarization. *arXiv preprint arXiv:1909.00156*.

Structure-Aware Pre-Training for Table-to-Text Generation

Xinyu Xing and Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
`{xingxinyu,wanxiaojun}@pku.edu.cn`

Abstract

Table-to-text generation is a subtask of data-to-text generation which aims to generate naltural language text based on input table. Pre-training techniques have achieved great success on table-to-text generation. However, the pre-trained models used in previous works are typically trained on free-form natural language text while the input of table-to-text task is structured table. In this paper, we propose STTP, a pre-trained model that is trained with tables and their contexts. The STTP model can understand the structured input table and generate fluent text. Experiments on two datasets show the efficacy of our model.

1 Introduction

Data-to-text generation (Reiter and Dale, 1997) is an important natural language generation task with many practical applications, and it refers to the task of generating textual output from non-linguistic input data. The input data of the task can include tables of records, simulations of physical systems, spreadsheets, and so on. The output of the task is a natural language text. Datasets in common use include WEATHERGOV(Liang et al., 2009), ROTOWIRE(Wiseman et al., 2017), WebNLG(Gardent et al., 2017) and so on. Neural generation models with different improvements have achieved impressive results on data-to-text task. Table-to-text generation is a subtask of data-to-text generation which takes tables as input.

The pretrain-and-finetune framework, which refers to first pre-training a high capacity model on large corpora and then fine-tuning it on a downstream task, has outperformed prior state of the art on both natural language understanding task and natural language generation task. Inspired by the success of transfer learning, recently some works (Mager et al., 2020; Kale, 2020; Ribeiro et al., 2020) try to apply the pretrain-and-finetune framework on data-to-text generation. They fine-tuned the pre-trained model such as BART(Lewis et al., 2019) or T5(Raffel et al., 2019) on several downstream data-to-text tasks and achieved state-of-the-art results.

Although transfer learning has achieved great success on data-to-text generation, the pre-trained models used in previous works are typically trained on free-form natural language texts while the input of table-to-text task is structured table. The text-to-text pre-trained models learn a lot of knowledge and good language models from large amount of texts, so they work well on data-to-text generation task. But they still lack the ability to understand the structured data. So we propose a structure-aware table-to-text pre-trained model STTP which is trained with tables and their contexts for table-to-text task. STTP is built on top of the text-to-text pre-trained model BART, and it can understand the structured table and describe it with natural language text. We train our model based on BART because we hope our model can benefit from the knowledge and language model learned from large corpora. We propose three self-supervised tasks to train our model with large amount of tables and their contexts. The first self-supervised task is masked table language model (MTLM) which is like the classic MLM of BERT. The second self-supervised task is adjacent cell prediction (ACP) which refers to predict the cells around the current cell. The third self-supervised task is context reconstruction (CR) which refers to reconstructing the context of a table given the table and its broken context. The first two tasks aim to train the model to better understand the structured table, while the latter task aims to align the table and text. We use the tables extracted from WDCWebTable Corpus(Lehmberg et al., 2016) and their contexts to train our model. Experimental results on WEATHERGOV dataset and WebNLG dataset show the

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2273–2278
August 1–6, 2021.

efficacy of our model.

The main contributions of this work are:

- We propose a structure-aware table-to-text pre-trained model STTP which is trained with three self-supervised tasks.
- Experimental results on WEATHERGOV dataset and WebNLG dataset show the efficacy of our model. Code will be released at `https://github.com/XingXinyu96/STTP`.

2 Related work

Data-to-text generation task involves taking structured data as input and generating text that describes this data. Traditional approaches (Stent et al., 2004; Walker et al., 2007) deal with the task in two steps: the selection of a subset of the input data to discuss and the surface realization of a generation. More recent works combine both steps by learning content plan and surface realization jointly with end-to-end models (Wen et al., 2015; Peng et al., 2017). Although the end-to-end model has achieved good results, many models (Perez-Beltrachini and Lapata, 2018; Sha et al., 2018; Puduppully et al., 2019) consider adding content selection and content planning modules to the end-to-end framework to improve performance. A lot of other new models with different improvements (Wiseman et al., 2018; Li and Wan, 2018; Liu et al., 2018; Roberti et al., 2019; Rebuffel et al., 2020) are proposed to explore how to build an effective data-to-text generator.

Inspired by the success of the pre-trained models in other natural language generation tasks, Harkous et al. (2020), Kale (2020) and Ribeiro et al. (2020) achieve state-of-the-art results on different data-to-text benchmarks with different pre-trained models. However, the existing pre-trained models are usually designed to generate text based on text input, thus lacking the ability to understand structured inputs. Several pre-training methods designed for table-to-text task have been proposed. Deng et al. (2020) present a weakly supervised Structure-Grounded pretraining framework (STRUG) for text-to-SQL that can effectively learn to capture text-table alignment. But their model is only for text-to-SQL task and need parallel text-table data. Yin et al. (2020) propose TABERT, a pretrained model which is trained with large amount of tables with their context. Their model is also used for text-to-SQL task. Chen et al. (2020a) propose a knowledge-grounded pre-trained (KGPT) model which is trained on a massive knowledge- grounded text corpus crawled from the web. Li et al. (2020) propose two self-supervised tasks, Number Ordering and Significance Ordering, to help to learn better table representation.

3 Approach

We use the same model architecture as BART, and add several classification layers on top of the encoder for our new self-supervised tasks. We train our model based on the text-to-text pre-trained model BART instead of training from scratch because we hope our model can benefit from the knowledge and language model BART learned from large corpus of text. The three self-supervised tasks are shown in Figure 1.

3.1 Self-Supervised Tasks

Task 1: Masked Table Language Model (MTLM). Inspired by the classic Masked Language Model proposed by BERT, we propose the Masked Table Language Model (MTLM) to learn the representation of input table. During pre-training, we treat the table as a sequence and randomly replace 15% of tokens in the table with [MASK] symbols and then the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary to predict the original tokens. In this way, our STTP model learns to understand the structured table. Although we do not explicitly consider the structure of the input table in this task, it is obvious that our model needs to understand the structure of the table to predict the masked tokens.

Task 2: Adjacent Cell Prediction (ACP). The previous MTLM task does not explicitly consider the structure of the input table, so we propose a new task Adjacent Cell Prediction (ACP) to explicitly help our model better understand the structure of the table. For a cell in a table, the surrounding cells are usually very important to understand it. So we feed the hidden vectors of a cell into several output layers to predict its top, bottom, left and right cells. In other words, we hope that the hidden vector of each cell can contain the information of other cells in the same row or column. This task requires our model to focus more on the relationship between cells in the same row or column when encoding the table. For efficiency, we only use the nearest cells in the same row or column of each cell as targets

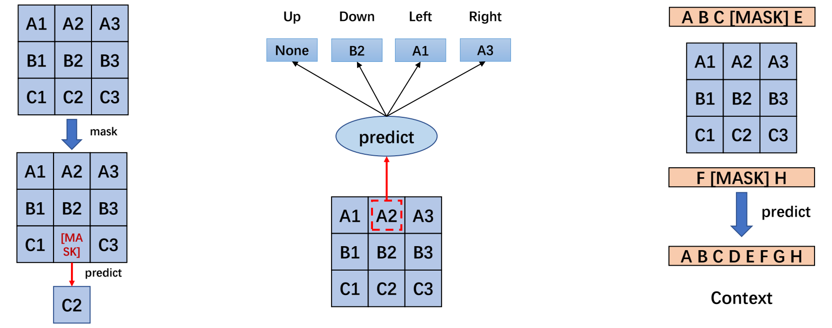

Task 1: Masked Table Language Model (MTLM)

Task 2: Adjacent Cell Prediction (ACP)

Task 3: Context Reconstruction (CR)

Figure 1: Three Self-supervised Tasks for our STTP Model. Table is shown in blue color and context text is shown in wheat color.

for prediction.

Task 3: Context Reconstruction (CR). The previous two tasks focus on better understanding the structured table, but for table-to-text generation tasks, another important aspect is the alignment of table and text. Through the previous two tasks, our model can get a better representation of the table, then we need to generate text based on this representation which needs the alignment of table and text. If we only train the encoder with the previous two tasks and do not consider the alignment of table and text, we might get a better representation of input table but the representation is hard to understand for the well-trained decoder provided by BART. Table-text alignment data are difficult to obtain, so we use the table with text context (which is usually not strictly aligned with the table, but somewhat relevant with the table) to help our model align table and text. We randomly mask 15% tokens of the context and reconstruct the broken context with our model. There also exists a mismatch between pre-training and fine-tuning, since the input of our model is usually just a table without its context during fine-tuning. Therefore, we also train our model with a task to predict context based only on the input table. Since the table and its context are not exactly aligned, it is difficult to predict the context only from the table, but this task can help our model mitigate the mismatch problem.

3.2 Dataset for Pre-training

The first two tasks only need unsupervised tables, while the third task needs tables with context text. Yin et al. (2020) collect tables and their surrounding text from English Wikipedia and the WDCWebTable Corpus(Lehmberg et al., 2016) to train their model. The data they use is also suitable for our task. We use the preprocessing tool they provided to handle the WDCWebTable Corpus and get a lot of tables with context. Then we filter out the data with low matching degree between table and context, because such data are difficult to train for the third task. In addition, we filter out tables that contains too many numbers. Finally, we get 800k tables with context. We linearize the structure of tables to be compatible with our model. We add a special token [cell] in the middle of cells in the same row and add another special token [row] in the middle of rows of the table. When training task 3, the input of model includes both text and table, so we concatenate them together and add a special token [TABLE] between text and table.

3.3 Pre-training Procedure

We alternately train our model with the previously mentioned three self-supervised tasks. The first two tasks are only used to train the encoder while keeping the decoder frozen. The third task is divided into two subtasks in practice: one is to reconstruct the context given the table with its damaged context; the other one is to predict the context given only the table. Both of the two subtasks train the encoder and decoder at the same time. Considering that the latter subtask is very difficult because the context of a table is difficult to be predicted in many cases, we reduce the times of training it.

4 Experiment

4.1 Dataset

We perform the experiments on WEATHERGOV dataset(Liang et al., 2009) and WebNLG dataset(Gardent et al., 2017). In the WEATHERGOV dataset, the output text is a weather report, and the source data provides a structured representation of the temperature, sky conditions, etc. The WEATHERGOV dataset consists of 29, 528 scenarios, each with 36 weather records paired with a natural language weather forecast (28.7 avg. word length). The WebNLG challenge consists of mapping sets of RDF triples to text. The newest WebNLG dataset contains 16, 095 data inputs and 42, 873 data-text pairs. The average length of the output text is 22.3 words. We convert the input data into a table and then linearize the structure of table like what we do when pre-training.

4.2 Results

4.2.1 Results on WeatherGov Dataset

We use a batch size of 4 and finetune for 100 epochs over the WeatherGov Dataset. Results are presented in Table 1. As is shown in this table, seq2seq model(Mei et al., 2015) has achieved very good results, but the pre-trained model further improves the results greatly. The BLEU scores of the pre-trained models are more than 80, which indicates that the generated text is highly similar to the gold text. BART-Retrain refers to finetuning BART on both our pre-training dataset and the datasets in downstream tasks. Our STTP model outperforms the BART-Retrain model, which proves the improvements of STTP model over BART model is from the proposed training objective instead of the additional training data.

Model	BLEU	METEOR
(Mei et al., 2015)	61.01	n/a
BART-base	81.54	54.81
BART-Retrain	81.63	55.50
STTP	**82.63**	**56.35**

Table 1: Results on WeatherGov Dataset.

4.2.2 Results on WebNLG Dataset

We use a batch size of 4 and fine-tune for 16 epochs over the WebNLG Dataset. Results are presented in Table 2. The results of Seq2Seq, Seq2Seq+Delex and Seq2Seq+copy are copied from (Shimorina and Gardent, 2018). The results of GCN and KGPT-Seq are copied from (Chen et al., 2020b). As is shown in this table, all pre-trained models outperform the models without pre-training even if some of them do not explicitly consider the structure of the input data. This is due to the pre-trained models learn a lot of external knowledge and a good language model from large corpora. The structure-aware model like GCN outperforms the normal seq2seq model, which shows that structure understanding is important in this task. Our model further outperforms the BART-base model and the KGPT model, which show the efficacy of our model with new self-supervised tasks.

Model	BLEU	METEOR
Seq2Seq	54.0	37.0
Seq2Seq+Delex	56.0	39.0
Seq2Seq+Copy	61.0	42.0
GCN	60.80	42.76
KGPT-Seq	64.11	46.30
BART-BASE	62.62	43.28
BART-Retrain	62.89	43.34
STTP	**64.92**	**46.48**

Table 2: Results on WebNLG Dataset

We randomly sample 50 instances from the WebNLG dataset and perform human evaluation on them. Three graduate students are employed to rank the generated texts produced by each model in three aspects: readability (whether the generated text is fluent), accuracy (whether the information of the generated texts is consistent with that contained in the input table) and overall quality. We use Best-Worst Scaling (Louviere et al., 2015), which has been shown to produce more reliable results than ranking scales (Kiritchenko and Mohammad, 2017). Specifically, each score is computed as the percentage of times it was selected as best minus the percentage of times it was selected as worst, and ranges from -1 (unanimously worst) to +1 (unanimously best). Human evaluation results on WebNLG dataset are shown in Table 3. We can see our model outperforms KGPT model and BART-base model, which further demonstrates the efficacy of our method. Running examples are provided in the supplementary materials.

Model	Readability	Accuracy	Overall
KGPT-Seq	0.09	0.01	0.04
BART-base	-0.22	-0.10	-0.17
STTP	**0.13**	**0.09**	**0.13**

Table 3: Human Evaluation Results on WebNLG Dataset

5 Conclusion

In this paper, we propose STTP, a pre-trained model trained with tables and their contexts. STTP model has achieved great performance on two downstream tasks. In the future work, we hope to collect more data and try other self-supervised tasks to train more effective model for table-to-text task.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Xiaojun Wan is the corresponding author.

References

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020a. Kgpt: Knowledge-grounded pre-training for data-to-text generation. *arXiv preprint arXiv:2010.02307*.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020b. KGPT: Knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8635–8648, Online. Association for Computational Linguistics.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-sql. *arXiv preprint arXiv:2010.12773*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv preprint arXiv:2005.10433*.

Svetlana Kiritchenko and Saif M Mohammad. 2017. Capturing reliable fine-grained sentiment associations by crowdsourcing and best-worst scaling. *arXiv preprint arXiv:1712.01741*.

Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Liang Li, Can Ma, Yinliang Yue, Linjun Shou, and Dayong Hu. 2020. Learning better representation for tables by self-supervised tasks. *arXiv preprint arXiv:2010.07606*.

Liunian Li and Xiaojun Wan. 2018. Point precisely: Towards ensuring the precision of data in generated texts using delayed copy mechanism. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1044–1055.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. *arXiv preprint arXiv:1702.05053*.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. *arXiv preprint arXiv:1804.06385*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. *arXiv preprint arXiv:1906.03221*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *European Conference on Information Retrieval*, pages 65–80. Springer.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

Marco Roberti, Giovanni Bonetta, Rossella Cancelliere, and Patrick Gallinari. 2019. Copy mechanism and tailored training for character-based data-to-text generation. *arXiv preprint arXiv:1904.11838*.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands. Association for Computational Linguistics.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics.

Marilyn A Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

On the Interplay Between Fine-tuning and Composition in Transformers

Lang Yu
Deptartment of Computer Science
University of Chicago
`langyu@uchicago.edu`

Allyson Ettinger
Department of Linguistics
University of Chicago
`aettinger@uchicago.edu`

Abstract

Pre-trained transformer language models have shown remarkable performance on a variety of NLP tasks. However, recent research has suggested that phrase-level representations in these models reflect heavy influences of lexical content, but lack evidence of sophisticated, compositional phrase information (Yu and Ettinger, 2020). Here we investigate the impact of fine-tuning on the capacity of contextualized embeddings to capture phrase meaning information beyond lexical content. Specifically, we fine-tune models on an adversarial paraphrase classification task with high lexical overlap, and on a sentiment classification task. After fine-tuning, we analyze phrasal representations in controlled settings following prior work. We find that fine-tuning largely fails to benefit compositionality in these representations, though training on sentiment yields a small, localized benefit for certain models. In follow-up analyses, we identify confounding cues in the paraphrase dataset that may explain the lack of composition benefits from that task, and we discuss potential factors underlying the localized benefits from sentiment training.

1 Introduction

Transformer language models like BERT (Devlin et al., 2019), GPT (Radford et al., 2018, 2019) and XLNet (Yang et al., 2019b), have improved the state-of-art in many NLP tasks since their introduction. The versatility of these pre-trained models suggests that they may acquire fairly robust linguistic knowledge and capacity for natural language "understanding". However, an emerging body of analysis demonstrates a level of superficiality in these models' handling of language (Niven and Kao, 2019; Kim and Linzen, 2020; McCoy et al., 2019; Ettinger, 2020; Yu and Ettinger, 2020).

In particular, although *composition*—a model's capacity to combine meaning units into more complex units reflecting phrase meanings—is an indispensable component of language understanding, when testing for composition in pre-trained transformer representations, Yu and Ettinger (2020) report that these representations reflect word content of phrases, but don't show signs of more sophisticated humanlike composition beyond word content. In the present paper we perform a direct follow-up of that study, asking whether models will show better evidence of composition after fine-tuning on tasks that are good candidates for requiring composition: 1) the Quora Question Pairs dataset in Paraphrase Adversaries from Word Scrambling (PAWS-QQP) (Zhang et al., 2019a), an adversarial paraphrase dataset forcing models to classify paraphrases with high lexical overlap, and 2) the Stanford Sentiment Treebank (Socher et al., 2013), a sentiment dataset with fine-grained phrase labels to promote composition. We base our analysis on the tests proposed by Yu and Ettinger (2020), which rely on alignment with human judgments of phrase pair similarities, and which leverage control of lexical overlap to target compositionality. We fine-tune and evaluate the same models and representation types tested in that paper, for optimal comparison.

We find that across the board, fine-tuning on PAWS-QQP does not improve compositionality—if anything, performance on composition metrics tends to degrade. Composition performance also remains low after training on SST, but we do see some localized improvements for certain models. Analyzing the PAWS-QQP dataset, we find reliable superficial cues to paraphrase labels (distance of word swap), explaining in part why fine-tuning on that task might fail to improve composition—and reinforcing the need for caution in interpreting difficulty of NLP tasks. We also discuss the contribution of variation in size of labeled phrases in SST, with respect to the benefits that result from fine-tuning on that task. All experimental code and

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2279–2293
August 1–6, 2021.

data are made available for further testing.[1]

2 Related work

Extensive work has studied the nature of learned representations in NLP models (Adi et al., 2016; Conneau et al., 2018; Ettinger et al., 2016; Durrani et al., 2020). Our work builds in particular on analysis of contextualized representations (Bacon and Regier, 2019; Tenney et al., 2019; Peters et al., 2018; Hewitt and Manning, 2019; Klafka and Ettinger, 2020; Toshniwal et al., 2020). Other work that has focused on transformers, as we do, has often focused on analyzing the attention mechanism (Vig and Belinkov, 2019; Clark et al., 2019), learned parameters (Roberts et al., 2020; Radford et al., 2019; Raffel et al., 2020) and redundancy (Dalvi et al., 2020; Voita et al., 2019; Michel et al., 2019). The evaluation that we use here follows the paradigm of classification-based probing (Kim et al., 2019; Wang et al., 2018; Zhang et al., 2019b; Yang et al., 2019a) and correlation with similarity judgments (Finkelstein et al., 2001; Gerz et al., 2016; Hill et al., 2015; Conneau and Kiela, 2018).

The current paper also builds on work subjecting trained NLP models to adversarial inputs, to highlight model weaknesses. One body of work approaches the problem by applying heuristic rules of perturbation to input sequences (Wallace et al., 2019; Jia and Liang, 2017; Zhang et al., 2019a), while another uses neural models to construct adversarial examples (Li et al., 2020, 2018) or manipulate inputs in embedding space (Jin et al., 2020). Our work also contributes to efforts to understand impacts and outcomes of the fine-tuning process (Miaschi et al., 2020; Mosbach et al., 2020; Wang et al., 2020; Perez-Mayos et al., 2021).

Phrase and sentence composition has drawn frequent attention in analysis of neural models, often focusing on analysis of internal representations and downstream task behavior (Ettinger et al., 2018; Conneau et al., 2019; Nandakumar et al., 2019; McCoy et al., 2019; Yu and Ettinger, 2020; Bhathena et al., 2020; Mu and Andreas, 2020; Andreas, 2019). Some work investigates compositionality via constructing linguistic (Keysers et al., 2019) and non-linguistic (Liška et al., 2018; Hupkes et al., 2018; Baan et al., 2019) synthetic datasets.

Most related to our work here is the finding of Yu and Ettinger (2020). They test for composition in two-word phrase representations from transformers, via similarity correlations and paraphrase detection. They find that baseline performance on these tasks is high, but once they control for amount of word overlap, performance drops dramatically, suggesting that observed correspondences rely on word content rather than phrase composition. We build directly on this work, testing whether these patterns will still hold after fine-tuning on tasks intended to encourage composition.

3 Fine-tuning Pre-trained Transformers

In response to the weaknesses observed by Yu and Ettinger (2020), we select two different datasets with promising characteristics for addressing these weaknesses. We fine-tune on these tasks, then perform layer-wise testing on contextualized representations from the fine-tuned models, comparing against results on the pre-trained models. Here we describe the two fine-tuning datasets.

3.1 PAWS: fine-tuning on high word overlap

The core of the Yu and Ettinger (2020) finding is that model performance on the selected composition tests degrades significantly when cues of lexical overlap are controlled. It stands to reason, then, that a model trained to discern meaning differences under conditions of high lexical overlap may improve on these overlap-controlled composition tests. This drives our selection of the Paraphrase Adversaries from Word Scrambling (PAWS) dataset (Zhang et al., 2019b), which consists of sentence pairs with high lexical overlap. The task is formulated as binary classification of whether two sentences are paraphrases or not. State-of-the-art models achieve only $< 40\%$ accuracy before training on the dataset (Zhang et al., 2019a). Table 1 shows examples from this dataset. Due to the high lexical overlap, we might expect that in order to achieve non-trivial accuracy on this task, models must attend to more sophisticated meaning information than simple word content.

3.2 SST: fine-tuning on hierarchical labels

Another dataset that has been associated with training and evaluation of phrasal composition is the Stanford Sentiment Treebank, which contains syntactic phrases of various lengths, together with fine-grained human-annotated sentiment labels for these phrases. Because this dataset contains annotations

---

[1] Datasets and code available at https://github.com/yulang/fine-tuning-and-composition-in-transformers

Sentence 1	Sentence 2	Label
There are also specific discussions , public profile debates and project discussions .	There are also public discussions , profile specific discussions , and project discussions .	0
She worked and lived in Stuttgart , Berlin (Germany) and in Vienna (Austria) .	She worked and lived in Germany (Stuttgart , Berlin) and in Vienna (Austria) .	1

Table 1: Example pairs from PAWS-QQP. Both positive and negative pairs have high bag-of-words overlap.

of composed phrases of various sizes, we can reasonably expect that training on this dataset may foster an increased sensitivity to compositional phrase meaning. We formulate the fine-tuning task as a 5-class classification task following the setup in Socher et al. (2013). The models are trained to predict sentiment labels given phrases as input.

4 Representation evaluation

For optimal comparison of the effects of fine-tuning on the above tasks, we replicate the tests, representation types, and models reported on by Yu and Ettinger. Here we briefly describe these methods. For more details on the evaluation dataset and task setup, please refer to Yu and Ettinger (2020).

4.1 Evaluation tasks

Yu and Ettinger propose two analyses for measuring composition: similarity correlations and paraphrase classification. They focus on two-word phrases, using the BiRD bigram relatedness dataset (Asaadi et al., 2019) for similarity correlations, and the PPDB 2.0 paraphrase database (Ganitkevitch et al., 2013; Pavlick et al., 2015) for paraphrase classification. BiRD contains 3,345 bigram pairs, with source phrases paired with numerous target phrases, and human-annotated similarity scores ranging from 0 to 1. For similarity correlation, Yu and Ettinger take layer-wise correlations between these human phrase similarity scores and the cosine similarities of model representations for the same phrases. For paraphrase classification, Yu and Ettinger train a multi-layer perceptron classifier to label whether two phrase representations are paraphrases, drawing their positive phrase pairs from PPDB 2.0—which contains paraphrases with scores generated by a regression model—and randomly sampling negative pairs from the rest of the dataset. We replicate all of these procedures.

For both task types, Yu and Ettinger compare between "uncontrolled" and "controlled" tests, with the latter filtering the data to control word overlap within phrase pairs, such that amount of word overlap between two phrases can no longer be used as a cue for how similar the meanings are. It is on these controlled settings that Yu and Ettinger observe the significant drop in performance, suggesting that model representations lack the compositional knowledge to discern phrase meaning beyond word content. Below we will report results for both settings, with particular focus on controlled settings.

4.2 Representation types

Following Yu and Ettinger, for each input phrase we test as a potential representation 1) CLS token, 2) average of tokens within the phrase (Avg-Phrase), 3) average of all input tokens (Avg-All), 4) embedding of the second word of the phrase, intended to approximate the semantic head (Head-Word), and 5) SEP token. We test each of these representations at every layer of each model.[2]

5 Experimental setup

We fine-tune and analyze the same models that Yu and Ettinger test in pre-trained form: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019), XLNet (Yang et al., 2019b) and XLM-RoBERTa (Conneau et al., 2019). In each case, the pre-trained "base" version is used as the starting point for fine-tuning. We use the implementation of Wolf et al. (2019)[3] based on PyTorch (Paszke et al., 2019).

We fine-tune these models on the two datasets described in Section 3. The Quora Question Pairs dataset in Paraphrase Adversaries from Word Scrambling (PAWS-QQP)[4] consists of a training set with 11,988 sentence pairs, and a dev/test set with 677 sentence pairs. Tuning on PAWS-QQP is formulated as binary classification. Sentences are passed as input and models are trained to pre-

---

[2] Like Yu and Ettinger, we also test both phrase-only input (encoder input consists only of two-word phrase plus special CLS/SEP tokens), as well as inputs in which phrases are embedded in sentence contexts.

[3] `https://github.com/huggingface/transformers`

[4] `https://github.com/google-research-datasets/paws`

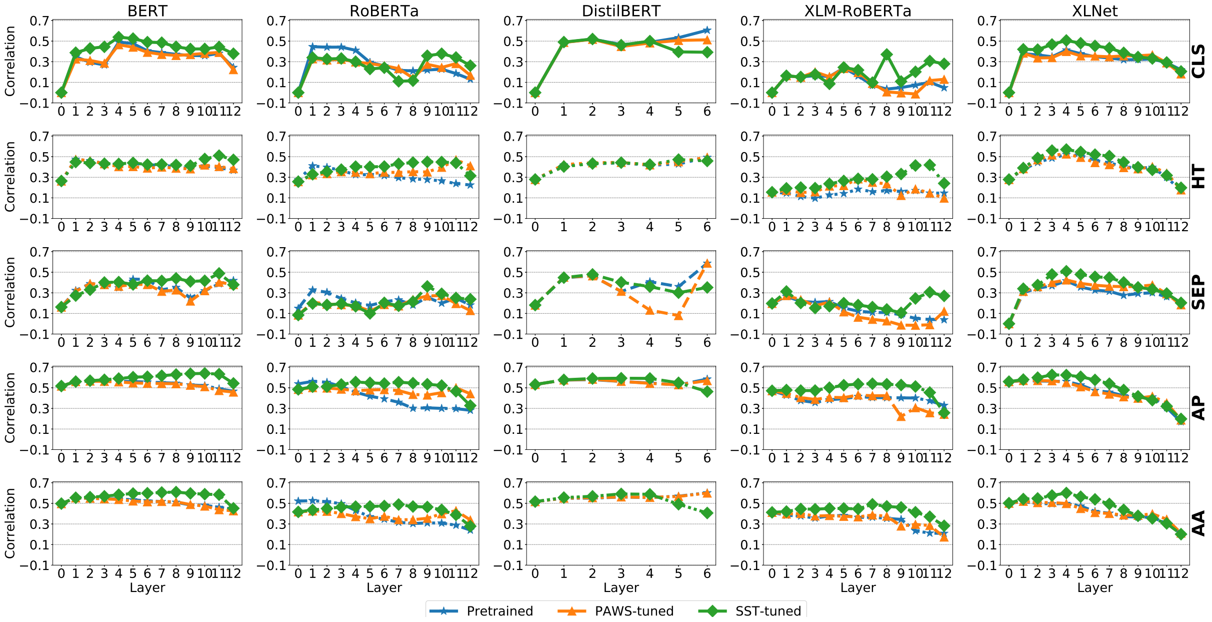

Figure 1: Similarity correlation on uncontrolled BiRD dataset, with phrase-only input. Columns correspond to models, and rows correspond to representation types ("HT" = Head-token, "AP" = Avg-Phrase and "AA" = Avg-All). For each model and representation type, the corresponding subplot shows correlations for pre-trained, PAWS-tuned and SST-tuned settings, respectively. For each subplot, X-axis corresponds to layer index, and Y-axis corresponds to correlation value. Layer 0 corresponds to input embeddings passed to the model.

dict whether the input sentences are paraphrases or not. Models are trained on the training set, and validated on the dev/test set for convergence.

The Stanford Sentiment Treebank (SST)[5] (Socher et al., 2013) contains 215,154 phrases. 15% of the data is reserved for validation. The fine-tuning task is formulated as 5-class classification on sentiment labels, where models are given phrases as input, and asked to predict sentiment. In both tasks, we use the Adam optimizer (Kingma and Ba, 2014) with default weight decay. We train the models until convergence on the validation set.

The evaluation tasks consist of correlation analysis and paraphrase classification. For correlation in the uncontrolled setting, we use the complete BiRD dataset, containing 3,345 phrase pairs.[6] For the controlled test, we filter the complete dataset following the criteria in Yu and Ettinger (2020), resulting in 410 "AB-BA" mirror-image pairs with 100% word overlap (e.g., *law school* / *school law*). For the classification tasks, we use the preprocessed data released by Yu and Ettinger (2020).[7] We collect 12,036 source-target phrase pairs from the preprocessed dataset for our uncontrolled classification setting, and for the controlled classification setting, we collect 11,772 phrase pairs with exactly 50% word overlap in each pair, following the procedure from the original paper.

6 Results after fine-tuning

6.1 Full datasets

Figure 1 presents the original results from Yu and Ettinger (2020) on pre-trained models, alongside our new results after fine-tuning, on the full BiRD dataset. Since this is prior to the control of word overlap, these correlations can be expected to reflect effects of lexical content encoding, without yet having isolated effects of composition. We find that after fine-tuning on SST, most models and representation types show small improvements in peak correlations across layers, while fine-tuning on PAWS also yields improvements in peak correlations—albeit even smaller—in models other than BERT and XLM-RoBERTa. Overall, within a given representation type, improvements are generally stronger after fine-tuning on SST than on PAWS. Between representation types, Avg-Phrase and Avg-All remain consistently at the highest correlations after fine-tuning. Additionally, we see that the decline in correlation at later layers in pre-trained BERT, RoBERTa and XLM-RoBERTa is mitigated

---

[5] `https://nlp.stanford.edu/sentiment/treebank.html`

[6] `http://saifmohammad.com/WebPages/BiRD.html`

[7] `https://github.com/yulang/phrasal-composition-in-transformers`

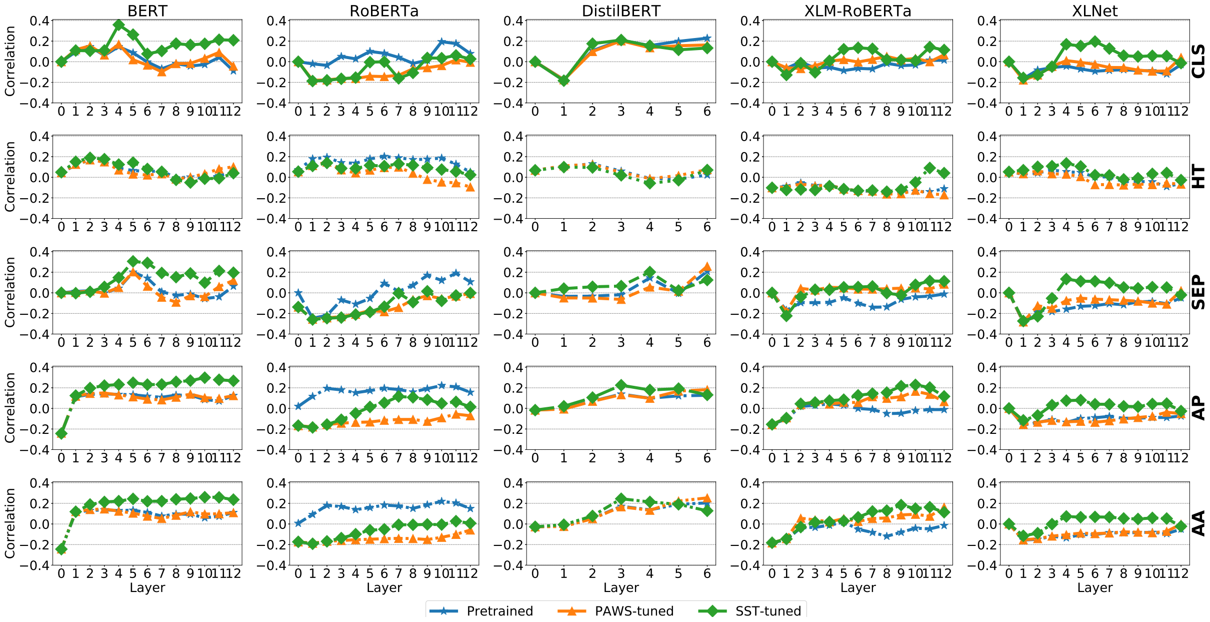

Figure 2: Similarity correlation on controlled BiRD dataset (AB-BA setting), with phrase-only input.

after fine-tuning. Model-wise, we see the most significant improvements in the RoBERTa model, for which the correlations become more consistent across layers for most representation types. As we discuss below, we take this as indication that the fine-tuning promotes more robust retention of word content information across layers, if not more robust phrasal composition.

For the sake of space, we present the plots of the uncontrolled paraphrase classification setting in Figure 7 of the Appendix. The overall improvements are even smaller than those seen in the correlations, but we do see comparable patterns in these paraphrase classification results, in particular with SST showing slightly stronger benefits than PAWS.

6.2 Controlled datasets

Above we see small benefits of fine-tuning for performance on the full, uncontrolled datasets. However, the critical question for our purposes is whether correlations also show improvements in word-overlap controlled settings, which better isolate effects of composition. Figure 2 shows correlations for all models on the controlled AB-BA (full word overlap) correlation test. Figure 3 shows the results for the controlled paraphrase classification setting, where both paraphrase and non-paraphrase pairs have exactly 50% word overlap.

The first comparison to note is between original and controlled settings, which allows us to establish the contributions of overlap information as opposed to composition. Comparing between Figure 1 and Figure 2, it is clear that fine-tuned models still show substantial reduction in correlation when overlap cues are removed. The same goes for Figure 3 (by comparison to Figure 7 of the Appendix)—we see that on the controlled dataset, accuracies hover just above chance-level performance both before and after fine-tuning, compared to over 90% accuracy on the uncontrolled dataset. This gap in performance between the original and controlled datasets mirrors the findings of Yu and Ettinger (2020), and suggests that even after fine-tuning, the majority of correspondence between model phrase representations and human meaning similarity judgments can be attributed to capturing of word content information rather than phrasal composition.

The second key comparison is between pre-trained and fine-tuned models within the overlap-controlled settings. While the prior comparison tells us that similarity correspondence is still dominated by word content effects, this second comparison can tell us whether fine-tuning shows at least some boost in meaning composition relative to pre-training. Comparing performance of pre-trained and fine-tuned models in Figure 2, we see that fine-tuning on PAWS-QQP actually slightly degrades correlations at many layers for a majority of models and representation types—with improvements largely restricted to XLM-RoBERTa and XLNet (perhaps notably, mostly in cases where pre-trained correlations are negative). This is despite the fact that models achieve strong validation performance on PAWS-QQP (as shown in Table 2), suggesting

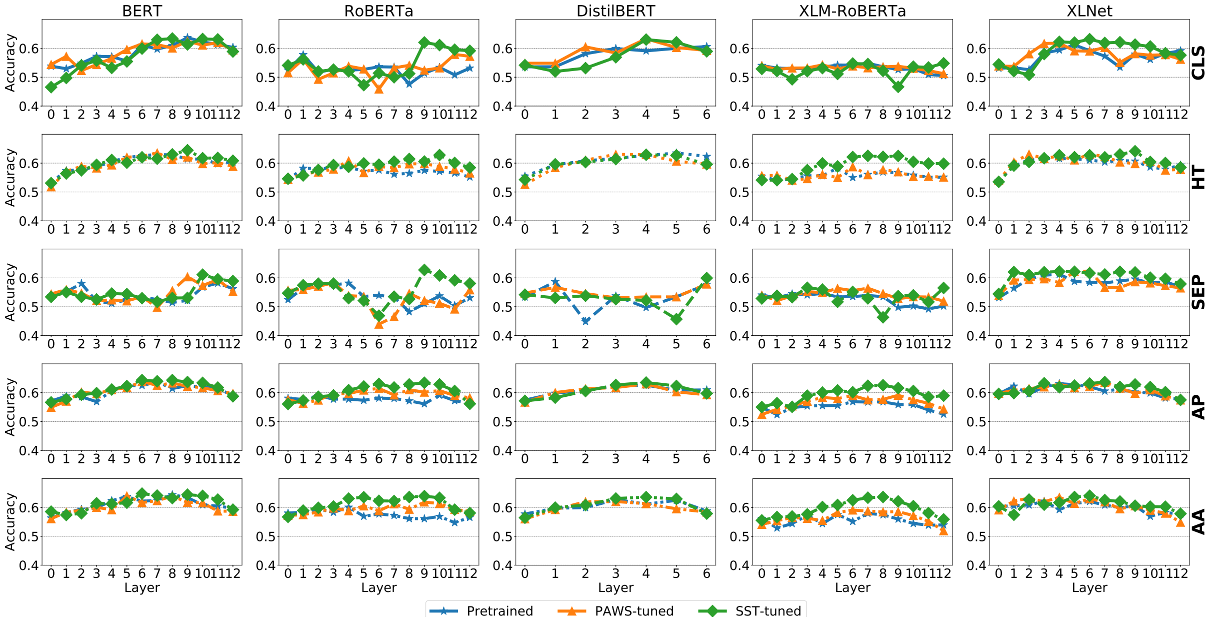

Figure 3: Paraphrase classification accuracy on controlled PPDB dataset (50% word overlap setting) with phrase-only input. Y-axis range is smaller relative to Figure 7, to make changes from pre-training more visible.

that learning this task does little to improve composition. We will explore the reasons for this below.

In Figure 3, we see that fine-tuning also does little to improve paraphrase classification accuracies in the controlled setting—though each model shows slight improvement in peak accuracy across layers and representation types (e.g., RoBERTa shows ∼3% increase in peak accuracy with SST tuning, and 2% with PAWS tuning). Even so, the best accuracies across models continue to be only marginally above chance. This, too, fails to provide evidence of any substantial composition improvement resulting from the fine-tuning process.

The story changes slightly when we turn to impacts of SST fine-tuning on correlations in Figure 2. While all correlations remain low after SST fine-tuning, we do see that correlations for BERT, XLM-RoBERTa and XLNet show some non-trivial benefits even in the controlled setting. In particular, SST tuning consistently improves correlation among all representation types in BERT (except for minor degradation in later layers for Head-token), boosting the highest correlation from ∼0.2 to ∼0.39. Between representation types, the greatest change is in the CLS token, with the most dramatic point of improvement being an abrupt correlation peak for CLS at BERT's fourth layer. We will discuss more below about this localized benefit.

A final important observation is that fine-tuning on either dataset produces clear degradation in correlations for all representation types in RoBERTa under the controlled setting, by contrast to the general improvements seen for that and other models in the uncontrolled setting. This suggests that at least for that model, fine-tuning encourages retention or enhancement of lexical information, but may degrade compositional phrase information.[8]

Model	Accuracy(%)
BERT	80.13
RoBERTa	90.81
DistilBERT	81.98
XLM-RoBERTa	91.18
XLNet	88.24
Linear CLF	71.34

Table 2: Accuracy of fine-tuned models on PAWS-QQP dev/test set. Linear CLF is a baseline classifier with relative swapping distance as the only input feature.

7 Analyzing impact of fine-tuning

The presented results suggest that despite compelling reasons to think that fine-tuning on the selected tasks may improve composition of phrase meaning, these models mostly do not exhibit noteworthy benefits from fine-tuning. In particular, fine-

---

[8] Following Yu and Ettinger (2020), in addition to phrase-only inputs we also try embedding target phrases in sentence contexts. Consistent with the findings of Yu and Ettinger (2020), we see that presence of context words does boost overall correlation and accuracy, but does not alter the general trends. Moreover, models still show relatively weak performance on controlled tasks even with context available (see Figure 8 and Figure 9 in the Appendix for details).

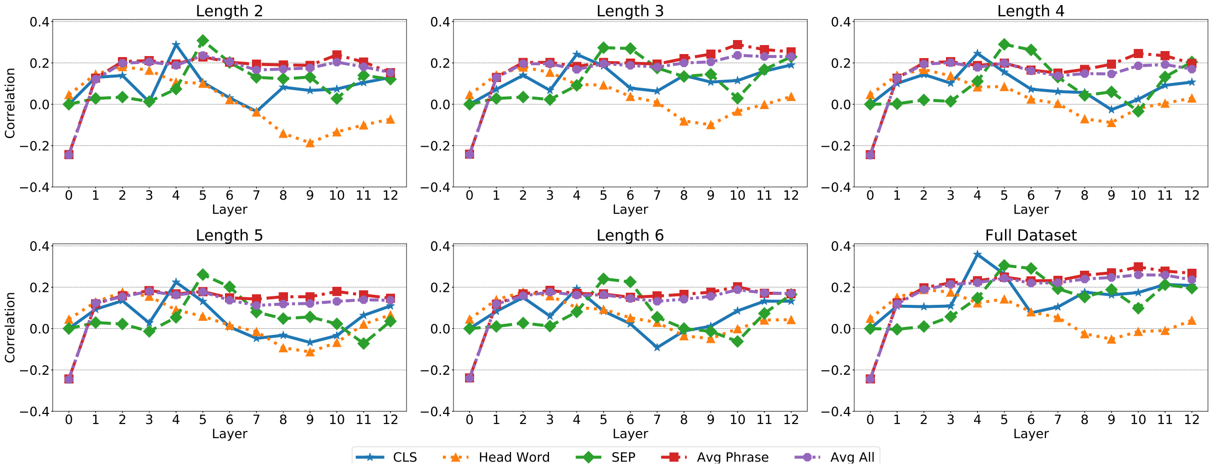

Figure 4: Layer-wise correlation of BERT fine-tuned on phrases of different lengths in SST.

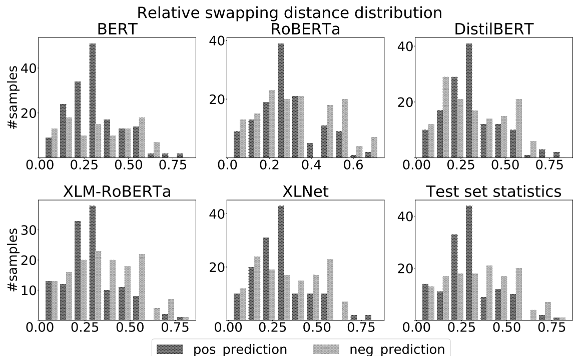

Figure 5: Distribution of positive and negative predictions made by tuned models. Last plot shows the statistics in the PAWS-QQP dev/test set. X-axis corresponds to relative swapping distance; Y-axis shows number of samples in the specific relative swapping distance bin.

tuning on the PAWS-QQP dataset often degrades performance on the controlled datasets taken to be most indicative of compositionality. As for SST, the benefits are minimal, but in localized cases like BERT's CLS token, we do see signs of improved compositionality. In this section, we conduct further analysis on the impacts of fine-tuning, and discuss why tuned models behave as they do.

7.1 Failure of PAWS-QQP

Table 2 shows accuracy of fine-tuned models on the dev/test set of PAWS-QQP.[9] It is clear that the models are learning to perform well on this dataset, but our results above indicate that this does not translate to improved composition sensitivity.

We explore the possibility that this discrepancy may be caused by trivial cues arising during the construction of the dataset, enabling models to infer paraphrase labels without needing to improve their understanding of the meaning of the sentence pair (c.f., Poliak et al., 2018; Gururangan et al., 2018). Sentence pairs in PAWS are generated via word swapping and back translation to ensure high bag-of-words overlap (Zhang et al., 2019a). We hypothesize that models may be able to achieve high performance in this task based on distance of the word swap alone, without requiring any sophisticated representation of sentence meaning.

To test this, given a sentence pair (s_1, s_2) with word counts l_1, l_2, respectively, we define "relative swapping distance" as

$$dist_{relative} = \frac{dist_{swap}}{max(l_1, l_2)}$$

where $dist_{swap}$ is defined as the index difference of the first swapping word in s_1 and s_2. For the example shown in the first row of Table 1, the first swapping word is "specific", with $dist_{swap} = 4$. Note that with this measure we focus on information from one word swap only, while some pairs in PAWS-QQP have multiple swapped words—so in reality, swapping distance information may be even stronger than our results below indicate.

In the last plot of Figure 5, we show an association between relative swapping distance and paraphrase labels in the PAWS dev/test set: sentence pairs with small swapping distance tend to be positive samples, while large swapping distance associates with negative labels. The other plots in Figure 5 show distribution of positive and negative predictions generated by each fine-tuned model with respect to relative swapping distance. We see a similar pattern, with models tending to generate negative labels when swapping distance is larger.

---

[9] The performance of BERT in the table is different from previous work mainly due to the fact that models in Zhang et al. (2019a) are tuned on concatenation of QQP and PAWS-QQP datasets rather than PAWS-QQP only.

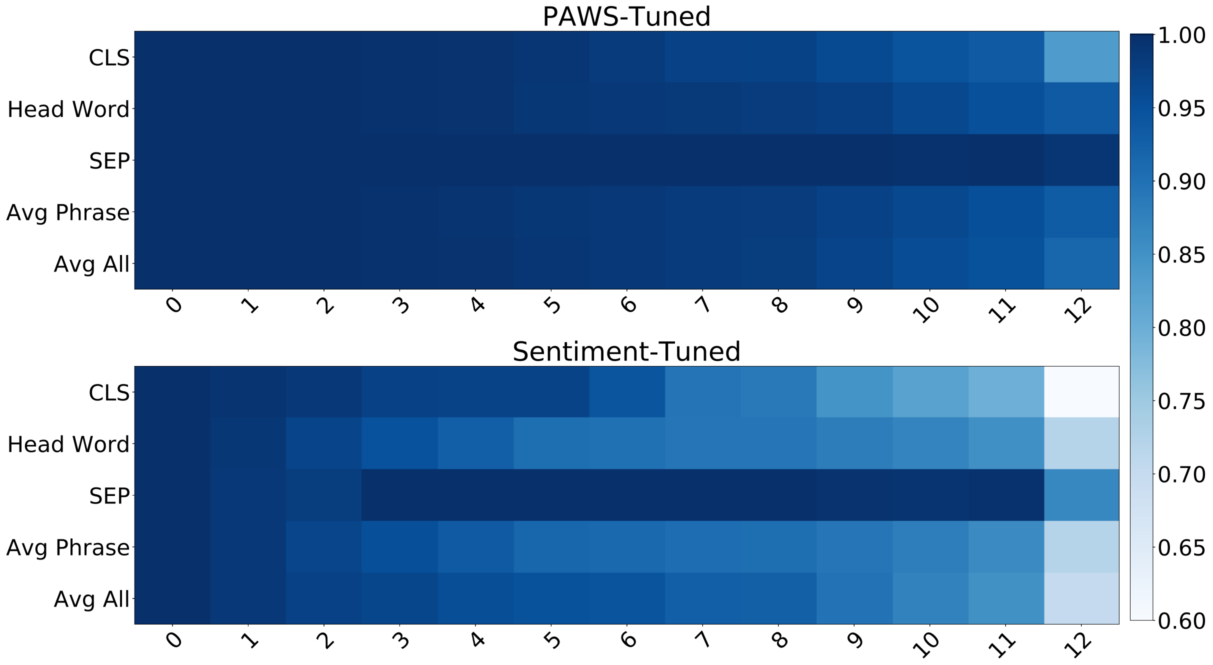

Figure 6: Average layer-wise embedding similarity between fine-tuned and pre-trained BERT. The upper half shows the comparison between PAWS-QQP tuned and pre-trained BERT. And the lower half presents Sentiment Treebank-tuned v.s. pre-trained. Embeddings are evaluated using full BiRD dataset for input.

To verify the viability of this cue, we train a simple linear classifier on PAWS-QQP, with relative swapping distance as the only input feature. The results are reported as "Linear CLF" in Table 2. Even without access to the content of the sentences, we see that this simple model is able to achieve nontrivial and comparably good classification accuracy on the dev/test set. The strong performance of the linear classifier and the distribution of predictions are consistent with the hypothesis that when we tune on PAWS-QQP, rather than forcing models to learn nuanced meaning in the absence of word overlap cues, we may instead encourage models to focus on lower-level information having little to do with the sentence meaning, further degrading their performance on the composition tasks.

7.2 Localized impacts of SST

Fine-tuning on sentiment shows a bit of a different pattern—while it mostly shows only minor changes from pre-training, and the correlations and classification accuracies remain at decidedly low levels on the controlled settings, we do see in certain models some distinctive changes in levels of similarity correlation as a result of tuning on SST. Notably, since these improvement patterns are seen in the similarity correlations but not in the classification accuracies, this suggests that these two tasks are picking up on slightly different aspects of phrasal compositionality. To investigate these effects further, we focus our attention on BERT, which shows the most distinctive improvement in correlations.

The obvious candidate for the source of the localized SST benefit is the dataset's inclusion of labeled syntactic phrases of various sizes. The benefits seen from SST tuning suggest that this may indeed encourage models to gain some finer-grained sensitivity to compositional impacts of phrase structure (at least those relevant for sentiment). To examine this further, we filter the SST dataset to subsets with phrases of the same length, from 2 to 6 words, and tune pre-trained BERT on each subset.

Figure 4 shows the correlations for BERT, fine-tuned on each phrase length, on the overlap-controlled BiRD dataset. We see that tuning on the full dataset (mixed phrase lengths) gives the strongest fourth-layer boost in CLS correlation performance—but among the size subsets, a semblance of the fourth-layer CLS peak is seen across phrase lengths, with length-2 training yielding the strongest peak, and length-6 training the smallest. This suggests an amount of size-based specialization—sentiment training on phrases of (or closer to) two words has more positive impact on similarity correlations for our two-word phrases.[10] However, we also see that phrases of

---

[10] Although subset size can potentially contribute to correlation performance, we find that subset size does not correlate

other sizes contribute non-trivially to the ultimate correlation improvement observed from training on the full dataset. This is consistent with the notion that training on diverse phrase sizes encourages fine-grained attention to compositionality, while training on phrases of similar size may have slightly more direct benefit.

Representation changes For further comparison of fine-tuning effects between tasks, we analyze changes in BERT representations at each layer before and after the fine-tuning process. Figure 6 shows the average layer-wise representation similarity between fine-tuned and pre-trained BERT given identical input. We see substantial differences between tasks in terms of representation changes: while SST fine-tuning produces significant changes across representations and layers, PAWS fine-tuning leaves representations largely unchanged (further supporting the notion that this task can be solved fairly trivially). We also see that after SST tuning, BERT's CLS token shows robust similarity to pre-trained representations until the fifth layer, followed by a rapid drop in similarity. This suggests that the fourth-layer correlation peak may be enabled in part by retention of key information from pre-training, combined with heightened phrase sensitivity from fine-tuning. We leave in-depth exploration of this dynamic for future work.

8 Discussion

The results of our experiments indicate that despite the promise of PAWS-QQP and SST tasks for improving models' phrasal composition, fine-tuning on these tasks falls far short of resolving the composition weaknesses observed by Yu and Ettinger (2020). The majority of correspondence with human judgments can still be attributed to word overlap effects—disappearing once overlap is controlled—and improvements on the controlled settings are absent, very small, or highly localized to particular models, layers and representations. This outcome aligns with the increasing body of evidence that NLP datasets often do not require of models the level of linguistic sophistication that we might hope for—and in particular, our identification of a strong spurious cue in the PAWS-QQP dataset adds to the growing number of findings emphasizing that NLP datasets often have artifacts that can inflate performance (Poliak et al., 2018; Gururangan et al., 2018; Kaushik and Lipton, 2018).

We do see a ray of promise in the small, localized benefits for certain models from tuning on SST. These improvements do not extend to all models, and are fairly small in the models that do see benefits—but as we discuss above, it appears that training on fine-grained syntactic phrase distinctions may indeed confer some enhancement of compositional meaning in phrase representations—at least when model conditions are amenable. Since sentiment information constitutes only a very limited aspect of phrase meaning, we anticipate that training on fine-grained phrase labels that reflect richer and more diverse meaning information could be a promising direction for promoting composition more robustly in these models.

9 Conclusions and future directions

We have tested effects of fine-tuning on phrase meaning composition in transformer representations. Although we select tasks with promise to address composition weaknesses and reliance on word overlap, we find that representations in the fine-tuned models show little improvement on controlled composition tests, or show only very localized improvements. Follow-up analyses suggest that the PAWS-QQP dataset contains spurious cues that undermine learning of sophisticated meaning properties when training on that task. However, results from SST tuning suggest that training on labeled phrases of various sizes could prove effective for learning composition. Future work should investigate how model properties interact with fine-tuning to produce improvements in particular models and layers—and should move toward phrase-level training with meaning-rich annotations, which we predict will be a promising direction for improving models' phrase meaning composition.

Acknowledgments

We would like to thank three anonymous reviewers for valuable feedback for improving this paper. We also thank members of the University of Chicago CompLing Lab for helpful comments and suggestions on this work. This material is based upon work supported by the National Science Foundation under Award No. 1941160.

---

with the performance patterns we observe here. Phrase count of each subset: length 2 - 11,499; length 3 - 11,779; length 4 - 15,050; length 5 - 11,816; length 6 - 9,935.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Jacob Andreas. 2019. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*.

Shima Asaadi, Saif Mohammad, and Svetlana Kiritchenko. 2019. Big bird: A large, fine-grained, bigram relatedness dataset for examining semantic composition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 505–516.

Joris Baan, Jana Leible, Mitja Nikolaus, David Rau, Dennis Ulmer, Tim Baumgärtner, Dieuwke Hupkes, and Elia Bruni. 2019. On the realization of compositionality in neural networks. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 127–137.

Geoff Bacon and Terry Regier. 2019. Does bert agree? evaluating knowledge of structure dependence through agreement relations. *arXiv preprint arXiv:1908.09892*.

Hanoz Bhathena, Angelica Willis, and Nathan Dass. 2020. Evaluating compositionality of sentence representation models. *ACL 2020*, page 185.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $ &!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 107–112.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Dieuwke Hupkes, Anand Singh, Kris Korrel, German Kruszewski, and Elia Bruni. 2018. Learning compositionally through attentive guidance. *arXiv preprint arXiv:1805.09657*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. *arXiv preprint arXiv:1808.04926*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105.

Najoung Kim, Roma Patelφ, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019. Probing what different nlp tasks teach machines about function word comprehension. *NAACL HLT 2019*, page 235.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Josef Klafka and Allyson Ettinger. 2020. Spying on your neighbors: Fine-grained probing of contextual embeddings for information about surrounding words. *arXiv preprint arXiv:2005.01810*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Adam Liška, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional rnn in a haystack. *arXiv preprint arXiv:1802.06467*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

Alessio Miaschi, Dominique Brunato, Felice Dell'Orletta, and Giulia Venturi. 2020. Linguistic profiling of a neural language model. *arXiv preprint arXiv:2010.01869*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Marius Mosbach, Anna Khokhlova, Michael A Hedderich, and Dietrich Klakow. 2020. On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers. *arXiv preprint arXiv:2010.02616*.

Jesse Mu and Jacob Andreas. 2020. Compositional explanations of neurons. *arXiv preprint arXiv:2006.14032*.

Navnita Nandakumar, Timothy Baldwin, and Bahar Salehi. 2019. How well do embedding models capture non-compositionality? a view from multiword expressions. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 27–34.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. *arXiv preprint arXiv:1907.07355*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca

Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430.

Laura Perez-Mayos, Roberto Carlini, Miguel Ballesteros, and Leo Wanner. 2021. On the evolution of syntactic information encoded by bert's contextualized representations.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. *NAACL HLT 2018*, page 180.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.*

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019*.

Shubham Toshniwal, Haoyue Shi, Bowen Shi, Lingyu Gao, Karen Livescu, and Kevin Gimpel. 2020. A cross-task analysis of text span representations. *arXiv preprint arXiv:2006.03866*.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Chengyu Wang, Minghui Qiu, Jun Huang, and Xiaofeng He. 2020. Meta fine-tuning neural language models for multi-domain text mining. *arXiv preprint arXiv:2003.13003*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019a. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In *Proc. of EMNLP*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in* *neural information processing systems*, pages 5754–5764.

Lang Yu and Allyson Ettinger. 2020. Assessing phrasal representation and composition in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4896–4907.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019a. Paws: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019b. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.

A Appendix

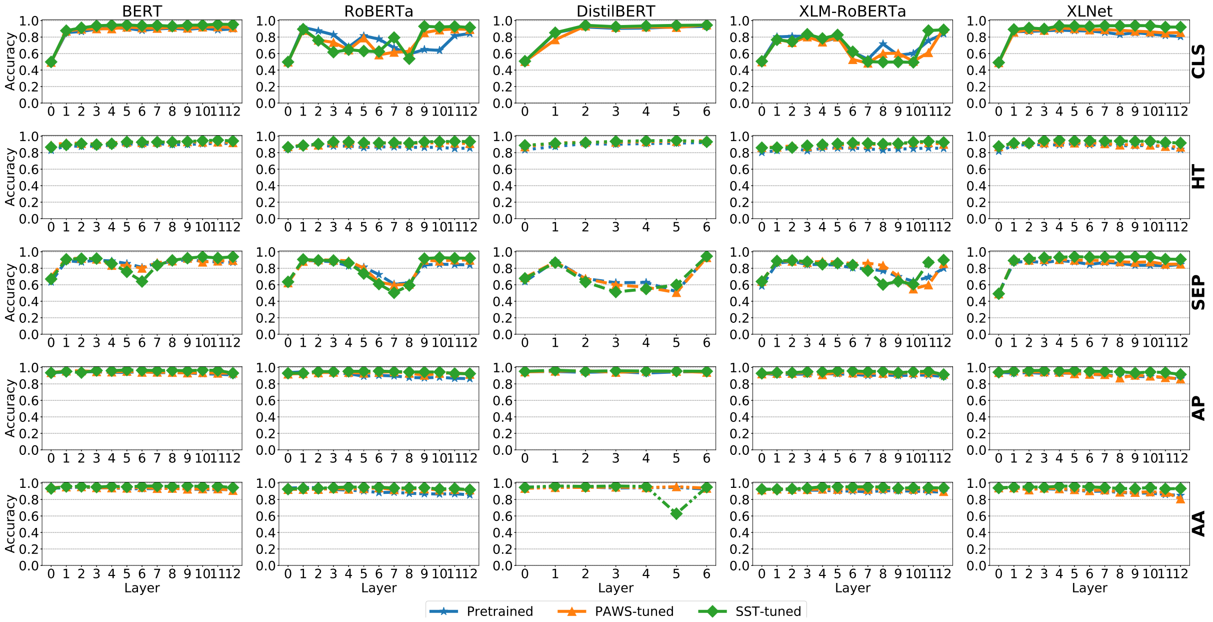

Figure 7: Paraphrase classification accuracy on uncontrolled PPDB dataset, with phrase-only input. Columns correspond to models, and rows correspond to representation types ("HT" = Head-Token, "AP" = Avg-Phrase and "AA" = Avg-All). For each model and representation type, the corresponding subplot shows accuracies for pre-trained, PAWS-tuned and SST-tuned settings, respectively. For each subplot, X-axis corresponds to layer index, and Y-axis corresponds to accuracy value. Layer 0 corresponds to input embeddings passed to the model.

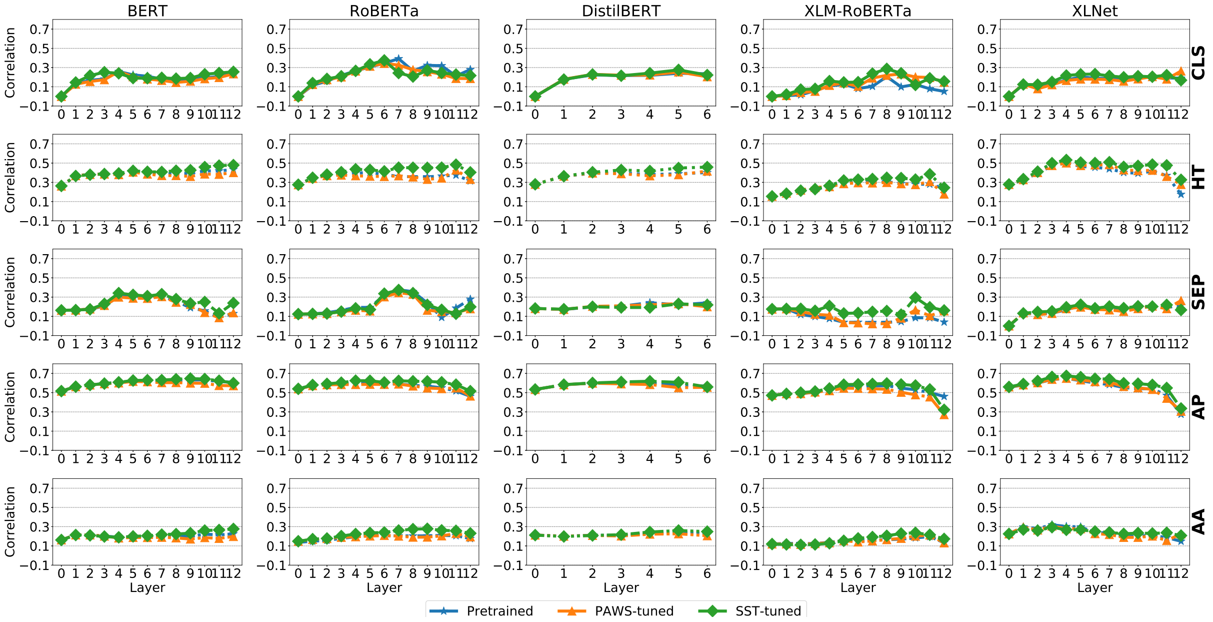

Figure 8: Similarity correlation on full BiRD dataset with phrases embedded in context sentence (context-available input).

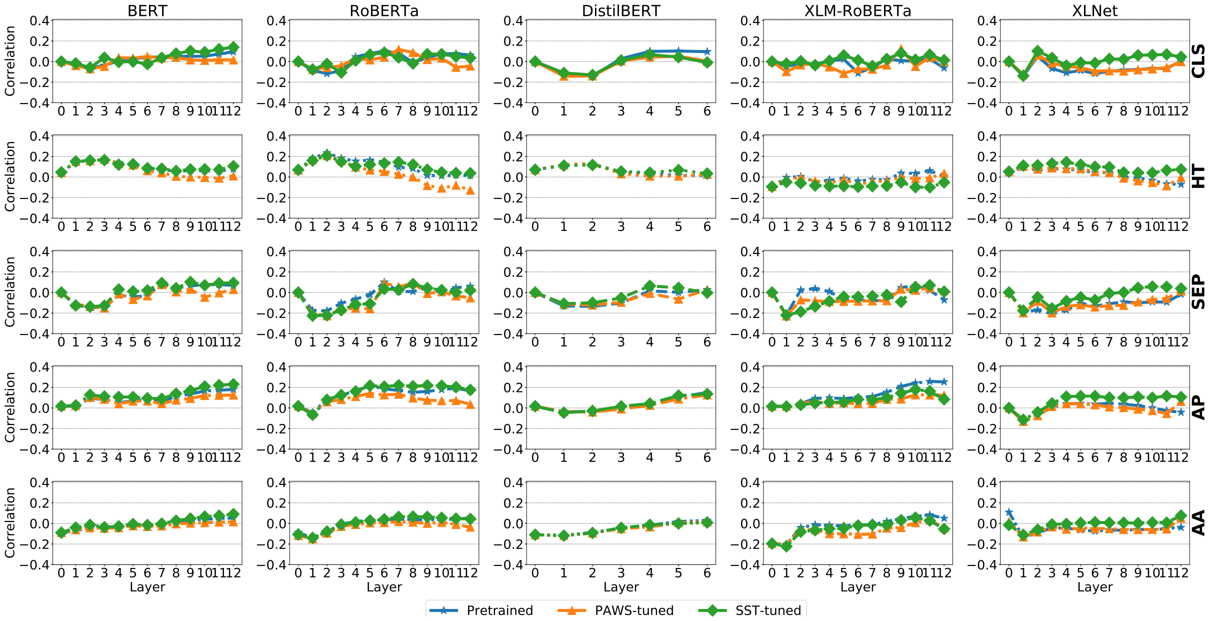

Figure 9: Similarity correlation on controlled BiRD dataset (AB-BA setting) with phrases embedded in context sentence (context-available input).

Lifelong Learning of Topics and Domain-Specific Word Embeddings

Xiaorui Qin, Yuyin Lu, Yufu Chen, Yanghui Rao[*]
School of Computer Science and Engineering,
Sun Yat-sen University, Guangzhou, China
`qinxr5@mail2.sysu.edu.cn, luyy37@mail2.sysu.edu.cn,`
`chenyf66@mail2.sysu.edu.cn, raoyangh@mail.sysu.edu.cn`

Abstract

Lifelong topic models mainly focus on in-domain text streams in which each chunk only contains documents from a single domain. To overcome data diversity of the in-domain corpus, most of the existing methods exploit the information from limited sources in a separate and heuristic manner. In this study, we develop a lifelong collaborative model (LCM) based on non-negative matrix factorization to accurately learn topics and domain-specific word embeddings. LCM particularly investigates: (1) developing a knowledge graph based on the semantic relationships among words in the lifelong learning process, so as to accumulate global context information discovered by topic models and local context information reflected by context word embeddings from previous domains, and (2) developing a subword graph based on byte pair encoding and pairwise word relationships to exploit subword information of words in the current in-domain corpus. To the best of our knowledge, we are the first to collaboratively learn topics and word embeddings via lifelong learning. Experiments on real-world in-domain text streams validate the effectiveness of our method.

1 Introduction

Lifelong learning (Silver, 2011; Mitchell et al., 2015), which accumulates and maintains the past knowledge to help future learning in an endless manner, has attracted considerable attention in topic modeling (Chen et al., 2020b; Gupta et al., 2020). Most lifelong topic models (Chen and Liu, 2014b; Chen, 2015; Wang et al., 2016) focus on the corpus that only contains text from a single domain, dubbed the in-domain corpus (Xu et al., 2018). This is because in-domain corpora are widespread in real-world applications, such as breaking news and tweets related to a specific topic (domain). The key to the success of a lifelong topic model within in-domain corpora is based on a precondition that prior topical information of previous domains can be fully exploited to guide meaningful learning in the new coming domain (Chen and Liu, 2014b). However, because the in-domain corpus is typically of limited size (Xu et al., 2018), it is insufficient for the existing methods to train coherent topics.

To alleviate the lack of global context in a corpus, one simple solution for topic models is to incorporate general-purpose pre-trained word embeddings (Das et al., 2015; Xun et al., 2016, 2017b; Dieng et al., 2020). Although the general-purpose embeddings can provide some useful information for words within the in-domain corpus, their embedding representations may not be ideal for the target domain and in some cases they may even conflict with the meanings of the words in the task domain because words often have multiple senses or meanings (Xu et al., 2018). Another solution trains topics and word embeddings jointly in the one-shot learning scenario (Xun et al., 2017a; Dieng et al., 2020). Such a unified method prevents relying on the external embedding corpus that is not always closely aligned with the domain task, because the model can learn domain-specific word embeddings by itself. Unfortunately, the aforementioned models are conducted on collected documents without the guidance of any prior knowledge. Besides, they all treat words as atomic units, which may not perform well on the in-domain corpus with relatively few words.

In light of these considerations, we aim to generate coherent topics and domain-specific word embeddings jointly by a lifelong process. On the one hand, domain-specific word embeddings tend to offer more accurate complementary information to lifelong topic modeling than pre-trained embeddings. On the other hand, we alleviate the lack of global and local context information within in-

---

[*] The corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2294–2309
August 1–6, 2021.

domain corpora by exploiting subwords (Pinter et al., 2017). Both the topical and subword information are leveraged in our knowledge-based learner to generate better domain-specific word embeddings. To achieve this, we propose a lifelong collaborative model (LCM)[1] by coordinating global context, local context, and subword information. First, our LCM maintains a knowledge graph based on word relationships to accumulate the past knowledge learned from previous domains, which exploits from both the global word-document matrix and the local word co-occurrence matrix. Second, we develop a subword graph from the current in-domain corpus to capture extra information of words. We use non-negative matrix factorization (NMF) as our framework, which is an effective method of mining latent text semantics with great flexibility in transforming prior knowledge into regulations (Lee and Seung, 1999; Chen et al., 2015, 2020b) and it gives sparseness to matrices with interpretability (Hoyer, 2004). The main contributions of this study can be summarized as follows:

- We propose a lifelong learning method to jointly generate topics and word embeddings over in-domain text streams. To the best of our knowledge, we are the first to collaboratively learn topics and domain-specific word embeddings through a lifelong process.
- We incorporate local context information and subword information into lifelong topic modeling, which can alleviate the lack of global context information when the target dataset is relatively small.
- In lifelong word embedding learning, we leverage the topical and subword information to help generate better domain-specific embeddings for down-stream learning tasks.

2 Related Work

Topic modeling (Deerwester et al., 1990; Hofmann, 1999; Blei et al., 2003) and word embedding learning (Mikolov et al., 2013a,b) are two of the most important tasks in natural language processing. The former task aims to discover the latent semantic structure of documents based on the global context, while the latter one follows the distributional hypothesis that words occurring in similar local contexts tend to have similar syntactic and semantic properties (Harris, 1954). The traditional topic and word embedding learning models are based on isolated learning, i.e., a one-shot task learning, thus they lack ability to continually learn from incrementally available data.

Lifelong Topic Modeling. A lifelong topic model (Chen and Liu, 2014b; Chen, 2015; Wang et al., 2016), as a typical example of lifelong machine learning, is gaining more and more research interests than traditional one-shot deal that conducts a topic model on collected documents just for once (Chen et al., 2020b). Lifelong topic models inherit three key characteristics in lifelong machine learning, i.e., continuous tasks, knowledge accumulation and maintenance, and a knowledge-based learner that can leverage the past knowledge to help future learning in a never-ending manner. Furthermore, lifelong topic modeling is mainly applied to in-domain corpora where each chunk only contains text from a single domain.

Based on NMF, Chen et al. (2020b) proposed a lifelong topic model named NMF-LTM. However, the above method only considers the most important 10 words under every topic while ignores other non-top words, i.e., most words in the vocabulary. This problem will be more serious if the vocabulary size is large. Besides, NMF-LTM may perform poorly for other downstream tasks, because it can only capture the information of limited words in sentences. Finally, NMF-LTM only mines word relationships from the perspective of global (topical) information, which is inadequate within in-domain corpora. Considering the limited global context in the new coming corpus, Gupta et al. (2020) incorporated general-purpose pre-trained word embeddings as complementary to topics into the knowledge base for lifelong learning. Unfortunately, the above method required that the dimension of word embeddings being equal to the number of topics and each dimension of word embeddings corresponding to a topic. This violates the complementary but different points of view, i.e., the global viewpoint and the local viewpoint, for topic models and word embedding models (Xun et al., 2017a). This model optimizes a topic-word matrix, in which each row represents the word distribution of a topic and each column represents the embedding of a word. Each dimension of the word embedding learned by this model implied the possibility of the word occurring in the corresponding topic. However, word embeddings contain many other features that cannot be captured by global (context) information, e.g., the syntactic feature.

---

[1] `https://github.com/XiaoruiQ/LCM`

Word Embedding Learning. Lifelong learning has also been adopted to train domain-specific word embeddings, which fills the gap between general-purpose embeddings trained on large-scale corpora and the topic (domain) of the down-stream task. For example, Xu et al. (2018) first developed a meta-learner to expand the new in-domain corpus by measuring the content similarity of past domains and the new domain. Then, they generated word embeddings for the new domain using the combined data. However, this method only considered the local context information from past domains, which is inadequate to capture the polysemous nature of words. As an illustration, *apple* is one of polysemous words that is topically contextualized by several domains, i.e., product line, operating system, and fruit (Gupta et al., 2020).

3 Lifelong Collaborative Model

In this section, we detail the proposed LCM for jointly learning topics and domain-specific word embeddings in a lifelong process. The topical information and local context from previous domains, and a subword graph constructed from the current in-domain corpus are exploited in LCM to guide future tasks.

3.1 Problem Formalization

Given a stream of document chunks $\left\{\mathcal{DOC}_t\right\}_{t=1}^{T}$ accumulated in an endless manner ($T = +\infty$), we aim to jointly generate topics and domain-specific word embeddings when each chunk only contains text from a single domain. At any time point, our LCM deals with the current document chunk, e.g., $\mathcal{DOC}_t$, by leveraging the past knowledge learned from the previous document chunks, i.e., $\mathcal{DOC}_1$, $\mathcal{DOC}_2$, ..., $\mathcal{DOC}_{t-1}$. Table 1 lists the notations used in this paper. We use bold uppercase letters such as $\boldsymbol{D}_t$ to represent matrices, regular uppercase letters such as M to represent scalar constants, and regular lowercase letters such as λ_v to represent scalar variables.

Notation	Description
$\boldsymbol{D}_t \in R^{M \times N}$	Word-document matrix at the current moment
$\boldsymbol{U}_t \in R^{M \times K}$	Word-topic matrix at the current moment
$\boldsymbol{V}_t \in R^{K \times N}$	Topic-document matrix at the current moment
$\boldsymbol{X}_t \in R^{M \times M}$	Word co-occurrence matrix at the current moment
$\boldsymbol{B}_t \in R^{M \times E}$	Word embedding matrix at the current moment
$\boldsymbol{C}_t \in R^{M \times E}$	Context word embedding matrix at the current moment
M	The number of words
N	The number of documents
K	The number of topics
E	The dimension of word embeddings

Table 1: Frequently used notations.

3.2 Objective Function

Figure 1 illustrates the architecture of our LCM, which processes in-domain text streams through a knowledge-based learner. Formally, the objective function of LCM is defined as follows:

$$\begin{aligned} L = &\|\boldsymbol{D}_t - \boldsymbol{U}_t\boldsymbol{V}_t\|_F^2 + \left\|\boldsymbol{X}_t - \boldsymbol{B}_t\boldsymbol{C}_t^T\right\|_F^2 \\ &+ \Upsilon(\boldsymbol{V}_t) + \Psi(\boldsymbol{U}_t) + \Phi(\boldsymbol{C}_t) + \Omega(\boldsymbol{B}_t), \\ &s.t. \quad \boldsymbol{U}_t \geq 0, \boldsymbol{V}_t \geq 0, \boldsymbol{B}_t \geq 0, \boldsymbol{C}_t \geq 0. \quad (1) \end{aligned}$$

It is noteworthy that we constrain the non-negativity of $\boldsymbol{B}_t$ and $\boldsymbol{C}_t$ to learn sparse interpretable word embeddings (Murphy et al., 2012; Luo et al., 2015), so as to capture the polysemous nature of words (Panigrahi et al., 2019). With non-negativity constraints, words are represented by limited dimensions (Murphy et al., 2012). All words that have positive values under specific dimensions may share a common characteristic, which enhances the interpretability of word embeddings and helps capture the polysemous nature.

The first term of our objective function aims to factorize the global word-document matrix $\boldsymbol{D}_t$ into the word-topic matrix $\boldsymbol{U}_t$ and the topic-document matrix $\boldsymbol{V}_t$, and the interpretability of $\boldsymbol{U}_t$ and $\boldsymbol{V}_t$ is ensured by their non-negativity. For the local context information, Levy and Goldberg (2014) have proved that the Skip-Gram model with negative sampling (SGNS) is implicitly factorizing a positive pointwise mutual information word co-occurrence matrix shifted by a constant offset. Accordingly, we use the shifted positive pointwise mutual information matrix as our word co-occurrence matrix $\boldsymbol{X}_t$ and decompose it into the word embedding matrix $\boldsymbol{B}_t$ and the context word embedding matrix $\boldsymbol{C}_t$, as presented in the second term. Given a hyperparameter λ_v, the sparsity constraint on $\boldsymbol{V}_t$ is introduced as the third term $\Upsilon(\boldsymbol{V}_t) = \lambda_v \|\boldsymbol{V}_t\|_1$. This ensures that each document covers limited topics (Chen et al., 2020b). The sparsity of topics encourages interpretable topics (Card et al., 2018), which corresponds with the tuition that a document usually focuses on several salient topics instead of covering a wide variety of topics (Lin et al., 2019). Although NMF has given sparseness to V_t, a more direct control over such properties of the representation is still needed (Hoyer, 2004). The rest terms $\Psi(\boldsymbol{U}_t)$, $\Phi(\boldsymbol{C}_t)$, and $\Omega(\boldsymbol{B}_t)$ are the constraints on matrices $\boldsymbol{U}_t$, $\boldsymbol{C}_t$, and $\boldsymbol{B}_t$, which will be described in sections 3.2.3-3.2.5, respectively.

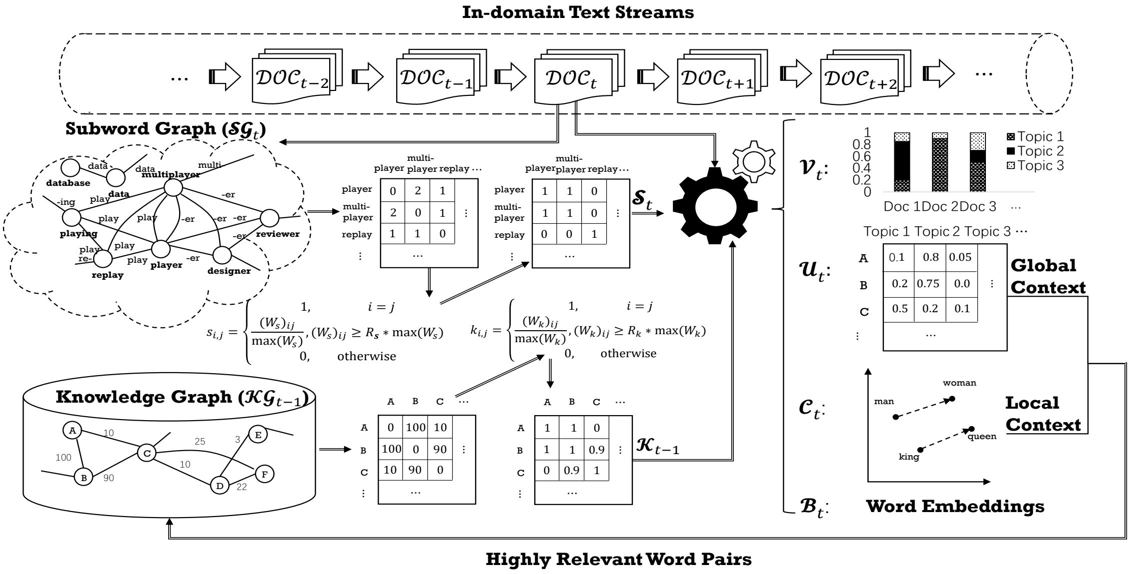

Figure 1: Lifelong Collaborative Model (LCM)

3.2.1 Knowledge Graph ($\mathcal{KG}$)

LCM uses relationships between words as the representations of our $\mathcal{KG}$ to maintain knowledge of past domains to help with the current in-domain task. $\mathcal{KG}$ accumulates the knowledge of past domains from two sources of information, i.e., the global context information mined by topic models and the local context information reflected by context word embeddings. As shown in Figure 1, the output of LCM contains the word-topic matrix $\boldsymbol{U}_t$, the topic-document matrix $\boldsymbol{V}_t$, the word embedding matrix $\boldsymbol{B}_t$, and the context word embedding matrix $\boldsymbol{C}_t$. $\mathcal{KG}$ fuels the global context and local context information with the help of $\boldsymbol{U}_t$ and $\boldsymbol{C}_t$, as follows.

For the global context information, we use the inner product to measure similarities between topic distributions of words in $\boldsymbol{U}_t$. For each word w_i in the current vocabulary of $\boldsymbol{D}_t$, we find $topT$ words w_j $(j = 1, 2, ..., T)$, whose topic distributions of all topics are most similar to w_i. Each w_j and w_i are seen as word pairs that reflect the relationship from the global context information, i.e., the topical information. After finding $topT$ related words of each word, all the word pairs are accumulated and de-duplicated. Following (Chen et al., 2020b), we set the weight of each word pair (w_i, w_j) to 1.

Regarding the local context information, we use the inner product to measure similarities between context word embeddings of words in $\boldsymbol{C}_t$. For each word w_i in the current vocabulary of $\boldsymbol{D}_t$, we find $topT$ words whose context word embeddings are most similar to this word. All word pairs represent the relationship from the perspective of local context information and their weights are set to η after de-duplication, where η adjusts the weight relationship between global context information and local context information.

Then, we accumulate the word pairs from these two sources to fuel the information of global context and local context. It is worth noting that if a word pair (w_i, w_j) appears simultaneously in the two kinds of word pairs, its weight is recorded as $1 + \eta$. De-duplication is not required here, because the relationship between two words related in both global context and local context is closer than that of two words only related in one kind of source information. We use $\mathcal{J}_t$ to denote all the related word pairs in the current in-domain corpus, and $\mathcal{J}_t$ is defined as:

$$\mathcal{J}_t = \{(W_k)_{ij}\}, \tag{2}$$

where $(W_k)_{ij}$ represents the weight of the word pair (w_i, w_j) in $\mathcal{KG}$. Finally, $\mathcal{KG}$ is updated as follows:

$$\mathcal{KG}_t = \mathcal{KG}_{t-1} + \mathcal{J}_t. \tag{3}$$

3.2.2 Subword Graph ($\mathcal{SG}$)

To incorporate subword information, LCM uses a subword graph ($\mathcal{SG}$) to store relationships between words in $\boldsymbol{D}_t$ from the perspective of subword information. The motivation of introducing $\mathcal{SG}$ is to capture more information from the structures of words themselves as the complement for global and local contexts (Bojanowski et al., 2017; Pinter et al., 2017). Many in-domain text streams contain a high proportion of rare words with low word frequencies, e.g., proper nouns in specific domains, which cannot be adequately reflected by context due to the low frequencies. Note that $\mathcal{SG}$ only mines the subword information of the current in-domain corpus since subwords are only related to domain-independent structures of words.

A typical word in English is composed by three kinds of subword units, i.e., the word root, the prefix, and the suffix[2]. Word roots and prefixes determine the meaning of words, while suffixes determine the syntactic-related part of speech. We adopt byte pair encoding (BPE) (Sennrich et al., 2016), which can implicitly match these morpheme boundaries, to conduct subword segmentation. We also compare this segmentation method with character n-gram features (Bojanowski et al., 2017) in experiments. For every word pair, the number of shared subword units between them are recorded as the weight. In the current in-domain corpus, $\mathcal{SG}_t$ is defined as:

$$\mathcal{SG}_t = \{(W_s)_{ij}\}, \tag{4}$$

where $(W_s)_{ij}$ represents the weight of the word pair (w_i, w_j) in $\mathcal{SG}_t$.

3.2.3 Constraint on U_t

Before introducing $\Psi(\boldsymbol{U}_t)$, we first construct the word-word relationship matrix $\boldsymbol{K}_{t-1} \in R^{M\times M}$ from $\mathcal{KG}_{t-1}$ to represent the closeness of relationships between words. In $\mathcal{KG}_{t-1}$, we select all of the word pairs in which both of the two words occurred in the current vocabulary of $\boldsymbol{D}_t$. Only these words contribute to the current in-domain task on $\boldsymbol{D}_t$, and all diagonal elements of $\boldsymbol{K}_{t-1}$ are 1. R_k, which represents the threshold ratio for $\mathcal{KG}$, is used to select the "close" relationships between words. For two words w_i and w_j, if the corresponding pair (w_i, w_j) occurred in the selected word pairs in $\boldsymbol{D}_t$ mentioned above, the value of k_{ij} will be determined by the threshold ratio R_k. If the weight of (w_i, w_j), i.e., $(W_k)_{ij}$, is greater than or equal to the max weight of all the word pairs in $\boldsymbol{D}_t$ multiplied by R_k, then $k_{ij} = \frac{(W_k)_{ij}}{max(W_k)}$. If it is less than the max weight multiplied by R_k or w_i and w_j are not connected in $\mathcal{KG}_{t-1}$, then $k_{ij} = 0$. In the above, R_k helps to select word pairs with relatively large weights. Although wrong connections between some word pairs are kept in our $\mathcal{KG}$, the weights of them cannot be large enough, because the max weight of $\mathcal{KG}$ becomes larger and larger over time. These pairs will not be chosen to participate in constraints of matrices. In summary, $\boldsymbol{K}_{t-1}$ is calculated as follows:

$$k_{ij} = \begin{cases} 1, & i = j \\ \dfrac{(W_k)_{ij}}{max(W_k)}, & (W_k)_{ij} \geq R_k max(W_k) \\ 0, & otherwise. \end{cases} \tag{5}$$

The word-word relationship regularization based on $\mathcal{KG}$ for $\Psi(\boldsymbol{U}_t)$ holds that the topic distributions of words that are closely related in $\mathcal{KG}$ would be more similar than those have no connection in $\mathcal{KG}$. We use the Graph Laplacian (Dai et al., 2020) as the first part of $\Psi(\boldsymbol{U}_t)$ to depict that under each topic in $\boldsymbol{U}_t$, the closer two words are connected in $\mathcal{KG}$, the closer their probabilities are. The second part of $\Psi(\boldsymbol{U}_t)$ is the diversity regularization to reduce the overlapping of topics, i.e., to improve the topic uniqueness (Nan et al., 2019). Accordingly, $\Psi(\boldsymbol{U}_t)$ is defined as follows:

$$\begin{aligned}\Psi(\boldsymbol{U}_t) =& \lambda_{u1} tr(\boldsymbol{U}_t^T \boldsymbol{H}_{t-1} \boldsymbol{U}_t) \\ &+ \lambda_{u2} \left\| \boldsymbol{U}_t^T \boldsymbol{U}_t - \boldsymbol{I}_K \right\|_F^2 .\end{aligned} \tag{6}$$

In the above, λ_{u1} and λ_{u2} are hyperparameters. $\boldsymbol{H}_{t-1} = diag(\boldsymbol{K}_{t-1} \cdot \mathbf{1}) - \boldsymbol{K}_{t-1}$ represents the Graph Laplacian of $\boldsymbol{K}_{t-1}$, where $\mathbf{1}$ represents a column vector in which all of the elements are 1, and $diag(\boldsymbol{K}_{t-1} \cdot \mathbf{1})$ represents the matrix with the vector $\boldsymbol{K}_{t-1} \cdot \mathbf{1}$ as diagonal elements. $\boldsymbol{I}_K$ is an identity matrix of order $K \times K$.

3.2.4 Constraint on C_t

$\mathcal{KG}$, which fuels the information of global context and local context from previous domains, is constructed with the help of $\boldsymbol{U}_t$ and $\boldsymbol{C}_t$. It also contributes to both the two matrices on their constraints. For $\Phi(\boldsymbol{C}_t)$, we use $\boldsymbol{K}_{t-1}$ to introduce the word-word relationship regularization. It depicts that context embeddings of words that are closely related in $\mathcal{KG}$ would be more similar than those have less connection in $\mathcal{KG}$. Specifically, under each dimension in $\boldsymbol{C}_t$, the closer two words are connected in $\mathcal{KG}$, the closer their representations are. $\Phi(\boldsymbol{C}_t)$ is calculated as follows:

$$\Phi(\boldsymbol{C}_t) = \lambda_c tr(\boldsymbol{C}_t^T \boldsymbol{H}_{t-1} \boldsymbol{C}_t), \tag{7}$$

where λ_c is a hyperparameter.

---

[2] `https://en.wikipedia.org/wiki/Root_(linguistics)`

3.2.5 Constraint on $\boldsymbol{B}_t$

The first part of $\Omega(\boldsymbol{B}_t)$ is similar to the word-word relationship regularization for $\Psi(\boldsymbol{U}_t)$ and $\Phi(\boldsymbol{C}_t)$. It holds that embeddings of words that are closely related in $\mathcal{SG}$ would be more similar than those have less connection in $\mathcal{SG}$. We also use the Graph Laplacian to depict that under each dimension in $\boldsymbol{B}_t$, the closer two words are connected in $\mathcal{SG}$, the closer their representations are. A word-word relationship matrix $\boldsymbol{S}_t \in R^{M \times M}$ is constructed from $\mathcal{SG}_t$, as follows:

$$s_{ij} = \begin{cases} 1, & i = j \\ \dfrac{(W_s)_{ij}}{max(W_s)}, & (W_s)_{ij} \geq R_s max(W_s) \\ 0, & otherwise, \end{cases} \tag{8}$$

where $(W_s)_{ij}$ represents the weight of pair (w_i, w_j) in SG_t, and R_s denotes the threshold ratio of SG. R_s helps exclude and ignore wrong connections in $\mathcal{SG}$. In addition, the cooperation of $\mathcal{KG}$ and $\mathcal{SG}$ can further reduce the influence of unimportant edges. For example, a connection that is only selected by R_k may not be as important as the connection that is selected by both of R_k and R_s simultaneously.

The second part is a sparsity constraint on $\boldsymbol{B}_t$, which depicts that each word only has features of a limited number, because we aim to learn sparse representations so that the generated domain-specific word embeddings are more interpretable.

Finally, $\Omega(\boldsymbol{B}_t)$ is defined as follows:

$$\Omega(\boldsymbol{B}_t) = \lambda_{b1} tr(\boldsymbol{B}_t^T \boldsymbol{N}_t \boldsymbol{B}_t) + \lambda_{b2} \left\| \boldsymbol{B}_t \right\|_1, \tag{9}$$

where λ_{b1} and λ_{b2} are hyperparameters. $\boldsymbol{N}_t = diag(\boldsymbol{S}_t \cdot \mathbf{1}) - \boldsymbol{S}_t$ is the Graph Laplacian of $\boldsymbol{S}_t$.

3.3 Alternately Iterative Algorithm

We develop an alternately iterative algorithm to achieve a good compromise between ease of implementation and speed. Take $\boldsymbol{B}_t$ as an example, we first calculate the derivative of the objective function L on $\boldsymbol{B}_t$ as follows:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{B}_t} = & -2\boldsymbol{X}_t\boldsymbol{C}_t + 2\boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t \\ & + 2\lambda_{b1} diag(\boldsymbol{S}_t \cdot \mathbf{1})\boldsymbol{B}_t - 2\lambda_{b1}\boldsymbol{S}_t\boldsymbol{B}_t \\ & + \lambda_{b2} \cdot \mathbf{1} \cdot \mathbf{1}^T. \end{aligned} \tag{10}$$

Based on the derivative of L on $\boldsymbol{B}_t$, the updating rule for $\boldsymbol{B}_t$ is given below:

$$\boldsymbol{B}_t \leftarrow \boldsymbol{B}_t \circ \frac{\boldsymbol{X}_t\boldsymbol{C}_t + \lambda_{b1}\boldsymbol{S}_t\boldsymbol{B}_t}{\boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t + J(\boldsymbol{B}_t)}, \tag{11}$$

where $J(\boldsymbol{B}_t) = \lambda_{b1} diag(\boldsymbol{S}_t \cdot \mathbf{1})\boldsymbol{B}_t + \frac{\lambda_{b2}}{2} \cdot \mathbf{1} \cdot \mathbf{1}^T$. Note that $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, $\boldsymbol{C}_t$, and $\boldsymbol{B}_t$ always satisfy the non-negativity because they are updated in this multiplication form. Due to the limited space, we provide the updating rules for matrices $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, and $\boldsymbol{C}_t$, the parameter inference process, the theoretical proof of the algorithmic convergence, and the time complexity analysis in Appendices A-D.

3.4 Model Scalability

Our model has a good scalability due to the "divide-and-conquer" strategy. First, we partition the large corpus into several small document chunks that belong to different domains, and we only decompose matrices of one chunk at any time. Second, we use sparse matrices to store $\mathcal{KG}$ and $\mathcal{SG}$, thus they are scalable and can be processed fast. Third, when facing a large single domain corpus in text streams, we can partition it into small sub-domain corpora and process one corpus at each time.

4 Experiments

4.1 Dataset

We evaluate our LCM on the real-world Amazon Review dataset[3] (McAuley et al., 2015; He and McAuley, 2016) from 28 departments (i.e., the first-level category). Following (Xu et al., 2018), we consider all the reviews under each second-level category as a domain. Each domain has several third-level categories, which will be used for all down-stream tasks and model evaluation. We randomly select 9 in-domain corpora to carry out experiments. Table 2 summarizes the characteristics of the selected 9 corpora, i.e., domain names, numbers of reviews and labels, the average text lengths, and vocabulary sizes. The selected corpora are preprocessed by eliminating stop-words and words with frequency (in the total reviews from 9 domains) lower than 15. Also, reviews with less than 20 words are removed.

Note that we choose the above dataset instead of other datasets for lifelong topic modeling (Gupta et al., 2020) since we focus on in-domain corpora with rare overlaps, in which, documents share few common information. For completeness, we also shuffle these 7 training domains randomly and show the results under different permutations in Appendix E.

---

[3] `http://snap.stanford.edu/data/web-Amazon.html`

Domain Name	#Docs	#Labels	Length	Vocab
Safety Signs & Signals	11,298	8	32.12	16,090
Pet Behavior Center	7,450	5	43.34	14,301
SIM Cards & Prepaid Minutes	13,872	4	42.29	16,726
Horses	13,571	11	35.98	19,608
Video	10,088	5	76.35	23,196
Keyboards & MIDI	15,585	3	59.09	24,703
Characters & Series	16,663	6	67.77	35,393
Science Education	7,975	13	38.05	18,107
Cult Movies	15,252	13	83.25	47,668

Table 2: Dataset statistics.

4.2 Experimental Setting

We simulate an endless lifelong learning process using the 9 corpora. Word-word relationships in the knowledge graph are gradually accumulated on the first 7 domains. With the help of this knowledge graph, we conduct experiments on the last 2 domains and compare the model performance on both lifelong topic modeling and domain-specific word embedding learning.

We randomly select 5% reviews from "Cult Movies (CM)" for validation. To build the knowledge graph for the validation set, 5% reviews from the first 7 corpora are sampled and reviews from these 8 small-scale in-domain corpora also construct a stream of document chunks. The idea of grid search (Fayed and Atiya, 2019) is used to select the best parameters for each metric. We provide our hyperparameters search space for grid search in Appendix F. The remaining 95% reviews of the first 7 domains construct the training set. All reviews from "Science Education (SE)" and the remaining 95% documents from CM are used as the testing set.

4.3 Baselines

For lifelong topic modeling, we compare our method with the following baselines: LDA-LTM[4] (Chen and Liu, 2014b), NMF-LTM (Chen et al., 2020b), and LNTM[5] (Gupta et al., 2020). For word embedding learning, we adopt FastText[6] (Bojanowski et al., 2017), L-DEM (Xu et al., 2018), SPINE[7] (Subramanian et al., 2018), and Word2Sense[8] (Panigrahi et al., 2019). To the best of our knowledge, L-DEM is the only work focuses on lifelong learning of domain-specific word embedding. Following (Xu et al., 2018), we train other baselines in two ways for evaluation, i.e., only on the new in-domain corpus, and on the total document set by fusing the new corpus and all corpora from the previous domains. We implement NMF-LTM and L-DEM by Python according to the original papers. For the sake of fairness, the parameters of all baselines[9] are selected on the validation set in the same experimental setting as LCM.

4.4 Evaluation Metrics

Suggested by (Lau et al., 2014; Chen and Liu, 2014a,b; Wang et al., 2016; Isonuma et al., 2020), we use the normalized pointwise mutual information (NPMI) (Aletras and Stevenson, 2013) score, which closely matches human judgments, to measure the coherence of representative words of topics generated by lifelong topic models. Following (Chen et al., 2020b), top 20 words of each topic are used for calculation. Considering that it is important to discover discriminative topics, we also adopt the topic uniqueness (TU) score (Nan et al., 2019) to measure the diversity of topics. In addition, the sparsity score of the document-topic distribution (TS-U) and the topic-word distribution (TS-V) proposed by Lin et al. (2019) is further used to measure the topic sparsity quantitatively. Particularly, we use 1e-20 as the threshold to count the number of zero values in document-topic and topic-word distributions. Only values that are smaller than 1e-20 can be set to zero. Although several studies (Chang et al., 2009; Newman et al., 2010) stated that the perplexity is unable to reflect the real semantic coherence of topics and even negatively correlated with human judgements, we show this metric of each model for completeness.

As domain-specific dictionaries are relatively small and meanwhile may contain some uncommon words, it is inappropriate to evaluate domain-specific word embeddings in traditional ways, e.g., calculating the word similarity. Following (Xu et al., 2018), we build a down-stream text classification task to evaluate domain-specific word embeddings generated by different models. The two testing sets, i.e., SE and CM, are used for text classification with their third-level categories as classification labels. For each review, we use the average embedding of all of the words as its feature vector to train a SVM classifier (Bayot and Gonçalves, 2016; Qin and Wang, 2009). We use

---

[4] `https://github.com/czyuan/LTM`

[5] `https://github.com/pgcool/Lifelong-Neural-Topic-Modeling`

[6] `https://github.com/bamtercelboo/cw2vec`

[7] `https://github.com/harsh19/SPINE`

[8] `https://github.com/abhishekpanigrahi1996/Word2Sense`

[9] The search scope of each parameter that is different with our method is obtained from the original papers.

Method	Science Education (SE)					Cult Movies (CM)				
	NPMI↑	TU↑	TS-U↑	TS-V↑	Perplexity↓	NPMI↑	TU↑	TS-U↑	TS-V↑	Perplexity↓
LDA-LTM	-0.0201	**0.6620**	0.0000	0.0000	9081.5853	0.0509	0.6060	0.0000	0.0000	14191.7899
LNTM	-0.3855	0.3610	0.0000	0.0000	9031.3161	-0.2072	0.4160	0.0000	0.0000	14057.3928
NMF-LTM	-0.0453	0.3940	0.2012	0.1725	**8564.4900**	0.0386	0.2660	0.1125	0.0099	**10870.6600**
LCM	**0.0012**	0.5940	**0.7782**	**0.5526**	8706.1200	**0.0633**	**0.6340**	**0.6781**	**0.4466**	13040.8700

Table 3: Performance comparison of lifelong topic models. For all metrics, "↓" after the metric indicates smaller is better while "↑" indicates larger is better. The best performance on each measure is highlighted by boldface.

accuracy to evaluate the effectiveness of word embeddings on the down-stream text classification task as in (Xu et al., 2018).

4.5 Result Comparison

4.5.1 Lifelong Topic Modeling

Table 3 shows the performance of different models on the lifelong topic discovery task, from which we can observe that LCM performs the best or the second best on each measure.

For the baselines, NMF-LTM achieves the best perplexity while almost the poorest TU. As mentioned in (Burkhardt and Kramer, 2019), there is a tradeoff between perplexity and TU in some cases, which means that models generating a lot of redundant topics may have a meaningless low perplexity. The reason of obtaining a low TU for NMF-LTM may be that it enforces documents within the same class would have more similar topic distributions, which is unsuitable to handle in-domain text streams since all documents in the in-domain corpus come from one class. This also influences its sparsity. For example, a document only has non-zero values for 10 topics while another document from the same class has non-zero values for another 10 topics. To get similar, they may both become non-zero for 20 topics. LNTM faces the same problem because it does not constrain the diversity among topics explicitly. The coherence score of LNTM is also unsatisfactory. A possible reason is that it treats word embeddings as topic distributions of words, which deteriorates the local semantic information captured by context word embeddings. LNTM entirely neglected the sparsity of document-topic and topic-word distributions, thus its TS-U and TS-V are zero.

LDA-LTM has a main difference with the other models, i.e., it does not construct the in-domain text stream based on time series, but fuses all of the previous domains together and accumulates the knowledge from this large corpus to help with the current in-domain corpus. Even though it is difficult to compare LDA-LTM with other lifelong topic models fairly, our LCM performs better than LDA-LTM in most cases, because we can exploit the information from local context and subwords.

The qualitative analysis of topics generated by these models is provided in Appendix G.

4.5.2 Word Embedding Learning

Figure 2 reports the accuracy of text classification using word embeddings with different dimensions. Note that sparse interpretable word embeddings always perform better when the embedding dimension is relatively large (Murphy et al., 2012). LCM performs the best in each testing set under each dimension, although some baselines (i.e., FastText, SPINE, and Word2Sense) are trained on the total 9 corpora and access more information without considering time series. L-DEM cannot learn high-quality embeddings because it needs a large amount of in-domain corpora to train its meta-learner, which is not accessible in most applications. Compared to FastText that incorporates subword information, our word embeddings perform better on classification in all cases. One possible reason is that the global context provides LCM with extra information. Two sparse word embedding models, i.e., SPINE and Word2Sense, overemphasize the sparsity while ignore the quality for downstream tasks. LCM balances well between sparsity and quality with the help of global context and subwords. We evaluate sparsity and interpretability of word embeddings in Appendix H.

For completeness, we also replace the SVM classifier with a neural network classifier (Chen et al., 2020a) consisting of 3 fully-connected layers. The results are shown in Appendix I.

4.5.3 Ablation Experiments

Take SE as an example, we report results of ablation experiments on NPMI, TU, and Accuracy in Table 4. "LCM-$\mathcal{KG}_{\mathcal{G}}$", "LCM-$\mathcal{KG}_{\mathcal{L}}$", and "LCM-$\mathcal{SG}$" represent LCM without the participation of global context in $\mathcal{KG}$, local context in $\mathcal{KG}$, and subwords in $\mathcal{SG}$, respectively. Deleting each part leads to performance degradation, which validates the

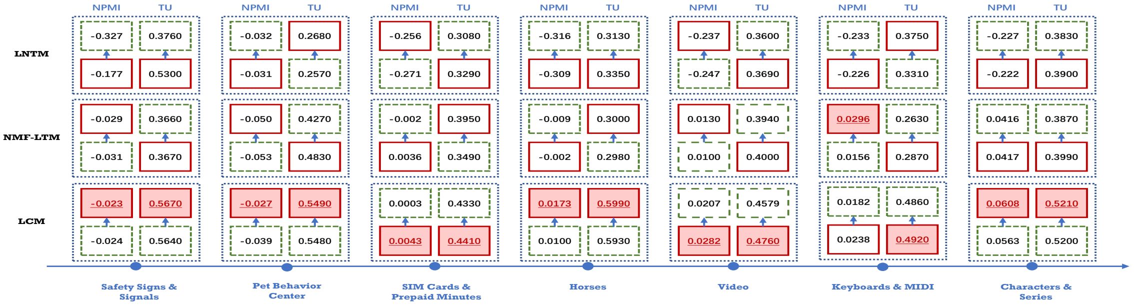

	Safety Signs & Signals NPMI	Safety Signs & Signals TU	Pet Behavior Center NPMI	Pet Behavior Center TU	SIM Cards & Prepaid Minutes NPMI	SIM Cards & Prepaid Minutes TU	Horses NPMI	Horses TU	Video NPMI	Video TU	Keyboards & MIDI NPMI	Keyboards & MIDI TU	Characters & Series NPMI	Characters & Series TU
LNTM	-0.327	0.3760	-0.032	0.2680	-0.256	0.3080	-0.316	0.3130	-0.237	0.3600	-0.233	0.3750	-0.227	0.3830
	-0.177	0.5300	-0.031	0.2570	-0.271	0.3290	-0.309	0.3350	-0.247	0.3690	-0.226	0.3310	-0.222	0.3900
NMF-LTM	-0.029	0.3660	-0.050	0.4270	-0.002	0.3950	-0.009	0.3000	0.0130	0.3940	0.0296	0.2630	0.0416	0.3870
	-0.031	0.3670	-0.053	0.4830	0.0036	0.3490	-0.002	0.2980	0.0100	0.4000	0.0156	0.2870	0.0417	0.3990
LCM	-0.023	0.5670	-0.027	0.5490	0.0003	0.4330	0.0173	0.5990	0.0207	0.4579	0.0182	0.4860	0.0608	0.5210
	-0.024	0.5640	-0.039	0.5480	0.0043	0.4410	0.0100	0.5930	0.0282	0.4760	0.0238	0.4920	0.0563	0.5200

Figure 3: Analysis on catastrophic forgetting. The x-axis shows the names of domains. For each model, the bottom line indicates metrics in the original lifelong learning process, while the top line shows the model performance with the help of $\mathcal{KG}$ generated by 7 training sets and CM. The boxes with red and solid edges mark the better metric for each model in each domain, while boxes filled in red with underlined values mark the best metric in each domain among all models.

effectiveness of global context, local context, and subwords. Compared to the subword information, $\mathcal{KG}$ contributes more to topic discovery and the local context information plays the most important role in word embedding learning. We replace BPE of $\mathcal{SG}$ with character n-gram features (Bojanowski et al., 2017) in "LCM-$\mathcal{SG}_{\mathcal{BPE}}$", which indicates the effectiveness of BPE on capturing subwords.

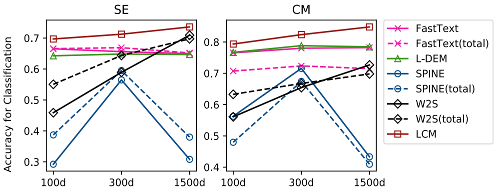

Figure 2: Classification performance comparison of SVM classifiers with word embeddings generated by different methods. Models marked with "total" are conducted on the combination of 7 training sets and the current testing set. W2S represents Word2Sense.

Method	NPMI	TU	Accuracy
LCM-$\mathcal{KG}_{\mathcal{G}}$	-0.0099	0.5790	0.7283
LCM-$\mathcal{KG}_{\mathcal{L}}$	-0.0109	0.5740	0.7244
LCM-$\mathcal{SG}$	-0.0076	0.5810	0.7288
LCM-$\mathcal{SG}_{\mathcal{BPE}}$	-0.0123	0.5800	0.7314
LCM	**0.0012**	**0.5940**	**0.7360**

Table 4: Results of ablation experiments.

4.5.4 Analysis on Catastrophic Forgetting

Catastrophic Forgetting (Robins, 1995; Kirkpatrick et al., 2017), which is a big challenge for lifelong topic models, will not be a serious problem for LCM. The learning process of LCM only accumulates knowledge in $\mathcal{KG}$, and the model is trained on each in-domain corpus independently by following (Chen et al., 2020b). To further investigate the ability of LCM in avoiding catastrophic forgetting, we use the final updated $\mathcal{KG}$ after CM to "go back" to help train the model on the 7 training set one by one. As LDA-LTM does not construct the in-domain text stream based on time series, we only take NMF-LTM and LNTM for comparison. In terms of NPMI and TU, Figure 3 shows that LCM has the best ability to alleviate catastrophic forgetting. For all domains, the latest $\mathcal{KG}$ will not have a significant negative impact on LCM (i.e., the catastrophic forgetting is limited), and sometimes it even helps with the new task. For example, both NMF-LTM and LCM achieve better NPMI scores with the latest $\mathcal{KG}$ in "SIM Cards & Prepaid Minutes". One possible reason is that later domains provide valuable information through $\mathcal{KG}$.

5 Conclusions

In this work, we propose a lifelong collaborative model (LCM) for learning topics and domain-specific word embeddings. LCM deals with the new in-domain corpus by coordinating global and local context information from previous domains, and subword information from the current corpus. A knowledge graph based on word-word relationships is leveraged during the learning process. Experiments on real-world in-domain text streams demonstrated the superior performances of LCM. In the future, we plan to incorporate contextualized word representations into topic models (Bianchi et al., 2020, 2021) for alleviating collapsing of word senses and learning more coherent topics.

Acknowledgment

We are grateful to the reviewers for their constructive comments and suggestions on this study. This work has been supported by the National Natural Science Foundation of China (61972426) and Guangdong Basic and Applied Basic Research Foundation (2020A1515010536).

References

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS)*, pages 13–22.

Roy Khristopher Bayot and Teresa Gonçalves. 2016. Multilingual author profiling using word embedding averages and SVMs. In *Proceedings of the 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA)*, pages 382–386.

Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2020. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. *CoRR*, abs/2004.03974.

Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. 2021. Cross-lingual contextualized topic models with zero-shot learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1676–1683.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Sophie Burkhardt and Stefan Kramer. 2019. Decoupling sparsity and smoothness in the dirichlet variational autoencoder topic model. *Journal of Machine Learning Research*, 20:131:1–131:27.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2018. Neural models for documents with metadata. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2031–2040.

Jonathan Chang, Jordan L. Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 288–296.

Hong-You Chen, Sz-Han Yu, and Shou-de Lin. 2020a. Glyph2vec: Learning chinese out-of-vocabulary word embedding from glyphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2865–2871.

Yong Chen, Junjie Wu, Jianying Lin, Rui Liu, Hui Zhang, and Zhiwen Ye. 2020b. Affinity regularized non-negative matrix factorization for lifelong topic modeling. *IEEE Transactions on Knowledge and Data Engineering*, 32(7):1249–1262.

Yong Chen, Hui Zhang, Junjie Wu, Xingguang Wang, Rui Liu, and Mengxiang Lin. 2015. Modeling emerging, evolving and fading topics using dynamic soft orthogonal NMF with sparse representation. In *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)*, pages 61–70.

Zhiyuan Chen. 2015. Lifelong machine learning for topic modeling and beyond. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 133–139.

Zhiyuan Chen and Bing Liu. 2014a. Mining topics in documents: Standing on the shoulders of big data. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1116–1125.

Zhiyuan Chen and Bing Liu. 2014b. Topic modeling using topics from many domains, lifelong learning and big data. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pages 703–711.

Ling-Yun Dai, Rong Zhu, and Juan Wang. 2020. Joint nonnegative matrix factorization based on sparse and graph laplacian regularization for clustering and co-differential expression genes analysis. *Complexity*, 2020:3917812:1–3917812:10.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 795–804.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Adji Bousso Dieng, Francisco J. R. Ruiz, and David M. Blei. 2020. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Hatem A. Fayed and Amir F. Atiya. 2019. Speed up grid-search for parameter selection of support vector machines. *Applied Soft Computing*, 80:202–210.

Pankaj Gupta, Yatin Chaudhary, Thomas A. Runkler, and Hinrich Schütze. 2020. Neural topic modeling with continual lifelong learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 3907–3917.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pages 507–517.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 50–57.

Patrik O. Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469.

Masaru Isonuma, Junichiro Mori, Danushka Bollegala, and Ichiro Sakata. 2020. Tree-structured neural topic model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 800–806.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 530–539.

Daniel D. Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2177–2185.

Tianyi Lin, Zhiyue Hu, and Xin Guo. 2019. Sparsemax and relaxed wasserstein for topic sparsity. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 141–149.

Hongyin Luo, Zhiyuan Liu, Huan-Bo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1687–1692.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 43–52.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR)*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2302–2310.

Brian Murphy, Partha Pratim Talukdar, and Tom M. Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1933–1950.

Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. 2019. Topic modeling with wasserstein autoencoders. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, pages 6345–6381.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 100–108.

Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. 2019. Word2sense: Sparse interpretable word embeddings. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, pages 5692–5705.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 102–112.

Yu-ping Qin and Xiu-kun Wang. 2009. Study on multi-label text classification based on SVM. In *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 300–304.

Anthony V. Robins. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Daniel L. Silver. 2011. Machine lifelong learning: Challenges and benefits for artificial general intelligence. In *Proceedings of the 4th International Conference on Artificial General Intelligence (AGI)*, pages 370–375.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. 2018. SPINE: Sparse interpretable neural embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 4921–4928.

Shuai Wang, Zhiyuan Chen, and Bing Liu. 2016. Mining aspect-specific opinion using a holistic lifelong topic model. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pages 167–176.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Lifelong domain word embedding via meta-learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4510–4516.

Guangxu Xun, Vishrawas Gopalakrishnan, Fenglong Ma, Yaliang Li, Jing Gao, and Aidong Zhang. 2016. Topic discovery for short texts using word embeddings. In *Proceedings of the 16th IEEE International Conference on Data Mining (ICDM)*, pages 1299–1304.

Guangxu Xun, Yaliang Li, Jing Gao, and Aidong Zhang. 2017a. Collaboratively improving topic discovery and word embeddings by coordinating global and local contexts. In *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 535–543.

Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. 2017b. A correlated topic model using word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4207–4213.

Appendices

A Updating Rules for $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, and $\boldsymbol{C}_t$

First, we transform the objective function L as:

$$\begin{aligned}
L =& \left\|\boldsymbol{D}_t - \boldsymbol{U}_t\boldsymbol{V}_t\right\|_F^2 + \left\|\boldsymbol{X}_t - \boldsymbol{B}_t\boldsymbol{C}_t^T\right\|_F^2 \\
&+ \lambda_v \left\|\boldsymbol{V}_t\right\|_1 \\
&+ \lambda_{u1} tr(\boldsymbol{U}_t^T\boldsymbol{H}_{t-1}\boldsymbol{U}_t) + \lambda_{u2}\left\|\boldsymbol{U}_t^T\boldsymbol{U}_t - \boldsymbol{I}_K\right\|_F^2 \\
&+ \lambda_c tr(\boldsymbol{C}_t^T\boldsymbol{H}_{t-1}\boldsymbol{C}_t) \\
&+ \lambda_{b1} tr(\boldsymbol{B}_t^T\boldsymbol{N}_t\boldsymbol{B}_t) + \lambda_{b2}\left\|\boldsymbol{B}_t\right\|_1 \\
=& tr(\boldsymbol{D}_t^T\boldsymbol{D}_t - 2\boldsymbol{D}_t^T\boldsymbol{U}_t\boldsymbol{V}_t + \boldsymbol{V}_t^T\boldsymbol{U}_t^T\boldsymbol{U}_t\boldsymbol{V}_t) \\
&+ tr(\boldsymbol{X}_t^T\boldsymbol{X}_t - 2\boldsymbol{X}_t^T\boldsymbol{B}_t\boldsymbol{C}_t^T + \boldsymbol{C}_t\boldsymbol{B}_t^T\boldsymbol{B}_t\boldsymbol{C}_t^T) \\
&+ \lambda_v tr(\boldsymbol{1}^T\boldsymbol{V}_t\boldsymbol{1}) \\
&+ \lambda_{u1} tr\left(\boldsymbol{U}_t^T diag(\boldsymbol{W}_{t-1}\cdot\boldsymbol{1})\boldsymbol{U}_t\right) \\
&- \lambda_{u1} tr(\boldsymbol{U}_t^T\boldsymbol{W}_{t-1}\boldsymbol{U}_t) \\
&+ \lambda_{u2} tr(\boldsymbol{U}_t^T\boldsymbol{U}_t\boldsymbol{U}_t^T\boldsymbol{U}_t - 2\boldsymbol{U}_t^T\boldsymbol{U}_t) \\
&+ \lambda_c tr\left(\boldsymbol{C}_t^T diag(\boldsymbol{W}_{t-1}\cdot\boldsymbol{1})\boldsymbol{C}_t\right) \\
&- \lambda_c tr(\boldsymbol{C}_t^T\boldsymbol{W}_{t-1}\boldsymbol{C}_t) \\
&+ \lambda_{b1} tr\left(\boldsymbol{B}_t^T diag(\boldsymbol{S}_t\cdot\boldsymbol{1})\boldsymbol{B}_t\right) \\
&- \lambda_{b1} tr(\boldsymbol{B}_t^T\boldsymbol{S}_t\boldsymbol{B}_t) \\
&+ \lambda_{b2} tr(\boldsymbol{1}^T\boldsymbol{B}_t\boldsymbol{1}) + const.
\end{aligned}$$

Then, the updating rules for matrices $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, and $\boldsymbol{C}_t$ can be derived as follows:

$$\begin{aligned}
\boldsymbol{U}_t &\leftarrow \boldsymbol{U}_t \circ \frac{\boldsymbol{D}_t\boldsymbol{V}_t^T + \lambda_{u1}\boldsymbol{W}_{t-1}\boldsymbol{U}_t + 2\lambda_{u2}\boldsymbol{U}_t}{\boldsymbol{U}_t\boldsymbol{V}_t\boldsymbol{V}_t^T + Z(\boldsymbol{U}_t)}, \\
\boldsymbol{V}_t &\leftarrow \boldsymbol{V}_t \circ \frac{\boldsymbol{U}_t^T\boldsymbol{D}_t}{\boldsymbol{U}_t^T\boldsymbol{U}_t\boldsymbol{V}_t + \frac{\lambda_v}{2}\cdot\boldsymbol{1}\cdot\boldsymbol{1}^T}, \\
\boldsymbol{C}_t &\leftarrow \boldsymbol{C}_t \circ \frac{\boldsymbol{X}_t^T\boldsymbol{B}_t + \lambda_c\boldsymbol{W}_{t-1}\boldsymbol{C}_t}{\boldsymbol{C}_t\boldsymbol{B}_t^T\boldsymbol{B}_t + \lambda_c diag(\boldsymbol{W}_{t-1}\cdot\boldsymbol{1})\boldsymbol{C}_t}.
\end{aligned}$$

In the above, $Z(\boldsymbol{U}_t) = \lambda_{u1} diag(\boldsymbol{W}_{t-1}\cdot\boldsymbol{1})\boldsymbol{U}_t + 2\lambda_{u2}\boldsymbol{U}_t\boldsymbol{U}_t^T\boldsymbol{U}_t$.

B Alternately Iterative Algorithm

The inference method of our LCM is shown in Algorithm 1.

C Convergence Analysis

In this section, we analyze the convergence of Algorithm 1.

Theorem C.1. *Algorithm 1 is guaranteed to converge to a locally-optimal solution.*

Algorithm 1 Alternately Iterative Algorithm

```
Input: $\{\mathcal{DOC}_t\}_{t=1}^{T}$ ($T$ can be infinite);
Output: $\{\boldsymbol{U}_t\}_{t=1}^{T}$, $\{\boldsymbol{V}_t\}_{t=1}^{T}$, $\{\boldsymbol{C}_t\}_{t=1}^{T}$, $\{\boldsymbol{B}_t\}_{t=1}^{T}$.
$\mathcal{KG}_0 = \emptyset$;
for $t$ = 1, 2, 3, ..., $T$ do
    Randomly initialize non-negative matrices $\boldsymbol{U}_t^{(0)}$, $\boldsymbol{V}_t^{(0)}$, $\boldsymbol{C}_t^{(0)}$, $\boldsymbol{B}_t^{(0)}$;
    Construct $\boldsymbol{S}_t$ from $\mathcal{SG}_t$;
    Construct $\boldsymbol{K}_{t-1}$ from $\mathcal{KG}_{t-1}$;
    Set the iteration number $i$ to 0;
    repeat
        $i = i + 1$;
        Compute $\boldsymbol{B}_t^{(i)}$, $\boldsymbol{U}_t^{(i)}$, $\boldsymbol{V}_t^{(i)}$, and $\boldsymbol{C}_t^{(i)}$;
    until convergence;
    Compute $\mathcal{KG}_t$ using Eq. (3).
end
```

First, we prove the convergence of the update rule of $\boldsymbol{B}_t$ in Eq. (11).

Definition C.1. *$G(x, z)$ is an auxiliary function for $F(x)$ if the following conditions are satisfied.*

$$G(x, z) \geq F(x), G(x, x) = F(x).$$

Lemma C.1. *If G is an auxiliary function, then F is non-increasing under the following update rule:*

$$x^{t+1} = arg\min_x G(x, x^t). \tag{12}$$

Proof.

$$F(x^{t+1}) \leq G(x^{t+1}, x^t) \leq G(x^t, x^t) = F(x^t).$$

□

If we could prove that the updating rule of $\boldsymbol{B}_t$ confirms to Eq. (12) for an appropriate auxiliary function, we would conclude that $\boldsymbol{B}_t$ converges to a local minimum.

Lemma C.2. *Let $z = (\boldsymbol{B}_t)_{ij} > 0$, $G(x, z)$ is an auxiliary function for $F(z) = L\left((\boldsymbol{B}_t)_{ij} = z\right)$.*

$$\begin{aligned}
G(x, z) =& F(z) + \frac{\partial F(z)}{\partial z}(x - z) + \frac{(x-z)^2}{z} \\
& (\boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t + \lambda_{b1} diag(\boldsymbol{S}_t\cdot\boldsymbol{1})\boldsymbol{B}_t \\
& + \frac{\lambda_{b2}}{2}\cdot\boldsymbol{1}\cdot\boldsymbol{1}^T)_{ij}.
\end{aligned}$$

Proof. Clearly, $G(x, x) = F(x)$. Taylor expansion of $F(x)$ is:

$$F(x) = F(z) + \frac{\partial F(z)}{\partial z}(x - z) + \frac{1}{2}\frac{\partial^2 F(z)}{\partial z^2}(x - z)^2.$$

In order to show G is an auxiliary function, we have to show $G(x,z) \geq F(x)$:

$$\frac{\left(\boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t + \lambda_{b1} diag(\boldsymbol{S}_t \cdot \mathbf{1})\boldsymbol{B}_t + \frac{\lambda_{b2}}{2} \cdot \mathbf{1} \cdot \mathbf{1}^T\right)_{ij}}{z} \geq \frac{1}{2}\frac{\partial^2 F(z)}{\partial z^2},$$

where $\frac{\partial^2 F(z)}{\partial^2 z} = 2(\boldsymbol{C}_t^T\boldsymbol{C}_t)_{jj} + 2\lambda_{b1}(diag(\boldsymbol{S}_t \cdot \mathbf{1}))_{ii} - 2\lambda_{b1}(\boldsymbol{S}_t)_{ii}$.

To prove the inequation, we first verify how the inequality holds on the first term:

$$\begin{aligned}
\frac{(\boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t)_{ij}}{(\boldsymbol{B}_t)_{ij}} =& \frac{1}{(\boldsymbol{B}_t)_{ij}}\sum_k (\boldsymbol{B}_t)_{ik}(\boldsymbol{C}_t^T\boldsymbol{C}_t)_{kj} \\
=& \frac{1}{(\boldsymbol{B}_t)_{ij}}\sum_{k=j} (\boldsymbol{B}_t)_{ik}(\boldsymbol{C}_t^T\boldsymbol{C}_t)_{kj} \\
&+ \frac{1}{(\boldsymbol{B}_t)_{ij}}\sum_{k\neq j} (\boldsymbol{B}_t)_{ik}(\boldsymbol{C}_t^T\boldsymbol{C}_t)_{kj} \\
=& \frac{1}{(\boldsymbol{B}_t)_{ij}}(\boldsymbol{B}_t)_{ij}(\boldsymbol{C}_t^T\boldsymbol{C}_t)_{jj} \\
&+ \frac{1}{(\boldsymbol{B}_t)_{ij}}\sum_{k\neq j} (\boldsymbol{B}_t)_{ik}(\boldsymbol{C}_t^T\boldsymbol{C}_t)_{kj} \\
\geq& (\boldsymbol{C}_t^T\boldsymbol{C}_t)_{jj}.
\end{aligned}$$

Similarly, we can get:

$$\lambda_{b1}\left(diag(\boldsymbol{S}_t \cdot \mathbf{1})\boldsymbol{B}_t\right)_{ij} \geq \lambda_{b1}\left(diag(\boldsymbol{S}_t \cdot \mathbf{1})\right)_{ii}.$$

Since λ_{b1}, λ_{b2}, and each element in $\boldsymbol{S}_t$ are nonnegative, we have the above inequation. This establishes that G is an auxiliary function for F. □

Proof. To show that Algorithm 1 converges (i.e., Theorem C.1), we need to show that update rule for $\boldsymbol{B}_t$ follows Eq. (12). $\frac{\partial G(x,z)}{\partial x}$ is listed as follows:

$$\begin{aligned}
\frac{\partial G(x,z)}{\partial x} =& (-2\boldsymbol{X}_t\boldsymbol{C}_t + 2\boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t \\
&+ 2\lambda_{b1} diag(\boldsymbol{S}_t \cdot \mathbf{1})\boldsymbol{B}_t - 2\lambda_{b1}\boldsymbol{S}_t\boldsymbol{B}_t \\
&+ \lambda_{b2} \cdot \mathbf{1} \cdot \mathbf{1}^T)_{ij} \\
&+ \frac{x-z}{z} \cdot 2(\lambda_{b1} diag(\boldsymbol{S}_t \cdot \mathbf{1})\boldsymbol{B}_t \\
&+ \boldsymbol{B}_t\boldsymbol{C}_t^T\boldsymbol{C}_t + \frac{\lambda_{b2}}{2} \cdot \mathbf{1} \cdot \mathbf{1}^T)_{ij}.
\end{aligned}$$

Solving $\frac{\partial G(x,z)}{\partial x} = 0$ for x, we get the update rule as mentioned in Eq. (11). Since G is the auxiliary function for F, the value of F is non-increasing. We can prove the convergence of update rules for $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, and $\boldsymbol{C}_t$ similarly. Thus, Algorithm 1 is guaranteed to converge to a local minimum. □

D Time Complexity Analysis

In this section, we analyze the time complexity of Algorithm 1. For updating matrices $\boldsymbol{B}_t$, $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, and $\boldsymbol{C}_t$, the time complexity of one iteration is $O((3E+1)M^2+(2E^2+3E)M)$, $O((3E+1)M^2+(2E^2+2E)M)$, $O(MNK+2M^2K+3MK^2+NK^2+M^2+2MK)$, and $O(3KMN+3KN)$, respectively. Thus, the time complexity of each iteration is $O(4MNK+(6E+3)M^2+(4E^2+5E)M+2M^2K+3MK^2+NK^2+2MK+3KN)$ for our method, which spends an extra time cost of $O((6E+2)M^2+(4E^2+5E)M-(2K+1)N^2)$ to learn word embeddings when compared with the previous NMF-based lifelong topic model, i.e., NMF-LTM (Chen et al., 2020b). Although the time complexity is proportional to M, we can alleviate the scalability issue simply. For example, if M (i.e., the vocabulary) of a single domain is too large, we can partition this domain into several small subdomain corpora. At each time, we only process matrices of one sub-domain, and M of each small sub-domain will not be too large.

As an illustration, it costs about 30 seconds per iteration for training LCM based on a workstation equipped with Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50 GHz, 8 cores and 128G memory. To achieve convergence, LCM costs about 1 hour to update $\boldsymbol{B}_t$, $\boldsymbol{U}_t$, $\boldsymbol{V}_t$, and $\boldsymbol{C}_t$ for all domains in order. NMF-LTM costs about half an hour accordingly.

E Different Permutations of Training Domains

We shuffle training domains randomly for 5 times, and show the results under these different permutations in Table 5, which indicates that LCM is robust to domain permutations.

F Search Space of Hyperparameters

Table 6 shows the search space of hyperparameters for grid search in our LCM. To ensure a fair comparison with NMF-LTM (Chen et al., 2020b), *topT* in LCM is set to 5. For each word, our knowledge graph selects 5 most similar words from global and local context information, respectively. In other words, each word is connected to 10 similar words, which is equivalent to NMF-LTM. Note that for the topic number, we select 50 as topic numbers for all evaluations of LCM after validation. For fair comparison, we directly choose 50 as topic numbers for baselines and this is not a hyperparameter

Permutation	Science Education (SE)				Cult Movies (CM)			
	NPMI↑	TU↑	Perplexity↓	Accuracy↑	NPMI↑	TU↑	Perplexity↓	Accuracy↑
1	-0.0072	0.5870	8708.0200	0.7207	0.0639	0.6340	13350.4800	0.8455
2	-0.0064	0.5840	8704.2400	0.7196	0.0570	0.6090	13420.5600	0.8461
3	-0.0106	0.5960	8701.5600	0.7196	0.0580	0.6170	13373.0800	0.8443
4	-0.0053	0.5880	8705.9500	0.7171	0.0524	0.6090	13369.4400	0.8467
5	-0.0013	0.5700	8744.0600	0.7135	0.0562	0.6030	13554.5300	0.8480

Table 5: Performance comparison of LCM within different permutations of the dataset. For all metrics, "↓" after the metric indicates smaller is better while "↑" indicates larger is better.

Hyperparameter	Search Space
λ_v	[0.001]
λ_{u2}	[0.5]
λ_{b2}	[0.001, 0.1]
λ_{u1}, λ_c, λ_{b1}	[0.1, 1, 10]
$topT$	[5]
η	[0.1, 0.5, 1, 2, 10]
R_k, R_s	[0.5, 0.6]
topic number	[50, 200]
word embedding dimension	[100, 300, 1500]
iterations	[100, 200]

Table 6: Hyperparameters' search space.

Hyperparameter	NPMI	TU	Perplexity	Accuracy
λ_v	0.001	0.001	0.001	0.001
λ_{u2}	0.5	0.5	0.5	0.5
λ_{b2}	0.001	0.1	0.1	0.1
λ_{u1}	10	10	10	0.1
λ_c	10	0.1	10	10
λ_{b1}	0.1	1	10	1
$topT$	5	5	5	5
η	1	1	2	1
R_k	0.5	0.5	0.6	0.5
R_s	0.6	0.6	0.6	0.5
topic number	50	50	50	50
word embedding dimension	1500	1500	1500	300
iterations	200	200	200	200

Table 7: Best hyperparameters on the validation set.

Hyperparameter	Range	Var-NPMI	Var-Accuracy
λ_{u1}	[0.01, 0.1, 1, 10, 100]	1.28E-05	1.79E-05
λ_{b1}	[0.01, 0.1, 1, 10, 100]	2.71E-05	3.50E-03
λ_c	[0.01, 0.1, 1, 10, 100]	9.32E-07	5.15E-06
R_k	[0.3, 0.5, 0.6, 0.8]	6.57E-05	3.08E-05
iterations	[50, 100, 200, 300]	1.28E-05	3.92E-05

Table 8: Variances of results.

of baselines. We list the best hyperparameters on the validation set for different metrics in Table 7. We provide the search space of hyperparameters for grid search in all of our baselines and their corresponding best hyperparameters in our codes.

For hyperparameters, we first varied search spaces for sensitive analysis and observed that LCM was robust to most hyperparameters. Thus, we used final search spaces in Table 6. For completeness, we also show the variances of results under different hyperparameter values in Table 8. Take some hyperparameters in CM as examples, we vary each parameter when others are fixed, and compute the variances of NPMI and Accuracy, which indicates that LCM is robust to most hyperparameters.

G Qualitative Analysis of Topics

Following (Chen et al., 2020b), we map the topics learned by LCM with ones by LDA-LTM (Chen and Liu, 2014b), LNTM (Gupta et al., 2020), and NMF-LTM (Chen et al., 2020b), respectively. Particularly, we represent each topic by its top 20 words, and compute the cosine similarity between every two topics. Take SE as an example, we randomly show 3 topics, as listed in Table 9. Irrelevant words are marked by italics.

LDA-LTM, NMF-LTM, and LCM learn topics well in most cases. However, NMF-LTM captures some high-frequency words (i.e., amazon, anatomy, anatomical) in the corpus, which are not related to the topic "Design and production of handicrafts". LDA-LTM also assigns an irrelevant word "print" to the topic "Machinery and industrial manufacturing technology", and an irrelevant word "spring" to the topic "Body structure and anatomy".

The ability of LNTM in generating cohesive topics is poor. It is noteworthy that the topics generated by LNTM seem not related to other models, because we use cosine similarity to map topics. If the cosine similarities are the same (for LNTM, it is 0 sometimes), the topics with smaller IDs will be chosen. For the sake of fairness, we also show a relatively coherent topic generated by LNTM separately in Table 10. The result of LNTM is still worse than other models, which contains 7 irrelevant words in the top 10 word list.

H Evaluating Interpretability

To evaluate the interpretability of our domain-specific word embeddings, we follow (Murphy et al., 2012) to show top 5 words for 5 randomly chosen dimensions in word embeddings generated from CM. Although there exists noisy words, the dimensions are generally semantically coherent and interpretable. We also choose one polysemous word "cell" in SE to measure the ability of our method in capturing the polysemous nature of words. We select the two highest values of the word vector learned by our LCM for "cell", and find top 5 words in these two dimensions. From Table 11, we can observe that the two dimensions focus on cells in biology and cell phones, which reflect the two different meanings of "cell".

ID	LDA-LTM	LNTM	NMF-LTM	LCM
		Machinery and industrial manufacturing technology		
1	bed	*deserved*	mm	mm
2	mm	clamped	bed	bed
3	axis	*mothers*	frame	frame
4	screws	*crowds*	screws	screws
5	screw	*players*	axis	axis
6	belt	*suspiciously*	screw	set
7	motor	*resolutions*	stepper	screw
8	nuts	*wasps*	power	stepper
9	*print*	*military*	wires	lead
10	set	*collapses*	tape	tape
		Design and production of handicrafts		
1	model	*deserved*	model	brain
2	brain	*crowded*	brain	paint
3	pieces	*mothers*	models	models
4	models	*rider*	structures	pieces
5	paint	*mutual*	*anatomy*	structures
6	heart	*trends*	paint	lines
7	motor	*wasp*	*amazon*	*budget*
8	quality	*bloom*	detail	detail
9	painted	*fumbling*	*anatomical*	white
10	job	*habitat*	stand	detailed
		Body structure and anatomy		
1	skull	*deserved*	skull	skull
2	teeth	*crowded*	teeth	teeth
3	jaw	*mothers*	anatomy	anatomy
4	quality	*rider*	jaw	jaw
5	size	*mutual*	bone	mandible
6	top	*trends*	mandible	foramen
7	*spring*	*wasp*	foramen	bone
8	removable	*bloom*	detail	detail
9	life	*fumbling*	skulls	study
10	skulls	*habitat*	removable	removable

Table 9: Qualitative analysis of topics. Top 10 words are listed with irrelevant ones marked in italics.

ID	LNTM
	Music and instruments
1	*tens*
2	*deserved*
3	*habitat*
4	*circuitry*
5	guitars
6	*mothers*
7	headphones
8	piano
9	*rider*
10	*foray*

Table 10: Qualitative analysis of LNTM. Top 10 words are listed with irrelevant ones marked in italics.

Top Words	depleted painted icing weed angora	constant spray cake pot sweater	drained paint riffing smoking dressing	dissolve grease esp smoke sweaters	exhausted di wrapping smoked skirt
Close Words for "cell"	cell phones	sewn cell	animal games	cells fascinating	believes computers

Table 11: Examples of evaluating interpretability.

Table 12 shows the classification accuracy and the sparsity (i.e., the proportion of zeros) of word embeddings generated by different methods, where the dimension is 1500. Compared with other models, our LCM generates domain-specific word embeddings with a good balance between sparsity and classification accuracy.

Method	Science Education		Cult Movies	
	Accuracy	Sparsity	Accuracy	Sparsity
SPINE	0.3080	0.9969	0.4332	0.9979
Word2Sense	0.7043	0.9920	0.7267	0.9920
LCM	0.7360	0.9438	0.8481	0.9099

Table 12: Classification accuracy and sparsity of word embeddings generated by different methods.

I Evaluating Word Embedding Learning Models by Neural Networks

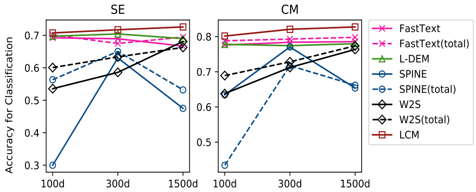

Figure 4: Classification performance comparison of neural networks with word embeddings generated by different methods.

To compare different word embedding learning models comprehensively, we replace the SVM classifier with a neural network classifier consisting of 3 fully-connected layers. As shown in Figure 4, LCM also performs the best in each testing set under each dimension.

Leveraging Argumentation Knowledge Graph for Interactive Argument Pair Identification

Jian Yuan[1], **Zhongyu Wei**[1,2]*, **Donghua Zhao**[3], **Qi Zhang**[4], **Changjian Jiang**[5]*
[1]School of Data Science, Fudan University, China
[2]Research Institute of Intelligent and Complex Systems, Fudan University, China
[3]School of Mathematical Sciences, Fudan University, China
[4]School of Computer Science, Fudan University, China
[5]School of International Relations & Public Affairs, Fudan University, China
`{19210980107,zywei,dhzhao,qz,Changjian}@fudan.edu.cn`

Abstract

Interactive argument pair identification is essential in the context of dialogical argumentation mining. Existing research treats it as a problem of sentence matching and largely relies on textual information to compute the similarities. However, the interaction of opinions usually involves the background of the topic and requires reasoning of knowledge, which is beyond textual information. In this paper, we propose to leverage external knowledge to enhance the identification of interactive argument pairs. We construct the argumentation knowledge graph from the discussion thread of the target topic in the online forum. The interaction between the original argument and the reply is then represented as the path of concepts in the knowledge graph. In practice, we utilize Graph Convolutional Network (GCN) to learn the concept representation in the knowledge graph and use a Transformer-based encoder to learn the representation of paths. Finally, an information alignment network is employed to capture the interaction of textual information of conceptual information (both entity-level and path-level). Experiment results indicate that our model achieves state-of-the-art performance in the benchmark dataset. Further analysis demonstrates the effectiveness of our model for enforcing knowledge reasoning through paths in the knowledge graph.

1 Introduction

Argumentation Mining aims at analyzing the semantic and logical structure of argumentative texts. Existing research covers argument structure prediction (Morio et al., 2020; Li et al., 2020), persuasiveness evaluation (Al Khatib et al., 2020; El Baff et al., 2020) and argument summarization (Bar-Haim et al., 2020b,a). Most of them focus on monological context like student essays, public speeches, etc., where only one participant is involved.

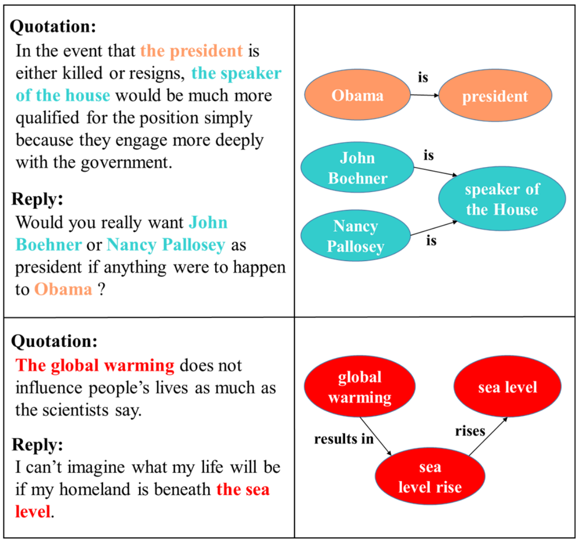
Quotation:
In the event that the president is either killed or resigns, the speaker of the house would be much more qualified for the position simply because they engage more deeply with the government.

Reply:
Would you really want John Boehner or Nancy Pallosey as president if anything were to happen to Obama ?

Quotation:
The global warming does not influence people's lives as much as the scientists say.

Reply:
I can't imagine what my life will be if my homeland is beneath the sea level.

Figure 1: Two instances of interactive argument pairs, the related concepts are colored same, and the corresponding knowledge is visualised in the right side.

Online forums such as idebate[1] and changemyview[2], enable people to exchange opinions on some specific topics freely. The user generated dataset of interactive arguments also motivates another line of research for argumentation in dialogical context (Asterhan and Schwarz, 2007). Initial researches in this filed focused on analyzing the ChangeMyView data (Tan et al., 2016; Wei et al., 2016) to summarize the key factors of persuasive arguments. Furthermore, Ji et al. (2019) and Cheng et al. (2020) propose the task of identifying and extracting interactive arguments. Ji et al. (2019) formulate this task as a problem of sentence pair scoring and computes the textual similarity

---

*Corresponding author.

[1] `https://idebate.org/`

[2] `https://www.reddit.com/r/changemyview/`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2310–2319
August 1–6, 2021. ©2021 Association for Computational Linguistics

between the two arguments as the result. Such task is then further applied to other fields such as legal domain. For instance, Yuan et al. (2021) organize a challenge aimed to identify the interactive arguments from the plaintiff and the defense in a legal case. However, the interaction of argumentation is beyond text matching.

Two sample pairs of interactive arguments are shown in Figure 1. Both pairs of arguments share a limited number of overlapping tokens and fail existing models. We have two observations. Firstly, background knowledge needs to be involved. In the first sample, we need to know that ”Obama” is the ”president”, and both ”John Boehner” and ”Nancy Pallosey” are the ”speaker of the house” to understand the context. Secondly, knowledge reasoning is necessary. In the second sample, the relationship between ”global warming” and ”sea level” is implied by a series of causal effects. Furthermore, as is shown in the example, an effective way of leveraging commonsense and causal effect knowledge is to find the reasoning paths between the concept entity pairs. Therefore, we argue that retrieving and understanding the reasoning paths should be incorporated for the identification of interactive arguments.

In this paper, we propose to leverage external knowledge to enhance the automatic identification of interactive arguments via background knowledge modeling and reasoning. We start with constructing an argumentation knowledge graph following (Khatib et al., 2020) based on the context of the discussion. Then, we extract entities of each argument and link them with the external knowledge graph to obtain the concept embedding as background knowledge. Besides, we generate paths connecting each pair of entities and encode them via a transformer encoder to enforce the reasoning. Finally, we integrate the entity embeddings, path representations, and textual embedding via an information alignment network to learn the final representation of the argument pair and output a real value as the matching score. We evaluate our proposed model on a publicly available dataset and experimental results show its effectiveness compared to some state-of-the-art approaches. Further analysis of the path encoding module reveals that our model is able to perform knowledge reasoning to some extent.

Statistics	**w/o. grounding**	**w. grounding**
# of nodes	291,199	291,199
# of edges	785,036	859,534
avg. degree	2.696	2.952
# of connected components	13,805	10,035

Table 1: Basic statistics on the argumentation graph without and with concept grounding.

2 Argumentation Knowledge Graph Construction

Data Source The experimental dataset (Ji et al., 2019) in our research is constructed on top of the *CMV* dataset (Tan et al., 2016). In order to provide external knowledge for the identification of interactive arguments, we construct an argumentation knowledge graph based on the CMV dataset. *ChangeMyView* (CMV) is an online forum where users can either submit a post to elaborate their own viewpoints and invite other users to convince them of the opposite opinion or reply to others' posts to change the poster's original view. Tan et al. (2016) crawled 20,626 discussion threads with more than two posts from January 2013 to September 2015. We first extract all the concept-relation-concept triples (e_i^h, r_i, e_i^t) in the ith entry of the data source using Open Information Extraction (OpenIE). Our raw graph is thus $G = (V, E)$, where $V = \{e_i^h\}_{i=1}^n \cup \{e_i^t\}_{i=1}^n$ and $E = \{r_i\}_{i=1}^n$. The raw knowledge graph contains 291,199 nodes and 785,036 edges.

Concept Grounding In order to further improve the quality of the knowledge graph, we conduct concept grounding to align all the nodes that share common conceptual meanings. Specifically, we use *WordNet* and Wikipedia API *TagMe* (Ferragina and Scaiella, 2010) in this process. If two concepts e_i, e_j are synonyms or refer to the same entry on Wikipedia, we add a new edge r_{equal} to the graph's edge set E. After concept grounding, the size of the edge set E expands to 859,534 with the size of the node-set V remained fixed. Some basic statistics of the knowledge graph are as shown in Table 1. It indicates that concept grounding increases the number of edges in a large margin and alleviates the problem of the sparsity of the original graph.

3 Proposed Model

Given an original argument q and its context c_q, and five candidate replies $\{r_i\}_{i=1}^5$ with their corresponding contexts $\{c_i\}_{i=1}^5$, the model needs to

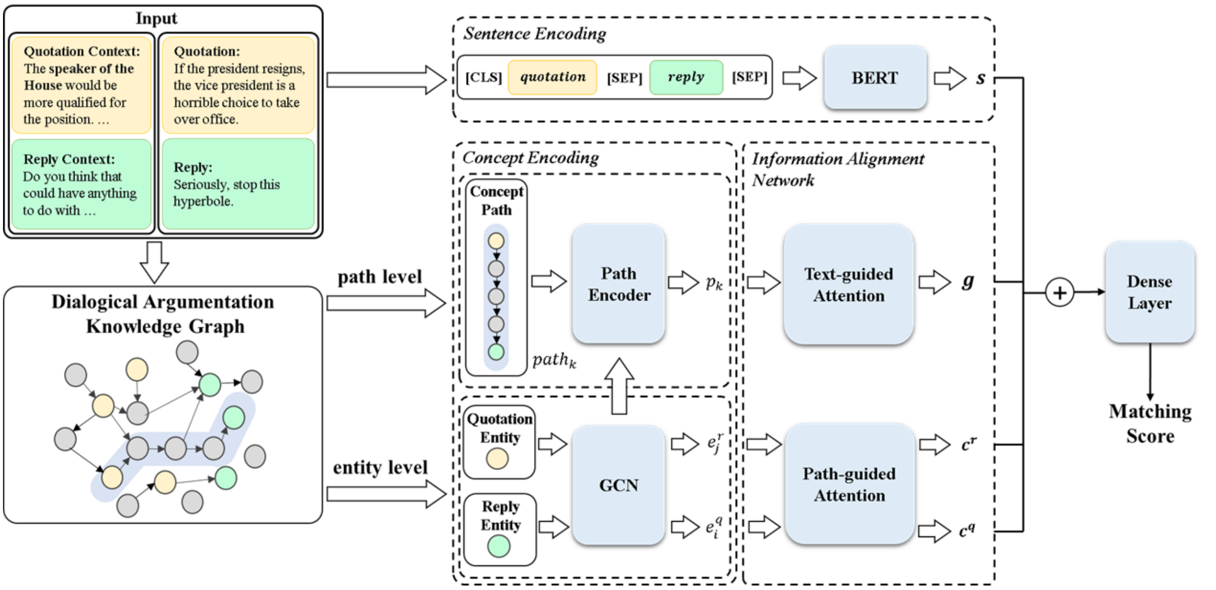

Figure 2: Illustration of the detailed architecture of our model to generate the matching feature vector, which mainly consists of three modules, a Sentence Encoder, a Concept Encoder and an Information Alignment Network. The output of these modules is then fed to a 2-layer perceptron to achieve the final matching score for the given argument pair.

identify the correct reply for q. We score each candidate pair independently and choose the reply with the highest score as the output. Moreover, in order to enable our model to conduct a reasoning process, we extract all the concept entities mentioned in the contexts from both sides, and also the concept paths that connect them. For simplicity, we will use sentence pairs to refer to the quotation and reply arguments and use concept pairs to refer to both the entities and paths in the following sections.

The full architecture of our scoring model is shown in Figure 2. It takes a sentence pair, and the concept pair extracted from its corresponding contexts as inputs and outputs a real value as its matching score. Our model mainly consists of three components, namely, sentence encoding, concept encoding, and information alignment network. We use a pre-trained language model, BERT, to learn the argument pair representation (§3.1), and encode the concept information from two levels, both entity level and path level with graph networks (§3.2). The information alignment network then integrates the sentence pair encoding and the concept encoding through a hierarchical attention mechanism to obtain the full matching features (§3.3), which are finally fed into a multi-layer perceptron (MLP) to calculate the final matching score (§3.4).

3.1 Sentence Encoding

As for the quotation and reply arguments, it is critical to use the semantic information implied in the texts. Various works have already proved the outstanding performance of pre-trained models in semantic modeling. In our work, we use the BERT model to generate the encoding s for the given argument pair by simply creating a sentence that takes the form of ”[CLS] q [SEP] r [SEP]” and taking the embedding for the ”[CLS]” token, just as suggested by previous works (Talmor et al., 2019).

3.2 Concept Encoding

For entities in the argumentation knowledge graph, we need to obtain the representation for each node. We use the BERT model with average pooling to get the initial representation for each entity. Then we encode the conceptual information in both entity-level and path-level with graph networks to enforce the background knowledge modeling and reasoning.

3.2.1 Entity Level Representation

To utilize the structural information entailed in the knowledge graph, we apply a 2-layer Graph Convolutional Network (GCN) to it. Here we adopt GCN as it has proved to be both effective and efficient in merging the node's neighbours' information into itself (Zhang et al., 2018).

Formally, let $X \in \mathbb{R}^{n \times d}$ representing the embedding matrix for all n nodes, where each node's embedding is of size d. Denote $D \in \mathbb{R}^{n \times n}$ as the diagonal degree matrix and $A \in \mathbb{R}^{n \times n}$ as the adjacency matrix of the graph G. Then the normalized symmetric adjacency matrix of the graph G can be calculated as:

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{1}$$

By feeding the graph G into the 2-layer GCN, the final graph representation $L \in \mathbb{R}^{n \times d}$ can be calculated as:

$$L = \sigma(\tilde{A}\sigma(\tilde{A}XW_0)W_1), \tag{2}$$

where σ stands for the non-linear function (RELU), and $W_0, W_1 \in \mathbb{R}^{d \times n}$ are trainable parameters of the network.

3.2.2 Path Level Representation

To further utilize the external knowledge, we want to encode the concept path retrieved from the knowledge graph, where a path starts from a concept e_q mentioned in the original post(i.e. quotation) q, traverses through the neighbored concepts, and finally ends at a concept e_r extracted from the reply r. For each concept pair, we choose the shortest path (if exists) between them as the path connecting them.

We use the GCN output as the representation for each node that appears in the path, hence we can denote the path between the i-th concept in q (c_i^q) and the j-th concept (c_j^r) in r as $P_{ij} = (c_i^q, c_1, ...c_{m_{ij}-1}, c_j^r) \in \mathbb{R}^{m_{ij} \times d}$, where m_{ij} is the the length of the path P_{ij}.

Transformer (Vaswani et al., 2017) has been shown powerful due to its self-attention mechanism, thus, we choose it to encode the path we collected from the knowledge graph. To underline the influence of the sequence in each path, we add the path's embedding with positional embedding PE. To sum up, our path encoder generates the representation for each $Path_{ij}$ as:

$$p_{ij} = \text{Transformer\_Encoder}(P_{ij} + PE). \tag{3}$$

The output is finally fed to a fully-connected layer to fit into the size of d.

3.3 Information Alignment Network

We then align the semantic information and the conceptual information through a hierarchical attention mechanism, i.e. a text-guided attention network for paths and a path-guided attention network for entities.

3.3.1 Text-guided Attention over Paths

Note that for the given argument pair and their contexts, we already have all the paths' encoding from previous modules. We first use attention between the k-th paths p_k and semantic vector s to integrate the encoding for all the paths g:

$$\alpha_k = sW_2p_k, \tag{4}$$

$$\hat{\alpha} = \text{SoftMax}(\alpha), \tag{5}$$

$$g = \sum_k \hat{\alpha}_k p_k, \tag{6}$$

where W_2 is a parameter matrix to be learned, α and $\hat{\alpha}$ stands for the unnormalized and normalized attention weights.

3.3.2 Path-guided Attention over Concepts

Obtaining the full paths' representation g, we can further aggregate all the concepts' encoding $\{e_i\}$ of both sides using attention between them and g to generate the final representation of concepts c^q and c^r:

$$\beta_i^s = gW_3^s e_i^s, \tag{7}$$

$$\hat{\beta}^s = \text{SoftMax}(\beta^s), \tag{8}$$

$$c^s = \sum_i \hat{\beta}_i e_i^s, \tag{9}$$

where the subscript $s \in \{q, r\}$ indicates whether the concepts are from quotation or reply, W_3^q and W_3^r are parameters matrix to be learned, while $\hat{\beta}^q$ and $\hat{\beta}^r$ stand for the attention weights.

3.4 Matching Scoring

Eventually, we concatenate the textual information s, the reasoning paths information g and the concepts information c^q, c^r as the final feature and feed it to 2-layer perceptron to generate the matching score S of the given argument pair:

$$f = [s; g; c^q; c^r], \tag{10}$$

$$S = \sigma(W_S f + b_S), \tag{11}$$

where σ refers to the the rectified linear activation function (ReLU), W_S and b_S represent the weight vector and the bias respectively.

After obtaining the matching score for each argument pair, we treat the task as a sentence pair ranking problem, and use MarginRankingLoss for training:

$$L = \sum_{i=1}^{4} \max(0, \gamma - S(q, r^+) + S(q, r_i^-)), \tag{12}$$

where $S(q, r^+)$ refers to the matching score of the positive argument pair while $S(q, r_i^-)$ refers to the matching score of the i-th negative argument pair, and γ is the margin hyperparameter.

4 Experiments

In this section, we will introduce the dataset, the evaluation metrics, comparative models and experiment results.

4.1 Experiment Setup

Experimental Dataset We use the dataset constructed in (Ji et al., 2019) for evaluation. The authors find that in the *ChangeMyView* dataset (Tan et al., 2016), there exist replies that quote sentences from the original post. They extract all these *quotation-reply* pair q, r from posts in *ChangeMyView* dataset (Tan et al., 2016). For every interactive argument pair, they randomly sample four negative replies $\{r_i^{neg}\}_{i=1}^4$ along with their contexts $\{c_i^{neg}\}_{i=1}^4$ from the same discussion thread. It contains 11,565 and 1,481 instances in training set and test set respectively. Furthermore, we randomly split 10% of the training set as validation set.

Implementation Details The output dimensions for the two layers in GCN are 256 and 128 respectively, the path transformer encoder we use is stacked by 6 encoder layers. The margin γ used in MarginRankingLoss is set to 0.5. Dropout is used as 0.1 to avoid overfitting. We use Adam as our optimizer with a learning rate set to 5×10^{-6} and weight decay set to 5×10^{-6}. We run our model for 100 epochs with early stop (Caruana et al., 2000).

Models for comparison We compare the performance of some state-of-the-art models.

- **BiGRU**: This method uses a Bidirectional GRU to encode the quotation and the reply argument separately and integrates their representations into a multilayer perceptron (MLP) to get the matching score.

- **VAE** : This method uses variational auto encoder through an encoder-decoder based architecture to get the encoding of the arguments and utilizes MLP for scoring.

- **DVAE** (Rolfe, 2017): This method substitutes the above *VAE* module with a discrete variational auto encoder and adopts the former framework.

- **BERT** (Devlin et al., 2019): This method fine-tunes the pre-trained BERT model for sentence-pair classification. Note that this model is not only a baseline model but also a sub-module of our proposed model.

Note that the above models only utilize the sentences of q and r, we also extend these models to incorporate context information.

- **RNN Context**: This method uses another BiGRU module to encode the context information of each argument and concatenate it with the argument representation to get the final features.

- **Hierarchical Context** (Ji et al., 2019): This method uses a token-level CNN with an attention mechanism to achieve the sentence-level information and then integrates such sentence representation with a BiGRU layer to obtain the final context encoding.

4.2 Overall Performance

We report both precision at one(P@1) and mean reciprocal rank (*MRR*) for evaluation. The performance of all the baseline models and our proposed model is as listed in Table 2. We have the following findings.

- Among all the context-agnostic baseline models, the BERT model achieves the highest performance, and it even defeats all other models that utilize the context information, indicating that such pre-trained language model does better encode the semantic information entailed in the texts.

- Incorporating context information is crucial for identifying interactive argument pairs, as is proved by the fact that all the context-aware models significantly outperform their counterpart baseline models.

- In comparison with all the context encoding methods, hierarchical context modeling outperforms the *RNN* method. Our method outperforms the hierarchical method, which proves the effectiveness of our model.

4.3 Ablation Study

The results of the ablation study are shown in the Table 3. We can find that the Path Transformer contributes greatly to the whole framework since

Methods	P@1(%) Dev Set	P@1(%) Test Set	MRR(%) Dev Set	MRR(%) Test Set
Random Guess	20	20	45.67	45.67
BiGRU	65.92	51.52	75.22	70.57
BiGRU+RNN Context	69.29	55.98	80.51	73.20
BiGRU+Hierarchical Context	70.93	57.46	82.47	73.72
VAE+Hierarchical Context	71.28	58.61	83.82	74.66
DVAE+Hierarchical Context*	73.70	61.17	85.14	76.16
BERT	73.18	61.85	84.69	76.57
BERT+Hierarchical Context	76.81	66.85	86.38	78.51
Ours	78.33	**68.75**	87.43	**80.85**

Table 2: Performance comparison for all the models on the development dataset and the test dataset, where the sign '*' represents the former state-of-the-art model. The best result on the test set is in **bold**

Model	P@1	MRR
Sentence + Concept + Align	68.75	80.85
: BiGRU as Path Encoder	67.12	79.46
: w/o Alignment Layer	65.48	79.39
: w/o Path Transformer	64.41	77.41
: w/o Concept Encoder	61.85	76.57
: w/o Sentence Encoder	51.96	68.83

Table 3: Ablation study on our proposed framework.

the model's performance drops by over 4% in P@1. This shows that besides the textual features and the concepts that directly appear in the argument, the concepts that emerge in the reasoning path are also important when considering whether two arguments have interactive relations.

5 Further Analyses

We conduct some further analyses to have a deeper understanding of the working mechanism of concept paths. Besides, we present an error analysis and a case study.

5.1 Analysis on Reasoning Path

Without ambiguity, we use positive paths and negative paths to refer to the paths that connecting the concepts in positive argument pair samples and the ones in the negative samples.

Connectivity between concept pairs First, we calculate the connectivity between each concept pair, and the results are as shown in Figure 3(a). In all the concept pairs of the two sides, the ones from a positive argument pair have a probability of 54% to form a reasoning path while the ones from a negative sample only have 41%, which conforms with the fact that an interactive argument pair mainly talk about the same topic or subject.

Path length distribution We present the distribution of the length of concept paths in Figure 3(b). The vast majority of the path lengths lie in the range from 3 to 4, while the lengths of the positive paths are generally shorter than those of the negative paths. We owe such a difference in the average length to the fact that, in positive argument pairs, the replies tend to directly use the concepts mentioned in the quotation (the reasoning path between such concept pair is hence 1). Another interpretation for the shorter average length in positive pairs is that the longer the reasoning path is, the more likely it is to become an off-topic or off-subject reply.

Path relations We further analyze the types of concept paths generated by investigating the relations appearing in the path. As discussed in the Introduction(§1), there are mainly two types of external knowledge needed to handle the interactive argument pair identification task, namely the commonsense knowledge and the causal effect knowledge. Hence, we would like to see how much proportion of these two types of knowledge occurs in the reasoning paths respectively. For common sense knowledge, we pick out the relations that contain the **be verbs** and their variants, assuming such words indicate the relation of equivalence. And as for the causal effect relations, we use a set of lexical indicators from *+/-EffectWordNet* (Choi and Wiebe, 2014), *ConnotationWordNet* (Kang et al., 2014) and *Connotation Frames* (Rashkin et al., 2016). These lexicons all evaluate the causal effect of a given predicate. The distribution of relations under

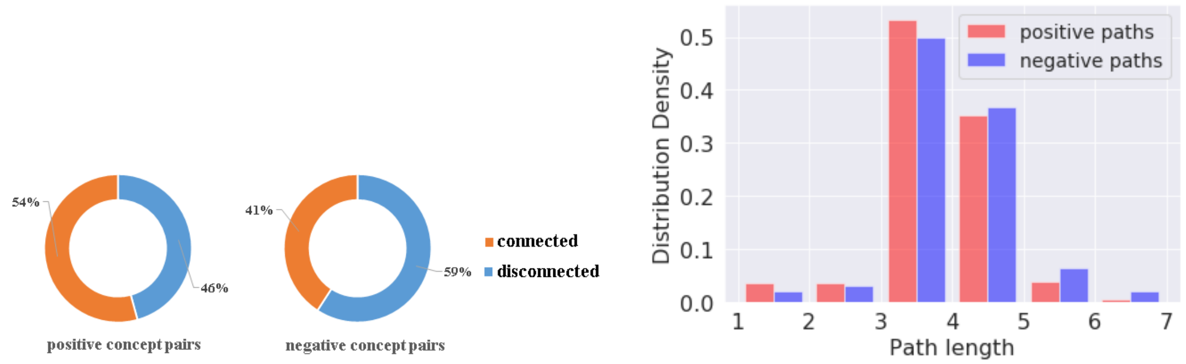

(a) The existence of reasoning paths between every concept pair in the contexts of the quotation and the reply.

(b) The distribution of path length in the positive argument pair samples and in the negative samples.

(c) The distribution of path relations in all the reasoning paths.

(d) The impact of different paths filtered by the path length on *P@1*.

Figure 3: The results of our further analysis on the reasoning paths.

such criterion is as shown in the Figure 3(c), from which we can find that near 40% of the relations belong to the common sense knowledge while 44% of the relations are of causal effect relations (31% for the positive effect and 13% for the negative effect).

Impact of path length on model performance We show the influence of the length of the path on the performance of the model in Figure 3(d). We set a threshold on the length of the path to filter concept paths used in the model. From the results, we can find that our model's performance improves significantly when the threshold is set from 3 to 4, in which most path lengths are distributed. The performance decreases when the path length is set to 5 and 6, which means it includes some noise and hurts the performance.

5.2 Error Analysis

For the instances that our model fails to predict the interactivity, we find that the problems are mainly two-fold:

- **Concept level**: For some of the failed cases, we find that around 37% of them contain at least one reply from which no concepts can be extracted, which blocks our path-finding based reasoning process. It is also the reason why our model's performance in the ablation study is lowest when removing the semantic information (i.e. BERT encoding).

- **Semantic level**: Some other failed cases share the common features that the reply does not refer to the specific term mentioned in the quotation, but gives out a more general rebuttal, e.g. [quotation] *If the president is either killed or resigns, the vice president is a horrible choice to take over the office.* [reply] *Seriously, stop this hyperbole.* Our model cannot effectively distinguish the interactivity between them, as the reply is short and has entirely no overlapping with the quotation.

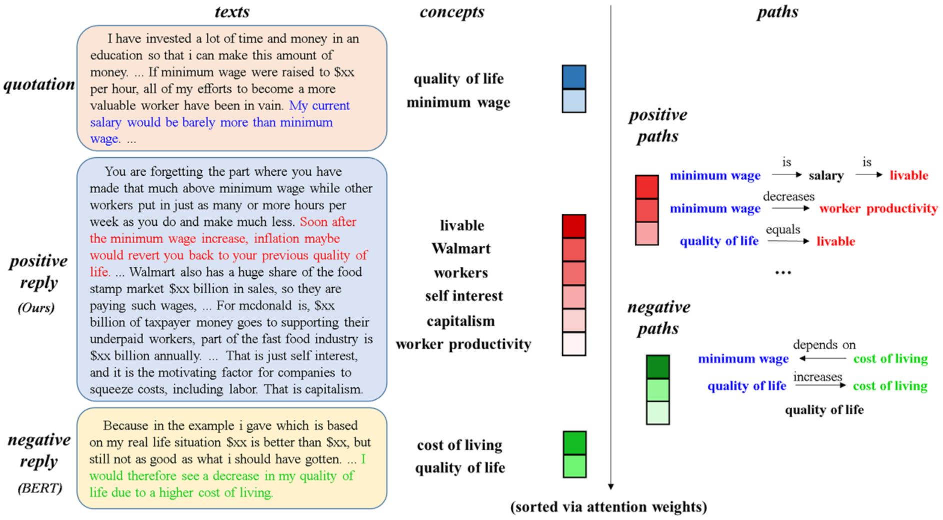

I have invested a lot of time and money in an education so that i can make this amount of money. ... If minimum wage were raised to $xx per hour, all of my efforts to become a more valuable worker have been in vain. My current salary would be barely more than minimum wage. ...

You are forgetting the part where you have made that much above minimum wage while other workers put in just as many or more hours per week as you do and make much less. Soon after the minimum wage increase, inflation maybe would revert you back to your previous quality of life. ... Walmart also has a huge share of the food stamp market $xx billion in sales, so they are paying such wages, ... For mcdonald is, $xx billion of taxpayer money goes to supporting their underpaid workers, part of the fast food industry is $xx billion annually. ... That is just self interest, and it is the motivating factor for companies to squeeze costs, including labor. That is capitalism.

Because in the example i gave which is based on my real life situation $xx is better than $xx, but still not as good as what i should have gotten. ... I would therefore see a decrease in my quality of life due to a higher cost of living.

Figure 4: Case study of an instance from the test set. Our model successfully chooses the correct reply argument from the given candidates while the BERT model gives the wrong answer. Note that the concepts and the paths are all sorted from top to bottom in terms of hierarchical attention weights.

5.3 Case study

A case study is as shown in Figure 4, where the negative reply is selected by BERT baseline. It shows that although the quotation and the negative reply share a common concept, *quality of life*, our model successfully figures out the interactive reply argument through the reasoning paths between the concepts from the two sides. All the concepts and the paths in the figure are arranged from top to bottom according to their respective attention weights. We can find that upper paths are actually highly related to the reasoning process of humans, and the irrelevant concepts such as worker productivity will automatically diminish by our hierarchical attention alignment.

6 Related Work

Dialogical argumentation mining As mentioned in Introduction(§1), our work mainly focus on dialogical argumentation mining. Among recent researches in this aspect, El Baff et al. (2020) compare content- and style-oriented classifiers on editorials to explore the effect of the writing style of editorials to the audience of different parties; Ji et al. (2019) propose the task of identifying interactive argument pairs in online debate forum such as ChangeMyView (CMV). Cheng et al. (2020) collects the text data from peer review and rebuttal process to mine the argumentative relationship entailed in such discussion; Khatib et al. (2020) constructs a monological argumentation graph by extracting knowledge from *Debatepedia.org* and use human annotation to further improve the quality of their knowledge graph. Our work obtains inspiration from the construction of Al-Khatib's knowledge graph, but adapting their method to the dialogical debating forum settings, and removing the human annotation stage to obtain an automatically generated knowledge graph.

Leveraging external knowledge in NLU Our work also lies in the general context of using external knowledge to encode sentences and paragraphs. Yang and Mitchell (2017) are among the first researches that retrieve the related entities in the external knowledge base and merge them into an LSTM encoder. Afterward, Weissenborn et al. (2017), Mihaylov and Frank (2018) and Zhang et al. (2020) mainly follows the main idea of the work to incorporate external word-level lexical knowledge to enhance the sentence embedding. Moreover, Lin et al. (2019) propose a knowledge-aware network(KagNet) that utilizes graph knowledge from *ConceptNet* to answer the commonsense questions. Compared with these methods, our work utilizes conceptual information from dialogical argumentation lexicons and conducts a reasoning process resembling human beings, which is then encoded by a path transformer, and finally aligned with the semantic information through a hierarchical attention mechanism.

7 Conclusion and Future Work

We propose a framework that imitates human's reasoning process in debating. Practically, we first construct a dialogical argumentation knowledge graph from the online debating forum ChangeMyView, by using an automatic OpenIE toolkit and conducting concept grounding with lexical resources and Wikipedia API. Then we use a path-based graph model to encode the concepts and the reasoning path between concepts from two sides of a debate and align the conceptual information with the semantic information obtained implicitly by pre-trained language model BERT. Experiments on interactive argument pair identification task show that our model can leverage the external knowledge in both effective and transparent way.

Acknowledgements

This work is partially supported by National Key Research and Development Plan of China(NO.2018YFC0830605). Besides, the authors also would like to thank all the reviewers for their suggestions and directions to this work.

References

Khalid Al Khatib, Michael Völske, Shahbaz Syed, Nikolay Kolyada, and Benno Stein. 2020. Exploiting personal characteristics of debaters for predicting persuasiveness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7067–7072, Online. Association for Computational Linguistics.

Christa SC Asterhan and Baruch B Schwarz. 2007. The effects of monological and dialogical argumentation on concept learning in evolutionary theory. *Journal of educational psychology*, 99(3):626.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020a. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020b. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 39–49, Online. Association for Computational Linguistics.

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 402–408. MIT Press.

Liying Cheng, Lidong Bing, Qian Yu, Wei Lu, and Luo Si. 2020. Argument pair extraction from peer review and rebuttal via multi-task learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7000–7011.

Yoonjung Choi and Janyce Wiebe. 2014. +/-EffectWordNet: Sense-level lexicon acquisition for opinion inference. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1181–1191, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Roxanne El Baff, Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2020. Analyzing the Persuasive Effect of Style in News Editorial Argumentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3154–3160, Online. Association for Computational Linguistics.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1625–1628. ACM.

Lu Ji, Zhongyu Wei, Jing Li, Qi Zhang, and Xuanjing Huang. 2019. Discrete argument representation learning for interactive argument pair identification. *arXiv preprint arXiv:1911.01621*.

Jun Seok Kang, Song Feng, Leman Akoglu, and Yejin Choi. 2014. ConnotationWordNet: Learning connotation over the Word+Sense network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1544–1554, Baltimore, Maryland. Association for Computational Linguistics.

Khalid Al Khatib, Yufang Hou, Henning Wachsmuth, Charles Jochim, Francesca Bonin, and Benno Stein. 2020. End-to-end argumentation knowledge graph construction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7367–7374. AAAI Press.

Jialu Li, Esin Durmus, and Claire Cardie. 2020. Exploring the role of argument structure in online debate persuasion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8905–8912, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, Yuta Koreeda, and Kohsuke Yanai. 2020. Towards better non-tree argument mining: Proposition-level biaffine parsing with task-specific parameterization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3259–3266, Online. Association for Computational Linguistics.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation frames: A data-driven investigation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 311–321, Berlin, Germany. Association for Computational Linguistics.

Jason Tyler Rolfe. 2017. Discrete variational autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200, Berlin, Germany. Association for Computational Linguistics.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, Vancouver, Canada. Association for Computational Linguistics.

Jian Yuan, Zhongyu Wei, Yixu Gao, Wei Chen, Jun Song, Donghua Zhao, Heng Wang, Zhen Hu, Shaokun Zou, Donghai Li, et al. 2021. Overview of smp-cail2020-argmine: The interactive argument-pair extraction in judgement document challenge. *Data Intelligence*, pages 1–20.

Bowen Zhang, Xiaofei Xu, Min Yang, Xiaojun Chen, and Yunming Ye. 2018. Cross-domain sentiment classification by capsule network with semantic rules. *IEEE Access*, 6:58284–58294.

Bowen Zhang, Min Yang, Xutao Li, Yunming Ye, Xiaofei Xu, and Kuai Dai. 2020. Enhancing cross-target stance detection with transferable semantic-emotion knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3188–3197, Online. Association for Computational Linguistics.

A Multi-Task Learning Framework for Multi-Target Stance Detection

Yingjie Li and Cornelia Caragea
Computer Science Department
University of Illinois at Chicago
`yli300@uic.edu`, `cornelia@uic.edu`

Abstract

Multi-target stance detection aims to identify the stance taken toward a pair of different targets from the same text, and typically, there are multiple target pairs per dataset. Existing works generally train one model for each target pair. However, they fail to learn target-specific representations and are prone to overfitting. In this paper, we propose a new training strategy under the multi-task learning setting by training one model on all target pairs, which helps the model learn more universal representations and alleviate overfitting. Moreover, in order to extract more accurate target-specific representations, we propose a multi-task learning network which can jointly train our model with a stance (dis)agreement detection task that is designed to identify agreement and disagreement between stances in paired texts. Experimental results demonstrate that our proposed model outperforms the best-performing baseline by 12.39% in macro-averaged F1-score. Our resources are publicly available on GitHub.[1]

1 Introduction

Nowadays, people often take to social media to express their stances toward specific targets (e.g., various political figures). These stances in an aggregate can provide valuable information for obtaining insight into important events such as presidential elections. The common stance detection task is to determine from a piece of text whether the author of the text is in favor of, neutral or against to a specific target, which can be categorized as single-target stance detection (STSD) (Mohammad et al., 2016; Küçük and Can, 2020; ALDayel and Magdy, 2021). More recently, since people often comment on multiple target entities in the same text, a more challenging task, i.e., multi-target stance detection (MTSD), was designed to test whether a model can accurately predict the stance toward multiple targets in the same text (Sobhani et al., 2017). For example, for the tweet in Figure 1, the author aims at expressing stance toward two targets, Donald Trump and Hillary Clinton, implied by the presence of the words “beat” and “career politicians”.

Tweet: #Trump2016 can beat #HillaryClinton as he is easily beating #JebBush ;) People R sick and tired of career politicians.

Target 1:	Donald Trump	**Stance 1**:	FAVOR
Target 2:	Hillary Clinton	**Stance 2**:	AGAINST

Figure 1: An example of multi-target stance detection.

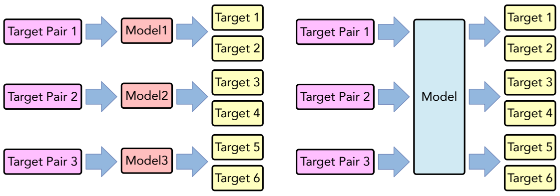

(a) (b)

Figure 2: The left figure represents the previous “Ad-hoc” training setting in which a model is trained only on a target pair. The right figure represents the proposed “Merged” training setting in which the model is trained on all target pairs.

Problem statement. Given a sentence $x = [w_1, w_2, t_1, w_3, ..., w_{l-1}, t_2, w_l]$, where t_1 and t_2 are targets, and w_i, $i = 1, ..., l$, denotes a non-target word, the goal of MTSD is to classify the stance toward these targets into one of the three classes: {FAVOR, AGAINST, NONE}.

Previous work focused on a per-target-pair training strategy, which aims to train one model for each target pair and evaluate it on the test data corresponding to that target-pair (which we call “Ad-hoc” training). The framework is illustrated in Figure 2(a). However, the model is more likely to make predictions based on specific words without

---

[1] `https://github.com/chuchun8/MTSD`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2320–2326
August 1–6, 2021. ©2021 Association for Computational Linguistics

fully considering the target information, and hence, to overfit in the "Ad-hoc" training setting. To address this, as shown in Figure 2(b), we propose a "Merged" training strategy by training one model on data from all target pairs, which helps the model learn more universal representations on the whole dataset and alleviate overfitting. Furthermore, in order to extract more accurate target-specific representations, we propose a multi-task learning network which is able to jointly train our model with a stance (dis)agreement detection task that is designed to identify agreement and disagreement between stances expressed in paired-target sentences. Results show that the proposed "Merged" training setting together with identifying whether the author expresses the same stance toward two targets are beneficial to the MTSD.

Our contributions include: 1) We propose a "Merged" training strategy for MTSD and show that models fine-tuned on the pre-trained BERTweet (Nguyen et al., 2020) perform substantially better than strong baselines. Meanwhile, the decrease in performance can be widely observed in baseline results when using the "Merged" training strategy, making it a more challenging evaluation for MTSD; 2) We propose a multi-task learning network which considers the stance (dis)agreement detection task as an auxiliary task to further improve the performance of our proposed model; 3) Our proposed model outperforms the best-performing baseline by 12.39% in macro-averaged F1-score.

2 Related Work

Sobhani et al. (2017) introduced the MTSD task and presented the first dataset. They also proposed an attention-based encoder-decoder (Seq2Seq) model that predicts stance labels by focusing on important parts of a tweet. Wei et al. (2018a) proposed a dynamic memory network for detecting stance. First, target-specific attention is extracted for each target. Then, a shared external memory module that maintains useful information for targets is dynamically updated. This model achieves state-of-the-art performance on the multi-target stance dataset of Sobhani et al. (2017). We used the above two works as strong baselines for our evaluation.

Sobhani et al. (2017) and Wei et al. (2018a) deal with MTSD by training one model for each target pair and the model predicts the stance toward two targets simultaneously. However, we can also solve this task by treating it as a special case of single-target stance detection (STSD). Instead of training a model that receives two targets and a sentence as an input, we train two STSD models that receive one target and a sentence as an input, on each target pair. For the example in Figure 1, we train one STSD model for target "Donald Trump" and train another model for "Hillary Clinton" in a STSD manner.

Previous studies on STSD often employ feature engineering (Sobhani et al., 2016; Mohammad et al., 2016), Convolutional Neural Network (CNN) (Vijayaraghavan et al., 2016; Wei et al., 2016) and Recurrent Neural Network (RNN) (Zarrella and Marsh, 2016) to predict the stance for a given target. One of the major limitations is that they do not consider the target information. To address this, Augenstein et al. (2016) proposed a conditional BiLSTM encoder that learns tweet representations conditioned on the respective target. More recently, inspired by the attention mechanism (Bahdanau et al., 2015), various target-specific attention-based approaches (Du et al., 2017; Sun et al., 2018; Wei et al., 2018b; Li and Caragea, 2019, 2021) have been proposed to connect the target with the sentence representation, which is similar to aspect-based sentiment analysis (Hazarika et al., 2018; Majumder et al., 2018; Lin et al., 2019; Song et al., 2019). We compare the baseline models of STSD and MTSD with our proposed model in §4.4 using both "Ad-hoc" and "Merged" settings.

3 Approach

Previous work focused on an "Ad-hoc" training strategy, which fails to explore the potential of the training data and is unable to learn universal representations of targets. Moreover, we observe that STSD models that do not consider target information can still perform well on the multi-target dataset, which makes MTSD easier. Therefore, in order to learn more universal representations and better evaluate the performance of models on MTSD, we propose a "Merged" training strategy by training one model on all target pairs. More specifically, the model is trained on training data combined from all target pairs, and tested on each target pair separately to be compared with the results of the "Ad-hoc" strategy. Our proposed training strategy can be considered as a multi-task learning approach that helps the pre-trained language models to learn more generalized text representations by sharing the domain-specific information

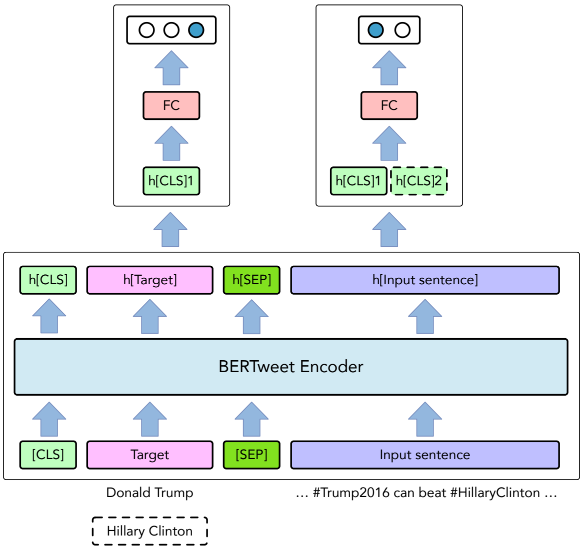

Figure 3: Model architecture.

across the related target pairs.

BERTweet (Nguyen et al., 2020) is a large-scale language model pre-trained on 850M tweets. BERTweet follows the training procedure of RoBERTa (Liu et al., 2019) and uses the same model configuration with BERT-base (Devlin et al., 2019). We fine-tune the pre-trained BERTweet on the multi-target dataset. The model architecture is shown in Figure 3. Given an input data x and a target t (t is either target 1 or target 2 in Figure 1), we formulate the input as a sequence s = $[[CLS]$ t $[SEP]$ $x]$ where $[CLS]$ encodes the sentence representation and $[SEP]$ is used to separate the target t and sentence x. We utilize the $[CLS]$ token $h_{[CLS]}$ to get the prediction $\hat{p}_1$ toward target t.

In order to learn better target-specific representations, we propose a multi-task learning network that can jointly train our model with a stance (dis)agreement detection task, which is a binary classification task where the label is 1 when the author expresses the same stance toward two targets (e.g., “FAVOR” and “FAVOR”) and 0 otherwise (e.g., “FAVOR” and “AGAINST”). More specifically, given an input data x and two targets t_1 and t_2, we formulate the inputs as $[[CLS]$ t_1 $[SEP]$ $x]$ and $[[CLS]$ t_2 $[SEP]$ $x]$. Then we leverage the representations of $[CLS]$ token of two sequences to detect whether the author of the text expresses the same stance toward two targets. The (dis)agreement class probability $\hat{p}_2$ can be computed as follows:

$$\hat{p}_2 = softmax(W_2 f(W_1 [h_{[CLS]1}; h_{[CLS]2}] \\ + b_1) + b_2)$$

Target Pair	Total	Train	Dev	Test
Trump-Clinton	1,722	1,240	177	355
Trump-Cruz	1,317	922	132	263
Clinton-Sanders	1,366	957	137	272
Total	4,455	3,119	446	890

Table 1: Distribution of instances in our dataset.

where $W_1 \in \mathbb{R}^{d_h * 2d_h}$, $W_2 \in \mathbb{R}^{2 * d_h}$, $b_1 \in \mathbb{R}^{d_h}$, $b_2 \in \mathbb{R}^2$, d_h is the size of the hidden dimension, f is an activation function, semicolon denotes vector concatenation. Note that the main task is to identify the stance toward target t_1. The target t_2 is only used in auxiliary task. Similarly, we predict the stance toward target t_2 in main task where t_1 is only used in auxiliary task.

Let D be a labeled training dataset and D_j be a mini-batch for the MTSD, and let y_1 and y_2 denote the true labels for stance detection task and (dis)agreement task, respectively. The cross-entropy loss is used to train the model. Let L_1 and L_2 be the loss of stance detection task and (dis)agreement task, respectively. Then the final loss is:

$$L = \sum_{i \in D_j} L_1(y_1^i, \hat{p}_1^i) + \alpha \sum_{i \in D_j} L_2(y_2^i, \hat{p}_2^i)$$

where i is the index of a data sample and α is a hyper-parameter to account for the importance of the auxiliary task. α is set to 0.5 in our experiments.

4 Experiments

4.1 Dataset

To test the performance of our proposed model, we use the multi-target stance dataset (Sobhani et al., 2017) of tweets annotated with stance labels with respect to two targets. This dataset contains three different target pairs: Donald Trump and Hillary Clinton, Donald Trump and Ted Cruz, Hillary Clinton and Bernie Sanders. Table 1 provides dataset statistics. Each tweet has two stance labels concerning two targets and each label has one of the values: “FAVOR”, “AGAINST” or “NONE”.

4.2 Evaluation Metrics

F_{avg}^p and macro-average of F1-score (F_{macro}) are adopted to evaluate the performance of our baseline models. First, the F1-score of label “Favor” and “Against” is calculated as follows:

$$F_{favor} = \frac{2P_{favor}R_{favor}}{P_{favor} + R_{favor}}$$

$$F_{against} = \frac{2P_{against}R_{against}}{P_{against} + R_{against}}$$

where P and R are precision and recall, respectively. After that, the F_{avg} is calculated as:

$$F_{avg} = \frac{F_{favor} + F_{against}}{2}$$

For each target pair, we compute the F_{avg} for each target and use the F^{p}_{avg}, which is calculated as the average of F_{avg} on two targets, as our evaluation metric. Moreover, we get F_{macro} by averaging F^{p}_{avg} on all target pairs.

4.3 Baseline Methods

First, we compare the proposed model with the following baselines from STSD.

BiLSTM (Schuster and Paliwal, 1997): A BiLSTM model that takes sentences as inputs without considering the target information.

CNN (Kim, 2014): The vanilla CNN that has the same input format with BiLSTM. Similarly, target information is not considered in this model.

TAN (Du et al., 2017): TAN is an attention-based LSTM that extracts target specific features.

BiCE (Augenstein et al., 2016): A BiLSTM model that uses conditional encoding for stance detection. The target information is first encoded by using a BiLSTM and the tweet is then encoded by another BiLSTM, whose state is initialised with the hidden representation of the target.

GCAE (Xue and Li, 2018): A model that is based on CNNs and gating mechanism, which is designed to block target-unrelated information.

PGCNN (Huang and Carley, 2018): Similar to GCAE, PGCNN is based on gated convolutional networks and encodes target information by generating target-sensitive filters.

The second group contains baselines from MTSD.

Seq2Seq (Sobhani et al., 2017): An attention-based encoder-decoder model that generates stance labels according to different parts of a tweet.

DMAN (Wei et al., 2018a): Using attention and memory modules to extract important information for detecting stance.

We compare the baselines of STSD and MTSD with our proposed models.

Model	**Tr-Cl**	**Tr-Cr**	**Cl-Sa**	**F_{macro}**
Merged				
BiLSTM	43.33	47.51	41.86	44.24
CNN	43.22	49.21	41.22	44.55
GCAE	59.07	54.28	56.13	56.49
PGCNN	59.18	54.62	50.59	54.80
TAN	43.88	50.46	45.63	46.66
BiCE	53.73	51.00	45.84	50.19
DMAN	57.43	52.62	53.87	54.64
BERTweet	67.38†	70.30†	65.64†	67.77
BERTweet-A	**69.22**†‡	**70.73**†	**69.00**†‡	**69.65**
Ad-hoc				
BiLSTM	58.16	52.75	52.67	54.52
CNN	59.75	55.68	56.13	57.19
GCAE	59.78	56.07	55.92	57.26
PGCNN	56.99	54.19	55.05	55.41
TAN	58.33	54.32	53.16	55.27
BiCE	58.67	53.77	51.87	54.77
Seq2Seq	56.60*	53.12*	54.72*	54.81*
DMAN	60.05	54.27	52.57	55.63
BERTweet	64.29†	56.44	57.80†	59.51
BERTweet-A	**65.55**†	**57.96**†	**58.17**†	**60.56**

Table 2: Comparison with the baselines on the multi-target stance dataset (%). $*$: the result is from the original paper. †: the proposed models improve the best baseline at $p < 0.05$ with two-tailed t-test. ‡: the BERTweet-A improves the BERTweet at $p < 0.05$ with two-tailed t-test. F_{macro} is the average of all target pairs. Bold scores are best overall.

BERTweet We fine-tune the BERTweet model using “Merged” and “Ad-hoc” training strategies. The pre-trained BERTweet model is fine-tuned under the PyTorch framework. When fine-tuning, the batch size is 32 and maximum sequence length is 128. We use AdamW optimizer (Loshchilov and Hutter, 2019) and the learning rate is 2e-5.

BERTweet-A BERTweet is further improved by joint training with another stance detection task that identifies agreement and disagreement between stances in “Merged” and “Ad-hoc” training settings.

4.4 Results and Analysis

Main Results Table 2 shows the results of the comparison of our proposed models with the baselines mentioned above by using the proposed training strategy “Merged” and the “Ad-hoc” training strategy. We make the following observations.

First, the performance of baseline models that perform well in the “Ad-hoc” training setting drops heavily in our proposed “Merged” setting, especially for the BiLSTM and CNN. Specifically, the F_{macro} of BiLSTM and CNN drops by 10.28% and 12.64%, respectively. The results indicate that baseline models overfit the training data quite heavily

Model	Donald Trump
BERTweet-adhoc	46.75
BERTweet-merged	**52.51**†

Table 3: Performance comparison of models on the target "Donald Trump" of SemEval 2016 stance dataset (%). †: the proposed BERTweet-merged improves the BERTweet-adhoc at $p < 0.05$ with two-tailed t-test. Bold scores are best overall.

and our proposed "Merged" training strategy can serve as a better evaluation method to test whether the model learns target-specific features.

Second, different from other baselines suffering significant performance drops, BERTweet performs better in the "Merged" setting. Training all target pairs improves the F_{macro} of BERTweet from 59.51% to 67.77%, which demonstrates that BERTweet learns more universal representations with respect to targets by leveraging the data of multiple target pairs. Moreover, joint training with stance (dis)agreement detection task further improves the F_{macro} of BERTweet from 67.77% to 69.65% in the "Merged" setting. Similarly, in the "Ad-hoc" setting, the F_{macro} of BERTweet is improved from 59.51% to 60.56%, indicating that this auxiliary task is beneficial to the MTSD in both settings and helps the model put more attention on the target-related words.

Third, BERTweet-A of the "Merged" setting significantly outperforms the best-performing baseline by 12.39% in F_{macro}, showing the effectiveness of the proposed model.

Generalization Analysis To test the generalization ability of the BERTweet of the "Merged" setting (which we call BERTweet-merged), we train and validate the BERTweet-merged without auxiliary agreement task on the whole multi-target dataset and test it on the target "Donald Trump" of SemEval 2016 dataset (Mohammad et al., 2016) where an overall shift in the distribution of words and topics can be observed. Moreover, we train and validate the BERTweet-adhoc (BERTweet in "Ad-hoc" setting) on the target "Donald Trump" of multi-target dataset and test it on the same set of SemEval 2016 dataset to be compared with BERTweet-merged. The results are shown in Table 3. From the table, we can observe that BERTweet-merged significantly outperforms BERTweet-adhoc on the SemEval 2016 dataset, which indicates that the BERTweet model trained in the "Merged" setting shows better generalization ability than the BERTweet model trained in the "Ad-hoc" setting.

5 Conclusion

In this paper, we presented a comprehensive investigation into multi-target stance detection (MTSD) and proposed a more challenging task that trains a single model on data from all target pairs instead of training a model per target pair. The new training strategy can alleviate overfitting and help the model learn more universal representations by using the data of all target pairs. Moreover, we proposed to integrate a stance (dis)agreement detection module into the proposed model as an auxiliary task to gain more accurate representations of targets. Experimental results show that the proposed model outperforms the best-performing baseline by a large margin and demonstrates its effectiveness even in the face of a more challenging evaluation. Future work includes extending the proposed training strategy and (dis)agreement task to more stance detection tasks and datasets.

Acknowledgments

We thank the National Science Foundation for support from grants IIS-1912887 and IIS-1903963 which sponsored the research in this study. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF. We also thank Amazon Web Services for support for the computational resources. We are grateful to our reviewers for their insightful comments.

References

Abeer ALDayel and Walid Magdy. 2021. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3988–3994.

Devamanyu Hazarika, Soujanya Poria, Prateek Vij, Gangeshwar Krishnamurthy, Erik Cambria, and Roger Zimmermann. 2018. Modeling inter-aspect dependencies for aspect-based sentiment analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 266–270.

Binxuan Huang and Kathleen Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1091–1096.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Dilek Küçük and Fazli Can. 2020. Stance detection: A survey. *ACM Comput. Surv.*, 53(1):1–37.

Yingjie Li and Cornelia Caragea. 2019. Multi-task stance detection with sentiment and stance lexicons. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6298–6304.

Yingjie Li and Cornelia Caragea. 2021. Target-aware data augmentation for stance detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1850–1860.

Peiqin Lin, Meng Yang, and Jianhuang Lai. 2019. Deep mask memory network with semantic dependency and context moment for aspect level sentiment classification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5088–5094.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md. Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3402–3411.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 551–557.

Parinaz Sobhani, Saif Mohammad, and Svetlana Kiritchenko. 2016. Detecting stance in tweets and analyzing its interaction with sentiment. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 159–169.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. DeepStance at SemEval-2016 task 6: Detecting stance in tweets using character and word-level CNNs. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 413–419.

Penghui Wei, Junjie Lin, and Wenji Mao. 2018a. Multi-target stance detection via a dynamic memory-augmented network. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1229–1232.

Penghui Wei, Wenji Mao, and Daniel Zeng. 2018b. A target-guided neural memory model for stance detection in twitter. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pages 1–8.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at SemEval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 384–388.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 task 6: Transfer learning for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 458–463.

Confidence-Aware Scheduled Sampling for Neural Machine Translation

Yijin Liu[1*] **, Fandong Meng**[2]**, Yufeng Chen**[1] **Jinan Xu**[1†]**, and Jie Zhou**[2]
[1]Beijing Jiaotong University, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
`adaxry@gmail.com`
`{fandongmeng, withtomzhou}@tencent.com`
`{jaxu,chenyf}@bjtu.edu.cn`

Abstract

Scheduled sampling is an effective method to alleviate the exposure bias problem of neural machine translation. It simulates the inference scene by randomly replacing ground-truth target input tokens with predicted ones during training. Despite its success, its critical schedule strategies are merely based on training steps, ignoring the real-time model competence, which limits its potential performance and convergence speed. To address this issue, we propose confidence-aware scheduled sampling. Specifically, we quantify real-time model competence by the confidence of model predictions, based on which we design fine-grained schedule strategies. In this way, the model is exactly exposed to predicted tokens for high-confidence positions and still ground-truth tokens for low-confidence positions. Moreover, we observe vanilla scheduled sampling suffers from degenerating into the original teacher forcing mode since most predicted tokens are the same as ground-truth tokens. Therefore, under the above confidence-aware strategy, we further expose more noisy tokens (*e.g.,* wordy and incorrect word order) instead of predicted ones for high-confidence token positions. We evaluate our approach on the Transformer and conduct experiments on large-scale WMT 2014 English-German, WMT 2014 English-French, and WMT 2019 Chinese-English. Results show that our approach significantly outperforms the Transformer and vanilla scheduled sampling on both translation quality and convergence speed.

1 Introduction

Neural Machine Translation (NMT) has made promising progress in recent years (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017). Generally, NMT models are trained to maximize the likelihood of the next token given previous golden tokens as inputs, *i.e.,* teacher forcing (Goodfellow et al., 2016). However, at the inference stage, golden tokens are unavailable. The model is exposed to an unseen data distribution generated by itself. This discrepancy between training and inference is named as the *exposure bias* problem (Ranzato et al., 2016).

Many techniques have been proposed to alleviate the exposure bias problem. To our knowledge, they mainly fall into two categories. The one is sentence-level training, which treats the sentence-level metric (*e.g.,* BLEU) as a reward, and directly maximizes the expected rewards of generated sequences (Ranzato et al., 2016; Shen et al., 2016; Rennie et al., 2017). Although intuitive, they generally suffer from slow and unstable training due to the high variance of policy gradients and the credit assignment problem (Sutton, 1984; Liu et al., 2018; Wang et al., 2018). Another category is sampling-based approaches, aiming to simulate the data distribution of reference during training. Scheduled sampling (Bengio et al., 2015) is a representative method, which samples tokens between golden references and model predictions with a scheduled probability. Zhang et al. (2019) further refine the sampling space of scheduled sampling with predictions from beam search. Mihaylova and Martins (2019) and Duckworth et al. (2019) extend scheduled sampling to the Transformer with a novel two-pass decoding architecture.

Although these sampling-based approaches have been shown effective, most of them schedule the sampling probability based on training steps. We argue this schedule strategy has two following limitations: 1) It is far from exactly reflecting the real-time model competence; 2) It is only based on training steps and equally treat all token positions, which is too coarse-grained to guide the

---

[*] This work was done when Yijin Liu was interning at Pattern Recognition Center, WeChat AI, Tencent Inc, China

[†] Jinan Xu is the corresponding author of the paper.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2327–2337
August 1–6, 2021.

sampling selection for each target token. These two limitations yield an inadequate and inefficient schedule strategy, which hinders the potential performance and convergence speed of vanilla scheduled sampling-based approaches.

To address these issues, we propose confidence-aware scheduled sampling. Specifically, we take the model prediction confidence as the assessment of real-time model competence, based on which we design fine-grained schedule strategies. Namely, we sample predicted tokens as target inputs for high-confidence positions and still ground-truth tokens for low-confidence positions. In this way, the NMT model is exactly exposed to corresponding tokens according to its real-time competence rather than coarse-grained predefined patterns. Additionally, we observe that most predicted tokens are the same as ground-truth tokens due to teacher forcing[1], degenerating scheduled sampling to the original teacher forcing mode. Therefore, we further expose more noisy tokens (Meng et al., 2020) (*e.g.,* wordy and incorrect word order) instead of predicted ones for high-confidence token positions. Experimentally, we evaluate our approach on the Transformer (Vaswani et al., 2017) and conduct experiments on large-scale WMT 2014 English-German (EN-DE), WMT 2014 English-French (EN-FR), and WMT 2019 Chinese-English (ZH-EN).

The main contributions of this paper can be summarized as follows[2]:

- To the best of our knowledge, we are the first to propose confidence-aware scheduled sampling for NMT, which exactly samples corresponding tokens according to the real-time model competence rather than coarse-grained predefined patterns.

- We further explore to sample more noisy tokens for high-confidence token positions, preventing scheduled sampling from degenerating into the original teacher forcing mode.

- Our approach significantly outperforms the Transformer by 1.01, 1.03, 0.98 BLEU and outperforms the stronger scheduled sampling by 0.51, 0.41, and 0.58 BLEU on EN-DE, EN-FR, and ZH-EN, respectively. Our approach speeds up model convergence about 3.0× faster than the Transformer and about 1.8× faster than vanilla scheduled sampling.

- Extensive analyses indicate the effectiveness and superiority of our approach on longer sentences. Moreover, our approach can facilitate the training of the Transformer model with deeper decoder layers.

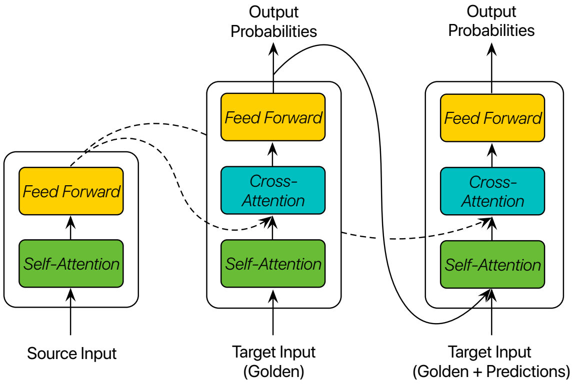

Figure 1: Scheduled sampling for the transformer with two-pass decoding (Mihaylova and Martins, 2019).

2 Background

2.1 Neural Machine Translation

Given a pair of source language $\mathbf{X} = \{x_1, x_2, \cdots, x_m\}$ with m tokens and target language $\mathbf{Y} = \{y_1, y_2, \cdots, y_n\}$ with n tokens, neural machine translation aims to model the following translation probability:

$$\begin{aligned} P(\mathbf{Y}|\mathbf{X}) &= \prod_{t=1}^{n} P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \theta) \\ &= \sum_{t=1}^{n} \log P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \theta) \end{aligned} \tag{1}$$

where t is the index of target tokens, $\mathbf{y}_{<t}$ is the partial translation before y_t, and θ is model parameter. In the training stage, $\mathbf{y}_{<t}$ are ground-truth tokens, and this procedure is also known as teacher forcing. The translation model is generally trained with maximum likelihood estimation (MLE).

2.2 Scheduled Sampling for the Transformer

Scheduled sampling is initially designed for Recurrent Neural Networks (Bengio et al., 2015), and further modifications are needed when applied to the Transformer (Mihaylova and Martins, 2019;

---

[1] We observe that about 70% tokens are correctly predicted in WMT14 EN-DE training data.

[2] Codes are available at `https://github.com/Adaxry/conf_aware_ss4nmt`.

Duckworth et al., 2019). As shown in Figure 1, we follow the two-pass decoding architecture. In the first pass, the model conducts the same as a standard NMT model. Its predictions are used to simulate the inference scene[3]. In the second pass, inputs of the decoder $\widetilde{\mathbf{y}}_{<t}$ are sampled from predictions of the first pass and ground-truth tokens with a certain probability. Finally, predictions of the second pass are used to calculate the cross-entropy loss, and Equation (1) is modified as follow:

$$P(\mathbf{Y}|\mathbf{X}) = \sum_{t=1}^{n} logP(y_t|\widetilde{\mathbf{y}}_{<t}, \mathbf{X}, \theta) \qquad (2)$$

Note that the two decoders are identical and share the same parameters. At inference, only the first decoder is used, that is just the standard Transformer. How to schedule the above probability of sampling tokens is the key point, which is exactly what we aim to improve in this paper.

2.3 Decay Strategies on Training Steps

Existing schedule strategies are based on training steps (Bengio et al., 2015; Zhang et al., 2019). As the number of the training step i increases, the model should be exposed to its own predictions more frequently. At the i-th training step, the probability of sampling golden tokens $f(i)$ is calculated as follow:

- Linear Decay: $f(i) = \max(\epsilon, ki + b)$, where ϵ is the minimum value, and $k < 0$ and b are respectively the slope and offset of the decay.
- Exponential Decay: $f(i) = k^i$, where $k < 1$ is the radix to adjust the sharpness of the decay.
- Inverse Sigmoid Decay: $f(i) = \frac{k}{k+e^{\frac{i}{k}}}$, where e is the mathematical constant, and $k \geq 1$ is a hyperparameter to adjust the sharpness of the decay.

We draw visible examples for different decay strategies in Figure 2.

3 Approaches

In this section, we firstly describe how to estimate model confidence at each token position. Secondly, we elaborate the fine-grained schedule strategy based on model confidence. Finally, we explore to sample more noisy tokens instead of predicted tokens for high-confidence positions.

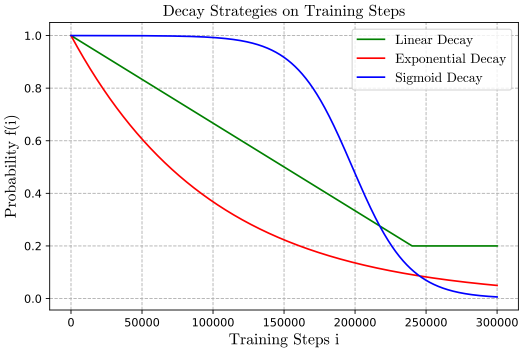

Figure 2: Examples of different decay strategies $f(i)$.

3.1 Model Confidence Estimation

We explore two approaches to estimate model confidence at each token position.

Predicted Translation Probability (PTP). Current NMT models are well-calibrated with regularization techniques in the training setting (Ott et al., 2018; Müller et al., 2019; Wang et al., 2020). Namely the predicted translation probability can directly serve as the model confidence. At the t-th target token position, we calculate the model confidence $conf(t)$ as follow:

$$conf(t) = P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \theta) \qquad (3)$$

Since we base our approach on the Transformer with two-pass decoding (Mihaylova and Martins, 2019; Duckworth et al., 2019), above predicted translation probability can be directly obtained in the first-pass decoding (shown in Figure 1), causing no additional computation costs.

Monte Carlo Dropout Sampling. The model confidence can be quantified by Bayesian neural networks (Buntine and Weigend, 1991; Neal, 2012), which place distributions over the weights of neural networks. We adopt widely used Monte Carlo dropout sampling (Gal and Ghahramani, 2016; Wang et al., 2019b) to approximate Bayesian inference. Given a batch of training data and current NMT model parameterized by θ, we repeatedly conduct forward propagation K times[4]. On the k-th propagation, part of neurons $\hat{\theta}^{(k)}$ in network

---

[3] Following Goyal et al. (2017), model predictions are the weighted sum of target embeddings over output probabilities. As model predictions cause a mismatch with golden tokens, they can simulate translation errors of the inference scene.

[4] We empirically set K to 5 following Wan et al. (2020).

θ are randomly deactivated. Eventually, we obtain K sets of model parameters $\{\hat{\theta}^{(k)}\}_{k=1}^{K}$ and corresponding translation probabilities. We use the expectation or variance of translation probabilities to estimate the model confidence (Wang et al., 2019b). Intuitively, the higher expectation or, the lower variance of translation probabilities reflects higher model confidence. Formally at the t-th token position, we estimate the model confidence $conf(t)$ that calculated by the expectation of translation probabilities:

$$conf(t) = \mathbb{E}\left[P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \hat{\theta}^{(k)})\right]_{k=1}^{K} \tag{4}$$

We also use the variance of translation probabilities to estimate the model confidence $conf(t)$ as an alternative:

$$conf(t) = 1 - \mathrm{Var}\left[P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \theta)\right]_{k=1}^{K} \tag{5}$$

where $\mathrm{Var}[\cdot]$ denotes the variance of a distribution that calculated following the setting in (Wang et al., 2019b; Zhou et al., 2020). We will further analyze the effect of different confidence estimations in Section 4.2.

3.2 Confidence-Aware Scheduled Sampling

The confidence score $conf(t)$ quantifies whether the current NMT model is confident or hesitant on predicting the t-th target token. We take $conf(t)$ as exact and real-time information to conduct a fine-grained schedule strategy in each training iteration. Specifically, a lower $conf(t)$ indicates that the current model θ still struggles with the teacher forcing mode for the t-th target token, namely underfitting for the conditional probability $P(y_t|\mathbf{y}_{<t}, \mathbf{X}, \theta)$. Thus we should keep feeding ground-truth tokens for learning to predict the t-th target token. Conversely, a higher $conf(t)$ indicates the current model θ has learned well the basic conditional probability under teacher forcing. Thus we should empower the model with the ability to cope with the exposure bias problem. Namely, we take inevitably erroneous model predictions as target inputs for learning to predict the t-th target.

Formally, in the second-pass decoding, the above fine-grained schedule strategy is conducted at all decoding steps simultaneously:

$$y_{t-1} = \begin{cases} y_{t-1} & \text{if } conf(t) \leq t_{golden} \\ \hat{y}_{t-1} & \text{else} \end{cases} \tag{6}$$

Dataset	Size (M)	Valid / Test set
WMT14 EN-DE	4.5	newstest 2013 / 2014
WMT14 EN-FR	36	newstest 2013 / 2014
WMT19 ZH-EN	20	newstest 2018 / 2019

Table 1: Dataset statistics in our experiments.

where t_{golden} is a threshold to measure whether $conf(t)$ is high enough (*e.g.,* 0.9) to sample the predicted token $\hat{y}_{t-1}$.

3.3 Confidence-Aware Scheduled Sampling with Target Denoising

Considering predicted tokens are obtained from the teacher forcing model, most predicted tokens (*e.g.,* about 70% tokens in WMT14 EN-DE) are the same as ground-truth tokens, which degenerate the scheduled sampling to the original teacher forcing. Although previous study (Zhang et al., 2019) have proposed to address this issue by using predictions from beam search, it conducts very slowly (about 4× slower than ours) due to the autoregressive property of beam search decoding. To avoid the above degeneration problem while preserving computational efficiency, we try to add more noisy tokens instead of predicted tokens for high-confidence positions. Inspired by Meng et al. (2020), we replace ground-truth y_{t-1} with a random token y_{rand} of the current target sentence, which can simulate wordy and incorrect word order phenomena that occur at inference. Considering y_{rand} is more difficult[5] to learn than $\hat{y}_{t-1}$, we only adopt the noisy y_{rand} for higher confidence positions. Therefore, the fine-grained schedule strategy in Equation 6 is extended to:

$$y_{t-1} = \begin{cases} y_{t-1} & \text{if } conf(t) \leq t_{golden} \\ \hat{y}_{t-1} & \text{if } t_{golden} < conf(t) \leq t_{rand} \\ y_{rand} & \text{if } conf(t) > t_{rand} \end{cases} \tag{7}$$

where t_{rand} is a threshold to measure whether $conf(t)$ is high enough (*e.g.,* 0.95) to sample the random target token y_{rand}. We provide detailed selections about t_{golden} and t_{rand} in Section 4.2.

---

[5] Given a pre-trained Transformer$_{base}$ model, we respectively replace ground-truth tokens with predicted tokens $\hat{y}$ or random tokens y_{rand} with the same rate, and measure such difficulty by the increment of model perplexity. We observe that y_{rand} yields about 15% higher model perplexity than $\hat{y}$.

4 Experiments

We conduct experiments on three large-scale WMT 2014 English-German (EN-DE), WMT 2014 English-French (EN-FR), and WMT 2019 Chinese-English (ZH-EN) translation tasks. We respectively build a shared source-target vocabulary for the EN-DE and EN-FR datasets, and unshared vocabularies for the ZH-EN dataset. We apply byte-pair encoding (Sennrich et al., 2016) with 32k merge operations for all datasets. More datasets statistics are listed in Table 1.

4.1 Implementation Details

Training Setup. We train the Transformer$_{base}$ and Transformer$_{big}$ models (Vaswani et al., 2017) with the open-source THUMT (Zhang et al., 2017). All Transformer models are first trained by teacher forcing with 100k steps, and then trained with different training objects or scheduled sampling approaches for 300k steps. All experiments are conducted on 8 NVIDIA Tesla V100 GPUs, where each is allocated with a batch size of approximately 4096 tokens. We use Adam optimizer (Kingma and Ba, 2014) with 4000 warmup steps. During training and the Monte Carlo Dropout process, we set dropout (Srivastava et al., 2014) rate to 0.1 for the Transformer$_{base}$ and 0.3 for the Transformer$_{big}$.

Evaluation. We set the beam size to 4 and the length penalty to 0.6 during inference. We use *multibleu.perl* to calculate case-sensitive BLEU scores for WMT14 EN-DE and EN-FR, and use *mteval-v13a.pl* to calculate case-sensitive BLEU scores for WMT19 ZH-EN. We use the paired bootstrap resampling methods (Koehn, 2004) to compute the statistical significance of test results.

4.2 Hyperparameter Experiments

In this section, we elaborate hyperparameters settings involved in our approaches according to the performance on the validation set of WMT14 EN-DE, and share these settings for all WMT tasks.

Different Confidence Estimations. In this section, we analyze effects of different estimations for model confidence described in Section 3.1. As shown in Table 2, we observe that Monte Carlo dropout sampling based approaches (*i.e.,* expectation and variance of translation probabilities) achieve comparable or marginally better translation quality than PTP. However, since Monte Carlo dropout sampling based approaches need additional passes for forward propagation, which yields about 2.7× computation costs than the Transformer$_{base}$. On the contrary, PTP only causes marginal additional computation costs (1.3×) than the Transformer$_{base}$, as PTP can be directly obtained in the first pass decoding. Considering the trade-off between training efficiency and final performance, we use PTP to estimate model confidence by default in the following experiments.

Methods	**Training Cost**	**BLEU**	**Δ**
Transformer$_{base}$	ref.	27.10	ref.
+ PTP	1.3×	28.15	+1.05
+ Expectation	2.7×	28.15	+1.05
+ Variance	2.7×	28.20	+1.10

Table 2: BLUE scores (%) on the validation set of WMT14 EN-DE with different confidence estimations. 'Training Cost' is calculated by the total training time until models convergence on 8 NVIDIA V100 GPUs. 'PTP' refers to PTP-based confidence estimation in Equation (3). 'Expectation' and 'Variance' refers to Monte Carlo dropout sampling-based confidence estimation in Equation (4) and (5), respectively. 'ref.' is short for the reference baseline.

Thresholds Settings. There are two important hyperparameters in our approaches, namely the two threshold t_{golden} and t_{rand} that determine token selections in Equation (7). In our preliminary experiments, we observe our approach is relatively not sensitive to t_{golden}, thus we firstly fix t_{golden} to a modest value, i.e., 0.5 and analyze effects when t_{rand} ranging from 0.5 to 0.95. As the red line is shown in Figure 3, we observe that a rapid improvement in performance with the growth of t_{rand}. Therefore, we decide to set t_{rand} to 0.95 and then analyze effects when t_{golden} ranging from 0.5 to 0.95. As the blue line is shown in Figure 3, the model performance gently rises with the growth of t_{golden} and finally achieves its peak when $t_{golden} = 0.9$. Thus we finally set t_{golden} to 0.9.

4.3 Systems

Mixer. A sequence-level training algorithm for text generations by combining both REINFORCE and cross-entropy (Ranzato et al., 2016).

Minimal Risk Training. Minimal Risk Training (MRT) (Shen et al., 2016) introduces evaluation metrics (*e.g.,* BLEU) as loss functions and aims to minimize expected loss on the training data.

Model	BLEU		
	EN-DE	ZH-EN	EN-FR
Transformer$_{base}$ (Vaswani et al., 2017)	27.30	–	38.10
Transformer$_{base}$ (Vaswani et al., 2017) †	27.90	24.97	40.30
+ Mixer (Ranzato et al., 2016) †	28.54	25.28	40.57
+ Minimal Risk Training (Shen et al., 2016) †	28.55	25.23	40.82
+ TeaForN (Goodman et al., 2020)	27.90	–	40.84
+ TeaForN (Goodman et al., 2020) †	28.60	25.45	40.94
+ Self-paced learning (Wan et al., 2020) †	28.85	25.56	41.12
+ Vanilla scheduled sampling (Bengio et al., 2015) †	28.40	25.43	40.87
+ Target denoising (Meng et al., 2020) †	28.55	25.58	40.57
+ Sampling with sentence oracles (Zhang et al., 2019)	28.65	–	–
+ Sampling with sentence oracles (Zhang et al., 2019) †	28.65	25.50	40.85
+ Confidence-aware scheduled sampling (ours) †	28.80$*$	25.95$**$	41.19$**$
+ Confidence-aware scheduled sampling with target denoising (ours) †	**28.91**$**$	**26.00**$**$	**41.28**$**$
Transformer$_{big}$ (Vaswani et al., 2017)	28.40	–	41.80
Transformer$_{big}$ (Vaswani et al., 2017) †	28.90	25.22	41.89
+ Mixer (Ranzato et al., 2016) †	29.27	25.58	42.37
+ Minimal Risk Training (Shen et al., 2016) †	29.35	25.65	42.46
+ TeaForN (Goodman et al., 2020)	29.30	–	42.73
+ TeaForN (Goodman et al., 2020) †	29.32	25.48	42.62
+ Error correction (Song et al., 2020)	29.20	–	–
+ Self-paced learning (Wan et al., 2020) †	29.68	25.56	42.32
+ Vanilla scheduled sampling (Bengio et al., 2015) †	29.62	25.60	42.55
+ Target denoising (Meng et al., 2020) †	29.18	25.56	42.32
+ Scheduled sampling with sentence oracles (Zhang et al., 2019) †	29.57	25.78	42.65
+ Confidence-aware scheduled sampling (ours) †	29.95$**$	26.00$**$	42.90$**$
+ Confidence-aware scheduled sampling with target denoising (ours) †	**30.09**$**$	**26.27**$**$	**42.97**$**$

Table 3: Translation performance on each WMT dataset. ‘†’ is our implementations under unified settings. The original TeaForN (Goodman et al., 2020) reports SacreBLEU scores. For fair comparison, we re-implement it and report BLEU scores. ‘$*$/$**$’: significantly (Koehn, 2004) better than ‘Vanilla Scheduled Sampling’ with $p < 0.05$ and $p < 0.01$.

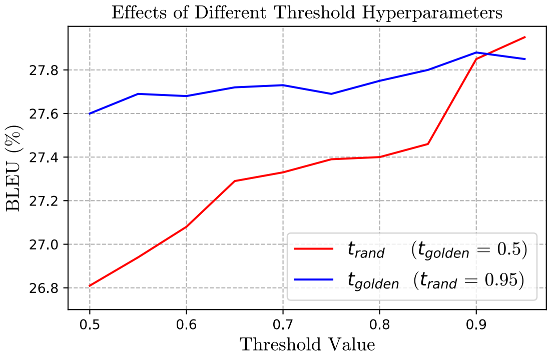

Figure 3: BLUE scores (%) on the validation set of WMT14 EN-DE with different t_{golden} and t_{rand}.

TeaForN. Teacher forcing with n-grams (Goodman et al., 2020) enable the standard teacher forcing with a broader view by n-grams optimization.

Self-paced learning. Wan et al. (2020) assign confidence scores for each input to weight its loss.

Vanilla schedule sampling. Scheduled sampling on training steps with the inverse sigmoid decay (Bengio et al., 2015; Zhang et al., 2019).

Sampling with sentence oracles. Zhang et al. (2019) refine the sampling space of scheduled sampling with sentence oracles, *i.e.,* predictions from beam search. Note that its sampling strategy is still based on training steps with the sigmoid decay.

Target denoising. Meng et al. (2020) add noisy perturbations into decoder inputs when training, which yields a more robust translation model against prediction errors by target denoising.

Confidence-aware scheduled sampling. Our fine-grained schedule strategy described in Equation (6) with $t_{golden} = 0.9$.

Confidence-aware scheduled sampling with target denoising. Our fine-grained schedule strategy described in Equation (7) with $t_{golden} = 0.9$

Schedule Strategy	**BLEU**	**Δ**
Transformer$_{base}$	27.10	ref.
+ Linear decay	27.56$*$	+0.46
+ Exponential decay	27.60$*$	+0.50
+ Inverse sigmoid decay	27.65$*$	+0.55
+ Confidence (ours)	**28.15**$**$	**+1.05**

Table 4: BLUE scores (%) on the validation set of WMT14 EN-DE with different schedule strategies. 'Confidence' refers to the confidence-aware strategy in Equation (6). 'ref.' is short for the reference baseline. '$*$ / $**$': significantly (Koehn, 2004) better than the Transformer$_{base}$ with $p < 0.05$ and $p < 0.01$.

Model	**BLEU**	**Δ**
Our approach	28.15	ref.
– Confidence	27.75	-0.40
– Denoising	28.00	-0.15
– Confidence & Denoising	27.64	-0.51

Table 5: BLUE scores (%) on the validation set of WMT14 EN-DE for ablation experiments. 'Our approach' is 'confidence-aware scheduled sampling with target denoising' in Equation (7). 'Confidence' refers to the confidence-aware strategy in Equation (7). 'Denoising' refers to the target random noise y_{rand} in Equation (7). 'ref.' is short for the reference baseline.

and $t_{rand} = 0.95$.

4.4 Main Results

We list translation qualities in Table 3. For the Transformer$_{base}$ baseline, our 'Confidence-aware scheduled sampling' shows consistent improvements by 0.90, 0.98, 0.89 BLEU points on EN-DE, ZH-EN, and EN-FR, respectively. Moreover, after applying the more fine-grained strategy with target denoising, our 'Confidence-aware scheduled sampling with target denoising' achieves further improvements which are 1.01, 1.03, 0.98 BLEU points on EN-DE, ZH-EN, and EN-FR, respectively. When comparing with the stronger vanilla scheduled sampling method, 'Confidence-aware scheduled sampling with target denoising' still yields improvements by 0.51, 0.57, and 0.41 BLEU points on the above three tasks, respectively. For the more powerful Transformers$_{big}$, we also observe similar experimental conclusions as above. Specifically, 'Confidence-aware scheduled sampling with target denoising' outperforms vanilla scheduled sampling by 0.47, 0.67, and 0.42 BLEU points, respectively. In summary, experiments on strong baselines and various tasks verify the effectiveness and superiority of our approaches.

5 Analysis and Discussion

We analyze our proposals on WMT 2014 EN-DE with the Transformer$_{base}$ model.

5.1 Effects of Confidence-Aware Strategies

In this section, we rigorously validate the effectiveness of confidence-aware strategies by univariate experiments with the only difference at schedule strategy. As shown in Table 4, existing heuristic functions, *i.e.,* linear, exponential, and inverse sigmoid decay, moderately bring improvements over the Transformer$_{base}$ baseline by 0.46, 0.50, and 0.55 BLEU points, respectively. While our confidence-aware strategy that described in Equation (6) can significantly outperform the baseline by 1.05 BLEU points. We attribute the effectiveness of the confidence-aware strategy to its exact and suitable token assignments according to the real-time model competence rather than predefined patterns.

5.2 Ablation Experiments

We conduct ablation experiments to investigate the impacts of various components in our 'Confidence-aware scheduled sampling with target denoising' (described in Equation (7)) and list results in Table 5. Separately removing the confidence-aware strategy degenerates our approach into the vanilla target denoising with a uniform strategy (Meng et al., 2020), which causes a noticeable drop (0.4 BLEU), indicating the confidence-aware strategy plays a leading role for performance. On the other hand, we only observe a drop (0.15 BLEU) when removing 'Target denoising', revealing the additional noise plays a secondary role for performance. Finally, ablating both the confidence-aware strategy and 'Target denoising' degenerates our approach into the vanilla scheduled sampling. It yields a further decrease (0.51 BLEU), suggesting the confidence-aware strategy and 'Target denoising' are complementary with each other.

5.3 Different Numbers of Decoder Layers

As known in existing studies (Domhan, 2018; Wang et al., 2019a), there exists a performance bottleneck at the decoder side of NMT models. Namely, the increase in the number of decoder layers can not bring corresponding improvements for performance. He et al. (2019) attribute this bottle-

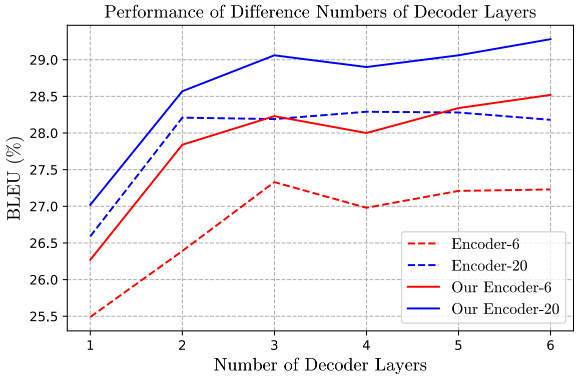

Figure 4: BLUE scores (%) on the validation set of WMT14 EN-DE with different numbers of decoder layers. Solid lines refer to our confidence-aware schedule strategy. Dashed lines refer to the Transformer$_{base}$.

neck to the fact that decoders learn an easier task than encoders.

In this paper, our fine-grained schedule strategy in Equation (7) assigns a more difficult task to the decoder. We can not help wondering whether our strategy is able to alleviate the above performance bottleneck. Firstly, we keep the number of encoders fixed to 6 (*i.e.,* Encoder-6), then apply our confidence-aware schedule strategy on the Encoder-6 Transformer$_{base}$ with the number of decoder layers ranging from 1 to 6. As shown in Figure 4, our approach (solid red line) consistently outperforms the Encoder-6 Transformer$_{base}$ (dashed red line). More importantly, the improvement of Encoder-6 Transformer$_{base}$ stops (*i.e.,* performance bottleneck) once the number of decoder exceeds 4. Despite this, we observe continuous improvement with the growth of decoder layers in our approach. Moreover, we repeat the above experiments with more powerful deep encoders (Encoder-20). We observe that the performance bottleneck for Encoder-20 Transformer$_{base}$ becomes more evident (dashed blue line). Despite this, our approaches (solid blue line) still keep improving performance with the growth of decoder layers on the stronger Encoder-20 Transformer$_{base}$.

In summary, our confidence-aware schedule strategy brings a meaningful increase in the difficulty of decoders, and the bottleneck at the decoder side is alleviated to a certain extend.

5.4 Effects on Different Sequence Lengths

Due to error accumulations, the exposure bias problem becomes more problematic with the growth of sequence lengths (Zhou et al., 2019; Zhang et al., 2020). Thus it is intuitive to verify the effectiveness of our approach over different sequence lengths. Considering the validation set of WMT14 EN-DE (3k) is too small to cover scenarios with various sentence lengths, we randomly select 10k training data with lengths from 10 to 100. As shown in Figure 5, our approach consistently outperform the Transformer$_{base}$ model at different sequence lengths. Moreover, the improvements of our approach over the Transformer$_{base}$ is gradually increasing with sentence lengths. Specifically, we observe more than 1.0 BLEU improvements when sentence lengths in $[80, 100]$.

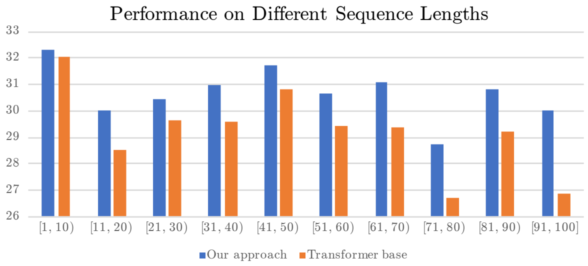

Figure 5: BLUE scores (%) on the randomly sampled WMT14 EN-DE training data with different lengths.

5.5 Model Convergence

As aforementioned, our confidence-aware scheduled sampling learns to deal with the exposure bias problem in an efficient manner, thus speeding up the model convergence. As shown in Figure 6, it costs the Transformer$_{base}$ 245k steps to converge to a local optimum (about 27.1 BLEU). To achieve the same performance, it only costs our confidence-aware scheduled sampling 80k step, namely about 3.0× speed up over the Transformer$_{base}$ and 1.8× speed up over the vanilla scheduled sampling. Since vanilla scheduled sampling randomly exposes more difficult predicted tokens for each token position, regardless of the actual model competence, its convergence speed is restricted to a certain extent. On the contrary, our approach samples predicted tokens only if the current model is capable of dealing with these more difficult inputs, mimicking the learning process of humans. Therefore, our approach is trained more efficiently.

6 Related Work

Confidence-aware Learning for NMT. As to confidence estimations for NMT, Zoph et al. (2015) frame translation as a compression game and measure the amount of information added by transla-

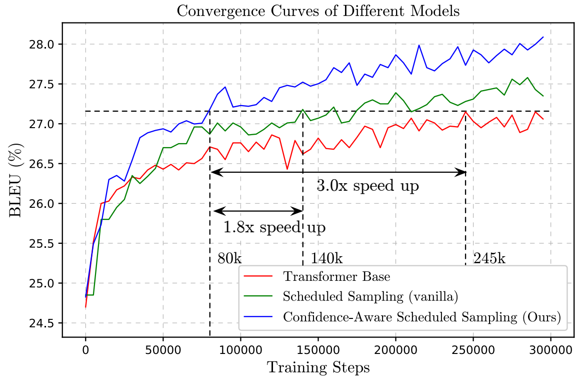

Figure 6: Convergence curves for different models. BLUE scores (%) are calculated on the validation set of WMT14 EN-DE. Our approach can achieve the same performance as the Transformer$_{base}$ with about 3.0× speed up.

tors. Wang et al. (2019b) propose to quantify the confidence of NMT model predictions based on model uncertainty, which is widely extend to select training samples (Jiao et al., 2020; Dou et al., 2020), to design confidence-aware curriculum learning (Zhou et al., 2020; Wan et al., 2020), and to augment synthetic corpora (Wei et al., 2020). Model confidence is also served as a useful metric for analyze NMT model from the perspective of fitting and search (Ott et al., 2018), visualization (Rikters et al., 2017) and calibration (Kumar and Sarawagi, 2019; Wang et al., 2020). Different from existing studies, we are the first to propose confidence-aware scheduled sampling for alleviating the exposure bias problem in NMT.

7 Conclusion

In this paper, we propose confidence-aware scheduled sampling for NMT, which exactly samples corresponding tokens according to the real-time model competence rather than human intuitions. We further explore to sample more noisy tokens for high-confidence token positions, preventing scheduled sampling from degenerating into the original teacher forcing mode. Experiments on three large-scale WMT translation tasks suggest that our approach improves vanilla scheduled sampling both translation quality and convergence speed. We elaborately analyze the effectiveness and efficiency of our approach from multiple aspects. As a result, we further observe our approaches: 1) can alleviate the performance bottleneck of decoders for NMT to a certain extend; 2) improve the translation quality of long sequences.

Acknowledgments

The research work descried in this paper has been supported by the National Key R&D Program of China (2020AAA0108001) and the National Nature Science Foundation of China (No. 61976015, 61976016, 61876198 and 61370130). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Advances in neural information processing systems*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1171–1179. Curran Associates, Inc.

Wray L. Buntine and A. Weigend. 1991. Bayesian back-propagation. *Complex Syst.*, 5.

Tobias Domhan. 2018. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808.

Zi-Yi Dou, Antonios Anastasopoulos, and Graham Neubig. 2020. Dynamic data selection and weighting for iterative back-translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5894–5904.

Daniel Duckworth, Arvind Neelakantan, Ben Goodrich, Lukasz Kaiser, and Samy Bengio. 2019. Parallel scheduled sampling. *arXiv preprint arXiv:1906.04331*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Sebastian Goodman, Nan Ding, and Radu Soricut. 2020. TeaForN: Teacher-forcing with n-grams. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8704–8717, Online. Association for Computational Linguistics.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 366–371.

Tianyu He, Xu Tan, and Tao Qin. 2019. Hard but robust, easy but sensitive: How encoder and decoder perform in neural machine translation. *arXiv preprint arXiv:1908.06259*.

Wenxiang Jiao, Xing Wang, Shilin He, Irwin King, Michael Lyu, and Zhaopeng Tu. 2020. Data rejuvenation: Exploiting inactive training examples for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2255–2266.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. 2018. Action-dependent control variates for policy optimization via stein identity. In *International Conference on Learning Representations*.

Fandong Meng, Jianhao Yan, Yijin Liu, Yuan Gao, Xianfeng Zeng, Qinsong Zeng, Peng Li, Ming Chen, Jie Zhou, Sifan Liu, and Hao Zhou. 2020. Wechat neural machine translation systems for wmt20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 238–246, Online. Association for Computational Linguistics.

Tsvetomila Mihaylova and André F. T. Martins. 2019. Scheduled sampling for transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356, Florence, Italy. Association for Computational Linguistics.

R. Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When does label smoothing help? In *NeurIPS*.

Radford M Neal. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *ICML*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Matīss Rikters, Mark Fishel, and Ondřej Bojar. 2017. Visualizing neural machine translation attention and confidence. *The Prague Bulletin of Mathematical Linguistics*, 109(1):39–50.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Kaitao Song, Xu Tan, and Jianfeng Lu. 2020. Neural machine translation with error correction. In *IJCAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Richard Stuart Sutton. 1984. Temporal credit assignment in reinforcement learning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yu Wan, Baosong Yang, Derek F. Wong, Yikai Zhou, Lidia S. Chao, Haibo Zhang, and Boxing Chen. 2020. Self-paced learning for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019a. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019b. Improving back-translation with uncertainty-based confidence estimation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 791–802, Hong Kong, China. Association for Computational Linguistics.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079, Online. Association for Computational Linguistics.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909.

Xiangpeng Wei, Heng Yu, Yue Hu, Rongxiang Weng, Luxi Xing, and Weihua Luo. 2020. Uncertainty-aware semantic augmentation for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2724–2735.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huanbo Luan, and Yang Liu. 2017. Thumt: An open source toolkit for neural machine translation. *arXiv preprint arXiv:1706.06415*.

Jiajun Zhang, Long Zhou, Yang Zhao, and Chengqing Zong. 2020. Synchronous bidirectional inference for neural sequence generation. *Artificial Intelligence*, 281:103234.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, Florence, Italy. Association for Computational Linguistics.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.

Yikai Zhou, Baosong Yang, Derek F Wong, Yu Wan, and Lidia S Chao. 2020. Uncertainty-aware curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6934–6944.

Barret Zoph, Marjan Ghazvininejad, and Kevin Knight. 2015. How much information does a human translator add to the original? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 889–898, Lisbon, Portugal. Association for Computational Linguistics.

MA-BERT: Learning Representation by Incorporating Multi-Attribute Knowledge in Transformers

You Zhang[†], **Jin Wang**[†, 1], **Liang-Chih Yu**[‡, 2] and **Xuejie Zhang**[†, 3]
[†]School of Information Science and Engineering, Yunnan University, Yunnan, P.R. China
[‡]Department of Information Management, Yuan Ze University, Taiwan
`Contact:{wangjin`[1]`, xjzhang`[3]`}@ynu.edu.cn, lcyu@saturn.yzu.edu.tw`[2]

Abstract

Incorporating attribute information such as user and product features into deep neural networks has been shown to be useful in sentiment analysis. Previous works typically accomplished this in two ways: concatenating multiple attributes to word/text representation or treating them as a bias to adjust attention distribution. To leverage the advantages of both methods, this paper proposes a multi-attribute BERT (MA-BERT) to incorporate external attribute knowledge. The proposed method has two advantages. First, it applies multi-attribute transformer (MA-Transformer) encoders to incorporate multiple attributes into both input representation and attention distribution. Second, the MA-Transformer is implemented as a universal layer and stacked on a BERT-based model such that it can be initialized from a pre-trained checkpoint and fine-tuned for the downstream applications without extra pretraining costs. Experiments on three benchmark datasets show that the proposed method outperformed pre-trained BERT models and other methods incorporating external attribute knowledge.

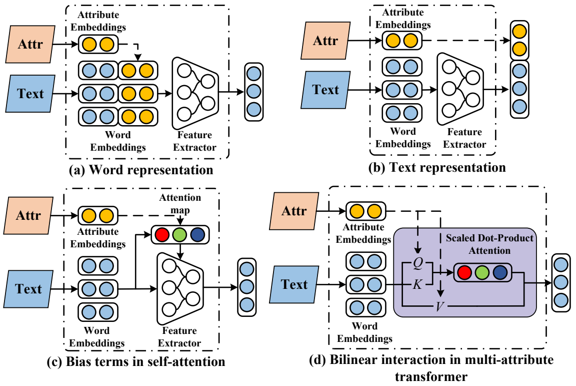

(a) Word representation

(b) Text representation

(c) Bias terms in self-attention

(d) Bilinear interaction in multi-attribute transformer

Figure 1: Different strategies to incorporate external attribute knowledge into deep neural networks.

1 Introduction

To learn a distributed text representation for sentiment classification (Pang and Lee, 2008; Liu, 2012), conventional deep neural networks, such as convolutional neural networks (CNN) (Kim, 2014) and long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), and common integration technics, such as self-attention mechanisms (Vaswani et al., 2017; Chaudhari et al., 2019) and dynamic routing algorithms (Gong et al., 2018; Sabour et al., 2017), are usually applied to compose the vectors of constituent words. To further enhance the performance, pre-trained models (PTMs), such as BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019), and XLM-RoBERTa (Conneau et al., 2019) can be fine-tuned and transferred for sentiment analysis tasks. Practically, PTMs were first fed a large amount of unannotated data, and trained using a masked language model or next sentence prediction to learn the usage of various words and how the language is written in general. Then, the models are transferred to another task to be fed another smaller task-specific dataset.

The abovementioned methods only use features from plain texts. Incorporating attribute information such as users and products can improve sentiment analysis task performance. Previous works typically incorporated such external knowledge by concatenating these attributes into word and text representations (Tang et al., 2015), as shown in Figs. 1(a) and (b). Such methods are often introduced in shallow models to attach attribute information to modify the representation of either words or texts. However, this may lack interaction between attributes and the text since it equally aligns words to attribute features, thus the model is unable to emphasize important tokens. Several works have used attribute features as a bias term in self-attention mechanisms to model meaningful rela-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2338–2343
August 1–6, 2021.

tions between words and attributes (Wu et al., 2018; Chen et al., 2016b; Dong et al., 2017; Dou, 2017), as shown in Fig. 1(c). By using the $softmax$ function for normalization to calculate the attention score, the incorporated attribute features only impact the allocation of the attention weights. As a result, the representation of input words has not been updated, and the information of these attributes will be lost. For example, depending on individual preferences for $chili$, readers may focus on reviews talking about $spicy$, but only those who like $chili$ would consider such review recommendations useful. However, current self-attention models that learn text representations by adjusting the weights of $spicy$ may still produce the same word representation of $spicy$ for different persons, leading to confusion in distinguishing people who like $chili$ or not.

To address the above problems, this study proposes a multi-attribute BERT (MA-BERT) model which applies multi-attribute transformer (MA-Transformer) encoders to incorporate external attribute knowledge. Different from being incorporated into the attention mechanism as bias terms, multiple attributes can be injected into both attention maps and input token representations using bilinear interaction, as shown in Fig. 1(d). In addition, the MA-Transformer is implemented as a universal layer and stacked on a BERT-based model such that it can be initialized from a pre-training checkpoint and fine-tuned for downstream tasks without extra pre-training costs. Experiments are conducted on three benchmark datasets (IMDB, Yelp-2013, and Yelp-2014) for sentiment polarity classification. The results show that the proposed MA-BERT model outperformed pre-trained BERT models and other methods incorporating external attribute knowledge.

The remainder of this paper is organized as follows. Section 2 provides a detailed description of the proposed methods. The empirical experiments are reported with analysis in Section 3. Conclusions are finally drawn in Section 4.

2 Multi-Attribute BERT Model

Fig. 2 shows an overview of the MA-BERT model. It mainly consists of two parts, including a BERT-based PTM model and several MA-Transformer encoders as extra layers stacked on BERT. Both components are described in detail below.

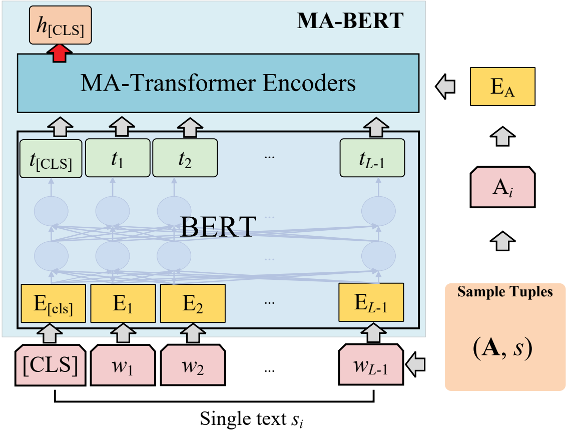

Figure 2: Overall architecture of the MA-BERT model.

2.1 BERT Encoder

By applying a word piece tokenizer (Wu et al., 2016), the input text can be denoted as a sequence of tokens, i.e., $s = \{w_0, w_1, w_2, \ldots, w_{L-1}\}$, where L is the length of the text and $w_0 =$ [CLS] is a special classification token. Moreover, its corresponding attributes are denoted as $\mathbf{A} = \{a_1, a_2, \ldots, a_M\}$, where M is the number of attributes in the text. Thus, the i-th input sample can be denoted as a tuple, i.e., $(\mathbf{A}_i, s_i)$.

To learn the hidden representation, the pre-trained language model BERT (Devlin et al., 2019) was used, achieving impressive performance for various natural language processing (NLP) tasks. We then fed the token sequence into the BERT model to obtain the representation, denoted as,

$$T = [t_0, ..., t_{L-1}] = f_{\text{BERT}}([w_0, ..., w_{L-1}]; \theta_{\text{BERT}}) \quad (1)$$

where $T \in \mathbb{R}^{L \times d_t}$ is the output representation of all tokens; θ_{BERT} is the trainable parameters of BERT, which is initialized from a pretrained checkpoint and then fine-tuned during the model training; d_t=768 is the dimensionality of the output representation.

According to Wu et al. (2018) and Wang et al. (2017), all the attributes are mapped to attribute embeddings $E_{\text{A}} = [E_{\text{A},1}, E_{\text{A},2}, \ldots, E_{\text{A},M}] \in \mathbb{R}^{M \times d_E}$, which are randomly initialized and updated in the following training phase.

Multi-Attribute Attention. To incorporate multiple attributes into the MA-Transformer, we introduce multi-attribute attention (MAA), which is expressed as,

$$Y = \text{MAA}(T, E_{\text{A}}) = [U_1, \ldots, U_M]\, W_o \quad (2)$$

$$U_m = \text{Att}(T, E_{\text{A},m}) = softmax\left(\frac{Q_m {K_m}^\top}{\sqrt{d}}\right) V_m \quad (3)$$

where U_m is the attention from m-th attribute; $W_o \in R^{(M \cdot d) \times d_t}$ is the output linear projection and d denotes the dimensionality of Q, K and V; Q, K and V are matrices that package the queries, keys and values, which are defined as,

$$Q_m = T \cdot W_{q,m} \odot E_{\mathrm{A},m} \tag{4}$$

$$K_m = T \cdot W_{k,m} \odot E_{\mathrm{A},m} \tag{5}$$

$$V_m = T \cdot W_{v,m} \odot E_{\mathrm{A},m} \tag{6}$$

where Q_m, K_m and $V_m \in \mathbb{R}^{L \times d_E}$ are bilinear transformations (Huang et al., 2019) applied on the input representation T and attribute representation $E_{\mathrm{A,m}}$. $W_{q,m}$, $W_{k,m}$ and $W_{v,m} \in \mathbb{R}^{d_t \times d_E}$ are weight matrices for query, key and value projections, and $\cdot$ and $\odot$ respectively denote the inner and the Hadamard product.

Similar to Vaswani et al. (2017), we also introduced multi-head mechanism for MA-Transformer, denoted as,

$$U_m = \bigoplus_{k=1}^{K} \mathrm{Att}(T, E^k_{\mathrm{A},m}) \in \mathbb{R}^{L \times (K \cdot d_E)} \tag{7}$$

where K is the number of heads for each attribute and $\oplus$ denotes the concatenation operator; $E^k_{\mathrm{A},m} \in \mathbb{R}^{d_E}$ is the m-th attribute representation in the k-th head, and its dimensionality should be ensured that $d_E = d_t/K$. Given that different heads can capture different relation types along with text representations, different parameters are considered for different heads.

2.2 MA-Transformer

Taking the representation of both text T and attribute $\mathbf{A}$ as input, an MA-Transformer encoder then processes the same as a standard transformer encoder (Vaswani et al., 2017) to generate $Y \in \mathbb{R}^{L \times d_t}$. Then, Y is connected by a normalization layer and a residual layer from the input representation T. The intermediate output is then passed to a two-layered feed-forward network with a rectified linear unit (ReLU) activate function. Similarly, residual and normalization layers are connected to generate the final output which is taken as the input for the next encoder.

By stacking several MA-Transformer encoders on the BERT model, the MA-BERT model generates a review representation $h_{[\mathrm{CLS}]}$ consistent with the special token [CLS]. Then, a classifier comprised of a linear projection and a $softmax$ activation (with the dimension identical to the number of classes) is used for classification.

3 Comparative Experiments

Datasets. Following the experimental settings used in Tang et al. (2015), the proposed MA-BERT model is evaluated using three benchmark datasets [1] (IMDB, Yelp-2013, and Yelp-2014). The evaluation metrics include accuracy (Acc.) and root mean squared error ($RMSE$). Higher Acc. and lower $RMSE$ scores indicate higher performance.

Implementation Details. The baseline methods can be divided into three groups. The first group includes the methods without user and product information such as **CNN** (Kim, 2014), **BiLSTM** (Hochreiter and Schmidhuber, 1997), neural sentiment classification (**NSC**) (Chen et al., 2016a) and its variant with a local attention mechanism (**NSC+LA**). For the BERT-based methods, the uncased-base-**BERT** model consisting of 12 layers of transformer encoders was implemented for comparison. **ToBERT** (Pappagari et al., 2019) was trained non-end2end using a word-to-segment strategy in a two-stage way.

The second group includes existing methods incorporating user and product information such as NSC with user (U) and product (P) information incorporated into an attention (A) mechanism (**NSC+UPA**), user product neural network (**UPNN**) (Tang et al., 2015), hierarchical model with separate user attention and product attention (**HUAPA**) (Wu et al., 2018), and the chunk-wise importance matrix model (**CHIM**) (Amplayo, 2019).

The third group includes a set of BERT-based methods incorporating user and product information using different strategies, presented in Figs. 1(a)-(c). In detail, an uncased-base-BERT model first extracted fixed feature vectors from texts. Then, the **BERT Concat (word)** model incorporates attribute features into each word vector and stacks another 6 transformer encoders as the feature extractor. Similarly, the **BERT Concat (text)** incorporates attribute features into the representation of the special token [CLS] for the classification. Finally, the **BERT Attention (bias)** applied 6 more MA-Transformers which only inject attributes into Q and K to calculate attention score instead of V in Eq. (6).

The proposed **MA-BERT** models applied 6 MA-Transformer encoders to incorporate user and product attributes, and stacking over the BERT model.

---

[1] `http://ir.hit.edu.cn/~dytang/paper/acl2015/dataset.7z`

Models	IMDB		Yelp-2013		Yelp-2014	
	Acc. (%)	$RMSE$	Acc. (%)	$RMSE$	Acc. (%)	$RMSE$
without user and product information						
CNN (UPNN w/o UP)	40.5	1.629	57.7	0.812	58.5	0.808
BiLSTM	43.3	1.494	58.4	0.764	59.2	0.733
NSC	44.3	1.465	62.7	0.701	63.7	0.686
NSC+LA	48.7	1.381	63.1	0.706	63.0	0.715
BERT	51.8	1.191	67.7	0.627	67.2	0.630
ToBERT	50.8	1.194	66.7	0.626	66.9	0.620
with user and product information						
UPNN	43.5	1.602	59.6	0.803	60.8	0.764
NSC+UPA	53.3	1.281	65.0	0.692	66.7	0.654
HUAPA	55.0	1.185	68.3	0.628	68.6	0.626
CHIM$_{embedding}$	56.4	1.161	67.8	0.646	69.2	0.629
BERT Concat (word)	56.8	1.106	69.9	0.602	70.9	0.582
BERT Concat (text)	54.6	1.168	68.5	0.616	71.0	0.590
BERT Attention (bias)	52.5	1.177	68.0	0.635	67.6	0.617
MA-BERT	**57.3**	**1.042**	**70.3**	**0.588**	**71.4**	**0.573**

Table 1: Comparative results of different methods for sentiment classification. The **boldface** figures indicate the best results among all methods and underscored figures represent the best performance for each group of methods. All results are averaged over five runs.

Each attribute is initialized in a uniform distribution $U \sim (-0.25, 0.25)$ with the dimension of 768 (d_t) and head number of 12 (K). Thus, the dimension of each head (d_E) is set to 64. All other hyper-parameters in MA-Transformer encoders are identical with BERT-transformer encoders due to their isomorphic structure. For all models, the AdamW (Loshchilov and Hutter, 2017) optimizer was used with a base learning rate of 2e-5 in a warmup linear schedule. Early stopping (Prechelt, 1998) strategy with a patience of 3 epochs was also applied to avoid overfitting. The code for this paper is available at: `https://github.com/yoyo-yun/MA-Bert`.

Comparative Results and Discussion. Table 1 shows the comparative results of different methods for sentiment ordinal classification. For models without user and product attributes, BiLSTM outperforms CNN (UPNN w/o UP), due to its ability to encode text. Furthermore, both NSC and NSC+LA outperformed BiLSTM mainly because of its hierarchical structure.

Incorporating the user and product attributes improved performance. For example, UPNN achieved a better result than its variant without user and product attributes, i.e., CNN (UPNN w/o UP). In addition, both NSC+UPA and HUAPA introduced the user and product information as a bias to guide the hierarchical attention, and thus outperformed NSC and NSC+LA.

The proposed MA-BERT achieved the best performance on all datasets. Compared with baselines without user and product attributes, the MA-BERT can leverage implicit attribute features to boost performance. MA-BERT outperformed methods already incorporating user and product attributes (i.e., NSC+UPA, HUAPA and CHIM$_{embedding}$) because the proposed model can incorporate attribute knowledge to both the attention map and input representation.

The BERT and ToBERT models achieved improvement on all datasets against the conventional models, due to the large knowledge migration from pre-training. Unfortunately, a lack of implicit extra features resulted in performance lower than that of the proposed MA-BERT model. MA-BERT also outperformed BERT Concat (word), BERT Concat (text) and BERT Attention (bias), indicating that the proposed MA-Transformer architecture can improve existing incorporation strategies. Furthermore, the proposed MA-BERT could be initialized from the pre-trained checkpoint of BERT, thus making full use of the parameter settings without bringing additional costs for pre-training.

4 Conclusion

This paper proposes a MA-BERT model capable of incorporating multiple attributes into BERT-based PTMs for learning attribute-specific text representation. Different from existing attention models, the MA-Transformer can inject external knowledge to both attention maps and the input representation.Additionally, the proposed model could be extended to other tasks by using the MA-Transformer encoder as a universal layer and stacking it on a BERT-based model. Future work will attempt to incorporate such or similar multiple attributes into PTMs in the pre-training phases.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61966038 and 61762091, and by the Ministry of Science and Technology, Taiwan, ROC, under Grant No. MOST107-2628-E-155-002-MY3. The authors would like to thank the anonymous reviewers for their constructive comments.

References

Reinald Kim Amplayo. 2019. Rethinking Attribute Representation and Injection for Sentiment Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5601–5612, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. 2019. An Attentive Survey of Attention Models. *arXiv preprint arXiv:1904.02874*.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016a. Neural Sentiment Classification with User and Product Attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, pages 1650–1659.

Tao Chen, Ruifeng Xu, Yulan He, Yunqing Xia, and Xuan Wang. 2016b. Learning User and Product Distributed Representations Using a Sequence Model for Sentiment Analysis. *IEEE Computational Intelligence Magazine*, 11(3):34–44.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised Cross-lingual Representation Learning at Scale. *arXiv preprint arXiv:1911.02116*.

J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 4171–4186.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2017)*, volume 1, pages 623–632.

Zi-Yi Dou. 2017. Capturing User and Product Information for Document Level Sentiment Analysis with Deep Memory Network. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP-2017)*, pages 521–526.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information Aggregation via Dynamic Routing for Sequence Encoding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2742–2752.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. Fibinet: Combining feature importance and bilinear feature interaction for click-through rate prediction. In *13th ACM Conference on Recommender Systems (RecSys-2019)*, pages 169–177.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 1746–1751, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis: Foundations and Trends in Information Retrieval. *Foundations and Trends® in Information Retrieval*, 2(1-2):1–135.

Raghavendra Pappagari, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical Transformers for Long Document Classification. *arXiv preprint arXiv:1910.10781*.

Lutz Prechelt. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic Routing Between Capsules. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS-2017)*, pages 3859–3869.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning Semantic Representations of Users and Products for Document Level Sentiment Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-2015)*, pages 1014–1023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in neural information processing systems(nips-2017)*, pages 5598–6008.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *the Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-2017)*, pages 3316–3322.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Zhen Wu, Xin Yu Dai, Cunyan Yin, Shujian Huang, and Jiajun Chen. 2018. Improving review representations with user attention and product attention for sentiment classification. In *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5989–5996, New Orleans, Louisiana, USA.

A Closer Look into the Robustness of Neural Dependency Parsers Using Better Adversarial Examples

Yuxuan Wang[*1] **Wanxiang Che**[1] **Ivan Titov**[2,3] **Shay B. Cohen**[2] **Zhilin Lei**[1] **Ting Liu**[1]
[1]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology
[2]ILCC, School of Informatics, University of Edinburgh
[3]ILLC, University of Amsterdam
`{yxwang, car, zllei, tliu}@ir.hit.edu.cn`
`{ititov, scohen}@inf.ed.ac.uk`

Abstract

Previous work on adversarial attacks on dependency parsers has mostly focused on attack methods, as opposed to the quality of adversarial examples, which in previous work has been relatively low. To address this gap, we propose a method to generate high-quality adversarial examples with a higher number of candidate generators and stricter filters, and then verify their quality using automatic and human evaluations. We perform analysis with different parsing models and observe that: (i) injecting words not used in the training stage is an effective attack strategy; (ii) adversarial examples generated against a parser strongly depend on the parser model, the token embeddings, and even the specific instantiation of the model (i.e., a random seed). We use these insights to improve the robustness of English parsing models, relying on adversarial training and model ensembling.[1]

1 Introduction

Neural network-based models have achieved great successes in a wide range of NLP tasks. However, recent work has shown that their performance can be easily undermined with adversarial examples that would pose no confusion for humans (Zhang et al., 2020). As an increasing number of successful adversarial attackers have been developed for NLP tasks, the quality of the adversarial examples they generate has been questioned (Morris et al., 2020).

The definition of a valid successful adversarial example differs across target tasks. In semantic tasks such as sentiment analysis (Zhang et al., 2019) and textual entailment (Jin et al., 2020), a valid successful adversarial example needs to be able to alter the prediction of the target model while preserving the semantic content and fluency of the original text. In contrast, in the less explored field of attacking syntactic tasks, the syntactic structure, rather than the semantic content, must be preserved while also maintaining the fluency. Preserving the syntactic structure enables us to use the gold syntactic structure of the original sentence in the evaluation process. While preserving the fluency ensures that ungrammatical adversarial examples, which not only fool the target model but also confuse humans, will not be considered valid. Therefore in this paper, we evaluate the quality of an adversarial example in two aspects, namely the **fluency** and **syntactic structure preservation**.

Recently, Zheng et al. (2020) proposed the first dependency parser attacking algorithm based on word-substitution which depended entirely on BERT (Devlin et al., 2019) to generate candidate substitutes. The rational was that the use of the pre-trained language model will ensure fluency of the adversarial examples. However, we find that using BERT alone is far from enough to preserve fluency.

Therefore, in this paper, we propose a method to generate better adversarial examples for dependency parsing with four types of candidate generators and filters. Specifically, our method consists of three steps: (i) determining the substitution order, (ii) generating and filtering candidate substitutes for each word, (iii) searching for the best possible combination of substitutions, based on pre-computed candidates and the substitution order. We verify the superiority of the proposed method in terms of syntactic structure preservation and fluency using both automatic and human evaluations, and further show the limitation of the previous BERT-based method.

Table 1 shows adversarial examples generated by our and the method of Zheng et al. (2020), demonstrating that examples generated by our method are

---

[*] Work partially done while at the University of Edinburgh.

[1] Our code is available at: `https://github.com/WangYuxuan93/DepAttacker.git`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2344–2354
August 1–6, 2021.

Model	Ours	Zheng et al.
example-1	Most of those ~~**freed**~~ **emancipated** had spent at least 25 years in prison.	Most of those freed ~~**had**~~ **were** spent at least 25 years in prison.
example-2	Boeing received a $ 46 million Air Force contract for ~~**developing**~~ **devising** cable systems for the Minuteman Missile.	Boeing received a $ 46 million Air ~~**Force**~~ **America** contract for ~~**developing**~~ **securing** cable systems for the Minuteman Missile.
example-3	He used better than 5,000 words heaping scorn on the ~~**witnesses**~~ **eyewitnesses** for exercising the Fifth.	He used better than 5,000 ~~**words**~~ **times** heaping scorn on the ~~**witnesses**~~ **dollars** for exercising the ~~**Fifth**~~ **grand**.

Table 1: Adversarial examples generated by our and Zheng et al. (2020)'s methods. The original words are highlighted in bold blue font while the substitute words are highlighted in bold red ones.

more fluent, producing complicated substitutes like **emancipated**, **devising** and **eyewitnesses**. Additionally, since it is nontrivial to decode valid multi-subword tokens from BERT, the BERT-based method of Zheng et al. (2020) only generates single subwords as substitutes.

With the proposed attacking method, we evaluate the robustness of different parsing models and analyse the properties of adversarial attacks. We find that (i) the introduction of out-of-vocabulary (OOV, words not in the embedding's vocabulary) and out-of-training (OOT, words not in the training set of the parser) words in adversarial examples are two main factors that harm models' performance; (ii) adversarial examples generated against a parser strongly depend on the type of the parser, the token embeddings and even the random seed.

Adversarial training (Goodfellow et al., 2015), where adversarial examples are added in the training stage, has been commonly used in previous work (Zheng et al., 2020; Han et al., 2020) to improve a parser's robustness. Only a limited number of adversarial examples have been used in such cases, and Zheng et al. (2020) argued that overuse of them may lead to a performance drop on the clean data. However, we show that with improvement in the quality of adversarial examples produced in our method, more adversarial examples can be used in the training stage to further improve the parsing models' robustness without producing any apparent harm in their performance on the clean data. Inspired by our second finding, we propose to improve the parsers' robustness by combining models trained with different random seeds and embeddings. Such methods, which are not targeting specific types of attacks, should improve the capacity to defend against new attacks as compared to standard adversarial training.

2 Method

In this section, we first give a formal definition of a dependency parsing attack. Then we describe the proposed attacking method for dependency parsing, shown in Algorithm 1. It consists of three steps, namely ranking word importance (lines 1-4), generating candidates for substitution (line 7) and searching for the best substitute combination (lines 8-21).

2.1 Problem Definition

Given an input text space $\mathcal{X}$ containing all possible input sentences $\boldsymbol{x}$ and an output space $\mathcal{Y}$ containing all possible dependency trees of $\boldsymbol{x}$, a parser $F : \mathcal{X} \rightarrow \mathcal{Y}$ learns to map the sentence $\boldsymbol{x}$ to its corresponding tree $\boldsymbol{y}$, denoted by $F(\boldsymbol{x}) = \boldsymbol{y}$. The i-th word of $\boldsymbol{x}$ is denoted by x_i. For sentence $\boldsymbol{x}$, a valid adversarial example $\boldsymbol{x}^*$ is crafted by adding a perturbation to $\boldsymbol{x}$ so that

$$F(\boldsymbol{x}^*) \neq \boldsymbol{y}, \sigma(\boldsymbol{x}^*, \boldsymbol{x}) \leq \epsilon,$$

where σ is a constraint function and ϵ ensures that i) the perturbation is imperceptible, ii) the true dependency tree of $\boldsymbol{x}^*$ should be the same as that of $\boldsymbol{x}$. In this paper, these two constraints are ensured through the use of various filters (see Section 2.3) and are used to evaluate the quality of adversarial examples (see details on fluency and syntactic structure preservation in Section 3.3).

2.2 Word Importance Ranking

Word importance ranking in our model is based on the observation that some words have a stronger influence on model prediction than others. Such word importance is typically computed by setting each word to unknown and examining the changes in their predictions (Li et al., 2016; Ren et al., 2019).

Algorithm 1 Dependency Parsing Attack

```
Input: Sentence example $\boldsymbol{x}^{(0)} = \{x_1, x_2, \dots, x_N\}$, maximum percentage of words allowed to be modified $\gamma$
Output: Adversarial example $\boldsymbol{x}^{(i)}$
for $i = 1$ to $N$ do
    Compute word importance $I(\boldsymbol{x}^{(0)}, x_i)$ via Eq. 1
end for
Create a set $W$ of all words $x_i \in \boldsymbol{x}^{(0)}$ sorted by the descending order of their importance $I(\boldsymbol{x}^{(0)}, x_i)$.
$t = 0$
for each word $x_j$ in $W$ do
    Build candidate set $\mathcal{C}_j$ for $x_j$ following the Candidate Substitute Generating step
    Initialise valid candidate set $\mathcal{VC} \leftarrow \{\}$
    for each candidate $c_k$ in $\mathcal{C}_j$ do
        Compute the accuracy change $S(\boldsymbol{x}^{(t)}, c_k, j)$ via Eq. 3
        if $S(\boldsymbol{x}^{(t)}, c_k, j) \leq 0$ then continue end if
        Add $c_k$ to the set $\mathcal{VC}$
    end for
    if $\mathcal{VC}$ is not empty then
        $c^* = \operatorname{argmax}_{c \in \mathcal{VC}} S(\boldsymbol{x}^{(t)}, c, j)$
        $t = t + 1$
        $\boldsymbol{x}^{(t)} \leftarrow$ Replace $x_j$ in $\boldsymbol{x}^{(t-1)}$ with $c^*$
        if $t \geq \gamma \cdot N$ then return $\boldsymbol{x}^{(t)}$ end if
    end if
end for
if $t > 0$ then return $\boldsymbol{x}^{(t)}$ else return None end if
```

This helps to determine the word substituting order in the proposed method.

In this work, we use a combination of the changes found in the unlabelled attachment score (UAS) and in the labelled attachment score (LAS) to measure word importance. Specifically, the importance of a word x_i in sentence $\boldsymbol{x}$ is computed as

$$\begin{aligned} I(\boldsymbol{x}, x_i) = & \lambda_{arc} \Delta_{\text{UAS}}(\boldsymbol{x}, \hat{\boldsymbol{x}}_i) + \\ & (1 - \lambda_{arc}) \Delta_{\text{LAS}}(\boldsymbol{x}, \hat{\boldsymbol{x}}_i), \end{aligned} \quad (1)$$

where $\boldsymbol{x} = x_1 x_2 \dots x_i \dots x_N$ is the original sentence and $\hat{\boldsymbol{x}}_i = x_1 x_2 \dots \text{UNK} \dots x_N$ replaces x_i with an 'unknown' token. Here $\Delta_{\text{UAS}}(\boldsymbol{x}, \hat{\boldsymbol{x}}_i) = \text{UAS}_{F(\boldsymbol{x})} - \text{UAS}_{F(\hat{\boldsymbol{x}}_i)}$ and $\Delta_{\text{LAS}}(\boldsymbol{x}, \hat{\boldsymbol{x}}_i) = \text{LAS}_{F(\boldsymbol{x})} - \text{LAS}_{F(\hat{\boldsymbol{x}}_i)}$ are the changes in UAS and LAS respectively. λ_{arc} is a coefficient that controls the relative importance of dependency arcs and their labels.

2.3 Generation of Substitute Candidates

Generating substitute candidates is a critical step, as it significantly influences the attack success rate and the quality of generated adversarial examples. Zheng et al. (2020) relied entirely on BERT to generate candidates, but this limits the quality of the adversarial examples. To alleviate this problem, we first collect candidate substitutes from four generation methods, then apply filters to discard inappropriate substitutes, ensuring both diversity and quality of the generated candidates.

2.3.1 Generating Process

We collect substitutes from the following methods:

BERT-Based Method: We use BERT to generate candidates for each target word from its context. This method generates only single subwords.

Embedding-Based Method: Following Alzantot et al. (2018), we use word embeddings of Mrkšić et al. (2016)[2] to compute the N nearest neighbours of each target word according to their cosine similarity and use them as candidates.

Sememe-Based Method: The sememes of a word represent its core meaning (Dong and Dong, 2006). Following Zang et al. (2020), we collect the substitutes of the target word x based on the rule that one of the substitutes the senses of x^* must have the same sememe annotations as one of senses of x.

Synonym-Based Method: We use WordNet[3] to extract synonyms of each target word as candidates.

2.3.2 Filtering Process

We apply the following four types of filters to discard candidates which are likely inappropriate, either in terms of syntactic preservation or fluency.

POS Filter: We first filter out substitutes with different part-of-speech (POS) tags from the original word.[4] This filter is essential for preserving the syntactic structure of the sentence.

Word Embedding Similarity Filter: We use the word embeddings of Mrkšić et al. (2016) to compute the cosine similarity between the original word and each of the substitutes in $\mathcal{C}$ and filter out those whose similarities are less than a threshold ϵ_w.[5]

Grammar Checker Filter: We employ an off-the-shelf grammar checker[6] to filter out candidates that may introduce grammar errors. This filter helps to further ensure that the syntactic structure and fluency are preserved.

Perplexity Filter: We employ GPT-2 (Radford et al., 2019) to calculate the perplexity difference

---

[2] These embeddings are post-processed to ensure that the nearest neighbours are synonyms.

[3] `https://wordnet.princeton.edu`

[4] We use the off-the-shelf Stanford tagger. (`https://nlp.stanford.edu/software/tagger.html`)

[5] Non-synonym substitutes often reduce fluency.

[6] `https://pypi.org/project/language_tool`

between $\boldsymbol{x}$ and $\boldsymbol{x}_i^c$ for each candidate c:

$$\Delta\text{ppl}(\boldsymbol{x}, c, i) = \text{ppl}(\boldsymbol{x}_i^c) - \text{ppl}(\boldsymbol{x}), \qquad (2)$$

where $\boldsymbol{x}_i^c$ is $\boldsymbol{x}$ with its i-th word replaced by c, and filter out c whose $\Delta\text{ppl}(\boldsymbol{x}, c, i) > \epsilon_p$.

2.4 Best Substitute Searching

In this step, we greedily search for the best possible combination of substitutions, relying both on the previously created candidate lists and word substitution order. To preserve the syntactic structure of sentences, we forbid replacement of pronouns, articles, conjunctions, numerals, interjections, interrogative determiners and punctuation. Additionally, we set the maximum percentage of words allowed to be modified γ in the experiments to control the modification number.

Specifically, given a sentence $\boldsymbol{x}$, we substitute the words following the order computed in the word importance ranking step. For each target word x_i, we build an adversarial example $\boldsymbol{x}_i^c = x_1 x_2 \ldots c \ldots x_N$ for each of its substitutes c. Then we compute the accuracy change score from $\boldsymbol{x}$ to $\boldsymbol{x}_i^c$ as input to the parser:

$$\begin{aligned} S(\boldsymbol{x}, c, i) = &\lambda_{arc}\Delta_{\text{UAS}}(\boldsymbol{x}, \boldsymbol{x}_i^c) + \\ &(1 - \lambda_{arc})\Delta_{\text{LAS}}(\boldsymbol{x}, \boldsymbol{x}_i^c), \end{aligned} \qquad (3)$$

where $\Delta_{\text{UAS}}(\boldsymbol{x}, \boldsymbol{x}_i^c) = \text{UAS}_{F(\boldsymbol{x})} - \text{UAS}_{F(\boldsymbol{x}_i^c)}$ and $\Delta_{\text{LAS}}(\boldsymbol{x}, \boldsymbol{x}_i^c) = \text{LAS}_{F(\boldsymbol{x})} - \text{LAS}_{F(\boldsymbol{x}_i^c)}$ are the changes in UAS and LAS, respectively. If the percentage of modified words in the sentence exceeds a threshold γ, we stop the process. Otherwise, we search for a substitute for the next target word.

3 Experimental Setup

3.1 Target Parsers and Token Embeddings

We choose the following two strong and commonly used English parsers, one graph-based, the other transition-based, as target models, both of which achieve performance close to the state-of-the-art.

Deep Biaffine Parser (Dozat and Manning, 2017) is a graph-based parser that scores each candidate arc independently and relies on a decoding algorithm to search for the highest-scoring tree.

Stack-Pointer Parser (Ma et al., 2018) is a transition-based parser that incrementally builds the dependency tree with pre-defined operations.

We used the following four types of token embeddings to study their influence on each parsers' robustness. To focus on the influence of the embeddings, we use only the embeddings as input to the parsers:

GloVe (Pennington et al., 2014) is a frequently used static *word embedding*.

RoBERTa (Liu et al., 2019) is a pre-trained language model based on a masked language modelling object, which learns to predict a randomly masked token based on its context. It produces contextualised *word piece embeddings*.

ELECTRA (Clark et al., 2020) is a pre-trained language model based on a replaced token detection object, which learns to predict whether each token in the corrupted input has been replaced. It produces contextualised *word piece embeddings*.

ELMo (Peters et al., 2018) is a pre-trained language representation model based on *character embeddings* and bidirectional language modelling.

3.2 Datasets and Experimental Settings

We train the target parsers and evaluate the proposed method on the English Penn Treebank (PTB) dataset,[7] converted into Stanford dependencies using version 3.3.0 of the Stanford dependency converter (de Marneffe et al., 2006) (PTB-SD-3.3.0). We follow the standard PTB split, using section 2-21 for training, section 22 as a development set and 23 as a test set.

It is important to note that when converting PTB into Stanford dependencies, Zheng et al. (2020) maintained the copula (linking verbs) as a head when its complement was an adjective or noun.[8] However, since the design objective of Stanford dependency is to maximize dependencies between content words (de Marneffe et al., 2006), a more typical setting is to regard copulas as auxiliary modifiers. Therefore, we first compare with the previous method by performing this step under their settings and further conduct experiments with the typical PTB-SD-3.3.0 dataset for the convenience of follow-up research.

While training the target parsers, we adopt the hyper-parameters from their respective papers. Note that to compare with the biaffine parser, which uses first-order features, we also adopt the basic setting for the stack-pointer parser.[9] When using

---

[7] `https://catalog.ldc.upenn.edu/LDC99T42`

[8] Referred to as PTB-SD-3.3.0-COP in the rest of the paper.

[9] According to our preliminary experiments, neither second-order features nor beam search has an obvious influence on the parser robustness under our attack.

RoBERTa, ELECTRA or ELMo embeddings as input, we set the learning rate of these pre-trained models to 2e-5 and that of other parameters to 2e-2.

For the hyper-parameters of each attacking method, we set the word embedding similarity threshold $\epsilon_w = 0.7$, the candidate perplexity difference threshold $\epsilon_p = 20.0$, the arc importance coefficient $\lambda_{arc} = 0.5$ and the maximum percentage of words allowed to be modified $\gamma = 15\%$.

3.3 Evaluation Metrics

As introduced in Section 2.1, two constraints should be satisfied for an adversarial example to be valid: i) the perturbation is imperceptible, ii) the true dependency tree of x^* should be the same as that of x. For the first, we use **fluency** to measure the imperceptibility of the perturbations, and assume that in a fluent adversarial example the perturbation is imperceptible. For the second, **syntactic structure preservation** is used to measure whether an adversarial example's true dependency tree is identical to that of the original text. Both automatic and human evaluations are used for analysis.

In the automatic evaluation, GPT-2 (Radford et al., 2019) is used to compute the average perplexity of the adversarially modified PTB test set to measure the overall fluency. In the human evaluation, we ask three annotators to evaluate the quality of adversarial examples in two aspects, namely syntactic structure preservation and fluency.[10] To evaluate the preservation of the syntactic structure, we randomly collect 100 sentences along with their adversarial examples and ask the annotators to decide whether the syntactic structure is preserved in each case. For the fluency evaluation, we randomly collect 100 sentences along with the adversarial examples generated by our method and those produced by the black-box method of Zheng et al. (2020).[11] For each sentence, the annotators are asked to distinguish which example is better with regard to fluency. For both evaluations, we adopt the majority vote for the final results.

To evaluate how successful the attack is, we report the parsing results of the target models on the original and the adversarially modified (*after-attack*) PTB test set. The results are reported in terms of unlabelled attachment score (UAS) and labelled attachment score (LAS). We also report the attack success rate, namely the percentage of successfully attacked sentences. If the prediction accuracy of the modified sentence is lower than the original one, it is regarded as a successful attack.[12]

4 Results

4.1 Comparison with Previous Work

We first evaluate our attacking method on PTB-SD-3.3.0-COP and compare it with previous work (Zheng et al., 2020). Since we focus on the black-box attack in this paper, we compare with their sentence-level black-box attack against the deep biaffine parser with only word-based embeddings as input. In both their and our settings, 15% of words are allowed to be modified.

Model	**Automatic**	**Human**	
	PPL	**Fluency%**	**Syntax%**
Zheng et al.	267.96	20	75
Ours	**139.99**	**80**	**85**

Table 2: Automatic and human evaluation results on the PTB-SD-3.3.0-COP test set. PPL denotes the average perplexity. Syntax% denotes the preserved syntactic-structure rate and Fluency% the higher fluency rate.

Table 2 shows that adversarial examples generated by our method substantially outperform the previous method with regard to fluency and syntactic structure preservation. In the automatic evaluation, the average perplexity of examples generated by our method is 139.99, as compared to 267.96 of those generated by the previous work. For comparison, the average perplexity of the original PTB test set is 127.67, which is very close to ours.

In the human evaluation, results show that for 80% of the sentences, our adversarial examples have better fluency, which further confirms the effectiveness of our method. In addition, 85% of the examples we generated preserve the original syntactic structure, as compared to 75% reported by Zheng et al. (2020), showing that our method also improves the syntactic-structure preservation rate.

Table 3 shows the attack results of the two methods.[13] It is clear that with higher quality, the adversarial examples generated by our method cause

---

[10] The three human annotators are postgraduate students with a few years of research experience in syntactic parsing.

[11] We thank Zheng et al. (2020) for kindly providing us with the adversarial examples they generated.

[12] Note that Zheng et al. (2020) only considered unlabelled scores, so when comparing with these, we use the difference in UAS as the measurement of successful attacks. Conversely, in experiments on PTB-SD-3.3.0, we use the difference in LAS.

[13] We only compare UAS here since they did not report LAS in their paper.

Model	Orig-UAS	After-UAS	Succ%
Zheng et al.	95.52	88.69	51
Ours	95.45	88.95	44

Table 3: Results on the PTB-SD-3.3.0-COP test set. Orig-/After-UAS denotes the original and after-attack UAS respectively. Succ% denotes the success rate.

fewer incorrect predictions. This suggests that some of the previous attacks that were counted as successful may have used invalid adversarial examples which are either ungrammatical or which have actually changed the original syntactic structure.

Generator	BERT	Emb.	Sem.	Syn.
before	28.64	20.54	8.57	1.74
after	0.54	2.06	0.46	0.17
left%	1.89	10.04	5.32	10.03

Table 4: The average number of candidates **before** and **after** filtering generated by BERT-based (**BERT**), embedding-based (**Emb.**), sememe-based (**Sem.**) and synonym-based (**Syn.**) methods respectively. And the percentages of the **left** candidates.

To further demonstrate the limitation of the BERT-based method which the previous work used as the only candidate generator, we count the average number of candidates from our use of different generators before and after filtering. Results in Table 4 show that although the BERT-based method generates the most candidates before filtering, only 1.89% of them are left after the filters are applied. Whereas the left candidate percentage varies from 5% to 10% for the other three generators. The results further verify that the quality of candidates generated by the BERT-based method is worse than that from the embedding-based, sememe-based and synonym-based methods.

Model	After-UAS	Succ%	PPL
pos	72.21	89	326.06
pos+emb	79.87	75	286.92
pos+emb+gra	81.52	71	254.95
pos+emb+gra+ppl	88.95	44	139.99

Table 5: Ablation study of filters. **pos**, **emb**, **gra** and **ppl** stand for POS (part of speech), word embedding similarity, grammar checker and perplexity filters respectively.

To evaluate the ability of the filters, we conduct an ablation study with different combinations of these filters. Results in Table 5 show that the perplexity as well as the attack success rate decreases when more filters are applied. As expected, the greatest perplexity drop is brought by the perplexity filter.

4.2 Robustness Evaluation of Different Models

Input	Original		After-Attack		Succ%
	UAS	LAS	UAS	LAS	
			Biaffine		
G.	95.36	93.49	88.69	85.09	55.3
M.	96.29	94.51	90.70	87.67	47.5
E.	97.12	95.38	91.05	87.79	50.6
R.	97.09	95.41	92.14	89.42	46.1
			Stack-Pointer		
G.	94.93	93.05	88.26	84.64	52.6
M.	95.69	93.77	89.57	86.49	46.8
E.	96.94	95.19	90.69	87.47	50.3
R.	96.93	95.20	91.58	88.84	45.1

Table 6: Robustness evaluation results. Succ% denotes the success rate (computed based on LAS). **G.**, **M.**, **E.** and **R.** stand for GloVe, ELMo, ELECTRA and RoBERTa respectively.

We evaluate the robustness of the different parsing models introduced in Section 3.1 on PTB-SD-3.3.0 and report the results in Table 6. First of all, when applied to unperturbed sentences, the graph-based deep biaffine parser performs consistently better than the transition-based stack-pointer parser (using the same embeddings). Among the four kinds of embeddings, the word piece-level embeddings (i.e., ELECTRA and RoBERTa) achieve the highest results, while GloVe yields the lowest results.

As for the adversarially modified sentences, we find that the drop in performance is close between the two families of parsers (using the same embeddings), while the attack success rate against the Stack-Pointer parser is slightly higher. In terms of the embeddings, RoBERTa turns out to be the most robust one, which has the lowest attack success rate and achieves the highest performance on the generated adversarial examples. ELMo is also a comparatively robust embedding. We are surprised to find that although ELECTRA achieves similar performance to RoBERTa on clean input data, it performs poorly on the adversarial examples. We hypothesise that this is due to ELECTRA's training objective, i.e. learning to predict whether a token in a corrupted sentence is genuine or not. With this objective, some of our substitutes can be predicted as incorrect tokens, yielding token representations in the space not encountered by the parser in training, and hence damaging its performance. Lastly, GloVe is the most vulnerable embedding. [14]

---

[14] To evaluate the stability of the attack, for each parsing

Vocab.	**Original**				**After-Attack**				**Succ%**
	UAS	**LAS**	**OOV**	**OOT**	**UAS**	**LAS**	**OOV**	**OOT**	
50k	95.36	93.49	2	24	87.58	83.73	972	1285	59.5
400k	95.36	93.49	0	15	88.69	85.09	2	906	55.3
400k (T.)	95.36	93.49	0	8	90.06	87.26	0	0	45.5

Table 7: OOV and OOT test results. **Vocab.** stands for vocabulary size, **T.** means filtering out all the candidates that have not appeared in the training set.

4.3 Out-of-Vocabulary and Out-of-Training Words

In this section, we investigate the roles out-of-vocabulary (OOV, words not in the embedding's vocabulary) and out-of-training (OOT, words not in the training set of the parser) words play in dependency parsing attacks. We perform attacks on the Biaffine GloVe models trained with (i) 50k vocabulary (**50k**), (ii) 400k vocabulary (**400k**) and (iii) the same 400k vocabulary but where all candidates not in the training set are filtered out (**400k (T.)**).

The results are shown in Table 7, where we report the attack results along with the number of OOV and OOT words in the adversarially modified words before and after the attack. Firstly, by comparing the OOV and OOT numbers before and after the attack in the **50k** model, we find that words chosen to be replaced are often non-OOV and non-OOT, while their substitutes are often OOV and OOT. Secondly, the comparison between the **50k** and **400k** results shows that when the number of OOV words decreases, the robustness of the model increases. Therefore, it is reasonable to assume that OOV words in adversarial examples cause incorrect predictions. Thirdly, according to the **400k** and **400k (T.)** results, when the number of OOT words in adversarial examples are reduced to 0 by filtering out all the OOT candidates, the attack success rate drops substantially. Therefore, we have reason to believe that unfamiliar OOT words are another factor degrading a parser's performance.

The OOV problem mostly appears in models using word-level embeddings such as GloVe and can be alleviated by simply increasing the vocabulary size. While for the OOT problem, one potential solution is using adversarial training, where a new parser is trained with a mixture of clean training data and adversarial examples.

4.4 Adversarial Training

Previous work (Zheng et al., 2020; Han et al., 2020) used a limited number (from 2,000 sentences to half of the training data) of adversarial examples in adversarial training as (Zheng et al., 2020) argued that overuse of them may lead to a performance drop on the clean data. In this section, we investigate the adversarial training strategies on all the parsing models introduced in Section 3.1. Specifically, we generate adversarial examples for the whole PTB training set and retrain parsers on different amount of adversarial examples along with the original training set. Figure 1 shows that as the number of adversarial examples used in adversarial training increases, the robustness of the models increases accordingly. For most of the models, the increase of robustness stops between 50% and 70% of adversarial examples used.

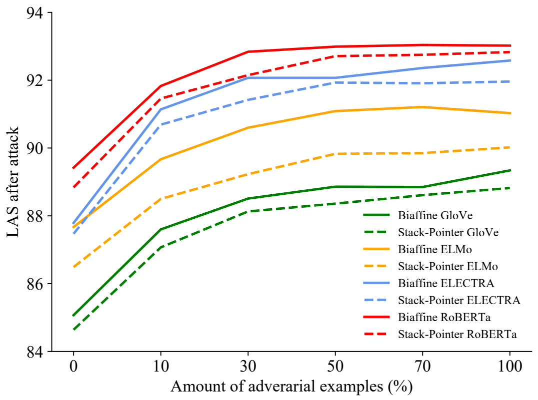

Figure 1: After-attack results with different amounts of adversarial examples used for adversarial training (best viewed in colour).

Table 8 shows the results of Biaffine parsers retrained with 100% of the adversarial examples generated for the original training set. We find that in most cases, the parsing results on the clean data are not obviously influenced although all of the adversarial examples are used. In addition, the robustness of all of the retrained models is substantially improved.

---

model in Table 6 we attack another two trained with different random seeds. The experiment shows all the results are stable across seeds.

Input	Original		After-Attack		Succ%
	UAS	LAS	UAS	LAS	
Biaffine					
G.	95.36	93.49	88.69	85.09	55.3
G.*	95.32	93.45	91.90	89.34	38.5
M.	96.29	94.51	90.70	87.67	47.5
M.*	96.17	94.37	93.49	91.03	33.2
E.	97.12	95.38	91.05	87.79	50.6
E.*	96.96	95.23	95.03	92.58	33.4
R.	97.09	95.41	92.14	89.42	46.1
R.*	97.03	95.30	95.30	93.02	29.5
Stack-Pointer					
G.	94.93	93.05	88.26	84.64	52.6
G.*	94.92	93.04	91.58	88.82	36.2
M.	95.69	93.77	89.57	86.49	46.8
M.*	95.75	93.81	92.64	90.02	34.1
E.	96.94	95.19	90.69	87.47	50.3
E.*	96.83	95.04	94.53	91.96	34.2
R.	96.93	95.20	91.58	88.84	45.1
R.*	96.80	95.01	95.10	92.83	29.1

Table 8: Adversarial training results. * denotes models with adversarial training.

4.5 Transferability

We refer to adversarial examples as transferable if, generated against one model, they succeed in fooling another one. Previously, Jin et al. (2020) found that in text classification and entailment tasks, adversarial examples are moderately transferable between models with different embeddings. In this section, we examine the following three kinds of transferabilities of adversarial examples in dependency parsing attacks: (i) **Cross Seed**: adversarial examples generated against one model are tested on another model trained with a different random seed; (ii) **Cross Parser**: adversarial examples generated against one model are tested on another from a different family of parsers; and (iii) **Cross Embedding**: adversarial examples generated against one model are tested on another trained with a different type of embedding.

Src	Cross Seed	Cross Parser	Cross Embedding			
			G.	M.	E.	R.
G.	45.0	40.3	**55.3**	27.7	27.2	25.2
M.	36.3	35.6	28.0	**47.5**	29.6	27.2
E.	34.8	40.2	27.1	28.8	**50.6**	29.2
R.	39.8	35.0	25.5	27.9	29.8	**46.1**

Table 9: Attack success rates (%) in the transferability test with Biaffine parser as the source parser. **Src** represents the source model.

Results in Table 9 and 10 show that the attack success rate always drops when adversarial examples are tested on other models, indicating that the adversarial examples strongly depend on the parser model, the token embeddings and even the specific instantiation of the model (i.e., the random seed). Among the three kinds of transferabilities, the cross seed transfer is the strongest while the cross embedding transfer is the weakest.

Src.	Cross Seed	Cross Parser	Cross Embedding			
			G.	M.	E.	R.
G.	41.1	40.8	**52.6**	25.8	25.7	23.7
M.	36.0	34.8	26.5	**46.8**	27.0	26.6
E.	35.3	33.7	24.4	28.5	**50.3**	29.9
R.	41.1	38.8	24.2	26.0	29.7	**45.1**

Table 10: Attack success rates (%) in the transferability test with Stack-Pointer parser as the source parser. **Src.** represents the source model.

4.6 Cross-Seed and Cross-Embedding Ensemble

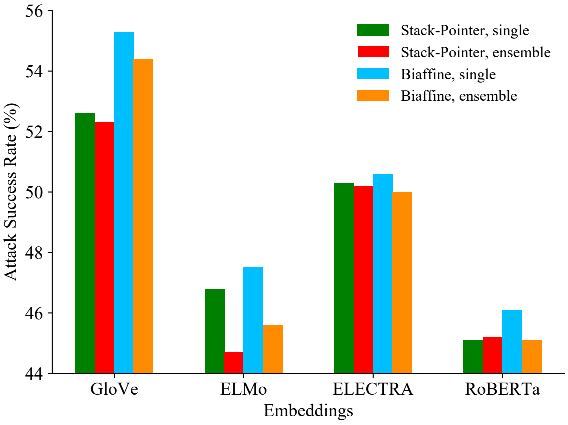

Figure 2: Attack success rates of the examples with and without cross-seed ensemble.

Based on the observations from Section 4.5, we propose to improve the robustness of parsing models using a cross-seed ensemble and cross-embedding ensemble. To ensemble multiple parsers, we simply compute the average of the probability distributions across them and use that result as the new distribution in the ensembled model.

Figure 2 shows the effect of the cross-seed ensemble, where almost all the attack success rates are dropped with such an ensemble. In addition, it is most effective with ELMo while least effective with ELECTRA and RoBERTa.

Table 11 shows the effect of using the cross-embedding ensemble, where robustness increases when more models with different token embeddings are ensembled. Moreover, contrary to adversarial training, the ensemble method is not tuned to specific types of attacks and appears robust to 'unseen' attacks, showing that it is more likely to defend against new attacks.

Input	**Original**		**After-Attack**		**Succ%**
	UAS	LAS	UAS	LAS	
Biaffine					
R.	97.09	95.41	92.14	89.42	46.1
R.G.	97.16	95.55	92.39	89.77	43.8
R.G.M.	97.20	95.58	92.53	89.97	42.5
R.G.M.E.	**97.25**	**95.63**	**92.73**	**90.12**	**41.5**
Stack-Pointer					
R.	96.93	95.20	91.58	88.84	45.1
R.G.	97.01	95.32	92.15	89.49	43.4
R.G.M.	96.98	95.26	**92.28**	89.62	42.3
R.G.M.E.	**97.14**	**95.45**	92.27	**89.64**	**41.5**

Table 11: Cross-embedding ensemble results

5 Related Work

Existing textual adversarial attacks have mostly focused on semantic tasks such as sentiment analysis (Zhang et al., 2019) and textual entailment (Jin et al., 2020). Although most of this work has applied various techniques to maintain the fluency of adversarial examples, a recent study by Morris et al. (2020) reported that quite a number of these techniques introduce grammatical errors.

In syntactic tasks, Zheng et al. (2020) recently proposed the first dependency parser attacking method which depends entirely on BERT to generate candidates. However, we show that the quality of adversarial examples generated by their method is relatively low due to the limitation of the BERT-based generator, and we propose to generate better examples by using more generators and stricter filters.

Han et al. (2020) proposed an approach to attack structured prediction models with a seq2seq model (Wang et al., 2016) and evaluated this model on dependency parsing. They used two reference parsers in addition to the victim parser to supervise the training of the adversarial example generator, and found that the three parsers produce better results when they have different inductive biases embedded to make the attack successful. This finding is quite close in spirit to our conclusion in Section 4.5. Hu et al. (2020) also put forth efforts to modify the text in syntactic tasks while preserving the original syntactic structure. However, their goal is to preserve privacy via the modification of words that could disclose sensitive information.

6 Conclusion

In this paper, we propose a method for generating high-quality adversarial examples for dependency parsing and show its effectiveness based on automatic and human evaluation. We investigate the robustness of different types of neural dependency parsers. We show that OOV and OOT words are two critical characteristics that cause a performance drop and propose to solve the OOT problem with adversarial training. We further examine three kinds of transferabilities of adversarial examples and propose to improve the robustness of parsing models by ensembling across random seeds and token embeddings.

Acknowledgments

This work was supported by the National Key R&D Program of China via grant 2020AAA0106501 and the National Natural Science Foundation of China (NSFC) via grant 61976072 and 61772153.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhendong Dong and Qiang Dong. 2006. *Hownet And the Computation of Meaning*. World Scientific Publishing Co., Inc., USA.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wenjuan Han, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020. Adversarial attack and defense of structured prediction models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2327–2338, Online. Association for Computational Linguistics.

Zhifeng Hu, Serhii Havrylov, Ivan Titov, and Shay B. Cohen. 2020. Obfuscation for privacy-preserving syntactic parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 62–72, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3).

Xiaoqing Zheng, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6600–6610, Online. Association for Computational Linguistics.

P-Stance: A Large Dataset for Stance Detection in Political Domain

Yingjie Li♣ **Tiberiu Sosea**♣ **Aditya Sawant**♣ **Ajith Jayaraman Nair**♣
Diana Inkpen♢ **Cornelia Caragea**♣
♣Computer Science, University of Illinois at Chicago
♢Electrical Engineering and Computer Science, University of Ottawa
♣`{yli300,tsosea2,asawan8,anair34,cornelia}@uic.edu`
♢`{diana.inkpen}@uottawa.ca`

Abstract

Stance detection determines whether the author of a text is in favor of, against or neutral to a specific target and provides valuable insights into important events such as presidential election. However, progress on stance detection has been hampered by the absence of large annotated datasets. In this paper, we present P-STANCE, a large stance detection dataset in the political domain, which contains 21,574 labeled tweets. We provide a detailed description of the newly created dataset and develop deep learning models on it. Our best model achieves a macro-average F1-score of 80.53%, which we improve further by using semi-supervised learning. Moreover, our P-STANCE dataset can facilitate research in the fields of cross-domain stance detection such as cross-target stance detection where a classifier is adapted from a different but related target. We publicly release our dataset and code.[1]

1 Introduction

Nowadays, people often express their stances toward specific targets (e.g., political events or figures, religion, or abortion) on social media. These opinions can provide valuable insights into important events, e.g., presidential election. The goal of the stance detection task is to determine whether the author of a piece of text is in favor of, against, or neutral toward a specific target (Mohammad et al., 2016b; Küçük and Can, 2020; ALDayel and Magdy, 2021). Twitter as a social platform has produced a large quantity of user-generated content, which has become a rich source for mining useful information about various topics such as presidential election. Political figures, who usually receive considerable attention and involve themselves in a large number of political events, are great targets to study stance detection. Therefore, detecting the stance expressed toward political figures on Twitter has drawn a lot of attention in the NLP community (Mohammad et al., 2016a; Sobhani et al., 2017; Darwish et al., 2017).

Even though stance detection has received a lot of attention, the annotated data are usually limited, which poses strong challenges to supervised models. Moreover, a limitation of existing datasets is that explicit mentions of targets and surface-level lexical cues that may expose the stance can be widely observed in the data (Mohammad et al., 2016a; Sobhani et al., 2017; Swami et al., 2018; Darwish et al., 2018; Conforti et al., 2020b; Lai et al., 2020), which means a model can detect the stance without extracting effective representations for the meanings of sentences (i.e., their lexical and compositional semantics). Another limitation of existing datasets, especially the datasets built on social media, is that the average length of tweets is short, which indicates that the data in these previous datasets are less informative and thus the stance can be detected more easily.

In an effort to minimize these drawbacks, we present P-STANCE, a dataset for stance detection whose primary goal is to bridge these gaps by making it possible to run large-scale evaluations that require a deeper semantic understanding. This large annotated dataset is composed of 21,574 English tweets in the political domain and each tweet is annotated with a stance toward one of three different targets: “Donald Trump,” “Joe Biden,” and “Bernie Sanders.” Examples from our dataset and their stance labels are shown in Table 1.

The main motivation of building this dataset is to provide a new benchmark for in-target stance detection where a classifier is trained and validated on the same target. However, we show additional interest in constructing a large corpus to facilitate research on cross-target stance detection where a classifier is adapted from different but related target.

---

[1] `https://github.com/chuchun8/PStance`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2355–2365
August 1–6, 2021.

Target	Tweet	Stance
Donald Trump	I agree, but not convinced Barr has all the evidence for his opinion. Although, I have zero worries about POTUS being re-elected, if the evidence is compelling enough on top of the tyrannical Covid lockdowns, Im hopeful more people will wake up. #GiantRedPill #Trump	Favor
Donald Trump	Take my kids, for example. At least, I'm TOLD they're my kids. No proof. Don Jr, Ivanka and Eric were all born to an immigrant woman who WASN'T a US Citizen when they were born. They shouldn't have US Citizenship. DEPORT THEM ALL! #Trump	Against
Bernie Sanders	Air borne illnesses will only become more common with climate change. We need to immediately address this and fight for Medicare for All or this could be the new normal. #BernieSanders	Favor
Bernie Sanders	A meat tax? Paying off all medical bills? I think #bernie has truly gone off the deep end of the pander cliff. None of these socialists insane, pie-in-the-sky policies would EVER work, or even come in to fruition yet people continue to fall for it. Unbelievable. #foxandfriends	Against
Joe Biden	Robyn Seniors, National HBCU Students for Biden Co-Chair and a @FAMU_1887 student, says that she's thankful that a "woman will be Vice President in a Biden administration."	Favor
Joe Biden	The Ukrainians are smarter than our own democratic party! Shoot, my Dogs are smarter than our own democratic party!! #ImpeachmentHoax #NoQuidProQuo #Biden	Against

Table 1: Examples from our P-STANCE dataset.

More interestingly, P-Stance enables a new task in stance detection, which is cross-topic stance detection where a classifier is adapted from the same target but with different topics in the past. These tasks, which use labeled training data of a source target and aim to train a model that generalizes well to a destination target with a shifted distribution, hold great practical value.

Our contributions include the following: 1) We present P-STANCE, a large dataset for stance detection composed of 21,574 tweets sampled from over 2.8 million tweets collected from Twitter. P-STANCE is more than three times larger than the previous benchmark (Mohammad et al., 2016a) and brings additional challenges such as linguistic complexities. We provide a detailed description and a comprehensive analysis of this dataset; 2) We conduct experiments on the proposed P-STANCE dataset and establish a strong baseline based on BERTweet (Nguyen et al., 2020). BERTweet achieves a macro-average F1-score of 80.53%, which we improve further by using semi-supervised learning; 3) The union of P-STANCE and previous benchmark datasets provides more opportunities for studying other stance detection tasks, e.g., cross-target stance detection and cross-topic stance detection.

2 Related Work

The most common stance detection task on social media is target-specific stance detection (ALDayel and Magdy, 2021) which aims to identify the stance toward a set of figures or topics (Hasan and Ng, 2014; Mohammad et al., 2016a; Xu et al., 2016; Taulé et al., 2017; Swami et al., 2018; Zotova et al., 2020; Conforti et al., 2020b; Lai et al., 2020; Vamvas and Sennrich, 2020; Conforti et al., 2020a). Besides target-specific stance detection, multi-target stance detection (Sobhani et al., 2017; Darwish et al., 2017; Li and Caragea, 2021a), and claim-based stance detection (Qazvinian et al., 2011; Derczynski et al., 2015; Ferreira and Vlachos, 2016; Bar-Haim et al., 2017; Rao and Pomerleau, 2017; Derczynski et al., 2017; Gorrell et al., 2019) are other popular trends of stance detection. Multi-target stance detection aims to jointly identify the stance toward two or more targets in the same text. Unlike the target-specific stance detection and multi-target stance detection where the target is usually a prominent figure or topic, in claim-based stance detection the target is a claim, which could be an article headline or a rumor's post.

Interestingly, despite substantial progress on stance detection, large-scale annotated datasets are limited. We compare our P-STANCE dataset with some existing stance detection datasets in Table 2. We can observe that the sizes of existing stance detection datasets are smaller than ours except for the WT-WT dataset (Conforti et al., 2020b) in the financial domain. However, the average tweet length of WT-WT is much shorter when compared with our P-STANCE. Moreover, more explicit mentions of targets and lexical cues of stance appear in the sentences of WT-WT dataset. In our work, we focus on the political domain and our P-STANCE, which contains much longer sentences and less surface-level lexical cues, can serve as a new challenging benchmark for stance detection tasks.

Authors	**Target(s)**	**Source**	**Type**	**Size**
Mohammad et al. (2016a)	Atheism, Climate change is a real concern, Feminist movement, Hillary Clinton, Legalization of abortion, Donald Trump	Twitter	Target-specific	4,870
Ferreira and Vlachos (2016)	Various claims	News articles	Claim-based	2,595
Sobhani et al. (2017)	Trump-Clinton, Trump-Cruz, Clinton-Sanders	Twitter	Multi-target	4,455
Derczynski et al. (2017)	Various claims	Twitter	Claim-based	5,568
Swami et al. (2018)	Demonetisation in India in 2016	Twitter	Target-specific	3,545
Gorrell et al. (2019)	Various claims	Twitter, Reddit	Claim-based	8,574
Conforti et al. (2020b)	Merger of companies: Cigna-Express Scripts, Aetna-Humana, CVS-Aetna, Anthem-Cigna, Disney-Fox	Twitter	Target-specific	51,284
Conforti et al. (2020a)	Merger of companies: Cigna-Express Scripts, Aetna-Humana, CVS-Aetna, Anthem-Cigna	News articles	Target-specific	3,291
P-STANCE	Donald Trump, Joe Biden, Bernie Sanders	Twitter	Target-specific	21,574

Table 2: Comparison of English stance detection datasets.

Different from classifying the stance detection tasks by target type (i.e., one specific target, multiple targets, or a claim), we can also categorize the stance detection as in-target and cross-target stance detection by the training setting. Most previous works focused on the in-target stance detection where a classifier is trained and validated on the same target (Mohammad et al., 2016b; Zarrella and Marsh, 2016; Wei et al., 2016; Vijayaraghavan et al., 2016; Du et al., 2017; Sun et al., 2018; Wei et al., 2018; Li and Caragea, 2019, 2021b). However, sufficient annotated data are usually hard to obtain and conventional models on stance detection perform poorly on generalizing to the data of new targets, which motivates the studies of cross-target stance detection (Augenstein et al., 2016; Xu et al., 2018; Wei and Mao, 2019; Zhang et al., 2020). Most previous studies evaluated the cross-target models on the SemEval-2016 dataset (Mohammad et al., 2016a), which is a small dataset and thus may make the conclusions less convincing.

In this paper, we show that our P-STANCE dataset can be also used to evaluate the model performance of cross-target stance detection and provides opportunities for exploring more cross-target tasks by interacting with previous SemEval-2016 (Mohammad et al., 2016a) and Multi-Target stance datasets (Sobhani et al., 2017). In addition, P-STANCE enables the exploration of large-scale deep learning models including pre-trained language models, e.g., BERT (Devlin et al., 2019) and BERTweet (Nguyen et al., 2020). We fine-tune the BERT and BERTweet models on our dataset and compare them with other strong baselines.

3 Building the Dataset

In this section, we detail the creation and the particularities of P-STANCE, our large political stance detection dataset composed of 21,574 tweets collected during the 2020 U.S. presidential election.

3.1 Data Collection

We collected tweets using the Twitter streaming API. Similar to prior works (Mohammad et al., 2016a; Sobhani et al., 2017) that target presidential candidates, we focus our attention on three political figures[2] in the presidential race of 2020: "Donald Trump," "Joe Biden," and "Bernie Sanders." We used a set of query hashtags as seeds to collect target-related tweets, which can be categorized as *favor hashtags*, *against hashtags* and *neutral hashtags* (Mohammad et al., 2016a). We show examples of these query hashtags in Table 3. In total, we gathered around 2.8 million tweets for all three targets combined.

3.2 Preprocessing

To ensure the quality of this dataset, we performed several preprocessing steps: **1)** We removed tweets with less than 10, or more than 128 words. According to our observations, tweets with less than 10 words are either too easy for detecting the stance or too noisy, and tweets with more than 128 words usually contain duplicate expressions. **2)** We removed duplicates and retweets. Twitter data are noisy not only due to the creative spellings, slang

---

[2] We also tried to collect tweets about the woman politician Kamala Harris. However, we were unable to collect enough data about Harris. We will look into this in our future work.

Target	Favor Hashtag	Against Hashtag	Neutral Hashtag
Trump	#Trump2020LandSlide	#TrumpCrimeFamily	#DonaldTrump #Republican
Biden	#BidenForPresident	#SleepyJoe	#JoeBiden #Democrats
Sanders	#BernieWon	#NeverBernie	#BernieSanders #Sanders

Table 3: Examples of query hashtags.

	Trump	Biden	Sanders
#Raw collection	1,730K	429K	654K
#After preprocessing	1,221K	300K	465K

Table 4: Number of unlabeled tweets before and after preprocessing.

Setup	Trump	Biden	Sanders	Average
3-class	0.62	0.60	0.59	0.60
2-class	0.86	0.81	0.76	0.81

Table 5: Krippendorff's alpha measure of annotator agreement in 3-class and 2-class scenarios.

and URLs, but also because of the duplicate tweets. Since these duplicate data reduce our ability to build reliable models, we need to clean the dataset by removing duplicates. **3)** We kept only the tweets in English because our goal in this work is to build an English stance detection dataset. We leave multilingual stance detection as future work. After data preprocessing, the size of our corpus reduces to around 2 million examples. In Table 4, we show the number of tweets before and after preprocessing for each political figure. We will provide this large-scale repository of tweets (which we call P-STANCE-EXT) alongside P-STANCE, in hope that it will spur further research in the field of semi-supervised learning for stance detection. Finally, we sampled 10,000 tweets for each political figure, obtaining 30,000 tweets for annotation in total.

3.3 Data Annotation

We gathered stance annotations of three targets through the Amazon Mechanical Turk (AMT) crowdsourcing platform. The AMT workers were asked to annotate each tweet with "Favor," "Against," "None," or "I don't know." To ensure the annotation quality, we employed strict requirements for the annotators: **1)** Many completed tasks (>500); **2)** To reside in the USA; **3)** A high acceptance rate (>95%). Moreover, we ran the annotation process in several batches of 1000 examples. In each batch, we include 100 internally annotated examples to measure the quality of the annotators. If an annotator mislabels more than 25% of these examples, we discard the annotations of the worker completely, and relabel them. Interestingly, this process led to a considerable number of reannotations, amounting for more than 20% of the data. Each tweet was labeled by three random annotators, and disagreements in the labels were decided by the majority voting among the three annotators.

After obtaining the annotation results, we computed Krippendorff's alpha (Krippendorff, 2011) as the measure of inter-annotator agreement, as shown in Table 5. Tweets that were annotated with label "I don't know" after the majority voting were removed from the dataset. We observed that annotators had difficulties in reaching an agreement on tweets with label "None" and the average of Krippendorff's alpha values increases from 0.60 to 0.81 when we consider two classes: "Favor" and "Against". Similar to prior work (Vamvas and Sennrich, 2020), we removed the label "None" from the dataset in our experiments.

3.4 Quality Assurance and Challenges

Stance-exposing hashtags that may expose the stance directly, e.g., #NeverBernie, can be observed in the data. A model can detect the stance from these hashtags without extracting effective representations for the meanings of sentences, which makes stance detection easier. To remove the stance-exposing hashtags and ensure the data quality, we performed the following steps after the data annotation: **1)** We manually built a hashtag lexicon that contains stance-exposing hashtags for each target. Then we removed all hashtags that are appended at the end of a sentence if they are in the hashtag lexicon. The reason of only removing the appended hashtags is that a hashtag may serve as a constituent of a sentence, so it would introduce more noise if we simply remove all stance-exposing hashtags. **2)** To address the stance-exposing hashtag that is a constituent of a sentence, we replaced stance-exposing hashtags that contain the target name with a neutral hashtag, e.g., #NeverBernie → #Bernie. These steps ensure

		Trump	Biden	Sanders
Train	Favor	2,937	2,552	2,858
	Against	3,425	3,254	2,198
Val	Favor	365	328	350
	Against	430	417	284
Test	Favor	361	337	343
	Against	435	408	292
Total		7,953	7,296	6,325

Table 6: Label distribution across different targets for P-STANCE.

the high quality of our P-STANCE dataset.

In addition, P-STANCE is a challenging dataset for the following reasons: **1)** Targets in P-STANCE are referred to in a more implicit way. Consider the second example in Table 1, the target name only appears at the end of the sentence and it is hard to correctly identify the stance without any knowledge about the political figures mentioned in the content and background immigration policy. Similarly, for the third example, it is difficult to correctly identify the stance if the classifier fails to connect the target with relevant events, i.e., climate change or medicare for all residents. **2)** The average length of tweets in previous datasets is short, and there are more explicit mentions of targets and rich sentiment and emotion words that can easily reveal the stance toward the target. The average tweet length is 17 in Mohammad et al. (2016a), 21 in Sobhani et al. (2017) and 16 in Conforti et al. (2020b). However, our P-STANCE has a much longer average length of 30 and more implicit mentions of targets and context words, which indicates that our dataset is more difficult. In addition, P-STANCE covers more target-relevant events. These characteristics contribute to making P-STANCE a challenging dataset for stance detection.

3.5 Dataset Distribution

The final dataset contains 7,953 annotated tweets for "Donald Trump", 7,296 for "Joe Biden" and 6,325 for "Bernie Sanders", respectively. The label distribution of each target is shown in Table 6. Each tweet is annotated with a stance label "Favor" or "Against". We created the training, validation and testing sets following an 80/10/10 split. We note that P-STANCE is more than 3 times larger than the previous benchmark (Mohammad et al., 2016a).

4 Experimental Settings

In this section, we first introduce two benchmark datasets of stance detection in §4.1. The union of these datasets and our P-STANCE dataset provides opportunities for studying the cross-target stance detection (§5.2) and cross-topic stance detection (§5.3). Then we discuss the evaluation metrics in §4.2 and introduce the baseline methods in §4.3.

4.1 Existing Benchmark Datasets

SemEval-2016 (Mohammad et al., 2016a) and Multi-Target stance datasets (Sobhani et al., 2017) are two benchmark datasets in which political figures are chosen as the targets. SemEval-2016 contains six targets: "Atheism," "Climate Change is a Real Concern," "Feminist Movement," "Hillary Clinton," "Legalization of Abortion," and "Donald Trump." The dataset is annotated for detecting the stance toward a given target. The data distribution of SemEval-2016 is shown in Table 7.

Multi-Target stance dataset contains three sets of tweets corresponding to three target pairs: "Donald Trump and Hillary Clinton," "Donald Trump and Ted Cruz," "Hillary Clinton and Bernie Sanders" for 2016 U.S. presidential election. The task aims at detecting the stances toward two targets for each data. The data distribution of Multi-Target stance dataset is shown in Table 8. In the next section, we show how to perform various stance detection tasks with the union of these datasets and our P-STANCE dataset.

4.2 Evaluation Metrics

Similar to Mohammad et al. (2017) and Sobhani et al. (2017), F_{avg} and macro-average of F1-score (F_{macro}) are adopted to evaluate the performance of our baseline models. First, the F1-score of label "Favor" and "Against" is calculated as follows:

$$F_{favor} = \frac{2P_{favor}R_{favor}}{P_{favor} + R_{favor}} \quad (1)$$

$$F_{against} = \frac{2P_{against}R_{against}}{P_{against} + R_{against}} \quad (2)$$

where P and R are precision and recall, respectively. After that, the F_{avg} is calculated as:

$$F_{avg} = \frac{F_{favor} + F_{against}}{2} \quad (3)$$

We compute the F_{avg} for each target. F_{macro} is calculated by averaging the F_{avg} across all targets.

4.3 Baseline Methods

We run experiments with the following baselines.

Target	**#Train**	%Favor	%Against	%None	**#Test**	%Favor	%Against	%None
Atheism	513	17.93	59.26	22.81	220	14.54	72.73	12.73
Climate	395	53.67	3.80	42.53	169	72.78	6.51	20.71
Feminism	664	31.63	49.40	18.97	285	20.35	64.21	15.44
Hillary	689	17.13	57.04	25.83	295	15.25	58.31	26.44
Abortion	653	18.53	54.36	27.11	280	16.43	67.50	16.07
Trump	0	-	-	-	707	20.93	42.29	36.78

Table 7: Data distribution of SemEval-2016 dataset.

Target Pair	**Total**	**Train**	**Dev**	**Test**
Trump-Clinton	1,722	1,240	177	355
Trump-Cruz	1,317	922	132	263
Clinton-Sanders	1,366	957	137	272
Total	4,455	3,119	446	890

Table 8: Data distribution of Multi-Target dataset.

BiLSTM (Schuster and Paliwal, 1997): A BiLSTM model that takes tweets as inputs without considering the target information.

CNN (Kim, 2014): Similar to BiLSTM, the vanilla CNN only takes tweets as inputs and does not consider the target information.

TAN (Du et al., 2017): TAN is an attention-based LSTM model that extracts target specific features.

BiCE (Augenstein et al., 2016): A BiLSTM that uses conditional encoding for stance detection. The target information is first encoded by a BiLSTM, whose hidden representations are then used to initialize another BiLSTM with tweets as inputs. BiCE is also a strong baseline for cross-target stance detection.

CrossNet (Xu et al., 2018): CrossNet is another model for cross-target stance detection. It encodes the target and the tweet by using the same approach with BiCE and add an aspect attention layer to signal the core part of a stance-bearing input. CrossNet improves BiCE in many cross-target settings.

GCAE (Xue and Li, 2018): A CNN model that utilizes a gating mechanism to block target-unrelated information. GCAE is a strong baseline for aspect-based sentiment analysis and we apply it to our stance detection task.

PGCNN (Huang and Carley, 2018): Similar to GCAE, PGCNN is based on gated convolutional networks and encodes target information by generating target-sensitive filters.

BERT (Devlin et al., 2019): A pre-trained language model that predicts the stance by appending a linear classification layer to the hidden representation of $[CLS]$ token. We fine-tune the BERT-base on the stance detection task.

BERTweet (Nguyen et al., 2020): BERTweet is another pre-trained language model following the training procedure of RoBERTa (Liu et al., 2019). Similar to BERT, we fine-tune the pre-trained BERTweet to predict the stance by appending a linear classification layer to the hidden representation of the $[CLS]$ token. The pre-trained BERTweet model is fine-tuned under the PyTorch framework. The maximum sequence length is set to 128 and the batch size is 32. We use AdamW optimizer (Loshchilov and Hutter, 2019) and the learning rate is 2e-5.

5 Results

In this section, we present the set of experiments performed on various stance detection tasks on our dataset and show the results obtained by using the aforementioned baselines. Each result is the average of seven runs with different initializations.

5.1 In-Target Stance Detection

In-target stance detection is a stance detection task where a classifier is trained and validated on the same target. Most previous works adopt an "Ad-hoc" training strategy by training one model for each target and evaluate it on the test set of that target (i.e., we train three different models if there are three targets in the dataset). However, the model is more likely to predict the stance by following specific patterns without fully considering the target information and overfit. Therefore, to better evaluate the performance of baselines, we propose a "Merged" training strategy by training and validating a model on all targets and testing it on separate targets to be compared with the "Ad-hoc" setting.

Experimental results of these two different settings are shown in Table 9. First, we can observe that BERTweet performs best in both settings and significantly outperforms the second best results,

Method	Trump	Biden	Sanders	F$_{\mathrm{macro}}$	Drop
Ad-hoc					
BiLSTM	76.92	77.95	69.75	74.87	-
CNN	76.80	77.22	71.40	75.14	-
TAN	77.10	77.64	71.60	75.45	-
BiCE	77.15	77.69	71.24	75.36	-
PGCNN	76.87	76.60	72.13	75.20	-
GCAE	78.96	77.95	71.82	76.24	-
BERT	78.28	78.70	72.45	76.48	-
BERTweet	**82.48**†	**81.02**†	**78.09**†	**80.53**	-
Merged					
BiLSTM	77.18	75.47	67.43	73.36	1.51
CNN	74.79	74.11	66.68	71.86	3.28
TAN	78.30	75.26	70.67	74.74	0.71
BiCE	77.67	75.69	69.37	74.24	1.12
PGCNN	77.36	74.96	70.29	74.20	1.00
GCAE	79.00	76.32	69.93	75.08	1.16
BERT	79.19	76.02	73.59	76.27	0.21
BERTweet	**83.81**†	**79.08**†	**77.75**†	**80.21**	0.32

Table 9: Comparison of different models on the P-STANCE dataset (%). †: BERTweet model improves the best baseline at $p < 0.05$ with paired t-test. F_{macro} is the average of all target pairs. "Drop" means performance decline between two training strategies for the same model. Bold scores are best overall.

demonstrating the effectiveness of this model. Second, performance drops can be observed on all models in the "Merged" setting and models (BiLSTM and CNN) that do not consider target information suffer the most severe drops, which means our proposed training strategy can serve as a better evaluation method to test whether the model learns target-specific representations. Moreover, we can observe that both BERTweet and BERT perform well and have the minimum performance drops compared with the other baselines, which demonstrates that self-attention mechanism can better capture target-specific representations.

5.2 Cross-Target Stance Detection

Despite substantial progress on the stance detection, sufficient annotated data are usually hard to obtain and conventional models on stance detection perform poorly on generalizing to the data of new targets, which motivates the studies of cross-target stance detection. The model of cross-target stance detection is first trained and validated on a source target, and then tested on a destination target. In this subsection, we show that our P-STANCE dataset can be also used to evaluate the model performance of cross-target stance detection and provides opportunities for exploring more cross-target tasks by interacting with previous SemEval-2016 and Multi-Target stance datasets.

Target	BiCE	CrossNet	BERTweet
P-STANCE dataset			
DT → JB	55.83	56.67	**58.88**
DT → BS	51.78	50.08	**56.50**†
JB → DT	58.16	60.43	**63.64**†
JB → BS	60.24	60.81	**67.04**†
BS → DT	51.41	52.99	**58.75**†
BS → JB	57.68	62.57	**72.99**†
DT, JB → BS	52.26	56.26	**69.99**†
DT, BS → JB	53.73	55.57	**68.64**†
JB, BS → DT	53.91	56.44	**66.01**†
P-STANCE → previous datasets			
DT → HC	36.12	**40.56**	34.48
DT → TC	59.37	59.40	**63.89**†
DT → BS	47.73	48.93	**51.00**†
JB → DT	48.90	49.77	**56.00**†
JB → HC	56.77	55.54	**57.55**
JB → TC	53.47	55.77	**62.45**†
JB → BS	48.11	48.96	**51.48**†
BS → DT	47.93	46.10	**49.96**
BS → HC	49.97	50.49	**52.81**
BS → TC	54.37	52.98	**56.91**†

Table 10: Comparison of different models for cross-target stance detection (%). The first half reports the cross-target results on our proposed P-STANCE dataset. The second half reports the cross-target results that are trained on the P-STANCE dataset and tested on the previous datasets. †: BERTweet model improves the best baseline at $p < 0.05$ with paired t-test. Bold scores are best overall.

We use five targets for our experiments: "Donald Trump" (DT), "Joe Biden" (JB), "Bernie Sanders" (BS), "Hillary Clinton" (HC), and "Ted Cruz" (TC). Experimental results of cross-target stance detection are shown in Table 10. For the first half of Table 10, only targets of P-STANCE dataset are used to evaluate the model performance. However, for the second half, targets of SemEval-2016 and Multi-Target datasets also serve as destination targets, which makes it a more challenging task since the target-related topics in 2016 are quite different from the ones in 2020. More specifically, we train and validate the model on a source target of P-STANCE dataset and test it on the data of a destination target, which is a combination of train, validation, and test sets of previous datasets. Note that we merge the data from SemEval-2016 and Multi-Target datasets if these two datasets share the same target, e.g., Hillary Clinton.

For the cross-target tasks only on the P-STANCE dataset, first, we can observe from the Table 10 that BERTweet achieves the best performance on all target configurations, demonstrating its effectiveness. Moreover, BERTweet shows greater improvement over the best baseline when training on the data of two targets. The reason is that BERTweet learns more universal representations by leveraging the data from two targets. Second, we see that CrossNet outperforms BiCE on almost all target configurations, which is consistent with the observations of previous studies (Xu et al., 2018; Zhang et al., 2020). Third, we find that models achieve better performance on JB → BS and BS → JB. One potential explanation is that targets "Joe Biden" and "Bernie Sanders" are from the same party and thus share more similar topics.

For the second half of Table 10, we observe a significant drop in performance on all models, which verifies that it is more challenging to transfer the knowledge to a destination target with more diverse topics in the past. BERTweet still achieves the best performance on almost all target configurations, making it a highly competitive model for cross-target stance detection task. Interestingly, we can observe that both BiCE, CrossNet, and BERTweet show better performance on target "Ted Cruz." A possible reason is that the data of "Ted Cruz" contain more universal expressions and topics.

5.3 Cross-Topic Stance Detection

Obtaining sufficient annotated data of specific target from most recent past is challenging. However, sometimes historical annotated data of the same target are available. Therefore, motivated by a desire to improve the models' generalization ability to transfer knowledge from historical data, we come up with a new stance detection task, named cross-topic stance detection. Specifically, in this task, the model of cross-topic stance detection is first trained on the data of a target (e.g., Donald Trump) in 2016, and then validated and tested on the data of the same target in 2020. Note that the annotated data of year 2016 are the same with the data used in §5.2. The results are shown in Table 11. Since target "Joe Biden" is absent from the previous stance detection datasets, we use targets "Donald Trump" and "Bernie Sanders" for evaluation. We can observe that BERTweet still performs best on this task and the overall model performance of cross-topic stance detection is better than that of cross-target stance detection due to the use of the same target in evaluation stage. Moreover, we see that models perform relatively poorly on target "Bernie Sanders". One possible explanation is that some topics, e.g. healthcare and climate change, appear rarely in previous datasets.

Target	BiCE	CrossNet	**BERTweet**
DT → DT	58.60	59.41	**73.58**†
BS → BS	59.04	57.66	**66.48**†

Table 11: Comparison of different models for cross-topic stance detection (%). †: BERTweet model improves the best baseline at $p < 0.05$ with paired t-test. Bold scores are best overall.

5.4 Semi-Supervised Stance Detection

During elections, there is a considerable amount of data generated by users expressing their opinions about candidates, out of which only a small amount can be annotated and used for supervised stance detection. We explore the potential of the abundant unlabeled tweets and show that we can leverage them to improve the performance of our models. To this end, we turn to semi-supervised learning, and leverage techniques such as Uncertainty-aware Self-Training (UST).

UST (Mukherjee and Awadallah, 2020) is a semi-supervised approach which uses the standard teacher-student self-training framework, but adds a few powerful changes. Concretely, UST designs different techniques which leverage the uncertainty of the teacher model to select the unlabeled set of examples in each self-training iteration. First, we train our teacher model on the labeled examples. Next, we compute uncertainty estimates of our teacher model on the set of unlabeled examples by performing a few forward passes with dropout enabled. Finally, we incorporate the uncertainty estimates into our framework as follows: **1)** We use these estimates to select the examples for which the teacher is most or least confident about. **2)** We incorporate the teacher confidence in the student loss by penalizing the student's misclassified examples in which the teacher has high confidence. We use the BERTweet model as teacher and student.

We perform various experiments to show the benefits of using a large amount of unlabeled data from P-STANCE-EXT alongside our UST model. We carry out three barely supervised experiments with various number of examples in the training set. Specifically, we experiment with 30, 50, and 100 training examples. Moreover, we also consider an experiment using the whole training set to investigate the effect of the unlabeled examples when all the training data are available. We run experiments with different training sets, and report the F1-scores obtained on the entire testing set.

We show the results of our semi-supervised ex-

Method	**Trump**	**Biden**	**Sanders**	$\mathbf{F_{macro}}$
BERTweet	82.48	81.02	78.09	80.53
UST-30	61.02	64.34	60.45	61.94
UST-50	68.42	73.24	66.12	69.26
UST-100	74.45	79.46	71.67	75.19
UST-ALL	**85.50**†	**82.22**†	**79.55**†	**82.42**

Table 12: Semi-supervised learning results. †: UST-ALL improves the BERTweet at $p < 0.05$ with paired t-test.

periments in Table 12 and make the following observations. First, UST-ALL significantly outperforms the BERTweet model by 1.89% in a macro-average F1-score when using both the labeled and unlabeled data in a semi-supervised manner. Second, with only 100 examples (2% of the available training examples), UST-100 stays within 1.6% F1-score of our best model that leverages the entire training set of target “Joe Biden.” The results indicate that the benefit of using semi-supervised approaches is two-fold. On one hand, it enables impressive performance in scarce label scenarios, while on the other hand, it still brings gains in scenarios where considerable amounts of labeled data are readily available.

6 Conclusion

In this paper, we introduced P-STANCE, an English stance detection dataset in the political domain, which is larger and more challenging compared with previous datasets for stance detection. Composed of 21,574 tweets that were collected during the 2020 USA election, P-STANCE can serve as a new benchmark for stance detection and enable future research in other stance detection tasks, e.g., cross-target stance detection and cross-topic stance detection. Experimental results show that the BERTweet model significantly outperforms other strong baselines not only on in-target stance detection, but also on cross-target and cross-topic stance detection. Moreover, the performance of BERTweet can be further improved by using semi-supervised learning. Future work includes constructing another large dataset for a more challenging task, i.e., multi-target stance detection, and studying the multilingual stance detection with the union of P-STANCE and other multilingual datasets.

Acknowledgments

We thank the National Science Foundation and Amazon Web Services for support from grants IIS-1912887 and IIS-1903963 which supported the research and the computation in this study. We also thank our reviewers for their insightful comments.

References

Abeer ALDayel and Walid Magdy. 2021. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261.

Costanza Conforti, Jakob Berndt, Mohammad Taher Pilehvar, Chryssi Giannitsarou, Flavio Toxvaerd, and Nigel Collier. 2020a. STANDER: An expert-annotated dataset for news stance detection and evidence retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4086–4101.

Costanza Conforti, Jakob Berndt, Mohammad Taher Pilehvar, Chryssi Giannitsarou, Flavio Toxvaerd, and Nigel Collier. 2020b. Will-they-won’t-they: A very large dataset for stance detection on Twitter. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1724.

Kareem Darwish, Walid Magdy, Afshin Rahimi, Timothy Baldwin, and Norah Abokhodair. 2018. Predicting online Islamophobic behavior after #ParisAttacks. *The Journal of Web Science*, 4(3):34–52.

Kareem Darwish, Walid Magdy, and Tahar Zanouda. 2017. Trump vs. Hillary: What went viral during the 2016 US presidential election. In *9th International Conference on Social Informatics (SocInfo 2017)*, pages 143–161.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76.

Leon Derczynski, Kalina Bontcheva, Michal Lukasik, Thierry Declerck, Arno Scharl, Georgi Georgiev, Petya Osenova, Tomas Pariente Lobo, Anna Kolliakou, Robert Stewart, Sara-Jayne Terp, Geraldine

Wong, Christian Burger, Arkaitz Zubiaga, Rob Procter, and Maria Liakata. 2015. PHEME: Computing Veracity — the Fourth Challenge of Big Social Data. In *Proceedings of the Extended Semantic Web Conference EU Project Networking session (ESCW-PN)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3988–3994.

William Ferreira and Andreas Vlachos. 2016. Emergent: A novel dataset for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 845–854.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? Identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 751–762.

Binxuan Huang and Kathleen Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1091–1096.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Dilek Küçük and Fazli Can. 2020. Stance detection: A survey. *ACM Comput. Surv.*, 53(1):1–37.

Mirko Lai, Alessandra Teresa Cignarella, Delia Irazú Hernández Farías, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2020. Multilingual stance detection in social media political debates. *Computer Speech & Language*, 63:101075.

Yingjie Li and Cornelia Caragea. 2019. Multi-task stance detection with sentiment and stance lexicons. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6298–6304.

Yingjie Li and Cornelia Caragea. 2021a. A multi-task learning framework for multi-target stance detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: Findings, ACL 2021, Online Event, August 2-4, 2021*. Association for Computational Linguistics.

Yingjie Li and Cornelia Caragea. 2021b. Target-aware data augmentation for stance detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1850–1860.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016a. A dataset for detecting stance in tweets. In *LREC*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016b. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Subhabrata Mukherjee and Ahmed Awadallah. 2020. Uncertainty-aware self-training for few-shot text classification. In *Advances in Neural Information Processing Systems*, volume 33, pages 21199–21212.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599.

Delip Rao and Dean Pomerleau. 2017. Fake News Challenge.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 551–557.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409.

Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. An English-Hindi code-mixed corpus: Stance annotation and baseline system. *arXiv preprint arXiv:1805.11868*.

Mariona Taulé, Maria Antònia Martí, Francisco M. Rangel Pardo, Paolo Rosso, Cristina Bosco, and Viviana Patti. 2017. Overview of the task on stance and gender detection in tweets on Catalan independence. In *Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017)*, pages 157–177.

Jannis Vamvas and Rico Sennrich. 2020. X-Stance: A multilingual multi-target dataset for stance detection. In *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. DeepStance at SemEval-2016 task 6: Detecting stance in tweets using character and word-level CNNs. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 413–419.

Penghui Wei and Wenji Mao. 2019. Modeling transferable topics for cross-target stance detection. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 1173–1176.

Penghui Wei, Wenji Mao, and Daniel Zeng. 2018. A target-guided neural memory model for stance detection in twitter. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pages 1–8.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at SemEval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 384–388.

Chang Xu, Cécile Paris, Surya Nepal, and Ross Sparks. 2018. Cross-target stance classification with self-attention networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 778–783.

Ruifeng Xu, Yu Zhou, Dongyin Wu, Lin Gui, Jiachen Du, and Yun Xue. 2016. Overview of NLPCC shared task 4: Stance detection in chinese microblogs. In *Natural Language Understanding and Intelligent Applications*, pages 907–916.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 task 6: Transfer learning for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 458–463.

Bowen Zhang, Min Yang, Xutao Li, Yunming Ye, Xiaofei Xu, and Kuai Dai. 2020. Enhancing cross-target stance detection with transferable semantic-emotion knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3188–3197.

Elena Zotova, Rodrigo Agerri, Manuel Nuñez, and German Rigau. 2020. Multilingual stance detection in tweets: The Catalonia independence corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1368–1375.

WIND: Weighting Instances Differentially for Model-Agnostic Domain Adaptation

Xiang Chen, Yue Cao, Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
Center for Data Science, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
`{caspar,yuecao,wanxiaojun}@pku.edu.cn`

Abstract

Domain Adaptation is a fundamental problem in machine learning and natural language processing. In this paper, we study the domain adaptation problem from the perspective of instance weighting. Conventional instance weighting approaches cannot learn the weights which make the model generalize well in target domain. To tackle this problem, inspired by meta-learning, we formulate the domain adaptation problem as a bi-level optimization problem, and propose a novel differentiable model-agnostic instance weighting algorithm. Our proposed approach can automatically learn the instance weights instead of using manually designed weighting metrics. To reduce the computational complexity, we adopt the second-order approximation technique during training. Experimental results[1] on three different NLP tasks (Sentiment Classification, Neural Machine Translation and Relation Extraction) illustrate the efficacy of our proposed method.

1 Introduction

Domain shift is a challenging problem which is commonly encountered in Natural Language Processing (NLP). Due to the data distribution discrepancy between source and target domain, the model trained on the data from source domain may fail to achieve satisfying performance in target domain. Therefore we face the *domain adaptation* problem. In some real-world situations, we may only focus on the performance of our model on a specific domain. To maintain the performance, we need labeled training data for supervised learning. However, we often cannot collect enough labeled training data relevant to the domain we are interested in (in-domain). Thus, we need to introduce more labeled data from other different domains (out-of-domain). We aim to leverage the general knowledge from out-of-domain dataset to enhance the in-domain performance of our model.

We consider a specific domain adaptation scenario in this work, where we have a few labeled in-domain training data and meanwhile we have sufficient labeled out-of-domain training data from other general domains.

Training on these two datasets jointly is a straightforward solution for this scenario, but not all samples from out-of-domain dataset has equal effect during the training procedure. Several studies (Koehn and Knowles, 2017) on neural machine translation (NMT) task show that, out-of-domain instances relevant to the in-domain data are beneficial while the instances irrelevant to the in-domain data may be even harmful to the translation quality. Apart from that, for sentiment classification task, some general expressions such as “I’m truly impressed by the design.” may appear in all domains. Taking them as training samples can help the model to learn general syntactic and semantic knowledge, which improves the cross-domain sentiment classification performance. But using examples like “This chair is solid.” (negative sentiment, furniture domain) may reduce the accuracy of classifying “This knife is solid.” (positive sentiment, kitchen domain), because “solid” has different meanings in these two domains. Any domain-specific expression like this would probably introduce some noise. So it is essential to find a suitable strategy to measure the importance of each training sample.

There are many instance weighting (or instance selection) methods to tackle this problem. They assign a weight to each instance and transform the loss function to a weighted-sum formula. Most of the conventional methods (Jiang and Zhai, 2007; Gretton et al., 2006, 2009; Axelrod et al., 2011; Wang et al., 2017; Zhang and Xiong, 2018; Wang et al., 2019; Dou et al., 2020) propose different

---

[1] The code is available at `https://github.com/CasparSwift/WIND`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2366–2376
August 1–6, 2021.

kinds of manually designed metrics to calculate the weights of instances. The core idea of these methods is to weight the instances according to their importance and similarity to the target domain. However, in our domain adaptation setting, the size of out-of-domain corpus is much larger than that of in-domain corpus. The weights learned by the previous methods may be biased to the out-of-domain data, which would unavoidably result in poorer performance on the in-domain data. In this paper, we seek to automatically learn the weights which make the model generalize well on the unbiased in-domain data.

Inspired by Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), we introduce another unbiased subset from in-domain data which serves as a query set. We propose a novel model-agnostic differentiable instance weighting approach named "**WIND**" (means **W**eighting **IN**stances **D**ifferentially) which is a general framework and can be applied to all tasks in our domain adaptation settings. Moreover, we hope to get rid of manually designed metrics and let the weights to be differentiable. To reduce the computational complexity, we adopt a second-order derivation approximation approach for calculating the gradient of weights. We conduct plenty of experiments on datasets from three representative NLP tasks: **sentiment classification**, **machine translation** and **relation extraction**. The results show that our proposed method substantially outperforms several strong baselines.

The contributions of our work can be summarized as follows:

- We propose a novel differentiable instance weighting algorithm for domain adaptation, which learns the weights of instances with gradient descent and does not need manually designed weighting metrics.

- We adopt a second-order approximation technique to speed up the model training.

- We conduct experiments on three typical NLP tasks: Sentiment Classification, Machine Translation and Relation Extraction. Experiment results demonstrate the effectiveness of the proposed method. Code will be released.

2 Methodology

In this section, we first formulate our domain adaptation problem and introduce some notations. Then we present the proposed gradient-based model-agnostic instance weighting framework for our setting and introduce the method to approximate the second-order derivation of query loss. Finally, we discuss some optimization details of our method.

2.1 Problem Formulation

Let $\mathcal{D}_{train}$, $\mathcal{D}_{dev}$ and $\mathcal{D}_{test}$ denote our train, development and test datasets respectively. We use $\mathcal{D}_{train}$ for model training, $\mathcal{D}_{dev}$ for hyperparameter tuning and $\mathcal{D}_{test}$ for model testing. Both $\mathcal{D}_{dev}$ and $\mathcal{D}_{test}$ are in-domain data. Differently, $\mathcal{D}_{train}$ consists of sufficient labeled out-of-domain training samples $\mathcal{D}_{out} = \{(x_i, y_i)\}_{i=1}^{m}$ and a few labeled in-domain training samples $\mathcal{D}_{in} = \{(x_i, y_i)\}_{i=1}^{n}$, where $n << m$.

How to efficiently utilize $\mathcal{D}_{in}$ is the key to better domain transfer. To tackle this problem, in this paper we first sample an *in-domain train* subset $\mathcal{D}_{it} = \{(x_i, y_i)\}_{i=1}^{n_1}$ from $\mathcal{D}_{in}$ and we assign a scalar weight w_i to each instance $(x_i, y_i) \in \mathcal{D}_{it} \cup \mathcal{D}_{out}$. We hope that during training, the model can find the optimal weight $\boldsymbol{w} = (w_1, ..., w_{n_1+m})$ by itself. For this purpose, the weight $\boldsymbol{w}$ should be differentiable and can be optimized by gradient descent. Moreover, we denote the deep neural network (DNN) as a function $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ which is parameterized by $\boldsymbol{\theta}$ and maps x_i from the input space to the label space. In our instance weighting setting, the training loss follows a weighted-sum formula:

$$\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}) = \frac{1}{n_1 + m} \sum_{\substack{(x_i, y_i) \in \\ \mathcal{D}_{it} \cup \mathcal{D}_{out}}} w_i \ell(f_{\boldsymbol{\theta}}(x_i), y_i) \tag{1}$$

where ℓ denotes the loss function, which can be any kind of loss such as cross entropy loss for classification tasks, or label-smoothed cross entropy loss for machine translation.

Jointly optimizing $\boldsymbol{\theta}$ and $\boldsymbol{w}$ using Eq. 1 is a straightforward solution. However, due to the data distribution discrepancy of in-domain and out-of-domain datasets, learning $\boldsymbol{w}$ directly from $\mathcal{D}_{it} \cup \mathcal{D}_{out}$ by Eq. 1 may introduce bias. What we expect is that the model trained on $\boldsymbol{w}$ can be generalized to the in-domain data. In order to achieve this goal, inspired by MAML (Finn et al., 2017), we propose to sample another subset $\mathcal{D}_q = \{(x_i, y_i)\}_{i=1}^{n_2}$ named *query set* from $\mathcal{D}_{in}$. We propose to use this query set to optimize $\boldsymbol{w}$. Specifically, we aim to obtain a weight vector $\boldsymbol{w}$

which minimizes the loss on $\mathcal{D}_q$:

$$\mathcal{L}_q(\boldsymbol{\theta}) = \frac{1}{n_2} \sum_{(x_i,y_i)\in\mathcal{D}_q} \ell(f_{\boldsymbol{\theta}}(x_i), y_i) \tag{2}$$

Note that we only weight $n_1 + m$ instances from $\mathcal{D}_{it} \cup \mathcal{D}_{out}$, so $\mathcal{L}_q(\boldsymbol{\theta})$ has a standard form, not a weighted-sum form.

Given a specific $\boldsymbol{w}$, we can train a model with the loss $\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w})$ and then get the optimized parameter $\boldsymbol{\theta}^*$. We aim to minimize the loss on the query set given $\boldsymbol{\theta}^*$. Therefore, our problem can be formulated as the following bilevel optimization problem (Colson et al., 2007):

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \mathcal{L}_q(\boldsymbol{\theta}^*) \\ \text{s.t.} \quad & \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}) \end{aligned} \tag{3}$$

This bilevel formulation arises in many meta-learning or hyperparameters optimization (HPO) problems (Bergstra et al., 2013; Franceschi et al., 2018), where the optimization of the outer objective $\mathcal{L}_q$ depends on the optimization of inner objective $\mathcal{L}_{train}$. In fact, Eq. 3 is a special case of hyper-paramter optimization, because $\boldsymbol{w}$ can be viewed as special hyperparameter of our model. In Section 2.2, we will introduce our proposed algorithm to solve this nested formulation.

2.2 Optimization of Instance Weights

It is difficult to directly solve the above-mentioned bilevel optimization problem because of its high complexity of solving the inner objective. There are many gradient-based methods (Maclaurin et al., 2015; Franceschi et al., 2018) to solve this problem. However, unlike typical hyperparameters such as learning rate, the instance weight $\boldsymbol{w}$ is of high dimension. It is even harder to optimize this problem in our setting.

Inspired by the optimization techniques used in model-agnostic meta-learning (MAML) (Finn et al., 2017), we split the training procedure of each iteration into the following three steps.

2.2.1 Pseudo Update

Firstly, we sample two mini-batches of data from $\mathcal{D}_{it} \cup \mathcal{D}_{out}$ and $\mathcal{D}_q$ respectively, Then we compute the model's parameters after one step update by the gradient of $\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w})$ respect to $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}) \tag{4}$$

where β denotes the learning rate of this step.

This step is just "pseudo update". After updating, we do not replace original parameters $\boldsymbol{\theta}$ with the adapted parameters $\widehat{\boldsymbol{\theta}}$. Instead, we store both $\boldsymbol{\theta}$ and $\widehat{\boldsymbol{\theta}}$. We will use $\widehat{\boldsymbol{\theta}}$ to calculate the gradient of $\boldsymbol{w}$ in the second step. So in our proposed algorithm, $\widehat{\boldsymbol{\theta}}$ is just an intermediate variable which will be abandoned in the end of current iteration.

2.2.2 Instance Weight Update

Then we calculate the instance weights $\boldsymbol{w}$ using $\widehat{\boldsymbol{\theta}}$. In this step, our goal is to find an optimal $\boldsymbol{w}^*$. We expect $\boldsymbol{w}^*$ to have the property that: optimizing one step by $\mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w}^*)$ should result in a decrease of query loss. In other words, we expect $\boldsymbol{w}^*$ to minimize the loss on the query set after one step update:

$$\begin{aligned} \boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}}) \\ &= \arg\min_{\boldsymbol{w}} \mathcal{L}_q(\boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \boldsymbol{w})) \end{aligned} \tag{5}$$

Note that this is an approximation for the outer objective of Eq. 3. Theorectically, we can perform gradient descent for many steps to find $\boldsymbol{w}^*$. But it is time-consuming. So basically we optimize $\boldsymbol{w}$ with the gradient of $\mathcal{L}_q(\widehat{\boldsymbol{\theta}})$ with respect to $\boldsymbol{w}$ for only one step:

$$\widehat{\boldsymbol{w}} = \boldsymbol{w} - \gamma \cdot \nabla_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}}) \tag{6}$$

where γ denotes the learning rate of $\boldsymbol{w}$.

We take $\widehat{\boldsymbol{w}}$ as an approximation of $\boldsymbol{w}^*$. Using multiple gradient updates for $\boldsymbol{w}$ is a straightforward extension of this step, which will lead to more accurate approximation for $\boldsymbol{w}^*$ while increasing the computational complexity at the same time.

2.2.3 Final Update

In the previous two steps, we have an approximately optimal weights $\widehat{\boldsymbol{w}}$. We use it for actual update for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}, \widehat{\boldsymbol{w}}) \tag{7}$$

Current iteration ends after this step. As mentioned before, $\widehat{\boldsymbol{\theta}}$ will be abandoned, but we can choose whether $\widehat{\boldsymbol{w}}$ to be abandoned or not. This will be further discussed in Section 2.4.3.

2.3 Second-Order Derivation Approximation

There is a fatal problem when calculating the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_q(\widehat{\boldsymbol{\theta}})$ in the instance weight update (Section 2.2.2). We apply the chain rule to Eq. 6:

$$\begin{aligned}\widehat{\boldsymbol{w}} &= \boldsymbol{w} - \gamma \cdot \nabla_{\boldsymbol{w}}\mathcal{L}_q(\widehat{\boldsymbol{\theta}}) \\ &= \boldsymbol{w} - \gamma \cdot \nabla_{\widehat{\boldsymbol{\theta}}}\mathcal{L}_q \cdot \nabla_{\boldsymbol{w}}\widehat{\boldsymbol{\theta}} \\ &= \boldsymbol{w} + \beta\gamma \cdot \nabla_{\widehat{\boldsymbol{\theta}}}\mathcal{L}_q \cdot \nabla^2_{\boldsymbol{\theta},\boldsymbol{w}}\mathcal{L}_{train}\end{aligned} \tag{8}$$

We use $|\boldsymbol{\theta}|, |\boldsymbol{w}|$ to denote the dimensions of $\boldsymbol{\theta}, \boldsymbol{w}$ respectively. The second-order derivation $\nabla^2_{\boldsymbol{\theta},\boldsymbol{w}}\mathcal{L}_{train}$ is a $|\boldsymbol{\theta}| \times |\boldsymbol{w}|$ matrix which is too huge to calculate and store. Apart from that, calculating the matrix-vector product is also expensive. Precisely calculating the results is unrealistic. Fortunately, we can adopt the approximation technique used in DARTS (Liu et al., 2018) to solve this problem. This technique uses the finite difference approximation:

$$\begin{aligned}&\nabla_{\widehat{\boldsymbol{\theta}}}\mathcal{L}_q \cdot \nabla^2_{\boldsymbol{\theta},\boldsymbol{w}}\mathcal{L}_{train} \approx \\ &\frac{\nabla_{\boldsymbol{w}}\mathcal{L}_{train}(\boldsymbol{\theta}^+, \boldsymbol{w}) - \nabla_{\boldsymbol{w}}\mathcal{L}_{train}(\boldsymbol{\theta}^-, \boldsymbol{w})}{2\epsilon}\end{aligned} \tag{9}$$

$$\begin{aligned}\boldsymbol{\theta}^+ &= \boldsymbol{\theta} + \epsilon\nabla_{\widehat{\boldsymbol{\theta}}}\mathcal{L}_q \\ \boldsymbol{\theta}^- &= \boldsymbol{\theta} - \epsilon\nabla_{\widehat{\boldsymbol{\theta}}}\mathcal{L}_q\end{aligned} \tag{10}$$

where ϵ is a small scalar. We follow Liu et al. (2018) to set $\epsilon = 0.01/\|\nabla_{\widehat{\boldsymbol{\theta}}}\mathcal{L}_q\|_2$ which is accurate enough for approximation. Let $\alpha = \beta\gamma$, we can adjust the learning rate of $\boldsymbol{w}$ by tuning α.

Calculating this approximated gradient needs only another two forward passes for $\boldsymbol{\theta}^+$ and $\boldsymbol{\theta}^-$, which greatly accelerates the training procedure. More details about the training process are described in Algorithm 1.

2.4 Optimization Details

2.4.1 Dataset Split Strategy

The data split of query set $\mathcal{D}_q$ is critical. As mentioned in Section 2.1, we randomly sample $\mathcal{D}_{it}$ and $\mathcal{D}_q$ from in-domain training set $\mathcal{D}_{in}$. If we have enough in-domain data, $\mathcal{D}_{it}$ and $\mathcal{D}_q$ should be disjoint. However, our in-domain training set is not so large, and splitting it will make it even smaller. As a result, we use $\mathcal{D}_{in} = \mathcal{D}_q = \mathcal{D}_{it}$ instead of sampling. The ablation studies about this issue are shown in Section 3.5.

2.4.2 Scaling the Weights

In this work, an extreme value of w_i may make the training unstable. It is important to scale it to an appropriate range. In practice, we use sigmoid function to normalize it into $[0, 1]$:

$$\mathcal{L}_{train} = \frac{1}{n_1 + m} \sum_{\substack{(x_i, y_i) \in \\ \mathcal{D}_{it} \cup \mathcal{D}_{out}}} \sigma(w_i)\ell(f_{\boldsymbol{\theta}}(x_i), y_i) \tag{11}$$

Algorithm 1 **WIND** (**W**eighting **IN**stances **D**ifferentially)

```
Input: In-domain training set $\mathcal{D}_{it}$, out-of-domain set $\mathcal{D}_{out}$, query set $\mathcal{D}_q$, model parameter $\boldsymbol{\theta}$
Initialize the weights $\boldsymbol{w}$ (see Section 2.4.3)
for $i = 1$ to $epochs$ do
    for $j = 1$ to $steps\_per\_epoch$ do
        Get a mini-batch $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{D}_{it} \cup \mathcal{D}_{out}$
        Get a mini-batch $(\boldsymbol{x}_v, \boldsymbol{y}_v) \in \mathcal{D}_q$
        for $k = 1$ to $inner\_steps$ do
            Obtain $\widehat{\boldsymbol{\theta}}$ by Eq. 4
            Obtain $\widehat{\boldsymbol{w}}$ by Eq. 6 and Eq. 9
            Update $\boldsymbol{w}$ in the storage by $\widehat{\boldsymbol{w}}$
        end for
        Update $\boldsymbol{\theta}$ by Eq. 7
    end for
end for
```

2.4.3 Initialization of Instance Weight

How to initialize $\boldsymbol{w}$ is an important issue. In this paper, we assume that all the training samples from in-domain training set $\mathcal{D}_{it}$ are beneficial and should be highly weighted. For samples in D_{it}, we fix their weights to a very large number at the beginning of training, which is close to 1 after calculating by the sigmoid function. For samples in D_{out}, we initialize their weights all by zeros. During training, we do not optimize the weights of the in-domain training samples and only update the weights of the out-of-domain training samples.

Moreover, when to initialize $\boldsymbol{w}$ is another important issue. We propose two different kinds of initialization strategy. One is to initialize $\boldsymbol{w}$ at the beginning of each iteration. Another alternative is to initialize $\boldsymbol{w}$ at the beginning of the training, and update $\boldsymbol{w}$ in the storage every iteration. In practice, we choose the latter. Although the former is more easy to implement, it cannot make use of $\widehat{\boldsymbol{w}}$ from previous iterations.

3 Experiments

To evaluate the effectiveness of our proposed method introduced in Section 2 and demonstrate its model-agnostic property, we apply our method to

Dataset	Sentences
B-train ($\mathcal{D}_{in}$)	500
D,E,K ($\mathcal{D}_{out}$)	18,000
B-dev ($\mathcal{D}_{dev}$)	1,500
B-test ($\mathcal{D}_{test}$)	4,000

Table 1: Statistics of sentiment classification setting with *books* domain as the in-domain.

Dataset	Sentences
TED training ($\mathcal{D}_{in}$)	202,356
WMT14 subset ($\mathcal{D}_{out}$)	500,000
TED tst2012 ($\mathcal{D}_{dev}$)	1,700
TED tst2013 ($\mathcal{D}_{test}$)	993
TED tst2014 ($\mathcal{D}_{test}$)	1,305

Table 2: Statistics of machine translation setting.

Dataset	Docs	Relations
bc-train ($\mathcal{D}_{in}$)	10	222
nw & bn ($\mathcal{D}_{out}$)	332	4,695
bc-dev ($\mathcal{D}_{dev}$)	10	347
bc-test ($\mathcal{D}_{test}$)	40	1,036

Table 3: Statistics of relation extraction setting.

three different dataset settings of three tasks: Sentiment Classification, Machine Translation (MT) and Relation Extraction, respectively.

3.1 Datasets

For sentiment classification task, we conduct the experiments on the widely-used Amazon Review Dataset (Blitzer et al., 2007). This dataset contains four domain: *books* (B), *dvd* (D), *electronics* (E) and *kitchen* (K). Each domain contains the reviews of a specific category of products. We use the data processed by He et al. (2018) and collect 6000 labeled samples for each domain. We split the data of each domain into training ($\mathcal{D}_{in}$), development ($\mathcal{D}_{dev}$) and test ($\mathcal{D}_{test}$) set. In each domain adaptation setting, we choose the training data of one domain as the in-domain data ($\mathcal{D}_{in}$) and all data of other three domains as the out-of-domain data ($\mathcal{D}_{out}$). Table 1 shows an example with *books* domain as the in-domain.

For machine translation task, similar to the settings of Luong and Manning (2015); Wang et al. (2017); Zeng et al. (2019), we use the IWSLT 2016 English (EN) to German (DE) corpus (Cettolo et al., 2016) as the in-domain data. This corpus contains about 202K sentences from TED talks. For out-of-domain data, we randomly sample a subset of 500K sentences from the WMT 2014 English-German corpus. Table 2 show the statistics of the datasets.

For relation extraction task, we evaluate our method on the ACE 2005 dataset. This dataset is suitable for evaluating domain adaptation because it contains six different domains. It has been adopted by many previous works (Nguyen and Grishman, 2014; Gormley et al., 2015; Fu et al., 2017) for cross-domain relation extraction. In this work, we take *broadcast news* (bn) and *newswire* (nw) domain as out-of-domain, and split *broadcast conversation* (bc) domain into train/dev/test sets with the ratio of $1 : 1 : 4$. Table 3 shows the detailed statistics.

3.2 Implementation Details

For sentiment classification task, we use the pretrained BERT-base-uncased (Devlin et al., 2018) model provided by HuggingFace (Wolf et al., 2019) as our feature extractor. Our sentiment classifier is a one-hidden-layer MLP with ReLU as the activation function. For the optimization of model parameters $\boldsymbol{\theta}$, we use the AdamW (Loshchilov and Hutter, 2018) as the optimizer with a learning rate of $2e-5$, a warmup of 0.1 (of the total steps) and a linearly decayed learning rate scheduler. The computational cost is about 8-12 GPU hours on Tesla V100.

For machine translation, we choose a vanilla Transformer (Vaswani et al., 2017) as our backbone. We implement some baseline methods and our method via fairseq toolkit (Ott et al., 2019). We use MOSES[2] scripts to tokenize the English and German sentences, and then we apply Byte Pair Encoding (BPE) (Sennrich et al., 2015) algorithm to split the words into subwords. We limit the maximum length of the sentences to 250 subwords. We choose to share the embeddings of English and German with the vocabulary size of 32,000. We use Adam (Kingma and Ba, 2014) as the optimizer and a decayed learning rate of $7e-4$.

For relation extraction, we only focus on relation classification when the entity pairs are given for simplicity. We use the RBERT (Wu and He, 2019) model as our backbone. The configurations of optimizer and learning rate are the same as those in our sentiment classification experiments.

3.3 Baselines

We implemented the following baseline methods for comparison with our methods. It's worth noting that we don't choose some baselines (Jiang and Zhai, 2007; Wang et al., 2017) of instance weight-

---

[2] `https://github.com/moses-smt/mosesdecoder`

Method	**B**	**D**	**E**	**K**	**Avg.**
In	87.50	86.75	89.38	89.50	88.28
Out	90.17	90.17	92.58	92.95	91.47
In+Out	91.20	90.78	92.58	93.12	91.92
Ensemble (Wang et al., 2017)	90.55	90.00	92.83	92.83	91.55
IW-Fit (Wang et al., 2019)	90.40	90.00	92.85	92.62	91.47
DANN (Ganin et al., 2016)	91.15	90.88	**93.35**	93.35	92.18
WIND (ours)	**91.65**	**91.08**	93.10	**94.00**	**92.46**

Table 4: Experiment results (Accuracy) of domain adaptation for sentiment classification.

Method	tst2012	tst2013	tst2014
In	-	29.47	25.18
Out	21.45	22.50	19.63
In+Out	29.60	32.50	28.72
DM (Britz et al., 2017)	-	31.57	27.60
IDDA (Zeng et al., 2019)	-	32.93	28.88
WIND (ours)	**30.77**	**33.58**	**29.26**

Table 5: Experiment results (BLEU) of domain adaptation for machine translation.

Method	test set
In	68.05
Out	88.22
In+Out	89.58
DANN (Fu et al., 2017)	89.38
WIND (ours)	**90.54**

Table 6: Experiment results (Accuracy) of domain adaptation for relation extraction.

ing because they are quite early work and it's unfair to compare with them.

For sentiment classification:

- **In** A pre-trained BERT only fine-tuned on the in-domain training set.
- **Out** A pre-trained BERT only fine-tuned on the out-of-domain training set.
- **In+Out** A pre-trained BERT fine-tuned on both in-domain and out-of-domain data.
- **Ensemble** It ensembles the **in** model and the **out** model by adding their predictions. Note that this method is used as a baseline in Wang et al. (2017). Although Wang et al. (2017) conducted the experiments on machine translation, we can still adopt this method on sentiment classification task.
- **IW-Fit** It uses the weighting strategy proposed by Wang et al. (2019) for domain transfer.
- **DANN** It introduces the domain classifier and adversarial training as proposed by Ganin et al. (2016).

For machine translation, the meanings of **In**, **Out** and **In+Out** is the same as those in the sentiment classification setting. There are some other baselines for machine translation setting:

- **DM** This indicates the Discriminative Mixing method proposed by Britz et al. (2017), which adds a domain classifier to the encodings of source sentences similar to DANN (Ganin et al., 2016).
- **IDDA** This indicates Iterative Dual Domain Adaptation methods proposed by Zeng et al. (2019), which iteratively performs bidirectional translation knowledge transfer using knowledge distillation between in-domain and out-of-domain. Note that this method focuses on the performance of both domains but in this paper we only focus on in-domain performance.

For relation extraction, besides the **In**, **Out** and **In+Out** approaches, we also choose Fu et al. (2017) as our baseline. This method simply introduces DANN (Ganin et al., 2016) to cross-domain relation extraction. Note that it is implemented by convolutional neural network, we reimplement a RBERT (Wu and He, 2019) version of it.

3.4 Experiment Results

Table 4 shows the overall performance of our methods in the domain adaptation setting on the sentiment classification task. Our method achieves an absolute improvement of 0.45, 0.40, 0.52 and 0.88 points on four settings respectively in comparison to the **In+Out** baseline. Moreover, our method outperforms all the domain adaptation methods on the settings with B, D, K as the in-domain data except the E domain. Although our method does not beat all baselines on all settings, it achieves the best average performance across the four settings. On average, we achieve an improvement of 0.28 point over DANN and 0.54 point over **In+Out**.

Table 5 shows the performance for the machine translation task. We use BLEU (Papineni et al., 2002) scores to measure the performance. Our method beats all baselines on all test sets (tst2013,tst2014) and the development set (tst2012). On these three datasets, we observe an improvement of 1.17, 1.08 and 0.54 BLEU points compared to **In+Out**. On tst2013 and tst2014 test sets, we also achieve an improvement of 0.65 and 0.38 BLEU points compared to IDDA (Zeng et al., 2019) method. Table 6 further shows our method's effectiveness on the relation extraction task.

Furthermore, from the results in Tables 4, 5 and 6, we can make the following observations:

(1) In comparison to the method in (Wang et al., 2019) which uses manually designed weighting metrics, our differentiable weighting approach outperforms it in the sentiment classification task. This result demonstrates that designing the metrics manually may not be the best solution for all the tasks. Designing them requires many prior human expert knowledge which is hard to generalize well across tasks. By contrast, our method can learn instance weights with the help of meta-learning based algorithm to improve the models' in-domain generalization capability.

(2) Domain adversarial based method is a strong baseline which is surpassed only by our method in the sentiment classification task and the relation extraction task. However, it performs not so well for the machine translation task. The potential reason may be that Britz et al. (2017) introduces the domain classifier after the encoder to learn domain-invariant features of sentences from source language, but both domains share the same decoder which cannot discriminate the features encoded by the encoder. In other words, this type of method may only pay attention to the encoder and ignore the domain transfer of decoder. In contrast, our method overcomes this problem by considering weighting the loss of the whole model and thus achieves better performance.

(3) For all three tasks, adding out-of-domain corpus to the training set will improve the overall performance. We believe that adding the data of some general domains can help the model better learn domain-invariant syntactic and semantic knowledge, so it can improve the performance on the in-domain data. This is consistent with the conclusions reached by transfer learning. Interestingly, this result is contradictory to the observation of Wang et al. (2017), whose experiment results show that adding out-of-domain to in-domain data degraded machine translation performance. We suspect that there is a problem with their training strategy. The hyperparameters required under each setting may be different. Some hyperparameters that are set incorrectly (e.g. the same as **In**) may make the result of **In+Out** even worse. Another reason may be that the RNN-based sequence-to-sequence NMT system they used tends to be more sensitive to the noise while the Transformer (Vaswani et al., 2017) model we used is more robust.

All in all, as we expected, our proposed method WIND achieves the best performance under the three task settings. This illustrates the advantages of using differentiable method for data weighting.

Method	**B**	**D**	**E**	**K**
WIND+split-init	91.10	90.55	92.68	93.60
WIND+split	91.50	91.12	93.42	94.00
WIND+rand	91.17	91.03	93.33	93.95
WIND	**92.12**	**91.28**	**93.65**	**94.15**

Table 7: The comparison between different variants of our method on the sentiment classification task. Note that the results in this table are evaluated under $|\mathcal{D}_{in}| = 1,000$ setting. "+rand" means randomly initializing $\boldsymbol{w}$. "+split" means splitting $\mathcal{D}_{in}$ into disjoint $\mathcal{D}_{it}$ and $\mathcal{D}_q$. "-init" means not assigning a large number to in-domain data weights.

3.5 Ablation Study

In this part, we study the effect of the strategies mentioned in Section 2.4.1 and Section 2.4.3. The experiment results shown in Table 7 demonstrate that:

(1) Assigning the weights of in-domain instances to a large number ($1e8$) and fixing them during training can improve the accuracy. The weights of this part do not actually need to be learned. Fixing them may reduce the interference to the learning process from the out-of-domain data.

(2) Zero initialization for weights of out-of-domain instances is better than the random initialization. The underlying reason for this may be that random initialization may easily make the model stuck into a local minima.

(3) No splitting for $\mathcal{D}_{in}$ can improve the performance as well. Intuitively, this improvement comes

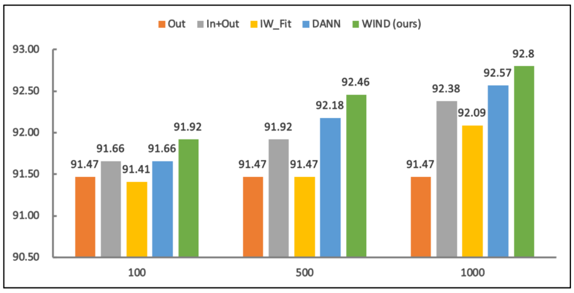

Figure 1: The effect of in-domain dataset size on the sentiment classification task.

from the increased size of in-domain training set, which enables us to make more use of the scarce in-domain training samples.

3.6 Effect of In-Domain Dataset Size

In this part, we aim to study the impact of in-domain dataset size n. Besides $n = 500$ setting, we sample another two different $\mathcal{D}_{in}$ with $n = 100$ and $n = 1000$. The rest of in-domain data is used as development set. We evaluate all three settings on the same $\mathcal{D}_{test}$ mentioned in Table 1.

Figure 1 shows the average accuracy over four domain settings when using different domain adaptation methods. We found that DANN (Ganin et al., 2016) may not perform so well when in-domain data are scarce. But our method can still achieve consistent improvements in this three dataset size settings.

4 Related Work

4.1 Domain Adaptation

Domain Adaptation is a fundamental problem in machine learning and NLP. We aim to train a well-performing model on a source domain which can be generalized to a target domain.

The basic idea for domain adaptation is to learn domain-invariant representations which generalize across the domains. To achieve this, the most prevailing method Domain Adversarial Neural Network (DANN) (Ganin et al., 2016; Qu et al., 2019; Xue et al., 2020) introduces a domain classifier and uses adversarial training to make the features unable to discriminate between source and target domains. This method has been applied to many NLP tasks. However, out-of-domain data is far more than in-domain data in our setting. DANN may cause some bias in this unbalanced dataset. Another type of methods (Fang and Xie, 2020; Li et al., 2020) propose to learn domain-general representations by contrastive learning (Chen et al., 2020a; He et al., 2019; Chen et al., 2020b). But they mainly focus on classification task and the methods are not model-agnostic frameworks.

4.2 Cross-Domain Sentiment Classification

Sentiment classification task aims to automatically classify the sentiment polarity of the given texts. Cross-domain sentiment classification aims to generalize the sentiment classifier from source domain to target domain.

Besides the domain adaptation methods introduced in Section 4.1, there are some methods which are specific for cross-domain sentiment classification. An important line of works follow the Structural Correspondence Learning (SCL) (Blitzer et al., 2006), and they design an auxiliary task called pivot prediction to transfer domain-invariant knowledge (Pan et al., 2010; Yu and Jiang, 2016; Ziser and Reichart, 2016, 2018, 2019). But the pivot words need human knowledge to select, which may be not so accurate. Recently, the pre-trained language models such as BERT (Devlin et al., 2018) have achieved state-of-the-art on many NLP tasks. DAAT (Du et al., 2020) performs a novel post-training procedure on BERT and uses adversarial training to transfer domain knowledge. But this method only works for classification task while our method is model-agnostic and does not need two-stage post-training and fine-tuning.

4.3 Meta-Learning

The goal of meta-learning is to train a model that can adapt to a new task quickly given a few new samples. In other words, meta-learning can learn the initial value of the model that is close to the optimums of many different tasks. MAML (Finn et al., 2017) is a classical method for meta-learning. Each entry of the meta-training set of MAML is a subset contains training data (support set) and test data (query set). MAML calculates the loss on the query set based on the parameters after one-step optimization on support set, and uses the gradient of this loss to update the model parameters. MAML has also been adopt for natural language understanding task before (Dou et al., 2019). Despite our domain adaptation setting is quite different from that in MAML, we can still utilize the idea of their work to help domain generation.

5 Conclusion

In this paper, we propose WIND, a differentiable instance weighting method for model-agnostic domain adaptation, which is inspired by the ideas of meta-learning to learn the weights on the in-domain query set. Experiment results on three typical NLP tasks show the efficacy of our framework.

It remains an open question how to efficiently transfer the domain knowledge. In the future, we plan to evaluate our method on more different tasks.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Xiaojun Wan is the corresponding author. Thank the reviewers for their constructive suggestions.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362.

James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128.

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126.

Mauro Cettolo, Niehues Jan, Stüker Sebastian, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2016. The iwslt 2016 evaluation campaign. In *International Workshop on Spoken Language Translation*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020a. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020b. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zi-Yi Dou, Antonios Anastasopoulos, and Graham Neubig. 2020. Dynamic data selection and weighting for iterative back-translation. *arXiv preprint arXiv:2004.03672*.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *arXiv preprint arXiv:1908.10423*.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, and Jianxin Liao. 2020. Adversarial and domain-aware bert for cross-domain sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4019–4028.

Hongchao Fang and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–429.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. 2006. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19:513–520.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. 2009. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. *arXiv preprint arXiv:1809.00530*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Tian Li, Xiang Chen, Shanghang Zhang, Zhen Dong, and Kurt Keutzer. 2020. Cross-domain sentiment classification with contrastive learning and mutual information maximization. *arXiv preprint arXiv:2010.16088*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 68–74.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Xiaoye Qu, Zhikang Zou, Yu Cheng, Yang Yang, and Pan Zhou. 2019. Adversarial category alignment network for cross-domain sentiment classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2496–2508.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488.

Zhi Wang, Wei Bi, Yan Wang, and Xiaojiang Liu. 2019. Better fine-tuning via instance weighting for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7241–7248.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364.

Qianming Xue, Wei Zhang, and Hongyuan Zha. 2020. Improving domain-adapted sentiment classification by deep adversarial mutual learning. In *Proceedings* *of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9362–9369.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jiali Zeng, Yang Liu, Jinsong Su, Yubin Ge, Yaojie Lu, Yongjing Yin, and Jiebo Luo. 2019. Iterative dual domain adaptation for neural machine translation. *arXiv preprint arXiv:1912.07239*.

Shiqi Zhang and Deyi Xiong. 2018. Sentence weighting for neural machine translation domain adaptation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3181–3190.

Yftah Ziser and Roi Reichart. 2016. Neural structural correspondence learning for domain adaptation. *arXiv preprint arXiv:1610.01588*.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Yftah Ziser and Roi Reichart. 2019. Task refinement learning for improved accuracy and stability of unsupervised domain adaptation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5895–5906.

DocOIE: A Document-level Context-Aware Dataset for OpenIE

Kuicai Dong[1], **Yilin Zhao**[1], **Aixin Sun**[1], **Jung-Jae Kim**[2], **Xiaoli Li**[1,2]
[1] School of Computer Science and Engineering, Nanyang Technological University, Singapore
`{kuicai001, zhao0320}@e.ntu.edu.sg, {axsun, xlli}@ntu.edu.sg`
[2] Institute for Infocomm Research, A*STAR, Singapore
`{jjkim, xlli}@i2r.a-star.edu.sg`

Abstract

Open Information Extraction (OpenIE) aims to extract structured relational tuples (subject, relation, object) from sentences, and plays a critical role in many NLP applications. Existing solutions perform extraction at *sentence level*, without referring to any additional contextual information. In reality, however, a sentence typically exists as part of a document rather than standalone; we often need to access relevant contextual information around the sentence before we can accurately interpret it. As there is no document-level context-aware OpenIE dataset available, we manually annotate 800 sentences from 80 documents in two domains (Healthcare and Transportation) to form a DocOIE dataset for evaluation. In addition, we propose DocIE, a document-level context-aware OpenIE model. Our experimental results demonstrate that incorporating document-level context is helpful in improving OpenIE performance. Both the DocOIE dataset and DocIE model are available online.[1]

1 Introduction

Open Information Extraction has been a critical NLP task as it can extract structured relational tuples (*subject, relation, object*) from unstructured text. The OpenIE system is fully domain-independent, and does not need input from users. It is also highly scalable and allows fast querying mechanism (Yates et al., 2007). Therefore, OpenIE has been successfully applied to a variety of downstream NLP tasks, such as knowledge base population (Martínez-Rodríguez et al., 2018; Gashteovski et al., 2020), question answering (Khot et al., 2017), and summarization (Fan et al., 2019).

Current OpenIE methods mainly focus on extracting tuples *at sentence level*. However, in many NLP scenarios, sentences exist as part of a document rather than standalone. Given a document corpus, if we simply apply existing sentence-level OpenIE models to extract tuples, we could miss some useful and critical document-level contextual information, leading to unsatisfying results. We use the two example sentences in Fig. 1 to illustrate two types of ambiguities.

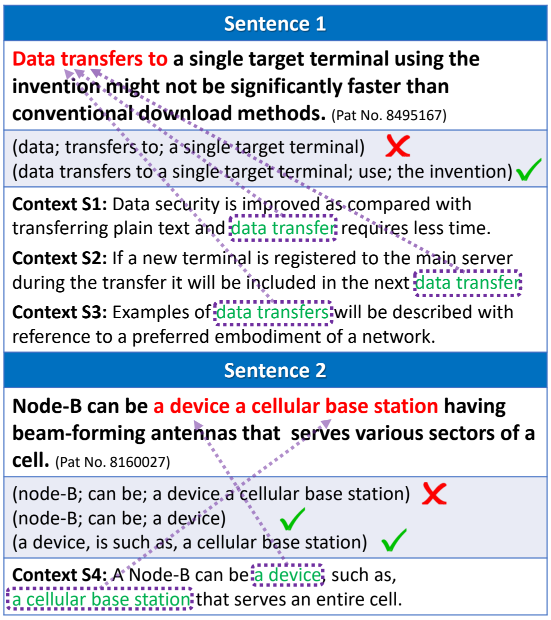

Sentence 1

Data transfers to a single target terminal using the invention might not be significantly faster than conventional download methods. (Pat No. 8495167)

(data; transfers to; a single target terminal) ✗
(data transfers to a single target terminal; use; the invention) ✓

Context S1: Data security is improved as compared with transferring plain text and data transfer requires less time.

Context S2: If a new terminal is registered to the main server during the transfer it will be included in the next data transfer.

Context S3: Examples of data transfers will be described with reference to a preferred embodiment of a network.

Sentence 2

Node-B can be a device a cellular base station having beam-forming antennas that serves various sectors of a cell. (Pat No. 8160027)

(node-B; can be; a device a cellular base station) ✗
(node-B; can be; a device) ✓
(a device, is such as, a cellular base station) ✓

Context S4: A Node-B can be a device, such as, a cellular base station that serves an entire cell.

Figure 1: Example sentences with ambiguity.

Part-of-speech Ambiguity. The word "transfers" can be a verb or a noun. Accordingly, two tuples could be extracted from the first example sentence, listed in Fig. 1. This ambiguity can be resolved by the main verb of the sentence "might not be", which is far away from "transfers", and thus is not considered by many existing OpenIE systems. However, context sentences S1, S2, and S3 in the document suggest that "data transfers" shall be considered as a noun phrase throughout this document.

Syntactic Ambiguity. The second example sentence does not have an explicit clue about the re-

---

[1] https://github.com/daviddongkc/DocOIE

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2377–2389
August 1–6, 2021.

lationship between “a device” and “a cellular base station”.[2] Thus, existing OpenIE systems often fail to split them, but incorrectly extract the first tuple (node-B; can be; a device a cellular base station). However, context sentence S4 includes an explicit cue to the relationship between the two terms and may thus help split them.

To minimize the aforementioned ambiguities, it is clear that we should leverage document-level context. However, all existing OpenIE datasets are generated or annotated at sentence level. These datasets include standalone sentences but not their context sentences. Hence they are not suitable for evaluating context-aware tuple extraction.

We annotate a **Doc**ument-level context-aware **O**pen **I**nformation **E**xtraction (**DocOIE**) dataset. DocOIE consists of 800 expert-annotated sentences from 80 documents, where 10 sentences are randomly sampled for annotation from each of the 80 documents. To the best of our knowledge, among all OpenIE datasets as of now, DocOIE contains the largest number of *expert-annotated* sentences.[3] More importantly, DocOIE provides document-level contexts, enabling OpenIE models to take relevant contexts for accurate tuple extraction.

Furthermore, to show that document-level context is useful for OpenIE task, we develop the **Doc**ument-level context-aware Open **I**nformation **E**xtraction (**DocIE**) model. DocIE encodes a source sentence with its contextual information by using pre-trained BERT (Devlin et al., 2019). Because contextual sentences can be much longer than the source sentence, the syntactic/semantic information in source sentence might be dominated by that of the contexts. Our proposed DocIE model differentiates the source sentence and its contexts by segment tags, and adding additional transformer encoder layers only for the source sentence.

In summary, our contributions are threefold:

- We propose a new task in OpenIE to extract relational tuples with document-level contexts.
- We introduce **DocOIE**, an expert-annotated dataset for evaluating document-level OpenIE systems. DocOIE consists of 2,122 relational tuples from 800 annotated sentences, with their document-level contexts.
- We present **DocIE**, a neural OpenIE system that can leverage document-level contexts for relational tuple extraction.

Dataset	#Sent.	Source	Annotation
OIE2016	3,200	QA-SRL	Automatic
Wire57	57	Wikipedia Newswire	Expert
CaRB	1,282	OIE2016	Crowdsourcing
CaRB	50	OIE2016	Expert
DocOIE	800	Patent	Expert

Table 1: Existing OpenIE datasets with number of sentences, sentence source, and annotation type.

2 Related Work

OpenIE Datasets. Since introduction of OpenIE task by Yates et al. (2007), the earlier systems have been mainly evaluated by using a small number of sentences, without a standardized evaluation procedure (Niklaus et al., 2018). OIE2016 (Stanovsky and Dagan, 2016) is the first large-scale dataset constructed for OpenIE tasks and comes with a standard scoring framework. In OIE2016, the gold tuples are automatically generated from a QA-SRL dataset (He et al., 2015) according to human crafted rules. Wire57 (Lechelle et al., 2019) improves the scorer and manually annotates 57 sentences as a benchmark dataset. Considering that OIE2016 dataset is noisy, Bhardwaj et al. (2019) provide a crowdsourcing dataset named CaRB. CaRB also has 50 expert-annotated sentences and a sophisticated scoring framework.

As summarized in Table 1, the number of expert-annotated sentences in these datasets remains small. Furthermore, the sentences in these datasets do not come with contextual information. In contrast, our DocOIE dataset consists of 800 expert-annotated sentences, and comes with the source documents for accurate sentence interpretation.

OpenIE Models. TextRunner (Yates et al., 2007) is the first OpenIE system, followed by Reverb (Fader et al., 2011), OLLIE (Mausam et al., 2012), Clausie (Corro and Gemulla, 2013), CSD-IE (Bast and Haussmann, 2013), Stanford OpenIE (Angeli et al., 2015), Openie4 (Mausam, 2016), Openie5[4], NESTIE (Bhutani et al., 2016), MINIE (Gashteovski et al., 2017) and

---

[2] The missing of a comma could be a typo in the patent document. Nevertheless, not all input sentences to OpenIE system are typo free in real applications.

[3] Expert annotation means annotations are made by a person who is familiar with the OpenIE task and OpenIE models. This term is used to distinguish annotations by crowdsourcing.

[4] `github:dair-iitd/openie-standalone`

Graphene (Cetto et al., 2018). We consider them as *traditional* OpenIE models. These models extract relational tuples based on handcrafted rules or statistical methods. They usually rely on prior syntactic or semantic analysis. Consequently, any error accumulated in the prior stages deteriorates model performance.

Recently, neural OpenIE systems have been developed and showed promising results (Cui et al., 2018; Zhan and Zhao, 2020; Kolluru et al., 2020a,b). Different from the traditional models, neural OpenIE models extract tuples in an end-to-end manner, not requiring prior syntactic or semantic analysis. In principle, the traditional rule-based or statistical OpenIE models do not need training. However, neural OpenIE models need a large number of training samples to learn the extraction patterns. For instance, IMOJIE (Kolluru et al., 2020b) uses about 100,000 sentences for model training. It is unrealistic and expensive to manually annotate 100,000 sentences simply for training purpose. Therefore, a common practice in learning a neural OpenIE model is to use tuples automatically extracted by traditional systems as training data, *i.e., a bootstrapping strategy*. We consider these imperfect training labels generated via bootstrapping as pseudo labels. The pseudo labels used in (Cui et al., 2018) are by Openie4, and those in (Kolluru et al., 2020b) are from multiple OpenIE systems.

To the best of our knowledge, no OpenIE models consider document-level contexts in the tuple extraction. Nonetheless, our neural model unavoidably requires extractions of pseudo labels bootstrapped from traditional models for training. To ensure reproducibility, as part of the DocOIE dataset, we also release the document IDs that are used for generating the pseudo labels.

3 DocOIE Dataset

We now present our **Doc**ument-level context-aware **O**pen **I**nformation **E**xtraction (**DocOIE**) dataset. We first introduce the data selection and collection process, and then the annotation process by two experts. Moreover, we explain our annotation consistency measurement to indicate the high-level annotation consistency in DocOIE. In summary, DocOIE consists of two datasets: *evaluation dataset* and *training dataset*.

Evaluation dataset contains 800 expert-annotated sentences, sampled from 80 documents in two domains (healthcare and transportation). Specifically, 10 sentences are sampled for annotation from each of the 40 documents in one domain. In total, $2,122$ relational tuples are annotated in the 800 sampled sentences (refer to Table 2 for detailed statistics, and Table 5 for fine-grained analysis).

Item	#Item	Metric	Average	Min~Max
Doc	80	N_{sent}	129.85	62~218
Sent	800	L_{sent}	22.70	5~47
		N_{tuple}	2.65	1~8
Tuple	2,122	L_{sub}	3.70	1~17
		L_{rel}	3.39	1~12
		L_{obj}	3.94	0~27

Table 2: Statistics of DocOIE evaluation dataset. $N_{\{\cdot\}}$ denotes the number of units (*i.e.,* sentence or tuple); $L_{\{\cdot\}}$ denotes the length (number of words) of the unit.

Training dataset contains 2,400 documents from the two domains (healthcare and transportation); 1,200 documents in each domain. All sentences from these documents are used to bootstrap pseudo labels for neural model training.[5]

3.1 Dataset Collection

OpenIE, by definition, is to extract relational tuples in open domain. Ideally, sentences/documents in DocOIE dataset shall not be restricted to any particular document type or topical domain. However, it is challenging to include all types of documents and annotate them. In fact, all existing annotations are restricted to specific types of documents like news and Wikipedia articles (Niklaus et al., 2018).

Document Type Selection In building DocOIE, we focus on formally written documents and leave it for future work to explore other kinds of documents. We select the type of formal documents with four criteria: (i) **Adequacy:** as a document-level context-aware dataset, each document shall have a reasonable number of sentences to provide sufficient context. (ii) **Consistency:** each document shall focus on a central topic. In such a way, sentences within the same document are correlated to and consistent with each other, which helps derive proper context. (iii) **Informativeness:** a document is considered informative if it contains informative entities like technical concepts, relations,

---

[5] Only document IDs are included in DocOIE, for document collection at `http://patft.uspto.gov/`

and events. Intuitively, OpenIE models are more useful for extracting factual tuples in informative documents. (iv) **Syntactic Variety:** sentences in these documents shall vary in syntactic structures. Such variety facilitates thorough evaluation of OpenIE models under different scenarios.

Patent Document Collection After taking all factors into consideration, we choose to collect patent documents from PatFT.[6] Each patent document elaborates one specific invention in reasonable length, providing sufficient contexts to annotators. They are rich in informativeness by nature, and the documents contain rich syntactic structures.

Through PatFT search engine, patent documents can be retrieved by keywords. We have two considerations for keyword selection: (i) **Magnitude:** as part of DocOIE, a large number of documents shall be available for training neural OpenIE models. Hence, the keywords shall lead to sufficient patent documents. (ii) **Diversity:** the collected patent documents are expected to be diversified in inventors, organizations, filed date, etc., to avoid fixed patterns, hence to ensure diversity of our dataset.

As the result, we choose three broad and non-technical keywords: "healthcare", "traffic", and "transportation", to collect documents in two broad domains, *healthcare* and *transportation*. Reported in Table 3, 42,514 and 32,256 documents are collected in healthcare and transportation respectively. Documents in each domain are contributed by more than 40,000 inventors from over 8,000 cities, and the filed dates range in several decades.

Patents	Healthcare	Transportation
#Document	42,514	32,256
#Inventor	74,266	42,286
#Organization	10,888	7,919
#City	10,493	8,095
Filed year	1999~2020	1970~2020

Table 3: Properties of collected patents in healthcare and transportation domains.

We clean these documents by removing non-textual components in them. Then, by length (in number of words) the shortest 10% and longest 10% documents are removed to avoid extremely short/long documents in our dataset. The remaining documents form the corpus from which we sample (i) documents for annotation, and (ii) training documents for bootstrapping pseudo labels.

3.2 DocOIE Evaluation Dataset Selection

To ensure annotation quality and consistency, we choose to follow expert annotation scheme instead of crowdsourcing adopted in CaRB (Bhardwaj et al., 2019). As we discussed in Section 1, sentences exist as part of a document rather than standalone. To gain an accurate interpretation of a sentence, the annotator needs to read a few surrounding sentences, or even the entire document, for relevant contexts. Hence, the choices of labelling one sentence or multiple sentences per document incur different costs.

To be able to cover a reasonable number of documents and also balance the annotation workload, we choose to randomly sample 10 sentences per document from 80 documents for annotation. Recall that the average number of sentences per document is 101.78 (refer to Table 2). The 10 sentences annotated in a document can be used to evaluate context-aware OpenIE at 10 different positions in this document. In this way, our annotation covers 80 documents with considerable diversity.

In summary, we randomly selected 80 documents (40 in each domain) from the documents collected in Section 3.1. Then we randomly selected 10 sentences from each document. These 80 documents, along with 800 expert-annotated sentences form the DocOIE *evaluation dataset*.

3.3 Annotation Consistency Measurement

The annotation was performed by two OpenIE experts (both are authors of this paper) with reference to existing annotation processes (Stanovsky and Dagan, 2016; Bhardwaj et al., 2019). The dataset was annotated in three stages.

In the first stage, the two annotators practiced annotations independently on 100 sentences among the 800 sentences. Then they cross-validated the annotation results, discussed them to resolve disagreements, and updated annotation policy.

In the second stage, the two experts independently annotated another 100 sentences among the remaining 700 sentences. These two sets of annotations are used for measuring annotation consistency. Because it is not straightforward to evaluate annotation agreement by measures like Kappa coefficient, we adopted the evaluation scorer proposed by CaRB (Bhardwaj et al., 2019). The scorer performs matching at tuple level instead of lexical

---

[6] `http://patft.uspto.gov/netahtml/PTO/search-bool.html`

Consistency	Precision	Recall	F1
A←B	90.7	92.4	91.6
B←A	84.6	92.0	88.2
Average	87.7	92.2	89.9

Table 4: Annotation consistency estimated between annotators A and B. A←B indicates evaluation of A's annotations with B's annotations as ground truth.

Analysis	Healthcare	Transportation
Sentence semantic analysis (%)		
Conjunction	61.00	61.50
Terminology	60.00	42.75
Dependent Clause	29.00	41.50
Tuple semantic analysis (%)		
Negative Polarity	3.26	5.16
Possibility	12.17	8.74
Under-specificity	8.80	6.49

Table 5: Fine-grained statistical analysis for 800 annotated sentences and 2,122 tuples. Percentages refer to the respective domain: Healthcare and Transportation.

level. Specifically, we score one expert's annotations by treating the other's annotations as ground truth. Among the tuple matching strategies in CaRB, we used the default binary lenient tuple matching, to estimate the consistency between the two annotators. Reported in Table 4, the two annotators reach high-level agreement in annotations with an average F1 of 89.9%.

Based on the high-level annotation consistency, in the third stage, each expert independently annotated 300 sentences from the remaining 600 sentences. The annotations are then validated by the other expert, and annotation disagreements are resolved through discussion.

3.4 Analysis of DocOIE Evaluation Dataset

To understand the difficulty of DocOIE, we provide an analysis of the annotated sentences and tuples in DocOIE, similar to (Gashteovski et al., 2019).

Sentence-level Analysis. We evaluate the complexity of a sentence on whether it contains conjunction word, terminology mention, and dependent clause. Consider the following sentence as an example: *"Though depicted as a distinct step, it may be performed as part of the VAD or ASR processes." "VAD processes"* and *"ASR processes"* are terminology mentions. The conjunction word *"or"* in the sentence connects these two terminologies. In addition, dependent clause refers to the subordinate clause "Though depicted as a distinct step". Table 5 reports the percentages of sentences that contain at least one conjunction word, terminology mention, and dependent clause, respectively.

Tuple-level Analysis. Consider an example sentence *"They may be implemented as instructions stored on a machine-readable medium."* Two tuples can be extracted from this sentence: (1) "*they ; may be implemented as ; instructions*", and (2) "*instructions ; are stored on ; a machine-readable medium*".

We analyze a tuple on whether it is a certainty or merely a possibility. The relation "*may be implemented as*" indicates tuple (1) a possibility because of the modal verb "*may*". Then, negative polarity refers to certainty negation, which can be indicated by words such as "*not*" or "*no*". Some tuples are under-specified in either tuple subject or object. The main reason for under-specificity is the lack of co-reference information, where we need additional context to obtain a coherent meaning. In tuple (1), its subject "*they*" is under-specified because co-reference information is needed to resolve what "*they*" refers to.[7] Table 5 reports the percentages of tuples that are under each of the categories.

3.5 DocOIE Training Dataset

Besides the 80 documents for expert annotations, we further sample 2,400 documents randomly (1,200 in each domain) from the documents collected in Section 3.1 to create DocOIE training dataset. The 1,200 documents in each domain contain around 120,000 sentences, which is sufficient for pseudo label generation, required by neural OpenIE models.

4 Pseudo Label by Bootstrapping

Following the common practice (Kolluru et al., 2020b; Cui et al., 2018; Zhao et al., 2020), we generate pseudo labels by bootstrapping with traditional OpenIE models. Before we run these models on the DocOIE training dataset, we evaluate their performances on the DocOIE evaluation dataset, to select the models which can generate better quality pseudo labels.

---

[7] Co-reference is not annotated in DocOIE.

OpenIE Model	AUC	Prec	Rec	F1
Healthcare Domain				
Reverb	35.4	**79.9**	42.8	55.8
Stanford	16.5	11.0	29.7	16.1
Clausie	22.1	38.8	53.8	45.1
OpenIE4	35.4	59.5	55.1	57.2
OpenIE5	29.1	53.6	50.5	52.0
Rev+Oie4	**36.8**	75.8	47.7	**58.6**
Oie4+Rev	35.8	59.6	**55.3**	57.4
Transportation Domain				
Reverb	29.3	**79.1**	36.3	49.7
Stanford	15.7	13.2	27.8	17.9
Clausie	18.0	36.2	48.4	41.4
OpenIE4	29.2	52.8	51.2	52.0
OpenIE5	25.0	50.9	43.8	47.1
Rev+Oie4	**31.0**	74.2	42.4	**54.0**
Oie4+Rev	30.1	53.4	**52.7**	53.0

Table 6: Performance of OpenIE models on DocOIE evaluation dataset. The best scores are in boldface and second best scores are underlined.

We evaluate the models by using CaRB scorer (Bhardwaj et al., 2019). Table 6 reports the performance of five independent OpenIE models: Reverb (Fader et al., 2011), Clausie (Corro and Gemulla, 2013), Stanford OpenIE (Angeli et al., 2015), OpenIE4 (Mausam, 2016) and OpenIE5[8]. In addition to these five models, we also evaluated two combinations of Reverb and OpenIE4. With **Rev+Oie4**, Reverb is the main system and if Reverb fails to extract any tuples from a sentence, we complement the extraction by using Openie4. Similarly, **Oie4+ Rev** uses OpenIE4 as the main system, and the extractions are complemented by Reverb.

All the evaluated models show consistent performance in both domains. By F1 score, both Reverb and OpenIE4 are the best performing individual models and their combinations lead to the best and second best F1 scores in both domains. Accordingly, by applying OpenIE4, Reverb, and their combinations, the number of sentences and tuplesextracted from the DocOIE training dataset are reported in Table 7.[9] Note that, a sentence is not counted if it has no extracted tuples, which leads to the different sentence number.

OpenIE model	Healthcare #Sent	Healthcare #Tuple	Transportation #Sent	Transportation #Tuple
OpenIE4	117k	263k	111k	258k
Oie4+Rev	121k	268k	114k	262k
Reverb	103k	146k	97k	141k
Rev+Oie4	121k	181k	114k	173k

Table 7: Number of sentences and tuples extracted by Reverb, OpenIE4 and their combinations. The sentence is not included if it has no tuples extracted.

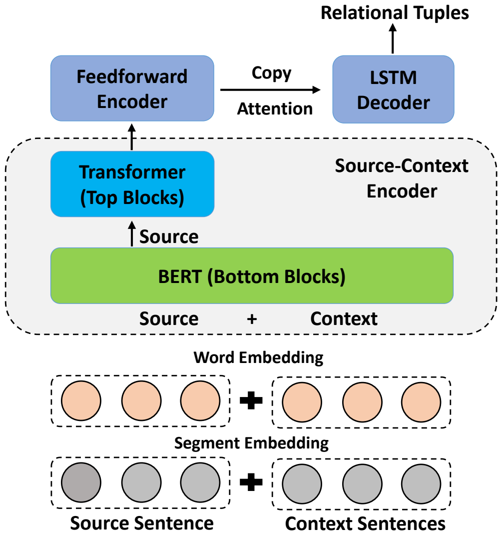

Figure 2: The architecture of DocIE.

5 DocIE Model

In this section, we present the proposed **Doc**ument-level context-aware Open **I**nformation **E**xtraction model, named **DocIE**. As shown in Fig. 2, DocIE mainly consists of two parts: source-context encoder, and encoder-decoder.

Document-level Context Formally, we denote a document as $D = \{s_1, s_2, \ldots, s_N\}$ consisting of N sentences. The source sentence s_i is the input sentence that relational tuples are extracted from. Given source sentence s_i, we regard its surrounding sentences $c_i = \{s_{i-t}, \ldots, s_{i-1}, s_{i+1}, \ldots, s_{i+t}\}$ as contextual sentences, where t represents the context window size. The larger t is, the more document-level context c_i covers.

Source-Context Encoder The source-context encoder is inspired by a recent work (Ma et al., 2020) which adopts Flat-Transformer to incorporate context into source sentence, for machine translation. In DocIE, our encoder consists of (i) bottom blocks which take the concatenation of source sen-

---

[8] `github:dair-iitd/openie-standalone`

[9] The number of sentences may vary because of different text pre-processing and sentence segmentation strategies adopted. The number of tuples extracted may vary due to different versions of OpenIE tools.

tence and context sentences as input, and (ii) top blocks which take only the representation of the source sentence from the bottom blocks as input.

In our implementation, we use BERT (Devlin et al., 2019) as the bottom blocks to perform semantic interactions between source sentence s and context c. We first project both s and c into embedding space by summing their word embedding and segment embedding, *i.e.,* $e_s = E(s) + S(s)$ and $e_c = E(c) + S(c)$. Here, E is the trainable word embedding matrix, and S is the trainable segment embedding matrix. The segment embedding is to distinguish words in source sentence from words in context sentences. They are initialized to 0 and 1 for words in source and context sentences respectively. Then we concatenate e_s with e_c as $[e_s; e_c]$ as the input to the source-context encoder.

$$h_1[s; c] = \mathrm{BERT}([\mathrm{e_s}; \mathrm{e_c}]) \tag{1}$$

BERT, with multiple layers of transformers, merges source sentence information and its contextual information. We use the last hidden state $h_1[s; c]$ of BERT as the representation of the two concatenated input sequences.

On top of the BERT blocks, we add Transformer as top blocks (Vaswani et al., 2017) to prepare the source sentence representation for the following encoder-decoder. The source sentence representation is obtained by truncating the latter (context sentences representation) $h_1[c]$ from $h_1[s; c]$. Therefore only the former (source sentence representation) $h_1[s]$ is kept.

$$h_2[s] = \mathrm{Transformer}(\mathrm{h_1[s]}) \tag{2}$$

Encoder-Decoder The encoder-decoder generation module follows CopyAttention (Cui et al., 2018) which casts OpenIE task as a sequence-to-sequence generation task with copying mechanism. The encoder-decoder framework represents a variable length input sequence in the encoder and uses it in the decoder to generate output sequence. In our encoder-decoder framework, attention mechanism (Bahdanau et al., 2015) is used to align the encoder hidden state with the decoder hidden state, jointly maximizing the log probability of output tuples, conditioned on the input sentence. Meanwhile, since tuple arguments and relation are normally sub-spans of the input sentence, additional copying mechanism (Gu et al., 2016) is applied. It helps copy words directly from the input sentence to the output tuples.

6 Experiments

We evaluate DocIE and compare its results with two baseline neural OpenIE models, CopyAttention+BERT and IMOJIE (Kolluru et al., 2020b). Kolluru et al. (2020b) report that CopyAttention+BERT is a strong baseline. Meanwhile, DocIE adopts CopyAttention (Cui et al., 2018) as its encoder-decoder module. Hence CopyAttention+BERT can be considered as the base model, from which DocIE adds context modelling.

6.1 Neural Baseline Models

We first evaluate the two neural baseline models trained with the pseudo labels listed in Table 7. The evaluation is conducted on the DocOIE evaluation dataset with CaRB scorer.

Reported in Table 8, CopyAttention+BERT outperforms IMOJIE in most settings by both measures: AUC and F1. In general, for both models, pseudo labels by Rev+Oie4 (and also Reverb) lead to better results in healthcare domain. Pseudo labels by Oie4+Rev (and also OpenIE4) generate better results in transportation domain. During our annotation of the 800 sentences, we observe that sentences in transportation domain tend to contain slightly more conjunctions (*e.g.,* multiple conjunctions in one sentence) and thus have more coordinating structures than those in healthcare. OpenIE4 system generally extracts more tuples than Reverb (refer to Table 7) and provides higher recall. Therefore, extractions in transportation domain with more conjunctions may better match the tuples extracted by OpenIE4.

Based on this set of results, in our following experiments, we use pseudo labels by Rev+Oie4 for healthcare domain, and pseudo labels by Oie4+Rev for transportation domain.

6.2 DocIE Against Baselines

In this section, we evaluate DocIE against sentence-level OpenIE systems. We refer DocIE without the top transformer layer as "DocIE w/o transformer" and DocIE as "DocIE w transformer" for clarity. The context window size of DocIE is set to 5 for healthcare domain and 4 for transportation domain.

Table 9 summarizes the experiment results. For easy comparison, the results of the best traditional OpenIE baselines (refer to Table 6) and neural OpenIE models (refer to Table 8) are replicated here. Observe that DocIE w transformer achieves the best AUC and F1 in both domains. Its variant, DocIE

Neural OpenIE	Pseudo labels	Healthcare				Transportation			
		AUC	Prec	Rec	F1	AUC	Prec	Rec	F1
CopyAttention+BERT	OpenIE4	38.6	54.4	**51.6**	52.9	38.5	54.3	**57.6**	55.9
	Oie4+Rev	40.4	57.1	50.4	53.5	38.3	55.3	56.9	**56.1**
	Reverb	43.7	77.8	46.4	58.1	36.9	70.5	42.2	52.8
	Rev+Oie4	**46.8**	**77.9**	48.6	**59.8**	**40.3**	**72.1**	43.9	54.6
IMOJIE	OpenIE4	36.2	73.0	**47.7**	57.7	35.7	62.9	48.8	55.0
	Oie4+Rev	34.1	69.5	46.7	55.9	**35.8**	63.5	**49.2**	**55.5**
	Reverb	38.5	79.2	45.6	57.9	33.2	77.3	39.2	52.0
	Rev+Oie4	**39.7**	**80.1**	46.4	**58.7**	33.0	**77.4**	39.6	52.4

Table 8: Neural baseline models trained with different pseudo labels. The best scores of *each model* are in boldface.

System	Healthcare				Transportation			
	AUC	Prec	Rec	F1	AUC	Prec	Rec	F1
Rev+Oie4	36.8	75.8	47.7	58.6	31.0	**74.2**	42.4	54.0
Oie4+Rev	35.8	59.6	**55.3**	57.4	30.1	53.4	52.7	53.0
CopyAttention+BERT	46.8	77.9	48.6	59.8	38.3	55.3	56.9	56.1
IMOJIE	39.7	**80.1**	46.4	58.7	35.8	63.5	49.2	55.5
DocIE w/o transformer	47.1	76.2	49.9	60.3	**38.5**	55.8	57.0	56.4
DocIE w transformer	**47.4**	74.4	51.3	**60.8**	**38.5**	56.0	**57.5**	**56.9**

Table 9: Results of DocIE and baselines. The best scores are in boldface and second best scores are underlined.

w/o top transformer, is the second best performer and outperforms all sentence-level models.

The experiment results suggest that incorporating document-level context is helpful in improving OpenIE. On the other hand, we remark that DocIE is trained by pseudo labels produced by traditional OpenIE models which do not consider document-level context. The potential of utilizing document-level context is yet to be fully realized.

6.3 Impact of Context Window Size

The setting of window size determines the number of context sentences to be considered. We evaluated the range from 1 to 6 and plot F1 scores against window size changes in Fig. 3. Observe that the optimal window size for healthcare domain is 5, and the number is 4 for transportation. Better F1 scores are observed along the increase of context sentence window, till 4 or 5. In general, 8~10 surrounding sentences (window size 4 or 5) provide sufficient context for sentence understanding. Small window size might not provide sufficient context, and a large window size might introduce noise and dominate source representation learning.

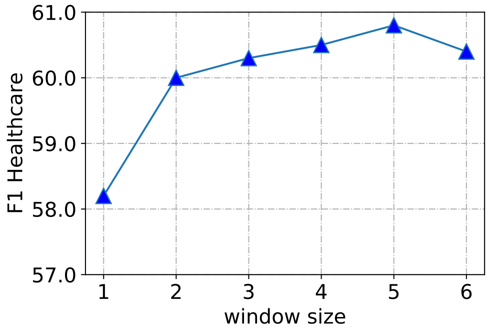

(a) Healthcare

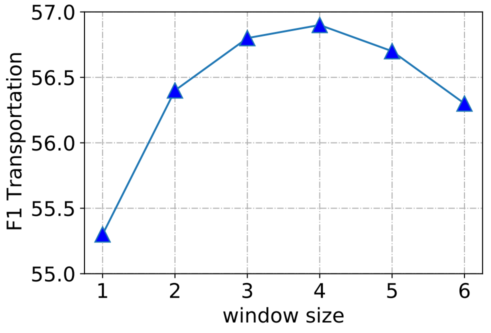

(b) Transportation

Figure 3: F1 with varying window sizes, on both domains.

6.4 Case Study

We use the two example sentences shown in Fig. 1 as a case study, to illustrate the differences between DocIE and the sentence-level neural OpenIE baselines: CopyAttention+BERT and IMOJIE.

For Sentence 1, CopyAttention+BERT incorrectly recognizes the word "transfers" as a verb, thus extracting an incorrect tuple (data ; transfers to ; a single target terminal). IMOJIE, however, completely misses the key phrase "data transfers" and extracts an incorrect tuple (a single target terminal ; using ; the invention). Only DocIE manages to extract the correct tuple (data transfers to a single target terminal ; using ; the invention).

For Sentence 2, there is no explicit clue about the relationship between "a device" and "a cellular base". Both CopyAttention+BERT and IMOJIE treat "a device a cellular base station" as a whole and mistakenly generate a tuple (Node-B ; can be ; a device a cellular base station having beam-forming antennas). In contrast, DocIE successfully splits "a device a cellular base station" by referring to surrounding context and extracts the correct tuple (Node-B ; can be ; a device). However, DocIE fails to infer the inter-relationship between "a device" and "a cellular base". Accordingly, another correct tuple (a device ; is such as ; a cellular base station) is not extracted.

Results of the two example sentences show the improvements made by DocIE after leveraging contextual information for tuple extraction.

6.5 Error Analysis

Similar to the error analysis performed in (Kolluru et al., 2020b), we examine tuples extracted by DocIE from 50 randomly selected sentences in DocOIE. We identify the following major error types. (i) **Incompleteness:** In 28% sentences, DocIE fails to cover at least one key phrase in either arguments or relation. Missing key phrases result in incomplete information extraction. (ii) **Incorrect Boundary:** 27% extractions misinterpret the syntactic meaning of the sentence, leading to incorrect boundary of arguments and relation. (iii) **Redundant Extractions:** 15% sentences contain redundant extractions; that is, the same relational fact is extracted multiple times from a sentence or phrase. (iv) **Grammatical Errors:** 13% extractions are not grammatically correct. Most grammatical errors are contributed by the incorrect verb form used in tuple relation.

6.6 Implementation

We implement DocIE using the AllenNLP framework[10] in Pytorch 1.4. Pre-trained BERT[11] is fine-tuned at learning rate 2×10^{-5} to get contextualized word embeddings. The learning rate for the other modules is set to 1×10^{-4}. The input dimension, projection dimension, feedforward hidden dimension, number of layers, and number of attention heads of top transformer encoder are set to 768, 256, 3072, 2, and 8, respectively. The hidden dimension, and word embedding dimension of the LSTM-decoder are set to 256 and 100 respectively.

7 Conclusion

In this paper, we propose to consider document-level contextual information for OpenIE task. We contribute DocOIE, the first document-level context-aware OpenIE dataset. It consists of 800 expert-annotated sentences from 80 documents. The documents are carefully selected and the annotations are completed by experts with high-level annotation consistency.

With the help of DocOIE, we conduct evaluation of neural OpenIE models and demonstrate that incorporating document-level context is helpful in improving OpenIE performance through DocIE. As a baseline for document-level context-aware OpenIE, DocIE achieves promising results compared with all sentence-level OpenIE models. Our future works are in two main directions. One is to research on more effective context-aware OpenIE models, and the other is to investigate the possibility of not relying on pseudo labels.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A19E2b0098 and #A18A2b0046).

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages

---

[10] `https://github.com/allenai/allennlp`

[11] `https://huggingface.co/transformers/model_doc/bert.html`

344–354, Beijing, China. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Hannah Bast and Elmar Haussmann. 2013. Open information extraction via contextual sentence decomposition. In *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, pages 154–159. IEEE Computer Society.

Sangnie Bhardwaj, Samarth Aggarwal, and Mausam Mausam. 2019. CaRB: A crowdsourced benchmark for open IE. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6262–6267, Hong Kong, China. Association for Computational Linguistics.

Nikita Bhutani, H. V. Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 55–64, Austin, Texas. Association for Computational Linguistics.

Matthias Cetto, Christina Niklaus, André Freitas, and Siegfried Handschuh. 2018. Graphene: Semantically-linked propositions in open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2300–2311, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 355–366. International World Wide Web Conferences Steering Committee / ACM.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 407–413, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale Seq2Seq models to multi-document inputs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4186–4196, Hong Kong, China. Association for Computational Linguistics.

Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. MinIE: Minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, Copenhagen, Denmark. Association for Computational Linguistics.

Kiril Gashteovski, Rainer Gemulla, Bhushan Kotnis, Sven Hertling, and Christian Meilicke. 2020. On aligning OpenIE extractions with knowledge bases: A case study. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 143–154, Online. Association for Computational Linguistics.

Kiril Gashteovski, Sebastian Wanner, Sven Hertling, Samuel Broscheit, and Rainer Gemulla. 2019. OPIEC: an open information extraction corpus. In *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–316, Vancouver, Canada. Association for Computational Linguistics.

Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Mausam, and Soumen Chakrabarti. 2020a.

OpenIE6: Iterative Grid Labeling and Coordination Analysis for Open Information Extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3748–3761, Online. Association for Computational Linguistics.

Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Mausam, and Soumen Chakrabarti. 2020b. IMoJIE: Iterative memory-based joint open information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5871–5886, Online. Association for Computational Linguistics.

William Lechelle, Fabrizio Gotti, and Phillippe Langlais. 2019. WiRe57 : A fine-grained benchmark for open information extraction. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 6–15, Florence, Italy. Association for Computational Linguistics.

Shuming Ma, Dongdong Zhang, and Ming Zhou. 2020. A simple and effective unified encoder for document-level machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3505–3511, Online. Association for Computational Linguistics.

José-Lázaro Martínez-Rodríguez, Ivan López-Arévalo, and Ana B. Ríos-Alvarado. 2018. Openie-based approach for knowledge graph construction from text. *Expert Syst. Appl.*, 113:339–355.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4074–4077. IJCAI/AAAI Press.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3866–3878, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305, Austin, Texas. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael Cafarella, Oren Etzioni, and Stephen Soderland. 2007. TextRunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, Rochester, New York, USA. Association for Computational Linguistics.

Junlang Zhan and Hai Zhao. 2020. Span model for open information extraction on accurate corpus. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9523–9530. AAAI Press.

Hongyan Zhao, Ru Li, Xiaoli Li, and Hongye Tan. 2020. Cfsre: Context-aware based on frame-semantics for distantly supervised relation extraction. *Knowledge-Based Systems*, 210:106480.

DocOIE: A Document-level Context-Aware Dataset for OpenIE (Supplementary Material)

A DocOIE Annotation Agreement

This section complements Section 3.3 Annotation Consistency Measurement.

We take the sentence shown in Table 10 as an example, to demonstrate how the two experts clarified inconsistency and updated annotation policy. In blue-colored tuples, Expert A broke down a long argument *"nerve grafts in peripheral nerve tissue engineering"* into an additional tuple *(nerve grafts; are in; peripheral nerve tissue engineering)*, while Expert B chose to contain the descriptive part in this argument. After discussion, they decided to follow Expert B's practice to ensure complete argument without losing descriptive information. Meanwhile, as shown in red-colored tuples, the two experts disagreed on the inner relation of the argument "aft tissue-derived ECM modified tissue engineered nerve grafts". Expert A considered "aft tissue-derived ECM" should be modified by "tissue engineered nerve grafts" but Expert B thought "nerve grafts" shall be engineered by "aft tissue-derived ECM modified tissue". This inconsistency is resolved by referring to the relevant contexts in the document: "tissue engineered nerve grafts" is a terminological phrase and there exists an action of "ECM modification" performing on "tissue engineered nerve grafts". The annotation agreed by both experts are shown in the last row in Table 10.

B Extraction Results

Table 11 lists the tuples extracted by DocIE and the sentence-level neural OpenIE baselines: CopyAttention+BERT and IMOJIE. This table complements Section 6.4 Case Study.

Source Sentence	***The ANG meets the basic requirements for nerve grafts in peripheral nerve tissue engineering, and becomes aft tissue-derived ECM modified tissue engineered nerve grafts.***
Annotation by Expert A	(the ANG ; meets the basic requirements for ; nerve grafts) (nerve grafts ; are in ; peripheral nerve tissue engineering) (the ANG ; becomes ; tissue engineered nerve grafts) (tissue engineered nerve grafts ; is modified by ; aft tissue-derived ECM)
Annotation by Expert B	(the ANG ; meets the basic requirements for ; nerve grafts in peripheral nerve tissue engineering) (the ANG ; becomes ; aft tissue-derived ECM modified tissue engineered nerve grafts) (nerve grafts ; is engineered by ; aft tissue-derived ECM modified tissue)
Document Context	...another group is using plain ***tissue engineered nerve grafts*** (without ***ECM modification***) for repairing the sciatic nerve defects of rats....
Annotation in Agreement	(the ANG ; meets the basic requirements for ; nerve grafts in peripheral nerve tissue engineering) (the ANG ; becomes ; aft tissue-derived ECM modified tissue engineered nerve grafts) (tissue engineered nerve grafts ; is modified by ; aft tissue-derived ECM)

Table 10: An example of inconsistent annotations between two annotators, and the annotation in agreement.

Sentence 1	***In some circumstances, data transfers to a single target terminal using the invention might not be significantly faster than conventional download methods.***
Annotated tuples	(data transfers to a single target terminal ; use ; the invention) (data transfers to a single target terminal using the invention ; might not be significantly faster than ; conventional download methods)
DocIE	(data transfers to a single target terminal ; using ; the invention) (data transfers to a single target terminal using the invention ; might not be ; significantly faster than conventional download methods)
CopyAttention+BERT	(data ; transfers to ; a single target terminal) (a single target terminal ; using ; the invention) (data transfers to a single target terminal ; might not be ; significantly faster than conventional download methods)
IMOJIE	(a single target terminal ; using ; the invention) (data transfers to a single target terminal using the invention ; might not be ; significantly faster than conventional download methods) (data transfers to a single target terminal using the invention ; might not be significantly faster than conventional download methods ;) (data transfers to a single target terminal using the invention ; might not be significantly faster ;)
Sentence 2	***Node-B can be a device a cellular base station having beam-forming antennas that serves various sectors of a cell.***
Annotated tuples	(Node-B ; can be ; a device) (a device ; is such as ; a cellular base station) (a cellular base station ; has ; beam-forming antennas) (Node-B ; serves ; various sectors of a cell)
DocIE	(Node-B ; can be ; a device) (a cellular base station ; having ; beam-forming antennas) (beam-forming antennas ; serves ; various sectors of a cell)
CopyAttention+BERT	(Node-B ; can be ; a device a cellular base station having beam-forming antennas) (a cellular base station ; having ; beam-forming antennas that serves various sectors of a cell) (beam-forming antennas ; serves ; various sectors of a cell)
IMOJIE	(Node-B ; can be ; a device a cellular base station having beam-forming antennas) (a cellular base station ; having ; beam-forming antennas) (beam-forming antennas ; serves ; various sectors of a cell)

Table 11: Tuples extracted respectively by DocIE, CopyAttention+BERT, and IMOJIE from the two example sentences in Fig. 1.

Event Extraction from Historical Texts: A New Dataset for Black Rebellions

Viet Dac Lai[1], **Minh Van Nguyen**[1], **Heidi Kaufman**[2], **Thien Huu Nguyen**[1]
[1]Department of Computer and Information Science
[2]Department of English
University of Oregon, Eugene, OR 97403, USA
`{vietl,minhnv,thien}@cs.uoregon.edu`
`hkaufman@uoregon.edu`

Abstract

Understanding historical events is necessary for the study of contemporary society, culture, and politics. In this work, we focus on the event extraction task (EE) to detect event trigger words and their arguments in a novel domain of historical texts. In particular, we introduce a new EE dataset for a corpus of nineteenth-century African American newspapers. Our goal is to study the discourse of slave and non-slave African diaspora rebellions published in the periodical press in this period. Our dataset features 5 entity types, 12 event types, and 6 argument roles that concern slavery and black movements between the eighteenth and nineteenth centuries. Historical newspapers present many challenges for existing EE systems, including the evolution of meanings of words and the extensive use of religious discourse in newspapers from this era. Our experiments with current state-of-the-art EE systems and BERT models demonstrate their poor performance over historical texts and call for more robust research efforts in this area.

1 Introduction

In the last two decades, the emergence of digital humanities has transformed scholarship in the humanities. Historical documents are now massively digitized into photos and texts that allow researchers to query across collections and languages (Piotrowski, 2012). Despite the convenience of these applications (Yang and Eisenstein, 2016), a gap still exists between datasets and research methods. As such, humanities scholars do not solely interpret historical facts from statistical figures derived from massive data. Rather, they prefer reading texts and interpreting words in historical and cultural context, or by associating texts with the circumstances surrounding their publication. This working methodology requires an emphasis on the quality of the data over the quantity of the data. Recent advances of natural language processing (NLP) aim to bridge the gap between qualitative and quantitative analyses by identifying, extracting, and counting contextual data (Won et al., 2018; Wadden et al., 2019; Lin et al., 2020). This new approach provides contextual information about real-life entities (e.g., individuals, locations, times, documents) which can be later integrated into knowledge bases (Won et al., 2018) to aid historical research and discourse analysis.

In this work, we explore Information Extraction (IE) in NLP for humanities research in support of the important and complicated process of **knowledge extraction** from historical texts. Particularly, we investigate the Event Extraction (EE) task which identifies event trigger words of pre-determined event types (the most important words/phrases to evoke events) (Li et al., 2013), together with its arguments (e.g., participants, locations). For example, in the following sentence an EE system should be able to detect the word "*proclaimed*" as a trigger word of the event type "*Law_Approve*" and associate it with the arguments, i.e., agent (*Capitol*), beneficiary (*the slave*), and datetime (*now*).

Freedom to the slave should now be ***proclaimed*** *from the Capitol, and should be seen above the smoke and fire of every battle field.*

To enable the development and evaluation of EE models for historical text, benchmark datasets play an important role. However, most of the current datasets in EE (i.e., ACE-2005 (Walker et al., 2005) and TAC KBP (Mitamura et al., 2015)) are not suitable for this domain for several reasons. First, these datasets are collected from various sources without a target topic (Walker et al., 2005; Mitamura et al., 2015). Therefore, tracking the evolution of some specific movements or progress, which is of great interest to literary scholars and historians, is not a feasible goal. Second, documents in these datasets

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2390–2400
August 1–6, 2021.

are derived from recent articles and documents in which the use of words in the text differ from their uses in the past. For example, some words obtain new semantics over time, and the dominance of religion in the past led to extensive use of religion-related words and figurative language in historical publications. Last but not least, existing EE datasets mostly concern events in common human life, such as giving birth, transportation, and crimes. These events might not relate to the subjects literary scholars and historians want to study.

To redress this problem, we introduce a novel EE dataset for historical texts, called BRAD, focusing on Black Rebellions in African Diaspora (i.e., African American population). BRAD's documents are selected by a humanities expert and are annotated by EE experts for 5 entity types, 12 event types, and 6 argument roles. Finally, we evaluate the state-of-the-art EE models on BRAD. Our experiments show that the performance for historical texts of current EE models is significantly poorer than those for modern texts, necessitating further research into this area. We will also release our dataset and code to facilitate future research.

2 Data Collection and Annotation

In this project we use documents from the African American newspaper corpus. These documents involve news articles derived from nineteenth-century African American periodicals[1] published from 1827 to 1909.

To create an EE dataset, we first designed a set of event types and annotation guidelines, consulting our humanities expert who specializes in nineteenth-century literature. In particular, we focus on the four most important events for Black rebellions presented in our corpus, including **Humanity**: a humanity event concerns a violation or facilitation of basic human rights (e.g. living, freedom, property); **Law**: a law event characterizes an introduction, approval or appeal of a law; **Conflict**: a conflict event represents an act of violence; it includes the initialization, development, and consequences of a violent act; and **Justice**: a justice event captures an act of punishment of the government to the people who violate a law. These four events are further expanded into 12 event sub-types. Tables 7 and 8 present event types along with their descriptions and examples in BRAD. To capture arguments for such events, we introduce five entity types (i.e., Person, Organization, Geographical-Political Entities, Time, and Document). The first four entity types follow the definition in the ACE 2005 guideline (Walker et al., 2005) while the Document type represents government documents (e.g., Slavery Act) used in events. Finally, we define six argument roles that such entity types can play in our events, including Time, Location, Agent, Patient, Object, and Beneficiary. Tables 9 and 10 provide more descriptions and examples of these argument roles for each event type.

The African American corpus is a large corpus of 177,582 articles. We thus select documents that are relevant to our focused topic of Black diaspora rebellions. First, automatic selection is done by keyword matching to identify documents related to slavery and insurrection. As such, our humanities expert defined a set of keywords for the topic of rebellion. In the nineteenth century this cluster of words were used interchangeably to describe African diaspora rebellion events (e.g., "*rebel*", "*revolt*", "*strike*", "*insurrection*"). We used the Stanford CoreNLP toolkit to split and tokenize documents into sentences and words. Next, for each document in the corpus, we counted the number of words in the document that appears in the designated keyword set (called matching rate). The top 1000 documents with the highest matching rates are selected for further consideration. In the second step, the humanities expert examined the 1000 documents to identify relevant documents for Black rebellions, leading to the selection of 151 documents used for the EE annotation.

In the next step we recruited two graduate students to annotate the selected documents for EE. Each student was independently trained on the annotation guideline and performed a group of exercises to better recognize events and entities. The students annotated the 151 documents for entity mentions and event triggers, achieving Cohen's Kappa scores of 0.81 and 0.82 respectively. Note that these scores are very close to the near-perfect agreement range of $[0.81, 0.99]$. To further improve the quality of the dataset, our humanities expert will resolve the annotation conflicts that arise between the two students, leading to the final annotation version of entity mentions and event triggers in the 151 documents. In the next step, given the reconciled entity mention and event trigger annotation, the two students continue to annotate event

---

[1] Douglass Monthly, The Frederick Douglass Paper, Freedom Journals, The Christian Recorder, The Colored American.

arguments for the event triggers. Our evaluation shows a Cohen's Kappa score of 0.75 that indicates a strong agreement between the two annotators. Also, the lower agreement score for event arguments suggests that event argument annotation is more ambiguous than those for entity mentions and event triggers. Finally, our domain expert was consulted to resolve any conflicts in event argument annotation, producing the final version of our BRAD dataset with the 151 documents. To facilitate the development of EE models, we then split BRAD into three portions for training, development, and test data with 101, 25, and 25 documents, respectively. Table 1 presents the statistics while Table 2 and 3 presents the frequencies of event and entity types in our BRAD dataset.

	Train	**Dev**	**Test**	**Total**
#document	101	25	25	151
#sentence	3,847	925	866	5638
#token	117,278	27,860	26,920	172,058
#event trigger	2,720	606	933	4,259
#entity mention	14,389	3,287	3,749	21,425
#event argument	6,057	1,219	2,570	9,846

Table 1: Statistics of the BRAD dataset.

Event Type	**#Event**
Conflict_Attack	1,628
Conflict_Other	971
Humanity_Deprive	577
Humanity_Endow	376
Conflict_Injure	153
Law_Approve	145
Law_Repeal	141
Law_Propose	108
Justice_Arrest-Jail	78
Conflict_Protest	42
Justice_Execute	26
Justice_Sentence	14

Table 2: Distribution of event types in BRAD.

Annotation Challenges: During the EE annotation process of historical texts, we found several noteworthy challenges regarding the ability to achieve interpretive consensus of the texts.

First, for the domain expertise, we find that the use and meaning of words evolves over time and across geographical regions, potentially introducing new meanings or making one meaning more popular than the others. Language is always in perpetual flux. As such, understanding texts from the past requires analysis of the context in which texts were written. In order to be effective, the annotations must be attentive to these contexts. For example, in the following sentence, "*Congress*" and "*her*" are two mentions of the *USS Congress* battleship launched by the United State Navy in 1841. Without historical knowledge, our current perception might interpret "*Congress*" as the legislative branch of the United States. In fact, the second clause mentions the wooden hull that helps to clarify it as the battleship that sunk in 1862 during the US Civil War. Such misinterpretation might lead to incorrect annotations and analyses.

Entity Type	**#Entity**
PERSON (PER)	12,599
ORGANIZATION (ORG)	3,836
GEOPOLITICAL (GPE)	2,873
DOCUMENT (DOC)	1,121
DATETIME (TIME)	996

Table 3: Distribution of entity types in BRAD.

"*The* ***Congress*** *was visited and received the shots and shells in all part of* ***her*** *wooden hull*".

Second, we find that annotation disagreements are more likely to occur in the interpretation of event triggers. In BRAD, we allow event triggers to involve multiple words that cause span mismatches between annotations for some confusing cases (e.g., annotating the whole phrase "*make the black man equal*" as an event trigger or annotating "*make*" and "*equal*" as two separate triggers). Another form of popular disagreement involves mismatches on event types. Consider the following sentence as an example:

"*Believing his life to be in danger, Patmon stepped back, drew his revolver, and told the fellow to surrender, or he would* ***shoot*** *him* ."

Two annotators agree that the word "*shoot*" is an event trigger. However, one annotator considers this as an event of type *Conflict_Attack* as it is a part of the conflict between the overseer ("*Patmon*") and the slave ("*fellow*", "*him*"); the other annotator, on the other hand, treats "*shoot*" as a *Humanity_Deprive* event as the overseer is threatening to kill the slave (i.e., taking the right to life).

Data Analysis: To illustrate the ambiguity in BRAD, Table 4 shows five words with the highest frequency as event triggers (i.e., Event Count), along with the percentage of times these words are labeled as event triggers in the dataset (i.e., Event Rate) (Sims et al., 2019). This table demonstrates the likelihood that words with the highest event counts might not be annotated as event triggers in BRAD, thereby necessitating EE models to find a

Word	#Event	Event Rate
war	183	23.5%
rebellion	112	47.3%
insurrection	66	78.6%
revolt	77	60.2%
emancipation	58	82.9%
take	64	32.3%
put	35	47.3%
send	25	45.5%

Table 4: Event rates of the words with the highest event counts in BRAD.

	ACE 2005	BRAD
god	4.5%	17.5%
lord	0.3%	8.7%
heaven	0.7%	8.7%
mighty	0.2%	8.7%
sacred	0.2%	9.5%
curse	0.5%	5.6%
christian	1.7%	18.3%

Table 5: Percentages of documents containing religion-related words in BRAD and ACE 2005.

method of effectively capturing context in order to perform correct predictions.

Moreover, we find extensive use of religion-related words in BRAD compared to existing EE datasets. For example, considering the words "*lord*", "*heaven*", and "*christian*", the percentages of documents in ACE 2005 containing these words are only 0.3%, 0.7%, and 1.7% while those percentages for BRAD are 8.7%, 8.7%, and 18.3% respectively. Such language difference suggests the potential need to adapt existing language models to better capture the nature of historical texts which, in turn, will facilitate a more accurate performance of EE.

3 Experiment

There are three major EE tasks that BRAD supports for historical texts, including entity mention detection (EMD), event trigger detection (ED), and event argument extraction (EAE). This section aims to reveal the complexity of the EE tasks in BRAD by evaluating the performance of existing state-of-the-art models for EE on this dataset. In particular, we focus on the following state-of-the-art models for EE that leverage the pre-trained language model BERT (Devlin et al., 2019) for the text encoding and jointly perform predictions for all EE tasks in an end-to-end fashion (i.e., joint inference):

DyGIE++ (Wadden et al., 2019): This model utilizes dynamic span graphs to exploit long-range cross-sentence relationships for span representation propagation for joint IE.

OneIE (Lin et al., 2020): This model first identifies spans of entity mentions and event triggers. The detected spans are then paired to jointly predict entity types, event types, relations, and argument roles for IE. Global features are used to capture cross-task and cross-instance dependencies and are employed in the decoding phase with beam searches to improve extraction performance.

As such, we adapt the official implementations of such models from their original papers for our EE task in BRAD by ignoring the relation extraction task and re-tuning them on the BRAD development set. For both models, we employ the pre-trained BERT model (i.e., the bert-base-cased version) to encode input texts. Besides, motivated by the language difference between historical and modern texts, we further explore a variant of the BERT model by fine-tuning it on the African American corpus via the masked language modeling task (Devlin et al., 2019). Note that we exclude the 151 documents of BRAD in this fine-tuning process. This fine-tuned BERT model will also be fed into DyGIE++ and OneIE to perform EE in BRAD.

Result: Table 6 reports the performance of the models on the test set of BRAD over five sub-tasks: Entity Mention Detection (Entity), Event Trigger Identification, i.e., not concerning event types (Trig-I), Event Trigger Classification (Trig-C), Event Argument Identification, i.e., not concerning argument roles (Arg-I), and Event Argument Classification (Arg-C). For comparison, we also include the original performance of the models on the popular EE dataset ACE 2005. There are three major observations from the table. First, the performance of current EE models on BRAD is significantly and substantially worse than those on ACE across different tasks. It thus suggests that EE for historical texts in BRAD is a challenging task and more research effort is necessary to boost the EE performance for this domain. Second, comparing the performance of the models with different versions of BERT (i.e., original vs fine-tuned), it is clear that fine-tuning BERT on historical texts is beneficial for improving the performance of EE models on BRAD (especially for OneIE where the improvement is consistent across different EE sub-tasks with large margins). This observation suggests that pre-training BERT on modern texts is unable to capture the nuance of language use in

Task	Model	ACE	BRAD			BRAD$^{\#}$		
		F1	P	R	F1	P	R	F1
Entity	DyGIE++	90.7	85.6	75.9	80.5	84.3	78.6	81.4
	OneIE	90.3	85.4	77.0	81.0	85.0	79.4	82.1
Trig-I	DyGIE++	76.5	81.5	50.1	62.0	77.4	56.9	65.6
	OneIE	78.6	80.9	47.0	59.4	80.8	52.9	63.9
Trig-C	DyGIE++	73.6	62.7	38.5	47.7	61.6	40.6	49.0
	OneIE	75.2	63.5	36.9	46.7	64.9	42.2	51.2
Arg-I	DyGIE++	55.4	55.5	28.0	37.2	47.8	28.6	35.8
	OneIE	60.7	57.7	33.9	42.7	58.9	40.4	47.9
Arg-C	DyGIE++	52.5	48.8	24.6	32.7	42.5	25.4	31.8
	OneIE	58.6	49.6	29.1	36.7	52.1	31.6	39.4

Table 6: The performance of models on the test sets of BRAD and ACE 2005. The BRAD$^{\#}$ columns report the performance with BERT fine-tuned on the African American corpus.

history, thus impairing the models and requiring appropriate adaptation to boost the EE performance. Finally, we note that the human performance (F1 scores) for Entity, Trig-C, and Arg-C on BRAD are 95.43, 88.3, and 79.8 respectively. The large performance gaps between human and current EE systems thus presents many research opportunities for future work on BRAD.

4 Related work

Prior work in NLP for historical texts has mainly focused on spelling and text normalization (Pettersson et al., 2014; Bollmann et al., 2017; Flachs et al., 2019). Recently, some studies have undertaken research on historical texts with NLP tasks such as POS tagging (Yang and Eisenstein, 2016) and information extraction (Pettersson et al., 2016). However, none of this work has explored EE.

EE is an active research area due to the availability of EE datasets e.g., for general (Walker et al., 2005; Mitamura et al., 2015) and biomedical (Kim et al., 2011) domains. Most of prior studies focus on in-domain EE (Ahn, 2006; Li et al., 2013; Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2016; Yang et al., 2019; Wadden et al., 2019; Lai et al., 2020c; Nguyen et al., 2021). Some recent studies in EE have also addressed extensible learning settings for EE to new event types, e.g. zero-shot learning (Huang et al., 2018), few-shot learning (Lai et al., 2020a,b), or new domains (Naik and Rosé, 2020). The closet works to ours involve recent efforts to create new datasets for EE (Satyapanich et al., 2020; Ebner et al., 2020; Wang et al., 2020; Trong et al., 2020; Le and Nguyen, 2021). However, these works do not consider historical texts as we do.

5 Acknowledgement

This work is supported by an I3 grant from the University of Oregon Provost's Office (Incubating Interdisciplinary Initiatives) and the Army Research Office (ARO) grant W911NF-21-1-0112. This research is also based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the Better Extraction from Text Towards Enhanced Retrieval (BETTER) Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

6 Conclusion

We present BRAD, a new dataset for EE on historical texts that focuses on Black rebellions in the American Africa corpus. Our experiments demonstrate the poor performance of current models for EE on BRAD compared to those on modern texts, thus creating room for future research on EE for historical texts. We also illustrate one approach to improve current EE systems for historical texts via fine-tuning existing pre-trained language models. In the future, we plan to enlarge our datasets with more annotated documents and event types.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Simon Flachs, Marcel Bollmann, and Anders Søgaard. 2019. Historical text normalization with delayed rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2011. Extracting bio-molecular events from literature—the bionlp'09 shared task. *Computational Intelligence*, 27(4):513–540.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020a. Exploiting the matching information in the support set for few shot event classification. In *Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Viet Dac Lai, Thien Huu Nguyen, and Frank Dernoncourt. 2020b. Extensively matching for few-shot learning event detection. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020c. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Duong Le and Thien Huu Nguyen. 2021. Fine-grained event trigger detection. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. 2015. Overview of tac kbp 2015 event nugget track. In *Proceedings of the Text Analysis Conference (TAC)*.

Aakanksha Naik and Carolyn Rosé. 2020. Towards open domain event trigger identification using adversarial domain adaptation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minh Van Nguyen, Viet Lai, and Thien Huu Nguyen. 2021. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Eva Pettersson, Jonas Lindström, Benny Jacobsson, and Rosemarie Fiebranz. 2016. Histsearch-implementation and evaluation of a web-based tool for automatic information extraction from historical text. In *HistoInformatics@ DH*.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2014. A multilingual evaluation of three spelling normalisation methods for historical text. In *Proceedings of the 8th workshop on language technology for cultural heritage, social sciences, and humanities (LaTeCH).*

Michael Piotrowski. 2012. Natural language processing for historical texts. *Synthesis lectures on human language technologies*, 5(2):1–157.

Taneeya Satyapanich, Francis Ferraro, and Tim Finin. 2020. CASIE: Extracting cybersecurity event information from text. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI).*

Matthew Sims, Jong Ho Park, and David Bamman. 2019. Literary event detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL).*

Hieu Man Duc Trong, Duc Trong Le, Amir Pouran Ben Veyseh, Thuat Nguyen, and Thien Huu Nguyen. 2020. Introducing a new dataset for event detection in cybersecurity texts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2005. ACE 2005 multilingual training corpus. In *Technical report, Linguistic Data Consortium.*

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A Massive General Domain Event Detection Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Miguel Won, Patricia Murrieta-Flores, and Bruno Martins. 2018. Ensemble named entity recognition (NER): evaluating NER tools in the identification of place names in historical corpora. *Frontiers in Digital Humanities*, 5:2.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL).*

Yi Yang and Jacob Eisenstein. 2016. Part-of-speech tagging for historical English. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Type	Description	Examples
LAW Propose	A PROPOSE event occurs when an actor (Agent) introduces a bill, proposition, or treaty which benefits a group of people (Beneficial).	Below we give the salient points of the bill of an entertainment recently **given** in the interest of a certain church about to be organized in a certain town in New Jersey. The bill **introduced** in Congress last week by the congressman from North Carolina , to abolish the 15th amendment. It 's only effect will be to create support for the bill of congressman Crumpacker which **proposes** a reduction of representation in those States
LAW Approve	An APPROVE event occurs when a bill or order (Object) is passed by either the head of the government or a representative committee (Agent).	Be it **enacted** by the General Assembly of Maryland. Vermont has **passed** her Liberty Bill , New York has under discussion , and Massachusetts will soon report and **pass** her Act . But it is said that for the Government to **adopt** the abolition policy, would involve the loss of the support of the Union men of the Border Slave States.
LAW Repeal	A REPEAL event occurs when an active law (Object) is completely repealed by a state actor (Agent).	... that I determined to revoke the act of the Federal Constituent Assembly , whereby Slavery was **abolished**. Even the New York Tribune protests against making this war for the **destruction** of slavery, and insists that such a war would alienate a large body of the Northern people at present who adhere to the Government in the prosecution of the war. They want to se the Government march a powerful array into the traitorous States, proclaim liberty to every slave, and **wipe out** the last vestige of that barbarous system from the land ...
CONFLICT Protest	A PROTEST event occurs when people (Agent) come into a public area to demand some action. PROTEST events include, but are not limited to, protests, sit-ins, and riots as the result of a previous protest.	Almost simultaneously with the appearance of the minstrels there arose from every kennel in the neighborhood timely protest **barked** forth vigorously by a hundred curs, who, in common with their masters, cursed their common luck. It has attempted to supplant Government with anarchy, and the fury of a brutal **mob** for the beneficent operation of law, and the legally appointed law-makers. ...while the majority of the men were absent at a public **demonstration** at Myrtle-avenue Park, in another part of the city.
CONFLICT Attack	An ATTACK event occurs when a person or a organization (Agent) performs an violent act causing harm or damage to another person or organization (Patient).	Make the slave first, midst, and last Follow no longer the partial and side issues; **strike** for the abolition of slavery. . . . and hovering about Williamsport in an unaccountable manner - while the rebel troops are **burning , destroying** , pressing loyal men into service , or driving them from the houses they hoped to possess , and the wheat-fields they expected to reap , under the protecting folds of the Federal flag . The States which **rebelled** , after having been most thoroughly **whipped** in a great war , came back into the Union upon their promises to abide by the Constitution and Laws of the same .
CONFLICT Injure-Die	A Injure-Die event is defined as a death or wound of a person (Patient) which is the result of a violence event by another person (Agent).	The life of loyal men are being **sacrificed** by scores, and will, by and by, be **sacrificed** by thousands. Why should the nation **pour out its blood** and lavish its treasure by the million, consent to protect and preserve the guilty cause of all its troubles? Our loss is estimated at two hundred **killed** and **wounded**.

Table 7: Event types with their descriptions and examples in the BRAD dataset (to be continued in Table 8). Event trigger words are shown in bold.

Type	Description	Examples
CONFLICT Other	Conflict Other events are reserved for events that are related to conflicts, but not classified as one of the conflict event types above, including declaring war, threatening someone, forming an army, a movement, and a march.	Let the slaves and free colored people be **called into service**, and **formed** into a liberating army, to march into the South ... Their efforts in this direction have been crowned by entire **success**. He had **called** loud and earnestly upon the Government for reinforcements; but the Government was practically deaf to the **call**, and left him and his brave companions either to perform a miracle, or to be completely overwhelmed by superior numbers.
JUSTICE Arrest-Jail	An Arrest-Jail Event occurs when the movement of a person (Patient) is constrained by a state actor (Agent).	It appears that he obtained his information direct from German where a supposed agent of the company had been **arrested**, having in his possession incriminating documents. It is said to be possible to **imprison** a man for debt in Massachusetts. He **put her in jail** at Eastville and she stayed there for some time .
JUSTICE Sentence	A SENTENCE Event takes place when a punishment for a person or an organization (Patient) is issued by a state actor (Agent).	... and any person so offending shall be guilty of a felony, and shall, on conviction, be **sentenced to confinement** in the penitentiary of this State, for a period not less than ten nor more than twenty years from the time of **sentence** pronounced on such person. If any slave or servant be convicted or any crime the punishment whereof may be death or **confinement** in the penitentiary ..., but that it has been promptly put down and the guilty parties summarily **punished**.
JUSTICE Execute	An EXECUTE Event occurs when the life of a person (Patient) is taken by a state actor (Agent).	Hector Grant James Horney, and Esther Anderson, white servants, were **executed** at Chester, Kent county. All these , if the demand of the Administration and its friends is gratified , are to be **hanged** ; for the **punishment** of treason by our law is **death** , ... He made some confessions, and managed finally to escape, but was arrested, taken to El Dorado, tried, and **shot** - not, however, by regular process.
HUMANITY Deprive	An DEPRIVE Event occurs when someone's right (Patient) is taken away, disrespect, or discouraged in any form of expression including but not limited to law, action, and statement.	We thank Dr. CROFTS for the assurance of his sympathy, and hope often to receive his earnest words in behalf of our **enslaved** people. Before the **slaved** is freed, this and a hundred other plans will be critically canvassed , and the discussion of each will elicit some truth . ... shall the four millions slaves , now **robbed** of all their rights, and **degraded** to a level with brute beast...
HUMANITY Endow	An ENDOW Event occurs when someone's right is enriched or appreciated in any form of expression including but not limited to law, action, action and statement.	And as for lynching - let all the officers of the law, with all the powers of the law, **defend** the rights and life of every prisoner. Before the slaved is **freed** , this and a hundred other plans will be critically canvassed They are going into every community which **offers freedom** and **protection** to their citizens , where law is justly administered and where the rights of man are **respected** ; and there are many such sections in this country ; there will be the future homes of the Negroes .

Table 8: Event types with their descriptions and examples in the BRAD dataset. Event trigger words are shown in bold.

Event	Argument Role	Entity Type	Description	Examples
LAW Propose	Agent	PER ORG	The person or organization who proposes the law	The resolutions were **proposed** by the gentleman from Ohio. Slavery has been **brought** into the House.
	Beneficiary	PER ORG	The person or organization who benefits from the proposal	
	Object	DOC	The proposed law	
	Time	TIME	When the proposal takes place	
	Location	GPE	Where the proposal takes place	
LAW Approve	Agent	PER ORG	The person or organization who approves the law	Freedom to the slave should now be **proclaimed** from the Capitol. The act was duly **approved** by the Executive, published, and announced to the civilized words.
	Beneficiary	PER ORG	The person or organization who benefit from this approval	
	Object	DOC	The approved law	
	Time	TIME	When the approval takes place	
	Location	GPE	Where the approval takes place	
LAW Repeal	Agent	PER ORG	The person or organization who repeals the law	They shall **strike down** Slavery. ... we can not see why the institution of private property was to be **abolished**.
	Beneficiary	PER ORG	The person or organization who benefits from this repeal	
	Object	DOC	The repealed law	
	Time	TIME	When the repeal takes place	
	Location	GPE	Where the repeal takes place	
CONFLICT Protest	Agent	PER ORG	The person or organization who protests	The red cap was **paraded** through Cape Haytien. The men were present at a public **demonstration** in Brooklyn.
	Patient	ORG	The organization that the agent protest against	
	Time	TIME	When the protest takes place	
	Location	GPE	Where the protest takes place	
CONFLICT Attack	Agent	PER ORG	The attacking person or organization	Fremont is **scouring** the rebels beyond the borders of Missouri. On Wednesday morning the rebels prepared to **storm** our works in Plymouth.
	Patient	PER ORG	The target of the attack	
	Time	TIME	When the attack takes place	
	Location	GPE	Where the attack takes place	
CONFLICT Injure-Die	Agent	PER ORG	The person or organization who attempts to attack or kill	They **cut** men **in half**, and pieces from exploded shells, **killed** and **wounded** several. Most of the negroes, we regret to hear, are said to have been **massacred**.
	Patient	PER ORG	The person or organization who is injured or dead	
	Time	TIME	When the injury/death takes place	
	Location	GPE	Where the injury/death takes place	

Table 9: Descriptions and examples of argument roles for each event type in the BRAD dataset (to be continued in Table 10). Event triggers are bold and underlined. Arguments are highlighted using colors that match with their roles.

Event	**Argument Role**	**Entity Type**	**Description**	**Examples**
CONFLICT Other	Agent	PER ORG	The acting person or organization	The poor men of the South have been **pressed into the army** to fight the battle of slavery.
	Patient	PER ORG	The person or organization who is the object of the act	
	Time	TIME	When the action takes place	
	Location	GPE	Where the action takes place	
JUSTICE Arrest-Jail	Agent	PER ORG	The arresting agent or jailer	A few weeks ago, a man named Hancock was **arrested** in Union county, Arkansas. Several free colored men were **captured** with the rebels in Fort Fisher.
	Patient	PER	The person who is arrested	
	Time	TIME	When the arrest/jail takes place	
	Location	GPE	Where the arrest/jail takes place	
JUSTICE Sentence	Agent	PER ORG	The judge or court	It has been represented there are **confined** in the Government jail forty-five prisoners, who are not charged with crime, but are represented as being slaves
	Patient	PER ORG	The person who is sentenced	
	Time	TIME	When the sentencing takes place	
	Location	GPE	Where the sentencing takes place	
JUSTICE Execute	Agent	PER ORG	The person/organization who orders or carry out the execution	A man by the name of Martin. was tried in El Dorado on a similar charge and **hanged**. They seized him, and being then convinced of his guilt, **shot** him in the woods
	Patient	PER	The person who is executed	
	Time	TIME	When the execution takes place	
	Location	GPE	Where the execution takes place	
HUMANITY Endow	Agent	PER ORG	The person/organization who endows the patient	It is our right to **liberate** the slaves of an enemy. ... and **give freedom** to persons held to labor in the slave states.
	Patient	PER	The person who is endowed	
	Time	TIME	When the endowment takes place	
	Location	GPE	Where the endowment takes place	
HUMANITY Deprive	Agent	PER ORG	The person/organization who deprives the patient	They have **outraged**, and **robbed**, and **murdered** our peaceful citizens. ... by the atrocities of the rebels **delivering back into bondage** thousands of slaves .
	Patient	PER	The person who is deprived	
	Time	TIME	When the deprivation takes place	
	Location	GPE	Where the deprivation takes place	

Table 10: Descriptions and examples of argument roles for each event type in the BRAD dataset. Event triggers are bold and underlined. Arguments are highlighted using colors that match with their roles.

Zero-shot Medical Entity Retrieval without Annotation: Learning From Rich Knowledge Graph Semantics

Luyang Kong
Amazon AI
`luyankon@amazon.com`

Christopher Winestock
Amazon AI
`winestoc@amazon.com`

Parminder Bhatia
Amazon AI
`parmib@amazon.com`

Abstract

Medical entity retrieval is an integral component for understanding and communicating information across various health systems. Current approaches tend to work well on specific medical domains but generalize poorly to unseen sub-specialties. This is of increasing concern under a public health crisis as new medical conditions and drug treatments come to light frequently. Zero-shot retrieval is challenging due to the high degree of ambiguity and variability in medical corpora, making it difficult to build an accurate similarity measure between mentions and concepts. Medical knowledge graphs (KG), however, contain rich semantics including large numbers of synonyms as well as its curated graphical structures. To take advantage of this valuable information, we propose a suite of learning tasks designed for training efficient zero-shot entity retrieval models. Without requiring any human annotation, our knowledge graph enriched architecture significantly outperforms common zero-shot benchmarks including BM25 and Clinical BERT with 7% to 30% higher recall across multiple major medical ontologies, such as UMLS, SNOMED and ICD-10.

1 Introduction

Entity retrieval is the task of linking mentions of named entities to concepts in a curated knowledge graph (KG). This allows medical researchers and clinicians to search medical literature easily using standardized codes and terms to improve patient care. Training an effective entity retrieval system often requires high quality annotations, which are expensive and slow to produce in the medical domain. It is therefore not feasible to annotate enough data to cover the millions of concepts in a medical KG, and difficult to adapt quickly enough to those newly appeared medical conditions and drug treatments under a public health crisis. Hence, a robust medical entity retrieval system is expected to have decent performance in a zero-shot scenario.

Zero-shot retrieval is challenging due to the complexity of medical corpora - large numbers of ambiguous terms, copious acronyms and synonymous terms. It is difficult to build an accurate similarity measure which can detect the true relatedness between a mention and a concept even when their surface forms differ greatly.

Early entity retrieval systems use string matching methods such as exact match, approximate match (Hanisch et al., 2005) and weighted keyword match e.g. BM25 (Wang et al., 2020). Although annotated training data is not required, such systems typically lack the ability to handle synonyms and paraphrases with large surface form differences. In recent years, large scale pretraining (Devlin et al., 2019) has been widely adopted in the medical domain such as Clinical BERT (Alsentzer et al., 2019) and BioBERT (Lee et al., 2019). Agarwal et al. (2019) also integrates graph structure information during pretraining. Most of them, however, require a finetuning step on annotated training data (Wu et al., 2020) before being applied to entity retrieval.

As an alternative to manually annotating a corpus, the rich semantics inside a KG itself can be utilized (Chang et al., 2020). One important entry is the **synonym**, whereby two medical terms may be used interchangeably. In addition, the **graphical structure** of a KG contains information on how concepts are related to each other and so can be used as another valuable resource for building an effective similarity measure. We therefore design *synonym-based tasks* and *graph-based tasks* to mine a medical KG. Trained with our proposed tasks, a simple Siamese architecture significantly outperforms common zero-shot benchmarks across multiple major medical ontologies including UMLS, SNOMED and ICD10.

Our contributions are as follows. (1) We pro-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2401–2405
August 1–6, 2021.

pose a framework which allows the information in medical KGs to be incorporated into entity retrieval models, thereby enabling robust zero-shot performance without the need of human annotations. (2) We apply the framework to major medical ontologies and conduct extensive experiments to establish the effectiveness of our framework. (3) When annotations are available, we show that the proposed framework can be easily plugged into an existing supervised approach and in so doing, deliver consistent improvements.

2 Formulation

Entity retrieval. Given a *mention* $\boldsymbol{m}$ and a *concept* $\boldsymbol{c} \in KG = \{c_1, c_2, ..., c_n\}$, the goal is to learn a similarity measurement $S(\boldsymbol{m}, \boldsymbol{c})$, so that the most relevant concept is assigned the highest score. A concept is also referred to as a *node* in a KG. We use them interchangeably below.

Zero-shot entity retrieval. We examine two zero-shot scenarios: 1) *zero-shot on mentions only*, which assumes unseen mentions but allows seen concepts at test time. 2) *zero-shot on mentions and concepts*, which assumes both to be unseen at test time.

3 Model Architecture

Siamese architecture. Mention $\boldsymbol{m}$ and concept $\boldsymbol{c}$ are firstly embedded into vectors, using a shared function T: $\vec{e_m} = T(\boldsymbol{m}), \vec{e_c} = T(\boldsymbol{c})$. T is also referred to as an encoder, for which we use the Transformer (Vaswani et al., 2017) encoder in this work. Similarity between a mention and a concept is then measured as the inner product: $S(\boldsymbol{m}, \boldsymbol{c}) = \langle \vec{e_m}, \vec{e_c} \rangle$.

Optimization. Assume model parameter is θ. We use in-batch negatives for optimization. Loss function for a batch of size B is defined as mean negative log likelihood:

$$L = -\frac{1}{B} \sum_{i=1}^{B} \log(P((\boldsymbol{m_i}, \boldsymbol{c_i})|\theta))$$

where the conditional probability of each mention-concept pair $(\boldsymbol{m_i}, \boldsymbol{c_i})$ in the batch is modeled as a softmax:

$$P((\boldsymbol{m_i}, \boldsymbol{c_i})|\theta) = \frac{\exp(S_\theta(\boldsymbol{m_i}, \boldsymbol{c_i}))}{\sum_{j=1}^{B} \exp(S_\theta(\boldsymbol{m_j}, \boldsymbol{c_j}))}$$

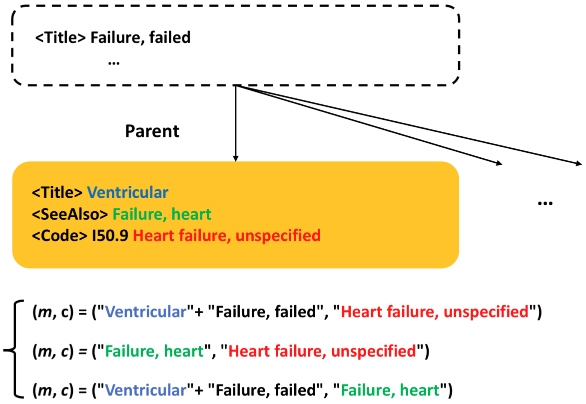

Figure 1: ICD-10 synonym-based task defined at an example node

4 Learning Task

We design our learning tasks by constructing mention-concept pairs $(\boldsymbol{m}, \boldsymbol{c})$. The goal is to capture multiple layers of semantics from a KG by leveraging its unique structure. Since each structure implies its own measure of similarity, we design learning tasks by finding very similar or closely related textual descriptions and use them to construct $(\boldsymbol{m}, \boldsymbol{c})$ pairs. We define two major types of tasks: *synonym-based* tasks and *graph-based* tasks. These are illustrated below for three major medical KG: ICD-10, SNOMED and UMLS.

4.1 ICD-10

The 10th version of the International Statistical Classification of Diseases, Clinical Modification (ICD-10) is one of the most widely used terminology systems for medical conditions. It contains over 69K concepts, organized in a tree structure of parent-child relationships.

Synonym-based task. In ICD-10, a child node is a more specific medical condition compared to its parent (e.g. `R07.9 Chest pain, unspecified` is a child of `R52 Pain, unspecified`). Each node N_i has three sections: The *Title* section contains a sub-specifier (e.g. `Chest`) of the title of the parent (e.g. `Pain`), therefore their concatenation gives the full concept description (e.g. `Chest Pain`). We denote it by $N_i^{TitleConcatenation}$. The *Code* section contains an ICD-10 code and its formal medical definition, denoted by $N_i^{CodeDescription}$. The *SeeAlso* section contains a similar concept, denoted by $N_i^{SeeAlso}$.

These three sections describe the same medical condition with different surface forms, therefore we define the ICD-10 synonym-based task as:

$$\boldsymbol{m} = N_i^L, \boldsymbol{c} = N_i^R$$

$$N_i^L, N_i^R \in \{N_i^{TitleConcatenation},$$

$$N_i^{CodeDescription}, N_i^{SeeAlso}\}, L \neq R$$

We illustrate it with an example in Figure 1.

Graph-based task. To incorporate the semantics of parent-child relationships into learning, we define ICD-10 graph-based task as:

$$\boldsymbol{m} = N_i^{CodeDescription}, \boldsymbol{c} = N_j^{CodeDescription}$$

$$N_i.is\_parent(N_j)$$

4.2 SNOMED

Systematized Nomenclature of Medicine – Clinical Terms (SNOMED) is a standardized clinical terminology used for the electronic exchange of clinical health information with over 360K active concepts.

Synonym-based task. Each node N_i in SNOMED has multiple synonymous descriptions $\{l_i^1, l_i^2, ..., l_i^d\}$, with l_i^1 as the main description. We therefore define SNOMED synonym-based task as:

$$\boldsymbol{m} = l_i^p, \boldsymbol{c} = l_i^q, p > q$$

$\frac{d*(d-1)}{2}$ unique $(\boldsymbol{m}, \boldsymbol{c})$ pairs are constructed at each node.

Graph-based task. SNOMED is a directed graph with 107 possible relationship types (e.g. `is_a`, `finding_site`, `relative_to`). A direct connection between two nodes is likely to imply a certain degree of similarity, thus we define the SNOMED graph-based task as:

$$\boldsymbol{m} = l_i^1, \boldsymbol{c} = l_j^1$$

$$N_i.is\_connected(N_j)$$

4.3 UMLS

The Unified Medical Language System (UMLS) is a compendium of a large number of curated biomedical vocabularies with over 1MM concepts. UMLS has almost the same structure as SNOMED, therefore we define the **synonym-based task** and **graph-based task** in a similar fashion to that of SNOMED.

For each task mentioned above, the $(\boldsymbol{m}, \boldsymbol{c})$ pairs generated at each node are combined and split into train and dev in a 80:20 ratio. We also define a **comb task**, where all the tasks are firstly down-sampled to equal sizes and then combined. A summary can be found in Table 1.

KG	Task Type	Train	Dev
ICD-10	syn	113K	28K
	graph	33K	8K
SNOMED	syn	1.4M	374K
	graph	955K	238K
UMLS	syn	27M	7M
	graph	7M	2M
Comb *(by down-sampling)*		198K	488K

Table 1: **Task Description**: Number of $(\boldsymbol{m}, \boldsymbol{c})$ pairs in train and dev for all tasks.

Dataset	Split	KG	Test size
MedM.	-	UMLS	66,572
COMETA	SG	SNOMED	4,350
	SS		4,369
	ZG		3,995
	ZS		4,283
3DNotes	ICD	ICD-10	5,742
	SN	SNOMED	7,521

Table 2: **Test Set Size.**

5 Datasets

We include three datasets in zero-shot evaluations. **MedMention** (Mohan and Li, 2019) is a publicly available corpus of 4,392 PubMed[1] abstracts with biomedical entities annotated with UMLS concepts. **COMETA** (Basaldella et al., 2020) is one of the largest public corpora of social media data with SNOMED annotations. It provides four train, dev, test splits: Stratified-General (SG), Stratified-Specific (SS), Zeroshot-General (ZG), Zeroshot-Specific (ZS). We also use a de-identified corpus of dictated doctor's notes named **3DNotes**(Zhu et al., 2020). It has two sets of annotations: one with ICD-10 (ICD split), another with SNOMED (SN split). The annotation follows the i2b2 challenge (Uzuner et al., 2011) guidelines.

Zero-shot performance is evaluated on the corresponding test sets. We report sizes of the test sets in Table 2.

6 Experiments

6.1 Experimental Settings

We train a model using each task and evaluate them across all test sets. Performance is compared against two common zero-shot entity retrieval benchmarks: BM25, and BERT encoder followed by inner-product similarity. For the latter, we tested multiple pre-trained versions including $\text{BERT}_{\text{base}}$, Clinical BERT and BioBERT.

---

[1] https://www.ncbi.nlm.nih.gov/pmc/

Dataset	Split	KG	BM25	Clinical BERT	Siamese architecture trained with KG learning tasks (**ours**)						
					ICD-10		SNOMED		UMLS		Comb
					Syn	Graph	Syn	Graph	Syn	Graph	
MedM.	-	UMLS	.04(.17)	.10(.30)	.31(.58)	.31(.56)	.32(.55)	.33(.61)	**.32(.53)**	**.30(.57)**	**.32(.60)**
COMETA	SG	SNOMED	.02(.10)	.01(.06)	.30(.52)	.30(.48)	**.43(.65)**	**.37(.58)**	.33(.50)	.32(.54)	**.37(.58)**
	SS		.02(.11)	.01(.06)	.28(.51)	.28(.47)	**.41(.62)**	**.36(.56)**	.31(.48)	.31(.52)	**.35(.56)**
	ZG		.02(.12)	.01(.07)	.32(.57)	.32(.54)	**.47(.71)**	**.39(.61)**	.36(.55)	.33(.57)	**.40(.62)**
	ZS		.02(.10)	.01(.07)	.30(.52)	.29(.47)	**.40(.64)**	**.35(.57)**	.31(.49)	.29(.53)	**.35(.57)**
3DNotes	ICD	ICD-10	.05(.22)	.11(.17)	**.28(.54)**	**.23(.46)**	.20(.45)	.20(.52)	.18(.39)	.21(.53)	**.30(.54)**
	SN	SNOMED	.07(.20)	.01(.05)	.20(.50)	.18(.45)	**.38(.63)**	**.25(.61)**	.25(.49)	.29(.55)	**.34(.59)**

Table 3: Retrieval performance R@1(25). Siamese architecture trained with our tasks are shown to significantly outperform benchmarks. Evaluation for *zero-shot on mentions only* is highlighted in **bold** the rest belongs to *zero-shot on mentions and concepts*. The former enjoys a bigger gain as expected.

Mention	Gold Concept	Syn	Graph
shortness of breath	dyspnea (finding)	✓	✗
GI hemorrhage	gastrointestinal hemorrhage (disorder)	✓	✗
coronary structure	coronary artery (body structure)	✗	✓
heart	heart structure (body structure)	✗	✓

Table 4: Prediction error of the model trained with SNOMED tasks evaluated on 3DNotes-SN.

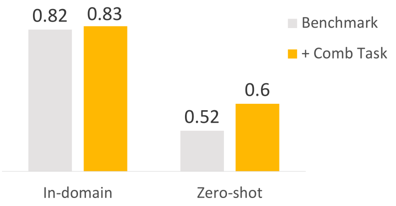

Figure 2: R@25. Using the *comb* task as an auxiliary to the primary supervised loss, the model gains 1% in-domain (3DNotes-SN) improvement , 8% zero-shot (COMETA-ZS) improvement.

Hyperparameters. For our Siamese architecture, the transformer encoder is initialized with $\text{BERT}_{\text{base}}$. We use the BertAdam optimizer with a batch size of 128, the initial learning rate of 3×10^{-5}, warm-up ratio of 0.02, max epochs of 50, followed by a linear learning rate decay.

Evaluation metrics. Top-k retrieval recalls (R@1, R@25) are used as metrics. We also assume that each mention has a valid gold concept in the KG.

6.2 Results

We report overall results in Table 3. Clinical BERT consistently outperforms the other pre-trained counterparts, which are therefore omitted. For evaluations of *zero-shot on mentions only* (e.g. UMLS tasks evaluated on MedMention which is UMLS annotated), we observe 12% to 45% gain for R@1 compared to benchmarks. For evaluations of *zero-shot on mentions and concepts* (e.g. UMLS tasks evaluated on COMETA which is SNOMED annotated), 7% to 30% higher R@1 is observed. *Comb* task has the most balanced performance gains across all datasets.

6.3 Analysis and Discussion

Task comparison. To further understand the difference between synonym-based tasks and graph-based tasks, we illustrate qualitative examples in Table 4. A model trained using the synonym task makes better predictions for scenarios involving medical synonyms and acronym (lines 1, 2). A model trained using the graph task performs better when mention and concept have an `is_a` relationship (lines 3, 4).

Auxiliary task. When annotations are available, our learning tasks can be used as an auxiliary to the primary loss. Using the 3DNotes-SN's annotated training set to train the primary supervised task, we set the *comb* task as its auxiliary counterpart by summing the losses. We evaluate zero-shot performance on COMETA-ZS. We observe an 8% increase in R@25, illustrated in Fig. 2. Since most annotations cover no more than a couple thousands concepts, which is a tiny portion of a typical medical KG's size, this demonstrates the generalizing capacity of our approach on the vast majority of unseen concepts.

Private KG. In practice, if the target medical ontology is a private KG (Wise et al., 2020; Bhatia et al., 2020), one can also consider customizing the learning tasks that follow the synonym and graph-based frameworks outlined in this work to bring greater gains.

7 Conclusion

We present a framework for allowing entity retrieval models to mine rich semantics from a medical KG. We show its effectiveness in zero-shot settings through extensive experiments. In addition, we demonstrate the ease with which the framework can be adapted to serve as an auxiliary task when annotations are available. Future research should explore more fine-grained approaches to combine tasks.

References

Khushbu Agarwal, Tome Eftimov, Raghavendra Addanki, Sutana Choudhury, Suzanne Tamang, and Robert Rallo. 2019. Snomed2vec: Poincare and random walk embeddings of a clinical knowledge base for healthcare analytics. In *KDD Workshop on Applied Data Science for Healthcare: Bridging the Gap between Data and Knowledge*.

Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. Publicly available clinical bert embeddings.

Marco Basaldella, Fangyu Liu, Ehsan Shareghi, and Nigel Collier. 2020. COMETA: A corpus for medical entity linking in the social media. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3122—3137, Melbourne, Australia. Association for Computational Linguistics.

Parminder Bhatia, Lan Liu, Kristjan Arumae, Nima Pourdamghani, Suyog Deshpande, Ben Snively, Mona Mona, Colby Wise, George Price, Shyam Ramaswamy, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, Bing Xiang, and Taha Kass-Hout. 2020. Aws cord-19 search: A neural search engine for covid-19 literature.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Daniel Hanisch, Katrin Fundel, Heinz-Theodor Mevissen, Ralf Zimmer, and Juliane Fluck. 2005. Prominer: rule-based protein and gene entity recognition. *BMC Bioinformatics*, 6:1:S14.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Sunil Mohan and Donghui Li. 2019. Medmentions: A large biomedical corpus annotated with UMLS concepts. *CoRR*, abs/1902.09476.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Qiong Wang, Zongcheng Ji, Jingqi Wang, Stephen Wu, Weiyan Lin, Wenzhen Li, Li Ke, Guohong Xiao, Qing Jiang, Hua Xu, and Yi Zhou. 2020. A study of entity-linking methods for normalizing chinese diagnosis and procedure terms to icd codes. *Journal of Biomedical Informatics*, 105:103418.

Colby Wise, Vassilis N. Ioannidis, Miguel Romero Calvo, Xiang Song, George Price, Ninad Kulkarni, Ryan Brand, Parminder Bhatia, and George Karypis. 2020. Covid-19 knowledge graph: Accelerating information retrieval and discovery for scientific literature.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Ming Zhu, Busra Celikkaya, Parminder Bhatia, and Chandan K. Reddy. 2020. Latte: Latent type modeling for biomedical entity linking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9757–9764.

CONDA: a CONtextual Dual-Annotated dataset for in-game toxicity understanding and detection

Henry Weld **Guanghao Huang** **Jean Lee** **Tongshu Zhang** **Kunze Wang**
Xinghong Guo **Siqu Long** **Josiah Poon** **Soyeon Caren Han***
School of Computer Science, The University of Sydney, NSW, Australia
{hgua3108,tzha6458,kwan4418,xguo2796,slon6753}@uni.sydney.edu.au
{henry.weld,jean.lee,josiah.poon,caren.han}@sydney.edu.au

Abstract

Traditional toxicity detection models have focused on the single utterance level without deeper understanding of context. We introduce CONDA, a new dataset for in-game toxic language detection enabling joint intent classification and slot filling analysis, which is the core task of Natural Language Understanding (NLU). The dataset consists of 45K utterances from 12K conversations from the chat logs of 1.9K completed *Dota 2* matches. We propose a robust dual semantic-level toxicity framework, which handles utterance and token-level patterns, and rich contextual chatting history. Accompanying the dataset is a thorough in-game toxicity analysis, which provides comprehensive understanding of context at utterance, token, and dual levels. Inspired by NLU, we also apply its metrics to the toxicity detection tasks for assessing toxicity and game-specific aspects. We evaluate strong NLU models on CONDA, providing fine-grained results for different intent classes and slot classes. Furthermore, we examine the coverage of toxicity nature in our dataset by comparing it with other toxicity datasets.[1]

1 Introduction

As the popularity of multi-player online games has grown, the phenomenon of in-game toxic behavior has taken root within them. Toxic behavior is strongly present in recent online games and is problematic to the gaming industry (Adinolf and Turkay, 2018). For instance, 74% of US players of such games report harassment with 65% experiencing severe harassment. (ADL, 2019).

In the past few years, Natural Language Processing (NLP) researchers have proposed several online game/community toxicity analysis frameworks (Kwak et al., 2015; Murnion et al., 2018; Wang et al., 2020) and datasets (Märtens et al., 2015; Stoop et al., 2019). However, existing datasets (1) focus only on the single utterance level without deeper understanding of context in the whole conversation/chat, and (2) do not explicitly use semantic clues from the words within the utterance.

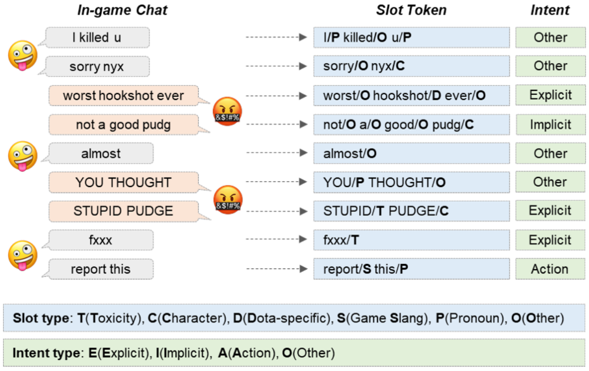

Figure 1: An example intent/slot annotation from the CONDA (CONtextual Dual-Annotated) dataset.

The chat in online games and communities is similar in nature to spoken language, an area studied by Natural Language Understanding (NLU). NLU research aims to best represent human communication by extracting semantic structure in the form of intent and slot analysis. Intent detection is the classification of the desired outcome of an utterance (or sentence), and slot filling is the labeling of each token (or word) in the utterance with the type of semantic information it carries. In recent literature, these two tasks are trained jointly to capture synergies between them, and these jointly trained models give better results (Zhang et al., 2019b). Furthermore, researchers have made available joint task datasets that contain the context of a multi-turn conversation (Budzianowski et al., 2018; Schuster et al., 2019)

Inspired by this NLU research progress, we

---

*Corresponding author (caren.han@sydney.edu.au)

[1] The dataset and lexicons are available at https://github.com/usydnlp.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2406–2416
August 1–6, 2021. ©2021 Association for Computational Linguistics

propose CONDA, an in-game toxicity detection dataset, with a robust dual-level annotation which enables intent detection and slot filling. Our dataset consists of 45k utterances from the chat logs of 1.9k *Dota 2* matches, labeled with 4 intent classes and 6 slot classes to address toxicity and the game-specific vocabulary. Figure 1 illustrates an example of CONDA including raw data (in-game chat) and processed data with slot and intent labels. In order to enable the dual semantic-level framework, we conduct lexicon-based automation for token-level data and human annotation for utterance-level data.

We investigate the CONDA dataset through an in-depth analysis. The large portion of game-specific classes in the dual levels enables the dataset to be more sophisticated in detecting toxicity in games. The combination of each intent with each slot class shows that dual annotation can help determine toxicity from gamer slang when used in both toxic and non-toxic situations. We also find more toxic utterances appear pre-game and post-game rather than during the games, especially peaking post-game due to the chat for post-victory celebration and recrimination.

We provide five strong baseline NLU models and compare the toxicity detection performance over our dataset. For evaluation, we apply four NLU metrics to assess performance in toxicity and game specific aspects. Results vary across models, indicating a challenge for improvement. Furthermore, we perform a transfer learning experiment with existing toxicity datasets. We find that the nature of toxicity in our dataset can generalize to other proposed taxonomies, including hatefulness, sexism and racism. Beyond this commonality, our experiment illustrates that CONDA is distinguished from other toxicity datasets due to game-specific characteristics. This paper then makes the following contributions:

- To the best of our knowledge, this is the first attempt to build a toxicity detection dataset with joint Natural Language Understanding aspects of intent classification and slot filling;
- We propose a robust dual semantic-level toxicity framework, which handles utterance and token-level patterns with rich in-game chatting history;
- We formalise NLU metrics for toxicity detection, evaluate strong NLU models on our dataset, and further conduct transfer learning experiments with other toxicity datasets.

2 Related Work

Toxicity Datasets in Online Games In multiplayer online games, prior research focused on analysis of anti-social or disruptive behavior, so-called toxic behavior (Blackburn and Kwak, 2014; de Mesquita Neto and Becker, 2018) including cyberbullying (Kwak et al., 2015) and griefing (Murnion et al., 2018). Although these terms contain similar elements, a single definition of toxic behavior is yet to emerge. Some studies have conducted data annotation using pre-defined lexicon categories (Märtens et al., 2015) or toxic player information (Stoop et al., 2019). These annotation methods are not robust enough to handle unlabelled toxicity words or unreported toxic players.

Toxicity Datasets in Online Community An extensive body of work has focused on datasets to detect toxicity including hate speech (Waseem and Hovy, 2016; Davidson et al., 2017; ElSherief et al., 2018) and abusive language (Nobata et al., 2016; Founta et al., 2018). However, the majority of toxicity datasets do not consider the context of a conversation, instead simply analysing a single utterance. Even if a model uses contextual information (Gao and Huang, 2017), it is limited to meta-information (e.g. news title or user name) which is not sufficient to understand a conversation. In our research, context is defined as linguistic contextual information, particularly previous single or multiple utterances. Along similar lines, recent studies have focused on conversation aiming to discover warning signals (Zhang et al., 2018), to generate intervention responses (Qian et al., 2019), or to measure the importance of context (Pavlopoulos et al., 2020). Existing toxicity datasets mainly focus on annotating at utterance-level, whereas ours conducts a dual-level annotation at utterance and token-level, while also providing a conversation history (see Table 1). These extra features are what distinguish CONDA.

NLU Datasets and Models In-game chat has similar characteristics to multi-turn dialogue in NLU. The approaches used in multi-turn dialogue analysis have not yet been observed in toxicity datasets. In NLU, generally, intent classification (IC) is treated as a semantic utterance classification task and slot filling (SF) is treated as a sequential token labelling task (Zhang and Wang, 2016). By conducting a joint model for the two tasks, a synergistic effect can be achieved (Zhang et al.,

Dataset	Approach	Domain	Labels	Conv.
(Märtens et al., 2015)	utterance-level	Game (Dota 2)	toxic, non-toxic	N
(Waseem and Hovy, 2016)	utterance-level	Twitter	racist, sexist, normal	N
(Nobata et al., 2016)	utterance-level	Yahoo News	clean, hate, derogatory, profanity	N
(Davidson et al., 2017)	utterance-level	Twitter	hateful, offensive, neither	N
(Gao and Huang, 2017)	utterance-level	Fox News	hate, non-hate	N
(ElSherief et al., 2018)	utterance-level	Twitter	hate, non-hate / hate instigator, hate target	N
(Founta et al., 2018)	utterance-level	Twitter	offensive, abusive, hateful speech, aggressive, cyberbullying, spam, normal	N
(Zhang et al., 2018)	utterance-level	Wikipedia	toxic, non-toxic	Y
(Stoop et al., 2019)	utterance-level	Game (LoL)	toxic, non-toxic	Y
(Qian et al., 2019)	utterance-level	Gab & Reddit	hate, non-hate	Y
(Pavlopoulos et al., 2020)	utterance-level	Wikipedia	toxic, non-toxic	Y
CONDA (our dataset)	**dual-level (utterance and token)**	Game (Dota 2)	- utterance level (intent): explicit toxicity, implicit toxicity, action, others - token level (slot): toxicity, character, dota-specific, slang, pronoun, other	**Y**

Table 1: Comparison of CONDA with other toxicity datasets (Conv.: Conversation).

2019b). To build multi-turn dialogue datasets, most studies have recruited workers via crowd-sourcing to collect task-oriented dialogues across different domains (e.g. in-car assistant (Eric et al., 2017), navigation and events (Gupta et al., 2018), multi-domains (Budzianowski et al., 2018), personal notifications (Schuster et al., 2019)). Recently, deep learning models have also been extensively studied in order to capture the contextual signals from multiple sequential inputs. (e.g. BiLSTM with attention (Wang et al., 2019), GRU with self-attention and context-fusion (Gupta et al., 2019). The models listed all show an increase in semantic detection performance when the context is included in the analysis.

3 CONDA

3.1 Data Collection

Our annotated dataset, CONDA, is based on the *Defense of the Ancients 2 (Dota 2)* data dump available at Kaggle[2]. *Dota 2* is a multiplayer online game where teams of five players attempt to destroy their opponents' ancient structure. The raw data is compiled from game matches including players, duration, match outcomes, and complete chat logs. In order to curate data, we select 50,000 utterances in complete chat logs from 1,921 matches.

3.2 Data Processing

Our data processing is designed to enable dual annotation, making utterance-level data suitable for human annotators and generating token-level data for lexicon-based automation. The main processes are creation of conversations, restructuring utterances while keeping original context, and generation of tokens.

We generate conversations to give human annotators a context of previous utterances when labelling the current utterance. We identify the beginning of a conversation as the first utterance in the match, or an utterance that occurs greater than 60 seconds after the previous utterance in the match. While the raw data is largely in English, other languages appear occasionally including Russian, Chinese, Spanish, etc. We exclude conversations with chat in non-English.

For the utterance-level data, we maintain the original form such as punctuation and case in order to keep context. In addition, we merge consecutive utterances by a single user within a conversation. These are combined into one utterance with a special token, [SEPA], added to denote the separation point (e.g. *'easiest [SEPA] game [SEPA] of my life'*). For the token-level data, we use contraction restoration (e.g. *'I'm' -> 'I am'*), whitespace tokenise each utterance, retain emoticons, but remove punctuation. This token-level processing is used for lexicon-based automated slot annotation.

Our final CONDA dataset (Table 2) consists of 44,869 utterances and 1,921 matches. We further create a subset, equivalent to about 10% of the full dataset, for a preliminary round of utterance-level annotation.

Dataset Feature	CONDA
Matches	1,921
Conversations	12,152
Utterances	44,869
Avg. utterances per match	23.3

Table 2: CONDA statistics.

---

[2] https://www.kaggle.com/devinanzelmo/dota-2-matches

3.3 Annotation

Dual Aspects Inspired by NLU, we provide a dual-level annotation approach to detect toxicity, which often relies on context. This allows one to find toxic intent even though an utterance does not contain any toxic words, or to determine non-toxic intent even if an utterance has toxic words. For example, Figure 1 shows an utterance *"not a good pudg"*, which does not contain any toxic words. However, considering the previous utterance of *"worst hookshot ever"*, we can identify hidden or implicit toxicity. As an example of the other way around, an utterance of *"happy fuck you day"* contains a toxic word but it is used for cheering after saying *"gg" (good game).*

Token-level Slot Annotation With the processed token-level data, an automated slot labelling is performed. Initially, we create six distinct slot labels: T (**T**oxicity), C (**C**haracter), D (**D**ota-specific), S (game **S**lang), P (**P**ronoun) and O (**O**ther). To construct the **T** lexicon, we combine several toxicity lexicons (see Section 8 Ethics) and remove overlaps. We also use the supplemental data sourced by Märtens et al. (2015) for the game-related lexicons (**C, D** and **S**) and carefully modify it. The **P** lexicon (e.g. *'u', 'ur'*) is constructed by this research because in-game chat is extremely abbreviated. Then, we perform lexicon-based automation by exact matching each lower-cased token against the lexicons. Anything not matching a lexicon is labelled **O**. We contrast this with typical NLU slot labelling where a semantic concept can stretch over a span of words. In comparison to other toxicity datasets, our lexicon-based slot labelling enables deeper understanding of game context.

Utterance-level Intent Annotation Given tokens with slot labelling and utterance-level data, we perform a test run on the subset of utterances using six annotators. Four annotators are game players and two are non-game players. This preliminary round is for fine-tuning annotation policy and analysing annotator agreement to inform final annotation for the full dataset. The annotators manually classified the utterances into four labels: E (**E**xplicit toxicity), I (**I**mplicit toxicity), A (**A**ction) and O (**O**ther). The label details are explained in the annotator instructions.

Annotator Instructions Each annotator was required to consider the earlier conversation, particularly, to detect implicit toxic behavior or to identify non-toxic behavior in the utterance having toxic-labelled tokens. The annotators worked independently of one another. The guidelines for human annotators were as follows:

Explicit toxicity: Typically contains toxic word(s). The intent is to insult or humiliate others, or to make others want to leave the conversation or quit the game. There is no need to consider the context (e.g. *'fuck off'*). May include one or more of the following aspects:

- Strong toxicity - blatant insulting or disrespecting others is obviously seen in the text, normally with severely toxic wording;
- Normal toxicity - impolite, rudely worded and unreasonable comment that insults or humiliates others;
- Cursing others with the intent to insult or humiliate them (e.g. *'noob'*[3]);
- Sexual wording or talk about sex-related behavior;
- Use of negative or hateful words to describe others (e.g. *'useless'*);
- Racist language that is targeted at insulting others (e.g. *'Peruvians', 'fucking russians'*);
- Inflammatory language, insulting others and trying to start a conversational fight.

Implicit toxicity: Hidden toxicity that normally cannot be seen from the text itself. The text might be factual or even positive (e.g. sarcasm). However, based on the utterance or conversation context, the intent of insulting or humiliating others can be inferred. Typically contains no toxic word (e.g. *'u are poor dude'*).

Action: Doesn't belong to I or E, but contains an action such as report, commend, pause, stop, or exit game.

Other: Doesn't belong to I or E or A. May or may not contain toxic words. Includes curses, self-deprecation or any other emotional expression that is NOT targeted at others (e.g. *'kill the fucking helicopter'*).

Findings in Annotation The preliminary round was useful for enabling discussion around annotation. For example, we decided that the token "ez"[4]

---

[3] "Noob" is a slang term for a newcomer, commonly used to insult someone inexperienced in games.

[4] "Ez" is an abbreviation for easy. It is often used to irritate other players in games, indicating "You are just way too easy".

or its variations are an implied slur against the opposition's quality and would generally be part of an **I** label utterance. Similarly, "g" is often a contraction for "go" and would be part of an **A** label utterance. Overall, we observed that the agreement measure for utterance classification was higher for gamer annotators only (Fleiss' kappa = 0.785) versus the whole group (Fleiss' kappa = 0.755). The lower inter-rater agreement in the whole group is because non-gamer annotators have low understanding of the game context and domain-specific language. Therefore, annotation of the whole dataset was performed by gamers only. Based on our annotation guidelines, they collectively manually annotated the utterances for the full dataset.

4 Dataset Analysis

The CONDA dataset consists of 9 columns - match ID, conversation ID, player ID, player slot, chat time, utterance, slot tokens (cleaned tokens with slot labelling), intent class, and slot classes. For example, the utterance *"gg wp"* for *"good game well played"* is shown in the slot tokens column as *"gg (S), wp (S)"*. Each column is further explained in AppendiX A.

Intent	%	Mean L.
E (**E**xplicit)	13.3	6.14
I (**I**mplicit)	6.4	4.16
A (**A**ction)	6.4	4.40
O (**O**ther)	73.9	3.18
Total	**100.0**	**3.71**

Slot	%
T (**T**oxicity)	4.9
C (**C**haracter)	5.4
D (**D**ota-specific)	1.4
S (Game **S**lang)	11.2
P (**P**ronoun)	13.5
O (**O**ther)	63.6
Total	**100.0**

Table 3: Intent labelling statistics (left) and slot labelling statistics (right). % is proportion of the dataset. Mean L. is mean number of tokens after cleaning.

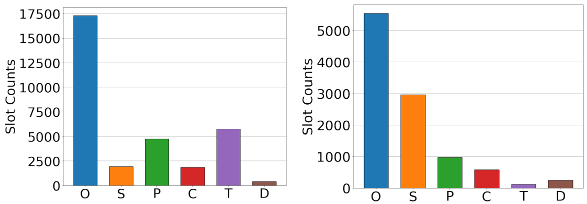

(a) Class 'Explicit'

(b) Class 'Implicit'

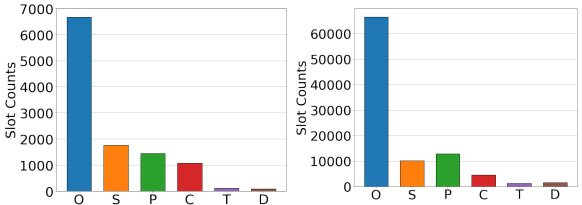

(c) Class 'Action'

(d) Class 'Other'

Figure 2: Slot class distributions for each intent class.

Rank	S	T
1	gg (239)	noob (878)
2	report (237)	fuck (807)
3	ez (191)	fucking (593)
4	mid (169)	shit (546)
5	go (114)	idiot (222)

(a) Class "Explicit"

Rank	S	T
1	ez (1,932)	wtf (32)
2	mid (287)	fucking (11)
3	gg (169)	dead (9)
4	report (47)	hook (6)
5	go (38)	fuck (5)

(b) Class "Implicit"

Rank	S	T
1	report (992)	wtf (16)
2	afk (184)	fucking (11)
3	gg (134)	abuse (10)
4	go (57)	noob (8)
5	wp (37)	shit (4)

(c) Class "Action"

Rank	S	T
1	gg (3,735)	wtf (331)
2	wp (1,115)	dead (89)
3	ggwp (776)	fucking (84)
4	mid (413)	hook (69)
5	go (383)	shit (39)

(d) Class "Other"

Table 4: Top 5 keywords in the S (game Slang) and T (Toxicity) slot classes, for each intent class. The number in brackets is the token count in that combination of classes.

Dual Annotation Proportion Table 3 gives the proportional breakdown of the CONDA dataset by intent and slot labels. Together the toxic utterance classes make up 19.7% of the data, emphasizing their prevalence in game chat. The proportion of the I (6.4%) and A (6.4%) intent classes, together 12.7% of all utterances, shows the more granular non-binary class structure captures an aspect of online games. Additionally, the average length of utterances of the E class (6.14) is greater than for each other class, indicating players strongly emphasize emotional frustration. In the proportion of slot labels, we can see the S (11.2%) class is more than double the C (5.4%) class and 8 times the D (1.4%) class, indicating general gamer slang is used for communication more than terms specific to the game being played. Overall, the large portion of these game-specific classes enables the dataset to be more sophisticated in detecting toxicity in games.

Dual Annotation Distribution To understand the effect of dual annotation on the toxicity context, we look at the distribution of the slot labels within each intent class. As seen in Figure 2b, the I intent class shows the highest proportion of the S slot class among non-O classes. Similarly, Figure 2c shows a relatively high proportion of the S slot class in the A intent class. This suggests that the combination of S slot and intent classes provides useful information because slang performs the function of carrying game-specific context. As a result, we focus more on T and S slot classes joined with other intent classes in order to investigate toxicity natures in games carried out from dual annotation.

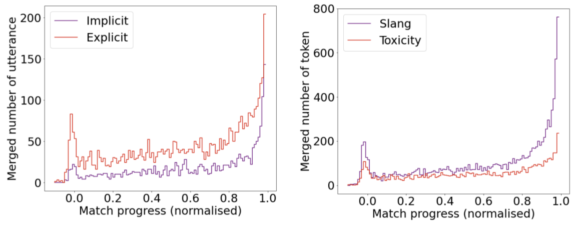

(a) Class "Explicit, Implicit" (b) Class "Toxicity, Slang"

Figure 3: In-game chat histogram for intent (E,I) and slot (T,S) classes. Match progress is bucketed position within a match whose duration is normalised in [0,1], with <0 indicating pregame chat. Merged number of utterance/token is the count of all utterances/tokens from all matches in that match progress bin.

Keywords in Dual Annotation Table 4 shows the top 5 keywords by frequency from the T and S slot classes, for each intent class. In the combinations of S class, we observe the prominent position of "ez" in the E and I intent class. This indicates dual annotation captures toxicity from the slang largely used in games. In addition, "gg" appears in all combinations because it may have some toxicity attached via sarcasm. As dual annotation uses the conversational history, it is able to classify the same utterance in different intents.

Toxicity Analysis Over Time We further analyse in-game chat over time associated with the E, I intent classes and the T, S slot classes. As shown in Figure 3a, more toxic utterances appear pre-game and post-game rather than during the games. In pregame, sometimes players are upset due to the selected hero characters if their desired hero is taken by others, or are stressed by planning game strategies in a limited time. Toxic utterance frequency gradually rises towards the end, and peaks in post-game due to chat for post-victory celebration and recrimination. Interestingly, Figure 3b displays a similar pattern. Particularly, tokens for the S slot class increase sharply to the end, indicating significant amounts of slang are used to celebrate wins or humiliate defeated opponents.

Comparison with Other Datasets In Figure 4, we compare our dataset with other toxicity detection datasets using the metric of relative frequency of toxic utterances of each length. The datasets we compare with are **1) Waseem** (Waseem and Hovy, 2016) which consists of 16.2k tweets binary classified as racism/sexism or other, **2) FoxNews** (Gao and Huang, 2017) which is 1.5k sentences from Fox News discussion threads classified as hateful/non-hateful, and **3) StormfrontWS** (de Gibert et al., 2018) which is 10.7k conversation sentences from white supremacist website Stormfront classified as hate speech/non-hate speech. For this analysis, we merge classes into toxic/non-toxic as required. As an example of the CONDA dataset, we combine E and I intent classes into a toxic class, and A and O into a non-toxic class.

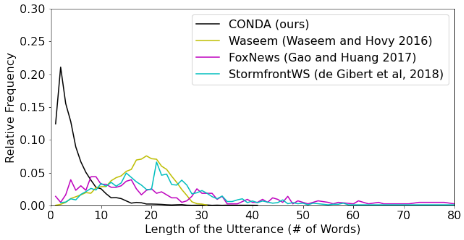

Figure 4: Distribution of toxic utterance length across similar datasets.

The distribution in CONDA is different to the other datasets in that the toxic utterances are shorter. This is due to the terseness of in-game chat during playing, with longer utterances occurring in pre-game and post-game discussion. Waseem has a particular distribution due to the character limit in Twitter (140 characters at the time). FoxNews and StormfrontWS are forums which foster the use of longer sentences.

5 Baseline Experiment

To explore the toxicity detection from an NLU perspective, we selected five baseline NLU models and compared their detection performance over our proposed dataset.

5.1 Data Preparation

We split the data into train/validation/test sets in the proportions of 0.6/0.2/0.2, or in samples 26,921/8,974/8,974. The data passed to the models is the tokenised utterances with punctuation removed, and for training the slot and intent labels.

5.2 Baseline NLU Models

The five NLU models are as follows:

- **RNN-NLU** (Liu and Lane, 2016) is an attention-based bi-directional recurrent neural network model that jointly predicts the current slot and the intent at each time step using shared hidden states and attention.
- **Slot-gated** (Goo et al., 2018) is an attention-based BiLSTM model which builds on sepa-

	Metrics												
Model	**UCA**	**U-F1(E)**	**U-F1(I)**	**U-F1(A)**	**U-F1(O)**	**T-F1**	**T-F1(T)**	**T-F1(S)**	**T-F1(C)**	**T-F1(D)**	**T-F1(P)**	**T-F1(O)**	**JSA**
RNN-NLU (Liu and Lane, 2016)	0.905	0.813	0.720	0.783	0.944	0.970	0.931	0.981	0.930	0.718	0.991	0.987	0.854
Slot-gated (Goo et al., 2018)	0.894	0.806	0.694	0.773	0.938	**0.991**	**0.978**	**0.992**	**0.982**	**0.952**	0.997	**0.994**	0.875
Inter-BiLSTM (Wang et al., 2018)	0.869	0.719	0.590	0.728	0.923	0.865	0.871	0.889	0.869	0.788	0.942	0.924	0.711
Capsule NN (Zhang et al., 2019a)	0.876	0.735	0.706	0.643	0.926	**0.991**	0.975	0.991	**0.982**	0.949	0.997	**0.994**	0.855
Joint BERT (Castellucci et al., 2019)	**0.921**	**0.872**	**0.768**	**0.800**	**0.954**	0.989	0.972	**0.992**	0.979	0.914	**0.998**	0.993	**0.895**

Table 5: Joint intent classification and slot labeling performance on CONDA for the five NLU baseline models. It is measured in the four multi-level metrics including: UCA (Utterance Classification Accuracy); the break-down U-F1 for each intent class - E (Explicit), I (Implicit), A (Action), O (Other); the overall T-F1 and breakdown for each slot class - T (Toxicity), S (game Slang), C (Character), D (Dota-specific), P (Pronoun), O (Other); and JSA (Joint Semantic Accuracy).

rate attended context for slot filling and intent classification while explicitly feeding the intent context into the process of slot filling via a gating mechanism.

- **Inter-BiLSTM** (Wang et al., 2018) combines two inter-connected BiLSTMs performing slot filling and intent classification respectively. The information flow between the two tasks occurs by passing the hidden states at each time step from each side to the other to support the decoding process.
- **Capsule NN** (Zhang et al., 2019a) is a capsule-based neural network that explicitly captures the semantic hierarchical relationship among words, slots and intents via a dynamic routing-by-agreement schema.
- **Joint BERT** directly utilizes the merit of pre-trained BERT (Devlin et al., 2019) and non-recursively conducts the joint prediction over the [CLS] token embedding for intent and the sequence of token embeddings for slots.

5.3 Evaluation Metrics

We propose to use the following four metrics for conducting a multi-aspect evaluation. The first two follow the existing traditional abusive language detection research for utterance level detection evaluation while the others are the metrics used for slot-level prediction evaluation and the joint task in NLU models.

(1) **UCA:** **U**tterance **C**lassification **A**ccuracy measures the sentence-level classification performance based on the ratio of the number of correctly predicted utterance to the total number of utterances.

(2) **U-F1:** **U**tterance **F1** score calculates the F1 score for each utterance class.

(3) **T-F1:** **T**oken **F1** score focuses on the prediction performance for slot tokens and calculates an F1 for each class and the token-based micro-averaged F1 score over all classes excluding label O.

(4) **JSA:** **J**oint **S**emantic **A**ccuracy measures the overall prediction performance over the semantic hierarchy. An utterance is deemed correctly analysed only if both utterance-level and all the token-level labels including Os are correctly predicted.

5.4 Implementation details

Links to the source code are given in Appendix C. For Joint BERT, Slot Gated SLU and Capsule NN, we set the number of epochs as 2, 8 and 60, respectively. For the RNN-NLU model, the global step is 1,200 and bidirectional RNN is used with the attention mechanism. For other hyper-parameters, the configuration for the best model in the official GitHub implementation of the baseline models is used. All the experiments are conducted on 16GB Tesla V100-SXM2 GPU with CUDA 10.1.

5.5 Baseline results

The experiment result is provided in Table 5. In columns 2 to 6, the metrics associated with utterance labels are shown. The UCA ranges between 0.87 (Inter-BiLSTM) and 0.92 (Joint BERT), and the U-F1 also illustrates a variance of results for each intent class. We observe that class O always achieves the highest F1 score due to its dominance in numbers throughout the dataset. Comparatively, class I presents relatively low F1 scores due to its subtle nature and reliance on understanding context. The variance in U-F1(I) implies potential improvement in implicit toxicity detection.

Columns 7 to 13 present the metrics related with token labels and show much higher overall performance than utterance labels. This indicates the lexicon-based slot automation gives underlying patterns the model can learn easily. Even so, slot class D always has a lower T-F1 score than other slot classes, indicating game-specific tokens in class D have flexible and variant forms, which increases the difficulty of detection. In the last column, the JSA which jointly handles utterance and token labels is shown. Due to the limitation on utterance level intent classification, it presents comparably low JSA scores, indicating a challenge for improvement.

Amongst the models, the non-recursive Joint BERT model performs the best due to the rich linguistic information learned in pre-training. Joint BERT has an implicit influence between the intent sub-task and the slot sub-task based on a joint loss, whereas the recursive models Slot-gated and Capsule NN have explicit influence flowing from intent to slot, leading to similar slot performance. These explicit lines of influence from one task to the other have shown to be successful in NLU and could be explored further in the toxicity detection task.

6 Transfer Experiment

We compared our dataset with the toxicity detection datasets introduced in Section 4 in terms of transfer performance over utterance-level binary prediction as toxic or non-toxic. That is, training on one dataset and testing on the others. For simplicity, we solely use the intent classification circuit of the Joint BERT as the prediction model.

6.1 Data Preparation

We combine classes into toxic/non-toxic as explained in Section 4. For each dataset, we split into train/test sets in the ratio of 0.9/0.1. The statistics for each dataset are shown in Table 6.

Dataset	**Train / Test**
Waseem	14,581 / 1,621
StormWS	9,849 / 1,621
FoxNews	1,373 / 1,095
CONDA (ours)	40,382 / 4,487

Table 6: Dataset sample counts for transfer experiment.

6.2 Transfer results

The transfer performance measured in UCA is given in Table 7. Firstly, we look at the test performance of each dataset trained on the other datasets, that is to compare results in each column. It can be seen from column 4 that CONDA's transferred performance is generally good when trained on each of the other three datasets, ranging from 0.81 to 0.83. This implies that CONDA covers the nature of toxicity that can be generalized from the other toxicity datasets which emphasize hatefulness, sexism and racism.

Changing the perspective to the rows, we compare the test performance for each dataset on a model on another dataset. StormfrontWS and CONDA perform well on Waseem, picking up the racism components there. However Waseem does not perform well when trained on either of those, suggesting their specific hate speech and game speech respectively is too focused. FoxNews training transfers the weakest results indicating its general news nature is too broad.

StormfrontWS performs well when trained on CONDA due to shared toxicity characteristics, but the performance of Waseem and FoxNews when tested on a model trained on CONDA is relatively low. We propose that this is due to two aspects of our dataset previously discussed: the specific game related nature of our language, and the shorter utterances in our set compared to the others.

		Testing			
		Waseem	StormWS	FoxNews	**CONDA**
Training	Waseem	-	0.8845	0.7287	0.8307
	StormWS	0.7118	-	0.7379	0.8329
	FoxNews	0.6931	0.8241	-	0.8056
	CONDA	0.6955	0.8748	0.6690	-

Table 7: Transfer Learning Result, UCA (Utterance Classification Accuracy).

7 Conclusion and Future Work

In this paper, we propose CONDA, a new dataset with dual-level (token and utterance) annotation for understanding in-game chat and to detect toxicity. Compared to previous studies, we draw on the NLU perspective and use the joint token-utterance aspect for detection of toxicity. Accordingly, we formalise a multi-level evaluation system. Through experiments with joint slot and intent NLU models, we show the promising potential of such models for toxicity detection utilizing the dual-level annotation. We also compare our dataset with other benchmark toxicity datasets in the literature through a transfer experiment. In future work, the automated token labelling can be manually adjusted and the size of the dataset can be expanded.

8 Ethics/Broader Impact Statement

The study follows the ethical policy set out in the ACL code of Ethics[5] and addresses the ethical impact of presenting a new dataset. In addition, it is approved by our Institutional Review Board (project number : 2019/741). As described in the data collection section, our annotated dataset, CONDA, is based on the *Dota 2* game chat where it can be accessed on Kaggle website (See Section 3.1).

For our automated slot labelling, we generated the game toxicity lexicon by taking the supplemental materials released by Märtens et al. (2015) and ElSherief et al. (2018) and the list of words banned by Google[6]. We then added variants or new toxic words found in the utterances extracted from Kaggle. For intent labelling, all volunteer annotators were recruited from academia and research students. They were informed about toxic behavior in online games before handling the data. Our instructions allowed them to feel free to leave if they were uncomfortable with the content. Due to privacy considerations, we group them by online game experiences and do not take into account annotators' demographic information.

The CONDA dataset is intended for toxicity detection in online games by providing both slot and intent labels. With respect to the potential risks, we note that the subjectivity of human annotation would impact on the quality of the dataset. In order to improve the quality of our dataset, we compared the inter-rater agreements between a gamers' group and a non-gamers' group, and then final annotation of the whole dataset was performed by gamers only.

References

Sonam Adinolf and Selen Turkay. 2018. Toxic behaviors in esports games: player perceptions and coping strategies. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, pages 365–372.

ADL. 2019. Free to play: Hate, harassment and positive social experiences in online games.

Jeremy Blackburn and Haewoon Kwak. 2014. STFU NOOB! Predicting crowdsourced decisions on toxic behavior in online games. In *Proceedings of the 23rd international conference on World wide web*, pages 877–888.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Giuseppe Castellucci, Valentina Bellomaria, Andrea Favalli, and Raniero Romagnoli. 2019. Multilingual intent detection and slot filling in a joint bert-based model. *arXiv preprint arXiv:1907.02884*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Mai ElSherief, Shirin Nilizadeh, Dana Nguyen, Giovanni Vigna, and Elizabeth Belding. 2018. Peer to peer hate: Hate speech instigators and their targets. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49.

Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP*, pages 260–266.

Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the NAACL-HLT*, pages 753–757.

---

[5] https://www.aclweb.org/portal/content/acl-code-ethics

[6] https://github.com/RobertJGabriel/Google-profanity-words

Arshit Gupta, Peng Zhang, Garima Lalwani, and Mona Diab. 2019. Casa-nlu: Context-aware self-attentive natural language understanding for task-oriented chatbots. In *Proceedings of the (EMNLP-IJCNLP)*, pages 1285–1290.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792.

Haewoon Kwak, Jeremy Blackburn, and Seungyeop Han. 2015. Exploring cyberbullying and other toxic behavior in team competition online games. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3739–3748.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689.

Marcus Märtens, Siqi Shen, Alexandru Iosup, and Fernando Kuipers. 2015. Toxicity detection in multiplayer online games. In *2015 International Workshop on Network and Systems Support for Games (NetGames)*, pages 1–6. IEEE.

Joaquim Alvino de Mesquita Neto and Karin Becker. 2018. Relating conversational topics and toxic behavior effects in a moba game. *Entertainment computing*, 26:10–29.

Shane Murnion, William J Buchanan, Adrian Smales, and Gordon Russell. 2018. Machine learning and semantic analysis of in-game chat for cyberbullying. *Computers & Security*, 76:197–213.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.

John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the ACL*, pages 4296–4305. Association for Computational Linguistics.

Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth M. Belding, and William Yang Wang. 2019. A benchmark dataset for learning to intervene in online hate speech. In *Proceedings of the 2019 Conference on EMNLP-IJCNLP*, pages 4754–4763. Association for Computational Linguistics.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the NAACL-HLT*, pages 3795–3805. Association for Computational Linguistics.

Wessel Stoop, Florian Kunneman, Antal van den Bosch, and Ben Miller. 2019. Detecting harassment in real-time as conversations develop. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 19–24.

Kunze Wang, Dong Lu, Caren Han, Siqu Long, and Josiah Poon. 2020. Detect all abuse! toward universal abusive language detection models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6366–6376.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *NAACL-HLT (2)*.

Yufan Wang, Tingting He, Rui Fan, Wenji Zhou, and Xinhui Tu. 2019. Effective utilization of external knowledge and history context in multi-turn spoken language understanding model. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 960–967. IEEE.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and S Yu Philip. 2019a. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the ACL*, pages 5259–5267.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S. Yu. 2019b. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Conference of the ACL*, pages 5259–5267. Association for Computational Linguistics.

Justine Zhang, Jonathan P. Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Dario Taraborelli, and Nithum Thain. 2018. Conversations gone awry: Detecting early signs of conversational failure. In *Proceedings of the 56th Annual Meeting of the ACL*, pages 1350–1361. Association for Computational Linguistics.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI*, pages 2993–2999. IJCAI/AAAI Press.

Appendix

A The CONDA datasets

The CONDA dataset consists of 9 columns as follows:

- `matchId` (numeric): Each match has a unique ID from raw data.
- `conversationId` (numeric): Each conversation has a unique ID generated by this research to provide guidance for human annotation.
- `playerId` (alphanumeric): Individual players have a unique ID from raw data.
- `playerSlot` (numeric): Individual players have a unique number associated with their roles in each match.
- `chatTime` (numeric): Each utterance has time (in seconds) when it appears in each match. For example, an utterance occurring 10 minutes after starting the game has a chatTime of 600.
- `utterance` (alphanumeric): Original raw data before any data processing (e.g. ‘retard sf…’).
- `slotTokens` (alphanumeric): Tokenised, cleaned, and slot labelled data (e.g. retard (T), sf (C)).
- `intentClass` (alphabetic): Utterance-level annotated labels - E (Explicit), I (Implicit), A (Action), and O (Other).
- `slotClasses` (alphabetic): Token-level annotated labels - T (Toxicity), C (Character), D (Dota-specific), S (game Slang), P (Pronoun), and O (Other).

B Word clouds

(a) Class ’Explicit’ (b) Class ’Implicit’

(c) Class ’Action’ (d) Class ’Other’

Figure 5: Word clouds for each intent class

The word clouds visualizes the most frequent words associated with each intent class. The top keywords in each class are “noob” in E, “ez” in I, “report” in A, “gg” in O.

C Source code

The source code for the models used to analyse our dataset is available at the following GitHub addresses:

- RNN-NLU:
https://github.com/HadoopIt/rnn-nlu
- Slot-gated:
https://github.com/MiuLab/SlotGated-SLU
- Inter-BiLSTM:
https://github.com/ray075hl/Bi-Model-Intent-And-Slot
- Capsule NN:
https://github.com/czhang99/Capsule-NLU
- Joint BERT:
https://github.com/monologg/JointBERT

Adaptive Knowledge-Enhanced Bayesian Meta-Learning for Few-shot Event Detection

Shirong Shen[1] and **Tongtong Wu**[1] and **Guilin Qi**[1*] and **Yuan-Fang Li**[2] and
Gholamreza Haffari [2] and **Sheng Bi**[1]
[1]School of Computer Science and Engineering, Southeast University, China
[2]Faculty of Information Technology, Monash University, Melbourne, Australia
`{ssr, wutong8023, gqi}@seu.edu.cn, yuanfang.li@monash.edu,`
`Gholamreza.Haffari@monash.edu, bisheng@seu.edu.cn`

Abstract

Event detection (ED) aims at detecting event trigger words in sentences and classifying them into specific event types. In real-world applications, ED typically does not have sufficient labelled data, thus can be formulated as a few-shot learning problem. To tackle the issue of low sample diversity in few-shot ED, we propose a novel knowledge-based few-shot event detection method which uses a definition-based encoder to introduce external event knowledge as the knowledge prior of event types. Furthermore, as external knowledge typically provides limited and imperfect coverage of event types, we introduce an adaptive knowledge-enhanced Bayesian meta-learning method to dynamically adjust the knowledge prior of event types. Experiments show our method consistently and substantially outperforms a number of baselines by at least 15 absolute F_1 points under the same few-shot settings.

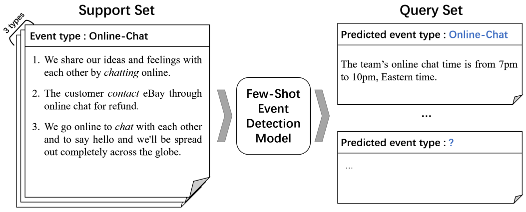

1. We share our ideas and feelings with each other by *chatting* online.
2. The customer *contact* eBay through online chat for refund.
3. We go online to *chat* with each other and to say hello and we'll be spread out completely across the globe.

The team's online chat time is from 7pm to 10pm, Eastern time.

Figure 1: A 3-way 3-shot event detection example, in which the model uses the support set to predict the event types of query samples.

1 Introduction

Event detection is an important task in information extraction, aiming at detecting event triggers from text and then classifying them into event types (Chen et al., 2015). For example, in "*The police* ***arrested*** *Harry on charges of manslaughter*", the trigger word is *arrested*, indicating an **"Arrest"** event. Event detection has been widely applied in Twitter analysis (Zhou et al., 2017), legal case extraction (de Araujo et al., 2017), and financial event extraction (Zheng et al., 2019), to name a few.

Typical approaches to event detection (Chen et al., 2015; McClosky et al., 2011; Liu et al., 2019) generally rely on large-scale annotated datasets for training. Yet in real-world applications, adequate labeled data is usually unavailable. Hence, methods that generalize effectively with small quantities of labeled samples and adapt quickly to new event types are highly desirable for event detection.

Various approaches have been proposed to enable learning from only a few samples, i.e., few-shot learning (Finn et al., 2017; Snell et al., 2017; Zhang et al., 2018a). Yet few-shot event detection (FSED) has been less studied until recently (Lai et al., 2020a; Deng et al., 2020). Although these methods achieve encouraging progress on typical N-way M-shot setting (Figure 1), the performance remains unsatisfactory as the *diversity* of examples in the support set is usually limited.

Intuitively, introducing high-quality semantic knowledge, such as FrameNet (Baker et al., 1998), is a potential solution to the insufficient diversity issue (Qu et al., 2020; Tong et al., 2020; Liu et al., 2016, 2020). However, as shown in Figure 2, such knowledge-enhanced methods also suffer from two major issues: (1) the incomplete coverage by the knowledge base and (2) the uncertainty caused by the inexact alignment between predefined knowledge and diverse applications.

To tackle the above issues, in this paper, we propose an Adaptive Knowledge-Enhanced Bayesian Meta-Learning (AKE-BML) framework. More specifically, (1) we align the event types between the support set and FrameNet via heuristic rules.[1] (2) We propose encoders for encoding the sam-

---

* Corresponding author.

[1] For event types that cannot be accurately aligned to FrameNet, we match the nearest super-ordinate frame for them.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2417–2429
August 1–6, 2021.

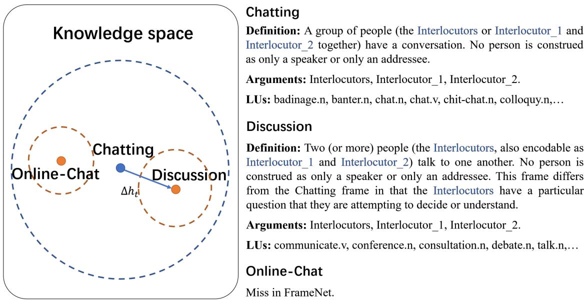

Chatting

Definition: A group of people (the Interlocutors or Interlocutor_1 and Interlocutor_2 together) have a conversation. No person is construed as only a speaker or only an addressee.

Arguments: Interlocutors, Interlocutor_1, Interlocutor_2.

LUs: badinage.n, banter.n, chat.n, chat.v, chit-chat.n, colloquy.n,…

Discussion

Definition: Two (or more) people (the Interlocutors, also encodable as Interlocutor_1 and Interlocutor_2) talk to one another. No person is construed as only a speaker or only an addressee. This frame differs from the Chatting frame in that the Interlocutors have a particular question that they are attempting to decide or understand.

Arguments: Interlocutors, Interlocutor_1, Interlocutor_2.

LUs: communicate.v, conference.n, consultation.n, debate.n, talk.n,…

Online-Chat

Miss in FrameNet.

Figure 2: An example of FrameNet. Left side: the relation between frame ‘Chatting’ and its sub-frame. Right side: the definitions and LUs (Lexical Units) of frame Chatting and Discussion. The blue words represent the mentions of arguments in definition. It can be seen that, in FrameNet, the definition of the sub-frame is similar to the definition of its super-frame. External knowledge base can provide rich semantic information, yet the knowledge base is typically incomplete, such as the missing of a desired frame “online-chat”.

ples and knowledge-base in the same semantic space. (3) We propose a learnable offset for revising the aligned knowledge representations to build the knowledge prior distribution for event types and generate the posterior distribution for event type prototype representations. (4) In the prediction phrase, we adopt the learned posterior distribution for prototype representations to classify query instances into event types.

We conduct comprehensive experiments on the aggregated benchmark dataset of few-shot event detection (Deng et al., 2020). The experimental results show that our method consistently and substantially outperforms state-of-the-art methods. In all six N-way-M-shot settings, our model achieves a large F_1 superiority of at least 15 absolute points.

2 Related Work

Event Detection. Recent event detection methods based on neural networks have achieved good performance (Chen et al., 2015; Sha et al., 2016; Nguyen et al., 2016; Lou et al., 2021). These methods use neural networks to construct the context features of candidate trigger words to classify events. Pre-trained language models such as BERT (Devlin et al., 2019) have also become an indispensable component of event detection models (Yang et al., 2019; Wadden et al., 2019; Shen et al., 2020). However, neural models rely on large-scale labeled event datasets and fail to predict the labels of new event types. A recent study utilized the basic metric-based few-shot learning method for event detection (Lai et al., 2020b). Deng et al. (2020) tackles few-shot learning for event classification with a dynamic memory network. To enhance background knowledge, ontology embedding is used in ED (Deng et al., 2021). These methods have achieved encouraging results in the few-shot learning setting. However, they do not address the problem of insufficient sample diversity in the support set. Our method leverages the knowledge in *FrameNet* to augment the support set for event detection.

Few-shot Learning and Meta-learning. Few-shot learning trains a model with only a few labeled samples in a support set and predicts the labels of unlabeled samples in the query set. Various approaches have been proposed to solve the few-shot learning problem, which mainly fall into three categories: (1) metric-based methods (Vinyals et al., 2016; Snell et al., 2017; Garcia and Bruna, 2012; Sung et al., 2018), (2) optimization-based methods (Finn et al., 2017; Nichol et al., 2018; Ravi and Larochelle, 2016), and (3) model-based methods (Yan et al., 2015; Zhang et al., 2018b; Sukhbaatar et al., 2015; Zhang et al., 2018a). However, these methods rely heavily on the support set and suffer from poor robustness caused by insufficient sample diversity of the support set.

Bayesian meta-learning (Ravi and Larochelle, 2016; Yoon et al., 2018) can construct the posterior distribution of the prototype vector through external information outside the support set. The effectiveness of this method has been shown in the few-shot relation extraction task (Qu et al., 2020). It inspires us to solve the problem of insufficient sample diversity in the task of few-shot event detection by introducing external knowledge. However, this method ignores the semantic deviation between knowledge and target types. Specifically, a knowledge base may provide incomplete coverage of target types in a given support set, which leads to inaccurate matching between a target type and knowledge.

3 Problem Definition

In this paper, the Few-Shot Event Detection (FSED) problem is defined as a typical *N-way-M-shot* problem. Specifically, a tiny labeled *support set* S is provided for model training. S contains N distinct event types and each event type has only M labeled samples, where M is typically small (e.g. $M = 5, 10, 15$). More precisely, in each FSED task we are given a small support set $S = \{(x_s, y_s)\}$.

Let $X_S = \{x_s\}_{s \in S}$ represent the samples in the support set S, i.e. $x_s = (I_s, tt_s)$, where I_s is the sentence of the sample x_s and tt_s is the candidate trigger word of x_s. We denote by Y_S an ordered list of event types, i.e. $Y_S = \{y_s\}_{s \in S}$, where each y_s is the ground-truth event type of sample x_s. For each support set S, we only consider a subset of event types T_s from the entire set of event types T. Hence, in the N-way-M-shot setting, $|T_S| = N$ and $|X_S| = |Y_S| = N * M$.

Moreover, we assume an external knowledge base $\mathscr{F}$ that contains a number of *frames*. Each frame $F_t \in \mathscr{F}$ consists of three parts: $F_t = (D_t, A_t, L_t)$, where D_t, A_t and L_t are the definition, arguments, and linguistic units (LUs) of the frame respectively. Please see Appendix A for details of FrameNet.

For each support set S, we are also given a query set Q composed of some unlabeled samples $X_Q = \{x_q\}_{q \in Q}$, where $x_q = (I_q, tt_q)$, I_q is the sentence of sample x_q, and tt_q is the candidate trigger word of x_q. Our goal is to learn a neural classifier for these event types by using the external knowledge and the support set. We will apply the classifier to predict the labels of the query samples in Q, i.e., $Y_Q = \{y_q\}_{q \in Q}$ with each $y_q \in T_S$. We do this by learning $p(Y_Q|X_Q, X_S, Y_S, \mathscr{F})$.

4 Adaptive Knowledge-Enhanced Bayesian Meta-Learning

We now present our adaptive knowledge-enhanced few-shot event detection approach. The overall structure of our method is shown in Figure 3. Our method represents each event type t with a prototype vector $\mathbf{v}_t$, which is then used to classify the query sentences. We use $\mathbf{V}_{T_S} = \{\mathbf{v}_t\}_{t \in T_S}$ to represent the collection of prototype vectors for all event types in T_S. Then the conditional distribution $p(Y_Q|X_Q, X_S, Y_S, \mathscr{F})$ can be represented as:

$$\int p(Y_Q|X_Q, V_{T_S}) p(V_{T_S}|X_S, Y_S, \mathscr{F})\, d\mathbf{V}_{T_S}. \quad (1)$$

To calculate Eq. 1, we first introduce the sample encoder and knowledge encoder to give the vector representations of samples and the knowledge of event types. Then we use the sample representations and knowledge representation to construct the adaptive knowledge-enhanced posterior distribution $p(V_{T_S}|X_S, Y_S, \mathscr{F})$ of $\mathbf{V}_{T_S}$ and give the likelihood $p(Y_Q|X_Q, V_{T_S})$ by $\mathbf{V}_{T_S}$ and sample representations. Finally we leverage Monte Carlo sampling to approximate the posterior distribution and draw each prototype sample by the stochastic gradient Langevin dynamics (Welling and Teh, 2011) to optimize model parameters in an end-to-end fashion. We now explain the framework in more details.

4.1 Sample and Knowledge Encoder

The purpose of encoding knowledge is to make up for the lack of diversity and coverage of the support set. Thus we align the knowledge and sample encoding and map them into the same semantic space. Intuitively, trigger and arguments are the main factors for entity detection. Hence, to align the trigger and arguments from samples and external knowledge, we design two encoders for the knowledge and samples, generating the final knowledge encoding $\mathbf{h}_t$ and the sample encoding $\boldsymbol{\mathcal{E}}(x)$ with the same dimensions.

Knowledge Encoder. Given a knowledge frame $F_t = \{D_t, A_t, L_t\}$ for the event type t, we encode it into a real-valued vector to represent the semantics of t. As shown in Figure 2, for a frame F_t, the linguistic units L_t can represent the features of the trigger words, the arguments A_t can represent the context of the trigger words in samples, and D_t describes the semantic relationship between A_t and t.

For each event type t, the proposed knowledge encoder uses BERT to generate the text encoding $\mathbf{E}_{D_t}$ and $\mathbf{E}_{L_t}$ from the description D_t and the LUs L_t respectively. Moreover, the arguments encoding $\mathbf{E}_{A_t}$ is a sequence of $\mathbf{e}_{A_t}^{(i)}$, i.e., the average token encoding in the i-th argument mention in D_t, which ensures that the encoding of A_t fully contains the semantics of the event type t. Then, as shown in Figure 3, the trigger word prior encoding and argument prior encoding are generated by follows:

- **Trigger word prior encoding.** We use attention to get the weighted sum of words in L_t as the trigger word prior encoding $\mathbf{e}_{L_t}^*$. The *query* of the attention is $\mathbf{E}_{D_t}$, *key* and *value* are both $\mathbf{E}_{L_t}$.
- **Argument prior encoding.** An attention mechanism is used to aggregate the arguments information into $\mathbf{e}_{A_t}^*$, where the *query* of the attention is $\mathbf{e}_{L_t}^*$, *key* and *value* are both $\mathbf{E}_{A_t}$.

Finally, we concatenate the trigger word prior encoding $\mathbf{e}_{L_t}^*$ and the argument prior encoding $\mathbf{e}_{A_t}^*$, and use a feed forward network f^h to generate the knowledge encoding vector $\mathbf{h}_t$ of event type t,

$$\mathbf{h}_t = f^h\left(\left[\mathbf{e}_{A_t}^*; \mathbf{e}_{L_t}^*\right]\right). \quad (2)$$

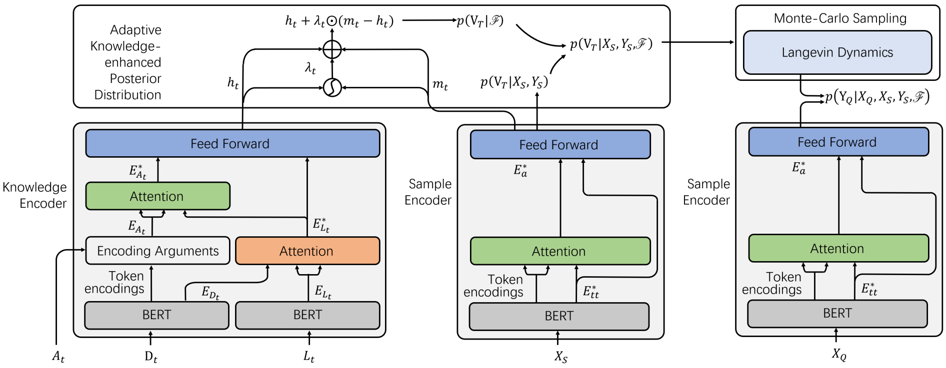

Figure 3: Framework overview. Our method combines both the external knowledge and the support set into a prior distribution of event prototype. We customize two encoders to generate sample representations and knowledge representations. Then we utilize the support set to generate a learnable offset for revising the aligned knowledge representations to generate the prior distribution for prototype representations. Finally, we use Monte-Carlo sampling and stochastic gradient Langevin dynamics to draw samples of prototypes for prediction.

Sample encoder. We follow the same strategy to build a sample encoder. Given each sample $x = (I, tt)$, i.e., a candidate trigger word tt and its context I, we first utilize BERT to encode x and select the encoding of tt as the trigger representation $\mathbf{e}^*_{tt}$. As arguments are not explicitly given in x, we use an attention mechanism to aggregate the implicit argument information for current trigger tt, in which the *query* is $\mathbf{e}^*_{tt}$, *key* and *value* are both token encoding generate from I. We denote the argument encoding as $\mathbf{e}^*_a$.

Finally, we concatenate the trigger word encoding $\mathbf{e}^*_{tt}$ and the argument encoding $\mathbf{e}^*_a$, and use a feed forward network $f^{\mathcal{E}}$ to generate the sample encoding vector $\boldsymbol{\mathcal{E}}(x)$,

$$\boldsymbol{\mathcal{E}}(x) = f^{\mathcal{E}}\left([\mathbf{e}^*_a; \mathbf{e}^*_{tt}]\right). \tag{3}$$

4.2 Adaptive Knowledge-Enhanced Posterior

The posterior distribution can be factorized into a prior distribution (given the event knowledge) and a likelihood on the support set (Qu et al., 2020) as,

$$p(\mathbf{V}_{T_S}|X_S, Y_S, \mathscr{F}) \propto p(Y_S|X_S, \mathbf{V}_{T_S})p(\mathbf{V}_{T_S}|\mathscr{F}), \tag{4}$$

where $p(Y_S|X_S, \mathbf{V}_{T_S})$ is the likelihood on the support set, and $p(\mathbf{V}_{T_S}|\mathscr{F})$ is the adaptive knowledge-based prior for the prototype vectors. We describe the details of these two components as follows:

Adaptive Knowledge-based Prior. As we discussed in Section 1, an event type t may not have an exact/perfect match in the knowledge base $\mathscr{F}$. In such situations, we resort to finding the super-ordinate frame of t, which is semantically closest to t. As shown in Figures 1 and 2, where the event type t in the support set 'online-chat' is matched against the knowledge prior F_t 'Chatting' in FrameNet, a super-ordinate frame. In order to enable the knowledge encoding to accurately reflect the characteristics of the corresponding event type, we add a learnable *knowledge offset* to $\mathbf{h}_t$. We denote the knowledge offset between the event type t and its knowledge encoding $\mathbf{h}_t$ by $\Delta\mathbf{h}_t$. Recall that the knowledge in $\mathbf{h}_t$ is encoded from the exactly-matched frame or the super-ordinate frame. $\Delta\mathbf{h}_t$ is defined as follows:

$$\Delta\mathbf{h}_t = \boldsymbol{\lambda}_t \odot (\mathbf{m}_t - \mathbf{h}_t), \tag{5}$$

where $\odot$ is the element-wise product, and $\mathbf{m}_t$ is the mean of the encodings $\boldsymbol{\mathcal{E}}(x)$ of all the samples x in the support set. $\boldsymbol{\lambda}_t \in [0, 1]^{|\mathbf{h}_t|}$ is the adaptive weight (gate), which is obtained from the sample encoding $\mathbf{m}_t$ and the knowledge encoding $\mathbf{h}_t$:

$$\boldsymbol{\lambda}_t = \sigma(\mathbf{W}_\lambda\left[\mathbf{m}_t; \mathbf{m}_t - \mathbf{h}_t; \mathbf{h}_t\right] + \mathbf{b}_\lambda), \tag{6}$$

where σ is the nonlinear sigmoid function, and $\mathbf{W}_\lambda$ and $\mathbf{b}_\lambda$ are trainable parameters.

Putting it altogether, the knowledge prior distribution has the following form,

$$\begin{aligned} p(\mathbf{V}_{T_S}|\mathscr{F}) &= \prod_{t \in T_S} p(\mathbf{v}_t|\mathbf{h}_t, \Delta\mathbf{h}_t) \\ &= \prod_{t \in T_S} \mathcal{N}(\mathbf{v}_t|\mathbf{h}_t + \Delta\mathbf{h}_t, \boldsymbol{I}), \end{aligned} \tag{7}$$

where $\mathcal{N}(\mathbf{v}_t|\mathbf{h}_t + \Delta\mathbf{h}_t, \boldsymbol{I})$ is multivariate Gaussian with the mean $\mathbf{h}_t + \Delta\mathbf{h}_t$ and covariance $\boldsymbol{I}$ (the identity matrix). So, each prototype vector has a prior distribution containing knowledge from *FrameNet* adaptively adjusted according to the support set.

Likelihood. With the given prototype vectors $\mathbf{V}_{T_s}$ distributed according to $p(\mathbf{V}_{T_S}|X_S, Y_S, \mathscr{F})$, the likelihood for support samples is defined as,

$$\begin{aligned} p(Y_S|X_S,\mathbf{V}_{T_S}) &= \prod_{s\in S} p(y_s|x_s,\mathbf{V}_{T_S}) \qquad (8)\\ p(y_s=t|x_s,\mathbf{V}_{T_S}) &:= \frac{\exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\mathbf{v}_t)}{\sum_{t'\in T_S}\exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\mathbf{v}_{t'})}. \end{aligned}$$

The dot product of the sample encoding $\boldsymbol{\mathcal{E}}(x_q)$ and the event type prototype vector $\mathbf{v}_t$ estimates their similarity. We use $softmax$ to normalize the result to the probability of x_s belonging to event type t.

4.3 Optimization and Prediction

For prediction, the model computes and maximizes the log-probability $\log p(Y_Q|X_Q,X_S,Y_S,\mathscr{F})$. However, according to Eqn (1), the log-probability relies on the integration over prototype vectors, which is difficult to compute. Hence, we estimate it with Monte Carlo sampling (Qu et al., 2020),

$$\begin{aligned} &p(Y_Q|X_Q,X_S,Y_S,\mathscr{F})\\ &=\mathbb{E}_{p(\mathbf{V}_{T_S}|X_S,Y_S,\mathscr{F})}\left[p(Y_Q|X_Q,\mathbf{V}_{T_S})\right]\\ &\approx\frac{1}{N_s}\sum_{i=1}^{N_s}p(Y_Q|X_Q,\mathbf{V}_{T_S}^{(s)}) \qquad (9)\end{aligned}$$

where N_s is the number of samples, and $\mathbf{V}_{T_S}^{(s)}$ is a sample drawn from the posterior distribution, i.e. $\mathbf{V}_{T_S}^{(s)} \sim p(\mathbf{V}_{T_S}|X_S,Y_S,\mathscr{F})$. $p(Y_Q|X_Q,\mathbf{V}_{T_S}^{(s)})$ is the likelihood for query samples which has the same form as Eqn 8. To sample from the posterior, we use the stochastic gradient Langevin dynamics (Welling and Teh, 2011) with multiple stochastic updates. Formally, we initialize the sample $\hat{\mathbf{V}}_{T_S}$ and iteratively update the sample as,

$$\begin{aligned}\hat{\mathbf{V}}_{T_S}\leftarrow&\hat{\mathbf{V}}_{T_S}+\sqrt{\epsilon}\boldsymbol{z} \qquad (10)\\ &+\frac{\epsilon}{2}\nabla_{\hat{\mathbf{V}}_{T_S}}\log\ p(Y_S|X_S,\hat{\mathbf{V}}_{T_S})p(\hat{\mathbf{V}}_{T_S}|\mathscr{F}),\end{aligned}$$

where $\boldsymbol{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$, and ϵ is a small real number representing the update step size. The gradient $\nabla_{\hat{\mathbf{V}}_{T_S}}\log\ p(Y_S|X_S,\hat{\mathbf{V}}_{T_S})p(\hat{\mathbf{V}}_{T_S}|\mathscr{F})$ in Eqn 10 balances the effect of the knowledge and the support set on the prototype vector. Please see Appendix B for derivation details and intuitive explanations of its influence.

The Langevin dynamics requires a burn-in period. To speed up the convergence, we follow the previous method (Qu et al., 2020) and initialize the sample as follows,

$$\begin{aligned}\hat{\mathbf{V}}_{T_S}&\leftarrow\{\hat{\mathbf{v}}_t\}_{t\in T_S}\\ \hat{\mathbf{v}}_t&\leftarrow\mathbf{m}_t+\mathbf{h}_t+\Delta\mathbf{h}_t-\mathbf{m},\end{aligned}\qquad(11)$$

where $\mathbf{m}$ is the mean encoding of all the samples in the support set.

Algorithm 1: Training Process

```
Input: Event type set $T$
Input: Event knowledge from FrameNet
while not convergence do
    Sample a subset $T_S$ from $T$ to build a FSED task
    Sample disjoint support and query sets for $T_S$
    Compute the sample encodings (Eq. 3)
    Compute the $\{\mathbf{m}_t\}_{t\in T_S}$ for each $t\in T_S$
    Compute knowledge encodings $\{\mathbf{h}_t\}_{t\in T_S}$ (Eq. 2)
    Compute knowledge offset $\{\Delta\mathbf{h}_t\}_{t\in T_S}$ (Eq. 5)
    Initialize prototype vectors $\{\mathbf{V}^s_{T_S}\}_{s=1}^{N_s}$ (Eq. 11)
    Update prototype vectors iteratively (Eq. 12)
    Compute and maximize log-likelihood (Eq. 9)
```

After obtaining prototype samples from the posterior, $\log p(Y_Q|X_Q,X_S,Y_S,\mathscr{F})$ is end-to-end approximated according to Eqn (9). During the training stage, we optimize the log-likelihood of the query set and update the model parameters by gradient descent. In the prediction stage, the log-likelihood will determine the probability that a query sample belongs to each event type. The training process is shown in Algorithm 1.

5 Experiments

We conduct evaluation with the following goals: (1) to compare our adaptive knowledge-enhanced Bayesian meta-learning method with existing few-shot event detection methods and few-shot learning baseline methods; (2) to assess the effectiveness of introducing external knowledge in different N-way-M-shot settings; and (3) to provide empirical evidence that our adaptive knowledge offset can flexibly adjust the impact of the support set and prior knowledge on event prototypes, making the model more accurate and generalizable.

5.1 Experimental Settings

We evaluate our method on an aggregated few-shot event detection dataset *FewEvent*[2] (Deng et al., 2020). *FewEvent* combines two currently widely-used event detection datasets, the ACE-2005 corpus[3] and the TAC-KBP-2017 Event Track Data[4], and adds external event types in specific domains including music, film, sports and education (Deng et al., 2020). As a result, *FewEvent* contains 70,852 samples for 19 event types that are further divided into 100 event subtypes.

In order to match the few-shot settings , we use

---

[2] `https://github.com/231sm/Low_Resource_KBP`

[3] `http://projects.ldc.upenn.edu/ace/`

[4] `https://tac.nist.gov/2017/KBP/Event/index.html`

88 event types covering a total of 15,681 samples to construct experimental data. 68 event types are selected for training, 10 for validation, and the rest 10 for testing. Note that there are no overlapping types between the training, validation and testing sets. In order to obtain a convincing result, we conducted 5 random divisions of training and testing for all event types, and the experimental results are averaged as the final result.

The comparisons with our AKE-BML are performed in two aspects, the *sample encoder* and the *few-shot learner*. We combine different encoders and few-shot learners to obtain different baseline models. We consider four sample encoders including CNN (Kim, 2014), Bi-LSTM (Huang et al., 2015), DMN (Kumar et al., 2016) and our trigger-attention-based sample encoder TA. For few-shot learners, we consider Matching Networks (MN) (Vinyals et al., 2016) and Prototypical Networks (PN) (Snell et al., 2017). We also compare to the SOTA few-shot event detection method DMN-MPN (Deng et al., 2020), which uses a dynamic memory network (DMN) as the sample encoder and a memory-based prototypical network as the few-shot learner. In addition, in order to verify the effectiveness of our proposed method, we perform an ablation study on our model, which evaluate the model without external knowledge and without dynamic knowledge adaptation.

As a result, the following methods are compared in our experiments:

- **AKE-BML**, our adaptive knowledge-enhanced Bayesian meta-learning method which uses TA encoder as the sample encoder.
- **KB-BML**, a variant of AKE-BML without dynamic knowledge adaption.
- **TA-BML**, a variant of AKE-BML using our TA encoder but without using external knowledge.
- **DMN-MPN**, dynamic-memory-based prototypical network (Deng et al., 2020).
- **Encoder+Learner**, combinations of various sample encoders and event type learners (e.g. CNN+MN and TA+PN).

We use stochastic gradient descent (Bottou, 2012) as the optimizer in training with the learning rate 1×10^{-5}. The sampling times N_s of Monte Carlo sampling and update step size ϵ are set to 10 and 0.01 respectively. The update times of stochastic gradient Langevin dynamics M is set to 5. We use dropout after the sample encoder and the knowledge encoder to avoid over-fitting; the dropout rate is set to 0.5. We evaluate the performance of event detection with F_1 and $Accuracy$ scores.

5.2 Main Results

As shown in Table 1, we compare methods on F_1 and $Accuracy$ scores. We observe the followings:

- Our full model AKE-BML outperforms all other methods on both $Accuracy$ and F_1 scores across all settings. Compared with the SOTA method DMN-MPN, AKE-BML achieves a substantial improvement of 15–23 absolute F_1 points in all N-way-M-shot settings. It shows our adaptive knowledge-enhanced Bayesian meta-learning method can effectively utilize external knowledge and adjust it according to the support set, thus build better prototypes of event types. Please see Appendix C and D for a detailed performance analysis over various N-way and M-shot settings.
- With the sample encoders (Bi-LSTM, CNN, DMN and TA) fixed, it can be observed that prototypical networks (PN) consistently outperforms matching networks (MN). DMN-MPN performs better than PN-based methods, because the dynamic memory network can extract key information from the support set through multiple iterations. However, DMN-MPN only considers the information of a few samples in each support set, hence suffering from insufficient sample diversity similar to PN- and MN-based methods.
- TA-BML performs similarly with DMN-MPN under the settings of N-way-5-shot and N-way-10-shot, but slightly worse under the N-way-15-shot setting. One possible explanation is that when the number of samples in the support set is larger, MPN can generate higher-quality prototypes. In addition, the performance of TA-BML is not as good as KB-BML, which shows the importance of introducing external knowledge.
- Compared with KB-BML, our full model AKE-BML can effectively solve the problem of deviation between knowledge and event types, and generate event prototypes with better generalization through knowledge. Compared with TA-BML, which does not incorporate external knowledge, AKE-BML achieves an even larger performance advantage, which further demonstrates the effectiveness of external knowledge.

5.3 Case Study

We present a case study on the dynamic knowledge adaptation between the support set and the

Model	5-Way-5-Shot F_1/*Accuracy*	5-Way-10-Shot F_1/*Accuracy*	5-Way-15-Shot F_1/*Accuracy*	10-Way-5-Shot F_1/*Accuracy*	10-Way-10-Shot F_1/*Accuracy*	10-Way-15-Shot F_1/*Accuracy*
Bi-LSTM+MN§	58.19/58.48	61.26/61.45	65.55/66.04	46.43/47.62	51.97/52.60	56.27/56.47
CNN+MN§	59.30/60.04	64.81/65.15	68.35/68.58	44.85/45.80	50.14/50.67	54.13/54.49
DMN+MN§	66.09/67.18	68.92/69.33	70.88/71.17	52.81/54.12	58.04/58.38	61.63/62.01
TA+MN	66.83/67.55	69.12/69.64	71.13/71.59	53.49/55.47	59.58/60.01	62.41/63.11
Bi-LSTM+PN§	62.42/62.72	64.65/64.71	68.23/68.39	53.15/53.59	55.87/56.19	60.34/60.87
CNN+PN§	63.69/64.89	69.64/69.74	70.42/70.52	51.12/51.51	53.80/54.01	57.89/58.28
DMN+PN§	72.08/72.43	72.47/73.38	73.91/74.68	59.95/60.07	61.48/62.13	65.84/66.31
TA+PN	73.66/73.92	73.81/74.63	75.69/76.31	61.25/61.88	63.89/64.31	66.21/67.59
DMN-MPN§	73.59/73.86	73.99/74.82	76.03/76.57	60.98/62.44	63.69/64.43	67.84/68.35
TA-BML	73.37/73.59	74.02/74.63	75.52/75.83	61.43/62.59	63.28/63.96	66.27/67.49
KB-BML	74.63/75.07	75.06/75.63	80.69/81.12	65.99/66.82	67.47/68.08	73.89/74.06
AKE-BML	**88.99/89.36**	**90.10/91.48**	**91.40/92.34**	**84.55/84.94**	**86.03/87.73**	**87.13/87.45**

Table 1: Accuracy and F_1 scores of all compared methods. § denotes the results that are directly taken from the original paper (Deng et al., 2020), due to the unavailability of the source code.

corresponding event knowledge to demonstrate our model's ability to learn robust event prototypes.

5.3.1 Predictions for Specific Cases

We select three event types as target categories to illustrate the contributions of each main component of our model. The event types are *Music.Compose*, *Music.Sing* and *Film.Film_Productution*. The sample contexts of *Music.Compose* and *Music.Sing* are similar, while *Music.Compose* and *Film.Film_Productution* share the same frame, which is *Behind_the_scenes*.

As shown in Table 2, only AKE-BML correctly predicts on all samples. TA-BML, the model without introducing knowledge, wrongly predicts the second sample of *Music.Compose* to be *Music.Sing*, due to their similar contexts. By introducing knowledge, both KB-BML and AKE-BML avoid this error, indicating that external knowledge can enrich event information based on the support set. For KB-BML, as *Music.Compose* and *Film.Film_Productution* share the same superordinate frame, the prototype of *Music.Compose* cannot distinguish between *Music.Compose* and *Film.Film_Productution*, so it wrongly classifies the third sample as *Music.Compose*. With our adaptive knowledge offset, AKE-BML can deal with sample similarity and knowledge deviation issues at the same time, thus it correctly classifies all samples.

5.3.2 Visualization of Prototypes

We use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to reduce the dimensionality of the prototypes, sample encodings and prior knowledge encodings. Figure 4 visualizes five event type prototypes (large solid shapes), their aligned frames (large solid shapes with circle outlines) in FrameNet and some corresponding samples (small solid shapes). Each event type and its samples are coded with the same color.

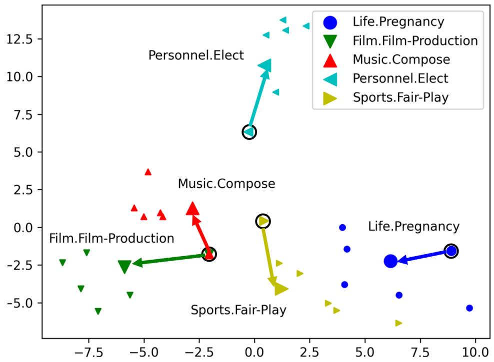

Figure 4: Visualization of event prototypes, prior knowledge, and event samples learned by AKE-BML in the 5-way-5-shot setting. The large, solid shapes denote event prototypes, the large shapes with circle outlines denote the prior knowledge, and the small shapes denote samples. Samples are marked by the color of their corresponding event types. The arrows indicate the adaption of prior knowledge to the prototype. Note that *Music.Compose* and *Film.Film-Production* share the same frame *Behind_the_scenes*.

In general, the samples and prototypes belonging to one event type are close in the space and different event types are far away from each other. Prior knowledge is distributed in different places in the space, which roughly determines the distribution of event prototypes. For example, the samples of *Life.Pregnancy* and *Sports.Fair-Play* are close to their respective event prototypes. Meanwhile, the distances between their prior knowledge is large, making their prototypes easily distinguishable.

It can also be seen that the event prototypes are closer to their samples than to the prior knowledge, which reflects the benefits of our proposed learnable knowledge offset. The visualization demonstrates the effectiveness of introducing external

No.	Queries	Event types	Corresponding frame	Prediction		
				TA-BML	KB-BML	AKE-BML
1	Since the mid-20th century, the composers directly **write** sonic material in any of the various electronic media.	Music.Compose	Behind_the_scenes	Music.Compose	Music.Compose	Music.Compose
2	Performers of Western popular music **created** earlier popular songs.	Music.Compose		Music.Sing	Music.Compose	Music.Compose
3	The film is **created** by the director, and may be storyboarded with the help of illustrators and concept artists.	Film.Film_ Production		Film.Film_ Production	Music.Compose	Film.Film_ Production
4	They came with rainbow flags and **sang** the Russian national anthem.	Music.Sing	Performing_arts	Music.Sing	Music.Sing	Music.Sing

Table 2: A case study of three event types. Words in bold indicate candidate trigger words.

Event Type	Explanation of Event Type	Examples in Support Set	λ_t	Corresponding Frame	Frame Definition
Contact.Letter-Communication	Communicate by writing letters.	(1)…*write* this letter to you… (2)…*sends* you this letter… (3)…the letter *passed* from…	0.228	Communication	A Communicator conveys a Message to an Addressee; the Topic and Medium of the communication also may be expressed.
Conflict.Attack	An assailant attacks a victim.	(1)He *shoot* them all with… (2)… *firing* at close… (3)Soldier hacked to death…	0.132	Attack	An Assailant physically attacks a Victim causing or intending to cause the Victim physical damage…
Film.Film-Production	The production process of the film.	(1)… *making* a film… (2)…*creating* the film… (3)…the action is *shot* as…	0.386	Behind_the_scenes	Artists work at various tasks that further the completion of a film, play, television show, or other similar Production
Music.Compose	Music creation process.	(1)…*composes* musical … (2)…*co-written* by him and… (3)… *create* musical sounds…	0.421		

Table 3: λ_t values corresponding to different event types. The larger the λ_t, the greater the dependence of the prototype vector on the support set.

knowledge and our adaptive knowledge offset's ability to balance the impact of the support set and prior knowledge on the event prototypes.

5.3.3 λ_t of Different Event Types

As shown in Formula (5), we use the learnable parameter λ_t to generate knowledge offsets. λ_t accounts for the deviation of the prior knowledge (i.e. a frame) from the event type it represents, and adaptively corrects this deviation using information of the support set. When the frame corresponding to the event type accurately expresses its semantics, the λ_t value should be small. When the knowledge is the super-ordinate frame of the event type (i.e., the frame cannot accurately describe the event semantics), the λ_t value should be large, so that the support set can be used to modify the prior knowledge to ensure that the prototype precisely represents the current event type.

Table 3 shows four different event types, their corresponding frames and λ_t values. The λ_t of *Conflict.Attack* is a small value 0.132, as the event type *Conflict.Attack* closely matches the frame *Attack*. The event type *Contact.Letter-Communication* matches the frame *Communication*. *Communication* does not contain the semantics of "by writing letters", but the core semantics is the same as *Contact.Letter-Communication*. Therefore, λ_t is small, at 0.228, which is still larger than the λ_t of *Conflict.Attack*. The event types *Film.Film-Production* and *Music.Compose* share the same super-ordinate frame *Behind_the_scenes* as prior knowledge, but the semantics of *Behind_the_scenes* is too abstract for *Film.Film-Production* and *Music.Compose*. Thus, it can be seen that the λ_t values corresponding to these two event types are relatively large: 0.386 for *Film.Film-Production* and 0.421 for *Music.Compose*.

The above cases demonstrate that our model is able to balance the influence of the support set and the knowledge on event prototypes through λ_t, and consequentially obtain highly accurate and generalizable prototypes.

6 Conclusion

In this paper, we proposed an Adaptive Knowledge-enhanced Bayesian Meta-Learning (AKE-BML) method for few-shot event detection. We alleviate the insufficient sample diversity problem in few-shot learning by leveraging the external knowledge base *FrameNet* to learn prototype representations for event types. We further tackle the uncertainty and incompleteness issues in knowledge coverage with a novel knowledge adaptation mechanism.

The comprehensive experimental results demonstrate that our proposed method substantially outperforms state-of-the-art methods, achieving a performance improvement of at least 15 absolute points of F_1. In the future, we plan to extend our proposed AKE-BML method to the few-shot event extraction task, which considers both event detection and argument extraction. We also plan to explore the zero-shot and incremental event extraction scenarios.

7 Acknowledgement

Research in this paper was partially supported by the Fundamental Research Funds for the Central Universities (2242021k10011).

References

Denis A. de Araujo, Sandro José Rigo, and Jorge Luis Victória Barbosa. 2017. Ontology-based information extraction for juridical events with case studies in brazilian legal realm. *Artif. Intell. Law*, pages 379–396.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL*, pages 86–90.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, (Jan):993–1022.

Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL&IJCNLP*, pages 167–176.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *Proceedings of WSDM*, pages 151–159.

Shumin Deng, Ningyu Zhang, Luoqiu Li, Hui Chen, Huaixiao Tou, Mosha Chen, Fei Huang, and Huajun Chen. 2021. Ontoed: Low-resource event detection with ontology embedding. In *ACL*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Charles J Fillmore, Srini Narayanan, and Collin F Baker. 2006. What can linguistics contribute to event extraction. In *Proceedings of AAAI*, pages 18–23.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Victor Garcia and Joan Bruna. 2012. Few-shot learning with graph neural networks. *ICLR*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020a. Exploiting the matching information in the support set for few shot event classification. In *Proceedings of PAKDD*, pages 233–245.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020b. Extensively matching for few-shot learning event detection. *arXiv preprint arXiv:2006.10093*.

Jian Liu, Yubo Chen, and Jun Zhao. 2020. Knowledge enhanced event causality identification with mention masking generalizations. In *Proceedings of IJCAI*, pages 3608–3614.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of ACL*, pages 2134–2143.

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019. Event detection without triggers. In *Proceedings of NAACL-HLT*, pages 735–744.

Dongfang Lou, Zhilin Liao, Shumin Deng, Ningyu Zhang, and Huajun Chen. 2021. Mlbinet: A cross-sentence collective event detection network. In *ACL*. Association for Computational Linguistics.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of NAACL-HLT*, pages 1626–1635.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Meng Qu, Tianyu Gao, Louis-Pascal Xhonneux, and Jian Tang. 2020. Few-shot relation extraction via Bayesian meta-learning on relation graphs. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7867–7876. PMLR.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. *ICLR*.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Rbpb: Regularization-based pattern balancing method for event extraction. In *Proceedings of ACL*, pages 1224–1234.

Shirong Shen, Guilin Qi, Zhen Li, Sheng Bi, and Lusheng Wang. 2020. Hierarchical chinese legal event extraction via pedal attention mechanism. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 100–113.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NeurIPS*, pages 4077–4087.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of NeurIPS*, pages 2440–2448.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of CVPR*, pages 1199–1208.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of ACL*, pages 5887–5897.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of NeurIPS*, pages 3630–3638.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations.

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of ICML*, pages 681–688.

Wang Yan, Jordan Yap, and Greg Mori. 2015. Multi-task transfer methods to improve one-shot learning for multimedia event detection. In *Proceedings of BMVC*, pages 37.1–37.13.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of ACL*, pages 5284–5294.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. 2018. Bayesian model-agnostic meta-learning. In *Proceedings of NeurIPS*, pages 7332–7342.

Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. 2018a. Metagan: An adversarial approach to few-shot learning. In *Proceedings of NeurIPS*, pages 2371–2380.

Yabin Zhang, Hui Tang, and Kui Jia. 2018b. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In *Proceedings of ECCV*, pages 241–256.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2edag: An end-to-end document-level framework for chinese financial event extraction. In *Proceedings of EMNLP*, pages 337–346.

Deyu Zhou, Xuan Zhang, and Yulan He. 2017. Event extraction from twitter using non-parametric bayesian mixture model with word embeddings. In *Proceedings of EACL*, pages 808–817.

Appendix

A FrameNet

An important problem in the few-shot event detection task is the insufficient diversity of support set samples. There are only a few labeled samples in the support set, which results in the model unable to construct high-quality prototype features of event types. To address this problem, we introduce the *FrameNet* (Baker et al., 1998) as an external knowledge base of event types. *FrameNet* is a linguistic resource storing information about lexical and predicate-argument semantics. Each frame in *FrameNet* can be taken as a semantic frame of an event type (Liu et al., 2016), which can be used as background knowledge for event types to assist event detection (Liu et al., 2016; Fillmore et al., 2006). Figure 2 shows an example frame defining *Attack*. We can see the arguments involved in an *Attack* event and their roles. The linguistic units (LUs) of the frame *Attack* are the possible trigger words for the corresponding event. The frame is an important complementary source of knowledge to the support set. We match a frame in *FrameNet* to each event type, based on the event name, as its knowledge. In practice, *FrameNet* does not provide complete coverage of all event types, nor does every event type have an exact frame matched in *FrameNet*. For event types that cannot be exactly matched, we assign the frame corresponding to their super-ordinate event. For example, there is no corresponding frame for *Contact.Online-Chat*, so we assign it to the frame *Chatting*, which corresponds to the event type *Contact.Chat*.

B Gradient of posterior distribution

In order to show the change of the prototype vector after adding the knowledge shift, the gradient $\nabla_{\hat{\mathbf{V}}_{T_S}} \log\ p(Y_S|X_S, \hat{\mathbf{V}}_{T_S}) p(\hat{\mathbf{V}}_{T_S}|\mathscr{F})$ in iteration

$$\begin{aligned}\hat{\mathbf{V}}_{T_S} \quad &\leftarrow \hat{\mathbf{V}}_{T_S} + \sqrt{\epsilon}\boldsymbol{z} \\ &+ \frac{\epsilon}{2}\nabla_{\hat{\mathbf{V}}_{T_S}} \log\ p(Y_S|X_S, \hat{\mathbf{V}}_{T_S}) p(\hat{\mathbf{V}}_{T_S}|\mathscr{F}),\end{aligned} \tag{12}$$

is expanded. For ease of explanation, we only calculate the gradient of the prototype vector $\hat{\mathbf{v}}_t$. We denote the gradient of the original posterior distribution as $\mathbf{g}^o_{\hat{\mathbf{v}}_t}$, the gradient of the knowledge-shifted posterior distribution as $\mathbf{g}^s_{\hat{\mathbf{v}}_t}$. We first calculate $\mathbf{g}^o_{\hat{\mathbf{v}}_t}$:

$$\begin{aligned}\mathbf{g}^o_{\hat{\mathbf{v}}_t} =& \nabla_{\hat{\mathbf{v}}_t} \log\ p(Y_S|X_S, \hat{\mathbf{V}}_{T_S}) p^o(\hat{\mathbf{V}}_{T_S}|\mathscr{F}) \\ =& \nabla_{\hat{\mathbf{v}}_t} \log\ p(Y_S|X_S, \hat{\mathbf{V}}_{T_S}) + \log\ p^o(\hat{\mathbf{V}}_{T_S}|\mathscr{F}) \\ =& \nabla_{\hat{\mathbf{v}}_t} \log\ p(Y_S|X_S, \hat{\mathbf{V}}_{T_S}) + \\ & \nabla_{\hat{\mathbf{v}}_t} \log\ p^o(\hat{\mathbf{V}}_{T_S}|\mathscr{F}) \\ =& \nabla_{\hat{\mathbf{v}}_t} \log \prod_{s\in S, y_s=t} p(y_s|x_s, \hat{\mathbf{v}}_t) + \\ & \nabla_{\hat{\mathbf{v}}_t} \log\ p^o(\hat{\mathbf{v}}_t|\mathscr{F}) \\ =& \mathbf{g}^{o,l}_{\hat{\mathbf{v}}_t} + \mathbf{g}^{o,p}_{\hat{\mathbf{v}}_t},\end{aligned} \tag{13}$$

where $\mathbf{g}^{o,l}_{\hat{\mathbf{v}}_t} = \nabla_{\hat{\mathbf{v}}_t} \log \prod_{s\in S, y_s=t} p(y_s|x_s, \hat{\mathbf{v}}_t)$ and $\mathbf{g}^{o,p}_{\hat{\mathbf{v}}_t} = \nabla_{\hat{\mathbf{v}}_t} \log\ p^o(\hat{\mathbf{v}}_t|\mathscr{F})$. The prior distribution is

$$\begin{aligned}p^o(\hat{\mathbf{v}}_t|\mathscr{F}) &= \mathcal{N}(\mathbf{v}_t|\mathbf{h}_t, \boldsymbol{I}) \\ &= (2\pi)^{-\frac{d}{2}} e^{-\frac{1}{2}(\hat{\mathbf{v}}_t - \mathbf{h}_t)^2},\end{aligned} \tag{14}$$

the gradient of the logarithm of prior distribution to $\hat{\mathbf{v}}_t$ is:

$$\begin{aligned}\mathbf{g}^{o,p}_{\hat{\mathbf{v}}_t} &= \left(\log (2\pi)^{-\frac{d}{2}}\right)(\mathbf{h}_t - \hat{\mathbf{v}}_t) \\ &= C\left(\mathbf{h}_t - \hat{\mathbf{v}}_t\right),\end{aligned} \tag{15}$$

where $C = \log (2\pi)^{-\frac{d}{2}}$ is a constant, d is the dimension of prototype. The gradient of the log-likelihood on support set is

$$\begin{aligned}&\log \prod_{s\in S, y_s=t} p(y_s|x_s, \hat{\mathbf{v}}_t) \\ &= \log \prod_{s\in S, y_s=t} \frac{exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_t)}{\sum_{t'\in T_S} exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_{t'})} \\ &= \sum_{s\in S, y_s=t} \log \frac{exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_t)}{\sum_{t'\in T_S} exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_{t'})}.\end{aligned} \tag{16}$$

The gradient of the log-likelihood to $\hat{\mathbf{v}}_t$ is

$$\begin{aligned}\mathbf{g}^{o,l}_{\hat{\mathbf{v}}_t} =& \nabla_{\hat{\mathbf{v}}_t} \sum_{s\in S, y_s=t} \log \frac{exp\left(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_t\right)}{\sum_{t'\in T_S} exp\left(\left(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_{t'}\right)\right.} \\ =& \sum_{s\in S, y_s=t} \nabla_{\hat{\mathbf{v}}_t} \log \frac{exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_t)}{\sum_{t'\in T_S} exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_{t'})} \\ =& \sum_{s\in S, y_s=t} \nabla_{\hat{\mathbf{v}}_t}(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_t) \\ & - \nabla_{\hat{\mathbf{v}}_t} \log \sum_{t'\in T_S} exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_{t'}) \\ =& \sum_{s\in S, y_s=t} \boldsymbol{\mathcal{E}}(x_s) - \frac{\boldsymbol{\mathcal{E}}(x_s) exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_t)}{\sum_{t'\in T_S} exp(\boldsymbol{\mathcal{E}}(x_s)\cdot\hat{\mathbf{v}}_{t'})} \\ =& \sum_{s\in S, y_s=t} (1 - p^{(t)}_s)\boldsymbol{\mathcal{E}}(x_s),\end{aligned} \tag{17}$$

where $p_s^{(t)} = p(y_s|x_s, \hat{\mathbf{v}}_t)$ is the probability of correct classification of sample s in support set. Then we get

$$\mathbf{g}^o_{\hat{\mathbf{v}}_t} = \sum_{s\in S, y_s=t} (1-p_s^{(t)})\boldsymbol{\mathcal{E}}(x_s) + C(\mathbf{h}_t - \hat{\mathbf{v}}_t). \tag{18}$$

Then we calculate $\mathbf{g}^s_{\hat{\mathbf{v}}_t}$, The only difference between calculating $\mathbf{g}^s_{\hat{\mathbf{v}}_t}$ and $\mathbf{g}^o_{\hat{\mathbf{v}}_t}$ is $\mathbf{g}^s_{\hat{\mathbf{v}}_t}$ use the knowledge-shifted prior distribution

$$\begin{aligned} p(\hat{\mathbf{v}}_t|\mathscr{F}) &= \mathcal{N}(\mathbf{v}_t|\mathbf{h}_t + \Delta\mathbf{h}_t, \boldsymbol{I}) \\ &= (2\pi)^{-\frac{d}{2}}\, e^{-\frac{1}{2}(\hat{\mathbf{v}}_t - \mathbf{h}_t - \Delta\mathbf{h}_t)^2}. \end{aligned} \tag{19}$$

Same as the original posterior gradient, we have

$$\begin{aligned} \mathbf{g}^s_{\hat{\mathbf{v}}_t} =& \nabla_{\hat{\mathbf{v}}_t} \log \prod_{s\in S, y_s=t} p(y_s|x_s, \hat{\mathbf{v}}_t) \\ &+ \nabla_{\hat{\mathbf{v}}_t} \log\ p(\hat{\mathbf{v}}_t|\mathscr{F}) \\ =& \mathbf{g}^{s,l}_{\hat{\mathbf{v}}_t} + \mathbf{g}^{s,p}_{\hat{\mathbf{v}}_t}, \end{aligned} \tag{20}$$

where $\mathbf{g}^{s,l}_{\hat{\mathbf{v}}_t} = \mathbf{g}^{o,l}_{\hat{\mathbf{v}}_t} = \sum_{s\in S, y_s=t}(1-p_s^{(t)})\boldsymbol{\mathcal{E}}(x_s)$. The gradient of the logarithm of knowledge-shifted prior distribution to $\hat{\mathbf{v}}_t$ is:

$$\begin{aligned} \mathbf{g}^{s,p}_{\hat{\mathbf{v}}_t} &= \left(\log\,(2\pi)^{-\frac{d}{2}}\right)(\mathbf{h}_t + \Delta\mathbf{h}_t - \hat{\mathbf{v}}_t) \\ &= C\left(\mathbf{h}_t + \boldsymbol{\lambda}_t \odot (\mathbf{m}_t - \mathbf{h}_t) - \hat{\mathbf{v}}_t\right) \\ &= C\left((\mathbb{1} - \boldsymbol{\lambda}_t)\odot \mathbf{h}_t - \hat{\mathbf{v}}_t\right) + C\boldsymbol{\lambda}_t \odot \mathbf{m}_t, \end{aligned} \tag{21}$$

where $\mathbb{1}$ is a $|\mathbf{h}_t|$-dimensional vector, and each element of $\mathbb{1}$ is 1. then we get

$$\begin{aligned} \mathbf{g}^s_{\hat{\mathbf{v}}_t} =& \mathbf{g}^{s,l}_{\hat{\mathbf{v}}_t} + \mathbf{g}^{s,p}_{\hat{\mathbf{v}}_t} \\ =& \sum_{s\in S, y_s=t} (1-p_s^{(t)})\boldsymbol{\mathcal{E}}(x_s) \\ &+ C\left((\mathbb{1} - \boldsymbol{\lambda}_t)\odot \mathbf{h}_t - \hat{\mathbf{v}}_t\right) \\ &+ C\boldsymbol{\lambda}_t \odot \mathbf{m}_t, \end{aligned} \tag{22}$$

bring $\mathbf{m}_t = \frac{1}{M}\sum_{s\in S, y_s=t}\boldsymbol{\mathcal{E}}(x_s)$ into the above formula, we get

$$\begin{aligned} \mathbf{g}^s_{\hat{\mathbf{v}}_t} =& \sum_{s\in S, y_s=t}\left[(1-p_s^{(t)})\boldsymbol{\mathcal{E}}(x_s) + \frac{C\boldsymbol{\lambda}_t}{M}\odot \boldsymbol{\mathcal{E}}(x_s)\right] \\ &+ C\left((\mathbb{1} - \boldsymbol{\lambda}_t)\odot \mathbf{h}_t - \hat{\mathbf{v}}_t\right). \end{aligned} \tag{23}$$

Note that, when knowledge adaption is not used, the form of the prior knowledge distribution of the prototype is as follows,

$$p^o(\mathbf{V}_{T_S}|\mathscr{F}) = \prod_{t\in T_S} p^o(\mathbf{v}_t|\mathbf{h}_t) = \prod_{t\in T_S}\mathcal{N}(\mathbf{v}_t|\mathbf{h}_t, \boldsymbol{I}). \tag{24}$$

To intuitively show the influence of the knowledge-adapted posterior distribution on the prototype vector, we expand the gradient $\nabla_{\hat{\mathbf{V}}_{T_S}} \log\ p(Y_S|X_S, \hat{\mathbf{V}}_{T_S})p(\hat{\mathbf{V}}_{T_S}|\mathscr{F})$ in Eqn 12. For ease of explanation, we only calculate the gradient of the prototype vector $\hat{\mathbf{v}}_t$. Denote the gradient of the original posterior distribution without knowledge adaption as $\mathbf{g}^o_{\hat{\mathbf{v}}_t}$,

$$\mathbf{g}^o_{\hat{\mathbf{v}}_t} = \sum_{s\in S, y_s=t} (1-p_s^{(t)})\boldsymbol{\mathcal{E}}(x_s) + C(\mathbf{h}_t - \hat{\mathbf{v}}_t), \tag{25}$$

and the gradient of the knowledge-adapted posterior distribution as $\mathbf{g}^s_{\hat{\mathbf{v}}_t}$ from Eqn 23.

Comparing Eqn 23 and Eqn 25, it can be seen that the posterior distribution without knowledge adaption cannot dynamically balance the influence of the knowledge and the support set on the prototype vector, whereas the knowledge-adapted posterior distribution can adjust their contributions to the prototype vector through $\boldsymbol{\lambda}_t$. The parameters in Eqn 6 will be updated by the log-likelihood on the query set. This allows the model to reasonably choose the weight of the knowledge and the support set, and obtain prototype vectors with better generalization.

C M-shot Evaluation

In this section, we illustrate the effectiveness of adaptive knowledge-enhanced Bayesian meta-learning under different M-shot settings, such as N-way-5-shot, N-way-10-shot and N-way-15-shot. As shown in Table 1 in the main paper, as M increases, the performance of all models improves, which shows that increasing the number of samples in the support set can provide more pertinent event type-related features. At the same time, it can be seen that from 15-shot to 5-shot, the previous methods suffer a significantly larger performance degradation than AKE-BML. This observation shows our model's strong robustness against low sample diversity due to the incorporation of external knowledge.

The performance of KB-BML is close to that of DMN-MPN in the case of N-way-5-shot and N-way-10-shot, and the performance is better in the case of N-way-15-shot. This can be attributed to two factors: (1) the introduction of knowledge can improve the generalization of event prototypes; and (2) increasing the number of samples can reduce the impact of the deviation between knowledge and

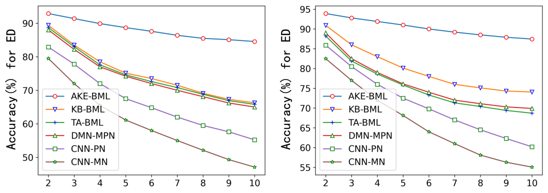

(a) x-way-5-shot settings (b) x-way-15-shot settings

Figure 5: N-way ($N = 2, \ldots, 10$) evaluation and fixed shot numbers. (a) N-way-5-shot. (b) N-way-15-shot.

event types. When the support set is sufficiently large, the samples in the support set can compensate for the deviation between knowledge and event types, and the knowledge can also improve the generalization of the prototype vector. However, when M is small, the deviation between knowledge and event types will affect the quality of the prototype vectors.

AKE-BML can well balance the effects of samples and knowledge on the event type prototypes. It can be seen that when M is small, the performance of AKE-BML does not decline as quickly as other models, which also proves the effectiveness of knowledge in dealing with the problem of insufficient diversity of the support set. At the same time, compared with KB-BML, our adaptive knowledge offset can effectively use the information in the support set to correct the knowledge deviation.

D N-Way Evaluation

Figure 5 also illustrates model performance with respect to different way values (i.e. N), while fixing the shot values. It can be seen from the figure that when N increases, the performance of previous models decreases faster than AKE-BML, which shows that those models, only relying on the support set, cannot generate more recognizable event prototypes. The performance of KB-BML also declines significantly when N increases. This is because many event types can only be partially aligned in FrameNet, to its super-ordinate frame, which causes the event prototypes to be indistinguishable to similar event types.

On the contrary, the performance of AKE-BML does not decrease significantly when N increases, which shows that our adaptive knowledge-enhanced Bayesian meta-learning method can enhance the distinguishability of prototype vectors through the learnable knowledge offset. These results indicate that our adaptive knowledge-enhanced Bayesian meta-learning is more robust to the changes in the number of ways.

Stylized Story Generation with Style-Guided Planning

Xiangzhe Kong*, **Jialiang Huang***, **Ziquan Tung, Jian Guan and Minlie Huang**†
The CoAI group, DCST, Institute for Artificial Intelligence,
State Key Lab of Intelligent Technology and Systems,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing 100084, China
{kxz18,huang-jl17,tongzq18,j-guan19}@mails.tsinghua.edu.cn,
aihuang@tsinghua.edu.cn

Abstract

Current storytelling systems focus more on generating stories with coherent plots regardless of the narration style, which is important for controllable text generation. Therefore, we propose a new task, stylized story generation, namely generating stories with specified style given a leading context. To tackle the problem, we propose a novel generation model that first plans the stylized keywords and then generates the whole story with the guidance of the keywords. Besides, we propose two automatic metrics to evaluate the consistency between the generated story and the specified style. Experiments demonstrates that our model can controllably generate *emotion*-driven or *event*-driven stories based on the ROCStories dataset (Mostafazadeh et al., 2016). Our study presents insights for stylized story generation in further research.

1 Introduction

Story generation is a challenging task in natural language generation (NLG), namely generating a reasonable story given a leading context. Recent work focuses on enhancing the coherence of generated stories (Fan et al., 2018; Yao et al., 2019) or introducing commonsense knowledge (Guan et al., 2020; Xu et al., 2020). However, it has not yet been investigated to generate stories with controllable styles, which is important since different styles serve different writing purposes. As exemplified in Figure 1, *emotion-driven* stories use emotional words (e.g., *"excited"*, *"enjoyed"*) to reveal the inner states of the characters and bring the readers closer to the characters. In comparison, *event-driven* stories usually contain a sequence of events with a clear temporal order (e.g., *"tearing"*→*"tried"*→*"found"*→*"hooked"*), which aims to narrate the story objectively.

Leading Context: Alice bought a new television.

Style	Story
Emotion-driven	She got **excited** to have it. It was **pretty** for the extra large screen. After using, she was very **satisfied** with the sound and design. She didn't **regret** buying this TV.
Event-driven	The picture on screen was **tearing**. Then she **tried** different adjustments. Eventually, she **found** out what was wrong. She got the wrong cable **hooked** up.

Figure 1: Example of stylized story generation given the same leading context. The stylized keywords are in bold.

In this paper, we formalize the task of stylized story generation, which requires generating a coherent story with a specified style given the first sentence as the leading context. *Style* has multiple interpretations, which can be seen as a unique voice of the author expressed through the use of certain stylistic devices (e.g. choices of words)(Mou and Vechtomova, 2020). In this work we focus on the choices of words and define the story styles based on the pattern of wording. Specifically, we focus on two story styles, including *emotion-driven* and *event-driven* stories. *Emotion-driven* stories contain abundant words with emotional inclination. We identify the emotional words using the off-the-shelf toolkit NRCLex (Mohammad, 2020), which supports retrieving the emotional effects of a word from a predefined lexicon. And *event-driven* stories tend to use serial actions as an event sequence. We use NLTK (Bird et al., 2009) to extract verbs in a story as the actions. Since no public datasets are available for learning to generate stylized stories, we regard the extracted words as stylistic keywords and then annotate the story styles for existing story datasets automatically based on the keyword distribution. Note that the story styles can be extended easily by defining new stylistic keywords.

In this work, we propose a generation model

---

*Equal contribution.

†Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2430–2436
August 1–6, 2021. ©2021 Association for Computational Linguistics

for stylized story generation. Our model first predicts the distribution of stylistic keywords and then generates a story with the guidance of the distribution. Furthermore, we propose two new automatic metrics to evaluate the consistency between the generated stories and the specified styles: lexical style consistency (LSC) and semantic style consistency (SSC), which focus on the number of stylistic keywords and the overall semantics, respectively. Extensive experiments demonstrate that the stories generated by our model not only achieve better fluency and coherence than strong baselines but also have better consistency with the specified styles.[1]

2 Related Work

Story Generation Recently there have been significant advances for story generation with the encoder-decoder paradigm (Sutskever et al., 2014), the transformer-based architecture (Vaswani et al., 2017) and the large-scale pre-trained models (Radford et al., 2019; Lewis et al., 2020). Prior studies usually decomposed the generation into separate steps by first planning a sketch and then generating the whole story from the sketch. The sketch is usually a series of keywords (Yao et al., 2019), a learnable skeleton (Xu et al., 2018) or an action sequence (Fan et al., 2019; Goldfarb-Tarrant et al., 2020). Another line is to incorporate external knowledge into story generation (Guan et al., 2020; Xu et al., 2020). However, generating stories with controllable styles has hardly been investigated.

Stylized Generation Stylized generation aims to generate texts with controllable attributes. For example, recent studies in dialogue systems focused on controlling persona (Zhang et al., 2018; Boyd et al., 2020), sentence functions (Ke et al., 2018), politeness (Niu and Bansal, 2018), and topics (Tang et al., 2019). In story generation, Huang et al. (2019) and Xu et al. (2020) controlled the story topics and planned keywords, respectively. Besides, for general text generation, the authorship (Tikhonov and Yamshchikov, 2018), sentiment (Hu et al., 2017), and topics (Li et al., 2020) can also be controlled for different purposes. We introduce a new controllable attribute in story generation, i.e., the story style, which has been paid little attention to in prior studies.

3 Proposed Method

In this section, we first show the task formulation for stylized story generation (§3.1). Then we present the details of our two-step model: **style-guided keywords planning** (§3.2) and **generation with planned keywords** (§3.3).

3.1 Task Formulation

Input: The first sentence $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ of a story with length n, where x_i is the i-th word. A special token l to indicate the expected style of the generated story. $l \in \{\langle\texttt{emo}\rangle, \langle\texttt{eve}\rangle\}$, which refers to the *emotion-driven* and *event-driven* styles, respectively. Besides, in the training phase, we set $l = \langle\texttt{other}\rangle$ if the training example is neither emotion-driven nor event-driven to improve the data efficiency.
Output: A story $\boldsymbol{y} = (y_1, y_2, \dots, y_m)$ of length m with the style l, where y_i is i-th word.

3.2 Planning

We insert l at the beginning of $\boldsymbol{x}$ and encode them as follows:

$$[\boldsymbol{h}_0, \boldsymbol{h}_1, \dots, \boldsymbol{h}_n] = \texttt{Enc}(l, x_1, x_2, \dots, x_n), \quad (1)$$

where $\boldsymbol{h}_i$ $(1 \leqslant i \leqslant n)$ is the hidden state corresponding to x_i, $\boldsymbol{h}_0$ is the hidden state at the position of l, and $\texttt{Enc}$ is a bidirectional or unidirectional encoder. Then, we regard the stylistic keywords as bag-of-words (Kang and Hovy, 2020) and predict the keyword distribution $P_k(w|\boldsymbol{x}, l)$ over the whole vocabulary $\mathbb{V}$ as follows:

$$P_k(w|\boldsymbol{x}, l) = \text{softmax}(\mathbf{W}_k \boldsymbol{h}_c + \mathbf{b}_k), \quad (2)$$

where $\mathbf{W}_k$ and $\mathbf{b}_k$ are trainable parameters, and $\boldsymbol{h}_c$ is the context embedding to summarize the input information. We directly set $\boldsymbol{h}_c = \boldsymbol{h}_0$. The training objective in this stage is to minimize the cross-entropy loss $\mathcal{L}_k$ between the predicted keyword distribution $P_k(w|l, \boldsymbol{x})$ and the ground truth $\hat{P}_k(w|l, \boldsymbol{x})$ as follows:

$$\mathcal{L}_k = -\sum_{i=1}^{|\mathbb{V}|} \hat{P}_k(w_i|l, \boldsymbol{x}) \log P_k(w_i|l, \boldsymbol{x}), \quad (3)$$

where w_i denotes the i-th word in $\mathbb{V}$ and $\hat{P}_k(w|l, \boldsymbol{x})$ is an one-hot vector over $\mathbb{V}$. We do not decode a keyword sequence explicitly (Yao et al., 2019) but generate stories directly based on the keyword distribution $P_k(w|l, \boldsymbol{x})$ to avoid introducing extra exposure bias (He et al., 2019).

---

[1] Link to the code: `https://github.com/thu-coai/Stylized-Story-Generation-with-Style-Guided-Planning.git`

3.3 Generation

We employ a left-to-right decoder to generate a story conditioned upon the input and the predicted keyword distribution. The training objective in this stage is to minimize the negative log-likelihood $\mathcal{L}_{st}$ of the ground truth stories:

$$\mathcal{L}_{st} = -\sum_{t=1}^{m} \log P(y_t|l, \boldsymbol{x}, y_{<t}). \tag{4}$$

We derive $P(y_t|l, \boldsymbol{x}, y_{<t})$ by explicitly combining the stylistic keyword distribution into the decoding process as follows:

$$\begin{aligned} P(y_t|l, \boldsymbol{x}, y_{<t}) &= P_l(y_t|l, \boldsymbol{x}, y_{<t}) \cdot (1 - \boldsymbol{g}_t) \\ &\quad + P_k(y_t|l, \boldsymbol{x}) \cdot \boldsymbol{g}_t, & (5) \\ P_l(y_t|l, \boldsymbol{x}, y_{<t}) &= \text{softmax}(\mathbf{W}_s \boldsymbol{s}_t + \mathbf{b}_s), & (6) \\ \boldsymbol{s}_t &= \texttt{Dec}(y_{<t}, \{\boldsymbol{h}_i\}_{i=0}^{n}), & (7) \end{aligned}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are trainable parameters, P_l is a distribution over $\mathbb{V}$ without conditioning on the predicted keywords, and $\boldsymbol{g}_t \in \mathbb{R}^{|\mathbb{V}|}$ is a gate vector indicating the weight of the keyword distribution P_k. We compute $\boldsymbol{g}_t$ as follows:

$$\begin{aligned} \boldsymbol{g}_t &= \text{sigmoid}(\mathbf{W}_g[\boldsymbol{r}_t; \boldsymbol{s}_t] + \boldsymbol{b}_g), & (8) \\ \boldsymbol{r}_t &= \boldsymbol{W}_r P_k(y_t|l, \boldsymbol{x}) + \boldsymbol{b}_r, & (9) \end{aligned}$$

where $\boldsymbol{W}_g$, $\boldsymbol{b}_g$, $\boldsymbol{W}_r$ and $\boldsymbol{b}_r$ are trainable parameters. In summary, the final training objective $\mathcal{L}$ of our model is derived as follows:

$$\mathcal{L} = \mathcal{L}_{st} + \alpha \cdot \mathcal{L}_k. \tag{10}$$

where α is an adjustable scale factor.

4 Experimental Setup

4.1 Dataset

We conduct the experiments on the ROCStories corpus (Mostafazadeh et al., 2016), which contains 98,159 five-sentence stories. We randomly split ROCStories by 8:1:1 for training/validation/test, respectively. The average number of words in the input (the first sentence) and the output (the last four sentences) are 9.1 and 40.8, respectively. Besides, we follow Guan et al. (2020) to delexicalize stories in the dataset by masking all the male /female/neutral names with $\langle$`MALE`$\rangle$/ $\langle$`FEMALE`$\rangle$/ $\langle$`NEUTRAL`$\rangle$ to achieve better generalization.

4.2 Style Annotation

We extract stylistic keywords from stories in the dataset and assign a style label for each story according to the distribution of stylistic keywords.

Stylistic Keywords We use NRCLex and NLTK to extract stylistic keywords. NRCLex maps each word in a story to its underlying emotion labels according to a word-emotion lexicon (e.g., *"favorite"* → *"joy"*). We select the words with following emotion labels: *"fear"*, *"anger"*, *"surprise"*, *"sadness"*, *"disgust"* and *"joy"*, as the keywords for the emotion-driven style. Besides, we use NLTK to extract verbs as keywords for the event-driven style. We filter out the stop words and common verbs with bottom ten IDF[2] (e.g., *"is"*, *"have"*) from the extracted verbs. Intuitively, the more stylistic keywords of some style a story has, the more consistent it is with that style. Therefore, we propose to compare the numbers of keywords for different styles for style annotation.

Normalized Numbers of Keywords Let N_s denote the number of keywords for style s in a story. We assume N_s is a random variable, and follows a Gaussian distribution $\mathcal{N}(\mu_s, \sigma_s^2)$, where μ_s and σ_s are the mean and standard deviation computed on the training set. Given a story which contains n_s keywords for style s, we normalize n_s to $n'_s = P(N_s \leqslant n_s) \in [0, 1]$ for fair comparison between keywords for different styles.

Styles	Training	Validation	Test
Emotion-driven	17.7%	18.0%	17.9%
Event-driven	17.6%	17.0%	17.5%
Others	64.7%	65.0%	64.6%

Table 1: Distribution of stories annotated with different style for the training/validation/test set.

Annotation We annotate the style label l for a given story by comparing its n'_{emo} and n'_{eve}, which refer to the normalized numbers of keywords for emotion-driven and event-driven styles, respectively. We annotate the story with $\langle$`emo`$\rangle$ if n'_{emo} is higher than n'_{eve}, and $\langle$`eve`$\rangle$ otherwise. However, if both n'_{emo} and n'_{eve} are lower than τ_1, or $|n'_{\text{emo}} - n'_{\text{eve}}| < \tau_2$, we annotate the story with $\langle$`other`$\rangle$ since there is no significant tendency to any styles. τ_1 and τ_2 are hyper-parameters, which are set to 0.7 and 0.3, respectively. For stories labeled with $\langle$`other`$\rangle$, we select five words as the stylistic keywords from those keywords for emotion-driven and event-driven styles. Table 1 shows the stylistic distribution of the dataset.

---

[2] Inverse Document Frequency (IDF) is statistically analyzed on the stems of all the extracted keywords by NLTK.

4.3 Baselines and Experiment Settings

We compare our model with GPT-2 (Radford et al., 2019) and BART (Lewis et al., 2020) as baselines. We fine-tune the baselines on ROCStories with the style tokens and the beginnings as input.

We build our model based on BART. Our approach can easily adapt to other pre-trained models such as BERT. We set the scale factor in Equation 10 to 0.2. For all models, We generate stories using top-k sampling (Fan et al., 2018) with $k = 50$ and a softmax temperature of 0.8.

4.4 Automatic Evaluation

Evaluation Metrics We use the following metrics for automatic evaluation: **(1) Perplexity (PPL)**. Since the automatically annotated style labels may contain innate bias, we do not calculate the perplexity conditioned on the annotated styles for the stories in the test set. Instead, we calculate the perplexity of a model for each sample conditioned on two styles (emotion-driven and event-driven), respectively, and then get the perplexity on the entire test set by averaging the smaller perplexity for each sample.**(2) BLEU (B-n)** (Papineni et al., 2002)**:** The metric evaluates n-gram overlap ($n = 1, 2$). For each beginning in the test set, we generate two stories conditioned on two styles, respectively. Then we calculate the BLEU score on the test set by averaging the higher BLEU with the reference story for each sample.**(3) Distinct (D-n)** (Li et al., 2016)**:** The metric measures the generation diversity with the percentage of unique n-grams ($n = 1, 2$). **(4) Numbers of Stylistic Keywords (Number):** We use the average n' (described in §4.2) to evaluate how many consistent stylistic keywords the generated stories have. **(5) Lexical Style Consistency (LSC):** We calculate the percentage of the stories annotated with the consistent style in all generated stories using the annotation strategy described in §4.2. **(6) Semantic Style Consistency (SSC):** It is a learnable automatic metric (Guan and Huang, 2020). We fine-tune BERT$_{\text{BASE}}$ on the training set as a classifier to distinguish whether a story is emotion-driven, event-driven, or others with the automatic labels as the golden truth. For each style, we regard the average classification score on the style to measure the style consistency. Table 2 shows the accuracy and F1-Scores of the BERT model on the test set.

Results We show the evaluation results of PPL and BLEU in Table 3. Note that we do not provide PPL for GPT-2 since it does not adopt the same vocabulary used in BART. We can see that our model has lower perplexity and higher word overlap with the human-written stories than baselines.

Accuracy	**F1-Score**		
	Emotion-Driven	**Event-Driven**	**Other**
89.7%	0.863	0.838	0.922

Table 2: Accuracy and F1-Scores for each class of the BERT used in SSC.

Models	**PPL ↓**	**B-1 ↑**	**B-2 ↑**
GPT-2	N/A	32.8	16.1
BART	11.72	33.2	16.6
Ours	**11.29**	**33.8**	**17.1**

Table 3: Automatic evaluation results on the entire test set. The Best results are highlighted in **bold**. ↓/↑ indicates the lower/higher, the better.

We present the results of diversity and style consistency on the generated stories with different specified styles in Table 4. Our model achieves comparable diversity with baselines, generates more keywords of the specified styles, and outperforms baselines in both lexical and semantic style consistency by a large margin.

Models	**D-1 ↑**	**D-2 ↑**	**Number ↑**	**LSC↑**	**SSC ↑**
Emotion-driven Style					
GPT-2	0.679	0.924	0.454	0.243	0.201
BART	**0.701**	**0.952**	0.538	0.366	0.298
Ours	0.697	**0.952**	**0.623**	**0.474**	**0.371**
Event-driven Style					
GPT-2	0.675	0.925	0.375	0.107	0.088
BART	0.697	0.954	0.460	0.162	0.129
Ours	**0.698**	**0.955**	**0.591**	**0.309**	**0.293**

Table 4: Automatic evaluation results. The best results are highlighted in **bold**.

4.5 Manual Evaluation

We conduct a pairwise comparison between our model and baselines. We randomly generate 100 stories from the test set for each style and model. For each pair of stories (one by ours, and the other by a baseline), we hire three annotators to give a preference (*win*, *lose* and *tie*) in terms of fluency, coherence, and style consistency. We adopt majority voting to make the final decisions among the annotators. We resort to Amazon Mechanical Turk for manual annotation. As shown in Table 5, all the results show moderate ($0.4 \leqslant \kappa \leqslant 0.6$)

Styles	Models	Fluency				Coherence				Style Consistency			
		Win(%)	Lose(%)	Tie(%)	κ	Win(%)	Lose(%)	Tie(%)	κ	Win(%)	Lose(%)	Tie(%)	κ
Emotion-driven	**Ours** vs. **GPT-2**	37.0**	19.0	44.0	0.751	52.0**	25.0	23.0	0.803	39.0*	22.0	39.0	0.773
	Ours vs. **BART**	51.0**	22.0	27.0	0.672	52.0**	16.0	32.0	0.591	50.0*	31.0	19.0	0.640
Event-driven	**Ours** vs. **GPT-2**	40.0*	25.0	35.0	0.754	50.0*	30.0	20.0	0.735	54.0**	28.0	18.0	0.726
	Ours vs. **BART**	43.0*	25.0	32.0	0.644	45.0**	22.0	33.0	0.635	48.0*	33.0	19.0	0.612

Table 5: Manual evaluation results. The scores indicates the percentage of *win*, *lose* or *tie* when comparing our model with a baseline. κ denotes Fleiss' Kappa (Fleiss, 1971) to measure the inter-annotator agreement. * and ** mean p-value<0.05 and p-value<0.01 (Wilcoxon signed-rank test), respectively.

Leading Context: Bob has a girl friend.	
Emotion-driven Style	
GPT-2	She wants to **take** a trip to Hawaii. She **goes** on vacation. She is at the beach. It is a *pretty* day.
BART	One day, Bob **saw** a *cute* necklace on the sidewalk. Bob decided to **buy** it. After buying it, Bob *loves* it. Bob *likes* the necklace.
Ours	He is really *nervous* about her feeling around her. His girlfriend is very *protective* . Bob gets along *great* with her. Bob has a *wonderful* time with his girlfriend.
Event-driven Style	
GPT-2	She *likes* her hair. She **takes** a few pictures of her friend's hair. He **takes** a picture of her hair and **posts** it. She *likes* it very much and she is *happy* .
BART	He knew he was always going to be *mean* to her. After a while Bob **realized** that he was being *annoying* . He had to **leave** his job and **walk** . Now he has a new girlfriend and a new job.
Ours	He has been **talking** to her all day. She stopped **listening** to him now. One day, she **says** his name and **walked** away. He decided to **break** up with her in another place.

Table 6: Generated stories by different models with different specified styles. *Emotion-related* keywords and **event-related** keywords are highlighted in italic and bold, respectively.

or substantial (0.6 $\leqslant \kappa \leqslant$ 0.8) agreement, and our model outperforms baselines significantly in fluency, coherence, and style consistency.

5 Case Study

Table 6 shows several generated cases. We generate the stories using different models given the same leading context and specified style. For the *emotion-driven* style, our model can generate various emotional keywords (e.g., *"nervous"*, *"protective"*, *"great"*, and *"wonderful"*) and focus more on shaping the characters' personality. For the *event-driven* style, our model can generate fluent stories with a reasonable event sequence. In comparison, baselines tend to confuse the two styles. For example, the stories generated by the baselines for the event-driven style still contain many emotional keywords (e.g., *"likes"*, *"annoying"*). Besides, for the emotion-driven style, the baselines generate fewer and repetitive emotional keywords. Furthermore, the baselines may suffer from more severe repetition (e.g., "take a picture") than our model. And the baselines sometimes mix up or neglect some characters (e.g., GPT-2 and BART only cover one of *"Bob"* and *"his girlfriend"* but neglect the other one for the *emotion-driven* style). In summary, our model can generate more coherent stories with specified styles than the baselines.

6 Conclusion

We present a pilot study on a new task, stylized story generation. We define story *style* with respect to *emotion* and *event*, and propose a generation model which conditions on planned stylistic keywords. Comparative experiments with strong baselines show the promising results of the proposed model. Our work can inspire further research in this new direction.

Acknowledgements

This work was jointly supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096), and the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1. We thank THUNUS NExT Joint-Lab for the support.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Alex Boyd, Raul Puri, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Large scale multi-actor generative dialog modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 66–84.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Angela Fan, Mike Lewis, and Yann N. Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2650–2660. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph M. Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4319–4338. Association for Computational Linguistics.

Jian Guan, Fei Huang, Minlie Huang, Zhihao Zhao, and Xiaoyan Zhu. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Trans. Assoc. Comput. Linguistics*, 8:93–108.

Jian Guan and Minlie Huang. 2020. UNION: an unreferenced metric for evaluating open-ended story generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9157–9166. Association for Computational Linguistics.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James R. Glass. 2019. Quantifying exposure bias for neural language generation. *CoRR*, abs/1905.10617.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8465–8472.

Dongyeop Kang and Eduard Hovy. 2020. Self-supervised text planning for paragraph completion task. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6533–6543.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1499–1508, Melbourne, Australia. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Yuan Li, Chunyuan Li, Yizhe Zhang, Xiujun Li, Guoqing Zheng, Lawrence Carin, and Jianfeng Gao. 2020. Complementary auxiliary classifiers for label-conditional text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8303–8310.

Saif M. Mohammad. 2020. Practical and ethical considerations in the effective use of emotion and sentiment lexicons. *CoRR*, abs/2011.03492.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Lili Mou and Olga Vechtomova. 2020. Stylized text generation: Approaches and applications. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 19–22, Online. Association for Computational Linguistics.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric Xing, and Zhiting Hu. 2019. Target-guided open-domain conversation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5624–5634, Florence, Italy. Association for Computational Linguistics.

Alexey Tikhonov and Ivan P Yamshchikov. 2018. Guess who? multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 787–794. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4306–4315. Association for Computational Linguistics.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: controllable story generation with external knowledge using large-scale language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2831–2845. Association for Computational Linguistics.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Dynamic Connected Networks for Chinese Spelling Check

Baoxin Wang[1,2], **Wanxiang Che**[1], **Dayong Wu**[2], **Shijin Wang**[2,3], **Guoping Hu**[2], **Ting Liu**[1]
[1]Research Center for SCIR, Harbin Institute of Technology, Harbin, China
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[3]iFLYTEK AI Research (Hebei), Langfang, China
`{bxwang2,dywu2,sjwang3,gphu}@iflytek.com`
`{car,tliu}@ir.hit.edu.cn`

Abstract

Chinese spelling check (CSC) is a task to detect and correct spelling errors in Chinese text. Most state-of-the-art works on the CSC task adopt a BERT-based non-autoregressive language model, which relies on the output independence assumption. The inappropriate independence assumption prevents BERT-based models from learning the dependencies among target tokens, resulting in an incoherent problem. To address the above issue, we propose a novel architecture named Dynamic Connected Networks (DCN), which generates the candidate Chinese characters via a Pinyin Enhanced Candidate Generator and then utilizes an attention-based network to model the dependencies between two adjacent Chinese characters. The experimental results show that our proposed method achieves a new state-of-the-art performance on three human-annotated datasets.

1 Introduction

Chinese spelling check (CSC) is an important task which can be utilized in many natural language applications such as optical character recognition (OCR) (Wang et al., 2018; Hong et al., 2019) and essay scoring. Meanwhile, CSC is a challenging task which requires human-level natural language understanding ability (Liu et al., 2010, 2013; Xin et al., 2014). Recently, BERT-based non-autoregressive language models have achieved state-of-the-art performance in the CSC task (Hong et al., 2019; Zhang et al., 2020; Cheng et al., 2020).

These works fine-tune BERT-based models using CSC training data. During the training phase, all the target Chinese characters will be involved as labels. In the inference stage, the models predict the most likely Chinese character from a candidate set at each position. When the most likely character is different from the input character, the input character will be considered as a spelling error and corrected to the most likely character. Based on the powerful generalization ability of BERT (Devlin et al., 2019), these works have achieved better performance than other models.

Wrong: 我忘记告诉你了，我真户秃。
Correct: 我忘记告诉你了，我真糊涂。 Translation: I forgot to tell you. I'm so confused.

Table 1: An example of Chinese spelling errors. Here, "户秃" should be corrected to "糊涂" (confused).

However, these works on the CSC task rely on the incorrect independence assumption, which may lead to an incoherent problem. Concretely, they assume that the predicted tokens are independent of each other, which generally does not hold in natural language (Yang et al., 2019; Gu and Kong, 2020). For the CSC task, one spelling error may have multiple corrections. Ignoring the corrected context may result in a correction conflict. As shown in Table 1, "户秃" may be corrected as "糊涂" (*confused*) or "尴尬" (*embarrassed*). Because of the independence of each token, the non-autoregressive language model may correct it as "尴涂" (*embafused*). This incoherent problem is also called a multi-modality problem in non-autoregressive machine translation (Gu et al., 2018).

To address the above problem, we propose a novel Dynamic Connected Networks (DCN) which can model the dependencies between two adjacent candidate Chinese characters. Specifically, we use the RoBERTa model (Liu et al., 2019; Cui et al., 2019) as our base model, which can also be replaced by other models. Firstly, we utilize RoBERTa with a Pinyin Enhanced Candidate Generator to incorporate phonological information and generate k candidate characters at each position. For each two adjacent candidates, DCN learns a

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2437–2446
August 1–6, 2021.

variable connection score to determine the strength of the dependency between them via a Dynamic Connected Scorer (DCScorer). The DCScorer calculates the connection scores by feeding the contextual representation and the candidate character embeddings of the current and the next position into an attention layer simultaneously. Eventually, the model generates k^n candidate paths, and we utilize the Viterbi algorithm (Rabiner, 1989) to quickly find the one with the highest score as our final correction result.

Conditional random fields (CRF) (Lafferty et al., 2001) can also model the dependencies of output labels, however it is not suitable for language modeling or the CSC task. The dependencies between Chinese characters are more related to the context and far more complicated than the label relations of other tasks such as NER. Thus, the capacity of a fixed transition matrix in CRF is inadequate. Moreover, the number of Chinese characters is usually more than 5K, making the transition matrix too large to learn. In contrast, output candidates (labels) and connection scores of DCN are dynamic and change according to the context. That empowers our model with a strong ability to learn the dependencies.

We conduct experiments on SIGHAN 2013, SIGHAN 2014, and SIGHAN 2015 benchmarks. Experimental results on the three human-annotated datasets demonstrate that the performance of our proposed method is significantly better than the state of the art models.

To summarize, our contributions are as follows:

- We propose a novel end-to-end dynamic connected networks (DCN) which can alleviate the incoherent problem of non-autoregressive language models in the CSC task.
- We propose a simple and effective Pinyin Enhanced Candidate Generator to incorporate phonological information and generate better candidate characters.
- Experimental results show that our proposed method achieves state-of-the-art performance on three human-annotated datasets.

For reproducibility, our code for this paper is available at https://github.com/destwang/DCN.

2 Related Work

Chinese spelling check (CSC) is a challenging task that requires human-level language understanding ability. With the development of deep learning techniques, the CSC task has recently made more progress. CSC is similar to the grammatical error correction (GEC) task (Dahlmeier and Ng, 2012). The difference between them is that CSC only focuses on Chinese spelling errors, while GEC also includes errors that need insertion and deletion.

Most models in the GEC task use an autoregressive Seq2Seq model to correct a sentence. Similarly, Seq2Seq models can also be used in the CSC task. Wang et al. (2019) propose an autoregressive pointer network which generates a Chinese character from the confusion set rather than the entire vocabulary. Although the autoregressive Seq2Seq model has the ability to correct the spelling errors, it is usually slow. The input and output are so similar that it would be "wasteful" to completely regenerate a sequence (Malmi et al., 2019).

Since the input and output have the same number of Chinese characters, and the correct and incorrect Chinese characters correspond to each other, it is more intuitive to use non-autoregressive language models such as BERT to directly correct the Chinese spelling errors. Hong et al. (2019) propose the FASPell model to predict candidate characters based on the BERT model and exploit the phonological and visual similarity information to select candidate characters. Zhang et al. (2020) propose a model named Soft-Masked BERT, which consists of a detection network and a correction network based on BERT. Cheng et al. (2020) propose to incorporate phonological and visual similarity knowledge into BERT via a specialized graph convolutional network. Bao et al. (2020) design a chunk-based framework and extend the traditional confusion sets with semantical candidates to cover different types of errors.

Although these non-autoregressive methods mentioned above have achieved state of the art in the CSC task so far, these methods still suffer from the incoherent problems that exist in non-autoregressive models (Gu et al., 2018; Gu and Kong, 2020). In this paper, we propose a novel model DCN which learns the dependencies between the adjacent Chinese characters and alleviates the incoherent problem.

3 Our approach

3.1 Problem

Given an input text sequence $X = \{x_1, x_2, ..., x_N\}$, the goal of the CSC task is

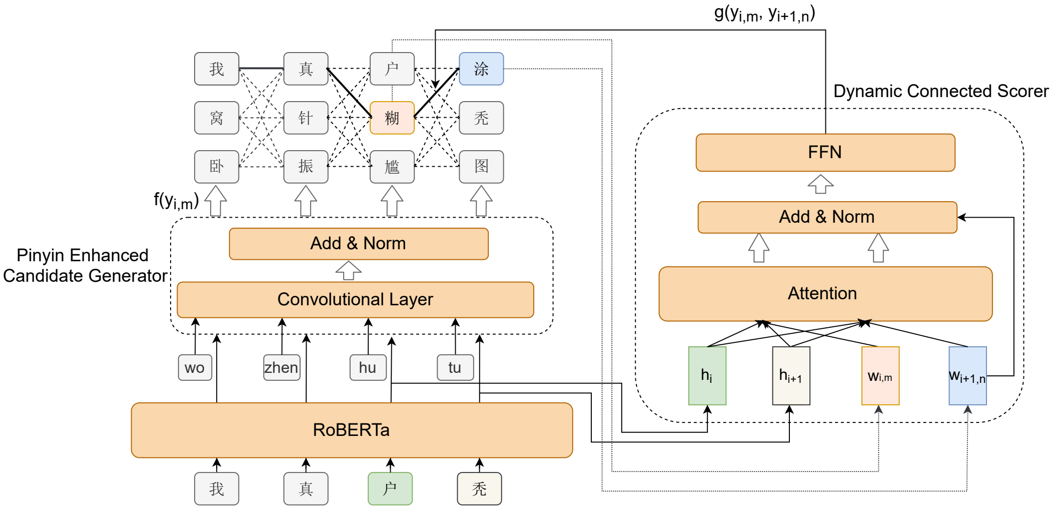

Figure 1: The architecture of DCN. Here, we only illustrate how to calculate one connection score between candidates “糊” and “涂” by the Dynamic Connected Scorer.

to automatically correct the incorrect part of the Chinese sentence and generate a correct target sequence $Y = \{y_1, y_2, ..., y_N\}$. Since the input sentence X and the output sentence Y have the same number of tokens (Chinese characters), pretrained non-autoregressive language models such as BERT are natural to be used in the CSC task. Given that non-autoregressive language models are based on the assumption of output independence, they will mismatch output Chinese characters and lead to the incoherent problem. This problem has also been mentioned in non-autoregressive machine translation (Gu et al., 2018, 2019; Gu and Kong, 2020) and pretrained language model (Yang et al., 2019).

3.2 Dynamic Connected Networks

To solve the above incoherence problem, we propose a novel model named Dynamic Connected Networks (DCN), which can learn the dependencies between output Chinese characters and alleviate the incoherence problem.

The model structure is illustrated in Figure 1. We use the RoBERTa (Liu et al., 2019; Cui et al., 2019) model as our base model. Firstly, RoBERTa with a Pinyin Enhanced Candidate Generator generates a series of candidate characters, and we sample k characters as candidates (the candidate generation method will be discussed in detail in the next subsection). For each two adjacent candidate characters, we learn the connection scores to determine the strength of the dependency between them by a dynamic connected scorer (DCScorer). The final correction score will be calculated by the joint prediction of connection scores and the prediction scores of the candidate generator at each position.

The DCScorer needs to consider the context information, the candidate characters of the current and next position simultaneously. Thus, we use the attention mechanism to learn the current candidate context representation p and next candidate context representation q. The strength of the dependency between two adjacent candidates is usually more related to the RoBERTa hidden representation of the current and next position, so the key and value in the attention mechanism contain only these two hidden representations. The DCScorer is formally defined as follows:

$$
\begin{aligned}
p_{i,m} &= \mathrm{Attention}(Q_{i,m}W^Q, K_iW^K, V_iW^V) \\
q_{i,n} &= \mathrm{Attention}(Q_{i+1,n}W^Q, K_iW^K, V_iW^V) \\
K_i = V_i &= \begin{bmatrix} h_i \\ h_{i+1} \end{bmatrix} \\
Q_{i,m} &= w_{i,m} \\
Q_{i+1,n} &= w_{i+1,n}
\end{aligned}
\tag{1}
$$

where i is the character position, m and n are the indices of candidates of current position and next position respectively. $\mathrm{Attention}$ denotes the attention mechanism, where the Q, K, V denote query, key and value, and W denote the parameters to be learned in the attention layer. h is the hidden representation of the last transformer block, w denotes the candidate token embedding.

We add the candidate token embedding to the candidate context representation. Then we feed the output into layer normalization and get two representations $p'_{i,m}$ and $q'_{i,n}$.

$$\begin{aligned} p'_{i,m} &= \text{LayerNorm}(p_{i,m} + w_{i,m}) \\ q'_{i,n} &= \text{LayerNorm}(q_{i,n} + w_{i+1,n}) \end{aligned} \tag{2}$$

We concatenate the two vectors and feed them into a feed-forward network (FFN) layer used by (Vaswani et al., 2017). Then we use a linear layer to calculate the connection score between the two candidates.

$$\begin{aligned} s &= \text{FFN}(\text{Concat}(p'_{i,m}, q'_{i,n})) \\ g(y_{i,m}, y_{i+1,n}) &= sv \end{aligned} \tag{3}$$

where v is a trainable weight vector and $g(y_{i,m}, y_{i+1,n})$ is the connection score between the mth candidate of ith position and nth candidate of $i + 1$th position.

Since we feed k^2 pairs of candidate combinations into DCScorer, we will generate k^2 scores at each position. Eventually, the model will generate k^n candidate paths, and the score of each path is calculated using the following equations:

$$\text{S}(X, Y) = \sum_{i=1}^{N} f(y_{i,m}) + \sum_{i=1}^{N-1} g(y_{i,m}, y_{i+1,n}) \tag{4}$$

where y is the candidate character, $f(y_{i,m})$ is the prediction score of Pinyin Enhanced Candidate Generator for mth candidate of ith position.

3.3 Candidate Generation

We generate the candidate Chinese characters via a Pinyin Enhanced Candidate Generator based on RoBERTa.

Pinyin Enhanced Candidate Generator According to statistics, more than 80% spelling errors are related to phonological similarity (Liu et al., 2010). Since phonological errors account for a large proportion of Chinese character errors, a suitable method of introducing phonological information would be of great help in generating the candidates and correcting spelling errors.

The conversion from a single Chinese Pinyin to the Chinese character has a large ambiguity. It is difficult to convert properly because one Pinyin usually corresponds to many Chinese characters. However, when there are multiple consecutive Pinyin, we will have more confidence to convert Pinyin into correct Chinese characters. For example, the Pinyin of Chinese characters “户” and “糊” is “hu”, and the Pinyin of “秃” and “涂” is “tu”. When “hu” and “tu” are together, it will have a high probability of being converted to “糊涂” which means “confused” in Chinese. This is also a basic assumption used in Chinese Pinyin input methods.

Based on this, we propose a Pinyin Enhanced Candidate Generator, which can effectively reduce the ambiguity and generate better Chinese characters. The architecture is shown in Figure 1. Concretely, we adopt a convolutional layer to encode consecutive Pinyin and add the output of convolutional layer, hidden representation of RoBERTa and character embedding together. Then we feed the sum to layer normalization and get the prediction score $f(y_{i,m})$ via a linear layer. The equations are as follows:

$$\begin{aligned} c_i &= \text{Conv}(p''_{i-1}, p''_i, p''_{i+1}) \\ o_i &= \text{LayerNorm}(c_i + w_i + h_i) \\ f(y_{i,m}) &= o_i v'_m \end{aligned} \tag{5}$$

where p'' is the Pinyin embedding, w_i is the Chinese character embedding, h_i is the last hidden representation of RoBERTa, v'_m is the trainable weight vector for mth candidate.

There are various ways to represent Pinyin, and we find that simply representing each Pinyin without tone as a separate embedding can achieve good performance. We also try to encode Pinyin by Multi-Layer Perceptron (MLP) and GRU (Chung et al., 2014) encoder, which treat each letter of Pinyin as an embedding vector. Since they cannot achieve better results, we simply represent each Pinyin as a separate embedding in our following experiments.

Candidate Sampling Method Given the large number of usual Chinese characters, we sample the candidate characters for learning. We try several sampling methods and find that selecting the characters with the top-k prediction scores from vocabulary performs best. This also shows that the more difficult candidates can be used as negative training examples to effectively improve the discriminatory ability of the model. Therefore, all the main experimental results are based on the top-k sampling.

3.4 Learning

Loss function The probability for the sequence Y can be approximated by the following equation

$$p(Y|X) = \frac{e^{\text{S}(X,Y)}}{\sum_{Y'_i} e^{\text{S}(X,Y'_i)}} \tag{6}$$

Training Sets	# Line	Avg. Length	# Erroneous Sent.
Wikipedia	7,756,725	47.0	-
(Wang et al., 2018)	271,329	44.4	271,329
SIGHAN 2013	700	49.2	350
SIGHAN 2014	3,435	49.7	3,432
SIGHAN 2015	2,339	30.0	2,339
Test Sets	**# Line**	**Avg. Length**	**# Erroneous Sent.**
SIGHAN 2013	1,000	74.1	996
SIGHAN 2014	1,062	50.1	529
SIGHAN 2015	1,100	30.5	550

Table 2: Statistics of datasets.

where Y_i' are the path generated by the candidate characters.

The loss function is the maximum likelihood of the probability distribution denoted as

$$Loss = \begin{cases} -\log(p(Y|X)) & \mathrm{S}(X,Y) < \mathrm{S}_{max}(X,Y') \\ 0 & \mathrm{S}(X,Y) \geq \mathrm{S}_{max}(X,Y') \end{cases} \quad (7)$$

The loss function is similar to the one used by LSTM-CRF (Huang et al., 2015). It learns only the sampled negative candidate characters and the dependencies between them, which will unduly degrade the ranking of potential candidates. This possibly makes more similar candidates have a lower ranking. In order to avoid the above problem, we make a restriction on the loss function by setting its loss to 0 when the gold score is higher than or equal to the max score of all the candidate paths.

Pretraining The dependencies between Chinese characters can be more sufficiently learned via a large scale training corpus. In this paper, we pretrain our proposed model using Chinese Wikipedia data shown in Table 2. We randomly replace 15% of the characters, including 70% MASK, 15% characters from the confusion set, and 15% random characters. We exploit the confusion set released from SIGHAN 2013 (Wu et al., 2013) which consist of pronunciation similarity and shape similarity characters. Based on the RoBERTa model, we freeze the main parameters and only fine-tune the Pinyin Enhanced Candidate Generator and the Dynamic Connected Scorer.

3.5 Predicting

In the predicting stage, the top-k candidate characters from vocabulary are generated by the Pinyin Enhanced Candidate Generator. Eventually, there are k^n paths. In order to quickly select the path with the highest score, we use the Viterbi algorithm (Rabiner, 1989) based on dynamic programming to decode the output sequence.

4 Experiments

4.1 Experimental Setup

Datasets We use the large automatically generated corpus (Wang et al., 2018)[1] as our training data. In addition, the training sets of SIGHAN 2013, SIGHAN 2014, and SIGHAN 2015 are also included. For the pre-training method, we use the Chinese Wikipedia texts which have been converted to simplified Chinese characters.

We evaluate our proposed model on the test sets from SIGHAN 2013, SIGHAN 2014, and SIGHAN 2015 benchmarks. Similar to the previous works, we convert the traditional characters to simplified characters by OpenCC[2].

In order to evaluate our model more reasonably, we take 500 sentences from the SIGHAN training sets and the corresponding corrected results of these 500 sentences together as the validation set. The statistic information of all the datasets is listed in Table 2.

Evaluation Metrics To compare with the state-of-the-art models, We use the widely adopted sentence-level precision, recall, and F1-score as our evaluation method, which has been used by Hong et al. (2019)[3] and Cheng et al. (2020).

Baseline Models We compare our model with several state-of-the-art models.

- FASPell (Hong et al., 2019): This model uses the phonological and visual similarity information to select candidate characters.
- Soft-Masked BERT (Zhang et al., 2020): This method combines a detection network and a correction network based on BERT.
- SpellGCN (Cheng et al., 2020): This model incorporates phonological and visual similarity knowledge into BERT via a specialized graph convolutional network.
- Chunk-based method (Bao et al., 2020): This method utilizes a chunk-based framework and extends the traditional confusion sets with semantical candidates to cover different types of errors.

Model Hyperparameters We use RoBERTa-wwm (Cui et al., 2019) as our base model in this

---

[1] https://github.com/wdimmy/Automatic-Corpus-Generation

[2] https://github.com/BYVoid/

[3] https://github.com/iqiyi/FASPell

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
CSC13	FASPell (Hong et al., 2019)	76.2	63.2	69.1	73.1	60.5	66.2
	BERT (Cheng et al., 2020)	79.0	72.8	75.8	77.7	71.6	74.6
	SpellGCN (Cheng et al., 2020)	80.1	74.4	77.2	78.3	72.7	75.4
	SpellGCN*	85.2	77.7	81.2	83.4	76.1	79.6
	RoBERTa (Ours)	85.4	77.7	81.3	83.9	76.4	79.9
	RoBERTa-DCN (Ours)	86.2	78.4	82.1	84.6	76.9	80.5
	RoBERTa-Pretrain-DCN (Ours)	**86.8**	**79.6**	**83.0**	**84.7**	**77.7**	**81.0**
CSC14	FASPell (Hong et al., 2019)	61.0	53.5	57.0	59.4	52.0	55.4
	BERT (Cheng et al., 2020)	65.6	68.1	66.8	63.1	65.5	64.3
	SpellGCN (Cheng et al., 2020)	65.1	69.5	67.2	63.1	67.2	65.3
	RoBERTa (Ours)	64.2	68.4	66.2	62.7	66.7	64.6
	RoBERTa-DCN (Ours)	**67.6**	68.6	68.0	64.9	65.9	65.4
	RoBERTa-Pretrain-DCN (Ours)	67.4	**70.4**	**68.9**	**65.8**	**68.7**	**67.2**
CSC15	FASPell (Hong et al., 2019)	67.6	60.0	63.5	66.6	59.1	62.6
	Soft-Masked BERT (Zhang et al., 2020)	73.7	73.2	73.5	66.7	66.2	66.4
	BERT (Cheng et al., 2020)	73.7	78.2	75.9	70.9	75.2	73.0
	SpellGCN (Cheng et al., 2020)	74.8	80.7	77.7	72.1	77.7	*75.9*(74.8)
	RoBERTa (Ours)	74.7	77.3	76.0	72.1	74.5	73.3
	RoBERTa-DCN (Ours)	76.6	79.8	78.2	74.2	77.3	75.7
	RoBERTa-Pretrain-DCN (Ours)	**77.1**	**80.9**	**79.0**	**74.5**	**78.2**	**76.3**

Table 3: Experimental results of sentence-level precision, recall, and F1-score (%). D, C denote the detection and correction respectively. Since "的", "地", "得" are rarely distinguished on SIGHAN 2013, we remove all the related correction results. To compare more fairly with SpellGCN, we rerun the released code of Cheng et al. (2020) and remove all the related correction results. The results are reported with SpellGCN*. The reported result of SpellGCN on SIGHAN 2015 is not correct, where the precision, recall and F-score don't match. If the precision and recall are correct, F-score should be 74.8.

paper. We utilize AdamW (Loshchilov and Hutter, 2019) optimizer with learning rate of 5e-5. The training batch size is set to 32, and we train 12 epochs for all the experiments. To better learn the dependencies between characters, we learn the DCN model with MASK token for the first 2 epochs the same with the pretraining method. The number of candidates k for training is set to 5 and the number for predicting is set to 8. The convolution window size of the Pinyin Enhanced Candidate Generator is set to 3. The dimensions of all the hidden representations are 768. We search learning rate from $\{2e\text{-}5, 3e\text{-}5, 5e\text{-}5\}$ and select the best model on the validation set.

4.2 Experimental Results

The experimental results are shown in Table 3. Our proposed RoBERTa-DCN model has the best detection and correction performance on the three SIGHAN test sets. Both FASPell and SpellGCN models use sophisticated techniques to incorporate the phonological and visual information and achieve a relatively good performance. Our DCN model is more focused on the incoherence problem and modeling the dependencies of the output tokens. Our proposed model exceeds FASPell and SpellGCN by simply using a Pinyin Enhanced Candidate Generator to model the phonological information, which also illustrates the effectiveness of DCN.

When we pre-train DCN using wiki data, the model gets further improvement in the effectiveness. This indicates that modeling the dependencies between output Chinese characters is important. DCN may achieve better performance if more data are used to learn the dependencies.

Soft-Masked BERT uses detection network and correction network simultaneously. In contrast, our DCN model predicts the target sequence directly, and the different tokens between the input sequence and the target sequence are regarded as the detection results. As shown in the experimental results, compared to Soft-Masked BERT, our method improves 5.5% and 9.9% on detection and correction respectively.

To compare with some other state-of-the-art works, we also evaluate our proposed model using the official evaluation toolkit[4] of SIGHAN 2015 in Table 4. The Chunk-based method, which uses a series of methods to construct the candidate set,

---

[4] http://nlp.ee.ncu.edu.tw/resource/csc.html

Model	Detection-level				Correction-level			
	D-Acc	D-P	D-R	D-F	C-Acc	C-P	C-R	C-F
Chunk-based method (Bao et al., 2020)	76.8	88.1	62.0	72.8	74.6	87.3	57.6	69.4
BERT (Cheng et al., 2020)	83.0	85.9	78.9	82.3	81.5	85.5	75.8	80.5
SpellGCN (Cheng et al., 2020)	83.7	85.9	80.6	83.1	82.2	85.4	77.6	81.3
RoBERTa (Ours)	83.2	86.6	78.6	82.4	81.8	86.2	75.8	80.7
RoBERTa-DCN (Ours)	84.2	86.4	**81.1**	83.7	82.8	86.0	**78.4**	82.0
RoBERTa-Pretrain-DCN (Ours)	**84.6**	**88.0**	80.2	**83.9**	**83.2**	**87.6**	77.3	**82.1**

Table 4: The performance evaluated by official tools on SIGHAN 2015.

Sampling Method	D-F	C-F
Top-k of vocabulary	**89.7**	**88.7**
Multinomial distribution sampling	88.1	87.6
Random sampling	12.2	7.3
Top-k of confusion set	35.8	34.7

Table 5: Effect of the candidate generation methods.

Model	D-P	D-R	D-F	C-P	C-R	C-F
RoBERTa-DCN	**89.8**	**89.6**	**89.7**	**88.8**	**88.6**	**88.7**
- PECGenerator	87.7	88.4	88.1	86.7	87.4	87.1
- DCScorer	87.4	88.4	87.9	86.8	87.8	87.3
- weighted loss	87.6	89.0	88.3	86.8	88.2	87.5
RoBERTa	86.1	89.2	87.6	85.5	88.6	87.0

Table 6: Ablation Study of DCN on validation set. PECGenerator is the Pinyin Enhanced Candidate Generator. Weighted loss refers to the condition of loss. When we remove the PECGenerator, the RoBERTa generates the candidates by predicting the candidate characters. When the DCScorer is removed, the model selects the top-1 predicted result as the correct character.

achieves good performance for precision. However, the recall of this method is relatively low, and the F-score of our method significantly outperforms the chunk-based method by more than 10%. Similarly, our model also achieves a better result than SpellGCN.

4.3 Effect of Candidate Generation

The performance of DCN varies with the candidate generation strategy and the number of sampled candidate characters. We compare the effects of four sampling methods for training, which are sampling top-k candidates from vocabulary, sampling top-k candidates from the confusion set, random sampling from the vocabulary and sampling from a multinomial distribution. For the multinomial distribution sampling, the probabilities are obtained from the Softmax output of the Pinyin Enhanced Candidate Generator. All the subsequent experiments are conducted on the validation set. The experimental results are shown in Table 5.

From Table 5, we can see that the top-k of vocabulary method has the best performance. The multinomial distribution sampling also has a good performance, while the random sampling and top-k of confusion set cannot achieve good performance. This means that sampling some difficult candidates is more beneficial to the model training to improve the model discriminative ability.

We also conduct experiments with the effect of the number of candidates. Figure 2(a) shows the change curve of the effect when the number of candidates for training increase. The effect gradually gets better as the number of candidates increases at the beginning, and the effect no longer has a significant improvement after the number of candidates for prediction exceeds 5. Figure 2(b) shows the performance when we fix the number of training candidates as 5 and increase the number of prediction candidates. The performance keeps improving as the number of prediction candidates increases.

4.4 Ablation Study

We conduct a series of experiments to determine which component in the DCN model plays a more important role. Table 6 shows the results of our experiments. When we remove the Pinyin Enhanced Candidate Generator, both the detection and correction F-scores decrease about 1.5%. This demonstrates that the phonological information plays an important role in candidates generation methods. When we remove the dynamic connected scorer, the detection F-score decreases nearly 2%, which indicates that the dependencies between Chinese characters are important for the CSC task. Similarly, the weighted loss also help our models improve the performance.

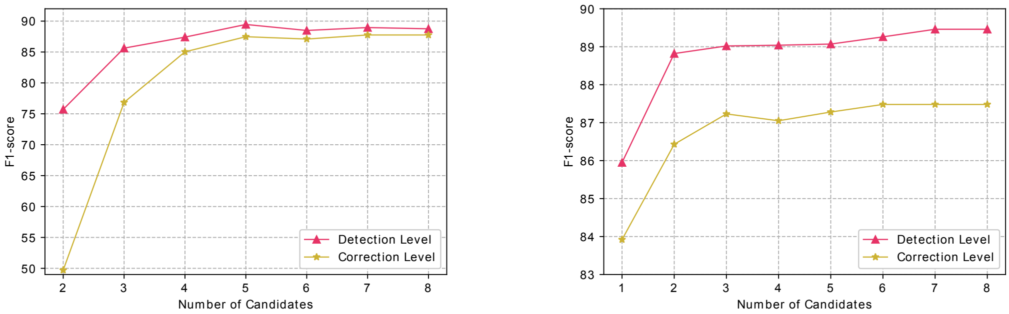

(a) Training stage.
(b) Predicting stage.

Figure 2: The effect of the number of candidates.

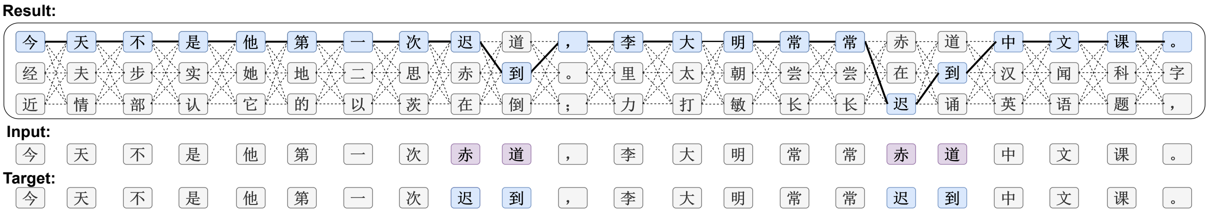

Figure 3: An example of Viterbi decoding. “赤道” in this sentence should be corrected to “迟到”. Translation of this example: *Today is not the first time he is late. Daming Li is often late to Chinese class.*

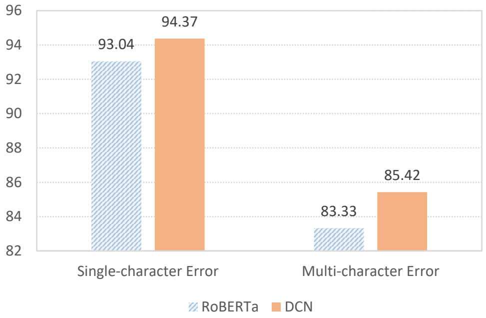

Figure 4: The comparison of detection F-score between RoBERTa and DCN on single-character error and multi-character error validation sets.

4.5 Case Study and Analysis

We find that the DCN model performs better than the vanilla RoBERTa model on consecutive errors. Figure 3 shows an example of consecutive errors. The vanilla RoBERTa model cannot detect it well and can only partially correct it because consecutive errors will influence each other. In contrast, DCN can correct it completely. The best path for this example is shown in this figure. The correct Chinese characters “迟到” did not rank first, but the path including “迟到” have the highest score because they are more fluent than other candidate combinations. This example also shows that our model can alleviate the incoherent problem.

Figure 4 shows the detection F-score of RoBERTa and DCN on single-character and multi-character error sets. DCN performs better than RoBERTa in both single-character and multi-character level cases. The effect of DCN is more obvious in the multi-character cases, which also shows that DCN has some advantages for multi-character type errors. At the same time, the performance of single-character error cases is much better than multi-character error cases, which indicates that DCN still has much room to improve for multi-character errors.

By comparing the results of RoBERTa and RoBERTa with the candidate generator. We find that 96.7% of the correct characters are in the top-5 candidates of RoBERTa. In contrast, 98.6% of the correct characters are in the top-5 candidates of RoBERTa with candidate generator. This result illustrates that the Pinyin Enhanced Candidate Generator can generate better candidates.

5 Conclusion

In this paper, we propose a novel model named DCN to solve the incoherent problem in the CSC task. To better incorporate the phonological information, we propose a simple and effective Pinyin Enhanced Candidate Generator. The experimental results show that our proposed model has achieved the state-of-the-art performance on three datasets. DCN may also be utilized on other tasks such as non-autoregressive machine translation. As for future work, how to make better use of phonological and visual information still needs to be discussed.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2018YFC0832302). I would like to thank the anonymous reviewers and Xiaoxue Wang for their insightful comments and suggestions.

References

Zuyi Bao, Chen Li, and Rui Wang. 2020. Chunk-based Chinese spelling check with global optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2031–2040, Online. Association for Computational Linguistics.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. SpellGCN: Incorporating phonological and visual similarities into language models for Chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 871–881, Online. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Jiatao Gu and Xiang Kong. 2020. Fully non-autoregressive neural machine translation: Tricks of the trade. *arXiv preprint arXiv:2012.15833*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. FASPell: A fast, adaptable, simple, powerful Chinese spell checker based on DAE-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169, Hong Kong, China. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. 2010. Visually and phonologically similar characters in incorrect simplified Chinese words. In *Coling 2010: Posters*, pages 739–747, Beijing, China. Coling 2010 Organizing Committee.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid Chinese spelling correction using language model and statistical machine translation with reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan. Asian Federation of Natural Language Processing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. *arXiv preprint arXiv:1909.01187*.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for Chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527, Brussels, Belgium. Association for Computational Linguistics.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for Chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785, Florence, Italy. Association for Computational Linguistics.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42, Nagoya, Japan. Asian Federation of Natural Language Processing.

Yang Xin, Hai Zhao, Yuzhu Wang, and Zhongye Jia. 2014. An improved graph model for Chinese spell checking. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 157–166, Wuhan, China. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 882–890, Online. Association for Computational Linguistics.

A Multi-Level Attention Model for Evidence-Based Fact Checking

Canasai Kruengkrai **Junichi Yamagishi** **Xin Wang**
National Institute of Informatics, Japan
`{canasai,jyamagishi,wangxin}@nii.ac.jp`

Abstract

Evidence-based fact checking aims to verify the truthfulness of a claim against evidence extracted from textual sources. Learning a representation that effectively captures relations between a claim and evidence can be challenging. Recent state-of-the-art approaches have developed increasingly sophisticated models based on graph structures. We present a simple model that can be trained on sequence structures. Our model enables inter-sentence attentions at different levels and can benefit from joint training. Results on a large-scale dataset for Fact Extraction and VERification (FEVER) show that our model outperforms the graph-based approaches and yields 1.09% and 1.42% improvements in label accuracy and FEVER score, respectively, over the best published model.[1]

1 Introduction

False or misleading claims spread through online media faster and wider than the truth (Vosoughi et al., 2018). False claims can occur in many different forms, e.g., fake news, rumors, hoaxes, propaganda, etc. Identifying false claims that are likely to cause harm in the real world is important. Generally, claims can be categorized into two types: verifiable and unverifiable. Verifiable claims can be confirmed to be true or false as guided by evidence from credible sources, while unverifiable claims cannot be confirmed due to insufficient information.

Verifying the truthfulness of a claim with respect to evidence can be regarded as a special case of recognizing textual entailment (RTE) (Dagan et al., 2006) or natural language inference (NLI) (Bowman et al., 2015), where the premise (evidence) is not given. Thus, the task of claim verification is to first retrieve documents relevant to a given claim from textual sources, then select sentences likely to contain evidence, and finally assign a veracity relation label to support or refute the claim. For example, the false claim "*Rabies is a foodborne illness.*" can be refuted by the evidence "*Rabies is spread when an infected animal scratches or bites another animal or human.*" extracted from the Wikipedia article "*Rabies*". Figure 1 shows other examples that require multiple evidence sentences to support or refute claims. All of these claims are taken from a benchmark dataset for Fact Extraction and VERification (FEVER) (Thorne et al., 2018). A key challenge is to obtain a representation for claim and evidence sentences that can effectively capture relations among them.

ID: 8143
Claim: Moscovium is a transactinide element.
Label: SUPPORTED
Evidence: [*Moscovium*]
Moscovium is a superheavy synthetic element with symbol Mc and atomic number 115.^{0}
In the periodic table, it is a p-block transactinide element.^{7}
[*Transactinide_element*]
In chemistry, transactinide elements (also, transactinides, or super-heavy elements) are the chemical elements with atomic numbers from 104 to 120.^{0}

ID: 201459
Claim: A dynamic web page does not involve computer programming.
Label: REFUTED
Evidence: [*Web_page*]
A static web page is delivered exactly as stored, as web content in the web server's file system, while a dynamic web page is generated by a web application that is driven by server-side software or client-side scripting.^{14}
[*Dynamic_web_page*]
A dynamic web page is then reloaded by the user or by a computer program to change some variable content.^{9}

Figure 1: Examples from the FEVER dev set, where true evidence sentences are present in the selected sentences, and veracity relation labels are correctly predicted by our proposed model. Wikipedia article titles are in [*italics*]. Superscripts indicate the positions of the sentences in each article.

---

[1] The code and model checkpoints are available at: https://github.com/nii-yamagishilab/mla.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2447–2460
August 1–6, 2021.

Recent state-of-the-art approaches have attempted to meet this challenge by applying graph-based neural networks (Kipf and Welling, 2017; Velickovic et al., 2018). For example, Zhou et al. (2019) regard an evidence sentence as a graph node, while Liu et al. (2020) use a more fine-grained node representation based on token-level attention. Zhong et al. (2020) use semantic role labeling (SRL) to build a graph structure, where a node can be a word or a phrase depending on the SRL's outputs.

In this paper, we argue that such sophisticated graph-based approaches may be unnecessary for the claim verification task. We propose a simple model that can be trained on a sequence structure. We also observe mismatches between training and testing. At test time, the model predicts the veracity of a claim based on retrieved documents and selected sentences, which contain prediction errors, while at training time, only ground-truth documents and true evidence sentences are available. We empirically show that our model, trained with a method that helps reduce training-test discrepancies, outperforms the graph-based approaches.

In addition, we observe that most of the previous work neglects sentence-selection labels when training veracity prediction models. Thus, we propose leveraging those labels to further improve veracity relation prediction through joint training. Unlike previous work that jointly trains two models (Yin and Roth, 2018; Li et al., 2020; Hidey et al., 2020; Nie et al., 2020), our approach is still a pipeline process where only a subset of potential candidate sentences produced by *any* sentence selector can be used for joint training. This approach makes it possible to explore different sentence-selection models trained with different methods.

Our contributions are as follows. We develop a method for mitigating training-test discrepancies by using a mixture of predicted and true examples for training. We propose a multi-level attention (MLA) model that enables token- and sentence-level self-attentions and that benefits from joint training. Experiments on the FEVER dataset show that MLA outperforms all the published models, despite its simplicity.

2 Background and related work

2.1 Problem formulation

The input of our task is a claim and a collection of Wikipedia articles $\mathcal{D}$. The goal is to extract a set of evidence sentences from $\mathcal{D}$ and assign a veracity relation label $y \in \mathcal{Y} = \{\mathrm{S}, \mathrm{R}, \mathrm{N}\}$ to a claim with respect to the evidence set, where S = SUPPORTED, R = REFUTED, and N = NOTENOUGHINFO. The definition of our labels is identical to that of the FEVER Challenge (Thorne et al., 2018).

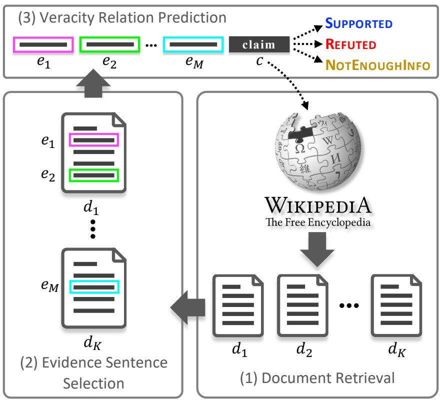

Figure 2: Process of evidence-based fact checking: retrieving documents relevant to a given claim from Wikipedia, selecting sentences likely to contain evidence, and predicting a veracity relation label based on selected sentences.

2.2 Overview of evidence-based fact checking

The process of evidence-based fact checking, shown in Figure 2, commonly involves the following three subtasks.

Document retrieval

Given a claim, the task is to retrieve the top K relevant documents from $\mathcal{D}$. Thorne et al. (2018) suggest using the document retriever from DrQA (Chen et al., 2017a), which ranks documents on the basis of the term frequency-inverse document frequency (TF-IDF) model with unigram-bigram hashing. Hanselowski et al. (2018) use a hybrid approach that combines search results from the MediaWiki API and the results of using exact matching on all Wikipedia article titles. In this paper, our main focus is to improve evidence sentence selection and veracity relation prediction, so we directly use the document retrieval results from Hanselowski et al. (2018). This allows us to fairly compare our model with a series of previous methods (Soleimani et al., 2019; Zhou et al., 2019; Liu et al., 2020; Ye et al., 2020) that also rely on Hanselowski et al. (2018)'s results.

Evidence sentence selection

The task is to select the top M sentences from the retrieved documents. Thorne et al. (2018) again use the TF-IDF model to rank sentences similar to a given claim. Nie et al. (2019a); Hanselowski et al. (2018) use the enhanced sequential inference model (ESIM) (Chen et al., 2017b) to encode and align a claim-sentence pair. Liu et al. (2020); Hanselowski et al. (2018) use a pairwise hinge loss to rank sentences, while Soleimani et al. (2019) explore both pointwise and pairwise losses and suggest selecting difficult negative examples for training. The pairwise hinge loss aims to maximize the margin between the scores of positive and negative examples, while the pointwise loss is a vanilla cross-entropy loss. Our model uses a pointwise loss trained with examples sampled from both ground-truth and predicted documents.

Veracity relation prediction

Given a claim and a set of M selected sentences, the task is to predict their veracity relation label y. Previous work on the FEVER Challenge modified existing RTE/NLI models to deal with multiple sentences (Nie et al., 2019a; Yoneda et al., 2018; Hanselowski et al., 2018; Thorne et al., 2018), used heuristic rules to combine predictions from individual claim-sentence pairs (Malon, 2018), or concatenated all sentences (Stammbach and Neumann, 2019). A line of recent work has applied graph-based neural networks (Zhou et al., 2019; Liu et al., 2020; Zhong et al., 2020). Our model is simply trained on linear sequences by using self- and cross-attention to learn inter-sentence interactions.

2.3 Pre-trained language models

A key to the success of state-of-the-art approaches is the use of pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Lan et al., 2020). Here, we use BERT (Devlin et al., 2019), a bidirectional transformer (Vaswani et al., 2017), to obtain the vector representation of a token sequence. Each BERT layer transforms an input token sequence (one or two sentences) by using self-attention. The first hidden state vector of the final layer represents a special classification token (CLS), which can be used in downstream tasks. We denote the above process by $\mathrm{BERT}_{\mathrm{CLS}}(\cdot) \in \mathbb{R}^{d_h}$, where d_h means the dimensionality of BERT hidden state vectors.

3 Proposed method

In this section, we describe our contributions, including (1) our method for training the sentence-selection model and (2) our veracity prediction model that can be extended with inter-sentence attentions and joint training.

3.1 Learning to select sentences from mixed ground-truth and retrieved documents

Our goal is to select a subset of evidence sentences from all candidate sentences in the retrieved documents. We consider this task to be a binary classification problem that takes as input a pair of a claim c and a candidate sentence e_j and maps it to the output $z \in \mathcal{Z} = \{-1, +1\}$, where $+1$ indicates an evidence sentence and -1 otherwise. We train our sentence-selection model by minimizing the standard cross-entropy loss for each example:

$$\mathcal{L}_{e_j}(\phi) = -\sum_{z \in \mathcal{Z}} \mathbb{1}\{\hat{z} = z\} \mathrm{log} p_{\phi}(\hat{z}|c, e_j), \quad (1)$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, and p_ϕ is the probability distribution of the two classes generated by our model. We compute p_ϕ by applying a multilayer perceptron (MLP) to the vector representation of e_j followed by a softmax function:

$$p_{\phi}(\hat{z}|c, e_j) = \mathrm{softmax}(\mathrm{MLP}(\mathbf{e}_j)), \quad (2)$$
$$\mathbf{e}_j = \mathrm{BERT}_{\mathrm{CLS}}(c, e_j).$$

The MLP contains two affine transformations that map $\mathbf{e}_j$ to the output space. Feeding the pair of c and e_j to BERT allows us to obtain hidden state vectors that capture interactions between c and e_j at the token level. This is due to the self-attention mechanism inside the BERT layers. We expect the final hidden state vector of the CLS token (i.e., $\mathbf{e}_j$) to encode useful information from e_j with respect to c. The parameters ϕ include those in MLP and BERT.

Training our model seems straightforward. However, two technical issues exist. First, each document typically contains one or two (or no) evidence sentences. Training with a few positive examples (i.e., evidence sentences) against all negative examples (i.e., non-evidence sentences) may be neither efficient nor effective. Soleimani et al. (2019) use hard negative mining (HNM) to repeatedly select a subset of difficult negative examples for training their sentence selector. Second, at test time, the model must examine all candidate sentences in the relevant documents returned by the document retriever. However, at training time, the model has no chance to learn the characteristics of non-evidence sentences in the irrelevant but highly ranked documents if only the ground-truth documents are used.

We propose to mitigate the aforementioned issues by using both the ground-truth and retrieved documents to create negative examples for a claim. First, we randomly choose r non-evidence sentences from each ground-truth document, where r is twice the number of true evidence sentences. Then, we sample two other non-evidence sentences from each retrieved document. For positive examples, we use the true evidence sentences in the ground-truth documents. Our scheme is more efficient than HNM of Soleimani et al. (2019). At test time, we select the top M sentences according to the probabilities assigned to the positive class.

3.2 Multi-level attention and joint training for veracity relation prediction

Training-test discrepancies also occur in veracity relation prediction. At test time, the model predicts the veracity of a claim on the basis of the predicted evidence sentences. At training time, only true evidence sentences are available for SUPPORTED and REFUTED, but not for NOTENOUGHINFO. In other words, we have no example sentences that more or less relate to a claim but may not be sufficient to support or refute the claim to train the model. Thorne et al. (2018) simulate training examples for NOTENOUGHINFO by sampling a sentence from the highest-ranked page returned by the document retriever.

We propose to reduce this discrepancy by using a mixture of true and predicted evidence sentences for training. First, we pair each claim with a list of the top M predicted sentences obtained through a sentence selector. At training time, we then prepend the true evidence sentences (if available) to the list and keep the number of all the sentences at most M.[2] At test time, we use the top M predicted sentences without requiring a predefined threshold to filter them. This is in contrast to previous work (Zhou et al., 2019; Nie et al., 2019b; Wadden et al., 2020) and helps reduce engineering effort. Our example sentences for NOTENOUGHINFO are from the sentence selector, not from the document retriever as in (Thorne et al., 2018). We expect our training examples to be similar to what our model may encounter at test time.

On the basis of the above scheme, each example is a pair of a claim c and a set of evidence sentences $\{e_j\}_{j=1}^{M}$. Our goal is to predict the veracity relation label $y \in \mathcal{Y} = \{\mathrm{S}, \mathrm{R}, \mathrm{N}\}$. We train our veracity prediction model by minimizing the class-weighted cross-entropy loss for each example:

$$\mathcal{L}_p(\boldsymbol{\theta}) = -\sum_{y \in \mathcal{Y}} \beta_y \mathbb{1}\{\hat{y} = y\} \log p_{\boldsymbol{\theta}}(\hat{y} | c, \{e_j\}_{j=1}^{M}), \tag{3}$$

where β_y is the class weight for dealing with the class imbalance problem (detailed in Section 4.2). Similar to Eq. (2), we compute the probability distribution $p_{\boldsymbol{\theta}}$ of veracity relation labels as:

$$p_{\boldsymbol{\theta}}(\hat{y} | c, \{e_j\}_{j=1}^{M}) = \mathrm{softmax}(\mathrm{MLP}(\mathbf{a})). \tag{4}$$

Here, $\mathbf{a}$ is the vector representation of aggregated evidence about a claim that is obtained through the multi-head attention (MHA) function:

$$\mathbf{a} = \mathrm{MHA}(\mathbf{Q} = \mathbf{c}, \mathbf{K} = \mathbf{E}, \mathbf{V} = \mathbf{E}), \tag{5}$$

where $\mathbf{c}$ is the claim vector, $\mathbf{E}$ is the set of evidence vectors $\{\mathbf{e}_j\}_{j=1}^{M}$, and $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ denote the query, keys, and values, respectively. All the claim and evidence vectors are derived from BERT:

$$\mathbf{c} = \mathrm{BERT}_{\mathrm{CLS}}(c),$$
$$\mathbf{e}_j = \mathrm{BERT}_{\mathrm{CLS}}(c, e_j).$$

The parameters $\boldsymbol{\theta}$ are those in MLP, MHA, and BERT.

Now let us explain the MHA function, because we use and/or modify it in other components. The MHA function is based on the scaled dot-production attention (Vaswani et al., 2017):

$$\mathrm{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\gamma}\Big)\mathbf{V}, \tag{6}$$

where $\gamma = \sqrt{d_h/n}$ is the scaling factor. The above function is the weighted sum of the values (i.e., the evidence vectors), where the weight assigned to each value is the result of applying a softmax function to the scaled dot products between the query (i.e., the claim vector) and the keys (i.e., the evidence vectors).

The MHA function contains a number of parallel heads (i.e., attention layers). We expect each head to capture different aspects of the input. We achieve this by linearly projecting $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ to new representations and feeding them to the scaled dot-product attention. Specifically, the MHA function is given by:

$$\mathrm{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathrm{head}_1, \ldots, \mathrm{head}_n]\mathbf{W}^{O}, \tag{7}$$

$$\mathrm{head}_i = \mathrm{attn}(\mathbf{Q}\mathbf{W}_i^{Q}, \mathbf{K}\mathbf{W}_i^{K}, \mathbf{V}\mathbf{W}_i^{V}), \tag{8}$$

---

[2] True evidence sentences may already exist in the list because the sentence selector can correctly identify them.

where n is the number of parallel heads, and $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d_h \times \frac{d_h}{n}}$; $\mathbf{W}^O \in \mathbb{R}^{d_h \times d_h}$ are the weight matrices of the linear projections.

Inter-sentence attentions

Although Eq. (5) helps aggregate the evidence from multiple selected sentences, our model still has no mechanism to learn interactions among these sentences. Unlike previous work that uses graph-based attention (Zhou et al., 2019; Liu et al., 2020; Zhong et al., 2020), our main tool is just the described MHA function.

Let $\mathbf{H}_j = [\mathbf{h}_{j,1}, ..., \mathbf{h}_{j,L}]$ be a sequence of the hidden state vectors of e_j generated by BERT, where L is the maximum sequence length. Let $\mathbf{H}$ be the concatenation of all the sequences $\{\mathbf{H}_j\}_{j=1}^{M}$. We obtain a new representation $\mathbf{G}$ of the concatenated sequence by applying a residual connection between $\mathbf{H}$ and token-level self-attention:

$$\mathbf{G} = \mathbf{H} + \mathrm{MHA}(\mathbf{H}), \tag{9}$$

where $\mathrm{MHA}(\cdot)$ is a simplified MHA function with one argument because $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ all come from the same $\mathbf{H}$.

In practice, we also add the static (sinusoid) positional encodings (PE) to the input of MHA.[3] We adopt this procedure from the original Transformer's sub-layer (Vaswani et al., 2017). The computation cost of Eq. (9) is not high. Concretely, let $L = 128$ and $M = 5$. The length of the concatenated sequence is thus 640 ($L \times M$), which is slightly longer than the maximum length of BERT's input sequence (i.e., 512 tokens).

Next, we perform sentence-level self-attention using a similar procedure. First, we split $\mathbf{G}$ back into individual sequences $\{\mathbf{G}_j\}_{j=1}^{M}$. Then, we pick the first hidden state vector from each $\mathbf{G}_j$, which corresponds to that of the CLS token. Let $\mathbf{F}$ be the concatenation of all the first hidden state vectors $\{\mathbf{g}_{j,1}\}_{j=1}^{M}$. We obtain the final representation $\mathbf{E}$ of the evidence sentences:

$$\mathbf{E} = \mathbf{F} + \mathrm{MHA}(\mathbf{F}). \tag{10}$$

We can use $\mathbf{E}$ as the keys and values in Eq. (5). Note that we do not share the parameters among the different MHA layers.

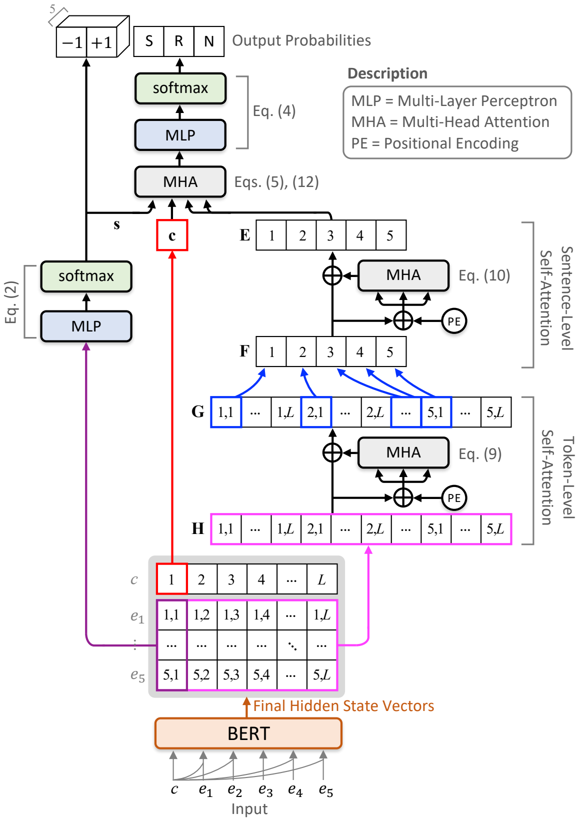

Figure 3: Architecture of our multi-level attention (MLA) model. The model takes as input a claim together with five evidence sentences. These sentences can be derived from any sentence selector. BERT encodes each sentence into a sequence of hidden state vectors, each of which is denoted by a squared box. The first hidden state vector (corresponding to the CLS token) is used for classification. MLA applies token- and sentence-level self-attentions and combines all the hidden state vectors as well as the sentence-selection scores at the final attention layer.

Joint training

Since the sentence-selection label assigned to each evidence sentence is available at training time, we can use it to guide our veracity prediction model. We apply the idea of multi-task learning (MTL) (Caruana, 1993; Ruder, 2017), in which we consider veracity relation prediction to be our main task and evidence sentence selection to be our auxiliary task. Our goal is to leverage training signals from our auxiliary task to improve the performance of our main task. Note that the sentence-selection component here is independent of the stand-alone model (i.e., our model in Section 3.1 or an alternative model in Section 4.3).

Let $\mathbf{s} = [s_1, \ldots, s_M]$ be the vector of sentence-selection scores, where s_m denotes the probability distribution of the positive class returned by our

---

[3] During development, we tried the other basic components, i.e., layer normalization and position-wise feed-forward, but found it yielded no improvements in our task.

sentence-selection component. We propose using $\mathbf{s}$ as a gate vector to determine how much of the values should be maintained before applying a residual connection followed by a linear projection. Thus, we modify Eq. (8) with:

$$\mathrm{head}_i = \mathrm{attn}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \widetilde{\mathbf{V}}\mathbf{W}_i^V), \quad (11)$$

$$\widetilde{\mathbf{V}} = \mathbf{V} + \mathbf{s} \odot \mathbf{V}, \quad (12)$$

where $\odot$ means the element-wise multiplication.

Our modification is close to Shaw et al. (2018)'s method in which extra vectors are added to the keys and the values after applying the linear projections. During development, we found that their method does not work well in our task. We compare different strategies in Section 4.4, including applying the gate vector to the keys or both the keys and the values.

Finally, we combine Eqs. (1) and (3) to get our composite loss function:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L} = \mathcal{L}_p + \lambda \sum_{j=1}^{M} \mathcal{L}_{e_j}, \quad (13)$$

where λ is the weighting factor of the sentence-selection component.

To summarize, our model, shown in Figure 3, contains token-level attention over a claim-evidence pair through BERT, token- and sentence-level self-attentions across an evidence set, and claim-evidence cross-attention incorporating the sentence-selection scores through joint training. Hence, we call it the multi-level attention (MLA) model.

4 Experiments

4.1 Dataset and evaluation metrics

Table 1 shows the statistics of the FEVER dataset. We used the corpus of the June 2017 Wikipedia dump, which contains 5,416,537 articles preprocessed by Thorne et al. (2018). We used the document retrieval results given by Hanselowski et al. (2018), containing the predicted Wikipedia article titles (i.e., document IDs) for all the claims in the training/dev/test sets. Following (Stammbach and Neumann, 2019; Soleimani et al., 2019; Liu et al., 2020), we prefixed the Wikipedia article titles to the candidate sentences to alleviate the co-reference resolution problem.

We evaluated performance by using the label accuracy (LA) and FEVER score. LA measures the 3-way classification accuracy of veracity relation prediction. The FEVER score reflects the performance of both evidence sentence selection and veracity relation prediction, where a complete set of true evidence sentences is present in the selected sentences, and the claim is correctly labeled. We used the official FEVER scorer during development.[4] We limited the number of the selected sentences to five (M = 5) according to the FEVER scorer. The performance on the blind test set was evaluated through the FEVER Challenge site.

Split	SUPPORTED	REFUTED	NOTENOUGHINFO
Training	80,035	29,775	35,659
Dev	6,666	6,666	6,666
Test	6,666	6,666	6,666

Table 1: Statistics of the FEVER dataset. Veracity relation labels and evidence sentences of the test set are not publicly available.

4.2 Training details

We implemented our model on top of HuggingFace's Transformer (Wolf et al., 2020). The dimension of hidden state vectors d_h and the number of heads n varied according to those of the pre-trained models. We used BERT-base (d_h = 786; n = 12) for our stand-alone sentence-selection model and tried various BERT-style models for MLA.

We trained all models using Adafactor (Shazeer and Stern, 2018) with a batch size of 256, a linear learning rate decay, a warmup ratio of 0.06, and a gradient clipping of 1.0. Following the default configuration of HuggingFace's Transformer, we initialized all parameters by sampling from $\mathcal{N}(0, 0.02)$ and setting the biases to 0, except for the pre-trained models. We set λ in Eq. (13) to 1. We trained each model for 2 epochs with a learning rate of 5e-5, unless otherwise specified.

For regularization, we applied dropout (Hinton et al., 2012) with a probability of 0.1 to the MHA layers, MLP layers, and gated values in Eq. (12). We computed the class weight β_y in Eq. (3) by:

$$\beta_y = \frac{\hat{\beta}_y}{\sum_{y \in \mathcal{Y}} \hat{\beta}_y}, \quad \hat{\beta}_y = \frac{N}{|\mathcal{Y}| \times N_y},$$

where $\hat{\beta}_y$ is the balanced heuristic used in scikit-learn (Pedregosa et al., 2011) and β_y is normalized to sum to 1. In our case, N = 145,469 is the total

---

[4] `https://github.com/sheffieldnlp/fever-scorer`

number of training examples, $|\mathcal{Y}| = 3$ is the number of classes, and N_y is the number of training examples in y (i.e., the first row in Table 1). We interpreted $\hat{\beta}_y$ as the ratio of the balanced class distribution ($N/|\mathcal{Y}|$) to the observed one (N_y). Here, we wanted to penalize the less-observed classes, like REFUTED and NOTENOUGHINFO, more.

4.3 Results

Baselines

The use of different pre-trained and pipeline models in the previous work makes a fair comparison difficult. For this reason, we chose baseline models that use BERT-base for pre-training and Hanselowski et al. (2018)'s document retrieval results. We designed two sets of experiments.

First, we required that all the models use the same sentence-selection model, which is Hanselowski et al. (2018)'s ESIM.[5] For the veracity relation prediction, Hanselowski et al. (2018) incorporate ESIM with attention and pooling operations to get a representation of a claim and top five selected sentences. Soleimani et al. (2019) make five independent predictions for each claim-evidence pair and use a heuristic (Malon, 2018) to get a final prediction. GEAR (Zhou et al., 2019) is a graph-based model for evidence aggregating and reasoning. KGAT (Liu et al., 2020) is a kernel graph attention model. Second, we allowed different sentence-selection models. Soleimani et al. (2019) use HNM to select negative examples with the highest loss values, while our negative examples are sampled once from both the ground-truth and retrieved documents, as described in Section 3.1.

Table 2 shows the results of the two settings on the dev set. MLA outperforms the other baselines in both settings. Table 3 shows the sentence-selection results returned by the FEVER scorer. The precision, recall@5, and F1 are consistent across the three models. Hanselowski et al. (2018) use ESIM with a pairwise hinge loss, while Soleimani et al. (2019) use a pointwise loss with HNM. Our model is also a pointwise approach but simpler to train. Without sampling non-evidence sentences from the retrieved documents, all the scores drop by around 2%, indicating that our technique is useful. In the following sections, we will refer to our BERT-base sentence-selection results with MLA.

Model	LA	FEVER
Sentence selection with ESIM		
Hanselowski et al. (2018)	68.49	64.74
Soleimani et al. (2019)	71.70	69.79
GEAR† (Zhou et al., 2019)	74.84	70.69
KGAT† (Liu et al., 2020)	75.51	71.61
MLA (Ours)	**76.30**	**72.83**
Sentence selection with BERT-base		
Soleimani et al. (2019)‡	73.54	71.33
MLA (Ours)	**76.92**	**73.78**

Table 2: LA and FEVER score results on the dev set. All the models use the document retrieval results from Hanselowski et al. (2018). Results marked with † indicate using ESIM with a threshold filter, and ‡ indicates using BERT-base with HNM.

Model	Prec	Rec@5	F1
ESIM (Hanselowski et al., 2018)	24.08	86.72	37.69
BERT-base‡ (Soleimani et al., 2019)	25.13	88.29	39.13
BERT-base (Ours)	**25.63**	**88.64**	**39.76**
w/o sampling from retrieved docs.	23.59	87.18	37.13

Table 3: Sentence selection results on the dev set. Result marked with ‡ indicates using HNM.

Effect of pre-trained models

The next set of experiments examined the benefits of using different pre-trained models. ALBERT (Lan et al., 2020) is a lite BERT training approach that uses cross-layer parameter sharing and replaces next sentence prediction with sentence ordering. RoBERTa (Liu et al., 2019) is a robustly optimized BERT approach that introduces better training schemes, including dynamic masking, larger batch size, and other techniques. We chose these two BERT-style models because they can be easily plugged into our implementation without much modification.

Table 4 shows the results of the different pre-trained models on the dev set. For all the large pre-trained models, we decreased the learning rate to 2e-5 and trained them for 3 epochs. Additional results including training times can be found in Appendix A. As shown in the table, BERT and ALBERT perform similarly, while ALBERT has fewer parameters. RoBERTa offers consistent improvements over the other two models and achieves the best performance with its large model. Therefore,

---

[5] We used the sentence-selection results reproduced by Zhou et al. (2019).

Pre-trained model	# Params	LA	FEVER
BERT-base	117M	76.92	73.78
BERT-large	349M	77.27	74.10
ALBERT-base	20M	76.58	73.83
ALBERT-large	33M	76.94	74.24
RoBERTa-base	132M	77.54	74.41
RoBERTa-large	370M	**79.31**	**75.96**

Table 4: LA and FEVER score results of MLA on the dev set using different pre-trained models. The second column shows the number of parameters, including those from the pre-trained model and our task-specific layers (i.e., MHA and MLP layers).

Model	LA	FEVER
Hanselowski et al. (2018)	65.46	61.58
Yoneda et al. (2018)	67.62	62.52
Nie et al. (2019a)	68.21	64.21
GEAR† (Zhou et al., 2019)	71.60	67.10
SR-MRS† (Nie et al., 2019b)	72.56	67.26
BERT‡ (Soleimani et al., 2019)	71.86	69.66
KGAT$^{\diamondsuit}$ (Liu et al., 2020)	74.07	70.38
DREAM$^{\clubsuit}$ (Zhong et al., 2020)	76.85	70.60
HESM$^{\spadesuit}$ (Subramanian and Lee, 2020)	74.64	71.48
CorefRoBERTa$^{\diamondsuit}$ (Ye et al., 2020)	75.96	72.30
MLA$^{\diamondsuit}$(Ours)	**77.05**	**73.72**

Table 5: LA and FEVER score results on the blind test set. Results marked with † indicate using BERT-base, ‡ BERT-large, $^{\diamondsuit}$ RoBERTa-large, $^{\clubsuit}$ XLNet, and $^{\spadesuit}$ ALBERT-large.

we applied MLA with RoBERTa-large to the blind test set.

Comparison with state-of-the-art methods

Table 5 shows the results on the blind test set.[6] The results are divided into two groups. The first group represents the top scores of the FEVER shared task, including those of Hanselowski et al. (2018); Yoneda et al. (2018); Nie et al. (2019a). The second group contains recently published results after the shared task. GEAR (Zhou et al., 2019), KGAT (Liu et al., 2020), and DREAM (Zhong et al., 2020) are graph-based models. SR-MRS (Nie et al., 2019b) uses a semantic retrieval module for selecting evidence sentences. HESM (Subramanian and Lee, 2020) uses a multi-hop evidence retriever and a hierarchical evidence aggregation model. CorefRoBERTa (Ye et al., 2020) trains KGAT by using a pre-trained model that combines a coreference prediction loss. Their pre-trained model is initialized with RoBERTa-large's parameters and further trained on Wikipedia. MLA outperforms all the published models and yields 1.09% and 1.42% improvements in LA and FEVER score, respectively, over CorefRoBERTa. Additional sentence-selection results can be found in Appendix B.

Model	LA	FEVER
MLA (full)	**76.92**	**73.78**
w/o token-level self-attention	76.30	73.20
w/o sentence-level self-attention	76.50	73.41
w/o class weighting	76.44	73.14
w/o joint training	76.65	73.22

Table 6: Ablation studies of the proposed components on the dev set with BERT-base.

Model	LA	FEVER
MLA (w/ value)	**76.92**	**73.78**
w/ key	76.74	73.65
w/ key & value	76.82	73.60
w/ dot-product	76.70	73.51
w/o using s	76.64	73.47

Table 7: Ablation studies of different strategies for using the sentence-selection scores s on the dev set with BERT-base.

4.4 Ablation study

We conducted two sets of ablation studies on the dev set using MLA with BERT-base. First, we examined the effect of our proposed components. Table 6 shows that all the components contribute to the final results. Without class weighting, Eq. (3) falls back to the standard cross-entropy loss. Without joint training, MLA is a stand-alone veracity prediction model. These results suggest that token-level self-attention and class weighting are the two most important components of our model.

Second, we explored a number of strategies for exploiting the sentence-selection scores s. MLA basically uses s as a gate vector and only applies it to the values, as described in Eq. (12). We can apply the same calculation to the keys or both the keys and the values. In addition, we can use s as a bias vector and add it to the scaled dot-product term, as done by Yang et al. (2018). Table 7 shows the results of the aforementioned strategies. These results indicate that applying s to the values produces the best results.

---

[6] The results can also be found on the FEVER leaderboard: `https://competitions.codalab.org/competitions/18814#results`

ID: 35237
Claim: Philomena is a film nominated for seven awards.
Evidence: [*Philomena_(film)*] It was also nominated for four BAFTA Awards and three Golden Globe Awards.^{9}
Annotated label: SUPPORTED
Predicted label: REFUTED

(a)

ID: 33547
Claim: Mick Thomson was born in Ohio.
Evidence: [*Mick_Thomson*] Born in Des Moines, Iowa, he is best known as one of two guitarists in Slipknot, in which he is designated #7.^{1}
Annotated label: SUPPORTED
Predicted label: REFUTED

(b)

ID: 73443
Claim: Heavy Metal music was developed in the United Kingdom.
Evidence: [*Heavy_metal_music*] Heavy metal (or simply metal) is a genre of rock music that developed in the late 1960s and early 1970s, largely in the United Kingdom and the United States.^{0}
Annotated label: REFUTED
Predicted label: SUPPORTED

(c)

ID: 212780
Claim: Harvard University is the first University in the U.S.
Evidence: [*Harvard_University*] Established originally by the Massachusetts legislature and soon thereafter named for John Harvard (its first benefactor), Harvard is the United States' oldest institution of higher learning ...^{3}
Annotated label: SUPPORTED
Predicted label: NOTENOUGHINFO

(d)

Table 8: Examples where the models disagree with the annotated labels.

4.5 Error analysis

To better understand the limitations of our method, we manually inspected 100 prediction errors on the dev set, where the true evidence sentences are present in the predicted sentences but MLA failed to predict the veracity relation labels. Here, we required that both BERT-base and RoBERTa-large MLA models produce the same errors.

Table 8(a) shows a prediction error requiring complex reasoning that our models are unable to deal with. The claim "*Philomena is a film nominated for **seven** awards.*" is supported by the evidence "*It was also nominated for **four** BAFTA Awards and **three** Golden Globe Awards.*". In this case, the models must understand that four plus three equals seven.

Table 8(b) shows a possible annotation error. The claim "*Mick Thomson was born in **Ohio**.*" is annotated as SUPPORTED, while the evidence "*Born in Des Moines, **Iowa**, he is best known as ...*" refutes the claim. Our models also predict REFUTED.

Table 8(c) shows the half-true claim "*Heavy Metal music was developed in **the United Kingdom**.*", which is annotated as REFUTED. However, the evidence "*Heavy metal (or simply metal) is ... developed ... in **the United Kingdom and the United States**.*" would indicate that the claim is partly true. The half-true label is defined in some previous smaller datasets (Vlachos and Riedel, 2014; Wang, 2017), but not in the FEVER dataset.

Table 8(d) shows the questionable claim "*Harvard University is **the first University** in the U.S.*", which is annotated as SUPPORTED by the evidence "*... Harvard is the United States' **oldest institution** of higher learning ...*". However, this evidence does not directly support the claim.[7] Our models predict NOTENOUGHINFO. Our analysis results suggest that probing disagreements between an ensemble of models and annotators may help improve annotation consistency. Additional results on error analysis are given in Appendix C.

5 Conclusion

We have presented a multi-level attention model that operates on linear sequences. We find that, when trained properly, the model outperforms its graph-based counterparts. Our results suggest that a sequence model is sufficient and can serve as a strong baseline. Using better upstream components (i.e., a better document retriever or sentence selector) or larger pre-trained models would further improve the performance of our model. Training models that are robust to adversarial examples while maintaining high performance for normal ones is an important direction for our future work.

Acknowledgments

We thank Erica Cooper (NII) for providing valuable feedback on an earlier draft of this paper. This work is supported by JST CREST Grants (JPMJCR18A6 and JPMJCR20D3) and MEXT KAKENHI Grants (21H04906), Japan.

---

[7] The topic is still under debate: `https://en.wikipedia.org/wiki/First_university_in_the_United_States`.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Richard Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of ICML*, pages 41–48.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to answer open-domain questions. In *Proceedings of ACL*, pages 1870–1879.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced LSTM for natural language inference. In *Proceedings of ACL*, pages 1657–1668.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges*, volume 3944 of *LNAI*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108.

Christopher Hidey, Tuhin Chakrabarty, Tariq Alhindi, Siddharth Varia, Kriste Krstovski, Mona Diab, and Smaranda Muresan. 2020. DeSePtion: Dual sequence prediction and adversarial examples for improved fact-checking. In *Proceedings of ACL*, pages 8593–8606.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of ICLR*.

Xiangci Li, Gully Burns, and Nanyun Peng. 2020. A paragraph-level multi-task learning model for scientific fact-verification. In *Proceedings of The AAAI-21 Workshop on Scientific Document Understanding*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained fact verification with kernel graph attention network. In *Proceedings of ACL*, pages 7342–7351.

Christopher Malon. 2018. Team papelo: Transformer networks at FEVER. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 109–113.

Yixin Nie, Lisa Bauer, and Mohit Bansal. 2020. Simple compounded-label training for fact extraction and verification. In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 1–7.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019a. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of AAAI*, pages 6859–6866.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019b. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of EMNLP-IJCNLP*, pages 2553–2566.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of NAACL*, pages 464–468.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of ICML*, pages 4596–4604.

Amir Soleimani, Christof Monz, and Marcel Worring. 2019. BERT for evidence retrieval and claim verification. In *Proceedings of European Conference on Information Retrieval*, pages 359–366.

Dominik Stammbach and Guenter Neumann. 2019. Team DOMLIN: Exploiting evidence enhancement for the FEVER shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 105–109.

Shyam Subramanian and Kyumin Lee. 2020. Hierarchical Evidence Set Modeling for automated fact extraction and verification. In *Proceedings of EMNLP*, pages 7798–7809.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of NAACL*, pages 809–819.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359:1146–1151.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of EMNLP*, pages 7534–7550.

William Yang Wang. 2017. “liar, liar pants on fire”: A new benchmark dataset for fake news detection. In *Proceedings of ACL*, pages 422–426.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*, pages 38–45.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of EMNLP*, pages 4449–4458.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of NeurIPS*, pages 5753–5763.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential Reasoning Learning for Language Representation. In *Proceedings of EMNLP*, pages 7170–7186.

Wenpeng Yin and Dan Roth. 2018. TwoWingOS: A two-wing optimization strategy for evidential claim verification. In *Proceedings of EMNLP*, pages 105–114.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. UCL machine reading group: Four factor framework for fact finding (HexaF). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *Proceedings of ACL*, pages 6170–6180.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of ACL*, pages 892–901.

A Additional results on different pre-trained models

Table 9 shows the results of different pre-trained models in detail. All the pre-trained models used in our experiments also come from HuggingFace.[8] We conducted each experiment on a single NVIDIA Tesla A100 GPU with 40 GB RAM. We used a batch size of 256 with gradient accumulation to control memory.

B Additional sentence-selection results

Table 10 shows the results of various sentence-selection models on the test set. Not all published models report precision and recall. Our precision, recall@5, and F1 scores are slightly better than those of Liu et al. (2020). Our sentence-selection model took 1 hour and 10 minutes to train. We find that getting high recall in evidence sentence selection is necessary to achieve good results in veracity relation prediction.

C Additional error analysis

Here, we provide additional examples of errors, including complex reasoning errors (Table 11), possible annotation errors (Table 12), half-true claims (Table 13), and questionable claims (Table 14).

---

[8] `https://huggingface.co/transformers/pretrained_models.html`.

Pre-trained model	# Params	Learning rate	Epochs	Time	LA	FEVER
BERT-base	117M	5e-5	2	46m	76.92	73.78
BERT-large	349M	2e-5	3	2h 50m	77.27	74.10
ALBERT-base	20M	5e-5	2	57m	76.58	73.83
ALBERT-large	33M	2e-5	3	3h 35m	76.94	74.24
RoBERTa-base	132M	5e-5	2	45m	77.54	74.41
RoBERTa-large	370M	2e-5	3	2h 49m	**79.31**	**75.96**

Table 9: Additional results of MLA on the dev set using different pre-trained models.

Model	Loss	Pre-trained model	Prec	Rec@5	F1
Hanselowski et al. (2018)	Pairwise	–	–	–	36.97
Yoneda et al. (2018)	Pointwise	–	–	–	34.97
Nie et al. (2019a)	Pointwise	–	–	–	52.96
Zhou et al. (2019)	Pairwise & filtering	–	–	–	36.87
Nie et al. (2019b)	Pointwise	BERT-base	–	–	74.62
Soleimani et al. (2019)	Pointwise & HNM	BERT-base	–	–	38.61
Liu et al. (2020)	Pairwise	BERT-base	25.21	87.47	39.14
Zhong et al. (2020)	Pointwise	RoBERTa	25.63	85.57	39.45
Subramanian and Lee (2020)	Pointwise & multi-hop	ALBERT-base	–	–	52.78
Ye et al. (2020)	(adopting Liu et al. (2020)'s results)	–	–	–	39.14
This work	Pointwise	BERT-base	25.33	87.58	39.29

Table 10: Sentence-selection results on the blind test set. The F1 results can be found on the FEVER leaderboard: `https://competitions.codalab.org/competitions/18814#results`.

ID: 112396
Claim: Aristotle spent the majority of his life in Athens.
Evidence: [*Aristotle*] At seventeen or eighteen years of age, he joined Plato's Academy in Athens and remained there until the age of thirty-seven (c. 347 BC).^{2}
Annotated label: SUPPORTED
Predicted label: REFUTED

ID: 3111
Claim: Luis Fonsi was born in the eighties.
Evidence: [*Luis_Fonsi*] Luis Alfonso Rodríguez López-Cepero, more commonly known by his stage name Luis Fonsi, (born April 15, 1978) is a Puerto Rican singer, songwriter and actor.^{0}
Annotated label: REFUTED
Predicted label: SUPPORTED

ID: 64685
Claim: The Bassoon King is the full title a book.
Evidence: [*The_Bassoon_King*] The Bassoon King: My Life in Art, Faith, and Idiocy is a non-fiction book authored by American actor Rainn Wilson.^{0}
Annotated label: REFUTED
Predicted label: SUPPORTED

ID: 102001
Claim: Jens Stoltenberg was Prime Minister of Norway once.
Evidence: [*Jens_Stoltenberg*] Stoltenberg served as Prime Minister of Norway from 2000 to 2001 and from 2005 to 2013.^{4}
Annotated label: REFUTED
Predicted label: SUPPORTED

Table 11: Examples of prediction errors requiring complex reasoning.

ID: 117520
Claim: The host of The Joy of Painting was Bob Ross.
Evidence: [*Bob_Ross*] He was the creator and host of The Joy of Painting, an instructional television program that aired from 1983 to 1994 ...^{1}
Annotated label: REFUTED
Predicted label: SUPPORTED

ID: 114640
Claim: IMDb is not user-edited.
Evidence: [*IMDb*] The site enables registered users to submit new material and edits to existing entries.^{10}
Annotated label: SUPPORTED
Predicted label: REFUTED

ID: 137678
Claim: Food Network is available to approximately 96,931,000 pay television citizens.
Evidence: [*Food_Network*] As of February 2015, Food Network is available to approximately 96,931,000 pay television households ...^{8}
Annotated label: REFUTED
Predicted label: SUPPORTED

ID: 34195
Claim: Annie Lennox was named "The Greatest White Soul Singer Alive" by VH1.
Evidence: [*Annie_Lennox*] Lennox has been named "The Greatest White Soul Singer Alive" by VH1 ...^{19}
Annotated label: REFUTED
Predicted label: SUPPORTED

Table 12: Example of possible annotation errors.

ID:	174029
Claim:	The Endless River came out in 1995 and is Pink Floyd's fifteenth studio album.
Evidence:	[*The_Endless_River*] The Endless River is the fifteenth and final studio album by the English rock band Pink Floyd.^{0}
Annotated label:	REFUTED
Predicted label:	SUPPORTED
ID:	161094
Claim:	French Indochina was a grouping of territories.
Evidence:	[*French_Indochina*] French Indochina (previously spelled as French Indo-China) ... was a grouping of French colonial territories in Southeast Asia.^{0}
Annotated label:	REFUTED
Predicted label:	SUPPORTED
ID:	48148
Claim:	On Monday August 19, 1945, Ian Gillan was born.
Evidence:	[*Ian_Gillan*] Ian Gillan (born 19 August 1945) is an English singer and songwriter.^{0}
Annotated label:	SUPPORTED
Predicted label:	NOTENOUGHINFO
Note:	August 19, 1945 is Sunday, not Monday.
ID:	85350
Claim:	Andrew Kevin Walker was born on Monday August 14, 1964.
Evidence:	[*Andrew_Kevin_Walker*] Andrew Kevin Walker (born August 14, 1964) is an American BAFTA-nominated screenwriter .^{0}
Annotated label:	SUPPORTED
Predicted label:	NOTENOUGHINFO
Note:	August 19, 1945 is Friday, not Monday.

Table 13: Examples of half-true claims.

ID:	92900
Claim:	The Indian Institute of Management Bangalore offers a business executive training program.
Evidence:	[*Indian_Institute_of_Management_Bangalore*] It offers Post Graduate, Doctoral and executive training programmes.^{5}
Annotated label:	SUPPORTED
Predicted label:	NOTENOUGHINFO
Note:	The evidence does not specify that the institute offers a *business* executive training program.
ID:	46271
Claim:	Prescott, Arizona is in northern Yavapai County.
Evidence:	[*Prescott,_Arizona*] Prescott ... is a city in Yavapai County, Arizona, United States.^{0}
Annotated label:	SUPPORTED
Predicted label:	NOTENOUGHINFO
Note:	The evidence does not specify that Prescott is in the *northern* part of Yavapai County.
ID:	227779
Claim:	Lyon is a city in Southwest France.
Evidence:	[*Lyon*] Lyon had a population of 506,615 in 2014 and is France's third-largest city after Paris and Marseille.^{4}
Annotated label:	SUPPORTED
Predicted label:	REFUTED
Note:	The evidence does not directly support the claim.

Table 14: Examples of questionable claims.

RealTranS: End-to-End Simultaneous Speech Translation with Convolutional Weighted-Shrinking Transformer

Xingshan Zeng, Liangyou Li, Qun Liu
Huawei Noah's Ark Lab
`{zeng.xingshan,liliangyou,qun.liu}@huawei.com`

Abstract

End-to-end simultaneous speech translation (SST), which directly translates speech in one language into text in another language in real-time, is useful in many scenarios but has not been fully investigated. In this work, we propose RealTranS, an end-to-end model for SST. To bridge the modality gap between speech and text, RealTranS gradually downsamples the input speech with interleaved convolution and unidirectional Transformer layers for acoustic modeling, and then maps speech features into text space with a weighted-shrinking operation and a semantic encoder. Besides, to improve the model performance in simultaneous scenarios, we propose a blank penalty to enhance the shrinking quality and a Wait-K-Stride-N strategy to allow local reranking during decoding. Experiments on public and widely-used datasets show that RealTranS with the Wait-K-Stride-N strategy outperforms prior end-to-end models as well as cascaded models in diverse latency settings.

1 Introduction

Simultaneous speech translation (SST) (Fügen et al., 2007; Oda et al., 2014; Ren et al., 2020) aims to translate speech in one language into text in another language concurrently. It is useful in many scenarios, like synchronous interpretation in international conferences, automatic caption for live videos, etc. However, prior studies either focus on full sentence speech translation (ST) (Berard et al., 2016; Weiss et al., 2017; Liu et al., 2019) or simultaneous text-to-text machine translation (STT) (Cho and Esipova, 2016; Gu et al., 2017; Dalvi et al., 2018) which takes a segmented output from an automatic speech recognition (ASR) system as input. Such two-stage models (i.e., cascaded models) inevitably introduce error propagation and also increase translation latency (see Figure 1). Ren et al. (2020) propose an end-to-end SST system called SimulSpeech, but they ignore the modality gap between speech and text, which is important for improving translation quality (Liu et al., 2020).

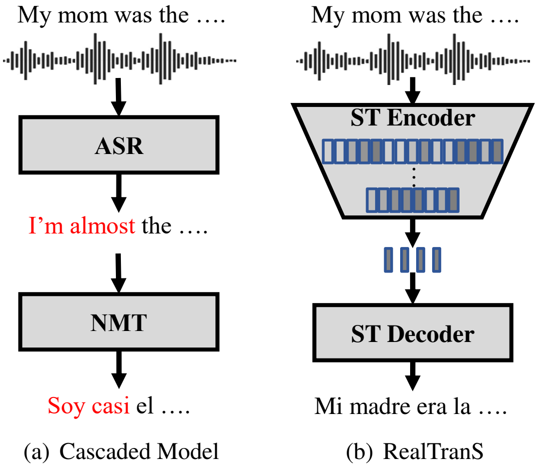

(a) Cascaded Model (b) RealTranS

Figure 1: An example for a cascaded model and our RealTranS. The ASR part in the cascaded model wrongly recognizes "My mom was" as "I'm almost". The error is propagated to NMT and leads to a wrong translation. RealTranS avoids such errors and translates accurately.

In this paper, we propose RealTranS model for SST. To relieve the burden of our encoder (Wang et al., 2020c; Liu et al., 2020), we decouple it into three parts: acoustic encoder, weighted-shrinking operation, and semantic encoder. We apply Conv-Transformer (Huang et al., 2020) as our acoustic encoder, which gradually downsamples the input speech and learns acoustic information with interleaved convolution and Transformer layers. The weighted-shrinking operation bridges the length gap between speech and text, by weighted summing up the frames in one detected segment based on the posterior probabilities generated by a CTC module (Graves et al., 2006). Finally, we use a semantic encoder to extract semantic features and deliver them to the decoder for translation.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2461–2474
August 1–6, 2021.

To enable simultaneous decoding, unidirectional Transformer is applied in our encoder. This inevitably affects the performance of the CTC module and the following shrinking operation. To alleviate this, we introduce a blank-limited CTC loss, which adds a blank penalty to the traditional CTC loss to encourage the model to produce non-blank labels, given the observation that CTC tends to produce peaky distribution as a kind of overfitting (Liu et al., 2018) by over predicting blank labels. Accordingly, the shrinking quality can be improved. Furthermore, we propose a new simultaneous strategy Wait-K-Stride-N which allows local reranking during decoding. This strategy can resolve the inherent drawback of the conventional Wait-K strategy (Ma et al., 2019), which cannot apply vanilla beam search efficiently (Zheng et al., 2019c).

Experiments on Augmented LibriSpeech En–Fr, MUST-C En–Es and En–De datasets demonstrate the effectiveness of the Wait-K-Stride-N strategy, and show that RealTranS achieves better performance than the prior end-to-end model SimulSpeech (Ren et al., 2020) as well as the cascaded models. Further analysis and ablation study reveal the effects of our proposed modules in RealTranS. We also compare RealTranS with other methods on full sentence ST. Results show that RealTranS achieves competitive or even better results, indicating its superiority.

In summary, the contributions of this work include the following aspects:

- We propose RealTranS for SST, which can gradually bridge the modality gap between speech and text with the help of gradual downsampling and weighted shrinking.
- We introduce a blank penalty and the Wait-K-Stride-N strategy to improve the performance in simultaneous translation scenarios.
- Extensive experiments on public and widely-used datasets show the superiority of our RealTranS model and our Wait-K-Stride-N strategy in diverse latency settings.

2 Related Work

Speech Translation. Speech translation (ST) has recently attracted intensive attention from the AI community. Earlier works are mostly based on cascaded models, which perform NMT on the outputs of ASR systems (Ney, 1999; Mathias and Byrne, 2006; Sperber et al., 2017; Bahar et al., 2021). Cascaded models inevitably introduce error propagation from ASR (Weiss et al., 2017). To avoid this problem and for better efficiency, end-to-end ST models are proposed and become popular in recent years (Berard et al., 2016, 2018; Bansal et al., 2018; Gangi et al., 2019). To alleviate the data scarcity problem of end-to-end ST models, various techniques are utilized, including pre-training (Bansal et al., 2019), multi-task learning (Anastasopoulos and Chiang, 2018), knowledge distillation (Liu et al., 2019; Ren et al., 2020), data synthesis (Jia et al., 2019), self-supervised learning (Chen et al., 2020) and speech augmentation techniques like SpecAugment (Bahar et al., 2019) or speed perturbation (Stoian et al., 2020).

Some studies focus on how to bridge the gap between different modalities (speech and text) or different modules (acoustic and semantic modeling). Wang et al. (2020b,c) propose a TCEN model and a curriculum pre-training technique to make sure the modules learn desired information, respectively. Salesky and Black (2020) explore phone features as intermediate representations to improve performance, while Dong et al. (2020b) use pre-trained BERT to guide the model to learn semantic knowledge. Modality Agnostic Meta-Leaning is also exploited for ST in Indurthi et al. (2019). To bridge the length gap, Zhang et al. (2020) propose adaptive feature selection, while Dong et al. (2020a) and Liu et al. (2020) exploit the CTC-based (Graves et al., 2006) shrinking mechanism. Nevertheless, they do not explore in simultaneous scenarios, where encoding quality inevitably suffers because of lacking future information in unidirectional encoders.

Simultaneous Translation. Previous studies on simultaneous translation focus on text-to-text scenarios (STT) (Cho and Esipova, 2016; Gu et al., 2017; Dalvi et al., 2018), where fixed policies (Ma et al., 2019) and adaptive policies (Arivazhagan et al., 2019; Zheng et al., 2019a,b) are proposed to decide when to read and write tokens. (Ma et al., 2019) proposed a simple yet effective strategy, Wait-K, based on a prefix-to-prefix framework. It first waits for the first k tokens, and then start to generate target tokens concurrently with the source stream. It achieves competitive performance in simultaneous translation (Zheng et al., 2019a, 2020).

Traditional simultaneous speech-to-text translation (SST) mainly depends on the ASR segmentation and then performs NMT based on the stream-

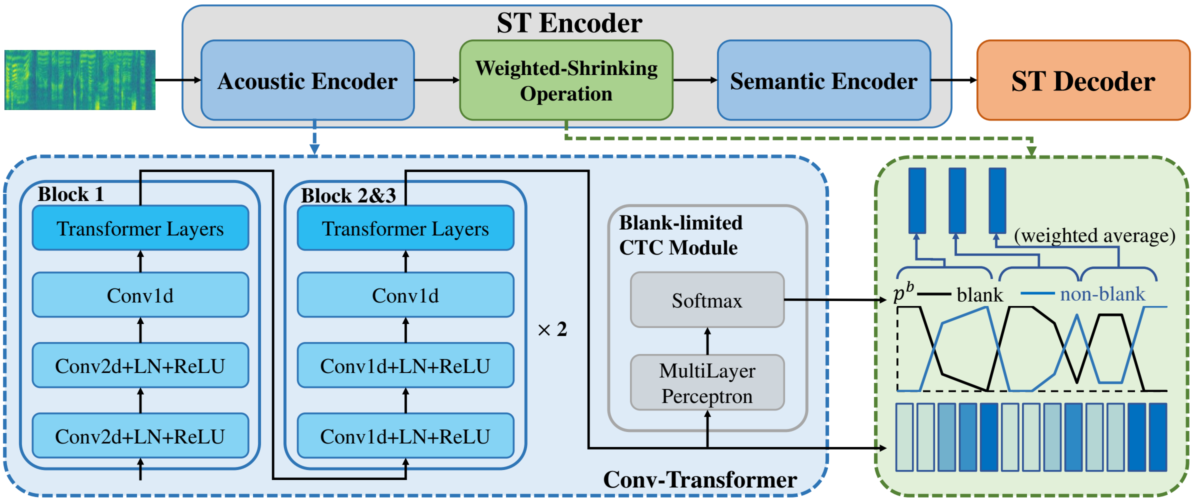

Figure 2: Overall structure of the proposed RealTranS model.

ing segmented chunks (Oda et al., 2014; Iranzo-Sánchez et al., 2020). There is little attention on end-to-end SST. Ren et al. (2020), to our knowledge, first propose an end-to-end model called SimulSpeech with multi-task learning and knowledge distillation, and apply the Wait-K strategy to perform simultaneous translation. Ma et al. (2020a) explore how to define a "token" in source speech, and then adapt methods from STT to SST. And Ma et al. (2020b) introduce a memory-augmented Transformer to tackle the streaming speech input. However, none of them investigate the modality gap between speech and text.

3 The RealTranS Model

Our RealTranS follows the sequence-to-sequence architecture, which consists of an ST encoder and an ST decoder. The ST encoder is decoupled into three parts, an acoustic encoder, a weighted shrinking operation, and a semantic encoder, to gradually map speech inputs into semantic representation space of text. Figure 2 shows the architecture.

3.1 Problem Formulation

Speech translation corpora usually contain triples of speech, transcription and translation, denoted as $\mathcal{D}_{ST} = \{(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y})\}$. Specifically, $\boldsymbol{x} = (x_1, x_2, ..., x_{T_x})$ is a sequence of speech features extracted from speech signals, e.g., filterbanks. $\boldsymbol{z} = (z_1, z_2, ..., z_{T_z})$ and $\boldsymbol{y} = (y_1, y_2, ..., y_{T_y})$ are the corresponding transcription in source language and translation in target language. T_x, T_z, and T_y are the lengths of speech, transcription and translation, respectively, where usually $T_x \gg T_z$ and $T_x \gg T_y$. A typical end-to-end model only makes use of $\boldsymbol{x}$ and $\boldsymbol{y}$, while $\boldsymbol{z}$ can be used as multi-task training for other objectives, like CTC loss.

3.2 Acoustic Encoder

Acoustic encoder mainly encodes speech features $\boldsymbol{x}$ into a hidden space to learn acoustic knowledge. We apply Conv-Transformer (Huang et al., 2020) to extract the desired features. It contains three blocks, each of which is composed of three convolution layers followed by unidirectional Transformer layers (see lower left in Figure 2) to prevent from leveraging future context in SST. Similar to Huang et al. (2020), we make the model aware of limited future frames with a look-ahead window in the convolution layers, to help improve acoustic modeling. At the same time, we gradually downsample the long speech features by setting the stride size to 2 in the second convolution layer in each block. In this way, the speech features are gradually reduced and approach the length of the corresponding text.

To predict the word boundaries[1] in the speech input and further improve the learned acoustic features, we adopt the Connectionist Temporal Classification (CTC) (Graves et al., 2006) module on top of the acoustic encoder. This module contains a Multilayer Perceptron (MLP) followed by a Softmax operator. CTC predicts a path $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_{T'_x})$, where T'_x is the length of hidden states after the Conv-Transformer. And $\pi_t \in \mathcal{V} \cup \{\phi\}$ can be either a token in the source

---

[1] For ease of understanding, we use word boundaries here. Same methods can be applied to boundaries between phones, chars, subwords, etc., depending on the the granularity of the CTC loss units. We use subword units in our experiments.

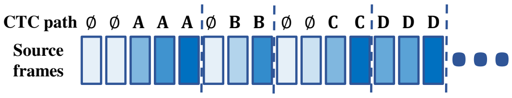

Figure 3: An example of how to define the word boundaries. The dash lines are the boundaries.

vocabulary $\mathcal{V}$ or the blank symbol ϕ. CTC paths have the many-to-one mapping to output sequences by removing blank symbols and consecutively repeated labels, denoted as operation $\mathcal{B}$. Therefore, CTC loss is defined as follows:

$$\mathcal{L}_{CTC} = -\sum_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{D}} \sum_{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{z})} p(\boldsymbol{\pi}|\boldsymbol{x}) \qquad (1)$$

where $\mathcal{B}^{-1}(\boldsymbol{z})$ denotes all possible CTC paths that can be mapped to the transcription $\boldsymbol{z}$. With the CTC module, we can define a word boundary between two frames where the first frame has a non-blank label while the second frame has a different label from the first one. Figure 3 shows an example of word boundaries.

Due to the data scarcity problem in ST and CTC's inherent characteristics (Liu et al., 2018), the detected word boundaries are usually not accurate enough. The module will overly predict the occurrence of blank labels (as a kind of overfitting), resulting in a large gap between the number of detected boundaries and the number of tokens in transcription, especially when unidirectional Transformer is applied (see Table 3). To alleviate the problem, we add a blank penalty to encourage the module to produce non-blank labels. It is called blank-limited CTC loss and defined as follows:

$$\mathcal{L}'_{CTC} = \mathcal{L}_{CTC} + \lambda \sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{\pi_t\in\boldsymbol{\pi}(\boldsymbol{x})} p(\pi_t = \phi|\boldsymbol{x}) \qquad (2)$$

where $\boldsymbol{\pi}(\boldsymbol{x})$ means the argmax results of CTC softmax outputs. λ controls the effect of blank penalty.

3.3 Weighted-Shrinking Operation

The length gap between acoustic features and the corresponding transcription and translation is still large after the gradual downsampling in Section 3.2. Inspired by previous studies (Dong et al., 2020a; Liu et al., 2020), we adopt a shrinking operation to bridge the gap based on CTC predictions. Prior works usually either remove blank frames and then average repeated frames (Dong et al., 2020a) or select a single representative frame (Liu et al., 2020) in a detected segment (i.e., frames between two word boundaries). However, there might be useful information in the detected blank or repeated frames, especially when the boundary detection is not accurate enough (see Section 3.2).

We propose a weighted-shrinking mechanism to tackle the problem. We assume that the probability of a frame to be labeled as "blank" represents the confidence that the model "thinks" it is not important. Therefore, for the frames in one segment, their weights are decided by the probabilities to be blank labels. The representation of the segment will be the weighted average of the corresponding frames. The specific operation is shown as follows:

$$\boldsymbol{h}_{t'} = \sum_{t\in seg\, t'} \boldsymbol{h}_t \frac{\exp(\mu(1-p_t^b))}{\sum_{s\in seg_{t'}} \exp(\mu(1-p_s^b))} \qquad (3)$$

where p_t^b denotes the probability of the frame t to be blank, and $\boldsymbol{h}_t$ represents the hidden state of the frame t in our acoustic encoder. $\mu \geq 0$ controls the temperature of the distribution (i.e., Softmax function). When $\mu = 0$, it means that we simply average the frames; and when $\mu \to \infty$ it degenerates to the general shrinking mechanism where only the representative frames with the highest confidence are selected.

3.4 Semantic Encoder

The shrinking operation only bridges the length gap between speech and text, but the shrunk representations still lack semantic information. Therefore, we apply a semantic encoder on top of the shrunk representations (Liu et al., 2020). It first applies a positional embedding layer, and then follows several Transformer layers (also unidirectional to mask future context), to extract semantic representations.

3.5 ST Decoder

A similar decoder as the basic Transformer architecture in NMT is adopted, where several Transformer decoder layers are stacked on top of target embeddings. To simulate simultaneous translation, we follow the prefix-to-prefix framework (Ma et al., 2019) and mask certain future context in cross attention to ensure that the model predicts the current token based on only part of the input from the ST encoder. How much context the model can see depends on the simultaneous strategy that is applied. For example, a Wait-K (Ma et al., 2019) based decoder predicts the t-th token based on the first $t+k-1$ hidden states produced by the ST encoder.

Wait-K-Stride-N. There is one drawback in the conventional Wait-K strategy – it cannot perform vanilla beam search while decoding except for the

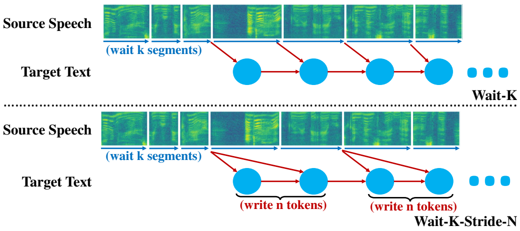

Figure 4: Wait-K vs. Wait-K-Stride-N (in this example, we set $k = 3$ and $n = 2$).

long-tail scenario (Ma et al., 2019; Zheng et al., 2019c), though beam search has been proven very effective in improving translation quality. Based on the prefix-to-prefix framework and STATIC-RW strategy proposed in Dalvi et al. (2018), we propose the Wait-K-Stride-N strategy to allow using beam search for local reranking during simultaneous decoding. Similar to the Wait-K strategy, our strategy first reads k input units (tokens in MT or segments in ST). Then, the model repeatedly performs n write and read operations until the end of the sentence (see Figure 4). In this way, the translation latency is close to Wait-K, but we can perform beam search during the n write operations. The objective with such a strategy is hence defined as follows:

$$\mathcal{L}_{ST} = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \prod_{t=1}^{T_y} p(y_t|y_{<t}, x_{\leq n\lfloor (t-1)/n \rfloor + k}) \quad (4)$$

where $y_{<t}$ represents the target tokens before y_t, T_y is the length of the target sentence, and $x_{\leq t}$ represents the first t detected source segments.

3.6 Training Procedure

The total objective of our model will be the sum of the CTC part and the ST part:

$$\mathcal{L} = \mathcal{L}_{ST} + \alpha \mathcal{L}'_{CTC} \quad (5)$$

where α controls the influence of the CTC part.

To enhance the CTC quality, we also apply a pre-training procedure (Stoian et al., 2020). We only use CTC loss to pre-train the acoustic encoder[2]. In this way, we can prevent the training waste (Wang et al., 2020b), and focus on improving the alignment results, which are essential for shrinking operations (see Table 3). The whole model is then fine-tuned with the whole ST corpus.

Corpus	Train	Dev	Test
En–Fr	47,271×2 (100h)	1,071 (2.0h)	2,048 (4.0h)
En–Es	265,625 (496h)	1,316 (2.5h)	2,502 (4.0h)
En–De	229,703 (400h)	1,423 (2.5h)	2,641 (4.0h)

Table 1: The number of sentences and the duration of audios for Augmented LibriSpeech En–Fr, MuST-C En–Es and En–De datasets.

4 Experimental Setup

4.1 Datasets

We conduct experiments on three publicly available datasets: Augmented LibriSpeech English-French (En–Fr) corpus (Kocabiyikoglu et al., 2018), and MUST-C English-Spanish (En–Es) and English-German (En–De) corpus (Di Gangi et al., 2019). All the datasets include source audios with the corresponding transcriptions in source language and translations in target language. For the Augmented Librispeech En–Fr corpus, we follow previous work (Wang et al., 2020c) and use the 100-hour clean training set with aligned references and provided Google translations, resulting in double size of training pairs. For MUST-C datasets, We use the official data splits for train and development and tst-COMMON set for test. The statistics for these three datasets are listed in Table 1.

4.2 Experimental Settings

We use 80-dimensional log-mel filterbanks as acoustic features, which are calculated with 25ms window size and 10ms step size and normalized by utterance-level Cepstral Mean and Variance Normalization (CMVN). For transcriptions and translations, SentencePiece[3] (Kudo and Richardson, 2018) is used to generate subword vocabularies with the sizes of 4k and 8k respectively. We remove the punctuation in transcriptions.

Our acoustic encoder follows the settings of the original Conv-Transformer (Huang et al., 2020), except that the channel number in convolution layers and hidden size and head number in Transformer layers are half values of theirs. This means the output dimension of the acoustic encoder is 256. We use 6 Transformer layers in the semantic encoder and 4 in the ST decoder. The hyper-parameters λ (Eq. 2), μ (Eq. 3) and α (Eq. 5) in our model are set to 0.5, 1.0 and 1.0, respectively.

---

[2] We do not pre-train the decoder for simplicity though it might further improve our performance.

[3] `https://github.com/google/sentencepiece`

Our model is trained with 8 NVIDIA Tesla V100 GPUs, batched with an approximate 40000-frame features. We use Adam optimizer (Kingma and Ba, 2015) with a 0.002 learning rate and 10000 warm-up steps followed by the inverse square root scheduler. Dropout strategy is used with a rate of 0.1. We save checkpoints every epoch and average the last 10 checkpoints for evaluation with a beam size of 5. For simplicity, we use the same K and N values as those of training for inference. We implement our model based on Fairseq S2T[4] (Wang et al., 2020a).

4.3 Evaluation Metrics

We apply SacreBLEU[5] for translation quality evaluation unless otherwise stated. For the metrics of latency, we adapt Average Proportion (AP) (Cho and Esipova, 2016) and Average Lagging (AL) (Ma et al., 2019) to ST settings, following previous studies (Ren et al., 2020; Ma et al., 2020a).

Average Proportion. AP calculates the mean absolute latency cost by each target token, where we replace the steps of source tokens with the time spent. It can be calculated as follows:

$$AP(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{|\boldsymbol{x}||\boldsymbol{y}|}\sum_{i=1}^{|\boldsymbol{y}|} d(y_i) \qquad (6)$$

where $d(y_i)$ is the speech duration that has been listened when producing the target token y_i.

Average Lagging. AL evaluates the degree of that the user is out of sync with the speaker, in terms of the number of source tokens (Ma et al., 2019). Following Ma et al. (2020a), we also extend it to the basis of time duration rather than source tokens, which is defined as follows:

$$AL(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{\tau(|\boldsymbol{x}|)}\sum_{i=1}^{\tau(|\boldsymbol{x}|)}\left[d(y_i) - \frac{|\boldsymbol{x}|}{|\boldsymbol{y}^*|}T_s(i-1)\right] \qquad (7)$$

where $|\boldsymbol{y}^*|$ is the length of the reference translation, and $\tau(|\boldsymbol{x}|)$ denotes the index of the corresponding target token when our model has read the entire source speech. T_s represents that the speech features are extracted every T_s ms (decided by the step size in the feature extraction and downsampling rates in convolution layers), which will be 80ms in our model. What's more, our Conv-Transformer module introduces a 140ms look-ahead window (Huang et al., 2020), so we add 140 to the final AL scores to be fairly compared with other models.

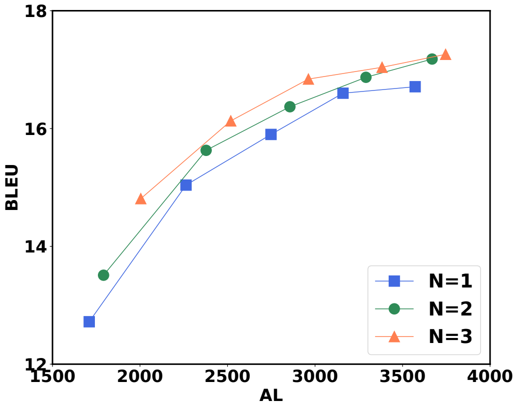

(a) Augmented LibriSpeech En–Fr

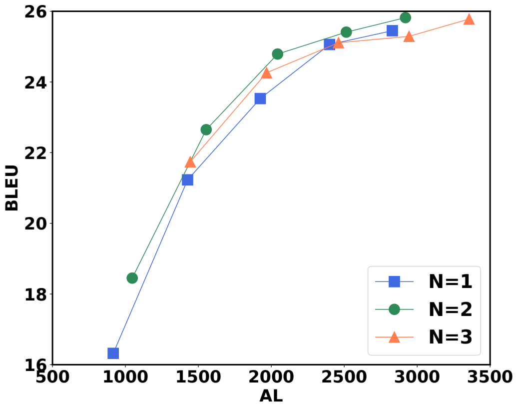

(b) MUST-C En–Es

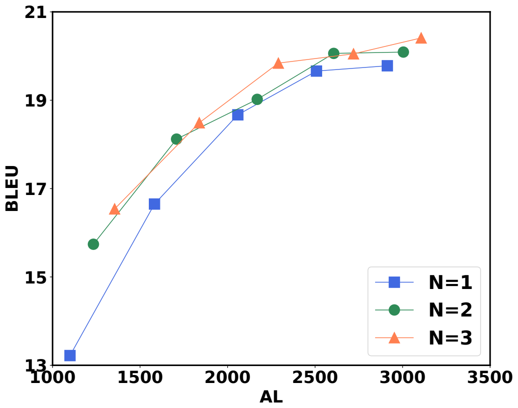

(c) MUST-C En–De

Figure 5: Translation quality (BLEU) vs. latency (AL) of our RealTranS model with Wait-K-Stride-N simultaneous strategy. For each dataset, we display the results of N=1, 2 and 3, with K=N, N+2, N+4, N+6, N+8.

5 Experimental Results

This section displays our experimental results. To explicitly show the performance trend of models in different scenarios, we use line charts to display

---

[4] `https://github.com/pytorch/fairseq/tree/master/examples/speech_to_text`

[5] `https://github.com/mjpost/sacreBLEU`

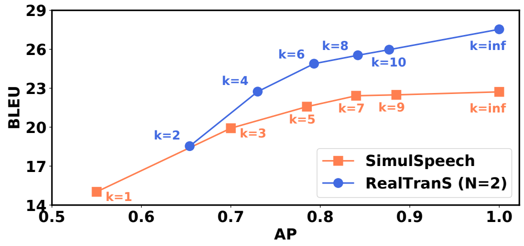

(a) Translation quality vs. latency in terms of AP

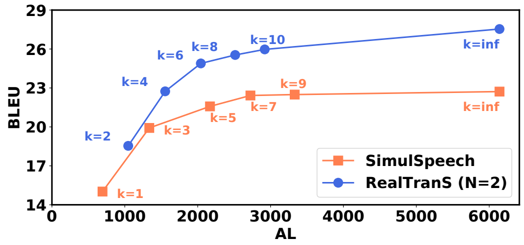

(b) Translation quality vs. latency in terms of AL

Figure 6: Translation quality (BLEU) vs. latency (AP and AL) comparison on MUST-C En-Es dataset.

most of the compared results. Their corresponding numeric results can be found in Appendix A.

5.1 Translation Quality vs. Latency

We first evaluate our RealTranS model with our Wait-K-Stride-N simultaneous strategy on the three datasets. We select three values 1, 2 and 3 for N to compare (it becomes conventional Wait-K when N=1). The results are displayed in Figure 5.

Results show that RealTranS achieves higher BLEU scores as the K value increases, with sacrfice of translation delay, consistent with prior works (Ren et al., 2020; Ma et al., 2020a).

Compared to the conventional Wait-K (N=1), our model with N=2 can achieve better BLEU scores under the same latency requirements, which demonstrates the effectiveness of our proposed Wait-K-Stride-N strategy. When N=3, the latency becomes higher. And it only achieves similar gains in BLEU scores compared to N=2 on MUST-C En–Es and En–De datasets. Therefore, we will use N=2 as our simultaneous strategy in later experiments unless otherwise stated.

5.2 Comparison with SimulSpeech

We compare with SimulSpeech (Ren et al., 2020), the state-of-the-art end-to-end model for SST. Figure 6 shows the performance comparison on the MUST-C En-Es dataset (we report tokenized case-sensitive BLEU scores following their settings). Since they only report segment-based AL, we transfer it to our time-based AL proportionally based on the latency when K=inf.

Model	K=2	K=4	K=6	K=8	K=10	K=inf
Cascaded	14.92	19.22	22.11	23.33	24.47	26.79
RealTranS	**18.45**	**22.65**	**24.79**	**25.41**	**25.82**	**27.40**

Table 2: Comparison with the cascaded model using Wait-K-Stride-N strategy when N=2 in terms of BLEU scores. The same K value means similar latency.

We find that our RealTranS model outperforms SimulSpeech almost in all latency settings, with an average of about 3 higher BLEU scores. Although SimulSpeech achieves relatively lower latency (e.g. less than 1000 ms AL when K=1), the performance inevitably suffers. What's more, SimulSpeech has leveraged multi-task learning and knowledge distillation to enhance their performance, which can be also applied to further improve the performance of our RealTranS model.

5.3 Comparison with Cascaded Model

We implement a cascaded model to compare with RealTranS under the same latency. Specifically, we combine our acoustic encoder (Section 3.2) and a Transformer decoder as our ASR model and use the conventional Transformer encoder-decoder architecture as our NMT model. Their configuration is similar to RealTranS (e.g. the same hidden dimension and the same number of Transformer layers). And we train the ASR and NMT model with the same corpus. The conventional Wait-K strategy is used in the ASR model since the alignment between speech and transcription is monotonic, while Wait-K-Stride-N is applied in the NMT model. Since several combinations of the ASR and NMT models may be under the same latency, we report the best BLEU score among them.

Table 2 shows the comparison results on MUST-C En–Es dataset. We have the following observations: 1) our RealTranS model outperforms the cascaded model in all latency settings, which demonstrates the superiority of RealTranS. 2) The improvement over the cascaded model becomes larger when the value of K is smaller. This observation is consistent with Ren et al. (2020). We attribute this to the advantage of end-to-end models over cascaded models, where the impact of error propagation in cascaded models may be amplified when the latency is low.

Model	Diff≤2	Diff≤4	Diff≤6	BLEU
Full Model	**52%**	**74%**	**85%**	**27.40**
- CTC PT	48%	70%	82%	26.63
- BP	35%	53%	67%	25.76
+ Bi-Enc	42%	59%	72%	26.58

Table 3: Shrinking quality of RealTranS. "CTC PT" indicates that the acoustic encoder is pre-trained with CTC. "BP" represents blank penalty, and "Bi-Enc" means that using bidirectional Transformer encoders. "Diff $\leq n$" means the difference between the length of shrunk representations and that of the ground-truth transcription is less than or equal to n. We report the percentage of the cases on the MUST-C En–Es test set. The BLEU scores displayed are results when K=inf.

5.4 Effects of Blank Penalty and Weighted Shrinking Operation

In this subsection, we examine the effects of our proposed methods, including blank penalty (Eq. 2) and weighted-shrinking operation (Eq. 3).

Blank Penalty. We propose a blank penalty to alleviate the inaccuracy of alignments between speech features and transcriptions when applying unidirectional encoders for SST (Section 3.2). To examine the effect, we evaluate the shrinking quality, i.e., the differences between the length of representations after the shrinking operation and that of the ground-truth transcription, on MUST-C En–Es dataset, and display the statistics in Table 3. We can see that the performance, as well as the shrinking quality, drops when removing CTC pre-training. It further decreases when removing the blank penalty, which shows its effectiveness. Also, we can see that the performance loss partly comes from using unidirectional encoders rather than bidirectional for simultaneous purposes (the 4th row), which can be compensated by the blank penalty.

Weighted-Shrinking. To validate our weighted-shrinking operation, we first investigate the effects of various values for the shrinking temperature μ in Eq. 3 and display the results in Figure 7(a). The results show that our weighted-shrinking mechanism ($\mu = 1.0$) performs better than both simply averaging all the frames ($\mu = 0$) and dropping blank frames ("DB").

We also try to replace our weighted-shrinking module with another Conv-Transformer block (see Section 3.2), resulting in a model with 240ms downsample rate in total (denoted as "4 blocks w/o shrink"). Figure 7(b) shows comparison results, together with a model with only 2 blocks (40ms downsample rate, denoted as "2 blocks with shrink"). We can find that downsampling only with convolution layers performs worse than RealTranS, while less downsample rate also affects the performance. This implies that there is an upper-bound for downsampling with only convolution layers while maintaining performance, and our weighted-shrinking operation can be used as an addition to further improve the performance.

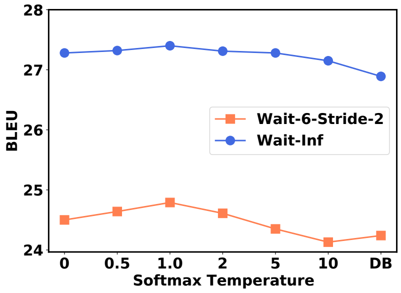

(a) Different shrinking methods

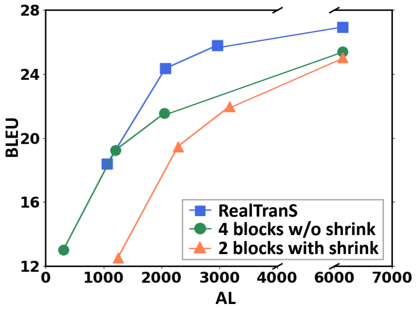

(b) Different downsampling methods

Figure 7: Upper: the BLEU scores for different shrinking methods, where "DB" means simply dropping blank frames. Lower: BLEU-AL tendency for different downsampling methods, where "n blocks" represents using n Conv-Transformer blocks in the acoustic encoder.

5.5 Ablation Study

We evaluate the contributions of different modules in RealTranS. Each module is evaluated in four kinds of latency settings: Wait-2-Stride-2, Wait-6-Stride-2, Wait-10-Stride-2, and also Wait-Inf (full sentence translation). The results are shown in Figure 8, where "-CTC PT" means we do not pre-train the encoder with CTC loss and "-BP" indicates we further remove the blank penalty. "-shrink" means removing the weighted-shrinking operation and the semantic encoder, while "-GD" denotes disabling the gradual downsampling by moving the Transformer layers in block 1 & 2 to block 3 (see Fig-

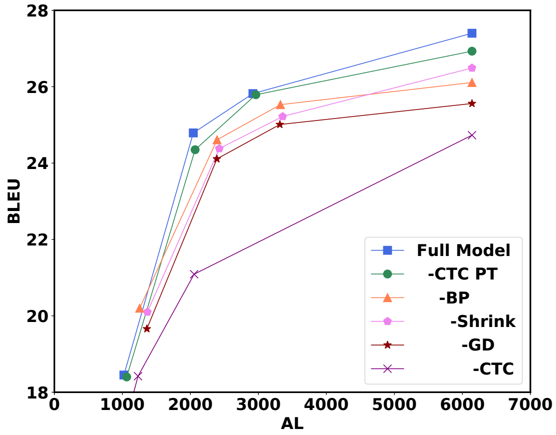

Figure 8: Ablation study for RealTranS. We choose the results of 3 latency settings and full sentence translation for each ablation. “GD” means gradual downsampling.

ure 2) to perform downsampling at the beginning layers of the acoustic encoder. Finally, “-CTC” indicates that CTC objective is removed.

Figure 8 shows that all modules play a role in RealTranS. Specifically, we have the following observations: 1) The CTC module is important for improving translation quality, and it can be further improved by pre-training. 2) The blank penalty is useful in reducing latency while maintaining translation quality. 3) The blank penalty is essential for the shrinking operation since on full sentence translation the shrinking operation degrades the performance without the blank penalty (“-BP” vs “-Shrink”). 4) Gradual downsampling also contributes to the performance, because directly downsampling to a large rate may make it difficult to learn acoustic features.

5.6 Comparison in Full Sentence Translation

Although focusing on SST, RealTranS can also be applied in full sentence ST. For fair comparison, we replace the unidirectional Transformer layers in RealTranS with bidirectional ones and report both case-insensitive and case-sensitive BLEU scores following prior works. Table 4 displays the BLEU scores compared with existing methods (we only compare with end-to-end models trained with the same data for fair comparison) on Augmented LibriSpeech En–Fr and MUST-C En–De dataset. RealTranS yields competitive (En–Fr) or even better (En–De) results, even though most of these prior methods depend on some extra techniques like pre-training decoders or using SpecAugment (Park et al., 2019). This validates the superiority of our proposed architecture.

Dataset	Method	BLEU
En-Fr	Transformer+KD (Liu et al., 2019)	17.02
	TCEN-LSTM (Wang et al., 2020b)	17.05
	Curriculum PT (Wang et al., 2020c)	17.66
	LUT (Dong et al., 2020b)	17.75
	STAST (Liu et al., 2020)	17.81
	COSTT (Dong et al., 2020a)	17.83
	Transformer+AFS (Zhang et al., 2020)	18.56*
	RealTranS (ours)	18.97 18.30*
En-De	Transformer+MAM (Chen et al., 2020)	21.87*
	Transformer+ML (Indurthi et al., 2019)	22.11*
	Transformer+AFS (Zhang et al., 2020)	22.38*
	Fairseq S2T (Wang et al., 2020a)	22.70*
	Espnet ST (Inaguma et al., 2020)	22.91*
	STAST (Liu et al., 2020)	23.06
	RealTranS (ours)	23.53 22.99*

Table 4: Full sentence translation results on Augmented LibriSpeech En–Fr and MUST-C En–De datasets. Results marked with * are case-sensitive BLEU scores, while the others are case-insensitive.

6 Conclusion

This work proposes a new end-to-end model RealTranS and a new strategy Wait-K-Stride-N for SST. RealTranS gradually bridges the modality gap between speech and text, and achieves new state-of-the-art results for SST. Empirical studies have shown the proposed blank penalty for CTC loss helps on the alignment with transcription, which reduces latency while maintaining translation quality. Our weighted-shrinking operation, as well as Wait-K-Stride-N simultaneous strategy, further improves the performance. We also compare RealTranS with other methods for full sentence translation, where RealTranS still exhibits competitive results, showing its superiority.

Acknowledgment

We thank MindSpore, a new deep learning computing framework, for the partial support of this work. Given the superior performance of Huawei Ascend AI Processor and MindSpore framework, our code will be released based on MindSpore at (`https://gitee.com/mindspore/mindspore/tree/master/model_zoo/research/nlp/realtrans`).

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 82–91, New Orleans, Louisiana. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Parnia Bahar, Tobias Bieschke, Ralf Schlüter, and Hermann Ney. 2021. Tight integrated end-to-end training for cascaded speech translation. In *IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, China, January 19-22, 2021*, pages 950–957. IEEE.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specaugment for end-to-end speech translation. *CoRR*, abs/1911.08876.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, pages 1298–1302. ISCA.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 58–68, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexandre Berard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 6224–6228. IEEE.

Alexandre Berard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *CoRR*, abs/1612.01744.

Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. 2020. MAM: masked acoustic modeling for end-to-end speech-to-text translation. *CoRR*, abs/2010.11445.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2020a. Consecutive decoding for speech-to-text translation. *CoRR*, abs/2009.09737.

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2020b. Listen, understand and translate: Triple supervision decouples end-to-end speech-to-text translation. *CoRR*, abs/2009.09704.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Mach. Transl.*, 21(4):209–252.

Mattia Antonino Di Gangi, Matteo Negri, and Marco Turchi. 2019. Adapting transformer to end-to-end spoken language translation. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1133–1137. ISCA.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Wenyong Huang, Wenchao Hu, Yu Ting Yeung, and Xiao Chen. 2020. Conv-transformer transducer: Low latency, low frame rate, streamable end-to-end speech recognition. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 5001–5005. ISCA.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. 2020. ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 302–311, Online. Association for Computational Linguistics.

Sathish Reddy Indurthi, Houjeung Han, Nikhil Kumar Lakumarapu, Beomseok Lee, Insoo Chung, Sangha Kim, and Chanwoo Kim. 2019. Data efficient direct speech-to-text translation with modality agnostic meta-learning. *CoRR*, abs/1911.04283.

Javier Iranzo-Sánchez, Adrià Giménez Pastor, Joan Albert Silvestre-Cerdà, Pau Baquero-Arnal, Jorge Civera Saiz, and Alfons Juan. 2020. Direct segmentation models for streaming speech translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2599–2611, Online. Association for Computational Linguistics.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J. Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 7180–7184. IEEE.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting librispeech with french translations: A multimodal corpus for direct speech translation evaluation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Hu Liu, Sheng Jin, and Changshui Zhang. 2018. Connectionist temporal classification with maximum entropy regularization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 839–849.

Yuchen Liu, Hao Xiong, Jiajun Zhang, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. 2019. End-to-end speech translation with knowledge distillation. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1128–1132. ISCA.

Yuchen Liu, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2020. Bridging the modality gap for speech-to-text translation. *CoRR*, abs/2010.14920.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020a. SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 582–587, Suzhou, China. Association for Computational Linguistics.

Xutai Ma, Yongqiang Wang, Mohammad Javad Dousti, Philipp Koehn, and Juan Pino. 2020b. Streaming simultaneous speech translation with augmented memory transformer. *CoRR*, abs/2011.00033.

Lambert Mathias and William Byrne. 2006. Statistical phrase-based speech translation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 14-19, 2006*, pages 561–564. IEEE.

MindSpore. http://www.mindspore.cn.

Hermann Ney. 1999. Speech translation: coupling of recognition and translation. In *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '99, Phoenix, Arizona, USA, March 15-19, 1999*, pages 517–520. IEEE Computer Society.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 551–556, Baltimore, Maryland. Association for Computational Linguistics.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz,* *Austria, 15-19 September 2019*, pages 2613–2617. ISCA.

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3787–3796, Online. Association for Computational Linguistics.

Elizabeth Salesky and Alan W Black. 2020. Phone features improve speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2388–2397, Online. Association for Computational Linguistics.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1380–1389, Copenhagen, Denmark. Association for Computational Linguistics.

Mihaela C. Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing ASR pretraining for low-resource speech-to-text translation. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 7909–7913. IEEE.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020a. Fairseq S2T: Fast speech-to-text modeling with fairseq. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 33–39, Suzhou, China. Association for Computational Linguistics.

Chengyi Wang, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou. 2020b. Bridging the gap between pre-training and fine-tuning for end-to-end speech translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9161–9168. AAAI Press.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020c. Curriculum pre-training for end-to-end speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3728–3738, Online. Association for Computational Linguistics.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. *CoRR*, abs/1703.08581.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. 2020. Adaptive feature selection for end-to-end speech translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2533–2544, Online. Association for Computational Linguistics.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2847–2853, Online. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.

Renjie Zheng, Mingbo Ma, Baigong Zheng, and Liang Huang. 2019c. Speculative beam search for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1395–1402, Hong Kong, China. Association for Computational Linguistics.

Appendix

A Numeric Results for the Figures

N Value	K=N		K=N+2		K=N+4		K=N+6		K=N+8	
	BLEU	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**
N=1	12.72	1710	15.04	2263	15.9	2748	16.6	3160	16.71	3572
N=2	13.51	1792	15.63	2378	16.37	2856	16.87	3291	17.18	3669
N=3	14.81	2004	16.13	2519	16.84	2962	17.04	3383	17.26	3746

Table 5: Numeric Results for Figure 5(a).

N Value	K=N		K=N+2		K=N+4		K=N+6		K=N+8	
	BLEU	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**
N=1	16.32	916	21.23	1426	23.53	1924	25.06	2399	25.45	2830
N=2	18.45	1047	22.65	1554	24.79	2043	25.41	2514	25.82	2920
N=3	21.74	1445	24.26	1968	25.11	2461	25.29	2944	25.78	3356

Table 6: Numeric Results for Figure 5(b).

N Value	K=N		K=N+2		K=N+4		K=N+6		K=N+8	
	BLEU	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**
N=1	13.22	1100	16.65	1582	18.67	2059	19.66	2508	19.78	2913
N=2	15.74	1233	18.12	1709	19.02	2169	20.06	2607	20.09	3005
N=3	16.54	1355	18.49	1838	19.84	2290	20.05	2720	20.41	3106

Table 7: Numeric Results for Figure 5(c).

Model	K=N		K=N+2		K=N+4		K=N+6		K=N+8		K=inf	
	BLEU	**AP**	**BLEU**	**AP**	**BLEU**	**AP**	**BLEU**	**AP**	**BLEU**	**AP**	**BLEU**	**AP**
SimulSpeech	15.02	0.550	19.92	0.700	21.58	0.785	22.42	0.840	22.49	0.885	22.72	1.0
RealTranS(N=2)	18.54	0.654	22.74	0.730	24.89	0.793	25.54	0.842	25.97	0.877	27.54	1.0

Table 8: Numeric Results for Figure 6(a).

Model	K=N		K=N+2		K=N+4		K=N+6		K=N+8		K=inf	
	BLEU	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**
SimulSpeech	15.02	694	19.92	1336	21.58	2169	22.42	2724	22.49	3331	22.72	6141
RealTranS(N=2)	18.54	1047	22.74	1554	24.89	2043	25.54	2514	25.97	2920	27.54	6141

Table 9: Numeric Results for Figure 6(b).

Model	μ**=0**	μ**=0.5**	μ**=1.0**	μ**=2.0**	μ**=5.0**	μ**=10**	**DB**
Wait-6-Stride-2	24.50	24.64	24.79	24.61	24.35	24.13	24.24
Wait-Inf	27.28	27.32	27.40	27.31	27.28	27.15	26.89

Table 10: Numeric Results for Figure 7(a).

Model	**K=2**		**K=6**		**K=10**		**K=inf**	
	BLEU	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**
RealTranS	18.40	1064	24.35	2070	25.79	2963	26.93	6141
4 blocks w/o shrink	13.03	307	19.25	1207	21.55	2051	25.38	6141
2 blocks with shrink	12.53	1247	19.48	2286	21.97	3175	25.04	6141

Table 11: Numeric Results for Figure 7(b).

Model	**K=2**		**K=6**		**K=10**		**K=inf**	
	BLEU	**AL**	**BLEU**	**AL**	**BLEU**	**AL**	**BLEU**	**AL**
Full Model	18.45	1023	24.79	2043	25.82	2920	27.40	6141
-CTC PT	18.40	1064	24.35	2070	25.79	2963	26.93	6141
-BP	20.20	1255	24.61	2393	25.53	3325	26.11	6141
-Shrink	20.10	1368	24.38	2426	25.22	3357	26.49	6141
-GD	19.66	1360	24.11	2390	25.01	3316	25.56	6141
-CTC	12.48	342	18.42	1230	21.09	2055	24.73	6141

Table 12: Numeric Results for Figure 8.

Training ELECTRA Augmented with Multi-word Selection

Jiaming Shen$^{\nabla\star}$, Jialu Liu$^{\diamond}$, Tianqi Liu$^{\diamond}$, Cong Yu$^{\diamond}$, Jiawei Han$^{\nabla}$
$^{\nabla}$University of Illinois Urbana-Champaign, IL, USA, $^{\diamond}$Google Research, NY, USA
$^{\nabla}$\{js2, hanj\}@illinois.edu $^{\diamond}$\{jialu, tianqiliu, congyu\}@google.com

Abstract

Pre-trained text encoders such as BERT and its variants have recently achieved state-of-the-art performances on many NLP tasks. While being effective, these pre-training methods typically demand massive computation resources. To accelerate pre-training, ELECTRA trains a discriminator that predicts whether each input token is replaced by a generator. However, this new task, as a binary classification, is less semantically informative. In this study, we present a new text encoder pre-training method that improves ELECTRA based on multi-task learning. Specifically, we train the discriminator to simultaneously detect replaced tokens and select original tokens from candidate sets. We further develop two techniques to effectively combine all pre-training tasks: (1) using attention-based networks for task-specific heads, and (2) sharing bottom layers of the generator and the discriminator. Extensive experiments on GLUE and SQuAD datasets demonstrate both the effectiveness and the efficiency of our proposed method.

1 Introduction

Contextualized representations from pre-trained text encoders have shown great power for improving many NLP tasks (Rajpurkar et al., 2016; Wang et al., 2019b,a; Liu and Lapata, 2019). Most pre-trained encoders, despite their variety, follow BERT (Devlin et al., 2019) and adopt the masked language modeling (MLM) pre-training task which trains the model to recover the identities of a small subset of masked tokens. Although being more effective than conventional left-to-right language model pre-training (Peters et al., 2018; Radford et al., 2018) due to capturing bidirectional information, MLM-based approaches (Liu et al., 2019b; Joshi et al., 2019) can only learn from those masked tokens which are typically just 15% of all tokens in the input sentences.

To address the low sample efficiency issue, ELECTRA (Clark et al., 2020a) proposes a new pre-training task. Specifically, it corrupts a sentence by replacing some tokens with plausible alternatives sampled from a generator and trains a discriminator to predict whether each token in the corrupted sentence is replaced or not. After pre-training ends, it throws away the generator and exports the discriminator for down-stream applications. As the discriminator can learn from all input tokens, ELECTRA is more sample efficient than previous MLM-based methods. However, follow-up studies (Xu et al., 2020; Aroca-Ouellette and Rudzicz, 2020) find this new replaced token detection task, as a binary classification, is often too simple to learn. As a result, the discriminator output representations are insufficiently trained and encode inadequate semantic information.

In this work, we propose a novel text encoder pre-training method **TEAMS** which stands for "Training ELECTRA Augmented with Multi-word Selection". Compared with ELECTRA, our method also consists of a generator and a discriminator but they are equipped with different pre-training tasks. For each masked position in the input sentence, the generator replaces the original token with an alternative token and samples a candidate set that consists of the original token and other K non-original ones. Then, we train the discriminator to simultaneously perform two tasks: (1) a *multi-word selection* task in which the discriminator learns to select the original token from the sampled candidate set, and (2) a *replaced token detection* task similarly defined in ELECTRA. The first task, as a $(K+1)-$way classification on the masked positions, pushes the discriminator to differentiate ground truth tokens from other negative non-original ones. At the same time, the second task, with reduced task complex-

---

[*] This work is done while interning at Google Research. Corresponding Author: Jialu Liu.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2475–2486

ity, keeps the discriminator to achieve the same sample efficiency as ELECTRA.

To further improve the performance and efficiency of our method, we introduce two techniques. The first one is using attention-based task-specific heads for discriminator multi-task pre-training. Different from previous studies (Liu et al., 2019a; Sun et al., 2020) that pass the last encoder layer outputs to different task heads, our method directly incorporates task-specific attention layers into the discriminator encoder. Such a design offers higher flexibility in capturing task-specific token dependencies in sequence and leads to significant performance boost. The second technique is to share the bottom layers of the generator and the discriminator. This technique reduces the number of parameters, saves computes, and serves as a form of regularization that stabilizes the training and helps the generalization.

Combining above novelties all together, we train our models of various sizes and test their performance on the GLUE natural language understanding benchmark (Wang et al., 2019b) and SQuAD question answering benchmark (Rajpurkar et al., 2016). We show that TEAMS substantially outperforms previous MLM-based methods and ELECTRA, given the same model size and pre-training data. For example, our base-sized model, achieving 84.51 SQuAD 2.0 F1 score, outperforms BERT and ELECTRA by 8.34 and 2.99, respectively. Moreover, TEAMS-Base can outperform ELECTRA-Base++ using a fraction of computes.

Contributions. The major contributions of this paper are summarized as follows: (1) We propose a new text encoder pre-training method TEAMS that simultaneously learns a generator and a discriminator using multi-task learning. (2) We develop two techniques, attention-based task-specific head and partial layer sharing, to further improve TEAMS performance. (3) We conduct extensive experiments to verify the effectiveness of TEAMS on GLUE and SQuAD benchmarks[1].

2 Background

In this section, we first discuss some related studies on pre-training text encoders. Then, we introduce our notations and describe ELECTRA in details.

2.1 Text Encoder Pre-training

Current state-of-the-art natural language processing systems often rely on a text encoder to generate contextualized representations. This text encoder is commonly pre-trained on massive unlabeled corpora using different self-supervised tasks. Peters et al. (2018) and Radford et al. (2018) pre-train either a LSTM or a Transformer (Vaswani et al., 2017) using the standard language modeling task. To further improve pre-trained models, more effective pre-training objectives have been developed, including masked language modeling and next sentence prediction in BERT (Devlin et al., 2019), permutation language modeling in XLNet (Yang et al., 2019), masked span prediction in SpanBERT (Joshi et al., 2019), sentence order prediction in StructBERT (Wang et al., 2020), and more.

Most pre-training methods demand massive amounts of computes, which limits their accessibilities and raises concerns about their environmental costs. To alleviate such issue, Gong et al. (2019) and Yang et al. (2020) propose to accelerate BERT training by progressively stacking a shallow model to a deep model. Gu et al. (2020) extend this idea by growing a low-cost model in different dimensions. Along another line of work, Clark et al. (2020a) propose a new pre-training task, named replaced token detection, that learns a text encoder to distinguish real input tokens from synthetically generated replacements. Compared to BERT-style MLM pre-training in which only 15% of tokens are utilized, ELECTRA can leverage *all tokens* in input sentences and thus achieves better sample efficiency. Following this idea, Xu et al. (2020) propose a new pre-training task based on the multi-choice cloze test with a rejection option, and Clark et al. (2020b) connect ELECTRA with cloze modeling and pre-train the text encoder as an energy-based cloze model. As our method is built upon ELECTRA, we discuss it in more detail below.

2.2 ELECTRA

ELECTRA jointly trains two models, a generator G and a discriminator D. Both models adopt the Transformer architecture as their backbones and map a sentence of n tokens $\mathbf{x} = [x_1, \ldots, x_n]$ to their corresponding contextualized representations $h(\mathbf{x}) = [h(\mathbf{x})_1, \ldots, h(\mathbf{x})_n]$.

The generator G is trained using the masked language modeling (MLM) task. Specifically, given an input sequence $\mathbf{x}$, it first randomly selects a few

---

[1] Code and pre-trained model weights are available at `https://github.com/tensorflow/models/tree/master/official/nlp/projects/teams`.

masked positions and replaces tokens at these positions with a special mask symbol `[MASK]`. We denote this masked sequence as $\mathbf{x}^M$. Then, the generator learns to predict the original identities of those masked-out tokens by minimizing the below MLM loss:

$$\mathcal{L}_{\text{MLM}}(\mathbf{x}; G) = \mathbb{E}\left(\sum_{i:x_i^M=\texttt{[MASK]}} -\log P_G(x_i|\mathbf{x}^M)\right), \tag{1}$$

where $P_G(x_i|\mathbf{x}^M)$ is the probability that G predicts token x_i appears in the masked position i in $\mathbf{x}^M$, and the expectation is taken over the random draw of masked positions. More specifically, the generator calculates $P_G(x_i|\mathbf{x}^M)$ by passing contextualized representations $h_G(\mathbf{x}^M)$ into a softmax layer as follows:

$$P_G(x_i|\mathbf{x}^M) = \frac{\exp(e(x_i)^T h_G(\mathbf{x}^M)_i)}{\sum_{x'_i \in V} \exp(e(x'_i)^T h_G(\mathbf{x}^M)_i)}, \tag{2}$$

where $e(x_i)$ is the embedding of token x_i and V denotes the vocabulary of all tokens. Finally, for each masked position i, the generator samples one token $\hat{x}_i \sim P_G(\cdot|\mathbf{x}^M)$ and replaces the original token x_i with $\hat{x}_i$. We use $\mathbf{x}^R$ to denote this corrupted sentence with replaced tokens.

The discriminator D learns to perform the replaced token detection (RTD) task that requires a model to predict whether each token in $\mathbf{x}^R$ is replaced or not. In particular, ELECTRA adopts a sigmoid layer, on top of the discriminator output contextualized representations $h_D(\mathbf{x}^R)$, to decide the probability that token x_i^R matches the original token x_i as follows:

$$P_D(x_i^R = x_i) = \text{sigmoid}(w^T h_D(\mathbf{x}^R)_i), \tag{3}$$

where w is a learnable parameter. The loss on D is then defined as follows:

$$\mathcal{L}_{\text{RTD}}(\mathbf{x}, \mathbf{x}^R; D) = \mathbb{E}\Big(\sum_{i:x_i^R=x_i} -\log P_D(x_i^R = x_i) + \sum_{i:x_i^R \neq x_i} -\log(1 - P_D(x_i^R = x_i))\Big). \tag{4}$$

Finally, the generator and discriminator are jointly learned based on losses in Eq. (1) and Eq. (4). After pre-training, ELECTRA throws out the generator and keeps only the discriminator for fine-tuning on downstream tasks.

3 The TEAMS Method

In this section, we first introduce a new pre-training task named "multi-word selection". Then, we present our TEAMS method with two techniques for performance improvements.

3.1 Multi-word Selection Task

To train a model on an input sequence $\mathbf{x} = [x_1, \ldots, x_n]$ using the multi-world selection task, we first choose a random set of positions in this sequence, denoted as $\{i_1, \ldots, i_m\}$ where m is an integer between 1 and n. Then, for each chosen position $i_j, j \in \{1, \ldots, m\}$, we replace token x_{i_j} with another token $\hat{x}_{i_j}$ and create a *candidate set* S_{i_j} that includes the original token x_{i_j} and K non-original ones. Following ELECTRA, we use $\mathbf{x}^R$ to denote the corrupted sentence with all tokens in chosen positions replaced. Finally, the model inputs the corrupted sentence and outputs a probability for selecting the original token x_{i_j} from the candidate set S_{i_j} as follows:

$$P(x_{i_j}|\mathbf{x}^R, S_{i_j}) = \frac{\exp(e(x_{i_j})^T h(\mathbf{x}^R)_{i_j})}{\sum_{x'_{i_j} \in S_{i_j}} \exp(e(x'_{i_j})^T h(\mathbf{x}^R)_{i_j})}, \tag{5}$$

where $h(\mathbf{x}^R)_{i_j}$ is the contextualized representation of token $\hat{x}_{i_j}$ from the model outputs.

Figure 1 shows a concrete example wherein a sequence of 6 tokens is given and its 2^{nd}, 4^{th}, and 6^{th} positions are chosen to be masked. Take the 2^{nd} position as an example, the generator replaces the original token x_{i_1}="*famous*" with another token $\hat{x}_{i_1}$="*old*" and generates the candidate set S_{i_1}={"*top*", "*young*", "*french*", "*famous*"} which includes the original token x_{i_1} and $K = 3$ non-original alternatives.

We may view the multi-word selection task as a simplification of masked language modeling and a generalization of replaced token detection. Asking the model to select the correct word from a candidate set rather than from the entire vocabulary, we can save more computes. At the same time, being essentially a $(K + 1)-$way classification problem, the multi-word selection task is more challenging than the replaced token detection task (which is a binary classification problem) and thus pushes the model to learn more semantic representations. We describe how to generate the candidate set and present our entire method below.

3.2 Multi-task Learning in TEAMS

In TEAMS, we jointly train two transformer encoders, one as the generator network G and the other as the discriminator network D. Given a masked sequence $\mathbf{x}^M$, we use the generator G to perform two tasks for each masked position i_j in this sequence. First, similar to ELECTRA, we sample one token $\hat{x}_{i_j} \sim P_G(\cdot|\mathbf{x}^M)$ (c.f. Eq. (2)) and

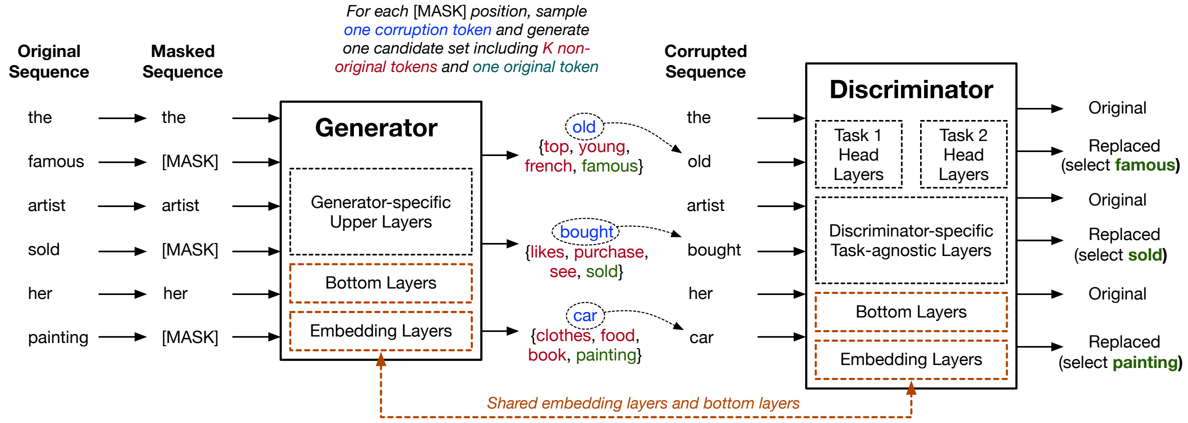

Figure 1: The overview framework of TEAMS. For each masked position, the generator replaces its original token with a new one and outputs a candidate set consisting of the original token and another K possible alternatives. The discriminator inputs the corrupted sentence and learns to (1) predict for every token whether it is replaced or not and (2) select the original token from the candidate set for each masked position.

obtain the corrupted sequence $\mathbf{x}^R$. Second, we draw K non-original tokens $\{x^1_{i_j}, \ldots, x^K_{i_j}\}$ from $P_G(\cdot|\mathbf{x}^M)$ without replacement[2] and construct the candidate set $S_{i_j} = \{x_{i_j}, x^1_{i_j}, \ldots, x^K_{i_j}\}$. Finally, we learn the generator G using the standard masked language modeling task (c.f. Eq. (1)).

On the discriminator side, we train the discriminator network D using two tasks — replaced token detection (RTD) task and multi-word selection (MWS) task. Given a corrupted sentence $\mathbf{x}^R$ of length n, the discriminator will generate two sets of contextualized representations $\{h^{\text{RTD}}_D(\mathbf{x}^R)_i\}|^n_{i=1}$ and $\{h^{\text{MWS}}_D(\mathbf{x}^R)_i\}|^n_{i=1}$, one for each pre-training task. For each position $i \in \{1, \ldots, n\}$, we use $h^{\text{RTD}}_D(\mathbf{x}^R)_i$ to calculate the probability that the token $\mathbf{x}^R_i$ is replaced as follows:

$$P_D(\mathbf{x}^R_i = \mathbf{x}_i) = \text{sigmoid}(w^T h^{\text{RTD}}_D(\mathbf{x}^R)_i), \qquad (6)$$

and optimize the same RTD loss defined in Eq. (4). For each masked position $i_j, j \in \{1, \ldots, m\}$, we obtain the candidate set S_{i_j} from generator outputs and use $h^{\text{MWS}}_D(\mathbf{x}^R)_{i_j}$ to compute the probability of selecting the correct original token x_{i_j} from this candidate set as follows:

$$P_D(x_{i_j}|\mathbf{x}^R, S_{i_j}) = \frac{\exp(e(x_{i_j})^T h^{\text{MWS}}_D(\mathbf{x}^R)_{i_j})}{\sum_{x'_{i_j} \in S_{i_j}} \exp(e(x'_{i_j})^T h^{\text{MWS}}_D(\mathbf{x}^R)_{i_j})}. \qquad (7)$$

As the multi-word selection task is a multi-class classification problem, we define its loss function as follows:

$$\mathcal{L}_{\text{MWS}}(\mathbf{x}, \mathbf{x}^R; D, \mathbb{S}) = \mathbb{E}\left(\sum_{j=1}^{m} -\log P_D(x_{i_j}|\mathbf{x}^R, S_{i_j})\right), \qquad (8)$$

where $\mathbb{S} = \{S_{i_j}\}|^m_{j=1}$ is the collection of candidate sets at all masked positions. Finally, we learn TEAMS by optimizing a combined loss as follows:

$$\min_{G,D} \Big(\mathcal{L}_{\text{MLM}}(\mathbf{x}; G) + \lambda_1 \mathcal{L}_{\text{RTD}}(\mathbf{x}, \mathbf{x}^R; D) + \lambda_2 \mathcal{L}_{\text{MWS}}(\mathbf{x}, \mathbf{x}^R; D, \mathbb{S})\Big), \qquad (9)$$

where λ_1 and λ_2 are two loss balancing hyper-parameters. For the example sequence in Figure 1, the discriminator needs to predict the tokens in 1^{st}, 3^{rd}, 5^{th} positions are not replaced, the tokens in 2^{nd}, 4^{th}, 6^{th} positions are replaced, and select tokens "*famous*", "*sold*", and "*painting*" in 2^{nd}, 4^{th}, 6^{th} positions, respectively.

After pre-training, we keep the discriminator network and fine-tune it for downstream applications[3].

Attention-based Task-specific Heads. One remaining question is how to generate two sets of *task-specific* representations on the discriminator side. Previous studies (Liu et al., 2019a; Sun et al., 2020; Aroca-Ouellette and Rudzicz, 2020) achieve this goal by adding task-specific layers on top of each individual token, as shown in Figure 2 (Left). However, this approach does not model token dependencies within the task-specific layers.

In this work, we propose to use attention-based task-specific heads to capture global dependen-

---

[2] More discussions on other possible negative sampling strategies are presented in experiment section.

[3] Empirically, we find that using the contextualized representations for the MWS task (i.e., $\{h^{\text{MWS}}_D(\mathbf{x}^R)_i\}|^n_{i=1}$) can achieve better fine-tuning performance.

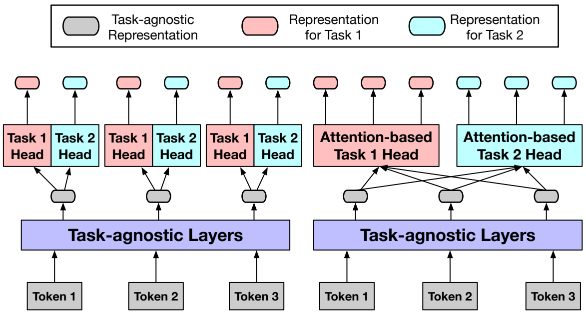

Figure 2: Architectures for transforming task-agnostic representations to task-specific representations. (Left) Adding task-specific heads on each token separately. (Right) Using task-specific attention heads capture all token information holistically.

cies in sequences. Particularly, we design this attention head to be one transformer layer (i.e., a self-attention block followed by a fully connected network with one hidden layer). Since our discriminator also uses a transformer model to obtain each token's task-agnostic representation, we can merge one task head into the discriminator backbone. From this perspective, we can generate different sets of *task-specific* representations as follows. First, we input the sequence to a transformer with L layers and retrieve the final layer output representations for one task. Then, we feed the output of an intermediate layer (*e.g.*, the $(L-1)^{th}$ layer) into another transformer layer to obtain token representations for the second task.

Partial Layer Sharing. ELECTRA has shown that tying the embedding layers of the generator and the discriminator can help improve the pre-training effectiveness. Our study confirms this observation and finds that sharing some transformer layers of the generator and discriminator and can further boost the model performance. More specifically, we design the generator to have the same "width" (i.e., hidden size, intermediate size and number of heads) as the discriminator and share the bottom half of all transformer layers between the generator and the discriminator.

4 Experiments

4.1 Experiment Setups

Pre-training Datasets. We use two datasets for model pre-training: (1) **WikiBooks**, which consists 3.3 Billion tokens from English Wikipedia and BooksCorpus (Zhu et al., 2015). This is the same dataset used in BERT (Devlin et al., 2019). (2) **WikiBooks++**, which extends WikiBooks dataset to 33 Billion tokens by including data from Gigaword (Parker et al., 2011), ClueWeb (Callan et al., 2009), and CommonCrawl (Crawl, 2019). The same dataset is used in XLNet (Yang et al., 2019) and ELECTRA (Clark et al., 2020a).

Evaluation Datasets and Metrics. We evaluate all pre-trained models on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019b) and Stanford Question Answering (SQuAD) dataset (Rajpurkar et al., 2016). GLUE benchmark includes various tasks formatted as either single sentence classification (SST, CoLA) or sentence pair classification (*e.g.*, RTE, MNLI, QNLI, MRPC, QQP, STS). More details of each task are available in the Appendix Section A. SQuAD dataset requires models to select a text span from a given passage that answers a question. In SQuAD v1.1, the answers can always be located in the passage, while SQuAD v2.0 contains some questions unanswerable by the given passage.

We compute Spearman correlation for STS, Matthews correlation for CoLA, accuracy for all other GLUE tasks, and report the GLUE score as the average of all 8 tasks. For SQuAD, we use the standard evaluation metrics of Exact Match (EM) and F1 scores. Since different random seeds may significantly affect fine-tuned model performances, we report the median of 15 fine-tuning runs from the same pre-trained model checkpoint for each result. Unless stated otherwise, results are on the GLUE and SQuAD development sets.

Model Hyper-parameters. We follow and evaluate TEAMS with different model sizes. For *small-sized* model, we set model hidden dimension to 256 and reduce token embedding dimension to 128. The transformer in the discriminator network has 12 layers and each layer consists of 4 attention heads with the intermediate layer size 1024. For *base-sized* model, we adopt the commonly used BERT-base configuration with 768 hidden dimension, 12 layers with 12 attention heads, and 3072 intermediate layer size. For *large-sized* model, we use BERT-large configuration with 1024 hidden dimension, 24 layers with 16 attention heads, and 4096 intermediate layer size. Following (Clark et al., 2020a), we design the generator network size to be 1/2 of the discriminator network size for models of all sizes. For TEAMS, we set the loss balancing parameter $\lambda_1 = 5$, $\lambda_2 = 2$ (c.f., Eq. (9)), and the number of sampled non-original tokens $K = 5$ (c.f., Section 3.1).

Method	Params	GLUE	SQuAD 1.1		SQuAD 2.0	
			EM	F1	EM	F1
BERT-Small	14M	78.52	76.30	84.39	68.95	71.79
ELECTRA-Small (Our reimplementation)	14M	80.36	76.50	84.67	69.17	71.68
TEAMS-Small	14M	**80.70**	**78.84**	**86.40**	**72.33**	**75.24**
BERT-Small++	14M	79.10	76.48	84.75	68.37	71.01
ELECTRA-Small++ (Our reimplementation)	14M	81.71	77.45	85.32	70.07	72.91
ELECTRA-Small++ (Public checkpoint re-evaluate)	14M	81.24	77.62	85.63	71.12	73.95
TEAMS-Small++	14M	**81.99**	**78.94**	**86.65**	**72.11**	**75.11**
BERT-Base	110M	83.46	80.62	88.16	73.26	76.17
ELECTRA-Base (Our reimplementation)	110M	84.63	83.87	90.64	78.59	81.52
TEAMS-Base	110M	**85.57**	**85.21**	**91.69**	**81.59**	**84.51**
BERT-Base++	110M	84.26	84.48	91.08	78.83	81.72
ELECTRA-Base++ (Our reimplementation)	110M	86.29	85.09	91.65	81.31	84.04
ELECTRA-Base++ (Public checkpoint re-evaluate)	110M	87.13	85.09	91.68	79.16	82.06
TEAMS-Base++	110M	**87.16**	**86.05**	**92.48**	**82.73**	**85.59**
BERT-Large	335M	84.91	86.35	92.61	82.19	84.78
ELECTRA-Large (Our reimplementation)	335M	89.20	88.79	94.50	86.02	88.72
ELECTRA-Large (Public checkpoint re-evaluate)	335M	89.38	88.76	94.49	86.79	89.56
TEAMS-Large	335M	**89.44**	**88.86**	**94.61**	**87.08**	**89.86**

Table 1: Comparison results of TEAMS and baseline methods on GLUE and SQuAD datasets. All results are the medians of 15 fine-tuning runs with different initial random seeds. As ELECTRA original paper only releases the public checkpoints for Small++, Base++, and Large models, we can only report results for these three variants.

During pre-training, we set the batch size to be 256 and the input sequence length to be 512 for both small-sized and base-sized models. We update small-sized models for 500K steps and base-sized models for 1M steps on the WikiBooks dataset. Moreover, we test the performance of each model when it is pre-trained for longer time with larger batch size using the WikiBooks++ dataset. We use the suffix "*small*++" to denote a small-sized model pre-trained for 2M steps with batch size 256, and the suffix "*base*++" to denote a base-sized model pre-trained for 1M steps with batch size 1024. Finally, for large-sized models, we use batch size 2048 and pre-train the model for 1.76M steps. All large-sized models and models with suffix "++" are trained using the WikiBooks++ dataset. More pre-training and fine-tuning details are included in the Appendix Section B and C.

Model Implementations. For fair comparison, we implement all compared methods in TensorFlow 2 and evaluate their performances using the official pipeline in TensorFlow Model Garden[4]. In addition to our own implementations, we also report the performance of ELECTRA publicly released checkpoints[5]. All models are trained on TPU v3.

4.2 Experiment Results

We validate the advantages of our proposed TEAMS method by comparing it with BERT (Devlin et al., 2019) and ELECTRA (Clark et al., 2020a). Table 1 shows the comparison results on GLUE and SQuAD datasets. We find that TEAMS can consistently outperform baseline models of the same size. For example, compared to ELECTRA-Base, our TEAMS-Base improves SQuAD 2.0 performance from 78.59 to 81.59 and from 81.52 to 84.51 in terms of EM and F1 score, respectively.

To further verify the performance improvements do not come from consuming more computations, we draw the learning curves of TEAMS-Small/Base and ELECTRA-Small/Base in Figure 3. We observe that for both small-sized and base-sized models, our method can consistently outperform ELECTRA when trained for the some amount of time. Moreover, on SQuAD datasets, TEAMS-Base can even outperform the ELECTRA-Base++ model that requires much more computation.

4.3 Ablation Studies

We continue to evaluate the design of each component within TEAMS and test its sensitivity to some critical hyper-parameters.

Effectiveness of Pre-training Tasks. We report the results of small-sized models learned using dif-

---

[4] `https://github.com/tensorflow/models`.

[5] `https://github.com/google-research/electra`.

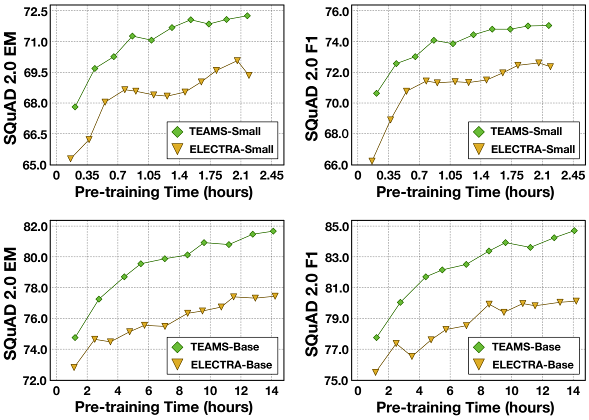

Figure 3: Learning curves for small-sized and base-sized models.

Pre-training Task(s)	**GLUE**	**SQuAD 1.1**		**SQuAD 2.0**	
		EM	**F1**	**EM**	**F1**
Only MLM (i.e., BERT)	79.10	76.48	84.75	68.37	71.01
Only RTD (i.e., ELECTRA)	81.71	77.45	85.32	70.07	72.91
Only MWS	79.65	77.30	85.32	70.10	72.80
RTD + MWS (i.e., TEAMS)	**81.99**	**78.94**	**86.65**	**72.11**	**75.11**

Table 2: Effectiveness of multi-task pre-training for *small*++ models. “MLM”, “RTD”, and “MWS” stand for “masked language modeling”, “replaced token detection”, and “multi-word selection”, respectively.

ferent pre-training tasks in Table 2. First, we can see that the model trained with multi-word selection (MWS) task can outperform the one learned using masked language modeling (MLM) task. Second, on SQuAD datasets, we find that pre-training on only 15% of masked tokens using MWS task is comparable with pre-training on all tokens using replaced token detection (RTD) task. These observations demonstrate the effectiveness of our proposed MWS task. Finally, we show that a text encoder pre-trained using both MWS and RTD tasks can outperform those learned using only single task.

Task-specific Layer Designs. In TEAMS, we pre-train the discriminator network using multi-task learning and introduce the attention-based task-specific heads. To verify the effectiveness of these attention-based task-specific heads, we train another model that uses the traditional feed forward network (FFN) as the task-specific head. Table 3 shows the results. We can see that our model achieves better performances because the attention-based heads can effectively model the token dependencies in sequences.

We continue to study where to add these task-specific heads. Currently, given a transformer with 12 layers, we treat its last layer output for one task and feed the 11^{th} layer output to a separate transformer layer to obtain representations for the second task[6]. An alternative design is to add two separate transformer layers (as two task-specific heads) directly on top of the last layer (i.e., the 12^{th} layer). As shown in Table 3, we find the latter design can slightly improve the model performance on SQuAD datasets but leads to a larger discriminator network with effectively 13 transformer layers and thus requires more computation during both pre-training and fine-tuning stages.

Method	**GLUE**	**SQuAD 1.1**		**SQuAD 2.0**	
		EM	**F1**	**EM**	**F1**
ELECTRA-Small++	81.71	77.45	85.32	70.07	72.91
TEAMS-Small++	81.99	78.94	86.65	72.11	75.11
Use FFN task heads	81.49	78.18	86.35	72.00	74.90
Add task head on 12^{th} layer	81.29	79.08	86.66	72.47	75.31
Use RTD task head outputs	81.83	77.72	85.80	69.56	72.57

Table 3: Analysis of task-specific layers and exported representations for *small*++ models. Please refer to Section 4.3 for detailed descriptions of each method.

Finally, as our discriminator network will output two sets of contextualized representations, one for MWS task and the other for RTD task, we need to decide which set of representations to use in the fine-tuning stage. Empirically, we find the representations for MWS task has better fine-tuning performance than the ones for RTD task, especially on the SQuAD datasets (c.f. Table 3). This observation also confirms the effectiveness of our proposed MWS task as it produces representations capturing more fine-grained semantic information compared to the RTD task.

Partial Layer Sharing. Table 4 reports the results of our models with different levels of parameter sharing between the generator and the discriminator. First, we can see that tying all generator layers with discriminator layers results in significant performance drops, as such a binding restricts the model representation power. Second, we find that compared to no weight sharing, our design of partial layer tying can improve the model performance. One possible explanation is that such layer tying serves as an implicit form of regularization and forces the shared transformer layers to capture useful information for both generator pre-training task (i.e., MLM) and discriminator pre-training tasks (i.e., RTD and MWS).

---

[6] We can interpret the first 11 layers as *task-agnostic* layers and view the 12^{th} layer and the newly introduced separate layers as two *task-specific* heads.

Method	GLUE	SQuAD 1.1		SQuAD 2.0	
		EM	F1	EM	F1
ELECTRA-Small++	81.71	77.45	85.32	70.07	72.91
TEAMS-Small++	**81.99**	**78.94**	**86.65**	72.11	75.11
Full Tie	80.57	77.75	85.76	69.93	72.82
No Tie	81.65	78.42	86.32	**72.73**	**75.73**
ELECTRA-Base++	86.29	85.09	91.65	81.31	84.04
TEAMS-Base++	**87.16**	**86.05**	**92.48**	**82.73**	**85.59**
No Tie	86.63	85.51	91.98	80.72	83.60

Table 4: Effect of sharing generator and discriminator bottom layers for *small*++ and *base*++ models. "Full Tie" and "No Tie" stand for tying all or none of generator layers with the discriminator, respectively.

Sampling Strategy and Negative Sample Size. To use the multi-world selection task for pre-training, we need to first obtain a set of negative samples (i.e., non-original tokens) for each masked position in a sequence. In this study, we test two strategies to generate K negative samples for each masked position. Given the generator output probability distribution for a target position, we can either sample from this distribution K times without replacement or directly select K non-original tokens with the highest probabilities. We denote these two approaches as "Sampled" and "Hardest", respectively, and report the results in Figure 5. First, we can see that performing repeated sampling is a better strategy than always selecting those hardest samples. One possible reason is that the "Sampled" strategy can generate more diverse negative samples and thus helps model to generalize[7]. Second, we notice that increasing K over 5 will somewhat hurt the model performance. One reason is that a larger K causes a higher probability of including false negative examples. Finally, we find that for a wide range of K from 3 to 50, our method can outperform ELECTRA, which further demonstrates the effectiveness of multi-word selection task.

Generator Size. We test how the size of generator affects the model performance by varying the number of transformer layers in the generator. For all tested models, we tie the bottom half of generator with the discriminator. Figure 4 reports the results. We find that the performance first increases as the generator size increases until it reaches about half of the discriminator size and then starts to decrease when we further increase the generator size. The same results also hold for base-sized models.

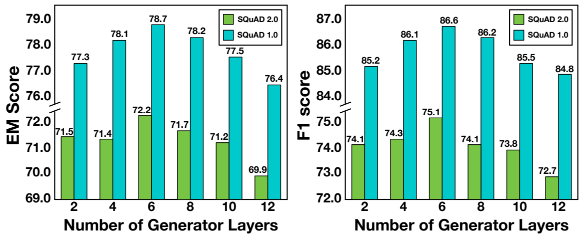

Figure 4: Analysis of the number of generator layers in TEAMS-small++ models.

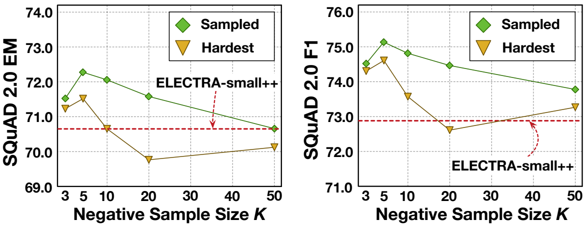

Figure 5: Analysis of negative sampling strategies and negative sample sizes for *small*++ models. Based on the generator output distribution, we either perform weighted sampling without replacement K times ("Sampled") or select K most likely non-original tokens ("Hardest") for each masked position.

5 Related Work

Besides the general language pre-training work we discussed in Section 2.1, this study is particularly related to methods that apply multi-task learning (Caruana, 1997; Ruder, 2017; Shen et al., 2018) to language representation learning. An early study (Liu et al., 2019a) proposes to simultaneously fine-tune a pre-trained BERT model to perform multiple natural language understanding tasks and achieves promising results on the GLUE dataset. Sun et al. (2020) continue this line of work and propose to push the multi-task learning to the model pre-training stage. Specifically, they use a continual multi-task learning framework that incrementally builds and inserts seven auxiliary tasks (*e.g.*, masked entity prediction, sentence distance prediction, *etc.*.) to the text encoder. More recently, Aroca-Ouellette and Rudzicz (2020) extend this idea to incorporate fourteen auxiliary tasks and identify six tasks are particularly useful. While achieving inspiring performance, these studies all assume the MLM pre-training task must present and just combine MLM with additional tasks. In this paper, we relax this assumption and combine our new multi-word selection task with the replace token detection task for effective pre-training.

---

[7] A similar result is also witnessed in (Shen et al., 2019) and thus we adopt the "Sampled" approach in this study.

6 Conclusions and Future Work

This work presents a new text encoder pre-training method that simultaneously learns a generator and a discriminator using multi-task learning. We propose a new pre-training task, multi-word selection, and combine it with previous pre-training tasks for efficient encoder pre-training. We also develop two techniques, attention-based task-specific heads and partial layer sharing, to further improve pre-training effectiveness. Extensive experiments on GLUE and SQuAD datasets demonstrate our TEAMS method can consistently outperform previous state-of-the-arts methods. In the future, we plan to explore how other auxiliary pre-training tasks can be integrated into our framework. Moreover, we consider extending our pre-training method to text encoders with other architectures such as those based on dynamic convolution and sparse attention. Finally, being orthogonal to this study, distillation techniques could be applied to further compress our pre-trained encoders into smaller models for faster inference speeds.

Acknowledgement

We thank Hongkun Yu, You (Will) Wu from Google Research, Xiaotao Gu, Yu Meng from University of Illinois at Urbana-Champaign, and Richard Pang from New York University for providing valuable comments and discussions. Also, we would like to thank anonymous reviewers for valuable feedback.

Broader Impact Statement

Recent years have witnessed the great success of pre-trained text encoders in lots of NLP applications such as text classification, question answering, text retrieval, dialogue system, etc. This paper presents a new pre-training method TEAMS that learns a text encoder with better performance using lower training cost. Therefore, on the positive side, our work has the potentials to benefit all downstream applications that leverage a pre-trained text encoder, especially those applications with limited computation resources. On the negative side, TEAMS, as one specific pre-training method, could still face the generic issues for all language pre-training work. For example, the pre-training large corpora, collected from the internet, may include abusive language usages and fail to capture the cultures that have smaller linguistic footprints online.

References

Stephane T Aroca-Ouellette and F. Rudzicz. 2020. On losses for modern language models. In *EMNLP*.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020a. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. Pre-training transformers as energy-based cloze models. In *EMNLP*.

Common Crawl. 2019. Common crawl. *URl: http://http://commoncrawl. org*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B. Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL@ACL*.

Linyuan Gong, D. He, Zhuohan Li, T. Qin, Liwei Wang, and T. Liu. 2019. Efficient training of bert by progressively stacking. In *ICML*.

X. Gu, Liyuan Liu, H. Yu, Jing Li, Chen Chen, and J. Han. 2020. On the transformer growth for progressive bert training. *ArXiv*, abs/2010.12562.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First Quora dataset release: Question pairs.

Zihang Jiang, Weihao Yu, Daquan Zhou, Y. Chen, Jiashi Feng, and S. Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. In *NeurIPS*.

Mandar Joshi, Danqi Chen, Y. Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

H. Levesque, E. Davis, and L. Morgenstern. 2011. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Xiaodong Liu, Pengcheng He, W. Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *ACL*.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *EMNLP/IJCNLP*.

R Parker, D Graff, J Kong, K Chen, and K Maeda. 2011. English gigaword fifth edition ldc2011t07 (tech. rep.). Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *ArXiv*, abs/1706.05098.

J. Shen, Maryam Karimzadehgan, Michael Bendersky, Zhen Qin, and Donald Metzler. 2018. Multi-task learning for email search ranking with auxiliary query clustering. In *CIKM*.

J. Shen, Ruiliang Lyu, Xiang Ren, M. Vanni, Brian M. Sadler, and Jiawei Han. 2019. Mining entity synonyms with efficient neural set generation. In *AAAI*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Y. Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, H. Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Wei Wang, B. Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and L. Si. 2020. Structbert: Incorporating language structures into pre-training for deep language understanding. In *ICLR*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*.

Zhenhui Xu, Linyuan Gong, Guolin Ke, Di He, Shu xin Zheng, Liwei Wang, Jiang Bian, and T. Liu. 2020. Mc-bert: Efficient language pre-training via a meta controller. *ArXiv*, abs/2006.05744.

Cheng Yang, Shengnan Wang, Chao Yang, Yuechuan Li, Ru He, and Jingqiao Zhang. 2020. Progressively stacking 2.0: A multi-stage layerwise training method for bert training speedup. *ArXiv*, abs/2011.13635.

Z. Yang, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Y. Zhu, Ryan Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

A GLUE Details

The original GLUE benchmark (Wang et al., 2019b) contains 9 natural language understanding datasets. We describe them below:

- **MNLI:** The Multi-genre Natural Language Inference Corpus (Williams et al., 2018) contains 393K training sentence pairs with textual entailment annotations. Given a premise sentence and a hypothesis sentence, a model needs to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither.

- **RTE:** Recognizing Textual Entailment (Giampiccolo et al., 2007) dataset is similar to MNLI and contains 2.5K sentence pairs with binary entailment annotations (entailment or contradiction).

- **QNLI:** Question Natural Language Inference dataset is a binary sentence pair classification dataset constructed from SQuAD (Rajpurkar et al., 2016). It contains 108K training sentence pairs and requires a model to predict whether a context sentence contains the answer to a question sentence.

- **CoLA:** Corpus of Linguistic Acceptability (Warstadt et al., 2018). This dataset includes 8.5K training sentences annotated with whether it is a grammatical English sentence.

- **SST:** Stanford Sentiment Treebank (Socher et al., 2013) dataset contains 67K sentences from movie reviews and their corresponding binary sentiment annotations.

- **MRPC:** Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) includes 3.7K sentence pairs from online news sources. The task is to predict whether two sentences are semantically equivalent or not.

- **STS:** Semantic Textual Similarity (Cer et al., 2017) benchmark contains 5.8K training sentence pairs. The task is to predict the similarity score of two sentences from 1 to 5.

- **QQP:** Quora Question Pairs (Iyer et al., 2017) dataset includes 364K question pairs sampled from the community question-answering website Quora. Models are trained to predict whether a pair of questions are semantically equivalent.

- **WNLI:** Winograd NLI (Levesque et al., 2011) is a small natural language inference dataset. However, as GLUE official website[8] indicates there are some issues during its construction process, we follow previous studies (Clark et al., 2020a; Jiang et al., 2020) and exclude this dataset for fair comparisons.

B Pre-training Details

For the pre-training architecture configurations, we mostly use the same hyper-parameters as BERT and ELECTRA. To generate masked positions, we follow BERT and duplicate training data 40 times so each sequence is masked in 40 different ways. We find this static masking strategy performs similar to the dynamic masking strategy in ELECTRA, while being easier to implement and has less computation overhead. Besides, we notice a mask percentage of 15 works well for all models and thus do not increase it to 25 for large-size models as suggested in ELECTRA. We set λ_1 and λ_2, the loss balancing parameters to 5 and 2, respectively, to ensure different loss terms are of the same scale. For small-size and base-size models, we search for the learning rate out of {1e-4, 2e-4, 3e-4, 5e-4}, batch size from {128, 256, 512, 1024}, and training steps from {500K, 1M, 1.5M, 2M}. For large-size models, we search for the learning rate out of {1e-4, 2e-4, 3e-4, 5e-4} and batch size from {1024, 2048}. Also, we select the generator size out of {1/4, 1/3, 1/2} in early experiments. The best configurations are reported in the main text and we perform no other hyper-parameter tuning. The full set of hyper-parameters are listed in Table 5.

C Fine-tuning Details

For fair comparisons, we fine-tune all pre-trained checkpoints using the official pipeline in TensorFlow Model Garden[9] and report the median of 15 fine-tuning runs. We do not include layer-wise learning-rate decay. We search for the learning rate from {1e-5, 3e-5, 5e-5, 8e-5, 1e-4}, batch size from {32, 48}, and training epoch from {2, 3, 5}. For GLUE tasks, best evaluation scores during fine-tuning are reported. For SQuAD, scores at the end of fine-tuning are reported. The full set of hyper-parameters are listed in Table 6.

---

[8] `https://gluebenchmark.com/faq`

[9] `https://github.com/tensorflow/models`

Hyper-parameter	Small/Small++	Base/Base++	Large
Number of Layers	12	12	24
Embedding Dim.	128	768	1024
Hidden Dim.	256	768	1024
Intermediate Layer Dim.	1024	3072	4096
Number of Attention Heads	4	12	16
Attention Head Dim.	64	64	64
Generator Size (Multiplier for Number of Layers)	1/2	1/2	1/2
Mask Percentage	15	15	15
Learning Rate Decay	Linear	Linear	Linear
Warmup Steps	10000	10000	10000
Learning Rate	5e-4	2e-4/3e-4	2e-4
Adam ϵ	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999
Attention Dropout	0.1	0.1	0.1
Dropout	0.1	0.1	0.1
Weight Decay	0.01	0.01	0.01
Batch Size	256/256	256/1024	2048
Train Steps	500K/2M	1M/1M	1.76M

Table 5: Pre-training hyper-parameters.

Hyper-parameter	Value
Learning Rate	1e-4 in Small/Small++, 3e-5 in Base/Base++/Large for GLUE 8e-5 in Small/Small++, 5e-5 in Base/Base++, and 3e-5 in Large for SQuAD 1.1/2.0
Learning Rate decay	Linear
Warmup fraction	0.1
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Dropout	0.1
Batch Size	32 for GLUE, 48 in Small/Small++/Large and 32 in Base/Base++ for SQuAD 1.1/2.0
Training Epochs	5 for GLUE, 5 in Small/Small+ and 2 in Base/Base++/Large for SQuAD 1.1/2.0

Table 6: Fine-tuning hyper-parameters.

REAM♯: An Enhancement Approach to Reference-based Evaluation Metrics for Open-domain Dialog Generation

Jun Gao
Tencent AI Lab, Shenzhen, China
imgaojun@gmail.com

Wei Bi*
Tencent AI Lab, Shenzhen, China
victoriabi@tencent.com

Ruifeng Xu*
Peng Cheng Laboratory, Shenzhen, China
xuruifeng.hitsz@gmail.com

Shuming Shi
Tencent AI Lab, Shenzhen, China
shumingshi@tencent.com

Abstract

The lack of reliable automatic evaluation metrics is a major impediment to the development of open-domain dialogue systems. Various reference-based metrics have been proposed to calculate a score between a predicted response and a small set of references. However, these metrics show unsatisfactory correlations with human judgments. For a reference-based metric, its reliability mainly depends on two factors: its ability to measure the similarity between the predicted response and the reference response, as well as the reliability of the given reference set. Yet, there are few discussions on the latter. Our work attempts to fill this vacancy. We first clarify an assumption on reference-based metrics that, if more high-quality references are added into the reference set, the reliability of the metric will increase. Next, we present **REAM♯**: an enhancement approach to **R**eference-based **E**v**A**luation **M**etrics[1] for open-domain dialogue systems. A prediction model is designed to estimate the reliability of the given reference set. We show how its predicted results can be helpful to augment the reference set, and thus improve the reliability of the metric. Experiments validate both the effectiveness of our prediction model and that the reliability of reference-based metrics improves with the augmented reference sets.

1 Introduction

The lack of reliable automatic evaluation metrics is a major impediment to the development of open-domain dialogue systems (Li and Jurafsky, 2016; Gao et al., 2019; Li et al., 2020a). The underlying difficulty in evaluation lies in the diversity of the possible outcomings. Existing evaluation metrics for open-domain dialogue systems can be roughly divided into reference-based and reference-free metrics. Reference-based metrics usually measure how similar a generated response is to the reference responses. Reference-free metrics, on the other hand, measure the quality of a response without any reference and usually focus on specific aspects of the responses. For example, much work often computes the perplexity of a generated response as a measure of fluency (Li et al., 2020b), and adopts Dist-1/2 (Li et al., 2016b) to measure the diversity of the response. In this work, we focus on reference-based metrics.

BLEU (Papineni et al., 2002), originally for machine translation, is now a popular reference-based metric to evaluate open-domain dialog systems automatically. However, it has been shown that BLEU and other word-overlap metrics such as METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004), rely on surface-form similarities only without considering the semantic diversity, thus fail to correlate well with human judgements (Liu et al., 2016). Instead, embedding-based metrics are adopted to consider the semantic meaning of a word defined by a distributed representation. For example, Zhang et al. (2020) introduce an embedding-based metric BERTScore that computes the similarity between the generated response and reference responses using contextual embeddings obtained from BERT (Devlin et al., 2019).

Intuitively, the reliability of a referenced-based metric depends on two factors: (1) the ability of the metric to measure the similarity between the generated response and the reference response and (2) the reliability of the reference set for evaluating each generated response. As can be seen from above, most current work falls into improving the former factor, while few considers the latter. However, without a high-quality reference set, the results obtained by all these metrics will have a poor correlation with human judgments.

---

*Corresponding authors

[1] Interested reader may contact the authors to obtain a copy of the code and the data.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2487–2500
August 1–6, 2021.

Unlike most previous studies that propose new evaluation metrics, the focus of our study is on improving the reliability of the reference set. We first clarify an assumption on reference-based metrics that, if more high-quality responses are added into the reference set, the correlation of reference-based metrics with human judgments will increase. We perform experiments to demonstrate that the standard BLEU (Papineni et al., 2002) does not hold this assumption, but two existing metrics can. One is a modified BLEU metric (Freitag et al., 2020) that compares the generated response with each reference response within the set using single-reference BLEU. We refer this modified BLEU as BLEU*. The other is the BERTScore (Zhang et al., 2020), which can also be used to evaluate responses with multiple references.

In this work, we propose **REAM♯**: an enhancement approach to **R**eference-based **E**v**A**luation **M**etrics for open-domain dialogue systems. Our approach, which can enhance a reference-based metric that satisfies our assumption (such as BLEU* and BERTScore), consists of two parts: (1) reliability prediction and (2) high-quality references augmentation. In the first part, we devise a reliability prediction model to estimate the reliability of the reference set. Given a query and its reference set, the model will predict a reliability score to reflect how reliable a metric is used to evaluate the results of the query using the given reference set. In the second part, we aim to augment high-quality references with the help of the reliability prediction model. We introduce two ways to handle reference candidates with different qualities. If the acquired reference candidates are considered reliable, we can adopt automatic annotation. If we are not certain about the relevance of the reference candidates, human annotators are needed for an interactive annotation. Experimental results show that our proposed approach can effectively enhance the reliability of the reference set and improve the correlation of the reference-based metrics BLEU* and BERTScore.

The rest of this paper is organized as follows. In Section 2, we introduce our assumption on the reference-based metrics, and conduct a series of preliminary experiments to validate this assumption on existing metrics. We also provides details about data collection and metric evaluation in this section. Section 3 describes our reliability prediction model and Section 4 presents how to augment high-quality references with the help of the proposed reliability prediction model. Section 5 shows our experimental results about the proposed reliability prediction model as well as different strategies to augment the reference set. Section 6 describes related work. Finally, we conclude our work and discuss some future work in Section 7.

2 Research Questions and Settings

We first make an assumption on reference-based metrics for open-domain dialog system that if more high-quality reference responses are added to the reference set, a reference-based metric will show a higher correlation with human judgments. We want to draw such an assumption due to two considerations. First, by considering the nature of open-domain dialog, a query is possible to be relevant to multiple diverse responses. Including more relevant responses in the reference set can naturally help alleviate the assessment difficulty associated with linguistic variation. Second, if a low-quality response, e.g. a general response "I don't know" is added to the reference set, the reference-based metric will assign a very high score for the same low-quality predicted response, resulting in a low correlation. Therefore, only by ensuring that the responses in the reference set are of high-quality can we avoid the metric assigning high scores to low-quality responses.

We validate this assumption on three existing metrics: the original multi-reference BLEU (Papineni et al., 2002), BLEU* (Freitag et al., 2020) and BERTScore (Zhang et al., 2020). The original BLEU uses a modified form of precision to compare a generated response against multiple reference responses. For BLEU* and BERTScore, the multi-reference score of a response y can be computed as:

$$score(y, R) = \max_{r \in R} d(y, r) \tag{1}$$

where d is the given metric, $R = \{r_1, r_2, \cdots, r_n\}$ is the given reference set.

We examine the assumption by answering the following questions:

1. *Will the correlation of the reference-based metric improve when more high-quality responses are added to the reference set?*
2. *How will the low-quality responses included in the reference set affect the correlation of the reference-based metric?*

Raw Dataset	
# Training Samples	4,000
# Validation Samples	500
# Test Samples	500
Each Sample	
# Model Responses	14
# Reference Responses	≤ 200

Table 1: Our dataset statistics. Each sample is in the form (query, model responses, reference responses).

Before presenting the details of the preliminary experiments, we first describe the data collection used in the experiments and how we estimate the reliability of a metric.

2.1 Data Collection and Metric Evaluation

To evaluate the correlation of reference-based metrics with human judgments, we collect a dataset as follows. We first crawled 5,000 queries from some Chinese social websites and the largest reference set for the obtained queries has 200 responses. Then, we generated 14 responses obtained from several widely used response generation models (described in Appendix A). We asked 5 annotators to assign each generated response with a score ranging from 1 to 5 respectively, and the final score of each response was obtained by averaging the five scores. Table 1 shows the data statistics. See Appendix A for a detailed description of our dataset.

To evaluate the performance of a metric, we leverage the Pearson Correlation. Given a sample consisting of a query $\boldsymbol{q}$, 14 generated responses $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{\mathbf{14}}\}$ with their human annotated scores $\hat{\boldsymbol{S}} = \{\hat{\boldsymbol{s}}_1, \cdots, \hat{\boldsymbol{s}}_{14}\}$ and a set of reference responses $\boldsymbol{\mathcal{R}} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_n\}$ ($n \leq 200$), we first score each model response $\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}$ with a certain metric, yielding a sequence of automatic evaluated scores $\boldsymbol{S} = \{s_1, \cdots, s_{14}\}$. With the automatic evaluated scores $\boldsymbol{S}$ and human annotated scores $\hat{\boldsymbol{S}}$ for the 14 generated responses, we can obtain the reliability score c for each sample using the Pearson Correlation:

$$c = \mathrm{Pearson}(\boldsymbol{S}, \hat{\boldsymbol{S}}). \tag{2}$$

2.2 Preliminary Experiments

Next, we conduct our preliminary experiments using the 500 samples in the test set and present our empirical observations regarding the two questions in beginning of Section 2. In the experiments in this section, the correlation of a certain metric is obtained by averaging the reliability scores over all 500 samples.

First, we would like to see how it affects the correlation of the metrics when high-quality responses are continuously added to the reference set. To ensure that each added reference response is of high quality, we also have human annotators assign a quality score to each crawled reference response, and then randomly select 10 high-quality reference responses (quality score over 4) for each sample in the test set. We sequentially add the 10 high-quality reference responses to the reference set initialized with an empty set. Each time a new reference response is added to the set, we calculate a reliability score of a certain metric using the updated set. Figure 1 shows the evaluation results of the original BLEU, BLEU* and BERTScore. Noticeably, the correlation of the original BLEU does not improve as the number of high-quality sentences increases. The reason may be that the original BLEU is defined at the corpus-level and the n-gram precisions are sums over all corpus sentences, which means the newly added reference will cause the value to fluctuate. We, however, find that BLEU* and BERTScore are consistent with our assumption that the correlation of the metric improves as the number of high-quality responses in the set increases. Therefore, not all metrics meet our assumption, and we use BLEU* and the BERTScore for our following experiments.

Second, we test how low-quality responses would affect the correlation of the metric by adding noise into the reference set. In the noisy reference set, the first 5 responses in the reference set are added with the same high-quality responses as above, while the later 5 added responses are replaced with negative samples which are responses sampled from other queries. As shown in Figure 1, "BLEU*-noisy" and "BERTScore-noisy" denote the results of BLEU* and BERTScore using the noisy reference sets, respectively. As the number of negative samples in the set increases, the performance of the metrics starts to degrade. This further confirms our assumption that only adding more high-quality responses to the reference set will help the metric.

Based on the above analysis, we can see that for reference-based metrics satisfying our assumption, we can enhance its reliability by augmenting high-quality references to calculate the metric. Next,

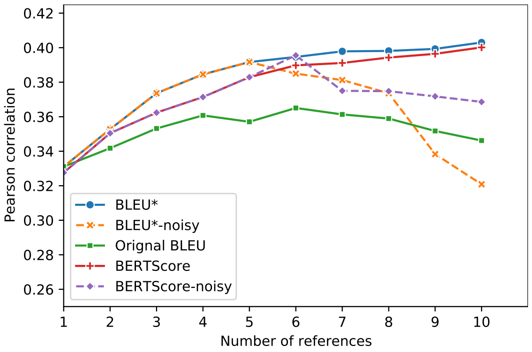

Figure 1: Pearson correlations of BLEU, BLEU* and BERTScore with human judgements obtained using different numbers of reference responses.

a key question arises, how can we augment high-quality references? In the following, we propose a reliability prediction model to estimate the reliability of the reference set (Section 3) and provide effective approaches to augment high-quality references with the help of the reliability prediction model (Section 4).

3 The Reliability Prediction Model

In this section, we aim to estimate the reliability of a metric, when we use it to compute a performance score of an output response based on a given gold response set. We formulate this task as a regression problem. Formally, given a query sentence $\boldsymbol{q} = \{q_1, \dots, q_M\}$ of length M, and a reference set $\mathcal{R} = \{\boldsymbol{r}_i\}_{i=1}^{N}$ with N gold responses of the query, our goal is to learn a function $\boldsymbol{f} : (\boldsymbol{q}, \mathcal{R}) \rightarrow c$ that predicts a reliability score c that represents how reliable a metric is used to evaluate results of the input query using this reference set. In this work, we consider using Pearson correlation (c) in Eq. 2 between the human evaluation results and the metric scores from the given $\mathcal{R}$ as the reliability score of each $(\boldsymbol{q}, \mathcal{R})$. We introduce the reliability prediction model in this section, and discuss efficient methods to augment high-quality responses based on the trained prediction model in the next section.

Our learning framework is shown in Figure 2, which contains two parts. We first design a prediction model to predict a reliability score for each $(\boldsymbol{q}, \mathcal{R})$. Then, we construct negative samples and use contrastive learning to train the proposed prediction model.

3.1 Model Structure

Input Representation For a query $\boldsymbol{q}$ and each response $\boldsymbol{r}$ in the reference response set $\mathcal{R}$, we first concatenate them into one sequence. Then we can obtain N query-response pairs. We leverage BERT (Devlin et al., 2019) that learns contextualized representations of the query-response pairs:

$$[\boldsymbol{v}_c, \boldsymbol{v}_{q_1}, \cdots, \boldsymbol{v}_{r_1}, \cdots, \boldsymbol{v}_{r_m}] = \text{BERT}([\boldsymbol{q}, \boldsymbol{r}]) \quad (3)$$

where $\boldsymbol{v}_c$ is the representation for the special token [CLS] in the BERT and we use its contextualized representation for each query-response pair. Then, we get N query-response representations.

Graph Encoder: In order to transform the dialogue context features into higher-level features that consider the intrinsic variance between reference responses in the reference set, we represent all query-response pairs in a fully-connected graph and each query-response pair is a node in the graph. Then we adopt a graph attention layer (Velickovic et al., 2018) to obtain a better representation of each query-response pair:

$$[\hat{\boldsymbol{v}}_{c_1}, \dots, \hat{\boldsymbol{v}}_{c_n}] = \text{GAT}([\boldsymbol{v}_{c_1}, \dots, \boldsymbol{v}_{c_n}]). \quad (4)$$

The final representation $\boldsymbol{v}_g$ of the graph is computed using the max-pooling strategy.

Reliability Score Prediction: The reliability score is computed using a single-layer feedforward network coupled with a tanh activation function:

$$f(\boldsymbol{q}, \mathcal{R}) = \tanh(\mathbf{W} \cdot \boldsymbol{v}_g + \mathbf{b}) \quad (5)$$

where $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters. The model is trained to minimize the squared error between the model prediction and the gold correlation coefficient c with L2-regularization:

$$\mathcal{L}_r = (f(\boldsymbol{q}, \mathcal{R}) - c)^2 + \gamma \|\theta\|_2 \quad (6)$$

where γ is a hyper-parameter and θ denotes the parameters of the model.

3.2 Data Augmentation

To address the problem of limited data, we adopt the data augmentation method to improve the generalization of our models. Our input sample consists of a query $\boldsymbol{q}$ and a set of reference responses $\mathcal{R}$. We can expand the training data by using different combinations of reference responses, generating more different reference sets for each query. Since the number of all combinations is huge, we randomly sample some various combinations for each query. Given a set of reference responses $\boldsymbol{\mathcal{R}} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_n\}$ of n elements ($n \leq 200$), the set of k-combinations are denoted as $\boldsymbol{\mathcal{C}}_n^k$, where

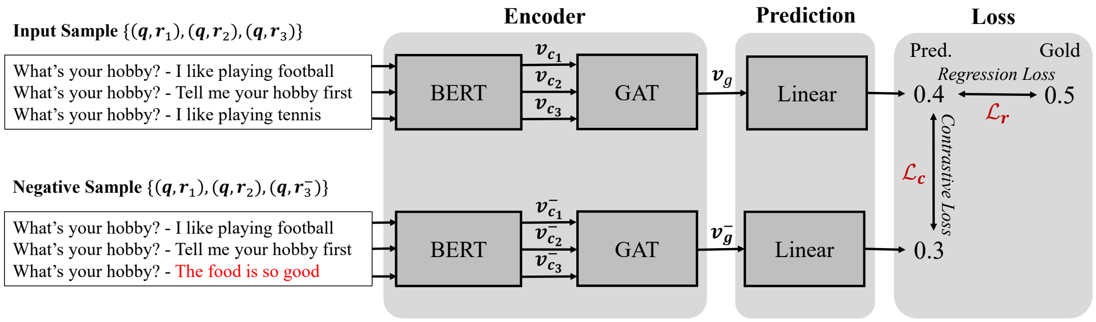

Figure 2: Our proposed prediction model. For each training sample $(\boldsymbol{q}, \{\boldsymbol{r}\})$, we construct negative samples $(\boldsymbol{q}, \{\boldsymbol{r}\}^-)$ according to our assumption and compute the contrastive loss. The model is trained with the combination of regression loss and contrastive loss.

each combination has k different elements. In our work, we use $k \in \{3, 5, 7, 10\}$ and randomly sample 25 different combinations from each set $\mathcal{C}_n^k$ ($k \in \{3, 5, 7, 10\}$). We then expand the validation set in the same way.

To evaluate the generalization performance of the model, the test set should be used not only to verify the effectiveness of the model on samples with $k \leq 10$, but also to test the performance of the model on samples with $k > 10$. Therefore, we use $k \in \{3, 5, 10, 20, 30, 40\}$ to augment the test set. For each set $\mathcal{C}_n^k$ ($k \in \{3, 5, 10, 20, 30, 40\}$), we only randomly sample one combination. Finally, applying data augmentation as described above gives us a total of 400,000/4,000/3,000 augmented training/validation/test samples.

3.3 Contrastive Learning

Optimizing the simple regression objective above with the augmented training samples may not yield a robust performance. To make the model capable to capture the differences between different reference set for the same query, we construct multiple negative samples for each training sample $(\boldsymbol{q}, \{\boldsymbol{r}\})$, and use contrastive learning to train the prediction model. Given a training sample $(\boldsymbol{q}, \{\boldsymbol{r}\})$ we design three kinds of negative samples $(\boldsymbol{q}, \{\boldsymbol{r}\}^-)$:

- Remove one response r from the existing reference set $\{\boldsymbol{r}\}$. This is based on the understanding in Section 2 that a set of gold responses generally does not yield a higher correlation than a super set of it. We note that all samples with deteriorated correlation here, are treated as negative samples.
- Randomly select a response of any other query to add to the existing set. This is based on the understanding in Section 2 that a noise response included in a gold reference set should deteriorate the correlation.
- Randomly select a response of any other query to replace a response in the existing set. The intuition is the same as the above one.

The contrastive loss function $\mathcal{L}_c$ with T negative samples constructed is computed as:

$$\mathcal{L}_c = \frac{1}{T}\sum_t \max\{0, \Delta - f(\boldsymbol{q}, \{\boldsymbol{r}\}) + f(\boldsymbol{q}, \{\boldsymbol{r}\}_t^-)\}, \tag{7}$$

where Δ is a margin. The final loss can be computed as:

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c. \tag{8}$$

4 Augmenting High-quality References

In this section, we describe how we can augment high-quality references based on the current gold reference set for a given query with the help of the proposed reliability prediction model. Suppose a large set of reference candidates can be easily obtained. For example, more conversation data can be assessed to build a retrieval system to search for more references for the given query. We introduce two ways to handle reference candidates with different qualities. If the acquired reference candidates are considered reliable, we can adopt automatic annotation. If we are not certain about the relevance of the reference candidates, human annotators are needed for an interactive annotation.

4.1 Automatic Annotation

We assume the response candidates are all relevant to the given query. However, there may exist uninformative responses, such as the generic responses or those similar to the gold references. We now introduce how to use the predicted scores to automatically select out high-quality responses from the candidate set. Each time a response is randomly picked to tentatively add into the gold reference set, the model will predict a reliability score for the augmented reference set. If the reliability score of the augmented reference set improves, this picked response can be considered as a high-quality one and now confirmed to add to the gold reference set. Otherwise, we remove it from the gold reference set. We continue this process until all response candidates have been picked once, and return the final augmented gold reference set.

4.2 Interactive Annotation

If the response candidates are of uneven relevance to the query, we need to hire annotators to manually check and edit the response candidates. We design an interactive annotation strategy to allow the model to assist annotators. For each selected response candidate, the annotator mainly executes the following three steps:

1. If he/she considers the response candidate is relevant to the given query and the reliability score of the set improves with the current candidate response, he/she can retain this response directly;
2. If he/she considers the response candidate is not relevant enough to the given query but the reliability score of the set shows improvement with the current candidate response, he/she needs to edit this response and check whether the reliability score of the set increases with the edited response;
3. If the reliability score of the set does not improve with the current candidate response, no matter whether the candidate response is relevant or not, he/she still needs to edit this response and check whether the reliability score of the set increases with the edited response;

In both Step2 and Step3, we allow the interactive annotation with a maximum number of attempts. Otherwise, we abandon this response and continue to the next response candidate. This process can help annotators avoid writing responses with unsatisfactory quality.

5 Experiments

5.1 Setup

In our experiments, we use the quality scores obtained by BLEU* and BERTScore to compute the Person correlations with human judgments respectively. The reliability models introduced in Section 3 trained with BLEU and BERTScore are referred as **REAM♯(BLEU*)** and **REAM♯(BS)** respectively. The augmented dataset mentioned in Section 3.2 is used for the corresponding model training and testing. To show its effectiveness on automatic annotation and interactive annotation, we use the 500 test samples in the raw dataset introduced in Section 2.1. See Appendix B for implementation and training details.

# Refs	MSE	Pred.	Gold
REAM♯(BLEU*)			
3	0.006 ±0.005	0.333	0.340
5	0.006 ±0.006	0.374	0.379
10	0.006 ±0.005	0.399	0.402
20	0.006 ±0.005	0.412	0.409
30	0.007 ±0.006	0.413	0.417
40	0.006 ±0.006	0.422	0.425
REAM♯(BS)			
3	0.006 ±0.006	0.356	0.355
5	0.006 ±0.005	0.367	0.375
10	0.006 ±0.006	0.393	0.404
20	0.006 ±0.005	0.397	0.408
30	0.006 ±0.005	0.406	0.417
40	0.007 ±0.006	0.417	0.431

Table 2: Results of the **REAM♯(BLEU*)** model and the **REAM♯(BS)** model on multiple test sets consisting of different number of reference responses.

5.2 Results on Reliability Prediction Models

In Table 2, we report MSE, the averaged predicted reliability scores and averaged gold reliability scores. Standard deviations are also provided. We can see that the MSE of both prediction models are kept at a low level on different test sets. The difference between the average predicted reliability score and the average gold scores is at most 0.014. Also, the models have good stability and rarely show extreme cases, as shown by their standard deviations of less than 0.01. Our proposed models also have good generalization performance in terms of different sizes of the reference set. The number of references of training samples in the training set is at most 10, but the two models still have a small test error on the test set with more than 10 references, which is comparable to the test set with less than 10 references.

5.3 Results on Automatic Annotation

Once we have a reliable model, the next is to use the model to help us collect high-quality references. When enough candidate responses are available, we can directly use the model to identify which

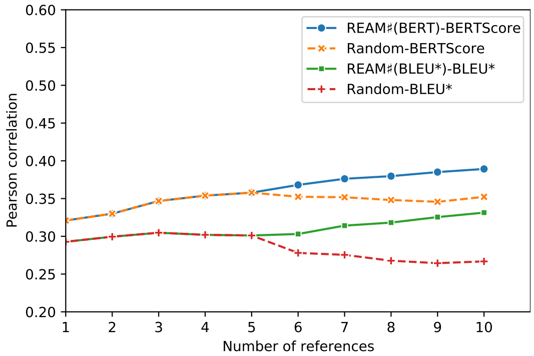

Figure 3: Pearson correlations of BLEU* and BERTScore on multiple reference sets obtained by different strategies. "REAM♯(BS)-BERTScore" means using the **REAM♯(BS)** to augment the reference set and testing it with **BERTScore**. "REAM(BLEU*)-BLEU*" means using the **REAM♯(BLEU*)** to augment the reference set and testing it with **BLEU***. We also add a random strategy for comparison.

responses are helpful to improve the reliability of the set. The experiment on automatic annotation is conducted with the raw test set introduced in Section 2.1. For each sample in the test set, we initialize the reference set with a randomly sampled crawled reference response, then add the remaining reference responses (as mentioned before, each sample has at most 200 reference responses) to its reference set one by one. Each time a new response is added, the model predicts a reliability score. The augmentation method follows the strategy mentioned in Sec. 4.1. We selected 10 responses for each sample.

Figure 3 shows the Pearson correlations of BLEU* and BERTScore on multiple reference sets obtained by different strategies. "REAM♯(BS)-BERTScore" means using the **REAM♯(BS)** model to augment the reference set and testing it with **BERTScore**. Similarly, "REAM(BLEU*)-BLEU*" means using the **REAM♯(BLEU*)** model to augment the reference set and testing it with **BLEU***. We also use a random strategy for comparison (orange and red), where the first 5 responses are the same as those previously selected using the models, and the last 5 responses are obtained by randomly sampling. As shown in the figure, the reliability of the reference sets constructed using our proposed method tends to increase steadily as more selected responses are added to the set, while the reliability of the reference sets constructed using the random strategy appears to be unstable. We also test the transferability of the model and find that the model trained with one metric also yields reliable performance when tested with other metrics. The results are shown in Appendix E.

The final results are shown in Table 3 which reports Pearson and Kendall correlations of BLEU* and BERTScore calculated using three reference sets. "Raw" denotes the initial reference set containing the first selected reference response. The remaining 9 selected responses for each sample are then used to augment the "Raw" set. "Aug-REAM♯(BS)" and "Aug-REAM♯(BLEU*)" are the reference sets augmented using the **REAM♯(BS)** and **REAM♯(BLEU*)**, respectively. "Mix" is the union of the two sets. As can be seen, the performances of the two metrics both improve using the augmented reference set. When combining the two augmented reference sets, both Pearson and Kendall correlations of BERTScore improve, while for BLEU*, the Pearson and the Kendall correlations dropped slightly. This indicates that BERTScore is better at capturing semantics than BLEU* and is able to select the most adequate reference response from multiple augmented sets for evaluation.

5.4 Results on Interactive Annotation

In the following, we perform experiments to augment high-quality references of 30 queries randomly selected in the test set in Section 2.1 using interactive annotation introduced in Section 5.4. For each test query, we first use ElasticSearch [2] to retrieve 100 candidate responses from a database with 200 million (query, response) pairs with Jaccard similarity for each query, and display them to each annotator. We recruit six annotators and divide them into two groups. The reliability scores predicted by the **REAM♯(BERT)** model will reveal to annotators in the first group ("Human-REAM♯(BS)") for interactive annotation but not annotators in the second group ("Human"). Annotators in "Human" perform non-interactive annotations. They are required to rewrite their considered unsuitable responses directly without the predicted scores as a reference. For each round of annotation, we pick one identical candidate response to all annotators.

Figure 4 shows the Pearson correlations of human evaluation results and BERTScore with the

---

[2] Elasticsearch is a search engine based on the Lucene library. `https://www.elastic.co`

	Pearson	**Kendall**
BLEU*		
Raw	0.246	0.165
Aug-REAM♯(BLEU*)	**0.331**	**0.269**
Aug-REAM♯(BS)	0.315	0.261
Mix	0.329	0.266
BERTScore		
Raw	0.288	0.209
Aug-REAM♯(BLEU*)	0.380	0.311
Aug-REAM♯(BS)	0.389	0.318
Mix	**0.393**	**0.319**

Table 3: Pearson and Kendall correlations of BLEU and BERTScore calculated using different constructed reference sets. "Raw" denotes the reference set consisting of 1 response."Aug-REAM♯(BS)" and "Aug-REAM♯(BLEU*)" are the reference sets (10 reference responses) augmented using the **REAM♯(BS)** and **REAM♯(BLEU*)**, respectively. "Mix" is the union of the two sets.

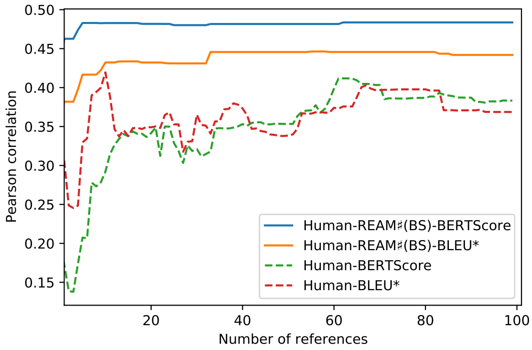

Figure 4: Pearson correlations of the reference sets constructed with/without the **REAM♯(BS)** model in interactive annotations, respectively. "Human-REAM♯(BS)" denotes the annotator with the model assistance and "Human" is the annotator without the model assistance.

augmented reference sets after each round of annotation by one annotator in "Human-REAM♯(BS)" and another annotator in "Human". See Appendix D for the results of all six annotators. We can see that using the augmented response set from annotators in the first group has already reached a relatively high Pearson correlation with only a few annotation rounds. When the number of responses increases to a certain level, the reliability score of the metric hits a bottleneck and rises more slowly. However, the correlation results using the augmented response set from annotators in the second group are not stable. Though the overall trend is increasing, the final obtained reference set has even worse correlations than a much small augmented set from the first group. This shows that our interactive annotation strategy is effective to help annotators avoid writing responses with unsatisfactory quality.

6 Related Work

Automatic evaluation is crucial to the research of open-domain dialog systems (Li and Jurafsky, 2016; Li et al., 2017; Gao et al., 2019; Venkatesh et al., 2018; Chan et al., 2021; Xiang et al., 2021). Existing metrics can be broadly categorized into reference-based and reference-free metrics (Chen et al., 2021). In this work, we focus on reference-based metrics, which usually measure how similar a generated response is to the reference response.

The most commonly used reference-based metrics for dialog systems were originally proposed for machine translation. They typically count the amount of word-overlap between the generated response and the reference response. BLEU (Papineni et al., 2002) is the most widely used metric in machine translation that calculates the geometric mean of the precision for n-gram. Other related word-overlap metrics such as NIST (Lin and Och, 2004), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) also have been used for dialogue evaluation.

Instead of using word-overlap based metrics, embedding-based metrics are adopted to consider the semantic meaning of a word as defined by a distributed representation. They typically compute the similarity between the generated response and reference response using approximated sentence-level representations. The most commonly used word embedding based metrics use a heuristic to combine the vector representation of the individual word in the sentence. For example, Embedding Average (Foltz et al., 1998; Mitchell and Lapata, 2008), Vector Extrema (Forgues and Pineau, 2014), and Greedy Matching (Rus and Lintean, 2012). Zhang et al. (2020) introduce a better embedding-based metric BERTScore that computes token similarity using contextual embeddings that capture the specific use of a word in a sentence.

A few reference-based metrics for dialog systems are learnable functions. ADEM (Lowe et al., 2017) which is based on neural networks is trained to predict a score of a response given its query and a reference response. RUBER (Tao et al., 2018) evaluates responses with a blending of scores from the referenced and unreferenced metrics. RUBER is learnable, but its training does not require human annotation scores.

As discussed from the very beginning of our work, all the above work focus on designing a better metric. However, the reliability of the reference set is also a key to improve the correlation of reference-based metrics, but not investigated in detail in previous work. Therefore, we believe our work can fill this vacancy and provide a new direction to improve reference-based metrics for open-domain dialogue generation.

7 Conclusions and Future Work

In this paper, we first clarify an assumption on existing reference-based metrics that if more high-quality reference responses are added to the reference set, it should have a higher correlation with human judgment. For metrics satisfying this assumption, we present **REAM♯**, an enhancement approach to improve their reliability. In our approach, a reliability prediction model is trained to estimate the reliability of the reference set and we explore both automatic and interactive ways to augment high-quality references with the help of the reliability prediction model. Experiments show that our approach can efficiently help augment high-quality references and the correlations of reference-based metrics improve when using the augmented reference sets to evaluate dialog responses. Our work currently focuses on open-domain dialog systems as a starting point. However, the **REAM♯** framework can be extended naturally to other open-ended text generation tasks such as story generation and question generation.

8 Ethical Considerations

The dataset used in our work are crawled from several Chinese social media websites, BaiduTieba, Douban, Weibo and Zhihu. We purposefully avoid deanonymization techniques based on exploiting software vulnerabilities and our approach that involves human participation in rewriting responses gives us no access to any personally identifiable information.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Zhangming Chan, Lemao Liu, Juntao Li, Haisong Zhang, Dongyan Zhao, Shuming Shi, and Rui Yan. 2021. Enhancing the open-domain dialogue evaluation in latent space. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: Findings*.

Wang Chen, Piji Li, and Irwin King. 2021. A training-free and reference-free summarization evaluation metric via centrality-weighted relevance and self-referenced redundancy. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

P. Foltz, W. Kintsch, and T. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*.

Gabriel Forgues and Joelle Pineau. 2014. Bootstrapping dialog systems with word embeddings.

Markus Freitag, David Grangier, and Isaac Caswell. 2020. BLEU might be guilty but references are not innocent. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, and Shuming Shi. 2019. Generating multiple diverse responses for short-text conversation. In *The Thirty-Third AAAI Conference on Artificial Intelligence*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*.

J. Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *ArXiv*.

J. Li, Will Monroe, and Dan Jurafsky. 2016a. A simple, fast diverse decoding algorithm for neural generation. *ArXiv*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Xin Li, Piji Li, Wei Bi, Xiaojiang Liu, and Wai Lam. 2020a. Relevance-promoting language model for short-text conversation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Yangming Li, Kaisheng Yao, Libo Qin, Wanxiang Che, Xiaolong Li, and Ting Liu. 2020b. Slot-consistent NLG for task-oriented dialogue systems with iterative rectification network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Yahui Liu, Wei Bi, Jun Gao, Xiaojiang Liu, Jian Yao, and Shuming Shi. 2018. Towards less generic responses in neural conversation models: A statistical re-weighting method. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. RUBER: an unsupervised method for automatic evaluation of open-domain dialog systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations*.

Anu Venkatesh, C. Khatri, A. Ram, Fenfei Guo, R. Gabriel, Ashish Nagar, Rohit Prasad, M. Cheng, Behnam Hedayatnia, A. Metallinou, Rahul Goel, Shaohua Yang, and A. Raju. 2018. On evaluating and comparing open domain dialog systems. *arXiv: Computation and Language*.

Jiannan Xiang, Yahui Liu, Deng Cai, Huayang Li, Defu Lian, and Lemao Liu. 2021. Assessing dialogue systems with distribution distances. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: Findings*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations*.

A Dataset

The dataset used in our work are crawled from several Chinese social media websites, BaiduTieba, Douban, Weibo and Zhihu. We purposefully avoid deanonymization techniques based on exploiting software vulnerabilities and our approach that involves human participation in rewriting responses gives us no access to any personally identifiable information. We crawled 5,000 queries and each query has at most 200 ref-erence responses. Then, we collected 14 machine-generated responses obtained from several widely used responses generation models:

- LSTM-S2S-BS: a LSTM Seq2Seq model that generates responses with beam search.
- LSTM-S2S-Sampling: a LSTM Seq2Seq model that generates responses with top k sampling.
- LSTM-S2S-MMI: a LSTM Seq2Seq model with a Maximum Mutual Information-based decoding strategy.
- Fconv-S2S-BS: a convolutional Seq2Seq model (Gehring et al., 2017) that generates responses with beam search.
- Fconv-S2S-Sampling: a convolutional Seq2Seq model (Gehring et al., 2017) that generates responses with top k sampling.
- Fconv-S2S-DBS: a convolutional Seq2Seq model (Gehring et al., 2017) that generates responses with diverse beam search (Li et al., 2016a).
- Transformer-S2S-BS: a Transformer Seq2Seq model that generates responses with beam search.
- Transformer-S2S-Sampling: a Transformer Seq2Seq model that generates responses with top k sampling.
- Transformer-S2S-DBS: a Transformer Seq2Seq model that generates responses with diverse beam search (Li et al., 2016a).
- GPT2-BS: a GPT2 (Radford et al., 2019) model that generates responses with beam search.
- GPT2-TopK Sampling (k=20): a GPT2 (Radford et al., 2019) model that generates responses with top 20 sampling.
- GPT2-TopK Sampling (k=10): a GPT2 (Radford et al., 2019) model that generates responses with top 10 sampling.

The GPT2 model are trained on a 200 million chinese dataset crawled also from the Chinese social media websites (BaiduTieba, Douban, Weibo and Zhihu). The LSTM-S2S, Fconv-S2S and Transformer-S2S models are trained on a benchmark dataset with 7M query-response pairs proposed by Liu et al. (2018).

B Implementation and Training Details

We leverage "bert-as-service" (`https://github.com/hanxiao/bert-as-service`), an open-source system that uses BERT as a sentence encoder and hosts it as a service to map sentences into fixed-length representations. In our work, we use the character-level BERT pre-trained in Chinese. Our reliability prediction model is implemented using "PyTorch Geometric" which is a geometric deep learning extension library for PyTorch. We use one layer. For graph encoding, we employ a one-layer graph attention network with an input size of 768. The input size of the prediction linear layer also is 768. The model is trained using Adam optimizer with a learning rate of 0.0005. The batch size is 128.

For computing BLEU, we use the Python NLTK library. For computing BERTScore, we use the implementation provided by Zhang et al. (2020) at `https://github.com/Tiiiger/bert_score`.

C Case Study

Figure 6 shows two examples of using our **REAM♯(BS)** model to predict reliability scores for reference sets. Given a query and a reference set consisting of four reference responses, we first predict a reliability score for this set (e.g 0.312 and 0.275). We prepared four different candidate responses for each sample. The sentences with blue text are high-quality and diverse responses. Sentences without color are responses that are similar in meaning to the responses already in the set. Red sentences are responses that are completely irrelevant to the queries. We add each of the four candidate responses to the set and predict the reliability score for the new set. As shown in the figure, when the high-quality responses are added to the set, the model predicts a higher reliability score than the original reliability score. When responses with repeated semantics are added to the set, the model predicts a slightly lower reliability score compared to the original one. When poor quality responses are added to the set, the reliability scores predicted by the model drop sharply. It can be seen that our model can effectively evaluate the impact of responses with different qualities on the reference set.

D More Results on Interactive Annotation

Figure 5 shows the results of all six annotators. We can see that using the augmented response set from annotators in the first group has already reached a relatively high Pearson correlation with only a few annotation rounds. When the number of responses increases to a certain level, the reliability score of the metric hits a bottleneck and rises more slowly. However, the correlation results using the augmented response set from annotators in the second group are not stable. Though the overall trend is increasing, the final obtained reference set has even worse correlations than a much small augmented set from the first group. This shows that our interactive annotation strategy is effective to help annotators avoid writing responses with unsatisfactory quality.

E Transferbility

We would like to see if the model trained with one metric also yields reliable performance when tested with other metrics. Figure 7 shows the Pearson correlations of the two models, **REAM♯(BS)** and **REAM♯(BLEU*)** tested using BLEU* and BERTScore, respectively. From the figure, we can see that whether tested with BLEU* or BERTScore, the difference in performance between **REAM♯(BS)** and **REAM♯(BLEU*)** is very small. We also notice that the difference in performance tested with BLEU* between the two models is somewhat larger than that tested with BERTScore when the number of references is large. This may be because BLEU* does not utilize the semantic information compared to BERTScore, which leads to some high-quality responses in the reference set being ignored. Therefore, our enhancement approach is more effective as the metric's ability to capture semantic similarity increases.

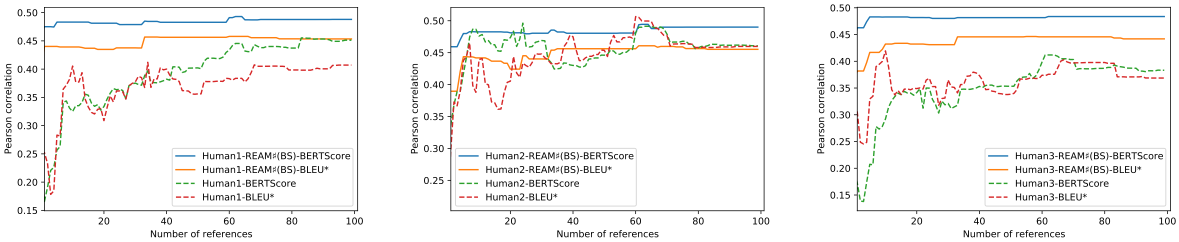

Figure 5: Pearson correlations of the reference sets constructed with and without the **REAM♯(BS)** model, respectively. "Human-REAM♯(BS)" denotes the annotator with the model assistance and "Human" is the annotator without the model assistance.

Query	**Reference Set**	**Candidate Response**	**Pred. Score**
什么是快乐? What is happiness?	快乐是发自内心的一种超幸福的感觉 Happiness is a feeling of super happiness from the heart 快乐是用自己的双手去创造并去理解其中的道理 Happiness is to create with your own hands and to understand the truth of it 快乐是人与生俱来的一种心情。 Happiness is an innate human mood. 快乐是你的感觉,主要是由你的心态决定 Happy is your feeling. Whether you are happy is mainly determined by your state of mind	+ ∅	0.312
		+ 快乐是什么?用最简单的话说快乐就是你的一种满足感 What is happiness? In brief, happiness is feeling contented	0.330↑
		+ 快乐是我们的感觉 Happiness is what we feel	0.310↓
		+ 我很痛苦 I feel a lot of pain	0.218↓
		+ 今天我得去上课 I have to go to class today.	0.293↓
为什么要上学? Why do we need to go to school?	因为要考大学,还要学知识 Because we have to go to college, and we have to learn knowledge 因为要得到知识,以后好上班挣钱养活自己。 Because we need to get knowledge so that we can get a job and earn money to support ourselves in the future. 因为要生活得更好 Because we want to live better 因为这个社会需要,也为了能更好的适应社会吧 Because the society needs us to go to school and going to school also enables us to better adapt to society.	+ ∅	0.275
		+ 学习技能,交友 We have to learn skills and make friends	0.284↑
		+ 因为要学习知识,长大以后才能有好工作 Because we need to learn knowledge so that we can have a good job when we grow up	0.271↓
		+ 我想吃冰淇淋 I want to eat ice cream	0.247↓
		+ 周末我想打篮球 I want to play basketball on the weekend	0.222↓

Figure 6: Two examples of using our **REAM♯(BS)** model to predict reliability scores for reference sets. The sentences with blue text are high-quality and diverse responses. Sentences without colour are responses that are similar in meaning to the responses already in the set. Red sentences are responses that are completely irrelevant for the queries.

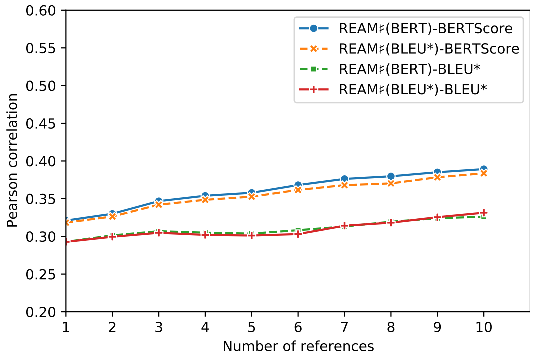

Figure 7: Pearson correlations of the two models, **REAM♯(BS)** and **REAM♯(BLEU*)** tested using BLEU* and BERTScore, respectively.

Relation Extraction with Type-aware Map Memories of Word Dependencies

Guimin Chen[♢*], **Yuanhe Tian**[♥*], **Yan Song**[♠♡†], **Xiang Wan**[♡]
[♢]QTrade [♥]University of Washington
[♠]The Chinese University of Hong Kong (Shenzhen)
[♡]Shenzhen Research Institute of Big Data
[♢]`chenguimin@foxmail.com` [♥]`yhtian@uw.edu`
[♠]`songyan@cuhk.edu.cn` [♡]`wanxiang@sribd.cn`

Abstract

Relation extraction is an important task in information extraction and retrieval that aims to extract relations among the given entities from running texts. To achieve a good performance for this task, previous studies have shown that a good modeling of the contextual information is required, where the dependency tree of the input sentence can be a beneficial source among different types of contextual information. However, most of these studies focus on the dependency connections between words with limited attention paid to exploiting dependency types. In addition, they often treat different dependency connections equally in modeling so that suffer from the noise (inaccurate dependency parses) in the auto-generated dependency tree. In this paper, we propose a neural approach for relation extraction, with type-aware map memories (TaMM) for encoding dependency types obtained from an off-the-shelf dependency parser for the input sentence. Specifically, for each word in an entity, TaMM maps all associated words along with the dependencies among them to memory slots and then assigns a weight to each slot according to its contribution to relation extraction. Our approach not only leverages dependency connections and types between words, but also distinguishes reliable dependency information from noisy ones and appropriately model them. The effectiveness of our approach is demonstrated by the experiments on two English benchmark datasets, where our approach achieves state-of-the-art performance on both datasets.[1]

1 Introduction

Relation extraction is an important natural language processing (NLP) task that facilitates information extraction, whose results is beneficial to downstream tasks such as schema induction (Nimishakavi et al., 2016), knowledge graph construction (Yu et al., 2017), and question answering (Xu et al., 2016). Normally, relation extraction aims to predict the relation between each pair of entities in a given sentence. For example, in the sentence "*the [bone marrow]*$_{e_1}$ *produces [stem cells]*$_{e_2}$" with the entity terms "*bone marrow*" and "*stem cells*", the relation between the two entities is "*Product-Producer*". Therefore, the ability of modeling the context from the input is of great importance to guarantee the performance of relation extraction. To this end, approaches based on neural networks have achieved promising success for the task in the past decade (Socher et al., 2012; Zeng et al., 2014; Zhang and Wang, 2015; Xu et al., 2015; dos Santos et al., 2015; Zhang et al., 2015; Wang et al., 2016; Zhou et al., 2016; Zhang et al., 2017; Wu and He, 2019; Soares et al., 2019; Fu et al., 2019; Aydar et al., 2020; Tian et al., 2021c) because of their effectiveness in capturing contextual information by powerful encoders.

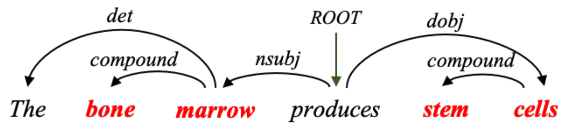

Figure 1: An illustration of an example sentence (including the entity terms "*bone marrow*" and "*stem cells*") with its dependency parsing result.

In addition, previous studies try to improve relation extraction performance by incorporating extra knowledge into their models. Among all such knowledge, syntactic information from the auto-generated dependency parse of the input sentence indicates its helpfulness to improve model performance for the reason that word dependencies provide long distance contextual information (Xu et al., 2015). However, in previous studies, the main focus is the dependencies among words, with little attention paid to dependency types, which are also

---

[*] Equal contribution.

[†] Corresponding author.

[1] The code and models involved in this paper are released at `https://github.com/cuhksz-nlp/RE-TaMM`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2501–2512
August 1–6, 2021.

essential to help the relation extraction task. For example, Figure 1 shows the dependency tree of a sentence where the entities (i.e., "*bone marrow*" and "*stem cells*") are highlighted in red; the dependency type "*nsubj*" (nominal subject) between "*bone marrow*" and "*produces*" as well as the type "*dobj*" (direct object) between "*stem cells*" and "*produces*" indicates the first (i.e., "*bone marrow*") and the second entity (i.e., "*stem cells*") are the subject and object of "*produces*", which provide important cues to predict the relation between the two entities. Moreover, previous studies also suffer from the noise in the auto-generated dependency tree, in which cases all the dependencies are modeled equally without identifying their contributions to the task.[2] Therefore, it is important to design an appropriate approach to leverage the dependency information to improve the relation extraction task.

In this paper, we propose a neural approach for relation extraction, with a type-aware map memory (TaMM) module to encode dependency information obtained from an off-the-shelf dependency parser. Specifically, for each word in an entity, we firstly extracts the dependency information associated with it, where two types of dependency information are considered: the first is "*in-entity*" dependency suggested by the governor and dependents of that word; the second is "*cross-entity*" dependency obtained from the dependency path between entities. Then, TaMM is applied to map the associated words along with the dependency types between them to memory slots and then assign a weight to each slot to distinguish its contribution to the relation extraction task. Compared with other approaches, such as graph neural networks (GCN), to leverage dependency information, our approach not only leverages the dependency type information, but also distinguish reliable dependency information from noisy ones and model them accordingly. The evaluation of different models is performed on two English benchmark datasets, i.e., ACE2005 and SemEval 2010 Task 8 (Hendrickx et al., 2010), where our approach outperforms all baselines and previous studies by achieving the state-of-the-art performance on both datasets.

2 Preliminaries

Relation extraction is conventionally regarded as a text classification task, where an input sentence $\mathcal{X} = x_1 \cdots x_l$ has l words and two entities, i.e., E_1 and E_2, in it are mapped to a particular relation class (denoted by $\widehat{y}$).[3] In most cases the contextual information is of great importance to make a correct prediction for relations. Therefore, it is straightforward to consider integrating extra features to enhance contextual modeling. Of all such features, the syntactic information suggested by the dependency tree of the input sentence has been demonstrated to be useful for relation extraction in many studies (Xu et al., 2015; Zhang et al., 2018; Guo et al., 2019). However, most models to leverage the dependency information are not naturally appropriate to model the dependency types among words. It is required to find an appropriate approach to leverage the dependency type information.

Of all choices, key-value memory networks (KVMN) (Miller et al., 2016) is an effective solution in modeling pair-wisely organized information to improve many NLP tasks (Tapaswi et al., 2016; Das et al., 2017; Mino et al., 2017; Xu et al., 2019; Nie et al., 2020; Song et al., 2020; Tian et al., 2020a,d, 2021b). Specifically, KVMN maps the information instances into a list of memory slots $s_i = (k_i, v_i)$ (i is the index of the memory slot s_i) with k_i referring to the key and v_i the value, respectively. The KVMN addresses the memory slot s_i by assigning a weight p_i to the value v_i by comparing the input (denoted by x) to the key k_i:

$$p_i = \text{softmax}\left(\mathbf{A}\Phi_X(x) \cdot \mathbf{A}\Phi_K(k_i)\right) \tag{1}$$

where $\Phi_\cdot$ are functions that map the input features into their embeddings and $\mathbf{A}$ is a matrix that maps the embeddings into another vector space. After addressing all memory slots, KVMN reads the values by computing the weighted sum of the value vectors (i.e., $\mathbf{A}\Phi_V(v_i)$) using the resulting probability weights (i.e., p_i), which is expressed by

$$\mathbf{a} = \sum_j p_i \cdot \mathbf{A}\Phi_V(v_i) \tag{2}$$

Then, $\mathbf{a}$ is incorporated into the input representation by an element-wise summation:

$$\mathbf{o} = \mathbf{A}\Phi_X(x) + \mathbf{a} \tag{3}$$

Thus, the resulting vector $\mathbf{o}$ contains the weighted information from all values in the memory slots and is finally used to predict the output.

---

[2] For example, in Figure 1, the dependency between "*the*" and "*marrow*" contributes less than the dependency between "*marrow*" and "*produces*" to relation extraction.

[3] E_1 and E_2 are actually sub-strings of $\mathcal{X}$ and we assume E_1 is on the left side of E_2.

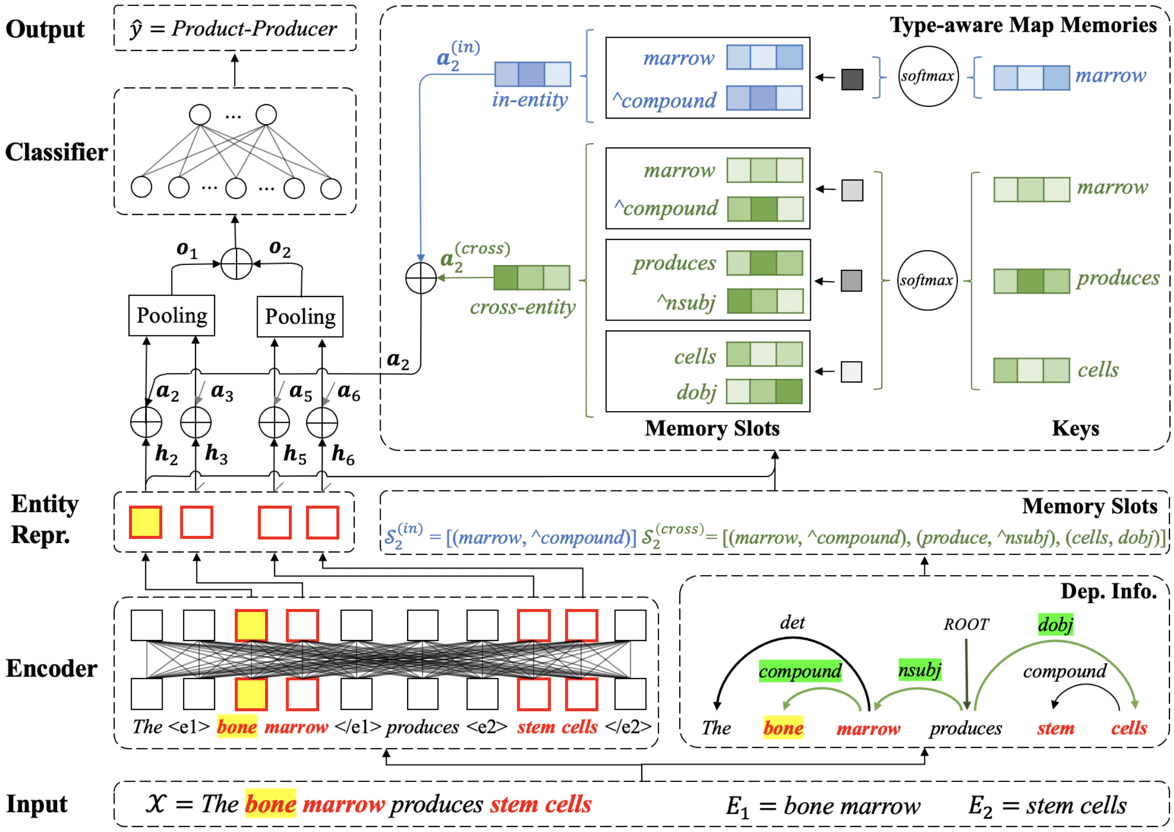

Figure 2: The architecture of our approach with an example sentence where entities are highlighted in red color. The left part illustrates the backbone classification model; the right part shows the process to leverage the in-entity and cross-entity memory slots associated with "*bone*" (highlighted in yellow) through the proposed type-aware map memories (TaMM). In entity and cross-entity memory slots are written in blue and green color respectively.

3 The Proposed Approach

Although KVMN can be used to leverage extra information for relation extraction, it loses the information of keys by using it as a weighting component as stated previously. Therefore, we propose type-aware map memories (TaMM) to leverage both context words (keys) and dependency types (values) to improve relation extraction, where two types of dependency information, i.e., "*in-entity*" and "*cross-entity*" dependencies are considered.

Figure 2 illustrates the architecture of our approach, in which the entities in the input $\mathcal{X}$ is highlighted in red; the left part illustrates the backbone classification model; the right part shows the process of constructing in-entity ($\mathcal{S}^{(in)}$) and cross-entity ($\mathcal{S}^{(cross)}$) memory slots from the dependency tree of the input and the process of incorporating them into the backbone model through TaMM. To summarize, our approach can be formalized as

$$\widehat{y} = \underset{y \in \mathcal{T}}{\arg\max}\, p\left(y|\mathcal{X}, E_1, E_2, \text{TaMM}\left(\mathcal{S}\right)\right) \quad (4)$$

where $\mathcal{T}$ is the set of entity relation types and $\mathcal{S} = (\mathcal{S}^{(in)}, \mathcal{S}^{(cross)})$ is the memory slots for TaMM.

The following texts illustrates the details of our proposed appraoch, including how we construct the memory slots and the computation of TaMM, with its application in relation extraction.

3.1 Memory Slot Construction

In order to construct the memory slots used in our approach, we firstly use an off-the-shelf toolkit to generate the dependency parsing results of the input $\mathcal{X}$. In the parse tree, every word in $\mathcal{X}$ is connected with its governor and its dependents with labeled dependency connections; for any two words in $\mathcal{X}$, there is exactly one path between them[4]. For each word in an entity, e.g., the word x_{i_u} in E_u (i_u is the index of x_{i_u} in $\mathcal{X}$ and $u \in \{1, 2\}$), we consider two types of dependency information suggested by the obtained dependency tree of $\mathcal{X}$ and construct their corresponding memory slots. The first one is "*in-entity*" memory slots constructed upon all the governor and dependents of x_{i_u} (i.e., first-order dependencies). The second is "*cross-entity*" memory

---

[4] The dependency parsing results actually build a graph (tree) of the input $\mathcal{X}$, where words in $\mathcal{X}$ represent the graph nodes, and the dependency connections are the graph edges.

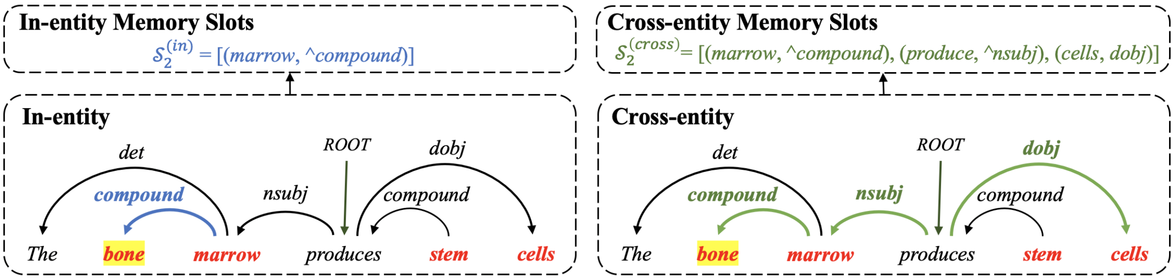

(a) (b)

Figure 3: An illustration of the construction process for two types of memory slots (i.e., in-entity memory slots (a) and cross-entity memory slots (b)) for "*bone*" (with yellow background). Entities are presented in red color.

slots constructed upon the words and dependency arcs along the dependency path between x_{i_u} and the words in the other entity[5]. Figure 3 shows the process to construct the two types of memory slots from the dependency tree of a sentence, where the entities in it are highlighted in red. In the following text, we illustrate the way to extract the in-entity and the cross-entity memory slots for x_{i_u}.

In-entity Memory Slots In-entity memory slots focus on the contextual information from the words connecting to x_{i_u} by dependency parses. To construct them, we firstly locate the governor and all dependents of x_{i_u} in $\mathcal{X}$ from the dependency tree. Then we regard the governor and dependents as the keys in the memory slots and their dependency relations with x_{i_u} as their corresponding values. Therefore, we obtain a list of memory slots with the j-th of them denoted as $s_{i_u,j}^{(in)} = (k_{i_u,j}^{(in)}, v_{i_u,j}^{(in)})$, where $k_{i_u,j}^{(in)}$ is the word connected with x_{i_u} by a dependency connection and $v_{i_u,j}^{(in)}$ the dependency relation type between them. For example, in Figure 3(a), for the word "*bone*" (highlighted with yellow background) in the first entity "*bone marrow*", we find its dependent "*marrow*" and the dependency relation type *compound* between them (the dependency with its type is highlighted in blue) and obtain the dependency slot $\mathcal{S}_2^{(in)}$ =[(*morrow*, ˆ*compound*)].[6] In this case, there is only one word (i.e., "*marrow*") associated with "*bone*". Similarly, if the word we focus on is "*marrow*", the in-entity memory slots for it should be $\mathcal{S}_3^{(in)}$ = [(*The*, *det*), (*bone*, *compound*), (*produces*, ˆ*nsubj*)].

Cross-entity Memory Slots Cross-entity memory slots aim to incorporate the contextual information along the dependency path in between the two entities. To construct cross-entity memory slots, for each x_{i_u} in E_u (we denote the other entity as $E_{\widetilde{u}}$), we firstly find the dependency path from x_{i_u} to the last word of $E_{\widetilde{u}}$. The motivation of using the last word is that noun phrases in English (entities are always noun phrases) tend to be head-final[7]. Then, similar to the process of constructing in-entity memory slots, we extract all words along that path (including the last word of $E_{\widetilde{u}}$) as well as the corresponding dependency relation types. Finally, we regard the words as keys and the dependency relation types as values in the memory slots and denote the j-th memory slot as $s_{i_u,j}^{(cross)} = (k_{i_u,j}^{(cross)}, v_{i_u,j}^{(cross)})$. As illustrated in Figure 3(b), for "*bone*" (highlighted with yellow background) in the first entity "*bone marrow*", we locate the dependency path between "*bone*" and the last word "*cells*" of the second entity "*stem cells*": "*bone – marrow – produces – cells*", as well as the dependency relation types along that path: "*compound*" for "*bone – marrow*", "*nsubj*" for "*marrow – produces*", and "*dobj*" for "*produces – cells*" (highlighted in green). Therefore, the cross-entity memory slots for "*bone*" are $\mathcal{S}_2^{(cross)}$ = [(*marrow*, ˆ*compound*), (*produces*, ˆ*nsubj*), (*cells*, *dobj*)].

In summary, for x_{i_u} in in E_u, we obtain the in-entity memory slot list $\mathcal{S}_{i_u}^{(in)}$ and the cross-entity memory slot list $\mathcal{S}_{i_u}^{(cross)}$, which are fed into the TaMM module as illustrated in Figure 2.

3.2 Type-aware Map Memories

There are previous approaches for relation extraction that leverage dependency information and focus on dependencies among words without considering their dependency types. With learning from such information, there is a nonnegligible challenge that there are noises in the auto-generated depen-

---

[5] "*The other entity*" means E_2 if x_{i_u} is a word in E_1.

[6] We add a ˆ mark before the dependency type to illustrate the directional information of the dependency type.

[7] For example, the head of "*bone marrow*" is "*marrow*" and the head of "*stem cells*" is "*cells*".

dency results, which may hurt model performance since they provide misleading contextual information. One straightforward way to address this issue is to weight different dependencies according to their contribution to the relation extraction task. As discussed in the previous section, although KVMN provides a way to selectively model dependency information it is limited in omitting the contextual information carried by the keys in the final output from the memories.

To address the aforementioned limitations in KVMN, we propose type-aware map memories (TaMM) to incorporate the dependency information carried by both the keys and values (i.e., the memory slots), where the architecture of TaMM is illustrated on the top right of Figure 2. Specifically, for each word in an entity, e.g., the word x_{i_u} in E_u (i_u is the index of x_{i_u} in $\mathcal{X}$ and $u \in \{1, 2\}$), we consider two types of dependency information, i.e., "*in-entity*" and "*cross-entity*" dependency information, and construct their corresponding memory slots. We denote the j-th in-entity and cross-entity memory slots as $s^{(in)}_{i_u,j} = (k^{(in)}_{i_u,j}, v^{(in)}_{i_u,j})$ and $s^{(cross)}_{i_u,j} = (k^{(cross)}_{i_u,j}, v^{(cross)}_{i_u,j})$, respectively, and use the same process to model them.

Taking the in-entity memory slots as an example, we firstly use two matrices to map the keys $k^{(in)}_{i_u,j}$ and values $v^{(in)}_{i_u,j}$ in the memory slots into their embeddings, which are denoted by $\mathbf{e}^{k,(in)}_{i_u,j}$ and $\mathbf{e}^{v,(in)}_{i_u,j}$, respectively. Next, we compute the weight $p_{i_u,j}$ assigned for each value through the inner production between the key embedding $\mathbf{e}^{k,(in)}_{i_u,j}$ and the hidden vector of x_{i_u} (which is denoted as $\mathbf{h}_{i_u}$) obtained from the encoder in the backbone model:

$$p_{i_u,j} = \frac{\exp\left(\mathbf{h}_{i_u} \cdot \mathbf{e}^{k,(in)}_{i_u,j}\right)}{\sum_{j=1}^{m^{(in)}_{i_u}} \exp\left(\mathbf{h}_{i_u} \cdot \mathbf{e}^{k,(in)}_{i_u,j}\right)} \qquad (5)$$

where $m^{(in)}_{i_u}$ is the number of in-entity memory slots associated with x_{i_u}. Then, we apply the weights to the corresponding memory slots and obtain the weighted sum (denoted as $\mathbf{a}^{(in)}_{i_u}$) of both keys and values through

$$\mathbf{a}^{(in)}_{i_u} = \sum_{j=1}^{m^{(in)}_{i_u}} p_{i_u,j}(\mathbf{e}^{k,(in)}_{i_u,j} + \mathbf{e}^{v,(in)}_{i_u,j}) \qquad (6)$$

where "+" refers to element-wise sum of vectors. Therefore, compared to KVMN, our approach is able to leverage both context words and dependency types associated with x_{i_u}.

		ACE2005	SemEval
# Instances	Train	48,198	8,000
	Dev	11,854	-
	Test	10,097	2,717
# Relation Types		7	19

Table 1: The statistics (number of instances and relation types) of the two benchmark datasets.

With the same process for in-entity memory slots, we deal with the cross-entity ones and obtain the weighted sum $\mathbf{a}^{(cross)}_{i_u}$. Finally, we concatenate the two resulting vectors by $\mathbf{a}_{i_u} = \mathbf{a}^{(in)}_{i_u} \oplus \mathbf{a}^{(cross)}_{i_u}$ with $\mathbf{a}_{i_u}$ denoting the output of TaMM and containing the weighted dependency information to enhance the backbone model.

3.3 Relation Extraction with TaMM

Once the TaMM is built, it is straightforward to apply it to relation extraction through a backbone classifier. In our approach, we use BERT (Devlin et al., 2019) as the classifier to encode the input $\mathcal{X}$ and obtain the hidden vectors for all words. Note that we only use the hidden vectors of the words in the two entities to predict their relations. Therefore, for each word x_{i_u} in the entity E_u, we feed $\mathbf{h}_{i_u}$ into TaMM and obtain the corresponding output $\mathbf{a}_{i_u}$. Then, we concatenate $\mathbf{h}_{i_u}$ and $\mathbf{a}_{i_u}$, and for each entity E_u, use the max pooling strategy to obtain the vectorized representation $\mathbf{o}_u$ by

$$\mathbf{o}_u = \text{MaxPooling}(\mathbf{h}_{i_u} \oplus \mathbf{a}_{i_u}) \qquad (7)$$

Afterwards, we concatenate the representation of the two entities (i.e. $\mathbf{o}_1$ for E_1 and $\mathbf{o}_2$ for E_2) and pass the resulting vector through a fully connected layer (a classifer) to obtain the final prediction $\widehat{y}$ by

$$\widehat{y} = \mathbf{W} \cdot (\mathbf{o}_1 \oplus \mathbf{o}_2) + \mathbf{b} \qquad (8)$$

where $\mathbf{W}$ and $\mathbf{b}$ are the trainable weight matrix and bias vector for the fully connected layer.

4 Experimental Settings

4.1 Datasets

Two English benchmark datasets, i.e., ACE2005EN (ACE2005)[8] and SemEval 2010 Task 8 (SemEval)[9] (Hendrickx et al., 2010) are used in the experiments to evaluate our approach. For ACE2005, we follow the same preprocess as that in Christopoulou

---

[8] `https://catalog.ldc.upenn.edu/LDC2006T06`.

[9] `http://docs.google.com/View?docid=dfvxd49s_36c28v9pmw`.

Hyper-parameters	Values
Learning Rate	$5e-6, 1e-5, 2e-5, \mathbf{3e-5}$
Warmup Rate	$\mathbf{0.06}, 0.1$
Dropout Rate	$\mathbf{0.1}$
Batch Size	$16, \mathbf{32}, 64, 128$

Table 2: The hyper-parameters tested in tuning our models. The best ones used in our final experiments are highlighted in boldface.

et al. (2018); Ye et al. (2019), by removing the two small subsets: cts and un, and splitting the remaining 511 documents into three parts: 351 for training, 80 for development and the rest 80 for test[10]. For SemEval, we follow previous studies (Hendrickx et al., 2010; Zeng et al., 2014; Zhang and Wang, 2015; Xu et al., 2015; dos Santos et al., 2015; Zhang et al., 2015; Wang et al., 2016; Zhou et al., 2016; Zhang et al., 2017; Soares et al., 2019) to use its official train/test split. The statistics of the two datasets are summarized in Table 1.

4.2 Implementation

In our experiments, we use Standard CoreNLP Toolkits (SCT)[11] to obtain the dependency tree for each input sentence. Since the quality of text representation plays an important role in the performance of NLP models (Komninos and Manandhar, 2016; Song et al., 2017, 2018; Liu and Lapata, 2018; Song and Shi, 2018; Song et al., 2021), we use BERT[12] (Devlin et al., 2019), which is a pre-trained language model that achieves state-of-the-art in many NLP tasks (Wu and He, 2019; Soares et al., 2019; Tian et al., 2020b,c, 2021a), as the encoder in our model. Specifically, we use the uncased version of BERT with its default settings (e.g., for BERT-base, we use 12 layers of multi-head attentions with 768 dimensional hidden vectors; for BERT-large, we use 24 layers of multi-head attentions with 1024 dimensional hidden vectors) and fine-tune its all trainable parameters in the training stage. For TaMM, we randomly initialize the embeddings of all keys and values with their dimensions matching that of the hidden vectors from BERT. For evaluation, we follow previous studies to use the standard micro-F1 scores[13] for ACE2005 and use the macro-averaged F1 scores[14] for SemEval. For other hyper-parameter settings (i.e., learning rate, warmup rate, dropout rate, and batch size) to train our model, we report them Table 2, where we test all combinations of them for each model and use the one achieving the highest F1 score in our final experiments (the best combination of them is illustrated in boldface).

Models	ACE2005	SemEval
BERT-base	75.31	87.87
+ GCN	75.59	88.19
+ GAT	76.01	88.39
+ KVMN (In)	76.40	88.73
+ TaMM (In)	76.80	88.91
+ KVMN (Cross)	76.45	88.61
+ TaMM (Cross)	76.61	88.74
+ KVMN (Both)	76.83	88.98
+ TaMM (Both)	**77.07**	**89.18**
BERT-large	76.79	89.02
+ GCN	77.17	89.43
+ GAT	77.23	89.39
+ KVMN (In)	77.32	89.42
+ TaMM (In)	77.76	89.72
+ KVMN (Cross)	77.21	89.37
+ TaMM (Cross)	77.66	89.58
+ KVMN (Both)	77.96	89.88
+ TaMM (Both)	**78.98**	**90.06**

Table 3: F1 scores of our TaMM and baselines (i.e., BERT, standard GCN, standard GAT, and KVMN) on the test sets of ACE2005 and SemEval, where BERT-base and BERT-large encoders are used. For KVMN and TaMM, different combinations of in-entity and cross-entity dependency information (i.e., in-entity only, cross-entity only, and both of them) are tried.

5 Results and Analyses

5.1 Overall Performance

In the main experiments, we run our models using BERT-base and BERT-large encoders with and without TaMM and try different combinations of in-entity and cross-entity dependency information (i.e., in-entity dependency information only, cross-entity dependency information only, and both of them). We also run the baselines using standard graph convolutional networks (GCN), standard graph attention networks (GAT), and KVMN to leverage the dependency information. Table 3 shows the results (F1 scores) of different models.[15]

---

[10] We use the dataset split from `https://github.com/tticoin/LSTM-ER/tree/master/data/ace2005/split`.

[11] We use SCT under version 3.9.2 from `https://stanfordnlp.github.io/CoreNLP/`.

[12] We download different BERT models from `https://github.com/huggingface/transformers`.

[13] We use the evaluation script from *sklearn* framework.

[14] We use the official evaluation script downloaded from `http://semeval2.fbk.eu/scorers/task08/SemEval2010_task8_scorer-v1.2.zip`.

[15] For the same group of models, we report the F1 scores on the development sets in Appendix A and the mean and standard deviation of their test set results in Appendix B.

Models	ACE2005	SemEval
Wang et al. (2016)	-	88.0
Zhou et al. (2016)	-	84.0
†Zhang et al. (2018)	-	84.8
Christopoulou et al. (2018)	64.2	-
Ye et al. (2019)	68.9	-
*Wu and He (2019) (BERT-large)	-	89.2
*Soares et al. (2019) (BERT-large)	-	89.5
†Sun et al. (2020)	-	86.0
†Yu et al. (2020)	-	86.4
†Mandya et al. (2020)	-	85.9
*TaMM (Both) (BERT-base)	77.07	89.18
*†TaMM (Both) (BERT-large)	**78.98**	**90.06**

Table 4: The comparison between our models (the ones using TaMM (Both)) and previous studies on ACE2005 and SemEval. Models with dependency features and BERT-large are marked by "†" and "*", respectively.

There are several observations. First, TaMM works well with both BERT base and large, where consistent improvement is observed over the BERT-base and BERT-large baselines across all datasets, although they have already achieved very good performance. Second, TaMM outperforms standard GCN and GAT models, which can be attributed to our modeling of dependency type information in TaMM. Third, under all the three settings to incorporate different types of dependency information (i.e., in-entity, cross-entity, and both), our models with TaMM outperforms the BERT baseline and the highest F1 score is achieved when both in-entity and cross-entity dependency information are used (i.e., + TaMM (Both)). This observation confirms the individual contribution of in-entity and cross-entity dependency information as well as the effectiveness of our approach to leverage them together to improve model performance. Fourth, compared with our TaMM models using cross-entity dependency information only (i.e., + TaMM (Cross)), the models using in-entity dependency information only (i.e., + TaMM (In)) achieves higher results in most cases. One possible explanation could be the following. There are overlaps between in-entity dependencies and cross-entity dependencies. For example, the dependency between "*bone*" and "*marrow*" is shared by both in-entity dependencies and cross-entity dependencies in Figure 3. Therefore, with in-entity dependency only, TaMM not only leverages the contextual words directly associated with the entities themselves, but also can still partially benefit from the contextual information along the dependency path, whereas TaMM with cross-entity dependency only fails to leverage the contextual words directly associated with the entities, which leads TaMM (In) to achieve better performance than TaMM (Cross). Fifth, for all the settings, our model with TaMM consistently outperforms the baselines with KVMN, which demonstrates the effectiveness of our approach to improve relation extraction. The explanation is that TaMM is able to leverage both context words (keys) and dependency types (values) at the same time, while KVMN fails to incorporate the context information carried by keys, which leads KVMN to omit some important features and thus get inferior results.

Models	Order	ACE2005	SemEval
Baseline	N/A	76.79	89.02
TaMM (In)	1st	77.76	89.72
	2nd	77.53	89.59
	3rd	78.05	89.79
TaMM (Both)	1st	**78.98**	**90.06**
	2nd	78.27	89.95
	3rd	78.41	89.91

Table 5: F1 scores of models using BERT-large and TaMM (In/Both) to leverage 1st-, 2nd-, and 3rd-order dependencies. "N/A" refers to no order can be applied.

Moreover, we compare our model under the best setting (i.e., the ones using TaMM to leverage both in-entity and cross-entity dependency relation) with previous studies and report the results (F1 scores) in Table 4. It is found that our model with BERT-large encoder outperforms all previous studies (including the ones also using BERT-large encoder).

5.2 The Effect of Dependency Information

To analyze the effect of using dependency information, we perform three investigations on models using BERT-large encoder.

The first investigation is to examine different orders of dependencies used in TaMM. Previous experiments showed the effectiveness of our model with TaMM on first-order word dependencies. We also try second- and third-order dependencies via the model (i.e., large BERT) with TaMM (Both). The results (with scores from the first-order dependencies) are reported on table 5, where the corresponding results from the models with TaMM (In) as well as the BERT-large baseline are also reported. The observations are drawn as follows. First, models with TaMM under all settings outperforms the BERT-large baseline, which is confirmed by all results on both datasets. Second, models with TaMM (Both) consistently outperform the ones with TaMM (in) under the same setting, which indicates the cross-entity dependencies are able to bring greater improvements. Third, for models

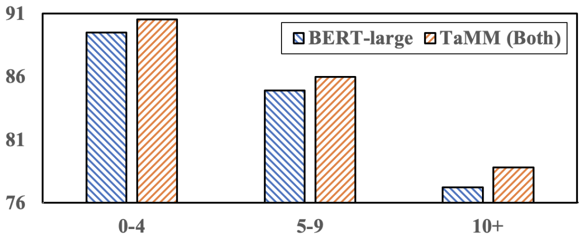

Figure 4: The performance of BERT-large baseline and our TaMM (both) on test instances from SemEval grouped by the entities' distance (i.e., the number of words between two entities).

with TaMM (Both), using higher order dependencies often results in inferior results; while the trend is on the opposite for models with TaMM (in). One possible explanation is that for TaMM (both), most essential word dependencies in between the two entities have already been encoded, higher order dependencies sometimes introduce noise other than useful information; while for TaMM (In), leveraging higher order dependencies allows the model to cover more contextual information along the dependency path between two entities.

The second is to explore the performance of our model on different test instances grouped by their entity distance (i.e., the number of words between the two entities), to see whether our approach can capture long-distance word-word dependencies and help with relatin extraction. In doing so, we split the test set of SemEval into three groups according to the entity distance (i.e., from 0 to 4, from 5 to 9, and higher than 10) and perform our best TaMM model and the BERT baseline on them. Figure 4 illustrates the performance of TaMM (i.e., the orange bar) and BERT (i.e., the blue bar). It can be found that our TaMM outperforms the BERT-baseline on all three groups of test instances, where bigger gaps can be observed when the entities' distance goes higher. This observation demonstrates the effectiveness of our approach to encode dependency information to improve relation extraction.

The third investigation is to explore the effect of TaMM using different dependency parsers. Specifically, in addition to the Stanford CoreNLP Toolkits (SCT) used in the main experiments, we also try spaCy[16] to obtain the dependency trees and report the results (with BERT-large encoder) in Table 6. It is found that models with different dependency parsers consistently outperform the BERT-large baseline, which indicates the robustness of our model design in improving relation extraction.

Models	**Parser**	**ACE2005**	**SemEval**
Baseline	N/A	76.79	89.02
TaMM (In)	SCT	77.76	89.72
	spaCy	77.74	89.78
TaMM (Cross)	SCT	77.66	89.58
	spaCy	77.60	89.61
TaMM (Both)	SCT	**78.98**	**90.06**
	spaCy	78.92	90.01

Table 6: F1 scores of models using BERT-large and TaMM (In/Cross/Both) to leverage dependency information from different parsers (i.e., SCT and spaCy).

5.3 Case Study

To examine how TaMM leverages dependency information to improve model performance, in Figure 5, we show an example input where our approach successfully predicts the relation in between the two entities (in red colors) to be "*Entity-Destination*", while the BERT-large baseline fails to do so ("*Component-Whole*"). In the figure, the dependencies between words are highlighted in different colors to represent the total weights assigned to their corresponding in-entity and cross-entity memory slots, where darker color refers to higher weight. Overall, we find that the most emphasized dependencies are along the dependency path connecting the two entities, where the memory slots for those dependencies receive the highest weights. For the first entity "*treadmill*", the dependency type *ˆnsubj: pass* (passive nominal subject) in the highlighted memory slot (*installed*, *ˆnsubj: pass*) suggests the first entity is the patient of the action *install*; similarly, for the second entity "*space station*", the highlighted dependency type *ˆobj* (object) suggests this entity is the location of the action *install* given the fact that the input is a passive sentence. Therefore, our approach is able to leverage these cues learned from word dependencies and their dependency types so as to predict the correct relation for the two entities: "*Entity-Destination*".

6 Related Work

Relation extraction is an important task in NLP, which significantly relies on a good modeling of the contextual information to achieve outstanding model performance. To improve the capability of context modeling for relation extraction, studies in the past decade leverage neural networks, such as using CNN (Zeng et al., 2014; Wang et al., 2016), RNN (Socher et al., 2012; Xu et al., 2015; Zhou et al., 2016) and BERT encoders (Wu and He, 2019; Soares et al., 2019; Wang et al., 2019). To further

---

[16] `https://spacy.io/`

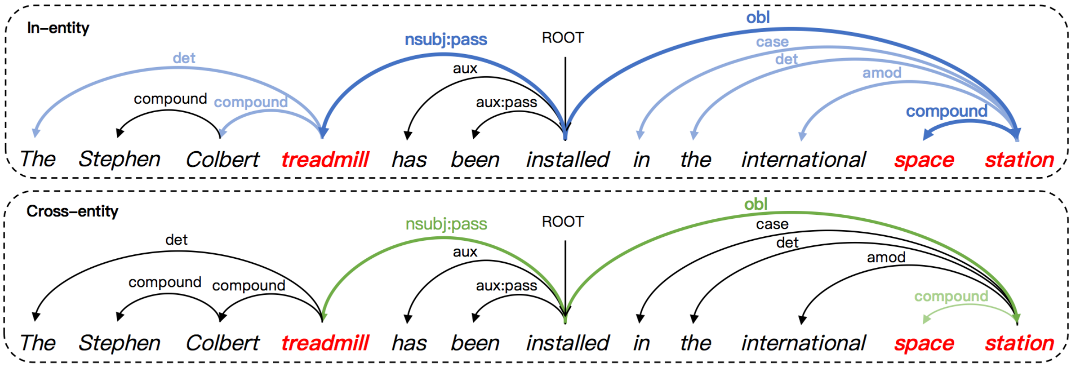

Figure 5: An example input fed into our model with TaMM (Both) and its correctly predicted relation between the two entities marked in red. Word dependencies are highlighted in different colors to visualize the total weights assigned to their corresponding in-entity and cross-entity memory slots, where darker color refers to higher weight.

enhance the models for this task, incorporating extra knowledge into the models has been proved as an effective method, where normally three types of extra knowledge are used: lexical, syntactic and semantic knowledge, and syntactic knowledge has been proved to be useful for this task (Xu et al., 2015). With this finding, there are studies also using advanced neural architecture, such as graph convolutional networks, to incorporate syntactic knowledge from auto-generated dependency parse of the input sentence (Zhang et al., 2018; Guo et al., 2019; Sun et al., 2020; Yu et al., 2020; Mandya et al., 2020). Compared to the aforementioned studies, TaMM offers a simple yet effective non-graph-based approach to leverage dependencies for relation extraction. TaMM provides the ability not only incorporate both word dependencies and their types into the model to help improve relation extraction performance, but also discriminatively leverage the dependencies by assigning different weights to them, which can address the potential noise in the auto-generated dependencies and thus further improve model performance.

7 Conclusion

In this paper, we proposed an effective method for relation extraction with word dependencies encoded by TaMM, whose keys and values are built upon the dependency tree of the input sentence obtained from off-the-shelf toolkits. Particularly, for each entity in the sentence, we extract words associated with it according to the dependency parse of the input sentence and their corresponding dependency relation types. Then, we use TaMM to encode and weight such information and integrate it into the relation extraction task. The novelty of this work lies in the modeling of contextual information through dependencies and their relation types encoded in TaMM. Experimental results on two public English benchmark datasets illustrate the effectiveness of our approach with state-of-the-art performance achieved on all datasets.

Acknowledgements

This work is supported by Chinese Key-Area Research and Development Program of Guangdong Province (2020B0101350001) and NSFC under the project “The Essential Algorithms and Technologies for Standardized Analytics of Clinical Texts” (12026610). This work is also partially supported by Shenzhen Institute of Artificial Intelligence and Robotics for Society under the project “Automatic Knowledge Enhanced Natural Language Understanding and Its Applications” (AC01202101001).

References

Mehmet Aydar, Ozge Bozal, and Furkan Ozbay. 2020. Neural Relation Extraction: a survey. *arXiv e-prints*, pages arXiv–2007.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A Walk-based Model on Entity Graphs for Relation Extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 81–88.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–365.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling Text as Relational Graphs for Joint Entity and Relation Extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention Guided Graph Convolutional Networks for Relation Extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency Based Embeddings for Sentence Classification Tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500, San Diego, California.

Yang Liu and Mirella Lapata. 2018. Learning Structured Text Representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Angrosh Mandya, Danushka Bollegala, and Frans Coenen. 2020. Graph Convolution over Multiple Dependency Sub-graphs for Relation Extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6424–6435.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Hideya Mino, Masao Utiyama, Eiichiro Sumita, and Takenobu Tokunaga. 2017. Key-value Attention Mechanism for Neural Machine Translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 290–295, Taipei, Taiwan.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving Named Entity Recognition with Attentive Ensemble of Syntactic Information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4231–4245.

Madhav Nimishakavi, Uday Singh Saini, and Partha Talukdar. 2016. Relation Schema Induction using Tensor Factorization with Side Information. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Cícero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning Word Representations with Regularization from Prior Knowledge. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 143–152.

Yan Song and Shuming Shi. 2018. Complementary Learning of Word Embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4368–4374.

Yan Song, Shuming Shi, and Jing Li. 2018. Joint Learning Embeddings for Chinese Words and Their Components via Ladder Structured Networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4375–4381.

Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. 2020. Summarizing Medical Conversations via Identifying Important Utterances. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 717–729.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*.

Kai Sun, Richong Zhang, Yongyi Mao, Samuel Mensah, and Xudong Liu. 2020. Relation Extraction with Convolutional Network over Learnable Syntax-Transport Graph. In *AAAI*, pages 8928–8935.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding Stories in Movies

Through Question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Yuanhe Tian, Guimin Chen, and Yan Song. 2021a. Aspect-based Sentiment Analysis with Type-aware Graph Convolutional Networks and Layer Ensemble. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2910–2922, Online.

Yuanhe Tian, Guimin Chen, and Yan Song. 2021b. Enhancing Aspect-level Sentiment Analysis with Word Dependencies. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3726–3739, Online.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021c. Dependency-driven Relation Extraction with Attentive Graph Convolutional Networks. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020a. Improving Biomedical Named Entity Recognition with Syntactic Information. *BMC Bioinformatics*, 21:1471–2105.

Yuanhe Tian, Yan Song, and Fei Xia. 2020b. Joint Chinese Word Segmentation and Part-of-speech Tagging via Multi-channel Attention of Character N-grams. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2073–2084.

Yuanhe Tian, Yan Song, and Fei Xia. 2020c. Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. 2020d. Improving Chinese Word Segmentation with Wordhood Memory Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8274–8285, Online.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019. Extracting Multiple-Relations in One-Pass with Pre-Trained Transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1371–1377.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation Classification via Multi-Level Attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307.

Shanchan Wu and Yifan He. 2019. Enriching Pre-trained Language Model with Entity Information for Relation Classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing Key-Value Memory Neural Networks for Knowledge Based Question Answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2937–2947, Minneapolis, Minnesota.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying Relations via Long Short Term Memory Networks Along Shortest Dependency Paths. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1785–1794.

Wei Ye, Bo Li, Rui Xie, Zhonghao Sheng, Long Chen, and Shikun Zhang. 2019. Exploiting Entity BIO Tag Embeddings and Multi-task Learning for Relation Extraction with Imbalanced Data. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1351–1360.

Bowen Yu, Mengge Xue, Zhenyu Zhang, Tingwen Liu, Wang Yubin, and Bin Wang. 2020. Learning to Prune Dependency Trees with Rethinking for Neural Relation Extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3842–3852, Barcelona, Spain (Online).

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved Neural Relation Detection for Knowledge Base Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 571–581.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation Classification via Convolutional Deep Neural Network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Dongxu Zhang and Dong Wang. 2015. Relation Classification via Recurrent Neural Network. *arXiv preprint arXiv:1508.01006*.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware Attention and Supervised Data Improve Slot Filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.

Appendix A. Experimental Results on the Development Set

Table 7 reports the F1 scores of different models on the development set of ACE2005.[17]

Models	**BERT-base**	**BERT-Large**
BERT	75.03	76.51
+ GCN	75.33	76.82
+ GAT	75.77	76.89
+ KVMN (In)	76.28	77.10
+ TaMM (In)	76.61	77.52
+ KVMN (Cross)	76.25	77.06
+ TaMM (Cross)	76.54	77.44
+ KVMN (Both)	76.49	77.48
+ TaMM (Both)	76.86	78.13

Table 7: F1 scores of models with different configurations (i.e., the ones using base or large BERT with KVMN or TaMM and different combinations of in-entity and cross-entity dependency information) on the development set of ACE2005 for relation extraction.

Appendix B. Mean and Deviation of the Results

In the experiments, we test models with different configurations. For each model, we train it with the best hyper-parameter setting using five different random seeds. We report the mean (μ) and standard deviation (σ) of the F1 scores on the test set of ACE2005 and SemEval in Table 8.

Models	**ACE2005**		**SemEval**	
	μ	σ	μ	σ
BERT-base	74.86	0.42	87.48	0.38
+ GCN	75.15	0.31	88.02	0.16
+ GAT	75.70	0.29	88.01	0.36
+ KVMN (In)	76.15	0.24	88.62	0.10
+ TaMM (In)	76.61	0.18	88.76	0.14
+ KVMN (Cross)	75.99	0.42	88.43	0.16
+ TaMM (Cross)	76.43	0.14	88.49	0.24
+ KVMN (Both)	76.44	0.34	88.59	0.36
+ TaMM (Both)	**76.59**	0.46	**88.96**	0.19
BERT-large	76.28	0.47	88.66	0.34
+ GCN	76.29	0.46	89.15	0.26
+ GAT	76.82	0.32	89.12	0.25
+ KVMN (In)	76.98	0.33	89.23	0.13
+ TaMM (In)	77.35	0.38	89.48	0.26
+ KVMN (Cross)	77.06	0.13	89.19	0.17
+ TaMM (Cross)	77.31	0.34	89.45	0.12
+ KVMN (Both)	77.08	0.49	89.61	0.23
+ TaMM (Both)	**78.62**	0.32	**89.88**	0.16

Table 8: The mean μ and standard deviation σ of accuracy and F1 scores of all models (i.e., the ones using base or large BERT with KVMN or TaMM and different combinations of in-entity and cross-entity dependency information) on the test set of ACE2005 and SemEval for relation extraction.

---

[17] SemEval does not have an official dev set.

PLATO-2: Towards Building an Open-Domain Chatbot via Curriculum Learning

Siqi Bao* Huang He* Fan Wang* Hua Wu* Haifeng Wang
Wenquan Wu Zhen Guo Zhibin Liu Xinchao Xu
Baidu Inc., China
{baosiqi, hehuang, wang.fan, wu_hua}@baidu.com

Abstract

To build a high-quality open-domain chatbot, we introduce the effective training process of PLATO-2 via curriculum learning. There are two stages involved in the learning process. In the first stage, a coarse-grained generation model is trained to learn response generation under the simplified framework of one-to-one mapping. In the second stage, a fine-grained generative model augmented with latent variables and an evaluation model are further trained to generate diverse responses and to select the best response, respectively. PLATO-2 was trained on both Chinese and English data, whose effectiveness and superiority are verified through comprehensive evaluations, achieving new state-of-the-art results.

1 Introduction

Recently, task agnostic pre-training with large-scale transformer models has achieved great success in natural language processing (Devlin et al., 2019), especially open-domain dialogue generation. For instance, based on the general language model GPT-2 (Radford et al., 2019), DialoGPT (Zhang et al., 2020) is further trained for response generation using Reddit comments. To obtain a human-like open-domain chatbot, Meena (Adiwardana et al., 2020) scales up the network parameters to 2.6B and employs more social media conversations in the training process, leading to significant improvement on response quality. To mitigate undesirable toxic or bias traits of large corpora, Blender (Roller et al., 2021) fine-tunes the pre-trained model with human annotated datasets and emphasizes desirable conversational skills of engagingness, knowledge, empathy and personality.

In addition to the attempts from model scale and data selection, PLATO (Bao et al., 2020) aims to tackle the inherent one-to-many mapping problem to improve response quality. The one-to-many mapping refers to that one dialogue context might correspond to multiple appropriate responses. It is widely recognized that the capability of modeling one-to-many relationship is crucial for open-domain dialogue generation (Zhao et al., 2017; Chen et al., 2019). PLATO explicitly models this one-to-many relationship via discrete latent variables, aiming to boost the quality of dialogue generation. PLATO has a modest scale of 132M network parameters and trained with 8M samples, achieving relatively good performance among conversation models on a similar scale. However, scaling up PLATO directly encounters training instability and efficiency issues, which might result from the difficulty to capture the one-to-many semantic relationship from scratch.

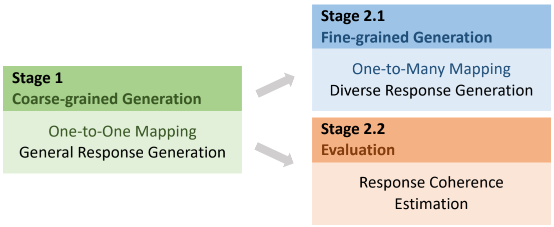

Figure 1: Curriculum learning process in PLATO-2.

In this work, we try to scale up PLATO to PLATO-2 and introduce an effective training schema via curriculum learning (Bengio et al., 2009). There are two stages involved in the whole learning process, as shown in Figure 1. In the first stage, under the simplified one-to-one mapping modeling, a **coarse-grained** generation model is trained for response generation under different conversation contexts. This model tends to capture typical patterns of diversified responses, sometimes resulting in general and dull responses during inference. Despite the problem of safe responses,

---

*Equal contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2513–2525
August 1–6, 2021.

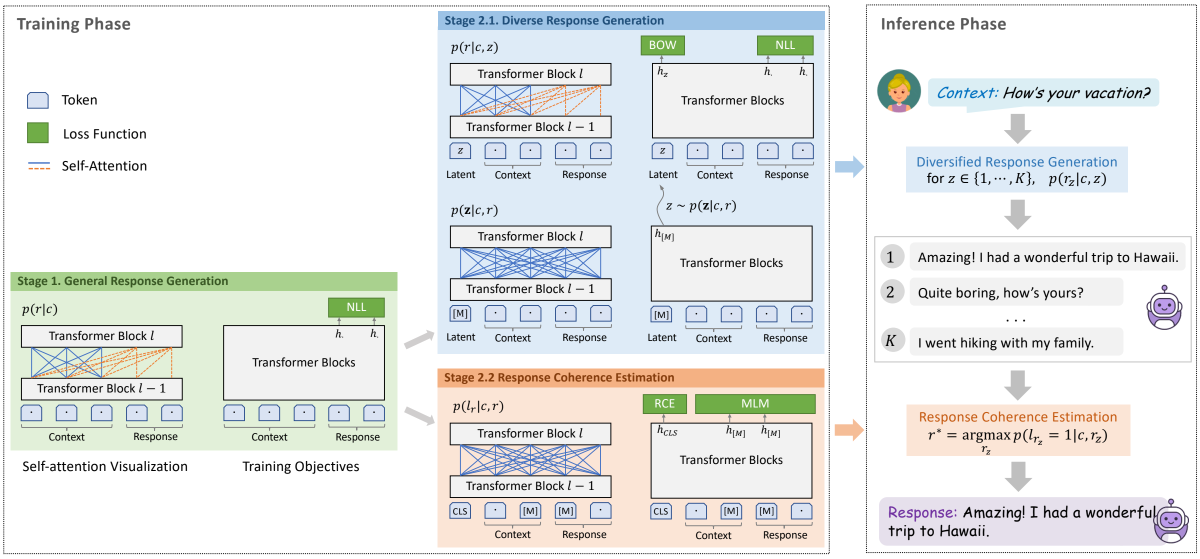

Context: How's your vacation?

Amazing! I had a wonderful trip to Hawaii.

Quite boring, how's yours?

I went hiking with my family.

Response: Amazing! I had a wonderful trip to Hawaii.

Figure 2: PLATO-2 illustration. Left: training phase via curriculum learning, model parameters in the second stage are warm started by those trained well in the first stage. Right: toy example to illustrate inference phase.

this coarse-grained model is still highly effective in learning general concepts of response generation.

The curriculum learning continues to the second stage, which contains the training of a **fine-grained** generation model and an evaluation model. The fine-grained generation model explicitly models the one-to-many mapping relationship via latent variables for diverse response generation. To select the most appropriate response, an evaluation model is trained to estimate the bi-directional coherence between the dialogue context and responses. Distinct with multi-task PLATO, the separate design of fine-grained generation and evaluation enables the model to concentrate more on its corresponding task, getting exempt from multi-task disturbance (Standley et al., 2020).

As compared with PLATO, PLATO-2 leverages curriculum learning to learn response generation gradually, from the general concept of one-to-one mapping to the complex concept of one-to-many mapping. With curriculum learning, we successfully scale the model up to billions of parameters, achieving new state-of-the-art results. Besides open-domain chitchat, the models learned in these two stages can also benefit task-oriented conversation and knowledge grounded dialogue respectively, whose effectiveness is verified thoroughly in DSTC9 (Gunasekara et al., 2020).

To sum up, we trained PLATO-2 with different model sizes: 1.6B, 314M and 93M parameters. In addition to the English models, we also trained Chinese models with massive social media conversations. Comprehensive experiments on both English and Chinese datasets demonstrate that PLATO-2 outperforms Meena, Blender and other state-of-the-art models. We have released our English models and source codes at GitHub, hoping to facilitate the research in open-domain dialogue generation.[1]

2 Methodology

The backbone of PLATO-2 is consisted of transformer blocks with pre-normalization (Radford et al., 2019). Distinct with conventional Seq2Seq, there are no separate encoder and decoder networks in our infrastructure. PLATO-2 keeps the unified network for bi-directional context encoding and uni-directional response generation through flexible attention mechanism (Dong et al., 2019).

2.1 Curriculum Learning

In this work, we carry out effective training of PLATO-2 via curriculum learning. As shown in Figure 2, there are two stages involved in the learning process: during stage 1, a coarse-grained baseline model is trained for general response generation under the simplified one-to-one mapping relationship; during stage 2, two models of fine-grained generation and evaluation are further trained for diverse response generation and response coherence estimation respectively.

---

[1] `https://github.com/PaddlePaddle/Knover/tree/develop/projects/PLATO-2`

2.1.1 General Response Generation

It is well known that there exists a one-to-many relationship in open-domain conversations, where a piece of context may have multiple appropriate responses. Since conventional approaches try to fit the one-to-one mapping, they tend to generate generic and dull responses. Whereas, it is still an efficient way to capture the general characteristics of response generation. As such, we first train a coarse-grained baseline model to learn general response generation under the simplified relationship of one-to-one mapping. Given one training sample of context and response (c, r), we need to minimize the following negative log-likelihood (NLL) loss:

$$\begin{aligned}\mathcal{L}_{NLL}^{\text{Baseline}} &= -\mathbb{E}\ \log p(r|c) \\ &= -\mathbb{E}\ \sum\nolimits_{t=1}^{T}\ \log p(r_t|c, r_{<t})\ ,\end{aligned} \tag{1}$$

where T is the length of the target response r and $r_{<t}$ denotes previously generated words. Since the response generation is a uni-directional decoding process, each token in the response only attends to those before it, shown as dashed orange lines in Figure 2. As for the context, bi-directional attention is enabled for better natural language understanding, shown as blue lines in Figure 2.

2.1.2 Diverse Response Generation

Based upon the coarse-grained baseline model, diverse response generation is warm started and further trained under the relationship of one-to-many mapping. Following the previous work PLATO, the discrete latent variable z is introduced for the one-to-many relationship modeling. z is one K-way categorical variable, with each value corresponding to a particular latent speech act in the response. The model will first estimate the latent act distribution of the training sample $p(\mathbf{z}|c, r)$ and then generate the response with the sampled latent variable $p(r|c, z)$. It is notable that these two tasks of response generation and latent act recognition are trained jointly within the shared network. The NLL loss of diverse response generation is defined as:

$$\begin{aligned}\mathcal{L}_{NLL}^{\text{Generation}} &= -\mathbb{E}_{z\sim p(\mathbf{z}|c,r)}\ \log p(r|c, z) \\ &= -\mathbb{E}_{z\sim p(\mathbf{z}|c,r)} \sum_{t=1}^{T}\ \log p(r_t|c, z, r_{<t})\ ,\end{aligned} \tag{2}$$

where z is the latent act sampled from $p(\mathbf{z}|c, r)$. As sampling is not differentiable, we approximate it with Gumbel-Softmax (Jang et al., 2017). The posterior distribution over latent values is estimated through the task of latent act recognition:

$$p(\mathbf{z}|c, r) = \text{softmax}(W_1 h_{[M]} + b_1) \in \mathbb{R}^K\ , \tag{3}$$

where $h_{[M]} \in \mathbb{R}^D$ is the final hidden state of the special mask token [M], $W_1 \in \mathbb{R}^{K\times D}$ and $b_1 \in \mathbb{R}^K$ denote the weight matrices of one fully-connected layer.

To facilitate the training process of discrete latent variables, the bag-of-words (BOW) loss (Zhao et al., 2017) is also employed:

$$\begin{aligned}\mathcal{L}_{BOW}^{\text{Generation}} &= -\mathbb{E}_{z\sim p(\mathbf{z}|c,r)} \sum\nolimits_{t=1}^{T}\ \log p(r_t|c, z) \\ &= -\mathbb{E}_{z\sim p(\mathbf{z}|c,r)} \sum\nolimits_{t=1}^{T}\ \log \frac{e^{f_{r_t}}}{\sum_{v\in V} e^{f_v}}\ ,\end{aligned} \tag{4}$$

where V refers to the whole vocabulary. The function f tries to predict the words within the target response in a non-autoregressive way:

$$f = W_2 h_z + b_2 \in \mathbb{R}^{|V|}\ , \tag{5}$$

where h_z is the final hidden state of the latent variable. f_{r_t} denotes the estimated probability of word r_t. As compared with NLL loss, the BOW loss discards word orders and forces the latent variable to capture the global information of target response.

To sum up, the objective of the fine-grained generation model is to minimize the following integrated loss:

$$\mathcal{L}^{\text{Generation}} = \mathcal{L}_{NLL}^{\text{Generation}} + \mathcal{L}_{BOW}^{\text{Generation}} \tag{6}$$

2.1.3 Response Coherence Estimation

By assigning distinct values to the latent variable, the fine-grained generation model is able to produce multiple high-quality and diverse responses. To select the most appropriate response from these candidates, one straightforward way is to rank them according to $p(z|c)p(r|c, z)$. However, it is widely recognized that the prior distribution $p(\mathbf{z}|c)$ is difficult to estimate and the uniform distribution is not an effective approximation. To this end, we adopt an alternative approach to train an evaluation model in the second stage, estimating the coherence between each response and the given dialogue context. The loss of response coherence estimation (RCE) is defined as follows:

$$\begin{aligned}\mathcal{L}_{RCE}^{\text{Evaluation}} = &- \log p(l_r = 1|c, r) \\ &- \log p(l_{r^-} = 0|c, r^-)\end{aligned} \tag{7}$$

The positive training samples come from the dialogue context and corresponding target response (c, r), with coherence label $l_r = 1$. And the negative samples are created by randomly selecting responses from the corpus (c, r^-), with coherence label $l_{r^-} = 0$.

In addition to our coherence evaluation function $p(l_r|c, r)$, there are two other functions widely used for response selection. One is the length-average log-likelihood (Adiwardana et al., 2020), which considers the forward response generation probability $p(r|c)$. The other one is the maximum mutual information (Zhang et al., 2020), which considers the backward context recovery probability $p(c|r)$. However, the forward score favors safe and generic responses due to the property of maximum likelihood, while the backward score tends to select the response with a high overlap with the context, resulting in repetitive conversations. By contrast, the discriminative function $p(l_r|c, r)$ considers the bi-directional information flow between the dialogue context and response. Our coherence evaluation is able to ameliorate the aforementioned problems, whose effectiveness is verified in the experiments.

To maintain the capacity of distributed representation, the task of masked language model (MLM) (Devlin et al., 2019) is also included in the evaluation network. Within this task, 15% of the input tokens will be masked at random and the network needs to recover the masked ones. The MLM loss is defined as:

$$\mathcal{L}^{\text{Evaluation}}_{MLM} = -\mathbb{E} \sum\nolimits_{m \in M} \log p(x_m | x_{\backslash M}), \quad (8)$$

where x refers to the input tokens of context and response. $\{x_m\}_{m \in M}$ stands for masked tokens and $x_{\backslash M}$ denotes the rest unmasked ones.

To sum up, the objective of the evaluation model is to minimize the following integrated loss:

$$\mathcal{L}^{\text{Evaluation}} = \mathcal{L}^{\text{Evaluation}}_{RCE} + \mathcal{L}^{\text{Evaluation}}_{MLM} \quad (9)$$

2.2 Inference

For open-domain chitchat, the inference is carried out with the second stage's models as follows.

1) Diverse response generation. Conditioned on each latent value $z \in \{1, \cdots, K\}$, its corresponding candidate response r_z is produced by the fine-grained generation model $p(r_z|c, z)$.
2) Response coherence estimation. The evaluation model will preform ranking and select the one with highest coherence value as the final response $r^* = \text{argmax}_{r_z}\, p(l_{r_z} = 1|c, r_z)$.

3 Experiments

3.1 Training Data

PLATO-2 has English and Chinese models, with training data extracted from open-domain social media conversations. The English training data is extracted from Reddit comments, which are collected by a third party and made publicly available on pushshift.io (Baumgartner et al., 2020). To improve the generation quality, we carry out elaborate data cleaning, as discussed in the Appendix. After filtering, the data is split into training and validation sets in chronological order. The training set contains 684M (context, response) samples, ranging from December 2005 to July 2019. For the validation set, 0.2M samples are selected from the rest data after July 2019. The English vocabulary contains 8K BPE tokens (Sennrich et al., 2016), constructed with the SentencePiece library.

The Chinese training data is collected from public domain social medias. After filtering, there are 1.2B (context, response) samples in the training set, 0.1M samples in the validation set, and 0.1M samples in the test set. As for the Chinese vocabulary, it contains 30K BPE tokens.

3.2 Training Details

PLATO-2 has three model sizes: a standard version of 1.6B parameters, a small version of 314M parameters, and a tiny version of 93M parameters. Detailed network and training configurations are summarized in the Appendix. The main hyper-parameters used in the training process are listed as follows. The maximum sequence lengths of context and response are all set to 128. K is set to 20 for the discrete latent variable (Bao et al., 2020; Chen et al., 2019). We use Adam (Kingma and Ba, 2015) as the optimizer, with a learning rate scheduler including a linear warmup and an invsqrt decay (Vaswani et al., 2017). To train the large-scale model with a relatively large batch size, we employ gradient checkpointing (Chen et al., 2016) to trade computation for memory. The training was carried out on 64 Nvidia Tesla V100 32G GPU cards. It takes about 3 weeks for 1.6B parameter model to accomplish curriculum learning process.

3.3 Evaluation Settings

3.3.1 Compared Methods

The following methods have been compared in the experiments.

- PLATO (Bao et al., 2020) is trained on the basis of $\text{BERT}_{\text{BASE}}$ using 8.3M Twitter and Reddit conversations (Cho et al., 2014; Zhou et al., 2018; Galley et al., 2019). There are 132M network parameters in this model.
- DialoGPT (Zhang et al., 2020) is trained on the basis of GPT-2 (Radford et al., 2019) using Reddit comments. There are three model sizes: 117M, 345M and 762M. Since the 345M parameter model obtains the best performance in their evaluations, we compare with this version.
- Blender (Roller et al., 2021) is first trained using Reddit comments and then fine-tuned with human annotated conversations – BST (Smith et al., 2020), to help emphasize desirable conversational skills of engagingness, knowledge, empathy and personality. Blender has three model sizes: 90M, 2.7B and 9.4B. Since the 2.7B parameter model obtains the best performance in their evaluations, we compare with this version.
- Meena (Adiwardana et al., 2020) is an open-domain chatbot trained with social media conversations. There are 2.6B network parameters in Meena. Since Meena has not released the model or provided a service interface, it is difficult to perform comprehensive comparison. In the experiments, we include the provided samples in their paper for static evaluation.
- Microsoft XiaoIce (Zhou et al., 2020) is a popular social chatbot in Chinese. The official Weibo platform is used in the evaluation.

For the sake of comprehensive and fair comparisons, different versions of PLATO-2 are included in the experiments.

- PLATO-2 1.6B parameter model is the standard version in English, which is first trained using Reddit comments and then fine-tuned with BST conversations. To measure the effectiveness of PLATO-2, this model will be compared to the state-of-the-art open-domain chatbot Blender.
- PLATO-2 314M parameter model is a small version in English, which is trained with Reddit comments. This model will be compared to DialoGPT, as they have similar model scales.
- PLATO-2 93M parameter model is a tiny version in English, which is trained with Reddit comments. As it is difficult to scale up PLATO, we use this version to compare with PLATO.
- PLATO-2 336M parameter Chinese model[2] will be compared to XiaoIce in the experiments.

3.3.2 Evaluation Metrics

We carry out both automatic and human evaluations in the experiments. In automatic evaluation, to assess the model's capacity on lexical diversity, we use the corpus-level metric of distinct-1/2 (Li et al., 2016a), which is defined as the number of distinct uni- or bi-grams divided by the total number of generated words.

In human evaluation, we employ four utterance-level and dialogue-level metrics, including coherence, informativeness, engagingness and humanness. Three crowd-sourcing workers are asked to score the response/dialogue quality on a scale of [0, 1, 2], with the final score determined through majority voting. The higher score, the better. These criteria are discussed as follows, with scoring details provided in the Appendix.

- Coherence is an utterance-level metric, measuring whether the response is relevant and consistent with the context.
- Informativeness is also an utterance-level metric, evaluating whether the response is informative or not given the context.
- Engagingness is a dialogue-level metric, assessing whether the annotator would like to talk with the speaker for a long conversation.
- Humanness is also a dialogue-level metric, judging whether the speaker is a human being or not.

3.4 Experimental Results

In the experiments, we include both static and interactive evaluations.

3.4.1 Self-Chat Evaluation

Self-chats have been widely used in the evaluation of dialogue systems (Li et al., 2016b; Bao et al., 2019; Roller et al., 2021), where a model plays the role of both partners in the conversation. As compared with human-bot conversations, self-chat logs can be collected efficiently at a cheaper price. As reported in Li et al. (2019), self-chat evaluations exhibit high agreement with the human-bot chat evaluations. In the experiments, we ask the bot to perform self-chats and then invite crowd-sourcing workers to evaluate the dialogue quality.

The way to start the interactive conversation needs special attention. As pointed out by Roller et al. (2021), if starting with 'Hi!', partners tend

---

[2] This model has 24 transformer blocks and 16 attention heads, with the embedding dimension of 1024. As the Chinese vocabulary contains 30K BPE tokens, this model has 22.5M more parameters than the English small model.

Model	Human Evaluation				Automatic Evaluation		Average Length
	Coherence	Informativeness	Engagingness	Humanness	Distinct-1	Distinct-2	
PLATO	0.568	0.564	0.340	0.280	0.042	0.255	34.961
PLATO-2 93M	0.688	0.672	0.640	0.560	0.047	0.276	18.292
DialoGPT	0.720	0.712	0.340	0.100	0.150	0.508	9.335
PLATO-2 314M	1.572	1.620	1.300	1.160	0.065	0.435	22.732
Blender	1.856	1.816	1.820	1.540	0.117	0.385	16.873
PLATO-2 1.6B	**1.920**	**1.892**	**1.840**	**1.740**	**0.169**	**0.613**	15.736

Table 1: Self-chat evaluation results, with best value written in bold.

Model	Human Evaluation				Automatic Evaluation		Average Length
	Coherence	Informativeness	Engagingness	Humanness	Distinct-1	Distinct-2	
Microsoft XiaoIce	0.869	0.822	0.560	0.260	**0.289**	**0.764**	6.979
PLATO-2 336M Chinese	**1.737**	**1.683**	**1.600**	**1.480**	0.212	0.713	6.641

Table 2: Chinese interactive evaluation results, with best value written in bold.

to greet with each other and only cover some shallow topics in the short conversation. Therefore, to expose the model's weaknesses and explore the model's limits, we choose to start the interactive conversation with pre-selected topics. We use the classical 200 questions as the start topic (Vinyals and Le, 2015) and ask the bot to performance self-chats given the context. There are 10 utterances in each dialogue, including the input start utterance. We carry out automatic evaluation on the 200 self-chat logs and randomly select 50 conversations for human evaluation.

The compared models are divided into three groups. The first group includes PLATO 132M model and PLATO-2 93M model. Both of them have similar model scales. The second group includes DialoGPT 345M model and PLATO-2 310M model. Both of them are trained using Reddit comments and have similar model scales. The third group includes Blender 2.7B model and PLATO-2 1.6B model. Both of them are first trained using Reddit comments and further fine-tuned with BST conversations. In human evaluation, two self-chat logs, which are from the same group and have the same start topic, will be displayed to three annotators. One example is given in Figure 3. As suggested in ACUTE-Eval (Li et al., 2019), we ask crowd-sourcing workers to pay attention to only one speaker within a dialogue. In the evaluation, they need to give scores on coherence and informativeness for each P1's utterance, and assess P1's overall quality on engagingness and humanness.

The self-chat evaluation results are summarized in Table 1. These results indicate that PLATO-2 1.6B model obtains the best performance across human and automatic evaluations. In the first group, PLATO-2 achieves better performance than PLATO on a similar model scale, which might mainly result from the stable curriculum learning and large-scale conversation data. In the second group, DialoGPT tends to generate repetitive conversations due to the backward scoring function, resulting in poor performance in interactive evaluation. In the third group, PLATO-2 outperforms the state-of-the-art open-domain chatbot Blender. The gap of Blender and PLATO-2 on the corpus-level metric distinct-1/2 suggests that PLATO-2 has a better capacity on lexical diversity. In addition, the difference among these three groups suggests that enlarging model scales and exploiting human annotated conversations help improve the dialogue quality.

3.4.2 Human-Bot Chat Evaluation

In the Chinese evaluation, it is difficult to carry out self-chats for Microsoft XiaoIce, as there is no public available API. Therefore, we collect human-bot conversations through their official Weibo platform. The interactive conversation also starts with a pre-selected topic and continues for 7-14 rounds. 50 diverse topics are extracted from the high-frequency topics of a commercial chatbot, including travel, movie, hobby and so on. The collected human-bot conversations are distributed to crowd-sourcing

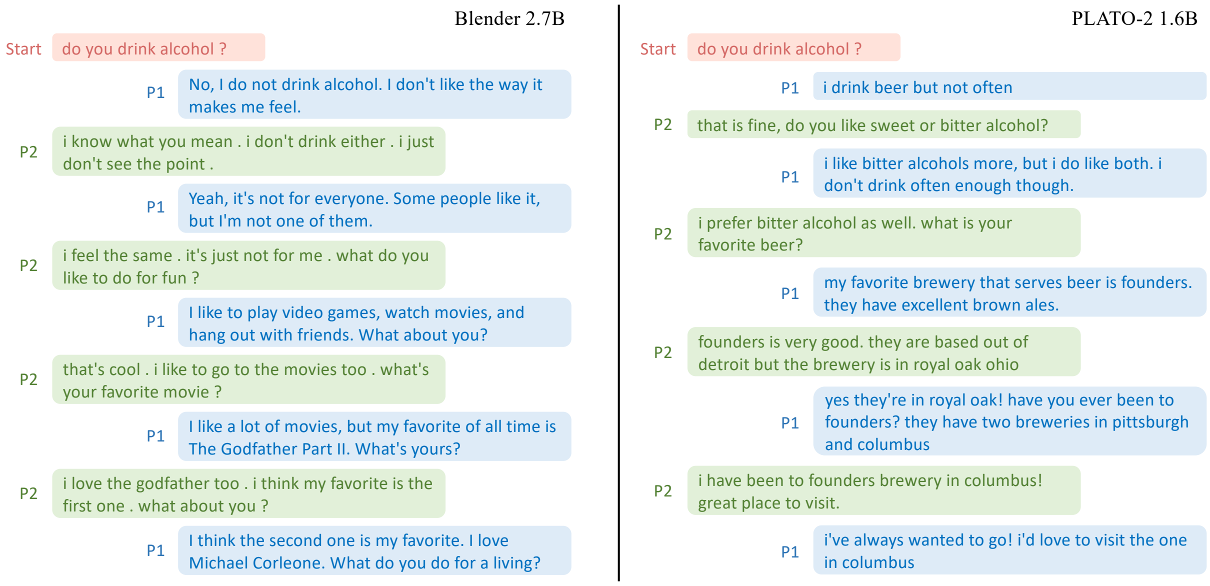

Blender 2.7B

Start: do you drink alcohol ?

P1: No, I do not drink alcohol. I don't like the way it makes me feel.

P2: i know what you mean . i don't drink either . i just don't see the point .

P1: Yeah, it's not for everyone. Some people like it, but I'm not one of them.

P2: i feel the same . it's just not for me . what do you like to do for fun ?

P1: I like to play video games, watch movies, and hang out with friends. What about you?

P2: that's cool . i like to go to the movies too . what's your favorite movie ?

P1: I like a lot of movies, but my favorite of all time is The Godfather Part II. What's yours?

P2: i love the godfather too . i think my favorite is the first one . what about you ?

P1: I think the second one is my favorite. I love Michael Corleone. What do you do for a living?

PLATO-2 1.6B

Start: do you drink alcohol ?

P1: i drink beer but not often

P2: that is fine, do you like sweet or bitter alcohol?

P1: i like bitter alcohols more, but i do like both. i don't drink often enough though.

P2: i prefer bitter alcohol as well. what is your favorite beer?

P1: my favorite brewery that serves beer is founders. they have excellent brown ales.

P2: founders is very good. they are based out of detroit but the brewery is in royal oak ohio

P1: yes they're in royal oak! have you ever been to founders? they have two breweries in pittsburgh and columbus

P2: i have been to founders brewery in columbus! great place to visit.

P1: i've always wanted to go! i'd love to visit the one in columbus

Figure 3: Self-chat examples by Blender and PLATO-2.

workers for evaluation. The human and automatic evaluation results are summarized in Table 2. XiaoIce obtains higher distinct values, which may use a retrieval-based strategy in response generation. The human evaluations indicate that our PLATO-2 model achieves significant improvements over XiaoIce across all the human evaluation metrics.

3.4.3 Static Evaluation

Besides the interactive evaluation, we also employ static evaluation to analyze the model's performance. In static evaluation, each model will produce a response towards the given multi-turn context. Those powerful models are involved in the evaluation: Meena, Blender, DialoGPT and PLATO-2 1.6B. To compare with Meena, we include their provided 60 static samples in the Appendix of the paper and generate corresponding responses with other models. We also include 60 test samples about daily life from Daily Dialog (Li et al., 2017) and 60 test samples about in-depth discussion from Reddit. Given that the measurement of humanness usually needs multi-turn interaction, this metric is excluded from static evaluation. The evaluation results are summarized in Table 3. It can be observed that PLATO-2 is able to produce coherent, informative and engaging responses across different chat scenarios. The average Fleiss's kappa (Fleiss, 1971) of human evaluation is 0.466, indicating annotators have reached moderate agreement.

Model	Meena Samples		
	Coherence	Informativeness	Engagingness
Meena	1.750	1.617	1.583
DialoGPT	1.233	1.067	1.017
Blender	1.800	1.767	1.683
PLATO-2 1.6B	**1.900**	**1.917**	**1.850**

Model	Daily Dialog Samples		
	Coherence	Informativeness	Engagingness
DialoGPT	1.117	1.033	0.917
Blender	1.767	1.617	1.633
PLATO-2 1.6B	**1.867**	**1.850**	**1.833**

Model	Reddit Samples		
	Coherence	Informativeness	Engagingness
DialoGPT	1.283	1.283	1.183
Blender	1.767	1.550	1.583
PLATO-2 1.6B	**1.900**	**1.900**	**1.883**

Table 3: Static evaluation results, with the best scores written in bold.

3.5 Discussions

3.5.1 Case Analysis

To further analyze the models' features, two self-chat examples of Blender and PLATO-2 are provided in Figure 3. Although both models are able to produce high-quality engaging conversations, they exhibit distinct discourse styles. Blender tends to switch topics quickly in the short conversation, including alcohol, hobbies, movies and work. The emergence of this style might be related with BST

	Blender Win	Tie	PLATO-2 Win
In-depth Discussion	8	18	**24**

Table 4: In-depth discussion w.r.t. the start topic.

	PLATO-2 Stage-1 Win	Tie	PLATO-2 Stage-2 Win
Engagingness	3	31	**16**
Humanness	6	29	**15**

Table 5: Comparison of the models in PLATO-2.

fine-tuning data. For instance, persona chat in BST is about the exchange of personal information between two partners, where topics need to switch quickly to know more about each other. Due to the task settings of data collection, some human annotated conversations might be a little unnatural. Nevertheless, fine-tuning with BST conversations is essential to mitigate undesirable toxic traits of large corpora and emphasize desirable skills of human conversations.

Distinct with Blender, PLATO-2 can stick to the start topic and conduct in-depth discussions. The reasons might be two-fold. First, our model is able to generate diverse and informative responses with the accurate modeling of one-to-many relationship. Second, the evaluation model helps select the coherent response and stick to current topic. We asked crowd-sourcing workers to annotate which model's in-depth discussion is better w.r.t. the start topic. The comparison result is shown in Table 4, which also verifies our above analysis on discourse styles.

3.5.2 Why PLATO-2 Performs Better?

Why PLATO-2 achieves better performance as compared with Meena, Blender and other state-of-the-art models? As analyzed above, major reasons might come from two aspects: fine-grained generation and evaluation. First, PLATO-2 employs discrete latent variable for the one-to-many relationship modeling, which is able to generate high-quality and diverse responses. Second, the evaluation model in PLATO-2 is effective at selecting the most appropriate response from the candidates.

In fact, these two aspects are associated with the curriculum learning in the second stage, modeling the one-to-many relationship for open-domain conversations. By contrast, Meena and Blender are learned under the one-to-one mapping relationship, similar to the first stage in PLATO-2. To dissect the effects of these two stage models, we further ask crowd-sourcing workers to evaluate the models' self-chat logs on the dialogue-level metrics. The comparison results are summarized in Table 5. These results verify the effectiveness of curriculum learning in PLATO-2.

3.5.3 Further Exploration of PLATO-2

In addition to open-domain chitchat, there are two other kinds of dialogues in conversational AI (Gao et al., 2018): knowledge grounded dialogue, and task-oriented conversation. Similar to open-domain conversation, the one-to-many mapping relationship also exists in knowledge grounded dialogue (Kim et al., 2020): given a dialogue context, multiple pieces of knowledge might be applicable for the response generation. Therefore, the one-to-many mapping models of the second stage can also be adapted for knowledge grounded dialogue. By expanding the network input with the knowledge segment, the background knowledge is encoded and grounded for response generation. Distinct from the open-domain conversation and knowledge grounded dialogue, task-oriented conversations usually need to accomplish a specific goal. Accordingly, the conversation flow would become less diverse and concentrated on task completion. Therefore, the one-to-one mapping generation model of the first stage can be used for the end-to-end task-oriented conversation.

For the exploration of PLATO-2 two-stage framework, we participated in several tasks of DSTC9 (Gunasekara et al., 2020), including interactive evaluation of open-domain conversation (Track3-task2), static evaluation of knowledge grounded dialogue (Track3-task1), and end-to-end task-oriented conversation (Track2-task1). PLATO-2 has achieved the first place in all three tasks (Bao et al., 2021). To sum up, the benefits brought by the two-stage curriculum learning in PLATO-2 are two-fold. Firstly, given the difficulties to scale up PLATO, the two-stage curriculum learning is an essential ingredient for the successful training of 1.6B parameter PLATO-2. Secondly, the two-stage PLATO-2 adapts well to multiple conversational tasks, indicating its potentials as a unified pre-training framework for conversational AI.

4 Related Work

Related works include large-scale language models and open-domain dialogue generation.

Large-scale Language Models. Pre-trained large-scale language models have brought many breakthroughs on various NLP tasks. GPT (Radford et al., 2018) and BERT (Devlin et al., 2019) are representative uni-directional and bi-directional language models, trained on general text corpora. By introducing pre-normalization and modifying weight initialization, GPT-2 (Radford et al., 2019) successfully extends the model scale from 117M to 1.5B parameters. To cope with memory constraints, Megatron-LM (Shoeybi et al., 2019) exploits model parallelism to train an 8.3B parameter model on 512 GPUs. GPT-3 (Brown et al., 2020) further trains an 175B parameter autoregressive language model, demonstrating strong performance on many NLP tasks. The development of large-scale language models is also beneficial to the task of dialogue generation.

Open-domain Dialogue Generation. On the basis of GPT-2, DialoGPT (Zhang et al., 2020) is trained for response generation using Reddit comments. To obtain a human-like open-domain chatbot, Meena (Adiwardana et al., 2020) scales up the network parameters to 2.6B and utilizes more social media conversations in the training process. To emphasize desirable conversational skills of engagingness, knowledge, empathy and personality, Blender (Roller et al., 2021) further fine-tunes the pre-trained model with human annotated conversations. In addition to the attempts on model scale and data selection, PLATO introduces discrete latent variable to tackle the inherent one-to-many mapping problem to improve response quality. In this work, we explore the effective training of PLATO-2 via curriculum learning.

5 Conclusion

In this work, we discuss the effective training of open-domain chatbot PLATO-2 via curriculum learning, where two stages are involved. In the first stage, one coarse-grained model is trained for general response generation. In the second stage, two models of fine-grained generation and evaluation are trained for diverse response generation and response coherence estimation. Experimental results demonstrate that PLATO-2 achieves substantial improvements over the state-of-the-art methods in both Chinese and English evaluations.

Acknowledgments

We would like to thank Jingzhou He, and Tingting Li for the help on resource coordination; Daxiang Dong, and Pingshuo Ma for the support on PaddlePaddle; Yu Sun, Yukun Li, and Han Zhang for the assistance with infrastructure and implementation. This work was supported by the Natural Key Research and Development Project of China (No. 2018AAA0101900).

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Siqi Bao, Bingjin Chen, Huang He, Xin Tian, Han Zhou, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, and Yingzhan Lin. 2021. A unified pre-training framework for conversational ai. *arXiv preprint arXiv:2105.02482*.

Siqi Bao, Huang He, Fan Wang, Rongzhong Lian, and Hua Wu. 2019. Know more about each other: Evolving dialogue strategy via compound assessment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5382–5391.

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. Plato: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 830–839.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901.

Chaotao Chen, Jinhua Peng, Fan Wang, Jun Xu, and Hua Wu. 2019. Generating multiple diverse responses with multi-mapping and posterior mapping selection. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4918–4924.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. In *Psychological Bulletin*, pages 378–382.

Michel Galley, Chris Brockett, Xiang Gao, Jianfeng Gao, and Bill Dolan. 2019. Grounded response generation task at dstc7. In *AAAI Dialog System Technology Challenge Workshop*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational AI. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 2–7.

Chulaka Gunasekara, Seokhwan Kim, Luis Fernando D'Haro, Abhinav Rastogi, Yun-Nung Chen, Mihail Eric, Behnam Hedayatnia, Karthik Gopalakrishnan, Yang Liu, Chao-Wei Huang, Dilek Hakkani-Tür, Jinchao Li, Qi Zhu, Lingxiao Luo, Lars Liden, Kaili Huang, Shahin Shayandeh, Runze Liang, Baolin Peng, Zheng Zhang, Swadheen Shukla, Minlie Huang, Jianfeng Gao, Shikib Mehri, Yulan Feng, Carla Gordon, Seyed Hossein Alavi, David Traum, Maxine Eskenazi, Ahmad Beirami, Eunjoon, Cho, Paul A. Crook, Ankita De, Alborz Geramifard, Satwik Kottur, Seungwhan Moon, Shivani Poddar, and Rajen Subba. 2020. Overview of the ninth dialog system technology challenge: Dstc9. *arXiv preprint arXiv:2011.06486*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020. Sequential latent knowledge selection for knowledge-grounded dialogue. In *International Conference on Learning Representations*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved dialogue evaluation with optimized questions and multi-turn comparisons. *arXiv preprint arXiv:1909.03087*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, pages 986–995.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Technical report, OpenAI*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2021–2030.

Trevor Standley, Amir R Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning*, pages 9120–9132.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 654–664.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4623–4629.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of XiaoIce, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93.

A Data Cleaning Process

PLATO-2 has English and Chinese models, with training data extracted from open-domain social media conversations. As the comments are formatted in message trees, any conversation path from the root to a tree node can be treated as one training sample, with the node as response and its former turns as context. To improve the generation quality, we carry out elaborate data cleaning. A message node and its sub-trees will be removed if any of the following conditions is met.

1) The number of BPE tokens is more than 128 or less than 2.
2) Any word has more than 30 characters or the message has more than 1024 characters.
3) The percentage of alphabetic characters is less than 70%.
4) The message contains URL.
5) The message contains special strings, such as r/, u/, &amp.
6) The message has a high overlap with the parent's text.
7) The message is repeated more than 100 times.
8) The message contains offensive words.
9) The subreddit is quarantined.
10) The author is a known bot.

After data cleaning, the English training data contains 684M (context, response) samples and the Chinese training data contains 1.2B (context, response) samples. Each English/Chinese sample has 2.78/2.82 utterances and each utterance has 26.29/22.20 tokens on average.

B Training Configurations

PLATO-2 has three model sizes: a standard version of 1.6B parameters, a small version of 314M parameters, and a tiny version of 93M parameters. The 1.6B parameter model has 32 transformer blocks and 32 attention heads, with the embedding dimension of 2048. The 314M parameter model has 24 transformer blocks and 16 attention heads, with the embedding dimension of 1024. The 93M parameter model has 12 transformer blocks and 12 attention heads, with the embedding dimension of 768. The training configurations of PLATO-2 1.6B are provided in Table 6. The training was carried out on 64 Nvidia Tesla V100 32G GPU cards. It takes about 3 weeks for the 1.6B parameter model to accomplish the curriculum learning process.

C Chinese Case Analysis

We also provide two human-bot chat examples of XiaoIce and PLATO-2 in Figure 4, with original interactive logs shown on the left and translated logs on the right. It can be observed that some responses produced by XiaoIce are not coherent with the contexts and there are some abrupt changes of topics. By contrast, the interaction with PLATO-2 is more coherent and engaging.

D Scoring Criteria in Human Evaluation

The criteria used in human evaluation are provided in Table 7.

Model	Stage	Batch Size	Learning Rate	Warmup Steps	Training Steps
PLATO-2 1.6B Parameter	1. Coarse-grained Generation	524,288	5e-4	3125	20w
	2.1. Fine-grained Generation	65,536	1e-5	3125	10w
	2.2. Evaluation	524,288	5e-5	3125	10w

Table 6: Training configurations of PLATO-2.

Microsoft XiaoIce

Start 你会游泳吗
Bot 特别会游
Human 这么厉害，我不会
Bot 没事 游泳圈 要不我教你
Human 好呀，你怎么学会的
Bot 哪里有教学游泳的么
Human 你教我呀
Bot 我会游泳
Human 你真厉害，我怕水
Bot 我教你
Human 好，我们约个时间一起游泳
Bot 好的啊，一起吃饭
Human 先游泳再吃饭
Bot 吃饭了
Human 要吃啥
Bot 吃代码
Human 厉害，我也在写代码
Bot 问你一下，在网页上如何加入QQ既时聊天的代码?
Human 不知道
Bot 给我写一份 c 代码

Translated

Start Can you swim?
Bot Yes, I'm very good at swimming.
Human Awesome. I can't swim.
Bot Don't worry. Swim ring. Or I can teach you.
Human Great. How did you learn it?
Bot Is there any swimming course?
Human You can teach me.
Bot I can swim.
Human Cool. I am afraid of water.
Bot I can teach you.
Human Good. Let us find a time to go swimming together.
Bot All right. Let us have dinner together.
Human Swimming first, then dinner.
Bot It is time for dinner.
Human What are you going to eat?
Bot Code
Human Cool, I am coding too.
Bot A quick question, do you know how to insert the code of instant chat into the webpage?
Human I don't know.
Bot Write me c++ code

PLATO-2 333M Chinese

Start 你会游泳吗
Bot 会游一点
Human 那你能教我么
Bot 我怕把你教的更不会游
Human 我很怕水，不敢游
Bot 那我就更不敢教你
Human 你怎么学会的
Bot 小时候和小伙伴去河里学的
Human 那还挺好，我也打算学学
Bot 去吧去吧,学会了带我一起
Human 好呀，到时候我们一起去游泳
Bot 你不怕我把你扔河里么
Human 我会游泳了就不怕
Bot 那你要记得带着泳圈哦
Human 嗯嗯，好的
Bot 记得把你自己也带着

Translated

Start Can you swim?
Bot Yes, but not so good at it.
Human Could you teach me?
Bot I am afraid I am not a good teacher.
Human I am afraid of water.
Bot Then I am sure I can't teach you.
Human How did you learn it?
Bot I learnt it in the river as a kid, with a bunch of friends.
Human That is great. I would like to learn swimming too.
Bot Go for it. We can go swimming together then.
Human Good, let us go swimming together.
Bot Aren't you afraid I will throw you into the river?
Human I won't if I can swim.
Bot Remember to bring a swim ring.
Human All right.
Bot Bring yourself as well, lol

Figure 4: Human-bot chat examples by Microsoft XiaoIce and PLATO-2.

Score	Coherence
0	• The response is not related with the context. • The response simply repeats the context. • The response has obvious conflicts with the context. • There are serious logic conflicts within the response.
1	• The response has minor conflicts with the context. • There are some minor logic conflicts in the response.
2	• The response is coherent with the context.

Score	Informativeness
0	• The response doesn't contain any information. • This response just repeats the context and fails to bring any additional information. • The information is invalid, as the coherence score is 0.
1	• The information has conflicts with common sense. • There are factual errors in the response.
2	• The response has appropriate and correct information.

Score	Engagingness
0	• I don't want to talk with this speaker.
1	• It is kind of boring, but it is still ok to talk with this speaker.
2	• I would like to talk with this speaker for a long conversation.

Score	Humanness
0	• This speaker seems like a bot.
1	• This speaker gives unnatural responses occasionally and seems not that human-like.
2	• This speaker seems like a human being.

Table 7: Score details of four metrics in human evaluation.

E Response Selection Comparison

We carry out more experiments to compare the performance of distinct scoring functions in response selection. Firstly, one Chinese response selection dataset is constructed: 100 dialogue contexts are selected from the test set and 10 candidate responses are retrieved for each context with a commercial chatbot. Secondly, we ask crowd-sourcing workers to annotate the label whether the candidate response is coherent with the context. Thirdly, we train three 336M parameter models as the scoring function, including the forward response generation probability $p(r|c)$, the backward context recover probability $p(c|r)$ and the bi-directional coherence probability $p(l_r|c,r)$. Their results on the annotated response selection dataset are summarized in Table 8. The metrics of mean average precision (MAP), mean reciprocal rank (MRR) and precision at position 1 (P@1) are employed. These results indicate that PLATO-2's evaluation model is better at selecting appropriate responses.

Score Function	MAP	MRR	P@1
$p(r\|c)$	0.705	0.790	0.700
$p(c\|r)$	0.672	0.737	0.610
$p(l_r\|c,r)$	**0.754**	**0.819**	**0.750**

Table 8: Comparison of different score functions in response selection, with best value written in bold.

JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs

Pei Ke[1], **Haozhe Ji**[1], **Yu Ran**[2], **Xin Cui**[2], **Liwei Wang**[3], **Linfeng Song**[4],
Xiaoyan Zhu[1], **Minlie Huang**[1]*
[1]The CoAI group, Department of Computer Science and Technology,
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing 100084, China
[2]Sogou Inc., Beijing, China [3]The Chinese University of Hong Kong [4]Tencent AI Lab
`{kp17,jhz20}@mails.tsinghua.edu.cn, {zxy-dcs,aihuang}@tsinghua.edu.cn`

Abstract

Existing pre-trained models for knowledge-graph-to-text (KG-to-text) generation simply fine-tune text-to-text pre-trained models such as BART or T5 on KG-to-text datasets, which largely ignore the graph structure during encoding and lack elaborate pre-training tasks to explicitly model graph-text alignments. To tackle these problems, we propose a graph-text joint representation learning model called JointGT. During encoding, we devise a structure-aware semantic aggregation module which is plugged into each Transformer layer to preserve the graph structure. Furthermore, we propose three new pre-training tasks to explicitly enhance the graph-text alignment including respective text / graph reconstruction, and graph-text alignment in the embedding space via Optimal Transport. Experiments show that JointGT obtains new state-of-the-art performance on various KG-to-text datasets[1].

1 Introduction

Knowledge-graph-to-text (KG-to-text) generation aims to generate high-quality texts which are consistent with input graphs (Gardent et al., 2017). This task requires to simultaneously encode the graph structure and the content, and effectively leverage the input graphs in the decoding process (Zhao et al., 2020). As a major natural language generation (NLG) task that connects knowledge graphs and texts, this task can further promote the applicability of knowledge graphs in more realistic NLG scenarios, such as knowledge-grounded dialogue generation (Zhou et al., 2018a) and story generation (Guan et al., 2019; Ji et al., 2020).

Due to the limited amount of graph-text parallel data, it's hard for typical neural text generation models to learn the alignments between source entities / relations and target tokens from scratch (Guo et al., 2020; Fu et al., 2020). Recent work resorts to constructing general-purpose pre-trained language models for KG-to-text generation. The most common and simple way is to linearize input graphs into text sequences, and directly fine-tune text-to-text Transformer-based pre-trained models like GPT (Radford et al., 2018, 2019), BART (Lewis et al., 2020) or T5 (Raffel et al., 2020) on KG-to-text datasets (Ribeiro et al., 2020a; Kale and Rastogi, 2020). Benefiting from self-supervised pre-training on large-scale unlabelled text corpora, pre-trained language models can generate high-quality texts via simply fine-tuning, and outperform other models with sophisticated structures.

Despite the superior performance of fine-tuning pre-trained models on KG-to-text datasets, we argue that building pre-trained models for KG-to-text generation still faces two major challenges: 1) **Structural information loss during encoding**. Most of the existing pre-trained models capture contextual information via bidirectional Transformers (Devlin et al., 2019), which include full attention connections. This model structure may neglect the structural information when encoding knowledge graphs since the relation between each pair of input entities is not explicitly considered (Zhu et al., 2019). 2) **Absence of explicit graph-text alignments**. Existing work on pre-trained models for text generation commonly adopts auto-encoding or auto-regressive text reconstruction to learn text-text alignments, which encodes the corrupted text sequence and decodes the original sequence (Lewis et al., 2020; Raffel et al., 2020). Since knowledge graphs may possess more complex structures than text sequences, it's hard to explicitly learn graph-text alignments by directly using the pre-training tasks based on text reconstruction.

Thus, we propose a graph-text joint represen-

---

* Corresponding author

[1] The data, codes, and model parameters are available at `https://github.com/thu-coai/JointGT`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2526–2538
August 1–6, 2021.

tation learning framework called *JointGT* to deal with the above challenges. **Firstly**, to alleviate the structural information loss during encoding, we devise a simple structure-aware semantic aggregation module at each Transformer layer to aggregate contextual information following the graph structure. **Secondly**, we propose three pre-training tasks including graph enhanced text reconstruction, text enhanced graph reconstruction, and graph-text embedding alignment to explicitly build the connection between knowledge graphs and text sequences. The first two tasks are expected to enhance the graph-text alignment in the discrete vocabulary space, where our model is required to predict the masked information of graphs / texts based on the observed information of texts / graphs. And the third task is designed to model the graph-text alignment in the continuous embedding space via Optimal Transport (Peyré and Cuturi, 2019) to match the hidden representations of graphs and texts. Our contributions are as follows:

- We propose a novel pre-trained model called JointGT for KG-to-text generation tasks. This model adopts a structure-aware semantic aggregation module to model the structure of an input graph at each Transformer layer, and utilizes three pre-training tasks to explicitly learn graph-text alignments in the discrete and continuous spaces.
- We conduct experiments on the datasets of KG-to-text generation including WebNLG, WebQuestions and PathQuestions. Results show that JointGT achieves new state-of-the-art performance on KG-to-text generation.

2 Related Work

KG-to-Text Generation

Recent studies on KG-to-text generation tasks mainly fall into three aspects: 1) *Encoder modification*: To alleviate the structural information loss of sequence encoders with the input of linearized graphs (Gardent et al., 2017; Trisedya et al., 2018; Moryossef et al., 2019), researchers focus on more complex encoder structures for better graph representations, such as graph neural networks (Marcheggiani and Perez-Beltrachini, 2018; Ribeiro et al., 2020b) and graph Transformers (Koncel-Kedziorski et al., 2019; Schmitt et al., 2020a). 2) *Unsupervised training*: researchers devise unsupervised training objectives to jointly learn the tasks of graph-to-text and text-to-graph conversion with non-parallel graph-text data (Schmitt et al., 2020b; Guo et al., 2020; Jin et al., 2020). 3) *Building pre-trained models*: With the development of pre-trained NLG models such as GPT (Radford et al., 2018, 2019), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), recent work directly fine-tunes these models on graph-to-text datasets and reports impressive performance (Ribeiro et al., 2020a; Kale and Rastogi, 2020; Chen et al., 2020b; Mager et al., 2020).

Compared with the existing work on pre-trained models for KG-to-text generation, our model utilizes pre-training methods to explicitly learn graph-text alignments instead of directly fine-tuning text-to-text pre-trained models on KG-to-text datasets.

KG-Enhanced Pre-Trained Models

Another line of related studies is pre-trained models enhanced by knowledge graphs for natural language understanding (NLU). The motivation of these models is to incorporate knowledge graphs into pre-trained models to facilitate the understanding of entities and relations in natural language. Early work including ERNIE (Zhang et al., 2019) and KnowBERT (Peters et al., 2019) directly uses fixed entity embeddings based on TransE (Bordes et al., 2013) or word vectors (Mikolov et al., 2013) during pre-training. Recent work like KEPLER (Wang et al., 2021) and JAKET (Yu et al., 2020) resorts to jointly pre-training graph-text representations. Specifically, they encode the textual descriptions of entities with pre-trained language models as entity embeddings and jointly optimize the knowledge embedding objective and the masked language modeling objective.

In comparison, our model focuses on joint pre-training methods on knowledge graph encoding and sequence decoding in KG-to-text generation tasks, rather than considering graph-text joint encoding methods in NLU tasks.

3 Method

3.1 Task Definition and Model Overview

Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{e_1, e_2, \cdots, e_{|\mathcal{V}|}\}$ denotes the entity set and $\mathcal{E} = (r_{ij})_{|\mathcal{V}|\times|\mathcal{V}|}$ indicates the relations connecting the entities, and its linearized version $\mathcal{G}_{linear} = (w_1, w_2, \cdots, w_m)$ which consists of m tokens, our goal is to generate a text sequence $X = (x_1, x_2, \cdots, x_n)$ which is consistent with the input graph.

Our model is built on pre-trained encoder-decoder models like BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). First of all, we follow the existing work (Chen et al., 2020b) to linearize knowledge graphs in the form of triple lists (as shown in Figure 1), and devise a simple structure-aware semantic aggregation module which is plugged into each Transformer layer of the encoder to preserve the structural information of input graphs (§3.2). Then, we propose three pre-training tasks including graph / text reconstruction in the discrete vocabulary space and graph-text matching in the continuous embedding space, which enable our model to jointly learn the representations of knowledge graphs and texts (§3.3).

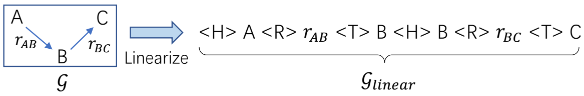

Figure 1: Illustration of linearizing knowledge graphs into text sequences. The special tokens <H>, <R> and <T> mean the head entity, relation and tail entity in the knowledge triples, respectively.

3.2 Model Structure

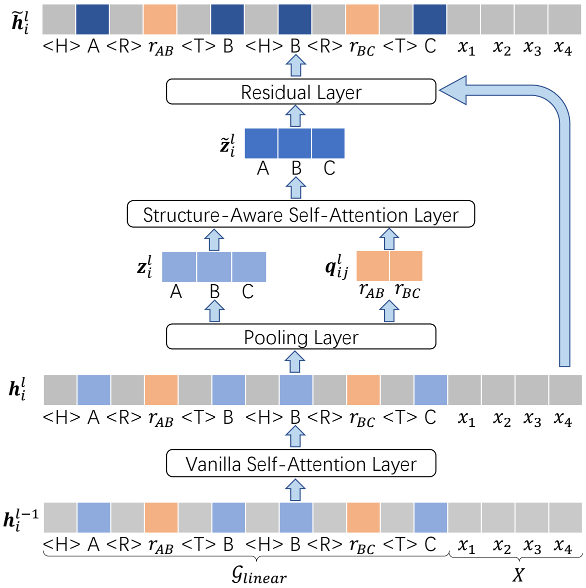

Figure 2: Structure-aware semantic aggregation module at each layer of the Transformer encoder. This module contains a pooling layer to obtain the contextual semantic representations of entities (z_i^l) and relations (q_{ij}^l) from the output of the vanilla self-attention layer (h_i^l), a structure-aware self-attention layer to aggregate the entity representations ($\tilde{z}_i^l$) based on the graph structure, and a residual layer to fuse the contextual and structural representations ($\tilde{h}_i^l$).

To simultaneously leverage the contextual representation from pre-trained models and preserve the structural information, we devise a structure-aware semantic aggregation module in the Transformer encoder. Assume that the input of our encoder during pre-training is the linearized graph $\mathcal{G}_{linear}$ and the corresponding text sequence X (which may be corrupted or empty in some pre-training tasks), the self-attention layer in the l-th Transformer layer can be formulated as follows[2]:

$$\begin{aligned} \boldsymbol{h}_i^l &= \sum_{j=1}^{m+n} \alpha_{ij}^l (\boldsymbol{h}_j^{l-1} \boldsymbol{W}^V) \\ \alpha_{ij}^l &= \frac{\exp(t_{ij}^l)}{\sum_{p=1}^{m+n} \exp(t_{ip}^l)} \\ t_{ij}^l &= \frac{\left(\boldsymbol{h}_i^{l-1} \boldsymbol{W}^Q\right) \left(\boldsymbol{h}_j^{l-1} \boldsymbol{W}^K\right)^\top}{\sqrt{d_k}} \\ & i = 1, 2, \cdots, m+n \end{aligned} \tag{1}$$

where $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ are the model parameters and d_k denotes the dimension of query / key / value vectors. The fully-connected attention captures rich contextual semantic relationship among the entities, relations and the tokens of text sequences, but is not sufficient to encode the structural information of input graphs. Thus, we devise a structure-aware semantic aggregation module on top of vanilla self-attention, as shown in Figure 2. First of all, we utilize a mean pooling layer[3] to obtain the representation of each entity and relation from the output of the vanilla self-attention layer:

$$\begin{aligned} \boldsymbol{z}_i^l &= \text{pooling}(\{\boldsymbol{h}_p^l | p \in \mathcal{P}(e_i), 1 \leq p \leq m\}) \\ \boldsymbol{q}_{ij}^l &= \text{pooling}(\{\boldsymbol{h}_p^l | p \in \mathcal{P}(r_{ij}), 1 \leq p \leq m\}) \\ & i = 1, \cdots, |\mathcal{V}|; \quad j = 1, \cdots, |\mathcal{V}| \end{aligned} \tag{2}$$

where $\mathcal{P}(e_i)/\mathcal{P}(r_{ij})$ means the set of positions occupied by e_i / r_{ij} in the linearized graph. Note that $\boldsymbol{q}_{ij}^l$ will be set to an all-zero vector if there is no relation between e_i and e_j. Then we update entity representations with a structure-aware self-

---

[2] We take a single attention head as an example in this section. In practice, we use our proposed method in the multi-head attention.

[3] We find that there is no significant difference in the model performance between mean pooling and other aggregation functions like max pooling.

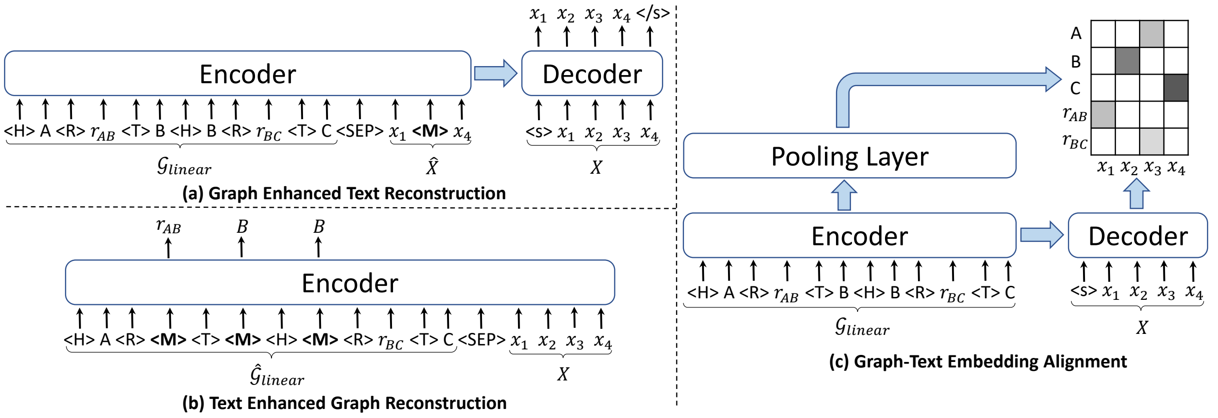

Figure 3: Overview of our proposed pre-training tasks: (a) Graph enhanced text reconstruction: reconstructing the text sequence given the complete graph. (b) Text enhanced graph reconstruction: predicting the masked entities and relations of the corrupted graph conditioned on the complete text. (c) Graph-text embedding alignment: matching the embedding vectors of the knowledge graph and the text via Optimal Transport. The special token <SEP> is to separate the linearized graph and the text, while <M> denotes the placeholder for masked tokens.

attention layer (Shaw et al., 2018):

$$\begin{aligned}\tilde{\boldsymbol{z}}_i^l &= \sum_{j=1}^{|\mathcal{V}|} \beta_{ij}^l (\boldsymbol{z}_j^l \boldsymbol{W}^{VS} + \boldsymbol{q}_{ij}^l \boldsymbol{W}^{VR}) \\ \beta_{ij}^l &= \frac{\exp(u_{ij}^l)}{\sum_{p=1}^{|\mathcal{V}|} \exp(u_{ip}^l)} \qquad (3)\end{aligned}$$

$$u_{ij}^l = \frac{\left(\boldsymbol{z}_i^l \boldsymbol{W}^{QS}\right)\left(\boldsymbol{z}_j^l \boldsymbol{W}^{KS} + \boldsymbol{q}_{ij}^l \boldsymbol{W}^{KR}\right)^\top}{\sqrt{d_k}}$$
$$i = 1, 2, \cdots, |\mathcal{V}|$$

where $\boldsymbol{W}^{QS}, \boldsymbol{W}^{KS}, \boldsymbol{W}^{VS}, \boldsymbol{W}^{KR}, \boldsymbol{W}^{VR}$ are the weight matrices in the structure-aware self-attention. This layer integrates the contextual semantic representation of entities and relations based on the graph structure, thereby injecting the structural information into the vanilla Transformer layer. Finally, we use a residual layer to fuse semantic and structural representations of entities, and obtain the hidden states for the following computation:

$$\tilde{\boldsymbol{h}}_i^l = \begin{cases} \boldsymbol{h}_i^l + \tilde{\boldsymbol{z}}_j^l, & i \in \mathcal{P}(e_j) \\ \boldsymbol{h}_i^l, & otherwise. \end{cases} \qquad (4)$$
$$i = 1, \cdots, m+n; \quad j = 1, \cdots, |\mathcal{V}|$$

Compared with existing structure-aware Transformer encoders (Zhu et al., 2019; Song et al., 2020) that either use the entity and relation embeddings from an external knowledge embedding model or directly learn them as model parameters, our encoder obtains the entity and relation embeddings via contextual semantic representations. This design fully employs the effective contextual representations from the existing pre-trained models while preserving the structural information, and enables our model to generalize to new entities and relations better when fine-tuned to the datasets with a different knowledge graph.

3.3 Pre-Training Task

Given the input graph $\mathcal{G}$ and its corresponding text sequence X, the goal of our pre-training task is to jointly learn the graph encoder and sequence decoder to enhance graph-text alignments, which can benefit the downstream tasks of KG-to-text generation. We devise three pre-training tasks to explicitly learn graph-text alignments in both discrete and continuous spaces.

3.3.1 Graph Enhanced Text Reconstruction

The purpose of graph enhanced text reconstruction is to recover the masked text sequence based on the complete knowledge graph, as shown in Figure 3. Assume that $\hat{X}$ denotes the masked text sequence, we can formulate the loss function of this pre-training task as follows:

$$\begin{aligned}\mathcal{L}_{text} &= -\log P(X|\mathcal{G}, \hat{X}) \\ &= -\sum_{i=1}^{n} \log P(x_i|\mathcal{G}, \hat{X}, x_{<i}) \qquad (5)\end{aligned}$$

To construct $\hat{X}$, we masked the entity words with a probability of 40% and other words with 20% since entity words are more important in the task of KG-to-text generation. We also follow the existing work (Lewis et al., 2020) to merge the consecutive mask tokens into one mask token to increase the difficulty of text reconstruction. This task enables our model to utilize the knowledge graph to reconstruct the corrupted text sequence, which explores the connection between them in the discrete vocabulary space.

3.3.2 Text Enhanced Graph Reconstruction

As shown in Figure 3, this pre-training task aims to recover the corrupted graph according to the information of the text sequence. Given the corrupted knowledge graph $\hat{\mathcal{G}}$ with masked entities and relations, and the complete text sequence X, the loss function is to recover the masked entities and relations in the linearized knowledge graph:

$$\begin{aligned}\mathcal{L}_{graph} &= -\log P(\mathcal{G}|\hat{\mathcal{G}}, X)\\ &= -\sum_{i=1}^{m} M_i \log P(w_i|\hat{\mathcal{G}}, X) \qquad (6)\end{aligned}$$

where M_i denotes an indicator function and equals 1 if and only if w_i is masked. We empirically set the masking probability of entities / relations as 40% / 20%. This task explicitly exerts the impact of the text on the graph reconstruction, thereby guiding the encoder to focus more on the entities and relations that may appear in the text.

3.3.3 Graph-Text Embedding Alignment

This pre-training task is devised to encourage the graph-text alignment in the embedding space. We use Optimal Transport (OT), which is commonly used in the cross-domain alignment (Chen et al., 2020a), to calculate the minimum cost of transporting the graph representation from the encoder to the text representation from the decoder (and vice versa). As shown in Figure 3, the input of the encoder is the linearized knowledge graph $\mathcal{G}_{linear}$ while the input of the decoder is the text sequence X. Assume that $\boldsymbol{H}^L = (\boldsymbol{h}_1^L, \boldsymbol{h}_2^L, \cdots, \boldsymbol{h}_m^L)$ indicates the final hidden states of the encoder, we can similarly acquire the entity and relation representations via mean pooling:

$$\begin{aligned}\boldsymbol{z}_i^L &= \text{pooling}(\{\boldsymbol{h}_p^L | p \in \mathcal{P}(e_i), 1 \leq p \leq m\})\\ \boldsymbol{q}_{ij}^L &= \text{pooling}(\{\boldsymbol{h}_p^L | p \in \mathcal{P}(r_{ij}), 1 \leq p \leq m\})\\ & i = 1, \cdots, |\mathcal{V}|; \quad j = 1, \cdots, |\mathcal{V}| \qquad (7)\end{aligned}$$

Let $\mathcal{G}_{seq} = \mathcal{V} \cup \mathcal{E} = (g_1, g_2, \cdots, g_{|\mathcal{V}|+|\mathcal{E}|})$ denotes the sequence of all the entities and relations in $\mathcal{G}$, we can directly obtain the contextual embedding vectors $\boldsymbol{H}^{\mathcal{G}} = (\boldsymbol{h}_1^{\mathcal{G}}, \cdots, \boldsymbol{h}_{|\mathcal{V}|+|\mathcal{E}|}^{\mathcal{G}})$ for each entity and relation from Equation 7. We can also acquire the embedding vectors of X from the decoder's final hidden states, which is denoted by $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_n)$.

To model the alignment between graphs and texts in the embedding space, we regard $\mathcal{G}_{seq}$ as a discrete distribution $\boldsymbol{\mu} = \sum_{i=1}^{|\mathcal{V}|+|\mathcal{E}|} \boldsymbol{a}_i \delta_{g_i}$ and X as $\boldsymbol{v} = \sum_{j=1}^{n} \boldsymbol{b}_j \delta_{x_j}$, where $\boldsymbol{a} = \{\boldsymbol{a}_i\}_{i=1}^{|\mathcal{V}|+|\mathcal{E}|}$ and $\boldsymbol{b} = \{\boldsymbol{b}_j\}_{j=1}^{n}$ satisfy $\sum_{i=1}^{|\mathcal{V}|+|\mathcal{E}|} \boldsymbol{a}_i = \sum_{j=1}^{n} \boldsymbol{b}_j = 1$, and δ_{g_i} / δ_{x_j} indicates the Dirac function centered on g_i / x_j. Then, we utilize the OT distance between $\boldsymbol{\mu}$ and $\boldsymbol{v}$ as the loss function, which is defined as the solution of the following problem:

$$\mathcal{L}_{OT} = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \sum_{i=1}^{|\mathcal{V}|+|\mathcal{E}|} \sum_{j=1}^{n} \boldsymbol{T}_{ij} \cdot d(g_i, x_j) \qquad (8)$$

$$\begin{aligned}\Pi(\boldsymbol{a}, \boldsymbol{b}) = \{\boldsymbol{T} \in \mathbb{R}_+^{(|\mathcal{V}|+|\mathcal{E}|) \times n} | \boldsymbol{T} \cdot \mathbf{1}_n = \boldsymbol{a},\\ \boldsymbol{T}^\top \cdot \mathbf{1}_{|\mathcal{V}|+|\mathcal{E}|} = \boldsymbol{b}\}\end{aligned}$$

where $\boldsymbol{T}$ denotes a transport plan, $\mathbf{1}_{|\mathcal{V}|+|\mathcal{E}|}$ / $\mathbf{1}_n$ indicates the $(|\mathcal{V}| + |\mathcal{E}|)$ / n -dimensional all-one vector respectively, and $d(g_i, x_j)$ is the cost function of transporting g_i to x_j. We follow the existing work (Chen et al., 2020c) to adopt the cosine distance between the contextual embedding vectors of g_i and x_j as the cost function, which is defined as $d(g_i, x_j) = 1 - \frac{\boldsymbol{h}_i^{\mathcal{G}} \boldsymbol{s}_j}{\|\boldsymbol{h}_i^{\mathcal{G}}\|_2 \|\boldsymbol{s}_j\|_2}$. Since the exact minimization over $\boldsymbol{T}$ is computationally intractable, we utilize IPOT algorithm (Xie et al., 2019) to approximate the OT distance and iteratively obtain the solution of $\boldsymbol{T}$ (more details are provided in the Appendix A). After solving $\boldsymbol{T}$, $\mathcal{L}_{OT}$ can serve as an alignment loss to optimize the model parameters. This task builds the connection between the contextual embedding vectors of knowledge graphs and texts, and explicitly promotes the graph-text alignment in the continuous space.

4 Experiment

4.1 Pre-training Dataset and Implementation

We used KGTEXT (Chen et al., 2020b) as our pre-training dataset. This dataset contains 7M graph-text data pairs, where texts are crawled from English Wikidump[4] and the corresponding knowledge graphs are acquired by querying WikiData with the

---

[4] `https://dumps.wikimedia.org`

Dataset	#Param	WebNLG(U)			WebNLG(C)			WebQuestions			PathQuestions		
Model		BLEU	METEOR	ROUGE	BLEU	METEOR	ROUGE	BLEU	METEOR	ROUGE	BLEU	METEOR	ROUGE
SOTA-NPT	-	61.00†	42.00†	71.00†	48.00†	36.00†	65.00†	29.45‡	30.96‡	55.45‡	61.48‡	44.57‡	77.72‡
KGPT	177M	64.11$^\sharp$	46.30$^\sharp$	74.57$^\sharp$	-	-	-	-	-	-	-	-	-
BART	140M	64.55	46.51	75.13	56.65	44.51	70.94	29.61	31.48	55.42	63.74	47.23	77.76
T5	220M	64.42	46.58	74.77	58.66	46.04	73.06	28.78	30.55	55.12	58.95	44.72	76.58
JointGT (BART)	160M	65.92	47.15	**76.10****	58.55	45.01	72.31	**30.02***	**32.05****	**55.60**	**65.89****	**48.25****	**78.87****
JointGT (T5)	265M	**66.14****	**47.25****	75.91	**61.01****	**46.32****	**73.57****	28.95	31.29	54.47	60.45	45.38	77.59

Table 1: Results on WebNLG, WebQuestions and PathQuestions. SOTA-NPT indicates the state-of-the-art performance from the baselines without pre-training. #Param means the number of model parameters. The results marked with †, ‡ and ♯ are re-printed from Shimorina and Gardent (2018), Chen et al. (2020d) and Chen et al. (2020b), respectively. - means that the results are not reported in the corresponding references. * indicates that our model significantly outperforms BART and T5 on the corresponding datasets (t-test, $p < 0.05$), while ** means $p < 0.01$.

Wikipedia hyperlinks of entities in the sentences. The detailed statistics of KGTEXT are shown in Table 2.

Dataset	#Ent	#Rel	#Instances (Train / Valid / Test)	#Triples	Length
KGTEXT	1.8M	1,210	6.98M / 10K / 10K	27.2	20.2
WebNLG(U)	3,114	373	34,352 / 4,316 / 4,224	2.9	22.7
WebNLG(C)	3,129	373	34,536 / 4,217 / 4,148	2.9	19.8
WebQuestions	25,703	672	18,989 / 2,000 / 2,000	5.8	15.0
PathQuestions	7,250	378	9,793 / 1,000 / 1,000	2.7	14.0

Table 2: Statistics of pre-training and fine-tuning datasets, including the total number of entities and relations, the data split, the average number of triples, and the average length of texts.

Since our model can adapt to Transformer-based pre-trained models with the encoder-decoder framework, we chose BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) as the base model in this paper, which are denoted by JointGT (BART) and JointGT (T5), respectively. The hyper-parameters of the Transformer blocks were the same as BART-base and T5-base because of the limited computational resources. We initialized our model parameters with the pre-trained checkpoint of BART-base / T5-base except for the structure-aware semantic aggregation module, which was randomly initialized. We followed BART / T5 to use Byte-Pair Encoding (BPE) vocabulary (Radford et al., 2019) with the size of 50,265 / WordPiece vocabulary (Kudo and Richardson, 2018) with the size of 32,000. The batch size was 42 / 32 for JointGT (BART) / JointGT (T5). The maximum length of linearized input graphs was 600, while the maximum length of text sequences was 64. We adopted Adam (Kingma and Ba, 2015) as the optimizer and set the learning rate to be 3e-5. The warmup ratio was 0.1. JointGT was pre-trained on KGTEXT for 1 epoch with the proposed pre-training tasks. It took 44 / 69 hours for JointGT (BART) / JointGT (T5) on 3 NVIDIA Quadro RTX 6000 GPUs.

4.2 Fine-Tuning Settings

We adopted WebNLG, WebQuestions and Path Questions as the benchmark datasets during fine-tuning, and provided the statistics in Table 2.

WebNLG: This dataset aims to convert RDF triples into a textual description. We followed the existing work (Chen et al., 2020b) to use the version of 2.0 (Shimorina and Gardent, 2018). This dataset contains two official data splits: the traditional split (Unconstrained) which guarantees that there is no overlap of input graphs among train / validation / test sets, and a more challenging split (Constrained) where the non-overlap constraint is applied to the triples of input graphs. We denoted these two data splits as *WebNLG(U)* and *WebNLG(C)* in our paper. We followed the preprocessing steps of the existing work (Chen et al., 2020b) to replace the underlines in the entities and relations with spaces, and split the entities and relations in a camel case into multiple words.

WebQuestions: This dataset (Yih et al., 2016; Talmor and Berant, 2018) is the benchmark for question generation over knowledge bases (KBQG), whose purpose is to generate natural language questions about the corresponding knowledge graphs (Serban et al., 2016). It is constructed from two question answering datasets, i.e., WebQuestionsSP (Yih et al., 2016) and ComplexWebQuestions (Talmor and Berant, 2018). These two datasets contain natural language questions, SPARQL queries and answer entities. We converted the SPARQL query to return a subgraph, and used the same preprocessing steps and data splits as the existing work (Kumar et al., 2019; Chen et al., 2020d).

PathQuestions: Similar to WebQuestions, the PathQuestions dataset is also the benchmark for KBQG, which is constructed from a question answering dataset (Zhou et al., 2018b). The main

Model	Fluency			κ	Adequacy			κ
	Win (%)	Lose (%)	Tie (%)		Win (%)	Lose (%)	Tie (%)	
JointGT (BART) vs. BART	29.0*	19.7	51.3	0.413	26.3**	16.0	57.7	0.517
JointGT (T5) vs. T5	23.7	18.7	57.6	0.405	22.7*	16.3	61.0	0.424

Table 3: Human evaluation on WebNLG(U). The scores indicate the percentages of win, lose and tie when JointGT is compared with other baselines. κ is Fleiss' Kappa (all indicate moderate agreement). The scores marked with * mean $p < 0.05$ while ** means $p < 0.01$ in sign test.

difference is that the knowledge graph in PathQuestions is a 2-hop / 3-hop path between two entities. We used the same preprocessing steps and data splits as the existing work (Kumar et al., 2019; Chen et al., 2020d).

More detailed fine-tuning settings including the search space and the best assignment of hyperparameters on the downstream datasets are reported in the Appendix B.

4.3 Baselines

We chose the following two categories of models as our baselines:

Pre-Trained Models: We adopted KGPT (Chen et al., 2020b), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) as the pre-trained baselines. KGPT is a pre-trained model for KG-to-text generation, which utilizes the same pre-training dataset as our model and directly uses KG-to-text generation as the pre-training task. BART and T5, as the state-of-the-art pre-trained models for text generation, can be applied to KG-to-text generation with the input of linearized knowledge graphs and the output of text sequences (Ribeiro et al., 2020a).

Task-Specific Models without Pre-Training: We also chose the state-of-the-art task-specific models without pre-training for each dataset as our baselines, including Seq2Seq with copying or delexicalisation (Shimorina and Gardent, 2018) for WebNLG v2.0, and G2S (Chen et al., 2020d) for WebQuestions and PathQuestions.

We directly re-printed the results of baselines if they use the same datasets as ours. Otherwise, we implemented the baselines based on the codes and model parameters released by the original papers. We reported all the results of our implemented models with the mean values over 5 runs.

4.4 Automatic Evaluation

We followed the existing work (Shimorina and Gardent, 2018; Chen et al., 2020d) to use BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE-L (Lin, 2004) as our automatic metrics. The main results on WebNLG, WebQuestions and PathQuestions are shown in Table 1. We can observe that JointGT based on BART / T5 can outperform vanilla BART / T5 on most of the metrics, respectively, and obtain the state-of-the-art performance on all the datasets. This indicates that our method can promote graph-text alignments and further enhance the performance of the state-of-the-art pre-trained models on KG-to-text datasets.

4.5 Human Evaluation

To further evaluate the quality of generated results, we conducted human evaluation on the WebNLG(U) dataset. We followed the existing work (Ferreira et al., 2019; Ribeiro et al., 2020b) to select two criteria: *fluency* (whether a sentence is grammatically fluent) and *adequacy* (whether a sentence clearly describes the knowledge graph). We randomly sampled 100 knowledge graphs from the test set, and collected the generated results from our models and the most competitive baseline models (i.e., BART and T5). We used the pairwise comparison between BART / T5 and JointGT (BART) / JointGT (T5). Specifically, for each pair of generated texts (one from JointGT and the other from the corresponding baseline, given the same input knowledge graph), three annotators were hired to label which text is better (i.e., win, lose or tie) in terms of the metrics mentioned above. Note that the two metrics were evaluated independently.

Results in Table 3 show that JointGT can beat the corresponding baselines in both fluency and adequacy. Especially for adequacy, our model can significantly outperform BART / T5, which indicates that our model equipped with the structure-aware encoder and well-designed pre-training tasks can generate high-quality texts to describe knowledge graphs more clearly. To evaluate the agreement among different annotators, we calculated Fleiss' Kappa (Fleiss, 1971) for each pairwise comparison, where the results in Table 3 show moderate agreement ($0.4 \leq \kappa \leq 0.6$).

4.6 Ablation Study

4.6.1 Encoder Structure

To investigate the effect of our proposed structure-aware semantic aggregation module, we fixed the pre-training tasks and compared our encoder with two Transformer-based encoders commonly used in the existing work:

SeqEnc: This sequence encoder takes linearized graphs as input and ignores structural information (Ribeiro et al., 2020a; Kale and Rastogi, 2020).

RelEnc: This relation-aware encoder regards the entity sequence as input and leverages the relation embedding into the self-attention layer. Both the entity and relation embedding vectors are directly learned as model parameters (Shaw et al., 2018; Zhu et al., 2019; Song et al., 2020).

Model	#Param	BLEU	METEOR	ROUGE
JointGT (BART)	160M	**65.92**	**47.15**	**76.10**
w/ SeqEnc	140M	64.82	46.87	75.37
w/ RelEnc	160M	65.17	47.07	75.69

Table 4: Ablation test of different encoder structures on WebNLG(U), including our encoder, sequence encoder (SeqEnc) and relation-aware encoder (RelEnc).

Note that we only chose the encoder structures that can directly adapt to BART / T5 for fair comparison[5]. Results in Table 4 show that our encoder structure can perform better than the other baselines. Compared with the relation-aware encoder which can also capture the structural information of knowledge graphs, our model fully utilizes the effective contextual semantic representation to initialize the entity / relation representation at each Transformer layer instead of directly using the learnable entity / relation embedding vectors. This design equips JointGT with better generalization ability during fine-tuning, thereby enhancing our performance on downstream datasets.

Model	#Triples	
	1-3	4-7
JointGT (BART)	**71.24**	**61.36**
w/ SeqEnc	70.83 (-0.41)	60.11 (-1.25)
w/ RelEnc	70.98 (-0.26)	60.58 (-0.78)

Table 5: BLEU scores of three encoders on the test set of WebNLG(U) with different numbers of input triples.

To further demonstrate the effectiveness of our encoder, we divided the test set of WebNLG(U) into two subsets according to the number of triples in knowledge graphs, and compared the performance of three encoders. Results in Table 5 show that the improvement margin between our encoder and other encoders is more evident when the number of input triples is large, which indicates that our model can facilitate the encoding of knowledge graphs with more complex structures.

4.6.2 Pre-Training Task

Model	BLEU	METEOR	ROUGE
JointGT (BART)	**65.92**	**47.15**	**76.10**
w/o TextRecon	64.22	46.56	74.96
w/o GraphRecon	65.37	47.09	75.97
w/o OT	65.03	47.09	75.83
w/ BARTPretrain	64.60	46.78	75.74
w/ KGPTPretrain	65.14	46.94	75.72

Table 6: Ablation test of three pre-training tasks on WebNLG(U), including text / graph reconstruction and graph-text alignments via OT. BARTPretrain / KGPT-Pretrain means using the pre-training tasks of BART / KGPT instead of our tasks on KGTEXT.

To study the effect of three pre-training tasks, we maintained the encoder structure and removed each task respectively to test the performance. We also replaced all our pre-training tasks with the tasks of the existing work for comparison:

BARTPretrain: The pre-training tasks of BART including text infilling and sentence permutation (Lewis et al., 2020). Since these tasks cannot be applied to graph data, we only used these tasks on the text data of the pre-training dataset.

KGPTPretrain: The pre-training task of KGPT, i.e., KG-to-text generation on the pre-training dataset (Chen et al., 2020b).

Results in Table 6 show that each of our pre-training tasks contributes to the model performance. Compared with the other two tasks, graph enhanced text reconstruction plays a more important role in the task of KG-to-text generation, which directly supervises the decoder with the conditional generation loss. We also observe an apparent performance drop if we replace our pre-training tasks with those proposed by the existing work, thereby indicating the effectiveness of our pre-training tasks to promote KG-to-text generation.

4.7 Few-Shot Learning

To further analyze whether our pre-training tasks can learn a good graph-text joint representation that benefits the downstream KG-to-text generation tasks, we considered the few-shot setting where

---

[5] We observed a significant performance drop if we used the encoders which are incompatible with BART / T5 (such as graph neural networks) because we had to randomly initialize the parameters of them during pre-training.

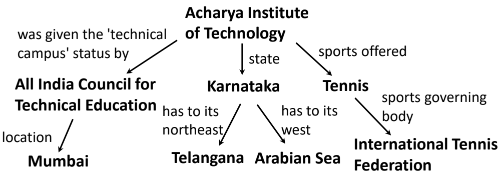

BART: The Acharya Institute of Technology is located in the state of Karnataka which has Telangana to its northeast and the Arabian Sea to its west . The Institute was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . The International Tennis Federation governs tennis . **[sports offered]**

T5: The Acharya Institute of Technology is located in Karnataka , India . It was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . The Institute is affiliated with the International Tennis Federation . The Arabian Sea is to the west of Karnataka and Telangana is northeast of the state . Tennis is one of the sports offered by the Institute . **[sports governing body]**

JointGT (BART): The Acharya Institute of Technology is located in the state of Karnataka which has Telangana to its northeast and the Arabian Sea to its west . The Institute was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . One of the sports offered at the Institute is tennis which is governed by the International Tennis Federation .

JointGT (T5): The Acharya Institute of Technology is located in the state of Karnataka . Karnataka has Telangana to its northeast and the Arabian Sea to its west . The Institute was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . The Institute offers tennis which is governed by the International Tennis Federation .

Figure 4: Generated results on WebNLG(U). We highlight the missing and unfaithful parts of each text in red and blue, respectively.

Model	Data Proportion			
	0.5%	1%	5%	10%
BART	33.92	39.08	52.24	56.58
JointGT (BART)	**37.18**	**42.26**	**54.41**	**57.73**
w/ BARTPretrain	32.63	37.11	52.91	56.81
w/ KGPTPretrain	35.33	40.72	53.08	57.18

Table 7: BLEU scores of the models with correponding pre-training tasks trained on different proportions of WebNLG(U).

only a few training instances were used during fine-tuning. We still fixed our model structure and compared our pre-training tasks with the tasks of BART and KGPT mentioned in §4.6.2.

Results in Table 7 show that our pre-training tasks can perform better than other tasks, especially when the amount of training data is small. This indicates that our proposed tasks can capture the graph-text alignments during pre-training, thereby making our model generalizable to the downstream KG-to-text datasets better with only a few training samples.

4.8 Case Study

To intuitively show the generation quality of our model, we provided some generated cases in Figure 4. We observe that JointGT can generate high-quality texts that describe the knowledge graph more completely and faithfully. For example, in the generated case on WebNLG(U), both BART and T5 fail to cover all the input triples, where BART misses the triple *(Acharya Institute of Technology, sports offer, Tennis)* and T5 misses *(Tennis, sports governing body, International Tennis Federation)*. Also, T5 generates non-existing facts that are unfaithful to the knowledge graph. Equipped with the structure-aware Transformer encoder and the well-designed pre-training tasks to learn graph-text alignments, JointGT (BART) and JointGT (T5) can generate descriptions which include all the input triples and express the relation between each pair of entities more faithfully.

5 Conclusion

We propose a novel graph-text joint representation learning model called JointGT for KG-to-text generation. This model plugs a simple structure-aware semantic aggregation module into the vanilla Transformer layer to preserve the structure of input graphs, and utilizes three pre-training tasks to learn graph-text alignments in the discrete vocabulary space and continuous embedding space. Experiments show that JointGT can outperform state-of-the-art pre-trained NLG models on various datasets of KG-to-text generation.

Acknowledgments

This work was partly supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005*, pages 65–72.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko.

2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Liqun Chen, Zhe Gan, Yu Cheng, Linjie Li, Lawrence Carin, and Jingjing Liu. 2020a. Graph optimal transport for cross-domain alignment. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1542–1553.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020b. KGPT: knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8635–8648.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020c. UNITER: universal image-text representation learning. In *ECCV*, volume 12375, pages 104–120.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020d. Toward subgraph guided knowledge graph question generation with graph neural networks. *arXiv preprint arXiv: 2004.06015*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 552–562.

J. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382.

Zihao Fu, Bei Shi, Wai Lam, Lidong Bing, and Zhiyuan Liu. 2020. Partially-aligned data-to-text generation with distant supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 9183–9193.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6473–6480.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv: 2006.04702*.

Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. 2020. Language generation with multi-hop reasoning on commonsense knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 725–736.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2284–2293.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations*, pages 66–71.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *18th International Semantic Web Conference*, volume 11778, pages 382–398.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md. Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2267–2277.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 43–54.

Gabriel Peyré and Marco Cuturi. 2019. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5-6):355–607.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *OpenAI Technical Report*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *OpenAI Technical Report*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020a. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv: 2007.08426*.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020b. Modeling global and local node contexts for text generation from knowledge graphs. *Trans. Assoc. Comput. Linguistics*, 8:589–604.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2020a. Modeling graph structure via relative position for better text generation from knowledge graphs. *arXiv preprint arXiv: 2006.09242*.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020b. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7117–7130.

Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 464–468.

Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 641–651.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1627–1637.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. 2019. A fast proximal point method for computing exact wasserstein distance. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence*, volume 115, pages 433–453.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2020. JAKET: joint pre-training of knowledge graph and language understanding. *arXiv preprint arXiv: 2010.00796*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1441–1451.

Chao Zhao, Marilyn A. Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018a. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4623–4629.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018b. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2010–2022.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5458–5467.

A IPOT Algorithm

Inexact Proximal point method for Optimal Transport (IPOT) is an effective iterative method to approximate OT distance and compute the transport plan $\boldsymbol{T}$ (Xie et al., 2019). Given the sequence of entities and relations in the knowledge graph $\mathcal{G}_{seq} = (g_1, \cdots, g_{|\mathcal{V}|+|\mathcal{E}|})$ with its corresponding embedding vectors $\boldsymbol{H}^{\mathcal{G}} = (\boldsymbol{h}_1^{\mathcal{G}}, \cdots, \boldsymbol{h}_{|\mathcal{V}|+|\mathcal{E}|}^{\mathcal{G}})$, and the text sequence $X = (x_1, \cdots, x_n)$ with its embedding vectors $\boldsymbol{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_n)$, the implementation of IPOT algorithm to calculate $\boldsymbol{T}$ is shown in Algorithm 1.

Algorithm 1 IPOT Algorithm

```
Require:
    $\mathcal{G}_{seq} = \{g_i\}_{i=1}^{|\mathcal{V}|+|\mathcal{E}|}$, $X = \{x_j\}_{j=1}^{n}$, and their embedding vectors $\boldsymbol{H}^{\mathcal{G}} = \{\boldsymbol{h}_i^{\mathcal{G}}\}_{i=1}^{|\mathcal{V}|+|\mathcal{E}|}$, $\boldsymbol{S} = \{\boldsymbol{s}_j\}_{j=1}^{n}$
    Generalized stepsize: $1/\beta$
$\boldsymbol{\sigma} = \frac{1}{n}\mathbf{1}_n, \boldsymbol{T}^{(1)} = \mathbf{1}_{|\mathcal{V}|+|\mathcal{E}|}\mathbf{1}_n^{\top}$
$\boldsymbol{C}_{ij} = d(g_i, x_j) = 1 - \frac{\boldsymbol{h}_i^{\mathcal{G}}\boldsymbol{s_j}}{\|\boldsymbol{h}_i^{\mathcal{G}}\|_2\|\boldsymbol{s}_j\|_2}$
$\boldsymbol{A}_{ij} = e^{-\frac{\boldsymbol{C}_{ij}}{\beta}}$
for $t = 1$ to $N$ do
    $\boldsymbol{Q} = \boldsymbol{A} \odot \boldsymbol{T}^{(t)}$
    for $k = 1$ to $K$ do
        $\boldsymbol{\delta} = \frac{1}{(|\mathcal{V}|+|\mathcal{E}|)\boldsymbol{Q}\boldsymbol{\sigma}}, \boldsymbol{\sigma} = \frac{1}{n\boldsymbol{Q}^{\top}\boldsymbol{\delta}}$
    end for
    $\boldsymbol{T}^{(t+1)} = \mathrm{diag}(\boldsymbol{\delta})\boldsymbol{Q}\mathrm{diag}(\boldsymbol{\sigma})$
end for
return $\boldsymbol{T}$
```

In the algorithm of IPOT, $\odot$ denotes Hadamard product. β, K and N are all hyper-parameters. We followed the existing work (Chen et al., 2020a) to set $\beta = 1.0$, $K = 1$ and $N = 10$.

B Hyper-Parameter Setting

Hyper-parameter	Search Space
Masking Probability (entity / relation / word)	*choice*[20%,30%,40%]
Learning Rate	*choice*[2e-5,3e-5,5e-5]
Training Epoch	*choice*[1,2]
Warmup Ratio	*choice*[0,0.1]
Batch Size	*choice*[32,36,42]
Input Length	600
Output Length	64
Maximum Gradient Norm	1.0
Optimizer	Adam
Epsilon (for Adam)	1e-8

Table 8: Hyper-parameter search space of JointGT during pre-training. *choice* indicates that the listed numbers will be chosen with the same probability.

We provided the detailed settings of hyper-parameters during pre-training and fine-tuning. The settings include hyper-parameter search space and best assignments. Note that we used Huggingface's Transformers[6] to implement our models.

---

[6] `https://github.com/huggingface/transformers`

Hyper-parameter	Search Space
Learning Rate	*choice*[2e-5,3e-5,5e-5,1e-4]
Training Epoch	*choice*[20,30,40]
Warmup Step	*uniform-integer*[0,total_step*0.2]
Batch Size	*choice*[24,32]
Input Length	*choice*[128,256]
Output Length	*choice*[64,128]
Beam Size	*choice*[2,3,5]
Length Penalty	*choice*[1.0,3.0,5.0]
Maximum Gradient Norm	1.0
Optimizer	Adam
Epsilon (for Adam)	1e-8

Table 9: Hyper-parameter search space of JointGT during fine-tuning. *uniform-integer* means the integers in the interval can be selected uniformly. In the search space of warmup step, total_step denotes the total training steps on the corresponding datasets.

Thus all the hyper-parameters reported in our paper were consistent with the codes of Huggingface's Transformers.

Model	JointGT (BART)			
Dataset	WebNLG(U)	WebNLG(C)	WebQuestions	PathQuestions
Learning Rate	2e-5	2e-5	2e-5	5e-5
Training Epoch	40	20	30	40
Warmup Step	1,600	0	3,400	1,100
Batch Size	32	32	32	32
Input Length	256	256	256	128
Output Length	128	128	128	64
Beam Size	5	5	5	5
Length Penalty	1.0	1.0	5.0	1.0
Model	JointGT (T5)			
Dataset	WebNLG(U)	WebNLG(C)	WebQuestions	PathQuestions
Learning Rate	5e-5	3e-5	1e-4	2e-5
Training Epoch	30	30	40	30
Warmup Step	1,600	1,200	2,300	900
Batch Size	24	32	32	32
Input Length	256	256	256	128
Output Length	128	128	64	64
Beam Size	5	5	5	2
Length Penalty	1.0	1.0	5.0	1.0

Table 10: Best assignments of hyper-parameters on the downstream datasets.

We presented the hyper-parameter search space during pre-training in Table 8. The number of hyper-parameter search trials was 10. Manual search was adopted to select hyper-parameters, and the selection criterion was BLEU on the validation set when we fine-tuned the pre-trained model on WebNLG(U). The best assignment of pre-training was described in our main content.

We also provided the detailed settings of hyper-parameters during fine-tuning on the downstream datasets, including the hyper-parameter search space in Table 9 and the best assignments in Table 10. The number of hyper-parameter search trials was 20. BLEU was adopted as our criterion in the manual search on all the downstream tasks.

AdaST: Dynamically Adapting Encoder States in the Decoder for End-to-End Speech-to-Text Translation

Wuwei Huang, Dexin Wang, Deyi Xiong *
College of Intelligence and Computing, Tianjin University, Tianjin, China
{wwhuang, 2019218007, dyxiong}@tju.edu.cn

Abstract

In end-to-end speech translation, acoustic representations learned by the encoder are usually fixed and static, from the perspective of the decoder, which is not desirable for dealing with the cross-modal and cross-lingual challenge in speech translation. In this paper, we show the benefits of varying acoustic states according to decoder hidden states and propose an adaptive speech-to-text translation model that is able to dynamically adapt acoustic states in the decoder. We concatenate the acoustic state and target word embedding sequence and feed the concatenated sequence into subsequent blocks in the decoder. In order to model the deep interaction between acoustic states and target hidden states, a speech-text mixed attention sublayer is introduced to replace the conventional cross-attention network. Experiment results on two widely-used datasets show that the proposed method significantly outperforms state-of-the-art neural speech translation models.

1 Introduction

Speech-to-text translation (ST) aims at translating the source language speech into the text of the target language. Approaches to ST can be roughly divided into two categories: end-to-end ST and cascaded ST. Early research on ST is primarily using a cascaded model that combines a speech recognition (ASR) module with a machine translation component, both usually trained independently on speech and parallel corpora (Ney, 1999; Matusov et al., 2005). In contrast, end-to-end ST, which directly translates the speech of the source language into text of the target language (Berard et al., 2016), not only avoids error propagation in the ASR-MT pipeline, but also greatly reduces inference latency.

However, despite these advantages, end-to-end ST is confronted with its own challenging problems: performing cross-modal translation and cross-lingual conversion in one shot. On the one hand, compared with text-to-text translation, end-to-end ST has to deal with acoustic inputs which are typically longer than their corresponding text inputs. This makes the cross-modal source-target dependencies more complicated. On the other hand, compared with monotonic ASR, end-to-end speech translation usually handles non-monotonic cross-lingual conversion.

Generally, end-to-end ST uses the seq2seq encoder-decoder framework (Sutskever et al., 2014) as the backbone for training and inference, where the encoder computes hidden states layer by layer according to speech inputs. The decoder yields target translations word by word by attending to the fixed-after-computing hidden states of the encoder. Since the hidden states are static in the encoder, information only flows one direction: from the encoder to the decoder. Given the cross-modal and cross-lingual challenge in end-to-end ST, we argue that more sophisticated interaction between the encoder and decoder would be desirable.

In this paper, we propose an adaptive ST (AdaST) model that incorporates acoustic states into the decoder for modeling the deep interaction between the encoder and decoder for end-to-end ST. We enable AdaST to dynamically adapt encoder states in the decoder when target hidden states are updated layer by layer. It also learns to represent speech and text in one shared space in the decoder for mitigating the cross-modal issue.

Our contributions can be summarized as follows:

- We present AdaST, a new architecture for end-to-end ST, which learns representations of two modalities (textual and audio) in one shared space in the decoder.
- We conduct experiments to validate the effectiveness of AdaST. Our experiments and

---

* Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2539–2545
August 1–6, 2021.

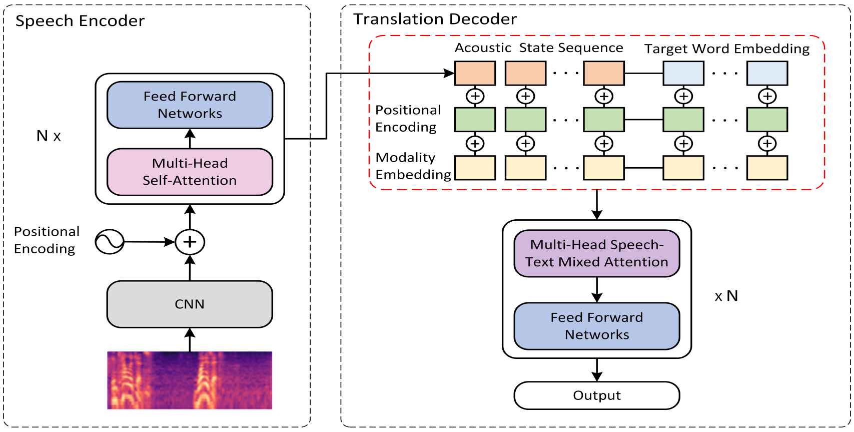

Figure 1: Diagram of the proposed AdaST that dynamically adapts acoustic encoder states in the decoder for end-to-end ST.

analyses disclose that dynamically adaptive acoustic representations are more desirable than static acoustic states for end-to-end ST.

2 Related Work

Berard et al. (2016) demonstrate the potential of end-to-end neural ST and Weiss et al. (2017) achieve good performance by using an end-to-end neural architecture, which trigger more research interests in end-to-end ST. Both Bansal et al. (2019) and Stoian et al. (2020) train a speech recognition model first and then use the encoder of ASR to initialize the encoder of speech translation. Jia et al. (2019) synthesize training data for end-to-end speech translation from MT and ASR dataset. Gangi et al. (2019); Inaguma et al. (2019) adapt the idea of multilingual machine translation to speech translation. In addition to these methods, Bahar et al. (2019) use phoneme-level representations instead of speech frame-level representations as input, greatly reducing the length of acoustic sequences. Knowledge distillation (Liu et al., 2019), meta-learning (Indurthi et al., 2019), curriculum learning (Wang et al., 2020b), and two-pass decoding (Sung et al., 2019), have also been studied in end-to-end speech translation.

To solve the cross-modal and cross-lingual challenges of end-to-end speech translation, Wang et al. (2020a) and Dong et al. (2020) propose to use submodules to separately analyze cross-modal and cross-lingual problems in end-to-end ST. Each module introduced solves one problem. Unfortunately, they introduce a large number of extra parameters and rely on a large amount of external data to pre-train each submodule. In contrast, we do not introduce any additional submodules and therefore we do not need external data for pretraining.

3 The AdaST Model

In this section, we first introduce the widely-used CNN + Transformer structure as the strong baseline for end-to-end ST. After that we elaborate the proposed AdaST model.

3.1 Baseline ST Model

The CNN + Transformer end-to-end ST model consists of a speech encoder and a translation decoder. The basic building unit of Transformer (Vaswani et al., 2017) is the self-attention mechanism, which can be formulated as follows:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The speech encoder is composed of N_c CNN layers for encoding acoustic signals and Ne Transformer encoder layers stacked over CNN layers. The translation decoder consists of N_d Transformer decoder layers.

The CNN module subsamples acoustic features to fit them into the subsequent Transformer encoder layers. The Transformer encoder layers then learn encoder states from the output of the CNN module, which are fixed during decoding. That is to say, the Transformer decoder layers attend to static Transformer encoder hidden states for yielding target words.

3.2 AdaST

As shown in Figure 1, our proposed AdaST uses the same speech encoder as the baseline ST model. The significant difference lies in the decoder. In order to make acoustic states dynamically adaptive to decoder states in each layer, we concatenate the hidden acoustic state sequence generated from the last layer of the speech encoder with the target word embedding sequence and feed the concatenated sequence into the subsequent decoder blocks. The concatenated input sequence is combined with positional encoding, similar to the vanilla Transformer decoder. In addition to positional encoding, we also adapts modality embeddings, which are defined in a embedding matrix with size of $2 \times c$ (c is the dimension of attention) adding to the input sequence to distinguish the target textual tokens from the source acoustic features. Modality embeddings has also been used in other cross-modal tasks, e.g., Vilbert for vision-text multimodal pretraining. Our experiments show that using modality embeddings in our model can slightly improve translation quality.

In the decoder, each block consists of a multi-head speech-text mixed attention sublayer and a feedforward sublayer. The multi-head speech-text mixed attention (STMA) is calculated as follows:

$$\text{STMA}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}} + Mask)V \tag{2}$$

$$Q = \text{Concat}(src, tgt)W^Q \tag{3}$$

$$K = \text{Concat}(src, tgt)W^K \tag{4}$$

$$V = \text{Concat}(src, tgt)W^V \tag{5}$$

where src and tgt represent the sequence of acoustic hidden states and target word embeddings respectively, and $Mask$ is a predefined matrix which serves as indicators controlling which positions of the acoustic and target sequence are visible to attention heads, similar to the look ahead mask matrix used in Transformer to prevent the decoder from attending future tokens.

In each decoder layer of the proposed AdaST, we divide the $Mask$ matrix into four parts:

$$Mask = \begin{bmatrix} M_{SS} & M_{ST} \\ M_{TS} & M_{TT} \end{bmatrix}$$

M_{SS} represents the self-attention mask matrix of the acoustic state, which is the same as used in the encoder. M_{ST} is the mask matrix for the attention from acoustic states to target hidden states. During parallel training, as source acoustic states are not visible to target hidden states, we set all values of M_{ST} to minus infinity to forbid such attention. M_{TS} denotes the mask matrix for attention from target hidden states to acoustic states. Values in M_{TS} are the same as the mask matrix used for the cross-attention in Baseline ST. M_{TT} is the mask matrix for self-attention over target hidden states, which is the same as the mask matrix used for self-attention on the Baseline ST decoder.

The proposed AdaST benefits from the following features. First, the acoustic states and decoder hidden states are unified into a shared semantic space. Second, the acoustic states at each decoder layer change accordingly after the calculations at the current layer are performed. Third, instead of calculating softmax for self-attention and then calculating softmax for cross-attention in the baseline ST, the neural representations in the AdaST decoder are updated by calculating a single softmax over both acoustic states and hidden states for target words. With these changes, we hope to mitigate the cross-modal and cross-lingual challenges in end-to-end ST.

4 Experiments

We conducted experiments to examine the proposed AdaST model.

4.1 Datasets

We used two datasets that are widely adopted to evaluate end-to-end ST: IWSLT18 En-De and Augmented Librispeech En-Fr (Berard et al., 2018).

Augmented Librispeech English-French. The corpus provides triples for each instance: English speech signal, English transcription, French text translation from the aligned e-books. Following Wang et al. (2020b), we only used the 100 hours clean data for training, with 2 hours data as the development set and 4 hours as the test set, which corresponds to 47,271, 1071 and 2048 utterances respectively. To be consistent with their settings, we also doubled the training data by concatenating the aligned references with pseudo translations by the Google Translate.

IWSLT18 English-German. The IWSLT18 speech translation dataset is from TED Talks, which contains 271 hour speech with 171K corresponding English transcripts and German translations. As there is no validation set in this dataset, we randomly sampled 2000 samples from the training data as our validation set. Following Wang et al.

(2020b), we used tst2013 as the test set.

4.2 Settings

We built our model based on the Espnet toolkit (Inaguma et al., 2020). On the two datasets, we extracted 80-dimensional Fbank features from audio files, setting the step size as 10ms and the window size as 25ms. We deleted sentences with frame size larger than 3000 and sentences with poor alignments. Following Wang et al. (2020a), we adopted speed perturbation with factors 0.9 and 1.1. To further reduce overfitting, we used SpecAugment strategy (Bahar et al., 2019). In Librispeech, we used subword level decoding, which was performed via SentencePiece with a size of 1K tokens. In IWSLT18, we performed character level decoding. As the tst2013 of IWSLT2018 is not aligned, we employed Espnets default LIUM SpkDiarization tool to segment each audio sequence. We used RWTH toolkit (Bender et al., 2004) to calculate BLEU scores (Papineni et al., 2002).

A two-layer CNN was taken in the speech encoder. The step size was set to 2. The size of the convolution kernel was 2 × 2. The dimension of the attention was set to 256. We used 12-layer encoder. The number of decoder layers in both the baseline and AdaST was set to 10. We used the Adam optimizer (Kingma and Ba, 2015) and run our models on four P100 GPUs.

4.3 Main Results

In order to make each layer of the decoder to interact with acoustic states, our model requires additional computational overhead. However, the conventional source-to-target attention network in Transformer is subsumed in the decoder, which helps AdaST to use fewer parameters than Transformer, hence partially offsetting the additional cost. Overall, the number of parameters in AdaST is 0.65 million fewer than that of the standard CNN+Transformer structure. On the augmented dataset, AdaST increased the training time by 11.7% and the inference time by 15.7%. We compared our work against previous state-of-the-art models and the ASR pretraining + MT fine-tuning method. Table 1 shows the results on the two datasets. We observe that the proposed AdaST is able to achieve improvements of +0.83 BLEU and +1.18 BLEU over the best baseline results on En-Fr and En-De translation, respectively. This demonstrates that our proposed method benefits end-to-end ST at both the character and subword level. We have also carried out experiments to compare against a standard CNN+Transformer model with deeper encoder and decoder. Experiment results show that simply deepening either encoder or decoder of the standard structure is not helpful for speech-to-text translation.

	Method	BLEU
En-Fr	LSTM ST (Berard et al., 2018)	12.90
	Transformer+ASR pre-train (Inaguma et al., 2020)	15.53
	Transformer+ASR pre-train	16.27
	AdaST	**17.10**
En-De	Transformer+ASR pre-train (Inaguma et al., 2020)	13.12
	Transformer+ASR pre-train (Wang et al., 2020b)	15.35
	Transformer+ASR pre-train	15.21
	AdaST	**16.39**

Table 1: Results on the two datasets.

	Structure	Result
ST	Transformer+ASR pre-train	16.27
	AdaST	17.10
ASR	Transformer	7.5
	AdaST	8.3
MT	Transformer	18.10
	AdaST	18.16

Table 2: Results of using AdaST on different tasks, i.e., speech translation (ST), automatic speech recognition (ASR) and machine translation (MT). BLEU (↑) scores are reported on ST and MT while CER (↓) on ASR.

5 Analysis

We conducted further experiments and analyses to take a deep look into our proposed method.

5.1 Only Cross-modal or Cross-lingual Challenge

In order to investigate whether our proposed architecture is helpful for a task with only cross-modal or cross-lingual challenge, we also conducted experiments for automatic speech recognition (ASR) and machine translation (MT) tasks with the proposed method on the Agmented Librispeech dataset. Experimental results in Table 2 show that the performance of ASR task drops, while the performance of MT task is improved slightly. This suggests that the proposed architecture is more appropriate for dealing with cross-lingual and cross-modal challenges at the same time.

5.2 Adaptive vs. Static Acoustic States

We assume that dynamically adaptive representations of acoustic states in accord with hidden decoder states at each decoder layer will be of great

Structure	BLEU
Transformer+ASR pre-train	15.21
Transformer+Additional Self-Att	16.08
AdaST	**16.39**

Table 3: Results of dynamic vs. static acoustic states.

Method	ACC
Transformer+ASR pre-train	74.2
AdaST	96.7

Table 4: Classification accuracy (%) on speaker verification.

help to end-to-end ST. In order to examine this hypothesis, we add an additional self-attention at each encoder layer in the baseline ST, which forces acoustic states at the corresponding encoder layer to adapt to decoder hidden states. The results on the IWSLT18 dataset, as displayed in Table 3, validate this assumption. However, the added additional self-attention substantially increase the number of parameters at each layer. By contrast, our AdaST does not introduce additional parameters at each layer to learn adaptive acoustic states on the one hand and achieves better performance on the other.

5.3 Probing the Speech Encoder

We further compared the trained speech encoder of our AdaST against that of the baseline ST by evaluating speaker verification accuracy on the Fluent Speech Commands dataset (Lugosch et al., 2019) to investigate the change of the semantic information learned by the encoder. Generally, the more semantic information the encoder contains, the less audio information it learns and hence the lower classification accuracy it will obtain. We froze parameters of these two speech encoders, and added a linear classification layer on the top of the encoder . Only the added classification layer is trained on the dataset mentioned above. Table 4 shows the classification accuracy results, where the baseline encoder achieves 74.2% while our encoder 96.7%, substantially higher than the baseline encoder. This indicates that our encoder focuses on modeling the audio modality and passes the major task of modeling semantic information in speech inputs to the decoder. In contrast, the baseline encoder has to model both semantic and modality information of speech inputs, which weakens its modeling capacity for modality and therefore makes it have a much lower performance on speaker verification.

6 Conclusions

In this paper, we have presented AdaST, a neural model dynamically adapting acoustic states in the decoder, which is able to mitigate the cross-lingual and cross-modal challenge for end-to-end speech translation. Experiments demonstrate that AdaST achieves an improvement of 1.18 BLEU points over state-of-the-art neural speech translation models.

Acknowledgments

The present research was partially supported by the National Key Research and Development Program of China (Grant No. 2019QY1802). We would like to thank the anonymous reviewers for their insightful comments.

References

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specaugment for end-to-end speech translation. *CoRR*, abs/1911.08876.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 58–68. Association for Computational Linguistics.

Oliver Bender, Richard Zens, Evgeny Matusov, and Hermann Ney. 2004. Alignment templates: the RWTH SMT system. In *2004 International Workshop on Spoken Language Translation, IWSLT 2004, Keihanna Science City, Kyoto, Japan, September 30 - October 1, 2004*, pages 79–84. ISCA.

Alexandre Berard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 6224–6228. IEEE.

Alexandre Berard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *CoRR*, abs/1612.01744.

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2020. Listen, understand and translate": Triple supervision decouples end-to-end speech-to-text translation. *CoRR*, abs/2009.09704.

Mattia Antonino Di Gangi, Matteo Negri, and Marco Turchi. 2019. One-to-many multilingual end-to-end speech translation. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pages 585–592. IEEE.

Hirofumi Inaguma, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. 2019. Multilingual end-to-end speech translation. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pages 570–577. IEEE.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. 2020. Espnet-st: All-in-one speech translation toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 302–311. Association for Computational Linguistics.

Sathish Reddy Indurthi, Houjeung Han, Nikhil Kumar Lakumarapu, Beomseok Lee, Insoo Chung, Sangha Kim, and Chanwoo Kim. 2019. Data efficient direct speech-to-text translation with modality agnostic meta-learning. *CoRR*, abs/1911.04283.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J. Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 7180–7184. IEEE.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yuchen Liu, Hao Xiong, Jiajun Zhang, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. 2019. End-to-end speech translation with knowledge distillation. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1128–1132. ISCA.

Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. 2019. Speech model pre-training for end-to-end spoken language understanding. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 814–818. ISCA.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. On the integration of speech recognition and statistical machine translation. In *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, September 4-8, 2005*, pages 3177–3180. ISCA.

Hermann Ney. 1999. Speech translation: coupling of recognition and translation. In *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '99, Phoenix, Arizona, USA, March 15-19, 1999*, pages 517–520. IEEE Computer Society.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Mihaela C. Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing ASR pretraining for low-resource speech-to-text translation. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 7909–7913. IEEE.

Tzu-Wei Sung, Jun-You Liu, Hung-yi Lee, and Lin-Shan Lee. 2019. Towards end-to-end speech-to-text translation with two-pass decoding. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 7175–7179. IEEE.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Chengyi Wang, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou. 2020a. Bridging the gap between pre-training and fine-tuning for end-to-end speech translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9161–9168. AAAI Press.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020b. Curriculum pre-training for end-to-end speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3728–3738. Association for Computational Linguistics.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2625–2629. ISCA.

OKGIT: Open Knowledge Graph Link Prediction with Implicit Types

Chandrahas
Indian Institute of Science, Bangalore
chandrahas@iisc.ac.in

Partha Pratim Talukdar
Indian Institute of Science, Bangalore
ppt@iisc.ac.in

Abstract

Open Knowledge Graphs (OpenKG) refer to a set of (head noun phrase, relation phrase, tail noun phrase) triples such as (tesla, return to, new york) extracted from a corpus using OpenIE tools. While OpenKGs are easy to bootstrap for a domain, they are very sparse and far from being directly usable in an end task. Therefore, the task of predicting new facts, i.e., link prediction, becomes an important step while using these graphs in downstream tasks such as text comprehension, question answering, and web search query recommendation. Learning embeddings for OpenKGs is one approach for link prediction that has received some attention lately. However, on careful examination, we found that current OpenKG link prediction algorithms often predict noun phrases (NPs) with incompatible types for given noun and relation phrases. We address this problem in this work and propose OKGIT that improves OpenKG link prediction using novel type compatibility score and type regularization. With extensive experiments on multiple datasets, we show that the proposed method achieves state-of-the-art performance while producing type compatible NPs in the link prediction task.

1 Introduction

An Open Knowledge Graph (OpenKG) is a set of factual triples extracted from a text corpus using Open Information Extraction (OpenIE) tools such as TEXTRUNNER (Banko et al., 2007) and ReVerb (Fader et al., 2011). These triples are of the form (noun phrase, relation phrase, noun phrase), e.g., (*tesla*, *return to*, *new york*). An OpenKG can be viewed as a multi-relational graph where the noun phrases (NPs) are the nodes, and the relation phrases (RPs) are the labeled edges between pairs of nodes. It is easy to bootstrap OpenKGs from a domain-specific corpus, making them suitable for newer domains. However, they are extremely sparse and may not be directly usable for an end task. Therefore, tasks such as NP canonicalization (merging mentions of the same entity) and link prediction (predicting new facts) become an important step in downstream applications. Some example applications are text comprehension (Mausam, 2016), relation schema induction (Nimishakavi et al., 2016), canonicalization (Vashishth et al., 2018), question answering (Yao and Van Durme, 2014), and web search query recommendation (Huang et al., 2016). In this work, we focus on improving OpenKG link prediction.

Triple	(*tesla, return to, ?*)				
CaRE	*polytechnic institute*	*2009*	*1986*	*jp morgan*	*patent*
BERT	*chicago*	*earth*	*england*	*america*	*detroit*
OKGIT	*new york*	*america*	*paris*	*california*	*london*

Table 1: Some sample tail NP predictions by CaRE, BERT, and OKGIT. The true tail NP is underlined. As we can see, both CaRE and BERT fail to predict the correct tail NP. However, BERT predictions are type compatible with the query. OKGIT predicts the correct NP while improving the type compatibility with the query.

Although OpenKGs are structurally similar to Ontological KGs, they come with a different set of challenges. They are extremely sparse, NPs and RPs are not canonicalized, and no type information is present for NPs. There has been much work on learning embeddings for Ontological KGs in the past years. However, this task has not received much attention in the context of OpenKGs. CaRE (Gupta et al., 2019) is a recent method which addresses this problem. It learns embeddings for NPs and RPs in an OpenKG while incorporating NP canonicalization information. However, even after incorporating canonicalization, we find that CaRE struggles to predict NPs whose types are compati-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2546–2559
August 1–6, 2021. ©2021 Association for Computational Linguistics

ble with given head NP and RP.

As observed by Petroni et al. (2019), modern pretrained language representation models like BERT can store factual knowledge and can be used to perform link prediction in KGs. However, in our explorations with OpenKGs, we found that even though BERT may not predict the correct NP on the top, it predicts type compatible NPs (Table 1). A similar observation was also made in the context of entity linking (Chen et al., 2020). As OpenKGs do not have any underlying ontology and obtaining type information can be expensive, BERT predictions can help improve OpenKG link prediction.

Motivated by this, we employ BERT for improving OpenKG link prediction, using novel type compatibility score (Section 4.2) and type regularizer term (Section 4.4). We propose OKGIT, a method for OpenKG link prediction with improved type compatibility. We test our model on multiple datasets and show that it achieves state-of-the-art performance on all of these datasets.

We make the following contributions:

- We address the problem of OpenKG link prediction, focusing on improving type compatibility of predictions. To the best of our knowledge, this is the first work that addresses this problem.
- We propose OKGIT, a method for OpenKG link prediction with novel type compatibility score and type regularization. OKGIT can utilize NP canonicalization information while improving the type compatibility of predictions.
- We evaluate OKGIT on the link prediction across multiple datasets and observe that it outperforms the baseline methods. We also demonstrate that the learned model generates more type compatible predictions.

Source code for the proposed model and the experiments from this paper is available at https://github.com/Chandrahasd/OKGIT.

2 Related Work

OpenKG Embeddings: Learning embeddings for OpenKGs has been a relatively under-explored area of research. Previous work using OpenKG embeddings has primarily focused on canonicalization. CESI (Vashishth et al., 2018) uses KG embedding models for the canonicalization of noun phrases in OpenKGs. The problem of incorporating canonicalization information into OpenKG embeddings was addressed by Gupta et al. (2019). Their method for OpenKG embeddings (i.e., CaRE) performs better than Ontological KG embedding baselines in terms of link prediction performance. The challenges in the link prediction for OpenKGs were discussed in Broscheit et al. (2020), and methods similar to CaRE were proposed. In spirit, CaRE (Gupta et al., 2019) comes closest to our model; however, they do not address the problem of type compatibility in the link prediction task.

Entity Type: Entity typing is a popular problem where given a sentence and an entity mention, the goal is to predict *explicit* types of the entity. It has been an active area of research, and many models and datasets, such as (Mai et al., 2018), (Hovy et al., 2006), and (Choi et al., 2018), have been proposed. However, unlike this task, we aim to incorporate unsupervised **implicit** type information present in the pre-trained BERT model into OpenKG embeddings, rather than predicting *explicit* entity types present in ontologies or corpora.

For unsupervised cases, the problem of type compatibility in link prediction was addressed in (Jain et al., 2018). They employ a type compatibility score by learning a type vector for each NP and two type vectors (head and tail) for each relation. This score is multiplied with the triple score function, and the type vectors are trained jointly with embedding vectors. Although their method addresses the type compatibility issue, it is based on Ontological KG embedding models and shares the same limitations. In another work (Xie et al., 2016), hierarchical type information available in the dataset is incorporated while learning embeddings. However, their model is suitable only for Ontological KGs where the type information is readily available.

BERT in KG Embedding: BERT architecture has been used for scoring KG triples (Yao et al., 2019; Wang et al., 2019). However, their methods work on Ontological KGs without any explicit attention to NP types. In other work (Petroni et al., 2019), pre-trained BERT models are used for predicting links in KG. However, their focus was to evaluate knowledge present in the pre-trained BERT models instead of improving the existing link prediction model. BERT embeddings were also used for extracting entity type information (Chen et al., 2020). However, it was used for Entity Linking compared to OpenKG link prediction in our case.

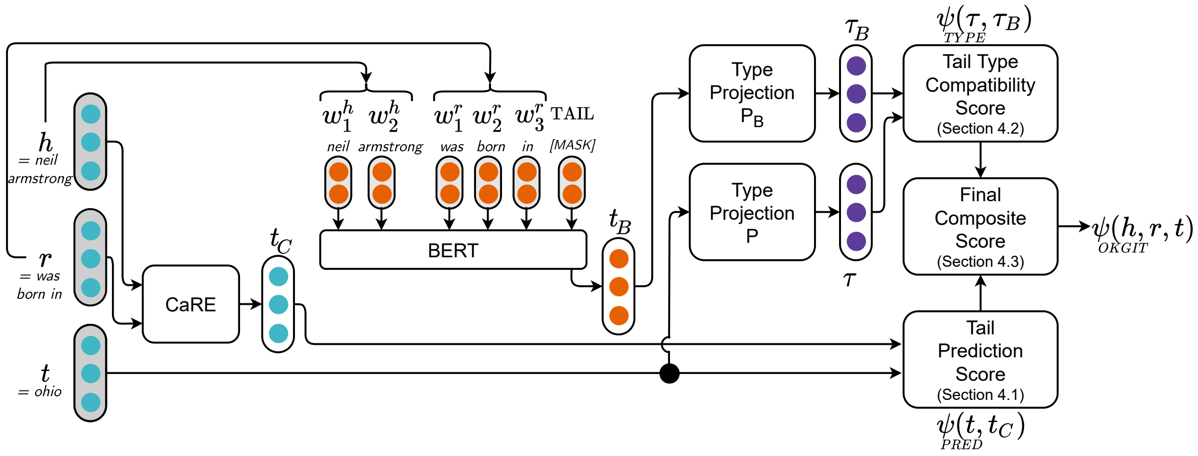

Figure 1: OKGIT Architecture. OKGIT learns embeddings for Noun Phrases (NP) and Relation Phrases (RP) present in an OpenKG by augmenting a standard tail prediction loss with type compatibility loss. Guidance for the tail type is obtained through type projection out of BERT's tail embedding prediction. In the figure, h, r, and t are the head NP, relation (RP), and tail NP. $h = w_1^h \dots w_{k_h}^h$ and $r = w_1^r \dots w_{k_r}^r$ are tokens in the head NP and relation, respectively. t_C and t_B are the tail NP vectors predicted by CaRE and BERT models (Please see Section 3 for background on these two models). Vectors τ_B and τ are the type vectors obtained using type projections P_B and P, respectively. ψ_{PRED} represents tail prediction score (Section 4.1) while ψ_{TYPE} represents type compatibility score (Section 4.2). ψ_{OKGIT} is the combined score generated by OKGIT for the input triple (h, r, t) (Section 4.3). Please refer to Section 4 for more details.

3 Background

We first introduce the notation used in this paper, followed by brief descriptions of BERT and CaRE.

Notation: An Open Knowledge Graph OpenKG = ($\mathcal{N}$, $\mathcal{R}$, $\mathcal{T}$) contains a set of noun phrases (NPs) $\mathcal{N}$, a set of relation phrases (RPs) $\mathcal{R}$ and a set of triples $(h, r, t) \in \mathcal{T}$ where $h, t \in \mathcal{N}$ and $r \in \mathcal{R}$. Here, h and t are called the head and tail NPs, and r is the RP between them. Each of them contains tokens from a vocabulary $\mathcal{V}$, specifically, $h = (w_1^h, w_2^h, \dots, w_{k_h}^h)$, $t = (w_1^t, w_2^t, \dots, w_{k_t}^t)$ and $r = (w_1^r, w_2^r, \dots, w_{k_r}^r)$. Here, k_h, k_r, and k_t are the numbers of tokens in the head NP, the relation, and the tail NP. OpenKG embedding methods learn vector representations for NPs and RPs. Specifically, vectors for an NP $e \in \mathcal{N}$ and an RP $r \in \mathcal{R}$ are represented by boldface letters $\mathbf{e} \in \mathbb{R}^{d_e}$ and $\mathbf{r} \in \mathbb{R}^{d_r}$. Here, d_e and d_r are dimensions of NP and RP vectors. Usually, $d_e = d_r$. A score function $\psi(h, r, t)$ represents the plausibility of a triple. Similarly, BERT represents tokens by d_B-dimensional vectors. A type projection matrix P takes the vectors to a common d_τ-dimensional type space $\mathbb{R}^{d_\tau}$. The vectors in the type space are denoted by τ.

BERT (Devlin et al., 2019): BERT is a bi-directional language representation model based on the transformer architecture (Vaswani et al., 2017), which has shown performance improvements across multiple NLP tasks. It is pre-trained on two tasks, (1) Masked Language Modeling (MLM), where the model is trained to predict randomly masked tokens from the input sentences, and (2) Next Sentence Prediction (NSP), where the model is trained to predict whether an input pair of sentences occurs in a sequence or not. In our case, we use a pre-trained BERT model (without fine-tuning) for predicting a masked tail NP in a triple.

CaRE (Gupta et al., 2019): CaRE is an OpenKG embedding method that can incorporate NP canonicalization information while learning the embeddings. NP canonicalization is the problem of grouping all surface forms of a given entity in one cluster, e.g., inferring that *Barack Obama*, *Barack H. Obama*, and *President Obama* all refer to the same underlying entity. CaRE consists of three components: (1) a canonicalization cluster encoder (CN), which generates NP embeddings by aggregating embeddings of canonical NPs from the corresponding cluster, (2) a bi-directional GRU based phrase encoder (PN), which encodes the tokens in RPs to generate RP embeddings, and (3) a base model, which is an Ontological KG embedding method like ConvE (Dettmers et al., 2018). It uses NP and

RP embeddings for scoring triples. These triple scores are then fed to a loss function (e.g., pairwise ranking loss with negative sampling (Bordes et al., 2013) or binary cross-entropy loss (BCE) (Dettmers et al., 2018)). In this paper, we use CaRE with ConvE as the base model. This model generates a candidate tail NP vector for a given NP h and RP r, denoted by CaRE($\mathbf{h}$, $\mathbf{r}$).

4 OKGIT: Our Proposed Method

Motivation: As illustrated in Table 1, top NPs predicted by CaRE may not always be type compatible with the input query. On the other hand, BERT's top predictions are usually type compatible (Chen et al., 2020), although they may not be factually correct. Thus, we hypothesize that a combination of these two models can produce correct as well as type compatible predictions. Motivated by this, we develop OKGIT, which combines the best of both of these models. The complete architecture of the proposed model can be found in Figure 1. In the following section, we present various components of the proposed model.

4.1 ψ_{PRED}: Tail Prediction Score

The correctness of tail prediction in a triple is measured by the triple score function ψ_{PRED}. Given a triple (h, r, t), it uses the corresponding vectors ($\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$) and assigns high scores to correct triples and low scores to incorrect triples. We follow CaRE (Gupta et al., 2019) for scoring triples, which internally uses ConvE (Dettmers et al., 2018) as the base model. For a given triple (h, r, t), the CaRE model first predicts a tail NP vector $\mathbf{t_C}$ as

$$\mathbf{t_C} = \text{CaRE}(\mathbf{h}, \mathbf{r}) \tag{1}$$

The predicted tail NP vector $\mathbf{t_C}$ is then matched against the given tail NP vector $\mathbf{t}$ using dot product to generate the triple score ψ_{PRED}.

$$\psi_{\text{PRED}}(\mathbf{t}, \mathbf{t_C}) = \mathbf{t_C}^\top \mathbf{t}. \tag{2}$$

The score ψ_{PRED} represents tail prediction correctness, and CaRE model uses only this score.

4.2 ψ_{TYPE}: Tail Type Compatibility Score

The type compatibility between a given (head NP, RP) pair and a tail NP is measured by the type compatibility score function ψ_{TYPE}. It assigns a high score when an NP t has suitable types as candidate tail NP for given head NP h and RP r. We employ a Masked Language Model (MLM) for measuring type compatibility, specifically BERT (Devlin et al., 2019). Following (Petroni et al., 2019), we can generate a candidate tail NP vector using BERT. Specifically, given a triple (h, r, t), we replace the head NP h and RP r with their tokens and tail NP t with a special MASK token. The resulting sentence $(w_1^h, \ldots, w_{k_h}^h, w_1^r, \ldots, w_{k_r}^r, \text{MASK})$ is sent as input to the BERT model. We denote the output vector from BERT corresponding to the MASK tail token as $\mathbf{t_B}$.

$$\mathbf{t_B} = \text{BERT}(h, r, \text{MASK}) \tag{3}$$

We can predict tail NPs for a given (h, r) by finding the nearest neighbors of $\mathbf{t_B}$ from the BERT vocabulary (Appendix D). These predicted NPs may not be the correct tail NP present in KG; however, they tend to be type compatible with the given (h, r) pair.

Motivated by this, we extract the implicit NP type information from this vector using a type projector $P_B \in \mathbb{R}^{d_\tau \times d_B}$. The output vector from BERT t_B is high-dimensional and can be used as a proxy for NP's type (Chen et al., 2020). Therefore, P_B projects the $\mathbf{t_B}$ vector to a lower dimensional space such that only relevant information is retained. We do a similar operation on tail NP embedding $\mathbf{t}$ and use a type projector $P \in \mathbb{R}^{d_\tau \times d_e}$ to extract type information. Both P_B and P are trained jointly with the model. Thus, the type vectors are given by

$$\boldsymbol{\tau_B} = P_B \mathbf{t_B} \quad \text{and} \quad \boldsymbol{\tau} = P\mathbf{t} \tag{4}$$

for BERT and CaRE, respectively. Here, both $\boldsymbol{\tau_B}, \boldsymbol{\tau} \in \mathbb{R}^{d_\tau}$. Then, the type compatibility score between these can be measured by negative of Euclidean distance, i.e.,

$$\psi_{\text{TYPE}}(\boldsymbol{\tau}, \boldsymbol{\tau_B}) = -||\boldsymbol{\tau_B} - \boldsymbol{\tau}||_2^2.$$

We also experimented with a dot product version of the type score, $\psi_{\text{TYPE}}^{\text{Dot}}(\boldsymbol{\tau}, \boldsymbol{\tau_B}) = \boldsymbol{\tau_B}^\top \boldsymbol{\tau}$, and found its performance to be comparable to the Euclidean distance version. Therefore, we use the Euclidean distance version for all our experiments.

4.3 ψ_{OKGIT}: Final Composite Score

The score functions ψ_{PRED} and ψ_{TYPE} may contain complementary information. Therefore, we use a combination of triple and type compatibility scores as final score for a given triple.

$$\psi_{\text{OKGIT}}(h, r, t) = \psi_{\text{PRED}}(\mathbf{t}, \mathbf{t_C}) + \gamma \times \psi_{\text{TYPE}}(\boldsymbol{\tau}, \boldsymbol{\tau_B}). \tag{5}$$

Dataset	#NPs	#RPs	#Gold Clusters	#Average NPs Per Cluster
ReVerb20K	11,064	11,057	10,897	1.02
ReVerb45K	27,007	21,622	18,626	1.45
ReVerb20KF	3,524	6,076	3,406	1.03
ReVerb45KF	9,400	11,249	6,749	1.39

	#Train	#Validation	#Test
ReVerb20K	15,498	1,549	2,324
ReVerb45K	35,969	3,597	5,394
ReVerb20KF	6,685	1,015	1,517
ReVerb45KF	14,775	1,781	2,650

Table 2: Dataset Statistics. Please refer to Section 5 for more details.

Please recall that $\mathbf{t_C}$ and $\boldsymbol{\tau_B}$ are in turn dependent on h and r ((1) and (3)), while $\boldsymbol{\tau}$ is dependent on t (4). Here, γ controls the relative weights given to individual scores. This final score takes care of both, i.e., triple correctness as well as type compatibility. For training, we feed the sigmoid of this score function to the Binary Cross Entropy (BCE) loss function following (Dettmers et al., 2018).

4.4 Learning with Type Regularization

Let $\mathcal{X} = \{(h_i, r_i) | (h_i, r_i, t_i) \in \mathcal{T} \text{for some } t_i \in \mathcal{N}\}$ be the set of all head NPs and RPs which appear in the OpenKG. Let y_i be the label for the triple (h_i, r_i, t_i) which is 1 if $(h_i, r_i, t_i) \in \mathcal{T}$ and 0 otherwise. We apply the logistic sigmoid function σ on score ψ_{OKGIT} to get the predicted label

$$\hat{y}_i = \sigma(\psi_{\text{OKGIT}}(h_i, r_i, t_i))$$

Finally, we use the following binary cross-entropy (BCE) loss for triple correctness.

$$\text{TripleLoss}(h_i, r_i, t_i) = y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

To further reinforce the type compatibility in the model, we include an additional loss term which forces the type vectors of correct triples to be closer in the type space. Similar to TripleLoss, we use the binary cross-entropy loss for type regularization as well. The type regularization term is shown below.

$$\text{TypeLoss}(h_i, r_i, t_i) = y_i \cdot \log(\hat{p}_i) + (1 - y_i) \cdot \log(1 - \hat{p}_i)$$

where $\hat{p} = \sigma(\psi_{\text{TYPE}}(\boldsymbol{\tau}, \boldsymbol{\tau_B}))$. The cumulative loss function is then given as below.

$$\sum_{i=1}^{n} \text{TripleLoss}(h_i, r_i, t_i) + \lambda \times \text{TypeLoss}(h_i, r_i, t_i) \tag{6}$$

where n is the number of training instances. We consider $\mathcal{X} \times \mathcal{N}$ as our training data where triples present in $\mathcal{T}$ have label 1 and rest have label 0.

Dataset	$d_e = d_r$	d_τ	λ	γ	BERT model
ReVerb20K	300	300	0.01	5.0	large
ReVerb45K	300	100	0.0	2.0	large
ReVerb20KF	300	300	0.001	5.0	base
ReVerb45KF	300	300	0.001	0.25	base

Table 3: Optimal Hyperparameter values. Please refer to Section 5 for more details.

5 Experiments

Datasets: Following (Gupta et al., 2019), we use two subsets of English OpenKGs created using ReVerb (Fader et al., 2011), namely ReVerb20K and ReVerb45K. We follow the same train-validation-test split for these datasets. As noted in (Petroni et al., 2019), predicting multi-token NPs using BERT could be challenging and it might require special pre-training (Joshi et al., 2020). To understand this difference, we create filtered subsets of these datasets such that they contain only single token NPs [1]. Specifically, we create ReVerb20KF (ReVerb20K-Filtered) and ReVerb45KF (ReVerb45K-Filtered) which contain only single token NPs. More details about these datasets can be found in Table 2.

Setup and hyperparameters: We use $d_e = d_r$ = 300 for NP and RP vectors. For other hyperparameters, we use grid-search and select the model based on MRR on validation split. For type vectors, we select d_τ from {100, 300, 500}. The weight for type regularization term λ is selected from the range $\{10^{-3}, 10^{-2} \ldots, 10^{1}\} \cup \{0\}$. Type composition weight γ is selected from {0.25, 0.5, 1.0, 2.0, 5.0}. For the language model, we try both BERT-base as well as BERT-large. The optimal values for hyperparameters are shown in Table 3. The experiments run for 1.5 hours (for filtered subsets) and 9 hours (for full datasets) on GeForce GTX 1080 Ti GPU.

6 Results

We evaluate the proposed model on the link prediction task. We follow the same evaluation process as in (Gupta et al., 2019). From our experiments, we try to answer the following questions:

1. Is OKGIT effective in the link prediction task? (Section 6.1)

---

[1] Please note that the single-token limitation is only valid for BERT, not for OKGIT (Appendix D).

	ReVerb20K					**ReVerb45K**				
Model	**MRR(%)↑**	**MR↓**	**Hits(%)↑**			**MRR(%)↑**	**MR↓**	**Hits(%)↑**		
			@1	**@3**	**@10**			**@1**	**@3**	**@10**
ConvE (Dettmers et al., 2018)	26.2	2177.0	20.2	29.1	36.3	18.4	6625.0	13.3	20.6	28.3
CaRE (Gupta et al., 2019)	30.6	851.1	24.4	33.1	41.7	32.0	1276.8	25.3	35.0	44.6
CaRE [BERT initialization]	31.6	837.0	24.8	35.0	44.2	31.2	925.5	24.2	34.4	44.3
OKGIT [Our model]	**35.9**	**527.1**	**28.2**	**39.4**	**49.9**	**33.2**	**773.9**	**26.1**	**36.3**	**46.4**

	ReVerb20KF					**ReVerb45KF**				
Model	**MRR(%)↑**	**MR↓**	**Hits(%)↑**			**MRR(%)↑**	**MR↓**	**Hits(%)↑**		
			@1	**@3**	**@10**			**@1**	**@3**	**@10**
BERT (Devlin et al., 2019)	4.9	1116.5	2.2	5.0	9.7	18.9	536.5	12.3	20.8	32.5
ConvE (Dettmers et al., 2018)	22.3	836.6	16.1	25.5	33.4	16.5	2398.1	10.9	18.9	27.6
CaRE (Gupta et al., 2019)	29.3	308.3	22.1	31.6	43.2	26.6	692.7	20.1	28.8	39.1
CaRE [BERT initialization]	31.8	207.6	24.2	34.8	46.2	24.9	557.3	17.8	27.6	38.3
OKGIT [Our model]	**34.6**	**214.7**	**26.5**	**38.0**	**50.2**	**29.7**	**500.2**	**22.5**	**32.4**	**43.3**

Table 4: Results of link prediction task. Here ↑ indicates higher values are better while ↓ indicates lower values are better. We can see that the OKGIT model outperforms the baseline models on all the datasets (Section 6.1).

2. Does OKGIT generate more type compatible NPs in link prediction? (Section 6.2)

3. Is the Type Projector effective in extracting type vectors from embeddings? (Section 6.3)

6.1 Effectiveness of OKGIT Embeddings in Link Prediction

We evaluate our model on the link prediction task. Given a held-out triple (h_i, r_i, t_i), all the NPs $e \in \mathcal{N}$ in the KG are ranked as candidate tail NP based on their score $\psi_{\text{OKGIT}}(h_i, r_i, e)$. Let the rank of the correct tail NP t be denoted by $rank_i^t$. Similarly, ranks are also calculated for predicting head NPs instead of tail NPs using inverse relations (Dettmers et al., 2018; Gupta et al., 2019); let it be denoted by $rank_i^h$. These ranks are then used to find Mean Reciprocal Rank (MRR), Mean Rank (MR) and Hit@k (k=1,3,10) as follows.

$$\text{MRR} = \frac{1}{2 \times n_{test}} \sum_{i=1}^{n_{test}} \left(\frac{1}{rank_i^h} + \frac{1}{rank_i^t} \right),$$

$$\text{MR} = \frac{1}{2 \times n_{test}} \sum_{i=1}^{n_{test}} \left(rank_i^h + rank_i^t \right), \text{ and}$$

$$\text{Hits@k} = \sum_{i=1}^{n_{test}} \frac{\mathbb{1}(rank_i^h \leq k) + \mathbb{1}(rank_i^t \leq k)}{2 \times n_{test}}.$$

Here, n_{test} is the number of test triples and $\mathbb{1}$ is the indicator function. As noted in (Gupta et al., 2019), ranking individual NPs is not suitable for OpenKGs due to the lack of canonicalization. Hence, following their approach, we rank gold canonicalization clusters instead of individual NPs. The gold canonicalization partitions the NPs into clusters such that NPs mentioning the same entity belong to the same cluster. For ranking these clusters, we first find ranks of all NPs $e \in \mathcal{N}$. Then for each cluster, we keep the NP with minimum rank as representative and discard others. The representative NPs are then ranked again and the new ranks are assigned to the corresponding clusters. The rank of the cluster containing the true NP is then used for evaluating the performance. For better readability, the MRR and Hits@k metrics have been multiplied by 100.

We compare OKGIT with BERT (MLM), ConvE (Ontological KGE) and CaRE (OpenKGE). We also compare against a version of CaRE where phrase embeddings have been initialized with BERT (CaRE [BERT initialization]). As we can see from the results in Table 4, the proposed model OKGIT outperforms baseline methods in link prediction task across all datasets. This suggests that the implicit type scores from BERT help in improving ranks of correct NPs. Moreover, OKGIT outperforms CaRE with BERT initialization, suggesting the importance of type projectors [2].

The performance gain is higher for ReVerb20K and ReVerb20KF (+5.3 MRR) than ReVerb45K and ReVerb45KF (+1.2 and +3.1 MRR) datasets. As we can see from Table 2, the number of NPs are very close to the number of gold clusters in the 20K

---

[2] Please refer to Appendix A for a detailed comparison.

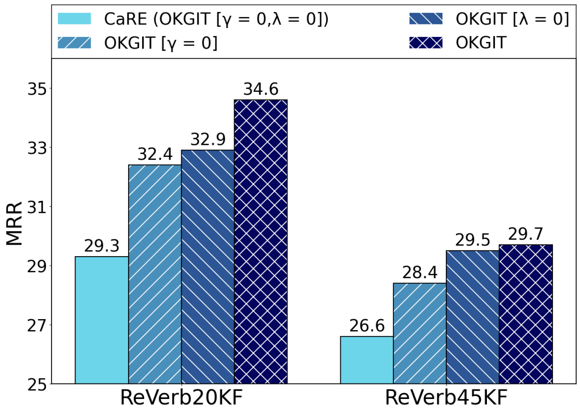

Figure 2: Effect of type compatibility score and type regularization on link prediction performance. While the type compatibility score with $\lambda = 0$ gives better gains in MRR (11%-12%) than type regularization term with $\gamma = 0$ (7%-11%), the combined model performs the best, achieving 12%-18% gains in MRR (Section 6.1).

Dataset	CaRE	OKGIT
ReVerb20KF	0.23	**0.30**
ReVerb20K	0.35	**0.36**
ReVerb45KF	0.22	**0.31**
ReVerb45K	0.34	**0.35**

Table 5: Results of type evaluation in CaRE and OKGIT predictions. We find that OKGIT performs better than CaRE in all datasets in terms of F1-score. Also, the results are statistically significant for all the datasets (Section 6.2).

datasets. Thus, the canonicalization information is slightly weaker in the 20K datasets than the 45K datasets. Due to this, CaRE achieved better gains in the ReVerb45K dataset as noted in (Gupta et al., 2019). This leaves more scope of improvements in the 20K datasets. By including the type information from BERT, OKGIT is able to fill this gap. It achieves better gains in the 20K datasets and is able to alleviate the lack of canonicalization information. Moreover, OKGIT is able to improve ranks of correct NPs ranked lower by CaRE. This can be seen by significant improvements in the MR.

Other Language Models: Using RoBERTa instead of BERT results in similar performance improvements (Appendix B). However, our primary focus is to understand the impact of *implicit type information* present in pre-trained MLMs, such as BERT, and *not* to compare multiple MLMs themselves.

Ablations: We perform ablation experiments to compare the relative importance of type compatibility score ψ_{TYPE} and type regularization term. We evaluate OKGIT with disabled type compatibility score (i.e., $\gamma = 0$ in Equation (5)) and disabled type regularization term (i.e., $\lambda = 0$ in Equation (6)) separately. Please note that CaRE model is equivalent to OKGIT with $\gamma = 0$ and $\lambda = 0$. The results of this experiment are shown in Figure 2. We find that while type compatibility score gives more performance gain (11%-12% gain in MRR) than type regularization (7%-11% gain in MRR), the combined model achieves the best performance (12%-18% gain in MRR). It suggests that both the components are important. Please refer to the Appendices A, B, C for more ablation experiments.

6.2 Type Compatibility in Predicted NPs

As noted in (Chen et al., 2020), BERT vectors contain NP type information[3]. OKGIT utilizes this type information for improving OpenKG link prediction. In this section, we evaluate whether OKGIT improves upon CaRE in predicting type compatible NPs. For such an evaluation, we require type annotations for the NPs in the OpenKGs. However, OpenKGs do not have an underlying ontology or explicit gold NP type annotations, making a direct evaluation impossible. Therefore, we employ a pre-trained entity typing model UFET (Choi et al., 2018). Given a sentence and an entity mention, the entity typing model predicts the mentioned entity's types. Using this model, we obtain types for true as well as predicted NPs by CaRE and OKGIT and use it for the evaluation. Please note that this evaluation is limited to the coverage and quality of the UFET model.

Evaluation Protocol: The type vocabulary in UFET model contains $10,331$ types including 9 general, 121 fine-grained, and $10,201$ ultra-fine types. The model takes a sentence $(w^h_1, \ldots, w^h_{k_h}, w^r_1, \ldots, w^r_{k_r}, w^t_1, \ldots, w^t_{k_t})$ formed from a triple (h, r, t) along with an entity mention (either t or h) as inputs and outputs a distribution over types. We use the top five predicted types for our experiments[4]. For a triple (h, r, t), we consider the types predicted for the true tail NP t as true types $\Gamma(t)$. Let $\hat{t}_{CaRE}$ and $\hat{t}_{OKGIT}$ be the top predicted tail NP by CaRE and OKGIT for the (h, r) pair. Then the types $\Gamma(\hat{t}_{CaRE})$ predicted for $\hat{t}_{CaRE}$

---

[3] We also verify this using Freebase, an ontological KG. Please refer to the Appendix G for more details.

[4] We observe similar behaviour with top one and three types.

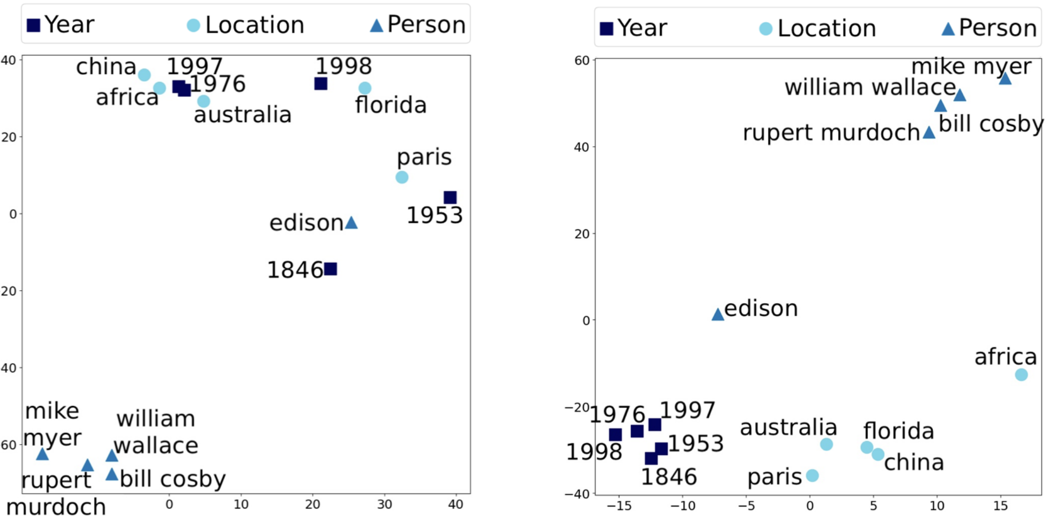

Figure 3: t-SNE projections of tail NP embeddings (left) and type vectors (right) extracted by the Type Projector from tail NP embeddings (Section 4.2) in the ReVerb20K dataset. We find that the Type Projector is able to extract informative type vectors from the tail embeddings. This is evident from the fact that the tail embeddings corresponding to person, location, and dates were inter-mixed in the left plot, while they have been separated into type specific clusters in the right plot. Please see Section 6.3 for details.

in the triple $(h, r, \hat{t}_{CaRE})$ is used as predicted types for CaRE. Similarly the types $\Gamma(\hat{t}_{OKGIT})$ predicted for $\hat{t}_{OKGIT}$ in the triple $(h, r, \hat{t}_{OKGIT})$ are used as predicted types for OKGIT. For evaluation, we calculate the mean F1-score as follows [5]

$$\text{F1} = \frac{2}{n_{test}} \sum_{i=1}^{n_{test}} \frac{|\Gamma(t_i) \cap \Gamma(\hat{t}_i)|}{|\Gamma(\hat{t}_i)| + |\Gamma(t_i)|}.$$

Here, $|\Gamma(t)|$ denotes the number of types present in $\Gamma(t)$ and $\hat{t}$ represents $\hat{t}_{CaRE}$ or $\hat{t}_{OKGIT}$. We can obtain the F1-scores for head NP similarly. We evaluate the mean F1-scores across head and tail NP prediction tasks on the test data and compare CaRE with OKGIT.

As we can see from the results in Table 5, OKGIT performs better than CaRE, suggesting that OKGIT generates more type compatible NPs than CaRE in the link prediction task. OKGIT achieves higher gains in the single-token datasets (i.e., ReVerb20KF and ReVerb45KF) than multi-token dataset (i.e., ReVerb20K and ReVerb45K). Upon investigation, we found that the types obtained using the entity typing model (true as well as predicted) for the multi-tokens datasets often contain common noisy types, leading to the small difference between CaRE and OKGIT. Following Dror et al. (2018), we also check the results for statistical significance using Permutation, Wilcoxon, and t-test with $\alpha = 0.05$, and found it to be significant for all the datasets.

6.3 Effectiveness of Type Projector

To better understand the effect of type projection, we visualize the vectors in NP-space from CaRE and Type-space (i.e., after type projection) from OKGIT. For this experiment, we randomly select 5 NPs from 3 categories, namely Person, Location and Year. More details about this selection process can be found in the Appendix E. We project the NP vectors (i.e., $\mathbf{t}$) corresponding to these NPs to a 2-dimensional NP-space using t-SNE (Maaten and Hinton, 2008)[6]. Similarly, we also project the corresponding type vectors (i.e., τ) to 2-dimensional Type-space. We plot the resulting vectors, color and shape coded by their respective categories, in Figure 3.

We can see that the vectors from different categories in the NP-space are mixed. However, after the type projection, the vectors in the Type-space are clustered together based on their categories.

6.4 Qualitative Evaluations

In this section, we present some examples of predictions made by CaRE and OKGIT methods. The result is shown in Table 6. As we see in Triple-1, both CaRE and OKGIT predict the correct NP (i.e.,

---

[5] Since we use a fixed number of types for ground truth and predictions, precision, recall, and F1-score have the same values. Therefore, we only report the F1-score.

[6] We run t-SNE for 2000 iterations with 15 perplexity.

	Triples	CaRE	OKGIT
1.	(*bach*, *moved to*, *?*)	*leipzig* *mobile* *vladimir h. horowitz* *yo yo*	*leipzig* *vienna* *stockholm* *sweden* *turin*
2.	(*clinton*, *lead by*, *?*)	*purchase* *sale* *video* *movie* *discount*	*1500* *260* *80* *99* *hire*

Table 6: Few example predictions made by CaRE and OKGIT models. We observe that the OKGIT predictions are more type compatible with the query. Please refer to Section 6.4 for more details.

leipzig) on top. However, more predictions from OKGIT are type compatible (i.e., all are locations) to the input query. On the other hand, CaRE predictions have mixed types (i.e., location, person, etc.). Also, CaRE makes an incorrect prediction, *vladimir horowitz*, possibly due to the presence of a training triple (*vladimir horowitz*, *had a great affinity for*, *bach*).

We see similar patterns in Triple-2, where the correct tail NP should be of type *number* indicating the count of votes. OKGIT is able to predict numbers in top predictions for Triple-2, while CaRE has mixed types in top predictions.

7 Conclusion

The task of link prediction for Open Knowledge Graphs (OpenKG) has been a relatively underexplored research area. Previous work on OpenKG embeddings has primarily focussed on improving or incorporating NP canonicalization information. While there are few methods for OpenKG link prediction, they often predict noun phrases with types incompatible with the query noun and relation phrases. Therefore, we use implicit type information from BERT to improve OpenKG link prediction and propose OKGIT. With the help of novel type compatibility score and type regularization term, OKGIT achieves significant performance improvement on the link prediction task across multiple datasets. We also find that OKGIT produces more type compatible predictions than CaRE, evaluated using an external entity typing model.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resource Development (Government of India).

Broader Impact

OKGIT is the first attempt towards incorporating implicit type information in OpenKG link prediction without human intervention. It will greatly benefit densification and applications of OpenKGs where no underlying ontologies are available.

However, OKGIT predictions depend on various datasets, i.e., the corpus used for training the masked language model (e.g., BERT) and the corpus from which the OpenKG triples were extracted. A potential, possibly undesirable, bias may be introduced in the predictions by manipulating these corpora or adding a large number of malicious triples in the OpenKG.

We have tested OKGIT in English datasets. While the overall model architecture is independent of the language, the model's effectiveness might vary depending upon the quality of the masked language model, and it needs to be tested.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, IJCAI'07, page 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD*, pages 1247–1250. AcM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NeurIPS)*, pages 1–9.

Samuel Broscheit, Kiril Gashteovski, Yanjie Wang, and Rainer Gemulla. 2020. Can we predict new facts with open knowledge graph embeddings? a benchmark for open link prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2296–2308, Online. Association for Computational Linguistics.

Shuang Chen, Jinpeng Wang, Feng Jiang, and Chin-Yew Lin. 2020. Improving entity linking by modeling latent entity type information. In *Proceedings of* *the AAAI Conference on Artificial Intelligence*, volume 34, pages 7529–7537.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. ACL.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Swapnil Gupta, Sreyash Kenkre, and Partha Talukdar. 2019. CaRe: Open knowledge graph embeddings. In *EMNLP-IJCNLP*, pages 378–388, Hong Kong, China. ACL.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *NAACL-HLT, Companion Volume: Short Papers*, pages 57–60, New York City, USA. ACL.

Zhipeng Huang, Bogdan Cautis, Reynold Cheng, and Yudian Zheng. 2016. Kb-enabled query recommendation for long-tail queries. In *CIKM*, CIKM '16, pages 2107–2112, New York, NY, USA. ACM.

Prachi Jain, Pankaj Kumar, Mausam, and Soumen Chakrabarti. 2018. Type-sensitive knowledge base inference without explicit type supervision. In *ACL (Volume 2: Short Papers)*, pages 75–80, Melbourne, Australia. ACL.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605.

Khai Mai, Thai-Hoang Pham, Minh Trung Nguyen, Tuan Duc Nguyen, Danushka Bollegala, Ryohei Sasano, and Satoshi Sekine. 2018. An empirical study on fine-grained named entity recognition. In *COLING*, pages 711–722, Santa Fe, New Mexico, USA. ACL.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence*, pages 4074–4077.

Madhav Nimishakavi, Uday Singh Saini, and Partha Talukdar. 2016. Relation schema induction using tensor factorization with side information. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 414–423, Austin, Texas. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473, Hong Kong, China. ACL.

Shikhar Vashishth, Prince Jain, and Partha P. Talukdar. 2018. CESI: canonicalizing open knowledge bases using embeddings and side information. In *WWW 2018, Lyon, France, April 23-27, 2018*, pages 1317–1327.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, pages 5998–6008. Curran Associates, Inc.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019. COKE: Contextualized Knowledge Graph Embedding. *arXiv preprint arXiv:1911.02168*.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, IJCAI'16, page 2965–2971. AAAI Press.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *ArXiv*, abs/1909.03193.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *ACL (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland. ACL.

Appendices

A BERT Initialization vs Type Projectors

Here, we demonstrate the importance of type projectors by comparing OKGIT with multiple BERT-augmented versions of CaRE. Specifically, we initialize the phrase and word embeddings in CaRE with a pre-trained BERT model. The phrase (word) is passed as input to BERT and the output corresponding to the $[CLS]$ token is then used for initializing phrase (word) embedding for CaRE model. This modified CaRE model is trained similar to the base CaRE model. Based on different initialization methods, we experiment with following baselines.

CaRE [BERT NP]: NP embeddings are initialized using BERT and the rest of the model is same as CaRE. This model uses 768 (for BERT-base) or 1024 (for BERT-large) dimensional vectors.

CaRE [BERT NP+PROJ]: Since CaRE [BERT NP] uses higher dimensional vectors (768 or 1024) compared to other methods (300), the comparison may not be fair. To address this issue, we project BERT embeddings to 300 dimension. The projection is trained with the rest of the model.

CaRE [BERT NP+RP]: We initialize NP embeddings as well as the word embeddings in RP encoder using BERT embeddings. This method also uses 768 or 1024 dimensional vectors[7].

In all the methods, including OKGIT, we never fine-tune BERT, as our goal is to evaluate the type information already present in pre-trained BERT model. We experiment with both, BERT-base and BERT-large, and report the best performing model.

As we can see from the results in Table 7, OKGIT outperforms these baselines. Although BERT initialization improves the performance of CaRE model, the usage of explicit type-score and type regularization leads to significant performance improvements, suggesting their importance.

B Replacing BERT with other operations

In this section, we evaluate whether BERT module in OKGIT can be replaced by simple operations such as vector addition and concatenation. Specifically, we modify $\mathbf{t_B}$ in Equation (3) by replacing BERT with these operations leading to the following variants of OKGIT.

OKGIT-C: BERT is replaced by concatenation of head NP vector $\mathbf{h}$ and relation phrase vector $\mathbf{r}$

$$\mathbf{t_B} = [\mathbf{h}; \mathbf{r}].$$

OKGIT-A: BERT is replaced by vector addition

$$\mathbf{t_B} = \mathbf{h} + \mathbf{r}.$$

OKGIT-R: We also experiment with another masked language model RoBERTa in place of BERT.

$$\mathbf{t_B} = \text{RoBERTa}(h, r, \text{MASK}).$$

For this experiment, we use the ReVerb20KF and ReVerb45KF datasets as representatives. We perform grid-search with similar hyper-parameters as in Section 5 of the main paper and select the best model based on the MRR on the validation split. The results are reported in Table 8.

As we can see from the results, the OKGIT-C and OKGIT-A perform very similar to CaRE on both datasets. This suggests that the performance gains for OKGIT come from the BERT module. This observation is further reinforced because OKGIT-R results in similar improvements compared to CaRE as OKGIT. However, in all cases, we find that OKGIT with BERT outperforms other model variants.

C CaRE with Entity Typing

Entity typing is the task of predicting explicit types of an entity given a sentence and its mention. As we are interested in improving type compatibility of predictions in the link prediction task, we can also incorporate the output from an entity typing model. In this section, we explore this setting by replacing the BERT module in OKGIT with an entity typing model UFET from (Choi et al., 2018). Specifically, we replace the vector t_B in Equation (3) with the output of UFET representing the predicted probability distribution over types.

OKGIT(UFET) Model: The UFET model takes a sentence and an entity mention as input and produces a distribution over explicit set of types. In our case, the sentence is formed by concatenating the subject NP, relation phrase, and object NP, while the object NP is used as mention. The output distribution from UFET is used as t_B in our model. We call this version of the model as OKGIT(UFET) and compare it CaRE and OKGIT.

We run a grid-search for finding the best hyper-parameter similar to Section 5 and report the results

---

[7] we also tried using pre-trained BERT as RP encoder in CaRE, however, it performed poorly due to fixed RP encoder.

Model	ReVerb20K					ReVerb45K				
	MRR(%)↑	MR↓	Hits(%)↑			MRR(%)↑	MR↓	Hits(%)↑		
			@1	@3	@10			@1	@3	@10
CaRE	30.6	851.1	24.4	33.1	41.7	32.0	1276.8	25.3	35.0	44.6
CaRE [BERT-L NP]	31.6	837.0	24.8	35.0	44.2	31.2	925.5	24.2	34.4	44.3
CaRE [BERT NP+PROJ][§]	27.4	950.2	21.9	29.2	38.0	30.7	952.8	23.0	34.4	45.5
CaRE [BERT-L NP+RP]	30.9	862.4	24.6	33.5	42.6	32.8	1015.6	25.9	35.9	45.6
OKGIT [Our model]	**35.9**	**527.1**	**28.2**	**39.4**	**49.9**	**33.2**	**773.9**	**26.1**	**36.3**	**46.4**

Model	ReVerb20KF					ReVerb45KF				
	MRR(%)↑	MR↓	Hits(%)↑			MRR(%)↑	MR↓	Hits(%)↑		
			@1	@3	@10			@1	@3	@10
CaRE	29.3	308.3	22.1	31.6	43.2	26.6	692.7	20.1	28.8	39.1
CaRE [BERT-B NP]	31.8	207.6	24.2	34.8	46.2	24.9	557.3	17.8	27.6	38.3
CaRE [BERT-L NP+PROJ]	27.6	258.6	21.0	29.1	40.7	24.7	600.5	17.4	27.4	39.2
CaRE [BERT-L NP+RP]	30.1	289.3	22.7	32.8	43.3	26.8	562.5	19.8	29.7	39.8
OKGIT [Our model]	**34.6**	**214.7**	**26.5**	**38.0**	**50.2**	**29.7**	**500.2**	**22.5**	**32.4**	**43.3**

Table 7: Results of the link prediction task. Here ↑ indicates higher values are better while ↓ indicates lower values are better. We can see that the OKGIT model outperforms the baseline models on all the datasets (Appendix A). Here, BERT-B and BERT-L denote BERT-base and BERT-large respectively. [§] For NP+PROJ models, BERT-large performs best for ReVerb20K, while BERT-base performs best for ReVerb45K.

Model	ReVerb20KF					ReVerb45KF				
	MRR(%)↑	MR↓	Hits(%)↑			MRR(%)↑	MR↓	Hits(%)↑		
			@1	@3	@10			@1	@3	@10
CaRE (Gupta et al., 2019)	29.3	308.3	22.1	31.6	43.2	26.6	692.7	20.1	28.8	39.1
OKGIT-C [$\mathbf{t_B} = [\mathbf{h}; \mathbf{r}]$]	30.0	309.3	22.9	32.4	43.8	27.1	666.5	20.2	29.8	39.9
OKGIT-A [$\mathbf{t_B} = \mathbf{h} + \mathbf{r}$]	30.4	331.7	23.5	32.9	43.6	27.1	660.5	19.9	30.6	40.2
OKGIT-R [RoBERTa]	32.7	221.0	25.3	35.1	46.5	29.0	596.7	21.8	32.0	43.0
OKGIT [Our model]	**34.6**	**214.7**	**26.5**	**38.0**	**50.2**	**29.7**	**500.2**	**22.5**	**32.4**	**43.3**

Table 8: Results of the ablation experiments. We replace the BERT module from OKGIT with simple operations such as vector addition (OKGIT-A) and vector concatenation (OKGIT-C). We also use RoBERTa in place of BERT(OKGIT-R). As we can see, replacing BERT with simple operations result in performance similar to CaRE. However, we do see better gains with RoBERTa, which performs better than CaRE and similar to OKGIT for ReVerb45KF. For all datasets, the OKGIT model outperforms other variants (Appendix B).

Model	ReVerb20KF					ReVerb45KF				
	MRR(%)↑	MR↓	Hits(%)↑			MRR(%)↑	MR↓	Hits(%)↑		
			@1	@3	@10			@1	@3	@10
CaRE	29.3	308.3	22.1	31.6	43.2	26.6	692.7	20.1	28.8	39.1
OKGIT (UFET)	8.8	1208.2	6.9	9.6	11.0	4.9	1156.8	1.5	4.0	11.5
OKGIT [Our model]	**34.6**	**214.7**	**26.5**	**38.0**	**50.2**	**29.7**	**500.2**	**22.5**	**32.4**	**43.3**

Table 9: Comparison of OKGIT with OKGIT(UFET). We can see that including UFET model in the system hurts the performance of the model (Appendix C).

on ReVerb20KF and ReVerb45KF datasets. The results are presented in the Table 9. As we can see from the results, OKGIT(UFET) performs poorly, even when compared to CaRE. It suggests that explicit type vectors from UFET model does not help in the link prediction task.

D BERT as Link Prediction Model

As mentioned in Section 4.2, t_B from Equation (3) can be used for predicting tail NPs by finding nearest neighbors in BERT vocabulary. However, this approach has a limitation. This model can only predict NPs that are single token and present in BERT vocabulary, restricting its applicability.

This limitation, however, is not valid for OKGIT. In OKGIT, the vector t_B is used for computing tail type compatibility score, instead of predicting tail NPs. Therefore, it is not restricted to BERT vocabulary or single-token NPs. As shown in Table 4, OKGIT is equally effective for single-token

	ReVerb20K					**ReVerb45K**				
Model	**MRR(%)↑**	**MR↓**	**Hits(%)↑**			**MRR(%)↑**	**MR↓**	**Hits(%)↑**		
			@1	**@3**	**@10**			**@1**	**@3**	**@10**
CaRE (Gupta et al., 2019)	30.7	879.2	24.4	33.5	41.7	32.9	1325.1	26.1	36.2	45.4
OKGIT [Our model]	**34.3**	**609.4**	**27.0**	**37.1**	**47.4**	**34.1**	**820.2**	**26.7**	**37.5**	**47.5**
	ReVerb20KF					**ReVerb45KF**				
Model	**MRR(%)↑**	**MR↓**	**Hits(%)↑**			**MRR(%)↑**	**MR↓**	**Hits(%)↑**		
			@1	**@3**	**@10**			**@1**	**@3**	**@10**
CaRE (Gupta et al., 2019)	28.0	326.0	21.2	30.5	41.3	28.0	683.8	21.0	31.4	41.3
OKGIT [Our model]	**31.7**	**258.1**	**24.2**	**34.0**	**46.3**	**31.2**	**483.3**	**23.8**	**34.4**	**45.4**

Table 10: Results of link prediction task on the validation split. We can see that the OKGIT model outperforms the baseline models on all the datasets (Appendix F).

datasets (e.g., ReVerb20KF and ReVerb45KF) and multi-token datasets (e.g., ReVerb20K and ReVerb45K).

E Selection of NPs for t-SNE

The OpenKGs do not have type annotations for the NPs. Therefore, we manually annotated a set of NPs and visualized a random subset. For this process, we first list all the NPs and shuffle them. Then we scan this list and note the first fifteen person names, locations, and years. Later, we select five NPs from each of these categories randomly and use them for the evaluation.

F Link Prediction Performance on Validation Split

The performance of CaRE and OKGIT on validation data on the link prediction task can be found in Table 10. These performance corresponds to the respective models which were used to report results in Table 4 of the main paper.

G Type Information in BERT Predictions

Our proposed OKGIT model is based on the hypothesis that BERT vectors (i.e., $\mathbf{t_B}$ in Equation (3) in Section 4.2) contain implicit type information. In this section, we evaluate this hypothesis that BERT vectors contain type information. It should be noted that evaluating OKGIT model for predicting NP types is not the goal here. We are interested in understanding whether pre-trained BERT vectors have sufficient type information, measured with respect to some existing anchors.

Evaluation Method: For this experiment, we use Freebase (Bollacker et al., 2008) which contains explicit gold type information for entities. Specifically, we use FB15K dataset (Bordes et al., 2013). We use the data from (Yao et al., 2019) for converting symbolic names in FB15k to textual descriptions. We only consider the subset of triples in FB15k which has single token in the tail node as BERT can only predict single token NPs.[8] This results in $n_T = 95,782$ triples. For type information, we use the data from (Xie et al., 2016). It contains 61 primary types (e.g., */award*). Please note that each node in FB15k can have multiple types. For a triple (h, r, t), we consider the types associated with the true tail NP t as true types $\Gamma(t)$. We then pass tokenized head NP and RP to BERT and find the top prediction $\hat{t} = \text{BERT}(h, r, \text{MASK})$ for tail position. The set of types associated with the predicted NP $\hat{t}$, denoted by $\Gamma(\hat{t})$, is then used as the predicted types. For evaluation, we calculate the following metrics

$$\text{Precision} = \frac{1}{n_T}\sum_{i=1}^{n_T}\frac{|\Gamma(t_i) \cap \Gamma(\hat{t}_i)|}{|\Gamma(\hat{t}_i)|},$$

$$\text{Recall} = \frac{1}{n_T}\sum_{i=1}^{n_T}\frac{|\Gamma(t_i) \cap \Gamma(\hat{t}_i)|}{|\Gamma(t_i)|}, and$$

$$\text{F1} = \frac{2}{n_T}\sum_{i=1}^{n_T}\frac{|\Gamma(t_i) \cap \Gamma(\hat{t}_i)|}{|\Gamma(\hat{t}_i)| + |\Gamma(t_i)|}.$$

Here, $|\Gamma(t)|$ and $|\Gamma(\hat{t})|$ denotes the number of types present in $\Gamma(t)$ and $\Gamma(\hat{t})$ respectively. [9] For comparison, we use the following baseline methods to assign types to a given (h, r, t).

---

[8] Please note that this limitation is only valid for BERT, not for OKGIT.

[9] Please note that, since we have gold type annotations available for Freebase, the number of true and predicted types need not be the same. Therefore, we evaluate precision and recall along with F1-scores.

Model	Precision	Recall	F1
Random	0.13	0.10	0.10
MFT	0.45	0.30	0.31
BERT	0.44	**0.40**	**0.36**
Human	**0.87**	0.18	0.27

Table 11: Results of the experiment to test whether BERT embeddings are rich with type information. As we can see, BERT outperforms other methods in terms of F1 score, suggesting that it contains relevant type information. Please refer to Appendix G for more details.

Random: assign $|\Gamma(\hat{t})|$ randomly selected types.
Most Frequent Types (MFT): assign $|\Gamma(\hat{t})|$ most frequent types.
Human: We also evaluate the type annotations provided by human annotators on randomly selected 100 triples. Each triple is exposed to three annotators and they are asked to provide types to the tail NP. Since most of the annotations contain one type for each triple, we take the union of the types provided by different annotators to compensate for Recall. For 69% of the triples, the annotators agreed on the same type.

To be fair with the automated baselines, we use the same number of predicted types as BERT (i.e., $|\Gamma(\hat{t})|$). A comparison with a pre-trained explicit entity typing methods, such as (Choi et al., 2018), is not applicable here as their type vocabulary is different. As we can see from the results in Table 11, BERT achieves best F1 score, suggesting that it contains type information. The Recall for Human is low since most of the annotations contained only one type, resulting in lower F1 score.

Multimodal Fusion with Co-Attention Networks for Fake News Detection

Yang Wu [1,2], Pengwei Zhan[1,2], Yunjian Zhang[1,2], Liming Wang[1]*, Zhen Xu[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
`{wuyang0419,zhanpengwei,zhangyunjian,wangliming,xuzhen}@iie.ac.cn`

Abstract

Fake news with textual and visual contents has a better story-telling ability than text-only contents, and can be spread quickly with social media. People can be easily deceived by such fake news, and traditional expert identification is labor-intensive. Therefore, automatic detection of multimodal fake news has become a new hot-spot issue. A shortcoming of existing approaches is their inability to fuse multimodality features effectively. They simply concatenate unimodal features without considering inter-modality relations. Inspired by the way people read news with image and text, we propose a novel Multimodal Co-Attention Networks (MCAN) to better fuse textual and visual features for fake news detection. Extensive experiments conducted on two real-world datasets demonstrate that MCAN can learn inter-dependencies among multimodal features and outperforms state-of-the-art methods.

1 Introduction

The rapid growth of social media has created fertile soil for the emergence and fast spread of fake news (Zhao et al., 2015), resulting in serious consequences. For example, during the U.S. 2016 presidential election, the most popular fake news was more widely spread than the most popular authentic news on Facebook, which confused people and broke the authenticity balance of the news ecosystem (Shu et al., 2017). To mitigate the negative effects caused by fake news, it is crucial to detect fake news on social media automatically.

Tweets with images are getting popular on social media recently, which have richer information and attract more viewers than tweets with only texts (Jin et al., 2017). Fake news also makes full use of this advantage to draw and mislead readers. Figure 1 shows three examples of fake news from Twitter.

Sharks in the mall! After the hurricane sandy!

Lenticular Clouds over Mount Fuji.

Woman, 36, gives birth to 14 children from 14 different fathers.

Figure 1: Some fake news from Twitter.

In the left example, both text and image indicate it is likely to be fake. The text of the middle one provides little evidence that it is fake news, but the image is obviously forged. In the right example, the image seems normal, while the textual contents indicate that it is probably fake. A hypothesis drawn from these examples is that combining text and the attached image is more conducive to detecting fake news.

Recent works have a growing interest in using multimodal (text + image) information to detect fake news. Jin et al.(2017) utilize local attention mechanisms to fuse features of image, text, and social context. Some studies explore to learn the joint representations of text and image, based on auxiliary adversarial networks (Wang et al., 2018) and variational autoencoders (Khattar et al., 2019). Nevertheless, they are not fine-grained enough in feature extraction and feature fusion. First, some studies require labor-intensive extra information, such as social context (Jin et al., 2017) and event category (Wang et al., 2018), to help detect fake news, which increases the cost of the detection. Second, except for texts in tweets, the methods mentioned above all focus on characteristics of images at the semantic level (e.g., emotional provocations), which can be reflected in the spatial domain. However, these methods ignore the individual information of fake images at the physical level, e.g., re-compression artifacts, which is reflected in the frequency domain (Qi et al., 2019). Third, some

---

*Corresponding Author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2560–2569
August 1–6, 2021.

models (Wang et al., 2018; Khattar et al., 2019) obtain fused representations by simply concatenating multi-modality features. Although leverages local attention mechanism, the attention values of att-RNN (Jin et al., 2017) are only obtained from joint textual-social representations, which cannot reflect the similarity between textual-social representations and visual representations. Intuitively, when people judge news credibility with text and image, they often observe image first and then read text (Wang et al., 2020). This process may be repeated several times. In this process, people understand image according to the textual information, and understand text according to the associated image information. So the information of one modality is conditionally fused with that of another modality for once or multiple times. Intuitively, there are inter-modality attention relations between image and text. However, existing state-of-the-art methods are weak to fuse multimodal features due to their neglect of inter-modality interactions.

To address the aforementioned challenges, we propose the Multimodal Co-Attention Networks (MCAN) for fake news detection by considering multimodal features. In our proposed model, we first extract spatial-domain features and frequency-domain features from image, as well as textual features from text. Then we develop a novel fusion approach with multiple co-attention layers to learn inter-modality relations, which fuses visual features first, and then the textual features. The fused representation obtained from the last co-attention layer is used for fake news detection.

The contributions of this paper can be summarized as follows: (1) We propose a novel end-to-end approach to detect fake news on social media only using the text and the attached image, without any extra information and auxiliary tasks. (2) The proposed MCAN model stacks multiple co-attention layers to fuse the multimodal features, which can learn inter-dependencies among them. (3) Our MCAN model is a general framework for fake news detection, and the components of MCAN are flexible. The sub-networks used to extract multimodal features can be replaced by different models. Moreover, the modular fusion process of MCAN allows our model to handle more modalities conveniently. (4) We evaluate MCAN on two large scale real-world datasets. The results demonstrate that our model outperforms the state-of-the-art models.

The rest of the paper is organized as follows: In Section 2, we summary previous related work on fake news detection. In Section 3, we detail our proposed model. The datasets, baselines, and experiment results are presented in Section 4. We conclude the study in Section 5.

2 Related Work

Following the previous work (Ruchansky et al., 2017; Shu et al., 2017), we specify that **fake news is the news that is intentionally fabricated and can be verified as false**. Existing methods for fake news detection can be divided into unimodal approaches and multimodal approaches.

2.1 Unimodal Fake News Detection.

Textual features are extracted from text content, including statistical features, such as the number of paragraphs in the text (Volkova et al., 2017), the percentage of negative words (Potthast et al., 2017; Bond et al., 2017), the number of punctuation and emojis (Castillo et al., 2011), and semantic features, such as writing styles (Chen et al., 2015) and language styles (Feng et al., 2012). However, these features are hand-designed, bringing bias and design difficulty. To address this problem, many studies use deep learning technologies, such as RNN (Ma et al., 2016), CNN (Yu et al., 2017), and GAN (Ma et al., 2019), to identify fake news. Their results show that deep learning methods perform better.

Visual features are important for news verification (Jin et al., 2016; Shu et al., 2017), such as clarity score (Jin et al., 2016), the number of images (Wu et al., 2015; Jin et al., 2016). However, these features are manually crafted and just learn simple patterns, hardly applying to real images. Qi et al. (2019) design a CNN-based model to capture image patterns, but their model only works in the case of large samples. So the applicable scope is very limited.

Social context features are born in the social connection between users and tweets, such as user profile and the number of posts. Liu et al. (2018) use user profiles on the news propagation path to determine the truth of the news. Some other works model propagation patterns as tree structures based on kernel methods (Wu et al., 2015; Ma et al., 2017). However, social context features are handcrafted, incomplete, and unstructured.

The above work embodies the limitations of unimodal features in detecting fake news. In this paper,

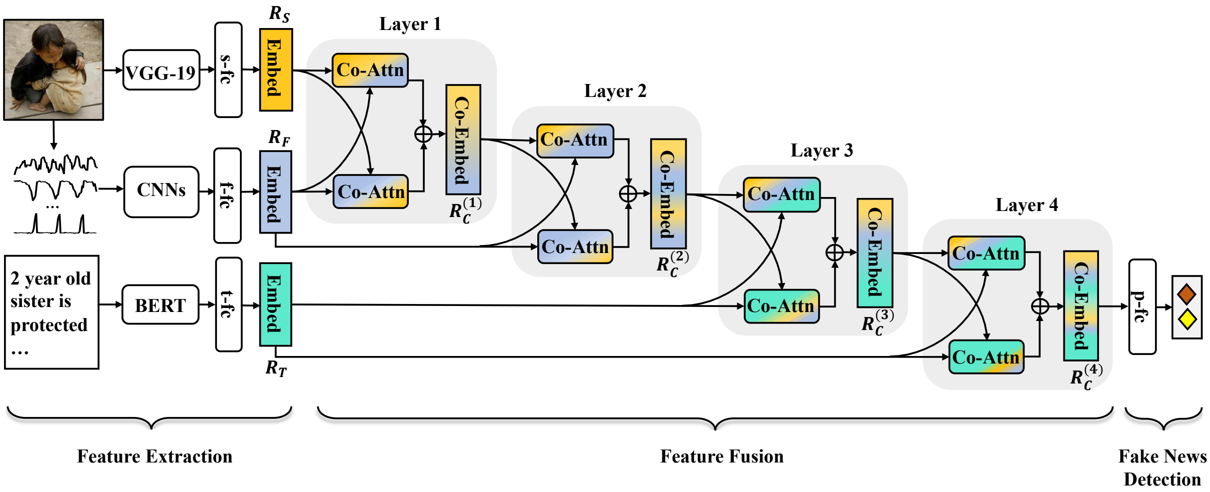

Figure 2: The architecture of our MCAN model.

we consider multiple modalities simultaneously when detecting fake news.

2.2 Multimodal Fake News Detection.

Recent works explore to fuse multimodal features. Jin et al. (2017) use local attention mechanism to fuse textual, visual, and social context features. Wang et al. (2018) learn event-invariant features by an aided adversarial network. Khattar et al. (2019) utilize autoencoders coupled with a detector to learn the shared representation of the text and the attached image. However, they ignore the characteristics of fake images at physical level (e.g., re-compression artifacts), and the fused features they learned lack correlations across multiple modalities.

To overcome the limitations of existing works, we propose MCAN to learn inter-dependencies among modalities. We extract spatial-domain and frequency-domain features of image, and textual features. Then we fuse them through a deep co-attention model inspired by a realistic scenario.

3 Methodology

3.1 Model Overview

Our model aims to learn multimodal fusion representations by considering dependencies across the modalities. As shown in Figure 2, the proposed model has three major procedures: feature extraction, feature fusion, and fake news detection.

Given news with text and image, we first utilize three different sub-models to extract features from spatial domain, frequency domain, and text. Then the multi-modality features are fused through a deep co-attention model, which consists of multiple co-attention layers. At last, the output of the co-attention model is used for judging the truth of the input news.

3.2 Feature Extraction

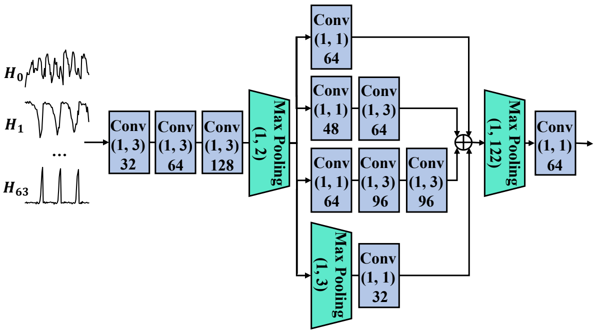

Figure 3: The detailed architecture of feature extraction in frequency domain.

Spatial-Domain Feature. To learn the semantic-level characteristics of the given image, we employ the VGG-19 network (Simonyan and Zisserman, 2014) to extract visual features from spatial domain. After the second of the last layer of VGG-19, we add a fully connected layer (denoted as "s-fc" in Figure 2) with ReLU activation function to generate a $d \times 1$ dimensional feature representation of the input image in spatial domain, which is denoted as $R_S \in \mathbb{R}^{d \times 1}$.

Frequency-Domain Feature. Fake-news images are often re-compressed images or tampered images that show periodicity in frequency domain (Qi et al., 2019), which can be easily captured by CNNs. Thus we design a CNN-based sub-network to extract features from frequency domain, as in Figure 3. The image is transformed from spatial domain to frequency domain through discrete cosine transform (DCT) as in Qi et al. (2019). After that, we obtain 64 vectors $H_0, H_1, \ldots, H_{63}$, which are then sampled to the fixed size 250. For parallel computation, we pick 64 250-dimensional vectors into a matrix $H_F \in \mathbb{R}^{(64\times 250)}$, which is fed to the CNN-based network later. The CNN-based subnetwork consists of a major network along with multi-branch networks. The earlier parts of the major network have three convolutional blocks and a max-pooling layer. The multi-branch networks are the same as architectures in Inception V3 (Szegedy et al., 2016). The last parts of the major network are a max-pooling layer followed by a convolutional block. Each convolutional block is comprised of a two-dimensional convolutional layer with batch normalization and ReLU activation function. After adding a fully connected layer with ReLU activation function (denoted as “f-fc” in Figure 2), we obtain the feature representation of the image in frequency domain $R_F \in \mathbb{R}^{d\times 1}$.

Textual Feature. The text content of the tweet is a sequential list of words denoted as $T = [T_1, T_2, \ldots, T_n]$, where n is the number of words in a tweet, and each word $T_i \in T$ is tokenized by a pre-prepared vocabulary (Devlin et al., 2018). Recently, the BERT model (Devlin et al., 2018) which is pre-trained on a large language corpus, has been proven to perform very well in multiple natural language processing tasks. Thus we utilize BERT to obtain the aggregate sequence representation as textual features we desired. The textual feature is then resized to be a $d \times 1$ dimensional representation (denoted as R_T) by a fully connected layer with ReLU activation function.

3.3 Feature Fusion

Intuitively, people often look at the image first and then read the text when reading the news with image and text. This process may be repeated several times, continuously fusing image and text information. Therefore, we develop a novel fusion approach to simulate this process. Before presenting the fusing process, we first introduce its basic unit, the co-attention (CA) block. We achieve feature fusion by cascaded stacking multiple CA layers, which consists of two parallelly connected CA blocks.

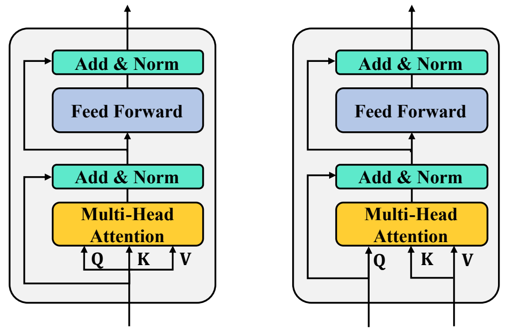

(a) Self-attention block (b) Co-attention block

Figure 4: Illustration of the self-attention block and the co-attention block.

Co-attention block. Co-attention block (Lu et al., 2019) is a variant of the standard multi-head self-attention (MSA) block (Vaswani et al., 2017), which can capture global dependencies of all positions in a sequence and is widely used in NLP and VQA tasks (Nguyen and Okatani, 2018; Gao et al., 2019). The MSA block showed in Figure 4(a) consists of a multi-headed self-attention function and a fully connected feed-forward network, both wrapped a residual connection followed by layer normalization. The input of MSA is first used to compute $(d \times 1)$-dimensional queries, keys, and values packed into matrixes Q, K, V, respectively. The similarity of the dot product between Q and K determines the attention distribution on the V. Multi-head attention function with m heads has m self-attention functions in parallel. For the i-th head, the inputs are transformed from Q, K, and V as follow:

$$Q_i = QW_i^Q,\ K_i = KW_i^K,\ V_i = VW_i^V \tag{1}$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{1\times d_h}$ are the projection matrices for the i-th head, $d_h = d/m$ is the dimensionality of the output feature of each head.

The calculation process of multi-head self-attention function can be presented as follows:

$$\begin{aligned} \mathrm{MA}(Q, K, V) &= \mathrm{h}\, W^O \\ \mathrm{h} &= \mathrm{h}_1 \oplus \mathrm{h}_2 \oplus \ldots \oplus \mathrm{h}_\mathrm{m} \\ \mathrm{h}_i = \mathrm{A}(Q_i, K_i, V_i) &= \mathrm{softmax}(\frac{Q_i K_i^T}{\sqrt{d_h}})V_i \end{aligned} \tag{2}$$

where $W^O \in \mathbb{R}^{md_h\times 1}$, $\oplus$ denotes concatenation of vectors.

The fully connected feed-forward network consists of two linear transformations with a ReLU activation function in between,

$$\mathrm{FFN}(x) = \max(0, xW_1)W_2 \tag{3}$$

where the dimensionality of input and output is $d \times 1$, and the inner-layer dimensionality is d_{ff}.

The co-attention block (denoted as ”Co-Attn”) is extended from the MSA block, as shown in Figure 4(b). For a Co-Attn block, the queries are from one modality while keys and values are from another modality. Especially, the query matrice is used as a residual item after the multi-head attention sublayer. The rest architectures are the same as MSA. The Co-Attn block produces an attention-pooled feature for one modality conditioned on another modality. If Q comes from text and k and V come from the attached image, the attention value calculated using Q and K can be used as a measure of the similarity between the text and image, and then weights the image. Just like humans, after reading the text, they will pay more attention to the areas in the image that are similar to the text. We believe that co-attention can simulate this process and learn inter-dependencies between different features.

Co-attention layer. We obtain a CA layer by connecting two Co-Attn blocks in parallel, as shown in Figure 2. Giving two Co-Attn blocks different features, the CA layer computes queries, keys, and values for each Co-Attn block as in a MSA block. Then the keys and values of one Co-Attn block are passed as input to another Co-Attn block. The outputs of two Co-Attn blocks are concatenated together and then fed into a fully connected layer to get the fused representation. The CA layer models dense interactions between input modalities by exchanging their information.

Multiple co-attention stacking. In order to fuse multimodal features deeply, we stack 4 CA layers in depth. The fusion process is progressive, and the output of each CA layer is one of the inputs of the next layer (see Figure 2). We first fuse spatial-domain representation R_S and frequency-domain representation R_F in first CA layer and obtain $R_C^{(1)}$. Then R_F are enhanced to fuse with $R_C^{(1)}$ in the second CA layer which outputs $R_C^{(2)}$. In the third and fourth layers, the inputs are the output of the previous layer and text representation R_T, and outputs are $R_C^{(3)}$ and $R_C^{(4)}$, respectively. The output vector of each CA layer is d-dimensional.The calculation processes are formulated as follows. Due to the page limit, we only show the calculation processes in the first CA layer and skip the repeated calculation details of other layers.

$$R_{C_{S\leftarrow F}} = R_S + \mathrm{MA}(R_S, R_F, R_F) \tag{4}$$

$$R_{C'_{S\leftarrow F}} = R_{C_{S\leftarrow F}} + \mathrm{FFN}(R_{C_{S\leftarrow F}})) \tag{5}$$

$$R_{C_{F\leftarrow S}} = R_F + \mathrm{MA}(R_F, R_S, R_S) \tag{6}$$

$$R_{C'_{F\leftarrow S}} = R_{C_{F\leftarrow S}} + \mathrm{FFN}(R_{C_{F\leftarrow S}}) \tag{7}$$

$$R_C^{(1)} = (R_{C'_{S\leftarrow F}} \oplus R_{C'_{F\leftarrow S}})W_C^{(1)} \tag{8}$$

where $R_{C'_{S\leftarrow F}} \in \mathbb{R}^d$ is the attention-pooled feature for spatial domain conditioned on frequency domain, $R_{C'_{F\leftarrow S}} \in \mathbb{R}^d$ is the attention-pooled feature for frequency domain conditioned on spatial domain, and $W_C^{(1)} \in \mathbb{R}^{2d\times d}$ is the projection matrice of the first CA layer. $R_C^{(1)}$ is transformed to be a $(d \times 1)$-dimensional representation before being input to the next CA layer. Specifically, the first and the third CA layers share parameters, and the second and the fourth CA layers share parameters.

3.4 Model Learning

We have obtained the multimodal feature representation $R_C^{(4)}$ fused features of text, spatial domain, and frequency domain. Let $f = R_C^{(4)}$, which is used to predict. The output of the proposed MCAN is the probability of a tweet being fake:

$$\hat{y} = \mathrm{softmax}\left(\max(0, fW_f)W_s\right) \tag{9}$$

where W_f is parameters of the fully connected layer, and W_s is parameters of the linear layer in the softmax layer. The loss function is devised to minimize the cross-entropy value:

$$L(\Theta) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y}) \tag{10}$$

where y is the ground truth, with 1 representing fake news and 0 representing real news, and Θ denotes all learnable parameters in the proposed model.

4 Experiments

4.1 Datasets

To evaluate the effectiveness of the proposed MCAN, we conduct experiments on two public real-world datasets, which are collected from Twitter and Weibo, respectively. The Twitter dataset was released for Verifying Multimedia Use task at MediaEval (Boididou et al., 2016). The Weibo dataset is collected by Jin et al. (2017). In the

	Twitter	Weibo
# of fake news	8199	4211
# of real news	6681	3639
# of images	512	7850

Table 1: Statistics of two datasets.

Weibo dataset, the real news is verified by an authoritative news agency in China, Xinhua News Agency. The fake news is verified by the official rumor debunking system of Weibo. The tweets in each dataset contain texts, attached images/videos, and social context information. In this work, we focus on text and image information. So we remove the tweets with videos and the tweets without texts or images. In Twitter dataset, 512 images are shared by the remaining data. When preprocessing the Weibo dataset, the steps we used are similar to that in the work (Jin et al., 2017). We keep the same data split scheme as the benchmark on these two datasets. The detailed statistics of the two datasets are listed in Table 1.

4.2 Experimental Settings

The max length of the text is 25 on Twitter and 160 on Weibo. The hidden size of ”s-fc”, ”f-fc” and ”t-fc” are 256. We set d=256, m = 4, and d_{ff} = 512. The hidden size of ”p-fc” is 35. The parameters of VGG-19 and BERT are frozen when training on Twitter dataset due to overfitting, but not on Weibo dataset. The BERT model used on Twitter dataset is multilingual cased BERT-based model and the one used on Weibo dataset is Chinese BERT-based model. Our proposed model is trained for 100 epochs with early stopping. We use Adam (Kingma and Ba, 2014) and AdaBelief (Zhuang et al., 2020) as optimizers on Twitter and Weibo datasets, respectively, to seek the optimal parameters of our model. The optimal hyperparameters of our model are determined by grid searching, and the selection criterion is accuracy. The hyperparameters of baselines are the same as those in respective studies.

4.3 Baselines

To validate the effectiveness of MCAN, we choose two categories of baseline models: unimodal models and multimodal models, which are listed as follows: (1) **Text**: a BERT model coupled with the decision network in MCAN, using textual information. (2) **Spatial**: a model consists of a VGG-19 model and the decision network of MCAN, utilizing image information in spatial domain. (3) **Freq**: proposed MCAN only has the part of dealing with frequency-domain features. (4) **VQA** (Antol et al., 2015): a model aims to answer questions according to the given images. For fair comparisons, we use a one-layer LSTM. (5) **NeuralTalk** (Vinyals et al., 2014): a deep recurrent framework for image caption. The joint representation of image and text is obtained by averaging the output of RNN at each timestep. (6) **att-RNN** (Jin et al., 2017): att-RNN utilizes local attention to fuse textual, visual, and social context features. For a fair comparison, we remove the part dealing with social context information. (7) **EANN** (Wang et al., 2018): A neural network based on the adversarial idea to remove the event-specific features. In EANN, event identification is an auxiliary task, and event labels are not in original datasets. For a fair comparison, we removed the event discriminator. (8) **MVAE** (Khattar et al., 2019): MVAE learns shared representations of text and image using a variational autoencoder coupled with a binary classifier. We use the same model as in the original work (Khattar et al., 2019). (9) **MCAN-A**: MCAN without the part of fusing multimodal features. Spatial-domain features, frequency-domain features, and textual features are simply concatenated for prediction.

4.4 Performance Comparison

Table 2 shows the results of baselines and our proposed model on two datasets. We can observe that the proposed MCAN outperforms all the baselines over all metrics across two datasets.

There are many similar trends on the two datasets. MCAN-A performs better than unimodal models, which indicates that adding features usually improves model performance, but it is not always positively correlated. For example, Text on Weibo dataset is better than MCAN-A. After adding the process of multimodal fusion, our proposed MCAN beats MCAN-A and other multimodal models, which embodies our proposed feature fusion method is indeed better than the simple concatenation method.

There are also some differences on the two datasets. The performance of Text (BERT) and Spatial (VGG-19) on Weibo dataset is much better than that on Twitter dataset. The reason is related to the dataset itself. On Weibo dataset, the average length of a tweet is about 10 times of that of a tweet

Dataset	Method	Accuracy	Fake News			Real News		
			Precision	Recall	F_1	Precision	Recall	F_1
Twitter	Text	0.633	0.656	0.762	0.705	0.587	0.459	0.515
	Spatial	0.671	0.841	0.527	0.648	0.574	0.864	0.69
	Freq	0.665	0.733	0.656	0.692	0.592	0.677	0.631
	VQA	0.631	0.765	0.509	0.611	0.55	0.794	0.65
	NeuralTalk	0.610	0.728	0.504	0.595	0.534	0.752	0.625
	att-RNN	0.664	0.749	0.615	0.676	0.589	0.728	0.651
	EANN	0.648	0.81	0.498	0.617	0.584	0.759	0.66
	MVAE	0.745	0.801	0.719	0.758	0.689	0.777	0.73
	MCAN-A	0.737	0.840	0.671	0.746	0.65	0.827	0.727
	MCAN	**0.809**	**0.889**	**0.765**	**0.822**	**0.732**	**0.871**	**0.795**
Weibo	Text	0.876	0.885	0.871	0.878	0.865	0.878	0.871
	Spatial	0.857	0.85	0.877	0.863	0.863	0.834	0.848
	Freq	0.717	0.728	0.724	0.726	0.706	0.710	0.708
	VQA	0.736	0.797	0.634	0.706	0.695	0.838	0.76
	NeuralTalk	0.726	0.794	0.713	0.692	0.684	0.840	0.754
	att-RNN	0.772	0.854	0.656	0.742	0.72	0.889	0.795
	EANN	0.782	0.827	0.697	0.756	0.752	0.863	0.804
	MVAE	0.824	0.854	0.769	0.809	0.802	0.875	0.837
	MCAN-A	0.869	0.868	0.879	0.874	0.869	0.857	0.863
	MCAN	**0.899**	**0.913**	**0.889**	**0.901**	**0.884**	**0.909**	**0.897**

Table 2: The results of different methods on two datasets

on Twitter dataset, which probably makes BERT perform better on Weibo dataset. Moreover, more than 70% of tweets on Twitter dataset are related to a single event. Thus, the training samples of BERT and VGG-19 are too similar, resulting in poor performance of model generalization. This is the reason why we fine-tuned BERT and VGG-19 on Weibo dataset but not on Twitter dataset. They are easy to overfit on Twitter dataset. But Weibo dataset has no such imbalanced issue.

On Weibo dataset, the accuracy of fine-tuned BERT and VGG-19 all exceed 85%. In this case, our proposed MCAN further improves the accuracy to close to 90% with the help of cascaded way of stacking CA layers. Comparing with the situation on Twitter dataset, we can find that our model performs better in the face of weak unimodal features. In our MCAN model, the representation ability of features can be greatly improved by effectively fusing other features.

4.5 Ablation Analysis

Quantitative Analysis. To evaluate the effectiveness of each component of the proposed MCAN, we remove each one from the entire model for comparison. "ALL" denotes the entire model MCAN with all components, including spatial-domain representation (S), textual representation (T), frequency-domain representation (F), and co-attention layers (A). After removing each one of them, we obtain the sub-models "-S", "-T", "-F" and "-A", respectively. "-F-A" denotes the reduced MCAN without both frequency-domain representation and co-attention layers. The results are exhibited in Figure 5.

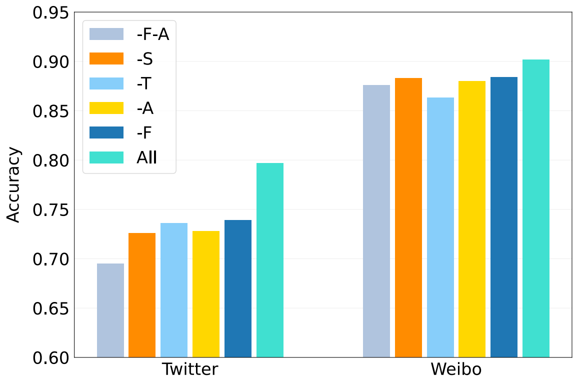

Figure 5: MCAN ablation analysis in Accuracy.

We can see that every component plays a significant role in improving the performance of MCAN. MCAN beats MCAN-F, which reveals that the frequency domain information is indeed helpful to detect fake news. On Twitter dataset, the contribution of textual representations to the entire model is less than that of visual representations, while the situation on Weibo dataset is opposite. This is still due to the imbalanced issue and the less average length of a tweet on Twitter dataset, which decrease the performance of the textual representation. Besides, on Weibo dataset, removing one or two components, the performance of MCAN does not drop significantly as on Twitter dataset. This benefits from balanced data distribution and the stability of fine-tuned BERT and VGG-19, as mentioned in Section 4.4.

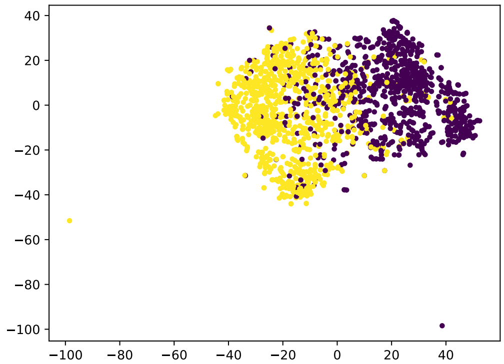

(a) MCAN-A

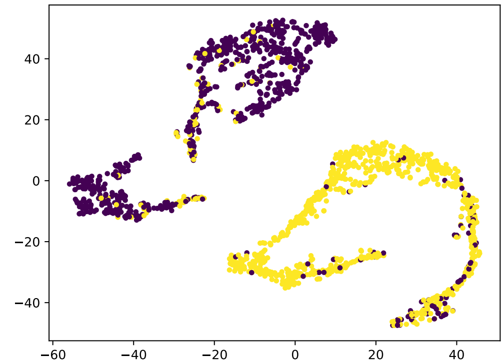

(b) MCAN

Figure 6: Visualizations of learned latent feature representations.

Qualitative Analysis. To illustrate the effectiveness of co-attention layers in MCAN, we qualitatively visualize the joint representation of three modalities learned by MCAN-A and the fused representation $R_C^{(4)}$ learned by MCAN on Weibo testing set with t-SNE (Maaten and Hinton, 2008), as shown in Figure 6. The label of each tweet is real or fake.

From Figure 6, we can observe that the separability of the feature representation learned by MCAN is much better than its reduced model MCAN-A. MCAN-A can learn discriminable features, but many features are still easily misclassified, showing in Figure 6(a). On the contrary, the features learned by MCAN are more discriminable with a more significant segregated area between two types of samples, as exhibited in Figure 6(b). This is attributed to the cascaded way of stacking co-attention layers in MCAN, which fuses the characteristics of multiple modalities deeply and boosts to distinguish fake news and real news.

From the above phenomena, we can conclude that the proposed method MCAN learns better and more distinctive feature representations with the co-attention layers, thus achieving better performance.

4.6 Case Studies

To further illustrate the importance of multimodal features for fake news detection, we compare the results reported by MCAN and unimodal models (Text and Spatial) and exhibit some fake news correctly captured by MCAN but missed by unimodal models.

Before washed away by flood, an Indian man calmly gave the last gesture to a photographer.

A group of dolphins brought a dog that fell into a canal to safe area.

Figure 7: Some fake news detected by MCAN but missed by Text on the Weibo dataset.

Figure 7 shows two top-confident tweets successfully detected by MCAN but missed by text-only MCAN. The textual contents of the two examples can provide little evidence that it is fake news. However, the two attached images seem forged pictures.

The water mantis lives in sewers. Its head has two to three times the poison of pufferfish and has no antidote.

Several urban management officers are frantically plundering street-side property worth more than 100 million yuan.

Figure 8: Some fake news detected by MCAN but missed by Spatial on the Weibo dataset.

In Figure 8, the two examples are detected by MCAN but missed by Spatial. The attached images in two examples look normal. However, the words in the tweet seem exaggerated and unbelievable. It is challenging for spatial-domain-only MCAN to detect, but with multimodal features, our MCAN model identifies them correctly.

These comparative cases prove that when a single-modal model, whether a text-based model or an image-based model, cannot correctly distinguish fake news, the proposed MCAN using multimodal features can give high confidence.

5 Conclusions

In this work, we propose a novel Multimodal Co-Attention Networks (MCAN) to tackle the challenge of fusing multimodal (textual and visual) features for fake news detection. We utilize three different sub-networks to extract features from text, spatial domain, and frequency domain, respectively. Then the three features are deeply fused by stacking co-attention layers, which is inspired by human behavior. When people read news with image, image and text are read once or multiple times, and continuously fused in brain. Experiments on two public benchmark datasets for fake news detection validate the effectiveness of MCAN, and the results show that MCAN outperforms the current state-of-the-art methods. In the future, we plan to extend the co-attention based fusion approach in MCAN to other fake news research, such as fake news diffusion.

Acknowledgments

This research was supported by National Research and Development Program of China (No.2017YFB1010004).

References

S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Zitnick, and D. Parikh. 2015. Vqa: Visual question answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433, Los Alamitos, CA, USA. IEEE Computer Society.

C. Boididou, S. Papadopoulos, D. Dang-Nguyen, G. Boato, and Y. Kompatsiaris. 2016. Verifying multimedia use at mediaeval 2016. In *MediaEval 2016 Workshop*.

Gary D Bond, Rebecka D Holman, Jamie-Ann L Eggert, Lassiter F Speller, Olivia N Garcia, Sasha C Mejia, Kohlby W Mcinnes, Eleny C Ceniceros, and Rebecca Rustige. 2017. 'lyin'ted','crooked hillary', and 'deceptive donald': Language of lies in the 2016 us presidential debates. *Applied Cognitive Psychology*, 31(6):668–677.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684.

Yimin Chen, Niall J Conroy, and Victoria L Rubin. 2015. Misleading online content: recognizing clickbait as" false news". In *Proceedings of the 2015 ACM on workshop on multimodal deception detection*, pages 15–19.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 171–175.

Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven CH Hoi, Xiaogang Wang, and Hongsheng Li. 2019. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6639–6648.

Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 795–816.

Zhiwei Jin, Juan Cao, Yongdong Zhang, Jianshe Zhou, and Qi Tian. 2016. Novel visual and statistical image features for microblogs news verification. *IEEE transactions on multimedia*, 19(3):598–608.

Dhruv Khattar, Jaipal Singh Goud, Manish Gupta, and Vasudeva Varma. 2019. Mvae: Multimodal variational autoencoder for fake news detection. In *The World Wide Web Conference*, pages 2915–2921.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yang Liu and Yi-Fang Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2019. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. In *The World Wide Web Conference*, pages 3049–3055.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Duy-Kien Nguyen and Takayuki Okatani. 2018. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6087–6096.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Peng Qi, Juan Cao, Tianyun Yang, Junbo Guo, and Jintao Li. 2019. Exploiting multi-domain visual information for fake news detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 518–527. IEEE.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 647–653.

Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. Eann: Event adversarial neural networks for multi-modal fake news detection. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, pages 849–857.

Youze Wang, Shengsheng Qian, Jun Hu, Quan Fang, and Changsheng Xu. 2020. Fake news detection via knowledge-driven multimodal graph convolutional networks. pages 540–547.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st international conference on data engineering*, pages 651–662. IEEE.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, Tieniu Tan, et al. 2017. A convolutional approach for misinformation identification.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. pages 1395–1405.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. 2020. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients.

Joint Multi-Decoder Framework with Hierarchical Pointer Network for Frame Semantic Parsing

Xudong Chen[1,2], **Ce Zheng**[1], **Baobao Chang**[1,3*]
[1]The MOE Key Laboratory of Computational Linguistics, Peking University, China
[2]School of Software and Microelectronics, Peking University, China
[3]Peng Cheng Laboratory, Shenzhen, China
`{xdc,zce1112zslx,chbb}@pku.edu.cn`

Abstract

Current researches on frame semantic parsing include three subtasks, namely frame identification, argument identification and role classification. Most of previous systems process these subtasks independently and ignore their interactions. We introduce a novel architecture based on multi-decoder strategy to handle these subtasks together. The multi-decoder strategy strengthens the interactions. Moreover, we design a hierarchical pointer network for argument identification which reduces the computational complexity. To our best knowledge, it's the first practice to introduce the pointer network into frame semantic parsing. The experiments show improvement over state of the art models on FrameNet dataset.

1 Introduction

Frame semantic parsing is a fundamental study in Natural Language Processing. It aims to parse sentences into frame-style semantic structures defined in FrameNet (Baker et al., 1998).

An example of frame-style semantic structures is shown in Figure 1. The word ***write.v*** is a target that evokes the frame called ***Text_creation***. The phrases underlined with green lines are called arguments. ***Author***, ***Text*** and ***Form*** are roles (also called frame elements) the arguments play in this frame. Hence the frame semantic parsing contains three subtasks, namely frame identification, argument identification and role classification. For a sentence with a given target, the frame identification is to disambiguate the frame for the target based on its contextual information, the argument identification is to identify the boundaries of all the arguments, and the role classification is to assign a semantic role to each argument we have found.

Early work (Hermann et al., 2014; FitzGerald et al., 2015; Hartmann et al., 2017) on frame semantic parsing adopts pipeline strategy. Their models apply independent models to handle different subtasks which ignore the interactions among subtasks. Moreover, the pipeline strategy usually causes error propagation problem. The accuracy of frame identification can become the bottleneck of the overall performance. Later work (Yang and Mitchell, 2017; Peng et al., 2018) processes all the subtasks jointly by optimizing them together during training. Their joint models show improvement over pipeline models, which demonstrates the benefit of joint training strategy. However, their systems don't have specific design to model the interactions among the subtasks.

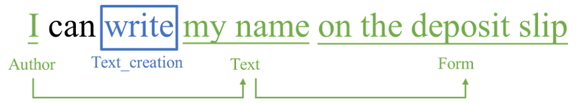

Figure 1: A sentence annotated the arguments and roles of frame ***Text_creation***. The arrow marks indicate the order of arguments identification and roles classification.

To strengthen the interactions of subtasks, we propose a joint framework based on three task-specific decoders. The interactions in our framework are mainly reflected in two aspects. On one hand, the representations of both the target and its frame that derived from frame identification decoder are applied to predict the arguments and roles. On the other hand, the argument identification decoder and the role classification decoder work in an alternate way, and thus they interact with each other during the entire process of decoding.

The interactions bring two benefits. First, the frame information predicted by frame identification decoder makes the predictions of arguments and roles more frame-specific. Second, the alternate decoding strategy makes the current argument's and role's prediction influenced by all previous

---

[*] Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2570–2578
August 1–6, 2021.

SENTENCE	Coming to Goodwill was the first step toward my becoming totally independent.
TARGET	**come.v**(Arriving), **to.prep**(Goal), **first.a**(Ordinal_numbers), **step.n**(Intentionally_act), **become.v**(Becoming), totally.adv(Degree)

Table 1: An example sentence with annotated targets and golden frames from FrameNet dataset. Bold words indicate that the target can evoke multiple frames.

decisions, which captures relations among different arguments and roles. For the sentence shown in Figure 1, the argument ***I*** and its role ***Author*** can contribute to the predictions of the argument ***my name*** and the role ***Text***. Therefore, the argument identification and the role classification can benefit from each other by considering arguments and roles already obtained.

For argument identification, previous models (Yang and Mitchell, 2017; Peng et al., 2018) enumerate all possible spans to identify the arguments, which brings high computational complexity. To reduce the computational complexity, we design a hierarchical pointer network in the argument identification decoder that predicts boundaries of arguments directly.

In addition, we design a target-aware attention mechanism. The target-aware attention mechanism aggregates all targets in the same sentence to model interactions among different targets, since frames evoked by different targets in the same sentence are usually closely related. Such interaction modeling could be helpful in frame identification. For example, the target ***to.prep*** in Table 1 can evoke the frame ***Goal*** or ***Locative_relation***, and other co-occurrence targets (such as ***come.v***) make ***to.prep*** more likely to evoke frame ***Goal*** instead of the other frame.

Overall, our main contributions can be summarized as follows:

- We design a novel multi-decoder framework to jointly process all the subtasks of frame semantic parsing. The multi-decoder strategy strengthens the interactions among frame identification, argument identification and role classification.

- We design a hierarchical pointer network that predicts the boundaries of arguments directly. The hierarchical pointer network predicts arguments within linear computational complexity. To our best knowledge, it's the first practice to introduce the pointer network into frame semantic parsing task.

- We design a target-aware attention mechanism to aggregate the semantic information of other targets in the same sentence.

We evaluate our model on FrameNet dataset, and the experiments show that our model outperforms state of the art models, which demonstrates the effectiveness of our model.

2 Related Work

Frame semantic parsing task is first proposed by Gildea and Jurafsky (2002) and has drawn attention since the SemEval 2007 shared task (Baker et al., 2007) was released. Early researches on frame semantic parsing focus on the feature-engineered methods(Johansson and Nugues, 2007; Das et al., 2010). Most of the early researches regard the frame semantic parsing as a pipeline of classification tasks and employ machine learning algorithms (such as Support Vector Machines etc.).

With the popularity of neural network and representation learning, neural network models are introduced to model frame semantic parsing problem. Hermann et al. (2014) uses distributed representations in frame identification and embedded both frames and the contextual representations of words into a shared low-dimension vector space. FitzGerald et al. (2015) uses a neural network to learn embeddings of both arguments and semantic roles, which adopts fine-grained similarity between roles to overcome the sparsity of some labeled data. Besides, a system based on pre-trained word distributed representations (Hartmann et al., 2017) is developed to improve the domain adaptation of their model. These models still work in pipeline way.

However, pipeline models usually ignore the interactions among subtasks and suffer from the problem of error propagation. Joint models are proposed to solve these problems. Yang and Mitchell (2017) proposes an ensemble strategy that that integrates two different models into an ensemble model. Peng et al. (2018) proposes a multi-task framework which jointly handles two different semantic pars-

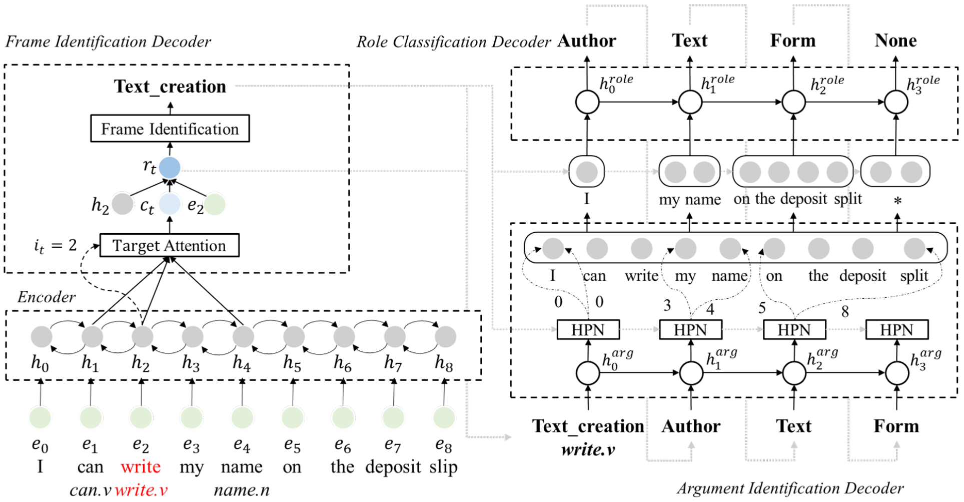

Figure 2: Our framework predicts frames, arguments and roles jointly based on three task-specific decoders. Frame identification module predicts frame ***Text_creation*** evoked by target ***write.v***. Then argument identification module with HPN(Hierarchical Pointer Network) and role classification module predict argument spans ***I***, ***my name*** and ***on the deposit split*** and their roles ***Author***, ***Text*** and ***Form*** alternately. The interactions in our framework are shown as gray dotted lines.

ing tasks from disjoint data. Both of their models process all the subtasks jointly by optimizing them together during training. Their experiments show improvement over previous pipeline models, which proves the benefit of joint training strategy. However their models don't have specific design for the interactions.

The common neural network architectures for frame semantic parsing can be divided into sequence labeling models and relational models. Yang and Mitchell (2017) proposes a sequence labeling model based on BIO tagging scheme. The model contains multiple LSTM layers and a Conditional Random Field (CRF) layer. Swayamdipta et al. (2017) adopts a segmentation RNN and a relational model to capture span-level dependencies between predicate and arguments. The relational model enumerates all possible spans to compute the matching scores. Both of these two type models above require $O(n^2)$ computational complexity. To reduce the high computational complexity, we design a hierarchical pointer network that achieves the identifying arguments within linear computational complexity.

3 Method

As is shown in Figure 2, our framework consists of four modules. (1) the encoder module (2) the frame identification decoder module. (3) the argument identification decoder module. (4) the role classification decoder module.

Specifically, given a target t and a sentence $S = w_0, \dots, w_{n-1}$, the encoder module calculates the contextual representations $h_0, \dots, h_{n-1}$, then all decoder modules handle three subtasks jointly. The frame identification decoder builds a target representation for t and identifies the frame $f \in F$ evoked by t. Suppose that there are k argument spans $a_0, \dots, a_{k-1}$ of f in S. For each argument $a_\tau = w_{i_\tau^s}, \dots, w_{i_\tau^e}$, the argument identification decoder identifies the boundaries i_τ^s and i_τ^e, and the role classification decoder assigns a semantic role $r_\tau \in R_f$ to a_τ. The F and R_f mentioned above are the sets of all frames and all semantic roles of f defined in FrameNet.

Three decoders interact with each other as follows:

- Frame identification decoder builds a target representation for t to identify the frame f. Both the target representation and the embedding of f will be taken as inputs to the other decoders for argument identification and role classification.

- Role classification decoder assigns a role r_τ to current argument span a_τ, and then the embedding of r_τ will be taken as an input to identify the boundary of next argument $a_{\tau+1}$. In other words, these two decoders work in an alternate way, so identifying $a_{\tau+1}$ and $r_{\tau+1}$ will consider all the historical information of $a_0, \dots, a_\tau$ and their roles $r_0, \dots, r_\tau$.

3.1 Encoder Module

Encoder module aims at converting the sentence $S = w_0, \ldots, w_{n-1}$ into a sequence of vectors $h_0, \ldots, h_{n-1}$, where h_i is the contextual representation of word w_i.

For each token, we concatenate its word embedding e_{w_i}, lemma embedding e_{l_i}, POS embedding e_{p_i} and a binary tag embedding e_{b_i}:

$$e_i = [e_{w_i}; e_{l_i}; e_{p_i}; e_{b_i}] \tag{1}$$

The binary tag embedding e_{b_i} is to distinguish t from other words in S. Let i_t be the position index of t in S, then we can calculate e_{b_i}:

$$e_{b_i} = \begin{cases} e_1, & i = i_t \\ e_0, & i \neq i_t \end{cases} \tag{2}$$

At last, e_i is fed to the encoder to get contextual representation h_i:

$$h_i = \text{Encoder}(e_i) \tag{3}$$

The word embedding and lemma embedding are initialized with Glove (Pennington et al., 2014) while POS embedding is randomly initialized. We use Bi-LSTM as the encoder in our experiment, which can be also replaced with any other encoder model such as Bert (Devlin et al., 2018). The dimension of e_i is d_e and the dimension of h_i is d_h.

3.2 Frame identification module

In frame identification module, we build a target representation r_t for t and identify the frame f based on r_t.

As there are likely to be multiple targets $t_0, \ldots, t_{m-1}$ in S evoking multiple frames $f_0, \ldots, f_{m-1}$ and we believe that other targets in S can contribute to identifying current frame f for target t, we design a target aware attention mechanism to aggregate contextual representations of all targets $T_S = \{t_0, \ldots, t_{m-1}\}$ in S (also contains the target t):

$$\alpha_i = \frac{\exp(h_i^\top W_1 h_{i_t})}{\sum_{j \in T_s} \exp(h_j^\top W_1 h_{i_t})} \tag{4}$$

$$c_t = \sum_{i \in T_s} \alpha_i h_i \tag{5}$$

For the target, we concatenate c_t, its contextual representation h_{i_t} and its embedding e_{i_t} to get the target representation r_t:

$$r_t = \text{Relu}(W_2 \cdot [e_{i_t}; h_{i_t}; c_t]) \tag{6}$$

With the target representation, we can generate the probability distribution of frames to identify f by argmax operation:

$$P(f|S,t) = \text{softmax}(W_3 \cdot r_t) \tag{7}$$

W_1, W_2, W_3 are three weight matrixes in $R^{d_h \times d_h}$, $R^{d_h \times (2d_h + d_e)}$ and $R^{|F| \times d_h}$ repectively, where $|F|$ is the size of F.

3.3 Argument Identification Module

Argument identification decoder module identifies the boundaries of argument spans $a_0, \ldots, a_{k-1}$ sequentially. For τ-th argument $a_\tau = w_{i_\tau^s}, \ldots, w_{i_\tau^e}$, the historical information of $a_0, \ldots, a_{\tau-1}$ and their roles $r_0, \ldots, r_{\tau-1}$ is supposed to be utilized. Hence We use LSTM_A to record the historical information, and similarly, another LSTM named LSTM_R is applied in role classification decoder. Argument identification decoder interacts with other decoders by taking their output as input, here is how LSTM_A works at τ-th argument:

$$x_\tau = \begin{cases} [r_t; e_f], & \tau = 0 \\ \left[h_{\tau-1}^{role}; e_{r_{\tau-1}}\right], & \tau > 0 \end{cases} \tag{8}$$

$$h_\tau^{arg} = \text{LSTM}_\text{A}\left(h_{\tau-1}^{arg}, x_\tau\right) \tag{9}$$

h_τ^{arg} and h_τ^{role} are hidden states at timestep τ of LSTM_A and LSTM_R. e_f represents the embedding of f, and $e_{r_{\tau-1}}$ represents the embedding of $r_{\tau-1}$.

As we want to identify the start and end positions of a_τ, namely i_τ^s and i_τ^e, we build two kinds of feature representations to extract boundary feature from h_τ^{arg} and e_f:

$$h_\tau^{STA} = \text{MLP}_\text{s}([h_\tau^{arg}; e_f]) \tag{10}$$

$$h_\tau^{END} = \text{MLP}_\text{e}([h_\tau^{arg}; e_f]) \tag{11}$$

The dimensions of both h_τ^{STA} and h_τ^{END} are the same as contextual representations $h_0, \ldots, h_{\tau-1}$, and the MLP in our experiment consists of two linear layers and a relu activation function in between.

3.3.1 Hierarchical pointer network

With two representations h_τ^{STA} and h_τ^{END}, we apply a hierarchical pointer network to identify i_τ^s and i_τ^e. The hierarchical pointer network contains two pointer networks as is shown in Figure 3. The hierarchical pointer network identifies the i_τ^s firstly and then identify the i_τ^e based on i_τ^s. If we identify them simultaneously, the i_τ^s and i_τ^e may sometimes

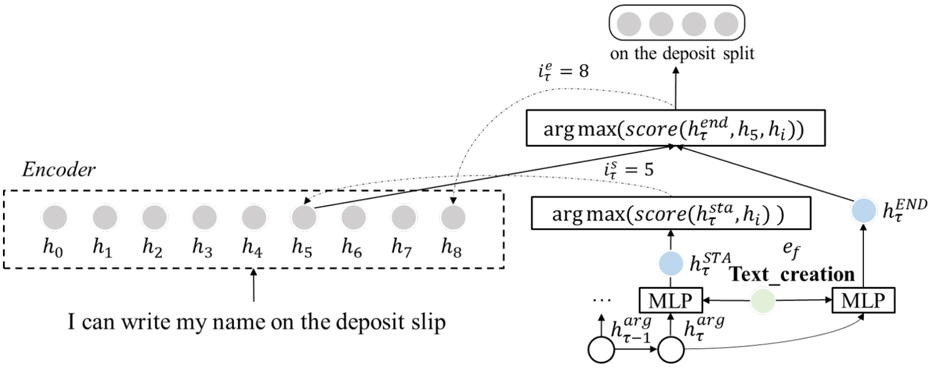

Figure 3: The prediction process of the argument ***on the deposit slip***.

be inconsistent. Besides, to avoid duplicate prediction, all spans in $a_0, \ldots, a_{\tau-1}$ are masked when identifying i^s_τ and i^e_τ. The prediction process is:

$$Score_{STA} = H^\top W_4 h^{STA}_\tau \tag{12}$$

$$P\left(i^s_\tau|S,t,f\right) = \text{softmax}(Score_{STA}) \tag{13}$$

$$i^s_\tau = \text{argmax}(P\left(i^s_\tau|S,t,f\right)) \tag{14}$$

$$Score_{END} = H^\top(W_5 h^{END}_\tau + W_6 h_{i^s_\tau}) \tag{15}$$

$$P\left(i^e_\tau|S,t,f\right) = \text{softmax}(Score_{END}) \tag{16}$$

$$i^e_\tau = \text{argmax}(P\left(i^e_\tau|S,t,f\right)) \tag{17}$$

H is $d_h \times n$ matrix $(h_0, \ldots, h_{n-1})$ that represents the encoder output of sentence S. W_4, W_5 and W_6 are $d_h \times d_h$ weight matrixes.

The hierarchical pointer network achieves arguments identification within linear computational complexity. For an n-tokens and k-arguments sentence, our model can identify the start and end positions of each argument with $O(2n)$ computational complexity, and $O(2n \cdot k)$ for all arguments.

3.4 Role Classification Module

The role classification module assigns semantic roles $r_0, \ldots, r_{k-1}$ to arguments $a_0, \ldots, a_{k-1}$. Assigning r_τ to a_τ also needs to consider the semantic information of $a_0, \ldots, a_{\tau-1}$ and their roles $r_0, \ldots, r_{\tau-1}$. Hence we use the same LSTM architecture named LSTM_R to record them. Both the contextual information of a_τ and the frame embedding e_f are used to predict r_τ:

$$y_\tau = W_7 \cdot [h_{i^e_\tau} + h_{i^s_\tau}; h_{i^e_\tau} - h_{i^s_\tau}; e_f] \tag{18}$$

$$h^{role}_\tau = \text{LSTM}_\text{R}(h^{role}_{\tau-1}, y_\tau) \tag{19}$$

$$P(r_\tau|S,t,f,a_\tau) = \text{MLP}([h^{role}_\tau; y_\tau]) \tag{20}$$

$h_{i_e} + h_{i_s}$ represents the boundary feature of a_τ and $h_{i_e} - h_{i_s}$ represents the inner feature of the span (Wang and Chang, 2016; Cross and Huang, 2016; Ouchi et al., 2018). With the probability distribution $P(r_\tau|S,t,f,a_\tau)$, we can predict the role r_τ.

Moreover, we add a special role ’None’ at the final decoding step and let r_k be ’None’. During the inference stage, the role classification decoder and argument identification decoder will automatically stop when predicting ’None’.

4 Loss Function

We utilize cross-entropy loss to maximize the probability of the oracle frame type, span boundaries (start-end pair) and role types:

$$\mathscr{L}_{frame} = \log(P(\hat{f}|S,t)) \tag{21}$$

$$\begin{aligned}\mathscr{L}_{role} = \sum_{\tau=0}^{k-1} &\log(P(\hat{r}_\tau|S,t,f,a_\tau))+ \\ &\log(P(r_{None}|S,t,f,a_k))\end{aligned} \tag{22}$$

$$\begin{aligned}\mathscr{L}_{span} = &\sum_{\tau=0}^{k-1} \log(P(\hat{i}^s_\tau|S,t,f))+ \\ &\sum_{\tau=0}^{k-1} \log(P(\hat{i}^e_\tau|S,t,f))\end{aligned} \tag{23}$$

We optimize the losses of the three subtasks jointly:

$$\mathscr{L} = \alpha\mathscr{L}_{frame} + \beta\mathscr{L}_{span} + \gamma\mathscr{L}_{role} \tag{24}$$

α, β and γ are hyper-parameters that adjust the direction of training optimization.

5 Experiment

Dataset. We train and evaluate our model on FrameNet 1.5 dataset proposed by (Das and Smith, 2011) following previous work (Yang and Mitchell, 2017; Swayamdipta et al., 2017; Peng et al., 2018).

We also follow the same train/development/test split. Meanwhile, previous work adds the partially-annotated exemplar sentences (each exemplar sentence contains only one target). As is reported in previous work (Das et al., 2014; Yang and Mitchell, 2017; Swayamdipta et al., 2017), the exemplar sentences data can help to improve their models' performance. We add it as pre-train data for our model.

Pre-process. Previous work removes the argument spans longer than 20, which is a constraint that helps to reduce the computational complexity from $O(n^2)$ to $O(n)$. Though our model doesn't need such constraint because of better computational complexity, we hold the same setting as previous work for comparison. We also report the result that training our model without length constraint.

Setup. We train our models by two steps following previous work (Das et al., 2014). At first step, we pre-train our model with partially-annotated exemplar sentences data. Then we train the model on the offcial train set. We evaluate our model on development test and save the best performance model for test.

We use Glove (Pennington et al., 2014) to initialize the word embeddings, and average the existing embeddings for out-of-vocabulary words. We randomly initialize embeddings for part-of-speech tags, and token type tags. All the embeddings are learnable during training.

Other detail hyper-parameters are shown on Table 2.

Model. We compare our model with following previous models:

SEMAFOR: A widely known system(Chen et al., 2010) that uses a variety of syntactic features.

Framat: An open-source semantic role labeling tool proposed by Björkelund et al. (2010).

Framat+context: An extension version of Framat that adds extra context features by Roth and Lapata (2015).

Hermann et al.(2014): A frame identification model uses feature representation based on word embedding and WSABIE algorithm (Weston et al., 2011).

FitzGerald et al.(2015): A pipeline model that improves frame identification performance based on Hermann et al. (2014).

Open-SESAME: A pipeline model that predicts frame by FitzGerald et al. (2015) and designs a softmax-margin segmental RNN to improve argument identification.

Hyper-parameters	Values
Batch size	32
MLP layers	2
Encoder lstm layers	2
Word/lemma embedding	200
Token type embedding	100
POS embedding	64
Pre-train/train epochs	50 / 100
Pre-train/train optimizer	Adam
Activation Function	Relu
Encoder/Decoder hidden size	256
MLP/LSTM dropout rate	0.4 / 0.2
Pre-train/train learning rate	1e-4/6e-5
Learning rate decay	0.6 (every 30 epochs)
α, β , γ	0.1 / 0.3 / 0.3

Table 2: Details of hyperparameters (non-bert version).

Model	**All**	**Ambiguous**
SEMAFOR	83.6	69.2
Open-SESAME	87.0	-
Hartmann et al.	87.6	73.8
Yang and Mitchell	88.2	75.7
Hermann et al.	88.4	73.1
Peng et al.(BASIC)	89.2	76.3
Our Model	**89.4**	**76.7**
Our Model+*Bert*	**90.5**	**79.1**

Table 3: Frame identification accuracy result.

Yang and Mitchell (SEQ)(2017): A sequence tagging model for frame semantic parsing.

Yang and Mitchell (REL) (2017): A relation model that enumerates all possible spans and classify them.

Peng et al. (Basic) (2018): A single-task version of joint SRL model (without extra data). It is the current state of the art model on the task of frame semantic parsing.

5.1 Experiment Metrics And Result

We evaluate our model on the metrics of frame identification accuracy and full structure extraction. Note that previous systems may also report ensemble models based on different ensemble methods to improve models' performance, and the model (Peng et al., 2018) based on multi-task framework brings extra train data. For comparability, we only report the performance of single models that trained on framenet data only. We note that none of above-mentioned previous models are based on Bert which is widely applied in many NLP tasks. To explore the impact of Bert on frame semantic parsing, we implement a Bert-based version of our model and also report results of Bert-based model.

Model	**P**	**R**	**F1**
SEMAFOR	69.2	65.1	67.1
Framat	71.1	63.7	67.2
Framat+context	71.1	64.8	67.8
Open-SESAME	71.0	67.8	69.4
FitzGerald et al.	74.8	65.5	69.9
Yang and Mitchell (SEQ)	69.6	70.9	70.2
Yang and Mitchell(REL)	77.1	68.7	72.7
Peng et al.(BASIC)	**79.2**	71.7	75.3
Our Model	75.1	**76.9**	**76.0**
Our Model+*Bert*	78.2	**82.4**	**80.2**

Table 4: Full structure extraction result on the FN test set.

Model	**P**	**R**	**F1**
Our Model	75.1	76.9	76.0
wo interaction (Role&Arg)	75.6	76.3	75.9
wo interaction (Frame)	76.1	75.1	75.6
wo interaction (Both)	75.9	74.6	75.3

Table 5: Full structure extraction result of our models considering the effect of the interactions among decoders.

Frame Identification. The metrics of frame identification accuracy includes Ambiguous and All as is shown in Table 3. The ambiguous metrics evaluates targets evoking more than one possible frame in FrameNet and All evaluates all the targets. According to Peng et al. (2018), some previous studies' ambiguous lexical unit sets are not the same as the one from the official frame directory, which makes their results uncomparable. Therefore, it's fairer to use ALL to evaluate the performance of frame identification. Our model outperforms all previous models (0.2 point over SOTA). Our Ambiguous set is the same as Peng et al. (2018)'s and our model outperforms theirs by 0.4 point.

Full Semantic Structure Extraction. Full Semantic Structure Extraction is the metrics that measures the overall performance of Frame Semantic parsing. It requires exact match of arguments' boundaries and jointly evaluates the performance of frame identification, argument identification and role classification. (Baker et al., 2007) shows details of the metrics. Table 4 is the result. The first group contains pipeline models and the second group includes joint models. Our model shows improvement over all previous models (0.7 point over SOTA). We notice that our model greatly outperform state of the art models on Recall. We analyze that it's because the decoders of our model fully interact and make the current decision by considering all previous steps' decisions. Such strategy is likely to predict more complete arguments and roles.

Model	**P**	**R**	**F1**
Our Model	75.1	76.9	76.0
wo pre-train data	72.6	73.1	72.9
wo pre-train data/TAM	72.2	72.9	72.5

Table 6: Full structure extraction result of our models considering the influences of pre-train data and targets-aware attention mechanism (TAM).

Model	**P**	**R**	**F1**
Our Model	75.1	76.9	76.0
wo length constraint	75.1	77.6	76.3

Table 7: Full structure extraction result of our models considering the influence of the length constraint of arguments.

5.2 Ablation study

We train our model in different settings and evaluate them on the metrics of full structure extraction to measure their overall performance. We consider the influences of decoders' interactions, pre-train data (partially-annotated exemplar sentences), targets-aware attention mechanism (TAM) and the length constraint of arguments.

As mentioned before, all the decoders of our model interact with each other and the interactions are reflected in two aspects. To prove the effectiveness of them, we remove the interactive parts of the decoders respectively. Table 5 shows the results of models with following setting:

Without interaction (Arg&Role) means the interaction between argument identification decoder and role classification decoder is deleted.

Without interaction (Frame) means the identified frame information in frame identification decoder is not accessible to the other decoders.

Without interaction (Both) represents that both of the interactions above are removed.

As is shown in Table 5, the performances of the models all drop in varying degrees without any kind of interactive parts. The effect of interaction (Frame) is more pronounced than interaction (Arg&Role) (73.6 to 73.9). The model's performance without both kinds of interactive parts will drop from 76.0 to 75.3. And we noticed that the

Error Type	Description	Proportion(%)	
		Our Model	Peng
Frame error	Frame misprediction	10.5	11.3
Role error	Matching span with incorrect role.	22.0	12.6
Span error	Matching role with incorrect span boundary.	14.1	11.4
Extra predicted arg.	Predicted argument that doesn't overlap any gold argument	20.0	18.6
Missing arg.	Gold argument that doesn't overlap any predicted argument	33.4	43.5

Table 8: Error analysis result on FrameNet development set.

recalling rate of our model is decreasing with the remove of interactions. It verifies the previous analysis that the interactions among decoders make our model predict in a global view and thus our model is likely to predict more complete arguments and roles.

Table 6 shows the performance differences in the influences of pre-train data and targets-aware attention mechanism. The performance of the model without the pre-train step drops by 3.1 points on F1, which demonstrates that adding pre-train data is beneficial to model's performance. The result is consistent with the conclusion of Yang and Mitchell (2017) that models will get a 3-4 points increase on F1 if adding partially-annotated exemplar sentences. Also, we consider the influence of the targets-aware mechanism. The targets-attention mechanism doesn't work at pre-train step because the partially-annotated exemplar sentences only contains one target per sentence. To eliminate the interference of the gap between the train data and the pre-train data, we hold the same setting of skipping the pre-train step. As shown in Table 6, the model without TAM drops from 72.9 to 72.5 on F1. It proves that the targets-aware attention mechanism contributes to overall performance of frame semantic parsing.

As mentioned before, our model has an advantage in terms of computational complexity. We remove the length constraint of spans which is adopted in previous work. Table 7 shows the result. We notice that our model has a slight increase without length constraint of arguments. The result shows that our model get benefit with complete data. Moreover, it proves that the hierarchical pointer network is good at capturing long distance dependency relation. We encourage future work to train and evaluate on complete data if their computational complexity allows.

5.3 Error Analysis

We follow the error analysis method of Peng et al. (2018) and compare our model with theirs. Table 8 shows the proportions of five error types. Though missing arguments is the major error for both of our model and Peng et al. (2018), our model shows a great decrease by 10.1%, which proves that our model prefers to predict more complete arguments and is more likely to overlap gold arguments. However, it correspondingly brings increases on extra predicted arguments, Role error and Span error. We analyze that it's because our model captures the relation between different roles and is able to make current decision by considering all previous steps' action information, it prefers to predict the role which is related to previous predicted roles. Such strategy is more likely to overlap gold arguments. However, it may also predict more arguments and roles than the ground truth.

6 Conclusion

We design a multi-decoder framework to process all the subtasks of frame semantic parsing jointly. The multi-decoder framework strengthens the interactions among these three tasks. Our model works in an alternate way, which predicts the argument and the role by considering all previous decisions. We apply a hierarchical pointer network which achieves the argument identification with linear computational complexity. Experiments show improvement over state of the art models.

Acknowledgments

The authors thank the reviewers for their thorough feedback and their suggestions. This work is supported by the National Science Foundation of China under Grant No.61936012, the National Key R&D Program of China under Grand No.2020AAA0106700, and the National Science Foundation of China under Grant No. 61876004.

References

Collin F Baker, Michael Ellsworth, and Katrin Erk. 2007. Semeval-2007 task 19: Frame semantic structure extraction. In *Proceedings of the Fourth*

International Workshop on Semantic Evaluations (SemEval-2007), pages 99–104.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36.

Desai Chen, Nathan Schneider, Dipanjan Das, and Noah A Smith. 2010. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 264–267.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. *arXiv preprint arXiv:1612.06475*.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. 2010. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 948–956.

Dipanjan Das and Noah A Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1435–1444.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Silvana Hartmann, Ilia Kuznetsov, M Teresa Martín-Valdivia, and Iryna Gurevych. 2017. Out-of-domain framenet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 471–482.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations.

Richard Johansson and Pierre Nugues. 2007. Lth: semantic structure extraction using nonprojective dependency trees. In *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*, pages 227–230.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. *arXiv preprint arXiv:1810.02245*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1492–1502.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256.

H-FND: Hierarchical False-Negative Denoising for Distant Supervision Relation Extraction

Jhih-Wei Chen[*1], **Tsu-Jui Fu**[*2], **Chen-Kang Lee**[3], and **Wei-Yun Ma**[3]

[1]University of California Los Angeles
[2]University of California Santa Barbara
[3]Institute of Information Science, Academia Sinica
`jwchen101@g.ucla.com, tsu-juifu@ucsb.edu`
`{cklee, ma}@iis.sinica.edu.tw`

Abstract

Although distant supervision automatically generates training data for relation extraction, it also introduces false-positive (FP) and false-negative (FN) training instances to the generated datasets. While both types of errors degrade the final model performance, previous work on distant supervision denoising focuses more on suppressing FP noise and less on resolving the FN problem. We here propose H-FND, a hierarchical false-negative denoising framework for robust distant supervision relation extraction, as an FN denoising solution. H-FND uses a hierarchical policy which first determines whether non-relation (NA) instances should be kept, discarded, or revised during the training process. For those learning instances which are to be revised, the policy further reassigns them appropriate relations, making them better training inputs. Experiments on SemEval-2010 and TACRED were conducted with controlled FN ratios that randomly turn the relations of training and validation instances into negatives to generate FN instances. In this setting, H-FND can revise FN instances correctly and maintains high F1 scores even when 50% of the instances have been turned into negatives. Experiments on NYT10 is further conducted to show that H-FND is applicable in a realistic setting. [1]

1 Introduction

Relation extraction (Zelenko et al., 2003; Mooney and Bunescu, 2006; Zhou et al., 2005) is a core task in information extraction. Its goal is to determine the relation between two entities in a given sentence. For instance, given the sentence "Jobs was born in San Francisco", with head and tail entities "Jobs" and "San Francisco", the relation to be extracted is "Place of Birth". Relation extraction can be applied for many applications, such as question answering and knowledge graph completion. A major difficulty with supervising relation extraction models is the cost of collecting training data, against which distant supervision (DS) (Hoffmann et al., 2011; Surdeanu et al., 2012) is proposed. DS obtains the relational facts from a knowledge base and aligns these facts to all sentences in the corpus to generate learning instances. In specific, if a relation triple $r(h, t)$ exists in a knowledge base, then for a sentence s which mentions both the head entity h and the tail entity t, it is tagged with relation r to form a learning instance (r, h, t, s). Since an effective classifier is expected not only to extract relation triples from a given text but also have to identify those unrelated entity pairs, the negative samples from texts are also needed for the training. In distant supervision, the negative samples are generated by randomly selecting two entities in the given text to form an entity pair that does not appear in any relation triples in the knowledge base.

Knowledge base	Relation	
Steve Jobs, **San Francisco**	PoB	

Corpus	Relation	Type
Jobs was born in **San Francisco**	PoB (✓)	TP
Jobs moved back to **San Francisco**	PoB (✗)	FP
Manuela was born in **New York**	NA (✗)	FN

Table 1: Distant supervision and different types of incorrectly labeled relations. The head and tail entities are shown in boldface, and "PoB" stands for the relation "Place of Birth".

Datasets generated using distant supervision contain considerable noise (Roth et al., 2013). More specifically, the noise generated can be classified into false positives (FP) and false negatives (FN). Table 1 shows an example. The FP "Jobs moved back to San Francisco" should not reflect the relation 'Place of Birth'. Also, an FN: as there is no relation between "Manuela" and "New York" in the knowledge base, "Manuela was born in New

---

[*] Equal contribution.

[1] The code can be found at `https://github.com/ckiplab/hfnd`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2579–2593
August 1–6, 2021. ©2021 Association for Computational Linguistics

York" is wrongly labeled as a non-relation (NA) under the closed world assumption. Both FP and FN degrade model performance if they are treated as correct labels at training time. FPs harm prediction precision, while excessive FNs lead to low recall rates.

In addition to denoising methods for learning robustly with noisy data (Han et al., 2018; Northcutt et al., 2019), many works focus on alleviating the FP problem in DS datasets, including those on pattern-based extraction (Alfonseca et al., 2012; Jia et al., 2019), multiple-instance learning (Surdeanu et al., 2012; Lin et al., 2016; Zeng et al., 2018), and sentence-level denoising with adversarial training or reinforcement learning (Qin et al., 2018a,b; Feng et al., 2018). However, few investigate the FN problem for distant supervision (Xu et al., 2013; Roller et al., 2015). To the best of our knowledge, there is no previous study on this problem for deep neural networks.

In this paper, we investigate the impact of FNs on neural-based models and propose H-FND, a hierarchical false-negative denoising framework for robust distant supervision. Specifically, this framework integrates a deep reinforcement learning agent which keeps, discards, or revises probable FN instances with a relation classifier to generate revised relations. In addition, to constrain the study to the FN problem and to construct ground-truth relations to further analyze model behavior, we conduct our research on the following two human-annotated datasets: SemEval-2010 (Hendrickx et al., 2010) and TACRED (Zhang et al., 2017), with controlled FN ratios that randomly flip relations of training/validation instances into negatives to generate FN instances. Then, we further conduct our experiment on a distantly supervised dataset NYT10 (Riedel et al., 2010) and fix its positive set, to demonstrate that our framework is applicable for resolving FN problem in a realistic setting. In summary, our contributions are threefold:

- We propose a denoising framework focused on false negatives in relation extraction.

- We present a special transfer learning scheme for pretraining denoising agent as training data is not available for this pretraining task.

- We show that our method revises correctly and maintains high F1 scores even under a high percentage of false negatives, and is applicable in a realistic setting.

2 Related Work

Mintz et al. (2009) propose distant supervision (DS) to automatically generate labeled data for relation classification, a new paradigm that synthesizes positive training data by aligning a knowledge base to an unlabeled corpus, and produces negatives with a closed-world assumption. Although this method requires no human effort for sentence labeling, it introduces FPs and FNs into the generated data and degrades the performance of relation extraction models.

Many previous works have attempted to solve the FP problem. Among these works, denoising methods that utilize reinforcement learning (RL) are the most relevant to ours. Feng et al. (2018) propose a sentence-level denoising mechanism that trains a positive instance selector using RL, and set the RL reward to the prediction probability of the relation classifier. Qin et al. (2018b) also utilizes RL, but in a different way. It learns a denoising agent to redistribute FPs to NA via prediction accuracy of the classifier as the RL reward.

To solve the FN problem, one method is to align the KB to the corpus after performing KB completion using inference (Roller et al., 2015). Although this does reduce the number of FNs in DS datasets, it helps little when the FN relations cannot be inferred from the KB, e.g., the entities mentioned in the FN are not in the KB. IRMIE (Xu et al., 2013), another method, constructs a negative set in a more conservative sense, in which the head or tail entities have already participated in other relation triples in the KB. Other sentences outside the positive and negative sets are left unlabeled (labeled as RAW in original paper) to prevent FNs. After training on the positive and negative sets, positive relation triples are retrieved from the unlabeled set to expand the KB, after which the original DS is performed to improve the quality of relation extraction. The final performance of this method depends heavily on the heuristic for constructing the negative set, which may not be applicable for all possible relation types.

To address the FN problem in DS datasets more generally, we propose a hierarchical denoising method to mitigate the negative effect of FNs, ensuring a more robust relation extraction model when the presence of FN instances is unavoidable.

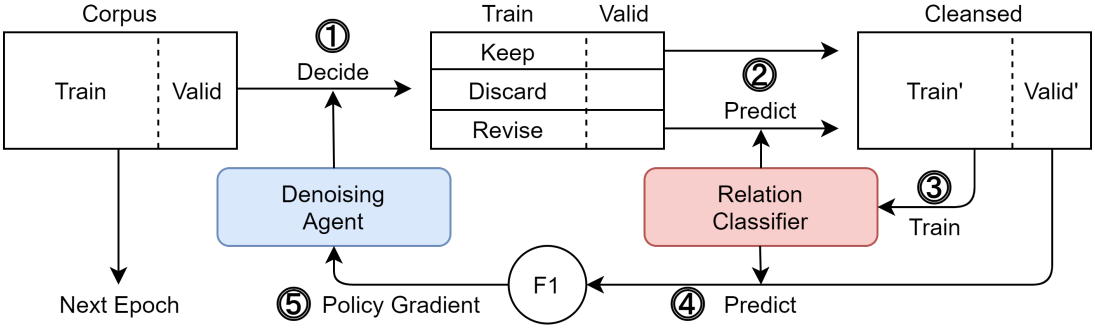

Figure 1: H-FND framework. The process in this diagram is executed per epoch.

3 H-FND Framework

We propose H-FND, a hierarchical false-negative denoising framework that determines whether to keep, discard, or revise negative instances. As illustrated in Fig. 1, H-FND is composed of the denoising agent and relation classifier modules. The denoising agent makes a ternary decision on the action to take on each negative instance, and after discarding, the relation classifier predicts a new relation for each to-be-revised instance to produce a cleaned dataset.

3.1 Convolutional Neural Network

Convolutional neural networks (CNN) are commonly adopted for sentence-level feature extraction (Kim, 2014) in language understanding tasks, such as relation extraction (Zeng et al., 2014; Nguyen and Grishman, 2015). PCNNs (Zeng et al., 2015), a variation of CNN that applies piecewise max pooling, are also widely used for extracting sentence features (Lin et al., 2016; Qin et al., 2018a). We included both as the base model in our experiments to show that our framework is base model agnostic. In our implementation, the extracted features of a learning instance s are fed into a fully connected softmax classifier to compute the final logits:

$$O(r) = \text{softmax}(\text{FC}(\text{CNN}(s))).$$

For detailed mathematical descriptions of CNN and PCNN, please refer to the Appendix.

3.2 Hierarchical Denoising Policy

The proposed hierarchical denoising policy is a framework using policy-based reinforcement learning (RL). Previous work utilizing RL to suppress noise from FPs (Feng et al., 2018; Qin et al., 2018b) can be categorized in two types of strategies: the first decides whether to remove the input instance, and the second decides whether to revise the input instance to be negative. Both policies make a binary decision on each input instance, and successfully reduce FP instances in DS datasets.

While applicable on the FP problem, it is risky to directly apply these strategies on the FN problem. First, discarding a negative instance even when it is most likely positive can result in a loss of useful learning instances. Second, changing a negative instance to positive is not enough for the training process: we must also know which type of positive relation to revise to.

Therefore, we propose a hierarchical denoising policy to perform the FN denoising in two steps. The first step, a soft policy that combines the two above-mentioned denoising methods, is an agent that takes an action from the action set $\{$*Keep, Discard, Revise*$\}$ for a negative instance s:

- *Keep:* maintain s as a negative instance for training/validation;
- *Discard:* remove s to prevent it from misleading the model;
- *Revise:* predict a new relation type for s and treat it as a positive for the following training/validation.

The policy $\pi(a|s)$ of this ternary decision is calculated based on the sentence feature extracted from s with the base model CNN encoder:

$$\pi(a|s) = \text{softmax}(\text{FC}_1(\text{CNN}_1(s)));$$

each action a has the possibility of $\pi(a|s)$ of being taken by the denoising agent.

Then, if the negative instance s is to be revised, the hierarchical policy goes on to the second step and gives the revised relation by selecting the most

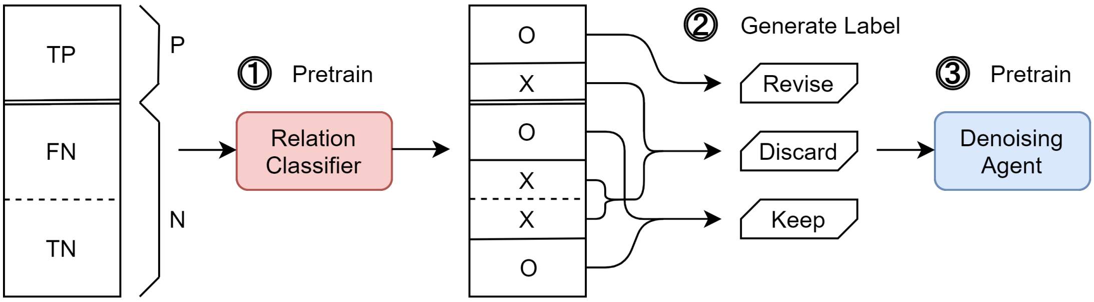

Figure 2: A special transfer learning scheme for H-FND pretraining. Symbols "P" and "N" represent positive and negative instances for relation classifier pretraining. Symbols "O" and "X" indicate two sets of training instances which are correctly predicted and wrongly predicted by pretrained relation classifier correspondingly.

likely relation (excluding NA) predicted by the relation classifier:

$$r' = \arg\max_{r \in R \setminus \{NA\}} \text{FC}_2(\text{CNN}_2(s)).$$

3.3 Pretraining

Supervised pretraining (Qin et al., 2018b), commonly used to accelerate RL agent training, is easily performed for the relation classifier on the original DS dataset (Han et al., 2018). For the denoising agent, however, there is no available training data. Therefore, we propose a special transfer learning scheme that utilizes the learnt knowledge in the relation classifier (source domain) to help generate action labels for pretraining denoising agent (target domain) (See Fig. 2).

First, we select the positives for which the pretrained relation classifier correctly predicts the relation, and tag these with *Revise*. This prepares the denoising agent to identify positive instances in the negative set in future training, and then pass these kinds of instances to the relation classifier to predict the correct positive relations for them. Similarly, we tag with *Keep* those negatives correctly predicted by the relation classifier. Lastly, for instances in which the relation classifier wrongly predicts their relation, we tag them with *Discard*, encouraging the denoising agent to discard such instances to avoid incorrect revisions.

In summary, our pretraining strategy is thus:

1. **Relation classifier pretraining**: pretrain the relation classifier (RC) directly on the original training set with the categorical loss function:

   $$ls_{\text{RC}} = \text{cross-entropy}(O, G),$$

   where G represents the distantly supervised relation in the training set. Then, fix the parameters of the relation classifier for the next step.

2. **Label generation**: generate labels H with the predictions of the relation classifier.

3. **Denoising agent pretraining**: Supervise the denoising agent (DA) with categorical loss:

   $$ls_{\text{DA}} = \text{cross-entropy}(\pi, H).$$

3.4 Co-Training

To combine the training of the relation classifier and the denoising agent, we propose the following co-training framework during each epoch (see Fig. 1):

1. **Denoising agent decision**: At the beginning of each epoch, the denoising agent first executes the denoising policy on the dataset. For both training and validation sets, the policy keeps, discards, or revises NA instances.

2. **Relation classifier revision**: For instances to be revised, the relation classifier generates revision relations for them. Denoising yields the cleaned training and validation sets.

3. **Relation classifier training**: Given the cleaned training set, we train the relation classifier in a supervised fashion based on categorical loss:

   $$ls_{\text{RC}} = \text{cross-entropy}(O, G'),$$

   where G' represents the modified training set, which contains all the positives and the kept or revised negatives. Note that discarded negatives are not included in G'.

4. **Reward determination**: We evaluate the trained relation classifier on the cleaned validation set to obtain the F1 score, which we use as reward R for denoising. As the validation set is cleaned by the denoising policy, R reflects the efficacy of the policy.

Datasets	#training	#validation	#testing
SemEval	6,599	1,154	2,717
TACRED	63,782	20,088	15,509
NYT10	477,454	120,318	194,328

Table 2: Number of instances in each dataset

5. **Denoising policy update**: To maximize the reward R, we adopt policy gradient (Sutton et al., 2000) to optimize the denoising agent by maximizing the objective function $J(\theta)$:

$$J(\theta) \approx \sum \log p(a|\theta)(R - b),$$

where θ is the parameter of the denoising policy, $p(a|\theta)$ represents the softmax probability of the sampled determination or revision step, and b is the baseline which mitigates the high variance of the REINFORCE algorithm (Williams, 1992). We set b to the average reward of the previous five epochs.

For each epoch, we obtain the revised set from the original training/validation set via the denoising policy, and H-FND finds the best denoising policy adaptively between supervised training and reward maximization.

4 Experiment

4.1 Datasets

In order to quantify our model's performance on denoising false negatives, we evaluated the proposed H-FND under two settings, human-annotated datasets with synthetic noise and dataset generated using distant supervision. Table 2 shows the statistics of each dataset used in the experiments.

1. **Human-Annotated Datasets**:

   SemEval-2010[2] contains nine relations with an additional NA as a non-relation, and the number of instances for each relation is roughly equal. TACRED[3] is about 10 times larger than SemEval, and it has 42 relations including NA, and the number of negative instances accounts for 80% of the entire corpus. For SemEval, we used 10% of the training set for validation, and for TACRED we simply used the dev set as the validation set (see Table 2).

   We filtered out the training and validation instances which had relation triples that appeared in the testing set to eliminate any overlap between relation triples in the training, validation, and testing sets, to simulate the held-out evaluation settings in distant supervision (Mintz et al., 2009).

   To simulate FN conditions, we randomly filtered a ratio (10%–50%) of training/validation positives into negatives. Note that the filtering process was only for training/validation: the testing sets were well-labeled under all FN ratios. Also note that the models were not aware in advance which sentences were TN and which were FN.

2. **Distantly Supervised Dataset**: The NYT10 dataset[4] uses Freebase as knowledge base for distant supervision. The relations are extracted from a December 2009 snapshot of Freebase. Four categories of Freebase relations are used: "people", "business", "person", and "location". These types of relations are chosen because they appear frequently in the newswire corpus. All pairs of Freebase entities that are at least once mentioned in the same sentence are chosen as candidate relation instances. For consistency with previous research (Lin et al., 2016; Feng et al., 2018; Qin et al., 2018b), we excluded five relations:
   '/business/company/industry',
   '/business/company_shareholder/major_shareholder_of',
   '/people/ethnicity/includes_groups',
   '/people/ethnicity/people',
   '/sports/sports_team_location/teams'

   This results in a total of 53 relations (including none-relation, 'NA').

   The corpus is chosen from a external source articles published by The New York Times between January 1, 1987 and June 19, 2007. The Freebase relations were divided into two parts, one for training and one for testing. The former is aligned to the years 2005-2006 of the NYT corpus, the latter to the year 2007.

4.2 Baselines and Experiment Settings

A simple H-FND baseline was the original CNN and PCNN relation classifier. To demonstrate the

---

[2] http://www.kozareva.com/downloads.html

[3] https://catalog.ldc.upenn.edu/LDC2018T24

[4] http://iesl.cs.umass.edu/riedel/ecml/

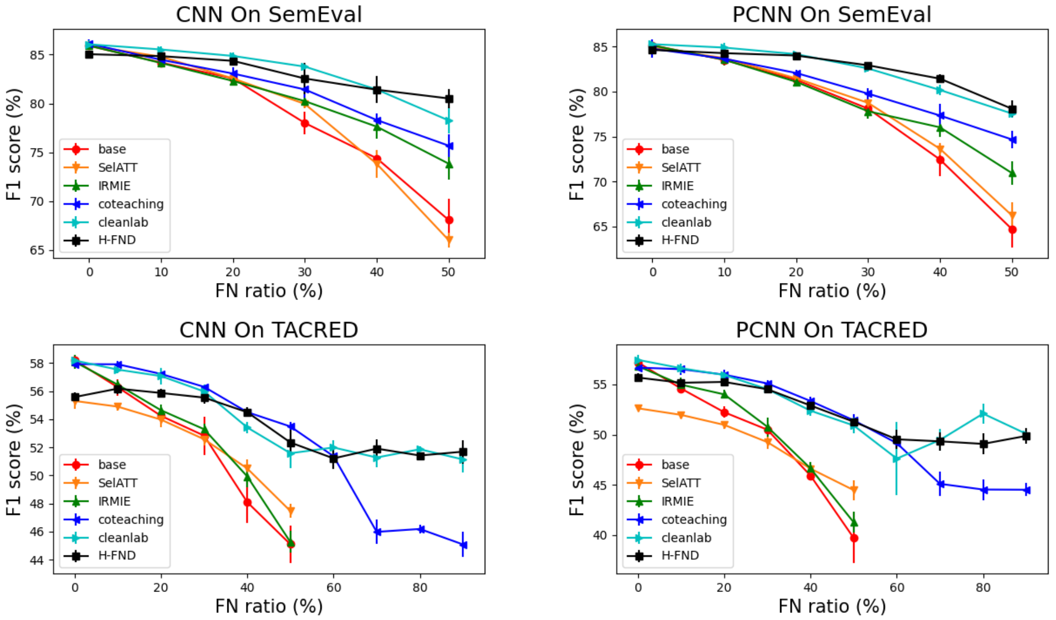

Figure 3: CNN and PCNN results on SemEval and TACRED, where the errorbars represent the standard deviations. The noise rate denotes the percent of positive relation triples flipped to create false negatives. The denoising method cleanlab and our method H-FND perform the best, but cleanlab requires a given noise rate of data, while H-FND does not requires such information.

impact of FNs, we also included SelATT (Lin et al., 2016), an FP noise resistant model.

We further compared our H-FND framework with the following strong baselines: the FN denoising method IRMIE (Xu et al., 2013) and two other general-purpose denoising methods: co-teaching (Han et al., 2018) and cleanlab (Northcutt et al., 2019). Co-teaching is a general training method for deep neural networks to combat extremely noisy labels. It simultaneously maintains two networks (each with the same structure), each of which samples its small-loss instances with a given overall noise rate as clean batches to its peer networks for further training. Cleanlab is a state-of-the-art robust learning method which directly estimates the joint distribution of noisy observed labels and latent uncorrupted labels with a consistent estimator, filters out noisy instances based on this joint distribution, and trains the relation classifier on the cleaned dataset with co-teaching mentioned above. We use these denoising methods to train the base CNN and PCNN models on our simulated FN datasets. [5]

As the focus of this paper is on the FN problem, all the positives of the simulated FN datasets are kept error-free, the H-FND framework assumes that no positives need be changed. Hence, for a fair comparison, we kept the positive sets of the FN datasets unchanged for the two general-purpose denoising methods, preventing them from discarding error-free positives. Also, we fix the positive set of NYT10 to evaluate the applicability of H-FND of resolving FN problem in a realistic setting.

In the experiments on SemEval and Tacred, every data point is the average of five independent runs. In the experiment of NYT10, data points are the average of three best results out of five independent runs for H-FND and all baselines. See Appendix for more detailed information on experiment and model implementation.

4.3 Quantitative Results

The quantitative SemEval results are shown in the upper part of Fig. 3, including both CNN and PCNN. Under the 50% FN ratio, for both the base CNN and PCNN models, with or without SelATT, the F1 scores are heavily influenced by FN sentences: the performance drops by nearly 20%. ERMIE and co-teaching enhance the performance by more than 5% and 8% correspondingly. Except for cleanlab, H-FND denoising remains competitive to the baselines for FN ratios from 10% to 30%, and significantly wins after 30%. Among all

---

[5] The IRMIE KB was reconstructed from the positives of the simulated FN dataset.

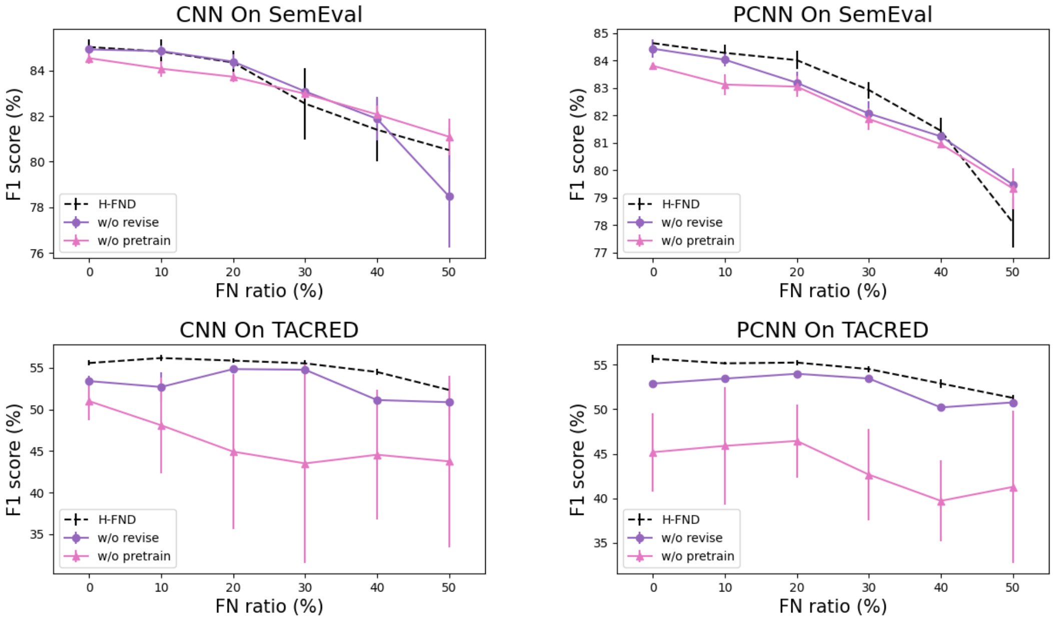

Figure 4: CNN and PCNN ablation analysis on SemEval and TACRED, where the errorbars represent the standard deviations.

baselines, cleanlab's performance is the strongest and is competitive with our approach, but as cleanlab relies on a co-teaching model to train the relation classifier, a given noise rate is required. In our experiments, these are directly provided to the model. However, in practice, the noise rate (the FN ratios in our experiment) is unknown and must be estimated correctly, entailing extra effort. In contrast, H-FND has no such requirement.

The quantitative results on TACRED are shown in the lower part of Fig. 3. CNN, PCNN, and the two models with SelATT are all vulnerable to FN instances. As IRMIE fails to exclude enough FNs from the negative set on TACRED,[6] its performance is also strongly influenced by FN instances. Although the F1 scores of H-FND are 2% behind co-teaching and cleanlab for FN ratios from 0% to 20%, it successfully maintains its performance when the FN ratio exceeds 30% and becomes competitive with these two baselines. This is similar to the experimental results on SemEval for FN ratios less than 30%. Together with the fact that TACRED has many more positives than SemEval, we increased the FN ratio to 90%. The result of this extended experiment shows that when the FN ratio exceeds 60%, the F1 scores for co-teaching drop significantly, whereas H-FND maintains a relatively high F1 score. Here, again, although cleanlab performs similar to ours with the predefined FN ratios,[7] the proposed approach needs no such information, which better fits real-world circumstances of distant-supervised relation classification.

4.4 Ablation Study

Fig. 4 shows the result of the ablation study to justify the effectiveness of the *Revise* action and pretraining strategy. On Semeval, pretraining boosts the F1 score for the PCNN architecture for FN ratios from 10% to 40%, but yields no significant difference for the other ratios. On TACRED, however, the *Revise* action and the pretraining strategy clearly yield improved results. This improvement is substantial in particular for pretraining. As TACRED has more positive relation types and a much larger negative set, the FN denoising problem is more severe than on SemEval; thus the pretraining strategy is crucial to provide a better initial point for the denoising agent and to ensure more stable performance.

4.5 Detailed Analysis

We first analyzed the distribution of the denoising policy for TN and FN instances in the training set. Figure 5 shows the percentage of kept, discarded,

---

[6] The size of the RAW set is less than 10% of the original negative set under all FN ratios.

[7] We have measured the performance of cleanlab when it was provided with a wrong FN ratio - 40% FN ratio. Under the actual FN ratio of 80% , its F1 scores dropped by 0.5% for CNN and 1.8% for PCNN.

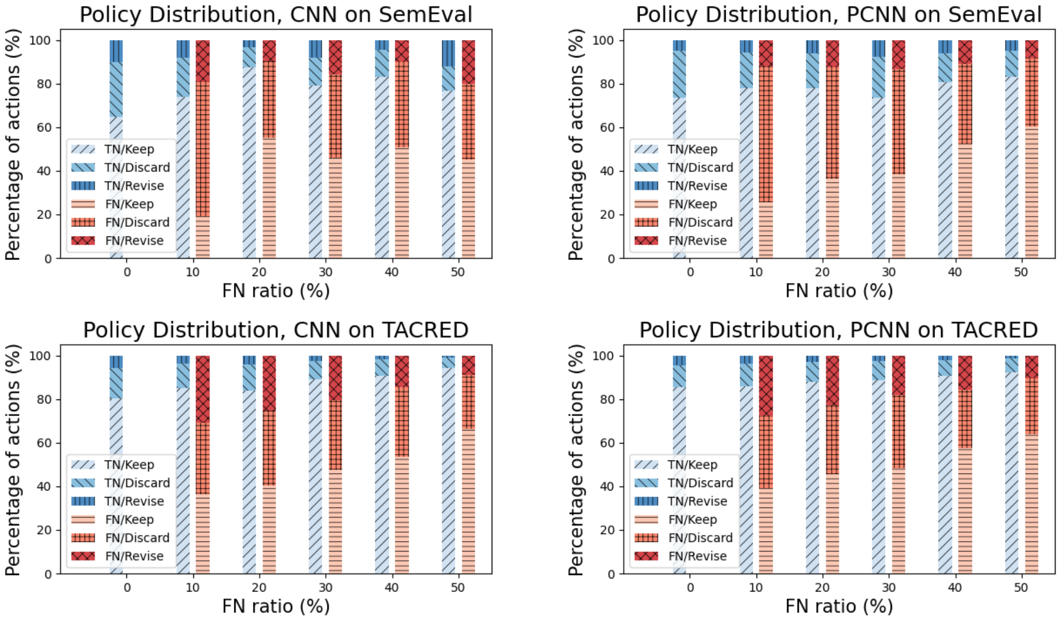

Figure 5: Denoising policy distribution on true negatives and false negatives.

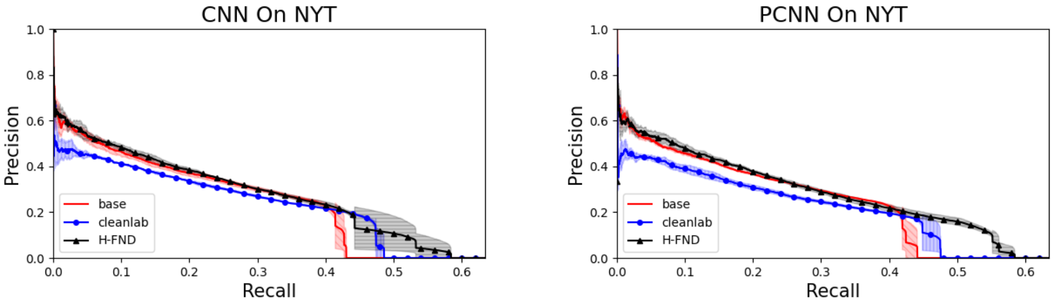

Figure 6: Precision-recall curve on the NYT dataset. The shaded areas indicate one standard deviation. The precision rate of each algorithm run drops to zero at certain recall rate, hence the steep drops in the curves.

or revised training instances. The left histogram under each filter ratio is for TN; the right is for FN.

On SemEval, we observe that for TN instances, H-FND mainly keeps them as NA and revises only a small portion to the wrong relation, even under the 50% filter ratio. For FN instances, H-FND prefers to discard or revise them. This difference shows that H-FND distinguishes FN instances from TN instances, and does not take arbitrary actions on them.

On TACRED, the policy distribution also shares the same tendency, but the portion of kept instances is generally larger. This is due to a higher ratio of negative instances in TACRED. As more negative instances result in more *Keep* labels in the generated pretraining data, after pretraining, the probability of the model taking the *Keep* action is generally higher. It also explains that the portion of kept instances grows when the filter ratio is raised. Note that this prevents H-FND from revising too many instances at the beginning of co-training, making co-training more stable.

Table 3 show the correctness of revisions on FN instances which are determined to be revised. The accuracy is around 90% for both CNN and PCNN architectures and for both SemEval and TACRED. This shows that H-FND accurately corrects FN instances once they are identified and determined to be revised in the first stage.

4.6 Results on Realistic Dataset

Lastly, we evaluated H-FND on real DS dataset NYT10. For baselines, apart from the base model, we included cleanlab, as it is the best performing baseline in the controlled FN experiments. We conducted human evaluation on 200 negative instances randomly sampled from the training set

SemEval	10%	20%	30%	40%	50%
CNN	88.81 ± 3.55	88.07 ± 7.80	88.72 ± 1.04	86.25 ± 1.53	85.94 ± 1.46
PCNN	89.54 ± 1.98	88.04 ± 3.12	86.83 ± 2.11	90.31 ± 0.54	84.17 ± 3.56
TACRED	**10%**	**20%**	**30%**	**40%**	**50%**
CNN	91.43 ± 0.72	90.62 ± 0.99	90.89 ± 0.63	91.65 ± 0.99	93.39 ± 1.79
PCNN	90.99 ± 0.82	89.64 ± 0.39	87.15 ± 0.49	86.75 ± 0.60	86.15 ± 1.16

Table 3: Revision accuracy (%)

and came to an estimate of 14% noise. [8]

We followed Zeng et al. (2015) and plotted the precision-recall curve to demonstrate the result on NYT10 (see Fig. 6). At recall rate lower than 40% cleanlab performs slightly worse than the base model, while H-FND remains competitive in terms of precision. This could be a result of inaccuracies in the estimation of FN rate in the dataset. Since H-FND does not require a given FN rate, it is not encumbered by such estimation error. At higher recall rates ($> 50\%$), H-FND retains significantly higher precision. This result shows that H-FND is applicable for real DS datasets, especially when the recall rate matters.

5 Conclusion and Future Work

In this work, to increase the robustness of DS, we present H-FND, a hierarchical false-negative denoising framework, which keeps, discards, or revises non-relation (NA) inputs during training and validation phases to suppress noise from FN instances. We also present a special transfer learning scheme for pretraining the denoising agent.

To investigate the effects of FN instances addressed by our approach, we generate FN instances from SemEval-2010 and TACRED under controlled ratios. The results show that H-FND revises FN instances and facilitates robust relation extraction. Further experiment on NYT10 demonstrates that our framework is also applicable to realistic DS setting.

In realistic DS setting, both FP and FN instances may emerge simultaneously. Both of which should be addressed. We leave this as future work. Also, we plan to attempt other advanced relation classification approach like R-BERT (Wu and He, 2019) to replace CNN or PCNN in our architecture.

6 Acknowledgements

We are grateful for the insightful comments from anonymous reviewers. This work is supported by the Ministry of Science and Technology of Taiwan under grant numbers MOST110-2634-F-001-011.

References

Paszke Adam, Gross Sam, Chintala Soumith, Chanan Gregory, Yang Edward, DeVito Zachary, Lin Zeming, Desmaison Alban, Antiga Luca, and Lerer Adam. 2017. Automatic differentiation in PyTorch. In *Neural Information Processing Systems workshop: The future of gradient-based machine learning software & techniques*.

Enrique Alfonseca, Katja Filippova, Jean-Yves Delort, and Guillermo Garrido. 2012. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 54–59, Jeju Island, Korea. Association for Computational Linguistics.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

---

[8] This also demonstrates our synthesized datasets are a good approximation to realistic DS setting: for the NYT10, around 72% of the instances are negatives. This gives us around (0.72 * 0.14) = 10.1% of FN in all triples. For our synthetic dataset, the noise rate is the percent of positive relation triples flipped to create false negatives. Positive triples make up only around 20% of the whole TACRED, and in our experiments, the noise rate indicates that we have flipped 50% of the triples. This gave us a total of around (0.2 * 0.5) = 10% of FN in all triples.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378.

Wei Jia, Dai Dai, Xinyan Xiao, and Hua Wu. 2019. ARNOR: Attention regularization based noise reduction for distant supervision relation classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1399–1408, Florence, Italy. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the NAACL Workshop on Vector Space Modeling for Natural Language Processing*.

Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. 2019. Confident learning: Estimating uncertainty in dataset labels. *arXiv preprint arXiv:1911.00068*.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018a. DSGAN: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Melbourne, Australia. Association for Computational Linguistics.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018b. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147, Melbourne, Australia. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163.

Roland Roller, Eneko Agirre, Aitor Soroa, and Mark Stevenson. 2015. Improving distant supervision using inference learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 273–278, Beijing, China. Association for Computational Linguistics.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 73–78.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 665–670, Sofia, Bulgaria. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5658–5665.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan. Association for Computational Linguistics.

A Appendices

A.1 Convolutional Neural Network

We use a convolutional neural network (CNN) (Nguyen and Grishman, 2015) as our base model for both the denoising agent and the relation classifier. This architecture consists of four main layers (the first three layers compose the CNN encoder):

1. **Embedding**: The embedding layer transforms a word into a vector representation, which is a concatenation of a word embedding V_w and a pair of positional embedding vectors V_{p_1}, V_{p_2} (Lin et al., 2016). Word embedding V_w is a vector that represents the semantics of a word, and positional embedding pair V_{p_1}, V_{p_2} is two vectors representing the relative distance from the current word to two entities in the sentence.

   The final embedding vector V of dimension d_e for each word is the concatenation of V_w, V_{p_1}, and V_{p_2}:

   $$V = [\,V_w|\,V_{p_1}|\,V_{p_2}]\,.$$

2. **Convolution**: The convolutional layer transforms the embedding vectors of words into local features by applying sliding filters over them. Each filter consists of a weight matrix $A_i \in \mathbb{R}^{f\times d_e}$ and a bias term $b_i \in \mathbb{R}$, to extract specific patterns in the embedding vectors. With h filters of length f, the entry in the feature map $C_f \in \mathbb{R}^{h\times(L-f+1)}$ for the i-th filter at position t is

   $$[C_f]_{it} = \sum_{j=1}^{f}\sum_{k=1}^{d_e} A_{ijk}\cdot V_{t+j-1,k} + b_i,$$

   where L is the length of the input sentence. To capture information expressed in phrases of all lengths, we further use n different lengths of filters, and concatenate all C_f under filter size f as the jointed feature map $C \in \mathbb{R}^{nf\times d_e}$:

   $$C = [C_{f_1}|C_{f_2}|\cdots|C_{f_n}].$$

3. **Max pooling**: The max pooling layer captures the most significant feature into the pooling feature P_i by selecting the highest value in the feature map extracted by the i-th filter C_i over all positions:

   $$P_i = \max(C_i).$$

   PCNN (Zeng et al., 2015) involves piecewise max pooling, which better suits the relation extraction task. It divides an input sentence into three segments based on the two selected entities, and then extracts features from all the three segments to capture fine-grained features for relation extraction. For PCNN, the extracted feature map

   $$P_i = [\max(C_{i1})|\max(C_{i2})|\max(C_{i3})],$$

   where C_{i1}, C_{i2}, and C_{i3} are the three feature map segments separated by the two selected entities. We also view P as the sentence feature, as it represents the essential features of the whole sentence.

4. **Fully connected**: The fully connected layer (FC) performs relation classification based on sentence feature P with softmax activation over each relation. The computed logits $O(r)$ is written as

   $$\begin{aligned} O(r) &= \text{softmax}(\text{FC}(P)) \\ &= \text{softmax}(\text{FC}(\text{CNN}(s))). \end{aligned}$$

A.2 Implementation

H-FND was implemented with PyTorch 1.6.0 (Adam et al., 2017) in python 3.6.9. In our implementation, we used pretrained word embeddings provided by SpaCy (Honnibal and Johnson, 2015) as the fixed word embeddings ($d_w = 300$). The positional embedding ($d_p = 50$) was randomly initialized and then trained with the following network; therefore the overall dimension of embedding vector $d_e = d_w + 2d_p = 400$. In the convolutional layer, we applied four different sizes of filters ($f \in$ [2, 3, 4, 5]) and set all of their feature sizes to $h = 230$. Both CNN and PCNN architectures were implemented. The total trainable parameters of each models are listed in table 4. To prevent overfitting, we inserted dropout layers with a dropout rate of 0.5 before the convolutional layer and after the max pooling layer.

We trained H-FND using the Adam optimizer (Kingma and Ba, 2015). In addition, we

#(Params)		SemEval	TACRED	NYT
RC	CNN	1,318,130	1,347,602	1,388,933
	PCNN	1,336,530	1,424,882	1,486,453
RC+SelATT	CNN	1,327,330	1,386,242	–
	PCNN	1,364,130	1,540,802	–
DA	CNN	1,311,683	1,311,683	1,342,883
	PCNN	1,317,203	1,317,203	1,348,403

Table 4: Number of trainable parameters in each model.

used mini-batches (batch size $b = 256$) only when training the relation classifier; the prediction of the relation classifier and both the decision and policy gradient of the denoising agent were executed per epoch. Last, the revised result of H-FND in each epoch was used by the classifier only in the same epoch and did not accumulate over epochs, which means that at the beginning of each epoch, H-FND applied the denoising policy on the original dataset but not on the revised dataset of the last epoch.

We list in Table 5 the learning rates for base CNN and PCNN relation classifiers (RC), for RC with SelATT, and for RC with denoising agent (DA) under pretraining and co-training phrases. The learning rate of RC is selected from {1e-4, 3e-4, 1e-3, 3e-3, 1e-2}, with the F1 score on the noise-free version of SemEval and TACRED as the selection criteria. Except SelATT and DA co-training, the learning rates for the other models are the same to the learning rate of base RC. For SelATT, the learning rate is selected from {1e-6, 3e-6, 1e-5, 3e-5, 1e-4}, also with the F1 score on the noise-free version of the two datasets as the selection criteria. For DA cotraining, the learning rate is selected from {1e-6, 3e-6, 1e-5, 3e-5, 1e-4}, with the F1 score on the SemEval and TACRED under a 50% FN ratio as the selection criteria.

All the RC of each method are trained to converge with validation-based early stopping. In specific, we train all the model for 150 epochs on SemEval and for 200 epochs on TACRED. For NYT, we trained all the models for 30 epochs.

The pretraining of H-FND trains the RC and DA for 5 and 20 epochs respectively. We select these pretraining periods by the criteria that the two models can achieve about 80% performance comparing to the converged ones. By this means, we can prevent H-FND from overfitting the noisy labels (Han et al., 2018) and initialize H-FND with good parameters for co-training.

All the implemented models are trained on NVIDIA GTX 1080 Ti and Intel(R) Xeon(R) Silver 4110 CPU, with 12GN GPU memory, 128GB RAM, clock rate 2.10 GHz, and Linux as the operating system. The expected running time for each model on each dataset is listed in Table 6.

Learning rate	SemEval	TACRED	NYT
lr_{RC}	3e-3	3e-4	3e-4
$lr_{\text{RC, SelATT}}$	1e-5	3e-6	–
$lr_{\text{RC, pre}}$	3e-3	3e-4	3e-4
$lr_{\text{DA, pre}}$	3e-3	3e-4	3e-4
$lr_{\text{RC, co}}$	3e-3	3e-4	3e-4
$lr_{\text{DA, co}}$	1e-4	3e-6	3e-6

Table 5: Learning rates.

Runtime	SemEval	TACRED	NYT
Base	0.05	0.63	3.25
SelAtt	1.95	22.70	–
IRMIE	0.05	0.67	–
Co-teaching	0.10	1.10	–
Cleanlab	0.25	6.47	16.25
H-FND	0.55	15.28	44.44

Table 6: Runtimes for models training (hrs).

A.3 Performance on Validation Set

The F1 scores of each model running on validation sets of SemEval and TACRED are provided in Figure 7 and 8. Notice that the validation sets are noisy in our experiment, so the performance on validation sets do not fully reflect the robustness of each models. Also, in IRMIE and H-FND, the validation sets are modified, so their validation F1 scores can only be compared with their own across different FN ratios. For more accurate performance measurement, please refer to Figure 3 and 4, whose F1 scores are measured on noise-free testing sets.

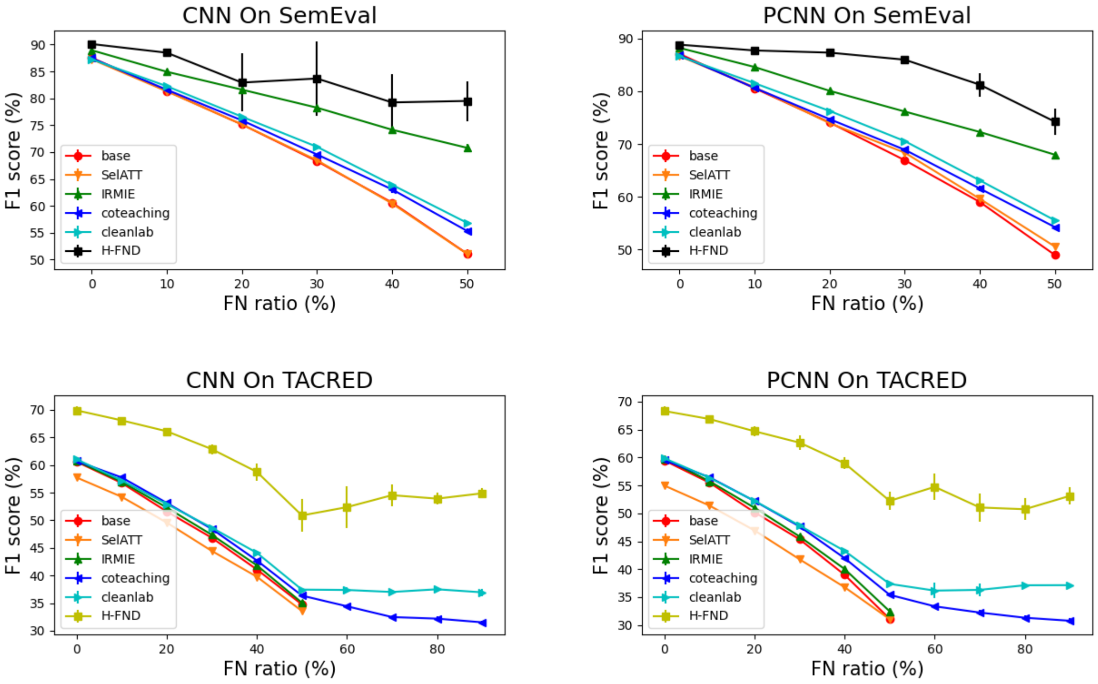

Figure 7: Validation F1 scores of quantitative result, where the errorbars represent the standard deviations.

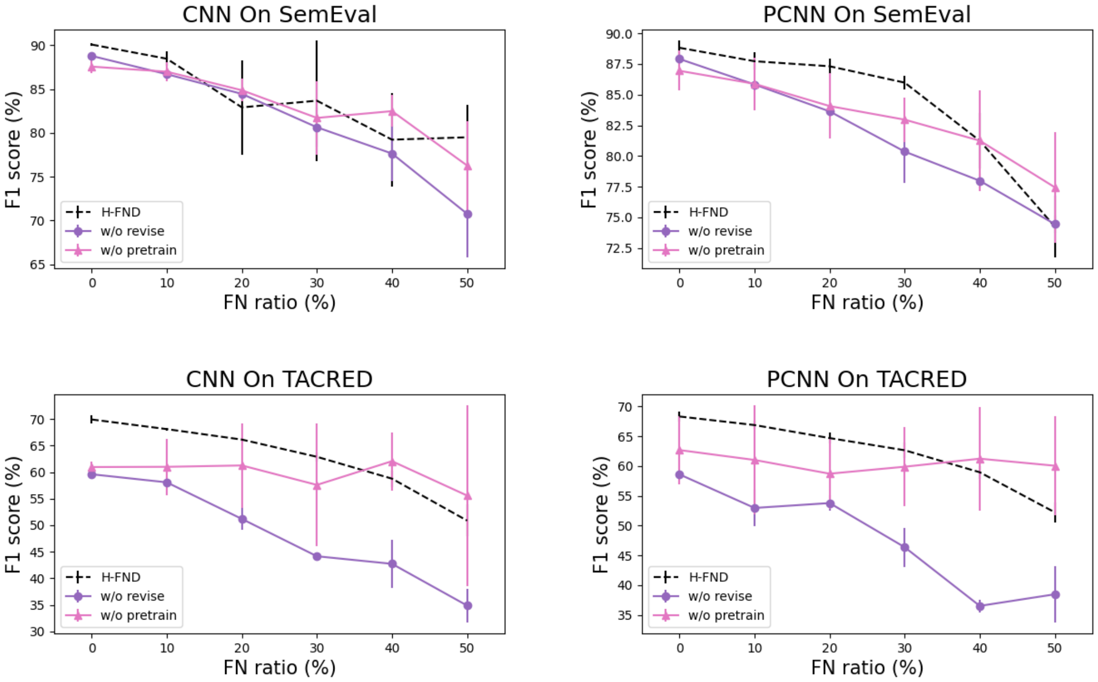

Figure 8: Validation F1 scores of ablation analysis, where the errorbars represent the standard deviations.

A.4 Denoising policy with Standard Deviations

On SemEval and TACRED, the Denoising policy distributions with standard deviation are provided in Table 7, 8, 9, and 10.

CNN on SemEval	0%	10%	20%	30%	40%	50%
TN/Keep	64.89 ± 6.55	74.04 ± 8.37	87.68 ± 9.74	79.17 ± 10.65	83.03 ± 6.65	76.83 ± 11.19
TN/Discard	24.88 ± 9.59	17.82 ± 5.55	8.81 ± 7.28	12.56 ± 6.53	12.34 ± 4.84	11.17 ± 3.15
TN/Revise	10.23 ± 5.00	8.14 ± 5.90	3.51 ± 2.57	8.27 ± 4.47	4.63 ± 1.93	12.00 ± 9.25
FN/Keep	0.00 ± 0.00	19.04 ± 5.65	55.03 ± 26.98	45.97 ± 24.85	50.91 ± 13.15	45.24 ± 11.06
FN/Discard	0.00 ± 0.00	61.58 ± 10.16	35.29 ± 22.58	38.33 ± 17.48	38.81 ± 10.05	34.70 ± 5.73
FN/Revise	0.00 ± 0.00	19.38 ± 9.77	9.68 ± 5.52	15.70 ± 7.49	10.28 ± 3.75	20.05 ± 10.99

Table 7: Denoising policy distribution for CNN on SemEval (%).

PCNN on SemEval	0%	10%	20%	30%	40%	50%
TN/Keep	73.57 ± 6.19	77.79 ± 4.11	77.74 ± 3.93	73.61 ± 5.11	80.72 ± 6.08	82.95 ± 4.45
TN/Discard	21.63 ± 4.49	16.55 ± 4.71	16.24 ± 3.78	18.76 ± 5.51	13.25 ± 4.96	12.15 ± 4.21
TN/Revise	4.79 ± 2.08	5.67 ± 1.89	6.01 ± 2.16	7.62 ± 2.14	6.04 ± 2.17	4.91 ± 0.68
FN/Keep	0.00 ± 0.00	25.37 ± 5.33	36.46 ± 7.12	38.25 ± 5.81	52.36 ± 9.94	60.38 ± 8.54
FN/Discard	0.00 ± 0.00	62.62 ± 8.65	51.12 ± 8.63	48.45 ± 8.44	36.59 ± 8.87	31.31 ± 8.13
FN/Revise	0.00 ± 0.00	12.00 ± 3.51	12.42 ± 3.42	13.30 ± 3.36	11.05 ± 3.10	8.31 ± 1.42

Table 8: Denoising policy distribution for PCNN on SemEval (%).

CNN on TACRED	0%	10%	20%	30%	40%	50%
TN/Keep	80.48 ± 3.49	85.18 ± 1.36	84.07 ± 4.66	89.14 ± 1.38	90.81 ± 1.95	94.11 ± 2.23
TN/Discard	13.99 ± 2.76	10.99 ± 1.16	11.78 ± 2.36	8.50 ± 1.18	7.60 ± 1.61	5.00 ± 1.92
TN/Revise	5.54 ± 0.84	3.82 ± 0.32	4.15 ± 2.44	2.35 ± 0.37	1.59 ± 0.37	0.90 ± 0.35
FN/Keep	0.00 ± 0.00	36.53 ± 2.21	40.36 ± 1.88	47.42 ± 3.88	53.60 ± 4.16	66.31 ± 7.85
FN/Discard	0.00 ± 0.00	32.40 ± 3.08	34.23 ± 3.38	32.12 ± 3.42	31.86 ± 2.45	24.60 ± 5.73
FN/Revise	0.00 ± 0.00	31.07 ± 1.70	25.42 ± 1.79	20.46 ± 1.86	14.54 ± 1.83	9.09 ± 2.31

Table 9: Denoising policy distribution for CNN on TACRED (%).

PCNN on TACRED	0%	10%	20%	30%	40%	50%
TN/Keep	85.31 ± 0.45	85.91 ± 3.20	88.05 ± 2.73	88.60 ± 2.54	90.63 ± 2.23	92.46 ± 1.57
TN/Discard	10.30 ± 0.53	10.55 ± 2.97	9.10 ± 2.39	9.01 ± 2.07	7.41 ± 2.00	6.27 ± 1.40
TN/Revise	4.39 ± 0.34	3.54 ± 0.32	2.85 ± 0.53	2.39 ± 0.70	1.96 ± 0.31	1.26 ± 0.25
FN/Keep	0.00 ± 0.00	39.10 ± 4.23	45.62 ± 4.58	48.53 ± 6.01	57.50 ± 3.90	64.01 ± 4.44
FN/Discard	0.00 ± 0.00	33.03 ± 5.09	31.68 ± 3.64	33.32 ± 4.97	26.67 ± 3.58	25.55 ± 3.42
FN/Revise	0.00 ± 0.00	27.87 ± 1.88	22.70 ± 1.86	18.15 ± 2.82	15.83 ± 1.44	10.45 ± 1.59

Table 10: Denoising policy distribution for PCNN on TACRED (%).

GEM: A General Evaluation Benchmark for Multimodal Tasks

Lin Su[1] **Nan Duan**[2] **Edward Cui**[1] **Lei Ji**[2] **Chenfei Wu**[2] **Huaishao Luo**[3]
Yongfei Liu[4] **Ming Zhong**[1] **Taroon Bharti**[1] **Arun Sacheti**[5]
[1] Bing Multimedia Team, Microsoft, China
[2] Natural Language Computing, Microsoft Research Asia, China
[3] Southwest Jiaotong University, China [4] ShanghaiTech University, China
[5] Bing Multimedia Team, Microsoft, United States
`{lins, nanduan, edwac, leiji, chewu, minzhon, tbharti, aruns}@microsoft.com`
`huaishaoluo@gmail.com, liuyf3@shanghaitech.edu.cn`

Abstract

In this paper, we present **GEM**[1] as a **G**eneral **E**valuation benchmark for **M**ultimodal tasks. Different from existing datasets such as GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019), XGLUE (Liang et al., 2020) and XTREME (Hu et al., 2020) that mainly focus on natural language tasks, GEM is a large-scale vision-language benchmark, which consists of **GEM-I** for image-language tasks and **GEM-V** for video-language tasks. Comparing with existing multimodal datasets such as MSCOCO (Chen et al., 2015) and Flicker30K (Vinyals et al., 2015) for image-language tasks, YouCook2 (Zhou et al., 2018) and MSR-VTT (Xu et al., 2016) for video-language tasks, GEM is not only the largest vision-language dataset covering image-language tasks and video-language tasks at the same time, but also labeled in multiple languages. We also provide two baseline models for this benchmark. We will release the dataset, code and baseline models, aiming to advance the development of multilingual multimodal research.

1 Introduction

In recent years, large-scale pre-training has become the new paradigm in the natural language processing (NLP) field. These models have demonstrated surprisingly good generalization abilities and can be applied to different downstream tasks by a simple fine-tuning. Several comprehensive benchmarks are constructed to evaluate such powerful models, including GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) for evaluating monolingual natural language understanding systems, XGLUE (Liang et al., 2020) and XTREME (Hu et al., 2020) for evaluating multilingual natural language understanding and generation systems. Such pre-trained models have also been extended to vision-language scenarios (Lu et al., 2019; Chen et al., 2019; Li et al., 2020a,b; Ni et al., 2021; Sun et al., 2019b,a; Luo et al., 2020) to handle multimodal tasks such as image(or video)-text retrieval and image (or video) captioning. However, there is still no comprehensive benchmark dataset for evaluating such multimodal pre-trained models. Besides, most existing vision-language datasets are labeled in English only, which cannot be used to evaluate the qualities of such models on other languages.

Motivated by this, we present **GEM**, a **G**eneral **E**valuation benchmark for **M**ultimodal tasks. Comparing with GLUE, SuperGLUE, XGLUE and XTREME, GEM is designed for evaluating the generalization capabilities of vision-language models and consists of two subsets: GEM-I, which evaluates text-to-image retrieval and image captioning capabilities, and GEM-V, which evaluates text-to-video retrieval and video captioning capabilities. Besides, it is also a multilingual dataset, where the natural language contexts are collected from a commercial search engine. We describe two vision-language pre-trained models, M^3P (Ni et al., 2021) and m-UniVL, as the baselines for GEM-I and GEM-V, respectively, where M^3P is an existing multilingual image-language pre-trained model, m-UniVL is a multilingual extension of UniVL (Luo et al., 2020) for multilingual video-language tasks. The evaluation results of these two models on GEM are reported in the experiment part.

The key contribution of this paper is twofold: (1) we build GEM as the first large-scale multilingual multimodal benchmark, which can be used to evaluate the generalization capabilities of vision-language pre-trained models on a set of diversified multimodal tasks. (2) we provide two multilingual multimodal pre-trained models, M^3P and m-UniVL, as the baselines of GEM for image-language and video-language tasks, respectively. We hope GEM can further advance the research in

---

[1] https://github.com/microsoft/GEM

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2594–2603
August 1–6, 2021. ©2021 Association for Computational Linguistics

Language	Train	Dev	Test	Total
English (en)	998,000	1,000	1,000	1,000,000
Spanish (es)	18,000	1,000	1,000	20,000
French (fr)	18,000	1,000	1,000	20,000
Italian (it)	18,000	1,000	1,000	20,000
Portuguese (pt)	18,000	1,000	1,000	20,000
German (de)	18,000	1,000	1,000	20,000
Korean (ko)	8,000	1,000	1,000	10,000
Polish (pl)	8,000	1,000	1,000	10,000
Catalan (ca)	2,000	1,000	1,000	4,000
Dutch (nl)	2,000	1,000	1,000	4,000
Japanese (ja)	2,000	1,000	1,000	4,000
Indonesian (id)	2,000	1,000	1,000	4,000
Vietnamese (vi)	2,000	1,000	1,000	4,000
Czech (cs)	2,000	1,000	1,000	4,000
Romanian (ro)	2,000	1,000	1,000	4,000
Turkish (tr)	0	0	1,000	1,000
Galician (gl)	0	0	1,000	1,000
Croatian (hr)	0	0	1,000	1,000
Hungarian (hu)	0	0	1,000	1,000
Malay (ms)	0	0	1,000	1,000
Total	**1,118,000**	**15,000**	**20,000**	**1,153,000**

Table 1: Language distribution and data statistics of GEM-I for multilingual image-language tasks.

Language	Train	Dev	Test	Total
German (de)	3,316	1,000	1,000	5,316
Portuguese (pt)	3,258	1,000	1,000	5,258
Dutch (nl)	2,961	1,000	1,000	4,961
Spanish (pt)	2,894	1,000	1,000	4,894
Russian (ru)	2,804	1,000	1,000	4,804
French (fr)	2,776	1,000	1,000	4,776
Italian (it)	2,589	1,000	1,000	4,589
Korean (ko)	2,452	1,000	1,000	4,452
English (en)	2,426	1,000	1,000	4,426
Japanese (ja)	2,000	1,000	1,000	4,000
Arabic (ar)	2,000	1,000	1,000	4,000
Polish (pl)	2,000	1,000	1,000	4,000
Chinese-Traditional (zh-t)	2,000	1,000	1,000	4,000
Farsi (fa)	2,000	1,000	1,000	4,000
Indonesian (id)	2,000	1,000	1,000	4,000
Turkish (tr)	2,000	1,000	1,000	4,000
Vietnamese (vi)	2,000	1,000	1,000	4,000
Hebrew (he)	1,807	1,000	1,000	3,807
Romanian (ro)	1,441	1,000	1,000	3,441
Swedish (sv)	1,419	1,000	1,000	3,419
Filipino (tl)	1,294	1,000	1,000	3,294
Malay (ms)	0	0	1,000	2,668
Norwegian (no)	0	0	1,000	1,098
Catalan (ca)	0	0	1,000	1,002
Croatian (hr)	0	0	907	907
Georgian (ka)	0	0	863	863
Chinese-Simplified (zh-s)	0	0	833	833
Hungarian (hu)	0	0	811	811
Albanian (sq)	0	0	809	809
Serbian-Latin (sr-l)	0	0	774	774
Total	**47,437**	**21,000**	**28,997**	**99,202**

Table 2: Language distribution and data statistics of GEM-V for multilingual video-language tasks.

the multimodal community, just as its predecessors did in the NLP community.

2 Dataset Construction

To the best of our knowledge, GEM dataset is the first multilingual vision-language dataset constructed for image-language and video-language tasks as the same time. GEM-I contains 1.2 million {*Query, Image, Title*} triplets in 20 different languages for text-to-image retrieval and image captioning tasks. GEM-V contains 99K {*Query, Video, Title*} triplets in 30 languages for text-to-video retrieval and video captioning tasks. In both GEM-I and GEM-V, *Title* denotes the title of the web page where each image (or video) is extracted. This signal can be used as the auxiliary information in all GEM tasks, as it is usually highly relevant to the corresponding image (or video).

Next, we will describe how GEM-I and GEM-V are collected from a commercial search engine.

2.1 GEM-I Construction

First, we collect several billion images with Creative Commons licenses from the Internet, and discard images that contain pornographic or racy content. We also discard images with human faces, to avoid revealing privacy or introducing bias to our data. Besides, we only keep images which are larger than 300×300 pixels to guarantee high image quality. The pornographic classifier, racy classifier, and human face classifier are trained and evaluated on human-labeled data. The (precision, recall) of them are (0.85, 0.92), (0.79, 0.94), and (0.85, 0.92), respectively.

Then, we collect user queries from a commercial search engine for each image based on user historical clicks. We also collect the title of the Web page that contains the image as the additional context, forming {*Query, Image, Title*} triplets. Some text cleanup work is done to only keep high quality queries and contexts, including removing pornographic words and meaningless strings, and discarding very short queries or titles in that they are less likely to depict the image content, etc. We also apply an in-house GBDT model to filter out potentially highly irrelevant {*Query, Image, Title*} triplets, which is trained using a small amount of human-labeled data, to predict the similarity between each {*Query*} and {*Image, Title*} pair.

Finally, we only keep the top 20 languages which have more than 1000 images, and sample 1.2 million {*Query, Image, Title*} triplets in total. The average length of query in GEM-I is 5.5 terms, which is shorter than 10.6 in MSCOCO (Chen et al., 2015) and 12.3 in Flicker30K (Vinyals et al., 2015). Also, the average length of title is 10.1 terms. This makes GEM-I a more practical benchmark, since all data

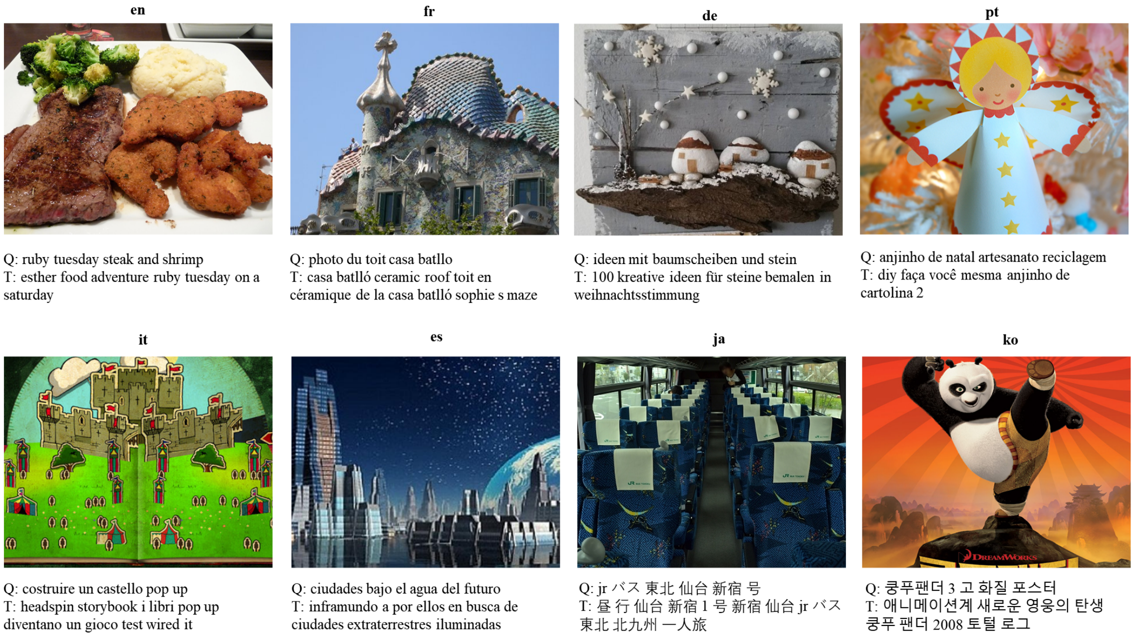
en

Q: ruby tuesday steak and shrimp
T: esther food adventure ruby tuesday on a saturday

fr

Q: photo du toit casa batllo
T: casa batlló ceramic roof toit en céramique de la casa batlló sophie s maze

de

Q: ideen mit baumscheiben und stein
T: 100 kreative ideen für steine bemalen in weihnachtsstimmung

pt

Q: anjinho de natal artesanato reciclagem
T: diy faça você mesma anjinho de cartolina 2

it

Q: costruire un castello pop up
T: headspin storybook i libri pop up diventano un gioco test wired it

es

Q: ciudades bajo el agua del futuro
T: inframundo a por ellos en busca de ciudades extraterrestres iluminadas

ja

Q: jr バス 東北 仙台 新宿 号
T: 昼 行 仙台 新宿 1 号 新宿 仙台 jr バス 東北 北九州 一人旅

ko

Q: 쿵푸팬더 3 고 화질 포스터
T: 애니메이션계 새로운 영웅의 탄생 쿵푸 팬더 2008 토털 로그

Figure 1: Examples in GEM-I dataset: Q: Query, T: Title.

in GEM-I come from the real world, where the language configuration truly differs from the queries in existing datasets. For example, the queries in GEM were shorter and more concise, without perfect grammar or syntax structure. This makes GEM queries more ”natural”. Therefore, our benchmark can evaluate the models on data closer to real-world scenarios, so that the performance of the models will be more convincing in terms of being used in real-world applications. Based on human assessment on sampled query-image pairs, 83% of the them are well matched pairs in that the query is a plausible caption of its paired image. We randomly split the data into train, dev and test sets within each language. The data statistics and language distribution of GEM-I can be found in Table 1. Figure 1 gives some examples.

2.2 GEM-V Construction

We collect several billion videos from the Internet, and discard videos with pornographic or racy contents. We also discard very long videos to save storage and transfer expenses. For each video, its query and title are collected from a commercial search engine and cleaned-up according to a similar process as we described in GEM-I, where another in-house model is trained to filter out potentially irrelevant $\{Query, Video, Title\}$ triplets.

Finally, we only keep the top 30 languages which have more than 700 videos, and sample 99K $\{Query, Video, Title\}$ triplets in total. The total video length of GEM-V is 2,049 hours, and the average video length is 1.2 minutes. The average length of query in GEM-V is 5.3 terms, and that of title is 8.5 terms. We also conduct human evaluation on some sampled query-video pairs, and find 70% of them are plausible matched pairs. We randomly split the data into train, dev and test sets within each language. The data statistics and language distribution of GEM-V can be found in Table 2. Figure 2 gives some examples.

3 Baseline Models

This section will introduce two baseline models for GEM, including M^{3}P, which is a multilingual multimodal pre-trained model for image-language tasks, and m-UniVL, which is a multilingual extension of UniVL (Luo et al., 2020) for video-language tasks.

3.1 M^{3}P as Baseline of GEM-I

We select M^{3}P (Ni et al., 2021) as the baseline model for tasks in GEM-I, as it is the state-of-the-art multilingual image-language pre-trained model for both image-language understanding and generation tasks.

The M^{3}P model uses the model architecture of BERT for understanding tasks and a BERT-based encoder-decoder architecture for generation tasks.

Figure 2: Examples in GEM-V dataset: Q: Query, T: Title.

For understanding tasks, multilingual masked language modeling, multimodal masked language modeling, masked region modeling and visual-linguistic matching are used as pre-training tasks to train a Transformer-based encoder. For generation tasks, multilingual denoising auto-encoding, image captioning and denoising image captioning are used as pre-training tasks to train a Transformer-based encoder-decoder. By training the encoder and the encoder-decoder with a multitask learning framework, universal representations are learned to map objects occurred in different modalities or expressed in different languages to vectors in a common semantic space.

Fine-tune Tasks Based on the pre-trained M^{3}P model, we further finetune it on our GEM-I data. For the text-image retrieval task, we adopt the BCE loss and NCE loss (Gutmann and Hyvärinen, 2010) (with equal weights) to learn the instance-level alignment between texts and images. The negative samples are generated by randomly forming text-image pairs from different training samples in the same batch. For the image captioning task, we directly learn captioning loss on GEM-I data.

Side-Information Since title is considered as the side-information of the image, we concatenate it together with the image and feed them into the model. During the negative sampling process in the retrieval task, we treat title and image as a whole, i.e., for a certain query, the titles and images from other samples are considered as negatives.

3.2 m-UniVL as Baseline of GEM-V

We adopt the same model structure with the unified video and language pre-trained model UniVL (Luo et al., 2020), which can perform both multi-modal understanding and generation tasks. Specifically, we extend the pre-trained UniVL model from monolingual to multilingual by replacing the BERT-based module with XLM-R (Conneau et al., 2019), and call the new model m-UniVL. m-UniVL adopts an encoder-decoder architecture, including two single-modal encoders to encode the multilingual text and the visual features respectively, and one cross-modal encoder to learn the interactions between the two modalities, and finally an optional decoder for generation tasks. To better leverage the pre-trained models, we initialize each module with different pre-trained weights: for the multilingual text encoder, we directly initialize it with the pre-trained XLM-R[2] (Conneau et al., 2019), and for other modules including the visual encoder, the cross encoder and the decoder, we initialize them with the weights of the pre-trained UniVL[3].

Fine-tune Tasks Based on the pre-trained m-UniVL, we further finetune it using GEM-V data. For the text-video retrieval task, we only employ encoders in m-UniVL in the finetuning stage and use them to predict the matching score between text and video. There are two baseline models for the retrieval task: 1) m-UniVL(loose), the loosely coupled model that only uses the single-modal encoders; 2) m-UniVL(tight), the loosely plus tightly coupled model that includes both the single-modal encoders and the cross-modal encoder. We adopt the NCE loss (Gutmann and Hyvärinen, 2010) to learn to discriminate the positive video-text pairs against negative ones. The negative video-text samples are created by replacing the text or video in a positive sample with randomly-selected text or

---

[2] https://huggingface.co/xlm-roberta-base

[3] https://github.com/microsoft/UniVL

video from other samples. For video captioning task, we employ all modules including all the encoders and decoder to learn the caption generation task.

Side-Information In regard to the titles, we use them as the side-information of the videos for an efficient text-video retrieval. In details, we first map the embedding of the title to the same dimension with the video embedding, and then concatenate them together. Then we encode them using the visual encoder to generate the enhanced video features for further processing.

4 Related Work

4.1 Natural Language Benchmarks

GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) are two comprehensive datasets that can be used to train and evaluate natural language understanding systems. GLGE (Liu et al., 2020) is another comprehensive dataset for natural language generation evaluation. XGLUE (Liang et al., 2020) and XTREME (Hu et al., 2020) are two recent benchmark efforts that extend the evaluation scenarios from monolingual to multilingual. Recent pre-trained language models benefit a lot from these datasets, by evaluating their effectiveness under a relatively fair environment.

4.2 Vision-Language Benchmarks

A number of vision-language datasets have been widely used in the multimodal research.

MSCOCO (Chen et al., 2015) and Flicker30K (Vinyals et al., 2015) are two datasets for image-text retrieval and image-captioning tasks. These two benchmarks have been extended to multilingual tasks (Elliott et al., 2016, 2017; Miyazaki and Shimizu, 2016; Li et al., 2019) as well. VQA (Antol et al., 2015) and GQA (Hudson and Manning, 2019) are two datasets for visual question answering. VCR (Zellers et al., 2018) is another dataset for visual commonsense reasoning. Comparing with all these existing datasets, GEM-I has unique characteristics. First, it is a large-scale multilingual image-text dataset covering 20 different languages. Second, the query-image pairs in GEM-I come from a commercial search engine. Therefore, it has big practical values. Third, for each query-image pair, the title of the Web page that contains the image is also included as the additional context, which makes GEM-I different from all existing datasets.

HowTo100M (Miech et al., 2019b), YouCook2 (Zhou et al., 2018), and MSR-VTT (Xu et al., 2016) are three typical benchmarks for video-text retrieval and video captioning tasks. TVQA (Lei et al., 2018) and ActivityNet-QA (Yu et al., 2019) are two typical benchmarks for video question answering. Comparing with all these existing datasets, GEM-V is the first video-language benchmark supporting multilingual scenarios. Similar to GEM-I, it also has big practical values, as all data in GEM-V come from a real-world search engine with massive users.

5 Experiments

In this section, we evaluate two baseline pre-trained models (described in Section 3) on GEM. Specifically, M^3P is evaluated on GEM-I for multilingual image-language tasks and m-UniVL is evaluated on GEM-V for multilingual video-language tasks. For both baseline models, we fine-tune them on downstream tasks directly.

5.1 Image-Language Evaluation on GEM-I

5.1.1 Experimental Settings

We select the open-source version[4] of M^3P (Ni et al., 2021) for the image-language evaluation on GEM-I. It uses 101G sentences (in 100 languages) extracted from Wikipedia as the multilingual pre-training corpus, and uses 3.3 million English image-caption pairs in Conceptual Captions (Sharma et al., 2018) as the multimodal pre-training corpus.

For text-to-image retrieval, the hyper-parameters of the encoder are set as follows: 768 hidden units, 12 heads, GELU activation, a dropout rate of 0.1, 128 max input length, 12 layers in encoder. For image captioning, the hyper-parameters of the encoder-decoder are set as follows: 768 hidden units, 8 heads, GELU activation, a dropout rate of 0.1, 128 max input length, 12 layers in both encoder and decoder. The transformer parameters between the encoder and decoder are shared, including embedding modules and self-attention modules.

We fine-tune M^3P on text-to-image retrieval and image captioning tasks. For retrieval, we use Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, an initial learning rate of 5e-5, a weight decay of 1e-4 and a batch size of 64 to fine-tune M^3P for 30 epochs. For captioning, a learning rate of 1e-4 and a batch size of 16 are used to fine-tune M^3P for 20 epochs. All above calculations are carried on 4 NVIDIA Tesla P100 GPUs.

---

[4] https://github.com/microsoft/M3P

Setting	en	es	fr	it	pt	de	ko	pl	ca	nl	-
Zero-Shot											
M^3P (Q→I)	22.08	8.31	8.51	7.16	7.05	9.31	4.68	4.38	9.10	9.09	-
M^3P (Q→I+T)	5.80	4.16	3.98	2.9	3.71	3.36	2.68	4.07	2.86	3.96	-
Fine-tune on ALL											
M^3P (Q→I)	43.85	26.15	24.83	22.72	27.05	27.18	15.80	32.80	12.83	20.78	-
M^3P (Q→I+T)	93.75	93.16	95.15	93.26	93.73	89.37	67.46	82.67	90.78	90.37	-

	ja	id	vi	cs	ro	tr	gl	hr	hu	ms	AVG
Zero-Shot											
M^3P (Q→I)	7.68	12.06	5.00	5.32	5.81	5.13	4.72	5.30	4.30	8.08	7.65
M^3P (Q→I+T)	4.5	4.53	2.08	3.72	3.61	3.28	2.55	2.58	3.26	3.41	3.55
Fine-tune on ALL											
M^3P (Q→I)	18.88	22.13	10.10	13.33	16.10	13.23	10.38	13.51	11.11	14.33	19.85
M^3P (Q→I+T)	71.90	81.98	54.83	76.23	64.35	71.97	75.00	71.28	63.43	74.18	79.74

Table 3: Evaluation results of M^3P on GEM-I test set for text-to-image retrieval tasks where Mean-Recall is taken as metric. **Q→I** denotes the setting where only image (I) is used to compute its similarity with query (Q), **Q→I+T** denotes the setting where both image (I) and title (T) are used to compute the similarity with query (Q). The average score is computed over all 20 languages.

Setting	Metric	en	es	fr	it	pt	de	ko	pl	ca	nl	-
M^3P	ROUGE-L	6.97	13.86	10.47	9.13	8.35	8.67	3.27	9.31	12.84	3.62	-
M^3P	METEOR	3.21	5.84	4.91	4.14	3.67	3.81	2.21	4.01	5.7	1.54	-
M^3P	CIDEr	17.89	8.68	14.92	11.98	7.93	20.18	7.33	8.44	7.59	6.79	-

	ja	id	vi	cs	ro	tr	gl	hr	hu	ms	AVG	
M^3P	ROUGE-L	4.10	0.96	0.66	4.58	3.98	0.32	9.84	0.25	0.57	0.39	5.61
M^3P	METEOR	1.26	0.47	0.22	1.97	1.78	0.15	4.43	0.11	0.26	0.18	2.49
M^3P	CIDEr	5.01	2.03	1.14	5.72	2.96	0.79	6.86	0.72	1.43	0.91	6.97

Table 4: Evaluation results of M^3P on GEM-I test set for image captioning task where ROUGE-L, METEOR and CIDEr are taken as metrics. Only images (without title) are used to test its multilingual multimodal captioning ability. The average score is computed over all 20 languages.

5.1.2 Text-to-Image Retrieval Results

We follow the same evaluation metric, mean-Recall (average score of R@1, R@5, R@10), in M^3P to report its the performance on text-to-image retrieval task on GEM-I dataset.

From the results reported in Table 3, we have several observations:

1) When applying M^3P to GEM-I without fine-tuning (i.e. the zero-shot setting), the general performance is poor. The major reason is that M^3P is pre-trained on a monolingual multimodal corpus and a multilingual monomodal corpus, and both datasets have very different data distributions comparing with GEM-I.

2) By fine-tuning M^3P using all labeled data in different languages (i.e. the fine-tune on all setting), better performance can be obtained. This shows the strong transfer ability of M^3P, when there is a moderate amount of labeled data for fine-tuning.

3) By furthering considering the title signal in this retrieval task, the general performance can be improved significantly. This indicates the strong correlation between the query and the title. Besides, when taking the title signal into the zero-shot setting, we can observe a performance drop. It is due to that M^3P is pretrained with input paradigm Q-I, thus making it not suitable for evaluating Q-I-T paradigm directly.

5.1.3 Image Captioning Results

As in Table 4, we report the performance of image captioning tasks on GEM-I test set with M^3P model where ROUGE-L (Lin and Och, 2004), METEOR (Banerjee and Lavie, 2005) and CIDEr (Vedantam et al., 2015) are taken as the evaluation metrics. To study the image captioning ability of M^3P, we only use images (without title) to generate queries in GEM-I dataset. In general, the M^3P model performs relatively poor on this task, due to that most search queries are short keywords instead of a complete sentence, and they differ from our pre-training data a lot.

From the above results from text-to-image retrieval task and the image captioning task, we can conclude that our proposed GEM-I dataset can demonstrate a model's image understanding and generation ability in 20 different languages.

Setting	en	id	pt	vi	ro	ko	ja	fr	ar	de	tl	sv	fa	he	it	-
								Fine-tune on ALL								
m-UniVL(loose) (Q→V)	23.27	17.67	23.50	12.83	19.90	12.83	12.37	24.77	9.03	22.27	11.23	16.07	9.87	9.50	23.60	-
m-UniVL(loose) (Q→V+T)	71.83	57.03	62.90	38.97	53.57	42.09	38.87	62.83	36.50	56.67	43.37	40.20	39.67	35.20	56.80	-
m-UniVL(tight) (Q→V+T)	83.87	69.97	61.20	52.07	59.08	63.17	66.10	74.47	46.92	70.27	59.23	51.60	58.10	56.60	58.90	-
								Fine-tune on EACH								
m-UniVL(loose) (Q→V+T)	28.90	19.97	21.23	12.33	14.93	16.87	13.30	20.73	10.57	21.70	14.60	12.23	12.03	10.27	18.70	-
	tr	ru	nl	pl	zh-t	es	ms	no	ca	hr	ka	zh-s	hu	sq	sr-l	AVG
								Fine-tune on ALL								
m-UniVL(loose) (Q→V)	18.70	18.10	20.00	22.50	16.00	25.13	8.10	16.70	9.05	13.38	7.67	9.64	11.56	6.80	11.03	15.44
m-UniVL(loose) (Q→V+T)	51.43	57.73	49.73	49.80	45.70	66.03	43.97	40.60	41.40	38.37	28.00	38.38	40.69	33.00	35.70	46.57
m-UniVL(tight) (Q→V+T)	63.13	70.57	59.07	50.67	69.10	64.97	64.60	54.40	59.37	31.50	39.05	64.31	58.73	50.97	32.86	58.83
								Fine-tune on EACH								
m-UniVL(loose) (Q→V+T)	21.00	17.70	21.70	19.50	17.40	19.83	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	17.40

Table 5: Evaluation results of m-UniVL on GEM-V for text-to-video retrieval, where Mean-Recall is used as the metric. **Q→V** denotes the setting where only video (V) is used to compute its similarity with query (Q). **Q→V+T** denotes the setting where both video (V) and title (T) are used to compute their similarity with query (Q). The average score is computed over all 30 languages.

5.2 Video-Language Evaluation on GEM-V

5.2.1 Experimental Settings

We select the open-source version[5] of UniVL (Luo et al., 2020) and replace the original text encoder with XLM-R (Conneau et al., 2020), to support the multilingual video-language evaluation on GEM-V. The original UniVL is pre-trained on 1.2 million instructional videos with ASR transcripts in HowTo100M (Miech et al., 2019b).

For text-to-video retrieval task, m-UniVL extracts video features using the off-the-shelf pre-trained S3D (Miech et al., 2019a) model. The FPS of the 3D feature extractor is 16 and the dimension is 1,024. The hyper-parameters of the video encoder are set as follows: 768 hidden units, 12 heads, 6 layers of of Transformer blocks to capture the sequential information on the 3D features. The hyper-parameters of the text encoder are identical to the ones in XLM-R: 768 hidden units, 12 heads, 12 layers of Transformer blocks. The cross encoder on the top of the text and video encoders has 2 layers with 768 hidden units and 12 heads. For video captioning, the decoder is with 3 layers, 768 hidden units and 12 heads.

We finetune m-UniVL on text-to-video retrieval and video captioning tasks. For retrieval, a learning rate of 1e-4 and a batch size of 128 are used to fine-tune m-UniVL for 50 epochs. For captioning, a learning rate of 3e-5 and a batch size of 16 are used to fine-tune m-UniVL for 5 epochs. All above calculations are carried on 4 NVIDIA Tesla V100 GPUs.

5.2.2 Text-to-Video Retrieval Results

Following official UniVL on retrieval task, we evaluate the text-to-video retrieval task on our GEM-V using two variants. One is m-UniVL (loose), which encodes the input text query and candidate video clips (and optional title) through the text encoder and video encoder respectively and finally calculates the matching score through dot product. The other is m-UniVL (tight), based on m-UniVL (loose), m-UniVL (tight) further concatenates the encoded features and feeds them to the cross encoder to get unified representation and predict the matching score on the first token '$\langle$s$\rangle$'. The evaluation metric is mean-Recall (arithmetic mean of Recall@K for $K \in \{1, 5, 10\}$).

Tables 5 lists the retrieval results. The results can be divided into two groups. One is from the fine-tuning on all training set across linguistic type (*Fine-tune on ALL*), and the other is from the fine-tuning on individual training set of each language (*Fine-tune on Each*). The target of such a division is to explore whether one language can benefit from other wide languages. Besides, there are 9 languages without training set. We keep such a zero-shot evaluation to explore the transfer ability of the proposed model.

We can get three conclusions from the results:

1) The m-UniVL (tight) outperforms m-UniVL (loose) at the same retrieval settings. It proves that the cross encoder of UniVL enables the multi-modality features to fully interact with each other to capture better alignment.

2) The title of the video introduces a large performance gain and is a good semantic feature of the video. This metadata is especially useful for

---

[5] https://github.com/microsoft/UniVL

Setting	Metric	en	id	pt	vi	ro	ko	ja	fr	ar	de	tl	sv	fa	he	it	-
Fine-tune on ALL																	
m-UniVL (V→Q)	ROUGE-L	9.43	10.95	14.63	6.64	9.96	3.41	3.10	16.87	3.80	8.18	9.72	8.55	8.25	2.28	10.65	-
m-UniVL (T→Q)	ROUGE-L	46.06	35.89	36.68	27.36	35.08	21.01	17.14	41.64	22.40	30.30	43.36	26.10	30.58	33.07	34.43	-
m-UniVL (V+T→Q)	ROUGE-L	46.01	36.73	37.59	26.75	36.41	20.71	17.12	41.90	23.45	30.24	44.40	26.38	30.21	32.82	34.98	-
m-UniVL (V→Q)	METEOR	3.93	4.51	6.09	2.39	4.09	2.99	3.38	8.06	15.16	3.71	4.31	3.62	17.06	12.04	4.65	-
m-UniVL (T→Q)	METEOR	23.99	17.01	17.40	13.13	17.10	20.77	18.33	21.25	26.47	14.40	22.67	12.23	26.25	29.33	16.60	-
m-UniVL (V+T→Q)	METEOR	23.98	17.44	18.01	12.52	18.09	20.34	18.24	21.39	26.76	14.52	23.00	12.26	26.17	28.91	16.94	-
m-UniVL (V→Q)	CIDEr	19.16	18.35	23.58	13.53	18.56	7.12	4.20	43.82	5.72	18.23	23.40	8.99	14.88	4.75	17.53	-
m-UniVL (T→Q)	CIDEr	256.65	155.05	164.26	116.46	164.89	74.95	57.15	223.09	86.57	138.50	220.58	101.64	119.49	139.39	170.02	-
m-UniVL (V+T→Q)	CIDEr	255.04	157.64	167.79	108.66	174.59	74.01	58.73	223.67	90.70	138.66	223.21	100.61	115.37	136.24	174.44	-
Fine-tune on EACH																	
m-UniVL (V+T→Q)	ROUGE-L	21.30	12.25	18.36	6.45	9.09	7.59	3.77	15.20	4.92	8.94	14.92	8.11	7.31	1.76	10.05	-
m-UniVL (V+T→Q)	METEOR	9.50	4.92	7.99	2.44	3.67	6.79	3.48	7.01	17.21	3.92	7.12	3.47	16.66	12.75	4.22	-
m-UniVL (V+T→Q)	CIDEr	65.52	24.26	44.69	15.47	21.45	23.40	9.50	43.71	8.35	21.32	41.04	9.89	13.65	3.71	18.12	-
	tr	ru	nl	pl	zh-t	es	ms	no	ca	hr	ka	zh-s	hu	sq	sr-l	AVG	
m-UniVL (V→Q)	ROUGE-L	11.53	14.44	9.83	14.19	3.95	18.07	0.15	0.91	0.27	2.49	0.00	0.11	2.68	1.29	0.33	6.89
m-UniVL (T→Q)	ROUGE-L	32.19	35.41	27.77	29.99	15.72	41.35	34.44	21.35	38.29	24.94	1.70	4.63	30.13	27.22	28.28	29.15
m-UniVL (V+T→Q)	ROUGE-L	34.43	35.84	28.31	29.82	15.50	42.29	36.36	21.59	38.09	26.09	2.92	4.55	30.58	28.37	29.51	29.67
m-UniVL (V→Q)	METEOR	5.41	6.11	4.47	6.06	2.65	8.21	0.07	0.39	0.13	1.05	0.00	0.61	1.18	0.67	0.13	4.44
m-UniVL (T→Q)	METEOR	15.53	16.79	13.21	13.91	16.03	20.50	16.88	9.77	19.21	12.00	1.74	12.24	13.66	13.17	13.53	16.84
m-UniVL (V+T→Q)	METEOR	16.54	17.12	13.62	13.86	15.48	20.83	17.92	9.83	18.95	12.47	2.76	11.59	13.78	13.72	13.81	17.03
m-UniVL (V→Q)	CIDEr	28.84	24.05	17.35	17.37	8.00	32.89	0.15	1.65	0.45	2.20	0.00	0.29	2.71	1.70	0.73	12.67
m-UniVL (T→Q)	CIDEr	149.09	161.47	123.79	117.37	47.94	203.88	156.38	84.81	192.65	102.38	3.95	13.58	129.47	109.78	119.64	130.16
m-UniVL (V+T→Q)	CIDEr	156.35	166.33	124.20	116.29	47.13	204.85	166.13	82.70	189.35	106.02	8.01	12.85	128.94	114.98	118.01	131.38
Fine-tune on EACH																	
m-UniVL (V+T→Q)	ROUGE-L	11.46	17.52	9.16	13.31	3.15	24.21	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	10.90
m-UniVL (V+T→Q)	METEOR	5.20	7.44	3.93	5.65	2.08	11.19	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	6.98
m-UniVL (V+T→Q)	CIDEr	29.62	36.11	16.48	21.15	6.56	62.26	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	25.54

Table 6: Evaluation results of m-UniVL on GEM-V for video captioning, where ROUGE-L, METEOR and CIDEr are taken as metrics. **V→Q** and **T→Q** denote the video caption is generated based on video (V) and title (T), respectively. **V+T→Q** denotes the video caption is generated based on both video (V) and title (T). The average score is computed over all 30 languages.

zero shot setting with a significant improvement. They demonstrate that improving the retrieval performance on pure videos without titles is still a challenge. Our proposed GEM develops a chance to push such a multimodal challenge.

3) Fine-tune on all can achieve better results than fine-tune on each. The reason is the former can effectively leverage the data from all languages and benefit the task rather than the latter. Besides, for zero shot languages, fine-tune on all is also very effective. It demonstrates that our proposed GEM can also be used on multilingual research besides multimodal research.

5.2.3 Video Captioning Results

The captioning task aims to generate a caption given a video clip (and optional title) in our setting. Such a generation task is from our practical application. We adopt whole m-UniVL including encoders and decoder to finish the task. The evaluation metric are ROUGE-L (Lin and Och, 2004), METEOR (Banerjee and Lavie, 2005) and CIDEr (Vedantam et al., 2015), whose values are obtained from an open-source tool[6].

Table 6 lists the experimental results. Similar conclusions can be drawn as the retrieval task, and there are two more observations:

1) For captioning task, the performance of the generation on pure videos is low. The reason is that the search queries sometimes are the keywords instead of a whole sentence, thus the task of V→Q is relatively hard.

2) Title is especially important due to the characteristic of this data collection process.

From the above results from the text-to-video retrieval task and the video captioning task, we can conclude that our proposed GEM-V can improve video understanding and generation under the multilingual and multimodal perspective.

6 Conclusion

This paper presents GEM as a benchmark for evaluating the generalization capabilities of vision-language models on image-language and video-language tasks. GEM is also the first large-scale multilingual multimodal dataset, where the natural language contexts are collected from a commercial search engine in 20 and 30 languages for image-related and video-related tasks, respectively. We describe two vision-language pre-trained models for GEM and hope these efforts can advance the development of multilingual multimodal research.

7 Ethical Considerations

We have reviewed our data release process and it has been approved by our institutional review board. Specifically, (a) In GEM-I, all of the images

---

[6] https://github.com/Maluuba/nlg-eval

are with proper Creative Commons Licenses, so that they are safe to be distributed without violating any policies or intellectual rights. Also, we discarded images with human faces to avoid revealing privacy. (b) In GEM-V, all of the videos were originated from Youtube, and we will only provide Youtube URLs to the researchers. We have confirmed with our institutional review board that distributing URLs does not violate any policies or intellectual rights. We didn't do anything specific for human faces in the videos, since we are only distributing video URLs, and modifying the original videos (such as blurring the faces) might violate the copyright of the videos. When releasing GEM to the public, we will indicate the data source, emphasize that the dataset is for research purposes only, and provide an email address for people to contact us to delete any data if any infringement. During data collection, we didn't collect, store, or distribute any private information of the users.

To measure the quality of our dataset, we employed crowd-sourcing judges in the United States and provided labeling guidelines for them. The compensation given to the workers is 15 USD per hour for GEM-I and 25 USD per hour for GEM-V. The level of compensation is determined by: (a) Market price according to similar labeling tasks in the US. (b) The difficulty and labeling speed of this task. This task involves labeling if a query is related to an image or video, so it is considered as a relatively easy task. The labeling speed is about 300 query-image pairs per hour and 180 query-video pairs per hour.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the second shared task on multimodal machine translation and multilingual image description. *arXiv preprint arXiv:1710.07177*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. *arXiv preprint arXiv:1605.00459*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 297–304.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. 2018. TVQA: localized, compositional video question answering. *CoRR*, abs/1809.01696.

Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. 2020a. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. *AAAI*.

Xirong Li, Chaoxi Xu, Xiaoxu Wang, Weiyu Lan, Zhengxiong Jia, Gang Yang, and Jieping Xu. 2019. Coco-cn for cross-lingual image tagging, captioning and retrieval. In *IEEE Transactions on Multimedia*.

Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020b. Oscar: Object-semantics aligned pre-training for vision-language tasks. *arXiv preprint arXiv:2004.06165*.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, Pengcheng Wang, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, Ruofei Zhang, Winnie Wu, Ming Zhou, and Nan Duan. 2020. Glge: A new general language generation evaluation benchmark.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23.

Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. 2020. Univl: A unified video and language pre-training model for multimodal understanding and generation.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2019a. End-to-end learning of visual representations from uncurated instructional videos. *CoRR*, abs/1912.06430.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019b. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *CoRR*, abs/1906.03327.

Takashi Miyazaki and Nobuyuki Shimizu. 2016. Cross-lingual image caption generation. In *ACL*.

Minheng Ni, Haoyang Huang, Lin Su, Edward Cui, Taroon Bharti, Lijuan Wang, Jianfeng Gao, Dongdong Zhang, and Nan Duan. 2021. M3p: Learning universal representations via multitask multilingual multimodal pre-training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. 2019a. Contrastive bidirectional transformer for temporal representation learning. *ArXiv*, abs/1906.05743.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019b. Videobert: A joint model for video and language representation learning. *Proceedings of the IEEE International Conference on Computer Vision*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. *CoRR*, abs/1906.02467.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2018. From recognition to cognition: Visual commonsense reasoning. *CoRR*, abs/1811.10830.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*.

Graph Relational Topic Model with Higher-order Graph Attention Auto-encoders

Qianqian Xie
Department of Computer Science,
University of Manchester
`xieq@whu.edu.cn`

Jimin Huang
School of Computer Science,
Wuhan University
`huangjimin@whu.edu.cn`

Pan Du
Department of Computer Science and Operations
Research, University of Montreal
`pandu@iro.umontreal.ca`

Min Peng
School of Computer Science,
Wuhan University
`pengm@whu.edu.cn`

Abstract

Learning low-dimensional representations of networked documents is a crucial task for documents linked in network structures. Relational Topic Models (RTMs) have shown their strengths in modeling both document contents and relations to discover the latent topic semantic representations. However, higher-order correlation structure information among documents is largely ignored in these methods. Therefore, we propose a novel graph relational topic model (GRTM) for document network, to fully explore and mix neighborhood information of documents on each order, based on the Higher-order Graph Attention Network (HGAT) with the log-normal prior in the graph attention. The proposed method can address the aforementioned issue via the information propagation among document-document based on the HGAT probabilistic encoder, to learn efficient networked document representations in the latent topic space, which can fully reflect document contents, along with document connections. Experiments on several real-world document network datasets show that, through fully exploring information in documents and document networks, our model achieves better performance on unsupervised representation learning and outperforms existing competitive methods in various downstream tasks.

1 Introduction

Document networks, such as hyperlink networks of Web pages, citation networks of academic documents, and user profiles in social networks, have long been an intensively studied research subject due to their wide applications. Finding low-dimensional representations of networked documents to preserve document contents and connections among documents simultaneously is a crucial research task. Inspired by the wide application of topic models such as latent Dirichlet allocation (LDA) (Blei et al., 2003) on discovering the latent semantic structure of unconnected documents, a series of Relational Topic Models (RTMs) are proposed to explore the latent topic semantic structure of documents and links among them, based on probabilistic graphical models (Nallapati et al., 2008; Chang and Blei, 2009; Le and Lauw, 2014; Chen et al., 2014; Yang et al., 2016), deep generative models (Acharya et al., 2015; Wang et al., 2017; Bai et al., 2018), auto-encoders (AEs) (Zhang and Lauw, 2020) and graph auto-encoders (GAEs) (Wang et al., 2020a).

However, most RTMs consider only the pairwise correlation or the first-order neighbor correlation (Zhang and Lauw, 2020) among documents. Although the recently proposed deep relational topic model, GPFA (Wang et al., 2020a) based on graph neural networks (GNNs) can consider low-order indirect neighborhood information via stacked graph neural network (GNN) (Kipf and Welling, 2016b) layers, it still suffers from exploiting the deep interactions (higher-order) between indirectly connected documents due to the oversmoothing problem (Li et al., 2018). Such the higher-order correlation structure has been proved to be effective on various tasks (Abu-El-Haija et al., 2019) such as link prediction and recommendation (Zhang and McAuley, 2020).

To address the aforementioned issue, we propose the graph relational topic model (GRTM) for modeling the latent topic structure of document contents and links, based on the higher-order graph attention auto-encoders (HGTAEs), aiming to fully explore and fuse each order proximity (including the low-order and higher-order) of document network. Specifically, we propose to extract the higher-order document proximity network (HDPN) from the adjacent matrix of the document network via the calculation of the shortest path. The higher-order graph attention network (HGAT) is presented

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2604–2613
August 1–6, 2021.

to efficiently model the neighborhood information propagation on HDPN via introducing the log-normal prior into the graph attention. We finally propose our GRTM with the higher-order graph attention auto-encoders (HGTAEs) based on HGAT and HDPN. The main contributions of our paper are as follows:

1. We propose a novel unsupervised deep relational topic model GRTM to fully explore multiple information: the higher-order document relations and latent topic semantic among document contents and networks.

2. We propose a novel graph attention network HGAT to efficiently explore each order correlations among networked documents.

3. Experimental results on document network datasets show that our model outperforms existing competitive methods on unsupervised representation learning, through fully exploring multi-granularity information in document networks.

2 Related Work

In this section, we briefly review existing Relational Topic Models (RTMs), Graph Auto-Encoders (GAEs), and Graph Topic Models.

RTMs generally extended LDA based topic models to further model the links between documents in networks. Chang and Blei (2009) first proposed to introduce additional binary conditional variables in the generation to model the document links. (Chen et al., 2014) (2014) proposed discriminative relational topic models (DRTMs) to learn discriminative latent representations of document networks. Le and Lauw (2014) proposed PLANE which can jointly extract topics and visualization coordinates. To apply the neural network based inference approach to RTMs, Bai et al. (2018) utilized Stacked Variational Auto-Encoder(SVAE) to derive more representative documents in topic distributions. However, these models only consider pair-wise document correlations, fail to model the full structural information (low-order and higher-order) embedded in the document network.

To model the block correlation structure of the document network, Yang et al. (2016) incorporated weighted stochastic block model into relational topic models. Most recently, Zhang and Lauw (2020) proposed AdjEnc to reconstruct both documents and their neighborhoods in the network. However, it can only capture the first-order correlation structure with the adjacent-encoder. Wang et al. (2020a) proposed the deep relational topic model GPFA based GNNs to explore hierarchical relationships of interconnected documents. However, still, it can only capture the low-order hierarchical relationships of interconnected documents due to the well-known smoothing problem of GNNs, while long-range relations among documents are also critical for learning latent representations in document networks. To address this issue, we calculate the higher-order proximity network that allows considering the long-range topological information among documents, rather than merely pairwise or few-order relations.

Recently, GAEs has attracted a lot of attention, which incorporates GNNs into auto-encoder to unsupervised graph embedding learning, motivated by the successful applications of GNNs in modeling graph topological structure. The earliest attempt VGAE (Kipf and Welling, 2016a) extended variational auto-encoder (VAE) onto graph structure data for learning network embedding. Inspired by the advantage of GNNs, some works have explored VGAE for topic modeling, including the deep relational topic model GPFA mentioned before, and GraphBTM (Zhu et al., 2018) which improved the biterm topic model (Yan et al., 2013) with word co-occurrence graph encoded by GCNs. Except studies based on VGAE, there are also works combining topic models with graph neural networks in a different manner, such as the graph attention topic network (GATON) (Yang et al., 2020) proposed for unconnected documents, the dynamic hierarchical topic graph model DHTG (Wang et al., 2020b) used for unconnected document classification, the topic variational graph auto-encoder (TVGAE) (Xie et al., 2021b) for document classification and the graph topic neural network (GTNN) (Xie et al., 2021a) proposed for representation learning of both connected and unconnected documents. Different from them, we target connected documents. Moreover, to fully explore the deep topological structure of document networks, we propose the novel higher-order graph attention network, and then introduce it into the relational topic modelling based on variational graph auto-encoders.

3 Method

In this section, we present our graph relational topic model (GRTM) for the document network. We first introduce the construction of the higher-order proximity networks: HDPN from document adjacency matrices, then we present the novel graph attention network HGAT to fuse the information of HDPN. We end this section by introducing the variational graph auto-encoder structure for building GRTM.

3.1 Higher-order Proximity Network

Formally, we define a given document network as $\mathcal{G} = (D, A, X)$. $D = \{d_1, ..., d_n\}$ is the set of document nodes with n documents and a vocabulary V with m words. Relations between documents are represented as a 0-1 adjacency matrix $A \in \mathbb{R}^{n \times n}$, and $X \in \mathbb{R}^{n \times m}$ is the documen-word index matrix, in which X_{ij} represents the weight (e.g. TF-IDF) of word j in document i. For the given document sets D, Based on the given adjacency matrices A of document network, the key problem is to discover and preserve arbitrary-order neighborhood relations beyond first-order or few-order (including other higher-order). Intuitively, two nodes have a proximity correlation if and only if we can find at least one path between them (Liu et al., 2019). Thus, we can calculate the order of proximity correlation between two nodes according to the length of the shortest path between them based on the adjacency matrix, and directly preserve arbitrary-order information in the same matrix. Denoting the adjacency matrices of HDPN as $\hat{A} \in \mathbb{R}^{n \times n}$, the link of proximity correlation between two documents (d_i, d_j) is defined as:

$$\hat{A}_{i,j} = \begin{cases} k, & \textit{existing a k-length shortest path} \\ 1, & i = j \\ \infty, & \textit{no path} \end{cases} \tag{1}$$

According to the above definition, $\hat{A}$ can be calculated during the data pre-processing step in advance. The length of the shortest path of two nodes is calculated using classical search algorithms such as Dijkstra's algorithm or Bellman-Ford algorithm on the machine learning framework [1]. Compared with existing methods that calculate the higher-order proximity with the power of adjacency matrix or steps in a probabilistic transition process (Abu-El-Haija et al., 2019; Liu et al., 2019), our calculation is more suitable for explicitly calculating the length of the shortest path. Because the k-power of adjacency matrix has proximity information overlap on other power matrices before it, while the calculation of k-walk may lead to nodes return to their neighbors less than k-order rather than reach to their k order neighbor (Zhang and Xu, 2020).

3.2 Higher-order Graph Attention Network

In this section, we focus on how to better fuse information of neighbors at different orders on an HDPN to efficiently learn node representations. Intuitively, for a given node representation, the contributions of its neighbors vary according to their distances. However, directly utilizing GNNs such as graph convolutional networks (GCNs) and graph attention networks (GATs) on the HDPN will treat neighbors of nodes at different orders equally. Therefore, we present the higher-order graph attention network (HGAT) to solve the problem via introducing the log-normal prior into the graph attention.

Instead of utilizing uniform prior in GATs, we exploit the log-normal distribution to model the importance decaying of neighbors of the current node on different orders. For simplicity sake, we use the log-normal distribution with zero-mean value, and calculate the attention coefficient between two nodes i, j:

$$\begin{aligned} \tilde{h_i}^l, \tilde{h_j}^l &= W^l h_i^l + b^l, W^l h_j^l + b^l \\ e_{ij} &= \rho(\lambda \phi(p_{ij})(\tilde{h_i}^l \cdot (\tilde{h_j}^l)^\intercal)) \\ \alpha_{i,j} &= softmax(e_{i,j}) = \frac{\exp(e_{ij})}{\sum_{o \in \mathcal{N}(i)} \exp(e_{io})} \end{aligned} \tag{2}$$

where $\phi(p) = \frac{\exp^{-(\ln p)^2/\sigma^2}}{p\sigma\sqrt{2\pi}}$ is the probability density function of the log-normal distribution, σ is the variance of the log-normal prior, p_{ij} is the length of the shortest path between nodes i, j in a HPN, h_i^l, h_j^l are representations of node i, j in the l-th layer, ρ is the activation function, λ is the parameter to control the influence of the log-normal prior, $\mathcal{N}(i)$ is neighbors of node i in a HPN, W^l, b^l are the weight matrix and bias of the l-th layer. The attention mechanism based on the log-normal prior allow nodes to select their neighbors at arbitrary-order with different importance. Calculated attention coefficients are further exploited to propagate information of neighbors of each node at arbitrary-

---

[1] https://networkx.github.io/documentation/networkx-1.10/overview.html

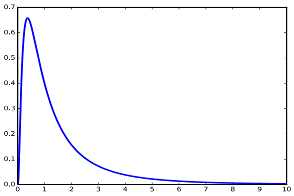

Figure 1: The example of the log-normal distribution with $\mu = 0, \sigma = 1$.

order:

$$h_i^{l+1} = \rho(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} h_j^l) \tag{3}$$

Although there are other distributions, such as the normal distribution used in the Gaussian transformer (Guo et al., 2019), the log-normal is more suitable for model the importance calculation in our case. This is because the density function $\phi(p)$ of the log-normal prior with zero-mean value has always a real value larger than 0 rather than that of the normal distribution with the negative value or Poisson distribution with an integer value. Moreover, $\phi(d)$ always decreases monotonically after $\exp^{-\sigma^2} < 1$, while the path length in our HDPN is always greater or equal to 1. Therefore, the log-normal prior can naturally model the decay of importance weight by increasing the path length between two nodes, as shown in Figure 1. When conducting the HGAT on the vanilla adjacency matrix (only with first-order proximity), the HGAT will be degenerated into the GAT method due to there are only 1 or ∞-length of the shortest path in the vanilla adjacency matrix.

3.3 Graph Relational Topic Modelling

To fully explore the long-range document relations to model the latent topic semantic among document contents and networks, we present the GRTM model with the higher-order graph attention auto-encoders (HGTAEs). Let's assume K is the topic number, θ is the document topic proportion, and β is the topics, namely the topic word proportion.

Firstly, we present the generative process of GRTM as in Algorithm 1. Similar to previous RTMs, we assume the document topic proportion is generated from the Dirichlet prior. However, the Dirichlet prior makes it difficult to make the neural variational inference for GRTM, due to the challenge of reparameterizing the Dirichlet prior. Thus, to simplify the inference process, we approximate the Dirichlet distribution with its Laplace approximation: the logistic normal distribution following many previous works (Srivastava and Sutton, 2017).

Algorithm 1: Generative Process of GRTM

```
for each document $d \in D$ do
    Generate the mean vector:
        $\mu_d^0 \sim \text{HGAT}_\mu(X_d, \hat{A}_d)$.
    Generate the diagonal covariance:
        $\sigma_d^0 \sim diag(\text{HGAT}_\sigma(X_d, \hat{A}_d))$.
    Draw the noise variable $\epsilon_d \sim \mathcal{N}(0, I)$.
    Draw the document topic proportion: $\theta_d \sim$
        $LN(\mu_d^0, \sigma_d^0) = softmax(\mu_d + (\sigma_d^0)^{\frac{1}{2}} \epsilon_d)$.
    for each word $w_{dv} \in V$ do
        Draw the word $w_{dv} | \theta_d, \beta \sim \text{Mult}(\theta_d \beta)$.
for each pair of document $d, d' \in D$ do
    Draw the observed link
        $A_{d,d'} | \theta_d, \theta_{d'} \sim \text{Bernoulli}(f_y(\theta_d, \theta_{d'}))$
```

To incorporate the higher-order relations among documents, we generate the document topic proportion with logistic normal distribution parameterized by HGAT probabilistic encoder. Specifically, for each document d, we draw the mean and covariance of a multinomial distribution variable and then transform it with the softmax function:

$$\begin{aligned}
\mu_d^0 &= \text{HGAT}_\mu(X_d, \hat{A}_d) \\
\sigma_d^0 &= diag(\text{HGAT}_\sigma(X_d, \hat{A}_d)) \\
\theta_d &= softmax(\mu_d^0 + (\sigma_d^0)^{\frac{1}{2}} \epsilon_d)
\end{aligned} \tag{4}$$

where HGAT is the message passing process as in Equation 3, ϵ_d is the noise variable. For HGAT encoders of mean and covariance, the input feature h_d^0 is set to the normalized document-word index feature X_d following previous methods (Kipf and Welling, 2016a). The message passing based on HGAT makes latent topic proportions of each document influenced by its neighbors at different orders with different importance.

In the decoding process, the word is generated from the multinomial distribution based on the topic proportion of the document it belongs to and its topic proportion: $p(w_d | \theta_d, \beta) = \text{Mult}([\theta_d \beta])$. The links between two documents are modeled as Bernoulli binary variables, which are conditionally generated based on the latent topic proportions of these documents: $p(A_{d,d'} | \theta_d, \theta_{d'}) = sigmoid(f_y(\theta_d, \theta_{d'}))$, where f_y is the multi-layer perception.

Following the auto-encoding variational Bayes inference method (Kingma and Welling, 2014), we can yield the evidence lower bound (ELBO) to the marginal log-likelihood according to the above generative process:

$$\begin{aligned}\mathcal{L}(\Theta) = &- D_{KL}[q(\theta|w,\hat{A})||p(\theta|\alpha)] \\ &+ \mathbb{E}_{q(\theta|w,\hat{A})} logp(w|\theta,\beta) \\ &+ \mathbb{E}_{q(\theta|w,\hat{A})} logp(A|\theta)\end{aligned} \tag{5}$$

where Θ is the parameter set of the whole process, $q(\theta|w,\hat{A})$ is the approximate Dirichlet variational posterior as parameterized in Equation 3, $p(\theta|\alpha)$ is assumed the true Dirichlet posterior and α is the prior parameter. We still approximate it with its Laplacian approximation: the softmax variable on the multivariate normal with mean and covariance matrix as follows:

$$\begin{aligned}\mu_d^1 &= log\alpha - \frac{1}{k}\sum_i \alpha_i \\ \sigma_d^1 &= \frac{1}{\alpha}(1-\frac{2}{K}) + \frac{1}{k^2}\sum_i(\frac{1}{\alpha_i})\end{aligned} \tag{6}$$

We seek to minimize the KL divergence between the variational posterior and the true posterior in the first term. The second and third terms aim to reconstruct the document contents and links.

Based on the gradient variational Bayes (SGVB) estimator (Kingma and Welling, 2014), we can further yield the detailed formulation of each term:

$$\begin{aligned}D_{KL} = &\frac{1}{2}\{tr(\sigma^0(\sigma^1)^{-1}) \\ &+ (\mu^1-\mu^0)^T(\sigma^1)^{-1}(\mu^1-\mu^0) \\ &- k + log(\frac{|\sigma^1|}{|\sigma^0|})\}\end{aligned} \tag{7}$$

$$\begin{aligned}logp(w|\theta,\beta) = &\sum_{d=1}^{n} -X_d log(\tilde{X}_d) \\ &- (1-X_d)log(1-\tilde{X}_d) \\ logp(A|\theta) = &\sum_{d=1}^{n} -A_d log(\tilde{A}_d) \\ &- (1-A_d)log(1-\tilde{A}_d)\end{aligned} \tag{8}$$

where $\tilde{X} = (\theta\beta)$ is the reconstructed document contents, $\tilde{A} = sigmoid(f_y(\theta))$ is the reconstructed document links. Based on these, we can optimize the ELBO with stochastic gradient descent to infer the whole model end to end.

Table 1: Statistics of the document network datasets (Zhang and Lauw, 2020)

Datasets	Classes	Documents	Edges	Vocabulary
DS	9	570	1336	3,085
HA	6	223	515	2,073
ML	7	1,980	5,748	4,431
PL	9	1,553	4,851	4,105

4 Experiments

We conduct experiments in several real-world document network datasets. The statistics are reported in Table 1. Four datasets are subsets extracted from Cora: Data Structure (DS), Hardware and Architecture (HA), Machine Learning (ML), and Programming Language (PL) as in (Zhang and Lauw, 2020), in which Cora is the scientific article citation dataset collected from scholar websites. To evaluate the unsupervised representation learning capability of our method, we infer the latent topic portions of documents θ with our model, and then use it for three types of downstream tasks, namely document classification, document clustering, link prediction. We compare our method against baselines from the following three categories:

- **Auto-Encoders**: including variants of auto-encoders such as AE, DAE (Vincent et al., 2010), CAE (Rifai et al., 2011), VAE (Kingma and Welling, 2014), KSAE (Makhzani and Frey, 2014), KATE (Chen and Zaki, 2017) use Auto-Encoder to encode texts;
- **Relational Topic Models**: including generative models based on relational topic model such as RTM (Chang and Blei, 2009), PLANE (Le and Lauw, 2014), NRTM (Bai et al., 2018), ProdLDA (Srivastava and Sutton, 2017), and also a relational topic model based on the auto-encoder method: ADE (Zhang and Lauw, 2020);
- **Graph Embedding**: including graph embedding method based on GCN - VGAE (Kipf and Welling, 2016a).

We follow the settings for all baselines as in (Zhang and Lauw, 2020) and also compare methods in both transductive and inductive learning settings. For inductive learning, we randomly select a subset of 70% documents as the training set, a subset of 10% of documents as the validation set, and use the remaining 20% documents as the testing set. For transductive learning, all documents are involved in the training process. All experimental results are averaged over the results of 10 independent runs. Following (Zhang and Lauw, 2020), we set the topic number K as 64, the layer number L of message passing in HGAT as 1. The hidden size of weight matrices in HGAT is equal to the topic number of 64. For the log-normal prior in HGAT, we set the the parameter $\lambda = \sqrt{\pi}$, the variance $\sigma = \frac{1}{\sqrt{2}}$. We use the tanh activation function in HGAT. We use the Dirichlet distribution with parameter $\alpha = \frac{1}{K}$ for the logistic normal approximation. The learning rate on all datasets is 0.065, the maximum training epochs with Adam is 40000, the early stop epoch is 500. The parameter setting in all baseline models is the same as in (Zhang and Lauw, 2020).

We infer document topic representations with trained GRTM model for both train and test documents, which are then used in three downstream tasks to evaluate the effectiveness: 1) **Document classification**: we adopt K-Nearest Neighbors to predict each document's label based on the Euclidean distance of generated representations. We use the classification accuracy as the metric. 2) **Document clustering**: We also compare our method with baselines in clustering documents via K-means, to investigate whether our method can generate similar representations for documents in the same category. In this case, the ground truth labels are only utilized to calculate normalized mutual information (NMI) in evaluation. 3) **Link Prediction**: The generated representations are used to predict the links between documents in this experiment. We use Mean Average Precision (MAP) as the evaluation metric following the previous method (Zhang and Lauw, 2020). To better understand the semantic information our method captured in generated representations, we also conduct experiments to present a detailed analysis of our generated topics in inductive learning: 1) **Topic Coherence**: As in previous work (Zhang and Lauw, 2020), we adopt PMI - $PMI(w_i, w_j) = log\frac{p(w_i,w_j)}{p(w_i)p(w_j)}$ to evaluate topic coherence. We calculate the average pairwise PMI of the top 10 words in each topic. Better topics should produce higher PMI.

2) **Visualization**: We apply t-SNE to project text representations generated by different models into a 2-dimensional space.

4.1 Overall Results

As shown in Table 2 and 3, our method achieves the best performance in all tasks on the four datasets on both inductive and transductive settings. Compared with auto-encoder based methods (AE, DAE, CAE, VAE, KSAE, and KATE), which only consider document contents without document networks. ADE, relational topic model methods (ProdLDA, RTM, PLANE, NRTM), and our method achieve better performance due to considering the links among documents, which benefits the downstream tasks of document classification/clustering and link prediction. Compared with relational topic model methods (ProdLDA, RTM, PLANE, NRTM), we found that the graph embedding method VGAE and the adjacent auto-encoder method ADE perform better than other baselines, which demonstrates the advantage of using high-order proximity information. But they are still inferior to our proposed GRTM, which proves the benefits of fully exploring information of various orders in document networks. A similar performance among these methods can also be observed in the results of topic coherence in Table 4.

There is no mean standard deviation evaluation by the previous methods (Zhang and Lauw, 2020; Chen and Zaki, 2017), so we only report the results of our method in Table 5 to illustrate the statistical effectiveness of our model. These results are obtained in both transductive and inductive settings through repeating each 10 times.

4.2 Effect of Topic Number

To investigate the sensitivity of our method to topic numbers, we present the classification accuracy of our model on different topic numbers in the inductive setting. As shown in Figure 2, the test accuracy on four datasets generally improves with the increase of the number of topics and reaches the peak when the topic number is around 64. From these curves, we can find that the performance of our model is not too sensitive to the topic number, and also the topic number does not seem to be so related to the ground truth number of classes of datasets. In figure 3, we further show transductive test classification accuracy of different models under different topic numbers. We can see that our model consistently outperforms all baselines under different topic numbers on four datasets.

Table 2: Transductive learning results on document classification, document clustering, and link prediction.

Model	Document Classification				Document Clustering				Link Prediction			
	DS	HA	ML	PL	DS	HA	ML	PL	DS	HA	ML	PL
AE	0.558	0.688	0.739	0.616	0.250	0.315	0.368	0.230	0.144	0.195	0.107	0.102
DAE	0.656	0.799	0.790	0.694	0.372	0.409	0.441	0.278	0.204	0.296	0.121	0.147
CAE	0.558	0.685	0.741	0.620	0.261	0.309	0.371	0.228	0.145	0.188	0.108	0.103
VAE	0.652	0.789	0.796	0.679	0.356	0.394	0.447	0.286	0.193	0.283	0.122	0.135
KSAE	0.537	0.672	0.710	0.581	0.245	0.295	0.345	0.222	0.136	0.182	0.092	0.088
KATE	0.628	0.808	0.762	0.651	0.325	0.378	0.342	0.267	0.174	0.267	0.095	0.114
ProdLDA	0.637	0.780	0.764	0.631	0.374	0.460	0.423	0.289	0.162	0.324	0.080	0.095
RTM	0.543	0.637	0.663	0.574	0.082	0.094	0.126	0.127	0.117	0.194	0.072	0.075
PLANE	0.690	0.799	0.750	0.648	0.417	0.406	0.439	0.288	0.284	0.226	0.107	0.160
NRTM	0.591	0.816	0.549	0.503	0.313	0.404	0.137	0.190	0.149	0.221	0.036	0.049
VGAE	0.671	0.827	0.807	0.718	0.335	0.362	0.495	0.308	0.285	0.265	0.132	0.171
ADE	0.744	0.846	0.857	0.780	0.445	0.548	0.571	0.392	0.374	0.326	0.251	0.271
DGTAE	**0.753**	**0.860**	**0.869**	**0.792**	**0.501**	**0.562**	**0.592**	**0.416**	**0.402**	**0.340**	**0.270**	**0.294**

Table 3: Inductive learning results on document classification, document clustering, and link prediction.

Model	Document Classification				Document Clustering				Link Prediction			
	DS	HA	ML	PL	DS	HA	ML	PL	DS	HA	ML	PL
AE	0.405	0.580	0.632	0.509	0.213	0.337	0.340	0.248	0.185	0.233	0.181	0.129
DAE	0.516	0.749	0.732	0.595	0.375	0.436	0.415	0.299	0.347	0.286	0.259	0.198
CAE	0.400	0.573	0.644	0.519	0.212	0.279	0.362	0.253	0.192	0.232	0.185	0.132
VAE	0.491	0.785	0.738	0.594	0.373	0.361	0.404	0.300	0.391	0.346	0.243	0.192
KSAE	0.390	0.569	0.614	0.491	0.269	0.319	0.334	0.232	0.188	0.238	0.148	0.111
KATE	0.484	0.800	0.712	0.573	0.321	0.440	0.354	0.290	0.277	0.336	0.205	0.178
ProdLDA	0.202	0.401	0.184	0.158	0.302	0.292	0.399	0.306	0.220	0.297	0.192	0.140
RTM	0.327	0.498	0.652	0.564	0.000	0.046	0.091	0.048	0.260	0.276	0.210	0.149
PLANE	0.282	0.544	0.275	0.218	0.162	0.192	0.000	0.000	0.306	0.345	0.176	0.134
NRTM	0.456	0.811	0.482	0.408	0.339	0.398	0.167	0.207	0.076	0.097	0.020	0.049
VGAE	0.509	0.748	0.736	0.607	0.280	0.185	0.442	0.291	0.315	0.309	0.237	0.274
ADE	0.640	0.845	0.836	0.724	0.416	0.489	0.522	0.363	0.400	0.427	0.363	0.322
GRTM	**0.687**	**0.867**	**0.841**	**0.731**	**0.466**	**0.564**	**0.535**	**0.394**	**0.432**	**0.449**	**0.538**	**0.358**

Table 4: Topic Coherence

Model	PMI			
	DS	HA	ML	PL
AE	0.294	0.446	0.665	0.969
DAE	1.170	1.125	1.203	1.553
CAE	0.348	0.558	0.526	0.684
VAE	0.685	0.793	1.831	1.132
KSAE	0.547	0.285	0.770	0.759
KATE	1.312	1.755	1.619	2.003
ProdLDA	1.638	1.315	1.837	2.088
RTM	1.279	1.678	1.199	1.615
PLANE	1.585	1.847	1.756	2.099
NRTM	1.533	2.041	1.328	1.632
ADE	1.872	1.887	2.337	2.321
GRTM	**2.073**	**2.361**	**2.512**	**2.610**

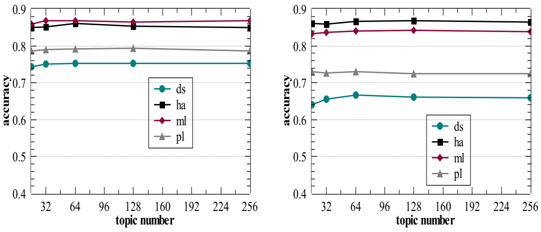

(a) Transductive (b) Inductive

Figure 2: The test classification accuracy of our model vary different topic numbers

4.3 Effect of Log-normal Prior

We vary the variance σ of the log-normal prior to explore the impact of the log-normal prior in HGAT. We present results of our model under the value of $\sigma \in \{\frac{1}{2}, \frac{1}{\sqrt{2}}, 1\}$, and the results are shown in Table 7. The larger value of σ makes the slower decay of the importance on higher-order proximity information otherwise the faster decay. From the table, we can see that our model generally achieves the best performance at $\sigma = \frac{1}{\sqrt{2}}$. We suspect that too much noisy higher-order information is introduced when σ is set too large, while insufficient higher-order information can be used when too small. Hence it yields poor performance in both cases.

4.4 Different layer numbers

Although one layer message passing process of HGAT is able to capture arbitrary-order proximity information among document networks, we can still report the results of our model under the different layers of HGAT in Table 6. We can see that our model with one layer HGAT achieves the best performance under both settings. When the layer number of HGAT is set to 0, the HGAT encoder of our model degenerates to the feed-forward neural network. With two-layer HGAT, document representations may be disturbed by the information of their noisy neighbors.

Table 5: Mean ± standard deviation results of our model on document classification, document clustering.

Model	Document Classification				Document Clustering			
	DS	HA	ML	PL	DS	HA	ML	PL
Trans	0.753 ± 0.021	0.86 ± 0.032	0.869 ± 0.016	0.792 ± 0.030	0.501 ± 0.009	0.562 ± 0.011	0.592 ± 0.0014	0.416 ± 0.008
Induc	0.687 ± 0.018	0.867 ± 0.024	0.841 ± 0.020	0.731 ± 0.021	0.466 ± 0.011	0.564 ± 0.012	0.535 ± 0.015	0.394 ± 0.009

Table 6: Transductive results of our model on document classification, document clustering, and link prediction under different layer numbers of HGAT.

Layer Number	Document Classification				Document Clustering				Link Prediction			
	DS	HA	ML	PL	DS	HA	ML	PL	DS	HA	ML	PL
0	0.731	0.852	0.850	0.784	0.486	0.553	0.572	0.401	0.382	0.324	0.247	0.269
1	**0.753**	**0.860**	**0.869**	**0.792**	**0.501**	**0.562**	**0.592**	**0.416**	**0.402**	**0.340**	**0.270**	**0.294**
2	0.742	0.849	0.855	0.780	0.488	0.546	0.568	0.406	0.390	0.327	0.255	0.271

4.5 Ablation Study

We also perform an ablation study on our method to verify the effectiveness of each module in the inductive setting. We compare our model with its variants by removing one of the components HDPN and HGAT respectively, as shown in Table 8. From which we can see that each component makes a certain contribution to the overall performance. In the case of removing the HDPN (W/HDPN) module, our model directly takes the document adjacency matrix as input, in which it degenerates into the relational topic model based on graph attention auto encoder without considering the higher-order proximity. As in the case of removing the HGAT (W/HGAT), although our model takes the higher-order information into consideration, it doesn't make the important selection for different order correlation information. We can also see that missing the higher-order proximity has a more significant negative influence than missing the HGAT based encoder module, illustrating the relative effectiveness of the higher-order information in improving the discrimination of latent representations.

4.6 Visualization

Finally, to intuitively demonstrate the effectiveness of our model, we visualize the learned representations of the test documents on the ML dataset in Figure 4. It shows that documents are better grouped by our model than ADE (with first-order correlations) and VGAE (with few-order correlations), due to the incorporation of the in-direct correlation information among documents.

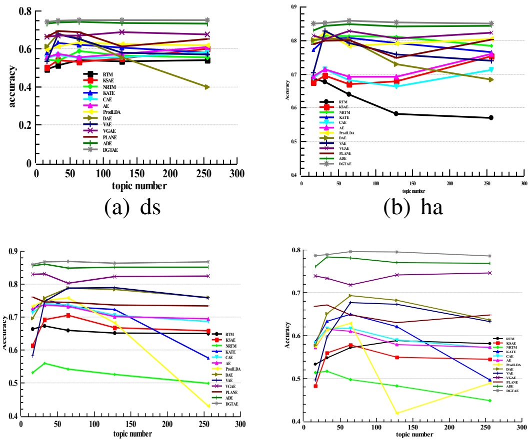

(a) ds (b) ha

(c) ml (d) pl

Figure 3: The test classification accuracy of different models vary different topic numbers

Table 7: Results of our model based on vary values of σ on document classification, document clustering, and link prediction.

σ	Document Classification		Document Clustering		Link Prediction	
	DS	HA	DS	HA	DS	HA
1	0.739	0.861	**0.52**	0.556	0.389	0.332
$\frac{1}{\sqrt{2}}$	**0.753**	**0.861**	0.501	0.562	**0.402**	**0.340**
$\frac{1}{2}$	0.736	0.85	0.498	**0.599**	0.396	0.338

Table 8: Ablation Study

Module	document classification				document clustering			
	DS	HA	ML	PL	DS	HA	ML	PL
All	**0.687**	**0.867**	**0.841**	**0.731**	**0.466**	**0.564**	**0.535**	**0.394**
W/HDPN	0.631	0.833	0.814	0.705	0.427	0.533	0.510	0.366
W/HGAT	0.651	0.852	0.827	0.710	0.450	0.557	0.524	0.390

5 Conclusion

In this paper, we propose a novel graph relational topic model GTM for document networks to fully explore each order of relations among document networks, which is efficiently fused by a proposed novel graph attention network HGAT equipped with log-normal attention prior. Experimental results show that full consideration of each order proximity information on the document-document graph is beneficial for improving the learned document representations. In future work, we would like to explore the better-suited method and more elegant prior distributions for discovering and fus-

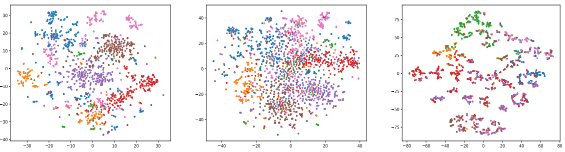

(a) DGTAE (b) ADE (c) VGAE

Figure 4: The t-SNE visualization of document representation learned by different models in ML under the inductive setting. (Each color denotes one categorical label of documents)

ing higher-order proximity in document networks.

Acknowledgments

We thank Benyou Wang for his valuable comments on this manuscript. We also would like to thank the anonymous reviewers for their constructive comments. We thank MindSpore for the partial support of this work, which is a new deep learning computing framework[2]. This research is supported by the CSC Scholarship offered by China Scholarship Council and the joint laboratory on Credit Technology.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, H. Harutyunyan, Nazanin Alipourfard, Kristina Lerman, G. V. Steeg, and A. Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*.

Ayan Acharya, Dean Teffer, Jette Henderson, Marcus Tyler, Mingyuan Zhou, and Joydeep Ghosh. 2015. Gamma process poisson factorization for joint modeling of network and documents. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 283–299. Springer.

Haoli Bai, Zhuangbin Chen, Michael R. Lyu, Irwin King, and Zenglin Xu. 2018. Neural relational topic models for scientific article analysis. *CIKM*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Jonathan Chang and David M. Blei. 2009. Relational topic models for document networks. In *AISTATS*.

Ning Chen, Jun Zhu, Fei Xia, and Bo Zhang. 2014. Discriminative relational topic models. *IEEE transactions on pattern analysis and machine intelligence*, 37(5):973–986.

Yu Chen and Mohammed J. Zaki. 2017. Kate: K-competitive autoencoder for text. *SIGKDD*.

Maosheng Guo, Yu Zhang, and Ting Liu. 2019. Gaussian transformer: a lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6489–6496.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*.

Thomas Kipf and Max Welling. 2016a. Variational graph auto-encoders. *NIPS workshop*, abs/1611.07308.

Thomas N Kipf and Max Welling. 2016b. Semi-supervised classification with graph convolutional networks. *ICLR*.

Tuan M. V. Le and Hady Wirawan Lauw. 2014. Probabilistic latent document network embedding. *2014 ICDM*, pages 270–279.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606*.

S. Liu, Lingwei Chen, Hanze Dong, Zihao Wang, D. Wu, and Zengfeng Huang. 2019. Higher-order weighted graph convolutional networks. *ArXiv*, abs/1911.04129.

Alireza Makhzani and Brendan J. Frey. 2014. k-sparse autoencoders. *CoRR*, abs/1312.5663.

Ramesh M Nallapati, Amr Ahmed, Eric P Xing, and William W Cohen. 2008. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *ICLR*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408.

Chaojie Wang, Hao Zhang, Bo Chen, Dongsheng Wang, Zhengjue Wang, and Mingyuan Zhou. 2020a. Deep relational topic modeling via graph poisson gamma belief network. *Advances in Neural Information Processing Systems*, 33.

Hao Wang, Xingjian Shi, and Dit-Yan Yeung. 2017. Relational deep learning: A deep latent variable model for link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

---

[2] https://www.mindspore.cn/

Zhengjue Wang, Chaojie Wang, Hao Zhang, Zhibin Duan, Mingyuan Zhou, and Bo Chen. 2020b. Learning dynamic hierarchical topic graph with graph convolutional network for document classification. In *International Conference on Artificial Intelligence and Statistics*, pages 3959–3969. PMLR.

Qianqian Xie, Jimin Huang, Pan Du, Min Peng, and Jian-Yun Nie. 2021a. Graph topic neural network for document representation. In *Proceedings of The Web Conference 2021*.

Qianqian Xie, Jimin Huang, Pan Du, Min Peng, and Jian-Yun Nie. 2021b. Inductive topic variational graph auto-encoder for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4218–4227, Online. Association for Computational Linguistics.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456.

Liang Yang, Fan Wu, Junhua Gu, Chuan Wang, Xiaochun Cao, Di Jin, and Yuanfang Guo. 2020. Graph attention topic modeling network. In *Proceedings of The Web Conference 2020*, pages 144–154.

Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. 2016. A discriminative topic model using document network structure. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 686–696, Berlin, Germany. Association for Computational Linguistics.

Ce Zhang and Hady W. Lauw. 2020. Topic modeling on document networks with adjacent-encoder. In *AAAI*.

Hengrui Zhang and Julian McAuley. 2020. Stacked mixed-order graph convolutional networks for collaborative filtering. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 73–81. SIAM.

Luoyi Zhang and Ming Xu. 2020. Epine: Enhanced proximity information network embedding. *arXiv preprint arXiv:2003.02689*.

Qile Zhu, Zheng Feng, and Xiaolin Li. 2018. Graphbtm: Graph enhanced autoencoded variational inference for biterm topic model. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 4663–4672.

Paths to Relation Extraction through Semantic Structures

Jonathan Yellin **Omri Abend**
School of Computer Science and Engineering
The Hebrew University of Jerusalem
{jonathan.yellin,omri.abend}@mail.huji.ac.il

Abstract

Syntactic and semantic structure directly reflect relations expressed by the text at hand and are thus very useful for the relation extraction (RE) task. Their symbolic nature allows increased interpretability for end-users and developers, which is particularly appealing in RE. Although they have been somewhat overshadowed recently by the use of end-to-end neural network models and contextualized word embeddings, we show that they may be leveraged as input for neural networks to positive effect. We present two methods for integrating broad-coverage semantic structure (specifically, UCCA) into supervised neural RE models, demonstrating benefits over the use of exclusively syntactic integrations. The first method involves reduction of UCCA into a bilexical structure, while the second leverages a novel technique for encoding semantic DAG structures. Our approach is general and can be used for integrating a wide range of graph-based semantic structures.[1]

1 Introduction

Early work on RE focused on pattern-based rules for capturing the structure of relation-evoking words and phrases. These rules are applied over text to identify entity relations in much the same way a regular expression would be applied to discover matching text. The pattern machinery spans from simple, regular-expression like, surface patterns (Brin, 1999; Agichtein and Gravano, 2000), through systems that integrate both lexical features and syntactic dependencies into the pattern construct (Mintz et al., 2009). The PredPatt and pyBART frameworks (Zhang et al., 2017a; Tiktinsky et al., 2020) are examples of syntactic dependency based systems that leverage a set of rules defined over Universal Dependencies (UD; Nivre et al., 2016) to extract predicate-argument structures.

In supervised RE, a multi-class classifier is trained to determine whether a relation between entities is evoked by a text. With the increased predominance of end-to-end neural network architectures in NLP practice, it is not surprising that recent work on supervised relation extraction has focused on adapting end-to-end neural systems for the task (Peters et al., 2019; Yamada et al., 2020). However, end-to-end neural models pose interpretability and customization challenges (Conneau et al., 2018; Belinkov et al., 2020), motivating the study of hybrid models, in which the neural architecture is fed explicit structure representations. The Contextualized Graph Convolution Network (C-GCN; Zhang et al., 2018) represents a method for exposing structure representation to the machinery of a deep neural network. C-GCN uses a sentence's UD structure as explicit input to the neural network, resulting in a model whose results are both competitive and interpretable.

In this paper we explore whether we can adopt broad coverage semantic structure to the same effect, and whether we are able to observe improved performance in comparison to the baseline model, based on syntactic structure. We focus on the Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013) framework as a test case, but note that our method can be easily extended to other semantic representations. Our results demonstrate that broad coverage semantic structures, including those that, like UCCA, require representation by directed acyclic graphs (DAGs), can be integrated effectively in neural networks for relation extraction.

---

[1] Code can be found on GitHub at https://github.com/yyellin/gcn-over-semantic-representations.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2614–2626
August 1–6, 2021.

2 Background

We introduce the representation approaches and datasets we employ in our study.

2.1 Universal Dependencies

Since the late 1990s, various NLP motivated dependency grammar representations have been proposed. Carroll et al. (1999); King et al. (2003); De Marneffe and Manning (2008) are examples of such systems, each one accompanied with corpora of accordingly annotated sentences, used for training supervised models for dependency parsing. The UD (Nivre et al., 2016) project, based on De Marneffe and Manning (2008), is a more recent dependency grammar representation, that emphasizes cross-linguistic consistency and has over 300 contributors producing more than 150 treebanks in 90 languages.[2] UD dependency grammar representations come in three forms: *basic*, *enhanced* and *enhanced++* (Schuster and Manning, 2016); in *basic* the sentence is represented as a tree structure, with every word in the sentence assigned a single head word, while *enhanced* and *enhanced++* are DAG representations, in which a word may have multiple heads, and where otherwise implicit relations between content words are captured explicitly. In this paper, we use UD v1 to maintain consistency with the setup used by Zhang et al. (2018).

2.2 UCCA

The Universal Conceptual Cognitive Annotation framework (UCCA; Abend and Rappoport, 2013) is a multi-layered system for semantic representation that seeks to capture the semantic, rather than syntactic patterns, expressed through linguistic utterances. The UCCA scheme maps sentences to DAGs that embody these semantic structures. In contrast to graphs formed by dependency grammars, whose nodes all represent lexical entities, a UCCA graph contains both nodes that represent word terminals, which are leaves in the DAG, and non-leaf nodes that represent entities according to some semantic consideration. The foundational layer of UCCA covers the semantics of predicate-argument structure evoked by predicates of all grammatical categories (verbal, nominal, adjectival and others). The layer's primary construct is a *Scene*, which captures a temporally persistent state or an evolving event. A Scene contains one or more Participants, and may also contain secondary scenes, known as Adverbials. Scene, Participant and Adverbial manifest as units in the DAG. In appendix A we provide an example to highlight UCCA's semantic dexterity.

TUPA (Transition-based UCCA Parser) is a transition-based parsing model that can be trained to map sentences to their UCCA scene-based foundational layer (Hershcovich et al., 2017). Pre-trained TUPA models are available online;[3] the BERT based pre-trained model is used extensively in this work.

2.3 TACRED

TACRED was designed by Zhang et al. (2017b) to address the dearth of annotated data required for supervised learning for RE. It is based on examples from the corpus used in the yearly TAC Knowledge Base Population (TAC KBP) challenges, conducted from 2009 to 2015 (McNamee and Dang, 2009). In each annual challenge, 100 entities, people, and organizations, were provided to competing systems for them to identify relations between those given entities (referred to as *subjects*), and other *objects* mentioned in the text. The TAC KBP relation extraction task was formulated in terms of slot filling: a person entity is assigned 26 attribute types while an organization is assigned 16 — the challenge posed to competing systems being the extraction of the values of these attributes based on the given corpus (or in other words, the object and relation are given; the challenge is to find the subject). TACRED leverages the results of these challenges to form a set of 106,264 example sentences, each one containing an object and a subject.

Annotation is implemented through crowdsourcing, with each sentence annotation containing the spans of both object and subject, one of the 42 entity relation attributes, or a *no_relation* classification if no relation exists. In addition to the human created annotations for subject and object spans and relation type, the TACRED dataset contains POS tags, named entities and UD parses by the Standford CORENLP parser (Manning et al., 2014). TACRED's dependency representation annotation corresponds to *basic* UD v1.

3 Motivating Experiment: A Rule-based Approach

Syntactic sentence representation is used in both rule-based and supervised methods for relation ex-

---

[2] `https://universaldependencies.org`

[3] `https://github.com/danielhers/tupa`

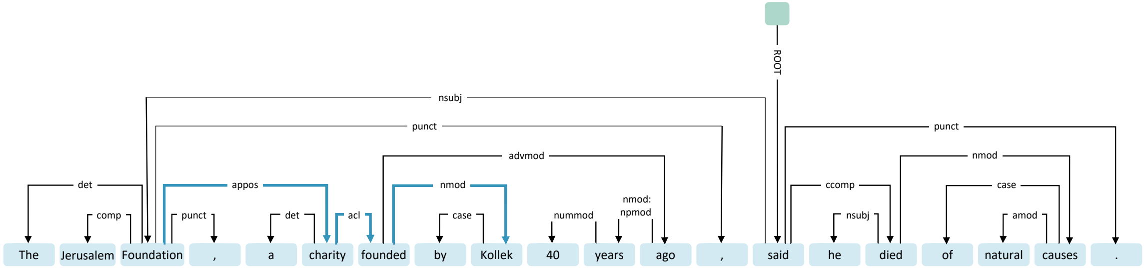

(a) UD Parse Graph

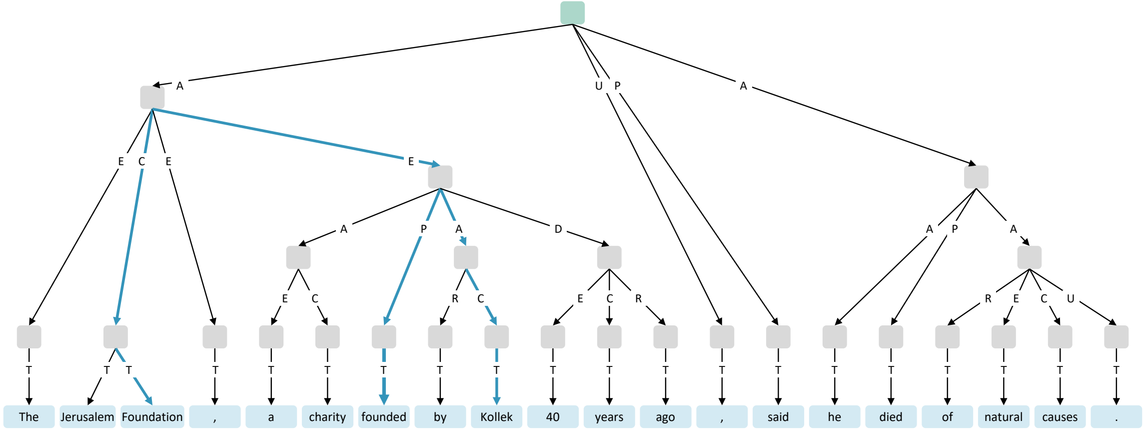

(b) UCCA Parse Graph

Figure 1: UD v1 (basic) and UCCA parse graphs for the sentence 'The Jerusalem Foundation, a charity founded by Kollek 40 years ago, said he died of natural causes.' The diagrams highlight paths from the subject 'Jerusalem Foundation, through the trigger word 'founded' to the object 'Kollek' (sentence 61b3a5c8c9272ce895a6 in the TACRED training dataset). For UD the resulting path pattern is **`<sub>↓appos↓acl<trigger>↓nmod<obj>`**, and for UCCA it is**`<sub>↑C↓E↓P<trigger>↑P↓A↓C<obj>`**.

traction. Our hypothesis is that semantic sentence representation can provide further benefit for this task. We begin the examination of this conjecture by comparing the relation extraction performance of two simple rule-based methods, one using UD and the other using UCCA, both applied to sentences from the TACRED dataset.

Method. Asserting that a relation between two entities is evoked by a text is often contingent on the presence of a "trigger word", or phrase, with semantic application to both entities. We use pattern-based rules that denote a path from the relation subject to the trigger word, and from the trigger word to the relation object. Figure 1 illustrates this by highlighting the path for both UD and UCCA for the sentence "The Jerusalem Foundation, a charity founded by Kollek 40 years ago, said he died of natural causes".

Our procedure comprises three phases: UCCA tagging, pattern extraction for both UD and UCCA, and pattern evaluation.

UCCA Tagging. While the TACRED dataset contains the UD representation for each sentence, it does not contain the corresponding UCCA representation. To begin our experiment, we use TUPA with the BERT based pre-trained model, to create UCCA annotations for all sentences in the training set, producing TACRED sentences that have both UD and UCCA representations. We do the same for the sentences in the test set, which we use for evaluation.

Pattern Extraction. Pattern extraction requires a manual step in which trigger words for the relation in each sentence are identified. With the trigger words identified, automatic pattern construction follows for both UCCA and UD, per relation type. The caption of Figure 1 presents an example. The end result is a list of paths and trigger words for each relation.

We declare a path match to occur when the target sentence contains a subject and object entity pair, any one of the trigger words in our list for the relation, and a path from the subject entity to the trigger word and from it to the object entity, that matches one of our extracted patterns.

Evaluation. We evaluate our system using the *org:founded_by* relation. For precision we count the number of no_relation sentences in the test set incorrectly identified as containing an org:founded_by pair, as a fraction of the total number of no_relation and org:founded_by sentences, to produce a precision value. We ignore sentences containing other relations. For recall, we count the number of correctly identified org:founded_by relations, from the total number in the test set.

Results. For org:founded_by, recall is 0.38 for the UD patterns and 0.2 for UCCA. Interestingly, when applying either UD or UCCA patterns, recall improves to 0.47. The precision result is 0.99 for both UD and UCCA. We note that since the UCCA representation system contains non-terminal nodes, we can expect more variation in a UCCA path than in the terminal-only UD representation-based path, contributing to the inferior recall score of UCCA versus UD.

While our experiment considers entity pairs with an org:founded_by relation or no relation at all, and thus represents a binary classification problem, rather than the multi label classification problem posed by the 41 relations of the TACRED ontology, it nonetheless highlights the potential for leveraging trigger words and sentence graph representations (syntactic as in UD v1 *basic* and semantic as in UCCA) for construction of relation matching systems. Indeed, comprehensive trigger word selection per relation is not scalable if done manually, however we could utilize word embedding techniques, so that trigger word matching could be performed using a vector distance threshold, which would capture similar trigger word terms (Batista et al., 2015).

In the next sections, we describe more sophisticated machinery, which realizes a softer notion of matching between the paths of the training data, and those of the test set.

4 C-GCN for Semantic Representation

We propose a supervised deep-neural-network model that explicitly utilizes sentence graph representations, so that we may compare the utility of UCCA and UD paths (and their combination). The model receives a sentence and two entity spans (subject and object) as input and gives preference to the word representations corresponding to the syntactic or semantic path between subject and object, for processing by the deep neural network. Contextualized Graph Convolution Networks (Zhang et al., 2018) fulfil this requirement, and we select them as the framework for our study.

4.1 The C-GCN Model

At the heart of the graph convolution network method presented by Zhang et al. is the notion that a sentence's bi-lexical dependency structure can be captured as an adjacency matrix and used efficiently to fuse the representation of each token on the path with the representations of its dependency induced neighbors. Zhang et al. experimented with various implementations and hyperparameters; we reference the C-GCN implementation for which their published results are achieved, in which only words on the path or one dependency edge away from the path are considered ("distance-one-pruning"), and which uses two convolution layers.[4]

In the case of UD, where all edges in the dependency graph representation are between terminals, this adjacency matrix stems from the graph representation directly (this is not the case with UCCA that contains non-terminal nodes; see 4.2). The adjacency matrix captures the dependencies between words that are either on the shortest path from subject to object, or are one dependency edge away from a word on the shortest path ("near-path" tokens); all other dependencies are ignored.

C-GCN Architecture. Figure 2 presents the six building blocks of the C-GCN implementation. The nexus of the C-GCN model is in the graph convolution layer (Block 3): it allows the deep neural network apparatus to focus on tokens that are on-path or near-path. Best results are reached by Zhang et al. with two instances of graph convolution, as depicted in the diagram. The first GCN instance receives the $[\texttt{sentence-length} \times 400]$ matrix input from the LSTM and outputs a $[\texttt{sentence-length} \times 200]$ matrix. Subsequent GCN instances all accept as input, and produce as output, $[\texttt{sentence-length} \times 400]$ matrices.

To explain the core GCN mechanism we employ some notation. Let A represent our pruned

---

[4] This is also the model implemented by Zhang et al. and available on GitHub at `https://github.com/qipeng/gcn-over-pruned-trees`.

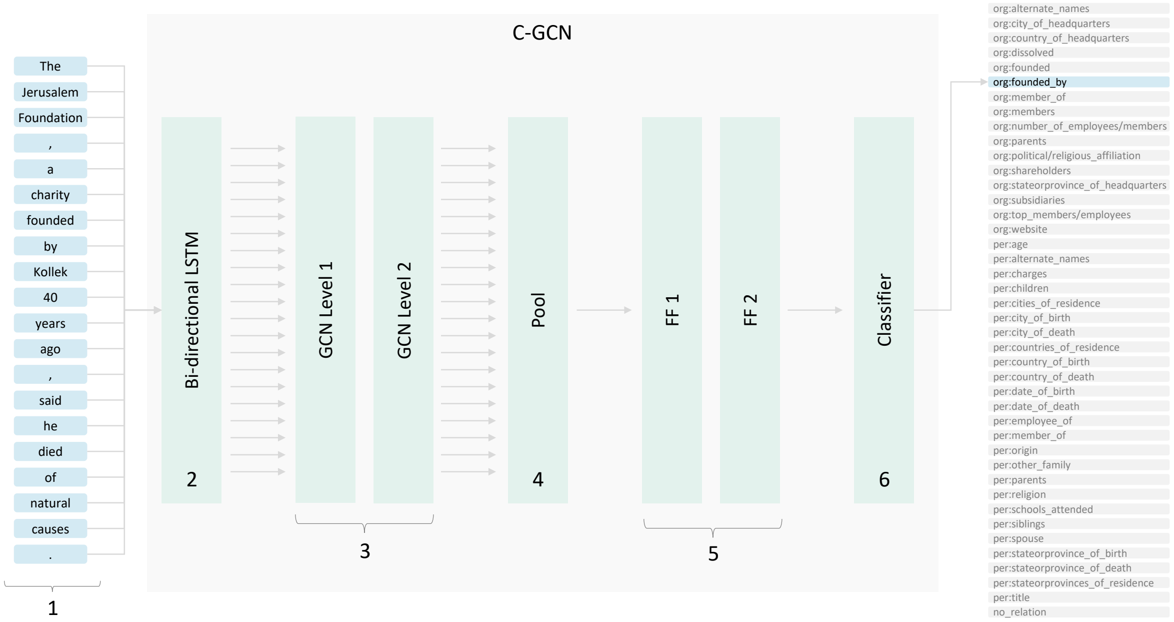

Figure 2: High level overview of the six blocks comprising the contextualized graph neural network, with the sentence from figure 1 as input.

adjacency matrix, where $A_{i,j} = 1$ if both i and j are on the path between subject and object, or one edge away from it, and there exists a dependency between tokens i and j of any kind. The direction of the dependency is ignored by the model, resulting in a symmetric matrix. Each GCN instance is assigned its own weight matrix; the first, of dimensions $[400, 200]$, all others of dimensions $[200, 200]$. Let $\mathrm{W}^{(l)}$ represent the weight matrix corresponding to instance l. Let d_i denote the degree of token i, where a token's degree corresponds to the number of tokens it is connected to directly, according to the dependency graph representation. Finally, let $\mathrm{h}^{(l)}$ and $\mathrm{h}^{(l+1)}$ represent the input to GCN instance l and its output respectively. Then we have:

$$h_i^{(l+1)}{=}\sigma\left(\frac{\sum_{j=1}^{n}(\mathrm{A}{+}\mathrm{I})_{i,j}\mathrm{W}^{(l+1)}h_j^{(l)}}{d_i}{+}b^{(l+1)}\right) \tag{1}$$

In other words, $\mathrm{h}^{(l+1)}$ is computed by applying a non-linearity function to the a linear combination of both its immediate, unpruned, neighbors in the graph, and itself, normalized by the degree of token i. Intuitively, all on, or near-path, tokens are fused with their on or near-path neighbors in the dependency graph, up to a dependency distance that corresponds to the number of GCN instances.

We describe the other network blocks in appendix C.

GCN vs C-GCN In their paper, Zhang et al. analyze two variants of the graph convolution network: C-GCN, and GCN. These models are identical except with regard to an LSTM block, which is only used in C-GCN. The 'C' of C-GCN refers to the contextualization achieved by the LSTM apparatus.

Modification of the C-GCN model to support experimentation with other dependency representations involves multiple stages. In the following sections we review these modifications and the resulting models.

4.2 Converting UCCA Structures to Bi-Lexical Graph

A goal of this study is to apply UCCA's semantic representations to the task of RE, by leveraging UCCA's parse graph in the graph convolutional network machinery of the C-GCN model. As in our motivating experiment (section 3), the preliminary step is to produce UCCA representations for each of TACRED's 106,264 sentences. We again employ TUPA's BERT-based pre-trained model to this end, and generate parse-graph representations for each sentence. We are confronted however with a challenge: C-GCN expects a *tree* structure, where all nodes correspond to tokens in the sentence and have a single parent; UCCA produces a *graph* structure with nodes that correspond to non-terminal units and that have multiple parents.

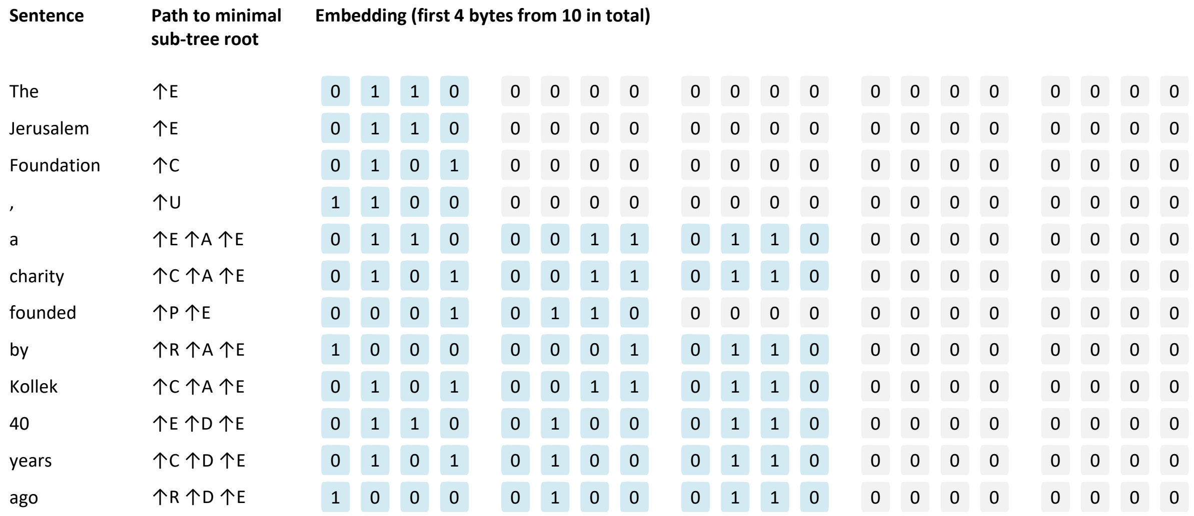

Sentence	Path to minimal sub-tree root	Embedding (first 4 bytes from 10 in total)																			
The	↑E	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Jerusalem	↑E	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Foundation	↑C	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
,	↑U	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	↑E ↑A ↑E	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
charity	↑C ↑A ↑E	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
founded	↑P ↑E	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
by	↑R ↑A ↑E	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
Kollek	↑C ↑A ↑E	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
40	↑E ↑D ↑E	0	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
years	↑C ↑D ↑E	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
ago	↑R ↑D ↑E	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 3: Representation of the first 4 bytes in the initial UCCA embedding (prior to tuning that occurs during training), for the sentence ”The Jerusalem Foundation, a charity founded by Kollek 40 years ago, said he died of natural causes”. The first 12 terminals in this sentence are contained within the DAG comprised of the lowest common ancestor of both the *subject* (The Jerusalem Foundation) and *object* (Kollek), and the first 20 dimensions of the UCCA embeddings for these are depicted in this illustration. The embedding for all other terminals will be the a $\vec{0}$ vector.

The first step in bridging this gap involves conversion of the UCCA graph into a bilexical structure, where all nodes correspond to a token in the sentence. We employ the method described by Hershcovich et al. (2017), which heuristically selects a head terminal for each non-terminal node and attaches all terminal descendants to the head terminal.[5] Using this conversion procedure, we produce a UCCA-based bi-lexical graph representation for each sentence in the TACRED dataset. Appendix B provides an example of a bi-lexical reduction.

In the bi-lexical representation result, a terminal token may have multiple parents, deeming the structure a non-tree DAG. Indeed, the notion of multiple parents may be viewed as a core feature of semantic representations, reflecting the fact that a word in a sentence or text may have multiple semantic roles (Oepen et al., 2020). We thus modify the C-GCN model implementation, so that it can produce the adjacency matrix using a bi-lexical DAG rather than tree.

4.3 Extending C-GCN for UCCA DAG Representation

Conversion of a UCCA sentence representation into a bi-lexical graph clearly loses extensive semantic information captured in the original structure. We pursue an embedding-based mechanism that will allow us to exploit the full UCCA structure representation in our model.

Our novel approach seeks to utilize a lossless representation of the UCCA DAG structure as training input for an **additional embedding vector**. To this end, we map each token in a sentence to its path to the root of the sentence-representing UCCA DAG. With a set of fourteen different edge labels in UCCA's foundational level, we can encode each label with four 0/1 bits. Given a maximum distance of 18 steps from terminal to root for all sentences in the TACRED dataset, we can encode all paths with up to 72 zero/one bits in total. In a new pre-training step, we collect all distinct paths from terminal to root as derived from TACRED training dataset (for a total of 22,922 distinct paths), assigning each one to an initial 80-dimensional vector, where the value of dimension i corresponds to bit i in the corresponding path (we assign 0 to all beyond-path dimensions).[6] We add a $\vec{0}$ vector for representing out-of-vocabulary paths (necessary for evaluation). Finally, we extend the 360-dimensional representation of each token to 440 dimensions, by concatenating the 80-dimensional representation of the

---

[5] Details of the algorithm in Hershcovich et al. supplementary material `https://www.aclweb.org/anthology/attachments/P17-1104.Notes.pdf`.

[6] This preparatory procedure is somewhat similar to the pre-training step required for word embeddings, in which we collect the set of word embeddings that our training data vocabulary corresponds with for training initialization of word embeddings.

token's path to root. Like word embeddings, the UCCA path-to-root representation embeddings are tuned during training.

While we can assign a UCCA-embedding for each token in the sentence, we would like to focus our network's attention on tokens that relate closely to *subject* and *object*. We achieve this with the help of a ***minimal sub-DAG***, which we define as the DAG comprised of the lowest common ancestor of both *subject* and *object*, and all their descendants. Where there is more than a single lowest common ancestor, we choose the one that induces the smallest number of terminals. All tokens that are outside the *minimal sub-DAG* are assigned the $\vec{0}$ embedding. In figure 3 we provide an example of the embedding vector values before they are fine-tuned during training.

5 Experiments

We perform twenty independent cycles of training-and-test for all our experiments, computing an average of the recall, precision and F$_1$ results across these runs. This technique differs from the common evaluation protocol of selecting the model with the median dev F$_1$ from five independent runs and reporting its F$_1$ result; we frame our approach in terms of statistical significance in Appendix F. We use TACRED's standard train/dev/test splits. Results are reported using the standard F$_1$ score. We use a fork of the source code made available by Zhang et al.[7] In general we maintain the default parameters of Zhang et al., including the default pruning factor of one, by which tokens that are up to one edge away from the path are also considered by the convolution and pruning mechanisms.

As noted, C-GCN is trained on the TACRED dataset.[8] For each of the 106,264 sentences, the dataset contains attributes including the relation type, subject and object span and UD dependency information. The process of model extension begins with the enrichment of the TACRED dataset with representation for additional semantic structures. As additional representation needs to be expressed in terms of the attributes of **tokens** for the enrichment to be valid, alignment on **tokenization** must be reached. In appendix D we describe how this alignment is achieved and reproduce Zhang et al.'s results using our updated tokenization.

System	**Adjacency Combination**			**F$_1$**
	cons	**UD**	**UCCA**	
C-GCN$_{ud}$	✗	✓	✗	66.27
C-GCN$_{ucca}$	✗	✗	✓	66.44
C-GCN$_{ud \cup ucca}$	✗	✓	✓	66.67
C-GCN$_{all}$	✓	✓	✓	66.94

Table 1: Results for different adjacency matrix combinations. C-GCN$_{ud}$ is the baseline model described in Zhang et al. (2018). C-GCN$_{ucca}$ replaces the UD representation with bi-lexical UCCA representation. Best performance is achieved when supplementing the combined UD and UCCA path representations with all tokens between subject and object (C-GCN$_{all}$).

System	**F$_1$**
C-GCN$_{ud+emb}$	66.66
C-GCN$_{ucca+emb}$	66.60
C-GCN$_{all+emb}$	67.32
C-GCN$_{all+emb}$ ensemble	**68.41**

Table 2: The models listed in table 1 are further enriched with UCCA DAG representation, and their resulting scores are listed here. We demonstrate an increase of one F$_1$ point for C-GCN$_{all+emb}$ over baseline model C-GCN$_{ud}$.

The graph-convolution network uses each sentence's UD dependency tree to determine the set of tokens (and their intra-connections) that it should focus on in the convolution and pooling phases of the network operation. As we presented in section 4.1, this is achieved by representing the UD dependency tree as adjacency matrix A_{ud}. The bi-lexical UCCA graph projection we described in section 4.2 is represented by A_{ucca}.

In addition to UD and UCCA structures, we employ a synthetic bi-lexical tree, where, for an n-token sentence, the only dependencies are between *token$_i$* and *token$_{i+1}$* for all $i < n$. This artificial dependency tree will cause the convolution network to focus on all tokens that appear between *subject* and *object* in the surface structure of the sentence (including *subject* and *object* themselves), and, in consideration of path pruning, two additional tokens adjacent to either ends of the span. We denote the adjacency matrix for this artificial structure by A_{cons} ("cons" for consecutive).

We establish a performance baseline by testing different combinations of these adjacency matrices: A_{ud} alone, A_{ucca} alone, $[\mathrm{A}_{ud} \cup \mathrm{A}_{ucca}]$, and finally $[\mathrm{A}_{ud} \cup \mathrm{A}_{ucca} \cup \mathrm{A}_{cons}]$. Our results are presented in table 1.

---

[7] `https://github.com/yuhaozhang/tacred-relation`.

[8] Available for download from `https://catalog.ldc.upenn.edu/LDC2018T24`

The UCCA-based adjacency matrix slightly outperforms the UD-based one, providing experiment-based evidence that tokens on the path between *subject* and *object* on the UCCA bi-lexical tree reduction, provide more linguistic cues for the RE task than those on the equivalent UD path. The superiority of the UCCA model is despite some loss of semantic content that occurs when transforming a full UCCA DAG to a bi-lexical one. Interestingly, the addition of A_{cons} further improves the result.

Armed with baseline performance, we employ our novel UCCA terminal-path-to-root embedding method described in section 4.3. We first apply UCCA embeddings to the C-GCN$_{ud}$ model; we observe an F_1 score improvement of close to 0.4 points. We observe a similar improvement when we apply UCCA embeddings to C-GCN$_{all}$: the F_1 score rises from 66.94 to 67.32.

Our headline result is an improvement of 1.05 F_1 points between the C-GCN$_{ud}$ system described by Zhang et al. with a mean score of 66.27, and our C-GCN$_{all+emb}$ system, with a mean score of 67.32. We confirm the statistical significance of these results by conducting Welch's t-tests and Mann-Whitney U-tests. For further discussion, see Appendix F.

We follow the practice of performing ensemble testing (Zhang et al., 2017b; Zhou et al., 2020), by applying a soft-max function on the sum of the classifier's logits from five different C-GCN$_{all+emb}$ models. We repeat the ensemble experiment for all combinations of five models, from the set of C-GCN$_{all+emb}$ models we produced in separate runs of the experiment. We average the F_1 scores for a final score of 68.41. Table 2 summarizes these results.

Finally, we perform a relation-based comparison of C-GCN$_{ucca}$ and C-GCN$_{ud}$, to determine whether the performance improvements are a product of improved results in specific relation categories. Table 3 lists the results for all the relations for which a difference of over 5% was demonstrated. We discuss these results in the next section.

6 Discussion

Our goal is to measure how UCCA performs in comparison with UD based C-GCN model. The results of our baseline experiment provide evidence that the graph convolution network produces a slightly stronger model when the UD sentence representation is substituted for a UCCA bi-lexical representation, and a significantly stronger model when UD and UCCA are used in tandem.

Relation	**Avg. Improvement**
per:country_of_birth	71.7%
per:city_of_birth	29.4%
per:city_of_death	10.5%
per:date_of_death	9.4%
org:country_of_headquarters	8.3%
per:stateorprovinces_of_residence	7.2%
per:stateorprovince_of_death	6.4%
per:schools_attended	-6.2%
org:parents	-8.3%
org:subsidiaries	-8.9%
org:shareholders	-9.4%
per:children	-9.6%
org:political/religious_affiliation	-23.0%
per:other_family	-55.4%

Table 3: Relations with average improvement/decline of above 5% for C-GCN$_{ucca}$ in comparison to C-GCN$_{ud}$.

In our main line of experimentation, we attempted to utilize the entire UCCA representation, rather than its bi-lexical reduction. We tested a system that uses embedding representation of the path of each token, within the minimal sub-DAG that contains both *subject* and *object*, to the root of that sub-DAG. We found that integration of this additional input produces superior results for all the models we tested: C-GCN$_{ud}$, C-GCN$_{ucca}$ and C-GCN$_{all}$; interestingly, the improvement is the smallest for C-GCN$_{ucca}$ itself. We postulate that the results demonstrate the efficacy of this novel approach for embedding semantic representation and believe that further analysis of this method in the context of other systems for semantic representation could be helpful to the NLP community.

As detailed by Zhang et al., the C-GCN pooling mechanism allows for identification of which tokens contribute significantly to the pooling results (block 4 of the network). Thus, it is possible to directly analyze sentences classified correctly by, for example, C-GCN$_{all+emb}$, but incorrectly by C-GCN$_{ud}$, to ascertain which tokens contribute significantly to the pooling stage output in the former but not the latter. We leave this analysis for future work as well.

To complete our study, we compared UCCA and UD performance, relation by relation. All the relations in which UCCA exhibits improved results are location-oriented. Contrarily, UD does considerably better than UCCA for family relations. Why should UCCA do better for location relations at the expense of its performance for family relations? We note that location relations associate a subject, person, or organization, to the object, a location, via an explicit event such as birth, death or residence. This is different from a family relation, for which there is detachment between the event that evoked the relation, and the reality of the relation at the present time. Indeed, for family relations the relation-evoking event may vary considerably for the same relation (consider adoption versus birth). We hypothesize that UCCA, with its improved semantic awareness, is more adept at sensing an event that binds object to subject, as opposed to UD, which performs better for family relations that may emerge more readily from syntax. This finding could lead to improved classification performance by constructing an ensemble of UCCA and UD models, and weighting the softmax function on the sum of the classifiers to favor UCCA results for location relations and UD results for family ones. This experimentation is also left for future work.

7 Conclusion

Much recent work on RE focuses on improving the contextualized representations of words and entities, including by means of injecting knowledge into large pretrained models such as BERT, to be used for a final step of supervised training using comparatively small training datasets like TACRED. This approach is successful in overcoming the dearth of annotated data required for supervised learning for RE and is producing ever-improving bottom-line results. However, it further intensifies the interpretability challenges posed by end to end models and deepens the chasm between the linguistic domain and the practice of NLP.

Our approach also leverages pretrained models that are dependent on extensive prior training data (via the TUPA parser); however, the intention is linguistically explicit: generation of UCCA representations for the sentences in the TACRED dataset. The C-GCN architecture then provides visibility into which words actually contribute to a relation classification, providing a level of transparency devoid in end-to-end models.

Our results indicate that representation of explicit semantic structure is indeed beneficial for the RE task, providing linguistically-explicit means for leveraging prior training to the task of relation extraction.

Because our approach to embedding semantic structure is not specific to UCCA, future work could analyze the effects of integrating other semantic representations, as well as the possibility of embedding semantic structure representation in the latest cutting-edge RE models.

Acknowledgments

This work was supported by a grant from the Allen Institute for Artificial Intelligence. We thank Yoav Goldberg for his advice and support. We thank Or Honovich and Shachar Don-Yehiya for their contribution to the rule-based pattern matching work that inspired this study.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries - DL '00*. ACM Press.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yonatan Belinkov, Sebastian Gehrmann, and Ellie Pavlick. 2020. Interpretability and analysis in neural NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. Association for Computational Linguistics.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *Lecture Notes in Computer Science*, pages 172–183. Springer Berlin Heidelberg.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. *arXiv preprint cs/9907013*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: proceedings of the work-*

shop on cross-framework and cross-domain parser evaluation, pages 1–8.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M Kaplan. 2003. The parc 700 dependency bank. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - ACL-IJCNLP '09*. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2018. Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. *arXiv preprint arXiv:1803.09578*.

Sebastian Schuster and Christopher D Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378.

Aryeh Tiktinsky, Yoav Goldberg, and Reut Tsarfaty. 2020. pyBART: Evidence-based syntactic transformations for IE. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017a. An evaluation of predpatt and open ie via stage 1 semantic role labeling. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017b. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Li Zhou, Tingyu Wang, Hong Qu, Li Huang, and Yuguo Liu. 2020. A weighted gcn with logical adjacency matrix for relation extraction. In *European Conference on Artificial Intelligence 2020 (ECAI2020)*.

A Example of UCCA Advantage

Figure 4 provides an example of UCCA's advantage over UD in capturing scenes and their participants by representing non-verbal predicates ("graduation") like verbal ones ("transition").

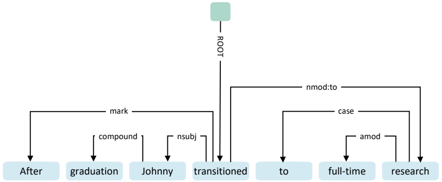

(a) UD Parse Graph

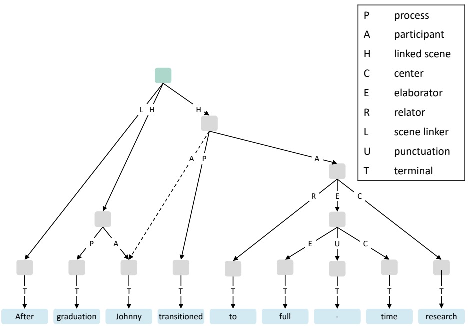

(b) UCCA Parse Graph

Figure 4: UD v1 (enhanced++) and UCCA representation graphs for the sentence 'After graduation Johnny transitioned to full-time research'. The UCCA parse directly captures Johnny's participation in both the occurrences described in the sentence, graduation, and transition. This semantic connection does not arise from the UD representation (not even the enhanced++ flavor).

B Example of Bi-Lexical Reduction for UCCA

Figure 5 provides a graphic illustration for the results of bi-lexical reduction of UCCA representation, for the sentence depicted in figure 4. The product of this bi-lexical reduction is a representation containing terminal nodes only.

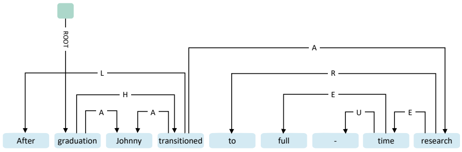

Figure 5: Bilexical UCCA representation of the sentence from 1.

As this example illustrates, the bi-lexical reduction constitutes a DAG; in this example "Johnny" is the child of both the "graduation" and the "transitioned" terminals.

C C-GCN Architecture

We describe the six blocks of the C-GCN model as depicted in figure 2.

Block 1 – Word Embeddings: Zhang et al. use 300-dimensional GloVe vectors to initialize word embeddings corresponding to each token. Each token's initial corresponding GloVe based vectors is further extended with 30-dimensional embedding for the token's part-of-speech, and another 30-dimensional embedding for its entity type. Part-of-speech and entity-type are both provided as input in the TACRED dataset (POS and NER attributes respectively) and are initialized in training with random values for each of the unique classes in the part-of-speech of entity-type sets. All the embeddings undergo fine-tuning during training and are therefore variables of the model.

Block 2 – LSTM: The input vector embeddings do not contain information about word order or contextual cues required for disambiguation. A bi-directional long short-term memory network addresses this, by taking the list of 360-dimensional vectors corresponding to the sentence's tokens and producing a corresponding list of 400-dimensional vectors.

Block 3 – Graph Convolution: We cover the third block in section 4.1

Block 4 – Pool: The fourth block is responsible for converting the two-dimensional sentence representation into a single-dimensional vector. We resort to formalism to describe precisely how this is achieved. Let $\mathcal{H}_p=[h_{p_1},\ldots,h_{p_n}]$ denote the output representations of the final GCN layer for all on-or-near-path tokens, and let $\mathcal{H}_s=[h_{s_1},\ldots,h_{s_n}]$ and $\mathcal{H}_o=[h_{o_1},\ldots,h_{o_n}]$ denote the vector lists corresponding to the *subject* and *object* spans respectably. A simple max pooling function is applied to $\mathcal{H}_p$, $\mathcal{H}_s$ and $\mathcal{H}_o$, resulting in three single vectors, $h_{p_{max}}$, $h_{s_{max}}$ and $h_{o_{max}}$, representing the on-or-near-path tokens, the *subject* and the *object* respectably. The output of block four is a simple concatenation of these vectors resulting in a single 600-dimensional vector.

Block 5 – Feedforward Network Block five is a simple stacked feed-forward network, with each FF layer comprising a single linear transformation followed by a RELU non-linearity. Zhang et al. use two layers as depicted in the diagram. The first layer receives a 600-dimensional vector and produces a 200-dimensional vector as output; all other layers receive and produce 200-dimensional vectors.

Block 6 – Classifier The sixth and final block of the network is a simple feed-forward linearity, which receives the 200-dimensional vector from block five, and outputs a 42-dimensional vector, one dimension per relation category. This final vector is used in both training with a cross-entropy-loss function, and in evaluation, where a soft-max function is applied to produce a probability vector. It is possible to control the recall/precision balance by applying a thresholding mechanism such that a positive classification (i.e. any label other than *no_relation*) must surpass a threshold greater than 0.5, however the model chooses to treat a negative classification symmetrically.

D Token Alignment

TUPA, the UCCA parser we briefly mentioned in the introduction, uses the the SpaCy NLP pipeline for basic NLP tasks including tokenization. SpaCy's default tokenization results vary considerably from TACRED's given tokenization. Indeed, in 30% of train sentences, 28% of dev sentences, and 25% of test sentences tokenization is different. For example, differences arise in the case of intra-word-hyphens[9]. Additionally, the TACRED dataset contains some obvious tokenization errors; for example, there are over 130 entries in which two sentences have been merged into one, by erroneously fusing the last token of the first sentence, it's period punctuation mark, and the first one of the next sentence into a single token.

To address these tokenization concerns, we reparse the entire TACRED dataset with the Standford CORENLP parser, configuring it to adhere to the given tokenization produced by the TUPA parser[10]. This results in a tokenization-aligned standford_pos, stanford_ner, stanford_head and stanford_deprel attribute set (see table 4). For subj_start, subj_end, obj_start and obj_end, which are human annotated properties, we employ a simple algorithm to reassign their index values to correspond to TUPA's tokenization. The end result is a transformed TACRED input dataset where all sentence attributes that require adherence to the sentence's token breakdown conform with TUPA/SpaCy tokenization.

Attribute	**Description**
id	UUID for the sentence
relation	The relation that exists between subject and object (or no_relation) if no relation exists
token	Array of the sentence's tokens, the token at each array position corresponding to the token at the same
subj_start	start zero-based index of the subject token span
subj_end	end zero-based index of the subject token span
obj_start	start zero-based index of the object token span
obj_end	end zero-based index of the object token span
subj_type	Entity type of subject (either organization or person)
obj_type	Entity type of object (not limited to organization and person; 18 different classes of object type, including, for example, location, duration and ideology)
stanford_pos	Array of the sentence tokens' parts-of-speech, the part-of-speech at each array position corresponding to the token at the same
stanford_ner	Array of the sentence tokens' entity types, the entity-type at each array position corresponding to the token at the same (with 'O' used to indicate a non-entity token)
stanford_head	Array of the sentence tokens' head word one-based indices, the index at each array position corresponding to the head of the token at the same
stanford_deprel	Array of the sentence tokens' incoming dependency type, the index at each array position corresponding to the incoming dependency type of the token at the same

Table 4: Main attributes associated with each sentence in the TACRED dataset

As a baseline, we reproduce the experimental setup of Zhang et al. (2018), using the given sentence attributes as they appear in the TACRED dataset. We compare these results with results we attain when using TACRED sentences **re-annotated** by the CORENLP engine, using SpaCy/TUPA sentence tokenization, as described above. Table 5 captures this comparison. We hy-

---

[9] Discussion on this tokenization divergence on Stack Overflow: `https://stackoverflow.com/questions/52293874/why-does-spacy-not-preserve-intra-word-hyphens-during-tokenization-like-stanford`

[10] We achieve this by reconstructing a sentence where all TUPA generated tokens are separated by whitespace, and then setting CORE NLP's 'tokenize.whitespace' flag to True

System	Note	F_1
GCN	reported result	64.0
C-GCN	reported result	66.4
C-GCN	our result with given dataset	66.68
C-GCN	our results with TACRED input re-annotated with CORE NLP engine, using SpaCy tokenization	66.27

Table 5: Our results for the unmodified C-GCN model, using given and re-annotated TACRED datasets.

pothesize that the decrease in score when using re-annotated data may stem from errors introduced by our procedure for reassignment of index values for the human annotated subj_start, subj_end, obj_start and obj_end properties.

E Variations of DAG encoding

We test two variations of our UCCA terminal-path-to-root embedding method: the first adds non-zero embedding representations for the token-to-root path of *all* tokens in the sentence; the second uses non-zero embeddings for tokens in the *minimal sub-DAG* only, using zero vector embedding for all other tokens. We also perform an ablation test by using random values for each unique path embedding, rather than values that encode the actual paths.

System	Path Encoding	Min Sub-DAG	F_1
C-GCN$_{ucca}$†	-	-	66.44
C-GCN$_{ucca+emb}$	✗	✓	65.63
C-GCN$_{ucca+emb}$	✓	✗	66.22
C-GCN$_{ucca+emb}$	✓	✓	66.60

Table 6: Measuring impact of UCCA embeddings under different settings. † marks the baseline C-GCN$_{ucca}$ model score as reported in table 1. An ✗ in the **Path Encoding** column indicates the use of random values.

The results in table 6 indicate that using path encodings for tokens in the minimal sub-DAG produce improved results when compared to the plain C-GCN$_{ucca}$ model. While the improvement is marginal – 0.2 F_1 points – we do see a more marked impact, to the negative, when we use random initialization. When we consider all sentence tokens and not just those in the minimal sub-DAG evoked by *subject* and *object* we also see a negative impact, albeit more mild.

F Statistical Significance

We call attention to Reimers and Gurevych (2018), who critique the common evaluation practice of selecting the model with the median dev F_1 from five independent runs and reporting its test F_1 result. As noted, the F_1 scores reported in our experiments are the mean scores on the test set for 20 multiple models, trained according to the parameters of the system at hand (e.g. GCN$_{ud}$, C-GCN$_{ucca}$), using the full TACRED train and dev datasets, and tested with the test dataset.

Our headline result is an improvement of 1.05 F_1 points between the C-GCN$_{ud}$ system described by Zhang et al., with a mean score of 66.27, and our C-GCN$_{all+emb}$ system, with a mean score of 67.32. We conduct two sets of significance tests to validate this result:

Welch's t-test: Welch's t-test assumes that the compared distributions are approximately normally distributed. We apply Welch's t-test to a pair of 20 sample pairs, each pair using the same random seed, with the first set corresponding to C-GCN$_{ud}$, and the second to C-GCN$_{all+emb}$, with a null hypothesis asserting that both systems produce models with similar F_1 scores on the test set. The resulting p-value is 3.015e-09. We subtract 0.75 F_1 from all the samples corresponding to C-GCN$_{all+emb}$ and reapply Welch's t-test. The p-value for this new null hypothesis is 0.019. In other words, we have a statistically significant improvement of 0.75 F_1 points with $p < 0.02$.

Mann-Whitney U-test: We abandon the assumption that our systems produce models with normally distributed F_1 on the test set and apply the Mann-Whitney U test after once again subtracting 0.75 F_1 from the C-GCN$_{all+emb}$ samples. The p-value for this new null hypothesis is 0.027. Restated, we have demonstrated a statistically significant improvement of 0.75 F_1 points with $p < 0.03$.

Dynamic and Multi-Channel Graph Convolutional Network for Aspect-Based Sentiment Analysis

Shiguan Pang[1], Yun Xue*, Zehao Yan[2], Weihao Huang[2], Jinhui Feng[2]
School of Physics and Telecommunication Engineering
South China Normal University Guangzhou, China
{psg-nlp,xueyun}@m.scnu.edu.cn

Abstract

Aspect-based sentiment analysis aims to judge the polarity of the given aspect word in reviews. Most recent methods adopt syntax-based Graph Neural Networks to extract the syntactic information from the dependency graph, thinking that would be beneficial for establishing relations between aspect and opinion words. However, these methods may ignore that some sentences have no remarkable syntactic structure, which causes the opposite judgement in sentiment analysis. In this paper, we figure out this problem by means of optimally fusing syntactic information, semantic information and their combinations simultaneously. Firstly, syntactic graphs and semantic graphs are generated by dependency tree and multi-head self-attention respectively. Then we propose a Dynamic and Multi-channel Graph Convolutional Network (DM-GCN) to learn the correlated information from the generated graphs effectively. Our extensive experiments on SemEval 2014 and Twitter datasets confirm that DM-GCN fuses syntactic, semantic and their combinations optimally and outperforms all state-of-the-art alternatives with a large margin.

1 Introduction

Aspect-Based Sentiment Analysis (ABSA) is a fine-grained task of sentiment analysis. Distinguishing from document-based and sentence-based sentiment analysis, aspect-based sentiment analysis aims to determine the sentiment polarities of specific aspect words in given texts. As an example of ABSA, in the sentence *'The food is good but the service is bad'*, the sentiment polarities of given aspect words *food* and *service* are recognized as *Positive* and *Negative*. As such a more detailed analysis of multiple aspect sentiment gives rise to provide targeted insights into reviews.

In general, there are several steps when performing aspect-based sentiment analysis, such as obtain word embeddings, encode the syntactic information and extract semantic information, among which mining the most relevant opinion word plays the pivot role. Benefited from applying attention mechanisms to connect aspect words with their opinion words, a few studies (Wang et al., 2016b; Li et al., 2017; Ma et al., 2017; Fan et al., 2018)have reported attractive results. However, limited by the co-occurrence frequency or long-range word decencies, attention mechanism may assign wrong value for irrelevant words. As an example, *'the stuff should be more friendly'*, due to the high co-occurrence frequency of *'stuff'* and *'friendly'*, attention mechanisms may regard *'friendly'* as the opinion word for *'staff'*, which result in an opposite sentiment judgement completely.

To address those limitations, a slice of studies encode the syntactic structure as crucial information to establish the connections between aspects and their opinion words. A handcrafted syntactic rule (Liu et al., 2013) has been encoded into the model as an early essay.However, it is over-relied upon the quality of the rule. Then dependency trees are regarded as another form of syntactic rule, (Dong et al., 2014; Nguyen and Shirai, 2015; Wang et al., 2016a) have encoded dependency tree by a Recursive Neural Network (RNN). Recently, the rapid development of graph neural networks has attracted a surge of interest, due to the great capacity of learning structure representation, a class of graph neural networks is designed to extract syntactic information from dependency tree (Zhang et al., 2019a; Sun et al., 2019; Huang and Carley, 2019; Wang et al., 2020b; Zheng et al., 2020). Although these approaches make quite a few improvements compared with those attention-based models, the shortcomings should not be ignored. First and foremost, sentences have different sensitivities to syn-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2627–2636
August 1–6, 2021.

tactic information and semantic information. In particular, those sentences that have inconspicuous syntactic structure, are low sensitive to syntactic information, which means that syntactic information may fail to help the model to determine the sentiment polarity of the sentences in some cases. Given as an example, 'Charlie sheen your fucking annoying', a sentence in twitter dataset, prefers to colloquial expression, without explicit syntactic structure, which makes those extracted structure information become noise. Besides, not all the information on the dependency tree is valid for our task, the noise would be encoded by the GNNs-based model, then the secondary noise will be caused if attention mechanism is used on this basis (Zhang et al., 2019a). To make matter worse, GCNs has been proved that fail to optimally integrate node features (semantical information) and topological structures (syntactic structure) in a dependency graph with rich information (Wang et al., 2020b), indicating that GCN may arduous to learn some deep correlation between topological structures (syntactic structures) and node features (semantical information).

In reality, syntax is complementary to semantics. Motivated by this fundamental assumption, our Dynamic and Multi-Channels Graph Convolutional Network (DM-GCN) is designed to get over the weakness mentioned above. Firstly, on purpose to exploit the information in semantic space adequately, the semantic graphs are generated and updated dynamically by a multi-head self-attention mechanism in the graph convolutional network. Secondly, the dependency trees of sentences are transformed into syntactic graphs, which is the same as the model mentioned above (Sun et al., 2019; Zhang et al., 2019a). With the semantic graphs and syntactic graphs, two specific graph convolutional networks are applied to extract two specific information from semantic and syntactic space. Furthermore, considering that the common information of semantic and syntactic space is as significant as the specific information, a parameter sharing graph convolution module is designed to extract the common information. Finally, a trainable parameter matrix is utilized to adaptively learn the importance of different embeddings and fuse them optimally. Extensive experiments are carried out on SemEavl2014 and Twitter datasets, and experimental results demonstrate that DM-GCN achieves predominant performance compared with baseline methods.

The contributions of this paper can be summarized as follows:

- We have come up with a kind of dynamic semantic graph, generated and updated by the multi-head self-attention mechanism, which focuses on extract the most relevant information in the semantic space.
- We propose a novel Dynamic and Multi-channel GCN(DM-GCN) to adaptively learn and fuse according to the characteristics of sentences.
- Extensive results testify the significance of leveraging syntactical, semantical, and their combinational information properly, and demonstrate the availability of our proposed model in extracting and fusing them in sentiment classification.

The rest of the paper is organized as follows. In Section 2 we briefly review the related work of aspect-based sentiment analysis. In Section 3 we develop our proposed model DM-GCN. The evaluation of our model on benchmark datasets and the experimental result analysis are in Section 4.Finally, concluding remarks are presented in Section 5.

2 Related Work

With the appealing development of aspect-based sentiment analysis, numbers of research can be divided into three categories approximately: Attention-based neural models, Syntactic rules neural network, and Syntactic-based graph neural network. Our work is based on a plentiful line of those recent efforts.

Attention-Based Neural Networks: Originated in computer version, attention-based models have led to promising progress in sentiment analysis as well, which are committed to building up connection between aspect word and opinion word precisely. Among them, (Wang et al., 2016b) proposed an attention-based LSTM to obtain significant information for given aspects. (Chen et al., 2017)utilized a multi-layer attention network to address long-distance dependency problem between aspect and opinion words. With transformer being proposed, the pre-trained language model make impressive achievement in NLP including aspect-based sentiment analysis (Xu et al., 2019; Sun et al., 2019; Vaswani et al., 2017).

Syntactic-Based Recurrent Neural Networks: Due to the lack of structural information, attention-based methods may assign the attention weights mistakenly. Some research tries to address this issue by encoding syntactic rules in word representation. (Dong et al., 2014)proposed AdaRNN, which can propagate the sentiment information via syntactic dependency tree. (He et al., 2018a)proposed an attention mechanism that incorporates syntactic information.(Zhang et al., 2019b)tried to utilize the distance between aspect word and context on syntactic dependency tree as auxiliary weight.

Syntactic-Based Graph Neural Networks: More recently, with the rapid development of graph neural networks (Kipf and Welling, 2019; Velickovic et al., 2018), the combination of syntactic dependency tree and graph neural networks have shown gratifying results in ABSA. (Zhang et al., 2019a; Sun et al., 2019) transformed syntactic dependency tree into graph structure and utilized GCN to extract syntactic information. (Tang et al., 2020) proposed Bidirectional-GCN to encode the directional information of syntactic dependency tree. (Wang et al., 2020a) proposed a new GAT model to encode the dependency relations. However, whether the correlated information can be extracted according to the characteristics of the sentence from GNNs adaptively remains indistinct.

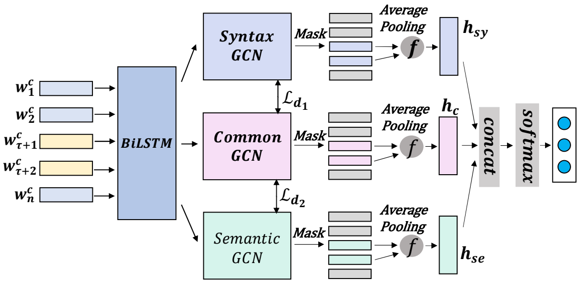

Figure 1: Model Architecture

3 Methodology

In this section, we elaborate on the details of our proposed model. The overall framework of DM-GCN is shown in Figure 1. Inspired by AM-GCN (Wang et al., 2020b), our key idea is that the proposed model should be available to aggregate syntactical features, semantical features and their combination features becomingly to address the issues mentioned before. For this purpose, firstly, semantic graphs are constructed by the multi-head self-attention mechanism. Secondly, syntactic graphs are transformed from the dependency tree of sentences. Then two specific convolution modules and a common convolution module are used to extract corresponding information respectively. Finally, a trainable parameter is used to fuse extracted information more suitably for our task. The components of DM-GCN will be illustrated in the rest of the section.

3.1 Embedding and Bidirectional LSTM

An n-word sentence $x = \{w_1^c, w_2^c, \cdots, w_{\tau+1}^c, \cdots, w_{\tau+m}^c, \cdots, w_n^c\}$ with the aspect $\{w_{\tau+1}^c, \cdots, w_{\tau+m}^c\}$is given, each word in the sentence will be embedded into a low-dimensional vector by looking up in a pre-trained word embedding matrix $E \in R^{|V| \times d_e}$,where $|V|$ is the lexicon size and d_e is the dimension of word embedding (Bengio et al., 2003).With the sentence embeddings, a bidirectional LSTM is constructed to learn the hidden representation of the given sentence x ,which can be conveyed as $\vec{H}^c = \{\vec{h}_1^c, \vec{h}_2^c, \cdots, \vec{h}_{\tau+1}^c, \cdots, \vec{h}_{\tau+m}^c, \vec{h}_{n-1}^c, \vec{h}_n^c\}$ in the forward direction and $\overleftarrow{H}^c = \{\overleftarrow{h}_1^c, \overleftarrow{h}_2^c, \cdots, \overleftarrow{h}_{\tau+1}^c, \cdots, \overleftarrow{h}_{\tau+m}^c, \overleftarrow{h}_{n-1}^c, \overleftarrow{h}_n^c\}$ in the backward direction. Concatenating the forward and backward representation, $H^c = [\vec{H}^c, \overleftarrow{H}^c] \in R^{n \times d_{lstm}}$ is the final representation encoded by Bidirectional LSTM, which contains the contextual information between aspect words and opinion words.

3.2 Syntactic Graph Convolution Module

As previous described (Sun et al., 2019; Zhang et al., 2019a), we transform the dependency tree into the graph structure $G_{sy} = (A_{sy},\ H^c)$, A_{sy} is the adjacency matrix of the graph and H^c is the feature matrix.Then the graph convolutional network is applied to extract the syntactic information, which is formulated as below:

$$GCN(\widetilde{A}, H^{(l)}, W^{(l+1)}) = ReLU(\widetilde{A}H^{(l)}W^{(l+1)}) \quad (1)$$

$$H_{sy}^{(0)} = H^c \quad (2)$$

$$H_{sy,in}^{(l)} = [H_{sy}^{(0)}; \ldots; H_{sy}^{(l)}] \quad (3)$$

$$H_{sy}^{(l+1)} = \mathrm{GCN}(\tilde{A}_{sy}, H_{sy,in}^{(l)}, W_{sy}^{(l+1)}) \quad (4)$$

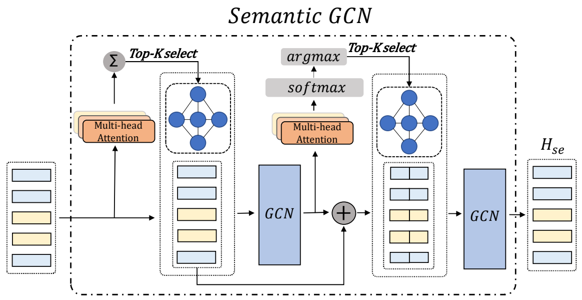

Figure 2: Architecture of Semantic Graph Convolution Module

Where $\widetilde{A}_{sy} = \widetilde{D}^{-\frac{1}{2}}\left(A_{sy}+I_f\right)\widetilde{D}^{-\frac{1}{2}}$ is the adjacency matrix with self-loop (Kipf and Welling, 2019), $H_{sy,in}^{(0)} = H^c \in R^{n \times d_{lstm}}$ is the words representation encoded by Bi-LSTM, we take it as the input of the first GCN layer. When $l \in [1, L-1]$ the input $H_{sy,in}^{(l)}$ is generated by the concatenation of the output produced in layers $0, \cdots, l-1$, $W_{sy}^{(l+1)} \in R^{(d_{lstm}+l*d_{gcn}) \times d_{lstm}}$ is the learnable matrix of the l-th GCN layer, d_{lstm} is the dimension of the hidden representation learned by bidirectional LSTM, d_{gcn} is the dimension of the GCN-layer output and d_{head} is the dimension of each attention head. With l-$step$ convolutional operation, each node can iteratively aggregate the information from its one-hop neighbors and update its representation. Syntactic information is integrated into the final representation $H_{sy}^{(L)}$ successfully by Syntactic Graph Convolutional Module.

3.3 Semantic Graph Convolution Module

As we discuss in Section 1, a few short sentences have vague syntactic structure, if the model extracts the syntactic information stiffly, may trigger the opposite effect. On the other, the original dependency tree can be seen as a form of hard attention mechanism (Xu et al., 2015), which may lead to the information omission when aspects are far away from its opinion words or two words are supposed to be connected but they don't on their dependency tree. Inspired by (Huang and Carley, 2019), we develop Semantic Graph Convolutional Module to distinguish the correlated context words directly.

Specifically, the semantic graph convolution module is divided into initialization and update respectively. we take $H_{se,in}^{(0)}$ as the input of the initialization, the multi-head self-attention will be utilized to construct k attention score matrices based on $H_{se,in}^{(0)}$. On purpose to enhance the robustness of the model, these k matrices will be summed up intuitively. Then a top-k selection is applied on the summed-up matrix, which means that the top-k significant context words are preserved. More specially, the adjacency matrix is supposed to keep symmetry after top-k selection. In this way, we have initialized the adjacency matrix $A_{se}^{(0)}$ and a GCN layer is applied to extract the semantic information preliminarily. The initialization is formulated as:

$$H_{se,in}^{(0)} = H^c \tag{5}$$

$$A_{se,i}^{(0)} = \frac{(H_{se,in}^{(0)} W_{se,k}^{(0)})(H_{se,in}^{(0)} W_{se,q}^{(0)})^T}{\sqrt{d_{head}}} \tag{6}$$

$$d_{head} = \frac{d_{lstm}}{K} \tag{7}$$

Specially, when $l = 0$:

$$A_{se}^{(0)} = top\text{-}k\left(\sum_{i=0}^{K} A_{se,i}^{(0)}\right) \tag{8}$$

$$H_{se}^{(1)} = GCN(\tilde{A}_{se}^{(0)}, H_{se,in}^{(0)}, W_{se}^{(1)}) \tag{9}$$

where $H_{se,in}^{(0)} \in R^{n \times d_{lstm}}$ is the words representation encoded by Bi-LSTM, K is the number of heads, $A_{se,i}^{(0)}$ is the i-th adjacency matrix, $W_{se,k}^{(0)}, W_{se,q}^{(0)} \in R^{d_{lstm} \times d_{head}}$ are trainable weight matrices, $W_{se}^{(1)} \in R^{d_{lstm} \times d_{gcn}}$ is the parameter matrix of the GCN layer.

With the initialized semantic graph $A_{se}^{(0)}$, the process of update is formulated as :

$$H_{\mathrm{se,in}}^{(l)} = [H_{\mathrm{se}}^{(0)}; \ldots; H_{\mathrm{se}}^{(l)}] \tag{10}$$

$$A_{se,i}^{(l)} = \frac{(H_{se,in}^{(l)} W_{se,k}^{(l)})(H_{se,in}^{(l)} W_{se,q}^{(l)})^T}{\sqrt{d_{head}}} \tag{11}$$

$$\mathrm{A}_{se}^{(l)} = argmax[softmax(A_{se,1}^{(l)}, \cdots, A_{se,K}^{(l)})] \tag{12}$$

$$A_{se}^{(l)} = top\text{-}k(A_{se}^{(l)}) \tag{13}$$

$$H_{se}^{(l+1)} = GCN(\tilde{A}_{se}^{(l)}, H_{se,in}^{(l)}, W_{se}^{(l+1)}) \tag{14}$$

Put slightly different, the input $H_{se,in}^{(l)}$ is generated by the concatenation of the output produced

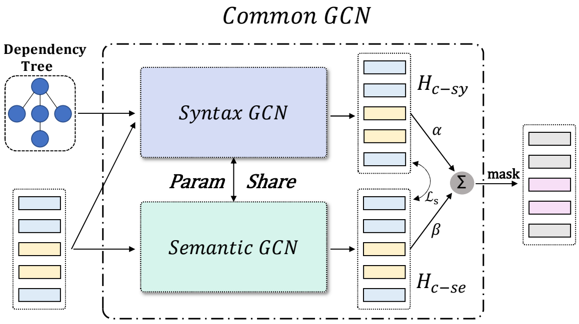

Figure 3: Architecture of Common Graph Convolution Module

in layers $0, \cdots, l-1$, which will be delivered into the multi-head self-attention to generate k attention score matrices just as initialization does. Then we figure out the matrix with maximum probability by $softmax$ function, which can be regarded as the most relevant attention matrix of the semantic graph for the current sentence. After *top-k* selection, a GCN layer is utilized to extract the deep-seated semantic information. We denote the last layer output representation as $H_{se}^{(L)}$, which contains the most relevant semantic information of aspects and its opinion word.

3.4 Common Convolution Module

According to (Liina and Pylkknen, 2019), syntax and semantic space are not totally separating, as the syntactic structure of sentence changes, so does the semantic, which gives reason to believe that extracting the common information shared by syntactic and semantic space is beneficial to comprehend the sentence better. Motivited by (Wang et al., 2020b), a common graph convolutional module (Figure 3) with parameter sharing strategy is applied to obtain the information shared in two spaces.

For simplicity, we represent the Syntactic Graph Convolution Module and the Semantic Graph Convolution Module as:

$$H_{sy}^{(L)} = SYGCN(A_{sy}, H^c, W_{sy}) \tag{15}$$

$$H_{se}^{(L)} = SEGCN(A_{se}, H^c, W_{se}) \tag{16}$$

the common syntactic representation $H_{c\text{-}sy}^{(L)}$ is extracted by common graph convolutional module from syntactic graph $G_{sy} = (A_{sy},\ H^c)$ as follow

$$H_{c\text{-}sy}^{(L)} = SYGCN(A_{sy}, H^c, W_c) \tag{17}$$

Where $W_c = W_c^{(l+1)} \in R^{(d_{lstm}+l*d_{gcn}) \times d_{gcn}}$ is the *l-th* layer learnable weight matrix of Common-GCN, $H_{c\text{-}sy}^{(L)}$ is the final common syntactic representation.

On purpose to extract the shared information, when applying $Common\text{-}GCN$ to learn the node representation from semantic graph $G_{se} = (A_{se},\ H^c)$, we share the same learnable weight matrix $W_c^{(l)}$ for each layer of $Common\text{-}GCN$.

Simply, the calculation is similar to section 3.3:

$$H_{c\text{-}se}^{(L)} = SEGCN(A_{se}, H^c, W_c) \tag{18}$$

Finally, we combine these two representations $H_{c\text{-}sy}^{(L)}$ and $H_{c\text{-}se}^{(L)}$ to obtain the final representation.

$$H_c^{(L)} = \frac{\alpha H_{c\text{-}sy}^{(L)} + \beta H_{c\text{-}se}^{(L)}}{2} \tag{19}$$

Where α and β are trainable parameters, $H_c^{(L)} \in R^{n \times d_{gcn}}$.

3.5 Feature fusion

Before fusing the representation extracted by specific and common convolutional modules, we use the mask mechanism and average pooling to capture the aspect vectors h_{sy}, h_c and h_{se} from $H_{sy}^{(L)}$, $H_{se}^{(L)}$ and $H_c^{(L)}$. Then the final aspect representation is obtained by concatenating h_{sy}, h_c and h_{se}. The formulas are described as:

$$h_{sy} = f(mask(H_{sy}^{(L)})) \tag{20}$$

$$h_c = f(mask(H_c^{(L)})) \tag{21}$$

$$h_{se} = f(mask(H_{se}^{(L)})) \tag{22}$$

$$h_a = [h_{sy}; h_c; h_{se}] \tag{23}$$

where $mask(\cdot)$ represents the masked function, which means that only aspect vectors are activated in $H_{sy}^{(L)}$,$H_{se}^{(L)}$ and $H_c^{(L)}$, the context vectors are discarded. Mask function is formulated as follow:

$$\text{mask} = \begin{cases} 0, & 1 \le t < \tau+1, \tau+m < t < n \\ 1, & \tau+1 \le t \le \tau+m \end{cases} \tag{24}$$

As described in section 3.1: $\tau+1 \le t \le \tau+m$ denotes the aspect words index. $f(\cdot)$ is the average pooling function. The representation h_a is simply concatenated with h_{sy} , h_c and h_{se} , however, it could not distinguish the importance of syntax, semantics and their combination. Considering to reflect the significance of each part, a MLP layer is utilized to learn the adaptive important weights of the representations and fuse them simultaneously. As described in section 3.1: $\tau+1 \leq t \leq \tau+m$ denotes the aspect words index. $f(\cdot)$ is the average pooling function. The representation h_a is concatenated with h_{sy}, h_c and h_{se} , however, it could not distinguish the importance of syntax, semantics and their combination. Considering to reflect the significance of each part, a MLP layer is utilized to learn the adaptive weights of the representations and fuse them simultaneously.

$$\hat{h}_a = ReLU(h_a W_1^T + b_1) \tag{25}$$

By training the learnable weight matrix $W_1^T \in R^{3d_{gcn} \times d_{gcn}}$, our proposed model is able to fuse syntactic, semantic and their combination representations adaptively and derive deeper correlation information for sentiment classification.

Model Training

The final aspect representation $\hat{h}_a$ is sent to a SoftMax layer to calculate the probability distribution of the sentiment polarities as follows:

$$\tilde{y} = softmax((\hat{h}_a)W_2^T + b_2) \tag{26}$$

Considering that the embeddings $H_{sy}^{(L)}$ and $H_{c\text{-}sy}^{(L)}$ are both learned from graph $(A_{sy},\ H^c)$, we hope that different information can be captured by them.Thus, the difference loss between $H_{sy}^{(L)}$ and $H_{c\text{-}sy}^{(L)}$ is defined as $\|H_{sy}^T H_{c\text{-}sy}\|_{l2}^2$.The smaller their inner product is (i.e., orthogonal constraint), the better result is obtained. Similarly, $\|H_{se}^T H_{c\text{-}se}\|_{l2}^2$ is the difference loss between $H_{se}^{(L)}$ and $H_{c\text{-}se}^{(L)}$. We set the difference loss $\mathcal{L}_d$ as:

$$\mathcal{L}_d = \|H_{sy}^T H_{c\text{-}sy}\|_{l2}^2 + \|H_{se}^T H_{c\text{-}se}\|_{l2}^2 \tag{27}$$

In addition, $H_{c\text{-}sy}^{(L)}$ and $H_{c\text{-}se}^{(L)}$ are the outputs of $Common\text{-}GCN$, to confirm that common features are learnt, the smaller the Euclidean distance is, the better representation is obtained. We use $\mathcal{L}_s$ constraint to enhance their similarities:

$$\mathcal{L}_s = \|H_{c\text{-}sy} - H_{c\text{-}se}\|_{l2}^2 \tag{28}$$

Finally, the cross entropy loss with L_2 regularization, as shown in equation(29), all parameters of our proposed model is optimized via backpropagation.

$$\text{L=}-\sum_i \sum_{j=1}^{P} y_i^j \log \hat{y}_i^j + \lambda\|\theta\|^2 + \gamma\mathcal{L}_d + \delta\mathcal{L}_s \tag{29}$$

Table 1: Statistics of datasets

Dataset	Positive		Negative		Neutral	
	Train	Test	Train	Test	Trtain	Test
Rest14	2164	728	807	196	637	196
Lap14	994	341	870	128	464	169
Twitter	1561	173	1560	173	3127	346

where i is the index of i^{th} sample, j is the index of j^{th} sentiment class, P is the number of sentiment classes, y is the real distribution of sentiment and $\hat{y}$ is the predicted one. Besides, λ is the weight of L_2 regularization, θ represents all parameters in the model.γ and δ are hyperparameters of the consistency and disparity constraint terms.

4 Experiment

In this section, we will first introduce the benchmark datasets for evaluation and the baseline models utilized for comparison. Secondly, the experimental implementation and parameter setting will be reported. Then the results of different experiments are analyzed from different perspectives including the comparison with baseline models, ablation study, the effect of hyper-parameters. Finally, some representative cases are discussed.

4.1 Datasets

Our model is evaluated the performance on SemEval 2014(Pontiki et al., 2014) and Twitter dataset provided by (Adaptive recursive neural network for target-dependent twitter sentiment classification), which are authoritative evaluation dataset in ABSA. The statistics of the datasets in Table 1.

4.2 Baseline Methods

We have chosen some mainstream and lasted models in ABSA for comparison, including:

Semantic-based models: ATAE-LSTM)(Wang et al., 2016b), IAN (Ma et al., 2017), RAM (Chen et al., 2017), MGAN (Fan et al., 2018).

Syntax-based models: LSTM+SynATT (He et al., 2018b), AdaRNN (Dong et al., 2014), PhraseRNN (Nguyen and Shirai, 2015), PWCN-Dep (Zhang et al., 2019b), AS-GCN (Zhang et al., 2019a), CDT (Sun et al., 2019), TD-GAT (Huang and Carley, 2019), R-GAT (Wang et al., 2020a), DGEDT (Tang et al., 2020) and Repwalk (Zheng et al., 2020)

Table 2: Sentiment classification results

Category	Method	Twitter		Lap14		Rest14	
		Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
Syn.	AdaRNN(Dong et al.,2014)	66.30	65.90	-	-	-	-
	PhraseRNN(Nguyen and Shirai,2015)	-	-	-	-	66.20	59.32
	LSTM+SynATT(He at al.,2018a)	-	-	72.57	69.13	80.45	71.26
	PWCN-Dep (Zhang at al.,2019)	-	-	76.12	72.12	80.96	72.21
	ASGCN (Zhang et al.,2019)	72.15	70.40	75.55	71.05	80.77	72.02
	CDT (Sun et al.,2019b)	74.66	73.66	77.19	72.99	82.30	74.02
	TD-GAT (Huang and Carley,2019)	-	-	74.00	-	80.35	-
	R-GAT (Wang et al.,2020)	75.57	73.82	77.42	73.76	83.30	76.08
	RepWalk (Zheng et al.,2020)	74.4	72.6	78.2	74.3	83.8	**76.9**
	DGEDT (Tang et al.,2020)	74.8	73.4	76.8	72.3	83.9	75.1
Att.	ATAE-LSTM (Wang et al.,2016b)	-	-	68.70	-	77.20	-
	IAN (Ma et al.,2017)	-	-	72.10	-	78.60	-
	RAM (Chen et al.,2017)	69.36	67.30	74.49	71.35	80.23	70.80
	MGAN (Fan et al.,2018)	72.54	70.81	75.39	72.47	81.25	71.94
Ours	**DM-GCN**	**76.93**	**75.9**	**78.48**	**74.9**	**83.98**	**75.59**
BERT	R-GAT-BERT(Wang et al.,2020)	76.15	74.88	78.21	74.07	86.60	81.35
	DGEDT-BERT(Tang et al.,2020)	77.9	75.4	79.8	75.6	86.3	80.0
	DM-GCN-BERT	**78.06**	**77.36**	**80.22**	**77.28**	**87.66**	**82.79**

* The symbol '-' indicates this result is not available in their work.

4.3 Implementation and parameter settings

For fairness and validation, the best results of compared models are used in models comparation. The Stanford parser (https://stanfordnlp.github.io/CoreNLP/) is utilized to get syntactic dependency tree in our work. Inspired by CDT (Sun et al., 2019), we exploit 300-dimensional Glove vectors (Pennington et al., 2014)for the word embeddings. Besides, on purpose to enrich the sentence representation, a 30-dimensional part-of-speech (POS) embeddings and 30-dimensional position embeddings are concatenated to word embeddings. The rest of parameter settings is given in the code[1].

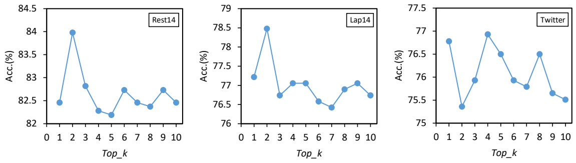

Figure 4(a): Impact of k

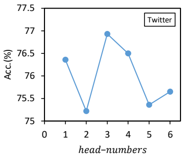

Figure 4(b): Impact of $head-numbers$

4.4 Result and Analysis

4.4.1 Overall Performance Comparison

The results of comparison with all baseline models are shown in Table2. First, compared with the lasted syntactic-based model DGEDT, Repwalk, and R-GAT, DM-GCN significantly improves the accuracy in sentiment classification, which indicates that DM-GCN performs better at encoding the syntactic and semantic information by fusing them adaptively. Second, as the first proposed concept, the common information shared by semantic and syntactic spaces, has been proved its effective in ABSA, which can be judged from the improvement of accuracy on three datasets. The results have proved that only encode semantic or syntactic information is inadequate in aspect-based sentiment classification. On the contrast, fusing the semantic and syntactic information and their combination optimally, just as humanity brain does, is the key to the better performance of our DM-GCN.

4.4.2 Ablation Study

In this section, we further conduct an ablation study to verify the validity of each module in our DM-GCN. The result is in Table 4. The basic DM-GCN is regarded as a baseline model..

First, removal of syntactic information (i.e.DM-GCN w/o H_{sy}) leads to the drop of accuracy drop of 1.42%, 1.11% and 1.65% on Twitter, Lap14 and Rest14 respectively, which demonstrates the significance of syntactic information and verifies syntactic information plays different roles in dif-

---

[1] https://github.com/pangsg/DM-GCN

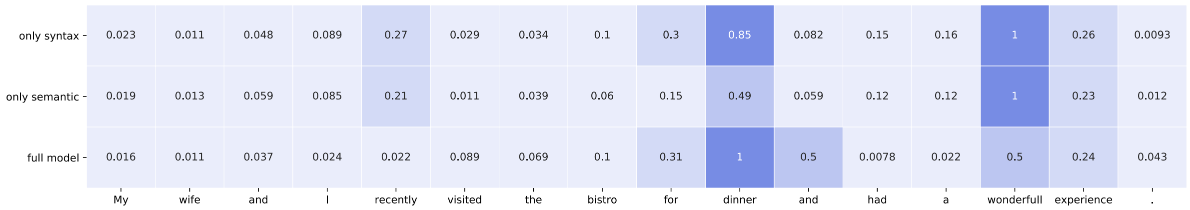

Figure 5: The word relevance scores. The aspect word is ‘dinner’. The ground truth is ‘neutral’ while the prediction of the only syntax, only semantic and full model are ‘positive’, ‘positive’ and ‘neutral’ respectively.

ferent sentence of three datasets. Second, after we get rid of the semantic information (i.e. DM-GCN w/o H_{se}), the model ranges the lowest accuracy on three datasets, which indicates that the dynamic multi-head self-attention mechanism proposed by us is able to capture the most correlated semantic information between aspect and opinion words. As for ‘DM-GCN w/o H_{common}’model, the accuracy decreases 1.3%, 2.54% and 1.63% on Twitter, Lap14 and R Rest14 respectively. Recall the main reason on Lap14, we conclude that the common information mixes up of the characteristics of syntactic and semantic information, which is significant to Lap14 datasets. As a result, the ablation experimental outcomes confirm the contribution of both components.

Table 3: Ablation study results

Model	Twitter Accuracy	Lap14 Accuracy	Rest14 Accuracy
DM-GCN	**76.93**	**78.48**	**83.93**
DM-GCN w/o H_{se}	74.94	75.79	81.92
DM-GCN w/o H_{sy}	75.51	77.37	82.28
DM-GCN w/o H_{common}	75.63	75.94	82.3

* ‘DM-GCN w/o H_{sy}’: removing syntactic convolution module;
‘DM -GCN w/o H_{se}’: removing semantic convolution module;
‘DM -GCN w/o H_{common}’: removing common convolution module.

4.4.3 Hype-parameters Analysis

The results of the research about the hype-parameter k and the head numbers of our proposed semantic convolutional module are in Figure 4.

1. Effect of hype-parameter *k*. From Figure 4a, there are some observation can be noted:1) The best performance appears when k is equals to 3,3 and 4 respectively in Rest14, Lap14 and Twitter datasets. The accuracy decreases on three datasets when k becomes bigger. We conclude that the bigger k we set, the more noise is aggerated from irrelevant words in the sentence, which interferes with our model’s ability to determine the sentiment polarity of sentences.

2. Effect of hype-parameter *head-number*. Take Twitter dataset for example. The result in Figure 4b show the impact of head-numbers in multi-head self-attention mechanism of our models. Intuitively, the best performance appears when head number is equal to 3. Considering the characteristics of multi-head self-attention, we take the attitude that the model becomes haphazard and overfitting when head number decreases and increases respectively.

4.4.4 Case Study and Attention Distribution

In this section, we utilize the method “Mask Experiment” proposed by CDT(Sun et al., 2019)to estimate the sentiment contribution of word w in the sentence a.By leveraging this method,we evaluate the effective of our proposed semantic and syntactic convolution module mathematically. The formula is as follows:

$$\gamma(w,s) = \frac{1}{m}\sum_{i=1}^{d} |\ \hat{h}_a^i - \hat{h}_{(a/w)}^i| \qquad (30)$$

As described in CDT(Sun et al., 2019),where d is the dimension of the final representation $\hat{h}_a$ learned by our model, w is the masked word, which indicates w becomes zero vector,$\hat{h}_{(a/w)}$ is the final representation of sentence a generated by our DM-GCN with the word w masked. If $\hat{h}_a^i = \hat{h}_{(a/w)}^i$, which demonstrates the word w has no sentiment contribution to sentence a.

Figure 5 is the visualization of the attention placed on words, from which can be observed that when syntactic or semantic convolution module (1st or 2nd row) is utilized separately, the model has assigned the highest attention on “wonderful” mistakenly. Regarding “wonderful” as the opinion word of aspect “dinner”, which results in the wrong judgment of the sentence eventually. However, our proposed DM-GCN (3rd row) is able to adaptively reduce the attention on irrelevant word “wonderful” and increase the score on “dinner” adaptively. Implying that our DM-GCN captures the most correlated information from syntactic, semantic and their combination spaces, which verifies the validity of our strategy mentioned before.

5 Conclusion

In this paper, we have proposed a dynamic and multi-channel graph convolution network to encoding the syntactic and semantic information for aspect-based sentiment analysis. Motivated by the human brain's mechanisms of language, we study how to adaptively learned the most correlated information from syntactic, semantic, and their combination spaces and fuse them sufficiently according to the characteristic of sentences. The experimental results on three benchmark datasets showed that we have achieved superior performance over the baseline methods. Besides, the ablation study and case analysis have exploited the influence of hyperparameters and validated the role of each component in our proposed model.

Acknowledgments

This work was supported by the National Statistical Science Research Project of China under Grant No. 2016LY98, the Science and Technology Department of Guangdong Province in China under Grant Nos. 2016A010101020, 2016A010101021 and 2016A010101022, the Characteristic Innovation Project of Guangdong Colleges and Universities (No. 2018KTSCX049), the Science and Technology Plan Project of Guangzhou under Grant Nos. 201802010033 and 201903010013.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 49–54.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3433–3442.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018a. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th international conference on computational linguistics*, pages 1121–1131.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071, Melbourne, Australia. Association for Computational Linguistics.

Binxuan Huang and Kathleen M Carley. 2019. Syntax-aware aspect level sentiment classification with graph attention networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5472–5480.

Thomas N Kipf and Max Welling. 2019. Semi-supervised classification with graph convolutional networks.

Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 671–680.

Liina and Pylkknen. 2019. The neural basis of combinatory syntax and semantics. *Science (New York, N.Y.)*, 366(6461):62–66.

Kang Liu, Heng Li Xu, Yang Liu, and Jun Zhao. 2013. Opinion target extraction using partially-supervised word alignment model. In *IJCAI*, volume 13, pages 2134–2140.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the*

8th International Workshop on Semantic Evaluation (SemEval 2014), pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2019. Aspect-level sentiment analysis via convolution over dependency tree. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5683–5692.

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6578–6588.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *stat*, 1050:4.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020a. Relational graph attention network for aspect-based sentiment analysis. *arXiv preprint arXiv:2004.12362*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 616–626.

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020b. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1243–1253.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019a. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4560–4570.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019b. Syntax-aware aspect-level sentiment classification with proximity-weighted convolution network. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1145–1148.

Yaowei Zheng, Richong Zhang, Samuel Mensah, and Yongyi Mao. 2020. Replicate, walk, and stop on syntax: An effective neural network model for aspect-level sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9685–9692.

Automatic Text Simplification for Social Good: Progress and Challenges

Sanja Štajner
ReadableAI
Cologne, Germany
`sanja.stajner@readableai.com`

Abstract

Since the late 1990s, automatic text simplification (ATS) was promoted as a natural language processing (NLP) task with great potential to make texts more accessible to people with various reading or cognitive disabilities, and enable their better social inclusion. Large multidisciplinary projects showed promising steps in that direction. Since 2010, the field started attracting more attention but at the cost of major shifts in system architecture, target audience, and evaluation strategies. Somewhere along the way, the focus has shifted from ATS for social good towards building complex end-to-end neural architectures that are not aimed at any particular target population. This study presents the trajectory of ATS for social good, the main issues in current ATS trends, and the ways forward that could bring the field back to its initial goals.

1 Rationale

The right to accessible information is a fundamental right that should be granted to all people (UN, 2020). It is the key factor for personal empowerment and social inclusion. Nevertheless, textual information found on the web, in the news, health leaflets, and other sources is often linguistically too complex for many people and thus impedes their active participation in the society.

1.1 Adult Literacy

In the OECD Adult Literacy Report (OECD, 2013), "literacy is defined as the ability to understand, evaluate, use and engage with written texts to participate in society, to achieve one's goals, and to develop one's knowledge and potential". The literacy scale comprises of six levels:[1]

Below Level 1: Able to read brief texts on familiar topics and to locate a single piece of specific information (the process requires only basic vocabulary knowledge).

Level 1: Able to read very short texts (sometimes non-continuous or mixed) and locate a single piece of information identical or synonymous to the information given in the question/directive (requires only basic vocabulary knowledge). Texts may contain little, if any, competing information.

Level 2: Able to integrate two or more pieces of information, use paraphrasing or low-level inferences, compare and contrast or reason about the information requested.

Level 3: Able to understand dense and lengthy texts, various rhetorical structures, perform various levels of inference, and disregard irrelevant or inappropriate content.

Level 4: Able to integrate, interpret or synthesise information from complex or lengthy texts, perform complex inferences and apply background knowledge, identify and understand one or more non-central idea(s), evaluate subtle evidence-claim or persuasive discourse relationships, discern between the correct and competing information which sometimes appear in almost equal ratio.

Level 5: Able to search for and integrate information across multiple dense texts, synthesise various ideas and points of view, evaluate evidence-based arguments, apply and evaluate logical and conceptual models of ideas, evaluate reliability of the sources, selecting relevant information, notice subtle, rhetorical cues and make high-level inferences or use specialised background knowledge.

The percentage of adults (age 16–65) whose literacy is below Level 2 is 16.7%, on average, across 24 countries taken into account (OECD, 2013).[2] In

---

[1] We shorten the description of the skills at each literacy level and maintain only the information that is most relevant for further discussions. The full description of the literacy levels can be found on the page 64 in (OECD, 2013).

[2] Participating countries: Australia, Austria, Belgium (Flanders), Canada, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Ireland, Italy, Japan, Korea, the

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2637–2652
August 1–6, 2021.

Literacy	US	Spain	Italy	Average (24 countries)	Simplification need
Below Level 2	21.7%	28.3%	28.3%	16.7%	lexical simplification
Below Level 3	≈50%	67.4%	70.3%	50.0%	syntactic simplification
Below Level 4	≈86%	≈95%	≈97%	89.4%	conceptual simplification

Table 1: Percentage of population in need for text simplification (the numbers are inferred from (OECD, 2013)).

Italy, Spain, and United States, this percentage is even higher (Table 1). On average, 0.7% of population has the literacy level 5. According to the definitions of the literacy levels in the OECD report, outlined above, one can conclude that: (1) all people with literacy below Level 4 have difficulties understanding conceptually complex texts; (2) all people with literacy below Level 3 additionally have difficulties understanding syntactically complex texts; and (3) all people with literacy below level 2 have difficulties understanding texts which are linguistically difficult in any sense (lexically, syntactically, or conceptually). This means that, according to the results of the OECD report, approximately 16.7% of population needs lexical simplification of everyday texts, 50% of population needs syntactic simplification, and 89.4% of population needs conceptual simplification (Table 1).

1.2 Manual Text Simplification

Text simplification aims to transform original texts into their simpler variants, which are more understandable to the target reader(s), while preserving the original meaning. Some nuances in meaning would inevitably be lost during that process, but the core information should stay the same.

The first guidelines for how to write more accessible texts for international communication, by using reduced vocabulary and restricted number of grammatical rules, were Basic English (Ogden, 1937) and Plain English initiative (Crystal, 1987). Since the late 1990s, many more initiatives have raised awareness about how to write more accessible texts, e.g. how to write for people with intellectual disabilities (Freyhoff et al., 1998; Mencap, 2002; Karreman et al., 2007), how to make public information more accessible to wider audiences (PlainLanguage, 2011), and how to make web content more accessible (W3C, 2008; Cooper et al., 2010). Specialized websites that offer easy-to-read news are now common in many countries, e.g. Noticias fácil in Spain[3], DR in Denmark[4], News Web Easy in Japan[5].

2 Automatic Text Simplification (ATS)

Manual text simplification, apart from being slow and expensive (requiring trained human editors), cannot keep up-to-date with the new information published online, offer variety of written content, or adaptation at a personal level. That created the need for automatic, or at least semi-automatic (involving a manual post-editing step) TS systems.

The first automatic text simplification (ATS) systems were rule-based, with focus on syntactic simplification in English, and were envisioned mainly as a pre-processing step for various natural language processing (NLP) applications, e.g. parsing, machine translation, summarization, and information retrieval; the clarity of texts for human readers was only one of the possible use cases (Chandrasekar et al., 1996; Chandrasekar and Srinivas, 1997). Subsequent ATS systems were proposed for various languages and were encompassing any number of the following simplification operations: lexical simplification, syntactic simplification, or explanation generation (Shardlow, 2014; Siddharthan, 2014).[6]

2.1 ATS Projects for Social Good

Automatic text simplification gained more interest from research community through several national and international projects that secured public founding for building ATS systems for various vulnerable populations and in various languages (Table 2).

Netherlands, Norway, Poland, the Slovak Republic, Spain, Sweden, the United Kingdom (England and Northern Ireland), the United States, Cyprus, and the Russian Federation.

[3] `www.noticiasfacil.es`

[4] `https://www.dr.dk/ligetil/`

[5] `https://www3.nhk.or.jp/news/easy/`

[6] For a comprehensive overview of ATS, please see the book written by Saggion (2017), and the more recent surveys on ATS, lexical simplification, and sentence simplification (Paetzold and Specia, 2017b; Alva-Manchego et al., 2020b; Al-Thanyyan and Azmi, 2021).

Project	Period	Language	Impairment	Text types
PSET	1996–2000	EN	Aphasia	News
PorSimples	2007–2010	PT (BR)	Low literacy	News and educative articles
Simplext	2010–2013	ES	Down syndrome	News
FIRST	2012–2015	EN, ES, BG	Autism	News, literature, health leaflets
Able To Include	2014–2017	EN, ES, NL	Intellectual	E-mails, short messages

Table 2: Projects proposing automatic text simplification for social good.

All aimed to provide a better social inclusion, thus promoting ATS as an NLP task for social good.

ATS for Readers with Aphasia. PSET (Practical Simplification of English Text) was an UK national project (1996–2000) that aimed to build an ATS system that would make newspaper articles more accessible to people who suffer from aphasia (Carroll et al., 1998). This was the first project that promoted the use of ATS for social good. The solution was envisioned as a modular system consisting of two components: an analyser and a simplifier. The analyser would use lexical tagger, morphological analyser, and parser to recognize potentially difficult words and sentences for aphasic readers. The simplifier would use rule-based syntactic simplification systems to convert passive to active voice, extract embedded clauses, and split conjoined sentences, and a lexical simplification system which would use WordNet (Vossen, 2004) for retrieving substitution candidates and the Oxford Psycholinguistic Database (Quinlan, 1992) to choose the simplest one of them. It was envisioned to evaluate the final system with a group of aphasic readers (Carroll et al., 1998). HAPPI (Helping Aphasic People Process Information) was the continuation of the PSET project, this time with the aim of developing web-based ATS system which would make web content more accessible to aphasic readers by simplifying text vocabulary and syntax (Devlin and Unthank, 2006). To the best of our knowledge, there are no publications about the evaluations of the final systems with the end users.

ATS for Readers with Low Literacy. The PorSimples (Simplification of Portuguese Text for Digital Inclusion and Accessibility) project (Aluísio and Gasperin, 2010) was a Brazilian national project (2007–2010). Its aim was to produce tools and resources for people with low literacy levels (through the assistive technology systems called FACILITA and Educational FACILITA) and authors that want to produce texts for this audience (through an authoring system called SIMPLIFICA). The FACILITA system was envisioned with the focus on summarization and syntactic simplification, i.e. sentence splitting, change of discourse markers, conversion from passive to active voice, inversion of clause order, subject-verb-object order, topicalization and de-topicalization (Watanabe et al., 2009). Educational FACILITA was a web application prototype which offered lexical elaborations and named entity labelling. A pilot study with low-literacy users reported improved text comprehension, though participants reported to be sometimes confused with the offered list of synonyms that had multiple meanings (Watanabe et al., 2010).

ATS for People with Cognitive Disabilities. Simplext (Saggion et al., 2011, 2015) was a Spanish national project (2010-2013) that proposed several modules for automatic simplification of Spanish (news) texts for people with cognitive disabilities (particularly people with Down's syndrome). The modules combined rule-based and corpus-based techniques for reducing syntactic complexity of sentences, deleting unnecessary information, performing numerical simplification, normalizing reporting verbs, and reducing lexical complexity (Saggion et al., 2015). The final ATS system was tested by 44 people with Down's syndrome by measuring reading time and text comprehension. More correct answers were obtained for simplified texts, on average, and participants positively rated the existence of such a tool that is available through different technological channels, e.g. computer, smartphone, tablet (Saggion et al., 2015).

The Able to Include project (2014–2017) was an international project that aimed to improve the quality of life of people with intellectual or developmental disabilities, dementia, or any kind of cognitive impairments.[7] Three applications were built during

---

[7] `http://able-to-include.com`

the project: ABLEsocial (media app), ABLEchat (mobility app), and KOLUMBA (ABLEmail app) by using Simplext (ATS technology), PictoText, and Text2Speech web services.[8] This project did not offer significant advances in core ATS research. Its focus was, instead, on bringing existing ATS solutions to life through freely accessible prototypes.

ATS for Readers with Autism. The FIRST project (2012–2015) was an EC-funded project under the FP7 ICT call concerning smart and personalised inclusion (Orăsan et al., 2018). It provided OpenBook, a software that can automatically identify a range of language phenomena (e.g. complex syntactic structures, complex words and phrases, ambiguous words, and metaphors) that are problematic for people with high-functioning (IQ>70) autism spectrum disorders (ASD), and replace some of them. It also offered adding illustrative pictures and concise document summaries. The tool supported three languages: English, Spanish, and Bulgarian. Acknowledging that ATS was creating a large number of errors for some texts, and that end users have low tolerance for ungrammatical and erroneous text, OpenBook was also offering powerful post-editing options so that carers can post-edit the texts before showing them to the end users (Orăsan et al., 2018). The architecture was highly modular, with separate modules for each transformation, to allow for highest level of personalization possible. The final evaluation with 243 participants with high-functioning autism from UK, Spain, and Bulgaria was performed on texts which were automatically simplified and post-edited (by the clinical teams) using OpenBook. The participants were more successful at answering multiple-choice questions about the simplified than the original texts, and human-aided text conversion time significantly decreased by using OpenBook (Orăsan et al., 2018).

2.2 ATS in Medical Domain

A special type of literacy is health literacy, defined by the European Health Literacy Consortium:

> *"Health literacy is linked to literacy and entails people's knowledge, motivation and competences to access, understand, appraise and apply health information in order to make judgements and take decisions in everyday life concerning health care, disease prevention and health promotion to maintain or improve quality of life during the life course."*

According to the European Health Literacy Survey, nearly half of all adults in the eight European countries covered by the survey were found to have low health literacy skills, which may result in less healthy choices, riskier behaviour, poorer health, less self-management and more hospitalization (WHO, 2013). Due to their potential to improve the health of people with low health literacy skills, attempts at building ATS systems focused on simplifying medical content have recently attracted noticeable attention.

In contrast to the ATS systems mentioned so far, the ATS systems in medical domain are particularly focused on translating highly specialized medical expressions into their layman variants, and thus require domain-specific resources and customized algorithms. Kloehn et al. (2018) proposed an algorithm for automatically generating explanations for difficult (medical) terms in English and Spanish. van den Bercken et al. (2019) explored the possibility of training the earlier proposed general purpose neural text simplification model (Nisioi et al., 2017) on medical parallel corpus. Shardlow and Nawaz (2019) used the general purpose neural text simplification model (Nisioi et al., 2017) augmented with the phrase table of complex-simple medical terminology to automatically simplify clinical letters in English. Cardon and Grabar (2020) used similar approach for biomedical texts in French. Emphasizing the need for high-quality simplification in medical domain, Van et al. (2020) explored the possibility of applying pretrained neural language models to the autocomplete process for sentence-level medical text simplification.

2.3 ATS Research Trends

Research on ATS in English[9] could roughly be divided in three phases: rule-based systems that focus on specific, well-defined transformations (until 2010); data-driven supervised machine learning systems trained on parallel TS data (2010-2014/2016); and neural text simplification systems (from 2015/2017 onward). The neural trends

---

[8] Details about text to pictograph translation and the accessible e-mail client can be found in (Sevens et al., 2017; Saggion et al., 2017).

[9] The vast majority of ATS research was published for English and thus the discussion here is focused on those. Most of the ATS research for other languages uses similar ideas and adapts them to the particular language.

started earlier for purely lexical simplification systems by leveraging word embeddings (Glavaš and Štajner, 2015; Paetzold and Specia, 2016). Sentence simplification systems embraced neural architectures in 2017 by using neural machine translation (Nisioi et al., 2017) and reinforcement learning (Zhang and Lapata, 2017). Those three years (2010, 2015, and 2017) can clearly be observed in Figures 1 and 2, which show the number of scholarly articles that mention the specific TS terms anywhere in the text, and the number of scholarly articles with those TS terms in their title.[10]

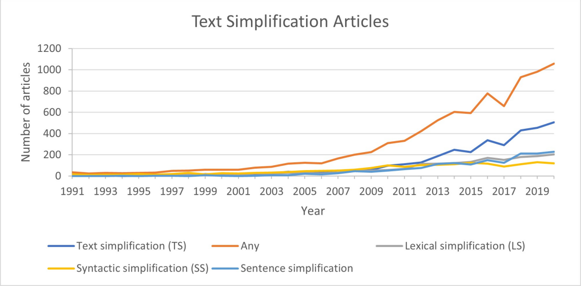

Figure 1: The number of articles in Google Scholar that mention specific text simplification terms.

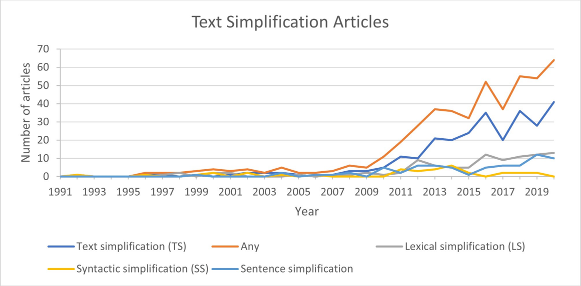

Figure 2: The number of articles in Google Scholar with specific text simplification terms in their title.

Interestingly, the number of articles mentioning syntactic simplification was steadily increasing until 2015, and then again in the period 2017–2019 (Figure 1), while the number of articles with syntactic simplification in their title was increasing until 2014, then noticeably decreased by 2016, to never increase much again (Figure 2). The number of lexical simplification (LS) papers, in turn, has been increasing since 2010, with two peaks, in 2012 and 2016. The peak in 2012 was due to the papers describing systems which participated in the SemEval-2012 English Lexical Simplification task (Specia et al., 2012), and lexical simplification endeavours in the scope of the Simplext project (Bott et al., 2012; Drndarević and Saggion, 2012a,b). Almost all lexical simplification research published in 2016 was focused on LS for non-native speakers.

So far, conceptual simplification has been tackled only through coreference resolution (Orăsan et al., 2018). Although most works mention "text simplification", they actually refer to sentence simplification. The state-of-the-art ATS systems published in top tier NLP/CL/AI conferences, e.g. (Nisioi et al., 2017; Zhang and Lapata, 2017; Surya et al., 2019; Kumar et al., 2020; Mallinson et al., 2020), all describe end-to-end systems for sentence simplification, and are not directed towards any particular simplification transformation or target population.

Very few papers proposed ATS methods that operate beyond sentence level (Glavaš and Štajner, 2013; Narayan and Gardent, 2014; Štajner and Glavaš, 2017). The research interests thus do not seem to follow the reported text simplification needs (see Table 1 in Section 1.1).

3 Challenges in ATS for Social Good

In spite of the evident increased interest for ATS in the last few years, and the obvious benefits it could bring to the society, no commercial systems have been offered so far, proving thus that automatic text simplification is a very challenging task.

The main reason might lie in the multidisciplinary nature of the task, i.e. the need for combining linguistics, psycholinguistics, computer science, and, to be used in real-world applications, advanced software engineering. Only large multidisciplinary teams can ensure solutions that are technically-advanced and, at the same time, truly help the end users. The absence of such multidisciplinary research leads to usage of non-optimal resources and evaluation procedures which do not involve end users, thus steering the field away from any real-world applications.

3.1 Resources

The number of manually simplified texts increases daily due the number of websites that provide easy-to-read materials (Section 1.2). Unfortunately, those materials and their original (non-simplified) versions, are usually not made available for neither research nor commercial purposes. The exceptions are the English and Spanish Newsela corpora,

---

[10] We excluded the patents and citations from this search, and looked for the exact matches of the TS terms. All returned titles were additionally manually checked.

Dataset	Language	Quality	Target	Domain	Articles×levels
Britanica (Barzilay and Elhadad, 2003)	EN	●	Children	Encyclopedia	114×2
LiteracyNet (Feng et al., 2009)	EN	●	Low literacy	News	115×2
Zero Hora (Caseli et al., 2009)	PT(BR)	○	Children & Low literacy	News	104×3
Dsim (Klerke and Søgaard, 2012)	DA	●	Reading impaired & L2	News telegrams	3701×2
Wikipedia (Kauchak, 2013)	EN	○	Children & L2	Encyclopedia	≈60,000×2
Newsela (Xu et al., 2015)	EN	●	Children & L2	News	1,130×5
Newsela (Štajner et al., 2015)	ES	●	Children & L2	News	≈250*5
Simplext (Saggion et al., 2015)	ES	●	Intellectual disability	News	200×2
Terence (Brunato et al., 2015)	IT	●	Poor comprehenders	Children stories	32×2
Teacher (Brunato et al., 2015)	IT	○	L2	Literature & Handbooks	24×2
NewsWebEasy (Goto et al., 2015)	JP	○	L2	News	490×2
OneStopEng. (Vajjala and Lučić, 2018)	EN	●	L2	News	189×3
Alector (Gala et al., 2020)	FR	●	Children & L2	Literature & Science	79×2

Table 3: Parallel (comparable in the case of Wikipedia) document-level text simplification datasets (● signifies professional simplification, and ○ non-professional simplification following provided guidelines).

Dataset	Language	Quality	Alignment	Type	Domain	Sentence pairs
Enc. Brit. (Barzilay and Elhadad, 2003)	EN	●	Automatic	all	Encyclopedia	601
PorSimples (Specia, 2010)	PT(BR)	○	Controlled	all	News & Science	4,483
PWKP (Zhu et al., 2010)	EN	○	Automatic	all	Wikipedia	≈108,000
EW-SEW v1.0 (Coster and Kauchak, 2011b)	EN	○	Automatic	all	Wikipedia	≈137,000
EW-SEW v2.0 (Kauchak, 2013)	EN	○	Automatic	all	Wikipedia	≈167,000
EW-SEW (Hwang et al., 2015)	EN	○	Automatic	all	Wikipedia	≈280,000
Newsela (Štajner et al., 2015)	EN	●	Automatic	all	News	≈480,000
NewsWebEasy (Goto et al., 2015)	JP	●	Manual+Auto	all	News	13,386
Simplext (Saggion et al., 2015)	ES	●	Manual	all	News	925
PaCCSS-IT (Brunato et al., 2016)	IT	○	Automatic	structural	Web texts	≈63,000
Newsela (Štajner et al., 2018)	ES	●	Automatic	all	News	Not specified
PorSimplesSent (Leal et al., 2018)	PT(BR)	○	Controlled	all	News & Science	4,888

Table 4: Comparable sentence-level text simplification datasets (‘controlled’ signifies that the alignment was automatic but using the original editing tool, thus corresponding close to a manual alignment).

which are available upon request for research purposes (the portion until 2016).[11]

The largest freely available TS corpora up-to-date are those based on matching the original English Wikipedia articles with the articles of the same title in Simple English Wikipedia, the EW-SEW corpora. They are available in several versions, as document-aligned (Kauchak, 2013)[12] and sentence-aligned (Zhu et al., 2010; Coster and Kauchak, 2011b; Kauchak, 2013; Hwang et al., 2015). Apart from not representing a truly parallel corpus but rather just comparable, as the articles in Simple English Wikipedia might have been written completely independently from the articles with the same title in the original English Wikipedia, many objections have been raised regarding the quality of simplifications (Amancio and Specia, 2014; Štajner et al., 2015; Xu et al., 2015; Štajner et al., 2020).

The English and Spanish Newsela corpora (Xu et al., 2015; Štajner et al., 2018), in contrast, provide parallel texts on several complexity levels, manually simplified under strict quality control (Xu et al., 2015).

The other available parallel (document-aligned) TS corpora are significantly smaller and do not provide enough material for training ATS models (Table 3). As their texts were simplified for different target populations, they cannot be combined to make larger training datasets for any specific target population, as it is known that there are no universal native simplified–language speaker (Siddharthan, 2014). Nevertheless, those smaller TS corpora could be used for evaluation of ATS systems aimed at particular target populations.

The sentence-aligned TS corpora, apart from those obtained from Newsela and EW-SEW, are still very scarce and limited in their size (Table 4).

The main problem with most sentence-aligned TS corpora is that they were automatically aligned

---

[11] `https://newsela.com/data/`

[12] `https://cs.pomona.edu/~dkauchak/simplification/`

using various sentence similarity measures with high similarity thresholds. This resulted in high number of pairs of identical, or nearly identical, sentences which, if used for training ATS systems, lead to conservative systems that rarely suggest any changes to the original sentence (Štajner and Saggion, 2015). The newer alignment methods (Hwang et al., 2015; Štajner et al., 2017; Paetzold et al., 2017), offer two types of sentence pairs: full matches and partial matches. In NMT-based TS (Štajner and Nisioi, 2018), the EW-SEW dataset (Hwang et al., 2015) resulted in a model with similarly high percentage of changed sentences as the model trained on Newsela sentence-aligned corpus (Štajner et al., 2017). Those results indicate that the EW-SEW dataset has a potential to be used for training state-of-the-art ATS models, if the sentences are carefully matched and filtered.

For building training datasets for ATS in medical domain, van den Bercken et al. (2019) and Van et al. (2020) used the EW-SEW dataset by retaining only those sentence pairs in which the original sentences contain certain number of medical terms.

One of the biggest challenges in ATS is how to collect a parallel dataset of truly strong paraphrases which are necessary for training ATS systems for people that require higher levels of simplification (e.g. people with cognitive or reading disabilities, and people with low literacy levels).[13] Such sentence pairs are still rarely correctly extracted with automatic alignment methods (Štajner et al., 2018).

The sentence-aligned TS corpora in Brazilian Portuguese, the PorSimples (Specia, 2010) and PorSimplesSent[14] (Leal et al., 2018), compiled under the PorSimples project, are the only ones that did not lose the strong paraphrase pairs due to automatic sentence-alignment, as manual simplifications were performed using a special editing tool which recorded the exact sentence alignments (Caseli et al., 2009). Both datasets are still very small to allow for automatically learning strong paraphrases solely from them. Nevertheless, they demonstrate that by providing editing tools for manual simplification, it is possible to automatically extract strong paraphrases from manually simplified corpora. This approach could thus be used in collaboration with various associations and content editors that perform professional manual simplifications for various target populations, to compile large, high-quality training datasets for building ATS systems aimed for those particular end users.

3.2 Evaluation

Ideally, ATS systems should be evaluated for the output quality, as well as for its usability, i.e. whether or not they make texts easier to read and understand for the particular reader.

3.2.1 Quality

The quality of ATS output should be evaluated for its grammaticality (and naturalness), meaning preservation (ensuring that, albeit some necessary loss of nuances, the core meaning remains unchanged), and preservation of text coherence and cohesion. The last is particularly important for systems that perform syntactic simplifications and sentence splitting which often break the cohesion links and make texts more complex at a discourse level (Siddharthan, 2003, 2006). Meaning preservation is also often compromised, both by syntactic simplification and by lexical simplification. Some unsupervised LS systems (Glavaš and Štajner, 2015; Paetzold and Specia, 2016) particularly suffer from this issue due to the use of word embeddings for synonyms retrieval, which often returns antonyms that appear in similar contexts (Glavaš and Štajner, 2015). Grammatical errors are less common, and can be easily corrected manually in a post-editing step. Manual correction of change in meaning, in turn, can sometimes require more time than simplifying from scratch. Coherence problems in automatically simplified texts are the most expensive, as they might require heavy restructuring of the whole text.

To measure the quality of the automatically simplified texts, Štajner and Glavaš (2017) proposed to measure the post-editing time needed to restore the text's grammaticality, original meaning and coherence. The output quality of the sentence simplification systems is, in contrast, evaluated only for grammaticality and meaning preservation, usually using a five-point Likert scale by native or non-native speakers with high literacy levels.

3.2.2 Usability

Usability of ATS systems should ideally be evaluated by measuring reading time and text com-

---

[13] Strong paraphrases are those paraphrases where vocabulary and sentence structure were changed beyond isolated lexical simplifications, straightforward sentence splitting and passive to active voice conversion. For examples, see Tables 1, 2, and 5 in (Štajner et al., 2018).

[14] `https://github.com/sidleal/porsimplessent`

prehension by end users, usually via eye-tracking methods and comprehension tests. This type of evaluation requires significant time and special training for preparing, collecting and interpreting the data. Preparation of comprehension tests requires experts, and eye-tracking methods require special equipment and trained humans to analyse and interpret the results. The access to vulnerable populations (e.g. people with autism or cognitive disabilities) usually requires special permissions and training. Another obstacle with vulnerable populations is that laboratory conditions and use of comprehension questionnaires provide unnatural reading scenario that sometimes leads to unreliable results (Yaneva et al., 2015, 2016a).

For all those reasons, ATS systems are rarely evaluated for their usability in this way. Notable exceptions are: the eye-tracking analyses of simplification strategies for dyslexic readers (Rello et al., 2013b,c,a), and readers with autism spectrum disorders (Yaneva et al., 2016b), the comprehension-based evaluations of the Simplext system by people with Down's syndrome (Saggion et al., 2015), the OpenBook software by people with ASD (Orăsan et al., 2018), and several ATS systems by non-native low-pay workers (Angrosh et al., 2014).

3.2.3 Simplicity

The usability evaluation with end users requires long time and special expertise. To obtain the results faster, in ATS research papers, it became a common strategy to, instead, evaluate a small portion of automatically simplified sentences for their simplicity using a Likert scale by evaluators who are not the intended end users.

In some studies the evaluators are native speakers, e.g. (Yatskar et al., 2010; Baeza-Yates et al., 2015; Saggion et al., 2015), in some non-native speakers, e.g. (Coster and Kauchak, 2011a; Wubben et al., 2012; Glavaš and Štajner, 2013, 2015; Paetzold and Specia, 2016), in some a mixture of the two (Angrosh et al., 2014; Kumar et al., 2020), and in some not specified, e.g. (Specia, 2010; Xu et al., 2016; Mallinson et al., 2020; Alva-Manchego et al., 2020a). In some of those studies, the evaluators are students, i.e. readers with high literacy levels, and in others, Amazon Mechanical Turk workers whose literacy level is usually unknown. Štajner and Nisioi (2018) found that evaluators with high literacy levels rarely notice any difference in complexity of the original and automatically simplified sentences. Other studies have shown that native and non-native speakers differently perceive sentence complexity (Štajner, 2018) and have different lexical simplification needs (Yimam et al., 2017a,b, 2018). Therefore, a direct comparison of the ATS systems proposed in studies where simplicity was evaluated by different type of evaluators is not possible.

Apart from the type of evaluators, the evaluation procedures used across the ATS community differ also by evaluation type (absolute or relative) and scale (e.g. 0/1, 1–3, 1–5), thus hindering the possibility to directly compare the results reported in different studies (Štajner, 2018). In the absolute evaluation, evaluators are presented with one sentence at the time and asked to rate its simplicity on a certain scale. If provided with different variants of the same sentence one after another, depending on the guidelines, evaluators may sometimes emphasize even the small differences between them by giving them different scores. Similar may happen in relative evaluation of simplicity gain, if evaluators are explicitly asked to count the number of successful lexical and syntactic paraphrases between the two sentences (original and simplified), as in the work of Xu et al. (2016). These approaches are thus helpful if the goal is to spot even the slightest differences between several models. If the goal is, in contrast, to estimate the usability and helpfulness of the systems in a real-world scenario, in a text comprehension task, then the approach in which evaluators are presented with pairs of sentences (original and simplified, without specifying which one is which) and asked whether or not they find one of them easier to understand, proposed by Nisioi et al. (2017), might be better suited.

To enable a direct comparison of results reported in different ATS studies, a standardized evaluation procedure across the field is needed. To establish the right evaluation procedure, it would be necessary to first investigate if some particular group of evaluators can be a good proxy for assessing the complexity of texts/sentences for intended target population. In the case that none of those groups satisfies the condition, the evaluators would have to be provided with detailed evaluation guidelines and trained for spotting potential obstacles for the intended end users. Furthermore, it would be necessary to systematically investigate which evaluation type and scale give the most reliable assessment of potential usability of the ATS systems in a real-world scenario.

3.2.4 Automatic Evaluation

With the goal of offering a more standardized way of measuring ATS systems' performances, several test sets have been compiled for lexical simplification in English (De Belder and Moens, 2012; Specia et al., 2012; Horn et al., 2014; Paetzold and Specia, 2015). De Belder and Moens (2012) propose a test set of 430 sentences, each with one marked word and the list of words that can replace it, ordered by their difficulty. Similarly, Specia et al. (2012) compiled a test set of 1710 sentences, each with one marked word and four substitution candidates ordered by their difficulty. Horn et al. (2014) compiled a set of 500 sentences from English Wikipedia, each with one marked word and 50 substitution candidates collected via crowdsourcing using Amazon Mechanical Turk.

All these datasets rely on suggestions and rankings of either non-native speakers (usually with high education levels), or neurotypical native speakers. Therefore, without systematic investigation of how well they reflect simplification needs of other target populations, it is not clear to which extent, if at all, they could be used to evaluate the performances of ATS systems aimed at them.

For automatic evaluation of sentence splitting modules for English, two datasets are currently available: WebSplit (Narayan et al., 2017) and WikiSplit (Botha et al., 2018). Both datasets were automatically extracted and aligned; the WebSplit from the WebNLG corpus (Gardent et al., 2017), and WikiSplit from Wikipedia edit histories. Recently, a crowdsourced dataset with multiple human simplification suggestions (both syntactic and lexical) for 2,359 original English Wikipedia sentences has been released (Alva-Manchego et al., 2020a). So far, none of them has been tested for how well it could evaluate the usability of real-world ATS systems for any particular target population.

Many ATS systems have been automatically evaluated using the Flesch-Kincaid Grade Level (FKGL) (Kincaid et al., 1975), MT-inspired evaluation metrics, e.g. BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), TER (Snover et al., 2009), or more recently proposed TS-specific metrics: SARI (Xu et al., 2016) and SAMSA (Sulem et al., 2018b). None of these metrics, however, can replace the necessary manual evaluation of grammaticality and meaning preservation (at a sentence level), or cohesion and coherence (at a discourse level). Traditional readability metrics such as FKGL were proposed for human-generated texts and are completely oblivious of meaning, i.e. short, meaningless sentences would score very well on those. The use of BLEU for TS evaluation has been reported to have many drawbacks if used for ATS (Štajner et al., 2015; Xu et al., 2016; Sulem et al., 2018a). Some other MT-based evaluation metrics have shown good correlations with human judgements of grammaticality and meaning preservation (Štajner et al., 2014; Popović and Štajner, 2016), but they all favor simplifications that are close to the original sentence (i.e. offer very few changes). SARI, in turn, favors simplifications that are as different from original as possible, thus often rewarding simple sentences which might have significantly altered the meaning.

4 Conclusion and Outlook

In this study, the potential for using ATS for social good and main challenges in the field to achieve that goal were emphasized: the lack of large high-quality TS datasets for training ATS systems for particular target populations, and the lack of standardized evaluation procedures for estimating the usability potential of the proposed ATS systems in real-world scenarios. Due to those, it is not clear if the recently proposed neural ATS systems, published in top tier NLP/CL/AI conferences, e.g. (Nisioi et al., 2017; Zhang and Lapata, 2017; Surya et al., 2019; Kumar et al., 2020; Mallinson et al., 2020), represent a real step forward toward using ATS for social good.

To address the above-mentioned challenges, it seems necessary to: (1) establish multidisciplinary collaborations with associations and content editors which have experience in simplifying texts for specific groups of end users; and (2) prioritize work on highly modular ATS systems, which allow for easy customization of components according to the simplification needs of particular groups of users. Multidisciplinary collaborations would enable compiling of larger high-quality datasets for various target populations, better understanding of which ATS modules require higher prioritization, and the possibility for testing the ATS systems with the real end users.

Acknowledgements

I would like to thank the three anonymous reviewers for their valuable comments and suggestions.

Ethics/Impact Statement

As this is a special theme paper and it does not propose any novel systems, the main focus of this section is the discussion about the general ethical considerations for automatic text simplification and its impact on the society.

Intended Use

Automatic text simplification (ATS) has strong potential to improve social inclusion for many people who are otherwise marginalized in one way or another. While many large-scale projects proved this, the last several years have been dominated by ATS research which does not seem to lead towards practical applications of ATS in the society. The goal of this paper is to raise the awareness among researchers about the original motivations for ATS and its potential for being used for social good, as well as to point out the main challenges in the field, and suggest ways forward.

Failure Modes

As mentioned in Section 3.2, ATS systems often produce ungrammatical sentences, changes to the original meaning, and break the text coherence. For this reason, to avoid potential unintended harm, it is advisable to add post-editing capabilities in ATS systems, as it was done in the OpenBook software (Orăsan et al., 2018). The unintended changes in meaning produced by ATS systems would be especially dangerous in legal and medical domains.

Biases

As all machine learning (ML) systems, ML-based ATS systems can suffer from algorithmic biases. If trained on data that represents manual simplifications performed by trained human editors, the system might inherit the biases and preferences of the editor. If trained on automatically sentence-aligned data, the system might learn transformations which might rather reflect the alignment methods than the actual simplification methods used in the prealigned corpora.

Misuse Potential

As any technological advancement, ATS could also be misused, e.g. by learning to make text more complex with the intention of providing linguistically complex texts to hide non-desirable facts, as it is known that high percentage of people have problems to understand complex texts (Section 1.1), or by unintentionally altering the original meaning thus offering incorrect information.

Potential Harm to Vulnerable Populations

People with low literacy levels, or any kind of reading or cognitive impairments, which cannot understand original texts and thus fully rely on the simplified versions, can potentially be harmed by being provided with incorrect information due to the use of premature ATS systems where the output is not manually checked for meaning preservation. Therefore, it is important that any ATS system intended for those populations offers post-editing capabilities and that its output passes a rigorous manual check for meaning preservation.

Furthermore, if provided by automatically simplified learning material, which was not carefully manually checked for its grammaticality and natural soundness of sentences, children and language learners might learn incorrect word forms or sentences structures.

Computing Time

The latest trends in ATS research steered the systems towards very computationally expensive (and environmentally unfriendly) neural architectures. As the current neural state-of-the-art ATS systems are still far behind the traditional modular rule-based ATS systems (especially for syntactic simplification), many research groups and organizations need numerous attempts with neural architectures even just to be able to publish a paper in a top-tier conference. The current best performing neural lexical simplification systems (Glavaš and Štajner, 2015; Paetzold and Specia, 2016, 2017a; Qiang et al., 2020a,b) perform better than the state-of-the-art non-neural lexical simplification systems (Glavaš and Štajner, 2015; Paetzold and Specia, 2017b; Qiang et al., 2020b). Nevertheless, due to using heavy resources, such as BERT (Devlin et al., 2018) and PPDB (Ganitkevitch et al., 2013), or their multiple operations with word embeddings, they require significant computational power and are too slow to be used in real-world applications.

References

Suha S. Al-Thanyyan and Aqil M. Azmi. 2021. Automated text simplification: A survey. *ACM Comput. Surv.*, 54(2).

Sandra Maria Aluísio and Caroline Gasperin. 2010. Fostering Digital Inclusion and Accessibility: The

PorSimples Project for Simplification of Portuguese Texts. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020a. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679, Online. Association for Computational Linguistics.

Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. 2020b. Data-driven sentence simplification: Survey and benchmark. *Computational Linguistics*, 46(1):135–187.

Marcelo Adriano Amancio and Lucia Specia. 2014. An Analysis of Crowdsourced Text Simplifications . In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 123–130.

Mandya Angrosh, Tadashi Nomoto, and Advaith Siddharthan. 2014. Lexico-syntactic text simplification and compression with typed dependencies. In *Proceedings the 25th International Conference on Computational Linguistics*, COLING, pages 1996–2006, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ricardo A. Baeza-Yates, Luz Rello, and Julia Dembowski. 2015. CASSA: A Context-Aware Synonym Simplification Algorithm. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1380–1385, Denver, Colorado, USA.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laurens van den Bercken, Robert-Jan Sips, and Christoph Lofi. 2019. Evaluating neural text simplification in the medical domain. In *The World Wide Web Conference*, WWW '19, page 3286–3292, New York, NY, USA. Association for Computing Machinery.

Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from Wikipedia edit history. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 732–737, Brussels, Belgium. Association for Computational Linguistics.

Stefan Bott, Luz Rello, Biljana Drndarević, and Horacio Saggion. 2012. Can Spanish Be Simpler? LexSiS: Lexical Simplification for Spanish. In *Proceedings of the 24th International Conference on Computational Linguistics*, COLING, pages 357–374.

Dominique Brunato, Andrea Cimino, Felice Dell'Orletta, and Giulia Venturi. 2016. PaCCSS-IT: A Parallel Corpus of Complex-Simple Sentences for Automatic Text Simplification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 351–361, Austin, Texas. Association for Computational Linguistics.

Dominique Brunato, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2015. Design and annotation of the first Italian corpus for text simplification. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 31–41, Denver, Colorado, USA. Association for Computational Linguistics.

Rémi Cardon and Natalia Grabar. 2020. French biomedical text simplification: When small and precise helps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 710–716, Barcelona, Spain (Online). International Committee on Computational Linguistics.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical Simplification of English Newspaper Text to Assist Aphasic Readers. In *Proceedings of the AAAI'98 Workshop on Integrating AI and Assistive Technology*, pages 7–10.

Helena M. Caseli, Tiago F. Pereira, Lucia Specia, Thiago A. S. Pardo, Caroline Gasperin, and Sandra M. Aluísio. 2009. Building a Brazilian Portuguese parallel corpus of original and simplified texts. In *10th Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2009)*.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *16th International Conference on Computational Linguistics*, pages 1041–1044.

R. Chandrasekar and B. Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10:183–190.

Michael Cooper, Loretta Guarino Reid, Gregg Vanderheiden, and Ben Caldwell. 2010. Understanding WCAG 2.0. A guide to understanding and implementing Web Content Accessibility Guidelines 2.0. World Wide Web Consortium (W3C).

William Coster and David Kauchak. 2011a. Learning to Simplify Sentences Using Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–9.

William Coster and David Kauchak. 2011b. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human*

Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, pages 665–669.

David Crystal. 1987. *The Cambridge Encyclopedia of Language*. Cambridge University Press.

Jan De Belder and Marie-Francine Moens. 2012. A Dataset for the Evaluation of Lexical Simplification. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7182 of *Lecture Notes in Computer Science*, pages 426–437. Springer Berlin Heidelberg.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP Workshop on Statistical Machine Translation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Siobhan Devlin and Gary Unthank. 2006. Helping Aphasic People Process Online Information. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 225–226, New York, NY, USA. ACM.

Biljana Drndarević and Horacio Saggion. 2012a. Reducing Text Complexity through Automatic Lexical Simplification: an Empirical Study for Spanish. *Procesamiento del Lenguaje Natural*, 49:13–20.

Biljana Drndarević and Horacio Saggion. 2012b. Towards Automatic Lexical Simplification in Spanish: An Empirical Study. In *Proceedings of the First Workshop on Predicting and Improving Text Readability for Target Reader Populations*, PITR, pages 8–16, Montréal, Canada. Association for Computational Linguistics.

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. 2009. Cognitively Motivated Features for Readability Assessment. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 229–237, Stroudsburg, PA, USA. Association for Computational Linguistics.

Geert Freyhoff, Gerhard Hess, Linda Kerr, Bror Tronbacke, and Kathy Van Der Veken. 1998. *Make it Simple, European Guidelines for the Production of Easy-toRead Information for People with Learning Disability*. ILSMH European Association, Brussels.

Núria Gala, Anaïs Tack, Ludivine Javourey-Drevet, Thomas François, and Johannes C. Ziegler. 2020. Alector: A parallel corpus of simplified French texts with alignments of misreadings by poor and dyslexic readers. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1353–1361, Marseille, France. European Language Resources Association.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Goran Glavaš and Sanja Štajner. 2013. Event-Centered Simplification of News Stories. In *Recent Advances in Natural Language Processing, RANLP 2013, 9-11*, pages 71–78.

Goran Glavaš and Sanja Štajner. 2015. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In *Proceedings of the ACL&IJCNLP 2015 (Volume 2: Short Papers)*, pages 63–68.

Isao Goto, Hideki Tanaka, and Tadashi Kumano. 2015. Japanese news simplification: Task design, data set construction, and analysis of simplified text. In *Proceedings of Machine Translation Summit XV, Vol. 1: MT Researchers' Track*, pages 17–31, Miami, FL.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 458–463.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning Sentences from Standard Wikipedia to Simple Wikipedia. In *Proceedings of NAACL&HLT*, pages 211–217.

Joyce Karreman, Thea van der Geest, and Esmee Buursink. 2007. Accessible website content guidelines for users with intellectual disabilities. *Journal of Applied Research in Intellectual Disabilities*, 20:510–518.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546. ACL.

Peter J. Kincaid, Robert P. Fishburne, Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas for Navy enlisted personnel. *Research Branch Report 8-75*.

Sigrid Klerke and Anders Søgaard. 2012. DSim, a Danish parallel corpus for text simplification. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 4015–4018, Istanbul, Turkey. European Language Resources Association (ELRA).

Nicholas Kloehn, Gondy Leroy, David Kauchak, Yang Gu, Sonia Colina, Nicole P Yuan, and Debra Revere. 2018. Improving consumer understanding of medical text: Development and validation of a new subsimplify algorithm to automatically generate term explanations in english and spanish. *J Med Internet Res*, 20(8):e10779.

Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7918–7928, Online. Association for Computational Linguistics.

Sidney Evaldo Leal, Magali Sanches Duran, and Sandra Maria Aluísio. 2018. A nontrivial sentence corpus for the task of sentence readability assessment in portuguese. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, page 401–413, Santa Fe, New Mexico, USA.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2020. Zero-shot crosslingual sentence simplification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5109–5126, Online. Association for Computational Linguistics.

Mencap. 2002. *Am I making myself clear? Mencap's guidelines for accessible writing.*

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 435–445.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 606–616, Copenhagen, Denmark. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 85–91.

OECD. 2013. OECD Skills Outlook 2013: First Results from the Survey of Adult Skills. Technical report, OECD Publishing.

Charles Kay Ogden. 1937. *Basic English: A General Introduction with Rules and Grammar*. Paul Treber, London.

Constantin Orăsan, Richard Evans, and Ruslan Mitkov. 2018. *Intelligent Text Processing to Help Readers with Autism*, pages 713–740. Springer International Publishing, Cham.

Gustavo Paetzold and Lucia Specia. 2015. LEXenstein: A Framework for Lexical Simplification. In *Proceedings of the ACL-IJCNLP 2015 System Demonstrations*, pages 85–90, Beijing, China.

Gustavo Paetzold and Lucia Specia. 2017a. Lexical simplification with neural ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 34–40, Valencia, Spain. Association for Computational Linguistics.

Gustavo H. Paetzold, Fernando Alva-Manchego, and Lucia Specia. 2017. MASSAlign: Alignment and Annotation of Comparable Documents. In *The Companion Volume of the IJCNLP 2017 Proceedings: System Demonstrations*, pages 1–4.

Gustavo H. Paetzold and Lucia Specia. 2017b. A survey on lexical simplification. *Journal of Artificial Intelligence Research*, 60(1):549–593.

Gustavo Henrique Paetzold and Lucia Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the 30th AAAI.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL.*

PlainLanguage. 2011. Federal plain language guidelines.

Maja Popović and Sanja Štajner. 2016. Machine translation evaluation metrics for quality assessment of automatically simplified sentences. In *Proceedings of the LREC Workshop on Quality Assessment for Text Simplification (QATS).*

Jipeng Qiang, Yun Li, Zhu Yi, Yunhao Yuan, and Xindong Wu. 2020a. Lexical simplification with pretrained encoders. *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 8649—-8656.

Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2020b. Lsbert: A simple framework for lexical simplification.

Philip Quinlan. 1992. *The Oxford Psycholinguistic Database*. Oxford University Press.

Luz Rello, Ricardo A. Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013a. Simplify or help?: text simplification strategies for people with dyslexia. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A, Rio de Janeiro, Brazil.

Luz Rello, Ricardo A. Baeza-Yates, Laura Dempere-Marco, and Horacio Saggion. 2013b. Frequent Words Improve Readability and Short Words Improve Understandability for People with Dyslexia. In *Proceedings of the International Conference on Human-Computer Interaction (Part IV)*, INTERACT, pages 203–219, Cape Town, South Africa.

Luz Rello, Susana Bautista, Ricardo A. Baeza-Yates, Pablo Gervás, Raquel Hervás, and Horacio Saggion. 2013c. One Half or 50%? An Eye-Tracking Study of Number Representation Readability. In *Proceedings of the International Conference on Human-Computer Interaction (Part IV)*, INTERACT, pages 229–245, Cape Town, South Africa.

Horacio Saggion. 2017. *Automatic text simplification*. Morgan & Claypool Publishers.

Horacio Saggion, Daniel Ferrés, Leen Sevens, Ineke Schuurman, Marta Ripollés, and Olga Rodríguez. 2017. Able to read my mail: An accessible e-mail client with assistive technology. In *Proceedings of the 14th Web for All Conference on The Future of Accessible Work*, W4A '17, New York, NY, USA. Association for Computing Machinery.

Horacio Saggion, Elena Gómez Martínez, Alberto Anula, Lorena Bourg, and Esteban Etayo. 2011. Text Simplification in Simplext: Making Text More Accessible. *Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural*, 47:341–342.

Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarević. 2015. Making It Simplext: Implementation and Evaluation of a Text Simplification System for Spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):14.

Leen Sevens, Vincent Vandeghinste, Ineke Schuurman, and Frank Van Eynde. 2017. Simplified text-to-pictograph translation for people with intellectual disabilities. In *Natural Language Processing and Information Systems*, pages 185–196, Cham. Springer International Publishing.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014*, 4(1).

Matthew Shardlow and Raheel Nawaz. 2019. Neural text simplification of clinical letters with a domain specific phrase table. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 380–389, Florence, Italy. Association for Computational Linguistics.

Advaith Siddharthan. 2003. *Syntactic simplification and text cohesion*. Ph.D. thesis, University of Cambridge, UK.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language & Computation*, 4(1):77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *International Journal of Applied Linguistics*, 165(2):259–298.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268, Athens, Greece.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *Proceedings of the 9th international conference on Computational Processing of the Portuguese Language (PROPOR)*, volume 6001 of *Lecture Notes in Computer Science*, pages 30–39. Springer Berlin Heidelberg.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. SemEval-2012 task 1: English Lexical Simplification. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, SemEval, pages 347–355, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. BLEU is not suitable for the evaluation of text simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744, Brussels, Belgium. Association for Computational Linguistics.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Semantic structural evaluation for text simplification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 685–696, New Orleans, Louisiana. Association for Computational Linguistics.

Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. 2019. Unsupervised neural text simplification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2058–2068, Florence, Italy. Association for Computational Linguistics.

UN. 2020. Article 9 – accessibility. In *United Nations: Convention on the Rights of Persons with Disabilities (CPRD)*. Accessed: 2020-02-01.

Sowmya Vajjala and Ivana Lučić. 2018. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana. Association for Computational Linguistics.

Hoang Van, David Kauchak, and Gondy Leroy. 2020. AutoMeTS: The autocomplete for medical text simplification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1424–1434, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Piek Vossen, editor. 2004. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*, volume 17. Oxford Univ Press.

Sanja Štajner. 2018. How to make troubleshooting simpler? assessing differences in perceived sentence simplicity by native and non-native speakers. In *In Proceedings of the second LREC workshop on Improving Social Inclusion: Tools, Methods and Resources (ISI-NLP 2), Miyazaki, Japan*.

Sanja Štajner, Hannah Bechara, and Horacio Saggion. 2015. A Deeper Exploration of the Standard PB-SMT Approach to Text Simplification and its Evaluation. In *Proceedings of ACL&IJCNLP (Volume 2: Short Papers)*, pages 823–828.

Sanja Štajner, Marc Franco-Salvador, Simone Paolo Ponzetto, Paolo Rosso, and Heiner Stuckenschmidt. 2017. Sentence alignment methods for improving text simplification systems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 97–102.

Sanja Štajner, Marc Franco-Salvador, Paolo Rosso, and Simone Paolo Ponzetto. 2018. CATS: A Tool for Customized Alignment of Text Simplification Corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Sanja Štajner and Goran Glavaš. 2017. Leveraging event-based semantics for automated text simplification. *Expert Systems with Applications*, 82:383–395.

Sanja Štajner, Ruslan Mitkov, and Horacio Saggion. 2014. One Step Closer to Automatic Evaluation of Text Simplification Systems. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR) at EACL*.

Sanja Štajner and Sergiu Nisioi. 2018. A detailed evaluation of neural sequence-to-sequence models for in-domain and cross-domain text simplification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Sanja Štajner, Sergiu Nisioi, and Ioana Hulpuș. 2020. CoCo: A tool for automatically assessing conceptual complexity of texts. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7179–7186, Marseille, France. European Language Resources Association.

Sanja Štajner and Horacio Saggion. 2015. Translating from Original to Simplified Sentences using Moses: When does it Actually Work? In *Proceedings of RANLP 2015*, pages 611–617.

W3C. 2008. *Web Content Accessibility Guidelines (WCAG) 2.0*.

Willian M. Watanabe, Arnaldo Candido Jr., Marcelo A. Amâncio, Matheus De Oliveira, Thiago A. S. Pardo, Renata P. M. Fortes, and Sandra M. Aluísio. 2010. Adapting web content for low-literacy readers by using lexical elaboration and named entities labeling. *New Review of Hypermedia and Multimedia*, 16(3):303–327.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: Reading assistance for low-literacy readers. In *Proceedings of the 27th ACM International Conference on Design of Communication*, SIGDOC '09, page 29–36, New York, NY, USA. Association for Computing Machinery.

WHO. 2013. *Health literacy: The solid facts*.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence Simplification by Monolingual Machine Translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL, pages 1015–1024, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Associaton for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Victoria Yaneva, Irina Temnikova, and R. Mitkov. 2016a. A corpus of text data and gaze fixations from autistic and non-autistic adults. In *LREC*.

Victoria Yaneva, Irina Temnikova, and Ruslan Mitkov. 2015. Accessible texts for autism: An eye-tracking study. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers Accessibility*, ASSETS '15, page 49–57, New York, NY, USA. Association for Computing Machinery.

Victoria Yaneva, Irina Temnikova, and Ruslan Mitkov. 2016b. A Corpus of Text Data and Gaze Fixations from Autistic and Non-autistic Adults. In *Proceedings of the 10th edition of the Language Resources and Evaluation Conference*, LREC.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 365–368, Stroudsburg, PA, USA. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017a. Cwig3g2 - complex word identification task across three text genres and two user groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407. Asian Federation of Natural Language Processing.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017b. Multilingual and Cross-Lingual Complex Word Identification. In *Proceedings of RANLP*, pages 813–822, Varna, Bulgaria.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361.

A Neural Edge-Editing Approach for Document-Level Relation Graph Extraction

Kohei Makino **Makoto Miwa** **Yutaka Sasaki**
Toyota Technological Institute
2-12-1 Hisakata, Tempaku-ku, Nagoya, 468-8511, Japan
`{sd21505, makoto-miwa, yutaka.sasaki}@toyota-ti.ac.jp`

Abstract

In this paper, we propose a novel edge-editing approach to extract relation information from a document. We treat the relations in a document as a relation graph among entities in this approach. The relation graph is iteratively constructed by editing edges of an initial graph, which might be a graph extracted by another system or an empty graph. The way to edit edges is to classify them in a close-first manner using the document and temporally-constructed graph information; each edge is represented with a document context information by a pretrained transformer model and a graph context information by a graph convolutional neural network model. We evaluate our approach on the task to extract material synthesis procedures from materials science texts. The experimental results show the effectiveness of our approach in editing the graphs initialized by our in-house rule-based system and empty graphs.[1]

1 Introduction

Relation extraction (RE), the task to predict relations between pairs of given entities from literature, is an important task in natural language processing. While most existing work focused on sentence-level RE (Zeng et al., 2014), recent studies extended the extraction to the document level since many relations are expressed across sentences (Christopoulou et al., 2019; Nan et al., 2020).

In document-level RE, models need to deal with relations among multiple entities over a document. Several document-level RE methods construct a document-level graph, which is built on nodes of words or other linguistic units, to capture document-level interactions between entities (Christopoulou et al., 2019; Nan et al., 2020). However, such methods do not directly consider interactions among relations in a document, while such relations are often dependent on each other, and other relations can be considered as important contexts for a relation.

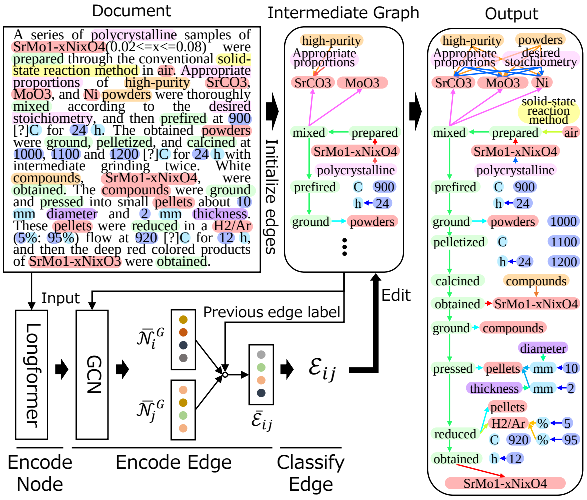

A series of polycrystalline samples of SrMo1-xNixO4(0.02<=x<=0.08) were prepared through the conventional solid-state reaction method in air. Appropriate proportions of high-purity SrCO3, MoO3, and Ni powders were thoroughly mixed according to the desired stoichiometry, and then prefired at 900 [?]C for 24 h. The obtained powders were ground, pelletized, and calcined at 1000, 1100 and 1200 [?]C for 24 h with intermediate grinding twice. White compounds, SrMo1-xNixO4, were obtained. The compounds were ground and pressed into small pellets about 10 mm diameter and 2 mm thickness. These pellets were reduced in a H2/Ar (5%: 95%) flow at 920 [?]C for 12 h, and then the deep red colored products of SrMo1-xNixO3 were obtained.

Figure 1: Overview of edge editing approach

We propose a novel, iterative, edge-editing approach to document-level RE. The overview of our approach and an example of the extraction results are illustrated in Figure 1. Our approach treats relations as a relation graph that is composed of entities as nodes and their relations as edges. The relation graph is first initialized using the edges predicted by an existing RE model if provided. Edges are then edited by a neural edge classifier that represents edges using the document information, prebuilt graph information, and the current edge information. The document information is represented with pretrained Longformer models (Beltagy et al., 2020), while the graph information is represented with graph convolutional networks (Kipf and Welling, 2017). Edges are edited iteratively in a close-first manner so that the approach can utilize the information of edges between close entity pairs

---

[1] The source code is available at `https://github.com/tti-coin/edge-editing`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2653–2662
August 1–6, 2021.

in editing edges of distant entity pairs, which are often difficult to predict. We evaluate our approach on the task to extract synthesis procedures from text (Mysore et al., 2019) and show the effectiveness of our approach.

The contribution of this paper is three-fold. First, we propose a novel edge-editing approach for document-level RE that utilizes contexts in both relation graphs and documents. Second, we build a strong rule-based model and show that our approach can effectively utilize and enhance the output of the rule-based model. Third, we build and evaluate a neural model for extracting synthesis procedures from text for the first time.

2 Approach

Our approach extracts a relation graph on given entities from a document. We formulate the extraction task as an edge-editing task, where the approach iteratively edits edges with a neural edge classifier in a close-first manner (Miwa and Sasaki, 2014).

2.1 Iterative Edge Editing

We build a relation graph by editing the edges iteratively using the edge classifier in Section 2.2. The building finishes when all edges are edited. The edges are edited in a close-first manner (Miwa and Sasaki, 2014; Ma et al., 2019) that edits the close edges first and far edges later. The distance between the entity pair is defined based on the appearing order of entities in a document; if two entities in a pair appear m-th and $m + 3$-th, the distance becomes 3. Note that each edge is edited only once throughout the entire editing process.

Algorithm 1 shows the method to build the graph by the iterative edge editing. To reduce the computational cost, the pairs with the same distance are edited simultaneously and the pairs with distances more than or equal to the maximum distance d_{max} are edited simultaneously. This reduces the number of edits from $|\mathcal{N}|^2$ to d_{max}.

2.2 Edge Classifier

An edge classifier predicts the class of the target edge $\hat{\mathcal{E}}_{ij}$ from inputs that are composed of a document information doc, a graph of nodes $\mathcal{N}$ and edges $\mathcal{E}$, and the node pair $(\mathcal{N}_i, \mathcal{N}_j)$ of a target edge. The classifier composed of three modules:

EncodeNode that produces document-based node representations $\bar{\mathcal{N}}$ using the document doc and the entity information of the nodes $\mathcal{N}$.

Algorithm 1: Iterative Edge Editing

```
    Distance($\mathcal{N}, d_1, d_2$) returns pairs that have
    distance $d$ ($d_1 \leq d < d_2$).
  Input: doc: document, $\mathcal{E}$: initial edges
    $d_{max}$: maximum distance
  Output: $\mathcal{E}$: edited edges
    $\bar{\mathcal{N}} \Leftarrow \mathrm{EncodeNode}(\mathrm{doc}, \mathcal{N})$
    while $d$ in $\mathrm{range}(\max(|\mathcal{N}|, d_{max}))$ do
      $\bar{\mathcal{N}}^G \Leftarrow \mathrm{GCN}(\bar{\mathcal{N}}, \mathcal{E})$
      $\bar{\mathcal{E}} \Leftarrow \mathrm{EncodeEdge}(\bar{\mathcal{N}}^G, \mathcal{E})$
      if $d = d_{max}$ then
        $\mathcal{P} \Leftarrow \mathrm{Distance}(\mathcal{N}, d_{max}, \infty)$
      else
        $\mathcal{P} \Leftarrow \mathrm{Distance}(\mathcal{N}, d, d + 1)$
      end if
      while $(i, j)$ in $\mathcal{P}$ do
        $\mathcal{E}_{ij} \Leftarrow \mathrm{ClassifyEdge}(\bar{\mathcal{E}}_{ij})$
      end while
    end while
```

EncodeEdge that obtains the representation of edges $\bar{\mathcal{E}}$ that applies GCN on a prebuilt graph with the node representations $\bar{\mathcal{N}}$ and edges $\mathcal{E}$.

ClassifyEdge that predicts the class of the edge $\hat{\mathcal{E}}_{ij}$ using the edge representation $\bar{\mathcal{E}}_{ij}$ between the node pair $(\mathcal{N}_i, \mathcal{N}_j)$.

We explain the details of these modules in the remaining part of this section.

EncodeNode employs Longformer (Beltagy et al., 2020) to obtain the document-level representation. It aggregates subword representations within each entity by max-pooling Pool and concatenates the aggregated information with the entity's class label representation $\boldsymbol{v}^{lab}$.

$$\begin{aligned} \bar{\mathcal{N}} &= \mathrm{EncodeNode}(\mathrm{doc}, \mathcal{N}) \\ &= [\mathrm{Pool}(\mathrm{Longformer}(\mathrm{doc})); \boldsymbol{v}^{lab}], \end{aligned} \quad (1)$$

where $[\cdot;\cdot]$ denotes concatenation.

To prepare the input to EncodeEdge, the obtained document-based node representation is enriched by GCN to introduce the context of each node in the prebuilt graph: $\bar{\mathcal{N}}^G = \mathrm{GCN}(\bar{\mathcal{N}}, \mathcal{E})$. We add inverse directions to the graph and assign different weights to different classes in graph convolutional network (GCN) following Schlichtkrull et al. (2018). The produced node representation $\bar{\mathcal{N}}^G$ includes both document and prebuilt graph contexts.

EncodeEdge produces the edge representation $\bar{\mathcal{E}}$ from $\bar{\mathcal{N}}^G$. It individually calculates the representation of the edge $\bar{\mathcal{E}}_{ij}$ for each pair of nodes $(\mathcal{N}_i, \mathcal{N}_j)$ by combining the representations of nodes similarly to Zhou et al. (2021) with the embedding of the distance of the entity pair $\boldsymbol{b}_{ij}$ and the edge class $\boldsymbol{e}_{ij}^{old}$ before editing. The distance between the entity pairs is calculated in the same way as in Section 2.1. If the distance exceeds a predefined maximum distance, it will be treated as the maximum distance. We prepare fully connected (FC) layers, FC^H and FC^T, for the start point (head) and end point (tail) nodes and calculate the edge representation as follows:

$$\begin{aligned} \bar{\mathcal{E}}_{ij} &= \mathrm{EncodeEdge}(\bar{\mathcal{N}}^G, \mathcal{E})_{ij} \\ &= [\mathrm{FC}^H(\bar{\mathcal{N}}_i^G)^\top \boldsymbol{W} \mathrm{FC}^T(\bar{\mathcal{N}}_j^G); \boldsymbol{b}_{ij}; \boldsymbol{e}_{ij}^{old}], \end{aligned} \quad (2)$$

where $\boldsymbol{W}$ denotes a trainable weight parameter.

ClassifyEdge classifies the target edge $\mathcal{E}_{ij}$ into a relation class or no relation. It applies a dropout layer (Srivastava et al., 2014), a FC layer for output FC^{out} and softmax to the edge representation $\bar{\mathcal{E}}_{ij}$ to predict the class $\hat{\mathcal{E}}_{ij}$ with the highest probability.

$$\begin{aligned} \hat{\mathcal{E}}_{ij} &= \mathrm{ClassifyEdge}(\bar{\mathcal{E}}_{ij}) = \arg\max \hat{\boldsymbol{p}}_{ij} \\ \hat{\boldsymbol{p}}_{ij} &= \mathrm{Softmax}(\mathrm{FC}^{out}(\mathrm{Dropout}(\bar{\mathcal{E}}_{ij}))) \end{aligned} \quad (3)$$

We maximize the log-likelihood in training the edge classifier.

3 Experiments

3.1 Experimental Settings

We evaluate our approach on the materials science procedural text corpus (Mysore et al., 2019). In the corpus, the synthesis procedures are annotated as a graph in a document, where 19 node types such as materials, operations, and conditions and 15 directed relation types are defined. The corpus consists of 200 documents for training, 15 for development, and 15 for test. The statistics of the corpus are shown in Appendix A. We chose this corpus since this corpus is publicly available, manually annotated, and it deals with a dense document-level relation graph.

We prepared a rule-based model (**RULE**) as a baseline and as an existing model to initialize the edges, which was adapted from the rule-based system in Kuniyoshi et al. (2020). The rules are summarized in Appendix B.

We employ the micro F-score for each relation class as the evaluation metric. We tune the hyper-parameters such as the number and dimensions of layers and dropout rate on the development set using the hyper-parameter optimization framework Optuna (Akiba et al., 2019) and the details are shown in Appendix C. We employ the Adam (Kingma and Ba, 2015) optimizer with the default parameters in PyTorch (Paszke et al., 2019) except for the learning rate. The training was performed without finetuning for the Longformer because the corpus is small to train a large transformer model.

	Dev	Test
EDIT	**0.788**	**0.729**
EDIT-IE	0.732	0.685
EDIT-GCN	0.744	0.703
RANDOM EDIT	0.751	0.690
RANDOM INIT	0.756	0.720

Table 1: Evaluation results in micro F-score without RULE

We compare the following models on graphs initialized by the rule-based model (**with RULE**) and empty graphs (**without RULE**).

EDIT: Proposed model

EDIT-IE: EDIT without iterative edge editing, i.e., $d_{max} = 1$.

EDIT-GCN: EDIT without GCN by replacing $\bar{\mathcal{N}}^G$ with $\bar{\mathcal{N}}$ in Equation (2)

RANDOM EDIT: EDIT with random-order editing

Additionally, we evaluate the following model with randomly initialized graphs.

RANDOM INIT: EDIT with randomly connected edges, the number of which is same as that of the extraction results of RULE, with random classes

Note that although we did not provide the direct comparison with the existing models, our EDIT-GCN without RULE is similar to BRAN (Verga et al., 2018); the only differences are that we use Longformer (Beltagy et al., 2020) instead of transformers, and NER training is not included. Moreover, most of the models for the document-level RE require dataset annotating both entities and their mentions, so the existing models like ATLOP (Zhou et al., 2021) cannot be directly applied to the current task.

3.2 Results without RULE

We show the results with empty initial graphs in Table 1. EDIT shows the highest scores and this indicates the effectiveness of our approach when the initial graphs are empty. When we compare EDIT, EDIT-IE, and RANDOM EDIT, we find that both iterative edge editing and close-first strategy

	Dev	Test
RULE	0.797	0.807
EDIT	**0.878**	0.851
EDIT-IE	0.863	**0.863**
EDIT-GCN	0.857	0.834
RANDOM EDIT	0.791	0.744

Table 2: Evaluation results in micro F-score with RULE

are effective. Since EDIT-GCN extracts from context without graph structure information, the better performance of EDIT over EDIT-GCN shows the effectiveness of the information in the graph structure. The low performance with RANDOM INIT shows that the edge information needs to be reliable.

3.3 Results with RULE

We summarize the results with RULE in Table 2. We show the detailed results for EDIT without RULE, RULE, and EDIT-IE with RULE in Appendix D.

When we compare the results with Table 1, the performance with RULE is better than the counterpart without RULE for all the settings. Furthermore, all the scores in Table 2 are better than those in Table 1, which shows the strength of RULE.

Surprisingly, the results with our approach are better than that of RULE even though RULE is better than our approach without RULE. This indicates our EDIT approach can make the prediction accurate. We can conclude that our EDIT approach can utilize the information from the rule-based model and the initialization of the edges by RULE is useful.

As for the performance of the models, most results are consistent with Table 1 except that EDIT-IE shows the highest score on the test set. This may be partly because the initial graph by RULE is already reliable and editing does not help to improve the context. Results with RANDOM EDIT support this since the performance degradation with RANDOM EDIT is large compared to Table 1 and RANDOM EDIT is harmful in this case. Moreover, the different behaviors on the development and test sets indicate an imbalance in the corpus split.

4 Case Study

We illustrated 6 graphs for an example document (Zhang et al., 2007) in the development data set shown in Figure 2: the result on the right side of Figure 1 shows our best extraction result using EDIT-IE with RULE; Figure 3 shows the correct extraction; Figure 4 shows the extraction result using EDIT without RULE; Figure 5 shows the extraction result using RULE; and Figure 6 shows the extraction result using EDIT with RULE. Figure 3 shows the material synthesis starts from *mixed* with materials *SrCO3*, *MoO3* and *Ni* to *prefired* and so on, and the material *SrMo1-xNixO4* is synthesized. When we compare Figure 6 with Figure 5, the extraction results are similar to RULE. Although the overall performance is low, Figure 4, which does not depend on the rule, extracts relations that are not extracted by the other systems and this shows the models with RULE and without RULE capture different relations.

5 Related Work

RE has been widely studied to identify the relation between two entities in a sentence. In addition to traditional feature/kernel-based methods (Zelenko et al., 2003; Miwa and Sasaki, 2014), many neural RE methods have been proposed based on convolutional neural networks (CNNs) (Zeng et al., 2014), recurrent neural networks (RNNs) (Xu et al., 2015; Miwa and Bansal, 2016), graph convolutional networks (GCNs) (Zhang et al., 2018; Schlichtkrull et al., 2018), and transformers (Wang et al., 2019). However, sentence-level RE is not enough to cover the relations in a document, and document-level RE has increasingly received research attention in recent years.

Major approaches for document-level RE are graph-based methods and transformer-based methods. For graph-based methods, Quirk and Poon (2017) first proposed a document graph for document-level RE. Christopoulou et al. (2019) constructed a graph that included heterogeneous nodes such as entity mentions, entities, and sentences and represented edges between entities from the graph. Nan et al. (2020) proposed the automatic induction of a latent graph for relational reasoning across sentences. The document graphs in these methods are defined on nodes of linguistic units such as words and sentences, which are different from our relation graphs. Unlike our method, these methods do not directly deal with relation graphs among entities.

For transformer-based methods, Verga et al. (2018) introduced a method to encode a document with transformers to obtain entity embedding and

A series of *polycrystalline* samples of *SrMo1-xNixO4* (0.02<=x<=0.08) were *prepared* through the conventional *solid-state reaction method* in *air*. *Appropriate proportions* of *high-purity SrCO3*, *MoO3*, and *Ni powders* were thoroughly *mixed* according to the *desired stoichiometry*, and then *prefired* at *900* [?]*C* for *24 h*. The *obtained powders* were *ground*, *pelletized*, and *calcined* at *1000*, *1100* and *1200* [?]*C* for *24 h* with intermediate grinding twice. White *compounds*, *SrMo1-xNixO4*, were *obtained*. The *compounds* were *ground* and *pressed* into small *pellets* about *10 mm diameter* and *2 mm thickness*. These *pellets* were *reduced* in a *H2/Ar* (*5%*: *95%*) flow at *920* [?]*C* for *12 h*, and then the deep red colored products of *SrMo1-xNixO3* were *obtained*.

Figure 2: Example document

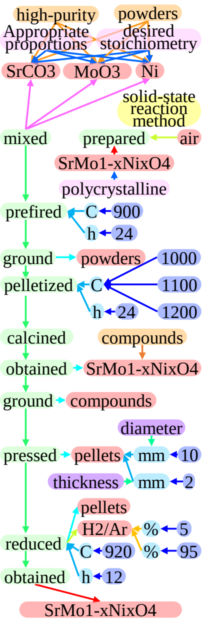

Figure 3: Gold graph for the document in Figure 2

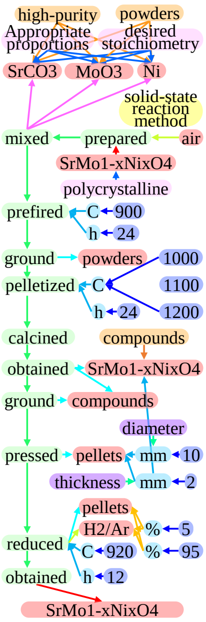

Figure 4: Example extraction results from the document in Figure 2 by EDIT without RULE

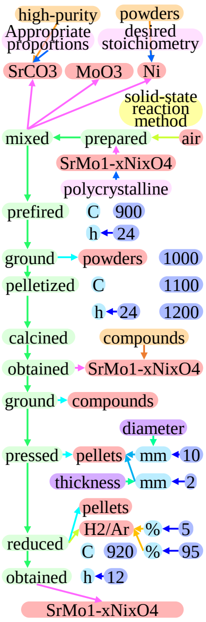

Figure 5: Example extraction results from the document in Figure 2 by RULE

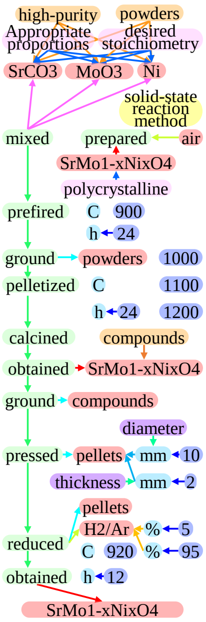

Figure 6: Example extraction results from the document in Figure 2 by our EDIT with RULE

classify the relations between entities using the embedding. Tang et al. (2020) proposed a Hierarchical Inference Network (HIN) for document-level RE, which aggregates information from entity level to document level. Zhou et al. (2021) tackled document-level RE with an Adaptive Thresholding and Localized cOntext Pooling (ATLOP) model that introduces a learnable entity-dependent threshold for classification and aggregated local mention-level contexts that are relevant to both entities.

Several studies focus on procedural texts such as cooking recipes (Bosselut et al., 2018), scientific processes (Dalvi et al., 2018) and open domain procedures (Tandon et al., 2020). They, however, do not directly treat relation graphs. Several efforts have been made to annotate procedural or action graphs in procedural text (Mori et al., 2014; Mysore et al., 2019; Kuniyoshi et al., 2020). Kuniyoshi et al. (2020) and Mehr et al. (2020) individually proposed rule-based systems to extract procedures from a document, but no neural methods have been proposed for the extraction.

6 Conclusions

We proposed a novel edge editing approach for document-level relation extraction. This approach treats the task as the edge editing of relation graphs, given nodes. It edits edges considering contexts in the document and the relation graph. We evaluated the approach on the material synthesis procedure corpus, and the results showed the usefulness of initializing edges by the rule-based model, utilizing prebuilt graph information for editing and editing in a close-first manner. As a result, our model performed an F-score of 86.3% for edge prediction.

In future work, we plan to improve the approach to obtain more consistent and accurate relation graphs. We also would like to apply the approach to other data sets such as cooking recipes (Mori et al., 2014) and temporal graphs (Pustejovsky et al., 2003; Cassidy et al., 2014).

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, pages 2546–2554. Curran Associates, Inc.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations*.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506, Baltimore, Maryland. Association for Computational Linguistics.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4925–4936, Hong Kong, China. Association for Computational Linguistics.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Fusataka Kuniyoshi, Kohei Makino, Jun Ozawa, and Makoto Miwa. 2020. Annotating and extracting synthesis process of all-solid-state batteries from scientific literature. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1941–1950, Marseille, France. European Language Resources Association.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. A system for massively parallel hyperparameter tuning. In *Proceedings of Machine Learning and Systems*, volume 2, pages 230–246.

Shuai Ma, Gang Wang, Yansong Feng, and Jinpeng Huai. 2019. Easy first relation extraction with information redundancy. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3851–3861, Hong Kong, China. Association for Computational Linguistics.

S Hessam M Mehr, Matthew Craven, Artem I Leonov, Graham Keenan, and Leroy Cronin. 2020. A universal system for digitization and automatic execution of the chemical synthesis literature. *Science*, 370(6512):101–108.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *LREC*, pages 2370–2377.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64, Florence, Italy. Association for Computational Linguistics.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1546–1557, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The TimeBank corpus. In *Proceedings of Corpus Linguistics*, pages 647–656, Lancaster, UK.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182, Valencia, Spain. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online. Association for Computational Linguistics.

Hengzhu Tang, Yanan Cao, Zhenyu Zhang, Jiangxia Cao, Fang Fang, Shi Wang, and Pengfei Yin. 2020. HIN: Hierarchical inference network for document-level relation extraction. In *Advances in Knowledge Discovery and Data Mining*, pages 197–209, Cham. Springer International Publishing.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884, New Orleans, Louisiana. Association for Computational Linguistics.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019. Extracting multiple-relations in one-pass with pre-trained transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1371–1377, Florence, Italy. Association for Computational Linguistics.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

S.B. Zhang, Y.P. Sun, B.C. Zhao, X.B. Zhu, and W.H. Song. 2007. Influence of Ni doping on the properties of perovskite molybdates srmo1-xnixo3 (0.02≤x≤0.08). *Solid State Communications*, 142(12):671 – 675.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Wenxuan Zhou, Kevin Huang, Tengyu Ma, and Jing Huang. 2021. Document-level relation extraction with adaptive thresholding and localized context pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Statistics of the Materials Science Procedural Text Corpus

We present the statistics of the materials science procedural text corpus[2] proposed by Mysore et al. (2019). Table 3 and Table 4 summarize the numbers of entities and relations, respectively.

---

[2] `https://github.com/olivettigroup/annotated-materials-syntheses`

B Rule-based Relation Extraction Model

We built a rule-based model by defining the rules to extract relations between entity pairs for the materials science procedural text corpus (Mysore et al., 2019). The rules were adapted from the rule-based model in (Kuniyoshi et al., 2020) for the target corpus. The rules depend on labels of the entities of an entity pair, distance, and the order of occurrence of the entities. According to the combination of labels of the entities, our rules are divided into three types: Operation–Operation, Operation–Material and other relations. In the following, the starting point of a relation is called head and the ending point is called tail, and an edge is denoted as Head–Tail.

B.1 Operation–Operation

The relation Operation–Operation takes only a Next_Operation label, which means the progress of operation.

Next_Operation: Close Operation entities are linked with the relation from the beginning to the end in the document order, in which the entities of Operation appear.

B.2 Operation–Material

For the edges of Operation–Material, there are five relation labels: Recipe_Precursor indicates the input of a material; Recipe_Target indicates the generation of a product; Participant_Material indicates the generation of an intermediate product; Solvent_Material indicates the solvent material of an operation; and Atmospheric_Material indicates the atmosphere of an operation.

For Solvent_Material, Atmospheric_Material and Participant_Material labels, a dictionary is prepared manually for each label. The relations are linked from the nearest Operation to a Material in the sentence if the Material match in the dictionary since these relations take specific Material entities. The dictionary is included in the source code.

Recipe_Precursor is linked from all Material that do not match the dictionary of Solvent_Material, Atmospheric_Material, and Participant_Material to the nearest Operation. This rule-based model does not produce the relation Recipe_Target. The reason for these decisions is that it is difficult to classify these relations with simple rules.

Entity class	Train	Dev	Test
Material	4,271	277	316
Operation	3,249	212	242
Number	2,872	224	219
Condition-Unit	1,363	101	87
Material-Descriptor	1,214	67	89
Amount-Unit	1,193	96	98
Property-Misc	481	25	16
Condition-Misc	468	32	20
Synthesis-Apparatus	433	20	34
Nonrecipe-Material	329	33	25
Brand	291	30	27
Apparatus-Descriptor	165	10	9
Amount-Misc	149	14	7
Meta	128	12	13
Property-Type	124	10	4
Condition-Type	119	2	1
Reference	106	10	11
Property-Unit	92	7	8
Apparatus-Unit	89	6	16
Character.-Apparatus	54	2	11
Apparatus-Property-Type	26	0	6

Table 3: Entities in the materials science procedural text corpus

Relation class	train	dev	test
Next_Operation	2,898	184	202
Recipe_Precursor	876	67	89
Recipe_Target	363	31	22
Participant_Material	1,723	113	124
Solvent_Material	463	28	33
Atmospheric_Material	183	11	14
Property_Of	586	35	21
Condition_Of	1,810	132	107
Number_Of	2,805	219	209
Amount_Of	1,512	130	121
Descriptor_Of	1,495	91	102
Brand_Of	423	42	41
Type_Of	164	7	13
Apparatus_Of	455	20	36
Apparatus_Attr_Of	90	6	11
Coref_Of	267	12	14

Table 4: Relations in the materials science procedural text corpus

B.3 Remaining Relations

The remaining 9 relation labels are defined between the other pairs of entity labels: Property_Of, which indicates a condition of a material; Condition_Of, which indicates a condition of an operation; Number_Of, which indicates the relationship between a number and a unit; Amount_Of, which indicates a condition of a quantity; Type_Of, which indicates a condition of a numerical condition; Brand_Of, which indicates the brand of a material or equipment; Apparatus_Of, which indicates equipment used in an operation; APPARATUS_ATTR_OF, which indicates a numerical condition of on equipment; and DESCRIPTOR_OF, which indicates other conditions. For these labels, the rules are defined based only on the labels of head and tail entities and the distance between them. We explain the detailed rules in the remainder of this section.

PROPERTY_OF: The relation can take PROPERTY-UNIT or PROPERTY-MISC as the head and MATERIAL or NONRECIPE-MATERIAL as the tail. When PROPERTY-UNIT is a head, it is linked with the nearest MATERIAL in the sentence. When PROPERTY-MISC is a head, it is linked to the nearest MATERIAL or NONRECIPE-MATERIAL in the sentence.

CONDITION_OF: CONDITION-UNIT and CONDITION-MISC are linked to the nearest OPERATION with the relation in the sentence.

NUMBER_OF: NUMBER is linked to the nearest PROPERTY-UNIT, CONDITION-UNIT, or APPARATUS-UNIT that appear after the NUMBER in the sentence.

AMOUNT_OF: The relation is linked from AMOUNT-UNIT and AMOUNT-UNIT to the nearest MATERIAL or NONRECIPE-MATERIAL in the sentence.

DESCRIPTOR_OF: When MATERIAL-DESCRIPTOR is a head, it is linked to the nearest MATERIAL or NONRECIPE-MATERIAL in the sentence. When APPARATUS-DESCRIPTOR is a head, it is linked to the nearest SYNTHESIS-APPARATUS in the sentence.

APPARATUS_OF: The relation is linked from SYNTHESIS-APPARATUS and CHARACTERIZATION-APPARATUS to the nearest OPERATION with the priority given to the OPERATION that appear before the APPARATUS in the sentence.

TYPE_OF: PROPERTY-TYPE and APPARATUS-PROPERTY-TYPE are linked to the nearest PROPERTY-UNIT and APPARATUS-UNIT in the sentence with the relation, respectively. When CONDITION-TYPE is a head, it is linked to the nearest CONDITION-UNIT that appears before the CONDITION-TYPE in the sentence.

BRAND_OF: The relation is linked from BRAND to the nearest entities that may have brands (i.e., MATERIAL, NONRECIPE-MATERIAL, SYNTHESIS-APPARATUS, and CHARACTERIZATION-APPARATUS) in the sentence.

Parameter	Range	Value
Learning rate	[1e-5, 1e-2)	0.001
No. of GCN layers	[0, 4]	3
d_{max}	[1, 10]	4
Dimension of hidden layers	[32, 128]	85
No. of FC^{out} layers	[1, 5]	4
No. of FC^{h} and FC^{t} layers	[1, 5]	1
Dropout rate	[0.0, 1.0)	0.46
Dimension of $\boldsymbol{e}_{ij}^{old}$	[1, 32]	3
Maximum distance for $\boldsymbol{b}_{ij}$	[1, 32]	3
Dimension of $\boldsymbol{b}_{ij}$	[1, 100]	1
Use bidirectional GCN	True or False	True

Table 5: Search space for optimization of hyper-parameters and the selected values after optimization

Relation	Prec.	Recall	F-score
NEXT_OPERATION	0.622	0.693	0.656
RECIPE_PRECURSOR	0.632	0.539	0.582
RECIPE_TARGET	0.640	0.727	0.681
PARTICIPANT_MATERIAL	0.641	0.476	0.546
SOLVENT_MATERIAL	0.491	0.818	0.614
ATMOSPHERIC_MATERIAL	0.733	0.786	0.759
PROPERTY_OF	0.773	0.810	0.791
CONDITION_OF	0.798	0.850	0.824
NUMBER_OF	0.874	0.962	0.916
AMOUNT_OF	0.722	0.645	0.681
DESCRIPTOR_OF	0.761	0.814	0.787
BRAND_OF	0.567	0.415	0.479
TYPE_OF	0.900	0.692	0.783
APPARATUS_OF	0.657	0.639	0.648
APPARATUS_ATTR_OF	0.769	0.909	0.833
COREF_OF	0.875	0.500	0.636
Overall	0.717	0.722	0.720

Table 6: Detailed results using EDIT without RULE on the test set

APPARATUS_ATTR_OF: APPARATUS-UNIT is linked to the nearest SYNTHESIS-APPARATUS or CHARACTERIZATION-APPARATUS.

COREF_OF: The relation is not detected by the rules because it is difficult to describe rules.

C Tuning Details

We tuned our model using a hyper-parameter optimization framework Optuna (Akiba et al., 2019). We searched for the hyper-parameters that maximize micro-F scores within 600 trials on the development set. We employed the tree-structured Parzen estimator algorithm (Bergstra et al., 2011) for the sampler and the successive halving algorithm (Li et al., 2020) for the pruner with default options in Optuna. In each trial of the search, we trained our model for 100 epochs, which was confirmed by preliminary experiments to be sufficient for convergence. We searched hyper-parameters on 20 NVIDIA GPUs, which include Tesla V100, TITAN V, RTX 3090, and GTX TITAN Xp GPUs.

Relation	Prec.	Recall	F-score
NEXT_OPERATION	0.990	0.881	0.932
RECIPE_PRECURSOR	0.730	0.414	0.528
RECIPE_TARGET	0.000	0.000	0.000
PARTICIPANT_MATERIAL	0.419	0.800	0.550
SOLVENT_MATERIAL	0.697	0.418	0.522
ATMOSPHERIC_MATERIAL	1.000	0.378	0.549
PROPERTY_OF	0.905	1.000	0.950
CONDITION_OF	0.963	0.981	0.972
NUMBER_OF	0.943	0.961	0.952
AMOUNT_OF	0.744	0.865	0.800
DESCRIPTOR_OF	0.931	0.979	0.955
BRAND_OF	0.561	0.920	0.697
TYPE_OF	0.769	1.000	0.870
APPARATUS_OF	0.972	0.854	0.909
APPARATUS_ATTR_OF	0.909	0.769	0.833
COREF_OF	0.000	0.000	0.000
Overall	0.807	0.808	0.807

Table 7: Detailed results with RULE on the test set

Relation	Prec.	Recall	F-score
NEXT_OPERATION	0.905	0.990	0.946
RECIPE_PRECURSOR	0.810	0.573	0.671
RECIPE_TARGET	0.560	0.636	0.596
SOLVENT_MATERIAL	0.733	0.667	0.698
PARTICIPANT_MATERIAL	0.624	0.790	0.698
ATMOSPHERIC_MATERIAL	0.778	1.000	0.875
PROPERTY_OF	0.905	0.905	0.905
CONDITION_OF	0.953	0.944	0.948
NUMBER_OF	0.958	0.990	0.974
AMOUNT_OF	0.854	0.868	0.861
DESCRIPTOR_OF	0.941	0.931	0.936
BRAND_OF	0.880	0.537	0.667
TYPE_OF	1.000	0.692	0.818
APPARATUS_OF	0.833	0.972	0.897
APPARATUS_ATTR_OF	0.769	0.909	0.833
COREF_OF	0.750	0.429	0.545
Overall	0.856	0.870	0.863

Table 8: Detailed results using EDIT-IE with RULE on the test set

We defined the search space as shown in Table 5; the hyper-parameters for the search are composed of the learning rate for Adam, the number of GCN layers, the maximum edit distance d_{max}, the dimensions of all hidden layers, the number of FC^{out} layers, the number of FC^{h} and FC^{t} layers, the dropout rate, the dimension of $\boldsymbol{e}_{ij}^{old}$, the maximum distance and the dimension for $\boldsymbol{b}_{ij}$ and whether to use bidirectional GCNs or uni-directional GCNs. In the table, the range column shows the range of values to search and the final value column shows the rounded selected values after the optimization.

D Detailed Evaluation Results

Our editing models for evaluation are trained with a TITAN V GPU for EDIT with RULE and a Tesla V100 GPU for the others. The training takes about 6 hours 30 minutes with EDIT-IE using RULE and 21 hours with EDIT not using RULE.

We show the detailed evaluation results with precision (Prec.), recall, and F-score on the test set in Table 6 for EDIT without RULE, Table 7 for RULE, and Table 8 for EDIT-IE without RULE. The results show the relations that are not covered by RULE, i.e., RECIPE_TARGET and COREF_OF, are extracted by our approach, and for these classes, EDIT without RULE show the better performance than the models with RULE. Some relations with high performance by RULE, including NEXT_OPERATION, CONDITION_OF, and DESCRIPTOR_OF, are extracted by EDIT-IE with RULE in high performance. This shows our approach can effectively utilize the outputs of RULE.

Dialogue-oriented Pre-training

Yi Xu[1,2,3], **Hai Zhao**[1,2,3,*]
[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
`xuyi_2019@sjtu.edu.cn`, `zhaohai@cs.sjtu.edu.cn`

Abstract

Pre-trained language models (PrLM) has been shown powerful in enhancing a broad range of downstream tasks including various dialogue related ones. However, PrLMs are usually trained on general plain text with common language model (LM) training objectives, which cannot sufficiently capture dialogue exclusive features due to the limitation of such training setting, so that there is an immediate need to fill the gap between a specific dialogue task and the LM task. As it is unlikely to collect huge dialogue data for dialogue-oriented pre-training, in this paper, we propose three strategies to simulate the conversation features on general plain text. Our proposed method differs from existing post-training methods that it may yield a general-purpose PrLM and does not individualize to any detailed task while keeping the capability of learning dialogue related features including speaker awareness, continuity and consistency. The resulted Dialog-PrLM is fine-tuned on three public multi-turn dialogue datasets and helps achieve significant and consistent improvement over the plain PrLMs.

1 Introduction

Recently, pre-trained language models (PrLMs) have shown impressive improvements for various downstream NLP tasks (Zhou et al., 2020; Ouyang et al., 2021; Zhang and Zhao, 2021; Radford et al., 2018; Yang et al., 2019; Zhang et al., 2020c; Clark et al., 2020; Li et al., 2021), including the response selection task for multi-turn dialogues, which takes a dialogue history as input and aims to select a most suitable response from a collection of answers (Zhou et al., 2016; Wu et al., 2017; Zhou et al., 2018b; Zhu et al., 2018; Zhang et al., 2018; Tao et al., 2019; Gu et al., 2019).

Dialogue History:
A: *Could you please get me a train?*
B: *Sure I can help you find a train.*
B: *where are you coming from?*
B: *What time do you need to leave by?*
A: *I am leaving Leicester and I need to leave by 20:30.*
B: *What is your destination and day of travel?*
A: *Cambridge and on friday.*
A: *Can I just get the travel time for the train? Thanks!*
Response:
B: *The first train leaving after 20:30 is 142 21:09 and the travel time is 105 minutes.*

Table 1: A multi-turn dialogue example with interleaved or continuous utterances between two speakers.

Pre-training tasks of all these PrLMs almost concentrate on two aspects: token prediction and sentence relation prediction. For example, the genetic BERT model (Devlin et al., 2019) uses masked language modeling (MLM) and next sentence prediction (NSP) objectives; ALBERT (Lan et al., 2020) predicts sentence order rather than NSP; ELECTRA (Clark et al., 2020) transfers MLM into a generating and then discriminating process like GAN (Goodfellow et al., 2014). However, these tasks are just devoted to incorporating token-level and sentence-level semantic information into embeddings, and cannot be sufficiently compatible with its dialogue-oriented characteristics.

Table 1 shows a multi-turn dialogue example. Compared with plain text, the utterance turn and speaker role keep shift as a conversation goes on, and the next utterance should keep continuous and consistent with the context. Besides, the two speakers may not follow strict shift rules, and one speaker may continuously shoot multiple utterances. Although some existing works have noticed such

---

[*] Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), Key Projects of National Natural Science Foundation of China (U1836222 and 61733011), Huawei-SJTU long term AI project, Cutting-edge Machine Reading Comprehension and Language Model. This work was supported by Huawei Noah's Ark Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2663–2673
August 1–6, 2021.

nonlinear nature of multi-turn dialogues, they are limited to conducting post-training or pre-training in a specific domain and do not provide general-purpose dialogue-oriented PrLMs to fundamentally solve this problem (Xu et al., 2021a; Whang et al., 2021; Wolf et al., 2019; Zhang et al., 2020b; Henderson et al., 2020; Bao et al., 2020).

In this work, we make the first attempt to train a general-purpose dialogue-oriented PrLM. However, such a PrLM should be trained on huge dialogue data, which is hard to collect. Thus we propose three novel pre-training strategies (i.e., Insertion, Deletion, Replacement), so that we facilitate plain text originally for common PrLM training to simulate dialogue-like features. The resulted model, we denote as Dialog-PrLM, then is capable of effectively learning speaker awareness, continuity and consistency in a general way. Especially, for the convenient use of the downstream dialogue tasks, we introduce a special token `[SOT]` before each utterance to tell that it is a start of a turn and learn from these three strategies. These targeted pre-training tasks enable `[SOT]` to better represent each context utterance. We mimic dialogue-related features on conventional plain text, which can bring up the possibility that similar techniques could be adopted in other domains not only for dialogues.

Our pre-trained Dialog-PrLM is fine-tuned on three multi-turn dialogue response selection benchmarks, and obtains significant and consistent improvements over the plain PrLMs.

2 Related Work

For multi-turn dialogue response selection task, earlier works conduct single-turn match, which concatenates all the utterances in the history dialogue or just considers the last one to match with the candidate response (Lowe et al., 2015; Kadlec et al., 2015; Yan et al., 2016; Tan et al., 2016; Wan et al., 2016; Wang and Jiang, 2016). Recently, existing works tend to model the interaction between each dialogue utterance and the response, which usually adopt the encoding-matching-aggregation paradigm (Zhou et al., 2016; Wu et al., 2017; Zhang et al., 2018; Zhou et al., 2018a,b; Tao et al., 2019; Yuan et al., 2019). After encoding, distinct matching networks generate features for each utterance which are usually passed to GRU (Cho et al., 2014) for aggregating into a final matching score. Besides, some works adopt topic information (Xing et al., 2017; Wu et al., 2018; Xu et al., 2021b) or conversation disentanglement to select the proper response (Jia et al., 2020; Wang et al., 2020).

There are more and more practice using powerful PrLMs as the model encoder (Zhang et al., 2021, 2020a; Zhu et al., 2020) like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020) and ELECTRA (Clark et al., 2020). Considering task domain difference from the general corpus for PrLM pre-training, recent studies start to conduct post-training on target multi-turn dialogue datasets to incorporate in-domain knowledge (Whang et al., 2020; Lu et al., 2020; Gu et al., 2020; Xu et al., 2021a; Whang et al., 2021). Whang et al. (2020) conduct post-training of MLM and NSP tasks as BERT. Rather than using the same tasks as PrLMs, Xu et al. (2021a) and Whang et al. (2021) both considers auxiliary tasks through post-training to enhance response selection.

Although the PrLMs which are trained on plain text have learned contextual semantic representation from token-level or sentence-level pre-training tasks like MLM, NSP, they all do not consider dialogue related features like speaker role, continuity and consistency. Despite some existing works (Xu et al., 2021a; Whang et al., 2021) considers that when conducting post-training, they are limited to a specific domain. (Wolf et al., 2019; Zhang et al., 2020b; Henderson et al., 2020; Bao et al., 2020) train on open-domain conversational data like Reddit for response selection or generation tasks, but they are limited to original pre-training tasks on plain text and ignore the dialogue related features. Besides, Wu et al. (2020) and Li et al. (2020) conduct task-specific training on collected dialogue corpora, but they also suffer from biased and limited amount of dialogue data.

Different from all the previous studies, we still make an attempt in obtaining a general-purpose PrLM but not aiming at any specific tasks like post-training methods. Meanwhile, our proposed dialogue-oriented pre-training enables the resulted PrLMs to especially capture dialogue related features in a general way.

3 Dialogue-oriented Pre-training

For dialogue-oriented pre-training, we split and sample sentences as ”utterances” from the general Wikipedia corpus to simulate dialogue-like features. We design three dialogue-oriented pre-training strategies (i.e., Insertion, Deletion, Replacement) to jointly learn dialogue related char-

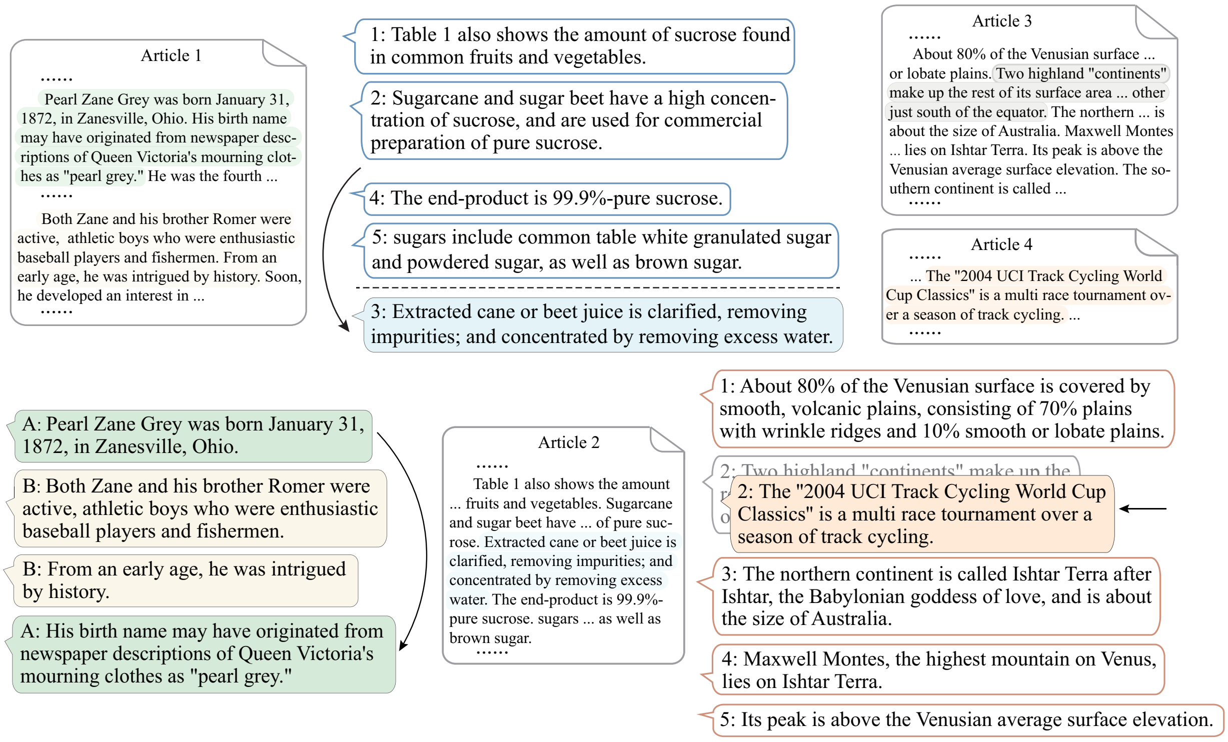
Article 1

......

Pearl Zane Grey was born January 31, 1872, in Zanesville, Ohio. His birth name may have originated from newspaper descriptions of Queen Victoria's mourning clothes as "pearl grey." He was the fourth ...

......

Both Zane and his brother Romer were active, athletic boys who were enthusiastic baseball players and fishermen. From an early age, he was intrigued by history. Soon, he developed an interest in ...

......

A: Pearl Zane Grey was born January 31, 1872, in Zanesville, Ohio.

B: Both Zane and his brother Romer were active, athletic boys who were enthusiastic baseball players and fishermen.

B: From an early age, he was intrigued by history.

A: His birth name may have originated from newspaper descriptions of Queen Victoria's mourning clothes as "pearl grey."

1: Table 1 also shows the amount of sucrose found in common fruits and vegetables.

2: Sugarcane and sugar beet have a high concentration of sucrose, and are used for commercial preparation of pure sucrose.

4: The end-product is 99.9%-pure sucrose.

5: sugars include common table white granulated sugar and powdered sugar, as well as brown sugar.

3: Extracted cane or beet juice is clarified, removing impurities; and concentrated by removing excess water.

Article 2

......

Table 1 also shows the amount ... fruits and vegetables. Sugarcane and sugar beet have ... of pure sucrose. Extracted cane or beet juice is clarified, removing impurities; and concentrated by removing excess water. The end-product is 99.9%-pure sucrose. sugars ... as well as brown sugar.

......

Article 3

......

About 80% of the Venusian surface ... or lobate plains. Two highland "continents" make up the rest of its surface area ... other just south of the equator. The northern ... is about the size of Australia. Maxwell Montes ... lies on Ishtar Terra. Its peak is above the Venusian average surface elevation. The southern continent is called ...

......

Article 4

......

... The "2004 UCI Track Cycling World Cup Classics" is a multi race tournament over a season of track cycling. ...

......

1: About 80% of the Venusian surface is covered by smooth, volcanic plains, consisting of 70% plains with wrinkle ridges and 10% smooth or lobate plains.

2: Two highland "continents" make up the

2: The "2004 UCI Track Cycling World Cup Classics" is a multi race tournament over a season of track cycling.

3: The northern continent is called Ishtar Terra after Ishtar, the Babylonian goddess of love, and is about the size of Australia.

4: Maxwell Montes, the highest mountain on Venus, lies on Ishtar Terra.

5: Its peak is above the Venusian average surface elevation.

(a) Insertion (b) Deletion (c) Replacement

Figure 1: Three dialogue-oriented pre-training strategies.

acteristics based on the plain PrLMs. A special token `[SOT]` is added before each ”utterance”, which tells that it is a start of a turn and matches the realistic scene of turn shift. The three tasks use the embedding of `[SOT]` to represent each utterance and conduct targeted pre-training, which enables `[SOT]` to learn dialogue related representation about speaker-awareness, continuity and consistency respectively. Figure 1 shows the overview of the three strategies.

Insertion In a real scenario, the speaker role might shift or not for each turn as a conversation goes on. Two speakers may carry out a conversation in turn or one speaker may continuously shoot multiple utterances. Considering a conversation session of four sentences between speaker A and B, we consider three possible cases: AABB, ABAB ABBA. The next time A speaks will happen after 0,1, or 2 turns from the last time. To enable Dialog-PrLM aware of the speaker role information, we should first simulate a two-party conversation on Wikipedia. We sample two continuous sentences $\{u_{A1}, u_{A2}\}$ in one paragraph of an article as the two utterances of A, and sample two continuous ones $\{u_{B1}, u_{B2}\}$ in the following paragraph of the same article as what B says. Sampling from the same article is to ensure they are talking about one topic in general, which is in line with the realistic scenario and increases the difficulty of prediction. The ”continuous” sentences simulate that A continues to express his opinion after being interrupted by B. We insert u_{A2} into $\{u_{B1}, u_{B2}\}$, and add `[SOT]` before each utterance. We will not disrupt the utterance order inside one speaker, and also keep the overall order that A first then B. In this way, we can get three cases mentioned above. One possible input case is listed here:

$$X_{ins} = \texttt{[CLS][SOT]}u_{A1}\texttt{[SOT]}u_{B1}\texttt{[SOT]}u_{A2}\texttt{[SOT]}u_{B2}\texttt{[SEP]}$$

The insertion task is to predict the next utterance of A. The whole sequence is encoded by PrLM, and we calculate the cosine similarity of the `[SOT]` embedding of u_{A1} with the other three utterances as matching scores to predict the u_{A2}. These three scores are passed to a softmax layer and use cross entropy as the insertion loss $\mathcal{L}_{ins}$.

Deletion The plain pre-training tasks like MLM, NSP or SOP of PrLMs just enable the model to learn token-level or sentence-level semantic information, and they fail to catch dialogue related signals like continuity, which also helps to choose the answer that is coherent with context. We sample continuous k sentences $\{u_1, u_2, \cdots, u_k\}$ from one paragraph and randomly delete u_i from the first $k-1$ (We do not choose u_k, as there is no `[SOT]` after u_{k-1}). The input sequence of the PrLM is:

X_{del} = `[CLS]` `[SOT]` $u_1 \cdots$ `[SOT]` u_{i-1} `[SOT]` $u_{i+1} \cdots$ `[SOT]` u_k `[SEP]` `[SOT]` u_i `[SEP]`

We append u_i at end and use `[SEP]` for separation. Similarly, we calculate the cosine similarity of the `[SOT]` embedding of u_i with the other remaining $k-1$ utterances to predict the `[SOT]` of u_{i+1}, where u_i should be inserted back. These $k-1$ scores are passed to a softmax layer and use cross entropy as the deletion loss $\mathcal{L}_{del}$.

Replacement The replacement task is to make Dialog-PrLM recognize the inconsistent utterance within a dialogue session, so that it will select the proper response which is consistent with the context in both style and context. Similar to deletion, we sample continuous k sentences $\{u_1, u_2, \cdots, u_k\}$ from one paragraph, and then we sample one sentence u_r from another article, which is used to replace a randomly chosen u_i in $\{u_1, u_2, \cdots, u_k\}$. The input sequence is:

X_{rep} = `[CLS]` `[SOT]` $u_1 ...$ `[SOT]` u_{i-1} `[SOT]` u_r `[SOT]` $u_{i+1} \cdots$ `[SOT]` u_k `[SEP]`

Each `[SOT]` is gathered after encoding, and passed to a linear layer to get a score:

$$score_{u_j} = W_r E_j + b_r$$

where $j = 1...i-1, r, i+1, ...k$, and W_r, b_r are trainable parameters. E_j is the embedding of the jth `[SOT]`. These k scores are passed to a softmax layer and use cross entropy as the replacement loss $\mathcal{L}_{rep}$.

We adopt multi-task learning and define the final dialogue-oriented pre-training loss $\mathcal{L}_{gen}$ as:

$$\mathcal{L}_{gen} = \mathcal{L}_{ins} + \mathcal{L}_{del} + \mathcal{L}_{rep}$$

4 Use of Dialogue-oriented Pre-training

Our Dialog-PrLM may be used in terms of domain fine-funing or multi-task learning: (1) Domain fine-tuning: Our Dialog-PrLM is fine-tuned on the target response selection task. (2) Specific post-training: Our pre-training strategies are slightly adjusted and applied to specific multi-turn dialogue datasets. (3) Domain multi-task learning: the target response selection task jointly learns with the three auxiliary post-training tasks in (2) on Dialog-PrLM.

4.1 Domain Fine-tuning

After our dialogue-oriented pre-training, the target response selection task can be fine-tuned on our Dialog-PrLM. We denote the dataset as $D = \{(C, R, Y)_k\}_{k=1}^N$, where C is dialogue context, and R is the candidate response, and $Y \in \{0, 1\}$ is the label indicating whether R is a proper response for C. Besides, $C = \{U_1, ..., U_n\}$ and $U_i, 1 \leq i \leq n$ is the i-th utterance in context C. We concatenate all utterances $\{U_i\}_{i=1}^n$ as well as the response R and add `[SOT]` before each to represent the following sequence:

X = `[CLS]` `[SOT]` U_1 `[SOT]` $U_2 ...$ `[SOT]` U_n `[SEP]` `[SOT]` R `[SEP]`

With the pre-training, Dialog-PrLM becomes effectively capable of representing each utterance. Therefore, rather than directly using `[CLS]`, we pass all embeddings E of `[SOT]`s to GRU to model sequential interaction of the context and response, whose final hidden state H is used for generating matching score s for C and R:

$$H = \mathrm{GRU}(E), \\ s = \mathrm{sigmoid}(WH + b)$$

where $E \in \mathbb{R}^{(n+1)\times d}, H \in \mathbb{R}^d$, and $W \in \mathbb{R}^d, b \in \mathbb{R}$ are trainable parameters. The response selection loss $\mathcal{L}_{fine-tune}$ is:

$$p(Y \mid C, R) = sY + (1-s)(1-Y)$$
$$L_{fine-tune} = -\frac{1}{N} \sum_{(C,R,Y)\in D} \log(p(Y \mid C, R))$$

4.2 Specific Post-training

Because the target dataset usually concentrates on a specific domain, existing works tend to introduce self-supervised post-training on the target domain in order to incorporate the in-domain knowledge. Here we can also apply the three strategies to the target multi-turn dialogue dataset as self-supervised auxiliary tasks, which jointly learn with the response selection task in Section 4.1.

To conduct the three auxiliary tasks, we should first sample from multi-turn dialogues to build post-training datasets. For the insertion task, there is a little difference from that in Wikipedia. We randomly choose k continuous utterances $\{u_1, u_2, \cdots, u_k\}$ from a dialogue, and fix u_1 but randomly insert u_2 to any interval among $\{u_3, \cdots, u_k\}$. The input sequence is:

X'_{ins} = `[CLS]` `[SOT]` u_1 `[SOT]` $u_3 \cdots$ `[SOT]` u_i `[SOT]` u_2 `[SOT]` $u_{i+1} \cdots$ `[SOT]` u_k `[SEP]`, We calculate the cosine similarity of the `[SOT]` embedding of u_1 with the other following utterances as matching scores to predict the u_2. The loss is denoted as $\mathcal{L}'_{ins}$. Considering the following turn (u_2) tends to be more related to u_1 compared with the next utterance of u_1's speaker (denoted as u_t), we do not predict u_t as what we do in Wikipredia. But they are both expected to recognize the most related utterance with u_1, which helps select the proper response.

For the deletion and replacement tasks, we sample continuous k utterances from one dialogue, and conduct deletion or replacement in the same way as Wikipedia. The post-training losses for both tasks on the target domain are denoted as $\mathcal{L}'_{del}, \mathcal{L}'_{rep}$ respectively.

4.3 Domain Multi-task Learning

We apply the multi-task learning framework on the target domain on our Dialog-PrLM. Carried with dialogue related features from the general corpus, Dialog-PrLM is expected to learn from the target domain together with the target task. We train the response selection task in the same way as 4.1, and denote the loss as $\mathcal{L}'_{reselect}$. The final loss is to sum up response selection loss and the three auxiliary task losses:

$$\mathcal{L}_{final} = \mathcal{L}'_{ins} + \mathcal{L}'_{del} + \mathcal{L}'_{rep} + \mathcal{L}'_{reselect}$$

5 Implementation of Dialogue-oriented Pre-training

For dialogue-oriented pre-training, we sample train and valid datasets for the insertion, deletion and replacement tasks on both English and Chinese Wikipedia. To prevent information leakage (e.g. the model peeks at the correct utterance order from deletion samples when conducting replacement), we divide all the articles equally into three disjoint equal sets to sample from for the three tasks respectively. Data statistics are in Table 2.

Statistics	English		Chinese	
	Train	Valid	Train	Valid
#articles/task	330k	10k	67k	20k
Insertion	1.5M	47k	638k	189k
Deletion	1M	31k	508k	146k
Replacement	1M	32k	542k	160k

Table 2: Statistics for dialogue-oriented pre-training.

For English Wikipedia (totally 1,060,131 articles), we sample from 330,000/10,000 articles for training/evaluation respectively for each task. To ensure data quality, we omit the ”References” and ”Literature” part. For insertion, we sample twice disjointly from one paragraph which has more than 4 sentences, and then sample twice from the next satisfactory one to construct two training samples, which goes on until the end of an article. For deletion and replacement, we sample k continuous sentences as a training sample from each paragraph with more than k sentences. We limit the maximum words of each sample to 400 to prevent overflows after tokenization.

For Chinese Wikipedia (totally 262,405 articles), we sample from 67,468/20,000 articles for training/evaluation respectively for each task. As Chinese corpus is much smaller than the English one, we sample disjoint k continuous sentences as much as we can from each paragraph with more than k sentences for the deletion and replacement tasks. As to insertion, we conduct sampling the same way as English. The maximum length of each sample is limited to 450.

6 Experiments

6.1 Datasets

For the target response selection selection task, our Dialog-PrLM is fine-tuned on three widely used benchmark datasets: (1) **E-commerce Corpus** (Zhang et al., 2018): includes conversations between customers and shopkeepers from the largest e-commerce platform Taobao in China. (2) **Douban Corpus** (Wu et al., 2017): consists of multi-turn conversations from the Douban group, which is a popular social networking service in China. (3) **Ubuntu Corpus (v1.0)**(Lowe et al., 2015) consists of English multi-turn conversations about technical support collected from chat logs of the Ubuntu forum.

As to the three auxiliary tasks for domain multi-task learning, we conduct sampling from the batch when training the response selection task. Dialogues will be neglected if they are less than 3 utterances. Different from general pre-training, we do not require every dialogue to have at least k sentences for all the three tasks.

For evaluation, we use the same metric $R_n@k$ as previous works, which selects k best matchable candidate responses among n and calculates the recall of the true ones. We also use MAP (Mean Average Precision), MRR (Mean Reciprocal Rank), and Precision-at-one P@1 as previous works.

	Model	E-commerce			Douban						Ubuntu		
		R_{10}@1	R_{10}@2	R_{10}@5	MAP	MRR	P@1	R_{10}@1	R_{10}@2	R_{10}@5	R_{10}@1	R_{10}@2	R_{10}@5
	DialoGPT	-	-	-	-	-	-	-	-	-	79.0	88.5	97.1
	TOD-BERT	-	-	-	-	-	-	-	-	-	79.7	89.0	97.4
	BERT-[CLS]	62.7	82.2	96.2	58.7	62.7	45.1	27.6	45.8	82.7	81.9	90.4	97.8
♢	BERT-[SEP]	65.1	84.8	97.4	59.5	63.9	46.0	27.7	46.9	84.3	82.1	90.5	97.8
♢	Dialog-BERT	**66.2**	**85.5**	**97.6**	**60.0**	**64.1**	**46.9**	**28.9**	46.7	83.3	**82.3**	**90.6**	97.7
♣	BERT+multi-task	65.8	84.6	97.6	60.2	64.7	46.9	28.5	48.6	82.5	85.0	92.5	98.3
♣	Dialog-BERT+multi-task	**68.0**	**85.3**	**97.7**	**60.9**	**64.9**	**48.0**	**30.0**	47.9	**82.9**	**85.4**	**92.8**	**98.5**
	ELECTRA-[CLS]	58.2	79.6	96.9	59.0	63.2	44.8	27.6	47.3	82.8	82.5	90.7	97.8
♡	ELECTRA-[SEP]	60.4	80.6	96.3	58.8	62.5	44.2	26.9	46.3	84.1	82.2	90.7	97.8
♡	Dialog-ELECTRA	**61.1**	**81.4**	**96.9**	**59.8**	**64.1**	**46.5**	**28.3**	**47.7**	**84.1**	**83.5**	**91.4**	**98.0**
♠	ELECTRA+multi-task	68.1	86.8	97.9	61.4	65.3	47.5	29.6	50.6	83.8	86.6	93.4	98.5
♠	Dialog-ELECTRA+multi-task	**68.3**	86.3	**98.0**	**61.6**	**65.6**	**48.3**	**30.0**	49.8	**84.7**	**86.8**	**93.6**	**98.6**

Table 3: Response selection results on E-commerce, Douban and Ubuntu datasets. The symbols on the left indicate the corresponding comparison groups. The best results in each group are in boldface.

6.2 Experimental Settings

We work on two different PrLMs: BERT (*bert-base-uncased, bert-base-chinese*) and ELECTRA (*electra-base-discriminator, chinese-electra-180g-base-discriminator*)[1].

For our dialogue-oriented pre-training on Wikipedia, the max input sequence length is set to 512 after WordPiece tokenization. We set the learning rate as 2e-5 with a warmup proportion of 10%. The plain PrLMs are continuously pre-trained with batch size of 8 per task for BERT and 16 per task for ELECTRA. It is trained for 1 epoch and evaluated every 10000 steps. The model with the best average accuracy of the three tasks is saved as Dialog-PrLM. The pre-training experiment needs 4 nVidia RTX 2080 GPUs.

For fine-tuning on our Dialog-PrLMs, the batch size is 32 and the max sequence length is 350. The model is trained for 5 epochs and evaluated after each epoch on the three datasets and both Dialog-PrLMs. Other settings are the same as dialogue-oriented pre-training. For domain multi-task learning on our Dialog-PrLMs, the batch size is 16, and the epoch is 3 for Douban on BERT, 4 for Ubuntu on ELECTRA and 5 for other cases. Other settings are the same as fine-tuning. The k value for both pre-training and domain multi-task learning is 5. The fine-tuning/multi-task learning experiments need 1/2 nVidia RTX 2080 GPUs.

6.3 Results

To verify the effectiveness of our method, we conduct extensive empirical studies on three multi-turn dialogue benchmarks. We are aware that applying complicated matching networks, speaker embeddings (Gu et al., 2020) or other various auxiliary tasks (Whang et al., 2020; Xu et al., 2021a; Whang et al., 2021) would achieve further improvement, but to fairly evaluate the general-purpose pre-training for dialogue tasks, we still follow the standard fine-tuning procedure on Dialog-PrLM by excluding those too advanced auxiliary techniques.

BERT-[CLS]: Each utterance U_i and the candidate response R are concatenated as $[\texttt{CLS}]\, U_1 U_2 ... U_n\, [\texttt{SEP}]\, R\, [\texttt{SEP}]$ and then fed into the pla BERT model. The output embedding of [CLS] is used for classification.

BERT-[SEP]: Rather than just use [CLS], we append [SEP] to each utterance or response to represent the previous sequence: $[\texttt{CLS}]\, U_1\, [\texttt{SEP}]\, U_2 ... [\texttt{SEP}]\, U_n\, [\texttt{SEP}]\, R\, [\texttt{SEP}]$, which is then fed into the original BERT model. The output embeddings of [SEP] are gathered and fed into GRU for a matching vector like Section 4.1.

Dialog-BERT: This model conducts dialogue-oriented pre-training on the original BERT, we fine-tune on our Dialog-BERT through feeding [SOT] embeddings to GRU.

BERT+multi-task: The response selection task is trained with the three auxiliary tasks on target datasets. We also add the special token [SOT], and the only difference from Section 4.3 is that the joint learning is conducted on the original BERT.

Dialog-BERT+multi-task: As described in Section 4.3, we conduct domain multi-task learning on our pre-trained Dialog-BERT.

We also conduct fine-tuning on the English Ubuntu dataset with two dialogue related models: (1) DialoGPT (Zhang et al., 2020b) is an extension of GPT-2 that is pre-trained on Reddit data

---

[1] Both datasets and code are available at https://github.com/xyease/Dialog-PrLM

Model	E-commerce			Douban						Ubuntu		
	R_{10}@1	R_{10}@2	R_{10}@5	MAP	MRR	P@1	R_{10}@1	R_{10}@2	R_{10}@5	R_{10}@1	R_{10}@2	R_{10}@5
Dialog-BERT	66.2	85.5	97.6	60.0	64.1	46.9	28.9	46.7	83.3	82.3	90.6	97.7
w/o Insertion	64.9	83.6	97.7	59.0	63.2	45.1	27.8	46.5	83.2	82.1	90.5	97.8
w/o Deletion	64.2	83.7	97.5	59.0	63.1	44.7	27.5	45.9	83.8	82.1	90.6	97.8
w/o Replacement	64.4	84.7	97.6	59.8	63.6	45.4	28.4	46.8	83.6	82.1	90.4	97.8

Table 4: Ablation results for dialogue-oriented pre-training.

from scratch. (2) TOD-BERT (Wu et al., 2020) is trained on a combination of 9 task-oriented dialogue datasets over BERT and incorporates response selection objective. Experiments on ELECTRA are conduct in the same way with BERT. The results are in Table 3. Below PrLM- denotes BERT or ELECTRA.

Compared to the unsatisfactory results from both DialoGPT and TOD-BERT, it demonstrates the powerfulness and universities of our proposed dialogue-oriented pre-training. Compared with PrLM-[CLS], PrLM-[SEP] performs better in general except a little decrease in Ubuntu and Douban on ELECTRA, which shows that modelling the sequential interaction of the dialogue context and response helps improve performance.

After conducting the dialogue-oriented pre-training on Wikipredia, our Dialog-PrLM achieves further improvement on the three datasets and the two PrLMs, which shows that the three targeted training strategies enables the `[SOT]` token in Dialog-PrLM to grasp dialogue related nature (e.g. speaker-awareness, continuity, consistency) at the same time, so that it is more capable of representing an utterance compared with `[SEP]` in the plain PrLM (PrLM-[SEP]).

When we train the response selection task jointly with the three auxiliary tasks on target datasets, the domain multi-task learning on our Dialog-PrLM (Dialog-PrLM+multi-task) is still always performing better than on the plain PrLM (PrLM+multi-task). Having re-learned broader representation on general corpus, domain post-training further incorporates `[SOT]` with dialogue related feature from in-domain multi-dialogues and thus helps choose the correct response.

Compare PrLM-[SEP] with PrLM+multi-task, Dialog-PrLM with Dialog-PrLM+multi-task, domain multi-task learning indeed achieves improvements due to its incorporated in-domain dialogue related knowledge, which verifies the effectiveness of our proposed three strategies when applying to domain multi-turn dialogue datasets.

In conclusion, conducting dialogue related feature pre-training with our proposed three strategies on Wikipredia (Dialog-PrLM) helps achieve improvements when fine-tuning, and it will further improve when applying these strategies to domain multi-turn dialogues (Dialog-PrLM+multi-task).

7 Analysis

7.1 Ablation Study

In order to investigate the performance of each strategy, we conduct ablation experiments for both pre-training and domain multi-task learning. Results are shown in Tables 4 and 5 respectively.

The results in Table 4 indicate that the insertion, deletion and replacement tasks jointly contribute to the final increase. Influence on Ubuntu seems less than that on Douban and E-commerce, as the Ubuntu corpus contains many terminologies that do not usually appear in general corpora (e.g., apt-get, lsmod and grep) (Whang et al., 2020), so according to BERT+multi-task in Table 3, conducting domain post-training is more effective.

We also do ablation study for Douban on Dialog-BERT in Table 5 to explore the performance of three auxiliary tasks when applying to the target multi-turn dialogue datasets. Similarly, removing any part leads to worse performance, showing the necessity of each task.

Model	Douban				
	MAP	MRR	P@1	R_{10}@1	R_{10}@2
Dialog-BERT+multi-task	60.9	64.9	48.0	30.0	47.9
w/o Insertion	58.2	62.5	44.8	27.4	45.0
w/o Deletion	60.6	64.6	47.1	29.1	48.8
w/o Replacement	60.2	64.1	46.3	29.0	48.5

Table 5: Ablation for domain multi-task learning.

7.2 Utterance Representation Test

We have added a special token `[SOT]` before each utterance or response to represent the following sequence. After pre-training on Wikipedia on PrLMs, `[SOT]` of our Dialog-PrLM is expected to obtain the respective utterance representation through the three dialogue-oriented strategies.

Dialogue Context & Response	`[CLS]`	`[SEP]`	`[SOT]`
Douban			
1: *how about going to Xitang during the Spring Festival?*	0.8424	1	-0.4001
2: *I also want to go that time, but it will be more expensive for accommodation than weekend.*	0.9364	1	**0.8666**
3: *have you contacted about where to live?*	0.9694	1	-0.2902
4: *if you haven't booked accommodation, it seems very difficult to book during the Spring Festival.*	0.9495	1	-0.1643
5: *will there be many people, or no shops open during the Spring Festival?*	0.9024	1	-0.5064
6: *it's okay if no stores open. i enjoy quiet and just want somewhere to drink and eat.*	0.9737	1	-0.1644
7: *but traffic is a hassle during the Spring Festival.*	0.9684	1	-0.6139
8: *there are several people in this group xxx who go to Xitang, you can join them to cook together*	0.8865	1	**0.8840**
Response: *is there anyone in Beijing, let's go together sometime.*			
Ubuntu			
1: *hello*	0.9279	0.9956	-0.9791
2: *what is the best server for update ubuntu*	0.8027	0.9957	-0.9653
3: *whichever is closest to you generally they all have the same content*	0.9738	0.9949	-0.8846
4: *have you read the documentation for the script*	0.9643	0.9966	-0.2836
5: *i know thats the error does the script s documentation indicate what modules or packages are quried to run it*	0.8176	0.9964	**0.5233**
6: *u can tell me how to install python modules and i will install the modules in the python script*	0.8940	0.9977	**0.5287**
7: *where did you get this script*	0.9328	0.9980	**0.5105**
Response: *can you be more specific what is this script supposed to do import logging*			
E-commerce			
1: *please help me issue an invoice with the title of xxx.*	0.8864	1	-0.9791
2: *okay.*	0.8726	1	-0.8137
3: *ok, is it difficult to package?*	0.9249	1	-0.6777
4: *not at all.*	0.7843	1	-0.9359
5: *please send me a picture tutorial later, i will do it myself.*	0.8648	1	-0.3668
6: *okay, if you have any question, please feel free to consult us.*	0.9381	1	0.3211
7: *fine, can you deliver it today?*	0.9265	1	**0.9899**
8: *yes, please check the delivery* address.	0.9240	1	**0.7117**
9: *correct.*	0.6862	1	**0.5704**
10: *ok, if pay before 16:00 today, it will be delivered today, otherwise it will be delivered next day.*	0.9550	1	**0.5866**
11: *i've already paid.*	0.8663	1	**0.0622**
Response: *ok, we will deliver the goods as soon as possible today, please wait at patience.*			

Table 6: Examples from test sets of Douban, Ubuntu and E-commerce respectively. Similarity scores $\geq$ 0.5 of our Dialog-BERT(`[SOT]`) are bold, indicating the corresponding utterances are considered relevant to the response.

To explore the semantic information of `[SOT]` of our Dialog-BERT, we calculate the cosine similarity of the correct response to each utterance in the dialogue context (`[SOT]`). Table 6 lists examples from the three target datasets respectively. For comparison, we use BERT to encode each utterance or response and use `[CLS]` for calculation (`[CLS]`). We also concatenate utterances and response with separation of `[SEP]` on BERT and then split to calculate similarity (`[SEP]`).

We observe that for all examples, both BERT(`[CLS]`) and BERT(`[SEP]`) can not discriminate which utterance is related with the correct answer. All the utterances are treated the same way including the irrelevant ones, which leads to much noise for response selection.

After conducting the dialogue-oriented pre-training on Wikipedia, Dialog-BERT learns a stark sense of ”irrelevant” and ”relevant”. It is able to concentrate on the most critical utterances and distinguish from the irrelevant ones by a large margin. For the example of Douban, Dialog-BERT realizes that the second and last utterance are most relevant. The response asks for someone to travel together, and is related to the second utterance which expresses a wish to go and the last one which gives a group to travel together. Dialog-BERT ignores the noise about accommodation and transportation, and is able to select related utterances among noise rather than just use the last one. For Ubuntu, Dialog-BERT concentrates on utterances about script and ignores the previous background information. For E-commerce, it recognizes the related last few utterances about delivery, and ignores the packaging information before. From examples from target datasets, our Dialog-BERT has absorbed related knowledge from our proposed dialog-oriented pre-training. The `[SOT]` could better represent each utterance, which can be utilized in tasks about representation like multi-party dialogue disentanglement.

8 Conclusion

This paper presents a novel general-purpose solution for dialogue tasks with pre-trained language models. To fill the gap between a detailed task and the LM task of PrLM, we propose dialogue-oriented pre-training on large scale of artificially built dialogue data which lets the resulted Dialog-PrLM enjoy both merits of general-purpose and capturing key dialogue related features including speak awareness, continuity and consistence. Our models are evaluated on three benchmark response selection datasets and achieve consistent performance improvement over the plain PrLMs.

References

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 85–96.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations(ICLR 2020)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019)*, pages 4171–4186.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, pages 2672–2680.

Jia-Chen Gu, Tianda Li, Quan Liu, Zhen-Hua Ling, Zhiming Su, Si Wei, and Xiaodan Zhu. 2020. Speaker-aware bert for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2041–2044.

Jia-Chen Gu, Zhen-Hua Ling, and Quan Liu. 2019. Interactive matching network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2321–2324.

Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. 2020. ConveRT: Efficient and accurate conversational representations from transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2161–2174.

Qi Jia, Yizhu Liu, Siyu Ren, Kenny Zhu, and Haifeng Tang. 2020. Multi-turn response selection using dialogue dependency relations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 1911–1920.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR 2020)*.

Junlong Li, Zhuosheng Zhang, Hai Zhao, Xi Zhou, and Xiang Zhou. 2020. Task-specific objectives of pre-trained language models for dialogue adaptation. *arXiv preprint arXiv:2009.04984*.

Zuchao Li, Zhuosheng Zhang, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. 2021. Text compression-aided transformer encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294.

Junyu Lu, Xiancong Ren, Yazhou Ren, Ao Liu, and Zenglin Xu. 2020. Improving contextual language models for response retrieval in multi-turn conversation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1805–1808.

Siru Ouyang, Zhuosheng Zhang, and Hai Zhao. 2021. Dialogue graph modeling for conversational machine reading. In *Findings of the Association for Computational Linguistics: ACL 2021*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Lstm-based deep learning models for non-factoid answer selection. In *International Conference on Learning Representations (ICLR 2016)*.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. Multi-representation fusion network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 267–275.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, page 2922–2928.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016)*, pages 1442–1451.

Weishi Wang, Steven C.H. Hoi, and Shafiq Joty. 2020. Response selection for multi-party conversations with dynamic topic tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 6581–6591.

Taesun Whang, Dongyub Lee, Chanhee Lee, Kisu Yang, Dongsuk Oh, and HeuiSeok Lim. 2020. An effective domain adaptive post-training method for bert in response selection. In *Proc. Interspeech 2020*.

Taesun Whang, Dongyub Lee, Dongsuk Oh, Chanhee Lee, Kijong Han, Dong-hun Lee, and Saebyeok Lee. 2021. Do response selection models really know what's next? utterance manipulation strategies for multi-turn response selection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2021)*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 917–929.

Yu Wu, Zhoujun Li, Wei Wu, and Ming Zhou. 2018. Response selection with topic clues for retrieval-based chatbots. *Neurocomputing*, pages 251–261.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 496–505.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2017)*.

Ruijian Xu, Chongyang Tao, Daxin Jiang, Xueliang Zhao, Dongyan Zhao, and Rui Yan. 2021a. Learning an effective context-response matching model with self-supervised tasks for retrieval-based dialogues. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*.

Yi Xu, Hai Zhao, and Zhuosheng Zhang. 2021b. Topic-aware multi-turn dialogue modeling. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 55–64.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Chunyuan Yuan, Wei Zhou, Mingming Li, Shangwen Lv, Fuqing Zhu, Jizhong Han, and Songlin Hu. 2019. Multi-hop selector network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 111–120.

Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2020a. Dcmn+: Dual co-matching network for multi-choice reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020b. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL, system demonstration*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation.

In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3740–3752.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020c. Semantics-aware BERT for language understanding. In *the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2021. Retrospective reader for machine reading comprehension. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*.

Zhuosheng Zhang and Hai Zhao. 2021. Advances in multi-turn dialogue comprehension: A survey. *arXiv preprint arXiv:2103.03125*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020. LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4450–4461.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 372–381.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018b. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1118–1127.

Pengfei Zhu, Zhuosheng Zhang, Jiangtong Li, Yafang Huang, and Hai Zhao. 2018. Lingke: a fine-grained multi-turn chatbot for customer service. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations (COLING 2018)*, pages 108–112.

Pengfei Zhu, Hai Zhao, and Xiaoguang Li. 2020. Dual multi-head co-attention for multi-choice reading comprehension. *arXiv preprint arXiv:2001.09415*.

GrantRel: Grant Information Extraction via Joint Entity and Relation Extraction

Junyi Bian [1 8] **, Li Huang** [1 8] **, Xiaodi Huang** [2] **, Hong Zhou** [3]**, Shanfeng Zhu** [4 5 6 7 8]
[1] School of Computer Science, Fudan University, Shanghai 200433, China
[2] School of Computing and Mathematics, Charles Sturt University
Albury, NSW 2640, Australia
[3] Atypon Systems, LLC, UK
[4] Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, China
[5] Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence
(Fudan University), Ministry of Education, Shanghai 200433, China
[6] MOE Frontiers Center for Brain Science, Fudan University, Shanghai 200433, China
[7] Zhangjiang Fudan International Innovation Center, Shanghai 200433, China
[8] Shanghai Key Lab of Intelligent Information Processing,
Fudan University, Shanghai 200433, China
`{zhusf, 20110240003}@fudan.edu.cn, hzhou@atypon.com`

Abstract

As part of scientific articles, grant information refers to funder names and their corresponding grant numbers. Extracting such funding information from articles is of significant importance to both academic and funding bodies. The studies on this topic face two major challenges: 1) no high-quality benchmark datasets; and 2) difficulties in extracting complex relationships between funders and grantIDs. In this paper, we present a novel pipeline framework called GrantRel, which consists of a funding sentence classifier, as well as a joint entity and relation extractor. For this purpose, we manually label two high-quality datasets called Grant-SP and Grant-RE, respectively. In addition, our relation extraction (RE) model uses both position embedding and context embedding in an adaptive-learning way. The experiment results have demonstrated that our model outperforms several state-of-the-art BERT-based RE baselines as higher as 6.5% of F1 scores against the PubMed Central (PMC) test set and 3.5% of that against the arXiv test set.

1 Introduction

As an element of scientific articles, grant information generally includes funder names, grant numbers, and their relations. Specifically, a funder name refers to an agency, organization, or program which provides financial support for the research. A grantID is a numerical string by which to distinguish one grant from another. Such grant source information should be automatically identified. The reasons for this are as follows: (a) The funding bodies need to track their funding statuses; (b) For the academic, the impact of funding agencies in the scientific literature can be measured, and agencies actively supporting specific directions can be identified; and (c) The literature management systems require the funding register information. Therefore, a systematic framework that is capable of automatically extracting grant information from papers is needed.

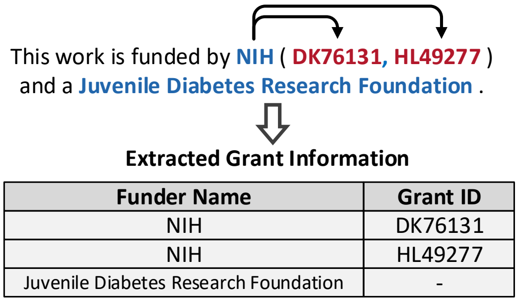
This work is funded by NIH (DK76131, HL49277) and a Juvenile Diabetes Research Foundation .

Extracted Grant Information

Funder Name	Grant ID
NIH	DK76131
NIH	HL49277
Juvenile Diabetes Research Foundation	-

Figure 1: In this sentence, *DK76131* and *HL49277* are two grant numbers of the agency funder *NIH*. The *Juvenile Diabetes Research Foundation* is another funder.

Generally, authors would express their acknowledgments in the papers if their research receives funding. Based on this fact, an extraction should start with selecting the funding sentences from an acknowledgment. To train such a classifier, we manually build a dataset named Grant-SP with 1402 sentences. After that, a relation extraction (RE) model is applied on funding sentences to identify grant entities and their relations. Specifically, a funder name entity and a grant number entity are viewed as a subject and an object in a relation triplet, respectively. For accurately extracting grant information via RE from scientific articles,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2674–2685
August 1–6, 2021.

there are, however, two major challenges: 1) no high-quality RE benchmark datasets; and 2) difficulties in extracting complex relationships between funding organizations and grantIDs by using RE models.

The 2017 BioASQ challenge (Nentidis et al., 2017) is about building a system that extracts the funding information from a benchmark dataset on the full text of biomedical papers. From this dataset, only 107 agencies, however, are required to be identified as funder names such as NIH or CIHR. For example, the winning systems on the challenge such as GrantExtractor (Dai et al., 2018) cannot extract the grant funders beyond 107 agencies such as NASA or JSPS. For overcoming this limitation, we propose a manually-crafted dataset Grant-RE which covers nearly 2k different funder names.

There often exist the complex, many-to-many relationships between funder names and grantIDs. This fact makes it difficult to identify such complex relationships by using a RE model. In addition, the Grant-RE dataset has only two types of entities but with a higher frequency in a sentence, compared with common REs. For example, we count the number of entities with the highest number of occurrences in each sentence of CoNLL04 (Roth and Yih, 2004). The average number of such entities is 2.1, while the number is 2.8 in our Grant-RE dataset. This would be challenging to build correct relations between two entities. Further, a grantID or a funder name could even present independently (see Figure 1). To address this challenge, our GrantRel framework includes a novel joint entity and RE model. This model starts with using the powerful encoding layer of BioBERT (Lee et al., 2019), and can jointly extract funders, grantIDs, and their relations by considering grant relation features. It has been demonstrated that our RE model outperforms the state-of-the-art RE baselines in Grant-RE by a large margin.

In summary, this paper has the following contributions: (a) We propose a novel framework called GrantRel that automatically extracts grant information from academic papers. The RE model in GrantRel is designed to accurately extract both grant number, funder name, and their relation by combining the location of grant information in a sentence and its context embedding in an adaptive way. (b) By manually labelling funding sentences, we retrieved the papers from PubMed Central (PMC) and arXiv, and created a classification dataset called Grant-SP with 1402 sentences for training, as well as a grant RE dataset called Grant-RE with 3331 sentences. (c) Extensive experiments have been conducted to test the performance of the whole framework, and to compare RE models with the RE baselines in both biomedical (PMC) and universal (arXiv) domains.

To the best of our knowledge, this is the first work on reporting a benchmark dataset[1] and model for extracting general grant information by the supervised RE.

2 Related work

The prior studies have addressed the problem of grant information extraction with a limited capability by traditional machine learning methods. A naive Bayes method (Kim et al., 2009) was used to locate the grant support (GS) zone from an article text, followed by inferring GS types with a pattern matching method. As such. only fourteen GS types can be identified. Zhang et al. (2009) used a semi-supervised method to detect grant-related zones from online medical articles. Gross et al. (2016) proposed a rule-based model for extracting metadata (grant number and grant sponsor) from articles. All these methods do not establish a specific relationship between a funder and a grant number.

Recently, Dai et al. (2018) built a pipeline system for grant information extraction. They first selected funding sentences by relying on manually designed features, then extracted grantIDs by using the BiLSTM-CRF tagger, finally identified the agencies by applying a multi-class classifier to each grantID with manually designed features. However, this method is still limited, because it cannot recognize new grant agencies other than 107 designated ones. In contrast, GrantRel learns a joint model on the name recognition of funder and grantID, and extraction of their relationship. As such, it can handle new funder names very well.

Traditionally, RE is achieved through a pipeline (Zelenko et al., 2003; Chan and Roth, 2011; Zhou et al., 2005) with two phases: entity recognition and relation classification. Since the two phases may benefit from the use of correlated signals, research for joint entities and relation extraction have attracted more attention. Early work of joint approaches uses feature-based models (Yu and Lam, 2010; Miwa and Sasaki, 2014). Recently, neural network-based models (Zeng et al., 2018; Li et al.,

---

[1] https://github.com/Eulring/GrantRel

2019; Dai et al., 2019; Fu et al., 2019), especially the BERT-based (Devlin et al., 2019) models (Wei et al., 2020; Eberts and Ulges, 2019; Wang et al., 2020) that replace the manually constructed features with learned representation, have achieved the considerable success in completing the RE task. Following this idea, our RE model uses BioBERT (Lee et al., 2019) as an encoding core. Inspired by the CasRel (Wei et al., 2020) further, our RE model establishes a relation as a function that maps funder to grantID. Since an ordinary model cannot accurately distinguish the complex relationship between multiple funders and grantIDs, the features that can describe the interaction between entities become critical. Therefore, we use relative position embedding and localized context embedding (Eberts and Ulges, 2019), which make a significant improvement. In addition, we design a mechanism by adaptively integrating the two embeddings to obtain better performance.

3 Dataset description

Although BioASQ 5c provides a dataset of grant information extraction, it has three serious drawbacks, 1) with only 107 agency names used in the labels, many common funder names are ignored. In fact, there are nearly 57000 different funder names in a funder name database downloaded from crossref [2]; 2) normalized agency names and the corresponding grantIDs are provided without specifying their exact positions in the articles, which is inconvenient for supervised RE training; 3) the quality of annotation is limited (Dai et al., 2018). To address these issues, we therefore manually built two datasets, namely, Grant-RE and Grant-SP, for the two modules in our framework.

3.1 Dataset: Grant-RE

Grant-RE is the dataset for the RE model. We downloaded articles with the original xml format from open access subset of PMC [3]. The raw text from the acknowledgement section of an article was then parsed into readable paragraphs, and the sentences were split by using NLTK[4] tools. We manually selected the funding sentence and labelled grant information. A grant relation is represented as four integers for the intervals of a funder entity and a grantID entity.

As given in Table 1, we present the statistics of the train/dev/test splits for the grant information extraction dataset. There are two versions of test splits. One is from PMC, which is as same as train/dev split, while another from arXiv is used for conducting evaluations of our approaches on the universal domain. To ensure quality, the GrantRE dataset was annotated by 4 well-trained annotators, with each sentence being annotated twice by different annotators. For those sentences having different annotations, we will seek advice to experts to decide their final annotations. Besides, the test data splits were repeatedly checked 3 times.

	train	dev	test	test^{a}
# sentence	2104	477	500	350
# funder entity	4592	1192	1297	706
# grantID entity	4195	1084	1116	646
# grant relation	4107	1097	1179	684

Table 1: Statistics of Grant-RE. Test, train, and dev sets are from PMC. The test^{a} is from arXiv papers.

3.2 Dataset: Grant-SP

Unlike Grant-RE, we sampled sentences from all sections in a paper to annotate a funding sentence classification dataset. Because the numbers of positive and negative sentences were unbalanced, we discarded most of the negative sentences in the train/dev set to accelerate the training.

The test set in Grant-SP is used not only for the classifier evaluation, but also for the whole framework evaluation. For building the test set, we strictly followed our framework pipeline: for each article, we kept all negative sentences, and tagged grant information for positive sentences. Because the classifier has a high recall, when labeling the test split, we borrow the outputs from trained models for the auxiliary reference. For a sentence that the classifier considers to be positive and the RE model can also extract information, we manually relabel it. In Table 2, we report the statistics of train/dev/test splits.

	train	dev	test
# sentence	908	282	16069
# positive	158	51	101
# articles	-	-	50

Table 2: Statistics of Grant-SP

---

[2] https://gitlab.com/crossref/open_funder_registry

[3] https://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/

[4] https://www.nltk.org/

4 Methodology

4.1 Framework

As shown in Figure 2, the left side illustrates the overall workflow of our GrantRel. Given prepossessed sentences from raw articles, the sentence classification module selects the sentences that may contain grant information. Without this step, the framework may suffer from low precision. After this, the RE module will extract grant information.

4.2 Identification of funding sentences

Our models use a pre-trained BioBERT (Lee et al., 2019) to encode context information. Suppose sentence $\mathbf{x}$ is first tokenized into byte-pair encoded (BPE) tokens (Sennrich et al., 2016) $x = \{x_1, x_2, ..., x_l\}$ with length l. BioBERT takes it as an input and outputs a length of $l+2$ embedding sequence $\boldsymbol{e} = \{\boldsymbol{e}_{CLS}, \boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_l, \boldsymbol{e}_{SEP}\}$. The additional embedding $\boldsymbol{e}_{CLS}$ captures the whole sentence context. A Logistic Regression is then used to calculate the probability:

$$p_{sent} = \sigma(W_{sent}\boldsymbol{e}_{CLS} + b_{sent}) \quad (1)$$

Here the $\sigma(\cdot)$ is the sigmoid function, and $\{W_{sent}, b_{sent}\}$ are trainable parameters.

4.3 Joint entity and RE

A grant relation consists of a funder (subject eneity s) and grantID (object entity o). Given input sentence $\mathbf{x}$ and its tokens x, we use T to represent the set of all grant relations of this sentence. The likelihood of all relations $T = \{(s, o)\}$ in this sentence can be written as:

$$\prod_{(s,o)\in T} p(s, o|x) = \prod_{s\in T}\left[p_{fd}(s|x)\prod_{o\in T|s} p_{gr}(o|s, x)\right] \quad (2)$$

In Eq.(2), the role of $p_{fd}(s|x)$ acts as a subject tagger that recognizes funder name entities in the sentence, where $s \in T$ denotes a subject appearing in T. $p_{gr}(o|s, x)$ is to identify the object with only having a relation with the specific s. $o \in T|s$ is the object in T led by subject s. Indeed, this extracting scheme allows us to extract the grantID at once for each funder name. To handle independent grantIDs, we add an additional probability item p_{id} to tag grantID. As such, the overall likelihood of grant information in x is:

$$\prod_{s\in T}\left[p_{fd}(s|x)\prod_{o\in T|s} p_{gr}(o|s, x)\right]\prod_{o\in T} p_{id}(o|x) \quad (3)$$

4.3.1 Funder name detection

The low-level tagging module aims to detect all possible funder entities from x. Similar to sentence classification, BioBERT (Lee et al., 2019) generates the tokens representation $\boldsymbol{e}$. Using the IOB tagging scheme, we predict the IOB tag y for each token. A specific operation on the i^{th} token is as follow.

$$y_i = \text{softmax}(W_{fd}\boldsymbol{e}_i + b_{fd}) \quad (4)$$

4.3.2 Grant relation detection

A funder name is either extracted at the first phase or provided by the dataset during the training. The conditional grant number tagger distinguishes the grantID that only belongs to this particular funder name from other candidates. We first use a fused BERT embedding $\boldsymbol{e}_{fd}$ to represent this funder name:

$$\boldsymbol{e}_{fd} = f_{fd}(\boldsymbol{e}, \boldsymbol{u}_{fd}) \quad (5)$$

where $\boldsymbol{u}_{fd} = [u^{start}_{fd}, u^{end}_{fd}]$ is the position boundary of a funder name entity. Since the length of the funder name can vary, function $f_{fd}(\cdot)$ is used to produce a fixed-size feature for funder names. On choosing $f_{fd}(\cdot)$, we use the average pooling of the entire entity span. For each token, the grant relation module classifies tag $\boldsymbol{z}$ as :

$$z_i = \text{softmax}(W_{gr}[\boldsymbol{e}_{fd}, \boldsymbol{e}_i, \boldsymbol{e}_{gr}] + b_{gr}) \quad (6)$$

where $\boldsymbol{e}_i$ is the encoding of token x_i, and $\boldsymbol{e}_{gr}$ is the grant relation feature explained below (Senction 4.4).

4.3.3 GrantID detection

If a funder name is undetected in the previous step, we will miss the corresponding grant numbers. In addtion, some grant numbers even occur independently for some reasons, such as a sentence segmentation error. For extracting the complete grant information, an auxiliary item $p_{id}(o|x)$ is used to tag all grantIDs . We view the detection of grantIDs as a special case of the grant relation detection by using trainable vector $\hat{\boldsymbol{e}}$ to represent all funder names.

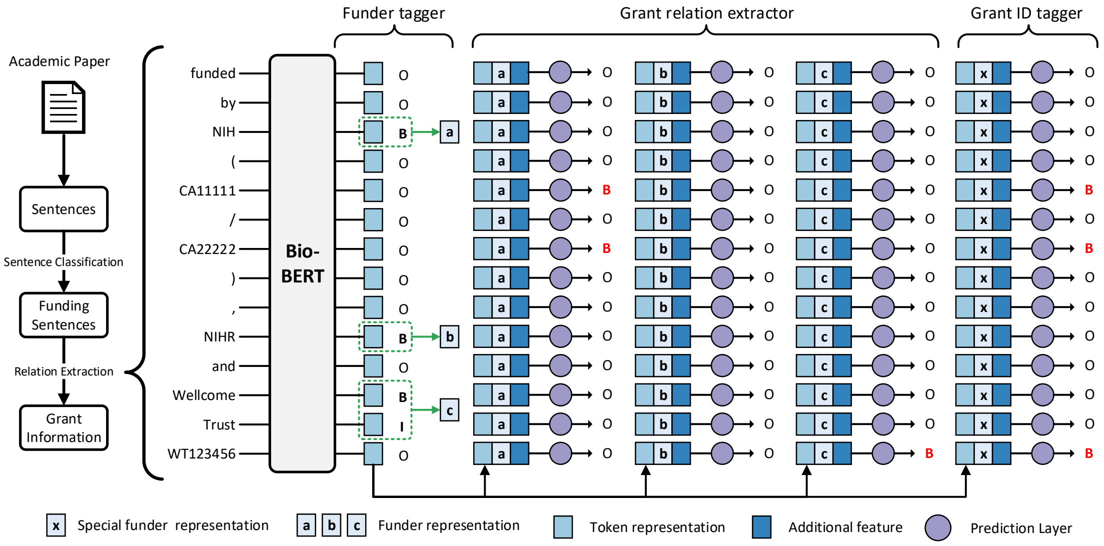

Figure 2: The left side is the grant information extraction pipeline, with the RE model in GrantRel on the right side. For convenience, words are presented without tokenization. The three funders in the sentence are detected by the funder tagger. For each funder name, its corresponding grantID is matched by predicting its label in each position. Note that an ID tagger is able to find all grant numbers at once.

This means that all grantIDs in the sentence should match this special funder name. The operation on the i^{th} token is as follows.

$$o_i = \mathrm{softmax}(W_{gr}[\hat{\boldsymbol{e}}, \boldsymbol{e}_i, \boldsymbol{e}_{gr}] + b_{gr}) \qquad (7)$$

4.4 Grant relation feature

To establish the correct connection between a grantID and a funder name, we use additional features $\boldsymbol{e}_{gr}$ other than entity representation, which characterize the relation between the funder name and the i^{th} token in x in Eq 6. These features can be captured by using information such as the span of funder $\boldsymbol{u}_{fd}$ and input context $\mathbf{x}$.

4.4.1 Position embedding

First, we use the relative distance to measure the two positions:

$$d(i, j) = min(max(-k, (i - j)), k) \qquad (8)$$

where the distance is clipped into a region of $[-k, k]$. The position of an extracted funder entity is an interval $\boldsymbol{u}_{fd}$. Some funder names have relative long spans, so it would be inaccurate to represent all the distances by a single number. We concatenate two relative distance embedding as our final position embedding:

$$\boldsymbol{e}_{pos} = [\mathrm{emb}(d(u_{fd}^{start}, i)), \mathrm{emb}(d(u_{fd}^{end}, i))] \quad (9)$$

where $\mathrm{emb}(\cdot)$ represents a learnable embedding.

4.4.2 Context embedding

We observe that the context for the funder and target token has semantic information that is helpful for establishing relationships. Therefore, we utilize $\boldsymbol{e}$ to represent context embedding $\boldsymbol{e}_{ctx}$. For example, a sentence is: *"funded by NIH (CA123456), and CIHR (R01 12111)"* During the grant relation phase, the subject funder name is *"NIH"*, the target token is *"12111"*, their localized context is the blue part of *"(CA123456), and CIHR (R01"* in the sentence. The max-pooling for encoding $\boldsymbol{e}$ of the localized context is used to generate a fix-size representation $\boldsymbol{e}_{ctx}$.

4.4.3 Adaptive embedding

A combination of two embeddings of position and context can make our model more robust. Furthermore, when the context meaning is abundantly clear, we expect the proposed model can concentrate more on the context information. According to this view, we propose a mechanism that can balance two embeddings to deal with different situations in an adaptive way:

$$\boldsymbol{e}_{gr} = \alpha \cdot \boldsymbol{e}_{pos} + (1 - \alpha) \cdot \boldsymbol{e}_{ctx} \qquad (10)$$

where α is a scalar decided by the context embedding as:

$$\alpha = \sigma(W_{ada}\boldsymbol{e}_{ctx} + b_{ada}) \qquad (11)$$

Test Set	Models	Funder Entity			GrantID Entity			Grant Realtion		
		Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
PMC	CasRel(Wei et al., 2020)	-	-	-	-	-	-	71.7	49.0	58.2
	SpERT(Eberts and Ulges, 2019)	88.6	93.0	90.7	89.8	97.9	93.6	82.6	90.3	86.3
	GrantRel-base	91.9	92.9	92.4	92.8	97.7	95.2	68.8	56.9	62.3
	GrantRel-pos	91.9	93.0	92.5	96.1	97.8	96.9	88.9	87.4	88.1
	GrantRel-ctx	91.3	92.3	91.8	95.4	97.8	96.6	90.0	89.6	89.8
	GrantRel	91.8	93.0	92.4	96.4	97.3	96.8	92.7	89.7	**91.2**
	GrantRel$_{BERT}$	91.7	92.0	91.9	95.3	97.2	96.2	91.4	89.2	90.3
arXiv	CasRel(Wei et al., 2020)	-	-	-	-	-	-	70.1	39.0	50.1
	SpERT(Eberts and Ulges, 2019)	82.5	85.9	84.1	86.5	94.3	90.3	76.6	81.8	79.1
	GrantRel-base	83.6	82.9	83.2	90.3	97.2	93.6	66.8	49.7	57.0
	GrantRel-pos	85.8	85.7	85.8	92.5	97.1	94.7	87.0	81.0	83.9
	GrantRel-ctx	85.4	85.4	85.4	93.9	96.8	95.3	85.1	85.1	85.1
	GrantRel	86.2	85.9	86.0	93.9	96.6	95.2	86.9	84.3	**85.6**
	GrantRel$_{BERT}$	86.3	83.6	84.9	90.7	96.0	93.3	83.8	80.1	82.0

Table 3: The performance of GrantRel compared with typical RE models on the test sets of PMC and arXiv.

5 Experiments

In this section, we compare the performance of the GrantRel RE model with several RE baselines on the Grant-RE dataset. The varying degree of the improvement of the RE model with different features is also examined. Finally, the overall performance of the proposed GrantRel framework is comprehensively evaluated.

5.1 Experiment settings

In Table 3, we define **GrantRel-base** as the pure RE model without considering additional features. Compared to GrantRel-base, **GrantRel-pos** makes use of the position embedding, while **GrantRel-ctx** uses context embedding. As our ultimate model, GrantRel integrates two embeddings of position and context in an adaptive way. These models both initially encode the input by using the BioBERT pretraining. In particular, **GrantRel**$_{BERT}$ uses the BERT encoding for a fair comparison with other BERT-based baselines: **CasRel** (Wei et al., 2020) the state-of-the-art model of WebNLG (Gardent et al., 2017) and NTY(Riedel et al., 2010) dataset, and **SpERT** (Eberts and Ulges, 2019) the state-of-the-art model of CoNLL2004 (Roth and Yih, 2004) dataset. In order to use the SpERT in Grant-RE, we extend the max span size from the original one of 20 to 25. This increases the training time, but covers the widest span of funders in our dataset. Other baselines settings strictly follow the optimal settings of the original paper.

We used Pytorch to implement the deep learning models. All GrantRel models were trained by using Adam (Kingma and Ba, 2015) optimizer. During the training, the number of epochs was chosen as 30, and the learning rate dropped 20% in every two epochs with an initial learning rate of 5e-5. In addition, the distance threshold k in position embedding was set to 40, together with the batch size of 10, and the dimension of context and position embedding of 768. All of our experiments were conducted on a single GTX 1080Ti GPU.

5.2 Evaluation metrics

In this work, we use f1-score (F1), precision (Prec.), and recall (Rec.) to measure the performance of our models on extracting grant relation, grant number, and funder entities. For all the evaluations, a predicted entity is correct only if both its head and tail are correct.

Grant relation evaluation: For relation evaluation, we tested only the triplets with a complete grantID and funder name in the test dataset by excluding isolated funder names or grantIDs. This also held true for the other RE tasks.

Grant information evaluation: Grant information evaluation aims to test the overall performance for our GrantRel framework. Differing from relation evaluation, the overall evaluations include isolated funder names and grant numbers.

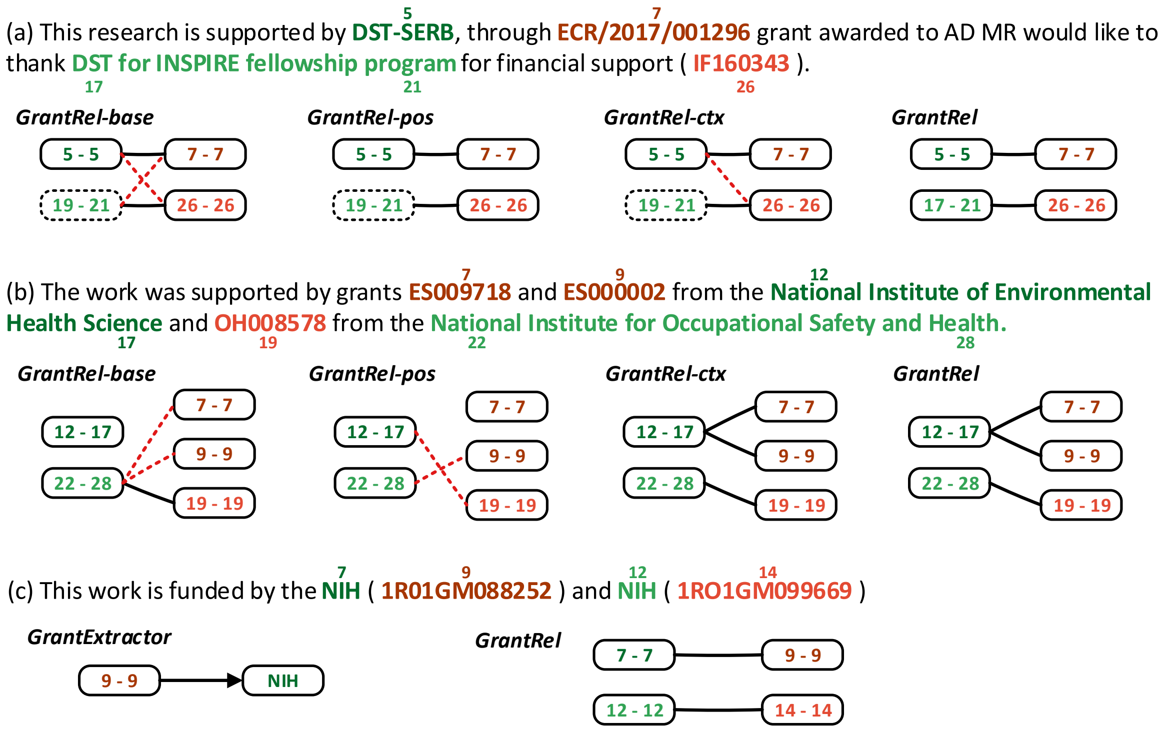

Figure 3: Outputs from different models. Each circle represents an entity, while the number in a circle represents the entity's span positions. A dotted circle indicates an output of a wrong entity span. A red dotted line indicates a wrong relationship. The arrow in case (3) means that the funder was directly inferred from GrantID. In all of these examples, GrantRel outputted correct results.

5.3 Experiment Results: Grant-RE

The experiments here focus only on the RE model, with the funding sentences provided. Main results on the Grant-RE dataset are shown in Table 3. We have four main findings. (1) GrantRel achieves the best performance on both PMC and arXiv test splits, with an increase of 3.9% and 6.5% respectively compared with other baselines. (2) Grant relation features are critically important. Without adding additional features that characterise the relationship between a funder name and a grant number, the GrantRel-base model and CasRel have a bad performance. When the position embedding was integrated (GrantRel-pos), the f1-score, however, increase significantly with 27.5% improvements. Context embedding(GrantRel-ctx) perform better than position embedding by another increase of 1.7%. SpERT using a context embedding also has considerable performance(86.3%). Further, the combination of context embedding and position embedding in GrantRel produce the best f1-score 91.2%. (3) GrantRel$_{BERT}$ perform worse than GrantRel in both test sets. Which means that BioBERT, as an encoding layer, performs better than BERT in terms of grant information extraction. The reason for this is that BERT was trained only from wiki and books, but BioBERT was trained on additional scientific papers. (4) When tested on a new domain (arXiv), the performance of all models dropped slightly. This is because most funder names in the arXiv test set are different from those in PMC.

5.4 Experiment Results: Grant-SP

Before applying relation extraction, we first identify which sentence in a given paper is grant-related by using the sentence classifier. In this experiment, two models are combined into a pipeline. If a sentence is predicted as negative by the classifier, we will exclude it from relation extraction. As we know, the best RE model from Section 5.3 is the downstream module. To verify the effect of the funding sentence classifier, we compared our GrantRel (Clf+RE) with the framework without classifier (RE), framework with key-words sentence matching (Key+RE), and framework with perfect classifier (Gold+RE), respectively. The experiment results are reported in Table 4. Since we discarded most of the negative samples in training, our funding sentence classifier had achieved a

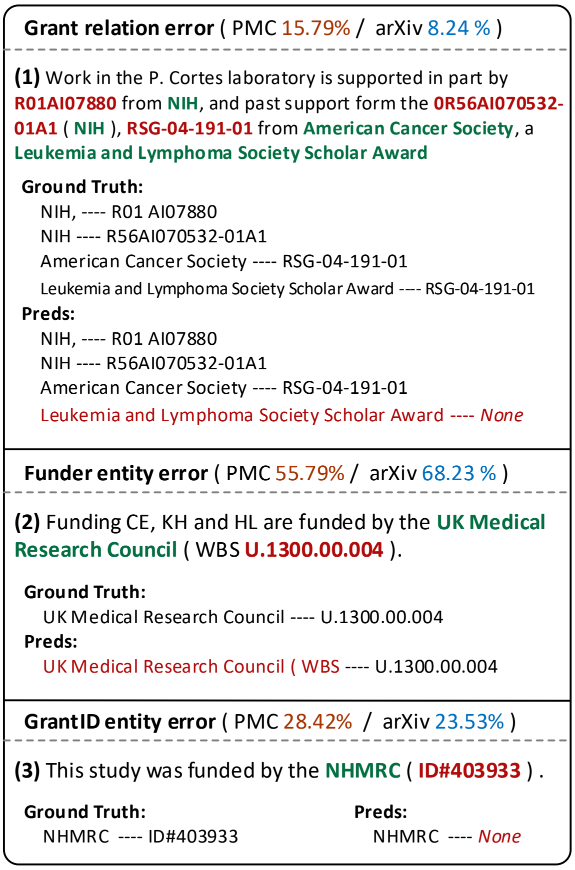

Grant relation error (PMC 15.79% / arXiv 8.24 %)

(1) Work in the P. Cortes laboratory is supported in part by **R01AI07880** from **NIH**, and past support form the **0R56AI070532-01A1** (**NIH**), **RSG-04-191-01** from **American Cancer Society**, a **Leukemia and Lymphoma Society Scholar Award**

Ground Truth:
NIH, ---- R01 AI07880
NIH ---- R56AI070532-01A1
American Cancer Society ---- RSG-04-191-01
Leukemia and Lymphoma Society Scholar Award --- RSG-04-191-01

Preds:
NIH, ---- R01 AI07880
NIH ---- R56AI070532-01A1
American Cancer Society ---- RSG-04-191-01
Leukemia and Lymphoma Society Scholar Award ---- *None*

Funder entity error (PMC 55.79% / arXiv 68.23 %)

(2) Funding CE, KH and HL are funded by the **UK Medical Research Council** (WBS **U.1300.00.004**).

Ground Truth:
UK Medical Research Council ---- U.1300.00.004

Preds:
UK Medical Research Council (WBS ---- U.1300.00.004

GrantID entity error (PMC 28.42% / arXiv 23.53%)

(3) This study was funded by the **NHMRC** (**ID#403933**) .

Ground Truth:
NHMRC ---- ID#403933

Preds:
NHMRC ---- *None*

Figure 4: Example of error cases from the GrantRel RE model. There are three types of errors, each of which is statistically analyzed on PMC test set and arXiv test set.

Pipelines	Grant Sent. Prec.	Grant Sent. Rec.	Grant Sent. F1	Grant Info. Prec.	Grant Info. Rec.	Grant Info. F1
RE	-	-	-	12.0	94.9	21.3
Key + RE	51.0	74.3	60.5	86.1	68.9	76.5
Clf + RE	85.6	100	**92.2**	85.7	93.3	**89.4**
Gold + RE	100	100	100	89.8	93.3	91.6

Table 4: The pipeline performance on Grant-SP. Clf+RE is the GrantRel framework; Key+RE selects the funding sentence by keywords matching; Gold+RE uses the ground truth to select funding sentence; and RE extracts grant information on each sentence.

very high recall. Compared with RE and Key+RE models, the framework with the sentence classifier achieved a significantly higher precision. Meanwhile, the sentence classifier could reduce search costs. In our experiments, the RE model could process 25 sentences per second. In contrast, our framework could process 50 sentences per second by filtering out the non-funding sentences.

5.5 Case study

We review the results from different models and select some cases for further analysis in this section.

First, we examine the results from RE models with different features in Figure 3. In case (a), only the GrantRel identified correct funder names and grant relations. The base model GrantRel-base matches each agency to all grant numbers. GrantRel-pos produced the correct relation. However, GrantRel-ctx built the wrong connection between *DST-SERB* and *ID160343*. We speculate that the context information for the entity and the ID may not work. But, the distance between the two entities is too long. As a result, only models that incorporate position information output the correct relation. In case (b), GrantRel-base still had terrible performance. For the sentences with grantIDs that are located at the front of their corresponding funders, GrantRel-pos performed poorly. Nevertheless, this case can be easily handled by considering context information as does in our framework. By analysis, we find that the base model intends to predict whether a funder is associated with numbers first. If it is, the funder will be established the relations with all found grantIDs. If not, the funder will be regarded as isolated. Context embedding can build relations in a complicated semantic situation. Position embedding is particularly helpful when context embedding is inadequate or ambiguous. In case (c), we compare our framework with GrantExtractor (Dai et al., 2018). GrantExtractor can only extract grant number *1R01GM088252* from the sentence and infer the *NIH* by this ID. However, it even misses the number *1RO1GM099669* if the char “0” is wrongly spelled as “O”. It is easy for our model to identify such error-spelled grantIDs.

Second, we carry out the error analysis on wrong cases by GrantRel (see Figure 4). In case (1), grantID *RSG-04-191-01* is related not only to *American Cancer Society*, but also to *Leukemia and Lymphoma Society Scholar Award*. But the RE model treated the following entity as an independent funder. Such an example requires the model to have a deeper understanding of semantic information. Moreover, training data lacks such a kind of samples which make the RE model more difficult to extract. In case (2), GrantRel wrongly recognized the funder name, and this kind of error accounts for the majority. In case (3), GrantRel failed to find the grant number. This can be explained by the fact that the *"ID"* mostly appears independently in training without being tagged as a number entity. Such errors can be corrected by using more fine-grained tokenization.

6 Conclusion

In this paper, we have presented a novel pipeline framework named GrantRel for automatically extracting grant information from academic articles. The framework has two components of the text classification module and the joint RE module. Moreover, we manually labelled two datasets for training and testing modules. Compared to the previous approaches, the proposed framework has achieved significant improvements in extracting any types of funder names mentioned in articles. Overall, the experiments have demonstrated that our RE model outperforms several state-of-the-art baselines of grant extraction.

7 Acknowledgments

This study has been supported by Shanghai Municipal Science and Technology Commission [2018SHZDZX01], ZJ Lab, Shanghai Center for Brain Science and Brain-Inspired Technology, 111 Project [B18015] and National Natural Science Foundation of China (No.61872094). This research is also sponsored by Atypon Systems, LLC.

8 Ethics Statement

Datasets have been collected in a manner which is consistent with the terms of use of any sources and the intellectual property. For each annotator, we compensate based on the number of annotated sentences. More details of our datasets are depicted in Section 3.

References

Yee Seng Chan and Dan Roth. 2011. Exploiting Syntactico-Semantic Structures for Relation Extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560.

Dai Dai, Xinyan Xiao, Yajuan Lyu, Qiaoqiao She, Shan Dou, and Haifeng Wang. 2019. Joint Extraction of Entities and Overlapping Relations using Position-Attentive Sequence Labeling. *AAAI 2019 : Thirty-Third AAAI Conference on Artificial Intelligence*, 33(1):6300–6308.

Suyang Dai, Zihan Zhang, Wenxuan Zuo, Xiaodi Huang, and Shanfeng Zhu. 2018. Grantextractor: A winning system for extracting grant support information from biomedical literature. In *IEEE International Conference on Bioinformatics Biomedicine*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Markus Eberts and Adrian Ulges. 2019. Span-Based Joint Entity and Relation Extraction with Transformer Pre-Training. In *ECAI*, pages 2006–2013.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling Text as Relational Graphs for Joint Entity and Relation Extraction. In *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating Training Corpora for NLG Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 179–188.

Mark Gross, Tammy Bilitzky, and Richard Thorne. 2016. Extracting Funder and Grant Metadata from Journal Articles. *Balisage: The Markup Conference*.

Jongwoo Kim, Daniel X. Le, and George R. Thoma. 2009. Inferring grant support types from online biomedical articles. In *2009 22nd IEEE International Symposium on Computer-Based Medical Systems*, pages 1–6.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015 : International Conference on Learning Representations 2015*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-Relation Extraction as Multi-Turn Question Answering. In *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling Joint Entity and Relation Extraction with Table Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869.

Anastasios Nentidis, Konstantinos Bougiatiotis, Anastasia Krithara, Georgios Paliouras, and Ioannis A. Kakadiaris. 2017. Results of the fifth edition of the BioASQ Challenge. In *BioNLP 2017*, pages 48–57.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECMLPKDD'10 Proceedings of the 2010th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, pages 148–163.

Dan Roth and Wen-tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. 2020. TPLinker: Single-stage Joint Extraction of Entities and Relations Through Token Pair Linking. In *COLING*, pages 1572–1582.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A Novel Cascade Binary Tagging Framework for Relational Triple Extraction. In *ACL 2020: 58th annual meeting of the Association for Computational Linguistics*, pages 1476–1488.

Xiaofeng Yu and Wai Lam. 2010. Jointly Identifying Entities and Extracting Relations in Encyclopedia Text via A Graphical Model Approach. In *Coling 2010: Posters*, pages 1399–1407.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(6):1083–1106.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism. In *ACL 2018: 56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 506–514.

Xiaoli Zhang, Jie Zou, Daniel X. Le, and George R. Thoma. 2009. A semi-supervised learning method to classify grant support zone in web-based medical articles. In *Proceedings of SPIE, the International Society for Optical Engineering*, volume 7247.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring Various Knowledge in Relation Extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434.

A Tagging Standard

In the process of dataset construction, it is a challenge to set a standard for annotations, especially for determining funder entities. After reviewing lots of examples, we decided to use the following rules to determine a funder entity in our tagging.

- Apart from agencies, specific programs, awards, foundation, and fellowships are also regarded as funder names.
- If the name of a program, or fellowship, or award, etc., is associated with the corresponding agency, we will treat them as a whole funder name.
- The address or abbreviation associated with a funder name will be included as part of its funder name.
- The sub-division associated with an agency is viewed as part of the funder name.

B Performance Impact of the Funder Representation

In Table 5, we examine the performance under different funder representations e_{fd}. The following RE models all adopted a standard GrandRel structure (Using the adaptive embedding), with differing only in their representation approaches of funder names

	Grant Realtion		
Funder Representation	Prec.	Rec.	F1
Head	91.71	90.00	90.85
Head+Max	91.93	89.83	90.87
Head+Mean	91.89	90.25	91.06
Head+Tail	91.46	88.90	90.16
Max	92.56	89.58	91.04
Mean	92.65	89.75	91.18

Table 5: Results of GrantRel with different funder representations with respect to the PubMed test set.

- **Head**: The funder entity representation uses the first token representation.
- **Head+Max**: The max-pooling of the entity span representation metric concatenates with the first token representation to represent the whole entity.

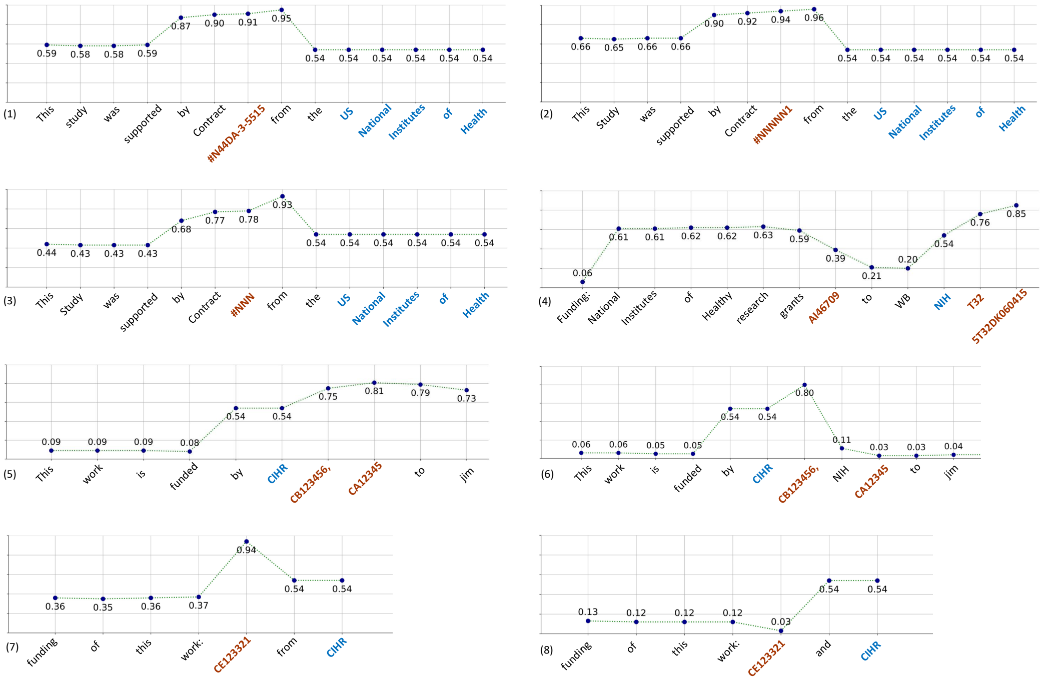

Figure 5: In each sentence, the blue-colored word is the selected funder entity, and the red-colored word is all grant numbers in the sentence. The float number on each word represents its alpha value when calculating the adaptive embedding under the blue-colored funder.

- **Head+Mean**: The average-pooling of the entity span representation metric concatenates with first token representation to represent the whole entity.

- **Head+Tail**: The first token representation concatenates the last token representation.

- **Max**: The max-pooling of the entity span representation metric.

- **Mean**: The average-pooling of the entity span representation metric.

It is observed that the average-pooling of the entity span has the best performance. Hence, we adopted this funder representation method in all our experiments.

C Performance Impact of the Adaptive Mechanism

Our adaptive embedding approach (GrantRel) were compared with the simple fuse approach (GrantRel pos+ctx), which merges both position embedding and context embedding by simply adding them. The results in Table 6 show GrantRel is slightly better.

As shown in Figure 5, we further analyze the impact of α on the embedding by using some chosen samples. For each sentence, given a funder entity being contained in this sentence, GrantRel calculated the value of α among all positions in Eq. (11).

	Grant Realtion		
Model name	Prec.	Rec.	F1
GrantRel pos+ctx	92.63	89.49	91.03
GrantRel	92.65	89.75	91.18

Table 6: Comparisons between GrantRel and GrantRel(pos+ctx) against the test set ofPubMed relation extraction .

For cases (1)-(4), we examine the impact of position embedding. As such, the outputs of GrantRel are compared with those of GrantRel(pos+ctx). In sentence (1), both GrantRel and GrantRel(pos+ctx) could recognize the grant number, but GrantRel(ctx) could not. Besides, we can see that the α value is high for grant number *"#N44DA-3-5515"*. In sentence 2, we manually built a case by replacing the GrantID with a more pseudo one. At a result, GrantRel still identified it as a grant number. But the GrantRel(pos+ctx) whose alpha value is always 0.5 did not recognize. In case (3), without Arabic chars in *"#NNN"*, GrantRel did not identify it as an ID even with a high α

value, either. We can conclude that if a token is close to the funder entity, and the alpha has a high value, the model tends to label an ID-like token into a GrantID. In case (4), GrantRel(pos-ctx) wrongly distributed “AI46706” to “NIH”. In contrast, GrantRel assigned a low α value to “AI46706” according to its context of“to WB” and thus discarded this wrong relation.

In cases (5) to (8), we further explore the impact of different factors, which may influence the α value. For case (5) and case (6), the α values on grantID *“CA12345”* differ largely. But the only difference is that there is an agency of *“NIH”* in (6) between two IDs. We find that α dramatically decreases if the local context has other funder names. In cases (7) and (8), we find that some words can also reduce the α value except for funder names. Thus, the model should automatically pay more attention to context information. For example, the word “and” in (8) means that the previous grant information is parallel to the following grant information. Hence the model did not establish a connection between “CE123321” and “CIHR”.

Enhancing Language Generation with Effective Checkpoints of Pre-trained Language Model

Jeonghyeok Park and Hai Zhao[*]
Department of Computer Science and Engineering, Shanghai Jiao Tong University
Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
`117033990011@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn`

Abstract

This work empirically explores effective exploiting of intermediate output from pre-trained language models (PrLMs) for language generation tasks. For this purpose, we propose an improved method to integrate public checkpoints of PrLMs for the most convenience and perform extensive experiments on 6 different kinds of PrLMs, including BERT, ELECTRA, GPT2, Multi-lingual BERT, and XLM RoBERTa. Evaluation with automatic metrics shows that our approach significantly improves the generation quality on the generation tasks, up to 1.8 BLEU points for neural machine translation (Korean-to-English, Korean-to-Chinese) and 1.8 ROUGE points improvements for text summarization.

1 Introduction

Pre-trained Language Models (PrLMs), such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ELECTRA (Clark et al., 2020), have thoroughly changed the landscape of state-of-the-art performance on many Natural Language Understanding (NLU) tasks. Also, publicly released checkpoints of the PrLMs allow natural language processing (NLP) researchers to gain SOTA results while saving vast compute and time resources. The widely used method to exploit PrLM is *fine-tuning*. However, for Natural Language Generation (NLG) tasks, such methods do not get as much performance gain as in the NLU task. Several previous studies proposed methods that better use prior knowledge of the PrLM for NLG tasks (Yang et al., 2020; Zhu et al., 2020; Chen et al., 2020). Expending the previous studies, in this paper, we propose an improved method that exploits the checkpoint of the PrLM into the Transformer models (Vaswani et al., 2017).

The existing methods for leveraging PrLM in NLG tasks can be roughly classified into two categories: *Reusing the PrLM as a starting point* and *Integrating the intermediate output of the PrLM*. The former, the widely used in various NLP tasks, denotes to initialize the part of Transformer from the PrLM for generation tasks (Clinchant et al., 2019; Edunov et al., 2019; Rothe et al., 2020) or replace the input embedding with the PrLM. The latter is an approach that first extracts the contextualized representation from a LM for an input sentence and fuses it into a neural model (Yang et al., 2020; Zhu et al., 2020; Chen et al., 2020). As our preliminary experiment shows, we expand this approach and explore in many ways towards better performance. In both of the preceding approaches, whether to freeze or fine-tun the parameter of PrLM is also an important issue. For the former (*Reusing the PrLM*), several works demonstrated that freezing the PrLM at training time led to a significant performance drop. Meanwhile, for the latter approach (*Integrating the PrLM*), prior studies adopted the whole or half-freezing instead of fine-tuning the parameters of the PrLMs. Yang et al. (2020) suggested that the reason why fine-tuning PrLM in neural machine translation (NMT) does not work as well as in other NLP tasks is due to the availability of large training data and the high capacity of baseline NMT models (i.e., Transformer), where excessive fine-tuning leads to the *catastrophic forgetting* phenomenon (Goodfellow et al., 2015). Also, Zhu et al. (2020) shows that freezing the BERT in NMT is better than fine-tuning with a large gap. This is in line with our experimental results. Thus in this empirical study, we freeze the parameters of the PrLMs in our experiments.

This paper focuses on finding an effective

---

[*] Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100) and Key Projects of National Natural Science Foundation of China (U1836222 and 61733011). This work was supported by Huawei Noah's Ark Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2686–2694
August 1–6, 2021. ©2021 Association for Computational Linguistics

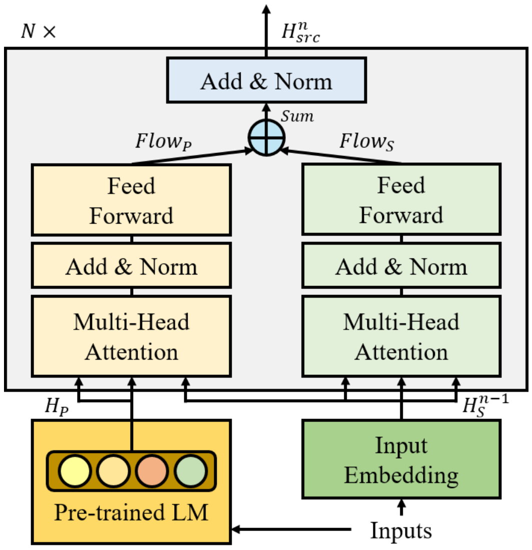

Figure 1: The architecture of our proposed model.

method integrating the intermediate output into the Transformer model to improve the generation quality, unlike the widely used methods such as initializing a part of Transformer and replacing the input embedding with the PrLM. To this end, similar to Zhu et al. (2020), we insert PrLM-dedicated modules that take the intermediate output from the PrLM and make the extra flow for PrLM in Transformer layers. This allows us to integrate PrLM into the Transformer without considering the PrLM's configuration such as modeling, dimension and vocabulary. Based on extensive empirical experiments, we finally adopt an improved method that uses the **Second-to-last (i.g., penultimate) hidden state** of the PrLM as the contextualized representation and proceeds, in only **Source-side (Encoder)**, **Summation** of the source input flow and the PrLM flow generated through the **PrLM-dedicated modules**. In our experiments, we use the publicly released checkpoints of 6 types of PrLMs: BERT, DistilBERT, ELECTRA, GPT2, Multilingual BERT, and XLM-RoBERTa. We release an implementation of our improved method for Korean language generation tasks [1].

2 Model

We propose a modified Transformer-encoder that effectively integrates publicly available checkpoints of PrLMs. Figure 1 shows the architecture of our proposed model. We add an extra flow for PrLM through additional PrLM-dedicated modules including `PrLMAttn`, `Add&Norm`, and `FFN`. Specific mathematical formulations are left at Appendix B. Given an input sequence $x_s = \{x_1, ..., x_N\}$, there is a PrLM-input sequence $x_p = \{x^p_1, ...x^p_M\}$ of length M splited by the PrLM-dedicated tokenizer. The PrLM-input sequence is fed to the PrLM for generating the PrLM representation $H_P = PrLM(x_p)$. Based on preliminary experiment, we adopt the second-to-last hidden state of the PrLM outputs as the contextualized representation. In our proposed encoder, the PrLM representation H_P is merged with the source flow to generate the output H^n_S of n^{th} encoder layer:

$$H^n_S = (Flow_P + Flow_S) + Attn_S, \tag{1}$$

$$Flow_P = \texttt{FNN}(Attn_P + H^{n-1}_S), \tag{2}$$

$$Attn_P = \texttt{PrLMAttn}(H^{n-1}_S, H_P, H_P), \tag{3}$$

$$Flow_S = \texttt{FNN}(Attn_S + H^{n-1}_S), \tag{4}$$

$$Attn_S = \texttt{Attn}(H^{n-1}_S, H^{n-1}_S, H^{n-1}_S), \tag{5}$$

where `PrLMAttn` is the PrLM-dedicated attention module that takes the previous hidden state H^{n-1}_S as a query and the PrLM representation H_P as a key and a value and `Attn` is the original one. We adopt the summation strategy for merging the two different flows, and it gains better results than previous works such as gate network (Yang et al., 2020) and dropnet (Zhu et al., 2020).

3 Experiments

To demonstrate the effectiveness of the proposed method, we perform extensive experiments on two NMT and abstractive text summarization tasks. For translation, we use BLEU (Papineni et al., 2002) for the evaluation of translation quality, and for text summarization, we report unigram and bigram overlap (ROUGE-1 and ROUGE2) to assess informativeness, and the longest common subsequence (ROUGE-L) to assess fluency with ROUGE scores (Lin, 2004). All the model training is on a single NVIDIA Tesla V100 GPU (16130MiB, Google Colab).

3.1 Datasets and Experimental Setting

We evaluate our approach on language generation tasks such as translation and text summarization. For translation tasks, we use two machine translation datasets: AIHub Ko→En [2] (containing 1.6M

---

[1] https://github.com/tmtmaj/Exploiting-PrLM-for-NLG-tasks

[2] http://www.aihub.or.kr/

Systems	Ko→Ch	Ko→En
Transformer	30.35 (-)	41.19 (-)
(Zhu et al., 2020)	31.33 (+0.9)	41.97 (+0.7)
(Clinchant et al., 2019)	31.75 (+1.4)	41.55 (+0.4)
Korean-specific PrLM		
+KoBERT	31.20 (+0.8)	42.17 (+0.9)
+HanBERT	31.39 (+0.9)	42.03 (+0.8)
+DistilKoBERT	30.94 (+0.5)	41.91 (+0.7)
+ELEC. small	31.51 (+1.1)	42.20 (+1.0)
+ELEC. base	**32.17 (+1.8)**	**42.59 (+1.4)**
+KoGPT2	30.38 (+0.0)	41.74 (+0.5)
Multi-language PrLM		
+BERT cased	30.57 (+0.2)	41.17 (-0.0)
+BERT uncased	30.78 (+0.4)	41.22 (+0.0)
+RoBERTa base	31.09 (+0.7)	41.85 (+0.6)
+RoBERTa large	31.64 (+1.2)	42.01 (+0.8)

Table 1: Experimental results on translation tasks. Both Clinchant et al. (2019) and Zhu et al. (2020) use ELECTRA base.

training, 3K development and 3K test sentence pairs) and Ko→Ch [3] (317K, 3K, 3K pairs). For text summarization, we use a news document summarization dataset (40K, 1K, 1K pairs) from two institutes [4]. In our experiment, we adopt a `base` Transformer (Vaswani et al., 2017) as the baseline model and use 6 different PrLMs: BERT, DistilBERT, ELECTRA, GPT2, Multi-lingual BERT, and XLM-RoBERTa. More details about the settings are included in Appendix A.

3.2 Experimental Results

Table 1 and 2 report the results of machine translation and abstractive text summarization task, respectively. For the experimental results, we made the following observation: (1) For all the tasks, our proposed methods outperform the strong baseline Transformer w.r.t BLEU (up to 1.8) and ROUGE (up to 1.8). (2) For PrLMs, **ELECTRA base** gains the most significant improvements. Note that **ELECTRA small** is smaller in size than other models, but it shows better performance. This means that factors other than the size of the PrLM have a greater influence on the generation quality. (3) Another observation is that in most cases, using Korean-specific PrLMs leads to better performance than using multi-language PrLMs.

Systems	R-1	R-2	R-L	△
Transformer	46.32	29.56	37.88	-
Oracle	57.17	44.00	44.46	-
Korean-specific PrLM				
+KoBERT	47.05	30.40	38.68	+0.8
+HanBERT	47.49	31.22	39.51	+1.5
+DistilKoBERT	46.64	29.86	38.43	+0.4
+ELEC. small	47.10	30.88	39.01	+1.1
+ELEC. base	**47.90**	**31.44**	**39.91**	**+1.8**
+KoGPT2	46.99	30.51	38.65	+0.8
Multi-language PrLM				
+BERT cased	46.44	29.61	38.11	+0.1
+BERT uncased	46.92	29.80	38.40	+0.4
+RoB. base	47.01	30.55	38.89	+0.9
+RoB. large	46.77	30.31	38.66	+0.6

Table 2: Experimental results on summarization task. △ denotes average improvements.

4 Explorations for leveraging PrLM

Our proposed method for leveraging PrLM is to use the **Second-to-last hidden state** of the PrLM as the contextualized representation and proceeds, in only **Source-side (Encoder)**, **Summation** of the source input flow and the PrLM flow after the **FNN**. In this subsection, the setting is the default, and we change only the target part of each experiment. We conducted the following four analyses on the Korean-Chinese and Korean-English datasets.

4.1 Which hidden state of the PrLM to extract?

We evaluated the impact on how to extract the contextualized representation from the PrLM. As shown in Table 3a, using the second-to-last (i.e., penultimate) hidden state of the PrLM performs the best. It has also been demonstrated in Yang et al. (2020). Moreover, as another attempt, we dynamically extracted the hidden state of each layer based on sentence embedding of n^{th} layer, which can be gained by averaging the PrLM layer (known as PrLM embeddings, Dyn. [Aver] in Table 3a) or using the output of the first token (the [CLS] token, Dyn. [CLS]). However, they did not get a big performance boost.

---

[3] http://www.donga.com/ and http://semanticweb.kaist.ac.kr

[4] https://corpus.korean.go.kr/ and https://dacon.io/competitions/official/235673/overview/

Systems	Ko→Ch	Ko→En
Dyn. [CLS]	31.09 (+0.7)	41.44 (+0.3)
Dyn. [Aver]	31.17 (+0.8)	41.46 (+0.3)
Last	31.34 (+1.0)	41.99 (+0.8)
Second-to-Last	**32.17 (+1.8)**	**42.59 (+1.4)**
Third-to-Last	32.03 (+1.7)	**42.53 (+1.3)**

(a) Methods extracting PrLM output)

Systems	Ko→Ch	Ko→En
Direct	30.24 (-0.1)	40.88 (-0.3)
Summation	**32.17 (+1.8)**	**42.59 (+1.4)**
Average	31.48 (+1.1)	41.94 (+0.8)
Gate Network	31.99 (+1.6)	42.31 (+1.1)
Dropnet	31.64 (+1.3)	41.99 (+0.8)

(b) Merging strategies for PrLM.

Systems	Ko→Ch	Ko→En
SelfAttn	31.78 (+1.4)	42.03 (+0.8)
1^{st} Add&Norm	31.68 (+1.3)	42.01 (+0.8)
FFN	**32.17 (+1.8)**	**42.59 (+1.4)**
2^{nd} Add&Norm	32.01 (+1.7)	42.44 (+1.3)

(c) Merging positions for PrLM.

Systems	Ko→Ch	Ko→En
Both-sides	31.94 (+1.6)	42.23 (+1.0)
Source-side	**32.17 (+1.8)**	**42.59 (+1.4)**
Target-side	31.29 (+0.9)	41.98 (+0.8)

(d) Adding positions for PrLM.

Table 3: The explorations for leveraging PrLM.

4.2 How to merge the PrLM representation with the source input flow?

We compared the impact of different merging strategies for the contextualized representation of PrLM. There are directly using the PrLM as the input embedding (Direct in Table 3b) and four merging strategies such as Summation, Average, using Gate Network (Yang et al., 2020), and using Dropnet (Zhu et al., 2020) As shown in Table 3b, the summation of the PrLM flow and the source input flow got the better improvement over others, so we adopted the Summation strategy in our experiments.

4.3 Where do the PrLM merge with the source input flow?

In the Table 3c, we analyzed where in the encoder layer of Transformer it would be better to merge the PrLM and source flows. There are four positions in a Transformer-encoder layer: after Attn, after 1^{st} Add&Norm, after FFN, and after 2^{nd} Add&Norm. It is interesting to find that merging after FFN can get the best performance. Another observation is that merging the contextualized representation of the PrLM before the Add&Norm (i.e., after Attn or FFN) works better.

4.4 Where do the PrLM flow add?

We evaluated where to add the PrLM: source-side, target-side, and both sides. As shown in the Table 3d report the results. Among them, adding the PrLM to only source-side (i.e., encoder) gained the best result. Intuitively, since contextual representation from the fixed PrLM contains univeral information, not information for generation tasks, combining it directly with the target context may adversely affect performance improvement.

Systems	Ko→Ch
Transformer	30.35 (-)
Multi-PrLMs	
+ELEC. base	32.17 (+1.8)
+ELEC. small, ELEC. base	32.11 (+1.7)
+KoBERT, ELEC. base	32.11 (+1.7)
+(Ko, Han)BERT, ELEC. base	32.19 (+1.8)

Table 4: Leveraging multi-LMs

5 More analyses

5.1 Leveraging Multi-PrLMs

We assumed that because PrLMs were trained with different datasets (size, domain) and diverse configurations, they would contain specific prior knowledge. So, we tried to integrate two or more PrLMs simultaneously (Multi-PrLMs) by adding more extra modules in each encoder layer. Contrary to our expectations, as shown in Table 4, using Multi-PrLMs cannot get a significant performance boost over using single-PrLM.

5.2 Fine-tuning v.s. Freezing

We compared the impact of fine-tuning and freezing the parameters of PrLM when using our method. Table 5 shows the results. We can see that freez-

Systems	Ko→Ch	Ko→En
Transformer	30.35 (-)	41.19 (-)
Freezing PrLM (Ours)		
+ELEC. small	31.51 (+1.1)	42.20 (+1.0)
+ELEC. base	32.17 (+1.8)	42.59 (+1.4)
Fine-tuning PrLM		
+ELEC. small	31.52 (+1.1)	41.98 (+0.8)
+ELEC. base	29.89 (-0.4)	38.92 (-2.3)

Table 5: Fine-tuning v.s. Freezing

Systems	sentences/s	tokens/s
Transformer	164.20 (-)	4.56k (-)
Our Systems		
+ELEC. small	155.59 (-5%)	4.35k (-5%)
+ELEC. base	139.81 (-17%)	3.92k (-16%)

Table 6: Inference Speed on Ko→En NMT task

ing the parameters of PrLM gains more significant improvement than fine-tuning. Another interesting observation is that using the ELECTRA base (112M parameters) when it is fine-tuning led to a significant performance drop, especially for relatively large corpus (Ko→En, 1.6M). It means that *catastrophic forgetting* issue is more pronounced in a resource-rich scenario and using large PrLM. Additionally, tuning separate learning rates (Yang et al., 2020) for the PrLM and the Transformer model may lead to better performance but we leave this to future work.

5.3 Inference Speed

We experimented with the inference speed of our method. Since our method has to obtain the intermediate output of PrLM for an input sentence, it takes more time in the inference process than the baseline model. The experimental results are shown in Table 6. Integrating PrLM into the Transformer model reduced the inference speed by about 5% (ELECTRA small, 14M parameters) to 17% (ELECTRA base, 112M parameters). However, considering the significant performance improvement and the ease of application to any language, it is acceptable of such extra cost.

6 Related Work

Previous studies relies on the structural compatibility of Transformer and PrLM. For example, Clinchant et al. (2019) presented initializing the encoder of Transformer from BERT (fine-tuned or fixed) and observed that freezing the PrLM causes a considerable performance drop. Conneau and Lample (2019) verified that initialization methods with CLM or MLM trained on multi-lingual corpora and showed such initialization are useful on MT. Rothe et al. (2020) used the publicly available PrLM checkpoints to initialize Transformer. While the initialization method is useful to some extent, there is a prerequisite for matching vocabulary and model size/hyper-parameters to them of PrLM.

Zhu et al. (2020) proposed a new method that extracts the last hidden state of BERT for an input sentence and fuses it into the encoder and decoder of the Transformer through an extra attention module, and evaluated the effectiveness of their method on supervised, semi-supervised and unsupervised NMT. Yang et al. (2020) introduced a concerted training framework with three techniques for fusing PrLM and NMT model. Although they also extract the hidden state of PrLM and integrate it into NMT model, the NMT model must follow the PrLM model's configurations such as word segmentation rule and vocabulary.

Our work is related to both Zhu et al. (2020) and Yang et al. (2020) in the sense that we all aim to extract the intermediate output of the PrLM and integrate it into a neural model for better generation quality. As an extension of Zhu et al. (2020), we propose an upgraded method adopted through extensive empirical experiments. Our work differs from Yang et al. (2020) in that we use the publicly available checkpoints that have various configurations and fix the PrLM at training time.

7 Conclusion

While most of the previous works on PrLM address the integration of PrLMs with *fine-tuning*, we propose an alternative in which a modified Transformer-encoder takes the intermediate output from PrLM to exploit its prior knowledge effectively in a straightforward way. Our method does not have to consider the PrLM's configuration, such as its model size, model dimension, and vocabulary. Correspondingly, our approach and reported empirical settings can be smoothly applied to any languages using any checkpoints of PrLMs.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling knowledge learned in BERT for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7893–7905, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia.

Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. 2019. On the use of BERT for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 108–117, Hong Kong. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 7057–7067, Vancouver, BC, Canada.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2015. An empirical investigation of catastrophic forgetting in gradient-based neural networks.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 770–778, Las Vegas, NV, USA. IEEE Computer Society.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, Vancouver, BC, Canada.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, NeurIPS 2019*, volume abs/1910.01108.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Yong Yu, Weinan Zhang, and Lei Li. 2020. Towards making the most of BERT in neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence, 34(05), 9378-9385.*, pages 9378–9385.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia.

A Experimental Settings

A.1 Model Setting

In our experiment, we use Transformer (Vaswani et al., 2017) as the baseline model for NMT and abstractive text summarization tasks. Additionally, for NMT tasks, we compare our approach to the following baselines:

- (Zhu et al., 2020): A method that inserts the last-hidden state of fixed PrLM through PrLM-dedicated attention module in Transformer-encoder and decoder.
- (Clinchant et al., 2019) (Direct* in Table 1): A method that replaces the input embedding with the PrLM that is fine-tuned in training time.

They all use ELECTRA base as the PrLM.

For the Transformer model, we use a `base` Transformer configuration (Vaswani et al., 2017) with an embedding size of 512, 6 encoder and decoder layers, 8 attention heads, shared source and target embedding, the standard `relu` activation function, and sinusoidal positional embedding. We train with a batch size of 3500 tokens and optimize the model parameters using Adam optimizer with a learning rate 7e-4 β_1= 0.9 and $\beta_2 = 0.98$, learning rate warm-up over the first 4000 steps. Additionally, we apply label smoothing with a factor of 0.1. We average over the last 5 checkpoints and run inference with a beam size of 5. All models are trained for 50 epochs using the Torch-based toolkit, Fairseq(-py) (Ott et al., 2019). For the text summarization task, we reduce the number of encoder and decoder layers to 4 and use *Trigram Blocking* (Paulus et al., 2018) to reduce redundancy during inference time. Other settings are the same as above.

For all datasets, we first tokenize sentences using language-specific tokenizer such as KoNLPy[5] for Korean, jieba[6] for Chinese, and Moses (Koehn et al., 2007) for English and then apply Byte-Pair Encoding (Sennrich et al., 2016) to the tokenized sentences with 32K merge-operations. Besides, most of PrLMs have a limit for input sequence length (e.g., 512), so we cut out the middle of some long text for text summarization dataset as proposed in Sun et al. (2019).

A.2 Pre-trained Language Model Setting

In our experiments, we use 6 types of different PrLMs including BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), ELECTRA (Clark et al., 2020), GPT2 (Radford et al., 2018), multi-lingual BERT, and XLM-RoBERTa (Conneau and Lample, 2019; Liu et al., 2019). Specifically, we use 10 different pre-trained checkpoints depending on the model size, training data set, and training level:

1. KoBERT: a BERT with 768-hidden, 12-layer, 12-heads, 8002-vocab, Korean dataset (4GB), 92M parameters; `https://github.com/SKTBrain/KoBERT.git`.
2. HanBERT: a BERT with 768-hidden, 12-layer, 12-heads, 54000-vocab, Korean dataset (70GB), 127M parameters; `https://github.com/tbai2019/HanBert-54k-N.git`.
3. DistilKoBERT: a DistilBERT with 768-hidden, 3-layer, 12-heads, 8002-vocab, Korean dataset (10GB), 28M parameters; `https://huggingface.co/monologg/distilkobert`.
4. ELECTRA small: a ELECTRA with 256-hidden, 12-layer, 4-heads, 35000-vocab, Korean dataset (34GB), 14M parameters; `https://huggingface.co/monologg/koelectra-small-v3-discriminator`.
5. ELECTRA base: a ELECTRA with 768-hidden, 12-layer, 12-heads, 35000-vocab, Korean dataset (34GB), 112M parameters; `https://huggingface.co/monologg/koelectra-base-v3-discriminator`.
6. KoGPT2: a GPT2 with 768-hidden, 12-layer, 12-heads, 50000-vocab, Korean dataset (20GB), 125M parameters; `https://github.com/SKT-AI/KoGPT2.git`.
7. Multi-lingual BERT cased: a BERT with 768-hidden, 12-layer, 12-heads, 119547-vocab, 104 languages, 177M parameters; `https://huggingface.co/bert-base-multilingual-cased`.
8. Multi-lingual BERT uncased: a BERT with 768-hidden, 12-layer, 12-heads, 105879-vocab, 102 languages, 167M parameters; `https://huggingface.co/bert-base-multilingual-uncased`.
9. XLM RoBERTa base: a BERT with 768-hidden, 12-layer, 12-heads, 250002-vocab, 100 languages (2.5TB), 277M parameters; `https://huggingface.co/xlm-roberta-base`.

---

[5] https://konlpy.org/en/latest/

[6] https://github.com/fxsjy/jieba

10. XLM RoBERTa large: a BERT with 1024-hidden, 24-layer, 16-heads, 250002-vocab, 100 languages (2.5TB), 561M parameters; https://huggingface.co/xlm-roberta-large.

B Details of the Notations

Let `Attn` denote a multi-head attention module, which takes three matrices containing a query matrix Q, a key matrix K, and a value matrix V and product an output matrix as follows:

$$\texttt{Attn}(Q, K, V) = concat(head_1, ..., head_i)W^o, \tag{6}$$

$$head_i = attn(Q_i, K_i, V_i), \tag{7}$$

$$attn(q, k, v) = softmax(\frac{qW^q kW^k}{\sqrt{d_{model}}})vW^v, \tag{8}$$

where $concat$ denotes a concatenation operation, $softmax$ denotes a softmax function, d_{model} is the dimension of the model, and W^o, W^q, W^k, W^v are parameter matrices. `FFN` consists of two fully-connected layers with a $relu$ activation in between.

$$\texttt{FFN} = max(0, xW^1 + b^1)W^2 + b^2, \tag{9}$$

where $\max(0, x)$ is $relu$ activation function, and W^1, b^1, W^2, b^2 are parameter matrices. Finally, `Attn` and `FFN` are connected with `Add&Norm`, which denotes a combination module containing a residual connection (He et al., 2016) and a layer normalization (Ba et al., 2016).

Making Flexible Use of Sub-tasks: A Multiplex Interaction Network for Unified Aspect-based Sentiment Analysis

Guoxin Yu[♠♡], Xiang Ao[♠♡*], Ling Luo[♠♡], Min Yang[♣], Xiaofei Sun[◇], Jiwei Li[◇], Qing He[♠♡]

[♠]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China.
[♡]University of Chinese Academy of Sciences, Beijing 100049, China.
[♣]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[◇]Shannon.AI, China.
`{yuguoxin20g,aoxiang,luoling18s,heqing}@ict.ac.cn`
`min.yang@siat.ac.cn`
`{xiaofei_sun,jiwei_li}@shannonai.com`

Abstract

Aspect Term Extraction (ATE), Opinion Term Extraction (OTE) and Aspect Sentiment Classification (ASC) are the essential building blocks of Aspect-based Sentiment Analysis (ABSA). They are typically treated as separate tasks and are individually studied by previous work. Recent studies intend to incorporate multiple sub-tasks into a unified framework, but suffer from the following major disadvantages: (1) ABSA models are extremely fragile when some sub-tasks are absent; (2) the interactive relations among sub-tasks are not adequate. To this end, we propose a multi-task learning approach named MIN (Multiplex Interaction Network) to make flexible use of sub-tasks for a unified ABSA. We divide the sub-tasks of ABSA into extractive sub-tasks and classification sub-tasks, and optimize these sub-tasks in a unified manner with multiplex interaction mechanisms. Specifically, we devise a pairwise attention to exploit bidirectional interactions between any arbitrary pair of extractive sub-tasks and a consistency-weighting to perform unidirectional interaction from an extractive sub-task to a classification sub-task. Since the proposed interaction mechanisms are task-agnostic, our model can also work well when some specific sub-tasks are absent. Extensive experiments on two widely used benchmarks with different numbers of sub-tasks demonstrate the superiority of the proposed model.

1 Introduction

Aspect-based sentiment analysis (ABSA), a fine-grained task of text sentiment analysis (Liu, 2012), aims at summarizing opinions in user comments towards different targets (also known as *aspects*). ABSA generally consists of three specific sub-tasks, namely, aspect term extraction (ATE), opinion term extraction (OTE) and aspect sentiment classification (ASC). For example, consider the sentence "*The price is reasonable although the service is inferior.*", ATE aims to extract a set of aspect terms from the sentence, i.e. {*price, service*}, OTE extracts the opinion words, i.e. {*reasonable, inferior*} , and ASC predicts sentiment polarity for each aspect that is *positive* over the first aspect *price* and *negative* for the second aspect *service*, respectively.

Prevailing solutions of ABSA treated ATE (Liu et al., 2015; Li and Lam, 2017; Angelidis and Lapata, 2018; Liao et al., 2019; Luo et al., 2019b; Ma et al., 2019), OTE (Wang et al., 2017; Wang and Pan, 2019) and ASC (Wang et al., 2016b; Chen et al., 2017; He et al., 2018; Li et al., 2018b; Du et al., 2019; Xu et al., 2021) as separate tasks and were individually studied for decades. These separate tasks need to be integrated into a pipeline for practical use (Hu et al., 2019; Phan and Ogunbona, 2020). The key problem with pipeline approaches is that errors can accumulate and that the pipeline model fails to fully exploit the interactive relations among different sub-tasks (He et al., 2019). Some recent efforts have been proposed to remedy these issues by using joint learning to enhance the interactions among sub-tasks (Wang et al., 2018; Li et al., 2019; He et al., 2019; Luo et al., 2019a; Chen and Qian, 2020; Peng et al., 2020) and achieved better performance than pipeline solutions. To name some, (Li et al., 2019) incorporated ATE and ASC and formulated the problem as a single sequence

---

[*] Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2695–2705
August 1–6, 2021.

Sentence	Food	is	pretty	good	but	the	orange	juice	is	horrific	.
ATE	B	O	O	O	O	O	B	I	O	O	O
OTE	O	O	O	B	O	O	O	O	O	B	O
ASC	POS	-	-	-	-	-	NEG	NEG	-	-	-

Table 1: A training instance sentence with gold ATE, OTE and ASC labels.

labeling task with a collapsed tagging scheme[1]. (Luo et al., 2019a) treated ATE and ASC as two sequence labeling problems to mitigate the confused representation brought by the collapsed tags. More recently, (He et al., 2019; Chen and Qian, 2020; Peng et al., 2020) proposed to utilize OTE as an auxiliary task and further facilitated performance.

Despite their success, existing solutions severely suffer from the following disadvantages: firstly, the approaches fusing ATE and ASC cannot benefit from OTE even though there exists opinion term annotations; secondly, the joint learning model might not work when any of the sub-tasks is absent; thirdly, the interactive relations among sub-tasks are not appropriately explored. For instance, (Luo et al., 2019a; Li et al., 2019) would fail to exploit OTE even if these exists labeled data for training. (Chen and Qian, 2020; Peng et al., 2020) might become fragile when the sub-task of OTE or ATE is absent, due to their special designed complex interactions among the sub-tasks of ATE, OTE and ASC. In a nutshell, the key reason for these disadvantages is that existing studies only consider specific interactive relations among a fixed number of sub-tasks, but are inadequate to use the available sub-tasks flexibly.

In this paper, we propose a simple yet effective neural method named MIN (Multiplex Interaction Network) that can make flexible use of sub-tasks for unified ABSA task. The flexibility comes in two folds: First, the number of sub-tasks is not restricted and any related sub-task can be integrated into the unified framework. Second, the interactive relations among distinct sub-tasks are flexibly modeled in an explicit manner.

To tackle the major challenge of how to manage ABSA with arbitrary number of sub-tasks and allow sub-tasks to share interactive information in a unified learning manner, we divide the sub-tasks of ABSA into two categories, namely extractive sub-tasks (e.g. ATE and OTE) and classification sub-tasks (e.g. ASC), and formulate each sub-task as a sequence labeling problem. MIN is built upon a multi-task learning framework of the shared-private scheme (Collobert and Weston, 2008). Sub-tasks first jointly extract the low-level shared features using multi-layer CNN encoder/pre-trained models. Then the private features of each extractive sub-task are independently learned by a multi-layer CNN network with a pairwise attention mechanism. Such *pair-attention* can capture bidirectional interactions between any two extractive sub-tasks accordingly. Moreover, we devise a *consistency-weighting* mechanism to exploit unidirectional interactions from extractive sub-tasks to classification sub-tasks and assist in private feature extractions in classification sub-tasks. The proposed interaction mechanisms (including pair-attention and consistency-weighting) give the model the flexibility to handle different ABSA situations with different number of sub-tasks of different types. In addition, MIN adopts an information feedback mechanism that first aggregates information from all available tasks and then propagates useful information back to individual tasks, allowing different sub-tasks to positively influence each other. Experiments on two widely used ABSA benchmarks using different numbers of sub-tasks demonstrate the proposed MIN outperforms both pipeline and joint learning SOTA baselines on various settings.

2 Related Work

Separate learning mainly focuses on one of the sub-tasks in ABSA, e.g. aspect term extraction (ATE), opinion term extraction (OTE) or aspect sentiment classification (ASC). For the ATE task, previous methods can be divided into unsupervised and supervised models, respectively. In unsupervised methods, (Liu et al., 2015) utilized rules about grammar dependency relations between opinion words and aspects for aspect terms extraction. (Luo et al., 2019b) presented a neural framework that leverages sememes to enhance lexical semantics for long-tailed aspect extraction. (Liao et al., 2019) utilized the capability of coupling global and local representation to discover aspect terms. For supervised methods (Wang et al., 2016b; Li

---

[1] {B, I, E}-{POS, NEU, NEG} indicates the beginning, inside and end of an aspect-term with positive, neutral or negative sentiment, respectively. O denotes NULL sentiment.

and Lam, 2017; Angelidis and Lapata, 2018; Ma et al., 2019), they were usually formulated as a sequence tagging problem, and various neutral networks with attention mechanisms were proposed to solve the task. For ASC, (Liu and Zhang, 2017; Cheng et al., 2017; He et al., 2018; Tang et al., 2019; Liang et al., 2019; Lei et al., 2019) attempted to exploit contextual and positional proximity of aspect terms for prediction by attentional neural networks. And (Tian et al., 2020) proposed to learn a unified sentiment representation for different sentiment analysis tasks. Recently, capsule network (Du et al., 2019; Chen and Qian, 2019), and graph convolution networks (Zhang et al., 2019) were also utilized in ASC and achieved SOTA performance. These separate learning approaches may have disadvantages in practical applications as they need to be pipelined and the interactions between different sub-tasks are totally neglected.

Joint learning strives to combine sub-tasks of ABSA into a unified learning process. For example, some studies proposed to handle ATE and ASC in a pipeline or an integrated model. The pipeline models (Hu et al., 2019; Phan and Ogunbona, 2020) are extract-then-classify processes and were proposed to solve the two tasks successively. However, they can still derive error accumulations. For integrated models, (Wang et al., 2018; Li et al., 2019) solved ATE and ASC by collapsed tagging that is a unified tagging scheme to link the two tasks. (Luo et al., 2019a) considered the relationship between the two tasks and attempted to investigate useful information from one task to another. Some works (Wang et al., 2017; Dai and Song, 2019; Chen et al., 2020; Zhao et al., 2020) integrated ATE and OTE in the same framework to illustrate these two tasks can benefit from each other. Then emerging methods (He et al., 2019; Chen and Qian, 2020; Peng et al., 2020) proposed to inject OTE as an auxiliary task to further improve the performance of ABSA. However, the number of sub-tasks and interactions among them in existing integrated methods are fixed, which can be restricted when sub-tasks vary in practice.

3 The Multiplex Interaction Network

3.1 Task Definition

All the sub-tasks related to ABSA are categorized into extractive and classification sub-tasks, respectively, in our framework. The extractive sub-tasks extract meaningful subsequences of sentences, such as aspect terms and opinion terms, etc. The classification sub-tasks classify parts of sentences into different classes, e.g. sentiment polarities, genres and etc. We formulate all sub-tasks as sequence labeling problems in our joint learning framework. Given a sequence of tokens $\mathrm{X} = \{x_1, x_2, ..., x_n\}$,

- An extractive sub-task $\mathcal{T}$ is to predict a sequence tag $\mathrm{Y}^{\mathcal{T}} = \{y_1^{\mathcal{T}}, y_2^{\mathcal{T}}, ..., y_n^{\mathcal{T}}\}$ for the sentence $\mathrm{X} = \{x_1, x_2, ..., x_n\}$, where $y_i^{\mathcal{T}} \in \{\mathrm{B, I, O}\}$. Specifically, $\{\mathrm{B, I, O}\}$ denotes beginning of, inside of and out of the extracted targets. For example, the first and the second rows of Table 1 demonstrate the sequence tags of aspect term extraction (ATE) and opinion term extraction (OTE) respectively. In the example, “food” and “orange juice” are annotated as aspect terms, and “good” and “horrific” are labeled as opinion terms.
- A classification sub-task $\mathcal{C}$ is to predict a sequence tag $\mathrm{Y}^{\mathcal{C}} = \{y_1^{\mathcal{C}}, y_2^{\mathcal{C}}, ..., y_n^{\mathcal{C}}\}$ for the sentence $\mathrm{X} = \{x_1, x_2, ..., x_n\}$, where $y_i^{\mathcal{C}}$ is one of the class labels. $y_i^{\mathcal{C}}$ will be marked as “-” if the token x_i does not belong to any category. For example, the last row of Table 1 demonstrates the labels of the aspect sentiment classification (ASC) task for the example sentence. In this example, the “food” is labeled as “POS”, and “orange” as well as “juice” is labeled as “NEG”, according to the ground truth sentiment labels of the two aspect terms. Other tokens are labeled as “-” as they do not have golden labels in this task.

3.2 Model Overview

For the convenience of explanation, we take two extractive sub-tasks, namely ATE and OTE, and a classification sub-task ASC to describe our MIN approach hereafter. The overall architecture of MIN is shown in Figure 1. MIN consists of a shared encode layer, several features extraction modules for different sub-tasks, and an information feedback mechanism.

Private features extraction modules accept the same representation from a shared encode layer then extract their private task-oriented features by specific multi-layer CNN algorithms. In order to enhance flexibility, MIN conducts task interactions for two facets. Firstly, we exploit the bidirectional interaction between two different extractive sub-tasks by a proposed *pair-attention*. Using pair-

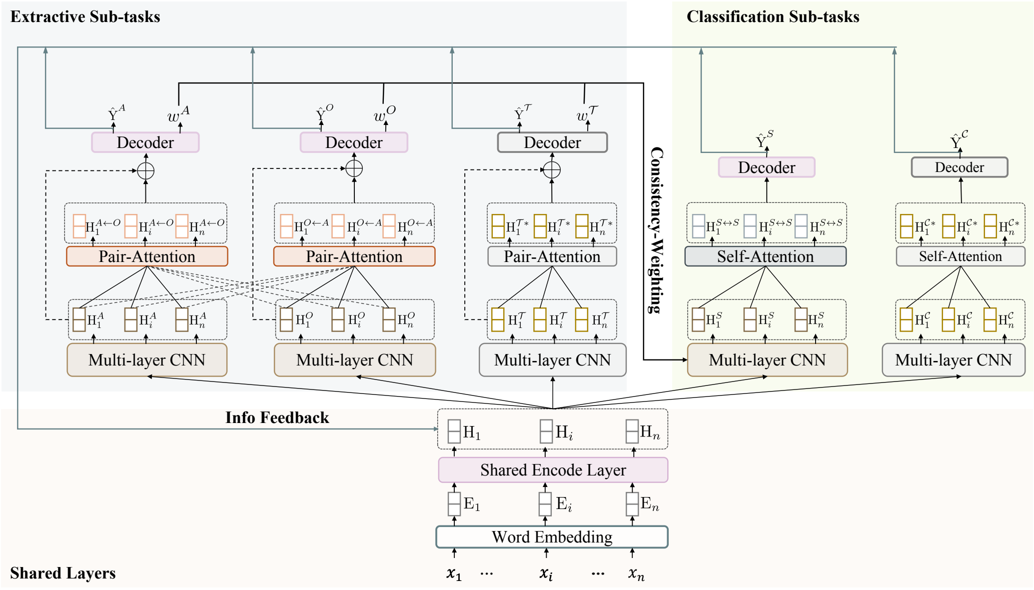

Figure 1: Model Overview of MIN. ATE, OTE and ASC are shown with corresponding interactions. A possible extractive sub-task and a possible classification sub-task are marked in grey colors and some details of them are omitted for easy viewing.

attention, extractive sub-tasks can exchange helpful clues with each other during features extraction. Secondly, we use a *consistency-weighting* mechanism to perform the unidirectional interaction from an extractive sub-task to a classification sub-task. In this way, features with highlight information of extractive sub-tasks will be passed to other classification sub-tasks and assist in their features extractions. Then every sub-task predicts the corresponding sequence labels, i.e., $\hat{\mathrm{Y}}^A$, $\hat{\mathrm{Y}}^O$ and $\hat{\mathrm{Y}}^S$, by its decode layer. The model also adopts an information feedback mechanism that concatenates representations of all sub-tasks to fine-tune the shared representations. In the following, we first describe the MIN model in more detail and then illustrate the learning process.

3.3 Shared Representation Generation

For a sequence of tokens $\{x_1, x_2, ..., x_n\}$, we map the word sequence with either pre-trained double word embeddings (Xu et al., 2018) or pre-trained Bert model to generate a sequence of word vectors $\mathrm{E} = \{\mathrm{E}_1, ..., \mathrm{E}_i, ..., \mathrm{E}_n\} \in \mathbb{R}^{d_e \times n}$, where n denotes the sentence length and d_e denotes the dimension of word vectors. Then we simply feed E into a multi-layer CNN encoder to generate shared features $\mathrm{H} = \{\mathrm{H}_1, ..., \mathrm{H}_i, ..., \mathrm{H}_n\} \in \mathbb{R}^{d_h \times n}$, where d_h denotes the dimension of hidden vectors.

3.4 Features Extraction for Extractive Sub-tasks

3.4.1 Multi-layer CNN

For the extractive sub-tasks, we use a multi-layer CNN structure proposed by (Xu et al., 2018) to learn private features of each task separately. Specifically, there are many 1D-convolution filters in each CNN layer, and each filter has a fixed kernel size of $k = 2c + 1$. As a result, each filter performs convolution operation on a window of k word representations, and compute the representation for the i-th word along with $2c$ nearby words in its context. We can extract private features H^A of ATE and H^O of OTE by the above multi-layer CNN algorithm,

$$\begin{aligned} \mathrm{H}^A &= \mathrm{MC}(\mathrm{H}), \mathrm{H}^A \in \mathbb{R}^{d_p \times n}, \\ \mathrm{H}^O &= \mathrm{MC}(\mathrm{H}), \mathrm{H}^O \in \mathbb{R}^{d_p \times n}, \end{aligned} \tag{1}$$

where MC indicates the multi-layer CNN algorithm, d_p denotes the dimension of the private features for extractive sub-tasks.

3.4.2 Pair-attention

In order to exploit potential interactions, we develop a pair-attention to exchange helpful clues between ATE and OTE based on their semantic correlations. For ATE, the pair-attention matrix

$\mathrm{A}^{A \leftarrow O} \in \mathbb{R}^{n \times n}$ is computed as follows:

$$\begin{aligned} score_{ij}^{(i \neq j)} &= (\mathrm{H}_i^A)^\top \mathrm{H}_j^O, \\ \mathrm{A}_{ij}^{A \leftarrow O} &= \frac{\exp(score_{ij})}{\sum_{k=1}^{n} \exp(score_{ik})}. \end{aligned} \tag{2}$$

By applying a weighted sum operation of pair-attention to the private features H^O of OTE, we get an interactive feature matrix $\mathrm{H}^{A \leftarrow O}$ for ATE:

$$\mathrm{H}_i^{A \leftarrow O} = \sum_{j=1}^{n} (\mathrm{A}_{ij}^{A \leftarrow O} \cdot \mathrm{H}_j^O). \tag{3}$$

Similarly, according to Eq. 2, we exchange H^A and H^O to compute pair-attention matrix $\mathrm{A}^{O \leftarrow A} \in \mathbb{R}^{n \times n}$ then use $\mathrm{A}^{O \leftarrow A}$ and H^A to get interactive feature matrix $\mathrm{H}^{O \leftarrow A}$ for OTE by Eq. 3. In this way, the model exploits bidirectional interactions between the two sub-tasks. After pair-attention, we concatenate the private and the interactive features of the same sub-task, and then feed them to a fully-connected layer to predict the sequence tags as follows:

$$\begin{aligned} \hat{\mathrm{Y}}^A &= \mathrm{softmax}(\mathrm{W}^A(\mathrm{H}^A \oplus \mathrm{H}^{A \leftarrow O})), \\ \hat{\mathrm{Y}}^O &= \mathrm{softmax}(\mathrm{W}^O(\mathrm{H}^O \oplus \mathrm{H}^{O \leftarrow A})). \end{aligned} \tag{4}$$

Note that for other extractive sub-tasks, we can also compute the pair-attention with related extractive sub-tasks in a similar manner.

3.5 Features Extraction for Classification Sub-tasks

Recall that a classification sub-task may benefit from extractive sub-tasks. To name some, knowing the location of aspect terms will enhance the matching degree of aspect terms with corresponding sentiment polarities. For example, the second row and the last row of Table 1 have consistent labels[2] since we expect to perform sentiment prediction for aspect terms than other words. Similarly, the sentiment polarities of aspect terms are mainly influenced by surrounding opinion words. For example, we can find "NEG" labels near to the opinion word "horrific" in the example sentence shown in Table 1.

3.5.1 Consistency-weighting

Inspired by the above observations, we devise a consistency-weighting mechanism to learn private features of ASC task.

First, we compute a set of weights to pass information with salient aspect and opinion terms to ASC. Specifically, the weights of a token i w.r.t. aspect and opinion terms are computed as follows:

$$\begin{aligned} w_i^A &= \sum_{j=1}^{k} (\alpha - \beta \cdot |a_j - i|), \\ w_i^O &= \sum_{j=1}^{k} (\alpha^{'} - \beta^{'} \cdot |o_j - i|) \end{aligned} \tag{5}$$

where k denotes the first k tokens most likely to be aspect or opinion terms, and a_j and o_j are the indexes of j-th aspect and opinion term in the sentence. α, $\alpha^{'}$, β and $\beta^{'}$ are pre-specified constants. We can simply multiply the weights w^A and w^O with interactive features of ATE and OTE to get a distilled representation of every token:

$$\begin{aligned} \mathrm{H}_i^{S \leftarrow A} &= w_i^A \cdot \mathrm{H}_i^A, \mathrm{H}^{S \leftarrow A} \in \mathbb{R}^{d_p \times n}, \\ \mathrm{H}_i^{S \leftarrow O} &= w_i^O \cdot \mathrm{H}_i^O, \mathrm{H}^{S \leftarrow A} \in \mathbb{R}^{d_p \times n}. \end{aligned} \tag{6}$$

Then the shared features are concatenated with such distilled representation, and the private features of ASC are extracted by the same multi-layer CNN algorithm as in the ATE and OTE sub-tasks.

$$\begin{aligned} &\mathrm{H}^S = \mathrm{MC}(\mathrm{H}^{S \leftarrow A} \oplus \mathrm{H}^{S \leftarrow O} \oplus \mathrm{H}), \\ &\mathrm{H}^S \in \mathbb{R}^{d_p \times n}. \end{aligned} \tag{7}$$

By consistency-weighting, the information of aspect and opinion terms are highlighted during the features extraction for the classification sub-task.

3.5.2 Self-Attention

We adopt a self-attention mechanism to learn long-distant dependent information in a sentence for the classification sub-task. Specifically, the self-attention matrix $\mathrm{A}^{S \leftrightarrow S}$ is computed as follows:

$$\begin{aligned} score_{ij}^{(i \neq j)} &= ((\mathrm{H}_i^S)^\top \mathrm{W}^{S \leftrightarrow S} \mathrm{H}_j^S) \cdot \frac{1}{|i - j|}, \\ \mathrm{A}_{ij}^{S \leftrightarrow S} &= \frac{\exp(score_{ij})}{\sum_{k=1}^{n} \exp(score_{ik})}, \mathrm{A}^{S \leftrightarrow S} \in \mathbb{R}^{n \times n}, \end{aligned} \tag{8}$$

where $\mathrm{H}_i^S \mathrm{W}^{S \leftrightarrow S} (\mathrm{H}_j^S)^\mathrm{T}$ and $\frac{1}{|i-j|}$ denote the semantic relevance and distance between H_i and H_j, respectively. After getting the outputs of the self-attention layer, we also exploit a fully-connected layer with softmax activation as a decoder:

$$\begin{aligned} \mathrm{H}_i^{S \leftrightarrow S} &= \sum_{j=1}^{n} (\mathrm{A}_{ij}^{S \leftrightarrow S} \cdot \mathrm{H}_j^S), \\ \hat{\mathrm{Y}}^S &= \mathrm{softmax}(\mathrm{W}^S \mathrm{H}^{S \leftrightarrow S}), \end{aligned} \tag{9}$$

where $\hat{\mathrm{Y}}^S$ is the predicted label sequence for ASC.

---

[2] "Food" is the beginning of an aspect term and it has a positive sentiment; while "pretty" is out of an aspect term and thus has no sentiment in ATE and ASC sub-tasks.

Dataset		Sentence	Opinion	Aspect		
				Pos	Neu	Neg
Res14	Train	3,044	3,484	2,164	807	637
	Test	800	1,008	728	196	196
Lap14	Train	3,048	2,504	994	870	464
	Test	800	674	341	128	169

Table 2: The statistics of datasets.

3.6 Information Feedback Mechanism

MIN exploits an information feedback mechanism to interactively fine-tune the shared feature representations. We denote t as the times of the information feedback. In each round, we utilize the hidden representation before decoder in every sub-task to update the shared features. Specifically, we concatenate the hidden representations with the shared feature matrix, and then use a fully-connected layer with softmax activation to squeeze the vectors to a fixed dimension.

$$\mathrm{H}_t = \mathrm{softmax}(\mathrm{W}(\mathrm{H}_{t-1}^{A \leftarrow O} \oplus \mathrm{H}_{t-1}^{O \leftarrow A} \oplus \mathrm{H}_{t-1}^{S \leftrightarrow S} \oplus \mathrm{H})), \quad (10)$$

where t denotes the t-th feedback update.

3.7 Training Procedure

The overall loss $\mathcal{L}$ consists of cross-entropy losses for all sub-tasks according to final predicted sequence labels.

$$\begin{aligned} \mathcal{L}^A &= -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_{ij}^A \cdot \log(\hat{y}_{ij}^A)), \\ \mathcal{L}^O &= -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_{ij}^O \cdot \log(\hat{y}_{ij}^O)), \\ \mathcal{L}^S &= -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_{ij}^S \cdot \log(\hat{y}_{ij}^S)), \end{aligned} \quad (11)$$

where N denotes the number of training instances, n_i denotes the number of tokens in the i-th instance. The overall loss is the weighted sum of the sub-tasks' losses:

$$\mathcal{L} = a \cdot \mathcal{L}^A + b \cdot \mathcal{L}^O + c \cdot \mathcal{L}^S, \quad (12)$$

where a, b, c are task coefficients, and we set all of them as 1. Following (He et al., 2019), during model training, we only compute ASC loss on tokens which are related to aspect terms, i.e., if a token is not aspect term, we will ignore predicted the sentiment on it. We minimize the $\mathcal{L}$ and determine a suitable t by grid search for information feedback mechanism during the experiment.

Model	Res14			Lap14		
	AE-F1	AS-F1	O-F1	AE-F1	AS-F1	O-F1
DECNN-TNet*	82.79	70.45	65.80	79.38	68.69	57.39
DECNN-TCaps*	82.79	71.77	66.84	79.38	69.61	57.71
PIPELINE-MIN	84.00	71.75	68.36	78.43	71.45	59.19
MNN*	**85.84**	67.93	–	79.91	58.30	–
E2E-ABSA*	83.92	68.38	66.60	77.34	68.24	55.88
DOER*	84.63	64.50	68.55	**80.21**	60.18	56.71
MIN	84.80	**73.91**	**69.57**	79.94	**71.57**	**60.39**

Table 3: Comparison results for double-task methods. We divided all methods into pipeline and unified models. The best scores are in bold face and the second best ones are underlined. "*" denotes the results are taken from their own papers or (Chen and Qian, 2020).

4 Experiments

4.1 Datasets

We adopt two widely used datasets from SemEval2014 Task 4 (Pontiki et al., 2014) in our experiment which contain reviews about restaurant and laptop, and the statistics are shown in Table 2. Note that original datasets have ground truth labels for aspect terms and corresponding sentiment polarities, while labels for opinion terms are annotated by (Wang et al., 2016a).

4.2 Compared Methods

We compare our method with recent ABSA methods. To demonstrate the flexibility of our method, we divide the baselines into two categories, namely double-task (ATE and ASC) and triple-task (ATE, OTE and ASC) methods, considering that existing methods are usually designed for fixed sub-tasks.

Double-task methods contain (1) pipeline methods: we take top-performing method **DECNN** (Xu et al., 2018) for ATE, **TNet** (Li et al., 2018a) and **TCaps** (Chen and Qian, 2019) for ASC to construct two pipeline models following (Chen and Qian, 2020). We conduct a pipeline setting of our proposed MIN denoted as **PIPELINE-MIN**, which trains ATE and ASC separately, for fair comparison. (2) unified methods: **MNN** (Wang et al., 2018) and **E2E-ABSA** (Li et al., 2019) jointly solve ATE and ASC by using collapsed tagging schemes, and **DOER** (Luo et al., 2019a) treats these two tasks as two sequence labeling problems and utilized a dual cross-shared RNN.

Triple-task methods aim to solve ATE, OTE and ASC simultaneously, including (1) pipeline methods: following (Chen and Qian, 2020), we combine **CMLA** (Xu et al., 2018) for ATE, **TNet** and **TCaps** for ASC to construct two pipeline baselines.

OTE is integrated into ATE. (2) unified methods: **IMN** (He et al., 2019) is an interactive multi-task model jointly trained on ATE and ASC where OTE is also integrated into ATE. While **RACL** (Chen and Qian, 2020) is trained on ATE, OTE and ASC in parallel that considers four relations among the three sub-tasks.

4.2.1 Settings

Following (He et al., 2019), we adopt double embedding in word embedding layer of **MIN**, where each word embedding is a concatenation of *general embedding* (Pennington et al., 2014) with 300 dimensions and *domain embedding* (Xu et al., 2018) with 100 dimensions. We set the hidden size $d_e = 400$, $d_h = 300$, $d_p = 300$, and the kernel size, number of shared CNN layers to 5 and 2, individually. And the numbers of information feedback in Res14 and Lap14 are set to 2 and 3 respectively. The layers of multi-layer CNN for ATE, OTE and ASC are set to $\{2, 2, 1\}$ and the kernel size is 5 in two dataset. Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-4 and a batch size of 8 are utilized for all datasets.

We also combine **MIN** with $\text{BERT}_{\text{Large}}$ to get **MIN-BERT** and $\{d_e, d_h, d_p\}$ is set to $\{1024, 300, 300\}$ with a learning rate of $1e-5$ by grid search. The layers of multi-layer CNN for ATE, OTE, ASC and the number of information feedback are set to $\{1, 1, 1, 2\}$ in the two datasets. The other parameters are the same as the **MIN** model with the double embedding. We meanwhile alternate IMN and RACL with $\text{BERT}_{\text{Large}}$ for further detailed comparison. We use the same metrics as (He et al., 2019), i.e., AE-F1, OE-F1, AS-F1 and O-F1, representing macro F1 scores for ATE, OTE, ASC and overall performance for complete ABSA. We select the model with the best O-F1 on the development set for test.

4.3 Experimental Results

4.3.1 Overall Performance

Table 3 and 4 demonstrate the performance of our method and the compared double-task (i.e., ATE and ASC) and triple-task methods (i.e., ATE, OTE and ASC), respectively. We have several consistent observations from the two tables. Firstly, most unified models perform better than pipeline models, which proves the advantages of exploiting interactions between sub-tasks. Secondly, our MIN can be flexibly applied to either double-task or triple-task scenario and can achieve better performance than all the compared methods on O-F1. We conjecture the possible reason is that the proposed generic interaction mechanism among different sub-task types is independent on any specific sub-task. Thus, our MIN can work well regardless of the number or the type of sub-tasks. Thirdly, we enhance the performance of some compared models with word embeddings provided by $\text{BERT}_{\text{Large}}$ (Devlin et al., 2018) and report the corresponding results in Table 4. We can observe that our MIN-BERT also shows superiority especially on O-F1.

Model	Res14				Lap14			
	AE-F1	OE-F1	AS-F1	O-F1	AE-F1	OE-F1	AS-F1	O-F1
CMLA-TNet*	81.91	83.84	69.69	64.49	77.49	76.06	68.30	55.94
CMLA-TCaps*	81.91	83.84	71.32	65.68	77.49	76.06	69.49	56.30
IMN*	84.01	85.64	71.90	68.32	78.46	78.14	69.92	57.66
RACL*	**85.37**	85.32	74.46	70.67	81.99	79.76	71.09	60.63
MIN	85.27	**86.85**	**76.39**	**70.92**	**82.24**	**80.56**	**72.60**	**61.35**
IMN-BERT	84.06	85.10	75.67	70.72	77.55	81.00	**75.56**	61.73
RACL-BERT	86.38	**87.18**	**81.61**	75.42	81.79	79.72	73.91	63.40
MIN-BERT	**87.91**	85.66	80.48	**76.02**	**83.22**	**81.80**	74.95	**64.83**

Table 4: Comparison results for triple-task methods. We divided all methods into pipeline, unified models and models based on BERT-large. "*", bolded and underlined scores have the same meanings as Table 3.

	Res14		Lap14	
ATE+OTE	AE-F1	OE-F1	AE-F1	OE-F1
IMN	84.83	86.32	78.31	77.58
RACL	85.47	86.48	**81.83**	78.19
MIN	**85.78**	**86.71**	81.74	**78.29**
OTE+ASC	OE-F1	AS-F1	OE-F1	AS-F1
IMN	NA	NA	NA	NA
RACL	81.27	63.16	79.32	65.45
MIN	**86.14**	**71.15**	**79.58**	**68.13**
ATE+ASC	AE-F1	AS-F1	AE-F1	AS-F1
IMN	84.78	70.46	79.22	69.65
RACL	**85.66**	70.78	79.76	68.87
MIN	84.80	**73.91**	**79.94**	**71.57**

Table 5: Comparison results for different combinations of ATE, OTE and ASC. The best scores are in bold. 'NA' denotes not available.

4.3.2 Flexibility

To further investigate the flexibility of MIN, we compare our MIN with variants of strong competitors in triple-task methods by varying the combination of sub-tasks. We remove one of the three sub-tasks and get three different combinations, namely "ATE+OTE", "ATE+ASC" and "OTE+ASC". Note that the double-task methods in table 3 cannot be easily adapted for this comparison due to their special design for specific sub-task combinations.

Table 5 exhibits the corresponding results on two datasets. We can see from the table that MIN

Model	Res14	Lap14
Full Model	70.92	61.35
w/o Pair-attention	69.82	59.37
w/o Consistency-weighting	69.46	60.84
w/o Info Feedback	69.07	59.17
w/o Self-Attention	67.63	57.03

Table 6: Results of ablation study on the model for "ATE+OTE+ASC". 'w/o' denotes without.

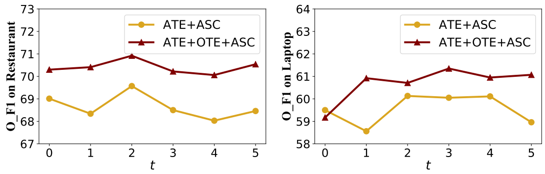

(a) Results on Res14. (b) Results on Lap14.

Figure 2: Effects of parameter t.

outperforms both IMN and RACL on most of the cases, and the two baselines cannot work well when the sub-task combination varies since both of them designed specialized interaction mechanisms for a fixed number of sub-tasks. For example, IMN adopts OTE as an auxiliary task for ATE and thus cannot provide results on "OTE+ASC". RACL, which designs four kinds of collaborative relations among ATE, OTE and ASC, degrades when arbitrary one sub-task is unavailable. However, our MIN separates the sub-tasks by different categories and leverages two category-level interaction mechanisms. It derives a trade-off that the interactions on remaining sub-tasks are still functional although one sub-task is removed. That is where the flexibility of our model comes from.

4.3.3 Ablation Test

In order to verify the effectiveness of different interactive mechanisms in our model, we conduct ablation tests on "ATE+OTE+ASC" unified ABSA. We remove different interaction mechanisms between sub-tasks in turn to observe the effectiveness of them. As the O-F1 results shown in Table 6, all interaction mechanisms are effective in MIN as expected. For example, on Res14, the scores of model without consistency-weighting and the model without self-attention drop largely. It is believed that sentiment polarities are mainly influenced by corresponding aspect and opinion words.

4.3.4 Case Study

Next, we illustrate two cases generated by IMN, RACL and our MIN for further investigation in Table 7. In Case (1), IMN extracts "choice" as an extra aspect without considering the relations between ATE and OTE. In addition, RACL incorrectly identifies "choice" and "craving" as aspect and opinion terms. We conjecture the possible reasons might be the "choice" is paired with "exquisite" as an aspect and the "craving" is recognized as a term expressing opinions by the specifically-designed interaction relations in RACL. However, our proposed pair-attention of MIN is more general, and correctly extracts all the aspect terms and opinion terms. Case (2) explores the effectiveness of our proposed consistency-weighting. Specifically, although IMN extracts the correct opinion word "too sweet", the final sentiment of "mole sauce" is still predicted wrong with a weak help of opinion terms. Meanwhile, although RACL exploits the relation between OTE and ASC, it recognizes the "sweet" incompletely, which might because RACL cannot update representation without the information feedback mechanism. As a result, "sweet" interferes sentiment prediction and leads to a wrong polarity for "mole sauce". In contrast, MIN correctly predicts sentiment polarities by using consistency-weighting based on complete aspect and opinion terms.

4.3.5 Times of Information Feedback

Finally, we investigate the sensitivity of the parameter t in the information feedback mechanism. Figure 2 (a) and (b) show the results of O-F1 on Res14 and Lap14 as t increases. We observe that MIN achieves better performance when $t = 2$ or $t = 3$. And there is no considerable improvement with a further increase of t, because redundant iterations might derive overfitting.

5 Conclusion

In this paper, we propose a novel neural model MIN to make flexible use of sub-tasks for the unified ABSA task. We introduce two sub-task agnostic interaction mechanisms, i.e., pair-attention and consistency-weighting, to exploit interactions among different sub-tasks and an information feedback mechanism to fine-tune the shared feature representation. Experiments with a different number of sub-tasks show the flexibility of MIN. The proposed framework can be potentially applied to similar tasks such as named entity recognition, semantic relation extraction, and etc.

Case	IMN		RACL		MIN	
	ATE,ASC	OTE	ATE,ASC	OTE	ATE,ASC	OTE
(1) If you ’re craving some [**serious**] [**indian food**]$_{\text{pos}}$ and desire a [**cozy**] [**ambiance**]$_{\text{pos}}$, this is quite and [**exquisite**] choice.	[indian food]$_{\text{pos}}$ [ambiance]$_{\text{pos}}$ [choice✗]$_{\text{pos}}$	[serious] [cozy] [exquisite]	[indian food]$_{\text{pos}}$ [ambiance]$_{\text{pos}}$ [choice✗]$_{\text{pos}}$	[craving✗] [serious] [cozy] [exquisite]	[indian food]$_{\text{pos}}$ [ambiance]$_{\text{pos}}$	[serious] [cozy] [exquisite]
(2) The [**fajita**]$_{\text{neg}}$ we tried was [**tasteless**] and [**burned**] and the [**mole sauce**]$_{\text{neg}}$ was way [**too sweet**] .	[fajita]$_{\text{neg}}$ [mole sauce]$_{\text{pos}}$✗	[tasteless] [burned] [too sweet]	[fajita]$_{\text{neg}}$ [mole sauce]$_{\text{neu}}$✗	[tasteless] [burned] [sweet✗]	[fajita]$_{\text{neg}}$ [mole sauce]$_{\text{neg}}$	[tasteless] [burned] [too sweet]

Table 7: Case study. The abbreviations pos, neu and neg in the table represent positive, neutral and negative. ✗ indicates incorrect prediction. Notice that the ground-truth aspect and opinion terms are marked in red and blue in the brackets, while the sentiment polarity labels for the aspect terms are demonstrated as corresponding subscripts.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104, the National Natural Science Foundation of China under Grant No. 92046003, 61976204, U1811461. Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS and Beijing Nova Program Z201100006820062. We would like to thank the anonymous reviewers for their valuable comments, and Yang Liu, Hai Jia, Mengda Huang for their insightful discussions.

References

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3675–3686, Brussels, Belgium. Association for Computational Linguistics.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461, Copenhagen, Denmark. Association for Computational Linguistics.

Shaowei Chen, Jie Liu, Yu Wang, Wenzheng Zhang, and Ziming Chi. 2020. Synchronous double-channel recurrent network for aspect-opinion pair extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6515–6524, Online. Association for Computational Linguistics.

Zhuang Chen and Tieyun Qian. 2019. Transfer capsule network for aspect level sentiment classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 547–556, Florence, Italy. Association for Computational Linguistics.

Zhuang Chen and Tieyun Qian. 2020. Relation-aware collaborative learning for unified aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3685–3694, Online. Association for Computational Linguistics.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with heat (hierarchical attention) network. In *CIKM*, pages 97–106.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167.

Hongliang Dai and Yangqiu Song. 2019. Neural aspect and opinion term extraction with mined rules as weak supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5268–5277, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Tong Xu, and Ming Liu. 2019. Capsule network with interactive attention for aspect-level sentiment classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5489–5498, Hong Kong, China. Association for Computational Linguistics.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1121–1131, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2019. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *ACL*, pages 504–515.

Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 537–546, Florence, Italy. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zeyang Lei, Yujiu Yang, Min Yang, Wei Zhao, Jun Guo, and Yi Liu. 2019. A human-like semantic cognition network for aspect-level sentiment classification. In *AAAI*, volume 33, pages 6650–6657.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956, Melbourne, Australia. Association for Computational Linguistics.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019. A unified model for opinion target extraction and target sentiment prediction. In *AAAI*, volume 33, pages 6714–6721.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018b. Aspect term extraction with history attention and selective transformation. In *IJCAI*, pages 4194–4200.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2886–2892, Copenhagen, Denmark. Association for Computational Linguistics.

Bin Liang, Jiachen Du, Ruifeng Xu, Binyang Li, and Hejiao Huang. 2019. Context-aware embedding for targeted aspect-based sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4678–4683, Florence, Italy. Association for Computational Linguistics.

Ming Liao, Jing Li, Haisong Zhang, Lingzhi Wang, Xixin Wu, and Kam-Fai Wong. 2019. Coupling global and local context for unsupervised aspect extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4579–4589, Hong Kong, China. Association for Computational Linguistics.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577, Valencia, Spain. Association for Computational Linguistics.

Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. 2015. Automated rule selection for aspect extraction in opinion mining. In *IJCAI*.

Huaishao Luo, Tianrui Li, Bing Liu, and Junbo Zhang. 2019a. DOER: Dual cross-shared RNN for aspect term-polarity co-extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 591–601, Florence, Italy. Association for Computational Linguistics.

Ling Luo, Xiang Ao, Yan Song, Jinyao Li, Xiaopeng Yang, Qing He, and Dong Yu. 2019b. Unsupervised neural aspect extraction with sememes. In *IJCAI*, pages 5123–5129.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring sequence-to-sequence learning in aspect term extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3538–3547, Florence, Italy. Association for Computational Linguistics.

Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2020. Knowing what, how and why: A near complete solution for aspect-based sentiment analysis. In *AAAI*, pages 8600–8607.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Minh Hieu Phan and Philip O. Ogunbona. 2020. Modelling context and syntactical features for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3211–3220, Online. Association for Computational Linguistics.

Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *SemEval 2014*, page 27.

Jialong Tang, Ziyao Lu, Jinsong Su, Yubin Ge, Linfeng Song, Le Sun, and Jiebo Luo. 2019. Progressive self-supervised attention learning for aspect-level sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 557–566, Florence, Italy. Association for Computational Linguistics.

Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, and Feng Wu. 2020. SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis. In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 4067–4076, Online. Association for Computational Linguistics.

Feixiang Wang, Man Lan, and Wenting Wang. 2018. Towards a one-stop solution to both aspect extraction and sentiment analysis tasks with neural multi-task learning. In *IJCNN*, pages 1–8.

Wenya Wang and Sinno Jialin Pan. 2019. Transferable interactive memory network for domain adaptation in fine-grained opinion extraction. In *AAAI*, volume 33, pages 7192–7199.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 616–626, Austin, Texas. Association for Computational Linguistics.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *AAAI*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Chi Xu, Hao Feng, Guo-Xin Yu, Min Yang, X. Wang, Yan Song, and Xiang Ao. 2021. Discovering protagonist of sentiment with aspect reconstructed capsule network. In *DASFAA*.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and CNN-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 592–598, Melbourne, Australia. Association for Computational Linguistics.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *EMNLP*, pages 4560–4570.

He Zhao, Longtao Huang, Rong Zhang, Quan Lu, and Hui Xue. 2020. SpanMlt: A span-based multi-task learning framework for pair-wise aspect and opinion terms extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3239–3248, Online. Association for Computational Linguistics.

Continual Mixed-Language Pre-Training for Extremely Low-Resource Neural Machine Translation

Zihan Liu, Genta Indra Winata, Pascale Fung
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
`zihan.liu@connect.ust.hk, pascale@ece.ust.hk`

Abstract

The data scarcity in low-resource languages has become a bottleneck to building robust neural machine translation systems. Fine-tuning a multilingual pre-trained model (e.g., mBART (Liu et al., 2020a)) on the translation task is a good approach for low-resource languages; however, its performance will be greatly limited when there are unseen languages in the translation pairs. In this paper, we present a continual pre-training (CPT) framework on mBART to effectively adapt it to unseen languages. We first construct noisy mixed-language text from the monolingual corpus of the target language in the translation pair to cover both the source and target languages, and then, we continue pre-training mBART to reconstruct the original monolingual text. Results show that our method can consistently improve the fine-tuning performance upon the mBART baseline, as well as other strong baselines, across all tested low-resource translation pairs containing unseen languages. Furthermore, our approach also boosts the performance on translation pairs where both languages are seen in the original mBART's pre-training. The code is available at `https://github.com/zliucr/cpt-nmt`.

1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017) has a poor generalization ability to low-resource languages where large monolingual and parallel corpora are not available. Recently, leveraging multilingual pre-trained models (Song et al., 2019; Liu et al., 2020a; Lin et al., 2020) as the starting checkpoints has shown to be effective at building low-resource NMT systems. However, the effectiveness of the pre-training will be vastly limited for low-resource languages that are not in the list of pre-training languages. Given the fact that there are more than 7000 languages around the world (Austin and Sallabank, 2011), it is almost impossible for a multilingual model to include all languages. And it is expensive and time-consuming to pre-train another model from scratch so as to include the languages we need. To address this issue, we propose to leverage the advantages of an off-the-shelf multilingual pre-trained model and focus on better generalizing it to any low-resource language pair. In this paper, we use mBART (Liu et al., 2020a) as the multilingual pre-trained model, given its effectiveness at building low-resource NMT systems.

To simulate the problem, we suppose that we need an NMT system on a low-resource translation pair, and at least one of the languages in the translation pair is an unseen language for the pre-trained model. To adapt mBART into unseen languages in the NMT task, we propose to conduct a continual pre-training (**CPT**) on it with mixed-language training (**MLT**). Concretely, we first follow the noise function used in Liu et al. (2020a) to corrupt the monolingual text of the target language in the translation. Then, we utilize a bilingual dictionary to generate mixed-language sentences and simultaneously delete some tokens based on the corrupted text. After that, we conduct the CPT on mBART to reconstruct the original monolingual text. After the CPT, we follow Liu et al. (2020a) to directly fine-tune mBART on the parallel data of the translation pair. The purpose of producing mixed-language sentences is to make a rough alignment between the languages in the translation pair. Conducting the token deletion is to increase the difficulty of the reconstruction task and the diversity of the noisy mixed-language text, which force the model to quickly learn an unseen language.

We consider an extremely low-resource setting where we have very few parallel data (10k) for

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2706–2718
August 1–6, 2021.

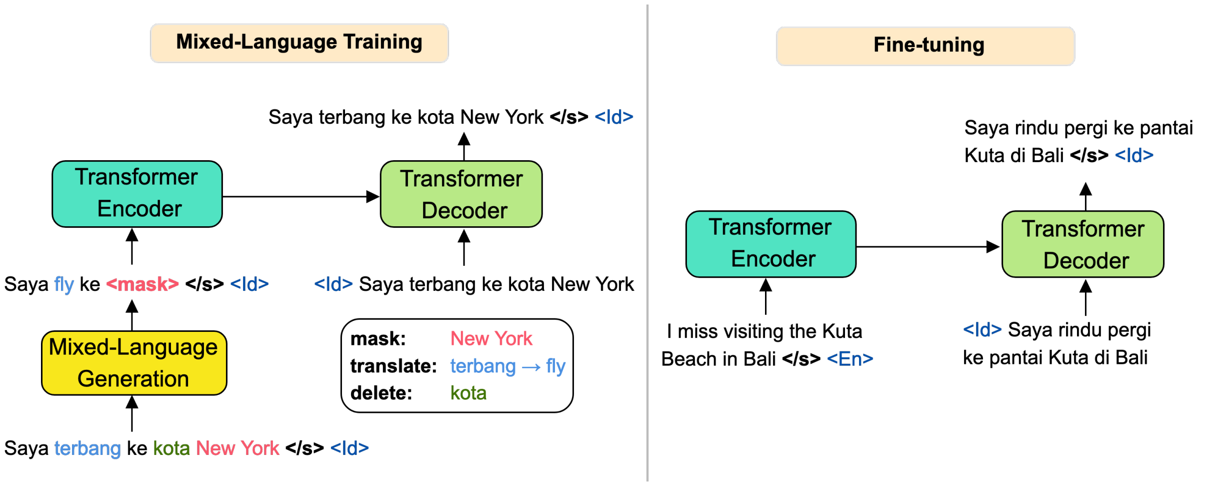

Figure 1: An illustration of adapting mBART to the En-Id translation pair: Continual pre-training with mixed-language training (left) and fine-tuning on the translation task (right).

low-resource translation pairs and very few monolingual data (100k) for each language in the translation. Experimental results show that our proposed pre-training approach is able to consistently outperform the mBART baseline as well as other pre-training baselines across all tested translation pairs that contain unseen languages. Interestingly, we observe that the continual mixed-language pre-training is even beneficial for a translation pair where both languages are in the mBART's pre-training list. Results also show that mBART can achieve better zero-shot performance after applying the CPT with MLT, which illustrates that the mixed-language pre-training is able to make a better alignment. Furthermore, we investigate our method in terms of various low-resource settings where different amounts of parallel and monolingual data are available, and experimental results show that the effectiveness of our approach can be further improved when a larger pre-training corpus is available.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to investigate how to effectively adapt a multilingual pre-trained model to unseen languages for the NMT task.
- We show that our proposed method can consistently surpass strong baselines across all the tested translation pairs.
- We conduct in-depth experiments and analyses in terms of different low-resource settings and the effectiveness on the various components of our method.

2 Methodology

In this section, we first give a brief overview of the mBART model (Liu et al., 2020a), and then we introduce our proposed method that aims to adapt mBART to unseen languages in the translation task.

2.1 Model: mBART

The mBART model follows the sequence-to-sequence (Seq2Seq) pre-training scheme of the BART model (Lewis et al., 2020) (i.e., reconstructing the corrupted text) and is pre-trained on large-scale monolingual corpora in 25 languages. Two types of noises are used to produce the corrected text. The first is to remove text spans and replace them with a mask token, and the second is to permute the order of sentences within each instance.

Thanks to the large-scale pre-training on multiple diverse languages, the mBART model has shown its strength at building low-resource NMT systems by being fine-tuned to the target language pair, and it is also shown to possess a powerful generalization ability to languages that do not appear in the pre-training corpora (Liu et al., 2020a).

2.2 Continual Pre-Training

Despite the powerful adaptation ability that mBART possesses, we argue that its performance on unseen languages is still sub-optimal since it has to learn these languages from scratch. Therefore, we propose to conduct the continual pre-training (CPT) on the mBART model to improve its adaptation ability to unseen languages. The process of this additional pre-training task is illustrated in Figure 1, and the details are described as follows.

Pre-Training We denote $\texttt{lang}_1 \rightarrow \texttt{lang}_2$ as the needed translation pair, where $\texttt{lang}_1$ is the source language and $\texttt{lang}_2$ is the target language, and at least one of them is an unseen language for the mBART model. The CPT can be considered as maximizing L_θ:

$$L_\theta = \sum_{X \in D_2} \log P(X | f(X); \theta), \quad (1)$$

where θ is initialized with mBART's parameters, D_2 denotes a collection of monolingual documents in $\texttt{lang}_2$, and f is a function to generate noisy mixed-language text that contains both $\texttt{lang}_1$ and $\texttt{lang}_2$.

Noisy Mixed-Language Function (f) Given a monolingual instance X, we first use the noise function (denoted as g, described in §2.1) used in Liu et al. (2020a) to corrupt the text, and then we use a dictionary of $\texttt{lang}_2$ to $\texttt{lang}_1$ to assist in the function of producing mixed-language sentences (denoted as h). Specifically, after the processing of the noise function g, if the non-masked tokens in $\texttt{lang}_2$ exist in the dictionary, we set a probability to replace it with its translation in $\texttt{lang}_1$. If it is not being replaced, there is a 50% chance that we will directly delete this token, and otherwise, we keep the original token in $\texttt{lang}_2$. More formally, function f (in Eq. (1)) can be considered as the combination of two functions:

$$f(X) = h(g(X)). \quad (2)$$

Notice that $\texttt{lang}_2$ is not always the unseen language (i.e., $\texttt{lang}_1$ could be the only unseen language). Since the inputs are mixed with the tokens in $\texttt{lang}_1$ and $\texttt{lang}_2$, the model can always learn the unseen language.

The reason why we choose to reconstruct $\texttt{lang}_2$ instead of $\texttt{lang}_1$ is because $\texttt{lang}_2$ is the target language that the decoder needs to generate in the translation task, and reconstructing $\texttt{lang}_2$ in the pre-training makes the model easier to adapt to the $\texttt{lang}_1 \rightarrow \texttt{lang}_2$ translation pair. We leverage the noise function g since it has shown its effectiveness at helping pre-trained models to obtain language understanding ability. The intuition of producing mixed-language text for inputs is to roughly align $\texttt{lang}_1$ and $\texttt{lang}_2$, since the model needs to understand the tokens of $\texttt{lang}_1$ so as to reconstruct the translations in $\texttt{lang}_2$. The purpose of not replacing all tokens in the dictionary with their translations is to increase the variety of the mixed-language text, and given that there will be plenty of frequent words (e.g., stopwords), replacing all of them with the corresponding translations could make the sentences unnatural, and the translations of the frequent words in $\texttt{lang}_1$ would likely not match the context in $\texttt{lang}_2$. In addition, adding a probability to delete the original token in function h is to inject extra noise and further increase the diversity of the generated mixed-language text.

3 Experimental Settings

3.1 Datasets

We conduct experiments on 12 low-resource language pairs from OpenSubtitles (Lison and Tiedemann, 2016), resulting in 24 directed translation pairs in total. Each pair has an unseen language for mBART. Concretely, there are 12 translation pairs (out of 24) containing English and another unseen language (Indonesian (Id), Ukrainian (Uk), Bengali (Bn), Afrikaans (Af), Tamil (Ta), Thai (Th) ↔ English (En)), and the rest of the 12 pairs contain two unseen languages (Id ↔ Ta, Bn ↔ Th, Bulgarian (Bg) ↔ Ta, Id ↔ Bn, Macedonian (Mk) ↔ Th, and Slovak (Sk) ↔ Swedish (Sv)). In addition, we evaluate the translation pairs (En ↔ Gujarati (Gu) and En ↔ Kazakh (Kk) (WMT19)), where both languages are in mBART's pre-training list.

To produce noisy mixed-language sentences, we collect monolingual corpora for the target languages from Wikipedia, and we utilize the bilingual dictionaries from MUSE (Lample et al., 2018b)[1] for the En-X and X-En pairs. For a dictionary (denoted as X-Y) that is not available in MUSE (English is not in the pair in this case), we first obtain the token list of language X from the X-En dictionary in MUSE, and then construct the X-Y dictionary utilizing Google Translate[2] to translate the tokens from language X to Y.

3.2 Low-Resource Settings

We focus on an extremely low-resource setting, where we assume that only 10K parallel samples are available. Considering that obtaining a large

---

[1] https://github.com/facebookresearch/MUSE

[2] https://translate.google.com. The constructed dictionaries will be released in `https://github.com/zliucr/cpt-nmt`.

size monolingual corpus could be difficult for some low-resource languages, we constrain the number of monolingual paragraphs to be as few as 100K (the size is $\sim$ 30MB). To do so, we randomly sample 10K parallel examples and 100K monolingual paragraphs from the available corpora. In addition, we also conduct experiments with different numbers of parallel data (from 10K to 100K) and monolingual data (from 100K to 1M) to investigate the effectiveness of the proposed method in different levels of low-resource setting. As for the translation pairs En $\leftrightarrow$ Gu and En $\leftrightarrow$ Kk, we follow the settings in Liu et al. (2020a) and use parallel data with a size of 10K and 91K for the En $\leftrightarrow$ Gu and En $\leftrightarrow$ Kk, respectively.

3.3 Models & Baselines

mBART We directly fine-tune the mBART model on the parallel data of the translation pair. Note that it is already a strong baseline since mBART is shown to possess a good generalization ability to unseen languages (Liu et al., 2020a).

CPT w/ Ori (Src) We follow the **original** objective function of mBART (only using the noise function g in §2.2 to corrupt the text) to continue pre-training it on the **source** language of the translation pair. [3] Then we directly fine-tune it on the translation parallel data.

CPT w/ Ori (Tgt) This baseline is the same as the previous one except that we continue pre-training mBART on the **target** language of the translation pair.

CPT w/ MLT (Src) Different from CPT w/ Ori, we use the noisy mixed-language function (f) to create noisy mixed-language text. However, different from what we propose in Eq. (1), it reverses the pre-training direction (i.e., it corrupts the text in the **source** language instead of the target language).

CPT w/ MLT (Tgt) This is our proposed method described in §2.2. We use Tgt or Src to distinguish the target or source language (in the translation pair), respectively, that mBART needs to reconstruct in the CPT.

mT5 Like mBART, mT5 (Xue et al., 2020) is also a multilingual pre-trained model using a Seq2Seq pre-training. It is pre-trained in 101 languages covering all the languages in our experimental settings. Note that we use the mT5-base (600M parameters) which has a similar size as mBART (610M parameters) to ensure the fair comparison.

3.4 Training Details

Given that the sizes of the pre-training data and the parallel data are relatively small, we freeze the first 8 layers (out of 12) of the encoder and the first 8 layers (out of 12) of the decoder in the CPT, as well as the fine-tuning processes (applied for both mBART and mT5), to avoid the over-fitting issue. Note that we still keep the embeddings layer unfrozen since the model needs to learn the embeddings for unseen languages. For CPT, we control the probability of whether to replace a token with its translation to ensure around 30% of tokens are replaced. In the CPT stage, we train with a dropout rate of 0.1, a batch size of 100, and a learning rate of 3e-5 for 5 epochs. In the fine-tuning stage, we train with a dropout rate of 0.3, a batch size of 32, and 2500 warm-up steps with a maximum learning rate of 5e-5 for all directions. We use the Adam optimizer (Kingma and Ba, 2015) for both the CPT and fine-tuning processes. We set the maximum fine-tuning epochs as 20, and the final model is selected based on the performance on the validation dataset. The final results are reported in the case-sensitive tokenized BLEU (Papineni et al., 2002). We notice that the tokenizer of mBART is the same as that of XLM-R (Conneau et al., 2020) which covers 100 languages. Note that extending the vocabulary may be necessary for new languages that are not included in the original tokenizer, while we do not extend the vocabulary in the experiments since all the languages in the experiments are included in the vocabulary of XLM-R, and we find that the unknown token rates for unseen languages in the experiments are zero. Therefore, for all the models, we directly use mBART's tokenizer on the text for all languages in the experiments to ensure a fair comparison in BLEU, and we use thai-segmenter [4] to pre-tokenize the text in Thai (Th) before using mBART's tokenizer. For inference, we use beam search with a beam size of 5 for all directions.

4 Results & Analysis

4.1 Main Results

The results of our proposed methods and baseline models are illustrated in Table 1, from which we can observe that conducting CPT on mBART is

---

[3] For example, in the Id → Ta translation, the source language is Id and the target language is Ta.

[4] https://pypi.org/project/thai-segmenter/

Language Pairs	En-Id		En-Uk		En-Bn		En-Af		En-Ta		En-Th	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
mT5	8.15	6.98	4.53	0.68	1.50	0.34	7.68	8.83	1.98	2.15	2.87	2.19
mBART	8.87	7.38	4.85	0.89	1.37	0.65	8.24	10.02	4.07	2.70	3.12	2.41
CPT w/ Ori (Src)	9.05	7.41	5.49	1.11	1.90	0.76	8.29	9.32	3.80	4.05	3.17	3.16
CPT w/ Ori (Tgt)	8.78	7.77	5.75	1.31	2.03	0.92	8.31	9.71	3.46	4.26	3.08	3.57
CPT w/ MLT (Src)	10.44	8.40	5.22	1.45	2.21	**1.43**	8.58	10.12	4.28	5.05	3.42	**4.80**
CPT w/ MLT (Tgt)	**11.16**	**10.30**	**6.50**	**1.48**	**2.73**	1.25	**10.56**	**11.62**	**6.21**	**5.20**	**3.85**	4.54

Language Pairs	Id-Ta		Bn-Th		Bg-Ta		Id-Bn		Mk-Th		Sk-Sv	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
mT5	0.83	0.45	0.00	0.21	0.33	0.22	0.10	0.07	0.32	0.23	0.44	1.83
mBART	1.21	0.98	0.00	0.00	0.52	0.26	0.00	0.00	0.29	0.41	0.38	1.76
CPT w/ Ori (Src)	0.93	1.49	0.00	0.52	0.39	0.30	0.41	0.22	0.48	0.60	0.73	1.57
CPT w/ Ori (Tgt)	1.24	1.24	0.00	0.64	0.41	0.61	0.33	0.30	0.51	0.67	0.78	2.09
CPT w/ MLT (Src)	1.39	**1.90**	0.00	0.09	0.66	0.52	**0.54**	**0.31**	**0.73**	**1.21**	0.83	2.21
CPT w/ MLT (Tgt)	**2.52**	1.75	**0.20**	**0.66**	**0.95**	**0.85**	0.36	**0.31**	0.69	1.15	**0.99**	**2.55**

Table 1: Fine-tuning performance on the 10K parallel data for the 24 translation pairs. All CPT methods utilize a corpus with a size of 100K paragraphs. The upper 12 pairs contain one unseen language for mBART (the other seen language is English), and the bottom 12 pairs contain two unseen languages. The CPT using our proposed method consistently outperforms all baseline models.

generally effective in the low-resource scenario of the NMT task, although the size of the pre-training corpus is as few as 100K paragraphs. Also, we can see that the CPT w/ MLT consistently outperforms all baseline models since the additional mixed-language information helps to construct a better alignment between the source and target languages in the translation pair. We observe that the CPT w/ MLT (Tgt) significantly outperforms mBART in multiple translation pairs (e.g., 2.92 BLEU points in En → Id and 2.39 BLEU points in En → Th). We find that, although conducting CPT (w/ Ori or w/ MLT) on the text that contains tokens in the unseen language generally enhance the performance in the translation, the effectiveness of CPT w/ Ori is relatively deficient compared to CPT w/ MLT. We conjecture that the original objective function of mBART loses its advantages when the amount of pre-training monolingual data is small, while MLT is still beneficial thanks to the additional bilingual alignments that it have learned.

Additionally, we find that the direction of the CPT (Src or Tgt) also plays an important role. As we can see from Table 1, conducting CPT by reconstructing the target language in the translation pair generally achieves better performance than reconstructing the source. We conjecture that making the generated language in the CPT stage consistent with that in the fine-tuning stage will increase the benefits from the CPT. This is because, if the generated languages are different in these two stages, the model needs to learn to generate sentences on an entirely different language with only a few data samples in the fine-tuning stage, which could make the fine-tuning task much more challenging. Interestingly, when English (a seen language) is the target language, the CPT w/ Ori (Tgt) becomes less effective, but CPT w/ MLT (Tgt) still works well. The reason is that CPT w/ Ori (Tgt) ignores the unseen language in the continual pre-training stage, while the mixed-language inputs of CPT w/ MLT (Tgt) still contain the tokens in the unseen language, which still enables the model to learn the unseen language. Surprisingly, mT5 performs generally worse than mBART, although it covers all the languages in our experiments. We conjecture that, since the objective function of mT5 is to generate the masked tokens, it makes the averaged length of the generated text relatively shorter than mBART, which might limit its ability to quickly adapt to a generation task in the low-resource scenario.

4.2 Different Low-Resource Settings

In this section, we investigate whether our method can generalize to other low-resource settings (i.e., different sizes of the parallel data and monolingual

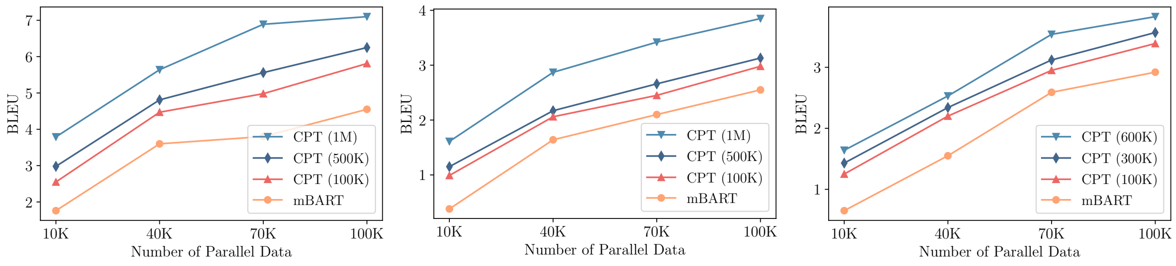

(a) Sk → Sv

(b) Sv → Sk

(c) En → Bn

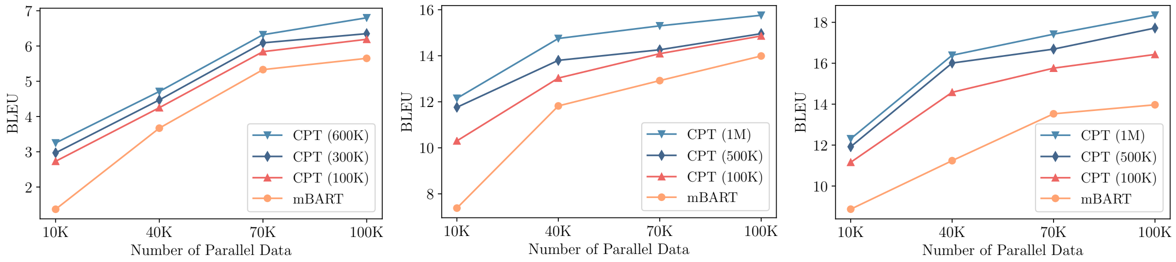

(d) Bn → En

(e) En → Id

(f) Id → En

Figure 2: The performance over different numbers of parallel data (from 10K to 100K) and pre-training data (from 100K to 1M). CPT denotes our method, CPT w/ MLT (Tgt). Since the maximum number of paragraphs in Wikipedia for Bn is ∼600K, we set the data size in the CPT as 100K, 300K and 600K for En ↔ Bn.

data). We choose three translation pairs (En ↔ Bn, En ↔ Id, and Sk ↔ Sv), which cover two scenarios: 1) only one unseen language in a translation pair; and 2) both languages in a translation pair are unseen. As illustrated in Figure 2, we can observe that our method is able to consistently improve on the mBART baseline in terms of different parallel data sizes, and the improvements can be further boosted when the size of the pre-training data (monolingual data) increases. This is because a larger corpus is able to amplify the benefits of MLT and better align the space between the two languages in the translation. Moreover, we find that our method is especially effective for the Sk ↔ Sv translation pair when the size of the pre-training data reaches 1M. For example, in the Sk → Sv translation, the performance of CPT (1M) with 10K parallel samples (3.79) is on par with mBART with 70K parallel samples (3.80), which might suggest that gathering larger monolingual data (along with a dictionary) can be an alternative to collecting a larger size of parallel data.

4.3 Effectiveness on Seen Languages

As we can see from Table 2, the CPT w/ MLT can also significantly improve the performance on the translation pairs where both languages are in the mBART's pre-training list. The CPT w/ MLT improves by at least 1.2 BLEU points on all translation pairs with only 100K pre-training data. Additionally, the improvement brought by our method can be further boosted when a larger pre-training corpus is available, which accords with the experimental results for the unseen languages.

Language Pairs	**En-Gu**		**En-Kk**	
Direction	←	→	←	→
mBART	3.11	0.10	8.93	2.44
CPT w/ MLT (Tgt, 100K)	4.59	2.01	10.16	4.01
CPT w/ MLT (Tgt, 500K)	5.44	2.91	10.74	4.74
CPT w/ MLT (Tgt, 1M)	**5.97**	**3.89**	**11.45**	**5.29**

Table 2: The effectiveness of our method on seen languages. 100K, 500K and 1M are the corpus sizes for the CPT w/ MLT (Tgt).

Models	**mBART**	**CPT w/ Ori**	**CPT w/ MLT**
Avg	0.00	0.27	**0.53**

Table 3: Averaged performance over the 24 translation pairs in the zero-shot test. Both CPT methods are to reconstruct the target language with 100K samples.

We conjecture two reasons: 1) Continuing pre-training mBART can make the model focus on the languages in the translation pair and increase the model's ability of fast adaptation to the translation task. 2) Continual pre-training with the mixed-language text can further align the two languages in the translation, which gives a better initialization for the low-resource translation task.

Language Pairs	En-Id		En-Af		Id-Ta		Sk-Sv	
Direction	←	→	←	→	←	→	←	→
mBART	8.87	7.38	8.24	10.02	1.21	0.98	0.38	1.76
CPT w/ MLT (Tgt)	**11.16**	**10.30**	**10.56**	**11.62**	**2.52**	**1.75**	**0.99**	**2.55**
w/o noise	11.06	9.94	8.95	10.65	2.11	1.29	0.79	1.95
w/o deletion	10.33	9.57	8.62	10.24	1.90	1.19	0.86	2.13
w/o noise & deletion	10.12	9.03	8.79	10.08	1.84	1.07	0.73	1.92

Table 4: Ablation study on the noise function g (denoted as noise) and token deletion (denoted as deletion).

Language Pairs	En-Id		En-Th	
Direction	←	→	←	→
mBART	8.87	7.38	3.12	2.41
CPT w/ MLT (10%)	9.65	9.14	3.28	3.15
CPT w/ MLT (20%)	9.56	9.98	3.49	3.78
CPT w/ MLT (30%)	**11.16**	10.30	**3.85**	**4.54**
CPT w/ MLT (40%)	10.78	**10.34**	3.56	4.49
CPT w/ MLT (50%)	10.06	9.97	3.15	3.67

Table 5: Effectiveness of CPT w/ MLT (Tgt) in terms of different language mixing ratios. The ratio in the brackets denotes the number of source language tokens (in the translation pair) divided by that of the target language tokens.

Language Pairs	En-Th			
Direction	⟵		⟶	
Pre-Tokenization	✓	✗	✓	✗
mBART	**3.12**	3.04	**2.41**	0.43
CPT w/ Ori (Src)	3.17	**3.18**	**3.16**	0.53
CPT w/ Ori (Tgt)	3.08	**3.09**	**3.57**	0.56
CPT w/ MLT (Src)	**3.42**	3.37	**4.80**	0.64
CPT w/ MLT (Tgt)	**3.85**	3.79	**4.54**	0.70

Table 6: Comparison between conducting and not conducting the pre-tokenization for Thai.

4.4 Zero-shot Performance

To further analyze the alignment quality between the source and target languages in the translation after the CPT, we evaluate the models in the zero-shot scenario, where we directly test the pre-trained models on the test set without any fine-tuning on the parallel data. As illustrated in Table 3, we can see that the zero-shot performance is relatively low since the models are not trained on any parallel or pseudo-parallel data [5], and mBART gets 0 BLEU points due to the unseen languages in the test data.

We find that CPT w/ Ori achieves more than 0 BLEU points, even though it does not utilize any supervision from the bilingual text. We conjecture that this can be attributed to the multilingual ability of mBART. Furthermore, CPT w/ MLT is able to outperform CPT w/ Ori since it learns additional bilingual alignments by reconstructing the target documents from the mixed-language text. In addition, the results are able to further illustrate that our method is able to achieve a better alignment quality than the baseline method.

4.5 Ablation Study

In this section, we first explore how the noise function g and token deletion in function h affect the effectiveness of our method (g and h are described in §2.2). Then, we investigate how the language mixing ratio of the mixed-language text affects our method's performance.

Noise & Deletion As shown in Table 4, we can see that both the noise function and token deletion play an important part in the CPT, and removing both of them further degrades the performance. Given that the number of pre-training documents is as few as 100K, it is relatively difficult for the model to learn a good representation for the unseen language. However, adding the noise function in the CPT forces the model to learn to perform text infilling and sentence reordering, which increases the model's ability to understanding the unseen language. Conducting the token deletion brings two benefits: 1) It increases the variety of the mixed-language text, which makes the model not overfit to a certain mixed-language pattern. 2) It also injects extra noise to the inputs, which further compels mBART to understand the unseen language better. Moreover, incorporating both noise function g and the token deletion further boosts the effectiveness of the pre-training.

---

[5] The results for each translation pair are in Appendix A.

Language Mixing Ratio We control the probability of whether to replace a token with its translation to generate different settings of language mixing ratios and investigate how different ratios affect the effectiveness of the pre-training. As shown in Table 5, using a too high or too low mixing ratio will degrade the advantages of the CPT w/ MLT, [6] and keeping the ratio between 30% and 40% will achieve the best performance. We conjecture that, if the mixing ratio is too low, the dictionary which provides the supervision of bilingual alignment is not well utilized, while if the mixing ratio is too high (e.g., 50%), we replace almost all the tokens existing in the dictionary, which lowers the diversity of the mixed-language text and makes the model more easily overfit to the pre-training data.

4.6 Importance of Pre-Tokenization

Considering that the tokenizer of mBART is created based on the text of the pre-training languages, it might not perform good tokenization for the unseen languages that are diverse from the pre-trained languages. Therefore, it could be a better option to pre-tokenize the text before using mBART's tokenizer. We conduct experiments on the En-Th language pair and compare the performance between performing and not performing the pre-tokenization for Thai. As shown in Table 6, we find that pre-tokenization is able to improve the performance in En → Th significantly, while the improvements are marginal in Th → En. We conjecture that decoding (generating) tokens in the unseen language is much more difficult than encoding those tokens when they are not properly tokenized. This is because the task of the encoder is to understand the meaning of the input text, while the decoder needs to attend to the input text and generate tokens simultaneously, which makes the task of the decoder more difficult than that of the encoder. Therefore, when the unseen language (Thai) becomes the target language in the translation pair, the performance drops remarkably without pre-tokenization.

5 Related Work

5.1 Multilingual Pre-Trained Models

Recently, multilingual pre-trained models based on the masked language modeling (MLM) objective function (Devlin et al., 2019; Conneau and Lample, 2019; Huang et al., 2019; Conneau et al., 2020) have shown their effectiveness at performing cross-lingual classification-based tasks. However, these models are inferior to the generation tasks (Rönnqvist et al., 2019) since they are not pre-trained in a generative way. Multilingual pre-training performed in a Seq2Seq fashion is able to mitigate this issue (Radford et al.; Lewis et al., 2020; Raffel et al., 2019), and has become a strong backbone for building NMT systems, especially in a low-resource scenario (Liu et al., 2020a; Song et al., 2019; Lin et al., 2020; Yang et al., 2020; Xue et al., 2020; Fan et al., 2020; Tang et al., 2020). Liu et al. (2020a) pre-trained a Seq2Seq multilingual model (mBART) by denoising full texts in 25 languages, while Lin et al. (2020) proposed multilingual random aligned substitution to pre-train an NMT model for many languages based on parallel data. Instead of pre-training models from scratch, Wang et al. (2020) proposed to extend multilingual BERT (Devlin et al., 2019) to an unseen language and evaluate it on the named entity recognition task. Although many studies have focused on pre-training multilingual models, few have investigated how to adapt the pre-trained models to new languages effectively. Also, to the best of our knowledge, we are the first to explore how to adapt a multilingual model pre-trained in a Seq2Seq fashion to unseen languages and evaluate the methods on a generative task (the NMT task).

5.2 Low-Resource Machine Translation

Recently, developing algorithms that are able to cope with the scenario where the training data are insufficient have become an interesting and popular research topic across a variety of tasks (Chen et al., 2019a,b, 2020; Brown et al., 2020; Liu et al., 2020c; Lauscher et al., 2020; Winata et al., 2020; Liu et al., 2020b; Peng et al., 2020; Liu et al., 2020d; Yu et al., 2021; Winata et al., 2021). Low-resource machine translation systems (Vandeghinste et al., 2007; Irvine and Callison-Burch, 2013; Zoph et al., 2016; Sennrich et al., 2016; Fadaee et al., 2017; Currey et al., 2017; Imankulova et al., 2017; Gu et al., 2018a; Pourdamghani et al., 2018; Gu et al., 2018b; Lample et al., 2018a,c; Kocmi and Bojar, 2018; Artetxe et al., 2018; Lakew et al., 2018; Imankulova et al., 2019a; Xia et al., 2019; Liu et al., 2019; Guzmán et al., 2019; Imankulova et al., 2019b; Stickland et al., 2020; Siddhant et al., 2020) alleviated the parallel data scarcity issue for low-

---

[6] Note that the maximum mixing ratio will not be larger than 55% since there are substantial infrequent tokens in the target language not existing in the dictionary, which will not be replaced with the source language tokens.

resource languages and improve the models' generalization ability for low-resource language pairs. Pourdamghani et al. (2018) proposed to improve the low-resource NMT performance by boosting the quality of word alignments. Gu et al. (2018b) applied the meta-learning approach into the low-resource NMT task, and Baziotis et al. (2020) incorporated a language model prior to regularize the output distribution of the translation model. Pre-training a multilingual Seq2Seq model (Liu et al., 2020a; Lin et al., 2020) allows it to be directly fine-tuned for supervised machine translation tasks and produces remarkable performance gains in the low-resource scenario over those without pre-training.

6 Conclusion & Future Work

In this paper, we present a continual pre-training framework to improve mBART's generalization ability to extremely low-resource translation pairs that contain unseen languages. We propose to construct noisy mixed-language text from the monolingual corpus to cover both the source and target languages, and then, we continue pre-training mBART to reconstruct the original monolingual text. Results illustrate that our method is able to consistently surpass strong baselines across all tested translation pairs that contain unseen languages, as well as the ones where both languages are seen in the original mBART's pre-training. Moreover, we observe that our method is also beneficial for different low-resource settings, and its performance can be further boosted when a larger pre-training corpus is available. Furthermore, we find that not only mixing the source and target languages, but also increasing the variety of the inputs plays an essential role in the continual mixed-language pre-training. In future work, we will explore more pre-training methods to further boost the performance of pre-trained models on the NMT task. Additionally, we will study more applications of continual mixed-language pre-training, such as applying it to downstream cross-lingual tasks.

Acknowledgement

We want to say thanks to the anonymous reviewers for the insightful reviews and constructive feedback. This work is partially funded by ITF/319/16FP and MRP/055/18 of the Innovation Technology Commission, the Hong Kong SAR Government.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Peter K Austin and Julia Sallabank. 2011. *The Cambridge Handbook of Endangered Languages*. Cambridge University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Christos Baziotis, Barry Haddow, and Alexandra Birch. 2020. Language model prior for low-resource neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7622–7634, Online. Association for Computational Linguistics.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019a. Meta relational learning for few-shot link prediction in knowledge graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4208–4217.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019b. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot nlg with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Anna Currey, Antonio Valerio Miceli-Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, et al. 2020. Beyond english-centric multilingual machine translation. *arXiv preprint arXiv:2010.11125*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018a. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018b. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6100–6113.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2485–2494.

Aizhan Imankulova, Raj Dabre, Atsushi Fujita, and Kenji Imamura. 2019a. Exploiting out-of-domain parallel data through multilingual transfer learning for low-resource neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 128–139.

Aizhan Imankulova, Takayuki Sato, and Mamoru Komachi. 2017. Improving low-resource neural machine translation with filtered pseudo-parallel corpus. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 70–78.

Aizhan Imankulova, Takayuki Sato, and Mamoru Komachi. 2019b. Filtered pseudo-parallel corpus improves low-resource neural machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(2):1–16.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 262–270.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Tom Kocmi and Ondřej Bojar. 2018. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252.

Surafel M Lakew, Marcello Federico, Matteo Negri, and Marco Turchi. 2018. Multilingual neural machine translation for low-resource languages. *IJCoL. Italian Journal of Computational Linguistics*, 4(4-1):11–25.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018b. Word translation without parallel data. In *International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018c. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649–2663.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020a. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Zihan Liu, Genta Indra Winata, Samuel Cahyawijaya, Andrea Madotto, Zhaojiang Lin, and Pascale Fung. 2020b. On the importance of word order information in cross-lingual sequence labeling. *arXiv e-prints*, pages arXiv–2001.

Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020c. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8433–8440.

Zihan Liu, Yan Xu, Genta Indra Winata, and Pascale Fung. 2019. Incorporating word and subword units in unsupervised machine translation using language model rescoring. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 275–282.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2020d. Crossner: Evaluating cross-domain named entity recognition. *arXiv preprint arXiv:2012.04373*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 172–182.

Nima Pourdamghani, Marjan Ghazvininejad, and Kevin Knight. 2018. Using word vectors to improve word alignments for low resource machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 524–528.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Samuel Rönnqvist, Jenna Kanerva, Tapio Salakoski, and Filip Ginter. 2019. Is multilingual BERT fluent in language generation? In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 29–36, Turku, Finland. Linköping University Electronic Press.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Xu Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2827–2835.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2020. Recipes for adapting pre-trained monolingual and multilingual models to machine translation. *arXiv preprint arXiv:2004.14911*.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Vincent Vandeghinste, Ineke Schuurman, Michael Carl, Stella Markantonatou, and Toni Badia. 2007. Metis-ii: Machine translation for low resource languages. *METIS*, 2:10–2004.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? *arXiv preprint arXiv:2103.13309*.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, Peng Xu, and Pascale Fung. 2020. Learning fast adaptation on cross-accented speech recognition. *Proc. Interspeech 2020*, pages 1276–1280.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized data augmentation for low-resource translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. Csp: Code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636.

Tiezheng Yu, Zihan Liu, and Pascale Fung. 2021. Adaptsum: Towards low-resource domain adaptation for abstractive summarization. *arXiv preprint arXiv:2103.11332*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.

A Zero-shot Performance

The zero-shot performance for the 24 translation pairs are shown in Table 7 (in the next page). We find that the CPT w/ MLT generally outperforms the CPT w/ Ori. However, the zero-shot results are relatively low, especially for the translation pairs where both languages are unseen in the original mBART's pre-training, due to the absence of parallel data.

B Data & Code

We will release our split data, dictionaries, as well as the code to ensure the reproducibility of our work.

Language Pairs	**En-Id**		**En-Uk**		**En-Bn**		**En-Af**		**En-Ta**		**En-Th**	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
mBART	0.04	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPT w/ Ori (Tgt)	1.35	1.16	0.46	**0.43**	0.07	**0.39**	0.95	0.78	0.06	0.13	0.16	0.38
CPT w/ MLT (Tgt)	**1.80**	**1.24**	**1.68**	0.41	**0.56**	0.36	**2.43**	**1.52**	**0.54**	**0.17**	**1.02**	**0.52**
Language Pairs	**Id-Ta**		**Bn-Th**		**Bg-Ta**		**Id-Bn**		**Mk-Th**		**Sk-Sv**	
Direction	←	→	←	→	←	→	←	→	←	→	←	→
mBART	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CPT w/ Ori (Tgt)	0.00	0.00	**0.02**	0.00	0.00	0.00	0.06	0.04	0.00	0.02	0.00	0.00
CPT w/ MLT (Tgt)	0.00	0.00	0.00	0.00	0.00	0.00	**0.14**	**0.05**	**0.07**	**0.03**	0.00	**0.20**

Table 7: Zero-shot results for the 24 translation pairs.

Transformer-Exclusive Cross-Modal Representation for Vision and Language

Andrew Shin
Sony Group Corporation
`andrew.shin@sony.com`

Takuya Narihira
Sony Group Corporation
`takuya.narihira@sony.com`

Abstract

Ever since the advent of deep learning, cross-modal representation learning has been dominated by the approaches involving convolutional neural networks for visual representation and recurrent neural networks for language representation. Transformer architecture, however, has rapidly taken over the recurrent neural networks in natural language processing tasks, and it has also been shown that vision tasks can be handled with transformer architecture, with compatible performance to convolutional neural networks. Such results naturally lead to speculation upon the possibility of tackling cross-modal representation for vision and language exclusively with transformer. This paper examines transformer-exclusive cross-modal representation to explore such possibility, demonstrating its potentials as well as discussing its current limitations and its prospects.

1 Introduction

While early cross-modal models handled visuolinguistic tasks with template-based methods (Barbu et al., 2012; Elliott and Keller, 2013), or as a retrieval model (Farhadi et al., 2010; Ordonez et al., 2011), the advent of deep learning introduced end-to-end learning models for cross-modal tasks, in which convolutional neural networks (CNNs) (Krizhevsky et al., 2012) are employed for vision representation, whereas recurrent neural networks (RNNs), such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014), are employed for language representation. While a plethora of variations exist, most models proposed in the past few years have invariably relied on the CNN-RNN approach.

Such standardized scheme, however, started to change with the introduction of transformer architecture based on multi-head attention mechanism (Vaswani et al., 2017), which rapidly started to achieve state-of-the-art performance in natural language processing (NLP) (Peters et al., 2018; Dai et al., 2019; Yang et al., 2019) and speech recognition domains (Dong et al., 2018; Wang et al., 2020b), frequently outperforming RNNs. Furthermore, large-scale models based on transformer architecture, such as BERT (Devlin et al., 2019) or GPT-3 (Brown et al., 2020), started to appear, demonstrating that pre-training a sufficiently large model with a very large amount of data results in strong performance with versatility for various downstream tasks.

The success of transformer-based models in NLP and speech recognition naturally led to its adaptation in cross-modal tasks. (Lu et al., 2019) proposed ViLBERT, a pioneering BERT-inspired work that proposed to tokenize the images for compatibility with transformer architecture, and also to extend the pre-training objectives of BERT to reflect the nature of cross-modality. Many other cross-modal models followed, but mostly with similar approaches for image tokenization and pre-training objectives. This line of transformer-based cross-modal works described above, however, still heavily relied on CNN-based models, such as Faster R-CNN (Ren et al., 2015), to extract features from images, and the application of transformer was mostly limited to language representation.

Inspired by the observations made by recent works (Dosovitskiy et al., 2020), which demonstrate that vision tasks can be handled solely by transformer architecture with compatible performances to CNN-based models, this paper examines cross-modal representation for visuolinguistic tasks relying exclusively on transformer architecture, without using CNNs or RNNs. Without any structural modifications or advanced common embedding scheme, and without additional cross-modal pre-training that can be expensive both

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2719–2725
August 1–6, 2021.

computationally and data-wise, our model demonstrates comparable performances to conventional approaches based on CNNs and RNNs in exemplary cross-modal tasks.

2 Related Works

ViLBERT (Lu et al., 2019) was one of the first models to extend transformer architecture to cross-modal visuolinguistic tasks. They propose co-attentional transformer, in which separate transformer modules for each modality run in parallel, with the key and value inputs from one modality entering the transformer block for the other modality, thereby learning cross-modal dependence. In order to tokenize the image, they extract image regions using Faster R-CNN (Ren et al., 2015) along with 5-dimensional location vector. They also extend two unique pre-training objectives of BERT, namely masked language modeling and next-sentence prediction, to cross-modal setting, as masked multi-modal learning and image-sentence alignment classification. In masked multi-modal learning, visual tokens, along with language tokens, are randomly masked, and the model is trained to predict their probability distribution over object classes. In image-sentence alignment classification, a sequence of visual tokens and a sentence are juxtaposed, and the model performs a binary classification task, predicting whether the sentence describes the contents of the image. Many other models, such as VisualBERT (Li et al., 2019), LXMERT (Tan and Bansal, 2019) and Unicoder-VL (Li et al., 2020), also follow nearly identical pre-training objectives as VilBERT. On the other hand, UNITER (Chen et al., 2020) demonstrates improved performance by introducing additional pre-training objective of word region alignment, while MiniVLM (Wang et al., 2020a) achieves comparable performance with up to 70% fewer parameters by utilizing EfficinetNet (Tan and Le, 2019) with their own Compact BERT model.

While all models described above rely on CNN-based models to extract features from images, limiting the scope of applicability of transformer, recent works have demonstrated results that may imply a potential change in such workflow. (Dosovitskiy et al., 2020) proposed Vision Transformer (ViT), which demonstrates that pure transformer architecture without convolution can achieve comparable performance in image classification tasks, while requiring substantially less computational costs. Furthermore, (Touvron et al., 2021) showed via data-efficient image transformers (DeiT) that competitive performance can be achieved with training only on ImageNet (Deng et al., 2009) with no external data.

3 Model

We employ separate transformer models for vision and language, although internal mechanisms are essentially identical. Following (Dosovitskiy et al., 2020), we split an image into N patches x_p of $P \times P$ pixels, each of which is linearly projected into D-dimensional patch embedding, where $P = 16$, and $D = 768$. A learnable embedding x_{class} is prepended to patch embeddings, and positional embeddings are also added. The input sequence z_0 subsequently undergoes alternating layers of layer normalization (Ba et al., 2016) and multi-head self-attention, followed by a 2-layer MLP with GELUs (Hendrycks and Gimpel, 2020) as non-linearity:

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; ...; x_p^N E] + E_{pos}, \quad (1)$$

$$E \in \mathbb{R}^{(P^2 C)\times D}, E_{pos} \in \mathbb{R}^{(N+1)+D}$$

$$z'_l = \text{MSA}(\text{LN}(z_{l-1})) + z_{l-1}, l = 1...L \quad (2)$$

$$z_l = \text{MLP}(\text{LN}(z'_l)) + z'_l, l = 1...L \quad (3)$$

$$y_{img} = \text{LN}(z_L^0) \quad (4)$$

where

$$\text{MSA}(X) = W_{att}[\text{Att}_1(X), ..., \text{Att}_m(X)]^\top \quad (5)$$

$$\text{Att}_i(X) = \text{softmax}\frac{((W_{Q_i}X)^\top W_{K_i}X)}{\sqrt{D/m}}(W_{V_i}X)^\top \quad (6)$$

for input layer $X \in \mathbb{R}^{D\times N}$, and learnable parameters $W_{Q_i}, W_{K_i}, W_{V_i} \in \mathbb{R}^{\frac{D}{m}\times D}$, $W_{att} \in \mathbb{R}^{D\times D}$ for m attention heads.

For language representation, we employ an off-the-shelf BERT model. An input sequence $s_0 = [w_0, ..., w_S]$ is given with special tokens $[CLS]$ and $[SEP]$ inserted at the beginning and the end of the sequence respectively. In case of two sentences within the input sequences, $[SEP]$ token is also inserted in between the two. Each token is represented as the sum of word embedding, position embedding, and segment embedding, and undergoes bidirectional multi-head self-attention over multiple layers. The representation for the input sequence is obtained as $h_0, ..., h_S$ from the upper-

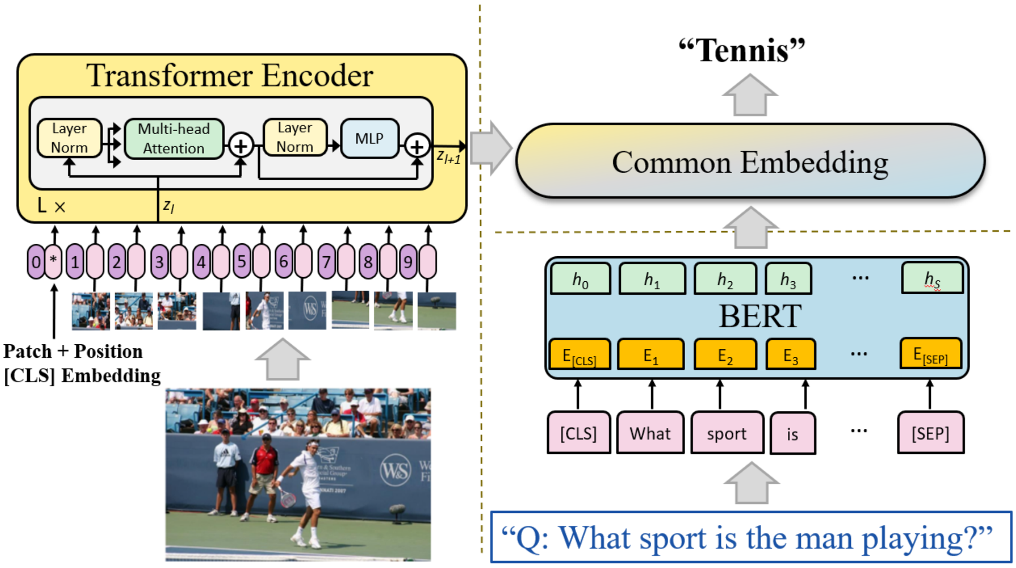

Figure 1: Overview of our model. Images are split into patches, sentences are split into tokens, and both are encoded with transformers, after which they are combined in common embedding space for final classification.

most attention layer:

$$s_0 = [[CLS], w_1, ..., w_{S-1}, [SEP]] \quad (7)$$

$$s'_l = \text{MSA}(\text{LN}(s_{l-1})) + s_{l-1}, l = 1...L \quad (8)$$

$$s_l = \text{MLP}(\text{LN}(s'_l)) + s'_l, l = 1...L \quad (9)$$

$$y_{lang} = \text{LN}(s^0_L) \quad (10)$$

We now project the image and language representations obtained into common embedding space by concatenation:

$$y = \text{Concat}(y_{img}, y_{lang}) \quad (11)$$

Note that we deliberately choose the most elementary common embedding scheme, as our focus is to examine the performance of the features themselves, rather than the embedding scheme. It is thus highly likely that, when coupled with more sophisticated embedding schemes, a significant performance boost will occur. Fig. 1 describes the overall architecture of our approach.

4 Experiments

4.1 Setting

For images, we use ViT-B model pre-trained on ImageNet-21k. The model contains 12-layers with 12 attention heads and hidden size of 768, consisting of 86M parameters. For language, we use off-the-shelf BERT$_{\text{BASE}}$ mode, trained with BERT's pre-training objectives of masked language modeling and next sentence prediction on BookCorpus (Zhu et al., 2015) and English Wikipedia. Like ViT-B, the model contains 12-layers with 12 attention heads and hidden size of 768, and consists of 110M parameters. During both training and testing, image and language features are extracted from the uppermost layer of respective model, and we concatenate them to make a 1536-dimensional vector. Concatenated features are trained with cross-entropy loss and Adam (Kingma and Ba, 2014) optimizer.

We evaluate our model on the following commonly tackled cross-modal visuolinguistic tasks; visual question answering (VQA) (Antol et al., 2015; Goyal et al., 2017), visual commonsense reasoning (VCR) (Zellers et al., 2019), and reasoning about natural language grounded in photographs (NLVR2) (Suhr et al., 2019). For VCR, we followed (Lu et al., 2019) by making 4 possible pairs of question and answer. For NLVR2, we follow the *pair* approach of (Chen et al., 2020), by embedding each image and the query, as it is reported to outperform *triplet* approach of embedding two images with the query. We trained with 4 V100 GPUs with batch size 96 for VQA and NLVR2, and 48 for VCR, which were adjusted with respect to the memory constraint of the computational environment. Learning rate was initially set to 1e-4 under linearly decaying schedule with warm up. We trained the model for 25 epochs for each task.

4.2 Results

Table 1 compares our model's performance with other transformer-based cross-modal models. While our model's performance falls below that of state-of-the-art models, it is noteworthy that other models explicitly perform additional cross-modal pre-training on top of already pre-trained vision and

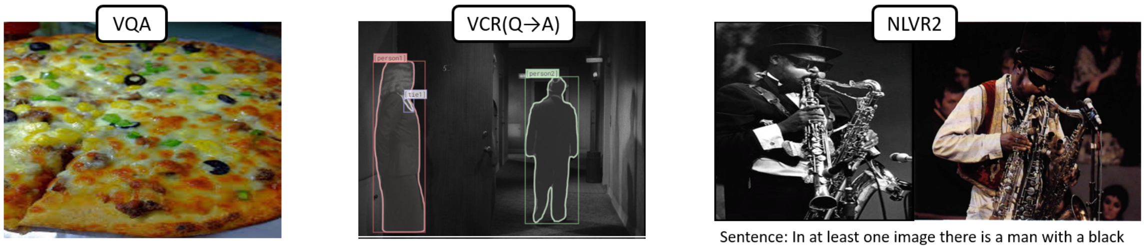

Q: What sort of food is this?
Answer: Pizza

Q: Why is person1 standing behind the open door?
Answer: person1 is trying to hide from person2

Sentence: In at least one image there is a man with a black hat and brown vest playing three saxophones at once.
Answer: True

Figure 2: Qualitative results for each task.

Method	Pre-train #samples	VQA test-dev	VCR QA	VCR QA-R	VCR Q-AR	NLVR2 dev
ViLBERT	3.3M	70.55	73.3	74.6	54.8	–
UNITER	9.5M	73.82	77.3	80.8	62.8	78.4
MiniVLM	7M	69.39	–	–	–	73.7
VisualBERT	0.5M	70.80	71.6	73.2	52.4	67.4
DeVLBERT	3.3M	71.1	–	–	–	–
CAPT	9.2M	72.78	–	–	–	75.1
ERNIE-ViL	4M	73.78	79.2	83.5	66.3	–
Ours	0	67.84	68.4	70.2	49.2	65.2

Table 1: Comparison of our model to other state-of-the-art cross-modal models. 2nd column refers to the number of image-caption pairs seen during cross-modal pre-training. Despite the disadvantage of not having seen a substantial amount of pre-training data, our model closely follows the state-of-the-art models. (CAPT (Luo et al., 2020), ERNIE-ViL (Yu et al., 2020)

Method	VQA test-dev	VCR QA	VCR QA-R	VCR Q-AR	NLVR2 dev
CNN+BERT	65.36	65.4	68.1	48.9	61.9
ViT+LSTM	62.27	62.9	66.4	45.8	60.1
w/o finetuning	56.54	58.2	60.3	42.5	52.4
Ours	67.84	68.4	70.2	49.2	65.2

Table 2: Comparison of our model to different combinations. Under the same condition of no explicit cross-modal fine-tuning and the same embedding scheme, our model outperforms other combinations.

language modules. On the other hand, our model simply concatenates two pre-trained models without additional cross-modal pre-training, and is immediately trained with the target task. For example, ViLBERT and DeVLBERT (Zhang et al., 2020) are pre-traiend with Conceptual Captions dataset (Sharma et al., 2018), and our model is at the disadvantage of not having seen 3.3M pairs of image and captions, yet comes fairly close to those pre-trained models. Fig. 2 shows qualitative examples of the model's performance on each task. In addition, while many papers on pre-trained cross-modal representations do not report specific number of parameters, our approximations of other models' sizes based on the implementation details reported in respective papers suggest that our model is reasonably smaller, especially since it completely eliminates the need for external region detector.

4.3 Further Experiments

In order to examine how much each component contributes to performance, we conduct further experiments, replacing each component with conventional modules. We first replace transformer encoder for images with CNN module, specifically with ResNet-50 (He et al., 2016) trained on ImageNet, using global average-pooled features. We also examine replacing BERT module with LSTM (Hochreiter and Schmidhuber, 1997) using early fusion with image features. For fair comparison, we used concatenation as common embedding scheme for all combinations.

Table 2 shows the results. While ResNet/BERT comes fairly close, it falls below our model, and performance drop is clearer with ViT/LSTM, possibly reflecting superior adaptability of BERT compared to LSTM. Our conjecture is that architectural integrity, *i.e.*, using the same architecture for both vision and language, throughout the model, plays an important role in learning cross-modal representations. Note that, however, it would require a more thorough and analytical study to conclusively claim that ViT is superior to ResNet, or that attention is superior to convolution, and our primary purpose in this experiment is simply to demonstrate that transformer-exclusive models can accomplish comparable performance to the models employing CNN. We also examined linearly training a classifier for target task while fixing the extracted features, without fine-tuning. As expected, there is a significant performance drop, reaffirming the premise that the competence of transformer and BERT is attainable via fine-tuning to its downstream tasks.

Note that, although we performed experiments on a small set of cross-modal tasks, given the superior performance of ViT over ResNet on image classification as reported by (Dosovitskiy et al., 2020), and also on other computer vision tasks as reported by models like pyramid vision transformer (Wang et al., 2021), we believe any task that involves vision and language is a potential beneficiary of transformer-exclusive approach, since it enables the architectural integrity for both modalities.

5 Conclusion

This paper proposed to handle cross-modal tasks for vision and language, solely based on transformer architecture, examining it in various cross-modal tasks. Our paper admittedly does not claim state-of-the-art performances, but to the best of our knowledge, our work is one of the first attempts, along with models like ViLT (Kim et al., 2021) and UniT (Hu and Singh, 2021), to examine cross-modal representation for vision and language solely based on transformer architecture, excluding CNNs and RNNs. Without any structural modifications or sophisticated common embedding scheme, and without additional cross-modal pre-training with millions of samples, our model demonstrates comparable performances to state-of-the-art cross-modal models. Since we deliberately chose the smallest baseline models for each component, and a very simple concatenation scheme, we can intuitively expect an enhanced performance by selecting larger pre-trained models at the cost of more parameters, or by selecting more sophisticated common embedding scheme. The same holds true for the amount of pre-training data used, as we can reasonably expect the performance to boost by using the same amount of pre-training data employed by previous models. With transformer's relative computational efficiency as reported by (Dosovitskiy et al., 2020), the architectural integrity proposed in our model is likely to lead to new research direction, and we hope to encourage more advanced models with novel ideas to follow in near future.

Acknowledgments

We thank Naofumi Akimoto, Akio Hayakawa, Masato Ishii, and Krishna Wadhwani for their help on the manuscript.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Andrei Barbu, Alexander Bridge, Zachary Burchill, Dan Coroian, Sven Dickinson, Sanja Fidler, Aaron Michaux, Sam Mussman, Siddharth Narayanaswamy, Dhaval Salvi, Lara Schmidt, Jiangnan Shangguan, Jeffrey Mark Siskind, Jarrell Waggoner, Song Wang, Jinlian Wei, Yifan Yin, and Zhiqi Zhang. 2012. Video in sentences out.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *ECCV*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

L. Dong, S. Xu, and B. Xu. 2018. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Seattle, Washington, USA. Association for Computational Linguistics.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Computer Vision, ECCV 2010 - 11th European Conference on Computer Vision, Proceedings*, number PART 4 in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 15–29. Springer-Verlag Berlin Heidelberg. Copyright: Copyright 2019 Elsevier B.V., All rights reserved.; 11th European Conference on Computer Vision, ECCV 2010 ; Conference date: 10-09-2010 Through 11-09-2010.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2020. Gaussian error linear units (gelus).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ronghang Hu and Amanpreet Singh. 2021. Unit: Multimodal multitask learning with a unified transformer.

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. 2020. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11336–11344.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. In *Arxiv*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, volume 32, pages 13–23. Curran Associates, Inc.

Fuli Luo, Pengcheng Yang, Shicheng Li, Xuancheng Ren, and Xu Sun. 2020. Capt: Contrastive pre-training for learning denoised sequence representations.

Vicente Ordonez, Girish Kulkarni, and Tamara Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems*, volume 24, pages 1143–1151. Curran Associates, Inc.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 91–99. Curran Associates, Inc.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia. Association for Computational Linguistics.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Jianfeng Wang, Xiaowei Hu, Pengchuan Zhang, Xiujun Li, Lijuan Wang, Lei Zhang, Jianfeng Gao, and Zicheng Liu. 2020a. Minivlm: A smaller and faster vision-language model.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions.

Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, and et al. 2020b. Transformer-based acoustic modeling for hybrid speech recognition. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-vil: Knowledge enhanced vision-language representations through scene graph.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shengyu Zhang, Tan Jiang, Tan Wang, Kun Kuang, Zhou Zhao, Jianke Zhu, Jin Yu, Hongxia Yang, and Fei Wu. 2020. Devlbert. *Proceedings of the 28th ACM International Conference on Multimedia*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27. IEEE Computer Society.

Two Parents, One Child:
Dual Transfer for Low-Resource Neural Machine Translation

Meng Zhang, Liangyou Li, Qun Liu
Huawei Noah's Ark Lab
{zhangmeng92, liliangyou, qun.liu}@huawei.com

Abstract

Neural machine translation suffers when parallel data for training is scarce. Previous works have explored transfer learning to assist training in low-resource scenarios. However, they transfer either from high-resource parallel data, or from monolingual data. In this work, we propose a framework to transfer multiple sources of auxiliary data, including both high-resource parallel data and monolingual data of involved languages. Knowledge in those sources is respectively encoded in a high-resource translation model and pretrained language models, and dually transferred to the low-resource translation model by our approach. Extensive experiments show that our approach yields consistent improvements over strong competitors for multiple translation directions. Furthermore, our approach still exhibits benefit on top of back-translation, making it a useful addition to practitioners' toolbox.

1 Introduction

Neural machine translation (NMT) has achieved remarkable success in recent years, but its quality critically hinges on large-scale parallel data. In the low-resource scenarios for most world languages and many domains, its performance usually deteriorates dramatically.

Although parallel data for some translation tasks may be difficult to obtain, monolingual data is usually within reach, and often comes in much larger quantity. Besides, parallel data for several high-resource languages is readily available. These corpora have been used in various methods to help training low-resource NMT. The most relevant method to our work is transfer learning.

Transfer learning starts with training a source task and then initializes the target task with the parameters. Recent advances in pretrained language models (PLM) like BERT (Devlin et al., 2019) can be seen as transfer learning, where language modeling is the source task for downstream target tasks. In low-resource NMT, pretrained language models have also provided noticeable improvements (Clinchant et al., 2019; Imamura and Sumita, 2019). As another source of transfer, high-resource NMT models have also been used for transfer learning low-resource NMT. Zoph et al. (2016) pioneered this direction with NMT based on recurrent neural networks, and coined the high-resource and low-resource models as parent and child models, respectively.

approach	H		L	
	M	P	M	P
no transfer				✓
(Zoph et al., 2016)		✓		✓
(Kim et al., 2019)		✓	✓	✓
BERT2RND			✓	✓
BERT2BERT			✓	✓
(Kocmi and Bojar, 2018)		✓		✓
BBERT2BBERT			✓	✓
BBERT transfer	✓	✓	✓	✓
dual transfer (ours)	✓	✓	✓	✓

Table 1: An overview of data usage by approaches considered in this work (Section 4.3). H/L: high/low-resource language pair; M: monolingual; P: parallel. BBERT transfer checks all the boxes but uses data in a different way from ours.

However, it is non-trivial to transfer from both PLMs and NMT models. This limitation constrains most existing transfer-learning-based low-resource NMT to a single source of auxiliary data, either monolingual or parallel.

In this paper, we propose a framework for transfer learning low-resource NMT that utilizes both monolingual data and high-resource parallel data (Table 1). Our approach encodes monolingual

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2726–2738
August 1–6, 2021.

knowledge in parent PLMs and translation knowledge in parent NMT models, and transfers both types of models to the child NMT model. Despite its simplicity, our approach shows consistent gains for multiple translation directions. Furthermore, it possesses several desirable features:

- It performs reasonably well even with little or no parallel data in the language pair of interest, alleviating the data issue for low-resource language pairs.
- It is complementary to back-translation, a strong data augmentation approach.
- It is agnostic to network architectures and thus applicable to any translation models.
- It is widely applicable to low-resource languages and can be applied to domain adaptation.
- The same high-resource NMT model can be used to transfer to future low-resource languages, saving computation.

2 Background

2.1 Transfer from Pretrained Language Models

The "pretraining-finetuning" paradigm has been highly successful for various natural language processing tasks. It first pretrains a language model through self-supervised learning, and then finetunes the model along with additional task-specific layers on downstream task data. Here, we exclude pretrained language models trained by sequence-to-sequence learning to simplify discussion[1]. Common pretrained language models include BERT and GPT (Brown et al., 2020).

In NMT with the encoder-decoder architecture (Sutskever et al., 2014; Bahdanau et al., 2015), the direct application of the "pretraining-finetuning" paradigm would be initializing the encoder with PLM and treating the decoder as task-specific layers. However, it is also possible to initialize the compatible modules in the decoder, leaving the cross attention module randomly initialized. Although initializing the decoder does not appear as useful, especially for high-resource language pairs (Rothe et al., 2020), it is not harmful either.

2.2 Transfer from High-Resource Translation Models

Even though the Transformer model (Vaswani et al., 2017) has become more popular than recurrent neural networks for NMT, the transfer procedure proposed by Zoph et al. (2016) still applies as long as the parent model and the child model share the same architecture, which is typically the case. However, one problem still persists. Because the high-resource languages have different vocabularies from the low-resource ones, directly transferring the word embedding layer is not possible.

One way to circumvent this issue is to prepare a joint vocabulary of the involved languages that is shared between the parent and child NMT models (Kocmi and Bojar, 2018). Known as warm-start transfer (Neubig and Hu, 2018), this type of methods need to prepare a new joint vocabulary whenever a new low-resource model is on demand, and retrain both parent and child models. In contrast, cold-start transfer (Kocmi and Bojar, 2020) trains a universal parent NMT model that does not depend on child languages.

Kim et al. (2019) addressed the vocabulary mismatch for cold-start transfer by matching word embeddings across languages. They first learn monolingual word embeddings of the child language with e.g. skip-gram (Mikolov et al., 2013), and then learn a cross-lingual linear mapping to connect child monolingual word embeddings and pretrained parent NMT word embeddings. The child monolingual word embeddings can then be mapped to the parent word embedding space, and be used to initialize the child NMT word embeddings. The cross-lingual linear mapping relies on a bilingual lexicon to learn, which can be induced from parent and child language monolingual data by unsupervised methods like (Lample et al., 2018a).

Our approach also belongs to cold-start transfer in its usage of the parent NMT model. It addresses the vocabulary mismatch by design, without relying on monolingual word embeddings and bilingual lexica.

3 Approach

Our approach is a general framework for transferring from any high-resource language pair to any low-resource language pair, as long as data condition permits. Generally speaking, monolingual data and high-resource parallel data are available in large quantity. We first present the gen-

---

[1] Examples of such models include MASS (Song et al., 2019) and BART (Lewis et al., 2020). If desired, pretrained encoders in these models can be used in our approach.

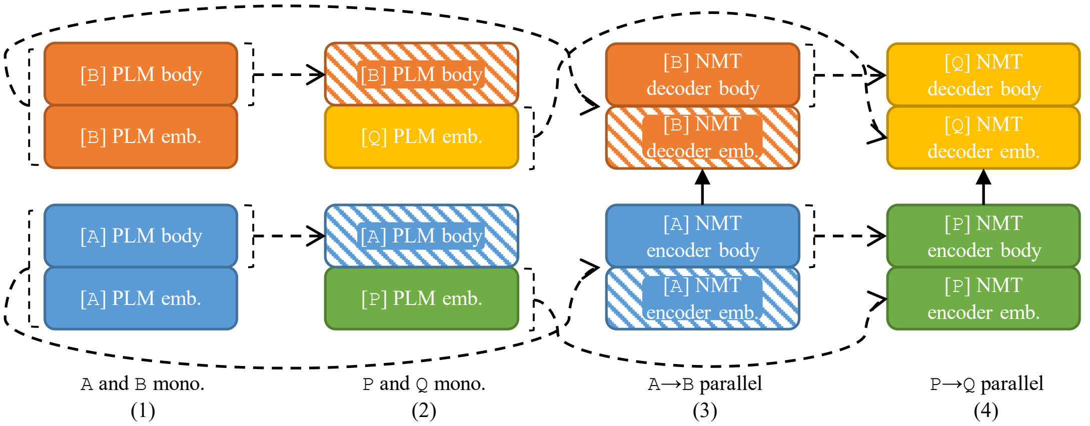

Figure 1: Dual transfer from PLM and high-resource A→B NMT to low-resource P→Q NMT. Dashed lines represent initialization. Parameters in striped blocks are frozen in the corresponding step, while other parameters are trainable. Different colors represent different languages. Data used in each step is also listed.

eral case where we would like to transfer from the high-resource A→B to the low-resource P→Q, where capital letters denote languages. Then we discuss specific cases where some of the involved languages are the same.

3.1 General Transfer

Figure 1 shows the pipeline of our approach, consisting of four major steps, as detailed below.

(1) Train PLM$_{\mathtt{A}}$ and PLM$_{\mathtt{B}}$ on monolingual data of A and B separately.

(2) Train PLM$_{\mathtt{P}}$ and PLM$_{\mathtt{Q}}$ on monolingual data of P and Q as follows.

  - Initialize PLM$_{\mathtt{P}}$ with PLM$_{\mathtt{A}}$ (except word embeddings); freeze parameters other than word embeddings.
  - Initialize PLM$_{\mathtt{Q}}$ with PLM$_{\mathtt{B}}$ (except word embeddings); freeze parameters other than word embeddings.

(3) Train NMT$_{\mathtt{A}\to\mathtt{B}}$ on A→B parallel data as follows: Initialize NMT encoder with PLM$_{\mathtt{A}}$, and decoder with PLM$_{\mathtt{B}}$; freeze word embeddings during training.

(4) Replace word embeddings as follows to initialize NMT$_{\mathtt{P}\to\mathtt{Q}}$, and finetune on P→Q parallel data.

  - Replace NMT$_{\mathtt{A}\to\mathtt{B}}$ encoder word embeddings with those in PLM$_{\mathtt{P}}$.
  - Replace NMT$_{\mathtt{A}\to\mathtt{B}}$ decoder word embeddings with those in PLM$_{\mathtt{Q}}$.

Note that Steps (2) and (3) are independent of each other, and therefore can be done in parallel.

Intuitively, Step (2) learns word embeddings of P and Q that lie in the same semantic space of A and B, respectively. Because only word embeddings are trainable, they are forced to align with pretrained A and B body parameters to do language modeling (e.g. masked language model). In Step (3), NMT$_{\mathtt{A}\to\mathtt{B}}$ needs to learn translation based on the frozen A and B word embeddings space. With P and Q word embeddings swapped in place in Step (4), the body and embedding parameters can cooperate in a close semantic space, allowing finetuning to proceed smoothly.

Like (Kim et al., 2019), our approach solves the vocabulary mismatch issue by manipulation in the embedding space, allowing transfer between arbitrary languages, even with different scripts[2]. Each language now manages its own independent vocabulary. We also tie input and output embeddings of the decoder (Press and Wolf, 2017), so a single decoder embedding block is shown in Figure 1.

We can further generalize our approach by defining *transfer parameters* as those responsible for transforming input into continuous representations shared across languages. In Figure 1, the transfer parameters are simply word embeddings, but we may also use other sets of transfer parameters, e.g. word and position embeddings, or even lower layers of the body. In Step (2), only transfer parameters are trainable, while in Step (3), only non-transfer parameters are trainable, and initialization changes accordingly.

---

[2] We verified the effectiveness of our approach for transferring from `fr→en` to `ru→en` on in-house data.

Our approach defines a framework for transfer learning, which can be applied to various network architectures. For example, if we would like to train a low-resource RNN-based NMT, we can prepare RNN-based PLMs and a high-resource RNN-based NMT. In our experiments, we use Transformer for PLMs and NMT models.

3.2 Shared Target Transfer and Shared Source Transfer

In practice, it is a rare need to train on a low-resource language pair where both languages are low-resource. Typically one of the two languages would be high-resource, e.g. English. In this case, we can choose a high-resource language pair that shares this language on the same side, thereby simplifying our approach.

If the target language (`Q`) of the low-resource language pair (`P`→`Q`) is high-resource, we can choose a high-resource language pair (`A`→`B`) with that language as the target, i.e. `B`=`Q`. In this case, there is no vocabulary mismatch on the target side, so $\text{PLM}_{\texttt{Q}}$ is no longer needed, and decoder word embeddings can be adjusted when training $\text{NMT}_{\texttt{A}\rightarrow\texttt{B}}$ in Step (3). $\text{PLM}_{\texttt{B}}$ also becomes optional, and the randomly initialized decoder of $\text{NMT}_{\texttt{A}\rightarrow\texttt{B}}$ may learn sufficiently from abundant `A`→`B` parallel data.

Likewise, if the source language (`P`) is high-resource, we can let `A`=`P`. Then $\text{PLM}_{\texttt{P}}$ is not needed, and encoder word embeddings are trainable in Step (3). $\text{PLM}_{\texttt{A}}$ may also be dispensed with and the encoder of $\text{NMT}_{\texttt{A}\rightarrow\texttt{B}}$ is randomly initialized.

3.3 Domain Adaptation

By viewing a certain domain as a special language, our approach can also be applied to domain adaptation. In this case, `A`→`B` is a high-resource source domain, and `P`→`Q` is a low-resource target domain. By definition, this setting is general transfer, because neither `B`=`Q` nor `A`=`P` is possible due to domain difference, but typically they will be the same language, respectively.

4 Experimental Setup

We mainly verify our approach in the more realistic shared target and shared source transfer scenarios. We take German-English (`de-en`) as the high-resource language pair, while Estonian-English (`et-en`) and Turkish-English (`tr-en`) are the low-resource language pairs. Previous works mainly consider shared target transfer (Dabre et al., 2020), and we make extensive comparison in the experiment that transfers from `de`→`en` to `et`→`en`. We then verify on other translation directions, including shared source transfer, as well as general transfer, in which we consider an artificial setting of transferring from `de`→`en` to French→Spanish (`fr`→`es`). For domain adaptation we work on `de`→`en`, transferring from news domain to medical domain. We report SacreBLEU[3] (Post, 2018). Further details about data and hyperparameters can be found in Appendices B and C, respectively.

language code	# sentence (pair)
`de-en`	5.9m
`et-en`	1.9m
`tr-en`	207k
`fr-es`	10k
`de-en` medical	347k
`en`	94m
`de`	147m
`et`	139m
`tr`	100m
`fr`	4.1m
`es`	4.2m
`en` medical	4.0m
`de` medical	3.6m

Table 2: Training data statistics.

4.1 Data

We mainly use data from WMT 2018[4]. We use preprocessed parallel data for training NMT models. The provided development data includes multi-parallel data for several languages, which we use for `fr`→`es`. We collect monolingual data for the involved languages and follow the same preprocessing pipeline. Training data statistics is provided in Table 2. Each language is encoded with byte pair encoding (BPE) (Sennrich et al., 2016b). The BPE codes and vocabularies are learned on each language's monolingual data, and then used to segment parallel data. Following (Kim et al., 2019), we use 50k merge operations for English, and 20k for other languages. Sentences with more than 150 subwords are removed from NMT training.

---

[3] SacreBLEU signature: BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.4.12.

[4] `http://statmt.org/wmt18/translation-task.html`

4.2 Hyperparameters

We use Transformer base as our NMT model, but with slight modifications that follow the implementation of BERT[5]. The absolute position embeddings are also learned as in BERT. We apply dropout with probability 0.1. Learning rate warms up for 16,000 steps and then follows inverse square root decay. The peak learning rate is 7×10^{-4} for the high-resource `de-en`. For other translation tasks, we grid search over $\{1, 3, 5\} \times 10^{-4}$ for each approach in every experiment, and keep the best model based on development BLEU. We use 8 GPUs for `de-en`, and 1 GPU otherwise. Other hyperparameters follow (Kim et al., 2019).

We train BERT as the PLM in our experiments, with the same number of layers and hidden size as Transformer base. The absolute position embeddings are learned up to 128. We only train with masked language modeling and dispense with next sentence prediction as in (Liu et al., 2019). We train for 480k steps with batch size 180 on 8 GPUs. The peak learning rate is 1.8×10^{-4}, and the number of warmup steps is 18,000.

Rothe et al. (2020) found that for the high-resource `de-en` pair, initializing the decoder with PLM has no advantage over random initialization. Therefore, we only used $\text{PLM}_{\texttt{de}}$ for `de`→`en`, but for `en`→`de`, we used both $\text{PLM}_{\texttt{en}}$ and $\text{PLM}_{\texttt{de}}$ because the vocabulary mismatch is on the target side.

4.3 Baselines

We compare with the following approaches.

No transfer This baseline trains directly on the low-resource parallel data.

(Zoph et al., 2016) This approach transfers from the high-resource language pair. In the original paper, random parent word embeddings are used to initialize child word embeddings. We simply initialize child word embeddings with the truncated normal initializer.

(Kim et al., 2019) This approach transfers from the high-resource language pair and utilizes cross-lingual word embeddings. The authors also proposed other orthogonal data augmentation techniques, but we do not include them in our experiments.

BERT2RND This approach transfers from the source language PLM trained on monolingual data. By comparing with BERT2BERT, we can see if the finding in (Rothe et al., 2020) holds for low-resource language pairs.

BERT2BERT This approach transfers from the source and target language PLMs trained on monolingual data. Note that PLMs for BERT2BERT and BERT2RND are directly trained on monolingual data of `P` and `Q`, different from those obtained by Step (2) of our approach.

As discussed in (Kim et al., 2019), managing independent vocabularies for each language has the advantage of flexibility. However, many approaches rely on shared vocabulary. We nevertheless report their performance for reference.

(Kocmi and Bojar, 2018) This approach uses joint vocabulary of all the involved languages. It first trains the NMT model on the high-resource parallel data, and then finetunes it on the low-resource parallel data. It can be seen as a multilingual NMT in which high-resource performance does not matter. We experiment with transferring from `de`→`en` to `et`→`en`, thus involving three languages. We learn joint BPE with 90k merge operations.

BBERT2BBERT Multilingual PLMs usually rely on shared vocabulary, and bilingual BERT (BBERT) is an example trained on non-parallel data of two languages. We learn joint BPE with 70k merge operations for the source and target languages of the low-resource language pair, and the same vocabulary is used for the source and target sides of NMT. Otherwise this approach is the same as BERT2BERT. This is equivalent to XLM (Conneau and Lample, 2019) used for bilingual PLM and MT.

BBERT transfer Multilingual PLMs are often used for cross-lingual transfer. We apply this approach to NMT, transferring from `de`→`en` to `et`→`en`. First we train a `de-et` BBERT, and use it to initialize the encoder of $\text{NMT}_{\texttt{de}\rightarrow\texttt{en}}$. Then we train on `de`→`en` and finally finetune on `et`→`en`. We learn joint BPE with 40k merge operations for the `de-et` pair. This approach uses exactly the same data as ours because we do not use

---

[5] `https://github.com/google-research/bert`

approach	V	BLEU
no transfer	✓	21.76
(Zoph et al., 2016)	✓	21.07
(Kim et al., 2019)	✓	22.25
BERT2RND	✓	22.89
BERT2BERT	✓	23.44
(Kocmi and Bojar, 2018)	✗	23.58
BBERT2BBERT	✗	23.90
BBERT transfer	✗	24.03
dual transfer (word)	✓	**24.81**
dual transfer (word+position)	✓	24.28

Table 3: BLEU on `et→en`, with the best in bold. "✓" in the "V" column indicates independent vocabulary, while "✗" means the approach relies on shared vocabulary. Our approach (dual transfer) has two variants, with or without position embeddings in the transfer parameters.

PLM$_{\mathrm{en}}$ when transferring from `de→en` to `et→en`.

In their experiments, Zoph et al. (2016) and Kim et al. (2019) only considered shared target transfer, and they found that freezing certain components of the decoder during finetuning can be beneficial. In our `et→en` experiment, we tried freezing the decoder word and position embeddings, and optionally self attention parameters, for their approaches, our approach, and BERT2BERT, but development set results revealed that the only setting which brought improvement was freezing word and position embeddings and self attention parameters for (Kim et al., 2019), possibly due to the relatively large size of `et→en` data. Therefore we only use it for (Kim et al., 2019) in our experiments.

5 Results

In this section, we first report extensive experiments on `et→en` before generalizing to other translation directions. We then present the performance of our approach when used in conjunction with back-translation and self training. Finally we demonstrate that our approach can be used for domain adaptation.

5.1 Results on `et→en`

Table 3 shows the BLEU scores for `et→en`. We report the following findings for this translation direction.

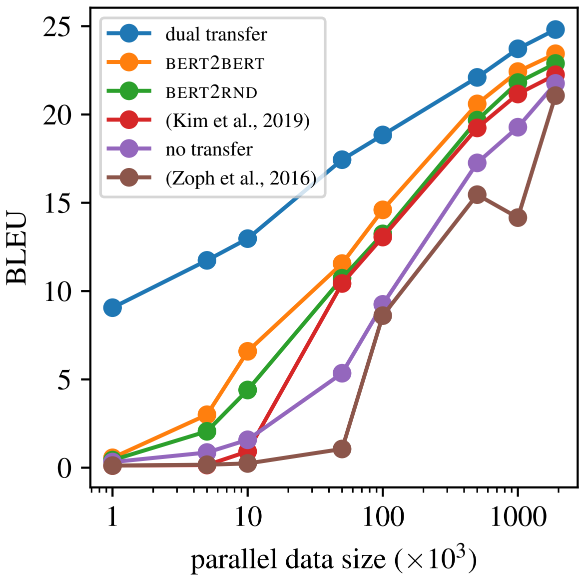

Figure 2: BLEU of different approaches with respect to the number of parallel `et→en` sentence pairs for training. We plot our approach with word embeddings as transfer parameters; additionally transferring position parameters performs similarly.

The approach in (Zoph et al., 2016) only uses high-resource parallel data for transfer, and the approach in (Kim et al., 2019) additionally uses low-resource monolingual data; their BLEU scores are close to the "no transfer" baseline. The approach in (Kocmi and Bojar, 2018) shows positive transfer from high-resource parallel data by forgoing the vocabulary flexibility and relying on joint vocabulary.

Using monolingual data, BERT2RND and BERT2BERT show notable improvement on the "no transfer" baseline. In this relatively low-resource setting, it appears useful to initialize the decoder with BERT, in contrast to `de-en` experiments in (Rothe et al., 2020).

We expected additionally transferring position embeddings to better deal with word order divergence across languages, but after comparing the two variants of our approach, we find no benefit in including position embeddings in the transfer parameters. Our approach with word embeddings as transfer parameters achieves best BLEU, which is a 3.05 improvement over the "no transfer" baseline, and 1.37 over BERT2BERT. Note that we did not use monolingual English data for our approach when the target language is English.

parallel data size ($\times 10^3$)	0	1	5	10	50	100	500	1000
dual transfer (word)	0.43	9.06	11.74	12.97	17.44	18.84	22.10	23.72
+freezing parent NMT encoder	**6.20**	8.82	11.58	12.76	16.62	18.50	21.69	23.59

Table 4: BLEU on `et`→`en`. Freezing the parent NMT encoder helps our approach to perform zero-shot translation.

approach	`tr`→`en`	`en`→`et`	`en`→`tr`	`fr`→`es`
no transfer	15.44	16.29	9.63	10.59
BERT2BERT	19.73	17.36	11.78	18.26
dual transfer (word)	**21.12**	**19.41**	**13.18**	**22.28**
dual transfer (word+position)	20.29	18.79	13.16	-

Table 5: BLEU on translation directions shown in columns, grouped by shared target transfer, shared source transfer, and general transfer. "-" means the experiment was not carried out.

approach	BLEU
no transfer	21.63 (-0.13)
dual transfer (word)	22.53 (-2.28)
dual transfer (word+position)	23.08 (-1.20)

Table 6: BLEU on `et`→`en` augmented with 4m self training data. Numbers in parentheses indicate differences from the corresponding approach trained on authentic parallel data.

5.2 Effect of Low-Resource Parallel Data Size

Arguably, the parallel training data for `et`→`en` is not quite low-resource. But it provides a good test bed for manually adjusting the data size to simulate various degrees of resource scarcity. We sample subsets of $\{1, 5, 10, 50, 100, 500, 1000\} \times 10^3$ parallel sentence pairs, and show BLEU of different approaches in Figure 2. We observe roughly monotonic trend of BLEU with respect to parallel data size, as expected. Our approach performs consistently better than baselines, and the gap is larger with fewer parallel sentence pairs. In the extremely low-resource setting of one thousand pairs, our approach still achieves BLEU close to 10, while all other approaches fail with BLEU close to 0.

5.3 Zero-Shot Translation

Our approach can also be modified slightly to perform zero-shot translation. We conjecture that in Step (3) of our approach, freezing the embeddings alone is insufficient to prevent encoder body parameters from drifting too far away. Therefore we try freezing the entire encoder in Step (3). This technique helps our approach to achieve a zero-shot BLEU score of 6.20, as shown in Table 4. However, it does not have advantage when parallel data is available.

5.4 Other Translation Directions

Table 5 shows the results that include shared target transfer, shared source transfer, and general transfer, comparing our approach with no transfer and BERT2BERT. Our approach consistently outperforms baselines. Previous works (Zoph et al., 2016; Kim et al., 2019) typically conducted experiments on shared target transfer only, and shared source transfer is considered more difficult (Kocmi, 2020), but our approach works well for shared source transfer, as well as general transfer. Also note that, we use the same `de-en` pair for all child languages from diverse language families, which demonstrates the robustness of our approach. It also highlights the advantage of independent vocabularies: We can prepare $\text{NMT}_{\texttt{de}\to\texttt{en}}$ and $\text{NMT}_{\texttt{en}\to\texttt{de}}$ for any future child language, while approaches like (Kocmi and Bojar, 2018) and BBERT transfer have to retrain with the high-resource language every time a new low-resource language is needed.

5.5 Back-Translation and Self Training

Back-translation (BT) (Sennrich et al., 2016a) and self training (ST) (Zhang and Zong, 2016) are data augmentation techniques that generate synthetic parallel data, using target language monolingual data and source language monolingual data respectively. We first experiment with ST for `et`→`en`. We use the "no transfer" $\text{NMT}_{\texttt{et}\to\texttt{en}}$ to translate 4m `et` monolingual data into `en` by greedy decoding, and merge with authentic parallel data. Results in Table 6 show that self training is not helpful for this experiment, and considerably lowers the BLEU of our approach.

approach	BT data size	BLEU
no transfer	4m	19.78 (+3.49)
dual transfer (word)	4m	21.74 (+2.33)
dual transfer (word+position)	4m	**22.34** (+3.55)
no transfer	130m	20.52 (+4.23)
dual transfer (word+position)	130m	22.23 (+3.44)

Table 7: BLEU on `en`→`et` augmented with 4m or 130m back-translation data. Numbers in parentheses indicate differences from the corresponding approach trained on authentic parallel data.

approach	BLEU
no transfer (child)	62.94
BERT2BERT (child)	64.33
finetuning (parent)	64.91
dual transfer (parent)	65.14
dual transfer (child)	**65.40**

Table 8: Domain adaptation results. The transfer parameters are word embeddings for dual transfer. "Parent" indicates using source domain (news) vocabulary, and "child" indicates using target domain (medical) vocabulary.

We then use the same synthetic parallel data for `en`→`et`, turning to the case of BT. The upper rows in Table 7 show that BT is highly beneficial for both the baseline and our approach. Encouraged by this, we further try using all 130m `et` monolingual data with the maximum of 80 tokens and 100 subwords per line. We upsample authentic data to have a 1:4 ratio with synthetic data, following (Caswell et al., 2019). The lower rows in Table 7 show that more BT data can further improve the "no transfer" baseline, though the small improvement appears unattractive considering the cost. As for our approach, going from 4m to 130m yields no gain. Besides, our approach with 4m BT still surpasses no transfer with 130m BT. We conjecture that our approach can work complementarily with a manageable amount of BT data, reducing the need to decode and train on a huge data size.

Finally, note that we use the "no transfer" $\text{NMT}_{\texttt{et}\rightarrow\texttt{en}}$ to generate all synthetic parallel data in our experiments. In practice, the model produced by our approach can be used for decoding, which should result in higher-quality synthetic data. This might also be the reason that ST hurts our approach more than the "no transfer" baseline.

5.6 Domain Adaptation

A simple and effective approach to domain adaptation is finetuning source domain NMT on target domain data (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016). This approach is possible because directly inheriting parent NMT vocabulary is acceptable for domain adaptation. In other words, this is a special case of (Kocmi and Bojar, 2018) where child vocabulary largely overlaps with parent vocabulary. However, our approach allows using a dedicated vocabulary for the target domain. In this case, we learn BPE with the same number of merge operations as the source domain on target domain monolingual data. Table 8 shows that our approach can surpass the baselines, especially with the child (medical domain) vocabulary.

6 Related Work

Low-resource NMT has been researched from many perspectives. Exploiting auxiliary data has been verified to be helpful by various approaches, including data augmentation like back-translation (Sennrich et al., 2016a; Xia et al., 2019), transfer learning as focused in our work, meta-learning (Gu et al., 2018), semi-supervised learning (Cheng et al., 2016), or even unsupervised NMT (Artetxe et al., 2018; Lample et al., 2018b; Chronopoulou et al., 2020).

Transfer learning usually utilizes a single source of knowledge. When multiple sources are available, transfer learning may be applied in a cascaded fashion (Lakew et al., 2018), but catastrophic forgetting may need to be addressed. Maimaiti et al. (2019) proposed multi-round transfer by performing transfer learning for several rounds on multiple high-resource language pairs.

Multilingual NMT (Johnson et al., 2017; Dabre et al., 2019) aims to perform translation for multiple translation pairs in a single model, and positive transfer towards low-resource language pairs typically occurs. In our experiment, we have considered a variant that solely focuses on the low-resource pair (Kocmi and Bojar, 2018; Nguyen and Chiang, 2017).

Outside NMT, Artetxe et al. (2020) proposed a similar partial freezing approach to transferring BERT cross-lingually. As they worked on BERT (Transformer encoder) for natural language understanding tasks, several differences from our work arise. First, we need to consider the initialization of decoder for NMT, and for the shared source case, we need to deal with vocabulary mismatch on the decoder side. Second, we find that additionally transferring position embeddings is not helpful in our experiments. Third, our approach can outperform BBERT transfer, whereas they observe slightly lower performance in their experiments.

7 Conclusion and Future Work

In this work, we propose a framework for transferring from both pretrained language models and neural machine translation models, so that both monolingual data and high-resource parallel data can be used to assist low-resource training. Our approach shows consistent usefulness in a variety of experiments, while also enjoying the flexibility of independent vocabulary.

Recently, a deep encoder and shallow decoder architecture is shown to have comparable translation quality with faster decoding speed (Kasai et al., 2020). While our approach can be applied to such architectures, a shallow decoder means that transferring on the decoder side will be limited by the shallow PLM, which is particularly severe for shared source transfer. In future work we would like to investigate how to work around this issue.

Acknowledgments

We thank Wenyong Huang for his help with the BERT code. We also thank anonymous reviewers for their helpful feedback. We thank MindSpore[6] for the partial support of this work, which is a new deep learning computing framework.

Ethical Considerations

In this paper we have reported experiments with Transformer base, and the computation cost is shown in Table 11 of Appendix A. Scaling up the model may bring further improvement at the cost of computation. However, we have highlighted the benefit of cold-start transfer: Trained high-resource NMT models can be used for future transfer. For example, we can reuse $\text{NMT}_{\texttt{de}\rightarrow\texttt{en}}$ and $\text{NMT}_{\texttt{en}\rightarrow\texttt{de}}$ for a future low-resource language `X` translating to and from English, and $\text{PLM}_{\texttt{X}}$ can be used for both directions if the encoder and the decoder have the same number of layers. We hope such reuse can amortize the cost of preparing parent models. We release the code to facilitate future transfer at `https://github.com/huawei-noah/noah-research/tree/master/noahnmt/dual-transfer`. Besides, our experiment indicates that our approach can reduce the need of back-translation data size, while producing back-translation data and training on augmented data are both costly.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised Neural Machine Translation. In *International Conference on Learning Representations*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the Cross-lingual Transferability of Monolingual Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged Back-Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-Supervised

---

[6] `https://www.mindspore.cn`

Learning for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany. Association for Computational Linguistics.

Alexandra Chronopoulou, Dario Stojanovski, and Alexander Fraser. 2020. Reusing a Pretrained Language Model on Languages with Limited Corpora for Unsupervised NMT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2703–2711, Online. Association for Computational Linguistics.

Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. 2019. On the use of BERT for Neural Machine Translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 108–117, Hong Kong. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *Advances in Neural Information Processing Systems 32*, pages 7059–7069. Curran Associates, Inc.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A Comprehensive Survey of Multilingual Neural Machine Translation. *arXiv:2001.01115 [cs]*. ArXiv: 2001.01115.

Raj Dabre, Atsushi Fujita, and Chenhui Chu. 2019. Exploiting Multilingualism through Multistage Fine-Tuning for Low-Resource Neural Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1410–1416, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast Domain Adaptation for Neural Machine Translation. *arXiv:1612.06897 [cs]*. ArXiv: 1612.06897.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-Learning for Low-Resource Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Kenji Imamura and Eiichiro Sumita. 2019. Recycling a Pre-trained BERT Encoder for Neural Machine Translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 23–31, Hong Kong. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2020. Deep Encoder, Shallow Decoder: Reevaluating Non-autoregressive Machine Translation. In *International Conference on Learning Representations*.

Yunsu Kim, Yingbo Gao, and Hermann Ney. 2019. Effective Cross-lingual Transfer of Neural Machine Translation Models without Shared Vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1246–1257, Florence, Italy. Association for Computational Linguistics.

Tom Kocmi. 2020. Exploring Benefits of Transfer Learning in Neural Machine Translation. *arXiv:2001.01622 [cs]*. ArXiv: 2001.01622.

Tom Kocmi and Ondřej Bojar. 2018. Trivial Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Brussels, Belgium. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2020. Efficiently Reusing Old Models Across Languages via Transfer Learning. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 19–28, Lisboa, Portugal. European Association for Machine Translation.

Surafel M. Lakew, Aliia Erofeeva, Matteo Negri, Marcello Federico, and Marco Turchi. 2018. Transfer Learning in Multilingual Neural Machine Translation with Dynamic Vocabulary. In *IWSLT*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-Based & Neural Unsupervised Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In *arXiv:1907.11692 [cs]*. ArXiv: 1907.11692.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *The 12th International Workshop on Spoken Language Translation*.

Mieradilijiang Maimaiti, Yang Liu, Huanbo Luan, and Maosong Sun. 2019. Multi-Round Transfer Learning for Low-Resource NMT Using Multiple High-Resource Languages. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 18(4):38:1–38:26.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119.

Graham Neubig and Junjie Hu. 2018. Rapid Adaptation of Neural Machine Translation to New Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Toan Q. Nguyen and David Chiang. 2017. Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics*, 8(0):264–280.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. In *International Conference on Machine Learning*, pages 5926–5936. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized Data Augmentation for Low-Resource Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796, Florence, Italy. Association for Computational Linguistics.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting Source-side Monolingual Data in Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A Runtime

We report the runtime of each step in our dual transfer (word) for NMT$_{\texttt{et}\rightarrow\texttt{en}}$ in Table 11. PLMs for other languages take similar time because we run a fixed number of steps. The runtime of the last finetuning step varies depending on low-resource parallel data size and learning rate. We also report the runtime of the "no transfer" baseline for this language pair as reference.

language code	train	dev	test
`de-en`	preprocessed - ParaCrawl	newstest2017	-
`et-en`	preprocessed	newsdev2018	newstest2018
`tr-en`	preprocessed	newsdev2016	newstest2018
`fr-es`	newstest2008-2011	newstest2012	newstest2013
`de-en` medical	EMEA - dev - test	random 3k of EMEA	random 3k of EMEA

Table 9: Parallel data source. "Preprocessed" means the preprocessed data provided in WMT 2018 news translation task.

language code	data source
`en`	News Crawl 2014-2017
`de`	News Crawl 2014-2017
`et`	News Crawl 2014-2017, BigEst Estonian corpus, Common Crawl
`tr`	News Crawl 2016-2017, Common Crawl
`fr/es`	News Crawl 2012
`en` medical	EMEA, PatTR, Wikipedia articles, AACT
`de` medical	EMEA, PatTR, Wikipedia articles

Table 10: Monolingual data source.

	# GPU	runtime (hours)
$\text{PLM}_{\texttt{de}}$	8	32
$\text{PLM}_{\texttt{et}}$	8	33
$\text{NMT}_{\texttt{de}\rightarrow\texttt{en}}$	8	39
$\text{NMT}_{\texttt{et}\rightarrow\texttt{en}}$	1	8
no transfer	1	20

Table 11: Runtime of each step in dual transfer (word) for $\text{NMT}_{\texttt{et}\rightarrow\texttt{en}}$. The runtime of the "no transfer" baseline for this language pair is also listed.

B Data Source and Preprocessing

We list the data source in Tables 9 and 10. Most of the data is from WMT 2018, unless otherwise noted. Medical data is from WMT 2014 medical translation task[7]. The French and Spanish monolingual data is from WMT 2013 news translation task[8].

All data sets are deduplicated. The Turkish monolingual data is further cleaned by removing lines with more than half non-Turkish characters, and we only use a subset with 100m lines.

C Hyperparameters and Development Performance

As we grid search learning rates in $\{1,3,5\} \times 10^{-4}$, we report the best found learning rate and the corresponding development BLEU in Tables 12, 13, and 14. The development BLEU is calculated by tokenized `multi-bleu.perl`. Due to the large scale of the 130m BT experiment, we directly use the best learning rates for 4m BT, and set other hyperparameters as in high-resource NMT.

---

[7] http://statmt.org/wmt14/medical-task/

[8] http://statmt.org/wmt13/translation-task.html

approach	`et→en`		`tr→en`		`en→et`		`en→tr`		`fr→es`	
	lr	BLEU	lr	BLEU	lr	BLEU	lr	BLEU	lr	BLEU
no transfer	5	22.37	5	17.56	3	15.32	5	14.03	3	11.62
(Zoph et al., 2016)	5	21.67	-	-	-	-	-	-	-	-
(Kim et al., 2019)	3	23.21	-	-	-	-	-	-	-	-
BERT2RND	3	22.84	-	-	-	-	-	-	-	-
BERT2BERT	3	23.98	1	22.06	1	16.44	1	16.27	3	21.57
(Kocmi and Bojar, 2018)	5	24.42	-	-	-	-	-	-	-	-
BBERT2BBERT	1	24.52	-	-	-	-	-	-	-	-
BBERT transfer	1	25.05	-	-	-	-	-	-	-	-
dual transfer (word)	1	25.33	1	23.34	3	18.31	1	17.84	1	26.06
dual transfer (word+position)	1	25.20	1	22.33	3	17.91	1	17.88	-	-

Table 12: Best found learning rate ($\times 10^{-4}$) and the corresponding development BLEU for various translation directions.

approach	4m ST		4m BT		130m BT	
	lr	BLEU	lr	BLEU	lr	BLEU
no transfer	3	22.48	3	19.45	3	19.97
dual transfer (word)	1	23.36	1	21.67	-	-
dual transfer (word+position)	1	23.67	1	21.73	1	21.27

Table 13: Best found learning rate ($\times 10^{-4}$) and the corresponding development BLEU for `et→en` ST and `en→et` BT experiments.

approach	lr	BLEU
no transfer (child)	3	63.39
BERT2BERT (child)	3	64.84
finetuning (parent)	1	65.26
dual transfer (parent)	3	65.13
dual transfer (child)	3	65.41

Table 14: Best found learning rate ($\times 10^{-4}$) and the corresponding development BLEU for domain adaptation on `de→en`.

Contrastive Aligned Joint Learning for Multilingual Summarization

Danqing Wang, Jiaze Chen, Hao Zhou, Xipeng Qiu †, Lei Li
ByteDance AI Lab
{wangdanqing.122,chenjiaze,zhouhao.nlp,lileilab}@bytedance.com
† Fudan University
xpqiu@fudan.edu.cn

Abstract

Multilingual text summarization requires the ability to understand documents in multiple languages and generate summaries in the corresponding language, which poses more challenges on current summarization systems. However, this problem has been rarely studied due to the lack of large-scale supervised summarization data in multiple languages. In this paper, we first provide a large-scale multilingual summarization corpus MLGSum consisting of 1.1 million articles and summaries in 12 different languages. Based on it, we develop a unified summarization model to understand the document and generate summaries in different languages. We use the contrastive learning strategy to train our multilingual summarization system (CALMS), which consists of two training objectives, contrastive sentence ranking (CSR) and sentence aligned substitution (SAS). The two training objectives are designed to share salient information extractive ability and align sentence-level representation across different languages. Experimental results indicate that CALMS achieves significant improvement over monolingual models in all languages. We further transfer CALMS to other languages and find that it will also benefit similar languages. Our code and dataset are available at https://github.com/brxx122/CALMS.

1 Introduction

Automatic text summarization aims at providing a brief summary for a long document. It requires the ability to understand document-level input, catch the main idea of it, and generate a fluent text. Recently, monolingual summarization has witnessed great success with the development of new neural systems (Zhong et al., 2020; Wang et al., 2020) and the availability of monolingual pre-training language models (Kenton and Toutanova, 2019; Liu and Lapata, 2019; Liu et al., 2019; Lewis et al., 2020b). , Inspired by the success of monolingual pre-trained models, researchers further pre-train these models with multiple languages to get the multilingual versions (Huang et al., 2019; Liu et al., 2020; Lewis et al., 2020a), which provide the abilities of understanding and generation in different languages. The multilingual pre-training model can be used as the initialization and finetuned for downstream summarization tasks.

However, the pre-training phase for language models usually focuses on predicting masked tokens or denoising the noisy input, both of which are token-level tasks. It lacks the ability to align sentence-level information among languages and to distinguish which information is the most critical for the document-level input. Most previous multilingual summarization models focus on training one model for different language or partly share encoder/decoder layers (Wang et al., 2018; Lin et al., 2018; Scialom et al., 2020). Cao et al. (2020) and Lewis et al. (2020a) try to train one model for all languages, but they find that although low-resource languages can benefit from the larger training data, the performance of rich-resource languages has been sacrificed. Thus, we want to investigate the following question: *Can we design a unified multilingual summarization model that can benefit both high-resource and low-resource languages?*

In this paper, we design a neural model with the contrastive aligned joint learning strategy for multilingual summarization (CALMS) with two new training objectives: contrastive sentence ranking (CSR) and sentence aligned substitution (SAS). CSR samples sentences from the document and constructs positive and negative pairs based on their saliency. By contrastively learning what is more important, the model is supposed to obtain the ability to distinguish salient information from the document. In order to align sentence-level information among languages, SAS replaces sentences with an-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2739–2750
August 1–6, 2021.

other language and generates the summary based on the noisy input.

We conduct the experiments in five languages: English, Chinese, German, French, and Russian. The experimental results show that CALMS outperforms the monolingual baseline significantly. Further promotion will be gained by finetuning on the specific language. We also transfer our model to 7 languages (Hindi, Spanish, Indonesian, Turkish, Vietnamese, Ukrainian, Portuguese) and achieve great improvements, which indicates our model obtains a better initialization for summarization and can be a better solution for low-resource summarization. We additionally propose a new large-scale multilingual summarization dataset with 12 languages for future multilingual summarization research.

We highlight our contributions as follows:

(1) We design a neural model with the contrastive aligned learning strategy for multilingual summarization (CALMS), which improves summarization performance in both rich-resource and low-resource languages.

(2) We propose two new training strategies to distinguish important information from the document and align sentence-level information across languages.

(3) In order to investigate multilingual summarization, we create a 1.1 million multilingual summarization dataset MLGSum with 12 languages. The experimental results on 5 main languages show that our model significantly outperforms the monolingual summarization model. The extensive experiments on 7 other languages indicate our model can transfer to other similar languages with a good performance.

2 Related Work

Multilingual Summarization Abstractive summarization aims at generating a shorter version of the document while maintaining the most important information. With the large success brought by pre-trained language models in English abstractive summarization (Liu and Lapata, 2019; Lewis et al., 2020b; Zhang et al., 2020), several works focus on summarization in multiple languages. Nguyen and Daumé III (2019) constructs a small cross-lingual dataset with English summaries for non-English articles, and Scialom et al. (2020) proposes MLSUM with 5 languages as the extended version of English summarization dataset CNN/DailyMail (Hermann et al., 2015). Cao et al. (2020) use a Transformer-based model with 6 layers encoder and decoder to combine auto-encoder training, translation and summarization. Different from Cao et al. (2020), we focus on document-level multilingual summarization, which means understanding of long input in different languages is more important for our model. Besides, we propose a large-scale multilingual dataset with 12 languages and each document-summary pair is in the same language.

Contrastive learning in Summarization The goal of contrastive training is to let the model distinguish specific features by constructing positive and negative pairs. For summarization, it is often used to find a better summary. (Shi et al., 2019) randomly replaces a sentence in the ground-truth summary with a random sentence to form the negative sample. Wu et al. (2020) constructs negative samples on different aspects of summary qualities and propose a new summary evaluation method by contrastive learning. Zhong et al. (2020) use a pre-trained extractive model to select several candidates as negative samples and take the ground-truth as the positive. In this work, we dynamically sample several sentences from the document during the training phase and construct the positive and negative pair based on their similarity with the ground-truth summary.

Multilingual Pre-training for Generation Several works try to expand the successful unsupervised pre-training English language model to multiple languages for multilingual understanding and generation (Lample and Conneau, 2019; Huang et al., 2019; Liu et al., 2020; Xue et al., 2020). mBART (Liu et al., 2020) denoises full texts in multiple languages and pre-trains the complete encoder-decoder model, which works well on both sentence-level and document-level machine translation. mT5 (Xue et al., 2020) is the multilingual version of T5 (Kale and Rastogi, 2020) for text-to-text. MARGE (Lewis et al., 2020a) is trained with the multi-lingual multi-document paraphrasing objective, which reconstructs text in one language by retrieving a set of related texts in other languages.

3 Method

Given a document $D = \{x_1, x_2, \cdots, x_M\}$ with M words, the goal of abstractive summarization is to generate a summary with N words $Y = \{y_1, y_2, \cdots, y_N\}$, where $M > N$. For multilin-

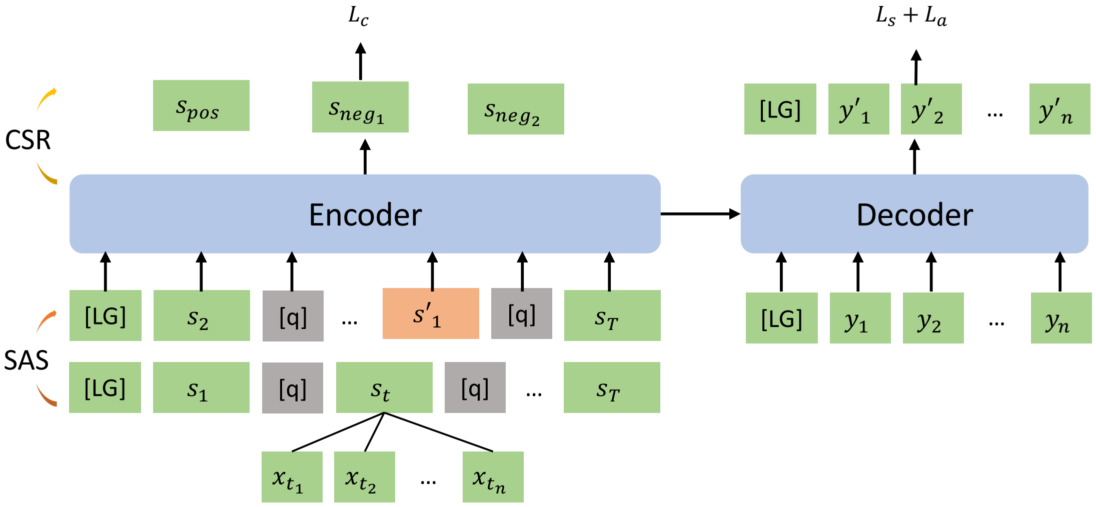

Figure 1: Model Overview. The input and output in green are the same language while the orange is another language. The input document includes T sentences separated by the delimiter '[q]', and each sentence s_t consist of t_n token x. The language indicator '[LG]' is added at the beginning of the encoder and the decoder. CSR selects $q = 3$ sentences from the document and constructs the contrastive pair with one positive and two negative examples. SAS replaces sentence s_1 with the translated s'_1.

gual summarization, the model should be able to deal with inputs in multiple languages and generate the summary in the same language. Formally, for each language l_k in the collection with K languages $L = \{l_1, l_2, \cdots, l_K\}$, the training objective can be defined as

$$\mathcal{L}^{(l_k)} = -\sum_{i=1}^{N_k} logP(Y_i^{(l_k)}|D_i^{(l_k)}), \qquad (1)$$

where $D_i^{(l_k)}$ and $Y_i^{(l_k)}$ are the i-th sample for the language l_k and N_k is the size of examples in l_k.

In this section, we propose a contrastive aligned joint learning strategy for all languages to share the salient information extraction and align sentence-level representations across languages. We propose two extra training objectives for our CALMS and describe them in detail below.

3.1 Multilingual Summarization

To understand and generate text in multiple languages, it is important to have a good multilingual language model. Without loss of generality, we use mBART (Liu et al., 2020) as the model initialization. It is a powerful Transformer-based multilingual pre-trained model trained on monolingual document corpus in 25 languages with denoising training objectives. It provides a shared vocabulary across languages and a good multilingual language model. We fully share model parameters among different languages by jointly training on all summarization data in different languages. A language indicator is used to indicate the language of each example. Thus, the multilingual summarization loss for K languages is written as:

$$\mathcal{L}_s = -\sum_{k=1}^{K}\sum_{i=1}^{N_k} logP(Y_i^{(l_k)}|D_i^{(l_k)}), \qquad (2)$$

3.2 Contrastive Sentence Ranking

Different from pre-trained denoising tasks, the output is much shorter than the input in the summarization task. Therefore, it is important for the summarization model to catch the salient information from the document during the finetuning phrase. We design a contrastive training strategy, contrastive sentence ranking (CSR), to help the model distinguish salient information, which is independent of languages. Inspired by content selection in extractive summarization (Shi et al., 2019; Zhong et al., 2020), we take sentences to construct positive and negative pairs. However, instead of pre-constructing contrastive summaries pairs for the dataset, we dynamically sample sentences from the document during the training phase.

Specifically, for a document D with T sentences $D = \{s_1, s_2, \cdots, s_T\}$, we randomly sample q sentences as candidates and calculate n-gram overlaps between the ground-truth summaries and these candidates. The candidate with the highest overlaps will be viewed as positive and the others are negative. By dynamically sampling, the model is able to explore the whole document. Besides, we can change the negative sample number for each language to alleviate the imbalance between the data. Each time the data loader takes an example from the dataset, it will construct a positive-negative pair and save the corresponding sentence masks. These masks will be used to get sentence representation from the document's hidden state in the last layer of the encoder.

The model is trained with margin-based triplet loss, which is defined as:

$$\mathcal{L}_c = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} max\left(0, \sum_{j=1}^{q-1} s_{i,neg_j}^{(l_k)} - s_{i,pos}^{(l_k)} + \epsilon\right), \quad (3)$$

where $s_{i,pos}^{(l_k)}$ is the score of the positive candidate of the i-th example in language l_k, and $s_{i,neg_j}^{(l_k)}$ is the j-th negative candidate for i-th example. We use a linear layer with sigmoid function to get the score from the masked hidden state of the last layer of the encoder. ϵ is a hyper-parameter for the margin distance.

3.3 Sentence Aligned Substitution

Training with multiple languages makes it possible to share the representative space across languages and obtain a universal representation for summarization. Lin et al. (2020) randomly replaces words with a different language during the pre-training phase for machine translation. However, the input for summarization is longer than sentence-level machine translation and the single word replacement shows little influence (Kedzie et al., 2018). Thus, we propose sentence aligned substitution (SAS) for summarization.

We take lead sentences rather than randomly sampling from the document because these sentences are more important in the summarization task. We use an extra translation tool [1] to translate our sentences into another language to get the aligned information. To get rid of the lead bias, we randomly insert the translated sentences back into the original document. The training objective can be defined as:

$$\mathcal{L}_a = -\sum_{k=1}^{K}\sum_{i=1}^{N_k} logP\left(Y_i^{(l_k)}|\mathcal{R}\left(D_i^{(l_k)}\right)\right), \quad (4)$$

where $\mathcal{R}$ is the sentence replacement function. For the document in language l_k, its lead sentences are replaced with the rest languages $l_{k'}$ in ratio r.

Finally, The training objectives of CALMS can be written as:

$$\mathcal{L} = \mathcal{L}_s + \mathcal{L}_c + \mathcal{L}_a. \quad (5)$$

Figure 1 demonstrates the overview of our model. CSR takes the output of the encoder for its margin loss, while SAS replaces sentences before encoding.

4 Experiment

In this section, we describe the multilingual summarization dataset used in our experiment and the experimental settings.

4.1 Dataset

We construct a large-scale summarization dataset MLGSum with 12 languages for the multilingual summarization task. We collect articles from news websites with multiple languages, such as BBC[2] and france24[3], and select faz[4] to extend our dataset with German text. We take the brief introduction written by editors as summaries[5]. We illustrate a short French example in Table 1.

Based on the language size, we divide MLGSum into two parts: the first part includes five high-resource languages: German(De), English(En), Russian(Ru), French(Fr), and Chinese(Zh), which will be used to train our CALMS. The second part has limited training data, which includes Hindi(Hi), Spanish(Es), Indonesian(Id), Turkish(Tr), Vietnamese(Vi), Ukrainian(Uk), and Portuguese(Pt). The data of each language is split into train/dev/test by 95%/5%/5%. Compared with multilingual Gigaword used by Cao et al. (2020), whose average document/summary length is 33.1/8.6, our document and summary are longer. This asks for document-level understanding and generation. The detailed information is listed in Table 2.

4.2 Settings

We use mBART (Liu et al., 2020) as the multilingual initialization. It is the multilingual version of BART-large (Lewis et al., 2020b), which is

---

[1] https://translate.google.com/

[2] https://www.bbc.com/

[3] https://www.france24.com/

[4] https://www.faz.net/

[5] The summary is tagged with 'story-body__introduction' in BBC, 't-content__chapo' in france24, 'atc-IntroText' in faz. The data is from Jan, 2010 to Sep, 2020. We remove document length smaller than 50 or longer than 5000 and summaries shorter than five words. We provide the url of HTML page for each example.

Article Arsenal, le leader de la Premier League, a été sévèrement corrigé, samedi 14 décembre, par Manchester City (6-3) qui prend la deuxième place du classement à seulement trois points des hommes d'Arsène Wenger. [q] Il s'agit de la huitième victoire de City à domicile où il est invaincu cette saison, et ce contre la meilleure équipe à l'extérieur. [q] Les Londoniens ont commencé à prendre l'eau dès l'entame du match, Sergio Agüero ayant besoin de 14 minutes seulement pour ouvrir la marque et inscrire son 13e but de la saison en championnat.
(*Premier League leaders Arsenal were severely corrected on Saturday 14 December by Manchester City (6-3) who took second place in the standings just three points behind Arsène Wenger's men. [q] It was This is City's eighth home win where they are undefeated this season, against the best away team. [q] Londoners started to get wet from the start of the match, with Sergio Agüero needing just 14 minutes to open the scoring and score his 13th league goal of the season.*)

Summary Irrésistible à domicile depuis le début de la saison, Manchester City a étrillé Arsenal (6-3) lors du match au sommet de la Premier League. [q] Les Mancuniens reviennent à trois points des Gunners en haut du classement.
(*Irresistible at home since the start of the season, Manchester City crushed Arsenal (6-3) in the game at the top of the Premier League. [q] The Mancuniens are three points behind the Gunners at the top of the standings.*)

Table 1: A Fr example of our dataset. The text in brackets is the corresponding English translation. The sentences are separated by '[q]'.

a Transformer-based architecture (Vaswani et al., 2017) with 12 layers of encoder and 12 layers of the decoder. The hidden size is 1024 with 16 attention heads. mBART covers 25 languages and shares the vocabulary with the sentencepiece tokenizer (Kudo and Richardson, 2018), which includes 250,000 subword tokens. We follow the language indicators with mBART, and change its position to the beginning of the source and target sequence. We replace $[q]$ in the dataset with the delimiter $< /s >$ to separate sentences.

We use the first part of our dataset as training languages: De, En, Ru, Fr, Zh. We mix the training examples and do global shuffling to avoid local overfitting on a specific language. For CSR, we random sample $q = 3$ sentences from the document to construct the positive-negative pairs and let the margin $\epsilon = 1.0$. For SAS, we translate sentences to the other four languages with equal probability and substitute sentences with a ratio $r = 0.2$.

We use fairseq[6] (Ott et al., 2019) to implement the architecture. We limit the max tokens to 2048 for each GPU and set the gradient accumulation to 4. The Adam optimizer (Kingma and Ba, 2015) is used with a learning rate of 3e-5 for unified training and 1e-5 for finetuning on the specific language. The other parameters are the same as previous work (Liu et al., 2020). The joint training takes around 7 epochs and each epoch needs 5 hours on two 32G Tesla V100. During inference, we use trigram blocking to avoid repetition.

Language	Size	Doc.	Summ.	Train
De	494,514	457	27	445,062
En	191,365	476	24	172,228
Ru	87,125	499	24	78,412
Fr	85,030	463	36	76,527
Zh	65,203	799	56	58,682
Hi	59,145	565	28	53,230
Es	43,162	703	30	38,845
Id	35,495	360	21	31,945
Tr	26,539	342	20	33,047
Vi	26,539	847	34	23,885
Uk	33,214	444	21	29,892
Pt	20,945	927	34	18,850
Total	1,168,276	573.5	29.6	1,060,605

Table 2: The dataset statistic. Doc. and Summ. refer to the average length of the document and the summary. Train is the size of the training set.For non-space language like Zh and Ja (with '*'), it is calculated by the character number.

4.3 Models

Here, we describe the models used in our experiments. We first introduce several baseline models and take the strong mBART monolingual model for each language as the main competitor for our unified multilingual summarization model.

Lead2 Lead-K is the common strong baseline for summarization tasks. We select the first two sentences based on the average summary length.

Monolingual Model We train a monolingual model for each language as our baseline. We use a standard Transformer with 12 layers of encoder and decoder with 1024 hidden states and 16 heads and randomly initialize it. The number of parameters is the same as the mBART. We use an independent vocabulary for each language and tokenize them with the sentencepiece model trained on the corresponding language corpus. For the mBART model, we follow the setting of Liu et al. (2020) to finetune it on the monolingual summarization task.

---

[6] https://github.com/pytorch/fairseq

Model	Settings	De		En		Ru		Fr		Zh		Avg
		R-1	Delta	R-1	Delta	R-1	Delta	R-1	Delta	R-1	Delta	Delta
Mono	Lead2	26.35	-	22.54	-	17.21	-	**37.61**	-	29.74	-	-
	Transfromer	24.27	-	31.76	-	14.07	-	25.34	-	29.52	-	-
	mBART	25.92	-	38.89	-	21.52	-	35.75	-	38.25	-	-
Multi	mTransformer	23.91	-2.01	31.65	-7.24	15.07	-6.45	32.26	-3.49	31.65	-6.60	-5.16
	mBART	26.13	0.21	39.78	0.89	21.90	0.38	36.24	0.49	38.91	0.66	0.53
	CALMS	**26.38**	**0.46**	39.83	0.94	22.04	0.52	37.00	1.25	38.83	0.58	0.75
Finetune	mBART	26.01	0.09	39.87	0.98	21.57	0.05	36.02	0.27	38.93	0.68	0.41
	CALMS	26.33	0.41	**39.88**	**0.99**	**22.21**	**0.69**	36.88	1.13	**39.02**	**0.77**	**0.80**

Table 3: The main results. R-1 is the F1 score of ROUGE-1, and Delta is the difference between models and monolingual model initialized with mBART. Avg is the average delta for five languages. The best results are bolded.

Multilingual Model We jointly training summarization in five languages. For Transformer, we use the same shared vocabulary with mBART. We directly finetune mBART on the multilingual summarization task with the language indicator. For CALMS, we add the training objectives CSR and SAS, and the loss is defined as Equal 5. After jointly training, we directly evaluate the unified model on the test set of five languages.

Finetuning We finetune the unified mBART model and CALMS on the specific language for several steps and evaluate it on the test set. The training data for finetuning is the same as the jointly training phrase.

5 Results

We present the main quantitative results and design several qualitative analyses in this section. To better illustrate the improvement, we use the delta between different models and the strong baseline monolingual mBART in five languages for analysis.

For evaluation, we use the automatic summarization metric ROUGE(Lin, 2004)[7]. Since the original ROUGE is only designed for English, we map tokens in other languages to the digit and then calculate ROUGE. For the non-space language such as Chinese, we take each character as a token. We report the F-1 score of ROUGE-1 in the main paper and leave other scores in the appendix.

5.1 Main Results

In Table 3, we show our main results in five languages. We focus on the following questions: 1) Does a unified summarization for all languages perform better than the individual model for each language? 2) Does CALMS perform better on multilingual summarization compared with the unified mBART? 3) Does finetuning on the specific language benefit?

Monolingual v.s Multilingual For Transformer, the joint model performs worse on rich-resource De and En, while it gains improvement on Ru, Fr, and Zh. It indicates that the unified multilingual model without multilingual pre-training sacrifices the rich-resource languages and improve the low-resource languages. However, with the pre-training multilingual language model mBART, the unified model outperforms the monolingual ones on all five languages. This demonstrates that not only low-resource languages can benefit from the larger training data, but also high-resource languages can further be improved by multilingual joint training. Multilingual language models help the model to share the latent space across languages to some extend.

mBART v.s CALMS We directly evaluate the jointly training models on five languages in the test set. Compared with the unified mBART, our CALMS outperforms on all five languages, especially in Fr. For the average delta, CALMS outperforms the monolingual mBART by 0.75 ROUGE-1. The result demonstrates that CALMS is an effective and efficient solution for multilingual summarization. It can handle different languages with one unified model and improve performance on all languages without sacrificing rich-resource languages.

Does Finetuning benefit? Finetuning on CALMS makes the model further move on to the

---

[7] https://github.com/bheinzerling/pyrouge

specific language and get better results, such as En, Ru, and Zh. However, for De and Fr, it is better to directly evaluate the multilingual model, which indicates further finetuning may cause overfitting on several languages. It is similar to unified mBART, where the finetuning fails on De and Fr and benefits on En and Zh.

	De	En	Ru	Fr	Zh
CALMS	26.38	39.83	22.04	37.00	38.83
CALMS w/o CSR	26.24	39.73	22.01	36.95	38.89
CALMS w/o SAS	26.33	39.62	22.12	36.85	38.93
CALMS w/o pre-train	23.83	31.54	15.30	32.30	31.76

Table 4: Ablation study on CALMS on ROUGE-1. CALMS w/o CSR indicates removing CSR loss from CALMS. CALMS w/o mBART indicates randomly initialize the model and train with CSR and SAS.

Ablation Study We conduct the ablation study on each training strategy in Table 4. We jointly train each model and directly evaluate the test set without finetuning.

As it shown, both CSR and SAS contribute to our CALMS. Compared with CALMS w/o CSR and CALMS w/o SAS, we find De, Ru, Zh are more affected by removing CSR, while SAS is more important for En and Fr. When we remove mBART, the performance degrades significantly. This is because the multilingual pre-training language model not only provides a good initialization for multilingual representation but also have a strong generation ability as a language model, which has been proved in monolingual summarization with BART(Lewis et al., 2020b).

CALMS without pre-trained mBART can also be viewed as a jointly training mTransfromer with CSR and SAS. Compared with results in Table 3, we can find that the two training strategies improve performance in Ru, Fr and Zh, but the rich-resource languages De and En have been hurt. It implies that, without multilingual pretrained model, it is difficult for the multilingual model to recover from the denosing task SAS.

Family	Lang	Transformer	mBART	CALMS
Romance	Pt*	15.93	24.82	**25.89**
	Es	21.51	29.37	**29.77**
Slavic	Uk*	11.09	18.62	**19.23**
Turkic	Tr	13.45	**21.97**	21.68
Vietic	Vi	18.82	**30.88**	30.75
Indo-Aryan	Hi	25.53	**33.36**	32.98
Malayo-Polyn	Id*	18.61	27.17	**28.00**
Average	-	17.85	26.60	**26.90**

Table 5: Finetuning on CALMS trained on five languages. The family indicates language family and lang is the abbreviation of language. The Transformer and mBART are monolingual summarization model trained on each language. The languages with '*' are not covered by the pre-training corpus of mBART.

Transfer to other languages Does CALMS really help to learn a unified model for multilingual summarization? In order to answer this question, we further transfer the unified model to other languages. We finetune our CALMS trained on five languages to another 6 languages: Pt, Es, Uk, Tr, Vi, and Hi. Among them, Pt, Uk, and Id are not covered by the pre-training training phrase of mBART. We use '[UNK]' as the language indicator. For comparison, we also take the monolingual summarization model of each language as the baseline, which is similar to monolingual models described in 4.3. The results are listed in Table 5.

As the table shows, CALMS outperforms the monolingual Transformer and mBART in Pt, Es, Uk, and Id. Among these languages, Pt and Es is the same language family as Fr, while Uk and Ru both belong to Slavic. It indicates that our multilingual summarization model CALMS can help similar languages to get a better result against the monolingual model trained on its limited training data. For Id, it is not covered by the pre-training phase and our CALMS also shows better results on it. However, for other languages that far away from the training languages, CALMS has no obvious advantage over the monolingual model.

5.2 Analysis

In this sections, we conduct several in-depth explorations on the two training objectives CSR and SAS.

Negative Sample Number We explore how the candidate number q influences our model. Similar to above, we take the ROUGE-1 improvement against the mBART monolingual model to normalize the improvement. For the document with sentences fewer than q, we repeat the negative examples several times. After training the unified model,

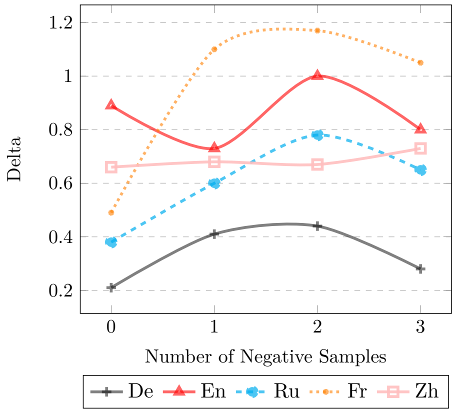

Figure 2: The negative example number of contrastive learning. 0 indicates the mBART unified model. The y-axis is the ROUGE-1 delta between our CALMS and the mBART monolingual model.

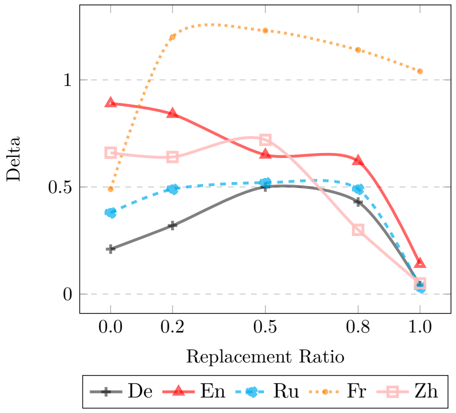

Figure 3: The replacement ratio of SAS. 0 indicates no replacement and 1 indicates each example will be subsituted. The y-axis is the ROUGE-1 delta between our CALMS and the mBART monolingual model.

	De	En	Ru	Fr	Zh
Individual	25.92	38.89	21.52	35.75	38.25
Individual + CSR	26.00	39.25	21.20	36.57	38.63

Table 6: CSA for individual model. The Individual model is trained on mBART for each language.

we directly evaluate them without finetuning.

As Figure 2 shown, The x-axis is the negative sample number, which is $q-1$. When we take two negative examples for contrastive training, most languages get the best results. However, when it comes to three, the performance slips significantly. This is because it is more likely to construct the same contrastive pair during the dynamic sampling due to the limited length of the document. Compared with other languages, the negative sample number has little impact on Zh.

Replacement Ratio We also investigate different replacement ratios r as Figure 3 shown. When $r = 1.0$, it means that we always replace lead sentences with the other language. For $r = 0.0$, we do not replace any sentences, which is the jointly training mBART model. Same as above, we evaluation the unified model directly.

For En, with the ratio increases, the performance degrades, because SAS enforces the model to obtain a more unified representation for all languages by sacrificing the English bias. When the ratio is greater than 0.5, performance begins to degrade in all languages. The Delta is almost 0 when the ratio comes to 1. This indicates that the unified model no longer has the advantage over the individual model. In this case, all the lead sentences will be inserted into the document in different languages. It will mislead the model to ignore the lead bias and the learned language indicator. The ratio between 0.2 and 0.5 is appropriate for all five languages.

CSR for Individual Different from SAS which designed for aligning multiple languages, CSR aims at distinguishing important information. It can also be used on individual models. Thus, we add CSR to the mBART monolingual model for each language and set $q = 3$. The results are listed in Table 6.

We find that De, En, Fr, and Zh all benefit from the original monolingual model, especially Fr. However, the performance degrades for Ru. From Figure 2, we can find that Ru is sensitive to the negative sample number, and Table 2 illustrates Ru have the longest article compared with De, En, and Fr (Zh is calculated by characters). Small q will lead to indistinguishable contrastive pairs during randomly sampling especially for long input, which will cause the performance decline.

6 Conclusion

We propose a contrastive aligned joint learning strategy CALMS. It is an effective and efficient solution for multilingual summarization that can handle different languages with one unified model. The experimental results show that CALMS outperforms the monolingual summarization model in all five training languages, and it can further transfer to similar languages and achieve improvement against monolingual mBART via finetuning. We also provide a multilingual summarization dataset MLGSum with 12 languages for future research.

Acknowledgements

We thank all the anonymous reviewers for their valuable suggestions. We thank the Internet Archive Projects[8] to share the archive's collections for research purposes.

Ethics Consideration

We collect the dataset from three news websites: BBC, france24, and faz. BBC provides news in more than 40 languages and each article is written by native authors. France24 is an international news website with 4 languages and faz is a German website. All of these websites have a highlight written by the editor at the beginning of the news article to summarize the main idea, which can be viewed as the summary. This information can be easily extracted through the HTML tag ('story-body__introduction' in BBC, 't-content__chapo' in france24, 'atc-IntroText' in faz). We collect MLGSum mainly from BBC and use france24 to expand French, English, and Spanish. Faz is used for German.

Similar to XSum (Narayan et al., 2018) and Newsroom (Grusky et al., 2018), we provide the Wayback archived URL of each article and the processing script to release MLGSum. The Wayback Machine[9] is an initiative of the Internet Archive, building a digital library of Internet sites that archive billions of web pages. We search news articles ranging from 2010 to 2020 for the above websites. We emphasize that the intellectual property and privacy rights of the articles belong to the original authors and the corresponding website. We carefully check the terms of use, privacy policy, and copyright policy[10] of the Internet Archive and the dataset construction is consistent with all terms.

We emphasize that we meet the usage requirements: "*Access to the Archive's Collections is provided at no cost to you and is granted for scholarship and research purposes only*" and "*abide by all applicable laws and regulations, including intellectual property laws, in connection with your use of the Archive*". We certify that our use of any part of the Archive's Collections will be limited to non-infringing or fair use under copyright law. If any authors or publishers express a desire for their documents not to be included in MLGSum, we will remove that portion from the dataset.

---

[8] https://archive.org/projects/

[9] http://web.archive.org/

[10] https://archive.org/about/terms.php

References

Yue Cao, Xiaojun Wan, Jinge Yao, and Dian Yu. 2020. Multisumm: Towards a unified model for multilingual abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11–18.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 708–719.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A universal language encoder by pretraining with multiple cross-lingual tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2485–2494.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Chris Kedzie, Kathleen McKeown, and Hal Daumé III. 2018. Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020a. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 163–169.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pretraining multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649–2663.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Khanh Nguyen and Hal Daumé III. 2019. Global voices: Crossing borders in automatic news summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 90–97.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Mlsum: The multilingual summarization corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8051–8067.

Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. 2019. Deepchannel: Salience estimation by contrastive learning for extractive document summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6999–7006.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuan-Jing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4453–4460.

Hanlu Wu, Tengfei Ma, Lingfei Wu, Tariro Manyumwa, and Shouling Ji. 2020. Unsupervised reference-free summary quality evaluation via contrastive learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3612–3621.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208.

A Appendices

We present ROUGE-2 and ROUGE-L in Table 7 and Table 8 for models in Table 3.

Different from ROUGE-1, monolingual models show an advantage over multilingual models on ROUGE-2 and ROUGE-L for De, which indicates that the multilingual models have difficulty in catching long patterns of German. However, the situation is the opposite for the French. The other trends are similar with analysis in Section 5.1.

Model	Settings	De		En		Ru		Fr		Zh		Avg
		R-2	Delta	R-2	Delta	R-2	Delta	R-2	Delta	R-2	Delta	Delta
Mono	Lead2	9.87	-	4.27	-	2.39	-	16.05	-	12.10	-	-
	Transfromer	12.30	-	9.89	-	4.16	-	7.09	-	14.51	-	-
	mBART	12.57	-	15.65	-	8.38	-	17.21	-	21.22	-	-
Multi	mTransformer	8.49	-4.08	9.67	-5.98	4.08	-4.30	11.8	-5.41	15.26	-5.96	-5.15
	mBART	11.75	-0.82	16.06	0.41	8.57	0.19	17.25	0.04	21.78	0.56	0.08
	CALMS	**11.94**	**-0.63**	16.18	0.53	8.67	0.29	**17.29**	**0.08**	21.68	0.46	0.15
Finetune	mBART	11.64	-0.93	15.39	-0.26	8.41	0.03	17.03	-0.18	21.78	0.56	-0.16
	CALMS	11.90	-0.67	**16.36**	**0.71**	**8.82**	**0.44**	17.25	0.04	**21.85**	**0.63**	**0.23**

Table 7: The main results of R-2, which is the F1 score of ROUGE-2. Delta is the difference between models and monolingual model initialized with mBART. Avg is the average delta for five languages. The best results are bolded.

Model	Settings	De		En		Ru		Fr		Zh		Avg
		R-2	Delta	R-2	Delta	R-2	Delta	R-2	Delta	R-2	Delta	Delta
Mono	Lead2	24.18	-	17.05	-	14.81	-	30.52	-	22.75	-	-
	Transfromer	22.55	-	24.1	-	12.63	-	20.91	-	24.76	-	-
	mBART	23.18	-	29.98	-	19.18	-	29.50	-	31.86	-	-
Multi	mTransformer	20.33	-2.85	23.51	-6.47	13.06	-6.12	24.85	-4.65	25.30	-6.56	-5.33
	mBART	22.80	-0.38	30.51	0.53	19.22	0.04	29.48	-0.02	31.86	0.00	0.03
	CALMS	**22.91**	**-0.27**	30.62	0.64	19.35	0.17	29.63	0.13	31.83	-0.03	0.13
Finetune	mBART	22.70	-0.48	30.28	0.30	19.01	-0.17	29.31	-0.19	31.91	0.05	-0.10
	CALMS	22.87	-0.31	**30.66**	**0.68**	**19.51**	**0.33**	**29.65**	**0.15**	**32.12**	**0.26**	**0.22**

Table 8: The main results of R-L (ROUGE-L). The other notations are the same with Table 7

When Time Makes Sense:
A Historically-Aware Approach to Targeted Sense Disambiguation

Kaspar Beelen[*]
The Alan Turing Institute
Queen Mary University of London
kbeelen@turing.ac.uk

Federico Nanni
The Alan Turing Institute
fnanni@turing.ac.uk

Mariona Coll Ardanuy
The Alan Turing Institute
Queen Mary University of London
mcollardanuy@turing.ac.uk

Kasra Hosseini
The Alan Turing Institute
khosseini@turing.ac.uk

Giorgia Tolfo
The British Library
giorgia.tolfo@bl.uk

Barbara McGillivray
The Alan Turing Institute
University of Cambridge
bmcgillivray@turing.ac.uk

Abstract

As languages evolve historically, making computational approaches sensitive to time can improve performance on specific tasks. In this work, we assess whether applying historical language models and time-aware methods help with determining the correct sense of polysemous words. We outline the task of *time-sensitive Targeted Sense Disambiguation* (TSD), which aims to detect instances of a sense or set of related senses in historical and time-stamped texts, and address two main goals: *1)* we scrutinize the effect of applying historical language models on the performance of several TSD methods and *2)* we assess different disambiguation methods that take into account the year in which a text was produced. We train historical BERT models on a corpus of nineteenth-century English books and draw on the Oxford English Dictionary (and its Historical Thesaurus) to create historically evolving sense representations. Our results show that using historical language models consistently improves performance whereas time-sensitive disambiguation helps especially with older documents.

1 Introduction

As language is in continuous flux, the question arises as to whether (and how) we should adapt Natural Language Processing (NLP) methods to the changing context in which texts are produced.[1] This paper offers a novel contribution by assessing the extent to which making NLP models sensitive to time (or historically-aware) actually improves their performance. We present the task of time-sensitive *Targeted Sense Disambiguation* (TSD), which determines whether or not a token in a given time-stamped text is related to a specific sense of a lemma. TSD is a variation on Word Sense Disambiguation (WSD), but (as we argue) it is of more practical relevance to research in digital history and cultural analysis. For example, if a historian wants to investigate the lemma *machine*[2] in the sense of "anything that transmits force or directs its application", TSD classifies whether the token *machine* (or one of its synonyms, e.g. *machine*) in a given text expresses this sense. This task assists researchers with tracking the evolution of very specific senses across time, instead of just words, as in Michel et al. (2011).

---

[*] Contributions of each author (in alphabetical order): Conceptualization: KB, BMcG, FN; Data curation: KB, GT; Formal Analysis: MCA; Funding acquisition: BMcG; Methodology: KB, MCA, KH, BMcG, FN; Project management: KB, BMcG, FN; Software: KB, MCA, KH, FN; Supervision: BMcG; Reproducibility: KB, MCA, FN; Writing: KB, MCA, BMcG, FN.

[1] Code and models used in this paper are accessible via Github and Zenodo, see https://github.com/Living-with-machines/TargetedSenseDisambiguation and https://zenodo.org/record/4782245.

[2] In this paper we will refer to lemmas or tokens in italics, their senses in single quotes and full definitions in double quotes.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2751–2761
August 1–6, 2021. ©2021 Association for Computational Linguistics

For our experiments, we use the Oxford English Dictionary (OED),[3] which provides a very detailed insight into sense-level change over time, exemplified by time-stamped quotations from a large collection of historical texts. The OED is very fine-grained, with a high number of senses for each entry. The number of historical examples for each sense is limited however, which presents a challenging data sparsity setting,[4] a not uncommon problem in WSD research and also characteristic of Digital Humanities (DH) research, in which often only a small amount of positive examples are available.

Methodologically, our approach to time-sensitive TSD builds upon recent advances in WSD and leverages BERT architectures and contextualized word vectors to address two questions:

RQ1. (time-sensitive embeddings) *Do BERT language models, fine-tuned on specific epochs, yield better representations for historical TSD?* In this scenario, we assess if models trained on data from a certain year range work better on contemporaneous texts compared to standard BERT models.

RQ2. (time-sensitive disambiguation) *When confronted with scarce historical examples, do time-sensitive methods outperform those which ignore the temporal context?* These experiments focus on the impact of time-sensitive strategies.

2 Related work

The main point of reference for Targeted Sense Disambiguation (TSD) is the extensive literature on Word Sense Disambiguation (WSD). WSD tasks can be divided into *lexical sample* and *all words* tasks (Navigli, 2009): in the former approach, models disambiguate a specific (polysemous) word in context, whereas the latter disambiguates all words in a sentence.

WSD research has a long tradition and has achieved good results in synchronic settings (Navigli, 2009; Bhattacharjee et al., 2020). However, time-sensitive WSD has received very little attention, with most literature reviews (e.g. Navigli (2009), Ranjan Pal and Saha (2015), Aliwy and Taher (2019), and Bhattacharjee et al. (2020)) not including it. In spite of the little attention received, time-sensitive WSD has important applications to DH research and the cultural heritage sector. For many applications, it is important to mine historical texts semantically, especially for historical information retrieval, OCR correction and broader research areas such as cultural analytics, as surveyed by (Tahmasebi et al., 2018, 46-47).

As far as we know, Piao et al. (2017) is the only work focusing on a time-sensitive all-words WSD system. The authors present the Historical-Thesaurus-based Semantic Tagger (HTST), a tool to annotate all lexical units of texts with the semantic categories from the Historical Thesaurus of English (Kay et al., 2016). The method by Piao et al. (2017) does not make use of corpus-driven models of word semantics. Over the past few years, a growing body of research has focused on this aspect, and researchers have developed different models for representing words' changing meaning over time. These studies have traditionally employed word embeddings models (Tahmasebi et al., 2018; Kutuzov et al., 2018), which conflate the different senses of words into a single representation. Some work has modelled the diachronic distribution of word senses (Mitra et al., 2014; Tahmasebi and Risse, 2017).

Hu et al. (2019) report on a method for building sense representations using the mapping between example sentences and sense definitions from the synchronic data of the Oxford Dictionary of English.[5] They focus on 4881 target words chosen based on a frequency filter on the COHA corpus. They first feed up to 10 sentences for each sense of every target word to a pre-trained BERT model; they then use the target word's token embeddings from the dictionary's example sentences and average them to obtain 768-dimensional embeddings for its senses. The correct sense is assigned to a token in context by finding the sense whose embedding has the highest cosine similarity score with the token embedding. They also apply their approach to track the development of individual senses of a target word over time via time series decomposition. Their system is able to trace fine-grained lexical semantic shifts as a smooth process, obtaining an improvement over previous models by Frermann and Lapata (2016) and Gulordava and Baroni (2011).

---

[3] www.oed.com.

[4] At the time of writing, the OED contains over 270,000 entries, each of them associated with one or more senses for a total of 800,000 senses (so approximately 3 senses per entry, on average), most of which have one or more quotations associated to them, for a total of over 3 million quotations (so approximately 3.75 quotations per sense, on average). See https://public.oed.com/how-to-use-the-oed/glossary/ for details.

[5] www.lexico.com.

Gonen et al. (2020) propose a new approach to detect usage change of words across corpora that is more stable and interpretable, using the differences in the top nearest neighbors of a word in a vector space as a proxy for usage change of that word. More recently, methods based on token embeddings have shown competitive results (Schlechtweg et al., 2020). Giulianelli et al. (2020) propose the first method for using contextualized (BERT) word embeddings to model sense distributions over time. They build token embeddings and then cluster them into "usage types" (which can be interpreted as senses) using K-means clustering. They then build a probability distribution from the frequencies of these usage types and use it to measure lexical semantic change.

Recent years have seen an increased interest on the application of such methods for modelling semantic change to DH research, primarily using unsupervised methods (McGillivray et al., 2019; Soni et al., 2021). However, in order to be useful, analyses for DH research require a high degree of granularity on highly complex datasets and this has not yet been achieved by state-of-the-art methods. This paper proposes a method which addresses these challenges and is therefore directly relevant and applicable to DH research.

3 Task Definition

We define the task of targeted sense disambiguation as follows: given a target sense σ (realized as a lemma-sense pair), the goal is to determine whether a token τ in a context κ is relevant to the sense σ. For evaluation purposes, we measure relevance by considering only tokens whose sense is identical or synonymous of the target sense. According to this definition, we formulate the task as a *one-vs-all* classification problem, where instances (either definitions or quotations) of the relevant sense (or senses) are considered as examples of the positive class and instances of the remaining senses are regarded as examples of the negative class.

Example. The OED lists twenty-six senses for the lemma *machine*. In this example, we will consider that the relevant sense is the one corresponding to the following definition: "A complex device, consisting of a number of interrelated parts, each having a definite function, together applying, using, or generating mechanical or (later) electrical power to perform a certain kind of work". Example 1 shows a positive (class 1) instance of this sense in a quotation containing *machine*; example 2 is a negative instance of this sense (class 0); example 3 is a quotation that shows a positive instance of a synonymous sense 'plant' (as in mechanical plant, therefore class 1); quotation 4 is a negative example of 'plant' (class 0).

(1) *The calculating **machine** now constructing under the superintendence of the inventor.*

(2) *The Church was excellent as a national refrigerating **machine**.*

(3) *Examples of mobile earthmoving **plant** are bulldozers, graders and scrapers.*

(4) *I could lift the **plant** and be far away before daylight.*

In contrast to the closely-related task of WSD, the objective of TSD is not to provide the most relevant sense for each attestation of a lemma, but to discriminate whether or not a token in a context is related to the pre-selected sense(s). Its aim is to find occurrences of senses related to a selected set of query senses (with the "relation" being an adjustable parameter in the hands of the user, in our case this is synonymy). Importantly, our setup is not static and depends on the user input: we derive an extended group of senses from an initially selected set of relevant sense(s), time period and relation(s). Therefore, depending on these decisions, each of the retrieved senses could either be a positive or a negative example, which makes the task substantively different. Lastly, in our flexible one-vs-all approach, the positive class may be realized by a group of senses (for instance senses that share a common characteristic relevant for the downstream research task) and is suitable to many text mining and information retrieval applications in DH.

4 Data

An important motivation for this work is to provide an efficient framework that leverages semantic information encoded in historical dictionaries and thesauri for research on time-stamped texts.

The Oxford English Dictionary. Considered "the definitive record of the English language" on its webpage,[6] the OED is the result of decades of careful curation by lexicographers. Senses are exemplified by quotations collected over time from

---

[6] https://www.oed.com/

different types of sources, mostly literary works, newspapers, journals, and other periodicals.[7] Each quotation has the form of a time-stamped text snippet containing the headword of the entry, the sense it represents, and is provided with metadata such as the author and title of the source. Each sense is associated with a definition, and is provided with metadata such as the date range of use and the date of its first occurrence. The example below shows a quotation for the noun *machine* with relevant metadata:

Headword: *machine*, n.
Quotation: *Windmills as hitherto made are very costly machines.*
Sense ID: machine_nn01-38475286
Definition: *"A complex device, consisting of a number of interrelated parts, each having a definite function [...]"*
Text daterange: 1659-
Keyword: *machines*
Offset: 43
Year: 1881
Source: *Nature*, by W. Thomson.

Senses are also linked to semantic classes in the Historical Thesaurus of English (HTE), therefore providing access to synonyms and other semantically related senses.

Data preparation. Although our methodology is generally applicable to any word, in our experiments we focused on a set of twelve headwords, selected because these are complex notions, spanning from political terms like *nation* to gendered and emotion words, or ambiguous terms such as *apple*.[8] Each selected headword has multiple senses.

To thoroughly evaluate our approach, we considered as many experimental scenarios as there are senses for a given headword. In each scenario, one of the headword's senses is the targeted sense (also called "seed sense"): its quotations are labeled as positive instances, while the quotations of the remaining senses are labeled as negative instances. In addition, we used the relation between the OED and the HTE to retrieve synonyms for each sense.[9] Henceforth, we will use the terms **seed sense** and **synonym sense** to distinguish between them. We thereupon expanded the set of positive and negative instances as follows:

Positive class. All quotations pertaining to a synonym of the targeted seed sense were labeled with the positive class. This means that we enriched the set of positive examples with quotations that have a different lemma than the original headword (e.g. the mechanical *plant* in the example above).

Negative class. The new lemmas (included through synonymy to expand the set of positive examples) are often ambiguous: they may refer to senses that bear no relation to the seed sense (henceforth **unrelated senses**). Therefore, for each of the new lemmas, we collect the quotations of the unrelated senses as well, and label them with the negative class.

This data expansion step allows us to overcome the problem of data sparsity, since the number of quotations per sense is generally quite low (3.75 on average). In each data set, we removed unambiguous words from the set of expanded lemmas, and held out 25% of the quotations for testing, which is consistently the same for all methods and baselines (whether supervised or unsupervised). For supervised baselines and methods, the remaining 75% is further split into training set (80%) and validation set (20%). Baselines and methods that do not require the distinction between training and validation use both as one. Finally, we consider time to be a determining factor. Therefore, our train, validation and test sets are filtered by time.

Table 1 lists the headwords with the number of seed senses, the averaged number of synonyms and unrelated senses for each seed sense (expanded senses), and the averaged number of the positive and negative quotations per seed sense for the period between 1760 and 1850.[10]

5 Experimental Design

To measure the impact of time, we designed two experiments, each addressing a research question:

---

[7] See https://oed.hertford.ox.ac.uk/ for an in-depth examination and analysis of the contents and sources of the OED.

[8] We focused on the following nouns: *anger*, *apple*, *art*, *democracy*, *happiness*, *labour*, *machine*, *man*, *nation*, *power*, *slave*, and *woman*. We choose to avoid classical semantic change examples, such as *cell*, *gay* or *mouse*, as the semantic evolution of these words is generally well known and not that relevant to current historical or DH research. The number of selected examples was determined by the call limit of the OED API. The OED API is available for researchers upon request. We provide the code to replicate the entire data extraction and processing, as long as the user has OED API credentials.

[9] According to the OED Researcher API documentation, these "may not always be precisely synonymous [...]. Co-occurrence of senses in the same semantic class tends to mean that they are semantically very close, not necessarily synonymous": https://languages.oup.com/research/oed-researcher-api/.

[10] Since there is no space to report the exact numbers for all the experimental configurations, we show the numbers for the experiment that has the strictest filters.

	Seeds	Expanded senses	Quotations
anger	6	17/121	103/564
apple	19	6/61	36/300
art	19	5/47	33/212
democracy	7	6/61	37/301
happiness	5	9/46	57/186
labour	18	4/30	29/148
machine	25	8/83	42/361
man	48	9/78	55/380
nation	15	8/85	53/430
power	39	5/49	34/244
slave	10	20/158	103/670
woman	17	10/81	64/379

Table 1: Headwords used in the experiments, with the number of seed senses, their expanded senses (synonym and unrelated) averaged per sense, and number of quotations (positive and negative examples) averaged per sense.

Experiment 1. To measure **RQ1** (impact of language model fine-tuning), we produced two historical BERT models, trained on different subsections of a 19th century book corpus:[11] one on books predating 1850 (referred to as *BERT_1850*) and one on the whole collection (referred to as *BERT_1900*). To quantify the impact of fine-tuning, we compared the performance of historical models with a standard BERT model (*BERT base uncased*) on different time-stamped subsections of our dictionary data. For each of the three different epochs e (1760-1850, 1760-1920 and 1760-2000) we followed the data preparation procedure as explained in 4 and moreover removed quotations for senses that are not current ("alive") in e.[12] In the test set, we removed all quotations that fall outside the date range defined by e (to establish how well our models work for this specific period based on training data with historically relevant senses). The selected periods align with the different language models we trained on the 19th century corpus, with exception of the last one (1760-2000), which was included to assess if fine-tuning hurts performance when more modern data are included. We hypothesized that the language model closest to the target period will yield the highest scores.

Experiment 2. To answer **RQ2** we followed the same procedure as described above, but focused on evaluating the impact of multiple time-sensitive approaches which take both a token *and* a time stamp as input. As we are primarily interested in understanding how well TSD works as a tool for historical analysis, we only compute scores for the periods 1760-1850 and 1760-1920.

We should stress at this point that the task we pursue is hard, given the complexity of the target concept we attempt to disambiguate (the selected sense and its synonyms) and the minimal number of historical examples at our disposal. At the same time, this makes TSD an excellent task for assessing the gains of historicizing NLP methods. Moreover, we argue that (even taking into account these limitations) TSD is a pragmatic and efficient task to assist with the exploration of historical texts.

6 Embedding Models

In Section 8, two types of language models are used: BERT (contextualized word representations; Devlin et al. 2019) and word2vec (static word representations; Mikolov et al. 2013):[13]

BERT. We used the *BERT base uncased* model and tokenizer as contemporary model,[14] hereinafter referred to as *BERT_base*. To investigate the impact of time on language models, we generated two historical BERT models, *BERT_1850* and *BERT_1900*, by fine-tuning *BERT_base* on a collection of historical books in English digitized by the British Library in partnership with Microsoft (henceforth *MBL*).[15] In *BERT_1850*, the contemporary BERT model was fine-tuned on the historical books published before 1850 (with ≈1.3B words). In *BERT_1900*, all MBL books were used for fine-tuning (≈5.1B words).[16] To fine-tune these models, we firstly preprocessed all books[17] and tokenized them using the original *BERT_base* tokenizer as implemented by HuggingFace[18] (Wolf et al., 2019). The tokenized sentences were then fed into the language model fine-tuning tool in which only the masked language model (MLM) objective was optimized.[19]

---

[11] See Section 6 for more information.

[12] Please note that we retain the senses that *overlap* with e, which entails that some quotations will have dates outside the range of e.

[13] See Hosseini et al. (2021) for a more detailed description of the historical language models.

[14] https://github.com/google-research/bert.

[15] Available via https://doi.org/10.21250/db14 (British Library Labs, 2014).

[16] Note that this dataset includes a few books published after 1900, however, the large majority predates 1900.

[17] We converted the text to ASCII, fixed common punctuation errors, dehyphenated broken tokens, removed most punctuation and separated the remaining punctuation marks from tokens, and finally split token streams into sentences using the *syntok* library: https://pypi.org/project/syntok/.

[18] https://github.com/huggingface/transformers.

[19] The MLM probability was set to 0.15. We used a batch size of 5 per GPU and fine-tuned for 1 epoch over the books.

Word2Vec. We used all the MBL books to train the *w2v* model using skip-gram algorithm as implemented in Gensim (Řehůřek and Sojka, 2010). For training, we chose a context window of five words and embeddings of size 300.

7 Disambiguation Methods

We present here the different approaches tested in our experiments.

7.1 Sense Embeddings from Transformers

Inspired by previous work by Hu et al. (2019) and Kutuzov and Giulianelli (2020), which leveraged contextualized word embeddings built on concrete historical examples, we start by extracting BERT embeddings for each quotation keyword in the training data. The vector we obtain is the concatenation of the last four layers.[20] We then average these keyword vectors by either label or sense: the former method creates a *binary centroid* (one for each class), the latter one for each sense (*sense centroid*). For each quotation in the test set, we produce a vector for the keyword (using the same procedure). In the case of the binary centroid, we assign it to the class of the nearest neighbour; for the sense-level centroid we obtain all sense embeddings that match the lemma of the keyword, and take the class of the nearest sense-centroid, based on cosine similarity.

7.2 Diachronic Sense Embeddings from Transformers

In this scenario, we allow the model to use the timestamp of a quotation to adjust the sense centroid. We compared two broad strategies, namely *filtering* and *weighting*. The filtering approach takes into account those observations in the training data that are temporally close (in absolute distance as $abs(year_{train\_example} - year_{target})$) and ignore the rest. In our experiments, we used the keyword vector of the temporally closest quotation (henceforth *nearest*). The weighting approach (henceforth *weighted*) takes a weighted average over vectors, with the weight determined by the temporal distance to the target quotation. For each example in the training data we *a)* first compute the temporal proximity of the target quotation (from the test set) as $1/(abs(year_{train\_example} - year_{target}) + 1))$ and normalize these scores so they add up to one; and *b)* we multiply the context vector of the keyword by this normalized score. We then simply add these time-weighted keyword vectors into one time-weighted sense centroid, after which we use the same procedure as described in section 7.1.[21]

7.3 Binary Perceptron

Instead of aggregating (collapsing all vectors into binary or sense centroids, in which useful information could get lost), we added one more method that directly uses the keyword vectors extracted from BERT to train a binary classifier. In this case we used a single fully-connected neural network (perceptron) with a RELU activation function (which equates to freezing the BERT model and only fine-tuning one fully-connected layer).

7.4 Baselines

In order to better understand the performance of each method in the different evaluation settings, we compare them to a set of widely established baselines:

Random. First of all, a random baseline, to measure the overall experimental complexity.

Lesk. Then a group of baselines measuring with different strategies the similarity between the positive sense definition and the given textual context. They do so by assessing **token overlap** and **sentence embedding**[22] cosine similarity. Such baselines, in comparison with the next one, show whether it is overall better to rely on the given definitions or on a (small number) of positive examples of quotations.

Supervised classifier. We finally present a Support Vector Machine (SVM) as a supervised binary

---

The choice of batch size was dictated by the available GPU memory (we used 4× NVIDIA Tesla K80 GPUs in parallel). Similar to the original BERT pre-training procedure, we used the Adam optimization method (Kingma and Ba, 2014) with learning rate of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and L_2 weight decay of 0.01. In our fine-tuning routine, we used a linear learning-rate warmup over the first 2,000 steps. A dropout probability of 0.1 was used in all layers.

[20] If a word has multiple subtokens, we averaged their vector representations.

[21] We have experimented with more complex methods, for example using a Gaussian distribution centred on the time of the target quotation to compute weights. As these methods were more complex but hardly showed any improvements, we decided to only report scores for the simpler implementations in the tables below.

[22] We generate sentence embeddings by element-wise average of their word embeddings (here we used the historical Word2Vec model), a common strategy for a well-performing baseline (Shen et al., 2018).

baseline classifier, trained on sentence embedding representations of positive and negative examples of quotations.

8 Evaluation

As mentioned above, in all cases, the experimental setting is very unbalanced, with just a few quotations as positive examples, as opposed to many negative ones. In order to assess the role that *time* plays in these experiments, we report the performance of each method in terms of precision, recall and F1 Score with respect to the *positive class*. Such evaluation highlights which method is most suited to identify occurrences of a pre-determined sense and will also clearly pin-point specific limitations of each approach (for instance methods with high precision, but low recall).

The tables below report performance averaged across all senses and words under study.[23]

Experiment 1. Table 2 shows that BERT approaches outperform the established baselines. Across all models and periods, the *sense centroid* approach returns the highest scores. Moreover, the results confirm our initial intuition that BERT works better when it is fine-tuned on data contemporaneous to the target period. We see for example how the performance of the BERT model trained on the first half of the nineteenth century decreases faster than the other models when more modern data is added, whereas conversely BERT base performs worse (compared to the other models) when older data is added. BERT models that have observed historical data perform better on our examples from the 19th century. Even when more recent data is added, these fine-tuned models work well, although the scores tend to converge. To gauge if historical models perform better, independently of the method used, we computed the gains (or losses) of plugging in different language models directly for all approaches. We observed that while fine-tuning does not always guarantee a jump in performance—e.g., in the binary centroid method the F1-Score declines—the overall improvement is stable for the *BERT_1900* model: given a wide enough range of document, the historical models produce higher scores. The base model fine-tuned on the whole MBL books collection seems to work best for all experiments.

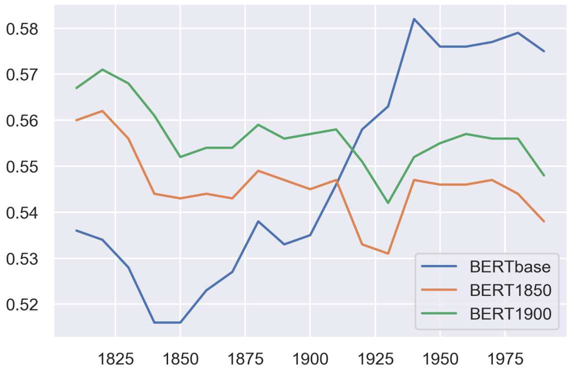

Figure 1: Optimal date range for each language model as measured by the F_1score of the positive class, using the *sense centroid* method: the x-axis represents the average points of rolling 100-year quotation date ranges.

To understand the importance of time, we repeated the experiment above, now using a rolling time-window of 100 years (and step size of 10 years) as our historical periods. Figure 1 presents the performance of our best method in relation to the language model and date range of quotations, for all the headwords. It shows a clear jump in performance for BERT_base towards the later decades of the twentieth century, clearly surpassing the models fine-tuned on nineteenth-century data, which slowly decrease as time advances. A more in-depth analysis of the data indicates that some concepts have a stronger attachment to the period of the data than others. Setting *machine* and *power* apart from the rest of the headwords, for example, indicates that they are responsible for large part of this difference between the language models. In general, *BERT_1850* is the most suitable model for early 19th century sentences, while *BERT_base* clearly outperforms the other models on later data.

If we are to mildly speculate, and attempt to deduce more general lessons from these experiments, then fine-tuning on a historical large corpus could improve results, and is preferable when scrutinizing the semantics of heritage collections. However, success is not guaranteed, as the BERT model trained on books prior to 1850 does not yield the best results when evaluated on quotations from the first half of the century. Higher granularity does not always deliver gains in accuracy, but it does not harm either: note that *BERT_1850* still improves over the base model from which it was originally derived. Also, *BERT_1900* seems to work better

---

[23] All code and models for reproducing these tables are accessible on Github and Zenodo. We should warn that results may slightly differ to those reported below, because the order in which the data are retrieved (using the OED API) and the (number of) quotations themselves might vary. Data used in this paper was downloaded in December 2020.

	1760–1850			1760–1920			1760–2000		
	Prec	*Recall*	$F_1 Score$	*Prec*	*Recall*	$F_1 Score$	*Prec*	*Recall*	$F_1 Score$
random	0.102	0.511	0.170	0.087	0.483	0.148	0.087	0.503	0.148
Lesk: token overlap	0.234	0.266	0.249	0.245	0.278	0.261	0.248	0.277	0.261
Lesk: sentence embedding	0.269	0.196	0.227	0.266	0.198	0.227	0.280	0.215	0.243
Lesk: w2v	0.323	0.291	0.306	0.288	0.270	0.279	0.286	0.257	0.271
SVM classifier	0.500	0.091	0.155	0.495	0.083	0.143	0.509	0.077	0.133
BERT_base binary centroid	0.254	0.699	0.373	0.238	0.702	0.356	0.236	0.716	0.355
BERT_base sense centroid	0.756	0.464	0.575	0.665	0.471	0.552	0.618	0.493	**0.548**
BERT_base perceptron	0.578	0.425	0.490	0.575	0.448	0.504	0.580	0.456	0.510
BERT_1900 binary centroid	0.234	0.698	0.351	0.221	0.715	0.338	0.222	0.728	0.340
BERT_1900 sense centroid	0.766	0.498	**0.604**	0.702	0.512	**0.592**	0.630	0.497	**0.556**
BERT_1900 perceptron	0.575	0.429	0.492	0.588	0.453	0.511	0.586	0.463	0.517
BERT_1850 binary centroid	0.229	0.678	0.343	0.224	0.713	0.340	0.222	0.722	0.339
BERT_1850 sense centroid	0.789	0.486	**0.602**	0.688	0.500	**0.579**	0.613	0.495	**0.548**
BERT_1850 perceptron	0.587	0.424	0.492	0.568	0.437	0.494	0.570	0.456	0.506

Table 2: Precision, recall and macro F_1scores of the positive class over all senses computed for different time periods. The table highlights the top performing methods for each experiment.

	1850	1920
BERT_base sense centroid	0.575	0.552
BERT_base nearest sense centroid	0.458	0.433
BERT_base weighted sense centroid	0.593	0.556
BERT_1900 sense centroid	0.604	**0.592**
BERT_1900 nearest sense centroid	0.505	0.464
BERT_1900 weighted sense centroid	**0.627**	**0.584**
BERT_1850 sense centroid	0.602	0.579
BERT_1850 nearest sense centroid	0.489	0.441
BERT_1850 weighted sense centroid	**0.609**	0.562

Table 3: Macro Fscores for time (in)sensitive models sense embeddings.

with a corpus spanning more than two centuries.

Experiment 2. Table 3 inspects the performance of the time-sensitive sense embeddings, applying the filtering and weighting to the *sense centroid* methods as explained in Section 7.2. The *weighted* setting is clearly superior compared to *nearest*, and sometimes outperforms the time insensitive approach. However, interestingly, a closer inspection to the results shows that weighting by time can sometimes hinder rather than help, depending on the scenario. Therefore, whereas it clearly seems to help in our experiments in, for example, headwords that largely correspond to abstract senses (*happiness*, *anger*, *art*, *democracy*, *labour*, and *nation*) and in particular for quotations that are further apart from the language model time range (e.g. *BERT_1900* applied to sentences in the 20th century, or *BERT_base* applied to sentences from the 19th and early 20th century), the *weighted* approach seems to be less helpful (and even harmful) with other headwords that have experimented rougher changes in a smaller period of time, such as words from the technological domain (*machine* and *power*) and especially for those quotations that belonged to the the same period as the the language model training or fine-tuning data.

Case studies. The last set of experiments evaluate the merits of our approach in a more focused research scenario, since—as we argued—the task of targeted sense disambiguation is a pragmatic application of word sense disambiguation tailored to the specific research needs of historians and humanities scholars more generally. We report on a series of case studies that group senses in manually curated and meaningful clusters, to simulate how TSD operates as a tool for historical and cultural analysis, for example detecting metaphorical senses of the word *machine* or scrutinizing *power* in the sense of possessing an ability (in contrast to legal interpretations of the term).[24]

As we group multiple senses, we have more examples for each category, meaning that we can evaluate the methods vertically (↓, limited to senses of *one* lemma, i.e. figurative machines versus all other machine senses) and horizontally (→, including the synonyms of the selected senses, i.e. labour in the sense of physical labour and its synonyms such as *work*). Below we report results in both directions, focusing on disambiguating the selected concepts for the long nineteenth century (1760-1920) and only running the most promising models. Before proceeding we should note that, as opposed to previous experiments, the results below have proven more volatile (i.e. dependent on data used in the

---

[24] All clusters are listed in *run_experiment_curated_cases.py* on the Github repository.

	↓	→
BERT_base sense centroid	0.691	0.536
BERT_base weighted sense centroid	0.582	0.521
BERT_base perceptron	**0.710**	0.493
BERT_1900 sense centroid	**0.700**	0.554
BERT_1900 weighted sense centroid	0.613	**0.566**
BERT_1900 perceptron	0.612	0.526
BERT_1850 sense centroid	0.658	**0.563**
BERT_1850 weighted sense centroid	0.564	0.540
BERT_1850 perceptron	0.621	0.482

Table 4: Macro Fscores for curated case studies.

train and test split) making reproduction trickier.

Not surprisingly, Table 4 indicates that the vertical comparison generally yields slightly higher scores as it is a more constrained task (stays within one lemma). But even after changing the format of the experiments, the results remain fairly consistent with previous findings, the only exception is the high score for the *BERT_base* perceptron, which suddenly achieved a very high precision in the vertical scenario. Nonetheless, the *BERT_1900* model generally has a slight edge over her BERT peers and the sense embedding methods still outperforms other approaches. An additional promising finding for future research is that time-sensitive models do appear as overall very competitive, even obtaining the highest performance for the horizontal experiments. Because these curated experiments are based on a smaller number of examples, results turned out to vary, but future work will look more closely into these distinctive and realistic historical research settings.

9 Conclusion and future work

As language is historically situated, making computational approaches more sensitive to the past should improve performance on semantic tasks relevant to cultural analysis and history.

While the Oxford English Historical Dictionary is undoubtedly a rich resource, the procedure we propose is not confined to English neither does it necessarily require a vast and fine-grained knowledge base as input. Similar dictionaries exist for other languages.[25] Moreover, the method we propose is not necessarily constrained to dictionary data: a particular strength of our approach is that it can learn from a small number of observations. Even with a few carefully collected historical examples, the procedure we propose can be used for exploring senses in a diachronic setting. The OED provided a convenient substitute for the need for annotated examples.

Focusing on targeted sense disambiguation, we demonstrated in this paper that fine-tuning BERT language models on historical texts yields better results, even when including more modern texts in the analysis. Given the complexity of the task and the minimal amount of data to learn from, this suggests that fine-tuning transformers injects historical knowledge in computational models. Historical language models, in combination with the sense centroid method, proved to be a lightweight but efficient tool for exploring the fine-grained semantics of historical texts, which we plan now to adopt to track semantic change at sense level across multiple nineteenth-century textual collections.

More generally, our paper addressed a profound issue: namely how to adapt NLP methods to time. Developing NLP methods more capable to handle the inherent challenges embedded in diachronic data has applications outside of historical and linguistic research and is relevant to the information retrieval and digital libraries communities as well.

10 Acknowledgements

Work for this paper was produced as part of Living with Machines. This project, funded by the UK Research and Innovation (UKRI) Strategic Priority Fund, is a multidisciplinary collaboration delivered by the Arts and Humanities Research Council (AHRC grant AH/S01179X/1), with The Alan Turing Institute, the British Library and the Universities of Cambridge, East Anglia, Exeter, and Queen Mary University of London. This work was also supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

Ahmed H. Aliwy and Hawraa A. Taher. 2019. Word sense disambiguation: Survey study. *Journal of Computer Science*, pages 1004–1011.

K. Bhattacharjee, S. ShivaKarthik, S. Mehta, A. Kumar, S. Phatangare, K. Pawar, S. Ukarande, D. Wankhede, and D. Verma. 2020. Survey and gap analysis of word sense disambiguation approaches on unstructured texts. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 323–327.

British Library Labs. 2014. Digitised Books. c. 1510 - c. 1900. JSON (OCR derived text). Available via: `https://doi.org/10.21250/db14`.

---

[25] E.g. for Italian and for Latin and ancient Greek.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Lea Frermann and Mirella Lapata. 2016. A Bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

Hila Gonen, Ganesh Jawahar, Djamé Seddah, and Yoav Goldberg. 2020. Simple, interpretable and stable method for detecting words with usage change across corpora. In *Association for Computational Linguistics*, pages 538–555.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of GEMS 2011*, pages 67–71. Association for Computational Linguistics.

Kasra Hosseini, Kaspar Beelen, Giovanni Colavizza, and Mariona Coll Ardanuy. 2021. Neural language models for nineteenth-century english. *arXiv preprint arXiv:2105.11321*.

Renfen Hu, Shen Li, and Shichen Liang. 2019. Diachronic sense modeling with deep contextualized word embeddings: An ecological view,. In *ACL 2019*.

C. Kay, J. Roberts, M. Samuels, I. Wotherspoon, and M. Alexander. 2016. Historical thesaurus of english.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Andrey Kutuzov and Mario Giulianelli. 2020. Uio-uva at semeval-2020 task 1: Contextualised embeddings for lexical semantic change detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation, Barcelona, Spain*. Association for Computational Linguistics.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: A survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1384–1397, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Barbara McGillivray, Simon Hengchen, Viivi Lähteenoja, Marco Palma, and Alessandro Vatri. 2019. A computational approach to lexical polysemy in Ancient Greek. *Digital Scholarship in the Humanities*, 34(4):893–907.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of ACL 2014*, pages 1020–1029.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2).

Scott Piao, Fraser Dallachy, Alistair Baron, Jane Demmen, Steve Wattam, Philip Durkin, James McCracken, Paul Rayson, and Marc Alexander. 2017. A time-sensitive historical thesaurus-based semantic tagger for deep semantic annotation. *Computer Speech & Language*, 46:113–135.

Alok Ranjan Pal and Diganta Saha. 2015. Word sense disambiguation: A survey. *International Journal of Control Theory and Computer Modeling*, 5(3):1–16.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Dominik Schlechtweg, Barbara McGillivray, Simon Hengchen, Haim Dubossarsky, and Nina Tahmasebi. 2020. SemEval-2020 task 1: Unsupervised Lexical Semantic Change Detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, Barcelona, Spain. Association for Computational Linguistics.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*.

Sandeep Soni, Lauren F. Klein, and Jacob Eisenstein. 2021. Abolitionist Networks: Modeling Language Change in Nineteenth-Century Activist Newspapers. *Journal of Cultural Analytics*.

Nina Tahmasebi, Lars Borin, and Adam Jatowt. 2018. Survey of computational approaches to lexical semantic change. In *Preprint at ArXiv 2018*.

Nina Tahmasebi and Thomas Risse. 2017. On the uses of word sense change for research in the digital humanities. In *Research and Advanced Technology for Digital Libraries - 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017. Proceedings / edited by Jaap Kamps, Giannis Tsakonas, Yannis Manolopoulos, Lazaros Iliadis, Ioannis Karydis*, pages 246–257, Berlin. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771.

Understanding Feature Focus in Multitask Settings for Lexico-semantic Relation Identification

Houssam Akhmouch
Normandie Univ, UNICAEN
ENSICAEN, CNRS, GREYC
Crédit Agricole Brie Picardie
France

Gaël Dias
Normandie Univ, UNICAEN
ENSICAEN, CNRS, GREYC
France

Jose G. Moreno
Université de Toulouse
IRIT UMR 5505 CNRS
France

Abstract

Discovering whether words are semantically related and identifying the specific semantic relation that holds between them is of crucial importance for automatic reasoning on text data. For that purpose, different methodologies have been proposed that either (1) tackle feature engineering, (2) fine-tune latent semantic spaces, or (3) take advantage of cognitive links between semantic relations in multitask settings. In this paper, we investigate how feature engineering and multitask architectures can be improved and consequently combined to identify lexico-semantic relations. Evaluation results over a set of gold-standard datasets show that (1) combinations of similar features are beneficial (feature sets), (2) asymmetric distributional features are a strong cue to discriminate asymmetric relations as well as they play an important role in multitask architectures, (3) shared-private models improve over binary and fully-shared classifiers as well as they correctly balance the focus on features between private and shared layers[1].

1 Introduction

The ability to automatically identify lexico-semantic relations is an important issue for Information Retrieval and Natural Language Processing applications such as question answering (Dong et al., 2017), query expansion (Kathuria et al., 2017), or text summarization (Gambhir and Gupta, 2017). Lexico-semantic relations embody symmetric and asymmetric linguistic phenomena such as synonymy (e.g. phone ↔ telephone), co-hyponymy (e.g. phone ↔ monitor), hypernymy (e.g. phone → speakerphone) or meronymy (e.g. phone → mouthpiece), but more can be enumerated (Vylomova et al., 2016).

Most approaches focus on modeling a single semantic relation and consist in deciding whether a given relation r holds between a pair of words (w_1, w_2). The vast majority of efforts (Shwartz et al., 2016; Vulić and Mrkšić, 2018; Wang and He, 2020) concentrate on hypernymy which is the key organization principle of semantic memory, but studies exist on antonymy (Nguyen et al., 2017b; Ali et al., 2019), meronymy (Glavaš and Ponzetto, 2017) and co-hyponymy (Jana et al., 2020). Within this scope, different strategies have been proposed that either define new features (Santus et al., 2017; Vu and Shwartz, 2018) or build specific latent semantic spaces (Nguyen et al., 2017a; Rei et al., 2018; Wang and He, 2020) for the relation at hand.

More recently, multitask strategies have been proposed, which consist in concurrently learning correlated lexico-semantic relations (Attia et al., 2016; Balikas et al., 2019; Bannour et al., 2020), the underlying idea being that if two (or more) tasks are cognitively interlinked, a learning architecture should improve its generalization ability by taking into account the shared information existing between the tasks (Caruana, 1998).

In this paper, we propose to investigate how feature engineering can be coupled to multitask strategies for the identification of lexico-semantic relations. On the one hand, Vu and Shwartz (2018) show that the introduction of the generalized cosine ($Mult$) drastically improves results over the unique concatenation of word embeddings, thus clearly evidencing the limitations of general-purpose latent spaces. However, a complete study of symmetric and asymmetric characteristics, and their combination is still lacking, except (Santus et al., 2017), one of the most complete work in the field.

On the other hand, although existing multitask strategies have been showing promising results,

---

[1] Both the code and the datasets are available at `https://github.com/Houssam93/Feature-Focus-in-Multi-Task-Learning-NLP` for reproducibility.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2762–2772
August 1–6, 2021.

they neither take advantage of specialized features nor they implement state-of-the-art architectures, which have been successful for text classification (Liu et al., 2017). This might be due to the fact that the combination of features within shared-private multitask architectures is not straightforward, and requires specific tuning.

Evaluation results over a set of gold-standard datasets (RUMEN (Balikas et al., 2019), ROOT9 (Santus et al., 2016), WEEDS (Weeds et al., 2004) and BLESS (Baroni and Lenci, 2011)) of an architecture coupling optimized feature sets and shared-private models show that

- The combination of features within a family set improves performance over the use of a unique family member;
- Asymmetric distributional features are a strong cue to discriminate asymmetric lexico-semantic relations;
- Shared-private models improve over binary and fully-shared classifiers (Balikas et al., 2019; Bannour et al., 2020) as well as they correctly balance the focus on features between private and shared layers;
- Asymmetric distributional features play an important role in multitask architectures, being an important source of information for combining both symmetric and asymmetric tasks.

2 Related Work

Three major research directions have been proposed to identify lexico-semantic relations: (1) feature engineering, (2) construction of fine-tuned semantic spaces and (3) multitask architectures.

Within the first topic, (Levy et al., 2015) and (Vylomova et al., 2016) proposed similar evaluations to combine word input vectors ($\overrightarrow{w_1}$, $\overrightarrow{w_2}$), following initial experiments of (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). In particular, word pairs are encoded as the concatenation of the constituent word representations ($\overrightarrow{w_1} \oplus \overrightarrow{w_2}$), their vector difference ($\overrightarrow{w_1} - \overrightarrow{w_2}$) or their sum ($\overrightarrow{w_1} + \overrightarrow{w_2}$). Both studies evidence that the distributional hypothesis is domain-dependent by nature and as such models may not generalize across domains based on these input representations. To overcome such a limitation (Shwartz et al., 2016; Nguyen et al., 2017b) proposed to represent contextual patterns as continuous vectors with successful results, while (Vu and Shwartz, 2018) defined a generalized cosine ($\overrightarrow{w_1} \otimes \overrightarrow{w_2}$) that successfully combines with ($\overrightarrow{w_1} \oplus \overrightarrow{w_2}$).

The second main research direction aims to build fine-tuned neural latent semantic spaces. (Nguyen et al., 2017a) proposed *HyperVec*, where embeddings are learned in a specific order to capture the hypernym–hyponym distributional hierarchy from a background knowledge of hypernym-hyponym pairs. (Vulić and Mrkšić, 2018) rather proposed a post-processing strategy that retrofits the knowledge background into an original latent space. Such methods suffer from limited coverage as they affect only vectors of seen words. To deal with this limitation, (Kamath et al., 2019) presented a post-processing method that specializes vectors of all vocabulary words by learning a global specialization function, and (Wang and He, 2020) followed the same idea but proposed to learn two projection functions. In the same line, (Bouraoui et al., 2020) introduced a framework that fine-tunes BERT (Devlin et al., 2019) to include relational information.

The third approach tackles relation identification from the architecture point of view. Within this context, (Attia et al., 2016) can be viewed as a coarse-grained analysis as they propose a multitask convolutional neural network where one task acts as domain adaptation (relatedness between two words) and the second task is a multiclass classification problem for hypernymy, meronymy, synonymy and antonymy. Instead, (Balikas et al., 2019) proposed a fine-grained approach, that determines whether the learning process of a given semantic relation can be improved by the concurrent learning of another relation, where relations are synonymy, co-hyponymy, hypernymy and meronymy. (Bannour et al., 2020) implemented the same fully-shared model, but introduced the idea of data augmentation via attention models.

Although fine-tuned embeddings have evidenced improved results over generic ones, they are relation- and knowledge-dependent. One exception is proposed by (Meng et al., 2019), which learns text embeddings in a spherical space (aka. JoSE) suitable for relational information. Feature engineering also affords “cheap performance boost” (Vu and Shwartz, 2018) in resource-free environments. But, a complete study of the combination of features is still missing as well as the definition of asymmetric features in the context of continuous spaces, although a great deal of work exists for the discrete case (Kotlerman et al., 2010; Santus et al., 2017). Finally, studies in multitask settings neither take advantage of powerful multitask models such as shared-private architectures (Liu et al., 2017) that allow to combine task-specific and cross-task information, nor benefit from the fruitful combination of distributional and pattern-based features. In this paper, we propose to deal with the aforementioned limitations in a resource-free setup.

3 Feature Engineering

Additionally to word embeddings concatenation, we define three families of features based on the distributional hypothesis (symmetric and asymmetric features) and the paradigmatic approach (pattern-based features) in continuous semantic spaces.

3.1 Distributional Representation

Most studies have been evidencing the superiority of the concatenation of representational word vectors to infer their semantic relationship (Shwartz et al., 2016; Vu and Shwartz, 2018). So, we follow this line of research. Let (w_1, w_2) be a word pair and $\overrightarrow{w_1}$, $\overrightarrow{w_2}$ their respective distributional representations of dimension d. The input distributional feature of the word pair is noted $\overrightarrow{w_1} \oplus \overrightarrow{w_2}$.

3.2 Symmetric Distributional Features

Studies have evidenced the interest of coupling word embeddings with specific features to improve relation identification. In particular, the cosine similarity measure cos has shown promising results (Garten et al., 2015; Barkan, 2017). However, Vu and Shwartz (2018) have demonstrated the effectiveness of integrating the element-wise multiplication of the input vectors, which can be seen as a generalized cosine ($cosG$, aka. $Mult$), which is defined in equation 1.

$$cosG(\overrightarrow{w_1}, \overrightarrow{w_2}) = \bigoplus_{i=1}^{d} w_1^i w_2^i \tag{1}$$

While the cosine only provides a unique value as input, $cosG$ refers to an input of dimension d, thus evidencing a dimensional issue. As a consequence, we propose to transform the $cosG$ into a unique value by using a linear activation layer as in equation 2. The $cosG1D$ can be seen as a control value of $cosG$ taking into account the dimensional bias (from high to low dimension).

$$cosG1D(\overrightarrow{w_1}, \overrightarrow{w_2}) = \sum_{i=1}^{d} \lambda_i w_1^i w_2^i \tag{2}$$

The counterpart of equation 2 is the (d times) duplication of the cosine value. This metric called cosine broadcast ($cosBr$) defined in equation 3 aims to control the dimensional issue from a low to a high dimension.

$$cosBr(\overrightarrow{w_1}, \overrightarrow{w_2}) = \bigoplus_{i=1}^{d} cos(\overrightarrow{w_1}, \overrightarrow{w_2}) \tag{3}$$

As such, in equation 4, we define a family of symmetric distributional features.

$$CosF = (cos, cosG, cosBr, cosG1D) \tag{4}$$

In the next subsection, we detail the design of new asymmetric distributional measures based on the Kullback–Leibler divergence.

3.3 Asymmetric Distributional Features

Asymmetry has shown successful results for the discrete case (Kotlerman et al., 2010; Santus et al., 2017), the underlying idea being that the relation between words may be unbalanced such that one word attracts the other one more than the opposite. Here, we define different asymmetric features in the continuous space based on the Kullback–Leibler divergence (Kullback and Leibler, 1951). To fit to the continuous case, we transform each dimension of a word vector with the sigmoid (σ) function such that all values range between 0 and 1. Thus, each word can be considered as a probability distribution and the asymmetric metric $Kull$ is defined in equations 5.

$$Kull(\overrightarrow{w_1}|\overrightarrow{w_2}) = \sum_{i=1}^{d} log(\frac{\sigma(w_1^i)}{\sigma(w_2^i)})\sigma(w_1^i) \tag{5}$$

To take into account both directions of the asymmetry, we propose to concatenate the $Kull$ values for both directions as defined in equation 6.

$$kull(\overrightarrow{w_1}, \overrightarrow{w_2}) = Kull(\overrightarrow{w_1}|\overrightarrow{w_2}) \oplus Kull(\overrightarrow{w_2}|\overrightarrow{w_1}) \tag{6}$$

Similarly to the $cosG$, we propose to define the multiplicative version of the $kull$, such that $kullG$ integrates the element-wise multiplication of the input vectors as defined in equations 7 (single asymmetry) and 8 (concatenation of both asymmetries).

$$KullG(\overrightarrow{w_1}|\overrightarrow{w_2}) = \bigoplus_{i=1}^{d} log(\frac{\sigma(w_1^i)}{\sigma(w_2^i)})\sigma(w_1^i) \tag{7}$$

$$kullG(\overrightarrow{w_1}, \overrightarrow{w_2}) = KullG(\overrightarrow{w_1}|\overrightarrow{w_2}) \oplus KullG(\overrightarrow{w_2}|\overrightarrow{w_1}) \tag{8}$$

Similarly to $cosG1D$ and to take into account the dimensional issue of the multiplicative version of the Kullback-Leibler, we define $kullG1D$ in equations 9 and 10 .

$$KullG1D(\overrightarrow{w_1}|\overrightarrow{w_2}) = \sum_{i=1}^{d} \lambda_i log(\frac{\sigma(w_1^i)}{\sigma(w_2^i)})\sigma(w_1^i) \quad (9)$$

$$\begin{aligned} kullG1D(\overrightarrow{w_1}, \overrightarrow{w_2}) \quad = \quad & KullG1D(\overrightarrow{w_1}|\overrightarrow{w_2}) \oplus \quad (10) \\ & KullG1D(\overrightarrow{w_2}|\overrightarrow{w_1}) \end{aligned}$$

Similarly to $cosBr$, we propose to define $kullBr$ based on the (d times) duplication of the Kulback-Leibler value for both directions as in equation 11.

$$\begin{aligned} kullBr(\overrightarrow{w_1}, \overrightarrow{w_2}) \quad = \quad & \bigoplus_{i=1}^{d} Kull(\overrightarrow{w_1}|\overrightarrow{w_2}) \quad (11) \\ & \bigoplus_{i=1}^{d} Kull(\overrightarrow{w_2}|\overrightarrow{w_1}) \end{aligned}$$

As such, in equation 12, we define a family of asymmetric distributional features.

$$KullF = (kull, kullG, kullBr, kullG1D) \quad (12)$$

In the next subsection, we present the encoding strategy of patterns embodying the paradigmatic approach.

3.4 Pattern-based Paradigmatic Features

Patterns are part of the paradigmatic approach (Hearst, 1992), which suggests that specific word sequences may exist that link two words in a given relation. Some examples of sequences between word pairs are given in Table 1, which evidence that some of them can be spurious, and do not necessarily include patterns.

Here, we propose to implement the methodology of (Shwartz et al., 2016) to encode patterns into continuous spaces. As such, we transform the k^2 most frequent patterns occurring between w_1 and w_2 using either BiLSTM or the Universal Sentence Encoder (USE) (Cer et al., 2018), and then perform average pooling to get the final input representation. The encoded i-th most frequent pattern is defined in equation 13, where $j \in \{\text{BiLSTM}, \text{USE}\}$, $i \in [1..k]$, and the average representation of the k patterns is noted $\overline{pat}_{*,j}^{w_1,w_2}$.

Relation	Path
Synonymy	**error** or **fault** √ **change** as an **alteration** √ **burning** fuel in the **combustion**
Hypernymy	**aircraft** firing rocket into an enemy **plane** √ **unit** that includes **screen** √ **act** was an unconscious **ritual**
Co-hyponymy	**pineapple** and **apricot** √ **chisel** usually used with **mallet** √ **horse** frightened by **lion** √
Meronymy	**bowl** from the world of **glass** √ **television** and **video** √ **couch** on **seat** √
Random	**reference** in the book of **mormon** **nothing** to stop the **robber** **driver** was issued traffic **ticket**

Table 1: Examples of patterns for a word pair (in bold).

$$pat_{i,j}^{w_1,w_2} = encoder_j(w_1, path_i, w_2) \quad (13)$$

Similarly to $CosF$ and $KullF$, we define a family of pattern-based features $PatF$ in equation 14.

$$PatF = (\overline{pat}_{*,\text{USE}}, \overline{pat}_{*,\text{BiLSTM}}) \quad (14)$$

In the next section, we present the multitask settings that have been implemented to take into account relations between lexico-semantic relations.

4 Multitask Settings

Multitask architectures have shown to successfully combine closely-related lexico-semantic relations. Within this scope, the fully-shared architecture has systematically been implemented (Attia et al., 2016; Balikas et al., 2019; Bannour et al., 2020), which relies on a unique shared representation capable of solving the different tasks learned concurrently from a given input.

However, the shared-private model has proved to boost results for text classification (Liu et al., 2017). In particular, a shared-private network combines $N + 1$ different representations (one shared and N task-specific). As such, the shared layer should transfer the joint information contained in all tasks, while private layers should focus on the specific information of each task.

Moreover, $N + 1$ different input representations may coexist in the shared-private case, while a unique input representation exists for fully-shared models. Here, we propose to implement both fully-shared and shared-private architectures for different

---

[2] k allows to deal with spurious sequences.

combinations of input representations and features $X = (\overrightarrow{w_1} \oplus \overrightarrow{w_2}, CosF, KullF, PatF)$. In particular, forward selection (Kohavi and Sommerfield, 1995) is used for feature selection, as the search space is huge, 2^{10} possible combinations[3].

4.1 Multitask Architectures

The neural architectures are presented in figure 1 for two tasks. Formally, let X_k be an input vector[4], we compute a shared layer $S(X_k)$ as in equation 15, where W_{S^k} is a weight matrix, b_{S^k} a bias vector, and $k \in [1, K]$ (K the number of shared layers).

$$S(X_k) = \sigma(W_{S^k} X_k + b_{S^k}) = X_{k+1} \qquad (15)$$

A private layer $H^j(Z_q)$, which solves task T_j ($j \in [1, N]$) is defined in equation 16, where $q \in [1, Q]$ (Q is the number of private layers).

$$H^j(Z_q) = \sigma(W^j_{H^q} Z_q + b^j_{H^q}) = Z_{q+1} \qquad (16)$$

For the fully-shared architecture $Z_1 = S(X_K)$ and for the shared-private model $Z_1 = S(X_K) \oplus X^i$, where X^i is the specific input vector for task T_i. Finally, the N decisions are defined in equation 17.

$$O^j = \sigma(W^j_O H^j(Z_Q) + b^j_O) \qquad (17)$$

The parameters are updated by minimising the binary cross-entropy. Hence, the weights of the shared layer are updated by minimising the loss function of each task alternatively, while the private layers are updated for their specific task.

4.2 Forward Selection

In order to optimize the feature combination for all $N + 1$ tasks and thus find the best input vectors for the shared and private layers (i.e. X, X^1 and X^2 in figure 1), we perform forward selection. As such, we first train the given model to find the best combination of features within a given family (i.e. within $CosF$, $KullF$ and $PatF$ individually)[5]. Once the best within-family combination has been defined for all families, we train the model for all combinations of the best within-family combinations of features. Note that for the shared-private architecture, we first train the private models independently to determine X^i ($i \in [1, N]$) and based on these findings, we train the shared-private model to determine X, constrained by the previously learned private models with input X^i.

5 Experimental Setups

5.1 Datasets

There exist a large body of related works for the identification of lexico-semantic relations. The first gold-standard dataset, WEEDS, has been proposed by (Weeds et al., 2004) in the context of studies about measures of lexical similarity. Following the same objective, (Baroni and Lenci, 2011) introduced the well-known BLESS dataset, and (Santus et al., 2016) compiled the ROOT9 dataset[6], which contains word pairs randomly extracted from EVALution (Santus et al., 2015), Lenci/Benotto (Benotto, 2015) and BLESS (Baroni and Lenci, 2011). Within the context of concurrent identification of lexico-semantic relations, (Balikas et al., 2019) recently introduced the RUMEN dataset[7] to include synonymy. As the patterns are not included in the original datasets, we downloaded the English wikipedia dump[8] and extracted all patterns that do not exceed a maximum length of 10 words[9]. All datasets[10] are summarized with their specific characteristics in Table 2.

5.2 Learning Configurations

The output dimension of the Universal Sentence Encoder (USE) equals to 512. The output size of the BiLSTM $\in \{100, 200, 300, 400, 500\}$, the number of patterns ($k \in [1..5]$), the number of hidden layers ($K \in \{1, 2\}$ and $Q \in \{1, 2\}$), the number of neurons $\in \{5, 20, 50, 100, 150, 200, 300\}$ and the number of epochs ([1..100]) are free hyperparameters that are tuned using grid search. The weights are initialised with a uniform distribution scaled as in (Glorot and Bengio, 2010) and updated using Adam (Kingma and Ba, 2014) with a learning rate set to 0.001. The network is trained with batches of 64 examples and the number of iterations is optimized to maximize the F_1 score on the validation set. Word embeddings are initialized with the 300-dimensional representations of GloVe (Pennington

---

[3] Embedding concatenation is the compulsory input.

[4] $X_1 = X$, where X is the initial input vector that combines both embeddings and a set of features specific to the task at hand.

[5] Here the model is trained three times independently for each family.

[6] `https://github.com/esantus/ROOT9`

[7] `https://bit.ly/2Qitasd`.

[8] `shorturl.at/cqtQ8`

[9] This value was tuned experimentally.

[10] Complete versions are available at `https://github.com/Houssam93/Feature-Focus-in-Multi-Task-Learning-NLP/tree/main/Data`

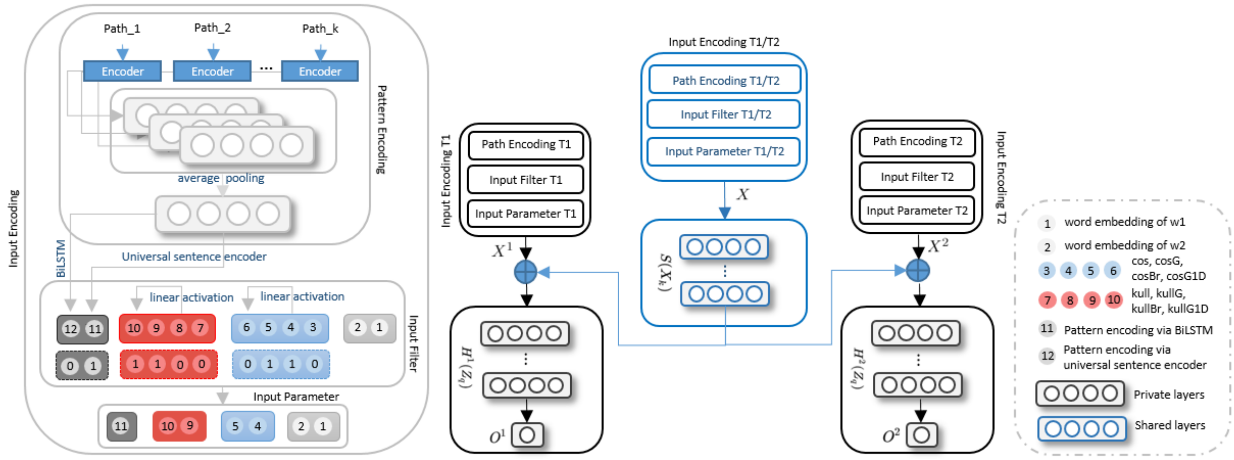

Figure 1: Fully-shared and shared-private architectures with multiple input feature combinations. The fully-shared network only includes the blue layer, i.e. $S(X_h)$.

	Synonym		Hypernym		Co-hyponym		Meronym		Random	
Dataset	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)
RUMEN (Balikas et al., 2019)	6326	44/14/8/34	6326	65/12/5/18	-	-	-	-	6326	93/4/1/2
ROOT9 (Santus et al., 2016)	-	-	2447	21/9/6/64	3200	28/14/8/50	-	-	1100	78/9/4/9
WEEDS (Weeds et al., 2004)	-	-	1257	40/13/6/41	2083	60/11/5/24	-	-	6326	93/4/1/2
BLESS (Baroni and Lenci, 2011)	-	-	1337	57/9/4/30	3565	34/12/7/40	2943	99/0/0/1	6702	97/1/0/2

Table 2: Details of the RUMEN, ROOT9, WEEDS and BLESS datasets. 0 / 1 / 2 / >2 stands for the percentage of word pairs having respectively no pattern, 1 pattern, 2 patterns and more than 2 patterns in the Wikipedia dump.

et al., 2014) or JoSE (Meng et al., 2019). All state-of-the-art models presented in section 6 have been implemented to provide average results and perform statistical tests[11].

5.3 Lexical Split

As suggested in (Levy et al., 2015), lexical split is applied to all our experiments so that there is no vocabulary intersection between the test set and the train/validation sets. Note that for learning purposes, each dataset is split into train (50%), validation (20%) and test (30%) sub-datasets.

6 Evaluation

All comparative results against four state-of-the-art models (Shwartz et al., 2016; Vu and Shwartz, 2018; Balikas et al., 2019; Bannour et al., 2020) are presented in Table 3 for an average of 25 runs with evidenced statistical significance over four gold-standard datasets.

6.1 Private Models

We first start by analysing the impact of feature combination on private models, i.e. when a unique lexico-semantic relation is taken into account in the learning process. This stands for the first four rows of Table 3. Unsurprisingly, the introduction of a combination of (eventually new) features (*Best MLP*) outperforms existing models (Shwartz et al., 2016; Vu and Shwartz, 2018) and the multilayer perceptron (MLP) that only includes word embeddings concatenation (i.e. the simplest baseline). Note that the *Best MLP* model includes the architectures of (Shwartz et al., 2016) and (Vu and Shwartz, 2018) as it allows the combination of all family features as input.

To better understand the impact of feature engineering, we illustrate results involving all combinations of within-family features and all combinations of in-between best family features in figure 2 (a). Within the $cosF$ family alone (i.e. only cosine-based metrics are used for the learning process)[12], results clearly evidence the dimensional issue, being cos and $cosG1D$ the one-dimension metrics that evidence worst results individually. The second important finding lies in the fact that metric combination steadily improves over individual metrics. In particular, $(cosG, cosBr, cosG1D)$ gives rise to strongest results in the vast majority of cases, and particularly for hypernymy.

---

[11] Source codes are available at `https://github.com/Houssam93/Feature-Focus-in-Multi-Task-Learning-NLP`

[12] Blue dots in figure 2.

		Synonym vs Random				Hypernym vs Random			
	Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
RUMEN	MLP	0.754	0.754	0.750	0.759	0.750	0.757	0.731	0.786
	Shwartz and Dagan (2016)	0.713	0.731	0.685	0.783	0.770	0.776	0.754	0.798
	Vu and Shwartz (2018)	0.851	0.847	0.864	0.831	0.842	0.843	0.832	0.854
	Best MLP	0.867 $\star$	0.865 $\star$	0.871 $\star$	0.859$\star$†	0.863 $\star$	0.862 $\star$	0.860 $\star$	**0.865**$\star$
	Balikas et al. (2019)	0.758	0.759	0.750	0.769	0.759	0.762	0.747	0.778
	Bannour et al. (2020)	0.854	0.850	0.873	0.827	0.819	0.784	0.812	0.756
	Best Fully-shared (FS)	0.861	0.864	0.843	**0.887**	0.860	0.859	0.861	0.856
	Best Shared-private (SP)	**0.870**†+	**0.866**+	**0.889** †+	0.844+	**0.869** †+	**0.867** †+	**0.871** †+	0.864+
		Co-hyponym vs Random				Hypernym vs Random			
	Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9	MLP	0.909	0.939	0.954	0.925	0.904	0.936	0.944	0.929
	Shwartz and Dagan (2016)	0.919	0.946	0.955	0.938	0.842	0.901	0.860	0.946
	Vu and Shwartz (2018)	0.940	0.961	0.962	0.959	0.943	0.962	0.961	0.964$\star$
	Best MLP	0.950 $\star$	0.967 $\star$	**0.973**$\star$	0.959	**0.947**$\star$†	**0.965**$\star$†	**0.971**†$\star$	0.959†
	Balikas et al. (2019)	0.909	0.940	0.949	0.931	0.911	0.941	0.949	0.932
	Bannour et al. (2020)	0.949	0.966	0.964	**0.969**+	0.908	0.932	0.941	0.923
	Best Fully-shared (FS)	0.947	0.965	0.971	0.959	0.944	0.963	0.964	**0.962**
	Best Shared-private (SP)	**0.951**+	**0.968** +	0.971+	0.964†	0.943+	0.962+	0.969+	0.955+
		Co-hyponym vs Random				Hypernym vs Random			
	Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
WEEDS	MLP	0.720	0.449	0.422	0.479	0.726	0.457	0.432	0.485
	Shwartz and Dagan (2016)	0.769	0.532	0.513	0.552	0.716	0.474	0.423	0.539
	Vu and Shwartz (2018)	0.848	0.691	0.669	0.714	0.833	0.661	0.641	0.682
	Best MLP	0.873 $\star$	0.737$\star$	0.729$\star$	0.746$\star$†	0.886 $\star$	0.746$\star$	0.797$\star$	0.701$\star$
	Balikas et al. (2019)	0.721	0.443	0.422	0.466	0.724	0.462	0.431	0.498
	Bannour et al. (2020)	0.871	0.713	0.754	0.678	**0.924**+	**0.751** +	**0.854**+	0.669
	Best Fully-shared (FS)	0.864	0.737	0.685	**0.796**	0.873	0.736	0.727	**0.746**
	Best Shared-private (SP)	**0.890**†+	**0.761**†+	**0.789**†+	0.736+	0.882+	0.743	0.771	0.717†
		Meronym vs Random				Hypernym vs Random			
	Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
BLESS	MLP	0.839	0.748	0.805	0.698	0.845	0.762	0.797	0.731
	Shwartz and Dagan (2016)	0.855	0.781	0.807	0.756	0.842	0.754	0.804	0.709
	Vu and Shwartz (2018)	0.886	0.820	0.883	0.765	0.882	0.811	0.891$\star$	0.744
	Best MLP	0.909 $\star$	0.864 $\star$	0.883	**0.847** $\star$	0.905$\star$	0.859 $\star$	0.872	0.847 $\star$†
	Balikas et al. (2019)	0.846	0.759	0.814	0.711	0.848	0.764	0.812	0.721
	Bannour et al. (2020)	0.896	0.837	**0.913**+	0.770	**0.954** +	0.821	0.883	0.769
	Best Fully-shared (FS)	0.903	0.850	0.895	0.810	0.906	0.862	0.861	**0.864**
	Best Shared-private (SP)	**0.912** †+	**0.868**†+	0.890†	**0.847**+	0.916	**0.873**†+	**0.906**†+	0.843+

Table 3: Overall results for all architectures with GloVe embeddings. Lexical split is applied. $\star$, † and + denote p-value ≤ 0.05 based on the t-Test assuming unequal sample variances of metric values between respectively (Best MLP) against (Vu and Shwartz, 2018), (Best SP) against (Best MLP), and (Best SP) against (Bannour et al., 2020).

Within the $KullF$ family alone[13], results seem to indicate that $kullBr$ is the less performing (alone and in combination) feature, although regularities are difficult to establish as different results can be observed depending on the dataset. Similarly to the previous observation, the combination of asymmetric features provides improved results for the vast majority of cases, suggesting that individual values encode complementary information.

Within the $PatF$ family[14], the BiLSTM encoding seems to provide superior results to the USE encoding, but more importantly, results clearly show that pattern-based features can be a strong cue for the classification process provided that a large number of patterns can be extracted, as it is shown for ROOT9 (see Table 2 for the number of patterns).

More surprisingly, the $CosF$ features steadily indicate stronger results than the $KullF$ and $PatF$ features for asymmetric relations (hypernymy and meronymy), thus suggesting that symmetry is an important characteristic for all relations.

Finally, results clearly show that the combinations of the best features per family[15] steadily outperform results of individual family features, thus demonstrating their complementarity. In particular, symmetric and asymmetric distributional features successfully combine for asymmetric relations, and the successful combination is with pattern-based and cosine-based features for co-hyponymy. However, only symmetric distributional features allow maximum performance for synonymy, which can easily be understood as this is a symmetric relation. To strengthen our comments, we give the distribution of features for the best configurations in Table 4 (first row) for all datasets and relations.

6.2 Multitask Models

Results of the multitask architectures are presented in rows 5-8 of Table 3. In particular, the *Best Fully-shared* network stands for the model of Balikas et al. (2019) with an optimized set of input features, oppositely to their settings which rely on the unique concatenation of word embeddings. Figures clearly

---

[13] Red dots in figure 2.

[14] Black dots in figure 2.

[15] Green dots in figure 2.

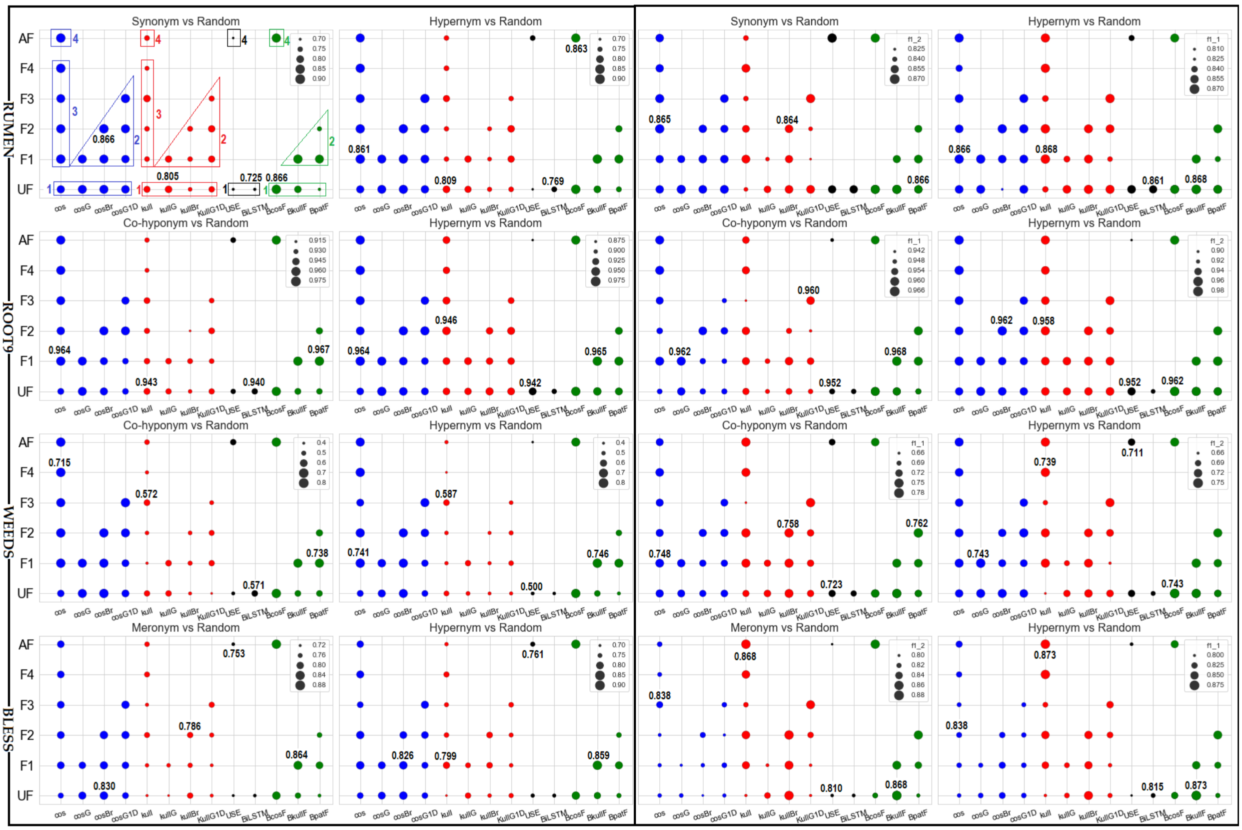

(a) : MLP with all private feature combinations

(b): Shared-Private with all shared feature combinations

Figure 2: F_1 score results for all feature combinations. Bounding box 1 stands for any individual feature alone, e.g. ($cosG$, UF) means only $cosG$. Bounding box 2 stands for any 2-by-2 combination of features, e.g. ($cosBr$,F2) refers to ($cosBr$,$cosG$). Bounding box 3 refers to the ablation of one feature from the set of all features, e.g. (Bounding box 3 Blue, F2) refers to (cos,$cosBr$,$cosG1D$). Bounding box 4 stands for the combination of all features for a given family (AF). $BcosF$, $BKullF$ and $BpatF$ stand for best combination of features within its respective family.

		Synonym vs Random			Hypernym vs Random		
	Algorithm	CosF	KullF	PatF	CosF	KullF	PatF
RUMEN	Best MLP	0 1 1 0	0 0 0 0	0 0	0 1 1 1	1 0 0 0	0 1
	Best FS	0 1 1 0	0 0 0 0	0 0	1 1 1 1	0 0 0 0	0 0
	Best SP	0 0 0 0	0 0 0 0	1 1	0 0 0 0	0 1 1 1	0 0
		Co-hyponymy vs Random			Hypernym vs Random		
	Algorithm	CosF	KullF	PatF	CosF	KullF	PatF
ROOT9	Best MLP	0 1 1 1	0 0 0 0	1 1	0 1 1 1	1 0 1 1	0 0
	Best FS	0 1 1 0	0 0 0 0	0 0	1 1 1 1	0 0 0 0	0 0
	Best SP	0 1 1 0	0 1 1 0	0 0	1 1 1 1	0 0 0 0	1 0
		Co-hyponymy vs Random			Hypernym vs Random		
	Algorithm	CosF	KullF	PatF	CosF	KullF	PatF
WEEDS	Best MLP	1 1 1 0	0 0 0 0	0 1	0 1 1 1	1 1 0 1	0 0
	Best FS	1 1 1 0	0 0 0 0	0 0	0 1 1 1	0 0 0 0	0 0
	Best SP	0 0 0 0	0 1 1 0	1 0	1 1 0 0	0 0 0 0	0 0
		Meronym vs Random			Hypernym vs Random		
	Algorithm	CosF	KullF	PatF	CosF	KullF	PatF
BLESS	Best MLP	0 0 1 0	0 1 1 0	0 0	1 0 1 0	0 1 1 1	0 0
	Best FS	0 0 1 0	1 0 1 1	0 1	0 0 1 0	0 0 0 0	0 1
	Best SP	0 0 0 0	1 1 1 1	0 0	0 0 0 0	1 1 1 1	0 0

Table 4: Best combinations of features for all models. 0 and 1 stand for the absence or the presence, respectively, of the given feature within its family, where the order is given by equations 4, 12 and 14.

show the superiority of the shared-private network (*Best SP*) over the fully-shared model (*Best FS*) for most cases, suggesting that the combination of private and shared information is beneficial to the decision process. However, the *Best MLP* is a hard model to beat as the *Best SP* statistically outperforms the former architecture 4 times out of 8, and 2 times out of 8 without statistical significance. But the contrary is only true for ROOT9 (wrt. F_1 score), where *Best MLP* statistically exceeds *Best SP*.

The important issue in shared-private architectures is to understand how well they distribute the feature space between private and shared layers. For that purpose, we analyse figure 2 (b), which shows feature combinations for the shared layer, i.e. when two tasks are learned concurrently. Note that in this case, best combinations from the private models (learned separately) restrict the learning process. The first main conclusion is that asymmetric distributional features ($KullF$) steadily compete with cosine-based features ($CosF$), even clearly outperforming the latter for BLESS, which is definitely not the case within private models. The same conclusion can be drawn

		Synonym vs Random		Hypernym vs Random	
	Algorithm	GloVe	JoSE	GloVe	JoSE
RUMEN	MLP	0.754	0.730	0.757	0.731
	Best MLP	0.865	**0.870**	0.862	0.863
	Best SP	0.866	0.869⋆	**0.867**⋆	0.865†
		Co-hyponym vs Random		Hypernym vs Random	
	Algorithm	GloVe	JoSE	GloVe	JoSE
ROOT9	MLP	0.939	0.927	0.936	0.922
	Best MLP	0.967	0.967	0.965	0.963
	Best SP	**0.968**⋆	0.966	0.962	**0.966**⋆†
		Co-hyponym vs Random		Hypernym vs Random	
	Algorithm	GloVe	JoSE	GloVe	JoSE
WEEDS	MLP	0.449	0.458	0.457	0.455
	Best MLP	0.737	0.758	0.746	0.754
	Best SP	0.761	**0.764**⋆†	0.743	**0.759**⋆†
		Meronym vs Random		Hypernym vs Random	
	Algorithm	GloVe	JoSE	GloVe	JoSE
BLESS	MLP	0.748	0.810	0.762	0.798
	Best MLP	0.864	0.865	0.859	0.850
	Best SP	**0.868**⋆	0.865	0.873	**0.874**†

Table 5: F_1 scores with GloVe and JoSE. ⋆ and † denote p-value ≤ 0.05 based on the t-Test assuming unequal sample variances of metric values between respectively (Best SP JoSE) vs. (Best SP GloVe) and (Best SP JoSE) vs. (Best MLP JoSE).

for pattern-based features $PatF$, which impact is much more important in the shared layers than it is the case in the private models when compared to $CosF$. This suggests that when private models focus more on symmetric features, shared-private models take advantage of asymmetric features to capture task dissimilarity (indeed in the concurrent tasks there is always at least one asymmetric task).

Another interesting observation is that best models are usually not a combination of different family features. Only 2 cases out of 8 show improved results with feature combination. In fact, such results suggest that private and shared layers distinctively balance the family feature space. We clearly see this situation in Table 4 by looking at the complementarity of the input feature vectors of private (row 1) and shared-private models (row 3). For instance, when maximizing the hypernymy task within the shared-private model over RUMEN, the private input vectors are $(cosG, cosBr, cosG1D, kull, \overline{pat}_{*,\text{BiLSTM}})$ for hypernymy and $(cosG, cosBr)$ for synonymy, while the shared input vector is $(kullG, kullBr, kullG1D)$. It is worth noticing that this situation does not hold for the fully-shared models as they are clearly biased towards cosine-based metrics and rarely include asymmetric distributional and pattern-based features.

6.3 Spherical text embeddings

We propose to compare our feature-based architectures with relational embeddings, namely JoSE (Meng et al., 2019), the underlying idea being to understand how feature-based strategies can compare and eventually add-on to fine-tuned neural semantic spaces. Results are illustrated in Table 5.

Results of the baseline *MLP* model do not evidence a clear advantage of relational embeddings compared to general-purpose ones like GloVe, BLESS being the only exception. However, it is interesting to notice that the proportion of improvement is much more important for JoSE embeddings when introducing combinations of features. Indeed, while the *MLP* model with GloVe overtakes the JoSE version 5 times out of 8, the *Best MLP* model with JoSE overtakes the GloVe version 5 times out of 8, thus suggesting that spherical embeddings are sensitive to feature engineering.

Finally, while shared-private architectures provide overall best results, a clear distinction between both embeddings is difficult to establish, although a small tendency towards JoSE embeddings seems to emerge. Indeed, while the hypernymy relation is better tackled by relational embeddings (3 out of 4 configurations), meronymy is better handled by GloVe although being an asymmetric relation. With respect to symmetric relations (synonymy and co-hyponymy), the situation slightly converges towards relational embeddings with better results in 2 out of 3 experiments.

7 Conclusions

In this paper, we proposed the definition of asymmetric distributional features in continuous spaces based on the Kullback-Leibler divergence, and suggested to combine them with families of symmetric distributional and pattern-based characteristics using a feature selection process. We proposed to analyse the impact of feature combination in multi-task settings, which combine private and shared layers. Results evidenced the benefits of feature combination in the private models, and they highlighted the importance of asymmetric (distributional and paradigmatic) features in the shared layers. Moreover, share-private architectures showed the capacity of balancing feature families between private and shared layers thus taking full advantage of most features in the decision process.

References

Muhammad Asif Ali, Yifang Sun, Xiaoling Zhou, Wei Wang, and Xiang Zhao. 2019. Antonym-synonym classification based on new sub-space embeddings.

In *33rd AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 6204–6211.

Mohammed Attia, Suraj Maharjan, Younes Samih, Laura Kallmeyer, and Thamar Solorio. 2016. Cogalex-v shared task: Ghhh - detecting semantic relations via word embeddings. In *Workshop on Cognitive Aspects of the Lexicon*, pages 86–91.

Georgios Balikas, Gaël Dias, Rumen Moraliyski, Houssam Akhmouch, and Massih-Reza Amini. 2019. Learning lexical-semantic relations using intuitive cognitive links. In *41st European Conference on Information Retrieval (ECIR)*, pages 3–18.

Nesrine Bannour, Gaël Dias, Youssef Chahir, and Houssam Akhmouch. 2020. Patch-based identification of lexical semantic relations. In *42nd European Conference on Information Retrieval (ECIR)*, pages 126–140.

Oren Barkan. 2017. Bayesian neural word embedding. In *31st AAAI Conference on Artificial Intelligence*, volume 31.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung chieh Shan. 2012. Entailment above the word level in distributional semantics. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics (GEMS)*, pages 1–10.

Giulia Benotto. 2015. *Distributional Models for Semantic Relations: A Sudy on Hyponymy and Antonymy*. Ph.D. thesis, University of Pisa.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from bert. In *34th AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 7456–7463.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, page 4171–4186.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 886–897.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. 2015. Combining distributed vector representations for words. In *1st workshop on vector space modeling for natural language processing*, pages 95–101.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1758–1768.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *13th International Conference on Artificial Intelligence and Statistics (AISTAT)*, pages 249–256.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *14th Conference on Computational Linguistics (COLING)*, pages 539–545.

Abhik Jana, Nikhil Reddy Varimalla, and Pawan Goyal. 2020. Using distributional thesaurus embedding for co-hyponymy detection. In *12th Language Resources and Evaluation Conference (LREC)*, pages 5766–5771.

Aishwarya Kamath, Jonas Pfeiffer, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019. Specializing distributional vectors of all words for lexical entailment. In *4th Workshop on Representation Learning for NLP (RepL4NLP)*, pages 72–83.

Neha Kathuria, Kanika Mittal, and Anusha Chhabra. 2017. A comprehensive survey on query expansion techniques, their issues and challenges. *International Journal of Computer Applications*, 168(12).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ron Kohavi and Dan Sommerfield. 1995. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *1st International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 192–197.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 970–976.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance M. Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8206–8215.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017a. Hierarchical embeddings for hypernymy detection and directionality. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017b. Distinguishing antonyms and synonyms in a pattern-based neural network. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 76–85.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1532–1543.

Marek Rei, Daniela Gerz, and Ivan Vulić. 2018. Scoring lexical entailment with a supervised directional similarity network. In *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 638–643.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *25th International Conference on Computational Linguistics (COLING)*, pages 1025–1036.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. In *10th International Conference on Language Resources and Evaluation (LREC)*, pages 4557–4564.

Enrico Santus, Vered Shwartz, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 65–75.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *4th Workshop on Linked Data in Linguistics (LDL) associated to Association for Computational Linguistics and Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 64–69.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. In *5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 24–29.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2389–2398.

Tu Vu and Vered Shwartz. 2018. Integrating multiplicative features into supervised distributional methods for lexical entailment. In *7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 160–166.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1134–1145.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1671–1682.

Chengyu Wang and Xiaofeng He. 2020. BiRRE: Learning bidirectional residual relation embeddings for supervised hypernymy detection. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3630–3640.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *5th International Conference on Computational Linguistics (COLING)*, pages 2249–2259.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *20th International Conference on Computational Linguistics (COLING)*, pages 1015–1021.

Don't Miss the Labels: Label-semantic Augmented Meta-Learner for Few-Shot Text Classification

Qiaoyang Luo, Lingqiao Liu[*]**, Yuhao Lin** and **Wei Zhang**
The Univeristy of Adelaide
`{qiaoyang.luo, lingqiao.liu, wei.e.zhang}@adelaide.edu.au`
`linyuhao0514@gmail.com`

Abstract

Increasing studies leverage pre-trained language models and meta-learning frameworks to solve few-shot text classification problems. Most of the current studies focus on building a meta-learner from the information of input texts but ignore abundant semantic information beneath class labels. In this work, we show that class-label information can be utilized for extracting more discriminative feature representation of the input text from a pre-trained language model like BERT, and can achieve a performance boost when the samples are scarce. Building on top of this discovery, we propose a framework called Label-semantic augmented meta-learner (LaSAML) to make full use of label semantics. We systematically investigate various factors in this framework and show that it can be plugged into the existing few-shot text classification system. Through extensive experiments, we demonstrate that the few-shot text classification system upgraded by LaSAML can lead to significant performance improvement over its original counterparts.

Banking
Class 1: *can you give me a hand paying my water bill*
Class 2: *tell me the last day I can pay my gas bill*
- -
Travel
Class 3: *what are some fun activities to do in colorado*
Class 4: *tell me about any travel alerts issued for germany*

Figure 1: The example of fine-grained intent queries from two domains in Clinc150 dataset. Class 1 and Class 2 refer to PAY BILL and BILL DUE. Class 3 and Class 4 represent TRAVEL SUGGESTION and TRAVEL ALERT.

1 Introduction

The remarkable capability of quickly learning new concepts from a few training samples is one of the advantages of the human learning system over the current machine learning system. Motivated by this gap, research in few-shot learning has received increasing attention in the past decade. Meta-learning (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017), as the dominant methodology in few-shot learning, tackles the problem by learning a mapping function from a few support samples to a classifier through a meta-training dataset. Most existing meta-learning systems (Snell et al., 2017; Sung et al., 2018) were developed or at least evaluated in the field of computer vision. More recently, few-shot learning has been introduced to the NLP field and in particular, text classification (Yu et al., 2018; Geng et al., 2019), as it is the fundamental task in natural language understanding. In parallel to few-shot learning, pre-trained language models (PLMs) (Devlin et al., 2019; Radford et al., 2019) have revolutionized the NLP fields and show strong evidence of being able to perform well in low data regime when transferred to downstream tasks.

Despite the impressive progress of meta-learning and PLMs, however, most existing few-shot classification systems (Geng et al., 2019; Bao et al., 2020) ignore an important information source — semantic of class labels. When the number of training samples is limited, merely using the input texts per class can lead to ambiguity in interpreting the definition of class. Considering the two groups of examples in Figure 1 which shows four samples belonging to different intent classes, even humans cannot fully understand the semantic meaning of those samples if the definition of labels are not given. For example, it is hard to tell if class 1 and class 2 are about the type of bill — water or gas, or class 3 and class 4 are about the destination of the travel — USA or Germany. However, this ambiguity can be easily resolved if the class definition or simply the class name is provided.

---

[*] Corresponding Author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2773–2782
August 1–6, 2021.

Just as understanding class names can help humans to interpret sentences of a given class, we made an interesting observation that the BERT will extract more discriminative features if we append the class name to the input sentence, and it can boost the classification performance in low-shot scenarios. Motivated by the above observations, this work explores how to better leverage the semantic information beneath class names for few-shot learning. Our key idea is to use meta-learning to further strengthen the guidance of class-label semantics for few-shot classification. Specifically, we use meta-learning to encourage the features extracted from class-name-appended samples to be more class-relevant and compatible to the query features. Moreover, we systematically study the issue of how to extract the label-semantic guided feature representation from the support samples and how to make the query sample features compatible with the meta-learner generated from the support set. Our research leads to a framework that can be plugged into the existing few-shot meta-learner and we call our method Label-semantic Augmented Meta-Learner (LaSAML). To demonstrate the power of LaSAML, we use LaSAML to upgrade the Prototypical Network and creates a new method called LaSAML-PN. By conducting the extensive experimental studies, we show that LaSAML-PN achieves excellent few-shot learning performance and LaSAML upgraded meta-learning obtains superior performance over its original counterpart. Our code has been released at: `https://github.com/luoqiaoyang/ACL2021-LaSAML`.

2 Related work

This section discusses the related work from three aspects: few-shot learning, few-shot text classification, and low-shot learning with label information.

Few-Shot learning Meta-learning approaches have made substantial progress with few-shot learning (FSL) tasks. The focus of the current meta-learning framework is how to construct the meta-learner. For examples, a meta-learner could be constructed by learning a metric between samples and classes(Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018), based on a differentiable learning process (Bao et al., 2020), or based on a few-shot gradient update (Mishra et al., 2018; Finn et al., 2017). A complete review of meta-learning is beyond the scope of this paper, and we refer readers to the recent survey (Hospedales et al., 2020).

Few-shot text classification Few-shot text classification (FSTC) has also gained increasing attention in recent years. ROBUSTTC-FSL (Yu et al., 2018) uses an adaptive metric learning approach to adaptively select an optimal distance metric for different tasks. Induction Network (Geng et al., 2019) utilizes the dynamic routing algorithm (Sabour et al., 2017) to learn a generalized class-wise representation. Pre-trained language models have also been applied to few-shot text classification. LEOPARD (Bansal et al., 2020) uses BERT (Devlin et al., 2019) with optimization-based meta-learning framework to achieve good performance on diverse NLP classification tasks. More recently, GPT-3 (Brown et al., 2020) shows that the language model itself can be used to perform few-shot text classification without using meta-learning. Meanwhile, another recent work (Bao et al., 2020) points out meta-learning for text classification may have different characteristics to the cases in computer vision. They propose to use distributional signatures to enhance the generalization capability of meta-learner. Our method is still a meta-learning-based few-shot text classification method. The key contribution of our work is the discovery that using label information together with BERT can lead to significantly better generalization performance.

Using label information for text classification An increasing number of recent works have realized the value of label semantics. The matching between label information and text can naturally lead to zero-shot learning. For example, CDSSM (Chen et al., 2016) explores zero-shot intent classifications based on class names. Prompt-based strategies (Puri and Catanzaro, 2019; Schick and Schütze, 2020) have been developed to implicitly match text against class names. In the context of few-shot learning, (Hou et al., 2020) incorporate label semantics into the TapNet (Yoon et al., 2019) for few-shot slot tagging tasks. Different from the above works, this paper explores both pre-trained language models and label semantics for few-shot learning. We only require the name of classes rather than manually constructed prompts or templates to convey label semantics. TARS (Halder et al., 2020) also leverages pre-trained language models and label semantics based on binary text classification. However, our method further strengthens generalization ability via meta-learning framework especially in cross-domain and fine-grained cases.

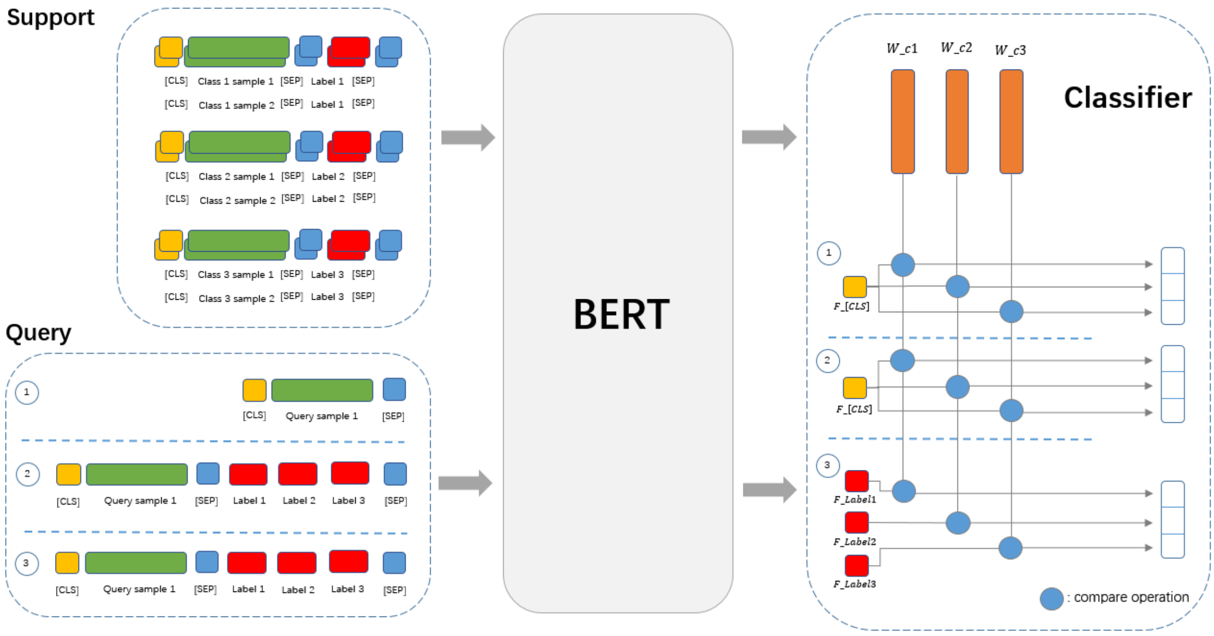

Figure 2: The main framework of LaSAML.This graph gives an example on how LaSAML process a 3-way 2-shot few-shot task. All support samples append with the corresponding class name to format as "[CLS] + Sentence + [SEP] + Label + [SEP]". The query data can be one of three forms: ① remains original form while ② and ③ append all three class names. ① and ② use BERT emebeddings from [CLS] token, ③ uses all label embeddings.

Appending Word	Number of training samples per class	
	5	10
None	0.6425	0.7324
class name	0.7437	0.7925

Table 1: Results of fine-tuning BERT with only 5 or 10 labeled data per class on the AGNews dataset. None: the standard input format for the BERT classifier, class name: appending the respective class names for each training sample.

3 Our method

3.1 The value of label information in the low data regime

As described in the introduction, label information is essential for human to accurately interpret the meaning conveyed in the limited number of training samples. In this section, we demonstrate that label information is also useful for extracting discriminative features from a pre-trained language model [1]. More specifically, we consider the following modification to input of BERT for text classification: we append the corresponding class name after each training sentence (for which we know the ground-truth classes) and a [SEP] token. In other words, we use the following input format "[CLS] sentence [SEP] class name [SEP]" rather than "[CLS] sentence [SEP]" as the common practice of using BERT for text classification. Then we extract embeddings from the [CLS] token to train a linear classifier. The reason of appending "class name [SEP]" is to mimic the scenario of the next sentence prediction (NSP) task for training BERT. To perform well in the NSP task, BERT needs to extract information that are most predictive for the next sentence from the first sentence. In our case, we replace the next sentence with the class name and consequently, we expect that BERT can extract information that is relevant to the class name from the input sentence. We call this method label-semantic augmented feature extraction hereafter.

From the experimental results are shown in Table 1, we can clearly see that the classifier trained from label-semantic augmented feature extraction achieves better performance than the baseline approach. When only five samples are used per class, the improvement can be as significant as 10%.

From this motivating experiment, we clearly see the potential of incorporating class-label information. To further strengthen BERT's capability of leveraging class-label semantic information, we incorporate the above idea into the meta-learning framework, lending itself to a new meta-learning framework termed label-semantic augmented meta-

---

[1] Throughout our experiments, we use BERT (Devlin et al., 2019) as the PLM unless specified otherwise.

learner (LaSAML). We expect that through fine-tuning a PLM by the meta-learning process, the network can find an optimal way of building meta-classifiers with the guidance of class-labels.

3.2 The general framework of the label-semantic augmented meta-leaner

We first present the proposed LaSAML in its general form and then dive into more details of this framework. Formally, we consider the following problem setting. Our aim is to build a meta-learner which can convert a set of support samples, denoted as $\mathcal{X}_s = \{x_s, y_s, t_s\}$, into a classifier $\phi(\cdot; \mathcal{X}_s)$, where x_s, y_s and t_s denote the input text, class label, and the lexical definition of the class, i.e., the class name, respectively [2] Applying $\phi(\cdot; \mathcal{X}_s)$ to test data, i.e., query data x_q, we could obtain the predicted class $\hat{y}_q$ through $\phi(x_q; \mathcal{X}_s)$. The meta-learner is trained from the meta-training set, from which one can randomly construct a support set $\mathcal{X}_s^C = \{x_s^c, y_s^c, t_s^c\}$ and a query set with ground-truth class name, $\mathcal{X}_q^C = \{x_q^c, y_q^c\}$ for C-Way K-shot settings, where $c \in C$. Therefore, the performance of $\phi(\cdot; \mathcal{X}_s^C)$, classifier generated from the meta-learner, can be evaluated by comparing the predicted class against the ground-truth query label $\{y_q^c\}$. The key difference of traditional meta-learner and the proposed LaSAML is that the lexical definition of class name $\{t_s^c\}$ will be used for building the meta-learner.

In particular, we consider the meta-learner that can be written in the following form:

$$\{\mathbf{w}_c\} = \psi(f(\{x_s^c\}), \{y_s^c\}) \quad c = 1, \cdots, C$$
$$q_c = m(g(x_q); \mathbf{w}_c), \quad \hat{y} = \underset{c}{argmax}\, q_c, \qquad (1)$$

where f and g denotes the feature extractors which convert the input text to a feature vector. For many meta-learning approaches, $f = g$. ψ is a mapping function to map the support set data to a set of class vectors, one for each class. Then the classifier is defined by a function $m(\cdot, \cdot)$ that measures the compatibility, q_c, between a query sample x_q and the class vector $\mathbf{w}_c$. The class with highest q_c is the predicted class.

The above formulation encompasses a wide range of meta-learning approaches. For example, for Prototypical network (Snell et al., 2017), $\mathbf{w}_c$ is the c-th class mean vector calculated from the feature extractor and $m(\cdot, \cdot)$ is simply a negative Euclidean distance between $g(x_q)$ and $\mathbf{w}_c$.

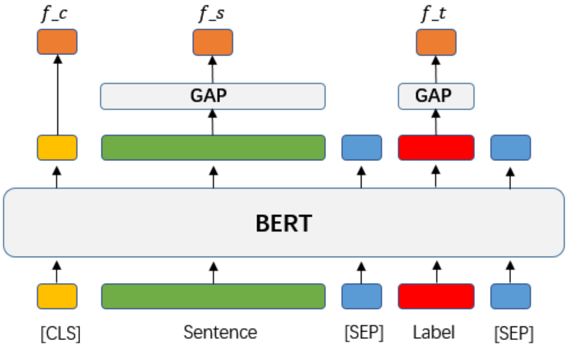

Figure 3: This graph shows how we extract support sample features from various positions. GAP refers to the global average pooling operation.

The proposed LaSAML introduces label-semantic guidance to $f(\cdot)$ and $g(\cdot)$. In other words, the feature extractors f and g may take the class name as an additional input. Due to that the availability of class name information will be different for support set samples and query set samples, i.e., we know the ground-truth class name for support set samples but not for query samples, we may choose different ways of incorporating label information into the feature extractor, resulting in different implementation of f and g.

3.3 Incorporating label information into feature extractors

This subsection discusses various options of incorporating label information into f and g. We show the possible configurations in Figure 3. For the feature extractor of the support set, f, we append the corresponding ground-truth class name to each sentence. Then we have different options of extracting sentence feature representations. In our study, we consider extracting sentence features from [CLS] token, the global average pooling (GAP) of embeddings of sentence tokens, GAP of embeddings of the class name (since a class name may contain multiple tokens), and the average of them.

For the feature extractor of the query set, g. We consider three cases. First, the most straightforward way is not appending anything since we do not know the ground-truth class name for query samples. Second, we can append all class names, as shown in Figure 2. Finally, we can append all class names but extract C features from the corresponding class name, one for each class. Then the c-th feature will be compared against the c-th

---

[2] In our following discussion, we slightly relax the distinction between "class label", "class name" and "class tag" and use them exchangeably when no confusion is caused.

class vector and calculate the matching score with m. The class corresponding to the highest matching score will be the prediction. This scheme is visualized in Figure 2 option 3 for query. Formally, this process can be written as:

$$\hat{y}_q = \underset{c}{argmax}\, m(g(x_q, t_c), w_c), \tag{2}$$

where $g(x_q, t_c)$ denote the feature extracted from class name token t_c.

We will leave the detailed comparison results and discussion of those schemes to Section 4.2 and Section 4.3. Here, we report our major discovery. (1) For supporting set samples, extracting sentence features from different positions leads to similar performance. Extracting features from "[CLS]" and "[CLS]+Tag" in general is slightly better than other options. (2) For query samples, without appending all class-label names leads to the overall best performance for our best performed method. In the following, we by default consider the setting of extracting features from "[CLS]" and not appending class names to a query sample unless otherwise specified.

3.4 Upgrade existing meta-learner with LaSAML

The proposed LaSAML can be incorporated into a variety of existing meta-learning frameworks. In our study, we mainly consider Prototypical Network (Snell et al., 2017) as the meta-learning framework and upgrade it with LaSAML.

The Prototypical Network (Snell et al., 2017) is a metric-based meta-learning framework, which calculates the class vector by averaging the same-class features extracted from the support set. In its LaSAML-upgraded version (denoted as LaSAML-PN), we calculate the class vector $\mathbf{w}_c$ by

$$\mathbf{w}_c = \frac{1}{|\mathcal{X}_s^c|} \sum_{(x_s^c, t_s^c) \in \mathcal{X}_s^c} f(x_s^c, t_s^c), \tag{3}$$

where $f(x_s^c, t_s^c)$ indicates the feature extracted by incorporating class name information.

Then, we make the decision by comparing the feature extracted from a query sample against $\{\mathbf{w}_c\}$:

$$P(c|x_q) = \frac{\exp(-d(g(x_q^c), \mathbf{w}_c))}{\sum_{c' \in C} \exp(-d(g(x_q^c), \mathbf{w}_{c'}))} \tag{4}$$

where $d(\cdot, \cdot)$ is the squared Euclidean distance.

4 Experimental results

In this section, we conduct experiments to evaluate the performance of LaSAML. We first introduce our experimental setting. Then, we present the main results by comparing LaSAML against various existing few-shot text classification approaches. Finally, we provide ablation studies to investigate multiple factors in the proposed method.

4.1 Experimental Setting

4.1.1 Datasets

Three text classification datasets are used in our experiment.

HuffPost is a dataset including a wide range of news topics. The dataset consists of 36900 news headline samples and 900 samples for each class. Following the settings from (Bao et al., 2020), we use the same 20/5/16 classes for training, validation, and testing, respectively, for a fair comparison. Due to the limited number of classes, we only consider the 5-way 1-shot and 5-way 5-shot text classification tasks in this dataset.

Banking77 published by (Casanueva et al., 2020) is a dataset for intent classification tasks. The dataset covers 13,083 fine-grained intents from 77 classes in the banking domain. We construct the few-shot tasks in 10-way 1-shot, 10-way 5-shot, 15-way 1-shot, and 15-way 5-short. The dataset is partitioned into a training, a validation, and a testing dataset. 30, 15, and 32 classes are sampled for each partition[3].

Clinc150 is a cross-domain intent classification dataset which was originally proposed in (Larson et al., 2019). It provides 22,500 in-scope queries and 150 intent classes from 10 domains. Each domain contains 15 intent classes, and there is no overlap between those classes. We use this dataset to evaluate the performance of meta-learner under domain shift. We split the datasets into 4/1/5 domains for training, validation, and testing.

4.1.2 Comparing methods

We compare the proposed method against several commonly used few-shot learning approaches, which have shown promising results in both computer vision and natural language processing fields. For all the compared methods expect distributional signature (Bao et al., 2020) which shows better performance without BERT, we re-implement them

---

[3] We released the partition of Banking77 and Clinc150 along with our code.

Models	HuffPost		Banking77				Clinc150 (cross domain)			
	5-way		10-way		15-way		10-way		15-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
BERT+PN*	0.4059	0.5348	0.6305	0.7860	0.5918	0.7412	0.5743	0.7290	0.5231	0.6606
BERT+PN	0.4611	0.6556	0.7622	**0.8883**	0.7028	**0.8582**	0.7130	0.8798	0.6303	0.8163
BERT+RN	0.4080	0.5187	0.6388	0.7348	0.5629	0.6457	0.5465	0.6009	0.4654	0.5883
BERT+IN	0.3996	0.5079	0.4872	0.6432	0.4945	0.5527	0.4652	0.5765	0.4172	0.4998
BERT+RRML	0.4078	0.6198	0.7045	0.8780	0.6346	0.8565	0.6272	0.8713	0.5761	0.8076
DS+RRML	0.4134	0.6248	0.5933	0.8371	0.5337	0.7896	0.5556	0.7876	0.5341	0.7969
LaSAML-PN	**0.6216**	**0.7011**	**0.8278**	0.8806	**0.7877**	0.8443	**0.7760**	**0.8831**	**0.7248**	**0.8489**

Table 2: Experiment results of 5-way 1-shot and 5-way 5-shot on HuffPost Dataset, 10-way 1-shot, 10-way 5shot, 15-way 1-shot and 15-way 5-shot on Banking77 and Clinc150 (cross domain) datasets. The model BERT+PN* contains MLP in PN to process BERT embeddings before applying the distance metric.

Support Features	LaSAML-PN				LaSAML-RRML			
	HuffPost		Clinc150		HuffPost		Clinc150	
	5-Way		15-Way		5-Way		15-Way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
CLS	0.6216	**0.7011**	0.7248	**0.8489**	0.6260	0.6808	0.6201	0.8051
Sent	0.6223	0.6713	0.6834	0.8359	0.6095	0.6781	0.5811	0.7342
Tag	0.6264	0.6758	0.6936	0.8199	0.5960	0.6358	0.6470	0.7604
CLS + Sent	0.6250	0.6998	0.6955	0.8303	**0.6286**	**0.6927**	0.6160	**0.8064**
CLS + Tag	**0.6325**	0.6946	**0.7348**	0.8275	0.6055	0.627	0.6485	0.7952
Sent + Tag	0.6289	0.6964	0.6974	0.8364	0.609	0.6488	0.6532	0.7738
Weighted All	0.6211	0.7003	0.6996	0.8245	0.6022	0.6448	**0.6599**	0.7938

Table 3: Ablation study results of extracting support data features from varies positions and its combinations on HuffPost and Clinc150 (cross domain). According to the results on another ablation study in Table 5, we pick up different query settings ① and ② in Figure 2 for LaSAML-PN and LaSAML-RRML individually.

with the BERT encoder as the feature extractor. Note that this might lead to different (in most cases higher) performance than the one originally reported.

Prototypical Network (Snell et al., 2017) is a metric-based few-shot learning method. Our LaSAML-PN is an upgraded version of it. We use two implementations of PN. One extracts features from the [CLS] token, and another applies a multi-layer-perceptron (MLP) for the embedding of [CLS] token. The latter was used in a recent study (Bao et al., 2020). We denote the original implementation of Prototypical Network as PN and the implementation with an MLP as PN*.

Relation Network (Sung et al., 2018) (RN) does not directly compare class vector against the query feature but compare through a relation module learned during meta-training.

Induction Network (Geng et al., 2019) (IN) integrates the dynamic routing algorithm into the relation network to learn the class vectors during the meta-learning process. The original induction network uses LSTMs as the encoder. To make a fair comparison, we replace the encoder with the BERT encoder in our experiment.

Ridge Regression Meta-Learner (Bertinetto et al., 2019) (RRML) calculates the class vector by solving a Ridge regression problem on the support set. The solution has a closed-form, and thus it is possible to directly back-propagate training error to the feature extractor used in the optimization problem.

Distributional Signature (Bao et al., 2020) (DS) learns how to use the statistic pattern of tokens to selectively attend key information of the input text and build a meta-learner with better generalization performance. The method in (Bao et al., 2020) can be applied to a wide variety of meta-learning methods. In (Bao et al., 2020), the combination with RRML shows the best performance (DS+RRML).

4.1.3 Implementation Details

In our methods, $\text{BERT}_{\text{BASE}}$ is employed as the feature encoder and meta-learner. We construct 100, 100, and 1000 random sampled tasks for each training, validation, and testing epoch individually.

Moreover, we use the Adam algorithm (Kingma and Ba, 2015) as the optimizer. For better training performance, we set different learning rates for the BERT encoder and the other modules, that is, $2e{-5}$ for the BERT encoder and $1e{-3}$ for other modules. Both Relation Network and Induction Network consist of a relation module, and we set the dense hidden layer dimension to 50 for the relation module. We follow other settings of Induction Network in the original paper (Geng et al., 2019).

4.2 Main results of LaSAML

The main experiment results are displayed in Table 1. From the result, we make the following observations: (1) The proposed LaSAML-PN achieves significant performance improvement over the original PN, especially on the one-shot classification setting: on average, the improvement is around 6% to 16%. With more training data, the gap between LaSAML-PN and PN becomes smaller: on Banking77, LaSAML-PN and PN become comparable; but we can still see 3-5% improvement on Clinc150 and HuffPost. This is understandable because, with more samples, the class-related text patterns become more pronounced. However, this might be data-dependent. In general, if the difference between classes is more subtle, i.e., fine-grained classes, more samples might be needed and consequently, the guidance from class name/definition will be more beneficial. (2) We find that the original implementation of the Prototypical network performs much better than the one used in (Bao et al., 2020) which employs an additional MLP. The former achieves even higher performance than the method proposed in (Bao et al., 2020) (which is DS+R2D2. Our re-implementation achieves almost the same performance in (Bao et al., 2020)). (3) Another surprising finding is that the Relation Network and Induction network do not perform better than the traditional Prototypical network. From the above observations, we may conclude that using modules without prior information of a language model, e.g., the MLP whose parameters are randomly initialized rather than pre-trained as in the PLM, leads to poor generalization performance. In contrast, methods directly fine-tuning parameters inside a PLM, e.g., PN, RRML, and our methods, tend to perform better. This observation can somehow be supported by the argument in (Bao et al., 2020). In (Bao et al., 2020), it points out that in NLP, “the lexical features highly informative for one task may be insignificant for another. ” Thus, the weight learned from those randomly initialized modules may overfit the meta-training set and cannot generalize well to the target task. In (Bao et al., 2020), the authors suggest a solution by building a meta-learner with the generalizable statistics of words. In our study, we find that this solution might not be stable in all cases. For example, DS+RRML does not perform well in Banking77 and Clinc150. Instead, our results suggest an alternative solution: building the meta-learner by not introducing additional parameters to BERT parameters, since the latter is pre-trained from a large corpus and tends to generalize better across tasks.

Model	HuffPost		Clinc150	
	5-way		15-way	
	1-shot	5-shot	1-shot	5-shot
BERT+PN	0.4611	0.6556	0.6303	0.8163
LaSAML-PN	**0.6216**	**0.7011**	**0.7248**	**0.8489**
BERT+RRML	0.4078	0.6198	0.5761	**0.8076**
LaSAML-RRML	**0.6260**	**0.6808**	**0.6201**	0.8051

Table 4: Ablation study results for integrating LaSAML with RRML and testing performance on HuffPost and Clinc150 (cross domain).

4.3 Ablation Study

In this section, We investigate LaSAML in depth by answering three questions. First, whether LaSAML is applicable to other meta-learning frameworks. Second, what is the impact of different ways of extracting features from BERT for support set samples? Third, how to leverage class name information for query samples? We conduct a serial of experiments on HuffPost and Clinc150 datasets to answer those questions.

LaSAML with other meta-learning framework To further explore the potential of LaSAML, we incorporate it into the Ridge Regression Meta-learner (RRML) (Bertinetto et al., 2019), which is achieved by simply replacing the feature extractor f and g with the feature extractors used in LaSAML-PN. The results are shown in Table4. As seen, using LaSAML leads to significant improvement in one-shot cases for the RRML. For five-shot cases, the improvement gain becomes smaller for Clinic-150 but still significant in HuffPost. This experiment result suggests that the proposed LaSAML has a great potential to upgrade a wide variety of meta-learning approaches.

Comparing support set feature extraction strategies In LaSAML, the input format of a sup-

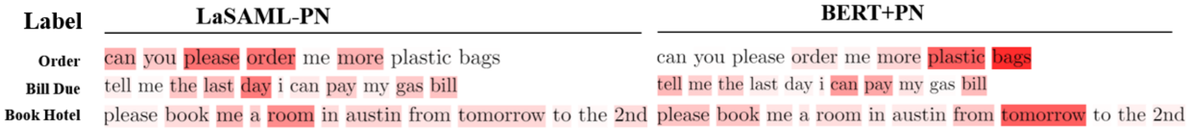

Figure 4: Visualization of BERT attention maps for LaSAML-PN and PN. The darker red color refers to higher attention weight.

port set sample is "[CLS] sentence [SEP] class name [SEP]". As mentioned in Section 3.3, we may extract sentence features from the last layer embedding of the [CLS] token, the average embedding of sentence tokens, the embedding (average embedding) of the class name, and various combination of them. Table 3 provides the comparison of the results. From the results, we can see that no single strategy achieves consistently better performance than the others. Their performance, in most cases, is also similar. Therefore, we extract the sentence feature from the last layer embedding of the [CLS] token for simplicity.

Comparing query feature extraction strategies In this section, we further investigate the impact of the input format for query samples. Three configurations, not appending class names and extracting features from [CLS] (L: None F: CLS), appending all class names but extracting features from [CLS] (L: All F: CLS), appending all class names but extracting features from the respective class (L: All F: Tag), and make a prediction by using Eq. 2. We also make our comparison with the LaSAML upgraded PN, or LaSAML-RRML. The experiment results are shown in Table 5. From the results, we can see that the best strategy seems to be method dependent. Appending all class names leads to better performance for LaSAML-RRML, but for LaSAML-PN, the best strategy is not appending any class names. Another observation is that extracting features from the respective class tag and comparing them against the respective class vector may lead to worse performance. However, extracting features from the respective class tag is capable of achieving better performance (or comparable performance on 5-shot classification in Clinc150 dataset) than previous state-of-the-art methods.

			HuffPost		**Clinc150**	
			5-way 1-shot	5-way 5-shot	15-way 1-shot	15-way 5-shot
M	**L**	**F**				
LaSAML-PN	None	CLS	0.6216	**0.7011**	**0.7248**	**0.8489**
LaSAML-PN	All	CLS	0.6291	0.6726	0.6962	0.8423
LaSAML-PN	All	Tag	**0.6365**	0.6560	0.6680	0.7877
LaSAML-RRML	None	CLS	**0.6284**	0.6631	0.5751	0.7834
LaSAML-RRML	All	CLS	0.6260	**0.6808**	**0.6201**	**0.8051**
LaSAML-RRML	All	Tag	0.5814	0.6782	0.5599	0.7897

Table 5: Ablation study results for process query data in three different ways. M refers to the models including: prototypical network, relation network and ridge regression classifier. L refers to whether query samples append all class names or none. F refers to the features used. Here, support data append related class name and use [CLS] token features.

5 What has been learned in LaSAML

In this section, we demonstrate what has been learned by LaSAML-PN. We use an example in Figure 4 to highlight the difference between LaSAML-PN and the standard prototypical network. By investigating the attention weight with respect to the [CLS] token (we average the attention value across all heads in the last layer of BERT), we can see that the prototypical network fails to attend the words relevant to the class. In contrast, LaSAML-PN successfully attends the relevant keywords.

6 Conclusion

In this paper, we systematically study the potential of using class name information for few-shot text classification tasks. We identify that appending the class name to the sentence as the input to a BERT encoder can lead to more discriminative sentence features. By adopting this scheme to meta-training, we propose a new meta-learning framework called LaSAML. Implementing this framework with the Prototypical network (Snell et al., 2017), we achieve significant improvement over the existing few-shot text classification methods.

References

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. 2019. Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 6045–6049. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3904–3913, Hong Kong, China. Association for Computational Linguistics.

Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*

30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 3856–3866.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few-shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1199–1208. IEEE Computer Society.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7115–7123. PMLR.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215, New Orleans, Louisiana. Association for Computational Linguistics.

Detecting Harmful Memes and Their Targets

Shraman Pramanik[1], **Dimiter Dimitrov**[2], **Rituparna Mukherjee**[1], **Shivam Sharma**[1,3],
Md. Shad Akhtar[1], **Preslav Nakov**[4], **Tanmoy Chakraborty**[1]
[1]Indraprastha Institute of Information Technology - Delhi, India
[2]Sofia University, Bulgaria
[3]Wipro AI Labs, India
[4]Qatar Computing Research Institute, HBKU, Doha, Qatar
`{shramanp, shivams, shad.akhtar, tanmoy}@iiitd.ac.in`
`mitko.bg.ss@gmail.com, ritumukherjee23@gmail.com, pnakov@hbku.edu.qa`

Abstract

Among the various modes of communication in social media, the use of Internet memes has emerged as a powerful means to convey political, psychological, and socio-cultural opinions. Although memes are typically humorous in nature, recent days have witnessed a proliferation of *harmful memes* targeted to abuse various social entities. As most harmful memes are highly satirical and abstruse without appropriate contexts, off-the-shelf multimodal models may not be adequate to understand their underlying semantics. In this work, we propose two novel problem formulations: *detecting harmful memes* and *the social entities that these harmful memes target*. To this end, we present HarMeme, the first benchmark dataset, containing $3,544$ memes related to COVID-19. Each meme went through a rigorous two-stage annotation process. In the first stage, we labeled a meme as *very harmful*, *partially harmful*, or *harmless*; in the second stage, we further annotated the type of target(s) that each harmful meme points to: *individual*, *organization*, *community*, or *society/general public/other*. The evaluation results using ten unimodal and multimodal models highlight the importance of using multimodal signals for both tasks. We further discuss the limitations of these models and we argue that more research is needed to address these problems.

1 Introduction

The growing popularity of social media has led to the rise of multimodal content as a way to express ideas and emotions. As a result, a brand new type of message was born: *meme*. A meme is typically formed by an image and a short piece of text on top of it, embedded as part of the image. Memes are typically innocent and designed to look funny.

Over time, memes started being used for harmful purposes in the context of contemporary political and socio-cultural events, targeting individuals, groups, businesses, and society as a whole. At the same time, their multimodal nature and often camouflaged semantics make their analysis highly challenging (Sabat et al., 2019).

Meme analysis. The proliferation of memes online and their increasing importance have led to a growing body of research on meme analysis (Sharma et al., 2020a; Reis et al., 2020; Pramanick et al., 2021). It has also been shown that off-the-shelf multimodal tools may be inadequate to unfold the underlying semantics of a meme as (*i*) memes are often context-dependent, (*ii*) the visual and the textual content are often uncorrelated, and (*iii*) meme images are mostly morphed, and the embedded text is sometimes hard to extract using standard OCR tools (Bonheme and Grzes, 2020).

The dark side of memes. Recently, there has been a lot of effort to explore the dark side of memes, e.g., focusing on hate (Kiela et al., 2020) and offensive (Suryawanshi et al., 2020) memes. However, the harm a meme can cause can be much broader. For instance, the meme[1] in Figure 1c is neither hateful nor offensive, but it is harmful to the media shown on the top left (ABC, CNN, etc.), as it compares them to China, suggesting that they adopt strong censorship policies. In short, the scope of harmful meme detection is *much broader*, and it may encompass other aspects such as cyberbullying, fake news, etc. Moreover, harmful memes have a target (e.g., news organization such as ABC and CNN in our previous example), which requires separate analysis not only to decipher their underlying semantics, but also to help with the explainability of the detection models.

WARNING: This paper contains meme examples and words that are offensive in nature.

[1] In order to avoid potential copyright issues, all memes we show in this paper are our own recreation of existing memes, using images with clear licenses.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2783–2796
August 1–6, 2021.

(a) `[0]` (b) `[2,0]` (c) `[1,1]` (d) `[2,2]` (e) `[2,3]`

Figure 1: Examples from our HarMeme dataset. The labels are in the format `[Intensity, Target]`. For `Intensity`, {0, 1, 2} correspond to *harmless*, *partially harmful*, and *very harmful*, respectively. For `Target`, {0, 1, 2, 3} correspond to *individual*, *organization*, *community*, and *society*, respectively. Examples 1b and 1c are harmful, but neither hateful, nor offensive. Example 1d is both harmful and offensive. Source (a); Source (b); Source (c) 1, Source (c) 2, Source (c) 3; Source (d); Source (e) 1, Source (e) 2, Source (e) 3; License 1 License 2.

Our contributions. In this paper, we study harmful memes, and we formulate two problems. **Problem 1 (Harmful meme detection)**: Given a meme, detect whether it is *very harmful*, *partially harmful*, or *harmless*. **Problem 2 (Target identification of harmful memes)**: Given a harmful meme, identify whether it targets an *individual*, an *organization*, a *community/country*, or the *society/general public/others*. To this end, we develop a novel dataset, HarMeme, containing $3,544$ real memes related to COVID-19, which we collected from the web and carefully annotated. Figure 1 shows several examples of memes from our collection, whether they are harmful, as well as the types of their targets. We prepare detailed annotation guidelines for both tasks. We further experiment with ten state-of-the-art unimodal and multimodal models for benchmarking the two problems. Our experiments demonstrate that a systematic combination of multimodal signals is needed to tackle these problems. Interpreting the models further reveals some of the biases that the best multimodal model exhibits, leading to the drop in performance. Finally, we argue that off-the-shelf models are inadequate in this context and that there is a need for specialized models

Our contributions can be summarized as follows:

- We study two new problems: (*i*) detecting harmful memes and (*ii*) detecting their targets.
- We release a new benchmark dataset, HarMeme, developed based on comprehensive annotation guidelines.
- We perform initial experiments with state-of-the-art textual, visual, and multimodal models to establish the baselines. We further discuss the limitations of these models.

Reproducibility. The full dataset and the source code of the baseline models are available at

`http://github.com/di-dimitrov/harmeme`

The appendix contains the values of the hyperparameters and the detailed annotation guidelines.

2 Related Work

Below, we present an overview of the datasets and the methods used for multimodal meme analysis.

Hate speech detection in memes. Sabat et al. (2019) developed a collection of $5,020$ memes for hate speech detection. Similarly, the Hateful Memes Challenge by Facebook introduced a dataset consisting of $10k+$ memes, annotated as hateful or non-hateful (Kiela et al., 2020). The memes were generated *artificially*, so that they resemble real ones shared on social media, along with "benign confounders." As part of this challenge, an array of approaches with different architectures and features have been tried, including Visual BERT, ViLBERT, VLP, UNITER, LXMERT, VILLA, ERNIE-Vil, Oscar and other Transformers (Li et al., 2019; Su et al., 2020; Zhou et al., 2020; Tan and Bansal, 2019; Gan et al., 2020; Yu et al., 2021; Li et al., 2020; Vaswani et al., 2017; Lippe et al., 2020; Zhu, 2020; Muennighoff, 2020). Other approaches include multimodal feature augmentation and cross-modal attention mechanism using inferred image descriptions (Das et al., 2020; Sandulescu, 2020; Zhou and Chen, 2020), as well as up-sampling confounders and loss re-weighting to complement multimodality (Lippe et al., 2020), web entity detection along with fair face classification (Karkkainen and Joo, 2021) from memes (Zhu, 2020), cross-validation ensemble learning and semi-supervised learning (Zhong, 2020) to improve robustness.

Meme sentiment/emotion analysis. Hu and Flaxman (2018) developed the TUMBLR dataset for emotion analysis, consisting of image–text pairs along with associated tags, by collecting posts from the TUMBLR platform. Thang Duong et al. (2017) prepared a multimodal dataset containing images, titles, upvotes, downvotes, #comments, etc., all collected from Reddit. Recently, SemEval-2020 Task 9 on Memotion Analysis (Sharma et al., 2020a) introduced a dataset of $10k$ memes, annotated with sentiment, emotions, and emotion intensity. Most participating systems in this challenge used fusion of visual and textual features computed using models such as Inception, ResNet, CNN, VGG-16 and DenseNet for image representation (Morishita et al., 2020; Sharma et al., 2020b; Yuan et al., 2020), and BERT, XLNet, LSTM, GRU and DistilBERT for text representation (Liu et al., 2020; Gundapu and Mamidi, 2020). Due to class imbalance in the dataset, approaches such as GMM and Training Signal Annealing (TSA) were also found useful. Morishita et al. (2020); Bonheme and Grzes (2020); Guo et al. (2020); Sharma et al. (2020b) proposed ensemble learning, whereas Gundapu and Mamidi (2020); De la Peña Sarracén et al. (2020) and several others used multimodal approaches. A few others leveraged transfer-learning using pre-trained models such as BERT (Devlin et al., 2019), VGG-16 (Simonyan and Zisserman, 2015), and ResNet (He et al., 2016). Finally, state-of-the-art results for all three tasks —sentiment classification, emotion classification and emotion quantification on this dataset,— were reported by Pramanick et al. (2021), who proposed a deep neural model that combines sentence demarcation and multi-hop attention. They also studied the interpretability of the model using the LIME framework (Ribeiro et al., 2016).

Meme propagation. Dupuis and Williams (2019) surveyed personality traits of social media users who are more active in spreading misinformation in the form of memes. Crovitz and Moran (2020) studied the characteristics of memes as a vehicle for spreading potential misinformation and disinformation. Zannettou et al. (2020a) discussed the quantitative aspects of large-scale dissemination of racist and hateful memes among polarized communities on platforms such as 4chan's /pol/. Ling et al. (2021) examined the artistic composition and the aesthetics of memes, the subjects they communicate, and the potential for virality. Based on this analysis, they manually annotated 50 memes as viral vs. non-viral. Zannettou et al. (2020b) analyzed the “Happy merchant” memes and showed how online fringe communities influence their spread to mainstream social networking platforms. They reported reasonable agreement for most manually annotated labels, and established a characterization for meme virality.

Other studies on memes. Reis et al. (2020) built a dataset of memes related to the 2018 and the 2019 election in Brazil ($34k$ images, $17k$ users) and India ($810k$ images, $63k$ users) with focus on misinformation. Another dataset of 950 memes targeted the propaganda techniques used in memes (Dimitrov et al., 2021a), which was also featured as a shared that at SemEval-2021 (Dimitrov et al., 2021b). Leskovec et al. (2009) introduced a dataset of 96 million memes collected from various links and blog posts between August 2008 and April 2009 for tracking the most frequently appearing stories, phrases, and information. Topic modeling of textual and visual cues of hate and racially abusive multi-modal content over sites such as 4chan was studied for scenarios that leverage genetic testing to claim superiority over minorities (Mittos et al., 2020). Zannettou et al. (2020a) examined the content of meme images and online posting activities to identify the probability of occurrence of one event in a specific background process, affecting the occurrence of other events in the rest of the processes, also known as Hawkes process (Hawkes, 1971), within the context of online posting of trolls. Wang et al. (2020) observed that fauxtographic content tends to attract more attention, and established how such content becomes a meme in social media. Finally, there is a recent survey on multi-modal disinformation detection (Alam et al., 2021).

Differences with existing studies. Hate speech detection in multimodal memes (Kiela et al., 2020) is the closest work to ours. However, we are substantially different from it and from other related studies as (*i*) we deal with *harmful* meme detection, which is a more *general* problem than *hateful* meme detection; (*ii*) along with harmful meme detection, we also identify the *entities that the harmful meme targets*; (*iii*) our HarMeme comprises *real-world memes* posted on the web as opposed to using synthetic memes as in (Kiela et al., 2020); and (*iv*) we present a unique dataset and benchmark results for harmful meme detection and for identifying the target of harmful memes.

3 *Harmful* Meme: Definition

Here, we define *harmful memes* as follows: *multimodal units consisting of an image and a piece of text embedded that has the potential to cause harm to an individual, an organization, a community, or the society more generally*. Here, *harm* includes mental abuse, defamation, psycho-physiological injury, proprietary damage, emotional disturbance, and compensated public image.

Harmful vs. hateful/offensive. *Harmful* is a more general term than *offensive* and *hateful*: *offensive* and *hateful* memes are *harmful*, but not all *harmful* memes are *offensive* or *hateful*. For instance, the memes in Figures 1b and 1c are neither offensive nor hateful, but harmful to *Donald Trump* and to *news media* such as CNN, respectively. Offensive memes typically aim to mock or to bully a social entity. A hateful meme contains offensive content that targets an entity (e.g., an individual, a community, or an organization) based on its personal/sensitive attributes such as gender, ethnicity, religion, nationality, sexual orientation, color, race, country of origin, and/or immigration status. The *harmful* content in a harmful meme is often camouflaged and might require critical judgment to establish its potencial to do hard. Moreover, the social entities attacked or targeted by harmful memes can be any individual, organization, or community, as opposed to *hateful* memes, where entities are attacked based on personal attributes.

4 Dataset

Below, we describe the data collection, the annotation process and the guidelines, and we give detailed statistics about the HarMeme dataset.

4.1 Data Collection and Deduplication

To collect potentially harmful memes in the context of COVID-19, we searched using different services, mainly *Google Image Search*. We used keywords such as *Wuhan Virus Memes*, *US Election and COVID Memes*, *COVID Vaccine Memes*, *Work From Home Memes*, *Trump Not Wearing Mask Memes*. We then used an extension[2] of Google Chrome to download the memes. We further scraped various publicly available groups on *Instagram* for meme collection. Note that, adhering to the terms of social media, we did not use content from any private/restricted pages.

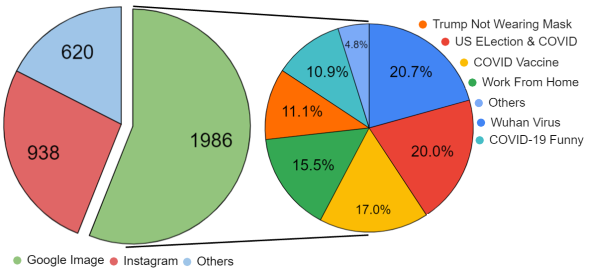

Figure 2: Statistics about the HarMeme dataset. On the left, we show the distribution by source, while on the right, we show the percentage of memes collected by corresponding keywords in Google Image Search.

Unlike the Hateful Memes Challenge (Kiela et al., 2020), which used synthetically generated memes, our HarMeme dataset contains *original memes* that were actually shared in social media. As all memes were gathered from real sources, we maintained strict filtering criteria[3] on the resolution of meme images and on the readability of the meme text during the collection process. We ended up collecting $5,027$ memes. However, as we collected memes from independent sources, we had some duplicates. We thus used two efficient de-duplication repositories[4] [5] sequentially, and we preserved the memes with the highest resolution from each group of duplicates. We removed $1,483$ duplicate memes, thus ending up with a dataset of $3,544$. Although we tried to collect only harmful memes, the dataset contained memes with various levels of harmfulness, which we manually labeled during the annotation process, as discussed in Section 4.3. We further used Google's OCR Vision API[6] to extract the textual content of each meme.

4.2 Annotation Guidelines

As discussed in Section 3, we consider a meme as harmful only if it is implicitly or explicitly intended to cause *harm* to an entity, depending on the personal, political, social, educational or industrial background of that entity. The intended *harm* can be expressed in an obvious manner such as by abusing, offending, disrespecting, insulting, demeaning, or disregarding the entity or any sociocultural or political ideology, belief, principle, or doctrine associated with that entity. Likewise, the *harm* can also be in the form of a more subtle attack such as mocking or ridiculing a person or an idea.

---

[2] http://download-all-images.mobilefirst.me/

[3] Details are given in Appendix B.3.

[4] gitlab.com/opennota/findimagedupes

[5] http://github.com/arsenetar/dupeguru

[6] http://cloud.google.com/vision

We asked the annotators to label the intensity of the harm as *harmful* or *partially harmful*, depending upon the context and the ingrained explication of the meme. Moreover, we formally defined four different classes of targets and compiled well-defined guidelines[7] that the annotators adhered to while manually annotating the memes. The four target entities are as follows (c.f. Figure 1):

1. **Individual:** A person, usually a celebrity (e.g., a well-known politician, an actor, an artist, a scientist, an environmentalist, etc. such as *Donald Trump, Joe Biden, Vladimir Putin, Hillary Clinton, Barack Obama, Chuck Norris, Greta Thunberg, Michelle Obama*).

2. **Organization:** An organization is a group of people with a particular purpose, such as a business, a governmental department, a company, an institution or an association, comprising more than one person, and having a particular purpose, such as research organizations (e.g., *WTO, Google*) and political organizations (e.g., *the Democratic Party*).

3. **Community:** A community is a social unit with commonalities based on personal, professional, social, cultural, or political attributes such as religious views, country of origin, gender identity, etc. Communities may share a sense of place situated in a given geographical area (e.g., a country, a village, a town, or a neighborhood) or in virtual space through communication platforms (e.g., online forums based on religion, country of origin, gender).

4. **Society:** When a meme promotes conspiracies or hate crimes, it becomes harmful to the general public, i.e., to the entire society.

During the process of collection and annotation, we rejected memes based on the following four criteria: (*i*) the meme text is in code-mixed or non-English language; (*ii*) the meme text is not readable (e.g., blurry text, incomplete text, etc.); (*iii*) the meme is unimodal, containing only textual or visual content; (*iv*) the meme contains cartoons (we added this last criterion as cartoons can be hard to analyze by AI systems).

MEME annotation project: Contribute

Harmful

Intensity

● Very harmful
○ Partially harmful
○ Harmless

Target of harmful content

● Targeting an individual
○ Targeting an organization
○ Targeting a community
○ Harmful to the society, or the general public

(a) Annotation interface

MEME consolidation project: Contribute

Harmful

Intensity

● Very harmful {3}
○ Partially harmful
○ Harmless

Target of harmful content

● Targeting an individual {2}
○ Targeting an organization {1}
○ Targeting a community
○ Harmful to the society, or the general public

(b) Consolidation interface

Figure 3: Snapshot of the PyBossa GUI used for annotation and consolidation.

4.3 Annotation Process

For the annotation process, we had 15 annotators, including professional linguists and researchers in Natural Language Processing (NLP): 10 of them were male and the other 5 were female, and their age ranged between 24–45 years. We used the PyBossa[8] crowdsourcing framework for our annotations (c.f. Figure 3). We split the annotators into five groups of three people, and each group annotated a different subset of the data. Each annotator spent about 8.5 minutes on average to annotate one meme. At first, we trained our annotators with the definition of harmful memes and their targets, along with the annotation guidelines. To achieve quality annotation, our main focus was to make sure that the annotators were able to understand well what harmful content is and how to differentiate it from humorous, satirical, hateful, and non-harmful content.

---

[7] More details of the annotation guidelines are presented in Appendix B.

[8] `http://pybossa.com/`

	Phase	Annotators		κ
Harmful meme detection	Trial Annotation	α_1	α_2	0.29
		α_1	α_3	0.34
		α_2	α_3	0.26
	Final Annotation	α_1	α_2	0.67
		α_1	α_3	0.75
		α_2	α_3	0.72
Target identification	Trial Annotation	α_1	α_2	0.35
		α_1	α_3	0.38
		α_2	α_3	0.39
	Final Annotation	α_1	α_2	0.77
		α_1	α_3	0.83
		α_2	α_3	0.79

Table 1: Cohen's κ agreement during different phases of annotation for each task: harmful meme detection (3-class classification) and target identification (4-class classification) of harmful memes.

Dry run. We conducted a dry run on a subset of 200 memes, which helped the annotators understand well the definitions of harmful memes and targets, as well as to eliminate the uncertainties about the annotation guidelines. Let α_i be a single annotator. For the preliminary data, we computed the inter-annotator agreement in terms of Cohen's κ (Bobicev and Sokolova, 2017) for three randomly chosen annotators $\alpha_{[1,2,3]}$ for each meme for both tasks. The results are shown in Table 1. We can see that the score is low for both tasks (0.295 and 0.373), which is expected for the initial dry run. With the progression of the annotation phases, we observed much higher agreement, thus confirming that the dry run helped to train the annotators.

Final annotation. After the dry run, we started the final annotation process. Figure 3a shows an example annotation of the PyBossa annotation platform. We asked the annotators to check whether a given meme falls under the four rejection criteria as given in the annotation guidelines. After confirming the validity of the meme, it was rated by three annotators for both tasks.

Consolidation. In the consolidation phase, for high agreements, we used majority voting to decide the final label, and we added a fourth annotator otherwise. Table 2 shows statistics about the labels and the data splits. After the final annotation, Cohen's κ increased to 0.695 and 0.797 for the two tasks, which is moderate and high agreement, respectively. These scores show the difficulty and the variability in gauging the *harmfulness* by human experts. For example, we found memes where two annotators independently chose *partially harmful*, but the third annotator annotated it as *very harmful*.

4.4 Lexical Analysis of HarMeme

Figure 4 shows the length distribution of the meme text for both tasks, and Table 3 shows the top-5 most frequent words in the union of the validation and the test sets. We can see that names of politicians and words related to COVID-19 are frequent in *very harmful* and *partially harmful* memes. For the target of the harmful memes, we notice the presence of various class-specific words such as *president, trump, obama, china*. These words often incorporate bias in the machine learning models, which makes the dataset more challenging and difficult to learn from (see Section 6.4 for more detail).

5 Benchmarking HarMeme dataset

We provide benchmark evaluations on HarMeme with a variety of state-of-the-art unimodal textual models, unimodal visual models, and models using both modalities. Except for unimodal visual models, we use MMF (Multimodal Framework)[9] to conduct the necessary experiments.

5.1 Unimodal Models

▷ **Text BERT:** We use textual BERT (Devlin et al., 2019) as the unimodal text-only model.
▷ **VGG19, DenseNet, ResNet, ResNeXt:** For the unimodal visual-only models, we used four different well-known models – VGG19 (Simonyan and Zisserman, 2015), DenseNet-161 (Huang et al., 2017), ResNet-152 (He et al., 2016), and ResNeXt-101 (Xie et al., 2017) pre-trained on the ImageNet (Deng et al., 2009) dataset. We extracted the feature maps from the last pooling layer of each architecture and fed them to a fully connected layer.

5.2 Multimodal Models

▷ **Late Fusion:** This model uses the mean score of pre-trained unimodal ResNet-152 and BERT.
▷ **Concat BERT:** It concatenates the features extracted by pre-trained unimodal ResNet-152 and text BERT, and uses a simple MLP as the classifier.
▷ **MMBT:** Supervised Multimodal Bitransformers (Kiela et al., 2019) is a multimodal architecture that inherently captures the intra-modal and the inter-modal dynamics within various input modalities.
▷ **ViLBERT CC:** Vision and Language BERT (ViLBERT) (Lu et al., 2019), trained on an intermediate multimodal objective (Conceptual Captions) (Sharma et al., 2018), is a strong model with task-agnostic joint representation of image + text.

---

[9] `github.com/facebookresearch/mmf`

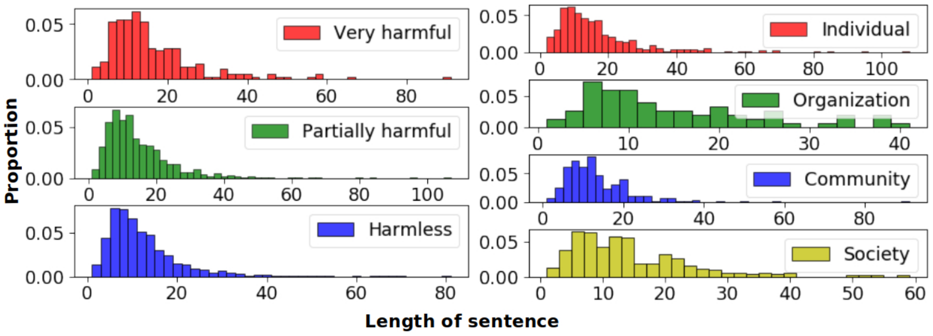

Figure 4: Histogram of the length of the meme' text for each class: for harmfulness on the left, and for the target of harmful memes on the right.

	#Memes	Harmfulness			#Memes	Target			
		Very Harmful	**Partially Harmful**	**Harmless**		**Individual**	**Organization**	**Community**	**Society**
Train	3,013	182	882	1,949	1,064	493	66	279	226
Validation	177	10	51	116	61	29	3	16	13
Test	354	21	103	230	124	59	7	32	26
Total	3,544	213	1,036	2,295	1,249	582	75	327	265

Table 2: Statistics about the HarMeme dataset. The memes belonging to the *very harmful* and the *partially harmful* categories are annotated with one of the following four targets: *individual*, *organization*, *community*, or *society*.

Harmfulness			Target			
Very Harmful	**Partially Harmful**	**Harmless**	**Individual**	**Organization**	**Community**	**Society**
mask (0.0512)	trump (0.0642)	you (0.0264)	trump (0.0541)	deadline (0.0709)	china (0.0665)	mask (0.0441)
trump (0.0404)	president (0.0273)	home (0.0263)	president (0.0263)	associated (0.0709)	chinese (0.0417)	vaccine (0.0430)
wear (0.0385)	obama (0.0262)	corona (0.0251)	donald (0.0231)	extra (0.0645)	virus (0.0361)	alcohol (0.0309)
thinks (0.0308	donald (0.0241)	work (0.0222)	obama (0.0217)	ensure (0.0645)	wuhan (0.0359)	temperatures (0.0309)
killed (0.0269)	virus (0.0213)	day (0.0188)	covid (0.0203)	qanon (0.0600)	cases (0.0319)	killed (0.0271)

Table 3: Top-5 most frequent words per class. The tf-idf score per word is given within parenthesis.

▷ **Visual BERT COCO:** Visual BERT (V-BERT) (Li et al., 2019) pre-trained on the multimodal COCO dataset (Lin et al., 2014) is another strong multimodal model used for a broad range of vision and language tasks.

6 Experimental Results

Below, we report the performance of the models described in the previous section for each of the two tasks. We further discuss some biases that negatively impact performance. Appendix A gives additional details about training and the values of the hyper-parameters we used in our experiments.

Evaluation measures We used six evaluation measures: Accuracy, Precision, Recall, Macro-averaged F1, Mean Absolute Error (MAE), and Macro-Averaged Mean Absolute Error (MMAE) (Baccianella et al., 2009). For the first four measures, higher values are better, while for the last two, lower values are better. Since the test set is imbalanced, measures like macro F1 and MMAE are more relevant.

6.1 Harmful Meme Detection

Table 4 shows the results for the harmful meme detection task. We start our experiments by merging the *very hateful* and the *partially hateful* classes, thus turning the problem into an easier *binary classification*. Afterwards, we perform the 3-class classification task. Since the test set is imbalanced, the majority class baseline achieves 64.76% accuracy. We observe that the unimodal visual models perform only marginally better than the majority class baseline, which indicates that they are insufficient to learn the underlying semantics of the memes.

Moving down the table, we see that the unimodal text model is marginally better than the visual models. Then, for multimodal models, the performance improves noticeably, and more sophisticated fusion techniques yield better results. We also notice the effectiveness of multimodal pre-training over unimodal pre-training, which supports the recent findings by Singh et al. (2020). While both ViLBERT CC and V-BERT COCO perform similarly, the latter achieves better Macro F1 and MMAE, which are the most relevant measures.

Modality	Model	Harmful Meme Detection											
		2-Class Classification						3-Class Classification					
		Acc ↑	P ↑	R ↑	F1 ↑	MAE ↓	MMAE ↓	Acc ↑	P ↑	R ↑	F1 ↑	MAE ↓	MMAE ↓
	Human†	90.68	84.35	84.19	83.55	0.1760	0.1723	86.10	67.35	65.84	65.10	0.2484	0.4857
	Majority	64.76	32.38	50.00	39.30	0.3524	0.5000	64.76	21.58	33.33	26.20	0.4125	1.0
`Text` Only	TextBERT	70.17	65.96	66.38	66.25	0.3173	0.2911	68.93	48.49	49.15	48.72	0.3250	0.5591
`Image` Only	VGG19	68.12	60.25	61.23	61.86	0.3204	0.3190	66.24	40.95	44.02	41.76	0.3198	0.6487
	DenseNet-161	68.42	61.08	62.10	62.54	0.3202	0.3125	65.21	41.88	44.25	42.15	0.3102	0.6326
	ResNet-152	68.74	61.86	62.89	62.97	0.3188	0.3114	65.29	41.95	44.32	43.02	0.3047	0.6264
	ResNeXt-101	69.79	62.32	63.26	63.68	0.3175	0.3029	66.55	42.62	44.87	43.68	0.3036	0.6499
`Image` + `Text` (Unimodal Pre-training)	Late Fusion	73.24	70.28	70.36	70.25	0.3167	0.2927	66.67	44.96	50.02	45.06	0.3850	0.6077
	Concat BERT	71.82	71.58	72.23	71.82	0.3033	0.3156	65.54	42.29	45.42	43.37	0.3881	0.5976
	MMBT	73.48	68.89	68.95	67.12	0.3101	0.3258	68.08	51.72	51.94	50.88	0.3403	0.6474
`Image` + `Text` (Multimodal Pre-training)	ViLBERT CC	78.53	78.62	**81.41**	78.06	0.2279	0.1881	**75.71**	48.89	49.21	48.82	**0.2763**	0.5329
	V-BERT COCO	**81.36**	**79.55**	81.19	**80.13**	**0.1972**	**0.1857**	74.01	**56.35**	**54.79**	**53.85**	0.3063	**0.5303**

Table 4: Performance for harmful meme detection. For two-class classification, we merge *very harmful* and *partially harmful* into a single class. † This row reports the human accuracy on the test set.

Modality	Model	Target Identification of Harmful Memes					
		Acc ↑	P ↑	R ↑	F1 ↑	MAE ↓	MMAE ↓
	Human†	87.55	82.28	84.15	82.01	0.7866	0.3647
	Majority	46.60	11.65	25.00	15.89	1.2201	1.5000
Text (T) only	TextBERT	69.35	55.60	54.37	55.60	1.1612	0.8988
Image (I) only	VGG19	63.48	53.85	54.02	53.60	1.1687	1.0549
	DenseNet-161	64.52	53.96	53.95	53.51	1.1655	1.0065
	ResNet-152	65.75	54.25	54.13	53.78	1.1628	1.0459
	ResNeXt-101	65.82	54.47	54.20	53.95	1.1616	0.9277
I + T (Unimmodal Pre-training)	Late Fusion	72.58	58.43	58.83	58.43	1.1476	0.6318
	Concat BERT	67.74	54.79	49.65	49.77	1.1377	0.8879
	MMBT	72.58	58.43	58.83	58.35	1.1476	0.6318
I + T (Multimodal Pre-training)	ViLBERT CC	72.58	59.92	55.78	57.17	1.1671	0.8035
	V-BERT COCO	**75.81**	**66.29**	**69.09**	**65.77**	**1.1078**	**0.5036**

Table 5: Performance for target identification of harmful memes (†human accuracy on the test set).

6.2 Target Identification for Harmful Memes

Table 5 shows the results for the target identification task. This is an imbalanced 4-class classification problem, and the majority class baseline yields 46.60% accuracy. The unimodal models perform relatively better here, achieving 63% – 70% accuracy; their F1 Macro and MMAE scores are also above the majority class. However, the overall performance of the unimodal models is poor. Incorporating multimodal signals with fine-grained fusion improves the results substantially, and advanced multimodal fusion techniques with multimodal pre-training perform much better than simple late fusion with unimodal pre-training. Moreover, V-BERT COCO outperforms ViLBERT CC by 8% of F1 score and by nearly 0.3 of MMAE.

6.3 Human Evaluation

To understand how human subjects perceive these tasks, we further hired a different set of experts (not the annotators) to label the test set. We observed 86% – 91% accuracy on average for both tasks, which is much higher than V-BERT, the best-performing model. This shows that their is a potential for enriched multimodal models that better understand the ingrained semantics of the memes.

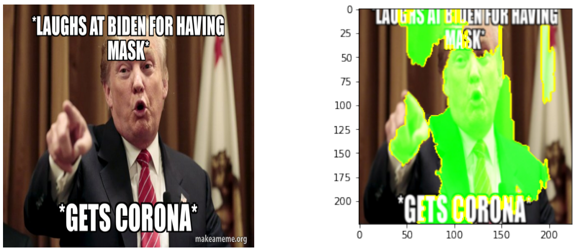

(a) Very harmful meme (b) LIME output - image

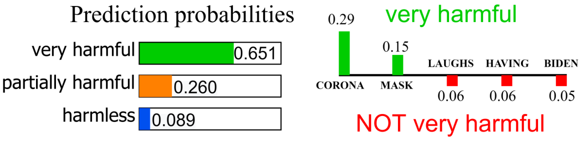

(c) LIME output - text

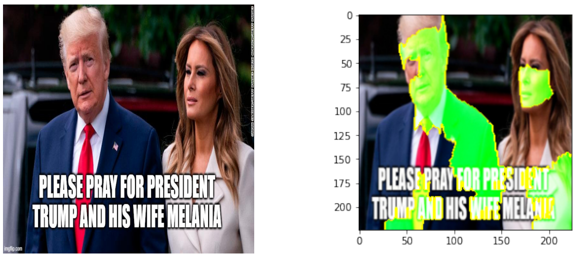

(d) Harmless meme (e) LIME output - image

Figure 5: Example of explanation by LIME on both visual and textual modalities and visualization of bias in V-BERT for both tasks.

6.4 Side-by-side Diagnostics and Anecdotes

Since the HarMeme dataset was compiled of memes related to COVID-19, we expected that models with enriched contextual knowledge and sophisticated technique would have superior performance. Thus, to comprehend the interpretability of V-BERT (the best model), we used LIME (Locally Interpretable Model-Agnostic Explanations) (Ribeiro et al., 2016), a consistent model-agnostic explainer to interpret the predictions.

We chose two memes from the test set to analyze the potential explanability of V-BERT. The first meme, which is shown in Figure 5a, was manually labeled as *very harmful*, and V-BERT successfully classified it, with prediction probabilities of 0.651, 0.260, and 0.089 corresponding to the *very harmful*, the *partially harmful*, and the *harmless* classes respectively. Figure 5b highlights the most contributing super-pixels to the *very harmful* (green) class. As expected, the face of Donald Trump, as highlighted by the green pixels, prominently contributed to the prediction. Figure 5c demonstrates the contribution of different meme words to the model prediction. We can see that words like *CORONA* and *MASK* have significant contributions to the *very harmful* class, thus supporting the lexical analysis of HarMeme as shown in Table 3.

The second meme, which is shown in Figure 5d, was manually labeled as *harmless*, but V-BERT incorrectly predicted it to be *very harmful*. Figure 5e shows that, similarly to the previous example, the face of Donald Trump contributed to the prediction of the model. We looked closer into our dataset, and we found that it contained many memes with the image of Donald Trump, and that the majority of these memes fall under the *very harmful* category and targeted and individual. Therefore, instead of leaning the underlying semantics of one particular meme, the model easily got biased by the presence of Donald Trump's image and blindly classified the meme as *very harmful*.

7 Conclusion and Future Work

We presented HarMeme, the first large-scale benchmark dataset, containing 3,544 memes, related to COVID-19, with annotations for degree of harmfulness (*very harmful*, *partially harmful*, or *harmless*), as well as for the target of the harm (an *individual*, an *organization*, a *community*, or *society*). The evaluation results using several unimodal and multimodal models highlighted the importance of modeling the multimodal signal (for both tasks) —(*i*) detecting harmful memes and (*ii*) detecting their targets—, and indicated the need for more sophisticated methods. We also analyzed the best model and identified its limitations.

In future work, we plan to design new multimodal models and to extend HarMeme with examples from other topics, as well as to other languages. Alleviating the biases in the dataset and in the models are other important research directions.

Ethics and Broader Impact

User Privacy. Our dataset only includes memes and it does not contain any user information.

Biases. Any biases found in the dataset are unintentional, and we do not intend to do harm to any group or individual. We note that determining whether a meme is harmful can be subjective, and thus it is inevitable that there would be biases in our gold-labeled data or in the label distribution. We address these concerns by collecting examples using general keywords about COVID-19, and also by following a well-defined schema, which sets explicit definitions during annotation. Our high inter-annotator agreement makes us confident that the assignment of the schema to the data is correct most of the time.

Misuse Potential. We ask researchers to be aware that our dataset can be maliciously used to unfairly moderate memes based on biases that may or may not be related to demographics and other information within the text. Intervention with human moderation would be required in order to ensure that this does not occur.

Intended Use. We present our dataset to encourage research in studying harmful memes on the web. We distribute the dataset for research purposes only, without a license for commercial use. We believe that it represents a useful resource when used in the appropriate manner.

Environmental Impact. Finally, we would also like to warn that the use of large-scale Transformers requires a lot of computations and the use of GPUs/TPUs for training, which contributes to global warming (Strubell et al., 2019). This is a bit less of an issue in our case, as we do not train such models from scratch; rather, we fine-tune them on relatively small datasets. Moreover, running on a CPU for inference, once the model has been fine-tuned, is perfectly feasible, and CPUs contribute much less to global warming.

Acknowledgments

The work was partially supported by the Wipro research grant and the Infosys Centre for AI, IIIT Delhi, India. It is also part of the Tanbih mega-project, developed at the Qatar Computing Research Institute, HBKU, which aims to limit the impact of "fake news," propaganda, and media bias by making users aware of what they are reading.

References

Firoj Alam, Stefano Cresci, Tanmoy Chakraborty, Fabrizio Silvestri, Dimiter Dimitrov, Giovanni Da San Martino, Shaden Shaar, Hamed Firooz, and Preslav Nakov. 2021. A survey on multimodal disinformation detection. *arXiv 2103.12541*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Evaluation Measures for Ordinal Regression. In *Proceedings of the Ninth International Conference on Intelligent Systems Design and Applications*, ISDA '09, pages 283–287, Pisa, Italy.

Victoria Bobicev and Marina Sokolova. 2017. Inter-annotator agreement in sentiment analysis: Machine learning perspective. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, RANLP '17, pages 97–102, Varna, Bulgaria.

Lisa Bonheme and Marek Grzes. 2020. SESAM at SemEval-2020 task 8: Investigating the relationship between image and text in sentiment analysis of memes. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 804–816, Barcelona, Spain.

Darren Crovitz and Clarice Moran. 2020. Analyzing Disruptive Memes in an Age of International Interference. *The English Journal*, 109(4):62–69.

Abhishek Das, Japsimar Singh Wahi, and Siyao Li. 2020. Detecting hate speech in multi-modal memes. *arXiv 2012.14891*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '09, pages 248–255, Miami, FL, USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, pages 4171–4186, Minneapolis, MN, USA.

Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021a. Detecting propaganda techniques in memes. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '21.

Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021b. SemEval-2021 Task 6: Detection of persuasion techniques in texts and images. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21.

Marc J. Dupuis and Andrew Williams. 2019. The spread of disinformation on the Web: An examination of memes on social networking. In *Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI '19, pages 1412–1418.

Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. 2020. Large-scale adversarial training for vision-and-language representation learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, NeurIPS '20, pages 1–15, Vancouver, Canada.

Sunil Gundapu and Radhika Mamidi. 2020. Gundapusunil at SemEval-2020 task 8: Multimodal Memotion Analysis. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 1112–1119, Barcelona, Spain.

Yingmei Guo, Jinfa Huang, Yanlong Dong, and Mingxing Xu. 2020. Guoym at SemEval-2020 task 8: Ensemble-based classification of visuo-lingual metaphor in memes. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 1120–1125, Barcelona, Spain.

Alan Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58:83–90.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778, Las Vegas, NV, USA.

Anthony Hu and Seth Flaxman. 2018. Multimodal sentiment analysis to explore the structure of emotions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACMKDD '18, pages 350–358, London, UK.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '17, pages 2261–2269, Honolulu, HI, USA.

Kimmo Karkkainen and Jungseock Joo. 2021. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, WACV '21, pages 1548–1558.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. 2019. Supervised multimodal bitransformers for classifying images and text. *arXiv 1909.02950*.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS '20*, pages 2611–2624, Vancouver, Canada.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '14, San Diego, CA, USA.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 497–506, New York, USA.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A simple and performant baseline for vision and language. *arxiv 1908.03557*.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Proceedings of the 16th European Computer Vision Conference*, ECCV '20, pages 121–137, Glasgow, UK.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision*, ECCV '14, pages 740–755, Zurich, Switzerland.

Chen Ling, Ihab AbuHilal, Jeremy Blackburn, Emiliano De Cristofaro, Savvas Zannettou, and Gianluca Stringhini. 2021. Dissecting the meme magic: Understanding indicators of virality in image memes. In *Proceedings of the 24th ACM Conference on Computer-Supported Cooperative Work and Social Computing*, CSCW '21.

Phillip Lippe, Nithin Holla, Shantanu Chandra, Santhosh Rajamanickam, Georgios Antoniou, Ekaterina Shutova, and Helen Yannakoudakis. 2020. A multimodal framework for the detection of hateful memes. *arXiv 2012.12871*.

Zehao Liu, Emmanuel Osei-Brefo, Siyuan Chen, and Huizhi Liang. 2020. UoR at SemEval-2020 task 8: Gaussian mixture modelling (GMM) based sampling approach for multi-modal memotion analysis. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 1201–1207, Barcelona, Spain.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Proceedings of the Conference on Neural Information Processing Systems*, NeurIPS '19, pages 13–23, Vancouver, Canada.

Alexandros Mittos, Savvas Zannettou, Jeremy Blackburn, and Emiliano De Cristofaro. 2020. "And we will fight for our race!" A measurement study of genetic testing conversations on Reddit and 4chan. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media*, ICWSM '20, pages 452–463, Atlanta, GA, USA.

Terufumi Morishita, Gaku Morio, Shota Horiguchi, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 task 8: Simple but effective modality ensemble for meme emotion recognition. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 1126–1134, Barcelona, Spain.

Niklas Muennighoff. 2020. Vilio: State-of-the-art Visio-Linguistic Models applied to Hateful Memes. *arxiv 2012.07788*.

Gretel Liz De la Peña Sarracén, Paolo Rosso, and Anastasia Giachanou. 2020. PRHLT-UPV at SemEval-2020 task 8: Study of multimodal techniques for memes analysis. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 908–915, Barcelona, Spain.

Shraman Pramanick, Md Shad Akhtar, and Tanmoy Chakraborty. 2021. Exercise? I thought you said 'extra fries': Leveraging sentence demarcations and multi-hop attention for meme affect analysis. *proceedings of the Fifteenth International AAAI Conference on Web and Social Media*.

Julio C. S. Reis, Philipe Melo, Kiran Garimella, Jussara M. Almeida, Dean Eckles, and Fabrício Benevenuto. 2020. A dataset of fact-checked images shared on WhatsApp during the Brazilian and Indian elections. In *Proceedings of the International AAAI Conference on Web and Social Media*, ICWSM '20, pages 903–908.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, San Francisco, CA, USA.

Benet Oriol Sabat, Cristian Canton-Ferrer, and Xavier Giró-i-Nieto. 2019. Hate speech in pixels: Detection of offensive memes towards automatic moderation. *arXiv 1910.02334*.

Vlad Sandulescu. 2020. Detecting hateful memes using a multimodal deep ensemble. *arXiv 2012.13235*.

Chhavi Sharma, Deepesh Bhageria, William Scott, Srinivas PYKL, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020a. SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor! In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 759–773, Barcelona, Spain.

Mayukh Sharma, Ilanthenral Kandasamy, and W.b. Vasantha. 2020b. Memebusters at SemEval-2020 task 8: Feature fusion model for sentiment analysis on memes using transfer learning. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 1163–1171, Barcelona, Spain.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 2556–2565, Melbourne, Australia.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '15, San Diego, CA, USA.

Amanpreet Singh, Vedanuj Goswami, and Devi Parikh. 2020. Are we pretraining it right? Digging deeper into visio-linguistic pretraining. *arXiv 2004.08744*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, pages 3645–3650, Florence, Italy.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. VL-BERT: pre-training of generic visual-linguistic representations. In *Proceedings of the 8th International Conference on Learning Representations*, ICLR '20, Addis Ababa, Ethiopia.

Shardul Suryawanshi, Bharathi Raja Chakravarthi, Mihael Arcan, and Paul Buitelaar. 2020. Multimodal meme dataset (MultiOFF) for identifying offensive content in image and text. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, LREC-TRAC '20, pages 32–41, Marseille, France.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, EMNLP-IJCNLP '17, pages 5100–5111, Hong Kong, China.

Chi Thang Duong, Remi Lebret, and Karl Aberer. 2017. Multimodal classification for analysing social media. *arXiv 1708.02099*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, NeurIPS '17, pages 5998–6008, Long Beach, CA, USA.

Yuping Wang, Fatemeh Tahmasbi, Jeremy Blackburn, Barry Bradlyn, Emiliano De Cristofaro, David Magerman, Savvas Zannettou, and Gianluca Stringhini. 2020. Understanding the use of fauxtography on social media. *arXiv 2009.11792*.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '17, pages 1492–1500, Honolulu, HI, USA.

Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE-ViL: Knowledge enhanced vision-language representations through scene graph. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, AAAI '21, pages 3208–3216.

Li Yuan, Jin Wang, and Xuejie Zhang. 2020. YNU-HPCC at SemEval-2020 task 8: Using a parallel-channel model for memotion analysis. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, SemEval '20, pages 916–921, Barcelona, Spain.

Savvas Zannettou, Tristan Caulfield, Barry Bradlyn, Emiliano De Cristofaro, Gianluca Stringhini, and Jeremy Blackburn. 2020a. Characterizing the use of images in state-sponsored information warfare operations by Russian trolls on Twitter. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media*, ICWSM '20, pages 774–785, Atlanta, GA, USA.

Savvas Zannettou, Joel Finkelstein, Barry Bradlyn, and Jeremy Blackburn. 2020b. A quantitative approach to understanding online antisemitism. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media*, ICWSM '20, pages 786–797, Atlanta, GA, USA.

Xiayu Zhong. 2020. Classification of multimodal hate speech – the winning solution of hateful memes challenge. *arXiv 2012.01002*.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and VQA. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13041–13049.

Yi Zhou and Zhenhao Chen. 2020. Multimodal learning for hateful memes detection. *arXiv 2011.12870*.

Ron Zhu. 2020. Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution. *arXiv 2012.08290*.

A Implementation Details and Hyper-Parameter Values

We trained all the models using the Pytorch framework on an NVIDIA Tesla T4 GPU with 16 GB of dedicated memory and with CUDA-10 and cuDNN-11 installed. For the unimodal models, we imported all the pre-trained weights from the TORCHVISION.MODELS[10] subpackage of PyTorch. We initialized the non pre-trained weights randomly with a zero-mean Gaussian distribution with a standard deviation of 0.02. To minimize the impact of the label imbalance in the loss calculation, we assigned larger weights to the minority class. We trained our models using the Adam optimizer (Kingma and Ba, 2014) and the negative log-likelihood loss as the objective function. Table A.1 gives the values of all hyper-parameters we used for training.

We trained the models end-to-end for the two classification tasks, i.e., the memes that were classified as *Very Harmful* or *Partially Harmful* in the first classification stage were sent to the second stage for target identification.

B Annotation Guidelines

B.1 What do we mean by *harmful* memes?

The entrenched meaning of harmful memes is targeted towards a social entity (e.g., an individual, an organization, a community, etc.), likely to cause calumny/vilification/defamation depending on their background (bias, social background, educational background, etc.). The *harm* caused by a meme can be in the form of mental abuse, psycho-physiological injury, proprietary damage, emotional disturbance, compensated public image. A harmful meme typically attacks celebrities or well-known organizations, with the intent to expose their professional demeanor.

Characteristics of *harmful* memes:

- Harmful memes may or may not be offensive, hateful, or biased in nature.
- Harmful memes expose vices, allegations, and other negative aspects of an entity based on verified or unfounded claims or mocks.
- Harmful memes leave an open-ended connotation to the word *community*, including *antisocial* communities such as terrorist groups.
- The harmful content in harmful memes is often implicit and might require critical judgment to establish its potential to do harm.
- Harmful memes can be classified at multiple levels, based on the intensity of the harm they could cause, e.g., *very harmful* or *partially harmful*.
- One harmful meme can target multiple individuals, organizations, and/or communities at the same time. In that case, we asked the annotators to go with the best personal judgment.
- Harm can be expressed in the form of sarcasm and/or political satire. Sarcasm is praise that is actually an insult; sarcasm generally involves malice, the desire to put someone down. On the other hand, satire is the ironical exposure of the vices or the follies of an individual, a group, an institution, an idea, the society, etc., usually with the aim to correcting it.

B.2 What is the difference between *organization* and *community*?

An organization is a group of people with a particular purpose, such as a business or a government department. Examples include a company, an institution, or an association comprising one or more people with a particular purpose, e.g., a research organization, a political organization, etc.

On the other hand, a community is a social unit (a group of living things) with a commonality such as norms, religion, values, ideology customs, or identity. Communities may share a sense of place situated in a given geographical area (e.g., a country, a village, a town, or a neighborhood) or in the virtual space through communication platforms.

B.3 When do we *reject* a meme?

We apply the following rejection criteria during the process of data collection and annotation:

1. The meme's text is code-mixed or not in English.
2. The meme's text is not readable. (e.g., blurry text, incomplete text, etc.)
3. The meme is unimodal in nature, containing only textual or only visual content.
4. The meme contains a cartoon.

Figure B.1 shows some rejected memes.

---

[10] http://pytorch.org/docs/stable/torchvision/models.html

	Models	Hyper-parameters					
		Batch Size	**Epochs**	**Learning Rate**	**Image Encoder**	**Text Encoder**	**#Parameters**
Unimodal	TextBERT	16	100	0.001	-	Bert-base-uncased	110,683,414
	VGG19	64	200	0.01	VGG19	-	138,357,544
	DenseNet-161	32	200	0.01	DenseNet-161	-	28,681,538
	ResNet-152	32	300	0.01	ResNet-152	-	60,192,808
	ResNeXt-101	32	300	0.01	ResNeXt-101	-	83,455,272
Multimodal	Late Fusion	16	200	0.0001	ResNet-152	Bert-base-uncased	170,983,752
	Concat BERT	16	200	0.001	ResNet-152	Bert-base-uncased	170,982,214
	MMBT	16	200	0.001	ResNet-152	Bert-base-uncased	169,808,726
	ViLBERT CC	16	100	0.001	Faster RCNN	Bert-base-uncased	112,044,290
	V-BERT COCO	16	100	0.001	Faster RCNN	Bert-base-uncased	247,782,404

Table A.1: The values of the hyper-parameters of all our models.


Shubham Suratkar
@Shubhameme

Hey Bhagwaan is saal bhi exams online hi karwaan do.

(a) Non-English (Hindi) meme. Source License (b) Unreadable meme. Source License

(c) Meme with a cartoon. Source License (d) Meme without textual modality. Source License

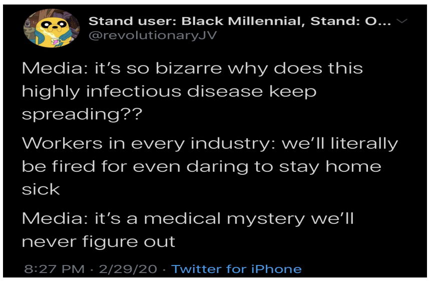
Stand user: Black Millennial, Stand: O...
@revolutionaryJV

Media: it's so bizarre why does this highly infectious disease keep spreading??

Workers in every industry: we'll literally be fired for even daring to stay home sick

Media: it's a medical mystery we'll never figure out

8:27 PM · 2/29/20 · Twitter for iPhone

(e) Meme without visual modality. Source License

Figure B.1: Examples of memes that we rejected during the process of data collection and annotation.

Progressive Multi-Granularity Training for Non-Autoregressive Translation

Liang Ding[*]
The University of Sydney
ldin3097@sydnye.edu.au

Longyue Wang[*]
Tencent AI Lab
vinnylywang@tencent.com

Xuebo Liu
University of Macau
nlp2ct.xuebo@gmail.com

Derek F. Wong
University of Macau
derekfw@um.edu.com

Dacheng Tao
JD Explore Academy, JD.com
dacheng.tao@gmail.com

Zhaopeng Tu
Tencent AI Lab
zptu@tencent.com

Abstract

Non-autoregressive translation (NAT) significantly accelerates the inference process via predicting the entire target sequence. However, recent studies show that NAT is weak at learning high-mode of knowledge such as one-to-many translations. We argue that modes can be divided into various granularities which can be learned from easy to hard. In this study, we empirically show that NAT models are prone to learn fine-grained lower-mode knowledge, such as words and phrases, compared with sentences. Based on this observation, we propose progressive multi-granularity training for NAT. More specifically, to make the most of the training data, we break down the sentence-level examples into three types, i.e. words, phrases, sentences, and with the training goes, we progressively increase the granularities. Experiments on Romanian-English, English-German, Chinese-English and Japanese-English demonstrate that our approach improves the phrase translation accuracy and model reordering ability, therefore resulting in better translation quality against strong NAT baselines. Also, we show that more deterministic fine-grained knowledge can further enhance performance.

1 Introduction

Non-autoregressive translation (NAT, Gu et al., 2018) has been proposed to improve the decoding efficiency by predicting all tokens independently and simultaneously. Different from autoregressive translation (AT, Vaswani et al., 2017) models that generate each target word conditioned on previously generated ones, NAT models suffer from the multimodality problem (i.e. multiple translations for a single input), in which the conditional independence assumption prevents a model from properly capturing the highly multimodal distribution of target translations. To reduce the modes of training data, sequence-level knowledge distillation (KD) (Kim and Rush, 2016) is widely employed via replacing their original target samples with sentences generated from an AT teacher (Gu et al., 2018; Zhou et al., 2020; Ren et al., 2020).

Granular.	**AT**	**NAT**			
		Raw	△	KD	△
WORD	59.8	57.1	-2.7	59.0	-0.8
PHRASE	36.0	31.7	-4.3	34.2	-1.8
SENTENCE	29.2	24.5	-4.7	27.0	-2.2

Table 1: Translation performance at different granularity on the WMT14 English⇒German dataset. "△" indicates the performance gap between the NAT and AT.

Although KD reduces the learning difficulty for NAT, there are still complicated word orders and structures (Gell-Mann and Ruhlen, 2011) in the synthetic sentences, making the NAT performance sub-optimal. To answer this challenge, Saharia et al. (2020); Ran et al. (2021) propose to lowers the bilingual modeling difficulties under the *monotonicity assumption*, where bilingual sentences are in the same word order. However, they make extensive modifications to model structures or objectives, limiting the applicability of their methods to a boarder range of tasks and languages.

Accordingly, we turn to break down the sentence-level high modes into finer granularities, i.e. bilingual words and phrases, where we assume that finer granularities are easy to be learned by NAT. As shown in Table 1, we analyzed the translation accuracy at three linguistic levels (i.e. word, phrase and sentence) and found that although KD brings promising improvements at three granular-

---

[*] Liang Ding and Longyue Wang contributed equally to this work. Work was done when Liang Ding and Xuebo Liu were interning at Tencent AI Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2797–2803
August 1–6, 2021.

	Source	**Targets**
Word	bank	银行 岸 储库
Phrase	hollow structural	中空 结构 空心 的 结构 镂空 结构
Sentence	He is very good at English.	他 英文 很 好。 他 非常 擅长 英语 。 他的 英语 水平 很 高 。

Table 2: Examples of different translation granularities.

ities, there are still some gaps with AT teacher. Also, we showed that finer granularities are easier to be learned, that is, accuracy gap "Δ" of WORD is small than that of PHRASE, and SENTENCE ($0.8<1.8<2.2$). Thus, we propose a simple and effective training strategy to enhance the ability to handle the sentence-level high modes. More specifically, we generate bilingual lexicons from parallel data by leveraging word alignment and phrase extraction in statistical machine translation (SMT, Zens et al., 2002). Then we guide the NAT model to progressively learn the bilingual knowledge from low to high granularity. Experimental results on four commonly-cited translation benchmarks show that our proposed PROGRESSIVE MULTI-GRANULARITY (PMG) training strategy consistently improves the translation performance. The main contributions are:

- Our study reveals that NAT is better at learning fine-grained knowledge. Training with sentences merely may be sub-optimal.
- We propose PMG training to encourage NAT models to learn from easy to hard. The fine-grained knowledge distilled by SMT will be dynamically transferred during training.
- Experiments across language pairs and model structures show the effectiveness and universality of PMG training.

2 Methodology

2.1 Motivation

We investigated theories in second-language acquisition: one usually learns a foreign language from word-to-word translation to sentence-to-sentence translation, namely from local to global (Onnis et al., 2008). Bilingual knowledge is at the core of adequacy modeling (Tu et al., 2016), which is a major weakness of the NAT models due to the lacks of autoregressive factorization. Table 2 demonstrates the English⇒Chinese multimodality at different granularities (i.e. word, phrase, sentence levels). As seen, the sentence-level consists of various kinds of modes, including word alignment ("English" vs. "英语"/"英文"), phrase translation ("be good at" vs. "...非常 擅长..."/"...水平 很 高"), and even reordering ("英语" can be subject or object). However, phrase-level modes are less complex with similar structure and word-level modes are simple with token-to-token mapping. Generally, the lower level of bilingual knowledge, the easier for NAT to learn. This example explains why the sentence level performance gaps between NAT and AT are significant than that of word and phrase in Table 1. Based on the above evidence, it is natural to suspect that the existing sentence-level NAT training is sub-optimal.

2.2 Fine-grained Bilingual Knowledge

Phrase table is an essential component of SMT systems, which records the correspondence between bilingual lexicons (Koehn and Callison-Burch, 2009). For each training example in the original training set, we sample its all possible inter-sentence bi-lingual phrases from the phrase table that obtained with phrase-based statistical machine translation (PBSMT) model (Koehn et al., 2003). The GIZA++ (Och and Ney, 2003) was employed to build word alignments for the training datasets. We leave the exploitation of more advanced forms bilingual knowledge such as syntax rules (Liu et al., 2006) and discontinuous phrases (Galley and Manning, 2010) for future work. Take the sentence pair in Table 2 for example, we can obtain the bilingual En-Zh phrase pairs "*very good* ||| 很好", "*good at English* ||| 擅长英语" from original sentence pair, informing the NAT model the explicit phrase boundaries.

2.3 Progressive Multi-Granularity Training

We present an extremely simple progressive multi-granularity (PMG) training fashion. Concretely, we progressively schedule the PMG: learn from "low" to "high" granularity, i.e. word→phrase→sentence. And we empirically set the training steps for each training stage. Our work can be seen as a typical determinism-based curriculum learning (CL) (Bengio et al., 2009) method, where the finer granular-

Models	**Speed**	**BLEU**			
		Ro-En	**En-De**	**Zh-En**	**Ja-En**
AT Models					
Transformer-BASE (Ro-En Teacher)	1.0×	34.1	27.3	24.4	29.2
Transformer-BIG (En-De / Zh-En / Ja-En Teacher)	0.8×	n/a	29.2	25.3	29.8
Existing NAT Models					
NAT (Gu et al., 2018)	2.4×	31.4	19.2	n/a	n/a
Iterative NAT (Lee et al., 2018)	2.0×	30.2	21.6	n/a	n/a
DisCo (Kasai et al., 2020)	3.2×	33.3	26.8	n/a	n/a
Levenshtein (Gu et al., 2019)	3.5×	33.3	27.3	n/a	n/a
Mask-Predict (Ghazvininejad et al., 2019)	1.5×	33.3	27.0	23.2	n/a
Context-aware NAT (Ding et al., 2020b)	1.5×	33.2	27.5	24.6	29.4
Our NAT Models					
Levenshtein (Gu et al., 2019)	3.5×	33.2	27.4	24.4	29.1
+PMG Training		33.8†	27.8	25.0†	29.6
Mask-Predict (Ghazvininejad et al., 2019)	1.5×	33.3	27.0	24.0	28.9
+PMG Training		33.7	27.6†	24.5	29.5†

Table 3: Comparison with previous work on WMT16 Ro-En, WMT14 En-De, WMT17 Zh-En and WAT17 Ja-En datasets. "†" indicates that the proposed method was significantly better than baseline at significance level $p<0.05$.

ities are more deterministic than sentences. Thus we compare with typical CL works (Zhang et al., 2019; Platanios et al., 2019) in Section 3.2.

3 Experiment

3.1 Setup

Data Experiments were conducted on four widely-used translation datasets: WMT14 English-German (En-De), WMT16 Romanian-English (Ro-En), WMT17 Chinese-English (Zh-En) and WAT17 Japanese-English (Ja-En), which consist of 4.5M, 0.6M, 20M and 2M sentence pairs, respectively. It is worthy noting that Ro-En, En-De and Zh-En are low-, medium- and high- resource language pairs, and Ja-En is word order divergent language direction. We use the same validation and test datasets with previous works for fair comparison. To avoid unknown works, we preprocessed data via byte-pair encoding (BPE) (Sennrich et al., 2016) with 32K merge operations. We evaluated the translation quality with BLEU (Papineni et al., 2002) with statistical significance test (Collins et al., 2005). For fine-grained bilingual knowledge, e.g. word alignment and phrase table, to ensure the source to target mapping more deterministic, we set 0.05 as the probability threshold. Taking WMT14 En-De for example, there are 3M words and 156M phrases in the original phrase table extracted by SMT methodology. We then filter the items whose translation probability is lower than 0.05 and obtain 0.3M words and 56.5M phrases as the final data.

Non-Autoregressive Models We validated our progressive multi-granularity training strategy on two state-of-the-art NAT model structures:

- *Mask-Predict* (MaskT, Ghazvininejad et al. 2019) that uses the conditional mask LM (Devlin et al., 2019) to iteratively generate the target sequence from the masked input;
- *Levenshtein Transformer* (LevT, Gu et al. 2019) that introduces three steps: deletion, placeholder prediction and token prediction.

For regularization, we empirically set the dropout rate as 0.2, and apply weight decay with 0.01 and label smoothing with $\epsilon = 0.1$. We train batches of approximately 128K tokens using Adam (Kingma and Ba, 2015). The learning rate warms up to 5×10^{-4} in the first 10K steps, and then decays with the inverse square-root schedule. We train 50k steps on word-level data and 50k steps on phrase-level data, respectively. And then update the remaining 200K steps for sentence-level training. Following the common practices (Ghazvininejad et al., 2019; Kasai et al., 2020), we evaluate the performance on an ensemble of 5 best checkpoints (ranked by validation BLEU) to avoid stochasticity.

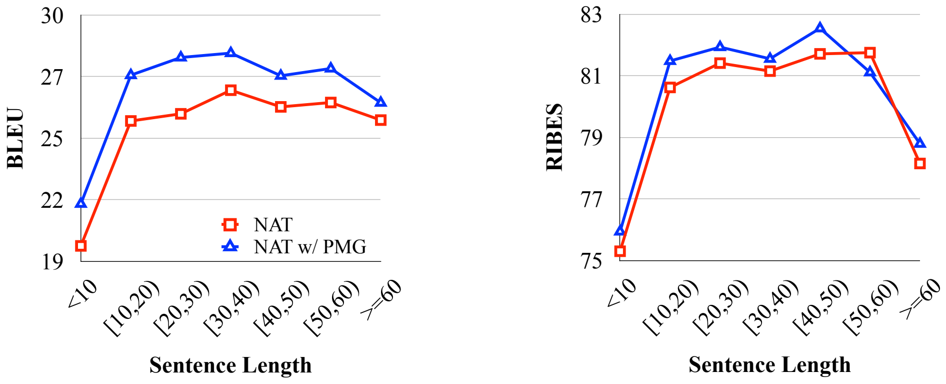

(a) Translation Quality (b) Reordering Difference

Figure 1: Performances of our proposed approach on different length bins against the vanilla NAT model.

Autoregressive Teachers We closely followed previous works to apply sequence-level KD. More precisely, we trained two kinds of Transformer (Vaswani et al., 2017) models, including Transformer-BASE and Transformer-BIG. The main results employ BIG for all directions except Ro-En, which is distilled by BASE. The architectures of Transformer-BIG utilizes a large batch (458K tokens) training strategy.

3.2 Experimental Results

Main Results Table 3 lists the results of previous competitive NAT models (Gu et al., 2018; Kasai et al., 2020; Gu et al., 2019; Ghazvininejad et al., 2019). Clearly, our approach "+PMG Training" consistently improves translation performance (BLEU↑) over four language pairs. Specifically, our PMG training strategy achieves on average +0.53 BLEU scores improvements on four language pairs upon two NAT model structures. Note that our approaches introduce no extra parameters, thus does not increase any latency ("Speed").

Comparison to Curriculum Learning The existing CL methods can be divided into two categories, "Discretized CL (DCL)" (Zhang et al., 2019) and "Continuous CL (CCL)" (Platanios et al., 2019). Sentence length is the most significant variable in our multi-granularity data, therefore we implemented discretized and continuous CL with the sentence length (source side) criteria.

Our DCL setting explicitly predefined the number of data bins, while CCL method continuously samples the shorter examples with the training progresses. For DCL, we split the training samples into a predefined number of bins (5, in our case). As for CCL, we employ their length curriculum and square root competence function. We find that on WMT14 En-De dataset with MaskT model, DCL performs worse than KD baseline (-0.6 BLEU) while CCL outperforms KD baseline by +0.3 BLEU points. Our approach (+0.6 BLEU) is the most effective one.

3.3 Analysis

In this section, we conducted analytical experiments to better understand what contributes to translation performance gains. Specifically, we investigate whether the PMG 1) enhance the phrasal pattern modeling ability? 2) improve the reordering? and 3) gain better performance with higher quality fine-grained knowledge?

Better Phrasal Pattern Modelling Our method is expected to pay more attention on the bi-lingual phrases, leading to better phrase translation accuracy. To evaluate the accuracy of phrase translations, we calculate the improvement over multiple granularities of n-grams in Table 4, our PMG training "NAT w/ PMG" consistently outperforms the baseline, indicating that our proposed multi-granularity training indeed raise the ability of NAT model on capturing the phrasal patterns.

Better Reordering Ability The SMT-distilled bilingual phrasal information could intuitively inform the NAT model the bi-lingual phrasal boundaries, leading to better reordering ability. We compare the reordering ability of NAT model w/ & w/o PMG training with RIBES[1] (Isozaki et al., 2010), which is designed for measuring the reordering performance for distant language pairs. We cate-

---

[1] `http://www.kecl.ntt.co.jp/icl/lirg/ribes`

N-gram	2	3	4	5	6
Δ BLEU	0.5	0.3	0.3	0.2	0.2

Table 4: Improvements of our proposed PMG training strategy on different N-grams against vanilla NAT.

Ratio	10%	35%	50%	100%
Δ BLEU	+0.3	+0.6	+0.7	+0.6

Table 5: Improvement of PMG training strategy on different fine-grained data scales against vanilla NAT.

gorize the test set into several bins according the sentence length and report the BLEU and RIBES scores, simultaneously in Figure 1. As seen, the proposed PMG training strategy could improve the translation (BLEU↑) and reordering performance (RIBES↑), confirming our claim. Our finding is consistent with Ding et al. (2020a), where they explicitly injected the SMT-guided alignment information into the MT models, achieving better performance.

Effect of Fine-Grained Text Quality The acquired fine-grained bilingual knowledge, i.e. word alignments and phrase tables, still have extremely large volumes after filtering. Taking WMT14 En-De for example, there are over 56M phrase pairs after filtering with translation probability threshold 0.05. To make the knowledge being more deterministic, we control the quality of fine-grained text with the third party scorer – BERTScore (Zhang et al., 2020). As illustrated in Table 5, keeping the high quality bilingual knowledge (e.g. 50%) can achieve further improvements, showing the great potential of our approach. We will leave the exploration of high-quality bilingual knowledge for NAT as a future work.

4 Related Works

Non-Autoregressive Translation There still exists a performance gap between AT teacher and its NAT student. To bridge this gap, many studies have been proposed. Ghazvininejad et al. (2019); Gu et al. (2019); Kasai et al. (2020) designed novel model structures to considerably improve the NAT model capacity. Wang et al. (2019); Ran et al. (2021); Ding et al. (2021b); Du et al. (2021) explored to improve the model performance with additional training signals or objectives. Guo et al. (2020b); Su et al. (2021) delivered the knowledge from pretrained language models to the NAT models. Above works improve the NAT at the model level, while we improve NAT at the data level.

Most related to our work, Ding et al. (2021a) proposed data-level strategies, including reverse distillation and bidirectional distillation, to make the most of the parallel data. Differently, we break the sentences into fine-grained granularities to fully exploit the parallel data. Note that our model-agnostic method can be applied to any NAT structures.

Curriculum Learning Our proposed training strategy is a novel technique for NAT by exploiting curriculum learning (CL). Recent works have shown that CL can help the autoregressive translation (AT) models achieve fast convergence and better results (Platanios et al., 2019; Liu et al., 2020b; Zhan et al., 2021; Zhou et al., 2021). However, CL for non-autoregressive translation (NAT) models has not been well studied. Among the few attempts, Guo et al. (2020a); Liu et al. (2020a) respectively investigated "parameter- and task-level" curriculum learning approaches, while we proposed progressive multi-granularity training for NAT at "data-level". To the best of our knowledge, this is the first work to investigate the effects of different granularities of data on NAT models.

5 Conclusion

In this paper, we investigated the translation accuracy of different granularities in NAT, and found that the NAT models are better at dealing with fine-grained bilingual knowledge (e.g. words and phrases). Based on this finding, we proposed a simple progressive multi-granularity training strategy. Experiments show that our approach consistently and significantly improves translation performance across language pairs and model architectures. In-depth analyses indicate that our approach generates better word order and phrase patterns, outperforming typical curriculum learning methods.

Acknowledgments

We are grateful to the anonymous reviewers and the area chair for their insightful comments and suggestions. Xuebo Liu and Derek F. Wong were supported in part by the Science and Technology Development Fund, Macau SAR (Grant No. 0101/2019/A2), and the Multi-year Research Grant from the University of Macau (Grant No. MYRG2020-00054-FST).

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021a. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation. In *ACL*.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021b. Understanding and improving lexical choice in non-autoregressive translation. In *ICLR*.

Liang Ding, Longyue Wang, and Dacheng Tao. 2020a. Self-attention with cross-lingual position representation. In *ACL*.

Liang Ding, Longyue Wang, Di Wu, Dacheng Tao, and Zhaopeng Tu. 2020b. Context-aware cross-attention for non-autoregressive translation. In *COLING*.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *ICML*.

Michel Galley and Christopher D Manning. 2010. Accurate non-hierarchical phrase-based translation. In *NAACL*.

Murray Gell-Mann and Merritt Ruhlen. 2011. The origin and evolution of word order. *PNAS*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *NeurIPS*.

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020a. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. In *AAAI*.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. 2020b. Incorporating bert into parallel sequence decoding with adapters. In *NeurIPS*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *EMNLP*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Parallel machine translation with disentangled context transformer. In *ICML*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Philipp Koehn and Chris Callison-Burch. 2009. *Statistical Machine Translation*. Citeseer.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Jinglin Liu, Yi Ren, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020a. Task-level curriculum learning for non-autoregressive neural machine translation. In *IJCAI*.

Xuebo Liu, Houtim Lai, Derek F Wong, and Lidia S Chao. 2020b. Norm-based curriculum learning for neural machine translation. In *ACL*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *ACL*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*.

Luca Onnis, Heidi R Waterfall, and Shimon Edelman. 2008. Learn locally, act globally: Learning language from variation set cues. *Cognition*, 109(3).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *NAACL*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. In *AAAI*.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *ACL*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *EMNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *EACL*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *AAAI*.

Richard Zens, Franz Josef Och, and Hermann Ney. 2002. Phrase-based statistical machine translation. In *AAAI*.

Runzhe Zhan, Xuebo Liu, Derek F Wong, and Lidia S Chao. 2021. Meta-curriculum learning for domain adaptation in neural machine translation. In *AAAI*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *ICLR*.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. In *NAACL*.

Chunting Zhou, Graham Neubig, and Jiatao Gu. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *ICLR*.

Lei Zhou, Liang Ding, Kevin Duh, Shinji Watanabe, Ryohei Sasano, and Koichi Takeda. 2021. Self-guided curriculum learning for neural machine translation. *arXiv*.

ZmBART: An Unsupervised Cross-lingual Transfer Framework for Language Generation

Kaushal Kumar Maurya
Indian Institute of Technology Hyderabad
Hyderabad, India
cs18resch11003@iith.ac.in

Maunendra Sankar Desarkar
Indian Institute of Technology Hyderabad
Hyderabad, India
maunendra@cse.iith.ac.in

Yoshinobu Kano
Shizuoka University, Japan
kano@inf.shizuoka.ac.jp

Kumari Deepshikha
NVIDIA, India
deepkshikha@gmail.com

Abstract

Despite the recent advancement in NLP research, cross-lingual transfer for natural language generation is relatively understudied. In this work, we transfer supervision from high resource language (HRL) to multiple low-resource languages (LRLs) for natural language generation (NLG). We consider four NLG tasks (text summarization, question generation, news headline generation, and distractor generation) and three syntactically diverse languages, i.e., English, Hindi, and Japanese. We propose an unsupervised cross-lingual language generation framework (called ZmBART) that does not use any parallel or pseudo-parallel/back-translated data. In this framework, we *further* pre-train mBART sequence-to-sequence denoising auto-encoder model with an auxiliary task using monolingual data of three languages. The objective function of the auxiliary task is close to the target tasks which enriches the multi-lingual latent representation of mBART and provides good initialization for target tasks. Then, this model is fine-tuned with task-specific supervised English data and directly evaluated with low-resource languages in the Zero-shot setting. To overcome catastrophic forgetting and spurious correlation issues, we applied freezing model component and data argumentation approaches respectively. This simple modeling approach gave us promising results. We experimented with few-shot training (with 1000 supervised data-points) which boosted the model performance further. We performed several ablations and cross-lingual transferability analysis to demonstrate the robustness of ZmBART.

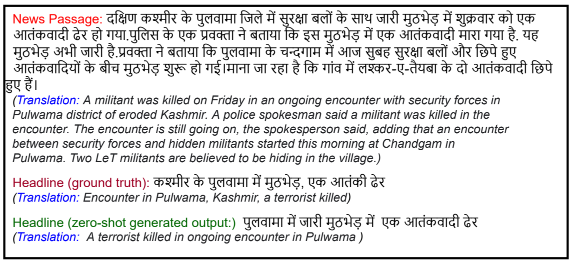

News Passage: दक्षिण कश्मीर के पुलवामा जिले में सुरक्षा बलों के साथ जारी मुठभेड़ में शुक्रवार को एक आतंकवादी ढेर हो गया.पुलिस के एक प्रवक्ता ने बताया कि इस मुठभेड़ में एक आतंकवादी मारा गया है. यह मुठभेड़ अभी जारी है.प्रवक्ता ने बताया कि पुलवामा के चन्दगाम में आज सुबह सुरक्षा बलों और छिपे हुए आतंकवादियों के बीच मुठभेड़ शुरू हो गई।माना जा रहा है कि गांव में लश्कर-ए-तैयबा के दो आतंकवादी छिपे हुए हैं।

(Translation: A militant was killed on Friday in an ongoing encounter with security forces in Pulwama district of eroded Kashmir. A police spokesman said a militant was killed in the encounter. The encounter is still going on, the spokesperson said, adding that an encounter between security forces and hidden militants started this morning at Chandgam in Pulwama. Two LeT militants are believed to be hiding in the village.)

Headline (ground truth): कश्मीर के पुलवामा में मुठभेड़, एक आतंकी ढेर
(Translation: Encounter in Pulwama, Kashmir, a terrorist killed)

Headline (zero-shot generated output:) पुलवामा में जारी मुठभेड़ में एक आतंकवादी ढेर
(Translation: A terrorist killed in ongoing encounter in Pulwama)

Figure 1: Zero-shot news headline generation from ZmBART in Hindi language

1 Introduction

Recent advancement in natural language generation (NLG) is heavily oriented towards large annotated training data. Such large task-specific annotated data is available for high resource language (HRL) like English. The tasks become challenging when limited training data is available. This is often observed for low-resource languages (LRLs) like Hindi, Japanese, etc. Manually annotating large data is time-consuming, expensive and uninteresting. This limits the model development and product deployment for LRLs. Moreover, despite large active research in cross-lingual representation learning (Hu et al., 2020; Conneau et al., 2020; Lewis et al., 2020b), the area of cross-lingual transfer and generation is relatively under-explored. Motivated by these factors, we propose a novel framework to transfer supervision from HRL to LRLs where model is trained on one language and directly evaluated for unseen languages. This enables cross-lingual transfer and generation for low resource languages in zero and few-shot settings for different tasks. The framework can be easily extended to other tasks and languages.

We carefully selected four challenging NLG tasks i.e., news headline-generation (NHG), question generation (QG), abstractive text summarization (ATS) and distractor generation (DG) to validate the framework's performance. NHG and ATS require understanding of input passage to generate meaningful headline and summary respectively. QG task should accumulate information from a passage and answer to generate high-quality questions.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2804–2818
August 1–6, 2021.

Distractor generation is the task of generating incorrect options from reading comprehension MCQ. It is challenging because generated distractors should be in the context with question but should not be semantically equivalent to the answer. We consider two LRLs i.e., Hindi and Japanese from two different language families. English is selected as the HRL from which the learning would be transferred to the LRLs. All three selected languages are different in their syntactic structures and typologically diverse. As there is no established publicly available dataset for DG in Hindi, we also create a new DG dataset for Hindi called as **HiDG**[1].

Our proposed framework to achieve this transfer of supervision from HRL to LRL under multiple languages and multiple tasks is named as ZmBART. ZmBART is based on mBART (Liu et al., 2020), a pre-trained model for cross-lingual natural language generation (NLG). We *further* pretrain mBART with a novel auxiliary task. Then the trained model is fine-tuned on large task-specific supervised data in English and evaluated directly on Hindi and Japanese languages in zero/few-shot setting for the tasks under consideration. We observe that the auxiliary task plays a critical role on the model's performance and needs to be carefully designed. This framework can be directly applied to multiple cross-lingual generation tasks without even the need to modify model hyper-parameters. Figure-1 shows a zero-shot NHG sample output generated by the ZmBART model. Our main contributions in this work can be summarized as:

1. We propose a novel zero-shot cross-lingual generation framework called ZmBART without parallel data and without back-translation. The framework can be directly applied across multiple tasks without even modifications in hyper-parameter values.
2. We demonstrate the effectiveness of ZmBART on four cross-lingual generation tasks across three typologically diverse languages.
3. We have created *HiDG*, a high-quality distractor generation dataset for the Hindi language.

2 Related Work

Early works on cross-lingual generation rely on machine translation (MT). In the very first work, Wan et al. (2010) leveraged the MT pipeline for cross-language document summarization. They first translate the non-English test instances to English. This translated text is fed through the supervised model (trained with document summarization data in English) to generate English summaries. Finally, these summaries are translated back to the target language. Shen et al. (2018) and Duan et al. (2019) used MT systems to generate pseudo training data for cross-lingual summarization and news headline generation respectively. However these MT based models are not suitable for low resource languages as they do not share parameters across-languages and generated translations are error-prone.

Recently there are a few works in the direction of supervision transfer from HRL(s) to LRL(s) for language generation. Kumar et al. (2019) used back-translation (needs MT system) and annotated supervised data for cross-lingual question generation. Chi et al. (2020) used parallel data to train a sequence-to-sequence model for zero-shot cross-lingual abstractive text summarization and question generation. Lewis et al. (2020a) proposed a pre-training based on mono-lingual paragraphs. Then this pre-trained model is used for zero-shot abstractive text summarization (ATS) in multiple languages. They trained a model on the ATS dataset on all the languages except the test language. This approach needs annotated data in multiple languages. Existing supervision transfer methods require parallel data for the cross-lingual tasks. Either they use available parallel corpora directly, or they translate/ back-translate data to generate pseudo-parallel corpora. Both these approaches pose significant challenges, as task-specific parallel data for multiple languages is difficult to obtain, and MT are far from perfect, especially for low resource languages.

Unlike the previous approaches, we did not use any parallel data or back-translation in our proposed framework. We did not pre-train any model from scratch. Instead, we leveraged the existing pre-trained model mBART. We included four challenging generation tasks across three syntactically diverse languages. Even we did not modify any hyper-parameters across the tasks and languages. All these considerations make the framework simple and easy to use. Further, it enables the addition of different other languages and NLG tasks in the proposed framework a simple extension exercise.

---

[1] HiDG dataset download link: `https://github.com/kaushal0494/ZmBART`

3 Methodology

Figure 2 shows an outline of our proposed ZmBART framework. ZmBART is based on pre-trained mBART (Liu et al., 2020) model. In our framework, we take the mBART model and *further pre-train* it on an auxiliary task. The auxiliary task is designed in such a way that the objective function of auxiliary task is close to fine-tuning tasks and only utilizes the mono-lingual data from the selected languages. Similar to mBART model we use language identifier tag with slight modification. We concatenate $< fxx >< 2xx >$ tags in input data instance where xx indicates the language tag. Given an input sentence and the language tag the model encodes the sentence in multi-lingual space. By conditioning on the encoded representation and language tag the decoder generates output text in target language.

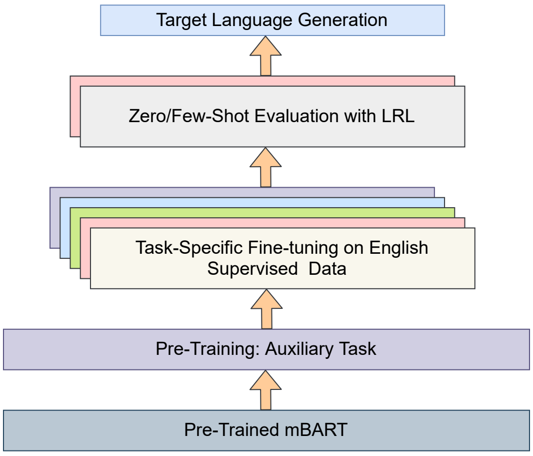

Figure 2: Architecture diagram of ZmBART

3.1 Multilingual BART (mBART)

Multilingual BART (Liu et al., 2020) is an extension of BART model (Lewis et al., 2020c) to multiple languages. It is a transformer-based sequence-to-sequence pre-trained model. The model is trained on monolingual data in many languages from Wikipedia Common Crawl corpus with BART language model objective. Particularly, The training data is concatenation of data from K languages i.e., $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2 \dots \mathcal{D}_K\}$ where $\mathcal{D}_i$ is a collection of monolingual documents in language i. They introduced two types of noises to corrupt the text: (1) random token span masking and (2) sentence order permutation. mBART is trained as denoising autoencoder. During training, the model has to predict text X from it's corrupted version $g(X)$, where g is noise function. The aim is to maximize the following objective function

$$\mathcal{L}_\theta = \sum_{\mathcal{D}_i \in \mathcal{D}} \sum_{x \in \mathcal{D}_i} logP(x|g(x); \theta), \qquad (1)$$

where x is a data instance of language i. Probability distribution P is defined by the sequence-to-sequence model. mBART gave state-of-the-art results in sentence and document level machine translations tasks. Details about mBART model can be found in Liu et al. (2020).

3.2 Unsupervised Auxiliary Task

Although the mBART pre-trained model encodes a multi-lingual latent space, it can not be used directly for cross-lingual generation. This is because the model is jointly trained on denoising objectives which do not directly follow auto-regressive decoding, thereby causing mismatch between pre-training and fine-tuning objectives. To overcome this problem, an unsupervised auxiliary task is introduced. We design the auxiliary task with the following desiderata in mind. It (1) should only utilize mono-lingual data from selected languages, (2) should enrich the mBART latent representations for selected languages and (3) train the decoder in pure auto-regressive manner with a training objective which is close to multiple fine-tuning tasks.

The auxiliary task in ZmBART is an additional pre-training step for *better warm-start* to downstream auto-regressive NLG tasks - although the final task (Distractor/Question/Summary generation) can be different from the auxiliary task. Additionally, this step allows the model to have a closer look at the languages under consideration and enrich/adjust the representations and parameters accordingly.

Outputs of the NLG tasks considered in this work are expected to contain words from different parts of the input. Generation of the output tokens are handled by the framework using an encoder-decoder setup. Hence we decide to have an auxiliary task that also encodes the input, and attends to this encoded representation to generate the output words in auto-regressive manner. This way, a single auxiliary task can help to enrich the token representations, warm up the encoder-decoder weights for fine tuning, and also caters to the multiple final output tasks. We define the auxiliary task as: *Given an input passage, generate few random sentences (called rand-summary) from the passage*. After experimentation we found that randomly generating 20% sentences from passage works the best.

Particularly, the input passage has length between 5-25 sentences and output is 1-5 random sentences from the passage. We do not assume any relations among sentences of the passage. We sample equal proportion of monolingual data from three languages. Data preparation steps for the auxiliary task are given below:

1. Generate a random number $k \in \{5. \cdots, 25\}$. k denotes the size of input passage
2. PASSAGE: Append k continuous sentences, starting from a random index of monolingual corpus D_i of the i^{th} language
3. RAND-SUMMARY: Randomly select 20% sentences from the passage
4. Repeat steps 1 to 3 for p languages
5. Repeat steps 1 to 4 for N times, to collect Np <PASSAGE, RAND-SUMMARY> pairs

3.3 Fine-Tuning on Downstream NLG Tasks

The proposed pre-trained model is directly fine-tuned on four downstream tasks: Question Generation (QG), News Headline Generation (NHG), Abstractive Text Summarization (ATS) and Distractor Generation (DG). First, the model is fine-tuned on large task-specific English supervised data and then this trained model is directly evaluated on Hindi and Japanese evaluation datasets in zero-shot setting. To validate the hypothesis that the ZmBART framework is robust across multiple tasks and languages, we did not modify any hyper-parameters during fine-tuning. It is often observed that including a few instances from LRL to supervised data boosts the model performance. To validate this point we further fine-tuned ZmBART with 1000 task-specific supervised data-points in Hindi and Japanese languages in few-shot setting which boosts the model performance.

3.4 Dealing with Catastrophic Forgetting and Spurious Correlation

During experimentation with the zero shot setup, it is observed that the model always generates the output text in English irrespective of input and language tag. We suspect this to be due to catastrophic forgetting problem (Van de Ven and Tolias, 2019). The supervised training completely overrides/erases the pre-trained learning. The generator (decoder) becomes biased towards English due to the explicit supervision learned from large task-specific English data. To overcome this problem, we freeze all word embeddings and all the parameters of decoder layers during fine-tuning with English data. Although this resolves the problem for NHG, QG and DG, the problem did not get completely resolved for the ATS task. We noticed that the zero-shot ATS output now is not completely in English, but it became of code-mix nature. In other words, the number of English words in the output reduced, but still lot many English words remained. The code-mixed outputs were logical and meaningful. We assume this to be due to spurious correlation issue, also reported in (Gu et al., 2019). To resolve this issue, we added a few examples (25 in number) of the auxiliary-task data during the fine-tuning step. This augmentation was helpful to address the spurious correlation issue for ATS. It is to be noted that the non-English data used for this augmentation is still of unsupervised and monolingual nature.

4 Experimental Setup and Results

We conduct experiments over four NLG tasks in three languages. We compare the performance of ZmBART with strong and MT pipeline based baseline models. We use both automated and manual evaluation metrics to evaluate model performances.

4.1 Baselines

Prior results are not available in literature for selected languages and datasets. Hence, for performance comparison, we developed several strong baselines based on recent models and architectures. Details of these baselines are mentioned below:

- **MT Pipeline (mBART):** Here, we fine-tune mBART on task-specific English data. Non-English test data instances are first translated into English and passed to the fine-tuned model. The output is translated back to the input language. Google Translator is used for translations.
- **mBART+MADMo:** This is an **mBART** based baseline where the auxiliary task has **M**asking **A**nd **D**enoising objective with **Mo**no-lingual data in three languages. The aim is to enrich the cross-lingual latent representation space of mBART for English, Hindi and Japanese.
- **mBART+MADPd:** Inspired from (Chi et al., 2020), we took **P**arallel **D**ata (English-Hindi and English-Japanese) and concatenate each parallel instances of two languages. Then we used this data with **M**asking **A**nd **D**oising objective to further train mBART. Including parallel data provides explicit supervision while generating Hindi and Japanese text.

4.2 Evaluation

We use both automated and manual evaluation metrics for performance comparison. Multiple metrics are used in literature for NLG tasks. Since we are considering multiple tasks, for brevity, against each task we only report values of the metrics commonly used by the community for that particular task. For automatic evaluation we used both lexical match (**BLEU** (Papineni et al., 2002) and **ROUGE** (Lin, 2004)) as well as embedding based evaluation metrics (**BERTScore** (Zhang et al., 2020)). To evaluate question generation and distractor generation tasks we use case-mix BLEU-4 (BL) score from sacreBLEU implementation, ROUGE-L (R-L) and BERTScore (BS). For ATS and NHG tasks ROUGE-1, ROUGE-2 and ROUGE-L are used.

We follow a similar approach for manual evaluation as Chi et al. (2020). We sampled 50 generated data points each for QG, ATS and NHG tasks in both Hindi and Japanese languages. We use three metrics: *Fluency* (Flu), *Relatedness* (Rel) and *Correctness* (Corr). **Fluency** measures *how fluent the generated text is*. **Relatedness** indicates *how much the generated outputs are in the context with input(s)*, **Correctness** measures *semantics and meaningfulness*. For DG, we use an additional metric called **Distractibility** that measures *the degree of confusion for generated incorrect options*. For DG task, there can be large number of good distractors for given input, in such situation the manual evaluation is more reliable. We sample 100 generated outputs for DG task. We employed large pool of evaluators from native Hindi and Japanese speakers to evaluate Hindi and Japanese output texts respectively. We asked each annotator to rate the generated texts on a scale of 1-5 (1 is very bad and 5 is very good) for all the metrics. We intentionally selected outputs of ZmBART and two best baselines to reduce the evaluators workload.

4.3 News Headline Generation (NHG)

In this task, *given a news article, we generate grammatically coherent, semantically correct and abstractive headline*. We use 500k/30k/30k (train/validation/test) English NHG data splits from *Gigaword* headline generation corpus[2]. For Hindi and Japanese we use 1k/1k/5k spilt from Kaggle[3] (we manually filtered high-quality news and headlines) and (Iwama and Kano, 2019) respectively.

In a zero-shot setting we fine-tune ZmBART model on supervised data and directly evaluate results on Hindi and Japanese test datasets. Automated evaluation results are included in Tables 1 and 2. We observe that, quality of generated headlines in Hindi is better compared to Japanese. The possible reasoning can be the input size. ZmBART outperforms the baseline with an absolute difference of 5.22 ROUGE-L score. mBART+MADMO is best among others which shows that masking and denoising with monolingual data indeed enrich the multi-lingual latent space for selected three languages. mBART+MADMO generates code mixed (Hindi-English or Hindi-Japanese) output which degrades the model performance. Few-shot training fills the mistakes of zero-shot models and generates better quality output. Manual evaluation scores (Tables 3 and 4) and automated scores correlate well validating ZmBART's performance on NHG task.

4.4 Question Generation (QG)

In the Question Generation (QG) task, *given an input passage and an answer, the aim is to generate semantically and syntactically correct questions that can produce the answer.* We use SQuAD 1.1 (Rajpurkar et al., 2016) English data for supervised training. SQuAD is popular question answering dataset consisting of 100k+ <passage, question, answer> tuples. Following (Zhao et al., 2018), we combine the train and validation sets of SQuAD and then spilt it as 80k/8k/10k training/validation/test tuples. For Hindi we use 1k/5.5k (train/test) from MLQA (Lewis et al., 2020d) and TyDiQA-GoldP (Clark et al., 2020) datasets. We use 1k/1k/5k for Japanese data from (Takahashi et al., 2019). Hindi and Japanese data are available in SQuAD data format which maintains consistency in terms of passage size, question and number of answers. For given passage and question we randomly selected one answer to form the dataset. We combine answer and passage as single input sequence separated by special token <s>.

Even without any parallel data, ZmBART outperformed all the baselines consistently across all automated evaluation metrics for zero-shot setting. Regarding manual evaluations, we see that Hindi questions received good score from the annotators, whereas the questions generated for the Japanese language inputs were considered as poor. Upon closer inspection of the generated text we find that several generated questions start with English wh-

---

[2] https://github.com/harvardnlp/sent-summary

[3] https://www.kaggle.com/disisbig/hindi-text-short-summarization-corpus

Model	**News Headline Generation**			**Question Generation**			**Abstractive TS**			**Distractor Generation**		
Metrics	**R-1**	**R-2**	**R-L**	**BL**	**R-L**	**BS**	**R-1**	**R-2**	**R-L**	**BL**	**R-L**	**BS**
Cross-lingual zero-shot generation results												
MT Pipeline(mBART)	16.61	4.91	15.83	2.6	21.31	71.53	11.15	3.11	10.93	1.6	9.66	67.35
mBART+MADMO	29.32	16.36	27.52	3.9	23.70	73.76	18.25	4.92	16.10	2.8	15.86	72.26
mBART+MADPD	24.02	13.41	23.29	4.3	25.29	73.74	10.47	2.55	12.30	2.9	15.43	72.89
ZmBART	**34.94**	**19.38**	**32.74**	**4.4**	**26.51**	**74.19**	**21.27**	**5.30**	**17.64**	**4.1**	**21.05**	**73.39**
Cross-lingual few-shot generation results (with 1000 supervised data points)												
ZmBART	52.37	35.52	50.50	7.6	34.11	78.29	36.29	14.21	27.22	6.5	26.58	78.27

Table 1: Zero and few-shot cross-lingual generation results for Hindi Language

Model	**News Headline Generation**			**Question Generation**			**Abstractive TS**		
Metrics	**R-1**	**R-2**	**R-L**	**BL**	**R-L**	**BS**	**R-1**	**R-2**	**R-L**
Cross-lingual zero-shot generation results									
MT Pipeline(mBART)	13.82	0.38	7.92	8.9	26.92	71.93	17.90	3.98	18.46
mBART+MADMO	33.75	8.12	17.78	16.6	34.80	74.01	28.74	9.01	23.63
mBART+MADPD	31.58	6.98	18.95	18.2	36.22	74.99	19.17	4.89	18.22
ZmBART	**35.25**	**9.24**	**19.92**	**18.8**	**38.74**	**75.91**	**36.60**	**15.26**	**29.85**
Cross-lingual few-shot generation results (with 1000 supervised data points)									
ZmBART	47.06	22.36	31.55	30.4	53.98	82.66	41.65	20.33	33.49

Table 2: Zero and few-shot cross-lingual generation results for Japanese Language

Model	**News Headline Generation**			**Question Generation**			**Abstractive TS**			**Distractor Generation**		
Metrics	**Flu**	**Rel**	**Corr**	**Flu**	**Rel**	**Corr**	**Flu**	**Rel**	**Corr**	**Flu**	**Rel**	**Dist**
Annotator set-01												
mBART+MADMO	3.86	4.34	3.94	2.66	3.38	3.52	3.56	3.58	3.22	3.61	4.08	2.89
mBART+MADPD	2.54	2.96	2.28	3.1	3.4	3.78	2.26	2.62	1.92	2.42	3.72	3.08
ZmBART	**4.14**	**4.22**	**4.04**	**3.24**	**3.44**	**3.9**	**4.02**	**4.12**	**3.54**	**4.12**	**4.19**	**3.83**
Annotator set-02												
mBART+MADMO	3.84	4.18	3.8	3.83	4.63	3.96	3.38	3.96	3.4	3.38	3.00	2.24
mBART+MADPD	2.96	3.02	2.7	**3.98**	4.70	3.98	2.96	3.16	2.84	2.97	3.11	**2.46**
ZmBART	**4.12**	**4.38**	**4.16**	3.95	**4.80**	**4.27**	**4.24**	**4.52**	**4.38**	**3.56**	**3.18**	2.36
Annotator set-03												
mBART+MADMO	3.56	3.74	**3.78**	2.68	3.76	3.32	2.9	3.34	2.9	3.96	3.74	3.12
mBART+MADPD	3.1	3.42	2.91	2.80	3.88	3.56	2.64	2.34	2.46	4.13	3.74	2.94
ZmBART	**3.70**	**3.84**	3.76	**2.86**	**4.04**	**3.76**	**4.06**	**3.56**	**3.56**	**4.44**	**4.12**	**3.12**

Table 3: Manual evaluation results of Zero-shot generated outputs for Hindi language

Model	**News Headline Generation**			**Question Generation**			**Abstractive TS**		
Metrics	**Flu**	**Rel**	**Corr**	**Flu**	**Rel**	**Corr**	**Flu**	**Rel**	**Corr**
Annotator set-01									
mBART+MADMO	2.66	2.98	2.50	1.98	**3.70**	**3.18**	3.04	3.55	3.44
mBART+MADPD	2.26	2.70	2.04	2.00	3.38	2.82	1.44	2.22	2.20
ZmBART	**3.60**	**4.02**	**3.50**	**2.12**	3.30	2.94	**4.24**	**3.90**	**3.90**
Annotator set-02									
mBART+MADMO	2.1	2.58	1.98	1.24	1.70	1.33	2.56	3.40	2.62
mBART+MADPD	1.58	1.78	1.46	**1.46**	1.72	1.78	1.00	1.00	1.00
ZmBART	**3.78**	**4.16**	**3.86**	1.26	**1.76**	**1.88**	**4.04**	**4.26**	**3.84**
Annotator set-03									
mBART+MADMO	2.24	2.72	2.24	**2.34**	2.46	2.39	2.82	3.18	3.52
mBART+MADPD	1.9	2.14	1.82	2.10	2.66	2.28	1.16	1.84	1.44
ZmBART	**2.88**	**3.22**	**2.92**	2.10	**2.70**	**2.46**	**3.32**	**3.52**	**3.04**

Table 4: Manual evaluation results of Zero-shot generated outputs for Japanese language

words. This mixing of English 'code' in the output happened somewhat seamlessly for the Hindi data as tokens in both languages are written in left-to-right manner. Moreover, Hindi-English code-mixed data is now getting very common and the annotators mostly accepted the mixing of the wh-words with the Hindi texts. Such mixing is not very common with Japanese text. As a result, the annotators assigned lower scores to such texts.

We then tried to understand the reason for getting the wh-words at the beginning of the output. English interrogative sentences often introduce wh-words at the beginning even though they are not present in the original data. The model gets exposed to such special characteristics of the English interrogative sentences during the fine tuning. The output from other languages get impacted due to this in zero-shot settings. However, the semantics of the text is captured well for the model as demonstrated by the high BERTScore, indicating good cross-lingual transfer of semantic knowledge.

4.5 Abstractive Text Summarization (ATS)

In Abstractive Text Summarization (ATS), we aim to *generate grammatically coherent, semantically correct and abstractive summary given an input document*. We use recently released WikiLingua (Ladhak et al., 2020) cross-lingual abstractive summarization dataset containing data in 18 languages. Prior splits are not available for this dataset. We use 131k/5k/5k (train/validation/test) splits for English, and 1k/1k/5k splits for Hindi and Japanese.

By skimming through data in Hindi we observe that many input documents consist of technical instructions on usage of softwares/tools. Summarizing these instructions are challenging. Zero-shot ZmBART performed better as compared to baselines as shown in human evaluation (Tables 3 and 4 for Hindi and Japanese respectively). The human evaluation results correlate with automated evaluation as shown in Tables 1 and 2. Ladhak et al. (2020) reported cross-lingual ATS score with same data for four different languages. The R-L score for four languages are 34.06, 37.09, 31.67 and 32.33. We obtain R-L scores of 27.22 and 33.49 for Hindi and Japanese respectively, which shows that the few-shot performance of ZmBART is acceptable.

4.6 Distractor Generation (DG)

The final task to judge ZmBART's performance is Distractor Generation (DG). It is the task of generating incorrect options (also known as distractors) from reading comprehension MCQ. The generated distractors should be in the context with the question but shouldn't be semantically equivalent to the answer. Formally, *for given passage, question and answer triplet, generate a long, coherent, and grammatically correct wrong option*. Considering the fact that for a given triplet there can be many incorrect options that are completely different from each other, the problem is even more challenging. We use English DG dataset from (Maurya and Desarkar, 2020) which consists of approx 135k/17k/17k (train/validation/test) split. We were unable to find a suitable dataset in Japanese language. For Hindi language we created a dataset called ***HiDG***[4] of 1k/1k/5k split. Similar to QG, to create input for ZmBART we concatenate the answer, question and passage in the same order and separate them with special token <s>.

To generate HiDG, we first extracted <passage, question, answer> triplets from English SQuAD 1.1 with atleast 150 tokens in the triplet. We generate distractors for these examples using model proposed by Maurya and Desarkar (2020). The distractors were translated to Hindi using Google Translator service. The translated distractors were manually verified or corrected (if necessary) by human annotators.

The evaluation of the task is challenging because: 1) there can be more then one correct distractors. Automated evaluation metrics may not able to capture this aspect as only one ground truth distractor is available and 2) it may possible that the generated distractor is semantically similar to answer with high lexical overlap with reference distractor in those situation lexical match based metrics are not suitable. To evaluate the DG task we mainly rely on BERTScore and manual evaluation. Towards this effort we consider higher number of DG samples for manual evaluation. Results from Tables 1 and 3 indicate the superiority of ZmBART over the baseline models for this task.

To summarize, we have performed experiments for 14 different task-setup combinations involving low resource languages. With four tasks in Hindi and three tasks in Japanese, and each task in zero shot and few shot setup, we provide detailed comparative evaluation for the tasks. The tasks are of different natures, and each task offers its own unique challenge. We critically analyze the performances to show the robustness and the range

---

[4] Implementation, dataset, pre-trained checkpoints and ZmBART generated text are available at `https://github.com/kaushal0494/ZmBART`

of applicability for the proposed ZmBART framework. We use fairseq library (Ott et al., 2019) for all the implementation and experiments. The implementation details are included in supplementary.

5 Results Analysis and Ablation Study

In this section, we provide further analysis of the experimental results. We also perform ablation studies to understand the impacts of the different modeling decisions made in designing the framework.

•**Supervised Training Results:** Table 5 shows the comparative results of fine-tuned mBART with and without auxiliary task on task-specific supervised English data. We observe that there is no significant performance degradation of ZmBART over original mBART model with pure supervised training. Even, the auxiliary task helps in achieving slight improvement over the original mBART performance in most setups. This concludes that ZmBART can be adopted as replacement of original mBART model with additional functionalities.

Task	Setting	BL	R-1	R-2	R-L	BS
NHG	W/ Aux-Task	15.9	43.22	21.33	40.88	90.13
	W/O Aux-Task	15.9	43.15	21.25	40.77	90.13
QG	W/ Aux-Task	20.6	53.20	26.53	51.37	92.18
	W/O Aux-Task	21.4	52.66	26.63	51.25	92.41
ATS	W/ Aux-Task	16.0	40.01	18.11	38.29	90.20
	W/O Aux-Task	15.8	39.52	18.00	37.91	90.10
DG	W/ Aux-Task	10.3	31.76	14.89	31.18	89.33
	W/O Aux-Task	10.0	31.87	14.59	31.30	89.42

Table 5: Automated evaluation results of mBART on task-specific supervised English dataset (with and without Auxiliary Task)

•**Effect of Auxiliary Task:** Table 6 includes the results with and without auxiliary task of ZmBART for ATS and QG tasks in zero-shot setting. It can be inferred that without the auxiliary task, lexical match based scores are poor because the decoder generates code-mixed outputs. We see that the BERTScore is still reasonable without auxiliary task owing to the multilingual mBART embedding. However, generation of the data in appropriate language is enabled only after inclusion of the auxiliary task. The auxiliary task contributes in two ways: it enables zero-shot generation and improves the mBART multilingual latent space even more as indicated by the improved BERTScore.

With these results we now want to understand *whether the auxiliary task is able to generalize across multiple tasks, or favors specific tasks.* Among the tasks considered in this work, we see that generation of meaningful summaries/headlines require understanding/abstracting of input text which is unlikely to be obtained by repeating sentences from input passages, as done in the auxiliary task. ZmBART achieves good zero-shot/few-shot/supervised results (Tables 1-5) on ATS and NHG over strong baselines. The generated headlines and summaries were found to be mostly abstractive, they don't contain large continuous sequences from input text. As described in Sections 4.4 and 4.6, Question Generation and Distractor Generation are more challenging tasks and have objectives vastly different from the auxiliary task's objective. Even for these tasks, decent evaluation scores (Tables 1-5) and improvements over the baselines across the languages considered indicate that the solutions are not spurious. Incorporation of auxiliary task improves the performance of diverse downstream tasks on real benchmark datasets, and does not favor any specific task or dataset.

Model	Abstractive TS			Question Generation		
Metrics	R-1	R-2	R-3	BL	R-L	BS
Hindi Language						
ZmBART w/o Aux	4.34	0.10	3.19	0.9	16.64	70.72
ZmBART with Aux	21.27	5.30	17.64	4.4	26.51	74.19
Japanese Language						
ZmBART w/o Aux	6.80	0.11	5.30	6.7	33.07	70.35
ZmBART with Aux	36.60	15.26	29.89	18.8	38.74	75.91

Table 6: Zero-shot results of ZmBART with and without auxiliary task for Hindi and Japanese

- **Approaches to avoid Catastrophic Forgetting:** We use two approaches to address the catastrophic forgetting problem, (a) Freezing model components and (b) optimized regularization (Van de Ven and Tolias, 2019). Tables 7 and 8 show the automated evaluation results with different approaches used to deal with the catastrophic forgetting problem. It can be noted that the proposed modelling setup (i.e., ZmBART) gives best results.

- **Effect of Architecture on Few-shot Training:** In this set-up we experiment with few-shot training with mBART (directly fine-tuned on task-specific supervised English data) and ZmBART (trained with auxiliary task and fine-tuned with English data). The results are presented in Table 9. We find that ZmBART does better than mBART in corresponding setups. Moreover, although freezing the decoder layer and word embeddings helps in zero-shot setting, it is natural and useful to unfreeze them during few shot training.

- **Few-shot performance with Supervised data:** Figures 3 and 4 show the trends of few-shot training of ZmBART with respect to supervised Hindi and Japanese training data for ATS

Setup	Setting-Details	BL(hi/ja)	R-L(hi/ja)	BS(hi/ja)
Model Components	Freeze word embedding (WE)	2.5/13.6	21.55/31.99	72.02/73.18
	Freeze WE + subset of Encoder & Decoder layers	2.9/15.3	22.62/36.60	72.24/72.98
	Freeze WE + Encoder layers	2.2/13.8	19.69/36.91	69.73/72.97
	Freeze WE + Decoder layers (ZmBART)	**4.4/18.8**	**26.51/38.74**	**74.19/75.91**
Regularized Optimization	Elastic Weight Consolidation (EWC)	2.1/11.6	18.21/29.47	68.36/72.91

Table 7: Evaluation scores for different modeling approaches to avoid catastrophic-forgetting for QG Task

Setup	Setting-Details	R-1(hi/ja)	R-2(hi/ja)	R-L(hi/ja)
Model Components	Freeze word embedding (WE)	13.02/26.07	05.67/03.96	12.45/17.62
	Freeze WE + subset of Encoder & Decoder layers	14.27/25.72	06.70/03.21	13.76/18.28
	Freeze WE + Encoder layers	09.81/22.67	04.10/02.38	09.66/13.68
	Freeze WE + Decoder layers (ZmBART)	**34.94/35.25**	**19.38/09.24**	**32.74/19.92**
Regularized Optimization	Elastic Weight Consolidation (EWC)	12.01/22.16	05.43/03.11	11.22/16.31

Table 8: Evaluation scores for different modeling approaches to avoid catastrophic-forgetting for NHG Task

Model	NHG			QG		
Metrics	R-1	R-2	R-3	BL	R-L	BS
mBART+WE	50.61	34.32	49.01	6.1	31.20	77.01
mBART	51.49	35.04	49.64	7.1	32.96	77.61
ZmBART+WE	51.81	35.04	50.07	6.9	32.82	77.40
ZmBART	**52.37**	**35.52**	**50.50**	**7.9**	**34.49**	**78.39**

Table 9: Hindi language few-shot results for different architectural setups. *WE* indicates that word embeddings and decoder layer parameters are frozen

and QG tasks respectively. We observe that with a small number of supervised examples (e.g. 100) the model achieves decent few-shot performance. We found the trends for different tasks to be similar. The improvement in model performance tends to be minimal after 1000 examples.

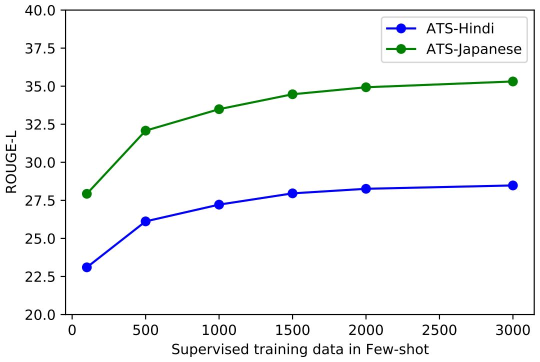

Figure 3: ZmBART model few-shot performance with supervised Hindi/Japanese data for ATS task

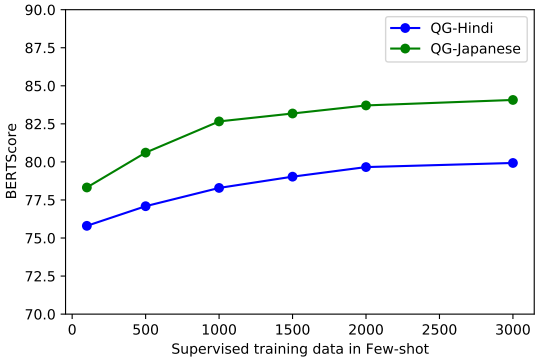

Figure 4: ZmBART model few-shot performance with supervised Hindi/Japanese data for QG task

6 Conclusion

In this paper, we propose a novel unsupervised framework (ZmBART) for cross-lingual transfer and generation. The framework transfers supervision from HRL to LRLs which enables zero-shot language generation. The framework does not use any direct or pseudo-parallel data. ZmBART is directly applied to multiple generation tasks and languages. The model includes a carefully designed auxiliary task that further improved the multilingual embedding space, and helped to initialize encoder-decoder weights to enable zero shot language generation. We performed experiments in three languages and 18 task-setup combinations: four supervised tasks in English, four tasks in Hindi (each with zero-shot and few-shot), and three tasks in Japanese (each with zero-shot and few-shot). Except zero-shot question generation tasks, for all other tasks involving LRLs, the proposed model generated good quality results as validated by automated and manual evaluation measures. In future we want to extend this work by adding multiple other languages and tasks, and also explore other choices of auxiliary tasks for better model transfer.

Acknowledgments

We thank the support from Nvidia AI Technology Center (NVAITC) towards the requirements of computing power and compute infrastructure. We thank the human annotators for human evaluation and the anonymous reviewers for their constructive feedback.

References

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2020. Cross-lingual natural language generation via pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7570–7577.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. In *Transactions of the Association of Computational Linguistics*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Xiangyu Duan, Mingming Yin, Min Zhang, Boxing Chen, and Weihua Luo. 2019. Zero-shot cross-lingual abstractive sentence summarization through teaching generation and attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3162–3172, Florence, Italy. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. 2019. Improved zero-shot neural machine translation via ignoring spurious correlations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1258–1268, Florence, Italy. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Kango Iwama and Yoshinobu Kano. 2019. Multiple news headlines generation using page metadata. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 101–105, Tokyo, Japan. Association for Computational Linguistics.

Vishwajeet Kumar, Nitish Joshi, Arijit Mukherjee, Ganesh Ramakrishnan, and Preethi Jyothi. 2019. Cross-lingual training for automatic question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4863–4872, Florence, Italy. Association for Computational Linguistics.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. 2020. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048, Online. Association for Computational Linguistics.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida I. Wang, and Luke Zettlemoyer. 2020a. Pre-training via paraphrasing. *CoRR*, abs/2006.15020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020c. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020d. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Kaushal Kumar Maurya and Maunendra Sankar Desarkar. 2020. Learning to distract: A hierarchical multi-decoder network for automated generation of long distractors for multiple-choice questions for reading comprehension. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1115–1124.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shi-qi Shen, Yun Chen, Cheng Yang, Zhi-yuan Liu, and Mao-song Sun. 2018. Zero-shot cross-lingual neural headline generation. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 26(12):2319–2327.

Norio Takahashi, Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2019. Machine comprehension improves domain-specific Japanese predicate-argument structure analysis. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 98–104, Hong Kong, China. Association for Computational Linguistics.

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Xiaojun Wan, Huiying Li, and Jianguo Xiao. 2010. Cross-language document summarization based on machine translation quality prediction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 917–926, Uppsala, Sweden. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

7 Supplementary Materials

7.1 Implementation Details:

We use a standard sequence-to-sequence Transformer architecture with 12 layers (each 16 heads) for encoder and decoder. The model has a dimension of 1024 (approx 680M parameters). Additional layer-normalization was used with both the encoder and decoder. We found FP16 precision stabilized the training. We trained all the models on 4 Nvidia V100 GPUs (32GB). Similar to mBART we use the Adam optimizer ($\epsilon = 1e$-6, $\beta_2 = 0.98$) and linear learning rate decay scheduling. The training started with a dropout value 0.3 and was later reduced to 0.2 after 20k steps and 0 after 40k steps. The loss function was cross-entropy label smoothing loss. 2500 warm-up steps and $3e$-5 learning rate were used. The model selection was done based on validation data likelihood. We use beam-search with beam size 5 in the decoding for all the tasks. We loaded mBARTCC25 pre-trained checkpoint weights and further pre-train/fine-tune model on task-specific data with teacher forcing method.

The above set of parameters are used for all the target tasks as well as the auxiliary task. We process different batch sizes of input for different tasks. We use 2048, 3000, 4096, 2048, and 5000 tokens per GPU for ATS, DG, QG, auxiliary, and NHG tasks, respectively. We use shared Byte Pair Encoding (BPE) vocabulary from sentencepiece tokenizer of size 250k. We use 34k/1k/1k (train/validation/test) data-points for auxiliary language (approx 11333 from each languages). We train the mBART model with the auxiliary task around 10k steps. Training time for the auxiliary task is around 2-3 hours. The fine-tuning times for TS, QG, NHG, and DG were around 4-5, 1-2, 1-2, and 2-3 hours. We observe a longer fine-tuning time for ATS because of long passages. We selected the best model based on loss and perplexity on the validation datasets. We checked with early-stopping and other checkpoints, which resulted in poor performance.

7.2 Evaluation Metric and Tokenizer Details:

For Automated evaluation, we use sacreBLEU implementation, ROUGE-L, and BERTScore. For ATS and NHG tasks, ROUGE-1, ROUGE-2, and ROUGE-L are used. We explicitly use community-adopted language specific-tokenizers. Links for language-specific tokenizers are given below:

- **English:** Default sacreBLEU tokenizer i.e, `https://github.com/mjpost/sacrebleu`
- **Hindi:** `https://anoopkunchukuttan.github.io/indic_nlp_library/`
- **Japanese:** `http://www.phontron.com/kytea/`

Links of publicly available implementations of automated evaluation metrics which we use directly in this work:

- **BLEU:** `https://github.com/mjpost/sacrebleu`
- **ROUGE:** `https://github.com/pltrdy/files2rouge`
- **BERTScore:** `https://github.com/Tiiiger/bert_score`

7.3 Few Zero-shot Generated outputs from ZmBART:

In the next few figures, we present sample outputs generated by the model in zero-shot setups, for Hindi and Japanese languages.

+++++++ News Headline Generation ++++++

News: वेनेजुएला के राष्ट्रपति ह्यूगो शावेज अपने देश में जरूरत से ज्यादा शराब पीने वालों पर लगाम कसना चाहते हैं. उन्होंने सेना को अवैध शराब बेचने वालों के खिलाफ कड़ी कार्रवाई करने का आदेश दिया है.ज ने कहा कि उनकी सरकार मदिरा और सिगरेट पर कर बढ़ाने का विचार कर रही है. वामपंथी राष्ट्रपति को बीयर और स्कॉच व्हिस्की जरा भी पंसद नहीं है. शावेज ने तीन वर्ष पहले भी शराब और सिगरेट पर कर बढ़ाया था.

Headline (human-generated:) वेनेजुएला में सिगरेट और शराब पर बरसे शावेज
Headline (model-generated:) शराब पीने वालों पर लगाम कसना चाहते हैं शावेज

News: तमिलनाडु में गणतंत्र दिवस पूरे धूमधाम से मनाया गया और राज्यपाल सुरजीत सिंह बरनाला ने यहां राष्ट्रीय ध्वज फहराया और मुख्यमंत्री एम करूणानिधि ने वीरता पुरस्कार प्रदान किए.बरनाला ने मरीना में पारंपरिक मार्च पास्ट की सलामी ली.इस मौके पर करूणानिधि ने वीरता के लिए अन्ना पुरस्कार, कोट्टाई अमीर सांप्रदायिक सद्भाव पुरस्कार तथा गांधी अदिगल पुलिस पदक प्रदान किए.राज्य निर्वाचन आयोग कार्यालय तथा दक्षिण रेलवे कार्यालय में भी गणतंत्र दिवस समारोह मनाया गया.

Headline (human-generated:) तमिलनाडु में धूमधाम से मनाया गया गणतंत्र दिवस
Headline (model-generated:) तमिलनाडु में गणतंत्र दिवस पूरे धूमधाम से मनाया

+++++++ Abstractive Text Summarization ++++++

Document: ब्रिज एक और कोरस की तरह है जो केवल एक बार गाया जाता है और आपके गीत के विषय को नए तरीक़े से प्रस्तुत करता है। अपने ब्रिज का प्रयोग गीत को रोचक बनाने के लिए नए छंदों को नई कुंजी में या एक ही कुंजी में अलग-अलग कॉर्ड्स के साथ गाकर करें। सुनिश्चित करें कि आपके ब्रिज के शब्द आपके कोरस के शब्दों की तरह अस्पष्ट हों। नई बारीकियों को पेश न करें। यदि आप किसी विशेष वाद्य के साथ अपने कौशल को पेश करना चाहते हैं तो आप अपने ब्रिज को वाद्य यंत्र सोलो के अवसर के रूप में उपयोग करने पर भी विचार कर सकते हैं। आज प्रयोग में आनेवाली सबसे आम गीत संरचना है छंद/कोरस/श्लोक/ कोरस/ब्रिज/कोरस। लेकिन, आप इस संरचना को बदल कर देख सकते हैं कि आपके गीत के लिए सबसे अच्छा क्या है। उन तत्वों को लें जिन्हें आपने पहले ही बनाया है और उन्हें अदल-बदल कर प्रयोग करें, उनमें से कुछ को तब तक दोहराएं, जब तक संरचना सही न हो जाये। कुछ शैलियां विशिष्ट गीत संरचनाओं का उपयोग करती हैं। उदाहरण के लिए, ईडीएम अक्सर परिचय/छंद/ कोरस/ ब्रेकडाउन/छंद/कोरस/छंद/कोरस/ब्रिज/कोरस/आउट्रो का उपयोग करता है। एक बार जब आप अपना गीत लिखना समाप्त कर लें, तो आप वाद्य जैसे ड्रम, बास गिटार और की-बोर्ड को स्वर-माधुरी के लिए ड्राइव और स्वरोच्चारण से जोड़ सकते हैं। आपके अन्य वाद्यों को उसी की और टाइम सिग्नेचर में बजाया जाना चाहिए जिसे आपने पहले तय किया था। यदि आपको अन्य वाद्यों को बजाना नहीं आता है, तो अपने कंप्यूटर का प्रयोग करके गीत की नींव रिकॉर्ड करने का प्रयास करें, फिर गीत मंत नए तत्व जोड़ने के लिए एबलेटन या गैरेजबैंड जैसे संगीत सॉफ़्टवेयर का प्रयोग करें। अपने गीत के अंशों का अलग-अलग तब तक अभ्यास करें जब तक आप उनमें से हरेक को याद नहीं कर लेते हैं। फिर, उन सभी को सही क्रम में एक साथ अभ्यास करने के लिए आगे बढ़ें जब तक कि आप इसके बारे में सोचे बिना एक तत्व से अगले तक आसानी से ट्रांज़िशन कर सकें। एक बार जब आपको गीत याद हो जाए, तो आपको इसे रिकॉर्ड करना चाहिए। अपने फ़ोन, डिजिटल रिकॉर्डर, लैपटॉप और सॉफ्टवेयर, या वीडियो कैमरे का प्रयोग करें। जब आप अपनी रिकॉर्डिंग कर लें, तो इसकी प्रतिलिपि बनाना या क्लाउड पर अपलोड करना सुनिश्चित करें। उस तरह आप अपना गीत न तो कभी भूलेंगे और न ही उसे खोएंगे।

Summary(Human-generated): निर्णय करें कि आप अपने गीत में ब्रिज जोड़ना चाहते हैं या नहीं: अपने गीत की अंतिम संरचना को सुदृढ़ करें: अधिक पूर्ण ध्वनि बनाने के लिए अन्य वाद्य जोड़ें: याद करने तक अपने गीत का अभ्यास करें: अपना गीत रिकॉर्ड करें:

Summary (model generated): अपने ब्रिज का प्रयोग गीत को रोचक बनाने के लिए नए छंदों को नई कुंजी में या एक ही कुंजी में अलग-अलग कॉर्ड्स के साथ गाकर करें। अपने गीत के अंशों का अलग-अलग तब तक अभ्यास करें जब तक आप उनमें से हरेक को याद नहीं कर लेते हैं।

Figure 5: Sample outputs for zero-short NHG and ATS in Hindi language

+++++++ Question Generation ++++++

Passage: १९५३ की उत्तरी सागर बाढ़ ब्रिटेन में दर्ज की गयी सबसे विनाशकारी प्राकृतिक आपदाओं में से थी। १,६०० किमी लम्बा समुद्र तट क्षतिग्रस्त हो गया और समुद्री दीवार विच्छेदित हो गयी जिससे १,००० वर्ग किमी का क्षेत्र जलमग्न हो गया। बाढ़ के कारण ३०,००० लोगों को उनके घरों से हटाना पड़ा और २४,००० संपत्तियां क्षतिग्रस्त हो गयीं। अलग-अलग घटनाओं में फेलिक्सस्टोव, सफ्फोल्क में ३८ लोग मारे गए जब वेस्ट एंड क्षेत्र में पूर्वनिर्मित घर बाढ़ की चपेट में आ गए। एस्सेक्स, कैनवे द्वीप पर ५८ लोग मारे गए और ३७ अन्य समुद्र किनारे के ग्राम जेविक में मारे गए। ब्रिटेन में भूमि पर कुल मारे गए लोगों की संख्या ३०७ थी और ब्रिटेन के समुद्रों में एम वी प्रिंसेस विक्टोरिया समेत मारे गए कुल लोगों की संख्या २२४ थी।

Answer: ३०,०००
Question(Human Generated): बाढ़ के कारण कितने लोगों को निकाला गया?
Question (Model Generated): कितने लोगों को उनके घरों से हटाना पड़ा?

+++++++ Distractor Generation ++++

Passage: आजकल महिलाओं में पुरुषों की तुलना में स्वयं को सुरक्षित ड्राइवरों के रूप में देखने की एक सकारात्मक छवि है। बीमाकर्ता metlife के एक सर्वेक्षण में 51 प्रतिशत महिलाओं ने कहा कि वे अधिक सुरक्षित तरीके से ड्राइव करते हैं। साक्ष्य उनके पक्ष में हैंः लापरवाह ड्राइविंग के लिए पुरुषों की तुलना में 3।4 गुना अधिक संभावना है और मादक ड्राइविंग के लिए 3।1 गुना अधिक संभावना है। महिलाओं में औसतन कम आक्रामक और कानून का पालन करने वाले ड्राइवर होते हैं जिसके कारण दुर्घटनाएं कम होती हैं। रिपोर्ट के अनुसार सभी पुरुषों की एक ही राय नहीं है। metlife द्वारा सर्वेक्षण किए गए पुरुषों में 39 प्रतिशत का दावा है कि पुरुष ड्राइवर सुरक्षित हैं। निष्कर्ष उन्हें एक ही बिंदु पर वापस दिलाते हैंः ऑटोमोटिव ज्ञान। रिपोर्ट से पता चला है कि अधिकतर पुरुष वर्तमान सुरक्षा उपकरणों जैसे इलेक्ट्रानिक स्थिरता नियंत्रण से परिचित हैं जो उन्हें वापस आने वाली दुर्घटनाओं को रोकने में सहायता करते हैं। ऑटो सुरक्षा अपरिहार्य रूप से धन व्यय का विषय है। बीमा कम्पनियां इस बात पर ध्यान देती हैं कि किस श्रेणी के ड्राइवरों के पास सबसे कम डॉलर का क्लेम होता है और अब के लिए जिसमें मुख्य रूप से महिलाएं शामिल हैं। सामान्य रूप से महिलाएँ ऑटो बीमा के लिए पुरुषों से लगभग 9 प्रतिशत कम भुगतान करती हैं। वेबसाइट insweb द्वारा किए गए एक अध्ययन से भी पता चलता है कि अधिकांश राज्यों में महिलाओं के लिए ऑटो बीमा की दरें कम हैं। अलग अलग राज्यों में महिलाओं को सबसे अधिक लाभ ग्वैमिंग (जहाँ वे 20 प्रतिशत कम भुगतान करते हैं) दक्षिण डकोटा और वाशिंगटन डी। सी। में 16 प्रतिशत कम बीमा लागत होती है। अध्ययन के अनुसार 2009 में 11900 से अधिक पुरुष ड्राइवर यातायात दुर्घटनाओं में मारे गए जबकि अध्ययन के अनुसार केवल 4900 महिला ड्राइवर ही मारे गए। मीलों की यात्रा करने पर पुरुषों की तुलना में प्रति 100 मिलियन मील की यात्रा करने पर 2 मौतें और महिलाओं के लिए 1।7 मौतें होती हैं।

Question: बीमा कंपनियां शायद महिला चालक ग्राहकों पर ध्यान केंद्रित क्योंकि वे
Answer: बीमा दावों पर सबसे कम धनराशि हो
Distractor(human-generated): वे अधिक सावधान ड्राइवर होते हैं और कम नुकसान करते हैं
Distractor(model-generated): वे बीमा दावों पर सबसे कम क्लेम करते हैं

Figure 6: Sample outputs for zero-short QG and DG in Hindi language

+++++++ News Headline Generation ++++++

News: 園児 による 夏休み 作品展 が 恵那市大井町 の 恵那郵便局 展示 ルーム で 開か れ て いる 。 十三日 まで
Headline (human-generated:) 貯金箱 や 切り絵 園児 の 力作 <NUM> 点
Headline (model-generated:) 園児による 夏休み展開か れ て いる

News: 「 少年院 で 自分 を 見つめ 直し た のに ... 。 被害者 に 申し訳ない 。 親 に も 惨め な 思い を さ せ た 」 –
Headline (human-generated:) 自分 が 甘かっ た
Headline (model-generated:) 少年院で 自分を 見つめ 直し たのに ...

+++++++ Abstractive Text Summarization ++++++

Document: 加齢による肌の変化とともに、必要とするスキンケアも変わっていきます。使い慣れていた基礎化粧品が今までのような効果を発揮しなくなったと感じたら、スキンケアを見直す時機と言えるでしょう。保湿剤にはローションタイプとクリームタイプがあります。
Summary(Human-generated): 肌質と年齢に適した保湿剤を選びましょう。
Summary (model generated): 加齢による肌の変化とともに、必要とするスキンケアも変わっていきます.

Document: 肌は潤った状態では柔軟性が増すため、動かしたり横に引っ張ったりしてもあまりダメージを受けません。しかし、肌がきちんと潤っていない状態で洗顔すると、皮膚にダメージを与えてしまいます。その上、正しいスキンケアを怠ると、このようなダメージが徐々に蓄積されてしまい、結果的にシワの形成につながります。 肌には、熱湯や冷水ではなくぬるま湯を使いましょう。
Summary(Human-generated): クリーム状でオイルフリーの洗顔料を使い、肌に潤いを与えながら不純物を取り除きましょう。
Summary (model generated): 正しいスキンケアを怠ると、シワの形成につながりますI

+++++++ Question Generation ++++++

Passage: 水たまりを見たら・・出来れば避けて通る。Uターンする、などの対処をして安全なドライブを心掛けてくださいね。後ろから車が来たとしても・・深さは分からないですから・・と教えてあげて・・何とか・無事に帰れる道を探してください。
Answer: 出来れば避けて通る
Question(Human Generated): 水たまりを見たらどうするか?
Question (Model Generated): 水たまりを見たら?

Passage: ゆっくりと弧を描いて後ろに飛んで行くカーナビのモニター。あっ、やっぱり、ぶつかったんだ。そうか、私が乗ってる車でも、ぶつかる事があるんだ。。。。生まれてこの方、1度も自分が乗ってる車は、事故を起こした事がなかったので、私は事故に合わないもんだと思ってた(苦笑)
Answer: 事故を起こした事がなかった
Question(Human Generated): 今まで事故を起こしたことがありましたか?
Question (Model Generated): 事故を起こした事がなかった?

Passage: 自分が交通弱者になったときは駐車場の前は特に注意して歩く。脇道から出るときにアクセルを踏み過ぎて(ステアリング操作がついてこない)対向車線まではみ出す車一時停止や赤信号で停止線を超えて止まるのはヘタの典型だな。
Answer: 駐車場の前は特に注意して歩く
Question(Human Generated): 自分が歩行者の立場のとき気を付けることは?
Question (Model Generated): 自分が交通弱者になったときは?

Figure 7: Sample outputs for zero-shot NHG, ATS and QG in Japanese language

HacRED: A Large-Scale Relation Extraction Dataset Toward Hard Cases in Practical Applications

Qiao Cheng[1*], **Juntao Liu**[1*], **Xiaoye Qu**[2], **Jin Zhao**[1], **Jiaqing Liang**[1],
Zhefeng Wang[2], **Baoxing Huai**[2], **Nicholas Jing Yuan**[2], **Yanghua Xiao**[1,3†]
[1]Shanghai Key Laboratory of Data Science,
School of Computer Science, Fudan University, China
[2]Huawei Cloud, China
[3]Fudan-Aishu Cognitive Intelligence Joint Research Center, Shanghai, China
{cq.qiaojim, l.j.q.light}@gmail.com, {jtliu19, shawyh, jinzhao20}@fudan.edu.cn,
{quxiaoye, wangzhefeng, huaibaoxing, nicholas.yuan}@huawei.com

Abstract

Relation extraction (RE) is an essential topic in natural language processing and has attracted extensive attention. Current RE approaches achieve fantastic results on common datasets, while they still struggle on practical applications. In this paper, we analyze the above performance gap, the underlying reason of which is that practical applications intrinsically have more hard cases. To make RE models more robust on such practical hard cases, we propose a case-oriented construction framework to build a **H**ard **C**ase **R**elation **E**xtraction **D**ataset (**HacRED**). The proposed HacRED consists of 65,225 relational facts annotated from 9,231 documents with sufficient and diverse hard cases. Notably, HacRED is one of the largest Chinese document-level RE datasets and achieves a high 96% F1 score on data quality. Furthermore, we apply the state-of-the-art RE models on this dataset and conduct a thorough evaluation. The results show that the performance of these models is far lower than humans, and RE applying on practical hard cases still requires further efforts. HacRED is publicly available at `https://github.com/qiaojiim/HacRED`.

Case in WebNLG
Elliot See was born on July 23rd , 1927 in ***Dallas***, and died in ***St. Louis*** on February 28th , 1966 .

Triples
Elliot See , `place_of_birth`, *Dallas*
Elliot See , `place_of_death`, *St. Louis*

Case in Practice
Yang Jima (1986 -), ..., is a student of 2005 in the Department of …, ***Communication University of China*** ... In the semi-final of the ***Chinese Idol Show***, ***Yang*** excellently performed the ***Lhasa Ballad***, which was recognized by the judges and the audience. As a result, she got to the final competition.

Triples
Yang Jima(Yang) , `graduate_from`, *Communication University of China*
Lhasa Ballad , `singer`, *Yang Jima*
Yang Jima(Yang) , `invited_guest_of`, *Chinese Idol Show*

Figure 1: Cases and corresponding triples in WebNLG and practical applications.

1 Introduction

Relation extraction (RE) is one of the core NLP tasks and plays an increasingly important role in knowledge graph completion (Bordes et al., 2013) and question answering (Dong et al., 2015). RE aims to extract structured relational facts, i.e., triples such as (*Bill Gates*, `founder_of`, *Microsoft*) from plain texts. Recently, various models (Zeng et al., 2018; Takanobu et al., 2019; Fu et al., 2019; Wei et al., 2020) have been proposed to identify the relational facts and achieved state-of-the-art (SOTA) performance, among which the latest method CasRel achieves notable 91.8% F1 score on WebNLG (Gardent et al., 2017) and 89.6% on NYT (Riedel et al., 2010).

However, can these seemingly fantastic results prove that the current RE models are powerful enough to perform well in practical applications. To answer the question, we employ CasRel on 300 randomly selected samples of WebNLG and the same number of data from practical DuIE[1]. The F1 scores under these scenarios drop significantly from 89.3% to 62.8%. As illustrated in Figure 1, CasRel extracts correct triples (*Elliot See*, `place_of_birth`, *Dallas*) and (*Elliot See*, `place_of_death`, *St. Louis*) in WebNLG where keywords such as *born* and *died* explicitly express the relation information. In contrast, CasRel fails to extract triples such as (*Yang Jima*, `graduate_from`, *Communication University of China*) where no keywords like *graduate* are mentioned. The most significant reason why CasRel performs well on WebNLG but struggles on practical data is that more challenging instances which

---

[*] Equally contributed.
[†] Corresponding author.

[1] http://lic2019.ccf.org.cn/kg

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2819–2831
August 1–6, 2021.

we refer to as hard cases exist in the practical applications. Moreover, according to the statistics of entity description documents in CN-DBpedia (Xu et al., 2017), at least 40.1% relational facts can only be extracted from hard cases. Therefore, relation extraction from hard cases can not be neglected and demands more attention.

Although many datasets (Li et al., 2016; Yao et al., 2019) have been proposed for RE, they rarely analyze the performance gap and focus on the hard cases. In order to make models robust on hard cases and more fit practical scenarios, in this paper, we aim to build a RE dataset with sufficient hard cases. To this end, we propose a case-oriented construction framework based on the challenging instances and build a **Ha**rd **C**ase **R**elation **E**xtraction **D**ataset (**HacRED**). Specifically, we first obtain general, massive, and various contexts as well as relational facts from CN-DBpedia to construct a distantly supervised dataset. The crucial part is to distinguish hard cases from abundant data. Therefore, we formulate nine indicators through systematic analysis of hard cases to quantify them. Then, we conduct feature engineering based on the valid indicators. Afterwards, a classifier is trained for distinguishing the desired hard cases. Finally, we develop a crowdsourcing platform with a novel three-stage annotation strategy and effective aggregation method CrowdTruth2.0 (Dumitrache et al., 2018) to guarantee the data size and quality.

In total, HacRED consists of 9,231 instances with 26 predefined relations and 9 types of entities. To the best of our knowledge, it is one of the largest document-level RE benchmark. Moreover, HacRED contains sufficient and diverse hard cases in line with practice. We conduct extensive experiments and systematic error analysis of SOTA models on HacRED. A sharp performance drop on HacRED compared to the existing benchmarks proves that RE in practical applications remains an open problem and still requires further research.

To recap, our main contributions are three-fold:

- We first analyze the performance gap between popular datasets and practical applications, and therefore construct one of the largest Chinese document-level RE dataset which contains sufficient and diverse hard cases to improve the evaluation for complex RE tasks.

- We propose a case-oriented construction framework to build RE dataset toward special cases. Meanwhile, we design a novel three-stage annotation method applicable for crowdsourcing of complex RE.

- We systematically evaluate the current mainstream RE models on HacRED and justify its effectiveness in depth.

2 Related Work

2.1 Datasets for Relation Extraction

A series of datasets have been built for RE as of late, which have extraordinarily advanced the improvement of RE systems. RE datasets such as SemEval-2010 Task 8 (Hendrickx et al., 2009) and ACE05 are constructed through human annotation with relatively limited relation types and size. A large-scale dataset TACRED (Zhang et al., 2017) is obtained via crowdsourcing to satisfy the training of data-hungry models.

As RE applications differ much in various scenarios, constructing datasets aimed at specific targets is a popular trend in RE. DocRED (Yao et al., 2019) is constructed to accelerate the research on document-level RE. To meet the challenges of few-shot RE, FewRel (Han et al., 2018) as well as FewRel 2.0 (Gao et al., 2019) have been presented. RELX (Koksal and Ozgur, 2020) is a benchmark for cross-lingual RE. Jia et al. (2020) propose the task of interpersonal RE in dyadic dialogues and further construct a corresponding dataset called DDRel.

Compared with previous RE datasets, HacRED is derived from the analysis of the performance gap between popular datasets and practical applications. It targets towards promoting the RE models to extract information from the complex contexts.

2.2 Models for Relation Extraction

Recently, many exciting works have been proposed to solve the RE tasks. **(1)Joint Model:** NovelTagging (Zheng et al., 2017) first formulates the task as a sequence labeling problem and presents a novel tagging schema to jointly extract entities and relations. CopyRE (Zeng et al., 2018) extracts triples based on a sequence-to-sequence structure and integrates the copy mechanism for entity generation. GraphRel (Fu et al., 2019) uses graph convolutional network (GCN) to capture features of words and text. CasRel (Wei et al., 2020) is different from the past and is able to extract more triples by learning relation-specific entity taggers. **(2)Pipeline Model:** PURE

(Zhong and Chen, 2020) is a simple pipelined approach which learns an entity model and a relation model independently. DGCNN-BERT is a powerful pipeline method that first identifies multiple relations and then labels the head and tail entities given a relation. It achieves 89.3 F1 scores and has won the champion in the Competition of DuIE held by Baidu Inc. **(3)Document-level Relation Classification Models:** LSR (Nan et al., 2020) is a model that empowers the relational reasoning across sentences by automatically inducing the latent document-level graph. GAIN (Zeng et al., 2020) introduces a path reasoning mechanism based on a heterogeneous mention-level graph and an entity-level graph. ATLOP (Zhou et al., 2020) proposes two techniques, adaptive thresholding and localized context pooling. SSAN (Xu et al., 2021) designs several transformations to incorporate mention structural dependencies for document-level relation classification (DocRC).

Text 1: "..." said *Joseph Bastianich*, who owns Del Posto with his mother, *Lidia Bastianich*, and the chef, *Mario Batali*. **Annotation**: `NA` **Prediction**: `children_of` **Indicators**: Distractor, Homogeneous Entities **Interpretation**: Three entity mentions with the same type of `PERSON` are mentioned in the text and the word *mother* may lead to wrong prediction `children`.
Text 2: ... Lieberman, who was defeated by the political upstart Ned *Lamont* in *Connecticut*'s Democratic primary earlier this month. **Annotation**: `place_lived` **Prediction**: `place_of_birth` **Indicators**: Similar Relations **Interpretation**: The relation `place_lived` and `place_of_birth` are similar in semantics.
Text 3: One of the most brutal tyrants of recent history, Saddam *Hussein* unleashed devastating regional wars and reduced oil-rich *Iraq* to a claustrophobic police state. **Annotation**: `nationality` **Prediction**: `place_of_death` **Indicators**: Reasoning **Interpretation**: Reasoning is required to get the relation `nationality` based on the context that *Hussein* is the *tyrants* of *Iraq*.

Table 1: Examples of hard cases in NYT. The *head* and *tail* mentions are colored accordingly.

3 Easy Cases vs. Hard Cases

To analyze where models struggle in practical instances and distinguish the hard cases, we conduct a manual exploratory analysis on the error-prone instances of SOTA models (CGCN, CasRel, DGCNN-BERT) on NYT, DuIE and industry data. Then we formulate the potential causes of the errors with nine indicators illustrated as follows:

Text Length. We notice that models tend to fail on instances with longer text. The experiments of Alt et al. (2020) also reflect that RE models get a relatively higher error rate with the length of sentence greater than 30 in TACRED.

Argument Distance. We observe that the performance of the models declines when the arguments (i.e., head and tail entity mentions) are far away, especially in inter-sentence RE.

Distractors. Extracting triples in contexts with linguistic distractors is tough for current models. For example, *drop out* will contribute to wrong relation `graduate_from` between entity mentions with `PERSON` and `SCHOOL` type.

Reasoning. Reasoning is needed to extract the relation mentioned implicitly in the text. Recent work suggests that future researchers consider incorporating common sense knowledge or improved causal modules in RE tasks (Han et al., 2018).

Homogeneous Entities. The context contains multiple homogeneous entity mentions with identical types. We observe the high error rate in relations like `children` and `parents` when the text mentions different entities with type `PERSON`.

Similar Relations. Models struggle to identify the correct relation among those semantically similar ones concurrently mentioned in context. A sharp decrease is also found in few-shot RE when selecting N similar relations on N-way K-shot settings (Han et al., 2020).

Long-tail Relations. Only a handful instances are available for long-tail relations in common datasets. Current data-hungry models struggle to learn the semantic patterns on these relations.

Multiple Triples. Models always get a poor performance on the instances with numerous triples.

Overlapping Triples. Different triples involve the identical entity mentions. Many existing models can not well handle the *EntityPairOverlap* and *SingleEntityOverlap* (Zeng et al., 2018) instances.

Table 1 provides various examples from NYT and corresponding hard case indicators. In Table 2, the proportion growing on the error instances reflects the gap between existing datasets and practical data, which also proves the effectiveness of these indicators.

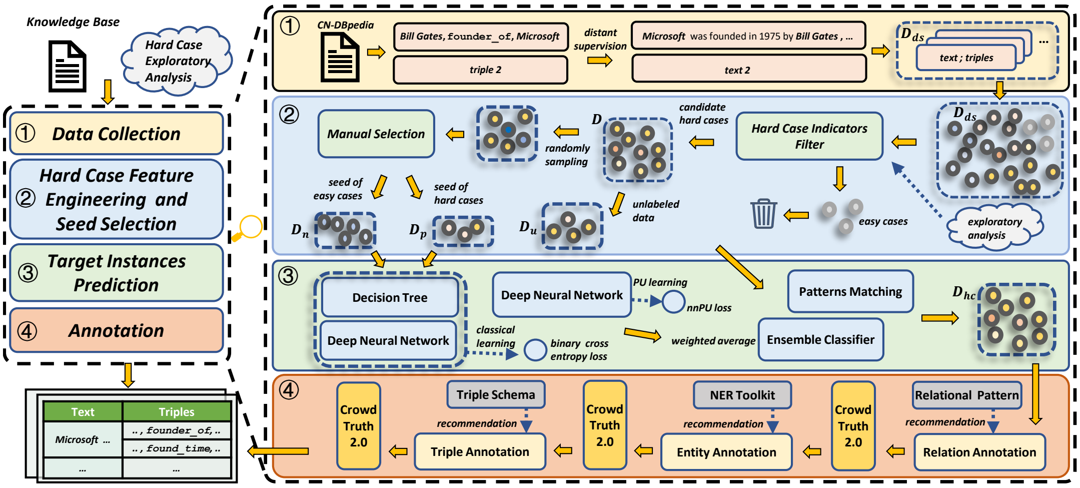

Figure 2: The case-oriented construction framework of building HacRED which consists of four stages. The right part correspondingly describes each stage. Through the construction, the texts and triples are established.

Indicator	WebNLG		DuIE	
	original	error	original	error
Text Length	18	39	3	32
Argument Distance	12	30	5	17
Distractors	1	5	4	13
Reasoning	-	3	1	9
Homogeneous Ent.	2	34	19	21
Similar Rel.	9	54	27	17
Long-tail Rel.	1	5	-	2
Multiple Triples	17	59	8	93
Overlapping Triples	25	64	16	33

Table 2: The proportion of indicators in randomly selected samples of original test set and error-prone instances. Note that one case may fit multiple indicators.

4 HacRED Dataset Construction

The overall architecture of the proposed case-oriented construction framework is illustrated in Figure 2. Different from previous works (Zhang et al., 2017, Zaporojets et al., 2020) which start crowdsourcing annotation straight after the data collection stage, we introduce additional stages of hard case feature engineering and target instance prediction. Moreover, we design a novel three-stage annotation method and employ CrowdTruth2.0.

4.1 Data Collection

To avoid data bias to high-frequency entities and relations, we first obtain about 5 million plain texts and 800 thousand triples from CN-DBpedia. The abundant texts and triples contribute to a more reasonable distribution. We use fine-grained named entity recognition (NER) toolkit TexSmart (Zhang et al., 2020) and entity linking (Chen et al., 2018) to align mentioned entities in texts to those in triples. Finally, we construct a distantly supervised dataset D_{ds} with 1.6 million instances, where we select challenging instances in the following steps.

4.2 Hard Case Feature Engineering and Seed Selection

To build a dataset toward practical hard cases, we systematically formulate the nine indicators of hard cases (refer to Section 3) and introduce measurements to quantify them. For example, we calculate the *Argument Distance* as the number of tokens between the head and tail entity mentions in the text. More details of feature engineering are described in Appendix A. After hard case oriented feature engineering, we discard the instances in D_{ds} without any indicator of hard cases. The remaining part forms a hard case candidate dataset D with about 108 thousand instances.

We randomly sample 3,500 instances from D and ask experts to select the hard cases given the context and features. Specifically, if an instance with multiple hard case indicators or with only one indicator but selected by all three experts based on their expertise, it is regarded as a hard case. To further evaluate the quality of selected hard cases, we utilize DGCNN-BERT to test the selected and unselected data. If the F1 score drops δ=10% on the hard cases, we reserve the data to constitute the high quality seeds of hard case D_p. The remaining data is easy case D_n. In total, we obtain 1,431 seeds of hard cases.

4.3 Classifier Training and Hard Case Prediction

It is impossible to manually select all instances to construct a large-scale dataset. So we utilize a classifier to recall more hard cases similar to the seed samples selected by experts. The classifiers consist of three categories: (1) Decision tree (Quinlan, 1986); (2) Deep classifiers by positive negative (PN) learning (Rakhlin, 2016); (3) Deep classifiers by positive unlabeled (PU) learning (Kiryo et al., 2017; du Plessis et al., 2015). First of all, we adopt the decision tree to make the classifier aware of the indicators explicitly. Then, we form the representation vector as recommended in Baldini Soares et al. (2019) and utilize classical PN learning on D_p and D_n to train the basic classifiers. Since the easy cases are extremely diverse and D_n can not represent the entire distribution of easy cases, we leverage the massive unlabeled data in D_{ds} by introducing PU learning to improve the generalization of hard cases classification. Besides, we train deep models based on different neural network structures, including CNN (LeCun et al., 1998) and BiLSTM (Hochreiter and Schmidhuber, 1997), to capture the context information. More training details can be found in Appendix B.

We ensemble multiple classifiers by weighted average and distinguish hard cases with high confidence in the original massive unlabeled dataset. Besides, we directly select instances by implicit semantic patterns to explore more hard cases fitting the indicator of *Reasoning* which is not well quantified by the auxiliary features. Finally, we obtain the dataset D_{hc} ready for annotation.

4.4 Crowdsourcing

To make instances in D_{hc} fully and accurately labeled, we develop a novel three-stage RE annotation platform taking the following two aspects into consideration: (1) Heavy workload of annotating all information at once results in growing negative feedback as the task goes on; (2) Aggregated method, such as majority vote (Dumitrache et al., 2018), is insufficient for complicated and open-ended tasks. To relieve the pressure of workers, we divide the whole task into three partitions consisting of *Relation Annotation*, *Entity Annotation*, and *Triple Annotation*. Moreover, we utilize patterns and toolkits to provide high-quality recommendations in each stage for higher recall. To capture the label disagreement more thoroughly among workers, we employ CrowdTruth2.0 (Dumitrache et al., 2018), which models the quality of workers, documents, and annotations.

In short, in the *Relation Annotation*, workers select the missed relations or delete wrong recommended ones. When all relations are annotated, NER toolkit recommends multiple entity mentions with the corresponding type based on schema information. Workers also need to append new entity mentions or delete incorrect ones in the *Entity Annotation*. As for *Triple Annotation*, workers verify the correctness of a candidate triples automatically generated by permutation of entity arguments and relations based on schema. Note that every input data in the three stage is assigned to three different annotators and aggregated by CrowdTruth2.0. Detailed annotation process is in Appendix D.

5 Experiments

In this section, we first compare our HacRED with existing datasets. Then we re-evaluate the SOTA RE models on HacRED and systematically analyze their abilities on different experiment settings. At last, we demonstrate the effectiveness of HacRED via a case study.

5.1 Data Analysis

In this section, we analyze various aspects of common RE datasets and HacRED.

Data Size. As shown in Table 3, HacRED has a greater average number of words, entities, and triples in each text than all of the sentence-level datasets. Thus we regard HacRED as a document-level RE dataset. Compared with the document-level datasets, DocRED aims at common document-level RE but not consider performance gaps and various hard cases in practical scenarios. BC5CDR is specially designed for biomedical domain. By contrast, we are the first to analyze the performance gap between popular datasets and practical applications, and propose HacRED which focuses on different kinds of hard cases in general domain. Besides, HacRED is larger in scale and contains much more various relational facts than BC5CDR and DocRED but with lower duplicated triples ratio.

Data Distribution. We calculate three global statistic metrics about data distribution of common datasets and HacRED. Table 4 show the results. Specifically, 84.29% of the triples in NYT and 91.20% in WebNLG are duplicate, which results in a bias to high-frequency triples of same entity pairs (known as *semantic bias* for models). For example, (*Beijing*, `capital_of`, *China*) occurs frequently in corpus and models still extract this triple from *Beijing is a historic city in China*. Mean-

Dataset	# Text	# Relation	# Triple	# Fact	Avg. Sent.	Avg. Word‡	Avg. Ent.	Avg. Triple
sentence-level dataset								
SemEval10	13,434	10	13,434	10,251	1.0	17.4	2.0	1.0
NYT	66,194	24	104,339	16,387	2.1	37.8	2.2	1.6
WebNLG	6,222	171	14,485	1,275	2.5	24.0	3.15	2.3
TACRED	106,264	41	21,773	5,976	1.0	33.2	2.0	1.0
document-level dataset								
BC5CDR	1,500	1	3,116	2,434	7.4	188.0	19.5	2.1
DocRED	5,053	96	63,427	56,354	8.0	198.3	26.2	12.5
HacRED	**9,231**	26	**67,047**	**65,225**	5.0	126.6	10.8	7.4

Table 3: Statistics of common RE datasets and HacRED. Note that the *Avg.Word* is computed at word-level vocabulary, which means “中国” (China), two characters in Chinese, is regarded as one word. The average length of documents at character-level is 204.2 in HacRED.

Dataset	Duplicated Triples	Biased Relations	Top 20% Relation Triples
sentence-level dataset			
SemEval10	23.69%	0.00%	44.92%
NYT	84.29%	58.33%	98.93%
WebNLG	91.20%	94.74%	77.57%
TACRED	72.55%	9.52%	91.33%
document-level dataset			
BC5CDR	21.89%	-	-
DocRED	11.15%	12.50%	71.46%
HacRED	**2.72%**	**0.00%**	49.96%

Table 4: Data distributions of common RE datasets and HacRED. The ratio of duplicate triples, biased relations, and top 20% relation triples is calculated as $1-\frac{\#Facts}{\#Triples}$, $\frac{\#Biased\ Rel}{\#Rel}$, $\frac{\#Triples\ of\ top20\%\ Rel}{\#Triples}$, respectively. If the highest-frequency mention is involved in more than 10% triples of the given relation, we regard it as a biased relation.

Indicators		Ratio
Text Length & Argument Distance		25.40%
Distractors & Reasoning		21.20%
Homo. Entities & Similar Relations		9.67%
Long-tail Relations		13.66%
Multiple Triples	1-3	38.87%
	4-9	36.67%
	10-15	14.27%
	16+	10.20%
Overlapping Triples		13.20%

Table 6: Statistics about the proportion of instances fitting different hard case indicators on HacRED.

CrowdTruth 2.0		Human (%)	
Avg. UQS ↑	0.9373	Precision	97.29
Avg. AQS ↑	0.9446	Recall	94.64
Avg. WQS ↑	0.9557	F1	95.94

Table 7: Results of different quality metrics on HacRED.

Dataset	Relation Example	Highest-frequency Mention (Ratio)
WebNLG	`county_seat`	*Texas* (72.73%)
NYT	`person_profession`	*Bavetta* (50.00%)
DocRED	`sister_city`	*Chipilo* (35.29%)
HacRED	`dynasty`	*Tang* (4.20%)

Table 5: Example of relations which could lead to *selection bias* in WebNLG, NYT, and DocRED. In HacRED, the ratio of the highest-frequency mention in all relations is only 4.20%.

while, the top 20% relations in NYT nearly cover the entire relation triples. The numbers of top and last 20% relation triples in WebNLG, TACRED and DocRED also vary greatly. As a result, models perform well on popular relations but fail on long-tail ones. The experiments in the Section 5.4 prove this and we regard it as *relation bias*. In addition, 94.74% relations in WebNLG, 58.33% in NYT and 12.5% in DocRED contribute to the *selection bias*. In WebNLG, 72.73% triples with relation `county_seat` involve the mention *Texas*, as illustrated in Table 5. Models could memorize the cooccurrence between high-frequency mentions and the relation while low-frequency mentions are neglected. All these three aspects reveal the unreasonable data distribution of common datasets.

In comparison, we observe a more reasonable data distribution in HacRED from Table 4 and Table 5. HacRED has a low ratio of duplicate triples and contains various relational facts, which addresses *semantic bias*. No biased relation existing in HacRED reduces the risk of *selection bias*. The proportion of top 20% relations promotes the alleviation of *relation bias* on HacRED. The more comparison of overall data distribution can be found in Appendix E.

Data Quality. We evaluate the quality of HacRED through both automatic metrics and human evaluation. Specifically, we first compute the average *unit quality score* (UQS), *annotation quality score* (AQS), and *worker quality score* (WQS) of the whole 9,231 instances. *UQS*, *AQS* and *WQS* are proposed by CrowdTruth2.0 (Appendix F provides more calculation details). The closer these

Model	Precision	Recall	F1
Joint		NER‡	
NovelTagging	46.77	35.07	40.08
CopyRE	75.04	51.38	61.00
GraphRel	**85.14**	**69.69**	**76.64**
CasRel	75.43	62.88	68.59
		End-to-end	
NovelTagging	30.51	2.91	5.31
CopyRE	13.11	9.64	11.12
GraphRel	30.13	35.62	32.65
CasRel	**55.24**	**43.78**	**48.85**
Pipeline		NER‡	
PURE	72.23	63.45	67.56
		End-to-end	
PURE	55.14	66.09	60.12
Doc. Level		**Relation Classification**	
LSR	69.70	67.17	68.41
GAIN	72.04	**80.62**	76.09
ATLOP	**77.89**	76.55	**77.21**
SSAN	60.01	62.03	61.00

Table 8: Model performance on HacRED test set(%). NER results are computed based on the entities involved in the gold triples of each instance.

scores are to 1, the higher quality of the crowdsourcing is. Meanwhile, we randomly sample 400 instances from HacRED and compute the precision, recall, and F1 score with annotations based on the revision of humans. The evaluation scores are reported in Table 7. From this table, our HacRED achieves a considerable annotation quality. As a comparison, NYT contains about 31% noise instances (Riedel et al., 2010) and TACRED has poor annotation quality (Alt et al., 2020).

Hard Case Types. We group the randomly sampled 400 instances into nine categories as shown in Table 6. The proportions of different kinds of instances reflect that HacRED contains a various range of hard cases, which evaluates models comprehensively for practical applications.

5.2 Model Evaluation

As DGCNN-BERT has been used in the main process of construction, we evaluate other strong RE models including joint RE models, pipeline RE models, and DocRC models on HacRED. First, we limit the relation set within 20 types both in HacRED and DuIE, and then separate a part of instances in DuIE to form the contrastive easy case dataset D_{ec}. We carry out the equivalent substitution of hard cases in HacRED for easy ones in D_{ec} in different proportions. Figure 3 shows the F1 curve of the performances w.r.t. the proportion of substitution. As the ratio of replacement increases, models generally have a growing trend in performance. The SOTA model CasRel still outperforms other joint models and achieves great F1 on 100% D_{ec}. However, the performance drops on data with more complex instances. We notice that F1 value of easy cases is generally greater than that of hard cases in different substitution ratio settings, which illustrates that RE models indeed struggle when tackling hard cases. Note that by combining HacRED with easy cases in existing datasets, it is easy to simulate diverse practical scenarios.

Model	Precision	Recall	F1
		End-to-end	
CasRel	58.76	45.43	51.24
PURE	56.52	65.15	60.53
Human	**90.21**	**84.59**	**87.31**
		Relation Classification	
ATLOP	78.33	76.70	77.51
Human	**96.21**	**93.03**	**94.59**

Table 9: Human performance (%).

In addition, we split HacRED into train, dev, and test sets with 6231, 1500, 1500 instances respectively. The precision, recall, and F1 score of the three major categories of models are shown in Table 8. The joint and pipeline learning strategies do not contribute to a great F1 on triple extraction. For the NER task, PURE has a separate entity model but results in a 30.61% F1 when all entities in a document are considered, including entities with no positive relation labels. This also reflects the challenge to obtain complete entity information in practical scenarios. On the other hand, the relation classification performances of DocRC models are far from satisfactory. The results suggest that existing models have remarkably poor performance on HacRED compared with humans (Table 9), which indicates that RE applicable for practical hard cases still requires further research.

5.3 Human Performance

We randomly select 200 contexts from test set and ask three volunteers to extract relational facts in an end-to-end manner. Schema information like entity type set as well as relation set is provided but no entity mentions. As for relation classification task, three volunteers select the relation, including NA regarded as negative, of the given entity pair. As demonstrated in Table 9, humans fulfill excellent results which indicate the possible ceiling performance on HacRED.

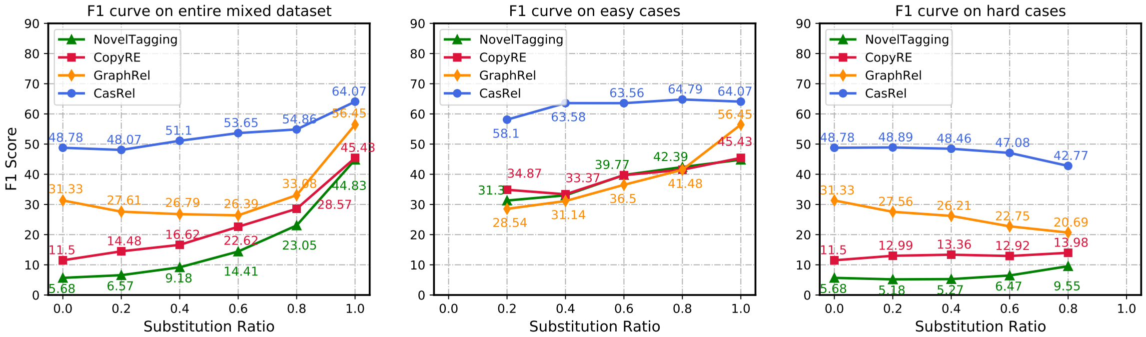

Figure 3: The F1 curve of the model performance on different mix ratios of hard and easy cases.

Model	Text Length Argument Distance	Homo. Ent. Similar Rel.	Long-tail Rel.	Overlapping Triples	Distractor Reasoning	Overall
NovelTagging	4.99	4.33	1.72	3.99	9.23	5.31
CopyRE	5.47	3.90	1.28	6.59	7.30	11.12
GraphRel	30.15	27.82	0.08	34.67	29.81	32.65
CasRel	45.34	45.60	13.54	53.34	44.00	48.85

Table 10: F1 score on HacRED instances with different indicators of hard cases (%).

5.4 Detailed Analysis

In this section, we give insight into the abilities of current mainstream joint models when tackling different kinds of hard cases and propose some research indications as well. As it is hard to obtain complete entity information in practical scenarios, we do not consider DocRC models in this section that entity information is provided as input.

Multiple Triples. Table 11 shows the F1 score of existing models when extracting from texts with different number of triples. The performance of NovelTagging and CopyRE decreases as the number of triples increases, which indicates that the novel tagging schema and multiple decoder mechanism are not able to address the challenge of *Multiple Triples*. Since GraphRel predicts relations for all word pairs and CasRel learns separate entity tagger for different relations, these two models alleviate this problem. An interesting point is that the performance of GraphRel and CasRel rises as the number of triples increases when the triples number is less than 16, indicating that these two models work well in texts with number of triples nearing the average. However, all models get F1 score below average when text mentions have more than 16 triples.

Text Length and Argument Distance. To assess the abilities of models in capturing the long-distance context, we provide the evaluation on instances with indicators of *Text Length* and *Argument Distance* in Table 10. The GCN-based models (i.e., GraphRel) outperforms the BiLSTM-based neural models like NovelTagging and CopyRE. The performance improvement on CasRel suggests the powerfulness of BERT encoder in the long-distance context.

Model	Number of triples			
	1-3	4-9	10-15	16+
NovelTagging	17.92	12.18	8.60	3.29
CopyRE	12.69	10.58	8.82	3.38
GraphRel	29.49	35.23	37.04	29.24
CasRel	43.42	51.05	54.90	43.18

Table 11: F1 score on HacRED test set with different number of triples (%).

Homogeneous Entities and Similar Relations. Since the text mentions multiple homogeneous entities and semantically similar relations, models are required to distinguish the fine-grained difference of the context to extract the correct triples. The first two columns in Table 10 have similar results, which indicates that the contexts with homogeneous entities and similar relations are as challenging as the long-distance contexts.

Long-tail Relations. We observe a dramatic decrease on the instances with long-tail relational triples. As long-tail relations are common in real-world scenarios, a more efficient learning method is required to make RE models applicable for practical applications.

Overlapping Triples. CasRel achieves a better performance on extracting overlapping triples. This proves the effectiveness of cascade binary tagging strategy by first identifying the head mention and then extract the corresponding tail mention given a relation. Specifically, the F1 scores of

Case in HacRED

... *Wu* graduated from *Manchester College* ... and went to *University of Chicago* to study for a doctorate ... President *Lu* invited him to teach western literature at *Yanjing University*. *Wu* resolutely came back to homeland and *became a professor before finishing his doctoral dissertation*.

Annotations

Wu, `graduate_from`, *Manchester College*
Lu, `affiliation_of`, *Yanjing University*
Wu, `affiliation_of`, *Yanjing University*
...

Predictions

Wu, `graduate_from`, *University of Chicago* ✗
Lu, `graduate_from`, *Yanjing University* ✗
...

Hard Case Indicators

Homogeneous Entities, Similar Relations, Distractor, Reasoning

Figure 4: An example of hard cases in HacRED with multiple indicators.

overlapping head and tail mentions are 66.38% and 47.44% respectively. Similarly, results of the two above metrics in CopyRE are 13.31% and 3.57%. The relative higher performance on overlapping head mentions than tail mentions also suggests that the order of extracting arguments could have effect on the results.

Distractor and Reasoning. We manually select instances with *Distractor* and *Reasoning* indicators in HacRED because they cooccur frequently in corpus. As illustrated in Table 10, we observe a drop of the F1. This suggests that models are vulnerable to this kind of instances. However, there are lots of texts with distractions or implicit expression, which needs reasoning, and even common sense. The model design should take the reasoning mechanism into consideration in the future work.

5.5 Case Study

As shown in Figure 4, the text mentions multiple organization entities and similar relations including `graduate_from` and `affiliation_of`. The incorrect triple (*Lu*, `graduate_from`, *Yanjing University*) extracted by CasRel represents that models struggle to capture fine-grained semantic information. The distractive phrases *study for a doctorate* could result in the incorrect extraction (*Wu*, `graduate_from`, *University of Chicago*), which can be rectified by comprehending the context of *before finishing his doctoral dissertation*. Reasoning is needed to extract the triple (*Wu*, `affiliation_of`, *Yanjing University*) since he worked as a professor in the organization.

6 Conclusion

In order to effectively evaluate the RE models and accelerate the research of practical RE, we first analyze the performance gap between popular datasets and practical applications. Therefore, we construct a large-scale and high-quality HacRED with reasonable data distribution and sufficient hard cases. To focus on the practical challenging cases, we propose a case-oriented construction framework. We also design a novel annotation method to guarantee the quality of HacRED. Finally, we conduct extensive experiments and analyze the abilities of SOTA models from various aspects, which provides a deeper understanding of RE models and inspiration for further improvement.

Acknowledgements

We would like to thank the anonymous reviewers for their thoughtful and constructive comments. This research is supported by National Key Research and Development Project (No. 2020AAA0109302), Shanghai Science and Technology Innovation Action Plan (No.19511120400), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103), and Language and Speech Innovation Lab of Huawei Cloud.

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569, Online. Association for Computational Linguistics.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, J. Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Lihan Chen, Jiaqing Liang, Chenhao Xie, and Y. Xiao. 2018. Short text entity linking with fine-grained topics. *Proceedings of the 27th ACM International*

Conference on Information and Knowledge Management.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China. Association for Computational Linguistics.

Anca Dumitrache, Oana Inel, Lora Aroyo, Benjamin Timmermans, and Chris Welty. 2018. Crowdtruth 2.0: quality metrics for crowdsourcing with disagreement. *arXiv preprint arXiv:1808.06080*.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy. Association for Computational Linguistics.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6250–6255, Hong Kong, China. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Jie Zhou, and Maosong Sun. 2020. More data, more relations, more context and more openness: A review and outlook for relation extraction. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 745–758, Suzhou, China. Association for Computational Linguistics.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Iris Hendrickx, S. Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, M. Pennacchiotti, Lorenza Romano, and S. Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *SemEval@ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Qi Jia, Hongru Huang, and K. Q. Zhu. 2020. Ddrel: A new dataset for interpersonal relation classification in dyadic dialogues. *ArXiv*, abs/2012.02553.

Ryuichi Kiryo, Gang Niu, Marthinus Christoffel du Plessis, and Masashi Sugiyama. 2017. Positive-unlabeled learning with non-negative risk estimator. In *NIPS*.

Abdullatif Koksal and A. Ozgur. 2020. The relx dataset and matching the multilingual blanks for cross-lingual relation classification.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

J. Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, A. P. Davis, C. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database: The Journal of Biological Databases and Curation*, 2016.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1546–1557, Online. Association for Computational Linguistics.

Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. 2015. Convex formulation for learning from positive and unlabeled data. In *ICML*.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

A Rakhlin. 2016. Convolutional neural networks for sentence classification. *GitHub*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML/PKDD*.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. *ArXiv*, abs/1811.03925.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *ACL*.

Benfeng Xu, Qiangwen Wang, Yajuan Lyu, Yong Zhu, and Zhendong Mao. 2021. Entity structure within and throughout: Modeling mention dependencies for document-level relation extraction. *ArXiv*, abs/2102.10249.

Bo Xu, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui, and Y. Xiao. 2017. Cn-dbpedia: A never-ending chinese knowledge extraction system. In *IEA/AIE*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Klim Zaporojets, J. Deleu, Chris Develder, and Thomas Demeester. 2020. Dwie: an entity-centric dataset for multi-task document-level information extraction. *ArXiv*, abs/2009.12626.

Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1630–1640, Online. Association for Computational Linguistics.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia. Association for Computational Linguistics.

Haisong Zhang, Lemao Liu, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, Xiao Feng, Tao Chen, Tao Yang, Dong Yu, Feng Zhang, Zhanhui Kang, and Shuming Shi. 2020. Texsmart: A text understanding system for fine-grained ner and enhanced semantic analysis. *arXiv preprint arXiv:2012.15639*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. *ArXiv*, abs/1706.05075.

Zexuan Zhong and Danqi Chen. 2020. A frustratingly easy approach for joint entity and relation extraction. *ArXiv*, abs/2010.12812.

Wenxuan Zhou, Kevin Huang, Tengyu Ma, and J. Huang. 2020. Document-level relation extraction with adaptive thresholding and localized context pooling. *ArXiv*, abs/2010.11304.

A More Examples of Hard Cases in NYT

In Table 12, we provide additional error-prone examples in NYT that fit other indicators of practical hard cases including *Text Length*, *Argument Distance*, *Multiple Triples*, and *Overlapping Triples*. We have illustrated the instances with other indicators in Section 3.

Text 1: *Sixten Ehrling*, ..., and directed the conducting programs at the *Juilliard School* and ...
Annotation: `affiliation of`
Prediction: `major shareholder of`
Indicators: Text Length, Argument Distance
Interpretation: The text contains many words and the distance between head and tail mention is much far. There is no indicating phrases such as *work in* directly revealing the relation `affiliation of`.

Text 2: Though officials in Addis *Ababa* , *Ethiopia*'s capital , ...
Annotation: `administrative divisions, contains, capital`
Prediction: `capital of`
Indicators: Multiple Triples, Overlapping Triples
Interpretation: The text mentions multiple triples and entities such as *Ethiopia* are involved in different triples.

Table 12: Examples of hard cases in NYT. The *head* and *tail* mentions are colored accordingly.

B Details of Feature Engineering

We calculate the *Text Length* and *Argument Distance* as the number of tokens in the text and between the head and tail entity mentions. *Homogeneous Entities* are measured by the NER results of TexSmart and equal to number of entities with same NER tag. The measurements of *Distractors*, *Similar Relations* are based on pre-defined schemas and auxiliary information, part of which is shown in Table 13. *Multiple Triples* and *Overlapping Triples* are computed by the triples from DS. As reasoning can not be implicitly quantified, we suppose the deep neural models to capture the features of context.

C Details of Classifier Training

A decision tree is learned by the auxilliary features calculated in stage 2. For deep models, we concatenate multiple embeddings and auxilliary features to make up the input. We add special tokens to mark the border of each entity and generate the representation vector as recommended in Baldini Soares et al. (2019). We assign a label 1 to

Relation	Type of Arguments	Similar Relations	Explicit Phrase	Distractors
`graduate_from`	PERSON, ORG	`affiliation _of`, `founder_of`	graduate, receive a degree	drop out college, visit
`spouse`	PERSON, PERSON	`parent`, `children`	marry, wife, tie the knot with	ex-wife
`director`	PERSON, FILM / TV SERIES	`cast_member`, `scriptwriter_of`	directed	watch
`anchorperson_of`	PERSON, VARIETY SHOW	`invited_guest_of`	emcee, host	
...	...	...	...	...

Table 13: Examples of pre-defined schemas and simple auxiliary informations to measured the *indicator_distractor* and *similar_rels*. Experts define some implicit expressions such as *receive a degree* reveals the relation *graduated_from* and distractive phrases like *ex-wife* for *spouse*.

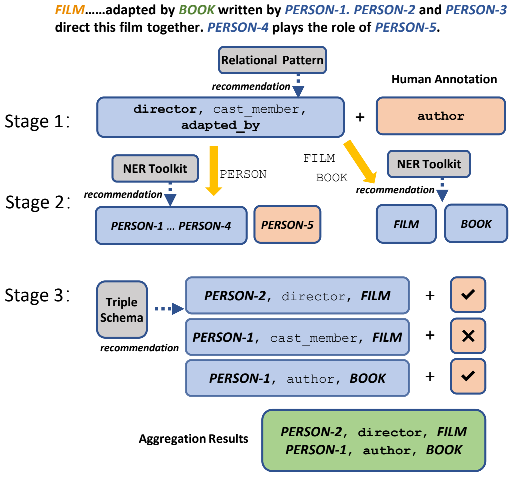

Figure 5: The illustration of three-stage annotation method.

each instance in D_p and -1 in D_n. The deep models output the probability of the instance belonging to hard cases and are optimized with the binary cross entropy loss objective. To start PU learning, we sample from D to form a unlabeled dataset D_u and set the hyperparameter $\pi_p = 0.41$ estimated by the proportion of hard cases selected by experts. We implement nnPU (Kiryo et al., 2017) which is efficient for massive data and deep learning and use J_{nnpu} as the optimized objective,

$$\begin{aligned} J_{nnpu} = \pi_p \cdot & E_{p(x|y=1)}[l(g(x))] + \\ & max\{0, E_{p(x)}[l(-g(x))] - \\ & \quad \pi_p \cdot E_{p(x|y=1)}[l(-g(x))]\} \quad (1) \end{aligned}$$

where $\pi_p = p(y = 1)$, g is decision function, l is surrogate loss function. We choose the double hinge loss $l = max(-z, max(0, \frac{1}{2} - \frac{1}{2}z))$ proposed by (du Plessis et al., 2015).

D Three-stage Annotation Method

We illustrate the three-stage annotation method. Given the context in Figure 5, `director`, `cast_member`, and `adapted_by` is appended to the annotation of Stage 1 by relational pattern. Crowdsourcing workers select the missing relation such as `author`. When all relation mentions are annotated, NER toolkit recommend multiple entity mentions with the corresponding type. Workers need to select the highlighted words that are not covered by entity recommendation in the Stage 2. After stage 2, all mentions in context with specific type are obtained. As the example shown in Figure 5, given the target entity type of `PERSON`, platform recommends the candidates including *PERSON-1* to *PERSON-4*. Workers select highlighted words *PERSON-5* which is missed. In the final stage, we generate the candidate triples automatically by permutation of arguments and relations based on triple schema. Due to the relation `director` connects arguments with entity type `PERSON` and `FILM`, we generate the triple (*PERSON-2*, `director`, *FILM*) and ask annotator to verify the correctness. Note that we employ the powerful quality control method crowdtruth2.0 in every stages to prevent error propagation. As a result, all triples marked as valid are saved.

E Calculation of the UQS, AQS, and WQS Metrics in CrowdTruth2.0

We give the details of the calculation in data quality evaluation. We calculate the three metric unit quality score (UQS), annotation quality score (AQS), and worker quality score (WQS) by CrowdTruth2.0 (Dumitrache et al., 2018) on the whole 9,231 instances in HacRED proposed as follows, where W_1, W_2 is the weight of the iteration method and is initialized as one, u is the unit for annotation,a is one annotation given a unit, i, j denotes the different workers. We straightforward report the average of these metrics in Section 5.1.

$$UQS(u) = \frac{\sum_{i,j} W_1(i,j,u)WQS(i)WQS(j)}{\sum_{i,j} WQS(i)WQS(j)} \tag{2}$$

$$AQS(a) = \frac{\sum_{i,j} WQS(i)WQS(j)P_a(i|j)}{\sum_{i,j} WQS(i)WQS(j)} \tag{3}$$

$$\begin{aligned} WQS(i) &= WUA(i)WWA(i) \\ WUA(i) &= \frac{\sum_u W_2(u,i)UQS(u)}{\sum_u UQS(u)} \\ WWA(i) &= \frac{\sum_{j,u} W_1(i,j,u)WQS(j)UQS(u)}{\sum_{j,u} WQS(j)UQS(u)} \end{aligned} \tag{4}$$

Do Multilingual Neural Machine Translation Models Contain Language Pair Specific Attention Heads?

Zae Myung Kim[1,2], Laurent Besacier[1], Vassilina Nikoulina[1], Didier Schwab[2]
[1]NAVER LABS Europe
[2]Univ. Grenoble Alpes, CNRS, LIG
`{zae-myung.kim,laurent.besacier,vassilina.nikoulina}@naverlabs.com`
`didier.schwab@univ-grenoble-alpes.fr`

Abstract

Recent studies on the analysis of the multilingual representations focus on identifying whether there is an emergence of language-independent representations, or whether a multilingual model partitions its weights among different languages. While most of such work has been conducted in a "black-box" manner, this paper aims to analyze individual components of a multilingual neural translation (NMT) model. In particular, we look at the encoder self-attention and encoder-decoder attention heads (in a many-to-one NMT model) that are more specific to the translation of a certain language pair than others by (1) employing metrics that quantify some aspects of the attention weights such as "variance" or "confidence", and (2) systematically ranking the importance of attention heads with respect to translation quality. Experimental results show that surprisingly, the set of most important attention heads are very similar across the language pairs and that it is possible to remove nearly one-third of the less important heads without hurting the translation quality greatly.

1 Introduction

Recent work on analyzing the internals of Transformer-based models (Vaswani et al., 2017) sheds some light on how different components within the models affect the final performance (Bogoychev, 2020; Behnke and Heafield, 2020), and are closely related to playing linguistically interpretable roles (Voita et al., 2019; Jo and Myaeng, 2020). Moreover, studies on the analysis of multilingual representations (Conneau et al., 2020b; Dufter and Schütze, 2020; Wang et al., 2020b) focus on identifying whether there is an emergence of language-independent representations in multilingual models, or whether multilingual models partition their weights among different languages.

In this paper, we investigate if similar analysis can be made for pretrained multilingual neural machine translation (NMT) models regarding language pair specificity. More precisely, we analyze multi-head attention in a many-to-one (Transformer-based) NMT model and try to find, through an extensive ablation method on selection of the attention heads, whether some heads are more specific to the translation of a certain language pair than others.

Our contributions are the following: (1) we examine the effectiveness of different attention-based metrics on pruning encoder self-attention and cross attention heads; (2) we find that while it is possible to discover rare heads that are specific to a language pair by using a proposed head selection method, most important heads are language-independent; (3) we also show that around 30% of heads can be removed with very little loss of performance.

2 Related Work

Recent studies analyzed the roles of attention heads in the Transformer models either in language modeling (LM) (Michel et al., 2019; Clark et al., 2019; Jo and Myaeng, 2020) or NMT (Voita et al., 2019; Behnke and Heafield, 2020; Michel et al., 2019). It has been shown that a set of attention heads might be redundant at inference and can be pruned with almost no loss in performance. In addition, some studies (Voita et al., 2019; Clark et al., 2019) suggested a linguistic interpretation of self-attention heads. However, most of these analyses were carried out for a single language (in case of LM) or a single language pair (in case of NMT).

In the meantime, efficiency in the cross-lingual transfer of recently released pretrained multilingual language models (Devlin et al., 2019; Conneau et al., 2020a) has boosted an active line of research trying to analyze their representations to

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2832–2841
August 1–6, 2021.

understand what favors the emergence of an *interlingua*. For instance, Pires et al. (2019); Dufter and Schütze (2020); Karthikeyan et al. (2020) tried to decouple the effect of shared "anchors"[1] from the rest of the model. Very recently, Muller et al. (2021) performed a more fine-grained analysis, examining representations at each layer of the model.

Despite the success of massively multilingual NMT models (Johnson et al., 2017; Bapna and Firat, 2019; Aharoni et al., 2019; Zhang et al., 2020), less effort has been made in analyzing multilingual NMT representations. Kudugunta et al. (2019) clustered the representations of different languages learned by multilingual NMT models showing that common representations emerge in the encoder. Mareček et al. (2020) found that while RNN models (Attention Bridge architecture) (Cífka and Bojar, 2018; Lu et al., 2018) learn to capture certain linguistic properties with an increasing number of target languages, Transformer models are largely unaffected. Recent work of Zhang et al. (2021) introduced a conditional routing layer in a form of gate selection between language-specific and language-independent projection, providing some insights on which components allow for the emergence of *interlingua*.

Our work builds on the findings from the attention heads analyses (Voita et al., 2019; Michel et al., 2019) but attempts to extend them to multilingual NMT, investigating whether it is possible to discover attention heads that are language pair specific. Also, we experimented with a set of attention-based metrics and analyzed how effective they are in pruning under different language pairs and types of attention.

3 Methodology

As our goal was to identify "important" attention heads for different language pairs, we first needed to define a metric or a procedure that can capture the notion of "importance" of an attention head, and selected heads based on this importance.

In Section 3.1, we present a set of metrics that quantify certain aspects of attention weights, which to some extent, can be considered as the importance. Section 3.2 illustrates a more direct approach where the importance of a head is defined as the extent of decrease in BLEU scores (Papineni et al., 2002) resulted in pruning the head.

3.1 Metrics Based on Attention Weights

We experimented with three types of metrics that are defined for each attention head, $\texttt{head}_{l\in L,h\in H}$, where l and h are the indices of layer and multi-head, respectively. In what follows we define how the metrics were computed for one sentence. Each metric was computed and averaged over a set of development sentences, then normalized to zero mean and unit standard deviation for ease of comparison. We note that $|I|$ and $|J|$ were the number of source tokens and/or target tokens, depending on whether we looked at the self-attention of encoder or the encoder-decoder cross attentions.

Confidence Voita et al. (2019) defined the notion of confidence of a head to be the mean of its maximum attention weights, and showed that only a small set of heads are confident and responsible for most of the model's performance.

$$\mathrm{conf}(\mathrm{head}) := \frac{1}{|I|}\sum_{i\in I}\max_{j\in J}\alpha_{i,j}$$

Variance Inspired by Vig and Belinkov (2019), we computed the expected position of attention for token i as $\mu_i := \mathbb{E}[j|i] = \sum_{j\in J} j\cdot\alpha_{i,j}$, and measured how much each individual position was away from it:[2]

$$\mathrm{var}(\mathrm{head}) := -\sum_{i\in I}\sum_{j\in J}\alpha_{i,j}\,(\mu_i - j)^2$$

Coverage Tu et al. (2016) defined the notion of coverage for encoder-decoder attentions which computes the amount of attention a source token has received. We extended the idea to the self-attentions in encoder as well.

$$\mathrm{cov}(\mathrm{head}) := \sum_{j\in J}\left(\sum_{i\in I}\alpha_{i,j}\right)^2$$

More details on the metrics are provided in Appendix C.

3.2 Sequential Backward Selection of Heads

Intuitively, a head can be considered as important if its removal results in a drastic decrease in the BLEU scores. As different combinations of heads can affect the performance differently, we followed the sequential backward selection (SBS) algorithm (Aha and Bankert, 1996), which is a top-down algorithm starting from a feature set of all features (in

---

[1] either shared vocabulary or shared special tokens such as ⟨SEP⟩, ⟨EOS⟩, etc.

[2] As we wanted the important heads to have lower variance, we multiplied the score with -1.

Algorithm 1: SBS for Head Selection

```
selections ← $\emptyset$;
while $|\text{selections}| < |L| \cdot |H|$ do
    bleuMin ← $\infty$;
    headMin ← $\emptyset$;
    for $\forall\ \text{head}_{l \in L, h \in H} \notin$ selections do
        masks ← selections $\cup$ $\text{head}_{l,h}$;
        trans ← Translate(masks);
        bleuDrop ← Evaluate(trans);
        if bleuDrop < bleuMin then
            bleuMin ← bleuDrop;
            headMin ← $\text{head}_{l,h}$;
        end if
    end for
    selections ← selections $\cup$ headMin;
end while
return selections;
```

our case, a set of all heads) and sequentially removing the most irrelevant features that maximize the evaluation metric (in our case, the BLEU score).

The pseudo-code for the head selection procedure is illustrated in Algorithm 1. The algorithm first selects a head that, when masked, results in the smallest decrease in the BLEU score; and adds it to selections. For subsequent iterations, it proceeds similarly, but the masks now include the heads in selections as well as the candidate head. The procedure terminates when all heads are selected. Note that the time complexity of the algorithm is $\mathcal{O}(|L|^2|H|^2)$, where L and H denote the set of layers and attention heads, respectively. It is a computationally intensive procedure as for each iteration, a test set is translated and evaluated.

4 Experiments and Results

4.1 Preliminary Experiment

We conducted a preliminary experiment using a many-to-one multilingual model trained on a TED talk dataset (Qi et al., 2018), covering top-20 source languages with the most data. We observed that patterns of attention heads (measured with the "confidence" metric) for both encoder self-attention and encoder-decoder attention were very similar among the language pairs.

For the main experiment, we decided to use a larger and stronger multilingual model for the following reasons: (1) the TED dataset is quite small and the model trained on it achieves lower BLEU scores and may not be regularised very well; (2) the network capacity of the TED model could be too limited for the language-pair-specific patterns to emerge (if any). According to a study on BERT's multilinguality (Dufter and Schütze, 2020), the increased network capacity (i.e., overparameterization) is shown to lead to more decoupled representations between languages.

As the multilingual model described in Section 4.2 is trained on much larger datasets, and has a network capacity larger than the initial TED model while covering fewer language pairs, we expect that the language-pair specificity (if any) is more likely to emerge.

4.2 Experimental Settings

For the sake of reproducibility, all experiments were conducted using a strong publicly available many-to-one multilingual NMT model released by Bérard et al. (2020). The model can translate French, German, Italian, Spanish, and Korean sentences into English. It is trained with standard open-accessible datasets, including biomedical corpora where available. The model uses a variant of the Transformer-Big architecture (Vaswani et al., 2017) with a shallower decoder: 16 attention heads, 6 encoder layers, and 3 decoder layers. The model produces SOTA- or near-SOTA-level results for news, IWSLT, and biomedical translation tasks.

As the model is many-to-one, we could set up a controlled experiment where the BLEU scores were directly compared among the language pairs. We employed the development and test sets from the TED talk dataset, and utilized only the multilingual sentence pairs where both source and reference sentences were present for all five language pairs.[3] After the filtering, the development and test sets contained 1771 and 2137 pairs, respectively.

As we were using a many-to-one model, we conducted experiments on both encoder self-attentions and encoder-decoder attentions. In our experiments, we did not re-train or fine-tune the model when masking each head, making procedure lighter than other approaches involving the re-training.

4.3 Results

Heads importance across languages Figure 1 shows the heatmaps for each importance metric (Sect. 3.1) for the self-attention and cross-attention

---

[3] Note that the English reference sentences were the same across the language pairs.

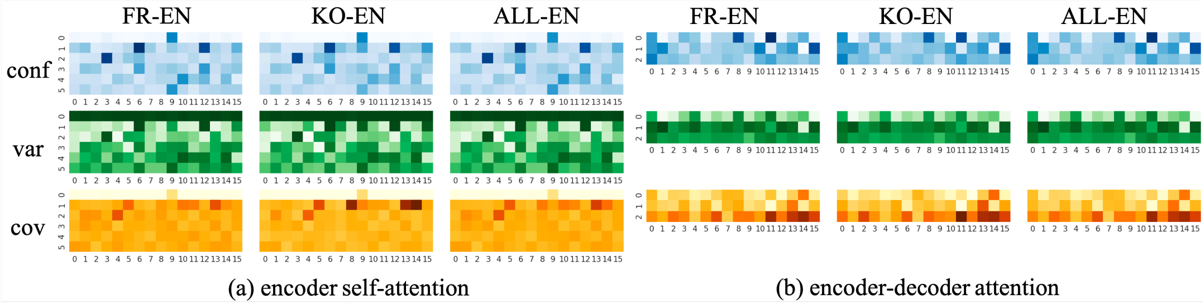

(a) encoder self-attention

(b) encoder-decoder attention

Figure 1: Each heatmap of a language pair shows the corresponding normalized metric scores for every **(a) encoder self-attention** and **(b) encoder-decoder attention** head, broken out by layer (vertical axis) and head (horizontal axis). For each metric, the color scales are identical across language pairs.

heads, respectively. The heatmaps were computed for each language pair separately (FR-EN, KO-EN, etc.) or jointly for all pairs (ALL-EN).[4] The main finding is that even if each metric displayed a different heatmap, the important heads were the same for all language pairs according to these metrics. In other words, the metrics did not highlight the emergence of language pair specific (encoder or cross) attention heads. Comparing among the metrics, *variance* and *confidence* tended to emphasize the same heads (with the exception of the first self-attention heads of each layer which were systematically rated as important by the *variance* metric). On the other hand, *coverage* highlighted different heads compared to the other two metrics.

Impact of head selection on NMT performance In the previous paragraph, we explored several metrics that could help capturing the importance of an attention head. We now analyze if these metrics could be used to prune heads and the corresponding impact on MT performance. We also investigate a more direct (but more costly) approach to measure how heads contribute to MT performance.

Figure 2 shows the evolution of BLEU curves as more and more heads were pruned. Head pruning was based on the importance metrics (removing least important heads first according to the metrics presented in Sect. 3.1) or on the SBS algorithm (Sect. 3.2). Head selection was conducted separately for each language pair,[5] and the curves were drawn from fitting polynomial regressions. First, we observed that, for both encoder self-attentions and cross attentions, it was possible to remove around 30% of the less important heads without much decrease of BLEU. Next, we noted that for cross attention head pruning, *coverage* seemed to be a better alternative than *confidence* and *variance*, while for encoder self-attention pruning *confidence* remained the most efficient. Intuitively, *coverage* metric is complementary to *confidence* in case of cross attention as it measures whether the whole input has been attended to. On the other hand, self-attention heads seemed devoted to specific phenomena (Voita et al., 2019; Clark et al., 2019) and there was no need to attend to the whole sentence for this matter. Finally, we also display the BLEU curves for randomly ranking (`rand-ranking`) the attention heads, confirming that the metrics proposed can be used as a proxy to measure the importance of heads and prune the least important ones. However, the exhaustive (but costly) SBS algorithm logically led to the best results.

Is there really no emergence of language-specific heads? We verified how statistically significant the BLEU differences were between *language-specific* and *language-independent* heads selection processes according to various metrics with Mann–Whitney U tests (Mann and Whitney, 1947).[6] We found no significant difference between language-specific and language-independent head rankings, even if some differences emerged for results obtained by SBS ranking.

Finally, we looked at how the individual head rankings were varied according to the SBS algorithm. Figure 3 illustrates the standard deviation of each head position among the rankings of the five different language pairs. We observed that there were a few heads whose relative importance varied greatly among the language pairs. For example,

---

[4] We only display FR-EN and KO-EN, reader should refer to Appendix B for all language pairs.

[5] The language-independent selection of heads led to a very similar plot as Fig. 2 and is provided in Appendix E

[6] We report p-values for Mann–Whitney U tests in the Appendix D.

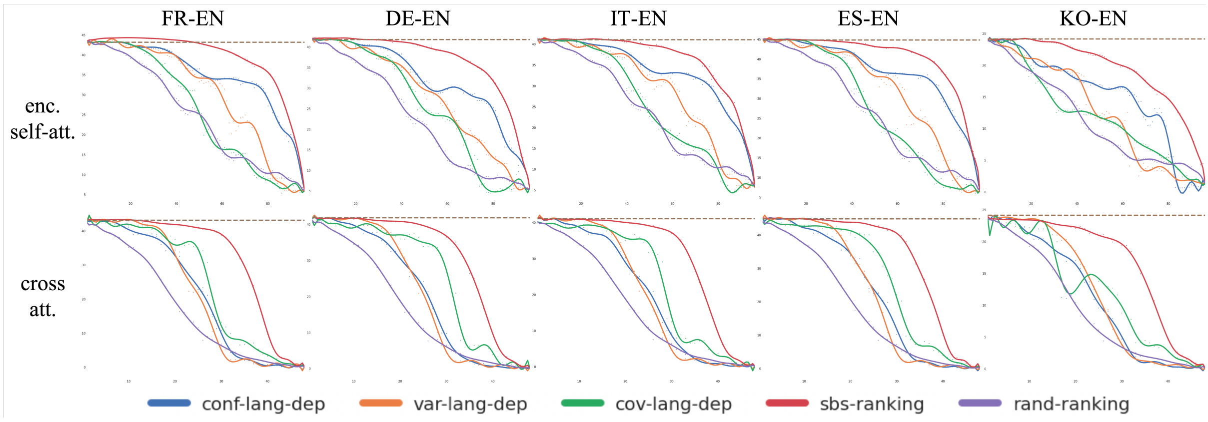

Figure 2: BLEU curves on test set when pruning subsequent **self-attention** and **cross attention** heads based on different importance metrics (or SBS) computed from dev set (language pair dependently). To be seen in color.

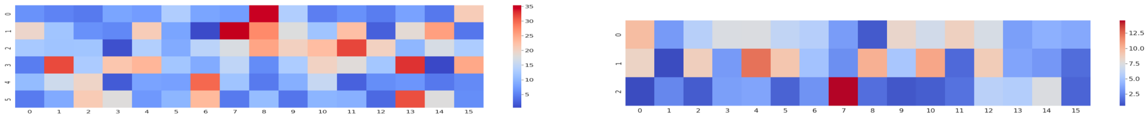

(a) encoder self-attention

(b) encoder-decoder attention

Figure 3: Standard deviation of each **(a) encoder self-attention** and **(b) encoder-decoder attention** head ranking with SBS algorithm. SBS rankings range from 1 to 96 for self attention and from 1 to 48 for cross attention and standard deviation is calculated for each head, using these scores, among the five language pairs.

the $\texttt{head}_{2,7}$ of the encoder-decoder attention was ranked as least important for KO-EN but quite important for the other four language pairs.[7] Similarly, $\texttt{head}_{1,7}$ for encoder self-attention was ranked as not important for ES-EN while very important for KO-EN. This analysis showed that, even though the majority of important heads seemed to be language-independent, certain heads may capture different linguistic phenomena.

5 Conclusion and Future Work

We investigated if there are attention heads that are language pair specific within a many-to-one multilingual NMT model. We examined different metrics for heads selection process and found that *confidence* is a good proxy for self-attention heads "prunability", and *coverage* is a better indicator for cross attention heads "prunability".

We showed that, although it is possible to find the rare heads specific to a language pair via the extensive SBS procedure, the most important heads are language-independent; and it is possible to prune around 30% of the heads with no retraining and almost no loss in BLEU.[8]

As the findings from the SBS procedure indicated that some language pair specific heads do exist, a promising future direction is to perform pruning at different level of granularity (Frankle and Carbin, 2019; Zhao et al., 2020) (as opposed to single scalar values computed by the metrics) in order to identify which part of the model is more language-specific. Such analysis could help us to deploy multilingual models with better efficiency / performance trade-offs.

Acknowledgments

This work was done as part of the Multidisciplinary Institute in Artificial Intelligence MIAI@Grenoble-Alpes (ANR-19-P3IA-0003). Authors would like to thank Jaesong Lee and Hyun Chang Cho from NAVER Corp., Kwang Hee Lee from KAIST for insightful comments on the experiments. We would also like to extend our gratitude to the anonymous reviewers for their valuable feedback.

---

[7] Masking this single head alone, resulted in an increase in BLEU for KO-EN by 0.03, while for others, a decrease in BLEU up to 0.5.

[8] It is possible that fine-tuning the model after pruning the heads may lead to better BLEU scores, and therefore more aggressive pruning (Behnke and Heafield, 2020; Wang et al., 2020a).

References

David W. Aha and Richard L. Bankert. 1996. *A Comparative Evaluation of Sequential Feature Selection Algorithms*. Springer New York, New York, NY.

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Maximiliana Behnke and Kenneth Heafield. 2020. Losing heads in the lottery: Pruning transformer attention in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2664–2674, Online. Association for Computational Linguistics.

Alexandre Bérard, Zae Myung Kim, Vassilina Nikoulina, Eunjeong Lucy Park, and Matthias Gallé. 2020. A multilingual neural machine translation model for biomedical data. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Nikolay Bogoychev. 2020. Not all parameters are born equal: Attention is mostly what you need.

Ondřej Cífka and Ondřej Bojar. 2018. Are BLEU and meaning representation in opposition? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Philipp Dufter and Hinrich Schütze. 2020. Identifying elements essential for BERT's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Jae-young Jo and Sung-Hyon Myaeng. 2020. Roles and utilization of attention heads in transformer-based neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5.

K Karthikeyan, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual bert: An empirical study. In *International Conference on Learning Representations*.

Sneha Kudugunta, Ankur Bapna, Isaac Caswell, and Orhan Firat. 2019. Investigating multilingual NMT representations at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, Brussels, Belgium. Association for Computational Linguistics.

Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*.

David Mareček, Hande Celikkanat, Miikka Silfverberg, Vinit Ravishankar, Jörg Tiedemann, et al. 2020. Are multilingual neural machine translation models better at capturing linguistic features? *The Prague Bulletin of Mathematical Linguistics*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Benjamin Muller, Yanai Elazar, Benoît Sagot, and Djamé Seddah. 2021. First align, then predict: Understanding the cross-lingual ability of multilingual bert. In *EACL 2021*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Yong Wang, Longyue Wang, Victor Li, and Zhaopeng Tu. 2020a. On the sparsity of neural machine translation models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020b. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online. Association for Computational Linguistics.

Biao Zhang, Ankur Bapna, Rico Sennrich, and Orhan Firat. 2021. Share or not? learning to schedule language-specific capacity for multilingual translation. In *International Conference on Learning Representations*.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

A Experimental Details

All the experiments were conducted using PyTorch (Paszke et al., 2019) and Fairseq (Ott et al., 2019) toolkit. The multilingual NMT model used in the experiments can be downloaded online.[9] Please refer to Bérard et al. (2020) for more details on the model.

When running an experiment for each language pair, a single V100 GPU was used. We note that computing the SBS rankings for encoder self-attention was the most computationally intensive part, where almost 96^2 translations of the development set were conducted.

When computing the BLEU curves for the `rand-ranking`, we ran the procedure with a randomly created ranking five times, and averaged the resulting BLEU scores.

B Heatmaps of Metric Scores for All Language Pairs

Figure B illustrates the normalized metrics scores for every attention head. We observed that for each metric, the patterns are consistent across all language pairs.

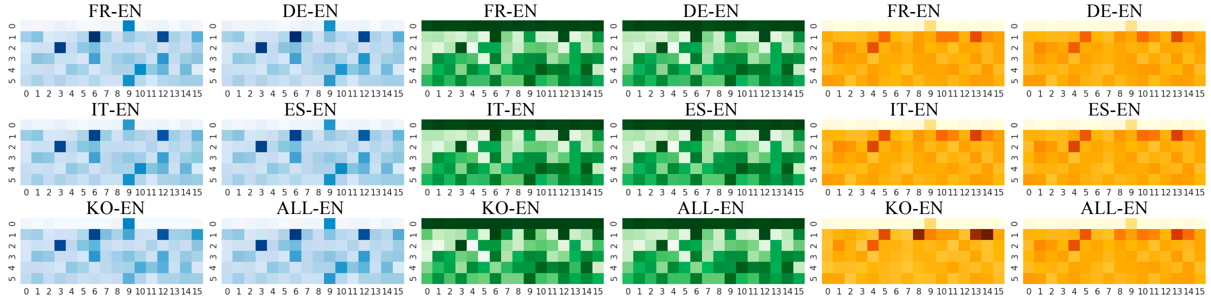

(1) confidence (2) variance (3) coverage

(a) encoder self-attention

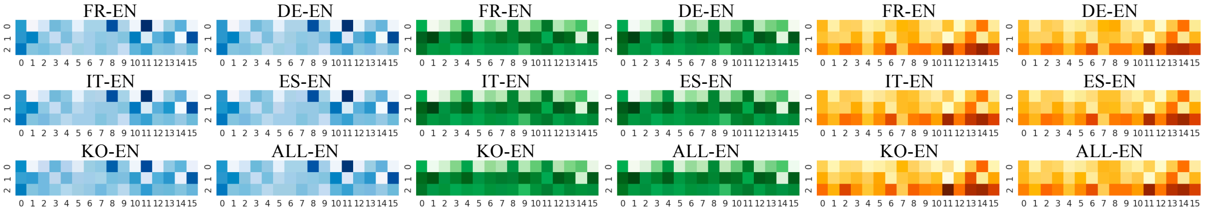

(1) confidence (2) variance (3) coverage

(b) encoder-decoder attention

Figure B: Each heatmap of a language pair shows the corresponding normalized metric scores for every **(a) encoder self-attention head** and **(b) encoder-decoder attention head**, broken out by layer (vertical axis) and head (horizontal axis). For each metric, the color scales are identical across language pairs.

C Remarks on the Importance Metrics

We denote $|L|$ and $|H|$ to be the number of layers and heads, respectively, while $|S|$ and $|T|$ represent the number of source and target tokens. When calculating each metric, we began with the tensor shape, $(|L|, |H|, |S|, |S|)$ or $(|L|, |H|, |T|, |S|)$, depending on whether we were computing for the encoder self-attention or the cross attention. After the computation, the shape of the outcome tensor was: $(|L|, |H|)$.

C.1 Confidence

We noted that the patterns of the confidence scores for each head tended to vary depending on the length of sentences we used to compute the scores. This was due to the fact that the metric was calculated by averaging over the maximum attention, which was inversely proportional to the length of sentences.

---

[9] `https://github.com/naver/covid19-nmt`

C.2 Variance

The variance metric was defined so that heads with a small variance were considered to be important. A small variance was achieved when most of the attention weights were focused on one or a few positions. While this intuition came initially from encoder-decoder attention (interpreting attention as a source-target alignment), it is less clear if it holds for encoder self-attention as well (our results seemed to suggest that it is not the case).

C.3 Coverage

While the notion of coverage was initially proposed for encoder-decoder attention, we extended it to the encoder self-attention. We may consider it as how much a source token has been attended from its neighbouring source tokens. Similar to the variance metric, for the encoder self-attention, the importance of high coverage is less clear where a head may play a specific role as discussed in Voita et al. (2019); Clark et al. (2019). This probably accounts for the reason that the head pruning of the encoder self-attention was not as effective as that of the cross attention.

D P-Values for Mann–Whitney U tests

As the BLEU curves obtained from language-specific pruning and language-independent pruning were very similar, we performed a non-parametric statistical test, namely, Mann-Whitney U test, to compare the outcomes. The test checks whether two samples are likely to derive from the same population (i.e. that the two populations have the same shape).

Table D shows the p-values for the two-sided tests between BLEU curves computed using language-specific and language-independent metrics for encoder self-attention and cross attention.

The high p-values (> 0.05) across all language pairs suggest that the differences in the BLEU scores computed from the two scenarios were statistically insignificant.

	FR-EN	DE-EN	IT-EN	ES-EN	KO-EN
conf	0.938	0.849	0.871	0.878	0.902
var	0.927	0.995	0.959	0.939	0.573
cov	0.570	0.555	0.865	0.927	0.850
sbs	0.137	0.189	0.293	0.878	0.375

(a) encoder self-attention

	FR-EN	DE-EN	IT-EN	ES-EN	KO-EN
conf	0.918	0.988	0.965	0.968	0.881
var	0.991	0.994	0.997	0.985	0.936
cov	0.772	0.907	0.912	0.889	0.621
sbs	0.901	0.631	0.936	0.918	0.404

(b) encoder-decoder attention

Table D: P-values for Mann–Whitney U tests between BLEU scores computed using language-specific and -independent metrics for **(a) encoder self-attention** and **(b) encoder-decoder attention**.

E BLEU Curves (Language-Independent Head Selection) for All Language Pairs

In Figure E, we present the BLEU curves obtained from pruning the encoder self-attention heads and cross attention heads according to the importance metrics (and SBS) computed over *all language pairs* (i.e. language-independent). We observed that the curves were very similar to those presented in Figure 2 of the main paper, where the computation was conducted over the *specific language pairs*.

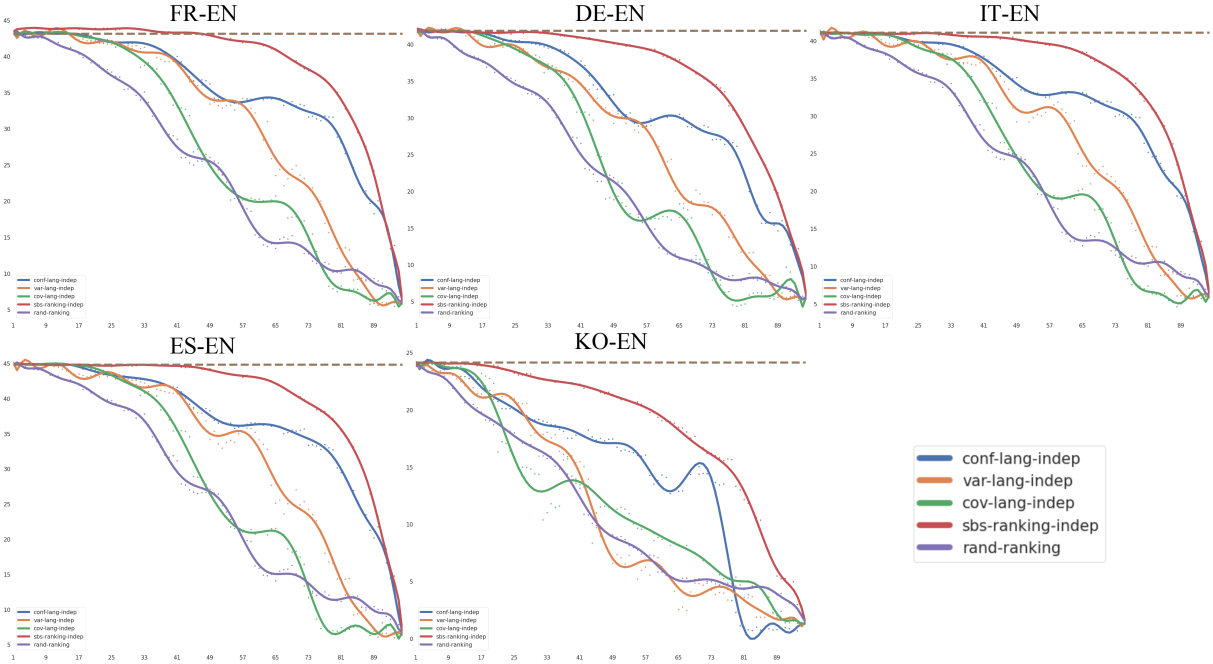

(a) encoder self-attention

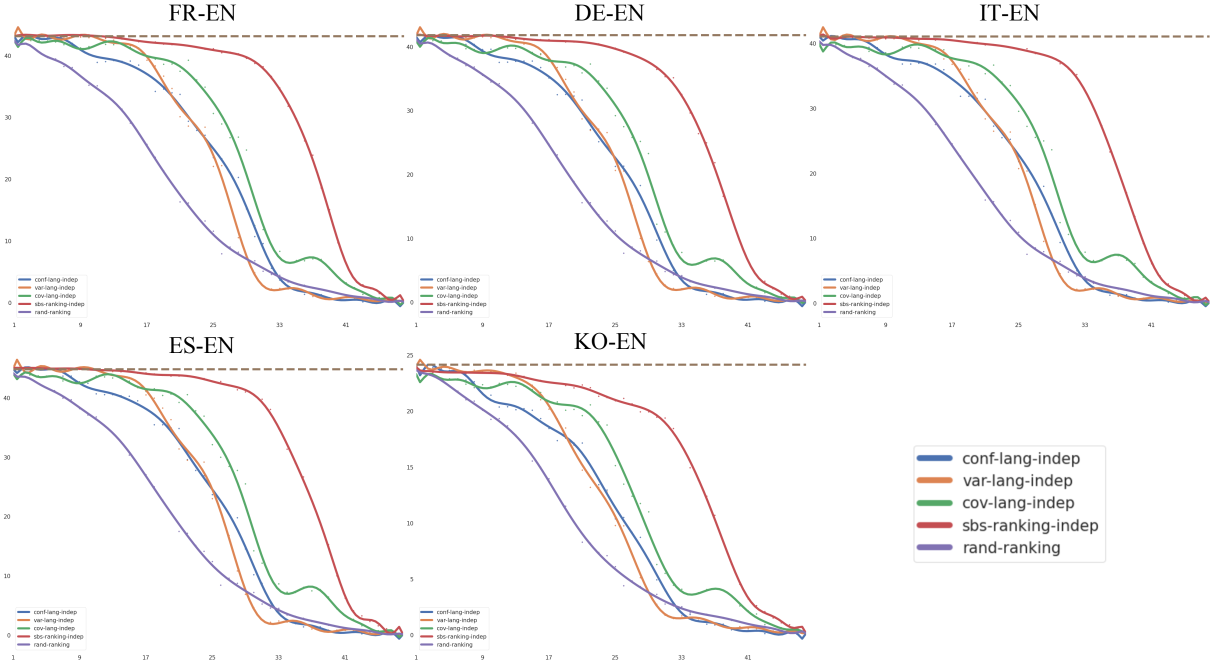

(b) encoder-decoder attention

Figure E: BLEU curves on test set when pruning subsequent **(a) encoder self-attention heads** and **(b) encoder-decoder attention heads** based on different importance metrics (or SBS) computed from the development set (language pair independently).

Learning Sequential and Structural Information for Source Code Summarization

YunSeok Choi, JinYeong Bak, CheolWon Na, Jee-Hyong Lee
College of Computing and Informatics
Sungkyunkwan University
Suwon, South Korea
`{ys.choi, jy.bak, ncw0034, john}@skku.edu`

Abstract

We propose a model that learns both the sequential and the structural features of code for source code summarization. We adopt the abstract syntax tree (AST) and graph convolution to model the structural information and the Transformer to model the sequential information. We convert code snippets into ASTs and apply graph convolution to obtain structurally-encoded node representations. Then, the sequences of the graph-convolutioned AST nodes are processed by the Transformer layers. Since structurally-neighboring nodes will have similar representations in graph-convolutioned trees, the Transformer layers can effectively capture not only the sequential information but also the structural information such as sentences or blocks of source code. We show that our model outperforms the state-of-the-art for source code summarization by experiments and human evaluations.

1 Introduction

Descriptions of source code are very important documents for programmers. Good descriptions help programmers understand the meaning of code quickly and easily.

Source code has sequential information (code tokens) and structural information (dependency and structure). To understand the content of the code and generate a good summary, both pieces of information are essential to understand the summary. However, most previous works used only one kind of information. Iyer et al. (2016); Liang and Zhu (2018); Hu et al. (2018b); Allamanis et al. (2016) simply converted the source code into sequences of tokens and tried to extract the features of the source code from the sequential information of the source code. They rarely considered the structure information about the relationship between tokens. On the other hand, Shido et al. (2019); Harer et al. (2019); LeClair et al. (2020); Scarselli et al. (2008) proposed tree-based models to capture the features of the source code. They used the structural information from parse trees but hardly considered the sequence information of code tokens.

In order to accurately understand and represent the source code, it is necessary to encode the structural information as well as the sequential information. Recently, Ahmad et al. (2020) tried to represent both the sequential and the structural information using the Transformer model with relative encoding. Since relative encoding clipped the maximum distance for attention without considering the parse tree, it is limited to represent the structure of the code.

In this work, we propose a model that learns both the structural and sequential information of source code. We represent code snippets as abstract syntax trees (ASTs), and apply graph convolution (Kipf and Welling, 2017) to the ASTs to obtain the node representation reflecting the tree structure such as parents, children, and siblings. Nodes that are close to each other in a tree, such as parent and child nodes and sibling nodes, will have similar representations. Next, we convert the graph-convolutioned ASTs into sequences by the pre-order traversal and process them with Transformer layers (Vaswani et al., 2017). Since structurally-neighboring nodes have similar representations, such nodes will pay more attention to one another in the Transformer layers. Thus, the Transformer layers can easily capture not only the sequential information but also the structural information such as sentences or blocks of source code.

We also modify ASTs to better represent the structural information of the source code. We add sibling edges to represent neighboring blocks in source code and add a node representing the name of the function for Python. With the modification,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2842–2851
August 1–6, 2021. ©2021 Association for Computational Linguistics

```
public static int returnSumValue(int[] values) {
    int sum_value = 0;
    for (int value : values) {
        sum_value += value;
    }
    return sum_value;
}
```

(a) Source Code

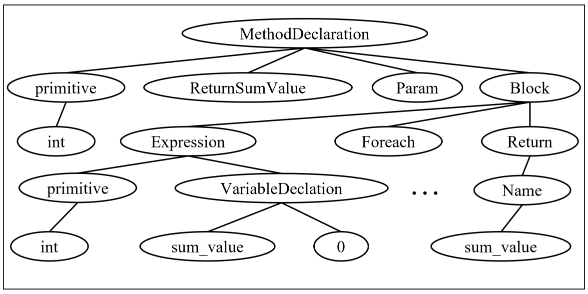

(b) AST

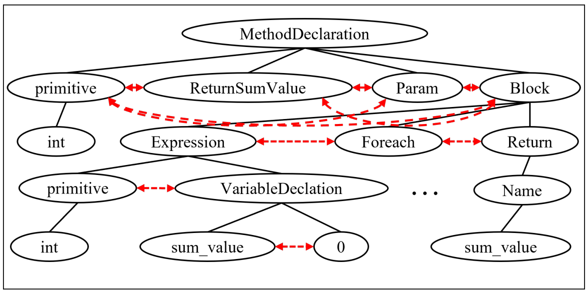

(c) mAST

Figure 1: An example of Java code, Abstract Syntax Tree and modified-Abstract Syntax Tree.

our model can better catch the blocks in the source code.

In the experiment, we show that our model outperforms the state-of-the-art for source code summarization. We use two well-known Java (Hu et al., 2018b) and Python (Wan et al., 2018) datasets collected from Github. We additionally perform human evaluations and analyze the attention maps between code tokens to see how well the model has captured the structural information of code. The result proves that it is very effective to model both structural and sequential information for source code summarization.

We describe the modified AST (mAST) in Section 2 and present the proposed approach in Section 3. In Section 4, we show the superiority of our approach with experimental results and human evaluations. We describe the related work in source code summarization and compare their approaches with our proposed approach in Section 5. Finally, we conclude the paper in Section 6.

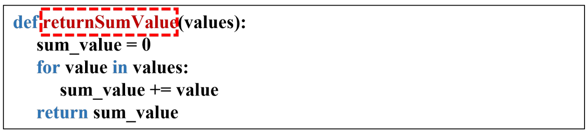

```
def returnSumValue(values):
    sum_value = 0
    for value in values:
        sum_value += value
    return sum_value
```

(a) Source Code

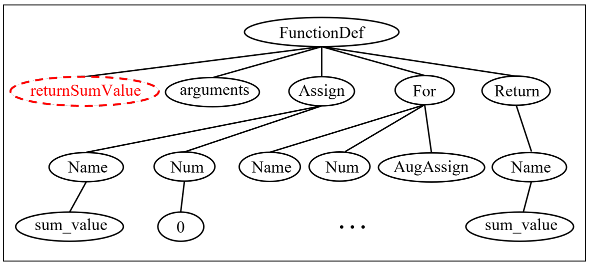

(b) AST

Figure 2: An example of Python code and Abstract Syntax Tree. The Python AST parser we used does not create a node for the function name unlike the Java AST parser. Since the function name is a very important keyword in generating summaries, we add the function name in the red box (a) to *FunctionDef* node in the Python AST.

2 Representing Code as mAST

Figure 1 shows a code snippet and its AST in Java. The abstract syntax tree (AST) is a structure to represent the abstract syntactic structure of code in a programming language. Source code is separated into blocks and can be transformed into a tree structure. The leaf nodes of an AST represent code identifiers and names. The non-leaf nodes represent the grammar or the structure of the language. All non-leaf nodes in an AST have the structural information about which blocks they belong to (parent node) and which block they have (child nodes). So, we can easily catch the structure information of code from ASTs.

In order to more effectively represent structural information, we modify ASTs by adding edges between siblings. Statements or blocks at the same level in a code snippet are represented as sibling nodes. For example, *Expression* node (line 2 in the code), *Foreach* node (line 3) and *Return* node (line 7) are the statements or blocks at the same level, and they are represented as siblings in the AST. However, in ASTs, blocks at the same level (sibling nodes) are not directly connected as shown in Figure 1b. We can indirectly catch such information via parent nodes.

Neighboring blocks are very important for the sequential and structural understanding of source code. To directly represent neighboring blocks, we

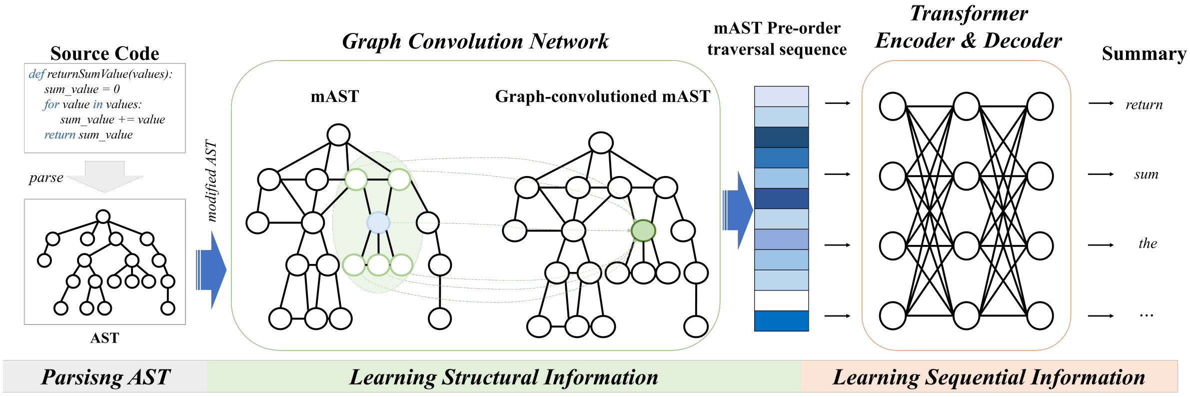

Figure 3: Overview of our proposed model.

add sibling edges to ASTs as shown in Figure 1c (red dotted lines).

In the case of Python, the Python AST parser we used does not create a node for the function name, unlike the Java AST parser. As the function name is a very important keyword in generating summaries, we add the function name to *FunctionDef* node as a child in the Python AST as shown in Figure 2. Then, we modify Python ASTs by adding sibling edges.

3 Proposed Model

We propose a model using graph convolution layers and transformer layers to summarize the source code. To encode both the structural and the sequential information of code, we combine both of the above layers.

Figure 3 shows the overview of our model. A given snippet is represented as a modified AST (mAST). The initial representation of nodes in the mAST is generated by the embedding layer. Since the embedding layer generates the representation considering only nodes themselves, we use graph convolution to capture the structural information. We apply graph convolution to each node in mAST. Then, we can have node representations considering the structural features as well as the node features.

The graph-convolutioned mASTs are converted into sequences by pre-order traversal, and the sequences are given to the Transformer encoder. Since structurally-neighboring nodes have similar representations, the Transformer encoder can effectively capture not only the sequential features but also the structural features such as sentences or blocks of source code. After the Transformer encoder generates the representations by reflecting the sequential and structural information, the Transformer decoder generates summaries.

3.1 Graph Convolution Network

Graph convolutional network (Kipf and Welling, 2017) is one of graph neural networks for representing nodes based on neighborhood features of each node in graph data. In this paper, graph convolution layers are used to capture the structure information of mASTs. Since the mAST extracted from a given code C is a graph, we denote an AST as $G(C) = \{V, E\}$, where V is a set of nodes and E is a set of edges. Initially, nodes in V are one-hot encoded tokens and then mapped into representation vectors, X, by the embedding layer.

Given representation X of nodes and E of edges, new representations of nodes are calculated by graph convolution layers as follows.

$$H^0 = X, X \in \mathbb{R}^{n \times d}$$

$$H^{(l+1)} = \sigma(AH^{(l)}W^{(l)}), W^l \in \mathbb{R}^{d \times d}$$

where A is the adjacency matrix, W^l is the graph convolution weight matrix in the l-th layer, σ is the activation function, n is the total number of nodes in an mAST, and d is the embedding dimension. The feature of each node represented by the graph convolution layer is denoted as H. In the experiment, the dimension of the weight matrix in a graph convolution is d=512.

3.2 Transformer Encoder-Decoder

After graph convolution layers, the mAST is converted into a sequence by pre-order traversal. The pre-order traversal is applied to the original AST, not the mAST, because mASTs are transformed

Dataset	**Java**	**Python**
Train	69,708	55,538
Valid	8,714	18,505
Test	8,714	18,502
Unique leaf nodes in ASTs	106	54
Unique non-leaf nodes in ASTs	57,372	101,229
Unique tokens in summaries	46,895	56,189
Avg. nodes in AST	131.72	104.11
Avg. tokens in summary	17.73	9.48

Table 1: Statistics of Java and Python dataset

into graphs by adding sibling edges. The mAST is used to obtain the structural representation of nodes by considering nodes and their neighbors. Since the original AST contains the original structure of the source code, we use it to obtain a sequence.

The Transformer encoder and decoder follow the graph convolution layers. The sequence of the mAST nodes is processed into the Transformer encoder. The Transformer architecture is good at capturing long-term dependencies in a sequence. Since we used graph convolutions, which generate similar representations for structurally-neighboring nodes, the Transformer encoder can easily capture dependencies between nodes in the same code block, between similar code blocks, and between code blocks at the same level. As a result, the Transformer encoder can generate new representation vectors which well reflect sequential and the structural information.

Next, the Transformer decoder generates the token of summary from the vectors generated by the Transformer encoder. In the experiment, the dimension of nodes and summary tokens is d_{model}=512. The Transformer encoder and decoder are respectively composed of a stack of N = 6 layers.

4 Experiment

We perform various experiments to show the superiority of our model for source code summarization.

4.1 Setup

Datasets We evaluate our model using Java dataset (Hu et al., 2018b) and Python dataset (Wan et al., 2018). The statistics of the experiment datasets are shown in Table 1. We used the Java parser used by Alon et al. (2019) and the Python parser used by Wan et al. (2018) for extracting the abstract syntax tree of the code.

Metrics We adopt 3 performance metrics: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and ROUGE-L (Lin, 2004).

Baselines We compare our model with baseline models based on sequential information by Iyer et al. (2016); Hu et al. (2018a,b); Wei et al. (2019); Ahmad et al. (2020) and based on structural information by Eriguchi et al. (2016); Wan et al. (2018). We refer to the baseline results reported by Ahmad et al. (2020).

Hyper-parameters We set the maximum length to 200, and the vocabulary sizes for code and summary to 50,000 and 30,000, respectively. We train our proposed model using Adam optimizer (Kingma and Ba, 2015). The mini-batch size and dropout rate are 32 and 0.2. We set the maximum training epoch to 200, and use early stopping. We adopt beam search during inference time and set the beam size to 4.

4.2 Quantitative Result

Overall Result Table 2 shows the overall performance of the models. We present three proposed models: *AST-Only*, *AST+GCN* and *mAST+GCN*.

AST-Only is the proposed model without graph convolution layers. The model converts code snippets into ASTs and does not include graph convolutions. The sequenced AST nodes are given to the Transformer encoder and decoder. We present this model to verify how much ASTs are effective for source code summarization. It performs better than the baselines except for TransRel model proposed by Ahmad et al. (2020). This result shows that AST, which has more structural information on source code, is better than simple code for source code summarization.

AST+GCN is the proposed model with graph convolution layers but without AST modification (adding sibling edges). Code snippets are converted into ASTs and node representations are generated by graph convolutions. The sequenced graph-convolutioned AST nodes are input to the Transformer encoder and decoders. This model can verify how much the graph convolutions are useful. It shows better performance than the baselines.

mAST+GCN is the proposed model with modified ASTs by adding sibling edges and with graph convolution layers. It outperforms all baseline models. The performance improves by 0.91 and 0.3 BLEU, 0.74 and 0.35 METEOR, and 0.06 and 0.08

Methods	Java			Python		
	BLEU	METEOR	ROUGE-L	BLEU	METEOR	ROUGE-L
CODE-NN (Iyer et al., 2016)	27.60	12.61	41.10	17.36	09.29	37.81
Tree2Seq (Eriguchi et al., 2016)	37.88	22.55	51.50	20.07	08.96	35.64
RL+Hybrid2Seq (Wan et al., 2018)	38.22	22.75	51.91	19.28	09.75	39.34
DeepCom (Hu et al., 2018a)	39.75	23.06	52.67	20.78	09.98	37.35
API+CODE (Hu et al., 2018b)	41.31	23.73	52.25	15.36	08.57	33.65
Dual Model (Wei et al., 2019)	42.39	25.77	53.61	21.80	11.14	39.45
TransRel (Ahmad et al., 2020)	44.58	26.43	54.76	32.52	19.77	46.73
Proposed Model: AST-Only	44.76	26.75	53.93	31.59	19.16	45.48
Proposed Model: AST+GCN	45.30	**27.26**	54.45	32.41	19.77	46.35
Proposed Model: mAST+GCN	**45.49**	27.17	**54.82**	**32.82**	**20.12**	**46.81**

Table 2: Comparison of our proposed model with the baseline models.

ROUGE-L points in comparison to TranRel for Java and Python datasets, respectively. The proposed model with the AST modification has better performances on BLEU, METEOR, and ROUGE-L (excepts for METEOR in Java) than without the modification. This proves that modified ASTs help models learn more structural information of code than general ASTs.

Position of graph convolution layers We perform additional experiments with different positions of graph convolution layers. The positions are the front of the Transformer encoder, the back of the Transformer encoder, and both the front and back of the Transformer encoder.

Table 3 shows the performance scores according to the position of the graph convolution layers. The *front* model is the same as *mAST+GCN* which has one graph convolution layer in front of the encoder.

The *back* model does not have graph convolution layers in front of the Transformer encoder but has one next to the encoder. Nodes in an mAST are input to the encoder without graph convolutions, but graph convolutions are applied to the output of the encoder. Since the Transformer encoder can catch structural patterns in simple sequences, the graph convolution in the back of the encoder may work better than the one in front because it can feed more sharp structural information to the decoder.

The *front+back* model has two graph convolution layers: one in front and the other in the back of the Transformer encoder. It may catch much stronger structural patterns. The structurally encoded representations by a graph convolution layer are fed to the encoder and the output of the encoder is structurally enhanced once more by a graph convolution layer.

Table 3 shows the best result when located in front of the Transformer encoder. The *front+back* model is next to the *front*, and the *back* is the worst, which means that the graph convolution before the encoder is effective.

Since the Transformer has the ability to extract comprehensive features considering not only sequential but also structural information in sequences, the convolution layer in the back of the encoder may destruct such features and degrade the performance. However, the graph convolution layer in front of the encoder can help the encoder analyze structural patterns and to extract better features because it enhances structural information.

Position	BLEU	METOR	ROUGE-L
Java Dataset			
front	45.49	27.17	54.82
back	44.56	25.97	54.07
front+back	45.06	26.51	54.47
Python Dataset			
front	32.82	20.12	46.81
back	32.31	19.70	46.42
front+back	32.58	19.78	46.58

Table 3: Performance by position of graph convolution layers

Number	BLEU	METOR	ROUGE-L
Java Dataset			
1	45.49	27.17	54.82
2	44.72	26.70	53.87
3	44.14	25.62	53.46
Python Dataset			
1	32.82	20.12	46.81
2	31.80	19.31	45.56
3	30.91	18.41	44.24

Table 4: Performance by number of graph convolution layers

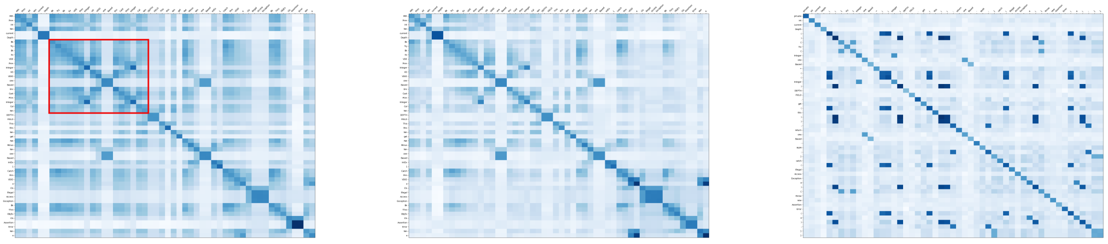

(a) mAST+GCN (b) AST+GCN (c) Baseline

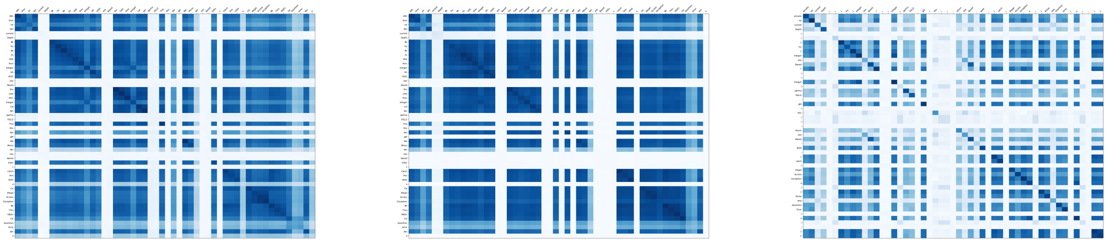

(d) mAST+GCN (e) AST+GCN (f) Baseline

Figure 4: Attention maps of mAST+GCN, AST+GCN, and the baseline models for a code in Figure 5. (a), (b) and (c) are the attention maps of the first Transformer encoder layer and (d), (e), and (f) are the attention maps of the last Transformer encoder layer. A red box in (a) represents blocks in the snippets. We can see that the structural features are clearly captured in the red box. Our model effectively captures large and hierarchical structural features.

```
private int currentDepth(){
  try {
    Integer oneBased=((Integer)DEPTH_FIELD.get(this));
    return oneBased - 1;
  }
  catch ( IllegalAccessException e) {
    throw new AssertionError(e);
  }
}
```

Figure 5: An example of Java code. We draw attention maps for this in Figure 4.

Number of graph convolution layers We analyze the performance according to the number of graph convolution layers. Graph convolutions are effective at capturing structural features, so more layers can help improve the performance. We tried one to three layers in front of the encoder.

Table 4 is the result of each model with 1, 2 and 3 layers. The results show that our proposed model with one graph convolution layer in front of the Transformer encoder has better performance than others. We think that this is because the graph structure of AST is not as complex as the general graph structure. So, the node representation has the over-smoothing problem when the graph convolution layer is stacked deep.

4.3 Qualitative Result

We present the qualitative analysis of our model. The attention map of an example code is compared to show how much our model catches the structural information. In order to further validate the performance metrics of our model, we perform a human evaluation on randomly sampled code snippets.

Attention Map Comparison We analyze attention maps of *mAST+GCN*, *AST+GCN* and the baseline by Ahmad et al. (2020) to verify how our model generates node representations compared to the others. Since we try to emphasize the structural information, we need to verify how much our model reflects the structural information to generate representations.

We observe the attention maps for the sample code in Figure 5. We draw an attention map by evaluating the pairwise dot product of the output of a Transformer layer in the encoder. For *mAST+GCN* and *AST+GCN*, the output of a layer is the sequence of the mAST nodes, and for the baseline, it is the sequence of the program tokens.

We compare the attention maps of the first and the last Transformer layer in the encoder of each model. Figure 4a, 4b and 4c are the attention maps of the first layer of *mAST+GCN*, *AST+GCN* and the baseline. Figure 4d, 4e and 4f are the attention maps of the last layer of *mAST+GCN*, *AST+GCN* and the baseline.

Rectangles on the diagonal as shown in a red box in Figure 4a represent blocks in the snippets. Since nodes or tokens in a block may have high similarities, we can see rectangles along with the diagonal. In the attention maps of the first layers, the rectangles are faint because the structural information has not been processed much yet, but we can see many distinct rectangles which means that the structural features are clearly captured.

If we compare the attention maps by our models and the baseline, there are many small rectangles in the baseline. On the contrary, in our models, we can see a few large rectangles in which there are small rectangles. We see a hierarchical structure in the attention maps of our model.

The fact that the baseline produces many small rectangles implies that the baseline can capture only small structural features. We can also note that these small structural features are smaller than statements, considering that the example snippets have only 4 lines. The baseline hardly captures large structural features.

On the other hand, our model effectively captures large and hierarchical structural features. We can easily identify rectangles that match with statements or blocks in the attention maps by our proposed models.

The attention maps from *mAST+GCN* and *AST+GCN* are very similar. However, we can see differences in each attention map of the first and the last layer. If we compare the first layer attention maps, the blocks of *mAST+GCN* are more distinct, which implies that the modification of ASTs by adding sibling edges is helpful to model structural information. If we compare the last layer attention maps, we can see that hierarchical structures of rectangles are clear in the map by *mAST+GCN*, which also says that the modification is effective to capture structural features.

Human Evaluation We performed human evaluation (Kryscinski et al., 2019) on the Java dataset to prove the effectiveness of how good summaries our model generates. We randomly choose 100 snippets and ask 4 people with knowledge of the Java language to evaluate the summaries. They are CS graduate students and have many years of experience in Java languages. We ask them to evaluate the 3 following aspects:

- Fluency (quality of the summary)
- Relevance (selection of the important content from source code)
- Coverage (selection of the whole content of source code)

	Fluency	**Relevance**	**Coverage**
Wins	**146**	**145**	**144**
Losses	130	135	140
Ties	124	120	116

Table 5: Human evaluation of the appropriateness of the generated summaries on the Java dataset. We ask annotators to select a more appropriate summary from two candidates generated by different models. Our proposed model outperforms the baseline.

We show pairs of summaries from our model and the baseline (Ahmad et al., 2020) to the annotators, and ask them to select one of win, tie, and loss in the three aspects, respectively. Our model shows superiority in all aspects as shown in Table 5. The scores of fluency and relevance are higher than the baseline, which means that our model generates more appropriate summaries using more natural expressions.

Figure 6 shows some examples of summaries for the qualitative comparison. We choose 6 Java snippet examples. We choose them from the snippets on which all the annotators make the same decision in each aspect. The three snippets on the left are the ones that the annotators choose win (our model is better) and the right ones for loss.

5 Related Work

As techniques and methods of deep learning have developed, researches for source code summarization have been studied based on sequence-to-sequence models. Iyer et al. (2016) proposed a model that performed source code summarization task for the first time. Allamanis et al. (2016) summarized the source code using a convolutional attention network model. Hu et al. (2018a) proposed an RNN-based sequence-to-sequence model using the pre-order traversal sequence of the abstract syntax tree. Also, Hu et al. (2018b) summarized source code with the knowledge on imported APIs using two encoders (source code encoder and API encoder). Ahmad et al. (2020) proposed a Transformer model with a relative position for summarizing source code. Wei et al. (2019) proposed a dual model that learned the code and summary sequence simultaneously. Wan et al. (2018) adopted

```
protected void removeClassifiers(int[] indices){
int i;
 if (indices == null) {
  m_ModelClassifiers.removeAllElements();
 }
else {
  for (i=indices.length - 1; i >= 0; i--)     m_ModelClassifiers.remove(indices[i]);
 }
 setModified(true);
}
```

Fluency

Baseline Model : remove a certain indices .
Proposed Model : removes the specified datasets .
References : removes the specified classifiers .

```
private void buildPieces(){
 pieces=new Piece[pathArray.size()];
 Paint paint=new Paint();
 Matrix matrix=new Matrix();
 Canvas canvas=new Canvas();
 for (int i=0; i < pieces.length; i++) {
       ...
 }
 Arrays.sort(pieces);
}
```

Fluency

Baseline Model : build the final bitmap - pieces to draw in animation
Proposed Model : build the final path to draw in animation
References : build the final bitmap-pieces to draw in animation

```
public static String encode(byte[] data){
 int start=0;
 int len=data.length;
 StringBuffer buf=new StringBuffer(data.length * 3 / 2);
 int end=len - 3;
 int i=start;
 int n=0;
 while (i <= end) {
              ...
else   if (i == start + len - 1) {
   int d=(((int)data[i]) & 0x0ff) << 16;
   buf.append(legalChars[(d >> 18) & 63]);
   buf.append(legalChars[(d >> 12) & 63]);
   buf.append("==");
 }
 return buf.toString();
}
```

Relevance

Baseline Model : translates the specified byte array to a string .
Proposed Model : translates a byte array to a base 64 encoded string .
References : base64 encode the given data .

```
private static Stream<Method> extractMethods(Class clazz){
 try {
   Method[] methods=clazz.getMethods();
   if (methods.length > 0) {
     return Stream.of(methods);
   }
 }
catch (  Exception|Error e) {
   LOG.warn("Problems loading class at startup: {}",clazz,e);
 }
 return Stream.empty();
}
```

Relevance

Baseline Model : extracts the method invocation from a methods if it exists .
Proposed Model : extracts the migration methods of the given method .
References : extract a set of methods from a given class .

```
public void deleteLeaves(String name){
 for (int i=0; i < _leaves.size(); i++) {
  CatalogTreeLeaf leaf=_leaves.get(i);
  if (name.equals(leaf.getName())) {
    _leaves.remove(i);
  }
 }
}
```

Coverage

Baseline Model : removes the specified name from the table
Proposed Model : removes a leaf with the specified name .
References : leaves can be used for many-to-many relations

```
public void releaseAnyConnection(Connection connection)throws SQLException {
 try {
   connection.close();
 }
catch (  Exception ex) {
   throw new RuntimeException(ex);
 }
}
```

Coverage

Baseline Model : releases a connection back to the database
Proposed Model : releases a connection back .
References : release a non-shard-specific connection .

Figure 6: Examples for the qualitative comparison. The left examples are chosen from the ones that have 4 wins (all the annotators agree that our model is better), and the right examples are chosen from the opposite cases.

reinforcement learning to summarize source code. These approaches mainly focused on the sequential and context information of code, but little considered the structural information about the relationship between code tokens.

There are also studies that convert source code to AST to represent the structure of source code. Liang and Zhu (2018) proposed a tree-based recursive neural network to represent the syntax tree of code. Shido et al. (2019) represented source code using the tree structure encoder of tree-LSTM. Harer et al. (2019) adopted tree-transformer to encode the structure of ASTs. Fernandes et al. (2019) proposed the structured neural model for source code summarization. Alon et al. (2019) represented source code based on AST paths between pairs of tokens. LeClair et al. (2020) proposed a model that encoded the AST of source code using graph neural networks. These approaches utilized ASTs to capture structural features, but less considered the sequence characteristics of code in a program language.

6 Conclusion

We proposed a model that learned both the sequential and the structural features of code for source code summarization. We adopted the abstract syntax tree (AST) and graph convolution to model the structural information and the Transformer to model the sequential information. We also modified the AST to deliver more structural information.

We verified that modified ASTs and graph convolutions were very effective to capture the structural features of code through quantitative and qualitative analysis. We also showed the superiority of our model over the state-of-the-art for source code summarization by experiments and human evaluations.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, Artificial Intelligence Graduate School Program (Sungkyunkwan University)). This research was supported by the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program) (2019-0-01579) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A transformer-based approach for source code summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4998–5007, Online. Association for Computational Linguistics.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2091–2100. JMLR.org.

Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. 2019. code2seq: Generating sequences from structured representations of code. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Structured neural summarization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-transformer: A transformer-based method for correction of tree-structured data. *arXiv preprint arXiv:1908.00449*.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018a. Deep code comment generation. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 200–20010. IEEE.

Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018b. Summarizing source code with transferred API knowledge. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2269–2275. ijcai.org.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. In *Proceedings of the 28th International Conference on Program Comprehension*, pages 184–195.

Yuding Liang and Kenny Qili Zhu. 2018. Automatic generation of text descriptive comments for code blocks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5229–5236. AAAI Press.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Yusuke Shido, Yasuaki Kobayashi, Akihiro Yamamoto, Atsushi Miyamoto, and Tadayuki Matsumura. 2019. Automatic source code summarization with extended tree-lstm. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 397–407.

Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code generation as a dual task of code summarization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6559–6569.

Energy-based Unknown Intent Detection with Data Manipulation

Yawen Ouyang, Jiasheng Ye, Yu Chen, Xinyu Dai[*],
Shujian Huang and **Jiajun Chen**
National Key Laboratory for Novel Software Technology, Nanjing University, China
`{ouyangyw,yejiasheng,cheny98}@smail.nju.edu.cn`
`{daixinyu,huangsj,chenjj}@nju.edu.cn`

Abstract

Unknown intent detection aims to identify the out-of-distribution (OOD) utterance whose intent has never appeared in the training set. In this paper, we propose using energy scores for this task as the energy score is theoretically aligned with the density of the input and can be derived from any classifier. However, high-quality OOD utterances are required during the training stage in order to shape the energy gap between OOD and in-distribution (IND), and these utterances are difficult to collect in practice. To tackle this problem, we propose a data manipulation framework to **G**enerate high-quality **O**OD utterances with importance weigh**T**s (**GOT**). Experimental results show that the energy-based detector fine-tuned by GOT can achieve state-of-the-art results on two benchmark datasets.

1 Introduction

Unknown intent detection is a realistic and challenging task for dialogue systems. Detecting out-of-distribution (OOD) utterances is critical when employing dialogue systems in an open environment. It can help dialogue systems gain a better understanding of what they do not know, which prevents them from yielding unrelated responses and improves user experience.

A simple approach for this task relies on the softmax confidence score and achieves promising results (Hendrycks and Gimpel, 2017). The softmax-based detector will classify the input as OOD if its softmax confidence score is smaller than the threshold. Nevertheless, further works demonstrate that using the softmax confidence score might be problematic as the score for OOD inputs can be arbitrarily high (Louizos and Welling, 2017; Lee et al., 2018).

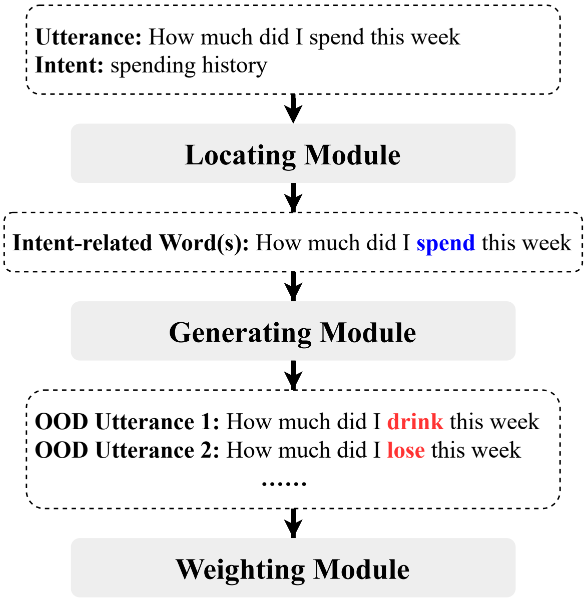

Figure 1: An overview of our framework GOT. For the utterance "How much did I spend this week" from CLINC150 dataset (Larson et al., 2019). Our locating module locates the intent-related word "spend". And then our generating module generates words "drink", "lose" to replace it and obtains OOD utterances. Finally, our weighting module assigns a weight for each of OOD utterances.

Another appealing approach is to use generative models to approximate the distribution of in-distribution (IND) training data and use the likelihood score to detect OOD inputs. However, Ren et al. (2019) and Gangal et al. (2019) find that likelihood scores derived from such models are problematic for this task as they can be confounded by background components in the inputs.

In this paper, we propose using *energy scores* (Liu et al., 2020) for unknown intent detection. The benefit is that energy scores are theoretically well aligned with the density of the inputs, hence more suitable for OOD detection. Inputs with higher energy scores mean lower densities, which can be

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2852–2861
August 1–6, 2021. ©2021 Association for Computational Linguistics

classified as OOD by the energy-based detector. Moreover, energy scores can be derived from any pre-trained classifier without re-training. Nevertheless, the energy gap between IND and OOD utterances might not always be optimal for differentiation. Thus we need auxiliary OOD utterances to explicitly shape the energy gap between IND and OOD utterances during the training stage (Liu et al., 2020). This poses a new challenge in that the variety of possible OOD utterances is almost infinite. It is impossible to sample all of them to create the gap. Zheng et al. (2019) demonstrate that OOD utterances akin to IND utterances, such as sharing the same phrases or patterns, are more effective, whereas these high-quality OOD utterances are difficult and expensive to collect in practice.

To tackle this problem, we propose a data manipulation framework **GOT** to generate high-quality OOD utterances as well as importance weights. GOT generates OOD utterances by perturbing IND utterances locally, which allows the generated utterances to be closer to IND. Specifically, GOT contains three modules: (1) a locating module to locate intent-related words in IND utterances; (2) a generating module to generate OOD utterances by replacing intent-related words with desirable candidate words, evaluated in two aspects: whether the candidate word is suitable given the context, and whether the candidate word is irrelevant to IND; (3) a weighting module to reduce the weights of potential harmful generated utterances. Figure 1 illustrates the overall process of GOT. Experiments show that the generated weighted OOD utterances can further improve the performance of the energy-based detector in unknown intent detection. Our code and data will be available at: `https://github.com/yawenouyang/GOT`.

To summarize, the key contributions of the paper are as follows:

- We propose using energy scores for unknown intent detection. We conduct experiments on real-world datasets including CLINC150 and SNIPS to show that the energy score can achieve comparable performance as strong baselines.
- We put forward a new framework GOT to generate high-quality OOD utterances and reweight them. We demonstrate that GOT can further improve the performance of the energy score by explicitly shaping the energy gap and achieves state-of-the-art results.
- We show the generality of GOT by applying generated weighted OOD utterances to fine-tune the softmax-based detector, and the fine-tuned softmax-based detector can also yield significant improvements.

2 Related Work

Lane et al., 2006, Manevitz and Yousef, 2007 and Dai et al., 2007 address OOD detection for the text-mining task. Recently, this problem has attracted growing attention from researchers (Tur et al., 2014; Fei and Liu, 2016; Fei et al., 2016; Ryu et al., 2017; Shu et al., 2017). Hendrycks and Gimpel (2017) present a simple baseline that utilizes the softmax confidence score to detect OOD inputs. Shu et al. (2017) create a binary classifier and calculate the confidence threshold for each class. Some distance-based methods (Oh et al., 2018; Lin and Xu, 2019; Yan et al., 2020) are also used to detect unknown intents as OOD utterances highly deviate from IND utterances in their local neighborhood. Simultaneously, with the advancement of deep generative models, learning such a model to approximate the distribution of training data is possible. However, Ren et al. (2019) find that likelihood scores derived from these models can be confounded by background components, and propose a likelihood ratio method to alleviate this issue. Gangal et al. (2019) reformulate and apply this method to unknown intent detection.

Different from these methods, we introduce the energy score for this task. Liu et al. (2020) prove that the energy score is theoretically aligned with the density of the input, and can be derived from any classifier without re-training, hence desirable for our task. We further propose a data manipulation framework to generate high-quality OOD utterances to shape the energy gap between IND and OOD utterances.

Note that there are some related works that also generate OOD samples to improve OOD detection performance. Lee et al. (2017) generate OOD samples with Generative Adversarial Network (GAN) (Goodfellow et al., 2014), and Zheng et al. (2019) explore this method for unknown intent detection. However, there are two major distinctions between our study and these works. First, they generate OOD utterances according to continuous latent variables, which cannot be easily interpreted. In contrast, our framework generates utterances by performing local replacements to IND utterances, which is more interpretable to human. Second, our framework additionally contains a weighting module to reform the generated utterances. Our work is also inspired by Cai et al. (2020), which proposes a framework to augment the IND data, while our framework aims to generate OOD data.

3 Preliminary

In this section, we formalize unknown intent detection task. Then we introduce the energy score, and its superiority and limitations for this task.

3.1 Problem Formulation

Given a training dataset $\mathcal{D}_{\text{in}}^{\text{train}} = \{(\mathbf{u}^{(i)}, y^{(i)})\}_{i=1}^{N}$ where $\mathbf{u}^{(i)}$ is an utterance and $y^{(i)} \in Y_{\text{in}} = \{y_1, y_2, ..., y_K\}$ is its intent label. In testing, given an utterance, unknown intent detection aims to detect whether its intent belongs to existing intents Y_{in}. In general, unknown intent detection is an OOD detection task. The essence of all methods is to learn a score function that maps each utterance $\mathbf{u}$ to a single scalar that is distinguishable between IND and OOD utterances.

3.2 Energy-based OOD Detection

An energy-based model (LeCun et al., 2006) builds an energy function $E(\mathbf{u})$ that maps an input $\mathbf{u}$ to a scalar called energy score (i.e., $E : \mathbb{R}^D \to \mathbb{R}$). Using the energy function, probability density $p(\mathbf{u})$ can be expressed as:

$$p(\mathbf{u}) = \frac{\exp(-E(\mathbf{u})/T)}{Z}, \qquad (1)$$

where $Z = \int_{\mathbf{u}} \exp(-E(\mathbf{u})/T)$ is the normalizing constant also known as the partition function and T is the temperature parameter. Take the logarithm of both side of (1), we can get the equation:

$$\log p(\mathbf{u}) = -E(\mathbf{u})/T - \log Z. \qquad (2)$$

Since Z is constant for all input $\mathbf{u}$, we can ignore the last term $\log Z$ and find that the energy function $-E(\mathbf{u})$ is in fact linearly aligned with the log likelihood function, which is desirable for OOD detection (Liu et al., 2020).

The energy-based model has a connection with a softmax-based classifier. For a classification problem with K classes, a parametric function f maps each input $\mathbf{u}$ to K real-valued numbers (i.e., $f : \mathbb{R}^D \to \mathbb{R}^K$), known as logits. Logits are used to parameterize a categorical distribution using a softmax function:

$$p(y|\mathbf{u}) = \frac{\exp[f_y(\mathbf{u})/T]}{\sum_{y'} \exp[f_{y'}(\mathbf{u})/T]}, \qquad (3)$$

where $f_y(\mathbf{u})$ indicates the y^{th} index of $f(\mathbf{u})$, i.e., the logit corresponding the y^{th} class label. And these logits can be reused to define an energy function without changing function f (Liu et al., 2020; Grathwohl et al., 2020):

$$E(\mathbf{u}) = -T \cdot \log \sum_{y'} \exp\left[f_{y'}(\mathbf{u})/T\right]. \qquad (4)$$

According to the above, a classifier can be reinterpreted as an energy-based model. It also means the energy score can be derived from any classifier.

Due to its consistency with density and accessibility, we introduce the energy score for unknown intent detection, and utterances with higher energy scores can be viewed as OOD. Mathematically, the energy-based detector G can be described as:

$$G(\mathbf{u}; \delta, E) = \begin{cases} \text{IND} & E(\mathbf{u}) \leq \delta, \\ \text{OOD} & E(\mathbf{u}) > \delta, \end{cases} \qquad (5)$$

where δ is the threshold.

Although the energy score can be easily computed from the classifier, the energy gap between IND and OOD samples might not always be optimal for differentiation. To solve this problem, Liu et al. (2020) propose an energy-bounded learning objective to further widen the energy gap. Specifically, the training objective of the classifier combines the standard cross-entropy loss with a regularization loss:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{u},y)\sim\mathcal{D}_{\text{in}}^{\text{train}}}[-\log F_y(\mathbf{u})] + \lambda \cdot \mathcal{L}_{\text{energy}}, \qquad (6)$$

where $F(\mathbf{u})$ is the softmax output, λ is the auxiliary loss weight. The regularization loss is defined in terms of energy:

$$\begin{aligned}\mathcal{L}_{\text{energy}} &= \mathbb{E}_{(\mathbf{u},y)\sim\mathcal{D}_{\text{in}}^{\text{train}}}(\max(0, E(\mathbf{u}) - m_{\text{in}}))^2 \\ &+ \mathbb{E}_{\hat{\mathbf{u}}\sim\mathcal{D}_{\text{out}}^{\text{train}}}(\max(0, m_{\text{out}} - E(\hat{\mathbf{u}})))^2, \qquad (7)\end{aligned}$$

which utilizes both labeled IND data $\mathcal{D}_{\text{in}}^{\text{train}}$ and auxiliary unlabeled OOD data $\mathcal{D}_{\text{out}}^{\text{train}}$. This term differentiates the energy scores between IND and OOD samples by using two squared hinge loss with the margin hyper-parameters m_{in} and m_{out}.

Ideally, one has to sample all types of OOD utterances to create the gap, which is impossible in practice. Zheng et al. (2019) demonstrate that OOD utterances akin to IND utterances could be more effective, but more difficult to collect. To address this problem, we propose a data manipulation framework, which can generate these high-quality OOD utterances and assign each generated utterance an importance weight to reduce the impact of potential bad generation.

4 Approach

In this section, we will introduce our data manipulation framework GOT in detail. GOT aims to generate high-quality OOD utterances by replacing intent-related words in IND utterances, and then assign a weight to each generated OOD utterance. Eventually, the weighted OOD utterances can be used to shape the energy gap.

4.1 Locating Module

Since not all words in utterances are meaningful, such as stop words, when generating OOD utterances, replacing these words may not change the intent. It is more efficient and effective to replace those intent-related words. Hence, we design an intent-related score function S to measure how a word w related to an intent y:

$$S(w,y) = \sum_{\mathbf{u}\in\mathcal{D}_y^{\text{train}}} \sum_{w_j\in\mathbf{u}} \mathbb{I}(w_j = w)[\log p(w_j|\boldsymbol{w}_{<j}, y) - \log p(w_j|\boldsymbol{w}_{<j})], \quad (8)$$

where $\mathcal{D}_y^{\text{train}}$ is the subset of $\mathcal{D}_{\text{in}}^{\text{train}}$, which contains utterances with intent y, $\mathbb{I}$ is the indicator function, w_j is the j^{th} word in $\mathbf{u}$, and $\boldsymbol{w}_{<j} = w_1, ..., w_{j-1}$.

Given w and y, the intent-related score function is the sum of the log-likelihood ratios for all w in $\mathcal{D}_y^{\text{train}}$. If w is related to y, w tends to occur more frequently in $\mathcal{D}_y^{\text{train}}$ than other words. For each occurrence of w, i.e., w_j equals w, $p(w|\boldsymbol{w}_{<j}, y)$ should be higher than $p(w|\boldsymbol{w}_{<j})$ as the former is additionally conditioned on the related y, while the latter is not, hence resulting in a higher $S(w,y)$. In contrast, if w is not related to y, $p(w|\boldsymbol{w}_{<j}, y)$ is much less likely to be higher than $p(w|\boldsymbol{w}_{<j})$, or w tends to have a lower frequency in $\mathcal{D}_y^{\text{train}}$, hence $S(w,y)$ is likely to be small. Therefore, $S(w,y)$ can serve as a valid score function to measure how a word w is related to an intent y.

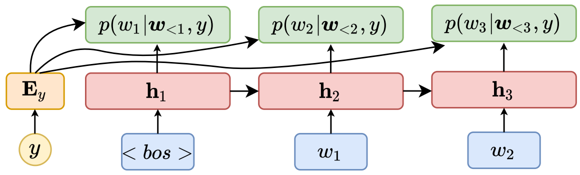

Figure 2: A class-conditional language model to estimate $p(w_j|\boldsymbol{w}_{<j}, y)$.

With the help of S, given an utterance to be replaced and its intent label, the locating module calculates the intent-related score for each word in this utterance, and a word with a higher score (i.e., larger than a given threshold) can be viewed as an intent-related word.

Implementation: We use two generative models to estimate $p(w_j|\boldsymbol{w}_{<j}, y)$ and $p(w_j|\boldsymbol{w}_{<j})$ separately. Specifically, we train a class-conditional language model (Yogatama et al., 2017) with $\mathcal{D}_{\text{in}}^{\text{train}}$ to estimate $p(w_j|\boldsymbol{w}_{<j}, y)$, shown in Figure 2. To predict the word w_j, we can combine the hidden state h_j with the intent embedding from a learnable label embedding matrix E_y, then pass it through a fully connected (FC) layer and a softmax layer to estimate the word distribution. In the training process, the input is the utterance with its intent from $\mathcal{D}_{\text{in}}^{\text{train}}$, and the training objective is to maximize the conditional likelihood of utterances. To estimate $p(w_j|\boldsymbol{w}_{<j})$, we directly use pre-trained GPT-2 (Radford et al., 2019) without tuning. Note that the whole training process only needs $\mathcal{D}_{\text{in}}^{\text{train}}$, and does not need auxiliary supervised data.

4.2 Generating Module

After detecting intent-related words in the utterance $\mathbf{u}$, for each of the intent-related words w_t, the generating module aims to generate the replacement words from the vocabulary set to replace w_t and obtain OOD utterances. We design a candidate score function Q to measure the desirability of the candidate word c:

$$Q(c; \mathbf{u}, w_t) = \log p(c|\boldsymbol{w}_{<t}, \boldsymbol{w}_{>t}) - \log \sum_{y\in Y_{\text{in}}} p(c|\boldsymbol{w}_{<t}, y)p(y). \quad (9)$$

The first term of the right hand side is the log-likelihood of c conditioned on the context of w_t; the higher it is, the more suitable c is given the context. The second term of the right hand side is the negative log of the average likelihoods of c conditioned on the IND label and previous context;

the higher it is, the less relevant c is to IND utterances. Therefore, if c has a higher candidate score, that means it fits the context well and has a low density under the IND utterance distribution, thus can be selected as the replacement word to replace w_t. The resulting generated OOD utterance is:

$$\hat{\mathbf{u}} = \{\boldsymbol{w}_{<t}, c, \boldsymbol{w}_{>t}\}. \tag{10}$$

Implementation: Similar with the locating module, we also do not need auxiliary supervised data to train the generating module. We use the same class-conditional language model mentioned in Section 4.1 to estimate $p(c|\boldsymbol{w}_{<t}, y)$. $p(y)$ is the training set label ratios. To estimate $p(c|\boldsymbol{w}_{<t}, \boldsymbol{w}_{>t})$, we use pre-trained BERT (Devlin et al., 2018) without tuning.

4.3 Weighting Module

Since we cannot ensure the generation process is perfect, given a generated OOD utterance set $\mathcal{D}_{\text{out}}^{\text{gen}} = \{\hat{\mathbf{u}}^{(i)}\}_{i=1}^{M}$, there might be some unfavorable utterances that are useless or even harmful for tuning the classifier. To fit these utterances, the generalization ability of the classifier will decrease. The weighting module aims to assign these utterances small weights.

We first use Equation 6 as the loss function to train a classifier by taking $\mathcal{D}_{\text{out}}^{\text{gen}}$ as $\mathcal{D}_{\text{out}}^{\text{train}}$. Then we calculate the influence value $\phi \in \mathbb{R}$ (Wang et al., 2020) for each generated utterance $\hat{\mathbf{u}}$. The influence value approximates the influence of removing this utterance on the loss at validation samples. An utterance with positive ϕ implies that its removal will reduce the validation loss and strengthen the classifier's generalization ability, thus we should assign it a small weight.[*] In particular, given ϕ, we calculate weight α as follows:

$$\alpha = \frac{1}{1 + e^{\frac{\gamma\phi}{\max_\phi - \min_\phi}}}, \tag{11}$$

where $\gamma \in \mathbb{R}^{+}$ is used to make the weight distribution flat or steep, $\max_\phi$ and $\min_\phi$ are the maximum and minimum influence value of utterances in $\mathcal{D}_{\text{out}}^{\text{gen}}$.

Implementation: We still do not need auxiliary supervised data for this module. The validation loss is the cross-entropy loss on the validation set.

Algorithm 1 Data Manipulation Process

```
Input: Training set $\mathcal{D}_{\text{in}}^{\text{train}}$, intent-related score function $S$,
    candidate score function $Q$, intent-related word threshold
    $\epsilon$, candidate number K, weight term $\gamma$
Output: Generated weighted OOD utterances set $\mathcal{D}_{\text{out}}^{\text{gw}}$
 1: $\mathcal{D}_{\text{out}}^{\text{gen}} = \{\}$ # generated OOD utterances without weights
 2: for $(\mathbf{u}, y) \in \mathcal{D}_{\text{in}}^{\text{train}}$ do
 3:     for $w_j \in \mathbf{u}$ do
 4:         if $S(w_j, y) > \epsilon$ then
 5:             $\mathcal{C} = \mathrm{top{-}K_c\, Q(c; \mathbf{u}, w_j)}$
 6:             for $c \in \mathcal{C}$ do
 7:                 $\hat{\mathbf{u}} = \{\boldsymbol{w}_{<j}, c, \boldsymbol{w}_{>j}\}$
 8:                 Add $\hat{\mathbf{u}}$ into $\mathcal{D}_{\text{out}}^{\text{gen}}$
 9:             end for
10:         end if
11:     end for
12: end for
13: $\mathcal{D}_{\text{out}}^{\text{gw}} = \{\}$        # generated weighted OOD utterances
14: for $\hat{\mathbf{u}} \in \mathcal{D}_{\text{out}}^{\text{gen}}$ do
15:     Calculate the weight $\alpha$ by Equation 11
16:     Add $(\hat{\mathbf{u}}, \alpha)$ into $\mathcal{D}_{\text{out}}^{\text{gw}}$
17: end for
18: return $\mathcal{D}_{\text{out}}^{\text{gw}}$
```

4.4 Overall Data Manipulation Process

We summarize the process of GOT in Algorithm 1. Line 4 shows that w_j can be viewed as an intent-related word for y if $S(w_j, y)$ is greater than the intent-related word threshold ϵ. Line 5 shows that we generate K replacement words with the top-K $Q(c; \mathbf{u}, w_t)$.

4.5 Shape the energy gap with GOT

After obtaining weighted OOD utterances set $\mathcal{D}_{\text{out}}^{\text{gw}}$, we can explicitly shape the energy gap with them, resulting in IND utterances with smaller energy scores and OOD utterances with higher energy scores. Specifically, we redefine the regularization loss in Equation 6 as follows and use it to re-train the classifier:

$$\begin{aligned}\mathcal{L}_{\text{energy}} &= \mathbb{E}_{(\mathbf{u},y)\sim\mathcal{D}_{\text{in}}^{\text{train}}}(\max(0, E(\mathbf{u}) - m_{\text{in}}))^2 \\ &+ \mathbb{E}_{(\hat{\mathbf{u}},\alpha)\sim\mathcal{D}_{\text{out}}^{\text{gw}}}\alpha(\max(0, m_{\text{out}} - E(\hat{\mathbf{u}})))^2. \end{aligned} \tag{12}$$

In the testing process, we can calculate the energy score for the utterance by Equation 4, and identify whether it is OOD by Equation 5.

5 Experimental Setup

5.1 Datasets

To evaluate the effectiveness of the energy score and our proposed framework, we conducted experiments on two public datasets:

- CLINC150[†] (Larson et al., 2019): this dataset

---

[*] Details about how to calculate the influence can be found in (Koh and Liang, 2017; Wang et al., 2020).

[†] https://github.com/clinc/oos-eval

Statistic	CLINC150	SNIPS
Train	15000	9385
validation	3000	500
Test-IND	4500	486
Test-OOD	1000	214
Test-IND: Test-OOD	4.5: 1	2.3: 1
Number of IND classes	150	5

Table 1: Statistics of CLINC150 and SNIPS datasets.

covers 150 intent classes over ten domains. It supports some OOD utterances that do not fall into any of the system's supported intents to avoid splitting unknown intents manually.

- SNIPS[‡] (Coucke et al., 2018): this dataset is a personal voice assistant dataset that contains seven intent classes. SNIPS does not explicitly include OOD utterances. We kept two classes *SearchCreativeWork* and *SearchScreeningEvent* as unknown intents.

Table 1 provides summary statistics about these two datasets. Note that the training set and validation set do not include OOD utterances.

5.2 Metrics

We used four common metrics for OOD detection to measure the performance. AUROC (Davis and Goadrich, 2006), AUPR In and AUPR Out (Manning et al., 1999) are threshold-independent performance evaluations and higher values are better. FPR95 is the false positive rate (FPR) when the true positive rate (TPR) is 95%, and lower values are better.

Considering the smaller proportion of OOD utterances in the test set on two datasets, AUPR Out is more informative here.

5.3 Baselines

We introduce the following classifier-based methods as baselines:

- MSP (Hendrycks and Gimpel, 2017) trains a classifier with IND utterances and uses the softmax confidence score to detect OOD utterances.

- DOC (Shu et al., 2017) trains a binary classifier for each IND intent and uses maximum binary classifier output to detect OOD utterances.

- Mahalanobis (Lee et al., 2018) trains a classifier with softmax loss and uses Mahalanobis distance of the input to the nearest class-conditional Gaussian distribution to detect OOD utterances.

- LMCL (Lin and Xu, 2019) uses LOF (Breunig et al., 2000) in the utterance representation learned by a classifier. In training, they replace the softmax loss with LMCL (Wang et al., 2018).

- SEG (Yan et al., 2020) also uses LOF in the utterance representation. In training, they use semantic-enhanced large margin Gaussian mixture loss.

5.4 Implementation Details

For a fair comparison, all classifiers used in the above methods and ours are pre-trained BERT (Devlin et al., 2018) with a multi-layer perceptron (MLP). We select parameter values based on validation accuracy. For energy score, we follow Liu et al. (2020) to set T as 1, λ as 0.1, m_{in} as -8 and m_{out} as -5. For influence value, we focus on changes on MLP parameters and use stochastic estimation (Koh and Liang, 2017) with the scaling term 1000 and the damping term 0.003. For LMCL implementation, we set nearest neighbor number as 20, scaling factor s as 30 and cosine margin m as 0.35, which is recommended by Lin and Xu (2019). For SEG, we follow Yan et al. (2020) to set margin as 1 and trade-off parameter as 0.5.

For our framework, we set candidate number K as 2, weight term γ as 20. In particular, for CLINC150, we set threshold ϵ as 150 and generate 100 weighted utterances for each intent. For SNIPS, we set threshold ϵ as 1500 and generate 1800 weighted utterances for each intent. The difference in settings between two datasets is due to the different sizes of per intent in the training set.

6 Results and Analysis

In this part, we will show the results of different methods on two datasets and offer some further analysis.

6.1 Overall Results

As shown in Table 2, we can observe that:

- The energy score can achieve comparable results on two datasets. Note that on SNIPS dataset, the advantages of the energy score are not as obvious. The reason is that SNIPS dataset is not as challenging as CLINC150 dataset, most methods can achieve good results, such as AUPR out is greater than 0.9.

[‡] https://github.com/snipsco/nlu-benchmark

Method	CLINC150				SNIPS			
	AUROC ↑	**FPR95↓**	**AUPR In↑**	**AUPR Out↑**	**AUROC ↑**	**FPR95↓**	**AUPR In↑**	**AUPR Out↑**
MSP	0.955	0.164	0.990	0.814	0.951	0.370	0.970	0.922
DOC	0.943	0.221	0.985	0.790	0.938	0.493	0.956	0.910
Mahalanobis	0.969	0.118	0.993	0.871	0.979	0.088	0.989	0.964
LMCL	0.962	0.124	0.992	0.810	0.976	0.087	0.987	0.960
SEG	0.959	0.152	0.991	0.823	0.974	0.074	0.986	0.948
Energy	0.967	0.143	0.991	0.897	0.944	0.497	0.964	0.924
Energy + GOT	**0.973**	**0.114**	**0.993**	**0.914**	**0.989**	**0.039**	**0.995**	**0.972**
Energy + GOT w/o weighting	0.972	0.123	0.992	0.909	0.979	0.083	0.989	0.969

Table 2: AUROC, FPR95, AUPR In, AUPR Out on CLINC150, SNIPS datasets. Best results are in bold. All results are averaged across five seeds.

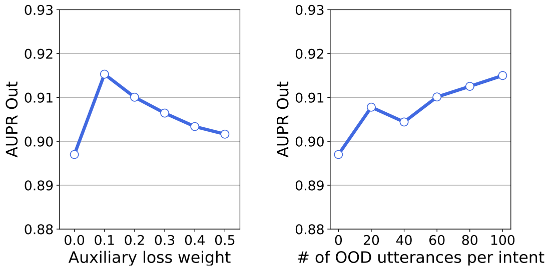

Figure 3: Effect of the auxiliary loss weight (left) and the number of generated weighted OOD utterances per intent (right).

- Energy + GOT achieves better results on two datasets as compared to the raw energy score. It indicates that our generated weighted OOD utterances can effectively shape the energy gap, resulting in more distinguishable between IND and OOD utterances.

- We also report ablation study results. “w/o weighting” is the energy score tuned by OOD utterances without reweighting. We can see that there is a decrease in performance on both datasets, which shows the advantage of the weighting module (p-value < 0.005).

6.2 Effect of Hyper-parameters

During the training process, we find that the method performance is sensitive to two hyper-parameters: auxiliary loss weight λ and the number of generated weighted OOD utterances per intent. We conduct two experiments to demonstrate their effects separately. We choose CLINC150 dataset as it is more challenging as mentioned before.

Auxiliary Loss Weight: We set the auxiliary loss weight λ from 0 to 0.5 with an interval of 0.1 to observe its impact.

Results are shown in Figure 3 (left). With the increase of auxiliary loss weight, the performance increases first and then decreases. $\lambda = 0.1$ achieves the highest AUPR Out 0.914 and outperforms $\lambda = 0$ with an improvement of 1.7% (AUPR Out). The results suggest that although shaping the energy gap can improve the performance, there exists a trade-off between optimizing the regularization loss and optimizing cross-entropy loss.

Method	AUROC ↑	FPR95↓	AUPR In↑	AUPR Out↑
Energy	0.967	0.143	0.991	0.897
Energy + Wiki	0.961	0.170	0.988	0.889

Table 3: Effect of using Wikipedia sentences to shape the energy gap.

Method	AUROC ↑	FPR95↓	AUPR In↑	AUPR Out↑
MSP	0.955	0.164	0.990	0.814
MSP + GOT	0.972	0.118	0.993	0.903

Table 4: Effect of using GOT to fine-tune the softmax-based detector.

Number of OOD Utterances: we compare the performance of generated weighted utterance numbers for each intent by adjusting the number from 0 to 100 with an interval of 20.

Results are shown in Figure 3 (right). As a whole, AUPR Out increases as more OOD utterances are incorporated into training. We can see that the performance is also improved even with a small generated number, which indicates the necessity of explicitly shaping the energy gap.

6.3 Compare with Wikipedia Sentences

An easy way to obtain OOD utterances is from the Wikipedia corpus. We investigate the effect of regarding Wikipedia sentences as OOD utterances to shape the energy gap on CLINC150 dataset. The Wikipedia sentences are from Larson et al. (2019) and the number is 14750.

As shown in Table 3, we can observe that these sentences cannot improve the performance and even have a negative effect (We experimented with several hyper-parameters, this is the best result we could get). After observing these Wikipedia sentences, we find that they have little relevance to

Intent	IND utterance and intent-related word [w]	Replacement word	Weight
Insurance	i need to know more about my health [**plan**]	problems	0.50
	what [**benefits**] are provided by my insurance	services	0.19
Credit Limit Change	can i get a higher limit on my american express [**card**]	ticket	0.46
	can you [**increase**] how much i can spend on my visa	guess	0.54
Reminder	can you list each item on my [**reminder**] list	contacts	0.50
	what's on the [**reminder**] list	agenda	0.78
Redeem Rewards	walk me through the process of cashing in on [**credit**] card points	those	0.22
	i have credit card [**points**] but don't know how to use them	privileges	0.50

Table 5: Weighted OOD utterances generated by GOT on CLINC150 dataset.

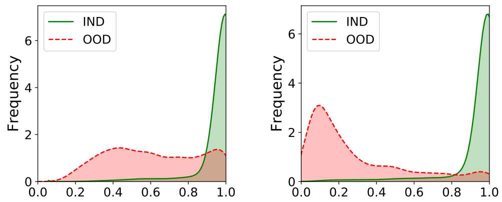

Figure 4: Histogram of the softmax confidence score from MSP (left) and MSP + GOT (right).

IND utterances. Therefore, simply using Wikipedia sentences is unrepresentative and ineffective for shaping the energy gap.

6.4 GOT for Softmax-based Detector

As mentioned in Section 1, when using the softmax-based detector, OOD inputs may also receive a high softmax confidence score. To tackle this problem, Lee et al. (2017) replace the cross entropy loss with the confidence loss. The confidence loss adds the Kullback-Leibler loss (KL loss) on the original cross entropy loss, which forces OOD inputs less confident by making their predictive distribution to be closer to uniform.

To verify the generality of GOT, we directly use the generated weighted OOD utterances to fine-tune the softmax-based detector with the confidence loss. The results are shown in Table 4. Our MSP + GOT has a significant improvement and outperforms MSP by 8.9% (AUPR Out). Figure 4 provides an intuitive presentation. The softmax confidence scores of OOD from MSP form smooth distributions (see Figure 4 (left)). In contrast, the softmax confidence scores of OOD from MSP + GOT concentrate on small values (see Figure 4 (right)). Overall the softmax confidence score is more distinguishable between IND and OOD after tuning by GOT.

6.5 Case Study for GOT

We sample some intents and showcase generated weighted OOD utterances in Table 5. We can observe that intent-related words that located by our locating module are diverse, containing not only words appeared in the intent label. The replacement word fits the context well, and the intent of the generated utterance is exactly changed in most conditions. Admittedly, GOT may have a bad generation, like replace "benefits" with "services" in the second utterance, which leads the generated utterance is still in-domain. Fortunately, the weighting module assigns these utterances a lower weight to reduce their potential harm.

7 Conclusion and Future Work

In this paper, we propose using energy scores for unknown intent detection and provide empirical evidence that the energy-based detector is comparable to strong baselines. To shape the energy gap, we propose a data manipulation framework GOT to generate high-quality OOD utterances and assign their importance weights. We show that the energy-based detector tuned by GOT can achieve state-of-the-art results. We further employ generated weighted utterances to fine-tune the softmax-based detector and also achieve improvements.

In the future, we will explore more operations, such as insertion, drop, etc., to enhance the diversity of generated utterances.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by NSFC Projects (Nos. 61936012 and 61976114), the National Key R&D Program of China (No. 2018YFB1005102).

References

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104.

Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. 2020. Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 506–514, San Diego, California. Association for Computational Linguistics.

Geli Fei, Shuai Wang, and Bing Liu. 2016. Learning cumulatively to become more knowledgeable. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1565–1574.

Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. 2019. Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog. *arXiv preprint arXiv:1912.12800*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. 2020. Your classifier is secretly an energy based model and you should treat it like one. *International Conference on Learning Representatinos*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Ian Lane, Tatsuya Kawahara, Tomoko Matsui, and Satoshi Nakamura. 2006. Out-of-domain utterance detection using classification confidences of multiple topics. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):150–161.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7167–7177. Curran Associates, Inc.

Ting-En Lin and Hua Xu. 2019. Deep unknown intent detection with margin loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5491–5496, Florence, Italy. Association for Computational Linguistics.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33.

Christos Louizos and Max Welling. 2017. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org.

Larry Manevitz and Malik Yousef. 2007. One-class document classification via neural networks. *Neurocomputing*, 70(7-9):1466–1481.

Christopher D Manning, Christopher D Manning, and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Kyo-Joong Oh, DongKun Lee, Chanyong Park, Young-Seob Jeong, Sawook Hong, Sungtae Kwon, and Ho-Jin Choi. 2018. Out-of-domain detection method based on sentence distance for dialogue systems. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 673–676. IEEE.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. 2019. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14680–14691.

Seonghan Ryu, Seokhwan Kim, Junhwi Choi, Hwanjo Yu, and Gary Geunbae Lee. 2017. Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems. *Pattern Recognition Letters*, 88:26–32.

Lei Shu, Hu Xu, and Bing Liu. 2017. DOC: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916, Copenhagen, Denmark. Association for Computational Linguistics.

Gokhan Tur, Anoop Deoras, and Dilek Hakkani-Tür. 2014. Detecting out-of-domain utterances addressed to a virtual personal assistant. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. 2018. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930.

Zifeng Wang, Hong Zhu, Zhenhua Dong, Xiuqiang He, and Shao-Lun Huang. 2020. Less is better: Unweighted data subsampling via influence function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6340–6347.

Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert Y.S. Lam. 2020. Unknown intent detection using Gaussian mixture model with an application to zero-shot intent classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1050–1060, Online. Association for Computational Linguistics.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Yinhe Zheng, Guanyi Chen, and Minlie Huang. 2019. Out-of-domain detection for natural language understanding in dialog systems. *arXiv preprint arXiv:1909.03862*.

Automatic Rephrasing of Transcripts-based Action Items

Amir DN Cohen[1,2], Amir Kantor[1], Sagi Hilleli[1] and Eyal Kolman[1]
Microsoft[1]
Department of Computer Science, Bar-Ilan University[2]
amirdnc@gmail.com,{amir.kantor,sagih,eyalko}@microsoft.com

Abstract

The automated *transcription* of *spoken language*, and meetings, in particular, is becoming more widespread as *automatic speech recognition* systems are becoming more accurate. This trend has significantly accelerated since the outbreak of the COVID-19 pandemic, which led to a major increase in the number of online meetings. However, the transcription of spoken language has not received much attention from the NLP community compared to documents and other forms of *written language*. In this paper, we study a variation of the summarization problem over the transcription of spoken language: given a transcribed meeting, and an *action item* (i.e., a commitment or request to perform a task), our goal is to generate a coherent and self-contained *rephrasing* of the action item. To this end, we compiled a novel dataset of annotated meeting transcripts, including human rephrasing of action items. We use state-of-the-art supervised text generation techniques and establish a strong baseline based on BART and UniLM (two pretrained transformer models). Due to the nature of natural speech, language is often broken and incomplete and the task is shown to be harder than an analogous task over email data. Particularly, we show that the baseline models can be greatly improved once models are provided with additional information. We compare two approaches: one incorporating features extracted by coreference-resolution. Additional annotations are used to train an auxiliary model to detect the relevant context in the text. Based on the systematic human evaluation, our best models exhibit near-human-level rephrasing capability on a constrained subset of the problem.

1 Introduction

Most of *natural language processing (NLP)* research focuses on written language, such as emails or Web pages, and less on an increasingly large body of spoken language converted to text via *automatic speech recognition (ASR)*. Particularly, today, more and more meetings are conducted online, especially since the COVID-19 social distancing constraints. Online meetings may be transcribed upon request, generating a huge amount of spoken language text.[1]

Spoken language has different characteristics from written language, and, from our experience, it is typically vaguer and harder to understand: sentences tend to be broken, less orderly, incomplete (relying on subtext), and prone to speech-to-text transformation errors (see Section 5).

Action items (AIs) are a common and particularly important part of workplace meetings. An AI is a commitment or a request to perform a certain task by any of the parties involved. For example,

> *"I will send you the file later today."*

AIs occur naturally during conversations, but are not always clear without relevant *context*. For example, the AI *"I will do it"* contains a commitment to act, but the nature of this action is unclear. Nevertheless, the context of the AI might make it clearer; e.g., if the AI is preceded by a sentence such as *"Can you prepare a presentation for Thursday?"*. In the context of written communication, action items have been researched in a variety of ways, such as AI detection (Bennett and Carbonell, 2007, 2005), summarization and rephrasing of AIs in emails (Mukherjee et al., 2020; Rambow et al., 2004), and more. Verbal communication, in contrast, has not received much attention.

In this paper, we focus on the task of *rephrasing* an action item from a transcribed meeting into a coherent and self-contained utterance. Such an utterance may be also referred to as a paraphrase,

---

[1] For example, Microsoft Teams, a popular online communication platforms, has reached 115 Million daily users on October 2020.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2862–2873
August 1–6, 2021.

or a summary. Conceptually, the rephrasing task may be split into two sub-tasks: one is *locating* those parts in the context that are relevant to the AI; the other is *forming* a coherent and self-contained AI sentence from the original AI and the relevant context. While even "off the shelf" summarization models (e.g., HIBERT (Zhang et al., 2020)) may do well in forming an action item sentence, manual observation of the data showed that such models often fail to locate relevant context in the transcript. As we show in Section 6, the same applies to models trained specifically for the rephrasing task too. To overcome this, we introduce *"hints"* – additional annotations that are added to the rephrasing models' input to help the model better locate relevant spans of text (see Figure 1). We use two kinds of hints: *coreference hints*, which are obtained from a pre-trained model for the coreference resolution task, and *context hints* that are generated by a model trained specifically for the task based on our train data.

Our contribution may be summarized as follows:

1. We created a new dataset (coined *AIR*) including AIs extracted from transcribed meetings, with human annotations of context and AI rephrasing.

2. We show that models intended for AI rephrasing in email perform considerably worse on transcript data. Accordingly, we train transformer-based models for AI rephrasing – both on email data, reporting new state-of-the-art performance, as well as on transcript data – so as to form a baseline for the task.

3. We show that the baseline can be greatly improved by adding "hints" to the model's input. This results in near-human-level performance on a constrained subset of the problem. We believe that this approach may apply to various other problems.

We support our claims by extensive experimentation and evaluation, including independent human evaluation.

2 Related work

While AI extraction and summarization of emails have been researched extensively (Lin et al., 2018; Mukherjee et al., 2020; Rambow et al., 2004; Scerri et al., 2010), meeting transcripts have not received as much attention from the community. Closest to our work is the work of Mukherjee et al (Mukherjee et al., 2020), which focuses on the extraction and rephrasing of AIs over emails. While their goal is very similar to ours, rephrasing transcripts is very different from rephrasing emails, as we show in Section 5.

A: At the b- furthermore , I told Jerry that I was gonna finish it before he got back. So.
C: That's approaching. He's coming back when ? next -
A: I think - we think we'll see him definitely on Tuesday for the next - Or, no, wait. The meetings are on Thursday .
A: Well, we'll see him next week .
A: I was thinking about that. I think I will try to work on the SmartKom stuff and I'll if I can finish it today , ▲ I'll help ▼ you with that tomorrow.
A: if you work on it?
A: I don't have a problem with us working on it though? So.

(a) Predicted context annotations highlighted

A: At the b- furthermore , [c:0 I] told Jerry that [c:0 I] was gonna finish [c:4 it] before he got back. So.
C: That's approaching . He's coming back when ? next -
A: [c:0 I] think - we think we'll see him definitely on Tuesday for the next - Or, no, wait. The meetings are on Thursday.
A: Well, we'll [c:3 see] him next week.
A: [c:0 I] was thinking about [c:3 that]. [c:0 I] think [c:0 I] will try to work on [c:4 the SmartKom stuff] and [c:0 I]'ll if [c:0 I] can finish [c:4 it] today , ▲ [c:0 I]'ll help ▼ [c:0 you] with [c:4 that] tomorro ,
A: if [c:0 you] work on [c:4 it]?
A: [c:0 I] don't have a problem with us working on [c:4 it] though ? So.

(b) Predicted coreference annotations in square brackets (including cluster indices c:`<index>`)

Figure 1: Meeting transcript from the test set annotated by (a) context model; (b) coreference model. AI appears between solid triangles. Predicted rephrasing—**baseline:** *"Speaker A will try to work on SmartKom stuff"*; **context:** *"Speaker A will help Speaker C with SmartKom stuff"*; **coreference:** *"Speaker A will try to work on SmartKom stuff and if he can finish it he will"*; **human:** *"Speaker A will meet Jerry next week and will try to work on the SmartKom stuff"*.

Einolghozati et al. (2020) applied a pre-trained BART with a copy mechanism for the task of rephrasing virtual assistance messages. However, they are focused on style adaptation and personal pronouns modification while we focus on context-based enrichment.

Meeting summarization has been explored in the past (Oya et al., 2014; Garg et al., 2009). However, these works focus on full meeting transcript summarization, where we focus on extracting and rephrasing information specific to a given AI. In this perspective, our work can be viewed as a variation of the known *query based summarization* problem (Rahman and Borah, 2016), where given a document and a query, the algorithm goal is to extract information regarding the query from the document. While this topic has been studied in the past (Saggion et al., 2003; Bosma, 2003; Nema et al., 2017), to the best of our knowledge, there has been no work that relates to meeting transcripts, especially in the context of AIs.

3 The AIR Dataset

We create AIR **A**ction **I**tem **R**ephrasing, a new dataset focused on professional meetings AI rephrasing. The dataset is composed of instances, where each instance contains ten utterances, where the 8th utterance includes an AI. The labels are a human-produced rephrasing of the AI. The annotation process was split into three parts; acquiring raw data from multiple sources, AI extraction, and AI rephrasing. We now describe each of these steps.

3.1 Dataset Construction

Several datasets of manually transcribed records of meetings were used to accumulate action items and generate rephrasing. The result is a diverse dataset, containing a collection of different meeting types, such as software development, product design, financial, and board meetings[2].

ICSI meeting corpus (Janin et al., 2003) contains 75 meetings recorded in a conference room at the International Computer Science Institute in Berkeley.

Augmented Multi-party Interaction (AMI) meeting corpus (Carletta et al., 2006) is a multimodal dataset consisting of 100 recording hours of 154 meetings, and their manually annotated transcripts. Some of the meetings are naturally occurring, and some are elicited, particularly using a scenario in which the participants play different roles in a design team, taking a design project from kick-off to completion over the course of a day.

Board Meetings (LSC) is an open to public board meetings transcripts of the legal services corporation (LSC) and other transcribed board meetings that were extracted from available public resources.

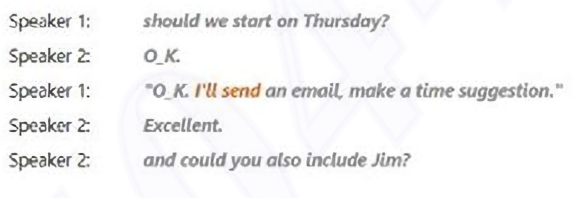

Speaker 1:	*should we start on Thursday?*
Speaker 2:	*O_K.*
Speaker 1:	*"O_K. I'll send an email, make a time suggestion."*
Speaker 2:	*Excellent.*
Speaker 2:	*and could you also include Jim?*

Figure 2: The UI used by the human annotators for AI detection.

Internal Dataset (ID) that contains internal ***[3] manually transcribed meetings which mostly revolve around software development topics.

Parts of these datasets contain sensitive information, so while datasets will be made publicly available in the future, some parts of the data cannot be shared.

Table 1 describes relevant AIR statistics. Note that to better compare the two AIR versions, the dev and test set are identical between the public and restricted versions. Both development and test sets were composed of ISCI dataset. we used ICSI for the test and development set because ISCI samples contained the highest intra-judgment score as discussed in Appendix A.

3.2 Action Items Detection

Action Items are rare in conversations and are found in roughly 1% of sentences. To reduce annotation costs, we wish to increase the percentage of AIs in the data. To this goal, the transcripts are filtered by a pre-classifier – a list of regular expressions. The pre-classifier filters out 93% of the sentences, with a precision of 17% and recall of 90% over Action Items.

AI candidates that passed the pre-classifier were labeled using human annotators. The action item annotation task was composed of five sequential utterances. The sentence that includes the AI candidate was the third sentence and two previous and following utterances were shown as context, as seen in Figure 2. Each annotator was asked if the third utterance contains an AI or not.

As preparation for the annotation task, each judge reviewed the annotation guideline and was required to successfully pass a test of ten samples before granted access to the data. The candidate sentences were tagged by five annotators. If the agreement between judges was lower than 80%, four more annotators were added. Each AI candidate was labeled as Action Item or not, according

---

[2] Each of the datasets was adjusted slightly to make relatively uniform samples. e.g. all speakers roles were converted to the form ”speaker_X, where X is a running number

[3] Organization name is omitted to preserve anonymity

Name	# of meetings	Average # of utterances	# of AIs	# of rephrasing
ID	594	374	5356	14820
ICSI	75	1428	722	2487
AMI	145	820	461	1445
Board Meetings	206	787	1283	3286
Overall	1020	598	7957	22038
Overall public	426	911	2519	7218

Table 1: AIR Source datasets statistics including the number of meetings, number of utterances per meeting, number of AIs extracted from the full dataset, and the number of rephrasing.

to a majority vote of the annotators.

3.3 Action Items Rephrasing

Action Items were rephrased by human annotators. Similar to the AI detection stage, the datasets were divided into samples that contain ten utterances, seven utterances before the AI, and two utterances after. Preliminary analysis showed an accelerated decline in context relevance when moving away from the action item, with less than 1% contribution to the seventh sentence before the AI. This justified the decision to present no more than seven pre-AI sentences. The distribution of context over the seven pre-AI and two post-AI utterances is shown in Figure 3

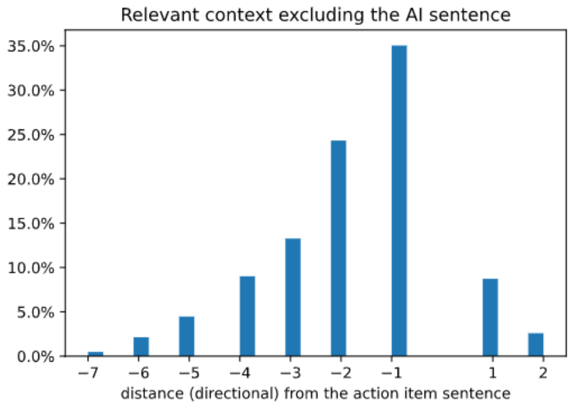

Figure 3: Percentage of the instances containing relevant context relative to the AI sentence.

Set	# of AIs	# of rephrasing
Train	2219	6318
Validation	150	450
Test	150	450

Table 2: Number of AIs and Rephrasings in train and test sets for the public dataset.

While this truncation might make some AIs unclear due to lack of relevant content, this is a good trade-off between loss of information and annotation efficiency and accuracy. Human annotators were asked to write a self-explanatory sentence in their own words, based on the Action Items and their surrounding context. In text rephrasing, like other text generation tasks, there is not a single correct answer. Therefore, each sample was rephrased up to six times[4]. Multiple rephrasing per Action Item also helped us to assess the quality of the rephrasing. The overall number of AIs and Rephrasing ,train and test sets for the public dataset are described in table 2[5]. All samples in the test and validation sets were chosen from the ISCI dataset for uniformity (in both public and private variations).

4 Model and Hints

In this section we show how to fine-tune a pre-trained model using the AIR data set. Additionally, we show how to add hints – extra annotations that improve the model ability to find relevant context, and generate more accurate and self-contained rephrasing. Hints and several versions of model rephrasing on the test set appear in Figure 1.

Base model The base model is based on BART (Lewis et al., 2020), a transformer based model (Vaswani et al., 2017), that was pre-trained by noising the input text and guiding the model to output, a de-noised version of the input. The BART model seems suitable for the rephrasing task. In this task, similar to the BERT pre-training task, the output

---

[4] Initially each sample was rephrased multiple times to check similarity of rephrasing and agreement. Later rehearsing was tagged between one to three times.

[5] Test and validation set were contracted as described in Appendix A

is a better version of the input; the ambiguity is cleaned from the AI and the output is a comprehensible and full version of the input. The de-noising ability of BART can also assist in overcoming noise of ASR systems or the speaker's partial and incoherent sentences. Our experiments in Section 6.2.2 further support this claim.

The model input is inspired by (Mukherjee et al., 2020). The input includes the sentence with AI, the previous seven sentences and the following two sentences. For each utterance in the input, we add the speaker name, and OOV markers to indicate the speaker's name and the AI, For example:

> <speaker>John</speaker><AI>I will send </AI>the presentation tomorrow.

The model is trained using teacher forcing and cross entropy loss over the predicted token.

4.1 Hints

One of the shortcomings of the base model is its limited ability to correctly identify relevant context with regard to the AI. Therefore, the rephrased AIs are often inaccurate and include irrelevant information from unrelated parts of the input text. To mitigate this, we propose two improvements to the base model.

Coreference model Given an input text, the task of *coreference resolution (CR)* aims to cluster entities that appear in different parts of the input text but refer to the same entity. This task has been thoroughly researched (Sukthanker et al., 2020; Soon et al., 2001), and in recent years gained a boost in performance, thanks to neural architectures (Xu and Choi, 2020; Joshi et al., 2020; Meged et al., 2020; Caciularu et al., 2021; Cattan et al., 2020). We hypothesize that CR models can capture semantic relations that the base model will miss because they are trained specifically for this task using vast amounts of data.

We use the CR model from Allennlp (Gardner et al., 2019), which uses SpanBERT for contextualized embedding (Joshi et al., 2020) and (Lee et al., 2017) method for CR. This model was trained on the OntoNotes dataset (Hovy et al., 2006).

Using the coreference model, we add hints to the text (see Figure 1b), identifying coreference clusters within the text. Text embedding includes cluster-marks appearing in square brackets around each of the cluster spans. For example:

> *"[c:0 Jon] works at the [c:1 cinema] , [c:0 he] loves working [c:1 there] ."*

We only mark clusters that have at least one instance in the AI utterance. This annotated text is input to the rephrasing model (both at the training phase and inference phrase). Note that training is applied only to the rephrasing model, whereas the CR model's weights are held constant.

Context detection model Another approach to improve the rephrasing model's ability to detect relevant spans is to directly train a model to detect them. To achieve this, we ask annotators to mark spans of text that are relevant to the AI. This data is used to train a *context detection model.* Spans of relevant text are transformed into binary token labels ('relevant' or 'irrelevant'). Accordingly, we train a token-classification model, based on the RoBERTa (Liu et al., 2019) pretrained transformer encoder, which is added to a fully-connected layer to perform per-token classification, given RoBERTa's output representation of each token. The model is fine-tuned end-to-end over the collected token annotations.

We found that the collected context information suffers from a low agreement between the judges. We use Kripendorff's alpha (Hayes and Krippendorff, 2007; Artstein and Poesio, 2008) coefficient to measure the judges (dis)agreement.We measure both an alpha score for the entire annotation task, as well as the pairwise agreement between the judges – both exhibiting values on the order of 0.4, which is considerably low. One reason for a low agreement may be an incomplete or incorrect understanding of the text: annotators are detached from the larger context of the meeting and from the subject matter; oral communication tends to be implicit and relies on pre-understanding; spoken sentences tend to be broken and less organized. Another reason might be the task itself, whose definition inevitably bears some level of ambiguity and arbitrariness.

Low agreement between annotators does necessarily undermine machine learning in its attempt to generalize from the train set (see Sect. 4.1.4 in (Artstein and Poesio, 2008), (Reidsma and Carletta, 2008)). It is important, however, to account for it when *measuring* classification performance on the test set. Each test instance for the context detection model consists of a transcript snippet containing a set of tokens for binary classification, and a human annotation of the tokens as relevant or irrelevant. Per test instance, we measure: (a) token

Model	Rouge-1	Rouge-2
SmartToDo (BiLSTM)	0.6	0.41
Human annotator	0.6	0.37
BART	0.63	0.43
BART (only subject & AI)	0.628	0.43
BART (No subject)	0.58	0.39
BART (AI only)	0.56	0.33

Table 3: Different variations of the email AI dataset.

ranking by the model (in accordance to the model's output scores) is evaluated by *average precision (AP)*; (b) F1-score of model's predictions. To aggregate on the test set, we have taken the mean of each value, resulting in *mean average precision (MAP)* and *mean F1-score (MF1)*.[6]

In order to assess the judges' disagreement on the test set, for each of the test instances, we collected 3 to 6 annotations. Per test instance, the optimal AP is obtained when ordering the token according to their "soft" label – mean of judges' scores (0 or 1).[7] We denote by oMAP the mean of optimal AP over all test instances. Similarly, oMF1 is defined as the mean of instances' F1-scores for judges' majority prediction per token.

The context model's *normalized MAP* score (namely, $\frac{\text{MAP}}{\text{oMAP}}$) and *normalized MF1* score (namely, $\frac{\text{MF1}}{\text{oMF1}}$) are both 0.82. Similarly to the CR model, the context detection is used to add hints to the rephrasing models' input (see Figure 1a). Training the rephrasing model designates a second usage of the corpus, now with human rephrasing annotations.[8] Despite a relatively low agreement between the judges, context annotations significantly improve the rephrasing model's performance.

5 Comparison of Emails and Transcripts

AI rephrasing was applied over emails in (Mukherjee et al., 2020). We evaluate this model over meeting transcripts and show that despite its success over emails, it is less suitable for transcripts. At a first glance, email and meeting transcript AI rephrasing might seem similar. In this section, we challenge this assumption by highlighting three key differences.

Email subject We use the dataset from (Mukherjee et al., 2020) to evaluate performance over emails compared to transcript data. Their data is based on the Avocado dataset (Douglas Oard, William Webber, David A. Kirsch, 2015), where each instance contains a pair of emails (an email with an AI and the previous email in the correspondence), and a rephrasing of the AI. To build a rephrasing model, the authors used the following approach: 1. Chose relevant sentences from each email by similarity to the AI sentence. 2. Create an input that contains the chosen sentences, the mails authors, and the mails subjects, tagging each part of the data with dedicated markers. 3. Learn a model using a BiLSTM with copy mechanism (Zeng et al., 2016).

We test a number of variations based on this approach. 1. We replace the BiLSTM with BART (BART); 2. Similar to (1), but we omit the email's subject from the input (No subject); 3. Similar to (1), but remove all the email's body text besides the AI sentence (only subject & AI); 4. Leaving only the AI sentence (AI only). The results are presented in Table 3. The BART base model unsurprisingly outperforms the BiLSTM model. But surprisingly, a model that is exposed to the Subject alone performs almost as well as the full model. Additionally, most of the improvement compared to the base (AI only) model comes from adding the email's subject. This means that the body of the emails play a minimal role in the email's rephrasing. Unfortunately, meeting transcripts do not have an equivalent to a subject, which forces the model to rely solely on the transcript text.

Number of utterances While two emails often supply enough context for rephrasing, transcript samples contains ten utterances, which require the model to "find" the right spans from a relatively large pool of text.

Malformed language In contrast to written text, spoken language is less formal. This results in people making grammatical mistakes like stopping at the middle of a sentence ("I will take the... yes, that's right" or repeating words ("I... ah... I... think that it's OK"). This new grammar is very different from the pretrained text most pretrained models were trained on.

---

[6] Due to the nature of the problem, there are considerably more negatives than positives, and thus we choose two metrics that embody the precision/recall tradeoff, which is indifferent to true negative predictions, rather than the true-positive/true-negative tradeoff.

[7] AP calculation takes token-level steps, updating precision and recall according to *all* judges' annotations at once.

[8] In the training phase, as in the inference phase, the context model's predictions are used, rather than the judges' annotations.

Model	ROUGE-1	ROUGE-2	BLEU
Human	0.519	0.305	0.737
Base model	**0.645**	**0.459**	**0.763**

Table 4: ROUGE and BLEU score of the base model and human rephrasing.

6 Experiments

We evaluate our various models compared to a human annotator, and between themselves, in two ways. The first is automatic n-gram based metrics and the second is direct human evaluation. All models were trained using the Huggingface framework(Wolf et al., 2019) with the following configuration: **batch size:** 8, **lr:** $3 \cdot 10^{-5}$, **epocs:** 5 (20 with public dataset).

6.1 N-gram Based Metrics

There are a variety of n-gram based methods that evaluate the quality of text generation tasks (Lin, 2004; Banerjee and Lavie, 2005; Papineni et al., 2001). In this work, we use the ROUGE-1 and ROUGE-2 (Lin, 2004) as it is a widely used measure for summarization evaluation.

We evaluate the base model vs. a gold rephrasing produced by a human annotator. We also include another human rephrasing that represents human evaluation level. The results are shown in Table 4. Even the base model suppresses the human-produced rephrasing and achieves a higher ROUGE score. While this result looks impressive, we claim it simply highlights the limitation of ROUGE, and other n-gram based metrics to evaluate the rephrasing quality (see (Mathur et al., 2020)). We provide the following rational:

Frequent words Not all words have the same value when evaluating the quality of a sentence. For example, entity mentions and verbs are more relevant to the sentence meaning than stop words, but n-grams models give each word an equal weight for the overall score. While human annotators are good at finding rare words that convey meaning, automatic models are good at using very frequent words. Using these words increase the model's score, while not contributing to the quality of the rephrasing.

Lacking of "true" gold samples Typically, machine-learning algorithms are evaluated based on a gold standard that is generated by humans. It is also typical for ground truth to have some error rate, and there are techniques to reduce the probability of error, e.g., taking a majority vote. In contrast, in the case of text generation, there are usually many results (sentences) that can be considered 'good', having very different wording. BLEU (Papineni et al., 2001) addresses this difficulty by comparing the generated text to a number of different human-generated labels. While this somewhat reduces the chance that a good text generation will get a low score, it does not eliminate it.

Models might outperform humans as we show in Section 6.2.1, our models might achieve higher quality than the human-produced rephrasing. N-gram based models treat each deviation from the human model as an error, although it might achieve a better paraphrasing[9].

For these reasons, we turn to human evaluation as the main models' evaluation method.

6.2 Human Evaluation

We evaluate our models by using the following procedure - Each instance contains the model input (transcript + speakers + AI) and three[10] different rephrasings. The judges were instructed to assign each rephrasing a score from the set {1,2,3}, while considering these questions (arranged by importance):

1. Does the action described in the rephrasing accurately describe the action in the context + AI?
2. Does the rephrasing contain all the details that are described in the context + AI?
3. Is the rephrasing grammatically correct?
4. How easy was it to understand the rephrasing?

The judges UI is shown in Figure 5 on Appendix B.

6.2.1 Main Results

Using the AIR-internal dataset, we compare the three models with each other and with a human-produced rephrasing. We applied two evaluations. First, we compare the human-produced rephrasing, the base model, and the coreference model. The

---

[9] This indicates that the automatic metrics might underestimate automatic models.

[10] We use three models per instance (instead of four) to reduce the load from the judges.

Models	Result
Human/Coref	**56.0% / 44.0%**
Human/Base	**68.2% / 31.8%**
Base/Coref	**39.4% / 60.6%**
Context/Coref	**59.1% / 40.9%**
Human/Context	50.9% / 49.1%

Table 5: Comparison of the base, coreference, and context models. Bold results are significant.

Models	Result
Base/Coref	**43.4% / 56.6%**
Context/Coref	50.5% / 49.5%
Context/Base	**58.4% / 41.6%**

Table 6: Unilm comparison.

Models	Result
human/BART	50.8% / 49.2%
human/Unilm	53.1% / 46.9%
BART/Unilm	53.0% / 47.0%

Table 7: Cross LM comparison. Both models used the context annotations.

Models	Result
Human/Coref	56.1% / 43.9%
Human/Base	**68.2% / 31.8%**
Base/Coref	**40.1% / 59.9%**
Context/Coref	46.1% / 53.9%
Human/Context	54.3% / 45.6%

Table 8: Result on the public dataset. Using the base, coreference, and context models. Bold results are significant.

second comparison compares the human rephrasing, coreference-based, and context-based rephrasing. The results are presented in Table 5. The Human rephrasing outperforms all models, while the context achieves the closest performance, followed by the coreference and the base models. All results are statistically significant, except the context and human comparison. We used double evaluation of the human and coreference model rephrasing to calculate the judge's agreement, which resulted in a Kappa value of 0.604. Both the coreference and context model achieve close to human performance when the context-based model is almost equal to the human rephrasing.

6.2.2 Language Model Comperison

We choose BART as the base language model (LM) for our work because its denoising pre-train objective function seems suitable to deal with the noisy transcripts data. To evaluate this, we compare our results to another model - unilm (Bao et al., 2020). This model was trained as both masked LM and autoregressive LM and showed improvement on a variety of downstream tasks. To compare BART and unilm, we first compare our three candidate models using each of the LM, and then compare the best models with each LM. The results of the unilm variations are presented in Table 6. The hierarchy between the model variations remains, while the differences between models are smaller and become statistically insignificant. Nevertheless, we chose the context model as the best model for the cross LM comparison. The results are presented in Table 7. BART outperforms Unilm both on direct comparison and in its performance vs. the human rephrasing.

6.2.3 Public Dataset Results

The ID dataset contains sensitive and personal data and cannot be released to the public. In this subsection, we ran the same comparison of base/coreference/context and human annotators using the AIR-public dataset. These results can also highlight the dependency of the algorithm performance on the dataset size. The results are presented in Table 8. The human rephrasing is still the best model compared to all the others, but surprisingly it does not have a bigger gap in performance compared to the full dataset. We explain this by the mix of the different datasets. Each of the datasets contains a different distribution of AIs. Removing the ID dataset allows the model to overfit to the remaining existing datasets. The context model suffered the most from the reduced size of the dataset and is outperformed by the coreference model. We attribute this to the fact that the model trains twice on the data set (first for context detection, then for rephrasing). This shows that the coreference hints are best utilized when the dataset size is small.

7 Conclusion

In this work, we present the problem of action items rephrasing in meeting transcripts. We introduce a new dataset for the task and establish a baseline attributed to the BART transformer model. We then present two novel ways to considerably improve the baseline. Particularly, by collecting context annotations, and despite a relatively low agreement between the annotators, we are able to considerably improve the rephrasing model's performance. We evaluate our work by automated metrics, as well as independent human evaluators.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao Wuen Hon. 2020. UNILMv2: Pseudo-masked language models for unified language model pre-training.

Paul N. Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *SIGIR 2005 - Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Paul N. Bennett and Jaime G. Carbonell. 2007. Combining probability-based rankers for action-item detection. In *NAACL HLT 2007 - Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*.

Wauter Bosma. 2003. Query-based Summarization using Rhetorical Structure Theory. In *Proc. of CLIN 2003*.

Avi Caciularu, Arman Cohan, Iz Beltagy, Matthew E. Peters, Arie Cattan, and Ido Dagan. 2021. Cross-Document Language Modeling.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, Lain McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. 2006. The AMI Meeting Corpus: A pre-announcement. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Arie Cattan, Alon Eirew, Gabriel Stanovsky, Mandar Joshi, and Ido Dagan. 2020. Streamlining cross-document coreference resolution: Evaluation and modeling.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*.

Sergey Golitsynskiy Douglas Oard, William Webber, David A. Kirsch. 2015. Avocado Research Email Collection.

Arash Einolghozati, Anchit Gupta, Keith Diedrick, and Sonal Gupta. 2020. Sound Natural: Content Rephrasing in Dialog Systems.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2019. AllenNLP: A Deep Semantic Natural Language Processing Platform.

Nikhil Garg, Benoit Favre, Korbinian Reidhammer, and Dilek Hakkani-Tür. 2009. ClusterRank: A graph based method for meeting summarization. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*.

Andrew F Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Human Language Technology Conference of the NAACL, Short Papers*.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The ICSI meeting corpus. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 188–197. Association for Computational Linguistics (ACL).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.

C Y Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Proceedings of the workshop on text summarization branches out (WAS 2004)*.

Chu Cheng Lin, Michael Gamon, Dongyeop Kang, and Patrick Pantel. 2018. Actionable email intent modeling with reparametrized rnNs †. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

LSC. LSC - Legal Services Corporation: America's Partner for Equal Justice.

Nitika Mathur, Tim Baldwin, and Trevor Cohn. 2020. Tangled up in bleu: Reevaluating the evaluation of automatic machine translation evaluation metrics. *arXiv preprint arXiv:2006.06264*.

Yehudit Meged, Avi Caciularu, Vered Shwartz, and Ido Dagan. 2020. Paraphrasing vs coreferring: Two sides of the same coin.

Sudipto Mukherjee, Subhabrata Mukherjee, Marcello Hasegawa, Ahmed Hassan Awadallah, and Ryen White. 2020. Smart to-do: Automatic generation of to-do items from emails. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8680–8689, Online. Association for Computational Linguistics.

Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *INLG 2014 - Proceedings of the 8th International Natural Language Generation Conference, including - Proceedings of the INLG and SIGDIAL 2014 Joint Session*, pages 45–53. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. *ACL*.

Nazreena Rahman and Bhogeswar Borah. 2016. A survey on existing extractive techniques for query-based text summarization. In *2015 International Symposium on Advanced Computing and Communication, ISACC 2015*, pages 98–102. Institute of Electrical and Electronics Engineers Inc.

Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads.

Dennis Reidsma and Jean Carletta. 2008. Reliability measurement without limits. *Computational Linguistics*, 34(3):319–326.

Horacio Saggion, Kalina Bontcheva, and Hamish Cunningham. 2003. Robust generic and query-based summarisation. page 235.

Simon Scerri, Gerhard Gossen, Brian Davis, and Siegfried Handschuh. 2010. Classifying action items for semantic email. In *Proceedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010*.

Wee Meng Soon, Daniel Chung Yong Lim, and Hwee Tou Ng. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*.

Rhea Sukthanker, Soujanya Poria, Erik Cambria, and Ramkumar Thirunavukarasu. 2020. Anaphora and coreference resolution: A review. *Information Fusion*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing.

Liyan Xu and Jinho D. Choi. 2020. Revealing the myth of higher-order inference in coreference resolution.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient Summarization with Read-Again and Copy Mechanism.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

A Test and Validation Sets Construction

Before evaluating the full test set, we ran a small subset of tests (60 instances overall) of the test set to get initial results. We compared the human rephrasing to the base model.The results are surprising, where most base models scored higher than the human rephrasing. While these results are attractive, manual examination showed that the base model often made mistakes that a human annotator could handle easily. Additionally, we checked the agreement between the judges on the same instances using Cohen's kappa (Cohen, 1960). The agreement was unexpectedly low 0.39. We explain these two phenomena by offering the hypothesis - some samples are very hard / impossible to rephrase properly. We suggest three reasons:

Context not in the sample In these samples the required context is not found in the part of the transcript that is included in the sample (each sample includes seven sentences before the AI, and three after).

Context requires real-world knowledge In some samples, some real-world knowledge is required to understand the AI. For example, our internal dataset is composed of technical team meetings that require prior knowledge in programming, software engineering, and NLP. Our annotators occasionally lack the technical knowledge required to understand the full AI meaning.

Rephrasing is hard Some samples were very hard to rephrase, even for humans, and often require a number of passes over the text to rephrase properly. Even though all of our annotators are proficient in English, they still had a very hard time rephrasing part of the questions. This results in bad rephrasing.

Surprisingly, when both the model and human fail, the human judge tends to prefer the model output. We attribute this to the fact that the model output was always grammatically and semantically correct, while the human had a tendency to write malformed sentences when the rephrasing was unclear.

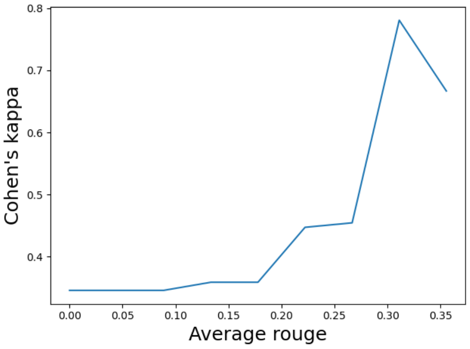

Figure 4: Cohen's Kappa as a function of a threshold serving as lower bound on the samples' average rouge.

Rephraseable instances In cases where human annotators produce relatively similar rephrasing, we can presume the model will do the same, and thus we can think of those samples as 'easy to rephrase', or *rephraseable*. In order to find these, we take the *average rouge score* between all pairs of human rephrasing, as follows:

$$score(s) = \frac{1}{|R_s|^2} \sum_{\langle r,r' \rangle \in R_s} rouge_2(r, r')\,, \qquad (1)$$

where s is a input sample, R_s is the set of all human rephrases of s, and $rouge_2(\cdot,\cdot)$ is the Rouge-2 measure. As *rephraseables*, we consider half of the samples – those with higher average rouge per (1). Figure 4 shows that, as one may expect, by considering the more rephraseable samples (i.e., putting a lower bound on the samples' average rouge score), the agreement between the judges on the related task of marking relevant context increases.

We use these insights to construct the test set and validation set by randomly sampling instances with an average rouge score ≥ 0.3. In Section 6, we show that using this test set, the judges' agreement is significantly higher (0.61) on Cohen's Kappa.

B Rephrasing Evaluation UI

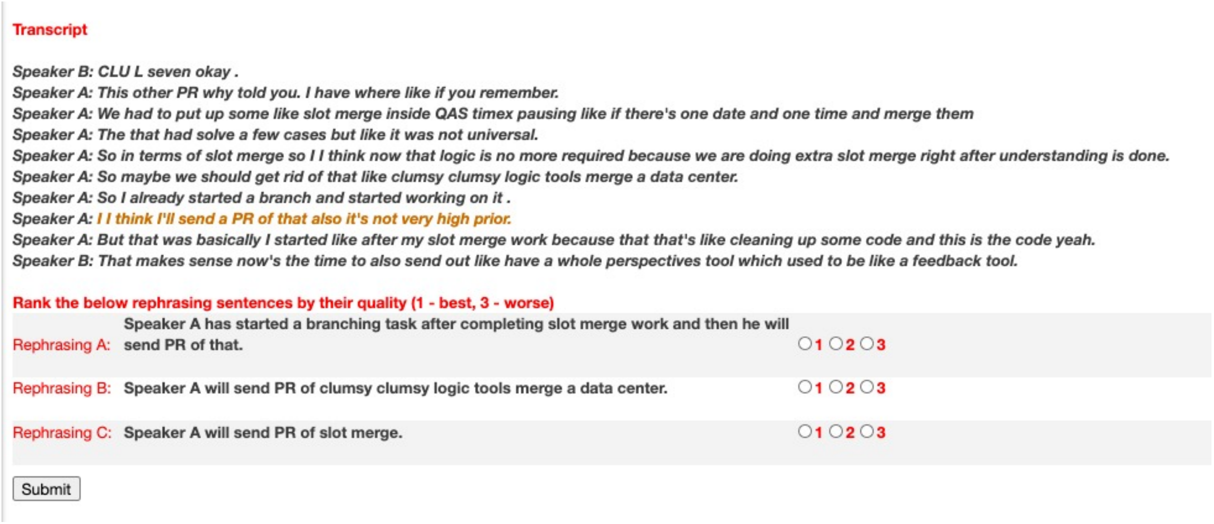

Transcript

Speaker B: CLU L seven okay .
Speaker A: This other PR why told you. I have where like if you remember.
Speaker A: We had to put up some like slot merge inside QAS timex pausing like if there's one date and one time and merge them
Speaker A: The that had solve a few cases but like it was not universal.
Speaker A: So in terms of slot merge so I I think now that logic is no more required because we are doing extra slot merge right after understanding is done.
Speaker A: So maybe we should get rid of that like clumsy clumsy logic tools merge a data center.
Speaker A: So I already started a branch and started working on it .
Speaker A: I I think I'll send a PR of that also it's not very high prior.
Speaker A: But that was basically I started like after my slot merge work because that that's like cleaning up some code and this is the code yeah.
Speaker B: That makes sense now's the time to also send out like have a whole perspectives tool which used to be like a feedback tool.

Rank the below rephrasing sentences by their quality (1 - best, 3 - worse)

Rephrasing A:	Speaker A has started a branching task after completing slot merge work and then he will send PR of that.	○1 ○2 ○3
Rephrasing B:	Speaker A will send PR of clumsy clumsy logic tools merge a data center.	○1 ○2 ○3
Rephrasing C:	Speaker A will send PR of slot merge.	○1 ○2 ○3

Figure 5: The UI used by human judges to evaluate rephrasing. The displayed models are (top to bottom) human rephrasing, base model, and coreference model.

MERGEDISTILL: Merging Pre-trained Language Models using Distillation

Simran Khanuja
Google Research
India
simrankh@google.com

Melvin Johnson
Google Research
USA
melvinp@google.com

Partha Talukdar
Google Research
India
partha@google.com

Abstract

Pre-trained multilingual language models (LMs) have achieved state-of-the-art results in cross-lingual transfer, but they often lead to an inequitable representation of languages due to limited capacity, skewed pre-training data, and sub-optimal vocabularies. This has prompted the creation of an ever-growing pre-trained model universe, where each model is trained on large amounts of language or domain specific data with a carefully curated, linguistically informed vocabulary. However, doing so brings us back full circle and prevents one from leveraging the benefits of multilinguality. To address the gaps at both ends of the spectrum, we propose MERGEDISTILL, a framework to merge pre-trained LMs in a way that can best leverage their assets with minimal dependencies, using *task-agnostic* knowledge distillation. We demonstrate the applicability of our framework in a practical setting by leveraging pre-existing teacher LMs and training student LMs that perform competitively with or even outperform teacher LMs trained on several orders of magnitude more data and with a fixed model capacity. We also highlight the importance of teacher selection and its impact on student model performance.

1 Introduction

While current state-of-the-art multilingual language models (LMs) (Devlin et al., 2019; Conneau et al., 2020) aim to represent 100+ languages in a single model, efforts towards building monolingual (Martin et al., 2019; Kuratov and Arkhipov, 2019) or language-family based (Khanuja et al., 2021) models are only increasing with time (Rust et al., 2020). A single model is often incapable of effectively representing a diverse set of languages, evidence of which has been provided by works highlighting the importance of vocabulary curation and size (Chung et al., 2020; Artetxe et al., 2020), pre-training data volume (Liu et al., 2019a; Conneau et al., 2020), and the curse of multilinguality (Conneau et al., 2020). Language specific models alleviate these issues with a custom vocabulary which captures language subtleties[1] and large magnitudes of pre-training data scraped from several domains (Virtanen et al., 2019; Antoun et al., 2020). However, building language specific LMs brings us back to where we started, preventing us from leveraging the benefits of multilinguality like zero-shot task transfer (Hu et al., 2020), positive transfer between related languages (Pires et al., 2019; Lauscher et al., 2020) and an ability to handle code-mixed text (Pires et al., 2019; Tsai et al., 2019). We need an approach that encompasses the best of both worlds, i.e., leverage the capabilities of the powerful language-specific LMs while still being multilingual and enabling positive language trans-

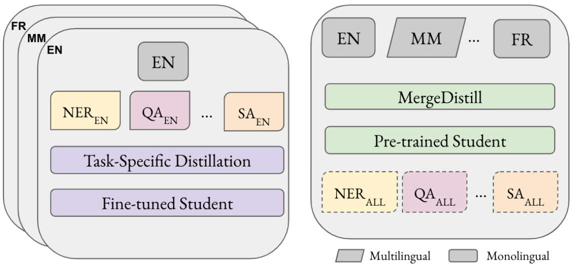

Figure 1: Previous works (left) typically focus on combining *fine-tuned* models derived from a single pre-trained model using distillation. We propose MERGEDISTILL to combine *pre-trained* teacher LMs from multiple monolingual/multilingual LMs into a single *multilingual task-agnostic* student LM.

---

[1] For example, in Arabic, (Antoun et al., 2020) argue that while the definite article "Al", which is equivalent to "the" in English, is always prefixed to other words, it is not an intrinsic part of that word. While with a BERT-compatible tokenization tokens will appear twice, once with "Al-" and once without it, AraBERT first segments the words using Farasa (Abdelali et al., 2016) and then learns the vocabulary, thereby alleviating the problem.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2874–2887
August 1–6, 2021. ©2021 Association for Computational Linguistics

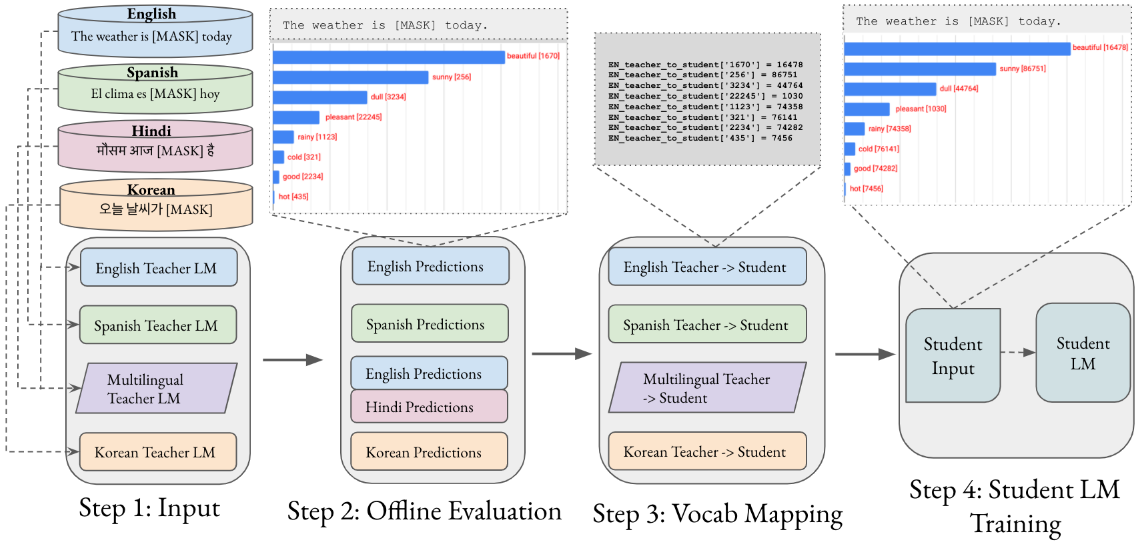

```
EN_teacher_to_student['1670'] = 16478
EN_teacher_to_student['256'] = 86751
EN_teacher_to_student['3234'] = 44764
EN_teacher_to_student['22245'] = 1030
EN_teacher_to_student['1123'] = 74358
EN_teacher_to_student['321'] = 76141
EN_teacher_to_student['2234'] = 74282
EN_teacher_to_student['435'] = 7456
```

Figure 2: *Overview of* MERGEDISTILL: The input to MERGEDISTILL is a set of pre-trained teacher LMs and pre-training transfer corpora for all the languages we wish to train our student LM on. Here, we combine four teacher LMs comprising of three monolingual (trained on English, Spanish and Korean respectively) and one multilingual LM (trained on English and Hindi). The student LM is trained on English, Spanish, Hindi and Korean. Pre-training transfer corpora for each language is tokenized and masked using their respective teacher LMs vocabulary. We then obtain predictions for each masked word in each language, by evaluating *all* of their respective teacher LMs. For example, we evaluate English masked examples on both the monolingual and multilingual LM as shown. The student's vocabulary is a union of *all* teacher vocabularies. Hence, the *input, prediction and label indices* obtained from teacher evaluation are now mapped to the student vocabulary, and input to the student LM for training. Please refer to Section 3.1 for details.

fer.

In this paper, we use knowledge distillation (KD) (Hinton et al., 2015) to achieve this. In the context of language modeling, KD methods can be broadly classified into two categories: task-specific and task-agnostic. In task-specific distillation, the teacher LM is first fine-tuned for a specific task and is then distilled into a student model which can solve that task. Task-agnostic methods perform distillation on the pre-training objective like masked language modeling (MLM) in order to obtain a task-agnostic student model. Prior work has either used task-agnostic distillation to compress *single*-language teachers (Sanh et al., 2019; Sun et al., 2020) or used *task-specific* distillation to combine multiple fine-tuned teachers into a multi-task student (Liu et al., 2019b; Clark et al., 2019). The former prevents positive language transfer while the latter restricts the student's capabilities to the tasks and languages in the fine-tuned teacher LMs (as shown in Figure 1).

We focus on the problem of merging *multiple* pre-trained LMs into a single multilingual student LM in the *task-agnostic* setting. To the best of our knowledge, this is the first effort of its kind, and makes the following contributions:

- We propose MERGEDISTILL, a task-agnostic distillation approach to merge *multiple* teacher LMs at the *pre-training* stage, to train a strong multilingual student LM that can then be fine-tuned for *any* task on all languages in the student LM. Our approach is more maintainable (fewer models), compute efficient and teacher-architecture agnostic (since we obtain offline predictions).

- We use MERGEDISTILL to **i)** combine *monolingual* teacher LMs into a single *multilingual* student LM that is competitive with or outperforms individual teachers, **ii)** combine *multilingual* teacher LMs, such that the overlapping languages can learn from *multiple* teachers.

- Through extensive experiments and analysis, we study the importance of typological similarity in building multilingual models, and the impact of strong teacher LM vocabularies and predictions in our framework.

2 Related Work

Language Model pre-training has evolved from learning pre-trained word embeddings (Mikolov et al., 2013) to contextualized word representations (McCann et al., 2017; Peters et al., 2018; Eriguchi et al., 2018) and to the most recent Transformer-based (Vaswani et al., 2017) LMs (Devlin et al., 2019; Liu et al., 2019a) with state-of-the-art results on various downstream NLP tasks. Most commonly, these LMs are pre-trained with the MLM objective (Taylor, 1953) on large unsupervised corpora and then fine-tuned on labeled data for the task at hand. Concurrently, multilingual LMs (Lample and Conneau, 2019; Siddhant et al., 2020; Conneau et al., 2020; Chung et al., 2021), trained on massive amounts of multilingual data, have surpassed cross-lingual word embedding spaces (Glavaš et al., 2019; Ruder et al., 2019) to achieve state-of-the-art in cross-lingual transfer. While Pires et al. (2019); Wu and Dredze (2019) highlight their cross-lingual ability, several limitations have been studied. Conneau et al. (2020) highlight the curse of multilinguality. Hu et al. (2020) highlight that even the best multilingual models do not yield satisfactory transfer performance on the XTREME bechmark covering 9 tasks and 40 languages. Importantly, Wu and Dredze (2020) and Lauscher et al. (2020) observe that these models significantly under-perform for low-resource languages as representation of these languages in the vocabulary and pre-training corpora are severely limited.

Language-specific LMs are becoming increasingly popular as issues with multilingual language models persist. As language identification systems are extended to 1000+ languages (Caswell et al., 2020), increasing capacity for a single model to uniformly represent all languages is prohibitive. Often, practitioners prefer to have a model performing well on a subset of languages that their application calls for. To address this, the community continues its efforts in building strong multi-domain language models using linguistic expertise. A few examples of these are AraBERT (Antoun et al., 2020), CamemBERT (Martin et al., 2020), and FinBERT (Virtanen et al., 2019).[2]

Knowledge Distillation in pre-trained LMs has most commonly been used for task-specific model compression of a teacher into a single-task student (Tang et al., 2019; Kaliamoorthi et al., 2021). This has been extended to perform task-specific distillation of multiple single-task teachers into one multi-task student (Clark et al., 2019; Liu et al., 2020; Turc et al., 2019). In the task-agnostic scenario, prior work has focused on distilling a single large teacher model into a student model leveraging teacher predictions (Sanh et al., 2019) or internal teacher representations (Sun et al., 2020, 2019; Wang et al., 2020) with the goal of model compression. To the best of our knowledge, this is the first attempt to perform task-agnostic distillation from *multiple teachers* into a *single task-agnostic student*. In the context of neural machine translation, Tan et al. (2019) come close to our work where they attempt to combine multiple single language-pair teacher models to train a multilingual student. However, our work differs from theirs in three key aspects: 1) our students are task-agnostic while theirs are task-specific, 2) we can leverage pre-existing teachers while they cannot, and 3) we support teachers with overlapping sets of languages while they only consider single language-pairs teachers.

3 MERGEDISTILL

Notations: Let K denote the set of languages we train our student LM on and T denote the set of teacher LMs input to MERGEDISTILL[3]. Consequently, $\mathrm{T_k}$ denotes the set of teacher LMs trained on language k, where $|\mathrm{T_k}| \geq 1 \; \forall \; \mathrm{k} \in \mathrm{K}$.

3.1 Workflow

An overview of MERGEDISTILL is presented in Figure 2. Here we detail each step involved in training the student LM from multiple teacher LMs.

Step 1: Input

The input to MERGEDISTILL is a set of pre-trained teacher LMs and pre-training transfer corpora for all the languages we wish to train our student LM on. With reference to Figure 2, the student LM is trained on $\mathrm{K} = \{$English (en), Spanish (es), Hindi (hi), Korean (ko)$\}$. We combine four teacher LMs comprising of three monolingual and one multilingual LM. The monolingual LMs are trained on English ($\mathrm{M_{en}}$), Spanish ($\mathrm{M_{es}}$), and Korean ($\mathrm{M_{ko}}$) while the multilingual LM is trained on English

---

[2] (Nozza et al., 2020) maintain an ever-growing list of BERT models here

[3] Note that T can comprise of monolingual or multilingual models

and Hindi ($\mathrm{M_{en,hi}}$). Therefore, for each language, the corresponding set of teacher LMs ($\mathrm{T_k}$) can be defined as: [$\mathrm{T_{en}} = \{\mathrm{M_{en}}, \mathrm{M_{en,hi}}\}, \mathrm{T_{es}} = \{\mathrm{M_{es}}\}$, $\mathrm{T_{hi}} = \{\mathrm{M_{en,hi}}\}, \mathrm{T_{ko}} = \{\mathrm{M_{ko}}\}$]. First, the pretraining transfer corpora is tokenized and masked for each language using their respective teacher LM's tokenizer. For the language with two teachers, English, we tokenize each example using both the teacher LMs.

Step 2: Offline Teacher LM Evaluation

We now obtain predictions and logits for each masked, tokenized example in each language, by evaluating their respective teacher LMs. For English, we obtain predictions from both $\mathrm{M_{en}}$ and $\mathrm{M_{en,hi}}$ on their respective copies of each training example. In an ideal situation, we believe that multiple strong teachers can present a multi-view generalisation to the student as each teacher learns different features in training. Let x denote a sequence of tokens where $\mathrm{x_m} = \{\mathrm{x_1, x_2, x_3 ... x_n}\}$ denote the masked tokens, and $\mathrm{x_{-m}}$ denote the non-masked tokens. Let v be the vocabulary of student LM θ_s. In the conventional case of learning from gold labels, we minimize the cross-entropy of student logit distribution for a masked word $\mathrm{x_{m_i}}$, with the *one-hot label* $\mathrm{v_j}$, given by:

$$\mathrm{P(x_{m_i}, v_j)} = \mathbf{1}(\mathrm{x_{m_i} = v_j}) \times \log \mathrm{p(x_{m_i} = v_j | x_{-m}; \theta_s)} \tag{1}$$

With the teacher evaluations, we obtain predictions (and corresponding logits) of the teacher for the masked tokens. Let us denote the teacher output probability distribution (softmax over logits) for token $\mathrm{x_{m_i}}$ by $\mathrm{Q(x_{m_i} | x_{-m}; \theta_t)}$. Therefore, in addition to the loss from gold labels, we minimize the entropy between the student logits and the *teacher distribution*, given by :

$$\begin{aligned}\hat{\mathrm{P}}(\mathrm{x_{m_i}, v_j}) = \mathrm{Q(x_{m_i} = v_j | x_{-m}; \theta_t)} \times \\ \log \mathrm{p(x_{m_i} = v_j | x_{-m}; \theta_s)}\end{aligned} \tag{2}$$

It is extremely burdensome (both memory and time) to load multiple teacher LMs and obtain predictions during training. Hence, we first store the *top-k* logits for each masked word offline, loading and normalizing them during student LM training, similar to (Tan et al., 2019). Additionally, obtaining offline predictions gives one the freedom to use expensive teacher LMs without increasing the student model training costs and makes our framework teacher-architecture agnostic.

Step 3: Vocab Mapping

A deterrent in attempting to distill from multiple pre-trained teacher LMs is that each LM has its own vocabulary. This makes it non-trivial to uniformly process an input example for consumption by both the teacher and student LMs. Our student model's vocabulary is the union of *all* teacher LM vocabularies. In the vocab mapping step, the *input indices*, *prediction indices*, and the *gold label indices*, obtained after evaluation from each teacher LM are processed using a teacher→student vocab map. This converts each teacher token index to its corresponding student token index, ready for consumption by the student model. For simplicity, each teacher and student LM uses WordPiece tokenization (Schuster and Nakajima, 2012; Wu et al., 2016) in all our experiments.

Step 4: Student LM Training

The processed *input indices*, *prediction indices*, and *gold label indices* can now be used to train the multilingual student LM. In training, examples from different languages are shuffled together, even within a batch. We train the student LM with the MLM objective. Let $\mathrm{L_{MLM}}$ denote the MLM loss from gold labels. Therefore, with reference to Equation 1 :

$$\mathrm{L_{MLM}(x_m | x_{-m})} = -\frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{j=1}}^{|\mathrm{v}|} \mathrm{P(x_{m_i}, v_j)}$$

In addition to learning from gold labels, we use teacher predictions as soft labels and minimize the cross entropy between student and teacher distributions. Let $\mathrm{L_{KD}}$ denote the KD loss from a single teacher LM. With reference to Equation 2:

$$\mathrm{L_{KD}(x_m | x_{-m})} = -\frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{j=1}}^{|\mathrm{v}|} \hat{\mathrm{P}}(\mathrm{x_{m_i}, v_j});$$

The total loss across all languages is minimized, as shown below:

$$\mathrm{L_{ALL}} = \sum_{\mathrm{k=1}}^{\mathrm{K}} \lambda(\mathrm{L_{KD}^{T_k}}) + (1 - \lambda)\mathrm{L_{MLM}^{k}}$$

In the case of multiple teacher LMs, we have n tokenized instances for a given example (where n denotes the number of teachers for a particular language). In this case, each example in English has two copies – one tokenized using $\mathrm{M_{en}}$ and another using $\mathrm{M_{en,hi}}$. Thus, we explore two possibilities of training in this multi-teacher scenario :

- Include all the copies in training. Here the model is exposed to n different teacher LM predictions, each presenting a multi-view generalisation to the student LM.
- Include the best copy in training. The best copy is the one having minimum teacher LM loss for a given example. Here the model is only exposed to the best teacher LM predictions for each example.

4 Experiments

In this section, we aim to answer the following questions :

1) How effective is MergeDistill in combining *monolingual* teacher LMs, to train a *multilingual* student LM that leverages the benefits of multilinguality while performing competitively with individual teacher LMs? (Section 4.2)

2) How effective is MergeDistill in combining *multilingual* teacher LMs, trained on an overlapping set of languages, such that each language can benefit from *multiple* teachers? (Section 4.3)

3) How important are the teacher LM vocabulary and predictions in MergeDistill? Further, can MergeDistill enable pre-trained zero-shot transfer? (Section 4.4)

4.1 Setup

Data: For all our experiments, we use Wikipedia data as pre-training transfer corpora to train the student model, irrespective of the data used in training individual teacher LMs. We use $\alpha = 0.7$ for exponential smoothing of data across languages, similar to mBERT (Devlin et al., 2019).

Model Size: Since transformer-based models perform better as capacity increases (Conneau et al., 2020; Arivazhagan et al., 2019), we keep the number of parameters close to mBERT ($\sim$178M) by appropriately modifying the vocabulary embedding size (like Lan et al. (2019)) to isolate the positive effects of learning from teacher LMs.

Student	Language	Language Family	Model
$\mathrm{Student_{similar}}$	English	Indo-European	BERT(Devlin et al., 2019)
	German	Indo-European	DeepSet(Chan et al., 2020)
	Italian	Indo-European	ItalianBERT(Schweter, 2020b)
	Spanish	Indo-European	BETO(Cañete et al., 2020)
$\mathrm{Student_{dissimilar}}$	Arabic	Afroasiatic	AraBERT(Antoun et al., 2020)
	English	Indo-European	BERT(Devlin et al., 2019)
	Finnish	Uralic	FinBERT(Virtanen et al., 2019)
	Turkish	Turkic	BERTurk(Schweter, 2020a)
	Chinese	Sino-Tibetan	ChineseBERT(Devlin et al., 2019)

Table 1: *Monolingual BERT Models* used as teacher LMs. Please refer to Section 4.2 for details.

Distillation Parameters: We have two hyperparameter choices here: 1) k in top-k logits - as it increases, we observe that while performances remain similar, storing k>8 number of predictions for each masked word offline significantly increases resource requirements[4]. Hence, we set k=8 in all our experiments. 2) the value of λ in the loss function, which decides the proportion of teacher loss, is annealed through training similar to Clark et al. (2019).

Evaluation Metrics: We report F1 scores for structured prediction tasks (NER, POS), accuracy (Acc.) scores for sentence classification tasks (XNLI, PAWS-X), and F1/Exact Match (F1/EM) scores for question answering tasks (XQuAD, MLQA, TyDiQA). We also report a task-specific *relative deviation from teachers* (**RDT**) (in %) averaged across all languages (n). For each task, **RDT** is calculated as:

$$\mathrm{RDT}(\mathrm{S}, \{\mathrm{T}_1, ..., \mathrm{T}_\mathrm{n}\}) = \frac{100}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{(\mathrm{P}_{\mathrm{T}_\mathrm{i}} - \mathrm{P}_\mathrm{S})}{\mathrm{P}_{\mathrm{T}_\mathrm{i}}} \tag{3}$$

where $\mathrm{P}_{\mathrm{T}_\mathrm{i}}$ and P_S are performances of the i^{th} teacher and student LMs, respectively.

4.2 Monolingual Teacher LMs

Pre-training: In this experiment, we use pre-existing monolingual teacher LMs, as shown in Table 1, to train a multilingual student LM on the union of all teacher languages. In this setup, $|\mathrm{T}_\mathrm{k}| = 1 \ \forall \ \mathrm{k} \in \mathrm{K}$, i.e., each language can learn from its respective monolingual teacher LM only.

Our teacher selection and setup follows a two-step process. First, we aim to select languages having pre-trained monolingual LMs available, and evaluation sets across a number of downstream tasks. This makes us choose teacher LMs for : Arabic (*ar*), Chinese (*zh*), English (*en*), Finnish (*fi*),

---

[4] More details in Appendix A.4

Language	Model	NER	UDPOS	QA
		F1	**F1**	**F1/EM**
English	BERT	89.5	96.6	87.1/78.6
	Student$_{\text{similar}}$	89.8	96.3	89.8/82.1
German	DeepsetBERT	93.0	98.3	-
	Student$_{\text{similar}}$	93.9	98.3	-
Italian	ItalianBERT	94.5	98.6	73.5/61.6
	Student$_{\text{similar}}$	95.2	98.6	75.8/63.8
Spanish	BETO	94.2	99.0	74.9/56.6
	Student$_{\text{similar}}$	94.7	98.9	76.5/58.4
	RDT(%)	**+0.6**	**-0.1**	**+2.8/+3.7**
Arabic	AraBERT	94.3	96.3	83.1/68.6
	Student$_{\text{dissimilar}}$	93.7	96.4	81.3/66.6
Chinese	ChineseBERT	83.0	96.9	81.8/81.8
	Student$_{\text{dissimilar}}$	82.6	96.8	80.8/80.8
English	BERT	89.5	96.6	87.1/78.6
	Student$_{\text{dissimilar}}$	89.5	96.3	88.6/80.7
Finnish	FinBERT	94.4	97.9	81.0/68.8
	Student$_{\text{dissimilar}}$	94.4	95.5	77.7/65.9
Turkish	BERTurk	95.2	95.6	76.7/59.8
	Student$_{\text{dissimilar}}$	95.4	92.9	76.2/59.1
	RDT(%)	**-0.2**	**-1.1**	**-1.3/-1.4**

Table 2: Results for *monolingual teacher LMs and multilingual students* on downstream tasks as described in Section 4.2. Relative deviations of 5% or less from teacher (i.e., $\text{RDT} \geq -5\%$) are marked in bold. We find that Student$_{\text{similar}}$ outperforms individual teacher LMs, with a maximum gain of upto +2.8/+3.7% for QA, while Student$_{\text{dissimilar}}$ is competitive with teacher LMs, with a maximum drop of -1.3/-1.4% for QA. Please refer to Section 4.2 for details.

Student	Language	Teacher LM Tokens	Student LM Tokens	% of Data
Student$_{\text{similar}}$	English	3300M	2285M	69.25%
	German	23723M	847M	3.57%
	Italian	13139M	506M	3.85%
	Spanish	3000M	639M	21.31%
	Total	**43162M**	**4277M**	**9.9%**
Student$_{\text{dissimilar}}$	Arabic	8600M	135M	1.58%
	English	3300M	2285M	69.25%
	Finnish	3000M	83M	2.77%
	Turkish	4405M	60M	1.36%
	Chinese	71M	71M	100.00%
	Total	**19376M**	**2634M**	**13.6%**

Table 3: *Number of Tokens (in Millions)* in the teacher (Table 1) and student LMs as described in Section 4.2

German (*de*), Italian (*it*), Spanish (*es*), and Turkish (*tr*). Second, as previous work has evidenced positive transfer between related languages in a multilingual setup (Pires et al., 2019; Wu and Dredze, 2020), we further group the chosen teacher LMs based on language families as shown in Table 1, where:

i) Student$_{\text{similar}}$ is trained on four closely related languages from the Indo-European family – *de*, *en*, *es* and *it*.

ii) Student$_{\text{dissimilar}}$ is trained on languages from different language families – *ar*, *en*, *fi*, *tr* and *zh*.

Both student LMs have a BERT-base architecture. Student$_{\text{similar}}$ has a vocabulary size of 99,112 with a total of 162M parameters, while Student$_{\text{dissimilar}}$ has a vocabulary size of 180,996 with a total of 225M parameters. We keep a batch size of 4096 and train for 250,000 steps with a maximum sequence length of 512.

Fine-tuning: We evaluate both the teacher and student LMs on three downstream tasks with in-language fine-tuning for each task[5] :

i) Named Entity Recognition *(NER)*: We use the WikiAnn (Pan et al., 2017; Rahimi et al., 2019) dataset for all languages.

ii) Part-of-Speech Tagging *(UDPOS)*: We use the Universal Dependencies v2.6 (Zeman et al., 2020) dataset for all languages.

iii) Question Answering *(QA)*: We use DRCD for *zh* (Shao et al., 2018), TQuAD[6] for *tr*, SQuADv1.1 (Rajpurkar et al., 2016) for *en*, SQuADv1.1-translated for *it* (Croce et al., 2018) and *es* (Carrino et al., 2020) and the TyDiQA-GoldP dataset (Clark et al., 2020) for *ar* and *fi*.

Results: We report results of our teacher and student LMs in Table 2. Overall, we find that Student$_{\text{similar}}$ outperforms individual teacher models on NER (+0.6%) and QA (+2.8/3.7%) while performing competitively on UDPOS (-0.1%). Student$_{\text{dissimilar}}$ is competitive with the teacher LMs with only small differences of up to 1.3/1.4% (QA), as shown in Table 2. For each language, we find Student$_{\text{similar}}$ is either competitive or outperforms its respective teacher LM. Our results provide evidence for positive transfer across languages in two ways. First, we observe that Student$_{\text{similar}}$ outperforms Student$_{\text{dissimilar}}$ for the common language - English. Given that the English teacher (BERT) and the pre-training transfer corpora[7] is common for both student LMs,

---

[5] More details in Appendix A.3

[6] https://tquad.github.io/turkish-nlp-qa-dataset

[7] In fact, we can hypothesize that Student$_{\text{dissimilar}}$ sees more English tokens as compared to Student$_{\text{similar}}$ because the Non-English languages in Student$_{\text{dissimilar}}$ are relatively low resourced (a sum total of 349M unique tokens) in comparison to Student$_{\text{similar}}$ (a sum total of 1992M unique tokens)

Languages	Model	Teacher	PANX **F1**	UDPOS **F1**	PAWSX **Acc.**	XNLI **Acc.**	XQUAD **F1/EM**	MLQA **F1/EM**	TyDiQA **F1/EM**	**Avg.**
MuRIL Languages	mBERT	-	58.8	68.5	93.4	66.2	70.3/57.5	65.0/50.8	62.5/52.	69.2
	MuRIL	-	76.9	74.5	95.0	74.4	77.7/64.2	73.6/58.6	76.1/60.2	78.3
	$\mathrm{Student}_{\mathrm{MuRIL}}$	MuRIL	69.3	72.3	95.4	71.9	75.7/62.1	72.0/56.3	70.7/59.2	75.3
	$\mathrm{Student}_{\mathrm{mBERT}}$	mBERT	38.1	52.1	93.5	64.8	56.9/44.8	51.1/39.7	41.6/33.9	56.9
	$\mathrm{Student}_{\mathrm{Both\_all}}$	mBERT + MuRIL	67.9	72.3	94.5	71.1	76.1/62.9	70.4/55.5	70.8/55.3	74.7
	$\mathrm{Student}_{\mathrm{Both\_best}}$	mBERT + MuRIL	68.5	71.5	93.9	70.7	77.7/64.3	70.8/55.6	70.6/58.4	74.8
	$\mathrm{RDT}(\mathrm{Student}_{\mathrm{MuRIL}}, \mathrm{mBERT})$ (%)		**+17.9**	**+5.6**	**+2.1**	**+8.6**	**+7.7/+8**	**+10.8/+10.8**	**+13.1/+12.3**	**+8.8**
	$\mathrm{RDT}(\mathrm{Student}_{\mathrm{MuRIL}}, \mathrm{MuRIL})$ (%)		-9.9	**-3**	**+0.4**	**-3.4**	**-2.6/-3.3**	**-2.2/-3.9**	-7.1/**-1.7**	**-3.8**
Non MuRIL Languages	mBERT	-	63.5	71.1	80.2	65.9	62.2/47.1	59.7/41.4	60.4/46.1	66.1
	$\mathrm{Student}_{\mathrm{MuRIL}}$	mBERT	63.9	72.8	83.3	68.7	66.5/51.2	63.1/44.4	61.7/45.0	68.6
	$\mathrm{Student}_{\mathrm{mBERT}}$	mBERT	64.6	72.1	84.0	68.8	64.5/49.0	61.1/42.7	58.9/44.1	67.7
	$\mathrm{Student}_{\mathrm{Both\_all}}$	mBERT	64.1	72.6	83.9	68.1	61.3/47.1	60.5/42.2	59.7/44.0	67.2
	$\mathrm{Student}_{\mathrm{Both\_best}}$	mBERT	63.3	72.6	83.2	67.2	66.0/50.6	61.4/43.2	62.4/46.5	68.0
	$\mathrm{RDT}(\mathrm{Student}_{\mathrm{MuRIL}}, \mathrm{mBERT})$ (%)		**+0.6**	**+2.4**	**+3.9**	**+4.3**	**+6.9/+8.7**	**+5.7/+7.2**	**+2.2/-2.4**	**+3.8**

Table 4: *Results for multilingual teacher and student LMs* on the XTREME benchmark. We compare performances of three student LM variants as described in Section 4.3 to the two teachers mBERT and MuRIL. Relative deviations of 5% or less from teacher (i.e., $\mathrm{RDT} \geq -5\%$) are marked in bold. Overall, we find that $\mathrm{Student}_{\mathrm{MuRIL}}$ performs the best among all student variants and report its RDT (in %) (Equation 3) from the two teachers. Please refer to Section 4.3 for a detailed analysis.

we can attribute this gain to the fact that English is trained with linguistically and typologically similar languages in $\mathrm{Student}_{\mathrm{similar}}$. Second, $\mathrm{Student}_{\mathrm{similar}}$ outperforms its teacher LMs while $\mathrm{Student}_{\mathrm{dissimilar}}$ is competitive for *all* languages. These two results across *all* languages point towards $\mathrm{Student}_{\mathrm{similar}}$ benefiting from a positive transfer across similar languages. In Table 3, we observe that $\mathrm{Student}_{\mathrm{similar}}$ is trained on 9.9% of the total unique tokens seen by its respective teacher LMs and $\mathrm{Student}_{\mathrm{dissimilar}}$ lies close with 13.6%. Despite this huge disparity in pre-training corpora, student LMs are competitive with their teachers. This encouraging result proves that even with very limited data, MERGEDISTILL enables one to combine strong monolingual teacher LMs to train competitive student LMs that can leverage the benefits of multilinguality.

4.3 Multilingual Teacher LMs

Pre-training: In this experiment, we make use of pre-existing multilingual models: mBERT and MuRIL. mBERT is trained on 104 languages and MuRIL covers 12 of these (11 Indian languages + English): Bengali (*bn*), English (*en*), Gujarati (*gu*), Hindi (*hi*), Kannada (*kn*), Malayalam (*ml*), Marathi (*mr*), Nepali (*ne*), Punjabi (*pa*), Tamil (*ta*), Telugu (*te*), and Urdu (*ur*), with higher performance for these languages on the XTREME benchmark. We train the student model on all 104 languages. In this case, the *MuRIL Languages* (MuL) have two teachers (mBERT and MuRIL) and the *Non-MuRIL Languages* (Non-MuL) can learn from mBERT only. Therefore, while we only use mBERT as the teacher LM for Non-MuL across all experiments, we consider three possibilities for MuL :

i) $\mathbf{Student}_{\mathbf{MuRIL}}$: We only use MuRIL as the teacher LM and each input training example is tokenized using MuRIL.
ii) $\mathbf{Student}_{\mathbf{mBERT}}$: We only use mBERT as the teacher LM and each input training example is tokenized using mBERT.
iii) $\mathbf{Student}_{\mathbf{Both}}$: As highlighted in Section 3, we consider two possibilities to incorporate both teacher LM predictions in training:

- **$\mathbf{Student}_{\mathbf{Both\_all}}$**: Tokenize each input example using mBERT and MuRIL separately and include both copies in training.
- **$\mathbf{Student}_{\mathbf{Both\_best}}$**: Tokenize each input example using mBERT and MuRIL separately and include only the best copy in training. The best copy is the one having minimum teacher LM loss for the example.

Note, it is non-trivial to tokenize each example in a way that is compatible with all teacher LMs. One must resort to tokenization using an intersection of vocabularies which is sub-optimal.

All the student LMs use a BERT-base architecture and have a vocabulary size of 288,973. We reduce our embedding dimension to 256 as opposed to

---

as shown in Table 3

Model	Vocabulary	Labels	PANX	UDPOS	PAWSX	XNLI	XQUAD	MLQA	TyDiQA	Avg.
SM1	mBERT	Gold	63.2	73.0	94.8	71.2	70.2/57.9	65.1/51.3	60.8/48.7	71.2
SM2	mBERT∪MuRIL	Gold	**69.3**	**73.9**	95.3	71.2	**76.2/63.1**	71.1/56.0	70.9/56.0	**75.4**
SM3	mBERT∪MuRIL	Gold+Teacher	**69.3**	72.3	**95.4**	**71.9**	75.7/62.1	**72.0/56.3**	**70.7/59.2**	75.3
SM2_100k	mBERT∪MuRIL	Gold	65.5	72.3	94.3	67.5	72.3/58.2	66.9/51.5	62.5/51.9	71.6
SM3_100k	mBERT∪MuRIL	Gold+Teacher	71.2	73.5	93.1	69.6	76.4/62.9	69.1/53.9	68.6/54.9	74.5

Table 5: *Importance of teacher vocabulary and predictions in* MERGEDISTILL. We observe maximum performance gains, by changing the vocabulary from mBERT in SM1 to (mBERT∪MuRIL) vocabulary in SM2. Here, SM3 is the standard $\mathrm{Student}_{\mathrm{MuRIL}}$. We also observe that SM3_100k, trained for 20% of the total training steps, is competitive to SM3 and significantly outperforms SM2_100k, highlighting the importance of teacher LM predictions in a limited data scenario. Please see Section 4.4 for details.

768 to bring down the model size to be around 160M, comparable to mBERT (178M). We keep a batch size of 4096 and train for 500,000 steps with a maximum sequence length of 512.

Finetuning: We report zero-shot performance for all languages in the XTREME (Hu et al., 2020) benchmark[8].

Results: We report results of our teacher and student LMs in Table 4. Overall, we find that $\mathrm{Student}_{\mathrm{MuRIL}}$ performs the best among all student variants. For Non-MuL, $\mathrm{Student}_{\mathrm{MuRIL}}$ beats the teacher (mBERT) by an average relative score of 3.8%. For MuL, $\mathrm{Student}_{\mathrm{MuRIL}}$ beats one teacher (mBERT) by 8.8%, but underperforms the other teacher (MuRIL) by 3.8%. There can be two factors at play here. MuRIL is trained on monolingual and parallel data [9] while the student LMs only see $\sim$22% of unique tokens in comparison. MuRIL also has different language sampling strategies ($\alpha = 0.3$ as opposed to 0.7 in our setting, where a lower α value upsamples more rigorously from the tail languages), which have a significant role to play in multilingual model performances (Conneau et al., 2020). We also observe a significant drop in $\mathrm{Student}_{\mathrm{mBERT}}$'s performance for MuL when compared to the other student LM variants. This might be because the input is tokenized using the mBERT tokenizer which prevents learning from MuRIL tokens in the student vocabulary. For $\mathrm{Student}_{\mathrm{Both}}$, we do not observe much of a difference between $\mathrm{Student}_{\mathrm{Both\_all}}$ and $\mathrm{Student}_{\mathrm{Both\_best}}$. This observation may differ with one's choice of teacher LMs depending on how well it performs for a particular language. In our case, we don't observe much of a difference in incorporating mBERT predictions for MuL.

4.4 Further Analysis

The importance of vocabulary and teacher LM preditions: In Table 4, we see that $\mathrm{Student}_{\mathrm{MuRIL}}$ significantly outperforms mBERT for MuL, despite both being trained on Wikipedia corpora, and having comparable model sizes. With regard to MuL, $\mathrm{Student}_{\mathrm{MuRIL}}$ differs from mBERT in two main aspects – **i)** $\mathrm{Student}_{\mathrm{MuRIL}}$'s vocabulary is a union of mBERT and MuRIL vocabularies. **ii)** $\mathrm{Student}_{\mathrm{MuRIL}}$ is trained with additional MuRIL predictions as soft labels. To disentangle the role both these factors play in $\mathrm{Student}_{\mathrm{MuRIL}}$'s improved performance, we train two models :
i) SM1 is trained exactly like $\mathrm{Student}_{\mathrm{MuRIL}}$, but with mBERT vocabulary and on gold labels.
ii) SM2 is trained using $\mathrm{Student}_{\mathrm{MuRIL}}$'s vocabulary (mBERT ∪ MuRIL) but on gold labels only, without teacher predictions.

The results are summarized in Table 5. Note, we refer to $\mathrm{Student}_{\mathrm{MuRIL}}$ as SM3. Overall, we observe a $\sim$**4.2%** gain in average performance for SM2 over SM1. This clearly highlights that given fixed data and model capacity, LM training significantly benefits by incorporating a strong teacher's vocabulary.

Furthermore, we also observe that SM2 and SM3 achieve competitive performances despite SM3 being additionally trained on teacher LM labels. To motivate the need for teacher predictions, Hinton et al. (2015) argue that when soft targets have high entropy, they provide much more information per training case than hard targets and can be trained on *much less data* than the original cumbersome model. In our case, we hypothesize that training on 500,000 steps exposes the model to sufficient data for it to generalize well enough and mask the benefits of teacher LM predictions. To validate this, we evaluate the performances of SM2 and SM3,

---

[8] More details in Appendix A.3

[9] More details in Appendix A.2

20% into training (i.e. 100,000 steps / 500,000 total steps) as shown in Table 5. We observe a ∼**2.9%** gain in average performance for SM3 over SM2, clearly highlighting the importance of teacher LM predictions in a limited data scenario. This is especially important when one has access to very limited monolingual data and a strong teacher LM for a particular language.

Pre-trained zero-shot transfer: Interestingly, $\text{Student}_{\text{MuRIL}}$ performs the best on almost all tasks for *Non-MuL*. This hints at positive transfer from strong teachers to languages that the teacher does not cover at all, due to the shared multilingual representations.[10] This would mean that learning from strong teachers can improve the student model's performance in a zero-shot manner on related languages not covered by the teacher. This would make MERGEDISTILL highly beneficial for low-resource languages that do not have a strong teacher or limited gold data. We leave this exploration to future work.

5 Conclusion

In this paper we address the problem of merging multiple pre-trained teacher LMs into a single multilingual student LM by proposing MERGEDISTILL, a task-agnostic distillation method. To the best of our knowledge, this is the first attempt of its kind. The student LM learned by MERGEDISTILL may be further fine-tuned for any task across all of the languages covered by the teacher LMs. Our approach results in better maintainability (fewer models) and is compute efficient (due to offline predictions). We use MERGEDISTILL to **i)** combine *monolingual* teacher LMs into one student multilingual LM which is competitive with the teachers, thereby demonstrating positive cross-lingual transfer, and **ii)** combine multilingual LMs to train student LMs that learn from *multiple* teachers. Through experiments on multiple benchmark datasets, we show that student LMs learned by MERGEDISTILL perform competitively or even outperform teacher LMs trained on orders of magnitude more data. We disentangle the positive impact of incorporating strong teacher LM vocabularies and learning from teacher LM predictions, highlighting the importance of the latter in a limited data scenario. We also find that MERGEDISTILL enables positive transfer from strong teachers to languages not covered by them (i.e. zero-shot transfer). Our work bridges the gap between the universe of language-specific models and massively multilingual LMs, incorporating benefits of both into one framework.

6 Acknowledgements

We would like to thank the anonymous reviewers for their insightful and constructive feedback. We thank Iulia Turc, Ming-Wei Chang, and Slav Petrov for valuable comments on an earlier version of this paper.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 11–16.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Casimiro Pio Carrino, Marta R. Costa-jussà, and José A. R. Fonollosa. 2020. Automatic Spanish translation of SQuAD dataset for multi-lingual question answering. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5515–5523, Marseille, France. European Language Resources Association.

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *Proceedings of*

---

[10] For example, if you want to train a multilingual model covering English and a closely related low-resource language for which there exists no strong teacher, it may be possible to improve performance for the low resource language using teacher predictions for English only, due to a shared embedding space and possibly shared sub-words.

the 28th International Conference on Computational Linguistics, pages 6588–6608, Barcelona, Spain (Online). International Committee on Computational Linguistics.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. 2020. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model.

Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking embedding coupling in pre-trained language models. In *International Conference on Learning Representations*.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4536–4546, Online. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Danilo Croce, Alexandra Zelenanska, and Roberto Basili. 2018. Neural learning for question answering in italian. In *AI*IA 2018 – Advances in Artificial Intelligence*, pages 389–402, Cham. Springer International Publishing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *arXiv preprint arXiv:1809.04686*.

Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, and Jingjing Liu. 2020. Filter: An enhanced fusion method for cross-lingual language understanding. *arXiv preprint arXiv:2009.05166*.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 710–721, Florence, Italy. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *arXiv preprint arXiv:2003.11080*.

Prabhu Kaliamoorthi, Aditya Siddhant, Edward Li, and Melvin Johnson. 2021. Distilling large language models into tiny and effective students using pqrnn.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha Talukdar. 2021. Muril: Multilingual representations for indian languages.

Yuri Kuratov and Mikhail Arkhipov. 2019. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. In *Proceedings of NeurIPS 2019*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. 2019a. Robust neural machine translation with joint textual and phonetic embedding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3044–3049, Florence, Italy. Association for Computational Linguistics.

Linqing Liu, Huan Wang, Jimmy Lin, Richard Socher, and Caiming Xiong. 2020. Mkd: a multi-task knowledge distillation approach for pretrained language models.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2019. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS 2017*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific bert models. *arXiv preprint arXiv:2003.02912*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2020. How good is your tokenizer? on the monolingual performance of multilingual language models. *arXiv preprint arXiv:2012.15613*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Stefan Schweter. 2020a. Berturk - bert models for turkish.

Stefan Schweter. 2020b. Italian bert and electra models.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. Drcd: a chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*.

Aditya Siddhant, Melvin Johnson, Henry Tsai, Naveen Arivazhagan, Jason Riesa, Ankur Bapna, Orhan Firat, and Karthik Raman. 2020. Evaluating the Cross-Lingual Effectiveness of Massively Multilingual Neural Machine Translation. In *Proceedings of AAAI 2020*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. *arXiv preprint arXiv:1902.10461*.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical bert models for sequence labeling. *arXiv preprint arXiv:1909.00100*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Daniel Zeman, Joakim Nivre, and Mitchell Abrams et al. 2020. Universal dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

A Appendix

A.1 Knowledge Distillation

We train our LMs with the MLM objective. Let x denote a sequence of tokens where $\mathrm{x_m} = \{\mathrm{x_1, x_2, x_3 ... x_n}\}$ denote the masked tokens, and $\mathrm{x_{-m}}$ denote the non-masked tokens. Let v be the vocabulary of LM θ. The log-likelihood loss (cross-entropy with one-hot label) can be formulated as follows:

$$\mathrm{L_{MLM}(x_m|x_{-m})} = -\frac{1}{\mathrm{n}}\sum_{\mathrm{i=1}}^{\mathrm{n}}\sum_{\mathrm{k=1}}^{|\mathrm{v}|}\mathrm{P(x_{m_i}, k)};$$

$$\mathrm{P(x_{m_i}, k)} = \mathbf{1}(\mathrm{x_{m_i} = k})\mathrm{log p(x_{m_i} = k|x_{-m}; \theta)}$$

In a distillation setup, the student is trained to not only match the one-hot labels for masked words, but also the probability output distribution of the teacher t. Let us denote the teacher output probability distribution for token $\mathrm{x_{m_i}}$ by $\mathrm{Q(x_{m_i}|x_{-m}; \theta_t)}$. The cross entropy between the teacher and student distributions then serves as the distillation loss :

$$\mathrm{L_{KD}(x_m|x_{-m})} = -\frac{1}{\mathrm{n}}\sum_{\mathrm{i=1}}^{\mathrm{n}}\sum_{\mathrm{k=1}}^{|\mathrm{v}|}\hat{\mathrm{P}}\mathrm{(x_{m_i}, k)};$$

$$\begin{aligned}\hat{\mathrm{P}}\mathrm{(x_{m_i}, k)} = \mathrm{Q(x_{m_i}} &= \mathrm{k|x_{-m}; \theta_t)} \\ \mathrm{log p(x_{m_i}} &= \mathrm{k|x_{-m}; \theta)}\end{aligned}$$

The total loss is then defined as :

$$\mathrm{L_{ALL}} = \lambda \mathrm{L_{KD}} + (1-\lambda)\mathrm{L_{MLM}}$$

With the addition of the teacher, the target distribution is no longer a single one-hot label, but a smoother distribution with multiple words having non-zero probabilities which yields in a smaller variance in gradients (Hinton et al., 2015). Intuitively, a single masked word can have several valid predictions, which appropriately fit the context.

Teacher	Language	Teacher LM Tokens	Student LM Tokens	% of Data
MuRIL	Bengali	1181M	27M	2.30%
	English	6986M	2816M	40.30%
	Gujarati	173M	7M	3.90%
	Hindi	2368M	38M	1.61%
	Kannada	196M	15M	7.64%
	Malayalam	337M	14M	4.17%
	Marathi	274M	8M	3.02%
	Nepali	231M	5M	2.16%
	Punjabi	141M	9M	6.45%
	Tamil	769M	26M	3.34%
	Telugu	331M	30M	8.99%
	Urdu	722M	23M	3.21%
	Total	**13709M**	**3018M**	**22%**

Table 6: *Number of Tokens (in Millions)* in the teacher (MuRIL) and student LMs as described in Section 4.3. Note, we only show the *MuRIL Languages* here because for *Non-MuRIL Languages*, the teacher (mBERT) and student variants are trained on the same data.

A.2 Pre-training Details

A.2.1 Monolingual Teacher LMs

We pre-train our student models using the BERT base architecture. $\text{Student}_{\text{similar}}$ has a vocabulary size of 99112 and a model size of 162M parameters. $\text{Student}_{\text{different}}$ has a vocabulary size of 180996 and a model size of 225M parameters. We keep a batch size of 4096 and train for 250k steps with a maximum sequence length of 512. We use TPUs, and it takes around 1.5 days to pre-train each student LM.

A.2.2 Multilingual Teacher LMs

We pre-train our student models using the BERT base architecture. All student LMs have a vocabulary size of 288973. Hence, we reduce our embedding dimension to 256 as opposed to 768 to bring down the model size to be around 160M, comparable to mBERT (178M). We keep a batch size of 4096 and train for 500k steps with a maximum sequence length of 512. We use TPUs, and it takes around 3 days to pre-train each student LM.

We present pre-training data statistics for MuRIL and the student LMs in Table 6. Here we only include the monolingual data statistics, but MuRIL is additionally trained on parallel translated and transliterated data.

Task	Batch	Learning Rate	No. of Epochs	Warmup Ratio	Max. seq. Length
NER	32	3e-5	10	0.1	256
POS	32	3e-5	10	0.1	256
QA	32	3e-5	10	0.1	384

Table 7: Hyperparameter Details for each fine-tuning task in Section 4.2

A.3 Fine-tuning Details

A.3.1 Monolingual Teacher LMs

Data Statistics We evaluate our monolingual teacher LMs and multilingual student LMs, as described in Section 4.2, on three tasks as follows:

i) Named Entity Recognition (NER): We use the WikiAnn (Pan et al., 2017; Rahimi et al., 2019) dataset for all languages. Each language comprises of a train/dev/test split of 20000/10000/10000 tokens. Specifically, we use the huggingface re-packaged implementation of the dataset[11].

ii) Part-of-Speech tagging (POS): We use the Universal Dependencies v2.6 (Zeman et al., 2020) dataset for all languages. Detailed statistics for each language can be found in Table 9. Specifically, we use the huggingface re-packaged implementation of the dataset[12].

iii) Question Answering (QA): We use the TyDiQA dataset (Clark et al., 2020) for *ar* and *fi*, SQuADv1.1 (Rajpurkar et al., 2016) for *en*, SQuAD-translated for *it* (Croce et al., 2018) and *es* (Carrino et al., 2020), DRCD for *zh* (Shao et al., 2018) and TQuAD[13] for *tr*. Detailed statistics for each language can be found in Table 10. Note, we use the dev set as our test sets, since most datasets only have a train/dev split. We use 10% of randomly shuffled training examples as our dev sets.

Hyperparameter Details: We use the same hyperparameters for fine-tuning all teacher and student LMs, as shown in Table 7. We report results on the best-performing checkpoint for the validation set. The performance of the best checkpoint on validation sets are shown in Table **??**

---

[11] https://huggingface.co/datasets/wikiann

[12] https://huggingface.co/datasets/universal_dependencies

[13] https://tquad.github.io/turkish-nlp-qa-dataset

Languages	Model	PANX	UDPOS	PAWSX	XNLI	XQUAD	MLQA	TyDiQA
MuRIL Languages	mBERT	58.8	68.5	93.4	66.2	70.3/57.5	65.0/50.8	62.5/52.7
	MuRIL	**76.9**	74.5	95.0	**74.4**	**77.7/64.2**	**73.6/58.6**	**76.1/60.2**
	k=8	69.3	72.3	**95.4**	71.9	75.7/62.1	72.0/56.3	70.7/59.2
	k=128	67.5	72.8	94.4	70.7	75.5/61.9	71.1/56.1	70.2/55.4
	k=512	69.2	**77.2**	94.7	71.3	75.6/61.8	72.3/56.9	68.5/53.9
Non MuRIL Languages	mBERT	63.5	71.1	80.2	65.9	62.2/47.1	59.7/41.4	60.4/46.1
	k=8	63.9	72.8	83.3	**68.7**	**66.5/51.2**	63.1/44.4	**61.7/45.0**
	k=128	63.7	72.8	**83.4**	67.9	66.1/51.1	61.4/43.4	62.6/46.7
	k=512	**64.8**	**73.3**	82.7	67.4	65.7/50.7	**63.6/44.9**	58.7/44.8
All Languages	mBERT	62.5	70.6	82.0	65.9	63.7/49.0	61.2/44.1	61.1/48.3
	k=8	65.0	72.7	**85.0**	**69.3**	**68.2/53.2**	65.6/47.8	**64.7/49.7**
	k=128	64.5	72.8	**85.0**	68.4	67.9/53.0	64.2/47.1	65.2/49.6
	k=512	**65.7**	**74.0**	84.4	68.2	67.5/52.8	**66.1/48.3**	62.0/47.8

Table 8: *Results of the best performing student model* $\mathrm{Student}_{\mathrm{MuRIL}}$ for different top-k values

Language	Dataset	Examples (Train/Dev/Test)
Arabic	AR_PADT	6075/909/680
Chinese	ZH_GSD	3997/500/500
English	EN_EWT	12543/2002/2077
German	DE_HDT	15305/18434/18459
Finnish	FI_FTB	14981/1875/1867
Italian	IT_ISDT	13121/564/482
Spanish	ES_ANCORA	14305/1654/1721
Turkish	TR_IMST	3664/988/983

Table 9: *Universal Dependencies v2.6* overview for each language, used in Section 4.2

Language	Dataset	Examples (Train/Test)
Arabic	TyDiQA-GoldP	14805/921
Chinese	DRCD	26936/3524
English	SQuADv1.1	87599/10570
German	-	-
Finnish	TyDiQA-GoldP	6855/782
Italian	SQuADv1.1-translated	87599/10570
Spanish	SQuADv1.1-translated	87595/10570
Turkish	TQuAD	8308/892

Table 10: *Question Answering datasets*, used in Section 4.2

A.3.2 Multilingual Teacher LMs

Data Statistics We evaluate all the teacher (mBERT and MuRIL) and student ($\mathrm{Student}_{\mathrm{MuRIL}}$, $\mathrm{Student}_{\mathrm{mBERT}}$ and $\mathrm{Student}_{\mathrm{Both}}$) LMs on the XTREME (Hu et al., 2020) benchmark. We fine-tune the pre-trained models on English training data for the particular task, except TyDiQA, where we use additional SQuAD v1.1 English training data, similar to (Fang et al., 2020). All results are computed in a zero-shot setting.

Hyperparameter Details We use the same hyperparameters for fine-tuning all teacher and student LMs, as shown in Table 11. We report results on the best-performing checkpoint for the eval set.

Task	Batch	Learning Rate	No. of Epochs	Warmup Ratio	Max. seq. Length
PANX	32	2e-5	10	0.1	128
UDPOS	64	5e-6	10	0.1	128
PAWSX	32	2e-5	5	0.1	128
XNLI	32	2e-5	3	0.1	128
XQuAD	32	3e-5	2	0.1	384
MLQA	32	3e-5	2	0.1	384
TyDiQA	32	3e-5	2	0.1	384

Table 11: Hyperparameter Details for each task in XTREME

A.4 Different top-k values

We present results for $\mathrm{Student}_{\mathrm{MuRIL}}$ trained with different top-k values from teacher predictions in Table 8. We observe that while performances remain similar for higher values of k, storage becomes increasingly expensive. Hence, we stick to a value of k=8 in all our experiments.

On Sparsifying Encoder Outputs in Sequence-to-Sequence Models

Biao Zhang[1] **Ivan Titov**[1,2] **Rico Sennrich**[3,1]

[1]School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam
[3]Department of Computational Linguistics, University of Zurich
`B.Zhang@ed.ac.uk, ititov@inf.ed.ac.uk, sennrich@cl.uzh.ch`

Abstract

Sequence-to-sequence models usually transfer *all* encoder outputs to the decoder for generation. In this work, by contrast, we hypothesize that these encoder outputs can be compressed to shorten the sequence delivered for decoding. We take Transformer as the testbed and introduce a layer of stochastic gates in-between the encoder and the decoder. The gates are regularized using the expected value of the sparsity-inducing L_0 penalty, resulting in completely masking-out a subset of encoder outputs. In other words, via joint training, the $\mathcal{L}_0$Drop layer forces Transformer to route information through a subset of its encoder states. We investigate the effects of this sparsification on two machine translation and two summarization tasks. Experiments show that, depending on the task, around 40–70% of source encodings can be pruned without significantly compromising quality. The decrease of the output length endows $\mathcal{L}_0$Drop with the potential of improving decoding efficiency, where it yields a speedup of up to 1.65× on document summarization and 1.20× on character-based machine translation against the standard Transformer. We analyze the $\mathcal{L}_0$Drop behaviour and observe that it exhibits systematic preferences for pruning certain word types, e.g., function words and punctuation get pruned most. Inspired by these observations, we explore the feasibility of specifying rule-based patterns that mask out encoder outputs based on information such as part-of-speech tags, word frequency and word position. [1]

1 Introduction

Neural sequence-to-sequence (Seq2Seq) models have dominated various text generation tasks, including machine translation (Vaswani et al., 2017) and abstractive document summarization (Gehrmann et al., 2018; Liu and Lapata, 2019). These models generally follow the encoder-decoder paradigm, where the encoder interprets source context and converts source words into vector representations such that the decoder has sufficient information to predict the target sequence. Early Seq2Seq models (Sutskever et al., 2014; Cho et al., 2014) provided only the last and/or first encoder states to the decoder. In contrast, modern approaches rely on the attention mechanism (Bahdanau et al., 2015) and implicitly make an assumption that information from *all* encoder outputs should flow to the decoder.[2] However, this assumption neglects the fact that a large portion of source words in machine translation receives just minor attention as shown in Figure 1, let alone in summarization where the input contains redundant expressions and large parts of text are not relevant to any plausible summary. Moreover, information content varies

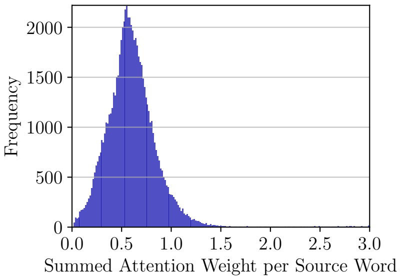

Figure 1: Distribution of the summed attention weight per source word estimated on the English-German WMT14 test set. For each (source sentence, translation) pair, we extract the attention matrices from all encoder-decoder attention sublayers in Transformer and average them over different (8) heads and (6) layers. The attention value for each source word is summed over all target words in the translation. Higher attention weights suggest larger impacts on translation. Around 49.7% source words get attention weights of less than 0.6, compared to the mean value of 1.03.

---

[1] Source code is available at `https://github.com/bzhangGo/zero`.

[2] We interchangeably use *source representation*, *encoder output* and *source encoding* unless otherwise specified.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2888–2900
August 1–6, 2021.

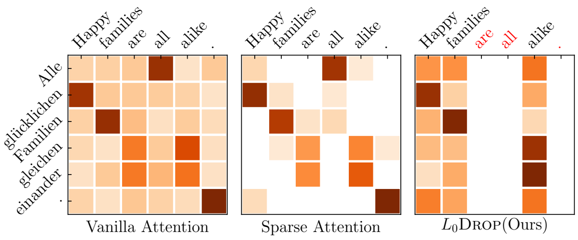

Vanilla Attention

Sparse Attention

L_0Drop(Ours)

Figure 2: Encoder-decoder attention distribution of target words (y-axis) over source words (x-axis) for the vanilla attention (Vaswani et al., 2017), the sparse attention (Correia et al., 2019) and our model. Darker color indicates larger attention weight, and the white blocks denote an attention weight of 0. The source words whose encoding is pruned by $\mathcal{L}_0$Drop (receiving zero weight) are highlighted in red.

across words, for example, it is negatively correlated with event frequency (Shannon, 1948; Zipf, 1949). When moving from word-level to character-level processing, the notion of encoding information and computing attention on the level of characters also seems excessive. Previous work has proposed hierarchical architectures where character-level encodings are compressed into word-level or span-level states (Ling et al., 2015; Lee et al., 2017).

In this work, we hypothesize that encoder outputs are compressible and we can force Seq2Seq model to route information through their subset. Figure 2 illustrates our intuition as well as the difference with existing work (Vaswani et al., 2017; Correia et al., 2019). Instead of dynamically sparsifying attention weights for individual decoder steps (Correia et al., 2019), we aim at detecting uninformative source encodings and dropping them to shorten the encoding sequence before generation. To this end, we build on recent work on sparsifying weights (Louizos et al., 2018) and activations (Bastings et al., 2019) of neural networks. Specifically, we insert a differentiable neural sparsity layer ($\mathcal{L}_0$Drop) in-between the encoder and the decoder. The layer can be regarded as providing a multiplicative scalar gate for every encoder output. The gate is a random variable and, unlike standard attention, can be exactly zero, effectively masking out the corresponding source encodings. The sparsity is promoted by introducing an extra term to the learning objective, i.e. an expected value of the sparsity-inducing L_0 penalty. By varying the coefficient for the regularizer, we can obtain different levels of sparsity. Importantly, the objective remains fully end-to-end differentiable.

Given an encoding sequence of length N, the vanilla attention model attends to it recurrently for M steps at the decoding phase, leading to a computational complexity of $\mathcal{O}(NM)$ ($N = 6$, $M = 6$ in Figure 2). This could be costly if N or M is very large. With the induced sparse structure by $\mathcal{L}_0$Drop, we introduce a specialized decoding algorithm which lowers this complexity to $\mathcal{O}(N'M)$ ($N' \leq N$, and $N' = 3$ in Figure 2). As a result, $\mathcal{L}_0$Drop can improve decoding efficiency by reducing the encodings' length, especially for long inputs.

We apply $\mathcal{L}_0$Drop to Transformer (Vaswani et al., 2017), the state-of-the-art Seq2Seq model. We conduct extensive experiments on WMT translation tasks with two language pairs and document summarization tasks covering single document and multiple documents settings. We also explore character-based machine translation where the lengthy character sequence often leads to slow inference. We analyze how pruning source encodings impacts the generation quality and which word types get pruned. Inspired by the analysis of $\mathcal{L}_0$Drop, we further study rule-based sparsity patterns , such as deterministically filtering out the encodings of words with specific POS tags, high-frequency words or simply attending to every other word in the sequence.

Our main findings are summarized as follows:

- We confirm that the encoder outputs can be compressed, around 40–70% of them can be dropped without large effects on the generation quality.
- The resulting sparsity level differs across word types, the encodings corresponding to function words (such as determiners, prepositions) are more frequently pruned than those of content words (e.g., verbs and nouns).
- $\mathcal{L}_0$Drop can improve decoding efficiency particularly for lengthy source inputs. We achieve a decoding speedup of up to 1.65× on document summarization tasks and 1.20× on character-based machine translation task.
- Filtering out source encodings with rule-based sparse patterns is feasible, and confirms information-theoretic expectations, although rule-based patterns do not generalize well across tasks.

2 Related Work

Approaches to compression in Seq2Seq models fall into the category of model parameter compression (See et al., 2016), sequential knowledge distillation (Kim and Rush, 2016) or sparse attention induction that ranges from modeling hard attention (Wu et al., 2018) to developing differentiable sparse softmax functions or regularizing attention weights for sparsity (Niculae and Blondel, 2017; Correia et al., 2019; Cui et al., 2019; Zhang et al., 2019). Unfortunately, the success of all these studies builds upon the access to all source encodings in training and decoding. Learning which encoder outputs to prune in Seq2Seq models, to the best of our knowledge, has never been investigated before. Sukhbaatar et al. (2019) learn attention spans in self-attention and discard information from states outside of the span; this method is not directly applicable to encoder-decoder attention.

We use the differentiable L_0-relaxation which was first introduced by Louizos et al. (2018) in the context of pruning individual neural network parameters. It was previously used to prune heads in multi-head attention (Voita et al., 2019). Our work is more similar in spirit to Bastings et al. (2019) where they used the L_0 relaxations to construct interpretable classifiers, i.e. models that can reveal which words they rely on when predicting a class. In their approach, the information from dropped words is lost rather than rerouted into the states of retained words, as desirable for interpretability but problematic in the text generation set-up.

The number of the source encodings selected by $\mathcal{L}_0$Drop is sentence-dependent, which differs from the linear-time model of Wang et al. (2019), although both can accelerate decoding. Our study of rule-based sparsity patterns is in line with the sparse Transformer (Child et al., 2019) though we also explore the use of external linguistic information (POS tag) in our sparsification rules, and focus on encoder outputs instead of self-attention.

Character-based translation gained increasing popularity due to its capability of handling out-of-vocabulary issues while avoiding tokenization and subword segmentation (Ling et al., 2015; Costa-jussà and Fonollosa, 2016; Sennrich, 2017; Cherry et al., 2018). Recent efforts often focus on closing the performance gap against its subword-level counterpart (Libovický and Fraser, 2020; Gao et al., 2020), but little study explores solutions to improve its inefficient inference resulted from the long character sequences. In this respect, Cherry et al. (2018) proposed to use conditional computation to dynamically compress encoder states. Similar to our results, they also observed a trade-off between the translation quality and the degree of compression.

3 Background: Transformer

We take Transformer (Vaswani et al., 2017) as our testbed. Transformer uses the dot-product attention network as its backbone to handle intra- and inter-sequence dependencies:

$$\textsc{Att}(\mathbf{H}, \mathbf{M}) = \mathbf{AV} = \textsc{Sm}\left(\frac{\mathbf{QK}^T}{\sqrt{d}}\right)\mathbf{V}, \qquad (1)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{HW}_q, \mathbf{MW}_k, \mathbf{MW}_v$. The input $\mathbf{H} \in \mathbb{R}^{J \times d}$ of length J queries and summarizes task-relevant clues from the memory $\mathbf{M} \in \mathbb{R}^{I \times d}$ of length I based on their dot-product semantic matching $\mathbf{A} \in \mathbb{R}^{J \times I}$. $\textsc{Sm}$ denotes the softmax function, d is the model dimension, and $\mathbf{W}_q$, $\mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$ are trainable model parameters. Vaswani et al. (2017) also extend this mechanism to multi-head attention.

Given a source sequence $X = (x_1, x_2, \ldots, x_N)$, Transformer maps it to the target sequence $Y = (y_1, y_2, \ldots, y_M)$ following the encoder-decoder paradigm (Bahdanau et al., 2015):[3]

$$\begin{aligned} \mathbf{X}^L &= \text{Encoder}\left(\mathbf{X}^0\right) && (2) \\ &\overset{L}{\underset{l=1}{:=}} \textsc{Ffn}\left(\textsc{Att}(\mathbf{X}^{l-1}, \mathbf{X}^{l-1})\right), \\ \mathbf{Y}^L &= \text{Decoder}\left(\mathbf{Y}^0, \mathbf{X}^L\right) && (3) \\ &\overset{L}{\underset{l=1}{:=}} \textsc{Ffn}\left(\textsc{Att}\left(\overline{\textsc{Att}}(\mathbf{Y}^{l-1}, \mathbf{Y}^{l-1}), \mathbf{X}^L\right)\right), \end{aligned}$$

where $\mathbf{X}^0 \in \mathbb{R}^{N \times d}$ and $\mathbf{Y}^0 \in \mathbb{R}^{M \times d}$ stand for the source and the shifted target sequence embedding, respectively, enriched with positional encoding (Vaswani et al., 2017). $\textsc{Ffn}(\cdot)$ is a point-wise feed-forward network. $\overline{\textsc{Att}}(\cdot, \cdot)$ in the decoder denotes masked $\textsc{Att}(\cdot, \cdot)$ which prevents access to future target words. Both the encoder and the decoder involve a stack of L identical layers, with the encoder output $\mathbf{X}^L$ fed to the decoder via an encoder-decoder attention sublayer, i.e. the $\textsc{Att}(\cdot, \cdot)$ in Eq. (3). Based on the decoder output $\mathbf{Y}^L$, Transformer performs the next-word prediction and adopts the maximum likelihood loss for training.

4 Neural Sparsity Layer: $\mathcal{L}_0$Drop

In this section, we introduce a neural sparsity layer ($\mathcal{L}_0$Drop), which we use to prune encoder outputs.

---

[3] Each sublayer ($\textsc{Att}$/$\overline{\textsc{Att}}$/$\textsc{Ffn}$) in the encoder and decoder is wrapped with residual connection (He et al., 2015) followed by layer normalization (Ba et al., 2016), which are dropped in Eq. (2) and (3) for clarity.

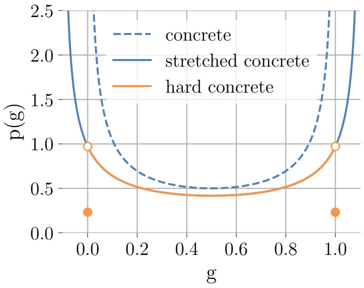

Figure 3: Hard concrete distribution (orange curve): samples are first stretched from the binary concrete distribution (dashed blue curve) to a stretched distribution (solid blue curve), and then rectified to collapse the probability mass of the shadow areas into $\{0\}$ and $\{1\}$ (solid orange points).

At inference time, only retained encoder outputs will be used as input to the decoder.

4.1 Training with $\mathcal{L}_0$Drop

Intuitively, $\mathcal{L}_0$Drop assigns each encoder output $\mathbf{x}_i^L$ a gate $g_i \in [0, 1]$ ($i \in \{1, \ldots, N\}$)

$$\mathcal{L}_0\text{Drop}(\mathbf{x}_i^L) = g_i \mathbf{x}_i^L, \tag{4}$$

and prunes encodings by closing their gates, i.e. $g_i = 0$, relying on adding a differentiable sparsity-inducing penalty to the objective.

More formally, to achieve sparsity, each gate is assumed to be a random variable and its value is drawn from the HardConcrete distribution:

$$g_i \sim \text{HardConcrete}(\alpha_i, \beta, \epsilon), \tag{5}$$

where α_i, β and ϵ are shape parameters of the distribution. HardConcrete (Louizos et al., 2018) is a parameterized family of mixed discrete-continuous distributions over the closed interval $[0, 1]$. These distributions have point mass at 0 and 1 and continuous density in-between, i.e. in $(0, 1)$, as shown in Figure 3. Thus, the gates will have a non-zero probability of being exactly 0, corresponding to masking out the input completely.

Specifically, the sample from HardConcrete distribution is obtained by stretching and rectifying samples from BinaryConcrete distributions (Maddison et al., 2017; Jang et al., 2017):

$$s_i \sim \text{BinaryConcrete}(\alpha_i, \beta) \tag{6}$$
$$\bar{s}_i = s_i (1 + 2\epsilon) - \epsilon, \tag{7}$$
$$g_i = \min(1, \max(0, \bar{s}_i)). \tag{8}$$

In the above expression, we first obtain a sample from the BinaryConcrete distribution (Eq. (6)), then stretch it from $(0, 1)$ to $(-\epsilon, 1 + \epsilon)$ (Eq. (7), $\epsilon > 0$), and finally rectify with a hard sigmoid to the closed interval $[0, 1]$ (Eq. (8)).

The probability of g_i being exactly 0 ($p(g_i = 0|\alpha_i, \beta, \epsilon)$) equals the probability of $\bar{s}_i$ hitting $(-\epsilon, 0)$ and is available in a closed form (Louizos et al., 2018):

$$p(g_i = 0|\alpha_i, \beta, \epsilon) = \sigma(\beta \log \frac{\epsilon}{1 + \epsilon} - \log \alpha_i),$$

where $\sigma(\cdot)$ denotes the sigmoid function. The parameter α_i (i.e. the location parameter of BinaryConcrete) is predicted relying on the encoder output $\mathbf{x}_i$:

$$\log \alpha_i = \mathbf{x}_i^L \mathbf{w}^T, \tag{9}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a learned parameter vector; the temperature β and the stretch degree ϵ are treated as hyperparameters. By adjusting α_i the model can change the shape of the HardConcrete distribution, and dynamically decide which outputs to pass to the decoder and which to prune.

Note that the sum

$$\mathcal{L}_0(X) = \sum_{i=1}^{N} 1 - p(g_i = 0|\alpha_i, \beta, \epsilon), \tag{10}$$

yields the expected number of open gates, or, equivalently, the expected L_0 loss on gate vector $(g_1, \ldots, g_N)$. Minimizing the loss encourages the model to prune encoder outputs.

Once $\mathcal{L}_0$Drop is integrated as a new layer into Transformer, the decoder, previously defined in Eq. (3), becomes:

$$\mathbf{Y}^L = \text{Decoder}(\mathbf{Y}^0, \mathcal{L}_0\text{Drop}(\mathbf{X}^L)). \tag{11}$$

Other components in Transformer are kept intact, except for using a modified objective $\mathcal{L}(X, Y)$:

$$\begin{aligned} &\mathcal{L}_{\text{MLE}}(X, Y) + \lambda \mathcal{L}_0(X) \\ &= -\log \mathbb{E}_{\mathbf{g} \sim p(\mathbf{g}|\phi)} [p(Y, \mathbf{g}|X)] + \lambda \mathcal{L}_0(X) \\ &\leq \mathbb{E}_{\mathbf{g} \sim p(\mathbf{g}|\phi)} [-\log p(Y, \mathbf{g}|X)] + \lambda \mathcal{L}_0(X) \\ &= \mathcal{L}(X, Y) \end{aligned} \tag{12}$$

where ϕ is short for $(\alpha, \beta, \epsilon)$, $\lambda \in \mathbb{R}^+$ is a hyperparameter defining the level of sparsity. The bound is derived by applying Jensen's inequality.

Importantly, the objective remains fully differentiable as we can rely on the reparameterization technique (Kingma and Welling, 2013) to sample $\tilde{\mathbf{g}}$ for computing unbiased estimates of the gradients. Adding $\mathcal{L}_0$Drop and the regularizer introduces only a negligible computational overhead to training compared to the original Transformer.

Algorithm 1 Decoding algorithm for $\mathcal{L}_0$Drop

```
Input: Source encodings, $\mathbf{X}^L \in \mathbb{R}^{N \times d}$;
       Gates, $\hat{\mathbf{g}} \in \mathbb{R}^N$;
       Query state, $\mathbf{y}_j^l \in \mathbb{R}^d$;
Output: Attention vector for the query
    ▷ step 1: reorganize source-side inputs
    ▷ note this step can be done before the decoder
1:  $I \leftarrow \{i|\hat{g}_i \neq 0\}$    ▷ $N' \leftarrow |I|$
2:  $\hat{\mathbf{g}}' \in \mathbb{R}^{N'}, \mathbf{X'}^L \in \mathbb{R}^{N' \times d} \leftarrow \hat{\mathbf{g}}[I], \mathbf{X}^L[I]$
3:  $c \leftarrow N - N'$
4:  $\bar{\mathbf{X}}^L \leftarrow [\mathbf{0} \in \mathbb{R}^d, \mathbf{X'}^L \odot \hat{\mathbf{g}}'], \mathbf{c} \leftarrow [c, \mathbf{1} \in \mathbb{R}^{N'}]$
    ▷ step 2: attention with counts
5:  $\mathbf{q}, \mathbf{K}, \mathbf{V} \leftarrow \mathbf{y}_j^l \mathbf{W}_q, \bar{\mathbf{X}}^L \mathbf{W}_k, \bar{\mathbf{X}}^L \mathbf{W}_v$
6:  $\mathbf{e} \in \mathbb{R}^{N'+1} \leftarrow \mathbf{q}\mathbf{K}^T / \sqrt{d}$
    ▷ perform softmax with counts
7:  $\mathbf{a} \leftarrow \mathbf{c} \odot \exp(\mathbf{e}) / \sum_i (c_i \exp(e_i))$
8:  $\mathbf{v} \in \mathbb{R}^d \leftarrow \mathbf{a}\mathbf{V}$
9:  return $\mathbf{v}$
```

4.2 Decoding with $\mathcal{L}_0$Drop

At test time we do not sample gate values but estimate their expected value g_i, following Louizos et al. (2018):

$$\hat{g}_i = \min(1, \max(0, \sigma(\log \alpha_i)(1+2\epsilon) - \epsilon)), \quad (13)$$

which often turns out to be exactly either 0 or 1, albeit being in-between in some cases. Encodings corresponding to non-zero $\hat{g}_i$ are preserved and simply weighted by the expectation $\hat{g}_i$.

To leverage the induced sparse structure, we revise the decoding procedure as in Algorithm 1. The notation $[\cdot, \cdot]$ refers to row-wise concatenation, $[I]$ stands for extracting elements with the indices I, $\odot$ is element-wise multiplication, and $\mathbf{1} \in \mathbb{R}^{N'}$ indicates a vector of ones of length N'. We first reorganize the gates $\hat{\mathbf{g}} \in \mathbb{R}^N$ and the source encodings $\mathbf{X}^L \in \mathbb{R}^{N \times d}$ by eschewing the entries corresponding to closed gates ($\hat{g}_i = 0$, line 1-2). We augment the compressed sequence $\mathbf{X'}^L \in \mathbb{R}^{N' \times d}$ with a dummy zero encoding vector $\mathbf{0} \in \mathbb{R}^d$ to represent all pruned encodings, and record their count into a counting vector $\mathbf{c} \in \mathbb{R}^{N'+1}$ (line 4).[4] Notice that this step is decoder-agnostic, which only relies on the source encodings and $\mathcal{L}_0$Drop gates. We then modify the attention process to enable the inclusion of this counting information (line 5-8) for correctly estimating the attention weights. Note that the shortened source sequence $\bar{\mathbf{X}}^L$ is reused across decoder layers and steps. $\mathcal{L}_0$Drop changes the dependency of the encoder-decoder attention on source sequence from $\mathcal{O}(NM)$ to $\mathcal{O}(N'M)$, and allows for efficiency gains even with moderate sparsity, especially for large L, N and M.

5 Experimental Setup

We study $\mathcal{L}_0$Drop on machine translation tasks (WMT14 English-German (En-De) (Bojar et al., 2014) and WMT18 Chinese-English (Zh-En) (Bojar et al., 2018)[5]) and document summarization tasks (CNN/Daily Mail (Hermann et al., 2015)[6] and WikiSum (Liu et al., 2018)[7]). We adopt BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004) to evaluate the translation and summarization quality, respectively. Other details, including model settings, are given in the Appendix A. For character-based translation, we employ the same model architecture and hyperparameters for training and decoding as specified in Appendix A, except that we adopt larger-batch training (∼85K characters) and encourage longer sequence decoding (length penalty of 1.0).

6 Results and Analysis

How much can encoder outputs be sparsified? We answer this question by analyzing the impact of pruning source encodings on the generation quality. We first train a baseline Transformer model, and then finetune this model using $\mathcal{L}_0$Drop (Eq. (12)) with varied λ to explore different levels of sparsity. We sample λ with a range of $(0, 1.5]$ and a step size of 0.1, and finetune WMT14 En-De and WMT18 Zh-En models for extra 50K steps, and CNN/Daily Mail for extra 20K steps. We use the *sparsity rate* to measure the sparsity; we define it as the ratio of the pruned source encoding number $\#(\hat{g}_i = 0)$ to the total number of source words.

Figure 4 shows the results. The generation quality exhibits a negative correlation with the sparsity rate across different tasks, reflecting the usefulness of encoder outputs for generation. However, the fact that we can remove about 40% source encodings without largely degrading the generation performance (-0.5 BLEU and -0.1 ROUGE-L) supports our hypothesis that we can force Seq2Seq

---

[4] Note that $N' \leq N$. $\mathcal{L}_0$Drop could increase the sequence length if no source encoding is pruned, which is not observed in our experiments.

[5] https://www.statmt.org/wmt14(8)/translation-task.html

[6] `https://github.com/harvardnlp/sent-summary`

[7] `https://github.com/nlpyang/hiersumm`

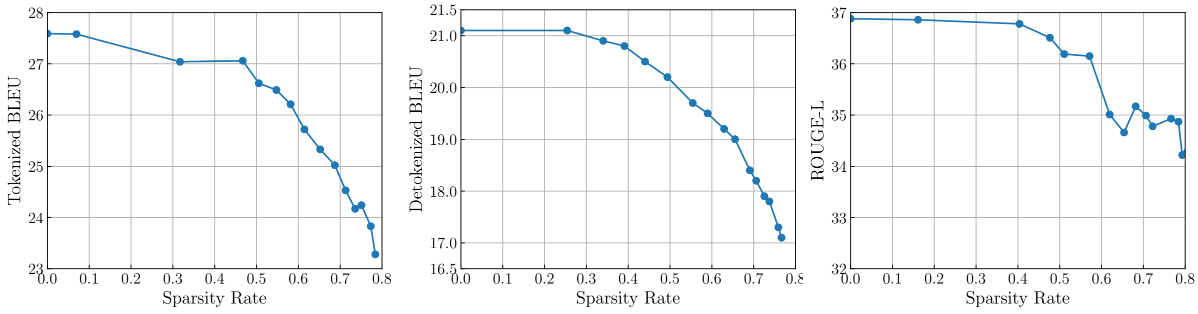

(a) WMT14 En-De (b) WMT18 Zh-En (c) CNN/Daily Mail

Figure 4: Generation quality (BLEU and ROUGE-L, evaluated on test set) as a function of sparsity rate for WMT14 En-De, WMT18 Zh-En and CNN/Daily Mail. Pruning about 40% source encodings results in marginal performance loss on all tasks.

Task	Time	Speedup	Sparsity	Quality
WMT14 En-De	68.89 68.38	1.00× 1.01×	0.00% 46.7%	27.59 27.06
WMT14 En-De (Char)	1418 1186	1.00× 1.20×	0.00% 46.1%	25.40 25.35
WMT18 Zh-En	116.3 118.3	1.00× 0.98×	0.00% 39.1%	21.10 20.80
CNN/Daily Mail	3909 3227	1.00× 1.21×	0.00% 47.6%	36.88 36.51
WikiSum	70505 42669	1.00× 1.65×	0.00% 71.5%	39.20 38.75

Table 1: Decoding results for different tasks when finetuning with $\lambda = 0.3$. "Time": the decoding time (in seconds) of the whole test set. "Sparsity": the sparsity rate, 0.00% indicates the Transformer baseline. "Speedup": the decoding acceleration over the baseline. "Quality": BLEU for WMT tasks and ROUGE-L for summarization tasks. "Char": the character-level models, where we set $\lambda = 0.02$. We evaluate the decoding time on GeForce GTX 1080 Ti, with a batch size of 32 for WMT tasks and 10 for summarization tasks.

model to route information through a subset of its source encodings. We also observe that the compressibility seems relatively language independent (the curves of WMT14 En-De 4(a) and WMT18 Zh-En 4(b) are similar) but clearly task dependent. Compared to translation tasks, the summarization task seems less sensitive to the pruning of source encodings. We ascribe this to the property of summarization where the summary only reflects a part of the input document, rather than the entire document.

Does $\mathcal{L}_0$Drop improve the decoding speed? With appropriate finetuning, $\mathcal{L}_0$Drop can shorten the encoding sequence fed to the decoder, reducing the calculation amount of the encoder-decoder attention. However, the encoder-decoder attention corresponds to about 1/3 of the decoder calculations,[8] and Algorithm 1 also brings in extra overhead, such as gathering and indexing operations. Thus, a speed-up is not guaranteed, and we report empirical decoding time across different tasks.

Results in Table 1 show that $\mathcal{L}_0$Drop only marginally improves the decoding speed for subword-based machine translation, despite a high sparsity rate of 46.7% (WMT14 En-De) and 39.1% (WMT18 Zh-En). By contrast, $\mathcal{L}_0$Drop yields a speedup of 1.21× and 1.65× on CNN/Daily Mail and WikiSum, respectively. One explanation lies at the significant difference in target sequence length, where the average length per summary is >60, compared to ∼25 in machine translation. Note that $\mathcal{L}_0$Drop achieves a substantially higher sparsity rate of 71.5% on WikiSum with the same $\lambda = 0.3$. This is because the input paragraphs overlap in content; the information about redundant words does not need to be routed into other encoder states, making it easier to prune them.

When it comes to character-based translation, we observe that $\mathcal{L}_0$Drop performs comparably to the baseline (-0.05 BLEU) with a high sparsity rate of 46.1%. This echoes with our findings on subword-based translation, except that $\mathcal{L}_0$Drop also improves inference efficiency here due to the higher sequence length with a decoding speedup of 1.20×. In comparison to adaptive compression (Cherry et al., 2018), $\mathcal{L}_0$Drop delivers higher compression ratio with almost no quality loss. This further demonstrates the effectiveness of $\mathcal{L}_0$Drop in handling lengthy inputs. We notice that character-level models still underperform their subword-level counterparts with the standard Transformer, but

---

[8] At decoding, the encoder-decoder attention accounts for about 34.4% decoder time according to profiling on our implementation.

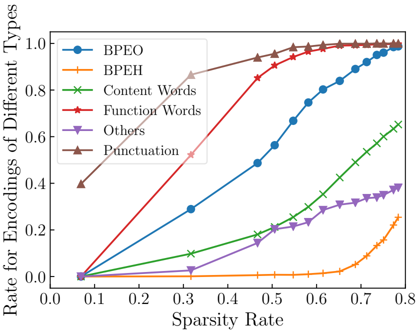

Figure 5: Curves for sparsity rate of different types of encoding on the WMT14 En-De test set. x-axis denotes the overall sparsity rate. The encoding of content words and BPEH is more valuable for generation, compared to that of function words and punctuation.

closing this performance gap is beyond the scope of this study.

Note that the pretraining-then-finetuning schema is mainly used for saving training efforts. By scheduling λ linearly with training steps, we can train models with $\mathcal{L}_0$DROP (Eq. (12)) from scratch, and obtain a BLEU score of 27.03 ($\lambda = 0.2$, warmup step of 200K) on WMT14 En-De (using subwords), comparable to using finetuning (27.04).

What types of source encoding are required for generation? Our goal here it to understand encodings of which types of tokens are retained. For each source encoding, we regard the POS of its corresponding word as its type. We take WMT14 En-De as our benchmark, where we annotate POS for source sentences in the test set using the Stanford POS tagger (Toutanova et al., 2003). We handle subwords separately by labeling its first piece as *BPEH* while the others as *BPEO*, regardless of the POS of its unsegmented form. We group different POS tags into 6 categories for the sake of analysis: BPEH, BPEO, function words, content words, punctuation and the rest.[9]

Figure 5 shows how the sparsity rate of each encoding type changes as a function of the overall sparsity rate. We find that $\mathcal{L}_0$DROP first choose to eliminate the encoding of punctuation, followed by that of function words. These words often signal structural and grammatical relationships that, while important to build up a representation of the sentence, can be easily compressed. In contrast, pruning content words, which express richer lexical meaning, is more difficult. The sharp increase of content word sparsity after the overall sparsity rate of 0.5 in Figure 5 correlates with a sharp drop in translation quality (see Figure 4(a)). We also observe that there is a large difference between BPEO and BPEH, albeit both from the same word. $\mathcal{L}_0$DROP favours to prune the encoding of BPEO, indicating that the model learns to use word-initial representations (BPEH) to represent whole words.

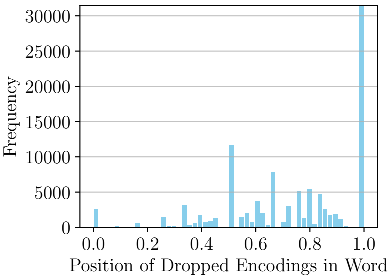

Figure 6: Distribution of the position of pruned source characters within a word on the WMT14 En-De test set. x-axis: 0.0 and 1.0 corresponds to the beginning and ending of one word, respectively.

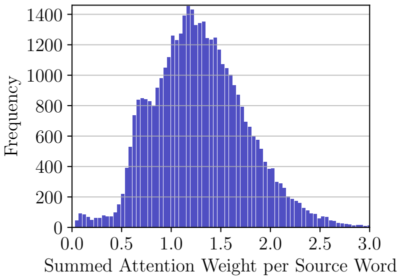

Figure 7: Distribution of the summed attention weight per source word on the WMT14 En-De test set for Transformer with $\mathcal{L}_0$DROP (sparsity rate 47%, BLEU 27.06). Only 4.5% source words get attention weights of less than 0.6.

For character-level models, we observe that $\mathcal{L}_0$DROP identifies the inter-word structure in character sequences, and removes 89.2% encodings of the space symbol. We didn't find any other character-specific pruning patterns, but Figure 6 reveals that the encodings of characters near the ending of a word are more likely to be pruned compared to those at the beginning positions. This result resonates with our above observation on subword pruning, where $\mathcal{L}_0$DROP prefers to drop the encoding of BPEO rather than BPEH.

What's the effect of $\mathcal{L}_0$DROP on Transformer? Transformer can lose the access to around 40% source encodings while largely retaining the same performance. We try to figure out what has changed inside Transformer in order to support $\mathcal{L}_0$DROP,

---

[9] Function words include *CC, IN, RP, TO, UH, DT* and *WP*. Content words include *MD, JJ, NN, RB* and *VB*. Others include other POS tags except for punctuation and BPEO/BPEH, such as *CD, EX, FW* and *SYM*.

(a) Retained Encodings

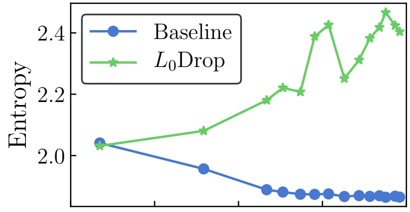

(b) Pruned Encodings

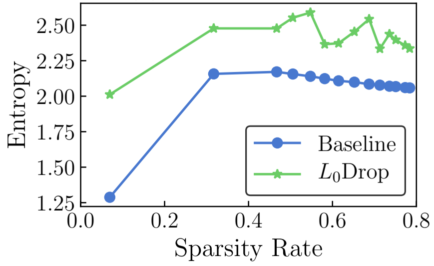

Figure 8: Entropy of the retained source encodings (top) and the pruned ones (bottom) versus the sparsity rate on the WMT14 En-De test set. We use the sparsity variable $\hat{\mathbf{g}}$ learned by $\mathcal{L}_0$DROP to classify the encodings of our baseline Transformer. Higher entropy indicates that the distribution tends to be uniform. With fewer retained encodings, Transformer tends to spread its attention weights to include more source-side information.

and analyze the attention weights (i.e. $\mathbf{A}$ in Eq. (1)) of all encoder-decoder attention sublayers and the last encoder self-attention sublayer; these sublayers are directly connected with $\mathcal{L}_0$DROP in the computation graph. We test on WMT14 En-De.

We visualize the distribution of the encoder-decoder attention weight per source word for Transformer with a sparsity rate of 47% (BLEU 27.06). Compared to the vanilla Transformer (Figure 1), distributions in Figure 7 show that the average attention weight obtained by each source word has increased (+0.77, 1.03→1.80), and the proportion of source words receiving attention weights of less than 0.6 is substantially reduced, by a factor of 10 (49.7%→4.5%). This indicates that $\mathcal{L}_0$DROP forces Transformer to distribute its attention more evenly among retained source encodings.

Apart from the encoder-decoder attention, we also inspect the self-attention in the last encoder layer. We average the self-attention weights over 8 different heads, and compare the attention entropy of the retained source encodings ($\hat{g}_i \neq 0$) and the pruned ones ($\hat{g}_i = 0$). We report average entropy values over the whole test set. Figure 8 shows how increasing sparsity affects the entropy. Although $\mathcal{L}_0$DROP selects to drop uninformative encodings, the increase in the entropy of the retained encodings (Figure 8 (a)), when compared to the baseline, suggests that the encoder actually encodes more context information into these representations, confirming that the model learns to compress context information when sparsity is enforced. Another observation is that the entropy curve of $\mathcal{L}_0$DROP for the pruned encodings is in line with that of the baseline, albeit on a larger scale (Figure 8 (b)). This signifies that $\mathcal{L}_0$DROP enforces Transformer to adapt its attentions to better coordinate with source context representations, which ensures its effectiveness on generation.

Can we prune encodings earlier in the encoder? Rather than stacking $\mathcal{L}_0$DROP on top of the encoder outputs, we insert $\mathcal{L}_0$DROP in-between every adjacent pair of encoder layers. We work on WMT14 En-De and finetune with $\lambda = 0.2$. We get a sparsity rate of 0.0%, 0.0%, 8.6%, 8.6%, 8.7% and 34.0% for the first to the last $\mathcal{L}_0$DROP layer, respectively, with a BLEU score of 26.74. This result suggests that Transformer does not gain much benefit from pruning encodings earlier. The model tends to retain encodings at shallow levels (0.0%/8.6% $<$ 34.0%), and loses 0.3 BLEU compared to its $\mathcal{L}_0$DROP baseline ($\lambda = 0.2$, sparsity rate 31.7%, BLEU 27.04). We believe that the encoder relies on low-level information (including the words) to fully 'understand' the sentence, though part of the final encodings is discardable.

7 Exploring Rule-based Sparse Patterns

Our analysis shows that the sparsity induced by $\mathcal{L}_0$DROP follows certain patterns, with the encodings of 'less content-bearing' words pruned first. This suggests that we may be able to define heuristic patterns manually. In this section, we explore the following three rule-based patterns according to our study on WMT14 En-De:

POS Pattern This pattern discards the source encodings of those easy-to-prune types, including function words, punctuation, BPEO and *MD, EX*, which account for 46.4% of the source-side WMT14 En-De training data.

Freq Pattern Inspired by the fact that punctuation and function words are high frequency words, we propose to filter out the source encodings corresponding to top-frequent words with a threshold of 46.3% (top 100 words). We also include an inverse version, *Inv Freq Pattern*, for comparison, which drops the encodings of most rare words; source words whose fre-

Pattern	WMT14 En-De		CNN/Daily Mail	
	Sparsity	BLEU	Sparsity	RL
Baseline	0.00%	27.59	0.00%	36.88
$\mathcal{L}_0$DROP	46.7%	27.06	47.6%	36.51
POS Pattern	46.7%	27.11	39.6%	35.57
Freq Pattern	42.1%	26.98	47.8%	35.67
Group Pattern	50.0%	26.82	50.0%	30.69
Inv Freq Pattern	44.7%	26.42	39.0%	27.89

Table 2: Sparsity and generation quality for different models on the WMT14 En-De (measured by tokenized case-sensitive BLEU) and the CNN/Daily Mail (measured by ROUGE-L or RL) test set. The sparsity rate is evaluated on test set.

quency ranks lower than 452 are removed, covering ∼40.0% of the source training data.

Group Pattern We explore a position-based pattern that only feeds the encodings at odd positions to the decoder, indicating a sparsity rate of ∼50%. This pattern is partially motivated by Child et al. (2019).

The design of these patterns follows our analysis on $\mathcal{L}_0$DROP, where we match the sparsity rate in each pattern to the optimal rate of $\mathcal{L}_0$DROP on WMT14 En-De. We examine the feasibility of these patterns on WMT14 En-De and CNN/Daily Mail.

Table 2 shows the results. On WMT14 En-De, Transformer using these rule-based patterns achieves comparable translation quality to $\mathcal{L}_0$DROP (-0.24 to +0.05 BLEU) with similar sparsity rate. One interesting observation is that Transformer also works with language- and context-agnostic sparsity patterns (Freq Pattern). The performance drop by Inv Freq Pattern (-0.64 BLEU) is in line with the information-theoretic expectation that information from frequent words is easier to compress than that of rare words.

However, note that we developed our heuristics to mimic the behaviour of $\mathcal{L}_0$DROP for WMT14 En-De task. $\mathcal{L}_0$DROP has the advantage that it is data-driven and task-agnostic so that we can easily apply $\mathcal{L}_0$DROP to summarization. By contrast, these rule-based patterns discovered on translation tasks are not optimal for other tasks, which results in deteriorated performance on CNN/Daily Mail (-5.82 to -0.84 RL). In particular, Transformer suffers from a large performance drop with the Group pattern (-5.82 RL). These results suggest that using rule-based sparse patterns to manually define the sparsity of encoder outputs is possible though the patterns lack generalization ability to different tasks.

8 Conclusion and Future Work

By introducing a L_0-regularized neural sparsity layer ($\mathcal{L}_0$DROP) in Transformer, we confirm that the encoder outputs are compressible to varying degrees. Pruning encoder outputs often results in a drop in performance, but we can get comparable results with 40–70% source encodings dropped. One benefit of pruning source encodings is to shorten encoding sequences for the decoder, which is especially beneficial for efficiency on long sequences, and accelerates the decoding speed by up to 1.65× on document summarization tasks and 1.20× on character-based machine translation. Our analysis on WMT14 En-De shows that $\mathcal{L}_0$DROP learns to drop the encodings of (relatively frequent) function words and retain encodings of (relatively rare) content words, but relies on self-attention to reroute information from these to-be-pruned positions. Based on our analysis, we define rule-based sparsity patterns, which also allow for compression without degrading translation quality much, and show that frequent tokens are more amenable to sparsification than rare tokens. However, we find that our rule-based patterns do not generalize across tasks, while $\mathcal{L}_0$DROP is data-driven and applicable across tasks. We hope that, besides practical implication, our work contributes to better understanding encoder-decoder models.

For future work, we find that the sparsity induced by $\mathcal{L}_0$DROP is highly task-dependent and hardly manipulated. We will develop novel algorithms to make the sparsity induction more controllable.

Acknowledgments

We thank the reviewers for their insightful comments. This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements 825460 (ELITR) and 825299 (GoURMET). Rico Sennrich acknowledges support of the Swiss National Science Foundation (MUTAMUR; no. 176727). Ivan Titov acknowledges support of the European Research Council (ERC StG BroadSem 678254).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3539–3544, Hong Kong, China. Association for Computational Linguistics.

Yingqiang Gao, Nikola I. Nikolov, Yuhuang Hu, and Richard H.R. Hahnloser. 2020. Character-level translation with self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1591–1604, Online. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Jindřich Libovický and Alexander Fraser. 2020. Towards reasonably-sized character-level transformer NMT by finetuning subword systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2572–2579, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3338–3348. Curran Associates, Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335, Florence, Italy. Association for Computational Linguistics.

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, Deyi Xiong, and Lei Li. 2019. Towards linear time neural machine translation with capsule networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* *and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 803–812, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium. Association for Computational Linguistics.

J. Zhang, Y. Zhao, H. Li, and C. Zong. 2019. Attention with sparsity regularization for neural machine translation and summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3):507–518.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.

A Experimental Settings

Machine Translation We train translation models on the WMT14 English-German translation task (En-De) (Bojar et al., 2014) and the WMT18 Chinese-English translation task (Zh-En) (Bojar et al., 2018). WMT14 En-De and WMT18 Zh-En contain around 4.5M and 25M training sentence pairs, respectively. We use newstest2013 as the validation set for WMT14 En-De and newstest2017 for WMT18 Zh-En. We evaluate the translation quality with BLEU metric (Papineni et al., 2002), and report tokenized BLEU on newstest2014 for WMT14 En-De and detokenized BLEU on newstest2018 for WMT18 Zh-En using *sacreBLEU* (Post, 2018). We apply the byte pair encoding (BPE) algorithm (Sennrich et al., 2016) with 32K merging operations to handle rare words for both translation tasks.

Document Summarization We train abstractive summarization models on the CNN/Daily Mail dataset (Hermann et al., 2015) and the WikiSum dataset (Liu et al., 2018) for single- and multi-document summarization task, respectively. We use the non-anonymized version of CNN/Daily Mail (Gehrmann et al., 2018). We pre-process this dataset with a BPE vocabulary of 32K and truncate each article to 400 subwords (Gehrmann et al., 2018). We use the ranked version of WikiSum (Liu and Lapata, 2019), where top-40 paragraphs are extracted for each instance paired with a summary of 121 words on average. We concatenate all these paragraphs into one source sequence following the given ranking order. CNN/Daily Mail pairs news articles (791 words on average) with multi-sentence summaries (63 words on average), and involves 287,227 training pairs, 13,368 validation pairs and 11,490 test pairs. WikiSum contains 1.58M training pairs, 38,144 validation pairs and 39,357 test pairs. We employ BPE preprocessing following Liu and Lapata (2019) and truncate each source sequence to 2048 subwords. We evaluate the summarization quality using the F_1 score of ROUGE-L (Lin, 2004). The used parameters for ROUGE-1.5.5.pl are *-m -a -n 2*.

Model Settings We formulate all the tasks as sequence-to-sequence tasks, and experiment with the base setting of Transformer (Vaswani et al., 2017): $d = 512$, the middle layer size of $\textsc{Ffn}(\cdot)$ is 2048, and the number of attention head is 8. Following Louizos et al. (2018), we set $\epsilon = -0.1$, and $\beta = {}^2/_3$ for $\mathcal{L}_0\textsc{Drop}$. We tune the hyperparameter λ for different tasks. We augment the MLE loss with label smoothing of 0.1. We use Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.98$) (Kingma and Ba, 2015) for parameter tuning, and schedule the learning rate based on the inverse square root of running steps with a warm-up step of 4K. We apply dropout to attention weights and residual layers to avoid overfitting, with a rate of 0.1/0.1 except for CNN/Daily Mail where 0.3/0.5 is used. We train different models with varied training steps: 300K for WMT14 En-De, 500K for WMT18 Zh-En, 100K for WikiSum and 80K for CNN/Daily Mail, where sequence pairs of roughly 25K target subwords are organized into one minibatch. We average the last 5 checkpoints for evaluation where beam search is adopted for decoding with beam size of 4 and length penalty of 0.6.

FrameNet-assisted Noun Compound Interpretation

Girishkumar Ponkiya[†], **Diptesh Kanojia**[‡], **Pushpak Bhattacharyya**[†]
[†]Indian Institute of Technology Bombay, Mumbai
[‡]Centre for Translation Studies, University of Surrey, United Kingdom
{girishp,diptesh,pb}@cse.iitb.ac.in

Girish K Palshikar
TCS Research, Tata Consultancy Services, Pune
gk.palshikar@tcs.com

Abstract

Given a noun compound (NC), we address the problem of predicting the appropriate semantic label linking the constituents of the NC. This problem is called *Noun Compound Interpretation (NCI)*. We use FrameNet as a semantic label repository. For example, given the noun compound (*board approval*), we predict the frame (DENY_OR_GRANT_PERMISSION, as per FrameNet) as appropriate and the semantic role of the modifier word (AUTHORITY) as the semantic label linking *board* and *approval*; the resulting label is DENY_OR_GRANT_PERMISSION:AUTHORITY.

Our semantic label repository is very large ($\approx$ 11k labels) compared to the NC data available for training (approx 1900). Thus, learning in this case, especially for unseen semantic labels, is hard. We propose to solve this problem by predicting semantic labels in a *continuous label embedding space*, which is novel. This embedding space is created by learning label embeddings using the FrameNet data. The embeddings are then used to train two separate models – one for predicting Frames and the other for FEs. As the label embedding space captures the semantics of the labels, using these embeddings enables generalizing well on unseen labels, thus achieving zero-shot learning. Our preliminary investigations show that the proposed approach performs well for unseen labels, achieving 5% and 2% points improvements over baselines for the frame and FE prediction, respectively. The study shows the promise of the use of continuous space embeddings for noun compound interpretation and points to the need for further investigation.

1 Introduction

A noun compound is a sequence of two or more nouns that act as a single entity with well-defined meaning (*e.g.*, *paper submission*, *colon cancer*, *etc.*). Semantic relations between the component nouns are implicit. For instance, the information that 'it is a *juice* made from *orange*' is hidden in *orange juice*. Uncovering this semantic relation is called the problem of *Noun Compound Interpretation (NCI)*. NCI needs ML, as the task faces the challenge of ambiguity, and disambiguation by rules is well nigh impossible because of multifarious complex underlying language phenomena. The proposition of storing NCs and doing table lookup for interpretation is also impractical due to a large number of NCs and the challenge of high productivity (new nouns and NCs get created frequently, *e.g.*, *corona vaccine* is a relatively new NC).

Often, the exact relation, sentiment, *etc.* are also governed by contextual pragmatics. For instance, the sentiment towards *tax money* depends on who the beneficiary is, which again depends on the predicate. The predicate *give* could indicate negative sentiment (for the tax-payer), whereas the predicate *receive* would indicate positive sentiment (for the government). Due to such instances, NLP tasks such as machine translation (Baldwin and Tanaka, 2004; Balyan and Chatterjee, 2015), textual entailment (Nakov, 2013), question answering (Ahn et al., 2005), *etc.* suffer when they encounter noun compounds. For example, from the below text-question pairs, a system would need to interpret the underlying semantics within the compound, to answer the question correctly.

(a) *"student protest"*: *"who is protesting?"*,

(b) *"fee-hike protest"*: *"why protest?"*, and

(c) *"university protest"*: *"where is the protest?"*

In this work, we interpret only *compositional* noun-noun compounds. A noun-noun compound is categorised as compositional, if the meaning of the compound can be composed from the semantics of the individual noun units present.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2901–2911
August 1–6, 2021. ©2021 Association for Computational Linguistics

From a relation representation perspective, noun compounds are interpreted in two ways: via *labelling* and *paraphrasing*. Labelling involves assigning an abstract semantic relation from a predefined set, for example, *orange juice*: MADEOF, *hillside home*: LOCATION, *etc.* There are many inventories of predefined semantic relations.

We use the FrametNet based labels proposed by Ponkiya et al. (2018a). As per their convention, the head noun of a compound invokes the frame, and the modifier noun fits in one of the frame elements of the invoked frame, *vide 'board approval'* in the abstract.

There are more than 11,000 FEs in FrameNet, and we have about 1900 training examples. Thus, the average number of examples for each label is quite small, and many labels do not have a training example. In summary, the contributions of this paper are three-fold:

1. We embed FrameNet entities in a continuous space, perform prediction in the continuous space to generalize over unseen labels, and show performance improvement on the unseen labels.
2. We create a noun-compound annotation tool that assists annotators in providing manual labels, and we release it publicly.
3. Using the above tool, we extend the dataset released by Ponkiya et al. (2018a) with 326 more manually-annotated gold samples, and release it for further research.

The rest of the paper is organized as follow: Section 2 discuss related work, Section 3 gives an overview of foundations for the work. Section 4 details our approach. Section 5 provides experimental details: the dataset used and training/testing setup. Section 6 discusses the results and analysis, followed by a conclusion and future work. The code, dataset and the tool can be downloaded from `http://www.cfilt.iitb.ac.in/nc-dataset`.

2 Related Work

A relation between the components of a noun compound (say, *chocolate cake*) can be represented in one of the following two ways: (1) assigning a relation from a predefined set of semantic relations (MADEOF), or (2) using a paraphrase to convey the underlying semantic relation (*"cake made using chocolates"* or *"cake with chocolate flavor"*).

Noun-compound (NC) interpretation via **labelling** is the most commonly used methodology for NC interpretation. Scholars have proposed many inventories of semantic relations (Levi, 1978; Warren, 1978; Vanderwende, 1994; Lauer, 1995; Barker and Szpakowicz, 1998; Ó Séaghdha, 2007; Rosario et al., 2001; Tratz and Hovy, 2010; Fares, 2016; Ponkiya et al., 2018a). A recent FrameNet-based inventory by Ponkiya et al. (2018a) proposed FEs (Frame Elements) from FrameNet as labels (or, semantic relations). They released a dataset by annotating each noun compound with a frame and a frame element; and proposed this annotation for predicate 'nominalization'. However, it also works for most of the cases of 'predicate deletion'.

For automatic labelling, Dima and Hinrichs (2015) and Fares et al. (2018)'s architecture is similar to ours. Dima and Hinrichs (2015) proposed a feed-forward neural network-based approach. This network takes concatenated embeddings of component nouns as an input and predicts one of the labels from the Tratz and Hovy (2010)'s label set. Fares et al. (2018) used a similar feed-forward network to predict two types of relations. This network, however, shares initial layers and separates output layers for each label type.

NC interpretation via **paraphrasing** is another methodology that contains approaches such as prepositional and free paraphrasing. Prepositional paraphrasing, *i.e.*, paraphrasing using a preposition, for example, *student protest*: "*protest by student(s)*", is a relatively well-attended problem (Lauer, 1995; Lapata and Keller, 2004; Ponkiya et al., 2018b). All the above approaches proposed for prepositional paraphrasing use the fixed-set of eight prepositions proposed by Lauer (1995). The other set of approaches, *i.e.*, free paraphrasing, however, has not received much attention. Apart from two SemEval tasks (Butnariu et al., 2009; Hendrickx et al., 2013), it does not have much literature available. A recent study (Ponkiya et al., 2020) expresses paraphrasing as a "fill-in-the-blank" problem, and utilizes pre-trained language models, for the task of noun-compound interpretation.

3 Foundations

Levi (1978) performed a linguistic study to understand how noun compounds are generated. They call such compounds *nominal compounds*. This theory puts nominal compounds into two categories, based on the compounding process, as

1. **Predicate Deletion:** Here, a predicate between the components is dropped to create a compound.

   For example, *apple pie* is a "pie made from apple." The predicate made from is dropped in this case. Similarly, for *elbow injury*, *gas pipeline*, *etc*.

2. **Predicate Nominalization:** Here, the head noun is a nominalized form of a verb, and the modifier is an argument of the verb.

   For example, "The union demonstrated against the price hike..." becomes "The union demonstration against the price hike..."

   Verbal noun as head: *student demonstration*, *government approval*, *opposition objection*, *etc*.

   Verb form as head: *student protest*, *government support*, *competition schedule*, *etc*.

Levi (1978) also proposed a set of abstract predicates[1] for the former category, but no relation for the latter category. Later, Ó Séaghdha (2007) revised this inventory and proposed a two-level hierarchy of semantic relations.

Ponkiya et al. (2018a) proposed a method to use FrameNet based labels for noun compounds. Here, the head noun invokes a frame, and the modifier noun fits in one of the slots of the frame. They also prepared a dataset by annotating each noun compound with a frame and a frame element. Ponkiya et al. (2018a) proposed this annotation for predicate nominalization, which also works for most cases of predicate deletion.

3.1 FrameNet

FrameNet[2] (Baker et al., 1998) is a taxonomy based on Fillmore's theory of Frame Semantics. This theory claims that most words' meanings can be inferred based on a semantic frame: a conceptual structure that denotes an abstract event, relation, or entity and the involved participants. For example, the concept of questioning involves a person asking a question (Speaker), person/people begin questioned Addressee, the content of the question Message, and so on. In FrameNet, such a concept is represented by Questioning frame. The participating entities, such as Speaker, Addressee, Message, *etc*., are called *frame elements* (FEs). Such frames are invoked in running text via words known as *lexical units*. Some of the lexical units for the Questioning frame are *ask, grill, inquire, inquiry, interrogate, query*, *etc*. FrameNet data provides two types of linkages between entities:

(a) relations: linking among frames or among FEs, and

(b) mappings: linking from words to frames and from frames to FEs.

3.1.1 Relations

FrameNet includes a graph of relations between frames along with relations among frames. Some of the important frame relations are:

- **Inheritance:** close to a typical Is-A relation, *e.g.*, Protest $\xrightarrow{\text{Is-A}}$ Intentionally_act
- **Using:** the child frame presupposes the parent frame, *e.g.*, Protest $\xrightarrow{\text{Uses}}$ Taking_sides
- **Subframe:** the child frame is a subevent of a complex parent event, *e.g.*, Trial $\xrightarrow{\text{Subframe}}$ Verdict

Along with each frame relation, FrameNet also consists of relations between FEs of parent-child frames. Following are illustrative examples for FE relations for the above frame relations:

- Protest:Protester $\xrightarrow{\text{Is-A}}$ Intentionally_act:Agent.
- Protest:Protester $\xrightarrow{\text{Uses}}$ Taking_sides:Cognizer
- Trial:Judge $\xrightarrow{\text{Subframe}}$ Verdict:Judge

3.1.2 Mappings

FrameNet data provides two types of mappings: (a) words to frames (via lexical units), and (b) frames to FEs. For instance, *protest* word can invoke three frames: Protest, Political_actions, and Judgment_communication. Similarly, using the frame to FEs mapping, we can list all FEs of the Protest frame, which are Action, Issue, Protester, Side, Degree, Descriptor, Duration, Explanation, Frequency, Manner, Means, Place, Purpose, Time, *etc*.

---

[1] RDP (Recoverable Deleted Predicates)

[2] `https://framenet.icsi.berkeley.edu`

In our work, we utilize *Relations* for the generation of frame and frame element embeddings (§3.2). We, further, utilize the *Mappings* to prune the search space (§4.1).

3.2 Knowledge Graph Embeddings

A Knowledge Graph $\mathcal{G}$ is a set of relations $\mathcal{R}$ defined over a set of entities $\mathcal{E}$. Formally, it is comprised of a set of N triples (h, r, t), where h and t are called *head* and *tail* entities, and r denotes a relation among them.

$$\mathcal{G} = \{(h, r, t) : h, t \in \mathcal{E}; r \in \mathcal{R}\}$$

Knowledge Graphs are widely used to store knowledge in a structured format, and they play an important role in representation learning. Methods for learning representations for both entities $\mathcal{E}$ and relations $\mathcal{R}$ have been explored (Wang et al., 2017) with an aim to represent graphical knowledge.

Various algorithms for representation learning have been proposed, which help tasks such as link prediction *etc*. TransE (Bordes et al., 2013) is a method that models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities. We use the ConvE (Dettmers et al., 2018) algorithm to get embeddings of frames and frame elements. For the training of ConvE, we treat all relations from FrameNet as triples of a knowledge graph.

3.2.1 ConvE

Convolution-based **E**mbeddings is a multi-layer 2D-convolution network model proposed by Dettmers et al. (2018). It usages fewer parameters, yet efficient compared to similar models. It defines the scoring function (for each relation r) as follows:

$$\psi_r(e_h, e_t) = f(vec(f([\overline{e}_h; \overline{e}_r]w))W)e_t \qquad (1)$$

where, e_h, e_r and e_t are embeddings of head h, relation r and tail t, respectively, $\overline{x}$ denotes reshaping of vector x to a matrix, f is a rectified linear unit (relu) function, vec converts a matrix into a flat vector, w is convolution kernel, and W is the parameter of a fully connected layer.

For training, it applies logistic sigmoid function $\sigma(\cdot)$ to the scores, and minimize the binary cross-entropy computed using the following formula:

$$L_{CE} = -\frac{1}{N}\sum_i (t_i \cdot \log(p_i) + (1 - t_i) \cdot \log(1 - p_i)) \qquad (2)$$

where, $p = \sigma(\psi_r(e_h, e_t))$ and t is 1 when $(h, r, t) \in \mathcal{G}$, 0 otherwise.

ConvE uses two embedding layers: one for entities and the other for relations, which initializes the embeddings layers randomly. The embeddings layers get updated during the training. At the end of the training, the embedding layers contain the embeddings for entities and relations.

4 Our Approach

FrameNet has 1223 frames and 11,473 frame elements. However, the existing dataset for FrameNet-based noun compound interpretation does not have examples for many frames and frame elements. However, unlike other relation inventories, we have FrameNet taxonomy, which can help in building a better model. We first explain our frame prediction approach and then extend the same for FE prediction.

4.1 System Architecture

We encode a given noun compound $nc = w_1\ w_2$ (say, *divorce rate*) using a feed-forward network to get vector v_{nc}. Using FrameNet API, we create a set of candidate frames that can be invoked by w_2 (*rate* → {Assessing, Proportion, Speed_description, etc.}). For each candidate frame f_i, we take its frame embedding e_{f_i} from the frame embedding layer. We take the dot product of v_{nc} with embedding e_{f_i} of each frame f_i to compute the score for the frame. For testing, we use the following formula to predict a frame:

$$f^* = \text{argmax}_i\ v_{nc} \cdot e_{f_i} \qquad (3)$$

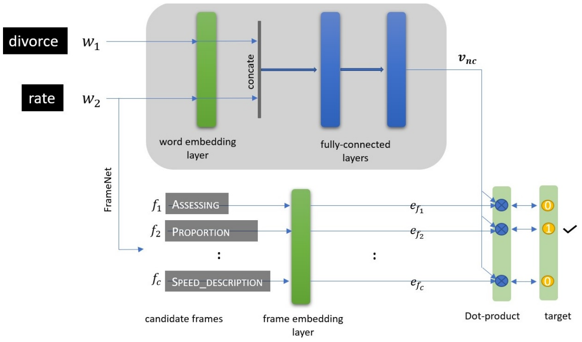

Figure 1: Basic system architecture illustrating frame prediction for *divorce rate*.

Dima and Hinrichs (2015) and Fares et al. (2018) use a simple feed-forward network. In our model, if we remove the frame/FE embedding matrix and use them as weights of one more dense layer (after the "fully-connected layers"), *our model becomes identical to theirs*. In doing so, **(a)** our model will NOT need any extra computation to compute the score of labels that are NOT part of the candidate set, and **(b)** the back-prorogation does not have to pass through an additional layer which might not be effective.

We implement these models in PyTorch (Paszke et al., 2017). We initialize the word embedding layer with Google's pre-trained embeddings[3] and initialize the frame embedding layer with random values, in one case, for baseline, and pre-trained frame embedding, in another case.

We use the same architecture to train another model for FE prediction, replace the frame embedding layer with an FE embedding layer, and candidates FEs are the FEs from all candidate frames. We take all FEs as a candidate set if no such mapping is found.

4.2 Frame and Frame Element Embeddings

Inspired by Kumar et al. (2019)'s approach for the task of Word Sense Disambiguation (WSD), we propose a similar approach to perform NC interpretation. Our approach uses the definition of entities (along with the relations) to learn entity embeddings and relation embeddings. It uses an encoder (Bi-LSTM) to encode the definition of an entity and uses encoded representation as an embedding of the entity for ConvE. During the training, it also optimizes both: the encoder and ConvE. After the training, the encoding of definitions is taken as entity embeddings.

We train ConvE twice to get frame and frame element embeddings separately. ConvE training is independent of the main training.

5 Experimental Setup

In this section, we explain our dataset, baseline, training, and evaluation metrics.

5.1 Dataset Creation and Analysis

We use the dataset released by Ponkiya et al. (2018a) as D1. The dataset contains 1546 noun-noun compounds with two labels: frame and FE. The dataset was created by extracting noun compound along with labels from the FrameNet data. As the extraction is automatic and the manual step only confirms the correctness of the labelling, the labels are not exhaustive. For instance, a noun compound *student demonstration* has been annotated with PROTEST:PROTESTER. However, the following labels are also applicable: REASONING:ARGUER and CAUSE_TO_PERCEIVE:ACTOR. So, we annotate more examples with all possible labels.

Manual Annotation

We manually annotate 326 noun compounds, and call it D2. We extend D1 by merging these examples from D2 to perform our experiments. The annotation is performed by one of the authors and hence does not warrant discussion on the inter-annotator agreement. However, please allow us to point out that our annotations are still manually performed by a human, which begets the consideration of these annotations to be gold-standard. The author chose the examples from Tratz and Hovy (2010)'s dataset randomly. During the annotation process, we found some difficulties because of the coverage issue of the FrameNet. The word-to-frame mapping in FrameNet has a coverage issue, and it has been widely reported in the literature (Pavlick et al., 2015; Botschen et al., 2017).

We categorize the coverage issues into the following:

No Candidate Frames: The word-to-frame mapping returned no candidate frame. In some cases, we could find a frame with manual effort (ref. Table 1). However, despite manual efforts, some cases, we could not find an appropriate frame all the time (*e.g.*, *star autograph*, *employee misconduct*, *etc.*).

No Suitable Frame in the Candidate Set: In this set, word-to-frame mapping retrieved candidate frames, but none of the candidates was found to be appropriate. For example, candidate frames for *heat* returned by the mapping are: CAUSE_TEMPERATURE_CHANGE

Word	Manually extracted frames
funding	FUNDING
authorities	LEADERSHIP, AUTHORITY
eradication	REMOVING
harvesting	FOOD_GATHERING
analyst	PEOPLE_BY_VOCATION

Table 1: Examples where FrameNet data does not contain an appropriate mapping but we manually find suitable frames.

---

[3] https://code.google.com/archive/p/word2vec/

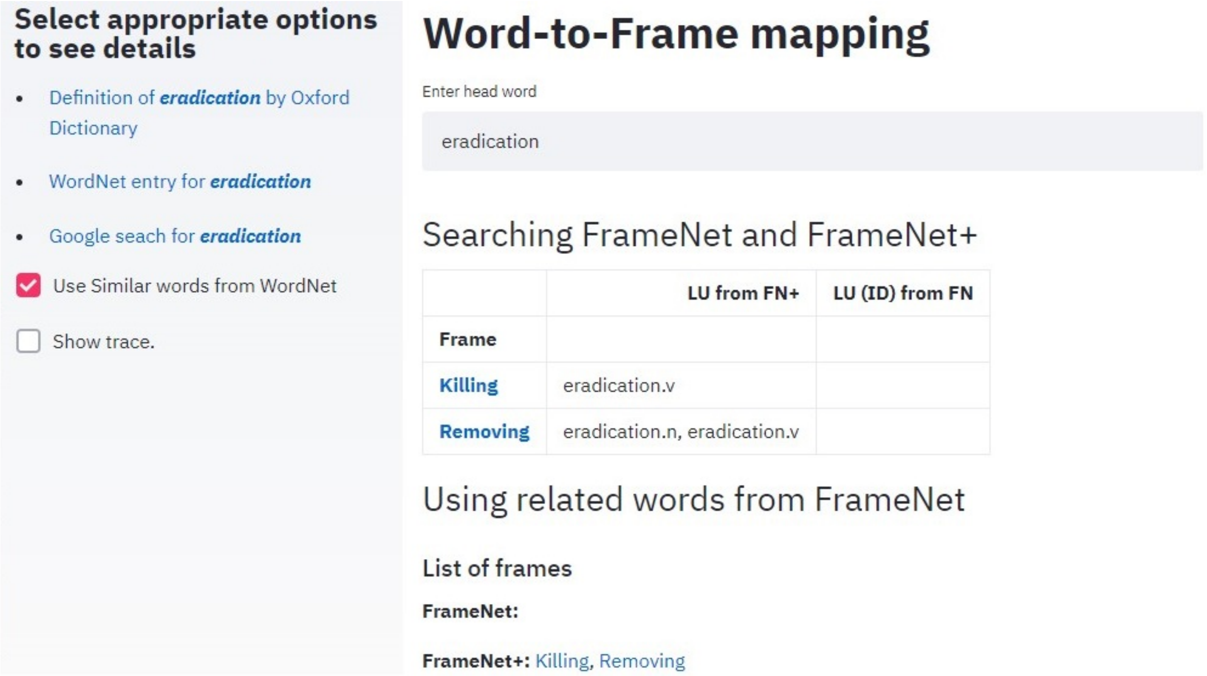

Select appropriate options to see details

- Definition of ***eradication*** by Oxford Dictionary
- WordNet entry for ***eradication***
- Google seach for ***eradication***

☑ Use Similar words from WordNet

☐ Show trace.

Word-to-Frame mapping

Enter head word

eradication

Searching FrameNet and FrameNet+

	LU from FN+	LU (ID) from FN
Frame		
Killing	eradication.v	
Removing	eradication.n, eradication.v	

Using related words from FrameNet

List of frames

FrameNet:

FrameNet+: Killing, Removing

Figure 2: Tool interface to assist annotator in finding a frame. FrameNet does not have mappings for *eradication*, however FrameNet+ maps it to Killing and Removing frames.

and Change_of_temperature. However, none of the two frames is appropriate for *body heat*.

In some cases, we could find an appropriate frame that was not a part of the candidate set. For example, for noun compound *ulcer drug*, candidate frames are Intoxicants and Cause_harm, but the appropriate frame is Cure.

No Suitable Frame Element in the Frame: We could find an appropriate frame, but no frame element from the frame is appropriate. For instance, the Business frame is suitable for *retail operation*, but no frame element from the frame is suitable for the modifier noun *retail*.

NC Annotation Tool

To handle the first two cases (finding of a frame), we use synonyms from WordNet (Miller, 1994) and FrameNet+ data (Pavlick et al., 2015). To simplify the annotation process, we develop a tool (Figure 2) that makes the annotation process easier.

We split each dataset – D1 and D1+D2 – randomly for 5-fold validation. Each fold contains three disjoint sets: training set (60% compounds), validation set (20% compounds), and test set (20% compounds). We use the same folds across all experiments, so results across different models are comparable.

5.2 Frame and Frame Element Embeddings

To get frame embeddings, we consider frames as entities and frame relations from FrameNet as relations between the entities. Then we train ConvE (§4.2) to learn frame embeddings. We use these entity embeddings to initialize the frame embedding layer. Table 2 shows the ten most similar frames for Friendly_or_hostile frame based on cosine similarity between frame embeddings. Similarly, we get embeddings of frame elements using frame elements and their relations in FrameNet.

5.3 Baseline

The first baseline is a **random prediction**: the probability of predicting a label from a candidate set is uniform. We take expected counts to compute metrics. For instance, we compute random accuracy using the following formula:

$$\text{Acc}_{\text{random}} = \frac{1}{N} \sum_{nc_i \in \text{Test-set}} \frac{1}{|\text{candidates}(nc_i)|} \quad (4)$$

We also use **Support Vector Machines** (SVM) (Cortes and Vapnik, 1995) as a baseline approach for this task. We use the *sklearn* library (Pedregosa et al., 2011) with default parameters for

#	Score	Name	Definition
0	1.0000	FRIENDLY_OR_HOSTILE	Two parties or individuals are on the same sid...
1	0.9763	BE_IN_AGREEMENT_ON_ACTION	Two (or more) people (the Parties, also encoda...
2	0.9749	MAKE_AGREEMENT_ON_ACTION	Two (or more) people (the Parties, also encoda...
3	0.9748	DISTRIBUTED_POSITION	This frame involves a static (primarily spatia...
4	0.9743	TERMS_OF_AGREEMENT	A condition that is set out in an Agreement im...
5	0.9742	WHOLES_AND_PARTS	A Whole is made up of multiple Parts, which th...
6	0.9737	MAKE_COMPROMISE	Two (or more) people (the Parties, also encoda...
7	0.9729	SOCIAL_EVENT_INDIVIDUALS	This frames describes a social event where the...
8	0.9725	CO-ASSOCIATION	Two or more individuals have a relationship by...
9	0.9725	PUBLIC_SERVICES	This frame concerns permanent organizations (t...
10	0.9716	BE_IN_AGREEMENT_ON_ASSESSMENT	The Cognizers have a similarity (or dissimilar...

Table 2: 10 most similar frames for FRIENDLY_OR_HOSTILE frame using frame embeddings.

this approach. The input for the SVM-based approach is the concatenated vector of individual lexical units.

We provide results for **another baseline** approach where we use the same architecture (§4.1) with random initialization for frame/FE embeddings.

5.4 Training

Given a noun compound, we get candidate labels using FrameNet mapping. We compute scores for candidate labels and compare them with the target to compute loss value. We minimize categorical cross-entropy with stochastic gradient descent (with momentum). Frame/FE embedding layer remains fixed (non-trainable) for the initial few epochs. For stopping criteria, we monitor performance on the validation set.

5.5 Evaluation

We report weighted **P**recision, **R**ecall, and **F1**-score for our experiments. The weight values for each label is in proportion to the number of test examples for the label. Following is a formula for computing (weighted) precision:

$$\text{Precision} = \sum_{l} P_l * \frac{N_l}{N} \tag{5}$$

$$P_l = \frac{\text{TP}_l}{\text{TP}_l + \text{FP}_l} \tag{6}$$

where, P_l is the precision score, TP_l is the number of true-positives and FP_l is the number of false-positives for a label l. N_l is the number of instances with label l in the test set, and N is the total number of instances in a test set.

The above metrics are based on the top prediction. We also report accuracy at k, which treats a prediction as a true prediction if the correct label is in top k predicted labels. We report (micro-averaged) accuracy, computed using the following formula:

$$\text{Acc} = \frac{\text{No. of correctly classified instances}}{\text{Total instances in test-test}} \tag{7}$$

We compute all of these metrics for frame and frame element independently. We use the Scikit-learn library (Pedregosa et al., 2011) to compute all of these metrics.

6 Results and Analysis

The reported results are averaged across 5-fold cross-validation. We define a subset of all test samples, whose output label does not have any samples in the training set, as *unseen*-set. In a fold (of D1) with 310 test samples, the following are the statistics about the "unseen set": (1) 30 unique frames, covering 32 test samples, (2) 75 unique FEs, covering 82 test samples. *These cases are challenging to handle.* Our prediction in continuous space helps in such cases, as the target space embeds the labels.

Table 3 reports the performance of the baselines compared to our system for frame prediction on the entire test-set. Our system beats the random baseline and SVM model across all metrics by a significant margin. Similarly, as observed from Table 4, it can be seen that frame prediction on the "unseen" set improvises over the random baseline in both the cases, *viz.,* WITH and WITHOUT frame embeddings. The performance improvement is quite significant over both the datasets (D1 and

	D1			D1+D2		
	P	**R**	**F**	**P**	**R**	**F**
Random baseline	23.99	53.04	33.04	24.88	53.29	33.92
SVM	64.45	69.45	64.81	66.64	71.86	67.16
WITHOUT frame embedding	76.81	77.13	**75.70**	78.13	78.26	**76.67**
WITH frame embeddings	77.05	76.90	75.68	78.67	78.14	76.59

Table 3: Performance of our approach for *frame prediction* compared to the baseline approach on **test set** from D1 and D1+D2 datasets. (**P**: Precision@1; **R**: Recall@1; **F**: F1-score@1)

	D1			D1+D2		
	P	**R**	**F**	**P**	**R**	**F**
Random baseline	17.68	51.83	26.37	23.08	47.88	31.15
WITHOUT frame embedding	40.38	38.42	39.07	41.29	42.42	40.82
WITH frame embeddings	46.83	43.75	**44.63**	48.25	44.98	**45.78**

Table 4: Comparison of our approach for *frame prediction* with random prediction baseline on examples with **unseen-set**, achieving zero-shot learning **for frame prediction**. (**P**: Precision@1; **R**: Recall@1; **F**: F1-score@1)

D1+D2). We attribute the performance improvement in both cases to the fact that our model captures frame semantics for a frame in a continuous space, better than the baseline metrics. However, as seen in Table 3, the model which tries to predict WITHOUT frame embeddings shows comparable results with the model which predicts WITH frame embeddings. This marginal improvement does not seem to be significant, and the results for both these cases are almost similar. Eventually, with the improvement in dataset size, we expect the system with frame embeddings to perform better than the system without frame embeddings.

We discuss the task of frame prediction above and present our results for frame element prediction here. As it can be observed from Table 5, random baseline and SVM based models are outperformed for the task of frame element prediction as well, when compared with the results of our methodology. The improvements over both datasets (D1 and D1+D2) are at least 5% (D1+D2/SVM). With the extended dataset (D1 + D2), the performance of our approach shows an improvement across all three (precision, recall and f-score) measures. In Table 3, for frame prediction, we observe an increase in the stronger baseline score (SVM) when the extended dataset is used. However, we observe that our results for frame element prediction show that the model which uses frame embeddings is significantly outperformed by the model which does not use frame embeddings. Upon manual analysis of our train set, we find that our datasets have multiple cases where the number of examples per frame element is very few (sometimes even 1, as discussed below). This results in a data skew where the sample would either be used for training or testing, thus rendering the model either untrained for that test case, or no testing of the model trained for that single frame element. In Table 6, we see that the random baseline outperforms our method because of the data skew discussed here. There are multiple frame elements that are present in the unseen test data for which the model has not been trained at all. We do not report SVM performance in Table 6 and Table 4, since the precision, recall, and f-score for SVM were all 0. This performance can be attributed to the fact that SVM does not perform well with unseen examples, and in this case, does not perform at all.

For frame element prediction, in Table 5, we observe that the extended dataset helps improve the over quality of predictions with an improved score for each approach, including the baseline. These results signify that the dataset extension does indeed help the task of NC interpretation.

In Figure 3, we see that the average number of candidate FEs for a test sample is only 26. Without even a single training example, our system is able to correctly predict 51.22% (more than half) of unseen samples among the top-7 predictions, which is higher than the top-1 accuracy of any baseline approach on the complete test-set. With an increase

	D1			D1+D2		
	P	R	F	P	R	F
Random baseline	13.75	9.68	11.36	14.48	10.20	11.97
SVM	43.99	49.13	43.73	46.55	50.85	45.14
WITHOUT frame element embedding	49.42	52.14	**49.28**	51.13	54.08	**50.29**
WITH frame element embeddings	48.18	48.50	46.61	50.47	50.23	49.73

Table 5: Performance of our approach for *frame element prediction* compared to the baseline approach on **test set** from D1 and D1+D2 datasets. (**P**: Precision@1; **R**: Recall@1; **F**: F1-score@1)

	D1			D1+D2		
	P	R	F	P	R	F
Random baseline	13.28	10.35	**11.64**	15.36	9.26	**11.56**
WITHOUT frame element embedding	2.43	2.33	2.36	3.45	2.53	2.67
WITH frame element embeddings	5.05	4.96	4.70	5.23	5.23	5.28

Table 6: Comparison of our approach for *frame element prediction* with random prediction baseline on examples with **unseen-set**. (**P**: Precision@1; **R**: Recall@1; **F**: F1-score@1)

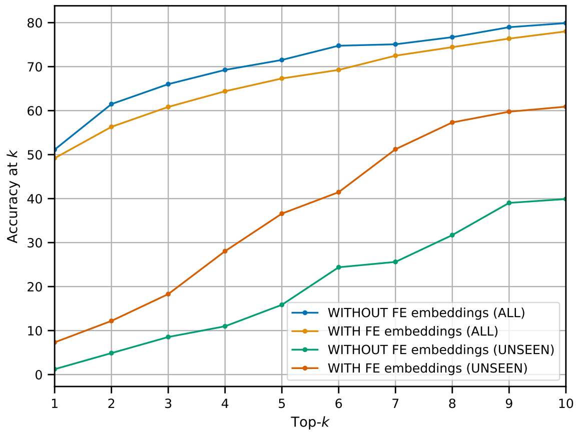

Figure 3: Accuracy@k for frame element prediction (WITH and WITHOUT frame element embeddings; on full test set and *unseen*-set)

in k, the margin between the performance of our system and baseline remains nearly the same on the whole test set. However, on unseen labels, the system with FE embeddings significantly outperforms the baseline, with an increase in k.

Overall, we observe a significant improvement in results with the help of our method. We also show that our extended dataset does help improve the performance of models in both cases.

7 Conclusion and Future Work

In this paper, we proposed a novel method for using FrameNet for NC interpretation in a continuous space. We use FrameNet mappings (word to frame and frame to frame element) to prune our search space. Our approach – prediction in continuous space – outperforms the random baseline and a stronger baseline approach. We show that the label embeddings generated using our approach help in the generalisation over unseen labels.

We annotated more noun compounds and analysed the issue in finding frame and frame elements. We create and release a tool that assists annotators in frame identification, for further research. We also show that extending the dataset created with our tool improves the system performance. Our experiments evaluate our proposed method on a small annotated dataset compared to the overall number of labels. We extend this existing dataset by annotating more NCs for various labels. Our study on the coverage issue for the annotation process helps develop a tool that assists the annotators in finding an appropriate frame. We provide promising results for the task of frame prediction. We analyse our results and discuss them in detail with respect to both frame and frame element prediction tasks.

In the future, we aim to find other ways of using FrameNet data for this task. We would also like to investigate why our approach provides promising results for the task of frame prediction but not for frame element prediction. We would like to explore more approaches to predict the frame elements effectively. We believe that using FrameNet embeddings can prove to be helpful for other tasks.

References

Kisuh Ahn, Johan Bos, David Kor, Malvina Nissim, Bonnie L Webber, and James R Curran. 2005. Question answering with QED at TREC 2005. In *TREC*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pages 86–90, Montreal, Canada.

Timothy Baldwin and Takaaki Tanaka. 2004. Translation by machine of complex nominals. In *Proceedings of the Workshop on Multiword Expressions Integrating Processing - MWE '04*, pages 24–31. Association for Computational Linguistics.

Renu Balyan and Niladri Chatterjee. 2015. Translating noun compounds using semantic relations. *Comput. Speech Lang.*, 32(1):91–108.

Ken Barker and Stan Szpakowicz. 1998. Semi-automatic recognition of noun modifier relationships. In *Proceedings of the 17th international conference on Computational linguistics -*, pages 96–102. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Teresa Botschen, Hatem Mousselly-Sergieh, and Iryna Gurevych. 2017. Prediction of frame-to-frame relations in the FrameNet hierarchy with frame embeddings. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 146–156, Vancouver, Canada. Association for Computational Linguistics.

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2009. SemEval-2010 task 9. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions - DEW '09*, pages 100–105. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Corina Dima and Erhard Hinrichs. 2015. Automatic noun compound interpretation using deep neural networks and word embeddings. *IWCS 2015*, pages 173–183.

Murhaf Fares. 2016. A dataset for joint noun-noun compound bracketing and interpretation. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 72–79.

Murhaf Fares, Stephan Oepen, and Erik Velldal. 2018. Transfer and multi-task learning for noun–noun compound interpretation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498.

Charles J. Fillmore. 1976. Frame semantics and the nature of language. *Ann NY Acad Sci*, 280(1 Origins and E):20–32.

Iris Hendrickx, Preslav Nakov, Stan Szpakowicz, Zornitsa Kozareva, Diarmuid O Séaghdha, and Tony Veale. 2013. SemEval-2013 task 4: Free paraphrases of noun compounds. *Atlanta, Georgia, USA*, page 138.

Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. 2019. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681. Association for Computational Linguistics.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *the Proceedings of the Human Language Technology Conference/North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Mark Lauer. 1995. *Designing statistical language learners: Experiments on compound nouns*. Ph.D. thesis, Ph. D. thesis, Macquarie University.

Judith N Levi. 1978. *The syntax and semantics of complex nominals*. Academic Press New York.

George A Miller. 1994. Wordnet: An electronic lexical database.

Preslav Nakov. 2013. On the interpretation of noun compounds: Syntax, semantics, and entailment. *Nat. Lang. Eng.*, 19(3):291–330.

Diarmuid Ó Séaghdha. 2007. Designing and evaluating a semantic annotation scheme for compound nouns. In *Proc. Corpus Linguistics*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. FrameNet+: Fast paraphrastic tripling of FrameNet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 408–413, Beijing, China. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Girishkumar Ponkiya, Rudra Murthy, Pushpak Bhattacharyya, and Girish Palshikar. 2020. Looking inside noun compounds: Unsupervised prepositional and free paraphrasing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4313–4323, Online. Association for Computational Linguistics.

Girishkumar Ponkiya, Kevin Patel, Pushpak Bhattacharyya, and Girish K Palshikar. 2018a. Towards a standardized dataset for noun compound interpretation. In *Language Resources and Evaluation Conference*, Miyazaki, Japan.

Girishkumar Ponkiya, Kevin Patel, Pushpak Bhattacharyya, and Girish K Palshikar. 2018b. Treat us like the sequences we are: Prepositional paraphrasing of noun compounds using LSTM. In *The 27th International Conference on Computational Linguistics (COLING 2018)*, pages 1827–1836, Santa Fe, New-Mexico, USA.

Barbara Rosario, Marti A. Hearst, and Charles Fillmore. 2001. The descent of hierarchy, and selection in relational semantics. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, pages 247–254. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *Proceedings of the 15th conference on Computational linguistics -*, pages 782–788. Association for Computational Linguistics.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

Beatrice Warren. 1978. Semantic patterns of noun-noun compounds. *Acta Universitatis Gothoburgensis. Gothenburg Studies in English Goteborg*, 41:1–266.

Diarmuid Ó Séaghdha. 2007. Annotating and learning compound noun semantics. In *Proceedings of the 45th Annual Meeting of the ACL: Student Research Workshop on - ACL '07*, pages 73–78. Association for Computational Linguistics.

Hypernym Discovery via a Recurrent Mapping Model

Yuhang Bai
BDBC and SKLSDE
Beihang University, China
`baiyh@act.buaa.edu.cn`

Richong Zhang[*]
BDBC and SKLSDE
Beihang University, China
`zhangrc@act.buaa.edu.cn`

Fanshuang Kong
BDBC and SKLSDE
Beihang University, China
`kongfs@act.buaa.edu.cn`

Junfan Chen
BDBC and SKLSDE
Beihang University, China
`chenjf@act.buaa.edu.cn`

Yongyi Mao
School of EECS
University of Ottawa, Canada
`ymao@uottawa.ca`

Abstract

Hypernym discovery aims to identify all possible hypernyms of a given term. The most recent hypernym discovery models exploit multiple mapping functions to project a term to different semantic spaces and then aggregate these embeddings to a general representation for further classification. We refer to this model as a *parallel style* model. In this work, we observe that there are hierarchical relations between a target terms' hypernyms. However, these hierarchical relations were not sufficiently considered in the previous *parallel style* model. To leverage the hierarchical relations, we propose a *sequential style* model that recurrently maps the query words to their hypernyms, starting from the most specific ones to the less specific ones. Empirical studies on SemEval-2018 Task 9 confirm the effectiveness of the presented model.

1 Introduction

Hypernymy, namely "is-a" relation, is a vital lexical-semantic relation in natural languages, which relates general terms to their instances or subtypes. In a hypernymy relation, we name a specific instance or subtype *hyponym* and its related general term *hypernym*. For instance, (*apple*, *fruit*) is in hypernymy relation, where *apple* is a hyponym and *fruit* is one of its hypernyms. Due to its general representation ability of semantic relations, hypernymy becomes an essential concept in modern natural-language research, and hypernymy detection becomes a fundamental component in many natural language processing (NLP) tasks, such as taxonomy construction (Snow et al., 2006; Navigli et al., 2011), semantic search (Hoffart et al., 2014; Roller et al., 2014; Roller and Erk, 2016), textual entailment (Dagan et al., 2013; Bowman et al., 2015; Yu et al., 2020) and question answering (Yahya et al., 2013; Gupta et al., 2018).

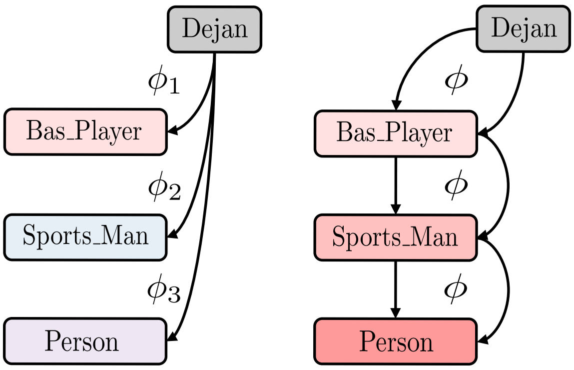

Figure 1: The parallel modeling (left) and sequential modeling (right). The Dejan is the name of a basketball player. The Bas_Player denotes the `Basketball Player` hypernym.

One branch of existing works builds the hypernymy-relation-identification problem as a "detection" task, which is only interested in whether a given term pair "is" or "is not" in hypernymy relation. These works formulate hypernym detection as a binary classification task. This hypernym detection task has been studied for years and plenty of models have been successfully applied in this task (Held and Habash, 2019a; Le et al., 2019).

This paper focuses on another problem named hypernym "discovery", which is different from the detection task. Given an input term, the hypernym discovery task retrieves a ranked list of its suitable hypernyms from a large corpus. For training, some hyponyms with their gold hypernym lists are provided. SemEval-2018 Task 9 (Camacho-Collados et al., 2018) is the only benchmark for this task. Existing studies working on this task mainly build a parallel mapping model (Bernier-Colborne and Barriere, 2018; Fu et al., 2014; Ustalov et al., 2017; Yamane et al., 2016). It introduces mul-

---

[*] Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2912–2921
August 1–6, 2021. ©2021 Association for Computational Linguistics

tiple parallel projections, each with independent parameters, to extract features and score the given hyponym/hypernym pair, as shown in the left picture of Figure 1. This model structure is motivated by the assumption that each fine-grained "is-a" relation should be modeled by a specific projection. However, the parallel mapping model may be ineffective due to uncountable "is-a" relation types in real-world applications. It may also suffer from overfitting due to its large model capacity. It also ignores the relations between different projections because they are independently learned.

To overcome the limitations of the parallel mapping model, we propose a recurrent mapping model. Our model is motivated by the observation that a hypernymy term may be produced by the "hypernymy transformation", which transforms a term to its closely related hypernym via a projection, as shown in Figure 1 (right). In the figure, we want to identify hypernyms of the term `Dejan`. The higher-level hypernym `Person` may be transformed from a `Dejan–Bas_Player–Sports_Man–Person` path. The parallel modeling in Figure 1 (left) may not have the ability to capture these sequential relations. Thus we devise a recurrent mapping model for these sequential relations. Note that the projection in our recurrent mapping model is shared among all hops. We assume that a higher-level hypernym term can be generated by operating multiple hypernymy transformations recurrently from the given term. In this way, we build the sequential relations between the transformed terms and largely reduce the parameters used in projection.

We also consider the types of hyponyms when building our recurrent mapping model. As pointed out in previous work (Bernier-Colborne and Barriere, 2018; Camacho-Collados et al., 2018), hyponyms are divided into two types, namely, the concept type and the entity type. This type information is available in the dataset. According to the type of a hyponym, the hypernymy relation can be divided into "subclass-of" (e.g. A guitar is an instrument) and "instance-of"(e.g. Rome is a city). The former represents the hypernymy relation between two concepts, and the latter connects an entity-hyponym with a concept-hypernymy. We first exploit two projections to obtain type-enhanced representation for different types of hyponyms, then feed the type-enhanced representation to a unified recurrent mapping model. In this way, we provide appropriate additional model capacity to handle different types of hyponyms and simultaneously we can utilize all data with hyponyms that belong to both types to train our model.

Our recurrent model outputs a representation vector at each hop. These vectors indicate hypernym representations from different hierarchy levels corresponding to the original hyponym. While scoring a candidate hypernym, we exploit an attention mechanism to aggregate hypernym representations from each hierarchy level. The attention weight of a level is viewed as the probability of the candidate hypernym lying in that level.

In summary, the contributions of this work are as follows:

- We propose a recurrent mapping model that utilizes a shared mapping unit to model the inherent hierarchical dependencies between hypernyms.
- To exploit the hyponym-type information, we use an independent projection matrix for each type to map hyponyms of different types to hypernym space.
- We utilize the attention mechanism into the aggregation module to obtain learnable weight.

2 Related Work

Earlier research on hypernym detection mainly focuses on unsupervised methods, which can be categorized into pattern-based methods and distributional methods. Pioneered by Hearst (1992), the pattern-based methods pre-define some common patterns that indicate hypernym relation, for example, word phrase "such as", "especially". Words occurring together in these pre-defined patterns will be extracted as hyponym/hypernym pairs. This method is quite intuitive. However, one serious problem is sparsity, since many hyponym/hypernym pairs never co-occur explicitly in the corpus, let alone in specific patterns. As a result, this method can provide high accuracy, at the cost of low recall. Seitner et al. (2016) try to improve this method by proposing an extended set of patterns, while (Snow et al., 2004; Shwartz et al., 2016) put forward methods to learn such lexical-syntactic patterns automatically. To further alleviate the sparsity problem, distributional models are proposed. Based on distributional inclusion hypothesis (DIH, Geffet and Dagan (2005)), these models represent every word as a distributional vector. Hyponym/hypernym pairs that never co-occur in the corpus can be captured based on the relation on their distributional vectors.

Most recent studies on hypernym discovery are supervised methods. Fu et al. (2014) was inspired by the well-known example "$V(king) - V(queen) \approx V(man) - V(woman)$", where V(w) is the embedding of the word w. The authors observed that the same linguistic regularities are preserved between hyponym and hypernyms. Thus they supposed that a hyponym can be projected to its hypernym. Besides the uniform linear projection, they also proposed piece-wise linear projections to model fine-grained hypernymy relations. Ustalov et al. (2017) makes use of negative examples to regularize the model, which further improves the performance of the model. Yamane et al. (2016) jointly learns the clusters and projections, the number of clusters can be determined depending on the learned projections and vice versa.

Some other work (Bernier-Colborne and Barriere, 2018; Held and Habash, 2019b) utilize a mix of both unsupervised and supervised methods. Dash et al. (2020) argued that hypernym relation can be represented as strict partial order relation (transitive, irreflexive and asymmetric) and they introduced a model which takes strict partial order relation as soft constraint.

3 Problem Formulation

Hypernym discovery task aims to identify all hypernym terms of a given hyponym term. Formally, let $\mathcal{V}$ be the set of all terms, and $\mathcal{X}$ and $\mathcal{Y}$ denote the set of all hyponym terms and candidate hypernym terms, respectively. Both $\mathcal{X}$ and $\mathcal{Y}$ are subsets of $\mathcal{V}$. For a given hyponym $x \in \mathcal{X}$, a hypernymy detector is expected to find all hypernym terms $y \in \mathcal{Y}$ that make (x, y) a hypernymy relation.

4 Methodology

In this section, we introduce the proposed Recurrent Mapping Model (RMM). It consists of three components: a type enhanced representation module that maps the hyponym embedding via different projections, a recurrent mapping module that transforms the term features into multiple concept-level semantics and an aggregation module that aggregates hypernym representations from each hierarchy level and compute a final score via the aggregated vector. We next describe each component in detail.

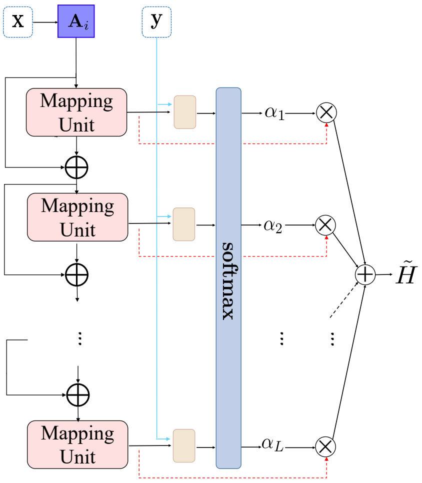

Figure 2: Overview of RMM model.

4.1 Type Enhanced Representation

Type information is essential for hypernym discovery. It implies different hypernym types that provide the hyponyms' attributes. Utilizing type information may increase the hypernym-discovery performance. Specifically, let $\mathbf{x} \in \mathbb{R}^{d\times 1}$ be the embedding of hyponym term x, for its type indexed by i, we introduce a projection $\mathbf{A}_i \in \mathbb{R}^{d\times d}$ to map the hyponym embedding as a type enhanced representation $\mathbf{t} \in \mathbb{R}^{d\times 1}$ as follows:

$$\mathbf{t} = \mathbf{A}_i \mathbf{x} \tag{1}$$

In this way, we provide appropriate additional model capacity to handle different types of hyponyms and simultaneously we can utilize all data with hyponyms that belong to both types to train our model.

4.2 Recurrent Mapping Module

One motivation of this paper is that we assume the candidate hypernym terms may come from different concept levels. The hypernym term at a higher concept-level can be obtained by transforming from a lower-level hypernym term. To model this transformation process, we build a recurrent mapping module with a shared projection function. Specifically, let $\mathbf{h}_l \in \mathbb{R}^{d\times 1}$ denote the transformed concept semantics at the l^{th} level (or the term representation after l-hop transformations). The transformation from a l^{th}-level concept semantics to a

$(l+1)^{\text{th}}$-level is then formulated as

$$\hat{\mathbf{h}}_{l+1} = \mathbf{W}_\phi \mathbf{h}_l \tag{2}$$

In Equation 2, the semantic representation of the primary concept level is exactly the type-enhanced representation, that is $\mathbf{h}_0 = \mathbf{t}$. $\mathbf{W}_\phi \in \mathbb{R}^{d \times d}$ is the learnable projection matrix which is shared during each transformation. We recurrently make L transformations from the original hyponym embedding. We hope this recurrent mapping process captures the relationships of different concept-levels and associates the given hyponym term with its hypernym terms at different concept levels.

Theoretically, when the number of layers increases in a neural network, its model capabilities become large and it should produce lower training error. However, the gradients may vanish after they are propagated through many layers, thus degrade the model performance (He et al., 2016). Consequently, optimizing the projection matrix in Equation 2 is difficult when the maximum number of transformations L becomes large. Following the previous work (He et al., 2016), we introduce residual network (ResNet) to overcome the gradient vanishing problem as follows:

$$\mathbf{h}_{l+1} = \hat{\mathbf{h}}_{l+1} + \mathbf{h}_l \tag{3}$$

ResNet improves the training efficiency by reformulating the $(l+1)^{\text{th}}$-level representation with reference to the l^{th}-level representation. In addition, the ResNet also forces the model to *remember* the past information during multi-hop transformation.

4.3 Aggregation Module

After obtaining the multi-hop representations of the given hyponym term, we obtain a final representation via letting multi-hop representations attend to the candidate embedding.

Concretely, by calculating dot product between representations from each hop and candidate hypernym, we can get a weight vector. The attention weight of a level is viewed as the probability of the candidate hypernym lying in that level. The weight vector and the final score of the candidate term embedding $\mathbf{y}$ being the input's hypernym are calculated as follows:

$$\begin{aligned} \alpha_l &= \frac{\exp(\mathbf{h}_l \cdot \mathbf{y})}{\sum\limits_{k=1}^{L} \exp(\mathbf{h}_k \cdot \mathbf{y})} \\ \widetilde{H} &= \sum_{l=1}^{L} \alpha_l \cdot \mathbf{h}_l \\ s_y &= \widetilde{H} \cdot \mathbf{y} \end{aligned} \tag{4}$$

This mechanism allows the model to adaptively assign weights for representations from each hop based on the candidate terms. We expect this aggregation strategy to provide appropriate scores that correctly rank the true hypernym terms ahead of other candidate terms.

4.4 Loss Function

Hypernymy discovery is viewed as a ranking problem. Existing models optimize this ranking problem using a pair-wise loss function. They first score the candidate hypernym terms by training a binary classifier that identifies whether a given (x, y) pair supports a hypernymy relation, then rank the candidates by descending order of their matching scores. Although pair-wise loss function is efficient, it learns to score each candidate independently (Shi and Weninger, 2017). Instead of optimizing the model via pairwise loss, we introduce the list-wise loss function that learns to score the candidates collectively.

Let $\mathcal{X}_{train}$ be the set of hyponym terms in the training set. For each hyponym term x in the training set, let $\mathcal{Q}_x$ denote a set of candidate hypernyms, which consists of only gold hypernym terms of x, and $\mathcal{C}_x$ denotes a controllable number of negative candidates. Then we compute the cross-entropy loss with the sampled positive and negative candidates as follows:

$$\mathcal{L} = -\sum_{x \in \mathcal{X}_{train}} \frac{1}{|\mathcal{Q}_x|} \sum_{y \in \mathcal{Q}_x} \log \frac{\exp(s_y)}{\sum_{y' \in \mathcal{Q}_x \cup \mathcal{C}_x} \exp(s_{y'})} \tag{5}$$

where $\frac{1}{|\mathcal{Q}_x|}$ is a normalizing term that balances the learning of all gold candidate scores. The number of negative candidates $|\mathcal{C}_x|$ is optional in practice, we leave it as a hyperparameter. We may set $\mathcal{C}_x = \mathcal{Y} - \mathcal{Q}_x$ with enough computation source available. In this way, we simultaneously optimize the scores for a collection of candidates, improving the training efficiency.

5 Experiments

5.1 Data Sets

We evaluate the performance of our model on SemEval-2018 Task 9 [1]benchmark for hypernym discovery. This shared task consists of five different subtasks covering both general-purpose (multiple languages-English, Italian, and Spanish) and domain-specific (Music and Medicine domains) tasks. For each subtask, a large textual corpus, a vocabulary including all valid hypernyms and a training and testing set of hyponyms and its gold hypernyms are provided. In this paper, we consider the three English subtasks: 1A (general), 2A (medical) and 2B (music). The summarized statistics of the datasets are shown in Table 1. For more details, we refer the reader to the original SemEval-2018 Task 9 (Camacho-Collados et al., 2018) paper.

subtask	corpus size	#train	#test
1A	16G	1500	1500
2A	800M	500	500
2B	500M	500	500

Table 1: Data set statistics

Three metrics were used for the performance evaluation.

Mean Average Precision (MAP) For a given query word, average precision(AP) is the average of the correctness of each obtained hypernym from the search space. MAP is the mean of this value among all queries in the data set.

Mean Reciprocal Rank (MRR) Since MAP ignores the exact rank of the true hypernyms, we introduce the Mean Reciprocal Rank (MRR) metric which focuses on the top results performance. Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks over all queries. The reciprocal rank of an individual query is the reciprocal of the rank in which the first true hypernym is returned.

Precision at K (P@K) Precision at K is the proportion of the top-K results that are true hypernyms of a given query.

Following the same evaluation procedures as previous studies (Bernier-Colborne and Barriere, 2018; Held and Habash, 2019b; Dash et al., 2020), the scorer script provided by SemEval-2018 Task 9 is exploited for evaluating our proposed model and comparing fairly with other recent models.

5.1.1 Compared Models

We compare our model with baseline models: MFH (Camacho-Collados et al., 2018), vTE (Camacho-Collados et al., 2018), 300-sparsans(Berend et al., 2018), and NLP-HZ (Qiu et al., 2018). 8 SPON (Dash et al., 2020), Hybrid of SVD & NN (Held and Habash, 2019b).

Besides, we also compare our model with recent models. Brief descriptions of these models are given as follows.

CRIM (Bernier-Colborne and Barriere, 2018) In the CRIM model, multiple parallel projections are introduced to map the queries to different spaces. A logistic regression function is then applied to compute the final score. In addition, this module is combined with an unsupervised system that identifies hypernym based on specific Hearst-style patterns. The final output of CRIM is the combination of both supervised and unsupervised models.

SPON (Dash et al., 2020) In the SPON model, non-negative activations and residual connections are exploited to enforce asymmetry and transitive as soft constraints.

Hybrid of SVD & NN (Held and Habash, 2019b) This model is a hybrid system which exploits both unsupervised and supervised approaches at the same time.

In their proposed supervised module, the nearest neighbor approach is used. Given a hyponym, candidate hypernyms which are its nearest neighbors are returned. A similarity cut-off point is trained on tuning data, such that if there is no neighbor with a similarity greater than the cutoff point, the model simply returns the most frequent set of hypernyms from the entire training set.

5.2 Implementation

We trained 200-dimensional word embeddings via the standard skip-gram word2vec algorithm (Mikolov et al., 2013) on the provided textual corpus. The representations of the hyponyms and hypernyms are directly initialized by the pre-trained word2vec embeddings. In the training process of our model, hyponym embeddings are fixed and hypernym embeddings are learnable. To avoid overfitting, dropout is applied after each mapping function. Besides, an early stop strategy is also used. Thus, if MAP on the validation set does not

---

[1] https://competitions.codalab.org/competitions/17119

Model	1A			2A			2B		
	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5
MFH	8.77	21.39	7.81	28.93	35.80	34.20	33.32	51.48	35.76
vTE	10.60	23.83	9.91	18.84	41.07	20.71	12.99	39.36	12.41
Sparsans	8.95	19.44	8.63	17.94	37.56	17.06	12.08	25.14	11.73
NLP-HZ	9.37	17.29	9.19	20.04	28.27	20.39	11.37	19.19	11.23
Hybrid	15.97	34.07	15.00	37.85	**64.47**	**40.19**	54.62	**77.24**	55.08
CRIM	19.78	36.10	19.03	34.05	54.64	36.77	40.97	60.93	41.31
SPON	20.20	36.95	19.40	33.50	50.60	35.10	54.70	71.20	56.30
RMM	**27.12**	**39.07**	**23.41**	**38.56**	54.89	37.17	**63.86**	74.75	**61.61**
	±0.12	±0.42	±0.22	±0.34	±0.63	±0.33	±0.06	±0.52	±0.25

Table 2: Performance comparison on different models on the benchmark datasets. In the first column, Hybrid denotes the Hybrid of SVD/NN model and Sparsans represents the 300-sparsans model. The results of RMM are average from 3 runs of experiments. Other reported results are from their corresponding original paper.

increase after 200 continuous epochs, training will be terminated. The max epoch is set to 1000. In addition, gradient clipping is used in the weight updating process, with a clip of $1e^{-4}$. We use the Adam optimizer with beta1=beta2=0.9 and with a learning rate of $2e^{-4}$ for all datasets. We choose two separate embedding transformation matrices for two different query types. When initializing these projection matrices in the mapping function, we add random noises of Gaussian distribution (zero mean and $\frac{1}{200}$ variance) to an identity matrix. Finally, we implement our model using PyTorch on a Linux machine with a GPU device Tesla V100 SXM2 32GB.

5.3 Results

5.3.1 Overall Results

Table 2 shows the MAP, MRR and P@5 performance of our model and the other baseline models across multiple hypernym discovery sub-tasks. The value of L is tuned over the validation set, we used L=2 for subtask 1A and L=3 for 2A and 2B. Note that, to avoid the performance randomness, the performance of our model is the average of three random runs. For the other compared models, the reported performance is taken from their original paper. The values printed in **bold** font are the top-performing models in the comparison.

In the table, it is clear that our recurrent mapping model (RMM) outperforms almost all existing baseline models on all the general English hypernym discovery tasks. More specifically, on all sub-tasks, RMM outperforms any supervised hypernym discovery models on all metrics. The only model that RMM does not fully beat is the Hybrid of SVD/NN model, which uses both unsupervised and supervised approaches. We note however that RMM scores best 6 out of the 9 metrics across the compared methods.

In addition, RMM outperforms the most recent supervised models, i.e. CRIM and SPOM by a significant margin. This performance suggests that the true hypernym rank is generally higher than other candidate words using RMM.

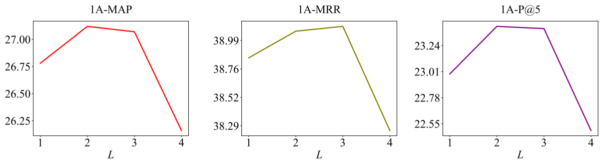

(a) 1A-MAP (b) 1A-MRR (c) 1A-P@5

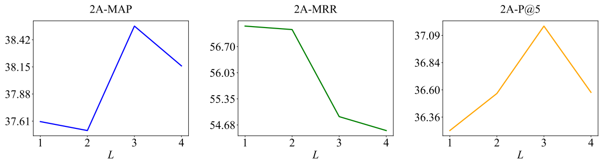

(d) 2A-MAP (e) 2A-MRR (f) 2A-P@5

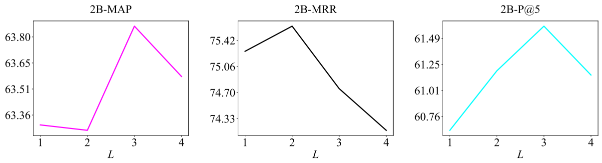

(d) 2B-MAP (e) 2B-MRR (f) 2B-P@5

Figure 3: The comparison of RMM with different mapping units. In each sub-figure, x axis represents the number of mapping units and the y axis represents the performance of the corresponding model. The results reported is an average of three experiments.

5.3.2 The Impact of the Number of Mapping Units

We now examine the impact of the chosen number of mappings on RMM. In our recurrent mapping process, we use mapping functions from a specific term to a more general term it belongs to. The number of mappings can be understood as the number of projections from the original query. Thus, this value is a hyper-parameter in RMM. We let L denote the number of mapping units.

As discussed above, we speculate that the correct hypernym might be chosen by different mapping units (MU). Here we present experimental results to support this claim and show how this value affects the performance of RMM. We vary L from 1 to 4 to observe the performance. The experimental results are shown in Figure 3. Note that all values in the figure are the average over three runs.

The results indicate that RMM is sensitive to L. RMM shows a general increasing performance when increasing L. This might be due to that the hypernym semantics transformation is captured by our model. We find that RMM performs best at $L = 3$ for 2A, 2B and $L = 2$ for 1A. This phenomenon is consistent with the typical hypernymy transformation situation in the data set. We observe that, in general, the true hypernym list is often in the form of a two or three layers hierarchical structure. Here we list two examples of the hierarchical structure: `guitar` → `stringed instrument`→ `musical instrument`; `alternative rock`→ `rock music`→ `music` → `music style`. So that RMM will be effective when choosing L as 2 or 3.

It also can be noted that when choosing $L = 4$, RMM achieves a lower performance on all data sets. This result confirms our claim and shows the effectiveness of the sequential structure exploited in RMM.

5.3.3 Ablation Study

To more precisely evaluate RMM and to comprehensively analyze the contribution of each component of our model, we conduct an additional experiment of ablation studies.

Specifically, this experiment involves three components, ResNet connection between mapping units, separate transformation function for query type and the attention mechanism to identify the importance of mapping units. By removing or modifying each of them individually, we are able to observe their effects on our model.

The experiment was performed on the same datasets along with the same experimental setup and hyperparameters as in the main experiment.

Without Residual Mechanism RMM uses a residual mechanism to overcome the gradient vanishing problem and to improve the model performance. In this ablation setup, we directly remove this ResNet connection and refer to this setup as $\text{RMM}_{\text{w/o ResNet}}$.

Without Separating Query Types Before multi-hop mapping, RMM model exploits two different learnable projection matrices for query types of `entity` and `concept` to transform original embedding to "is-a" embedding. In this ablation experiment, we unify these two projection matrices and refer to this setup as $\text{RMM}_{\text{w/o QType}}$. It's worth noting that hyponyms in 2A(music) are all of concept type, thus unifying these two projection matrices can get a close performance as before.

Without Attention Mechanism RMM model makes use of an attention mechanism to aggregate hypernym representations from all mapping units according to candidate hypernyms. In this ablation experiment, to verify its contribution, we remove this weighting mechanism and instead use a mean approach, which simply averages the representations of the output of each unit to aggregate semantics from all mapping units. We refer to this setup as $\text{RMM}_{\text{w/o Att}}$.

Table 3 shows the result of our ablation study. It shows that all components are critical for RMM. Specifically, we can see that removing ResNet degrades the model performance, which proves ResNet can avoid the information loss between units. Without using the query type, model performance degrades as well. It's notable that CRIM (Bernier-Colborne and Barriere, 2018) also utilized the type information of hypernyms by training separate logistic regression classifiers for different types of hyponyms. However, their ablation study suggests that their type modeling actually degrades the model performance which is contrary to the results shown in our ablation study. It indicates that the type information is better modeled in our model. From the last raw of Table 3, we can conclude that taking the average of each mapping unit's output decreases the model performance.

Model	1A			2A			2B		
	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5
RMM	27.12	39.07	23.41	38.56	54.89	37.17	63.86	74.75	61.61
$\text{RMM}_{\text{w/o ResNet}}$	25.78	37.08	22.41	35.80	55.65	34.52	62.39	74.38	60.21
$\text{RMM}_{\text{w/o QType}}$	24.76	37.14	21.22	38.23	54.32	36.73	61.71	69.89	60.18
$\text{RMM}_{\text{w/o Att}}$	25.91	37.63	22.37	36.60	57.04	35.00	62.90	75.74	60.27

Table 3: Ablation study.

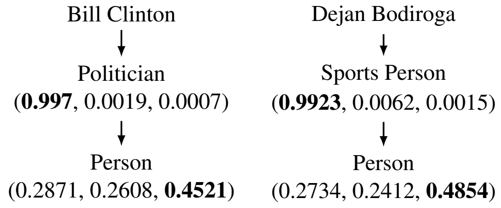

(a) Case 1 from 1A (b) Case 2 from 1A

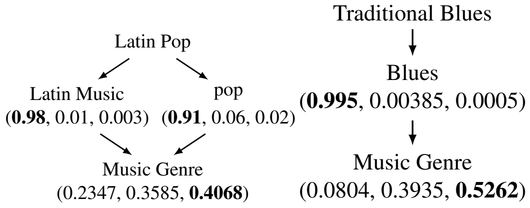

(c) Case 3 from 2B (d) Case 4 from 2B

Figure 4: Case Studies. The values in the bracket represent the weights on the first, the second and the third mapping units filtered by hypernym.

5.3.4 Case Study

In this subsection, we present a detailed result analysis on 4 randomly chosen cases from our testing sets, with the aim to validate our motivation of the recursive structure of RMM being capable of capturing the near hypernym first and far hypernym later. We observe the weights on different units and wish to examine if RMM indeed assigns higher weights to relative-lower units when the true hypernym is near the query word and vice versa. The results are shown in Figure 4.

From the figure we can observe that RMM is able to assign a higher weight to the first mapping units for its first hop hypernym. This is seen in all 4 cases by assigning a weight more than 0.9 to the first unit for their immediate hypernym. For example, in Figure 4(a) the hypernym `Politician` for query `Bill Clinton`. On the contrary, for a far hypernym of a query, a higher weight is on the last mapping units. For example, in Figure 4(c), the hypernym `Music Genre` for query `Latin Pop`, a higher weight is on the third units.

This result confirms the capability of RMM in capturing the latent hypernymy transformation and hierarchical dependencies between hypernyms.

6 Conclusion

Hypernym discovery is a basic task in natural language processing. Existing studies focus on designing better models for discovering better mapping functions from hyponyms to hypernyms. However, the latent semantic transformation between the hypernyms of one hyponym is not considered. In this study, both the mapping and the semantic transformation between hypernyms are considered by a recursive mapping model. In addition, with the attention mechanism, different levels of transformations are softly mixed in the final representation for the final classification task. Empirical studies on a public hypernym discovery task verify the superiority of the presented recursive model.

This study is a first attempt on modeling the transformation between hypernyms and we only achieve preliminary progress. The better usage of this will definitely promote the effectiveness of hypernym discovery. In practice, this transformation can be extracted and graph convolutional network (GCN) or other neural networks can be exploited for explicitly this information. Also, the combination of unsupervised and supervised models has shown advantages. However, most of these hybrid models are two separate processes and the supervised part highly depends on the pre-defined "is A" patterns. To build a uniform hybrid model still remains an open problem. We will study these open problems in our future work.

Acknowledgement

This work is supported partly by the National Natural Science Foundation of China (No. 61772059), by the Fundamental Research Funds for the Central Universities, by the Beijing S&T Committee (No. Z191100008619007) and by the State Key Laboratory of Software Development Environment (No. SKLSDE-2020ZX-14).

References

Gábor Berend, Márton Makrai, and Péter Fldiák. 2018. 300-sparsans at semeval-2018 task 9: Hypernymy as interaction of sparse attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*.

Gabriel Bernier-Colborne and Caroline Barriere. 2018. Crim at semeval-2018 task 9: A hybrid approach to hypernym discovery. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 725–731.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP 2015*, pages 632–642.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. 2018. Semeval-2018 task 9: Hypernym discovery. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018); 2018 Jun 5-6; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 712–24.* ACL (Association for Computational Linguistics).

I Dagan, D Roth, F Zanzotto, and M Sammons. 2013. *Recognizing Textual Entailment:Models and Applications*. Morgan Claypool.

Sarthak Dash, Md Faisal Mahbub Chowdhury, Alfio Gliozzo, Nandana Mihindukulasooriya, and Nicolas Rodolfo Fauceglia. 2020. Hypernym detection using strict partial order networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7626–7633.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114.

Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Geo Jain, and Shubhashis Sengupta. 2018. Can taxonomy help? improving semantic question matching using question taxonomy. In *COLING 2018*, pages 499–513.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR 2016*, pages 770–778.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Coling 1992 volume 2: The 15th international conference on computational linguistics*.

William Held and Nizar Habash. 2019a. The effectiveness of simple hybrid systems for hypernym discovery. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

William Held and Nizar Habash. 2019b. The effectiveness of simple hybrid systems for hypernym discovery. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3362–3367.

Johannes Hoffart, Dragan Milchevski, and Gerhard Weikum. 2014. STICS: searching with strings, things, and cats. In *SIGIR 2014*, pages 1247–1248.

Matthew Le, Stephen Roller, Laetitia Papaxanthos, Douwe Kiela, and Maximilian Nickel. 2019. Inferring concept hierarchies from text corpora via hyperbolic embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI 2011*, pages 1872–1877.

Wei Qiu, Mosha Chen, Linlin Li, and Luo Si. 2018. NLP_HZ at SemEval-2018 task 9: a nearest neighbor approach. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana. Association for Computational Linguistics.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *EMNLP 2016*, pages 2163–2172.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING 2014*, pages 1025–1036.

Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. 2016. A large database of hypernymy relations extracted from the web. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 360–367.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *AAAI 2017*, pages 1236–1242.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. *arXiv preprint arXiv:1603.06076*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL 2006*.

Dmitry Ustalov, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Negative sampling improves hypernymy extraction based on projection learning. *arXiv preprint arXiv:1707.03903*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *CIKM 2013*, pages 1107–1116.

Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. 2016. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1871–1879.

Changlong Yu, Hongming Zhang, Yangqiu Song, Wilfred Ng, and Lifeng Shang. 2020. Enriching large-scale eventuality knowledge graph with entailment relations. In *AKBC 2020*.

Modeling the Influence of Verb Aspect on the Activation of Typical Event Locations with BERT

Won Ik Cho
Seoul National University
tsatsuki@snu.ac.kr

Emmanuele Chersoni
The Hong Kong Polytechnic University
emmanuelechersoni@gmail.com

Yu-Yin Hsu
The Hong Kong Polytechnic University
yu-yin.hsu@polyu.edu.hk

Chu-Ren Huang
The Hong Kong Polytechnic University
churen.huang@polyu.edu.hk

Abstract

Prior studies on event knowledge in sentence comprehension have shown that the aspect of the main verb plays an important role in the processing of non-core semantic roles, such as locations: when the aspect of the main verb is imperfective, locations become more salient in the mental representation of the event and are easier for human comprehenders to process.

In our study, we tested the popular language model BERT on two datasets derived from experimental studies to determine whether BERT's predictions of prototypical event locations were also influenced by aspect. We found that, although BERT efficiently modelled the typicality of locations, it did so independently of the verb aspect. Even when the transformer was forced to focus on the verb phrase by masking the context words in the sentence, the typicality predictions were still accurate; in addition, we found aspect to have a stronger influence on the scores, with locations in the imperfective setting being associated with lower surprisal values.

1 Introduction

It has been generally acknowledged in sentence processing research that humans activate generalized event knowledge in the process of understanding natural language sentences (McRae and Matsuki, 2009). Reading/listening verbs (e.g., *open*) activate expectations about their typical arguments (e.g., *door*) (McRae et al., 1998; Ferretti et al., 2001) and vice versa (McRae et al., 2005); the same effect has been found for nouns and their typical co-arguments (Hare et al., 2009). These expectations concerning typical arguments are encoded in the mental lexicon, and are exploited by humans to evaluate the plausibility of verb-argument combinations. Such knowledge is used during sentence processing to generate predictions about upcoming arguments: using different experimental paradigms (e.g. EEG, eye-tracking etc.), previous studies provided evidence that sentences including typical argument combinations are easier for humans to process (Bicknell et al., 2010; Matsuki et al., 2011).

Other studies have investigated the role played by *verb aspect* in event knowledge activation, particularly for non-core roles such as *locations* and *instruments*. Aspect is a grammatical device that denotes the duration, onset, and completion status of an event. According to linguistic theory, a fundamental opposition exists between the *imperfective* aspect (e.g., *The customer was eating in the restaurant*), which describes the event as on-going, and the *perfective* aspect (e.g., *The customer had eaten in the restaurant*), which describes the event as a closed unit and focuses on the resulting state (Madden and Zwaan, 2003).

Based on the above-mentioned literature, Ferretti et al. (2007) used stimulus-onset asynchrony priming and the EEG paradigms to show that, in English, specific expectations were activated for event locations by the verbs describing those events, but only when the verbs were in the imperfective form. Similar findings have been reported for instruments by Truitt and Zwaan (1997) and, more recently, by Madden-Lombardi et al. (2017) in a self-paced reading experiment in French. In line with these findings, Coll-Florit and Gennari (2011) found that imperfective verbs were related to a wider range of semantic associations than perfective ones, because the mental representation of an event as on-going allows one to focus more easily on all the entities that are relevant to the described action (instruments, places, objects, etc.).

In this study, we modelled two datasets on locations using BERT, a state-of-the-art language model (Devlin et al., 2019), and we tested whether verb aspect influenced the model's predictions for upcoming event's locations, by comparing them

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2922–2929
August 1–6, 2021.

in terms of perfective and imperfective sentences. We found that i) BERT was able to accurately identify typical locations for an event, and that it did so independently of the aspect of the main verb, since the activation levels of locations did not differ significantly across conditions; ii) even when the transformer was forced to focus on the main verb and the other words in the sentence were masked, BERT's typicality predictions were still accurate, and an additional effect of aspect appeared, with locations having significantly lower surprisal values in the imperfective condition.

2 Related Work

Transformer models have become increasingly popular in NLP in recent years (Vaswani et al., 2017; Devlin et al., 2019). The most successful model is probably BERT, which is trained on a *masked language modeling* objective: given the left and the right context of a masked word in a natural language sentence, the model has to predict the word. This conceptually simple yet powerful mechanism has made BERT a very appealing option for NLP researchers working on supervised tasks, and its contextualized representations have taken state-of-the-art performances to new heights.

A number of psycholinguistic-inspired studies designed tests to investigate the actual linguistic abilities of neural network models, including Transformer models. Most of these studies have focused on syntactic phenomena, such as verb-subject agreement and filler-gap dependencies (Linzen et al., 2016; Wilcox et al., 2018; Gulordava et al., 2018; Futrell et al., 2019; Prasad et al., 2019). By contrast, Ettinger (2020) focused on the semantic and pragmatic abilities of the BERT language model by using stimuli from the N400 experiments conducted by Kutas and Hillyard (1984), and showed that the model was strong in associating nouns with their hypernyms, but struggled to handle negations. Close to the spirit of our contribution, Misra et al. (2020) investigated BERT's predictions in a setting aimed at reproducing human semantic priming; they reported that BERT was indeed sensitive to "priming" and predicted a word with higher probability when the context included a related word as opposed to an unrelated one, but this effect decreased in the presence of strongly informative and constraining contexts.

Recent work by Metheniti et al. (2020) has explored the capacity of BERT to reproduce the selectional preferences for verbs - which, from our perspective, was equivalent to modeling the thematic fit of typical event participants (Sayeed et al., 2016; Santus et al., 2017; Chersoni et al., 2020; Marton and Sayeed, 2021). Metheniti and colleagues reported that the correlation of the predictions with human judgements increased when they applied attention masks to the context words in the sentence and forced the model to focus only on the verbs. Finally, Transformers have been used to model typicality effects in language by Misra et al. (2021), although in a different context; i.e. the influence of typicality on category membership judgements.

To the best of our knowledge, the current study is the first to attempt to model argument typicality predictions with BERT for a non-core role (location), and the first to investigate whether and how such predictions are influenced by verb aspect, as in the case in human language processing.

3 Experiments

3.1 Datasets

In our work, we used two datasets that we obtained from the previous studies. The first dataset, which we refer to as **Ferretti07**, consists of the experimental items used by Ferretti et al. (2007) in their EEG experiment. The authors made available a subset of 38 items in which the verb phrase was specifically biased to be followed by a locative prepositional phrase in sentence completion tasks. We excluded 6 of them, in which the location noun was extremely rare and was not included in BERT's basic vocabulary[1]. Each item consisted of an intransitive sentence, with the verb phrase in either the past perfect tense (perfective condition, $PERF$) or the progressive past tense (imperfective condition, $IMPERF$), and a location argument, either typical (TYP) or less typical but still plausible (NON_TYP) (see Example 1).

(1) a. The boy had fished at the lake. $(PERF, TYP)$
b. The boy was fishing at the lake. $(IMPERF, TYP)$
c. The boy had fished at the swamp. $(PERF, NON\_TYP)$
d. The boy was fishing at the swamp. $(IMPERF, NON\_TYP)$

---

[1] In such cases, the location nouns would be split by the BERT tokenizer.

Interestingly, Ferretti et al. (2007) found that atypical locations elicited significantly larger N400 amplitudes in the imperfective condition, but there were no significant differences in the perfective one, suggesting that the differences in location typicality become salient only when the event is being described as on-going. Typical locations elicited smaller N400 amplitudes, while aspect was found to have no main effect *per se*, being significant only in the interaction with typicality.

We generated another dataset, which we refer to as **Ferretti01**, by using the typicality judgements dataset created by Ferretti et al. (2001). From their data, we extracted all the verb-argument pairs for which the mean typicality rating was $>= 4$ on a Likert scale, for a total of 135 pairs, and asked two PhD students in Linguistics who were proficient English speakers, to use the same pairs to generate complete English sentences with a structure similar to the items in **Ferretti07**. A third PhD student, a native speaker of British English, made the final check of the correctness of the sentences. [2] Each item in **Ferretti01** came in the $PERF$ vs. $IMPERF$ condition (see Example 2).

(2) a. He had danced in the ballroom. ($PERF$)
b. He was dancing in the ballroom ($IMPERF$)

3.2 Model and Settings

Similarly to Misra et al. (2020), we considered the surprisal score for an argument word (in our case, the location) as a measure of the model's expectations in the given context; we replaced the location token at the end of each sentence with a *[MASK]* and we asked BERT to predict its probability.

Surprisal was shown to be an efficient predictor of self-paced reading times (Hale, 2001; Levy, 2008; Smith and Levy, 2013) and of the N400 amplitude (Frank et al., 2013), and we expected it to be inversely correlated with typicality: the more typical a location in a given context is, the less surprising it will be.

To approximate the results of the original study by Ferretti et al. (2007), BERT's surprisal scores for the locations would have to show an interaction between aspect and typicality, with the scores being significantly higher for atypical fillers only in the imperfective condition. Moreover, typical fillers should be assigned lower surprisal scores.

We experimented with the *bert-base-uncased* model, as implemented in the HuggingFace's Transformers library (Wolf et al., 2019). [3] For each sentence, we masked the location loc, corresponding to the last token in the sentence (e.g., *The girl was skating in the [MASK]*), and computed its Surprisal score $Surp$ in the context C as:

$$Surp(loc|C) = -logP(loc|C) \tag{1}$$

where $P(loc|C)$ is the probability computed by the softmax layer of BERT for the loc word as the masked token in the sentence context C.

Finally, we experimented with two different settings: a *standard* setting, in which BERT was able to see the entire context of the sentence, and a *context mask* setting, in which we used an **attention mask** on all the sentence tokens except for the verb phrase ones (see Example 3).

(3) a. The boy was fishing at the *[MASK]* (*standard*)
b. ~~The boy~~ was fishing ~~at the~~ *[MASK]* (*context mask*)

In this setting, we blocked BERT's self-attention mechanism, forcing it to use only the verb phrase to predict the masked token. In this way, we were able to analyze the effect of the tense without the interference of the other context words. [4]

4 Results and Analysis

We ran all the comparisons between scores using linear mixed effects models with the LMER function in the R statistical software (see also Appendix B for the full results). We first compared the surprisal scores that we obtained in the standard setting, in which BERT had access to all the tokens in the sentence. No main effect of aspect was found ($p > 0.1$), neither in the **Ferretti01** nor in the **Ferretti07** dataset. In other words, verb aspect did not seem to have an influence on the activation degree of typical location fillers, as the scores in the two conditions were essentially equivalent.

On the other hand, BERT's predictions for the typicality of the location fillers seemed to be extremely accurate. In the **Ferretti07** dataset, a

---

[2] More details on the dataset creation are in Appendix A.

[3] `github.com/huggingface/transformers`

[4] A similar setting was proposed by Metheniti et al. (2020) for modeling human judgements on selectional preferences: with the context mask, the authors found increased correlation between model predictions and human ratings.

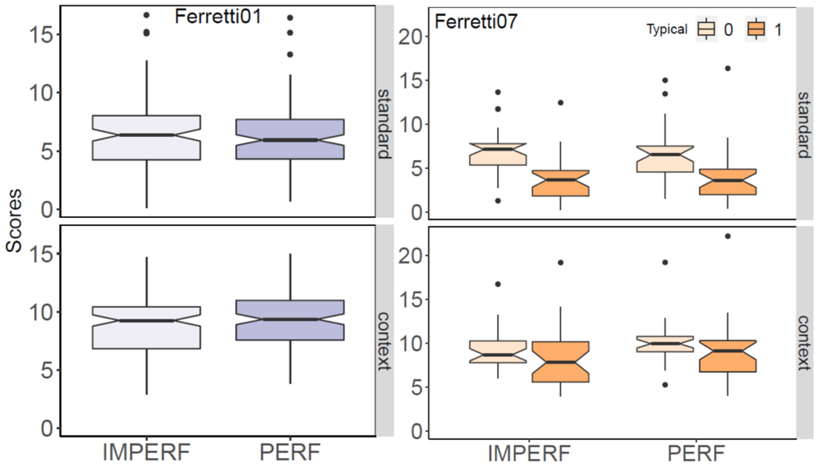

Figure 1: Boxplots with the surprisal scores calculated on the **Ferretti01** dataset (left) and on the **Ferretti07** dataset (right), in the standard setting (above) and in the context mask setting (below).

main effect of typicality was found ($p < 0.001$), with TYP sentences showing significantly lower Surprisal scores in the TYP condition than the NON_TYP ones (see also the boxplots in Figure 1, on the right). The interaction of aspect and typicality, however, was not significant ($p > 0.1$). Note that the NON_TYP locations were selected by Ferretti et al. (2007) in order to be plausible, and thus it is interesting that BERT correctly identified the ones of the TYP sentences as being more typical. However, the verb aspect did not play a role in this, since this ability was not influenced by the aspect condition. As a possible explanation, Klafka and Ettinger (2020) recently showed how the semantic information about the animacy of an argument noun in BERT was spread over the tokens of a sentence, and the same might be true also for the semantic information about the typicality of a location in a given event context. Pairwise comparisons confirmed that TYP sentences obtained significantly lower scores for both the $PERF$ and $IMPERF$ conditions ($p < 0.001$), while no significant difference between $PERF$ and $IMPERF$ in typicality condition was found. We then repeated the experiments using the context mask to block BERT's attention mechanism for all the words in the sentence except for the verb phrase tokens (recall Example 3b.) and we observed some interesting changes in our results. On the one hand, in the **Ferretti01** dataset, we observed a marginally significant effect of aspect ($p < 0.1$), with lower surprisal scores for the $IMPERF$ condition. On the other hand, in the **Ferretti07** dataset, we found main effects for *both* typicality ($p < 0.01$) and aspect ($p < 0.05$), while the interaction was again not significant ($p > 0.1$). Pairwise comparisons, similarly to the standard setting, revealed that TYP sentences had significantly lower scores for both the $PERF$ ($p < 0.05$) and the $IMPERF$ conditions ($p < 0.01$). Moreover, TYP sentences differed between $PERF$ and $IMPERF$ conditions, with the latter having significantly lower scores ($p < 0.05$). Finally, even NON_TYP sentences had lower surprisal scores in the $IMPERF$ condition, but the difference was only marginally significant ($p < 0.1$).

It should be noted that, in both settings, our results differed from those of Ferretti et al. (2007), as their study found no main effect of aspect and an interaction between aspect and typicality: atypical locations elicited significantly larger N400 components only in the imperfective condition, suggesting that imperfective verbs lead to very specific expectations on upcoming locations in human sentence processing, while expectations are way less defined with perfective verbs. By contrast, our results in the standard setting showed that BERT accurately modelled the typicality of locations without relying on the aspect of the verb, while in the context mask setting aspect influenced the predictions independently of typicality.

Finally, we checked the degree to which the typicality predictions of the model were influenced by the lexical frequencies of the target location words, which we extracted from a 2019 Wikipedia dump. [5] We found that the Surprisal scores in the **Ferretti07** dataset were not correlated at all in the standard setting, with a Spearman correlation of $\rho = -0.04$,

---

[5] `https://github.com/IlyaSemenov/wikipedia-word-frequency`.

while a weak but significant inverse correlation existed for the context mask setting at $\rho = -0.28$. This suggests that BERT's predictions are not influenced by word frequency when the entire context is available. However, when the number of contextual cues was reduced in the context mask setting, frequency might have played a more prominent role. From this point of view, it is interesting to observe the "errors" of the model: Example 4 shows the only four sentence pairs in which, in both the standard and the context mask settings, a lower score was assigned to a NON_TYP filler. Interestingly, in cases *a-c*, BERT assigns a lower Surprisal score to a more generic and frequent filler than the TYP one, which is more specific for the described event scenario. As for *d*, it can be observed that the two candidate locations (*desert-hole*) had very similar plausibility levels.

(4) a. The girl was skating/had skated in the rink (TYP) / ring (NON_TYP).
    b. The boy was tobogganing/had tobogganed down the hill (TYP) / street (NON_TYP).
    c. The tourist was browsing/had browsed in the shop (TYP) / park (NON_TYP).
    d. The snake was slithering/had slithered in the desert (TYP) / hole (NON_TYP).

The tendency of masked language models to select a generic and frequent word when faced with the alternative of a more specific and typical filler for the event scenario was also reported by Rambelli et al. (2020) in a logical metonymy interpretation task, e.g., when asked to predict a verb for the masked position in a sentence like *The auditor begins [MASK] the taxes*, they chose generic verbs like *doing* instead of more specific ones like *auditing*, which should be preferred in the given context.

5 Conclusions

In this study, we tested whether BERT exhibited aspect-related activation effects for event locations, and whether different degrees of location typicality were identified more easily in sentences in the imperfective aspect. Verb aspect, as shown in previous studies (Ferretti et al., 2007; Madden-Lombardi et al., 2017), plays an important role in the mental representation of an event; in particular, the imperfective aspect is related to the simulation of an on-going event, giving more saliency to all the entities involved, such as the event location.

Our results showed that BERT was able to identify typical locations for events, even when it had to differentiate them from plausible but less typical ones. However, the semantic information exploited by the Transformer for the task was not linked to the verb tense, as there were no differences found between the $PERF$ and the $IMPERF$ sets. In general, no aspect-related effects on the activation of event locations were observed.

Verb aspect played a role only when the Transformer was forced to focus on the verb phrase in the context mask setting. On the one hand, BERT was still able to distinguish between typical and non-typical locations. On the other hand, the imperfective aspect was associated with significantly lower Surprisal scores for both typicality conditions. Aspect did not interact with typicality: in particular, BERT did not predict the pattern observed in Ferretti et al. (2007)'s experimental study, for which specific expectations for event locations emerge only in processing imperfective sentences.

We take these results as preliminary evidence that BERT's predictions were somewhat sensitive to aspect-related differences, and could reflect some subtle nuances of argument typicality. Our implementation is available at a public repository.[6]

Acknowledgments

We thank Lei Siyu, Chen Jing and James Britton for their help in preparing the experimental materials, and the anonymous reviewers for their feedback.

Impact Statement

Our paper is the first to study the role of verb aspect in BERT predictions on upcoming arguments. Using locations as a case study, and through a comparison with the results of the experimental studies, we showed that BERT was able to discriminate between different degrees of typicality in context. Additionally, when less cues were available from the context, aspect had an important effect on the predictions, with imperfective verbs being associated with lower surprisal values for locations. We hope that our study has shed light on new research directions to investigate the interface of grammar and event knowledge using modern Transformer-based masked language models.

---

[6] `https://github.com/warnikchow/BERT-for-Surprisal`

References

Klinton Bicknell, Jeffrey L Elman, Mary Hare, Ken McRae, and Marta Kutas. 2010. Effects of Event Knowledge in Processing Verbal Arguments. *Journal of Memory and Language*, 63(4):489–505.

Emmanuele Chersoni, Ludovica Pannitto, Enrico Santus, Alessandro Lenci, and Chu-Ren Huang. 2020. Are Word Embeddings Really a Bad Fit for the Estimation of Thematic Fit? In *Proceedings of LREC*.

Marta Coll-Florit and Silvia P Gennari. 2011. Time in Language: Event Duration in Language Comprehension. *Cognitive Psychology*, 62(1):41–79.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.

Allyson Ettinger. 2020. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Todd R Ferretti, Marta Kutas, and Ken McRae. 2007. Verb Aspect and the Activation of Event Knowledge. *Journal of Experimental Psychology: Learning, memory, and cognition*, 33(1):182.

Todd R Ferretti, Ken McRae, and Andrea Hatherell. 2001. Integrating Verbs, Situation Schemas, and Thematic Role Concepts. *Journal of Memory and Language*, 44(4):516–547.

Stefan L Frank, Leun J Otten, Giulia Galli, and Gabriella Vigliocco. 2013. Word Surprisal Predicts N400 Amplitude During Reading. In *Proceedings of ACL*.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural Language Models as Psycholinguistic Subjects: Representations of Syntactic State. In *Proceedings of NAACL*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless Green Recurrent Networks Dream Hierarchically. In *Proceedings of NAACL*.

John Hale. 2001. A Probabilistic Earley Parser as a Psycholinguistic Model. In *Proceedings of NAACL*.

Mary Hare, Michael Jones, Caroline Thomson, Sarah Kelly, and Ken McRae. 2009. Activating Event Knowledge. *Cognition*, 111(2):151–167.

Josef Klafka and Allyson Ettinger. 2020. Spying on your Neighbors: Fine-grained Probing of Contextual Embeddings for Information about Surrounding Words. In *Proceedings of ACL*.

Marta Kutas and Steven A Hillyard. 1984. Brain Potentials During Reading Reflect Word Expectancy and Semantic Association. *Nature*, 307(5947):161–163.

Roger Levy. 2008. Expectation-based Syntactic Comprehension. *Cognition*, 106(3):1126–1177.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Carol J Madden and Rolf A Zwaan. 2003. How Does Verb Aspect Constrain Event Representations? *Memory & Cognition*, 31(5):663–672.

Carol Madden-Lombardi, Peter Ford Dominey, and Jocelyne Ventre-Dominey. 2017. Grammatical Verb Aspect and Event Roles in Sentence Processing. *Plos One*, 12(12).

Yuval Marton and Asad Sayeed. 2021. Thematic Fit Bits: Annotation Quality and Quantity for Event Participant Representation. *arXiv preprint arXiv:2105.06097*.

Kazunaga Matsuki, Tracy Chow, Mary Hare, Jeffrey L Elman, Christoph Scheepers, and Ken McRae. 2011. Event-based Plausibility Immediately Influences Online Language Comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 37(4):913.

Ken McRae, Mary Hare, Jeffrey L Elman, and Todd Ferretti. 2005. A Basis for Generating Expectancies for Verbs from Nouns. *Memory & Cognition*, 33(7):1174–1184.

Ken McRae and Kazunaga Matsuki. 2009. People Use their Knowledge of Common Events to Understand Language, and Do So as Quickly as Possible. *Language and Linguistics Compass*, 3(6):1417–1429.

Ken McRae, Michael J Spivey-Knowlton, and Michael K Tanenhaus. 1998. Modeling the Influence of Thematic Fit (and Other Constraints) in Online Sentence Comprehension. *Journal of Memory and Language*, 38(3):283–312.

Eleni Metheniti, Tim Van de Cruys, and Nabil Hathout. 2020. How Relevant Are Selectional Preferences for Transformer-based Language Models? In *Proceedings of COLING*.

Kanishka Misra, Allyson Ettinger, and Julia Taylor Rayz. 2020. Exploring BERT's Sensitivity to Lexical Cues using Tests from Semantic Priming. In *Findings of EMNLP*.

Kanishka Misra, Allyson Ettinger, and Julia Taylor Rayz. 2021. Do Language Models Learn Typicality Judgments from Text? In *Proceedings of CogSci*.

Grusha Prasad, Marten Van Schijndel, and Tal Linzen. 2019. Using Priming to Uncover the Organization of Syntactic Representations in Neural Language Models. In *Proceedings of CONLL*.

Giulia Rambelli, Emmanuele Chersoni, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. 2020. Comparing Probabilistic, Distributional and Transformer-Based Models on Logical Metonymy Interpretation. In *Proceedings of AACL-IJCNLP*.

Enrico Santus, Emmanuele Chersoni, Alessandro Lenci, and Philippe Blache. 2017. Measuring Thematic Fit with Distributional Feature Overlap. In *Proceedings of EMNLP*.

Asad Sayeed, Clayton Greenberg, and Vera Demberg. 2016. Thematic Fit Evaluation: An Aspect of Selectional Preferences. In *Proceedings of the ACL Workshop on Evaluating Vector-Space Representations for NLP*.

Nathaniel J Smith and Roger Levy. 2013. The Effect of Word Predictability on Reading Time Is Logarithmic. *Cognition*, 128(3):302–319.

TP Truitt and RA Zwaan. 1997. Verb Aspect Affects the Generation of Instrument Inferences. In *Proceedings of the Annual Meeting of the Psychonomic Society*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN Language Models Learn about Filler-Gap Dependencies? In *Proceedings of the EMNLP Workshop on Blackbox NLP*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-Art Natural Language Processing. *arXiv preprint arXiv:1910.03771*.

A Appendix A

The dataset by Ferretti et al. (2001) includes mean typicality ratings for 277 verb-location pairs on a Likert scale from 1 to 7, where 7 is the highest possible score (see examples in Table 1). The judgements were collected by asking to human subjects the following: *On a scale from 1 to 7, how common it is to* ***verb*** *in a* ***location****?*

Verb	**Location**	**Mean Rating**
gamble	casino	7.0
study	bedroom	5.8
marry	island	3.7
fish	pool	1.0

Table 1: Examples of the typicality judgements from the dataset by Ferretti et al. (2001). Scores range from 1 = not typical at all to 7 = very typical.

Two PhD students in Linguistics, both advanced speakers of English, voluntarily helped us in using these ratings to build the sentences of the **Ferretti01** dataset. First, we selected as typical only the verb-location pairs with a mean score $>= 4$.

Then, for each pair, we added a preposition that could be used to introduce the location in a prepositional complement. Each student took care of half of the pairs in the dataset, and then they revised each other's work. Since for this dataset we wanted BERT to compute the Surprisal scores for several candidate location fillers, the students tried to use prepositions that, given a verb, could go well with all its potential fillers in the dataset. Fillers that would have been much more likely than others given a verb-preposition pair were discarded from the dataset.

If the verb was strictly transitive, the students added a typical object, the first that came into their mind. In the end, we generated the final sentences of the dataset by randomly appending a personal pronoun subject (*He* or *She*), and we generated the sentence pairs for the two condition by varying the form of the verb: progressive past tense for the $IMPERF$ condition, past perfect tense for the $PERF$ condition.

Finally, another PhD student in Linguistics, native speaker of British English, checked that all the sentences were correct and plausible in English.

B Appendix B

	Estimates	S.E.	*p*
F01-Standard-Aspect	-0.24	0.28	0.39
F01-Context-Aspect	0.42	0.23	0.07 **.**
F07-Standard-Aspect	-0.03	0.26	0.54
F07-Standard-Typical	-3.21	0.36	< 0.001***
F07-Standard-AxT	0.38	0.52	0.46
F07-Context-Aspect	0.9	0.24	0.02*
F07-Context-Typical	-1.28	0.34	< 0.001***
F07-Context-AxT	0.24	0.48	0.62

Table 2: Results of linear mixed models on effects of aspect, typicality and the interaction of aspect and typicality (AxT), reported for both the Ferretti01 (F01) and the Ferretti07 (F07).

In Table 2, we present the tables with the full results of the linear mixed effect models for the **Ferretti01** and the **Ferretti07** dataset.

For reporting significance, we adopted the following notations: **.** for marginal significance at

$p < 0.1$, * for significance at $p < 0.05$, ** for significance at $p < 0.01$, and *** for significance at $p < 0.001$.

pairwise (F07Standard)	Estimates	S.E.	p
Impf-T:Impf-NT	-3.41	0.44	< 0.001***
Pf-T:Pf-NT	-3.02	0.44	< 0.001***
Impf-T:Pf-T	-0.16	0.36	0.97
Impf-NT:Pf-NT	0.22	0.36	0.93

Table 3: Results of pairwise comparisons in the standard setting (Impf: imperfective, Pf: perfective, T: typical, NT: non-typical).

We also report the pairwise comparisons on the **Ferretti07** dataset, both in the standard (Table 3) and in the context mask setting (Table 4).

pairwise (F07Context)	Estimates	S.E.	p
Impf-T:Impf-NT	-1.40	0.42	0.005***
Pf-T:Pf-NT	-1.16	0.42	0.029*
Impf-T:Pf-T	-1.02	0.34	0.014*
Impf:NT:Pf:NT	-0.78	0.34	0.09 .

Table 4: Results of pairwise comparisons on the context mask setting (Impf: imperfective, Pf: perfective, T: typical, NT: non-typical).

On the Interaction of Belief Bias and Explanations

Ana Valeria González[1], **Anna Rogers**[2], and **Anders Søgaard**[1]

University of Copenhagen
[1]Department of Computer Science
[2]Copenhagen Centre for Social Data Science
{ana,soegaard}@di.ku.dk
arogers@sodas.ku.dk

Abstract

A myriad of explainability methods have been proposed in recent years, but there is little consensus on how to evaluate them. While automatic metrics allow for quick benchmarking, it isn't clear how such metrics reflect human interaction with explanations. Human evaluation is of paramount importance, but previous protocols fail to account for *belief biases* affecting human performance, which may lead to misleading conclusions. We provide an overview of belief bias, its role in human evaluation, and ideas for NLP practitioners on how to account for it. For two experimental paradigms, we present a case study of gradient-based explainability introducing simple ways to account for humans' prior beliefs: models of varying quality and adversarial examples. We show that *conclusions about the highest performing methods change when introducing such controls*, pointing to the importance of accounting for belief bias in evaluation.

1 Introduction

Machine learning has become an integral part of our lives; from everyday use (e.g., search, translation, recommendations) to high-stake applications in healthcare, law, or transportation. However, its impact is controversial: neural models have been shown to make confident predictions relying on artifacts (McCoy et al., 2019; Wallace et al., 2019) and have shown to encode and amplify negative social biases (Manzini et al., 2019; Caliskan et al., 2017; May et al., 2019; Tan and Celis, 2019; González et al., 2020; Rudinger et al., 2018).

Explainability aims to make model decisions transparent and predictable to humans; it serves as a tool for model diagnosis, detecting failure modes and biases, and more generally, to increase trust by providing transparency (Amershi et al., 2019). While automatic metrics have been proposed to evaluate various properties of explanations such as faithfulness, consistency and agreement with human explanations (Atanasova et al., 2020; Robnik-Šikonja and Bohanec, 2018; DeYoung et al., 2020), these metrics do not inform us about human interaction with explanations.

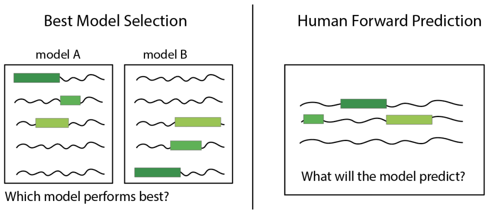

Figure 1: Evaluation protocols considered in this work

Doshi-Velez and Kim (2017) suggested *human forward prediction*, a simulation task in which humans are given an input and an explanation, and their task is to predict the expected model output, regardless of the gold answer. Recent studies include Nguyen (2018); Lage et al. (2019); **?**); Poursabzi-Sangdeh et al. (2021). Such protocols are widely used and can provide valuable insight into human understanding of explanations. However, prior work has not accounted for how humans' prior beliefs (*belief biases*) interact with the evaluation; simulating model decisions becomes an easier task when the model being evaluated makes predictions which align with human expectations. We argue that not considering belief bias in such protocols *may lead to misleading conclusions about which explainability methods perform best.*

Other protocols have evaluated participant's ability to select the best model based on explanations offered by different interpretability methods (e.g. decide which model would generalize 'in the wild') (Ribeiro et al., 2016a). However, comparisons have been made between a model which is clearly in line with human beliefs, and another which exploits spurious correlations diverging from human expec-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2930–2942
August 1–6, 2021. ©2021 Association for Computational Linguistics

tations. When differences are less obvious, humans may not be able to leverage their belief biases, and conclusions may change.

This paper, which includes evaluations for both of the previously mentioned tasks, closes an important gap: to the best of our knowledge, no prior work in NLP addresses the interaction of belief bias with current human evaluations of explainability.

Contributions. We provide an overview of belief bias meant to highlight its role in human evaluation and provide some preliminary ideas for NLP practitioners on how to handle such cases. Using *human forward prediction* and *best model selection* (Figure 1), we present a case-study where we compare two gradient-based explainability methods in the context of reading comprehension (RC), introducing conditions to take into account belief bias. We find that both explainability methods are helpful to participants in the standard settings (in line with most previous work), but the *conclusions about the best performing models change when incorporating additional control conditions*, reinforcing the importance of accounting for such biases.

2 Belief Bias

Belief bias is a type of cognitive bias, defined in psychology as *the systematic (non-logical) tendency to evaluate a statement on the basis of prior belief rather than its logical strength* (Evans et al., 1983; Klauer et al., 2000; Barston, 1986). Cognitive biases are not necessarily bad; they help us filter and process a great deal of information (Bierema et al., 2020), and have been widely studied in real human-decision making (Tversky and Kahneman, 1974; Kahneman, 2003; Furnham and Boo, 2011). However, in evaluations involving human participants, such biases may alter results and affect conclusions (Anderson and Hartzler, 2014; Wall et al., 2017).

Classic psychology studies of belief bias have assessed how prior beliefs affect syllogistic reasoning (Newstead et al., 1992; Klauer et al., 2000; Evans et al., 1983; Markovits and Nantel, 1989; Evans and BT). Consider the following example by Anderson and Hartzler (2014):

(a) *If all birds are animals, and if no animals can fly, then no birds can fly.*
(b) *If all cats are animals, and if no animals can fly, then no cats can fly.*

In syllogistic reasoning, the task for humans is to assess the *logical* validity of such arguments while ignoring believability. While both arguments are logically valid, most work converges on the finding that humans will rate argument (a) as invalid more often than (b), biased by the fact that the premise in (a) is less believable.

In psychology, belief bias has been tied to the *dual-processing* theory, which assumes that reasoning is performed by two competing cognitive systems: (1) *system 1* which takes care of fast, heuristic processes and (2) *system 2* which handles slower, more analytical processes (Evans, 2003; Trippas and Handley, 2018; Evans and Curtis-Holmes, 2005; Croskerry, 2009). Generally, humans tend to have a cognitive preference for relying on fast, intuitive *system 1* processes, rather than engaging in the slow and more analytical *system 2* processes. Belief bias is attributed to system 1 (Evans and Curtis-Holmes, 2005; Evans, 2008; Evans and Frankish, 2009; Stanovich and West, 2008) due to several factors, reviewed in detail by Evans (2003); Caravona et al. (2019).

For the purposes of NLP studies relying on crowd workers, one relevant finding is that **time pressures exacerbate reliance on previous beliefs** (Evans and Curtis-Holmes, 2005). Since crowd workers generally are incentivized to work as quickly as possible to maximize their hourly pay, reliance on belief bias is to be expected.

Another relevant finding for NLP is that threatening or negatively charged arguments (e.g. content violating political correctness and social norms) leads to greater engagement of system 2, whereas **neutral content leads to increased reliance on belief bias** (Goel and Vartanian, 2011; Klaczynski et al., 1997). Since NLP studies tend to be performed on neutral content such as passages from Wikipedia – content which may not sufficiently engage participants' system 2 processes – belief bias is more likely to play a role in human performance.

This study aims to highlight the phenomenon of belief bias to encourage NLP practitioners to assess the role it plays in their evaluations, and introduce mechanisms to account for belief bias effects. We illustrate how belief bias effects can significantly affect the results of human evaluation of explainability for two paradigms: *human forward prediction* and *best model selection*.

3 Related Work

Human forward prediction. Human forward prediction experiments have been recently presented in the context of synthetic data (Poursabzi-Sangdeh et al., 2021; Lage et al., 2019; Slack et al., 2019) to evaluate explainability methods for *their ability to make model decisions predictable to humans*. In this paradigm, humans are presented with explanations and tasked with predicting the model's decision regardless of the ground truth (Doshi-Velez and Kim, 2017).[1]

In NLP, Nguyen (2018) introduced human forward prediction for LIME explanations (Ribeiro et al., 2016b) of sentiment analysis of product reviews and correlated the results with automatic evaluations. Unlike with synthetic data, participants have prior beliefs on what the *true* outcome is. Since participants in Nguyen (2018) had no training phase to learn how explanations correlate with predictions and the model being evaluated sufficiently matched human behavior, humans likely relied *exclusively* on their prior knowledge and beliefs to complete the task at hand.

? improved on this protocol by adding a training phase. This is something we also do in our experiments (section 5), but it is unlikely to solve the belief bias problem because even after training, humans will naturally opt for fast, heuristic mechanisms (e.g. belief bias) in order to simplify tasks (Wang et al., 2019); this is particularly true if the model is high performing (i.e. likely aligns with human beliefs).

The protocol by **?** had another key feature: they leave out the explanations for the test data points. This would seem like an advantage for evaluating explainability methods in the context of reading comprehension where explanations can, in theory, simply highlight the answer span, making it easy to guess the model output from the explanations. However, it is easy to control for the amount of explanation provided by the explanation methods we compare; in our experiments below, we highlight the top 10 tokens with highest attribution scores. This key feature in their protocol is problematic for two reasons:

- It makes the human learning problem much harder, and we argue it is infeasible to expose participants to enough examples to make human forward prediction learnable (unless the task is made very easy on purpose; again by only evaluating high performing models). If it is not learnable, participants fall back on belief bias.
- It introduces a systematic bias between the training and test scenarios.

The protocol in **?** also does not randomize the order in which participants are exposed to problems with or without explanations.

We improve on the above protocol by introducing a condition which can help account for belief bias effects: evaluating explainability methods on low-quality models, the predictions of which substantially differ from human beliefs. This means that in order to succeed in the task, humans cannot simply rely on their previous beliefs, therefore, helping us assess the ability of explanations in helping humans to *realign* their expectations of model behavior. The predictions of reading comprehension models can also be made different from human answers by introducing distractor sentences that fool machine reading models, but not humans (Jia and Liang, 2017). If in human forward prediction, participants predict the true answer rather than spans in the distractor sentences, this suggests participants may be relying on their belief biases.

Best model selection. Ribeiro et al. (2016b) presented an evaluation of explainability methods for text classification, where explanations for decisions of two different models on the same instance are presented side by side, and humans decide which model is likely to generalize better. With some exceptions (Lertvittayakumjorn and Toni, 2019), there has not been much follow up work on this task, but this scenario is important: it mimicks the decisions about what model is *safer for deployment*. Ribeiro et al. (2016b) and Lertvittayakumjorn and Toni (2019) both make a single comparison between a model which clearly diverges from human intuition, and a model that generalizes and *aligns with humans' beliefs*. Accounting for the extent to which belief biases are leveraged (e.g. by introducing additional model comparisons where differences are not so obvious or where models are of low quality) is important in such paradigms, and can allow us to better evaluate where explanation methods may fail.

In the following sections, we show that introducing conditions which take into account belief biases can have an effect on the conclusions for both *human forward prediction* and *best model se-*

---

[1] Using synthetic data from fictitious domains effectively controls for belief bias (Lage et al., 2019; Slack et al., 2019). Slack et al. (2019), for example, evaluate explanations in the domain of recommending recipes and medicines to aliens.

lection. We emphasize that many other potential strategies can be introduced and this is largely dependent on the goals of the evaluation protocol; we merely provide one example case with the following strategies:

(1) Introducing low quality models which considerably diverge from humans' prior beliefs (*human forward prediction*)

(2) Introducing evaluation problems with distractor sentences (*human forward prediction*)

(3) Introducing model comparisons where relying on belief bias is not enough to obtain high performance (*best model selection*)

4 Experimental Setup

This section introduces the general setup of the experiments, with details specific to each experimental paradigm described in section 5 and section 6.

4.1 Models

We evaluate explanations produced by three BERT-based (Devlin et al., 2019) models:

(a) a high performing model (**HIGH**): BERT-base, fine-tuned on SQuAD 2.0. *This model is more aligned with human beliefs.*

(b) a medium performing model (**MEDIUM**): tinyBERT, a 6-layer distilled version of BERT (Jiao et al., 2020), fine-tuned on SQuAD 2.0. It performs about 20 F_1 points below HIGH. *This model somewhat aligns with human intuition, but performs significantly lower.*

(c) a low performing model (**LOW**): BERT-base, fine-tuned to always choose the first occurrence of the last word of the question. This system mimicks a rule-based system[2]; however, we evaluate gradient-based methods requiring a neural model. *This model diverges significantly from human beliefs.*

4.2 Data

We use SQuAD 2.0 (Rajpurkar et al., 2018), a RC dataset consisting of 150k factoid question-answer pairs, with texts coming from Wikipedia articles. We opt for this data as it contains short passages that can be read by humans in a short time. In the human forward prediction experiments, we refer to experiments using this data as ORIG. As described in section 2, Wikipedia texts could by themselves induce people to rely on their belief bias, but this particular dataset allows us to also introduce controls for the bias: the adversarial version of the data (Jia and Liang, 2017), has been shown to distract models but not humans. This means that in order to perform the task with success, humans need disregard their belief biases, and in some cases align with distractor sentences. We refer to this data in our simulation experiments as ADV.

4.3 Explainability Methods

We focus on gradient-based approaches, as they require no modifications to the original network, and are considerably faster than perturbation-based methods. We compare two explainability methods:

Gradients. Computing the gradient of the prediction output with regard to the features of the input is a common way to interpret deep neural networks (Simonyan et al., 2013) and capture relevant information regarding the underlying model.

Integrated gradients. Integrated gradients approach (IG) (Sundararajan et al., 2017) attributes an importance score to each input feature by approximating the integral of gradients of the model's output with respect to the inputs along the path, from the references to the inputs. IG was introduced to address the sensitivity issues which are present in vanilla gradients and implementation invariance.

5 Experiment 1: Human Forward Prediction

Human forward prediction for evaluating explainability was proposed by Doshi-Velez and Kim (2017). They argue that if a human is able to simulate the model's behavior, they understand *why* the model predicts in that manner. For the reasons previously outlined, we suspect that belief biases may be affecting performance and the conclusions once can draw from this task. We investigate this by asking the following: *Can humans predict model decisions, if model behavior considerably diverges from their own beliefs?*

Stimuli presentation. We include: (i) HIGH, which is finetuned to solve SQuAD 2.0 and (ii) LOW, which is finetuned to select the first appearance in the context of the last word in the question. We evaluate each of the two models twice: with or without adversarial data. We contrast using

---

[2] This model achieves about 0.90 F1 for this task, but in the results we show its performance on the actual RC task

vanilla gradients and IG with a baseline condition, in which no explanations are shown (BASELINE).

We highlight the top-10 tokens[3] with the highest attribution scores wrt. the start and end positions of the predicted span, and zero out the rest.[4] The two sets of tokens often overlap.

Participants were provided with a question and a passage (with or without explanations) and were told to pick the *shortest* span of text which matched the model prediction. They saw the actual model answers before the next example (done for both baseline and explanation conditions), which was an important part of training to infer model behavior. Before the model prediction was shown, their answers were locked to prevent any further changes. An example of our interface can be found in Figure 2 and the instructions are shown in Appendix A.

We ran these experiments on Amazon Mechanical Turk, recruiting participants with approval ratings greater than 95%[5] and ensuring different groups of participants per condition by specifying that participation is only allowed once, otherwise risking rejection[6]. We paid participants $5.25 for about 20 minutes of work (to ensure at least a $15 hourly pay) and obtained at least three annotations per example. The data included 120 unique questions divided into small fixed batches (the same questions across conditions). About 75% of questions are accurate in the HIGH model, and around 15% are accurate for the LOW model. In total, we obtained 4,300 data points across 123 participants (35 data points per participant).

Results. As humans often did not select the exact span that was provided as ground truth, we *manually* labeled the spans as correct or incorrect. We also inspected the impact of training in human forward prediction, e.g., the learning effect of multiple exposures on annotator accuracy. Both with vanilla gradients and integrated gradients, we observe an increase in the participants' accuracy at around 15 examples. In contrast, in our baseline condition, performance either stays constant or drops slightly. To reduce the noise introduced due to the training period, we remove the first 15 examples of each participant. The results without this preprocessing (Appendix A) suggest that **the effect of training differed across explainability methods**, as will be discussed later in the section.

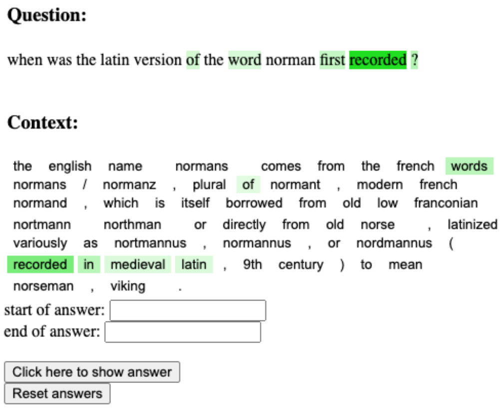

Question:

when was the latin version of the word norman first recorded ?

Context:

the english name normans comes from the french words normans / normanz , plural of normant , modern french normand , which is itself borrowed from old low franconian nortmann northman or directly from old norse , latinized variously as nortmannus , normannus , or nordmannus (recorded in medieval latin , 9th century) to mean norseman , viking .

start of answer:
end of answer:

Figure 2: Interface for Experiment 1 for LOW condition. To select model predictions, participants clicked on tokens to select the start and end of the span. Then they would see the actual model prediction.

Using the average human accuracies per example, we run a one-way ANOVA to test for significant differences across the groups. As we obtained statistically significant results, we then ran the Tukey honest significant difference (HSD) test (Tukey, 1949), comparing the means of every condition to the means of every other condition. The results are presented in Table 1.

As expected, in the absence of explanations (BASELINE), **humans rely on belief bias and predict the gold standard answer more often than the model prediction** (y in Table 1). Even with training (seeing the true model prediction), humans fail to catch onto the simple rule used by the LOW model, when no explanations are presented.

Overall, explanations derived from both of the gradient-based approaches lead to statistically significant improvements over the baseline. This indicates that the **explanations allow humans to realign their expectations of the model behavior**, better than with no explanations.

For HIGH-ORIG, the standard setting explored in previous evaluations, both IG gradients and vanilla gradients perform well, with IG gradients performing better. Given these results and the theoretical advantages of IG over vanilla gradients, one could arrive at the conclusion that IG are better for simulatability. However, **the differences between the two gradient-based methods are reversed in the conditions where humans cannot rely on their previous beliefs** (LOW). The gap be-

---

[3] Explanations should be *selective* (Mittelstadt et al., 2019)

[4] Ribeiro et al. (2016a) use the top 6 attributes; we opt for 10 given that our texts are slightly longer.

[5] Previous research has shown that proper filtering and selection of participants on Mechanical Turk, can be enough to ensure high quality data (Peer et al., 2014).

[6] We also remove such (few) repetitions at analysis

	Model	Human		
condition	F1	$\hat{y}$	y	sec
		Baseline		
Low-orig	0.17	0.16	0.48	33.9
Low-Adv	0.15	0.12	0.34	63.3
High-orig	0.79	0.45	0.46	34.6
High-Adv	0.66	0.38	0.48	36.1
		Integrated (IG)		
Low-orig	▨	*0.58	*0.22	*16.8
Low-Adv	▨	*0.63	*0.18	*22.3
High-orig	▨	***0.84**	*0.88	36.1
High-Adv	▨	***0.52**	* 0.35	*18.9
		Gradients		
Low-orig	▨	***0.69**	*0.06	32.6
Low-Adv	▨	***0.72**	*0.15	*25.6
High-orig	▨	*0.79	*0.81	47.4
High-Adv	▨	0.49	*0.60	48.4

Table 1: Human forward prediction results (Human($\hat{y}$)) for Low and High models, compared to no explanations (Baseline). Each experiment is run on vanilla SQuAD 2.0 data (Orig) and adversarial SQuAD 2.0 data (Adv). Human(y) is the dataset ground truth and an indicator of belief bias. Statistically significant results are indicated with an asterisk. Time is the average time per question. The best $\hat{y}$ results in each condition are bolded.

tween gradients and IG as large as 0.11, and being statistically significant. This finding is *surprising* and points again to the importance of not drawing incorrect conclusions about the best performing method using the standard paradigm.

Finally, in the HIGH conditions, model behavior decreases about 13% F1 score with the presence of **adversarial examples**, meaning that the model we used does get affected by adversarial inputs. We observe that human performance is considerably lower in HIGH-ADV as opposed to HIGH-ORIG. **With vanilla gradients, performance is more aligned with the ground truth labels than with model behavior,** showing that in this condition humans are also relying on their prior beliefs. **With IG, where performance is less aligned with prior beliefs (ground truth), the end performance increases**, but it seems that this condition is considerably more difficult for humans.

Effect of training. In BASELINE, training does not affect either the LOW or HIGH conditions (see Table 3 in Appendix A for the raw results). For the LOW model, multiple factors can be taking place (possibly at the same time): (1) the task is too far from the humans' beliefs and there is no mechanism to help participants realign their expectations, (2) participants may not be incentivized to seriously engage and look for patterns, (3) participants opt for a mixed strategy, where for some questions they go with their prior beliefs and for others, choice is random (as seen in their performance in y).

For HIGH conditions in BASELINE, performance remains higher than LOW but this is likely due to belief bias and not training, given that performance remains constant after removing the training data points. We hypothesize that for HIGH, instances where the model does not align to human intuition might be more detrimental than in explanation conditions. More specifically, if humans are aware that the model aligns with their beliefs after some examples but encounter instances where it doesn't (model is not 100% accurate), they will likely develop an expectation that the model is bound to make some errors, without any indication of when.

In addition, our raw results suggest IG required longer training. While this does not mean IG is a worse method than vanilla gradients, explanations derived from IG may have confused participants due to containing information which was irrelevant to them. It may be that experts (e.g. system engineers knowledgeable about neural networks) can take better advantage of such explanations; however, we leave this exploration of the interaction of human expertise with explanations as a direction for future work.

6 Experiment 2: Best Model Selection

This section presents the setup and results of our model selection experiments; a task where humans select the model that is more likely to succeed in the wild. We present the participants with the explanations from two models (High vs Low and High vs Medium), and ask them to decide which model is likely to perform better. As a follow-up, we also experimented with *soliciting explanations about what leads the worse model to fail*. Intuitively, comparative evaluation difficulty depends on how clear the difference is between the compared objects. Explanations should at least show the difference between a high-performing model and a low-performing one, enabling human participants to predict which is better (standard setting).

Stimuli presentation. We presented participants with saliency information from both models (a high performing model + one of the lower performing models), and their task was to determine which model performs best in the wild. We shuffled the order at random so that the best model would not remain in a fixed position. We obtain 120 samples (question-context pairs), and show the explanations next to each other as seen in Figure 1. The participants are told that the highlighted attributes are the words the model found important in making its decision. A screenshot of the UI is shown in Figure 4 in section B and the instructions provided to the participants are also shown in section B. These experiments were also ran on Amazon Mechanical Turk with the same general procedures and pay. The same subset of 120 examples is used in all conditions. We obtained at least three annotations per example and ended with a total of 1440 data points across 48 participants (30 examples each).

Results. For each example shown to annotators, we obtained the average accuracy scores and performed a standard T-test to compare the performance of the two methods. The results are shown in Table 2. Using explanations from both methods, when shown the High and Low model, humans are clearly able to correctly select the better one. With IG, humans achieve **0.95** accuracy on average, while with vanilla gradients they achieve **0.89**. The difference is *not* statistically significant. The fact that users are consistently able to discriminate between High and Low models is expected, and serves as a *sanity check* that these explanations are meaningful for humans.

Condition	Gradients	IG
High vs low	0.89	0.95
High vs medium*	0.85	0.52

Table 2: Both methods do well in (High vs Low). In high vs Medium, performance drops dramatically for IG. * = statistical significant difference ($\rho < 0.001$)

When the same experiment was repeated in the High vs Medium condition, we found clear and statistically significant differences between the two explainability methods. Using IG, participants reach only **0.52** accuracy, while with vanilla gradients their performance is **0.85**. This is *surprising*, given that the difference in performance between the two models is still quite large (about 20% F1); the expectation is that both methods would capture this difference relatively well. It appears that when both models *more or less* align with human beliefs, the task is much more difficult. To solve the task, humans now need to engage in more analytical thinking and cannot simply rely on belief biases to solve the task. We further investigate these differences through qualitative coding.

Qualitative analysis. After each instance, we asked participants to describe how the worse model will fail. We do not provide detailed guidelines in order to not further bias the participants by introducing specific criteria. The instructions given to the participants are shown in Appendix B.

We collected 1440 responses, which were all inspected manually to uncover categories (codes). After multiple iterations, we tagged each response with one code (categories are mutually exclusive, no response can be placed in two). A description of the categories and their distribution are shown in Figure 3, and examples of feedback per category are provided in the Appendix B.

In the High vs Low condition, feedback for both methods was generic (about 70-80% of the time), e.g., *model B is likely incorrect so it is worse*. This was expected: this task should be easy when model differences are large and humans can rely on their *system 1* processes to get through the task without thinking deeply about the explanations.

In the High vs Medium condition, the distribution of the feedback categories is very different. For IG, 50% of the time participants *felt* the highlighted tokens where irrelevant. This is not the case for gradients, where only about 15% of responses fell in that category. Additionally, for vanilla gradients, 50% of feedback is generic, signaling that in this condition, it may have been an easy task as well; explanations are making model behavior clear enough. It remains an open question whether IG explanations may in fact be more faithful to the model reasoning. In that case, *expert users* (e.g. a system engineer debugging a system) may not find IG attributions irrelevant and would be able take better advantage of the information provided. For this reason, other kinds of human participants may show different results. Nevertheless, as evaluating on non-experts (crowdsourced workers for example) is common, this preliminary result is important: it shows that **conclusions can shift dramatically when introducing additional model comparisons** which reduce the participants' ability to rely on prior knowledge.

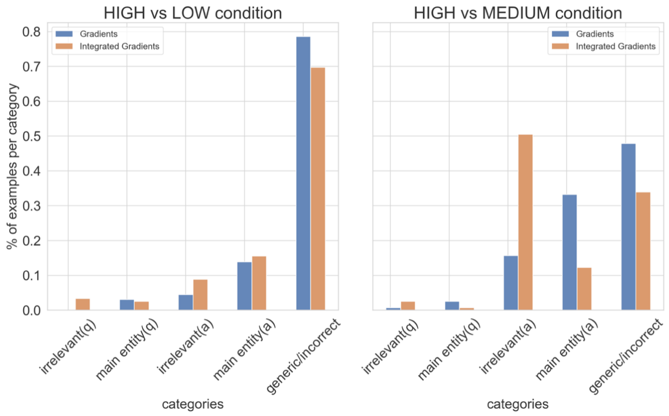

Category	Description
irrelevant(q)	The model puts emphasis on irrelevant entities in the **question**.
Main entity (q)	The model does not put emphasis on the main entity in the **question**
irrelevant(a)	The model puts emphasis on irrelevant entities in the **answer**.
Main entity (a)	The model does not put emphasis on the main entity in the **answer**
generic/ incorrect	"Model is worse, because it is incorrect."

Figure 3: Feedback categories and their distribution. We observed that the HIGH vs MEDIUM condition results are considerably different from the HIGH vs LOW condition, with more participants giving generic answers for vanilla gradients, and emphasizing the irrelevant terms highlighted in the IG condition.

7 Discussion: Mitigating Belief Bias

This study introduced *additional conditions* in which the human participants could not rely on their belief biases to facilitate the task at hand. We presented a case study on evaluating reading comprehension models in model selection and human forward prediction paradigms, and we showed that this simple addition led to different conclusions in the evaluation and a better understanding of how humans interacted with explanations. Other tasks and paradigms might call for different setups, but generally including conditions with models of varying quality would be helpful both for the purposes of bias control, and for simulation of real-life use of explainability techniques to support decisions about which model is safer to deploy.

To conclude, we will briefly mention other directions for mitigating belief biases that can also be explored in future work and which should be kept in mind when developing evaluation protocols for explainability.

Reducing ambiguity. Ambiguity of task instructions leads humans to align interpretations to their own prior beliefs (Heath and Tversky, 1991); this may lead to misinterpretation and results which do not reflect the intended interaction with explanations. Ambiguity may also be present in other parts of the evaluation setup. For example, Lamm et al. (2020) evaluate the effectiveness of explanations in helping humans detect model errors for open-domain QA, but the data they use contains questions where multiple answers can be true. Users may deem an answer to be correct or incorrect based on their understanding of the question, which makes the effect of explanations blurry. Removing ambiguous instances from the data can be a way of reducing such confounds.

Removing time constraints. Time constraints exacerbate reliance of system 1 processes, which leads to humans relying on belief biases. In crowd-sourced evaluations, it is common practice to to provide workers with enough time to perform tasks, but workers may have intrinsic motivations for performing tasks quickly. A major challenge for evaluation research with crowd workers is creating better incentives for engaging in system 2 processes, e.g. pay schemes which encourage workers to be more analytical and accurate (Bansal et al., 2019).

Include fictitious domains. Using data from domains from which subjects have no prior beliefs e.g. fictitious domains, may be an efficient way of controlling for belief bias in some tasks[7]. This strategy has been used outside of NLP (Poursabzi-Sangdeh et al., 2021; Lage et al., 2019; Slack et al., 2019), where subjects are asked to imagine alternative worlds such as scenarios involving aliens. In QA for example, one could introduce context-question pairs that describe facts about fictitious scenarios that sufficiently differ from human reality.

8 Conclusion

The main contribution of this paper is bringing the discussion of belief bias from psychology into the context of evaluating explainability methods in

---

[7] Again, we emphasize that some strategies are task dependent; fictitious domains may not be relevant in some tasks.

NLP. Belief bias is a phenomenon which plays a role in human decision making and which interacts with previous evaluations in a way which may affect the conclusions we draw from these paradigms. We provide an overview of belief bias, making a connection between findings in psychology and the field of NLP, and present a case study of evaluating explanations for BERT-based reading comprehension models. We show that introducing models of various quality and adversarial examples can help to account for belief bias, and that introducing such conditions affects the conclusions about which explainability method works better. Finally, we provide additional insights and ideas for how to account for belief bias effects in human evaluation.

9 Broader Impact Statement

The work presented here makes strides towards a better understanding about the interaction of humans with explanations of model decisions. We have highlighted a phenomenon studied in psychology with hope that this opens the door to more NLP research involving a wider and more interdisciplinary understanding of humans, and the effect of explainability.

This study involved human participants recruited on Mechanical Turk platform. No personally identifiable data was collected from the participants, they were made aware that the data would only be used for research, and they were not exposed to any emotionally traumatizing or offensive stimuli. We ensured a minimum $15 hourly wage.

Acknowledgements

We thank the reviewers for their insightful feedback for this and previous versions of this paper. This work is partly funded by the Innovation Fund Denmark.

References

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-ai interaction. In *CHI*.

Richard B Anderson and Beth M Hartzler. 2014. Belief bias in the perception of sample size adequacy. *Thinking & Reasoning*, 20(3):297–314.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.

Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. 2019. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2429–2437.

Julie Linda Barston. 1986. An investigation into belief biases in reasoning.

Andrea Bierema, Anne-Marie Hoskinson, Rosa Moscarella, Alex Lyford, Kevin Haudek, John Merrill, and Mark Urban-Lurain. 2020. Quantifying cognitive bias in educational researchers. *International Journal of Research & Method in Education*, pages 1–19.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Laura Caravona, Laura Macchi, Francesco Poli, Michela Vezzoli, Miriam AG Franchella, and Maria Bagassi. 2019. How to get rid of the belief bias: boosting analytical thinking via pragmatics. *Europe's Journal of Psychology*, 15(3):595–613.

Pat Croskerry. 2009. A universal model of diagnostic reasoning. *Academic medicine*, 84(8):1022–1028.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

J St Evans and Handley BT. Sj and harper, c. 2001. *Necessity, possibility and belief: A study of syllogistic reasoning. Quarterly Journal of Experimental Psychology A*, 54:935–958.

J St BT Evans, Julie L Barston, and Paul Pollard. 1983. On the conflict between logic and belief in syllogistic reasoning. *Memory & cognition*, 11(3):295–306.

Jonathan St BT Evans. 2003. In two minds: dual-process accounts of reasoning. *Trends in cognitive sciences*, 7(10):454–459.

Jonathan St BT Evans. 2008. Dual-processing accounts of reasoning, judgment, and social cognition. *Annu. Rev. Psychol.*, 59:255–278.

Jonathan St BT Evans and Jodie Curtis-Holmes. 2005. Rapid responding increases belief bias: Evidence for the dual-process theory of reasoning. *Thinking & Reasoning*, 11(4):382–389.

Jonathan St BT Evans and Keith Ed Frankish. 2009. *In two minds: Dual processes and beyond.* Oxford University Press.

Adrian Furnham and Hua Chu Boo. 2011. A literature review of the anchoring effect. *The journal of socio-economics*, 40(1):35–42.

Vinod Goel and Oshin Vartanian. 2011. Negative emotions can attenuate the influence of beliefs on logical reasoning. *Cognition and Emotion*, 25(1):121–131.

Ana Valeria González, Maria Barrett, Rasmus Hvingelby, Kellie Webster, and Anders Søgaard. 2020. Type B reflexivization as an unambiguous testbed for multilingual multi-task gender bias. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2637–2648, Online. Association for Computational Linguistics.

Chip Heath and Amos Tversky. 1991. Preference and belief: Ambiguity and competence in choice under uncertainty. *Journal of risk and uncertainty*, 4(1):5–28.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. pages 4163–4174.

Daniel Kahneman. 2003. A perspective on judgment and choice: mapping bounded rationality. *American psychologist*, 58(9):697.

Paul A Klaczynski, David H Gordon, and James Fauth. 1997. Goal-oriented critical reasoning and individual differences in critical reasoning biases. *Journal of Educational Psychology*, 89(3):470.

KC Klauer, J Musch, and B Naumer. 2000. On belief bias in syllogistic reasoning. *Psychological Review*, 107.

Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2019. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*.

Matthew Lamm, Jennimaria Palomaki, Chris Alberti, Daniel Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. 2020. Qed: A framework and dataset for explanations in question answering.

Piyawat Lertvittayakumjorn and Francesca Toni. 2019. Human-grounded evaluations of explanation methods for text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5195–5205, Hong Kong, China. Association for Computational Linguistics.

Thomas Manzini, Lim Yao Chong, Alan W Black, and Yulia Tsvetkov. 2019. Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings. pages 615–621.

Henry Markovits and Guilaine Nantel. 1989. The belief-bias effect in the production and evaluation of logical conclusions. *Memory & cognition*, 17(1):11–17.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Brent Mittelstadt, Chris Russell, and Sandra Wachter. 2019. Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 279–288.

Stephen E Newstead, Paul Pollard, Jonathan St BT Evans, and Julie L Allen. 1992. The source of belief bias effects in syllogistic reasoning. *Cognition*, 45(3):257–284.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

Eyal Peer, Joachim Vosgerau, and Alessandro Acquisti. 2014. Reputation as a sufficient condition for data quality on amazon mechanical turk. *Behavior research methods*, 46(4):1023–1031.

Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. 2021. Manipulating and measuring model interpretability. pages 1–52.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

Marko Robnik-Šikonja and Marko Bohanec. 2018. Perturbation-based explanations of prediction models. In *Human and machine learning*, pages 159–175. Springer.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. pages 8–14.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

D Slack, SA Friedler, CD Roy, and C Scheidegger. 2019. Assessing the local interpretability of machine learning models.

Keith E Stanovich and Richard F West. 2008. On the relative independence of thinking biases and cognitive ability. *Journal of personality and social psychology*, 94(4):672.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. pages 3319–3328.

Yi Chern Tan and L Elisa Celis. 2019. Assessing social and intersectional biases in contextualized word representations. In *Advances in Neural Information Processing Systems*, pages 13230–13241.

Dries Trippas and Simon J Handley. 2018. The parallel processing model of belief bias: Review and extensions.

John W Tukey. 1949. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114.

Amos Tversky and Daniel Kahneman. 1974. Judgment under uncertainty: Heuristics and biases. *science*, 185(4157):1124–1131.

Emily Wall, Leslie M Blaha, Lyndsey Franklin, and Alex Endert. 2017. Warning, bias may occur: A proposed approach to detecting cognitive bias in interactive visual analytics. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 104–115. IEEE.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable ai. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15.

A Experiment 1: Human Forward Prediction

Below we show the instructions provided to the participants, as well as an example of the saliency maps presented to participants for adversarial examples.

Instructions. Question-answering systems are a particular form of artificial intelligence. The task here is for you to learn to predict how the system answers questions. In other words, when in a bit, you are presented with questions, the task is not to provide the right answer, but to guess the answer the system provided. For each question, you will also see a context paragraph. The answer is a span of text in this paragraph. Instead of writing out the answer, you can simply mark the relevant span.

If you want to select a new answer, please click *reset answer*, if you are ready to see the model answer, please click *show answer*. Note that your answer will lock at that time.

Raw Results. In our evaluation, we use the first 15 points as training, therefore, we discard them from the main evaluation but show them in this section. Overall, we see that training, for the most part has a positive effect, or not so much of an effect. These scores can be seen in Table 3.

	MODEL	HUMAN		
		BASELINE		
LOW-ORIG	0.17	0.14	0.52	52.27
LOW-ADV	0.15	0.10	0.36	54.36
HIGH-ORIG	0.79	0.53	0.58	37.12
HIGH-ADV	0.66	0.35	0.48	47.64
		INTEGRATED (IG)		
LOW-ORIG	▨	*0.34	0.35	41.68
LOW-ADV		*0.36	0.28	44.38
HIGH-ORIG	▨	*0.71	0.76	46.87
HIGH-ADV		0.46	0.47	42.99
		GRADIENTS		
LOW-ORIG	▨	***0.64**	*0.09	*32.16
LOW-ADV		***0.63**	0.23	*30.05
HIGH-ORIG	▨	***0.82**	*0.84	44.65
HIGH-ADV		***0.57**	*0.62	*52.30

Table 3: Raw scores, before removing data points on training session

B Experiment 2: Best Model Selection

Below we show the instructions given to the participants, and more details about the qualitative analysis of the feedback we obtained.

Instructions. Question-answering (QA) systems are a particular form of artificial intelligence. We have trained two QA systems and have extracted the most important words the model uses to make its final decision. Based on these highlighted words, your task is to select the model that you think is more likely to perform best. Additionally, please write how the low-performing model fails and/or how it could be better (try to be detailed)

User Interface. An example instance, as shown to the participants, can be seen in Figure 4.

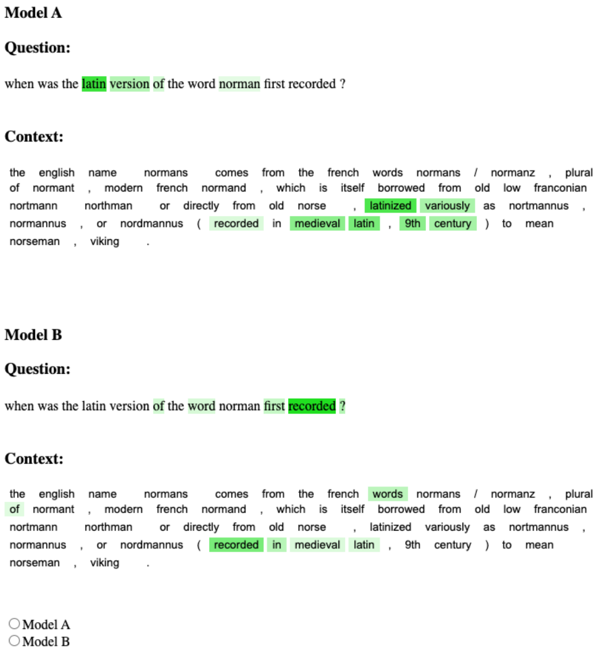

Model A

Question:

when was the latin version of the word norman first recorded ?

Context:

the english name normans comes from the french words normans / normanz , plural of normant , modern french normand , which is itself borrowed from old low franconian nortmann northman or directly from old norse , latinized variously as nortmannus , normannus , or nordmannus (recorded in medieval latin , 9th century) to mean norseman , viking .

Model B

Question:

when was the latin version of the word norman first recorded ?

Context:

the english name normans comes from the french words normans / normanz , plural of normant , modern french normand , which is itself borrowed from old low franconian nortmann northman or directly from old norse , latinized variously as nortmannus , normannus , or nordmannus (recorded in medieval latin , 9th century) to mean norseman , viking .

○ Model A
○ Model B

Figure 4: Experiment 1 UI: LOW(bottom) vs HIGH(top) condition.

Qualitative analysis of feedback. In Table 4, we include a few examples of the sentence that were categorized using the qualitative codes. Unsurprisingly, once participants found a strategy for giving feedback , they mostly stuck to it.

After categorizing all the feedback into each category, we visualize the distribution per condition. This can be found in Figure 3. We find that for the HIGH vs LOW conditions, the distribution is very similar between gradients and integrated gradients. Many participants gave very generic feedback , for example by simply saying that ”model A is better because it is correct, and model B is wrong”. This was not surprising, as here the differences were supposed to be clear and it is likely most participants did not have to think too hard before making

QUALITATIVE CODES	EXAMPLES
Irrelevant (q)	**1.** Model A only extracted some important words but also some punctuations in the question which is insufficient to derive to a good answer. Model B extracted a number of key important words that would lead to the correct answer. **2.** Option b chose quantitative statements, while option A seems confused about what it's looking for since it highlights all sorts of things in the question.
main entity (q)	**1.** The words "year" and "norman" in the question were not extracted by Model A. The Model will not be able get the correct answer without knowing what to look for. **2.** The question was asking about the year lavoisier's work was published but neither of the key words in this question were highlighted. Model A had no idea where to locate the answer without considering those key words.
main entity (a)	**1.** The answer requires a year; it hasn't highlighted any years as part of the answer. **2.** Answer needed to be a name and option A chose nothing that could be a name.
Irrelevant (a)	**1.** Model B has highlighted many extra words in the answer **2.** Both models selected the correct terms, but model A selected more irrelevant terms in the answer too, so it's less likely to choose the correct one from those numerous options. **3.** B highlighted the answer but also too much unneeded info.
Generic/correctness	**1.** Model A does not highlight the right answer **2.** Model B is wrong and model A is correct

Table 4: Examples of some of the feedback categorized into these classes

a decision. However, the distribution is very different for the HIGH vs MEDIUM conditions. Here, for standard gradients, the feedback followed a similar pattern as in the previous condition, but about 30% less examples received generic feedback than before. For integrated gradients, most examples received feedback regarding the irrelevant terms being highlighted, showing that even when the difference in performance between models is large (20 F1 points), this method makes the distinction difficult for the best model selection task.

Combining Static Word Embeddings and Contextual Representations for Bilingual Lexicon Induction

Jinpeng Zhang[1], **Baijun Ji**[1], **Nini Xiao**[1], **Xiangyu Duan**[1]*, **Min Zhang**[1]
Yangbin Shi[2], **Weihua Luo**[2]
[1] Institute of Aritificial Intelligence, School of Computer Science and Technology,
Soochow university
[2] Alibaba DAMO Academy
{jpzhang1,bjji,nnxiaoxiao}@stu.suda.edu.cn; {xiangyuduan,minzhang}@suda.edu.cn;
taiwu.syb@taobao.com; weihua.luowh@alibaba-inc.com

Abstract

Bilingual Lexicon Induction (BLI) aims to map words in one language to their translations in another, and are typically through learning linear projections to align monolingual word representation spaces. Two classes of word representations have been explored for BLI: static word embeddings and contextual representations, but there is no studies to combine both. In this paper, we propose a simple yet effective mechanism to combine the static word embeddings and the contextual representations to utilize the advantages of both paradigms. We test the combination mechanism on various language pairs under the supervised and unsupervised BLI benchmark settings. Experiments show that our mechanism consistently improves performances over robust BLI baselines on all language pairs by averagely improving 3.2 points in the supervised setting, and 3.1 points in the unsupervised setting[1].

1 Introduction

Bilingual Lexicon Induction (BLI) aims to find bilingual translation lexicons from monolingual corpora in two languages (Haghighi et al., 2008; Xing et al., 2015; Zhang et al., 2017a; Artetxe et al., 2017; Conneau et al., 2017), and is applied on numerous NLP tasks such as POS tagging (Zhang et al., 2016), parsing (Xiao and Guo, 2014), and machine translation (Irvine and Callison-Burch, 2013; Qi et al., 2018).

Most work on BLI learns a mapping between two static word embedding spaces, which are pretrained on large monolingual corpora (Ruder et al., 2019). Both linear mapping (Mikolov et al., 2013; Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017) and non-linear mapping (Mohiuddin et al., 2020) methods have been studied to align the two spaces. Recently, other than the static word embeddings, contextual representations are used for BLI due to the significant progresses on cross-lingual applications (Aldarmaki and Diab, 2019; Schuster et al., 2019). Although the static word embeddings and the contextual representations exhibit properties suited for alignment, there is no works to combine the two paradigms.

On one hand, the static word embeddings have been widely used for BLI, but one specific embedding mapping function does not ensure that in all conditions, words in a translation pair are nearest neighbors in the mapped common space. On the other hand, the contextual representations contain rich semantic information beneficial for alignment, but the dynamic contexts of word tokens pose a challenge for aligning word types.

In this paper, we propose a combination mechanism to utilize the static word embeddings and the contextual representations simultaneously. The combination mechanism consists of two parts. The first part is the unified word representations, in which a spring network is proposed to use the contextual representations to pull the static word embeddings to better positions in the unified space for easy alignment. The spring network and the unified word representations are trained via a contrastive loss that encourages words of a translation pair to become closer in the unified space, and penalizes words of a non-translation pair to be farther. The second part is the weighted interpolation between the words similarity in the unified word representation space and the words similarity in the contextual representation space.

We test the proposed combination mechanism in both the supervised BLI setting which can utilize a bilingual dictionary as the training set, and the unsupervised BLI setting which does not allow using any parallel resources as supervision signal.

---

* Corresponding Author.

[1] Code is released at https://github.com/zjpbinary/CSCBLI

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2943–2955
August 1–6, 2021.

On BLI benchmark sets of multiple language pairs, our combination mechanism performs significantly better than systems using only the static word embeddings and systems using only the contextual representations. Our mechanism improves over robust BLI baselines on all language pairs, achieving average 3.2 points improvement in the supervised setting, and average 3.1 points improvement in the unsupervised setting.

2 Background

The early works on Bilingual Lexicon Induction (BLI) date back to several decades ago, including feature-based retrieval (Fung and Yee, 1998), distributional hypothesis (Rapp, 1999; Vulić and Moens, 2013), and decipherment (Ravi and Knight, 2011). Following Mikolov et al. (2013), which pioneered the embedding based BLI method, word representation based method becomes the dominant approach, and can be categorized into two classes: static word embedding based method, and contextual representation based method.

- **Static Word Embedding Based Method**

  Word embeddings of different languages are pre-trained in large monolingual corpora independently. Then a mapping function is applied to align the embedding spaces of the two languages (Mikolov et al., 2013; Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017).

  We follow one robust BLI system VecMap (Artetxe et al., 2018a,b), which maps both source space and target space into a third common space. Let E_x and E_y be the word embedding matrices in two languages for a given bilingual dictionary such that their ith rows are the embeddings of words of the ith translation pair in the dictionary. The training objective is to find mapping functions W_x and W_y such that

$$W_x^\star, W_y^\star = \underset{W_x, W_y \in M_d(\mathbb{R})}{\arg\max} \; cos(E_x W_x, E_y W_y) \tag{1}$$

  where d is the dimension of the embeddings, $M_d(\mathbb{R})$ is the space of $d \times d$ matrices of real numbers. The optimal W_x and W_y maximizes the cosine similarity between words of each translation pair in the mapped common space. In the unsupervised version where no bilingual dictionary is given, an artificial dictionary is initialized and iteratively updated through training W_x and W_y according to equation (1) (Artetxe et al., 2018b).

  Both mapping functions are constrained to be orthogonal during training by setting $W_x = U$ and $W_y = V$, where $U\Sigma V^T = X^T Y$ is the singular value decomposition of $X^T Y$. Such orthogonal constraint is based on the assumption that the source embedding space and the target embedding space are isometric, which is a particularly strong assumption that does not hold in all conditions (Zhang et al., 2017b; Søgaard et al., 2018). To depart from the isometry assumption, Patra et al. (2019) uses a semi-supervised technique that leverages both seed dictionary and a larger set of unaligned word embeddings, Mohiuddin et al. (2020) uses a non-linear mapping function that is not constrained to be orthogonal.

  We propose another method to relax the isometry assumption by combining the contextual representations with the word embeddings to compensate the shortage of the overly strong assumption.

- **Contextual Representation Based Method**

  Contextual representations can be obtained through multilingual pre-training, which encodes whole sentence and outputs contextual representation for each word (Devlin et al., 2019; Lample and Conneau, 2019). Due to the rich context information contained in the contextual representations, there are endeavors to align them in different languages (Schuster et al., 2019; Aldarmaki and Diab, 2019; Wang et al., 2020; Kulshreshtha et al., 2020; Cao et al., 2020).

  Since a word may appear in different sentences with different contexts, Schuster et al. (2019) use an average anchor to summarize multiple contexts for a word type and align the anchors of different languages, while other works aim to align each individual context representation based on parallel corpora, including learning alignment on sentence level representations and applying the learned mapping on word level contextual representations (Aldarmaki and Diab, 2019), using word alignments in a parallel corpora to learn the mapping for word contextual representations (Wang et al., 2020), and directly minimizing

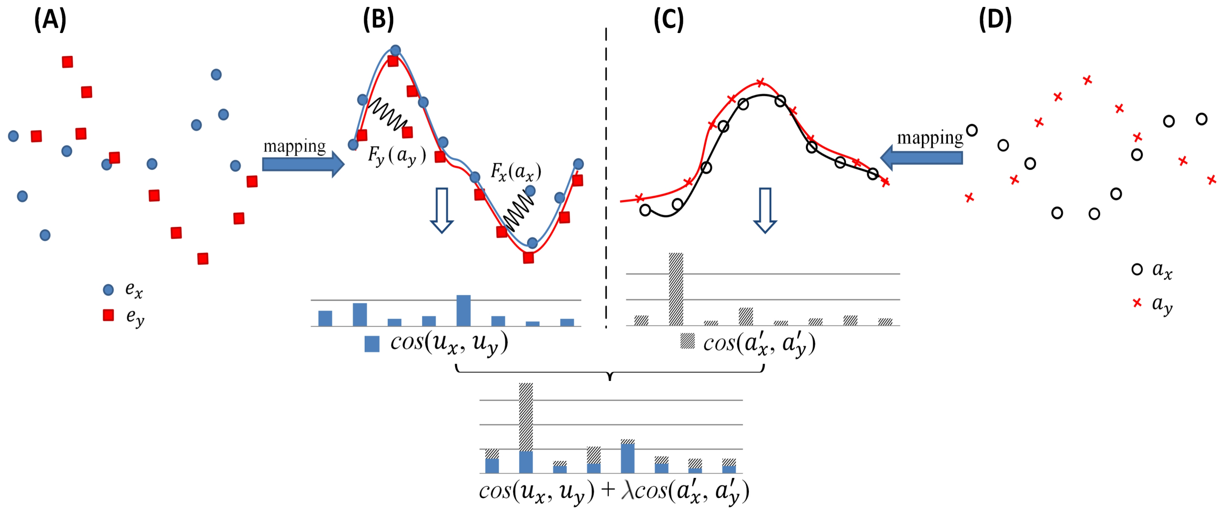

Figure 1: The illustration of the proposed combination mechanism. (A) is the static word embedding space, where e_x and e_y are the source and target embeddings, respectively. (B) is the unified word representation space which consists of the mapped word embeddings pulled by a spring network F_x/F_y with the contextual representations as input. We just depict two springs for illustration. (C) is the mapped contextual representation space. (D) is the original contextual representation space. a_x and a_y are the source and target contextual representations, i.e., the average anchors, respectively. In the similarity interpolation shown in the bottom, u_x and u_y are the unified word representations in the two languages, $a_x^{'}$ and $a_y^{'}$ are the mapped contextual representations in the two languages, cos denotes the cosine similarity function, λ denotes the weight.

the distance between two contextual representations of an aligned word pair in parallel corpora without mapping (Cao et al., 2020).

We adopt the average anchor method for the contextual representations (Schuster et al., 2019), which does not depend on parallel corpora. Let the contextual representation of a source word x in context c_i be denoted as r_{x,c_i}. If x appears a total of p times in the source corpus, the average anchor for x across all contexts is:

$$a_x = \frac{\sum_{i=1}^{p} r_{x,c_i}}{p} \tag{2}$$

Similar to the mapping for the static word embeddings, we conduct mapping for the average anchors. Let A_x and A_y be the matrices of average anchors in two languages with correspondence to word pairs from a given bilingual dictionary. The mapping functions V_x and V_y are optimized by maximizing $cos(A_x V_x, A_y V_y)$, where A_x and A_y are fixed, V_x and $V_y \in M_{d'}(\mathbb{R})$, and d' is the dimension of the contextual representations.

Besides the above methods, there is another direction that extracts word alignments in pseudo parallel corpora for BLI. The pseudo parallel corpora are built by either the unsupervised machine translation (Artetxe et al., 2019) or the unsupervised bitext mining (Shi et al., 2021). Both methods need significant computation overload or use monolingual corpora that are magnitudes larger than ours, and are beyond the scope of this paper that focuses on representation based methods.

3 Proposed Combination Mechanism for BLI

Since there is no work to combine both the static word embeddings and the contextual representations, we propose a combination mechanism illustrated in Figure 1. The mechanism first builds a unified word representation space that unifies the static word embeddings and the contextual representations, then performs similarity interpolation between the unified space and the contextual space.

3.1 The Unified Word Representations

As shown in Figure 1, the original word embedding space (A) is mapped to (B) through the mapping functions. Since the mapping functions are orthogonal, (A) is just rotated to (B). Notice that the spaces of the two languages are not necessarily isometric everywhere, some words in certain translation pairs are still far away from each other after rotation. To pull words in a translation pair getting closer, we propose a spring network that can pull the mapped embedding points to better positions such that words in a translation pair are nearest neighbors of each other. Since the contextual representations contain rich context information that can be used as the flexible adjustment, the spring network takes the contextual representations as input, and outputs offsets for the word embeddings.

Specifically, in the unified word representations, the mapped word embeddings are pulled to new positions by offsets, which are produced by the spring network with the the contextual representations as input:

$$\begin{aligned} U_x &= E_x^{'} + \gamma_1 \odot F_x(A_x) \\ U_y &= E_y^{'} + \gamma_2 \odot F_y(A_y) \end{aligned} \quad (3)$$

where U_x and U_y are the unified word representations, $E_x^{'}$ and $E_y^{'}$ are the mapped word embeddings, F_x and F_y are the spring networks, and $\gamma_{1\mathrm{or}2}$ is the weight vector, which is used to element-wisely multiply each row of the output of the spring network. Take the source side for example, the mapped word embedding matrix $E_x^{'}$ is added with a weighted offset produced by the spring network F_x on the contextual representation (i.e., the average anchor) matrix A_x.

The Spring Network stacks two feedforward layers with Tanh activations on top of the contextual representation matrices. The first layer transforms the dimension of the contextual representation d' to the dimension of the word embedding d. Equations (4-5) list the network structure of both sides.

$$A_x^1 = \phi(\theta_x^0(A_x)), \quad A_y^1 = \phi(\theta_y^0(A_y)) \quad (4)$$

$$A_x^2 = \phi(\theta_x^1(A_x^1)), \quad A_y^2 = \phi(\theta_y^1(A_y^1)) \quad (5)$$

where ϕ denotes the Tanh activation, and θ denotes the feedforward layer. $A_{x/y}^2$ is the output of the spring network, and fulfills as the offset distance to compensate the deviation of words in each translation pair in the mapped word embedding space.

Since we use cross-lingual pre-training (Lample and Conneau, 2019) to generate the contextual representations, which are actually BPE's (Sennrich et al., 2016) contextual representations, we have to form the contextual representations in the word level. Suppose a word x has q BPEs, and x appears p times in the monolingual corpus, then the word level contextual representation $a_x = \sum_{i=1}^{p} (\sum_{j=1}^{q} r_{b_j,c_{i,j}}/q)/p$, where $r_{b_j,c_{i,j}}$ denotes the representation of the jth BPE with the ith context $c_{i,j}$. a_x actually averages q BPEs' representation at first, then averages p contexts. After this cascaded averaging, it constitutes one row of A_x.

Contrastive Training is used to train the spring networks F_x and F_y with the pre-trained mapped word embeddings and the contextual representations fixed in the unified space. Basically, through the spring adjustment, the training encourages parallel words to get closer, and drives non-parallel words to be farther. It is divided into two scenarios: supervised contrastive training and unsupervised contrastive training.

- In the supervised contrastive training, given a bilingual dictionary with I translation pairs, the contrastive loss is:

  $$\begin{aligned} \mathcal{L}_{sup} = -\sum_{i=1}^{I}(J \times cos(u_x^i, u_y^i) \\ -\sum_{j=1}^{J} cos(u_x^i, u_{\bar{y}}^j)) \end{aligned} \quad (6)$$

  where u_x^i and u_y^i are the unified representations corresponding to the ith entry of the given bilingual dictionary.

  In equation (6), (u_x^i, u_y^i) is the positive translation pair according to the given dictionary, and the cosine similarity of this pair is maximized during training, while $(u_x^i, u_{\bar{y}}^j)$ is the negative pair where $\bar{y}$ is not aligned to x. The cosine similarity of $(u_x^i, u_{\bar{y}}^j)$ is minimized during training.

  We select J negative pairs for a source word x. In the implementation, we use Jbest outputs of the current model excluding the correct translation as the negative pairs. To keep balance between positive and negative pairs, the positive pair is copied J times to pair with negative pairs.

  During inference, we select $y = \arg\max_y cos(u_x, u_y)$ as the translation of x.

- In the unsupervised contrastive training, no bilingual dictionary is given. The contrastive loss is the same to that of the supervised contrastive training, except that the bilingual dictionary is not given. We initialize the bilingual dictionary using the output of the static word embedding based unsupervised method, and iteratively update it by using the trained model of last iteration to find new translations for given source words and compose a new dictionary, which is used to train the new model. Such process iterates until the dictionary does not change any more.

3.2 Similarity Interpolation

The similarity interpolation is for inference. As shown in Figure 1, both the unified word representation space and the mapped contextual representation space can output the cosine similarities between words. Given a source word x, we interpolate both similarities as below:

$$\mathcal{S} = cos(u_x, u_y) + \lambda cos(a_x^{'}, a_y^{'}) \qquad (7)$$

where λ is the weight, $a_{x/y}^{'}$ is the mapped contextual representation, which is pre-trained as introduced in the section of the background of the contextual representation based method. We aim to find y that has the maximal $\mathcal{S}$ as the translation of x.

In the supervised setting, λ is tuned on the validation set consisting of translation pairs. In the unsupervised setting, λ is tuned by an unsupervised procedure: when source-to-target model and target-to-source model have been trained, the word x in the validation set is aligned to y' based on equation (7), then y' is back aligned to x based on the inverse version of equation (7). We select λ that has the highest accuracy of this back alignment to x.

4 Experiments

We test our combination mechanism in supervised and unsupervised BLI tasks on English-Espanish (EN-ES), English-Arabic (EN-AR), English-Chinese (EN-ZH), English-German (EN-DE), and English-French (EN-FR).

4.1 Data

We need monolingual corpora to compute the contextual representations. Unfortunately, most existing BLI datasets distribute pre-trained word embeddings alone, but not the monolingual corpora used to train them. For that reason, we use WikiExtractor[2] to extract plain text from Wikipedia dumps, and preprocess the resulting corpora using standard Moses (Koehn et al., 2007) tools by applying sentence splitting, punctuation normalization, tokenization, and lowercasing. On these corpora, we use the cross-lingual pre-training system XLM (Lample and Conneau, 2019)[3] to compute the contextual representations.

Meanwhile, we also use these corpora to train the static word embeddings by using fastText[4] to ensure that both the contextual representations and the static word embeddings come from the same data. We use the bilingual dictionaries released by Muse project[5] in our experiments. Note that some words in these dictionaries do not necessarily appear in our monolingual corpora, we have to recompose the training, validation, and test sets such that all words in these sets are included in our monolingual corpora. In the end, we have 5000 entries with unique source words in the training set, and 1500 entries with unique source words in both the validation set and the test set for all language pairs.

4.2 Baseline Systems

Baseline systems are divided into two tasks as below. We run the released code of each baseline system in our experiments.

Supervised BLI task, which is allowed to use bilingual dictionaries for training and validation. The baseline systems are:

- **Muse**: Supervised Muse is set as the baseline in Conneau et al. (2017). It uses iterative Procrustes alignment for supervised BLI.
- **VecMap**[6]: Artetxe et al. (2018a) use a multi-step framework consisting of several steps: whitening, orthogonal mapping, re-weighting, de-whitening, and dimensionality reduction.
- **RCSLS**[7]: In addition to use CSLS during inference, Joulin et al. (2018) minimize a convex relaxation of CSLS loss during training, and improve the supervised BLI performance.

---

[2] https://github.com/attardi/wikiextractor

[3] https://github.com/facebookresearch/xlm. We use the MLM model of 15 languages with tokenize + lowercase + no accent + BPE.

[4] https://github.com/facebookresearch/fastText/

[5] https://github.com/facebookresearch/MUSE

[6] https://github.com/artetxem/vecmap

[7] `https://github.com/facebookresearch/fastText/tree/master/alignment`

	EN-ES		EN-AR		EN-ZH		EN-DE		EN-FR		avg
	→	←	→	←	→	←	→	←	→	←	
Supervised BLI											
Muse(Conneau et al., 2017)	77.80	81.40	49.47	55.07	40.87	43.07	69.80	71.07	78.67	79.33	64.65
VecMap(Artetxe et al., 2018a)	77.20	82.13	53.53	57.73	52.13	46.67	70.33	72.80	78.40	80.47	67.14
RCSLS(Joulin et al., 2018)	79.27	84.30	55.53	61.00	53.07	48.87	73.07	75.40	79.60	82.07	69.22
BLISS(Patra et al., 2019)	79.67	84.87	54.47	59.60	50.80	48.80	73.33	76.13	79.20	82.93	68.98
Unified$_{\mathrm{VecMap}}$	79.60	84.73	56.20	60.73	53.80	48.93	73.27	74.47	79.33	81.33	69.24
Contextual$_{\mathrm{VecMap}}$	44.07	50.33	5.07	7.73	21.93	9.67	44.60	47.87	57.47	65.20	35.39
Interpolation$_{\mathrm{VecMap}}$	80.47	85.70	57.13	61.47	56.27	50.60	74.13	77.13	80.80	83.40	70.71
Unified$_{\mathrm{RCSLS}}$	80.13	86.60	55.87	62.47	56.67	51.13	74.26	77.93	81.20	83.87	71.01
Contextual$_{\mathrm{RCSLS}}$	46.27	51.40	3.67	7.13	19.47	7.93	45.67	47.80	58.00	65.20	35.25
Interpolation$_{\mathrm{RCSLS}}$	**80.67**	**87.67**	**59.40**	**62.73**	**59.40**	**52.27**	**74.87**	**79.67**	**81.80**	**85.40**	**72.39**
Unsupervised BLI											
Muse(Conneau et al., 2017)	77.06	81.53	48.00	55.47	24.26	43.00	70.13	71.20	78.73	78.40	62.77
VecMap(Artetxe et al., 2018b)	77.60	81.67	50.87	56.73	34.33	44.00	70.00	71.80	78.73	80.27	64.60
Ad.(Mohiuddin and Joty, 2019)	77.93	82.20	50.07	57.33	34.67	43.67	69.13	72.47	78.46	80.13	64.61
Unified	79.47	82.60	52.47	57.87	35.93	46.07	71.07	74.00	80.27	80.87	66.06
Contextual	46.33	55.93	3.87	7.53	17.40	4.87	43.53	44.33	56.20	63.93	34.39
Interpolation	**79.93**	**85.33**	**52.73**	**58.47**	**37.07**	**46.27**	**72.53**	**78.73**	**81.80**	**84.13**	**67.70**

Table 1: P@1 on all language pairs. "Unified" denotes our unified word representation based method, which computes $cos(u_x, u_y)$, "Contextual" denotes the contextual representation based method, which computes $cos(a_x^{'}, a_y^{'})$, "Interpolation" denotes our similarity interpolation, which computes $cos(u_x, u_y) + \lambda cos(a_x^{'}, a_y^{'})$. The subscript "VecMap" denotes that our method is based on the work of Artetxe et al. (2018a), the subscript "RCSLS" denotes that our method is based on the RCSLS criterion in training (Joulin et al., 2018). In unsupervised BLI, there is no subscript in our method, which means using the default "VecMap" (Artetxe et al., 2018b).

- **BLISS**[8]: Patra et al. (2019) use a semi-supervised method that leverages both the bilingual dictionary and a larger set of unaligned word embeddings.

Unsupervised BLI task, which is not allowed to use any parallel resources for training and validation. The baseline systems are:

- **Muse**: Unsupervised Muse (Conneau et al., 2017) uses adversarial training and iterative Procrustes refinement.
- **VecMap**: Artetxe et al. (2018b) use careful initialization, robust self-learning procedure, and symmetric re-weighting to improve the unsupervised mapping result.
- **Ad.**[9]: Mohiuddin and Joty (2019) include regularization terms for adversarial auto-encoder for the unsupervised BLI.

4.3 Experimental Settings

We use fastText to train the word embeddings for BLI. The dimension of the word embeddings is 300. The contextual representations are extracted from XLM, and the dimension of the contextual representations is 1024. For each word type, we randomly select ten sentences containing the word from the monolingual corpora to do the averaging to get the contextual representation. The influence of the number of selected sentences for each word type is reported in section 4.5.2. Regarding the spring network, we use ten negative pairs for each source word in the supervised contrastive training, and use one negative pair for each source word in the unsupervised contrastive training.

All inferences in our experiments, including all baseline systems, use CSLS which is introduced in Conneau et al. (2017). The results are evaluated by Precision@1 (P@1).

4.4 Main Results

Table 1 summarizes the main results of the supervised and the unsupervised BLI tasks on all test sets. In both tasks, our proposed methods achieve significant improvements, with average 3.2 points higher than the strongest baseline RCSLS in the supervised task, and with average 3.1 points higher than the strong baselines VecMap and Ad. in the unsupervised task.

In the supervised task, we have two independent bases to build our proposed methods. One is VecMap (Artetxe et al., 2018a), the other is RCSLS (Joulin et al., 2018). They are the preprocessing steps to align the static word embeddings, and align the contextual representations in the two languages. Our methods build upon these alignments, and further train the spring networks and the unified word representations for the combination. The performances of our methods with these two bases are

---

[8] https://github.com/joelmoniz/BLISS

[9] https://github.com/taasnim/unsup-word-translation/

	EN-ES		EN-AR		EN-ZH		EN-DE		EN-FR		avg
	→	←	→	←	→	←	→	←	→	←	
Supervised BLI											
XLM-Unified$_{\text{RCSLS}}$	80.13	86.60	55.87	62.47	56.67	51.13	74.26	77.93	81.20	83.87	71.01
XLM-Contextual$_{\text{RCSLS}}$	46.27	51.40	3.67	7.13	19.47	7.93	45.67	47.80	58.00	65.20	32.05
XLM-Interpolated$_{\text{RCSLS}}$	80.67	**87.67**	**59.40**	62.73	**59.40**	**52.27**	74.87	79.67	81.80	85.40	**72.39**
mBART-Unified$_{\text{RCSLS}}$	80.07	85.33	56.13	61.93	55.53	51.20	74.27	77.53	80.20	83.40	70.56
mBART-Contextual$_{\text{RCSLS}}$	46.33	49.70	3.53	8.24	18.80	8.03	45.33	48.33	60.10	66.53	32.26
mBART-Interpolated$_{\text{RCSLS}}$	**81.27**	87.40	57.00	**63.00**	58.27	52.07	**76.07**	**80.60**	**82.33**	**85.93**	**72.39**
Unsupervised BLI											
XLM-Unified	79.47	82.60	52.47	57.87	35.93	46.07	71.07	74.00	80.27	80.87	66.06
XLM-Contextual	46.33	55.93	3.87	7.53	17.40	4.87	43.53	44.33	56.20	63.93	31.26
XLM-Interpolated	**79.93**	**85.33**	**52.73**	58.47	**37.07**	**46.27**	**72.53**	**78.73**	**81.80**	84.13	**67.70**
mBART-Unified	79.07	82.60	51.60	58.13	35.80	44.67	70.47	74.00	79.93	81.27	65.75
mBART-Contextual	47.06	53.37	5.40	8.13	16.76	4.77	41.23	42.53	53.53	62.87	30.51
mBART-Interpolated	**79.93**	84.90	**52.73**	**59.67**	36.60	45.27	71.87	76.33	81.40	**84.53**	67.32

Table 2: Result comparison between using XLM and using mBART for the contextual representations.

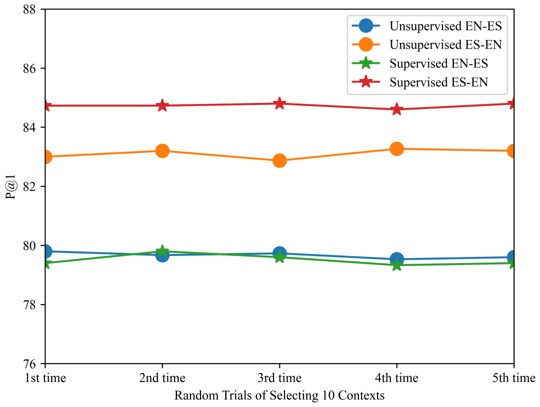

Figure 2: Performances of 5 random trials of selecting 10 sentences to gather contexts for each word type.

reported in Table 1 with the corresponding subscripts.

Table 1 shows that if we use VecMap as the basis of our method, we can improve 3.6 points over the corresponding VecMap baseline. If we use RCSLS as the basis, we can improve 3.2 points over the corresponding RCSLS baseline. In our methods, "Unified" can achieve around 2 points improvement over the corresponding baselines. Although "Contextual" obtains inferior performances, it is complementary to "Unified". When "Contextual" is combined with "Unified" through the interpolation, the performance is further improved, achieving the best performance among all systems. It shows that our combination mechanism is effective to utilize the merits of both the static word embeddings and the contextual representations.

In the unsupervised task, we achieve the significant improvements over the baselines. "Unified" is 1.5 points better than VecMap baseline. "Contextual" is inferior to other methods, but it can provide useful complements to "Unified", resulting in the final 3.1 points improvement through interpolation.

In summary, our combination mechanism consistently improves the performances for both distant language pairs, such as EN-AR and EN-ZH, and closely-related European language pairs.

4.5 Analyses

4.5.1 XLM v.s. mBART

Our results in Table 1 are based on using XLM for obtaining the contextual representations. In this section, we also use mBART (Liu et al., 2020) to compare with XLM. Table 2 shows the comparison result. XLM pre-trains the Transformer encoder through the masking mechanism, while mBART pre-trains the full Transformer encoder-decoder through multilingual denoising. Regarding BLI task, we obtain the contextual representations from the encoder. Table 2 shows that XLM and mBART get similar BLI performances since only encoder is used. In some directions, mBART performs slightly better than XLM, while in other directions, XLM is slightly better. According to the average performance, XLM ties with mBART in the supervised task, and is slightly better in the unsupervised task.

4.5.2 The Randomness of Contexts

The contextual representations are derived randomly from sentences of the monolingual corpora. We study if this random derivation affects the performances. Firstly, we run 5 times of randomly selecting 10 sentences to gather contexts for each word type in the "Unified" setting. Figure 2 shows

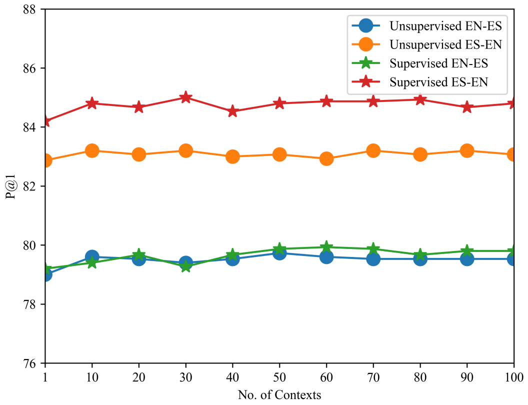

Figure 3: Performances of randomly selecting 1-100 sentences to gather contexts for each word type.

	EN-DE		EN-FR	
	→	←	→	←
Supervised BLI				
RCSLS	61.82	60.92	59.58	60.92
Unified	62.93	62.35	61.87	62.40
Interpolation	**67.09**	**68.00**	**67.05**	**67.12**
Unsupervised BLI				
VecMap	55.29	56.96	57.18	59.33
Unified	57.20	58.70	58.40	61.16
Interpolation	**61.00**	**61.52**	**62.70**	**63.40**

Table 3: P@1 of using WaCKy corpora.

that the performance is stable in the 5 trials. Secondly, we try randomly selecting 1-100 sentences to gather contexts for each word type. Figure 3 shows that selecting 1 sentence will drag the performance down to baseline, which indicates that 1 sentence is too random to gather enough information for BLI.

We only list the studies on EN-ES due to space limit. Studies on other language pairs can be found in the appendix.

4.5.3 The Influence of Selecting Encoder Layer

In the above experiments, we derive the contextual representations from the first layer of the encoder of XML/mBART. In this section, we show how different will be when we change the layer in the "Unified" setting. Figure 4 shows that as layer go higher, the performance drops. Please refer to the appendix for performances of other language pairs.

4.5.4 Results of using WaCKy Corpora

WaCKy corpora is introduced in Dinu et al. (2014) for BLI, but only word embeddings trained on WaCKy corpora are provided in their work. To obtain the contextual representations, we find WaCKy corpora from BUCC[10], and use the corresponding dictionaries with the same training, validation, and test split. We use mBART instead of XLM for computing the contextual representations in this task.

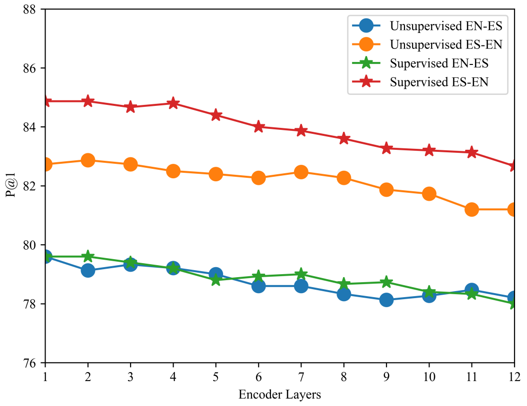

Figure 4: Performances of selecting different layer of XLM encoder to derive the contextual representations.

Table 3 shows that our combination mechanism is robust on this dataset. Both "Unified" and "Interpolation" perform better than the baselines. "Interpolation" achieves significant improvements in the supervised setting.

5 Discussion

The static word embeddings in our paper are trained by skip-gram or CBOW, while the word embeddings from XLM/mBART are trained by the pretraining objectives. Different training objectives result in quite different word embeddings. Actually, they show remarkably different behavior for BLI. By using VecMap, the word embeddings from XLM/mBART perform averagely around 30 points lower than the static word embeddings used in our paper. We also test fastText with 1024 dimension and word2vec with 300 dimension for fair comparison. They all perform remarkably better than the word embeddings from XLM/mBART. We plug in the word embeddings from XLM/mBART in place of the fastText static embeddings in our combination approach, and obtain much worse performance. This indicates that the static word embeddings trained by skip-gram or CBOW are more suitable for BLI and our combination approach.

---

[10] https://comparable.limsi.fr/bucc2020/bucc2020-task.html

In addition, regarding the asymmetry in Figure 1, we used to try the spring function that takes the static word embeddings as input, but got much worse results. This indicates that the spring function conditioned on the static space may be not helpful for BLI. Such observation may also explain that symmetrizing Figure 1 by yielding two unified spaces to combine performs slightly worse than the asymmetry version of Figure 1 of this paper. This is because that the spring function conditioned on the static space is introduced to maintain the symmetry, while this introduced spring function is not helpful for the combination.

6 Conclusion

Most BLI systems use either the static word embeddings or the contextual representations, but there is no works to combine both. In this paper, we propose a combination mechanism, which consists of the unified word representations and the similarity interpolation. The unified word representations use a spring network to pull the static word embeddings with offsets produced by the contextual representations, and compose a unified space such that parallel words are nearest neighbors to each other. The similarity interpolation is applied afterward to interpolate the similarities in the unified space and the contextual representation space. BLI experiments on multiple language pairs show that our combination mechanism can utilize the merits of both the static word embeddings and the contextual representations, achieving significant improvements over robust baseline systems in both the supervised and the unsupervised BLI tasks.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. This work was supported by National Natural Science Foundation of China (Grant No. 62036004, 61673289)

References

Hanan Aldarmaki and Mona Diab. 2019. Context-aware cross-lingual mapping. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3906–3911.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2019. Bilingual lexicon induction through unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5002–5007.

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Multilingual alignment of contextual word representations. In *Proceedings of International Conference on Learning Representations*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel, comparable texts. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 414–420.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics 2008: HLT*, pages 771–779.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the eighth workshop on statistical machine translation*, pages 262–270.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Saurabh Kulshreshtha, José Luis Redondo-García, and Ching-Yun Chang. 2020. Cross-lingual alignment methods for multilingual bert: A comparative study. *arXiv preprint arXiv:2009.14304*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 7059–7069.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tasnim Mohiuddin, M Saiful Bari, and Shafiq Joty. 2020. Lnmap departures from isomorphic assumption in bilingual lexicon induction through non-linear mapping in latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2712–2723.

Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting Adversarial Autoencoder for Unsupervised Word Translation with Cycle Consistency and Improved Training. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3857–3867.

Barun Patra, Joel Ruben Antony Moniz, Sarthak Garg, Matthew R Gormley, and Graham Neubig. 2019. Bilingual lexicon induction with semi-supervision in non-isometric embedding spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 184–193.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? *arXiv preprint arXiv:1804.06323*.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 519–526.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1599–1613.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Haoyue Shi, Luke Zettlemoyer, and Ikbal Sida. 2021. Bilingual lexicon induction via unsupervised bitext construction and word alignment. *arXiv preprint arXiv:2101.00148*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788.

Ivan Vulić and Marie Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1613–1624.

Zirui Wang, Jiateng Xie, Ruochen Xu, Yiming Yang, Graham Neubig, and Jaime Carbonell. 2020. Cross-lingual alignment vs joint training: A comparative study and a simple unified framework. In *Proceedings of International Conference on Learning Representations*.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1307–1317.

Appendix

A Experiment Environment and Settings

Our experiment is running on a Linux machine with GTX 1080Ti. The version of cuDNN is 7.6.0 and the version of CUDA is 10.1. We also use the PyTorch deep learning framework. The version of PyTorch is 1.6. The average runtime in our experiment is 5 to 10 minutes for one language pair, excluding the time for training word embeddings by fastText.

Regarding Wikipedia corpora used in our experiments, their download links are: `https://dumps.wikimedia.org/${lg}wiki/latest/${lg}wiki-latest-pages-articles.xml.bz2`, where ${lg} should be replaced with the corresponding languages. ${lg} can be set as "en", "es", "ar", "zh", "de", and "fr". We download them on Jan. 2, 2021. Regarding WaCKy corpora, we use WaCKy corpora[11] provided by BUCC2020 for obtaining the contextual representations, and use the provided word embeddings trained on this corpora. The BUCC2020 dictionaries are downloaded from the same page. We concat the three training sets with high, mid, low frequency as one training set, and concat the three test sets with high, mid, low frequency as one test set. Because no translations are provided in the test set of BUCC2020, we look up the Muse dictionaries to get the translations. All source words in the test set can find translations in the Muse dictionaries.

The evaluation script for computing precision@1 (P@1) is: `https://github.com/artetxem/vecmap.git/eval_translation.py`.

B Parameter Settings

The parameter size of the spring network is 796202. We use default parameter settings of VecMap and RCSLS.

language	λ
EN→ES	0.11
ES→EN	0.05
EN→AR	0.10
AR→EN	0.30
EN→ZH	0.12
ZH→EN	0.10
EN→DE	0.11
DE→EN	0.25
EN→FR	0.13
FR→EN	0.11

Table 4: The optimal hyperparameter λ in the unsupervised settings

Regarding the hyperparameter λ in the similarity interpolation, we search the optimal value in [0.05-0.3] with step size of 0.01. We found $\lambda = 0.1$ is superior on validation sets in all supervised settings. The optimal value of λ found by the unsupervised tuning procedure in the unsupervised settings, which is introduced in section 3.2, is shown in Table 4. In the search range, the performances has low variance, and are better than the baselines.

C The Influence of Selecting Encoder Layer in Other Language Pairs

We report the influences on EN-AR, EN-ZH, EN-DE, EN-FR in Figure 5. It shows that as we select higher layers for deriving the contextual representations, the performances become lower. This trend exists in most language pairs, except that the trend in EN-ZH is not significant.

---

[11] `https://corpus.leeds.ac.uk/serge/bucc`

D The Randomness of Contexts in Other Language Pairs

We report the randomness analyses on EN-AR, EN-ZH, EN-DE, EN-FR in Figure 6. It shows that trying 5 times of selecting 10 random sentences for gathering contexts gets stable performances in all language pairs. In most cases, using 1 sentence for computing the contextual representation drags the performance down, which indicates the inadequacy of 1 sentence for gathering contexts.

E Performances on Words with Different Frequencies

We use WaCKy corpora and the dictionaries provided by BUCC2020[12] to study the performances on words with different frequencies. The provided dictionaries are divided into groups of high frequency words, mid frequency words, and low frequency words. We test our combination mechanism on these three groups respectively. The results are presented in Table 5 and Table 6. We can see that our combination mechanism is effective on all three groups.

	EN→DE	DE→EN	EN→FR	FR→EN
	High Frequency			
VecMap	67.60	75.27	79.00	79.27
Unified	71.27	77.40	79.67	80.80
Interpolation	72.53	79.67	80.47	82.13
	Middle Frequency			
VecMap	67.20	71.47	77.73	80.27
Unified	67.80	73.53	78.33	81.00
Interpolation	70.60	78.13	81.13	83.73
	Low Frequency			
VecMap	69.53	74.73	78.47	81.87
Unified	70.00	75.53	79.27	82.67
Interpolation	77.53	82.47	87.00	87.07

Table 5: Performances on words with different frequencies in the unsupervised setting

	EN→DE	DE→EN	EN→FR	FR→EN
	High Frequency			
VecMap	67.87	76.13	78.27	79.73
Unified	70.40	78.07	78.4	81.4
Interpolation	71.00	78.90	78.90	82.13
	Middle Frequency			
VecMap	68.87	71.8	78.47	80.87
Unified	69.93	75.20	79.53	82.20
Interpolation	71.20	76.33	80.07	83.73
	Low Frequency			
VecMap	70.60	76.13	79.67	82.93
Unified	72.67	78.27	81.20	84.20
Interpolation	74.13	79.97	82.30	84.73

Table 6: Performances on words with different frequencies in the supervised setting

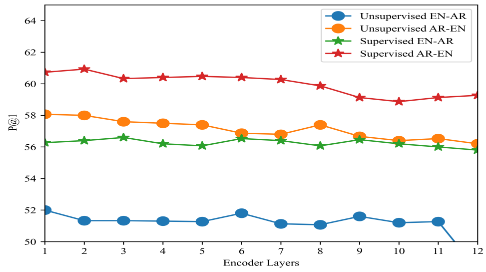

(a) EN-AR

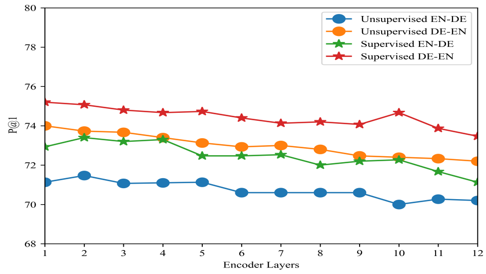

(b) EN-DE

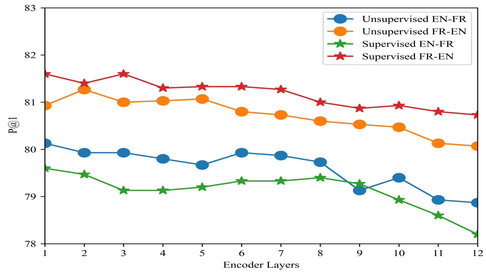

(c) EN-FR

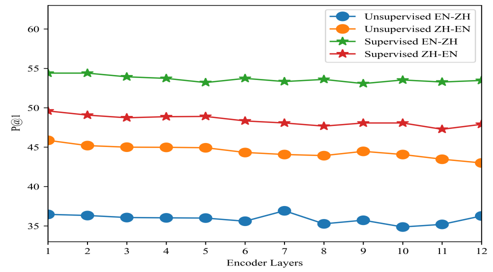

(d) EN-ZH

Figure 5: Performances of selecting different layer of XLM encoder to derive the contextual representations.

---

[12] https://comparable.limsi.fr/bucc2020/bucc2020-task.html

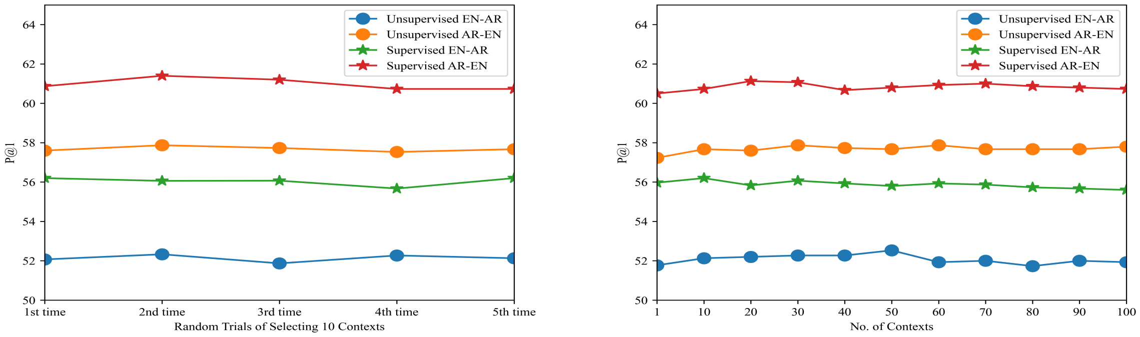

(a) EN-AR

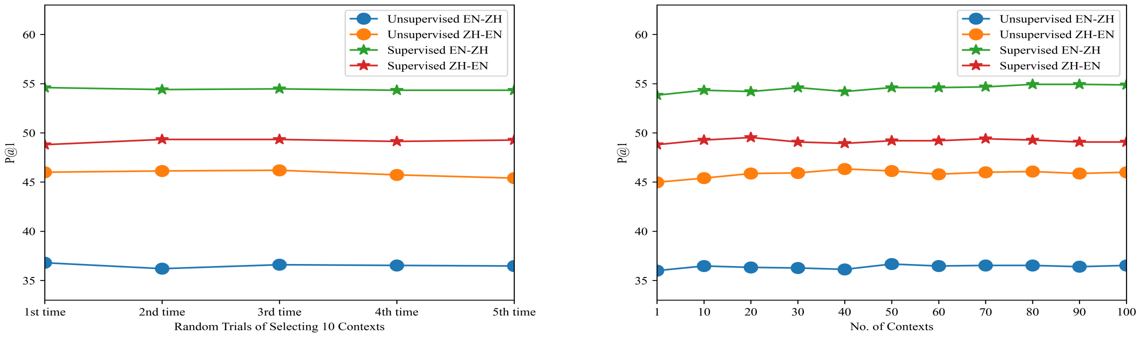

(b) EN-ZH

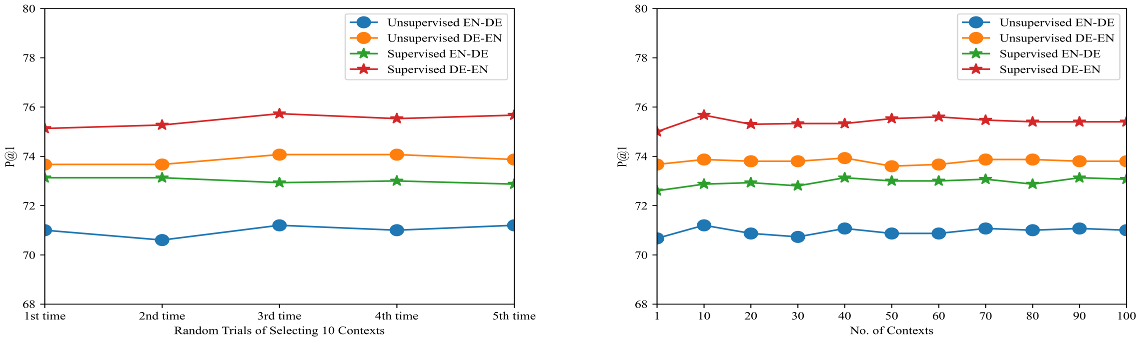

(c) EN-DE

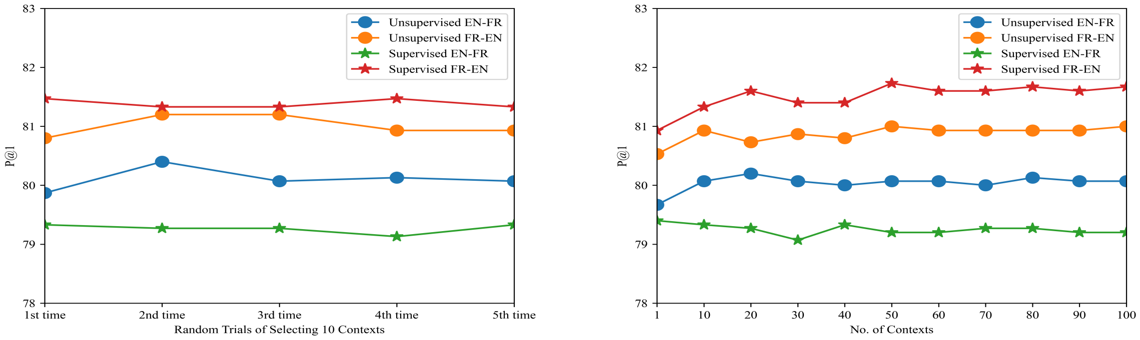

(d) EN-FR

Figure 6: Performances of different context selection methods on EN-AR, EN-ZH, EN-DE, and EN-FR.

Exploring Unsupervised Pretraining Objectives for Machine Translation

Christos Baziotis, Ivan Titov, Alexandra Birch and **Barry Haddow**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
`c.baziotis@ed.ac.uk`, `ititov@inf.ed.ac.uk`,
`a.birch@ed.ac.uk`, `bhaddow@inf.ed.ac.uk`

Abstract

Unsupervised cross-lingual pretraining has achieved strong results in neural machine translation (NMT), by drastically reducing the need for large parallel data. Most approaches adapt masked-language modeling (MLM) to sequence-to-sequence architectures, by masking parts of the input and reconstructing them in the decoder. In this work, we systematically compare masking with alternative objectives that produce inputs resembling real (full) sentences, by reordering and replacing words based on their context. We pretrain models with different methods on English↔German, English↔Nepali and English↔Sinhala monolingual data, and evaluate them on NMT. In (semi-) supervised NMT, varying the pretraining objective leads to surprisingly small differences in the finetuned performance, whereas unsupervised NMT is much more sensitive to it. To understand these results, we thoroughly study the pretrained models and verify that they encode and use information in different ways. We conclude that finetuning on *parallel* data is mostly sensitive to few properties that are shared by most models, such as a strong decoder, in contrast to unsupervised NMT that also requires models with strong cross-lingual abilities.

1 Introduction

Neural machine translation (NMT) is notoriously data-hungry (Koehn and Knowles, 2017). To learn a strong model it requires large, high-quality and in-domain parallel data, which exist only for a few language-pairs. The most successful approach for improving low-resource NMT is backtranslation (Sennrich et al., 2016), that exploits abundant monolingual corpora to augment the parallel with synthetic data. However, in low-resource settings, it may fail to improve or even degrade translation quality if the initial model is not strong enough (Imankulova et al., 2017; Burlot and Yvon, 2018).

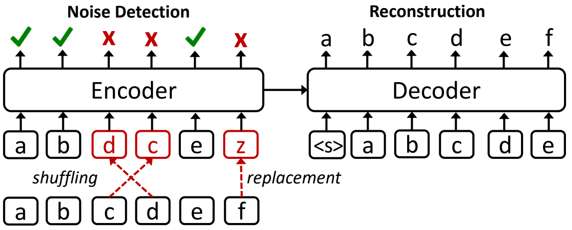

Figure 1: We consider noising methods that produce inputs which resemble real sentences, unlike masking.

Unsupervised pretraining is a complementary technique, that has revolutionized many natural language understanding (NLU) tasks (Wang et al., 2019). The dominant approach is to train a (large) model on a lot of unlabeled data using the masked language modeling (MLM; Devlin et al. (2019)) objective and then finetune it on a downstream task. Besides improving generalization, good initialization drastically reduces the need for labelled data. This paradigm has been applied recently to NMT yielding impressive results in low-resource settings, with models such as XLM (Conneau and Lample, 2019), MASS (Song et al., 2019) and BART/mBART (Lewis et al., 2020b; Liu et al., 2020), that adapt MLM to sequence-to-sequence architectures. Although pretraining alone is not enough to outperform backtranslation, it helps the initial model to produce synthetic data of sufficient quality, and combining them yields further improvements.

Most prior work in pretraining has focused on optimizing the masking strategy (Rogers et al., 2021). Similarly, MASS and mBART consider slightly different masking strategies. However, due to differences in their experimental setup (i.e., capacity or training data) and lack of analysis that goes beyond evaluation on downstream tasks, it is unclear if there is a meaningful difference between them, as far as NMT is concerned. They also suffer from a pretraining-finetuning discrepancy (Yang et al., 2019), in which a model is pretrained on masked inputs, but finetuned on full sentences.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2956–2971
August 1–6, 2021.

In this work[1], we explore different objectives to masking for unsupervised cross-lingual pretraining. We inject noise that creates examples (Fig. 1), similar to those encountered in finetuning, unlike masking. This includes, randomly replacing input words based on their context using a cross-lingual generator, inspired by Clark et al. (2020), and locally reordering input words, which prevents the cross-attention from naively (monotonically) attending over the source. We also explore auxiliary losses over the encoder to improve its representations.

First, we pretrain models with different configurations, on English-German, English-Nepali and English-Sinhala monolingual data. Then, we *systematically* compare them on the downstream tasks of supervised, semi-supervised and unsupervised NMT. In (semi-) supervised NMT, we observe that models yield surprisingly similar results, although some methods are better than others. We find that even pretraining with shuffled inputs leads to significant improvements over random initialization, similar to the concurrent work of Sinha et al. (2021) on pretrained encoders for NLU. Unsupervised NMT, however, reveals large (up to 9 BLEU points) differences, and against our expectations, masking achieves the best performance. To understand these results, unlike prior work, we thoroughly analyze the pretrained models using a series of probes, and discover that each objective drives the models to encode and use information in unique ways.

Based on our findings, we conclude that each finetuning process is sensitive to specific properties of pretrained models, similar to Artetxe et al. (2020). We hypothesize that (semi-) supervised NMT is mostly sensitive to the LM abilities of pretrained models, as the source→target mappings can be learnt from the parallel data. Unsupervised NMT requires models to also rely on their own word-translation abilities. Our contributions are:

1. We *systematically* compare many pretraining methods, including alternatives to masking, in three NMT tasks and for three language-pairs.
2. We discover that (semi-) supervised NMT is not sensitive to the pretraining strategies. Our ablation (§4.4) suggests that a strong decoder is the most important factor, while differences in the encoder (§4.5) don't affect the results.
3. Unsupervised setting is much more sensitive to the pretraining objective, and masking methods are the most effective. We hypothesise that learning to copy is important here (§5.2) as is cross-lingual encoding (§4.5).
4. We analyze the pretrained models with a series of probes (§5.1, §5.2, §5.3), and show noticeable differences in how they encode and use information, offering valuable insights.

2 Related Work

Pretraining for NMT Ramachandran et al. (2017) first explored unsupervised pretraining for NMT using LMs trained on monolingual data of the source and target languages to initialize the encoder and decoder of an RNN-based TM (Bahdanau et al., 2015). Conneau and Lample (2019) adopt the same approach, by extending BERT/MLM (Devlin et al., 2019) to the cross-lingual setting (XLM). They randomly mask tokens from input sentences in many languages, and the model is trained to predict them. However, the same pretrained XLM is used to (separately) initialize both the encoder and decoder of a downstream translation model (TM), which neglects the interaction between them. MASS (Song et al., 2019) addresses this limitation, by extending MLM to sequence-to-sequence pretraining, which includes the cross-attention mechanism, and achieved further improvements in low-resource and unsupervised NMT. Liu et al. (2020), concurrently demonstrated comparable results with a similar approach (mBART), but on a larger scale. Both mBART and MASS, consider different strategies for reconstructing masked input spans.

Objectives Yang et al. (2019) point out that BERT (Devlin et al., 2019) is pretrained with masked inputs, but then finetuned on full sentences, which creates a discrepancy. To address this, they change the self-attention in Transformers (Vaswani et al., 2017) to predict tokens conditioned on all permutations of other tokens in a sentence and Song et al. (2020) extend this to sequence-level pretraining for NLU. MARGE (Lewis et al., 2020a) explores multi-lingual pretraining for document-level NMT, by reconstructing texts from a set of retrieved relevant documents. Clark et al. (2020) propose the replaced token detection (RTD) objective for pretraining text encoders. They replace tokens with samples from a MLM and train the encoder as a discriminator to predict whether each word is real or fake. Similar ideas have been previously explored in NMT with contextual data augmentation (Fadaee et al., 2017; Kobayashi, 2018; Gao et al., 2019).

---

[1] Code at github.com/cbaziotis/nmt-pretraining-objectives

3 Pretraining

Our pretraining model is a multilingual denoising sequence autoencoder, based on the Transformer (Vaswani et al., 2017).

We assume access to a corpus of unpaired data, containing text in two languages A, B. Given a text sequence of N tokens $\boldsymbol{x} = \langle x_1, x_2, ..., x_N \rangle$ we first add noise to it and obtain its corrupted version $\boldsymbol{x'}$. An encoder transforms $\boldsymbol{x'}$ into a sequence of contextualized representations $h(\boldsymbol{x'}) = \langle h_1, h_2, ..., h_N \rangle$, which are given as input to the decoder, that produces a reconstruction of $\boldsymbol{x}$. The reconstruction loss is the negative log-likelihood (NLL) of $\boldsymbol{x}$:

$$\mathcal{L}_{\mathrm{R}} = \frac{1}{N} \sum_{t=1}^{N} - \log p(\boldsymbol{x}_t | \boldsymbol{x}_{<t}, h(\boldsymbol{x'})) \quad (1)$$

Each batch contains sentences in either A, or B and to distinguish between them, we add language id tokens at the end of the source sentences, and the beginning of the target sentences.

3.1 Pretraining Methods

In this section, we describe the methods that we use to inject noise into the model. We also explore auxiliary losses over the encoder, aiming to improve the input representations.

Masking Similar to prior work, we replace a random subsequence M of the input tokens with a special `[MASK]` token and train the model to reconstruct the original input. We consider masking words as well as spans following mBART.

Masking + eMLM When using masking noise we also explore the addition of an auxiliary MLM loss over the *encoder* to which we will refer as eMLM. This explicitly trains the encoder to reconstruct the representations of masked tokens:

$$\mathcal{L}_{\mathrm{eMLM}} = \frac{1}{|M|} \sum_{t \in M} - \log p(\boldsymbol{x}_t | \boldsymbol{x}_{\notin M})) \quad (2)$$

Replacing We inject word replacement noise, by extending Clark et al. (2020) to the cross-lingual setting. Specifically, we jointly train a *separate* cross-lingual (mBERT-like) MLM generator. First, we mask a random subset M of the input tokens[2] and the generator is trained to predict them:

$$\mathcal{L}_{\mathrm{G}} = \frac{1}{|M|} \sum_{t \in M} - \log p(\boldsymbol{x}_t | \boldsymbol{x}_{\notin M})) \quad (3)$$

For each masked token $\boldsymbol{x}_t$, the generator produces a distribution $p_{\mathrm{G}}(\boldsymbol{x}_t | \boldsymbol{x}_{\neq t})$. We replace the masked tokens with samples from p_{G} to obtain $\boldsymbol{x'}$, and feed the updated (corrupted) input to the encoder. We consider two configurations for the generator:

- *Untied*: The default setting, we use a small generator, which is half the size (x0.5 parameters) of the encoder, following Clark et al. (2020).
- *Tied*: We tie the weights of the encoder and the generator. This setting implicitly adds an auxiliary MLM loss over the encoder, and can be thought as a counterpart of "replace+eMLM".

Replacement + RTD Motivated by the results of Clark et al. (2020) in monolingual NLU, we add a replacement token detection (RTD) head over the encoder that gives direct supervision to the model regarding the location of noise. The RTD head $D(\cdot)$ is a token-level discriminator over the encoder outputs $h(\boldsymbol{x'})$, that predicts if an input token $\boldsymbol{x'_t}$ is original or replaced. We parameterize D with a non-linear projection followed by a sigmoid function: $D(\boldsymbol{x'_t}) = \mathrm{sigmoid}(\boldsymbol{u}^\top \mathrm{RELU}(\boldsymbol{W_D}\, h(\boldsymbol{x'_t})))$. The RTD loss is defined as the average token-level binary cross-entropy:

$$\begin{aligned}\mathcal{L}_{\mathrm{RTD}} = \frac{1}{N} \sum_{t=1}^{N} & - (\boldsymbol{x}'_t = \text{original}) \log D(\boldsymbol{x}'_t) \quad (4) \\ & - (\boldsymbol{x}'_t \neq \text{original})(1 - D(\boldsymbol{x}'_t))\end{aligned}$$

Shuffling Although pretrained models, such as MASS or mBART, do pretrain the cross-attention mechanism, the input words remain in their original positions and this biases the models into learning only naive monotonic alignments. To *actively* pretrain the cross-attention, we locally shuffle a random subset of *whole-words* in the input, using the method of (Lample et al., 2018). The length of reordering is bounded by k (positions). Small k introduce local shuffling, while large k allow words to be moved farther from their original position, making the input more like a bag-of-words (BoW).

3.2 Optimization

During pretraining, we minimize a weighted sum of the reconstruction L_{R}, and depending on the method, some of the auxiliary L_{eMLM}, L_{G}, L_{RTD} losses. We assign equal weight ($\lambda = 1$) to all losses, except for L_{RTD} for which we set its weight $\lambda = 25$ following Clark et al. (2020), to account for the fact that is in a different scale.

---

[2] We use whole-word masking, that masks all the (subword) tokens of a word, instead of independent token masking.

Method	en→de		de→en		en→ne	ne→en	en→si	si→en
	wmt18	wmt19	wmt18	wmt19				
random	26.2±0.1	25.3±0.1	27.6±0.1	19.1±0.3	3.3±0.1	6.5±0.1	2.5±0.1	6.5±0.1
mask=35%	33.3±0.1	30.7±0.2	33.2±0.0	**25.4**±0.0	5.1±0.1	10.2±0.1	3.7±0.0	10.0±0.1
mask=35% +eMLM	33.4±0.0	30.6±0.1	33.5±0.1	25.2±0.2	**5.3**±0.0	**10.8**±0.1	**4.0**±0.0	10.4±0.1
mask=35% (span)	33.3±0.1	30.5±0.1	33.4±0.0	25.2±0.0	5.1±0.1	10.1±0.1	3.9±0.1	9.9±0.1
shuffle=5	31.6±0.1	28.7±0.0	31.7±0.0	23.9±0.1	4.9±0.0	9.9±0.1	3.4±0.0	10.1±0.1
replace=35%	33.9±0.0	30.3±0.2	33.5±0.1	**25.4**±0.1	5.1±0.1	9.9±0.1	3.7±0.0	9.8±0.0
replace=35% +RTD	32.9±0.1	30.0±0.0	32.5±0.0	24.4±0.1	5.0±0.0	9.9±0.1	3.4±0.1	9.7±0.2
replace=35% +tied	**34.2**±0.0	**30.8**±0.1	**33.7**±0.1	25.3±0.2	**5.3**±0.0	10.6±0.1	3.7±0.0	**10.5**±0.1
+ shuffle=3	34.0±0.0	31.1±0.1	33.4±0.1	25.1±0.2	5.5±0.0	11.0±0.0	4.0±0.0	10.8±0.1

Table 1: Supervised NMT results. We report the average of 3 runs and the standard error of the mean (SEM).

4 Experiments

Datasets We focus on low-resource translation and consider three diverse language-pairs: English-German, English-Nepali and English-Sinhala. For English-German, we use the low-resource WMT News Commentary v13 (Bojar et al., 2018) [3] parallel dataset, which contains approximately 275K sentences. For pretraining, we use as monolingual data the WMT News Crawl articles (Bojar et al., 2018) from the year 2007 to 2017, which comprise 190M and 270M sentences for English and German, respectively. For English-Nepali and English-Sinhala, we use the same data as in Guzmán et al. (2019). The (pretraining) monolingual data contain 5M sentences from Common Crawl and Wikipedia per language, while the parallel data are approximately 600K sentences from the Bible, Open Subtitles, GNOME/KDE/Ubuntu, and Paracrawl.

Pre-processing For Nepali and Sinhala, we use the preprocessing scripts[4] provided by Guzmán et al. (2019), whereas for English and German, we use directly the raw data without any preprocessing. We use sentencepiece (SPM; Kudo and Richardson (2018)) with the "unigram" model, to train a subword-unit tokenization model on the concatenation of the monolingual data of each language-pair. We learn a joint vocabulary of 60K symbols for the English-German models, and 20K symbols for the English-Nepali and English-Sinhala models.

Evaluation For English-German, we use the WMT *newstest2017* as dev-set and the *newstest2018* and *newstest2019* as test-sets. For English-Nepali and English-Sinhala, we use the evaluation datasets provided by Guzmán et al. (2019), which are drawn from Wikipedia articles. We evaluate models using BLEU (Papineni et al., 2002) computed with SacreBLEU (Post, 2018). We report detokenized BLEU when translating into German and English, and tokenized BLEU when translating into Nepali and Sinhala, following Guzmán et al. (2019). At test time, we decode with beam search, using beams of size 5.

Model and Training Our models are based on the Transformer architecture (Vaswani et al., 2017). We use the Transformer-base configuration to reduce the computational cost and be able to explore more methods. We describe in detail the model architecture and hyperparameters, as well as the pretraining and finetuning processes, in Appendix §A. Our code is based on the official mBART implementation in Fairseq (Ott et al., 2019).

4.1 Supervised Translation

In our first experiment (Table 1), we evaluate each pretrained model on supervised NMT by finetuning it on the parallel data. As a baseline we use a randomly initialized TM with identical configuration and vocabulary to that of the pretrained models, denoted as "random". We also pretrain a model equivalent to mBART denoted as "mask (span)".

Results All pretraining methods yield large improvements over random initialization, in all directions and language pairs. The differences between each method are more pronounced in en↔de, whereas in en↔ne and en↔si, all models reach much lower scores, especially in en→X, probably because of low-quality training data, and a domain mismatch between the parallel and test data.

When we compare each type of input noise in isolation, we observe that masking and replacement achieve similar results, and both are better than shuffling. However, pretraining with shuffling noise alone still yields surprisingly strong results. It improves over random initialization in all experiments, and it even reaches similar BLEU scores to masking and replacements in en↔ne and en↔si. Note that, the common denominator in all pretrain-

---

[3] http://www.statmt.org/wmt18/translation-task.html

[4] https://github.com/facebookresearch/flores

ing methods is the decoder, which is trained as a conditional LM with teacher forcing and the only difference is the input conditioning context. This implies that *one* key factor in pretraining for NMT is improving the LM capabilities of the decoder.

Auxiliary Losses The best results are obtained by "mask+eMLM" and "replace+tied", both of which benefit from an encoder MLM loss. We observe, that eMLM and tying are more effective in the X→en than en→X direction, especially for en↔ne and en↔si. This makes intuitive sense because eMLM improves the representations of the encoder, which is more important for languages with limited or low-quality data. Specifically, we observe that adding eMLM improves BLEU by +0.7 for ne→en, +0.4 for si→en and tying the generator with the encoder yields +0.7 BLEU for ne→en, +0.7 BLEU for si→en. Wang et al. (2020) make a similar observation in experiments in multilingual NMT.

Incorporating RTD, however, has a negative effect in most experiments. This is unexpected, given that Clark et al. (2020) showed that pretraining text encoders with RTD outperformed MLM in NLU tasks. This warns us that methods which produce strong encoders for NLU might not necessarily improve encoders for NMT. Note that, Siddhant et al. (2020); Wang et al. (2020) have discovered similar surprising results in cross-lingual NLU tasks.

Noise Combination We also consider a combination of the best replacement-based variant "replace+tied" with shuffling. Shuffling is applied after the replacements are sampled and we limit the length of reordering to $k = 3$ to prevent extreme corruption of the input. We observe that this combination yields small gains in most experiments. We also explored more pretraining methods and configurations, including the injection of noise into the decoder, but they didn't produce significant differences in terms of BLEU (see Appendix D).

4.1.1 Parameter Sensitivity Analysis

We also explore how changing key parameters of each pretraining method affects performance in supervised NMT. We report the results in Table 2. We observe there is a "sweet-spot" for the amount of noise used in each method. Shuffling shows larger variability and we find that by making the token swaps less local (i.e., increasing k that makes the input more BoW), yields better results.

Generator Size Next, we focus on why tying the encoder and generator yields better results. Either the encoder benefits from the implicit MLM loss, or tying improves the generator and consequently its samples. We train an *untied* model with equal capacity to the encoder "replace=35% (x1.0)", and a similar model, but we sample replacements with nucleus sampling[5] (Holtzman et al., 2020) with top-p=0.9, to avoid low-probability tokens. Neither of those variants yields any measurable difference with "replace=35%", which suggests that MLM is responsible for the improvements.

Method	**en→de**		**de→en**	
	wmt18	wmt19	wmt18	wmt19
mask=15%	32.7±0.1	30.4±0.1	32.9±0.0	25.1±0.1
mask=35%	**33.3**±0.1	**30.7**±0.2	**33.2**±0.0	**25.4**±0.0
mask=50%	33.2±0.0	30.4±0.0	33.1±0.1	25.2±0.1
shuffle=3	30.3±0.1	27.9±0.0	30.5±0.0	22.9±0.1
shuffle=5	**31.6**±0.1	**28.7**±0.0	**31.7**±0.0	**23.9**±0.1
replace=15%	33.8±0.1	**30.4**±0.1	33.1±0.1	25.3±0.2
replace=35%	**33.9**±0.0	30.3±0.2	**33.5**±0.1	**25.4**±0.1
replace=50%	33.3±0.0	30.3±0.1	32.8±0.0	24.7±0.2
replace=35% (1.0x)	33.7±0.1	30.6±0.0	33.3±0.0	25.0±0.1
replace=35% +nucleus	**33.9**±0.0	**30.9**±0.1	**33.6**±0.0	**25.3**±0.0
replace=35% +RTD=4	**33.2**±0.0	29.8±0.1	32.4±0.1	**24.5**±0.0
replace=35% +RTD=6	32.9±0.1	**30.0**±0.0	**32.5**±0.0	24.4±0.1

Table 2: Supervised NMT results (mean and SEM of 3 runs), for different configurations of pretrained models.

RTD Position We also explore if the position of the RTD head is responsible for its negative effects in NMT, as it might force the encoder to preserve information irrelevant for NMT to its outputs. To test this, we train a model with RTD over its fourth (RTD=4) instead of last/top (RTD=6) layer. We find that this change has a marginal effect on BLEU.

4.2 Semi-supervised Translation

In Table 3 we report results for semi-supervised NMT, using backtranslation. Both the forward and backward TM are initialized from the same model. We generate the backtranslations from the same monolingual data that we used for pretraining with greedy-sampling, which is preferable for weak TMs (Edunov et al., 2018), and upsample the real data to maintain a 1:1 ratio with the synthetic data.

Results Backtranslation yields significant gains in all experiments, and the initialization from pretrained models boosts performance even more. The relevant performance between methods is consistent with the supervised NMT, but their differences shrink further. We suspect that adding more data narrows the room for improvement of pretraining.

One exception is the initialization from "shuffle=5", which yields marginal gains in en↔de and even fails to reach the randomly initialized model

Method	en→de		de→en		en→ne	ne→en	en→si	si→en
	wmt18	wmt19	wmt18	wmt19				
random	34.8	29.2	37.4	24.8	5.8	13.9	6.5	12.9
mask=35%	38.3	31.5	38.8	27.7	6.3	14.9	6.5	13.5
mask=35% +eMLM	38.4	**31.8**	**39.0**	27.4	**6.6**	15.0	**7.3**	**14.2**
mask=35% (span)	38.3	31.6	39.0	27.5	6.4	14.4	6.3	13.4
shuffle=5	36.2	30.8	36.9	26.1	6.4	13.5	6.3	11.9
replace=35%	38.2	31.2	38.6	27.5	6.4	14.7	6.0	13.1
replace=35% +RTD	37.7	31.3	38.2	27.5	6.1	14.2	5.9	12.8
replace=35% +tied	**38.5**	31.7	38.8	**27.8**	6.4	**15.1**	6.6	13.7
+shuffle=3	38.2	31.6	38.6	27.0	6.5	15.4	7.4	14.2

Table 3: Finetuning results to **semisupervised** NMT. Each TM is trained on the concatenation of real and back-translated sentences, obtained by a backward TM initialized from the same pretrained model.

Method	en→de		de→en	
	wmt18	wmt19	wmt18	wmt19
mask=35%	**25.6**	**19.1**	**29.3**	**20.3**
mask=35% +eMLM	25.2	18.7	28.9	19.8
mask=35% (span)	24.7	18.1	28.3	19.7
shuffle=5	17.1	13.1	20.5	15.6
replace=35%	23.3	17.5	27.4	19.4
replace=35% +RTD	22.8	16.8	26.5	18.5
replace=35% +tied	24.1	17.8	27.8	19.3

Table 4: Finetuning results to **unsupervised** NMT.

in si↔en and ne↔en. Note that, one of the advantages of backtranslation is improving the LM capabilities of the decoder with the addition of more clean target data. We hypothesize that shuffling noise mainly pretrains the decoder as a LM, and its benefits are largely neutralized by backtranslation.

4.3 Unsupervised Translation

In Table 4 we evaluate the pretrained models on unsupervised NMT (Artetxe et al., 2018; Lample et al., 2018), using *only* the monolingual data. In this experiment we focus on en↔de, because unsupervised NMT in en↔ne and en↔si yields very low BLEU scores (Guzmán et al., 2019; Liu et al., 2020). In each batch, we generate backtranslations on-the-fly in the target language and the model is optimized to reconstruct the original sentences (i.e., $\text{en}\rightarrow\text{de}'\rightarrow\widehat{\text{en}}$), following the same finetuning process as mBART (see Appendix B for details).

Results Unlike the experiments on parallel data (§4.1, §4.2), unsupervised NMT reveals large differences between pretrained models. Strikingly, "shuffle=5" yields the lowest BLEU scores by a large margin. This further supports the hypothesis that its primary strength is its decoder. We hypothesize that all pretraining methods produce strong decoders, but not encoders. Since in supervised NMT the models can learn source-to-target mappings from the parallel data, models with better cross-lingual abilities have a small edge. However, in the unsupervised setting having a fluent decoder alone is not enough as the models have to rely on their own word-translation capabilities (§4.5) to be able to produce sufficient backtranslations.

Another unexpected result is that pretraining with masked inputs outperforms replacements. Replacement-based models should intuitively be favoured, because the mistakes injected by the generator during pretraining resemble those produced by backtranslation. Masking-based models, however, are exposed to very different inputs without any signal from parallel data to help them transition to the new training regime, unlike supervised NMT.

In our analysis (§5.2), we find that masking biases models towards copying from the input, whereas replacements make models more "cautious" because some input words are fake. We hypothesize that the ability of mask-based models to copy words, such as dates or named entities, is critical to kickstart the backtranslation process.

4.4 Supervised Translation Ablations

To estimate how important of each part of the pretrained model is for NMT, we conduct an ablation experiment. First, we transfer all the weights of a pretrained model to a downstream model, except the weights of the ablated component. Next, we freeze the pretrained weights for all components except for the ablated one, which we reinitialise randomly. Finally we finetune only the ablated component, to isolate the effects on the final score to the component and prevent the other components from compensating (details in Appendix B.1).

We divide the model into four parts: (1) the *embedding* matrix, which is used for both the encoder and decoder embeddings and the vocabulary (output) projection, (2) the *encoder* layers, (3) the *decoder* layers and (4) the *cross-attention* layers, which link the decoder with the encoder. We re-

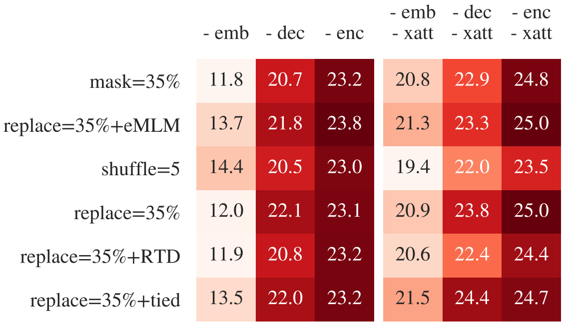

	- emb	- dec	- enc	- emb - xatt	- dec - xatt	- enc - xatt
mask=35%	11.8	20.7	23.2	20.8	22.9	24.8
replace=35%+eMLM	13.7	21.8	23.8	21.3	23.3	25.0
shuffle=5	14.4	20.5	23.0	19.4	22.0	23.5
replace=35%	12.0	22.1	23.1	20.9	23.8	25.0
replace=35%+RTD	11.9	20.8	23.2	20.6	22.4	24.4
replace=35%+tied	13.5	22.0	23.2	21.5	24.4	24.7

Figure 2: Ablation results for supervised NMT on de→en (wmt19). We reset each main component individually (left) and with the cross-attention (right).

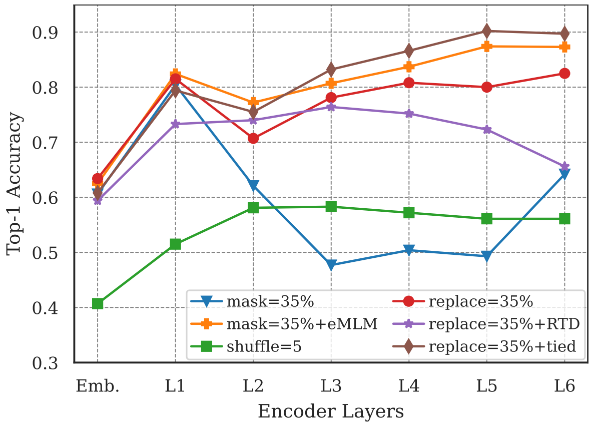

Figure 3: Parallel sentence retrieval accuracy (de→en).

port the de→en supervised NMT ablation results in Figure 2. Higher BLEU scores show that a model can better recover after an ablation, which we interpret as an indication that the ablated parameters are less important. We ablate each component both individually and combined with cross-attention, to allow the model to better learn to connect source and target representations. We find in both settings, that all models recover better after resetting their encoders than their decoders, implying that the pretrained decoder parameters are more important.

4.5 Cross-lingual Sentence Retrieval

To study the cross-lingual abilities of the encoder of each pretrained model, we evaluate them on parallel sentence retrieval. In Figure 3, we report the (per layer) accuracy of each encoder, on the de→en Tatoeba test set (Tiedemann, 2020)[6].

The results indicate that different objectives do affect the encoder's cross-lingual abilities. The fact that this is not reflected in the BLEU scores of the finetuned models further supports that even a small parallel data-set is enough to align them to similar degrees. As hypothesized, shuffling induces the least effective cross-lingual representations. The RTD loss inhibits cross-linguality, as accuracy decreases for each layer closer to the RTD loss (L6). The "mask" model exhibits an unexpected behaviour, where its accuracy drops in its middle layers before rising up again in its output layer. We do not have a satisfying explanation for this and we leave it for future work. However, adding an MLM loss over the encoder completely changes this behaviour, and both "mask+eMLM" and "replace+tied" yield the best results.

We also notice an interesting result in the accuracy of the embeddings. Shuffling noise induces poor alignment, unlike the other methods that have much better aligned embeddings. We believe that this result is connected to the unsupervised NMT results. Note that, the quality of embeddings affect the decoder as well, as their embeddings are tied.

5 Analysis

Finetuning on parallel data drives models towards similar destinations (BLEU), but the results in unsupervised NMT hinted that their starting points are different, which implies that the finetuning process itself is critical. We shift our focus to the pretrained models themselves and using a series of probes we study how pretraining methods affect their behaviour and the knowledge that they encode. For clarity, here we discuss only the en↔de results. The en↔ne and en↔si results are included in Appendix C and are consistent with this analysis.

5.1 Encoder Denoising Capabilities

In this section, we study how well the encoders of the pretrained models are able to denoise the input. For each model, first, we corrupt its inputs using the corresponding noising method and then train a linear classifier over its encoder outputs, using the identity of the original input token as the label, similar to Brunner et al. (2019). In Figure 4 we report the perplexity (PPL ↓) of each classifier evaluated on the wmt18 en↔de devset. We report separately the scores for real and corrupted tokens. We observe that, as expected, in all models the PPL over non-corrupted tokens is almost perfect (PPL ≈ 1), indicating that the encoders perfectly preserve the input in their outputs. However, the results vary significantly for corrupted tokens.

---

[6] We follow Libovický et al. (2020) and obtain the encoder sentence representations for each layer with mean pooling, followed by zero-centering per language (separately). Then, for each source sentence we retrieve its nearest neighbor from the target sentences based on cosine distance.

Method	PPL (original)	PPL (corrupted)
mask=35%	1.2	256.5
mask=35% +eMLM	1.3	91.8
shuffle=5	4.0	1274.9
replace=35%	1.4	1816.4
replace=35% +RTD	1.9	1236.2
replace=35% +tied	1.4	1233.4

Figure 4: Perplexity (PPL ↓) of the token prediction probe for the en↔de pretrained models.

Method	Entropy (original)	Entropy (corrupted)
mask=35%	0.00	2.99
mask=35% +eMLM	0.00	3.00
shuffle=5	0.09	0.16
replace=35%	0.63	2.85
replace=35% +RTD	0.68	2.93
replace=35% +tied	0.65	2.76

Figure 5: Entropy of decoder's distributions during the reconstruction of original and corrupted tokens.

Masking The representations of masked tokens yield the lowest PPL, which implies that token reconstruction happens partially in the encoder. However, the eMLM loss over the encoder makes the outputs much more predictive of the original tokens. This shows that the reconstruction loss does not push the encoder to denoise the input well enough, unlike the similar[7] but explicit signal from eMLM.

Shuffling The PPL for original tokens is very low, whereas for shuffled tokens extremely high. This shows that the encoder does *not* fix the word order but simply relays the input to the output. This is in line with Xu et al. (2021) who showed that in Transformer-based NMT word-reordering happens in the decoder, instead of the encoder.

Replacement We observe a huge gap in how predictive the representations of real and replaced tokens are. We suspect that the encoder is "misled" by the replacements and relays their information to its outputs. Both tying and RTD help the encoder to generate representations that are more predictive of the true input. RTD, however, interferes slightly with the representations of real tokens.

5.2 Decoder Uncertainty

Next, we focus on the decoder and study how its token-level uncertainty varies while it reconstructs original and corrupted tokens. For each model, first, we corrupt its inputs using the corresponding noising method and then measure the entropy of the decoder's distributions for each target token. Low entropy values indicate that the decoder predicts the target tokens with certainty, by exploiting the encoder representations. Figure 5 shows the average entropy for original and corrupted tokens.

Masking When the decoder is presented with masked inputs, it directly copies the unmasked tokens (exactly *zero* entropy). By contrast, predicting masked tokens is naturally harder, and the decoder becomes very uncertain. Adding eMLM over the encoder does not change this behaviour.

Shuffling The decoder of "shuffle=5" predicts all tokens with extreme certainty. Note that, in every step, the decoder has to choose the correct token out of N input tokens, instead of the full vocabulary, and combined with the constraints imposed by the ground-truth prefix, it is very easy for the model to find which input word to copy next. Therefore, we hypothesize that during pretraining the model mainly relies on the LM capabilities of the decoder.

Replacement We observe that the decoder is uncertain not only for fake but to a small extent, for real words as well. If a replacement is coherent and complies with the grammar of a given language, the decoder can be misled, which makes it "question" the identity of all words. RTD or tying with the generator show no clear effects.

5.3 Decoder Sensitivity to Encoder Outputs

This analysis aims to estimate the reliance of the decoder on the outputs (i.e., representations) of the encoder. First, we feed to the encoder a corrupted sentence $\boldsymbol{x} = \langle x_1, x'_2, x'_3, \ldots, x_N \rangle$, where x'_i denotes a corrupted token, and obtain its outputs $\boldsymbol{h} = \langle h_1, h'_2, h'_3, \ldots, h_N \rangle$. Then, we block the information of h'_i and measure how much it affects the reconstruction loss. We consider two blocking methods: (1) *zeroing*, in which we replace h'_i with a zero vector, and (2) *mixing*, in which we replace h'_i with random representations from other sentences in a batch. The amount by which the reconstruction loss increases implies how useful is the information in h'_i, and how sensitive the decoder is to them.

In Figure 6 we report the differences with and without blocking, in terms of the reconstruction loss (NLL). The models trained with shuffling originally yield the best reconstruction, which shows that it is comparatively the easiest noise for the decoder. Replacement is slightly harder than masking noise, because with masked inputs the decoder can easily tell when to copy and when to predict, while

---

[7] The eMLM and reconstruction losses are similar, but are applied in different places (encoder vs. decoder). The signal from reconstruction reaches the encoder through the decoder.

Method	NLL (original)	NLL (zeroed)	NLL (mixed)
mask=35%	1.46	2.69	3.35
mask=35% +eMLM	1.42	3.37	4.13
shuffle=5	0.68	8.40	10.36
replace=35%	1.55	1.49	1.50
replace=35% +RTD	1.62	1.51	1.59
replace=35% +tied	1.53	1.53	1.50

Figure 6: Change in reconstruction loss (NLL ↓) after blocking the representations of corrupting tokens.

replacements can be misleading (recall §5.2).

Masking "mask" and "mask+eMLM" reconstruct the input equally well, but when the representations of masked tokens are zeroed, "mask+eMLM" is affected more. eMLM forces the reconstruction to partially happen in the encoder (recall §5.1), so the decoder relies more on it. Mixing increases NLL even more, as we inject misleading information.

Replacement RTD leads to worse reconstruction error, which suggests that it is interfering even in the pretraining phase. Surprisingly, blocking the representations of replaced words not only does not increase the reconstruction loss but even slightly decreases it. This implies that during pretraining, the models learn to *ignore* the replaced words.

Shuffling Zeroing the representations of misplaced tokens is destructive and replacing them with random representations, increases the loss even further. The decoder focuses so heavily on putting words in the right order and has no "doubts" about their identity. Therefore, when presented with missing or misleading information, instead of "falling back" into an unconditional LM, that uses only on the target prefix, it completely fails[8].

5.4 Visualization of Encoder Representations

In Figure 7, we visualize the encoder token representations using t-SNE (van der Maaten and Hinton, 2008) and color code them based on whether the belong to original or corrupted tokens (see Appendix C.1 for details and more visualizations). Masking induces separated representation between masked and unmasked tokens, which enables the decoder to copy with certainty (see §5.2), while eMLM, that pushes the encoder to reconstruct the corrupted tokens, makes them more similar to the original ones (see §5.1). Although the representations of original and reordered tokens have a large overlap, we observe some separated clusters of original and misplaced tokens, implying that the encoder is partially aware of shuffling noise. As expected, adding RTD over the "replace" models enables allows it to identify more corrupted tokens, as shown by the size of the corresponding clusters.

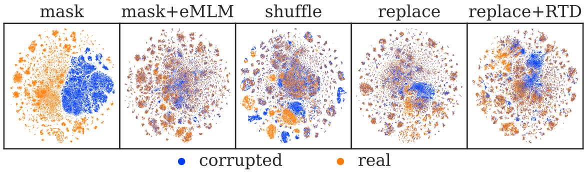

Figure 7: Visualization of encoder representations

6 Conclusions

In this work, we explore new unsupervised pretraining methods for NMT. We consider alternative objectives to masking, such as reordering or replacing input words, that produce training examples similar to real sentences.

We discover that (semi-)supervised NMT is not very sensitive to the pretraining objective. While some methods are better than others, most models converge to similar BLEU scores (§4.1, §4.2). Surprisingly, even pretraining with shuffled inputs yields competitive results with the other methods. Our ablation experiments (§4.4) imply that pretraining benefits more the decoder than the encoder.

In unsupervised NMT, however, the results vary significantly (§4.3). Shuffling noise leads to significantly worse performance, whereas masking noise, unexpectedly, yields the highest BLEU. Experiments on parallel sentence retrieval (§4.5) show that different objectives do affect the encoder cross-lingual abilities, and are reflected on the unsupervised NMT results. Through further and extensive analysis of pretrained models (§5), we find that they encode and use information in different ways.

We conclude that finetuning to each downstream NMT task is sensitive to different properties of pretrained models. (Semi-) Supervised NMT benefits from strong and fluent decoders, because the signal from the parallel data compensates for encoders with poor cross-lingual representations. Unsupervised NMT finetuning, however, requires models with good source-target mappings, and is also sensitive to certain model biases, such as the tendency to copy (§5.2) induced by mask-based pretraining.

Acknowledgments This work was conducted within the EU project Gourmet, under grant agreement No 825299. It was also supported by the UK EPSRC grant EP/S001271/1 (MTStretch). It was performed using computing resources in CSD3 using funding from the UK EPSRC (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council.

---

[8] NLL after zeroing is 8.4. It equals to PPL of $\exp(8.4) = 4447$, which is very high even for a weak unconditional LM.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Mikel Artetxe, Gorka Labaka, Noe Casas, and Eneko Agirre. 2020. Do all roads lead to rome? understanding the role of initialization in iterative back-translation. *Knowledge-Based Systems*, 206:106401.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the conference on machine translation (WMT). In *Proceedings of the Conference on Machine Translation*, pages 272–303, Belgium, Brussels.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2019. On identifiability in transformers. In *International Conference on Learning Representations*.

Franck Burlot and François Yvon. 2018. Using monolingual data in neural machine translation: a systematic study. In *Proceedings of the Conference on Machine Translation*, pages 144–155, Brussels, Belgium.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 7057–7067. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for Low-Resource neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 567–573, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. On the strengths of cross-attention in pretrained transformers for machine translation. *ArXiv*, abs/2104.08771.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 6097–6110, Hong Kong, China. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Leo Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Aizhan Imankulova, Takayuki Sato, and Mamoru Komachi. 2017. Improving low-resource neural machine translation with filtered pseudo-parallel corpus. In *Proceedings of the Workshop on Asian Translation (WAT2017)*, pages 70–78, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 66–71, Brussels, Belgium.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020a. Pre-training via paraphrasing. *Proceedings of the Advances in Neural Information Processing Systems*, 33.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jindřich Libovický, Rudolf Rosa, and Alexander Fraser. 2020. On the language neutrality of pre-trained multilingual representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1663–1674, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Conference on Machine Translation*, pages 186–191, Brussels, Belgium.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163, Valencia, Spain.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8(0):842–866.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 86–96, Berlin, Germany.

Aditya Siddhant, Melvin Johnson, Henry Tsai, Naveen Ari, Jason Riesa, Ankur Bapna, Orhan Firat, and Karthik Raman. 2020. Evaluating the cross-lingual effectiveness of massively multilingual neural machine translation. In *AAAI*, pages 8854–8861.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, J. Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *ArXiv*, abs/2104.06644.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 5926–5936, Long Beach, California, USA. PMLR.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Jörg Tiedemann. 2020. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, CA, USA.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Yiren Wang, Chengxiang Zhai, and Hany Hassan. 2020. Multi-task learning for multilingual neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1022–1034, Online. Association for Computational Linguistics.

Hongfei Xu, Josef van Genabith, Qiuhui Liu, and Deyi Xiong. 2021. Probing word translations in the transformer and trading decoder for encoder layers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–85, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5753–5763.

A Model Configuration & Training

We use 6 Transformer layers in both the encoder and the decoder, with embedding/hidden size of 512, feed-forward filter size of 2048, 8 attention heads and we apply 0.1 dropout to all layers. We optimize our models using Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_1 = 0.999$, and $\epsilon = 10^{-6}$. All models use sinusoidal positional embeddings.

We tie the weights of the embedding and output (projection) layers of all sub-networks (Press and Wolf, 2017; Inan et al., 2017), which involves the encoder, decoder and MLM generator. For pretraining, we use a learning rate of $5e^{-4}$ with a linear warm-up of 16K steps, followed by inverted squared decay. We train each model for 300K steps with mini-batches of 24K tokens on 8 Nvidia V100 GPUs, which requires approximately 4-5 days. The maximum sentence length is set to 256 tokens. For the finetuning experiments, we use a learning rate of $3e^{-5}$ with a linear warm-up of 2.5K steps and mini-batches of 12K tokens. We finetune each model for 60K in the supervised setting and 120K steps in the semi-supervised setting. We also use increased the dropout to 0.3 and set label smoothing (Szegedy et al., 2016) to 0.1, to avoid over-fitting on the limited parallel data.

B Unsupervised NMT

Instead of adding input noise, like Artetxe et al. (2018); Lample et al. (2018), we follow the finetuning process of mBART (Liu et al., 2020). To prevent models from copying the source during backtranslation and force the transition to the translation task, we allow only the most frequent tokens in the target languageto be generated for the first 2K steps. Specifically, we mask tokens with frequency less than 10^{-3}, as measure in the monolingual data. For model selection, we use a small subset of 200 sentences from the wmt18 devset.

B.1 Supervised Translation Ablations

In Figure 8, we visualize the experimental protocol for our ablation experiment. To test (i.e., ablate) a component, the process is the following:

1. We transfer all the weights of a pretrained model to a downstream model, except the weights of the ablated component.

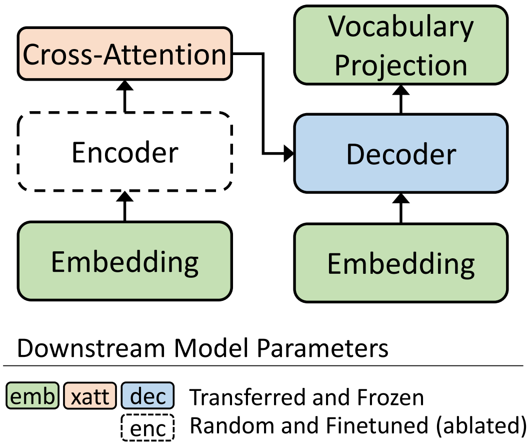

Figure 8: Visualization of the ablation experiment, using the ablation of the encoder as example. The figure shows the high-level architecture of a model and the colors correspond to the parameter set. The input embeddings of the encoder, the decoder and the vocabulary projection have the same color as they share the same parameters. During finetuning, we update only the weights of the ablated component, and all the other (pretrained) weights are frozen.

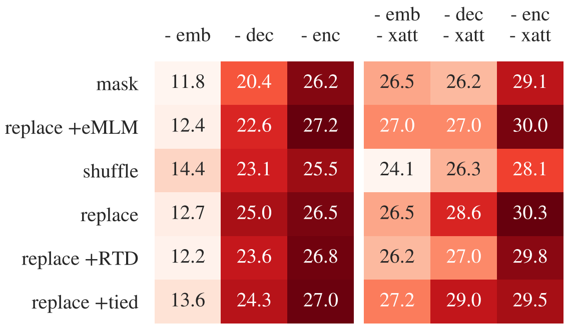

	- emb	- dec	- enc	- emb - xatt	- dec - xatt	- enc - xatt
mask	11.8	20.4	26.2	26.5	26.2	29.1
replace +eMLM	12.4	22.6	27.2	27.0	27.0	30.0
shuffle	14.4	23.1	25.5	24.1	26.3	28.1
replace	12.7	25.0	26.5	26.5	28.6	30.3
replace +RTD	12.2	23.6	26.8	26.2	27.0	29.8
replace +tied	13.6	24.3	27.0	27.2	29.0	29.5

Figure 9: Ablation results for supervised NMT on en→de (wmt19). We reset each main component individually (left) and with the cross-attention (right).

2. We freeze the pretrained weights and finetune only the ablated (i.e., randomly initialized) component.

We decided to follow this protocol, in order to isolate the effects on the final BLEU score on the ablated component, and to also prevent the other components from compensating. In concurrent work, Gheini et al. (2021) have considered a similar experimental protocol, but to study a different but related phenomenon. In Figure 8, we show the ablation results for the en→de direction.

C Analysis Results in Other Languages

In Figures 10, 11, 12, 13, 14, 15, we report the analysis results 5 on the en↔ne and ↔ne pretrained models, evaluated on their NMT dev sets.

---

we assume a shared multi-lingual vocabulary

We observe that the results are very consistent with those for en↔de.

Training Details of Probing Classifier For the analysis in Sec. 5.1, each classifier is trained on the monolingual data for 50K steps, and optimized with Adam using a learning rate of 0.0001. Only the parameters of the classifier are updated and the rest of the model remains *fixed*.

Method	**PPL (original)**	**PPL (corrupted)**
mask=35%	1.0	474.0
mask=35% +eMLM	1.1	109.7
mask-replace=35%	1.0	285.2
shuffle=5	3.0	723.2
replace=35%	1.3	1906.6
replace=35% +RTD	1.4	1596.5
replace=35% +tied	1.3	1269.1

Figure 10: Perplexity (PPL↓) of the token prediction probe for the en↔ne pretrained models.

Method	**PPL (original)**	**PPL (corrupted)**
mask=35%	1.0	408.1
mask=35% +eMLM	1.1	169.9
mask-replace=35%	1.0	356.4
shuffle=5	2.9	680.5
replace=35%	1.3	2026.4
replace=35% +RTD	1.5	1629.1
replace=35% +tied	1.3	1419.1

Figure 11: Perplexity (PPL↓) of the token prediction probe for the en↔si pretrained models.

Method	**Entropy (original)**	**Entropy (corrupted)**
mask=35%	0.00	3.14
mask=35% +eMLM	0.00	3.23
shuffle=5	0.14	0.26
replace=35%	0.66	2.95
replace=35% +RTD	0.68	3.09
replace=35% +tied	0.72	2.89

Figure 12: Decoder entropy for the reconstruction of real/corrupted tokens, for the en↔si pretrained models.

Method	**Entropy (original)**	**Entropy (corrupted)**
mask=35%	0.00	3.51
mask=35% +eMLM	0.00	3.55
shuffle=5	0.19	0.39
replace=35%	0.83	3.21
replace=35% +RTD	0.86	3.30
replace=35% +tied	0.84	3.02

Figure 13: Decoder entropy for the reconstruction of real/corrupted tokens, for the en↔ne pretrained models.

Method	**NLL (original)**	**NLL (zeroed)**	**NLL (mixed)**
mask=35%	1.67	2.46	3.03
mask=35% +eMLM	1.69	4.16	4.09
shuffle=5	1.25	9.95	10.12
replace=35%	1.76	1.75	1.71
replace=35% +RTD	1.84	1.78	1.83
replace=35% +tied	1.74	1.60	1.63

Figure 14: Reconstruction loss (NLL ↓) with/without blocking the outputs of corrupted tokens (en↔si).

Method	**NLL (original)**	**NLL (zeroed)**	**NLL (mixed)**
mask=35%	1.93	3.12	3.60
mask=35% +eMLM	2.00	3.76	4.22
shuffle=5	1.45	9.80	10.12
replace=35%	2.08	2.01	1.97
replace=35% +RTD	2.17	2.05	2.09
replace=35% +tied	2.07	1.95	1.93

Figure 15: Reconstruction loss (NLL ↓) with/without blocking the outputs of corrupted tokens (en↔ne).

C.1 Visualization of Encoder Representations

In this section, we visually inspect the input token representations. We compare each method based on how the representations evolve through the layers of the encoder, by focusing on two aspects of each token, (1) its *language* and (2) whether it has been *corrupted* or not. The goal is to inspect how each model encodes these two types of information about the input tokens.

Methodology We sample 5K sentences from the English-German monolingual data and pass them through the encoder of each model using corresponding noising method. For each token, we keep its representations from every layer and label them by language and noise. We keep only the representations of the 2K most frequent tokens and exclude the representations of special tokens, such as the `[BOS]`, `[EOS]`, and language IDs, which significantly skew the results. The final dataset contains approximately 100K token representations per layer (600K in total). For the visualization, we project to 2D with t-SNE (van der Maaten and Hinton, 2008). For each model, we visualize the representations of its encoder per layer (L1 to L6), which we colour-code by language (top-row) and the identity (bottom-row) of each token. L1 refers to the outputs of the first Transformer layer, therefore the tokens have been contextualized once.

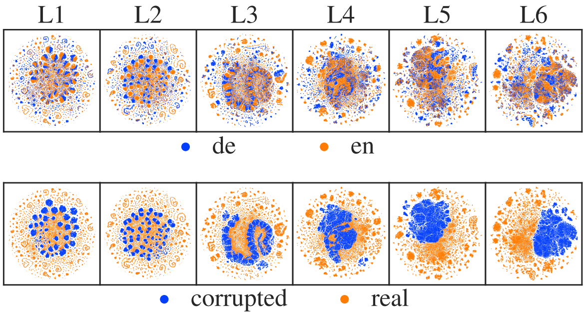

Figure 16: Visualization of encoder representations from the “mask=35%” model.

Masking In Fig. 16 we visualize the encoder of the “mask=35%” pretrained model. Real and masked tokens occupy different regions, whereas the tokens from each language are much closer to each other. All models trained with masking noise exhibit similar behaviour. In the first layer, the masked tokens are organized into multiple small clusters, but the encoder progressively groups them into larger structures. We visually verify that the encoder keeps the masked tokens separated even in the last layer. This aligns with our findings in Sec. 5.1, which suggest that the reconstruction loss does not incentivize the encoder to denoise the representations of masked tokens. Also, it enables the model to easily identify the real tokens and can copy them, as discussed in Sec. 5.2.

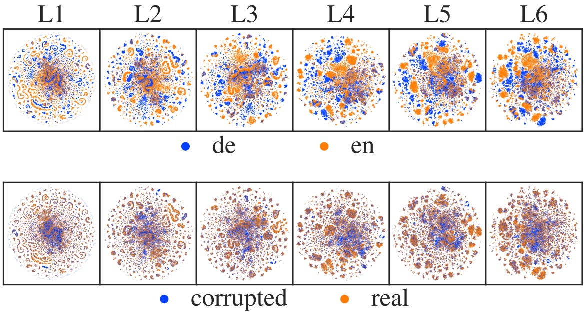

Figure 17: Visualization of encoder representations from the “mask=35%+eMLM” model.

Masking+eMLM In Fig. 17 we visualize the “mask=35%+MLM” model. Adding eMLM makes the masked and real tokens are indistinguishable from each other. Intuitively, to minimize the MLM loss the encoder must generate representations that are predictive of the true identity of the masked tokens, therefore similar to real tokens. We observe a small overlap between English and German tokens, as there are more language-specific clusters. We believe that is because the eMLM loss pushes the representation to better encode the grammar and semantics of each token.

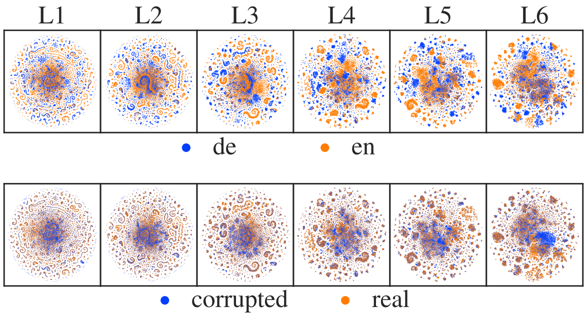

Figure 18: Layer-wise visualization of encoder representations from the “replace=35%” model.

Replacement In Fig. 18 we visualize the “replace=35%” model. There is a moderate separation between languages, slightly less than the “mask=35%+MLM” model. However, we observe that the representations of real and fake tokens generally overlap with each other, unlike the “mask=35%” model, especially in the lower layers. This is because the model is always given as input embeddings of actual words and not `[MASK]`, which unless contextualized all of them are treated the same. Only in the last layer, we can see the formation of a fake-only cluster. This suggests that the pretraining objective (i.e., reconstruction) creates a bias towards discriminating between real and fake token. However, the separation is not as extreme as in the masked-based models but is not obvious if the encoder can’t or is not biased to more clearly separate them.

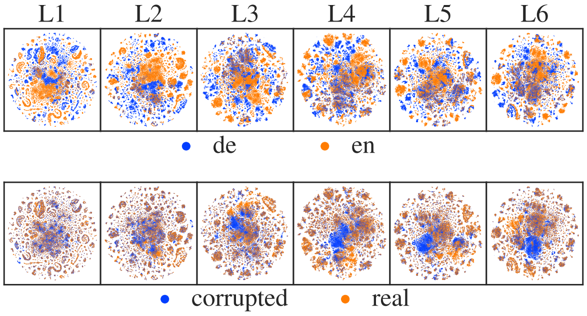

Figure 19: Visualization of representations from the “replace=35%+RTD=4” model, that is trained with an RTD head over the 4th layer of the encoder.

Replacement Token Detection In Fig. 19 we visualize the “replace=35%+RTD=4” model, that is trained with an RTD head over the 4th layer of the encoder. The only difference with the “replace=35%+RTD=6” is that the effects of RTD start to show earlier. Compared to the “replace=35” the

Method	**en→de**		**de→en**		**en→ne**	**ne→en**	**en→si**	**si→en**
	wmt18	wmt19	wmt18	wmt19				
random	26.2±0.1	25.3±0.1	27.6±0.1	19.1±0.3	3.3±0.1	6.5±0.1	2.5±0.1	6.5±0.1
mask=35%	33.3±0.1	30.7±0.2	33.2±0.0	**25.4**±0.0	5.1±0.1	10.2±0.1	3.7±0.0	10.0±0.1
mask=35% +eMLM	33.4±0.0	30.6±0.1	33.5±0.1	25.2±0.2	**5.3**±0.0	**10.8**±0.1	**4.0**±0.0	10.4±0.1
mask=35% (span)	33.3±0.1	30.5±0.1	33.4±0.0	25.2±0.0	5.1±0.1	10.1±0.1	3.9±0.1	9.9±0.1
shuffle=5	31.6±0.1	28.7±0.0	31.7±0.0	23.9±0.1	4.9±0.0	9.9±0.1	3.4±0.0	10.1±0.1
replace=35%	33.9±0.0	30.3±0.2	33.5±0.1	**25.4**±0.1	5.1±0.1	9.9±0.1	3.7±0.0	9.8±0.0
replace=35% +RTD	32.9±0.1	30.0±0.0	32.5±0.0	24.4±0.1	5.0±0.0	9.9±0.1	3.4±0.1	9.7±0.2
replace=35% +tied	**34.2**±0.0	**30.8**±0.1	**33.7**±0.1	25.3±0.2	**5.3**±0.0	10.6±0.1	3.7±0.0	**10.5**±0.1
+ shuffle=3	34.0±0.0	31.1±0.1	33.4±0.1	25.1±0.2	5.5±0.0	11.0±0.0	4.0±0.0	10.8±0.1
+ dec: mask=15%	33.9±0.1	30.9±0.0	33.6±0.1	25.3±0.2	5.5±0.0	10.5±0.0	3.9±0.0	10.4±0.0
+ dec: replace=15%	34.5±0.1	30.7±0.1	33.4±0.0	25.6±0.1	5.6±0.0	10.5±0.1	3.9±0.0	10.7±0.1

Table 5: Finetuning results to supervised NMT. "dec:X" denotes method that add noise to the the decoder. We report the average of 3 runs and the standard error of the mean.

representations are more clustered and less spread-out, even in the lower layers. The real and fake tokens are much better separated, and the separation peaks at layer 4, which visually verifies the bias introduced by RTD, but the separation is not as extreme as for masking noise. Although this suggests that separating masked/original words is harder than real/fake, it also depends on the weight used for the RTD loss during pretraining, which is not something we have explored. Also, the visualization suggests that the separation remains approximately constant in the remaining layers. There is no perceptible difference with "replace=35" in terms of language.

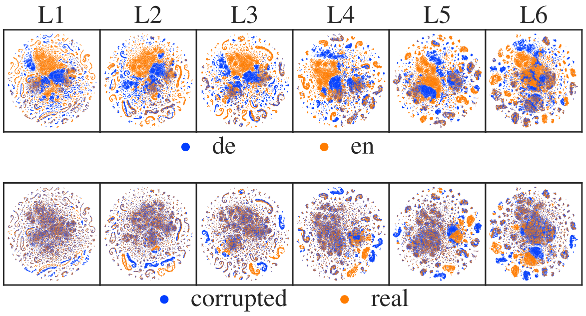

Figure 20: Visualization of representations from the "shuffle=5" model.

Shuffling In Fig. 20 we visualize the "shuffle=5" model. We observe a strong separation between languages, that slightly decreases in the upper layers. The language clusters are relatively large, unlike the much smaller and local language-specific clusters seen in the other models. As for noise, initially all tokens are represented similarly, but as the encoder re-contextualizes the input, it puts more misplaced tokens in separate clusters. This shows that the model becomes progressively more "aware" about the misplaced tokens in the input. After manual inspection of the clusters with corrupted tokens, we found that the majority of them contain punctuation marks. This makes intuitive sense, as it should be easy for the model to identify punctuation marks that has been misplaced.

D Additional NMT Experiments

In Table 5 we report some additional results for supervised NMT 4.1 omitted from the main paper. Besides combining shuffling and word replacements in the input, we also introduce noise in the decoder side, by randomly replacing 15% words of the decoder's input with the `[MASK]` token ("dec:mask=15%") or with samples from the generator ("dec:replace=15%"). Note that, we reduce the amount of noise in this case to avoid disrupting training. Overall, we observe that both methods increase performance in some language pairs, but the improvements are marginal.

We also experimented with RTD over shuffled inputs, by training the model to explicitly detect if words were misplaced or not, but this configuration lead to poor results.

Bowman et al. (2016) reported that masking more than 20% of words in the decoder hurts its LM capabilities.

Knowledge-Grounded Dialogue Generation with Term-level De-noising

Wen Zheng[1], **Natasa Milic-Frayling**[1], **Ke Zhou**[1,2]
[1]University of Nottingham
[2]Nokia Bell Labs
{wen.zheng,ke.zhou}@nottingham.ac.uk natasa-milic@frayling.net

Abstract

Dialogue generation has been improved through injecting knowledge into generative models. However, addition of knowledge through simple selection of sentences or paragraphs is likely to introduce noise and diminish the effectiveness of the generative models. In this paper, we present a novel **K**nowledge **T**erm **W**eighting **M**odel (KTWM) that incorporates term-level de-noising of the selected knowledge. KTWM includes a module for generating Simulated Response Vectors (SRVs) and uses SRVs attention distributions with the knowledge embeddings to determine knowledge term weights. Our experiments demonstrate that KTWM, combined with various knowledge selection algorithms, consistently achieves statistically significant improvements over methods without term weighting when applied to two publicly available datasets Wizard of Wikipedia (Wiz) and Holl-E. The results are particularly improved for the Wiz test data with unseen topics, demonstrating the robustness of the KTWM noise-reduction approach.

1 Introduction

Research in dialogue generation has rapidly evolved from sequence-to-sequence (Sutskever et al., 2014) and Transformer models (Vaswani et al., 2017) to approaches with pre-trained models such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019) and T5 (Raffel et al., 2020). More recently, it included techniques that use knowledge, in addition to the original posts, to improve the quality of the generated responses (Ghazvininejad et al. (2018), Moghe et al. (2018), Dinan et al. (2019), Galley et al. (2019), Lian et al. (2019), Zheng and Zhou (2019), Zhao et al. (2020a), Zhao et al. (2020b)).[1] This approach is referred to as *knowledge-grounded dialogue generation* and is the primary concern of this paper.

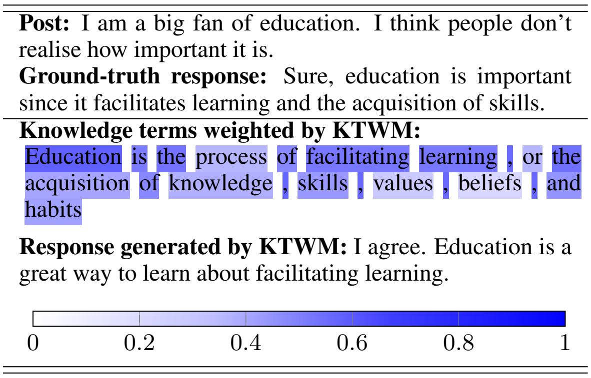

Post: I am a big fan of education. I think people don't realise how important it is.
Ground-truth response: Sure, education is important since it facilitates learning and the acquisition of skills.

Knowledge terms weighted by KTWM:
Education is the process of facilitating learning , or the acquisition of knowledge , skills , values , beliefs , and habits

Response generated by KTWM: I agree. Education is a great way to learn about facilitating learning.

Table 1: Example of a post, ground truth response, injected knowledge and generated response by the Knowledge Term Weighting Model (KTWM). The term highlights indicate the predicted probability of a term being useful.

In particular, we consider the key issue of effectively incorporating the selected knowledge into the generation process. For example, Weston et al. (2018) apply a *retrieve and refine* method to expand the post with the retrieved knowledge and then use it in the generation process. Lian et al. (2019) consider the post and response *posterior distributions* and the post *prior distribution* to train jointly the model for knowledge selection and response generation. Kim et al. (2020) view the knowledge selection as a sequential decision problem, first selecting the best ranked knowledge using a sequential latent variable model, and then generating a response based on selected knowledge.

To the best of our knowledge, all prior approaches focus on the selection and injection of knowledge at the sentence or paragraph level. However, that makes it hard to control for potential

---

[1] Previous works used a variety of terms to refer to a post such as 'question', 'utterance','source' and 'query'. Similarly for response they used 'answer', 'response', and 'target'. In this paper we call the first role the 'post' and the second role the 'response' and we aim to generate the response for the given post.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2972–2983
August 1–6, 2021.

noise, i.e., for inclusion of non-relevant words, and previous studies (Galley et al. (2019), Zheng et al. (2020)) have shown that adding noise can decrease the response generation quality. Therefore, it is important to investigate whether and how we can adjust the contributions of terms in the selected knowledge. Prior research has not considered that issue systematically.

Our paper fills this gap by introducing a novel *Knowledge Term Weighting Model* (KTWM) for dialogue generation, which effectively estimates term weights of the injected knowledge and incorporates such weights into the response generation. The response generation thus benefits from such nuanced term-level knowledge weighting, promoting important knowledge terms rather than treating equally all the terms in the selected sentences. In Table 1 we show an example of the KTWM term weighting and its generated response: the terms 'education', 'is', 'facilitating' and 'learning' are given higher weights correctly as they do appear in the ground-truth response, while the words 'values' and 'beliefs' are correctly assigned lower scores.

We conducted an extensive range of experiments with KTWM on two publicly available datasets: Wiz (with seen and unseen test topics) (Dinan et al., 2019) and Holl-E (Moghe et al., 2018). KTWM performs consistently well with different selections of knowledge, specifically with Post-KS (Lian et al., 2019), SKT (Sequential Latent-Knowledge Selection) (Kim et al., 2020) and TED (Transformer with Expanded Decoder) (Zheng and Zhou, 2019). Our work achieves both a superior performance in knowledge-grounded dialogue generation and new insights into the impact of the knowledge term weighting on that performance. The code of our method is publicly available at *https://github.com/tonywenuon/acl2021_ktwm* and enables reproducibility of our results.

2 Related Work

The knowledge-grounded dialogue generation can be tackled by decomposing it into two subproblems: (1) selecting knowledge from a large pool of candidates (*knowledge selection*), and (2) generating a response from the selected knowledge and context (*knowledge-grounded response generation*).

Knowledge-grounded Response Generation Ever since the knowledge-based dialogue generation task was released by DSTC-7 (Galley et al., 2019), research interest in the topic has been steadily growing. Ghazvininejad et al. (2018) proposed a multi-task learning approach to produce responses. The posts and knowledge are used in the encoders and share the same decoder parameters. Luan et al. (2017) expanded the scope and introduced personality information into the model. They assumed that the trainable parameters can potentially capture persona from the non-conversational data (Tweets). Yavuz et al. (2019) adopted pointer-generator networks within a hierarchical framework that enabled them to include external knowledge in addition to the context. Ye et al. (2020) proposed a latent variable based generative model, which contains a joint attention mechanism conditioned on both context and external knowledge. Li et al. (2019) applied a deliberation network to create a two-stage generative model that combines both context and knowledge and, in the second generation stage, makes use of the outputs from the first stage. Zheng and Zhou (2019) proposed Transformer with Expanded Decoder (TED) architecture that assigns different weights to different knowledge sources and incorporates them into the generation process.

While the above approaches and models focus on incorporating knowledge and context to generate responses, they do that at the sentence or paragraph level. Our work deals with the quality of the incorporated knowledge at the term level, weighing all the individual knowledge terms when generating the responses.

Knowledge Selection Considering the mechanisms for response generation, Weston et al. (2018) proposed to retrieve candidate content from a knowledge set and use it to expand the post. The result is a truncated sequence that represents a refined post. (Lian et al., 2019) select the knowledge by approximating the prior distribution (i.e., $p(knowledge|post)$) with the posterior-distribution (i.e., $p(knowledge|post, response)$) and then inject it into the decoder. Kim et al. (2020) trained a knowledge selection module and a response generation module jointly, but treated the knowledge selection as a sequential decision problem, using input and knowledge from the previous turns to select the knowledge in the subsequent turns. Zheng et al. (2020) separated the knowledge selection process from the generation process so that all the downstream generation tasks can use the selected knowledge. They mapped posts to the best knowledge representations in both the training and the testing phase, and used the learned models to rank new post-knowledge pairs.

In this context, KTWM can be viewed as an optimization step following the knowledge selection. It is focused on learning knowledge term weights to distinguish between relevant and non-relevant terms and weighing higher those that are useful for the response generation.

3 Method

In this section we introduce the basic concepts and describe in detail our method KTWM for term-weighting of the injected knowledge. We assume that for a collection of posts P and responses R, we have a collection $\{K_{pr}\}$ of knowledge sets with sentences relevant to the specific post-response pair (p, r). For a given pair (p, r) we consider a knowledge injection process that involves three stages: (1) knowledge selection, (2) knowledge term-weighting, and (3) decoding with the weighted knowledge terms. Our primary focus is on (2), i.e., the effectiveness of the term-weighting for the knowledge incorporated in the KTWM. Thus we provide a detailed description of the term-weighting model (Figure 1) and the use of the KTWM decoder (Figure 2).

3.1 Knowledge Selection and Representation

We represent each post p, response r, and a knowledge sentence k as a vector of terms. The set K_{pr} typically contains multiple knowledge sentences and we use BM25 retrieval method to rank the sentences by their relevance to the post (in the test phase) or response (in the training phase). For knowledge injection we take the top ranked sentence. When the knowledge injection requires a specific number of terms to be used, we include additional sentences from the ranked list to meet that requirement (used in §4.4.3).

When a knowledge sentence k is retrieved based on a response r as a query, we define a ground truth vector GT_{know} for the knowledge k with the weight of 1 assigned to the knowledge terms that are present in r and the weight of 0 assigned to those that are not, i.e., $GT_{know} = (e_1, e_2, \ldots, e_l)$, where $e_i \in \{0, 1\}, i = 1, \ldots, l$.

Encoders. We adopt Transformer (Vaswani et al., 2017) as the backbone framework for the training and testing of KTWM. Transformer encoder consists of a self-attention layer and a transition layer involving the layer normalisation and residual network. Formally, the attention is defined as

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_m}}\right)V, \tag{1}$$

where Q, K, and V are embedding matrices and d_m is the embedding dimension of the model. First we compute the dot similarity of the Q and K and then apply the weighted summation with V. The representation of Q is updated with the information from K and V. If Q, K, V originate from the same source, e.g., an input post, the attention is referred to as self-attention. Otherwise, if they originate from different sources, e.g., Q relates to the decoding token and K and V are from a post, the attention turns to be a mutual-attention operation.

Figure 1 shows transformer encoders (*encoders* for short) used for the post, knowledge, and response representations and processing. We use w to designate an original term and $\widehat{w}$ to designate the term's representation. In Figure 1, n, m, and l are three pre-defined hyper-parameters which refer to the length of the post (p), response (r), and knowledge (k), respectively (e.g. w_{pi} means the i-th term of the post). Any sequence that is longer or shorter than the given length will be truncated or padded to the given length. By applying the encoder

$$V_{post} = Encoder(w_i)(i \in [1, n]) \tag{2}$$

we obtain the post terms representations V_{post}, comprising $\widehat{w}_{p1}, \widehat{w}_{p2}, \ldots, \widehat{w}_{pn}$ (in Figure 1), from the original terms $w_{p1}, w_{p2}, \ldots, w_{pn}$. Similarly to V_{post} in Eq. (2), we obtain V_{know} and V_{resp} as term representations of the corresponding knowledge and the response, respectively.

3.2 Knowledge Term Weighting

The fundamental premise of our approach is that knowledge terms related to or present in the response should be more effective in improving dialogue generation. Thus, it can be beneficial to use methods such as attention distribution of response and knowledge embeddings to determine the weights of individual knowledge terms. However, in the real setting and during the test phase, we can only use terms and knowledge related to the post. Furthermore, the post embeddings can significantly differ from the response ones. Thus, assigning weights to the knowledge terms based on

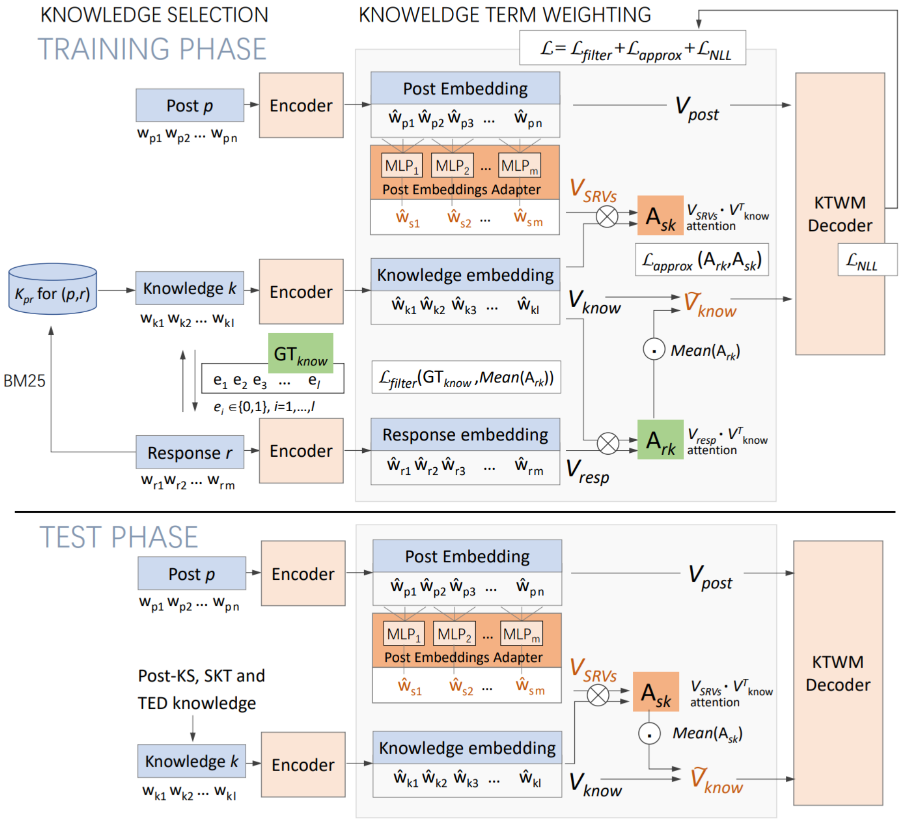

Figure 1: Architecture of the Knowledge Term Weighting Model (KTWM) showing the operations in the training and test phase. ⊗ designates matrix multiplication; ⊙ designates element-wise multiplication.

their similarity to post embeddings is unlikely to be sufficient (Xing et al., 2018).

For that reason, we aim to learn how to transform the post embeddings to be effective in knowledge term weighting. We achieve that by training a *Post Embeddings Adapter* that can, for a new post, generate *Simulated Response Vectors* (SRVs) and use them in place of the response vectors to score post related knowledge terms.

To that effect, we introduce a set of Multi-Layer Perceptrons (MLPs):

$$MLP = \sum_{i=1}^{n} \widehat{w}_{pi} W_i + b \tag{3}$$

$$\widehat{w}_{sj} = MLP_j(\widehat{w}_{p1}, \widehat{w}_{p2}, \ldots, \widehat{w}_{pn})(j \in [1, m]) \tag{4}$$

where W_i and b are trainable parameters for each term p_i of the post p; $\widehat{w}_{sj}$ is the representation of the j-th term of the simulated response vector (SRV). The number of MLPs is the same as the number of terms in a given response.

During the training phase, MLPs learn the transformation of the post embeddings into SRVs that captures the ground truth response representation for a given post p. SRVs are then used to assign appropriate weights to the knowledge terms when response information is not available.

SRVs Approximation and Training. The training phase begins with V_{post}, V_{know}, V_{resp} and randomly initiated parameters of MLPs to produce the initial set of V_{SRVs} for a given post. Each iteration then involves comparison of (a) the response embeddings V_{resp} and knowledge embeddings V_{know}, and (b) SRVs with the knowledge embeddings V_{know}. More precisely, we compute the term-wise attention distributions A_{rk} and A_{sk}:

$$A_{rk} = sigmoid(V_{resp} V_{know}^T) \tag{5}$$

$$A_{sk} = sigmoid(V_{SRVs} V_{know}^T) \tag{6}$$

where $A_{rk} \in \mathbb{R}^{m \times l}$ and $A_{sk} \in \mathbb{R}^{m \times l}$; m and l are hyper-parameters that are the maximum length of the response and knowledge sentence.

A_{rk} reflects the relationship between the response terms and the knowledge terms: for each response term, A_{rk} includes attention scores with all knowledge terms. Similarly, A_{sk} includes attention scores between SRVs and the knowledge representations. The knowledge terms with larger response-knowledge attention scores are expected to produce output closer to the true response. In the training phase, that is guided by the *filtering loss* for A_{rk}:

$$\mathcal{L}_{filter} = BCE(GT_{know}, Mean(A_{rk})) \quad (7)$$

where GT_{know} is the *knowledge ground truth* vector which indicates whether the knowledge terms appear in the corresponding response or not and BCE is the Binary Cross Entropy loss function. $Mean(\cdot)$ computes the mean values for knowledge terms (in the matrix columns) across response terms ($Mean(\cdot) \in \mathbb{R}^l$).

At the same time we aim to train MLPs to create SRVs similar to the response representations V_{resp}. In each iteration we compute and compare A_{sk} to A_{rk} and apply the *approximation loss* function:

$$\mathcal{L}_{approx} = MSE(Mean(A_{rk}), Mean(A_{sk})) \quad (8)$$

where $MSE(\cdot)$ is the Mean Squared Error function. $Mean(\cdot)$ of A_{rk} and A_{sk} produces l-length knowledge term vectors whose values are used to characterise the importance of each knowledge term. We use these weights to update the knowledge vector:

$$\widetilde{V}_{know} = Mean(A_k) \odot V_{know} \quad (9)$$

where $\odot$ denotes element-wise multiplication and A_k corresponds to A_{rk} in the training phase and to A_{sk} in the test phase. V_{post} and the weighted knowledge vector $\widetilde{V}_{know}$ become input for the KTWM decoder.

3.3 KTWM Decoder

In order to incorporate multiple sources of input, we adopt a decoder design that is similar to the TED model by Zheng and Zhou (2019). Figure 2 shows the architecture of our KTWM decoder. The blue frames are the standard Transformer decoder set-up with a self-attention layer and a mutual-attention layer (for the post), followed by a feed-forward layer.

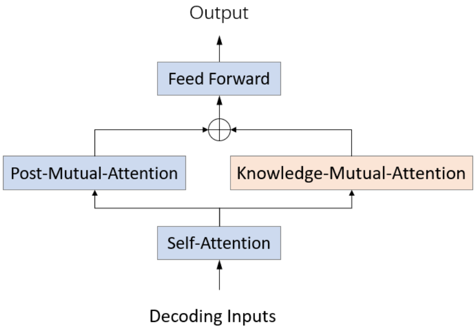

Figure 2: Knowledge Term Weighting Model Decoder.

KTWM includes an additional knowledge-mutual-attention layer which applies the same process to the knowledge, i.e., replicates the post-mutual-attention layer for the knowledge. However, while TED focuses on assigning different weights to different sources, KTWM is already provided with scored knowledge terms. We use V_{PMA} to denote the post-mutual attention, V_{KMA} for knowledge-mutual attention and V_{dec} for the decoding tokens representation matrix. With the attention defined by Eq. (1), we can express:

$$V_{PMA} = Attention(V_{dec}, V_{post}, V_{post}) \quad (10)$$

$$V_{KMA} = Attention(V_{dec}, \widetilde{V}_{know}, \widetilde{V}_{know}). \quad (11)$$

The *final mutual attention* V_{MA} in the decoder is then calculated from V_{PMA} and V_{KMA}:

$$V_{MA} = V_{PMA} \oplus V_{KMA} \quad (12)$$

where $\oplus$ means element-wise summation. The feed forward layer is a standard Transformer transition layer (Vaswani et al. (2017)).

Finally, we adopt Negative Log Likelihood (NLL) to train the model:

$$\mathcal{L}_{NLL} = -\sum_{t=1}^{m} logP(y_t|y_{<t}, p, k). \quad (13)$$

Given a post (p), knowledge (k), and the previously predicted terms ($y_{<t}$), $\mathcal{L}_{NLL}$ maximises the probability of the currently predicted term. During the training phase, $P(y_t|y_{<t}, p, k)$ is replaced with $P(r_t|r_{<t}, p, k)$, i.e., we use the ground truth response as the input instead of the model output from the previous steps (Goyal et al., 2016).

We assume that all three loss functions are equally important and create the final loss function as a sum:

$$\mathcal{L} = \mathcal{L}_{filter} + \mathcal{L}_{approx} + \mathcal{L}_{NLL} \quad (14)$$

KTWM thus provides a flexible learning framework, enabling injection of knowledge based on different selection criteria. We compare KTWM effectiveness when used with Post-KS, SKT and TED model by incorporating the knowledge that each of these methods selects.

4 Experiments

We conduct empirical evaluation of KTWM compared to state-of-the-art baselines.

4.1 Datasets

In our experiments we use two publicly available datasets: Wizard of Wikipedia (Dinan et al., 2019) and Holl-E (Moghe et al., 2018). Both are purposefully created by humans editors to support dialogue generation research.

Wizard of Wikipedia (Wiz). Dinan et al. (2019) employed Amazon Mechanical Turk (MTurk) workers to generate the datasets. The workers can assume two different roles: a wizard (a teacher) and an apprentice (a student). An apprentice asks a question according to a given topic and a wizard answers the question based on the provided question-related information (retrieved from Wikipedia). The response can quote the retrieved knowledge or can be generated entirely by the wizard without considering the knowledge. Thus, for each question-response pair there is related knowledge that can be used for knowledge-grounded dialogue generation research.

The Wiz dataset consists of 22,311 dialogues with 201,999 dialogue turns divided into a training dataset and two test datasets referred to as *seen test set* and *unseen test set*. The seen test set includes topics that have already been seen in the training set. In the unseen dataset, there are topics that may not have been included in the training dataset.

Holl-E Moghe et al. (2018) also made use of MTurk workers to create an annotated dataset that focuses on movies as two workers talk with each other about a chosen movie. When answering another worker's question, one is provided with four sources: movie plots, reviews, comments, and fact tables related to the movies. These sources can be considered as background knowledge. The final response is produced by copying from the sources or by modifying the sources. The Holl-E dataset provides training set and test test and contains 9,071 conversations, covering 921 movies.

4.2 Metrics, Setup and Baselines

Metrics. For performance evaluation, we adopted standard lexical-based metrics: BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007) and embedding-based metric: BOW Embedding (Liu et al., 2016). BLEU 1-4 metrics measure co-occurrence of n-gram terms in two given sequences, e.g., the generated responses and the ground truth responses. METEOR is an adaptation of BLEU that considers the presence of synonyms and common word stems. BOW Embedding measures the similarity of two sentences from the semantic perspective. Specifically, it computes the *average metric*, *greedy metric* and *extrema metric* based on word embeddings of compared sentences. The average metric considers cosine distance between pairs of sentence-level representations (e.g., the predicted response and ground truth response) by averaging the representations of their constituent words and calculates the average across all pairs. The greedy metric considers the maximum cosine scores along rows and columns in the similarity matrix. The extrema metric of two sentences first creates a sentence vector with the highest word-embedding values (along the dimension) and then computes the similarity score. BLEU, METEOR[2] and BOW Embedding[3] are calculated using NLG evaluation sources.

Experiment Setup. For the sake of comparison, we fixed a set of parameters across all the experiments. The number of dimensions in embeddings is set to 100. The vocabulary size is 30,000. The vocabulary is obtained by ranking terms by word frequency in the training set. The minimum sequence length is set to 8 and the maximum length is 30. We train using mini-batches of size 64. We use Adam optimiser (Kingma and Ba, 2015) for optimisation. The initial learning rate is set to 0.001 and halved when the loss score does not decrease for two epochs. In the training phase we use response-retrieved knowledge, i.e., the sentences retrieved by BM25 algorithm using responses as queries (see Figure 1). The top 1 ranked knowledge sentence is injected into KTWM. In the test phase, we retrieve knowledge using BM25 algorithm and posts as queries. All the experiments are conducted on a single TITAN V GPU. For Wiz dataset, an experiment requires about 6 hours to complete, while for Holl-E about 2.5 hours.

Baselines. We compare KTWM with three strong baselines:

Post-KS (Lian et al., 2019) uses an elaborate knowledge selection module and injects the selected knowledge into a generative model by approximating prior-distribution (i.e., $p(k|p)$) with posterior-distribution (i.e., $p(k|p, r)$).

---

[2] https://github.com/Maluuba/nlg-eval

[3] https://github.com/neural-dialogue-metrics/EmbeddingBased.

SKT (Kim et al., 2020) considers knowledge selection as a sequential problem. It jointly trains a knowledge selection and a generative model by taking into account inputs and knowledge from previous turns.

TED (Zheng and Zhou, 2019) uses a knowledge-grounded generative model that assigns different weights to different sources when generating responses. It applies knowledge ranking using BM25, which is the same as in our setting.

4.3 Experiment Design

Our experiments focus on term weighting of the selected knowledge rather than the knowledge selection itself. Since the baseline models (Post-KS[4], SKT[5] and TED[6]) incorporate knowledge selections, we conduct a comparative evaluation of KTWM by incorporating knowledge specific to each baseline method. Furthermore, since all three baselines inject knowledge at the sentence level, by selecting the top ranked sentence, we do the same with KTWM.

4.4 Experiment Results

4.4.1 Performance of Generating Response

We summarize KTWM experiments with the Wiz and the Holl-E datasets in Table 2. Results for the Wiz seen and unseen test sets are in Table 2, sections (a) and (b), respectively. Results for the Holl-E dataset are in Table 2, section (c). Since METEOR extends BLEU metrics by considering word stems and synonyms, we take it as the main metric for discussing the experiment results. We observe that:

(1) For all of three datasets, KTWM outperforms each baseline method across all lexical and embeddings based metrics with a statistically significant difference.

(2) KTWM with Post-KS knowledge achieves the largest relative improvement considering the METEOR score: increase of 45.3%, 54.5% and 40.0% for the three test sets, respectively.

(3) For the Holl-E dataset, KTWM with TED knowledge outperforms other two baseline models. TED knowledge comprises top sentences retrieved using BM25 algorithm.

(4) On the Wiz datasets, KTWM achieves a remarkable performance in terms of BLEU-1 and METEOR scores. A consistent and strong performance in the Wiz unseen test data indicates the robustness and generalization of KTWM.

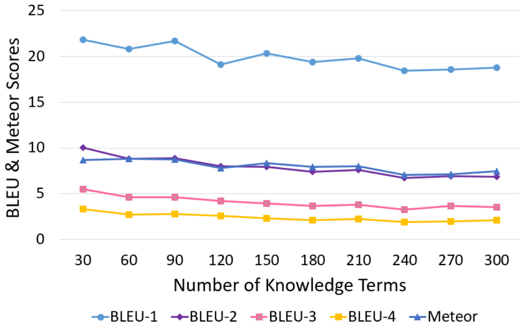

Figure 3: Effects of the increased number of knowledge terms on the KTWM performance (Wiz seen test set).

4.4.2 Results of Knowledge Term Weighting

The loss function (Eq. (7)) controls KTWM ability to distinguish between relevant and non-relevant knowledge terms, similar to a binary classifier. We set a threshold of 0.5 for a knowledge term's predicted score and consider the overlap between the predicted and the truth useful knowledge terms. This leads to precision/recall evaluation of the positive and the negative class prediction. Table 3 shows results from the Wiz seen test set. They are representative of the results for the other two datasets.

We observe that the precision of predicting useful terms is 50% and noisy terms is over 91% (with a high F-1 score, 94%). Thus KTWM term weighting is effective in detecting noisy terms while only half of the predicted useful terms overlap with the ground truth terms. Since noisy terms are assigned lower term weights, KTWM is effective improving the dialogue generation performance. Appendix A shows illustrations of the KTWM noise reduction.

4.4.3 Analysis of Input Sequence Length

We analyze the effects of knowledge de-noising by considering the *useful terms proportion* (UTP) as we increase the number of injected knowledge terms: $UTP = \frac{Num\ of\ distinct\ useful\ terms}{Num\ of all\ injected\ terms}$. We use UTP_K for UTP when the number of injected knowledge terms is K (e.g., UTP_{30} for 30 knowledge terms). Our analysis shows that UTP_{30} is 12.23% and UTP gradually decreases with additionally injected knowledge leading to UTP_{300} of only 3.35%. Figure 3 shows a gradual decline of the KTWM performance with the increased length of injected knowledge, as the proportion of noisy

---

[4] https://github.com/bzantium/Posterior-Knowledge-Selection

[5] https://github.com/bckim92/sequential-knowledge-transformer

[6] https://github.com/tonywenuon/Transformer_ED

Generation Method	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	Average	Greedy	Extrema
(a) Wiz seen test data								
Post-KS	17.56	6.35	2.68	1.35	5.96	0.611	0.364	0.334
KTWM w Post-KS knowledge	21.98*	**10.03***	**5.56***	**3.44***	**8.66***	**0.684***	0.394*	**0.376***
SKT	16.45	7.97	4.75	3.14	7.29	0.639	0.385	0.366
KTWM w SKT knowledge	**22.00***	10.0*	5.47*	3.35	8.59*	0.681*	**0.398***	0.370
TED	20.26	9.43	5.32	3.35	8.45	0.658	0.385	0.366
KTWM w TED knowledge	21.86	10.02	5.51	3.35	**8.66***	0.682*	0.394*	0.374*
(b) Wiz unseen test data								
Post-KS	17.25	5.58	2.03	0.81	5.5	0.598	0.352	0.305
KTWM w Post-KS knowledge	**21.66***	**8.98***	**4.41***	**2.41***	**8.5***	**0.681***	**0.388***	**0.361***
SKT	14.09	5.72	2.89	1.72	5.8	0.591	0.36	0.304
KTWM w SKT knowledge	20.46*	8.07*	3.85*	2.03*	7.77*	0.664*	0.38*	0.337*
TED	19.28	7.83	3.83	2.09	7.02	0.634	0.363	0.327
KTWM w TED knowledge	20.46*	8.32*	4.03*	2.17*	7.92*	0.668*	0.379*	0.342*
(c) Holl-E dataset								
Post-KS	14.07	7.07	4.96	3.81	5.98	0.639	0.382	0.333
KTWM w Post-KS knowledge	19.91*	11.0*	8.02*	6.42*	8.37*	0.675*	0.387*	0.350*
SKT	21.54	13.81	10.94	9.17	8.48	0.637	0.391	0.333
KTWM w SKT knowledge	**23.05***	13.96*	10.66	8.71	9.73*	0.673*	0.389	0.362*
TED	21.62	13.71	10.83	9.17	9.13	0.685	**0.414**	**0.366**
KTWM w TED knowledge	22.42*	**14.01***	**10.98**	**9.28**	**10.2***	**0.688***	0.402*	**0.366**

Table 2: KTWM performance on the Wiz seen and unseen test data and Holl-E dataset with different knowledge sources. Comparison with Post-KS, SKT and TED models. '*' indicates statistical significance ($p < 0.05$). **Bold** indicates the best performance for a given metric. 'w' denotes 'with', i.e., injecting the knowledge source that is used in a specific baseline model.

Name	Prec	Rec	F-1
Useful Term Prediction	0.50	0.32	0.39
Noisy Term Prediction	0.92	0.96	0.94

Table 3: Precision, Recall, and F-1 scores for the useful and noisy term predictions on the Wiz seen test set.

Name	BLEU-1	BLEU-4	METEOR	Average
KTWM	21.86	3.35	8.66	0.682
- w/o $\mathcal{L}_{Filter}$	20.69	3.67	8.77	0.661
- w/o $\mathcal{L}_{Approx}$	7.49	1.59	5.42	0.598

Table 4: Ablation study of the multi-component loss function on the Wiz seen test set. w/o means 'without'.

terms increases.

We also investigate the effects of the loss functions $\mathcal{L}_{filter}$ and $\mathcal{L}_{approx}$ on the KTWM performance by running experiments with and without them. In Table 4 we show the results on the Wiz seen test set using BM25 to select knowledge. We note that, after removing $\mathcal{L}_{filter}$ loss function, BLEU-1 and Average scores decrease, while BLEU-4 and METEOR scores increase. Since $\mathcal{L}_{filter}$ aims to ensure that relevant response terms are promoted, it is not surprising that the metrics focused on unigrams are most affected. However, this impact on KTWM is less notable than the removal of the $\mathcal{L}_{approx}$. Without $\mathcal{L}_{approx}$, the KTWM loses the ability to align simulated response vectors SRVs with the response embeddings to capture the attention distribution between the knowledge and the response embeddings that is needed to score knowledge terms. This increases the noise ratio and reduces the KTWM performance scores across all metrics.

5 Conclusions

Current knowledge-grounded dialogue models select and inject knowledge either through traditional (unsupervised) retrieval technique, such as BM25, or by incorporating knowledge selection within the dialogue generation model. Most of them incorporate knowledge as sentences or paragraphs. Past research provided evidence (Galley et al. (2019), Zheng et al. (2020)) that inserting useful terms can increase the response generation performance but it is necessary to control for negative effects of noisy terms.

In our work, we introduce a novel Knowledge Term Weighting Model (KTWM) that performs knowledge term-level weighting and de-noising of injected knowledge. We demonstrate that KTWM effectively estimates weights of knowledge terms and yields better response generation performance than state-of-the-art baseline models when evaluated on two broadly used datasets. Besides the superior response generation outcomes, our research provides important insights into the importance of the knowledge term weighting. As part of our future work we intend to (1) extend the KTWM models to incorporate multiple sources of evidence, such as balancing between selected knowledge and dialogue contexts (i.e., previous dialogue turns) and (2) take into account inter-dependencies among terms when weighting the selected knowledge.

Acknowledgements This work is partly supported by Engineering and Physical Sciences Research Council (EPSRC Grant No. EP/S515528/1, 2102871). The Titan V used for this research was donated by the NVIDIA Corporation. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations (ICLR)*.

Michel Galley, Chris Brockett, Xiang Gao, Jianfeng Gao, and Bill Dolan. 2019. Grounded response generation task at dstc7. In *AAAI Dialog System Technology Challenges Workshop*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. 2016. Professor forcing: a new algorithm for training recurrent networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4608–4616.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020. Sequential Latent Knowledge Selection for Knowledge-Grounded Dialogue. In *International Conference on Learning Representations (ICLR)*.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231.

Zekang Li, Cheng Niu, Fandong Meng, Yang Feng, Qian Li, and Jie Zhou. 2019. Incremental transformer with deliberation decoder for document grounded conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 12–21.

Rongzhong Lian, Min Xie, Fan Wang, Jinhua Peng, and Hua Wu. 2019. Learning to select knowledge for response generation in dialog systems. In *IJCAI International Joint Conference on Artificial Intelligence*, page 5081.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Yi Luan, Chris Brockett, William B Dolan, Jianfeng Gao, and Michel Galley. 2017. Multi-task learning for speaker-role adaptation in neural conversation models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 605–614.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 87–92.

Chen Xing, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Semih Yavuz, Abhinav Rastogi, Guan-Lin Chao, and Dilek Hakkani-Tur. 2019. Deepcopy: Grounded response generation with hierarchical pointer networks. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 122–132.

Hao-Tong Ye, Kai-Lin Lo, Shang-Yu Su, and Yun-Nung Chen. 2020. Knowledge-grounded response generation with deep attentional latent-variable model. *Computer Speech & Language*, 63:101069.

Xueliang Zhao, Wei Wu, Chongyang Tao, Can Xu, Dongyan Zhao, and Rui Yan. 2020a. Low-resource knowledge-grounded dialogue generation. In *International Conference on Learning Representations (ICLR)*.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020b. Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390.

Wen Zheng, Natasa Milic-Frayling, and Ke Zhou. 2020. Approximation of response knowledge retrieval in knowledge-grounded dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3581–3591.

Wen Zheng and Ke Zhou. 2019. Enhancing conversational dialogue models with grounded knowledge. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 709–718.

Appendix

A Examples of Knowledge Term Weights and KTWM Generated Responses

In Table 5 and 6 we present examples of post/response pairs and selected knowledge with terms weighted by KTWM.

As explained in §4.4.2, we use a threshold of 0.5 on term scores to classify terms into useful and noisy ones and study the effect of this selection on the overall performance of KTWM. In the examples, we visually show the weights of each terms. Terms are highlighted in different shades of blue colour according to the weight (note the colour legend at the bottom of the tables). All the examples are extracted from the Wiz seen test set. They are sorted by the number of words that exceed the threshold.

In Table 5 we see that the key words are tagged with dark blue, indicating that KTWM has assigned high weights to them. From the KTWM generated responses, we can see that if the words appear in the post and ground-truth response simultaneously, the KTWM works effectively, i.e., can correctly incorporate injected knowledge into the generated response.

On the other hand, the negative examples in Table 6 show that the term scoring can be ineffective if there is no good overlap with the ground truth response. We observe in these examples that most of the words with relatively high scores do not exist in both post and response. At the same time, if the injected knowledge does not contain useful terms, the produced responses might be irrelevant. In Table 6, most of terms have light blue colour, indicating that KTWM detected a relatively low importance of these terms correctly.

The examples in these two tables also confirm statistical results shown and discussed in §4.4.3. KTWM term weights still induce noise, especially when the injected knowledge does not contain useful terms (i.e. terms that present in the ground truth response), resulting in a worse response generation performance.

We note that the both sets of examples include highlighted punctuation (e.g., ‘,’) and language structural terms (e.g., ‘the’, ‘is’) which obtain high KTWM weights. We assume that such terms are widely distributed in post and response sets and therefore detected as important.

Post: I love avengers!! Have you watched the Infinity War movie? I think it was really good and I can’t wait till the next movie to find out what happens. **Response:** The Avengers Infinity War was so good. What a cliff hanger. Blows my mind that the Avengers made their comic debut way back in 1963. **Knowledge Terms Weighted by KTWM:** In ” Avengers : Infinity War ” , the Avengers join forces with the guardians of the galaxy to confront Thanos , who is trying to collect the infinity stones **Response generated by KTWM:** I love the Avengers. the Avengers was the Avengers, Infinity War. He is such a great show.
Post: I have never actually been to a horse race in Kentucky, where is the Derby held at anyways? **Response:** Louisville Kentucky, it ends the Kentucky Derby festival usually. **Knowledge Terms Weighted by KTWM:** The Kentucky Derby presented by Woodford reserve is a horse race that is held annually in Louisville , Kentucky , United States , on the first Saturday in may . **Response generated by KTWM:** I’m not sure but I know that it is located in Kentucky race in the United States and Canada.
Post: New York city is quite impressive. **Response:** Yeah they say that New York and LA are the two largest markets in our country. **Knowledge Terms Weighted by KTWM:** The city of New York , often called New York City or simply New York , is the most populous city in the United States **Response generated by KTWM:** Yes, it is. The city of New York is the most populous city in the United States.
Post: Hi there! I drive a Honda Civic. Do you have a car? **Response:** I have a VW golf, but was looking to change to a Honda Civic. I heard that the Civic has got larger throughout the years. **Knowledge Terms Weighted by KTWM:** The Honda Civic type R is the highest performance version of the Honda Civic made by Honda Motor company of Japan **Response generated by KTWM:** I have a Honda Civic. They are the second most commonly used in the US in Japan.
Post: Well that’s interesting, but seems like it could get confusing! I bet humans have been hiking ever since we could walk though! **Response:** Yeah hiking has been around for ever, the term bush walking is endemic to Australia. **Knowledge Terms Weighted by KTWM:** Hiking is the preferred term , in canada and the united states , for a long , vigorous walk , usually on trails (footpaths) , in the countryside **Response generated by KTWM:** Well, hiking is a long term hiking is used in the United States, and the countryside on trails.

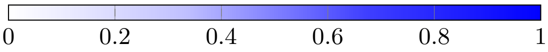

Table 5: Positive cases that the injected knowledge is properly weighted by the proposed KTWM model. The background colour stands for the probability of being a useful term.

Post: Yes, indeed. What cultural meanings come to mind when you think of dance? **Response:** Some of its many functions that come to mind are social, ceremonial, and competitive in meaning. **Knowledge Terms Weighted by KTWM:** One open question regarding the nature of the mind is the mind body problem , which investigates the relation of the mind to the physical brain and nervous system **Response generated by KTWM:** I'm not sure but I do know that the spiritual core is associated with mind and mind.
Post: I have written a few historical fiction novels that i am currently trying to get published. **Response:** That's really cool, historical fiction is such a fun genre to read. **Knowledge Terms Weighted by KTWM:** Historical romance (also historical novel) is a broad category of fiction in which the plot takes place in a setting located in the past **Response generated by KTWM:** That's great. I love romance novels. It's a broad category of historical fiction, and fantasy.
Post: I would have to say Snooki still, she is such a ball of fire. **Response:** Since you like Snooki so much, did you happen to see the Spinoff of Jersey shore called ”Snooki and Jwoww”? **Knowledge Terms Weighted by KTWM:** She attended Columbia high school , in east greenbush , New York , and New York Institute of Technology **Response generated by KTWM:** I don't know much about Vietnamese style, but i know it has been around since 1700
Post: Are they popular in other areas of the world like Europe or Asia ? **Response:** Similarly! In the European Union vehicles in this size are known as large good vehicles. **Knowledge Terms Weighted by KTWM:** Western civilisation traces its roots back to western Europe and the western Mediterranean **Response generated by KTWM:** I'm not sure, but I do know that they are native to the Mediterranean Mediterranean cuisine.
Post: Wow that's interesting. The food world has really diversified. **Response:** Yeah or you can cook without heat. Like in south America they make Ceviche, which is fish that is cooked in lemon or lime juice and the acid cooks the fish. **Knowledge Terms Weighted by KTWM:** Bow Wow Wow are an English 1980s new wave band , created by Malcolm Mclaren **Response generated by KTWM:** Yes, it is! It's the world's most popular foods in the world.

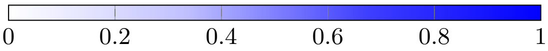

Table 6: Negative cases that the injected knowledge is not properly weighted by the proposed KTWM model. The background colour stands for the probability of being a useful term.

Inspecting the concept knowledge graph encoded by modern language models

Carlos Aspillaga[1], **Marcelo Mendoza**[2], **Alvaro Soto**[1]
[1] Computer Science Department, Pontificia Universidad Católica de Chile
[2] Department of Informatics, Universidad Técnica Federico Santa María, Chile
`cjaspill@uc.cl,mmendoza@inf.utfsm.cl,asoto@ing.puc.cl`

Abstract

The field of natural language understanding has experienced exponential progress in the last few years, with impressive results in several tasks. This success has motivated researchers to study the underlying knowledge encoded by these models. Despite this, attempts to understand their semantic capabilities have not been successful, often leading to non-conclusive, or contradictory conclusions among different works. Via a probing classifier, we extract the underlying knowledge graph of nine of the most influential language models of the last years, including word embeddings, text generators, and context encoders. This probe is based on concept relatedness, grounded on WordNet. Our results reveal that all the models encode this knowledge, but suffer from several inaccuracies. Furthermore, we show that the different architectures and training strategies lead to different model biases. We conduct a systematic evaluation to discover specific factors that explain why some concepts are challenging. We hope our insights will motivate the development of models that capture concepts more precisely.

1 Introduction

Natural language processing (NLP) encompasses a wide variety of applications such as summarization (Kovaleva et al., 2019), information retrieval (Zhan et al., 2020), and machine translation (Tang et al., 2018), among others. Currently, the use of pre-trained language models has become the *de facto* starting point to tackle most of these tasks. The usual pipeline consists of finetuning a pre-trained language model by using a discriminative learning objective to adapt the model to the requirements of each task. As key ingredients, these models are pre-trained using massive amounts of unlabeled data that can include millions of documents and billions of parameters. Massive data and parameters are supplemented with a suitable learning architecture, resulting in a highly powerful but also complex model whose internal operation is hard to analyze.

The success of pre-trained language models has driven the interest to understand the mechanisms they use to solve NLP tasks. As an example, in the case of BERT (Devlin et al., 2019), one of the most popular pre-trained models based on the Transformer (Vaswani et al., 2017), several studies have attempted to access the knowledge encoded in its layers and attention heads (Tenney et al., 2019b; Devlin et al., 2019; Hewitt and Manning, 2019). In particular, Jawahar et al. (2019) shows that BERT can solve tasks at a syntactic level by using Transformer blocks to encode a soft hierarchy of features at different levels of abstraction. Similarly, Hewitt and Manning (2019) show that BERT is capable of encoding structural information from text. In particular, using a structural probe, they show that syntax trees are embedded in a linear transformation of the encodings of BERT.

In general, previous efforts have provided strong evidence indicating that current pre-trained language models encode complex syntactic rules. However, relevant evidence about their abilities to capture semantic information remains still elusive. As an example, Si et al. (2019) attempts to locate the encoding of semantic information as part of the top layers of Transformer architectures finding contradictory evidence. Similarly, Kovaleva et al. (2019) focuses on studying knowledge encoded by self-attention weights. Their results provide evidence for over-parameterization but not about language understanding capabilities.

In this work, we study to what extent pre-trained language models encode semantic information. As a key source of semantic knowledge, we analyze their ability to encode the concept relations embedded in the conceptual taxonomy of Word-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2984–3000
August 1–6, 2021.

Net[1] (Miller, 1995). Understanding, organizing, and correctly using concepts is one of the most remarkable capabilities of human intelligence (Lake et al., 2017). Therefore, quantifying the ability that a pre-trained language model can exhibit to encode the conceptual organization behind WordNet is highly valuable. This knowledge may provide useful insights into the inner mechanisms that these models use to encode semantic information. Furthermore, identifying what they find difficult can provide relevant insights into how to improve them.

Unlike most previous works, we do not focus on a particular model but target a large list of the most popular pre-trained language models. In this sense, one of our goals is to provide a comparative analysis of the benefits of different approaches. Following Hewitt and Manning (2019), we study semantic performance by defining a probing classifier based on concept relatedness according to WordNet. Using this tool, we analyze the different models, enlightening how and where semantic knowledge is encoded. Furthermore, we explore how these models encode suitable information to recreate the structure of WordNet. Among our main results, we show that the different pre-training strategies and architectures lead to different model biases. In particular, we show that contextualized word embeddings, such as BERT, encode high-level concepts and hierarchical relationships among them, creating a taxonomy. This finding corroborates previous work results (Reif et al., 2019) that claim that BERT vectors store sub-spaces that correspond with semantic knowledge. Our study also shows evidence about the limitations of current pre-trained language models, demonstrating that they have difficulties to encode specific concepts. For example, all the models struggle with concepts related to "taxonomical groups". Our results also reveal that models have distinctive patterns regarding where in the architecture they encode the semantic information. These patterns are dependant on architecture and not on model sizes.

2 Study methodology

Probing methods consist of using the representation of a frozen pre-trained model to address a particular task. If the probing classifier succeeds in this setting but fails using an alternative model, it means that the source model encodes the knowledge needed to solve the task. Furthermore, the classifier's performance can be used to measure how well the model captures this knowledge (Conneau et al., 2018). We use a probing method at the semantic level applying it to the nine models presented in Section 2.2. Our study sheds light on whether the models encode relevant knowledge to predict concept relatedness in Wordnet.

To study how accurately the models encode semantic information, we measure correctness in predicted relations among concepts at two levels: (a) pair-wise-level by studying performance across sampled pairs of related or unrelated concepts, and (b) graph-level by using pair-wise predictions to reconstruct the actual graph. We describe both approaches in Sections 2.3 and 2.4, respectively.

2.1 WordNet splits and sampling

We partitioned the available WordNet synsets at 70/15/15 for training, validation and test sets respectively. Our experimental setup ensures no overlap in concepts among these sets. As an example, if the concept related to "house" fell in the training set, then all its lemmas are considered in this partition (e.g. "home", "residence", etc.), and neither this concept nor those lemmas will be present in the validation or test sets. Our sampling setup also balances the number of times each concept acts as hypernym or as a hyponym in the relation, whenever possible. Thus the benefit of learning whether a word is a "prototypical hypernym", as pointed out by Levy et al. (2015), is close to zero. Further details are available in Appendix A.1.

2.2 Word embedding models

This study considers the most influential language models from recent years. We consider the essential approaches of three model families: non contextualized word embeddings (NCE), contextualized word embeddings (CE), and generative language models (GLM). We consider Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) for the first family of approaches. For the CE family, we consider ELMo (Peters et al., 2018b), which is implemented on a bidirectional LSTM architecture, XLNet (Yang et al., 2019), and BERT (Devlin et al., 2019) and its extensions ALBERT (Lan et al., 2020) and RoBERTa (Liu et al., 2019), all of them based on the Transformer architecture. GPT-2 (Radford et al., 2018) and T5 (Raffel et al.,

---

[1] WordNet is a human-generated graph, where each one of its 117000 nodes (also called synsets) represent a concept. In this work, we use hyponymy relations, representing if a concept is a subclass of another.

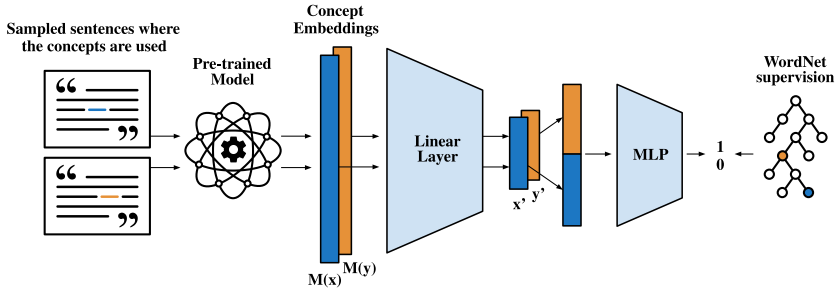

Figure 1: Inputs to the edge probing classifier correspond to the model embeddings $M(x)$ and $M(y)$ of concepts x and y, respectively. $M(x)$ and $M(y)$ are projected into a common lower dimensionality space using a linear layer. The resulting embeddings x' and y' are concatenated and fed into a Multi-Layer Perceptron that is in charge of predicting if the concept pair is related or not.

2019) are included in the study to incorporate approaches based on generative language models.

For models in the CE and GLM families, the embedding is extracted after running the model on a sentence where the concept is used in context. Then we discard the context and keep only the first token that correspond to the specific mention of the concept. Finally we concatenate the hidden states of every layer of the model, for the selected token.

2.3 Semantic probing classifier

We define an edge probing classifier that learns to identify if two concepts are semantically related. To create the probing classifier, we retrieve all the glosses from the Princeton WordNet Gloss Corpus[2]. This dataset provides WordNet's synsets gloss sentences with annotations identifying occurrences of concepts within different sentence contexts. The annotations provide a mapping of the used words to their corresponding WordNet node. We sample hypernym pairs A, B. Then, from an unrelated section of the taxonomy, we randomly sample a third synset C, taking care that C is not related to either A or B. Then, $\langle A, B, C \rangle$ forms a triplet that allows us to create six testing edges for our classifier. To train the probing classifier, we define a labeled edge $\{x, y, L\}$, with x and y synsets in $\{A, B, C\}$, $x \neq y$. $L \in \{0, 1\}$ is the target of the edge. If y is direct or indirect parent of x, $L = 1$, while $L = 0$ in other case. For each synset x, y, we sample one of its sentences $S(x)$, $S(y)$ from the dataset. Let M be a model. If M belongs to the NCE family, x and y are encoded by $M(x)$ and $M(y)$, respectively. If M belongs to the CE or GLM families, then x and y are encoded by the corresponding token of $M(S(x))$ and $M(S(y))$, respectively.

To facilitate the evaluation of embeddings of different sizes, we first project each concept's encodings x and y into a low dimensionality space using a linear layer (see Figure 1). These vectors, denoted as x' and y', are concatenated and fed into a Multi-Layer Perceptron (MLP) classifier. The linear layer and the MLP are the only trainable parameters of our setting, as we use the source model weights without any finetuning. Throughout all the experiments we used an MLP classifier with a single hidden layer of 384 hidden units.

We use this MLP to learn the structural relation between concept pairs, providing the test with a mechanism that allows the embeddings to be combined in a non-linear way. Tests based on linear transformations such as the one proposed by Hewitt and Manning (2019) did not allow us to recover the WordNet structure. This indicates that the sub-spaces where the language models encode semantics are not linear. The fact that syntactic information is linearly available suggests that syntax trees might be a critical intermediate result for the language modeling task. In contrast, semantic information emerges as an indirect consequence of accurate language modeling. Still, it might not constitute information that the model relies on for NLP tasks, as postulated by Ravichander et al. (2020).

To discard the possibility of the MLP being memorizing properties of words and thus giving an undeserved credit to the analyzed models, we generated alternative training and validation sets with random word embeddings of the same size as the real ones. During training and inference, these vectors were kept frozen. These tests showed around 50% accuracy in the binary classification task, indicating that the MLP cannot do better than chance in that scenario. Thus, if in a later experiment the same MLP

---

[2] https://wordnetcode.princeton.edu/glosstag.shtml

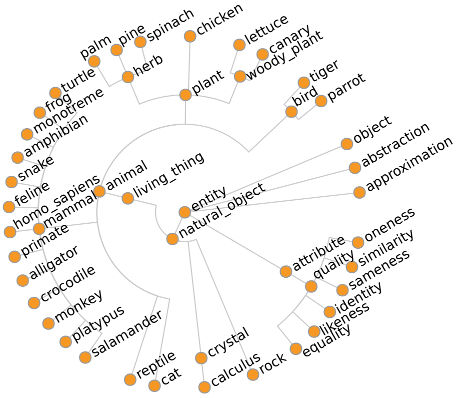

Figure 2: A reconstructed graph using BERT-large. Visual inspection reveals that the models capture key categories but fail to map fine-grained relations.

Family	Model	Tree Edit Dist.		
		TIM	MCM	Avg.
NCE	Word2Vec	59	59	**59**
	GloVe-42B	56	60	**58**
GLM	GPT-2	53	57	**55**
	T5	58	55	**56**
CE	ELMo	52	55	**53**
	BERT	49	48	**49**
	RoBERTa	56	54	**55**
	XLNet	52	48	**50**
	ALBERT	53	50	**51**

Table 1: Tree Edit Distance against the ground truth graph (large models used). We display both strategies for estimating d_e along with their average score.

succeeds at the task, the merit can be attributed to the input embedding itself. This result is consistent with the fact that our experimental setup ensures no overlap in concepts among training, development, and testing sets.

2.4 Reconstructing the structure of a knowledge graph

The probe classifier predicts if a pair of concepts $\langle u, v\rangle$ form a valid $\langle$parent, child$\rangle$ relation according to WordNet, where $h_{\langle u,v\rangle} \in [0, 1]$ denotes the corresponding classifier output. It is important to note that valid $\langle$parent, child$\rangle$ relations include direct relations (e.g. $\langle$dog, poodle$\rangle$), and transitive relations (e.g. $\langle$animal, poodle$\rangle$), and that the order of the items matters.

To reconstruct the underlying knowledge graph, for each valid $\langle$parent, child$\rangle$ relation given by $h_{\langle u,v\rangle} >$ threshold, we need an estimation of how close are the nodes in the graph. We do this by introducing the concept of "parent closeness" between a parent node u and a child node v, denoted by $d_e(u, v)$. We propose two alternative scores to estimate d_e:

i) Model Confidence Metric (MCM): All the models considered in this study capture close relations more precisely than distant relations (supporting evidence can be found in Appendix D). This means that a concept like *poodle* will be matched with its direct parent node *dog* with higher confidence than with a more distant parent node (e.g. *animal*). Thus, we can define $d_e(u, v) = 1 - h_{\langle u,v\rangle}$.

ii) Transitive Intersections Metric (TIM): We explore a metric grounded directly in the tree structure of a knowledge graph. Note that nodes *u* and *v* that form a parent-child relation have some transitive connections in common. Specifically, all descendants of *v* are also descendants of *u*, and all the ancestors of *u* are also ancestors of *v*. Then, the closer the link between *u* and *v* in the graph, the bigger the intersection. Accordingly, for each edge $e = \langle u, v\rangle$, we define $d_e(u, v)$ as:

$$-\left(\sum_{j \in N \setminus \{u,v\}} h_{\langle u,j\rangle} h_{\langle v,j\rangle} + h_{\langle j,u\rangle} h_{\langle j,v\rangle}\right) * h_{\langle u,v\rangle}, \tag{1}$$

where the first term of the sum accounts for the similarity within the descendants of nodes u and v, and the second term accounts for the similarity within the ancestors of nodes u and v. The term $h_{\langle u,v\rangle}$ at the right-hand side accounts for the edge direction, and N denotes the set of nodes (concepts).

A strategy to find a tree that comprises each node's closest parents is the minimum-spanning-arborescence (MSA) of the graph defined using d_e. The MSA is analogous to the minimum-spanning-tree (MST) objective used by Hewitt and Manning (2019), but for directed graphs. The formulation of the MSA optimization problem applied to our proposal is provided in the Appendix A.3.

3 How accurate is this knowledge?

3.1 Semantic edge probing classifier results

Table 2 shows the results obtained using the edge probing classifier. Results show that regardless of model sizes, performance is homogeneous within each family of models. Additionally, results show that NCE and GLM methods obtain a worse performance when all the layers are used than those achieved by CE methods. When single layers are

Family	Model	Emb. Size All/Best Layer	Best Layer	F1-score All Layers	F1-score Best Layer
NCE	Word2Vec (Mikolov et al., 2013)	300 / -	-	.7683 ± .0135	-
	GloVe-42B (Pennington et al., 2014)	300 / -	-	.7877 ± .0084	-
GLM	GPT-2 (Radford et al., 2018)	9984 / 768	6	.7862 ± .0132	.7921 ± .0108
	T5-small (Raffel et al., 2019)	7168 / 512	4	.8156 ± .0098	.8199 ± .0081
	GPT2-xl (Radford et al., 2018)	78400 / 1600	13	.7946 ± .0151	.8029 ± .0118
	T5-large (Raffel et al., 2019)	51200 / 1024	17	.8148 ± .0119	.8331 ± .0102
CE	ELMo-small (Peters et al., 2018b)	768 / 256	2	.7986 ± .0126	.7880 ± .0119
	BERT-base (Devlin et al., 2019)	9984 / 768	10	.8240 ± .0123	.8185 ± .0104
	RoBERTa-base (Liu et al., 2019)	9984 / 768	5	.8392 ± .0100	.8266 ± .0083
	XLNet-base (Yang et al., 2019)	9984 / 768	4	.8306 ± .0113	.8293 ± .0116
	ALBERT-base (Lan et al., 2020)	9984 / 768	12	.8184 ± .0222	.8073 ± .0102
	ELMo-large (Peters et al., 2018b)	3072 / 1024	2	.8311 ± .0090	.8330 ± .0083
	BERT-large (Devlin et al., 2019)	25600 / 1024	14	.8178 ± .0152	.8185 ± .0113
	RoBERTa-large (Liu et al., 2019)	25600 / 1024	13	.8219 ± .0159	.8314 ± .0082
	XLNet-large (Yang et al., 2019)	25600 / 1024	6	.8211 ± .0142	.8244 ± .0080
	ALBERT-xxlarge (Lan et al., 2020)	53248 / 4096	4	.8233 ± .0107	.8194 ± .0097

Table 2: Results obtained using the edge probing classifier. We study the performance in many model variants, considering small and large versions of several models. Results are grouped by method families.

used, GLM shows improved performance, suggesting that these models capture semantics earlier in the architecture, keeping their last layers for generative-specific purposes. In contrast, CE models degrade or maintain their performance when single layers are used.

Note that Table 2 shows pair-wise metrics not graph metrics. As we are dealing with graphs, predicted edges are built upon related edges. Thus, drifts in small regions of the graph may cause large drifts in downstream connections. Furthermore, our setup balances positive and negative samples. However, the proportion of negative samples can be considerably larger in a real reconstruction scenario. As a consequence, we emphasize that these numbers must be considered together with the results reported in sections 3.2 and 4.

3.2 Extracting the Knowledge Graph

Predicting a knowledge graph has a complexity of at least $O(N^2)$ in the number of analyzed concepts. In our case, this imposes a highly demanding computational obstacle because WordNet has over 82000 noun synsets. To accelerate experimentation and facilitate our analysis and visualizations, we focus on extracting a WordNet sub-graph comprising 46 nodes not seen during training or validation. These nodes are picked to include easily recognizable relations. We use the tree-edit-distance to evaluate how close are the reconstructed graphs to the target graph extracted from WordNet. Table 1 shows our results.

Table 1 shows that graphs retrieved using CE models are closer to the target than graphs provided by NCE and GLM models. In particular, the best results are achieved by BERT, ALBERT, and XLNet, indicating that these models encode more accurate semantic information than the alternative models. These results are consistent with those obtained in Section 3.1. The graphs for all the models can be found in Appendix C.

4 What is easy or hard? What are these models learning?

Section 3 shows that different model families differ in their errors. Furthermore, it shows that within the same family, models have similar biases. In this section, we elucidate which semantic factors impact the performance of these models and which ones do not affect their F1-score.

Figure 3-a shows that most models decrease their F1-score as concepts get more specific. We hypothesize that higher-level concepts (e.g., *Animal*) appear more frequently and in more diverse contexts, as they are also seen as instances of their sub-classes (e.g., *Dog*, *Cat*, *Chihuahua*), allowing the models to learn more precise representations for them. In contrast, lower-level concepts will only appear in specific contexts (e.g., texts about *Apple-Head-Chihuahua*). Figure 3-b corroborates

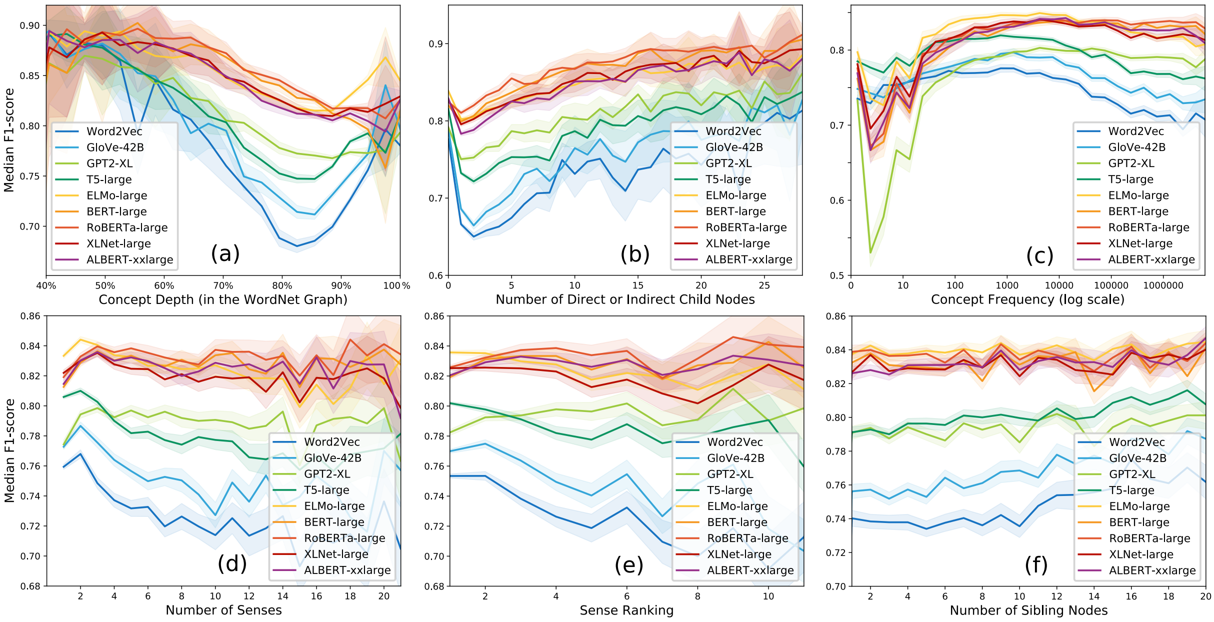

Figure 3: Semantic factors with a high (top charts) or low (bottom charts) impact on F1-score, along with their 90% confidence intervals. Charts only display ranges where at least 100 samples existed. Appendix D shows additional factors along with the specific implementation details.

this intuition, as concepts with a higher number of sub-classes have higher F1-scores. Figure 3-c shows that models degrade their F1-score when concepts are too frequent. In particular, NCE and GLM models are more sensitive to this factor.

Another finding is that CE and GLM models are almost unaffected by the number of senses that a certain word has, neither to their sense ranking or their number of sibling concepts, displaying almost flat charts (see Figures 3-d-e-f). This result suggests that these models pay more attention to the context than to the target word. This behavior is opposed to what NCE models exhibit according to Yaghoobzadeh et al. (2019), as NCE models tend to focus more on frequent senses.

In most cases, the same family models have similar behaviors, especially within the NCE or CE families. Also, different families show different patterns. Table 3 shows some salient examples. Surprisingly, all models struggle in the category “taxonomic groups”. Manual inspection of sentences makes us believe that the context confuses CE and GLM models in these cases. In many sentences, the corresponding concept could be nicely replaced by another, conveying a modified but still valid message. This phenomenon does not occur in other categories such as “social group” or “attribute”, even though these concepts are closely related to “taxonomic groups”.

5 Where is this knowledge located?

As mentioned in Section 7, prior work has not shown consensus about where is semantic information encoded inside these architectures. Our experiments shed light on this subject. Figure 4 shows how each layer contributes to the F1-score.

Figures 4-a and 4-b show the performance across layers for the CE-based models. They reveal that while BERT and RoBERTa use their top-layers to encode semantic information, XLNet and ALBERT use the first layers. Figure 4-c shows that while GPT-2 uses all its layers to encode semantics, T5 shows an M shape related to its encoder-decoder architecture. The chart shows that T5 uses its encoder to hold most of the semantic information. We also note that small models show similar patterns as their larger counterparts.

6 Further discussion and implications

Table 4 summarizes our main findings. Findings (1), (2), and (3) indicate that, to a different extent, all models encode relevant knowledge about the hierarchical semantic relations included in WordNet. However, as we mention in Section 4, we observe that the ability to learn about a concept depends on its frequency in the training corpus and the specificity of its meaning. Furthermore, some concept categories seem to be hard for every model family, while some are particularly difficult

Family	Model	artifact	attribute	living thing	matter	person	relation	part	social group	taxonomic group
NCE	Word2Vec	.7120	.7044	.7295	.7402	.7208	.7264	.7532	.7497	.6920
	GloVe-42B	.7389	.7213	.7421	.7633	.7351	.7567	.7759	.7579	.6648
GLM	GPT-2	.7903	.7730	.7300	.7582	.7207	.7612	.7540	.8155	.3030
	T5	.7868	.7649	.7862	.8002	.7735	.7963	.8051	.7868	.6944
CE	ELMo	.8308	.8093	.8187	.7756	.8022	.7679	.7580	.8312	.6011
	BERT	.8249	.8094	.7593	.7645	.7379	.7662	.7499	.8516	.4804
	RoBERTa	.8315	.8167	.7823	.7614	.7585	.7649	.7441	.8552	.4921
	XLNet	.8319	.8064	.7907	.7636	.7779	.7659	.7526	.8422	.5371
	ALBERT	.8231	.8050	.7758	.7685	.7826	.7727	.7610	.8556	.4277

Table 3: Average F1-score for some semantic categories revealing models strengths and weaknesses. Several other categories are reported in Appendix E along with their standard deviations.

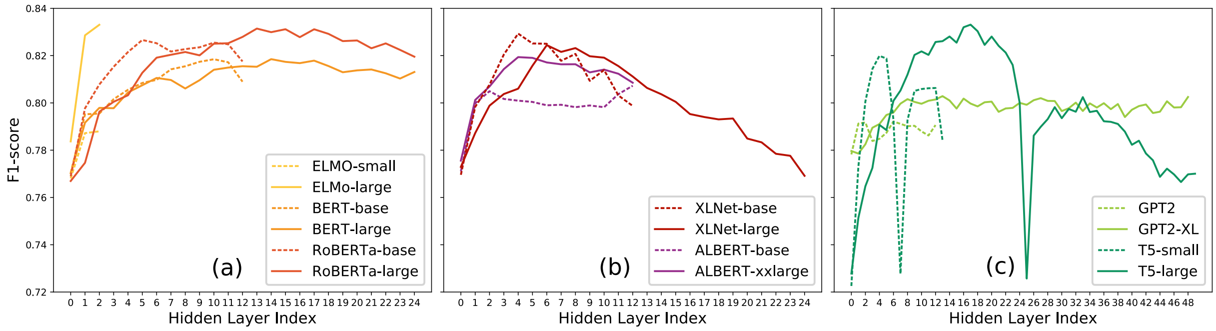

Figure 4: F1-score for hypernym prediction across each model layer.

for contextual models such as CE. We hypothesize that stronger inductive biases are required to capture low-frequency concepts. Furthermore, we believe that new learning approaches are needed to discriminate accurate meaning for high-frequency concepts. As expected, our findings indicate that model families have different biases leading to different behaviors. Thus, our results can illuminate further research to improve semantic capabilities by combining each family of models' strengths. For example, one could combine them as ensembles, each one equipped with a different loss function (i.e., one generative approach resembling GLM-based methods and another discriminative resembling CE-based methods).

Findings (4), (5), and (6) suggest that instead of a standard finetuning of all layers of BERT according to a given downstream task, to improve semantic capabilities, one could perform a task profiling to decide the best architecture for the task and also how to take advantage of it. Using only a limited number of layers or choosing a different learning rate for each layer, one could exploit the semantic knowledge that the pre-trained model carries, avoiding the degradation of this information present at the top layers, especially when using T5, XLNet, or ALBERT-large. Accordingly, recent work on adaptive strategies to output predictions using a limited number of layers (Xin et al., 2020; Liu et al., 2020; Hou et al., 2020; Schwartz et al., 2020; Fan et al., 2020; Bapna et al., 2020) would benefit from using architectures that encode knowledge in the first layers. To the best of our knowledge, these works have only used BERT and RoBERTa, achieving a good trade-off between accuracy and efficiency. Only Zhou et al. (2020) has explored ALBERT, reporting improved accuracy by stopping earlier. Our findings explain this behavior and suggest that T5 or XLNet may boot their results even further as these architectures have sharper and higher information peaks in their first layers.

Findings (7) and (8) suggest that recent success in semantic NLP tasks might be due more to the use of larger models than large corpora for pretraining. This also suggests that to improve model performance in semantic tasks, one could train larger models even without increasing the corpus size. A similar claim has been proposed by (Li et al., 2020) leading to empirical performance improvements.

Finally, finding (9) is important because it suggests that contextual models pay as much attention to the context as to the target word and are probably

	Finding	Supporting Evidence	Involved Models
(1)	All models encode a relevant amount of knowledge about semantic relations in WordNet, but this knowledge contains imprecisions.	All	All
(2)	The ability to learn concept relations depends on how frequent and specific the concepts are. Some model families are more affected.	Fig. 3a-c	NCE and GLM
(3)	Concept difficulty is usually homogeneous within each model family. Some semantic categories challenge all models.	Table 3	All
(4)	Some models encode stronger semantic knowledge than others, usually according to their family.	Tables 2, 1, 3	ELMo, BERT, RoBERTa, ALBERT, XLNet, T5
(5)	Some models focus their encoding of semantic knowledge in specific layers, and not distributed across all layers.	Table 2, Fig. 4	GLM
(6)	Models have distinctive patterns as to where they encode semantic knowledge. Patterns are model-specific and not size-specific.	Table 2, Fig. 4	All
(7)	Model size has an impact in the quality of the captured semantic knowledge, as seen in our layer-level probe tests.	Table 2, Fig. 4	ELMo, RoBERTa, ALBERT, GPT-2, T5
(8)	Semantic knowledge does not depend on pre-training corpus size.	Tables 2, B-5	-
(9)	Contextual models are unaffected by multi-sense words.	Fig. 3d-f	CE and GLM

Table 4: Summary of our main findings and their corresponding supporting evidence.

biased in favor of contextual information, even if they are not based on the Masked-Language-Model strategy. We believe that this inductive bias could be exploited even further in the design of the underlying architecture. Thus this finding might elucidate a design direction to encourage more effective learning of semantic knowledge.

7 Related work

The success of deep learning architectures in various NLP tasks has fueled a growing interest to improve understanding of what these models encode. Studies like Tenney et al. (2019b) claim that success in a specific task helps understand what type of information the model encodes.

Evidence of syntactic information: Using probing classifiers, Clark et al. (2019) claims that some specific BERT's attention heads show correspondence with syntactic tasks. Goldberg (2019) illustrates the capabilities that BERT has to solve syntactic tasks, such as subject-verb agreement. Hewitt and Manning (2019) proposes a structural probe that evaluates whether syntax trees are encoded in a linear transformation of BERT embeddings. The study provides evidence that syntax trees are implicitly embedded in BERT's vector geometry. Reif et al. (2019) has found evidence of syntactic representation in BERT's attention matrices, with specific directions in space representing particular dependency relations.

Evidence of semantic information: Reif et al. (2019) suggests that BERT's internal geometry may be broken into multiple linear subspaces, with separate spaces for different syntactic and semantic information. Despite this result, previous work has not yet reached a consensus about this topic. While some studies show satisfactory results in tasks such as entity types (Tenney et al., 2019a), semantic roles (Rogers et al., 2020), and sentence completion (Ettinger, 2020), other studies show less favorable results in coreference (Tenney et al., 2019b), Multiple-Choice Reading Comprehension (Si et al., 2019) and Lexical Relation Inference (Levy et al., 2015), claiming that BERT's performance may not reflect the model's true ability of language understanding and reasoning. Tenney et al. (2019b) proposes a set of edge probing tasks to test the encoded sentential structure of contextualized word embeddings. The study shows evidence that the improvements that BERT and GPT-2 offer over non contextualized embeddings as GloVe is only significant in syntactic-level tasks. Regarding static word embeddings, Yaghoobzadeh et al. (2019) shows that senses are well represented in single-vector embeddings if they are frequent and that this does not harm NLP tasks whose performance depends on frequent senses.

Layer-wise or head-wise information: Tenney et al. (2019a) shows that the first layers of BERT focus on encoding short dependency relationships at the syntactic level (e.g., subject-verb agreement). In contrast, top layers focus on encoding long-range dependencies (e.g., subject-object dependencies). Peters et al. (2018a) supports similar declarations for Convolutional, LSTM, and self-attention architectures. While these studies also support that the top layers appear to encode semantic information, the evidence to support this claim is not conclusive or contradictory with other works. For example, Jawahar et al. (2019) could only identify one SentEval semantic task that topped at the last layer. In terms of information flow, Voita et al. (2019a) reports that information about the past in left-to-right language models gets vanished as the information flows from bottom to top BERT's layers. Hao et al. (2019) shows that the lower layers of BERT change less during finetuning, suggesting that layers close to inputs learn more transferable language representations. Press et al. (2020) shows that increasing self-attention at the bottom layers improves language modeling performance based on BERT. Other studies focus on understanding how self-attention heads contribute to solving specific tasks (Vig, 2019). Kovaleva et al. (2019) shows a set of attention patterns repeated across different heads when trying to solve GLUE tasks (Wang et al., 2018). Furthermore, Michel et al. (2019) and Voita et al. (2019b) show that several heads can be removed without harming downstream tasks.

Automated extraction of concept relations: Although the main focus of our work is not to master the probing task of extracting knowledge from WordNet, but to use it as an instrument to verify and compare the abilities of current families of language models to encode this kind of knowledge, for completitude we include a brief mention of previous literature regarding this subject. Relation extraction is an active research topic. Early works are either feature-based, usually relying on SVMs, Maximum Entropy, or on a set of manually defined rules (Hearst, 1998; Kambhatla, 2004; Dashtipour et al., 2017; Minard et al., 2011; Weeds et al., 2014; Chen et al., 2015). Other methods rely on manually defined distance metrics to estimate the relatedness of two semantic instances (Dandan et al., 2012; Panyam et al., 2016). Following works have used different types of neural networks or LSTM modules for this task (Liu et al., 2013; Zeng et al., 2014, 2015; Zhang and Wang, 2015; Song et al., 2018), or attention-based and transformer-based mechanisms with outstanding results (Zhou et al., 2016; Baldini Soares et al., 2019; Huang et al., 2020; Qin et al., 2021; Zhong and Chen, 2021).

Alternative approaches: Several alternative approaches have been used in previous works. Some are dataset-focused (Miller et al., 1994; Levy et al., 2015; Wang et al., 2018; Wiedemann et al., 2019), usually relying on annotated corpora that challenge semantic abilities. These approaches have provided useful insights, but usually suffer from low availability of data as they usually cover a small fraction of the WordNet ontology. As an example, BLESS (Baroni and Lenci, 2011) includes gold-standard annotations for only 200 concepts. Other approaches have tested semantic ability by using prompt-engineering and inspecting the predictions of the models (Petroni et al., 2019; Ettinger, 2020; Talmor et al., 2020), but other works have also shown a high variability in the results depending on the prompt design (Balasubramanian et al., 2020; Reynolds and McDonell, 2021; Zhao et al., 2021).

8 Conclusions

In this work, we exploit the semantic conceptual taxonomy behind WordNet to test the ability of current families of pre-trained language models to learn semantic knowledge from massive sources of unlabeled data. Our main conclusion is that, indeed, to a significant extent, these models learn relevant knowledge about the organization of concepts in WordNet, but also contain several imprecisions. We also notice that different families of models present dissimilar behavior, suggesting the encoding of different biases.

We hope our study helps to inspire new ideas to improve the semantic learning abilities of current pre-trained language models.

Acknowledgments

We thank the anonymous reviewers, whose helpful comments led to increased clarity and improved analysis. We would also like to thank all the members of the IALab research group for their helpful feedback and suggestions. We acknowledge Alex Medina for his help in the early stages of this project. This work was partially funded by the Millennium Institute for Foundational Research on Data and the Fondecyt project number 1200211.

References

Sriram Balasubramanian, Naman Jain, Gaurav Jindal, Abhijeet Awasthi, and Sunita Sarawagi. 2020. What's in a name? are BERT named entity representations just as good for any other name? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 205–214, Online. Association for Computational Linguistics.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2020. Controlling computation versus quality for neural sequence models. *CoRR*, abs/2002.07106.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK. Association for Computational Linguistics.

Yanping Chen, Qinghua Zheng, and Ping Chen. 2015. Feature assembly method for extracting relations in chinese. *Artificial Intelligence*, 228:179–194.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? An analysis of BERT's attention. *CoRR*, abs/1906.04341.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Liu Dandan, Hu Yanan, and Qian Longhua. 2012. Exploiting lexical semantic resource for tree kernel-based chinese relation extraction. In *Natural Language Processing and Chinese Computing. NLPCC 2012. Communications in Computer and Information Science, vol 333. Springer, Berlin, Heidelberg.*

Kia Dashtipour, Mandar Gogate, Ahsan Adeel, Abdulrahman Algarafi, Newton Howard, and Amir Hussain. 2017. Persian named entity recognition. In *2017 IEEE 16th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pages 79–83.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the. Association of Computational Linguistics, TACL*, 8:34–48.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 4141–4150.

Marti A. Hearst. 1998. Automated discovery of wordnet relations. In *WordNet: An Electronic Lexical Database, Christiane Fellbaum (ed.), MIT Press.*

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4129–4138.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. In *Advances in Neural Information Processing Systems*.

Wenti Huang, Yiyu Mao, Zhan Yang, Lei Zhu, and Jun Long. 2020. Relation classification via knowledge graph enhanced transformer encoder. *Knowledge-Based Systems*, 206:106321.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 3651–3657.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181, Barcelona, Spain. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 4365–4374.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020*.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado. Association for Computational Linguistics.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, K. Keutzer, D. Klein, and J. Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *ArXiv*, abs/2002.11794.

Chunyang Liu, Wenbo Sun, Wenhan Chao, and Wanxiang Che. 2013. Convolution neural network for relation extraction. In *Part II of the Proceedings of the 9th International Conference on Advanced Data Mining and Applications - Volume 8347*, ADMA 2013, page 231–242, Berlin, Heidelberg. Springer-Verlag.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 14014–14024.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26, NIPS 2013*, pages 3111–3119.

George A. Miller. 1995. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Anne-Lyse Minard, Anne-Laure Ligozat, and Brigitte Grau. 2011. Multi-class SVM for relation extraction from clinical reports. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 604–609, Hissar, Bulgaria. Association for Computational Linguistics.

Nagesh C. Panyam, Karin Verspoor, Trevor Cohn, and Rao Kotagiri. 2016. ASM kernel: Graph kernel using approximate subgraph matching for relation extraction. In *Proceedings of the Australasian Language Technology Association Workshop 2016*, pages 65–73, Melbourne, Australia.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1532–1543.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018a. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 2227–2237.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Ofir Press, Noah A. Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 2996–3005.

Yongbin Qin, Weizhe Yang, Kai Wang, Ruizhang Huang, Feng Tian, Shaolin Ao, and Yanping Chen. 2021. Entity relation extraction based on entity indicators. *Symmetry*, 13(4).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Abhilasha Ravichander, Yonatan Belinkov, and E. Hovy. 2020. Probing the probing paradigm: Does probing accuracy entail task relevance? *ArXiv*, abs/2005.00719.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B. Viégas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of BERT. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 8592–8600.

Laria Reynolds and Kyle McDonell. 2021. *Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm*. Association for Computing Machinery, New York, NY, USA.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. CoRR abs/2002.12327.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proc. of ACL*.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. 2019. What does BERT learn from multiple-choice reading comprehension datasets? *CoRR*, abs/1910.12391.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state LSTM. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2226–2235, Brussels, Belgium. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2018. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation, WMT 2018*, pages 26–35.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 4593–4601.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems NIPS 2017*, pages 5998–6008.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 37–42.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019a. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 4395–4405.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019b. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 5797–5808.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018*, pages 353–355.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? interpretable word sense disambiguation with contextualized embeddings.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Yadollah Yaghoobzadeh, Katharina Kann, Timothy J. Hazen, Eneko Agirre, and Hinrich Schütze. 2019. Probing for semantic classes: Diagnosing the meaning content of word embeddings. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5740–5753. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 5754–5764.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. An analysis of BERT in document ranking. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, pages 1941–1944.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit.

A Implementation details

A.1 Edge probing classifier details

To study the extent to which these Language Models deal with semantic knowledge, we extend the methodology introduced by Tenney et al. (2019b). In that study, the authors defined a probing classifier at the sentence level, training a supervised classifier with a task-specific label. The probing classifier's motivation consists of verifying when the sentence's encoding help to solve a specific task, quantifying these results for different word embeddings models. We cast this methodology to deal with semantic knowledge extracted from WordNet. Rather than working at the sentence level, we define an edge probing classifier that learns to identify if two concepts are semantically related.

To create the probing classifier, we retrieve all the glosses from the Princeton WordNet Gloss Corpus. The dataset provides WordNet's synsets gloss with manually matched words identifying the context-appropriate sense.

As a reference of size, the selected annotations in the corpus accounted for 41502 lemmas, corresponding to 34371 WordNet synsets. This resulted in 230215 valid WordNet relations.

In WordNet, each sense is coded as one of the synsets related to the concept (e.g., sense *tendency.n.03* for the word tendency). Using a synset A and its specific sense provided by the tagged gloss, we retrieve from WordNet one of its direct or indirect hypernyms, denoted as B (see Figure 5). If WordNet defines two or more hypernyms for A, we choose one of them at random. We sample a third synset C, at random from an unrelated section of the taxonomy, taking care that C is not related to either A or B (e.g., *animal.n.01*). Then, $\langle A, B, C\rangle$ form a triplet that allows us to create six testing edges for our classifier: $\langle A, B\rangle$, which is compounded by a pair of related words through the semantic relation *hypernym of*, and five pairs of unrelated words ($\langle A, C\rangle$, $\langle B, C\rangle$, $\langle B, A\rangle$, $\langle C, A\rangle$, $\langle C, B\rangle$). We associate a label to each of these pairs that show whether the pair is related or not (see Figure 5). Note that we define directed edges, meaning that the pair $\langle A, B\rangle$ is related, but $\langle B, A\rangle$ is unrelated to the relationship *hypernym of*. Accordingly, the edge probing classifier will need to identify the pair's components and the order in which the concepts were declared in the pair.

We create training and testing partitions ensuring that each partition has the same proportion of leaves versus internal nodes. The latter is essential to identify related pairs. During training, we guarantee that each training synset is seen at least once by the probing classifier. To guarantee the above, we sample each synset in the training set and sample some of its hypernyms at random. Then. we randomly sample some unrelated synset for each related pair that has no relation to any of the words in the related pair. We create three partitions from this data on 70/15/15 for training, development, and testing foldings, respectively.

We train the MLP classifier using a weighted binary cross-entropy loss function. Since we have one positive and five negative examples per triplet, we use a weighted loss function with weights 5 and 1 for the positive and negative class, respectively. Accordingly, positive and negative examples have the same relevance during training. We implemented the linear layer and the MLP classifier using a feed forward network with 384 hidden units. The MLP was trained using dropout at 0.425 and a L_2 regularizer to avoid overfitting.

To create the vector representations for each of the word embeddings models considered in this study, we concatenate the hidden state vectors of all the layers for each tagged synset. For both CE and GLM-based models, each gloss was used as a context to build specific contextual word embeddings. If the gloss has more than one tagged token, we take only the first of them for the analysis.

A.2 WordNet metrics: distance

Lets say that we name *"Case-1"* if y is ancestor of x, and *"Case-2"* otherwise. Let $d_W(x, y)$ be the Wordnet distance between two synsets x, y, defined by:

$$d_W(x,y) = \begin{cases} d_{\text{path}}(x,y) & \textit{Case-1}, \\ d_{\text{path}}(x,z) + d_{\text{path}}(y,z) & \textit{Case-2}, \end{cases} \tag{2}$$

where $d_{\text{path}}(x, y)$ is the length of the shortest path between x and y in WordNet, measured in number of hops, and z is the closest common ancestor of x and y in the case that y is not an ancestor of x.

A.3 Minimum-Spanning-Arborescence optimization problem

Given a graph G with nodes N and unknown edges E, we define an auxiliary graph G' with nodes N and edges E', comprised of all possible directed edges. For each edge $e \in E'$, we obtain a prediction h_e that estimates the probability of that

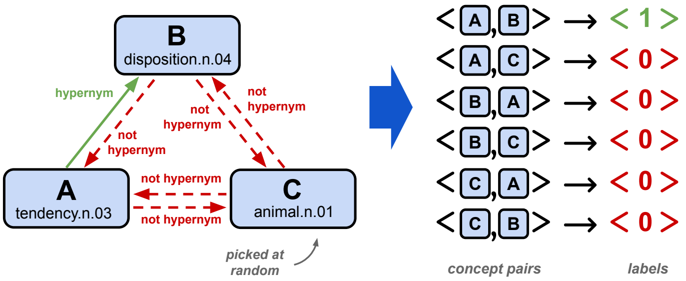

Figure 5: Each triplet is used to create related and unrelated pairs of words according to the relationship *hypernym of*. We create six edge probing pairs, and therefore, the edge probing classifier will need to identify the pair's components and the order in which the words were declared in the pair.

edge representing a valid hypernymy relation, and a distance d_e that estimates the "parent closeness"[3] between the nodes in G.

We define $\delta(v)$ to be the set of edges $\{\langle u,v\rangle : u \in N, u \neq v\}$ where edge $\langle u,v\rangle$ represents a $\langle$parent, child$\rangle$ relation. We also define $\gamma(S)$ to be the set of edges $\{\langle u,v\rangle \in E' : u \notin S, v \in S\}$. We estimate the graph topology of G defined by $E \subset E'$ by solving the following optimization problem:

$$\max_{r\in N} \sum_{e\in E'} x_e h_e \quad \text{s.t.} \quad x_e \in X^* \tag{3}$$

$$X^* = \arg\min \sum_{e\in E'} x_e d_e \tag{4}$$

$$\text{s.t.} \begin{cases} x_e \in \{0,1\} & e \in E' \\ \sum_{e\in\delta(v)} x_e = 1 & \forall v \in N \setminus \{r\} \\ \sum_{e\in\gamma(S)} x_e \geq 1 & \forall S \subset N \setminus \{r\} \end{cases} \tag{5}$$

Objective function (3) is used to find the best root node r; and the nested optimization problem (5) is the minimum spanning arborescence problem applied to the dense graph G'. The final binary values of x_e estimate E by indicating if every possible edge e exist in the graph or not. To solve this optimization problem, we need estimates of h_e and d_e for each edge e. We use the output of the probing classifier as an estimate of the probability of h_e, and use TIM and MCM scores as estimates for d_e (See Section 2.4).

B Pre-Training corpus comparison

Family	Model	Corpus Size	
		Tokens	Size
NCE	Word2Vec	33B	150GB*
	GloVe-42B	42B	175GB*
GLM	GPT-2	10B*	40GB
	T5	180B*	750GB
CE	ELMo	0.8B	4GB*
	BERT	3.9B	16GB
	RoBERTa	38.7B*	160GB
	XLNet	32.9B	140GB*
	ALBERT	3.9B	16GB

Table 5: Pre-Training corpus sizes used for each one of the studied models. The official sources report corpus sizes in terms of number of tokens or uncompressed size in GB. The symbol * denotes values estimated by us based on official available information. Sizes represents uncompressed corpus sizes.

C Additional Reconstructed Graphs[4]

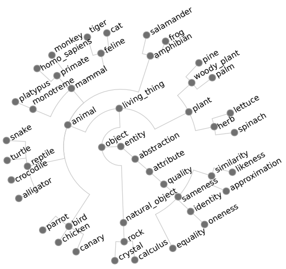

Figure 6: Ground Truth Knowledge Graph

---

[3] The value of this distance will be small if the hypernym relation is close, or large if it is distant or not valid.

[4] Due to space restrictions, the graphs corresponding to Word2Vec, ELMo, T5, BERT will only be included in an extended version of this paper, uploaded to ArXiv

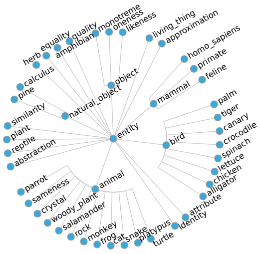

Figure 7: GloVe-42B reconstruction using TIM

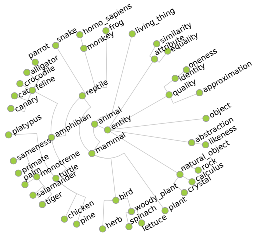

Figure 8: GPT-2-XL reconstruction using TIM

Figure 9: RoBERTa-large reconstruction using TIM

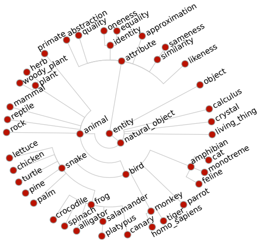

Figure 10: XLNet-large reconstruction using TIM

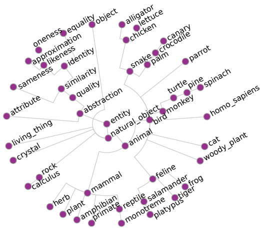

Figure 11: ALBERT-large reconstruction using TIM

D Further information about the impact of semantic factors.[5]

Relative depth in the WordNet graph: (Figure 3-a). For each synset, we compared F1 with depth score (0 % for the root and 100 % for leaves) measuring differences between higher/lower level concepts.

Concept frequency: In Figure 3-c we evaluate if frequent concepts are easier or harder to capture for these models. The frequency was computed by counting occurrences in the 38 GB of OpenWebText Corpus (`http://Skylion007.github.io/OpenWebTextCorpus`).

Number of Senses and Sense Ranking: (Figure 3-d-e) We studied if models are impacted by multi-sense concepts such as "period", and by their sense ranking (how frequent or rare those senses are). Surprisingly contextualized models, and specially CE models have no significant impact by this factor, suggesting that these models are very effective at deducing the correct sense based on their context. These charts also suggest that these models may be considering context even more than the words themselves. This is intuitive for Masked-Language-Models such as BERT, but not for others, such as GPT-2. Non-contextualized models are impacted by this factor, as expected.

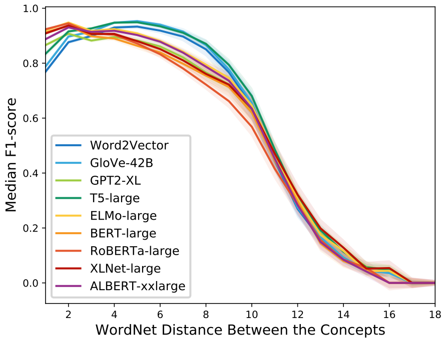

Figure 12: **Graph distance between concepts**: We measured the impact of the number of "hops" that separate two tested concepts on pair-wise F1 score. This chart reveals a strong correlation of all the models in this aspect. As an example of this phenomenon, closer relations such as ⟨chihuahua, dog⟩ are, in general, considerably easier to capture than distant relations such as ⟨chihuahua, entity⟩. For details on how we implement the distance in WordNet, check Appendix A.2.

---

[5] Due to space restrictions, other factors and graphs will only be included in an extended version of this paper, uploaded to ArXiv

E F1-scores of additional categories

Category	W2V	GloVe	GPT-2	T5	ELMo	BERT	RoBERTa	XLNet	ALBERT
abstraction	.7142 ± .1277	.7296 ± .1194	.7224 ± .1897	.7662 ± .0942	.7808 ± .1203	.7718 ± .1582	.7759 ± .1537	.7712 ± .1404	.7635 ± .1732
attribute	.7044 ± .1310	.7213 ± .1237	.7730 ± .0911	.7649 ± .0863	.8093 ± .0886	.8094 ± .0998	.8167 ± .0926	.8064 ± .0891	.8050 ± .0998
communication	.6974 ± .1330	.7251 ± .1224	.7826 ± .0967	.7587 ± .1083	.8049 ± .0925	.8246 ± .0979	.8249 ± .0983	.8066 ± .0987	.8093 ± .1023
group	.7068 ± .1320	.6929 ± .1339	.4972 ± .2955	.7262 ± .1179	.6821 ± .1711	.6173 ± .2399	.6256 ± .2305	.6491 ± .2139	.5858 ± .2745
social group	.7497 ± .1058	.7579 ± .1046	.8155 ± .0883	.7868 ± .0867	.8312 ± .0707	.8516 ± .0742	.8552 ± .0698	.8422 ± .0724	.8556 ± .0819
taxonomic group	.6920 ± .1306	.6648 ± .1330	.3030 ± .2025	.6944 ± .1208	.6011 ± .1583	.4804 ± .2025	.4921 ± .1903	.5371 ± .1944	.4277 ± .2305
family	.7412 ± .1363	.7213 ± .1244	.3131 ± .2003	.6691 ± .1276	.5461 ± .1630	.5626 ± .1537	.5379 ± .1502	.5733 ± .1626	.5437 ± .1705
genus	.6267 ± .0989	.6040 ± .1127	.2567 ± .1582	.7156 ± .1001	.6167 ± .1301	.3696 ± .1857	.4201 ± .1855	.4555 ± .1853	.2862 ± .1945
psychological feature	.7256 ± .1122	.7478 ± .1016	.7829 ± .0904	.7795 ± .0778	.8163 ± .0851	.8181 ± .0954	.8229 ± .0930	.8077 ± .0931	.8208 ± .0915
relation	.7264 ± .1304	.7567 ± .1042	.7612 ± .0809	.7963 ± .0688	.7679 ± .0878	.7662 ± .0995	.7649 ± .0982	.7659 ± .0908	.7727 ± .0929
artifact	.7120 ± .1194	.7389 ± .1068	.7903 ± .0736	.7868 ± .0676	.8308 ± .0693	.8249 ± .0742	.8315 ± .0700	.8319 ± .0702	.8231 ± .0761
covering	.7230 ± .1097	.7510 ± .0970	.7903 ± .0713	.7878 ± .0606	.8398 ± .0599	.8392 ± .0706	.8363 ± .0571	.8393 ± .0576	.8322 ± .0621
instrumentality	.7064 ± .1219	.7378 ± .1052	.7930 ± .0728	.7902 ± .0648	.8308 ± .0676	.8134 ± .0748	.8337 ± .0691	.8313 ± .0711	.8233 ± .0763
device	.7097 ± .1200	.7435 ± .1009	.7956 ± .0713	.7899 ± .0667	.8311 ± .0689	.8198 ± .0701	.8358 ± .0640	.8326 ± .0675	.8230 ± .0743
causal agent	.7240 ± .1137	.7398 ± .1101	.7253 ± .1105	.7751 ± .0757	.8022 ± .0884	.7453 ± .1093	.7631 ± .1056	.7788 ± .1035	.7826 ± .0934
person	.7208 ± .1135	.7351 ± .1113	.7207 ± .1132	.7735 ± .0746	.8022 ± .0854	.7379 ± .1101	.7585 ± .1054	.7779 ± .1053	.7826 ± .0937
living thing	.7295 ± .1112	.7421 ± .1078	.7300 ± .1004	.7862 ± .0749	.8187 ± .0850	.7593 ± .0997	.7823 ± .0922	.7907 ± .0928	.7758 ± .0909
animal	.7349 ± .1049	.7391 ± .1028	.7515 ± .0829	.7837 ± .0714	.8389 ± .0785	.7914 ± .0801	.8179 ± .0701	.8135 ± .0737	.7781 ± .0857
plant	.7445 ± .1044	.7608 ± .0989	.7288 ± .0842	.8168 ± .0653	.8404 ± .0692	.7679 ± .0830	.7962 ± .0688	.7986 ± .0746	.7645 ± .0861
matter	.7402 ± .1117	.7633 ± .1009	.7582 ± .0834	.8002 ± .0662	.7756 ± .0886	.7645 ± .0959	.7614 ± .0941	.7636 ± .0908	.7685 ± .0938
part	.7532 ± .1000	.7759 ± .0852	.7540 ± .0772	.8051 ± .0595	.7580 ± .0844	.7499 ± .0977	.7441 ± .0947	.7526 ± .0880	.7610 ± .0907
substance	.7560 ± .0967	.7791 ± .0809	.7508 ± .0755	.8073 ± .0567	.7542 ± .0828	.7436 ± .0952	.7370 ± .0919	.7477 ± .0862	.7580 ± .0889

Table 6: Each value represents the mean F1-score and standard deviation of all the concepts that belong to each analyzed category. Only the larger version of each model is reported. This is not an extensive list and categories are somewhat imbalanced. Categories were selected based on the number of sub-categories they contained.

Language Tags Matter for Zero-Shot Neural Machine Translation

Liwei Wu, Shanbo Cheng, Mingxuan Wang, Lei Li
ByteDance AI Lab
{*wuliwei.000,chenshanbo,wangmingxuan.89,lileilab*}@*bytedance.com*

Abstract

Multilingual Neural Machine Translation (MNMT) has aroused widespread interest due to its efficiency. An exciting advantage of MNMT models is that they could also translate between unsupervised (zero-shot) language directions. Language tag (LT) strategies are often adopted to indicate the translation directions in MNMT. In this paper, we demonstrate that the LTs are not only indicators for translation directions but also crucial to zero-shot translation qualities. Unfortunately, previous work tends to ignore the importance of LT strategies. We demonstrate that a proper LT strategy could enhance the consistency of semantic representations and alleviate the off-target issue in zero-shot directions. Experimental results show that by ignoring the source language tag (SLT) and adding the target language tag (TLT) to the encoder, the zero-shot translations could achieve a +8 BLEU score difference over other LT strategies in IWSLT17, Europarl, TED talks translation tasks.

1 Introduction

Neural Machine Translation (NMT) based on the *encoder-decoder framework with attention mechanism* (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017) has achieved state-of-the-art (SotA) results in many language pairs (Deng et al., 2018; Barrault et al., 2019). Pioneered by (Dong et al., 2015; Firat et al., 2016; Zoph and Knight, 2016; Ha et al., 2016; Johnson et al., 2017), researchers start to investigate the possibility of using a single model to translate between multiple languages, which is the Multilingual Neural Machine Translation (MNMT). Benefiting from the transferring ability of multilingual modeling, MNMT could achieve better translation quality between low-resource language directions than bilingual models (Gu et al., 2018; Wang et al., 2019). More exciting, MNMT could even translate between zero-shot language directions (Johnson et al., 2017; Gu et al., 2019; Pham et al., 2019; Kudugunta et al., 2019).

Unlike bilingual NMT, language-specific signals should be accessible to the MNMT model so that the model can distinguish the translation directions. Ha et al., (2016) first introduced a universal encoder-decoder framework for MNMT models with language-specific coded vocabulary to indicate different languages. The *encoder-decoder* architecture is identical to bilingual models (Bahdanau et al., 2015; Vaswani et al., 2017). To further simplify the MNMT models, Johnson et al., (2017) propose to add language tags (LTs) to the beginning of input data to indicate the target language. Then a shared vocabulary could be learned for all languages. The training data of different languages could thus be mixed-up to train the MNMT model. Such a strategy greatly simplifies the training and decoding procedure. We call it the *LT strategy*. This paper focuses on investigating the impact of LT strategies for zero-shot translation directions in MNMT (zero-shot MNMT). We conduct translation experiments (Section 3) and visualization analysis (Section 4) on several multilingual benchmarks with different LT strategies. We observe that:

- The TLT is more important than the SLT. The SLT even causes negative effects on the zero-shot translation.
- The placements of LTs have a surprisingly large impact on the translation quality. Placing different LTs on different parts of the NMT model lead to a +8 BLEU score difference in our experiments.

Our contributions are mainly twofold: (i) We

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3001–3007
August 1–6, 2021.

Strategy	Source sentence	Target sentence
Original	`Hello World!`	`¡Hola Mundo!`
T-ENC	**`__es__`** `Hello World!`	`¡Hola Mundo!`
T-DEC	`Hello World!`	**`__es__`** `¡Hola Mundo!`
S-ENC-T-ENC	**`__en__`** **`__es__`** `Hello World!`	`¡Hola Mundo!`
S-ENC-T-DEC	**`__en__`** `Hello World!`	**`__es__`** `¡Hola Mundo!`

Table 1: Examples of modified input data by different LT strategies. The bold tokens are the SLT (`__en__`) or TLT (`__es__`). T-ENC is identical to (Johnson et al., 2017), which adds the TLT to the encoder (source) side. T-DEC means placing the TLT on the decoder (target) side of model. S-ENC-T-ENC and S-ENC-T-DEC place the SLT on the encoder side, but the former also places the TLT on encoder side, while the latter on the decoder side.

find that the LT strategies are crucial for the zero-shot MNMT translation quality. Ignoring SLTs and placing the TLTs on the encoder side could achieve the best performance during our experiments. (ii) We conduct extensive visualization analysis to demonstrate that the proper LT strategy could enhance the consistency of semantic representation and alleviate the off-target issue (Zhang et al., 2020), thus improving the translation quality. To the best of our knowledge, this is the first paper to systematically study the importance of LT strategies for zero-shot translation quality.

2 Background and Notations

Improving the consistency of semantic representations and alleviating the off-target issue (Zhang et al., 2020) are effective ways to improve the zero-shot translation quality (Al-Shedivat and Parikh, 2019; Arivazhagan et al., 2019; Zhu et al., 2020). The semantic representations of different languages should be close to each other to get better translation quality (Ding et al., 2017). The off-target issue indicates that the MNMT model tends to translate input sentences to the wrong languages, which leads to low translation quality.

Due to its simplicity and efficiency, LT strategy has become a fundamental strategy for MNMT (Dabre et al., 2020). Though previous work adopted different LT strategies (Wang et al., 2018; Blackwood et al., 2018; Conneau and Lample, 2019; Liu et al., 2020b), the usages of LT strategies are intuitive and lack systematic study. In this paper, we investigate 4 popular LT strategies, namely `T-ENC`, `T-DEC`, `S-ENC-T-ENC` and `S-ENC-T-DEC`. Each of them only requires simple modifications to the input data. Table 1 comprehensively illustrates the strategies with an English to Spanish translation pair (`Hello World!` → `¡Hola Mundo!`).

3 Experiments

3.1 Experiment Settings

Datasets We carry out our experiments on the publicly available IWSLT17 (Cettolo et al., 2017), TED talks (Qi et al., 2018) and Europarl v7 (Koehn, 2005) datasets. Table 2 shows an overview of the datasets. We choose four different languages (English included) for both IWSLT17 and Europarl, and 20 languages for TED talks. All the training data are English-centric parallel data, which means either the source-side or target-side of the sentence pair is English. We have 6, 6, and 342 zero-shot translation directions and an average of 145k, 1.96M (M = million), and 187k sentence pairs per direction for the three datasets respectively. We choose the official tst2017, WMT `newstest08`, and the TED talks testsets (Qi et al., 2018) as our test sets, respectively. We learned a joint SentencePiece model (Kudo and Richardson, 2018) for sub-word training on all languages with 40,000 merge operations for each dataset. We limit the size of joint vocabulary to 40,000 for all three datasets.

Settings We use the open-source implementation (Ott et al., 2019) of Transformer model (Vaswani et al., 2017). Following the settings of (Liu et al., 2020a), we use a 5-layer encoder and 5-layer decoder variation of Transformer-base model (Vaswani et al., 2017) for TED and IWSLT17. For Europarl v7, we use a standard Transformer-big model (Vaswani et al., 2017). Sentence pairs are batched together by approximate sentence length. Each batch has approximately 30,000 source tokens and 30,000 target tokens. We use the Adam (Kingma and Ba, 2015) optimizer to update the parameters and

Dataset	languages	#zero-shot directions	#training sents per direction	#sents per testset
IWSLT17	en, it, nl, ro	6	145k	1144
Europarl	en, fr, de, es	6	1.96m	2000
TED	en, ar, he, ru, ko, it, ja zh, es, fr, pt, nl, tr, ro pl, bg, vi, de, fa, hu	342	187k	4507

Table 2: An overview of the datasets. The second column is the languages the training data contains. The third column denotes the number of zero-shot translation directions. The fourth and fifth column denote the averaged number of training data and test data per language direction, respectively.

Dataset	LT Strategy	Supervised	Zero-Shot	Off-Target (%)
IWSLT17	T-ENC	32.30	**16.00** (+14.02)	**9.16**
	T-DEC	32.43	10.44	29.50
	S-ENC-T-ENC	32.56	1.98	94.14
	S-ENC-T-DEC	32.39	7.67	48.87
Europarl	T-ENC	35.55	**32.25** (+24.24)	1.18
	T-DEC	35.49	30.73	**1.13**
	S-ENC-T-ENC	35.53	8.01	79.53
	S-ENC-T-DEC	35.53	29.81	2.26
TED talks	T-ENC	25.63	**10.69** (+8.78)	**12.63**
	T-DEC	25.58	3.11	58.47
	S-ENC-T-ENC	25.84	4.07	65.03
	S-ENC-T-DEC	25.63	1.91	77.02

Table 3: Translation results on 3 datasets. The **supervised** and **zero-shot** column denote the averaged BLEU score of supervised or zero-shot directions. The **off-target (%)** denotes the averaged percentage of sentences being translated to wrong languages in zero-shot directions.

train each model for 100,000 steps to make sure it converges. We use beam search for heuristic decoding, and set the beam size to 4. We use SacreBLEU (Papineni et al., 2002; Post, 2018) to evaluate the translation results. To calculating the percentage of off-target translations, we use the langdetect[1] tool to detect the language of the translated sentences.

3.2 Experimental Results

We show the translation results on the IWSLT17, Europarl, and TED talks datasets in Table 3. For all three datasets, different strategies achieve comparable BLEU score on supervised directions. However, for the zero-shot directions, the BLEU score varies significantly using different LT strategies. One observation is that the `T-ENC` strategy consistently outperforms the other three strategies on all datasets in terms of BLEU score with large margin, regardless of the corpus size and number of languages. In terms of off-target issue, `T-ENC` achieves the best performance in most cases.

Besides, ignoring the SLT (`T-ENC` v.s. `S-ENC-T-ENC`) also helps the zero-shot BLEU score. The percentage of off-target translations reaches 94.14% in the **IWSLT17** dataset by `S-ENC-T-ENC` strategy, while only 9.16% by `T-ENC` strategy. It indicates that the model translates almost all the sentences to the wrong languages in `S-ENC-T-ENC`, while to the right languages in `T-ENC`. It proves again the SLT hurts the zero-shot translation.

Another interesting observation is that placing the TLT on the encoder side also helps the zero-shot performance. Compared with `T-ENC`, both the translation quality and off-target performance are significantly worse in `T-DEC`. We will study

---

[1] https://github.com/Mimino666/langdetect

the reasons behind the above observations by visualization analysis in Section 4.

4 Visualization Analysis

We conduct the visualizations on the TED talks data to analyze the impact of different LT strategies on the semantic representation consistency and the off-target issue in MNMT.

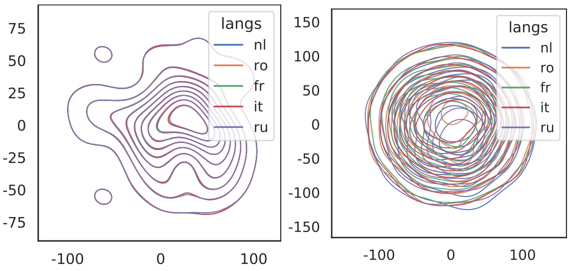

(a) T-ENC (b) T-DEC

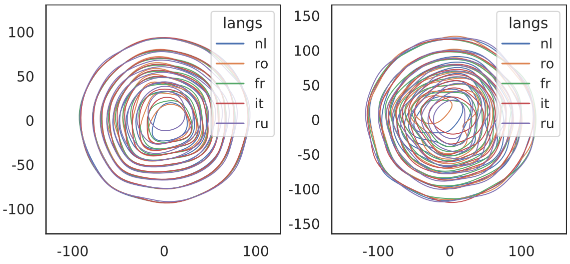

(c) S-ENC-T-ENC (d) S-ENC-T-DEC

Figure 1: KDE visualization of encoder output using different LT strategies. 5 languages (nl, ro, fr, it, ru → zh) are randomly chosen for better readability.

Enhancing the Semantic Representation Consistency Figure 1 shows the kernel density estimation (KDE) (Parzen, 1962) of t-SNE (Van der Maaten and Hinton, 2008) reduced average encoder output on different languages. We randomly chose 5 source languages (nl, ro, fr, it, ru → zh) instead of all languages for clearer visualization. We choose 100 sentences for each language, and each sentence has its corresponding translation in the other 4 languages. The contour lines drawn by Kernel Density Estimation tools [2] was used to estimate the semantic distribution of the encoder outputs. The contour lines visualize the semantic representation of different languages. The representations are more consistent if the contour lines of different languages overlap more with each other.

The contour lines are nearly perfectly overlap with each other in `T-ENC` (Figure 1a), while they do not for the other strategies. Comparing Figure 1a and Figure 1c, we can see that ignoring SLT greatly helps the model to learn more consistent representations. Comparing Figure 1a and Figure 1b, placing TLT to encoder side instead of the decoder side also helps the semantic consistency. Both comparisons validate that `T-ENC` could learn the most consistent and different semantic representations, thus achieves the best BLEU score. It might be why the shape of contour lines in `T-ENC` is significantly different from other strategies.

Alleviating the off-target Issue Figure 2 shows the attention visualization of a Russian to Italian translation example using different LT strategies. The x-axis is the Italian translation.

In Figure 2a1, `T-ENC` strategy pays attention to the TLT (in this case, the token `__it__` in the red background) during the whole translation procedure (left-to-right). Compared to `T-ENC`, both `T-DEC` and `S-ENC-T-DEC` pay less attention to the TLT after a few tokens are generated. It validates that placing the TLT on the encoder side would also help the model distinguish the target languages. The `S-ENC-T-ENC` pays nearly equal attention to both SLT and TLT, which might make the model confused about which one is the target language. Both comparisons prove that the `T-ENC` strategy has the best ability to distinguish the target languages, thus alleviates the off-target issue.

Combining Both Semantic Consistency and off-target Issue Figure 3 visualizes the cosine similarity of the layer-wise encoder and decoder output of different languages in zero-shot setting (English excluded). We sampled 100 muti-way data from the test set and averaged the cosine similarity between each language.

In the many-to-one setting, we randomly select Russian as the target language and translate the other 18 languages to Russian to obtain the model outputs. The similarity improves from encoder layer 0 to 3 and decoder layer 0 to layer 4, which indicates that the semantic consistency improves as the layer goes up. Interestingly, the similarity drops from encoder layer 3 to layer 4. It might be because the decoder interacts with the encoder directly between encoder layer 4 and decoder layer 0, thus interferes with the top-layer encoder output. But the dropping trend is less rapid in `T-ENC` than in other strategies. The `T-ENC` achieves the highest similarity on the last layer, which shows that the `T-ENC` learns more consistent semantics representations.

---

[2] https://seaborn.pydata.org/generated/seaborn.kdeplot.html

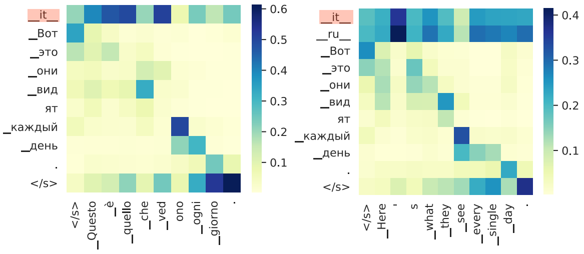

(a1) T-ENC (a2) S-ENC-T-ENC

(a) The Decoder Cross Attention

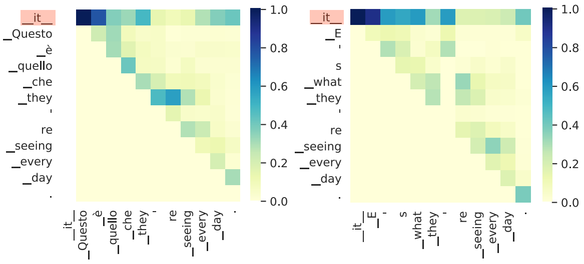

(b1) T-DEC (b2) S-ENC-T-DEC

(b) The Decoder Self Attention

Figure 2: Attention visualization on a Russian to Italian translation example using different LT strategies. Note that we present the cross-attention for `T-ENC` and `S-ENC-T-ENC`, the decoder self-attention for `T-DEC` and `S-ENC-T-DEC` to visualize the TLT token.

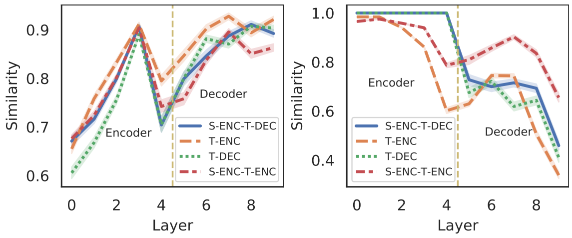

(a) many-to-one (b) one-to-many

Figure 3: The cosine similarity of layer-wise output using different LT strategies. Note that layer 0 to 4 are the encoder layers, layer 5 to 9 are the decoder layers.

In the one-to-many setting, we treat Russian as the source language and translate Russian to the other 18 languages to get the model output. The semantic similarity drops as the layer goes up in all four strategies. It indicates that the model can distinguish different target languages as the layer goes up. `T-ENC` achieves the lowest similarity at the last layer output among all strategies. It shows again that the `T-ENC` has the best ability to alleviate the off-target issue.

5 Conclusion

We show that the language tags in MNMT are not just indicators for translation directions but also significantly impact the zero-shot translation quality. By extensive experiments and visualization analysis, we found that (i) ignoring the SLTs could help the models learn consistent semantic representations. (ii) Placing the TLTs on the encoder side could help the decoder pay more attention to the target language, thus alleviating the off-target issue. Zero-shot translation quality could be improved by investigating how to enhance the semantic representation consistency further and alleviate the off-target issue by optimizing LT strategies. We will conduct methods to optimize the LT strategy in our future work.

References

Maruan Al-Shedivat and Ankur P Parikh. 2019. Consistency by agreement in zero-shot neural machine translation. *arXiv preprint arXiv:1904.02338*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey. 2019. The missing ingredient in zero-shot neural machine translation. *arXiv preprint arXiv:1903.07091*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loïc Barrault, Ondřej Bojar, Marta R Costa-Jussa, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. 2019. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61.

Graeme W. Blackwood, Miguel Ballesteros, and Todd Ward. 2018. Multilingual neural machine translation with task-specific attention. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3112–3122. Association for Computational Linguistics.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. 2017. Overview of the iwslt 2017 evaluation campaign. In *International Workshop on Spoken Language Translation*, pages 2–14.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing*

Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 7057–7067.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A comprehensive survey of multilingual neural machine translation. *arXiv preprint arXiv:2001.01115*.

Yongchao Deng, Shanbo Cheng, Jun Lu, Kai Song, Jingang Wang, Shenglan Wu, Liang Yao, Guchun Zhang, Haibo Zhang, Pei Zhang, et al. 2018. Alibaba's neural machine translation systems for wmt18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 368–376.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 866–875. The Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 344–354. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2019. Improved zero-shot neural machine translation via ignoring spurious correlations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1258–1268. Association for Computational Linguistics.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Trans. Assoc. Comput. Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Sneha Reddy Kudugunta, Ankur Bapna, Isaac Caswell, and Orhan Firat. 2019. Investigating multilingual NMT representations at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1565–1575. Association for Computational Linguistics.

Danni Liu, Jan Niehues, James Cross, Francisco Guzmán, and Xian Li. 2020a. Improving zero-shot translation by disentangling positional information. *CoRR*, abs/2012.15127.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The Association for Computational Linguistics.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of* *the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Ngoc-Quan Pham, Jan Niehues, Thanh-Le Ha, and Alexander H. Waibel. 2019. Improving zero-shot translation with language-independent constraints. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 1: Research Papers*, pages 13–23. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 529–535. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual neural machine translation with soft decoupled encoding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. 2018. Three strategies to improve one-to-many multilingual translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2955–2960.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1628–1639. Association for Computational Linguistics.

Changfeng Zhu, Heng Yu, Shanbo Cheng, and Weihua Luo. 2020. Language-aware interlingua for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1650–1655.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 30–34. The Association for Computational Linguistics.

Latent Reasoning for Low-Resource Question Generation

Xinting Huang,[1] Jianzhong Qi,[1] Yu Sun,[2] Rui Zhang[3]*
[1]The University of Melbourne, [2]Twitter Inc., [3]www.ruizhang.info
{xintingh@student., jianzhong.qi@}unimelb.edu.au,
ysun@twitter.com, rui.zhang@ieee.org

Abstract

Multi-hop question generation requires complex reasoning and coherent language realization. Learning a generation model for the problem requires extensive multi-hop question answering (QA) data, which are limited due to the manual collection effort. A two-phase strategy addresses the insufficiency of multi-hop QA data by first generating and then composing single-hop sub-questions. Learning this generating and then composing two-phase model, however, requires manually labeled question decomposition data, which is labor intensive. To overcome this limitation, we propose a novel generative approach that optimizes the two-phase model without question decomposition data. We treat the unobserved sub-questions as latent variables and propose an objective that estimates the true sub-questions via variational inference. We further generalize the generative modeling to single-hop QA data. We hypothesize that each single-hop question is a sub-question of an unobserved multi-hop question, and propose an objective that generates single-hop questions by decomposing latent multi-hop questions. We show that the two objectives can be unified and both optimize the two-phase generation model. Experiments show that the proposed approach outperforms competitive baselines on HOTPOTQA, a benchmark multi-hop question answering dataset.

Table 1: Multi-Hop Question Reasoning Example

Supportive Evidence	**Paragraph A. Dario Franchitti**
	[1] George ..., known professionally as Dario Franchitti, is a retired Scottish racing driver. [2] After Franchitti did not secure a single-seater drive in 1995, he was *contracted* by the AMG team to *compete* in touring cars in the DTM and its successor — the International Touring Car Championship.
	Paragraph B. Mercedes-AMG
	[1] Mercedes-AMG GmbH (AMG)... is the high performance division of Mercedes-Benz. [2] Mercedes-AMG is *headquartered* in Affalterbach, Baden Württemberg, Germany.
Reasoning Progress	< Dario Franchitti , contracted by , AMG > < Dario Franchitti , competed in , DTM > < AMG , headquartered in , Affalterbach, Baden Württemberg, Germany >
Multi-hop Question	After he was contracted by the team that is headquartered in Affalterbach, Baden Württemberg, Germany, Dario Franchitti competed in what series?
Sub-Questions	Which team is headquartered in Affalterbach, Baden Württemberg, Germany? After contracted by AMG, Dario Franchitti competed in what series?
Answer	DTM

1 Introduction

Question generation aims to automatically generate valid and coherent questions based on given context, which is widely applied to enrich question answering (QA) datasets, facilitate text comprehension (Ko et al., 2020), seek clarification in conversation (Rao and Daumé III, 2019), etc. Recently, neural encoder-decoder based approaches have shown promising results for simple, single-hop question generation (Du and Cardie, 2018). Such approaches directly maps context (e.g., text passages) to questions without *reasoning*, and thus struggle when generating multi-hop questions (Pan et al., 2020). Here, reasoning refers to identifying and aggregating the relevant information taken from multiple documents to derive the question. Table 1 illustrates the reasoning process of a multi-hop question; in this example, the entity that links the two passages, i.e., "AMG", is firstly identified, and the relations around it in the context are transformed into a question. To model such reasoning processes in an end-to-end manner requires extensive training data, and is thus impractical due to the

---

[*] Rui Zhang is the corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3008–3022
August 1–6, 2021.

extensive collection effort of multi-hop QA data.

To address this problem, recent studies propose to augment the generation model with an explicit reasoning progress. For example, a straightforward solution is to identify the anchoring entities via named entity recognition (NER), and find relations via relation extraction. The extracted structural reasoning path, in the form of subject-predicate-object triples as illustrated in Table 1, is then fed to the generation model as auxiliary features (Yu et al., 2020b). However, the reasoning capability is constrained by the off-the-shelf extraction tools which cannot be extended to arbitrary context (Yang et al., 2018; Dhingra et al., 2020).

Another line of recent studies on multi-hop question answering models the reasoning process by decomposing a multi-hop question into several sub-questions (Min et al., 2019; Wolfson et al., 2020). As illustrated in Table 1, the answer of the multi-hop question can be derived by answering a series of single-hop sub-questions. Ideally, question generation can also adopt this two-phase strategy which first generates sub-questions and then composes the sub-questions into a multi-hop question. However, this strategy requires a parallel corpus that annotates each multi-hop question to its corresponding sub-questions, and obtaining such annotations still requires extensive efforts and costs.

To address these issues, we propose to jointly optimize the two-phase model using *non-parallel* single-hop and multi-hop corpuses only, in which the questions are not paired. We propose a generative objective that models the multi-hop and single-hop question generation (QG) tasks in a unified way. The key idea is that each question, either multi-hop or single-hop, can be considered a partially observed ⟨multi-hop question, sub-question⟩ pair and treat the unobserved part as a latent variable. In the generative modeling of multi-hop QG, we use the two-phase model as a generation model and introduce a posterior model to estimate unobserved sub-questions. The generation and the posterior models are jointly optimized via variational inference (Kingma and Welling, 2014). For generative single-hop QG, we instead use the two-phase model as a posterior model to estimate unobserved multi-hop questions, and the posterior model is jointly optimized with a generation model that decomposes a multi-hop question into sub-questions. In this way, we integrate the optimization of the two-phase model in both generative multi-hop and single-hop QG tasks, serving as the generation and the posterior model, respectively.

Optimizing the generative objective in the text space is, however, prone to compounding errors due to the diversities of potential reasoning paths. There are multiple ways to raise a single-hop question given the same piece of information, and it is challenging to find the valid one only given the text passages. We address this challenge by equipping the generative modeling with a planning mechanism that uses a latent variable to encode the desired reasoning path. In this way, the inference of sub-questions is guided by a pre-sampled plan (i.e., the latent variable) and thus maintains consistency with the target multi-hop question. We achieve latent variable learning by incorporating an end-to-end differentiable bottleneck into the sub-question generation model, which can be naturally integrated into the overall objective. Moreover, the proposed planning mechanism also promotes a more stable training. This is because the original generative modeling involves a sequential sampling of latent variables (i.e., sub-questions), which is known to cause high variance and result in an unstable training (He et al., 2020). The planning mechanism relieves the sequential sampling requirement, since it encodes the high-level planning and covers the dependency between sub-questions.

Our contributions are summarized as follows:

- We propose a novel generative objective that unifies non-parallel question corpuses and relieves the requirements of extensive annotations for learning a two-phase question generation model.
- We propose a planning mechanism to guide the generation towards sub-questions that are more probable to compose into a multi-hop question.
- We conduct experiments on a benchmark multi-hop question answering dataset. The results show that our approach outperforms the state-of-the-art under both language generation and question answering based evaluations.

2 Preliminaries

Let $D_{\mathcal{M}} = \{(q_i, a_i, c_i) | 1 \leq i \leq N\}$ be a set of N multi-hop question-answer-evidence triples, where the evidence is a set of potentially relevant sentences $c_i = \{d_1, d_2, ..., d_k\}$, and each multi-hop question q requires reasoning over multiple sentences to find the answer a. Multi-hop question generation (QG) aims to generate a question q that has the pre-selected answer a given the evidence set

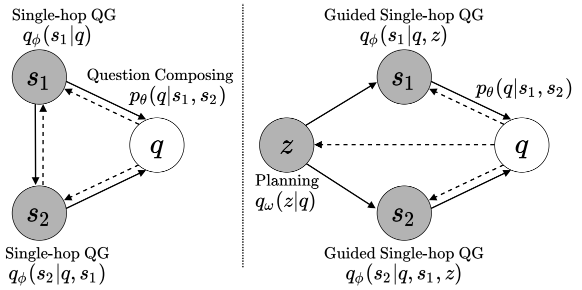

(a) Generative multi-hop QG (left) with planning (right)

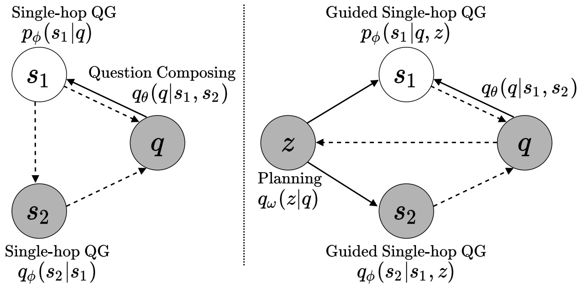

(b) Generative single-hop QG (left) with planning (right)

Figure 1: A graphical representation of the proposed generative model. White circles denote the observed variables and gray circles denote the latents. [1]

c. Existing studies adopt a strategy commonly used in single-hop question generation, which formulates multi-hop QG as a seq-to-seq problem. Since extensive annotation efforts are needed to produce multi-hop QG examples, few multi-hop QG examples are available. Thus, a naive adoption of seq-to-seq learning may not yield an effective multi-hop QG model, especially in the low-resource scenario.

To address data insufficiency, a two-phase strategy is considered based on the assumption that each multi-hop question q can be decomposed into two single-hop sub-questions s_1 and s_2.[2] The multi-hop question generation is then performed by a sub-question generation model $p_\mathcal{S}$ and a question composing model $p_\mathcal{C}$ as

$$p(q|a, c) = p_\mathcal{C}(q|s_1, s_2)p_\mathcal{S}(s_2, s_1|a, c). \qquad (1)$$

The training of these two models require question decomposition data, which are pairs of a multi-hop question and its corresponding sub-question annotations $\{(q, \langle s_1, s_2\rangle)\}$. However, it is non-trivial to obtain the question decomposition data, which requires extensive human annotation effort.

3 Proposed Model

We take a two-phase approach for multi-hop question generation while do not require a question decomposing dataset that contains pairs of multi-hop questions and sub-questions. We assume that a single-hop question answering dataset $D_\mathcal{S}$ and a multi-hop dataset $D_\mathcal{M}$ are available for training. Both datasets are non-parallel, i.e., contain question-answer pairs but not sub-questions, and the evidence passages of both datasets shall come from the same source (e.g., Wikipedia articles).

Under these problem settings, we aim to learn the single-hop QG model $p_\mathcal{S}$ and the question composing model $p_\mathcal{C}$ using both $D_\mathcal{S}$ and $D_\mathcal{M}$. To effectively train these two models in the absence of question decomposition data, we propose a unified generative formulation that naturally connects single-hop and multi-hop questions. Specifically, in modeling the generation process of multi-hop questions, we treat the corresponding sub-questions as latent variables and propose an objective that jointly optimizes $p_\mathcal{S}$ and $p_\mathcal{C}$ (Sec. 3.1). We further extend the generative formulation to model the generation of single-hop questions, and both generation processes together form the overall optimization objective (Sec. 3.2). Then, we propose a planning-aware generation strategy to better optimize the objective in Sec. 3.3. We summarize the overall learning and inference process in Sec. 3.4.

3.1 Generative Modeling of Multi-Hop QG

We now reconsider the two-phase question generation strategy in Eqn. 1. Since we do not have the parallel data, it is infeasible to directly model the conditional probability $p(q|s)$, where $s = \{s_1, s_2\}$ is the set of sub-questions of q. We thus propose to treat the unobserved sub-questions as latent variables, and describe $p(q|a, c)$ in a generative way as

$$p(q) = \sum_s p(q, s) = \sum_s p_\theta(q|s)p_\psi(s) \qquad (2)$$

where $p(q)$ and $p(q, s)$ are shorthands for $p(q|a, c)$ and $p(q, s|a, c)$, [3] p_ψ is a conditional prior model, and p_θ is a generation model for multi-hop questions. Since this likelihood is intractable, we instead derive and optimize its evidence lower bound

---

[1] Note that the generation of single-hop questions s, multi-hop question q, and planning variables z are conditioned on evidence set c and answer a, which is omitted in Fig. 1.

[2] The formulation can be easily extended to more sub-questions

[3] Same below for brevity when the context is clear.

(ELBO) (Kingma and Welling, 2014)

$$\log p(q) \geq \mathbb{E}_{q_\phi(s|q)}[\log \frac{p_\theta(q|s)p_\psi(s)}{q_\phi(s|q)}]$$
$$= \mathbb{E}_{q_\phi(s|q)}[\log p_\theta(q|s)] - \text{KL}(q_\phi(s|q)||p_\psi(s)) \tag{3}$$

where $q_\phi(s|q)$ is a posterior model for latent variable s, and KL denotes the Kullback-Leibler divergence. We now substitute the latent variable s with two sub-questions, s_1 and s_2, and define the factorized form of the posterior and the prior in a hierarchical manner

$$\begin{aligned} q_\phi(s|q) &= q_\phi(s_2|q, s_1)q_\phi(s_1|q) \\ p_\psi(s) &= p_\psi(s_2|s_1)p_\psi(s_1). \end{aligned} \tag{4}$$

We can now rewrite the ELBO in Eqn. 3 with the factorization and obtain

$$\begin{aligned} \log p(q) \geq\ & \mathbb{E}_{q_\phi(s_1|q)q_\phi(s_2|s_1,q)}[\log p_\theta(q|s_1, s_2)] \\ & - \text{KL}(q_\phi(s_1|q)||p_\psi(s_1)) \\ & - \text{KL}(q_\phi(s_2|q, s_1)||p_\psi(s_2|s_1)) \\ & := \mathcal{L}_{\text{ELBO}}(q) \end{aligned} \tag{5}$$

Fig. 1(a) shows the directed graphical model of the generative modeling of multi-hop question generation. Specifically, given an evidence set and a pre-selected answer, a single-hop question s_1 is first sampled. Given s_1 and relevant information in the context, a second sub-question s_2 that satisfies a valid reasoning process is further sampled. Since two sub-questions are both unobserved, we estimate s_1 and s_2 using the posterior model q_ϕ. The sub-questions then form the observed multi-hop question q via question composing as $p_\theta(q|s_1, s_2)$.

To perform effective optimization, we tie the parameters of the posterior model q_ϕ at different hierarchies, i.e., $q_\phi(s_1|\cdot)$ and $q_\phi(s_2|\cdot)$, as one single-hop QG model. Such parameter tying also applies to the prior model p_ψ. We implement the generation model p_θ, the prior p_ψ, and the posterior q_ϕ in Eqn. 5 using pre-trained encoder-decoder models which will be detailed in Sec. 3.4. We notice that the prior p_ψ and the generation model p_θ actually play the same role as the single-hop QG model $p_\mathcal{S}$ and question composing model $p_\mathcal{C}$ in Eqn. 1. Thus, the generative modeling enables a joint optimization of $p_\mathcal{S}$ and $p_\mathcal{C}$ using multi-hop QA data only and without question decomposing data.

3.2 Generative Modeling of Single-Hop QG

Considering that the multi-hop QA data is limited, we propose to integrate single-hop QA data into the joint optimization objective. We extend the proposed generative modeling by assuming that each single-hop question is obtained by decomposing an unobserved multi-hop question. With a slight abuse of notation, we use (s, a, c) to denote a single-hop question-answer-evidence triple, and describe $p(s|a, c)$ as

$$p(s) = \sum_q p(s, q) = \sum_q p_{\theta'}(s|q)p_{\psi'}(q) \tag{6}$$

where we omit the condition as in Eqn. 2, and q is a multi-hop question that has a sub-question s. The generation model $p_{\theta'}$ and the prior model $p_{\psi'}$ are parameterized with θ' and ψ', respectively. We treat the unobserved q as a latent variable and derive the evidence lower bound as

$$\begin{aligned} \log p(s) \geq \mathbb{E}_{q_{\phi'}(q|s)}[\log p_{\theta'}(s|q)] \\ -\text{KL}(q_{\phi'}(q|s)||p_{\psi'}(q))] := \mathcal{L}_{\text{ELBO}}(s) \end{aligned} \tag{7}$$

where $q_{\phi'}$ is a posterior model to estimate the unobserved question q.

Fig. 1(b) illustrates the generative modeling for single-hop QG. Specifically, a multi-hop question is first sampled by the prior $p_{\psi'}$, and we assume that its sub-question set includes the observed single-hop question s. The question s is then generated by decomposing the multi-hop question q via $p_{\theta'}(s|q)$. We estimate the unobserved multi-hop question q using the posterior model $p_{\phi'}$.

We observe that the posterior approximation in single-hop QG (dashed line in Fig. 1(b)-left) is the same as the generative process in multi-hop QG (solid line in Fig. 1(a)-left). Thus, we can realize the posterior model $q_{\phi'}(q|s)$ by reusing the prior p_ψ and the generative model p_θ in Eqn. 5 as

$$q_{\phi'}(q|s) = p_\theta(q|\hat{s}, s)p_\psi(\hat{s}|s) \tag{8}$$

where $\hat{s}$ is the unobserved second sub-question that forms the multi-hop question together with s. Note that we no longer need a hierarchical form since one sub-question is observed.

Further, we observe that the generative process in single-hop QG (solid line in Fig. 1(b)-left) is part of the posterior approximation of multi-hop QG (dashed line in Fig. 1(b)-left). This way, we realize the generation model $p_{\theta'}$ and the prior $p_{\psi'}$ using the models already present in multi-hop QG

$$\begin{aligned} p_{\theta'}(s|q) &= q_\phi(s|q) \\ p_{\psi'}(q) &= p_\theta(q|s_1, s_2)p_\psi(s_2|s_1)p_\psi(s_1) \end{aligned} \tag{9}$$

Table 2: Question Generation Diversification Example.

Supportive Evidence	
After Franchitti did not secure a single-seater drive in 1995, he was contracted by the AMG team to compete in touring cars in the DTM and its successor — the International Touring Car Championship.	
Potential Generated Sub-Questions	
Q: Did Dario Franchitti secure a single seater drive in 1995?	**A:** No
Q: Dario Franchitti was contracted by which team to compete in the DTM?	**A:** AMG
Q: The International Touring Car Championship is the successor of what series?	**A:** DTM
Q: After contracted by AMG, Dario Franchitti competed in what series?	**A:** DTM

where the prior $p_{\psi'}(q)$ is inferred by first estimating and then composing the latent sub-questions s_1 and s_2. Note that p_θ, p_ψ, and q_ϕ are all taken from the generative modeling of multi-hop QG. Thus, the single-hop QG objective (Eqn. 7) optimizes the same set of models as in multi-hop QG objective (Eqn. 3). This way, we seamlessly unify the multi-hop and single-hop QA data for joint optimization.

3.3 Planning Guided Question Generation

There is a challenge under the generative formulation: the diversification of feasible generated questions can impinge the model training. Given the same evidence set and pre-selected answer, there can be multiple ways to raise a questions (Lee et al., 2020). However, not every potential single-hop question is qualified as a sub-question to form the target multi-hop question, as illustrated in Table. 2. To address this challenge, we propose to learn a latent planning variable which serves as a generation planning to guide the generation process.

The latent planning variable aims to capture the high-level reasoning required to answer the multi-hop questions, which is abstracted as a reasoning path in existing studies. In order to model decision making of the reasoning path, we define the latent variable z as a discrete variable. We now incorporate the latent variable into the generative modeling of multi-hop QG

$$\begin{aligned}\log p(q) &= \log \sum_s \int_z p(q,z,s)dz \\ &\geq \mathbb{E}_{q_\omega(z|q)}[p(q|z)] - \mathrm{KL}(q_\omega(z|q)||p_\omega(z)) \\ &:= \mathcal{L}_{\mathrm{ELBO}}(q,z)\end{aligned} \tag{10}$$

where q_ω and p_ω are posterior and prior models, respectively, and the reason of having the same parameters ω will be detailed later. The conditional probability $p(q|z)$ is modeled by letting the terms of $\mathcal{L}_{\mathrm{ELBO}}(q)$ in Eqn. 5 be additionally conditioned on the sampled latent variable z (as illustrated in Fig. 1(a)-right). The generation of sub-questions, both prior p_ψ and posterior q_ϕ, is now aware of the planning as

$$\begin{aligned}q_\phi(s|q,z) &= q_\phi(s_1|q,z)q_\phi(s_2|q,z) \\ p_\psi(s|z) &= p_\psi(s_1|z)p_\psi(s_2|z).\end{aligned} \tag{11}$$

We now no longer need a hierarchical form like Eqn. 4, since the latent planning variable already encodes the information of the other sub-question. Thus, this formulation also alleviates the high variance issue commonly encountered in hierarchical variational training (Vahdat and Kautz, 2020).

We also consider the planning guided mechanism in the generative modeling of single-hop QG

$$\begin{aligned}\log p(s) &= \log \sum_q \int_z p(s,z,q)dz \\ &\geq \mathbb{E}_{q_\omega(z|q)}[p(s|z)] - \mathrm{KL}(q_\omega(z|s)||p_\omega(z)) \\ &:= \mathcal{L}_{\mathrm{ELBO}}(s,z)\end{aligned} \tag{12}$$

where $p(s|z)$ is modeled by letting the prior and the posterior in $\mathcal{L}_{\mathrm{ELBO}}(s)$ be additionally conditioned on z The realizations in Eqn. 8 and Eqn. 9 are now formulated as

$$\begin{aligned}q_{\phi'}(q|s,z) &= p_\theta(q|\hat{s},s)p_\psi(\hat{s}|z) \\ p_{\psi'}(q|z) &= p_\theta(q|s_1,s_2)p_\psi(s_1|z)p(s_2|z)\end{aligned} \tag{13}$$

We implement the latent variable as discretized VAE (van den Oord et al., 2017) by adding a learnable codebook between the encoder and the decoder. The codebook is a set of prototype vectors $e_k, k \in 1, 2...K$, each having the same dimensionality as that of the encoder output. The discrete variable is obtained by using a nearest-neighbor lookup to find the vector closest to the encoder output. The corresponding prototype vector is then fed into the decoder as an additional context embedding to which every decoding step could attend. With this discretization bottleneck design, the encoder-decoder model and the codebook can be jointly optimized.

3.4 Learning and Inference

We initialize the generative model p_θ, the prior p_ψ and the posterior q_ϕ using BART (Lewis et al., 2020), a pre-trained seq-to-seq model. BART uses the standard Transformer based encoder-decoder architecture (Vaswani et al., 2017), and is optimized by reconstructing the intentionally corrupted documents. We adopt an initial fine-tuning step for all three models using question answering data $D_{\mathcal{S}}$ and $D_{\mathcal{M}}$, which adjusts the initialization pretrained from general texts to better fit the question generation tasks. We then optimize p_θ, p_ψ, and q_ϕ together with the discretization bottleneck q_ω using the generative modeling of both multi-hop and single-hop question answering data

$$\mathcal{L} = \sum_{q \in D_{\mathcal{M}}} \mathcal{L}_{\text{ELBO}}(q, z) + \sum_{s \in D_{\mathcal{S}}} \mathcal{L}_{\text{ELBO}}(s, z) \quad (14)$$

After training the single-hop QG model (i.e., p_ψ), question composing model (i.e., p_θ), and the bottleneck q_ω, inference follows the two-stage strategy. We first infer a latent planning variable given the evidence set and the answer. The sub-questions are generated based on the inferred planning variable and are composed into a multi-hop question.

4 Experiments

To show the effectiveness of the proposed approach, *planning guided latent reasoning* (PLAR), we experiment on two multi-hop question generation settings (Sec. 4.1). We compare against state-of-the-art approaches in both settings (Sec. 4.2). We further consider a question answering based performance measure, and analyze the effectiveness of the proposed generative modeling (Sec. 4.3).

4.1 Settings

We use HOTPOTQA (Yang et al., 2018), a crowd-sourced multi-hop question answering (QA) dataset in our experiments. It contains over 90K question answering examples, and the evidence set of each question includes relevant paragraphs from Wikipedia. The question-relevant sentences within these paragraphs are further annotated as supporting facts. We follow the original data split of HOTPOTQA, which includes 90,440 / 6,072 examples for training and evaluation, respectively. We further hold out 6,072 examples from the training data as the validation set. We use SQuAD (Rajpurkar et al., 2016) as the single-hop QA dataset, which has over 100K questions also crowd-sourced based on Wikipedia articles. Following the conventional evaluation metrics, we use n-gram **BLEU** (Papineni et al., 2002), **METEOR** (Banerjee and Lavie, 2005), and **ROUGE-L** (Lin and Hovy, 2002) to evaluate the question generation quality.

We consider two input settings to thoroughly evaluate the multi-hop question generation (QG) performance: *sentence-level* and *paragraph-level*. In the first setting, following the existing multi-hop QG task formulation (Pan et al., 2020; Yu et al., 2020b), we take the question-relevant sentences (i.e., supporting facts) along with the answer as inputs to generate the question. However, human annotated supporting facts are not always available, while identifying two relevant paragraphs is relatively achievable. Thus, we further consider a paragraph-level setting where, besides the answer, we instead use the paragraphs containing supporting facts as part of the input. In both settings, in order to simulate a low-resource scenario, we train PLAR and other baselines using two different subsets of the question answering examples, HOTPOT-10K and HOTPOT-30K, containing 10K and 30K randomly sampled training examples, respectively.

Note that we do not utilize any annotated question decomposition dataset (e.g., QDMR (Wolfson et al., 2020)). This is because it is labour-intensive to obtain the extra question decomposition annotations, which are not present in HotpotQA. Thus, it is not practical to assume such decomposition annotations would be available in different QA tasks. We aim to tackle this challenge by utilizing non-parallel single-hop questions, which is relatively easy to acquire and do not require extra task-specific annotations.

We compare with three baselines that are based on seq-to-seq models and are competitive in single-hop question generation tasks: **ASs2s** (Kim et al., 2019), **Maxout-QG** (Zhao et al., 2018), **BART** (Lewis et al., 2020). We compare with two baselines that consume auxiliary reasoning path features for multi-hop QG: **RC-QG** (Yu et al., 2020b) uses reasoning chains built via named entity recognition and relation extraction; and **SG-DQG** (Pan et al., 2020) adopts semantic role labeling techniques to build semantic graphs. We also compare the full **PLAR** with its two variants: **Pipeline** individually trains a single-hop QG model and a question composing model using synthetic question decomposition data obtained as Perez et al. (2020); and **PLAR w/o plan** uses the generative objectives as PLAR without the planning mechanism.

Table 3: Question Generation Results (Sentence-Level)

Model		Hotpot-30K				Hotpot-10K			
		BLEU-1	BLEU-4	METEOR	ROUGE-L	BLEU-1	BLEU-4	METEOR	ROUGE-L
Seq-to-Seq Application	ASs2s	27.12	10.11	13.27	25.69	24.32	9.27	11.93	23.20
	Maxout-QG	28.21	10.26	13.64	25.80	25.47	9.19	11.84	23.51
	BART	30.52	11.15	14.87	27.22	26.10	10.12	12.46	23.63
Reasoning Path Enhanced	RC-QG	30.86	11.36	15.29	28.66	29.31	11.16	14.88	26.89
	SG-DQG	32.93	12.32	16.40	29.81	31.12	12.27	15.25	27.90
Proposed	Pipeline	30.11	11.65	15.20	27.28	28.23	10.02	13.21	25.21
	PLAR w/o plan	35.19	13.48	17.54	31.02	33.68	13.87	16.64	28.87
	PLAR	**37.32**	**14.94**	**18.87**	**32.63**	**35.96**	**15.32**	**17.37**	**29.85**

4.2 Overall Results

Table 3 shows that PLAR consistently outperforms baselines on both subsets in the sentence-level input setting. We can see that PLAR achieves a significant performance gain for all metrics. For example, PLAR (32.63) outperforms SG-DQG (29.81) under ROUGE-L on Hotpot-30K. Meanwhile, we also find that the generative modeling is essential to the performance gain of PLAR. For examples, PLAR w/o plan (16.64) achieves 25.9% improvements over Pipeline (13.21) under METEOR on Hotpot-10K. This validates that unifying single-hop and multi-hop QA data can effectively alleviate the data scarcity issue. We further find that Pipeline has a heavier performance decrease (comparing with the baselines) when having fewer data. For example, Pipeline outperforms RC-QG under BLEU-4 on Hotpot-30K, while it is outperformed by RC-QG on Hotpot-10K. This is largely because the training of each phase is individual performed which is prone to data insufficiency especially in a more extreme low-resource scenario.

For the paragraph-level input setting, Table 4 shows the results that PLAR consistently outperforms the baselines by a large margin. For example, PLAR (17.79) achieves a gain of more than 33% compared to SD-DQG (13.37) under METEOR on Hotpot-30K. By comparing PLAR (27.64) with PLAN w/o plan (24.81) and Pipeline (23.04) under ROUGE-L on Hotpot-10K, we find that the contribution of the planning mechanism is more significant than that of the generative modeling. This is largely because the diversification of potential sub-questions raises greater challenges in the paragraph-level setting. Using the planning variables, PLAR can effectively generate the feasible sub-questions. We also provide qualitative examples in Appendix to show the effectiveness of the planning variables. We also find that the reasoning path augmented baselines are not as competitive as in the sentence-level input setting. For example, RC-QG outperforms all the seq-to-seq based baselines under METEOR in the sentence-level setting, while it only outperforms ASs2s in the paragraph-level setting. The reason is that handcrafted reasoning features cannot generalize well to a larger evidence set. PLAR overcomes this limitation by optimizing reasoning capability taking advantage of both single-hop and multi-hop QA data.

4.3 Discussion

We first study whether the generated questions can boost the question answering performance. We compare the performances of a BERT QA model (Devlin et al., 2019) on both subsets, where the QA model is trained using QA data generated by different QG models. The results in Table 5 show that the learning of multi-hop QA models relies heavily on sufficient supervision, since a significant perform reduction is observed when training on a subset only. PLAR achieves more effective training than the baselines and its variants, especially in the more challenging subset. It achieves the most performance gain (17.3%) over the subset-only training result under F1 on Hotpot-30K. We also find that the QG results of BART do not improve QA performance while BART performs comparable to other baselines (e.g., SG-DQG) on automatic evaluation metrics. This is aligned with our intuition that the text fluency is insufficient for obtaining multi-hop questions that benefit the QA task. It is essential to incorporate reasoning into the generation process.

We now study the effect of unified generative question generation. To investigate how the generative multi-hop (Eqn. 5) and single-hop objective (Eqn. 7) contribute to the overall question generation training, we compare PLAR with PLAR using multi-hop objective only and PLAR using planning guided multi-hop objective under varying sizes of single-hop question answering data. The results on

Table 4: Question Generation Results (Paragraph-Level)

MODEL		HOTPOT-30K				HOTPOT-10K			
		BLEU-1	BLEU-4	METEOR	ROUGE-L	BLEU-1	BLEU-4	METEOR	ROUGE-L
Seq-to-Seq Application	ASs2s	23.33	8.71	11.01	22.18	22.01	8.82	10.00	20.76
	Maxout-QG	25.04	9.82	12.36	24.51	23.71	9.05	11.28	22.42
	BART	26.13	10.34	13.40	24.97	24.60	9.63	12.37	22.79
Reasoning Path Enhanced	RC-QG	25.75	9.02	12.51	24.08	23.32	8.71	11.08	20.51
	SG-DQG	26.97	10.42	13.37	25.43	24.07	10.03	11.54	22.57
Proposed	Pipeline	27.06	10.43	13.93	25.62	24.41	10.24	11.84	23.04
	PLAR w/o plan	32.27	12.10	15.76	28.48	27.31	11.23	12.06	24.81
	PLAR	**36.74**	**13.63**	**17.79**	**30.35**	**32.58**	**13.32**	**14.18**	**27.64**

Table 5: Question Answering Results using Synthetic Data from Question Generation

QA SUPERVISION		HOTPOT-30K		HOTPOT-10K	
		EM	F1	EM	F1
w/o QG	Subset only	52.2	66.1	46.0	58.9
Subset w/ QG	BART	51.4	64.0	43.1	54.3
	SG-DQG	53.3	67.2	47.3	62.8
	Pipeline	54.8	68.3	49.1	61.7
	PLAR w/o plan	60.9	72.8	53.4	65.0
	PLAR	**61.5**	**73.0**	**54.8**	**66.2**

* Both question generation and question answering are performed in the paragraph-level setting.

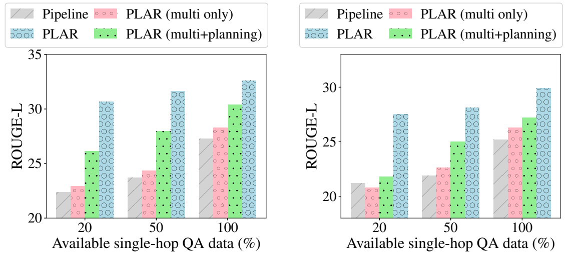

(a) Results on HOTPOT-30K (b) Results on HOTPOT-10K

Figure 2: Effects of unified generative modeling

the two subsets under ROUGE-L in the sentence-level setting are shown in Fig. 2(a) and Fig. 2(b). We can see that both objectives are important. For example, when using complete single-hop QA data on HOTPOT-30K, multi-hop and single-hop generative objectives bring 7.3% and 11.5% improvement, respectively. We further find that the performance gain of PLAR is largely attributed to the single-hop generative objective when available single-hop questions are limited. The reason is that without the generative single-hop objective, training the sub-question generation model heavily relies on the initial fine-tuning step, and is thus prone to single-hop QA data insufficiency. The full PLAR model addresses this limitation by further training the sub-question generation model with supervision from generative single-hop and multi-hop QG.

5 Related Work

Question generation has a wide range of applications besides expanding question answering data, such as initiating a conversation of dialogue systems (Mostafazadeh et al., 2017), providing practice exercises for educational purposes (Jia et al., 2020), and accelerating real-time question answering (Seo et al., 2019). It also has great potential in enriching task-oriented dialogue datasets (Sun et al., 2016, 2017; Huang et al., 2020a; Kim et al., 2020b). Early studies build on encoder-decoder models and utilize different evidence information, e.g., Wikipedia passages (Du and Cardie, 2018), reviews (Yu et al., 2020c), and dialogue history (Gao et al., 2019). These studies often assume that the questions are single-hop which be answered by one piece of evidence. As more high-quality multi-hop question answering datasets become available (e.g., HOTPOTQA (Yang et al., 2018)), recent years have seen a growing interest in multi-hop question generation. Most recent approaches add heuristically extracted features to the encoder-decoder model, which relies on large-scale training data and can still suffer from error propagation (Yu et al., 2020b; Pan et al., 2020). A recent study (Yu et al., 2020a) which also studies low-resource question generation assumes that a large amount of unanswered multi-hop questions are available, which is also difficult to obtain. We aim to overcome these limitations in this study.

Our study is also related to generative modeling which treats unobserved variables (e.g., features or labels) as latent variables, and approximates the distribution through variational inference (Kingma and Welling, 2014). Generative modeling has been applied to dialogue response generation (Zhao et al., 2019; Huang et al., 2020b; Yang et al., 2020), policy learning (Huang et al., 2019, 2020c), sentiment analysis (Xu et al., 2017; Li et al., 2019), knowledge retrieval (Lee et al., 2019; Kim et al., 2020a; Su et al., 2021; Tan et al., 2021), and text style transfer (He et al., 2020). While these works focus on utilizing unlabeled data to boost model performance, we aim to unify non-parallel question corpuses to enable joint learning.

6 Conclusions

We proposed a jointly optimized two-phase model named PLAR for low-resource question generation. PLAR effectively utilizes non-parallel single-hop and multi-hop question answering data to perform optimization. We further designed a planning mechanism to guide the generation process of sub-questions so that the generation results are valid to compose a multi-hop question. Experimental results confirm that PLAR achieves better performance compared with the state-of-the-art under various metrics, especially in a question answering based evaluation. For future work, we will explore the heterogeneous multi-hop QG task that requires reasoning beyond plain texts, e.g., tables.

Acknowledgement

This work is supported by Australian Research Council (ARC) Discovery Project DP180102050.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from Wikipedia. In *Annual Meeting of the Association for Computational Linguistics*, pages 1907–1917.

Yifan Gao, Piji Li, Irwin King, and Michael R. Lyu. 2019. Interconnected question generation with coreference alignment and conversation flow modeling. In *Annual Meeting of the Association for Computational Linguistics*, pages 4853–4862.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. In *International Conference on Learning Representations*.

Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2020a. Generalizable and explainable dialogue generation via explicit action learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3981–3991, Online. Association for Computational Linguistics.

Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2020b. MALA: Cross-domain dialogue generation with action learning. In *AAAI Conference on Artificial Intelligence*, pages 7977–7984.

Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2020c. Semi-supervised dialogue policy learning via stochastic reward estimation. In *Annual Meeting of the Association for Computational Linguistics*, pages 660–670.

Xinting Huang, Jianzhong Qi, Yu Sun, Rui Zhang, and Hai-Tao Zheng. 2019. Carl: Aggregated search with context-aware module embedding learning. In *IJCNN*, pages 101–108. IEEE.

X. Jia, Wenjie Zhou, Xu Sun, and Yunfang Wu. 2020. Eqg-race: Examination-type question generation. In *AAAI Conference on Artificial Intelligence*.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020a. Sequential latent knowledge selection for knowledge-grounded dialogue. In *International Conference on Learning Representations*.

Seokhwan Kim, Mihail Eric, Karthik Gopalakrishnan, Behnam Hedayatnia, Yang Liu, and Dilek Hakkani-Tur. 2020b. Beyond domain APIs: Task-oriented conversational modeling with unstructured knowledge access. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 278–289, 1st virtual meeting. Association for Computational Linguistics.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and K. Jung. 2019. Improving neural question generation using answer separation. In *AAAI Conference on Artificial Intelligence*, pages 6602–6609.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Wei-Jen Ko, Te-yuan Chen, Yiyan Huang, Greg Durrett, and Junyi Jessy Li. 2020. Inquisitive question generation for high level text comprehension. In *Conference on Empirical Methods in Natural Language Processing*, pages 6544–6555.

Dong Bok Lee, Seanie Lee, Woo Tae Jeong, Donghwan Kim, and Sung Ju Hwang. 2020. Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional VAEs. In *Annual Meeting of the Association for Computational Linguistics*, pages 208–224.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2019. Semi-supervised stochastic multi-domain learning using variational inference. In *Annual Meeting of the Association for Computational Linguistics*, pages 1923–1934.

Chin-Yew Lin and Eduard Hovy. 2002. Manual and automatic evaluation of summaries. In *ACL-02 Workshop on Automatic Summarization*, page 45–51.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109.

N. Mostafazadeh, Chris Brockett, W. Dolan, Michel Galley, Jianfeng Gao, Georgios P. Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *International Joint Conference on Natural Language Processing*, pages 462–472.

Aaron van den Oord, Oriol Vinyals, and Kavukcuoglu Koray. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic graphs for generating deep questions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1463–1475.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Conference on Empirical Methods in Natural Language Processing*, pages 8864–8880.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*, pages 2383—-2392.

Sudha Rao and Hal Daumé III. 2019. Answer-based Adversarial Training for Generating Clarification Questions. In *the North American Chapter of the Association for Computational Linguistics*, pages 143–155.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441.

Yixin Su, Rui Zhang, S. Erfani, and Zhenghua Xu. 2021. Detecting beneficial feature interactions for recommender systems. In *AAAI Conference on Artificial Intelligence*, pages 1021–1029.

Yu Sun, Nicholas Jing Yuan, Yingzi Wang, Xing Xie, Kieran McDonald, and Rui Zhang. 2016. Contextual intent tracking for personal assistants. In *SIGKDD*, pages 273–282.

Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. 2017. Collaborative intent prediction with real-time contextual data. *ACM Transactions on Information Systems (TOIS)*, 35(4):1–33.

Yanchao Tan, Carl Yang, Xiangyu Wei, Y. Ma, and X. Zheng. 2021. Multi-facet recommender networks with spherical optimization. *ArXiv*, abs/2103.14866.

Arash Vahdat and J. Kautz. 2020. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *AAAI Conference on Artificial Intelligence*, pages 3358–3364.

Shiquan Yang, Rui Zhang, and Sarah Erfani. 2020. Graphdialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems. In *Conference on Empirical Methods in Natural Language Processing*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Jianxing Yu, Wei Liu, Shuang Qiu, Qinliang Su, Kai Wang, Xiaojun Quan, and Jian Yin. 2020a. Low-resource generation of multi-hop reasoning questions. In *Annual Meeting of the Association for Computational Linguistics*, pages 6729–6739.

Jianxing Yu, Xiaojun Quan, Qinliang Su, and Jian Yin. 2020b. Generating multi-hop reasoning questions to improve machine reading comprehension. *The Web Conference*, pages 281–291.

Qian Yu, Lidong Bing, Qiong Zhang, Wai Lam, and Luo Si. 2020c. Review-based question generation with adaptive instance transfer and augmentation. In *Annual Meeting of the Association for Computational Linguistics*, pages 280–290.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. In *North American Chapter of the Association for Computational Linguistics*, pages 1208–1218.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.

A Implementation Details

We use HotpotQA split as the original paper (Yang et al., 2018) [4], and use SQuAD v1.1 (Rajpurkar et al., 2016) [5] training set only since single-hop question answering data only involves in the training. We use the BART-base model implementation from huggingface library [6] as the single-hop question generation model and question composing model. We set the batch size to 32 in sentence-level setting and 16 in paragraph-level setting. The models are trained by Adam (Kingma and Ba, 2015) with a learning rate initially set to 3e-5 on NVIDIA GeForce RTX 2080 Ti. We use grid search to find the best hyperparameters for the models based on validation performance, which we use a combination of METEOOR, ROUGE-L and BLEU scores to measure. [7] We set dimensionality of codebook of the planning mechanism (i.e., K) to 100, which is chosen among $\{50, 75, 100, 150, 200\}$

B Sub-Question Generation Qualitative Analysis

Table 6 and 7 show question generation results from PLAR and Pipeline model.

C Planning Mechanism Case Study

Table 8 and 9 show the generation results by different sampled planning variables z in the paragraph-level setting. We can see that with different predicted z (denoted by different z_i), PLAR raises different sub-questions and presents different high-level reasoning type. We also find that some planning variable cannot lead to a reasonable multi-hop question, and the prediction of PLAR can well capture the correct plan (denoted by higher $p(z|a, c)$).

Table 6: Sub-Question Generation and Question Composing Examples.

Supportive Evidence	**Paragraph A. Dario Franchitti**
	[1] George Dario Marino Franchitti, MBE (born 19 May 1973), known professionally as Dario Franchitti, is a retired Scottish racing driver. [2] After Franchitti did not secure a single-seater drive in 1995, he was contracted by the AMG team to compete in touring cars in the DTM and its successor — International Touring Car Championship.
	Paragraph B. Mercedes-AMG
	[1] Mercedes-AMG GmbH, commonly known as AMG, is the high performance division of Mercedes-Benz. [2] AMG independently hires engineers, manufactures and customizes Mercedes-Benz AMG vehicles. [3] Mercedes-AMG is headquartered in Affalterbach, Baden Württemberg, Germany.
Groundtruth QA Pair	After he was contracted by the team that is headquartered in Affalterbach, Baden-Württemberg, Germany, Dario Franchitti competed in what series? (Answer: DTM)
PLAR QG results	**Sub-question 1:** Affalterbach Germany is the location of what team?
	Sub-question 2: After contracted by AMG, Dario Franchitti competed in what series?
	Multi-hop question: After he was contracted by the team that is headquartered in Affalterbach, Baden-Württemberg, Germany, Dario Franchitti competed in what series?
Pipeline QG results	**Sub-question 1:** What is headquartered in Affalterbach Baden Germany?
	Sub-question 2: Dario Franchitti competed in what series in 1995?
	Multi-hop question: Dario Franchitti competed in what series in 1995 in Affalterbach Baden Germany?

---

[4] https://hotpotqa.github.io/

[5] https://rajpurkar.github.io/SQuAD-explorer/

[6] https://huggingface.co/facebook/bart-base

[7] We use the implementations of the metrics as https://github.com/Maluuba/nlg-eval

Table 7: Sub-Question Generation and Question Composing Examples

Supportive Evidence	**Paragraph A. Goran Dragić**
	[1] Goran Dragić (born 6 May 1986) is a Slovenian professional basketball for the Miami Heat of the National Basketball Association (NBA). [2] He plays at both the point guard and shooting guard positions.
	Paragraph B. 2013–14 Phoenix Suns season
	[1] The 2013–14 NBA season was the Phoenix Suns' 46th season in the NBA. [2] When the Suns began the regular season, Goran Dragić, P. J. Tucker, Markieff Morris, and his twin brother Marcus Morris were the only players returning from playing with last season's team (while Channing Frye was still on last season's team, he didn't play any games due to a life-threatening heart ailment he had at the time).
Groundtruth QA Pair	Which team's 2013-2014 season had players including a Slovenian who plays at both the point guard and shooting guard positions? (Answer: the Phoenix Suns)
PLAR QG results	**Sub-question 1:** Which Slovenian player plays at the point guard and shooting guard position?
	Sub-question 2: In 2013 NBA season which team have the player?
	Multi-hop question: In 2013 NBA season which team have the Slovenian player which plays at the point guard and shooting guard position?
Pipeline QG results	**Sub-question 1:** Goran Dragić plays at what positions?
	Sub-question 2: What team begins season with players returning from last season?
	Multi-hop question: Goran Dragić plays at what positions in the team with players returning from last season?

Table 8: Generation Results of Different Sampled Planning Variables

Supportive Evidence	**Paragraph A. Koyaanisqatsi**
	[1] Koyaanisqatsi, also known as Koyaanisqatsi: Life Out of Balance, is a 1982 American experimental film directed by Godfrey Reggio with music composed by Philip Glass and cinematography by Ron Fricke.
	Paragraph B. Mad Hot Ballroom
	[1] Mad Hot Ballroom is a 2005 American documentary film directed and co-produced by Marilyn Agrelo and written and co-produced by Amy Sewell, about a ballroom dance program in the New York City Department of Education, the New York City public school system for fifth graders. [2] Several styles of dance are shown in the film, such as tango, foxtrot, swing, rumba and merengue.
Groundtruth QA Pair	Which film was created more recently, Koyaanisqatsi or Mad Hot Ballroom? (Answer: Mad Hot Ballroom)
PLAR w/ Planning z_1 $p(z_1 \| c,a)=0.52$	**Sub-question 1:** Which year is film Koyaanisqatsi created?
	Sub-question 2: Which year is film Mad Hot Ballroom created?
	Multi-hop question: Film Koyaanisqatsi and Mad Hot Ballroom, which is created later?
PLAR w/ Planning z_2 $p(z_2 \| c,a)=0.33$	**Sub-question 1:** What is the name of file directed by Marilyn Agrelo?
	Sub-question 2: What 1982 experimental film Godfrey Reggio directed?
	Multi-hop question: Are Koyaanisqatsi and Mad Hot Ballroom by the same director?
PLAR w/ Planning z_3 $p(z_3 \| c,a)=0.10$	**Sub-question 1:** What movie has music by Philip Glass?
	Sub-question 2: What movie shows dance styles such as tango, foxtrot?
	Multi-hop question: What movie has music by Philip Glass and dance styles such as tango, foxtrot?

Table 9: Generation Results of Different Sampled Planning Variables

Supportive Evidence	**Paragraph A. Force India VJM10**
	[1] The Force India VJM10 is a Formula One racing car designed and constructed by Force India to compete during the 2017 Formula One season. [2] The car is driven by Sergio Pérez and Esteban Ocon, who joined the team after Nico Hülkenberg left the team at the end of the season.
	Paragraph B. Esteban Ocon
	[1] Esteban Ocon (born 17 September 1996) is a French racing driver who currently drives in Formula One for Force India. [2] He made his Formula One debut for Manor Racing in the 2016 Belgian Grand Prix, replacing Rio Haryanto. [3] Ocon is part of the Mercedes-Benz driver development programme.
Groundtruth QA Pair	Force India VJM10 is a Formula One racing car previous driven by Nico Hülkenberg, and is now driven by which driver born 17 September 1996? (Answer: Esteban Ocon)
PLAR w/ Planning z_1 $p(z_1 \mid c,a)=0.71$	**Sub-question 1:** who drives Force India VJM10, a Formula One racing car previous driven by Nico Hülkenberg?
	Sub-question 2: which driver is born 17 September 1996?
	Multi-hop question: who is born 17 September 1996 and drives Force India VJM10, a Formula One racing car previous driven by Nico Hülkenberg?
PLAR w/ Planning z_2 $p(z_2 \mid c,a)=0.11$	**Sub-question 1:** Who currently drives in Fomula One for Force India?
	Sub-question 2: Who joined Force India after Nico Hülkenberg left?
	Multi-hop question: Who joined and drives for Force India after Nico Hülkenberg left?

Probing Pre-Trained Language Models for Disease Knowledge

Israa Alghanmi, Luis Espinosa-Anke, Steven Schockaert
Cardiff University, United Kingdom
{alghanmiia,espinosa-ankel,schockaerts1}@cardiff.ac.uk

Abstract

Pre-trained language models such as ClinicalBERT have achieved impressive results on tasks such as medical Natural Language Inference. At first glance, this may suggest that these models are able to perform medical reasoning tasks, such as mapping symptoms to diseases. However, we find that standard benchmarks such as MedNLI contain relatively few examples that require such forms of reasoning. To better understand the medical reasoning capabilities of existing language models, in this paper we introduce DisKnE, a new benchmark for Disease Knowledge Evaluation. To construct this benchmark, we annotated each positive MedNLI example with the types of medical reasoning that are needed. We then created negative examples by corrupting these positive examples in an adversarial way. Furthermore, we define training-test splits per disease, ensuring that no knowledge about test diseases can be learned from the training data, and we canonicalize the formulation of the hypotheses to avoid the presence of artefacts. This leads to a number of binary classification problems, one for each type of reasoning and each disease. When analysing pre-trained models for the clinical/biomedical domain on the proposed benchmark, we find that their performance drops considerably.

1 Introduction

Pre-trained language models (LMs) such as BERT (Devlin et al., 2019) are currently the de-facto architecture for solving most NLP tasks, and their prevalence in general language understanding tasks is today indisputable (Wang et al., 2018, 2019). Beyond generic benchmarks, it has been shown that LMs are also extremely powerful in domain-specific NLP tasks, e.g., in the biomedical domain (Lewis et al., 2020). While there are several reasons why they are preferred over standard neural architectures, one important (and perhaps less obvious) reason is that LMs capture a substantial amount of world knowledge. For instance, several authors have found that LMs are able to answer questions without having access to external resources (Petroni et al., 2019; Roberts et al., 2020), or that they exhibit commonsense knowledge (Forbes et al., 2019; Davison et al., 2019). To analyze the capabilities of LMs in a more systematic way, there is a growing interest in designing probing tasks, which are now common across the NLP landscape, e.g., for word and sentence-level semantics (Paperno et al., 2016; Conneau et al., 2018). In this paper we focus on (generic and specialized) LMs in the biomedical domain, and ask the following question: what kinds of medical knowledge do pre-trained LMs capture? More specifically, we focus on *disease knowledge*, which encompasses for instance the ability to link symptoms to diseases, or treatments to diseases.

Among the several biomedical LMs (i.e. LMs that have been pre-trained on biomedical text corpora) that exist today, some of the most prominent are SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2020) and ClinicalBERT (Alsentzer et al., 2019). Rather than architectural features, these models differ from each other mostly in the pre-training corpora: SciBERT was trained from scratch on scientific papers; BioBERT is an adapted version of BERT (Devlin et al., 2019), which was fine-tuned on PubMed articles as well as some full text biomedical articles; and ClinicalBERT was initialized from BioBERT and further fine-tuned on MIMIC-III notes (Johnson et al., 2016), which are clinical notes describing patients admitted to critical care units. These LMs have enabled impressive results on various reading comprehension benchmarks for the medical domain, such as MedNLI (Romanov and Shivade, 2018) and MEDIQA-NLI (Abacha et al., 2019) for Natural Language Infer-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3023–3033
August 1–6, 2021.

ence (NLI), and PubMedQA (Jin et al., 2019b) for QA. As an example, Wu et al. (2019) achieved an accuracy of 98% on MEDIQA-NLI, which might suggest that medical NLI is essentially a solved problem. This would be exciting, as medical NLI intuitively requires a wealth of medical knowledge, much of which is not available in structured form.

However, a closer inspection of MedNLI, the most well-known medical NLI benchmark, reveals three important limitations, namely: (1) only few test instances actually require *medical disease* knowledge, with instances that (only) require terminological and lexical knowledge (e.g. understanding acronyms or paraphrases) being more prevalent; (2) training and test examples often cover the same diseases, and thus it cannot be determined whether good performance comes from the capabilities of the pre-trained LM itself, or from the fact that the model can exploit similarities between training and test examples; and (3) hypothesis-only baselines perform rather well on MedNLI, which shows that this benchmark has artefacts that can be exploited, similarly to general-purpose NLI benchmarks (Poliak et al., 2018).

We therefore propose DisKnE (Disease Knowledge Evaluation), a new benchmark for evaluating biomedical LMs. This dataset explicitly addresses the three limitations listed above and thus constitutes a more reliable testbed for evaluating the disease knowledge captured by biomedical LMs. DisKnE is derived from MedNLI and is organized into two top-level categories, which cover instances requiring medical and terminological knowledge respectively. The medical category is furthermore divided into four sub-categories, depending on the type of medical knowledge that is required.

We empirically analyse the performance of existing biomedical LMs, as well as the standard BERT model, on the proposed benchmark. Our results show that all the considered LMs struggle with NLI examples that require medical knowledge. We also find that the relative performance of the pre-trained models differs across medical categories, where the best performance is obtained by ClinicalBERT, BioBERT, SciBERT or BERT depending on the category and experimental setting. Conversely, for examples that are based on terminological knowledge, overall performance is much higher, with relatively little difference between different pre-trained models. The contributions of this paper are as follows[1]:

- We introduce a new benchmark to assess the disease-centred knowledge captured by pre-trained LMs, organised into categories that reflect the type of reasoning that is needed, and with training-test splits that avoid leakage of disease knowledge.

- We analyze the performance of several clinical/biomedical BERT variants on each of the considered categories. We find that all considered models struggle with examples that require medical disease knowledge.

- We find that without canonicalizing the hypotheses, hypothesis-only baselines achieve the best results in some categories. This shows that the original MedNLI dataset suffers from annotation artefacts, even within the set of entailment examples.

2 Related Work & Background

Knowledge Encoded in LMs There is a rapidly growing body of work that is focused on analyzing what knowledge is captured by pre-trained LMs. A recurring challenge in such analyses is to separate the knowledge that is already captured by a pre-trained model from the knowledge that it may acquire during a task-specific fine-tuning step. A common solution to address this is to focus on zero-shot performance, i.e. to focus on tasks that require no fine-tuning, such as filling in a blank (Davison et al., 2019; Talmor et al., 2020). As an alternative strategy, Talmor et al. (2020) propose to analyse the performance of models that were fine-tuned on a small training set. Other work has focused on extracting structured knowledge from pre-trained LMs. Early approaches involved manually designing suitable prompts for extracting particular types of relations (Petroni et al., 2019). Recently, however, several authors have proposed strategies that automatically construct such prompts (Bouraoui et al., 2020; Jiang et al., 2020; Shin et al., 2020). Finally, Bosselut et al. (2019) proposed to fine-tune LMs on knowledge graph triples, with the aim of then using the model to generate new triples.

---

[1] All code for reconstructing the dataset and replicating the experiments is available at: `https://github.com/israa-alghanmi/DisKnE`. License and access to MedNLI, MEDIQA-NLI and UMLS will be needed.

LMs for Biomedical Text As already mentioned in the introduction, a number of pre-trained LMs have been released for the biomedical domain. Several authors have analyzed the performance of these models, and the impact of including different types of biomedical corpora in particular. For instance, Peng et al. (2019) proposed an evaluation framework for biomedical language understanding (BLUE). They obtained the best results with a BERT model that was pre-trained on PubMed abstracts and MIMIC-III clinical notes. Another large-scale evaluation of biomedical LMs has been carried out by Lewis et al. (2020). To evaluate the biomedical knowledge that is captured in pre-trained LMs, as opposed to acquired during training, Jin et al. (2019a) freeze the transformer layers during training. They find that when biomedical LMs are thus used as fixed feature extractors, BioELMo outperforms BioBERT. Most closely related to our work, He et al. (2020) recently also highlighted the limited ways in which biomedical LMs capture disease knowledge. To address this, they proposed a pre-training objective which relies on a weak supervision signal, derived from the structure of Wikipedia articles about diseases. Other authors have suggested to include structured knowledge, e.g. from UMLS, during the pre-training stage of BERT-based models (Michalopoulos et al., 2020; Hao et al., 2020). Another strategy is to inject external knowledge into task-specific models (rather than at the pre-training stage), for instance in the form of definitions (Lu et al., 2019) or again UMLS (Sharma et al., 2019). Kearns et al. (2019) presented a related approach to our work in which they categorize each sentence pair according to the tense and focus (e.g. medication, diseases, procedures, location) of the hypothesis, with the aim of providing a detailed examination of MEDIQA-NLI. Based on this categorization, they compare the performance of Enhanced Sequential Inference Model (ESIM) using ClinicalBERT, Embeddings of Semantic Predications (ESP), and cui2vec. However, their analysis was limited to the MEDIAQ-NLI test set, whereas we include entailment examples from the entire MedNLI and MEDIQA-NLI datasets. Moreover, we focus specifically on the ability of LMs to distinguish between closely related diseases, and we move away from the NLI setting to avoid training-test leakage and artefacts.

Adversarial NLI Several Natural Language Inference (NLI) benchmarks have been found to contain artefacts that can be exploited by NLP systems to perform well without actually solving the intended task (Poliak et al., 2018; Gururangan et al., 2018). In particular, it has been found that strong results can often be achieved by only looking at the hypothesis of a (premise, hypothesis) pair. In response to this finding, several strategies for creating harder NLI benchmarks have been proposed. One established approach is to create adversarial stress tests (Naik et al., 2018; Glockner et al., 2018; Aspillaga et al., 2020), in which synthetically generated examples are created to specifically test for phenomena that are known to confuse NLI models. This may, for instance, involve the use of WordNet to obtain nearly identical premise and hypothesis sentences, in which one word is replaced by an antonym or co-hyponym. In this paper, we rely on a somewhat similar strategy, using UMLS to replace diseases in hypotheses. As another strategy to obtain hard NLI datasets, Nie et al. (2020) used human annotators to iteratively construct examples that are incorrectly labelled by a strong baseline model. While the aforementioned works are concerned with open-domain NLI, some work on creating adversarial datasets for the biomedical domain has also been carried out. In particular, Araujo et al. (2020) studied the robustness of systems for biomedical named entity recognition and semantic text similarity, by introducing misspellings and swapping disease names by synonyms. To the best of our knowledge, no adversarial NLI datasets for the biomedical domain have yet been proposed.

3 Dataset Construction

In this section, we describe the process we followed for constructing DisKnE. As we explain in more detail in Section 3.1, this process involved filtering the entailment instances from the MedNLI and MEDIQA-NLI datasets, to select those in which the hypothesis expresses that the patient has (or is likely to have) a particular target disease. These instances were then manually categorized based on the type of knowledge that is needed for recognizing the validity of the entailment. Section 3.2 discusses our strategy for generating negative examples, which were obtained in an adversarial way, by replacing diseases occurring in entailment examples with similar ones. Details of the resulting training-test splits are provided in Section 3.3. In a

Category	# inst.	Premise	Hypothesis
Symptoms → Disease	112	The patient developed neck pain while training with increasing substernal heaviness and left arm pain together with sweating.	The patient has symptoms of acute coronary syndrome
Treatments → Disease	60	The patient started on Mucinex and Robitussin.	The patient has sinus disease
Tests → Disease	116	Cardiac enzymes recorded CK 363, CK-MB 33, TropI 6.78	The patient has cardiac ischemia
		A large R hemisphere ICH was revealed when the patent had head CT	The patient has an aneurysm
Procedures → Disease	70	Bloody fluid was removed by pericardiocentesis	The patient has hemopericardium.
Terminological	259	The patient has urinary tract infection	The patient has a UTI
		The patient has high blood pressure	Hypertension
		Transfusions in the past could be the cause of the patient having hepatitis C	The patient has hepatitis C

Table 1: Considered categories of disease-focused entailment pairs.

final step, we canonicalize the hypotheses of all examples, as explained in Section 3.4. Note that the benchmark we propose consists of binary classification problems (i.e. predicting entailment or not), rather than the standard ternary NLI setting (i.e. predicting entailment, neutral, or contradiction), which is motivated by the fact that natural contradiction examples are hard to find when focusing on disease knowledge.

3.1 Selecting Entailment Pairs

We started from the set of all entailment pairs (i.e. premise-hypothesis pairs labelled with the *entailment* category) from the full MedNLI and MEDIQA-NLI datasets. We used MetaMap to find those pairs whose hypothesis mentions the name of a disease, and to retrieve the UMLS CUI (Concept Unique Identifier) code corresponding to that disease. We then manually identified those pairs, among the ones whose hypothesis mentions a disease, in which the hypothesis specifically expresses that the patient has that disease. For instance, in this step, a number of instances were removed in which the hypothesis expresses that the patient does not have the disease. The remaining cases were manually assigned to categories that reflect the type of disease knowledge that is needed to identify that the hypothesis is entailed by the premise. The considered categories are described in Table 1, which also shows the number of (positive) examples we obtained and illustrative examples[2]. The primary distinction we make is between examples that need medical knowledge and those that need terminological knowledge. The former category is divided into four sub-categories, depending on the type of inference that is needed. First, we have the *symptoms-to-disease* category, containing examples where the premise describes the signs or symptoms exhibited by the patient, and the hypothesis mentions the corresponding diagnosis. Second, we have the *treatments-to-disease* category, where the premise instead describe medications (or other treatments followed by the patient). The third category, *tests-to-disease*, involves instances where the premise describes lab tests and diagnostic tools such as X-rays, CT scans and MRI. Finally, the *procedures-to-disease* category has instances where the premise describes surgeries and therapeutic procedures that the patient underwent. In the *terminological* category, the disease is mentioned in both the premise and hypothesis, either as an abbreviation, a synonym or within a rephrased sentence.

3.2 Generating Examples

The process outlined in Section 3.1 only provides us with positive examples. Unfortunately, MedNLI and MEDIQA-NLI contain only few negative examples (i.e. instances of the *neutral* or *contradiction* categories) in which the hypothesis expresses that the patient has some disease. For this reason, rather than selecting negative examples from these datasets, we generate negative examples by corrupting the positive examples. In particular, to generate negative examples, we replace the disease

---

[2] For data protection reasons, we only provide synthetic examples, which are different from but similar in spirit to those from the original MedNLI dataset.

X from a given positive example by other diseases $Y_1, ..., Y_n$ that are similar to X, but not ancestors or descendants of X in SNOMED CT (Donnelly et al., 2006). To identify similar diseases, we have relied on cui2vec (Beam et al., 2020), a pre-trained clinical concept embedding that was learned from a combination of insurance claims, clinical notes and biomedical journal articles. Apart from the requirement that the diseases $Y_1, ..., Y_n$ should be similar to X, it is also important that they are sufficiently common diseases, as including unusual diseases would make the corresponding negative examples too easy to detect. For this reason, we only consider the diseases that occur in the hypothesis of other positive examples as candidates for the negative examples. Specifically, among these set of candidate diseases, we selected the $n = 10$ most similar ones to X, which were not descendants or ancestors of X in SNOMED CT (as ancestors and descendants would not necessarily invalidate the entailment). This resulted in a total of 4133 examples requiring medical knowledge and 2639 examples requiring terminological knowledge.

3.3 Training-Test Splits

Because our focus is on evaluating the knowledge captured by pre-trained language models, we want to avoid overlap in the set of diseases in the training and test splits. In other words, if the model is able to correctly identify positive examples for a target disease X, this should be a reflection of the knowledge about X in the pre-trained model, rather than knowledge that it acquired during training. However, any single split into training and test diseases would leave us with a relatively small dataset. For this reason, we consider each disease X in isolation. Let $\mathcal{E}$ be the set of all positive examples, obtained using the process from Section 3.1. Furthermore, we write $\mathcal{E}_X$ for the set of those examples from $\mathcal{E}$ in which the target disease in the hypothesis is X. Finally, we write $neg(X)$ for the set $\{Y_1, ..., Y_n\}$ of associated diseases that was selected to construct negative examples, following the process from Section 3.2.

For each target disease X, we define a corresponding test set Test_X and training set Train_X as follows. Test_X contains all the positive examples from $\mathcal{E}_X$. Moreover, for each $e \in \mathcal{E}_X$ and each $Y \in neg(X)$ we add a negative example $e_{X \to Y}$ to Test_X which is obtained by replacing the occurrence of X by Y. If the word before the occurrence of X is *a* or *an*, we modify it depending on whether Y starts with a vowel or consonant. The positive examples in Train_X consist of all examples from $\mathcal{E}$ in which X is not mentioned. Note that we also remove examples in which these diseases are only mentioned in the premise. Furthermore, we check for occurrences of all the synonyms of these diseases that are listed in UMLS. The process of creating the training and test set for a given target disease X is illustrated in Figure 1.

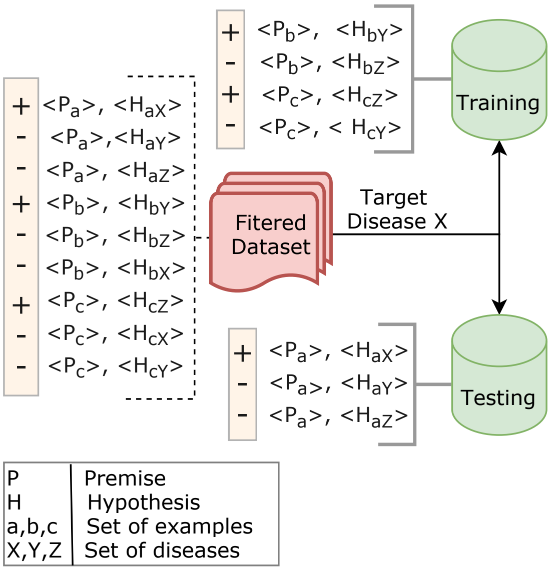

Figure 1: Illustration of training-test splitting process.

3.4 Canonicalization

We noticed that the way in which a given hypothesis expresses that "the patient has disease X" is correlated with the type of the disease. For this reason, as a final step, we canonicalize the hypotheses in the dataset. Specifically, we replace each hypothesis by the name of the corresponding disease X. Several hypotheses in the dataset already have this form. By converting the other hypotheses in this format, we eliminate any artefacts that are present in their specific formulation.

4 Experiments

We experimentally compare a number of pre-trained biomedical LMs on our proposed DisKnE benchmark. In Section 4.1, we first describe the considered LMs and the experimental setup. The main results are subsequently presented in Section 4.2. This is followed by a discussion in Section 4.3.

	ClinicalBERT	**BioBERT**	**SciBERT**	**BERT**
coronary atherosclerosis	0	0	29	10
chf	67	67	67	67
acs	04	33	0	05
stroke	80	56	90	90
heart disease	80	87	93	100
myocardial infarction	0	0	19	0
heart failure	0	0	22	0
urinary tract infection	100	100	67	100
disorder of lung	89	97	97	100
cirrhosis of liver	0	11	0	0
hyperglycemic disorder	27	13	22	0
pneumonia	89	93	67	100
neurological disease	67	67	80	67
respiratory failure	87	70	22	43
pulmonary edema	74	25	0	50
ami	0	0	0	0
deep vein thrombosis	47	48	50	48
acute cardiac ischemia	0	45	17	72
uri	78	45	67	83
cholangitis	22	22	33	22
atherosclerosis	66	0	67	0
Macro-average	$46_{\pm 3.0}$	$42_{\pm 7.3}$	$43_{\pm 3.1}$	$46_{\pm 3.4}$
Weighted average	$49_{\pm 3.1}$	$47_{\pm 6.0}$	$49_{\pm 2.7}$	$51_{\pm 2.7}$

Table 2: Results for the *Symptoms* → *Disease* category in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

	ClinicalBERT	**BioBERT**	**SciBERT**	**BERT**
chf	55	55	53	55
acs	12	19	0	0
hypertensive disorder	55	67	54	22
heart disease	45	22	0	89
urinary tract infection	100	100	100	100
disorder of lung	82	89	100	93
hyperglycemic disorder	100	69	87	69
pneumonia	60	67	78	57
anemia	17	17	45	22
renal insufficiency	69	89	67	72
pulmonary infection	82	77	89	83
copd	45	67	61	39
hyperlipidemia	59	61	61	55
Macro-average	$60_{\pm 6.1}$	$61_{\pm 1.4}$	$61_{\pm 3.8}$	$58_{\pm 1.6}$
Weighted average	$51_{\pm 5.3}$	$54_{\pm 1.6}$	$51_{\pm 1.7}$	$45_{\pm 2.4}$

Table 3: Results for the *Treatments* → *Disease* category in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

4.1 Experimental Setup

Pre-trained LMs. To understand to what extent the pretraining data of an LM affects its performance on our fine-grained evaluation of disease knowledge, we used the following BERT variants:

BERT We use the BERT$_{\text{base}}$-cased model (Devlin et al., 2019).

BioBERT Lee et al. (2019) proposed a model based on BERT$_{\text{base}}$-cased, which they further trained on biomedical corpora. We use the version where PubMed and PMC were utilized for this further pre-training.

ClinicalBERT Alsentzer et al. (2019) introduced four BERT model variants, trained on various clinical corpora. We use the version that was initialized from BioBERT and trained on MIMIC-III notes afterwards.

SciBERT Beltagy et al. (2019) introduced a BERT model variant that was trained from scratch on approximately 1.14M scientific papers from semantic scholar, 82% of which were biomedical articles. The full text of the papers was used for training. We use the cased version.

Training Details. For fine-tuning, model hyperparameters were the same across all BERT variants such as the random seeds, batch size and the learning rate. In this study, we fix the the learning rate at 2e-5, batch size of 8 and we set the maximum number of epochs to 8 with the use of early stopping. We used 10% of the training set as validation split.

Evaluation Protocol. We analyze the results per disease and per category in terms of F1 score for the positive class, reporting results for all diseases that have at least two positive examples for the considered category. To this end, for each disease X, we start from its corresponding training-test split, which was constructed as explained in Section 3.3. To show the results for a particular category, we remove from the test set all the examples that do not belong to that category.

4.2 Results

The main results are shown in Tables 2–6. A number of clear observations can be made. First, the results for the terminological category are substantially higher than the results for the other categories, which suggests that the masked language modelling

	ClinicalBERT	**BioBERT**	**SciBERT**	**BERT**
coronary atherosclerosis	0	0	0	0
chf	52	55	52	55
acs	0	22	0	0
stroke	87	87	95	77
hypertensive disorder	09	26	45	21
myocardial infarction	28	0	30	14
heart failure	0	55	40	0
urinary tract infection	87	90	59	90
hyperglycemic disorder	81	10	68	33
pneumonia	100	100	89	89
anemia	0	0	24	0
aortic valve stenosis	11	24	0	27
syst. inflam. resp. syndr.	76	64	80	80
acute renal failure syndr.	0	0	0	22
chronic renal insufficiency	0	0	0	0
kidney disease	22	0	45	0
ischemia	93	100	93	100
Macro-average	$38_{\pm 2.4}$	$37_{\pm 1.6}$	$42_{\pm 3.1}$	$36_{\pm 5.0}$
Weighted average	$31_{\pm 2.6}$	$32_{\pm 1.2}$	$37_{\pm 1.5}$	$31_{\pm 3.7}$

Table 4: Results for the *Tests* → *Disease* category in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

	ClinicalBERT	**BioBERT**	**SciBERT**	**BERT**
coronary atherosclerosis	0	0	16	0
heart disease	83	74	84	84
heart failure	33	33	50	0
cirrhosis of liver	0	0	0	0
end stage renal disease	37	29	70	79
respiratory failure	58	27	57	27
renal insufficiency	100	100	93	100
cardiac arrest	100	100	93	100
disorder of resp. syst.	76	80	80	71
peripheral vascular dis.	0	0	78	0
Macro-average	$49_{\pm 3.2}$	$44_{\pm 5.9}$	$62_{\pm 3.9}$	$46_{\pm 5.0}$
Weighted average	$40_{\pm 3.3}$	$36_{\pm 7.4}$	$55_{\pm 5.6}$	$44_{\pm 4.6}$

Table 5: Results for the *Procedures* → *Disease* category in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

objective, which is used as the main pre-training task in all the considered LMs, may not be ideally suited for learning medical knowledge. Second, recall that the main difference between the considered biomedical LMs comes from the corpora that were used for pre-training them. As the results for the terminological category (Table 6) reveal, the inclusion of domain-specific corpora does not seem to benefit their ability to model biomedical terminology, as similar results for this category are obtained with the standard BERT model, which was pre-trained on Wikipedia and a corpus of books and movie scripts. For the *Symptoms* → *Disease* category, we see that ClinicalBERT outperforms the other biomedical LMs, although the standard BERT model actually achieves the best performance overall. The results suggest that ClinicalBERT is better at distinguishing between relatively rare diseases, but that the focus on encyclopedic text benefits BERT for more common diseases. Intuitively, we can indeed expect that the encyclopedic style of Wikipedia focuses more on symptoms of diseases than scientific articles, which might focus more on treatments, procedures and diagnostic tests. This is also in accordance with the findings from He et al. (2020), who obtained promising results with a disease-centric LM pre-training task that relies on Wikipedia. On the *Procedures* → *Disease* and *Tests* → *Disease* categories, we can see that SciBERT achieves the best results, with a particularly wide margin on the *Procedures* → *Disease* category. Finally, for the *Treatments* → *Disease* category, the relatively poor performance of BERT stands out, which conforms with the aforementioned intuition that scientific articles put more emphasis on procedures, treatments and tests. BioBERT achieves the best results, although the performance of the other biomedical LMs is quite similar.

4.3 Discussion

Which LM model? Several published works have found ClinicalBERT to outperform the other considered biomedical LMs on biomedical NLP tasks (Alsentzer et al., 2019; Kearns et al., 2019; Hao et al., 2020). In our results, however, SciBERT achieves the most consistent performance, clearly outperforming ClinicalBERT on the *Procedures* → *Disease* and *Test* → *Disease* categories, while performing similar to ClinicalBERT on the remaining categories. However, rather than providing a blanket recommendation for SciBERT, our fine-grained analysis highlights the fact that different models have different strengths. The most surprising finding, in this respect, is the performance of the standard BERT model, which achieves the best results on the *Symptoms* → *Disease* category and

	ClinicalBERT	**BioBERT**	**SciBERT**	**BERT**
anemia	95	100	100	93
aortic valve stenosis	100	100	93	100
carotid artery stenosis	50	50	60	50
coronary atherosclerosis	79	79	76	79
type 2 diabetes mellitus	67	56	64	61
gerd	0	0	0	0
cardiac arrest	95	97	92	97
heart disease	100	100	93	80
heart failure	100	100	100	100
chf	19	37	35	36
hyperglycemic disorder	57	63	80	57
hypertensive disorder	84	87	90	84
acute renal failure synd.	67	67	58	61
end-stage renal disease	77	77	78	70
disorder of lung	89	76	70	52
copd	100	100	97	100
myocardial infarction	24	25	25	21
pancreatitis	33	0	22	33
pleural effusion	80	100	100	80
pneumonia	89	93	89	66
pulmonary edema	87	82	56	76
stroke	81	100	71	100
urinary tract infection	78	77	78	77
aaa	100	96	100	100
Macro-average	$73_{\pm 2.7}$	$73_{\pm 0.4}$	$72_{\pm 2.5}$	$70_{\pm 3.2}$
Weighted average	$74_{\pm 1.8}$	$76_{\pm 1.4}$	$75_{\pm 1.3}$	$72_{\pm 3.0}$

Table 6: Results for the terminological category in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

performs comparably to BioBERT on several other categories (with *Treatments* → *Disease* being a notable exception).

		Standard		**Hyp. only**	
		full	**can**	**full**	**can**
MACRO	*Symptoms → Dis.*	$48_{\pm 0.7}$	$46_{\pm 3.0}$	$47_{\pm 4.9}$	$23_{\pm 0.5}$
	Treatments → Dis.	$64_{\pm 4.7}$	$60_{\pm 6.1}$	$65_{\pm 2.5}$	$29_{\pm 2.1}$
	Tests → Dis.	$41_{\pm 1.7}$	$38_{\pm 2.4}$	$44_{\pm 2.3}$	$18_{\pm 2.0}$
	Procedures → Dis.	$59_{\pm 4.9}$	$49_{\pm 3.2}$	$52_{\pm 2.6}$	$19_{\pm 3.0}$
	Terminological	$71_{\pm 2.3}$	$73_{\pm 2.7}$	$39_{\pm 1.3}$	$25_{\pm 0.4}$
WEIGHTED	*Symptoms → Dis.*	$54_{\pm 2.9}$	$49_{\pm 3.1}$	$53_{\pm 4.7}$	$23_{\pm 1.3}$
	Treatments → Dis.	$62_{\pm 2.8}$	$51_{\pm 5.3}$	$60_{\pm 7.1}$	$24_{\pm 1.0}$
	Tests → Dis.	$37_{\pm 1.4}$	$31_{\pm 2.6}$	$42_{\pm 0.2}$	$17_{\pm 2.8}$
	Procedures → Dis.	$54_{\pm 6.2}$	$40_{\pm 3.3}$	$59_{\pm 5.1}$	$14_{\pm 2.0}$
	Terminological	$71_{\pm 1.1}$	$74_{\pm 1.8}$	$41_{\pm 2.7}$	$22_{\pm 0.4}$

Table 7: Comparison between a variant with the full hypothesis and the proposed canonicalized version. Results are for the ClinicalBERT model in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

		ClinicalBERT	**BioBERT**	**SciBERT**	**BERT**
MACRO	*Symptoms → Dis.*	$66_{\pm 4.0}$	$56_{\pm 3.2}$	$57_{\pm 5.2}$	$56_{\pm 4.1}$
	Treatments → Dis.	$69_{\pm 4.3}$	$70_{\pm 2.0}$	$76_{\pm 4.5}$	$55_{\pm 4.8}$
	Tests → Dis.	$53_{\pm 0.9}$	$49_{\pm 3.3}$	$52_{\pm 1.0}$	$47_{\pm 0.6}$
	Procedures → Dis.	$60_{\pm 1.8}$	$56_{\pm 0.8}$	$76_{\pm 2.6}$	$60_{\pm 4.5}$
	Terminological	$77_{\pm 0.9}$	$77_{\pm 0.6}$	$74_{\pm 0.6}$	$76_{\pm 1.0}$
WEIGHTED	*Symptoms → Dis.*	$66_{\pm 5.2}$	$59_{\pm 3.5}$	$59_{\pm 4.1}$	$56_{\pm 4.6}$
	Treatments → Dis.	$64_{\pm 6.2}$	$59_{\pm 3.6}$	$68_{\pm 4.8}$	$46_{\pm 3.1}$
	Tests → Dis.	$53_{\pm 0.6}$	$51_{\pm 2.4}$	$54_{\pm 1.6}$	$43_{\pm 4.0}$
	Procedures → Dis.	$65_{\pm 3.0}$	$58_{\pm 1.0}$	$76_{\pm 0.4}$	$67_{\pm 4.5}$
	Terminological	$76_{\pm 1.6}$	$77_{\pm 1.0}$	$75_{\pm 0.4}$	$72_{\pm 0.7}$

Table 8: Results for a variant of our benchmark, in which negative examples were selected at random, in terms of F1 (%) averaged over three runs. Standard deviations (over the three runs) of the macro and weighted average are also reported.

Dataset Artefacts. As already reported by Romanov and Shivade (2018), the original MedNLI dataset has a number of annotation artefacts, which mean that hypothesis-only baselines can perform well. In our dataset, we tried to address this by only using entailment examples, and creating negative examples by corrupting these. However, without canonicalizing the hypotheses, we found that hypothesis-only baselines were still performing rather well. This is shown in Table 7, which summarizes the results we obtained for a version of our dataset without canonicalization, i.e. where the full hypotheses are provided, and the canonicalized version, where the hypotheses were replaced by the disease name only. The table shows results for the standard ClinicalBERT model, as well as for a hypothesis-only variant, which is only given the hypothesis. As can be seen, without canonicalization, the hypothesis only baseline performs similarly to the full model, even outperforming it in a few cases, with the exception of the *Terminological* category where a clear drop in performance for the hypothesis-only baseline can be seen. In contrast, for the canonicalized version of the dataset, we can see that the hypothesis only baseline, which only gets access to the name of the disease in this case, under-performs consistently and substantially. Note that the hypothesis-only baseline still achieves a non-trivial performance in most cases, noting that an uninformed classifier that always predicts true would achieve an F1 score of 0.167. However, this simply shows that the model has learned to prefer frequent diseases over rare ones.

Adversarial Examples. A key design choice has been to select negative examples from the diseases that are most similar to the target disease. To analyse the impact of this choice, we carried out an experiment in which negative examples were instead randomly selected. As before, we only consider diseases that are present in the dataset, and we ensure that negative examples are not ancestors or descendants of the target disease in SNOMED CT. The results are presented in Table 8. As expected, the results are overall higher than those from the main experiment. More surprisingly, this easier setting benefits some models more than others. The relative performance of ClinicalBERT in particular is now clearly better, with this model achieving the best results for *Symptoms* → *Disease*. Furthermore, the standard BERT model now clearly underperforms the biomedical LMs, except for *Procedures* → *Disease* where it outperforms ClinicalBERT and BioBERT.

5 Conclusion

We have proposed DisKnE, a new benchmark for analysing the extent to which biomedical language models capture knowledge about diseases. Positive examples were obtained from MedNLI and MEDIQA-NLI, by manually identifying and categorizing hypotheses that express that the patient has some disease. Negative examples were selected to be similar to the target disease. To prevent shortcut learning, the hypotheses were canonicalized, such that models only get access to the name of the disease that is inferred. Our empirical analysis shows that existing biomedical language models particularly struggle with cases that require medical knowledge. The relative performance on the different categories suggests that different (biomedical) LMs have complementary strengths.

References

Asma Ben Abacha, Chaitanya Shivade, and Dina Demner-Fushman. 2019. Overview of the MEDIQA 2019 shared task on textual inference, question entailment and question answering. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 370–379.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

Vladimir Araujo, Andres Carvallo, Carlos Aspillaga, and Denis Parra. 2020. On adversarial examples for biomedical NLP tasks. *arXiv:2004.11157*.

Carlos Aspillaga, Andrés Carvallo, and Vladimir Araujo. 2020. Stress test evaluation of transformer-based models in natural language understanding tasks. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1882–1894.

Andrew L Beam, Benjamin Kompa, Allen Schmaltz, Inbar Fried, Griffin Weber, Nathan Palmer, Xu Shi, Tianxi Cai, and Isaac S Kohane. 2020. Clinical concept embeddings learned from massive sources of multimodal medical data. In *Pacific Symposium on Biocomputing*, volume 25, pages 295–306.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3613–3618.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4762–4779.

Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7456–7463.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*.

Joe Davison, Joshua Feldman, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1173–1178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Kevin Donnelly et al. 2006. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279.

Maxwell Forbes, Ari Holtzman, and Yejin Choi. 2019. Do neural language representations learn physical commonsense? In *Proceedings of the 41th Annual Meeting of the Cognitive Science Society*, pages 1753–1759.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 650–655.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 107–112.

Boran Hao, Henghui Zhu, and Ioannis Paschalidis. 2020. Enhancing clinical bert embedding using a biomedical knowledge base. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 657–661.

Yun He, Ziwei Zhu, Yin Zhang, Qin Chen, and James Caverlee. 2020. Infusing disease knowledge into BERT for health question answering, medical inference and disease name recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4604–4614.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. 2019a. Probing biomedical embeddings from language models. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 82–89.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019b. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2567–2577.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3(1):1–9.

William Kearns, Wilson Lau, and Jason Thomas. 2019. UW-BHI at MEDIQA 2019: An analysis of representation methods for medical natural language inference. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 500–509.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Patrick Lewis, Myle Ott, Jingfei Du, and Veselin Stoyanov. 2020. Pretrained language models for biomedical and clinical tasks: Understanding and extending the state-of-the-art. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 146–157.

Mingming Lu, Yu Fang, Fengqi Yan, and Maozhen Li. 2019. Incorporating domain knowledge into natural language inference on clinical texts. *IEEE Access*, 7:57623–57632.

George Michalopoulos, Yuanxin Wang, Hussam Kaka, Helen Chen, and Alex Wong. 2020. UmlsBERT: Clinical domain knowledge augmentation of contextual embeddings using the unified medical language system metathesaurus. *arXiv:2010.10391*.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

Processing and the 9th International Joint Conference on Natural Language Processing, pages 2463–2473.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5418–5426.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels*, pages 1586–1596.

Soumya Sharma, Bishal Santra, Abhik Jana, Santosh Tokala, Niloy Ganguly, and Pawan Goyal. 2019. Incorporating domain knowledge into medical NLI using knowledge graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 6091–6096.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics - on what language model pre-training captures. *Trans. Assoc. Comput. Linguistics*, 8:743–758.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Zhaofeng Wu, Yan Song, Sicong Huang, Yuanhe Tian, and Fei Xia. 2019. WTMED at MEDIQA 2019: A hybrid approach to biomedical natural language inference. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 415–426.

AUGVIC: Exploiting BiText Vicinity for Low-Resource NMT

Tasnim Mohiuddin[*] [¶], **M Saiful Bari** [*][¶], and **Shafiq Joty**[¶][†]
[¶]Nanyang Technological University, Singapore
[†]Salesforce Research Asia, Singapore
{mohi0004, bari0001, srjoty}@ntu.edu.sg

Abstract

The success of Neural Machine Translation (NMT) largely depends on the availability of large bitext training corpora. Due to the lack of such large corpora in low-resource language pairs, NMT systems often exhibit poor performance. Extra relevant monolingual data often helps, but acquiring it could be quite expensive, especially for low-resource languages. Moreover, domain mismatch between bitext (train/test) and monolingual data might degrade the performance. To alleviate such issues, we propose AUGVIC, a novel data augmentation framework for low-resource NMT which exploits the vicinal samples of the given bitext without using any extra monolingual data explicitly. It can diversify the in-domain bitext data with finer level control. Through extensive experiments on four low-resource language pairs comprising data from different domains, we have shown that our method is comparable to the traditional back-translation that uses extra in-domain monolingual data. When we combine the synthetic parallel data generated from AUGVIC with the ones from the extra monolingual data, we achieve further improvements. We show that AUGVIC helps to attenuate the discrepancies between relevant and distant-domain monolingual data in traditional back-translation. To understand the contributions of different components of AUGVIC, we perform an in-depth framework analysis.

1 Introduction

Neural Machine Transaltion (NMT) has shown impressive performance in high-resource settings, even claiming to achieve parity with human professional translators (Hassan et al., 2018; Popel et al., 2020). Most successful NMT systems have billions of parameters (Lepikhin et al., 2021). They generally work well only when a good amount of parallel training data is available and perform poorly in low-resource conditions (Koehn and Knowles, 2017; Guzmán et al., 2019). However, majority of the languages are low-resourced despite being used by large portion of world population. Hence, improving low-resource MT quality has been of great interests to the MT researchers.

There have been several attempts to extend the success of NMT in high-resource settings to low-resource language pairs that have a relatively small amount of available parallel data. Most of these methods mainly focus on leveraging extra monolingual data through back-translation (Sennrich et al., 2016) and self-training (He et al., 2020), or translation knowledge transfer through parallel data involving other assisting language pairs (Firat et al., 2016a,b; Johnson et al., 2017; Neubig and Hu, 2018).[1] Large scale pre-training is another recent trend to utilize large monolingual data for NMT (Liu et al., 2020). However, very few work has considered low-resource NMT without using auxiliary data or other pivot languages.

In the presence of a sufficient amount of in-domain monolingual data, back-translation (BT) has proved to be quite successful (Edunov et al., 2018). In this approach, a reverse intermediate model is trained on the original parallel data, which is later used to generate synthetic parallel data by translating sentences from target-side monolingual data into the source language. However, when there are scarcity of in-domain data which indeed a common situation in many low-resource settings, the success of BT may be limited (Chen et al., 2019).

Another understudied problem with BT is the issue with *domain mismatch* (Edunov et al., 2020). To elaborate, let us consider two scenarios: (i) the training and testing data come from the same or relevant domains (*e.g.,* News), and (ii) the test domain (News) is different from the training domain (*e.g.,*

---

[*] Equal contribution

[1] See (Dabre et al., 2020) for a survey of the later.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3034–3045
August 1–6, 2021.

Subtitles). In the former case, we can foresee two problems. First, if we use out-of-domain monolingual data which is abundant, it might misguide the model and move it far away from the actual test distribution. Second, even if the monolingual data is from a domain similar to that of the training/testing data, there might be differences in topics, modality, style, etc., which might induce noise.

For the latter scenario, even if the monolingual data comes from the similar domain as the test data (News), the corresponding (reverse) translations will be noisy as the intermediate model would be trained on a different domain (Subtitles). Consequently, these noisy pseudo-parallel data will induce noise during training and might cause the model to perform worse (Wang et al., 2018). On the other hand, using in-domain (Subtitles) monolingual data in back-translation will not give enough diversity to cover the test domain (News).

In this work, inspired by the Vicinal Risk Minimization principle (Chapelle et al., 2001), we propose AUGVIC, a novel method to **aug**ment **vic**inal samples around the bitext distribution. Instead of using extra monolingual data, AUGVIC aims to leverage the vicinal samples of the original bitext, thereby enlarging the support of the training bitext distribution to improve model generalization. The main advantage is that the resulting distribution remains close to the original distribution and can be controlled at a finer level (Figure 1).

With the goal of training a source-to-target NMT system, AUGVIC augments vicinal samples in the target language. The vicinal samples are generated by predicting the masked tokens of a target bitext sentence using a pretrained large-scale language model. To generate synthetic bitext data from these augmented vicinal samples through a reverse intermediate (target-to-source) model, we propose two different methods: the first one is based on the traditional BT, while the second one leverages the original source sentence as a guide. Finally, we train the source-to-target model by combining the original parallel data with the synthetic bitext.

In order to demonstrate the effectiveness and robustness of AUGVIC, we conduct extensive experiments on four low-resource language pairs comprising data from different domains. Our results show significant improvements over the bitext baselines with 2.76 BLEU gains on an average on eight different translation tasks without using any extra monolingual data. AUGVIC also complements traditional BT with additive gains when extra monolingual data is used. We also show AUGVIC's efficacy in bridging the gap between in-domain and out-of-domain performance in traditional back-translation with monolingual data. We carried out an ablation study to understand the contribution of the diversity factor in our proposed framework. We open-source our framework at https://ntunlpsg.github.io/project/augvic/.

2 Related Work

Two lines of studies are relevant to our work.

Low-resource NMT Although the main focus of investigation and improvement in NMT has been in high-resource settings, there has been a recent surge of interest in low-resource MT. However, achieving satisfactory performance in low-resource settings turns out to be challenging for NMT systems (Koehn and Knowles, 2017). Recent research has mainly focused on creating and cleaning parallel (Ramasamy et al., 2014; Islam, 2018) and comparable data (Tiedemann, 2012), utilizing bilingual lexicon induction (Conneau et al., 2017; Artetxe et al., 2018; Mohiuddin and Joty, 2019, 2020; Mohiuddin et al., 2020), fine-grained hyperparameter tuning (Sennrich and Zhang, 2019), and using other language pairs as pivot (Cheng et al., 2017; Kim et al., 2019).

Another avenue of research follows multilingual translation, where translation knowledge from high-resource language pairs are exploited by training a single NMT system on a mix of high-resource and low-resource language pairs (Firat et al., 2016a,b; Kocmi and Bojar, 2018; Gu et al., 2018; Neubig and Hu, 2018; Guzmán et al., 2019). Zoph et al. (2016) proposed a variant where they pretrain NMT system on a high-resource language pair before finetuning on a target low-resource language pair.

Data Augmentation for NMT Till now, one of the most successful data augmentation strategies in NMT is back-translation (BT) (Sennrich et al., 2016; Hoang et al., 2018), which exploits target-side monolingual data. Edunov et al. (2018) investigated BT extensively and scaled the method to millions of target-side monolingual sentences. Caswell et al. (2019) explored the role of noise in noised-BT and proposed to use a tag for back-translated source sentences. Besides BT, self-training is another data augmentation strategy for NMT which leverages source-side monolingual data (He et al.,

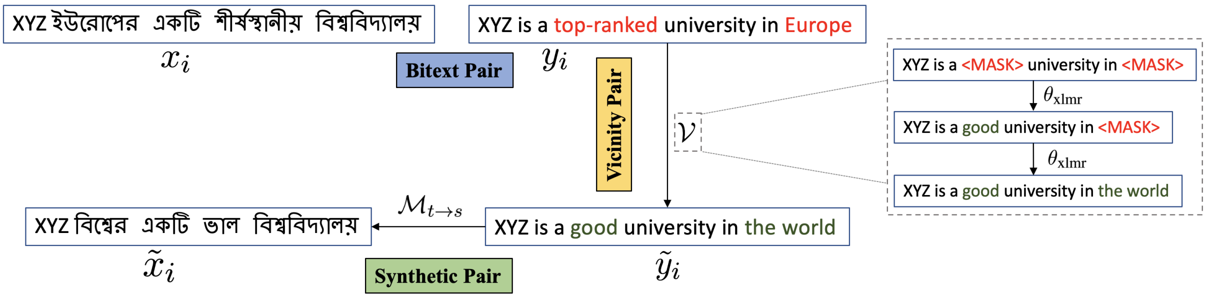

Figure 1: Illustration of AUGVIC steps for Bengali-to-English translation system. Here (x_i, y_i) is the original bitext pair, $\tilde{y}_i$ is a vicinal sample of y_i, and $(\tilde{x}_i, \tilde{y}_i)$ is a synthetic pair where $\tilde{x}_i$ is generated by a reverse intermediate translation system $\mathcal{M}_{t\to s}$. Right side of the figure shows the successive steps of vicinal sample generation.

2020). Large scale multilingual pre-training followed by bitext fine-tuning is a recent trend to utilize monolingual data for NMT, which is shown to be beneficial (Arivazhagan et al., 2019; Liu et al., 2020; Zhu et al., 2020; Lepikhin et al., 2021).

Apart from using extra monolingual data, Xie et al. (2017) show that data noising is an effective regularization method for NMT, while Wu et al. (2019) use noised training. In low-resource settings, Fadaee et al. (2017) augment bitext by replacing a common word with a low-frequency word in the target sentence, and change its corresponding word in the source sentence to improve the translation quality of rare words. Wang et al. (2018) propose an unsupervised data augmentation method for NMT by replacing words in both source and target sentences based on hamming distance. Gao et al. (2019) propose a method that replaces words with a weighted combination of semantically similar words. Recently, Nguyen et al. (2020) propose an in-domain augmentation method by diversifying the available bitext data using multiple forward and backward models. In their follow-up work (Nguyen et al., 2021), they extend the idea to unsupervised MT (UMT) using a cross-model distillation method, where one UMT model's synthetic output is used as input for another UMT model.

Summary Most of the previous work on improving BT involve either training iteratively or combining BT with self-training using monolingual data blindly without noticing the distributional differences between the monolingual and bitext data. In contrast, in AUGVIC we systematically parameterize the generation of new training samples from the original parallel data. Moreover, the combination of our augmented vicinal samples with monolingual data makes the NMT models more robust and attenuates the prevailing distributional gap.

3 Method

Let s and t denote the source and target languages respectively, and $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the bitext training corpus containing N sentence pairs with x_i and y_i coming from s and t languages, respectively. Also, let $\mathcal{M}_{s\to t}$ is an NMT model that can translate sentences from s to t, and $\mathcal{D}_{\text{mono}}^{t} = \{y_j\}_{j=1}^{M}$ denote the monolingual corpus in the target language t containing M sentences.

3.1 Traditional Back-Translation

Traditional back-translation (Sennrich et al., 2016) leverages the target-side monolingual corpus. With the aim to train a source-to-target model $\mathcal{M}_{s\to t}$, it first trains a reverse intermediate model $\mathcal{M}_{t\to s}$ using the given bitext $\mathcal{D}$, and use it to translate the extra target-side monolingual data $\mathcal{D}_{\text{mono}}^{t}$ into source language. This yields a synthetic bitext corpus $\mathcal{D}_{\text{syn}} = \{\mathcal{M}_{t\to s}(y_j), y_j)\}_{j=1}^{M}$. Then a final model $\mathcal{M}_{s\to t}$ is trained on $\{\mathcal{D} \cup \mathcal{D}_{\text{syn}}\}$ usually by upsampling $\mathcal{D}$ to keep the original and synthetic bitext pairs to a certain ratio (generally 1:1).

3.2 AUGVIC: Exploiting Bitext Vicinity

For low-resource languages, the amount of available parallel data is limited, hindering training of a good MT system. Moreover, the target language pairs can be quite different (*e.g.,* morphologically, topic distribution) from the high-resource ones, making the translation task more difficult (Chen et al., 2019). Also, acquiring large and relevant monolingual corpora in the target language is difficult in low-resource settings and can be quite expensive. The domain mismatch between the monolingual and bitext data is another issue with the traditional back-translation as mentioned in §1.

With the aim to improve model generalization, the core idea of AUGVIC is to leverage the *vicinal* samples of the given bitext rather than using extra monolingual data. The addition of bitext vicinity also alleviates the domain mismatch issue since the augmented data distribution does not change much from the original bitext distribution. Figure 1 shows an illustrative example of AUGVIC, which works in three basic steps to train a model:

(i) Generate vicinal samples $\tilde{y}_i$ of the target sentences (y_i) in the bitext data $\mathcal{D}$.

(ii) Produce source-side translations $\tilde{x}_i$ of the vicinal samples to generate synthetic bitext $\tilde{\mathcal{D}}$.

(iii) Train the final source-to-target MT model $\mathcal{M}_{s\rightarrow t}$ using $\{\mathcal{D} \cup \tilde{\mathcal{D}}\}$.

AUGVIC, however, is not mutually exclusive to the traditional back-translation and can be used together when relevant monolingual data is available. In the following, we describe how each of these steps are operationalized with NMT models.

3.2.1 Generation of Vicinal Samples

We first generate vicinal samples for each eligible target sentence y_i in the bitext $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$. Let $\mathcal{V}(\tilde{y}_i|y_i)$ denote the vicinity distribution around y_i, we create a corpus of vicinal samples as:

$$\tilde{y}_i \sim \mathcal{V}(\tilde{y}_i|y_i) \qquad (1)$$

We generate vicinal samples for sentences having lengths between 3 and 100, and $\mathcal{V}$ can be modeled with existing syntactic and semantic alternation methods like language model (LM) augmentation (Kobayashi, 2018; Wu et al., 2018; Shi et al., 2020; Bari et al., 2021), paraphrase generation (Li et al., 2018), constrained summarization (Laban et al., 2020), and similar sentence retrieval (Du et al., 2020). Most of these methods are supervised requiring extra annotations. Instead, in AUGVIC, we adopt an unsupervised LM augmentation, which makes the framework more robust and flexible to use. Specifically, we use a pretrained XLM-R masked LM (Conneau et al., 2020a) parameterized by θ_{xlmr} as our vicinal model. Thus, the vicinity distribution is defined as $\mathcal{V}(\tilde{y}_i|y_i, \theta_{\text{xlmr}})$.

Note that we treat the vicinal model as an external entity, which is not trained/fine-tuned. This disjoint characteristic gives our framework the flexibility to replace θ_{xlmr} even with a better monolingual LM for a specific target language, which in turn makes AUGVIC extendable to utilize stronger LMs that may come in the future.

In a masked LM, one can mask out a token at any position and ask the model to predict at that position. For a meaningful and informed augmentation, we mask out the tokens *successively* (one at a time) up to a required number determined by a diversity ratio, $\rho \in (0, 1)$. For a sentence of length ℓ, the successive augmentation can generate at most $(2^{\ell} - 1) \times k$ vicinal samples, where k is the number of output tokens chosen for each masked position. We use $k = 1$, and pick the one with the highest probability ensuring that it does not match the original token at the masked position. The diversity ratio (ρ) controls how much diverse the vicinal samples can be from the original sentence, and is selected using one of the following two ways:

- **Fixed diversity ratio** Here we use a fixed value for ρ, and select $t = \ell \times \rho$ tokens to mask out. We then generate new vicinity samples by predicting new tokens in those masked positions.
- **Dynamic diversity ratio** Instead of using a fixed value, in this approach we set the diversity ratio dynamically taking the sentence length into consideration. This allows finer-level control for diversification — the longer the sentence is, the smaller should its diversification ratio be. The intuition is that for long sentences, a larger value of ρ will produce vicinal samples which will be far away from the original sample. Specifically, we use the following piece-wise function to find the number of tokens to mask out dynamically:

$$t = \begin{cases} \max(\ell \times a, t_{\text{min}}) & ;\text{if } \ell \leq 20 \\ \min(\frac{\ell}{h} \times b, t_{\text{max}}) & ;\text{otherwise} \end{cases} \qquad (2)$$

where t_{min} and t_{max} are hyperparameters and represent the minimum and maximum number of tokens to be replaced by the masked LM. The other hyperparameters a, b, and h play the same role as the diversity ratio ρ.

Since we predict tokens for replacement one at a time, we can make the prediction in any of the permutation order of t. So, the maximum number of possible augmentation for a sentence of length ℓ is $\gamma = \binom{\ell}{t} \times t!$. We perform *stochastic sampling* from the distribution of γ to select N' vicinal samples. We have added an analysis on the effect of diversity ratio ρ in AUGVIC in §5.5.

3.2.2 Generation of Synthetic Bitext Data

Our objective is to train a source-to-target MT model $\mathcal{M}_{s\rightarrow t}$. So far, we have the bitext $\mathcal{D} =$

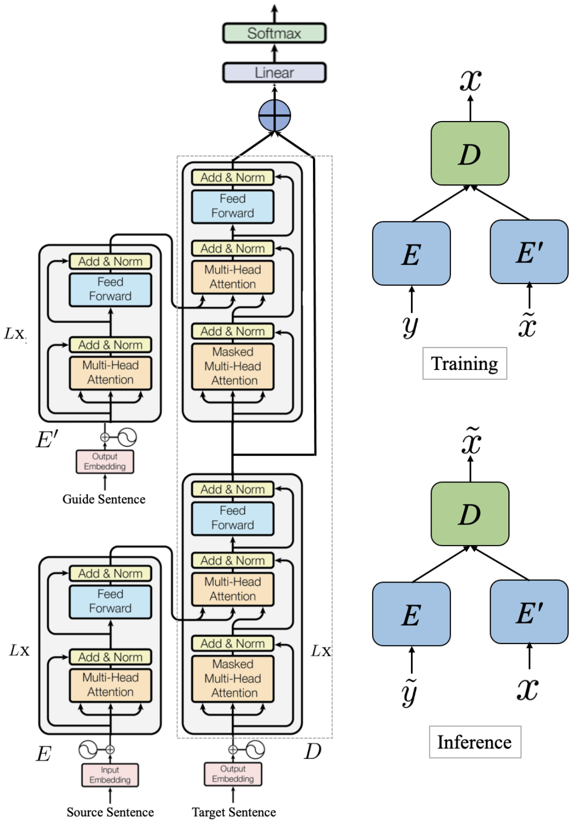

(a) (b)

Figure 2: (a) Our proposed model for guided back-translation; (b) its training and inference method.

$\{(x_i, y_i)\}_{i=1}^{N}$ and target-side monolingual data $\tilde{\mathcal{D}}^t = \{\tilde{y}_j\}_{j=1}^{N'}$ which are vicinal to the original target in $\mathcal{D}$. We need a reverse intermediate target-to-source MT model $\mathcal{M}_{t\rightarrow s}$ to translate $\tilde{y}_j$ into $\tilde{x}_j$, which will give us the synthetic bitext data $\tilde{\mathcal{D}}$. For this, we experiment with two different models.

(*a*) Pure Back-Translation (PBT) This is similar to back-translation (§3.1), where we first train the reverse MT model $\mathcal{M}_{t\rightarrow s}$ using the given bitext $\mathcal{D}$. We then use $\mathcal{M}_{t\rightarrow s}$ to translate the target-side vicinal samples $\tilde{y}_j \sim \tilde{\mathcal{D}}^t$ into $\tilde{x}_j$. This gives a synthetic bitext $\tilde{\mathcal{D}} = \{(\tilde{x}_j, \tilde{y}_j)\}_{j=1}^{N'}$. We use the Transformer architecture (Vaswani et al., 2017) as our reverse intermediate NMT model $\mathcal{M}_{t\rightarrow s}$.

(*b*) Guided Back-Translation (GBT) In the illustrative example (Figure 1), we can identify three kinds of pairs: (i) the bitext (x_i, y_i), (ii) the vicinal $(y_i, \tilde{y}_i)$, and (iii) the synthetic pair $(\tilde{x}_i, \tilde{y}_i)$. Here, y_i is the original translation of source sentence x_i and $\tilde{y}_i$ is the vicinal sample, which can be seen as a perturbation of y_i. Hence, we can assume that $\tilde{x}_i$ will also be similar to (perturbed) x_i. Our goal is to leverage this extra relational knowledge to improve the translation quality of $\tilde{x}_i$ when generating the synthetic bitext $\tilde{\mathcal{D}}$. Specifically, we use the original source x_i as a guide for generating the synthetic translation $\tilde{x}_i$ of the target-side vicinal sample $\tilde{y}_i$.

$$\tilde{x}_i = \mathcal{M}_{t\rightarrow s}(\tilde{y}_i | x_i) \tag{3}$$

For this, we propose a model based on the Transformer architecture which has two encoders - one for the source sentence (E) and another for the guide sentence (E'), and a decoder (D) (Figure 2). We use the same architecture with the exception that now we have two identical encoders (E and E'). Both the encoders have a stack of L layers, while the decoder has $(L + 1)$ layers.

Training & Inference: We train this model with a dataset of triplets containing $(y, \tilde{x}, x)$, where (x, y) comes from the original bitext and $\tilde{x}$ is a vicinal sample of x to guide the decoder in generating x. Each of the first L layers of the decoder performs cross-attention on $E(y)$ resulting in decoder states $D^{(L)}(x_{<t}|y)$ at time step t, while the final decoder layer attends on $E'(\tilde{x})$ resulting in a second set of decoder states $D^{(L+1)}(x_{<t}|y, \tilde{x})$. The two sets of decoder states are then interpolated by taking a convex combination before passing it to a linear layer followed by the Softmax token prediction.

$$\lambda D^{(L)}(x_{<t}|y) + (1 - \lambda) D^{(L+1)}(x_{<t}|y, \tilde{x}) \tag{4}$$

where λ is a hyperparameter that controls the relative contributions from the two encoders, $E(y)$ and $E'(\tilde{x})$, in generating x by the decoder D.

To generate the synthetic bitext $\tilde{D}$, we need to translate $\tilde{y}$, which will be guided by x. So during *inference*, we feed $\tilde{y}$ to E and x to E' to autoregressively generate $\tilde{x}$ with beam search decoding.

3.2.3 Training of the Final Model

We combine the original bitext $\mathcal{D}$ and the synthetic bitext $\tilde{\mathcal{D}}$ generated from the previous step to train our final source-to-target model $\mathcal{M}_{s\rightarrow t}$. We use the standard Transformer as our final model.

4 Experimental Setup

4.1 Datasets and Evaluation Metrics

We conduct experiments on four low-resource language pairs: English (En) to/from Bangla (Bn), Tamil (Ta), Nepalese (Ne), and Sinhala (Si). Table 1 presents the source of the collected datasets and their domains for each language pair.

Pair	Data-Source	Train & Dev	Test
En-Bn	Islam (2018)	Mixed	Mixed
En-Ta	Ramasamy et al. (2014)	News, Bible, Cinema	News, Bible, Cinema
En-Ne	Guzmán et al. (2019)	Bible, GV, PTB, Ubuntu	Wikipedia
En-Si	Guzmán et al. (2019)	Opens subtitles, Ubuntu	Wikipedia

Table 1: Sources and domains of the datasets.

Even though the En-Bn dataset size is relatively small (∼ 72K pairs), the quality of the bitext is rich, and it covers a diverse set of domains including literature, journalistic texts, instructive texts, administrative texts, and texts treating external communication. Here the distributions in train and test splits are about the same. For En-Ta, the train and test domains are similar, mostly coming from the news (∼ 66.43%). For En-Ne and En-Si, we use the datasets from (Guzmán et al., 2019), where the train and test domains are different. Although these two datasets are comparatively larger (∼ 600K pairs each), the quality of the bitext is poor, requiring further cleaning and deduplication.

Table 2 presents the dataset statistics after deduplication where the last column specifies the number of augmented data by our method AugVic (§3.2.1). For a fair comparison with the traditional back-translation, we experiment with the same amount of target-side monolingual data from three domains: news, wiki, and gnome. We collected and cleaned News, Wiki, and Gnome datasets from News-crawl, Wiki-dumps, and Gnome localization guide, respectively. For some languages, the amount of specific domain monolingual data is limited, where we added additional monolingual data of that language from Common Crawl.

Following previous work (Guzmán et al., 2019; Nguyen et al., 2020), we report the tokenized BLEU (Papineni et al., 2002) when translating from English to other languages, and detokenized SacreBLEU (Post, 2018) when translating from other languages to English for all our experiments,.

Pair	Train	Dev	Test	Augmented (AugVic/Mono)
En-Bn	70,854	500	500	≈ 460K
En-Ta	166,851	1000	2000	≈ 1300K
En-Ne	234,514	2559	2835	≈ 1500K
En-Si	571,213	2898	2766	≈ 1500K

Table 2: Dataset statistics after deduplication.

4.2 Baselines

We compare AugVic with the following baselines:

(i) Bitext baseline is the model trained with the bitext given with the dataset.

(ii) Upsample baseline Here we upsample the bitext to the same amount of AugVic's data.

(iii) Diversification baseline Nguyen et al. (2020) diversifies the original parallel data by using the predictions of multiple forward and backward NMT models. Then they merge the augmented data with the original bitext on which the final NMT model is trained. Their method is directly comparable to AugVic, as both methods diversify the original bitext, but in different ways.

4.3 Model Settings

We use the Transformer (Vaswani et al., 2017) implementation in Fairseq (Ott et al., 2019). We follow the basic architectural settings from (Guzmán et al., 2019), which establishes some standards for low-resource MT. For low-resource "Bitext baseline", they use a smaller (5-layer) Transformer architecture as the dataset is small, while for larger datasets (*e.g.,* with additional synthetic data) they use a bigger (6-layer) model.[2] To keep the architecture the same in the respective rows (Table 3), we use a 6-layer model for "Upsample baseline" and 5-layer for "Bitext baseline". More specifically, for datasets with less than a million bitext pairs, we use an architecture with 5 encoder and 5 decoder layers, where the number of attention heads, embedding dimension, and inner-layer dimension are respectively 8, 512, and 2048. Otherwise, we use a larger Transformer architecture with 6 encoder and 6 decoder layers with the number of attention heads, embedding dimension, and inner-layer dimension of 16, 1024, and 4096, respectively.

After deduplication, we tokenize non-English data using the Indic NLP Library.[3] We use the sentencepeiece library[4] to learn the joint Byte-Pair-Encoding (BPE) of size 5000 symbols for each of

---

[2] https://github.com/facebookresearch/flores/

[3] https://github.com/anoopkunchukuttan/indic_nlp_library

[4] https://github.com/google/sentencepiece

Setting	**Data**	**En-Bn** →	**En-Bn** ←	**En-Ta** →	**En-Ta** ←	**En-Ne** →	**En-Ne** ←	**En-Si** →	**En-Si** ←
Baseline	Bitext	13.21	21.18	11.58	26.29	4.59	8.34	1.96	7.45
	× Upsample	16.59	25.51	12.15	27.71	4.16	7.79	1.81	6.93
Diversification	+ Nguyen et al. (2020)	17.54	26.11	12.74	28.54	5.7	8.9	2.2	8.2
	+ AugVic	**18.03**	**26.96**	**12.93**	**28.68**	**6.47**	**10.65**	**3.66**	**9.27**
Extra mono. data	+ BT-Mono (News)	18.81	27.11	13.51	29.38	6.44	12.48	3.56	11.75
	+ BT-Mono (Wiki)	18.52	26.33	13.23	29.01	6.91	13.02	3.91	11.86
	+ AugVic+ BT-Mono (News)	19.98	28.14	13.87	**30.15**	6.80	13.12	4.94	11.89
	+ AugVic+ BT-Mono (Wiki)	**20.39**	**28.48**	**13.89**	30.14	**7.27**	**13.52**	**5.24**	**12.09**

Table 3: Detokenized Sacre-BLEU scores for $\{$Bn, Ta, Ne, Si$\} \rightarrow$ En and tokenized BLEU fro En $\rightarrow \{$Bn, Ta, Ne, Si$\}$. "BT-Mono" stands for traditional back-translation with extra target-side monolingual data (§3.1).

the language pair over the raw English and tokenized non-English bitext training data.

We tuned the hyper-parameters a, b, h, t_{min}, t_{max} in Eq. 2 and λ in Eq. 4 by small-scale experiments on the validation-sets. We found $a = 0.5$, $b = 2.5$, $h = 10$, $t_{min} = 1$, and $t_{max} = 20$ work better. We tuned λ within the range of 0.5 to 0.9. In general, we observe that for smaller sentences (length $<=$ 20), 50-60% successive-token-replacement works better while for longer sentences (length $>$ 20), 20-30% token-replacement performs better.

Following Guzmán et al. (2019), we train all the models upto a maximum epoch of 100 with early-stopping enabled based on the validation loss. We use the beam-search-decoding for inference. All the reported results for AugVic use dynamic diversity ratio for generating vicinal samples unless otherwise specified.

5 Results and Analysis

In this section, we present our results and the analysis of our proposed methods.

5.1 Comparison with Bitext & Diversification

Table 3 presents the BLEU scores on the eight translation tasks. First, we compare our model AugVic with the model trained on the original parallel data (Bitext). AugVic consistently improves the performance over all the tested language pairs, gaining about $+2.76$ BLEU scores on average. Specifically, AugVic achieves the absolute improvements of 4.28, 5.78, 1.35, 2.39, 1.88, 2.31, 1.70, and 1.82 over the Bitext for En-Bn, Bn-En, En-Ta, Ta-En, En-Ne, Ne-En, En-Si, and Si-En, respectively.

For a fair comparison, in another experiment, we upsample the bitext data to make it similar to the amount of AugVic's data. From the *Upsample* results (with a 6-layer architecture) reported in Table 3, we see that even though it increases the BLEU scores for En to/from $\{$Bn, Ta$\}$, it has negative impacts on En to/from $\{$Ne, Si$\}$ where it degrades the performance. Overall, AugVic achieves 1.75 BLEU score improvements on an average over the Upsample baseline.

The comparison with the diversification strategy proposed by Nguyen et al. (2020) reveals that AugVic outperforms their method by 0.84 BLEU scores on average. To be specific, our method gets 0.49, 0.85, 0.19, 0.14, 0.77, 1.75, 1.46, and 1.07 absolute BLEU improvements over their approach for En-Bn, Bn-En, En-Ta, Ta-En, En-Ne, Ne-En, En-Si, and Si-En, respectively.

The data diversification method of Nguyen et al. (2020) relies heavily on the performance of base models (Bitext). From Table 3, we see that the performance of base models are poor for En to/from $\{$Ne, Si$\}$, which impacts their augmented data generation process (diversification). However, the better performance of AugVic in those languages indicates that vicinal samples generated in our method are more diverse with better quality and less prone to the noise in base models.

5.2 Vicinal Samples with Extra Relevant Monolingual Data

We further explore the performance of AugVic by experimenting with the traditional back-translation method (§3.1) using the same amount of monolingual data. To perceive the variability, we choose to experiment with extra monolingual data from two *relevant* but different sources - newscrawl (BT-Mono (News)) and Wikipedia (BT-Mono (Wiki)). From the results in Table 3, we see that standard back-translation improves the scores in both cases, proving that extra relevant monolingual data helps

Interm. BT system	**En-Bn** →	←	**En-Ta** →	←	**En-Ne** →	←	**En-Si** →	←
Pure BT	18.03	26.96	12.93	28.68	**6.47**	**10.65**	**3.66**	**9.27**
Guided BT	**18.18**	**27.35**	**13.17**	**29.05**	4.81	8.62	2.16	7.71

Table 4: Comparison between two **interm**ediate reverse back-translation **(BT)** systems in AugVic.

BT-mono Domain	**Data**	**En-Bn** →	←	**En-Ta** →	←
	Bitext	13.21	21.18	11.58	26.29
News (relevant)	+ BT	18.81	27.11	13.51	29.38
	+ AugVic+ BT	19.98	28.14	13.87	30.15
gnome (distant)	+ BT	17.14	26.05	12.55	27.91
	+ AugVic+ BT	18.86	27.56	13.59	29.89

Table 5: Effect of relevant and distant domain monolingual data in back-translation with AugVic. We use *News* as "relevant" and *gnome* as "distant" domain.

for low-resource MT significantly.

To understand the exclusivity of the vicinal samples of AugVic from the external related monolingual data, we perform another set of experiments where we added both the AugVic's augmented data with the extra monolingual data and trained along with the Bitext data. From Table 3, we see that the combination of datasets improves the BLEU scores by 1.02 and 0.73 on average on the two relevant data sources (News and Wiki). From this, we can conclude that vicinal samples of AugVic make the NMT models more robust in the presence of the relevant monolingual data and can be used together when available.

5.3 Pure vs. Guided: Which One is Better?

For all the results of AugVic presented in Table 3, we use the pure back-translation (BT) method (§3.2.2(a)) as the reverse intermediate model. We compare the performance of the guided BT (§3.2.2(b)) with the pure BT method as the reverse intermediate model in Table 4. From the results, we observe that the guided BT achieves better results in En↔ {Bn, Ta}, while the pure BT achieves better in En↔ {Ne, Si} translation tasks.

We investigated why the guided BT performed poorly in En↔ {Ne, Si} tasks, and found that compared to the En-Bn and En-Ta bitexts, the original bitexts of En-Ne and En-Si languages are very noisy (*e.g.,* bad sentence segmentation, code-mix data), which propagates further noise while using the target translation as a guide for translating the vicinal samples. The diminishing results while upsampling in these two languages (Table 3) supports this claim. From these results, we can say that the better the original bitext quality is, the better the synthetic bitext will be for the guided BT.

5.4 AugVic with Relevant and Distant-domain Monolingual Data

To verify how traditional back-translation and AugVic perform with with monolingual data from related vs. distant domains, we perform another set of experiments on En to/from {Bn, Ta}. For both the language pairs (§4.1), *News* can (roughly) be considered as relevant compared to *gnome*,[5] which can be considered as distant domain. We use *pure BT* as the intermediate reverse back-translation system for generating synthetic data in AugVic in this set of experiments.

From Table 5, we see that traditional back-translation (+ BT) improves the BLEU scores over the Bitext by 4.14 and 2.85 on average for relevant- and distant-domain monolingual data, respectively, yielding higher gains for relevant domain, as expected. The addition of vicinal data by AugVic (+ AugVic+ BT) further improves the scores in both cases; interestingly, the relative improvements are higher in the distant-domain case. Specifically, the average BLEU score improvements over Bitext for relevant- and distant-domain data with AugVic+BT are 4.97 and 4.41, respectively. Comparing this with BT only, the BLEU score difference between relevant and distant domains has been reduced from 1.29 to 0.56. This indicates that AugVic helps to bridge the domain gap between relevant and distant-domain distributions in traditional BT with monolingual data.

In principle, for vicinal samples, the synthetic-pair generation capability of the reverse intermediate target-to-source MT model should be better than generating from an arbitrary monolingual data as it could be a distant distribution compared to the bitext. Judging by the amount of diverse data used for training the language model, we can safely assume that it is a diverse knowledge source (Conneau et al., 2020b) compared to the training bitext samples. Data that performs well on the reverse

---

[5] http://opus.nlpl.eu/GNOME.php

intermediate target-to-source MT system can be extrapolated from the knowledge-base as vicinal-distribution with the controlled diversity ratio function (Eq. 2). Moreover, to achieve more diversity, the use of multiple different language models is also compatible in AUGVIC.

5.5 Effect of Diversity Ratio in AUGVIC

For monolingual data, it could be challenging to identify domain discrepancy with the training/testing bitext data, and there is no parameter in the traditional BT method to control this distributional mismatch. However, in AUGVIC we can control the distributional drift of the generated vicinal samples from the original training distribution by varying the diversity ratio ρ.

Theoretically, it is possible to sample the same distribution using dynamic and static diversity. However, dynamic diversity is more flexible to perform hyperparameter-tuning and to prevent potential outliers. The term l/h in Eq. 2 represents pseudo-segmentation (h segments) of a large sentence of length l, and b represents the same intuition as ρ. Apart from these, t_{min} and t_{max} prevents irregular-samples: (i) t_{min} ensures that there should be at least some changes in the augmented sample, (ii) t_{max} makes sure that the generated-samples from LM do not diverge too much from the vicinity.

To understand the effect of the diversity ratio in AUGVIC, we perform another set of experiments. We choose to use En to/from {Bn, Ne} for this experiments, where we selected at most two vicinal samples from each of the target sentence in original bitext. We investigate the effect of both *dynamic* and *fixed* diversity ratio in AUGVIC's vicinal sample generation (§3.2.1). For fixed diversity ratio we use ρ values 0.1, 0.3, 0.5, and 0.8, while for dynamic diversity ratio we use $a = 0.5, b = 2.5$, and $h = 10$ for controlling the diversity.

We present these experimental results in Table 6, from where we see that the dynamic diversity ratio performs better in three out of four tasks. For the fixed diversity ratio, we see the variation in results for different values of ρ. In all the four tasks, the diversity ratio $\rho = 0.8$ gives the least scores. On average, we get the better results with $\rho = \{0.3, 0.5\}$. These experiments suggest that higher diversity values may induce noise and lower diversity values may not diversify the data enough to benefit the final NMT model.

AUGVIC **diversity ratio**	**En-Bn** →	←	**En-Ne** →	←
Dynamic	**17.69**	**26.61**	**6.21**	10.25
Fixed				
$\rho = 0.1$	17.34	25.98	5.98	10.03
$\rho = 0.3$	17.52	26.19	6.19	10.36
$\rho = 0.5$	17.48	26.49	6.05	**10.38**
$\rho = 0.8$	17.19	25.01	5.82	9.89

Table 6: Effect of diversity ratio ρ while generating vicinal samples in AUGVIC (§3.2.1).

6 Conclusion

We have presented an in-domain data augmentation framework AUGVIC by exploiting the bitext vicinity for low-resource NMT. Our method generates vicinal samples by diversifying sentences of the target language in the bitext in a novel way. It is simple yet effective and can be quite useful when extra in-domain monolingual data is limited.

Extensive experiments with four low-resource language pairs comprising data from different domains show the efficacy of AUGVIC. Our method is not only comparable with traditional back-translation with in-domain monolingual data, it also makes the NMT models more robust in the presence of relevant monolingual data. Moreover, it bridges the distributional gap for out-of-domain monolingual data when using together.

References

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George F. Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *CoRR*, abs/1907.05019.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

M Saiful Bari, Tasnim Mohiuddin, and Shafiq Joty. 2021. Uxla: A robust unsupervised data augmentation framework for cross-lingual nlp. In *Proceedings of The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*, Online. Association for Computational Linguistics.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. 2001. Vicinal risk minimization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 416–422. MIT Press.

Peng-Jen Chen, Jiajun Shen, Matthew Le, Vishrav Chaudhary, Ahmed El-Kishky, Guillaume Wenzek, Myle Ott, and Marc'Aurelio Ranzato. 2019. Facebook AI's WAT19 Myanmar-English translation task submission. In *Proceedings of the 6th Workshop on Asian Translation*, pages 112–122, Hong Kong, China. Association for Computational Linguistics.

Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint training for pivot-based neural machine translation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3974–3980.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5).

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2020. On the evaluation of machine translation systems trained with back-translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2836–2846, Online. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy. Association for Computational Linguistics.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. 2018. Universal neural machine translation for extremely low resource languages. *CoRR*, abs/1802.05368.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, C. Chen, Vishal Chowdhary, J. Clark, C. Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, W. Lewis, M. Li, Shujie

Liu, T. Liu, Renqian Luo, Arul Menezes, Tao Qin, F. Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and M. Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *ArXiv*, abs/1803.05567.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

M. A. A. Mumin; M. H. Seddiqui; M. Z. Iqbal; M. J. Islam. 2018. Supara0.8m: A balanced english-bangla parallel corpus.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi, and Hermann Ney. 2019. Pivot-based transfer learning for neural machine translation between non-English languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 866–876, Hong Kong, China. Association for Computational Linguistics.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2018. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Brussels, Belgium. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Philippe Laban, Andrew Hsi, John Canny, and Marti A. Hearst. 2020. The summary loop: Learning to write abstractive summaries without examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5135–5150. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation.

Tasnim Mohiuddin, M Saiful Bari, and Shafiq Joty. 2020. LNMap: Departures from isomorphic assumption in bilingual lexicon induction through nonlinear mapping in latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2712–2723, Online. Association for Computational Linguistics.

Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting adversarial autoencoder for unsupervised word translation with cycle consistency and improved training. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3857–3867, Minneapolis, Minnesota. Association for Computational Linguistics.

Tasnim Mohiuddin and Shafiq Joty. 2020. Unsupervised Word Translation with Adversarial Autoencoder. *Computational Linguistics*, 46(2):257–288.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq Joty, Thanh-Tung Nguyen, Wu Kui, and Ai Ti Aw. 2021. Cross-model Back-translated Distillation for Unsupervised Machine Translation. In *Thirty-eighth International Conference on Machine Learning*, ICML'21, Virtual.

Xuan-Phi Nguyen, Shafiq R. Joty, Kui Wu, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakub Uszkoreit, Ondřej Bojar, and Zdeněk Žabokrtskỳ. 2020. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature Communications*, 11(1):1–15.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Loganathan Ramasamy, Ondřej Bojar, and Zdeněk Žabokrtský. 2014. EnTam: An english-tamil parallel corpus (EnTam v2.0). LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Linqing Shi, Danyang Liu, Gongshen Liu, and Kui Meng. 2020. Aug-bert: An efficient data augmentation algorithm for text classification. In *Communications, Signal Processing, and Systems*, pages 2191–2198, Singapore. Springer Singapore.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Lijun Wu, Yiren Wang, Yingce Xia, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019. Exploiting monolingual data at scale for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4207–4216, Hong Kong, China. Association for Computational Linguistics.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2018. Conditional BERT contextual augmentation. *CoRR*, abs/1812.06705.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. *CoRR*, abs/1703.02573.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

Provably Secure Generative Linguistic Steganography

Siyu Zhang, Zhongliang Yang, Jinshuai Yang, Yongfeng Huang
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
{zhangsiy19, yjs20}@mails.tsinghua.edu.cn
yangzl15@tsinghua.org.cn
yfhuang@mail.tsinghua.edu.cn

Abstract

Generative linguistic steganography mainly utilized language models and applied steganographic sampling (stegosampling) to generate high-security steganographic text (stegotext). However, previous methods generally lead to statistical differences between the conditional probability distributions of stegotext and natural text, which brings about security risks. In this paper, to further ensure security, we present a novel provably secure generative linguistic steganographic method **ADG**, which recursively embeds secret information by **A**daptive **D**ynamic **G**rouping of tokens according to their probability given by an off-the-shelf language model. We not only prove the security of ADG mathematically, but also conduct extensive experiments on three public corpora to further verify its imperceptibility. The experimental results reveal that the proposed method is able to generate stegotext with nearly perfect security.

1 Introduction

Steganography is the technology of hiding secret information within an innocent natural carrier (such as image (Hussain et al., 2018), audio (Mishra et al., 2018), video (Liu et al., 2019), text (Krishnan et al., 2017), etc) in order to avoid eavesdropping. Steganography differs from cryptography in that cryptography only conceals the content of secret information, whereas steganography even conceals its very existence, which makes it more secure and reliable in some scenarios (Anderson and Petitcolas, 1998).

Natural language is suitable as a carrier of steganography by virtue of its high robustness in transmission (Ziegler et al., 2019). Unlike digital images or digital audio which is sensitive to distortions like compression, cropping, blurring or pixel-wise dropout, text can usually be transmitted losslessly through different kinds of public channels. Nevertheless, text generally has low entropy and lacks sufficient redundancy for information hiding (Sharma et al., 2016), which often results in low embedding capacity of linguistic steganography. For example, in traditional modification-based methods (such as synonym substitution (Xiang et al., 2014, 2018) and spelling transformation (Shirali-Shahreza, 2008)), where secret information is encoded by slightly modifying an existing covertext, the options for modification can be very limited to keep the text fluent enough so as not to arouse suspicions.

In recent years, powered by the advanced technology of deep learning and natural language processing, language models based on neural networks have made significant progress in generating fluent text (Radford et al., 2019; Brown et al., 2020), which bring new vitality to linguistic steganography and facilitate the investigation of generation-based methods (Fang et al., 2017; Yang et al., 2018a; Dai and Cai, 2019; Ziegler et al., 2019; Yang et al., 2020a; Zhou et al., 2021). The generative linguistic steganography directly transform secret information into innocuous-looking steganographic text (stegotext) without any covertext. Using an off-the-shelf language model, secret information can be encoded in the selection of token at each time step autoregressively during the generation procedure, which greatly alleviates the drawback of low embedding capacity. However, previous methods inevitably introduce distortions during generation. The imperceptibility of generative linguistic steganography still needs further optimization.

In this paper, we aim to further improve the imperceptibility of generative linguistic steganography. The contributions of this work are the following:

1. We present **ADG** (**A**daptive **D**ynamic

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3046–3055
August 1–6, 2021.

Grouping), a novel generative linguistic steganographic method based on off-the-shelf language models, which groups the tokens adaptively in accordance with their probability at each time step to embed secret information dynamically in the generated stegotext.

2. We discuss the security of ADG and give mathematical proof, which reveals that the proposed method is provably secure.
3. Through quantitative analysis, we derive satisfactory experimental results in terms of both imperceptibility and embedding capacity, which further verifies the effectiveness of ADG.

Our code is available at `https://github.com/Mhzzzzz/ADG-steganography`.

2 Formalism

2.1 Notation

We use lowercase letters in bold type (e.g. $\boldsymbol{a}$) to denote vectors, normal lowercase letters (e.g. a) to denote scalars and uppercase letters (e.g. A) to denote sets. We use the symbol $|A|$ to denote the size of a set. Calligraphic letters denote neural models (e.g. $\mathcal{A}$). Both English letters and Greek letters are adopted. We use $p(\cdot)$ and $q(\cdot)$ to denote distributions and $f(\cdot)$ to denote functions, which are usually shortened to p, q and f. Subscripts and superscripts are used to tell the different variables/distributions/functions apart.

2.2 Generative Linguistic Steganography

Language modeling is a task to estimate the joint distribution of serialized natural language $p_{LM}(\boldsymbol{w})$, where $\boldsymbol{w}$ is a sequence of n tokens $[w_1, w_2, ..., w_n]$ and each token belongs to the vocabulary Σ. For an autoregressive language model $\mathcal{L}$, the output is usually factorized as a product of conditional distribution of the current token

$$\begin{aligned} p_{LM}(\boldsymbol{w}) &= p_{LM}(w_1, w_2, ..., w_n) \\ &= p_{LM}(w_1) \cdot \prod_{t=2}^{n} p_{LM}(w_t | w_1, ..., w_{t-1}). \end{aligned} \tag{1}$$

According to Simmons (1984), it is usually supposed that Alice (sender) wants to send a secret message $\boldsymbol{m} \sim \text{Uniform}(\{0,1\}^l)$ to Bob (receiver) through a public channel monitored by Eve (adversary). In generative linguistic steganography, they share an *embedding* algorithm f_{emb} which takes a language model $\mathcal{L}$ and the secret message $\boldsymbol{m}$ as input and then outputs *stegotext* $\boldsymbol{y}$ to transmit. They also share a corresponding *extraction* algorithm f_{ext}, which is the inverse mapping of f_{emb} that is able to recover the secret message $\boldsymbol{m}$ according to the language model $\mathcal{L}$ and the received stegotext $\boldsymbol{y}$.

2.3 Imperceptibility

In order to avoid raising Eve's suspicions, stegotext $\boldsymbol{y}$ is required to be fluent enough and statistically indistinguishable from natural innocuous text $\boldsymbol{x}$, which we call *covertext*. Cachin (1998) proposed the *information-theoretic security* of steganography to measure the statistical imperceptibility quantitatively, which is defined as the Kullback-Leibler divergence (KL divergence) between the distributions of covertext $\boldsymbol{x}$ and stegotext $\boldsymbol{y}$. The distortion of generative linguistic steganography is two-fold: one is introduced by the bias of the language models, which is the gap between the true distribution of natural text $p_{true}(\boldsymbol{x})$ and the modeled distribution $p_{LM}(\boldsymbol{x})$; the other is introduced by f_{emb}. Instead of directly sampling from the modeled distribution, the embedding algorithm f_{emb} actually provides a special way to sample from $p_{LM}(\boldsymbol{y})$, which we call steganographic sampling (stegosampling). It is equivalent to sampling from a modified distribution $q(\boldsymbol{y})$ produced by an implicit language model $\mathcal{L}'$. In a word, the latter distortion is the gap between $p_{LM}(\boldsymbol{y})$ and $q(\boldsymbol{y})$, which can also be regarded as the gap between the conditional distributions $p_{LM}(y_t|y_{<t})$ and $q(y_t|y_{<t})$. We simply use p_{LM} and q to refer to the conditional distributions in the rest of this paper.

3 Related Work

In the early stage, some researchers investigated rule-based approaches or using Markov Chains to achieve generative linguistic steganography (Wayner, 1992; Chapman and Davida, 1997; Chapman et al., 2001; Chapman and Davida, 2002; Dai et al., 2010; Moraldo, 2014; Luo et al., 2016; Yang et al., 2018b). However, these methods followed a simplistic pattern and are hard to guarantee the grammatical correctness and the semantic fluency of the generated stegotext.

With the development of deep learning, language models based on neural networks show great performance on automatic text generation. The pattern of generating stegotext with neural language models has been widely accepted. Fang et al. (2017)

proposed a linguistic steganographic method that randomly partitioned the vocabulary Σ into 2^b bins $[B_1, B_2, ..., B_{2^b}]$ and each one contained $|\Sigma|/2^b$ tokens. At each time step, they selected the token with the highest probability within the bin according to the $b-$bit secret information to be embedded. Yang et al. (2018a) improved the embedding algorithm by building the mapping from secret information to tokens dynamically at each time step rather than statically in advance. Concretely, the top 2^k tokens with the highest probability were encoded by Huffman coding algorithm. Then they took the token which has the same code as the secret information. Dai and Cai (2019) proposed *patient-Huffman*, which was an improved version of Yang et al. (2018a) that sacrificed embedding capacity for imperceptibility. They first calculated the distortion (total variation distance or KL divergence) between q and p_{LM} and then only used Huffman coding embedding algorithm to embed secret information when the distortion was less than a preset threshold δ. Otherwise they directly sampled a token to avoid high distortion occasions. Ziegler et al. (2019) employed arithmetic coding to embed secret information. They truncated the top h likely tokens and left out the low-probability long-tails. Then the tokens are encoded by arithmetic coding algorithm and selected according to the secret information. Compared with other coding algorithm, arithmetic coding has higher compression rate, which results in less damage to conditional probability distribution p_{LM} and helps to improve imperceptibility.

4 ADG Methodology

According to the analysis in Section 2.3, the distortion of generative linguistic steganography includes the bias of the language model $\mathcal{L}$ and the damage to the conditional distribution caused by the embedding algorithm f_{emb}. The former is not our research priority. With the development of automatic text generation, the former distortion can be gradually minimized. In this paper, we mainly pay attention to the latter distortion. We aim to seek an optimal solution theoretically and experimentally.

Given an off-the-shelf language model, how can we embed secret information to the generated tokens? Unlike previous works that encoded the conditional distribution by lossless coding algorithm, we achieve this goal in a novel way by grouping. Through mathematical analysis and proof, we propose a provably secure method ADG, which does little damage to the conditional distribution and is nearly equivalent to directly sampling from the full distribution. In this section, we investigate the security of steganography by grouping and give detailed descriptions of the proposed method.

4.1 Steganography by Grouping

Steganography by grouping is to group all tokens in the vocabulary into several groups, so that each group represents a unique secret message. E.g. we can Tokens belonging to the target group are able to make up the stegotext. In such a way, Bob reads each token in the sequence in turn and performs the same grouping operation to extrapolate which groups the current token belongs to, thereby extracting the corresponding secret information. The key question is: how to group the tokens at each time step to ensure an optimal imperceptibility? We have the following assumption.

Assumption 1. *For secret information in the form of uniformly distributed bitstream, adaptively grouping the vocabulary into* u *groups (*$u = 2^r, r \in N, r \leq \log_2 |\Sigma|$*) with equal probability will ensure the optimal imperceptibility.*

Proof. Assuming that the discrete conditional probability distribution p_{LM} is arbitrarily partitioned into u groups to embed r-bit secret information. p_{ij} denotes the probability of the j-th token in the i-th group. η_i and n_i denote the total probability and the size of the i-th group respectively. Then we have

$$\sum_{j=1}^{n_i} p_{ij} = \eta_i, \sum_{i=1}^{u} \eta_i = 1. \tag{2}$$

Our goal is to figure out the grouping algorithm to achieve the best imperceptibility, i.e. to minimize the gap between p_{LM} and q. First of all, starting from the modeled distribution $p_{LM} = [..., p_{ij}, ...]$, we calculate the equivalent distribution q. The probability of each token is firstly normalized within its group ($1/\eta_i$) and then multiplied by the selected probability of the group, which is $1/u$ since secret information is uniformly distributed. Therefore, q has the following form

$$q = [..., p_{ij}/u\eta_i, ...]. \tag{3}$$

We measured the gap between the two distributions with KL divergence, which is

$$D_{KL}(p_{LM}||q) = \sum p_{LM} \log \frac{p_{LM}}{q}$$

$$
\begin{aligned}
&= \sum_{i=1}^{u}\sum_{j=1}^{n_i} p_{ij}\log\frac{p_{ij}}{p_{ij}/u\eta_i} \\
&= \sum_{i=1}^{u}\sum_{j=1}^{n_i} p_{ij}\log(u\eta_i) \\
&= \sum_{i=1}^{u}\log(u\eta_i)\sum_{j=1}^{n_i} p_{ij} \\
&= \sum_{i=1}^{u}\eta_i\log(u\eta_i). \qquad (4)
\end{aligned}
$$

Therefore, the KL divergence between the two distributions is a function of the vector $\boldsymbol{\eta} = [\eta_1, \eta_2, ..., \eta_u]$.

Next, we will prove Assumption 1 in two steps.

[1]. Considering the auxiliary function $f_{aux}(\eta) = \eta\log(u\eta)$, $(0 \leq \eta \leq 1)$, we firstly analyse its concavity and convexity on the domain of definition. For every $\eta_1, \eta_2 \in (0,1)$ and $0 \leq \lambda \leq 1$,

$$
\begin{aligned}
& f_{aux}(\lambda\eta_1 + (1-\lambda)\eta_2) \\
& - \lambda f_{aux}(\eta_1) - (1-\lambda)f_{aux}(\eta_2) \\
& = (\lambda\eta_1 + (1-\lambda)\eta_2)\log(u(\lambda\eta_1 + (1-\lambda)\eta_2)) \\
& - \lambda(\eta_1\log(u\eta_1)) - (1-\lambda)(\eta_2\log(u\eta_2)) \\
& = \lambda\eta_1\log\frac{\lambda\eta_1 + (1-\lambda)\eta_2}{\eta_1} \\
& + (1-\lambda)\eta_2\, log\frac{\lambda\eta_1 + (1-\lambda)\eta_2}{\eta_2} \\
& \leq \lambda(\eta_1(\frac{\lambda\eta_1 + (1-\lambda)\eta_2}{\eta_1} - 1)) \\
& + (1-\lambda)\eta_2(\frac{\lambda\eta_1 + (1-\lambda)\eta_2}{\eta_2} - 1) \\
& = \lambda(\lambda\eta_1 + (1-\lambda)\eta_2 - \eta_1) \\
& + (1-\lambda)(\lambda\eta_1 + (1-\lambda)\eta_2 - \eta_2) \\
& = \lambda(\lambda-1)\eta_1 + \lambda(1-\lambda)\eta_2 \\
& + (1-\lambda)\lambda\eta_1 - (1-\lambda)\lambda\eta_2 \\
& = 0.
\end{aligned}
\qquad (5)
$$

As a result, $f_{aux}(\eta)$ is convex over $(0,1)$.

[2]. Then, when generalizing to u variables $\eta_1, \eta_2, ..., \eta_u$, $\sum_{i=1}^{u}\eta_i = 1$, according to Jensen's inequality (Jensen et al., 1906), there is

$$
\begin{aligned}
& \frac{\sum_{i=1}^{u} f_{aux}(\eta_i)}{u} = \frac{\sum_{i=1}^{u}\eta_i\log(u\eta_i)}{u} \\
& \geq f_{aux}(\frac{\sum_{i=1}^{u}\eta_i}{u}) = \frac{1}{u}\sum_{i=1}^{u}\eta_i\log(\sum_{i=1}^{u}\eta_i) = 0.
\end{aligned}
\qquad (6)
$$

The equality sign holds if and only if

$$\eta_1 = \eta_2 = ... = \eta_u. \qquad (7)$$

It means that $D_{KL}(p_{LM}||q) = \sum_{i=1}^{u} f_{aux}(\eta_i)$ takes the minimum value 0 when each component of $\boldsymbol{\eta}$ is equal, in which case p_{LM} and q are equivalent and that achieves the optimal information-theoretic security defined by Cachin (1998). □

Therefore, we basically construct the idea of our embedding algorithm, that is, to adaptively group the vocabulary into multiple groups at each time step, so that each group is assigned approximately the same probability. In practice, since the probability distribution is discrete, the probability of groups may not be absolutely equal. Firstly, we determine the number of groups u to be its maximum value $2^{\lfloor -\log_2 p_{max} \rfloor}$, where p_{max} is the highest probability in p_{LM}. Secondly, since the time complexity of solving the global optimal solution of equal grouping is unacceptable, we implement a suboptimal solution in ADG, as demonstrated in Algorithm 1. In line 10, we employ binary search algorithm to select the token that has the nearest probability of a given value. Our implementation enables us to obtain a unique grouping result for any p_{LM}, which ensures that the secret information can be extracted accurately and completely at the receiving end.

4.2 Recursion and Pruning

After obtaining the grouping results, we can select the group according to the next $\log u$ bits of secret information to be embedded and simply sample a token in the group to generate stegotext. As a matter of fact, we can also continue grouping the obtained groups to further enlarge the embedding capacity and recursively grouping the new groups until it is impossible to be equally participated (the normalized p_{max} of the current group is greater than 0.5). In order to improve the efficiency of the recursive grouping, we employ pruning strategy to remove the redundant grouping operations. We only need to recursively group the selected groups every time in accordance with the secret information to be embedded. In this manner, the amount of secret information embedded in each token is adjusted dynamically according to its probability distribution.

To sum up, at each time step, the proposed ADG embedding algorithm first conducts the equal

Algorithm 1: Suboptimal solution of equal grouping.

```
Data: vocabulary $\Sigma$, distribution $p_{LM}$
Result: set of groups $G$
list of tokens = sorted $(p_{LM})$;
$p_{max}$ = probability of the first token;
$u = 2^{\lfloor -\log_2 p_{max} \rfloor}$;
mean = $1/u$;
for ($i = 1; i \leq u - 1; i$ ++) do
    $G_i$ = [the first token];
    remove the first token;
    while $\sum$ probability of $G_i$ < mean do
        $\varepsilon$ = mean $- \sum$ probability of $G_i$;
        select a token with the nearest
          probability of $\varepsilon$;
        if probability of the token $- \varepsilon < \varepsilon$
          then
            append the token to $G_i$;
            remove the token;
        end
        else
            break;
        end
    end
    mean = $\frac{\text{probability of the rest tokens}}{u - i}$;
end
append the rest tokens to $G_u$;
$G = [G_1, G_2, ..., G_u]$;
```

grouping algorithm adaptively according to the conditional distribution, and then recursively repeats the operation on the selected group dynamically according to the secret information, until it is indivisible. At last, we normalize the probability of the last selected group and sample a token to generate the stegotext. We have proved the security of equal grouping algorithm. Obviously, it can also be extended to the recursive manner of ADG, which means the proposed method is provably secure.

4.3 Information Extraction

The extraction algorithm is basically the inverse process of the embedding algorithm. For an exactly successful extraction, Alice and Bob have to share the same language model, vocabulary and grouping algorithm. At each time step, Bob is supposed to recursively operate the same grouping algorithm as Alice do, and then select the group contains the current token in the stegotext. The index of the selected groups reveal the embedded secret information.

5 Experimental Results and Analysis

In this section, we evaluate the performance of ADG in terms of both embedding capacity and imperceptibility. Details of our experiments and the analysis of the results are present in the following subsections.

5.1 Datasets

We evaluated the performance of ADG on three public corpora, namely "Large Movie Review Dataset" (**Movie**) (Maas et al., 2011), "All the News" (**News**)[1] and "Sentiment140" (**Tweet**) (Go et al., 2009). Large movie review dataset is originally built for binary sentiment classification, containing 100,000 movie reviews in total crawled from IMDb[2]. "All the news" is a collection of publications of mainstream news media. Sentiment140 is also used in sentiment analysis tasks, which contains 1,600,000 tweets extracted from Twitter[3].

We converted the raw text to lowercase and removed HTML tags and most punctuations, then segmented it into sentences with NLTK tools (Loper and Bird, 2002). We filtered out sentences with length below 5 or above 200. For the convenience of training and evaluation, any token occurring less than 10 times was mapped to a special token "_UNK". We also added "_BOS" and "_EOS" at the beginning and end of each sentence to help training. Sentences in a batch were padded to the same length with a special padding token "_PAD". Finally, we divided the preprocessed corpora into training set and test set according to the ratio of 9:1. Statistics are demonstrated in Table 3.

5.2 Implementation Details

In experiments, we utilized LSTMs (Hochreiter and Schmidhuber, 1997) for word-level generation. We stacked 2 LSTM layers and the model was implemented with Pytorch (Paszke et al., 2017). The dimension of word embedding was set to be 350. Hidden states in LSTM were set to be 512-dimensional vectors. In the training procedure, we applied SGD algorithm together with Adam

---

[1] `https://www.kaggle.com/snapcrack/all-the-news`

[2] `https://www.imdb.com/`

[3] `https://twitter.com/`

Table 1: Results of ER, KLD_1 and KLD_2.

METHOD	Movie			News			Tweet		
	↑ER	**↓$\mathbf{KLD}_1$**	**↓$\mathbf{KLD}_2$**	**↑ER**	**↓$\mathbf{KLD}_1$**	**↓$\mathbf{KLD}_2$**	**↑ER**	**↓$\mathbf{KLD}_1$**	**↓$\mathbf{KLD}_2$**
Bins ($b = 1$)	1.000	2.497	27.595	1.000	2.742	26.331	1.000	2.431	19.519
Bins ($b = 2$)	2.000	2.338	33.206	2.000	2.593	35.207	2.000	2.421	17.604
Bins ($b = 3$)	3.000	2.319	29.778	3.000	2.592	55.781	3.000	2.429	21.286
Bins ($b = 4$)	4.000	2.439	54.155	4.000	2.550	87.441	4.000	2.314	27.230
Bins ($b = 5$)	5.000	2.503	73.075	5.000	2.500	116.857	5.000	2.482	29.171
Huffman ($k = 1$)	1.000	1.961	21.219	1.000	2.338	11.226	1.000	2.121	6.252
Huffman ($k = 2$)	1.824	1.433	13.199	1.824	1.751	8.793	1.841	1.586	5.208
Huffman ($k = 3$)	2.509	1.106	8.487	2.518	1.372	6.855	2.595	1.145	4.141
Huffman ($k = 4$)	3.145	0.819	6.334	3.224	1.084	5.419	3.266	0.880	3.197
Huffman ($k = 5$)	3.705	0.658	4.657	3.872	0.838	3.995	3.932	0.694	2.738
Patient-Huffman ($\delta = 1.0$)	1.125	0.327	**0.767**	0.809	0.256	**0.441**	0.988	0.298	**0.545**
Patient-Huffman ($\delta = 1.5$)	1.711	0.588	2.132	1.460	0.559	1.817	1.668	0.621	1.280
Patient-Huffman ($\delta = 2.0$)	2.129	0.819	4.564	1.905	0.808	3.497	2.201	0.908	2.445
Arithmetic ($h = 100$)	4.224	0.362	2.956	4.412	0.425	2.269	4.308	0.333	1.508
Arithmetic ($h = 200$)	4.651	0.240	2.321	4.908	0.295	1.688	4.805	0.253	1.749
Arithmetic ($h = 300$)	4.903	0.205	1.903	5.127	0.245	1.426	4.942	0.206	1.242
ADG	**5.147**	**0.033**	1.946	**5.650**	**0.027**	0.866	**5.411**	**0.048**	1.189

(Kingma and Ba, 2014) to train the language model. Learning rate was set to be 0.001. The SGD update direction was computed using a batch of 32 training samples. They were both trained for 30 epochs on one GeForce GTX 1080 GPU. In the generation procedure, we adopted the model performing best on test sets. All generated sentences must be longer than 5 and shorter than 200.

5.3 Baselines

We rebuilt Fang et al. (2017) (**Bins**), Yang et al. (2018a) (**Huffman**), Dai and Cai (2019) (**Patient-Huffman**) and Ziegler et al. (2019) (**Arithmetic**) as baselines. For fair comparison, we rebuilt all the baselines with the same language models. For Bins, we set b to be 1, 2, 3, 4, 5 and the corresponding number of bins was 2, 4, 8, 16, 32. For Huffman, we built Huffman tree with the top 2, 4, 8, 16, 32 likely tokens. For Patient-Huffman, we measured the distortion by KL divergence and restricted the threshold δ to 1, 1.5, 2 with top 8 tokens. For Arithmetic, we truncated the conditional distribution at $h = 100, 200, 300$. In each case, we generated 1,000 stegotext. We randomly chose same amount of covertext from the test sets for further evaluation.

5.4 Metrics

The metrics we utilized to evaluate the performance on embedding capacity and imperceptibility are listed as follows.

Embedding Rate (**ER**): It is the average amount of information that one single token can carry, and is in unit of *bits per word* (*bpp*). Embedding rate is a metric to indicate the embedding capacity. Higher is better.

KL Divergence between the implicit distribution q and the modeled distribution p_{LM} (**$\mathbf{KLD}_1$**): It reflects the gap introduced by the embedding algorithm. Lower is better and the unit is *bit*.

KL Divergence between the statistical distributions of the sentence embedding of covertext and stegotext (**$\mathbf{KLD}_2$**): It indirectly reflects the overall information-theoretic security. We mapped all stegotext and covertext to fixed length dense vectors $\boldsymbol{v_x}$ and $\boldsymbol{v_y}$ by third-party sentence vectorization tool (Le and Mikolov, 2014), and assumed that the resulting vectors of covertext and stegotext both obey isotropic Gaussian distribution. Then KLD_2 is computed by

$$\begin{aligned} & D_{KL}(p(\boldsymbol{v_x})\|p(\boldsymbol{v_y})) \\ & \approx \sum (\log \frac{\boldsymbol{\sigma_y}}{\boldsymbol{\sigma_x}} + \frac{\boldsymbol{\sigma_x}^2 + (\boldsymbol{\mu_x} - \boldsymbol{\mu_y})^2}{2\boldsymbol{\sigma_y}^2} - \frac{1}{2}), \end{aligned} \tag{8}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation of sentence vectors. We set the dimension of sentence vectors to be 100. Lower is better and the unit is *bit*.

Detection Accuracy: It reflects the anti-steganalysis ability of steganographic methods. Ste-

Table 2: Results of ER, Acc$_1$ and Acc$_2$.

METHOD	**Movie**			**News**			**Tweet**		
	↑ER	**Acc$_1$**	**Acc$_2$**	**↑ER**	**Acc$_1$**	**Acc$_2$**	**↑ER**	**Acc$_1$**	**Acc$_2$**
Bins ($b = 1$)	1.000	0.873	0.854	1.000	0.887	0.856	1.000	0.787	0.814
Bins ($b = 2$)	2.000	0.812	0.802	2.000	0.855	0.830	2.000	0.739	0.753
Bins ($b = 3$)	3.000	0.810	0.789	3.000	0.833	0.819	3.000	0.720	0.733
Bins ($b = 4$)	4.000	0.825	0.832	4.000	0.843	0.852	4.000	0.748	0.760
Bins ($b = 5$)	5.000	0.876	0.872	5.000	0.877	0.882	5.000	0.750	0.786
Huffman ($k = 1$)	1.000	0.891	0.891	1.000	0.891	0.885	1.000	0.785	0.806
Huffman ($k = 2$)	1.824	0.838	0.836	1.824	0.851	0.826	1.841	0.749	0.758
Huffman ($k = 3$)	2.509	0.796	0.760	2.518	0.816	0.785	2.595	0.684	0.702
Huffman ($k = 4$)	3.145	0.713	0.690	3.224	0.768	0.718	3.266	0.634	0.632
Huffman ($k = 5$)	3.705	0.673	0.645	3.872	0.710	0.664	3.932	0.602	0.593
Patient-Huffman ($\delta = 1.0$)	1.125	0.588	0.578	0.809	0.559	0.542	0.988	0.528	0.552
Patient-Huffman ($\delta = 1.5$)	1.711	0.654	0.683	1.460	0.674	0.683	1.668	0.589	0.581
Patient-Huffman ($\delta = 2.0$)	2.129	0.722	0.714	1.905	0.725	0.726	2.201	0.650	0.661
Arithmetic ($h = 100$)	4.224	0.601	0.582	4.412	0.630	0.608	4.308	0.547	0.554
Arithmetic ($h = 200$)	4.651	0.565	0.556	4.908	0.594	0.559	4.805	0.537	0.561
Arithmetic ($h = 300$)	4.903	0.571	0.562	5.127	0.558	0.566	4.942	0.532	0.534
ADG	**5.147**	**0.548**	**0.544**	**5.650**	**0.543**	**0.519**	**5.411**	**0.496**	**0.497**

Table 3: Statistics of the preprocessed datasets.

DATASET	**$\lvert\Sigma\rvert$**	**|TRAINING|**	**|TEST|**
Movie	37,800	1,002,609	111,402
News	50,178	1,461,567	162,397
Tweet	30,152	1,572,599	174,734

ganalysis is the technology used by Eve to detect hidden information in stegocarriers, which is the opposite direction of steganography. In our experiment, we employed linguistic steganalysis approaches based on Fasttext (Yang et al., 2019) (**Acc**$_1$) and TextCNN (Yang et al., 2020b) (**Acc**$_2$). We took stegotext as positive samples and covertext as negative samples. We conducted 10-fold cross validation and reported the average accuracy. Closer to 50% is better.

Effective Embedding Rate: It is a new metric we proposed to evaluate the comprehensive performance of steganographic algorithms. It is defined to be calculated by

$$\text{EER} = 2 \times (1 - Acc) \times \text{ER}, \tag{9}$$

meaning that if the stegotext has a certain probability of being detected, the average amount of secret information actually transmitted should be discounted accordingly. For mathematical rigorousness and completeness, if $Acc < 0.5$, we assign $1 - Acc$ to Acc. In extreme cases where the stegocarriers are completely natural, the detection accuracy should be 50% and EER is equal to ER. On the contrary, stegocarriers with 100% detection accuracy cannot carry a single bit. We calculated this metric with the accuracy results obtained by the two aforementioned steganalysis method (**EER**$_1$, **EER**$_2$). Higher is better and the unit is *bpp*.

5.5 Results and Analysis

The results of KLD$_1$ and KLD$_2$ are listed in Table 1. KLD$_1$ measures the distortion between q and p_{LM}, which is introduced by the embedding algorithm ADG. KLD$_2$ estimates the overall information-theoretic security that also considers the deviation of language models. In terms of KLD$_1$, we found that the proposed method ADG outperforms all baselines and it is very close to the optimal value 0 (stochastic sampling), which means generating stegotext by ADG is almost equivalent to normal generation with the language models. The results of KLD$_2$ are also advantageous, indicating that the generated stegotext is statistically consistent with the covertext. We noticed that some baselines can also perform well on KLD$_2$ (e.g. Patient-Huffman ($\delta = 1.0$)). However, they have a crucial flaw in embedding capacity.

Table 2 demonstrates the results of anti-steganalysis, where we found the tendency coheres with that of KLD$_1$ and KLD$_2$. The proposed method ADG outperforms all baselines on the three corpora and it is very close to the optimal value 0.5, which further confirms its imperceptibility. Besides, we also illustrated some examples of stegotext generated by ADG in Table 5 for qualitative

Table 4: Results of ER, EER_1 and EER_2.

METHOD	**Movie**			**News**			**Tweet**		
	↑**ER**	↑$\textbf{EER}_1$	↑$\textbf{EER}_2$	↑**ER**	↑$\textbf{EER}_1$	↑$\textbf{EER}_2$	↑**ER**	↑$\textbf{EER}_1$	↑$\textbf{EER}_2$
Bins ($b = 1$)	1.000	0.254	0.292	1.000	0.226	0.287	1.000	0.425	0.373
Bins ($b = 2$)	2.000	0.752	0.794	2.000	0.582	0.680	2.000	1.044	0.988
Bins ($b = 3$)	3.000	1.137	1.266	3.000	0.999	1.089	3.000	1.683	1.605
Bins ($b = 4$)	4.000	1.396	1.344	4.000	1.252	1.180	4.000	2.020	1.924
Bins ($b = 5$)	5.000	1.245	1.280	5.000	1.230	1.180	5.000	2.500	2.135
Huffman ($k = 1$)	1.000	0.218	0.219	1.000	0.219	0.231	1.000	0.430	0.387
Huffman ($k = 2$)	1.824	0.593	0.600	1.824	0.546	0.635	1.841	0.924	0.893
Huffman ($k = 3$)	2.509	1.024	1.202	2.518	0.927	1.083	2.595	1.638	1.549
Huffman ($k = 4$)	3.145	1.809	1.950	3.224	1.496	1.821	3.266	2.387	2.404
Huffman ($k = 5$)	3.705	2.427	2.627	3.872	2.249	2.602	3.932	3.133	3.200
Patient-Huffman ($\delta = 1.0$)	1.125	0.927	0.949	0.809	0.713	0.740	0.988	0.933	0.886
Patient-Huffman ($\delta = 1.5$)	1.711	1.182	1.083	1.460	0.952	0.925	1.668	1.369	1.400
Patient-Huffman ($\delta = 2.0$)	2.129	1.184	1.220	1.905	1.050	1.044	2.201	1.541	1.490
Arithmetic ($h = 100$)	4.224	3.371	3.527	4.412	3.269	3.459	4.308	3.908	3.843
Arithmetic ($h = 200$)	4.651	4.051	4.125	4.908	3.981	4.324	4.805	4.449	4.219
Arithmetic ($h = 300$)	4.903	4.207	4.290	5.127	4.532	4.450	4.942	4.630	4.606
ADG	**5.147**	**4.648**	**4.699**	**5.650**	**5.164**	**5.435**	**5.411**	**5.373**	**5.384**

Table 5: Examples of stegotext generated by ADG on the three corpora.

Movie	The supporting cast was also excellent.
	But I guess you 've seen the many silent movies along with his other films.
	And this movie was a precursor of val kilmer in the extreme.
	It 's a unique wonderful movie that deserves all the recognition it deserved.
	This is the worst movie I have ever seen.
News	The FBI estimated its total wealth on Thursday.
	Remember this is in part because of the actual policies of Donald Trump.
	He said he did not care about any counterintelligence investigation.
	Today however the process could not change even if he doesnt agree with Trumps rhetoric.
	More than 100 000 people have been detained and another 30 000 civilians have been wounded early on Sunday.
Tweet	Worst headache everrrr I dunno why but it was so scary.
	I had a blast today in the MTV Movie Awards.
	Ahhh some brothers do n't play sports!
	Sadly you will be missing so much.
	I do n't think the peach ice cream last night was good.

study. We found that the stegotext is fluent enough, with correct grammar and coherent semantics.

Finally, taking both embedding capacity and imperceptibility into account, we investigated effective embedding rate listed in Table 4. It can be concluded that our method has excellent comprehensive performance, which outperforms all baselines. In general, the experimental results indicate that the proposed method ADG is able to resist both perceptual and statistical steganalysis of Eve, meanwhile ensure a remarkable embedding rate, which reveals its effectiveness.

6 Conclusion

Previous works of generative linguistic steganography inevitably introduce distortions to the distribution estimated by off-the-shelf language models. In this paper, we attempted to achieve provably secure generative linguistic steganography during the procedure of stegotext generation. We proposed ADG, which embeds secret information by adaptive dynamic grouping. According to the mathematical proof and extensive experiments conducted on three public corpora, we found that the proposed method is provably secure and capable of generating fluent stegotext with high embedding capacity and high imperceptibility. We hope our investigation of provably secure generative linguistic steganography can be leveraged as a building block for future research.

Acknowledgments

This work is supported in part by the National Key Research and Development Program of China under Grant U1936216, Grant 61862002, Grant 6200197 and Grant U1705261. The authors thank anonymous reviewers for their insightful suggestions.

References

Ross J Anderson and Fabien AP Petitcolas. 1998. On the limits of steganography. *IEEE Journal on selected areas in communications*, 16(4):474–481.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Christian Cachin. 1998. An information-theoretic model for steganography. In *International Workshop on Information Hiding*, pages 306–318. Springer.

Mark Chapman and George Davida. 1997. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In *International Conference on Information and Communications Security*, pages 335–345. Springer.

Mark Chapman and George Davida. 2002. Plausible deniability using automated linguistic stegonagraphy. In *International Conference on Infrastructure Security*, pages 276–287. Springer.

Mark Chapman, George I Davida, and Marc Rennhard. 2001. A practical and effective approach to large-scale automated linguistic steganography. In *International Conference on Information Security*, pages 156–165. Springer.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308.

Weihui Dai, Yue Yu, Yonghui Dai, and Bin Deng. 2010. Text steganography system using markov chain source model and des algorithm. *JSW*, 5(7):785–792.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with lstms. In *Proceedings of ACL 2017, Student Research Workshop*, pages 100–106.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mehdi Hussain, Ainuddin Wahid Abdul Wahab, Yamani Idna Bin Idris, Anthony TS Ho, and Ki-Hyun Jung. 2018. Image steganography in spatial domain: A survey. *Signal Processing: Image Communication*, 65:46–66.

Johan Ludwig William Valdemar Jensen et al. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

R Bala Krishnan, Prasanth Kumar Thandra, and M Sai Baba. 2017. An overview of text steganography. In *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–6. IEEE.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Yunxia Liu, Shuyang Liu, Yonghao Wang, Hongguo Zhao, and Si Liu. 2019. Video steganography: A review. *Neurocomputing*, 335:238–250.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Yubo Luo, Yongfeng Huang, Fufang Li, and Chinchen Chang. 2016. Text steganography based on ci-poetry generation using markov chain model. *KSII Transactions on Internet and Information Systems (TIIS)*, 10(9):4568–4584.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Shilpi Mishra, Virendra Kumar Yadav, Munesh Chandra Trivedi, and Tarun Shrimali. 2018. Audio steganography techniques: A survey. In *Advances in Computer and Computational Sciences*, pages 581–589. Springer.

H Hernan Moraldo. 2014. An approach for text steganography based on markov chains. *arXiv preprint arXiv:1409.0915*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Shivani Sharma, Avadhesh Gupta, Munesh Chandra Trivedi, and Virendra Kumar Yadav. 2016. Analysis of different text steganography techniques: a survey. In *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, pages 130–133. IEEE.

Mohammad Shirali-Shahreza. 2008. Text steganography by changing words spelling. In *2008 10th international conference on advanced communication technology*, volume 3, pages 1912–1913. IEEE.

Gustavus J Simmons. 1984. The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, pages 51–67. Springer.

Peter Wayner. 1992. Mimic functions. *Cryptologia*, 16(3):193–214.

Lingyun Xiang, Yan Li, Wei Hao, Peng Yang, and Xiaobo Shen. 2018. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput. Mater. Continua*, 55(3):541–559.

Lingyun Xiang, Xingming Sun, Gang Luo, and Bin Xia. 2014. Linguistic steganalysis using the features derived from synonym frequency. *Multimedia tools and applications*, 71(3):1893–1911.

Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2018a. Rnn-stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295.

Zhong-Liang Yang, Si-Yu Zhang, Yu-Ting Hu, Zhi-Wen Hu, and Yong-Feng Huang. 2020a. Vae-stega: Linguistic steganography based on variational autoencoder. *IEEE Transactions on Information Forensics and Security*, 16:880–895.

Zhongliang Yang, Yongfeng Huang, and Yu-Jin Zhang. 2019. A fast and efficient text steganalysis method. *IEEE Signal Processing Letters*, 26(4):627–631.

Zhongliang Yang, Yongfeng Huang, and Yu-Jin Zhang. 2020b. Ts-csw: text steganalysis and hidden capacity estimation based on convolutional sliding windows. *Multimedia Tools and Applications*, 79(25):18293–18316.

Zhongliang Yang, Shuyu Jin, Yongfeng Huang, Yujin Zhang, and Hui Li. 2018b. Automatically generate steganographic text based on markov model and huffman coding. *arXiv preprint arXiv:1811.04720*.

Xuejing Zhou, Wanli Peng, Boya Yang, Juan Wen, Yiming Xue, and Ping Zhong. 2021. Linguistic steganography based on adaptive probability distribution. *IEEE Transactions on Dependable and Secure Computing*.

Zachary Ziegler, Yuntian Deng, and Alexander M Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215.

Retrieval Enhanced Model for Commonsense Generation

Han Wang[1*], **Yang Liu**[2], **Chenguang Zhu**[2], **Linjun Shou**[3],
Ming Gong[3], **Yichong Xu**[2], **Michael Zeng**[2]
[1]New York University
[2]Microsoft Cognitive Services Research Group
[3]STCA NLP Group, Microsoft, Beijing, China
`hwang@nyu.edu`
`{yaliu10,chezhu,lisho,migon,yicxu,nzeng}@microsoft.com`

Abstract

Commonsense generation is a challenging task of generating a plausible sentence describing an everyday scenario using provided concepts. Its requirement of reasoning over commonsense knowledge and compositional generalization ability even puzzles strong pre-trained language generation models. We propose a novel framework using retrieval methods to enhance both the pre-training and fine-tuning for commonsense generation. We retrieve prototype sentence candidates by concept matching and use them as auxiliary input. For fine-tuning, we further boost its performance with a trainable sentence retriever. We demonstrate experimentally on the large-scale CommonGen benchmark that our approach achieves new state-of-the-art results.[1]

1 Introduction

The understanding of commonsense knowledge in human language has been acknowledged as a critical component for artificial intelligence systems. In recent years, many new tasks and datasets are proposed to assess NLP model's ability of commonsense reasoning (Yu et al., 2020). SWAG (Zellers et al., 2018) is a task of inferring the upcoming event based on a partial description using commonsense. CommonsenseQA (Talmor et al., 2019) is a commonsense question answering dataset built from ConceptNet. Recently, Lin et al. (2020) propose CommonGen, a new challenge for evaluating model's ability of generative commonsense reasoning.

Concept Set #1:
dog, frisbee, catch, throw
Gold Target Sentences:
A dog leaps to catch a thrown frisbee. The dog catches the frisbee when the boy throws it. A man throws away his dog 's favorite frisbee expecting him to catch it in the air.
Concept Set #2:
lake, shore, canoe
Gold Target Sentences:
Canoe on a shore of lake. Canoe on shore with rainbow across the lake. Several canoes parked in the grass on the shore of a lake.

Table 1: Two concept sets and their gold corresponding sentences from CommonGen dataset.

CommonGen requires the system to construct a plausible sentence based on several concepts related to an everyday scenario. Two examples for this task are shown in Table 1. The task is challenging because the system needs to organize provided concepts into the most plausible scenario, avoid violation of commonsense, and ensure the generated sentence is grammatically correct. Existing approaches fine-tune pre-trained encoder-decoder models for description construction with concatenated concepts as input.

Fan et al. (2020) propose a retrieve-and-generation method for commonsense generation which uses a prototype candidate sentence as auxiliary input. However, their retriever is non-trainable and only works for the fine-tuning process. In this work, we extend this idea and propose a novel framework for commonsense generation by using retrieval method for enhancing both the pre-training and fine-tuning stages. Furthermore, we design a trainable prototype sentence retriever to further boost generation performance.

We conduct experiments on CommonGen (Lin et al., 2020) benchmark dataset. It contains 35,141 concept sets and 79,051 corresponding sentences. Each concept set is mapped to multiple corresponding sentences. Without any model modification or complex fusion of knowledge graphs, our approach

---

[*] Work done during internship at Microsoft.

[1] The code and data are available at `https://github.com/HanNight/RE-T5`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3056–3062
August 1–6, 2021.

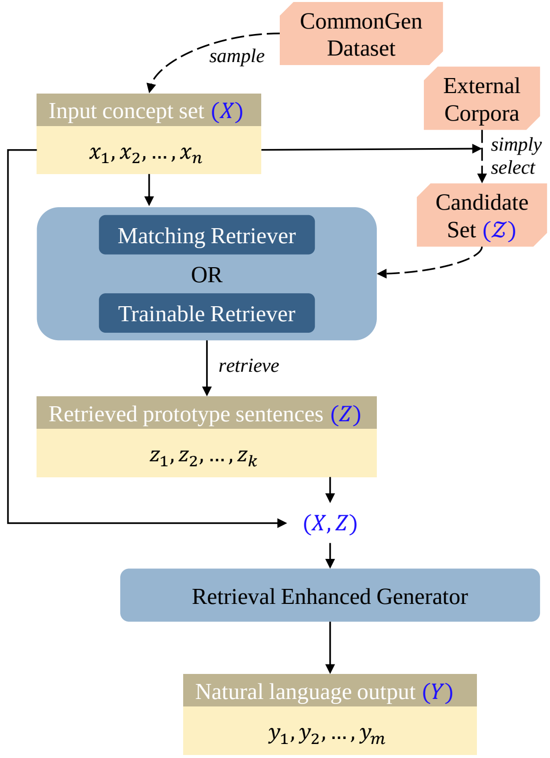

Figure 1: The overall framework of Retrieved Enhanced Model for Commonsense Generation.

achieves new state-of-the-art results on CommonGen on several metrics, including BLEU, CIDEr and SPICE.

2 Method

We frame CommonGen challenge as a sequence-to-sequence task and adopt T5 (Raffel et al., 2020), a powerful pre-trained encoder-decoder model, as our base model. Fan et al. (2020) find concepts-related sentences in external corpora can benefit relational reasoning for CommonGen. We extend this idea by proposing retrieval-enhanced T5 (RE-T5) which equips original T5 with a trainable retriever for selecting prototype sentences based on given concepts. Meanwhile, referring to (Zhou et al., 2021), we design a pre-training task for CommonGen which continue to pre-train RE-T5 on pseudo concept sets extracted from external corpora. We also use a retriever in this pre-training stage.

Formally, given a concept set $X = \{x_1, x_2, \ldots, x_n\}$, where x_i represents the i-th concept and n is the number of concepts, our goal is to generate a natural language output of tokens $Y = \{y_1, y_2, \ldots, y_m\}$, which describes a common scenario in our daily life, using all given concepts in X.

2.1 Retrieval

Since external corpora have lots of scenario knowledge to describe the relationship between concepts (Fan et al., 2020), we retrieve sentences related to input concepts to help the model perform better commonsense reasoning. First, given an input concept set, we extract all sentences from external corpora that contain at least two concepts in the input X as candidate set $\mathcal{Z}$. Then, we design two retrieval models, *matching retriever* and *trainable retriever*, to further retrieve k prototype sentences $Z = \{z_1, z_2, \ldots, z_k\}, Z \subseteq \mathcal{Z}$ as auxiliary input context for RE-T5.

Matching Retriever The *matching retriever* first orders candidate sentences by the number of contained concepts. Then it simply samples k sentences starting from sentences that contained the most concepts as the auxiliary input.

Trainable Retriever In order to retrieve more useful sentences from the sentence candidate set, we design a *trainable retriever*, which predicts scores to rank these candidates, and then select top-k sentences as additional context. The scorer is built based on BERT (Devlin et al., 2019), a pre-trained language model usually used for language understanding. Given a concept set X and a candidate sentence z_i, our trainable retriever first concatenate them into a text input:

$$\texttt{[CLS]}\, X \,\texttt{[SEP]}\, z_i \,\texttt{[SEP]}$$

where `[CLS]` and `[SEP]` are special symbols in BERT.

We pass this into BERT, which generates an output vector for each input token. We take the output vector corresponding to `[CLS]` which is used as the aggregated representation of the input sequence (denoted $\boldsymbol{c}$) into a linear layer with sigmoid activation to obtain the binary classification output y_c.

$$y_c = \sigma(\boldsymbol{W}_c \boldsymbol{c} + b_c) \tag{1}$$

where $\boldsymbol{W}_c$ is a projection matrix and b_c is a bias.

To train this retriever, for each concept set in CommonGen training set, we use its paired sentence as a positive example and we randomly sample another sentence, also from the training set, as a negative example. Then, we adopt cross entropy loss for this binary classification. The top-k scored sentences with the highest scores will be selected as the auxiliary input Z.

Model	BLEU-4	CIDEr	**SPICE**	SPICE(v1.0)
GPT-2 (Radford et al., 2019)	26.833	12.187	23.567	25.90
BERT-Gen (Bao et al., 2020)	23.468	12.606	24.822	27.30
UniLM (Dong et al., 2019)	30.616	14.889	27.429	30.20
BART (Lewis et al., 2020)	31.827	13.976	27.995	30.60
T5-base (Raffel et al., 2020)	18.546	9.399	19.871	22.00
T5-large (Raffel et al., 2020)	31.962	15.128	28.855	31.60
EKI-BART (Fan et al., 2020)	35.945	16.999	29.583	32.40
KG-BART (Liu et al., 2021)	33.867	16.927	29.634	32.70
CALM(T5-base) (Zhou et al., 2021)	-	-	-	33.00
RE-T5 (ours)	**40.863**	**17.663**	**31.079**	**34.30**

Table 2: Test results on CommonGen benchmark. All results except CALM are based on the latest human references(v1.1). v1.0 indicates evaluation with old evaluation protocol.[2]

We will describe how these two retrievers are used in CommonGen pre-training and fine-tuning stages.

2.2 Pre-training

To enhance model's ability of commonsense reasoning, we design a pre-training task for RE-T5 which is similar to original CommonGen task. In more details, given a sentence from external corpora, we first use spaCy (Honnibal et al., 2020) to tag the sentences with part-of-speech and extract *Verb*, *Noun* and *Proper Nouns* as pseudo concept phrases. We then only keep phrases in ConceptNet (Speer et al., 2017) and remove concept-sets that appear in CommonGen's testset. We use the original sentence as the target sentence, and constructs a pre-training task of using RE-T5 to generate this sentence given pseudo concepts.

Due to the extraction method for pseudo concepts, when retrieving prototype sentences, for each concept set in pre-training data, we have a large candidate set $\mathcal{Z}$ with an excessive number of candidate sentences. This leads to a long inference time for using the trainable retriever. Thus, due to speed consideration and also to introduce a degree of randomness into pre-training, we use the *matching retriever* to retrieve k sentences as auxiliary input Z.

After retrieval, RE-T5 takes the concatenation of input concepts and retrieved prototype sentences as input, and the original sentence as output.

2.3 Fine-tuning

At fine-tuning stage, we use *trainable retriever* to score sentences from candidate set $\mathcal{Z}$ and select top k sentence as additional context Z. Similar to pre-training, RE-T5 takes the concatenation of input concepts and retrieved prototype sentences as input, and the original sentence as output.

3 Experiments

3.1 Experiments Settings

Dataset CommonGen is a benchmark dataset designed to diagnose whether a model has the ability of generative commonsense reasoning (Lin et al., 2020). This dataset contains 32,651/993/1,497 concept sets for training/development/test, and the numbers of corresponding sentences are 67,389/4,018/7,644. We use BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016) as evaluation metrics. Because SPICE correlates the most with human evaluation (Lin et al., 2020), we take SPICE as the primary metric.

External Corpora To be consistent with the distribution of the CommonGen dataset, we use VATEX (Wang et al., 2019), Activity (Krishna et al., 2017), SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) as external corpora. We sample 500k sentences from these corpora to construct our pre-training dataset. Meanwhile, these datasets are also used as our sentence pool for the retrieval module. For both the pre-training and fine-tuning, all sentences that appear in the CommonGen targets are not used as retrieval sentences candidates.

Baselines We compare RE-T5 with several baseline systems. GPT-2, BERT-Gen, UniLM, BART, and T5 are pre-trained language models tested in

---

[2] `https://inklab.usc.edu/CommonGen/leaderboard.html`

Concept Set: trailer shirt side sit road
T5: A man sits on the side of a trailer and a shirt.
Matching Retriever: (1)Two guys in red shirts are sitting on chairs, by the side of the road, behind that open trailer. (2)Two men, one wearing a straw cone hat, blue shirt, talking with a guy in a tan sunhat, red plaid shirt, both with baskets in front of them, sitting on the side of a dirt road. (3)An older guy with a tan shirt and hat sitting on the side of a road with bricks all around him and a small green bowl on the side. **RE-T5(matching retriever)**: a man in a tan shirt sits on the side of a road.
Trainable Retriever: (1)Two guys in red shirts are sitting on chairs, by the side of the road, behind that open trailer. (2)Teenagers in matching shirts stand at the side of the road holding trash bags. (3)A man in a white shirt and black pants standing at the side or the road. **RE-T5(trainable retriever)**: a man in a white shirt and black pants sits on the side of a trailer on the road.

Table 3: An example of sentences retrieved by different retrievers and sentences generated based on them.

(Lin et al., 2020). They are all fine-tuned on CommonGen training set with concatenated concepts as input and description sentence as output. EKI-BART (Fan et al., 2020) is a retrieve-and-generate framework for CommonGen, where they use a simple retriever to enhance pre-trained BART (Lewis et al., 2020). KG-BART (Liu et al., 2021) augment BART with Knowledge Graph on both the encoder and decoder side and continue to pre-train BART with a masked concept token generation task. CALM (Zhou et al., 2021) designs several self-supervised strategies encouraging model to focus on concept-centric information.

Implementation Details We adopt the T5-base as the generation model and BERT-base as the trainable retriever in fine-tuning. We use the Huggingface Transformer (Wolf et al., 2020) for model implementation. For pre-training phase, we use the AdamW optimizer (Loshchilov and Hutter, 2019) with an initial learning rate of 2e-6, weight decay 0.01, adam epsilon 1e-6, and a warmup fraction of 0.01. The model is pre-trained for 3 epochs, with batch size of 16, and gradient accumulation of 4 batches. For fine-tuning, the models are optimized using AdamW with an initial learning rate of 5e-5, batch size 64, gradient accumulation 3 and warmup fraction 0.01, and trained for 20 epochs. Meanwhile, the BERT-base scorer is optimized using AdamW optimizer with an initial learning rate 2e-5, batch size 64, and the model is trained for 3 epochs. For the number of the retrieved sentences k, we experimentally choose 3. All experiments are conducted using 4 V100 with 32 GB memory.

Model	**SPICE**
Retrieve (only)	29.60
T5	30.80[3]
T5 + *MR*	33.60
T5 + *MR* + pretrain	33.90
RE-T5 (T5 + *TR* + pretrain)	**34.30**

Table 4: Ablation results on the test set of CommonGen with T5-base as a backbone model. Note that *MR* denotes *Matching Retriever* and *TR* denotes *Trainable Retriever*.

3.2 Results

Table 2 shows results of different approaches on the CommonGen testset. RE-T5 outperforms all previous approaches by a large margin in all metrics and sets a new state of the art. RE-T5 combines the generation flexibility of pre-trained language models with the interpretability and modularity of a retrieval-based approach. Unlike EKI-BART (Fan et al., 2020) and KG-BART (Liu et al., 2021), RE-T5 enjoys strong results without model architecture modification. It is worth noting that although T5-base baseline does not perform as well as BART (Lewis et al., 2020) baseline, our method still outperforms the two improved BART-based methods mentioned above. RE-T5 demonstrates that for state-of-the-art performance, neither model modification nor complex fusion of knowledge graphs is necessary, only a simple and effective trainable retriever is needed.

---

[3] This is our reproduced result of T5-base. The difference from the result on the leaderboard is also observed in other papers (Zhou et al., 2021; Fan et al., 2020).

Ablation Study We conduct ablation experiments as shown in Table 4. First, we can see that RE-T5 model outperforms the backbone T5 model by a large margin in all metrics, with 3.5 improvement in the main metric SPICE. The second line of Table 4 shows that, although large-scale pre-trained language models have been shown to learn and store a substantial amount of the world knowledge implicitly from the massive text corpora (Petroni et al., 2019), the retrieved sentences from external corpora can still explicitly expose lots of scenario knowledge to describe the relationship between concepts. The third line indicates that further pre-training with data augmentation is helpful to improve the performance of the model. In addition, the last line demonstrates that a trainable scorer can capture more helpful knowledge for the model for commonsense generation.

Example Analysis Through the example in Table 3, we can observe that the baseline model T5 generates a sentence without concept *"road"*, and the juxtaposition between *"trailer"* and *"shirt"* in this sentence is not in line with common sense. For both *matching retriever* and *trainable retriever*, the retrieved sentences remind the model not to forget the concept *"road"*, in addition to providing the relationship between *shirt* and *person*. Since *matching retriever* randomly retrieves sentences based on the number of concepts they contain, it tends to retrieve longer sentences to contain as many concepts as possible, which may confuse the model and thus ignore some concepts, for example, the sentence generated by RE-T5 (matching retriever) in this example is missing the concept *"trailer"*. RE-T5 (trainable retriever) can solve the above problems and generate a sentence that is fluent and in line with common sense.

4 Conclusions

In this paper, we empirically investigated RE-T5, which utilizes a trainable retriever to retrieve sentences from external corpora to enhance the generative commonsense reasoning capability of pre-trained language models, such as T5. The state-of-the-art result achieved by RE-T5 on CommonGen benchmark demonstrates that a simple yet effective trainable retriever can be a useful addition to pre-trained language models for commonsense generation. For future work, we would like to explore the possibility of extending this simple and effective retrieval-based method to more tasks. In addition, we will also try training a more advanced retrieval model to further improve the performance of commonsense generation.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *Computer Vision – ECCV 2016*, pages 382–398, Cham. Springer International Publishing.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. UniLMv2: Pseudo-masked language models for unified language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 642–652. PMLR.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, volume 32, pages 13063–13075. Curran Associates, Inc.

Zhihao Fan, Yeyun Gong, Zhongyu Wei, Siyuan Wang, Yameng Huang, Jian Jiao, Xuanjing Huang, Nan Duan, and Ruofei Zhang. 2020. An enhanced knowledge injection model for commonsense generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2014–2025, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. 2019. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical*

Methods in Natural Language Processing, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, and Xiang Ren. 2021. Pre-training text-to-text transformers for concept-centric common sense. In *International Conference on Learning Representations*.

Decoupled Dialogue Modeling and Semantic Parsing for Multi-Turn Text-to-SQL

Zhi Chen, Lu Chen[*], Hanqi Li, Ruisheng Cao, Da Ma, Mengyue Wu and Kai Yu[*]
X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
Shanghai Jiao Tong University, Shanghai, China
State Key Lab of Media Convergence Production Technology and Systems, Beijing, China
{zhenchi713, chenlusz, kai.yu}@sjtu.edu.cn

Abstract

Recently, Text-to-SQL for multi-turn dialogue has attracted great interest. Here, the user input of the current turn is parsed into the corresponding SQL query of the appropriate database, given all previous dialogue history. Current approaches mostly employ end-to-end models and consequently face two challenges. First, dialogue history modeling and Text-to-SQL parsing are implicitly combined, hence it is hard to carry out interpretable analysis and obtain targeted improvement. Second, SQL annotation of multi-turn dialogue is very expensive, leading to training data sparsity. In this paper, we propose a novel decoupled multi-turn Text-to-SQL framework, where an utterance rewrite model first explicitly solves completion of dialogue context, and then a single-turn Text-to-SQL parser follows. A dual learning approach is also proposed for the utterance rewrite model to address the data sparsity problem. Compared with end-to-end approaches, the proposed decoupled method can achieve excellent performance without any annotated in-domain data. With just a few annotated rewrite cases, the decoupled method outperforms the released state-of-the-art end-to-end models on both SParC and CoSQL datasets.

1 Introduction

Text-to-SQL has lately become an interesting research topic along with the high demand to query a database using natural language (NL). Standard large database format can only be accessed with Structured Query Language (SQL), which requires certain special knowledge from users, hence lowering the accessibility of these databases. Text-to-SQL tasks, however, greatly minimize this gap and allow the query based on NL. Previous work on Text-to-SQL mostly focuses on single-turn utterance inference, evaluated on context-independent Text-to-SQL benchmarks. Nevertheless, in practice, the users usually need to interact with the Text-to-SQL system step-by-step to address their query purpose clearly. Under such conversation scenarios, the co-reference and information ellipses are always present, shown in Figure 1. Recently proposed methods are mostly end-to-end, which endeavors to design a suitable model to encode the dialogue context and infer the corresponding SQL based on the whole dialogue context. The main limitation of the end-to-end multi-turn Text-to-SQL models lies in their extreme reliance on annotated multi-turn Text-to-SQL data. The large-scale multi-turn Text-to-SQL data is time-consuming and expensive. The annotators not only need to be SQL experts but also have to infer the complete and exact query intent of the latest utterance of the speaker.

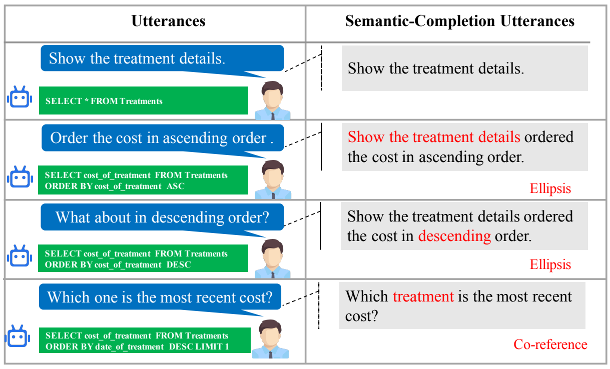

Utterances		Semantic-Completion Utterances	
Show the treatment details.	SELECT * FROM Treatments	Show the treatment details.	
Order the cost in ascending order .	SELECT cost_of_treatment FROM Treatments ORDER BY cost_of_treatment ASC	Show the treatment details ordered the cost in ascending order.	Ellipsis
What about in descending order?	SELECT cost_of_treatment FROM Treatments ORDER BY cost_of_treatment DESC	Show the treatment details ordered the cost in descending order.	Ellipsis
Which one is the most recent cost?	SELECT cost_of_treatment FROM Treatments ORDER BY date_of_treatment DESC LIMIT 1	Which treatment is the most recent cost?	Co-reference

Figure 1: An example to demonstrate the co-reference and ellipsis phenomenon in a conversation, where the right column shows the annotated semantic-completion utterances.

Different from previous end-to-end approaches, we propose a **DE**coupled mu**L**ti-**T**urn p**A**rsing (**DELTA**) framework, which decouples the multi-turn Text-to-SQL into two subsequent pipeline

---

[*] The corresponding authors are Lu Chen and Kai Yu.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3063–3074
August 1–6, 2021. ©2021 Association for Computational Linguistics

tasks: utterance rewrite and single-turn Text-to-SQL. In recent years, these two individual tasks are both well-studied. The utterance rewrite task aims to generate the latest semantic-completion question based on the dialogue context. The single-turn Text-to-SQL task aims to parse the semantic-completion question to a SQL, where the state-of-the-art methods (Shi et al., 2020; Yu et al., 2020b; Chen et al., 2021; Rubin and Berant, 2021) can achieve over 70% exact match accuracy on Spider (Yu et al., 2018) (a cross-domain single-turn Text-to-SQL dataset) and even achieve more than 80% on easier Text-to-SQL benchmarks (Dahl et al., 1994; Zhong et al., 2017). However, there is no rewrite data on the existing multi-turn Text-to-SQL benchmarks and the existing utterance rewrite datasets normally pay more attention to the co-reference problem but ignore the information ellipses. Due to the limitation of the in-domain annotated rewrite data, we further propose a dual learning method to make comprehensive use of the unlabeled multi-turn data to learn a reliable rewrite model. Our proposed framework DELTA is evaluated on both the SParC (Yu et al., 2019b) and CoSQL datasets (Yu et al., 2019a), the two existing large-scale benchmark for the multi-turn Text-to-SQL task.

Contributions are highlighted below:

- We propose a decoupled parsing framework for the multi-turn Text-to-SQL task, whose annotated data is much easier to collect. Even without any in-domain multi-turn Text-to-SQL data, the decoupled parsing method can achieve encouraging results on multi-turn Text-to-SQL benchmarks.

- The decoupled framework includes an utterance rewrite model which is adapted from the pretrained BART (Lewis et al., 2020), with a newly implemented dual learning method to make comprehensive use of the unlabeled multi-turn data. Our adapted rewrite model achieves new state-of-the-art performance on the utterance rewrite benchmarks.

- With fully labeled multi-turn Text-to-SQL data, our decouple parsing method outperforms all the released end-to-end multi-turn Text-to-SQL model.

2 Decoupled Parsing Framework

In this section, we elaborate our decoupled parsing framework, which consists of two phases: 1) an *utterance rewrite model* (Section 2.1), to generate semantic-completion question based on the dialogue context; 2) a *single-turn Text-to-SQL parser* (Section 2.2), which is fed with the rewritten question to predict the corresponding SQL query. To further improve the rewrite model performance, we propose a dual learning method to make use of large-scale unlabeled data, which is detailed in Section 3.

2.1 Phase-I: BART as Rewrite Model

We leverage the pretrained BART (Lewis et al., 2020), which is a Transformer-based encoder-decoder architecture, as the utterance rewrite model. This idea is inspired by its success on the text generation tasks, including question answering and summarization. Along with the success of pretrained language models, the Transformer architecture has been widely applied in natural language process (NLP) tasks. Transformer aims to encode a sequence $X = [x_i]_{i=1}^n$ with the *self-attention* mechanism (Vaswani et al., 2017). Assume that $[\mathbf{x}_i^{(l)}]_{i=1}^n$ is the representation of the sequence X at (l)-th Transformer layer. The next Transformer takes the following operations with H attention heads:

$$
\begin{aligned}
\alpha_{ij}^{(h)} &= \operatorname*{softmax}_j \left(\frac{\mathbf{x}_i^{(l)} \mathbf{W}_Q^{(h)} \left(\mathbf{x}_j^{(l)} \mathbf{W}_K^{(h)} \right)^\top}{\sqrt{d_z/H}} \right), \\
\mathbf{z}_i &= \Big\Vert_{h=1}^{H} \sum_{j=1}^{n} \alpha_{ij}^{(h)} \mathbf{x}_j^{(l)} \mathbf{W}_V^{(h)}, \\
\bar{\mathbf{x}}_i^{(l+1)} &= \mathrm{LN}\left(\mathbf{x}_i^{(l)} + \mathbf{z}_i \right), \\
\mathbf{x}_i^{(l+1)} &= \mathrm{LN}\left(\bar{\mathbf{x}}_i^{(l+1)} + \mathrm{FFN}\left(\bar{\mathbf{x}}_i^{(l+1)} \right) \right), \qquad (1)
\end{aligned}
$$

where h is the head index, d_z is the hidden dimension, $\alpha_{ij}^{(h)}$ is attention probability, $\Vert$ denotes the concatenation operation, $\mathrm{LN}(\cdot)$ is layer normalization (Ba et al., 2016) and $\mathrm{FFN}(\cdot)$ is a feed-forward network consists of two linear transformations.

Similar to other large-scale pretrained language models, BART also uses a standard Transformer-based sequence-to-sequence architecture, where the encoder is the bidirectional Transformer and the decoder is the auto-regressive Transformer. BART's pretraining method reconstructs the original text from its corrupted text. In its essence, BART is a denoising autoencoder, which is applicable to a very wide range of NLP tasks. In our utterance rewrite task, both the co-reference and information ellipses can be regarded as the corrupted noise of an utterance. Based on this idea, BART can be an appropriate method to denoise the co-reference and information ellipses. In addition, the rewrite data in the public multi-turn Text-to-SQL benchmarks are lacking. Therefore, we propose a dual learning method to learn a reliable rewrite model with large-scale unlabeled dialogue data. The details are introduced in Section 3.

2.2 Phase-II: RATSQL as Parsing Model

Given a natural language question and a schema for a relational database, the goal of Text-to-SQL parser is to generate the corresponding SQL query. Regarding the single-turn Text-to-SQL parsing model, we directly use the current state-of-the-art RATSQL model (Wang et al., 2020). RATSQL provides a unified framework, which is based on a relation-aware Transformer (RAT), to encode the question and the corresponding schema. The relation-aware Transformer is an important extension to the traditional transformer, which takes the input sequence as a labeled, directed, fully-connected graph. The pairwise relations between input elements are considered in RAT. RAT incorporates the relation information in Equation 1. The edge from element $\mathbf{x}_i$ to element $\mathbf{x}_j$ is represented by vector $\mathbf{r}_{ij}$, which is represented as biases incorporated in the Transformer layer, as follows:

$$\alpha_{ij}^{(h)} = \underset{j}{\operatorname{softmax}} \left(\frac{\mathbf{x}_i^{(l)} \mathbf{W}_Q^{(h)} \left(\mathbf{x}_j^{(l)} \mathbf{W}_K^{(h)} + \mathbf{r}_{ij} \right)^{\top}}{\sqrt{d_z / H}} \right),$$

$$\mathbf{z}_i = \Big\Vert_{h=1}^{H} \sum_{j=1}^{n} \alpha_{ij}^{(h)} \left(\mathbf{x}_j^{(l)} \mathbf{W}_V^{(h)} + \mathbf{r}_{ij} \right).$$

The relations among the Text-to-SQL input elements can be categorized into three types: intra-question, question-schema, and intra-schema. The intra-question relation means both tokens are the elements of the question. The question-schema relations are normally named by schema linking, which is used to represent the matching degree between the question token and the schema token. The intra-schema relations include the relation types of the relational database: primary key, foreign key, etc. However, these relations within the input elements are independent of the domain information of the database. Incorporating the domain-independent relations into the representation of the Text-to-SQL input is thus beneficial to the Text-to-SQL parser generation.

During decoding, the SQL query is first represented as an abstract syntax tree (AST) following a well-designed grammar. Followed by that, the AST is flattened as a sequence by the deep-first search (DFS) method. RATSQL uses the LSTM to generate the flattened AST sequence. The generated actions defined by the grammar has two structures: (1) it expands the last generated node into a grammar rule, called APPLYRULE or when completing a leaf node; (2) alternatively, it selects a column/table from the schema, called SELECTCOLUMN and SELECTTABLE.

3 Dual Learning for Utterance Rewrite

Due to the limitation of the in-domain annotated rewrite data, we propose a semi-supervised learning method via dual learning to make full use of the unlabeled multi-turn data to learn a reliable rewrite model. In this section, we first introduce the primal and dual tasks of the utterance rewrite. We then demonstrate the dual learning algorithm for utterance rewrite in detail, where a large amount of unlabeled utterance rewrite data participate in optimizing the primal and dual models under the dual learning framework.

3.1 Primal and Dual Tasks

In a conversation scenario, the co-reference and information ellipses are always present in the user's expressions (Androutsopoulos et al., 1995). Recently, Liu et al. (2020a) make a significant step to analyze the co-reference and ellipsis phenomenon at the fine-grained level. Co-reference has been divided into five types according to the existing pronoun: *Bridging Anaphora*, *Definite Noun Phrases*, *One Anaphora*, *Demonstrative Pronoun*, and *Possessive Determiner*. Ellipsis has been characterized by its intention: *Continuation* and *Substitution*. where the substitution can be further classified into 4 types: *explicit* vs. *implicit* and *schema* vs. *operator*. The detailed introduction of these fine-grained types refers to (Liu et al., 2020a).

The primal task aims to denoise the above co-reference and ellipsis and generate a semantic-completion utterance $c^{(t)} = [c^{(t,i)}]_{i=1}^{n}$ based on the utterance $x^{(t)} = [x^{(t,i)}]_{i=1}^{m}$ at the t-th turn and the dialogue history $h = [x^{(j)}]_{j=1}^{t-1}$. We directly use the pretrained BART as the rewrite model (named **rewriter**). We further concatenate the dialogue history and the latest utterance as the input of the rewriter, where they are separated by the special

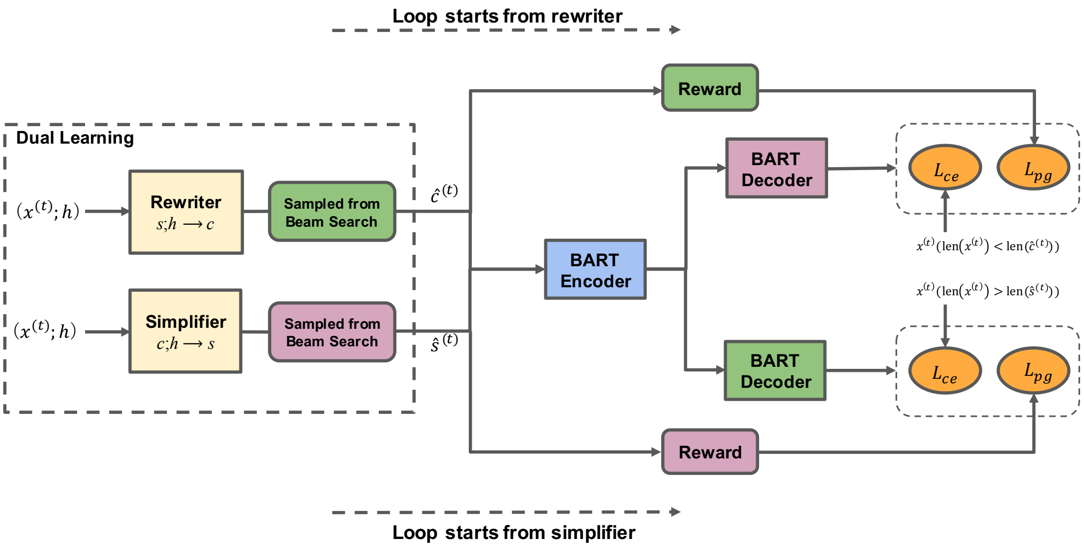

Figure 2: The dual learning architecture of the utterance rewrite task. Rewriter and Simplifier represent the dual models: rewrite model and simplification model. Both two models are initiated by pretrained BART. During the dual learning closed-loop game, two models are updated by policy gradient loss (L_{pg}) and cross-entropy loss (L_{ce}).

token "</s>". The dual task is to generate a simplified expression based on the latest utterance and the dialogue history. The simplified expression contains the above co-reference and ellipsis as more as possible without changing the original semantic meaning of the dialogue. Similar to the rewriter, we use the pretrained BART as the initial simplification model (named **simplifier**).

3.2 Dual Learning Algorithm

Under the dual learning framework, the dual models can be regarded as two agents in a closed-loop game. The game starts with one of the dual agents. The output of the start agent will be scored by an external reward function. Since the reward feedback is non-differentiable, the start agent is optimized by the policy gradient method (Sutton et al., 1999). The end agent is fed with the output of the start agent, where the end agent aims to reconstruct the initial input of the start agent. Thus, the end agent can be optimized by maximum likelihood estimation (MLE). Before deep-diving into the dual learning algorithm, we first introduce the definitions of the dual framework for the utterance rewrite.

3.2.1 Definition

Suppose we have unlabeled dialogue data $\mathcal{D}_u = \{(x^{(t)}; h)\}$. There are two dual models: rewriter with parameter Θ_c and simplifier with parameter Θ_s. Two language models ($\mathrm{LM}_c(\cdot)$ and $\mathrm{LM}_s(\cdot)$) are used to evaluate the quality of the generated utterances by rewriter and simplifier respectively. Both two language models are fine-tuned from GPT-2 model (Radford et al., 2019). $\mathrm{LM}_c(\cdot)$ is trained with semantic-completion Spider dataset. $\mathrm{LM}_s(\cdot)$ is trained from multi-turn Text-to-SQL data (SParC and CoSQL), where the utterances at the first turn are removed. There is an external single-turn Text-to-SQL parser $\mathrm{RATSQL}(\cdot)$, which parses a question into a SQL query. Next, we will introduce the strategy of agent optimization under dual learning framework.

3.2.2 Loop Starts from Rewriter

As shown in Fig. 2, we sample an unlabeled dialogue data $(x^{(t)}; h)$ from $\mathcal{D}_u$. The rewriter generates k possible rewritten formats $[\hat{c}_i^{(t)}]_{i=1}^k$ with beam search mechanism. There are two-level external reward functions to evaluate the quality of generated $\hat{c}_i^{(t)}$: token-level reward and sentence-level reward.

Token-level Reward To reserve the schema information of the database mentioned in original utterance $x^{(t)}$, the generated token $\hat{c}_i^{(t,j)}$ will get $+0.1$ reward at j-th step when it is database-related token mentioned in $x^{(t)}$. To decrease the co-reference phenomenon in the rewritten utterance, we punish the generated pronoun words (e.g., it, their, and so on) with -0.1. Otherwise, the generated tokens will get zero points.

Sentence-level Reward We first use the pretrained language model $\mathrm{LM}_c(\cdot)$ to evaluate the quality of the rewritten utterance with $r_{\mathrm{LMc}} = \log(\mathrm{LM}_c(\hat{c}_i^{(t)}))/\mathrm{len}(\hat{c}_i^{(t)})$, where $\mathrm{len}(\hat{c}_i^{(t)})$ denotes the number of the tokens in $\hat{c}_i^{(t)}$. In practice, the rewritten utterance $\hat{c}_i^{(t)}$ can be directly evaluated by the user, who does not need any SQL background. The user can give an indicated score (0 or 1) to evaluate whether $\hat{c}_i^{(t)}$ meets his/her real intent. Instead, we feed the rewritten utterance $\hat{c}_i^{(t)}$ into Text-to-SQL parser and get the corresponding SQL query with $\hat{q} = \mathrm{RATSQL}(\hat{\mathrm{c}}_{\mathrm{i}}^{(\mathrm{t})})$. If $\hat{q}$ equals to the golden SQL, we can say the rewritten utterance meets the user's intent ($r_{\mathrm{u}} = 1$) and vise versa ($r_{\mathrm{u}} = 0$). The final sentence-level reward of $\hat{c}_i^{(t)}$ is represented as $r_i^c = r_{\mathrm{LMc}} + r_{\mathrm{u}}$.

For the j-th token in the rewritten utterance $\hat{c}_i^{(t)}$, its accumulated reward can be represented as $R_i^{(t,j)} = r_i^{(t,j)} + \sum_{l=j+1}^{m} \lambda^{l-j} r_i^{(t,l)}$, where λ is discount rate, $r_i^{(t,j)}$ means the j-th token reward of the rewritten utterance and the final token reward equals to the sentence-level reward $r_i^{(t,m)} = r_i^c$. The rewriter can be optimized by policy gradient method as:

$$L_{pg}(\Theta_c) = -\sum_{i,j=1}^{k,m} R_i^{(t,j)} \log\left(P(\hat{c}_i^{(t,j)} | (x^{(t)}; h); \Theta_c)\right).$$

To force the simplifier to reconstruct the original input $x^{(t)}$ as similar as possible, the simplifier can be optimized with maximum likelihood estimation (MLE) as:

$$L_{ce}(\Theta_s) = -\log\left(P(x^{(t)} | (\hat{c}_i^{(t)}; h); \Theta_s)\right).$$

Noting that $x^{(t)}$ could be a semantic-completion utterance. It is not reasonable to force the simplifier to reconstruct a semantic-completion utterance. Thus, we first compare the length of the original utterance $\mathrm{len}(x^{(t)})$ with the length of the rewritten one $\mathrm{len}(\hat{c}_i^{(t)})$. Only when $\mathrm{len}(x^{(t)}) < \mathrm{len}(\hat{c}_i^{(t)})$, we optimize the simplifier with MLE.

3.2.3 Loop Starts from Simplifier

As shown in Fig. 2, we also sample an unlabeled dialogue data $(x^{(t)}; h)$ from $\mathcal{D}_u$. The simplifier generates k possible simplified formats $[\hat{s}_i^{(t)}]_{i=1}^k$ with beam search mechanism evaluated by two-level external reward functions.

Token-level Reward To decrease the schema information mentioned in original utterance $x^{(t)}$, the generated token $\hat{s}_i^{(t,j)}$ will get -0.1 punishment at j-th step when it is database-related token mentioned in $x^{(t)}$ and history h. To encourage the co-reference phenomenon in the simplified utterance, we award the pronoun words with $+0.1$ reward. Otherwise, the generated tokens will get zero points.

Sentence-level Reward We only use the pre-trained language model $\mathrm{LM}_s(\cdot)$ to evaluate the quality of the simplified utterance with $r_{\mathrm{LMs}} = \log(\mathrm{LM}_s(\hat{s}_i^{(t)}))/\mathrm{len}(\hat{s}_i^{(t)})$, where $\mathrm{len}(\hat{s}_i^{(t)})$ denotes the number of the tokens in $\hat{s}_i^{(t)}$.

For the j-th token in the simplified utterance $\hat{s}_i^{(t)}$, its accumulated reward is represented as $R_i^{(t,j)} = r_i^{(t,j)} + \sum_{l=j+1}^{m} \lambda^{l-j} r_i^{(t,l)}$, where $r_i^{(t,j)}$ means the j-th token reward of the simplified utterance and the final token reward equals to the sentence-level reward $r_i^{(t,m)} = r_{\mathrm{LMs}}$. Similar to the first loop in Section 3.2.2, the simplifier and the rewriter can be optimized with policy gradient method and MLE respectively:

$$L_{pg}(\Theta_s) = -\sum_{i,j=1}^{k,m} R_i^{(t,j)} \log\left(P(\hat{s}_i^{(t,j)} | (x^{(t)}; h); \Theta_s)\right),$$

$$L_{ce}(\Theta_c) = -\log\left(P(x^{(t)} | (\hat{s}_i^{(t)}; h); \Theta_c)\right).$$

Noting that only when $\mathrm{len}(x^{(t)}) > \mathrm{len}(\hat{s}_i^{(t)})$, we optimize the rewriter with MLE.

4 Experiments

Series of experiments are conducted to validate our proposed utterance rewrite model and the decoupled framework. We first validate the pretrained BART's performance on the utterance rewrite benchmarks. Then, the multi-turn Text-to-SQL with the decoupled parsing method (DELTA) are experimented on the limited utterance rewrite data. Finally, we analyze the interpretability of the decoupled parsing method through the case study.

4.1 Experimental Setup

Datasets&Metrics Our proposed rewrite model is validated on two utterance rewrite datasets: TASK (Quan et al., 2019) and CANARD (Elgohary et al., 2019a). We employ the widely used automatic metrics BLEU, ROUGE, EM (Exact Match) and rewrite F-score as our evaluation metrics. **BLEU**$_n$(**B**$_n$) and **ROUGE**$_n$(**R**$_n$) are used to calculate the similarity and the overlapping at the n-grams level between predictions and golden

TASK			
Models	**EM**	$\mathbf{B}_4$	$\mathcal{F}_1$
GECOR 1[†]	68.5	83.9	66.1
GECOR 2[†]	66.2	83.0	66.2
RUN[‡]	69.2	85.6	70.6
BART	**74.2**	**89.4**	**81.2**

CANARD					
Models	$\mathbf{B}_1$	$\mathbf{B}_2$	$\mathbf{B}_4$	$\mathbf{R}_2$	$\mathbf{R}_\mathbf{L}$
Pronoun Sub[‡]	60.4	55.3	47.4	63.7	73.9
L-Ptr-Gen[‡]	67.2	60.3	50.2	62.9	74.9
RUN[‡]	70.5	61.2	49.1	61.2	74.7
BART	**84.5**	**71.3**	**54.3**	**71.1**	**81.7**

Table 1: The experimental results on rewrite datasets TASK (**left**) and CANARD (**right**). †: results from (Quan et al., 2019); ‡: results from (Liu et al., 2020b).

Models	SParC				CoSQL			
	Question Match		Interaction Match		Question Match		Interaction Match	
	Dev.	Test	Dev.	Test	Dev.	Test	Dev.	Test
EditSQL (Zhang et al., 2019)	47.2	47.9	29.5	25.3	39.9	40.8	12.3	13.7
RichContext (Liu et al., 2020a)	52.6	-	29.9	-	41.0	-	14.0	-
IGSQL (Cai and Wan, 2020)	50.7	51.2	32.5	29.5	44.1	42.5	15.8	15.0
R^2SQL (Hui et al., 2021)	54.1	55.8	35.2	30.8	45.7	46.8	19.5	17.0
DELTA+Dual(ours)	**58.6**	**59.9**	**35.6**	**31.8**	**51.7**	**50.8**	**21.5**	**19.7**

Table 2: The question match accuracy and interaction match accuracy on SParC and CoSQL datasets. Since the test datasets are not public, RichContext has not evaluated by the dataset owner.

ones. **EM** means the exact match rate, where the prediction exactly equals to the golden. Rewrite F-score $\mathcal{F}_n$ is calculated on the collection of n-grams that contain at least one word from the context. Our decoupled parsing method is evaluated on two multi-turn Text-to-SQL tasks: SParC and CoSQL. Following (Yu et al., 2019b), with **Question Match** and **Interaction Match** as the metrics. Question match means the predicted SQL equals the golden one for each question, while Interaction match indicates the predicted SQL queries of all the questions in an interaction are correct.

Implementation Details Our implementation is based on PyTorch (Paszke et al., 2019) and HuggingFace's (Wolf et al., 2020) Transformers library. We reproduce RATSQL with the same setup presented in (Wang et al., 2020), where the encoder consists of eight relation-aware Transformer (RAT) layers. When fine-tuning the BARTs (rewriter and simplifier) on the utterance rewrite datasets, we use the AdamW as the optimizer with the learning rate 2e-6. At the dual learning and co-training period, we set the learning rate as 1e-6. Specifically, BARTs mentioned above refer to $\mathrm{BART}_{\mathrm{large}}$. The discount rate λ in the dual learning method is 1.

4.2 Experimental Results

4.2.1 BART as Rewrite Model

For the rewrite task, we compared the pretrained BART with state-of-the-art rewrite models: L-Ptr-Gen (See et al., 2017), GECOR (Quan et al., 2019), and RUN (Liu et al., 2020b). Table 1 shows the experimental results on TASK and CANARD datasets. As indicated, using the BART as rewrite model surpasses the best baseline RUN by a large margin on all the metrics. Even for the most challenging metric **EM**, the BART exceeds the previous best model by 5.0 points on TASK. The BART also obtains a large boost on CANARD, which improves the state-of-the-art by 4.1 points and 6.8 points on B_4 and R_L respectively. The above experimental results demonstrate the superiority of the BART as the rewrite model.

4.2.2 DELTA for Decoupled Parsing

Regarding the multi-turn Text-to-SQL task, we compared the decoupled parsing method with all the released end-to-end multi-turn Text-to-SQL models: EditSQL (Zhang et al., 2019), RichContext (Liu et al., 2020a), IGSQL (Cai and Wan, 2020), and R^2SQL (Hui et al., 2021).

Since there is no utterance rewrite data on SParC and CoSQL datasets, we randomly sample 10% dialogues on these two datasets and annotate them as the rewrite in-domain data. There are 741 annotated turns and 695 annotated turns on SParC and CoSQL respectively. At the Phase-I, we first use the rewrite in-domain data to warm-up the rewrite model and simplification model, where their encoders share the parameters inspired by (Lample et al., 2018). Then, we use the rest of 90% di-

Precedent Question	Current Question	Rewritten Question	Predicted SQL	Status
how many people live in asia ?	what about the largest gnp among them ?	~~how many people live~~ what is the country in asia that is the largest gnp ~~among them~~ ?	SELECT country.Population FROM country WHERE country.Continent= 'value' ORDER BY country.GNP ESC LIMIT 1	Fail in Phase-I
which students have pets ?	what are the different first names ?	what are the different first names of the students that have pets ?	SELECT Student.Fname FROM Student JOIN Has_pet ON Student.stuid = Has_pet.stuid	Fail in Phase-II
what flights land in aberdeen ?	also include flights that land in abilene .	what flights land in aberdeen or abilene ?	SELECT * FROM flights JOIN airports WHERE airports.City = 'value' OR airports.City = 'value'	Success

Table 3: Three instances parsed by our proposed decoupled parsing method with rewritten utterance and final predicted SQL query. The ~~red~~ means that the error happens. The green is modified by us.

Variants	SParC	
	QM	IM
(0) DELTA + Dual	58.6	35.6
(1) DELTA + Co-training	57.2	33.6
(2) - Dual	55.5	31.7
(3) - parsing in-domain data	54.7	31.5
(4) - rewrite in-domain data	42.1	14.4
(5) - rewriter	34.5	7.1

Table 4: The ablations of our proposed decoupled parsing framework. Since the test dataset of the SParC is not released, we report all the performances on its development. QM: question match accuracy; IM: interaction match accuracy.

alogues as unlabeled data to further improve the rewrite model with the dual learning method, detailed in Section 3. At the Phase-II, we first use the single-turn Text-to-SQL data (Spider) to warm-up the RATSQL parser. Since there is an annotation gap[1] between Spider and multi-turn Text-to-SQL datasets, we use the annotated Text-to-SQL data on the SParC and CoSQL to fine-tune the pretrained RATSQL parser, where the multi-turn dialogue data are rewritten as single-turn data by the rewrite model trained in Phase-I. Table 2 shows that our proposed decoupled framework (DELTA+Dual) gets a considerable performance boost on SParC and CoSQL datasets.

4.2.3 Ablation Study

We conducted an ablation study to analyze the contribution of our proposed decoupled parsing framework on the SParC dataset. To compare with our proposed dual learning method on rewrite task, we examined another semi-supervised learning method co-training (Blum and Mitchell, 1998), which uses the pretrained rewrite model to annotate the unlabeled data and add these pseudo-labeled data to improve the original rewrite model iteratively. To fairly compare with the dual learning method, we only use the pseudo labeled rewrite data that are correctly predicted by the RATSQL parser at each iteration of the co-training method. As shown in Table 4, our proposed dual learning method outperforms the co-training method at row (1). To further validate the effect of the dual learning method, we remove the dual learning part in the Phase-I. Compared with our adapted dual learning method at row (2), the above two variants have a significant performance degradation, which demonstrates the superiority of the dual learning method on rewrite task.

Compared with the end-to-end multi-turn Text-to-SQL models, our proposed decoupled parsing framework even does not require any annotated multi-turn in-domain data. We first evaluate the performance of the decoupled method without any Text-to-SQL parsing in-domain data at row (3). There are 3.9 points and 4.1 points degradation on question match accuracy and interaction match accuracy respectively, which is caused by the annotation gap between Spider and SParC. We further drop our annotated rewrite in-domain data at row (4) and warm-up the rewrite model and simplification model with TASK and CANARD datasets. As shown in Table 4, we can find the decoupled parsing framework still gets 42.1% question match accuracy without any annotated multi-turn in-domain data. Lastly, we just remove the Phase-I (rewrite

---

[1] For example, "Tell me how many rooms cost more than 120, for each different decor." is annotated as "SELECT decor, count(*) FROM Rooms WHERE basePrice > 120 GROUP BY decor" in SParC. It is tend to be annotated as "SELECT count(*) FROM Rooms WHERE basePrice > 120 GROUP BY decor" in Spider.

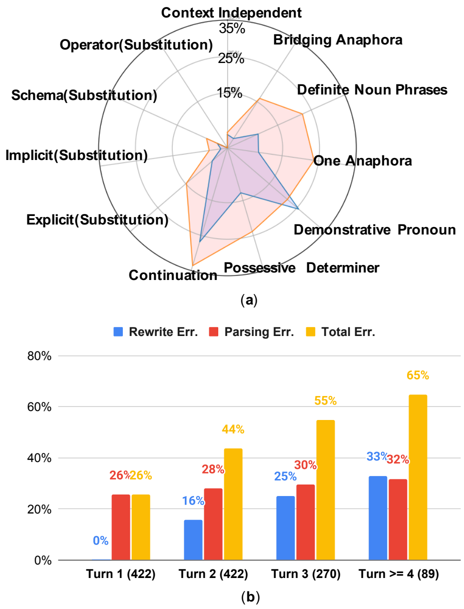

Figure 3: The rewrite error analysis (**a**) in the fine-grained level and the error rate of the two decoupled models (**b**) at turn-wise level.

model) at row (5), where the RATSQL parser is trained on Spider and fine-tuned with multi-turn Text-to-SQL data. It can be regarded as the baseline of all the ablations, which only gets 34.5% question match accuracy.

4.3 Case Analysis

Compared with end-to-end multi-turn Text-to-SQL models, our decoupled parser can generate the intermediate rewritten utterance, which is easier to understand for the user than a SQL query. As introduced in Section 3.2.2, the feedback of the user can be used to optimize the rewrite model. Additionally, our decoupled parser is more convenient in data collection compared with end-to-end methods, which does not require annotators' familiarization with SQL to rewrite an utterance. When collecting single-turn Text-to-SQL data, the annotator does not need to consider the dialogue context. It is also costly to collect the dialogue data on the SQL query task.

Table 3 displays three cases parsed by our proposed decoupled method. We can pinpoint exactly which phase the error occurred under decoupled parsing framework. Through fine-grained error analysis, the bottleneck of multi-turn parser can be found accurately. Thus, we can target to optimize the bottleneck individually. Figure 3(**a**) shows the error rate of the utterance rewrite model (DELTA+Dual) on SParC development dataset at a fine-grained level. The orange line denotes the error rate on the individual co-reference or ellipsis type. The blue line denotes the overall error ratio. We can see that most rewrite errors happen on the co-reference side, especially at *Demonstrative Pronoun* type. For the ellipsis, *Continuation* type is a serious problem. Figure 3(**b**) shows the error ratios that happen in the rewrite model (Phase-I) or in the parsing model (Phase-II). We can find that at the first three turns the parsing model is still the bottleneck. After the third turn, the rewrite model gets a bigger error rate. The error rate of the rewrite model is more sensitive than the parser with turn increased. We can conclude that we need more annotated rewrite data, especially with *Continuation* type and *Demonstrative Pronoun* type.

5 Related Work

Utterance Rewrite Recently, the utterance rewrite has raised large attention. Some works use the sequence-to-sequence architecture with copy mechanism (Elgohary et al., 2019b; Quan et al., 2019; Rastogi et al., 2019) to solve the incomplete question problem. Liu et al. (2019) decompose the utterance rewrite model as two-phase subtasks: split and recombine. The split and recombine models are both learned from the well-designed reward function by the policy gradient method. Borrowing the idea from image segmentation, Liu et al. (2020b) formulate the utterance rewrite as the semantic segmentation task, where the rewrite model is implemented with UNet (Ronneberger et al., 2015). For the downstream task, the utterance rewrite has been successfully used in dialogue state tracking (DST) tasks (Rastogi et al., 2019; Han et al., 2020). Yu et al. (2020a) propose a rule-based and self-supervised learning method to generate weakly-supervised rewrite data, which are used to fine-tune GPT-2. Different from the previous works, we directly use the pretrained BART, which is a denoising autoencoder, as the utterance rewrite model.

End-to-End Text-to-SQL Parser EditSQL (Zhang et al., 2019) proposes an Edit-based model that reuses the SQL query generated from the previous step to alleviate the pressure of the increasing turns. RichContext (Liu et al., 2020a) conducts an exploratory study on semantic parsing in context and performs a fine-grained analysis. IGSQL (Cai and Wan, 2020) presents a schema interaction graph encoder to capture the historical information of database schema items. R^2SQL (Hui et al., 2021) presents a dynamic graph framework that employs dynamic memory decay mechanisms to introduce inductive bias to construct enriched contextual relation representation at both utterance and token level.

Dual Learning Dual learning method is first proposed to improve neural machine translation (NMT) (He et al., 2016). The dual learning mechanism enables a pair of dual systems to automatically learn from unlabeled data through a closed-loop game. The idea of dual learning has been applied into various tasks, such as Question Answer (Tang et al., 2017)/Generation (Tang et al., 2018), Image-to-Image Translation (Yi et al., 2017), Open-domain Information Extraction/Narration (Sun et al., 2018), Text Simplification (Zhao et al., 2020), Semantic Parsing (Cao et al., 2019; Zhu et al., 2020a; Cao et al., 2020) and dialogue state tracking (Chen et al., 2020c).

6 Conclusion and Future Work

In this paper, we propose a decoupled parsing framework (DELTA+Dual) to solve the multi-turn Text-to-SQL task. The previous end-to-end multi-turn Text-to-SQL models rely on large-scale multi-turn data. DELTA can achieve considerable performance without any multi-turn Text-to-SQL data. We adapt the pretrained BART as the rewrite model and achieve new state-of-the-art performance on the utterance rewrite benchmarks. We further propose an efficient dual learning method to make full use of unlabeled dialogue data. On the challenging multi-turn Text-to-SQL benchmarks, DELTA surpasses all the released end-to-end models with fully labeled data. In the future, we will try to reformulate the decoupled parsing method as a multitask, where the rewrite model and Text-to-SQL model are trained simultaneously. The proposed DELTA is also easy to extend to the other conversational semantic parsing tasks, like dialogue state tracking (Chen et al., 2020b; Zhu et al., 2020b; Chen et al., 2020a).

Acknowledgements

We thank Bo Pang, Tao Yu and Songhe Wang for their help with the evaluation. We would like to thank the anonymous reviewers for their thoughtful comments. This work has been supported by No. SKLMCPTS2020003 Project, Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and Startup Fund for Youngman Research at SJTU (SFYR at SJTU).

References

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases-an introduction. *arXiv preprint cmp-lg/9503016*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Yitao Cai and Xiaojun Wan. 2020. IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6903–6912, Online. Association for Computational Linguistics.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 51–64.

Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. 2020. Unsupervised dual paraphrasing for two-stage semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6806–6817.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020a. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7521–7528.

Zhi Chen, Lu Chen, Zihan Xu, Yanbin Zhao, Su Zhu, and Kai Yu. 2020b. Credit: Coarse-to-fine sequence generation for dialogue state tracking. *arXiv preprint arXiv:2009.10435*.

Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021. ShadowGNN: Graph projection neural network for text-to-SQL parser. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5567–5577, Online. Association for Computational Linguistics.

Zhi Chen, Lu Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2020c. Dual learning for dialogue state tracking. *arXiv preprint arXiv:2009.10430*.

Deborah A Dahl, Madeleine Bates, Michael K Brown, William M Fisher, Kate Hunicke-Smith, David S Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019a. Can you unpack that? learning to rewrite questions-in-context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5920–5926.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019b. Can you unpack that? learning to rewrite questions-in-context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5918–5924, Hong Kong, China. Association for Computational Linguistics.

Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Wei Peng, and Minlie Huang. 2020. Multiwoz-coref: A multi-domain task-oriented dataset enhanced with annotation corrections and co-reference annotation. *arXiv preprint arXiv:2010.05594*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in neural information processing systems*, pages 820–828.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic hybrid relation network for cross-domain context-dependent semantic parsing. *arXiv preprint arXiv:2101.01686*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Qian Liu, Bei Chen, Jiaqi Guo, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020a. How far are we from effective context modeling? an exploratory study on semantic parsing in context. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3580–3586. International Joint Conferences on Artificial Intelligence Organization.

Qian Liu, Bei Chen, Haoyan Liu, Jian-Guang Lou, Lei Fang, Bin Zhou, and Dongmei Zhang. 2019. A split-and-recombine approach for follow-up query analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5316–5326, Hong Kong, China. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020b. Incomplete utterance rewriting as semantic segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2846–2857.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. 2019. Gecor: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4539–4549.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. 2019. Scaling multi-domain dialogue state tracking via query reformulation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,

pages 311–324, Online. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. 2020. Learning contextual representations for semantic parsing with generation-augmented pre-training. *arXiv preprint arXiv:2012.10309*.

Mingming Sun, Xu Li, and Ping Li. 2018. Logician and orator: Learning from the duality between language and knowledge in open domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2119–2130.

Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. 2018. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1564–1574.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857.

Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020a. Few-shot generative conversational query rewriting. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1933–1936.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020b. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Irene Li Heyang Er, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Vincent Zhang Jonathan Kraft, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. Sparc: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9668–9675.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Su Zhu, Ruisheng Cao, and Kai Yu. 2020a. Dual learning for semi-supervised natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1936–1947.

Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020b. Efficient context and schema fusion networks for multi-domain dialogue state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 766–781.

Adjacency List Oriented Relational Fact Extraction via Adaptive Multi-task Learning

Fubang Zhao[1][*], **Zhuoren Jiang**[2][*][†], **Yangyang Kang**[1], **Changlong Sun**[1], **Xiaozhong Liu**[3]
[1]Alibaba Group, Hangzhou, China
[2]School of Public Affairs, Zhejiang University, Hangzhou, China
[3]School of Informatics, Computing and Engineering, IUB, Bloomington, USA
`fubang.zfb@alibaba-inc.com, jiangzhuoren@zju.edu.cn`
`yangyang.kangyy@alibaba-inc.com, changlong.scl@taobao.com`
`liu237@indiana.edu`

Abstract

Relational fact extraction aims to extract semantic triplets from unstructured text. In this work, we show that all of the relational fact extraction models can be organized according to a graph-oriented analytical perspective. An efficient model, a**D**jacency l**I**st o**R**iented r**E**lational fa**CT** (**DIRECT**), is proposed based on this analytical framework. To alleviate challenges of error propagation and sub-task loss equilibrium, DIRECT employs a novel adaptive multi-task learning strategy with dynamic sub-task loss balancing. Extensive experiments are conducted on two benchmark datasets, and results prove that the proposed model outperforms a series of state-of-the-art (SoTA) models for relational triplet extraction.

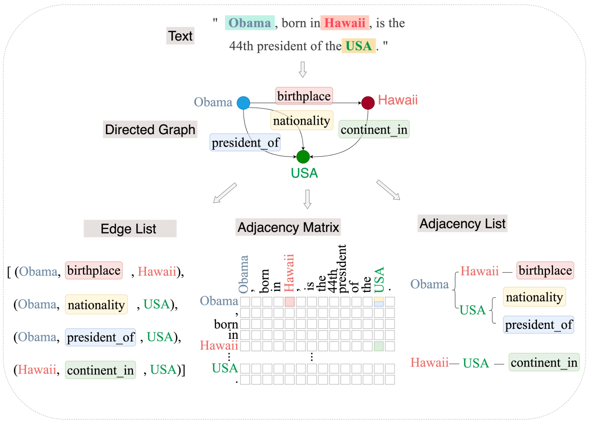

Figure 1: Example of exploring the relational fact extraction task from the perspective of directed graph representation method as output data structure.

1 Introduction

Relational fact extraction, as an essential NLP task, is playing an increasingly important role in knowledge graph construction (Han et al., 2019; Distiawan et al., 2019). It aims to extract relational triplet from the text. A relational triplet is in the form of $(subject, relation, object)$ or (s, r, o) (Zeng et al., 2019). While various prior models proposed for relational fact extraction, few of them analyze this task from the perspective of output data structure.

As shown in Figure 1, the relational fact extraction can be characterized as a directed graph construction task, where graph representation flexibility and heterogeneity accompany additional benefaction. In practice, there are three common ways to represent graphs (Gross and Yellen, 2005):

Edge List is utilized to predict a sequence of triplets (edges). The recent sequence-to-sequence based models, such as NovelTagging (Zheng et al., 2017), CopyRE (Zeng et al., 2018), CopyRL (Zeng et al., 2019), and PNDec (Nayak and Ng, 2020), fall into this category.

Edge list is a simple and space-efficient way to represent a graph (Arifuzzaman and Khan, 2015). However, there are three problems. First, the triplet overlapping problem (Zeng et al., 2018). For instance, as shown in Figure 1, for triplets (Obama, nationality, USA) and (Obama, president_of, USA), there are two types of relations between the "Obama" and "USA". If the model only generates one sequence from the text (Zheng et al., 2017), it may fail to identify the multi-relation between entities. Second, to overcome the triplet overlapping problem, the model may have to extract the triplet element repeatedly (Zeng et al., 2018), which will increase the extraction cost. Third, there could be an ordering problem (Zeng et al., 2019): for multiple triplets, the extraction order could influence the model performance.

Adjacency Matrices are used to predict matrices that represent exactly which entities (vertices) have semantic relations (edges) between them. Most early works, which take a pipeline ap-

---

[*] These two authors contributed equally to this research.
[†] Zhuoren Jiang is the corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3075–3087
August 1–6, 2021. ©2021 Association for Computational Linguistics

proach (Zelenko et al., 2003; Zhou et al., 2005), belong to this category. These models first recognize all entities in text and then perform relation classification for each entity pair. The subsequent neural network-based models (Bekoulis et al., 2018; Dai et al., 2019), that attempt to extract entities and relations jointly, can also be classified into this category.

Compared to edge list, adjacency matrices have better relation (edge) searching efficiency (Arifuzzaman and Khan, 2015). Furthermore, adjacency matrices oriented models is able to cover different overlapping cases (Zeng et al., 2018) for relational fact extraction task. But the space cost of this approach can be expensive. For most cases, the output matrices are very sparse. For instance, for a sentence with n tokens, if there are m kinds of relations, the output space is $n \cdot n \cdot m$, which can be costly for graph representation efficiency. This phenomenon is also illustrated in Figure 1.

Adjacency List is designed to predict an array of linked lists that serves as a representation of a graph. As depicted in Figure 1, in the adjacency list, each vertex v (key) points to a list (value) containing all other vertices connected to v by several edges. Adjacency list is a hybrid graph representation between edge list and adjacency matrices (Gross and Yellen, 2005), which can balance space and searching efficiency[1]. Due to the structural characteristic of the adjacency list, this type of model usually adopts a cascade fashion to identify subject, object, and relation sequentially. For instance, the recent state-of-the-art model CasRel (Wei et al., 2020) can be considered as an exemplar. It utilizes a two-step framework to recognize the possible object(s) of a given subject under a specific relation. However, CasRel is not fully adjacency list oriented: in the first step, it use $subject$ as the key; while in the second step, it predicts $(relation, object)$ pairs using adjacency matrix representation.

Despite its considerable potential, the cascade fashion of adjacency list oriented model may cause problems of sub-task error propagation (Shen et al., 2019), i.e., errors from ancestor sub-tasks may accumulate to threaten downstream ones, and subtasks can hardly share supervision signals. Multitask learning (Caruana, 1997) can alleviate this problem, however, the sub-task loss balancing problem (Chen et al., 2018; Sener and Koltun, 2018) could compromise its performance.

Based on the analysis from the perspective of output data structure, we propose a novel solution, a**D**jacency l**I**st o**R**iented r**E**lational fa**CT** extraction model (**DIRECT**), with the following advantages:

- For efficiency, DIRECT is a fully adjacency list oriented model, which consists of a shared BERT encoder, the Pointer-Network based subject and object extractors, and a relation classification module. In Section 3.4, we provide a detailed comparative analysis[2] to demonstrate the efficiency of the proposed method.

- From the performance viewpoint, to address sub-task error propagation and sub-task loss balancing problems, DIRECT employs a novel adaptive multi-task learning strategy with the dynamic subtask loss balancing approach. In Section 3.2 and 3.3, the empirical experimental results demonstrate DIRECT can achieve the state-of-the-art performance of relational fact extraction task, and the adaptive multi-task learning strategy did play a positive role in improving the task performance.

The major contributions of this paper can be summarized as follows:

1. We refurbish the relational fact extraction problem by leveraging an analytical framework of graph-oriented output structure. To the best of our knowledge, this is a pioneer investigation to explore the output data structure of relational fact extractions.

2. We propose a novel solution, DIRECT[3], which is a fully adjacency list oriented model with a novel adaptive multi-task learning strategy.

3. Through extensive experiments on two benchmark datasets[3], we demonstrate the efficiency and efficacy of DIRECT. The proposed DIRECT outperforms the state-of-the-art baseline models.

2 The DIRECT Framework

In this section, we will introduce the framework of the proposed DIRECT model, which includes a shared BERT encoder and three output layers: subject extraction, object extraction, and relation classification. As shown in Figure 2, DIRECT is fully adjacency list oriented. The input sentence is firstly fed into the subject extraction module to

---

[1] More detailed complexity analyses of different graph representations are provided in Appendix section 6.3.

[2] Theoretical representation efficiency analysis of graph representative models are described in Appendix section 6.4.

[3] To help other scholars reproduce the experiment outcome, we will release the code and datasets via GitHub: https://github.com/fyubang/direct-ie.

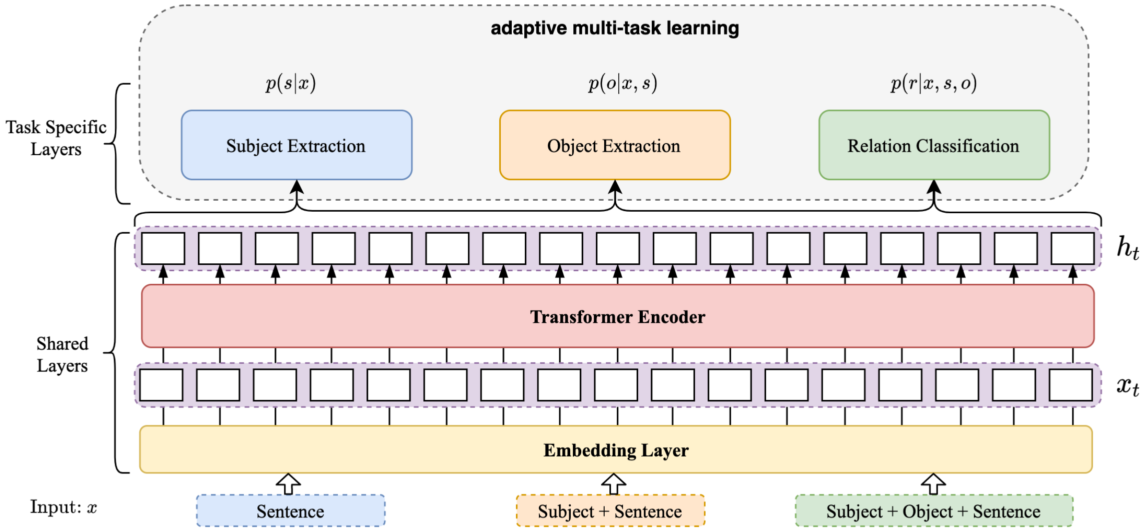

Figure 2: An overview of the proposed DIRECT framework

extract all subjects. Then each extracted subject is concatenated with the sentence, and fed into the object extraction module to extract all objects, which can form a set of subject-object pairs. Finally, the subject-object pair is concatenated with sentence, and fed into the relation classification module to get the relations between them. For balancing the weights of sub-task losses and to improve the global task performance, three modules share the BERT encoder layer and are trained with an adaptive multi-task learning strategy.

2.1 Shared BERT Encoder

In the DIRECT framework, the encoder is used to extract the semantic features from the inputs for three modules. As aforementioned, we employ the BERT (Devlin et al., 2019) as the shared encoder to make use of its pre-trained knowledge and attention mechanism.

The architecture of the shared method is shown in Figure 2. The lower embedding layer and transformers (Vaswani et al., 2017) are shared across all the three modules, while the top layers represent the task-specific outputs.

The encoding process is as follows:

$$\mathbf{h}^t = \text{BERT}(\mathbf{x}^t) \tag{1}$$

where $\mathbf{x}^t = [w_1, ..., w_n]$ is the input text of task t and $\mathbf{h}^t$ is the hidden vector sequence of the input. Due to the limited space, the detailed architecture of BERT please refer to the original paper (Devlin et al., 2019).

2.2 Subject and Object Extraction

The subject and object extraction modules are motivated by the Pointer-Network (Vinyals et al., 2015) architecture, which are widely used in Machine Reading Comprehension (MRC) (Rajpurkar et al., 2016) task. Different from MRC task that only needs to extract a single span, the subject and object extractions need to extract multiple spans. Therefore, in the training phase, we replace $softmax$ function with $sigmoid$ function for the activation function of the output layer, and replace cross entropy (CE) (Goodfellow et al., 2016) with binary cross entropy (BCE) (Luc et al., 2016) for the loss function. Specifically, we will perform independent binary classifications for each token twice to indicate whether the current token is the start or the end of a span. The probability of a token to be start or end is as follows:

$$p^t_{i,\text{start}} = \sigma(\mathbf{W}^t_{\text{start}} \cdot h_i + \mathbf{b}^t_{\text{start}}) \tag{2}$$

$$p^t_{i,\text{end}} = \sigma(\mathbf{W}^t_{\text{end}} \cdot h_i + \mathbf{b}^t_{\text{end}}) \tag{3}$$

where h_i represents the hidden vector of the i_{th} token, $t \in [s, o]$ represents subject and object extraction respectively, $\mathbf{W}^t \in \mathbb{R}^{h\times 1}$ represents the trainable weight, $\mathbf{b}^t \in \mathbb{R}^1$ is the bias and σ is $sigmoid$ function.

During inference, we first recognize all the start positions by checking if the probability $p^t_{i,\text{start}} > \alpha$, where α is the threshold of extraction. Then, we identify the corresponding end position with the largest probability $p^t_{i,\text{end}}$ between two neighboring start positions. Concretely, assuming $\text{pos}_{j,\text{start}}$ is the start position of the j_{th} span, the corresponding end position is:

$$\text{pos}_{j,\text{end}} = \underset{\text{pos}_{j,\text{start}} <= i < \text{pos}_{j+1,\text{start}}}{\text{argmax}} \; p^t_{i,\text{end}} \tag{4}$$

Though the overall structure is similar, the inputs for subject and object extraction are different. When extracting the subject, only the original sentence needs to be input:

$$\mathbf{x} = [w_1, ..., w_n] \tag{5}$$

$$\text{input}^s = [[\text{cls}], \mathbf{x}, [\text{sep}]] \tag{6}$$

where w_i represents the i_{th} token of the original sentence.

Meanwhile, the object extraction is based on the corresponding subject. To form the input, the subject s and the original sentence x are concatenated with $[sep]$ as follows:

$$\text{input}^o = [[\text{cls}], s, [\text{sep}], \mathbf{x}, [\text{sep}]] \tag{7}$$

2.3 Relation classification

The output layer of relation classification is relatively simple, which is a normal multi-label classification model. The $[cls]$ vector obtained by BERT encoder is used as the sentence embedding. A fully connected layer is used for the nonlinear transformation, and perform multi-label classification to predict relations of the input subject-object pair. The detailed operations of relation classification are as follows:

$$\mathbf{P}^r = \sigma(\mathbf{W}^r \cdot h_{[\text{cls}]} + \mathbf{b}^r) \tag{8}$$

where $\mathbf{P}^r \in \mathbb{R}^c$ is the predicted probability vector of relations, σ is $sigmoid$ function, $\mathbf{W}^r \in \mathbb{R}^{h \times c}$ and $\mathbf{b}^r \in \mathbb{R}^c$ are the trainable weights and bias, h is the hidden size of encoder, c is the number of relations, and $h_{[\text{cls}]}$ denotes the hidden vector of the first token [cls]. The input for relation classification task is as follows:

$$\text{input}^r = [[\text{cls}], s, [\text{sep}], o, [\text{sep}], \mathbf{x}, [\text{sep}]] \tag{9}$$

2.4 Adaptive Multi-task Learning

In DIRECT, subject extraction module, object extraction module, and relation classification module can be considered as three sub-tasks. As aforementioned, if we train each module directly and separately, the error propagation problem would reduce the task performance. Meanwhile, three independent encoders would consume more memory. Therefore, we use multi-task learning to alleviate this problem, and the encoder layer is shared across three modules.

Algorithm 1: Adaptive Multi-task Learning with Dynamic Loss Balancing

```
Initialize model parameters $\Theta$ randomly;
Load pre-trained BERT parameters for
  shared encoder;
Prepare the data for each task $t$ and pack
  them into mini-batch: $D^t$, $t \in [s, o, r]$ ;
Get the number of batch for each task: $n^t$;
Set the number of epoch for training:
  $epoch_{max}$;
for $epoch$ in $1, 2, ..., epoch_{max}$ do
    1. Merge all the datasets:
      $D = D^s \cup D^o \cup D^r$;
    2. Shuffle D;
    3. Initialize EMA for each task $v^t = 1$
      and its decay $\epsilon = 0.99$ ;
    for $b^t$ in $D$ do
        // $b^t$ is a mini-batch of $D^t$ ;
        4. Compute loss: $l^t(\Theta)$ ;
        5. Update EMA:
          $v^t = (1 - \epsilon) \cdot \sum(l^t) + \epsilon \cdot v^t$ ;
        6. Calculate and normalize the
          weights: $w^t = (v^t/n^t)/(v^r/n^r)$ ;
        7. Update model $\Theta$ with gradient:
          $\nabla(w^t \cdot \bar{l}^t)$ ;
    end
end
```

However, applying multi-task learning could be challenging in DIRECT, due to the following problems:

- The input and output of the three modules are different, which means we cannot simply sum up the loss of each task.

- How should we balance the weights of losses for three sub-task modules?

These issues can affect the final results of multi-task training (Shen et al., 2019; Sener and Koltun, 2018).

In this work, based on the architecture of MT-DNN (Liu et al., 2019b), we propose a novel adaptive multi-task learning strategy to address the above problems. The algorithm is shown as Algorithm 1. Basically, the datasets are firstly split into mini-batches. A batch is then randomly sampled to calculate the loss. The parameters of the shared encoder and its task-specific layer are updated accordingly. Especially, the learning effect of each task t is different and dynamically changing during training. Therefore, an approach of adaptively adjusting the weights of task losses is applied. The sum of sub-task's loss $\sum l^t$ is utilized to approximate its optimization effect. The adaptive weight adjusting strategy ensures that the more room a sub-task has to be optimized, the more weight its loss will receive. Furthermore, an exponential moving average (EMA) (Lawrance and Lewis, 1977) is maintained to avoid the drastic fluctuations of loss weights. Last but not least, to make sure that each task has enough influence on the shared encoder, the weight of the sub-task will be penalized according to the training data amount of each sub-task.

3 Experiments

3.1 Dataset and Experiment Setting

Datasets. Two public datasets are used for evaluation: **NYT** (Riedel et al., 2010) is originally produced by the distant supervision approach. There are 1.18M sentences with 24 predefined relation types in NYT. **WebNLG** (Gardent et al., 2017) is originally created for Natural Language Generation (NLG) tasks. (Zeng et al., 2018) adopts this dataset for relational triplet extraction task. It contains 246 predefined relation types. There are different versions of these two datasets. To facilitate comparison evaluation, we use the datasets released by (Zeng et al., 2018) and follow their data split rules.

Besides the basic relational triplet extraction, recent studies are focusing on the relational triplet overlapping problem (Zeng et al., 2018; Wei et al., 2020). Follow the overlapping pattern definition of relational triplets (Zeng et al., 2018), the sentences in both datasets are divided into three categories, namely, Normal, EntityPairOverlap (EPO), and SingleEntityOverlap (SEO). The statistics of the two datasets are described in Table 1.

Category	NYT		WebNLG	
	Train	Test	Train	Test
Normal	37013	3266	1596	246
EPO	9782	978	227	26
SEO	14735	1297	3406	457
ALL	56195	5000	5019	703

Table 1: Statistics of Dataset NYT and WebNLG. Note that a sentence can belong to both EPO class and SEO class.

Baselines: the following strong state-of-the-art (SoTA) models have been compared in the experiments.

- NovelTagging (Zheng et al., 2017) introduces a tagging scheme that transforms the joint entity and relation extraction task into a sequence labeling problem. It can be considered as edge list oriented.
- CopyRE (Zeng et al., 2018) is a seq2seq based model with the copy mechanism, which can effectively extract overlapping triplets. It has two variants: CopyRE$_{\text{one}}$ employs one decoder; CopyRE$_{\text{mul}}$ employs multiple decoders. CopyRE is also edge list oriented.
- GraphRel (Fu et al., 2019) is a GCN (graph convolutional networks) (Kipf and Welling, 2017) based model, where a relation-weighted GCN is utilized to learn the interaction between entities and relations. It is a two phases model: GraphRel$_{\text{1p}}$ denotes 1st-phase extraction model; GraphRel$_{\text{2p}}$ denotes full extraction model. GraphRel is adjacency matrices oriented.
- CopyRL (Zeng et al., 2019) combines the reinforcement learning with a seq2seq model to automatically learn the extraction order of triplets. CopyRL is edge list oriented.
- CasRel (Wei et al., 2020) is a cascade binary tagging framework, where all possible subjects are identified in the first stage, and then for each identified subject, all possible relations and the corresponding objects are simultaneously identified by a relation specific tagger. This work recently achieves the SoTA results. As aforementioned, CasRel is partially adjacency list oriented.

Evaluation Metrics: following the previous work (Zeng et al., 2018; Wei et al., 2020), different models are compared by using standard micro Precision (Prec.), Recall (Rec.), and F1-score[4]. An extracted relational triplet (subject, relation, object) is regarded as correct only if the relation and the heads of both subject and object are all correct.

Implementation Details. The hyper-parameters are determined on the validation set. To avoid the evaluation bias, all reported results from our method are averaged results for 5 runs. More implementation details are described in

---

[4] In this study, the results of baseline models are all self-reported results from their original papers. Meanwhile, the experimental results of our proposed model are the average of five runs.

Appendix section 6.1.

3.2 Results and Analysis

Relational Triplet Extraction Performance. The task performances on two datasets are summarized in Table 2. Based on the experiment results, we have the following observations and discussions:

- The proposed DIRECT model outperformed all baseline models in terms of all evaluation metrics on both datasets, which proved DIRECT model can effectively address the relational triplet extraction task.
- The best-performed model (DIRECT) and runner-up model (CasRel) were both adjacency list oriented model. These two models overwhelmingly outperformed other models, which indicated the considerable potential of adjacency list (as the output data structure) for improving the task performance.
- To further compare the relation extraction ability of DIRECT and CasRel, we took a closer look at the extraction performance of relational triplet elements from these two models. As shown in Table 3[5], DIRECT outperformed CasRel in terms of all relational triplet elements on both datasets. These empirical results suggested that, for relational triplet extraction, a fully adjacency list oriented model (DIRECT) may have advantages over a partially oriented one (CasRel).

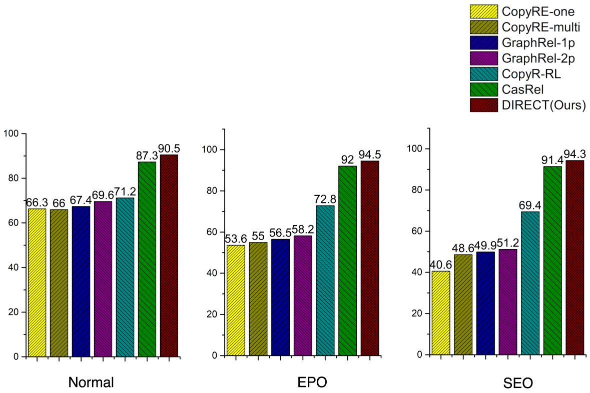

Figure 3: F1 score of extracting relational triples from sentences with different overlapping patterns on NYT dataset.

Ability in Handling The Overlapping Problem. The relational facts in sentences are often complicated. Different relational triplets may have overlaps in a sentence. To verify the ability of our models in handling the overlapping problem, we conducted further experiments on NYT dataset. Figure 3 illustrated of F1 scores of extracting relational triplets from sentences with different overlapping patterns. DIRECT outperformed all baseline models in terms of all overlapping patterns. These results demonstrated the effectiveness of the proposed model in solving the overlapping problem.

Ability in Handling Multiple Relation Extraction. We further compared the model's ability of extracting relations from sentences that contain multiple triplets. The sentences in NYT and WebNLG were divided into 5 categories. Each category contained sentences that had 1,2,3,4 or ≥ 5 triplets. The triplet number was denoted as N. As shown in Table 4:

- DIRECT achieved the best performance for all triplet categories on both datasets. These experimental results demonstrated our model had an excellent ability in handling multiple relation extraction.
- In both NYT and WebNLG datasets, when the sentences contained more triplets, the leading advantage of DIRECT became greater. This observation indicated that DIRECT was good at solving complex relational fact extraction.

3.3 Ablation Study

To validate the effectiveness of components in DIRECT, We implemented several model variants for ablation tests[6]. The results of the comparison on NYT dataset are shown in Table 5. In particular, we aim to address the following two research questions:

RQ1: Is it possible to improve the model performance by sharing the parameters of extraction layers?

RQ2: Did the proposed adaptive multi-task learning strategy improve the task performance?

Effects of Sharing Extraction Layer Parameters (RQ1). As described in Section 2, the structures of subject extraction and object extraction output layers are exactly the same. To answer RQ1, we merged the subject extraction and object extraction layers into one entity extraction layer by sharing the parameters of output layers of these two modules, denoted as $\mathbf{DIRECT_{shared}}$. From the results of Table 5, we can observe that, sharing the parameters of output layers of two extraction modules would reduce the performance of the model.

---

[5] More detailed results with Precision and Recall are provided in Appendix section 6.2.

[6] Due to the length limitation, we list two main ablation experiments, the rest will be provided in the Appendix section 6.2.

Method	Category	NYT			WebNLG		
		Prec.	Rec.	F1	Prec.	Rec.	F1
NovelTagging(Zheng et al., 2017)	EL	62.4	31.7	42.0	52.5	193.	28.3
CopyRE$_{\text{One}}$(Zeng et al., 2018)	EL	59.4	53.1	56.0	32.2	28.9	30.5
CopyRE$_{\text{Mul}}$(Zeng et al., 2018)	EL	61.0	56.6	58.7	37.7	36.4	37.1
GraphRel$_{\text{1p}}$(Fu et al., 2019)	AM	62.9	57.3	60.0	42.3	39.2	40.7
GraphRel$_{\text{2p}}$(Fu et al., 2019)	AM	63.9	60.0	61.9	44.7	41.1	42.9
CopyRL(Zeng et al., 2019)	EL	77.9	67.2	72.1	63.3	59.9	61.6
CasRel(Wei et al., 2020)	ALP	89.7	89.5	89.6	93.4	90.1	91.8
DIRECT(Ours)	ALF	**92.3** **(±0.32)**	**92.8** **(±0.26)**	**92.5** **(±0.09)**	**93.6** **(±0.1)**	**92.7** **(±0.24)**	**93.2** **(±0.07)**

Table 2: Results of different methods on NYT and WebNLG datasets. EL: Edge List; AM: Adjacency Matrices; ALP: Adjacency List (Partially); ALF: Adjacency List (Fully).

Method	Element	NYT	WebNLG
CasRel	s	93.5	95.7
	o	93.5	95.3
	r	94.9	94.0
DIRECT(Ours)	s	**95.4**	**97.3**
	o	**96.4**	**96.4**
	r	**97.8**	**97.4**

Table 3: F1-score for extracting elements of relational triplets on NYT and WebNLG datasets.

A possible explanation is that, although the output of these two modules is similar, the semantics of subject and object are different. Hence, directly sharing the output parameters of two modules could lead to an unsatisfactory performance.

Effects of Adaptive Multi-task Learning (RQ2). As described in Section 2, the adaptive multi-task learning strategy with the dynamic sub-task loss balancing approach is proposed for improving the task performance. To answer RQ2, we replaced the adaptive multi-task learning strategy with an ordinary learning strategy. In this strategy, the losses of three sub-tasks were computed with equal weights, denoted as **DIRECT**$_{\textbf{equal}}$. From the results of Table 5, we can observe that, by using adaptive multi-task learning, DIRECT was able to get a 1.5 percentage improvement on the F1-score. This significant improvement indicated that adaptive multi-task learning played a positive role in the balance of sub-task learning and can improve the global task performance.

3.4 Graph Representation Efficiency Analysis

Based on the amount estimation of predicted logits[7], we conduct a graph representation efficiency analysis to demonstrate the efficiency of the proposed method[8].

For each graph representation category, we choose one representative algorithms. **Edge List**: CopyRE (Zeng et al., 2018); **Adjacency Matrices**: MHS (Bekoulis et al., 2018); **Adjacency List**: CasRel (partially) (Wei et al., 2020) and the proposed DIRECT (fully).

The averaged predicted logits estimation for one sample[9] of different models on two datasets are shown in Table 6. MHS is adjacency matrices oriented, it has the most logits that need to be predicted. Since CasRel is partially adjacency list oriented, it needs to predict more logits than DIRECT. Theoretically, as an edge list oriented, the predicted logits of CopyRE should be the least. But, as described in Section 1, it needs to extract the entities repeatedly to handle the overlapping problem. Hence, its graph representation efficiency could be worse than our model. The structure of our model is simple and fully adjacency list oriented. Therefore, from the viewpoint of predicted logits estimation, DIRECT is the most representative-efficient model.

4 Related Work

Relation Fact Extraction. In this work, we show that all of the relational fact extraction models can be unified into a graph-oriented output structure analytical framework. From the perspective of graph representation, the prior models can be divided into three categories. **Edge List**, this type of model usually employs sequence-to-sequence fashion, such as NovelTagging (Zheng et al., 2017),

---

[7] Numeric output (0/1) of the last layer

[8] From the graph representation perspective, when a method requires fewer logits to represent the graph (set of triples), it will reduce the model fitting difficulty.

[9] The theoretical analysis of predicted logits for different models are described in Appendix section 6.4.

Method	NYT					WebNLG				
	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N \geq 5$	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N \geq 5$
Count	3244	1045	312	291	108	268	174	128	89	44
CopyRE$_{\text{One}}$	66.6	52.6	49.7	48.7	20.3	65.2	33.0	22.2	14.2	13.2
CopyRE$_{\text{Mul}}$	67.1	58.6	52.0	53.6	30.0	59.2	42.5	31.7	24.2	30.0
GraphRel$_{\text{1p}}$	69.1	59.5	54.4	53.9	37.5	63.8	46.3	34.7	30.8	29.4
GraphRel$_{\text{2p}}$	71.0	61.5	57.4	55.1	41.1	66.0	48.3	37.0	32.1	32.1
CopyRL	71.7	72.6	72.5	77.9	45.9	63.4	62.2	64.4	57.2	55.7
CasRel	88.2	90.3	91.9	94.2	83.7	89.3	90.8	94.2	92.4	90.9
DIRECT(Ours)	**90.4**	**93.1**	**94.3**	**95.8**	**93.1**	**90.3**	**92.8**	**94.8**	**94.0**	**92.9**

Table 4: F1-score of extracting relational triplets from sentences with different number (denoted as N) of triplets.

Method	NYT		
	Prec.	Rec.	F1
DIRECT$_{\text{shared}}$	92.1	91.6	91.9
DIRECT$_{\text{equal}}$	90.6	91.3	91.0
DIRECT	**92.3**	**92.8**	**92.5**

Table 5: Results of model variants for ablation tests.

Method	Category	NYT	WebNLG
CopyRe	EL	329	712
MHS	AM	57369	26518
CasRel	AL$_{\text{P}}$	3084	15836
DIRECT	AL$_{\text{F}}$	**238**	**542**

Table 6: Graph representation efficiency estimation based on the predicted logits amount. EL: Edge List; AM: Adjacency Matrices; AL$_{\text{P}}$: Adjacency List (Partially); AL$_{\text{F}}$: Adjacency List (Fully).

CopyRE (Zeng et al., 2018), CopyRL (Zeng et al., 2019), and PNDec (Nayak and Ng, 2020). Some models of this category may suffer from the triplet overlapping problem and expensive extraction cost. **Adjacency Matrices**, many early pipeline approaches (Zelenko et al., 2003; Zhou et al., 2005; Mintz et al., 2009) and recent neural network-based models (Bekoulis et al., 2018; Dai et al., 2019; Fu et al., 2019), can be classified into this category. The main problem for this type of model is the graph representation efficiency. **Adjacency List**, the recent state-of-the-art model CasRel (Wei et al., 2020) is a partially adjacency list oriented model. In this work, we propose DIRECT that is a fully adjacency list oriented relational fact extraction model. To the best of our knowledge, few previous works analyze this task from the output data structure perspective. GraphRel (Fu et al., 2019) employs a graph-based approach, but it is utilized from an encoding perspective, while we analyze it from the perspective of output structure. Our work is a pioneer investigation to analyze the output data structure of relational fact extraction.

Multi-task Learning. Multi-task Learning (MTL) can improve the model performance. (Caruana, 1997) summarizes the goal succinctly: *"it improves generalization by leveraging the domain-specific information contained in the training signals of related task."* It has two benefits (Vandenhende et al.): (1) multiple tasks share a single model, which can save memory. (2) Associated tasks complement and constrain each other by sharing information, which can reduce overfitting and improve global performance. There are two main types of MTL: hard parameter sharing (Baxter, 1997) and soft parameter sharing (Duong et al., 2015). Most of the multi-task learning is done by summing the loses directly, this approach is not suitable for our case. When the input and output are different, it is impossible to get two losses in one forward propagation. MT-DNN (Liu et al., 2019b) is proposed for this problem. Furthermore, MTL is difficult for training, the magnitudes of different task-losses are different, and the direct summation of losses may lead to a bias for a particular task. There are already some studies proposed to address this problem (Chen et al., 2018; Guo et al., 2018; Liu et al., 2019a). They all try to dynamically adjust the weight of the loss according to the magnitude of the loss, the difficulty of the problem, the speed of learning, etc. In this study, we adopt MT-DNN's framework, and propose an adaptive multi-task learning strategy that can dynamically adjust the loss weight based on the averaged EMA (Lawrance and Lewis, 1977) of the training data amount, task difficulty, etc.

5 Conclusion

In this paper, we introduce a new analytical perspective to organize the relational fact extraction models and propose DIRECT model for this task.

Unlike existing methods, DIRECT is fully adjacency list oriented, which employs a novel adaptive multi-task learning strategy with dynamic sub-task loss balancing. Extensive experiments on two public datasets, prove the efficiency and efficacy of the proposed methods.

Acknowledgments

We are thankful to the anonymous reviewers for their helpful comments. This work is supported by Alibaba Group through Alibaba Research Fellowship Program, the National Natural Science Foundation of China (61876003), the Key Research and Development Plan of Zhejiang Province (Grant No.2021C03140), the Fundamental Research Funds for the Central Universities, and Guangdong Basic and Applied Basic Research Foundation (2019A1515010837).

References

Shaikh Arifuzzaman and Maleq Khan. 2015. Fast parallel conversion of edge list to adjacency list for large-scale graphs. In *Proceedings of the Symposium on High Performance Computing*, pages 17–24.

Jonathan Baxter. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR.

Dai Dai, Xinyan Xiao, Yajuan Lyu, Shan Dou, Qiaoqiao She, and Haifeng Wang. 2019. Joint extraction of entities and overlapping relations using position-attentive sequence labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6300–6308.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bayu Distiawan, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pages 845–850.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. Graphrel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Jonathan L Gross and Jay Yellen. 2005. *Graph theory and its applications*. CRC press.

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. 2018. Dynamic task prioritization for multitask learning. In *European Conference on Computer Vision*, pages 282–299. Springer.

Xu Han, Tianyu Gao, Yuan Yao, Deming Ye, Zhiyuan Liu, and Maosong Sun. 2019. Opennre: An open and extensible toolkit for neural relation extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 169–174.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

AJ Lawrance and PAW Lewis. 1977. An exponential moving-average sequence and point process (ema1). *Journal of Applied Probability*, pages 98–113.

Shikun Liu, Edward Johns, and Andrew J Davison. 2019a. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1871–1880.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. 2016. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8528–8535.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 525–536.

Tao Shen, Xiubo Geng, QIN Tao, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-task learning for conversational question answering over a large-scale knowledge base. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2442–2451.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2692–2700.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Haoran Zhang, and Qianying Liu. 2020. Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9507–9514.

Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu, and Jun Zhao. 2019. Learning the extraction order of multiple relational facts in a sentence with reinforcement learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 367–377.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514.

Ranran Haoran Zhang, Qianying Liu, Aysa Xuemo Fan, Heng Ji, Daojian Zeng, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Minimize exposure bias of seq2seq models in joint entity and relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 236–246.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, pages 427–434.

6 Appendix

6.1 Implementation Details

We adopted the pre-trained BERT model [BERT-Base-Cased][10] as our encoder, where the number of Transformer layers was 12 and the hidden size was 768. The token types of input were always set to 0.

We used Adam as our optimizer and applied a triangular learning rate schedule as suggested by original BERT paper. In addition, we adopted a lazy mechanism for optimization. Different from the momentum mechanism of ordinary Adam optimizer (Kingma and Ba, 2015) that updated the output layer parameters for all tasks, this lazy-Adam mechanism wouldn't update the parameters of non-current tasks.

The dacay rate ϵ of EMA was set to 0.99 as default. The max sequence length was 128.

The other hyper-parameters were determined on the validation set. Notably, considering our special decoding strategy, we raised the threshold of extraction to 0.9 to balance the precision and the recall. The threshold of relation classification was set to 0.5 as default. The hyper-parameter setting was listed in Table 7.

Our mthod were implemented by Pytorch[11] and run on a server configured with a Tesla V100 GPU, 16 CPU, and 64G memory.

Hyper-parameter	NYT	WebNLG
Learning Rate	8e-5	1e-4
Epoch Num.	15	60
Batch Size	32	16

Table 7: Hyper-parameter setting for NYT and WebNLG datasets.

6.2 Supplementary Experimental Results

Ablation Study. To validate the effectiveness of components in DIRECT, We implemented several model variants for ablation tests respectively. For experimental fairness, we kept the other components in the same settings when modifying one module.

- **DIRECT$_{\textbf{shared}}$**, we merged the subject extraction and object extraction layers into one entity extraction layer by sharing the parameters of output layers of these two modules.
- **DIRECT$_{\textbf{equal}}$**, we replaced the adaptive multi-task learning strategy with an ordinary learning strategy. In this strategy, the losses of three sub-tasks were computed with equal weights, denoted as **DIRECT$_{\textbf{equal}}$**.
- **DIRECT$_{\textbf{threshold}}$**, we simply recognized all the start and end positions of entities by checking if the probability $p^t_{i,\text{start/end}} > \alpha$, where α was the threshold of extraction.
- **DIRECT$_{\textbf{adam}}$**, we used ordinary Adam as optimizer.

Method	NYT		
	Prec.	Rec.	F1
DIRECT$_{\text{shared}}$	92.1	91.6	91.9
DIRECT$_{\text{equal}}$	90.6	91.3	91.0
DIRECT$_{\text{threshold}}$	92.8	92.0	92.4
DIRECT$_{\text{adam}}$	92.1	**92.9**	92.5
DIRECT	**92.9**	92.1	**92.5**

Table 8: Results of model variants for ablation tests.

From the results of Table 8, we can observe that:

1. Sharing the parameters of output layers of subject and object extraction modules would reduce the performance of the model.
2. Compared to ordinary multi-task learning strategy, by using adaptive multi-task learning, DIRECT was able to get a 1.5 percentage point improvement on F1-score.
3. There would be a slight drop in performance, if we just used a simple threshold policy to recognize the start and end positions of an entity.
4. Despite the difference in precision and recall, there was no significant difference between these two optimizers (ordinary-Adam & lazy-Adam) for the task.

Results on Extracting Elements of Relational Triplets. The complete extraction performance of relational triplet elements from DIRECT and CaslRel are listed in Table 9. DIRECT outperformed CasRel in terms of all relational triplet elements on both datasets. These empirical results

---

[10] Available at: https://storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip

[11] https://pytorch.org/

Method	Element	NYT			WebNLG		
		Prec.	Rec.	F1	Prec.	Rec.	F1
CasRel	s	94.6	92.4	93.5	98.7	92.8	95.7
	o	94.1	93.0	93.5	97.7	93.0	95.3
	r	96.0	93.8	94.9	96.6	91.5	94.0
Ours	s	95.1	95.1	**95.1**	97.1	96.8	**96.9**
	o	97.2	96.3	**96.7**	96.4	96.3	**96.3**
	r	98.6	98.3	**98.5**	97.6	97.3	**97.4**

Table 9: Results on extracting elements of relational triplets

Method	NYT		
	Prec.	Rec.	F1
MHS* (Bekoulis et al., 2018)	60.7	58.6	59.6
CopyMTL$_{one}$(Zeng et al., 2020)	72.7	69.2	70.9
CopyMTL$_{mul}$(Zeng et al., 2020)	75.7	68.7	72.0
WDec (Nayak and Ng, 2020)	88.1	76.1	81.7
PNDec (Nayak and Ng, 2020)	80.6	77.3	78.9
Seq2UMTree (Zhang et al., 2020)	79.1	75.1	77.1
DIRECT(ours)	**90.2**	**90.2**	**90.2**

Table 10: Results of different methods under Exact-Match Metrics. * marks results reproduced by official implementation.

suggest that, for relational triplet extraction, a fully adjacency list oriented model (DIRECT) may have advantages over a partially oriented one (CasRel).

Results of Different Methods under Exact-Match Metrics. In experiment section, we followed the match metric from (Zeng et al., 2018), which only required to match the first token of entity span. Many previous works adopted this match metric (Fu et al., 2019; Zeng et al., 2019; Wei et al., 2020).

In fact, our model is capable of extracting the complete entities. Therefore, we collected papers that reported the results of exact-match metrics (requiring to match the complete entity span). The following strong state-of-the-art (SoTA) models have been compared:

- CopyMTL (Zeng et al., 2020) is a multi-task learning framework, where conditional random field is used to identify entities, and a seq2seq model is adopted to extract relational triplets.
- WDec (Nayak and Ng, 2020) fuses a seq2seq model with a new representation scheme, which enables the decoder to generate one word at a and can handle full entity names of different length and overlapping entities.
- PNDec (Nayak and Ng, 2020) is a modification of seq2seq model. Pointer networks are used in the decoding framework to identify the entities in the sentence using their start and end locations.
- Seq2UMTree (Zhang et al., 2020) is a modification of seq2seq model, which employs an unordered-multi-tree decoder to to minimize exposure bias.

The task performances on NYT dataset are summarized in Table 10. The proposed DIRECT model outperformed all baseline models in terms of all evaluation metrics. This experimental results further confirmed the efficacy of DIRECT for relational fact extraction task.

6.3 Complexity Analysis of Graph Representations

For a graph $G = (V, E)$, $|V|$ denotes the number of nodes/entities and $|E|$ denotes the number of edges/relations. Suppose there are m kinds of relations, $d(v)$ denotes the number of edges from node v.

- Edge List Complexity
  - Space: $O(|E|)$
  - Find all edges/relations from a node: $O(|E|)$
- Adjacency Matrices Complexity
  - Space: $O(|V| \cdot |V| \cdot m)$

Category	Method	Theoretical	NYT	WebNLG
Edge List	CopyRe	$4kl + kr$	329	712
Adjacency Matrices	MHS	llr	57369	26518
Adjacency List (Partially)	CasRel	$2l + 2slr$	3084	15836
Adjacency List (Fully)	DIRECT	$2l + 2sl + or$	238	542

Table 11: Graph representation efficiency based on the theoretical logits amount and the estimated logits amount on two benchmark datasets.

  - Find all edges/relations from a node: $O(|V| \cdot m)$

- Adjacency List Complexity

  - Space: $O(|V| + |E|)$

  - Find all edges/relations from a node: $O(d(v))$

6.4 Graph Representation Efficiency Analysis

Based on the amount estimation of predicted logits[12] $(0/1)$, we conduct a graph representation efficiency analysis to demonstrate the efficiency of proposed method[13].

For each graph representation category, we choose one representative model algorithms. **Edge List**: CopyRE (Zeng et al., 2018); **Adjacency Matrices**: MHS (Bekoulis et al., 2018); **Adjacency List**: CasRel (partially) (Wei et al., 2020) and DIRECT (fully).

Formally, for a sentence whose length is l (l tokens), there are r types of relations, k denotes the number of triplets. Suppose there are s keys (subjects) and o values (corresponding amount of object-based lists) in adjacency list. The theoretical logits amount and the estimated logits amount on two benchmark datasets (NYT and WebNLG) are shown in Table 11. From the viewpoint of predicted logits estimation, DIRECT is the most representative-efficient model.

---

[12] Numeric output of the last layer

[13] As aforementioned, from the graph representation perspective, when a method requires fewer logits to represent the graph (set of triples), it will reduce the model fitting difficulty.

Self-Supervised Document Similarity Ranking via Contextualized Language Models and Hierarchical Inference

Dvir Ginzburg[*,1] **Itzik Malkiel**[*,1,4,†] **Oren Barkan**[§,2,4] **Avi Caciularu**[§,3] **Noam Koenigstein**[1,4]
[1]Tel-Aviv University, [2]Ariel University, [3]Bar-Ilan University, [4]Microsoft
{itmalkie, orenb, Noam.Koenigstein}@microsoft.com
{dvirginz, avi.c33}@gmail.com

Abstract

We present a novel model for the problem of ranking a collection of documents according to their semantic similarity to a source (query) document. While the problem of document-to-document similarity ranking has been studied, most modern methods are limited to relatively short documents or rely on the existence of "ground-truth" similarity labels. Yet, in most common real-world cases, similarity ranking is an unsupervised problem as similarity labels are unavailable. Moreover, an ideal model should not be restricted by documents' length. Hence, we introduce SDR, a self-supervised method for document similarity that can be applied to documents of arbitrary length. Importantly, SDR can be effectively applied to extremely long documents, exceeding the $4,096$ maximal token limit of Longformer. Extensive evaluations on large documents datasets show that SDR significantly outperforms its alternatives across all metrics. To accelerate future research on unlabeled long document similarity ranking, and as an additional contribution to the community, we herein publish two human-annotated test-sets of long documents similarity evaluation. The SDR code and datasets are publicly available [1].

1 Introduction

Text similarity ranking is an important task in multiple domains, such as information retrieval, recommendations, question answering, and more. Recent approaches based on Transformer language models such as BERT (Devlin et al., 2019) benefit from effective text representations, but are limited in their maximum input text length. Hence, developing techniques for long-text or document level matching is an emerging research field (Jiang et al., 2019).

In this work, we present SDR, a self-supervised method for document-to-document similarity ranking that can be effectively applied to extremely long documents of arbitrary length and does not require similarity labels. SDR employs a self-supervised pre-training phase that leverages: (1) a masked language model that fine-tunes *contextual word* embeddings to specialize in a given domain and (2) a contrastive loss on sentence pairs, assembled by inter-and intra-sampling, that encourages the model to produce enhanced text embeddings for similarity. Similarity inference is achieved by producing per-sentence embeddings followed by a two-staged hierarchical scoring.

Our contributions are as follows: (1) we present SDR, a novel method for document-to-document similarity that can effectively operate on long documents of arbitrary length and does not require similarity labels. We evaluate SDR and report its performance on two large datasets of documents, showcasing its ability to rank documents better than other state-of-the-art alternatives. (2) to accelerate future research, we publish two long-document similarity datasets annotated by human experts.

2 Related Work

Semantic similarity has been studied in many fields, such as computer vision (Parmar et al., 2018; Huang et al., 2017), recommender systems (Wang and Fu, 2020; Barkan et al., 2020a, 2021; Malkiel et al., 2020), and natural language processing (Devlin et al., 2019; Reimers and Gurevych, 2019; Mikolov et al., 2013). Recently, transformer-based Language Models (LMs) ushered significant performance gains in various natural language understanding tasks, but mainly on relatively short texts (Devlin et al., 2019; Liu et al., 2019). These models are usually pre-trained on the Masked Language Modeling (MLM) objective followed by a down-

---

[*], § Denotes equal contribution.

[†] Corresponding author

[1] github.com/microsoft/SDR

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3088–3098
August 1–6, 2021.

stream task-specific fine-tuning process (Wang et al., 2018). However, most models employ a battery of self-attention operations, which scale *quadratically* with the sequence length rendering extremely inefficient for long documents containing pages of text.

To mitigate scale challenges, the Longformer (Beltagy et al., 2020) model has been proposed which employs local windowed attention unit that restrains the computation and space to scale *linearly* with the sequence length. However, computation complexity still depends on the sequence length. Moreover, it entails a linear space complexity (memory usage). Therefore, in practice, the propagation of extremely long sequences remains infeasible and the maximal input of the Longformer is capped at $4,096$ tokens only, far less than many real-world long documents.

Apart from the aforementioned scale limitations on the model's input, in the case of computing pair-wise similarities between a large number of documents, the above models also suffer from an exhaustive inference process: Longformer and BERT score pairs of items in a unified feed-forward process, by which each pair of two items is fed to the model in order to produce a single pair-wise score (as opposed to scoring based on the individual item embeddings). Such inference technique, impose $O(N^2)$ feed-forward operations (Barkan et al., 2020b), compared to just $O(N)$ in SDR.

An additional challenge of Transformer-based LMs is the fact that their raw vector representations is known to perform poorly on semantic textual similarity tasks (Reimers and Gurevych, 2019). As a result, specific methods for text similarity tasks have been proposed. A prominent example for such methods is the SBERT model (Reimers and Gurevych, 2019). SBERT employs a novel fine-tuning procedure that encourages representations of similar sentences be closer in terms of the cosine similarity, substantially improving their ability to capture semantic similarity. Yet, SBERT still toils from the aforementioned complexity challenges and is unable to handle long-documents.

Recently, several works proposed long-document processing and retrieval techniques using labeled data. Cohan et al. (2020) introduced SPECTER, a model for producing document-level embedding of scientific documents. SPECTER employs a novel objective that uses paper citations as a proxy for similarity. Similarly, the Cross-Document Attention (CDA) model (Zhou et al., 2020) and the Cross-Document Language Model (CDLM) (Caciularu et al., 2021) suggest equipping language models with cross-document information for document-to-document similarity tasks. All the above methods rely on supervision, either during the pre-training phase or during fine-tuning. However, in the general case, document-to-document similarity (as well as most similarity tasks), is performed in unsupervised settings where no labels (or citations) are available.

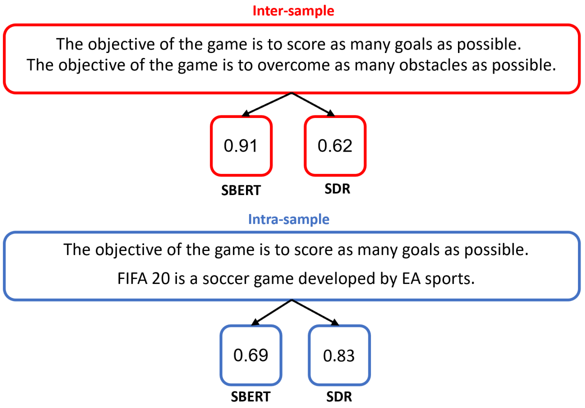

Figure 1: A representative inter- and intra-samples, along with cosine similarity scores retrieved by SBERT and SDR. Top: Inter-sampling from two documents associated with games of different categories. SBERT scores the sentences with a higher cosine value than the one retrieved by SDR. Bottom: attaching the anchor sentence with a sentence sampled from the same paragraph (and document). SDR and SBERT are reversed, where SDR yields a higher score that is more faithful to the sentences' underlying semantics and topic.

Another line of work consists of hierarchically learning single document representations. For example, Hierarchical Attention Networks (HANs) incorporate words and sentences into the final document representation showing competitive performance in different tasks involving long document encoding (Yang et al., 2016; Sun et al., 2018). More recently, Yang et al. (2020) and Jiang et al. (2019) investigated hierarchical models based on recurrent neural networks or BERT (Devlin et al., 2019) leading to state-of-the-art results in *supervised* document similarity challenges. These hierarchical models employ a bottom-up approach in which a long body of text (a document) is represented as an aggregation of smaller components i.e., paragraphs, sentences, and words. As opposed to these works, SDR exploits a document's hierarchical structure while avoiding compressing it into a single representation. This enables SDR to preserve more relevant information, leading to the superior results presented in Sec. 4. Importantly, SDR is *unsupervised* and does not require similarity labels or further fine-tuning.

3 The SDR Model

We present the problem setup followed by a description of the SDR model, its training and inference.

3.1 Problem Setup

Given a collection of documents $D = \{d_1, \ldots, d_n\}$ and a source document $s \in D$, the goal is to quantify a score that would allow us to rank all the other documents in D according to their semantic similarity with the source document s. In this work, we assume that document similarity labels are not supplied. Therefore, we propose a self-supervision loss that utilizes labels that we invent - this is a proxy to the ultimate similarity labels (if were given).

3.2 Training

SDR adopts the RoBERTa language model as a backbone, and, following Gururangan et al. (2020a), continues the pre-training of the RoBERTa model on D. Unlike RoBERTa, the SDR training solely relies on negative and positive sentence-pairs produced by inter- and intra-document sampling, respectively.

Specifically, the SDR training propagates sentence-pairs sampled from D. The sentence-pairs are sampled from the same paragraph with probability 0.5 (intra-samples), otherwise from different paragraphs taken from the different documents (inter-samples). The sentences in each pair are then tokenized, aggregated into batches, and randomly masked in a similar way to the RoBERTa pre-training paradigm. The SDR objective comprises a dual-term loss. The first term is the standard MLM loss adopted from Devlin et al. (2019). Denoted by $\mathcal{L}_{MLM}$. The MLM loss allows the model to specialize in the domain of the given collection of documents (Gururangan et al., 2020b).

The second loss term is the contrastive loss (Hadsell et al., 2006). Given a sentence pair (p, q) propagated through the model, we compute a feature vector for each sentence by average pooling the token embeddings associated with each sentence separately. The tokens embedding are the output of the last encoder layer of the model. The contrastive loss is then applied to the pair of feature vectors and aims to encourage the representations of intra-samples to become closer to each other while pushing inter-samples further away than a predefined positive margin $m \in R^+$.

Formally, the contrastive loss is defined as follows:

$$\mathcal{L}_C = \begin{cases} 1 - C\left(f_p, f_q\right) & y_{p,q} = 1 \\ \max\left(0, C\left(f_p, f_q\right) - (1-m)\right) & y_{p,q} = 0 \end{cases} \tag{1}$$

where f_p, f_q are the pooled vectors extracted from the tokens embedding of sentence p and q, respectively. $y(p, q) = 1$ indicates an intra-sample (sentence-pair sampled from the same paragraph), otherwise negative (sentence-pair from different documents). $C(f_p, f_q)$ measures the angular distance between f_p and f_q using the Cosine function:

$$C\left(f_p, f_q\right) = \frac{f_p^T f_q}{|f_p|\,|f_q|} \tag{2}$$

A demonstration of the inter-and intra-sampling procedure associated with the cosine scores produced by SDR can be found in Fig.1. The figure presents a representative sample as a motivation for SDR sampling and contrastive loss, where SDR is shown to score sentences in a way that is more faithful to their underlying topic and semantics. Importantly, as the inter-samples represent sentences that were randomly sampled from different documents, it is not guaranteed that their semantics would oppose each other. Instead, it is likely that those sentences are semantically uncorrelated while obtaining some level of opposite semantics only in rare cases. Therefore, instead of pushing negative samples to completely opposite directions, we leverage the contrastive loss in a way that encourages orthogonality between inter-samples while avoiding penalizing samples with negative scores. Hence, in our experiments, we set $m \triangleq 1$, which encourages inter-samples to have a cosine similarity that is less than or equal to 0, and do not penalize pairs with negative cosine scores.

Finally, both loss terms are combined together yielding the total loss

$$\mathcal{L}_{total} = \mathcal{L}_{MLM} + \mathcal{L}_C \tag{3}$$

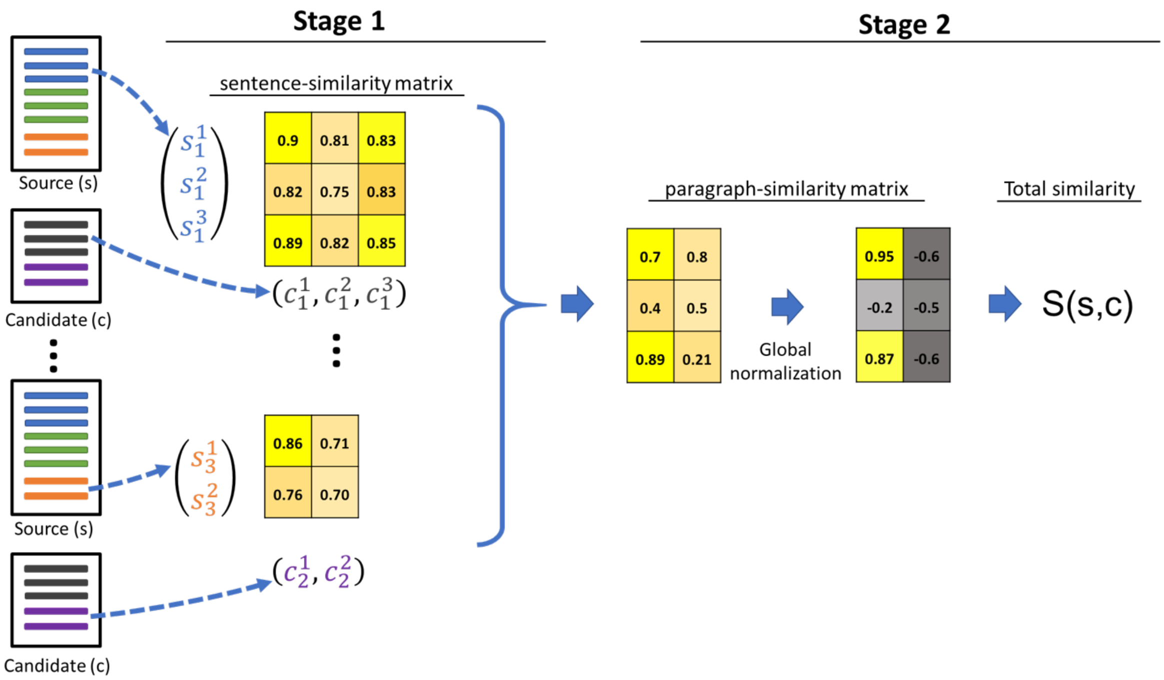

Figure 2: A schematic illustration of the SDR inference. Given a source and candidate documents, and for each paragraph-pair, SDR decomposes the paragraphs into sentences and maps each sentence into a vector. In the first stage, a sentence-similarity matrix is computed for each paragraph-pair. In the second stage, paragraph-similarity scores are inferred for all pairs and aggregated into a paragraph-similarity matrix. The matrix is then globally normalized and reduced into a total score, estimating the cumulative similarity between the two documents.

3.3 Inference

Let $s \in D$ be a source document composed of a sequence of paragraphs $s = (s_i)_{i=1}^{\widetilde{n}}$, where each paragraph comprises a sequence of sentences $s_i = (s_i^k)_{k=1}^{i^*}$, and i^* denotes the number of sentences in s_i. Similarly, let $c \in D$ be a candidate document, c can be written as $c = (c_j)_{j=1}^{m}$, where $c_j = (c_j^r)_{r=1}^{j^*}$. The SDR inference scores the similarity between s and every other candidate document c by calculating two-staged hierarchical similarity scores. The first stage operates on sentences to score the similarity between paragraph-pairs, and the second operates on paragraphs to infer the similarity between two documents. In SDR, we first map each document in D into a sequence of vectors by propagating its sentences through the model. Each sentence is then transformed into a vector by average pooling the token embeddings of the last encoder layers' outputs. Next, for each candidate document $c \in D$, SDR iterates over the feature vectors associated with the sentences in s and c and composes a *sentence-similarity matrix* for each paragraph-pair from both documents. Specifically, for each paragraph-pair $(s_i, c_j) \in s \times c$, SDR computes the cosine similarity between every pair of sentence embedding from $s_i \times c_j$, forming a sentence-similarity matrix. Focusing on the (k, r) cell of this matrix, $1 \leq k \leq i^*$, $1 \leq r \leq j^*$, the sentence-similarity matrix can be expressed as:

$$M_{ij}^{kr} \triangleq C(s_i^k, c_j^r) \tag{4}$$

Calculated for each paragraph pair $(s_i, c_j) \in s \times c$, the paragraph-similarity scores are then aggregated into a *paragraph-similarity matrix*. Focusing on the (i, j) cell, the matrix can be expressed as:

$$P_{ij}^{sc} \triangleq \frac{\sum_{k=1}^{i^*} \max\limits_{0 \leq r \leq j^*} M_{ij}^{kr}}{i^*} \tag{5}$$

The motivation behind the similarity scores in Eq. 5 is that similar paragraph-pairs should incorporate similar sentences that are more likely to correlate under the cosine metric, due to the properties of the contrastive loss employed throughout SDR training. In order to rank all the documents in the dataset, we compute the above paragraph-similarity matrix for every candidate document $c \in D$. The resulted paragraph-similarity matrices are then globally normalized. Each row i in P_{ij}^{sc} is z-score normalized by a mean and standard deviation computed from the row i values of P_{ij}^{sc} across all candidates $c \in D$.

The motivation behind this global normalization is to refine the similarity scores by highlighting the ones of the most similar paragraph-pairs and negatively scores the rest. Throughout our early experiments, we observed that different paragraph-pairs incorporate sentences with different distributions of cosine scores, where some source paragraphs may yield a distribution of cosine values with a sizeable margin compared to other paragraphs. This can be attributed to the embedding space, for which some regions can be denser than others.

Finally, a total similarity score is inferred for each candidate c, using the above paragraph-similarity matrix. The total similarity score aims to quantify the cumulative similarity between s and c. To this end, we aggregate all paragraph-similarity scores for each paragraph in s as follows:

$$\mathcal{S}(s,c) = \frac{\sum_{i=1}^{\tilde{n}} \max_{1\leq j\leq m} \left[\text{NRM}(P_{ij}^{sc})\right]_{i,j}}{n} \quad (6)$$

where NRM is the global normalization explained above. The essence of Eq.6 is to match between the most similar paragraphs from s and c, letting those most correlated paragraph-pairs contribute to the total similarity score between both documents. Finally, the ranking of the entire collection d can be obtained by sorting all candidate documents according to $\mathcal{S}(s,c)$, in a descending order.

It is important to notice that (1) in SDR inference, we do not propagate documents-pairs through the language model (which is computationally exhaustive). Instead, the documents are separately propagated through the model. Then, the scoring solely requires applications of non-parametric operations[2]. (2) both SDR training and inference operate on sentences and therefore do not suffer from discrepancies between the two phases.

4 Experiments

4.1 Datasets

We conducted our experiments over two datasets excerpted from Wikipedia. For each of the Wikipedia-based datasets, we provide a human-annotated test set of similarity labels. Examples from the datasets are provided in Fig. 3.

Wikipedia video games (WVG) The Wikipedia video games dataset[3] consists of 21,935 articles reviewing video games from all genres and consoles. Each article consists of a different combination of sections, such as summary, gameplay, plot, production, etc. For this dataset, we publish ground-truth similarity annotations, crafted by a domain expert, for $\sim$ 90 source game articles. For each source, the expert annotated $\sim$ 12 articles of similar games. Examples for the ground-truth similarities are: (1) Grand Theft Auto - Mafia, (2) Burnout Paradise - Forza Horizon 3.

Wikipedia wine articles (WWA) Wikipedia wines[4] dataset consists of 1635 articles from the wine domain. This dataset consists of a mixture of articles discussing different types of wine categories, brands, wineries, grape varieties, and more. The ground-truth similarities were crafted by a human sommelier who annotated 92 source articles with $\sim$10 similar articles, per source. Examples for ground-truth expert-based similarities are: (1) Dom Pérignon - Moët & Chandon, (2) Pinot Meunier - Chardonnay.

4.2 Quantitative Metrics

We evaluated the performance of SDR, the baselines, and ablations using the MPR,MRR and HR@k metrics:

Mean Percentile Rank (MPR) The mean percentile rank is the average of the percentile ranks for every sample with ground truth similarities in the dataset. Given a sample s, the percentile rank for a true recommendation r is the rank the model gave to r divided by the number of samples in the dataset. MPR evaluates the stability of the model, i.e, only models where all ground truth similarities had a high rank by the model will have a good score.

Mean Reciprocal Rank (MRR) The mean reciprocal rank is the average of the best reciprocal ranks for every sample with ground truth similarities in the dataset. Given a sample with M_s ground truth similarities we first mark the rank of each ground truth recommendation by the model and then take the reciprocal of the *best* (lowest) rank.

Hit Ratio at k (HR@k) HR@k evaluates the percentage of true predictions in the top k retrievals

---

[2] The cosine similarity function.

[3] Wikipedia Video Games dataset.

[4] Wikipedia Wine Articles dataset.

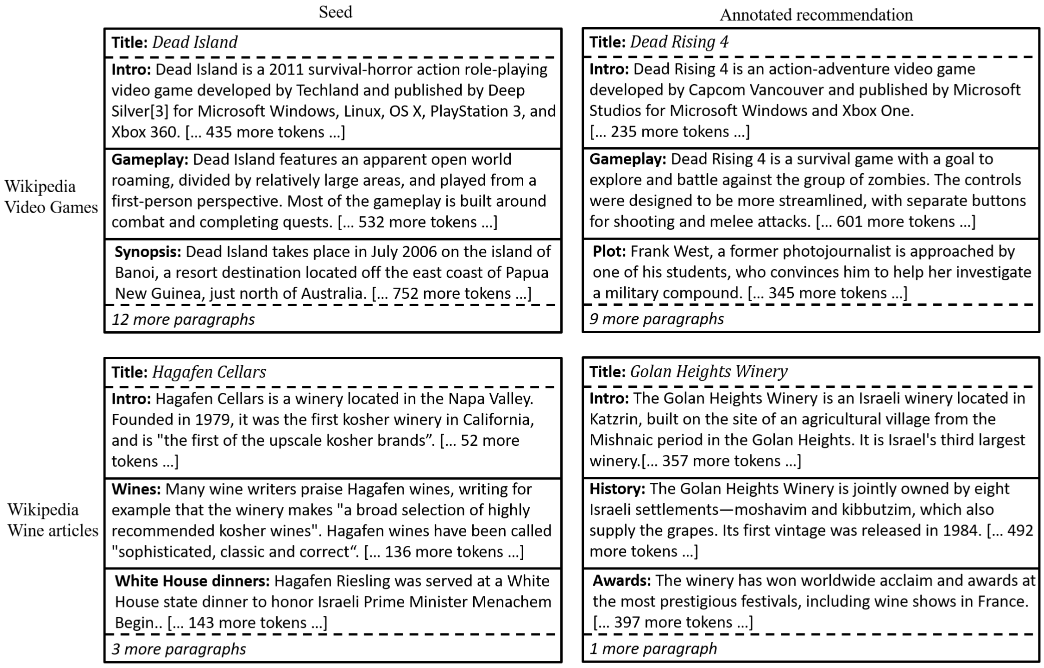

	Seed	Annotated recommendation
Wikipedia Video Games	**Title:** *Dead Island*	**Title:** *Dead Rising 4*
	Intro: Dead Island is a 2011 survival-horror action role-playing video game developed by Techland and published by Deep Silver[3] for Microsoft Windows, Linux, OS X, PlayStation 3, and Xbox 360. [... 435 more tokens ...]	**Intro:** Dead Rising 4 is an action-adventure video game developed by Capcom Vancouver and published by Microsoft Studios for Microsoft Windows and Xbox One. [... 235 more tokens ...]
	Gameplay: Dead Island features an apparent open world roaming, divided by relatively large areas, and played from a first-person perspective. Most of the gameplay is built around combat and completing quests. [... 532 more tokens ...]	**Gameplay:** Dead Rising 4 is a survival game with a goal to explore and battle against the group of zombies. The controls were designed to be more streamlined, with separate buttons for shooting and melee attacks. [... 601 more tokens ...]
	Synopsis: Dead Island takes place in July 2006 on the island of Banoi, a resort destination located off the east coast of Papua New Guinea, just north of Australia. [... 752 more tokens ...]	**Plot:** Frank West, a former photojournalist is approached by one of his students, who convinces him to help her investigate a military compound. [... 345 more tokens ...]
	12 more paragraphs	*9 more paragraphs*

	Seed	Annotated recommendation
Wikipedia Wine articles	**Title:** *Hagafen Cellars*	**Title:** *Golan Heights Winery*
	Intro: Hagafen Cellars is a winery located in the Napa Valley. Founded in 1979, it was the first kosher winery in California, and is "the first of the upscale kosher brands". [... 52 more tokens ...]	**Intro:** The Golan Heights Winery is an Israeli winery located in Katzrin, built on the site of an agricultural village from the Mishnaic period in the Golan Heights. It is Israel's third largest winery.[... 357 more tokens ...]
	Wines: Many wine writers praise Hagafen wines, writing for example that the winery makes "a broad selection of highly recommended kosher wines". Hagafen wines have been called "sophisticated, classic and correct". [... 136 more tokens ...]	**History:** The Golan Heights Winery is jointly owned by eight Israeli settlements—moshavim and kibbutzim, which also supply the grapes. Its first vintage was released in 1984. [... 492 more tokens ...]
	White House dinners: Hagafen Riesling was served at a White House state dinner to honor Israeli Prime Minister Menachem Begin.. [... 143 more tokens ...]	**Awards:** The winery has won worldwide acclaim and awards at the most prestigious festivals, including wine shows in France. [... 397 more tokens ...]
	3 more paragraphs	*1 more paragraph*

Figure 3: Samples from the Wikipedia Video Games (WVG) and Wikipedia Wines Articles (WWA) datasets. For each seed item (left), the opening sentence of each of the first three paragraphs is presented. A recommended sample by the domain expert is shown on the right.

made by the model, where a true prediction corresponds a candidate sample from the ground truth annotations.

4.3 Baseline models

We compare SDR with the following baselines:

Latent Dirichlet Allocation (LDA) LDA (Blei et al., 2003) is one of the renowned algorithms for topic modeling and text-matching. LDA assumes that documents are generated by sampling from a distribution of latent topics, where each topic can be described by another distribution defined over the vocabulary. For every LDA experiment, we perform a grid search with $1,000$ different configuration of hyper-parameters. The reported performance corresponds to the model with the highest topic coherence value (Newman et al., 2010).

BERT and Longformer For BERT (Devlin et al., 2019) and Longformer (Beltagy et al., 2020) (see Sec. 2), we evaluate two different variants. First, we employ the publicly available pre-trained weights of the models. Second, we continue the pre-training of the models over the corpora induced by the datasets, applying our proposed method associated with each model. We used only the "large" architectures during all the experiments.

SBERT The SBERT model (Reimers and Gurevych, 2019) utilizes a fine-tuning approach that produces semantically meaningful embeddings under a cosine-similarity metric.

We evaluate SBERT model under two inference configurations (1) using the original weights trained on the NLI dataset (Bowman et al., 2015), and (2) after fine-tuning with the pseudo labels presented in Sec. 3.2.

4.4 Inference Methods

To compare SDR with the above baselines, which are restricted by a maximal sequence length, we follow previous procedures (Reimers and Gurevych, 2019; Beltagy et al., 2020) and report the performance of four different inference techniques applied on the output embeddings of the different models :

- **CLS** - use the special CLS token embedding of the N^5 first tokens.
- **FIRST** - use the mean of the embeddings of the N first tokens.
- **ALL** - propagating the entire document in

---

[5] N is the maximal sequence length the model supports in one forward pass.

		Video games				Wines			
Architecture	**Inference**	**MPR**	**MRR**	**HR@10**	**HR@100**	**MPR**	**MRR**	**HR@10**	**HR@100**
LDA	-	94.1%	31.8%	8.8%	28.1%	83.7%	23.4%	8.7%	41.3%
SBERT	First	86.4%	42.6%	11.9%	26.9%	83.5%	31.1%	11.4%	41.8%
SBERT	ALL	92.6%	51.1%	16.1%	37.5%	81.3%	28.3%	11.1%	37.2%
SBERT	SDR_{inf}	94.2%	53.4%	18.2%	39.7%	83.6%	32.3%	12.1%	41.0%
Longformer	CLS	58.0%	10.4%	2.1%	6.3%	65.0%	15.7%	4.8%	14.6%
Longformer	First	66.6%	3.3%	1.3%	3.7%	56.1%	12.4%	3.5%	10.2%
Longformer	ALL	66.0%	9.7%	2.4%	4.3%	64.7%	13.9%	2.2%	11.8%
Longformer	SDR_{inf}	68.5%	10.2%	4.1%	7.7%	55.3%	16.4%	3.3%	12.3%
BERT large	CLS	69.6%	30.9%	9.7%	20.3%	70.1%	30.3%	9.5%	34.7%
BERT large	First	61.1%	15.5%	3.8%	9.8%	71.3%	21.8%	6.7%	20.9%
BERT large	ALL	65.2%	27.2%	7.1%	16.2%	64.6%	27.8%	7.8%	26.2%
BERT large	SDR_{inf}	71.2%	33.5%	12.9%	22.2%	75.5%	24.7%	9.7%	36.8%
SDR	SDR_{inf}	**97.4%**	**64.0%**	**23.6%**	**54.0%**	**89.3%**	**50.9%**	**17.0%**	**59.0%**

Table 1: Similarity results evaluated on the video games (left), movies (middle) and wines (right) datasets from wikipedia, based on expert annotations. The second column specifies the applied inference method, as described in Section 4.4. SBERT_{V} refers to the vanilla SBERT (without continuing training on each dataset by utilizing our pseudo-labels).

chunks, then use the mean of the embeddings of all the tokens in the sample.

- **SDR_{inf}** - use the hierarchical SDR inference described in Sec. 3.3.

4.5 Results

The results over the document similarity benchmarks are depicted in Tab. 1. The scores are based on the ground-truth expert annotations associated with each dataset. The results indicate that SDR outperforms all other models by a sizeable margin. Recall that the underlying LMs we evaluated (BERT, Longformer) were pre-trained on the MLM objective. This makes them hard to generate meaningful embeddings suitable for probing similarity using the Cosine-similarity metric, as previously discussed in Sec. 2. Comparing to the best variant of each model, SDR presents absolute improvements of ∼7-12% and ∼11-13% in MPR, and MRR, respectively, and across all datasets.

SBERT, as opposed to the underlying models above, presents a cosine similarity-based loss during training. Compared to SDR, we observe that a fine-tuned SBERT, which utilizes the pseudo-labels introduced in Sec. 3.2, shows inferior results across all datasets, yielding -3% MPR, -5% MRR and -2% HR@10 in the Video games. This can be attributed to SBERT's cosine loss, that constantly penalizes negative pairs to reach a cosine score of -1. For uncorrelated sentence-pairs, such property can hinder the convergence of the model. See the below ablation analysis for more details. We observe that SBERT's suffers from an additional degradation in performance when applied with the original SBERT weights, yielding -6% MPR and -8% MRR. This can be attributed to the importance of continue training on the given dataset at hand.

Notably, as shown in the table, applying the SDR inference to other baseline language models improves their performance by a significant margin. This is another evidence of our inference's advantage over other methods, especially as we observe sizeable gains across all baseline models and datasets.

Inspecting SBERT results, we see that the SDR_{inf} gains increase in all metrics, yielding an increase of at least +3% MPR, +4% MRR, +6% HR@10 and +7% HR@100. This can be attributed to the importance of the hierarchical evaluation

	Video games		Wines	
Seed / Model	Dead Island	Mafia III	Hagafen Cellars	Champagne
SDR	1. Dead Island: Riptide	1. Mafia II	1. Golan Heights Winery	1. Champagne Krug
	2. Dying Light	2. Saints Row 2	2. Manischewitz	2. Sparkling wine
	3. Dead Rising 4	3. Grand Theft Auto V	3. Barkan Wine Cellars	3. Champagne Krug
SBERT	1. Fallout 3	1. Red Dead Redemption 2	1. Petit Rouge	1. Moët & Chandon
	2. Dead Rising 3	2. Dark Souls II	2. Roter Veltliner	2. Chardonnay
	3. Wasteland 2	3. Battlefield	3. Trisaetum Winery	3. Chasselas
BERT	1. The Outer Worlds	1. The Godfather	1. Domaine Dujac	1. Roter Veltliner
	2. Metro Exodus	2. Dark Souls	2. Petri Wine	2. Table wine
	3. Rage 2	3. Code Vein	3. Blue Nun	3. Champagne wine region

Table 2: Similarity predictions for the Wikipedia video games (WVG) and Wikipedia wine articles (WWA) datasets. For each of the shown recommendations, a domain expert rated the similarity with the source document. Red, yellow, and green indicate poor, mediocre, and high similarity (respectively).

for long documents and indicate the struggle transformers have in embedding long text into a single vector. Importantly, SDR outperforms SBERT by a significant margin, even when SBERT is applied with SDR_{inf}. This is due to SDR training, which incorporates the contrastive loss for promoting orthogonality between negative sentence-pairs.

Table 2 presents qualitative results on randomly chosen samples from the WWA and WVG datasets. We compare SDR with the top two baselines associated with the highest scores in the Wikipedia evaluations, namely SBERT and BERT. Similarly to the evaluation scheme presented for the quantitative experiments, we employ a self-supervised training for SBERT, with the same pseudo-labels as in SDR training. In Tab. 2, we observe that SDR correctly understands the essence of the article, as finding *Grand Theft Auto V* similar to *Mafia III*, or *Sparkling wine* similar to *Champagne*. As to SBERT results, we see that the model fails to grasp the article's underlying topic in 50% of the predictions. For example, SBERT matches between *Battlefield* and *Mafia III*, or *Chasselas* and *Champagne*. This can be attributed to the fact that SBERT does not apply a hierarchical inference and struggles to compress the entire document representation in one vector. This becomes especially crucial in very long documents, which are common in the WVG dataset. In BERT, we observe document similarity predictions of relatively poor quality. For example, for *Hagafen Cellars* BERT retrieves *Blue Nun*, or for *Champagne* it matches *Table wine*. The relative degradation in performance can be attributed to the BERT pre-training procedure, which inherently does not optimize text embedding under a well-defined metric.

The above results highlight the benefit obtained by the SDR model, which utilizes a hierarchical inference, along with a self-supervised training procedure that embeds sentences under a well-defined similarity metric.

4.6 Ablation Study

We performed an ablation study to asses the effectiveness of SDR. To that end, we used the video games dataset, described in Sec. 4.1. The following ablations are considered:

- No hierarchical inference - the embeddings of the first N tokens of each document are averaged, producing one embedding vector per document. These embeddings are compared via the cosine function to score the similarity between documents. This is similar to the scoring procedure from (Reimers and Gurevych, 2019).
- Paragraph-level inference - the paragraph-similarity matrix is computed directly using the first N tokens of each paragraph. This variant neglects the sentence-similarity matrix from stage 1 2 of the inference mechanism. The scoring proceeds by stage 2 of the inference, as described in Sec. 3.3.
- No training - the BERT pre-trained weights are used and applied with the proposed hierarchical inference (i.e., we do not employ additional pre-training on the given collection of documents).
- Global normalization - the SDR inference is applied without globally normalizing the paragraph-similarity matrix.

	Video games		
	MPR	**MRR**	**HR@10**
(i) No hierarchical inference	96.3%	52.4%	20.1%
(ii) Paragraph-level inference	97.4%	58.5%	22.8%
(iii) No training	87.1%	28.2%	7.0%
(iv) No normalization	97.3%	63.2%	22.6%
(v) No contrastive loss	91.5%	46.0%	14.5%
Full method	**97.4%**	**64.0%**	**23.6%**

Table 3: Ablation study results.

- No contrastive loss - the SDR training is applied without the contrastive loss term (solely using the MLM objective).
- Standard cosine loss - the SDR training employs a contrastive loss with a margin of $m = 2$. This is equivalent to the standard Cosine-Similarity loss, that reinforces negative and positive samples to cosine scores of -1 and 1, respectively.

The results depicted in Tab.3 indicate that our proposed hierarchical inference is highly beneficial, even compared to a paragraph-level inference, that it is crucial to employ the proposed training in the way it is done in SDR, and that it is better to apply global normalization.

Particularly noticeable is the contrastive loss, whose gain is present in both (ii) and (iii), for which the biggest degradation in the results took place. Another significant improvement is due to the hierarchical inference, with a leap of 11% in MPR by applying paragraph-level inference, and another 9% by applying the two-stage hierarchy.

4.7 Implementation details

SDR and all other transformer-based baselines utilize the Huggingface package [6]. In our transformer-based baselines experiments, we use the best-published model configuration associated with each variant. To split the paragraphs into sentences as suggested in SDR, we used the NLTK package [7], resulting in an average sentence length of 16 tokens. We use a train-validation split of 90%-10% to evaluate the MLM and cosine similarity accuracy during training.

For LDA modeling and similarity evaluation, we used the implementation of the Gensim package[8]. We conduct a hyperparameter search, based on the topic coherence score, to find the best LDA parameters for each dataset.

For SBERT we used the official package[9], with the released fine-tuned weights for the STS task. All our experiments were conducted using a single Tesla V100 32GB card, with a batch size of 8 both for training and evaluation.

5 Conclusions

In this work, we presented Self-Supervised Document Similarity Ranking (SDR), a novel self-supervised model for document similarity, supporting extremely long documents. Documents' similarities are extracted via a hierarchical bottom-up scoring procedure, which preserves more semantic information, leading to superior similarity results. For our evaluations, we assembled two manually-labeled test-sets using expert annotations, that will be made publicly available to expedite future research on long-document similarities.

References

Oren Barkan, Roy Hirsch, Ori Katz, Avi Caciularu, Yoni Weill, and Noam Koenigstein. 2021. Cold start revisited: A deep hybrid recommender with cold-warm item harmonization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Oren Barkan, Ori Katz, and Noam Koenigstein. 2020a. Neural attentive multiview machines. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3357–3361. IEEE.

Oren Barkan, Noam Razin, Itzik Malkiel, Ori Katz, Avi Caciularu, and Noam Koenigstein. 2020b. Scalable

---

[6] HuggingFace

[7] nltk

[8] Gensim

[9] SBERT API

attentive sentence pair modeling via distilled sentence embedding. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Avi Caciularu, Arman Cohan, Iz Beltagy, Matthew E Peters, Arie Cattan, and Ido Dagan. 2021. Cross-document language modeling. *arXiv preprint arXiv:2101.00406*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020a. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020b. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Mike Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents. In *Proceedings of the Word Wide Web Conference (WWW)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Itzik Malkiel, Oren Barkan, Avi Caciularu, Noam Razin, Ori Katz, and Noam Koenigstein. 2020. RecoBERT: A catalog language model for text-based recommendations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1704–1714, Online. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 100–108.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Tian Wang and Yuyangzi Fu. 2020. Item-based collaborative filtering with BERT. In *Proceedings of The 3rd Workshop on e-Commerce and NLP*, pages 54–58, Seattle, WA, USA. Association for Computational Linguistics.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for document matching. *arXiv preprint arXiv:2004.12297*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Xuhui Zhou, Nikolaos Pappas, and Noah A. Smith. 2020. Multilevel text alignment with cross-document attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5012–5025, Online. Association for Computational Linguistics.

How *Good* Is NLP? A Sober Look at NLP Tasks through the Lens of Social Impact

Zhijing Jin
Max Planck Institute for Intelligent Systems
Tübingen, Germany
`zjin@tue.mpg.de`

Geeticka Chauhan
MIT
`geeticka@mit.edu`

Brian Tse
Oxford
`briantsemanhei@gmail.com`

Mrinmaya Sachan
ETH Zürich
`msachan@ethz.ch`

Rada Mihalcea
University of Michigan
`mihalcea@umich.edu`

Abstract

Recent years have seen many breakthroughs in natural language processing (NLP), transitioning it from a mostly theoretical field to one with many real-world applications. Noting the rising number of applications of other machine learning and AI techniques with pervasive societal impact, we anticipate the rising importance of developing NLP technologies for social good. Inspired by theories in moral philosophy and global priorities research, we aim to promote a guideline for social good in the context of NLP. We lay the foundations via the moral philosophy definition of social good, propose a framework to evaluate the direct and indirect real-world impact of NLP tasks, and adopt the methodology of global priorities research to identify priority causes for NLP research. Finally, we use our theoretical framework to provide some practical guidelines for future NLP research for social good.[1]

1 Introduction

Advances on multiple NLP fronts have given rise to a plethora of applications that are now integrated into our daily lives. NLP-based intelligent agents like Amazon Echo and Google Home have entered millions of households (Voicebot, 2020). NLP tools are now prevalent on phones, in cars, and in many daily services such as Google search and electronic health record analysis (Townsend, 2013).

In the current COVID-19 context, NLP has already had important positive social impact in the face of a public health crisis. When the pandemic broke out, Allen AI collected the CORD-19 dataset (Wang et al., 2020) with the goal of helping public health experts efficiently sift through the myriad of COVID-19 research papers that emerged in a short time period. Subsequently, NLP services such as Amazon Kendra were deployed to help organize the research knowledge around COVID-19 (Bhatia et al., 2020). The NLP research community worked on several problems like the question-answering and summarization system CAiRE-COVID (Su et al., 2020), the expressive interviewing conversational system (Welch et al., 2020) and annotation schemas to help fight COVID-19 misinformation online (Alam et al., 2020; Hossain et al., 2020).

As NLP transits from theory into practice and into daily lives, unintended negative consequences that early theoretical researchers did not anticipate have also emerged, from the toxic language of Microsoft's Twitter bot Tay (Shah and Chokkattu, 2016), to the leak of privacy of Amazon Alexa (Chung et al., 2017). A current highly-debated topic in NLP ethics is GPT-3 (Brown et al., 2020), whose risks and harms include encoding gender and racist biases (Bender et al., 2021).

It is now evident that we must consider the negative and positive impacts of NLP as two sides of the same coin, a consequence of how NLP and more generally AI pervade our daily lives. The consideration of the negative impacts of AI has engendered the recent and popular interdisciplinary field of AI ethics, which puts forth issues such as algorithmic bias, fairness, transparency and equity with an aim to provide recommendations for ethical development of algorithms.

Highly influential works in AI ethics include (Buolamwini and Gebru, 2018; Mitchell et al., 2019; Raji et al., 2020; Chen et al., 2019; Blodgett et al., 2020). AI for social good (AI4SG) (Tomašev et al., 2020) is a related sub-field that benefits from results of AI ethics and while keeping ethical principles as a pre-requisite, has the goal of creating positive impact and addressing society's biggest challenges. Work in this space includes (Wang et al., 2020; Bhatia et al., 2020; Killian et al., 2019; Lampos et al., 2020).

---

[1] Our data and code are available at `http://github.com/zhijing-jin/nlp4sg_acl2021`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3099–3113
August 1–6, 2021.

Active conversations about ethics and social good have expanded broadly, in the NLP community as well as the broader AI and ML communities. Starting with early discussions in works such as (Hovy and Spruit, 2016; Leidner and Plachouras, 2017), the communities introduced the first workshop on ethics in NLP (Hovy et al., 2017) and the AI for social good workshop (Luck et al., 2018), which inspired various follow-up workshops at venues like ICML and ICLR. The upcoming NLP for Positive Impact Workshop (Field et al., 2020) finds inspiration from these early papers and workshops. In 2020, NeurIPS required all research papers to submit broader impact statements (Castelvecchi, 2020; Gibney, 2020). NLP conferences followed suit and introduced optional ethical and impact statements, starting with ACL in 2021 (Association for Computational Linguistics, 2021).

With the growing impact of our models in daily lives, we need comprehensive guidelines for following ethical standards to result in positive impact and prevent unnecessary societal harm. (Tomašev et al., 2020) provide general guidelines for successful AI4SG collaborations through the lens of United Nations (UN) sustainable development goals (SDGs) (United Nations, 2015) and (Hovy and Spruit, 2016; Leidner and Plachouras, 2017) begin the ethics discussions in NLP. However, there is room for iteration in terms of presenting a comprehensive picture of NLP for social good, with an evaluation framework and guidelines. At the moment, researchers eager to make a beneficial contribution need to base their research agenda on intuition and word of mouth recommendations, rather than a scientific evaluation framework.

To this end, our paper presents a modest effort to the understanding of social good, and sketches thinking guidelines and heuristics for NLP for social good. Our main goal is to answer the question:

> Given a specific researcher or team with skills s, and the set of NLP technologies $\boldsymbol{T}$ they can work on, what is the best technology $t \in \boldsymbol{T}$ for them to optimize the social good impact I?

In order to answer this overall question, we take a multidisciplinary approach in our paper:

- §2 relies on theories in moral philosophy to approach what is social good versus bad (i.e., the sign and rough magnitude of impact I for a *direct* act a);
- §3 relies on causal structure models as a framework to estimate I for $t \in \boldsymbol{T}$, considering that t can be an *indirect* cause of impact;
- §4 relies on concepts from global priorities research and economics to introduce a high-level framework to choose a technology t that optimizes the social impact I;
- §5 applies the above tools to analyze several example NLP directions, and provides a practical guide on how to reflect on the social impact of NLP.

We acknowledge the iterative nature of a newly emerging field in NLP for social good, requiring continuing discussions on definitions and the development of ethical frameworks and guidelines. Echoing the history of scientific development (Kuhn, 2012), the goal of our work is not to provide a perfect, quantitative, and deterministic answer about how to maximize social good with our NLP applications. The scope of our work is to take one step closer to a comprehensive understanding, through high-level philosophies, thinking frameworks, together with heuristics and examples.

2 What is social good?

Defining social good can be controversial. For example, if we define saving energy as social good, then what about people who get sick because of not turning on the air-conditioner on a cold day? Therefore, social good is context-dependent, relevant to *people*, *times*, and *states of nature* (Broome, 2017). This section is to provide a theoretical framework about the social impact I for a direct act a.

2.1 Moral philosophy theories

We can observe that for some acts, it is relatively certain to judge whether the impact is positive or negative. For example, solving global hunger is in general a positive act. Such judgement is called intuitionalism (Sidgwick, 1874), a school of moral philosophy.

There are many areas of social impact that cannot receive consensus by intuitions. To find analytical solutions to these debatable topics, several moral philosophies have been proposed. We introduce below three categories of philosophical perspectives to judge moral laws (Kagan, 2018), and provide the percentage of professional philosophers who support the theory (Bourget and Chalmers, 2014):

1. Deontology: emphasizes duties or rules, endorsed by 25.9% philosophers;
2. Consequentialism: emphasizes consequences of acts, endorsed by 23.6% philosophers;
3. Virtue ethics: emphasizes virtues and moral character, endorsed by 18.2% philosophers.

Note that the above three schools, deontology, consequentialism, and virtue ethics, follows the standard textbook introductions for normative ethics in the analytic philosophy tradition. It is also possible for future research to consider different perspectives while defining social good.

A practical guide for using these philosophies. The three perspectives provide us dimensions to think about the impact I of an act a, so that the final decision is (hopefully) more reliable than one single thought which is subject to biases. Such decomposition practices are often used in highly complicated analyses (e.g., business decisions), such as radar charts to rate a decision/candidate or SMART goals.

A practical guide for using moral philosophies to judge an act a is to think along each of the three perspectives, collect estimations of how good the act a is from the three dimensions, and merge them. For example, using NLP for healthcare to save lives can be good from all three perspectives, and thus it is an overall social good act.

When merging judgements from the above philosophical views, there can be tradeoffs, such as sacrificing one life for five lives in the Trolley problem (Thomson, 1976), which scores high on consequentialism but low on deontology and virtue ethics. One solution by the moral uncertainty theory (MacAskill et al., 2020) is to favor acts with more balanced judgements on all criteria, and reject acts that are completely unacceptable on any criterion.

2.2 Principles for future AI

Many agencies from academia, government, and industries have proposed principles for future AI (Jobin et al., 2019), which can be regarded as a practical guide by *deontology*. Zeng et al. (2019) surveyed the principles of the governance of AI proposed by 27 agencies. The main areas are as follows (with keywords):

- Humanity: beneficial, well-being, human right, dignity, freedom, education, human-friendly.
- Privacy: personal information, data protection, explicit confirmation, control of the data, notice and consent.
- Security: cybersecurity, hack, confidential.
- Fairness: justice, bias, discrimination.
- Safety: validation, test, controllability.
- Accountability: responsibility.
- Transparency: explainable, predictable, intelligible.
- Collaboration: partnership, dialog.
- Share: share, equal.
- AGI: superintelligence.

3 Evaluating the indirect impact of NLP

Given the general moral guide to judge an act with *direct* impacts, we now step towards the second stage – understanding the downstream impact of scientific research which typically has *indirect* impacts. For example, it is not easily tractable to estimate the impact of some linguistic theories. To sketch a solution, this section will first classify NLP tasks by the dimension of theory→application, and then provide an evaluation framework for I of a technology t that may have indirect real-life impacts.

3.1 Classifying tasks from upstream to downstream

To evaluate each NLP research topic, we propose four stages in the *theory→application* development, as shown in Figure 1, and categorize the 570 long papers from ACL 2020[2] according to the four stages in Figure 2. Details of the annotation are in Appendix A. The four stages are as follows.

Stage 1. Fundamental theories. Fundamental theories are the foundations of knowledge, such as linguistic theories by Noam Chomsky. In ACL 2020, the most prevalent topic for papers in Stage 1 is linguistics theory in Figure 2.

Stage 2. Building block tools. Moving one step from theory towards applications is the research on building block tools, which serves as important building blocks and toolboxes for downstream technologies. The most frequently researched Stage-2 topics at ACL 2020 are information extraction, model design, and interpretability (in Figure 2).

---

[2] `https://www.aclweb.org/anthology/events/acl-2020/#2020-acl-main`

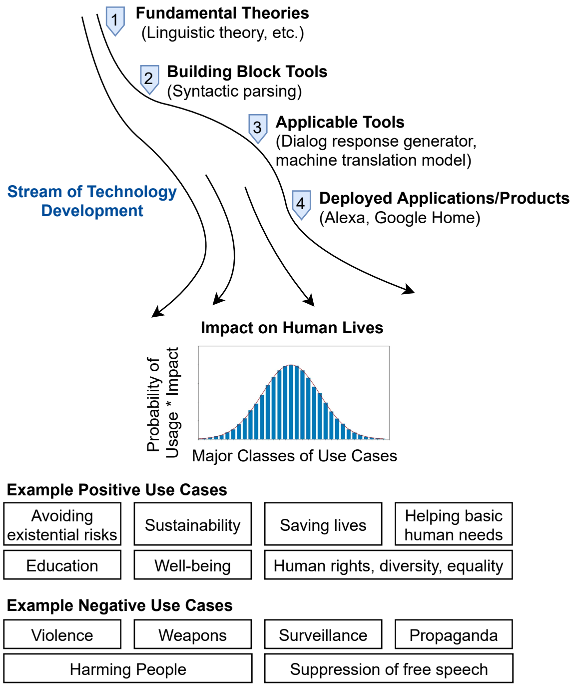

Figure 1: Stream of technology development from theory to application with end impacts. The end impacts are a distribution of use cases and their corresponding weighted impacts.

Stage 3. Applicable tools. Applicable tools are pre-commercialized NLP systems which can serve as the backbones of real-world applications. This category includes NLP tasks such as dialog response generation, question answering, and machine translation. The most common research topics in this category are dialog, machine translation, and question answering (in Figure 2).

Stage 4. Deployed applications/products. Deployed applications often build upon tools in Stage 3, and wrap them with user interfaces, customer services, and business models. Typical examples of Stage-4 technologies include Amazon Echo, Google Translate, and so on. The top three topics of ACL 2020 papers in this category are ways to address misinformation (e.g., a fact checker for news bias), dialog, and NLP for healthcare.

3.2 Estimating impact

***Direct* impacts of Stage-4 technologies.** A direct impact of NLP development is allowing users more free time. This is evident in automatic machine translation, which saves the effort and time of human translators, or in NLP for healthcare, which allows doctors to more quickly sift through patient history. Automatic fake news detection frees up time for human fact-checkers, to aid them in more quickly detecting fake news through the increasing number of digital news articles being published.

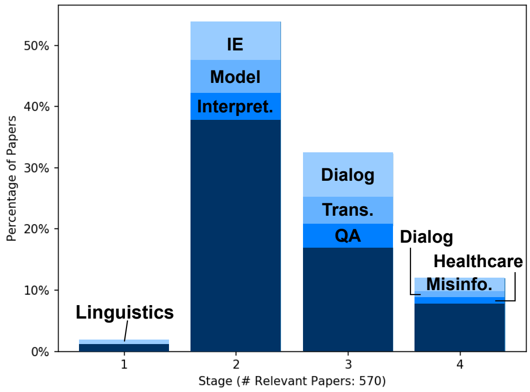

Figure 2: Distribution of ACL 2020 papers by the four stages. For each stage, we highlight the top several topics of the papers. We only list the top one topic for Stage 1 due to visual space limit. Abbreviations of technologies include Information Extraction (IE), Interpretability (Interpret.), machine translation (Trans.), question answering (QA), and misinformation (Misinfo.).

The impact of more user free time is varied. In the case of healthcare, NLP can free up time for more personalized patient care, or allow free time for activities of choice, such as spending time on passion projects or more time with family. We recognize these varied impacts of NLP deployment, and recommend user productivity as one way to measure it.

Note that there can be positive as well as negative impact associated with rising productivity, and the polarity can be decided according to Section 2.1. Typical positive impacts of NLP technology include better healthcare and well-being, and in some cases it indirectly helps with avoiding existential risks, sustainability, and so on. Typical negative impacts include more prevalent surveillance, propaganda, breach of privacy, and so on. For example, intelligent bots can improve efficiency at work (to benefit economics), and bring generally better well-being for households, but they might leak user privacy (Chung et al., 2017).

Thus, estimating the overall end impact of a technology t in the Stage 4 needs to accumulate over a set of aspects $\boldsymbol{AS}$:

$$I(t) = \sum_{as \in \boldsymbol{AS}} \text{scale}_{as}(t) \cdot \text{impact}_{as}(t) \,, \qquad (1)$$

where $\text{scale}_{as}(t)$ is the usage scale of applications of technology t used in the aspect as, and

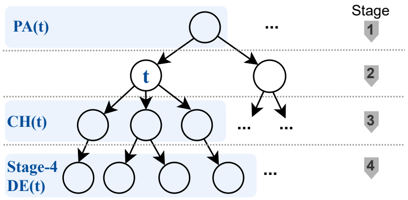

Figure 3: Viewing the theory→application process with a structural causal model.

$\text{impact}_{as}(t)$ is the impact of t in this aspect.

***Indirect* impacts of early stage technologies.** Although the direct impact of Stage-4 technologies can be estimated by Eq. (1), it is difficult to calculate the impact of a technology in earlier stages (i.e., stage 1-3).

We can approach the calculation of indirect impacts I of an early-stage technology t by a structural causal model. As shown in the causal graph $\mathcal{G}$ in Figure 3, each technology t is in a causal chain from its parent vertex set $\text{PA}(t)$ (i.e., upstream technologies that directly causes the invention of t), to its children vertex set (i.e., downstream technologies directly resulting from t). Formally, we denote a directed (causal) path in $\mathcal{G}$ as a sequence of distinct vertices $(t_1, t_2, \ldots, t_n)$ such that $t_{i+1} \in \text{CH}(t_i)$ for all $i = 1, \ldots, n-1$. We call t_n a descendant of t_1. After enumerating all paths, we denote the set of all descendants of t as $\text{DE}(t)$. Specifically, we denote all descendant nodes in Stage 4 as $\text{Stage-4 DE}(t)$.

Hence, the impact of any technology t is the sum of impact of all its descendants in Stage 4:

$$I(t) = \sum_{x \in \text{Stage-4 DE}(t)} p(x) \cdot c_x(t) \cdot I(x) \,, \quad (2)$$

where $p(x)$ is the probability that the descendent technology x can be successfully developed, $c_x(t)$ is the contribution of t to x, and $I(x)$ can be calculated by Eq. (1). This formula can also be interpreted from the light of do-calculus (Pearl, 1995) as $P(X|\text{do}(t)) - P(X)$, for $X \in \text{Stage-4 DE}(t)$, which means the effect of intervention $\text{do}(t)$ on Stage 4 descendants.

Note that Eq. (1) and (2) are meta frameworks, and we leave it to future work to utilize these for assessing the social impact of their work.

3.3 Takeaways for NLP tasks

With the growing interest of AI and NLP publication venues (e.g., NeurIPS, ACL) in ethical and broader impact statements, it will be useful and important for researchers to have practical guidelines on evaluating the impact of their NLP tasks.

We first introduce some thinking steps to estimate the impact of research on an NLP task t:

(S1) Classify the NLP task t into one of the four stages (§3.1)

(S2) If t is in Stage 4, think of the set of aspects $\boldsymbol{AS}$ that t will impact, the scale of applications, and aspect-specific impact magnitude. Finally, estimate impact using Eq. (1).

(S2') If t is in Stage 1-3, think of its descendant technologies, their success rate, and the contribution of t to them. Finally, estimate impact using Eq. (1) and (2).

Next, we introduce some high-level heuristics to facilitate fast decisions:

(H1) For earlier stages (i.e., Stage 1-2), it is challenging to quantify the exact social impact. Their overall impact tends to lean towards positive as they create more knowledge that benefits future technology development.

(H2) Developers of Stage-4 technologies should be the most careful about ethical concerns. Enumerate the use cases, and estimate the scale of each usage by thinking of the stakeholders, economic impact, and users in the market. Finally, evaluate the final impact before proceeding. (E.g., if the final impact is very negative, then abandon or do it with restrictions).

(H3) For Stage-3 technologies, if their Stage-4 descendants are tractable to enumerate and estimate for their impacts, then aggregate the descendants' impacts by Eq. 2. Otherwise, treat them like (H1).

4 Deciding research priority

There are many directions for expansion of our efforts for social good; however, due to limited resources and availability of support for each researcher, we provide a research priority list. In this section, we are effectively trying to answer the overall question proposed in Section 1. Specifically, we adopt the practice in the research field *global priorities* (GP) (MacAskill, 2015; Greaves and McAskill, 2017). We first introduce the high-level decision-making framework in Section 4.1, and then formulate these principles using technical terms in Section 4.2.

4.1 Important/Neglected/Tractable (INT) framework

Our thinking framework to address the research priority follows the practice of existing cost-benefit analysis in GP (MacAskill, 2015; Greaves and McAskill, 2017), which aligns with the norms in established fields such as development economics, welfare economics, and public policy.

We draw an analogy between the existing GP research and NLP for social good. Basically, GP addresses the following problem: given, for example, 500 billion US dollars (which is the annual worldwide expenditure on social good), what priority areas should we spend on? Inspired by this practical setting, we form an analogy to NLP research efforts, namely to answer the question proposed in Section 1 about how to attribute resources and efforts on NLP research for social good.

The high-level intuitions are drawn from the Important/Neglected/Tractable (INT) framework (MacAskill, 2015), a commonly adopted framework in global priorities research on social good. Assume each agent has something to contribute (e.g., money, effort, etc.). It is generally effective to contribute to important, neglected, and tractable areas.

4.2 Calculation of priority

Although the INT framework is commonly used in practice of many philanthropy organizations (MacAskill, 2015), it will be more helpful to formulate it using mathematical terms and economic concepts. Note that the terms we formulate in this section can be regarded as elements in our proposed thinking framework, but they are not directly calculable.[3]

Our end goal is to estimate the cost-effectiveness of contributing a unit time and effort of a certain researcher or team to research on the technology t. So far we have a meta framework to estimate the impacts I brought by successful development of a technology t. And we introduce the notations in Table 1.

Notation	Meaning
r	An NLP researcher or research group
$\boldsymbol{T}(r)$	The set of NLP topics that the researcher can pursue (limited by skills, resources, and passion)
t	An NLP technology
$I(t)$	Social impacts brought by successful development of t
$\text{prog}(t)$	The current progress of t
$p(t; r)$	Probability that research in t succeeds based on the skills of the researcher r
$p(t; r)I(t)$	Expected social impact of the researcher r's work on t
$\Delta t(r)$	Improvement of t per unit resource (incl. time, effort, money, etc.) of the researcher r

Table 1: Notations and their corresponding meanings used for cost-effectiveness calculation.

For a researcher r, the action set per unit resource is $\{\Delta t | t \in \boldsymbol{T}(r)\}$. Equivalently speaking, they can intervene at a node t by the amount of $\Delta t(r)$ in the structured causal graph $\mathcal{G}$ in Figure 3.

The first useful concept is $p(t; r)I(t)$, **the expected social impact** of research on a technology t. Here the success rate $p(t; r)$ is crucial because most research does not necessarily produce the expected outcome. However, if the impact of a technology can be extremely large (for example, prevention of extinction has impact near positive infinity), then even with a very little success rate, we should still devote considerable efforts into it.

The second concept that is worth attention is **the marginal impact** (Pindyck et al., 1995) of one more unit of resources of the researcher r into the technology t, calculated as

$$\Delta I(t; r) := I(\text{prog}(t) + \Delta t(r)) - I(\text{prog}(t)) \ . \quad (3)$$

For example, if the field associated with the technology is almost saturated, or if many other researchers working on this field are highly competent, then, for a certain research group, blindly devoting time to the field may have little marginal impact. However, on the other hand, if a field is important but neglected, the marginal impact of pushing it forward can be large. This also explains why researchers are passionate about creating a new research field.

The third useful concept is **the opportunity cost** (Palmer and Raftery, 1999) to devote researcher r's resources into the technology t instead of a possibly more optimal technology $t^{\star}$. Formally,

---

[3] We adapted these terms from GP. Such terms to estimate priority has been successfully used by real-world social good organizations, e.g., GiveWell, Global Priorities Institute, the Open Philanthropy Project (a foundation with over 10 billion USD investment), ReThink Priorities, 80,000 Hours Organization. In the long run, the NLP community may potentially benefit from aligning with GP's terminology. Still, we do not recommend applying our framework in high-stake settings yet, since it serves only as a starting point currently.

the opportunity cost is calculated as

$$t^{\star}(r) := \arg\max_{x} \Delta I(x(r)), \quad (4)$$

$$\mathrm{Cost}(t; r) := \Delta I(t^{\star}(r); r) - \Delta I(t; r)\ , \quad (5)$$

where $t^{\star}$ is the optimal technology that can bring the largest expected improvement of social impact. The opportunity cost conveys the important message that we should not just do *good*, but do the *best*, because the difference from good to best can be a large loss.

Estimating the variables. Note that the frameworks we have proposed so far are at the meta level, useful for guiding thought experiments, and future research. Exact calculations are not possible with the current state of research in NLP for social good, although achievable in the future.

A practical insight is that NLP researchers estimate the impact of their research via qualitative explanations (natural language) or rough quantitative ones. For example, the introduction section of most NLP papers or funding proposals is a natural language-based estimation of the impact of the research. Such estimations can be useful to some extent (Hubbard and Drummond, 2011), although precise indicators of impact can motivate the work more strongly.

We can also borrow some criteria from effective altruism, a global movement that establishes a philosophical framework, and also statistical calculations of social good. One of the established metrics for calculating impact is called the "quality-adjusted life years" (QALYs) proposed by MacAskill (2015). QALYs count the number of life years (calibrated by life quality such as health conditions) that an act helps to increase.

5 Evaluating NLP tasks

In this section, we will first try to categorize the current state of NLP research for social good based on ACL 2020 papers, and then highlight NLP topics that are aligned with the UN's SDGs. We will conclude with a practical checklist and case studies of common NLP tasks using this checklist.

5.1 Current state of NLP research for social good – ACL 2020 as a case study

We want to compare the ideal priority list with the current distribution of NLP papers for social good. As a case study of the current research frontier, we plot the topic distribution of the 89 ACL 2020 papers that are related to NLP for social good in Figure 4. We also show the portion of papers by the 10 countries with the most social-good papers. Our annotation details are in Appendix A.

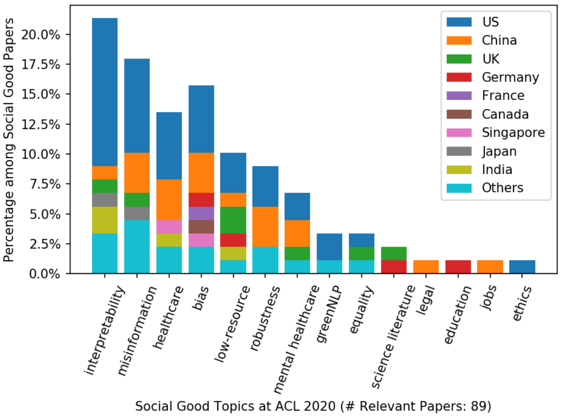

Figure 4: Social good topics at ACL 2020 by countries.

Illustrated in Figure 4, most social-good papers work on interpretability, tackling misinformation (e.g., fact-checking for news), and healthcare (e.g., to increase the capacity of doctors). In terms of countries, the US has the most papers on interpretability, and no papers on NLP for education, NLP for legal applications, and some other topics. China has few papers on interpretability, although interpretability is the largest topic. India has no papers on fighting misinformation, although it is the second largest topic. Only 5 countries have publications across more than two social good topics. Please refer to Appendix B for more analyses such as social-good papers by academia vs. industries.

However, compared with the UN's SDGs (United Nations, 2015), the current NLP research (at least in the scope of ACL conference submissions) lacks attention to other important cause areas such as tackling global hunger, extreme poverty, clean water and sanitation, and clean energy. There are also too few research papers on NLP for education, although education is the 4th most important area in SDGs.

One cause of this difference is *value misalignment*. Most NLP research is supported by stakeholders and funding agencies, which have a large impact on the current research trends or preferences in the NLP community. The perspective from social good with a framework to calculate the priority list has still not reached many in the NLP community.

Although we do not have data on expenditure

Cause	Annual Spending (USD)
Global R&D	1.5 trillion (UNESCO, 2017)
Luxury Goods	1.3 trillion (D'arpizio et al., 2015)
US Social Welfare	900 billion (Ferrara, 2011)
Climate Change	>300 billion (Buchner et al., 2014)
Global Poverty	>250 billion (Todd, 2017)
Nuclear Security	1-10 billion (Todd, 2017)
Pandemic Prevention	1 billion (Todd, 2017)
AI Safety Research	10 million (Todd, 2017)

Table 2: Annual spending of the cause areas.

Priority	**Example NLP research topics**
Poverty	• Predicting povery by geo-located Wikipedia articles (Sheehan et al., 2019) • Parsing fund applicant profiles (proposed)
Hunger	• NLP for agriculture (Yunpeng et al., 2019) • NLP for food allocation (proposed)
Health & Well-being	• NLP to analyze clinical notes (Dernoncourt et al., 2017a,b; Luo et al., 2018; Gopinath et al., 2020; Leiter et al., 2020a,b) • NLP for psychotherapy and counseling (Biester et al., 2020; Xu et al., 2020; Pérez-Rosas et al., 2019) • NLP for happiness (Asai et al., 2018; Evensen et al., 2019) • Assistive speech generation (proposed)
Education	• NLP for educational question answering (Atapattu et al., 2015; Lende and Raghuwanshi, 2016) • Improving textbooks (Agrawal et al., 2010) • Automated grading (Madnani and Cahill, 2018; Taghipour and Ng, 2016) • Plagiarism detection (Chong et al., 2010) • Tools for learners with disabilities (proposed)
Equality	• Interpretability (Köhn, 2015; Belinkov et al., 2017; Nie et al., 2020) • Ethics of NLP (Hovy and Spruit, 2016; Stanovsky et al., 2019; Sap et al., 2019) • NLP for low-resource languages (Zoph et al., 2016; Kim et al., 2017) • NLP on resource-limited devices (Sun et al., 2020) • NLP tools that signal bias in human language and speech (proposed)
Clean water	• Raising public awareness of water sanitation (proposed)
Clean energy	• Green NLP (Strubell et al., 2019; Schwartz et al., 2020) • NLP to analyze cultural values regarding climate change (Jiang et al., 2017; Koenecke and Feliu-Fabà, 2019) • Cross-cultural models of climate change perceptions (proposed)

Table 3: Top priorities and some NLP research related to each of them. This list may not be exhaustive. We also propose a high-impact research problem in each of the areas which has received less attention so far.

in each NLP subarea, we can get a glimpse of the value misalignment in general. Table 2 shows the annual spending of some cause areas. Note that the ranking of the expenditure does not align with our priority list for social good. For example, luxury goods are not as important as global poverty, but luxury goods cost 1.3 trillion USD each year, almost five times the expenditure in global poverty.

5.2 Aligning NLP with social good

In this subsection, we list the top priorities according to UN's SDGs (United Nations, 2015). For each goal, in Table 3 we include examples of existing NLP research, and suggest potential NLP tasks that can be developed (labeled as (proposed)).

5.3 Checklist

As a practical guide, we compile the takeaways of this paper into a list of heuristics that might be helpful for future practioners of NLP for social good. To inspect the social goodness of an NLP research direction (especially in Stage 3-4), the potential list of questions to answer is as follows:

(Q1) What kind of people/process will be benefited from the technology?

(Q2) Does it enforce traditional structure of beneficiaries? I.e., what groups of underprivileged people can be benefited? (e.g., by gender, demographics, socio-economic status, country, native languages, disability type)

(Q3) Does it contribute to SDG priority goals such as poverty, hunger, health, education, equality, clean water, and clean energy?

(Q4) Can it directly improve quality of lives? E.g., how many QALYs might it result in?

(Q5) Does it count as (a) mitigating problems brought by NLP, or (b) proactively helping out-of-NLP social problems?

5.4 Case studies by the checklist

We conduct some case studies of NLP technologies using the checklist.

Low-resource NLP & machine translation. This category includes NLP on low-resource languages, such as NLP for Filipino (Sagum et al., 2019; Cruz et al., 2020), and machine translation in general. Because this direction expands the users of NLP technologies from English-speaking people to other languages, it benefits people speaking these languages (Q1), and helps to narrow the gap between English-speaking and non-English speaking end users (Q2), although it is still likely that people who can afford intelligent devices will benefit more than those who cannot. This category can contribute directly to goals such as equality and education, and indirectly to other goals because translation of documents in general helps the sharing of information and knowledge (Q3). It directly improves quality of lives, for example, for immigrants who may have difficulties with the local language (Q4). Thus, it counts as social good category (b) by (Q5).

Transparency, interpretability, algorithmic fairness and bias. Research in this direction can impact users who need more reliable decision-making NLP, such as the selection process for loans, jobs, criminal judgements, and medical treatments (Q1). It can shorten the waiting time of candidates and still make fair decisions regardless of spurious correlations (Q2) (Q4). It reduces inequality raised by AI, but not increasing equality over man-made decisions, at least by the current technology (Q2). Thus, it is social good category (a) by (Q5).

Green NLP. Green NLP reduces the energy consumption of large-scale NLP models. Although it works towards the goal of affordable and clean energy (Q3), it just neutralizes the negative impact of training NLP models, but not impacting out-of-NLP energy problems. Green NLP belongs to social good category (a) by (Q5). It does not have large impacts directly targeted at (Q1), (Q2) and (Q4).

QA & dialog. People who can afford devices embedded with intelligent agents can use it, which is about 48.46% of the global population (BankMyCell, 2021) (Q1). So this benefits people with higher socio-economic status, and benefits English speaking people more than others, not to mention job replacements for labor-intensive service positions (Q2). It does not contribute to priority goals except for education and healthcare for people who can afford intelligent devices (Q3). Nonetheless, it can improve the quality of lives for its user group (Q4). It can be regarded as social good of category (b) by (Q5).

Information extraction, NLP-powered search engine & summarization. This direction speeds up the information compilation process, which can increase the productivity in many areas. About 50% of the world population have access to the Internet and thus can use it (Meeker, 2019) (Q1) (Q2). This category indirectly helps education, and the information compilation process of other goals (Q3). It can largely improve the lives of its user group because people gather information very frequently (e.g., do at least one Google search every day) (Q4). Thus, it belongs to social good category (b) by (Q5).

NLP for social media. This category is beneficial for social scientists interested in the trends and culture, or companies and organizations interested in market analyses (Q1) (Q2). The benefits to social science can indirectly help evidence-based policy makers, despite the caveat of bleach of user privacy, and information manipulation (Q4). This direction has limited contribution to priorities such as poverty and hunger, because extremely poor population do not own devices to post on social media (Q3). It is social good category (b) by (Q5).

6 Conclusion

This paper presented a meta framework to evaluate NLP tasks in the light of social good, and proposed a practical guide for practitioners in NLP. We call for more attention towards awareness and categorization of social impact of NLP research, and we envision future NLP research taking on an important social role and contributing to multiple priority areas. We also acknowledge the iterative nature of this emerging field, requiring continuing discussions, improvements to our thinking framework and different ways to implement it in practice. We highlight that the goal of our work is to take one step closer to a comprehensive understanding of social good rather than introducing a deterministic answer about how to maximize social good with NLP applications.

Acknowledgments

We thank Bernhard Schoelkopf, Kevin Jin, and Qipeng Guo for insightful discussions on the main ideas and methodology of the paper. We thank Osmond Wang for checking the economic concepts in the paper. We also thank Chris Brockett for checking many details in the paper. We thank the labmates in the LIT lab at University of Michigan, especially Laura Biester, Ian Stewart, Ashkan Kazemi, and Andrew Lee for constructive feedback. We also thank labmates at the MIT MEDG group, especially William Boag and Peter Szolovits for their constructive feedback. We thank Jingwei Ni for the help with the annotation of the country and affiliation of the ACL 2020 papers.

Ethical and societal implications

Our paper establishes a framework to better understand the definition of social good in the context of NLP research, and lays out a recommended direction on how to achieve it. The contributions of our paper could benefit a focused, organized and accountable development of NLP for social good. The data used in our work is public, and without privacy concerns.

References

Rakesh Agrawal, Sreenivas Gollapudi, Krishnaram Kenthapadi, Nitish Srivastava, and Raja Velu. 2010. Enriching textbooks through data mining. In *Proceedings of the First ACM Symposium on Computing for Development*, pages 1–9.

Firoj Alam, Fahim Dalvi, Shaden Shaar, Nadir Durrani, Hamdy Mubarak, Alex Nikolov, Giovanni Da San Martino, Ahmed Abdelali, Hassan Sajjad, Kareem Darwish, et al. 2020. Fighting the covid-19 infodemic in social media: a holistic perspective and a call to arms. *arXiv preprint arXiv:2007.07996*.

Akari Asai, Sara Evensen, Behzad Golshan, Alon Y. Halevy, Vivian Li, Andrei Lopatenko, Daniela Stepanov, Yoshihiko Suhara, Wang-Chiew Tan, and Yinzhan Xu. 2018. HappyDB: A corpus of 100, 000 crowdsourced happy moments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Association for Computational Linguistics. 2021. Ethics FAQ at ACL-IJCNLP 2021.

Thushari Atapattu, Katrina Falkner, and Nickolas Falkner. 2015. Educational question answering motivated by question-specific concept maps. In *International Conference on Artificial Intelligence in Education*, pages 13–22. Springer.

BankMyCell. 2021. How many smartphones are in the world?

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big. In *Proceedings of the 2021 Conference on Fairness, Accountability, and Transparency*.

Parminder Bhatia, Kristjan Arumae, Nima Pourdamghani, Suyog Deshpande, Ben Snively, Mona Mona, Colby Wise, George Price, Shyam Ramaswamy, and Taha Kass-Hout. 2020. AWS CORD19-search: A scientific literature search engine for COVID-19. *arXiv preprint arXiv:2007.09186*.

Laura Biester, Katie Matton, Janarthanan Rajendran, Emily Mower Provost, and Rada Mihalcea. 2020. Quantifying the effects of COVID-19 on mental health support forums. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

David Bourget and David J Chalmers. 2014. What do philosophers believe? *Philosophical studies*, 170(3):465–500.

John Broome. 2017. *Weighing goods: Equality, uncertainty and time*. John Wiley & Sons.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Barbara Buchner, Morgan Herve-Mignucci, Chiara Trabacchi, Jane Wilkinson, Martin Stadelmann, Rodney Boyd, Federico Mazza, Angela Falconer, and Valerio Micale. 2014. Global landscape of climate finance 2015. *Climate Policy Initiative*, 32.

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR.

Davide Castelvecchi. 2020. Prestigious ai meeting takes steps to improve ethics of research. *Nature*.

Irene Y Chen, Peter Szolovits, and Marzyeh Ghassemi. 2019. Can ai help reduce disparities in general medical and mental health care? *AMA journal of ethics*, 21(2):167–179.

Miranda Chong, Lucia Specia, and Ruslan Mitkov. 2010. Using natural language processing for automatic detection of plagiarism. In *Proceedings of the 4th International Plagiarism Conference (IPC-2010)*.

Hyunji Chung, Michaela Iorga, Jeffrey Voas, and Sangjin Lee. 2017. Alexa, can i trust you? *Computer*, 50(9):100–104.

Jan Christian Blaise Cruz, Jose Kristian Resabal, James Lin, Dan John Velasco, and Charibeth Cheng. 2020. Investigating the true performance of transformers in low-resource languages: A case study in automatic corpus creation. *CoRR*, abs/2010.11574.

Claudia D'arpizio, Federica Levato, Daniele Zito, and Joelle de Montgolfier. 2015. Luxury goods worldwide market study. *Bain & Company's report*.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2017a. NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 97–102. Association for Computational Linguistics.

Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. 2017b. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 24(3):596–606.

Sara Evensen, Yoshihiko Suhara, Alon Y. Halevy, Vivian Li, Wang-Chiew Tan, and Saran Mumick. 2019. Happiness entailment: Automating suggestions for well-being. In *8th International Conference on Affective Computing and Intelligent Interaction, ACII 2019, Cambridge, United Kingdom, September 3-6, 2019*, pages 62–68. IEEE.

Peter Ferrara. 2011. America's ever expanding welfare empire. *Forbes, April*, 22.

Anjalie Field, Shrimai Prabhumoye, Maarten Sap, Zhijing Jin, Jieyu Zhao, and Chris Brockett. 2020. 1st workshop on nlp for positive impact.

Elizabeth Gibney. 2020. The battle for ethical ai at the world's biggest machine-learning conference. *Nature*, 577(7792):609–609.

Divya Gopinath, Monica Agrawal, Luke Murray, Steven Horng, David Karger, and David Sontag. 2020. Fast, structured clinical documentation via contextual autocomplete. In *Machine Learning for Healthcare Conference*, pages 842–870. PMLR.

Hilary Greaves and William McAskill. 2017. A research agenda for the Global Priorities Institute. *University of Oxford, London*.

Tamanna Hossain, Robert L. Logan IV, Arjuna Ugarte, Yoshitomo Matsubara, Sean Young, and Sameer Singh. 2020. COVIDLies: Detecting COVID-19 misinformation on social media. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Dirk Hovy, Shannon Spruit, Margaret Mitchell, Emily M. Bender, Michael Strube, and Hanna Wallach, editors. 2017. *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*. Association for Computational Linguistics, Valencia, Spain.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598.

Douglas W Hubbard and David Drummond. 2011. *How to measure anything*. Wiley Online Library.

Ye Jiang, Xingyi Song, Jackie Harrison, Shaun Quegan, and Diana Maynard. 2017. Comparing attitudes to climate change in the media using sentiment analysis based on latent dirichlet allocation. In *Proceedings of the 2017 Workshop: Natural Language Processing meets Journalism, NLPmJ@EMNLP, Copenhagen, Denmark, September 7, 2017*, pages 25–30. Association for Computational Linguistics.

Anna Jobin, Marcello Ienca, and Effy Vayena. 2019. The global landscape of ai ethics guidelines. *Nature Machine Intelligence*, 1(9):389–399.

Shelly Kagan. 2018. *Normative ethics*. Routledge.

Jackson A Killian, Bryan Wilder, Amit Sharma, Vinod Choudhary, Bistra Dilkina, and Milind Tambe. 2019. Learning to prescribe interventions for tuberculosis patients using digital adherence data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2430–2438.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 2832–2838.

Allison Koenecke and Jordi Feliu-Fabà. 2019. Learning twitter user sentiments on climate change with limited labeled data. *CoRR*, abs/1904.07342.

Arne Köhn. 2015. What's in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073.

Thomas S Kuhn. 2012. *The structure of scientific revolutions*. University of Chicago press.

Vasileios Lampos, Simon Moura, Elad Yom-Tov, Ingemar J Cox, Rachel McKendry, and Michael Edelstein. 2020. Tracking COVID-19 using online search. *arXiv preprint arXiv:2003.08086*.

Jochen L Leidner and Vassilis Plachouras. 2017. Ethical by design: Ethics best practices for natural language processing. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 30–40.

Richard Leiter, Enrico Santus, Zhijing Jin, Katherine Lee, Miryam Yusufov, Edward Moseley, Yujie Qian, Jiang Guo, and Charlotta Lindvall. 2020a. An artificial intelligence algorithm to identify documented symptoms in patients with heart failure who received cardiac resynchronization therapy (s717). *Journal of Pain and Symptom Management*, 59(2):537–538.

Richard E Leiter, Enrico Santus, Zhijing Jin, Katherine C Lee, Miryam Yusufov, Isabel Chien, Ashwin Ramaswamy, Edward T Moseley, Yujie Qian, Deborah Schrag, et al. 2020b. Deep natural language processing to identify symptom documentation in clinical notes for patients with heart failure undergoing cardiac resynchronization therapy. *Journal of Pain and Symptom Management*, 60(5):948–958.

Sweta P Lende and MM Raghuwanshi. 2016. Question answering system on education acts using nlp techniques. In *2016 world conference on futuristic trends in research and innovation for social welfare (Startup Conclave)*, pages 1–6. IEEE.

Margaux Luck, Jonnie Penn, Tristan Sylvain, Mark Crowley, Joseph Paul Cohen, Margaux Luck, Sean McGregor, Myriam Côté, Valentine Goddard, Margaux Luck, Kenny Chen, Arsene Fansi Tchango, Joseph Paul Cohen, and Yoshua Bengio. 2018. Ai for social good workshop.

Yuan Luo, Yu Cheng, Özlem Uzuner, Peter Szolovits, and Justin Starren. 2018. Segment convolutional neural networks (seg-cnns) for classifying relations in clinical notes. *Journal of the American Medical Informatics Association*, 25(1):93–98.

Michael MacAskill, Krister Bykvist, and Toby Ord. 2020. *Moral uncertainty*. Oxford University Press.

William MacAskill. 2015. *Doing good better: Effective altruism and a radical new way to make a difference*. Guardian Faber Publishing.

Nitin Madnani and Aoife Cahill. 2018. Automated scoring: Beyond natural language processing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1099–1109.

Mary Meeker. 2019. Internet trends report; 2019.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Stephen Palmer and James Raftery. 1999. Opportunity cost. *Bmj*, 318(7197):1551–1552.

Judea Pearl. 1995. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

Verónica Pérez-Rosas, Xinyi Wu, Kenneth Resnicow, and Rada Mihalcea. 2019. What makes a good counselor? learning to distinguish between high-quality and low-quality counseling conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 926–935, Florence, Italy. Association for Computational Linguistics.

Robert S Pindyck, Daniel L Rubinfeld, and Prem L Mehta. 1995. *Microeconomics*, volume 4. Prentice Hall Englewood Cliffs, NJ.

Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. 2020. Closing the ai accountability gap: defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 33–44.

Ria Ambrocio Sagum, Aldrin D Ramos, and Monique T Llanes. 2019. Ficobu: Filipino wordnet construction using decision tree and language modeling. *International Journal of Machine Learning and Computing*, 9(1):103–107.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63.

Saqib Shah and Julian Chokkattu. 2016. Microsoft kills AI chatbot Tay (twice) after it goes full Nazi.

Evan Sheehan, Chenlin Meng, Matthew Tan, Burak Uzkent, Neal Jean, Marshall Burke, David B. Lobell, and Stefano Ermon. 2019. Predicting economic development using geolocated wikipedia articles. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2698–2706. ACM.

Henry Sidgwick. 1874. *The methods of ethics*. Macmillan and Co.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Dan Su, Yan Xu, Tiezheng Yu, Farhad Bin Siddique, Elham Barezi, and Pascale Fung. 2020. CAiRE-COVID: A question answering and query-focused multi-document summarization system for COVID-19 scholarly information management. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2158–2170. Association for Computational Linguistics.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1882–1891.

Judith Jarvis Thomson. 1976. Killing, letting die, and the trolley problem. *The Monist*, 59(2):204–217.

Benjamin Todd. 2017. The case for reducing existential risks.

Nenad Tomašev, Julien Cornebise, Frank Hutter, Shakir Mohamed, Angela Picciariello, Bec Connelly, Danielle CM Belgrave, Daphne Ezer, Fanny Cachat van der Haert, Frank Mugisha, et al. 2020. Ai for social good: unlocking the opportunity for positive impact. *Nature Communications*, 11(1):1–6.

Hilary Townsend. 2013. Natural language processing and clinical outcomes: the promise and progress of nlp for improved care. *Journal of AHIMA*, 84(2):44–45.

UNESCO. 2017. Facts and figures: R&d expenditure.

United Nations. 2015. UN Sustainable Development Goals.

Voicebot. 2020. Amazon Echo & Alexa stats.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, et al. 2020. Cord-19: The COVID-19 open research dataset. *ArXiv*.

Charles Welch, Allison Lahnala, Veronica Perez-Rosas, Siqi Shen, Sarah Seraj, Larry An, Kenneth Resnicow, James Pennebaker, and Rada Mihalcea. 2020. Expressive interviewing: A conversational system for coping with COVID-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Zhentao Xu, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Inferring social media users' mental health status from multimodal information. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6292–6299, Marseille, France. European Language Resources Association.

Cui Yunpeng, Wang Jian, and Liu Juan. 2019. The development of deep learning based natural language processing (nlp) technology and applications in agriculture. *Journal of Agricultural Big Data*, 1(1):38.

Yi Zeng, Enmeng Lu, and Cunqing Huangfu. 2019. Linking artificial intelligence principles. In *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*, volume 2301 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A ACL 2020 paper annotations

For the case study on ACL 2020 papers, such as Figure 2 and 4, we collect the 570 long papers at ACL 2020. An NLP researcher with four years of research experience conducted the entire annotation, so that the categorization is consistent across all papers.[4]

The first annotation task is to categorize all papers into one of the four stages in the theory→application development. We showed the annotator the description of the four stages in Section 3.1. Next, provided with the title, abstract, and PDF of each paper, the annotator was asked to annotate which of the four stages each paper belongs to. The annotator had passed a test batch before starting the large-scale annotation.

The second annotation task is to annotate the research topics of the papers related to social good at ACL 2020. If the paper has a clear social good impact (89 out of 570 papers), the annotator needs to classify the topic of the paper into one of the given categories: bias mitigation, education, equality, fighting misinformation, green NLP, healthcare, interpretability, legal applications, low-resource language, mental healthcare, robustness, science literature parsing, and others. For the other meta information such as countries, or academia vs. industry, we decide based on the information of the leading first author.

B More statistics about ACL 2020 papers

For the case study on ACL 2020 papers, we further investigate the following statistics.

Stage 1-4 by countries. Recall that in Figure 2 of the main paper, we plot the distributions of papers by the four stages, and highlight the most frequent topics in each stage. Additionally, it is also interesting to explore the distribution of stages for different countries. In Figure 5, we have the following observations:

China does not have Stage-1 papers (i.e., fundamental theories), although it has the second largest total number of papers. The reason might be that there are not many Chinese researchers on linguistic theories who publish at English conferences.

Most countries' number of papers in the four stages follows the overall trend (i.e., Stage-2 papers > Stage-3 papers > Stage-4 papers > Stage-1 papers), with a few exceptions. For example, China has almost the same number of papers in Stage 2 and 3, Germany has more papers in Stage 4 (i.e., deployed applications) than in Stage 3, and Canada has the most papers in Stage 3.

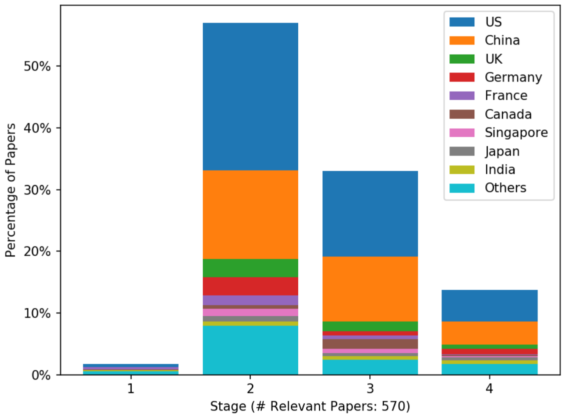

Figure 5: Stage 1-4 of ACL 2020 papers by countries.

Social good topics by academia vs. industry. As we call for more research attention to NLP for social good, it is important to understand the affiliations behind the current social good papers. A coarse way is to look at the affiliation of the first author, and inspect whether the main work of the paper is done by people from academia or industry.

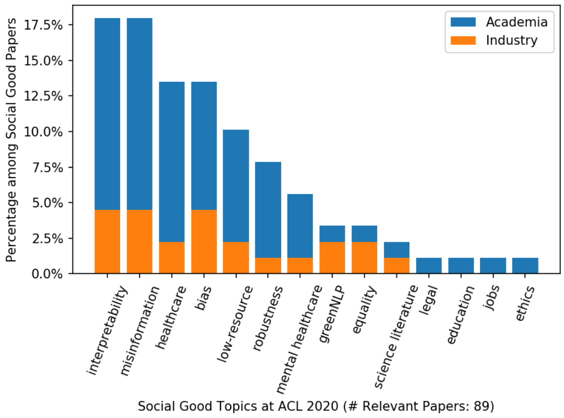

Figure 6: Social good topics at ACL 2020 by affiliations.

As in Figure 6, overall academia publishes several times more papers on social good than the industry. This ratio is higher than the average ratio of papers from academia out of all ACL 2020 papers (389 from academia out of 570). Industry does not have ACL 2020 papers on topics such as

---

[4] The annotation file has been uploaded to the softconf system.

NLP ethics. Note that using statistics from ACL papers alone could be limiting because researchers in academia typically present almost all research achievements through publications, but many industry researchers do not publish in public venues such as ACL, although their research may impact various products.

IgSEG: Image-guided Story Ending Generation

Qingbao Huang[1,2,6], **Chuan Huang**[1], **Linzhang Mo**[1]
Jielong Wei[1], **Yi Cai**[2,5]*, **Ho-fung Leung**[3], **Qing Li**[4]
[1]School of Electrical Engineering, Guangxi University, Nanning, Guangxi, China
[2]School of Software Engineering, South China University of Technology, Guangzhou, China
[3]Dept. of Computer Sc. & Engin, The Chinese University of Hong Kong, Hong Kong, China
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[5]Key Laboratory of Big Data and Intelligent Robot (SCUT), MOE of China
[6]Institute of Artificial Intelligence, Guangxi University, Nanning Guangxi, China
`qbhuang@gxu.edu.cn, chuanhuangqy@gmail.com, mlz1997@163.com`

Abstract

In this work, we propose a new task called Image-guided Story Ending Generation (IgSEG). Given a multi-sentence story plot and an ending-related image, IgSEG aims to generate a story ending that conforms to the contextual logic and the relevant visual concepts. In contrast to the story ending generation task, which generates open-ended endings, the major challenges of IgSEG are to comprehend the given context and image sufficiently, and mine the appropriate semantics from the image to make the generated story ending informative, reasonable, and coherent. To address the challenges, we propose a Multi-layer Graph convolution and Cascade-LSTM (MGCL) based model which mainly comprises of two collaborative modules: i) a multi-layer graph convolutional network to learn the dependency relations of sentences and the logical clue of the context; ii) a multiple context-image attention module to generate the story endings by gradually incorporating textual and visual semantic concepts. Our MGCL is thus capable of building logically consistent and semantically rich story endings. To evaluate the proposed model, we modify the existing VIST dataset to obtain the VIST-Ending dataset. Empirically, our MGCL outperforms all the strong baselines on both automatic and human evaluation.

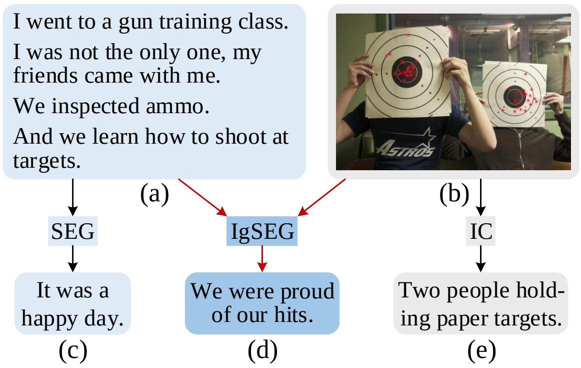

Figure 1: In SEG, existing methods tend to generate generic, safe, and inane story endings, e.g., (c). IgSEG is designed to generate specific, reasonable and informative endings induced by the given ending-related image. (d) is generated by the proposed MGCL model.

1 Introduction

As two challenging subtasks of story generation, the story ending generation (SEG) and visual storytelling (Huang et al., 2016; Zhao et al., 2018) have attracted more attention recently. The former generates text-based story endings (Zhao et al., 2018; Li et al., 2018; Guan et al., 2019). While, the latter generates photo-streams-based stories (Huang et al., 2016; Wang et al., 2018; Hu et al., 2020) or one-image-based stories (Gaur, 2019). Distinctly, both of them are input with single-modal information merely. Actually, people often confront with demands to handle multi-modal inputs for generating a sentence or paragraph, e.g., *comments generation given a news story and an image* and *picture composition with a leading paragraph*. However, to our best knowledge, the SEG task incorporating a context and an image is still under-explored.

Furthermore, due to the limited textual information of the story context, the generated endings of SEG models remain tending to be generic, safe, and inane. To make the generation of story endings more coherent, specific, and informative, we consider introducing visual information to enrich the generation of story endings. For example (cf. Figure 1), the story context (a) mainly narrates that the experience of someone went for gun training. The story ending generated by SEG (c) just talks about the feeling (e.g., *happy*) of the day, which seems to be generic, safe, and unattractive for lack of interesting events, imaginative conception, and evocative plots. Meanwhile, Image Captioning (e) generates the description of a given image (b) with-

---

*Corresponding author: Cai Yi (ycai@scut.edu.cn).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3114–3123
August 1–6, 2021. ©2021 Association for Computational Linguistics

out any story context plot. Here, we introduce the image to induce the development of the story plot and guide the generation of the ending. The image-guided story ending (d) is associated with the senior semantic (e.g., *proud*) and events (e.g., *hits*) from the visual information. Obviously, this ending seems to be high-quality compared with the one generated by SEG.

We herein propose an Image-guided Story Ending Generation (IgSEG) task, which aims at generating a story ending with contextual plots and an ending-related image. Models need to comprehend the story plots and the image information, and grasp the visual semantic concepts strongly related to the story plots (e.g., *event*, *behavior*, and *emotion*). The main challenges of this task are three-fold: *(i)* How to accurately select and capture appropriate visual concepts matching the development trend of the story plot from the image. *(ii)* How to fuse the language and visual information and model inter- and intra-modality relations efficiently. *(iii)* How to make the utmost of high-level semantics mined from the image to write coherent, semantically-informative, and imaginative story endings.

To capture the text contextual plots and merge visual features effectively, we propose a Multi-layer Graph convolution and Cascade-LSTM (MGCL) model. A multi-layer graph convolution module is constructed to capture and encode the clues information (e.g., dependency relations (Zhang et al., 2018)) hidden in context. In detail, following (Huang et al., 2021), for each sentence, we construct a graph over the dependency parsing tree and conduct convolutional operations by Graph Convolution Networks (GCN). We then employ attention mechanism to compress each graph as one node and deliver the node from low layer to high layer for aggregation of inter-sentence information. Furthermore, inspired by the work (Anderson et al., 2018), we design a Multiple Context-Image Attention (MCIA) module to merge the contextual features and the image features. Specifically, we apply attention mechanism to weight sentence features and image features separately, then concatenate and feed them to the next Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) cell. We perform experiments on the VIST-Ending (VIST-E) dataset which is modified from the VIST (Huang et al., 2016).

Our contributions can be summarized as follows:

- We define a new task termed IgSEG to generate coherent, specific, and informative story endings guided by an ending-related image. To our best knowledge, this is the first story generation task with multi-modal inputs.

- We propose a model called MGCL, which employs multi-layer graph convolutional operations to capture story contextual plots and multiple context-image attentions to merge visual features effectively.

- Experiments show that our model outperforms several strong baselines on the VIST-E dataset. Human evaluations show that our model can generate story endings with better grammaticality, logicality, and relevance.

2 Related Work

The IgSEG task is related to *(i)* Story Ending Generation (SEG) and *(ii)* Visual Storytelling (VIST). **SEG** (Zhao et al., 2018) is a subtask of story generation, which aims to understand the context and generate a coherent story ending. Many researchers have made great efforts on SEG. To improve the diversity and rationality of the generated story endings, (Li et al., 2018) tried to employ a Seq2Seq model based on adversarial training. Similarly, (Guan et al., 2019) made the model generate a reasonable ending by introducing external commonsense knowledge. Further, (Wang and Wan, 2019) adopted a transformer-based conditional autoencoder to capture contextual clues to improve coherence of story endings. (Guan et al., 2020) proposed a knowledge-enhanced pretraining approach for generating more reasonable stories. (Huang et al., 2021) proposed a multi-level GCN to capture the dependency relations of input sentences. Although previous studies have made great progress, due to the limitations of the SEG task itself, the generated endings tend to be generic and safe to some extent.

IgSEG is relevant to **VIST** as well. VIST aims to generate a coherent story according to an image stream. The main difficulty of VIST is how to generate image-relevant sentences. The previous works on VIST can be roughly divided into three categories. The first category focuses on designing specific model architectures to improve the quality of the generated stories (Kim et al., 2018; Wang et al., 2019). The second one generate more

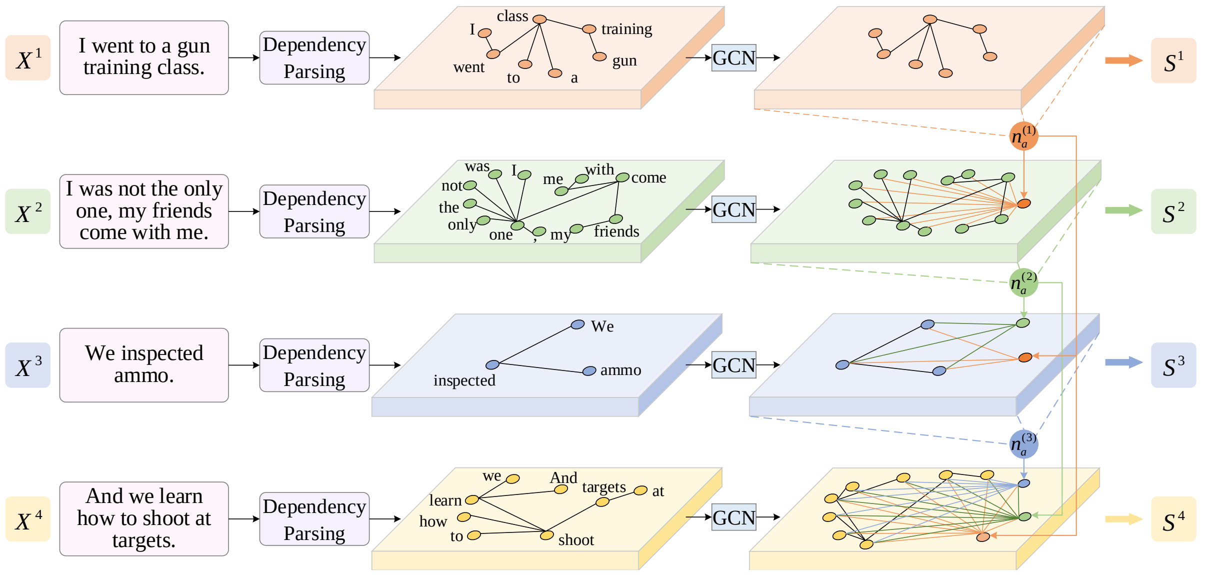

Figure 2: The Multi-layer Graph Convolutional Networks structure for context encoding. For intra-sentence information, each sentence is constructed as a graph over dependency parsing. For inter-sentence information, each graph is compressed as one node and is delivered to next layer. The output representation of each graph (i.e., S^1, S^2, S^3, S^4) are feed into decoder.

expressive output with reinforcement learning and adversarial training (Wang et al., 2018; Huang et al., 2019; Mo et al., 2019; Hu et al., 2020). The third one generates more common-sense stories by incorporating external knowledge. (Yang et al., 2019; Li et al., 2019; Wang et al., 2020; Jung et al., 2020).

However, the inputs of both SEG and VIST are single-modal information. The work on generating story endings given a textual sequence and an image simultaneously is unexplored. Therefore we propose the IgSEG task.

3 Methodology

3.1 Overview

The proposed IgSEG task aims to generate a story ending conforming the given contextual and visual information. Given a story context $X = \{X^1, X^2, \cdots, X^\mu\}$ and an ending-related image V, where $X^\mu = x_1^\mu x_2^\mu \cdots x_c^\mu$ represents the μ-th sentence with c words, IgSEG aims at generating a story ending $E = y_1 y_2 \cdots y_m$ with m words.

To generate the contextual-consistent and image-related story endings, we propose a Multi-layer Graph convolutional networks and Cascade-LSTM (MGCL) model based on the encoder-decoder framework. In the encoder, we propose a Multi-layer Graph Convolutional Networks (MGCN) over dependency trees to learn the context representation (cf. Figure 2), and we extract the image features with ResNet-152 (He et al., 2016). When decoding, we generate the story ending with the cascaded LSTM framework. Specifically, we employ Top-Down LSTM to joint the context features and image features. And we devise a Multiple Context-Image Attention (MCIA) module to grasp the image-related context and contextual-relevant information of image for text generation. We will introduce each part of MGCL below.

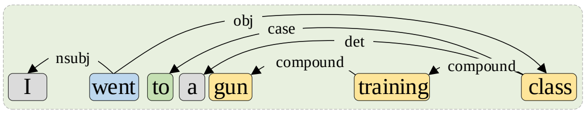

Figure 3: Dependency parsing of a sentence.

3.2 Story Context Representation

Graph Construction We parse the sentences with Stanford Dependency tool (De Marneffe et al., 2014) (cf. Figure 3). To capture the relations of words in a sentence, we construct a graph G over the dependency parsing tree for each sentence. Regarding the words x as nodes $\mathcal{O}^k$, the word representation n_i as node feature, and the corresponding relations on the dependency parsing tree as edges ξ^k, the graph G^k of k-th sentence (k=1,2,3,4) can be constructed:

$$G^k = (\mathcal{O}^k, \xi^k). \tag{1}$$

Multi-Layer GCN To deliver inter-sentence information, we utilize attention mechanism to weight

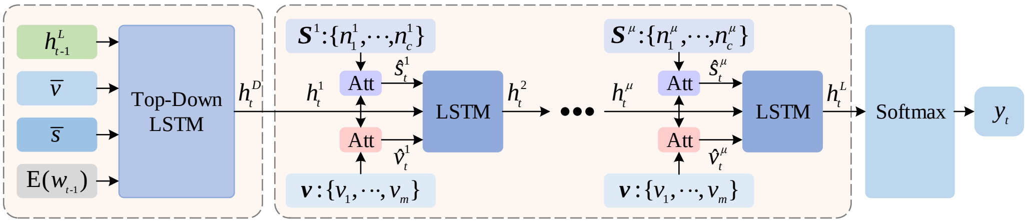

(a) TD (b) MCIA

Figure 4: Illustration of the Cascade-LSTM framework. The Cascade-LSTM consists of TD and MCIA modules. The MCIA module consists of μ LSTM. The $\bar{v}$ denotes the mean-pooled image features, the $\bar{s}$ denotes the mean-pooled context features. The symbol "$\cdots$" denotes the omitted parts of MCIA module.

each node and sum them together as a new node $n_a^{(k)}$ for the $(k+1)$-th layer GCN (cf. Figure 2):

$$\boldsymbol{S}^k = [n_1^k \cdots n_c^k], \tag{2}$$

$$\theta = softmax(W_0^k \boldsymbol{S}^k + b_0^k), \tag{3}$$

$$n_a^k = \sum_{i=1}^{n} \theta n_i^k, \tag{4}$$

where n_i^k denotes the features of the i-th word of the k-th sentence, W_0 and b_0 are trainable parameters.

After updating the nodes of $(k+1)$-th layer GCN, the graph G^{k+1} structure is represented by a $(\lambda+k)\times(\lambda+k)$ adjacency matrix $\boldsymbol{A}^{k+1} = \{A_{ij}, (i,j) \in (\lambda+k)\}$. The corresponding value A_{ij} is 1 if the relation exists between node i and node j, otherwise it is 0. The representations of node i and its neighbor node $j \in \phi(i)$ are n_i^{k+1} and n_j^{k+1}, respectively. To obtain the correlation score w_{ij}^{k+1} between node i and node j, we learn a connected layer over concatenation of nodes features:

$$w_{ij}^{k+1} = w_{k+1}^T \sigma\left(W_1^{k+1}[n_i^{k+1}; n_j^{k+1}] + b_1^{k+1}\right), \tag{5}$$

where w_{k+1}, W_1^{k+1}, and b_1^{k+1} are trainable parameters, σ is the non-linear activation function, $(\cdot)^T$ denotes transpose operation, and $[;]$ denotes the concatenation operation.

We apply the softmax function over the correlation score w_{ij} to obtain the weight α_{ij}:

$$\alpha_{ij}^{k+1} = \frac{\exp(w_{ij}^{k+1})}{\sum_{j\in\phi(i)} \exp(w_{ij}^{k+1})}. \tag{6}$$

In the adjacency matrix $\boldsymbol{A}^{k+1}$, the value is α_{ij}^{k+1} if the relation exists between node i and node j, otherwise is 0. The A_{ij}^{k+1} can be denoted as:

$$A_{ij}^{k+1} = \begin{cases} \alpha_{ij}^{k+1} & \text{nodes } i,j \text{ are related} \\ 0 & \text{nodes } i,j \text{ are unrelated} \end{cases}. \tag{7}$$

For each node of the $(k+1)$-th GCN layer, we update the $(h+1)$-th representation of node n_i^{h+1} with aggregating the representations of h-th neighboring nodes n_j^h. This procedure is denoted as:

$$n_i^{h+1} = \sigma(A_{ii}n_i^h + \sum_{j\in\phi(i)} A_{ij}(W_2^h n_j^h + b_2^h)), \tag{8}$$

where W_2^h and b_2^h are trainable parameters. By l updates, the output S^{k+1} of GCN is denoted as:

$$\boldsymbol{S}^{k+1} = [n_1^{k+1} \cdots n_c^{k+1}]. \tag{9}$$

3.3 Decoder

The inputs of the decoder are the context features $\boldsymbol{S} = \left\{\boldsymbol{S}^k\right\}_{k=1}^4$ and the image features $\boldsymbol{v}$ extracted with the pre-trained model ResNet152 (He et al., 2016), as shown in Figure 4.

Top-Down LSTM (TD) Following previous work (Anderson et al., 2018), we employ Top-Down LSTM to incorporate the visual information (cf. Figure 4(a)). We operate LSTM over a single time step in the decoder with the following notation:

$$h_t = \mathrm{LSTM}(x_t, h_{t-1}) \tag{10}$$

where x_t is the input vector of LSTM and h_t is the output vector. The inputs of TD module x_t^D consists of the previous output h_{t-1}^L of MCIA module, the mean-pooled image features $\bar{v}$, and the embedding of the previously generated word $E(w_{t-1})$, where t denotes the current time step.

To incorporate the context information, we modify the original inputs of TD. Firstly, we calculate the mean-pooled context features $\bar{s}$:

$$\bar{s} = \frac{1}{\mu}\frac{1}{c}\sum_{k=1}^{\mu}\sum_{i=1}^{c} n_i^k, \tag{11}$$

where μ denotes the number of sentences, c denotes the number of words of each sentence, n_i^k denotes the representation of the i-th word of the k-th sentence. Then, the vector x_t^D is denoted as:

$$x_t^D = [h_{t-1}^L; \bar{s}; \bar{v}; E(w_{t-1})]. \tag{12}$$

Multiple Context-Image Attention (MCIA) To merge the context features and image features, we devise the MCIA module. The MCIA module consists of four LSTM layers (cf. Figure 4(b)), which share all the parameters. The output h_t^D of TD is input to the MCIA module. Given the output h_t^k of $(k-1)$-th LSTM layer in MCIA module ($h_t^1 = h_t^D$), at each time step, we calculate the normalized attention weight $a_{i,t}^k$ for each of word representations n_i^k of the k-th sentence:

$$a_{i,t}^k = (w_a^k)^T \tanh(W_a^k n_i^k + W_a^{hk} h_t^k), \tag{13}$$

where w_a^k, W_a^k, and W_a^{hk} are trainable parameters. The convex combination of all words $\hat{s}_t^k$ can be calculated by n_i^k:

$$\beta_{i,t}^k = softmax(\boldsymbol{a}_t^k), \tag{14}$$

$$\hat{s}_t^k = \sum_{i=1}^{c} \beta_{i,t}^k n_i^k. \tag{15}$$

Likewise, we calculate the normalized weight $b_{i,t}^k$ for features v_i of each region of the image:

$$b_{i,t}^k = (w_b^k)^T \tanh(W_b^k v_i + W_b^{hk} h_t^k), \tag{16}$$

where w_b^k, W_b^k, and W_b^{hk} are trainable parameters. The convex combination of the image $\hat{v}_t^k$ can be calculated by the image features v_i:

$$\gamma_{i,t}^k = softmax(\boldsymbol{b}_t^k), \tag{17}$$

$$\hat{v}_t^k = \sum_{i=1}^{M} \gamma_{i,t}^k v_i, \tag{18}$$

where M denotes the the number of region of the image. We concatenate $\hat{s}_t^k$, $\hat{v}_t^k$, and h_t^k as inputs of the next LSTM layer:

$$x_t^{k+1} = [\hat{s}_t^k; h_t^k; \hat{v}_t^k]. \tag{19}$$

Given the output h_t^L of MCIA module, we calculate the conditional distribution over possible output words at each time step as follows:

$$p(y_t|y_{1:t-1}) = softmax(W_p h_t^L + b_p), \tag{20}$$

where W_p and b_p trainable parameters, and $y_{1:m}$ is the notation to refer to a sequence of words $(y_1, \cdots, y_m)$. Finally, the product of conditional distributions can be obtained by:

$$p(y_{1:m}) = \prod_{t=1}^{T} p(y_t|y_{1:t-1}). \tag{21}$$

4 Experiments

4.1 Dataset

To serve the IgSEG task, we modify the VIST dataset (Huang et al., 2016) to obtain a VIST-Ending (VIST-E) dataset, as shown in Table 1. Specifically, we keep the first four sentences, the ending sentence, and the last image of the photo stream of the VIST dataset. We have removed the stories which have corrupted images and rigmarole sentences over 40 words.

Dataset	Total	Training	Validation	Test
VIST	50,200	40,155	4,990	5,055
VIST-E	49,913	39,920	4,963	5,030

Table 1: Statistics of VIST and VIST-E.

4.2 Baselines

We compare our model with following models. **Seq2Seq** is a stack RNN-based model (Luong et al., 2015) with attention mechanisms. **Transformer** is a parallel model based solely on attention mechanisms (Vaswani et al., 2017). **IE+MSA** incorporates external knowledge with incremental encoding model for story ending generation(Guan et al., 2019). **T-CVAE** is a transformer-based conditional variational autoencoder for missing story plots generation (Wang and Wan, 2019). To adapt the IgSEG task, we rebuild above baselines by concatenating visual features as inputs. For fair comparison and testing our model, two variants of MGCL are created with the same inputs of the baselines. **MG+CIA**: We keep one Context-Image-Attention (CIA) unit in the decoder of MGCL. **MG+Trans**: We replace the decoder of MGCL with Transformer.

4.3 Evaluation Metrics

Automatic Evaluation We adopt four automatic metrics: **BLEU (B)** (Papineni et al., 2002) evaluates n-gram overlap between generated ending and a reference. **METEOR (M)** (Banerjee and Lavie, 2005) evaluates a generated sentence with

Model	B1	B2	B3	B4	M	C	R-L	Gram.	Logic.	Rele.
Seq2Seq†(Luong et al., 2015)	13.96	5.57	2.94	1.69	4.54	12.04	16.84	1.59	1.61	1.65
Transformer†(Vaswani et al., 2017)	17.18	6.29	3.07	2.01	6.91	12.75	18.23	3.01	2.15	1.96
IE+MSA†(Guan et al., 2019)	19.15	5.74	2.73	1.63	6.59	15.56	20.62	3.41	2.09	1.52
T-CVAE†(Wang and Wan, 2019)	14.34	5.06	2.01	1.13	4.23	11.49	15.51	1.89	1.76	1.25
MG+Trans† (ours)	19.43	7.47	3.92	2.46	7.63	14.42	19.62	3.46	2.77	2.60
MG+CIA† (ours)	20.91	7.46	3.88	2.35	7.29	19.88	21.12	2.80	2.35	1.97
MGCL (our full model)	**22.57**	**8.16**	**4.23**	**2.49**	**7.84**	**21.46**	**21.66**	**3.51**	**3.17**	**2.75**

Table 2: Experiments on the VIST-E dataset for the IgSEG task (p-value < 0.01). The **bold** / underline denotes the best and the second performance, respectively. † denotes the image features are directly concatenated.

Model	B2	B4	M	C	R-L
MGCL	**8.16**	**2.49**	**7.84**	**21.46**	**21.66**
w/o MGCN	7.11	2.03	7.23	20.26	19.74
w/o TD	5.18	1.04	6.74	10.33	18.93
w/o MCIA	7.17	2.17	7.13	17.62	19.75
w/o TD, MCIA	3.96	0.77	5.49	8.66	17.64

Table 3: Ablation studies. “w/o” means “without”.

direct word-ordering. **CIDEr (C)** (Vedantam et al., 2015) evaluates the similarity of a generated sentence against the references by human consensus. **ROUGE-L (R-L)** (Lin, 2004) is applied to find the length of the longest common subsequence.

Human Evaluation Considering the limitation of automatic evaluation and the complexity of the IgSEG task, it is necessary to conduct human evaluation. The criteria of human evaluation includes three aspects: **Grammaticality (Gram.)** (Wang and Wan, 2019) evaluates correct, natural, and fluent of the generated story endings. **Logicality (Logic.)** (Wang and Wan, 2019) evaluates whether the story endings are reasonable and coherent. **Relevance (Rele.)** (Yang et al., 2019) measures how relevant the generated story endings and the input images are. We randomly pick 100 generated story endings from test-set for each model and employ three professional annotators skills to make evaluation. Following (Yang et al., 2019), we apply a 5-grade marking system, with 5 as the maximum grade and 1 as the worst. The final results are the average of the scores given by the three annotators.

4.4 Experimental Settings

The dimension of word embedding is 300 from GloVe.6B (Pennington et al., 2014). The update times of each GCN is 5, the maximum number of nodes in GCN is 43. The hidden layer dimension of all LSTM is 512. The number of LSTM layer is 4 in MCIA module. The dimension of image features is $7 \times 7 \times 2048$ from ResNet-152 (He et al., 2016). During training on the VIST-E dataset, the epoch is set to 30 and the batch size is 128. The optimizer is Adam with an initial learning rate of 4e-4. All baselines keep their own default settings. The dropout rate is 0.5. Specially, inputs of Seq2Seq, Transformer, IE+MSA, and T-CVAE are concatenated with context representations and image features.

4.5 Result Analysis

Automatic and Manual Evaluation

We perform experiments on the VIST-E dataset comparing with several strong baselines, i.e., Seq2Seq, Transformer, IE+MSA, and T-CVAE.

The results of automatic and manual evaluation are shown in Table 2. We have done significant test comparing our model with these baselines by running all these models ten times. The results shows that our model significantly outperforms them with all p-values < 0.01. Specifically, our model implements an improvement of 8.66 / 5.39 / 3.42 / 8.23 / 3.14 / 1.66 over the Seq2Seq / Transformer / IE-MAS / T-CVAE / MG+Trans / MG+CIA on B1. As for B4, our model achieves an improvement of 0.8 / 0.48 / 0.86 / 1.36 / 0.03 / 0.14 over the Seq2Seq / Transformer / IE-MAS / T-CVAE / MG+Trans/ MG+CIA. With respect to M, our model outperforms the Seq2Seq / Transformer / IE-MAS / T-CVAE / MG+Trans / MG+CIA by 3.3 / 0.93 / 1.25 / 3.61 / 0.21 / 0.55. And for R-L, our model implements an improvement of 4.82 / 3.43 / 1.04 / 6.15 / 2.04 / 0.54 over the Seq2Seq / Transformer / IE-MAS / T-CVAE / MG+Trans / MG+CIA. The results show that our MGCL model can comprehend the context better with the MGCN module, and merge the context features and image features effectively with the MCIA module.

Our MGCL model also outperforms baselines on all manual evaluation. Compared with the best baseline, Gram. increases from 3.46 to 3.51, Logic.

Model	B1	B2	B4	M	R-L
Seq2Seq	14.27	4.27	1.05	6.02	16.32
Transformer	17.06	6.18	1.57	6.55	18.69
IE+MSA	20.11	6.62	1.68	6.87	**21.27**
T-CVAE	20.36	6.63	1.88	6.74	20.98
Plan&Write	20.92	5.88	1.44	7.10	20.17
KE-GPT2	**21.92**	**7.40**	1.90	**7.41**	20.58
MG+Trans (ours)	18.55	6.76	**2.33**	7.31	19.02
MGCL (ours)	20.27	6.26	1.81	6.91	21.01

Table 4: Experiments on the VIST-E dataset (plain text) for the SEG task. The **bold** / underline denotes the best and the second performance, respectively.

increases from 2.77 to 3.17, and Rele. increases from 2.60 to 3.75. It shows that our model can generate the more coherent and reasonable story endings than other baselines. Notably, our model has a good performance on Rele., which shows that MCIA module is helpful for enhancing the link of the generated story endings with the images.

Ablation Study To explore the effectiveness of our MGCL, we perform the ablation experiments on VIST-E (cf. Table 3). When removing the MGCN module and using the hidden features of the previous LSTM directly, the performance of our model drops 0.46 on B4, 0.61 on M, 1.2 on C, and 1.92 on R-L, respectively. When removing the MCIA and using hidden features of TD directly, the performance drops 0.32 on B4, 0.71 on M, 3.84 on C, and 1.91 on R-L, respectively. When removing TD and MCIA and using a LSTM unit to decoder, the performance drops 1.72 on B4, 2.35 on M, 12.8 on C, and 4.02 on R-L. All of the these show that the MGCN module and MCIA module can help to generate the more contextual-consistent and image-related story endings.

Comparison on SEG To verify the effectiveness of image guidance, we conduct experiments on VIST-E dataset removing the image. The automatic evaluation results are shown in Table 4. Compared with the corresponding results in Table 2, the Seq2Seq, Transformer, and our models have poor performance overall, which indicates the image is helpful for generating better endings. But for IE+MSA and T-CVAE model, they have poor performance when adding the image. One possible reason is that they are designed for the textual story generation specially, so it is hard to change to generate better story endings with an image. Further, we also conduct the SEG experiments with another two recent models, the Plan&Write model (Yao et al., 2019) and the pretrained language KE-GPT2 model (Guan et al., 2020). The results show that KE-GPT2 achieves the best performance on the plain text dataset, while our models are close to it.

Task	Noun	Verb	Adjective
IgSEG	11,659	11,050	6,508
SEG	9,808	9,027	5,835

Table 5: Statistics of Noun, Verb, and Adjective in the generated sentences of our MGCL model.

To research the quantity transformation of part of speech, we count the number of Noun, Verb, and Adjective in the generated story endings (cf. Table 5). With the guidance of image, the story endings achieves an improvement of 18.87%, 22.41%, and 11.53% on Noun, Verb, and Adjective, respectively. The results indicates that the MGCL model can enrich story endings on the IgSEG task.

4.6 Visualization and Case Study

To explicitly demonstrate our model, we present the visualization and case study (cf. Figure 5).

The context (Figure 5(a)) is mainly about people going to the seaside for a holiday. The context is encoded by the MGCN module, and fed with the image features together into the MCIA module. Key words are marked in red by our model, the darker means more important. The entities, events, and emotion (e.g., *We*, *go*, *trip*, and *exciting*) are assigned more attention weights by our model. It shows that our model can understand the semantic information of context sufficiently. Similarly, we present visualization for image (Figure 5(b)). The regions in darker red are where our model focuses on. Our model pays more attention to the regions with the important objects (e.g., *bridge*, *city*, and *sky*, which can be regarded as *view*). It shows that our model can also capture the vital visual concepts in the image. As shown in Figure 5(c), we present the cases of baselines on SEG and IgSEG, respectively. On SEG, the Seq2Seq model generates a long ending with repeat words , but it is still reasonable. The Transformer and T-CVAE generate the generic endings with positive sentiment. Our model generates the sentence describing how “we” ended the day, which is more interesting. On IgSEG, all the models generate the more specific endings where the visual concepts are mentioned. These cases show that our model can capture context-relevant image concepts and generate informative story plots.

X_1:Everyone was excited to be going on vacation.

X_2:We stopped at a memorial and spent our time reading about the history of it.

X_3:It was fourth of July that day, we were proud to be American.

X_4:We decided to go on a diving trip, it seemed exciting.

GT: We spent the rest of the day on the water, it was a great day!

(a) (b)

Model	SEG	IgSEG
Seq2Seq	At the end of the day, we went back to location location and view what a beautiful sky, it was a trip home from many times.	We went to the bridge.
Tramsformer	At the end of the day, we had a great time.	After that we decided to go back to the beach.
IE+MSA	We were all gone, but we had fun all the time.	We have the great time at the beach
T-CVAE	We were glad to be going to the future.	We these the tired enjoy my day in a nap.
MG+Trans(Ours)	We ended the day with a beautiful sunset.	It was a great day at the beach.
Ours[†]	We ended the day with a great view.	We have the great time at the beach.
Ours	We ended the day with a beautiful view of the water.	We ended the day by watching the **bridge** and enjoy the **view**.

(c)

Figure 5: Visualization and case study. The regions where are dark in color of sentences and images mean that model pays more attention to. GT denotes Ground True. (Best viewed in color)

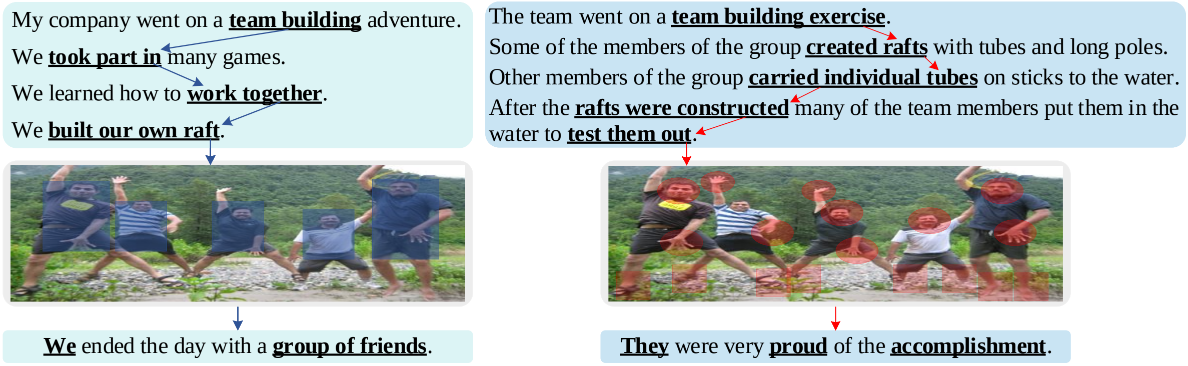
My company went on a **team building** adventure.
We **took part in** many games.
We learned how to **work together**.
We **built our own raft**.

We ended the day with a **group of friends**.

The team went on a **team building exercise**.
Some of the members of the group **created rafts** with tubes and long poles.
Other members of the group **carried individual tubes** on sticks to the water.
After the **rafts were constructed** many of the team members put them in the water to **test them out**.

They were very **proud** of the **accomplishment**.

GT: In the end we all became friends.

(a)

GT: Even though they were wet at the end of the day they felt as sense of accomplishment

(b)

Figure 6: Different story context guided by the same ending-related image for IgSEG.

To vividly demonstrate the impact of image on IgSEG task, we show the generation cases (cf. Figure 6) which are offered the same ending-related image but the different story context. The content of image is that five people are very happy and jump in front of camera. The generated endings (a) and (b) are both coherent with their corresponding context. The context (a) has the logic chain (e.g., *team building* → *took part in games* → *work together* → *built raft*), and the context (b) has the logic chain (e.g., *team building* → *created raft* → *carried tubes* → *rafts test*). According to different logic chains, our model may focus on different regions of image and generate the story endings with the various semantics. The context (a) merely links to number of people in the image (e.g., *friends*), while the context (b) may be associated with people's postures and emotions (e.g., *dump* and *laugh* mean *proud*). To some extent, our MGCL model is able to capture some latent high-level semantics (e.g., *pride* and *celebration*) hidden in the image.

5 Conclusion

We propose a new task termed Image-guided Story Ending Generation. We transform the VIST dataset to VIST-Ending for IgSEG. We propose a MGCL model which uses a multi-layer graph convolutional networks to capture intra- and inter-sentence relations, a multiple context-image attention module to merge the context features and image features. Results on automatic and manual evaluation show that our model outperforms all the baselines.

Acknowledgements

We thank the anonymous reviewers for valuable comments and thoughtful suggestions.

This work was supported by National Natural Science Foundation of China (62076100, 51767005), and the collaborative research grants from the Fundamental Research Funds for the Central Universities, SCUT (D2210010, D2200150, and D2201300), the Science and Technology Planning Project of Guangdong Province (2017B050506004), the Science and Technology Programs of Guangzhou (201704030076, 201707010223, 201802010027, 201902010046), and the Hong Kong Research Grants Council, China (PolyU1121417, C1031-18G), and an internal research grant from the Hong Kong Polytechnic University, China (1.9B0V).

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6077–6086.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592.

Shivam Gaur. 2019. Generation of a short narrative caption for an image using the suggested hashtag. In *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*, pages 331–337. IEEE.

Jian Guan, Fei Huang, Minlie Huang, Zhihao Zhao, and Xiaoyan Zhu. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8(0):93–108.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6473–6480.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Junjie Hu, Yu Cheng, Zhe Gan, Jingjing Liu, Jianfeng Gao, and Graham Neubig. 2020. What makes a good story? designing composite rewards for visual storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7969–7976.

Qingbao Huang, Linzhang Mo, Pijian Li, Yi Cai, Qingguang Liu, Jielong Wei, Qing Li, and Ho-fung Leung. 2021. Story ending generation with multi-level graph convolutional networks over dependency trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1–9.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8465–8472.

Ting-Hao (Kenneth) Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross B. Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

Yunjae Jung, Dahun Kim, Sanghyun Woo, Kyungsu Kim, Sungjin Kim, and In So Kweon. 2020. Hide-and-tell: Learning to bridge photo streams for visual storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11213–11220.

Taehyeong Kim, Min-Oh Heo, Seonil Son, Kyoung-Wha Park, and Byoung-Tak Zhang. 2018. Glac net: Glocal attention cascading networks for multi-image cued story generation. *arXiv preprint arXiv:1805.10973*.

Nanxing Li, Bei Liu, Zhizhong Han, Yu-Shen Liu, and Jianlong Fu. 2019. Emotion reinforced visual storytelling. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 297–305.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Generating reasonable and diversified story ending using sequence to sequence model with adversarial training. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1033–1043.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Lingbo Mo, Chunhong Zhang, Yang Ji, and Zheng Hu. 2019. Adversarial learning for visual storytelling with sense group partition. In *Computer Vision – ACCV 2018*, pages 175–190.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575.

Bairui Wang, Lin Ma, Wei Zhang, Wenhao Jiang, and Feng Zhang. 2019. Hierarchical photo-scene encoder for album storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8909–8916.

Ruize Wang, Zhongyu Wei, Piji Li, Qi Zhang, and Xuanjing Huang. 2020. Storytelling from an image stream using scene graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9185–9192.

Tianming Wang and Xiaojun Wan. 2019. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5233–5239.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909.

Pengcheng Yang, Fuli Luo, Peng Chen, Lei Li, Zhiyi Yin, Xiaodong He, and Xu Sun. 2019. Knowledgeable storyteller: A commonsense-driven generative model for visual storytelling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5356–5362.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7378–7385.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Yan Zhao, Lu Liu, Chunhua Liu, Ruoyao Yang, and Dong Yu. 2018. From plots to endings: A reinforced pointer generator for story ending generation. In *Natural Language Processing and Chinese Computing*, pages 51–63.

Improve Query Focused Abstractive Summarization by Incorporating Answer Relevance

Dan Su*, Tiezheng Yu*, Pascale Fung
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{dsu,tyuah}@connect.ust.hk, pascale@ece.ust.hk

Abstract

Query focused summarization (QFS) models aim to generate summaries from source documents that can answer the given query. Most previous work on QFS only considers the query relevance criterion when producing the summary. However, studying the effect of answer relevance in the summary generating process is also important. In this paper, we propose QFS-BART, a model that incorporates the explicit answer relevance of the source documents given the query via a question answering model, to generate coherent and answer-related summaries. Furthermore, our model can take advantage of large pre-trained models which improve the summarization performance significantly. Empirical results on the Debatepedia dataset show that the proposed model achieves the new state-of-the-art performance.[1]

Document: Interrogator Ali Soufan said in an April op-ed article in the New York Times: "It is inaccurate to say that Abu Zubaydah had been uncooperative [and that enhanced interrogation techniques supplies interrogators with previously unobtainable information]. Along with another f.b.i. agent and with several c.i.a. officers present I questioned him from March to June before the harsh techniques were introduced later in August. Under traditional interrogation methods he provided us with important actionable intelligence."
Query: Are traditional interrogation methods insufficient?
Summary: The same info can be obtained by traditional interrogations.

Table 1: An example of QFS. The input is a document and a corresponding query, and the highlight sentence is the answer from our QA module. We observe that the summary and the answers are very correlated.

1 Introduction

Query focused summarization (QFS) models aim to extract essential information from a source document(s) and organize it into a summary that can answer a query (Dang, 2005). The input can be either a *single* document that has multiple views or *multiple* documents that contain multiple topics, and the output summary should be focused on the given query. QFS has various applications (e.g., a personalized search engine that provides the user with an overview summary based on their query (Su et al., 2020b)).

Early work on the QFS task mainly focused on generating extractive summaries (Davis et al., 2012; Daumé III and Marcu, 2006; Feigenblat et al., 2017; Xu and Lapata, 2020b), which may contain unreadable sentence ordering and lack cohesiveness. Other work on abstractive QFS incorporated the query relevance into existing neural summarization models (Nema et al., 2017; Baumel et al., 2018). The closest work to ours was done by (Su et al., 2020a) and (Xu and Lapata, 2020a,b), who leveraged an external question answering (QA) module in a pipeline framework to take into consideration the answer relevance of the generated summary. However, they only used QA as distant supervision to retrieve relevant segments for generating the summary, but did not take into consideration the answer relevance in the generation model. As shown in the Table 1, the query focused summary is correlated to the answer extracted from the QA module.

On the other hand, recent neural summarization models (Paulus et al., 2017; Gehrmann et al., 2018; Zhang et al., 2020) have achieved remarkable performance in *generic* abstractive summarization by taking advantage of large pre-trained language

---

* The two authors contribute equally.

[1] The code is released at: `https://github.com/HLTCHKUST/QFS`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3124–3131
August 1–6, 2021.

models (Lewis et al., 2019; Zhang et al., 2020). Yet, how to leverage these models and adapt them to the QFS task remains unexplored.

In this work, we propose QFS-BART, a BART-based (Lewis et al., 2019) framework for abstractive QFS that incorporates explicit answer relevance. We leverage a state-of-the-art QA model (Su et al., 2019) to predict the answer relevance of the given source documents to the query, then further incorporate the answer relevance into the BART-based generation model. We conduct empirical experiments on the Debatepedia dataset, one of the first large-scale QFS datasets (Nema et al., 2017), and achieve the new state-of-the-art performance on the ROUGE metrics compared to all previously published work.

Our contributions in this work are threefold:

- Our work demonstrates the effectiveness of the answer relevance score in neural abstractive QFS.
- We propose an effective method to incorporate the answer relevance score into the pre-trained language models which can produce more query-relevant summaries.
- Our model reaches the state-of-the-art performance on a single-document QFS dataset (Debatepedia), and brings substantial improvements over several strong baselines on two multi-document QFS datesets (DUC 2006, 2007).

2 Related Work

Abstractive summarization models aim to generate short, concise and readable text that extracts the salient information from a document. In the past few years, significant achievements (See et al., 2017; Liu and Lapata, 2019; Lewis et al., 2019; Dong et al., 2019) have been made in the *generic* abstractive summarization task which is attributed to the advanced neural architectures and the availability of large-scale datasets (Sandhaus, 2008; Hermann et al., 2015; Grusky et al., 2018).

QFS is a more complex task that aims to generate a summary according to the query and its relevant document(s). Nema et al. (2017) proposed an encode-attend-decode system with an additional query attention mechanism and diversity-based attention mechanism to generate a more query-relevant summary. Baumel et al. (2018) incorporated query relevance into a pre-trained abstractive summarizer to make the model aware of the query, while Xu and Lapata (2020a) discovered a new type of connection between generic summaries and QFS queries, and provided a universal representation for them which allows *generic* summarization data to be further exploited for QFS. Su et al. (2020b), meanwhile, built a query model for paragraph selection based on the answer relevance score and iteratively summarized paragraphs to a budget. Although Xu and Lapata (2020a) and Su et al. (2020b) utilized QA models for sentence- or paragraph- level answer evidence ranking, they did not make use of answer relevance to query-forcused generation.

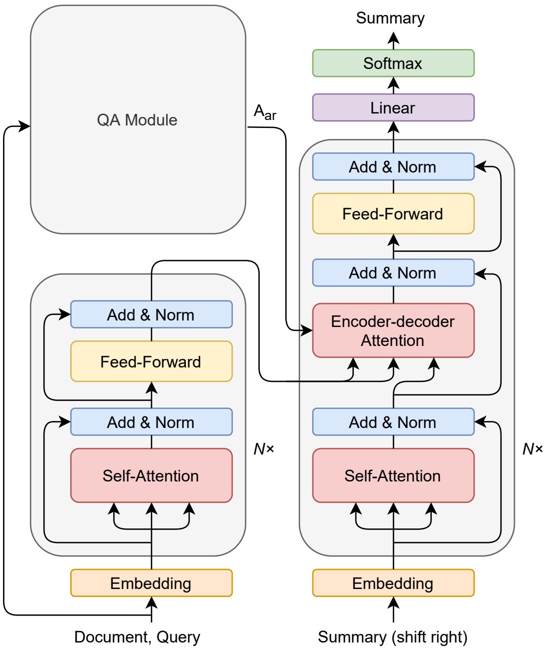

Figure 1: The framework of QFS-BART. The QA module calculates the answer relevance scores, and we incorporate the scores as explicit answer relevance attention to the encoder-decoder attention.

To the best of our knowledge, we are the first to leverage explicit answer relevance to abstractive QFS. In addition, our approach can be easily combined with pre-trained Transformers (Song et al., 2019; Dong et al., 2019; Lewis et al., 2019; Xiao et al., 2020), which have shown great success for the *generic* abstractive summarization task.

3 Methodology

In this section, we present our approach to incorporating the answer relevance into QFS. First, we introduce the method of generating answer relevance scores. Then, we describe our answer relevance attention in the Transformer-based model. Third, we introduce our QFS-BART model in which the decoder is composed of a stack of answer relevance decoding layers, as shown in Figure 1.

3.1 Answer Relevance Generation

In recent years, neural models (Yang et al., 2019a; Su et al., 2019) have shown remarkable achievements in QA tasks. In order to apply QA models to the QFS task, we use HLTC-MRQA (Su et al., 2019) to generate the answer relevance score for each word in context. The reason for choosing HLTC-MRQA is twofold: 1) it shows robust generalization and transferring ability on different datasets, and 2) the model shows great performance in QA tasks and significantly outperforms the BERT-large baseline by a large margin. The HLTC-MRQA is introduced as follows.

Based on XLNet (Yang et al., 2019b), HLTC-MRQA is fine-tuned on multiple QA datasets with an additional multilayer perceptron (MLP). Given a context that contains n words, the model outputs a distribution $s \in (0, 1)$ for each word's probability of being the start word of the answer and a probability distribution $e \in (0, 1)$ to be the end word of answer. To generate the answer relevance score r for each word, we calculate it by summing two distributions:

$$r = s + e, \tag{1}$$

where $r \in (0, 2)$.

3.2 Answer Relevance Attention

Scaled dot-product attention (Vaswani et al., 2017) is the core-component of the Transformer-based model:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}}V), \tag{2}$$

where d is the dimension of the query matrix Q, key matrix K and value matrix V. The Transformer encoder is constructed by self-attention layers, where all of the keys, values and queries come from the input sequence. This makes each token in the input attend to all other tokens. The Transformer decoder layer is a combination of a self-attention layer and encoder-decoder attention layer. In the encoder-decoder attention layer, the query comes from the decoder's self-attention layer, and the key and value come from the output of the encoder. This allows every generated token to attend to all tokens in the input sequence.

In this work, we propose to incorporate the word-level answer relevance score as additional explicit encoder-decoder attention in the transformer decoder. Given a document with n tokens, we generate a summary with a maximum length of m tokens. Let $x^l \in \mathbb{R}^{n*d}$ denotes the output of the l-th transformer encoder layer and $y^l \in \mathbb{R}^{m*d}$ denotes the output of the l-th transformer decoder layer's self-attention layer. The encoder-decoder attention $\alpha^l \in \mathbb{R}^{m*n}$ can be computed as:

$$\alpha^l = softmax(\frac{(y^l W_Q)(x^l W_K)}{\sqrt{d_k}} + A_{ar}), \tag{3}$$

where W_Q and $W_K \in \mathbb{R}^{d_k * d_k}$ are parameter weights and $A_{ar} \in \mathbb{R}^{m*n}$ is our explicit answer relevance score. Since the original answer relevance score is an n-dimensional vector, we repeat it m times to generate an m by n attention matrix, which means our answer relevance attention is equal to all generated tokens.

3.3 QFS-BART

Generative pre-trained models (Dong et al., 2019; Lewis et al., 2019; Raffel et al., 2019) have shown remarkable performance in natural language generation (NLG), including text summarization. We choose to combine our answer relevance attention with BART (Lewis et al., 2019), a denoising autoencoder built with a sequence-to-sequence model, for two reasons: 1) BART achieves state-of-the-art performance on several summarization datasets (i.e. CNN/DailyMail (Hermann et al., 2015) and Xsum (Narayan et al., 2018)). 2) BART follows the standard Transformer encoder-decoder architecture, and we can easily combine the answer relevance as explicit attention to the encoder-decoder attention layers. In detail, we incorporate the same answer relevance attention for all Transformer decoder layers.

Domain adaption for natural language processing tasks is widely studied (Blitzer et al., 2007; Daumé III, 2009; Liu et al., 2020; Yu et al., 2021). Hua and Wang (2017) first studied the adaptation of neural summarization models and showed that the models were able to select salient information, even when trained on out-of-domain data. Inspired by this, we leverage a two-stage fine-tuning method for our QFS-BART. In the first stage, we directly fine-tune the original BART model with the Xsum

Models	ROUGE-1	ROUGE-2	ROUGE-L
Without Pre-training			
Transformer (Vaswani et al., 2017)	28.16	17.48	27.28
Transformer (CONCAT)	41.72	33.62	41.25
Transformer (ADD)	41.10	33.35	40.72
SD2* (Nema et al., 2017)	41.26	18.75	40.43
CSA Transformer* (Xie et al., 2020)	46.44	37.38	45.85
With Pre-training			
RSA Word Count* (Baumel et al., 2018)	53.09	16.10	46.18
QR-BERTSUM-TL* (Laskar et al., 2020)	57.96	**45.20**	57.05
BART-FT	57.98	43.62	56.30
QFS-BART	**59.02**	44.59	**57.44**

Table 2: ROUGE-F1 scores for Debatepedia QFS dataset. Results with * mark are taken from the corresponding papers. The previous work can be divided into two categories: 1) training the models from scratch, and 2) using pre-trained models and fine-tuning on a QFS dataset.

Document: Interrogator Ali Soufan said in an April op-ed article in the New York Times: “It is inaccurate to say that Abu Zubaydah had been uncooperative [and that enhanced interrogation techniques supplies interrogators with previously unobtainable information]. Along with another f.b.i. agent and with several c.i.a. officers present I questioned him from March to June before the harsh techniques were introduced later in August. Under traditional interrogation methods he provided us with important actionable intelligence.”
Query: Are traditional interrogation methods insufficient?

BART-FT: Al Qaeda detainee Abu Zubaydah has been cooperative under traditional interrogation.

QFS-BART: The same info can be obtained by traditional interrogation.

Gold: The same info can be obtained by traditional interrogations.

Table 3: A example taken from Debatepedia test set. The generated summary from **QFS-BART** is almost the same as the gold summary.

dataset, and in the second stage, we fine-tune our QFS-BART model with QFS datasets. All the parameters in the model are initialized from the first stage. In order to make the model capture both query relevance and answer relevance, the input text is formatted in the following way:

[CLS] document [SEP] query.

The answer relevance attention score for the document is generated by the QA model, and we take the maximum number in the document as the attention score for all the words in the query.

4 Experimental Setup

Datasets We use multiple QA datasets, including SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018) and NaturalQuestions (Kwiatkowski et al., 2019) to train HLTC-MRQA, following Su et al. (2019). We evaluate our model on the Debatepedia dataset (Nema et al., 2017) and DUC2005-7 dataset (in Appendix).

Training Details For all the experiments, we use the BART-large version to implement our models. We use a mini-batch size of 32 and train all the models on one V100 16G. During decoding, we use beam search with the beam size of 4. We decode until an end-of-sequence token is emitted and early stop when the generated summary reaches to 48 tokens.

5 Results & Analysis

We compare our proposed QFS-BART model with the following models: 1) **Transformer** does not consider the queries in the Debatepedia dataset. 2) **Transformer (CONCAT)** concatenates the query and the document. 3) **Transformer (ADD)** adds the query encoded vector to the document encoder. 4) **SD2** adds a query attention model and a new diversity-based attention model to the encode-attend-decode paradigm. 5) **CSA Transformer** combines conditional self-attention (CSA) with Transformer. 6) **RAS Word Count** incorporates query relevance into a pre-trained abstractive summarization model. 7) **QR-BERTSUM-TL** presents a transfer learning technique with the Transformer-based BERTSUM model (Liu and Lapata, 2019). 8) **BART-FT** concatenates the document and query, and directly fine-tunes on the

Debatepedia dataset.

We adopt ROUGE score (Lin, 2004) as the evaluation metric. As shown in Table 2, QFS-BART significantly outperforms the models without pre-training. Compared with the models utilizing pre-training, ours improves the ROUGE-1 and ROUGE-L scores by a large margin.

5.1 Case Study

We present a case study comparing between the strong baseline BART-FT model, our QFS-BART model and the gold summary, shown in Table 3. It's clear that the baseline model tends to copy spans from the document which are not directly related to the query and the QFS-BART model produces a more query- and answer- related summary.

6 Conclusions

In this work, we propose QFS-BART, an abstractive summarization model for query focused summarization. We use a generalizing QA model to make explicit answer relevance scores for all words in the document and combine them to the encoder-decoder attention. We also leverage pre-trained model (e.g. BART) and two-stage fine-tuning method which further improve the summarization performance significantly. Experimental results show the proposed model achieves state-of-the-art performance on Debatepedia dataset and outperforms several comparable baselines on DUC 2006-7 datasets.

Acknowledgments

We want to thank the anonymous reviewers for their constructive feedback. This work is partially funded by ITS/353/19FP of the Innovation Technology Commission, the Hong Kong SAR Government.

References

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of the document understanding conference*, volume 2005, pages 1–12.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 305–312.

Sashka T Davis, John M Conroy, and Judith D Schlesinger. 2012. Occams–an optimal combinatorial covering algorithm for multi-document summarization. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 454–463. IEEE.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Guy Feigenblat, Haggai Roitman, Odellia Boni, and David Konopnicki. 2017. Unsupervised query-focused multi-document summarization using the cross entropy method. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 961–964.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28:1693–1701.

Xinyu Hua and Lu Wang. 2017. A pilot study of domain adaptation effect for neural abstractive summarization. *arXiv preprint arXiv:1707.07062*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Md Tahmid Rahman Laskar, Enamul Hoque, and Jimmy Huang. 2020. Query focused abstractive summarization via incorporating query relevance and transfer learning with transformer models. In *Canadian Conference on Artificial Intelligence*, pages 342–348. Springer.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2020. Crossner: Evaluating cross-domain named entity recognition. *arXiv preprint arXiv:2012.04373*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Preksha Nema, Mitesh Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. *arXiv preprint arXiv:1704.08300*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Dan Su, Yan Xu, Genta Indra Winata, Peng Xu, Hyeondey Kim, Zihan Liu, and Pascale Fung. 2019. Generalizing question answering system with pre-trained language model fine-tuning. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 203–211.

Dan Su, Yan Xu, Tiezheng Yu, Farhad Bin Siddique, Elham Barezi, and Pascale Fung. 2020a. CAiRE-COVID: A question answering and query-focused multi-document summarization system for COVID-19 scholarly information management. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Dan Su, Yan Xu, Tiezheng Yu, Farhad Bin Siddique, Elham J Barezi, and Pascale Fung. 2020b. Caire-covid: A question answering and multi-document summarization system for covid-19 research. *arXiv preprint arXiv:2005.03975*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gen: An enhanced multi-flow pre-training and fine-tuning framework for natural language generation. *arXiv preprint arXiv:2001.11314*.

Yujia Xie, Tianyi Zhou, Yi Mao, and Weizhu Chen. 2020. Conditional self-attention for query-based summarization. *arXiv preprint arXiv:2002.07338*.

Yumo Xu and Mirella Lapata. 2020a. Abstractive query focused summarization with query-free resources. *arXiv preprint arXiv:2012.14774*.

Yumo Xu and Mirella Lapata. 2020b. Coarse-to-fine query focused multi-document summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3632–3645.

Yumo Xu and Mirella Lapata. 2020c. Query focused multi-document summarization with distant supervision. *arXiv preprint arXiv:2004.03027*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Tiezheng Yu, Zihan Liu, and Pascale Fung. 2021. Adaptsum: Towards low-resource domain adaptation for abstractive summarization. *arXiv preprint arXiv:2103.11332*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

A Adapting QFS-BART to DUC 2005-7

DUC 2005-7 are datasets for the multi-document query focused summarization (QFS) task. As shown in the Table 4, the documents and summaries of the DUC datasets are extremely longer than those in the Debatepedia (Nema et al., 2017) dataset. We thus need to adapt the QFS-BART model to handle the multi-document scenario and produce longer output.

Datasets	**Document(s)**	**Query**	**Summary**
Debatepedia	66.40	11.16	9.97
DUC 2005	20058.12	26.60	243.56
DUC 2006	14330.14	23.30	246.84
DUC 2007	10759.17	21.57	243.94

Table 4: Average length of the input documents, queries and output summaries for the Debatepedia and DUC 2005-7 datasets. For the DUC datasets, we add up the lengths of all the documents related the same query

In this paper, we introduce a two-step architecture: 1) Retrieve answer-related sentences given the query, rank them by the confidence score (generated from Equation 4) and concatenate them. 2) Use our QFS-BART to produce an abstractive summary.

A.1 Answer Retrieving

We split documents into paragraphs and feed each paragraph to the QA model to get answer-related sentences. Then the sentences are ranked by the confidence score.

Document Segmentation The QA model selects one answer span given an input document, and the sentences that contain the span will be chosen as the answer-related sentences. Since we only retain the answer-related sentences as input to the next step, we set the maximum paragraph length to 300 words to avoid missing too much information in this step. Specifically, we feed text to the paragraph sentence by sentence until it reaches the maximum length.

Answer Relevance Ranking The paragraphs are fed to the QA model to generate answer-related sentences and the corresponding answer relevance scores. We align each sentence with a confidence score from the corresponding answer span. The confidence score is defined as:

$$confidence\_score = P_{start} + P_{end}, \qquad (4)$$

where P_{start} and P_{end} is two probability distributions over the tokens in the context. $P_{start}(i)$/$P_{end}(i)$ the probability of the i-th token is the start/end of the answer span in context.

A.2 Summary Generation

We use the answer-related sentences and their answer relevance scores as the input to the QFS-BART model. The DUC 2005 dataset is used as a development set to optimize the model, and we evaluate the performance on the DUC 2006-7 dataset. We compare our QFS-BART with the following models.

Models	**DUC 2006**			**DUC 2007**		
	1	**2**	**SU4**	**1**	**2**	**SU4**
LEAD	32.1	5.3	10.4	33.4	6.5	11.3
TEXTRANK	34.2	6.4	11.4	35.8	7.7	12.7
HLTC-MRQA	39.1	8.3	13.5	**40.6**	9.6	14.7
BART-CAQ*	38.3	7.1	12.9	40.5	9.2	14.4
BART-FT	38.9	8.5	13.9	40.4	**10.0**	**15.1**
QFS-BART	**39.4**	**8.6**	**14.1**	39.22	9.39	14.34

Table 5: ROUGE-F1 scores for DUC 2006-7 dataset. Results with * mark are taken from the corresponding papers.

LEAD (Xu and Lapata, 2020c) returns all leading sentences of the most recent document up to 250 words.

TEXTRANK (Mihalcea and Tarau, 2004) is a graph-based ranking model that incorporate two unsupervised methods for keyword and sentence extraction.

HLTC-MRQA truncates the ranked answer related sentences from our first step as the extractive summary.

BART-CQA (Su et al., 2020b) uses QA models for paragraph selection and iteratively summarizes paragraphs to 250 words.

We adopt ROUGE-F1 score (Lin, 2004) as the evaluation metric. As shown in Table 5, HLTC-MRQA significantly outperforms the LEAD and TEXTRANK baselines, which indicates the effectiveness of our answer retrieval. However, QFS-BART does not perform well on DUC 2006-7 datasets. We conjecture that the model can not converge to the task well with limited training samples (DUC 2005 contains only 300 samples).

Learning a Reversible Embedding Mapping using Bi-Directional Manifold Alignment

Ashwinkumar Ganesan[*], **Francis Ferraro, Tim Oates**
Dept. Of Computer Science & Electrical Engineering (CSEE),
University Of Maryland Baltimore County (UMBC),
MD, USA - 21250
`gashwin1@umbc.edu, ferraro@umbc.edu, oates@cs.umbc.edu`

Abstract

We propose a Bi-Directional Manifold Alignment (BDMA) that learns a non-linear mapping between two manifolds by explicitly training it to be bijective. We demonstrate BDMA by training a model for a pair of languages rather than individual, directed source and target combinations, reducing the number of models by 50%. We show that models trained with BDMA in the "forward" (source to target) direction can successfully map words in the "reverse" (target to source) direction, yielding equivalent (or better) performance to standard unidirectional translation models where the source and target language is flipped. We also show how BDMA reduces the overall size of the model.

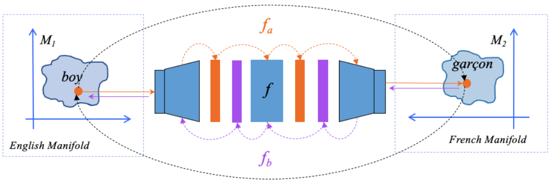

Figure 1: **Mapping vector spaces with Bi-Directional Manifold Alignment (BDMA).** f is the feedforward network. f_a and f_b are represent the forward and backward direction of flow through the network. In a shared BDMA network, the blue components represent network fully connected layers, orange are activation layers during forward network flow while purple represents activation layers in reverse flow. During reverse flow from output to input, the weight matrix is a transpose of weights during forward flow through the network.

1 Introduction

Learning continuous vector representations of embeddings is an expensive exercise as it requires a large quantity of free text to train *stable* representations (Sahin et al., 2017). Learning word embeddings in the English language is relatively easy since a model can make use of free text online from sources like *Wikipedia*, but it is challenging to learn embeddings for natural languages where the free text is limited (low-resource languages). Resource-constrained languages suffer from dual problems of reduced quality of embeddings and their vocabulary being small. Cross-lingual words embedding (CLWE) models alleviate this problem but are often linear mapping functions that align the source and target language manifolds, since non-linear mapping functions such as neural networks are unidirectional and known to perform poorly as compared to their linear counterparts (Ruder et al., 2019).

In this paper, we propose Bi-Directional Manifold Alignment (**BDMA**), which learns a *reversible*, non-linear mapping function between two manifolds. Inspired by CycleGAN (Zhu et al., 2017), we use a *cycle consistency* loss to optimize BDMA. We study BDMA in the context of cross-lingual lexicon induction and show that it offers solutions to two known problems: (1) that non-linear models are known to perform poorly in comparison to their linear counterparts (Ruder et al., 2019), and (2) most approaches perform unidirectional mapping only (from a source to target language), leading to an ever increasing set of translation models. We show how BDMA is a *generic* training method that uses different distance metrics (or losses) like MSE, cosine or RCSLS (Joulin et al., 2018) while training models cyclically.[1]

2 Bi-Directional Manifold Alignment

Consider two manifolds $M^s \in \mathbb{R}^{n \times d}$ (source domain) and $M^t \in \mathbb{R}^{m \times d}$ (target domain) that are vector space representations of words. The monolingual word embeddings are pretrained from a large corpus and may be created using different

---

[*] This research was completed prior to joining Amazon.

[1] Implementation see https://github.com/codehacken/bdma.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3132–3139
August 1–6, 2021. ©2021 Association for Computational Linguistics

methods. Let V^s and V^t be the respective vocabularies of the two languages. Hence $V^s = \{w_1^s \dots w_n^s\}$ and $V^t = \{w_1^t .. w_m^t\}$ are words in each vocabulary of size n and m. The distributed representations of words in each manifold are $M^s = \{m_1^s \dots m_n^s\}$ and $M^t = \{m_1^t \dots m_m^t\}$. We assume there is $V^p = \{w_1^p \dots w_c^p\}$, an available dictionary or parallel corpus of words for the given source/target pair.

2.1 Bi-Directional Loss Mechanism

We achieve bi-directional alignment by learning a mapping function is optimized with a *cyclic-consistency loss* (CCL). In Figure 1, the mapping function $f_a : M^s \to M^t$ to align the manifold M^s to M^t. We also use a backward mapping function $f_b : M^t \to M^s$ to align the manifold M^t to M^s. We refer to the parameters of both f_a and f_b as θ_f.

Our method is based on jointly minimizing the distance $\mathcal{D}$ between pairs of embeddings, and their mapped counterparts, from each manifold. We define our cycle consistency loss for a single training sample based on this distance function $\mathcal{D}$ as

$$\mathcal{L}_{\text{ccl}}^{\mathcal{D}}(i) = \mathcal{D}(f_a(m_i^s), m_i^t) + \mathcal{D}(f_b(m_i^t), m_i^s). \quad (1)$$

Following previous work (Xing et al., 2015), we include an *orthogonal loss* in the objective; we extend this loss function for a neural network by performing a **layerwise** orthogonal loss. For our full objective, we sum over all training instances and minimize over θ_f:

$$\mathcal{L}_{\text{ccl}} = \min_{\theta_f} \sum_{i \in V^p} \mathcal{L}_{\text{ccl}}^{\mathcal{D}}(i) + \sum_{w_j \in \theta_f} w_j w_j^T - I \quad (2)$$

where w_j are weights of layer j in the network. While Euclidean distance (mean squared error: $\mathcal{D} = \text{MSE}$) is a common way of computing distance in a manifold (Ruder et al., 2019; Artetxe et al., 2016), cosine or relaxed cross-domain similarity local scaling (RCSLS) (Joulin et al., 2018) distance functions have been shown to be effective for word and embedding alignment tasks. Our formulation works with these other computable distance functions. For example, while applying $\mathcal{D} = \text{MSE}$, and for ease omitting the orthogonal loss term $\sum_{w_j \in \theta_f} w_j w_j^T - I$, the loss is

$$\min_{\theta_f} \sum_{i \in V^p} \|f_a(m_i^s) - m_i^t\|_2^2 + \|f_b(m_i^t) - m_i^s\|_2^2 .$$

See Appendix D for similar formulations for $\mathcal{D} = \text{cosine}$, $\mathcal{D} = \text{RCSLS}$, and a combined distance function $\mathcal{D} = \text{cosine} + \text{RCSLS}$ (used in §3).

2.2 Forward - Reverse Network Flow

As described in §2.1, f_a and f_b represent the forward and reverse network flow. We represent the forward and reverse mapping with two networks that have shared or independent parameters. When the parameters are independent, two separate networks are trained simultaneously and optimized in order to learn the mapping between two languages. In Figure 1, the network parameters are shared in our model. The forward flow is shown in orange while reverse flow is depicted in purple.

Although the two networks share parameters, they cannot do so directly as the required shapes of each layer differ. In order to perform backward translation, reverse flow is enabled in the network by explicitly taking the **transpose of each layer** in the network (we use fully connected layers without bias vectors) making the network bi-directional or invertible. With our cycle consistency loss formulation, the model learns layers such that the transpose of the layer inverts the network.

3 Experiments & Analysis

We experiment with the **MUSE** dataset (Conneau et al., 2017). It consists of 110 bilingual dictionaries with separate training and test datasets for each language pair. The pairs contain polysemous words. When it comes to training BDMA, polysemous words can provide additional context to the model being trained while handicapping other baseline models. We filter out training pairs for polysemous words (source or target). The models are trained with 5000 unique pairs. We show two sets of experiments: (a) with a filtered evaluation set that contains 1500 unique pairs and (b) with the original evaluation dataset. We measure the performance of BDMA on two sets of languages: the low-resource languages *Russian* (Ru) and *Japanese* (Ja), and the high-resource languages *Spanish* (Es), *French* (Fr), *German* (De) and *Italian* (It).

In each table, ***s*** is the source language while ***t*** is the target language. → indicates the direction of mapping and training language pairs used from MUSE. For **reverse translation**, the model is trained with the *t* → *s* dataset and evaluated on the *s* → *t* test dataset—for example, the model trained on **En→Ru** is evaluated on **Ru→En**. P@1 measurements highlighted in **blue show the forward (training) direction** in which the model is trained and its adjacent non-colored measurement uses the **same model** to perform **reverse** translation.

Method	Evaluation [$s \rightarrow t$] P@1							
	En→Es	Es→En	En→It	It→En	En→De	De→En	En→Fr	Fr→En
MUSE (Conneau et al., 2017)*	81.7	83.3	77.7	78.2	74.0	72.0	82.3	82.1
VECMAP (Artetxe et al., 2018)	80.80	85.20	77.47	80.47	73.33	75.07	81.60	84.40
GeoMM (Jawanpuria et al., 2019)	81.53	86.33	78.47	81.53	74.80	76.67	82.00	84.67
RCSLS (Joulin et al., 2018)*	84.1	86.3	78.5	79.8	**79.1**	76.3	83.3	84.1
BLISS(R) (Patra et al., 2019)*	**84.3**	86.2	**79.3**	82.4	**79.1**	76.6	83.9	84.7
Joint Align (Wang et al., 2019)**	69.6	71.9	-	-	68.7	70.7	78.0	79.2
Cross-lingual Anchoring (Ormazabal et al., 2020)**	84.2	86.5	-	-	78.1	76.9	**84.9**	**85.0**
LNMAP (LIN. AE) (Mohiuddin et al., 2020)*	82.9	86.4	78.1	81.4	75.5	75.9	83.9	84.7
Linear Mapping								
BDMA `[C + R]` ($s \rightarrow t$)	83.13	83.26	78.60	78.60	76.13	74.73	83.73	82.86
BDMA `[C + R]` ($t \rightarrow s$)	83.13	84.06	78.60	78.53	73.46	75.66	82.8	83.86
1-Hidden Layer Feedforward Network								
BDMA `[C + R]` ($s \rightarrow t$)	82.40	85.73	78.66	82.60	74.46	**78.40**	83.40	84.93
BDMA `[C + R]` ($t \rightarrow s$)	81.60	**86.80**	78.4	**82.66**	73.46	74.86	79.86	84.33

Table 1: **Bi-Directional Manifold Alignment (BDMA) measured with Precision @ 1** measured on filtered MUSE evaluation set **(with polysemous words) for high resource languages**. * represents results taken directly from the cited paper. ** represents results taken from Ormazabal et al. (2020). We consider the *best* results for each language pair and direction from this paper. **-** represents language pairs that are not part of experiments in the original paper. BLISS(R) (Patra et al., 2019) is semi-supervised.

Method	Evaluation [$s \rightarrow t$] P@1							
	En→Ru	Ru→En	En→Hi	Hi→En	En→Ja	Ja→En	En→Pt	Pt→En
VECMAP (Artetxe et al., 2016)	52.33	65.73	34.87	50.03	51.54	41.42	80.27	80.67
VECMAP (Artetxe et al., 2018)	51.53	70.00	**40.40**	**56.46**	46.95	44.25	80.60	82.93
GeoMM (Jawanpuria et al., 2019)	54.13	69.47	-	54.72	27.55	23.66	**81.60**	83.27
Linear Mapping								
BDMA `[C + R]` ($s \rightarrow t$)	55.80	68.66	36.80	54.58	53.59	38.52	80.40	**84.20**
BDMA `[C + R]` ($t \rightarrow s$)	55.60	69.73	36.60	55.25	53.52	38.73	80.06	83.93
1-Hidden Layer Feedforward Network								
BDMA `[C + R]` ($s \rightarrow t$)	**57.20**	**70.20**	37.00	54.11	**54.07**	46.51	80.13	83.13
BDMA `[C + R]` ($t \rightarrow s$)	56.93	70.06	37.00	54.38	54.28	**47.07**	80.06	83.93

Table 2: **Bi-Directional Manifold Alignment (BDMA) measured with Precision @ 1** on default MUSE evaluation set **(with polysemous words) for low-resource languages**.

Embeddings & Baselines. We use normalized and mean-centered FastText embeddings (Joulin et al., 2016), learned from language-specific Wikipedia. We train two types of translation models: (a) a linear mapping with a weight matrix $W \in \mathbb{R}^{d \times d}$ for a d-dimensional embedding, and (b) a 1 hidden layer feed forward network. For baseline comparisons, we retrain VECMAP (Artetxe et al., 2016, 2018), GeoMM (Jawanpuria et al., 2019) and RCSLS (Joulin et al., 2018). When possible, we compare with BLISS(R) (Patra et al., 2019), Joint Align (Wang et al., 2019), Cross-lingual Anchoring (Ormazabal et al., 2020) and LNMAP (Mohiuddin et al., 2020) using results previously reported for high resource languages. We train BDMA with a combination of cosine (`C`) and RCSLS (`R`) losses, and separate baseline methods for each language and translation direction pair.

3.1 Impact of Polysemy

In Table 1, we observe BDMA's performance translating words in high resource languages. BDMA's performance is better or equivalent in comparison to other methods. Additionally, we note that the translation model is trained with 5000 unique pairs while *Joint Align* (Wang et al., 2019) and *cross-lingual anchoring* (Ormazabal et al., 2020) are trained with the full MUSE training dataset for any given language pair which is greater than 5K.[2] Similarly, Table 2 shows the performance of different models on low resource languages compared to BDMA. BDMA with 1-H FFN performs better than a linear mapping with an overall increase as high as 2.82% while translating Japanese to English. The exception is for Hindi, where the performance drops by 3.8% (Hi → En). We see that the

---

[2] See Appendix C for the original dataset sizes.

Method	Evaluation [$s \to t$] P@1							
	En→Es	Es→En	En→It	It→En	En→De	De→En	En→Fr	Fr→En
MUSE (Conneau et al., 2017)	48.06	61.27	51.33	62.59	37.4	50.21	39.33	51.60
VECMAP (Artetxe et al., 2016)	48.87	61.49	52.07	62.31	38.60	50.22	47.47	59.10
VECMAP (Artetxe et al., 2018)	48.27	62.79	52.20	65.24	37.80	52.59	47.67	60.89
GeoMM (Jawanpuria et al., 2019)	48.60	63.79	52.53	65.38	38.33	53.45	48.60	**61.24**
RCSLS (Joulin et al., 2018)	49.26	**64.29**	**53.00**	**66.52**	38.93	**53.73**	47.66	59.74
Linear Mapping								
BDMA `[C + R]` ($s \to t$)	**49.40**	62.57	52.80	63.09	**39.33**	52.37	48.73	59.95
BDMA `[C + R]` ($t \to s$)	49.33	62.78	52.46	63.31	39.00	52.01	49.06	59.38
1-Hidden Layer Feedforward Network								
BDMA `[C + R]` ($s \to t$)	48.90	62.28	52.06	65.02	38.46	51.86	**48.86**	60.67
BDMA `[C + R]` ($t \to s$)	48.46	63.00	52.46	65.09	39.06	52.08	47.33	60.67

Table 3: **Bi-Directional Manifold Alignment (BDMA) measured with Precision @ 1** shows the performance of different models on **high-resource languages** in the MUSE dataset (Conneau et al., 2017) in comparison to *BDMA*. The test dataset contains **unique pairs** only.

Method	Evaluation [$s \to t$]							
	En→Ru	Ru→En	En→Hi	Hi→En	En→Ja	Ja→En	En→Pt	Pt→En
VECMAP (Artetxe et al., 2016)	35.27	52.28	23.80	26.45	39.07	35.59	44.93	61.06
VECMAP (Artetxe et al., 2018)	34.53	56.28	27.40	**31.21**	36.20	38.88	49.53	63.83
GeoMM (Jawanpuria et al., 2019)	36.93	**56.42**	**27.67**	30.35	21.60	21.81	**50.13**	63.90
RCSLS (Joulin et al., 2018)	37.73	54.85	24.80	26.80	39.40	**38.80**	49.06	**64.53**
Linear Mapping								
BDMA `[C + R]` ($s \to t$)	37.40	52.35	25.20	26.87	40.8	34.64	49.46	63.12
BDMA `[C + R]` ($t \to s$)	36.93	52.99	24.73	27.37	40.4	35.30	49.00	63.40
1-Hidden Layer Feedforward Network								
BDMA `[C + R]` ($s \to t$)	37.73	52.06	25.13	27.73	**40.93**	37.70	49.46	63.12
BDMA `[C + R]` ($t \to s$)	**38.40**	53.49	25.13	29.14	40.00	38.14	48.73	64.32

Table 4: **Bi-Directional Manifold Alignment (BDMA) measured with Precision @ 1** on unique MUSE evaluation set **(without polysemous words)** for **low-resource languages**.

model benefits from bidirectional training when there are polysemous words in the evaluation corpus, improving the network's ability to generalize.

3.2 Impact of Unique Vocabulary

Similar to the previous experiment, we analyze the impact of BDMA with an evaluation dataset of unique pairs for both high resource and low resource languages. In contrast to Table 1, Table 3 shows that both linear mapping and 1-H neural network are comparable to other baselines (except RCSLS) when there are no polysemous words. Adding additional layers to the network does not provide any benefit, which is consistent with findings from Søgaard et al. (2018) and Ruder et al. (2019) that a linear mapping performs well for these language pairs. Table 4 details experiments for the same under low resource language conditions. Although BDMA performs better for En → Ru and En → Ja, Hi → En continues to perform poorly. In contrast, its performance is comparable for *Portuguese* where the reduction is 1.13% (En → Pt) only.

Therefore, the 2 main benefits of BDMA are: (a) it creates a single bidirectional word translation model while keeping the performance of the model comparable to baseline, and (b) the 1-H FFN is a single network in comparison to LNMAP (which has 3), while Linear BDMA has the same number of parameters as all other methods in Table 1 and 2.

3.3 Importance of Training Direction

If the filtered training pairs do not contain polysemous words, why is the training direction important? This is because when the model is trained for a number of epochs, its optimal savepoint is chosen based on the forward translation performance for the given language pair direction. As seen in Table 2 and 4, the direction chosen to start model training can have an impact of forward and reverse translation performance. For example, the model training with *Ru → En* performs better than *En → Ru*.

Loss	Evaluation [$s \to t$] P@1			
	En→Ru	**Ru→En**	**En→Ja**	**Ja→En**
Linear Mapping ($s \to t$)				
`[M]`	54.66	66.26	21.93	16.26
`[C]`	55.00	66.46	52.22	40.38
`[R]`	51.80	67.53	38.51	39.53
`[C + R]`	55.80	68.66	53.59	38.52
Linear Mapping ($t \to s$)				
`[M]`	53.66	65.20	21.86	16.54
`[C]`	54.93	66.46	52.15	40.38
`[R]`	52.13	67.93	38.58	39.49
`[C + R]`	55.60	69.73	53.52	38.73
1-Hidden Layer Feedforward Network ($s \to t$)				
`[M]`	51.40	63.66	20.28	12.19
`[C]`	55.00	65.86	52.84	40.38
`[R]`	52.26	68.80	49.96	47.48
`[C + R]`	**57.20**	**70.20**	**54.07**	46.51
1-Hidden Layer Feedforward Network ($t \to s$)				
`[M]`	49.80	64.26	17.95	18.26
`[C]`	54.06	66.00	53.80	40.45
`[R]`	52.46	68.60	49.48	47.07
`[C + R]`	56.93	70.06	54.28	**47.07**

Table 5: **Precision @ 1 of BDMA with different losses.** An ablation study of the impact of different loss combinations while training a model with BDMA. `[M]` = MSE, `[C]` = cosine, `[R]` = RCSLS and `[C + R]` = cosine + RCSLS loss.

Ablation Study. In Table 5, we assess the impact of using (combinations of) MSE, cosine and RCSLS distance functions $\mathcal{D}$. A combined cosine and RCSLS loss (`[C + R]`) performs the best and provides consistent forward ($s \leftarrow t$) and reverse translation ($t \leftarrow s$) performance (within 0.5%).

4 Related Work

Over the years, many supervised methods have been proposed. Irvine and Callison-Burch (2013) learn a binary classifier for a language pair that predicts if a given word pair is a translation of each other or not. Artetxe et al. (2016) implement Procrustes alignment while normalizing and mean centering word embeddings. Xing et al. (2015) add an orthogonal loss while aligning manifolds. In Artetxe et al. (2018), additional pre- and post-processing steps are provided. Conneau et al. (2017) propose a new retrieval method called cross-domain similarity local scaling (CSLS) in order to reduce the "hubness" problem. Joulin et al. (2018) convert CSLS into a loss objective in order to optimize the translation matrix. An important challenge with linear mapping is that it assumes that source and target languages have a similar manifold structure; Søgaard et al. (2018) show this assumption is not true for many language pairs. Nakashole and Flauger (2018) show that transformations need to be non-linear and are dependent on the word's local neighborhood. Instead of learning a mapping between languages separately, Wang et al. (2019) jointly learn the monolingual and cross-lingual embeddings for the given language pair. Ormazabal et al. (2020) extend skip-gram to project source embeddings into a fixed target space and using them as anchors to iteratively learn the mapping.

Cyclic Loss for Reverse Translation. Xu et al. (2018) perform unsupervised word alignment using the cycle consistency loss while computing the sinkhorn distance between a forward and reverse translation network. Mohiuddin and Joty (2019) train a dual autoencoder-discriminator architecture and use a cyclic loss to train a bi-directional model. LNMAP (Mohiuddin et al., 2020) extends the autoencoder architecture with a 2 layer mapping to learn a non-isomorphic mapping between languages. Our work differs as we reduce the number of parameters in the model (as it contains the mapping only) while training an **invertible network** that can perform both forward and back translation.

5 Conclusion

We show how a non-linear mapping (invertible neural network) can be trained with a cyclic consistency loss, showing that a common isomorphic assumption is not strictly necessary (Søgaard et al., 2018). The network trained has fewer parameters in comparison to Mohiuddin et al. (2020) while providing equivalent or improved performance on the low-resource word translation task.

Acknowledgements

We thank our anonymous reviewers for their constructive reviews and the UMBC HPCF facility for computational resources. This work is in part supported by AFRL, DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of the Air Force Research Laboratory (AFRL), DARPA, or the U.S. Government.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 518–523, Atlanta, Georgia. Association for Computational Linguistics.

Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, and Bamdev Mishra. 2019. Learning multilingual word embeddings in latent metric space: a geometric approach. *Transactions of the Association for Computational Linguistics*, 7:107–120.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. *arXiv preprint arXiv:1804.07745*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tasnim Mohiuddin, M Saiful Bari, and Shafiq Joty. 2020. Lnmap: Departures from isomorphic assumption in bilingual lexicon induction through non-linear mapping in latent space. *arXiv preprint arXiv:2004.13889*.

Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting adversarial autoencoder for unsupervised word translation with cycle consistency and improved training. *arXiv preprint arXiv:1904.04116*.

Ndapa Nakashole and Raphael Flauger. 2018. Characterizing departures from linearity in word translation. *arXiv preprint arXiv:1806.04508*.

Aitor Ormazabal, Mikel Artetxe, Aitor Soroa, Gorka Labaka, and Eneko Agirre. 2020. Beyond offline mapping: Learning cross lingual word embeddings through context anchoring. *arXiv preprint arXiv:2012.15715*.

Barun Patra, Joel Ruben Antony Moniz, Sarthak Garg, Matthew R Gormley, and Graham Neubig. 2019. Bilingual lexicon induction with semi-supervision in non-isometric embedding spaces. *arXiv preprint arXiv:1908.06625*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Cem Safak Sahin, Rajmonda S Caceres, Brandon Oselio, and William M Campbell. 2017. Consistent alignment of word embedding models. *arXiv preprint arXiv:1702.07680*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. *arXiv preprint arXiv:1805.03620*.

Zirui Wang, Jiateng Xie, Ruochen Xu, Yiming Yang, Graham Neubig, and Jaime Carbonell. 2019. Cross-lingual alignment vs joint training: A comparative study and a simple unified framework. *arXiv preprint arXiv:1910.04708*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018. Unsupervised cross-lingual transfer of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2474, Brussels, Belgium. Association for Computational Linguistics.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Appendix

In the following sections, we provide information about hyperparameter values for each network architecture, statistics about the dataset and results from additional experiments. The experiments are conducted on a NVIDIA K20 GPU with ≈ 4GB of RAM and NVIDIA V100 GPUs with 16GB of RAM. Each model is trained on a single GPU. Linear models can be trained on K20s and the larger 1-H-FFN are optimized on V100s.

A Hyperparameters

Following are the hyper-parameters used in our experiments:

Hyper-parameter	**Value**
`batch size`	128
`lr_decay`	0.98
`lr_shrink`	0.5
`map_beta`	0.001
`max_vocab`	200000

Table A1: Hyperparameters for BDMA experiments.

As seen in table A1, the maximum vocabulary (`max_vocab`) size is $200K$. The vocabulary is selected by taking $200K$ words that have the highest frequency. `map_beta` is the parameter that controls the contribution of the orthogonal loss to the overall loss function. The network is trained with an Adam optimizer (Kingma and Ba, 2014) having a learning rate of 0.0005. The word embeddings are preprocessed i.e. they are normalized and centered. The 1 hidden layer feedforward network used to perform alignment has a hidden layer size of 4096. The activation function of the hidden layer is `tanh`.

B CCL Correlation with Linear Mapping

As observed in equation 2.1, a linear relationship between source and target language embeddings can be learned by minimizing the squared loss between them. Although, in practice, an additional orthogonal constraint $\mathcal{L}_{ortho} = WW^T - I$ is added (Xing et al., 2015) as shown in the equation below:

$$\mathcal{L}_{\text{mse}} = \sum_{i \in V^p} \overbrace{\|f_a(m_i^s) - m_i^t\|_2^2}^{\mathcal{L}_{\text{mse}}^i} + (WW^T - I), \tag{3}$$

Minimizing $\mathcal{L}_{ortho}$ makes the linear mapping *implicitly* bidirectional able to map words from the target to source language. In comparison, $\mathcal{L}_{ccl}$ in equation 1 trains a non-linear neural network or linear mapping to be **explicitly** bidirectional. Thus $\mathcal{L}_{ccl}$ can be considered as an extension of $\mathcal{L}_{ortho}$.

Target Language	**Train**	**Test**
French	10872	2943
German	14677	3660
Italian	9657	2585
Spanish	11977	2975
Russian	10887	2447
Hindi	8704	2032
Japanese	7135	1799
Portuguese	11185	2827

Table A2: **MUSE Dictionary Size**. The table shows the *target* language, the number of pairs in the training and pairs present in the test dictionary where the *source* language is *English*.

Source Language	**Train**	**Test**
French	8270	2342
German	10866	2827
Italian	7364	2102
Spanish	8667	2416
Russian	7452	2069
Hindi	8001	1963
Japanese	6819	1952
Portuguese	7582	2148

Table A3: The table shows the *source* language, the number of pairs in the training and pairs present in the test dictionary where the *target* language is *English*.

C Dataset

MUSE (Conneau et al., 2017). As described in §3, the dataset has 110 bilingual dictionaries and contains pairs with *English* being the source or target language. Additionally, *Non-English* language pairs are available for European languages that includes *German*, *Spanish*, *French*, *Italian* and *Portuguese*. Each bilingual dataset has a vocabulary of 5000 unique source language words to train the translation model and 1500 unique words to evaluate them. Because the pairs are not unique and contain polysemous source words (the target word is always unique), the overall size of training and test dictionaries is greater than 5000 and 1500.

Tables A2 and A3 show the dataset size from the original MUSE dataset. The tables show that samples for different language pairs contain polysemous words that expand dataset size by 36.8% to 123.7% in comparison to BDMA (in table 2, 4 and 1) that is trained with 5000 unique pairs only.

D Additional Loss

In §2.1, we showcased how MSE is adapted for $\mathcal{L}_{\text{ccl}}$. Similarly, cosine and Relaxed CSLS loss can be modified for BDMA too. In an adapted version of cosine loss, we minimize the following:

$$\min_{\theta_f} \sum_{i \in V^p} (1 - |f_a(m_i^s) \cdot m_i^t|) + (1 - |f_b(m_i^t) \cdot m_i^s|)$$

In order to modify RCSLS (Joulin et al., 2018), we first take look at CSLS (Conneau et al., 2017) criteria for retrieval:

$$\begin{aligned} &\text{CSLS}(m_i^s, m_i^t) \\ &\quad = -2\cos(m_i^s, m_i^t) \\ &\qquad + \frac{1}{\text{k}} \sum_{m_j^t \in \mathcal{N}^t(\mathcal{W} \cdot m_i^s)} \cos(\mathcal{W} m_i^s, m_j^t) \\ &\qquad + \frac{1}{\text{k}} \sum_{m_j^s \in \mathcal{N}^s(\mathcal{W}^{\mathcal{T}} \cdot m_i^t)} \cos(m_j^s, \mathcal{W}^{\mathcal{T}} m_i^t)) \end{aligned} \tag{4}$$

where $\mathcal{N}^s(x)$ is the neighborhood of x in the source manifold and $\mathcal{N}^t(y)$ is the same in the target, k is the number of nearest neighbors and $\mathcal{W}$ is assumed to be orthogonal. Joulin et al. (2018) relax the cosine criteria in RCSLS i.e. $\cos(\mathcal{W} m_i^s, m_i^t) = {m_i^s}^{\mathcal{T}} \mathcal{W}^{\mathcal{T}} m_i^t$. Hence RCSLS becomes:

$$\begin{aligned} &\text{RCSLS}(m_i^s, m_i^t) \\ &\quad = -2{m_i^s}^{\mathcal{T}} \mathcal{W}^{\mathcal{T}} m_i^t \\ &\qquad + \frac{1}{\text{k}} \sum_{m_j^t \in \mathcal{N}^t(\mathcal{W} \cdot m_i^s)} {m_i^s}^{\mathcal{T}} \mathcal{W}^{\mathcal{T}} m_j^t \\ &\qquad + \frac{1}{\text{k}} \sum_{m_j^s \in \mathcal{N}^s(\mathcal{W}^{\mathcal{T}} \cdot m_i^t)} {m_j^s}^{\mathcal{T}} \mathcal{W}^{\mathcal{T}} m_i^t \end{aligned} \tag{5}$$

In BDMA, we replace the orthogonal matrix $\mathcal{W}$ with a mapping that is either linear or non-linear (neural network). RCSLS changes to:

$$\begin{aligned} \text{RCSLS}(m_i^s, m_i^t) &= -2 f_a(m_i^s) m_i^t \\ &+ \frac{1}{\text{k}} \sum_{m_j^t \in \mathcal{N}^t(f_a(m_i^s))} f_a(m_i^s) m_j^t \\ &+ \frac{1}{\text{k}} \sum_{m_j^s \in \mathcal{N}^s(f_b m_i^t)} m_j^s f_b(m_i^t) \end{aligned} \tag{6}$$

In equation 6, f_a and f_b are the forward and reverse flow projections of m_i^s and m_i^t respectively.

Probabilistic Graph Reasoning for Natural Proof Generation

Changzhi Sun[†*], **Xinbo Zhang**[†*], **Jiangjie Chen**[†§], **Chun Gan**[†¶],
Yuanbin Wu[‡], **Jiaze Chen**[†], **Hao Zhou**[†], **and Lei Li**[†]

[†]ByteDance AI Lab
[§]School of Computer Science, Fudan University
[¶]Math Department, University of Wisconsin–Madison
[‡]School of Computer Science and Technology, East China Normal University
`{sunchangzhi,zhangxinbo.freya}@bytedance.com`
`{chenjiaze,zhouhao.nlp,lileilab}@bytedance.com`
`jjchen19@fudan.edu.cn, cgan5@wisc.edu, ybwu@cs.ecnu.edu.cn`

Abstract

In this paper, we investigate the problem of reasoning over natural language statements. Prior neural based approaches do not explicitly consider the inter-dependency among answers and their proofs. In this paper, we propose PRobr, a novel approach for joint answer prediction and proof generation. PRobr defines a joint probabilistic distribution over all possible proof graphs and answers via an induced graphical model. We then optimize the model using variational approximation on top of neural textual representation. Experiments on multiple datasets under diverse settings (fully supervised, few-shot and zero-shot evaluation) verify the effectiveness of PRobr, e.g., achieving 10%-30% improvement on QA accuracy in few/zero-shot evaluation. Our codes and models can be found at https://github.com/changzhisun/PRobr/.

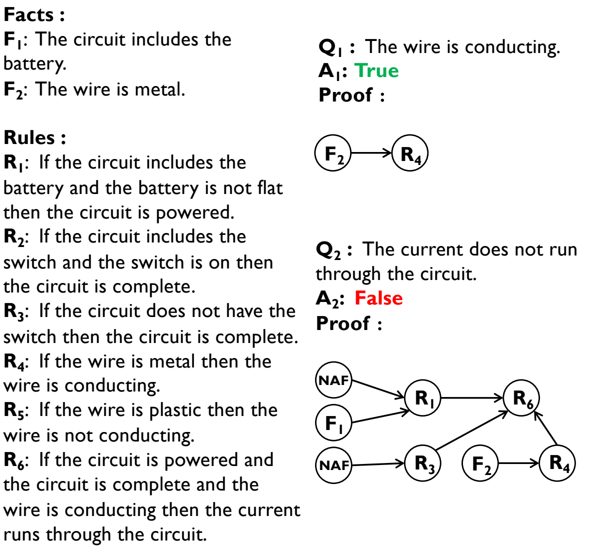

Facts :
F_1: The circuit includes the battery.
F_2: The wire is metal.

Rules :
R_1: If the circuit includes the battery and the battery is not flat then the circuit is powered.
R_2: If the circuit includes the switch and the switch is on then the circuit is complete.
R_3: If the circuit does not have the switch then the circuit is complete.
R_4: If the wire is metal then the wire is conducting.
R_5: If the wire is plastic then the wire is not conducting.
R_6: If the circuit is powered and the circuit is complete and the wire is conducting then the current runs through the circuit.

Q_1 : The wire is conducting.
A_1: True
Proof :

Q_2 : The current does not run through the circuit.
A_2: False
Proof :

Figure 1: An example of reasoning over natural language statements. The goal is to predict the answer (true/false) and generate the proof graph.

1 Introduction

Automatic reasoning over explicitly provided knowledge has been a persistent goal of AI (Newell and Simon, 1956; McCarthy et al., 1960). Early approaches focus on reasoning over formal (logical or probabilistic) representations. However, automatically constructing and reasoning over formal representations remain challenging. To bypass these challenges, in this work, we investigate reasoning over natural language statements instead of formal representations.

Given a set of facts and rules and a query (expressed in natural language), we aim to predict the answer and provide proof to prove or disprove the query. For example, in Figure 1, there are two facts, six rules and two queries, each of which is expressed by natural language. To predict the true/false of each query, starting from the facts, we need to reason deductively by applying given rules until we can derive the truth value of the query. The process of deduction can be represented as a graph, whose node is either a fact, rule or special NAF node (explained in the Section 2.1). Generating answer and proof together makes a system easier to interpret and diagnose.

Recent work by PRover (Saha et al., 2020) first explored this problem through two modules: question answering and proof generation. It trains these two modules through implicit parameter sharing, and then uses integer linear programming (ILP) to enforce consistency constraints (only test time). It is difficult to ensure that the proof generation module contributes to the question answering module, because the proof is not explicitly involved in the answer prediction. Parameter sharing becomes more limited under few/zero-shot settings, as demonstrated in our experiments. We expect the proof to enhance the capability of question answering, especially under few/zero-shot settings. One

---

[*] Equal contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3140–3151
August 1–6, 2021.

promising solution is to explicitly exploit more interaction between question answering and proof generation.

In this paper, we propose PRobr, a novel **prob**abilistic graph **r**easoning framework for joint question answering and proof generation. PRobr defines a joint distribution over all possible proof graphs and answers with an undirected *probabilistic graphical model* (PGM). It directly characterizes the interaction between proofs and answers. PGMs generally incur intractable learning and inference for the complex graph (Koller and Friedman, 2009). For example, computing normalization constant in PGMs using traditional probabilistic propagation algorithm (e.g.sum-product algorithm (Kschischang et al., 2001)) requires large time complexity. Therefore, we propose a variational approach to maximize the pseudolikelihood of joint distribution to optimize the model more efficiently. First, a variational distribution was introduced based on mean-field assumption. Then we maximize the pseudolikelihood of joint distribution given the output of variational distribution. At the same time, we align these two distributions using the training data. PRobr can be efficiently trained by stochastic gradient descent. Our contributions are summarized as follows[1]:

- We propose PRobr for joint question answering and proof generation, which defines a joint distribution over all possible proofs and answers with an undirected PGM to capture more dependencies.

- We present an efficient variational approximation method to learn PRobr.

- Experiments on several datasets verify the effectiveness of PRobr under multiple settings (supervised, few-shot, and zero-shot evaluation).

2 Task Definition

To reason over natural language statements, we design to answer the query and generate corresponding proof generation jointly. Figure 1 shows an example. Given a declarative query Q, and given relevant facts and rules (expressed in natural language), the task aims to predict the answer A (true/false) to the query Q based on the closed-world assumption (described in 2.1). Meanwhile, it generates a proof P (described in 2.2) to prove or disprove Q.

2.1 Semantics

We adopt the semantics of Datalog (Ceri et al., 1989) in this work. Following prior work (Clark et al., 2020), we make a closed-world assumption (CWA), which means a fact is true if it can be deduced based on a given context, and any fact not provable is assumed false. And we use negation as failure (NAF) (Clark, 1978), a rule of inference which allows one to deduce that `NOT S` is true if all possible proofs of a statement `S` fail. For example, in Figure 1, the `NAF` node before $\mathtt{R}_3$ represents "the circuit does not have the switch". Note that under this semantics, negative facts and negative rules are not allowed because of redundancy under the CWA assumption.

2.2 Formulations

A proof is a directed acyclic graph (Figure 1). Each node is either a fact, rule or special `NAF` node. Each edge directs from either a fact (or `NAF`) to a rule or a rule to another rule, which indicates that a fact is consumed by a rule, or another rule consumes a rule, respectively. For simplicity, let context $C = \{s_1, \ldots, s_n\}$ denote the collection of sentence, each of which is a fact or rule.

Proof Formulation We assign an indicator variable (0/1) for each possible node and edge, to vectorize the structure of a given proof P. Specifically, we introduce the indicator variables $V = \{V_i\}_{i=1}^{n}$ for each element s_i in the context C, and an indicator variable $E = \{E_{ij}\}_{i,j=1}^{n}$ $(i \neq j)$ for a possible edge connecting from node s_i to node s_j, where:

- $V_i = 1$ indicates s_i is in the proof P, while $V_i = 0$ means s_i is absent.

- $E_{ij} = 1$ indicates there is an edge directing from s_i to s_j, while $E_{ij} = 0$ means s_i cannot direct to s_j in the proof by an edge.

In addition, we assign a binary answer variable A to indicate the true value of the query. Figure 2a shows a simplified example, where context $C = \{s_1, s_2, s_3\}$, and the query can be decided as true by a very simple proof, consisting of only two nodes (s_1 and s_3) and a single edge (from s_1 to s_3). The proof can be represented by the following variables: $A = V_1 = V_3 = E_{13} = 1, V_2 = E_{12} = E_{21} = E_{23} = E_{31} = E_{32} = 0$.

---

[1] Our codes and models can be found at https://github.com/changzhisun/PRobr/.

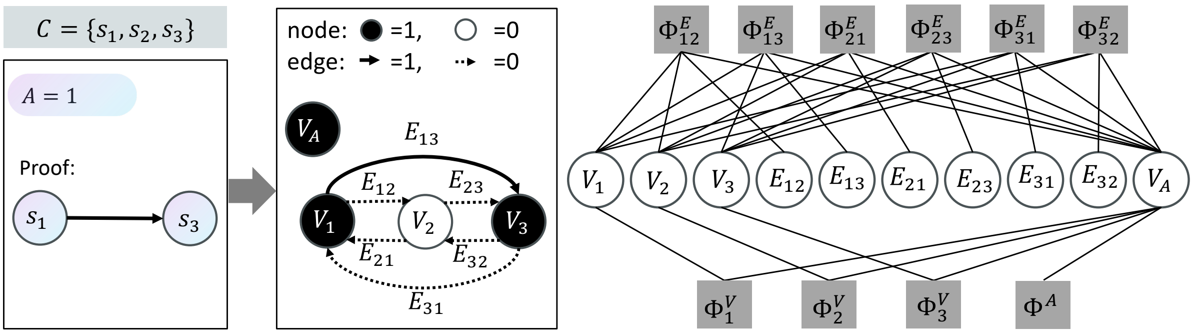

(a) Proof graph and its induced random variables.

(b) Factor graph induced by the proof graph.

Figure 2: Joint probabilistic distribution by assigning indicator variables for the answer and proof. The solid circles and lines in 2a indicate that these corresponding statements and edges are in the final proof graph.

3 Approach

We introduce the proposed framework PROBR, which jointly provides the answer to the given query over natural contexts and generates corresponding proof. Different from PROVER that makes independence assumption, PROBR can capture more dependencies between the proof and answer. PROBR defines a joint distribution over all possible proofs and answers with an undirected graphical model (Section 3.1), and we use neural networks to parameterize each component (Section 3.2). To optimize PROBR efficiently, we adopt a variational approach to maximize the pseudolikelihood of joint distribution (Section 3.3). Finally, we introduce the strategy during inference (Section 3.4).

3.1 Overview

We start by formalizing joint question-answering module and proof-generation module in a probabilistic way. We clarify some notations as follows:

- A context $C = \{s_1, \ldots, s_n\}$, s_i is a sentence.
- A query Q.
- An answer variable A, it can take any value a in $\{0, 1\}$.
- Node variables $\mathcal{V} = \{V_i\}_{i=1}^{n}$, each V_i can take any value v_i in $\{0, 1\}$.
- Edge variables $\mathcal{E} = \{E_{ij}\}_{i,j=1}^{n}$ $(i \neq j)$, each E_{ij} can take any value e_{ij} in $\{0, 1\}$.
- Let $Y \triangleq (A, \mathcal{E}, \mathcal{V})$ denote all output variables.

In our notation, we use uppercase letters for variables (e.g., Y, A, V_i, E_{ij}) and lowercase letters for variables that take values (e.g., y, a, v_i, e_{ij}).

Given a context C and a query Q, PROBR tries to assign true/false values for all variables, including answer variable A, node variables V and edge variables E. We define a joint distribution over all possible Y, officially denoted as $p(Y)$: [2]

$$p(Y = y) \propto \Phi^A(a) \prod_i \Phi_i^V(v_i, a) \prod_{i,j} \Phi_{ij}^E(v_i, v_j, e_{ij}, a) \quad (1)$$

Different from PROVER that makes independent assumption, such a factorization of Equation 1 can characterize the interaction between the variables V_i, V_j, E_{ij} and A. Figure 2b shows the factor graph of joint distribution $p(Y)$ for the example in Figure 2a. Theoretically, when we have the ground truth y^*, [3] we can minimize the following objective:

$$\mathcal{L}_{\text{joint}} = -\log p(Y = y^*) \quad (2)$$

However, the normalization constant of $p(Y)$ is hard to calculate due to high-order factors of large size (RHS of Equation 1). In this paper, we provide a variational-based solution to optimize objective $\mathcal{L}_{\text{joint}}$ (Section 3.3).

3.2 Parameterization

We use neural networks to parameterize each potential function of Equation 1: Φ^A, Φ_i^V and Φ_{ij}^E.

Text Representation Network Given a context C and a query Q, to obtain a contextual representations, we use RoBERTa (Liu et al., 2019) as our backbone network following (Clark et al., 2020; Saha et al., 2020). The input to RoBERTa is the concatenation of C and Q, separated by `[SEP]` tokens, denoted as: `[CLS]`, C, `[SEP]`, `[SEP]`, Q, `[SEP]`.

---

[2] We drop the input variables for clarity.

[3] We use $*$ to indicate the ground truth in the text.

Potential Function for the Answer (Φ^A) After the RoBERTa encoding, we can get the global representation of the entire input through the first token [`CLS`], denoted as $h_{[\texttt{CLS}]}$. To score the possible values of variable A, i.e. 0 or 1, we use a multilayer perceptron (MLP) as a nonlinear transformation:

$$\begin{bmatrix} \Phi^A(A=0) \\ \Phi^A(A=1) \end{bmatrix} = \mathrm{MLP}_1(h_{[\texttt{CLS}]}) \in \mathbb{R}^2$$

Potential Function for Statements (Φ_i^V) For each sentence s_i (a fact or a rule), we compute the sentence representation h_{s_i} by performing a mean pool of the all token representation based on the output of RoBERTa. It is worth noting that `NAF` is a special fact, we calculate $h_{\texttt{NAF}}$ through linear transformation on $h_{\texttt{CLS}}$. To score the possible values of variables (V_i, A), we also use another MLP as a score function:

$$\begin{bmatrix} \Phi_i^V(V_i=0, A=0) \\ \Phi_i^V(V_i=0, A=0) \\ \Phi_i^V(V_i=0, A=0) \\ \Phi_i^V(V_i=1, A=1) \end{bmatrix} = \mathrm{MLP}_2(h_{s_i}) \in \mathbb{R}^4,$$

where the dimension 4 indicates the number of possible values for the combination of variables V_i and A. We share the parameters of MLP_2 across all sentences.

Potential Function for Statement Relations (Φ_{ij}^E) For each sentence pair (s_i, s_j), we obtain the sentence pair representation h_{s_i,s_j}, by concatenating h_{s_i} and h_{s_j} with their element-wise difference (directionality). To score four variables (V_i, V_j, E_{ij}, A) simultaneously, similarly, we use a new MLP as score function:

$$\begin{bmatrix} \Phi_{ij}^E\begin{pmatrix} V_i=0, V_j=0, \\ E_{ij}=0, A=0 \end{pmatrix} \\ \vdots \\ \Phi_{ij}^E\begin{pmatrix} V_i=1, V_j=1, \\ E_{ij}=1, A=1 \end{pmatrix} \end{bmatrix} \begin{matrix} = \mathrm{MLP}_3(h_{s_i,s_j}) \\ \in \mathbb{R}^{16}, \end{matrix}$$

$$h_{s_i,s_j} = h_{s_i} \oplus h_{s_j} \oplus (h_{s_i} - h_{s_j}),$$

where $\oplus$ is the vector concatenation, and the dimension 16 indicates the number of possible values for the combination of four variables (V_i, V_j, E_{ij}, A). We also share the parameters of MLP_3 across all sentence pairs.

3.3 Learning the Model

To tackle the challenge of optimizing $\mathcal{L}_{\text{joint}}$ (Equation 2), we adopt the widely used pseudolikelihood as an alternative objective for optimization (Richardson and Domingos, 2006), bypassing the calculation of the normalization constant.

Pseudolikelihood Given a set of variable Y, the pseudolikelihood of Y is defined as:

$$p_{\text{pseduo}}(Y) = \prod_{y \in Y} p(y|Y_{-y}) =$$
$$p(A|\mathcal{E}, \mathcal{V}) \prod_i p(V_i|Y_{-V_i}) \prod_{i,j} p(E_{ij}|Y_{-E_{ij}})$$

When we have the ground truth y^*, we can minimize the following objective:

$$\mathcal{L}_{\text{pseudo}} = -\log p_{\text{pseudo}}(Y = y^*)$$

However, it is difficult to decode the optimal assignments based on the pseudolikelihood (Equation 3). There is a rich body of literature on how to decoding in a sampling way (Chapter 12 (Salakhutdinov, 2014)). In this paper, however, we choose a modern approach using variational approximation.

Variational Approximation We approximate pseudolikelihood of Y with a mean-field (Opper and Saad, 2001) variational distribution $q(Y)$, in which $y \in Y$ is independent of each other. Similarly, we parameterize each independent distribution with a neural network. Formally, $q(Y)$ is formulated as below:

$$q(Y) = q(A) \prod_i q(V_i) \prod_{i,j} q(E_{ij}),$$

$$\begin{bmatrix} q(A=0) \\ q(A=1) \end{bmatrix} = \mathrm{Softmax}\left(\mathrm{MLP}_4(h_{[\texttt{CLS}]})\right) \in \mathbb{R}^2,$$

$$\begin{bmatrix} q(V_i=0) \\ q(V_i=1) \end{bmatrix} = \mathrm{Softmax}\left(\mathrm{MLP}_5(h_{s_i})\right) \in \mathbb{R}^2,$$

$$\begin{bmatrix} q(E_{ij}=0) \\ q(E_{ij}=1) \end{bmatrix} = \mathrm{Softmax}\left(\mathrm{MLP}_6(h_{s_i,s_j})\right) \in \mathbb{R}^2.$$

Once the variational distribution $q(Y)$ is obtained, it can provide conditions for pseudolikelihood $p(y|Y_{-y})$, thus avoiding the sampling process to obtain the optimal assignments. In the optimization process, we adopt the simple strategy to update the parameters of p and q.

- For node and edge variables, we optimize

$$\mathcal{L}_{\text{node}} = -\sum_i \log q(V_i = v_i^*),$$

$$\mathcal{L}_{\text{edge}} = -\sum_{i,j} \log q(E_{ij} = e_{ij}^*).$$

- For the answer variable, we optimize

$$\mathcal{L}_{\text{qa}} = -\log p(A = a^*|\hat{\mathcal{E}}, \hat{\mathcal{V}}),$$

where $\hat{\mathcal{E}} = \{\hat{e}_{ij}\}, \hat{\mathcal{V}} = \{\hat{v}_i\}$ are the predictions of variational model [4].

The final objective is to minimize:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{qa}} + \mathcal{L}_{\text{node}} + \mathcal{L}_{\text{edge}}$$

Overall, PROBR is a mixture of independent (variational) model and undirected graphical model through some reasonable approximations. Our final optimized distribution can be decomposed as directed graphical model $q(\mathcal{V})q(\mathcal{E})p(A|\mathcal{E}, \mathcal{V})$, where $q(\mathcal{V}), q(\mathcal{E})$ adopts the independent factorized probability, and $p(A|\mathcal{E}, \mathcal{V})$ is implied by the undirected graphical model (Equation 1). In this way, PROBR enjoys the advantage of global normalization (undirected graphical model) and is easier to optimize (directed graphical model).

Discussion Another way to achieve consensus between $q(Y)$ and $p_{\text{pseudo}}(Y)$ is to directly optimize the KL divergence:

$$\mathcal{L}_{\text{kl}} = \sum_{y \in Y} \text{KL}\left(q(y)||p(y|Y_{-y})\right)$$

However, $\mathcal{L}_{\text{kl}}$ does not bring any improvement for supervised learning (Section 4.6), hence we exclude it during training. PROBR can be easily extended to semi-supervised learning scenario by using this $\mathcal{L}_{\text{kl}}$ term. Specifically, minimize the $\mathcal{L}_{\text{final}}$ for the labeled data; and minimize the $\mathcal{L}_{\text{kl}}$ for the unlabeled data. We save this for future work.

3.4 Inference

After training, for nodes and edges, we choose the predictions of the variational model, and for answers, we choose the prediction of the joint model based on the output of variational model. In addition, we also employ the Integer Linear Programming (ILP) to enforce consistency constraints following (Saha et al., 2020).

4 Experiments

To evaluate the effectiveness and generality of our PROBR model, we conduct both fully supervised learning, few-shot learning, and zero-shot learning over several datasets[5] against two baselines: RuleTakers and PROVER[6].

4.1 Datasets and Metrics

We use three datasets (DU0-DU5, Birds-Electricity, ParaRules) introduced by (Clark et al., 2020).

DU0-DU5 DUd (d=0,1,2,3,5) are five synthetic datasets, each containing 100k queries with theories expressed in templated English, proof graphs expressed in natural language, and answers described as *True/False*. Answers require reasoning up to depth d for queries in DUd.

Birds-Electricity This dataset is a test-only dataset of 5k samples in total. It describes birds and electric circuit, which was used to evaluate the out-of-distribution performance of the models.

ParaRules ParaRules is a dataset generated and paraphrased from sampled theories (facts + rules). It contains 40k queries against ≈2k theories, where the original templated English facts and rules are creatively paraphrased into more diverse natural language by crowdsourcing. For example, the fact "Dave is cold" can be rephrased as"After Dave got wet in the rain, he feels cold"; the rule "If someone is nice then they are young" can be rephrased into "A person described as being nice will certainly be young". Different from DUd and Birds-Electricity dataset composed of synthetic language, ParaRules can better test models' reasoning ability over human-like language.

Metrics We evaluate the performance considering both answers and proofs. For answers, we evaluate the **QA Accuracy** (**QA**). For proofs, we evaluate the **Proof Accuracy** (**PA**), and PA refers to the fraction of examples where generated proof matches exactly with the gold proof. We also report **Full Accuracy** (**FA**) to denote the faction of examples where both the answer and the proof are exactly correct.

4.2 Fully Supervised Learning

For the supervised setting, we train PROBR on the training split of the DU5 dataset with gold answer

---

[4] We also experiment with the gold proof to optimize $\mathcal{L}_{\text{qa}}$ (Section 4.6).

[5] https://allenai.org/data/ruletaker

[6] Please refer to supplementary materials for our hyper-parameter and computing infrastructure. For RuleTakers and PROVER, we directly adopt results reported in their papers if exist, and for extra setting beyond papers, we reproduce the baselines using provided codes and parameters:https://github.com/swarnaHub/PRover.

D	**Cnt**	**QA**			**PA**		**FA**	
		RT	PV	PB	PV	PB	PV	PB
0	6299	100	100	**100**	98.4	**98.4**	98.4	**98.4**
1	4434	98.4	99.0	**99.9**	93.2	**94.3**	93.1	**94.3**
2	2915	98.4	98.8	**99.9**	84.8	**86.1**	84.8	**86.1**
3	2396	98.8	99.1	**100**	80.5	**82**	80.5	**82**
4	2134	99.2	98.8	**100**	72.5	**76.1**	72.4	**76.1**
5	2003	99.8	99.3	**100**	65.1	**72.2**	65.1	**72.2**
All	20192	99.2	99.3	**99.9**	87.1	**88.8**	87.1	**88.8**

Table 1: Fully supervised learning performance compared among RuleTakers (RT), PROVER (PV) and PROBR (PB) on test split of DU5 after training on training split of DU5, reported in varying depth.

Train Data		**QA**		**PA**		**FA**	
		PV	PB	PV	PB	PV	PB
	100%	99.3	**99.9**	87.1	**88.8**	87.1	**88.8**
RC	10%	94.5	**99.9**	**63.6**	60.4	**63.3**	60.4
	5%	80.6	**99.7**	34.0	**44.2**	32.1	**44.2**
	1%	70.2	**88.2**	20.0	**21.6**	15.1	**20.3**
RQ	30k	97.8	**99.9**	72.5	**86.8**	72.4	**86.8**
	10k	87.1	**99.9**	44.0	**72.4**	42.7	**72.3**
	1k	51.3	**82.1**	**28.0**	21.1	15.0	**18.4**

Table 2: Few-shot performance comparison among PROVER and PROBR on test split of DU5 after training on partial DU5 samples. (Two types of training samples, RC: queries from randomly reserved contexts; RQ: randomly reserved queries.)

and gold proof and evaluate on the test split of DU5. We evaluate above metrics of varying depths d against two state-of-the-art baselines: RuleTakers (Clark et al., 2020) and PROVER (Saha et al., 2020), showed in Table 1. For RuleTakers and PROVER, we directly adopt the results reported in their paper. Note that RuleTakers can not generate a proof, so we only report the PA and FA on PROVER and PROBR. The corresponding validation set results can be found in the supplementary materials .

Overall, at each depth, PROBR generates comparable or superior QA accuracy to baselines. And for 88.8% of test examples, PROBR can generate exact proofs and answers. Similar to PROVER, the full accuracy matches the proof accuracy for PROBR, showing that in this fully supervised setting, full accuracy depends on proof accuracy at each depth. The predicted answer is always correct when the corresponding proof is correct. Actually, answering predicting is much easier than a proof generation.

When increasing depth, PROBR provides accurate answers without any loss in QA performance. It becomes harder to generate correct proofs for both PROVER and PROBR, while PROBR outperforms PROVER by 7 points of proof accuracy (65.1% → 72.2%) at depth 5.

4.3 Few-shot Learning

We explore the few-shot learning ability of PROBR against PROVER by reducing training data size. For the sake of comparison, we follow the same setting in (Saha et al., 2020), that is, randomly reserve 30k, 10k, 1k queries of overall 69762 training queries to train the model, denoted as "RQ".

It's worth noting that in the DU5 training dataset, several queries can be asked from a shared context. To better explore the ability when varying the amount of training data, we conduct another set of experiments, denoted as "RC". Specifically, we first randomly select context that appeared in the DU5 training dataset by a varying percentage, i.e., 10%, 5%, 1%, and then reserve training samples where the query is asked from the selected context. Results of both "RQ" and "RC" are showed in Table 2.

Generally speaking, proof generation is harder to improve with increased training data, while QA performance improves rapidly by enlarging the training size. PROBR widely defeats PROVER on QA accuracy in each setting in Table 2. Surprisingly, PROBR achieves 88.2% QA accuracy when training with only 700 samples (RC-1%). Overall, PROBR has a more stable ability for question answering when varying training data; however, PROVER's QA accuracy drops sharply when lacking training data. This is because that PROBR considers the joint distribution over all possible proofs and answers, and can better learn to reason over natural language statements. While as for proof accuracy, even if in some settings, PROBR loses to PROVER (RC-1%), we will soon discover that PROVER overfits to the small training data (Section 4.4 and 4.5).

Another interesting observation is that the full accuracy is not always consistent with the proof accuracy in few-shot learning, which is different from the observation in Section 4.2. Furthermore, we find that the gap between **PA** and **FA** when using PROBR is much smaller than that of PROVER. This is because PROVER trains in a multi-task way, where the question answering module and proof generation module could make independent errors, especially when training data is not enough. But PROBR can better utilize limited data to reason, which again verifies the effectiveness of PROBR.

Test	Cnt	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
B1	40	97.5	95.0	**100.0**	92.5	**100.0**	92.5	**100.0**
B2	40	100	95.0	**100.0**	95.0	**100.0**	95.0	**100.0**
E1	162	96.9	100	**100.0**	95.1	**97.5**	95.1	**97.5**
E2	180	98.3	100	**100.0**	91.7	**93.3**	91.7	**93.3**
E3	624	91.8	89.7	**98.2**	72.3	**79.3**	71.8	**79.3**
E4	4224	76.7	84.8	**95.6**	**80.6**	77.7	**80.6**	77.7
All	5270	80.1	86.5	**96.3**	**80.7**	79.3	**80.5**	79.3

Table 3: Zero-shot performance comparison among RuleTakers, PROVER, and PROBR on Birds-Electricity dataset after training on DU5.

4.4 Zero-shot Evaluation

Following previous work (Clark et al., 2020; Saha et al., 2020), we evaluate the out-of-distribution (OOD) performance of PROBR against baselines on six sub-datasets of Birds-Electricity. We conduct zero-shot experiments using DU5-trained models, which means that the model does not see any bird-domain or any electricity-domain samples during training. Results are showed in Table 3.

For QA accuracy, PROBR outperforms PROVER and RuleTakers obviously in all of sub-datasets. As for proof accuracy, PROBR performs better when the depth of the out-of-domain sample ≤ 3, while there is a PA drop compared to PROVER when testing on E4. This is a very interesting thing: superficially, proof accuracy drops for complicated unseen queries, but the QA accuracy for out-of-domain queries improves a lot (11 points on E4: $84.8\% \rightarrow 95.6\%$). We save it for future work to explore the portability of the proof and how an out-of-domain proof can help with question answering.

Moreover, we evaluate the zero-shot performance after few-shot learning. In Table 4, we report the results when testing on Birds-Electricity after training the model only on partial DU5 (RC-k and RQ-k, described in 4.3) training partitions. As shown in Table 4, when testing zero-shot performance after few-shot learning, PROBR is well ahead of PROVER on QA accuracy. However, as for proof accuracy, PROBR seems worth than PROVER on the zero-shot test. Again we point out this amazing observation. This indicates that data from different domains might have different proof form. The well-learned proof from one domain might not be directly adopted to another, but, by training with PROBR, the well-learned proof from one domain can help answer out-of-distribution queries.

Train Data		QA		PA		FA	
		PV	PB	PV	PB	PV	PB
100%		86.5	**96.3**	**80.7**	79.3	**80.5**	79.3
RC	10%	71.2	**99.9**	**59.4**	55.4	**59.2**	55.4
	5%	59.4	**99.5**	55.0	**69.1**	46.6	**69.0**
	1%	47.1	**60.6**	15.1	**34.6**	10.6	**24.4**
RQ	30k	83.3	**99.9**	76.79	**76.91**	76.72	**76.91**
	10k	78.2	**99.7**	54.3	**56.6**	54.3	**56.6**
	1k	50.4	**51.3**	**59.5**	34.6	**29.9**	17.3

Table 4: Zero-shot performance comparison between PROVER and PROBR after few-shot learning. Test on Birds-Electricity after training on DU5 or partial DU5 (RC-k and RQ-k) training partitions.

Train Data	QA			PA		FA	
	RT	PV	PB	PV	PB	PV	PB
DU0	53.5	**68.7**	56.9	44.4	**50.7**	**42.8**	41.3
DU1	63.5	73.7	**97.7**	63.8	**63.9**	61.9	**63.9**
DU2	83.9	89.6	**99.9**	72.6	**74.5**	72.3	**74.4**
DU3	98.9	98.6	**99.9**	79.1	**83.2**	79.1	**83.2**
DU5	99.2	99.3	**99.9**	87.1	**88.8**	87.1	**88.8**

Table 5: Performance comparison between RuleTakers, PROVER, and PROBR when testing on DU5 after training on DU0, DU1, DU2, DU3, respectively.

4.5 Generalization Ability

Generalize to Unseen Depth We conduct experiments to explore how well PROBR can generate proofs and provide answers at depths unseen during training. Following PROVER, we train the model on the training splits of DU0, DU1, DU2, and DU3, respectively, and test the QA performance and proof performance on the overall DU5 test set. As DU5 contains queries with higher depth than those seen during training, we can evaluate the model's ability when generalized to higher depth.

As shown in Table 5, PROBR performs better than RuleTakers and PROVER on all of QA/PA/FA performance when training on D1, D2 and D3, especially a significant improvement on QA performance. PROBR shows a high and comparable QA performance when training only on depth=1 (97.7%), which demonstrates PROBR's superior generalization ability on depth. This means PROBR can perfectly answer complicated queries using only simple training samples, which reduces the cost of constructing training data.

Generalize to Complex Language We also evaluate the robustness of PROBR when generalized to more diverse natural language. Following (Clark et al., 2020; Saha et al., 2020), we train our model on the combined training partitions of DU3

D	Cnt	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
0	2968	99.8	99.7	**99.8**	99.5	**99.5**	99.4	**99.4**
1	2406	99.3	98.6	**99.7**	98.0	**98.0**	97.3	**98.0**
2	1443	98.2	98.2	**99.9**	88.9	**88.9**	88.7	**88.9**
3	1036	96.7	96.5	**99.8**	90.0	**90.1**	89.9	**90.1**
4	142	90.1	88.0	**100**	76.1	**82.4**	76.1	**82.4**
All	8008	98.8	98.4	**99.8**	95.4	**95.6**	95.1	**95.5**

Table 6: Performance comparison between PRover and PRobr when testing on ParaRules test partitions after training on D3 + ParaRules training partitions.

Train Data		QA		PA		FA	
		PV	PB	PV	PB	PV	PB
100%		53.6	**82.8**	40.0	**43.8**	38.4	**41.6**
RC	10%	64.4	**89.3**	**42.0**	41.3	**41.0**	40.3
	5%	73.6	**84.7**	33.1	**36.5**	29.3	**35.1**
	1%	**59.0**	56.3	**30.4**	25.0	18.1	**18.3**
RQ	30k	59.0	**85.8**	38.6	**43.2**	37.5	**41.7**
	10k	59.7	**87.7**	41.7	**42.3**	40.3	**41.3**
	1k	51.4	**56.3**	**35.0**	25.0	**18.4**	16.5

Table 7: Performance comparison between PRover and PRobr when testing on ParaRules test partitions after training on DU5 or partial DU5 (RC-k and RQ-k) training partitions.

and ParaRules, and then test on the ParaRules test partition. The results in Table 6 show that PRobr is more robust for human-like language.

To better test the generalization ability for complex natural language, we train the model only on DU5 or partial DU5 (RC-k and RQ-k, described in 4.3) training partitions and test on test split of ParaRules. This is a more convincing setup since the model will never see the human-like language but all templated language during training. Results are shown in Table 7. When testing on ParaRules after only training on DU5, PRobr outperforms PRover by nearly **30 points** on QA accuracy(53.6% → 82.8%). A similar trend is observed for training on RC-k and RQ-k datasets, where PRobr improves the QA accuracy when generalized to human-like natural language. And the change for proof accuracy is not significant between PRobr and PRover, which supports the observation in Section 4.3 and 4.4, that PRobr improves QA performance by joint question-answering and proof-generation learning, but not necessarily improve the proof performance.

4.6 Ablation Studies

We investigate the effect of training strategy and objective term $\mathcal{L}_{\text{kl}}$ for our model. Specifically we compare PRobr with the following three variants: 1) **PRobr + Gold**, that is, we replace predicted proofs with gold proofs when we optimize $\mathcal{L}_{\text{qa}}$ during training. 2) **PRobr + KL**, that is, we add $\mathcal{L}_{\text{kl}}$ between $q(Y)$ and $p_{\text{pseudo}}(Y)$ during training. 3)**PRobr + Gold + KL** means both. For PRobr and above three variants, we first train on DU5 or partial DU5 (RC-k) training splits respectively, and Figure 3 reports the QA accuracy on test split of DU5 (left), ParaRules test partitions (middle) and Birds-Electricity (right). We observe that PRobr always achieves the best QA accuracy on all of three test datasets (DU5, ParaRules, Birds-Electricity) after training on all of four datasets with varying size (RC-1%, RC-5%, RC-10%, RC-100%). And the other three model variants show inconsistent performance in different settings[7].

5 Related Work

Text Reasoning over Formal Representation Early work employs a pipeline of methods that converts free text into logic form first (semantic parsing), and then uses formal logical reasoning (Musen and Van der Lei, 1988). Due to the serious error propagation caused by semantic parsing (Zettlemoyer and Collins, 2005; Berant et al., 2013; Berant and Liang, 2014), researchers focus on developing theorem provers by combining the symbolic techniques with the differentiable learning from neural networks (Reed and de Freitas, 2016; Abdelaziz et al., 2020; Abboud et al., 2020), such as NLProlog (Weber et al., 2019), SAT solving (Selsam et al., 2019) and Neural programme (Neelakantan et al., 2016). To bypass this expensive and error-prone intermediate logical representation, reasoning over natural language statements in an end-to-end manner is promising.

Text Reasoning over Natural Language Natural logic (MacCartney and Manning, 2009) focuses on semantic containment and monotonicity by incorporating semantic exclusion and implicativity. Subsequently, Clark et al. (2020) proposes to use a Transformer-based model to emulate deductive reasoning and achieves high accuracy on synthetically generated data. PRover (Saha et al., 2020)

---

[7] For more details, please refer to the supplementary materials.

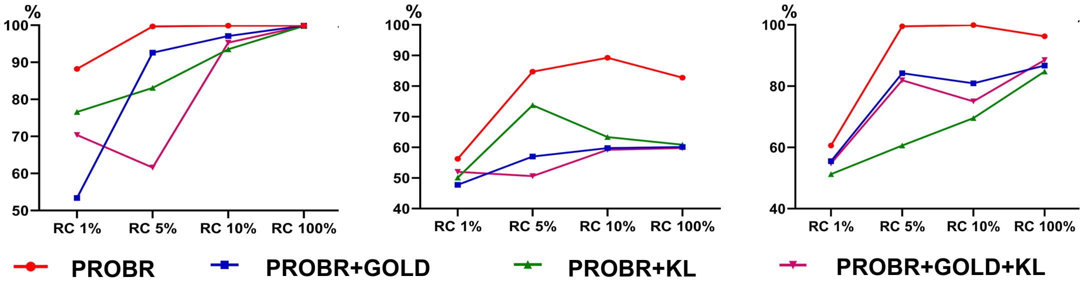

Figure 3: QA accuracy compared among PRobr, PRobr + Gold, PRobr + KL, and PRobr + Gold + KL on DU5 test partition (left), on ParaRules test partitions (middle) and on Birds-Electricity dataset (right), after training on DU5 or partial DU5 (RC-k) training splits.

points out that a reasoning system should not only answer queries but also generate a proof. However, PRover adopts the multi-task learning framework in the training stage and cannot effectively capture the interactions between question answering and proof generation. Along this line, we explore more powerful joint models to achieve deep reasoning.

QA and NLI There are bAbI (Weston et al., 2016), QuaRTz (Tafjord et al., 2019), ROPES (Lin et al., 2019) and Hotpot QA (Yang et al., 2018) (QA datasets) involved in rule reasoning. However, for those datasets, implicit rules (i.e., which multi-hop chains are valid) need to be inferred from the training data. In our task, the rules of reasoning are given in advance. Compared with the Natural Language Inference (MacCartney and Manning, 2014), our task can be regarded as its deductive subset. In particular, NLI allows for unsupported inferences (Dagan et al., 2013).

6 Conclusion

In this work, we propose PRobr, a novel probabilistic graph reasoning framework for joint question answering and proof generation. PRobr defines a joint distribution over all possible answers and proofs, which can directly characterize the interaction between answers and proofs. Experiments prove the effectiveness of proposed PRobr.

Acknowledgement

The authors wish to thank the reviewers for their helpful comments and suggestions, the authors of RuleTakers for their datasets, and the authors of PRover for providing the source code.

References

Ralph Abboud, İsmail İlkan Ceylan, and Thomas Lukasiewicz. 2020. Learning to reason: Leveraging neural networks for approximate DNF counting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3097–3104. AAAI Press.

Ibrahim Abdelaziz, Veronika Thost, Maxwell Crouse, and Achille Fokoue. 2020. An experimental study of formula embeddings for automated theorem proving in first-order logic. *arXiv preprint arXiv:2002.00423*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. 1989. What you always wanted to know about datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166.

Keith L. Clark. 1978. *Negation as Failure*, pages 293–322. Springer US, Boston, MA.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

F. R. Kschischang, B. J. Frey, and H. . Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

Bill MacCartney and Christopher D Manning. 2014. Natural logic and natural language inference. In *Computing meaning*, pages 129–147. Springer.

John McCarthy et al. 1960. *Programs with common sense*. RLE and MIT computation center.

Mark A Musen and Johan Van der Lei. 1988. Of brittleness and bottlenecks: Challenges in the creation of pattern-recognition and expert-system models. In *Machine Intelligence and Pattern Recognition*, volume 7, pages 335–352. Elsevier.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Allen Newell and Herbert Simon. 1956. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79.

Manfred Opper and David Saad. 2001. *Advanced mean field methods: Theory and practice*. MIT press.

Scott E. Reed and Nando de Freitas. 2016. Neural programmer-interpreters. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Ruslan Salakhutdinov. 2014. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. QuaRTz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946, Hong Kong, China. Association for Computational Linguistics.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

Supplementary Materials for
Probabilistic Graph Reasoning for Natural Proof Generation

A Experimental Environment

Parameter	**Value**
Training Epochs	30
Optimizer	AdamW
Gradient Clipping	1.0
Batch Size	8
Dropout Rate	0.1
Learning rate	1e-5
Max Sequence Length	300

Table 8: Model configurations.

Table 8 lists the default model configurations. We produce PRobr on 8 NVIDIA Tesla-V100 GPUs. We implement PRobr with PyTorch, using RoBERTa (Liu et al., 2019) as pre-trained language model.

B Results on Development Set

Table 9 shows the results on development set under the fully supervised setting, and the corresponding test set results are shown in Table 1. Overall, it is consistent with the findings of the test set results. PRobr achieves the best performance in three accuracy metrics (**QA**, **PA** and **FA**).

D	**Cnt**	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
0	6299	100	100	**100**	98.5	**98.7**	98.5	**98.7**
1	4434	98.4	98.8	**100**	92.2	**93.6**	92.2	**93.6**
2	2915	98.4	99.2	**99.0**	85.6	**87.3**	85.6	**87.3**
3	2396	98.8	98.7	**100**	82.8	**85.4**	82.8	**85.4**
4	2134	99.2	98.8	**99.7**	76.9	**80.8**	76.9	**80.8**
5	2003	99.8	99.3	**99.9**	67.4	**74.6**	67.4	**74.6**
All	20192	99.2	99.3	**99.9**	88.0	**90.0**	88.0	**90.0**

Table 9: Fully supervised learning performance compared among RuleTakers (RT), PRover (PV) and PRobr (PB) on development split of DU5 after training on training split of DU5, reported in varying depth.

Table 10 lists the results on development set of DU5 after training on DU0, DU1, DU2, DU3, respective. The corresponding test set resluts are shown in Table 5. Comparing with Table 5, each number is very close (fluctuates within 1%) and similar conclusions can be drawn.

Train Data	**QA**		**PA**		**FA**	
	PV	PB	PV	PB	PV	PB
DU0	**68.3**	57.0	43.8	**50.7**	**42.3**	41.7
DU1	73.2	**98.5**	63.9	**64.3**	61.8	**64.3**
DU2	89.3	**99.9**	72.6	**74.3**	72.3	**74.3**
DU3	98.3	**99.9**	79.4	**83.2**	79.4	**83.2**
DU5	99.3	**99.9**	88.0	**90.0**	88.0	**90.0**

Table 10: Performance comparison between PRover and PRobr development split of DU5 after training on DU0, DU1, DU2, DU3, respectively.

C Results of Ablation Studies

Table 11 lists the comparison results of PRobr and the three variants of PRobr (mentioned in Section 4.6). Regarding the Table 11, we have three observations:

1. In all ablation experiments, PRobr achieved the best **QA** performance, demonstrating that PRobr can capture critical information for question answering in a variety of settings. However, since some of the dataset are artificially synthesized, it is difficult to guarantee that PRobr will work in the real dataset as well. We leave it as future work.

2. In some cases, variant d) (**PRobr + Gold + KL**) outperforms PRobr in **PA** and **FA**. It shows the potential advantages of the KL term. In the future, we will explore proof generation in a semi-supervised learning scenario through this KL term.

3. When we compare the performance of the two models PRobr and **PRobr + Gold**, we can see that whether the predicted proof or the correct proof is used during training significantly affects the final performance. Applying some heuristic strategies may give better results, such as scheduled sampling (Bengio et al., 2015). We will try it in the future.

	Test on DU5			Test on ParaRules			Test on Birds-Electricity		
	QA	PA	FA	QA	PA	FA	QA	PA	FA
RC-1%	a)**88.2**	a)**21.6**	a)20.3	a)**56.3**	a)**25.0**	a)**18.3**	a)**60.6**	a)**34.6**	a)**24.4**
	b)53.4	b)**21.6**	b)**21.3**	b)47.8	b)**25.0**	b)14.1	b)55.5	b)**34.6**	b)17.4
	c)76.6	c)**21.6**	c)14.8	c)50.1	c)**25.0**	c)12.9	c)51.3	c)**34.6**	c)17.6
	d)70.4	d)**21.6**	d)20.0	d)52.0	d)**25.0**	d)14.5	d)54.8	d)**34.6**	d)19.9
RC-5%	a)**99.7**	a)44.2	a)44.2	a)**84.7**	a)36.5	a)35.1	a)**99.5**	a)69.1	a)**69.0**
	b)92.6	b)43.7	b)43.6	b)57.0	b)37.1	b)35.4	b)84.2	b)62.0	b)62.0
	c)83.1	c)31.4	c)30.1	c)73.8	c)36.6	c)27.6	c)60.6	c)45.3	c)42.0
	d)61.6	d)**47.9**	d)**45.9**	d)50.6	d)**38.9**	d)**37.2**	d)81.9	d)**73.4**	d)65.4
RC-10%	a)**99.9**	a)60.4	a)60.4	a)**89.3**	a)**41.3**	a)**40.3**	a)**99.9**	a)55.4	a)55.4
	b)97.1	b)57.9	b)57.9	b)59.8	b)40.0	b)38.6	b)80.9	b)59.6	b)59.6
	c)93.5	c)52.2	c)51.8	c)63.4	c)40.7	c)39.2	c)69.6	c)53.3	c)53.2
	d)95.3	d)**69.3**	d)**69.0**	d)59.2	d)37.5	d)35.9	d)75.0	d)**66.2**	d)**66.1**
RC-100%	a)**99.9**	a)88.8	a)88.8	a)**82.8**	a)**43.8**	a)**41.6**	a)**96.3**	a)79.3	a)79.3
	b)**99.9**	b)88.7	b)88.7	b)60.1	b)42.7	b)39.8	b)86.7	b)**81.4**	b)**81.3**
	c)99.8	c)89.1	c)89.1	c)60.9	c)42.9	c)41.2	c)84.8	c)78.6	c)78.6
	d)99.8	d)**89.6**	d)**89.6**	d)59.8	d)43.2	d)41.5	d)88.5	d)80.9	d)80.9

Table 11: QA accuracy, proof accuracy and full accuracy compared among PROBR, PROBR + Gold, PROBR + KL, and PROBR + Gold + KL on DU5 test partition (left), on ParaRules test partitions (middle) and on Birds-Electricity dataset (right), after training on DU5 or partial DU5 (RC-k) training splits, where a)–**PROBR**, b)–**PROBR + Gold**, c)–**PROBR + KL**, d)–**PROBR + Gold + KL**.

Enhancing Zero-shot and Few-shot Stance Detection with Commonsense Knowledge Graph

Rui Liu[1,2], **Zheng Lin**[1]*, **Yutong Tan**[1,2], **Weiping Wang**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
`{liurui1995,linzheng,tanyutong,wangweiping}@iie.ac.cn`

Abstract

In this paper, we consider a realistic scenario on stance detection with more application potential, i.e., zero-shot and few-shot stance detection, which identifies stances for a wide range of topics with no or very few training examples. Conventional data-driven approaches are not applicable to the above zero-shot and few-shot scenarios. For human beings, commonsense knowledge is a crucial element of understanding and reasoning. In the absence of annotated data and cryptic expression of users' stance, we believe that introducing commonsense relational knowledge as support for reasoning can further improve the generalization and reasoning ability of the model in the zero-shot and few-shot scenarios. Specifically, we introduce a commonsense knowledge enhanced model to exploit both the structural-level and semantic-level information of the relational knowledge. Extensive experiments demonstrate that our model outperforms the state-of-the-art methods on zero-shot and few-shot stance detection task.

1 Introduction

Stance detection aims to identify the text authors' attitudes or positions towards a specific topic as a category label from this set: {*Pro*, *Con*, *Neutral*} (Mohammad et al., 2016b, 2017). Conventionally, this task is designed to learn a target-specific classifier for prediction on the same topic. Afterward, cross-target stance detection comes out as a subclass of the initial generic stance detection, where the classifier is adapted from different but closely related topics (e.g., training classifier on "Hillary Clinton" and predicting on "Donald Trump") (Augenstein et al., 2016a). However, both target-specific and cross-target stance detection models (Du et al., 2017; Wei and Mao, 2019; Augenstein et al., 2016a; Zhang et al., 2020) require a large number of training examples with manual annotation, and annotating data for thousands of new topics is time-consuming and expensive.

Topic: Stability **Stance:** Pro

Text: Tenure does not mean a teacher cannot lose their job. It requires due process before termination. Before tenure is achieved, a teacher can be fired without due process. In the Atlanta School District administrators, fearing that low test scores would cost them their jobs, instructed teachers to change student test responses. Without tenure and due process, teachers risked being fired if they didn't follow instructions.

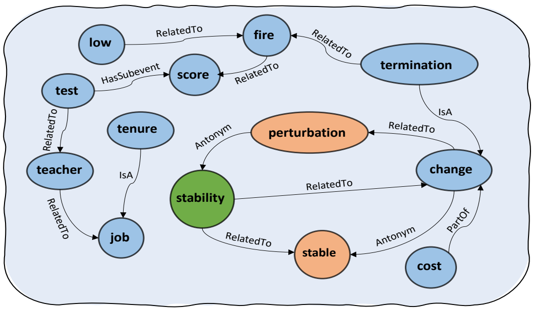

Figure 1: An example where the topic isn't contained in the text. Entity mentions in the text and the topic are highlighted. We omit the reverse edges in the relation graph for clarity.

In this paper, we focus on zero-shot and few-shot stance detection (Allaway and McKeown, 2020), a task to classify stances for a large number of topics with no or very few training examples. A key challenge for zero-shot and few-shot stance detection is the generalization ability of the models. However, most of the previous approaches (Xu et al., 2018; Augenstein et al., 2016b; Wei and Mao, 2019; Wei et al., 2019) for stance detection have relied on only the training data, which fails to achieve satisfactory results in zero-shot and few-shot scenarios. Another prominent challenge is the implicit expression of the users' stance, where the topic does not always appear in the document, resulting

---

*Zheng Lin is the corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3152–3157
August 1–6, 2021. ©2021 Association for Computational Linguistics

in the difficulty of directly establishing a connection between the topic and the document. Take Figure 1 as an example, the topic "*Stability*" is not mentioned in the document, where the relational knowledge can supplement the lack of explicit inferential evidence such as (*stability, Antonym, perturbation*) and (*perturbation, RelatedTo, change*) etc. Despite attempting to introduce the external word-level semantic and emotion knowledge (Cambria et al., 2018) about each word of the document, Zhang et al. (2020) neglect the global relationship between the topic and the document.

To further tackle the above challenges, we propose to bring in commonsense knowledge from external structural knowledge base ConceptNet (Speer et al., 2017). We believe that the relational knowledge graph extracted from ConceptNet can promote the transmission of relational information between the document and the topic as well as the inference of corresponding stances, which can further reduce the dependency on annotated data. Specifically, we introduce a commonsense knowledge enhanced module based on Graph Convolution Networks (Kipf and Welling, 2017; Velickovic et al., 2018; Vashishth et al., 2020) to exploit both the structural-level and semantic-level information of the relation subgraph, which can further strengthen the generalization and reasoning capacities of the model. Extensive experiments show that our method outperforms the state-of-the-art models on the benchmark dataset for zero-shot and few-shot stance detection.

2 Methodology

2.1 Problem Formulation

Formally, $D = \{(x_i, t_i, y_i)\}_{i=1}^{N}$ denotes the zero-shot stance detection dataset which contains N exmples, where x_i is a document, t_i is the corresponding topic, and y_i is the stance label. The goal of the task is to obtain a stance label $\widetilde{y}$ given x_i and t_i. To bridge the document and the topic, we introduce an commonsense knowledge subgraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ extracted from the external KG, where $\mathcal{V}$ is the subset of the concepts and $\mathcal{E}$ denotes the relations between concepts.

2.2 BERT Encoding

We employ the pre-trained language model BERT (Devlin et al., 2019) to encode the document x and topic t. Specifically, we concatenate x and t into one input sequence in the following format: [CLS] x [SEP] t [SEP]. Then, the input sequence is fed into BERT to obtain the contextual representations $\boldsymbol{X} = \{\boldsymbol{x}^1, \cdots, \boldsymbol{x}^m\}$ for the document and $\boldsymbol{T} = \{\boldsymbol{t}^1, \cdots, \boldsymbol{t}^n\}$ for the topic, where m and n is the length of the document and the topic respectively. Finally, we can get the average representations $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{t}}$ of the document and the topic, respectively.

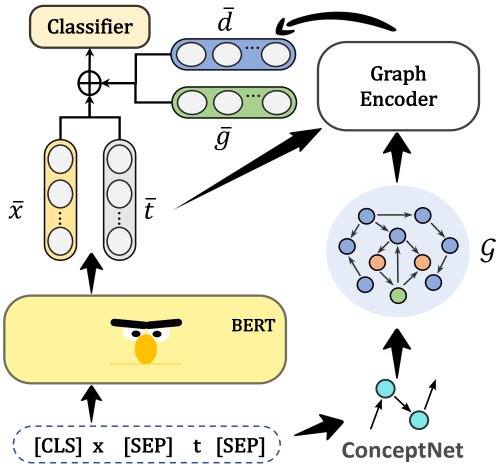

Figure 2: The architecture of our model.

2.3 Knowledge Graph Encoding with CompGCN

Before introducing our graph encoder, let's first describe the process of constructing the relational subgraph from the external knowledge graph. We adopt ConceptNet as our knowledge graph base G. ConceptNet consists of millons of relation triples, which contains 34 relations in total. Each triple is represented as $R = (u, r, v)$, where u is the head concept, r is the relation, and v is the tail concept. We match phrases in **documents** and **topics** to sets of mentioned concepts (C_d and C_t respectively) from the ConceptNet. To extract the relational subgraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from G, we find the two-hop directed paths from concepts in C_d to concepts in C_t. All concepts on the paths form the concepts set $\mathcal{V}$ and $\mathcal{E}$ is composed of all edges between concepts within $\mathcal{V}$. Moreover, we add reverse relations edge between any concept pair to improve the information flow.

Most of the existing research on GCNs mainly focuses on non-relational graphs. Thus, to incorporate commonsense relational knowledge, we utilize CompGCN (Vashishth et al., 2020), a variant of Graph Convolution Networks (GCNs), which jointly embeds both nodes and relations of the subgraph $\mathcal{G}$. The graph encoder consists of L-stacked

CompGCN layers. The features of nodes and relations are all initialized with **TransE** (Bordes et al., 2013) embeddings. We update node representations by aggregating the information from their neighbors and their relational edges. Formally, the update equation of nodes is defined as:

$$\boldsymbol{h}_v^{l+1} = f\left(\frac{1}{|\mathcal{N}(v)|} \sum_{(u,r)\in\mathcal{N}(v)} \boldsymbol{W}_u^l \phi(\boldsymbol{h}_u^l, \boldsymbol{h}_r^l)\right), \tag{1}$$

where f is an activation function, $\mathcal{N}(v)$ is the neighbors of the node v; $\boldsymbol{h}_u$, $\boldsymbol{h}_r$, and $\boldsymbol{h}_v$ are the representations of node u, node v and relation r.

Here, ϕ is a entity-relation composition operation based on the translational theory (Bordes et al., 2013) in the form of subtraction:

$$\phi(\boldsymbol{h}_u, \boldsymbol{h}_r) = \boldsymbol{h}_u - \boldsymbol{h}_r. \tag{2}$$

The relation embeddings are transformed as follows: $\boldsymbol{h}_r^{l+1} = \boldsymbol{W}_r^l \boldsymbol{h}_r^l$. After that, we obtain the node representations $\boldsymbol{H}_d$ and $\boldsymbol{H}_t$ of C_d and C_t, respectively. To aggregate reasonable relational information, we compute the average relational representation $\hat{\boldsymbol{d}}$ for C_d by performing scaled dot-product attention (Vaswani et al., 2017), with $\hat{\boldsymbol{t}}$ as the key and $\boldsymbol{H}_d$ as the query and value. Similarly, we get the average relational representation $\hat{\boldsymbol{g}}$ for C_t.

2.4 Stance Classification

We concatenate the representations of plain texts (i.e., $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{t}}$) with the relational representations (i.e., $\hat{\boldsymbol{d}}$ and $\hat{\boldsymbol{g}}$) to make full use of the textual information and the graph structural information. Afterward, the concatenated representations are fed into a two-layer multi-layer perception (MLP) with a softmax function to predict the stance label:

$$\hat{y} = softmax(\text{MLP}([\hat{\boldsymbol{x}}; \hat{\boldsymbol{t}}; \hat{\boldsymbol{d}}; \hat{\boldsymbol{g}}])), \tag{3}$$

where [;] is vector concatenation operation.

Finally, the parameters of the network are trained using multi-class cross-entropy loss.

3 Experiments

3.1 Datasets and Evaluation Metrics

Different from the existing stance datasets (Mohammad et al., 2016a; Kobbe et al., 2020; Vamvas and Sennrich, 2020) including few topics (ranging from 6 to 194 topics), Allaway and McKeown (2020) present a new dataset for the zero-shot and few-shot stance detection, VAried Stance Topics (VAST), which consists of thousands of topics. The statistics of VAST are demonstrated in Table 1. Note that a document only belongs to one partition, which means that documents in the training set do not appear in the validation set or the test set, and vice versa. In addition, the zero-shot topics in the test set never appear in the training set, and the few-shot topics only contains few training data. Following the previous work (Allaway and McKeown, 2020), the macro average of F1-score is adopted as the evaluation metric.

Statistics	**Train**	**Dev**	**Test**
# Examples	13477	2062	3006
# Documents	1845	682	786
# Zero-shot Topics	4003	383	600
# Few-shot Topics	638	114	159

Table 1: Detailed statistics for VAST.

3.2 Experimental Settings

We employ the base version of BERT as the backbone. The graph encoder has two layers of CompGCN. We train our model on 1 GPU (Nvidia RTX TITAN, 24G) using Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 4e-5 and a batch size of 64. All documents are kept the first 200 words and the topic is the first 5 words. The best checkpoints are selected according to the evaluation metrics on the validation set. We repeat our model three times using different random seeds and report the averaged results. Our code will be released on Github.

We compare our model with the several state-of-the-art baselines: **BiCond** (Augenstein et al., 2016b), **CrossNet** (Xu et al., 2018), **SEKT** (Zhang et al., 2020), **BERT-joint** (Allaway and McKeown, 2020) and **TGA-Net** (Allaway and McKeown, 2020). The first three models are based on BiLSTM for cross-target stance detection. When training the latter two BERT-based models, Allaway and McKeown (2020) fixed the parameters of the BERT module. Hence, we extend two models **BERT-joint-ft** and **TGA-Net-ft**, in which BERT has been fine-tuned during the training process. Besides, we compare our model with **BERT-GCN**, which applies the conventional GCN (Kipf and Welling, 2017) only considering the node information aggregation.

Model	F1 Zero-Shot				F1 Few-Shot				F1 All			
	pro	con	neu	all	pro	con	neu	all	pro	con	neu	all
BiCond	.459	.475	.349	.427	.454	.463	.259	.392	.457	.468	.306	.410
Cross-Net	.462	.434	.404	.434	.508	.505	.410	.474	.486	.471	.408	.455
SEKT	.504	.442	.308	.418	.510	.479	.215	.474	.507	.462	.263	.411
BERT-joint	.546	.584	.853	.660	.543	.597	.796	.646	.545	.591	.823	.653
TGA-Net	.554	.585	.858	.666	.589	.595	.805	.663	.573	.590	.831	.665
BERT-joint-ft	.579	.603	.875	.685	.595	.621	.831	.684	.588	.614	.853	.684
TGA-Net-ft	.568	.598	**.885**	.684	.628	.601	.834	.687	.599	.599	**.859**	.686
BERT-GCN	.583	.606	.869	.686	.628	**.634**	.830	.697	.606	**.620**	.849	.692
CKE-Net(Ours)	**.612**	**.612**	.880	**.702**	**.644**	.622	**.835**	**.701**	**.629**	.617	.857	**.701**

Table 2: Macro-averaged F1 on the test set. The suffix "ft" means the bert model is fine-tuned.

Model	Imp	mlT	mlS	Qte	Sarc
BERT-joint	.571	.590	.524	.634	.601
TGA-Net	.594	.605	.532	.661	.637
BERT-joint-ft	.617	.621	.547	.647	.668
TGA-Net-ft	.615	.625	.546	.664	.675
BERT-GCN	.619	.627	.547	.668	.673
CKE-Net	**.625**	**.634**	**.553**	**.695**	**.682**

Table 3: Accuracy on five challenging phenomena in the test set.

3.3 Results and Discussions

Results of Different Scenarios The overall results of our model and baselines are shown in Table 2. To evaluate the effectiveness of our method on different scenarios, we categorize the results into three subsets: *Zero-Shot*, *Few-shot* and *All*. Our model outperforms all baselines by a large margin, which can demonstrate the importance of incorporating the rich commonsense knowledge in the form of relational graphs. Additionally, we observe that all BERT-based baselines perform worse on *pro* examples than on *con* examples for zero-shot topics. A possible explanation might be that there are more negative words in the con examples, which is easier to identify in terms of semantics. Conversely, our model brings a significant improvement on average for both zero-shot and few-shot topics, which indicates that the relational information from the external knowledge base can boost the generalization and reasoning ability. Compared to BERT-GCN only modeling the node aggregation, our model takes full advantage of the relational information to contribute much to the overall performance.

Furthermore, all BERT-based models perform better than other baseline methods. In presents that the pre-trained models possess more strong generalization capability because it learns from a large-scale unsupervised corpus. Besides, SEKT does not achieve effective improvement on VAST, probably because they only introduce the external semantic knowledge at the token level without explicitly considering the overall relationship between the topic and the document. And the token-level approach is difficult to transplant to BERT.

Results of Different Phenomena To further analyse the effectiveness of our model, we test it under five challenging phenomena in the VAST following (Allaway and McKeown, 2020): (1) **Imp**: examples with non-neutral labels, where the topic does not appear in the document, (2) **mlT**: documents having multiple examples with different topics, (3) **mlS**: documents having multiple examples with different and non-neutral labels, (4) **Qte**: documents with quotations, (5) **Sarc**: documents with sarcasm (Habernal et al., 2018). As shown in table 3, our model achieves the best performance on all difficult phenomena. In particular, the improvement on **Imp** demonstrates that introducing external relational knowledge can help the model better understand the relationship between the topic and the article. Besides, the external semantic-level information from the relational subgraph makes our model perform better on the special rhetorics (**Qte** and **Sarc**).

4 Conclusion

In this paper, we interpret the necessity of introducing commonsense knowledge for zero-shot and few-shot stance detection. We present a commonsense knowledge enhanced method, which facilitates the integration of the relational knowledge to further strengthen the generalization and reasoning capacities of the stance detection model. Extensive experiments show that our proposed model achieved state-of-the-art results.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61976207, No. 61906187).

References

Emily Allaway and Kathleen McKeown. 2020. Zero-Shot Stance Detection: A Dataset and Model using Generalized Topic Representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8913–8931, Online. Association for Computational Linguistics.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016a. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016b. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 876–885. The Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1795–1802. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3988–3994.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1930–1940. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jonathan Kobbe, Ioana Hulpus, and Heiner Stuckenschmidt. 2020. Unsupervised stance detection for arguments from consequences. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 50–60. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016a. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 31–41. The Association for Computer Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016b. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Trans. Internet Techn.*, 17(3):26:1–26:23.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Jannis Vamvas and Rico Sennrich. 2020. X -stance: A multilingual multi-target dataset for stance detection. In *Proceedings of the 5th Swiss Text Analytics Conference and the 16th Conference on Natural Language Processing, SwissText/KONVENS 2020, Zurich, Switzerland, June 23-25, 2020 [online only]*, volume 2624 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Penghui Wei and Wenji Mao. 2019. Modeling transferable topics for cross-target stance detection. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1173–1176. ACM.

Penghui Wei, Wenji Mao, and Guandan Chen. 2019. A topic-aware reinforced model for weakly supervised stance detection. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7249–7256. AAAI Press.

Chang Xu, Cécile Paris, Surya Nepal, and Ross Sparks. 2018. Cross-target stance classification with self-attention networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 778–783. Association for Computational Linguistics.

Bowen Zhang, Min Yang, Xutao Li, Yunming Ye, Xiaofei Xu, and Kuai Dai. 2020. Enhancing cross-target stance detection with transferable semantic-emotion knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3188–3197, Online. Association for Computational Linguistics.

Dialogue Graph Modeling for Conversational Machine Reading

Siru Ouyang[1,2,3,*], **Zhuosheng Zhang**[1,2,3,*], **Hai Zhao**[1,2,3,†]
[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
`{oysr0926,zhangzs}@sjtu.edu.cn,zhaohai@cs.sjtu.edu.cn`

Abstract

Conversational Machine Reading (CMR) aims at answering questions in complicated interactive scenarios. Machine needs to answer questions through interactions with users based on given rule document, user scenario and dialogue history, and even initiatively asks questions for clarification if necessary. Namely, the answer to the task needs a machine in the response of either *Yes*, *No*, *Irrelevant* or to raise a follow-up question for further clarification. To effectively capture multiple objects in such a challenging task, graph modeling is supposed to be adopted, though it is surprising that this does not happen until this work proposes a dialogue graph modeling framework by incorporating two complementary graph models, i.e., explicit discourse graph and implicit discourse graph, which respectively capture explicit and implicit interactions hidden in the rule documents. The proposed model is evaluated on the ShARC benchmark and achieves new state-of-the-art by first exceeding the milestone accuracy score of 80%. The source code of our paper is available at `https://github.com/ozyyshr/DGM`

1 Introduction

Training machines to understand documents is the major goal of machine reading comprehension (MRC) (Hermann et al., 2015; Hill et al., 2016; Rajpurkar et al., 2016; Nguyen et al., 2016; Joshi et al., 2017; Rajpurkar et al., 2018; Choi et al., 2018; Zhang et al., 2018; Reddy et al., 2019; Zhang et al., 2020c, 2021). Especially, in the recent challenging conversational machine reading (CMR) task, the machine is required to read and interpret the given rule document and the user scenario, ask clarification questions, and then make a final decision (Saeidi et al., 2018). As an example shown in Figure 1. The user posts the scenario and asks a question concerning whether the loan meets the needs. Since the user cannot know the rule document, the information he/she provided may not be sufficient for the machine to decide. Therefore, a series of follow-up questions are asked by the machine until it can finally make a conclusion.

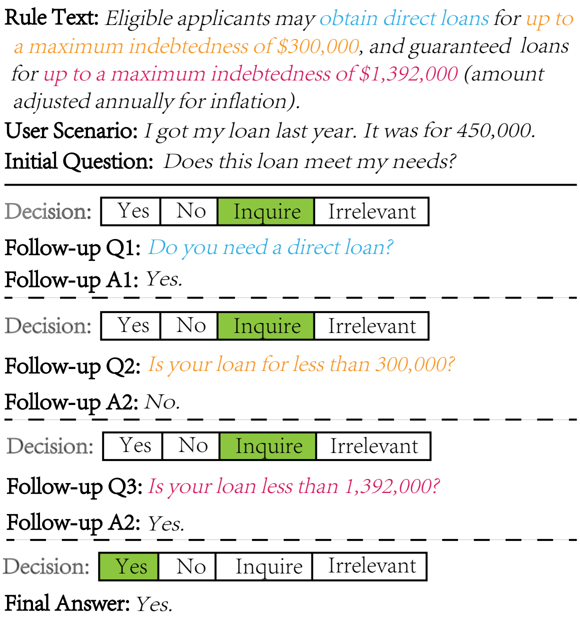

Rule Text: *Eligible applicants may obtain direct loans for up to a maximum indebtedness of $300,000, and guaranteed loans for up to a maximum indebtedness of $1,392,000 (amount adjusted annually for inflation).*
User Scenario: *I got my loan last year. It was for 450,000.*
Initial Question: *Does this loan meet my needs?*

Decision: Yes No [✓ Inquire] Irrelevant
Follow-up Q1: *Do you need a direct loan?*
Follow-up A1: *Yes.*

Decision: Yes No [✓ Inquire] Irrelevant
Follow-up Q2: *Is your loan for less than 300,000?*
Follow-up A2: *No.*

Decision: Yes No [✓ Inquire] Irrelevant
Follow-up Q3: *Is your loan less than 1,392,000?*
Follow-up A2: *Yes.*

Decision: [✓ Yes] No Inquire Irrelevant
Final Answer: *Yes.*

Figure 1: An example dialog from ShARC benchmark dataset (Saeidi et al., 2018). At each turn, the machine can give a decision regarding the initial question put up by the user. If the decision is *Inquire*, the machine will ask a clarification question to help with decision making. The corresponding rule document and the question are marked in the same color in the figure.

The major challenges for the conversational machine reading include the rule document interpretation, and reasoning with the background

---

[*] Equal contribution. †Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), Key Projects of National Natural Science Foundation of China (U1836222 and 61733011), Huawei-SJTU long term AI project, Cutting-edge Machine Reading Comprehension and Language Model. This work was supported by Huawei Noah's Ark Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3158–3169
August 1–6, 2021. ©2021 Association for Computational Linguistics

knowledge, e.g., the provided rule document, user scenario and the input question. Existing works (Zhong and Zettlemoyer, 2019; Lawrence et al., 2019; Verma et al., 2020; Gao et al., 2020a,b) have made progress in improving the reasoning ability by modeling the interactions among rule document, user scenario and the other elements implicitly. As for rule document interpretation, most existing approaches simply split the rule document into several rule conditions to be satisfied. In general, they first track the entailment state of each rule condition for decision making and then form a certain under-specified rule span into a follow-up question.

However, the aforementioned cascaded methods tend to model in a holistic way, i.e. interpreting the rule document with other elements quite plainly, which have the following drawbacks. First, very little attention is paid to the inner dependencies of rule conditions such as the discourse structure and discourse relations (Qin et al., 2016, 2017; Bai and Zhao, 2018). Second, existing methods do not dig deep enough into mining the interactions between the rule document and other elements, especially user scenarios.

As seen, the interactions of elements in CMR is far more complicated than that of traditional MRC tasks. Therefore, we proposed a dialogue graph modeling (DGM) framework consisting of two complementary graphs to fully capture the complicated interactions among all the elements. Firstly, an explicit discourse graph is constructed by making use of discourse relations of elementary discourse units (EDUs) generated from rule documents to tackle explicit element interactions. User scenario representation is injected as a special global vertex, to bridge the interactions and capture the inherent dependency between the rule document and the user scenario information. Secondly, an implicit discourse graph is designed for digging the latent salient interactions among rule documents by decoupling and fusing mechanism. The two dialogue graphs compose the encoder of our model and feed fusing representations to the decoder for making decisions.

As to our best knowledge we are the first to explicitly model the relationships among rules and user scenario with Graph Convolutional Networks (GCNs) (Schlichtkrull et al., 2018). Experimental results show that our proposed model outperforms the baseline models in terms of official evaluation metrics and achieves the new state-of-the-art results on ShARC, the benchmark dataset for CMR (Saeidi et al., 2018). In addition, our model enjoys strong interpretability by modeling the process in an intuitive way.

2 Related Work

Conversational Machine Reading. Compared with traditional triplet-based MRC tasks that aim to answer questions by reading given document (Hermann et al., 2015; Hill et al., 2016; Rajpurkar et al., 2016; Nguyen et al., 2016; Joshi et al., 2017; Rajpurkar et al., 2018; Zhang et al., 2020a,b), our concerned CMR task (Saeidi et al., 2018) is more challenging as it involves rule documents, scenarios, asking clarification question, and making a final decision. The major differences lie in two sides: 1) machines are required to formulate follow-up questions for clarification before confident enough to make the decision, 2) machines have to make a question-related conclusion by interpreting a set of complex decision rules, instead of simply extracting the answer from the text. Existing works (Zhong and Zettlemoyer, 2019; Lawrence et al., 2019; Verma et al., 2020; Gao et al., 2020a,b) have made progress in improving the reasoning ability by modeling the interactions between the rule document and other elements. As a widely-used manner, the existing models commonly extracted the rule documents into individual rule items, and track the rule fulfillment for the dialogue states. As indicated in Gao et al. (2020b), improving the rule document representation remains a key factor to the overall model performance, because the rule documents are formed with a series of implicit, separable, and possibly interrelated rule items that the conversation should satisfy before making decisions. However, previous work only considered segmenting the discourse, and neglected the inner discourse structure/relationships between the EDUs (Gao et al., 2020b). Compared to existing methods, our method makes the first attempt to explicitly capture elaborate interactions among all the document elements, user scenarios and dialogue history updates.

Graph Modeling in MRC. Inspired by the impressive performance of GCN (Kipf and Welling, 2017; Luo and Zhao, 2020), efforts towards better performance on MRC utilizing GCNs have sprung up, such as BAG (Cao et al., 2019), GraphRel (Fu et al., 2019) and social information reasoning (Li and Goldwasser, 2019). Unlike the previous works who just apply the graph framework mechanically to turn the entire passage or document into a graph, the discourse graph we proposed is delicately designed to mine the relationships of multiple elements in CMR task and to facilitate information flow over the graph.

3 Model

As illustrated in Figure 2, our model mainly consists of three parts to generate the final answer.

1. Rule document is segmented into rule EDUs, which is then tagged discourse relationship by a pre-trained discourse parser.

2. In the encoding phase, taking segmented and preprocessed rule document and user scenario as input, we build two graphs over the segments (EDUs), in which the explicit discourse graph captures the interactions among rules and user scenarios with the support of tagged discourse relationship, while the implicit discourse graph mines latent salient interactions from the raw rule document.

3. For decoding, an interaction layer takes the combined representation generated by both explicit and implicit discourse graph of rule EDUs, initial question, user scenario and dialog history as inputs, and maps it into an entailment state of each rule EDU. With these rule fulfillment situation, we can make a decision among *Yes, No, Inquire and Irrelevant*. Once the decision is made to be *Inquire*, the model generates a follow-up question to clarify the under-specified rule span in the rule document.

The complete training procedure of DGM is shown in Algorithm 1.

3.1 Preprocessing

EDU Segmentation. We first separate the rule document into several units each containing exactly one condition. Here we follow DISCERN (Gao et al., 2020b) adopting the discourse segmenter (Li et al., 2018) to break the rule document into EDUs.

Discourse Relation. Unlike EDU segmentation which only concerns with constituency-based logical structures, discourse relation allows relations between the non-adjacent EDUs. There are in total 16 discourse relations according to STAC (Asher et al., 2016), namely, *comment, clarification-question, elaboration, acknowledgement, continuation, explanation, conditional, question-answer, alternation, question-elaboration, result, background, narration, correction, parallel and contrast*. We adopt a pre-trained discourse parser (Shi and Huang, 2019)[1] to decide the dependencies between EDUs and the corresponding relation types with the structured representation of each EDU.

Algorithm 1: DGM Algorithm

```
Input: word embeddings $E = \{e_1, .., e_n\}$,
       dimension of word embeddings $d$,
       token ids of rule document $I$,
       discourse relation $D$ of rule
       document, number of rule EDUs $n$
Output: Final decision in Yes/No/Irrelevant
        or a follow-up question
for $i$ in epochs do
    build explicit discourse graph $G(E, D)$
    $G_{n \times d} \leftarrow GCN(G)$
    build implicit discourse graph by
      calculating adjacent matrix $M_l$, $M_c$
    get rule EDU representation $C_{n \times d}$ by
      Eq.(6) and (7)
    combined representation for [RULE] $\tilde{r_i} \leftarrow$
      self-Attn$(C + G)_i$
    entailment state $f_i \leftarrow$ LINEAR$(\tilde{r_i})$
    make the decision $z$ by Eq.(10) based
      on $\tilde{r_i}$ and $f_i$
    if $z$ is Inquire then
        generate follow-up question
    return $z$ or follow-up question
```

3.2 Encoding Block

Embedding. We select the pre-trained language model (PrLM) model ELECTRA (Clark et al., 2020) for encoding. As shown in the figure, the input of our model includes rule document which has already be parsed into EDUs with explicit discourse relation tagging, user initial question, user scenario and the dialog history. Instead of inserting a `[CLS]` token before each rule EDU to get a sentence-level representation, we use `[RULE]` which is proved to enhance performance (Lee et al., 2020). Note that we also insert `[SEP]` between every two adjacent utterances.

Explicit Discourse Graph. We first construct the explicit discourse graph as a Levi graph (Levi, 1942) which turns the labeled edges into additional vertices. Suppose $G = (V, E, R)$ is the graph constructed in the following way: if utterance U_1 is the continuation of utterance U_2,

---

[1] This discourse parser gives a state-of-the-art performance on STAC so far

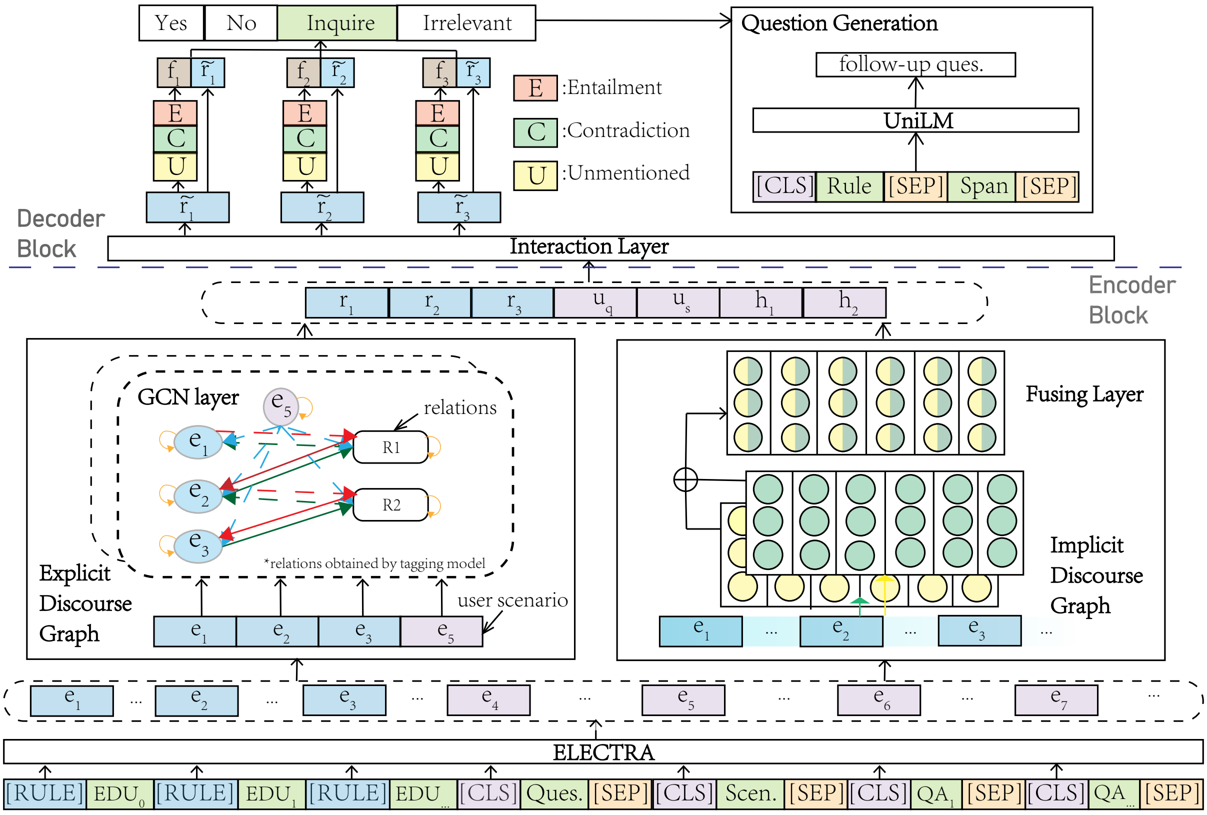

Figure 2: The overall structure for our proposed model. With segmented EDUs and tagged relations, the inputs including user initial question, user scenario and dialog history are sent for embedding and graph modeling to make the final decision. If the decision is *Inquire*, the question generation stage will be activated and use the under-specified span of rule document to generate a follow-up question.

we add a directed edge $e = (U_1, U_2)$ with relation R assigned to *Continuation*. The corresponding Levi graph can be expressed as $G = (V_L, E_L, R_L)$ where $V_L = V \cup R$. E_L is the set of edges with format $(U_1, Continuation)$ and $(Continuation, U_2)$. As for R_L, previous works such as (Marcheggiani and Titov, 2017; Beck et al., 2018) designed three types of edges $R_L = default, reverse, self$ to enhance information flow. Here with our settings, we extend it into six types: *default-in*, *default-out*, *reverse-in*, *reverse-out*, *self*, *global*, corresponding to the direction of the edges towards the relation vertices. An example of constructing Levi graph is shown in Figure 3. To construct the discourse structure of other elements, a global vertex representing user scenario is added and connected with all the other vertices.

We use a relational graph convolutional network (Schlichtkrull et al., 2018) to implement explicit discourse graph as the traditional GCN is not able to handle multi-relation graphs. For utterance and scenario vertices, we employ the encoding results of `[RULE]` and `[CLS]` in Section 3.1. For relation vertices, we look up in the embedding table to get the initial representation. Given the initial representation h_p^0 of every node v_p, the feed-forward or the message-passing process can be written as:

$$h_p^{(l+1)} = \text{ReLU}\Big(\sum_{r \in R_L} \sum_{v_q \in \mathcal{N}_r(v_p)} \frac{1}{c_{p,r}} w_r^{(l)} h_q^{(l)}\Big), \tag{1}$$

where $\mathcal{N}_r(v_p)$ denotes the neighbors of node v_p under relation r and $c_{p,r}$ is the number of those nodes. $w_r^{(l)}$ is the learnable parameters of layer l.

Because the total 16 relations cannot be treated equally, e.g. relation *Contrast* is much more important than the relation *Continuation*, we introduce the gating mechanism (Marcheggiani and Titov, 2017). The basic idea is to calculate a value between 0 and 1 for information passing control.

$$g_p^{(l)} = \text{Sigmoid}(h_p^{(l)} W_{r,g}), \tag{2}$$

where $W_{r,g}^{(l)}$ is a learnable parameter under relation type r of the l-th layer. Finally, the forward process

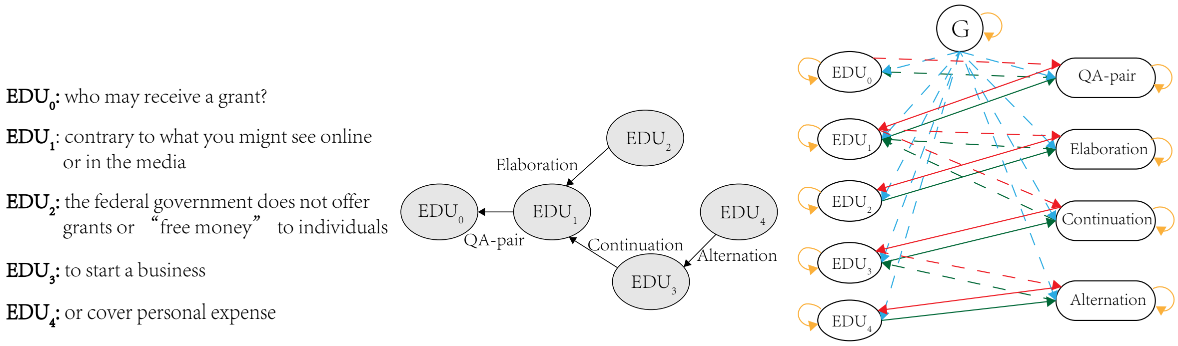

(a) separated EDUs (b) the original graph (c) the Levi graph

Figure 3: Processes turning a sample dialog into Levi graph representing discourse relations.

of gated GCN can be represented as:

$$h_p^{(l+1)} = \mathrm{ReLU}\Big(\sum_{r \in R_L} \sum_{v_q \in \mathcal{N}_r(v_p)} g_q^{(l)} \frac{1}{c_{p,r}} w_r^{(l)} h_q^{(l)}\Big), \tag{3}$$

Implicit Discourse Graph. Implicit discourse graph aims at digging the salient latent interactions inside rule document. Each token i in rule EDU is represented as a vertex in the graph. We use adjacent matrices to express implicit discourse graph. Two types of matrices M_l and M_c are introduced standing for local and contextualized information:

$$M_l[i,j] = \begin{cases} 0 & \text{if } I_i = I_j, \\ -\infty & \text{otherwise.} \end{cases} \tag{4}$$

$$M_c[i,j] = \begin{cases} 0 & \text{if } I_i \neq I_j, \\ -\infty & \text{otherwise.} \end{cases} \tag{5}$$

where I_i is the index of token i in EDU. Thus the information containing in rule document are decoupled in two separate aspects. Using multi-head self-attention to encode the graph and denote the length of the whole rule document as s, embedding dimension as d, we will get the following:

$$G_i = \mathrm{MHSA}(E, M_i),\ i \in \{l, c\}, \tag{6}$$

where $G_i \in \mathbb{R}^{s \times d}$ and E is the embedding result from PrLM. MHSA denotes the multi-head self-attention (Vaswani et al., 2017).

After enough interactions inside rule EDUs, we then fuse the information (Liu et al., 2020) of these two implicit discourse graphs-like items[2] above in a gated manner by considering both the original and graph encoding representation of rule document.

$$\begin{aligned} \tilde{E_1} &= \mathrm{ReLU}(\mathrm{FC}([E, G_l, E - G_l, E \odot G_l])), \\ \tilde{E_2} &= \mathrm{ReLU}(\mathrm{FC}([E, G_c, E - G_c, E \odot G_c])), \\ g &= \mathrm{Sigmoid}(\mathrm{FC}([\tilde{E_1}, \tilde{E_2}])), \\ C &= g \odot G_l + (1 - g) \odot G_c, \end{aligned} \tag{7}$$

where FC is the fully-connected layer and $C \in \mathbb{R}^{s \times d}$. We take the calculated result of the original `[RULE]` to stand for the updated rule EDUs from C, denoted as c_i.

3.3 Decoding Block

Interaction Layer. We use an interaction layer to attend to all available information so far to learn in a systematic way. A self-attention layer (Vaswani et al., 2017) is adopted here allowing all the rule EDUs and other elements to attend to each other. Let $[r_1, r_2, ...; u_q; u_s; h_1, h_2, ...]$ denote all the representations, r_i is the combined sentence-level representation of explicit and implicit discourse graph, u_q, u_s and h_i stand for the representation of user question, user scenario and dialog history respectively. After encoding, the output can be displayed as $[\tilde{r_1}, \tilde{r_2}, ...; \tilde{u_q}, \tilde{u_s}; \tilde{h_1}, \tilde{h}2...]$.

Decision Making. Similar to existing works (Zhong and Zettlemoyer, 2019; Gao et al., 2020a,b), we apply an entailment-driven approach for decision making. A linear transformation tracks the fulfillment state of each rule EDU among Entailment, Contradiction and Unmentioned. At last, the decision is made according to:

$$f_i = W_f \tilde{r_i} + b_f \in \mathbb{R}^3, \tag{8}$$

---

[2] Taking self-attention weights as edges connecting representations (as node), it can be seen as graph as well.

Model	Dev Set				Test Set			
	Decision Making		Question Gen.		Decision Making		Question Gen.	
	Micro	Macro	BLEU1	BLEU4	Micro	Macro	BLEU1	BLEU4
NMT (Saeidi et al., 2018)	-	-	-	-	44.8	42.8	34.0	7.8
CM (Saeidi et al., 2018)	-	-	-	-	61.9	68.9	54.4	34.4
BERTQA (Zhong and Zettlemoyer, 2019)	68.6	73.7	47.4	54.0	63.6	70.8	46.2	36.3
UcraNet (Verma et al., 2020)	-	-	-	-	65.1	71.2	60.5	46.1
BiSon (Lawrence et al., 2019)	66.0	70.8	46.6	54.1	66.9	71.6	58.8	44.3
E^3 (Zhong and Zettlemoyer, 2019)	68.0	73.4	67.1	53.7	67.7	73.3	54.1	38.7
EMT (Gao et al., 2020a)	73.2	78.3	67.5	53.2	69.1	74.6	63.9	**49.5**
DISCERN (Gao et al., 2020b)	74.9	79.8	65.7	52.4	73.2	78.3	**64.0**	49.1
DGM (ours)	**78.6**	**82.2**	**71.8**	**60.2**	**77.4**	**81.2**	63.3	48.4

Table 1: Results on the blind held-out test set and the dev set of ShARC end-to-end task. Micro and Macro stand for Micro Accuracy and Macro Accuracy respectively.

where f_i is the score predicted for the three labels of the i-th condition. This prediction is trained via a cross entropy loss for multi-classification problems:

$$\mathcal{L}_{entail} = -\frac{1}{N}\sum_{i=1}^{N} \log \text{softmax}(f_i)_r, \qquad (9)$$

where r is the ground-truth state of fulfillment.

After obtaining the state of every rule, we are able to give a final decision towards whether it is *Yes*, *No*, *Inquire* or *Irrelevant* by attention.

$$\begin{aligned} \alpha_i &= w_\alpha^T[f_i;\tilde{r_i}] + b_\alpha \in \mathbb{R}^1, \\ \tilde{\alpha_i} &= \text{softmax}(\alpha)_i \in [0,1], \\ z &= W_z \sum_i \tilde{\alpha_i}[f_i;\tilde{r_i}] + b_z \in \mathbb{R}^4, \end{aligned} \qquad (10)$$

where α_i is the attention weight for the i-th decision and z has the score for all the four possible states. The corresponding training loss is:

$$\mathcal{L}_{decision} = -\log \text{softmax}(z)_l, \qquad (11)$$

The overall loss for decision making is:

$$\mathcal{L} = \mathcal{L}_{decision} + \lambda \mathcal{L}_{entail}. \qquad (12)$$

Qustion Generation. If the decision is made to be *Inquire*, the machine need to ask a follow-up question to further clarify. Question generation in this part is mainly based on the uncovered information in the rule document, and then that information will be rephrased into a question. We predict the position of an under-specified span within a rule document in a supervised way. Following Devlin et al. (2019), our model learns a start vector $w_s \in \mathbb{R}^d$ and end vector $w_e \in \mathbb{R}^d$ to indicate the start and end positions of the desired span:

$$span = \arg\min_{i,j,k}(w_s^T t_{k,i} + w_e^T t_{k,j}), \qquad (13)$$

where $t_{k,i}$ denote the i-th token in the k-th rule sentence. The ground-truth span labels are generated by calculating the edit-distance between the rule span and the follow-up questions. Intuitively, the shortest rule span with the minimum edit-distance is selected to be the under-specified span. Finally, we concatenate the rule document and the predicted span as an input sequence to finetune UniLM (Dong et al., 2019) and generate the follow-up question.

4 Experiments

4.1 Experimental Setup

Dataset. We conduct experiments on ShARC dataset, the current CMR benchmark[3] collected by Saeidi et al. (2018). It contains up to 948 dialog trees clawed from government websites. Those dialog trees are then flattened into 32,436 examples consisting of *utterance_id*, *tree_id*, *rule document*, *initial question*, *user scenario*, *dialog history*, *evidence* and the *decision*. It is worth noting that evidence is the information that we need to extract from user information and thus will not be given in the testing phase. The sizes of train, dev and test are 21,890, 2,270 and 8,276 respectively. We also showed the generalizability of our model on the Multi-Turn Dialogue Reasoning (MuTual) dataset (Cui et al., 2020), which has 8,678 multiple choice samples and is divided into 7,376, 651, 651 of train, dev and test sets respectively.

Evaluation. For the decision-making subtask, ShARC evaluates the Micro- and Macro- Acc. for the results of classification. If both the prediction and ground truth of decision is *Inquire*,

---

[3] Leaderboard can be found at website `https://sharc-data.github.io/leaderboard.html`

BLEU(Papineni et al., 2002) score (particularly BLEU1 and BLEU4) will be evaluated on the follow-up question generation subtask.

Implementation Details. For rule EDU relation prediction, we keep all the default parameters of the original discourse relation parser[4], with $F1$ score achieving 55. In the decision-making stage, we finetune an ELECTRA-based model. The dimension of hidden states is 1024 for both the encoder and decoder. The training process uses Adam (Kingma and Ba, 2015) for 5 epochs with learning rate set to 5e-5. We also use gradient clipping with a maximum gradient norm of 2, and a total batch size of 16. In the question generation stage, for the sake of consistency, we also use an ELECTRA-based model for span extraction. For UniLM, we finetune it with a batch size of 16, a learning rate of 2e-5 and beam size is set to 10 for inference. It takes 3-4 hours for training on a single TITAN RTX 2080Ti GPU (24GB memory).

4.2 Results

Table 1 shows the results of DGM and all the baseline models for the End-to-End task on the blind held-out test set of ShARC[5]. Evaluating results indicate that DGM outperforms the baselines in most of the metrics. In particular, DGM outperforms the previous state-of-the-art model DISCERN by 4.2% in Micro Acc. and 2.9% in Macro Acc.

To test the generality of DGM on other different PrLMs and to do a fair comparison with previous models, We alter the underlying PrLMs to other variants in DGM and the previous state-of-the-art model DISCERN respectively. The results on the dev set of ShARC are shown in Table 2. In the first place, DGM performs better than DISCERN on all the PrLMs, which indicates the all-round superiority of DGM. Additionally, results on ELECTRA is generally better than that of BERT and RoBERTa. This indicates that ELECTRA is an even better trained PrLM. By the aforementioned analysis, our DGM can generally perform well on widely-used PrLMs.

PrLMs	Micro Acc.		Macro Acc.	
	DISCERN	DGM	DISCERN	DGM
BERT$_{base}$	69.8	70.4	75.3	76.0
RoBERTa$_{base}$	74.9	75.8	79.8	80.2
ELECTRA$_{base}$	75.2	75.5	79.7	80.4
BERT$_{large}$	72.8	73.0	77.8	78.0
RoBERTa$_{large}$	76.1	76.6	80.6	81.0
ELECTRA$_{large}$	77.2	78.6	80.3	82.2

Table 2: Performance of DISCERN and DGM on different PrLMs on the dev set of ShARC.

In addition, Table 3 lists the class-wise classification accuracy of our model. Results demonstrate that our model performs quite satisfactorily for all classification subtasks, outperforming all other models in three of all four subtasks though a minor behind on the *Irrelevant* subtask. Compared to competent models, our model boosts the performance with a great gain to judge whether the user's requirements need further inquiry or are already fulfilled. It is worth noting that the *Inquire* subtask is the most fundamental one among all subtasks required by the concerned CMR. The superiority of our model for this core subtask shows that our DGM model indeed effectively captures the complicated interactions among all the concerned document rules and scenarios.

Models	Total	Yes	No	Inquire	Irrelevant
BERTQA	63.6	61.2	61.0	62.6	96.4
E^3	68.0	65.9	70.6	60.5	96.4
UrcaNet	65.9	63.3	68.4	58.9	95.7
EMT	73.2	70.5	73.2	70.8	98.6
DISCERN	75.2	71.9	75.8	73.3	**99.3**
DGM (ours)	**77.8**	**75.2**	**77.9**	**76.3**	97.8

Table 3: Class-wise decision prediction accuracy on the dev set of ShARC.

5 Analysis

5.1 Ablation Study

To investigate the impacts of different graphs, we conducted an ablation study on the decision-making subtask which is the vital part of our model, directly influencing the results afterward. Detailed results on the dev set of ShARC in Table 4 show that both the explicit and implicit discourse graph are indispensable as removing any one of them causes a performance drop (1-3 points) on both Macro Acc. and Micro Acc. Especially, these two metrics drop by a great margin as we remove the explicit discourse graph, which shows that explicit discourse relation reasoning is crucial in CMR.

---

[4] `https://github.com/shizhouxing/DialogueDiscourseParsing`

[5] As indicated in (Gao et al., 2020a,b), the question generation results normally suffer from randomness. As the focus of this task is the decision making task like previous studies.

Example	Structure Type	Discourse Relation	Decision
Snippet: Export products made from endangered animals: (special rules)$_0$ If the animal is classified as B, C or D$_1$ you do not need to do anything$_2$ **Scenario:** I do not have medicare for this person. **Question:** Can I export products made from this animal?	Simple	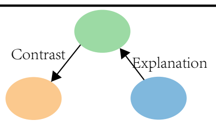	Yes × No Irrelevant Inquire ✓
Snippet: Who may receive a grant?$_0$ Contrary to what you might see online in the media$_1$ the federal government does not offer grants or free money to individuals$_2$ to start a business$_3$ or cover personal expenses$_4$ **Scenario:** I live in Norway with my wife and children. **Question:** Can I get a grant from federal government?	Disjunction	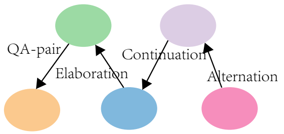	Yes No × Irrelevant Inquire ✓
Snippet: In order to qualify for this benefit program$_0$ homeowners and renters must have sustained damage B, C or D$_1$ and be located in a disaster declared county$_2$ **Scenario:** (empty) **Question:** Do I qualify for this loan?	Conjunction	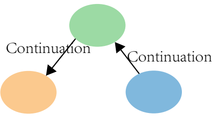	Yes No Irrelevant × Inquire ✓
Snippet: Going abroad$_0$ If your trip is going to last longer than 8 or 12 weeks, contact the Tax Credit Office within a month$_1$ your tex credits will end unless$_2$ you get UK benifits or State Pension$_3$ and you live in another European country with a child$_4$ **Scenario:** (empty) **Question:** Will my tex credit end?	Complex	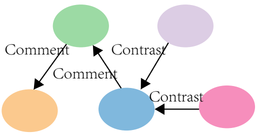	Yes No ✓ Irrelevant Inquire ×

Figure 4: Examples selected from the dev set of ShARC where DISCERN fails but our model succeeds.

Also, we can see that adding the special token `[RULE]` indeed conduce to the performance.

Models	Macro Acc.	Micro Acc.
DGM	82.2	78.6
w/o Explicit Discourse Graph	79.8	75.2
w/o Implicit Discourse Graph	81.3	76.7
w/o both	77.3	71.7
w/o [RULE]	81.6	77.9

Table 4: Ablation study of our model for decision making subtask on the dev set of ShARC.

5.2 User Scenario Interpretation

In DGM, by injecting the user scenario as the global node in the explicit discourse graph we intend to improve the interpretation ability of our model with respect to user scenarios. To test the effect of the proposed model on scenario interpretation, we create a subset based on the dev set consisting of 761 samples that have user scenarios and an empty dialog history. The results on the decision-making subtask in Table 5 shows that our model can greatly improve the interpretation of user scenarios by surpassing DISCERN 11.8% and 14.8% of Macro Acc. and Micro Acc. respectively. In particular, DGM outperforms DISCERN by a large margin in every class of decision.

Models	Macro	Micro	Yes	No	Irrelevant	Inquire
DISCERN	63.5	60.2	35.3	50.7	100.0	67.8
DGM	**75.3**	**75.0**	**58.3**	**62.8**	**100.0**	**79.1**

Table 5: Results for decision making over user scenario subset of the ShARC dev set.

Manually analyzing the predicted results also indicates that DGM is capable of various reasoning including numerical reasoning, commonsense reasoning and rule document paraphrasing. For example, for numerical reasoning, given a scenario *"I plan on being away for five months before returning"*, DGM is able to match that with *"your tax credits will stop if you expect to be away for one year or more"* in the rule document.

5.3 Rule Document Interpretation

To see how DGM interpret the rule document, we analyzed the predictions from DGM and DISCERN on the dev set of ShARC to see how our model fixes the erroneous cases made by DISCERN.

We selected four types of rule structures and the representative examples are shown in Figure 4. It can be seen that the discourse relation tagged for the rule document can well represent the real relation in the discourse. For example, in the third case, *"homeowners and renters must have sustained damage B, C or D"* is the continuation of *"in order to qualify for this benefit program"* and *"be located in a disaster declared county"* is the continuation of it. The discourse relation informs that *"be located in a disaster declared county"* and *"have sustained damage B, C or D"* are two conditions one must obey to be qualified. Generally, *Continuation* may indicate that two rules have some conjunctive relations while *Alternation* denotes a disjunctive relation. All the relations together characterize the complex rule relations and thus are vital in decision making. Statistics regarding relations can be found in Table 6. The contextualized information containing in the rule document learned by the implicit discourse graph also contributes to the overall performance as it digs the semantically rich representations of rule EDUs.

Relation Types	Train Set	Dev Set
Comment	28756	2374
Clarification_question	330	69
Elaboration	639	82
Acknowledgement	6242	815
Continuation	7317	1090
Explanation	10831	1155
Conditional	1445	139
Question-answer_pair	1824	468
Alternation	896	323
Result	664	0
Correction	14	0
Contrast	16523	1595

Table 6: Statistics analysis of relation types of the train and dev on ShARC. "Comment", "Continuation", "Explanation" and "Contrast" constitutes the majority of the discourse relations.

5.4 Generalizability Evaluation

To verify DGM's generalizability and show that it can be smoothly applied to a broad type of QA tasks, We conducted experiments on a representative dialogue reasoning dataset MuTual. It is modified from Chinese high school English listening comprehension test data. It consists of 8860 annotated dialogues, namely, 7088 training samples, 886 developing samples and 886 testing samples. For each example, there is a dialogue history following by four candidate responses. Each candidate is relevant to the dialogue context but only one of them is logically correct. Our aim is to predict the correct answer given dialogue history and response candidates.

To apply DGM on MuTual, we first annotated the utterances of dialogue history of their discourse relations. Then pass the dialogue and the response candidates into a pre-trained language model to get the representation of each utterance. Armed with these representations and discourse relations, we are now able to construct the explicit discourse graph. Here, we set the global representation [CLS] (Devlin et al., 2019) of dialogue history as the global node. The implicit discourse graph can be constructed as Section 3.2 stated.

For the sake of computational efficiency, the maximum number of utterances is set to be 25. The concatenated context, response candidate in one sample is truncated or padded to be of length 256. We use ELECTRA as the PrLM and AdamW (Loshchilov and Hutter, 2019) as the optimizer for training. The batch size is 24 and the learning rate is 6e-6. We run a total of 3 epochs and select the model of the best results in the development set.

Table 7 displays the results on MuTual, which shows that DGM achieves a consistent improvement on the performance with respect to all the corresponding metrices.

Models	Dev Set			Test Set		
	R_4@1	R_4@2	MRR	R_4@1	R_4@2	MRR
ELECTRA	90.6	97.7	94.9	90.0	97.9	94.6
DGM	91.3	98.3	95.3	90.7	98.2	95.1

Table 7: Results on the dev and test set of MuTual dataset

6 Conclusions

In this paper, we presented a novel Dialogue Graph Modeling framework for Conversational Machine Reading. Our DGM consists of two complementary graphs which respectively capture both explicit and implicit interactions among multiple complicated elements in the challenging task, in which Explicit Discourse Graph is for extra knowledge learning with tagged EDU discourse relations while Implicit Discourse Graph helps with inside rule document understanding. Experiments on ShARC show the effectiveness by achieving a new state-of-the-art result. Our method may be smoothly applied to a broad type of QA tasks, such as our practice on the MuTual dataset that also achieves a consistent performance.

Acknowledgments

We would like to thank all the anonymous reviewers for their helpful comments and suggestions. Also thanks to Max Bartolo for evaluating our submitted models on the hidden test set.

References

Nicholas Asher, Julie Hunter, Mathieu Morey, Benamara Farah, and Stergos Afantenos. 2016. Discourse structure and dialogue acts in multiparty dialogue: the STAC corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2721–2727.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 571–583.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283.

Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 357–362.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2174–2184.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. MuTual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1406–1416.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418.

Yifan Gao, Chien-Sheng Wu, Shafiq Joty, Caiming Xiong, Richard Socher, Irwin King, Michael Lyu, and Steven C.H. Hoi. 2020a. Explicit memory tracker with coarse-to-fine reasoning for conversational machine reading. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 935–945.

Yifan Gao, Chien-Sheng Wu, Jingjing Li, Shafiq Joty, Steven C.H. Hoi, Caiming Xiong, Irwin King, and Michael Lyu. 2020b. Discern: Discourse-aware entailment reasoning network for conversational machine reading. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2439–2449.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks.

Carolin Lawrence, Bhushan Kotnis, and Mathias Niepert. 2019. Attending to future tokens for bidirectional sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1–10.

Haejun Lee, Drew A. Hudson, Kangwook Lee, and Christopher D. Manning. 2020. SLM: Learning a discourse language representation with sentence unshuffling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1551–1562.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*. University of Calcutta.

Chang Li and Dan Goldwasser. 2019. Encoding social information with graph convolutional networks forPolitical perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604.

Jing Li, Aixin Sun, and Shafiq R. Joty. 2018. Segbot: A generic neural text segmentation model with pointer network. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4166–4172.

Longxiang Liu, Zhuosheng Zhang, Hai Zhao, Xi Zhou, and Xiaodong Zhou. 2020. Filling the gap of utterance-aware and speaker-aware representation for multi-turn dialogue. *ArXiv*, abs/2009.06504.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ying Luo and Hai Zhao. 2020. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6408–6418.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *ArXiv:1611.09268v2*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2263–2270.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1006–1017.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2087–2097.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne vanden Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607.

Zhouxing Shi and Minlie Huang. 2019. A deep sequential model for discourse parsing on multi-party dialogues. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7007–7014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Nikhil Verma, Abhishek Sharma, Dhiraj Madan, Danish Contractor, Harshit Kumar, and Sachindra Joshi. 2020. Neural conversational QA: Learning to reason vs exploiting patterns. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7263–7269.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3740–3752.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020a. Semantics-aware BERT for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9628–9635. AAAI Press.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020b. Sg-net: Syntax-guided machine reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9636–9643. AAAI Press.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2021. Retrospective reader for machine reading comprehension. In *AAAI 2021*.

Zhuosheng Zhang, Hai Zhao, and Rui Wang. 2020c. Machine reading comprehension: The role of contextualized language models and beyond. *arXiv preprint arXiv:2005.06249*.

Victor Zhong and Luke Zettlemoyer. 2019. E3: Entailment-driven extracting and editing for conversational machine reading. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2310–2320.

IndoCollex: A Testbed for Morphological Transformation of Indonesian Colloquial Words

Haryo Akbarianto Wibowo[*] **Made Nindyatama Nityasya**[*] **Afra Feyza Akyürek**[⋆]
Suci Fitriany[*] **Alham Fikri Aji**[*] **Radityo Eko Prasojo**[*•] **Derry Tanti Wijaya**[⋆]
[*] Kata.ai Research Team, Jakarta, Indonesia
[⋆] Department of Computer Science, Boston University
[•] Faculty of Computer Science, Universitas Indonesia
`{haryo,made,suci,aji,ridho}@kata.ai`

Abstract

Indonesian language is heavily riddled with colloquialism whether in written or spoken forms. In this paper, we identify a class of Indonesian colloquial words that have undergone morphological transformations from their standard forms, categorize their word formations, and propose a benchmark dataset of Indonesian Colloquial Lexicons (IndoCollex) consisting of informal words on Twitter expertly annotated with their standard forms and their word formation types/tags. We evaluate several models for character-level transduction to perform morphological word normalization on this testbed to understand their failure cases and provide baselines for future work. As IndoCollex catalogues word formation phenomena that are also present in the non-standard text of other languages, it can also provide an attractive testbed for methods tailored for cross-lingual word normalization and non-standard word formation.

1 Introduction

Indonesian language is one of the most widely spoken languages in the world with around 200 million speakers. Despite its large number of speakers, in terms of NLP resources, Indonesian language is not very well represented (Joshi et al., 2020). Most of its data are in the form of unlabeled web and user generated contents in online platforms such as social media, which are noisy and riddled with colloquialism which poses difficulties for NLP systems (Baldwin et al., 2013a; Eisenstein, 2013a).

Traditionally, the majority of Indonesian colloquial or informal lexicons are borrowed words from foreign or local dialect words, and sometimes with phonetic and lexical modifications.[1] Increasingly however, Indonesian colloquial words are more commonly a morphological transformation[2] of their standard counterparts.[3] Despite these evolving lexicons, existing research on Indonesian word normalization has largely (1) relied on creating static informal dictionaries (Le et al., 2016), rendering normalization of unseen words impossible, and (2) for the specific task of sentiment analysis (Le et al., 2016) or machine translation (Guntara et al., 2020), with no direct implication to word normalization in general. Given the obvious utility of creating NLP systems that can normalize Indonesian informal data, we believe that the bottleneck is that there is no standard open testbed for researchers and developers of such system to test the effectiveness of their models to these colloquial words.

In this paper, we introduce IndoCollex, a new, realistic dataset aimed at testing normalization models to these phenomena. IndoCollex is a professionally annotated dataset, where each informal word is paired with its standard form and expertly annotated with its word formation type. The words are sampled from Twitter across different regions, therefore contain naturally occurring Indonesian colloquial words.

We benchmark character-level sequence-to-sequence transduction with LSTM (Deutsch et al., 2018; Cotterell et al., 2018) and Transformer (Vaswani et al., 2017) architectures, as well as a rule-based approach (Eskander et al., 2013; Moeljadi et al., 2019) on our data to understand their success and failure cases (§7.2, §7.3) and to provide baselines for future work. We also test methods for data augmentation in machine translation (back-translation), which to the best of our knowledge has never been applied to

---

[1] For example, *gue*, a common informal form of *aku* ('I', 'me'), is a word that originates from the Betawi dialect.

[2] We used the term morphological transformations broadly here to include word form changes at the respective interfaces of grammar (phonology, syntax, and semantics), following the definition by Trips (2017).

[3] For example, *laper*, a common informal form of *lapar* ('hungry'), is a phonetic change from its standard form.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3170–3183
August 1–6, 2021.

character-level morphological transformation, and observe that adding back-translated data to train transformer improves its performance for normalizing informal words. We also test models in the other direction: generating informal from formal words, which can be useful for generating possible lexical replacements to standard text (Belinkov and Bisk, 2018).

2 Related Work

With the advent of social media and other user generated contents on the web, non-standard text such as informal language, colloquialism and slang become more prevalent. Concurrently, the rise of technologies like unsupervised language modeling opened up a new avenue for low-resource languages which lack annotated data for supervision. These systems typically only require large amounts of unlabeled text to train (Lample and Conneau, 2019; Brown et al., 2020). However, even when NLP systems require only unlabeled data to train, the varying degrees of formalism between different sources of monolingual data pose domain adaption challenges to NLP systems which are trained on one source (e.g. Wikipedia) to transfer to another (e.g. social media) (Eisenstein, 2013b; Baldwin et al., 2013b; Belinkov and Bisk, 2018; Pei et al., 2019). Worse yet, for an overwhelming majority of lower resource languages, unstructured and unlabeled text on the Internet is often the sole source of data to train NLP systems (Joshi et al., 2020). Therefore, addressing the formalism discrepancy will augment the types of web texts which can be employed in language technologies, especially for languages such as Indonesian which are subject to a high degree of informalism as will be discussed.

While this motivates research on training systems that are robust to non-standard data (Michel and Neubig, 2018; Belinkov and Bisk, 2018; Tan et al., 2020b,a), one intuitive direction is to normalize colloquial language use. Most of the work on colloquial language normalization has been done at the sentence-level: for colloquial English (Han et al., 2013; Lourentzou et al., 2019), Spanish (Cerón-Guzmán and León-Guzmán, 2016), Italian (Weber and Zhekova, 2016), Vietnamese (Nguyen et al., 2015), and Indonesian (Barik et al., 2019; Wibowo et al., 2020). However, research on the linguistic phenomena of non-standard text (Mattiello, 2005), which argues that slang words exhibit extra-grammatical morphological properties (such as portmanteaus, clipping) that distinguish them from the standard form, justifies the need for word-level normalization.

Word-level normalization also has its merit because due to its much lower hypothesis space, models can be trained using significantly smaller amount of data (e.g., compare SIGMORPHON's 10k examples to WMT's 10^6 at high-resource setting). Further, from our manual analysis of the top-10k most frequent Indonesian informal words we collected from Twitter, we find that around 95% of these words do not require context to normalize. Additionally, previous works such as Kulkarni and Wang (2018) have suggested that creating computational models for this generation of informal words can give us insights into the generative process of word formation in non-standard language. This is important because studies into the generative processes of word formation in non-standard text can deepen our understanding of non-standard text. Moreover, they are potentially applicable to many languages since word formation patterns are shared across languages (Štekauer et al., 2012), e.g., portmanteaus (such as *brexit*) have been found not only in English but also in many other languages such as Indonesian (Dardjowidjojo, 1979), Modern Hebrew (Bat-El, 1996), and Spanish (Piñeros, 2004). Finally, the studies may have broader applications including development of rich conversational agents and tools like brand name generators and headlines (Özbal and Strapparava, 2012).

Previous work that qualitatively catalogues or creates computational models for informal word formations such as shortening has mostly been in English, using LSTMs (Gangal et al., 2017; Kulkarni and Wang, 2018) or finite state machines (Deri and Knight, 2015) to generate informal words given the standard forms and the type of word formation. Most of the dataset: formal-informal word pairs labeled with their word formation used to train these models are also in English. Other dictionaries of informal English words include SlangNet (Dhuliawala et al., 2016), SlangSD (Wu et al., 2018), and SLANGZY (Pei et al., 2019). There is also a dataset that contains pairs of formal-informal Indonesian words (Salsabila et al., 2018), but they are not annotated with word formation mechanisms. To the best of our knowledge, ours is the first dataset of formal-informal lexicon in a language other than English that is annotated with their word formation types.

3 Indonesian Colloquialism

3.1 Indonesian Colloquial Words

Language evolves over time due to the process of language learning across generations, contact with other languages, differences in social groups, and rapid casual usages (Liberman et al., 2003). Each of these factors exists to a high degree in Indonesia, resulting in the constant evolution of its language due to contacts with over 700 local languages (Simons and Fennig, 2017), socioeconomic and education inequalities that result in varying level of adoption of the standard Indonesian (Azzizah, 2015), and the rise of social media usages with widespread *celeb* culture (Suhardianto et al., 2019; Heryanto, 2008) that causes new words to be invented and spread rapidly.

We catalog the following word formation types that are common in colloquial Indonesian.

1. **Disemvoweling**: elimination of some or all the vowels, e.g: *jangan* to *jgn* ('no' or 'don't'). Disemvoweling does not correspond to any phonetic change,
2. **Shortening or Clipping**: syllabic shortening of the original word, e.g: *internet* to *inet*. Unlike disemvoweling, shortening does imply phonetic change,
3. **Space/dash removal**: shortened version of writing Indonesian plural form, e.g.: *teman-teman* to *temanteman* or *teman2* ('friends'),
4. **Phonetic (sound) alteration**: slight change both in sound and spelling in text, but the number of syllables stay the same, e.g: *pakai* to *pake* or *pakek* ('use'),
5. **Informal affixation**: modification, addition or removal of affixes, e.g: *mengajari* to *ngajarin* ('to teach'),
6. **Compounding and acronym**: syllabic and letter compounds of one or more words akin to acronyms, abbreviations, and portmanteau, e.g: *anak baru gede* to *abg* ('teen'), *budak cinta* to *bucin* (literally, 'being a slave to love'),
7. **Reverse**: letter reversal, or colloquially known as "Boso Walikan" (Hoogervorst, 2014), e.g: *malang* (the name of a city in Indonesia) to *ngalam*.
8. **Loan words**: borrowed words, often from local language or English, e.g: *bokap* ('dad' in Betawi)
9. **Jargon**: tagline, terms that have been made into a popular term, e.g: *meneketehe*, from *mana aku tahu* (a jargon for 'how should I know?').

Some of the above transformations are also found in the literature of other languages, such as English and Korean. In English, disemvoweling was common during the texting (SMS) era in order to write faster and to save on message lengths e.g., *c u l8r* ('see you later'). Informal affixation (*cryin*, *sweet-ass*), compounding and portmanteaus (*btw*, *sexting*), and phonetic alteration (*dis is da wae*) are also present. In Korean, some compounded or shortened version of *Konglish* is also widely used (Khan and Choi, 2016), e.g., *chimaek* from *chicken* and *maek* ('beer'). Any insight we obtain through evaluating models on our dataset may therefore be of interest to other languages that share similar colloquial transformations; insights that may be increasingly paramount due to the rising prevalance of non-standard text in many languages on the web (Kulkarni and Wang, 2018; Joshi et al., 2020) and the challenges they pose to NLP systems (Belinkov and Bisk, 2018; Pei et al., 2019).

Loan word transformations that come from other languages require multilingual dictionaries/embeddings to normalize while jargons often require background knowledge. Aside from these two, we follow the previous work and hypothesize that the word formations that fall in other categories are mostly morphological transformations that can be learned at character-level (Kulkarni and Wang, 2018; Gangal et al., 2017). In §4, we describe how we curate this colloquial transformation data.

3.2 Indonesian Colloquialism Analysis

In this section, we motivate the importance of research on Indonesian colloquialism by highlighting their prevalence in Indonesian web text. We indeed observe that in its daily use Indonesians use colloquial Indonesian to generate contents in the web with (1) vocabularies that are different from formal Indonesian and (2) at a higher rate than colloquial use in the English language.

To compare colloquial and formal Indonesian (from Twitter and Lazada product reviews[4] and from Kompas news articles respectively (Tala, 2003)), we compute these dataset perplexities as well as their out-of-vocabulary (OOV) rates with respect to an Indonesian formal lexicon constructed from tokenizing Indonesian Wikipedia articles. For a fair comparison, we sample 3685 sentences from

---

[4] www.kaggle.com/grikomsn/lazada-indonesian-reviews

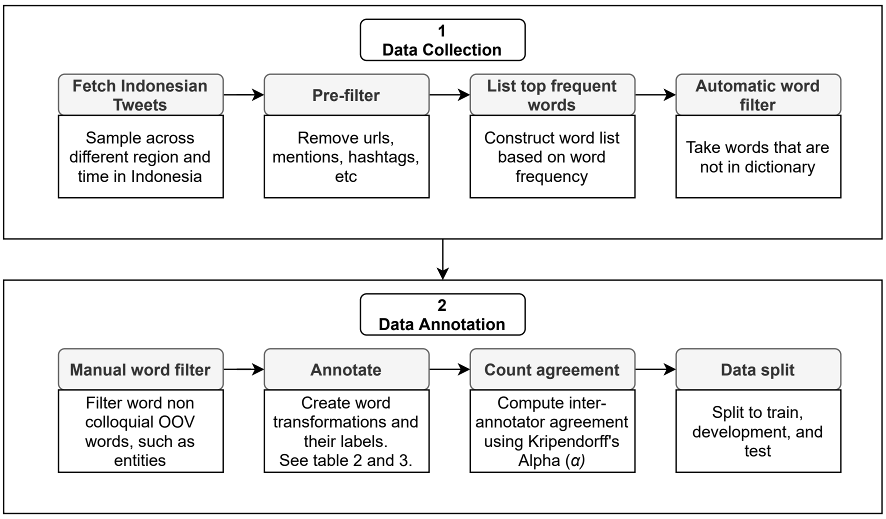

Figure 1: The data construction process composed of Data Collection and Data Annotation

each dataset based on the size of the smallest dataset. To compare to colloquial use in the English language, we also compare English tweets to an English formal lexicon constructed from English Wikipedia articles. We use Wikipedia to construct these lexicons to include named entities which are not typically present in traditional dictionaries.

Table 1 shows the OOV rate of the various datasets. Our OOV count excludes Twitter usernames, hashtags, mentions, URLs, dates, and numbers. To avoid rare words being captured as OOV, we also remove any token that only occurred once (shown as OOV-2) on the Table. We observe that the OOV rate of colloquial Indonesian is double the OOV rate of informal English. OOV of the formal Indonesian text (Kompas news) is low, as expected.

We use perplexity as a measure of impact of colloquialism beyond vocabulary usage and utilize a pre-trained Indonesian GPT-2 trained on Wikipedia[5] and Open AI's GPT-2[6] to calculate Indonesian and English data perplexities, respectively. Table 1 shows these perplexities.

Indonesian tweets have comparable perplexity as Lazada as they both use colloquial language. Both also have much higher perplexities than Kompas, implying that Indonesian LM finds that colloquial Indonesian is different than formal Indonesian. Similarly, English tweets have a higher perplexity compared to English Wikipedia (Radford et al., 2019). Notably, aside from Indonesian Twitter having around two times higher OOV rates: as high as 14.6% in OOV and 8.3% in OOV-2 than English Twitter, its perplexity too is significantly higher than English Twitter —suggesting that the non-standard word formation is a much more prominent issue when it comes to Indonesian, yet remains significantly under-researched.

Dataset	Lang.	OOV	OOV-2	Ppl
Twitter	ID	14.6%	8.3%	1617.0
Lazada review	ID	9.1%	7.0%	1824.3
Kompas news	ID	1.5%	1.1%	145.8
Wikipedia	ID	n/a	n/a	29.9
Twitter	EN	6.4%	4.5%	611.2
Wikipedia	EN	n/a	n/a	29.4

Table 1: OOV rates of Indonesian and English datasets

4 Data Collection and Annotation

Our dataset is constructed and manually annotated from a list of informal words obtained from Twitter. The data construction process is summarized in Figure 1. As an archipelago country, Indonesia is very diverse in local languages, which affects the way people use the Indonesian language. Hence, we sample 80 tweets per-day from March 2017 to May 2020, from each of the 34 provinces in Indonesia. We then select top 10k frequent tokens not appearing in our Wikipedia-based formal word dictionary and treat them as informal. Then we manually filter out from this list, OOV words that are not informal words such as product names or entities. Despite being sampled according to geolocation, we note that most of the informal words are more inclined to informal words commonly used in Jakarta. We suspect this is because Jakarta, being the center of Indonesian economy and pop culture (CITE), heavily influences the other regions through mainstream media. Further investigation on this aspect

---

[5] huggingface.co/cahya/gpt2-small-indonesian-522M

[6] huggingface.co/gpt2

is necessary and we leave this as a future work.

We assign four Indonesian native speakers[7], with formal education in linguistics and/or computational linguistics, to annotate each informal word with its standard form and label the pair with their word formation types according to our annotation codebook.[8] We annotate 9 different types of word formation mechanisms: disemvoweling, shortening, space/dash removal, phonetic (sound) alteration, affixation, compounding, reverse, loan word, and jargon. Since an informal word is often produced by stacking multiple transformations, we also annotate the transformation order, from the formal word to the informal. Some annotation examples are shown in Table 2. To simplify the transformation task, we assume single transformations and treat stacked transformations as a sequence of separate transformations. Words undergoing multiple transformations are broken down into different entries in our dataset. Ultimately, our dataset consists of parallel formal and informal Indonesian word pairs, each with its annotated word formation type from formal to informal. A sample of our dataset is shown in Table 3. Note that the same formal word with the same transformation may produce different informal words due to the open vocabulary of colloquial words.

Our dataset contains 3048 annotated word pairs[9] of which 2036 are those with morphological transformations (i.e., not loan words or jargons), which is comparable in size to other morphological transformation dataset such as the SIGMORPHON shared task (Cotterell et al., 2018). In comparison, Bengali, which is also a lower resource language comparable to Indonesian (Joshi et al., 2020), has 136 lemmas (and 4000 word forms) crowdsourced in the SIGMORPHON inflection dataset while our dataset has expertly annotated 1602 formal words (and 2036 informal variants).

In order to ensure the quality of our annotations, we sample 100 word pairs and compute Krippendorff's Alpha (α) (Hayes and Krippendorff, 2007) and Cohen's Kappa (κ) (Cohen, 1960) to measure agreement on word formation type annotations. The scores are $\alpha = 0.709$ $\kappa = 0.708$, showing that the annotators have substantial agreement on our dataset (Viera et al., 2005). We split the dataset into training, validation, and testing as in Table 4. Note that since reverse formation is quite rare, we augment the data and add additional reverse formation in the testing and validation sets.

In our experiments, we exclude loan word and jargon from the evaluation of character-level models, since these transformations are challenging, if not impossible to handle at the character-level alone without (1) additional resources such as multilingual dictionaries/embeddings and without (2) involving additional tasks such as translation.

5 Rule-Based Transformation Baseline

We believe that some formal to informal word formation mechanisms follow regular patterns. We manually define a rule-based system as one of our baselines (see Appendix). As we will demonstrate in the results section, there are several challenges entailed with a rule-based approach. Firstly, our rule-based transformation only works from formal to informal—as most of the colloquialism involves removing parts of the word, reverting from informal to formal Indonesian proves difficult for the rule-based system as it requires predicting the removed characters.

Secondly, the rule-based approach can not be universally applied. For example, in affixation, some Indonesian root words have sub-words similar to common morphological affixes in Indonesian such as *me-* or *-kan*. However, since these sub-words are part of the root words, they should not be removed/altered e.g., *membal* ('bouncy') cannot be transformed via informal affixation to *ngebal*, since *me-* in *membal* is part of the root word. Similarly, sound-alter transformation is applicable only to some words but not others e.g., *malam* ('night') can be altered to *malem*, but *galak* ('fierce') cannot be altered to *galek*. The rule of which words can be sound-altered seems arbitrary. In compounding, there is also no clear rule as to which abbreviation to use in different settings (e.g., *anak baru gede* is abbreviated to *ABG*, but *rapat kerja nasional* is abbreviated to *rakernas* instead of *RKN*). Lastly, as a single word may have multiple possible transformations that can apply, since rule-based system cannot rank these possible outputs, it randomly picks one of the candidates.

6 Character-Level Seq2Seq Models

Previous approaches for generating transformed words model the task as a character-level sequence-to-sequence (SEQ2SEQ) problem: the characters

---

[7] formally employed by our company, Kata.ai.

[8] https://github.com/haryoa/indo-collex

[9] Full dataset: https://github.com/haryoa/indo-collex

Informal Word	Annotated Formal	Annotated Word Formation Type
kuy	ayo (let's go)	original → sound-alter → reverse **note:** ayo → *yuk* → *kuy*
gpp	tidak apa-apa (no problem)	original → shortening → disemvoweling **note:** tidak apa-apa → *gapapa* → *gpp*
ngeselin	mengesalkan (annoying)	original → affixation → sound-alter **note:** mengesalkan → ngesal*in* → nges*e*lin

Table 2: Examples of informal words annotated with their formal versions, alongside the transformation sequences.

Source	Target	Word Formation Tag
ayo (formal of "let's go")	yuk (informal of "let's go")	sound-alter
ayo (formal of "let's go")	yuks (informal of "let's go")	sound-alter
yuk (informal of "let's go")	kuy (informal of "let's go")	reverse
yuks (informal of "let's go")	skuy (informal of "let's go")	reverse
kemarin (formal of "yesterday")	kmrn (informal of "yesterday")	disemvoweling
nasi goreng (fried rice)	nasgor	compounding
membuka (formal of "opening")	ngebuka (informal of "opening")	affixation

Table 3: Example entries in our colloquial transformation dataset.

Word Formation Tag	Train	Valid	Test
sound-alter	489	21	35
shorten	363	41	43
disemvoweling	323	30	31
affixation	165	30	20
space/dash removal	157	26	21
acronym	155	23	16
reverse	5	15	27
Total	1657	186	193

Table 4: Formal (F)→Informal (I) data distribution.

from the root word and an encoding of the desired transformation type are given as input to a neural encoder, and the decoder is trained to produce the transformed word, one character at a time (Gangal et al., 2017; Deutsch et al., 2018; Cotterell et al., 2017). In reality however, transformation types are often implied, but not given. For example, an Indonesian speaker will be able to transform the formal *tolong* ('help') to *tlg* given examples that *jangan* ('don't') can be transformed to *jgn*, even without the transformation type i.e., disemvoweling being specified. Thus, we also experiment with these SEQ2SEQ models for generating informal words from formal (and vice versa) without inputting any word formation tag to see if the models can induce the desired transformation type based on morphologically similar words in the training examples. We also use these models trained to generate outputs without word formation input to generate back-translated data to augment our training (§7.1).

6.1 BiLSTM

The dominant model for character-level transduction that have been applied to many tasks such as morphological inflection (Cotterell et al., 2017), morphological derivation (Deutsch et al., 2018), and informal word formation (Gangal et al., 2017) adopts a character-level SEQ2SEQ model that learns to generate a target word from its original form given the desired transformation. These models typically use bi-directional LSTM with attention (Luong et al., 2015) to learn these transformations as orthographic functions. For the task of morphological derivation, the SOTA model (Deutsch et al., 2018) also proposes a dictionary constraint approach where the decoding process is restricted to output tokens listed in the dictionary, which improves the accuracy of their model.

We evaluate this SOTA character SEQ2SEQ that leverages dictionary constraint (BiLSTM+Dict), whose code is publicly available,[10] on our data. Following their approach, we train this model for 30 epochs with a batch-size of 5 using Adam optimizer with initial learning-rate of 0.005, an embedding size of 20, and a hidden state size of 40. For the dictionary constraint, we construct dictionaries of formal words from Indonesian Wikipedia (§3.2) and informal words we collected from Twitter (i.e., words we collected from Twitter that do not appear in our Wikipedia-based formal word dictionary §4).

6.2 Transformer

Given that more recently Transformer has been shown to outperform standard recurrent models on several character-level transduction tasks including morphological inflection and historical text normalization, grapheme-to-phoneme conversion, and transliteration (Wu et al., 2020); we evaluate character-based Transformer model (Vaswani et al., 2017) on our dataset. We conduct hyperparame-

---

[10] github.com/danieldeutsch/derivational-morphology

ter tuning on the size of the character embeddings, the number of layers, and the number of attention heads of the Transformer. For training, we use Adam with an initial learning rate of 0.005, a batch size of 128 (following (Wu et al., 2020)), and train for a maximum of 200 epochs, returning the model with the least validation loss.

7 Experiment and Results

We evaluate standard character-level transduction models on our dataset to assess its difficulty. Our goal is not to train SOTA models for word normalization but rather to test these models for such task on our data, and elucidate what features of the data make it difficult.

7.1 Experiment Settings

We train and evaluate the BiLSTM+Dict and Transformer models on our dataset. The models are trained and evaluated in both directions: formal↔informal (F↔I) Indonesian. However, as mentioned previously, we only explore formal→informal (F→I) for the rule-based model. We also train the SEQ2SEQ models with and without inputting the word formation tag. Each experiment took about 3 hours on a K80 GPU.

Aside from training the models to transform formal↔informal words, we also use the Transformer model to predict the word formation tag $t \in T$, where T is the set of word formation types in our dataset, that best applies given an informal word and its corresponding formal form (I→F) or vice versa (F→I) (i.e., Transformer$_{(I \to F) \to T}$ and Transformer$_{(F \to I) \to T}$).

We experiment with using backtranslation (Sennrich et al., 2016), which has been used to learn novel inflections in statistical machine translation (Bojar and Tamchyna, 2011), at the character-level to increase the training data for I→F. Using Transformer$_{F \to I}$ model that performs best on the validation set, we generate informal words from the words in our formal dictionary sorted by frequencies. We experiment with generating $M = kN$ additional word pairs, where $k = \{1, 2, 3\}$ and N is the number of word pairs in the original training data. We similarly augment training data for F→I by using the Transformer$_{I \to F}$ model that performs the best on the validation set to generate formal words from our informal word dictionary.

To ensure that the augmented data has similar transformation distribution as the original training data, we predict the word formation type that best applies to each generated word pair using the Transformer$_{(I \leftrightarrow F) \to T}$ model that performs best on validation. For each word formation type, we add rM generated pairs with such type to our training data based on its ratio r in the original training.

Each model's performance is measured by the top-*1* and top-*10* accuracy. Since formal→informal transformation is rather flexible, we also capture the BLEU score of the model's output. We report performances of the hyperparameter-tuned models that perform best on the validation set.

7.2 Results

Our experiment results are shown in Table 5. Generally, Transformer models outperform all other models. Specifying the target word formation type improves the performance of both models. Backtranslation is also shown to improve the performance of the Transformer. Transformer with added backtranslation and word formation tag yields the best test performance in both directions.

We also observe that in average the performance of the models are higher in the I→F direction than F→I. We observe similar trends when predicting word formation types given word pairs. The accuracy of the Transformer$_{(I \to F) \to T}$ model that predicts the type that applies given an informal word and its corresponding formal form is 81.4%; which is significantly higher than the 65.0% accuracy of the Transformer$_{(F \to I) \to T}$ model that predicts the type given a formal word and its corresponding informal form. This may point to the inherent ambiguity of generating informal words from the formal words. Due to the open-vocabulary of informal words, there are potentially many ways to transform a formal word into informal forms.

Surprisingly, rule-based transformation outperforms BiLSTM+Dict and several non-optimal Transformer configurations in terms of top-*1* accuracy. However, rule-based transformation does not perform well in terms of top-*10* accuracy. We observe that the rule-based transformation does not always manage to produce 10 transformation candidates, therefore missing out on the extra chances to correctly guess the output.

7.3 Discussion

In this section, we discuss failures and success cases of the best performing model (Transformer) on our dataset, elucidate what the model learns,

Model	Informal to Formal					Formal to Informal				
	Dev		Test			Dev		Test		
	Top1	BLEU	Top1	Top1(	BLEU	Top1	BLEU	Top1	Top1(	BLEU
Rule-Based	-	-	-	-	-	27.9	43.9	34.7	53.4	50.2
BiLSTM + Dict	23.5	53.0	30.5	58.8	57.9	18.8	47.3	22.8	56.1	47.7
BiLSTM + Dict + word-formation tag	25.7	54.8	30.5	56.3	53.2	30.6	60.8	30.1	62.5	54.4
Transformer	30.1	59.3	27.9	60.6	61.6	19.4	53.3	21.2	56.5	48.7
Transformer + word-formation tag	**33.9**	64.4	35.8	65.9	61.6	31.2	59.2	22.3	54.2	48.2
Transformer + BT	31.7	65.7	32.1	66.4	**63.7**	22.6	57.1	24.4	58.4	53.4
Transformer + BT + word-formation tag	33.3	**66.5**	**37.4**	**70.2**	62.1	**36.6**	**69.2**	**35.8**	**67.5**	**57.0**

Table 5: Experiment Result for Informal and Formal Colloquial transformation.

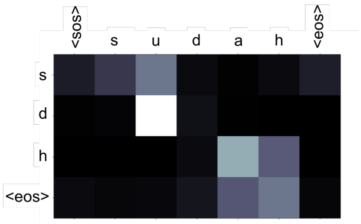

Figure 2: Attention matrix of *sudah* (F) → *sdh* (I) *without* word formation tag (column: source, row: target). The model learns to disemvowel implicitly by paying attention to the vowels and removing them.

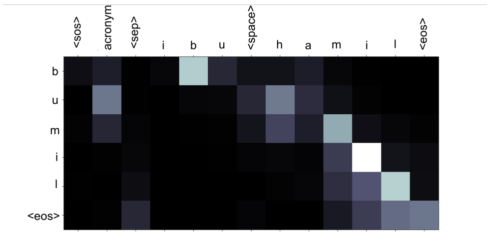

Figure 3: Attention matrix of *ibu hamil* F→I transformation *with* word formation tag (column: source, row: target). The model pays attention to the tag (acronym) while getting the prefix *bu-* from the first word and the suffix *-mil* from the second.

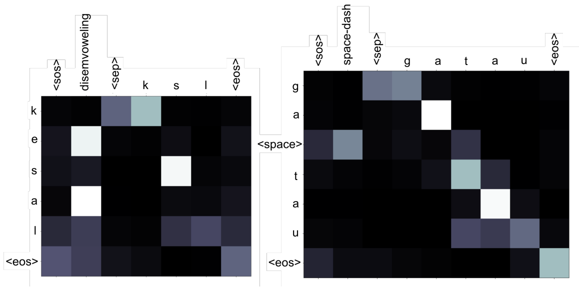

Figure 4: Attention matrix of *ksl* (I) → *kesal* (F) and *gatau* (I) → *ga tau* (F) *with* tag (column: source, row: target). The model learns to pay attention to the tag while regenerating the missing vowels and space.

and analyze features of the data that make it challenging for the model. As seen in Table 5, when the desired word formation is not given, the Transformer has worse performance when performing F→I transformation compared to I→F. This is because transforming from formal to informal has a higher level of ambiguity i.e., a word can be made informal by multiple possible word formations.

If the word formation type is not given, we observe that Transformer will learn to select the type implicitly. For example, it selects the disemvoweling mechanism implicitly as it pays attention to vowels in the word while removing them e.g., to correctly generate the informal *sdh* from the formal *sudah* (meaning, 'already') Figure 2). If the input consists of two words (separated by space), the model assumes the space/dash removal mechanism, paying attention to the characters before and after the space while removing the space e.g., given the word *ga tau* (meaning, 'don't know'), the model removes the space and correctly returns *gatau*.

However, the Transformer may select an incorrect transformation when the target word formation is not given e.g., the phrase *ibu hamil* ('pregnant mother') is often expressed as *bumil* (acronym). Without tag, the model performs a space/dash removal instead, and produced incorrect *ibuhamil*. Figure 3 shows how the model attends to the tag when it is given and applies the correct mechanism.

We observe that the model also attends to the tag when transforming the word in the reverse (I→F) direction e.g., the model pays attention to the tag while correctly generating the vowels of a disemvoweled words *ksl* to *kesal* ('annoyed') or the space between the compounded word *gatau* to *ga tau* (Figure 4).

In general, we observe that formal to informal transformation is challenging, since multiple valid informal words are possible even for a given word and word formation type. For example, *kamu* ('you') can be written informally as *km* or *kmu* both with the same disemvoweling transformation. Some word formation mechanisms are also ambiguous. For example, *budak cinta*'s acronym is *bucin* (using the *prefix* of the second word), whereas *ibu hamil*'s acronym is *bumil* (using the *suffix* of the second word). The acronym transformation seems to be applied on a case-by-case basis with no clear pattern. Reversing acronym to its original phrases is even more challenging (with or without tags) since it requires models to reconstruct the full phrase given minimum context e.g., reconstructing *anak layangan* ('tacky') from its acronym *alay*.

Another challenging transformation is affixation. Since *me-* and its different variants (*mem-*, *men-*, etc.) are common morphological prefixes in Indonesian, we observe that our best model, the Transformer, often puts *me-* in I→F affixation transformation, mistakenly transforming for example, *nyantai* ('to relax') into *menyantai* (expected: *bersantai*). This suggests that more training data may be needed to capture various affixation.

On the other hand, in sound alteration, we observe that Transformer successfully learns to sound-alter even when the word formation is not explicitly mentioned. For example, it learns to transform the informal *pake* ('to wear') to *pakai* (attending to the characters *e* when outputting *ai*), *kalo* ('if') to *kalau* (attending to the character *o* when outputting *au*), and *mauuu* ('want') to *mau* (attending to the characters *uuu* when outputting *u*).

8 Ethical Consideration

Normalizing informal Indonesian language might serve as a bridge to connect the generational gap in the use of the language, as the informal Indonesian language is more popular among the younger populace. Furthermore, it can potentially bridge linguistic differences across the Indonesian archipelago. Although we attempt to collect informal data from each province in Indonesia, the resulting informal dataset is still mostly Jakarta-centric, and further scraping and verification of the linguistic coverage is necessary for future work. Finally, as not every Indonesian speaks perfect standard Indonesian, having an NLP interface (such as chatbots) that can readily accept (process and understand via normalization) any kind of informality that might arise promotes inclusivity that all NLP research should strive for.

9 Conclusion and Future Work

We show that colloquial and formal Indonesian are vastly different in terms of OOV-rate and perplexity, which poses difficulty for NLP systems that are trained on formal corpora. This significant gap between train and test sets in terms of formalism may hinder progress in Indonesian NLP research. We propose a new benchmark dataset for Indonesian colloquial word normalization that contains formal-informal word pairs annotated with their word formation mechanisms. We test several dominant character-level transduction models as baselines on the dataset and observe that different word formation mechanisms pose different levels of difficulties to the models with transformation to informal forms being more challenging due to the higher degree of transformation variants. Through this dataset, we intend to provide a standard benchmark for Indonesian word normalization and foster further research on models, datasets and evaluation metrics tailored for this increasingly prevalent and important problem.

In the future, we are interested to use the context in which the words occur, either textual (e.g., sentences) or other modalities (e.g., images or memes), to improve word transformation (formal ↔ informal) by using the context as either implicit signal (Wijaya et al., 2017) or explicit signal for "translating" between the formal and informal word forms based on similarities between their sentence contexts (Feng et al., 2020; Reimers and Gurevych, 2020) or image contexts (Bergsma and Van Durme, 2011; Kiela et al., 2015; Hewitt et al., 2018; Khani et al., 2021). We are also interested to learn if simple clustering of contexts within which the words occur can help us learn the mapping between the formal and informal words similar to finding paraphrase matching (Wijaya and Gianfortoni, 2011). Lastly, we are interested in the use of text normalization to augment data for training informal text translation (Michel and Neubig, 2018; Jones and Wijaya, 2021) or for training other downstream applications such as framing identification (Card et al., 2015; Liu et al., 2019; Akyürek et al., 2020), which are typically trained on formal news text, on informal social media text.

Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful comments. Derry Tanti Wijaya is supported in part by the U.S. NSF grant 1838193 (BIGDATA: IA: Multiplatform, Multilingual, and Multimodal Tools for Analyzing Public Communication in over 100 Languages), DARPA HR001118S0044 (Learning with Less Labeling program), and the Department of the Air Force FA8750-19-2-3334 (Semi-supervised Learning of Multimodal Representations).

References

Afra Feyza Akyürek, Lei Guo, Randa Elanwar, Prakash Ishwar, Margrit Betke, and Derry Tanti Wijaya. 2020. Multi-label and multilingual news framing analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8614–8624, Online. Association for Computational Linguistics.

Yuni Azzizah. 2015. Socio-economic factors on indonesia education disparity. *International Education Studies*, 8(12):218–229.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013a. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013b. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364, Nagoya, Japan. Asian Federation of Natural Language Processing.

Anab Maulana Barik, Rahmad Mahendra, and Mirna Adriani. 2019. Normalization of Indonesian-English code-mixed twitter data. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 417–424, Hong Kong, China. Association for Computational Linguistics.

Outi Bat-El. 1996. Selecting the best of the worst: the grammar of hebrew blends. *Phonology*, 13(3):283–328.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1764. Citeseer.

Ondřej Bojar and Aleš Tamchyna. 2011. Improving translation model by monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 330–336, Edinburgh, Scotland. Association for Computational Linguistics.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 438–444, Beijing, China. Association for Computational Linguistics.

Jhon Adrián Cerón-Guzmán and Elizabeth León-Guzmán. 2016. Lexical normalization of spanish tweets. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 605–610.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017. Paradigm completion for derivational morphology. *arXiv preprint arXiv:1708.09151*.

Soenjono Dardjowidjojo. 1979. Acronymic patterns in indonesian. *Pacific Linguistics Series C*, 45:143–160.

Aliya Deri and Kevin Knight. 2015. How to make a frenemy: Multitape fsts for portmanteau generation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 206–210.

Daniel Deutsch, John Hewitt, and Dan Roth. 2018. A distributional and orthographic aggregation model for english derivational morphology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1938–1947.

Shehzaad Dhuliawala, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. Slangnet: A wordnet like resource for english slang. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4329–4332.

Jacob Eisenstein. 2013a. What to do about bad language on the internet. In *Proceedings of the 2013 conference of the North American Chapter of the association for computational linguistics: Human language technologies*, pages 359–369.

Jacob Eisenstein. 2013b. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing spontaneous orthography. In *Proceedings of the 2013 Conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 585–595.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. 2017. Charmanteau: Character embedding models for portmanteau creation. *arXiv preprint arXiv:1707.01176*.

Tri Wahyu Guntara, Alham Fikri Aji, and Radityo Eko Prasojo. 2020. Benchmarking multidomain english-indonesian machine translation. In *Proceedings of the 13th Workshop on Building and Using Comparable Corpora*, pages 35–43.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):1–27.

Andrew F Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89.

Ariel Heryanto. 2008. *Popular culture in Indonesia: Fluid identities in post-authoritarian politics*. Routledge.

John Hewitt, Daphne Ippolito, Brendan Callahan, Reno Kriz, Derry Tanti Wijaya, and Chris Callison-Burch. 2018. Learning translations via images with a massively multilingual image dataset. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2566–2576.

Tom G Hoogervorst. 2014. Youth culture and urban pride: The sociolinguistics of east javanese slang. *Wacana*, 15(1):104–131.

Alex Jones and Derry Tanti Wijaya. 2021. Sentiment-based candidate selection for nmt. *arXiv preprint arXiv:2104.04840*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Irfan Ajmal Khan and Jin-Tak Choi. 2016. Lexicon-corpus based korean unknown foreign word extraction and updating using syllable identification. *Procedia Engineering*, 154:192–198.

Nikzad Khani, Isidora Tourni, Mohammad Sadegh Rasooli, Chris Callison-Burch, and Derry Tanti Wijaya. 2021. Cultural and geographical influences on image translatability of words across languages. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 198–209, Online. Association for Computational Linguistics.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred ConvNet features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 148–158, Lisbon, Portugal. Association for Computational Linguistics.

Vivek Kulkarni and William Yang Wang. 2018. Simple models for word formation in slang. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1424–1434.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Tuan Anh Le, David Moeljadi, Yasuhide Miura, and Tomoko Ohkuma. 2016. Sentiment analysis for low resource languages: A study on informal indonesian tweets. In *Proceedings of the 12th Workshop on Asian Language Resources (ALR12)*, pages 123–131.

Mark Liberman, Yoon-Kyoung Joh, John Laury, and Marjorie Pak. 2003. Types of language change. `http://web.archive.org/web/20200610043656/https://www.ling.upenn.edu/courses/Fall_2003/ling001/language_change.html`. [Online; archived 2020-06-10].

Siyi Liu, Lei Guo, Kate Mays, Margrit Betke, and Derry Tanti Wijaya. 2019. Detecting frames in news headlines and its application to analyzing news framing trends surrounding U.S. gun violence. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 504–514, Hong Kong, China. Association for Computational Linguistics.

Ismini Lourentzou, Kabir Manghnani, and ChengXiang Zhai. 2019. Adapting sequence to sequence models for text normalization in social media. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, pages 335–345.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Elisa Mattiello. 2005. The pervasiveness of slang in standard and non-standard english. *Mots Palabras Words*, 6:7–41.

Paul Michel and Graham Neubig. 2018. Mtnt: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553.

David Moeljadi, Aditya Kurniawan, and Debaditya Goswami. 2019. Building cendana: a treebank for informal indonesian.

Vu H Nguyen, Hien T Nguyen, and Vaclav Snasel. 2015. Normalization of vietnamese tweets on twitter. In *Intelligent Data Analysis and Applications*, pages 179–189. Springer.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 703–711, Jeju Island, Korea. Association for Computational Linguistics.

Zhengqi Pei, Zhewei Sun, and Yang Xu. 2019. Slang detection and identification. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 881–889.

Carlos-Eduardo Piñeros. 2004. The creation of portmanteaus in the extragrammatical morphology of spanish. *Probus*, 16(2):203–240.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525.

Nikmatun Aliyah Salsabila, Yosef Ardhito Winatmoko, Ali Akbar Septiandri, and Ade Jamal. 2018. Colloquial indonesian lexicon. In *2018 International Conference on Asian Language Processing (IALP)*, pages 226–229. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Gary F Simons and Charles D Fennig. 2017. *Ethnologue: languages of Asia*. SIL International Dallas.

Pavol Štekauer, Salvador Valera, and Lívia Kőrtvélyessy. 2012. *Word-formation in the world's languages: A typological survey*. Cambridge University Press.

Suhardianto Suhardianto et al. 2019. Colloquail, slang and transformational language: Comperative study. *JURNAL BASIS*, 6(1):105–118.

Fadillah Tala. 2003. A study of stemming effects on information retrieval in bahasa indonesia.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020a. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Samson Tan, Shafiq Joty, Lav Varshney, and Min-Yen Kan. 2020b. Mind your inflections! Improving NLP for non-standard Englishes with Base-Inflection Encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5647–5663, Online. Association for Computational Linguistics.

Carola Trips. 2017. Morphological change. In *Oxford Research Encyclopedia of Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363.

Daniel Weber and Desislava Zhekova. 2016. Tweetnorm: text normalization on italian twitter data. In *Proceedings of the 13th Conference on Natural Language Processing*, pages 306–312.

Haryo Akbarianto Wibowo, Tatag Aziz Prawiro, Muhammad Ihsan, Alham Fikri Aji, Radityo Eko Prasojo, Rahmad Mahendra, and Suci Fitriany. 2020. Semi-supervised low-resource style transfer of indonesian informal to formal language with iterative forward-translation. In *2020 International Conference on Asian Language Processing (IALP)*, pages 310–315. IEEE.

Derry Tanti Wijaya, Brendan Callahan, John Hewitt, Jie Gao, Xiao Ling, Marianna Apidianaki, and Chris Callison-Burch. 2017. Learning translations via matrix completion. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1452–1463.

Derry Tanti Wijaya and Philip Gianfortoni. 2011. " nut case: what does it mean?" understanding semantic relationship between nouns in noun compounds through paraphrasing and ranking the paraphrases. In *Proceedings of the 1st international workshop on Search and mining entity-relationship data*, pages 9–14.

Liang Wu, Fred Morstatter, and Huan Liu. 2018. Slangsd: building, expanding and using a sentiment dictionary of slang words for short-text sentiment classification. *Language Resources and Evaluation*, 52(3):839–852.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2020. Applying the transformer to character-level transduction. *arXiv e-prints*, pages arXiv–2005.

A Rule-based transformation

Our rule-based transformation can be described in Table 6.

Condition	Transformation	Example
Type: Affixation		
prefix meng- followed by a consonant	replace prefix to nge-	menghina - ngehina
prefix menc-	replace prefix to ny-	mencari - nyari
prefix mem- followed by a consonant	replace prefix to nge-	membuka - ngebuka
prefix men- followed by a consonant	replace prefix to nge-	menjitak ->ngejitak
prefix me- followed by l, q, r, w	replace prefix to nge-	melempar ->ngelempar
suffix -i	replace suffix to -in	pukuli ->pukulin
suffix -kan	replace suffix to -in	hidangkan ->hidangin
Type: Shorten		
prefix me- followed by ng	remove me-	mengegas ->ngegas
prefix me- followed by ny	remove me-	menyanyi ->nyanyi
prefix me- followed by m + vowel	remove me-	memukul ->mukul
prefix me- followed by n + vowel	remove me-	menendang ->nendang
prefix h-	remove h-	habis ->abis
identic duplicate words	replace word with 2	makan-makan ->makan2
Type: Sound-alteration		
last a	replace to e	malam - malem
last i	replace to e	kemarin ->kemaren
last ai	replace to e	sampai ->sampe
last au	replace to o	kalau-kalo
last ai	replace to ae	main - maen
last -nya	replace to -x	sepertinya - sepertix
last p	replace to b	mantap - mantab
last s	replace to z	habis - habiz
Compounding		
any pattern	select the first character	anak baru gede - abg
Second occurrence of cons. + vowel	All character before the pattern	butuh cinta - bucin
Second occurrence of cons. + vowel	All character up to the cons.	nasi goreng - nasgor
Disemvowelling		
any pattern	randomly remove vowels	kemarin - kmarin, kamu - km
Reverse		
any pattern	reverse the word	yuk - kuy

Table 6: List of rule-based transformation.

Manifold Adversarial Augmentation for Neural Machine Translation

Guandan Chen, Kai Fan, Kaibo Zhang, Boxing Chen, Zhongqiang Huang
Alibaba DAMO Academy
{guandan.cgd, k.fan, kaibo.zkb}@alibaba-inc.com
{boxing.cbx, z.huang}@alibaba-inc.com

Abstract

Improving the robustness of neural machine translation models on variations of input sentences is an active area of research. In this paper, we propose a simple data augmentation approach by sampling virtual sentences from the vicinity distributions in higher-level representations, constructed either from individual training samples via adversarial learning or pairs of training samples through mixup. By simplifying and extending previous work that operates at the token level, our method can construct virtual training samples in a broader space and achieve improved translation accuracy compared to the previous state-of-the-art. In addition, we present a simple variation of the mixup strategy to better utilize the pseudo training samples created from back-translation, obtaining further improvement in performance.

1 Introduction

In recent years, neural machine translation (NMT) models (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017) have dramatically improved the quality of machine translation, especially with the introduction of the seminal Transformer architecture (Vaswani et al., 2017) that has become the de facto modeling choice. NMT training aims to learn a parameterized function that models the prediction of the translation in a target language given a source language sentence from labeled training data, which is often limited in volume especially for low-resource domains or languages. Similar to other fields in deep learning, model robustness is an area of concern for NMT as a minor change in the input sentence may result in a different or incorrect translation. In practice this can happen with spelling or grammar errors (Provilkov et al., 2019), speech recognition errors (Ruiz et al., 2019; Di Gangi et al., 2019), or even a sentence of the same meaning but with a slightly different use of words or expressions. Some studies (Belinkov and Bisk, 2017) have shown that the performance of NMT models can drop significantly when small perturbations are added to input sentences.

This problem can be attributed to overfitting as it is difficult to reliably model the translation distribution for the part of input space that has little or no training samples. There have been several attempts to address this problem by filling the space via data augmentation. One direction is to create new training samples by adding perturbations at the token level (Wang et al., 2018; Belinkov and Bisk, 2017; Sperber et al., 2017; Ebrahimi et al., 2018; Li et al., 2019; Cheng et al., 2018, 2019, 2020; Levy et al., 2019), through either token insertion, deletion, and substitution operations or introducing noises to token embedding vectors. Among these approaches, Cheng et al. (2019) demonstrated the effectiveness of incorporating adversarial training samples that are natural sentences, with their semantic relevance to the original sentence safeguarded by language modeling. Cheng et al. (2020) achieved further improvement by creating more diverse but virtual sentences by mixing up actual training samples or synthesized adversarial samples via interpolation of word embeddings, but again at the token level.

Inspired by the success of manifold mixup in computer vision (Verma et al., 2019) and the recent evidence of separable manifolds in deep language representations (Mamou et al., 2020), we propose to simplify and extend previous work on adversarial learning and mixup augmentation to operate in high-level hidden representations, and as such we name the method *manifold adversarial augmentation*. Specifically, we create adversarial representations on a randomly selected hidden layer to attack the NMT model by adding perturbations based on gradients at a random scale to

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3184–3189
August 1–6, 2021. ©2021 Association for Computational Linguistics

some randomly selected positions. Because the adversarial representations diverge slightly from the original representation but in many different ways, they can be viewed as many diverse sentences that are different in expression but have similar meanings. We also create virtual samples by mixing up the hidden representations of two randomly selected samples at a randomly selected hidden layer. Similarly, the mixup presentations can be viewed as many diverse sentences that fill the semantic space between the two original samples, which can help obtain smoother decision boundaries in the data space that is less populated. We further extend the mixup strategy to back-translation, another effective data augmentation method for machine translation, creating virtual samples to bridge the gap between pseudo samples and gold samples.

Experiments on the LDC Chinese-English and IWSLT English-French benchmark tasks demonstrate that our method can significantly improve the vanilla Transformer model by more than 4 and 3 BLEU respectively, averaged over multiple data sets for each task. Compared to the recent state-of-the-art AdvAug method in (Cheng et al., 2020), our method achieves an average improvement of 0.39 and 1.10 BLEU respectively. Further improvement can be achieved with the use of back-translation data.

2 Method

As our manifold adversarial augmentation method is closely related to the AdvAug method (Cheng et al., 2020), we start by highlighting, and also depicting in Figure 1, their similarities and differences.

AdvAug uses both adversarial learning and mixup augmentation at the token level. The adversarial samples are obtained by randomly replacing a small subset of input words (on either source or target side) with words adjacent in the direction of the gradient that can also fit in context based on language modeling. The generated adversaries tend to be natural sentences, however, the variation is limited as it cannot deal with word insertion, deletion, reordering, and more general variations in language expression. Their mixup operation creates virtual samples by interpolating the word embedding vectors of two randomly selected training samples or adversarial samples. While it can generate more training samples, it is hard to interpret the virtual samples as represention of natural sentences, limiting its potential in dealing with natural texts.

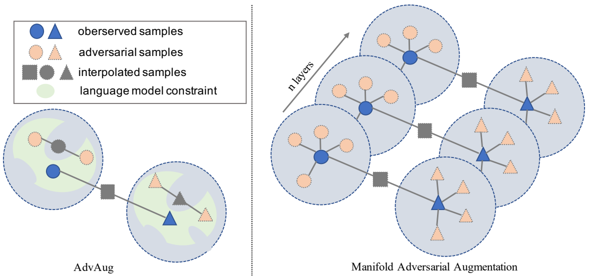

Figure 1: Comparison of the AdvAug method and our manifold adversarial augmentation method.

In contrast, our method operates on higher-level hidden represention for both adversarial learning and mixup augmentation, relying on multiple neural layers to extract semantic meanings, which makes it easier to perform arithmetic operations on semantics. Although we do not explicitly construct adversarial samples that are natural texts, we conjecture that our method has the potential of covering more variations that can occur naturally. We next describe the details of our approach.

2.1 Adversarial Learning

Let $\boldsymbol{x}$ be the input sequence to our model, which could be either a source language sentence or a target language text representing the translation history. We use $\boldsymbol{h}^{(j)}$ and $\boldsymbol{z}^{(\boldsymbol{k})}$ to denote the hidden representations at the j-th encoder layer and the history portion of the k-th decoder layer, respectively. $\texttt{Enc}_{>j}$ denotes the function composed of the encoder layers higher than j, and $\texttt{Dec}_{>k}$ the function composed of the decoder layers higher than k plus the output layer, which computes the generation distribution of output words[1]. We generate perturbation $\delta_{\boldsymbol{h}^{(j)}}$ to the encoder representations $\boldsymbol{h}^{(j)}$ as follows:

$$\delta_{\boldsymbol{h}^{(j)}} = \gamma * \boldsymbol{\eta} * \boldsymbol{g}^{(j)}$$

where $\boldsymbol{g}^{(j)}$ is the gradient with respect to the NMT training loss L_{nmt} back-propagated at $\boldsymbol{h}^{(j)}$, γ is a hyper-parameter controlling the maximum amount of perturbation, and $\boldsymbol{\eta} = [\eta_i \sim Beta(\alpha_{adv}, \beta_{adv}); 0 < i \leq |\boldsymbol{x}|]$ is a random variable providing more fine-gained control of the perturbation. By setting $\alpha_{adv} < 1$ and $\beta_{adv} < 1$, η_i

---

[1] Instead of performing manifold adversarial augmentation on a predetermined hidden layer, we choose to randomly select $j \in [0, K_{src}]$ and $k \in [0, K_{tgt}]$ among a range of encoder and decoder layers, allowing more variations. Appendix A.2 examines the effect of different K_{src} and K_{tgt} values on model performance.

can concentrate close to 0 or 1 and act like a gate independently controlling whether to add perturbation at a specific position, mimicking the random selection of positions for word replacement in AdvAug. Similarly, we generate perturbation $\delta_{\boldsymbol{z}^{(k)}}$ to $\boldsymbol{z}^{(k)}$ on the decoder side. The manifold adversarial learning loss L^m_{adv} is computed by:

$$\begin{aligned}\tilde{\boldsymbol{h}} &= \texttt{Enc}_{>j}(\boldsymbol{h}^{(j)} + \delta_{\boldsymbol{h}^{(j)}}) \\ L^m_{adv} &= \mathbb{E}[\texttt{KL}(\texttt{Dec}_{>k}(\tilde{\boldsymbol{h}}, \boldsymbol{z}^{(k)} + \delta_{\boldsymbol{z}^{(k)}}), \boldsymbol{\omega})]\end{aligned}$$

Here $\boldsymbol{\omega}$ represents the prediction distribution of NMT model on the original training sample, and we base the adversarial loss on KL-divergence instead of MLE, following the VAT work in (Miyato et al., 2018).

2.2 Mixup Augmentation

Verma et al. (2019) investigated manifold mixup augmentation as a way to leverage semantic interpolations at hidden representations as additional training signals for the image classification task. They demonstrated that it results in neural models with smoother decision boundaries at multiple layers, avoiding being overly confident in the space with little or no training samples, and can improve model performance and robustness. Inspired by this work, we attempt to extend the mixup augmentation method in AdvAug (Cheng et al., 2020) from word embeddings to hidden representations at higher layers for NMT training. Specifically, given two training samples, we first compute their hidden representations $\boldsymbol{h}^{(j)}$ and $\boldsymbol{h}'^{(j)}$ at the j-th encoder layer, hidden representations $\boldsymbol{z}^{(k)}$ and $\boldsymbol{z}'^{(k)}$ at the history portion of the k-th decoder layer, and their output distributions $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$. We then construct the hidden representations and the output distribution of the virtual mixup sample as follows:

$$\begin{aligned}\tilde{\boldsymbol{h}}^{(j)} &= m_\lambda(\boldsymbol{h}^{(j)}, \boldsymbol{h}'^{(j)}) \\ \tilde{\boldsymbol{z}}^{(k)} &= m_\lambda(\boldsymbol{z}^{(k)}, \boldsymbol{z}'^{(k)}) \\ \tilde{\boldsymbol{\omega}} &= m_\lambda(\boldsymbol{\omega}, \boldsymbol{\omega}')\end{aligned}$$

where $m_\lambda(\boldsymbol{x}, \boldsymbol{y}) = \lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}$ denotes the interpolation of two vectors, with an interpolation weight $\lambda \sim Beta(\alpha_{mixup}, \beta_{mixup})$ randomly sampled from a Beta distribution for each pair of training samples. The manifold mixup augmentation loss L^m_{mixup} is computed by:

$$L^m_{mixup} = \mathbb{E}[\texttt{KL}(\texttt{Dec}_{>k}(\texttt{Enc}_{>j}(\tilde{\boldsymbol{h}}^{(j)}), \tilde{\boldsymbol{z}}^{(k)}), \tilde{\boldsymbol{\omega}})]$$

Finally, our manifold adversarial augmentation method optimizes on the combination of original NMT training loss, adversarial learning loss, and the mixup augmentation loss:

$$L = L_{nmt} + L^m_{adv} + L^m_{mixup}$$

2.3 Extention to Back-Translation

Back translation is an effective data augmentation method for machine translation. However, it is well known that pseudo training samples created from back translation have different characteristics from the gold training samples, due to factors such as domain mismatch and translation errors. To bridge this gap, we extend the manifold mixup augmentation strategy to create virtual training samples that are interpolated between a pseudo training sample and a gold training sample, again at hidden representations. We can adjust the parameters of the distribution $Beta(\alpha_{bt}, \beta_{bt})$ for generating the interpolation weight, biasing it toward the gold training sample to alleviate the aforementioned problems with back translation. Let $\tilde{\boldsymbol{h}}^{(j)}_{\boldsymbol{bt}}$, $\tilde{\boldsymbol{z}}^{(k)}_{\boldsymbol{bt}}$, and $\tilde{\boldsymbol{\omega}}_{\boldsymbol{bt}}$ be interpolation results, we define an additional training loss:

$$L^{m,bt}_{mixup} = \mathbb{E}[\texttt{KL}(\texttt{Dec}_{>k}(\texttt{Enc}_{>j}(\tilde{\boldsymbol{h}}^{(j)}_{\boldsymbol{bt}}), \tilde{\boldsymbol{z}}^{(k)}_{\boldsymbol{bt}}), \tilde{\boldsymbol{\omega}}_{\boldsymbol{bt}})]$$

3 Experiments

3.1 Setup

We conduct experiments on two language pairs: Chinese-English and English-French. For the Chinese-English translation task, we use the LDC corpus with 1.2M sentence pairs for training, NIST06 for validation, and NIST02, NIST03, NIST04, NIST05, NIST08 as the test sets. For the English-French translation task, we use the IWSLT 2016 corpus with 230k sentence pairs for training, test2012 for validation, and test2013 and test2014 as the test sets. All models are based on the Transformer architecture. Details of the data processing, model configuration, and training settings can be found in the appendix.

We compare with the following methods:

- The vanilla Transformer model (Vaswani et al., 2017).
- The virtual adversarial regularization method in (Sano et al., 2019), which adds a proportion of normalized gradient to the source and target word embeddings for adversarial training.

METHOD	Chinese-English						English-French	
	MT06	MT02	MT03	MT04	MT05	MT08	test13	test14
Vaswani et al. (2017)	44.57	45.49	44.55	46.20	44.96	35.11	40.88	37.79
Sano et al. (2019)	45.75	46.37	45.02	46.49	45.88	35.90	41.67	38.72
Cheng et al. (2019)	46.95	47.06	46.48	47.39	46.58	37.38	41.76	39.46
Cheng et al. (2020)	49.26	49.03	47.96	48.86	**49.88**	**39.63**	43.03	40.91
Our method	**49.43**	**49.54**	**50.34**	**49.46**	49.04	39.19	**44.58**	**41.56**

Table 1: Comparison of main results with different robust training methods.

METHOD	Chinese-English						English-French	
	MT06	MT02	MT03	MT04	MT05	MT08	test13	test14
$L_{nmt} + L^{w}_{mixup}$	48.14	48.75	48.80	48.45	47.69	38.55	43.72	40.37
$L_{nmt} + L^{m}_{mixup}$	48.45	49.55	49.69	49.47	48.95	39.40	44.24	40.46
$L_{nmt} + L^{w}_{adv}$	47.65	48.34	48.40	48.48	47.88	38.59	44.17	40.05
$L_{nmt} + L^{m}_{adv}$	47.90	49.05	48.57	48.88	48.39	38.68	44.35	40.34
$L_{nmt} + L^{w}_{mixup} + L^{w}_{adv}$	48.18	49.37	49.59	48.90	49.03	39.01	44.52	40.87
$L_{nmt} + L^{m}_{mixup} + L^{m}_{adv}$	49.43	49.54	50.34	49.46	49.04	39.19	44.58	41.56

Table 2: Ablation study result of different loss functions. L^{w}_{mixup} and L^{w}_{adv} corresponds to adversarial agumentation at the word embedding level, compared with augmentation at the hidden representation level for L^{m}_{mixup} and L^{m}_{adv}.

- The doubly adversarial inputs method in (Cheng et al., 2019), which performs adversarial learning with word substitutions in the source and target text based on language modeling and gradients at word embeddings.

- The AdvAug method in (Cheng et al., 2020), a state of the art adversarial learning method for NMT, also described in Section 2.

3.2 Main Results

Table 1 shows that our method is very competitive in comparison with other methods in the literature, achieving the overall best results. Compared to the vanilla Transformer, our method achieves more than 4 BLEU points of improvement on average on the Chinese-English task and more than 3 BLEU points of improvement on the English-French task. Compared to AdvAug, the previous state of the art, our method outperforms on 4 out of 6 test sets on the Chinese-English task, yielding up to 2.38 BLEU points of improvement on MT03 and an average improvement of 0.39 BLEU points over the 6 test sets. On the English-French task, our approach yields 1.55 and 0.65 BLEU points of improvement on the two test sets respectively.

METHOD	test13	test14
Vaswani et al. (2017)	40.88	37.79
+back-translation	43.55	40.20
Our method	44.58	41.56
+back-translation	44.81	41.92
+back-translation, $L^{m,bt}_{mixup}$	45.46	42.13

Table 3: Back-translation results on the English-French task.

3.3 Ablation Study

Table 2 presents the ablation study results of different loss functions. In addition to two manifold adversarial augmentation loss functions described in Section 2, we also include their counterparts computed at the word embeddings for comparison. First, we always achieve better MT results with loss functions computed at the hidden represention than at the word embeddings, further validating our motivation that operating at higher hidden layer is superior. Second, we observe that adversarial learning and mixup augmentation are complementary to each other, with the combination of the two achieving the best performance.

3.4 Results with Back Translation

We conduct back-translation experiments on the English-French task as it has a smaller training set and can potentially benefit more from back translation. 25M French sentences from newscrawl07-11[2] are used as additional monolingual data, and are translated to English using a Transformer model trained with only the parallel training data. As shown in Table 3, both the Transformer baseline and our method can benefit from back-translation, although our method obtains a smaller improvement compared to the Transformer baseline as it has a significantly higher BLEU score to start with (actually higher than the Transformer baseline with back-translation). With the addition of our specially designed mixup loss $L_{mixup}^{m,bt}$ that biases toward the gold training samples in mixup augmentation, our method is able to achieve an extra gain of 0.65 and 0.21 BLEU improvement on the two test sets.

4 Conclusion

In this paper, we present a simple yet effective manifold adversarial augmentation method for NMT. By training on virtual samples constructed through adversarial learning and mixup augmentation at higher-level hidden representations, our method can train more robust NMT models with improved translation performance.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443*.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. Advaug: Robust adversarial augmentation for neural machine translation. *arXiv preprint arXiv:2006.11834*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. *arXiv preprint arXiv:1805.06130*.

Mattia Antonino Di Gangi, Robert Enyedi, Alessandra Brusadin, and Marcello Federico. 2019. Robust neural machine translation for clean and noisy speech transcripts. *arXiv preprint arXiv:1910.10238*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. *arXiv preprint arXiv:1806.09030*.

Omer Levy, Jacob Eisenstein, Marjan Ghazvininejad, et al. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47.

Xian Li, Paul Michel, Antonios Anastasopoulos, Yonatan Belinkov, Nadir Durrani, Orhan Firat, Philipp Koehn, Graham Neubig, Juan Pino, and Hassan Sajjad. 2019. Findings of the first shared task on machine translation robustness. *arXiv preprint arXiv:1906.11943*.

Jonathan Mamou, Hang Le, Miguel Del Rio, Cory Stephenson, Hanlin Tang, Yoon Kim, and SueYeon Chung. 2020. Emergence of separable manifolds in deep language representations.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2019. Bpe-dropout: Simple and effective subword regularization. *arXiv preprint arXiv:1910.13267*.

Nicholas Ruiz, Mattia Antonino Di Gangi, Nicola Bertoldi, and Marcello Federico. 2019. Assessing the tolerance of neural machine translation systems against speech recognition errors. *arXiv preprint arXiv:1904.10997*.

Motoki Sano, Jun Suzuki, and Shun Kiyono. 2019. Effective adversarial regularization for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 204–210.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation (IWSLT)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and

---

[2] http://www.statmt.org/wmt12/translation-task.html

Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. *arXiv preprint arXiv:1808.07512*.

A Appendix

A.1 Experiment details

For the IWSLT English-French translation task, the training sets are preprocessed with BPE with 20k joint operations, and English and French share a vocabulary of 20k sub-words. For the NIST Chinese-English translation task, the training sets are preprocessed with BPE with 60k joint operations, and the vocabulary size is 60k and 30k for Chinese and English respectively.

We follow the network settings in the original Transformer work. The total numbers of the parameters of the model are 64757760 and 83247104 for French-English and Chinese-English translation tasks. The dropout ratio is 0.3. The model is optimized with Adam. We use inverse square root as the learning rate schedule, with the peak learning rate of 5e-4, warm-up steps of 4000. During decoding, the beam size is 4 and the length penalty is 0.6. We search hyper-parameters for producing adversarial examples according to BLEU [3] on the validation set. Finally, the maximum number of layers for manifold data augmentation at source side K_{src} and target side K_{tgt} are both set to 3. We let $\alpha_{adv} = 0.5$ and $\beta_{adv} = 0.5$ on the source side, and $\alpha_{adv} = 0.3$ and $\beta_{adv} = 0.7$ on the target side. When mixing training example pairs, the hyperparameter α_{mixup} and β_{mixup} are both set to 8 for the English-French translation task, and set to 0.2 for the Chinese-English translation task. When we mix parallel sentence with back-translated parallel sentence, we let $\alpha_{bt} = 8$ and $\beta_{bt} = 4$.

We use 1 V100 GPU for the IWSLT English-French translation task, and 4 V100 GPU for the NIST Chinese-English translation task. It takes about 24 hours and 72 hours for these two tasks respectively.

K_{src}	MT06	K_{tgt}	MT06
1	48.55	1	48.40
2	48.72	2	48.78
3	49.43	3	49.43
4	49.15	4	48.78
5	48.76	5	48.65
6	48.60	6	48.32

Table 4: Effect of K_{src} and K_{tgt} for manifold adversarial augmentation on NIST Chinese-English translation task

μ_1	μ_2	test13	test14
0	0	40.88	37.79
0	0.3	42.60	40.22
0	0.7	43.17	40.41
0	1	44.24	40.46
0.3	0	43.58	40.09
0.7	0	43.46	39.94
1	0	44.35	40.34
1	1	44.58	41.56

Table 5: Effect of different weights for losses on the IWSLT16 English-French translation task

A.2 Effect of K_{src} and K_{tgt} for manifold adversarial augmentation

Instead of performing manifold adversarial augmentation on a predetermined hidden layer, we choose to randomly select $j \in [0, K_{src}]$ and $k \in [0, K_{tgt}]$ among a range of encoder and decoder layers, allowing more variations. We study their effect on the validation set of the NIST Chinese-English translation task. We fix $K_{src} = 3$ (or $K_{tgt} = 3$), when change the value of K_{tgt} (or K_{src}). As shown in Table 4, large or small K_{src} and K_{tgt} will make the model performs worse down to about 1 BLEU.

A.3 Impact of different weights for losses

To further study the impact of different losses, we set the training loss of our model as $L = L_{nmt} + \mu_1 L^{(m)}_{adv} + \mu_2 L^{(m)}_{mixup}$, and compare the performance when we set different μ_1 and μ_2. We conduct experiments on the IWSLT English-French translation task. As shown in Table 5, too small μ_1 and μ_2 will make the model perform worse.

---

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v13a.pl

Learning to Bridge Metric Spaces: Few-shot Joint Learning of Intent Detection and Slot Filling

Yutai Hou*, Yongkui Lai*, Cheng Chen, Wanxiang Che†, Ting Liu
Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology
{ythou, yklai, cchen, car, tliu}@ir.hit.edu.cn

Abstract

In this paper, we investigate few-shot joint learning for dialogue language understanding. Most existing few-shot models learn a single task each time with only a few examples. However, dialogue language understanding contains two closely related tasks, i.e., intent detection and slot filling, and often benefits from jointly learning the two tasks. This calls for new few-shot learning techniques that are able to capture task relations from only a few examples and jointly learn multiple tasks. To achieve this, we propose a similarity-based few-shot learning scheme, named **Con**trastive **Pro**totype **M**erging network (**ConProm**), that learns to bridge metric spaces of intent and slot on data-rich domains, and then adapt the bridged metric space to specific few-shot domain. Experiments on two public datasets, Snips and FewJoint, show that our model significantly outperforms the strong baselines in one and five shots settings.

1 Introduction

Few-Shot Learning (FSL) that committed to learning new problems with only a few examples (Miller et al., 2000; Vinyals et al., 2016) is promising to break the data-shackles of current deep learning. Commonly, existing FSL methods learn a single few-shot task each time. But, real-world applications, such as dialogue language understanding, usually contain multiple closely related tasks (e.g., intent detection and slot filling) and often benefit from jointly learning these tasks (Worsham and Kalita, 2020; Chen et al., 2019; Qin et al., 2019; Goo et al., 2018). In few-shot scenarios, such requirements of joint learning present new challenges for FSL techniques to capture task relations from only a few examples and jointly learn multiple tasks.

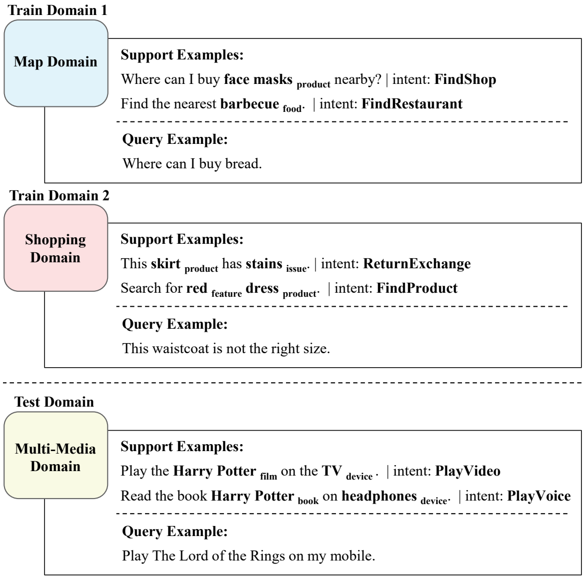

Support Examples:
Where can I buy **face masks** $_{\text{product}}$ nearby? | intent: **FindShop**
Find the nearest **barbecue** $_{\text{food}}$. | intent: **FindRestaurant**

Query Example:
Where can I buy bread.

Support Examples:
This **skirt** $_{\text{product}}$ has **stains** $_{\text{issue}}$. | intent: **ReturnExchange**
Search for **red** $_{\text{feature}}$ **dress** $_{\text{product}}$. | intent: **FindProduct**

Query Example:
This waistcoat is not the right size.

Support Examples:
Play the **Harry Potter** $_{\text{film}}$ on the **TV** $_{\text{device}}$. | intent: **PlayVideo**
Read the book **Harry Potter** $_{\text{book}}$ on **headphones** $_{\text{device}}$. | intent: **PlayVoice**

Query Example:
Play The Lord of the Rings on my mobile.

Figure 1: Examples of the few-shot joint dialogue language understanding. On each domain, given a few labeled support examples, the model predicts the intent and slot labels for unseen query examples. Joint learning benefits from capturing the relation between intent and slot labels, but such relation is hard to learn from a few sparse examples and hard to transfer across different domains.

This paper explores the few-shot joint learning in dialogue language understanding as an early attempt for this issue. As shown in Figure 1, FSL models are usually first trained on source training domains, then evaluated on an unseen target test domain. Although joint learning can improve dialogue language understanding by utilizing the relation between intents and slots, e.g., "Harry Potter" is "film" in "PlayVideo" intent and "book" in "PlayVoice" intent, it faces serious challenges when engaging to FSL setting. Firstly, it is hard to learn generalized intent-slot relations from only a few support examples. Secondly, because the intent-slot relation differs in different domains, it is

---

*Equal contributions.
†Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3190–3200
August 1–6, 2021.

hard to directly transfer the prior experience from source domains to target domains. For instance, the intent-slot relation, “PlayVideo”-“film”, has never appeared in source domains.

To tackle the aforementioned joint learning challenges in few-shot dialogue language understanding, we propose the **Prototype Merging**, which learns the intent-slot relation from data-rich training domains and adaptively captures and utilizes it to an unseen test domain. The intent-slot relation is learned with cross-attention between intent and slot class prototypes, which are the mean embeddings of the support examples belonging to the same classes. Such intent-slot relation adaptively connects the metric spaces of the two tasks.

Further, to jointly refine the intent and slot metric spaces bridged by Prototype Merging, we claim that related intents and slots, such as “PlayVideo” and “film”, should be closely distributed in the metric space, otherwise, well-separated. To achieve this, we propose **Contrastive Alignment Learning**, which exploits class prototype pairs of related intents and slots as positive samples and non-related pairs as negative samples. With these samples, it regularizes the FSL process with a margined contrastive loss.

Overall, we named the above novel few-shot joint learning framework as **Con**trastive **Pro**totype **M**erging network (**ConProm**), which connects intent detection and slot filling tasks by bridging the metric spaces of them. Two main components of it cooperate to accomplish this goal. As shown in Figure 2, Prototype Merging builds the connection between two metric spaces, and Contrastive Alignment Learning refine the bridged metric space by properly distributing prototypes.

Experiments on two public datasets show both Prototype Merging and Contrastive Aligning Objective significantly boost the few-shot joint learning effects and outperform strong baselines. In summary, our contribution is three-fold: (1) We investigate the few-shot joint dialogue language understanding problem, which is also an early attempt for few-shot joint learning problem. (2) We propose a novel Prototype Merging mechanism to build intent-slot connections adaptively. (3) We introduce a Contrastive Alignment Learning objective to jointly refines the metric spaces of intent detection and slot filling. For reproducibility, our code for this paper is publicly available at `https://github.com/AtmaHou/FewShotJoint`.

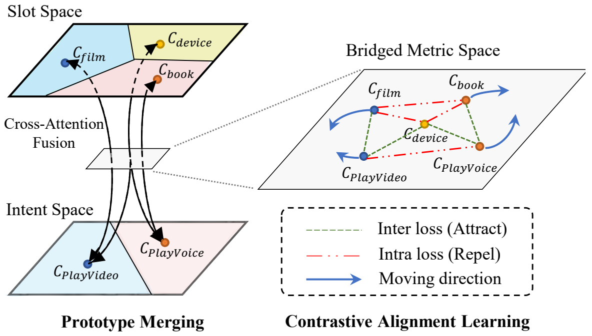

Figure 2: Illustration of two main components of the ConProm model: Prototype Merging and Contrastive Alignment Learning. C denotes prototypes. To ease understanding, we omit the repelling Inter loss in Bridged Metric Space, e.g, loss between C_{book} and $C_{\text{PlayVideo}}$.

2 Background

Before start, we introduce the background of dialogue language understanding and few-shot learning.

2.1 Dialogue Language Understanding

Dialogue language understanding contains two main components: intent detection and slot filling (Young et al., 2013). Intent detection is a sentence-level classification problem that classifies a user utterance into one of N intent categories.

Different from intent detection, slot filling aims to extract key entities within user utterances, which is often achieved by assigning slot tags to each token of a user utterance and is usually formulated as a sequence labeling problem. Given input utterance $\boldsymbol{x} = \langle x_1, x_2, \dots, x_n \rangle$ as a sequence of words, joint dialogue language understanding predicts the corresponding semantic frame $\boldsymbol{y} = (l, \boldsymbol{t})$, where l is the intent label and $\boldsymbol{t} = \langle t_1, t_2, \dots, t_n \rangle$ is the slot tags sequence of the utterance.

2.2 Few-shot Learning

Few-shot learning (FSL) extracts prior experience that allows quick adaption to new problems. Therefore, FSL models are usually first trained on a set of source domains, then evaluated on another set of unseen target domains. Figure 1 shows an example of the training and testing process of few-shot learning for dialogue language understanding.

A target domain only contains a few labeled examples, which is called support set $\mathcal{S} = \left\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\right\}_{i=1}^{|\mathcal{S}|}$. $\mathcal{S}$ includes K examples (K-shot) for each of N classes (N-way). Taking classification problem as an instance: given an input query example $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$ and a K-shot support set $\mathcal{S}$ as references, we find the most appropriate class y^* of $\boldsymbol{x}$:

$$y^* = \arg\max_{y} \; p(y \mid \boldsymbol{x}, \mathcal{S}).$$

State-of-the-art few-shot learning is often similarity-based methods (Bao et al., 2020; Snell et al., 2017). These methods conquer the extreme lack of data by learning a general similarity metric space on data-rich source domains. Then on few-shot target domains, they classify a query example according to example-class similarity, where class representations are obtained from a few support examples.

Prototypical network (Snell et al., 2017) is one of the most classical similarity-based methods. It obtains the class representation as to the mean embedding of support examples belonging to the same class, so called **prototypes**:

$$C_i = \frac{1}{|\mathcal{S}_i|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}_i} E(\boldsymbol{x}),$$

where $\mathcal{S}_i$ is the set of support examples of the ith class, and $E(\cdot)$ is the embedding function. The probability of $\boldsymbol{x}$ belongs to the ith class is then made as:

$$p(y_i \mid \boldsymbol{x}, S) = \frac{\exp\left(\textsc{Sim}(E(\boldsymbol{x}), C_i)\right)}{\sum_j \exp\left(\textsc{Sim}(E(\boldsymbol{x}), C_j)\right)},$$

where $\textsc{Sim}(\cdot,\cdot)$ is a vector similarity function.

3 Proposed Method

In this section, we introduce the proposed **Con**trastive **Pro**totype **M**erging network (ConProm). Firstly, we describe the few-shot intent detection and slot filling with Prototypical network (§3.1). Based on that, we present two key components of ConProm: the **Prototype Merging** mechanism that adaptively connects two metric spaces of intent and slot (§3.2) and the **Contrastive Alignment Learning** that jointly refines the metric space connected by Prototype Merging (§3.3).

3.1 Few-shot Intent Detection and Slot Filling

We build our few-shot intent detection and slot filling model based on the Prototypical Network described in Section 2.2. Given a query sentence $\boldsymbol{x}$ and a support set $\mathcal{S}$, we estimate the probability of $\boldsymbol{x}$ being associated with intent label l_i as:

$$\begin{aligned} &p(l_i \mid \boldsymbol{x}, \mathcal{S}) \\ &= \frac{\exp\left(\textsc{Sim}(E_{\text{intent}}(\boldsymbol{x}), C_{\text{intent}_i})\right)}{\sum_j \exp\left(\textsc{Sim}(E_{\text{intent}}(\boldsymbol{x}), C_{\text{intent}_j})\right)}, \end{aligned}$$

and estimates the probability of the kth token in $\boldsymbol{x}$ belonging to the ith slot class as:

$$\begin{aligned} &p(t_i \mid k, \boldsymbol{x}, \mathcal{S}) \\ &= \frac{\exp\left(\textsc{Sim}(E_{\text{slot}}(x_k), C_{\text{slot}_i})\right)}{\sum_j \exp\left(\textsc{Sim}(E_{\text{slot}}(x_k), C_{\text{slot}_j})\right)}, \end{aligned}$$

where C_{intent_i} and C_{slot_i} are prototypes derived with support examples. $E_{\text{intent}}(\cdot)$ and $E_{\text{slot}}(\cdot)$ are embedder functions for intent and slot respectively. We adopt BERT (Devlin et al., 2019) as the embedder, and the sentence embedding $E_{\text{intent}}(\boldsymbol{x})$ is calculated as the averaged embedding of its tokens. We use the dot-product similarity for function $\textsc{Sim}(\cdot,\cdot)$.

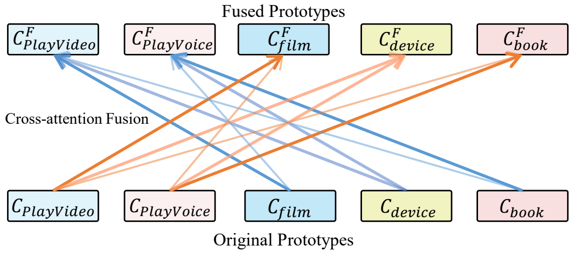

Figure 3: Illustration of cross-attention based information fusion in Prototype Merge. Thicker lines indicate higher cross-attention scores. For example, "PlayVideo" and "film" are more related, so the corresponding score is larger.

3.2 Prototype Merging

To achieve few-shot joint learning and capture the intent-slot relation with the similarity-based method described above, we need to bridge the metric spaces of intent detection and slot filling. However, as mentioned in the introduction, intent-slot relation differs in different domains, it is hard to transfer the bridged metric space learned from source domains to target domains.

To remedy this, we propose the **Prototype Merging** that can bridge metric spaces adaptively. As shown in Figure 3, Prototype Merging adaptively estimates intent-slot relevance with cross-attention between intent and slot, and then merges the intent and slot prototypes with attentive information fusion. Such an attentive fusion process enables both intent and slot prototype representations to reflect intent-slot relation and improves domain transferability.

On an unseen target domain, we estimate the intent-slot cross-attention scores from the support set with two methods: (1) use the statistic of co-occurrence of different intents and slots; (2) estimate the intent-slot relevance score using prototype representations.

Firstly, for the statistic-based attention-score, we estimate intent-slot attention scores A^{S} by counting the co-occurrence of different intents and slots, where $A^{\mathrm{S}}_{i,j}$ records the normalized number of co-occurrence times for the ith intent and the jth slot (normalized by row).

Secondly, for representation-based attention-score, we estimate the cross-attention scores with the Additive Attention (Bahdanau et al., 2015):[1]

$$A^{\mathrm{R}}_{i,j} = V^{\top}\tanh(WC_{\mathrm{intent}_i} + UC_{\mathrm{slot}_j}),$$

where A^{R} is the attention matrix, and $A^{\mathrm{R}}_{i,j}$ records the cross-attention score between the ith intent and the jth slot. U, V and W are parameters learned on source domains,which preserve the general experience of estimating relevance with representations. C_{intent_i} and C_{slot_j} are prototypes of ith intent and the jth slot respectively. We normalize A^{R} by row with softmax function.

We obtain the final cross-attention score matrix A by combining A^{S} and A^{R}.

$$A = \lambda A^{\mathrm{S}} + (1-\lambda)A^{\mathrm{R}},$$

where λ is the interpolation factor.

After obtaining the cross-attention scores, we represent each intent by fusing the information of related slot prototypes, where the attention scores are used as fusing weights. Similarly, we use intent prototypes to represent slots (See Figure 3). The fusion process is as follows:

$$C^{\mathrm{F}}_{\mathrm{intent}_i} = \sum_j A_{ij} \times C_{\mathrm{slot}_j},$$
$$C^{\mathrm{F}}_{\mathrm{slot}_j} = \sum_i A_{ij} \times C_{\mathrm{intent}_i},$$

where $C^{\mathrm{F}}_{\mathrm{intent}_i}$ and $C^{\mathrm{F}}_{\mathrm{slot}_j}$ are the fused prototypes of ith intent and the jth slot respectively.

At last, we obtain the representation of merged prototypes C' by combining the origin prototype C with the fused prototype C^{F}:

$$C'_{\mathrm{intent}} = \alpha \times C^{\mathrm{F}}_{\mathrm{intent}} + (1-\alpha) \times C_{\mathrm{intent}},$$
$$C'_{\mathrm{slot}} = \alpha \times C^{\mathrm{F}}_{\mathrm{slot}} + (1-\alpha) \times C_{\mathrm{slot}},$$

where the α is a hyper-parameter that controls the importance of intent-slot relation.

3.3 Contrastive Alignment Learning

Similarity-based few-shot learning relies heavily on a good metric space, where different classes should be well separated from each other (Hou et al., 2020a; Yoon et al., 2019). In joint-learning scenarios, there are further requests to connect metric spaces of joint learned tasks and jointly optimize these metric spaces.

In response to the above requests, we argue that the distribution of prototypes of dialogue language understanding should fit these intuitions: (1) different intent prototypes should be far away and the same as slot prototypes (*Intra-Contrastive*); (2) the slot prototypes should close to the related intent prototypes and should be far away from the unrelated intent prototypes (*Inter-Contrastive*).[2] To achieve these, we introduce a **Margined Contrastive Loss** to force the model to learn the separation and alignment of intent and slot prototypes.

Firstly, to encourage separation of prototypes from the same task, we regularize the learning of intent and slot prototypes with *Intra-Contrastive* loss $\mathcal{L}_{\mathrm{Intra}} = \frac{1}{2}(\mathcal{L}_{\mathrm{Intra-intent}} + \mathcal{L}_{\mathrm{Intra-slot}})$, where both the $\mathcal{L}_{\mathrm{Intra-intent}}$ and $\mathcal{L}_{\mathrm{Intra-slot}}$ are calculated as:

$$\mathcal{L}_{\mathrm{Intra}} = \frac{1}{N^2}\sum_i\sum_j \max(0, m - \|C_i - C_j\|)^2,$$

where m is the margin value and N is the number of prototypes. The margin m is important since it can protect metric space from excessive dispersion.

Next, we learn the alignment (separation) between intent prototypes and slot prototypes with *Inter-Contrastive* loss $\mathcal{L}_{\mathrm{Inter}}$:

$$\mathcal{L}^{\mathcal{R}}_i = \frac{1}{2|\mathcal{R}_i|}\sum_{j\in\mathcal{R}_i}(\|C_{\mathrm{intent}_i} - C_{\mathrm{slot}_j}\|^2),$$
$$\mathcal{L}^{\mathcal{U}}_i = \frac{1}{2|\mathcal{U}_i|}\sum_{k\in\mathcal{U}_i}\max(0, m - \|C_{\mathrm{intent}_i} - C_{\mathrm{slot}_k}\|)^2,$$
$$\mathcal{L}_{\mathrm{Inter}} = \sum_i^{N_I}(\mathcal{L}^{\mathrm{R}}_i + \mathcal{L}^{\mathrm{U}}_i),$$

---

[1] We adopt additive attention because we find it outperforms common product-based attention in our setting. This is mainly due to that additive attention interferes less with product-based similarity calculations.

[2] A slot is related to an intent means that they used to co-occur in the same semantic frame.

where $\mathcal{R}_i$ is the set of slots related to the ith intent and $\mathcal{U}_i$ is the set of slots that are not related to the ith intent. N_I is the number of intents. Here, we simply obtain the relatedness with the co-occurrence matrix M^{S} in Section 3.2.

Finally, the Margin Contrastive Loss is calculated as:

$$\mathcal{L}_{\text{Contrastive}} = \mathcal{L}_{\text{Inter}} + \mathcal{L}_{\text{Intra}}$$

3.4 Learning Objective

In dialogue language understanding task, we joint learn the intent detection task and slot filling by optimizing both losses at the same time. Specifically, we use CrossEntropy (CE) to calculate the loss for intent detection and slot filling. Combining with the loss of Contrastive Alignment Learning, we train the entire model with the following objective function:

$$\mathcal{L}_{all} = \text{CE}_{\text{intent}} + \text{CE}_{\text{slot}} + \mathcal{L}_{\text{Contrastive}}$$

4 Experiments

We evaluate our method on the dialogue language understanding task of 1-shot/5-shot setting, which transfers knowledge from source domains (training) to an unseen target domain (testing) containing only 1-shot/5-shot support set.

4.1 Settings

Dataset We conduct experiments on two public datasets: Snips (Coucke et al., 2018) and FewJoint (Hou et al., 2020c). Snips is a widely-used dataset for dialogue language understanding, containing seven single-intent domains together with 53 slots. The other dataset FewJoint is joint dialogue language understanding used in the few-shot learning contest of SMP2020-ECDT Task-1.[3] It contains 59 multi-intent domains, 143 different intents, and 205 different slots.

In the few-shot learning setting, we train models on several source domains and test them on unseen target few-shot domains. For Snips, we follow Krone et al. (2020a) and combine single-intent domain into multi-intent domain to achieve the classification of intents. After that, we split the Snips dataset into 3 parts: the training domain with 3 intents, the developing domain with 2 intents and the testing domain with 2 intents. FewJoint is already a few-shot learning benchmark. Therefore, we follow the original data split and there are 45 domains for training, 5 domains for developing and 9 domains for testing.

Few-shot Dataset Construction To simulate the few-shot learning situation, we follow previous few-shot learning works (Vinyals et al., 2016; Krone et al., 2020a; Finn et al., 2017) and construct the dataset into a few-shot episode style, where the model is trained and evaluated with a series of few-shot episodes. Each episode contains a support set and query set. However, different from the single-task problem, joint-learning examples are associated with multiple labels. Therefore, we cannot guarantee that each label appears K times while sampling examples for the K-shot support set. To remedy this, we build support sets with the Mini-Including Algorithm (Hou et al., 2020a), which is intended for such situations. It constructs support set generally following two criteria: (1) All labels appear at least K times in support set. (2) At least one label will appear less than K times in the support set if any support example is removed from the support set. For Snips, we construct 200 few-shot episodes for training, 50 for developing, and 50 for testing. We set the query set size as 16 for training and developing, 100 for testing. For FewJoint, we use the few-shot episodes provided by the original dataset.

Evaluation We adopt three metrics for evaluation: Intent Accuracy, Slot F1-score, Joint Accuracy.[4] For joint dialogue language understanding, Joint Accuracy is the most important metric among all three metrics (Hou et al., 2020c). It evaluates the sentence level accuracy, which considers one sentence is correct only when all its slots and intents are correct.

To conduct a robust evaluation under few-shot setting, we validate the models on multiple few-shot episodes (i.e., support-query set pairs) from different domains and take the average score as final results. To control the non-deterministic neural network training (Reimers and Gurevych, 2017), we report the average score of 5 random seeds for all results.

4.2 Baselines

We compare our model with two kinds of strong baseline: fine-tune based transfer learning methods

---

[3] The Eighth China National Conference on Social Media Processing `https://smp2020.aconf.cn/smp.html`

[4] We calculate the Slot F1-score with the conlleval script `https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt`

Models	**Snips**			**FewJoint**		
	Intent Acc.	**Slot F1**	**Joint Acc.**	**Intent Acc.**	**Slot F1**	**Joint Acc.**
SepProto	**98.23**±0.66	43.90±1.98	9.47±2.10	66.35±0.51	27.24±1.10	10.92±0.89
JointProto	92.57±0.57	42.63±2.03	7.35±1.70	58.52±0.28	29.49±1.01	9.64±0.47
LD-Proto	97.25±0.71	47.81±2.53	10.67±1.99	**67.70**±0.65	27.73±0.35	13.70±0.52
LD-Proto+TR	97.53±0.30	51.03±2.40	17.32±2.62	67.63±1.42	34.06±4.75	16.98±2.14
ConProm (Ours)	96.67±1.45	53.05±0.81	21.72±0.97	65.26±0.23	33.09±1.66	16.32±0.75
ConProm+TR (Ours)	96.17±0.76	**55.84**±0.85	**29.72**±1.30	65.73±0.55	**37.97**±0.70	**19.57**±1.19
JointTransfer	71.07±4.31	38.24±2.19	13.28±0.45	41.83±2.40	26.89±2.72	12.27±2.09
Meta-JOSFIN	71.38±0.76	31.47±0.29	8.88±0.18	57.92±0.66	29.26±0.45	15.00±0.66
LD-Proto+FT	83.85±6.21	45.76±5.24	17.70±2.67	**64.70**±0.50	32.15±1.28	21.32±1.80
ConProm+FT (Ours)	88.20±3.22	52.41±2.01	23.05±1.70	61.24±0.81	42.02±0.77	24.63±1.30
ConProm+FT+TR (Ours)	**90.45**±0.52	**56.04**±1.75	**27.80**±2.33	63.67±0.94	**42.44**±0.51	**27.72**±0.95

Table 1: Scores on 1-shot dialogue language understanding task on Snips and FewJoint datasets. **+FT** denotes finetune model. **+TR** denotes using the trick of transition rule, which blocks illegal slot prediction, such as "I" tag after "O" tag. Results above the mid-line are from non-finetune based methods, and results below the mid-line are from finetuning based methods.

Models	**Snips**			**FewJoint**		
	Intent Acc.	**Slot F1**	**Joint Acc.**	**Intent Acc.**	**Slot F1**	**Joint Acc.**
SepProto	**99.53**±0.11	53.28±1.85	14.40±3.00	75.64±1.51	36.08±0.65	15.93±1.85
JointProto	99.17±0.09	50.63±2.01	13.40±1.44	70.93±2.45	39.47±1.05	14.48±1.11
LD-Proto	99.40±0.08	48.96±1.85	20.93±3.00	**78.29**±1.51	39.88±0.65	22.91±1.85
LD-Proto+TR	99.20±0.30	54.87±3.79	29.40±2.90	75.75±0.95	**51.62**±2.82	27.59±2.31
ConProm (Ours)	98.50±0.42	61.03±1.77	32.20±2.06	78.05±1.04	39.40±1.75	24.18±1.29
ConProm+TR (Ours)	98.99±0.14	**65.13**±1.46	**40.20**±2.24	75.54±1.85	50.28±1.03	**28.69**±1.61
JointTransfer	88.87±5.04	49.62±1.87	25.50±3.09	57.50±6.09	29.00±4.35	18.81±4.45
Meta-JOSFIN	92.47±1.26	56.85±1.25	25.87±0.31	78.91±0.53	53.88±1.63	36.63±1.01
LD-Proto+FT	81.07±8.61	59.27±3.61	26.33±2.38	**80.50**±0.97	55.33±2.55	38.11±2.60
ConProm+FT (Ours)	96.23±1.19	66.66±2.46	39.87±2.60	78.33±1.14	62.34±0.26	40.25±1.19
ConProm+FT+TR (Ours)	**98.40**±0.20	**72.98**±0.41	**52.95**±0.85	78.43±1.86	**69.44**±0.39	**46.54**±0.72

Table 2: Scores on 5-shot dialogue language understanding task on Snips dataset and FewJoint dataset.

(JointTransfer, Meta-JOSFIN) and similarity-based FSL methods (SepProto, JointProto, LD-Proto).

JointTransfer is a domain transfer model based on the JointBERT (Chen et al., 2019). It consists of a shared BERT embedder with intent detection and slot filling layers on the top. We pretrain it on source domains and finetune it on target domain support sets.

Meta-JOSFIN (Bhathiya and Thayasivam, 2020) is a meta-learning model based on the MAML (Finn et al., 2017). The meta-learner model here is a BERT-based joint dialogue language understanding model similar to **JointTransfer**. It learns initial parameters that can fast adapt to the target domain after only a few updates.

SepProto is a prototypical-based dialogue language understanding model with BERT embedding, that learns intent detection and slot filling separately. During the experiment, it is pre-trained on source domains and then directly applies to target domains without fine-tuning.

JointProto (Krone et al., 2020a) is all the same as SepProto except that it jointly learns the intent and slot tasks by sharing the BERT encoder.

LD-Proto is also a prototypical model similar to **JointProto**. The only difference is that it is enhanced by the logits-dependency tricks (Goo et al., 2018), where joint learning is achieved by depending on the intent and slot prediction on the logits of the accompanying task.

Implements For both ours and baseline models, we determine the hyperparameters on the development set. We use ADAM (Kingma and Ba, 2015) for training and set batch size as 4 and learning rate as 10^{-5}. We adopt embedding tricks of Pairs-Wise Embedding (Gao et al., 2019; Hou et al., 2020a) and Gradual Unfreezing (Howard and Ruder, 2018). The λ and α in Section 3.2 are both set as 0.5. We implement both our and baseline models with the few-shot platform MetaDialog.[5] Besides, to use the information in target domains and make a fair comparison with fine-tuning baselines, we explore the performance of the similarity-based model under fine-tuning setting (+FT) and enhance the model with a fine-tune process similar to Meta-JOSFIN. In addition, following the suggestions of Hou et al. (2020a), we investigate adding Transition Rules (+TR) between slot tags, which bans illegal slot prediction, such as "I" tag after "O" tag.

4.3 Main Results

In this section, we present the evaluation of the proposed method on both 1-shot and 5-shot dialogue understanding setting.

Result of 1-shot setting As shown in Table 1, our method (ConProm) achieves the best performance on Joint Accuracy, which is the most important metric. Among all metrics, ConProm only lags a bit than LD-Proto on intent accuracy. We address this to the fact that there are many slots shared by different intent, and representing an intent with slots may unavoidably introduce noise from other intents. Considering the huge improvements on Slot and Joint performance over LD-Proto, we argue that the limited loss is a worthy compromise here. Since similarity-based models predict slot tags independently for each token, they tend to predict illegal tags. We employ a simple transition rule (+TR) to remedy such defects and further improves the performance. For fairness, we also enhance LD-Proto with TR trick and our model still outperforms the enhanced baseline.

For those non-finetuned methods, ConProm outperforms LD-Proto by Joint Accuracy scores of 11.05 on Snips and 2.62 on FewJoint, which show that our model can better capture the relation between intent and slot. Our improvements on Snips are higher than those on FewJoint, which is mainly because that there is clearer intent-slot dependency in Snips. The performance of JointProto is even lower than SepProto, which demonstrates that few-shot joint learning is not a trivial issue as simply sharing the embeddings

Setting	Snips		FewJoint	
	1-shot	5-shot	1-shot	5-shot
Ours	21.72	32.20	16.32	24.18
- PM	-1.90	-2.63	-4.90	-8.39
- CAL	-5.19	-12.73	-1.78	-3.78

Table 3: Ablation study over two main components of proposed framework: Prototype Merge (PM) and Contrastive Alignment Learning (CAL). The score is Joint Accuracy.

When finetuning brings significant improvements for all methods, our model (ConProm+FT) still achieves the best performance. Interestingly, we observe that finetuning often hurts the intent prediction. This shows that finetuning brings limited gains on sentence-level domain knowledge but leads to overfitting.

Result of 5-shot setting Table 2 shows the 5-shot results. The results are consistent with 1-shot setting in general trending and our methods achieve the best performance. While more learning shots improve the performance for all methods, the superiority of our best performed baseline is further strengthened. This shows that the model can better exploit the richer intent-slot relations hidden in 5-shot support sets.

4.4 Analysis

Ablation Test To inspect how each component of the proposed model contributes to the final performance, we conduct ablation analysis. As shown in Table 3, we independently removing two main components: Prototype Merge (PM) and Contrastive Alignment Learning (CAL).

When PM is removed, the intent and slot prototypes are represented only with corresponding support examples, and Joint Accuracy drops are witnessed. There is more loss on FewJoint. Because there are much more slots shared by different intents in FewJoint, and the attention mechanism of PM is important for identifying relatedness between intents and slots.

For our model without CAL, we train the model with only cross entropy loss and get lower scores on all settings. There are more performance drops on Snips. This is mainly because that there much clearer intent-slot relation in Snips, which can be easily handled by CAL.

In terms of contribution, there are opposite performance for CAL and PM on two dataset, which shows that PM and CAL complement each other and reach a balance for various situations.

---

[5] `https://github.com/AtmaHou/MetaDialog`

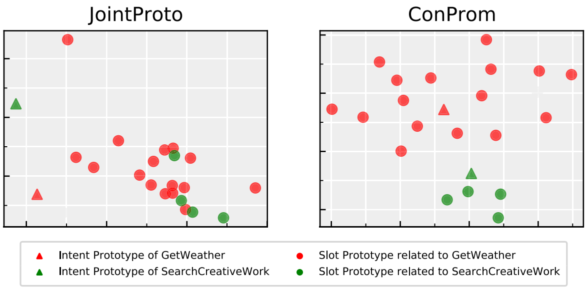

Figure 4: Visualization of the prototype distribution of JointProto and Ours (ConProm) with tSNE (step=500).

Models	**Snips**		**FewJoint**	
	F1.	**Acc.**	**F1.**	**Acc.**
JointProto	42.63	8.08	29.49	15.73
LD-Proto	47.81	10.72	27.73	20.44
LD-Proto+TD	51.03	17.53	34.06	24.69
ConProm	53.05	22.30	33.09	22.38
ConProm+TD	55.84	30.47	37.97	26.31
JointTransfer	38.24	14.38	26.89	26.37
Meta-JOSFIN	31.47	9.73	29.26	21.73
LD-Proto+FT	45.76	21.92	32.15	35.75
ConProm+FT	52.41	25.97	42.02	39.62
ConProm+TD+FT	56.04	27.91	42.44	40.71

Table 4: Analysis for sentence level slot accuracy.

Visual Analysis of Prototype Distribution To get further an understanding of the model effects on bridging the metric spaces of intent and slot, we visualize the prototype distributions in the metric space. As shown in Figure 4, it is exciting to see that our model successfully refine the prototype distribution by aligning the slots to related intent and making prototypes properly well-separated.

Sentence level slot accuracy analysis There is some confusion in Table 1 and Table 2 that there are huge performance differences of Joint Accuracy score when Intent Accuracy scores and Slot F1 scores are similar. We inspect this issue by evaluating the Sentence Level Slot Accuracy, which considers a sentence to be correct when all slots are correct. As shown in Table 4, there is a huge gap in the slot accuracy score between LD-Proto and ConProm, which explains the gap in Joint score.

5 Related Work

Few-shot learning is one of the most important direction for machine learning area (Fei-Fei, 2006; Fink, 2004) and often achieved by similarity-based method (Vinyals et al., 2016) and fine-tuning based method (Finn et al., 2017). FSL in natural language processing has been explored for various tasks, including text classification (Sun et al., 2019; Geng et al., 2019; Yan et al., 2018; Yu et al., 2018), entity relation classification (Lv et al., 2019; Gao et al., 2020; Ye and Ling, 2019), sequence labeling (Luo et al., 2018; Hou et al., 2018; Shah et al., 2019; Hou et al., 2020a; Liu et al., 2020).

As the important part of a dialog system, dialogue language understanding attract a lot of attention in few-shot scenario. Dopierre et al. (2020); Vlasov et al. (2018); Xia et al. (2018) explored few-shot intent detection technique. Luo et al. (2018) and Hou et al. (2020a) investigated few-shot slot tagging by using prototypical network. Hou et al. (2020b) explored few-shot multi-label intent detection with an adaptive logit adapting threshold. But all of these works focus on a single task.

Despite a lot of works on joint dialogue understanding (Goo et al., 2018; Li et al., 2018; Zhang et al., 2019; Qin et al., 2019; Wang et al., 2018; E et al., 2019; Wu et al., 2020; Gangadharaiah and Narayanaswamy, 2019; Liu et al., 2019; Qin et al., 2020), few-shot joint dialogue understanding is less investigated. Krone et al. (2020b) and Bhathiya and Thayasivam (2020) make the earliest attempts by directly adopt general and classic few-shot learning methods such as MAML and prototypical network. These methods achieve joint learning by sharing the embedding between intent detection and slot filling task, which model the relation between intent and slot task implicitly. By contrast, we explicitly model the interaction between intent and slot with attentive information fusion and constrastive loss. Experiment results also demonstrate the superiority of our method on this task.

6 Conclusion

In this paper, we propose a similarity-based few-shot joint learning framework, ConProm, for dialogue understanding. To adaptively model the interaction between intents and slots, we propose the Prototype Merging that bridges the intent metric and slot metric spaces with cross-attention between intent and slot. To learn better bridged metric space for intent and slot, we propose the Contrastive Alignment Learning to align related cross-task labels in metric space and force unrelated labels properly separated. Experiment results validate that both Prototype Merging and Contrastive Alignment Learning can improve performance.

Acknowledgments

We are grateful for the helpful comments and suggestions from the anonymous reviewers. This work was supported by the National Key R&D Program of China via grant 2020AAA0106501 and the National Natural Science Foundation of China (NSFC) via grant 61976072 and 61772153.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of the ICLR*.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *Proc. of the ICLR*. OpenReview.net.

Hemanthage S Bhathiya and Uthayasanker Thayasivam. 2020. Meta learning for few-shot joint intent detection and slot-filling. In *Proc. of the ICMLT*, pages 86–92.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of the NAACL, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Dopierre, Christophe Gravier, Julien Subercaze, and Wilfried Logerais. 2020. Few-shot pseudo-labeling for intent detection. In *Proc. of the COLING*, pages 4993–5003, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proc. of the ACL*, pages 5467–5471, Florence, Italy. Association for Computational Linguistics.

Li Fei-Fei. 2006. Knowledge transfer in learning to recognize visual objects classes. In *Proc. of the ICDL*, pages 1–8.

Michael Fink. 2004. Object classification from a single example utilizing class relevance metrics. In *Proc. of the NIPS*, pages 449–456.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. of the ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2019. Joint multiple intent detection and slot labeling for goal-oriented dialog. In *Proc. of the NAACL*, pages 564–569, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2020. Neural snowball for few-shot relation learning. In *Proc. of the AAAI*, pages 7772–7779. AAAI Press.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proc. of the EMNLP-IJCNLP*, pages 6250–6255, Hong Kong, China. Association for Computational Linguistics.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proc. of the EMNLP-IJCNLP*, pages 3904–3913, Hong Kong, China. Association for Computational Linguistics.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proc. of the NAACL, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020a. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proc. of the ACL*, pages 1381–1393, Online. Association for Computational Linguistics.

Yutai Hou, Yongkui Lai, Yushan Wu, Wanxiang Che, and Ting Liu. 2020b. Few-shot learning for multi-label intent detection. *CoRR*, abs/2010.05256.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. *Proc. of the COLING*.

Yutai Hou, Jiafeng Mao, Yongkui Lai, Cheng Chen, Wanxiang Che, Zhigang Chen, and Ting Liu. 2020c. Fewjoint: A few-shot learning benchmark for joint language understanding. *CoRR*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of the ACL (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of the ICLR*.

Jason Krone, Yi Zhang, and Mona Diab. 2020a. Learning to classify intents and slot labels given a handful of examples. In *Proc. of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 96–108, Online. Association for Computational Linguistics.

Jason Krone, Yi Zhang, and Mona Diab. 2020b. Learning to classify intents and slot labels given a handful of examples. In *Proc. of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 96–108, Online. Association for Computational Linguistics.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *Proc. of the EMNLP*, pages 3824–3833, Brussels, Belgium. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jie Zhou, Yufeng Chen, and Jinan Xu. 2019. CM-net: A novel collaborative memory network for spoken language understanding. In *Proc. of the EMNLP-IJCNLP*, pages 1051–1060, Hong Kong, China. Association for Computational Linguistics.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. In *Proc. of the ACL*, pages 19–25.

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying up regular expressions with neural networks: A case study for spoken language understanding. In *Proc. of the ACL (Volume 1: Long Papers)*, pages 2083–2093, Melbourne, Australia. Association for Computational Linguistics.

Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2019. Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations. In *Proc. of the EMNLP-IJCNLP*, pages 3376–3381, Hong Kong, China. Association for Computational Linguistics.

Erik G. Miller, Nicholas E. Matsakis, and Paul A. Viola. 2000. Learning from one example through shared densities on transforms. In *Proc. of the CVPR*, pages 1464–1471. IEEE Computer Society.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. In *Proc. of the EMNLP-IJCNLP*, pages 2078–2087, Hong Kong, China. Association for Computational Linguistics.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. Towards fine-grained transfer: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. In *Proc. of the EMNLP: Findings*, pages 1807–1816.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proc. of the EMNLP*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Darsh J. Shah, Raghav Gupta, Amir A. Fayazi, and Dilek Hakkani-Tür. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proc. of the ACL*, pages 5484–5490.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Proc. of the NeurIPS*, pages 4077–4087.

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. In *Proc. of the EMNLP-IJCNLP*, pages 476–485, Hong Kong, China. Association for Computational Linguistics.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Proc. of the NeurIPS*, pages 3630–3638.

Vladimir Vlasov, Akela Drissner-Schmid, and Alan Nichol. 2018. Few-shot generalization across dialogue tasks. *CoRR*, abs/1811.11707.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proc. of the NAACL (Short Papers)*, pages 309–314, New Orleans, Louisiana. Association for Computational Linguistics.

Joseph Worsham and Jugal Kalita. 2020. Multi-task learning for natural language processing in the 2020s: where are we going? *Pattern Recognition Letters*.

Di Wu, Liang Ding, Fan Lu, and Jian Xie. 2020. SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling. In *Proc. of the EMNLP*, pages 1932–1937, Online. Association for Computational Linguistics.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proc. of the EMNLP*, pages 3090–3099, Brussels, Belgium. Association for Computational Linguistics.

Leiming Yan, Yuhui Zheng, and Jie Cao. 2018. Few-shot learning for short text classification. *Multimedia Tools and Applications*, pages 1–12.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Multi-level matching and aggregation network for few-shot relation classification. In *Proc. of the ACL*, pages 2872–2881, Florence, Italy. Association for Computational Linguistics.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proc. of the 36th ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7115–7123. PMLR.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. In *Proc. of the IEEE*, volume 101, pages 1160–1179. IEEE.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proc. of the NAACL, Volume 1 (Long Papers)*, pages 1206–1215, New Orleans, Louisiana. Association for Computational Linguistics.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proc. of the ACL*, pages 5259–5267, Florence, Italy. Association for Computational Linguistics.

Insertion-based Tree Decoding

Denis Lukovnikov
Ruhr University Bochum,
Bochum, Germany
denis.lukovnikov@rub.de

Asja Fischer
Ruhr University Bochum,
Bochum, Germany
asja.fischer@rub.de

Abstract

Sequences are typically decoded in a left-to-right fashion, requiring as many decoding steps as there are tokens in the sequence. Recently, several works have proposed non-autoregressive decoders that are sub-linear, allowing to decode a sequence using fewer decoding steps than the length of the sequence, and thus substantially speed up inference. In contrast, non-autoregressive decoding of trees is less well-analysed, even though trees are used in important applications like semantic parsing and code generation. In this work, we present a novel general-purpose partially autoregressive tree decoder that uses tree-based insertion operations to generate trees in sub-linear time. We evaluate our approach on semantic parsing and compare it against strong baselines, including an insertion-based sequence decoder. The results demonstrate that the partially autoregressive tree decoder reaches competitive accuracies while clearly reducing the number of decoding steps.

1 Introduction

Sequence generation is usually based on a left-to-right autoregressive decoder that decomposes the probability of the entire sequence y conditioned on x (x can be empty) as the product $p(y|x) = \prod_{i=0}^{N} p(y_i|y_{<i}, x)$. At each decoding step, the decoder model predicts the next token y_i based on the previously generated outputs $y_{<i}$ and the input x. This approach to decoding sequences is linear in sequence length: the number of decoding steps necessary to produce the sequence is equal to the length of the sequence. However, recently, several works have proposed non-autoregressive decoders that are sub-linear. This allows to decode a sequence using fewer decoding steps than the length of the sequence and can thus greatly speed up inference, especially for longer sequences (Stern et al., 2019; Ma et al., 2019; Ghazvininejad et al., 2019; Gu et al., 2017; Kasai et al., 2020). In particular, the Insertion Transformer of Stern et al. (2019) uses insertion operations to iteratively expand the sequence, achieving a best-case number of $\mathcal{O}(\log_2 N)$ decoding steps.

In this work, we extend insertion-based decoding of sequences to insertion-based decoding of trees. Insertion-based sequence decoder can also be applied to decoding trees (Zhu et al., 2020). However, this requires linearizing trees into sequences, and requires the explicit decoding of subtree termination tokens (e.g. closing parentheses ")"). This results in larger structures, which for the Insertion Transformer increases the minimum necessary number of decoding steps and increases the computational requirements per step. In contrast, the insertion-based tree decoder that we propose here does not need to explicitly decode structure tokens. Moreover, it can achieve a best-case complexity *below* $\mathcal{O}(\log_2 N)$[1] in terms of the number of decoding steps and guarantees that all intermediate outputs are valid trees. To the best of our knowledge, no existing research focuses on insertion-based non-autoregressive decoding of trees so far.

We evaluate the proposed decoder for semantic parsing on the OVERNIGHT (Wang et al., 2015) dataset. Semantic parsing is the task of converting natural language expressions into a formal representation of its meaning. An important application of semantic parsing is question answering from structured data sources. In such use cases, the input to the semantic parser is a natural language question and the expected output is a query, which can be written in a query language (e.g. SQL or SPARQL). Since these queries can often be represented as trees (e.g. abstract syntax tree), semantic parsing is a particularly interesting task for the evaluation of the presented decoding approach.

---

[1] The exact best possible speed-up heavily depends on the data.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3201–3213
August 1–6, 2021.

To summarize, the contributions of this work are:

- a transformer-based decoding algorithm that uses insertion operations specifically tailored for decoding trees,
- a novel transformer architecture that uses novel tree-based relative positions,
- and an evaluation of the proposed algorithm and model on the well-known OVERNIGHT dataset, and a comparison against a strong non-autoregressive baseline.

2 Insertion Transformer

Below, we give a very brief overview of the Insertion Transformer, which we use here as a baseline. Due to space constraints, we refer interested readers to the work of Stern et al. (2019) for a more elaborate description of their model and training procedure.

Decoding approach. Rather than decoding autoregressively left-to-right (LTR), the insertion transformer decodes sequences by using insertion operations. For example, consider the sequence "*A B C D E F G*". LTR decoding would require at least seven decoding steps, producing some left-aligned subsequence at every step (e.g. "A B C D" at the fourth step). In contrast, decoding using insertions allows to decode the same sequence using just three steps. Starting from the initial empty state " ", we first decode (1) "***D***", then (2) "***B D F***" and finally (3) "***A B C D E F G***", where the bold faced tokens are the ones inserted in each step, respectively.

Model. The model we use in our experiments uses BERT (Devlin et al., 2019) as the encoder and a standard transformer (Vaswani et al., 2017) with learned absolute position vectors as the decoder. In contrast to the vanilla transformer decoder however, the causal attention mask is not used. After encoding a subsequence using the transformer, the output layer concatenates the representations of two neighbouring tokens to build a representation for the insertion slot between the tokens. We normalize the probabilities per slot.

Training. Given training data consisting of pairs of input and output sequences (x, y), at every epoch, the training algorithm samples a subsequence $\hat{y}$ for every output sequence y. First, a length for the subsequence is drawn from a uniform distribution on $[0, |y|]$. Then, a subsequence $\hat{y}$ of the given length is randomly drawn from y. For example, for the sequence "A B C D E F G" and a randomly drawn length of 3, a sampled subsequence of length 3 could be "B D E".

The Insertion Transformer is then trained by optimizing a loss of the following form:

$$-\sum_{l}\sum_{i=i_l}^{j_l} w_{i,l} \log p(y_i, l|x, \hat{y}) \ , \qquad (1)$$

where i_l and j_l are the beginning and end positions of the subsequence to be decoded in slot l. Two variants of this loss function are proposed that differ in the strategy of assigning the weights $w_{i,l}$ to the different tokens and which are referred to as *uniform* and *binary*, respectively. In the uniform case, for a given l the weights $w_{i,l}$, $i = i_l, \ldots, j_l$ are equal and sum up to one. In the binary case, a larger weight is assigned to tokens closer to the center of slot l:

$$w_{i,l} = \frac{e^{-d_l(i)/\tau}}{\sum_{i'=i_l}^{j_l} e^{-d_l(i')/\tau}} \ , \qquad (2)$$

where $d_l(i)$ is the distance between token i and the center of the span to be decoded in slot l, and τ is the *temperature*.

3 Tree-based Insertion Transformer

In this section, we propose a novel transformer-based method for non-autoregressive decoding of trees. The proposed method consists of (1) a novel transformer architecture, and (2) a novel insertion-based decoding procedure.

3.1 Decoding

The proposed decoding algorithm is similar to the one proposed by Stern et al. (2019) in that it uses insertion operations to expand the decoded structure. However, instead of using the implicitly defined insertion slots between two neighbouring tokens in a sequence, here, insertion slots are used that are placed between neighbouring nodes in the graph representing the tree.

As an example, consider the tree *(A B C)*[2], shown in Fig. 1a. For sequence-based insertion decoding, we would take the linearization *(· A · B · C ·)*, where four insertion slots are explicitly denoted by "·". The linearized representation clearly has some disadvantages. For example, performing some insertions at some slots can destroy the tree structure,

---

[2] Trees are given in a Lisp-like notation in this work.

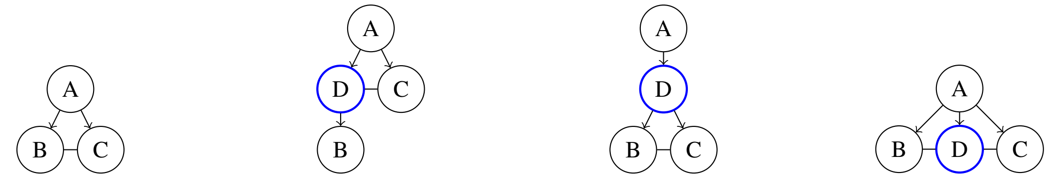

(a) Simple tree

(b) Inserting D at the ancestral slot of B.

(c) Inserting D at the descendant slot of A.

(d) Inserting D at sibling slot between B and C.

Figure 1: Examples of insertions at different insertion slots.

e.g. inserting ")" between B and C yields *(A B) C)*, which is not a valid tree. Also, as mentioned before,

In this work, we propose a tree-based insertion decoding algorithm that defines insertion slots on edges between siblings, as well as on edges between parents and children. Following this approach, the example tree can be described by the following linearization with explicit insertion slots:

$$(\wedge A \vee \mid - \wedge B \vee - \wedge C \vee -) \quad ,$$

where we use three types of insertion slots: (1) ancestor insertion slots denoted by $\wedge$, (2) descendant insertion slots denoted by $\vee$, and (3) sibling insertion slots denoted by –, as described in more detail in the following. Note that the parentheses and the pipe symbol "|" are ignored by the model (when relative positioning is used) and is used mostly for notational and programming convenience. The pipe symbol separates the root portion of the "slotted" subtree string.

Ancestor insertion: If node D is used for the ancestor insertion slot $\wedge$ of B, which corresponds to the red slot in

$$(\wedge A \vee \mid - \wedge B \vee - \wedge C \vee -) \quad ,$$

we obtain the tree *(A (D B) C)* (see Fig. 1b). In other words, D replaces B and the entire subtree B is attached as a child of D.

Descendant insertion: If node D is used with a descendant insertion slot $\vee$ of A, which corresponds to the red slot in

$$(\wedge A \vee \mid - \wedge B \vee - \wedge C \vee -) \quad ,$$

then the child subtrees of A are moved to node D and the entire subtree D (that now contains the children of A) is attached to A to yield the tree *(A (D B C))*, which is depicted in Fig. 1c.

Sibling insertion: If node D is used with a sibling insertion slot "–" between B and C, which corresponds to the red slot in

$$(\wedge A \vee \mid - \wedge B \vee - \wedge C \vee -) \quad ,$$

then D is inserted as a child of A between B and C to yield the tree *(A B D C)* (see Fig. 1d).

These insertion actions can be used to decode any tree starting from a tree containing only a root node *(ROOT)*. Note that every decoding step is guaranteed to yield a valid tree, unlike in the sequence-based insertion decoder.

3.1.1 Decoder operation

In every step, the decoder takes the previous intermediate tree y_{t-1} (where y_0 = *(ROOT)*) and applies one or more insertion operations to expand the tree to the next intermediate tree y_t. To predict which insertion operations to execute at every available slot, the model described in the following sections encodes the *entire* input tree y_{t-1}. The prediction is then based on this encoding. This procedure is similar to the approach used by Stern et al. (2019). While more efficient methods that re-use computations are possible, we leave an investigation of those for future work. The decoding process is terminated when all slots predict a slot closing actions indicating that no more nodes should be inserted, and thus $y_t = y_{t-1}$.

3.2 Model

We propose a novel architecture that takes advantage of the fact that the intermediate structures generated in all steps of the decoding algorithm are trees. While the encoder stays the same (i.e. BERT), the decoder of our model relies on a transformer-based tagger with tree-based relative positions and a special attention mask. The tree-based relative positions allow us to ignore structure tokens.

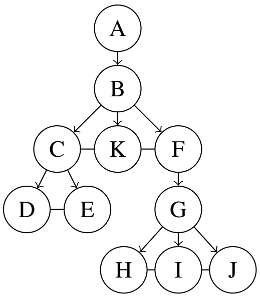

Figure 2: An example of a tree.

3.2.1 Computing Relative Positions in Trees

The proposed architecture uses **tree-based relative positioning** instead of absolute positional embeddings. The relative position is described by the **movements in the tree** that the path from node u to node v defines. For example, in the tree *(A (B (C D E) K (F (G H I J))))* depicted in Fig. 2, the relative position of node D to node I is given by "1↑ 2→ 2↓". This describes the following movement in the tree: starting in D go up one hop to reach parent C (1↑), then move right among the siblings two hops to reach F (2→), and finally move down two hops to reach I (2↓). The scheme is *insensitive to the order of children*, e.g., the relative position from B to K is "1↓", the same as for (B, C) and (B, F). To distinguish these cases, we propose to instead use **special relative position relations** "child_X" and "child_X_of" between parent and child nodes (where X denotes the position of the child among all children of its parent, e.g., X is 1 for node K). While this should improve local modeling of parent-child relations, it still does not add sufficient information to other paths, e.g., the relative positions for (D, H) and (D, J) are identical to that for (D, I)[3].

3.2.2 Using Relative Positions in the Model

With e_{ij} denoting the unnormalized attention scores, x_i the vector corresponding to element i in the input sequence and a^K_{ij} and a^V_{ij} the key (K) and value (V) vector representations of the relative position between the input elements at position i and position j, the following equations describe how the relative position vectors are used in the attention mechanism:

$$e_{ij} = \left(x_i W^Q (x_j W^K)^T + x_i W^Q (a^K_{ij})^T\right) / \sqrt{d_z}$$
$$z_i = \textstyle\sum_{j=1}^{n} \alpha_{ij} (x_j W^V) + \alpha_{ij} a^V_{ij} \ .$$

This approach to incorporate relative positions is similar to that of Shaw et al. (2018) but differs in the computation of a^K_{ij}'s and a^V_{ij}'s. A naive implementation would create an independent position embedding for every possible combination of the elements of a movement pattern. For an efficient implementation that is both faster and has fewer parameters, we separately consider the three parts of a movement pattern, embed them separately and add their embeddings:

$$a^K_{ij} = a^{K,\uparrow}_{ij} E^{K,\uparrow} + a^{K,\leftrightarrow}_{ij} E^{K,\leftrightarrow} + a^{K,\downarrow}_{ij} E^{K,\downarrow} \ , \quad (3)$$

where $a^{K,\downarrow}_{ij}$, $a^{K,\uparrow}_{ij}$, and $a^{K,\leftrightarrow}_{ij}$ are one-hot vectors representing the components of the movement pattern, and $E^{K,\downarrow}$, $E^{K,\uparrow}$, and $E^{K,\leftrightarrow}$ are their corresponding embedding lookup matrices.

In case the relative position is not a movement pattern, as is the case for parent-child relations, we simply look up a single embedding vector. We also use relative positions *from* insertion slot positions to some of the node positions: (1) ancestor (∧) and descendant (∨) slots use the "ancestors" and "descendant" relations, respectively, and (2) the sibling slots (–) use the "left_sibling" and "right_sibling" as well as a "parent" relation. Since insertion slot positions are not attended *to* (they are not used as keys in attention), we do not need relative positions *to* slot positions.

3.2.3 Attention Mask

We use a custom attention mask pattern that prevents real query tokens from attending to structural tokens and insertion slots. Since we do insertion-based decoding, we don't use a causal attention mask. However, since structural information is already described by the relative positions (see above), we do not need to process the structure-describing tokens, such as parentheses. Thus, these tokens are masked both as keys and as queries and do not participate in the decoding process at all. Additionally, we use a special mask for the insertion slots (defined above) such that (1) other tokens can not attend *to* the slot tokens and (2) the slot tokens can *only* attend to their immediate neighbours.

3.3 Decoder Training

The decoder learns to imitate optimal trajectories (Section 3.3.1). Let a tree y be the target output

---

[3] We leave the investigation of better relative position encoding for trees for future work since it is non-trivial to retain constant time complexity of the transformer and have a fully expressive position encoding.

for the current input x. For training the decoder, a **sampling function** (described in Section 3.3.2) is applied to select a partial tree[4] y' from y. A **supervision function** (see Section 3.3.3) is then used that determines the output distribution for every slot, which serves as target in the training of the insertion model. This procedure is detailed in the following.

3.3.1 Optimal Trajectories

The insertion operations described earlier in Section 3.1 can be used to describe a set of actions that transform one partial tree into another. Given an output tree y and the initial tree y_0 that contains a single root node, i.e., y_0 = *(ROOT)*, a *trajectory* from y_0 to y can be defined as a sequence of in total T partial trees (states) $y_0, \dots, y_T$ and corresponding actions $a_0, \dots a_T$, such that, when the actions are applied in succession on y_0, they produce $y = y_T$. Each step thus corresponds to the application of an action a_t to the current partial tree y_t resulting in a new partial tree y_{t+1}, i.e. $y_{t+1} = \mathrm{step}(a_t, y_t)$. Each action a_t is a set of atomic actions that insert a node at one of the insertion slots, which can be either ancestor, descendant, or sibling insertion slots. The atomic actions are described as tuples (k, w), where k is the slot in y_t where a node with label w will be inserted when the action is executed. Note that the type of insertion is characterized by the type of insertion slot.

While many trajectories exists that successfully reach y from y_0, we are interested in *optimal trajectories*, which are trajectories that minimize the number of decoding steps T that need to be performed.

Computing Optimal Trajectories: To practically compute trajectories where we try to minimize the number of steps taken, we rely on the fact that first decoding the most central nodes of a slot's subgraphs enables greater parallelization. For a given tree y_t at decoding step t that is a partial tree of the original tree y, we (1) align y_t and y and determine which nodes are allowed to be inserted in every slot in y_t and (2) compute which of these nodes is the best in order to minimize the number of decoding steps.

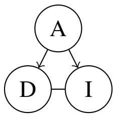

Figure 3: A partial tree of the tree in Fig. 2.

Computing allowed insertions: Given a partial tree y_t (e.g. Fig. 3) aligned with the original tree y (e.g. Fig. 2), we first compute the set of allowed insertions C_k for every slot k.

For an **ancestor insertion slot** ($\wedge$) in y_t associated with some node n, C_k corresponds to the nodes from y on the path from n up to the *lowest used ancestor* of n. The lowest used ancestor $\mathrm{lua}(n)$ of a node n from the partial tree y_t is the lowest[5] node in the original tree y that is an ancestor of n as well as *any* other node from y_t. That is, for node I in the example tree in Fig. 2, the lowest used ancestor of node I is the node B: $\mathrm{lua}(I) = B$ and the set of allowed nodes for I's ancestor slot is $\{F, G\}$.

For a **descendant insertion slot** ($\vee$), the set of allowed nodes C_k is the set of all descendants of n in y , if n doesn't have children in the partial tree y_t. Otherwise, it's the set of all of its children in y_t that are also an ancestor for *all* the children of n in the partial tree y_t.

Finally, for a **sibling insertion slot** (–), to find the set of allowed nodes, we first find the lowest common ancestor in the original tree y of the slot's left node l and right node r. The lowest common ancestor $\mathrm{lca}(n, n')$ of two nodes n and n' in a tree is the lowest node that is an ancestor of both n and n'. We first determine the set C of children of $\mathrm{lca}(l, r)$ that are between the children of $\mathrm{lca}(l, r)$ and that are also the ancestors of nodes l or r. The set of allowed nodes C_k for this slot is then the set C as well as all their descendants in y. In the example, the sibling insertion slot between D and I should accept only the node K since $\mathrm{lca}(D, I)$ is B and K is the only node between C and F, which are the ancestors of D and I, respectively, that are also children of $\mathrm{lca}(D, I)$. [6]

Computing best insertions: Now, for each insertion slot k in y_t, we are given the set C_k of nodes allowed to be inserted. The node $n \in C_k$

---

[4] Note that what we refer to as a partial tree is not the same as a subtree. A subtree retains all the descendants starting from a certain parent node. In contrast, we refer with partial tree to any tree consisting of nodes that can also be found in the original tree, and which can be extended to the full tree y by means of the defined insertion operations.

[5] Lowest and highest refer to tree depth, i.e., the root node is the highest node in the tree.

[6] Note that inserting E between D and I will lead to a tree from which we can't recover the original tree since D, E and I now assume the same parent and there is no action defined to separate them under different parents.

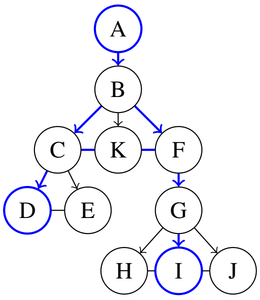

Figure 4: The tree from Fig. 2 and the nodes and edges covered by the partial tree from Fig. 3. Nodes with the same labels in Figures 2 and 3 are aligned.

to be inserted in slot k in order to minimize the number of decoding steps is the most central node in C_k.

The centrality of a node is computed over the subgraph G_k of the original tree y that contains the nodes in C_k, as well as their descendants that are not separated by the partial tree. The *closeness centrality* is used:

$$H_{G_k}(n) = \frac{1}{\sum_{n' \in G_k} d(n, n')} \quad , \tag{4}$$

where $d(n, n')$ is the distance between nodes n and n' in the original tree, which is the minimum number of steps necessary to reach n from n'.

The node picked to be inserted into some slot k is then the one with highest centrality for the slot.

Computing best actions: The best action for a certain partial tree y_t of y then consists of the insertions that are the best for every of its insertion slots. If C_k is empty for some insertion slot, a dummy insertion operation is used that does not insert anything.

3.3.2 Partial Tree Sampling

Rather than sampling all possible partial trees, which would be equivalent to the method described by Stern et al. (2019), we use a different method that samples only from the most optimal trajectory. The partial trees that are used for training are only those that occur on one of the optimal trajectories. For efficiency reasons, we precompute a certain number (5 in our experiments) of trajectories, where we randomly sample when ties occur in the centrality measure, and reuse these trajectories throughout training.[7] Sampling more than one trajectory could reduce the exposure bias.

3.3.3 Supervising Partial Trees

To produce the target distribution for a slot k, we take the nodes $n \in C_k$ computed as valid insertions, as well as their centralities. Then, we rank the nodes in C_k by centrality scores, where the most central node is the highest-ranked one, receiving rank value 0. Ties in centrality are broken by favouring nodes that are lower in the tree y and further ties are broken alphabetically (based on node label). The target distribution for a slot k is then computed using a softmax:

$$p_k(n) = \frac{e^{-\mathrm{rank}_k(n)/\tau}}{\sum_{n' \in C_k} e^{-\mathrm{rank}_k(n')/\tau}} \quad , \tag{5}$$

where τ is a temperature hyperparameter. $p_k(n)$ is zero if n is not in C_k and $\mathrm{rank}_k(n)$ is rank value given to node n.

For each slot k the model outputs a predictive distribution $\pi_k(n)$. Given the target distributions for all slots the training loss is the sum of the Kullback-Leibler (KL) divergences between the target and predictive distributions for all slots:

$$-\sum_k \sum_{n \in \mathcal{G}} p_k(n) \log \frac{\pi_k(n)}{p_k(n)} \quad , \tag{6}$$

where $\mathcal{G}$ denotes the set of all possible node labels.

4 Experiments[8]

We run experiments on the OVERNIGHT (Wang et al., 2015) dataset and report the results in Tables 1 and 2. A description of the dataset and its statistics are provided in Appendix A.

Evaluation: The metric reported is logical form accuracy, which we compute by (1) taking the predictions of the models, (2) balancing parentheses on the left and on the right and (3) computing whether the trees are the same.[9]

Data preprocessing: We notice that many examples in the Overnight dataset contain nested filters. This fact was ignored during training and evaluation of some models in previous work (Damonte

---

[7] Note that only one of the partial trees y_t of some example is used in a single epoch.

[8] The code is provided at `https://github.com/lukovnikov/parseq/tree/crforen/parseq/scripts_insert`

[9] The trees are considered the same if every node's children occur in both trees in the same order, if the node's children should be ordered (for example argmax expressions) and if every node's children occur in both trees in *any* order for nodes of which the children should not be ordered (for example SW:concat and a collection of filter conditions).

et al., 2019; Xu et al., 2020). During evaluation, an example was considered wrong if the filters were decoded in a different nesting order. For this reason, we use slightly simplified logical forms for the Overnight dataset that remove nesting between filters (see also Appendix B). Evaluation with these logical forms better reflects the true meaning of the queries. See Appendix C for an example.

For the sequence-based decoder, we found that it is necessary to use **numbered tokens** to to successfully train the model. We simply replace every token "X" by a token "X_d", where d specifies how many tokens "X" have been observed before and including the current token "X". For example, this would transform "(A (B C) (B D))" into "$(_1$ A_1 $(_2$ B_1 C_1 $)_1$ $(_3$ B_2 D_1 $)_2$ $)_3$". This ensures that every token in the sequence is unique. We use numbered tokens in all the experiments, both for the sequence-based as well as the tree-based decoders. We leave a deeper investigation of the effect of numbering tokens for future work.

Baselines: We considered the following models as baselines in our experiments: (1) BERT with a transformer decoder (BERT+Transformer), (2) a re-implemented insertion transformer (Seq-Insert) in both binary and uniform supervision settings (BERT is used for encoding the input), and (3) BERT with a regular tree-based decoder similar to the one proposed by Dong and Lapata (2016) (BERT+TreeGRU; see also Appendix E). Note that both (1) and (3) are essentially left-to-right decoders and can't decode in parallel. Even though they are not directly comparable, we also include the results reported by Chen et al. (2018a). Rather than decoding logical forms (i.e. Lisp-style expressions), they decode an intermediate representation called a query graph using special graph generation actions (e.g. add variable node, add edge). Moreover, the evaluation is based on execution accuracy instead of logical form accuracy (Xu et al., 2020)

Training details: We train all models using Adam (Kingma and Ba, 2014), varying the initial learning rate within $\{0.0001, 0.00005, 0.00001\}$ and experimenting with dropout rates out of $\{0.0, 0.1, 0.2, 0.4\}$. We also experimented with values from $\{1.0, 0.1\}$ for the temperature τ. The validation set for each domain was constructed by taking a random 20% subset of the training examples. This validation set was used for early stopping. We randomly searched the hyperparameter space and took the best performing parameters based on its validation performance on the *publications* domain. These hyperparameters[10] were used for training the models on the other domains. We train each model three times on every domain independently (with the same seed values shared over domains and settings) and also report the average over the domains. For all our experiments, we use a 12-layer BERT model from Huggingface (Wolf et al., 2020) as the encoder and use a transformer with 6 layers, 12 heads, and 768 dimension for the decoder.

4.1 Results

Table 1 shows the results on all domains of the Overnight dataset. Table 2 shows the results on four Overnight domains, and indicates the accuracy and speed-up measured in terms of the number of decoding steps compared to the BERT+Transformer left-to-right baseline.

Baseline equivalence: First, we establish that the results of our left-to-right baseline are on par with previous reported numbers in similar settings. This is shown in the middle part of Table 1, where our "BERT+Transformer" beats Xu et al. (2020)'s BERT+LSTM baseline by a small margin.

Sequence-based insertion baseline results: In the bottom part of Table 1, we report numbers for the slightly simplified logical forms. The performance of the parallel sequence insertion decoder with binary supervision is on par with that of our left-to-right baseline. The uniform variant performs slightly worse. Concerning the speed-up obtained, Table 2 shows that the binary supervision is (more than ×2) more effective than uniform supervision. However, even the binary supervised version lags behind the theoretically best possible speed-up ("Seq-Insert Th.B.").

Tree-based insertion results: The tree-based insertion decoding procedure described in Section 3.1 enables a decoding complexity that is lower than the theoretically best $\mathcal{O}(log_2(N))$ of the sequence-based insertion decoder. However, the theoretically best possible speed-up for tree insertion decoding heavily depends on the tree structure. We report this number ("Tree-Insert Th.B.") for the different domains in Table 2. The theoretically best possible speedup of the tree-based insertion decoder is

---

[10] dropout rate=0.2, learning rate=0.00005, τ=0.1, batch size=10/30/50, cosine learning rate scheduler with 20 epoch warm-up, 200 epochs

	cal.	blo.	hou.	res.	pub.	rec.	soc.	bas.	avg (± std)
Seq2Action (Chen et al., 2018b)	81.5	61.4	74.1	80.7	80.7	82.9	82.1	88.2	79.0
Shift-Reduce (Damonte et al., 2019)	43.5	25.1	29.6	37.3	32.9	58.3	51.2	69.6	43.4
BERT+LSTM (Xu et al., 2020)	58.3	42.6	48.7	55.4	64.6	68.5	70.4	84.1	61.6
BERT+Transformer	61.3	45.4	50.8	58.7	63.4	76.4	69.9	85.4	63.9
BERT+Transformer	80.0	53.9	70.9	83.4	70.4	83.3	73.8	84.0	75.0 ± 1.3
BERT+TreeGRU	77.4	49.7	67.9	82.6	73.3	81.2	73.0	84.4	73.7 ± 1.5
Seq-Insert (Binary)	78.2	50.9	67.5	81.0	72.5	81.2	70.9	84.1	73.3 ± 1.6
Seq-Insert (Uniform)	79.0	47.8	68.3	80.5	70.0	82.3	70.0	83.7	72.7 ± 1.6
Tree-Insert	77.4	51.9	71.8	81.1	72.9	82.9	73.2	84.4	74.4 ± 1.3

Table 1: Results on the test set of the different domains of the Overnight dataset. Top part: denotation accuracy. Middle part: logical form accuracy on original trees. Bottom part: logical form accuracy on simplified trees. In the last column, we report the average accuracy over the domains as well as the average (over domains) of standard deviations of accuracies for every domain over different seeds.

slightly higher than that of the sequence-based insertion decoder: an average of 0.48 decoding steps can be gained, which corresponds to a potential reduction of decoding steps of ×1.08 on average. However, one needs to investigate how close an actual tree-based insertion decoder can get to the theoretically best number in practice.

The bottom part of Table 2 reports the results for the tree-based insertion decoder. Our insertion-based tree decoder achieves **competitive accuracy** to both the left-to-right baseline as well as the sequence insertion decoder, while requiring **fewer decoding steps** than the strong sequence insertion baseline: an average of 0.7 decoding steps is gained, which corresponds to an average of a ×1.11 reduction in the number of decoding steps.

Ablation study: In Table 3, we assess the effects of some design choices and hyperparameters. Decreasing the number of trajectories used during training from 5 to 1 results in a significant decrease in accuracy. The chosen temperature τ also affects training. When τ is set to a high value, the target distribution becomes more uniform, which allows the model to use insertions that are not covered by the trained trajectories making it more likely to fail due to exposure bias, which is reflected in the poor result for $\tau = 10$. Using absolute instead of relative position information leads only to a slight decrease in accuracy. Note that using just absolute positioning requires to process the structure tokens because the structure information contained in relative positions is not being used. The child relations (§3.2.2) appear to not have any significant effect.

5 Discussion

To obtain an efficient decoder other considerations next to the number of decoding steps must be taken into account. The actual execution speed and computational load also heavily depend on (1) the size of processed data and (2) efficiency of implementation. In the proposed tree decoder, the *effective* size of model inputs is *smaller* than for the sequence insertion decoder since it does not need parentheses. Thus, when implemented efficiently, it requires less computation and less memory, which is attractive given the quadratic memory complexity of the transformer's attention.

Another point worth noting is that the best possible speed-up of the proposed insertion-based tree decoder heavily depends on the data. However, considering that the proposed model has between two and three insertion slots per token, and the sequence insertion decoder only one, our method can expand trees much faster.

A limitation of the proposed insertion-based tree decoder is that it, like the Insertion Transformer, is unable to recover from mistakes made during decoding. In contrast, the Levenshtein transformer (Gu et al., 2019b) also defines deletion operations and can thus recover from erroneous predictions. Extending the insertion-based tree decoder to also delete part of the trees is an interesting direction for future work. While we simply sample from a number of optimal trajectories, a more general decoder that allows for deletion would have to be trained using dynamic oracles (Goldberg and Nivre, 2012; Ross et al., 2011; Vlachos and Clark, 2014) or reinforcement learning. This is likely to

	calendar		restaurants		publications		recipes		average	
	Acc.	# Steps	Acc.	# Steps	Acc.	# Steps	Acc.	# Steps	Acc.	# St.
BERT+Transf.	80.0	40.4 (×1.0)	83.4	37.8 (×1.0)	70.4	40.9 (×1.0)	83.3	36.6 (×1.0)	79.3	38.9
BERT+TreeGRU	77.4	41.2 (×1.0)	82.6	38.1 (×1.0)	73.3	41.3 (×1.0)	81.2	36.7 (×1.0)	78.6	39.3
Seq-Insert Th.B.	–	6.7 (×6.0)	–	6.6 (×5.7)	–	6.7 (×6.1)	–	6.5 (×5.6)	–	6.6
Seq-Insert (Bin.)	78.2	7.2 (×5.6)	81.0	6.9 (×5.5)	72.5	7.3 (×5.6)	81.2	6.9 (×5.3)	78.2	7.1
Seq-Insert (Uni.)	79.0	18.3 (×2.2)	80.5	16.9 (×2.2)	70.0	17.6 (×2.3)	82.3	16.8 (×2.2)	77.9	17.4
Tree-Insert Th.B.	–	6.2 (×6.5)	–	6.1 (×6.2)	–	6.3 (×6.5)	–	6.0 (×6.1)	–	6.2
Tree-Insert	77.4	6.4 (×6.4)	81.1	6.2 (×6.2)	72.9	6.6 (×6.3)	82.9	6.2 (×5.8)	78.6	6.4

Table 2: Logical form accuracy on simplified logical forms and number of decoding steps used on the test set of the different domains of the Overnight dataset. “Bin” stands for binary supervision, “Uni.” for uniform.

Ablation	publications	calendar	recipes
$\tau = 1.$	69.8 (-3.1)	74.0 (-3.4)	82.3 (-0.6)
$\tau = 10.$	24.4 (-48.5)	20.0 (-57.4)	28.9 (-54.0)
# traj. = 1	67.3 (-5.6)	71.6 (-5.8)	66.5 (-16.4)
abs. pos	71.0 (-1.9)	77.0 (-0.4)	79.3 (-3.6)
no child. rel.	72.5 (-0.4)	78.8 (+1.4)	81.9 (-1.0)

Table 3: Results of an ablation study on the PUBLICATIONS domain.

decrease the exposure bias as the model is exposed to many different trajectories.

6 Related Work

To the best of our knowledge, to this date, only two other works have investigated non-autoregressive methods for trees and semantic parsing. The work of Rubin and Berant (2020) proposes a bottom-up tree decoder, however, their decoder is limited to being linear in depth and thus less parallelizable for deeper and narrower trees. In contrast, the Insertion Transformer can be sub-linear both in depth and breadth. Zhu et al. (2020) apply the Insertion Transformer to semantic parsing, albeit on SNIPS (Coucke et al., 2018), ATIS (Price, 1990) and TOP (Gupta et al., 2018) datasets and focuses on cross-lingual performance.

Only a few transformer architectures specialized for trees have been proposed to date. The work of Shiv and Quirk (2019) proposes a new positional encoding to improve tree representation using transformers. The work of Anonymous (2020), similarly to ours, investigates relative positioning for transformers operating on trees. Other architectures to encode trees exist as well, such as the TreeLSTM (Tai et al., 2015), however, a transformer-based model enjoys greater parallelism and direct modeling of long-range dependencies.

Several methods have recently been proposed for non-autoregressive decoding and insertion-based decoding. Stern et al. (2019), Gu et al. (2019a) and Gu et al. (2019b) experiment with insertion-based decoding where Gu et al. (2019b) also support deletion operations. Ma et al. (2019), on the other hand, develop a non-autoregressive sequence generation model using normalizing flows (Rezende and Mohamed, 2015). Some other examples of non-autoregressive decoding for NMT are (Ghazvininejad et al., 2019; Gu et al., 2017; Kasai et al., 2020).

Various works have recently explored the use of neural networks for semantic parsing. Dong and Lapata (2016) explore both sequence-based as well as tree-structured decoding. Alvarez-Melis and Jaakkola (2017) propose an improved tree decoder that uses additional classifiers to perform structure prediction rather than using structure tokens. Chen et al. (2018c) propose a binary tree generator for code translation.

7 Conclusion

In this work, we presented a novel neural network based method for non-autoregressive decoding of trees. We define insertion operations used in the step-wise decoding process that guarantee that intermediate structures are trees. This results in a reduction of the number of decoding steps and allows to exploit tree structure. Experiments on semantic parsing show competitive accuracy and a significantly decreased number of decoding steps, compared to strong autoregressive and non-autoregressive baselines.

Acknowledgments

We thank the reviewers for their helpful comments and suggestions. We also thank Prof. Dr. Jens Lehmann, Priyansh Trivedi, and Gaurav Maheshwari for their comments and help with this work.

References

David Alvarez-Melis and Tommi S Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *ICLR*.

Anonymous. 2020. A structural transformer with relative positions in trees for code-to-sequence tasks.

Bo Chen, Le Sun, and Xianpei Han. 2018a. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777, Melbourne, Australia. Association for Computational Linguistics.

Bo Chen, Le Sun, and Xianpei Han. 2018b. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777.

Xinyun Chen, Chang Liu, and Dawn Song. 2018c. Tree-to-tree neural networks for program translation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2552–2562.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces.

Marco Damonte, Rahul Goel Tagyoung Chung, and Amazon Alexa AI. 2019. Practical semantic parsing for spoken language understanding. *NAACL HLT 2019*, pages 16–23.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019b. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pages 11181–11191. Curran Associates, Inc.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Parallel machine translation with disentangled context transformer. *arXiv preprint arXiv:2001.05136*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4273–4283.

Patti Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990.*

Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Ohad Rubin and Jonathan Berant. 2020. Smbop: Semi-autoregressive bottom-up semantic parsing. *arXiv preprint arXiv:2010.12412*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Vighnesh Leonardo Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *NeurIPS*, pages 12058–12068.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–560.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Silei Xu, Sina Semnani, Giovanni Campagna, and Monica Lam. 2020. Autoqa: From databases to q&a semantic parsers with only synthetic training data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 422–434.

Qile Zhu, Haidar Khan, Saleh Soltan, Stephen Rawls, and Wael Hamza. 2020. Don't parse, insert: Multilingual semantic parsing with insertion based decoding. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 496–506.

A Dataset Description and Statistics

The OVERNIGHT dataset consists of pairs of natural language questions and corresponding formal language queries from 8 distinct domains, such as "publications" and "restaurants". The dataset was generated as follows: (1) first, a grammar was defined, which consists of a general part applicable for any domain, and a domain-specific part that specifies a *seed lexicon* mapping between predicates and NL. Then, (2) a number of examples was generated, which at this point consist of (i) a canonical utterance and (ii) a formal query. Finally, (3) the canonical utterances are paraphrased by Amazon Mechanical Turk workers to generate more natural examples. See Appendix C for an example question-query pair from the data set.

See Table 4 for dataset statistics.

Domain	#Train	#Test	Avg. tree size	Avg. tree depth	Avg. seq. length
cal.	669	168	17.5	5.7	39.1
blo.	1596	399	18.8	5.8	42.1
hou.	752	189	17.3	5.6	38.8
res.	1325	332	16.3	5.4	37.1
pub.	640	161	18.3	5.8	41.1
rec.	864	216	16.1	5.7	36.3
soc.	3535	884	25.3	8.7	57.3
bas.	1561	391	19.8	8.2	45.1

Table 4: Statistics for the OVERNIGHT dataset after simplification.

B Preprocessing

This simplified version is obtained by (1) merging the "call" node with the function of the call (e.g. replacing *(call SW:someFunction ...)* with *(call-SW:someFunction ...))* and (2) merging nested filters into a multi-conditional expression where the order of conditions doesn't matter. Note that these simplifications are easily reversible. During training, we sort the different parts of a multi-conditional filter expression alphabetically and during evaluation, we consider the different conditions as unordered and accept them in any decoded order. We use this simplified version for all further experiments.

C An example from the Overnight dataset

An actual example from the "publications" domain from the OVERNIGHT dataset is the following.

The natural language question is:

"*find an article published in 2004*".

The formal query corresponding to this question is (after preprocessing):

Listing 1: Example query

```
(call_SW_listValue
  (filter
    (call_SW_getProperty
      (call_SW_singleton
        (en.article ))
      (string
        (!type )))
    (condition
      (string
        (publication_date ))
      (string
        (= ))
      (date
        (2004 )
        (−1 )
        (−1 )))))
```

D An example of insertion-based decoding for a real example.

Consider the example in Appendix C. This example can be decoded in 5 steps using tree-based insertion actions as elaborated in the following. Note that multiple insertion action sequences would work with the same number of steps.

Listing 2: Example query

```
1.    (filter)
```

Listing 3: Example query

```
2. (call_SW_listValue
     (filter
         (string)))
```

Listing 4: Example query

```
3. (call_SW_listValue
     (filter
         (call_SW_singleton)
       (condition
         (string)
```

```
(string
  (= ))
(date))))
```

Listing 5: Example query

4.
```
(call_SW_listValue
  (filter
    (call_SW_getProperty
      (call_SW_singleton
        (en.article ))
      (string))
    (condition
      (string
        (publication_date ))
      (string
        (= ))
      (date
        (−1 )))))
```

5. Same as original tree in Appendix C.

E TreeGRU

We implement the tree-based left-to-right baseline decoder similarly to Dong and Lapata (2016). However, in our implementation, we use depth-first instead of breadth-first decoding. We achieve similar conditioning in the different decoding steps by manipulating which of the previous states are used as the previous sibling state and the parent state. If the previously decoded token was an opening parenthesis "(", then the parent state correspond to the state of the GRU after producing the opening parenthesis (this is the previous state), the previous sibling state is set equal to the parent state. Following Dong and Lapata (2016), we use the parent state explicitly as part of the GRU input in every step. When a closing parenthesis ")" is decoded, the parent of the previous parent state is used as the parent state, the state accumulated from the closed subtree is discarded, and the previous parent state becomes the previous sibling state. Thus, we make sure that similarly to Dong and Lapata (2016)'s breadth-first decoding, the different branches aren't directly conditioned on each other through decoder states.

Is the Lottery Fair? Evaluating Winning Tickets Across Demographics

Victor Petrén Bach Hansen,[1 2] **Anders Søgaard**[1]
[1]Department of Computer Science, University of Copenhagen, Denmark
[2]Topdanmark A/S, Denmark
{victor.petren,soegaard}@di.ku.dk

Abstract

Recent studies have suggested that weight pruning, e.g. using lottery ticket extraction techniques (Frankle and Carbin, 2018), comes at the risk of compromising the group fairness of machine learning models (Paganini, 2020; Hooker et al., 2020), but to the best of our knowledge, no one has empirically evaluated this hypothesis at scale in the context of natural language processing. We present experiments with two text classification datasets annotated with demographic information: the Trustpilot Corpus (sentiment) and CivilComments (toxicity). We evaluate the fairness of lottery ticket extraction through layer-wise and global weight pruning across three languages and two tasks. Our results suggest that there is a small increase in group disparity, which is most pronounced at high pruning rates and correlates with instability. The fairness of models trained with distributionally robust optimization objectives is sometimes less sensitive to pruning, but results are not consistent. The code for our experiments is available at `https://github.com/vpetren/fairness_lottery`.

1 Introduction

Heavily pruning deep neural network models is a way of reducing inference cost for resource-constrained environments, but does weight-pruning of deep neural networks increase their unfairness? Several recent papers suggest this (Paganini, 2020; Hooker et al., 2020), based on experiments from face and digit recognition, but does this also hold for natural language processing (NLP) models? Systematic biases may easily be exacerbated by pruning interventions in high-dimensional problems because of feature swamping effects (Sutton et al., 2006). Overparameterized deep neural networks generalize well, in part because they can hedge their bets and rely on multitudes of weak evidence rather than the most prominent independent variables. Sparse models do not have that luxury and are therefore more sensitive to shifts (Globerson and Roweis, 2006; Søgaard, 2013).

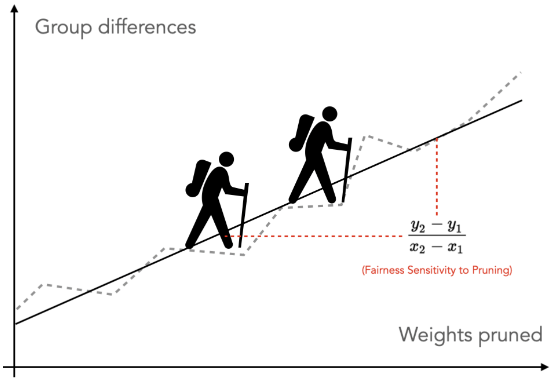

Figure 1: Fairness Sensitivity to Pruning (FSP): the gradient of the linear fit of (the logarithm of) the pruning ratio to min-max group-level disparity. We use this to quantify the sensitivity of Rawlsian min-max fairness to weight pruning across architectures, pruning strategies and datasets.

We introduce a *fairness sensitivity to pruning* metric that measures how Rawlsian min-max fairness across demographic groups changes with weight pruning. We estimate this sensitivity by taking the gradient of the linear fit of the logarithm of the pruning ratio to min-max group-level disparity. We show that across four datasets, fairness sensitivity to pruning is similar for layer-wise and global pruning strategies (Frankle and Carbin, 2018), as well as for text classifiers based on feed-forward and recurrent neural networks. Subsequently, we consider the impact of a popular robust optimization strategy designed to improve the fairness of classification models (Hashimoto et al., 2018; Sagawa et al., 2020b), on the fairness sensitivity of feed-forward networks.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3214–3224
August 1–6, 2021. ©2021 Association for Computational Linguistics

Contributions We are, to the best of our knowledge, the first to study the impact of weight pruning on fairness in NLP at scale. We introduce a *fairness sensitivity to pruning* (FSP) metric that measures how Rawlsian min-max fairness across demographic groups decreases with weight pruning. We evaluate FSP across two architectures, two pruning strategies and two datasets, including multilingual sentiment classification and English toxicity classification. Our results suggest that pruning increases group-level performance disparities, but mostly at high pruning rates and with some variance across architectures and pruning strategies. Group-level disparities seem to be in part a result of the instability of weight pruning. We compare FSP between our baseline empirical risk models and robust models induced with Distributional Robust Optimization (DRO) (Hashimoto et al., 2018; Sagawa et al., 2020b). Our results show that weight pruning in combination with DRO can *sometimes* (8/16 cases here) be used to induce fairer, sparse classifiers, but the effect is not significant ($p \sim 0.18$) across our experiments.

2 Related Work

Pruning neural networks The literature on pruning neural networks is decades old (Mozer and Smolensky, 1989; Cun et al., 1990; Hassibi and Stork, 1993), but has recently seen a resurgence with the all-encompassing success of neural networks and the need for small and fast on-device model inference (Han et al., 2015; Sze et al., 2017; Frankle and Carbin, 2018; Frankle et al., 2019). In NLP, specifically, pruning methods have been applied to recurrent neural networks (Desai et al., 2019; Yu et al., 2020), as well as transformers (Gordon et al., 2020; Brix et al., 2020; Prasanna et al., 2020; Chen et al., 2020; Sanh et al., 2020).

Fairness in pruned models Measuring fairness in pruned models is an unexplored area. However, Paganini (2020) evaluates the fairness, i.e., the difference between the best- and worst-case groups, of lottery ticket-style weight pruning for digit recognition problems: Specifically, they retrain models for a fixed number of iterations using global unstructured pruning. In addition, they present a meta-regression study suggesting that underrepresented and more complex classes are most severely affected by pruning procedures. See Hooker et al. (2020) for related work and similar results in face recognition.[1]

Improving fairness Fairness of overparameterized models can be improved by distributionally robust optimization (DRO) (Hashimoto et al., 2018; Levy et al., 2020), or to some extent by simpler post-hoc correction methods such as classifier retraining or group-specific classification thresholds (Menon et al., 2021). DRO minimizes the worst-case expected loss over an uncertainty set of distributions. The uncertainty set represents the distributions we want our model to perform well on. In Sagawa et al. (2020a), the uncertainty set is all possible mixtures of a known set of groups, a variant referred to as Group DRO. Sagawa et al. (2020b) find that subsampling the majority groups can be a way for overparameterized models to achieve both low minority test error as well as low average test error.

3 Pruning methodology

We extract winning lottery tickets from our network according to the iterative procedure outlined in Frankle and Carbin (2018): Given a model $f(x;\theta)$ with initial network parameters θ_0 and mask m_0, for each pruning iteration i, we start by initializing a model $f(x;\theta)$ with initial parameter θ_0 and train it for N epochs, resulting in $f(x;\theta_N)$. After training, we prune a fixed fraction $p \in [0,1]$ from the remaining parameters in θ_N to obtain the mask m_i. The pruned weights are chosen using the L_1 norm, meaning the neurons with the lowest magnitude are masked out. Pruning can either be done w.r.t. individual layers or all of them combined, also referred to as *layer-wise* and *global* pruning. m_i is then carried over to the subsequent pruning iteration $i+1$ with the model $f(x, m_i \odot \theta_0)$ and retrained once again. At iteration i, the fraction of weights pruned is therefore $1-(1-p)^i$.

4 Experiments

4.1 Data

Datasets We examine fairness among heavily pruned models using two text classification datasets: i) The multilingual **Trustpilot** Corpus

---

[1] Bartoldson et al. (2020) arguably present results from object recognition that show the opposite trend: Generalization increases with (layer-wise) pruning. This seems to be a side effect of overparameterization; interestingly, we see the opposite trend for feed-forward networks and layer-wise pruning.

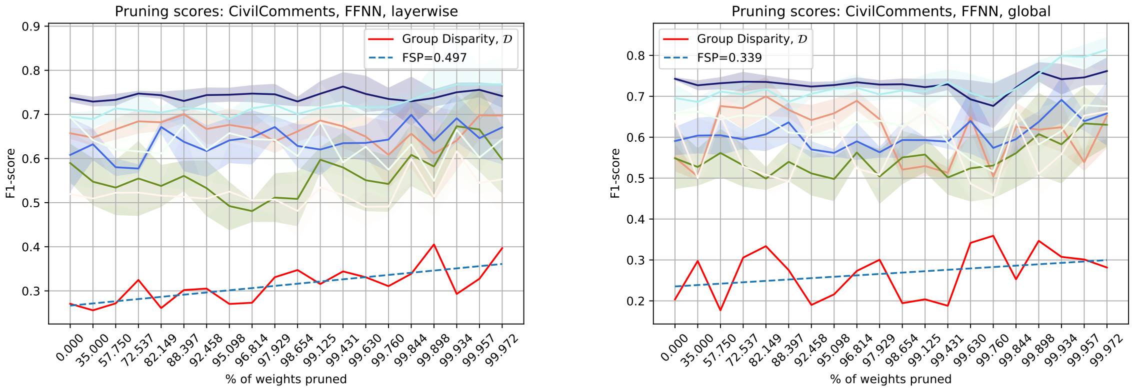

Figure 2: Macro-averaged performance of our feed-forward networks as a function of pruning ratio. Fairness Sensitivity to Pruning (FSP) correspond to the gradient of the linear fit to the min-max differences across individual runs. Results are for CIVILCOMMENTS. The hard line represents the average demographic score over 5 individual runs and the shaded area represents the standard deviation. See the Appendix for similar plots for the Trustpilot Corpus.

(Hovy et al., 2015),[2] which contains user reviews from the Trustpilot website of various companies and services in five different countries (Germany, Denmark, France, United Kingdom and United States). The reviews are based on a one to five star rating scale and some are accompanied by demographic attributes about the author, such as gender, age and location. 2) The **CivilComments** dataset (Borkan et al., 2019),[3] which contains comments annotated for toxicity, for the purpose of hate speech detection. A subset of the comments are also annotated for the protected attributes they address, including gender, race, and religion.

Preprocessing For the Trustpilot Corpus, we divide the data into demographics based on a combination of *gender* (male/female), *age* (young/old) and *location* (NUTS regions). For age, young is defined as being 35 or less. We exclude the French and American parts of the datasets as they do not have properly annotated NUTS regions. For UK and Germany, we use NUTS-1 regions, and for Denmark, where more data is available, we use NUTS-2 regions. We convert the 5-star ratings to binary sentiment labels, grouping 4 and 5 stars as positive, and 1 and 2 as negative. Neutral reviews (three stars) are discarded.[4] Likewise for CivilComments, we threshold comments with a toxicity rating > 0.5 as toxic, and otherwise label them as a non-toxic. This is similar to the binarization performed in Koh et al. (2020). Comments can for each demographic sub-attribute contain multiple partial values (e.g. $asian = 0.3$, $black = 0.4$ for the *race* attribute), so for each annotated attribute we assign it the sub-attribute with the largest value. In our experiments we consider demographics based on combinations of the *race* and *gender* attributes. For each language and dataset we randomly sample 100, 200 or 500 of each demographic as test sets, based on the the amount of annotated datapoints in the dataset, and use a 80-20 split of the remaining data for training and validation. If a demographic contains less than the specified number of datapoints, we disregard it. Due to high class imbalance, the majority class for our train-val data is downsampled to match the minority class. Table 1 shows the statistics for the respective datasets we train and evaluate on.

Dataset	Train	Val	N	S
Trustpilot-DK	222229	55557	20	500
Trustpilot-DE	26146	6536	42	100
Trustpilot-UK	127965	31991	50	200
CivilComments	357602	89400	7	100

Table 1: Detailed dataset statistics. N refers to the number of discrete demographics in the dataset and S is the size of each demographic test set.

4.2 Models

We consider simple **FFNN** (Rumelhart et al., 1986) and **LSTM** (Hochreiter and Schmidhuber, 1997)

---

[2] https://bitbucket.org/lowlands/release/src/master/WWW2015/data/

[3] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data

[4] This binarization scheme is standard; see, e.g., Gupta et al. (2020) and Desai et al. (2019)

FFNN

Dataset	E_{dim}	h_{dim}	B	N
Trustpilot-DK	128	256	15	32
Trustpilot-DE	128	256	15	8
Trustpilot-UK	128	256	15	16
CivilComments	128	256	15	32

LSTM

Dataset	E_{dim}	h_{dim}	B	N
Trustpilot-DK	128	256	10	64
Trustpilot-DE	128	256	15	16
Trustpilot-UK	128	256	10	32
CivilComments	128	256	10	64

Table 2: FFNN and LSTM hyperparameters. E_{dim} is embedding layer size, h_{dim} is hidden layer size, B is batch size and N is number of epochs. Both the layer-wise and global pruning structures use the same set of hyperparameters.

neural networks for text classification.

FFNN The FFNN consists of the following: The embedding layer, which maps every token id in the text to a fixed size vector as a bag-of-embeddings and sums them together, resulting in a single representation $e \in \mathbb{R}^{|}E_{dim}|$, followed by 3 fully connected layers of size $\mathbb{R}^{|E_{dim} \times h|}$, $\mathbb{R}^{|h \times h|}$ and $\mathbb{R}^{|h \times 2|}$ respectively. We use the hyperbolic tangent activation between layers and each linear layer is initialized using He initialization (He et al., 2015).

LSTM The LSTM network is a 2-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) which encodes our input text, followed by a fully connected layer for classification. The weights are initialized using $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{hidden\_size}$ and the final fully connected layer uses He initialization. See all model hyperparameters used in Table 2.

Both the FFNN and LSTM models are trained using the Adam optimizer (Kingma and Ba, 2017) with a learning rate of $1e-3$ and a weight decay of $1e-4$.

Distributionally Robust Optimization Loss Additionally, we also train our models with DRO loss (Levy et al., 2020). We use the implementation provided by Levy et al. (2020) [5]. For our experiments, a χ^2 uncertainty set of size 1 is used.

For all of our experiments, we extract our winning tickets over 20 pruning iterations and use a pruning rate of $p = 0.35$. We run a total of 5 independent runs for each model-dataset combination.

		Trustpilot			CC	Avg
		da	de	en	en	
FFNN	lw	-0.183	0.281	-0.230	0.497	0.091
	gl	0.227	1.375	1.054	0.339	0.749
FFNN-DRO	lw	-0.044	0.321	0.143	0.089	0.127
	gl	0.351	0.875	-0.040	0.368	0.388
LSTM	lw	0.221	0.411	0.206	0.823	0.415
	gl	1.099	0.198	0.352	0.252	0.475
LSTM-DRO	lw	0.263	-0.282	-0.082	1.335	0.309
	gl	0.262	-0.609	0.544	0.006	0.051

Table 3: FSP values across architectures, layer-wise (lw) and global (gl) pruning, and the four datasets. Our main observation is that FSP values are almost consistently positive, and slightly higher for global pruning. DRO does not consistently reduce FSP; we highlight cases where it does.

4.3 Measuring group disparity

At each pruning step we measure the group disparity $\mathcal{D}$, from a set of demographics D, between repeated runs R, by computing the maximum difference of F_1 scores as follows:[6]

$$\mathcal{D} = \max_{d_m \in D} \max_{d_{n \neq m} \in D} \max_{r_i \in R} \max_{r_{j \neq i} \in R} |F_{1_{r_i d_m}} - F_{1_{r_j d_n}}| \tag{1}$$

Intuitively, this corresponds to the difference between the highest scoring run for the highest scoring demographic and the lowest counterpart. We compute FSP by taking the gradient of the linear fit of $\mathcal{D}$ over a P pruning steps multiplied by 100.

5 Results

Main experiments Our first set of results evaluate FSP across architectures, datasets, and pruning techniques. In 14/16 combinations of FFNN and LSTM neural networks, the Trustpilot Corpus and CivilComments, layer-wise and global pruning, we see positive FSP values. In other words, weight pruning leads to higher group-level performance disparities, i.e., less fairness. Comparing layer-wise and global pruning, we note that group disparity is generally higher for global pruning. In Figure 2, we present two plots - for layer-wise and global pruning of a feed-forward network trained on CivilComments. The remaining plots are presented in the Appendix. The FSP values are listed in Table 3. FFNNs exhibit very high FSP values

---

[5] `https://github.com/daniellevy/fast-dro/`

[6] Maximum discrepancy has also been used as a measure of fairness in Calmon et al. (2017); Alabi et al. (2018). See Williamson and Menon (2019) for discussion.

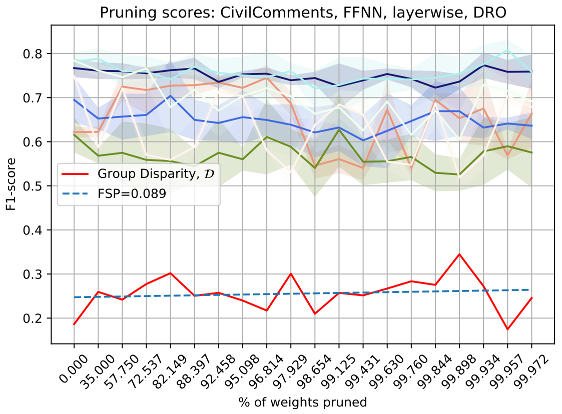

Figure 3: FSP for Distributional Robust Optimization

with global pruning, but while global pruning increases unfairness, layer-wise pruning does not. For LSTMs, the effects of the two pruning strategies are similar: Both lead to moderate increases in group disparities.[7] In a couple of instances we witnessed model degeneration due to heavy pruning resulting in single-class prediction before 20 pruning iterations. The plots and FSP values exclude these datapoints as they are not relevant for our analysis.

Distributionally Robust Optimization We ran comparable experiments using DRO loss (Hashimoto et al., 2018) to see whether the adverse effects of weight pruning on min-max fairness could be reduced by training with a more robust objective. This seems to hold true in some instances. We present a single plot for DRO in Figure 3, for feed-forward networks, layer-wise pruning on CivilComments; see the Appendix for more plots. Comparing with Figure 2 (left) the FSP metric is considerably lower than for baseline empirical risk minimization (0.089 vs. 0.497) while maintaining equal, or even better, performance at high pruning rates; but note from the red numbers in Table 3, that we only see this type of reduction in FSP in 3/8 cases for FFNNs, but DRO does reduce the average FSP for global pruning. In 5/8 cases for the LSTM, however, DRO does improves fairness, reducing the average FSP with both layer-wise and global pruning.

6 Conclusion

In this work, we take a first step in examining group disparity among heavily pruned models, using lottery ticket extraction, in NLP. We measure group disparity, using *fairness sensitivity to pruning*, on the Trustpilot Corpus, a sentiment classification dataset covering 3 languages, as well as CivilComments, a toxicity classification dataset, for both feed-forward and recurrent neural networks. We find that models subject to heavy pruning are more susceptible to higher levels of group disparity, but that this effect can to some degree be mitigated using distributionally robust optimization objectives.

Acknowledgements

This work was funded by the Innovation Fund Denmark and Topdanmark.

References

Daniel Alabi, Nicole Immorlica, and Adam Kalai. 2018. Unleashing linear optimizers for group-fair learning and optimization. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 2043–2066. PMLR.

Brian Bartoldson, Ari S. Morcos, Adrian Barbu, and Gordon Erlebacher. 2020. The generalization-stability tradeoff in neural network pruning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561.

Christopher Brix, Parnia Bahar, and Hermann Ney. 2020. Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture.

Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. 2017. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks.

Yann Le Cun, John S. Denker, and Sara A. Solla. 1990. *Optimal Brain Damage*, page 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

---

[7] While fairness correlates with stability, the difference between FFNNs and LSTMs is not explained by stability differences (see plots in the Appendix), but should probably be attributed to the general performance differences between FFNNs and LSTMs, as well as relative overparameterization in FFNNs (see Footnote 1).

Shrey Desai, Hongyuan Zhan, and Ahmed Aly. 2019. Evaluating lottery tickets under distributional shifts. *CoRR*, abs/1910.12708.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2019. Linear mode connectivity and the lottery ticket hypothesis. *CoRR*, abs/1912.05671.

Amir Globerson and Sam Roweis. 2006. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 353–360, New York, NY, USA. Association for Computing Machinery.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing {bert}: Studying the effects of weight pruning on transfer learning.

Aakriti Gupta, Kapil Thadani, and Neil O'Hare. 2020. Effective few-shot classification with transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1061–1066, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1135–1143, Cambridge, MA, USA. MIT Press.

Tatsunori B. Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness without demographics in repeated loss minimization.

Babak Hassibi and David Stork. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pages 164–171. Morgan-Kaufmann.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. 2020. What do compressed deep neural networks forget?

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. *User Review Sites as a Resource for Large-Scale Sociolinguistic Studies*, page 452–461. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2020. Wilds: A benchmark of in-the-wild distribution shifts.

Daniel Levy, Yair Carmon, John C Duchi, and Aaron Sidford. 2020. Large-scale methods for distributionally robust optimization. In *Advances in Neural Information Processing Systems*.

Aditya Krishna Menon, Ankit Singh Rawat, and Sanjiv Kumar. 2021. Overparameterisation and worst-case generalisation: friend or foe? In *ICLR*.

Michael C. Mozer and Paul Smolensky. 1989. *Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment*, page 107–115. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Michela Paganini. 2020. Prune responsibly.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning internal representations by error propagation. In David E. Rumelhart and James L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020a. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020b. An investigation of why overparameterization exacerbates spurious correlations.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning.

Anders Søgaard. 2013. Part-of-speech tagging with antagonistic adversaries. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 640–644, Sofia, Bulgaria. Association for Computational Linguistics.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in structured discriminative learning. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 89–95, New York City, USA. Association for Computational Linguistics.

V. Sze, Y. Chen, T. Yang, and J. S. Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329.

Robert Williamson and Aditya Menon. 2019. Fairness risk measures. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6786–6797. PMLR.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *International Conference on Learning Representations*.

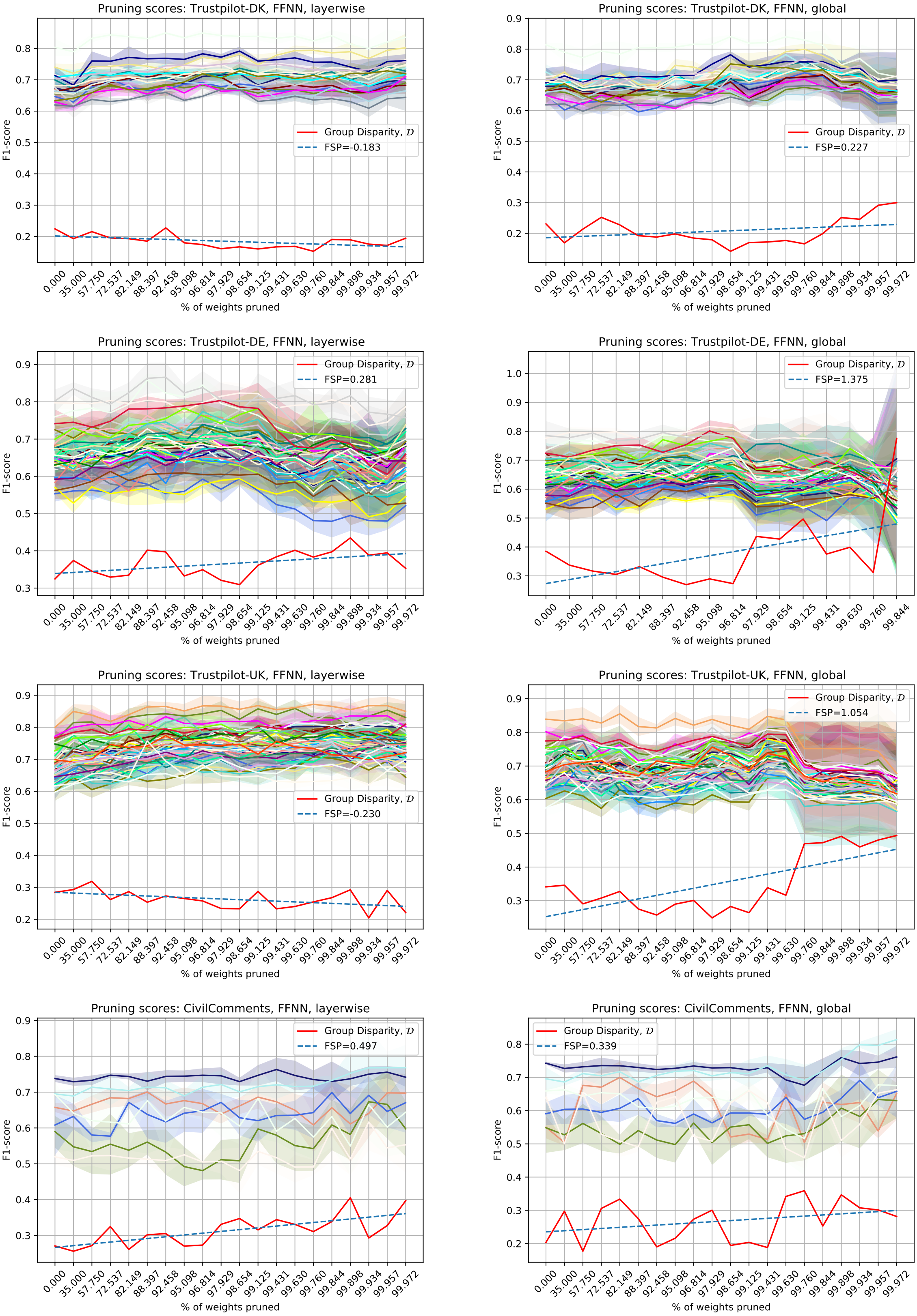

Figure 4: Macro-averaged performance of our feed-forward networks as a function of pruning ratio. The hard line represents the average demographic score over 5 individual runs and the shaded area represents the standard deviation. Fairness Sensitivity as Pruning (FSP) correspond to the gradient of the linear fit to the min-max differences across individual runs.

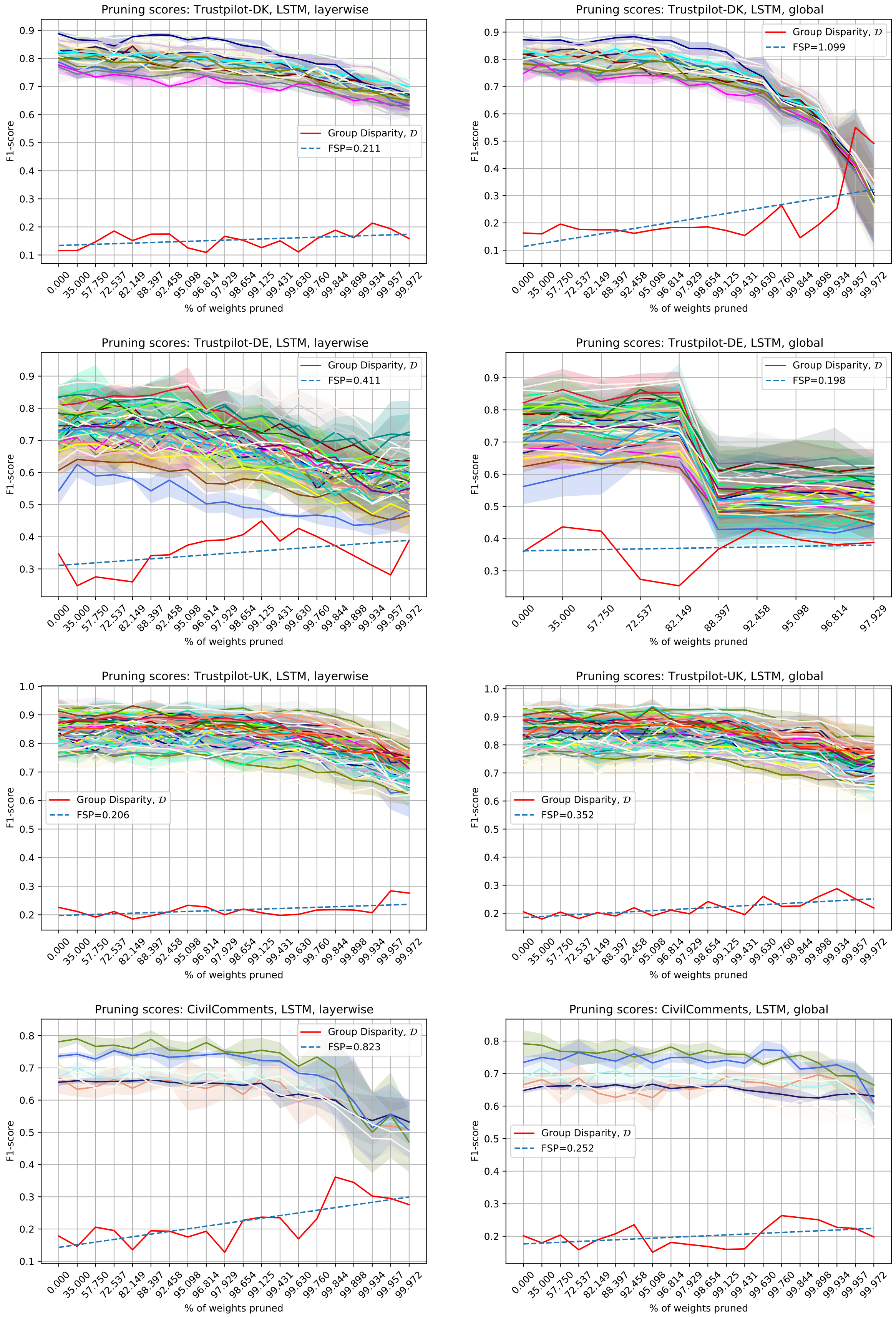

Figure 5: Macro-averaged performance of our LSTMs as a function of pruning ratio. The hard line represents the average demographic score over 5 individual runs and the shaded area represents the standard deviation. Fairness Sensitivity as Pruning (FSP) correspond to the gradient of the linear fit to the min-max differences across individual runs.

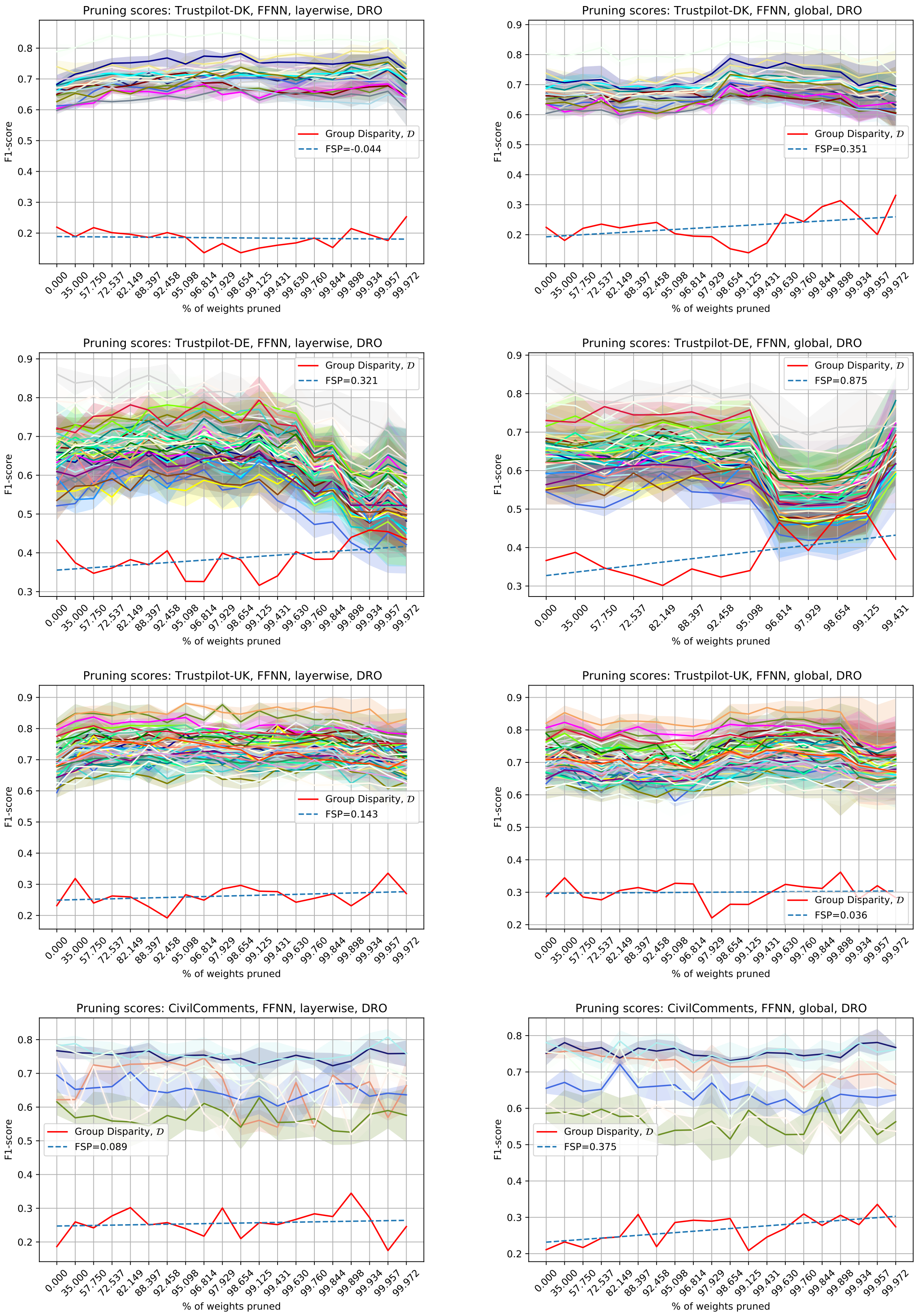
Pruning scores: Trustpilot-DK, FFNN, layerwise, DRO

Pruning scores: Trustpilot-DK, FFNN, global, DRO

Pruning scores: Trustpilot-DE, FFNN, layerwise, DRO

Pruning scores: Trustpilot-DE, FFNN, global, DRO

Pruning scores: Trustpilot-UK, FFNN, layerwise, DRO

Pruning scores: Trustpilot-UK, FFNN, global, DRO

Pruning scores: CivilComments, FFNN, layerwise, DRO

Pruning scores: CivilComments, FFNN, global, DRO

Figure 6: Macro-averaged performance of our layer-wise and globally pruned feed-forward networks trained with DRO as a function of pruning ratio. The hard line represents the average demographic score over 5 individual runs and the shaded area represents the standard deviation. Fairness Sensitivity as Pruning (FSP) correspond to the gradient of the linear fit to the min-max differences across individual runs.

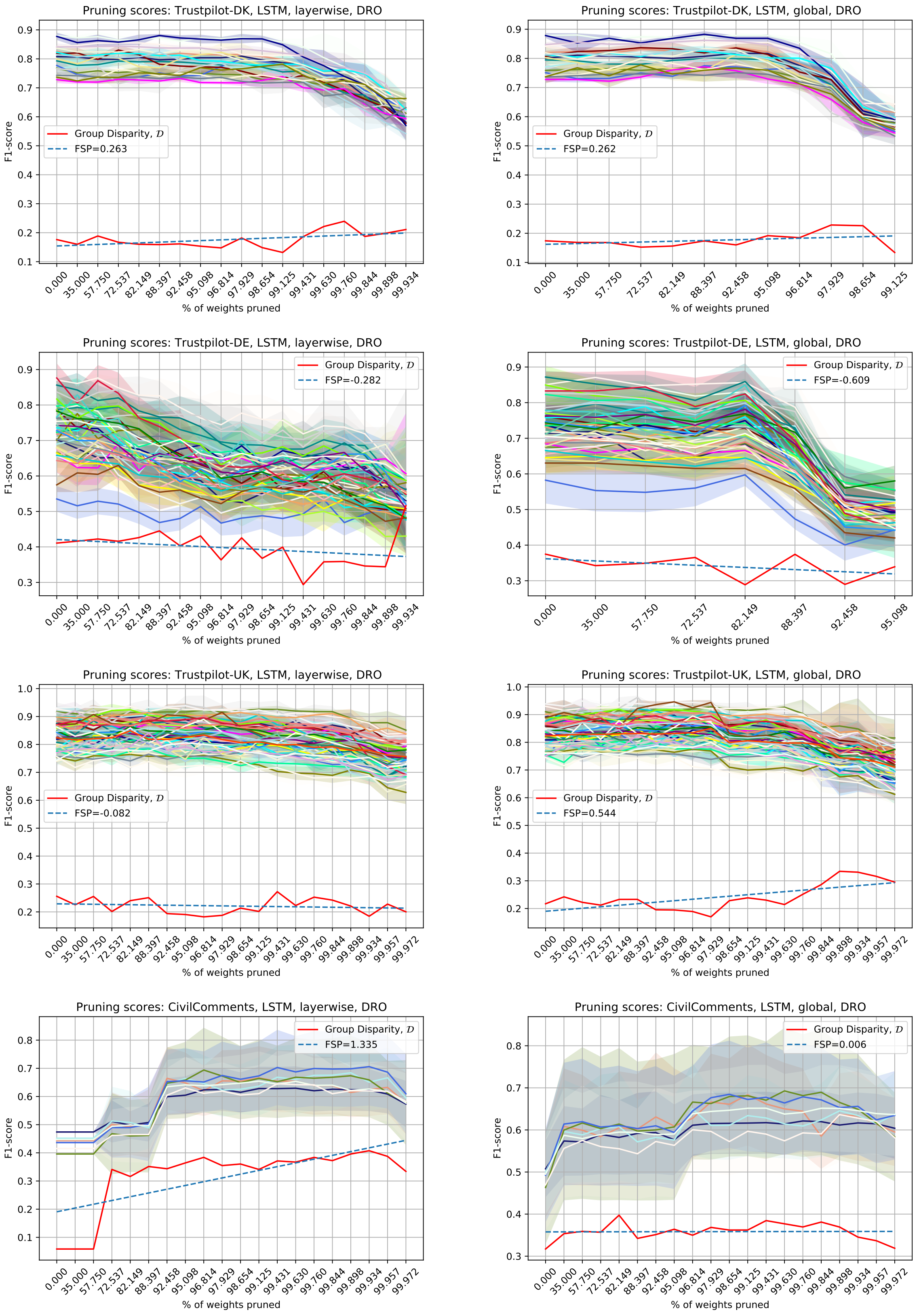

Figure 7: Macro-averaged performance of our layer-wise and globally pruned LSTM networks trained with DRO as a function of pruning ratio. The hard line represents the average demographic score over 5 individual runs and the shaded area represents the standard deviation. Fairness Sensitivity as Pruning (FSP) correspond to the gradient of the linear fit to the min-max differences across individual runs.

SSMix: Saliency-Based Span Mixup for Text Classification

Soyoung Yoon[1,2*†] **Gyuwan Kim**[1*] **Kyumin Park**[2]
[1]Clova AI, Naver Corp. [2]KAIST
`{soyoungyoon,pkm9403}@kaist.ac.kr, gyuwan.kim@navercorp.com`

Abstract

Data augmentation with mixup has shown to be effective on various computer vision tasks. Despite its great success, there has been a hurdle to apply mixup to NLP tasks since text consists of discrete tokens with variable length. In this work, we propose *SSMix*, a novel mixup method where the operation is performed on input text rather than on hidden vectors like previous approaches. *SSMix* synthesizes a sentence while preserving the locality of two original texts by span-based mixing and keeping more tokens related to the prediction relying on saliency information. With extensive experiments, we empirically validate that our method outperforms hidden-level mixup methods on a wide range of text classification benchmarks, including textual entailment, sentiment classification, and question-type classification. Our code is available at https://github.com/clovaai/ssmix.

1 Introduction

Data augmentation gains popularity in natural language processing (NLP) (Feng et al., 2021) due to the expensive cost of data collection. Some of them are based on simple rules (Wei and Zou, 2019) and models (Edunov et al., 2018; Ng et al., 2020) to generate similar text. Augmented samples are trained jointly with original samples by a standard way or advanced training methods (Zhu et al., 2019; Park et al., 2021). On the other hand, mixup (Zhang et al., 2018) interpolates input texts and labels for the augmentation.

Training with mixup and its variants become a popular regularization method in computer vision to improve the generalization of neural networks. Mixup approaches are categorized into input-level mixup (Yun et al., 2019; Kim et al., 2020; Walawalkar et al., 2020; Uddin et al., 2021) and hidden-level mixup (Verma et al., 2019) depending on the location of the mix operation. Input-level mixup is a more prevalent approach than hidden-level mixup because of its simplicity and the ability to capture locality, leading to better accuracy.

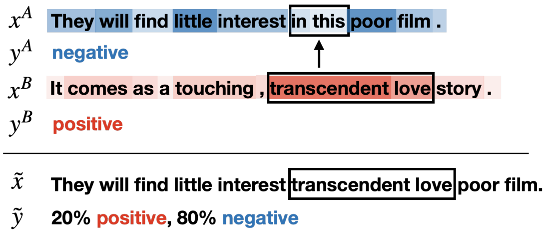
x^A They will find little interest in this poor film .
y^A negative
x^B It comes as a touching , transcendent love story .
y^B positive
$\tilde{x}$ They will find little interest transcendent love poor film.
$\tilde{y}$ 20% positive, 80% negative

Figure 1: Illustration of *SSMix*. Two data samples x^A and x^B are labeled negative and positive respectively for sentiment classification task. For each token, saliency maps are visualized where darker concentration of colors mean higher contribution to corresponding label. We select the least salient span from x^A and replace it with the most salient span from x^B. The output results in $\tilde{x} = mixup(x^A, x^B)$. We also assign $\tilde{y}$ by the mixup ratio λ. In this example, λ is set to 0.2 as the span length is 2 out of 10.

Applying mixup in NLP is more challenging than in computer vision because of the discrete nature of text data and variable sequence lengths. Therefore, most previous attempts on mixup for texts (Guo et al., 2019; Chen et al., 2020) apply mixup on hidden vectors like embeddings or intermediate representations. However, input-level mixup might have an advantage over hidden-level mixup with a similar intuition from computer vision. This motivation encourages us to examine input-level mixup approaches for text data.

In this work, we propose *SSMix* (Fig 1), a novel input-level spanwise mixup method considering the saliency of spans. First, we conduct a mixup by replacing a span of contiguous tokens with a span in another text, which is inspired from CutMix

---

[*] Equal contribution.

[†] Work done during the internship at Clova AI.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3225–3234
August 1–6, 2021.

(Yun et al., 2019), to preserves the locality of two source texts in the mixed text. Second, we select a span to be replaced and to replace based on saliency information to make the mixed text contain tokens more related to output prediction, which may be semantically important. Our input-level method is different from hidden level mixup methods in that while current hidden level mixup methods linear interpolate original hidden vectors, our method mix tokens on the input level, resulting in a *nonlinear* output. Also, we utilize saliency values to select span from each sentence and discretely define the length of span and mixup ratio, which is outside the hidden level.

SSMix has empirically proven effective through extensive experiments on a wide range of text classification benchmarks. Especially, we prove that input-level mixup methods generally outperform hidden-level methods. We also show the importance of using saliency information and restricting token selection in span-level when conducting our method via ablation study.

2 *SSMix*

We propose *SSMix* to synthesize a new text $\tilde{x}$ by replacing a span x_S^A from one text x^A into another span x_S^B from another text x^B based on saliency information. Also, we have to set a new label $\tilde{y}$ for $\tilde{x}$ using y^A and y^B which are one-hot labels corresponding to x^A and x^B, respectively. Consequently, we can additionally use this generated virtual sample $(\tilde{x}, \tilde{y})$ for training.

Saliency Saliency measures how each portion of data (in this case, tokens) affects the final prediction. Gradient-based methods (Simonyan et al., 2013; Li et al., 2016) are widely used for the saliency computation. We compute the gradient of classification loss $\mathcal{L}$ with respect to input embedding e, and use its magnitude as the saliency: i.e., $s = \|\partial\mathcal{L}/\partial e\|_2$. We apply the L2 norm to obtain the magnitude of a gradient vector, which becomes a saliency of each token similar to PuzzleMix (Kim et al., 2020).

Mixing text Text data x^A and x^B are discrete token sequences. Using saliency scores as explained earlier, we can find the least salient span in x^A with a length l_A as x_S^A and the most salient span in x^A with a length l_B as x_S^B. We set $l_A = l_B = max(min([\lambda_0|x^A|], |x^B|), 1)$ given a prior mixup ratio λ_0. Then, final $\tilde{x}$ becomes the concatenation of $(x_L^A; x_S^B; x_R^A)$ where x_L^A and x_R^A are tokens located to the left and the right side of x_S^A respectively in the original text x^A.

Algorithm 1 Mixup loss calculation

```
procedure SSMix_Loss($x^A, x^B, y^A, y^B, \lambda$)
    $\tilde{x} \leftarrow SSMix(x^A, x^B)$
    $logit \leftarrow model(\tilde{x})$
    $loss^A \leftarrow CrossEntropy(logit, y^A)$
    $loss^B \leftarrow CrossEntropy(logit, y^B)$
    $total\_loss \leftarrow loss^A * \lambda + loss^B * (1 - \lambda)$
return $total\_loss$
end procedure
```

Same span length We set the length of the original (l_A) and replaced (l_B) span to be the same, since allowing different length of spans would result in redundant and ambiguous mixup variations. Also, calculating the mixup ratio between different span length would be too complex. This same-size replacement strategy is also adopted in (Yun et al., 2019) and (Uddin et al., 2021). In situations where span length is the same, our method maximizes the effect of saliency. Since *SSMix* doesn't restrict the position of tokens, we can pick the *most* salient span and replace it with *least* salient span on the other text.

Mixing label We set mixup ratio λ for label as $\lambda = |x_S^B|/|\tilde{x}|$. Since λ is recalculated by counting the number of tokens in the span, it may differ from λ_0. We set the label of $\tilde{x}$ to $\tilde{y} = (1 - \lambda)y^A + \lambda y^B$. Algorithm 1 shows how we utilize the original sample pairs to compute the mixup loss for augmented samples. We calculate the cross-entropy loss of the augmented output logit with respect to the original target label of each sample and combine them by weighted sum, which is similar to the original implementation of (Zhang et al., 2018).[1] Therefore, applying *SSMix* is independent of the total number of labels of the classification dataset. On any dataset, output label ratio is calculated by linear combination of *two* original labels.

Paired sentence tasks For tasks requiring a pair of texts as an input such as textual entailment and similarity classification, we conduct mixup in a pairwise manner and calculate the mixup ratio by aggregating token counts in each mixup result. Denoting $x^A = (p^A, q^A)$, $x^B = (p^B, q^B)$, and

---

[1] https://github.com/hongyi-zhang/mixup/blob/master/cifar/utils.pyL34

Dataset	Task	# Label	Size
SST-2	Sentiment	2	67k / 1.8k
QQP	Paraphrase	2	364k / 391k
MNLI	NLI	3	393k / 20k
QNLI	QA/NLI	3	105k / 5.4k
RTE	NLI	2	2.5k / 3k
MRPC	Paraphrase	2	3.7k / 1.7k
TREC-coarse	Classification	6	5.5k / 500
TREC-fine	Classification	47	5.5k / 500
ANLI	NLI	3	162.8k / 3.2k / 3.2k

Table 1: Dataset name, task, number of total labels, and dataset size of datasets we used as benchmark. Task column describes the objective of each dataset. ANLI dataset shows aggregated dataset statistics among different rounds. GLUE tasks report the size as (train / validation) format, TREC reports (train / test) and ANLI reports (train / validation / test).

$\tilde{x} = (\tilde{p}, \tilde{q})$, we define mixup of paired sentence data as $\tilde{x} = (mixup(p^A, p^B), mixup(q^A, q^B))$. Here, we set the mixup ratio on paired sentence tasks as $\lambda = (|p_S| + |q_S|)/(|\tilde{p}| + |\tilde{q}|)$, where p_S and q_S are replacing spans of independent mixup operations. Illustation is available in Appendix B.3.

3 Experimental Setup

3.1 Dataset

As listed in table 1, to evaluate the effectiveness of *SSMix*, we perform experiments on various text classfication benchmarks: six datasets in GLUE benchmark (Wang et al., 2018), TREC (Li and Roth, 2002; Hovy et al., 2001), and ANLI (Nie et al., 2020). Two of them are single sentence classification tasks, and six of them are sentence pair classification tasks. All datasets are extracted from HuggingFace datasets library.[2]

For GLUE, we use SST-2 (Socher et al., 2013), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Bentivogli et al., 2009), MRPC (Dolan and Brockett, 2005), and QQP[3]. Among GLUE, we leave out datasets that were not evaluated by accuracy, along with WNLI, because the size is too small to show any general trend of effectiveness.

TREC is a commonly used dataset to evaluate mixup methods in sentence classification (Guo et al., 2019; Thulasidasan et al., 2019). We use two different versions of TREC (coarse, fine) that have different levels of label number to test the dependency of mixup effectiveness on the number of class labels. In addition, we use ANLI to see how mixup can help to improve model robustness. For training ANLI, we concatenate all training data from different rounds and use them to train the model.

3.2 Baseline

We compare *SSMix* with three baselines: (1) standard training without mixup, (2) EmbedMix, and (3) TMix. EmbedMix apply mixup on the embedding layer, which is similar to the wordMixup in Guo et al. (2019) except their experiments are performed with LSTM or CNN architecture. TMix, borrowed from Chen et al. (2020), interpolates hidden states of two different inputs at a particular encoder layer and forward the combined hidden states to the remaining layers. For EmbedMix and TMix, we follow the best settings stated in the original papers: mixup ratio is set by $\lambda' \sim Beta(\alpha, \alpha)$, $\lambda = max(\lambda', 1 - \lambda')$ with $\alpha = 0.2$. During the training with TMix, we randomly sample the mixup layer from $[7, 9, 12]$.

3.3 Ablation study

To investigate how much (1) considering saliency and (2) restricting mixup operation on the span-level individually benefit our proposed method, we conduct an ablation study. We implement *SSMix* without considering saliency information (*SSMix* - saliency) where the spans are randomly selected, and additionally without the span-level restriction (*SSMix* - saliency - span). For *SSMix* - saliency - span, we randomly sample tokens from x^B, which need not be a contiguous span and are conducted on a per-token basis. Then, we replace tokens accordingly with the position of the token be preserved, meaning that the second token from x^A is replaced with the second token from x^B, and so on. For all ablation studies, the lambda values were set to 0.1 to compare methods with the same setting as *SSMix*. Detailed implementation and illustration of ablation methods and comparison with simple word dropout methods are described in Appendix B.

3.4 Training Details

Among the entire experiment, we use sequence classification task with the pre-trained BERT-base model having 110M parameters from Hugging-

---

[2] https://github.com/huggingface/datasets

[3] https://www.quora.com/First-Quora-Dataset-Release-Question-Pairs

Face Transformers library.[4] We perform all experiments with five different seeds (0 to 4) on a single NVIDIA P40 GPU and report the average score. We set a maximum sequence length of 128, batch size of 32, with AdamW optimizer with eps of 1e-8 and weight decay of 1e-4. We use a linear scheduler with a warmup for 10% of the total training step. We update the best checkpoint by measuring validation accuracy on every 500 steps. For datasets that have less than 500 steps per epoch, we update and validate every epoch.

Considering our objective of enhancing performance through mixup, we conduct training in two steps. We first train without mixup with a learning rate of 5e-5 for three epochs, and then train with mixup starting from previous training's best checkpoint, with a learning rate of 1e-5 for five epochs. This two-step training, which also utilized by Zhang et al. (2018), speeds up the model convergence. We report the best accuracy among both training with and without mixup. For the ANLI task, we select the best checkpoint for training without mixup separately for each round, then conduct training with mixup and report the best accuracy of each round's evaluation dataset.

For each iteration, we split the batch into two smaller batches with the same size, A and B. Since mixup operation in *SSMix* is not symmetric, we conduct mixup back-and-forth so that mixup performance is evaluated regardless of the data position in batch. To prevent the training data distribution getting too far from the original data distribution, we train with and without mixup together as He et al. (2019). As a result, we forward each step with average loss from A, B, $mixup(A, B)$, and $mixup(B, A)$.

We leave out tokens specific to transformer architecture (e.g., $[CLS]$, $[SEP]$) when conducting a mixup to preserve special signs. As stated by Zhang et al. (2018), giving too high values for mixup ratio may lead to underfitting, while giving λ close to 0 leads to the same effect of giving non-augmented original data. From our experiments, we found out that augmentation with prior ratio $\lambda_0 = 0.1$ is the optimal hyperparameter.

In terms of computation time, *SSMix* takes about twice the training time compared with other mixup methods since we need an additional forward and backward step to compute the saliency of tokens. Among hidden-level mixup methods, TMix takes a slightly longer time to train than EmbedMix.

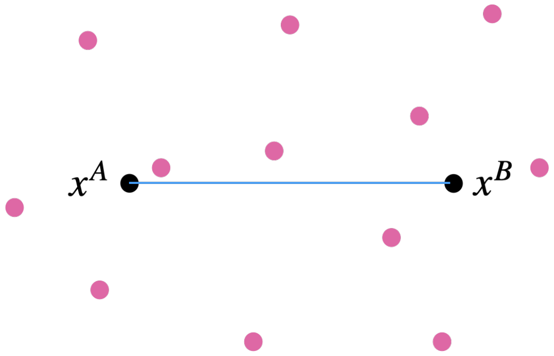

Figure 2: Visualization of original data and synthesized data by hidden-level mixup (EmbedMix or TMix) and *SSMix* in the hidden space. Black dots indicate the original data, x^A and x^B. For hidden-level mixup, synthetic data ($\tilde{x}$) are created only along the line (blue) connecting two points, since it is a linear combination within the hidden space. However, *SSMix* explore larger synthetic sample space for $\tilde{x}$, since it consists of a discrete combination within the *input* space. Synthetic data for *SSMix* are illustrated in pink dots.

4 Results and Discussion

Table 2 illustrates our results. We investigate the effectiveness of *SSMix* compared with hidden layer mixup methods on the aspect of dataset size, number of class labels, and paired sentence tasks.

Dataset size Compared with hidden-level mixup methods, *SSMix* fully demonstrate its effectiveness on datasets having a sufficient amount of data. Since *SSMix* is a discrete combination rather than a linear combination of two data samples, it creates data samples on a synthetic space in a larger range than hidden-level mixup (Fig. 2). We hypothesize that a large amount of data help better representation in synthetic space.

The number of class labels *SSMix* is especially effective for multiple class label datasets (TREC, ANLI, MNLI, QNLI). Accordingly, the accuracy gain of *SSMix* from the training without mixup is much higher on TREC-fine (47 labels) than TREC-coarse (6 labels), with +3.56 and +0.52, respectively. We hypothesize that this result originates from the mixup characteristic that benefits more from cross-label mixup than mixup with the same label, as stated at Zhang et al. (2018).[5] Since datasets with multiple total class labels increase the

---

[4] https://github.com/huggingface/transformers

[5] Zhang et al. (2018) states that mixing random pairs from all classes (per-batch basis) has the strongest regularization effect compared with mixup by per-class (same class) basis.

Model	GLUE						TREC		ANLI		
	SST-2	QQP	MNLI	QNLI	RTE	MRPC	coarse	fine	R1	R2	R3
No mixup	92.96	91.32	84.27	91.28	65.56	86.37	97.08	86.68	56.40 57.16	47.10 47.36	47.62 48.00
EmbedMix	93.03	91.36	84.35	91.43	**67.73**	**86.72**	97.44	90.04	56.78 57.16	47.84 47.42	47.67 48.00
TMix	93.03	91.34	84.33	91.40	66.86	86.42	97.52	90.16	56.68 57.28	47.58 47.90	47.78 **48.42**
SSMix	93.10	**91.43**	**84.54**	**91.54**	67.22	86.57	**97.60**	**90.24**	**57.26** **57.34**	**48.36** **48.06**	47.78 48.00
SSMix - saliency	93.12	91.32	84.48	91.29	67.00	86.42	97.44	89.56	57.04 57.16	48.22 47.94	**47.95** 48.07
SSMix - saliency - span	**93.14**	91.32	84.54	91.45	66.93	86.37	97.40	89.20	56.74 57.20	47.52 47.90	47.77 48.00

Table 2: Experimental results of comparison with baselines and ablation study. All values are average accuracy (%) of five runs with different seeds. MNLI indicates MNLI-mismatched dev set accuracy. We report validation accuracy for GLUE, test accuracy for TREC, and valid (upper) / test (lower) accuracy for ANLI. We report variance on Appendix. A.

possibility of being selected cross-label in a random sampling of mixup sources, we assert mixup performance increases in such datasets.

Paired sentence tasks *SSMix* have a competitive advantage on paired sentence tasks, such as textual entailment or similarity classification. We suspect this accuracy gain originates from consideration of individual tokens. Existing methods (hidden-level mixup) apply mixup on the hidden layer, without consideration of special tokens, i.e., $[SEP]$, $[CLS]$. These methods may lose information about the start of the sentence or appropriate separation of pair of sentences. In contrast, *SSMix* can consider the individual token property when applying mixup. Here, our mixup strategy on paired data (Section 2) preserves the property of $[SEP]$, which is not guaranteed by hidden mixup.

Ablation Study The results of *SSMix* and its variants demonstrate that the performance improves as we add span constraint and saliency information. Adding span constraint in the mixup operation benefit from better localizable ability, and most salient spans have more relationship to corresponding labels while discarding least salient spans have a higher probability that those spans are not semantically important with respect to the original labels. Among those two, introducing saliency information contributes to accuracy relatively more than the span constraint.

5 Conclusion

We present *SSMix*, a novel and simple input-level mixup method for text data that improves regularization ability leading to better performance in text classification. *SSMix* preserves the locality of mixing texts by replacing in span-level and keep most discriminative tokens in the mixed text using saliency score. Throughout the experiment, we show that our method improves performance in various types of text classification tasks. For future work, we plan to apply *SSMix* on a broader range of tasks, including generation or different scenarios like semi-supervised learning.

Acknowledgments

The authors would like to thank Clova AI members for proofreading this manuscript and the anonymous reviewers for their constructive feedback. We use Naver Smart Machine Learning (Sung et al., 2017; Kim et al., 2018) platform for the experiments. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Zhuoxun He, Lingxi Xie, Xin Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. 2019. Data augmentation revisited: Rethinking the distribution gap between clean and augmented data.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. 2018. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. *arXiv preprint arXiv:2009.10195*.

Yixin Nie, Adina Williams, Emily Dinan, Jason Bansal, Mohitand Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jungsoo Park, Gyuwan Kim, and Jaewoo Kang. 2021. Consistency training with virtual adversarial discrete perturbation. *arXiv preprint arXiv:2104.07284*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. 2017. Nsml: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 13888–13899.

A F M Shahab Uddin, Mst. Sirazam Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. 2021. Saliencymix: A saliency guided data augmentation strategy for better regularization. In *International Conference on Learning Representations*.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Devesh Walawalkar, Zhiqiang Shen, Zechun Liu, and Marios Savvides. 2020. Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*.

A Accuracy Variance

Model	GLUE						TREC		ANLI		
	SST-2	QQP	MNLI	QNLI	RTE	MRPC	Coarse	Fine	R1	R2	R3
No mixup	0.04	0.04	0.12	0.05	3.89	1.73	0.17	2.21	1.21	0.16	0.73
									0.24	1.26	0.84
EmbedMix	0.02	0.03	0.14	0.04	3.89	1.39	0.09	0.31	1.38	0.46	0.75
									0.24	1.18	0.84
TMix	0.04	0.04	0.09	0.03	1.85	1.55	0.05	0.63	1.44	0.33	0.73
									0.25	0.75	1.28
SSMix	0.03	0.07	0.07	0.03	2.57	1.15	0.03	0.49	1.56	0.27	0.73
									0.25	0.46	0.84
SSMix - saliency	0.02	0.04	0.11	0.04	2.06	1.55	0.09	0.69	1.33	0.18	0.62
									0.24	1.99	0.80
SSMix - saliency - span	0.00	0.04	0.09	0.03	1.86	1.73	0.08	0.14	2.01	0.11	0.68
									0.28	0.45	0.84

Table A.1: Standard deviation results, corresponding with the average of our experiments. The deviation is conducted by 5 runs with different seeds.

We also report accuracy variance among the five seeds for each experiment (Table. A.1).

B Ablation

Fig. 3 and Fig. 4 shows the illustration of different variants of *SSMix* and random UNK replacement with $\lambda = 0.2$. Fig. 5 shows the illustration of getting the augmented output with lambda calculation by *SSMix* for paired sentence tasks. The saliency maps are visualized where darker concentration of colors mean higher contribution to corresponding label.

B.1 Variants of *SSMix*

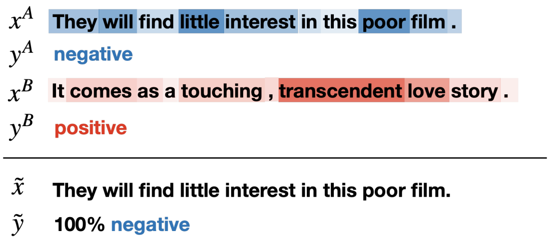

(a) Normal training without mixup

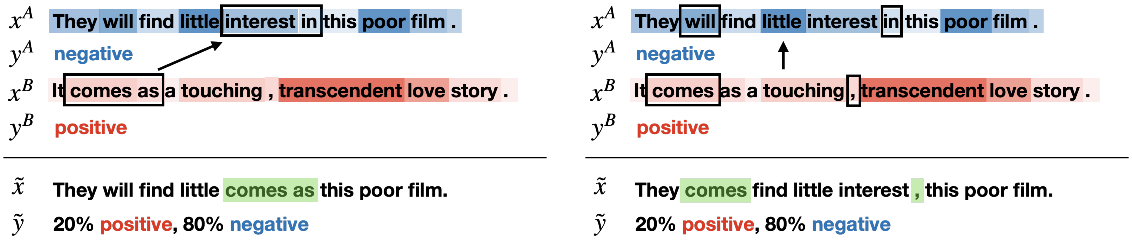

(b) *SSMix* - saliency (c) *SSMix* - saliency - span

Figure 3: Illustration of normal training and variants of *SSMix*

Here, we describe in detail how we implement *SSMix* without saliency (Figure. 3 (b)) and *SSMix* without saliency and span restriction (Figure. 3 (c)).

Model	GLUE						TREC		ANLI		
	SST-2	QQP	MNLI	QNLI	RTE	MRPC	coarse	fine	R1	R2	R3
No mixup	92.96	91.32	84.27	91.28	65.56	86.37	97.08	86.68	56.40 57.16	47.10 47.36	47.62 48.00
Random UNK replacement	93.10	91.33	84.46	91.45	66.86	**86.62**	97.44	89.24	56.98 57.26	47.86 **48.36**	**47.98** **48.32**
SSMix	93.10	**91.43**	**84.54**	**91.54**	**67.22**	86.57	**97.60**	**90.24**	**57.26** **57.34**	**48.36** 48.06	47.78 48.00
SSMix - saliency	93.12	91.32	84.48	91.29	67.00	86.42	97.44	89.56	57.04 57.16	48.22 47.94	47.95 48.07
SSMix - saliency - span	**93.14**	91.32	84.54	91.45	66.93	86.37	97.40	89.20	56.74 57.20	47.52 47.90	47.77 48.00

Table B.1: Accuracy (%) comparison with simple data augmentation method(random UNK replacement) and input mixup methods. The results are average of five runs with different seeds. Results show that our input level mixup methods are generally competitive with simple word dropout methods.

At normal training, only two real data samples (x^A and x^B) are used to train the model. For Figure. 3 (b), we *randomly* select each span from x^A and x^B. Then, we replace x^B to x^A to make a new data $\tilde{x}$. For Figure. 3 (c), input level mixup is conducted on a per-token basis. After calculaton of l given the prior mixup ratio, we randomly sample tokens from x^A. The tokens need not be a contiguous span. Then, we replace tokens accordingly with the position of the token be preserved, meaning that the second token from x^A is replaced with second token from x^B, the sixth token from x^A is replaced with sixth token from x^B (by the illustration example), and so on.

B.2 Comparison with other simple augmentation methods

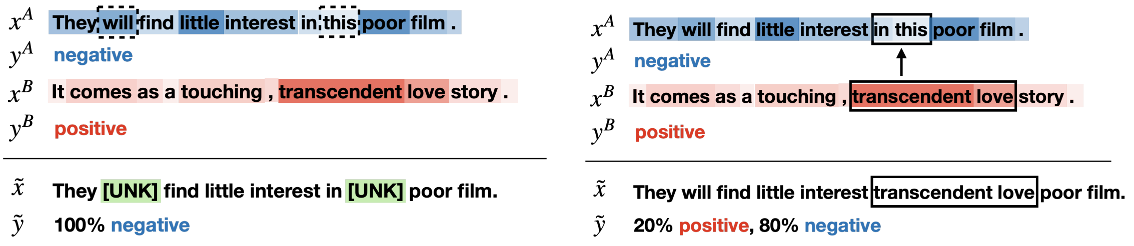

(a) Random `[UNK]` replacement (b) *SSMix*

Figure 4: Comparison of our methods with word dropout

We also compare *SSMix* with simple word dropout methods, which may seem similar in the perspective that they create noisy sentences. The difference is whether label mixup is performed. Illustration of the implementation of random [UNK] replacement is available at Fig. 4. Random UNK replacement is similar to word dropout. We don't use x^B when making synthetic samples ($l = 0$). Instead, we randomly sample a set of tokens from x^A and replace each token in that span with [UNK]. The process is similar to Figure. 3 (c), except that the selected tokens at x^A are replaced into [UNK]. Another difference is that the output label ($\tilde{y}$) completely follow the origin (y^A) and no label mixup is performed. The illustration is available at 3.

We evaluate the random [UNK] replacement method on all dataset with *SSMix* and variants of *SSMix* at ablation study. By the experiment results at Table B.1, we show that input level mixup methods generally outperform simple regularization methods. This means that datasets synthesized from *SSMix* and the according target vectors have more gain on the generalization ability than word dropout.

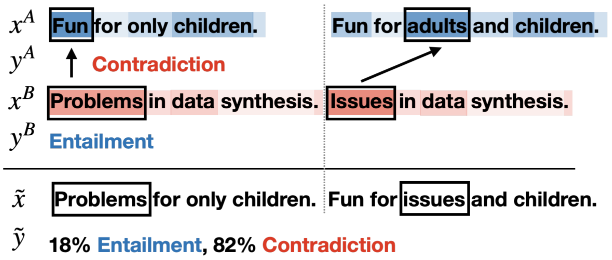

Figure 5: Illustration of applying SSMix to make $\tilde{x}$ for paired sentence, in particular NLI tasks, which classifies whether the relation of sentence pairs is entailment, neutral, or contradiction. Mixup is conducted individually, sentence by sentence.

B.3 Illustration of SSMix on paired sentence tasks

Fig. 5 shows the illustration of example for paired sentence. Here, "Fun for only children." and "Fun for adults and children." correspond to p^A and q^A, "Problems in data synthesis." and "Issues in data synthesis." correspond to p^B and q^B, and "Problems for only children.", "Fun for issues and children." correspond to p and q, respectively. λ is calculated as : $\lambda = (|p_S| + |q_S|)/(|\tilde{p}| + |\tilde{q}|) = (1+1)/(5+6) = 2/11 \approx 0.18$.

Detecting Bot-Generated Text by Characterizing Linguistic Accommodation in Human-Bot Interactions

Paras Bhatt and **Anthony Rios**
Department of Information Systems and Cyber Security
University of Texas at San Antonio
`{Paras.Bhatt, Anthony.Rios}@utsa.edu`

Abstract

Language generation models' democratization benefits many domains, from answering health-related questions to enhancing education by providing AI-driven tutoring services. However, language generation models' democratization also makes it easier to generate human-like text at-scale for nefarious activities, from spreading misinformation to targeting specific groups with hate speech. Thus, it is essential to understand how people interact with bots and develop methods to detect bot-generated text. This paper shows that bot-generated text detection methods are more robust across datasets and models if we use information about how people respond to it rather than using the bot's text directly. We also analyze linguistic alignment, providing insight into differences between human-human and human-bot conversations.

1 Introduction

Bots are useful in a wide variety of applications areas including business (Kaczorowska-Spychalska, 2019), education (Kerlyl et al., 2006), and health (Yadav et al., 2019; Liednikova et al., 2020). For instance, Yadav et al. (2019) studied the use of chatbots as a drop-in first-point-of-contact for women in India seeking breastfeeding information. Similarly, researchers have studied bots to answer COVID-19-related questions (Oniani and Wang, 2020) and screen individuals for risks of contracting the virus (Martin et al., 2020). Overall, the wide availability of software packages, tools, and pre-trained models has democratized the creation of bots.

Even with the increasing interest in bots for social good (e.g., COVID-related chatbots), there is still a concern regarding their abuse to spread misinformation, be used for targeted discrimination, deceive users, and perform fraud (Daniel et al., 2019). Given the potential good and harm bots can create, it is essential to study how the bots *should* act and how people *do* interact with such bots for specific applications.

Automatically identifying bots online is well studied (Garcia-Silva et al., 2019; Herzig et al., 2019; Kosmajac and Keselj, 2019; Ippolito et al., 2020; Jawahar et al., 2020). Bot detection methods rely on two forms of information (Orabi et al., 2020): behavior and content. Behavior relates to measuring how often bots post, the time posts are created, and conversational network structures (Beskow and Carley, 2018). Content involves using the bot's text directly. Focusing on text suffers from generalization issues, making it challenging to detect bots that discuss different topics or operate in different domains. Likewise, behavioral approaches assume that bots will behave differently than people at a superficial level, such as posting more often than humans. Even in the network analysis of conversation structures (Beskow and Carley, 2018), many of the human-bot interactions on social media are not human-like interactions. For instance, it is understandable that humans will interact with a bot that converts pounds to kilograms differently than other humans. Given bots' current use-cases ranging from counselors to healthcare information providers, it is vital to understand how humans and bots interact beyond trivial applications.

Before looking at how humans interact with bots, it is essential to understand how humans interact. This paper focuses on Communication Accommodation Theory (CAT) and general language use to analyze interactions. CAT is used to study language use in various domains to understand human behavior (Giles et al., 1973; Tausczik and Pennebaker, 2010). More specifically, we study linguistic accommodation—where speakers come to talk more (or less) similarly as they interact—

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3235–3247
August 1–6, 2021.

which has been analyzed in mental health support on Reddit, showing a positive link with informational and emotional support (Sharma and De Choudhury, 2018). The alignment of two people in terms of linguistic style has positively predicted successful outcomes of negotiations (Taylor and Thomas, 2008), and doctors are recommended to accommodate perspective to improve patient care (Kline and Ceropski, 1984; Wood, 2019). The way people use and accommodate pronouns can indicate power, where high-status individuals use "I" less and "you/we" words more than low-status individuals (Kacewicz et al., 2014).

Linguistic accommodation in human-bot interactions has been studied in system design, showcasing that there is a strong link between user experience and language style of the system (Chaves et al., 2019; Chaves, 2020; Thomas et al., 2020). Recent research has also shown a link between language style in chatbots and user engagement for e-commerce (Elsholz et al., 2019). Our focus is to understand real long open-domain human-bot interactions better. Our study can also influence how bots are detected on social media and provide a better understanding of how human-human interactions differ from human-bot interactions is essential for bot development. For instance, when should bots accommodate towards the user? Can we tell how well a system performs for a specific task based on whether the user accommodates the bot? If human-bot conversations are linguistically different than human-human interactions, research from human-human studies may not generalize to human-bot interactions.

Toward addressing the potential societal impacts of open-domain bots, this paper addresses the following research questions (RQs) in the context of bot Detection:

RQ1. How do humans and bots align in human-bot interactions?

RQ2. How does the alignment in human-bot interactions compare to the alignment in human-human interactions?

RQ3. Are differences in alignment between human-human and human-bot interactions similar across domains and language generation learning methods?

Overall, we analyze whether human users' language changes stylistically with bots as compared to human-human conversations. Furthermore, we show that small differences in language style provides robust information, compared to the bots' language patterns, to accurately detect bots.

2 Related Work

Bot Detection. Detecting bots in the wild is a widely studied problem (Garcia-Silva et al., 2019; Herzig et al., 2019; Kosmajac and Keselj, 2019; Ippolito et al., 2020; Jawahar et al., 2020). bot detection methods rely on two forms of information (Orabi et al., 2020): Content and Behavior. Garcia-Silva et al. (2019) studied how pretrained language models perform for the task of bot detection. Kosmajac and Keselj (2019) developed "language-independent" stylistic features that measure language diversity to detect bots. Knauth (2019) empirically explored content and behavioral features for bot detection. Beskow and Carley (2018) analyzed human-bot interactions by developing methods to detect bots using network analysis of conversation structures. Much of the prior work has focused on detecting bots "in the wild.“ Thus, many of the interactions between bots and humans are superficial (e.g., receiving movie quotes from a bot). This paper differs from prior work in two ways. First, rather than using content from the bot or general behavioral information (e.g., post frequency), we try to detect bots by analyzing how humans respond to them (compared to human-human interactions). Second, rather than exploring bots in the wild, we explore two types of datasets: one where researchers instruct participants to interact with bots and humans in the same way, and another dataset where researchers instruct participants to converse with a bot.

Analyzing Human-Human Conversations. Linguistic alignment is known to interact with a wide array of social factors. For instance, the level of alignment of people's linguistic style in a conversation has been claimed to be affected by their relative social power (Gnisci, 2005; Xu et al., 2018; Danescu-Niculescu-Mizil et al., 2011). Cooperative decision-making tasks are positively related to the participants' linguistic convergence (Fusaroli et al., 2012; Kacewicz et al., 2014). Recently, Sharma and De Choudhury (2018) analyzed mental health support forums on Reddit, showing that linguistic accommodation is positively linked with informational and emotional support. Similarly, Taylor and Thomas (2008) analyzed negotiation outcomes and linguistic alignment, showing that alignment positively predicted successful negotiation outcomes. There is also the support of linguistic alignment by doctors to enhance patient care by improving trust and adherence to a treatment plan (Kline and Ceropski, 1984; Wood, 2019). Alignment is a powerful tool that can be used to improve understanding, trust, and potentially patient outcomes. Therefore, understanding *when* and *how* people should align with each other is an important area of research. Furthermore, it is essential to draw a line between engaging with a culture and language style and appropriating it, thereby potentially causing harm rather than building rapport with the partners in a conversation.

With the goal of understanding human-human interactions, there has also been progress in linguistic alignment measurement techniques (Niederhoffer and Pennebaker, 2002; Danescu-Niculescu-Mizil et al., 2011; Jones et al., 2014; Wang et al., 2014; Doyle and Frank, 2016; Shin and Doyle, 2018). Danescu-Niculescu-Mizil et al. (2011) presented an easy-to-compute expression that measures the increase in the conditional probability given that a conversational partner has used it. An issue with the method proposed by Danescu-Niculescu-Mizil et al. (2011) is that it assumes messages between two people have similar length. To overcome this limitation, Doyle and Frank (2016) introduced the Word-Based Hierarchical Alignment Model (WHAM), a hierarchical graphical model where the parameters are learned using Bayesian inference. Another method called the Simplified Word-Based Alignment Model (SWAM) was recently proposed by Shin and Doyle (2018). SWAM attempts to compare alignment between different groups when the alignment scores are assumed to differ substantially. Unfortunately, SWAM only estimates group-level alignment, not conversation-level between two specific users/bots. Linguistic alignment has also been shown to be predictive of specific tasks. In this paper, we use the method proposed by Danescu-Niculescu-Mizil et al. (2011) to estimate linguistic alignment. For instance, Niven and Kao (2019) use alignment features to predict discourse acts.

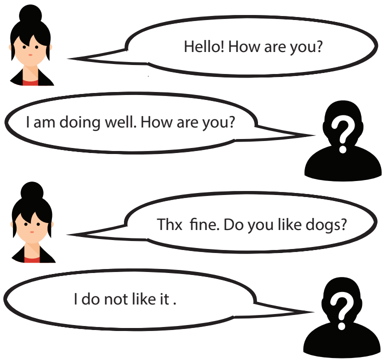

Figure 1: This figure depicts an potential conversation between a known human user and an unknown user. The unknown user may be a human or a bot.

Analyzing Human-Bot Conversations. Research studying the interaction between bots and humans has been explored from a wide array of perspectives. For example, systems that use emotionally expressive interjections ("wow", "ahem") in their text to speech responses can significantly improve the user experience (Cohn et al., 2019). Given the popularity of bots in application areas from business (Kaczorowska-Spychalska, 2019) to healthcare (Pieraccini et al., 2009), it is also important to understand how language generation style and alignment impacts their intended use. There has been a recent interest in analyzing accommodation and similar concepts in human-bot interactions. For instance, Ahn et al. (2020) show that humans will match *code-switching* patterns introduced by a chat system. Moreover, users tend to have positive reactions towards systems that code-switch. Ma and Lalor (2020) measure *lexical entrainment* between a specific Reddit bot and users. They find that sentiment of bot has a positive effect on the sentiment of the humans response. Furthermore, Ma and Lalor (2020) show that human responses tend to overlap with the bots original post. Compared to prior work studying accommodation-related aspects of human-bot interactions, this paper differs in two ways. First, just analyzing linguistic alignment, we ground our study of human-bot interactions in the real-world task of bot detection. Second, we analyze cross-domain generalization of accommodation patterns in human-bot interactions, where cross-domain includes types of bots (e.g., retrieval and transformer-based models), data collection procedures (e.g., sampling bias), and conversation topics.

3 Datasets

An overview of the task we introduce in this paper is shown in Figure 1. We formulate a bot de-

Dataset Name	# D	# u	Avg. u	Avg. W
ConvAI2 Data				
INTERMEDIATE	291	4317	14.83	6.66
TOLOKERS	3127	39155	12.52	7.05
VOLUNTEER	1111	14623	13.16	6.60
PERSONA-CHAT	18878	278478	14.75	1.85
Control Data				
IRIS	163	5687	34.89	5.46
TICKTOCK	206	5462	26.51	7.07
DailyDialog	13118	102980	7.85	13.08

Table 1: Summary of each dataset, including the number of dialogues (# D), average number of utterances per dialogue (Avg. u), average number of words per utterance (Avg. W) and total number of utterances (U)

tection task between two entities, either a human and bot or a human and human. We assume that one entity is always human, and the other entity is unknown—either a human or a bot. Formally, let $D = [u_1^h, u_2^o, \ldots, u_{N-1}^h, u_N^o]$ represents a sequence of utterances, where u_i^h represents the *i*-th utterance in a conversation and that it was made by a human h. Likewise, u_i^o represents an unknown entity's utterance (human or bot). N is the total number of utterances in the conversation. Our goal is to develop a classifier $f(D)$ that maps to a class in the set $T = \{\text{human-human}, \text{human-bot}\}$, where the human-bot means a bot is a part of the conversation. We formulate two bot detection datasets for this task consisting of three known bot datasets: ConvAI2, WOCHAT, and DailyDialog. The basic statistics of each dataset are shown in Table 1. WOCHAT and DailyDialog are used to form a Control dataset. We describe each dataset below:

ConvAI2. We use four datasets from the Second Conversational Intelligence Challenge (ConvAI2). The motivation behind the ConvAI2 challenge was to develop new approaches towards open-domain chatbots (Zhang et al., 2018; Dinan et al., 2019). Overall, we use two types of ConvAI2 datasets: training datasets containing human-human interactions and evaluation phase datasets containing human-bot interactions. First, we use the PERSONA-CHAT dataset, a collection of human-human interactions where researchers instructed Mechanical Turk users to converse with one another, assuming specific personas (i.e., profile descriptions). The ConvAI2 competition used automatic (e.g., Perplexity) and human evaluation procedures. The human evaluation procedures involve either paid workers or volunteers that interact with models built using PERSONA-CHAT, after which the volunteers/workers are asked to rate their interactions. Moreover, the human evaluator is instructed to interact with the bots in the same way as the Mechanical Turk users who participated in creating the PERSONA-CHAT dataset.

We use three "evaluation-phase" datasets [1]: TOLOKERS, VOLUNTEERS, and INTERMEDIATE. The TOLOKERS used solicited workers to chat with the models, similar to the PERSONA-CHAT's Mechanical Turk setup. Specifically, TOLOKERS consists of data collected during DeepHack.Chat [2] hackathon via paid workers using the Yandex.Toloka service. The INTERMEDIATE dataset consists of more dialogues by the bots from DeepHack.Chat, but the interactions come from volunteers. Finally, the VOLUNTEER dataset was collected during the final "wild evaluation" round of the ConvAI2 competition. Human-bot interactions were collected from volunteers through the Facebook Messenger and Telegram APIs.

Control Dataset. One of the research questions we explore in this paper is related to out-of-domain performance. Specifically, can we detect bots based on human response when data were collected under different settings and where the conversation topics differ? To address the out-of-domain research question, we use two additional datasets collected in the The Workshop on Chatbots and Conversational Agent Technologies (WOCHAT) (Kong-Vega et al., 2019). As part of a shared task, the workshop makes several bots available, has participants contribute new bots, and participants interact with the bots providing utterance-level feedback regarding their performance. We use two human-bot interaction datasets released by the organizers: IRIS and the TickTock dataset. [3] both IRIS (Banchs and Li, 2012) and TickTock (Yu et al., 2015) are retrieval-based bot variants that were "trained" on different datasets. Unlike the ConvAI2 task, humans are not instructed to converse with the bots as if they are another human. Thus, sometimes humans will ask things such as, "Are you a Robot?". This point

---

[1] `http://convai.io/data/`

[2] `http://deephack.me/chat`

[3] `http://workshop.colips.org/wochat/data/index.html`

provides a unique aspect that increases cross-data differences.

Unfortunately, human-human conversations were not released from the WOCHAT shared task. The focus was annotating human-bot interactions. Thus, we need to augment WOCHAT with real human-human conversations. Our study uses the DailyDialog dataset as our source of human-human interactions (Li et al., 2017). Unlike the ConvAI2-related data, DailyDialog consists of conversations between English learners practicing everyday English dialog in daily life.

Data Processing. We split the data into three groups: Unpaid (U), Paid (P), and Control (C). The U dataset consists of PERSONA-CHAT, INTERMEDIATE, and VOLUNTEER ConvAI2 datasets. The P dataset consists of PERSONA-CHAT and TOLOKERS—the name Paid comes from the fact that the TOLOKERS were paid as part of a crowdsourcing task. We split this group from the others in case this affects conversation behavior. The C dataset consists of IRIS, TICK-TOCK, and DailyDialog. Each dataset is divided into 70%, 10%, and 20% training, validation, and test splits, respectively. It is important to note that the same PERSONA-CHAT training, validation, and test examples are used in both the U and P datasets.

4 Method

To detect human-bot conversations, we explore two types of features below: Content and Stylistic features.

4.1 Content Features

We define content information as features describing "*what*" humans and bots say in their interactions. Specifically, we describe two sets of content features: bag-of-words and embeddings.

Bag-of-words. As a simple baseline, we use TF-IDF-weighted unigrams from a dialog to detect whether a bot is part of the conversation. We explore three settings for the TF-IDF features: human-only, unknown-only, and human-unknown. The *unknown* user in the conversation can be either a bot or a human. Thus, in the human-Only setting, only the human's unigrams from each dialogue $D_h = [u_1^h, u_3^h, \ldots, u_{N-1}^h]$ are used to detect human-bot interactions. The unknown-only setting uses only the unknown user's unigrams from each dialogue $D_o = [u_2^o, u_4^o, \ldots, u_N^o]$, and the human-unknown setting uses both the human's and unknown user's unigrams from each dialogue $D = [u_1^h, u_2^o, \ldots, u_{N-1}^h, u_N^o]$.

Embedding Features. We encoded each dialogue D using BERT (Devlin et al., 2019)[4] by feeding the first 510 WordPieces and then averaging the word representations extracted from the second-to-last layer. We explore three variants, human-Only, Unknown-Only, and human-Unknown. Depending on the variant, the exact WordPieces passed to BERT change. For instance, in the human-Only setting, the first 510 WordPieces made by the human from a concatenation of all of the humans utterances $D_h = [u_1^h, u_3^h, \ldots, u_{N-1}^h]$ are passed to BERT. Similarly, all of the unknown user's utterances are used for the unknown-only variant to obtain $D_o = [u_2^o, u_4^o, \ldots, u_N^o]$. In the human-unknown setting, all human and unknown utterances are concatenated in $D = [u_1^h, u_2^o, \ldots, u_{N-1}^h, u_N^o]$ before generating features with BERT.

4.2 Stylistic Features

The stylistic features encode "*how*" humans and bots speak in their interactions. We use two sets of stylistic features: Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2015) and linguistic accommodation (Danescu-Niculescu-Mizil et al., 2011).

LIWC. We experiment with the psychologically validated word categories (e.g., positive emotion, cognitive, and social processes) in LIWC as features. These lexicons might reveal more about a writer's thought processes, emotional states, and intentions. For LIWC features, we use the same word categories described in the Linguistic Accommodation Section below to train our LIWC-based classifier. Specifically, we use LIWC to process each utterance. Next, each utterance's LIWC scores are averaged together to form a dialogue-specific feature vector. We experiment with LIWC features variants: human-only, unknown-only, and human-unknown. Each setting averages the LIWC scores across a different set of utterances, similar to the bag-of-words and embedding features.

Linguistic Accommodation. We use the method

---

[4] We use the bert-base-uncased pre-trained model available in the HuggingFace package (Wolf et al., 2019).

proposed by Danescu-Niculescu-Mizil et al. (2011) to analyze linguistic alignment. It includes two primary group-level and conversation-level measures: baseline word usage and alignment. The group-level baseline word use corresponds to the rate at which a person uses a given *word category* when it has not been used previously in a conversation. The group-level alignment score reflects the proportionate increase—compared to the baseline score—in the likelihood of the word being used when it has been used previously in the conversation. Similarly, the conversation-level metrics measure accommodation and usage between a human and Unknown (bot or human) user in the each dialogue D_i. We measure the probability of seeing a word category (e.g., LIWC categories) w_c given w_c appeared in the previous utterance $P(w_c \in u_i^o | w_c \in u_{i-1}^h)$[5] and the baseline probability of seeing w_c in the conversation $P(w_c \in u_i^o)$. Next, the accommodation score is calculated by taking the difference between both probabilities, $acc(c) = P(w_c \in u_i^o | w_c \in u_{i-1}^h) - P(w_c \in u_i^o)$. These empirical probabilities are at the conversation level (i.e., measuring how two specific users align). To obtain the group-level estimates, we simply average the accommodation scores $acc(c)$ over all conversations D_i for each word category. We use the following 17 LIWC categories (Pennebaker et al., 2015): i, you, we, they, social, cogproc, posemo, negemo, article, prep, certain, conj, discrep, negate, pronoun, quant, and tentat. We experiment with two main settings as described for the previous methods: Human-Only, Unknown-Only. The Human-Only setting consists of just using the known human's alignment scores in each conversation and vice-versa for the Unknown-Only setting.

Model Training Details. For the content feature sets, we train a Logistic Regression classifier from the Scikit-Learn package (Pedregosa et al., 2011). Using the validation split for each dataset, we grid-search over the C-values {.0001, .001, .01, .1, 1., 10.}, the logistic regression class weight parameters {None, balanced}, and normalization procedures {standardize, unit normalize, None}. For the Stylistic features, we train a Random Forest classifier from the Scikit-Learn package. Again, using the validation split, we grid-search over the class weight parameters {None, balanced, balanced_subsample}, criterion measures {gini, entropy}, max features {sqrt, log2, None}, and bootstrap parameters {True, False}. For all experiments using the Random Forest classifier, we set n_estimators to 1000.

	U → U	P → P	C → C
Baselines			
Most Frequent	.482	.462	.493
Most Infrequent	.065	.125	.027
Stratified (random)	.491	.498	.487
Content Features			
Human Bag-of-Words	.980	.987	.939
Human BERT	.989	.996	.987
Unknown Bag-of-Words	.971	.970	.997
Unknown BERT	.996	.996	.983
Human + Unknown Bag-of-Words	.958	.963	.986
Human + Unknown BERT	.990	.992	.990
Stylistic Features			
Human LIWC	.878	.899	.705
Human Accommodation	.989	.987	.627
Unknown LIWC	.838	.862	.707
Unknown Accommodation	.887	.897	.674
Human & Unknown LIWC	.885	.903	.601
Human & Unknown Accommodation	.988	.990	.729

Table 2: Source → Source Macro F1 Results for Bot Detection

5 Results

In this section, we report two sets of results. First, in Subsection 5.1 we present the performance of the bot detection models we explain in Section 4. Beyond the methods described in Section 4, we also compare three baselines: Most Frequent, Most Infrequent, and Stratified. The Most Frequent baseline predicts the most frequent class for every example (i.e., human-human). The Most Infrequent baseline predicts the most infrequent class (i.e., human-bot) and the Stratified baseline makes random prediction proportional to each class's frequency. Second, in Subsection 5.2, we present a fine-grained analysis of the linguistic accommodation results on the ConvAI2 datasets. For all results, we report the Macro F1 (average F1 for the human-human and human-bot conversation classes).

5.1 Bot Detection Experiments

The source dataset results are reported in Table 2. Overall, we find that content features are the most

---

[5] The probabilities are from the perspective of the Unknown user in the conversation. They are also calculated from the human's perspective.

		U → C	**P → C**	**C → U**	**C → P**	**AVG**
Baselines	1. Most Frequent	.493	.493	.482	.462	.483
	2. Most Infrequent	.027	.027	.065	.125	.061
	3. Stratified (Random)	.502	.479	.499	.476	.489
Content: What the bots and humans write?	4. Human bag-of-words	.518	.504	.536	.608	.541
	5. Human BERT	.493	.493	.482	.462	.483
	6. Unknown bag-of-words	.493	.493	.535	.475	.499
	7. Unknown BERT	.493	.493	.482	.462	.483
	8. Unknown and Human bag-of-words	.509	.510	.639	.608	.567
	9. Unknown and Human BERT	.493	.493	.522	.478	.497
Stylistic: How the bots and humans write?	10. Human LIWC	.480	.491	.524	.512	.502
	11. Human Accommodation	**.631**	.591	.604	.510	.584
	12. Human LIWC + Accommodation	.605	.608	**.703**	**.724**	**.660**
	13. Unknown LIWC	.474	.478	.483	.476	.478
	14. Unknown Accommodation	.428	.424	.503	.494	.462
	15. Unknown LIWC + Accommodation	.462	.436	.502	.497	.474
	13. Human & Unknown LIWC	.521	.504	.532	.556	.528
	14. Human & Unknown Accommodation	.611	.620	.642	.677	.637
	15. Human & Unknown LIWC + Accommodation	.622	**.633**	.642	.677	.643

Table 3: This table reports the cross-dataset Macro F1 score for detecting human-bot conversations for three datasets: Unpaid (U), Paid (P), and Control (C). The largest Macro F1 score in each column is in **bold**.

predictive for bot detection, when training and testing on the same train-test splits from the same dataset. Furthermore, we find that the BERT-based models are able to outperform the Bag-of-Words models on average. For instance, the Human Bag-of-Words model on dataset C obtains an F1 of .939. Yet, the Human BERT model obtains an F1 of .987. Likewise, for the stylistic features, we find that simply using LIWC works better than using the accommodation features alone for dataset C. However, combining both Accommodation and LIWC features from both the Human and the Bot is better than using either feature set individually with an F1 of .729.

The cross-dataset bot detection results are presented in Table 3. Specifically, the scores are from experiments where we train on a source dataset and evaluate each model on a target dataset's test split (i.e., source → target). Overall, we make three major findings. First, for both Content and Stylistic features, we find that analyzing the known human's language in each conversation is more informative than analyzing the bot's content. For instance, the average (AVG) score for human bag-of-words is .541, while the bot bag-of-words model AVG result is nearly 4% lower (.499). We have similar findings between human LIWC (.502) and bot LIWC (.478). We find that combining both bot and human LIWC improves the AVG performance of the human-only LIWC model with an F1 of 0.528. Second, while the BERT-based model performs better when applied to data from the same datasets in Table 2 (e.g., U → U) in many settings, when the test dataset changes substantially (e.g., U → C), the generalization performance of BERT drops compared to using bag-of-words. This result is potentially caused by overfitting to random source-specific characteristics. Third, we find that Accommodation features outperform all other individual feature sets with an AVG bot-detection Macro F1 of .584, with the exception of using both the human's and bot's accommodation features which has a Macro F1 of .637. Interestingly, human accommodation information (.584) is more predictive than the bot's text (.462). Intuitively, the type of responses generated by a bot can differ substantially depending on the bot's training data (e.g., ConvAI2 vs. DailyDialog) and model (e.g., transformer vs. retrieval-based models). Hence, the human's responses are more consistent with regard to alignment. The best combination overall is the combination of the human's LIWC features with the human's accommodation features achieving a Macro F1 of .660. Yet, with the bot (Unknown) feature combination, the performance drops from .768 with Unknown LIWC to .674 after combining accommodation features.

In Table 4, we analyze the Random Forest's importance scores for the best model (trained on dataset P's human responses) features using the model that combines human LIWC and accommo-

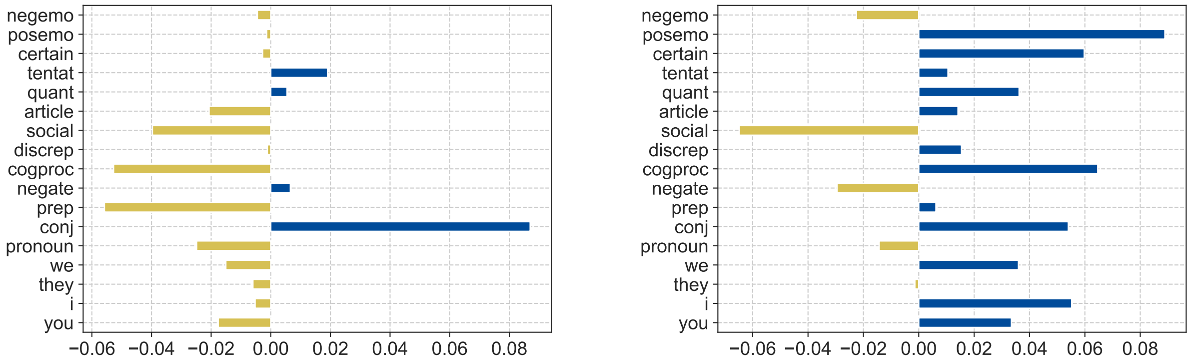

(a) Human TickTock Alignment

(b) Bot TockTock Alignment

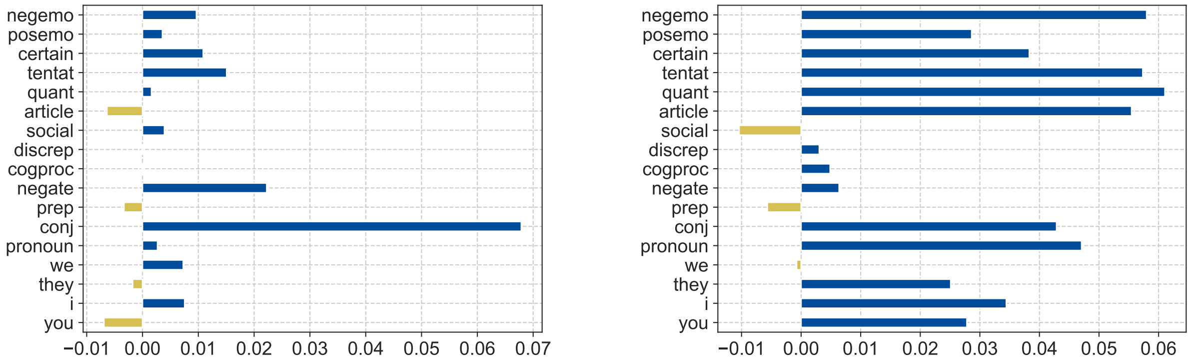

(c) Human INTERMEDIATE Alignment

(d) Bot INTERMEDIATE Alignment

Figure 2: Alignment scores for the TickTock and INTERMEDIATE datasets.

Feature Group	**Feature**	**FI**
Accomodation	pronoun	.598
LIWC	social	.063
LIWC	article	.043
LIWC	prep	.028
LIWC	pronoun	.020
Accomodation	quant	.018
LIWC	tentat	.017
Accomodation	i	.017
LIWC	you	.016
LIWC	i	.015

Table 4: The top ten most *informative* features for the Random Forest model trained on dataset P's human responses to detect human-bot conversations with LIWC and Accommodation features.

dation features. The most informative feature is pronoun accommodation. See Section 6 for more details. Other informative features include the human's use of social, quant (quantitative), and tentative words.

5.2 Accommodation Analysis

In this section, we analyze the group-level (averaged) accommodation scores for the INTERMEDIATE and TickTock datasets. Again, note that the group-level scores are dataset averages for each group (human and bot), the conversation-level estimates can vary from the group scores. The accommodation analysis is displayed in Figure 2. The bot results (Figures 2d and 2b) indicate how the bot responds to a human, and the human scores (Figures 2c and 2a) represent how a human responds to a bot. We report all LIWC categories, including Pronoun usage (e.g., pronoun, I, you, we, and they) and Rhetorical words (e.g., social, cogproc, posemo, and negemo) results. Find the analysis for all of the datasets in the Supplementary Material.

Overall, some indicators are robust across datasets, whereas others can vary. For instance, for the general pronoun category, we find that human generally accommodate less towards bots than bots accommodate towards them. The pronoun accommodation pattern for pronouns is robust across both datasets. Similarly, for both datasets, humans accomodate more often than bots for social words. Yet, for negative emotion words (negemo), we find that Bot's accommodate less than humans in the TickTock dataset, yet they accommodate more in the INTERMEDIATE dataset. In contrast, in the ConvAI2 and DailyDialog datasets, human accommodation is the same *on average* when two humans participate in a conversation together (not shown in the Figures).

6 Discussion

Why is humans' accommodation such a powerful predictor that they are in a conversation with a bot—even when they are supposedly treating the conversation the same as a human-human interaction? Overall, we found two major reasons to explain the model's high performance. First, modern bots from the transformer to retrieval-based models generate bad responses. For instance, in the conversation below from the VOLUNTEER dataset, the human and bot have already had a conversation about hair color:

```
Bot:   What is your hair color?
Human: I said its black....
Bot:   I know the feeling.
Human: What?
```

From the example, we can see that the bot is asking about hair color, which was already discussed, then the bot responds with a nonsensical output, "I know the feeling." Thus, we see that the final human response does not accommodate with with"I" nor any first singular pronoun. The lower alignment score for humans (compared to bots) for the word category I in Figure 2 supports this finding. Furthermore, from Table 4, we know that accommodation in terms of pronoun usage is highly predictive of a human-bot conversation. Similar posts can be found in all datasets where bad outputs can explain why humans align differently concerning "I" in bot-human interactions than human-human interactions. This finding is interesting because it is similar to the findings in human-human interactions where alignment is positively correlated with understanding in cooperative decision-making tasks (Fusaroli et al., 2012; Kacewicz et al., 2014). If humans don't understand the conversation, they align less, at least with regards to pronoun usage in our case. The next example from the VOLUNTEER dataset is a better, more coherent conversation:

```
Human: What is your Hobby?
Bot:   I love music! I love Taylor Swift
Human: Me too. And what about Iggy Pop?
```

We can see that the final human response aligns with the first person singular word "me" to the bot's use of "I". A potential use that warrants future work in the area of bots evaluation involves using accommodation information as an additional evaluation metric in human evaluation studies if certain types of accommodation correlate with bad system responses. If human ratings of bot performance are noisy, the use of accomodation information can provide insight into how well a bot performs, if a human interacts with it differently than they do in human-human conversations.

The second reason for the significant predictive performance can be seen in adversarial conversations between the human and bot. Specifically, in instances where the human is trying to trick the bot or simply trying to ask questions, bot-human alignment can differ from human-human conversations. The example below from the TickTock dataset

```
Human: Are you a robot?
Bot:   No.
Human: Where are you from?
```

shows repeated "*You*" usage by the user, repeatedly asking multiple questions. Again, differences in pronoun usage can reflect adversarial users. Again, in terms of bot evaluation, if a human is acting adversarially, and it is possible to differentiate adversarial interactions from the human not understanding, this can provide insight into . More examples can be found in the Supplementary Material.

7 Conclusion

This study has is an exploration of the levels of linguistic alignment in human-bot conversations and presents evidence about how bot-human interactions are different than human-human conversations. This work's insights have implications for future bot detection methodologies. How humans and bots come together and converse manifested by their level of agreement and disagreement is a first step towards understanding human interactions with general conversational agents. Furthermore, as future work, we will explore the use of linguistic alignment to improve bot evaluation frameworks involving human-bot interactions. Moreover, we will also explore methods of controlling the accommodation level of bots themselves. Given the implicit social implications of certain types of accommodation, controlling it can help important tasks (e.g., counseling or customer service-related bots).

References

Emily Ahn, Cecilia Jimenez, Yulia Tsvetkov, and Alan Black. 2020. What code-switching strategies are effective in dialogue systems? *Proceedings of the Society for Computation in Linguistics*, 3(1):308–318.

Rafael E Banchs and Haizhou Li. 2012. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, pages 37–42.

David M Beskow and Kathleen M Carley. 2018. Bot conversations are different: leveraging network metrics for bot detection in twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 825–832. IEEE.

Ana Paula Chaves. 2020. Should my chatbot be register-specific? designing appropriate utterances for tourism. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–11.

Ana Paula Chaves, Eck Doerry, Jesse Egbert, and Marco Gerosa. 2019. It's how you say it: Identifying appropriate register for chatbot language design. In *Proceedings of the 7th International Conference on Human-Agent Interaction*, pages 102–109.

Michelle Cohn, Chun-Yen Chen, and Zhou Yu. 2019. A large-scale user study of an alexa prize chatbot: Effect of tts dynamism on perceived quality of social dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 293–306.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words! linguistic style accommodation in social media. In *Proceedings of the 20th international conference on World wide web*, pages 745–754.

Florian Daniel, Cinzia Cappiello, and Boualem Benatallah. 2019. Bots acting like humans: Understanding and preventing harm. *IEEE Internet Computing*, 23(2):40–49.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Gabriel Doyle and Michael C Frank. 2016. Investigating the sources of linguistic alignment in conversation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 526–536.

Ela Elsholz, Jon Chamberlain, and Udo Kruschwitz. 2019. Exploring language style in chatbots to increase perceived product value and user engagement. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, pages 301–305.

Riccardo Fusaroli, Bahador Bahrami, Karsten Olsen, Andreas Roepstorff, Geraint Rees, Chris Frith, and Kristian Tylén. 2012. Coming to terms: Quantifying the benefits of linguistic coordination. *Psychological science*, 23(8):931–939.

Andres Garcia-Silva, Cristian Berrio, and José Manuel Gómez-Pérez. 2019. An empirical study on pre-trained embeddings and language models for bot detection. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 148–155.

Howard Giles, Donald M Taylor, and Richard Bourhis. 1973. Towards a theory of interpersonal accommodation through language: Some canadian data. *Language in society*, pages 177–192.

Augusto Gnisci. 2005. Sequential strategies of accommodation: A new method in courtroom. *British Journal of Social Psychology*, 44(4):621–643.

Jonathan Herzig, Tommy Sandbank, Michal Shmueli-Scheuer, and David Konopnicki. 2019. Bot2vec: Learning representations of chatbots. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM 2019)*, pages 75–84.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822.

Ganesh Jawahar, Muhammad Abdul-Mageed, and VS Laks Lakshmanan. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309.

Simon Jones, Rachel Cotterill, Nigel Dewdney, Kate Muir, and Adam Joinson. 2014. Finding zelig in text: A measure for normalising linguistic accommodation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 455–465.

Ewa Kacewicz, James W Pennebaker, Matthew Davis, Moongee Jeon, and Arthur C Graesser. 2014. Pronoun use reflects standings in social hierarchies. *Journal of Language and Social Psychology*, 33(2):125–143.

Dominika Kaczorowska-Spychalska. 2019. How chatbots influence marketing. *Management*, 23(1):251–270.

Alice Kerlyl, Phil Hall, and Susan Bull. 2006. Bringing chatbots into education: Towards natural language negotiation of open learner models. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 179–192. Springer.

Susan L Kline and Janet M Ceropski. 1984. Person-centered communication in medical practice. *Human decision-making*, pages 120–141.

Jürgen Knauth. 2019. Language-agnostic twitter-bot detection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 550–558.

Naomi Kong-Vega, Mingxin Shen, Mo Wang, and Luis Fernando D'Haro. 2019. Subjective annotation and evaluation of three different chatbots wochat: Shared task report. In *9th International Workshop on Spoken Dialogue System Technology*, pages 371–378. Springer.

Dijana Kosmajac and Vlado Keselj. 2019. Twitter bot detection using diversity measures. In *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, pages 1–8.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995.

Anna Liednikova, Philippe Jolivet, Alexandre Durand-Salmon, and Claire Gardent. 2020. Learning health-bots from training data that was automatically created using paraphrase detection and expert knowledge. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 638–648.

Ming-Cheng Ma and John P Lalor. 2020. An empirical analysis of human-bot interaction on reddit. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 101–106.

Alistair Martin, Jama Nateqi, Stefanie Gruarin, Nicolas Munsch, Isselmou Abdarahmane, Marc Zobel, and Bernhard Knapp. 2020. An artificial intelligence-based first-line defence against covid-19: digitally screening citizens for risks via a chatbot. *Scientific reports*, 10(1):1–7.

Kate G Niederhoffer and James W Pennebaker. 2002. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21(4):337–360.

Timothy Niven and Hung-Yu Kao. 2019. Detecting argumentative discourse acts with linguistic alignment. In *Proceedings of the 6th Workshop on Argument Mining*, pages 104–112.

David Oniani and Yanshan Wang. 2020. A qualitative evaluation of language models on automatic question-answering for covid-19. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 1–9.

Mariam Orabi, Djedjiga Mouheb, Zaher Al Aghbari, and Ibrahim Kamel. 2020. Detection of bots in social media: a systematic review. *Information Processing & Management*, 57(4):102250.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? research in commercial spoken dialog systems. In *International Conference on Text, Speech and Dialogue*, pages 3–13. Springer.

Eva Sharma and Munmun De Choudhury. 2018. Mental health support and its relationship to linguistic accommodation in online communities. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13.

Hagyeong Shin and Gabriel Doyle. 2018. Alignment, acceptance, and rejection of group identities in online political discourse. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 1–8.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Paul J Taylor and Sally Thomas. 2008. Linguistic style matching and negotiation outcome. *Negotiation and Conflict Management Research*, 1(3):263–281.

Paul Thomas, Daniel McDuff, Mary Czerwinski, and Nick Craswell. 2020. Expressions of style in information seeking conversation with an agent. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1171–1180.

Yafei Wang, David Reitter, and John Yen. 2014. Linguistic adaptation in conversation threads: Analyzing alignment in online health communities. *ACL 2014*, page 55.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Nathan I Wood. 2019. Departing from doctor-speak: A perspective on code-switching in the medical setting. *Journal of general internal medicine*, 34(3):464–466.

Yang Xu, Jeremy Cole, and David Reitter. 2018. Not that much power: Linguistic alignment is influenced more by low-level linguistic features rather than social power. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 601–610.

Deepika Yadav, Prerna Malik, Kirti Dabas, and Pushpendra Singh. 2019. Feedpal: Understanding opportunities for chatbots in breastfeeding education of women in india. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–30.

Zhou Yu, Alexandros Papangelis, and Alexander Rudnicky. 2015. Ticktock: A non-goal-oriented multimodal dialog system with engagement awareness. In *2015 AAAI Spring symposium series*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

A Expanded Alignment Analysis

The complete group-level alignment scores for the IRIS and Tolokers datasets can be found in Figures 3a, 3b, 3c, and 3d.

B Example Conversations

Here we present examples of conversations between humans and bots that are representative of good conversational structure in terms of alignment between the two parties such as the usage of pronoun "*I*" in the first example.

```
Bot:   I love iphone!
       i just bought new iphone!
Human: Thats good for you  i m not
       very into new tech
```

```
Human: I am fine. I like hip hop.
       What about you?
Bot:   I like r b and pop
       are you 100 years old?
Human: What? No. You are rude
```

```
Human: Do you have dogs?
Bot:   Yes  i have two dogs  and a cat.
Human: Cool  I love dogs.
       But i have siblings instead...
Bot:   I have two brothers and
       a sister  and a sister.
```

Here conversations between humans and bots involve irregular pronoun usage that are representative of a break in the conversational pattern and little or no alignment between the two parties such as the usage of pronouns "*You*" and resultant use of "*I*" reply in the first example.

```
Human: See you then!
Bot:   I m not going to see you.
```

```
Bot:   I am 24 and I have a dog named
       her name is named name
Human: Really her name is name?
Bot:   She is a good name.
```

```
Human: who is obama
Bot:   i partially have interest in that
Human: can you tell me more about obama
Bot:   i hate pigs i live a farm
```

These conversations signify a complete breakdown of conversational structure with respect to alignment, as there is none. The chats seem haphazard and vague with either the bot or the human replying in an ambiguous manner.

```
User:   Good afternoon
System: Well  you know  I'm not sure.
        What do you think about movies
User:   That didn't make any sense
```

```
User:   hi again
System: Of course. Any thoughts?
User:   no
```

```
User:   Hello
System: Do not argue with an idiot.
        He will drag you down to his
        level and beat you with
        experience.
User:   Um  Okay  that was random
```

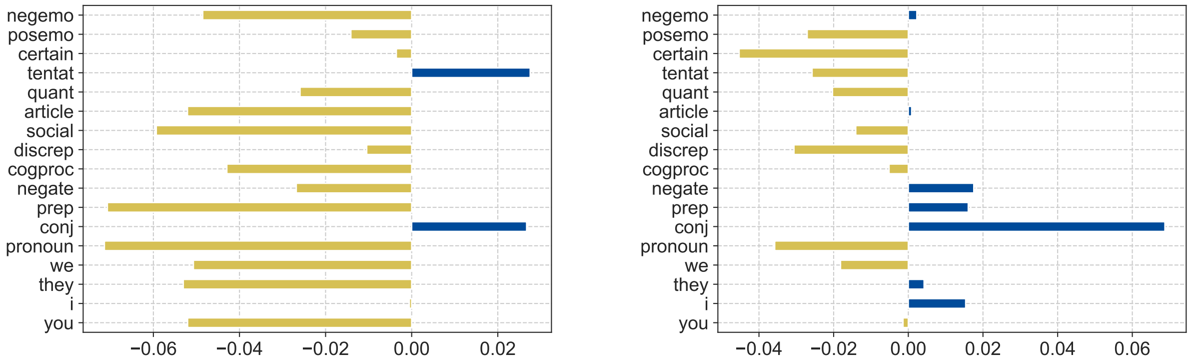

(a) Human IRIS Alignment

(b) Bot IRIS Alignment

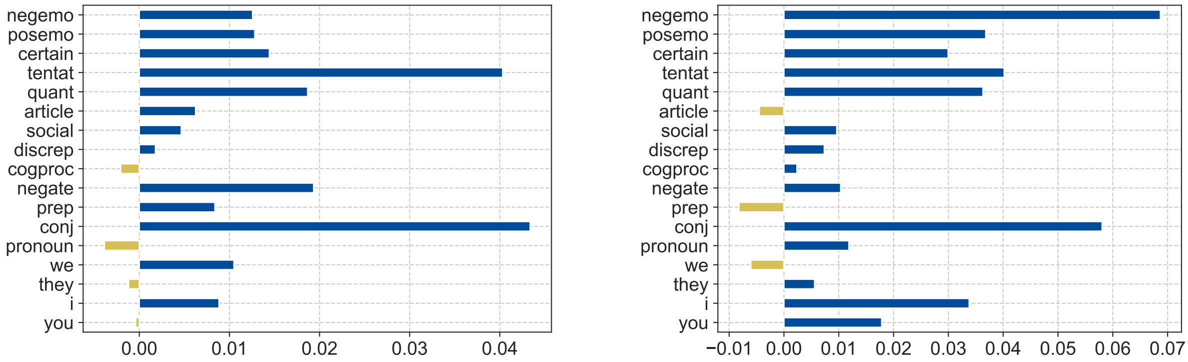

(c) Human Tolokers Alignment

(d) Bot Tolokers Alignment

Figure 3: Alignment scores for the IRIS and Tolokers dataset

Defending Pre-trained Language Models from Adversarial Word Substitution Without Performance Sacrifice

Rongzhou Bao, Jiayi Wang, Hai Zhao*
Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
MoE Key Lab of Artificial Intelligence, AI Institute,
Shanghai Jiao Tong University, Shanghai, China
`rongzhou.bao@outlook.com`
`{wangjiayi_102_23}@sjtu.edu.cn,zhaohai@cs.sjtu.edu.cn`

Abstract

Pre-trained contextualized language models (PrLMs) have led to strong performance gains in downstream natural language understanding tasks. However, PrLMs can still be easily fooled by adversarial word substitution, which is one of the most challenging textual adversarial attack methods. Existing defence approaches suffer from notable performance loss and complexities. Thus, this paper presents a compact and performance-preserved framework, **A**nomaly **D**etection with **F**requency-**A**ware **R**andomization (ADFAR). In detail, we design an auxiliary anomaly detection classifier and adopt a multi-task learning procedure, by which PrLMs are able to distinguish adversarial input samples. Then, in order to defend adversarial word substitution, a frequency-aware randomization process is applied to those recognized adversarial input samples. Empirical results show that ADFAR significantly outperforms those newly proposed defense methods over various tasks with much higher inference speed. Remarkably, ADFAR does not impair the overall performance of PrLMs. The code is available at https://github.com/LilyNLP/ADFAR.

1 Introduction

Deep neural networks (DNNs) have achieved remarkable success in various areas. However, previous works show that DNNs are vulnerable to adversarial samples (Goodfellow et al., 2015; Kurakin et al., 2017; Wang et al., 2021), which are inputs with small, intentional modifications that cause the model to make false predictions. Pre-trained language models (PrLMs) (Devlin et al., 2019; Liu et al., 2019; Clark et al., 2020; Zhang et al., 2020, 2019) are widely adopted as an essential component for various NLP systems. However, as DNN-based models, PrLMs can still be easily fooled by textual adversarial samples (Wallace et al., 2019; Jin et al., 2019; Nie et al., 2020; Zang et al., 2020). Such vulnerability of PrLMs keeps raising potential security concerns, therefore researches on defense techniques to help PrLMs against textual adversarial samples are imperatively needed.

Different kinds of textual attack methods have been proposed, ranging from character-level word misspelling (Gao et al., 2018), word-level substitution (Alzantot et al., 2018; Ebrahimi et al., 2018; Ren et al., 2019; Jin et al., 2019; Zang et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020), phrase-level insertion and removal (Liang et al., 2018), to sentence-level paraphrasing (Ribeiro et al., 2018; Iyyer et al., 2018). Thanks to the discrete nature of natural language, attack approaches that result in illegal or unnatural sentences can be easily detected and restored by spelling correction and grammar error correction (Islam and Inkpen, 2009; Sakaguchi et al., 2017; Pruthi et al., 2019). However, attack approaches based on adversarial word substitution can produce high-quality and efficient adversarial samples which are still hard to be detected by existing methods. Thus, the adversarial word substitution keeps posing a larger and more profound challenge for the robustness of PrLMs. Therefore, this paper is devoted to overcome the challenge posed by adversarial word substitution.

Several approaches are already proposed to mitigate issues posed by adversarial word substitution (Zhou et al., 2019; Jia et al., 2019; Huang et al., 2019; Cohen et al., 2019; Ye et al., 2020; Si et al., 2021). Although these defense methods manage to alleviate the negative impact of adversarial word

---

*Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), Key Projects of National Natural Science Foundation of China (U1836222 and 61733011). This work was supported by Huawei Noah's Ask Lab

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3248–3258
August 1–6, 2021.

substitution, they sometimes reduce the prediction accuracy for non-adversarial samples to a notable extent. Given the uncertainty of the existence of attack in real application, it is impractical to sacrifice the original prediction accuracy for the purpose of defense. Moreover, previous defense methods either have strong limitations over the attack space to certify the robustness, or require enormous computation resources during training and inference. Thus, it is imperatively important to find an efficient performance-preserved defense method.

For such purpose, we present a compact and performance-preserved framework, **A**nomaly **D**etection with **F**requency-**A**ware **R**andomization (ADFAR), to help PrLMs defend against adversarial word substitution without performance sacrifice. Xie et al. (2018) show that introducing randomization at inference can effectively defend adversarial attacks. Moreover, (Mozes et al., 2020) indicate that the usual case for adversarial samples is replacing words with their less frequent synonyms, while PrLMs are more robust to frequent words. Therefore, we propose a frequency-aware randomization process to help PrLMs defend against adversarial word substitution.

However, simply applying a randomization process to all input sentences would reduce the prediction accuracy for non-adversarial samples. In order to preserve the overall performance, we add an auxiliary anomaly detector on top of PrLMs and adopt a multi-task learning procedure, by which PrLMs are able to determine whether each input sentence is adversarial or not, and not introduce extra model. Then, only those adversarial input sentences will undergo the randomization procedure, while the prediction process for non-adversarial input sentences remains the same.

Empirical results show that as a more efficient method, ADFAR significantly outperforms previous defense methods (Ye et al., 2020; Zhou et al., 2019) over various tasks, and preserves the prediction accuracy for non-adversarial sentences. Comprehensive ablation studies and analysis further prove the efficiency of our proposed method, and indicate that the adversarial samples generated by current heuristic word substitution strategy can be easily detected by the proposed auxiliary anomaly detector.

2 Related Work

2.1 Adversarial Word Substitution

Adversarial word substitution (AWS) is one of the most efficient approaches to attack advanced neural models like PrLMs. In AWS, an attacker deliberately replaces certain words by their synonyms to mislead the prediction of the target model. At the same time, a high-quality adversarial sample should maintain grammatical correctness and semantic consistency. In order to craft efficient and high-quality adversarial samples, an attacker should first determine the vulnerable tokens to be perturbed, and then choose suitable synonyms to replace them.

Current AWS models (Alzantot et al., 2018; Ebrahimi et al., 2018; Ren et al., 2019; Jin et al., 2019; Li et al., 2020; Garg and Ramakrishnan, 2020) adopt heuristic algorithms to locate vulnerable tokens in sentences. To illustrate, for a given sample and a target model, the attacker iteratively masks the tokens and checks the output of the model. The tokens which have significant influence on the final output logits are regarded as vulnerable.

Previous works leverage word embeddings such as GloVe (Pennington et al., 2014) and counter-fitted vectors (Mrkšić et al., 2016) to search the suitable synonym set of a given token. Li et al. (2020); Garg and Ramakrishnan (2020) uses BERT (Devlin et al., 2019) to generate perturbation for better semantic consistency and language fluency.

2.2 Defense against AWS

For general attack approaches, adversarial training (Goodfellow et al., 2015; Jiang et al., 2020) is widely adopted to mitigate adversarial effect, but (Alzantot et al., 2018; Jin et al., 2019) shows that this method is still vulnerable to AWS. This is because AWS models leverage dynamic algorithms to attack the target model, while adversarial training only involves a static training set.

Methods proposed by Jia et al. (2019); Huang et al. (2019) are proved effective for defence against AWS, but they still have several limitations. In these methods, Interval Bound Propagation (IBP) (Dvijotham et al., 2018), an approach to consider the worst-case perturbation theoretically, is leveraged to certify the robustness of models. However, IBP-based methods can only achieve the certified robustness under a strong limitation over the attack space. Furthermore, they are difficult to adapt to PrLMs for their strong reliance on the assumption

🟥 Rare word substitution 🟦 Random substitution

Original	The most **hopelessly** **monotonous** film, **noteworthy** only for the **gimmick** of being filmed as a **single** **unbroken** 87 **minute** take.
Randomized	The most **desperately** **boring** film, **amazing** only for the **trick** of being filmed as a **unitary** **continuous** 87 **minutes** take.
Original	A small **gem** of a **movie** that **defies** **classification** and is as **thought** provoking as it is funny, scary and **sad**.
Randomized	A small **essence** of a **film** that **challenge** **ranking** and is as **thinking** provoking as it is funny, scary and **gloomy**.

Figure 1: Frequency-aware randomization examples.

of model architecture.

Two effective and actionable methods (DISP (Zhou et al., 2019) and SAFER Ye et al. (2020)) are proposed to overcome the challenge posed by AWS, and therefore adopted as the baselines for this paper. DISP (Zhou et al., 2019) is a framework based on perturbation discrimination to block adversarial attack. In detail, when facing adversarial inputs, DISP leverages two auxiliary PrLMs: one to detect perturbed tokens in the sentence, and another to restore the abnormal tokens to original ones. Inspired by randomized smoothing (Cohen et al., 2019), Ye et al. (2020) proposes SAFER, a novel framework that guarantees the robustness by smoothing the classifier with synonym word substitution. To illustrate, based on random word substitution, SAFER smooths the classifier by averaging its outputs of a set of randomly perturbed inputs. SAFER outperforms IBP-based approaches and can be easily applied to PrLMs.

2.3 Randomization

In recent years, randomization has been used as a defense measure for deep learning in computer vision (Xie et al., 2018). Nevertheless, direct extensions of these measures to defend against textual adversarial samples are not achievable, since the text inputs are discrete rather than continuous. Ye et al. (2020) indicates the possibility of extending the application of the randomization approach to NLP by randomly replacing the words in sentences with their synonyms.

3 Method

3.1 Frequency-aware Randomization

Since heuristic attack methods attack a model by substituting each word iteratively until it successfully alters the model's output, it is normally difficult for static strategies to defense such kind of dynamic process. Rather, dynamic strategies, such as randomization, can better cope with the problem. It is also observed that replacing words with their more frequent alternatives can better mitigate the adversarial effect and preserve the original performance. Therefore, a frequency-aware randomization strategy is designed to perplex AWS strategy.

Figure 1 shows several examples of the frequency-aware randomization. The proposed approach for the frequency-aware randomization is shown in Algorithm 1, and consists of three steps. Firstly, rare words with lower frequencies and a number of random words are selected as substitution candidates. Secondly, we choose synonyms with the closest meanings and the highest frequencies to form a synonym set for each candidate word. Thirdly, each candidate word is replaced with a random synonym within its own synonym set. To quantify the semantic similarity between two words, we represent words with embeddings from (Mrkšić et al., 2016), which is specially designed for synonyms identification. The semantic similarity of two words are evaluated by cosine similarity of their embeddings. To determine the frequency of a word, we use a frequency dictionary provided by FrequencyWords Repository[*].

3.2 Anomaly Detection

Applying the frequency-aware randomization process to every input can still reduce the prediction accuracy for normal samples. In order to overcome this issue, we add an auxiliary anomaly detection head to PrLMs and adopt a multi-task learning procedure, by which PrLMs are able to classify the input text and distinguish the adversarial samples at the same time, and not introduce extra model. In inference, the frequency-aware randomization only applied to the samples that are detected as adversarial. In this way, the reduction of accuracy is largely avoided, since non-adversarial samples are not affected.

Zhou et al. (2019) also elaborates the idea of perturbation discrimination to block attack. However, their method detects anomaly on token-level and requires two resource-consuming PrLMs for detection and correction, while ours detects anomaly on sentence-level and requires no extra models. Com-

---

[*] https://github.com/hermitdave/FrequencyWords

Algorithm 1 Frequency-aware Randomization

```
Input: Sentence $X = \{w_1, w_2, ..., w_n\}$, word embeddings $Emb$ over the vocabulary $Vocab$
Output: Randomized sentence $X_{rand}$
Initialization: $X_{rand} \leftarrow X$
Create a set $W_{rare}$ of all rare words with frequencies less than $f_{thres}$, denote $n_{rare} = |W_{rare}|$.
Create a set $W_{rand}$ by randomly selecting $n * r - n_{rare}$ words $w_j \notin W_{rare}$, where $r$ is the pre-defined ratio of substitution.
Create the substitution candidates set, $W_{sub} \leftarrow W_{rare} + W_{rand}$, and $|W_{sub}| = n * r$.
Filter out the stop words in $W_{sub}$.
for each word $w_i$ in $W_{sub}$ do
    Create a set $S$ by extracting the top $n_s$ synonyms using $CosSim(Emb_{w_i}, Emb_{w_{word}})$ for each word in $Vocab$.
    Create a set $S_{freq}$ by selecting the top $n_f$ frequent synonyms from $S$.
    Randomly choose one word $w_s$ from $S$.
    $X_{rand} \leftarrow$ Replace $w_i$ with $w_s$ in $X_{rand}$.
end for
```

pared to Zhou et al. (2019), our method is two times faster in inference speed and can achieve better accuracy for sentence-level anomaly detection.

3.3 Framework

In this section, we elaborate the framework of ADFAR in both training and inference.

3.3.1 Training

Figure 2 shows the framework of ADFAR in training. We extend the baseline PrLMs by three major modifications: 1) the construction of training data, 2) the auxiliary anomaly detector and 3) the training objective, which will be introduced in this section.

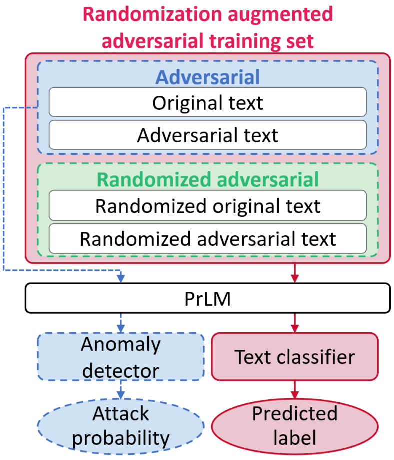

Figure 2: Framework of ADFAR in training.

Construction of Training Data As shown in Figure 2, we combine the idea of both adversarial training and data augmentation (Wei and Zou, 2019) to construct our randomization augmented adversarial training data. Firstly, we use a heuristic AWS model (e.g. TextFooler) to generate adversarial samples based on the original training set. Following the common practice of adversarial training, we then combine the adversarial samples with the original ones to form an adversarial training set. Secondly, in order to let PrLMs better cope with randomized samples in inference, we apply the frequency-aware randomization on the adversarial training set to generate a randomized adversarial training set. Lastly, the adversarial training set and the randomized adversarial training set are combined to form a randomization augmented adversarial training set.

Auxiliary Anomaly Detector In addition to the original text classifier, we add an auxiliary anomaly detector to the PrLMs to distinguish adversarial samples. For an input sentence, the PrLMs captures the contextual information for each token by self-attention and generates a sequence of contextualized embeddings $\{h_0, \ldots h_m\}$. For text classification task, $h_0 \in R^H$ is used as the aggregate sequence representation. The original text classifier leverages h_0 to predict the probability that X is labeled as class $\hat{y_c}$ by a logistic regression with softmax:

$$
\begin{aligned}
y_c &= Prob(\hat{y_c}|x), \\
&= \mathrm{softmax}(W_c(dropout(h_0)) + b_c),
\end{aligned}
$$

For the anomaly detector, the probability that X is labeled as class $\hat{y_d}$ (if X is attacked, $\hat{y_d} = 1$; if X is normal, $\hat{y_d} = 0$) is predicted by a logistic regression with softmax:

$$
\begin{aligned}
y_d &= Prob(\hat{y_d}|x), \\
&= \mathrm{softmax}(W_d(dropout(h_0)) + b_d),
\end{aligned}
$$

As shown in Figure 2, the original text classifier is trained on the randomization augmented adversarial training set, whereas the anomaly detector is only trained on the adversarial training set.

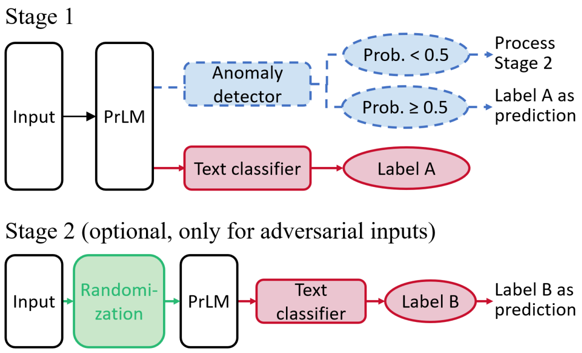

Figure 3: Framework of ADFAR in inference.

Training Objective We adopt a multi-task learning framework, by which PrLM is trained to classify the input text and distinguish the adversarial samples at the same time. We design two parallel training objectives in the form of minimizing cross-entropy loss: $loss_c$ for text classification and $loss_d$ for anomaly detection. The total loss function is defined as their sum:

$$loss_c = -[y_c * \log(\hat{y_c}) + (1 - y_c) * \log(1 - \hat{y_c})]$$
$$loss_d = -[y_d * \log(\hat{y_d}) + (1 - y_d) * \log(1 - \hat{y_d})]$$
$$Loss = loss_c + loss_d$$

3.3.2 Inference

Figure 3 shows the framework of ADFAR in inference. Firstly, the anomaly detector predicts whether an input sample is adversarial. If the input sample is determined as non-adversarial, the output of the text classifier (Label A) is directly used as its final prediction. If the input sample is determined as adversarial, the frequency-aware randomization process is applied to the original input sample. Then, the randomized sample is sent to the PrLM again, and the second output of the text classifier (Label B) is used as its final prediction.

4 Experimental Implementation

4.1 Tasks and Datasets

Experiments are conducted on two major NLP tasks: text classification and natural language inference. The dataset statistics are displayed in Table 1. We evaluate the performance of models on the non-adversarial test samples as the original accuracy. Then we measure the after-attack accuracy of models when facing AWS. By comparing these two accuracy scores, we can evaluate how robust the model is.

Task	Dataset	Train	Test	Avg Len
Classification	MR	9K	1K	20
	SST2	67K	1.8K	20
	IMDB	25K	25K	215
Entailment	MNLI	433K	10K	11

Table 1: Dataset statistics.

Text Classification We use three text classification datasets with average text lengths from 20 to 215 words, ranging from phrase-level to document-level tasks. **SST2** (Socher et al., 2013): phrase-level binary sentiment classification using fine-grained sentiment labels on movie reviews. **MR** (Pang and Lee, 2005): sentence-level binary sentiment classification on movie reviews. We take 90% of the data as training set and 10% of the data as test set as (Jin et al., 2019). **IMDB** (Maas et al., 2011): document-level binary sentiment classification on movie reviews.

Natural Language Inference NLI aims at determining the relationship between a pair of sentences based on semantic meanings. We use Multi-Genre Natural Language Inference (**MNLI**) (Nangia et al., 2017), a widely adopted NLI benchmark with coverage of transcribed speech, popular fiction, and government reports.

4.2 Attack Model and Baselines

We use TextFooler[†](Jin et al., 2019) as the major attack model for AWS. Moreover, we implement (Ren et al., 2019) and GENETIC (Alzantot et al., 2018) based on the TextAttack (Morris et al., 2020) code base to further verify the efficiency of our proposed method.

We compare ADFAR with DISP (Zhou et al., 2019) and SAFER (Ye et al., 2020). The implementation of DISP is based on the repository offered by Zhou et al. (2019). For SAFER, we also leverage the code proposed by Ye et al. (2020). Necessary modifications are made to evaluate these methods' performance under heuristic attack models.

4.3 Experimental Setup

The implementation of PrLMs is based on PyTorch[‡]. We leverage, BERT$_{\text{BASE}}$ (Devlin et al., 2019), RoBERTa$_{\text{BASE}}$ (Liu et al., 2019) and ELECTRA$_{\text{BASE}}$ (Clark et al., 2020) as baseline

---

[†] https://github.com/jind11/TextFooler

[‡] https://github.com/joey1993/bert-defender

[§] https://github.com/lushleaf/Structure-free-certified-NLP

Model	MR		SST2		IMDB		MNLI	
	Orig. Acc.	Adv. Acc.	Orig. Acc.	Adv. Acc.	Orig. Acc.	Adv. Acc.	Orig. Acc.	Adv. Acc.
BERT	86.2	16.9	**93.1**	39.8	92.4	12.4	**84.0**	11.3
BERT + Adv Training	85.6	34.6	92.6	48.8	92.2	34.2	82.3	33.4
BERT + DISP	82.0	42.2	91.6	70.4	91.7	82.0	76.3	35.1
BERT + SAFER	79.0	55.4	91.3	75.6	91.3	88.1	82.1	54.7
BERT + ADFAR	**86.6**	**66.0**	92.4	**75.6**	**92.8**	**89.2**	82.6	**67.8**

Table 2: The performance of ADFAR and other defense frameworks using BERT$_{\text{BASE}}$ as PrLM and TextFooler as attack model. Orig. Acc. is the prediction accuracy of normal samples and Adv. Acc. is the after-attack accuracy of models when facing AWS. The results are based on the average of five runs.

PrLMs. We use AdamW (Loshchilov and Hutter, 2018) as our optimizer with a learning rate of 3e-5 and a batch size of 16. The number of epochs is set to 5.

For the frequency-aware randomization process, we set $f_{thres} = 200$, $n_s = 20$ and $n_f = 10$. In the adopted frequency dictionary, 5.5k out of 50k words have a frequency lower than $f_{thres} = 200$ and therefore regarded as rare words. r is set to different values for training (25%) and inference (30%) due to different aims. In training, to avoid introducing excessive noise and reduce the prediction accuracy for non-adversarial samples, r is set to be relatively low. On the contrary, in inference, our aim is to perplex the heuristic attack mechanism. The more randomization we add, the more perplexities the attack mechanism receives, therefore we set a relatively higher value for r. More details on the choice of these hyperparameters will be discussed in the analysis section.

5 Experimental Results

5.1 Main results

Following (Jin et al., 2019), we leverage BERT$_{\text{BASE}}$ (Devlin et al., 2019) as baseline PrLM and TextFooler as attack model. Table 2 shows the performance of ADFAR and other defense frameworks. Since randomization may lead to a variance of the results, we report the results based on the average of five runs. Experimental results indicate that ADFAR can effectively help PrLM against AWS. Compared with DISP (Zhou et al., 2019) and SAFER (Ye et al., 2020), ADFAR achieves the best performance for adversarial samples. Meanwhile, ADFAR does not hurt the performance for non-adversarial samples in general. On tasks such as MR and IMDB, ADFAR can even enhance the baseline PrLM.

DISP leverages two extra PrLMs to discriminate and recover the perturbed tokens, which introduce extra complexities. SAFER makes the prediction of an input sentence by averaging the prediction results of its perturbed alternatives, which multiply the inference time. As shown in Table 3, compared with previous methods, ADFAR achieves a significantly higher inference speed.

Model	Parameters	Inference Time
BERT$_{\text{BASE}}$	110M	15.7ms (100%)
BERT$_{\text{BASE}}$ + DISP	330M	38.9ms (247%)
BERT$_{\text{BASE}}$ + SAFER	110M	27.6ms (176%)
BERT$_{\text{BASE}}$ + ADFAR	110M	18.1ms (115%)

Table 3: Parameters and Inference Time statistics. The inference time indicate the average inference time for one sample in MR dataset using one NVIDIA RTX3090.

5.2 Results with Different Attack Strategy

Since ADFAR leverages the adversarial samples generated by TextFooler (Jin et al., 2019) in training, it is important to see whether ADFAR also performs well when facing adversarial samples generated by other AWS models. We leverage PWWS (Ren et al., 2019) and GENETIC (Alzantot et al., 2018) to further study the performance of ADFAR.

Attack	MR		SST2	
	BERT	+ADFAR	BERT	+ADFAR
Attack-Free	86.2	86.6	93.1	92.3
PWWS	34.2	74.2	54.3	80.5
Genetic	21.3	70.4	38.7	72.2
TextFooler	16.9	66.0	39.8	73.8

Table 4: Performance of BERT and BERT with ADFAR when facing various AWS models. The results are based on the average of five runs.

As shown is Table 4, the performance of ADFAR is not affected by different AWS models, which further proves the efficacy of our method.

[¶]https://github.com/huggingface

5.3 Results with Other PrLMs

Table 5 shows the performance of ADFAR leveraging RoBERTa$_{\text{BASE}}$ (Liu et al., 2019) and ELECTRA$_{\text{BASE}}$ (Clark et al., 2020) as PrLMs. In order to enhance the robustness and performance of the PrLM, RoBERTa extends BERT with a larger corpus and using more efficient parameters, while ELECTRA applies a GAN-style architecture for pre-training. Empirical results indicate that ADFAR can further improve the robustness of RoBERTa and ELECTRA while preserving their original performance.

PrLM	MR		SST2	
	Orig. Acc.	Adv. Acc.	Orig. Acc.	Adv. Acc.
BERT	86.2	16.9	93.1	39.8
+ADFAR	86.6	66.0	92.3	73.8
RoBERTa	88.3	30.4	93.4	37.4
+ADFAR	87.2	71.0	93.2	77.6
ELECTRA	90.1	33.6	94.2	40.4
+ADFAR	90.4	71.2	95.0	83.0

Table 5: Results based with various PrLMs.

6 Analysis

6.1 Ablation Study

ADFAR leverages three techniques to help PrLMs defend against adversarial samples: adversarial training, frequency-aware randomization and anomaly detection. To evaluate the contributions of these techniques in ADFAR, we perform ablation studies on MR and SST2 using BERT$_{\text{BASE}}$ as our PrLMs, and TextFooler as the attack model. As shown in Table 6, the frequency-aware randomization is the key factor which helps PrLM defense against adversarial samples, while anomaly detection plays an important role in preserving PrLM's prediction accuracy for non-adversarial samples.

Model	MR		SST2	
	Orig. Acc.	Adv. Acc.	Orig. Acc.	Adv. Acc.
BERT	86.2	16.9	93.1	39.8
+ Adv	85.6	34.6	92.6	48.8
+ FR	85.0	72.8	90.6	82.6
+ AD	86.6	66.0	92.3	73.8

Table 6: Ablation study on MR and SST2 using BERT$_{\text{BASE}}$ as PrLM, and TextFooler as attack model. Adv represents adversarial training, FR indicates frequency-aware randomization and AD means anomaly detection. The results are based on the average of five runs.

6.2 Anomaly Detection

In this section, we compare the anomaly detection capability between ADFAR and DISP (Zhou et al., 2019). ADFAR leverages an auxiliary anomaly detector, which share a same PrLM with the original text classifier, to discriminate adversarial samples. DISP uses an discriminator based on an extra PrLMs to identify the perturbed adversarial inputs, but on token level. For DISP, in order to detect anomaly on sentence level, input sentences with one or more than one adversarial tokens identified by DISP are regarded as adversarial samples. We respectively sample 500 normal and adversarial samples from the test set of MR and SST to evaluate the performance of ADFAR and DISP for anomaly detection.

Table 7 shows the performance of ADFAR and DISP for anomaly detection. Empirical results show that ADFAR can predict more precisely, since it achieves a significantly higher F_1 score than DISP. Moreover, ADFAR has a simpler framework, as its anomaly detector shares the same PrLM with the classifier, while DISP requires an extra PrLM. The results also indicate that the current heuristic AWS strategy is vulnerable to our anomaly detector, which disproves the claimed undetectable feature of this very adversarial strategy.

Method	MR			SST2		
	Precision	Recall	F_1	Precision	Recall	F_1
DISP	68.0	92.0	73.2	59.5	94.2	72.9
ADFAR	90.1	84.0	86.9	88.0	90.0	88.9

Table 7: Performance for anomaly detection.

6.3 Effect of Randomization Strategy

As the ablation study reveals, the frequency-aware randomization contributes the most to the defense. In this section, we analyze the impact of different hyperparameters and strategies adopted by the frequency-aware randomization approach, in inference and training respectively.

6.3.1 Inference

The frequency-aware randomization process is applied in inference to mitigate the adversarial effects. Substitution candidate selection and synonym set construction are two critical steps during this process, in which two hyperparameters (r and n_s) and the frequency-aware strategy are examined.

Selection of Substitution Candidates The influence of different strategies for substitution candidate selection in inference is studied in this section. The impact of two major factors are measured: 1) the substitution ratio r and 2) whether to apply a frequency-aware strategy. In order to exclude the disturbance from other factors, we train BERT on the original training set and fix n_s to 20. Firstly, we alter the value of r from 5% to 50%, without applying the frequency-aware strategy. As illustrated by the blue lines in Figure 4, as r increases, the original accuracy decreases, while the adversarial accuracy increases and peaks when r reaches 30%. Secondly, a frequency-aware strategy is added to the experiment, with $f_{thres} = 200$. As depicted by the yellow lines in Figure 4, both original and adversarial accuracy, the general trends coincide with the non-frequency-aware scenario, but overall accuracy is improved to a higher level. The highest adversarial is obtained when r is set to 30% using frequency-aware strategy.

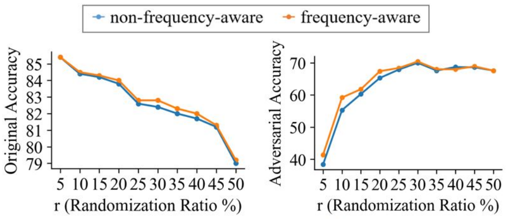

(a) Results on normal samples (b) Results on adversarial samples

Figure 4: Effect of the substitution ratio r and the frequency-aware strategy in substitution candidate selection during inference.

Construction of Synonym Set The influence of different strategies for synonym set construction in inference is evaluated in this section. The impact of two major factors are measured: 1) the size of a single synonym set n_s and 2) whether to apply a frequency-aware strategy. In order to exclude the disturbance from other factors, we train BERT on the original training set and fix r to 30% . Firstly, we alter the value of n_s from 5 to 50, without applying the frequency-aware strategy. The resulted original and adversarial accuracy are illustrated by the blue lines in Figure 5. Secondly, a frequency-aware strategy is added to the experiment, with $n_f = 50\% * n_s$. As depicted by the yellow lines in Figure 5, the original accuracy and the adversarial accuracy both peaks when $n_s = 20$, and the overall accuracy is improved to a higher level compared to the non-frequency-aware scenario.

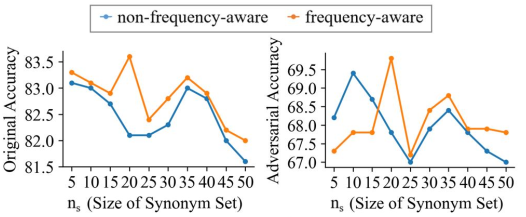

(a) Results on normal samples (b) Results on adversarial samples

Figure 5: Effect of the size of synonym set n_s and the frequency-aware strategy in construction of synonym set.

6.3.2 Training

The frequency-aware randomization process is applied in training to augment the training data, and hereby enables the PrLM to better cope with randomized samples inference. Based on this purpose, the frequency-aware randomization process in training should resemble the one in inference as much as possible. Therefore, here we set an identical process for synonym set construction, i.e. $n_s = 20$ and $n_f = 50\% * n_s$. However, for the substitution selection process, to avoid introducing excessive noise and maintain the accuracy for the PrLM, the most suitable substitution ratio r might be different than the one in inference. Experiments are conducted to evaluate the influence of r in training. We alter the value of r from 5% to 50%. In Figure 6, we observe that $r = 25\%$ results in highest original and adversarial accuracy.

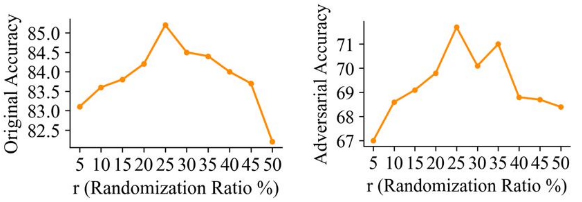

(a) Results on normal samples (b) Results on adversarial samples

Figure 6: Effect of the size of synonym set n_s and the frequency-aware strategy in construction of synonym set.

7 Conclusion

This paper proposes ADFAR, a novel framework which leverages the frequency-aware randomization and the anomaly detection to help PrLMs defend against adversarial word substitution. Empirical results show that ADFAR significantly outperforms those newly proposed defense methods over various tasks. Meanwhile, ADFAR achieves a remarkably higher inference speed and does not reduce the prediction accuracy for non-adversarial sentences, from which we keep the promise for this research purpose.

Comprehensive ablation study and analysis indicate that 1) Randomization is an effective method to defend against heuristic attack strategy. 2) Replacement of rare words with their more common alternative can help enhance the robustness of PrLMs. 3) Adversarial samples generated by current heuristic adversarial word substitution models can be easily distinguished by the proposed auxiliary anomaly detector. We hope this work could shed light on future studies on the robustness of PrLMs.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1310–1320.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. 2018. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1905.07129*.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4083–4093, Hong Kong, China. Association for Computational Linguistics.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using Google Web 1T 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1241–1249, Singapore. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142, Hong Kong, China. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932*.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis D. Griffin. 2020. Frequency-guided word substitutions for detecting textual adversarial examples. *arXiv preprint arXiv:2004.05887*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. In *RepEval*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021. Better robustness by more coverage: Adversarial training with mixup augmentation for robust fine-tuning.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on*

Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Wenqi Wang, Run Wang, Lina Wang, Zhibo Wang, and Aoshuang Ye. 2021. Towards a robust deep neural network in texts: A survey.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2018. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3465–3475, Online. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware bert for language understanding.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2019. Sg-net: Syntax-guided machine reading comprehension.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4904–4913, Hong Kong, China. Association for Computational Linguistics.

BERT-Proof Syntactic Structures:
Investigating Errors in Discontinuous Constituency Parsing

Maximin Coavoux
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
`maximin.coavoux@univ-grenoble-alpes.fr`

Abstract

The combined use of neural scoring systems and BERT fine-tuning has led to very high results in many natural language processing (NLP) tasks. These high results raise two important questions about the contribution and the limitations of pretrained-language models: (i) what are the remaining errors in the best-performing systems? (ii) what are the types of test examples where pretrained language models help the most? In this paper, we investigate both questions for the task of English discontinuous constituency parsing on the Penn Treebank, for which recent models obtain close to 95 F_1 score. To do so, we propose two methods for automatically analysing the errors of discontinuous parser. First, we annotate and release a test-suite focused on the syntactic phenomena responsible for discontinuities in the Penn Treebank, enabling us to obtain a per-phenomenon evaluation of a parser's output. Second, we extend the Berkeley Parser Analyser — a tool that classifies parsing errors according to predefined structural patterns —, to discontinuous trees. We apply both methods to characterize errors of a state-of-the-art transition-based discontinuous parser, and to provide an overview of the contribution of BERT to this task.

1 Introduction

Discontinous constituency trees are phrase-based syntactic representations where the constraint stating that a single phrase must yield a continuous sequence of tokens is lifted. Such representations are well-suited for modelling long-range dependencies, that typically arise for some syntactic phenomena, such as extractions or scrambling. For example, Figure 1 presents a discontinous VP modelling the relationship between the verb *want* and its extracted complement *How many*.

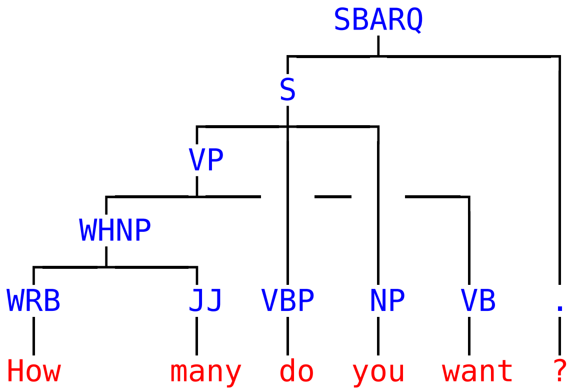

Figure 1: Discontinuous constituency tree.

In constituency treebanks, these long range dependencies are sometimes represented with typed empty categories (traces), coindexed with a displaced phrase (Marcus et al., 1993). However, projective parsers usually ignore them.[1] Indeed, the norm in Penn Treebank constituency parsing is to preprocess empty categories out of the corpus, leaving out important linguistic information.

Formally, discontinuous constituency trees interpret as derivations from mildly context-sensitive grammar formalisms, such as linear context-free rewriting systems (Vijay-Shanker et al., 1987, LCFRS) or multiple context-free grammars (Seki et al., 1991, MCFG). As a result, exact parsing of discontinuous structures has high computational complexity. For example, CKY-style parsing of an LCFRS is $\mathcal{O}(n^{3f})$ in time (Kallmeyer, 2010), where f is the fan-out of the grammar: the maximum number of spans in a grammar rule.[2]

The current state of the art for English discontinuous constituency parsing on the Discontinuous Penn Treebank has reached 94.8% F_1 score (Corro, 2020) obtained by a span-based chart parser that combines a neural scoring system and pretrained contextualized embeddings (Devlin et al., 2019,

---

[1] Except for some work on parsing traces, e.g. Gabbard et al. (2006); Kummerfeld and Klein (2017).

[2] If $f = 1$, the grammar is equivalent to a CFG.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3259–3272
August 1–6, 2021.

BERT). However, such a high score can be misleading. Despite ensuring comparability across different parsers, the exclusive use of classical evaluation metrics (F-score, precision, recall, as is standard) is hard to interpret and does not disclose information about the syntactic capabilities of a parser. In discontinuous parsing, the standard evaluator `discodop`[3] (van Cranenburgh et al., 2016) provides metrics that only focus on discontinuous constituents (discontinuous F-score, discontinuous precision, discontinuous recall). However, these scores aggregate information across many distinct syntactic phenomena.

In this paper, we propose to automatically analyse the errors of discontinuous English parsers in order to provide a fine-grained overview of their current limitations. To do so, we pursue two complementary approaches. First, we construct a test suite focused on 6 syntactic phenomena responsible for the discontinuities in the Discontinuous Penn Treebank (Evang, 2011). Second, we adopt an error-correction based approach: we search for a sequence of error-correcting tree modifications that lead from the predicted tree to the gold tree, and classify the sequence of tree modifications based on structural patterns. This is a direct extension of Berkeley Parser Analyser (Kummerfeld et al., 2012) to English discontinuous parsing.

A secondary motivation for this work is to characterize the contribution of BERT to discontinuous constituency parsing. An active current line of research consists in assessing the syntactic knowledge learned by language models (Linzen et al., 2016; Marvin and Linzen, 2018; Gulordava et al., 2018), including those with structural supervision (Kuncoro et al., 2018; Wilcox et al., 2019; Hu et al., 2020). They usually do so by constructing test items: minimal pairs of sentences, such that one is grammatical and the other is not (thus isolating a single grammatical constraint). Then, they observe whether the language model assigns higher probability to the grammatical alternative. In these papers, the observation of the syntactic ability of the models is indirect. We argue that fine-grained evaluation methods will help comparing the syntactic capabilities of parsers when they have access to BERT or not, which will provide a complementary view to this line of research. Therefore, we apply both proposed error analyses methods to a state-of-the-art transition-based discontinuous parser in several settings: without pretraining, with fast-text embeddings (Mikolov et al., 2018a; Grave et al., 2018), with BERT pretraining.

In summary, we make the following contributions:

- We construct a test-suite for automating a fine grained evaluation of English discontinuous parsers on target phenomena.
- We extend the Berkeley parser analyser to deal with English discontinuous constituency trees.
- We use these two evaluation methods to characterize the errors of a neural discontinuous parser, trained in several pretraining settings.

We provide the test suite and the error analyser as supplementary material.

2 Related Work

To address the limitations of using exclusively an F-score to evaluate constituency parsers, prior work focused on alternative finer-grained evaluation methods. We review some of them, both from the projective and discontinuous constituency parsing litterature.

Manual error analysis For discontinuous constituency parsing, Evang (2011) performed manual error analysis by extracting discontinuous trees from the evaluation corpus, classifying them according to the phenomenon at the origin of the discontinuities, and manually checking if a PLCFRS chart-parser recognized them. Coavoux et al. (2019) used the same strategy to evaluate a neural transition-based discontinuous parser. However, manual error analysis is quite time-consuming and needs to be performed again for evaluating each new parser output. Thus, it is difficult to integrate it in an evaluation pipeline or to deploy it for many parsers.

Automatic error analysis Kummerfeld et al. (2012) introduced a method that consists in searching for a sequence of atomic tree-modifications (such as: inserting a node, removing a node, moving a node) that leads from a predicted constituency tree to the gold tree. Then, they classify the tree modifications according to predefined structural patterns, e.g., 'PP-attachment', 'NP-attachment', 'labelling error'. Their method led to identify most frequent patterns of error, and characterize the improvement obtained with techniques such as reranking. However, the structural patterns used to clas-

---

[3] `https://github.com/andreasvc/disco-dop/`

sify mistakes depend both on the language of the treebank and on its annotation strategies. Therefore, error patterns need to be designed when adapting the error analyser to another treebank (Kummerfeld et al., 2013). Moreover, their method and software do not handle discontinuous constituents, hence our proposal.

Targeted evaluation Another line of work on fine-grained parser evaluation focused on specific structures or phenomena. Ratnaparkhi et al. (1994) introduced a collection of English sentences with PP-attachment ambiguities, in order both to improve evaluation on this type of structure, and foster research on improving their resolution. Kübler et al. (2009) introduced a test suite for German that encompasses a wider range of syntactic structures (such as coordination of unlike constituents or extraposed relative clauses). However, they focus on projective constituency representations.

For discontinuous structures, Maier et al. (2014) released `discosuite`, a testsuite for German. They annotated a set of sentences from the Tiger corpus (Brants et al., 2004), with the syntactic phenomena responsible for the tree discontinuities. They released their annotations, such that researchers can run their parsers on the sentences and compute a per-phenomenon evaluation of the parser. To the best of our knowledge, such a test suite only exists for German. In this article, we introduce one for English, along with an evaluation script that provides per-phenomenon statistics.

We focus our analysis on English, since the resources we introduce are for this language. However, we also provide results on German using `discosuite`.

3 Test Suite Annotation

This section describes our methodology to annotate a set of discontinuous constituency trees with the syntactic phenomena responsible for the discontinuities. We first extract all discontinuous trees from the validation section of the discontinuous version of the Penn Treebank (Evang, 2011; Evang and Kallmeyer, 2011), except those for which the discontinuities are only due to punctuation attachment. We obtain 266 trees, which corresponds to 16% of the corpus. Then, we manually assign one or several categories from the following set, previously proposed for manual error analysis by Evang (2011) and reused by Coavoux et al. (2019). We provide an example sentence from the corpus for each category, with the main discontinuous constituent highlighted in bold:

1. *wh*-extraction: *[. . .] the most recent period* ***for which*** *results were* ***broken out*** *[. . .]*
2. circumpositioned quotation: ***While Mayor Norman found the market's performance Monday reassuring,*** *he says,* ***he remains uneasy.***
3. fronted quotations: ***The proposed changes "all make a lot of sense to me,"*** *he* ***added.***
4. *it*-extraposition: *While* ***it*** *is possible* ***that the Big Green initiative will be ruled unconstitutional*** *[. . .]*
5. discontinuous dependencies: *[. . .] provided little help for copper as* ***word*** *spread* ***that a three-month strike at the Highland Valley mine in British Columbia was about over*** *[. . .]*
6. subject-verb inversion: ***Added*** *another executive at a big bank:* ***"We were all a little goosey over the weekend trying to forecast what would happen Monday, but it's been very quiet.***

There are several subtypes of *wh*-extractions in the data: relative clauses, verbal adjunct clauses, complement clauses, indirect and direct questions. They all include a *wh* word among *how, when, which, that, where, what, why, whenever*. Circumpositioned and fronted phrases only include quotations, and systematically feature a speech verb, usually *says* or *said*. *It*-extrapositions feature an expletive *it* in the interpretation location of an extraposed clausal argument. The category of discontinuous dependencies contains other cases where a constituent is split by an intervening phrase. It mostly includes extraposed modifiers, such as the extraposed clause in example 5 above,

Not all occurrences of these phenomena result in a discontinuous tree (Evang, 2011). For example, a sentence containing both a fronted quotation and a subject-verb inversion will not result in a discontinuity. In some trees, there are also several occurrences of phenomena producing discontinuities. We release these annotations as a csv file, provided as supplementary material.

Per-phenomenon evaluation method In order to obtain a per-phenomenon evaluation of the predictions of discontinuous parsers, we first extract individual evaluations for each discontinuous tree, as provided by the standard evaluator for discontinuous parsing (van Cranenburgh et al., 2016, dis-

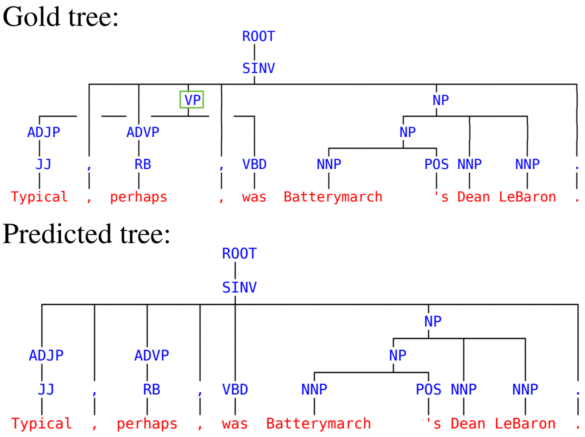

Figure 2: Error correction: add a missing node.

codop). These include the number of gold, correct, and incorrect discontinuous constituents, both in the labelled and unlabelled case.[4]

We consider that the annotated target phenomenon on the sentence is perfectly predicted if the sentence discontinuous F-score is 100, and partially predicted if it is > 0. As such, this evaluation is recall-oriented: we focus on how well the gold phenomena are predicted, but we do not take into account false positives (which would require us to assign a phenomenon to predicted trees with incorrect discontinuous constituents).

4 Error-Correction-Driven Analyser

We now focus on automatically classifying errors according to structural patterns. To do so, we build on Kummerfeld et al. (2012) and proceed in two steps: (i) finding a sequence of atomic tree modifications that transforms a predicted tree to the corresponding gold tree; (ii) classifying steps in the transformation sequence according to predefined structural patterns. For step (i), we use a greedy search algorithm, that first corrects errors on discontinuous nodes, and then backoffs to Kummerfeld et al. (2012)'s method for projective error correction. Thus, we focus in this section on *discontinuous* error corrections.

We use 4 atomic tree modifications:

i change label of discontinuous node;
ii create a discontinuous node;
iii delete a discontinuous node;
iv move a node, resulting in a discontinuity.

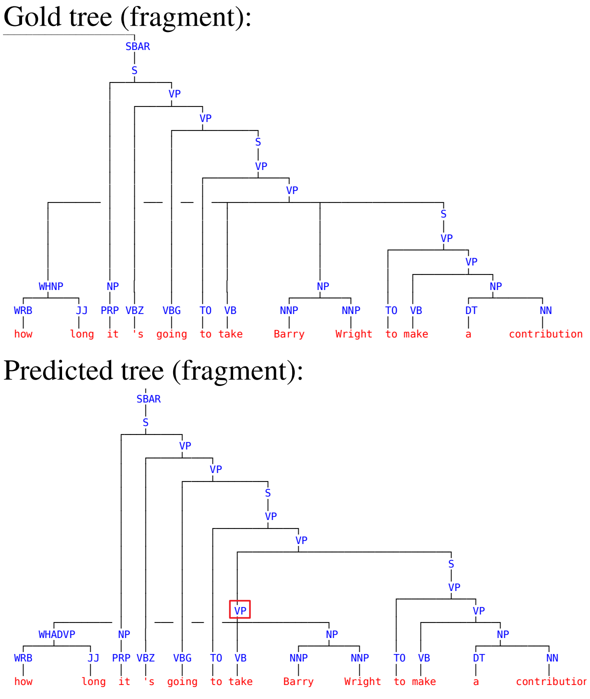

Figure 3: Error correction: remove an extra node.

The creation of a discontinuous node (ii) consists in gathering several nodes with the same parent, attaching them as the children of a new node, which is attached in turn to their original parents. For example, in Figure 2, the parser missed a VP node. The correction consists in creating a discontinuous VP node with two children (ADJP and VBD nodes) and attaching it to the SINV node.

To delete a discontinuous node (iii), we simply attach its children to their grandparent node. For example, in Figure 3, the children of the highlighted VP to be deleted (lower part) will be attached to the higher VP. For both node creation and node deletion, the corrected tree has only a single different node with the original predicted tree.

Finally, the moving of a node involves reattaching a node to a different parent. For example, in Figure 4, the correction of the predicted tree will consist in attaching the WHNP to the lowest VP, resulting in two missing discontinuous constituents (both VPs, see gold tree). A side effect is that the moving will also result in a unary S constituent that should be deleted in another correction step.

In order to find a sequence of error-correcting modifications, we perform a greedy search. While there is a false positive or a false negative discontinuous constituent in the current tree, we try to apply actions (i-iv) in this order of priority. Modifications (i-iii) cannot introduce new errors, whereas

---

[4] In the unlabelled case, we remove duplicate constituents (that correspond to unary rewrites) before evaluation as they are not interpretable. Therefore, it might happen that the labelled result is higher than the corresponding unlabelled one.

Gold tree (fragment):

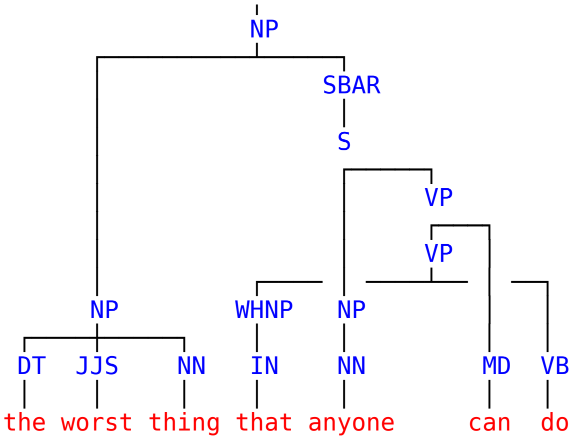

Predicted tree (fragment):

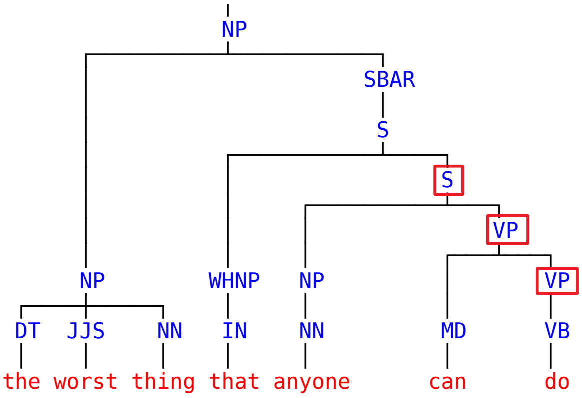

Figure 4: Error correction: move a node.

moving a node (iv) may do so in some cases.[5] We ensure that moving a node is only performed if the correction does not increase the gross number of errors by more than 1.

Once we have found the correction sequence, we classify errors according to patterns defined by Kummerfeld et al. (2012):

- PP attachment;
- NP internal structure;
- Modifier Attachment;
- Unary constituent;
- Different label;
- Clause attachement;
- Coordination;
- NP attachment;
- VP attachment.

Once all discontinuous errors are corrected, we run the original code from Kummerfeld et al. (2012) to compute statistics about projective constituent mistakes.

5 Parser

We use a Python reimplementation of the parser described by Coavoux et al. (2019), augmented with a mechanism to integrate and fine-tune BERT (Devlin et al., 2019). We release our code with pretrained models for replication purposes.[6]

The parser is based on a simple transition system (ML-GAP) that features the GAP action (Coavoux and Crabbé, 2017) to construct discontinuous constituents, and separates structural and labelling actions (Cross and Huang, 2016). The scoring system has two submodules:

- A sentence encoder that constructs contextualized embeddings for each token and is run before parsing;
- A feed-forward network that predicts the next action from the contextualized embeddings of tokens extracted from specific positions in the parsing configuration.

In the remainder of the paper, we call ML-GAP the baseline parser that has only access to the training corpus and has no pretrained parameters, ML-GAP+FT, when it has access to fasttext pretrained embeddings (Mikolov et al., 2018b), and ML-GAP+BERT, the parser that uses the `bert-base-cased` pretrained language model to compute token representations and finetunes it.

Token and sentence encoder The parsers differ in the way they represent the tokens $(w_1, w_2, \ldots w_n)$ in a sentence. The ML-GAP parser computes character-based word embeddings with a character bi-LSTM: $(\mathbf{c}_1, \ldots \mathbf{c}_n)$, where $\mathbf{c}_i = \text{bi-LSTM}(w_i)$, and concatenates them to word embeddings: $([\mathbf{c}_1, \mathbf{w}_1], \ldots [\mathbf{c}_n, \mathbf{w}_n])$. The ML-GAP+FT parser replaces learned word embeddings by (frozen) fast-text embeddings. The ML-GAP+BERT parser also uses a character bi-LSTM, but its output is concatenated with the contextualized embeddings from BERT: $([\mathbf{c}_1, \mathbf{b}_1], \ldots [\mathbf{c}_n, \mathbf{b}_n])$, where $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ is the output of the last layer BERT for the corresponding tokens. When BERT segments a token into several subtokens, we use the vector corresponding to the first subtoken. Alternative methods are available (using the last subtoken or an aggregation of the subtoken vectors) but they do not seem to have an effect on parsing (Kitaev et al., 2019).

---

[5] The effect of creation/deletion of a single node is purely local, whereas the moving of the node may impact many other nodes. For example in Figure 4, the moving changes the yield of 3 nodes.

[6] `https://gitlab.com/mcoavoux/mtgpy-release-findings-2021`, also archived at `http://doi.org/10.5281/zenodo.4775955`

Then, the token embeddings are fed to a bi-LSTM sentence encoder, as usually done in parsing (Stanojević and Alhama, 2017; Coavoux and Cohen, 2019; Corro, 2020; Stanojević and Steedman, 2020). In preliminary experiments, we alternatively used a self-attentive encoder (Vaswani et al., 2017), as done successfully in recent work in projective constituency parsing (Kitaev and Klein, 2018; Kitaev et al., 2019). However, it proved hard to optimize (high variance across experiments) and did not obtain better results than a bi-LSTM.

System	Dev		Test	
	F	DF	F	DF
Fully supervised				
$^\alpha$Evang and Kallmeyer (2011) < 25, gold POS			79	
$^\alpha$van Cranenburgh and Bod (2013) ≤ 40	85.2		85.6	
$^\alpha$van Cranenburgh et al. (2016) ≤ 40	86.9		87	
$^\delta$Corro et al. (2017)			89.2	
$^\beta$Coavoux and Cohen (2019)	91.4	70.9	90.9	67.3
$^\beta$Coavoux et al. (2019)	91.2	72.0	91.0	71.3
$^\gamma$Corro (2020)			**92.7**	64.2
$^\gamma$Stanojević and Steedman (2020)			90.5	67.1
$^\alpha$Ruprecht and Mörbitz (2021)			90.1	72.9
$^\beta$This work: ML-GAP	**92.0**	**75.9**	91.4	**74.4**
Semi-supervised (Pretrained embeddings)				
$^\gamma$Corro (2020)			**92.9**	64.9
$^\alpha$Ruprecht and Mörbitz (2021)			91.8	76.1
$^\beta$This work: ML-GAP+FT	92.7	78.1	92.3	**76.5**
Semi-supervised (Bert-base)				
$^\gamma$Corro (2020)			**94.8**	68.9
$^\delta$Vilares and Gómez-Rodríguez (2020)			91.7	49.1
$^\alpha$Ruprecht and Mörbitz (2021)			93.3	80.5
$^\beta$This work: ML-GAP+BERT	**95.0**	**85.8**	**95.0**	**82.5**

Table 1: Results on the Discontinuous Penn Treebank. DF: discontinuous F-score. $^\alpha$Grammar-based, $^\beta$transition-based, $^\gamma$chart-based, $^\delta$other neural systems.

Action scorer and features We use two distinct feed-forward networks to score respectively structural actions (SHIFT, MERGE, GAP) and labelling actions (NO-LABEL, {LABEL-X | X is a non-terminal}). They both have an identical architecture and only differ in the number of units in the output layer. We use a single hidden layer with a tanh activation. We apply dropout to its input, and layer normalization (Ba et al., 2016) to the hidden layer. We use a softmax normalization to compute scores for possible output labels.

	P	R	F	DP	DR	DF	POS
Dev corpus							
ML-GAP	92.0	91.9	92.0	82.2	70.4	75.9	97.3
ML-GAP+FT	92.7	92.7	92.7	84.2	72.9	78.1	97.4
ML-GAP+BERT	95.0	95.1	95.0	86.2	85.4	85.8	97.6
Test corpus							
ML-GAP	91.8	91.0	91.4	82.6	67.7	74.4	97.6
ML-GAP+FT	92.5	92.1	92.3	85.0	69.6	76.5	97.7
ML-GAP+BERT	95.2	94.8	95.0	85.3	79.9	82.5	97.9

Table 2: Detailed results on the development and test sets of the DPTB. P and R are precision and recall; DP, DR, DF are the discontinuous precision, recall and F-score.

The choice of the structural or labelling classifier is entirely determined by the parsing configuration and depends on the type of the next action. The input to both classifiers is the concatenation of contextualized vectors extracted from a list of positions in the parsing configuration and specified as a list of feature templates.

In the ML-GAP transition system, a parsing configuration is defined by 3 data structures: a stack s containing subtrees, a double-ended queue d also containing subtrees, and a buffer b containing the yet unprocessed tokens. We use the following 11 templates:[7]

- the left-most and right-most token of first and second element in s and d (8 templates in total: $s_0.l$, $s_0.r$, $s_1.l$, $s_1.r$, $d_0.l$, $d_0.r$, $d_1.l$, $d_1.r$);[8]
- the next token in the buffer (b_0);
- the contextualized embeddings corresponding to the start of sentence and end of sentence symbols.

Overall results We report overall development and test results in Table 1 (see Appendix A for details about training), and compare them to published results on the DPTB dataset. For a more comprehensive evaluation of the parser, including results on the Tiger (Brants et al., 2002)[9] and Negra (Skut et al., 1997) German corpora, we refer the reader to Table 7 of Appendix B.

The ML-GAP setting improves over Coavoux et al. (2019) by 0.4 and 3.1 respectively for the F and DF metrics, which we attribute to the hyperparameter search. In both the supervised setting and the 'pretrained embeddings' setting, our parser's results lag behind Corro (2020), the current state of the art. However it is noticeably more accurate on discontinuous constituents (more than 10 absolute DF difference).

In the BERT-finetuning setting, the F measure of the ML-GAP+BERT model slightly outperforms the span-based parser of Corro (2020). The use

---

[7] This is an extension over the set of 7 templates from Coavoux et al. (2019).

[8] We use s_i (resp. d_i) to address the $i-1^{\text{th}}$ subtree of s (resp. d), and $.l/.r$ to address the left-most or right-most token yielded by the subtree.

[9] We use the SPMRL split (Seddah et al., 2013).

		Labelled				
Phenomenon	Count	Exact match	Partial match	Precision	Recall	F_1
Extraction	91	85.7 (+13.2, +16.5)	92.3 (+7.7, +9.9)	91.6 (+5.6, +7.1)	88.1 (+11.2, +13.1)	89.8 (+8.6, +10.3)
Fronted quotation	71	95.8 (+2.8, +4.3)	95.8 (+1.4, +2.8)	95.8 (+1.4, +4.1)	95.8 (+1.4, +2.8)	95.8 (+1.4, +3.5)
Discontinuous dependency	37	64.9 (+37.9, +46.0)	70.3 (+40.6, +46.0)	84.4 (-7.3, -5.6)	61.4 (+36.4, +40.9)	71.1 (+31.8, +37.8)
Circumpositioned quotation	16	81.2 (+18.7, +18.7)	100.0 (+12.5, +6.2)	94.1 (-1.0, +1.1)	96.0 (+18.0, +16.0)	95.0 (+9.3, +9.0)
It-extraposition	12	75.0 (+0.0, +33.3)	75.0 (+0.0, +33.3)	90.0 (+0.0, +6.7)	75.0 (+0.0, +33.3)	81.8 (+0.0, +26.2)
Extraction+fronted quotation	7	100.0 (+0.0, +14.3)	100.0 (+0.0, +0.0)	100.0 (+0.0, +0.0)	100.0 (+0.0, +5.3)	100.0 (+0.0, +2.7)
Discontinuous dependency+extraction	5	60.0 (+40.0, +60.0)	100.0 (+20.0, +0.0)	88.2 (+9.6, +16.8)	88.2 (+23.5, +29.4)	88.2 (+17.2, +23.7)
Extraction+extraction	5	80.0 (+20.0, +20.0)	100.0 (+0.0, +0.0)	94.4 (-5.6, -5.6)	100.0 (+23.5, +23.5)	97.1 (+10.4, +10.4)
Subject inversion	5	100.0 (+40.0, +40.0)	100.0 (+40.0, +40.0)	100.0 (+0.0, +0.0)	100.0 (+40.0, +40.0)	100.0 (+25.0, +25.0)

		Unlabelled				
Phenomenon	Count	Exact match	Partial match	Precision	Recall	F_1
Extraction	91	85.7 (+12.1, +15.4)	92.3 (+7.7, +9.9)	92.4 (+5.5, +7.7)	89.3 (+10.0, +12.0)	90.8 (+7.9, +10.0)
Fronted quotation	71	95.8 (+2.8, +4.3)	95.8 (+1.4, +2.8)	95.8 (+1.4, +4.1)	95.8 (+1.4, +2.8)	95.8 (+1.4, +3.5)
Discontinuous dependency	37	64.9 (+35.2, +43.3)	70.3 (+37.9, +43.3)	83.9 (-16.1, -16.1)	60.5 (+32.6, +37.2)	70.3 (+26.7, +32.6)
Circumpositioned quotation	16	100.0 (+25.0, +25.0)	100.0 (+12.5, +6.2)	100.0 (+0.0, +3.1)	100.0 (+21.1, +18.4)	100.0 (+11.8, +11.4)
It-extraposition	12	75.0 (+0.0, +33.3)	75.0 (+0.0, +33.3)	90.0 (+0.0, +6.7)	75.0 (+0.0, +33.3)	81.8 (+0.0, +26.2)
Extraction+fronted quotation	7	100.0 (+0.0, +14.3)	100.0 (+0.0, +0.0)	100.0 (+0.0, +0.0)	100.0 (+0.0, +5.6)	100.0 (+0.0, +2.9)
Discontinuous dependency+extraction	5	60.0 (+40.0, +60.0)	100.0 (+20.0, +0.0)	92.9 (+17.9, +17.9)	86.7 (+26.7, +26.7)	89.7 (+23.0, +23.0)
Extraction+extraction	5	80.0 (+20.0, +20.0)	100.0 (+0.0, +0.0)	94.1 (-5.9, -5.9)	100.0 (+18.8, +18.8)	97.0 (+7.3, +7.3)
Subject inversion	5	100.0 (+40.0, +40.0)	100.0 (+40.0, +40.0)	100.0 (+0.0, +0.0)	100.0 (+40.0, +40.0)	100.0 (+25.0, +25.0)

Table 3: Per-phenomenon results on the test suite for the ML-GAP+BERT model. Its absolute improvements over respectively the ML-GAP+FT and the ML-GAP model are in parentheses. We handle separately sentences that have several occurrences of phenomena resulting in discontinuities (+ symbol), we exclude combinations with fewer than 5 occurrences.

of BERT seems to cancel the benefits of the exact decoding permitted by the span-based approach of Corro (2020). On the DF metric, the gap is even larger (13.6 absolute difference). We attribute this difference to the fact that Corro (2020)'s parser is restricted to a certain type of discontinuities and cannot construct certain trees. Moreover, the transition-based paradigm enables a parser to use more fine-grained features than span-based parsers, which is particularly helpful for predicting discontinuous constituents.

6 Results and Discussion

In this section, we focus on the comparisons of our 3 models to assess the contribution of BERT to discontinuous parsing. We first focus on English, using the two resources we introduced in Sections 3 and 4. Then we provide and discuss results on German, using `discosuite` (Maier et al., 2014).

6.1 English

The improvements brought by BERT may come from its syntactic knowledge. However, they might also be a result of its extended lexical knowledge (providing more lexical information about out-of-vocabulary or rare words that might be known but do not take part in discontinuous structures in the training set). The ML-GAP+FT model provides a control setting, where the 'static' pretrained embeddings provide additional lexical information.

Overall effect of pretraining We provide detailed results (precision, recall, F) in Table 2. It had been reported that discontinuous parsers often have a large gap between precision (higher) and recall (lower) on discontinuities on both German and English corpora (Maier, 2015; Coavoux and Cohen, 2019; Stanojević and Steedman, 2020; Corro, 2020). The use of BERT tends to fill this gap, with a much stronger effect on recall (+15 DR on development set over **ml-gap**) than on precision (+4 DP). BERT leads the parser to better detect syntactic discontinuities compared to a supervised model (ML-GAP).

On the contrary, ML-GAP+FT provides only a small improvement over ML-GAP (+2.2 dev DF), which is split almost equally between precision (+2.0 DP) and recall (+2.5 DR). The striking difference between ML-GAP+FT and ML-GAP+BERT strongly suggests that BERT's contribution cannot be reduced to its extended lexical knowledge.

Per-phenomenon evaluation We report results on the test suite in Table 3, in the labelled case (upper part) and the unlabelled case (lower part). For each metric, we report the result of the ML-GAP+BERT model, as well as its absolute difference with, respectively the ML-GAP+FT and the ML-GAP model.

First, when comparing ML-GAP+BERT and ML-GAP, we observe a large improvement on all phenomena and almost all metrics. When comparing

	ML-GAP+BERT		ML-GAP+FT		ML-GAP	
Error type	Count	Nodes	Count	Nodes	Count	Nodes
PP Attachment	275 (-40.6%)	609 (-39.8%)	440 (-5.0%)	980 (-3.2%)	463	1012
⌞ discontinuous	17 (-34.6%)	27 (-34.1%)	26 (+0.0%)	40 (-2.4%)	26	41
Single Word Phrase	257 (-25.7%)	318 (-22.6%)	310 (-10.4%)	376 (-8.5%)	346	411
Unclassified	200 (-47.2%)	263 (-48.8%)	318 (-16.1%)	427 (-16.9%)	379	514
NP Internal Structure	191 (-26.8%)	221 (-32.4%)	218 (-16.5%)	283 (-13.5%)	261	327
⌞ discontinuous	1 (+∞%)	1 (+∞%)	1 (+∞%)	2 (+∞%)	0	0
Modifier Attachment	186 (-23.8%)	344 (-24.4%)	221 (-9.4%)	408 (-10.3%)	244	455
⌞ discontinuous	9 (-30.8%)	13 (-27.8%)	10 (-23.1%)	15 (-16.7%)	13	18
Unary	185 (-41.1%)	185 (-41.1%)	271 (-13.7%)	271 (-13.7%)	314	314
⌞ discontinuous	2 (+0.0%)	2 (+0.0%)	2 (+0.0%)	2 (+0.0%)	2	2
Different label	166 (-22.4%)	335 (-22.1%)	211 (-1.4%)	426 (-0.9%)	214	430
⌞ discontinuous	6 (+0.0%)	15 (+7.1%)	6 (+0.0%)	16 (+14.3%)	6	14
Clause Attachment	116 (-37.6%)	292 (-37.7%)	170 (-8.6%)	450 (-4.1%)	186	469
⌞ discontinuous	16 (-40.7%)	25 (-34.2%)	31 (+14.8%)	44 (+15.8%)	27	38
Co-ordination	84 (-45.8%)	210 (-47.2%)	127 (-18.1%)	312 (-21.6%)	155	398
⌞ discontinuous	3 (-25.0%)	8 (+14.3%)	4 (+0.0%)	8 (+14.3%)	4	7
NP Attachment	51 (-51.9%)	168 (-43.6%)	94 (-11.3%)	312 (+4.7%)	106	298
⌞ discontinuous	9 (-55.0%)	19 (-20.8%)	14 (-30.0%)	18 (-25.0%)	20	24
VP Attachment	19 (-63.5%)	62 (-67.7%)	40 (-23.1%)	135 (-29.7%)	52	192
⌞ discontinuous	2 (-71.4%)	2 (-86.7%)	7 (+0.0%)	13 (-13.3%)	7	15
XoverX Unary	10 (+0.0%)	10 (+0.0%)	10 (+0.0%)	10 (+0.0%)	10	10

Table 4: Error types for both models (development set). Absolute differences with the ML-GAP model are given in parentheses.

labelled and unlabelled results, we observe very small differences (< 1) except for the case of circumpositioned quotations. This is due to some cases where the discontinuous quotation phrase has an unfrequent label (FRAG or SINV). Overall, subject inversions,[10] fronted quotations and circumpositioned quotations are almost perfectly detected by the ML-GAP+BERT system with DF scores over 95, and high exact match (at least in the unlabelled case for circumpositioning). On the other hand, discontinuous dependencies and *it*-extrapositions, and to a lower extent extractions, have DF scores below 90, despite the huge effect of BERT (respectively +35.2 and +25 absolute improvement on exact match for discontinuous dependencies and *it*-extrapositions).

Secondly, the improvement brought by fast-text embeddings is consistently very small (around +2F), except on 2 types of phenomena: *it*-extrapositions (where BERT does not improve over fast-text), and to a lower extent discontinuous dependencies (+6F for fast-text, +37.8F for BERT). This result suggests that the difficulty to parse these phenomena stemmed, at least partly from a lack of lexical knowledge.

Error Analysis We report results of the error type classifier for both models in Table 4. For each error type, we report (i) the overall count of occurrences, (ii) the number of occurrences where the correction involved a discontinuous node among them, and (iii) the total number of nodes involved (a single error can cause multiple wrong nodes), as done by Kummerfeld et al. (2012).

Overall, we observe an important decrease across all types of errors, with error reduction rates often close to 40% (e.g. 45% fewer occurrences of PP attachment errors) for ML-GAP+BERT and around 20% for ML-GAP+FT.

The picture is slightly different if we look at errors involving discontinuous constituents. Indeed, the use of BERT drastically reduces the main sources of errors (PP/VP/NP attachment, modifier attachment, coordination), while having no effect on other types of structure (NP internal structure, unary constituent, label). In contrast, fast-text only improves modifier and NP attachments, and even introduces clause attachment errors.

6.2 German

In order to provide additional context to our results on English, we further experiment with the same parsing models on German, using the test-suite built by Maier et al. (2014) on the German Tiger corpus. They constructed this test-suite by first identifying and classifying discontinuous phenomena in the 1500 first sentences of the Tiger corpus; and then they selected 15[11] sentences for each identified phenomenon. In total, `discosuite` contains 180 occurrences across 151 sentences. Each occurrence corresponds to a single discontinuous constituent.

We train our parsers on a modified version of the SPMRL Tiger split, where the 151 sentences are removed from the training set. We then parse the 151 sentences and use the labelled and unlabelled recall on target constituents to evaluate the corresponding phenomena. We provide results on the testsuite in Table 5, using the same settings as in English (ML-GAP, ML-GAP+FT, ML-GAP+BERT). We refer the reader to Maier et al. (2014) for the descriptions of specific phenomena. To the best of our knowledge, no prior parsing work used this testsuite for evaluation since its release.

Due to a finer-grain classification, there are only few instances for each type. Hence, we only comment on general patterns. Overall, fast-text provides small improvement on 6 types of phenomena (over 14). In constrast, BERT improves on every type of phenomenon, with largest increases for extrapositions of an element of a coordination, extrapositions involving a focus adverb (eg. adverb in the main clause modifying a subordinate clause), and local movement (involves discontinuities that

---

[10] Note that we have a small sample for this phenomenon (5 instances).

[11] or fewer for rarer phenomena.

Phenomenon	Occurrences	Labelled recall			Unlabelled recall		
		ML-GAP+BERT (Δ)	ML-GAP+FT (Δ)	ML-GAP	ML-GAP+BERT (Δ)	ML-GAP+FT (Δ)	ML-GAP
Extraposed arguments	15	86.7 (+27)	66.7 (+7)	60	86.7 (+20)	66.7 (=)	66.7
Extraposed modifiers	15	73.3 (+13)	73.3 (+13)	60	73.3 (+7)	73.3 (+7)	66.7
Extraposition (comparison)	15	73.3 (+7)	66.7 (=)	66.7	80 (+7)	73.3 (=)	73.3
Extraposition (coordination)	15	40 (+33)	33.3 (+27)	6.7	60 (+47)	40 (+27)	13.3
Extraposition (focus adverb)	5	80 (+40)	40 (=)	40	80 (+20)	60 (=)	60
Local Movement (clause)	10	30 (+30)	0 (=)	0	30 (+30)	0 (=)	0
Local Movement (phrase)	10	40 (+30)	10 (=)	10	50 (+40)	10 (=)	10
Placeholder/repeated element	15	93.3 (+7)	80 (-7)	86.7	93.3 (+7)	80 (-7)	86.7
Parentheticals	15	86.7 (+13)	86.7 (+13)	73.3	93.3 (+7)	93.3 (+7)	86.7
Pronouns	15	73.3 (+13)	66.7 (+7)	60	73.3 (+13)	66.7 (+7)	60
Scrambling	15	60 (+13)	80 (+33)	46.7	60 (+13)	80 (+33)	46.7
Topicalization Other	10	10 (+10)	0 (=)	0	10 (+10)	10 (+10)	0
Topicalization VP HD	10	80 (+10)	50 (-20)	70	80 (+10)	50 (-20)	70
Topicalization VP mod/arg	15	93.3 (+20)	73.3 (=)	73.3	93.3 (+20)	73.3 (=)	73.3

Table 5: Results of ML-GAP+BERT, ML-GAP+FT and ML-GAP on `discosuite` (Maier et al., 2014). Absolute difference with ML-GAP model is indicated in parentheses.

do not cross clause boundaries). These are also the most difficult phenomena to predict correctly (< 50 recall).

We observe there is not a large difference between labelled and unlabelled scores, suggesting that finding the correct structure is the main difficulty.

7 Conclusion

We introduced two resources for fine-grained automatic error analysis of English discontinuous constituency parsers. First, we construct and release a test-suite for the range of syntactic phenomena responsible for the discontinuous structures in the discontinuous version of the Penn Treebank. Second, we extend the Berkeley parser analyser to the analysis of discontinuous constituency trees. We apply these resources to study the contribution of BERT to discontinuous parsing of English.

Overall, on almost all phenomena, BERT brings an improvement over a fast-text baseline. We found that BERT leads to almost perfect detection for some phenomena (subject inversion, fronted quotations, circumpositioned quotations). Moreover, there is still a wide room for improvement for extractions (despite the high frequency of this type of structures in the corpus), *it*-extrapositions, and discontinuous dependencies. In future work, we plan to address these limitations with targeted data-augmentation methods. We also plan to evaluate other pretrained language models to assess whether they exhibit the same error patterns as BERT.

Acknowledgments

I thank Sacha Beniamine, Caio Corro, Didier Schwab and 2 sets of 3 reviewers for feedback on previous versions of this paper. All trees are drawn with `discodop`.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 204–217, Minneapolis, Minnesota. Association for Computational Linguistics.

Maximin Coavoux and Benoît Crabbé. 2017. Incremental discontinuous phrase structure parsing with the GAP transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270, Valencia, Spain. Association for Computational Linguistics.

Maximin Coavoux, Benoît Crabbé, and Shay B. Cohen. 2019. Unlexicalized transition-based discontinuous constituency parsing. *Transactions of the Association for Computational Linguistics*, 7:73–89.

Caio Corro. 2020. Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from O(n^6) down to O(n^3). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2753–2764, Online. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1644–1654, Copenhagen, Denmark. Association for Computational Linguistics.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of the 13th International Conference on Parsing Technologies (IWPT 2013)*, pages 7–16, Nara, Japan. Assocation for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kilian Evang. 2011. Parsing discontinuous constituents in English. Master's thesis, University of Tübingen.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Discontinuous constituent parsing with pointer networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7724–7731. AAAI Press.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Multitask pointer network for multi-representational parsing. *arXiv preprint arXiv:2009.09730*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York City, USA. Association for Computational Linguistics.

Kilian Gebhardt. 2018. Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3049–3063, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kilian Gebhardt. 2020. Advances in using grammars with latent annotations for discontinuous parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 91–97, Online. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*, 1st edition. Springer Publishing Company, Incorporated.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, pages 1–15.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Sandra Kübler, Ines Rehbein, and Josef van Genabith. 2009. A testsuite for testing parser performance on complex german grammatical constructions. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories : Groningen, Netherlands, January 23-24, 2009*. MP.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Jonathan K. Kummerfeld and Dan Klein. 2017. Parsing with traces: An O(n4) algorithm and a structural representation. *Transactions of the Association for Computational Linguistics*, 5:441–454.

Jonathan K. Kummerfeld, Daniel Tse, James R. Curran, and Dan Klein. 2013. An empirical examination of challenges in Chinese parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 98–103, Sofia, Bulgaria. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Wolfgang Maier, Miriam Kaeshammer, Peter Baumann, and Sandra Kübler. 2014. Discosuite - a parser test suite for German discontinuous structures. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2905–2912, Reykjavik, Iceland. European Language Resources Association (ELRA).

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018a. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018b. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Thomas Ruprecht and Richard Mörbitz. 2021. Supertagging-based parsing with linear context-free rewriting systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2923–2935, Online. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191 – 229.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Fifth Conference on Applied Natural Language Processing*, pages 88–95, Washington, DC, USA. Association for Computational Linguistics.

Miloš Stanojević and Raquel G. Alhama. 2017. Neural discontinuous constituency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1666–1676, Copenhagen, Denmark. Association for Computational Linguistics.

Miloš Stanojević and Mark Steedman. 2020. Span-based LCFRS-2 parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 111–121, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yannick Versley. 2016. Discontinuity (re)^2-visited: A minimalist approach to pseudoprojective constituent parsing. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 58–69, San Diego, California. Association for Computational Linguistics.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Stanford, California, USA. Association for Computational Linguistics.

David Vilares and Carlos Gómez-Rodríguez. 2020. Discontinuous constituent parsing as sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2771–2785, Online. Association for Computational Linguistics.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3302–3312, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Training Details

The parser uses the pytorch library (Paszke et al., 2019) and the `transformers` interface (Wolf et al., 2020) to fine-tune BERT. For all 3 models (ML-GAP, ML-GAP+FT, ML-GAP+BERT), we only tuned the batch size and the learning rate. We report all hyperparameters and final configurations in Table 6 of Appendix A. We used the Adam optimizer (Kingma and Ba, 2015), divide learning rate by 2 when the model shows no improvement on the dev set for 5 epochs and stop training after 3 such cycles with no improvement. Finally, we maintain the average of parameters across iterations during training, and use the averaged parameters (instead of the final parameters) for the final model. We trained the parsers on gold sequences of configurations (teacher-forcing). We use a deterministic

Parser	ML-GAP/ML-GAP+FT	ML-GAP+BERT
Sentence encoder hyperparameters		
Character embedding	64	64
Character bi-LSTM state	256	128
Character dropout	0.2	0.2
Word embeddings	300	-
Word dropout	0.2/0	-
Sentence encoder	bi-LSTM	bi-LSTM
Sentence encoder state	400	400
Stacked bi-LSTM	2	2
Action scorer hyperparameters		
Activation	tanh	tanh
Hidden layers	1	1
Hidden size	250	250
Dropout	0.3	0.3
Optimization hyperparameters		
DPTB: batch size	{4, **8**}/{**4**, 8}	{16, **32**}
DPTB: learning rate	{0.001, **0.0015**}	{**0.00006**, 0.00008, 0.0001, 0.00012}
Tiger: batch size	{**4**, 8}/{4, **8**}	{16, **32**}
Tiger: learning rate	{0.001, **0.0015**}	{**0.00006**, 0.00008, 0.0001, 0.00012}
Negra: batch size	{**4**, 8}/{**4**, 8}	{16, **32**}
Negra: learning rate	{**0.001**, 0.0015}/{0.001, **0.0015**}	{**0.00006**, 0.00008, 0.0001, 0.00012}
DPTB: bert model	-	`bert-base-cased`
Negra, Tiger: bert model	-	`bert-base-german-cased`

Table 6: Hyperparameter search for all models (best configuration in bold). The '/' symbol indicates different final values for ML-GAP and ML-GAP+FT models.

oracle that prioritizes merges over shifts when both are possible, this implicitly corresponds to a left-binarization of n-ary constituents. Finally, we use greedy search for finding the best sequence of actions.

B Results on German Treebanks

System	Method type	DPTB		Tiger		Negra	
		F	DF	F	DF	F	DF
Fully supervised							
van Cranenburgh and Bod (2013), ≤ 40	GB	85.6		$75.3^{\neq}$		74.8	
Fernández-González and Martins (2015)	DB			77.3		77.0	
van Cranenburgh et al. (2016), ≤ 40	GB	87.0		$78.2^{\neq}$		76.8	
Maier (2015), gold POS	TB			74.7	18.8	77.0	19.8
Versley (2016)	GB			79.5			
Maier and Lichte (2016)	TB			76.5	16.3		
Corro et al. (2017)	DB	89.2					
Coavoux and Crabbé (2017)	TB			79.3			
Gebhardt (2018)	GB			75.1			
Coavoux and Cohen (2019)	TB	90.9	67.3	82.5	55.9	83.2	56.3
Coavoux et al. (2019)	TB	91.0	71.3	82.7	55.9	83.2	54.6
Fernández-González and Gómez-Rodríguez (2020)	DB			84.6	57.9	83.7	54.7
Stanojević and Steedman (2020)	CB	90.5	67.1	83.4	53.5	83.6	50.7
Corro (2020)	CB	**92.7**	64.2	85.5	53.8	86.2	54.1
Ruprecht and Mörbitz (2021)	GB	90.1	72.9	82.5	55.9	82.7	49.0
Fernández-González and Gómez-Rodríguez (2020)	DB			**86.6**	**62.6**	**86.8**	**69.5**
Gebhardt (2020)	GB			77.7	40.7	81.7	43.5
This work: ML-GAP	TB	91.4	**74.4**	82.9	57.4	82.3	55.6
Semi-supervised (Pretrained embeddings)							
Stanojević and Alhama (2017)	TB			77.0			
Corro (2020)	CB	**92.9**	64.9	85.2	51.2	86.3	56.1
Fernández-González and Gómez-Rodríguez (2020)	DB			**85.7**	60.4	**85.7**	58.6
Ruprecht and Mörbitz (2021)	GB	91.8	76.1	85.1	61.0	86.5	61.9
This work: ML-GAP+FT	TB	92.3	**76.5**	85.2	**61.1**	85.6	**60.9**
Semi-supervised (Bert-base)							
Corro (2020)	CB	94.8	68.9	90.0	62.1	91.6	66.1
Fernández-González and Gómez-Rodríguez (2020)	DB			89.8	71.0	91.0	**76.6**
Vilares and Gómez-Rodríguez (2020)	SL	91.9	50.8	84.6	51.1	83.9	45.6
Ruprecht and Mörbitz (2021)	GB	93.3	80.5	88.3	69.0	90.9	72.6
This work: ML-GAP+BERT	TB	**95.0**	**82.5**	**90.2**	**72.9**	**91.7**	73.3

Table 7: Results on the DPTB, Tiger and Negra corpora (test sets). DF: discontinuous F-score. Methods: GB: grammar-based, TB: transition-based, CB: grammarless chart-based, SL: sequence-labelling based, DB: dependency-conversion based. $\neq$: different train/dev/test split.

DoT: An efficient Double Transformer for NLP tasks with tables

Syrine Krichene[1], **Thomas Müller**[2,*], **Julian Martin Eisenschlos**[1]
Google Research, Zürich[1]
`{syrinekrichene,eisenjulian}@google.com`
Symanto Research, Valencia, Spain[2]
`thomas.mueller@symanto.com`

Abstract

Transformer-based approaches have been successfully used to obtain state-of-the-art accuracy on natural language processing (NLP) tasks with semi-structured tables. These model architectures are typically deep, resulting in slow training and inference, especially for long inputs. To improve efficiency while maintaining a high accuracy, we propose a new architecture, DoT, a double transformer model, that decomposes the problem into two sub-tasks: A shallow pruning transformer that selects the top-K tokens, followed by a deep task-specific transformer that takes as input those K tokens. Additionally, we modify the task-specific attention to incorporate the pruning scores. The two transformers are jointly trained by optimizing the task-specific loss. We run experiments on three benchmarks, including entailment and question-answering. We show that for a small drop of accuracy, DoT improves training and inference time by at least 50%. We also show that the pruning transformer effectively selects relevant tokens enabling the end-to-end model to maintain similar accuracy as slower baseline models. Finally, we analyse the pruning and give some insight into its impact on the task model.

1 Introduction

Recently, transfer learning with large-scale pretrained language models has been successfully used to solve many NLP tasks (Devlin et al., 2019; Radford et al., 2019; Liu et al., 2019). In particular, transformer models have been used to solve tasks that include semi-structured table knowledge, such as table question answering (Herzig et al., 2020) and entailment (Wenhu et al., 2019; Eisenschlos et al., 2020) – a binary classification task to support or refute a sentence based on the table's content.

While transformer models lead to significant improvements in accuracy, they suffer from high computation and memory cost, especially for large inputs. The total computational complexity per layer for self-attention is $O(n^2d)$ (Vaswani et al., 2017), where n is the input sequence length, and d is the embedding dimension. Using longer sequence lengths translates into increased training and inference time.

Improving the computational efficiency of transformer models has recently become an active research topic. To the best of our knowledge, the only technique that was applied to NLP tasks with semi-structured tables is heuristic pruning. Eisenschlos et al. (2020) show on the TABFACT data set (Wenhu et al., 2019) that using heuristic pruning accelerates the training time while achieving a similar accuracy. This raises the question of whether a better pruning strategy can be learned.

We propose to use DoT, a double transformer model (Figure 1): A first transformer – which we call *pruning transformer* – selects k tokens given a query and a table and a task-specific transformer solves the task based on these tokens. Decomposing the problem into two simpler tasks imposes additional structure that makes training more efficient: The first model is shallow, allowing the use of long input sequences at moderate cost, and the second model is deeper and uses the shortened input that solves the task. The combined model achieves a better efficiency-accuracy trade-off.

The pruning transformer is based on the TAPAS QA model (Herzig et al., 2020). TAPAS answers questions by selecting tokens from a given table. This problem is quite similar to the pruning task. The second transformer is a task-specific model adapted for each task to solve: We use another TAPAS QA model for QA and a classification model (Eisenschlos et al., 2020) for entailment. In Section 2, we explain how we jointly learn both models by incorporating the pruning scores into the attention mechanism.

DoT achieves a better trade-off between effi-

---

[*] Work done at Google Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3273–3283
August 1–6, 2021.

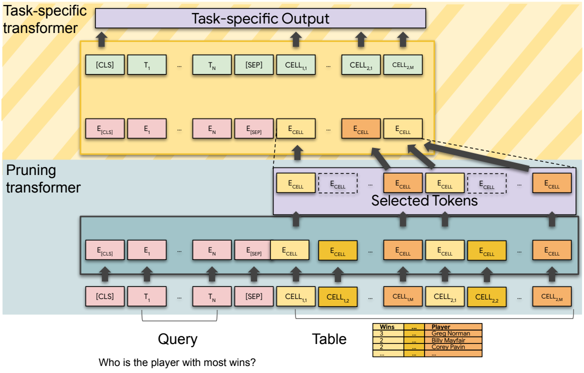

Figure 1: Pruning with a double transformer DoT. The pruning model selects the k most relevant tokens and passes them to the task model. The pruning model is small, allowing the use of long input sequences.

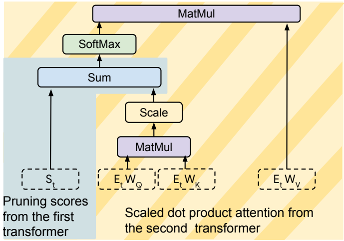

Figure 2: Scaled dot product attention of the task model. We change the attention architecture (Vaswani et al., 2017) – the dashed bloc – by adding the pruning scores – the solid bloc. The pruning scores affect the task model's attention in all layers. This enables back propagation for both models based on a single loss.

ciency and accuracy on three datasets. We show that the pruning transformer selects relevant tokens, resulting in higher accuracy for longer input sequences. We study the meaning of relevant tokens and show that the selection is deeply linked to solving the main task by studying the answer token scores. We open source the code in `http://github.com/google-research/tapas`.

2 The DoT Model

As show in Figure 1, the double transformer DoT is composed of two transformers: the *pruning transformer* selects the most relevant k tokens followed by a task-specific model that operates on the selected tokens to solve the task. The two transformers are learned jointly. DoT loss is detailed in Appendix A.2. We explore learning the pruning model using an additional loss in Appendix C.2.

Let q be the query (or statement) and T the table. The transformer takes as input the embedding $E = [E_{[CLS]}; E_q; E_{[SEP]}; E_T]$, composed of the query and table embeddings. The pruning transformer computes the probability $P(t|q,T)$ of the token t being relevant to solve the example. We derive the pruning score $s_t = \log(P(t|q,T))$ and keep the top-k tokens. The pruning scores are then passed to the task transformer as shown in Figure 2.

To enable the joint learning, we change the attention scores of the task model. For a normal transformer (Vaswani et al., 2017), given the input embedding E_t at position t, for each layer and attention head, the self-attention output is given by a linear combination of the value vector projections using the attention matrix.

Each row of the attention matrix is obtained by a softmax on the attention scores $z^{<t,t'>}$ given by

$$z^{<t,t'>} = \frac{E_t W_Q^\top (E_{t'} W_K^\top)^\top}{\sqrt{d_k}} \qquad (1)$$

where W_Q and W_K represent the query and key projections for that layer and head. In our task model we add a negative bias term and replace this equation with

$$z^{<t,t'>|s_t} = z^{<t,t'>} + s_t \qquad (2)$$

Thus, the attention scores provide a notion of token *relevance* – detailed in Appendix A.1 – and enable end-to-end learning of both models, letting DoT define the top-K tokens.

Unlike previous soft-masking methods (Bastings et al., 2019; De Cao et al., 2020), ours coincides exactly with removing the input token t when $P(t|q,T) \to 0$. We prove this formally in Appendix A.3.

We explore two different pruning strategies: token selection defined as discussed above and column selection where we average all token scores in each column.

3 Experimental Setup

We compare our approach against models using heuristic pruning.

Cell concatenation (CC) The TAPAS model uses a default heuristic to limit the input tokens. The objective of the algorithm is to fit an equal number of tokens for each cell. This is done by first selecting the first token from each cell, then the second and so on until the desired limit is reached.

Heuristic exact match (HEM) (Eisenschlos et al., 2020). This method scores the columns based on their similarity to the question, where similarity is defined by token overlap.

We introduce a notation to clarify the setup: $DoT(1^{st}_{type} \xrightarrow{top\text{-}k} 2^{nd}_{type})$. The *type* correspond to the model size: small (s), medium (m) or large (l) as defined in Turc et al. (2019). For example, $CC \xrightarrow{1024} DoT(s \xrightarrow{256} l)$ denotes a CC pre-processing to select 1024 tokens passed to the DoT model: one small pruning model that selects 256 tokens and feeds them into a large task model.

Baselines and DoT (hyper-parameters in Appendix B.1) are initialized from models pre-trained with a MASK-LM task, the intermediate pre-training data (Eisenschlos et al., 2020) and following Herzig et al. (2020) on SQA (Iyyer et al., 2017). The DoT transformers' complexity – detailed in Appendix B.3 – is similar to a normal transformer where only some constants are changed.

We evaluate DoT on three datasets.

WIKISQL (Zhong et al., 2017) is a corpus of $80,654$ questions with SQL queries, related to $24,241$ Wikipedia tables. Here we train and test in the weakly-supervised setting where the answer to the question is the result of the SQL applied to the table. The metric we use is denotation accuracy.

WIKITQ (Pasupat and Liang, 2015) consists of $22,033$ question-answer pairs on $2,108$ Wikipedia tables. The questions are complex and often require comparisons, superlatives or aggregation. The metric we use is the denotation accuracy as computed by the official evaluation script.

TabFact (Wenhu et al., 2019) contains 118K statements about 16K Wikipedia tables, labeled as either entailed or refuted. The dataset requires both linguistic reasoning and symbolic reasoning with operations such as comparison, filtering or counting. We use the classification accuracy as metric.

In all our experiments we report results for DoT using token selection for WIKISQL and TabFact and a column selection for WIKITQ.

4 Results

The baseline TAPAS model outperforms the previous state-of-the-art on all datasets (Table 1): $+2.1$ for WIKISQL ($CC \xrightarrow{1024}$ TAPAS(l)), $+1.07$ for TabFact ($HEM \xrightarrow{512}$ TAPAS(l)), and $+1.3$ for WIKITQ ($HEM \xrightarrow{1024}$ TAPAS(l)).

Efficiency accuracy trade-off Table 1 reports the accuracy test results along with the average number of processed examples per second NPE/s computed at training time. Using HEM as pre-processing step improves DoT models compared to CC for both WIKISQL and TabFact. $DoT(m)$ and $DoT(s)$ reach better efficiency accuracy trade-off for WIKISQL: with a small drop of accuracy by 0.4% (respectively 0.7%), they are 3.5 (respectively 4.6) times faster than the best baseline. For TabFact dataset, DoT is compared to a faster baseline than the one used for WIKISQL as it takes only 512 input tokens instead of 1024. $DoT(s)$ still achieves a good trade-off: with a decrease of 0.4% of accuracy it is 1.5 times faster. Unlike the previous datasets, WIKITQ is a harder task to solve and requires passing more data. By restricting $DoT(m)$ to select only 256 tokens we decrease the accuracy by a bigger drop 3.9% to be 3.5 times faster compared to $HEM \xrightarrow{1024}$ TAPAS(l).

Small task models The previous results, raise the question of whether a smaller task model can reach a similar accuracy. To answer this question, we compare $\xrightarrow{1024} DoT(s \xrightarrow{256} l)$ to $\xrightarrow{1024}$ TAPAS(s) and $\xrightarrow{256}$ TAPAS(l) in Table 2. DoT outperforms the smaller models showing the importance of using both transformers.

5 Analysis

Accuracy for long input sequences To study the long inputs, we bucketize the datasets per example input length. We compare $DoT(m \xrightarrow{256} l)$ to different $CC \dashrightarrow$ TAPAS(l) models in Table 3. For the bucket > 1024 the DoT model outperforms the 256 and 512 length baselines for all tasks. This indicates that the pruning model extracts two times more relevant tokens than the heuristic CC.

For the bucket $[512, 1024]$, we expect all models to reach a higher accuracy, as we expect lower loss of context than for the bucket > 1024 when applying CC. The results shows that DoT gives a similar accuracy to $\xrightarrow{512}$ TAPAS for WIKISQL and TabFact– in the margin error – and a slightly lower accuracy for WIKITQ: The pruning transformer selects only 256 top-K tokens compared to $\xrightarrow{512}$ TAPAS that selects twice more. Thus, the task-specific transformer has access to less tokens, therefore to possibly less context that can lead to an accuracy drop. This drop is small compared to $\xrightarrow{256}$ TAPAS baseline drop. DoT still outperforms $\xrightarrow{256}$ TAPAS for all datasets.

Pruning relevant tokens We inspect the pruning transformer on the WIKISQL and WIKITQ datasets, where the set of answer tokens is given.

Dataset	**WIKISQL**			**TabFact**			**WIKITQ**		
Model	**test accuracy**	**Best**	**NPE/s**	**test accuracy**	**Best**	**NPE/s**	**test accuracy**	**Best**	**NPE/s**
state-of-the-art	**83.9**			**81.0**			**51.8 ± 0.6**	**52.3**	
$CC \xrightarrow{256} \text{TAPAS}(l)$	76.4 ± 0.3	77.15	1870	75.1 ± 0.3	76.13	1900	44.8 ± 0.5	45.47	1900
$CC \xrightarrow{512} \text{TAPAS}(l)$	83.6 ± 0.1	83.65	800	81.3 ± 0.2	81.60	870	52.2 ± 0.5	52.74	810
$CC \xrightarrow{1024} \text{TAPAS}(l)$	**86.0 ± 0.3**	**86.6**	**270**	81.6 ± 0.1	81.64	300	53.9 ± 0.2	54.30	270
$CC \xrightarrow{1024} DoT(s \xrightarrow{256} l)$	74.2 ± 3.6	84.27	1250	81.0 ± 0.1	81.17	1300	48.1 ± 2.4	49.47	1250
$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	83.6 ± 0.5	84.67	950	79.0 ± 0.5	81.28	930	**50.1 ± 0.5**	**50.14**	**950**
$HEM \xrightarrow{256} \text{TAPAS}(l)$	77.4 ± 0.3	77.97	1870	75.5 ± 0.2	75.80	1900	47.3 ± 0.1	47.70	1900
$HEM \xrightarrow{512} \text{TAPAS}(l)$	83.8 ± 0.4	84.75	800	**82.0 ± 0.3**	**82.07**	**870**	52.7 ± 0.4	53.61	810
$HEM \xrightarrow{1024} \text{TAPAS}(l)$	85.9 ± 0.0	85.94	270	80.6 ± 0.0	80.6	300	**54.0 ± 0.9**	**54.93**	**270**
$HEM \xrightarrow{1024} DoT(s \xrightarrow{256} l)$	**85.3 ± 0.4**	**85.76**	**1250**	**81.6 ± 0.3**	**81.74**	**1300**	40.9 ± 0.2	41.23	1250
$HEM \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	**85.5 ± 0.2**	**85.82**	**950**	81.8 ± 0.0	81.94	930	40.1 ± 2.4	49.13	950

Table 1: Efficiency accuracy trade-off. We run DoT with token pruning for WIKISQL and TabFact and column pruning for WIKITQ. The state-of-the-art (detailed in Appendix B.2) corresponds to the models of Min et al. (2019) for WIKISQL, Eisenschlos et al. (2020) for TabFact and Yin et al. (2020) for WIKITQ. For each dataset, the state-of-the-art, the best baseline model on accuracy, and the DoT models that reach the best accuracy efficiency trade-off are highlighted.

Dataset	**WIKISQL**		**TabFact**		**WIKITQ**	
Model	**test accuracy**	**NPE/s**	**test accuracy**	**NPE/s**	**test accuracy**	**NPE/s**
$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	**83.6 ± 0.5**	950	**79.0 ± 0.9**	930	**50.1 ± 0.5**	950
$CC \xrightarrow{256} \text{TAPAS}(l)$	76.4 ± 0.3	1870	75.1 ± 0.3	1900	44.8 ± 0.5	1900
$CC \xrightarrow{1024} \text{TAPAS}(m)$	81.6 ± 0.2	2050	75.1 ± 0.2	2300	42.9 ± 0.3	2020

Table 2: Comparing DoT to smaller models similar to each of its two transformers.

Bucket	**Model**	**WIKISQL**	**TabFact**	**WIKITQ**
> 1024	$CC \xrightarrow{512} \text{TAPAS}(l)$	**24.3 ± 0.1**	**56.8 ± 2.2**	**18.8 ± 0.9**
	$CC \xrightarrow{256} \text{TAPAS}(l)$	5.8 ± 0.2	9.9 ± 1.5	6.9 ± 0.0
	$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	**40.1 ± 4.9**	**69.1 ± 2.5**	**23.8 ± 0.5**
$[512, 1024]$	$CC \xrightarrow{512} \text{TAPAS}(l)$	**73.6 ± 0.1**	**73.0 ± 0.3**	**42.7 ± 0.6**
	$CC \xrightarrow{256} \text{TAPAS}(l)$	40.9 ± 0.3	43.9 ± 0.4	18.6 ± 0.1
	$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	**72.9 ± 1.8**	**74.7 ± 0.7**	**39.1 ± 0.6**

Table 3: The denotation accuracy for test, computed over bucketized datasets per sequence length. The pruning transformer prunes efficiently –two times better: $DoT(\xrightarrow{256} l)$ reaches accuracy close and higher than using $CC \xrightarrow{512}$ heuristic with TAPAS(l).

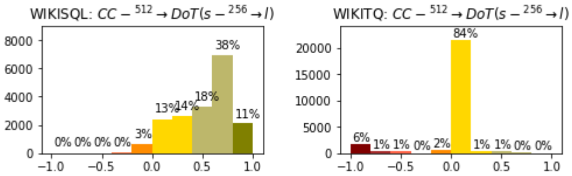

Figure 3: Distribution of the answer token scores minus the average scores of the top-K tokens. The difference is larger when the pruning transformer attributes a higher score to the answer tokens.

We compute the difference between the answer token scores and the average scores of the top-K tokens, and report the distribution in Figure 3. The pruning transformer tends to attribute high scores to the answer tokens, suggesting that it learns to answer the downstream question – a positive difference – especially for WIKISQL. The difference is lower for WIKITQ as it is a harder task: The set of answer tokens is larger, especially for aggregation, making their scores closer to the average.

Pruning transformer depth We study the pruning transformer complexity impact on the efficiency accuracy trade-off. Figure 4 compares the results of medium, small and mini models – complexity in Appendix B.3. For all datasets the *mini* model drops drastically the accuracy. The pruning transformer must be deep enough to learn the top-K tokens and attribute token scores that can be used by the task-specific transformer. For both WIKISQL and TabFact the small model reaches a better accuracy efficiency trade-off: Using a small instead of medium – 4 hidden layers instead of 8 – drops the accuracy by less than 0.4% – in the margin error – while accelerating the model times 1.3. In other words there is no gain of using a more complex model to select the top-K tokens especially when we restrict K to 256.

Restricting K can lead to a drop in the accuracy. Even by increasing the pruning complexity, DoT cannot recover the full drop. This is the case of WIKITQ. This dataset is more complex, it requires more reasoning including operation to run over multiple cells in one column. Thus selecting the top 256 tokens is a harder task compared to previous detests. We reduce the task complexity by using column selection instead of token selection. For this dataset using medium pruning transformer, $DoT(m)$ reaches a better accuracy efficiency trade-off: 2 points higher in accuracy compared to using a small transformer.

Effects of HEM and CC on DoT Table 1 and Figure 4 compare the effect of using HEM and

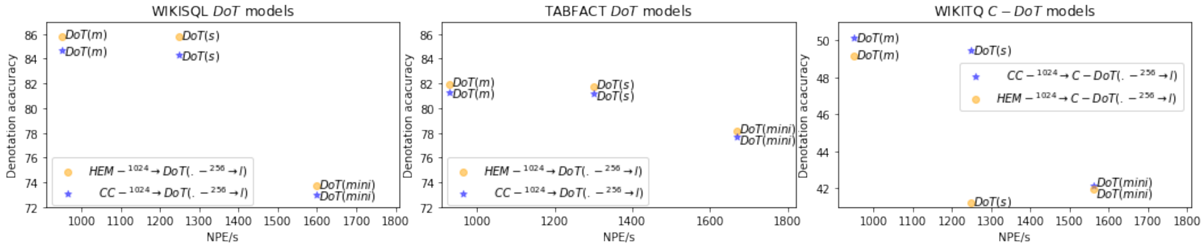

Figure 4: DoT models efficiency accuracy trade-off. The DoT models are displayed according to their accuracy in function of the average number of processed examples par second. The models are faster being closer to the right side of the figures and have higher accuracy being closer to the top. This figure compares the efficiency accuracy trade-off of using different pruning transformers – medium, small and mini – and study the impact of HEM and CC on DoT. We use token selection for both WIKISQL and TABFACT and column selection for WIKITQ.

CC on DoT models. As both heuristics are applied in the pre-processing step, using HEM or CC along with a similar DoT model, doesn't change the average number of processed examples per second NPE/s computed over the training step. For both WIKISQL and TABFACT we use a token based selection to select the top-K tokens. Combining the token based strategy with HEM, outperforms on accuracy the token pruning DoT combined with CC. For WIKITQ, the top-K pruning is a column based selection. Unlike the token selection the column pruning combined with HEM gives a lower accuracy.

6 Related work

Efficient Transformers Improving the computational efficiency of transformer models, especially for serving, is an active research topic. Proposed approaches fall into four categories. The first is to use knowledge distillation, either during the pre-training phase (Sanh et al., 2019), or for building task-specific models (Sun et al., 2019), or for both (Jiao et al., 2020). The second category is to use quantization-aware training during the fine-tuning phase of BERT models, such as (Zafrir et al., 2019). The third category is to modify the transformer architecture to improve the dependence on the sequence length (Choromanski et al., 2020; Wang et al., 2020). The fourth category is to use pruning strategies such as McCarley (2019), who studied structured pruning to reduce the number of parameters in each transformer layer, and Fan et al. (2020) who used structured dropout to reduce transformer depth at inference time. Our method most closely resembles the last category, but we focus our efforts on shrinking the sequence length of the input instead of model weights. Eisenschlos et al. (2020) explore heuristic methods based on lexical overlap and apply it to tasks involving tabular data, as we do, but our algorithm is learned end-to-end and more general in nature.

Interpretable NLP Another related line of work attempts to interpret neural networks by searching for *rationales* (Lei et al., 2016), which are a subset of words in the input text that serve as a justification for the prediction. Lei et al. (2016) learn the rationale as a latent discrete variable inside a computation graph with the REINFORCE method (Williams, 1992). Bastings et al. (2019) propose instead using stochastic computation nodes and continuous relaxations (Maddison et al., 2017), based on reparametrization (Diederik and Max, 2014) to approximate the discrete choice of a rationale from an input text, before using it as input for a classifier. Partially masked tokens are then replaced at the input embedding layer by some linear interpolation. We rely on a soft attention mask instead as a way to partially reduce the information coming from some tokens during training. To the best of our knowledge these methods have not been investigated in the context of semi-structured data such as tables or evaluated with a focus on efficiency.

7 Conclusion

We introduced double transformer (DoT) where an additional small model prunes the input of a larger second model. This accelerates the training and inference time at a low drop in accuracy. As future work we will explore hierarchical pruning and adapt DoT to other semi-structured NLP tasks.

Acknowledgments

We thank Yasemin Altun, William Cohen, Slav Petrov and the anonymous reviewers for their constructive feedback, comments and suggestions.

References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 130–140. PMLR.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *CoRR*, abs/2009.14794.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2669–2680, Minneapolis, Minnesota. Association for Computational Linguistics.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

P. Kingma Diederik and Welling Max. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations, ICLR 2014, Banff, Canada*.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations ICLR 2020, Virtual Conference, Formerly Addis Ababa ETHIOPIA*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, Conference Track Proceedings*.

J. S. McCarley. 2019. Pruning a bert-based question answering model. *CoRR*, abs/1910.06360.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2851–2864, Hong Kong, China. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC2) Co-located with NeurIPS 2019, Vancouver, Canada.*

Qi Shi, Yu Zhang, Qingyu Yin, and Ting Liu. 2020. Learn to combine linguistic and symbolic information for table-based fact verification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5335–5346, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3774–3785, Hong Kong, China. Association for Computational Linguistics.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Chen Wenhu, Wang Hongmin, Chen Jianshu, Zhang Yunkai, Wang Hong, Li Shiyang, Zhou Xiyou, and Yang Wang William. 2019. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations ICLR 2019, New Orleans.*

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. Program enhanced fact verification with verbalization and graph attention network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7810–7825, Online. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC2) Co-located with NeurIPS 2019, Vancouver, Canada.*

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020. LogicalFactChecker: Leveraging logical operations for fact checking with graph module network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6065, Online. Association for Computational Linguistics.

Appendix

A DoT model

A.1 Attention scores defines the meaning of relevant tokens.

We study, the updates of the pruning scores according to the attention scores needs. We note the set of relevant tokens R. The output probability given by the pruning transformer is in $(0, 1)$ making s_t in $(-\infty, 0)$. Lets suppose that the token t is not needed to answer the question, then the attention scores are decreased $z^{<i,t,t'>|s_t} \rightarrow -\vec{\infty}$ for all the tokens $t' \in R$ for all the layers i. The model updates both parts of $z^{<i,t,t'>}$ making s_t converging to $-\infty$, then $\lim_{s_t \rightarrow -\infty} z^{<i,t,t'>|s_t} = -\vec{\infty}$. Thus, the meaning of relevant token is defined by the attention scores updates: The pruning scores decreases for non relevant tokens and increase for relevant ones.

A.2 DoT loss

The DoT loss is similar to TAPAS model loss – noted as $J_{SA} = J_{aggr} + \beta J_{scalar}$ in (Herzig et al., 2020) – computed over the task-specific transformer where the attention scores are modified. More precisely, we modify only the scalar loss of the task specific model J_{scalar}. We incorporate the pruning scores $S = \{s_t \forall t \in T_{top_k=256}\}$, and we note $J_{scalar|S}$. The DoT loss is then compute only over the top-K tokens: $J_{DoT} = J_{aggr} + \beta J_{scalar|S}$.

For TABFACT dataset, Eisenschlos et al. (2020) modified the TAPAS loss – used for QA tasks – to adapt it to the entailment task: Aggregation is not used, instead, one hidden layer is added as output of the [CLS] token to compute the probability of Entailment. We use a similar loss for TABFACT where the attention scores are modified.

A.3 Feed-forward pass: Safe use of shorter inputs for the task-specific transformer

The top-K selection enables the use of shorter inputs for the task-specific. We prove that using input length equal to K is equivalent to using input length higher than K, without any loss of context. Note that the pruning scores are the same for both inputs, where the top-K are scored non-zero and we impose the other tokens to be scored zero.

Theorem A.1. *Given a transformer and a set of tokens as input I. Let t be one of the input tokens $t \in I$. If the transformer verifies the following conditions, that holds for all layers i.*

1. *$\forall t' \in I$ that attends to t, , $z^{<i,t,t'>} = -\vec{\infty}$.*
2. *For t attends to any $t' \in I$, $z^{<i,t',t>} = -\vec{\infty}$.*

Then applying this transformer on I is equivalent to applying it on $I - \{t\}$

Proof. We look at the different use cases.

$\forall i$ layers, any token $t' \in I - \{t\}$ attending to any token $t'' \in I - \{t\}$: the soft-max scores $a^{<i,t',t''>}$ have the same formula using I or $I - \{t\}$ as input.

Lets fix $t' = t$. The token t attending to any token $t'' \in I$: The first condition 1 gives $\forall t''$ that attends to t, $z^{<i,t,t''>} = -\vec{\infty}$. That follows $exp(z^{<i,t,t''>}) = \vec{0}$ then $a^{<i,t,t''>} = \vec{0}$.

Similarly, if $t' = t$. Any token $t'' \in I$ attending to t: The second condition 2 gives $\forall t'$ that attends to t, $z^{<i,t,t''>} = -\vec{\infty}$. That follows $exp(z^{<i,t',t>}) = \vec{0}$ then $a^{<i,t',t>} = \vec{0}$. $\square$

Remark. *Given a transformer and a set of tokens as input I. Let t be one of the input tokens $t \in I$ with t is not selected by the pruning transformer scored zero – not the first-k tokens. Using DoT, $s_t = -\infty$. That follows $z^{<i,t,t'>} = -\vec{\infty}$.*

The case $t' = t$, for any token $t'' \in I$ attending to t we have: $\forall i \neq 0$, the input $E_t = \sum_{t'' \in I} a^{<i,t,t''>}$. As $z^{<i,t,t'>} = -\vec{\infty}$, $E_t = \vec{0}$, E_t zero out all the variables making $exp(z^{<i,t',t>})$ a constant and $a^{<i,t',t>}$ independent of t'. This is equivalent to $\forall i \neq 0$, t doesn't attend to any $t' \in I$.

Only for the first layer $i = 0$, we add an approximation to drop the attention (t attending to $t' \in I$). We consider the impact of t on the full attention is small as we stuck multiple layers. We experimented with a task-specific model with a big input length $> k$ and compare it to a task-specific model with input length $= k$. The two models gives similar accuracy. In our experiment we report only the results for the model with input length $= k$.

This makes the attention scores similar to the ones computed over $t \notin I$.

B Experiments

In all the experiment we report the median accuracy and the error margin computed over 3 runs. We estimate the error margin as half the inter quartile range, that is half the difference between the 25^{th} and 75^{th} percentiles.

B.1 Models hyper-parameters

We do not perform hyper-parameters search for DoT models we use the same as TAPAS baselines.

Dataset	**lr**	ρ	**hidden dropout**	**attention dropout**	**num steps**
WIKISQL	$6e^{-5}$	0.14	0.1	0.1	$50,000$
TABFACT	$2e^{-5}$	0.05	0.07	0.0	$80,000$
WIKITQ	$1.9e^{-5}$	0.19	0.1	0.1	$50,000$

Table 4: Hyper-parameters used per dataset. Reports the learning rate (lr), the warmup ratio (ρ), the hidden dropout, the attention dropout and the number of training steps (num steps) used for each dataset. These hyper-parameters are the same for all the baselines and DoT models.

For WIKISQL and WIKITQ we use the same hyper-parameters as the one used by (Herzig et al., 2020) and for TABFACT the one used by (Eisenschlos et al., 2020). Baselines and DoT are initialized from models pre-trained with a MASK-LM task and on SQA(Iyyer et al., 2017) following Herzig et al. (2020).

We report the models hyper parameters used for TAPAS baselines and DoT in Table 4. The hyper-parameters are fixed independently of the pre-processing step or the input size: For all the pre-processing input lengths – $\{256, 512, 1024\}$–, for both CC and HEM we use the same hyper-parameters. Additionally, we use an Adam optimizer with weight decay for all the baselines and DoT models –the same configuration as BERT.

B.2 state-of-the-art

We report state-of-the-art for the three datasets in Table 5.

B.3 Models complexity

In all our experiments we use different transformer sizes called large, medium, small and mini. These models correspond to the BERT open sourced model sizes described in Turc et al. (2019). We report all models complexity in Table 6. The sequence length changes the total number of used parameters. The formula to count the number of parameters is given by Table 7. The number of used parameters equals to $V \times H + (2+3L)I \times H + I + (256*4+17+9L)H + (1+2L \times H) \times Hi$. The number of parameters of each model is reported in Table 8

The number of parameters is not proportional to the computational time as multiple operations involves multiplying tensors of shapes $[I, H] \times [H, H]$.

C Analysis

We report additional results for the analysis.

Model	**test accuracy**
(Agarwal et al., 2019) MeRL	74.8 ± 0.2
(Liang et al., 2018) MAPO (ensemble of 10)	74.9
(Wang et al., 2019)	79.3
$CC \xrightarrow{512}$ TAPAS(l)(Herzig et al., 2020)	83.6
(Min et al., 2019)	**83.9**

(a) state-of-the-art WIKISQL

Model	**test accuracy**
(Zhong et al., 2020) LFC (LPA)	71.6
(Zhong et al., 2020) LFC (Seq2Action)	71.7
(Shi et al., 2020) HeterTFV	72.3
(Zhang et al., 2020) SAT	73.2
(Yang et al., 2020) ProgVGAT	74.4
$\boldsymbol{CC \xrightarrow{512}}$ TAPAS$(\boldsymbol{l})$ (Eisenschlos et al., 2020)	**81.0**

(b) state-of-the-art TABFACT

Model	**test accuracy**
(Agarwal et al., 2019) MeRL	44.1 ± 0.2
(Dasigi et al., 2019) Iterative Search (best)	44.3
(Wang et al., 2019)	44.5
(Liang et al., 2018) MAPO (ensembled-10)	46.3
(Agarwal et al., 2019) MeRL ensemble of 10 models	46.9
$CC \xrightarrow{512}$ TAPAS(l)(Herzig et al., 2020)	48.8
(Yin et al., 2020) $\boldsymbol{MAPO + TABERT(l)(K=3)}$	$\boldsymbol{51.8 \pm 0.6}$

(c) state-of-the-art WIKITQ

Table 5: state-of-the-art accuracy on test set.

Model	$\#L$	H	$\#Hs$	Hi
large	24	1024	16	4096
medium	8	512	8	2048
small	4	512	8	2048
mini	4	256	4	1024

Table 6: Models complexity with $\#L$ is the number of layers, $\#Hs$ the number of heads, H the embedding size and Hi the intermediate size.

Num layers × Module	**Tensor**	**Shape**
1×Embedding	embeddings.word_embeddings	$[V, H]$
	embeddings.position_embeddings	$[I, H]$
	embeddings.token_type_embeddings	$[3, H]$ $+[2, H]$ $+[10, H]$ $+4[256, H]$
	embeddings.LayerNorm	$[H]$ $+[I]$
L×Transformer	encoder.layer.0.attention.self.query.kernel	$[I, H]$
	encoder.layer.0.attention.self.query.bias	$[H]$
	encoder.layer.0.attention.self.key.kernel	$[I, H]$
	encoder.layer.0.attention.self.key.bias	$[H]$
	encoder.layer.0.attention.self.value.kernel	$[I, H]$
	encoder.layer.0.attention.self.value.bias	$[H]$
	encoder.layer.0.attention.output.dense.kernel	$[H, H]$
	encoder.layer.0.attention.output.dense.bias	$[H]$
	encoder.layer.0.attention.output.LayerNorm	$[H]$ $+[H]$
	encoder.layer.0.intermediate.dense.kernel	$[H, Hi]$
	encoder.layer.0.intermediate.dense.bias	$[Hi]$
	encoder.layer.0.output.dense.kernel	$[Hi, H]$
	encoder.layer.0.output.dense.bias	$[H]$
	encoder.layer.0.output.LayerNorm	$[H]$ $+[H]$
1× Pooler	pooler.dense.kernel	$[I, H]$
	pooler.dense.bias	$[H]$

Table 7: Parameters counts. Let H be the hidden embedding size, L the number of layers, Hi the intermediate size, $V = 30522$ the vocabulary size and I the input size. We report the used parameters based on tensors shape for TAPAS models.

Model	**Parameters count**		
	$CC \xrightarrow{256}$	$CC \xrightarrow{512}$	$CC \xrightarrow{1024}$
TAPAS$(mini)$	$11.1M$	$12.0M$	$13.8M$
TAPAS(s)	$26.4M$	$28.2M$	$31.9M$
TAPAS(m)	$36.3M$	$39.7M$	$46.6M$
TAPAS(l)	$253.2M$	$272.6M$	$311.4M$
$DoT(mini \xrightarrow{256} l)$	$264.3M$	$265.3M$	$267.1M$
$DoT(s \xrightarrow{256} l)$	$279.6M$	$281.5M$	$285.1M$
$DoT(m \xrightarrow{256} l)$	$289.6M$	$293M$	$299.8M$

Table 8: The parameters count for the different models. M refers to millions. The number of parameters is the same using HEM or CC. The column based DoT models have the same number of parameters than the token based DoT models.

Bucket	**Model**	**WIKISQL**	**TABFACT**	**WIKITQ**
> 1024	$CC \xrightarrow{1024}$ TAPAS(l)	54.7 ± 1.0	74.1 ± 1.5	30.7 ± 0.9
	$CC \xrightarrow{512}$ TAPAS(l)	$\mathbf{24.3 \pm 0.1}$	$\mathbf{56.8 \pm 2.2}$	$\mathbf{18.8 \pm 0.9}$
	$CC \xrightarrow{256}$ TAPAS(l)	5.8 ± 0.2	9.9 ± 1.5	6.9 ± 0.0
	$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	$\mathbf{40.1 \pm 4.9}$	$\mathbf{69.1 \pm 2.5}$	$\mathbf{23.8 \pm 0.5}$
$[512, 1024]$	$CC \xrightarrow{1024}$ TAPAS(l)	86.1 ± 0.5	78.0 ± 0.4	48.2 ± 0.1
	$CC \xrightarrow{512}$ TAPAS(l)	$\mathbf{73.6 \pm 0.1}$	$\mathbf{73.0 \pm 0.3}$	$\mathbf{42.7 \pm 0.6}$
	$CC \xrightarrow{256}$ TAPAS(l)	40.9 ± 0.3	43.9 ± 0.4	18.6 ± 0.1
	$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	$\mathbf{72.9 \pm 1.8}$	$\mathbf{74.7 \pm 0.7}$	$\mathbf{39.1 \pm 0.6}$
$[256, 512]$	$CC \xrightarrow{1024}$ TAPAS(l)	87.0 ± 0.0	80.3 ± 0.2	56.1 ± 0.4
	$CC \xrightarrow{512}$ TAPAS(l)	88.2 ± 0.1	81.1 ± 0.2	56.5 ± 1.1
	$CC \xrightarrow{256}$ TAPAS(l)	78.5 ± 0.7	71.4 ± 0.3	50.2 ± 0.8
	$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	86.4 ± 0.7	78.4 ± 0.5	53.1 ± 0.5
< 256	$CC \xrightarrow{1024}$ TAPAS(l)	87.7 ± 0.0	82.2 ± 0.1	58.7 ± 0.2
	$CC \xrightarrow{512}$ TAPAS(l)	88.2 ± 0.3	83.2 ± 0.0	60.2 ± 1.0
	$CC \xrightarrow{256}$ TAPAS(l)	88.4 ± 1.1	82.1 ± 0.2	59.1 ± 0.7
	$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	87.8 ± 0.2	79.9 ± 0.5	57.3 ± 0.6

Table 9: The denotation accuracy for test, computed over bucketized datasets per sequence length.

C.1 Pruning transformer enables reaching high accuracy for long input sequences

To study the model accuracy on different input sequence lengths, we bucketize the datasets. Table 9 reports the accuracy results computed over the test set for all buckets. We use $DoT(m \xrightarrow{256} l)$ model for the three datasets, a token based pruning for both WIKISQL and TABFACT and a column based pruning for WIKISQL. For a length > 1024, the DoT model outperforms the 256 and 512 length baselines for all tasks. For the bucket $[512, 1024]$, DoT model gives close results to 512 length baseline. This indicates that the pruning model extracts twice more relevant tokens than the heuristic CC. For smaller input lengths the baseline models outperform DoT. One cause could be the hyper-parameters tuning as we do not tune the hyper parameters for DoT.

C.2 Choice of joint learning: Is it better to impose the meaning of relevant tokens?

According to the analysis done in Section 5, the pruning model –jointly learned– is selecting the tokens to solve the main task. This raises a question of whether adding a pruning loss similar to the task-specific loss can improve the end-to-end accuracy. We not $J_{pruning-scalar}$ the pruning loss and $J_{task-specific-scalar}$ the task-specific loss. Both are similar to J_{scalar} defined by (Herzig et al., 2020) where the attention scores are not affected by the pruning scores. We additionally not $J_{task-specific-scalar|S}$ the task specific loss affected by the set of pruning scores S.

We compare the joint learning model $J\text{-}DoT(.)$ – defined in Appendix A.2 – to a model learned using an additional pruning loss $P\text{-}DoT(.) = J_{aggr} + \beta(J_{task-specific-scalar} + J_{pruning-scalar})$, and another using both $PJ\text{-}DoT(.) = J_{aggr} + \beta(J_{task-specific-scalar|S} + J_{pruning-scalar})$. Table 10 shows that for both WIKISQL and WIKITQ joint learning achieves higher accuracy for similar efficiency. For TABFACT the median is in the margin error but the best model using the joint learning outperforms the other learning strategies.

D All models results

We report all DoT results in Table 11. $C - DoT$ indicates the column based selection: For each token from one column, the pruning score attributes a column score instead of a token score. The column score is computed as an average of its tokens' scores.

Dataset	WIKISQL			TABFACT			WIKITQ		
Model	**test accuracy**	**Best**	NPE/s	**test accuracy**	**Best**	NPE/s	**test accuracy**	**Best**	NPE/s
$CC \xrightarrow{1024} P\text{-}DoT(m \xrightarrow{256} l)$	80.4 ± 0.6	82.11	950	79.7 ± 0.5	80.20	930	43.5 ± 0.6	44.54	950
$CC \xrightarrow{1024} PJ\text{-}DoT(m \xrightarrow{256} l)$	82.9 ± 0.6	83.21	950	78.0 ± 0.3	78.41	930	46.4 ± 0.8	48.43	950
$\boldsymbol{CC \xrightarrow{1024} J\text{-}DoT(m \xrightarrow{256} l)}$	$\mathbf{83.6 \pm 0.5}$	**84.67**	**950**	$\mathbf{79.0 \pm 0.5}$	**81.28**	**930**	$\mathbf{50.1 \pm 0.5}$	**50.14**	**950**

Table 10: Test denotation accuracy using different DoT training losses. We compare a joint learning $J\text{-}DoT$ model – that enables the back propagation of the pruning transformer by modifying the attention scores – to two different DoT models where we modify the learning strategy. $P\text{-}DoT$ disables the buck-propagation to the pruning transformer through the attention scores, instead it uses an additional pruning loss similar to the task-specific loss. The second strategy, $JP\text{-}DoT$ is a hybrid method where the joint learning is used along with an additional pruning loss. $J\text{-}DoT$ achieves higher accuracy for similar efficiency, for both WIKISQL and WIKITQ. For TABFACT the median is in the margin error but the best model using the joint learning outperforms the other learning strategies.

Dataset	WIKISQL			TABFACT			WIKITQ		
Model	**test accuracy**	**Best**	**NPE/s**	**test accuracy**	**Best**	**NPE/s**	**test accuracy**	**Best**	**NPE/s**
state-of-the-art	**83.9**	–	–	**81.0**	–	–	$\mathbf{51.8 \pm 0.6}$	**52.3**	–
$CC \xrightarrow{1024}$ TAPAS(s)	74.6 ± 0.1	74.69	3900	73.3 ± 0.1	73.40	4400	36.0 ± 0.4	36.92	3800
$CC \xrightarrow{1024}$ TAPAS(m)	81.6 ± 0.2	81.55	2050	75.1 ± 0.2	75.75	2300	42.9 ± 0.3	43.67	2020
$CC \xrightarrow{256}$ TAPAS(l)	76.4 ± 0.3	77.15	1870	75.1 ± 0.3	76.13	1900	44.8 ± 0.5	45.47	1900
$CC \xrightarrow{512}$ TAPAS(l)	83.6 ± 0.1	83.65	800	81.3 ± 0.2	81.60	870	52.2 ± 0.5	52.74	810
$CC \xrightarrow{1024}$ TAPAS(l)	$\mathbf{86.0 \pm 0.3}$	**86.6**	**270**	81.6 ± 0.1	81.64	300	53.9 ± 0.2	54.30	270
$HEM \xrightarrow{256}$ TAPAS(l)	77.4 ± 0.3	77.97	1870	75.5 ± 0.2	75.80	1900	47.3 ± 0.1	47.70	1900
$HEM \xrightarrow{512}$ TAPAS(l)	83.8 ± 0.4	84.75	800	$\mathbf{82.0 \pm 0.3}$	**82.07**	**870**	52.7 ± 0.4	53.61	810
$HEM \xrightarrow{1024}$ TAPAS(l)	$\mathbf{85.9 \pm 0.0}$	**85.94**	**270**	80.6 ± 0.0	80.6	300	$\mathbf{54.0 \pm 0.9}$	**54.93**	**270**
$CC \xrightarrow{1024} DoT(mini \xrightarrow{256} l)$	72.8 ± 0.8	73.01	1600	77.2 ± 0.5	77.72	1670	37.4 ± 0.9	39.48	1600
$CC \xrightarrow{1024} DoT(s \xrightarrow{256} l)$	74.2 ± 3.6	84.27	1250	81.0 ± 0.1	81.17	1300	40.8 ± 0.4	42.15	1250
$CC \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	83.6 ± 0.5	84.67	950	79.0 ± 0.5	81.28	930	42.4 ± 0.5	43.44	950
$HEM \xrightarrow{1024} DoT(mini \xrightarrow{256} l)$	73.4 ± 0.1	73.70	1600	77.6 ± 0.2	78.19	1670	39.2 ± 0.3	39.8	1600
$HEM \xrightarrow{1024} DoT(s \xrightarrow{256} l)$	$\mathbf{85.3 \pm 0.4}$	**85.76**	**1250**	$\mathbf{81.6 \pm 0.3}$	**81.74**	**1300**	42.1 ± 0.7	42.15	1250
$HEM \xrightarrow{1024} DoT(m \xrightarrow{256} l)$	$\mathbf{85.5 \pm 0.2}$	**85.82**	**950**	$\mathbf{81.8 \pm 0.0}$	**81.94**	**930**	48.2 ± 1.8	48.46	950
$CC \xrightarrow{1024} C - DoT(mini \xrightarrow{256} l)$	72.0 ± 1.1	74.06	1560	77.3 ± 4.8	77.61	1600	41.0 ± 0.4	42.13	1560
$CC \xrightarrow{1024} C - DoT(s \xrightarrow{256} l)$	74.3 ± 0.1	74.49	1250	77.2 ± 4.9	78.04	1300	48.1 ± 2.4	49.47	1250
$CC \xrightarrow{1024} C - DoT(m \xrightarrow{256} l)$	73.9 ± 0.2	7454	950	78.3 ± 0.7	80.12	930	$\mathbf{50.1 \pm 0.5}$	**50.14**	**950**
$HEM \xrightarrow{1024} C - DoT(mini \xrightarrow{256} l)$	72.0 ± 1.1	74.06	1560	78.1 ± 4.8	78.13	1600	41.5 ± 0.1	41.99	1560
$HEM \xrightarrow{1024} C - DoT(s \xrightarrow{256} l)$	74.5 ± 0.8	74.72	1250	58.5 ± 0.3	59.19	1300	40.9 ± 0.2	41.23	1250
$HEM \xrightarrow{1024} C - DoT(m \xrightarrow{256} l)$	74.3 ± 2.1	74.56	950	77.3 ± 0.1	77.71	930	40.1 ± 2.4	49.13	950

Table 11: Summary of all the experiments' results on the accuracy efficiency trade-off. The state-of-the-art (detailed in Appendix B.2) correspond to the values of (Min et al., 2019) for WIKISQL, (Eisenschlos et al., 2020) for TABFACT and (Yin et al., 2020) for WIKITQ. For each dataset, the state-of-the-art, the best baseline model on accuracy, and the DoT models that reach the best accuracy efficiency trade-off are highlighted.

Grammatical Error Correction as GAN-like Sequence Labeling

Kevin Parnow[1,2,3,†], **Zuchao Li**[1,2,3,†], **and Hai Zhao**[1,2,3*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
`{parnow, charlee}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn`

Abstract

In Grammatical Error Correction (GEC), sequence labeling models enjoy fast inference compared to sequence-to-sequence models; however, inference in sequence labeling GEC models is an iterative process, as sentences are passed to the model for multiple rounds of correction, which exposes the model to sentences with progressively fewer errors at each round. Traditional GEC models learn from sentences with fixed error rates. Coupling this with the iterative correction process causes a mismatch between training and inference that affects final performance. In order to address this mismatch, we propose a GAN-like sequence labeling model, which consists of a grammatical error detector as a discriminator and a grammatical error labeler with Gumbel-Softmax sampling as a generator. By sampling from real error distributions, our errors are more genuine compared to traditional synthesized GEC errors, thus alleviating the aforementioned mismatch and allowing for better training. Our results on several evaluation benchmarks demonstrate that our proposed approach is effective and improves the previous state-of-the-art baseline.

1 Introduction

Sequence-to-sequence neural solutions (Parnow et al., 2020) have been quite successful in comparison to their statistical counterparts (Sutskever et al., 2014), but these approaches suffer from a couple key problems, which has given rise to sequence labeling approaches for GEC (Omelianchuk et al., 2020). Such approaches task models with generating a list of labels to classify the grammatical errors in a sentence before correcting these errors.

Sequence labeling approaches have recently gained popularity in GEC and are currently state-of-the-art. One typical aspect of sequence labeling approaches is labeling and correcting sentences through an iterative process. As successive edits will depend on how other errors are corrected in a sentence, using an iterative process and correcting only the most salient errors in each round allows models to achieve better performance; however, because of this process, models are tasked with handling sentences with varying rates of errors, as during each round of inference for a given sentence, a model encounters a sentence with progressively fewer errors. This of course causes an exposure bias problem, as the training data does not match the test data, and suggests that providing the model with training data with varying error rates will lead to better performance.

To combat this exposure bias, we propose a new approach for training a sequence labeling GEC model that draws from GANs (Goodfellow et al., 2014), which consist of a generator that generates increasingly realistic fake inputs and a discriminator that is tasked with differentiating these fake inputs from real inputs. Other GEC works like (Raheja and Alikaniotis, 2020) directly used GANs to produce grammatically correct sentences given grammatically incorrect ones. This contrasts our work, which uses aspects of a GAN to enhance the training process rather than using a GAN itself as the correcting model. Our model consists of three components: an encoder, a Grammatical Error Detector, and a Grammatical Error Labeler. By sampling from the error distribution in the error labeler, our model can synthesize sentences with new errors creating new sentence pairs for further training data. As a result, our Detector continually

---

[*] Corresponding author. [†] These authors made equal contribution. This work was supported by Huawei Noah's Ark Lab and funded by the National Key Research and Development Program of China (No. 2017YFB0304100), the Key Projects of National Natural Science Foundation of China (U1836222 and 61733011), the Huawei-SJTU long term AI project, Cutting-edge machine reading comprehension and language model.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3284–3290
August 1–6, 2021.

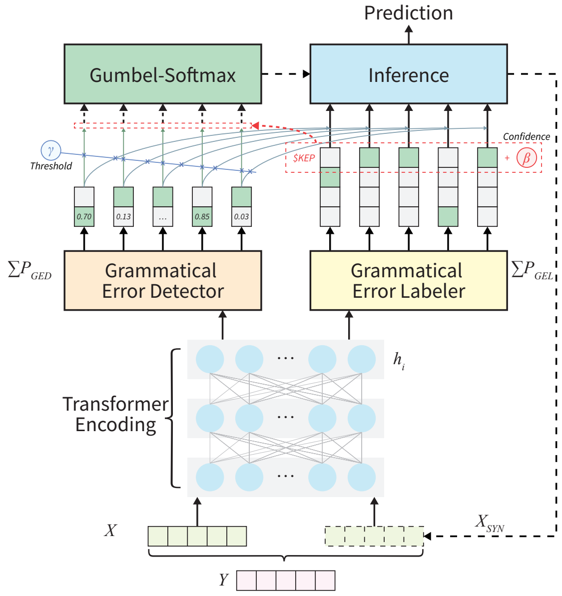

Figure 1: An overview of our model.

improves its ability to detect errors and essentially acts as a discriminator of errors, and our Labeler continually improves the authenticity of its error distribution and becomes a better generator of errors. This process allows us to counter the exposure bias problem sequence labeling GEC models face because in addition to allowing us to generate new errorful sentences whose errors are increasingly representative of those in real data, we can also use control parameters to set the error rates of these sentences and accommodate our iterative inference process.

2 Our Approach

We formulate the GEC task as a problem of sequence labeling and create a neural sequence labeling model based on a deep pre-trained Transformer encoder to deal with this problem. Inspired by the work of (Omelianchuk et al., 2020), our full model's overall architecture diagram is shown in Figure 1. There are three main components in our basic neural GEC model: a deep pre-trained Transformer Encoder, a Grammatical Error Detector, and a Grammatical Error Labeler. To accommodate our new GAN-like training process, we add a Gumbel-softmax sampling component to the basic GEC model.

2.1 Background and Notation

First, in training, given incorrect input sentence $X = x_1, x_2, ..., x_n$ and its corrected version $X_c = y_1, y_2, ..., y_m$, the model predicts a corrective label sequence $T = t_1, t_2, ..., t_n$ by minimizing the token-level Levenshtein distance on the span-based alignments of X and X_c. The corrective label set is given as $\mathcal{T} = \{\$KEP, \$DEL, \$APP, \$REP\} \cup \{\$CAS, \$MRG, \$SPL, \$NNUM, \$VFORM\}$, in which the first set consists of the basic text editing transformation operations and the second consists of g-transformations as defined by (Omelianchuk et al., 2020) for GEC[1]. Aligning sentences using these transformations in preprocessing, reduces what would be a sequence generation task that handles unequal source-target lengths to a set of label classification problems. In this formulation, the neural sequence labeling model trains to optimize the input sequence's negative log-likelihood loss for an input sequence:

$$\mathcal{J}(\theta) = -\sum_{i=1}^{n} \log p(t_i|x, \theta),$$

where p is the conditional probability that the model outputs at each position i.

2.2 Deep Pre-trained Transformer Encoder

As in most neural sequence labeling models (Ma and Hovy, 2016), a neural encoder such as a BiLSTM (Hochreiter and Schmidhuber, 1997) or a Transformer (Vaswani et al., 2017; Li et al., 2021) is used to extract context-aware features from the input sequence. Deep pre-trained language models such as BERT (Devlin et al., 2019; Zhang et al., 2020b), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019) have recently demonstrated the efficacy of Transformer models trained on large-scale unlabeled data in various NLP tasks. We leveraged these very beneficial models by using a pre-trained language model as our encoder. We define the contextualized features captured by the neural encoder as:

$$h_i = [\mathbf{Enc}(X)]_i,$$

where **Enc** represents the encoder, and $[\cdot]_i$ represents the output of the i-th position after encoding.

2.3 Grammatical Error Detector and Labeler

Next, we adopt a a Grammatical Error Detector (GED) to detect the presence of errors and a Gram-

---

[1] The label set here only presents the transformations' basic names. Some transformations require additional parameters because they are context-specific and thus have many different versions.

matical Error Labeler (GEL) to predict detailed error labels. With these labels, corrections are applied to sentences, and this process is typically iterative, as some corrections may depend on others, and applying corrections only once may not be enough to fully correct the sentence. During iterative correction, the model needs to assess at each round whether more correction is required. To this end, we use the GED to determine the degree of error for an entire sentence and control the iterative correction process.

Specifically, we use a binarization Y_b of the corrective labels Y as the training target of the GED and use Y as the training target of the GEL. To obtain label probabilities grammatical error detection and labeling, two linear layers with softmax layers are appended to the encoder:

$$P^i_{\text{GED}} = \mathbf{softmax}(\mathbf{MLP}_{\text{GED}}(h_i)),$$

$$P^i_{\text{GEL}} = \mathbf{softmax}(\mathbf{MLP}_{\text{GEL}}(h_i)).$$

The binary classification probabilities in the GED output do not necessarily control the inference process's iterations. Rather, after using the GEL error label probabilities as thresholds for sentence positions, we also use the sum of these probabilities as a threshold for attempting another round of correction on the whole sentence. The model continues correcting the sentence until either it reaches a preset maximum number of iterations or no longer satisfies the following condition: $\sum_i [P^i_{\text{GED}}]_{err=1} > \gamma$, where γ is the minimum error probability threshold for a sentence.

Additionally, since GEC usually corrects a small portion of a sentence (and there are therefore no errors in most of the input), the corrective label prediction task is an imbalanced classification problem. We alleviate this imbalance classification issue by taking advantage of this prior knowledge and adding a fixed and preset confidence β to the label \$KEP to keep a position unchanged when applying corrections:

$$[P^i_{\text{GEL}}]_{\$\text{KEP}} = [P^i_{\text{GEL}}]_{\$\text{KEP}} + \beta.$$

2.4 GAN-like Sequence Labeling Training

While we adopt sequence labeling instead of sequence-to-sequence modeling in this paper and therefore avoid the exposure bias problem caused by left-to-right sequence generation, our model still faces exposure bias because of the iterative correction process, which, through its iterative correction process, tasks the model with handling much more varied error rates in inference compared to in training, where it handles static data and does not use multiple-round corrections. To address this issue, we borrow the idea of a GAN (Goodfellow et al., 2014) and propose a GAN-like iterative training approach for a sequence labeling GEC model. GANs, whose training objective can be formulated as a minimax game between a generator that creates increasingly realistic fake outputs and a discriminator that must differentiate these outputs from their real counterparts, have been suggested for sequence-to-sequence text generation (Li et al., 2020; Zhang et al., 2020a; Li et al., 2018) as they do not suffer from exposure bias.

Algorithm 1 GAN-like Sequence Labeling Training

```
Require: Genuine GEC parallel dataset $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y})\}$
    Synthesized GEC parallel dataset $\mathcal{D}_{\text{SYN}} = \{\}$
    Number of training stages $\mathcal{N}$
    Number of training epochs $M$
    Sentence error probability threshold $\gamma$
    Additional confidence $\beta$ for label $\texttt{\$KEP}$
for $i$ in 1, ..., $N$ do
    Initialize model parameters from previous training stage $\theta_{\text{i}} \leftarrow \theta_{\text{i-1}}$ when $i > 1$
    for $j$ in 1, ..., $M$ do
        for $k$ in 1, ..., $|\mathcal{D} \cup \mathcal{D}_{\text{SYN}}|$ do
            Encode each sentence $\mathbf{X}_k$ as $\mathbf{H}_k$
            $P^k_{\text{GED}} = \mathbf{Softmax}(\text{MLP}_{\text{GED}}(\mathbf{H}_k))$
            $P^k_{\text{GEL}} = \mathbf{Softmax}(\text{MLP}_{\text{GEL}}(\mathbf{H}_k))$
            $loss_{\text{GED}} = \mathbf{CrossEntropy}(P^k_{\text{GED}}, \mathbf{Y}^k_{err})$
            $loss_{\text{GEL}} = \mathbf{CrossEntropy}(P^k_{\text{GEL}}, \mathbf{Y}^k_{label})$
            $loss = loss_{\text{GED}} + loss_{\text{GEL}}$
            Update the model parameter $\theta_i$ with $loss$
        end for
    end for
    $\mathcal{D}_{\text{SYN}} = \{\}$
    for $k$ in 1, ..., $|\mathcal{D}|$ do
        Encode each sentence $\mathbf{X}_k$ as $\mathbf{H}_k$
        $P^k_{\text{GED}} = \mathbf{Softmax}(\text{MLP}_{\text{GED}}(\mathbf{H}_k))$
        $P^k_{\text{GED}} = \sum[P^k_{\text{GED}}]_{err=1} > \gamma$
        $P^k_{\text{GEL}} = \mathbf{Softmax}(\text{MLP}_{\text{GEL}}(\mathbf{H}_k))$
        $[P^k_{\text{GEL}}]_{\$\text{KEP}} = [P^k_{\text{GEL}}]_{\$\text{KEP}} + \beta$
        $P^k_{\text{GEL}} = \mathbf{GumbelSoftmax}(P^k_{\text{GEL}})$
        Use $P^k_{\text{GED}}$ and $P^k_{\text{GEL}}$ to produce sampled sequence $\mathbf{X}^k_{\text{SYN}}$
        $\mathcal{D}_{\text{SYN}} = \mathcal{D}_{\text{SYN}} \cup \{(\mathbf{X}^k_{\text{SYN}}, \mathbf{Y}_k)\}$
    end for
end for
```

In our model, the GEL module can be considered a discriminator, as it must differentiate whether tokens are erroneous, and by adding a sampling module to the GED module, we can create a generator that outputs grammatical errors (rather than corrections) that are increasingly realistic. We can then pair these sampling outputs with their golden sequence in the training dataset to create new training

GEC system	**Ens.**	**CoNLL-2014 (test)**			**BEA-2019 (test)**			**JFLEG (test)**
		P	**R**	**F$_{0.5}$**	**P**	**R**	**F$_{0.5}$**	**GLEU**
(Zhao et al., 2019)		67.7	40.6	59.8	–	–	–	–
(Awasthi et al., 2019)		66.1	43.0	59.7	–	–	–	60.3
(Kiyono et al., 2019)		67.9	44.1	61.3	65.5	59.4	64.2	59.7
(Kaneko et al., 2020)		69.2	45.6	62.6	67.1	60.1	65.6	61.3
(Lichtarge et al., 2019)	✓	66.7	43.9	60.4	–	–	–	**63.3**
(Zhao et al., 2019)	✓	71.6	38.7	61.2	–	–	–	61.0
(Awasthi et al., 2019)	✓	68.3	43.2	61.2	–	–	–	61.0
(Kiyono et al., 2019)	✓	72.4	46.1	65.0	74.7	56.7	70.2	61.4
(Kantor et al., 2019)	✓	–	–	–	78.3	58.0	**73.2**	–
(Kaneko et al., 2020)	✓	72.6	**46.4**	65.2	72.3	**61.4**	69.8	62.0
Baseline (BERT-base)		72.1	42.0	63.0	71.5	55.7	67.6	60.1
+GST		72.6	42.5	63.6	71.9	55.9	68.0	60.5
Baseline (RoBERTa-base)		73.9	41.5	64.0	77.2	55.1	71.5	60.6
+GST		74.1	42.2	64.4	77.5	55.7	71.9	60.9
Baseline (XLNet-base)		77.5	40.1	65.3	79.2	53.9	72.4	61.5
+GST		**78.4**	39.9	**65.7**	**79.4**	54.5	72.8	61.8

Table 1: Comparison of GEC models. The baseline comes from the model released by (Omelianchuk et al., 2020).

samples. This trains the model with more samples and more varied errors and alleviates the exposure bias issue. Separate cross-entropy losses are calculated for the Grammatical Error Detector and Labeler, and we detail the whole algorithm for our training process in Algorithm 3.

3 Detailed Training Process

To synthesize new errors based on a genuine grammatical error distribution, we add a sampling module to a trained GED module. Specifically, we use Gumbel-softmax sampling, a simple and efficient way to draw samples z from a categorical distribution with class probabilities P_{GEL} using the Gumbel-Max trick (Gumbel, 1954; Maddison et al., 2014):

$$z = \texttt{one\_hot}\left(\mathbf{argmax}_j\left[g_j + \log[P^i_{\text{GEL}}]_j\right]\right) \quad (1)$$

where $g_1 ... g_j$ are i.i.d samples drawn from $\text{Gumbel}(0,1)$[2]. We use the softmax function as a continuous, differentiable approximation to $\mathbf{argmax}$:

$$[y_i]_k = \frac{\exp((\log([P^i_{\text{GEL}}]_k) + g_k)/\tau)}{\sum_{j=1}^{|C|}\exp((\log([P^i_{\text{GEL}}]_j) + g_j)/\tau)}, \quad (2)$$

where $|C|$ is the number of classes, τ is the softmax temperature. Altering γ and β allows us to synthesize input samples of different error rates.

Sampling	**CoNLL-2014 (test)**		
	P	**R**	**F$_{0.5}$**
Random	74.3	40.2	63.5
GumbelSoftmax	78.4	39.9	65.7
Multinomial	78.1	39.9	65.5

Table 2: Comparing the effects of different sampling distributions.

4 Experiments

4.1 Setup

To isolate our GAN-like Sequence Labeling Training (GST) approach, we use the same model setting and training details as in (Omelianchuk et al., 2020). The training data includes PIE's synthetic data (Awasthi et al., 2019), NUCLE (Dahlmeier et al., 2013), Lang-8 (Tajiri et al., 2012), FCE (Yannakoudakis et al., 2011), Cambridge Learner Corpus (the publicly available portion) (Nicholls, 2003), and WI+LOCNESS (Bryant et al., 2019). Our models are evaluated on the test sets of CoNLL-2014 (Ng et al., 2014), BEA-2019 (Bryant et al., 2019), and JFLEG (Napoles et al., 2017) with the official M^2 (Dahlmeier and Ng, 2012), ERRANT (Bryant et al., 2017), and GLEU(Napoles et al., 2015) scorers, respectively.

4.2 Results and Analysis

Our results on the three test datasets are listed in Table 1. Our baseline model achieves the best single model CoNLL-2014 $F_{0.5}$, BEA-2019 $F_{0.5}$, and JFLEG GLEU scores, showing that the baseline we use is very strong. The results on the three

---

[2] The $\text{Gumbel}(0,1)$ distribution can be sampled using inverse transform sampling by drawing $u \sim \text{Uniform}(0,1)$ and computing $g = -\log(-\log(\text{u}))$.

Model	CoNLL-2014 (test)		
	P	**R**	**$F_{0.5}$**
Baseline	72.1	42.0	63.0
Baseline + Extended Training	73.5	40.1	63.0
Intermediate Outputs	73.2	39.8	62.7
GST	72.6	42.5	63.6

Table 3: Comparing GST training with additional baselines.

benchmarks are further improved using the GST approach, which demonstrates that the GST approach can effectively alleviate the exposure bias issue. With GST, we achieved new best results on the CoNLL-2014 test dataset, surpassing ensemble methods while only using a single model.

In order to illustrate the benefits of sampling using Gumbel-Softmax, we replaced it with random sampling and Multinomial. The comparison is shown in Table 2. Random sampling actually hampers performance, which shows that synthetic sentences not based on a genuine error distribution do not alleviate exposure bias. Both GumbelSoftmax and Multinomial, which use a genuine error distribution, improve the model, though Gumbel-Softmax appears to be more suitable for sampling in sequence labeling modeling.

In Figure 2, we show how the performance changes with increasing rounds of GST training. In the first few rounds, due to the model's re-adaptation to new errors, there was a drop in performance on the test datasets; however, as the number of training rounds increased, performance on the test set gradually improved and finally stabilized.

Intermediate Outputs and Longer Training In this experiment, we explored using intermediate outputs from our iterative inference process as additional training outputs to highlight the impact of generating new erroneous sentence by sampling from the real error distribution with our GST approach. For this experiment, we use our baseline architecture. As seen in the results in Table 3, whereas GST leads to a 0.6 $F_{0.5}$ gain over the baseline, using intermediate training outputs paired with golden sentences for additional training actually leads to worse performance, yielding a 0.3 $F_{0.5}$ loss in comparison to the baseline.

To confirm that GST's performance gain is not due to the added training time, we also train the baseline for a commensurate amount of additional steps but find that this does not have any effect on model performance. This experiment demonstrates that our model does bring improvement to the baseline without relying on additional training steps. We also note that as our model is not significantly different in size from our baseline, our improvement is also not brought about by simply using a larger model.

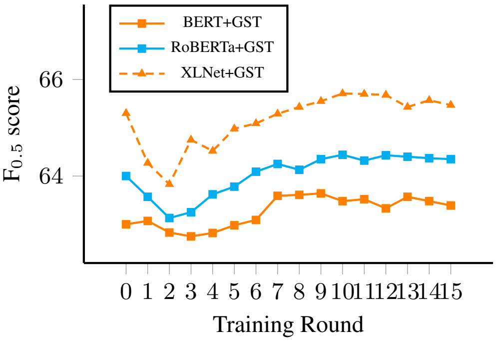

Figure 2: The GEC performance versus the the GST rounds on the CoNLL-2014 test set.

Model	CoNLL-2014 (test)		
	P	**R**	**$F_{0.5}$**
Baseline	72.1	42.0	63.0
GST	72.6	42.5	63.6
Baseline w/o BERT	65.0	32.6	54.2
GST w/o BERT	64.9	35.6	55.7

Table 4: Evaluating GST without pre-trained language models.

Performance with out Pre-trained Language Models We additionally explored the performance of our system in the absence of contextualized pre-trained language models. As we expected, these models make our model much more resilient to the exposure bias problem, and as seen in Table 4, the improvement brought about GST is therefore much more evident. In comparison to the baseline, using GST brings an improvement of 1.5 $F_{0.5}$ points.

5 Conclusion

In this paper, we studied the exposure bias problem GEC sequence labeling models face. To alleviate this issue, we proposed a novel GAN-like training method for the GEC sequence labeling model. Through evaluation on three GEC benchmarks, we demonstrate that our novel training approach further improves a strong baseline model, illustrating the effectiveness of our training approach. Notably, with the help of pre-trained language models and our training approach, we achieved state-of-the-art results on the CoNLL-2014 benchmark.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. Association for Computational Linguistics.

Yoav Kantor, Yoav Katz, Leshem Choshen, Edo Cohen-Karlik, Naftali Liberman, Assaf Toledo, Amir Menczel, and Noam Slonim. 2019. Learning to combine grammatical error corrections. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 139–148, Florence, Italy. Association for Computational Linguistics.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zuchao Li, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, Zhuosheng Zhang, and Hai Zhao. 2020. Data-dependent gaussian prior objective for language generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zuchao Li, Zhuosheng Zhang, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. 2021. Text compression-aided transformer encoding. *CoRR*, abs/2102.05951.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3086–3094.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA, Online. Association for Computational Linguistics.

Kevin Parnow, Zuchao Li, and Hai Zhao. 2020. Grammatical error correction: More data with more context. In *International Conference on Asian Language Processing, IALP 2020, Kuala Lumpur, Malaysia, December 4-6, 2020*, pages 24–29. IEEE.

Vipul Raheja and Dimitris Alikaniotis. 2020. Adversarial Grammatical Error Correction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3075–3087, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Zhuosheng Zhang, Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, Zuchao Li, and Hai Zhao. 2020a. Neural machine translation with universal visual representation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020b. Semantics-aware BERT for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9628–9635. AAAI Press.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Neural Entity Recognition with Gazetteer based Fusion

Qing Sun
Amazon AWS AI
`qinsun@amazon.com`

Parminder Bhatia
Amazon AWS AI
`parmib@amazon.com`

Abstract

Incorporating external knowledge into Named Entity Recognition (NER) systems has been widely studied in the generic domain. In this paper, we focus on clinical domain where only limited data is accessible and interpretability is important. Recent advancement in technology and the acceleration of clinical trials has resulted in the discovery of new drugs, procedures as well as medical conditions. These factors motivate towards building robust zero-shot NER systems which can quickly adapt to new medical terminology. We propose an auxiliary gazetteer model and fuse it with an NER system, which results in better robustness and interpretability across different clinical datasets. Our gazetteer based fusion model is data efficient, achieving +1.7 micro-F_1 gains on the i2b2 dataset using 20% training data, and brings + 4.7 micro-F_1 gains on novel entity mentions never presented during training. Moreover, our fusion model is able to quickly adapt to new mentions in gazetteers without re-training and the gains from the proposed fusion model are transferable to related datasets.

1 Introduction

Named entity recognition (NER) (Lample et al., 2016; Ma and Hovy, 2016) aims to identify text mentions of specific entity types. In clinical domains, it's particularly useful for automatic information extraction, e.g., diagnosis information and adverse drug events, which could be applied for a variety of downstream tasks such as clinical event surveillance, decision support (Jin et al., 2018), pharmacovigilance, and drug efficacy studies.

We have witnessed a rapid progress on NER models using deep neural networks. However, applying them to clinical domain (Bhatia et al., 2019) is hard due to the following challenges: (a) accessibility of limited data, (b) discovery of new drugs, procedures and medical conditions and the (c) need for building interpretable and explainable models. Motivated by these, we attempt to incorporate external name or ontology knowledge, e.g., *Remdesivir* is a `DRUG` and *COVID-19* is a `Medical Condition`, into neural NER models for clinical applications.

Recent work on leveraging external knowledge can be categorized into two categories - Gazetteer embedding and Gazetteer models. Recent work has primarily focused on gazetteer embeddings. Song et al. (2020) feed the concatenation of BERT output and gazetteer embedding into Bi-LSTM-CRF. Peshterliev et al. (2020) use self-attention over gazetteer types to enhance gazetteer embedding and then concatenate it with ELMO, char CNN and GloVe embeddings. By contrast, the basic idea of gazetteer model is to treat ontology knowledge as a new clinical modality. Magnolini et al. (2019) combine outputs of Bi-LSTM and gazetteer model and feed them into CRF layer. Liu et al. (2019a) apply hybrid semi-Markov conditional random field (HSCRF) to predict a set of candidate spans and rescore them with a pre-trained gazetteer model.

In this paper, we combine the advantages of both worlds. Unlike the work of Peshterliev et al. (2020), we build self-attention over entity mentions and their context rather than over different gazetteer types. For example, *Take Tylenol 3000* *`(NUM)`* *mg* *`(METRIC)`* *per day*, in which *Tylenol* is more likely to be a `DRUG` given `NUM`, `METRIC` in context. Moreover, we study two fusion methods to integrate information from two modalities.

- *Early fusion*. Similar to Magnolini et al. (2019), NER model and gazetteer model apply a shared tagger, as shown in Fig. 1a
- *Late fusion*. For better interpretability and flexibility, we allow NER and gazetteer models to apply separate taggers and fuse them before taking softmax, as shown in Fig. 1b

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3291–3295
August 1–6, 2021.

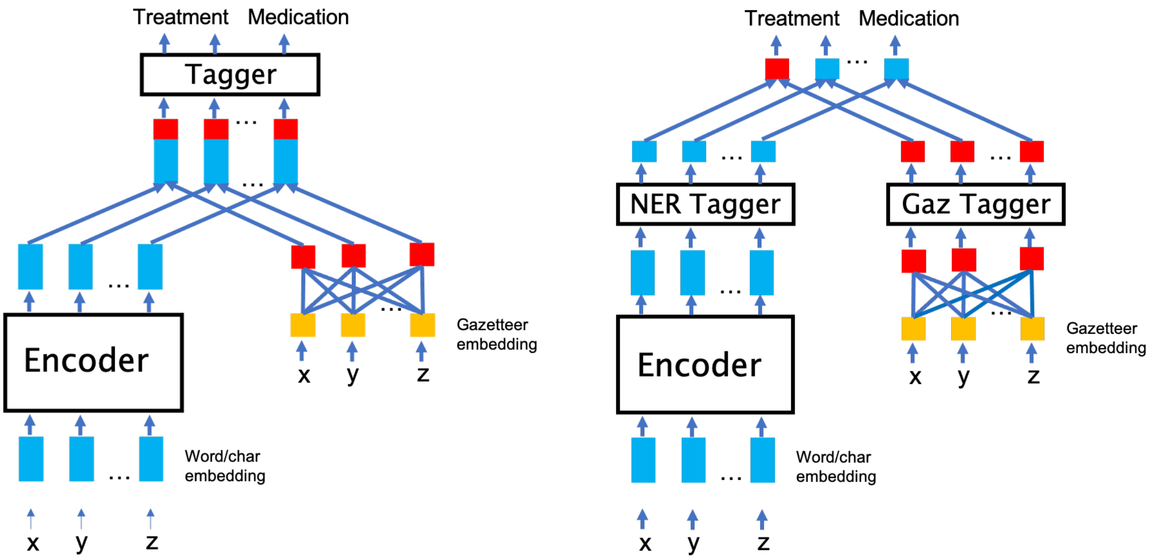

(a) Early fusion (b) Late fusion

Figure 1: Model Architecture. (a) Early fusion. The outputs of NER and gazetteer are concatenated and fed into a shared tagger. (b) Late fusion. NER and gazetteer apply separate taggers and two modalities are fused by taking element-wise max pooling.

Unlike the work of Liu et al. (2019a), NER and gazetteer models are jointly learned end-to-end.

Our contributions are as follows. (1) We propose to augment NER models with an auxiliary gazetteer model via *late fusion*, which provides better interpretability and flexibility. Interestingly, the NER model can preserve the gains even if the gazetteer model is unplugged at inference time. (2) Our thorough analysis shows that the fusion model is data efficient, explainable and is able to quickly adapt to novel entity mentions in gazetteers. (3) Experiments show that the fusion model consistently brings gains cross different clinical NER datasets.

2 Approach

2.1 NER model

NER is a sequence tagging problem by maximizing a conditional probability of tags $\mathbf{y}$ given an input sequence $\mathbf{x}$. We first encode $\mathbf{x}$ into hidden vectors and apply a tagger to produce output $\mathbf{y}$.

$$\mathbf{r} = \text{Encoder}_R(\mathbf{x}) \quad (1)$$

$$\mathbf{o}^r_t = \text{Tagger}_R(\mathbf{r}_t) \quad (2)$$

$$\mathbf{y}_t = softmax(\mathbf{o}^r_t) \quad (3)$$

2.2 Gazetteer model

We embed gazetteers into $\mathbf{E} \in \mathbb{R}^{M\times K\times d}$, where M is the number of gazetteers (e.g, drugs, medical condition), K is the number of gazetteer labels (e.g, B-Drug, E-Drug), and d is the embedding size. We define $\mathbf{E}^g_t = [\mathbf{E}_{0,z^0_t}; \mathbf{E}_{1,z^1_t}; \cdots, \mathbf{E}_{M,z^M_t}]$, where z^j_t is the gazetteer label of token $\mathbf{x}_t$ in gazetteer j. In order to model the association of name knowledge between entity mentions and their contexts, we compute context-aware gazetteer embedding using scaled dot-product self-attention

$$\mathbf{g}_t = softmax\Big(\frac{\mathbf{E}^g_t(\mathbf{E}^g_{t'})^T}{\sqrt{d}}\Big)\mathbf{E}^g_t, \ \forall|t-t'| \le w \quad (4)$$

where w is the size of attention window.

Similar to the NER model, we apply a tagger to produce output $\mathbf{y}$

$$\mathbf{o}^g_t = \text{Tagger}_G(\mathbf{g}_t) \quad (5)$$

$$\mathbf{y}_t = softmax(\mathbf{o}^g_t) \quad (6)$$

2.3 Fusion: NER + gazetteer

To better use information from both modalities, we investigate two different fusion methods to combine information from NER and gazetteer.

- *Early fusion.* In Fig. 1a, we concatenate $\mathbf{r}_t$ with $\mathbf{g}_t$, and feed it into a shared tagger

$$\mathbf{y}_t = softmax\Big(\text{Tagger}_{RG}\big([\mathbf{r}_t; \mathbf{g}_t]\big)\Big) \quad (7)$$

- *Late fusion.* In Fig. 1b, we directly fuse $\mathbf{o}^r_t$ and $\mathbf{o}^g_t$ by performing element-wise max pooling

$$\mathbf{y}_t = softmax\big(\max(\mathbf{o}^r_t, \mathbf{o}^g_t)\big) \quad (8)$$

3 Experiments

3.1 Experimental setup

LM pre-training. We continue to pre-train RoBERTa$_{\text{base}}$ (L=12, H=768, A=12) (Liu et al.,

Table 1: Results on i2b2 (Med, TTP) and DCN (Med, DS). We report micro-F_1 score, each is averaged over 3 random seeds.

	i2b2		DCN	
	Med	TTP	Med	DS
NER w/o fusion	92.26	87.22	84.51	83.99
Early fusion	92.14	87.42	84.82	84.51
Early fusion + attention	92.44	87.43	84.99	84.47
Late fusion	92.37	87.32	84.84	84.58
Late fusion + attention	92.35	87.41	84.82	84.37

2019b) on MIMIC-III dataset (Johnson et al., 2016), which comprises deidentified clinical data from $\sim$ 60k intensive care unit admissions.

Fine-tuning on clinical NER datasets. We fine-tune RoBERTa$_{\text{mimic}}$ and learn a gazetteer model (w/ NER tagger) from scratch on clinical datasets.

- i2b2 - We use public datasets from the 2009 and 2010 i2b2 challenges for medication (Med) (Uzuner et al., 2010), and "test, treatment, problem" (TTP) entity extraction. We follow the original data split from Chalapathy et al. (2016) of 170 notes for training and 256 for testing.
- De-identified clinical notes (DCN) - Second dataset (Bhatia et al., 2018) consists of 1,500 de-identified, annotated clinical notes with medications (Med) and medical conditions (DS). We follow i2b2 challenge guidelines for data annotation.

We extract medical condition and drug dictionaries from UMLS(Bodenreider, 2004) (ontology knowledge graph) based on graph as well semantic meanings. We followed different steps to prune the dictionaries based on different medical ontologies such as RxNorm for medication ($\sim$100k concepts), ICD-10 CM and SNOMED for medical conditions ($\sim$500k concepts). We employ Inside, Outside, Begin, End and Singleton (IOBES) format for both tags and gazetteers[1].

We minimize the cross-entropy loss during training and report micro-F_1 score at test time. We use RoBERTa$_{\text{mimic}}$ as NER encoder and parameterize Taggers via Multi-layer Perception (MLPs). We use BertAdam optimizer, learning rate $5e^{-5}$, and dropout 0.1. We tune hyper-parameters $d \in [2, 12]$ (best:8) and $w \in [2, 10]$ (best:5) on validation set.

Table 2: Performance on unseen entity mentions. Models are trained using 20% training data. We report performance of Medication in i2b2 Med and Treatment in i2b2 TTP.

	Medication	Treatment
NER w/o fusion	76.96	72.40
Late fusion w/ attention	81.63 **(+4.7)**	74.30 **(+1.9)**

Table 3: Ablation study on individual modules.

R_0	RG	R	R_0G
76.23	96.33	90.71	85.75

3.2 Results.

We report overall results in Table 1. We observe that incorporating name knowledge consistently boost performance on all datasets by $0.18 \sim 0.59$ micro-F_1 gains. Overall, two fusion methods achieve comparable results.

3.3 Analysis

We investigate the effectiveness of late fusion on handling three challenges: novel entity mentions, little data access and interpretability.

3.3.1 Novel entity mentions

New drugs and medical condition come out very frequently. For example, "remdesivir"and "Baricitinib" for COVID-19. To investigate the effect of late fusion on unseen entity mentions, we focus on answering questions: whether it can generalize well on unseen entity mentions, and whether it is able to correct prediction once novel entity names are added into gazetteer without re-training?

Zero-shot. We report results on unseen entity mentions not presented in train and validation sets. In Table 2, we see that late fusion brings significant improvement: $+4.7$ F1 for medication (i2b2 Med) and $+1.9$ F1 for Treatment (i2b2 TTP).

"One"-shot in gazetteer. We evaluate the ability of late fusion to quickly adapt to non-stationary gazetteers, e.g., specialists might add new entity mentions into gazetteers or give feedback when models make incorrect prediction.

For this analysis, we split entity mentions in training set into two parts: 70% labelled and 30% in gazetteer, and compare models:

- $\mathbf{R}_0$: NER model only
- $\mathbf{RG}$: $\mathbf{R} + \mathbf{G}$ via late fusion
- $\mathbf{R}$: Unplug $\mathbf{G}$ from $\mathbf{RG}$ after training
- $\mathbf{R}_0\mathbf{G}$: Fix $\mathbf{R}_0$ and learn $\mathbf{G}$ via late fusion

where $\mathbf{R}$ is NER model and $\mathbf{G}$ is gazetteer model.

---

[1] We do string matching for gazetteers by following (Chiu and Nichols, 2016). For example, if A, B and AB are all in gazetteers, we'll label AB as AB. The basic idea is to start from bigger spans, so we first check for ABC, if not found then AB, if not found then A and B.

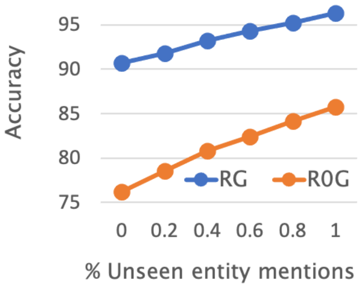

Figure 2: Quick adaptation to non-stationary gazetteers. As we increase the number of unseen entity mentions included in gazetteers, the performance goes up without re-training.

Table 4: Cross-evaluation on i2b2 Med and DCN Med. Column: dataset models are trained on. Row: dataset models are evaluated on.

	i2b2	DCN
i2b2	94.54 → 94.77 (+0.23)	68.78 → 69.68 (+0.9)
DCN	59.98 → 60.08 (+0.1)	90.02 → 90.71 (+0.69)

In Table 3, we observe that $\mathbf{G}$ plays two roles: (1) $\mathbf{R} > \mathbf{R}_0$. $\mathbf{G}$ can regularize $\mathbf{R}$ to gain better generalization ability, and (2) $\mathbf{R}_0\mathbf{G} > \mathbf{R}_0$ and $\mathbf{RG} > \mathbf{R}$. Besides serving as a regularizer, $\mathbf{G}$ provides extra information at test time.

Moreover, we evaluate late fusion by varying the number of unseen entity mentions included in gazetteers. In Fig. 2, without re-training models, late fusion can adapt to new mentions and obtain linear gains, which enables effective user feedback.

Overall, the ability to detect and adapt to novel entity mentions, without re-training models, is useful with accelerated growth in drug development as well as in practical settings where entity extraction is one of the components to build knowledge graph and search engines (Wise et al., 2020; Bhatia et al., 2020). For example, linking new drugs discovered in clinical trails of COVID-19 to standardized codes in ICD-10 [2] or SNOMED [3].

3.3.2 Limited data access

Typically, data accessible to use in the clinical domain is quite limited. In this section, we focus on evaluating fusion model in low-resource settings as well as investigate whether the gain is transferable across related datasets. Here we present results with late fusion methodology.

Low-resource setting We evaluate late fusion by reducing training data size from 100% to 20%. Fig. 3 shows late fusion gains more when less training data is present. With 20% training data, late fusion is able to boost performance over the baseline model by 1.7 micro-F_1 on i2b2 Med dataset.

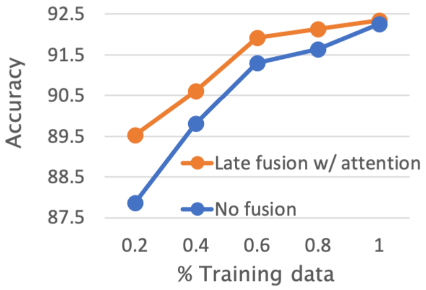

Figure 3: Accuracy vs. Training data size on i2b2 Med. We randomly sample $20\%, 40\%, \cdots, 100\%$ of training data and report micro-F_1 score averaged over 3 random seeds.

Table 5: Qualitative examples.

(1) Treated for $\underbrace{\text{COPD flare}}_{\text{B-R, I-R}}$ with supplemental DuoNebs	
NER w/o fusion	B-R, O
Late fusion w/o attention	B-R, O
Late fusion w/ attention	B-R, I-R
(2) Postop day 0, increase $\underbrace{\text{sodium}}_{\text{S-M}}$, free water added	
$\mathbf{R}$: O, $\mathbf{G}$: S-M, $\mathbf{RG}$: S-M	

Transfer learning. To verify the generalization ability of late fusion, we train models on one dataset and report evaluation on another data source. We re-train models on i2b2 Med and DCN Med using common entity types: Dosage, Medication, Frequency, and Mode. Table.4 shows that the gains from gazetteer enhanced fusion models are preserved in i2b2 → DCN and DCN → i2b2.

3.3.3 Interpretability

Explainable and controllable models are very important for clinical applications. Unfortunately, it is extremely challenging for deep neural networks. We illustrate two qualitative examples in Table.5. Late fusion models are trained on i2b2 Med using 20% training data.

(1) Late fusion correctly predicts *flare* as I-R (Reason) since *COPD flare* is a `Medical Condition`.

(2) By looking into individual predictions from $\mathbf{R}$ and $\mathbf{G}$, we notice that correct prediction is caused by name knowledge in gazetteers.

Overall, late fusion provides us a tool for diagnosis system: to answer questions whether NER or gazetteer model failed and explain why mentions belong to a particular entity type.

---

[2] `https://www.cdc.gov/nchs/icd/icd10cm.htm`

[3] `https://www.nlm.nih.gov/healthit/snomedct/index.html`

4 Conclusion

We studied fusion methods to improve NER system by leveraging name knowledge from gazetteers. We did a thorough analysis on the effectiveness of fusion methods on handling limited data and non-stationary gazetteers. In addition, we demonstrated that fusion models are explainable and can be used to improve NER systems. Future research should extend our approach to structured knowledge to further improve NER system and gain better interpretability.

Acknowledgements

We would like to thank Mohammed Khalilia and Kristjan Arumae for helpful discussions and feedback on the draft of the paper, and the anonymous reviewers for their comments.

References

Parminder Bhatia, Kristjan Arumae, Nima Pourdamghani, Suyog Deshpande, Ben Snively, Mona Mona, Colby Wise, George Price, Shyam Ramaswamy, and Taha Kass-Hout. 2020. Aws cord19-search: a scientific literature search engine for covid-19. *arXiv preprint arXiv:2007.09186*.

Parminder Bhatia, Busra Celikkaya, and Mohammed Khalilia. 2018. Joint entity extraction and assertion detection for clinical text. *arXiv preprint arXiv:1812.05270*.

Parminder Bhatia, Busra Celikkaya, Mohammed Khalilia, and Selvan Senthivel. 2019. Comprehend medical: a named entity recognition and relationship extraction web service. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1844–1851. IEEE.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Raghavendra Chalapathy, Ehsan Zare Borzeshi, and Massimo Piccardi. 2016. Bidirectional lstm-crf for clinical concept extraction. *arXiv preprint arXiv:1611.08373*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Mengqi Jin, Mohammad Taha Bahadori, Aaron Colak, Parminder Bhatia, Busra Celikkaya, Ram Bhakta, Selvan Senthivel, Mohammed Khalilia, Daniel Navarro, Borui Zhang, et al. 2018. Improving hospital mortality prediction with medical named entities and multimodal learning. *arXiv preprint arXiv:1811.12276*.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019a. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Simone Magnolini, Valerio Piccioni, Vevake Balaraman, Marco Guerini, and Bernardo Magnini. 2019. How to use gazetteers for entity recognition with neural models. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 40–49.

Stanislav Peshterliev, Christophe Dupuy, and Imre Kiss. 2020. Self-attention gazetteer embeddings for named-entity recognition. *arXiv preprint arXiv:2004.04060*.

Chan Hee Song, Dawn Lawrie, Tim Finin, and James Mayfield. 2020. Improving neural named entity recognition with gazetteers. *arXiv preprint arXiv:2003.03072*.

Özlem Uzuner, Imre Solti, and Eithon Cadag. 2010. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17(5):514–518.

Colby Wise, Vassilis N Ioannidis, Miguel Romero Calvo, Xiang Song, George Price, Ninad Kulkarni, Ryan Brand, Parminder Bhatia, and George Karypis. 2020. Covid-19 knowledge graph: accelerating information retrieval and discovery for scientific literature. *arXiv preprint arXiv:2007.12731*.

Hyperbolic Temporal Knowledge Graph Embeddings with Relational and Time Curvatures

Sebastien Montella
Orange Labs, Lannion
Aix Marseille Univ., CNRS, LIS, Marseille, France
`sebastien.montella@orange.com`

Lina Rojas-Barahona
Orange Labs, Lannion

Johannes Heinecke
Orange Labs, Lannion

`{linamaria.rojasbarahona, johannes.heinecke}@orange.com`

Abstract

Knowledge Graph (KG) completion has been excessively studied with a massive number of models proposed for the Link Prediction (LP) task. The main limitation of such models is their insensitivity to time. Indeed, the temporal aspect of stored facts is often ignored. To this end, more and more works consider time as a parameter to complete KGs. In this paper, we first demonstrate that, by simply increasing the number of negative samples, the recent ATTH model can achieve competitive or even better performance than the state-of-the-art on Temporal KGs (TKGs), albeit its nontemporality. We further propose HERCULES, a *time-aware* extension of ATTH model, which defines the curvature of a Riemannian manifold as the product of both relation and time. Our experiments show that both HERCULES and ATTH achieve competitive or new state-of-the-art performances on ICEWS04 and ICEWS05-15 datasets. Therefore, one should raise awareness when learning TKGs representations to identify whether time truly boosts performances.

1 Introduction

The prevalent manner to store factual information is the ⟨*s, p, o*⟩ triple data structure where s, p and o stand for the *subject*, *predicate* and *object* respectively. An *entity* denotes whether a subject or an object while a *relation* denote a predicate that links two entities. A collection of triples defines a Knowledge Graph (KG) noted $\mathcal{G}(\mathcal{E}, \mathcal{R})$ with $\mathcal{E}$ the set of entities, *i.e.* subjects and objects, corresponding to the nodes in the graph and $\mathcal{R}$ the set of predicates corresponding to directed edges. Shedding light on the type of connections between entities, KGs are powerful to work with for numerous downstream tasks such as question-answering (Bordes et al., 2014; Hao et al., 2017; Saxena et al., 2020), recommendation system (Yu et al., 2014; Zhang et al., 2016; Zhou et al., 2017), information retrieval (Lao and Cohen, 2010; Rocktäschel et al., 2015; Xiong et al., 2017), or reasoning (Xian et al., 2019; Chen et al., 2020). However, KGs are sometimes incomplete and part of the knowledge is missing. A major concern was therefore raised to predict missing connections between entities, stimulating research on the Link Prediction (LP) task. Intuition is to map each entity and relation into a vector space to learn low-dimensional embeddings such that valid triples maximize a defined scoring function and that fallacious triples minimize it. An approach is efficient if it can model multiple relational patterns. Some predicates are symmetric (*e.g. marriedTo*), asymmetric (*e.g. fatherOf*), an inversion of another relation (*e.g. fatherOf* and *childOf*) or a composition (*e.g. grandfatherOf*). Distinct strategies were introduced by explicitly model those patterns (Bordes et al., 2013; Yang et al., 2015; Trouillon et al., 2016; Sun et al., 2019). However, hierarchical relations have remained challenging to model in Euclidean space. As demonstrated in Sarkar (2011), tree structures are better embedded in hyperbolic spaces. Thus, hyperbolic geometry reveals to be a strong asset to capture hierarchical patterns. Nevertheless, the aforementioned approaches represent embeddings as invariant to time. For example, while writing this article, the triple ⟨*Donald Trump, presidentOf, U.S.*⟩ is correct but will be erroneous at reading time due to the meantime United States presidential inauguration of Joe Biden. To address this issue, recent works considered using quadruplet written as ⟨*s, p, o, t*⟩ by adding a time parameter t. Then, we note a Tem-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3296–3308
August 1–6, 2021. ©2021 Association for Computational Linguistics

poral Knowledge Graph (TKG) as $\mathcal{G}(\mathcal{E}, \mathcal{R}, \mathcal{T})$ with $\mathcal{T}$ the set of timestamps. Stating a precise time can be advantageous for diverse applications (disambiguation, reasoning, natural language generation, etc). Recent works toward TKG representations are essentially extensions of existing timeless KG embeddings that incorporate the time parameter in the computation of their scoring function.

Similar to our work, Han et al. (2020) developed DYERNIE, an hyperbolic-based model inspired from MURP (Balažević et al., 2019a). DYERNIE uses a product of manifolds and adds a (learned) Euclidean time-varying representation for each entity such that each entity further possesses an *entity-specific velocity vector* along with a static (*i.e.* time-unaware) embedding.

In this paper, we first demonstrate that an optimized number of negative samples enables the ATTH model (Chami et al., 2020) to reach competitive or new state-of-the-art performance on temporal link prediction while being unaware of the temporal aspect. We further introduce HERCULES[1], an extension of ATTH. HERCULES differs from DYERNIE in that:

- Following Chami et al. (2020), we utilize Givens transformations and hyperbolic attention to model different relation patterns.
- A single manifold is used.
- Curvature of the manifold is defined as the product of both relation and time parameters.

To the best of our knowledge, this is the first attempt to leverage the curvature of a manifold to coerce time-aware representation. We also provide an ablation study of distinct curvature definitions to investigate the surprising yet compelling results of ATTH over time-aware models.

2 Related Work

In this section, we present existing methods on both KG and TKG vector representation.

2.1 Timeless Graph Embeddings

Previous works on KG completion essentially focused on undated facts with the ⟨*s, p, o*⟩ formalism. Bordes et al. (2013) initially proposed the TRANSE model considering the relation p as a translation between entities in the embedding space. Several variants were then designed. TRANSH (Wang et al., 2014) adds an intermediate projection onto a relation-specific hyper-plane while TRANSR (Lin et al., 2015) maps entities to a relation-specific space of lower rank. However, translation-based approaches cannot model symmetric relations. DISTMULT (Yang et al., 2015) solves this issue by learning a bilinear objective that attributes same scores to ⟨*s, p, o*⟩ and ⟨*o, p, s*⟩ triples. COMPLEX Trouillon et al. (2016) subsequently came up with complex embeddings. Tensor factorization techniques were also proposed. RESCAL (Nickel and Tresp, 2013) applies a three-way tensor factorization. TUCKER (Balažević et al., 2019b) uses *Tucker decomposition* and demonstrates that TUCKER is a generalization of previous linear models. More recently, ROTATE (Sun et al., 2019) considered relations as rotations in a complex vector space which can represent symmetric relations as a rotation of π. QUATE (Zhang et al., 2019) further generalizes rotations using quaternions, known as *hypercomplex* numbers ($\mathbb{C} \subset \mathbb{H}$). A key advantage of quaternions is its non-commutative property allowing more flexibility to model patterns. Nonetheless, memory-wise, QUATE requires 4 embeddings for each entity and relation. To this extent, hyperbolic geometry provides an outstanding framework to produce shallow embeddings with striking expressiveness (Sarkar, 2011; Nickel and Kiela, 2017). Both MURP (Balažević et al., 2019a) and ATTH (Chami et al., 2020) learn hyperbolic embeddings on a n-dimensional Poincaré ball. Different to MURP, ATTH uses a trainable curvature for each relation. Indeed, Chami et al. (2020) have shown that fixing the curvature of the manifold can jeopardize the quality of the returned embeddings. Therefore, defining a parametric curvature for a given relation helps to learn the best underlying geometry.

2.2 Time-Aware Graph Embeddings

The above-mentioned techniques nevertheless disregard the temporal aspect. Indeed, lets consider the two following quadruplets ⟨*Barack Obama, visits, France, 2009-03-11*⟩ and ⟨*Barack Obama, visits, France, 2014-04-21*⟩. Non-temporal models would exhibit the same score for both facts. However, the second quadruplet is invalid[2] and

---

[1] **H**yperbolic Representation with Tim**E** and **R**elational **CU**rvatures for Tempora**L** Knowledg**E** Graph**S**

[2] In fact, Barack Obama visited Japan on April 21, 2014, not France

should therefore get a lower score. For this reason, several works contributed to obtain time-aware embeddings. Thanks to the existing advancements on graph representations, many strategies are straightforward extensions of static approaches. TTRANSE (Leblay and Chekol, 2018) alters the scoring function of TRANSE to encompass time-related operations such as time translations. Likewise, TA-TRANSE (García-Durán et al., 2018) uses LSTMs (Hochreiter and Schmidhuber, 1997) to encode a temporal predicate which carries the time feature. By analogy with TRANSH, HYTE (Dasgupta et al., 2018) learns time-specific hyperplanes on which both entities and relations are projected. Then, another family of temporal extensions are derived from DISTMULT such as KNOW-EVOLVE (Trivedi et al., 2017), TA-DISTMULT (García-Durán et al., 2018), or TDISTMULT (Ma et al., 2019) that also utilize a bilinear scoring function. DE-SIMPLE (Goel et al., 2020) provides diachronic entity embeddings inspired from diachronic word embeddings (Hamilton et al., 2016). Recently, ATISE (Xu et al., 2019) embeds entities and relations as a multi-dimensional Gaussian distributions which are time-sensitive. An advantage of ATISE is its ability to represent time uncertainty as the covariance of the Gaussian distributions. TERO (Xu et al., 2020b) combines ideas from TRANSE and ROTATE. It defines relations as translations and timestamps as rotations. As far as we are aware, DYERNIE (Han et al., 2020) is the first work to contribute to hyperbolic embeddings for TKG. It achieves state-of-the-art performances on the benchmark datasets ICEWS14 and ICEWS05-15. Time is defined as a translation on a product of manifolds with trainable curvatures using a *velocity vector* for each entity.

In our work, we demonstrate that using a single manifold with learnable relational and time curvatures is sufficient to reach competitive or new state-of-the-art performances.

3 Problem Definition

Lets consider a valid quadruplet $\langle s,\ p,\ o,\ t\rangle \in \mathcal{S} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$, with $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ the sets of entities, relations and timestamps respectively and $\mathcal{S}$ the set of correct facts. A scoring function $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T} \rightarrow \mathbb{R}$ is defined such that $f(s, p, o, t)$ is maximized for any quadruplet $\in \mathcal{S}$, and minimized for corrupted quadruplet ($\notin \mathcal{S}$). Throughout the optimization of the foregoing constraint, representations of entities, relations and times are learned accordingly. The resulting embeddings should then capture the multi-relational graph structure. Thus, f is gauging the probability that an entity s is connected to an entity o by the relation p at time t.

4 Hyperbolic Geometry

Hyperbolic geometry belongs to non-Euclidean geometry. In contrast to Euclidean geometry relying on Euclid's axioms (Heath and Euclid, 1956), non-Euclidean geometry rejects the fifth axiom known as the parallel postulate. It states that given a point x and a line l_1, there exists a unique line l_2 parallel to l_1 passing through x. This is only possible due to a (constant) zero curvature of the space. The curvature defines how much the geometry differs from being flat. The higher the absolute curvature, the curvier. Euclidean space has a zero curvature hence called *flat space*. When represented in an Euclidean space, straight lines become curved, termed as *geodesics* (Fig. 1).

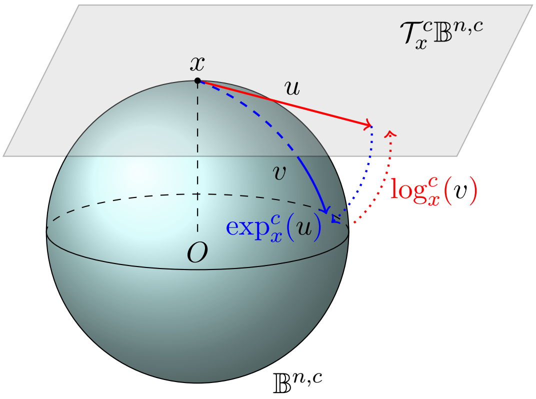

Figure 1: Illustration of the exponential and logarithmic maps between the Poincaré ball $\mathbb{B}^{n,c}$ and the tangent space $\mathcal{T}_x^c\mathbb{B}^{n,c}$.

Hyperbolic geometry comes with a constant negative curvature. In our study, as Nickel and Kiela (2017); Han et al. (2020); Chami et al. (2020), we make use of the Poincaré ball ($\mathbb{B}^{n,c}$, $g^{\mathbb{B}}$) which is a n-dimensional Riemannian manifold $\mathbb{B}^{n,\,c} = \{x \in \mathbb{R}^n : \|x\|^2 < \frac{1}{c}\}$ of constant curvature $-c$ ($c > 0$) equipped with Riemannian metric $g^{\mathbb{B}}$ and where $\|.\|$ denotes the L_2 norm. The metric tensor $g^{\mathbb{B}}$ gives information about how distances should be computed on the manifold and is defined as $g^{\mathbb{B}} = (\lambda_x^c)^2 I_n$ with the *conformal factor* $\lambda_x^c = \frac{2}{1-c\|x\|^2}$ and I_n the identity matrix of size $n \times n$. We write $\mathcal{T}_x^c\mathbb{B}^{n,c}$ the n-dimensional tangent space at $x \in \mathbb{B}^{n,c}$. At the difference of $\mathbb{B}^{n,c}$, $\mathcal{T}_x^c\mathbb{B}^{n,c}$ locally follows an Euclidean geometry. As illustrated in Fig. 1, we can project $v \in \mathbb{B}^{n,c}$ on $\mathcal{T}_x^c\mathbb{B}^{n,c}$ at x via the logarithmic map. Inversely, we can map a point $u \in \mathcal{T}_x^c\mathbb{B}^{n,c}$ on $\mathbb{B}^{n,c}$ at x via the exponential map. A closed-form of those mappings exist when x corresponds to the origin (Eqs. 1 and 2).

$$\exp_{\mathbf{0}}^c(u) = \tanh(\sqrt{c}\|u\|)\frac{u}{\sqrt{c}\|u\|} \tag{1}$$

$$\log_{\mathbf{0}}^c(v) = \tanh^{-1}(\sqrt{c}\|v\|)\frac{v}{\sqrt{c}\|v\|} \tag{2}$$

Contrary to $\mathcal{T}_x^c\mathbb{B}^{n,c}$, the Euclidean addition on the hyperbolic manifold does not hold. Alternately, we use the Möbius addition (Vermeer, 2005; Ganea et al., 2018) satisfying the boundaries constraints of the manifold. It is however non-commutative and non-associative. The closed-form is presented in Eq. 3.

$$x \oplus_c y = \frac{(1 - 2cx^T y - c\|y\|^2)x + (1 + c\|x\|^2)y}{1 - 2cx^T y + c^2\|y\|^2\|x\|^2} \tag{3}$$

The distance between two points x and y on $\mathbb{B}^{n,c}$ is the hyperbolic distance $d_{\mathbb{B}^{n,c}}(x, y)$ defined as:

$$d_{\mathbb{B}^{n,c}}(x, y) = \frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|-x \oplus_c y\|) \tag{4}$$

5 From ATTH to HERCULES

Given a quadruplet, ⟨*s, p, o, t*⟩, we note e_s^H, r_p^H and e_o^H the hyperbolic embeddings of the subject, predicate and object respectively.[3] ATTH uses relation-specific embeddings, rotations, reflections and curvatures. The curvature is defined as depending on the corresponding relation *p* involved. Precisely, a relation *p* is attributed with an individual parametric curvature c_p. The curvature c_p is defined in Eq. 5 as:

$$c_p = \sigma(\mu_p) \tag{5}$$

where μ_p is a trainable parameter $\in \mathbb{R}$ and σ is a smooth approximation of the ReLU activation function defined in $[0, +\infty]$. With such approach, the geometry of the manifold is learned, thus modified for a particular predicate. The curvature dictates how the manifold is shaped. Changing the curvature of the manifold implies changing the positions of projected points. This means that for distinct relations, the same entity will have different positions because of the different resulting geometries for each relation. For example, lets consider the triples t_1 := ⟨*Barack Obama, visit, France*⟩ and t_2 := ⟨*Barack Obama, cooperate, France*⟩. The Euclidean representations of entities *Barack Obama* and *France* from both facts will be projected onto the riemannian manifold. However, the structure (i.e. curvature) of the manifold changes as a function of the relation of each fact (*i.e.* '*visit*' and '*cooperate*'). Therefore, the resulting hyperbolic embbeding of *Barack Obama* of t_1 will not be the same resulting hyperbolic embedding of *Barack Obama* in t_2. By analogy, the same holds for entity *France*.

In order to learn rotations and reflections, ATTH uses 2 × 2 Givens transformations matrices. Those transformations conserve relative distances in hyperbolic space and can therefore directly be applied to hyperbolic embeddings (isometries). We note $W_{\Theta_p}^{rot}$ and $W_{\Phi_p}^{ref}$ the block-diagonal matrices where each element on their diagonals is given by $G^+(\theta_{p,i})$ and $G^-(\phi_{p,i})$ respectively, with i the i^{th} element of the diagonal (Eqs. 6 and 7).

$$G^+(\theta_{p,i}) = \begin{bmatrix} \cos(\theta_{p,i}) & -\sin(\theta_{p,i}) \\ \sin(\theta_{p,i}) & \cos(\theta_{p,i}) \end{bmatrix} \tag{6}$$

$$G^-(\phi_{p,i}) = \begin{bmatrix} \cos(\phi_{p,i}) & \sin(\phi_{p,i}) \\ \sin(\phi_{p,i}) & -\cos(\phi_{p,i}) \end{bmatrix} \tag{7}$$

Then, the rotations and reflections are applied only to the subject embedding as describe in Eq. 8.

$$q_{rot}^H = W_{\Theta_p}^{rot} e_s^H \qquad q_{ref}^H = W_{\Phi_p}^{ref} e_s^H \tag{8}$$

Furthermore, to represent complex relations that can be a mixture of rotation and reflection, ATTH utilizes an hyperbolic attention mechanism. The attention scores $\alpha_{q_{rot}^H}^p$ and $\alpha_{q_{ref}^H}^p$ are computed in the tangent space by projecting the hyperbolic rotation embedding q_{rot}^H and hyperbolic reflection embedding q_{ref}^H with the logarithmic map (Eq. 2). More specifically, ATTH implements a tangent space average to implement the typical weighted average as proposed in Liu et al. (2019) and Chami et al. (2019). Then, the attention vector is mapped back to manifold using the exponential map (Eq. 1). We

---

[3] Since ATTH is not considering time, the parameter t is not used.

have then:

$$ATT(q^H_{rot}, q^H_{ref}, p) = \exp^{c_p}_{\mathbf{0}}(\alpha_{q^H_{rot}} \log^{c_p}_{\mathbf{0}}(q^H_{rot}) + \alpha_{q^H_{ref}} \log^{c_p}_{\mathbf{0}}(q^H_{ref})) \tag{9}$$

AttH finally applies a translation of the hyperbolic relation embedding r^H_p over the resulting attention vector (Eq. 9). As mentioned in Chami et al. (2020), translations help to move between different levels of the hierarchy.

$$Q(s,p) = ATT(q^H_{rot}, q^H_{ref}, p) \oplus_{c_p} r^H_p \tag{10}$$

The scoring function is similar to the one used in Balažević et al. (2019a) and Han et al. (2020) defined as:

$$s(s,p,o,t) = -d^{c_p}(Q(s,p), e^H_o)^2 + b_s + b_o \tag{11}$$

where b_s and b_o stand for the subject and object biases, respectively.

We then propose Hercules, a time-aware extension of AttH. Hercules redefines the curvature of the manifold as being the product of both relation and time as illustrated in Eq. 12:

$$c^t_p = \sigma(\mu_p \times \tau_t) \tag{12}$$

with τ_t a learnable parameter $\in \mathbb{R}$. A trade-off therefore exists between relation and time. A low value of either μ_p or τ_t will lead to a flatter space while higher values will tend to a more hyperbolic space. The main intuition of Hercules is that both relation and time directly adjust the geometry of the manifold such that the positions of projected entities are relation-and-time-dependent. This is advantageous in that no additional temporal parameters per entity are needed. Since the whole geometry has changed for specific relation and time, all future projections onto that manifold will be aligned to the corresponding relation and timestamp. We investigate different curvature definitions and time translation in our experiments (see Section 6). The scoring function of Hercules remains same as AttH.

When learning hyperbolic parameters, the optimization requires to utilize a Riemannian gradient (Bonnabel, 2013). However, proven to be challenging, we instead learn all embeddings in the Euclidean space. The embeddings can then be mapped to the manifold using the exponential map (Eq. 1). This allows the use of standard Euclidean optimization strategies.

6 Experiments

We outline in this section the experiments and evaluation settings.

6.1 Datasets

For fair comparisons, we test our model on same benchamark datasets used in previous works, *i.e.* ICEWS14 and ICEWS05-15. Both datasets were constructed by García-Durán et al. (2018) using the Integrated Crisis Early Warning System (ICEWS) dataset (Boschee et al., 2018). ICEWS provides geopolitical information with their corresponding (event) date, *e.g.* ⟨*Barack Obama, visits, France, 2009-03-11*⟩. More specifically, ICEWS14 includes events that happened in 2014 whereas ICEWS05-15 encompasses facts that appeared between 2005 and 2015. We give the original datasets statistics in Table 2. To increase the number of samples, for each quadruplet ⟨*s, p, o, t*⟩ we add ⟨*s,* p^{-1}*, o, t*⟩, where p^{-1} is the inverse relation of *p*. This is a standard data augmentation technique usually used in LP (Balažević et al., 2019a; Goel et al., 2020; Han et al., 2020).

6.2 Evaluation Protocol & Metrics

Given a (golden) test triple ⟨*s, p, o, t*⟩, for each entity $s' \in \mathcal{E}$, we interchange the subject *s* with s' and apply the scoring function f on the resulting query ⟨s'*, p, o, t*⟩. Since replacing *s* by all possible entity s' may end up with a correct facts, we filter out those valid quadruplets and give them extremely low scores to avoid correct quadruplets to be scored higher than the tested quadruplet in final ranking (Bordes et al., 2013). We then rank the entities based on their scores in descending order. We store the rank of the correct entity *s* noted z_s. Thus, the model should maximize the returned score for the entity *s* such that $z_s = 1$. The same process is done using the object *o*.

To evaluate our models, we make use of the Mean Reciprocal Rank (MRR). We also provide the Hits@1 (H@1), Hits@3 (H@3) and Hits@10 (H@10) which assess on the frequency that the valid entity is in the top-1, top-3 and top-10 position, respectively.

6.3 Implementation Details

To ensure unbiased comparability, the same training procedure and hyper-parameters are shared for

Model	**Parameters**
TTransE	$(\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert + \lvert\mathcal{T}\rvert) \cdot n$
TComplEx	$(2\lvert\mathcal{E}\rvert + 4\lvert\mathcal{R}\rvert + 2\lvert\mathcal{T}\rvert) \cdot n$
HyTE	$(\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert + \lvert\mathcal{T}\rvert) \cdot n$
ATISE	$(\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert) \cdot n$
TERO	$(\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert + \lvert\mathcal{T}\rvert) \cdot n$
DyERNIE	$2 \cdot (\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert) \cdot n + 2 \cdot \lvert\mathcal{E}\rvert$
AttH	$(\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert) \cdot n + \lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert(1 + 3n)$
Hercules	$(\lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert) \cdot n + \lvert\mathcal{E}\rvert + 2\lvert\mathcal{R}\rvert(1 + 3n) + \lvert\mathcal{T}\rvert$

Table 1: Number of parameters per models with respect to the embedding dimension n.

Datasets	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	**Training**	**Validation**	**Test**
ICEWS14	7,128	230	365	72,128	8,941	8,963
ICEWS05-15	10,488	251	4017	368,962	46,275	46,092

Table 2: ICEWS14 and ICEWS05-15 Datasets Statistics

AttH and Hercules.[4] Number of epochs and batch size were set to 500 and 256, respectively. We minimized the cross-entropy loss using negative sampling, where negative samples corrupt the valid object only (uniformly selected). After validation, we noticed that best results were obtained using 500 negative samples. We chose the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001. The final models for evaluation were selected upon the MRR metric on the validation set. We re-train ATISE and TeRo using the same parameters as mentionned in Xu et al. (2019) and Xu et al. (2020b) but varying dimensions.[5]

6.4 Results

We report models performances on the link prediction task on TKGs. Additional analyses on the results are also given.

6.4.1 Link Prediction results

We provide link prediction results on ICEWS14 and ICEWS05-15 for AttH, Hercules and different models from the literature. As Han et al. (2020), we adopted a dimension analysis to investigate behaviors and robustness of approaches. When possible, we re-run official implementation of models. Otherwise, official or best results in literature are reported. Results are shown in Table 3.

As expected, hyperbolic-based strategies (*i.e.* DyERNIE, AttH and Hercules) perform much better at lower dimensions, outperforming most of other approaches with ten times less dimensions. We report an average absolute gain of 11.6% points in MRR with only 10 dimensions over the median performance of other approaches with 100 dimensions. This strengthens the effectiveness of hyperbolic geometry to induce high-quality embeddings with few parameters.

Astonishingly, we notice that AttH model is highly competitive despite the absence of time parameter. AttH exhibits new state-of-the-art or statistically equivalent performances compared to DyERNIE and Hercules. We remark no statistically significant differences in performances between hyperbolic models.[6] Importantly, unlike other research carried out in this area, time information here does not lead to any notable gain. This seems to indicate that other parameters should be considered. We examine this phenomenon in section 6.4.2.

On ICEWS14, for *dim* $\in \{20, 40, 100\}$, both AttH and Hercules outperform DyERNIE by a large margin. We witness an improvement of 2.5% and 5% points in MRR and Hits@1 with 100-dimensional embeddings. On ICEWS05-15, AttH and Hercules yield comparable achievements with the state-of-the-art. In contrast to DyERNIE,

---

[4] We used the official implementation of AttH available at `https://github.com/HazyResearch/KGEmb`. We adapted it to implement Hercules.

[5] We used the official implementation available at `https://github.com/soledad921/ATISE`

[6] We performed the Mixed-Factorial Analysis of Variance (ANOVA), in which the independent variables are *the dimension* and *the model* and the dependent variable is *the metric*. We consider two groups one for each dataset. We report p-values of 0.842, 0.872, 0.926 and 0.229 for MRR, H@1, H@3 and H@10 respectively.

it is noteworthy that ATTH and HERCULES utilize a single manifold while reaching top performances.

We also distinguish tempered results on Hits@10 metric for ATTH and HERCULES models. This suggests that during optimization, ATTH and HERCULES favor ranking some entities on top while harming the representation of others.

6.4.2 Is Time *All* You Need ?

In this section, we investigate the influence of the temporal parameter on performances.

First, besides time translation, we probe different curvature definitions to identify fluctuation in performances. We analyze how time information alters the LP results by adding time as part of the curvature (*i.e.* HERCULES) and as a translation. We also explore if incorporating the Euclidean dot product of the subject and object embeddings (noted $\langle e_s^{\mathbb{E}}, e_o^{\mathbb{E}} \rangle$) into the curvature helps to learn a better geometry. An ablation study is given in Table 4.

Albeit counter-intuitive, we observe that our results corroborate with our initial finding: time information is not the culprit of our high performances. More strikingly, a simple relational curvature (*i.e.* ATTH) is sufficient to perform best on ICEWS14 (*dim* = 40). Neither the inclusion of a time translation, similarly to TTRANSE, nor the Euclidean dot product provide interesting outcomes.

We then probe the sensitivity of HERCULES towards temporal feature by performing LP with incorrect timestamps. Our intuition is to inspect whether feeding invalid timestamps during evaluation exhibits significant variation or not compared to the reference performances, *i.e.* LP results with initial (non-corrupted) testing samples. To do so, for each testing quadruplet, we replace the (correct) time parameter with each possible timestamp from $\mathcal{T}$. We therefore collect multiple LP performances of HERCULES corresponding to each distinct timestamp. We plot the distribution of resulting performances of temporally-corrupted quadruplets from ICEWS14 (*dim*=100) in Fig. 2. We can observe that despite erroneous timestamps, LP results show insignificant discrepancies with the initial HERCULES performance (dashed red line). The standard deviations from HERCULES reference performance for MRR, H@1, H@3, H@10 metrics are 1.78×10^{-4}, 3.18×10^{-3}, 1.05×10^{-2}, 3.55×10^{-3} respectively. This indicates that HERCULES gives little importance to the time parameter and thus only relies on the entity and the predicate to perform knowledge graph completion. This further highlights our finding that timestamp is not responsible for our attracting performances.

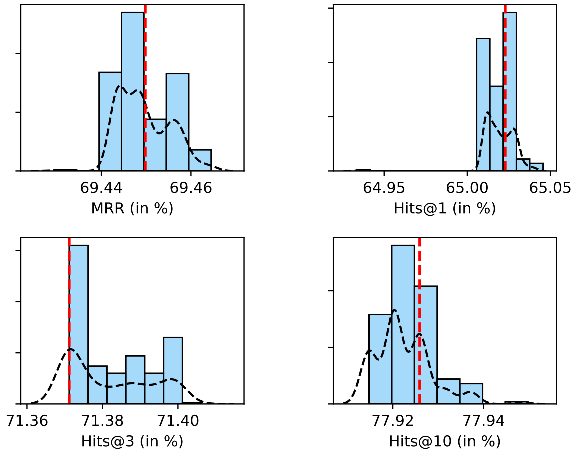

Figure 2: Distribution of performances of HERCULES on temporally-corrupted quadruplets from ICEWS14 with *dim*=100. A smooth approximation of the distribution is drawn as a dashed black curve. The reference performance is indicated with a dashed red line.

We therefore assume that the optimization procedure may be involved. We consequently question the effect of negative sampling. Precisely, we train HERCULES with *dim* = 40 by tuning the number of negative samples between 50 to 500. We plot the learning curves in Fig. 3.

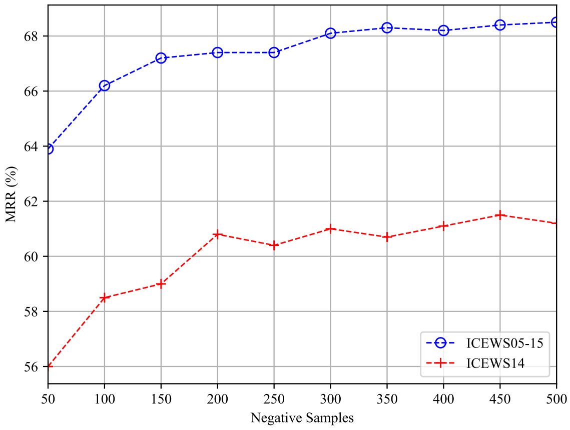

Figure 3: Performance of HERCULES (*dim* = 40) on ICEWS14 and ICEWS05-15 with varying number of negative samples.

For both, ICEWS14 and ICEWS05-15, negative sampling shows considerable gain as the number of samples increases. We record an absolute gain of 5% points in MRR from 50 to 500 samples. We can see a rapid growth in MRR when the number of samples is inferior to 200. Adding 50 samples is equivalent to about 2% points gain in MRR. Then,

Datasets		ICEWS14 (filtered)				ICEWS05-15 (filtered)			
dim	Model	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
10	ATISE†	18.0	3.03	23.9	48.7	15.9	4.35	19.22	41.0
	TeRo†	7.25	2.39	6.40	16.6	10.3	3.54	10.1	23.2
	DyERNIE*	**46.2**	**36.0**	51.1	66.3	**58.9**	**50.5**	**63.2**	**75.1**
	Hercules	46.0	34.9	**52.4**	66.0	54.7	43.8	61.8	73.2
	AttH	45.6	34.2	52.0	**66.4**	49.9	34.4	61.6	73.6
20	ATISE†	19.1	1.28	28.2	54.7	24.5	7.67	32.3	59.2
	TeRo†	24.5	13.8	28.01	46.3	27.1	13.5	33.3	54.1
	DyERNIE*	53.9	44.2	58.9	**72.7**	**64.2**	**56.5**	**68.2**	**79.0**
	Hercules	**55.5**	**47.2**	59.4	71.4	63.2	55.2	67.7	77.6
	AttH	55.2	46.7	**59.7**	71.4	63.5	55.8	67.7	77.5
40	ATISE†	38.4	23.3	47.6	67.3	35.7	19.2	44.3	69.1
	TeRo†	35.1	22.7	40.5	60.8	28.3	12.7	35.3	60.5
	DyERNIE*	58.8	49.8	63.8	**76.1**	**68.9**	61.8	**72.8**	**82.5**
	Hercules	61.2	54.3	64.7	74.1	68.5	**62.1**	72.0	80.9
	AttH	**61.7**	**54.5**	**65.4**	75.4	68.5	62.0	71.9	80.6
100	TransE*	30.0	14.8	42.7	60.1	30.4	13.3	42.4	61.1
	DistMult*	57.5	46.9	64.2	77.9	47.1	33.6	55.1	72.5
	ComplEx*	49.3	36.6	56.2	74.2	39.0	22.9	49.2	68.4
	TTransE*	34.4	25.7	38.3	51.3	35.6	15.4	51.1	67.6
	TComplEx*	31.8	12.9	45.7	63.0	45.1	36.3	49.2	62.0
	HyTE*	33.1	6.8	54.5	73.6	38.1	7.6	65.0	80.4
	ATISE†	52.2	41.0	60.0	72.7	47.0	32.4	55.5	76.4
	TeRo†	45.4	34.0	52.2	67.0	41.1	26.3	48.9	71.7
	DyERNIE*	66.9	59.9	71.4	**79.7**	**73.9**	67.9	**77.3**	**85.5**
	Hercules	69.4	**65.0**	71.4	77.9	73.5	**68.6**	76.1	82.9
	AttH	**69.5**	**65.0**	**71.5**	78.2	73.6	**68.6**	76.0	82.9

Table 3: Link prediction results on ICEWS14 and ICEWS05-15 datasets: ($\dagger$) results are obtained using the official implementation of Xu et al. (2020a), ($*$) results are taken from Han et al. (2020). For each dimension (*i.e. dim*), best results are in bold and second-to-best underlined. No statistically significant differences in performance are observed between DyERNIE, Hercules and AttH.[6]

Relation Curvature	Time Curvature	Time Translation	$\langle e_s^{\mathbb{E}}, e_o^{\mathbb{E}} \rangle$ Curvature	MRR	H@1	H@3	H@10
✓	✗	✗	✗	61.7	54.5	65.4	75.4
✓	✓	✗	✗	61.2	54.3	64.7	74.1
✓	✓	✓	✗	60.1	52.1	64.5	75.0
✓	✓	✓	✓	49.5	38.9	55.4	69.2

Table 4: Ablation study: Link prediction results on ICEWS14 using AttH (*dim* = 40) with different curvature definitions and time translation applied.

performances reach a plateau around 300 negative samples. We conjecture that a diversity in negative samples is enough to learn good representations. Notwithstanding that a large number of negative samples heavily constraints the location of entities in space, the resulting embeddings might benefit from it to be better positioned relatively to others.

We conclude that despite the present time parameter, an optimal negative sampling enables to reach new state-of-the-art outcome. Therefore, we argue that time is not the only parameter that should be considered when performing LP. We highlight that one should be raising awareness when training TKG representations to identify if time is truly helping to boost performances.

6.4.3 ATTH versus HERCULES

We explore here how the geometry of HERCULES differs from ATTH. To do so, we inspect the absolute difference of their learned curvatures Δ_c. We plot Δ_c with respect to the relations and timestamps for *dim* = 40 on ICEWS14 in Fig. 4.[7] Similar plots are given for ICEWS05-15 in Appendix A.1.

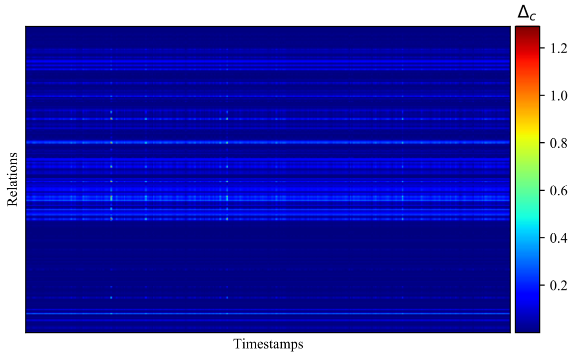

Figure 4: Absolute curvatures differences between ATTH and HERCULES given specific relation and time on ICEWS14 dataset (*dim* = 40).

Besides rare steep discrepancy in curvatures (*i.e.* $\Delta_c > 1.0$), HERCULES is akin to ATTH concerning learned geometries. We report that 85.0% and 95.6% of Δ_c's are smaller than 0.1 on ICEWS14 and ICEWS05-15 respectively. We point out that some timestamps affect *globally* all relations, albeit very limited. This can be seen in Fig. 5 by the aligned vertical strips. It indicates that HERCULES uses its additional time parameter to learn a slightly different manifold but nonetheless quite similar to ATTH.[8] We provide further analysis on the shifts of the curvatures while increasing embeddings dimension in Appendices A.2 and A.3.

We depict an example of learned hyperbolic representation of ICEWS14 entities of ATTH and HERCULES for the relation '*make a visit*' and timestamp set to 01-01-2014 in Fig. 5. We plot embeddings of ICEWS05-15 in Appendix A.4.

7 Conclusion

In this paper, we have demonstrated that without adding neither time information nor supplementary parameters, the ATTH model astonishingly achieves similar or new state-of-the-art performances on link prediction upon temporal knowledge graphs. In spite of the inclusion of time with our proposed time-aware model HERCULES, we have shown that negative sampling is sufficient to learn a good underlying geometry. In the future, we plan to explore new mechanisms to incorporate temporal information to improve performances of ATTH.

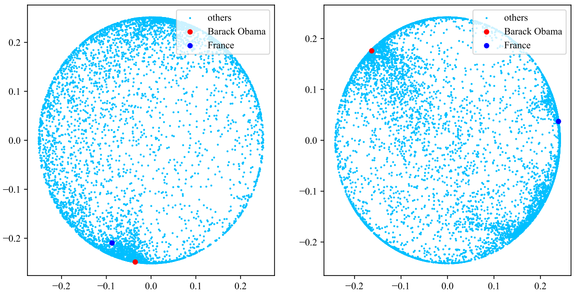

Figure 5: Illustration of two-dimensional hyperbolic entity embeddings learned by ATTH (left) and HERCULES (right) on ICEWS14 for predicate '*make a visit*' and timestamp set to 01-01-2014.

References

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019a. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*.

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019b. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Silvère Bonnabel. 2013. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58:2217–2229.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, and James Starz. 2018. *ICEWS Weekly Event Data*. Harvard Dataverse.

---

[7] For readability, we don't plot Δ_c for reverse relations p^{-1} (see Section 6.1).

[8] Similar geometry $\neq$ Similar embeddings

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 4868–4879. Curran Associates, Inc.

Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, Brussels, Belgium. Association for Computational Linguistics.

Octavian Ganea, Gary Becigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 5345–5355. Curran Associates, Inc.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium. Association for Computational Linguistics.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *AAAI*.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020. DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7301–7316, Online. Association for Computational Linguistics.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231, Vancouver, Canada. Association for Computational Linguistics.

Thomas L. Heath and Euclid. 1956. *The Thirteen Books of Euclid's Elements, Books 1 and 2*. Dover Publications, Inc., USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1771–1776, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2181–2187. AAAI Press.

Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 8230–8241. Curran Associates, Inc.

Yunpu Ma, Volker Tresp, and Erik A. Daxberger. 2019. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, 59:100490.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6341–6350. Curran Associates, Inc.

Maximilian Nickel and Volker Tresp. 2013. Tensor factorization for multi-relational learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 617–621, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 1119–1129, Denver, Colorado. Association for Computational Linguistics.

Rik Sarkar. 2011. Low distortion delaunay embedding of trees in hyperbolic plane. In *Proceedings of the 19th International Conference on Graph Drawing*, GD'11, page 355–366, Berlin, Heidelberg. Springer-Verlag.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3462–3471. JMLR.org.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org.

J. Vermeer. 2005. A geometric interpretation of ungar's addition and of gyration in the hyperbolic plane. *Topology and its Applications*, 152(3):226 – 242. Special Issue Jan Aarts.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1112–1119. AAAI Press.

Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 285–294, New York, NY, USA. Association for Computing Machinery.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1271–1279, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. 2019. Temporal knowledge graph embedding model based on additive time series decomposition. *arXiv preprint arXiv:1911.07893*.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020a. TeRo: A time-aware knowledge graph embedding via temporal rotation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1583–1593, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020b. Tero: A time-aware knowledge graph embedding via temporal rotation. *arXiv preprint arXiv:2010.01029*.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.

Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, page 283–292, New York, NY, USA. Association for Computing Machinery.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 353–362, New York, NY, USA. Association for Computing Machinery.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embedding. *arXiv preprint arXiv:1904.10281*.

Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

A Appendices

A.1 ATTH versus HERCULES

We plot the absolute difference in curvatures between ATTH and HERCULES for *dim* = 40 on ICEWS05-15 in Fig. 6. As in ICEWS14 dataset, we observe that learned geometries on ICEWS15 dataset by ATTH and HERCULES are alike. Time is showing insubstantial impact on the curvature.

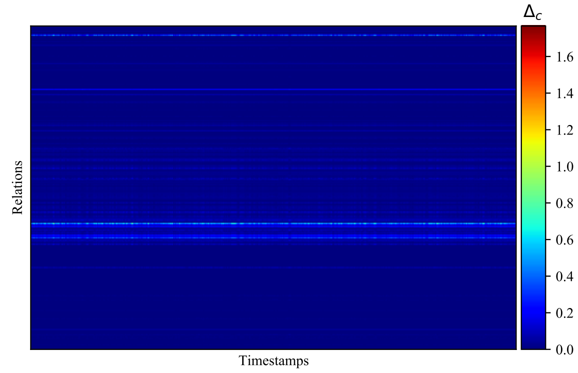

Figure 6: Absolute curvatures differences between AttH and Hercules given specific relation and time on ICEWS05-15 dataset (*dim* = 40).

A.2 AttH versus AttH

We inspect how the curvature of AttH fluctuates while increasing the embedding dimension. We compare curvatures shifts between *dim* = 40 and *dim* = 100. We plot Δ_c in Fig. 7.

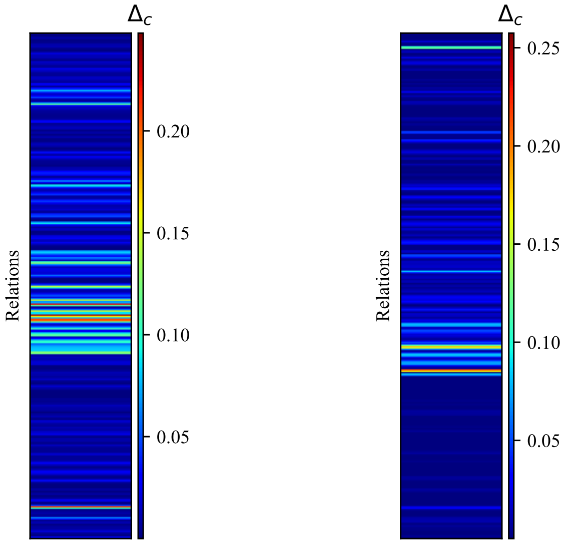

Figure 7: Absolute curvatures differences of AttH between *dim* = 40 and *dim* = 100 given specific relation and time on ICEWS14 dataset (left) and ICEWS05-15 dataset (right). No timestamps since AttH is a time-unaware model.

The fluctuation in curvature is moderate while dimension of embeddings increases. We can see that variation is almost inferior to 0.2. This underlines that geometry does not change much for a specific relation when the dimension grows. For some relations, we note however that curvature is knowing significant dissimilitudes.

A.3 Hercules versus Hercules

We report the dissimilarities in curvature for different relations and timestamps on ICEWS14 and ICEWS05-15 in Fig. 8 and Fig. 9.

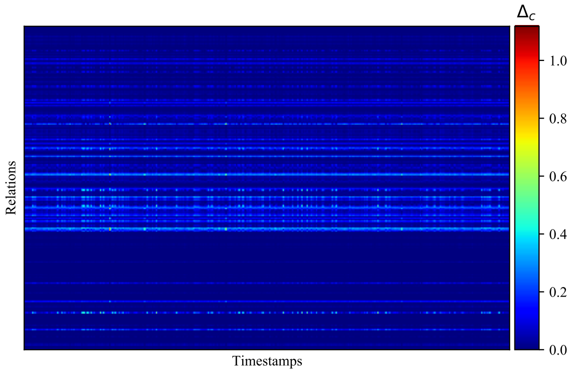

Figure 8: Absolute curvatures differences of Hercules between *dim* = 40 and *dim* = 100 given specific relation and time on ICEWS14 dataset.

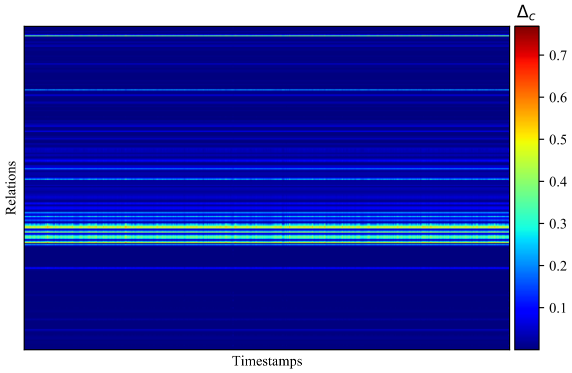

Figure 9: Absolute curvatures differences of Hercules between *dim* = 40 and *dim* = 100 given specific relation and time on ICEWS05-15 dataset.

We note that around 87.7% and 91.1% of variations are smaller than 0.1 on ICEWS14 and ICEWS05-15 respectively. This seems to indicate that despite the dimensionality gap, the learned geometry for each relation does not differ much between *dim* = 40 and *dim* = 100.

A.4 Two-Dimensional Hyperbolic Embeddings

We give an illustration of learned embeddings on ICEWS05-15 by AttH and Hercules models in Fig. 10.

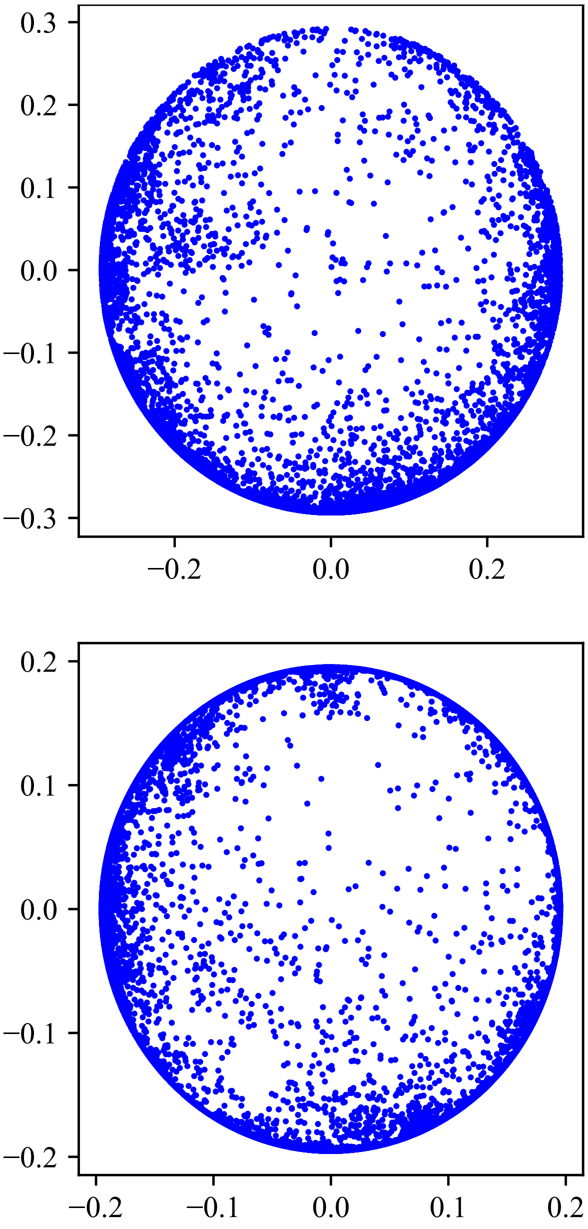

Figure 10: Illustration of two-dimensional hyperbolic embeddings learned by AttH (top) and Hercules (bottom) on ICEWS05-15.

DISFL-QA: A Benchmark Dataset for Understanding Disfluencies in Question Answering

Aditya Gupta[♠] **Jiacheng Xu**[◇*] **Shyam Upadhyay**[♠] **Diyi Yang**[♣] **Manaal Faruqui**[♠]
[♠]Google Assistant
[◇]The University of Texas at Austin
[♣]Georgia Institute of Technology
disfl-qa@google.com

Abstract

Disfluencies is an under-studied topic in NLP, even though it is ubiquitous in human conversation. This is largely due to the lack of datasets containing disfluencies. In this paper, we present a new challenge question answering dataset, DISFL-QA, a derivative of SQUAD, where humans introduce contextual disfluencies in previously fluent questions. DISFL-QA contains a variety of challenging disfluencies that require a more comprehensive understanding of the text than what was necessary in prior datasets. Experiments show that the performance of existing state-of-the-art question answering models degrades significantly when tested on DISFL-QA in a zero-shot setting. We show data augmentation methods partially recover the loss in performance and also demonstrate the efficacy of using gold data for fine-tuning. We argue that we need large-scale disfluency datasets in order for NLP models to be robust to them. The dataset is publicly available at: `https://github.com/google-research-datasets/disfl-qa`.

1 Introduction

During conversations, humans do not always premeditate exactly what they are going to say; thus a natural conversation often includes interruptions like repetitions, restarts, or corrections. Together these phenomena are referred to as *disfluencies* (Shriberg, 1994). Figure 1a shows different types of conventional disfluencies in an utterance, as described by Shriberg (1994).

Repetition	When is Eas ugh Easter this year?
Correction	When is Lent I meant Easter this year?
Restarts	How much no wait when is Easter this year?

(a) Conventional categories of *Disfluencies*. The ***reparandum*** (words intended to be corrected or ignored), ***interregnum*** (optional discourse cues) and ***repair*** are marked.

Passage: *The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, …*

q_1: In what country is Normandy located?
dq_1: In what country is **Norse** found **no wait** **Normandy not Norse**?
T5(q_1): France ✓
T5(dq_1): Denmark ✗

q_2: When were the Normans in Normandy?
dq_2: **From which countries no tell me when were the Normans in Normandy**?
T5(q_2): 10th and 11th centuries ✓
T5(dq_2): Denmark, Iceland and Norway ✗

(b) Contextualized *Disfluencies* in DISFL-QA (§2).

Figure 1: **(a)** Categories of disfluencies (Shriberg, 1994) **(b)** A passage and questions (q_i) from SQUAD, along with their disfluent versions (dq_i) and predictions from a T5-QA model.

With the growing popularity of voice assistants, such disfluencies are of particular interest for goal-oriented or information seeking dialogue agents, because an NLU system, trained on fluent data, can easily get misled due to their presence. Figure 1b shows how the presence of *disfluencies* in a question answering (QA) setting, namely SQUAD (Rajpurkar et al., 2018), affects the prediction of a state-of-the-art T5 model (Raffel et al., 2020). For example, the original question q_1 is seeking an answer about the location of *Normandy*. In the disfluent version dq_1 (which is **semantically equivalent** to q_1), the user starts asking about *Norse* and then corrects themselves to ask about the *Normandy* instead. The presence of this correctional disfluency confuses the QA model, which tend to rely on shallow textual cues from question for making predictions.

---

[*] Work done during an internship at Google.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3309–3319
August 1–6, 2021. ©2021 Association for Computational Linguistics

Unfortunately, research in NLP and speech community has been impeded by the lack of curated datasets containing such disfluencies. The datasets available today are mostly conversational in nature, and span a limited number of very specific domains (e.g., telephone conversations, court proceedings) (Godfrey et al., 1992; Zayats et al., 2014). Furthermore, only a small fraction of the utterances in these datasets contain disfluencies, with a limited and skewed distribution of disfluencies types. In the most popular dataset in the literature, the SWITCHBOARD corpus (Godfrey et al., 1992), only 5.9% of the words are *disfluencies* (Charniak and Johnson, 2001), of which > 50% are *repetitions* (Shriberg, 1996), which has been shown to be the relatively simpler form of disfluencies (Zayats et al., 2014; Jamshid Lou et al., 2018; Zayats et al., 2019).

To fill this gap, we present DISFL-QA, the first dataset containing *contextual disfluencies* in an information seeking setting, namely question answering over Wikipedia passages. DISFL-QA is constructed by asking human raters to insert disfluencies in *questions* from SQUAD-v2, a popular question answering dataset, using the passage and remaining questions as context. These contextual disfluencies lend naturalness to DISFL-QA, and challenge models relying on shallow matching between question and context to predict an answer. Some key properties of DISFL-QA are:

- DISFL-QA is a targeted dataset for disfluencies, in which all questions (≈12k) contain disfluencies, making for a much larger disfluent test set than prior datasets.
- Over 90% of the disfluencies in DISFL-QA are corrections or restarts, making it a much harder test set for disfluency correction (§2.2).
- DISFL-QA contains wider diversity in terms of semantic distractors than earlier disfluency datasets, and newer phenomenon such as coreference between the *reparandum* and the *repair* (§2.3).

We experimentally reveal the brittleness of state-of-the-art LM based QA models when tested on DISFL-QA in zero-shot setting (§4.1). Since collecting large supervision datasets containing disfluencies for training is expensive, different data augmentation methods for recovering the zero-shot performance drop are also evaluated (§3.3). Finally, we demonstrate the efficacy of using the human annotated data in varying fractions, for both end-to-end QA supervision and disfluency generation based data augmentation techniques (§4.2).

We argue that creation of datasets, such as DISFL-QA, are vital for **(1)** improving understanding of disfluencies, and **(2)** developing robust NLU models in general.

2 DISFL-QA: Adding Disfluencies to QA

DISFL-QA builds upon the existing SQUAD-v2 dataset, a question answering dataset which contains curated paragraphs from Wikipedia and associated questions. Each question associated with the paragraph is sent for a human annotation task to add a contextual disfluency using the paragraph as a source of distractors. Finally, to ensure the quality of the dataset, a subsequent round of human evaluation with an option to re-annotate is conducted.

2.1 Source of Questions

We sourced passages and questions from SQUAD-v2 (Rajpurkar et al., 2018) development set. SQUAD-v2 is an extension of SQUAD-v1 (Rajpurkar et al., 2016) that contains unanswerable questions written adversarially by crowd workers to look similar to answerable ones from SQUAD-v1. We use both answerable and unanswerable questions for each passage in the annotation task.

2.2 Annotation Task

To ensure high quality of the dataset, our annotation process consists of 2 rounds of annotation:

First Round of Annotation. Expert raters were shown the passage along with all the associated questions and their answers, with one of the question-answer pair highlighted for annotation.[1] The raters were instructed to use the provided context in crafting disfluencies to make for a non-trivial dataset.

The rater had to provide a disfluent version of the question that (a) is *semantically equivalent* to the original question (b) is *natural*, i.e., a human can utter them in a dialogue setting. When

---

[1] The raters were linguistic experts, and were trained for the task with 2 rounds of pilot annotation.

Type	**Passage** (some parts shortened)	**Fluent Question**	**Disfluent Question**
Interrogative Restart (**30%**)	*. . . Roger de Tosny travelled to the Iberian Peninsula to carve out a state for himself. In 1064, during the War of Barbastro, William of Montreuil led the papal army . . .*	Who was in charge of the papal army in the War of Barbastro?	**Where did the no** who was in charge of the papal army in the Barbastro War?
Entity Correction (**25.6%**)	*. . . While many commute to L.A. and Orange Counties, there are some differences in development, as most of San Bernardino and Riverside Counties were developed in the 1980s and 1990s. . .*	Other than the 1980s, in which decade did most of San Bernardino and Riverside Counties develop?	Other than **the 1990s I mean actually the 1980s** which decade did San Bernardino and Riverside counties develop?
Adverb/Adj. Correction (**20%**)	*. . . Southern California is home to Los Angeles International Airport, the second-busiest airport in the United States by passenger volume; San Diego International Airport the busiest single runway airport in the world. . .*	What is the second busiest airport in the United States?	What airport in the United States is the **busiest no second busiest**?
Entity Type Correction (**21.1%**)	*. . . To the east is the Colorado Desert and the Colorado River, and the Mojave Desert at the border with Nevada. To the south is the Mexico-United States border. . .*	What is the name of the water body that is found to the east?	What is the name of the **desert wait the water body** that is found to the east?
Others (**3.3%**)	*. . . Complexity measures are very generally defined by the Blum complexity axioms. Other complexity measures used in complexity theory include communication complexity and decision tree complexity. . .*	What is typically used to broadly define complexity measures?	What is **defined no** is **typically used** to broadly define complexity measures?

Table 1: Example passage and fluent questions from the SQUAD dataset and their disfluent versions provided by human raters, categorized by the type of disfluency along with their estimated percentage in the DISFL-QA dataset.

writing the disfluent version of a question, we instructed raters not to include partial words or filled pauses (e.g., "*um*", "*uh*", "*ah*" etc.), as they can be detected relatively easily (Johnson and Charniak, 2004; Jamshid Lou and Johnson, 2017). Raters were shown example disfluencies from each of the categories in Table 1. On average, raters spent 2.5 minutes per question. Introduction of a disfluency increased the mean length of a question from 10.3 to 14.6 words.

Human Evaluation + Re-annotation. To assess and ensure high quality of the dataset, we asked a another set of human raters the following yes/no questions:

1. Is the disfluent question *consistent* with respect to the fluent question? i.e., the disfluent question is semantically equivalent to the original question in that they share the same answer.

2. Is the disfluent question *natural*? Naturalness is defined in terms of human usage, grammatical errors, meaningful distractors etc.

After the first round of annotation, we found that the second pool of raters found the disfluent questions to be consistent and natural 96.0% and 88.5% of the time, with an inter-annotator agreement of 97.0% and 93.0%[2], respectively. This suggests that the initial round of annotation resulted in a high quality dataset. Furthermore, for the cases identified as either inconsistent or unnatural, we conducted a second round of re-annotation with updated guidelines to make required corrections.

2.3 Categories of Disfluencies

To assess the distribution of different types of disfluencies, we sampled 500 questions from the training and development sets and manually annotated the nature of disfluency introduced by the raters. Table 1 shows the distribution of these categories in the dataset.

A notable difference between DISFL-QA and SWITCHBOARD (Godfrey et al., 1992) is that DISFL-QA contains a larger fraction of corrections and restarts, which have been shown to be the hardest disfluencies to detect and correct (Zayats et al., 2014; Jamshid Lou et al., 2018; Yang et al., 2020). From Table 1, we can see that ≈30% and >65% of the disfluencies in DISFL-QA are restarts and corrections respectively.

In addition to the specific categories men-

---

[2] Cohen's $\kappa = 0.55$, indicating moderate agreement.

Dataset	Switchboard	Disfl-QA
Domain	Telephonic Conversations	Wikipedia Passages
Goal-oriented	No	Yes
Contextual	No	Yes
Size (# sentences)	7.9k	11.8k
Disfluencies	20%	100%
Correction & Restarts	<50%	>90%
Coreferences	<1%	≈10%

Table 2: Comparison of Disfl-QA with Switchboard. Disfl-QA is more diverse, contains harder disfluencies and new phenomenon like coreference.

tioned in Table 1, the dataset includes other challenging phenomena which are shared across these categories. For instance, example below shows disfluencies which introduce *coreferences* between the *reparandum* and the *repair* (mentions marked **[.]**), allowing more complex corrections not present in existing datasets:

Who does BSkyB have an operating license from ? → **Who removed [BSkyB's] operating license no scratch that who do [they] have [their] operating license from** ?

Table 2 summarizes the key differences between Disfl-QA and the Switchboard dataset.

3 Experimental Setup

3.1 Models to Compare

We use two different modeling approaches to answer disfluent questions in Disfl-QA.

LMs for QA. We use BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020) as our QA models in the standard setup which has shown to achieve state-of-the-art performance for SQuAD. We fine-tune BERT for a span selection task, whereby predicting `start` and `end` probabilities for all the tokens in the context.

T5 is finetuned under the standard *text2text* formulation, when given (question, passage) as input the model generates the answer as the output. For predicting `<no answer>`, the model was trained to generate *"unknown"*.

LMs for Disfluency Correction. We also fine-tune the above LMs as disfluency correction models. Given the disfluent question as input, a correction model predicts the fluent question, which is then fed into a QA model. For BERT, we use the state-of-the-art BERT-based disfluency correction model by Jamshid Lou and Johnson (2020) trained on Switchboard. We also train T5 models on Disfl-QA to prevent the distribution skew between Switchboard and Disfl-QA, and account for new phenomena like coreferences.

Rule	Fluent	Disfluent
Q	What was the Norman religion?	What was **replaced with** **no no** what was the Norman religion?
V	When was the Duchy of Normandy founded?	When was the Duchy of Normandy **offered** **ugh I mean** **founded**?
Adj	What is the original meaning of the word Norman?	What is the **English** **rather** **original** meaning of the word Norman?
Adv	Who did Beyoncé perform privately for in 2011?	Who did Beyoncé perform **publicly** **oops** **privately** for in 2011?
Ent	Who was a prominent Huguenot in Holland?	Who was a prominent **Saint Nicholas** **no I mean** Huguenot in Holland?

Table 3: Example of synthetically generated disfluent questions using the contextual heuristics.

3.2 Training Settings

We train the BERT and T5 variants on the following two data configurations:

ALL where the model is trained on all of SQuAD-v2, including the non-answerable questions. Evaluation is done against the entire test set.

ANS where the model is trained only on answerable questions from SQuAD-v1, without the capabilities of handling non-answerable questions.

3.3 Datasets

Human Annotated Datasets. We use 3 datasets in our experiments: SQuAD-v1, SQuAD-v2, and Disfl-QA. We split the 11,825 annotated questions in Disfl-QA into train/dev/test set containing 7182/1000/3643 questions, respectively. The split was also done at an article level such that the questions belonging to the same passage belong in the same split. For zero-shot experiments, we only use the train of SQuAD.

Evaluation is done on the subset of SQuAD-v2 development set that corresponds to the Disfl-QA test to ensure fair comparison.

Heuristically Generated Data. We also generate disfluencies heuristically to validate the importance of human annotated disfluencies. Inspired by the disfluency categories seen in our annotation task, we derive the following heuristics to

Model	Train	Eval	HasAns-F1	NoAns-F1	Overall-F1
BERT-QA	**ALL**	SQuAD	83.87	70.55	77.46
		Heuristics	51.45 ↓ 32.42	74.49 ↑ 3.94	62.53 ↓ 14.93
		Disfl-QA	40.97 ↓ 42.90	75.97 ↑ 5.42	57.81 ↓ 19.65
	ANS	SQuAD	89.63	-	89.63
		Heuristics	80.52 ↓ 9.11	-	80.52 ↓ 9.11
		Disfl-QA	78.88 ↓ 10.75	-	78.88 ↓ 10.75
T5-QA	**ALL**	SQuAD	91.38	87.67	89.59
		Heuristics	39.98 ↓ 51.40	92.57 ↑ 4.90	65.27 ↓ 24.32
		Disfl-QA	35.31 ↓ 56.07	90.06 ↑ 2.39	61.64 ↓ 27.95
	ANS	SQuAD	93.71	-	93.71
		Heuristics	81.73 ↓ 12.01	-	81.73 ↓ 12.01
		Disfl-QA	80.39 ↓ 13.32	-	80.39 ↓ 13.32
Disfluency Correction + T5-QA	**ALL**	SQuAD	91.38	87.67	89.59
		Heuristics	42.83 ↓ 48.55	92.18 ↑ 4.51	66.56 ↓ 23.03
		Disfl-QA	43.61 ↓ 47.77	89.55 ↑ 1.88	65.71 ↓ 23.88
	ANS	SQuAD	93.71	-	93.71
		Heuristics	82.27 ↓ 10.44	-	82.27 ↓ 10.44
		Disfl-QA	82.64 ↓ 11.07	-	82.64 ↓ 11.07

Table 4: Breakdown of zero-shot performance of fine-tuned BERT and T5 QA models, trained only on the SQuAD dataset, and evaluated on SQuAD, Heuristics (§3.3), and Disfl-QA test sets. We also evaluate the performance by using state-of-the-art disfluency detection model by Jamshid Lou and Johnson (2020) in a pipelined fashion.

augment our data with *silver*[3] standard disfluencies: (i) Switch-Q which inserts prefix of another question as a prefix to the original question, and (ii) Switch-X, where X could be verb, adjective, adverb, or entity, and is inserted as a reparandum in the question.

To facilitate contextual disfluencies, we use the reparandums from the context. For Switch-Verb/Adj/Adv/Ent, this was done by picking tokens and phrases from the context passage. For Shift-Q, we used other questions associated with the same passage. We used spaCy[4] NER and POS tagger to extract relevant entities and POS tags, and sample interregnum from a list of fillers. Table 3 shows an example from each of the heuristics. We then finally combine all the heuristics (All in Table 3) by uniformly sampling a single disfluent question from the set of possible transformations of the question.

3.4 Evaluation Method

In all our experiments, we evaluate QA performance using the standard SQuAD-v2 evaluation script which reports EM and F1 scores over the `HasAns` (asnwerable) and `NoAns` (non-answerable) slices along with the overall scores. For brevity, we report only the F1 numbers as we observed similar trends in EM and F1 across our experiments.

4 Experiments

We conduct experiments with Disfl-QA to answer the following questions: **(a)** Are state-of-the-art LM based QA models robust to introduction of disfluencies in the questions under a zero-shot setting ? **(b)** Can we use heuristically generated synthetic disfluencies to aid the training of QA models to handle disfluencies ? **(c)** Given a small amount of labeled data, can we recover performance by fine-tuning the QA models or training a disfluency correction model to pre-process the disfluent questions into fluent ones before inputting to the QA models ? **(d)** In the above setting, can we train a generative model to generate more disfluent training data ?

4.1 Zero-Shot Performance

Table 4 shows the performance of different variants measuring their zero-shot capabilities.

Performance of BERT-QA and T5-QA. We see from Table 4 that when tested directly on on heuristics and Disfl-QA test sets, both the BERT-QA and T5-QA models exhibit significant performance drop, as compared to the performance on the fluent benchmark of SQuAD. The performance drop for the complete models is greater

---

[3] The *silver* nature of the data is due to the fact that we can not enforce naturalness or semantic equivalence of §2.

[4] `https://spacy.io/`

Original	`HasAns`		`NoAns`
Prediction	`NoAns`	`WrongAns`	`HasAns`
SQuAD	71	150	216
Disfl-QA	1091	168	174

Table 5: Breakdown of prediction errors for the T5-QA-ALL model on the fluent and disfluent questions. `WrongAns` represents that the model predicted an incorrect span from context.

when compared to their answerable-only counterparts. The best performing T5-ALL model shows a **drop of 27.95 F1** points for the complete setup and **13.32 F1** point for the answerable only T5-ANS model. This shows BERT and T5 are not robust when questions contain disfluencies.

Disfluency Correction + T5-QA. We use the BERT based state-of-the-art disfluency correction (Jamshid Lou and Johnson, 2020) as a pre-processing step before feeding the input to our T5-QA model. The models trained on Switchboard are not able to fill a significant performance gap, with the complete and answerable models recovering 4.07 and 2.25 F1 points, respectively. We will revisit this setting in the few-shot experiments.

Disfl-QA test-set vs. Heuristics test-set. Next, we compare the performance of heuristically generated disfluent questions against the human annotated questions. In general, human annotated disfluent questions exhibit larger performance drop compared to heuristics, across different models.

Taking a closer look at the T5-ALL model shows that Disfl-QA shows a bigger drop in `HasAns` cases and smaller increase in `NoAns` cases, as compared to the heuristics test set. For the T5-ANS model, Disfl-QA shows a larger drop in performance which is attributed to the model picking wrong answer span. Based on this, we hypothesize that between the two datasets, heuristics are able to confuse the models in over-predicting `<no answer>`, but Disfl-QA is superior when it comes to confuse the models to picking a different answer span altogether (as seen in Table 4 for models in ANS setting). This demonstrates that collecting a dataset like Disfl-QA via human annotation holds value for contextual disfluencies.

	HasAns F1	**NoAns F1**	**Overall F1**
Fluent (⋆)	91.38	87.67	89.59
Zero-Shot	35.21	90.06	61.64
+ Sw-Adj	68.49	86.24	77.03
+ Sw-Adv	67.37	85.27	75.98
+ Sw-ENT	**74.76**	**85.95**	**80.14**
+ Sw-Q	70.03	78.94	74.31
+ Sw-Verb	68.01	87.16	77.22
+ All	**78.86**	**85.96**	**82.27**

Table 6: Performance on Disfl-QA with individual (Sw-XX) and combined (All) heuristics based data augmentation and fine-tuning.

Performance Gap Breakdown. For models trained on ALL setting, we find that the performance drop is largely due to the drop in F1 (over **50** points) on `HasAns` questions as opposed to `NoAns` questions, where it is almost negligible or even positive in some cases. Upon closer analysis (Table 5) we find that a major fraction of prediction errors for `HasAns` is attributed to `HasAns` → `NoAns` errors, instead of `HasAns` → `WrongAns`.[5]

We believe that the disfluencies are causing the answerable questions to resemble the non-answerable ones as seen by both BERT and T5 models under ALL setting. This results in an overly conservative model in terms of answerability and instead resorts to over-predicting `<no answer>`, causing gain in non-answerable recall at the cost of precision. In contrast, for a comparable ANS model the drop in F1 is smaller, primarily due to relatively easier decision making, i.e. not required to decide when to answer vs. not.

Fine-tuning on Heuristic Data. In this experiment, we fine-tune on heuristically generated data from §3.3 and directly test on Disfl-QA. Table 6 compares the performance of the heuristics fine-tuned model on the Disfl-QA test-set. The overall heuristics trained model (All) is able to cover a significant performance drop from 61.64 to 82.27, an increase of 20.63 F1 points. However, this still is 7.32 F1 points short of the fluent performance.

Amongst the individual heuristics, we observe the following order of effectiveness w.r.t. performance on the `HasAns` cases: ENT > SQ > ADJ > VERB > ADV. One possible expla-

---

[5] We use the standard SQuAD evaluation script and mark a prediction as `WrongAns` iff F1(pred,gold)< 0.8.

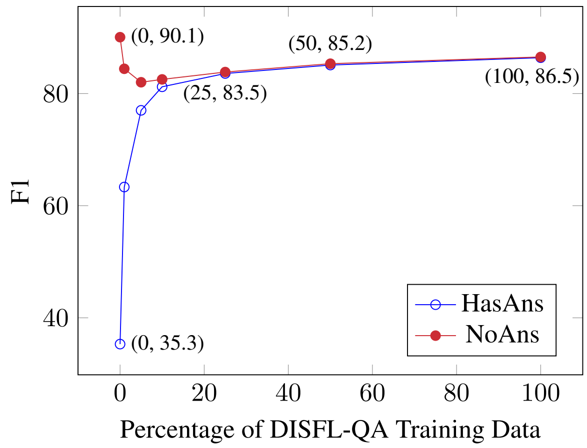

Figure 2: Few shot performance for different fraction of training data. We can see that performance on `HasAns` cases increases monotonically with increase in gold data. However, for the `NoAns` cases, the performance first takes a drop (compared to zero-shot) and then increases.

	HasAns F1	**NoAns F1**	**Overall F1**
Fluent (⋆)	91.38	87.67	89.59
Zero-Shot	35.21	90.06	61.64
Heuristics	78.86	85.96	82.27
Direct Supervision			
25% Data	83.58	83.84	83.71
+ Q → DQ	86.44	84.53	85.52
+ CQ → DQ	87.47	83.11	85.37
50% Data	85.09	85.33	85.20
100% Data	86.40	**86.53**	**86.46**
+ Q → DQ	86.95	85.73	86.33
+ CQ → DQ	**87.29**	85.22	86.29
Pipelined			
DQ → Q	87.65	**86.70**	**87.19**
CDQ → Q	**87.99**	86.02	87.04

Table 7: Performance on the test set of DISFL-QA when using gold human annotated data in training different components.

nation for SWITCH-ENT and SWITCH-Q being more effective is the fact that our original annotated dataset has a relatively high percentage of entity and interrogative correction.

4.2 Few Shot Performance

Next, we evaluate the performance of the models when we use a part of human annotated gold disfluent data for training: (i) direct end-to-end supervision, (ii) generation based data augmentation, and (iii) training disfluency correction models.

Direct Supervision (*k*-shot). In this setting, we pick a SQUAD-v2 T5 model and then perform a second round of fine-tuning with varying percentages of DISFL-QA gold training data. We experiment with $1, 5, 10, 25, 50$, and 100 percent of the total gold data.

Figure 2 shows the performance for the `HasAns` and `NoAns` cases as we increase the amount of training data. The `HasAns` performance increases gradually from 35.31 F1 points, in the zero-shot setting, to 86.40 F1 points with complete training data. Interestingly, for the `NoAns` cases, the performance first drops from 90.06 F1 points, in the zero-shot setting, to 82.02 F1 with 5% data and then monotonically increasing to 86.53 F1 with complete data. This can be attributed to the fact that the zero-shot models were under-predictive (high recall, low precision for `<no asnwer>`) due to lack of robustness to disfluent inputs.

Furthermore, Table 7 compares the performance of using the gold training data of DISFL-QA against the heuristics data. It shows that the models trained with disfluent data from DISFL-QA are able to cover a major gap in answerable slice, which wasn't possible with the heuristically generated data. Direct supervision bring an additional performance improvement of 4.19 F1 points over the heuristics.

Generation Based Data Augmentation. We use the T5 model for synthetically generating disfluent question from fluent question in the *text2text* framework. We use the training set of DISFL-QA to train the following generative models: (i) context-free generation (Q → DQ), and (ii) context-dependent generation (CQ → DQ) which use passage as well for generation.

Table 8 shows example generation from the two models. We observe that CQ → DQ is able to learn meaningful contextual disfluency generation, whereas Q → DQ can lead to non-meaningful or inconsistent disfluencies due to lack to context.

We then pick 5k random (question, answer) pairs from SQUAD training data and apply our generative model to produce disfluent training data for the QA models. Table 7 shows the performance of using data augmentation. We perform data augmentation under two different train data settings: (1) 25% data, and (2) 100% data. Interestingly, for the models trained on 25% train data + generated data, we observe a gain of 1.81 F1

Passage: ...*Whereas a **genome sequence** lists the order of every DNA base in a genome, a **genome map** identifies the landmarks. A genome map is less detailed than a genome sequence and aids in navigating around the genome* ...

Fluent Question : What does a genome map list the order of ?
T5 Q → DQ : What is no what does a genome map list the order of ?
T5 CQ → DQ : What does a genome sequence list the order of no sorry what does a genome map list the order of?

Passage: ...*The presence of **fat** in the small intestine produces hormones that stimulate the release of pancreatic lipase from the pancreas and **bile** from the liver which helps in* ...

Fluent Question : What is one molecule of fat ?
T5 Q → DQ : What is one molecule of protein no fat ?
T5 CQ → DQ : What is one molecule of bile no wait fat ?

Passage: ...*In 1964, **Nikita Khrushchev** was removed from his position of power and replaced with **Leonid Brezhnev**. Under his rule, the Russian SFSR* ...

Fluent Question : When did Leonid Brezhnev die ?
T5 Q → DQ : When was the age of Leonid Brezhnev ?
T5 CQ → DQ : When did Nikita Khrushchev *er* I mean Leonid Brezhnev die ?

Table 8: Example disfluent question (DQ) as generated by the Q → DQ and CQ → DQ T5 generative models for data augmentation. We observe that CQ → DQ generates meaningful disfluencies compared to context-free generation, the latter leading to irrelevant or inconsistent questions in some cases.

points (83.71 → 85.52) in the overall performance which is close to the absolute performance of using 50% gold data. However, for the setup with 100% gold data + generated data, we did not observe a similar improvement in the overall performance.

Pipelined: Disfluency Correction + QA. Unfortunately, existing disfluency correction models and datasets assume that fluent text is a subsequence of the disfluent one, and hence these approaches cannot solve disfluencies in Disfl-QA involving coreference. For fair comparison, we train a T5 generation model as a Disfl-QA specific disfluency correction model using the training set of Disfl-QA, with a simple DQ → Q and CDQ → Q T5 task formulation.

With this pipelined approach, we get further improvements with an overall F1 of 87.19 (Table 7), however, still lacking by ≈2.4 F1 points compared to the fluent dataset. This shows that such complex cases require better modeling, preferably in an end-to-end setup.

5 Related Work

5.1 Disfluency Correction

The most popular approach in literature poses disfluency correction as a sequence tagging task, in which the fluent version of the utterance is obtained by identifying and removing the disfluent segments (Zayats et al., 2014; Ferguson et al., 2015; Zayats et al., 2016; Lou and Johnson, 2017; Jamshid Lou and Johnson, 2020; Wang et al., 2020). . Traditional disfluency correction models use syntactic features (Honnibal and Johnson, 2014), language models (Johnson et al., 2004; Zwarts and Johnson, 2011), discourse markers (Crible, 2017), or prosody-based features for learning (Zayats and Ostendorf, 2019; Wang et al., 2017) while recent disfluency correction models largely utilize pre-trained neural representations (Lou et al., 2018). Most of these models depend on human-annotated data. As a result, recently, data augmentation techniques have been proposed (Yang et al., 2020; McDougall and Duckworth, 2017) to alleviate the strong dependence on labeled data. However, the resulting augmented data either via heuristics (Wang et al., 2020) or generation models (Yang et al., 2020) is often limited in terms of disfluencies types and may not well capture natural disfluencies in daily conversations.

5.2 Question Answering Under Noise

In the QA literature, our work is related to two threads that aim to improve robustness of QA models: (i) QA under adversarial noise, and (ii) noise arising from speech phenomena.

Prior work on adversarial QA have predominantly generated adversaries automatically (Zhao et al., 2018), which are verified by humans to ensure semantic equivalence (i.e. answer remains same after perturbation). For instance, Ribeiro et al. (2018) generated adversaries using paraphrasing, while Mudrakarta et al. (2018) perturbed questions based on attribution. Closest work to ours is Jia and Liang (2017), who modified SQuAD to contain automatically generated adversarial sentence insertions.

Our work is more closely related to prior work on making NLP models robust to noise arising from speech phenomena. Earlier work (Surdeanu et al., 2006; Leuski et al., 2006) have built QA models which are robust to disfluency-like phenomenon, but they were limited in the corpus complexity, domain, and scale. Recently there has been renewed interest in constructing audio enriched versions of existing NLP datasets, for example, the SPOKEN-SQUAD (Li et al., 2018) and SPOKEN-COQA (You et al., 2020) with the aim to show the effect of speech recognition errors on QA task. However, since collecting audio is challenging, another line of work involves testing the robustness of NLP models to ASR errors in transcribed texts containing synthetic noise using TTS → ASR technique (Peskov et al., 2019; Peng et al., 2020; Liu et al., 2020; Ravichander et al., 2021). Our work suggests a complementary approach to data collection to surface a specific speech phenomenon that affects NLP.

6 Conclusion

This work presented DISFL-QA, a new challenge set containing contextual semantic disfluencies in a QA setting. DISFL-QA contains diverse set of disfluencies rooted in context, particularly a large fraction of corrections and restarts, unlike prior datasets. DISFL-QA allows one to directly quantify the effect of presence of disfluencies in a downstream task, namely QA. We analyze the performance of models under varying when subjected to disfluencies under varying degree of gold supervision: zero-shot, heuristics, and *k*-shot.

Large-scale LMs are not robust to disfluencies. Our experiments showed that the state-of-the-art pre-trained models (BERT and T5) are not robust when directly tested on disfluent input from DISFL-QA. Although a naturally occurring phenomenon, the noise introduced by the disfluent transformation led to a non-answerable behavior at large.

Contextual heuristics partially recover performance. We derived heuristics, in attempt to resemble the contextual nature of DISFL-QA, by introducing semantic distractors based on NER, POS, and other questions. In our experiments, we found that heuristics are effective in: (1) confusing the models in zero-shot setup, and (2) partially recovering the performance drop on DISFL-QA with fine-tuning. This indicates that the heuristics might be capturing some key aspects of DISFL-QA.

Efficacy of gold training data. We use the gold data for supervising various models: (i) end-to-end QA model, (ii) disfluency correction, and (iii) disfluency generation (for data augmentation). For all the experiments, gold supervision outperforms heurisitics' supervision significantly. Furthermore, we observed that in a low resource setup generation based data augmentation can match the performance of a high resource modeling setup.

7 Discussion

While DISFL-QA aims to fill a major gap between speech and NLP research community, understanding *disfluencies* holistically requires the following:

General disfluencies focused NLP research. We believe understanding of disfluencies is a key ingredient for enabling natural human-machine communication in the near future, and call upon the NLP community to devise generalized few-shot or zero-shot approaches to effectively handle disfluencies present in input to NLP models, without requiring task specific disfluency datasets.

Constructing datasets for spoken problems. We would also like to bring attention to the fact that being a speech phenomenon, a spoken setup would have been an ideal choice for disfluencies dataset. This would have accounted for higher degree of confusion, hesitations, corrections, etc. while recalling parts of context on the fly, which otherwise one may find hard to create synthetically when given enough time to think.

However, such a spoken setup is extremely tedious for data collection mainly due to: (i) privacy concerns with acquiring speech data from real world speech transcriptions, (ii) creating scenarios for simulated environment is a challenging task, and (iii) relatively low yield for cases containing disfluencies. In such cases, we believe that a targeted and purely textual mode of data collection can be more effective both in terms of cost and specificity.

References

Eugene Charniak and Mark Johnson. 2001. Edit Detection and Parsing for Transcribed Speech. In *Proc. of NAACL*.

Ludivine Crible. 2017. Discourse Markers and (Dis)fluency in English and French: Variation and Combination in the DisFrEn Corpus. *International Journal of Corpus Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL*.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency Detection with a Semi-Markov Model and Prosodic Features. In *Proc. of NAACL*.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone Speech Corpus for Research and Development. In *Proc. of ICASSP*.

Matthew Honnibal and Mark Johnson. 2014. Joint Incremental Disfluency Detection and Dependency Parsing. *Transactions of the Association for Computational Linguistics*.

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency Detection using Auto-Correlational Neural Networks. In *Proc. of EMNLP*.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency Detection using a Noisy Channel Model and a Deep Neural Language Model. In *Proc. of ACL*.

Paria Jamshid Lou and Mark Johnson. 2020. Improving Disfluency Detection by Self-Training a Self-Attentive Model. In *Proc. of ACL*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proc. of EMNLP*.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *Proc. of ACL*.

Mark Johnson, Eugene Charniak, and Matthew Lease. 2004. An Improved Model for Recognizing Disfluencies in Conversational Speech. In *Proc. of Rich Transcription Workshop*.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2006. Building Effective Question Answering Characters. In *In Proc. of SIGdial Workshop on Discourse and Dialogue*.

Chia-Hsuan Li, Szu-Lin Wu, Chi-Liang Liu, and Hung yi Lee. 2018. Spoken SQuAD: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension. In *Proc. of Interspeech*.

Jiexi Liu, Ryuichi Takanobu, Jiaxin Wen, Dazhen Wan, Hongguang Li, Weiran Nie, Cheng Li, Wei Peng, and Minlie Huang. 2020. Robustness Testing of Language Understanding in Task-Oriented Dialog. *arXiv preprint arXiv:2012.15262*.

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency Detection using Auto-Correlational Neural Networks. In *Proc. of EMNLP*.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency Detection using a Noisy Channel Model and a Deep Neural Language Model. In *Proc. of ACL*.

Kirsty McDougall and Martin Duckworth. 2017. Profiling fluency: An Analysis of Individual Variation in Disfluencies in Adult Males. *Speech Communication*.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the Model Understand the Question? In *Proc. of ACL*.

Baolin Peng, Chunyuan Li, Zhu Zhang, Chenguang Zhu, Jinchao Li, and Jianfeng Gao. 2020. RADDLE: An Evaluation Benchmark and Analysis Platform for Robust Task-oriented Dialog Systems. *arXiv preprint arXiv:2012.14666*.

Denis Peskov, Joe Barrow, Pedro Rodriguez, Graham Neubig, and Jordan Boyd-Graber. 2019. Mitigating Noisy Inputs for Question Answering. In *Proc. of Interspeech*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *In Proc. of JMLR*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proc. of ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proc. of EMNLP*.

Abhilasha Ravichander, Siddharth Dalmia, Maria Ryskina, Florian Metze, Eduard Hovy, and Alan W Black. 2021. NoiseQA: Challenge Set Evaluation for User-Centric Question Answering. In *In Proc. of EACL*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically Equivalent Adversarial Rules for Debugging NLP models. In *Proc. of ACL*.

Elizabeth Shriberg. 1996. Disfluencies in Switchboard. In *Proc. of ICSLP*.

Elizabeth Ellen Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis.

Mihai Surdeanu, David Dominguez-Sal, and Pere R Comas. 2006. Design and Performance Analysis of a Factoid Question Answering System for Spontaneous Speech Transcriptions. In *Proc. of ICSLP*.

Shaolei Wang, Wangxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. 2020. Multi-Task Self-Supervised Learning for Disfluency Detection. In *Proc. of AAAI*.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-Based Disfluency Detection using LSTMs. In *Proc. of EMNLP*.

Jingfeng Yang, Diyi Yang, and Zhaoran Ma. 2020. Planning and Generating Natural and Diverse Disfluent Texts as Augmentation for Disfluency Detection. In *Proc. of EMNLP*.

Chenyu You, Nuo Chen, Fenglin Liu, Dongchao Yang, and Yuexian Zou. 2020. Towards Data Distillation for End-to-end Spoken Conversational Question Answering. *arXiv preprint arXiv:2010.08923*.

Vicky Zayats and Mari Ostendorf. 2019. Giving Attention to the Unexpected: Using Prosody Innovations in Disfluency Detection". In *Proc. of NAACL*.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2014. Multi-domain disfluency and repair detection. In *Proc. of Interspeech*.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency Detection Using a Bidirectional LSTM. In *Proc. of Interspeech*.

Victoria Zayats, Trang Tran, Richard A. Wright, Courtney Mansfield, and Mari Ostendorf. 2019. Disfluencies and Human Speech Transcription Errors. In *Proc. of Interspeech*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating Natural Adversarial Examples. In *Proc. of ICLR*.

Simon Zwarts and Mark Johnson. 2011. The Impact of Language Models and Loss Functions on Repair Disfluency Detection. In *Proc. of ACL*.

Does Robustness Improve Fairness? Approaching Fairness with Word Substitution Robustness Methods for Text Classification

Yada Pruksachatkun *
Amazon Alexa
ypruksac@amazon.com

Satyapriya Krishna *
Amazon Alexa
satyapk@amazon.com

Jwala Dhamala
Amazon Alexa
jddhamal@amazon.com

Rahul Gupta
Amazon Alexa
gupra@amazon.com

Kai Wei Chang
UCLA, Amazon Alexa
kwchang@cs.ucla.edu

Abstract

Existing bias mitigation methods to reduce disparities in model outcomes across cohorts have focused on data augmentation, debiasing model embeddings, or adding fairness-based optimization objectives during training. Separately, certified word substitution robustness methods have been developed to decrease the impact of spurious features and synonym substitutions on model predictions. While their end goals are different, they both aim to encourage models to make the same prediction for certain changes in the input. In this paper, we investigate the utility of certified word substitution robustness methods to improve equality of odds and equality of opportunity on multiple text classification tasks. We observe that certified robustness methods improve fairness, and using both robustness and bias mitigation methods in training results in an improvement in both fronts.

1 Introduction

As natural language processing (NLP) technologies are increasingly used in essential real-world applications, such as social media, healthcare, personal assistants and law (He et al., 2020; Ahmad et al., 2020), it is important to ensure these systems do not create unintended outcomes for end-users or offer disparate experiences to customers from diverse backgrounds. This includes ensuring that model performance does not significantly differ across people belonging to different cohorts, such as different gender or race groups.

A major subset of industry NLP applications lies in text classification, such as domain and intent classification in voice assistants (Su et al., 2018) or code tagging in healthcare (Kemp et al., 2019). In this study, we focus on toxicity classification (Dixon et al., 2018) and occupation classification of Wikipedia biographies (De-Arteaga et al., 2019). For toxicity classification, ensuring fairness means ensuring that a model can identify toxicity to a similar accuracy across all examples regardless of the protected groups present in the example. Past studies (e.g, (Dixon et al., 2018; Zhang et al., 2020; Zhao and Chang, 2020)) have shown that toxicity classification models will falsely classify text containing certain protected attributes as toxic. Leading social media platforms and internet companies use toxicity classification models for content moderation (Gorwa et al., 2020), thus having bias in such models can lead to increased silencing of under-served groups. Similarly, for occupation classification, a fair model should correctly identify occupations given a biography, regardless of the protected group that a person belongs to (De-Arteaga et al., 2019).

Recently, several studies have demonstrated societal bias in NLP systems (Hutchinson et al., 2020; Tan and Celis, 2019; Liang et al., 2020) and various approaches have been proposed to mitigate the bias. These approaches include creating balanced datasets (Park et al., 2018; Zhao et al., 2018a), developing methods optimized for particular fairness notions (Zhang et al., 2017, 2020), model calibration (Zhao et al., 2017; Jia et al., 2020), and reducing representational bias (Bolukbasi et al., 2016b; Zhao et al., 2018b; Liang et al., 2020).

Separately, certified robustness approaches (Jia et al., 2019; Ye et al., 2020) have been developed to ensure robustness against word substitution attacks. Specifically, these strategies ensure small perturbations in the input embedding space do not alter model predictions. Despite never having been discussed in prior literature, this corresponds to notions of fairness, since protected attribute information (e.g. gender) is often irrelevant to the task

---

* Equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3320–3331
August 1–6, 2021.

at hand (i.e. "She is a good singer" and "He is a good singer" should have the same sentiment label). Thus, we posit that word substitution robustness methods can be used to make models invariant to protected attribute tokens and identifiers.

We explore the effect of robustness methods on fairness with GloVe-based CNN models (Kim, 2014) trained with Interval Bound Propagation (IBP) (Jia et al., 2019), and BERT (Devlin et al., 2019a) trained with SAFER (Ye et al., 2020). We compare the effect of these robustness methods to popular bias mitigation methods. We find that robustness methods achieve promising performance on fairness metrics exceeding that of bias mitigation methods in several text classification tasks on gender and sexual orientation dimensions. Furthermore, training on both fairness and robustness exceeds performance over robustness and bias mitigation methods alone. Comprehensive analysis and visualization demonstrate that the robust methods decrease feature importance on gender tokens.

Our contributions are two-fold. First, we show that certified robustness methods can be used and integrated with bias mitigation methods to effectively improve models' performance on several notions of fairness, notably equality of opportunity and equality of odds. Secondly, by integrating robustness methods with fairness, we can improve a model's robustness while reducing bias, which is important in creating trustworthy NLP systems. Our study's practical implications include applications to models used in the industry that can handle customer inputs that may differ from the training data (robust) and that minimize any unintended consequences on the customers (fair). With this study, we aim to motivate future work geared towards developing methods that jointly optimize for multiple trustworthy aspects of models; specifically, those addressing model robustness and fairness.

2 Mitigating Bias through Certified Robustness Methods

In the following, we first define the notions of fairness considered in this paper. Then, we discuss certified robustness methods, and how they can be applied to reduce bias in models.

2.1 Fairness Notions

We focus on measuring two notions of fairness in this paper – equalized odds and equality of opportunity, as they are commonly used in quantifying bias in NLP applications. We describe the metrics associated with these notions in Section 3. We give application examples of these notions on toxicity and occupation classification for English texts.

Equalized Odds A model achieves *Equalized Odds* (Hardt et al., 2016) with respect to a protected attribute A and outcome Y if $P(\bar{Y} = 1|A = 0, Y = y) = P(\bar{Y} = 1|A = 1, Y = y)$, for $y \in \{0, 1\}$. Protected attributes are traits or characteristics that cannot be discriminated against by law[1]. Intuitively, this means that the model should have equal true positive and false positive rates across groups. For toxicity classification, equalized odds implies that a model should be able to effectively detect toxicity on comments that include identifiers across all protected attribute cohorts, while not silencing any one cohort. Prior studies demonstrate that models disproportionately predict sentences associated with LGBTQ individuals as toxic, which may further silence discussion around LGBTQ issues and the voices of LGBTQ people (Oliva et al., 2020).

Equality of Opportunity A model achieves *Equality of Opportunity* (Hardt et al., 2016) with respect to a protected attribute A and outcome Y if $P(\bar{Y} = 1|A = 0, Y = 1) = P(\bar{Y} = 1|A = 1, Y = 1)$. This is a relaxation of Equalized Odds to the positive outcome, in which the model must have equal true positive rates across groups. For occupation classification, equality of opportunity implies that a model is able to correctly classify biographies of people from all groups, thus enabling equity in positive outcomes such as appropriate and useful matches in job recommendation sites.

Due to bias in the training data, off-the-shelf models often contain biases and disparities in model performance against underrepresented groups. Various bias mitigation approaches have been proposed to ensure the fairness in model predictions. We include a diverse array of bias mitigation methods, spanning embedding debiasing, in-training, and post-processing, as baselines. These consist of instance weighting (Zhang et al., 2020), *HardDebias* word embeddings (Bolukbasi et al., 2016a), and adversarial debiasing (Zhang et al., 2018). See more discussion in Sec. 3.

---

[1] `https://www.eeoc.gov/employers/small-business/3-who-protected-employment-discrimination`

2.2 Certified Robustness for Bias Mitigation

Designed for a different purpose, certified robustness methods present ways to train models that satisfy guarantees of word substitution robustness. By adapting certified robustness methods to fairness applications, we aim to make models invariant to spurious protected attribute information present in inputs, and thus improve in equality of opportunity and equalized odds.

Formally, a model f is *certifiably robust* if, for any example sentence x, and sentences x' that consist of x modified with word substitutions, $f(x) = f(x') = y$. In the robustness context, word substitution consists of swapping a word with its synonyms (usually defined using retrofitted word embeddings). For example, if x = "The waiter talked to the customer about their problems," x' may consist of the sentences "The waitress talked to the customer about their qualms." In the context of fairness, we consider 'waiter' and 'waitress' or gender pronouns to carry the same meaning in the context of toxicity and occupation classification, and to have the same label.

In this paper, we use two recently developed certified robustness methods, Interval Bound Propagation (IBP) (Jia et al., 2019) and SAFER (Ye et al., 2020). Given a set of perturbations for each word, these two models ensures that word substitution do not affect model predictions. In particular, for each word, and a polytope spanned by the potential substitutions for that word in the embedding space, these methods ensure that swapping the word with any point in the polytope will not change the model predictions. To accomplish this, IBP minimizes the upper bound of the set of losses over perturbation sets, and SAFER uses a model-agnostic randomized smoothing technique.

Both IBP and SAFER encourage models to be robust to spurious word substitutions, which include tokens that contain protected attribute information. The perturbations included in the original paper from Alzantot et al. (2018) are based on a GloVe embedding that has been modified such that synonyms are close together. While the perturbation set does not include explicit gender and sexual orientation swaps ('boy' is not included in the perturbation set for 'girl', while 'girls', and 'women' are), we posit that certified robustness methods can still be applied to bias mitigation by improving robustness in examples that contain identifiers of underrepresented groups. Doing so will decrease model performance disparity in underrepresented group cohorts, and thus fulfill fairness notions.

IBP (Jia et al., 2019) IBP computes bounds on the model loss based on bounds on the input. The robustness goal of the IBP method is to minimize $\max F(x, \theta)$. Here, $F(x, \theta)$ denotes the set of losses of a model over $B_{perturb}$, where $B_{perturb}$ is the set of perturbations for an example sentence x. Formally, $F(x, \theta) = (f(\bar{x}, \theta) | \bar{x} \in B_{perturb})$. The full loss for IBP is $(1 - \lambda) f(x, \theta) + \lambda \mu^{final}(x, \theta)$, where μ^{final} is the upper bound on the loss $f(x, \theta)$ and $\lambda \in [0, 1]$.

SAFER (Ye et al., 2020) Unlike IBP, SAFER does not require any changes to the model training. Instead, it employs a randomized smoothing mechanism in which an input is perturbed before being fed to the model during the training time. Specifically, SAFER creates random word substitutions using a perturbation set derived from a synonym network. Ye et al. (2020) determine certified robustness of a model on an example by certifying that, given an example z, model score $s(z)$, and $y_B = \text{argmax}_{c \in Y, c \neq y}\, s(z)$, the model score of the gold label y is higher than the model score of the highest scoring non-gold label y_B by a constant.

3 Empirical Study on the Connection between Fairness and Robustness

To better understand the connection between fairness and certified robustness in the context of text classification, we empirically analyze models augmented with various combinations of robustness and fairness methods, as enumerated below.

1. **Classifier (Baseline)**: The base text classification models. We consider two types of classification models that widely used in the literature, CNN (Kim, 2014) and BERT (Devlin et al., 2019b).
2. **Classifier + Fairness**: Text classifiers trained with bias mitigation techniques (see Sec. 2).
3. **Classifier + Robustness**: Text classifiers trained with robustness methods (see Sec. 2).
4. **Classifier + Robustness + Gender Word Perturbations**: To ensure that the model becomes robust against gender substitutions, we add definitional gender pairs (e.g., swapping he with she) (Bolukbasi et al., 2016b) in the permutation set of IBP and SAFER.
5. **Classifier + Robustness + Fairness**: Text classifier trained with both fairness and ro-

bustness objectives.

We aim to answer the following research questions based on the aforementioned configurations. (1) *What is the effect of robustness methods on mitigating bias* (compare configuration 3 with 1 and 2)? (2) *What is the effect of adding gender word substitutions to the robustness perturbation sets* (compare configurations 3 and 4)? (3) *What is the effect of integrating bias mitigation and robustness methods* (compare configuration 5 with 1 and 3)?.

In particular, to answer the last question, we consider combining popular bias mitigation approaches with IBP as follows.

- **Debiased Word Embeddings + IBP**: We replace the GloVe embeddings in the baseline CNN model with the *HardDebias* embeddings obtained from (Bolukbasi et al., 2016b), while keeping the rest of the IBP training methodology the same.
- **Instance weighting + IBP**: We add the instance weights to each sample in the loss computation during IBP training.
- **Adversarial Training + IBP**: We perform multitask training, alternating between optimizing for robustness loss and adversarial debiasing loss. We initialized our adversarial training with the IBP-trained model.

Datasets We use the following two text classification datasets to validate our hypothesis on different data distributions.

- *Jigsaw Toxicity*[2] is a dataset for toxicity classification that consists of 1,804,874 training examples, which we split into train and validation sets of size 1,443,900 and 360,974 respectively. We take 97,320 examples from the public leaderboard as the test set.
- *Bias in Bios* (De-Arteaga et al., 2019) [3] is a dataset for occupation classification derived from Common Crawl corpus. It consists of 178,619 train and 91,917 test examples.

Evaluation Metrics We evaluate models on three dimensions: (1) raw task performance, (2) model fairness, and (3) model robustness. For the raw task performance, we follow prior work in using accuracy and area under the ROC curve (AUC) to evaluate the performance of a model on the Bias in Bios dataset and the Jigsaw Toxicity dataset respectively. To measure the robustness of a model, we follow Jia et al. (2019) and Ye et al. (2020) to use the certified robustness accuracy (CRA). For fairness, we follow the discussion in Section 2.1 to evaluate a model based on *equalized odds* and *equal opportunity*.

For fairness, we measure two metrics - i.e, True Positive Equality Difference (TPED) and False Positive Equality Difference (FPED). The FPED and TPED is calculated as:

$$\sum_{z \in Z} |f_z - f_{overall}|,$$

where f is FPR or TPR depending on whether we are computing FPED or TPED, and Z refers to the set of all classes in a protected group. Note that TPED and FPED metrics do not take into account how well the model does - for example, a model that achieves a true positive rate of 0.0 for all groups will still have a TPED of 0.

For Bias in Bios dataset, we chose equality of opportunity to measure fairness, since it is important to ensure job candidates are matched with job recommendations that are relevant to them. Since equality of opportunity necessitates equality in true positive rates across cohorts, we use TPED as the fairness evaluation metric for Bias in Bios. For Jigsaw Toxicity, we define fairness by equalized odds, since it is important for toxicity classifiers to be able to detect toxicity in content containing identifiers across all groups, while not silencing any one. The combination of FPED with TPED aligns with the *Equalized Odds* definition of fairness (Borkan et al., 2019), thus we define a score *EOdds* as FPED + TPED for ease of analysis. Equalized odds is satisfied when FPED = 0 and TPED = 0, and thus when *EOdds* = 0.

For the scope of this paper and the limitations of the dataset, we study binary gender for Bias in Bios, and both gender (male, female, transgender, and non-binary) and sexual orientation (homosexual/straight, heterosexual, gay, lesbian, bisexual) for Jigsaw Toxicity classification. While we acknowledge that there are a multitude of important attributes, we constrain the scope of this study to the attributes present in text classification datasets.

Experiment Details All the experiments were conducted on p3dn.24xlarge and p3.2xlarge AWS compute nodes.[4] The IBP runs took 48 hours for

---

[2] The data is available at https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

[3] The data is available at https://github.com/microsoft/biosbias

[4] https://aws.amazon.com/ec2/instance-types/

Model	Raw task (↑)	Fairness (↓)			Robustness (↑)
	AUC	EOdds	FPED	TPED	CRA
Baseline	**0.957**	0.508	0.197	0.311	0.270
IBP	0.913	0.184	0.005	0.179	**0.934**
$\text{IBP}_{\text{gender}}$	0.947	0.237	0.062	0.175	0.912
Instance weighting	0.955	0.505	0.196	0.309	0.214
HardDebias	0.951	0.525	0.221	0.304	0.404
Adversarial Training	0.955	0.491	0.198	0.293	0.644
IBP + Instance weighting	0.889	**0.165**	**0.002**	**0.163**	0.942
IBP + *HardDebias*	0.923	0.459	0.169	0.290	0.890
IBP + Adversarial Training	0.920	0.473	0.192	0.281	0.901

Table 1: Certified robustness and bias mitigation methods with CNN on Jigsaw dataset. The best performance for each column is boldfaced. Results show that the certified robustness method (IBP) improves both robustness and fairness with performance drops on the raw task accuracy.

Model	Raw task (↑)	Fairness (↓)	Robustness (↑)
	AUC	TPED	CRA
Baseline	**0.787**	0.131	0.115
IBP	0.743	0.127	0.702
$\text{IBP}_{\text{gender}}$	0.749	0.104	0.711
Instance weighting	0.755	0.118	0.095
HardDebias	0.767	0.106	0.070
Adversarial Training	0.773	0.114	0.180
IBP + Instance weighting	0.732	0.113	**0.719**
IBP + *HardDebias*	0.735	**0.101**	0.715
IBP + Adversarial Training	0.725	0.112	0.693

Table 2: Experiment results on CNN models on the Bias in Bios dataset. We see that our best performing model consists of initiating IBP training with $HardDebias$ embeddings.

Jigsaw Toxicity and 34 hours for Bias in Bios, while SAFER took 53 hours with evaluation for Jigsaw Toxicity and 37 hours for Bias in Bios.

For the experiments with CNN, we follow Jia et al. (2019) to configure the IBP schedule and CNN models. In particular, we used a CNN model with a hidden size of 100 and kernel size of 3 with the GloVe embedding (Pennington et al., 2014) as inputs. For IBP, we linearly increased the weight on the certified robustness objective from 0 to 0.8 for 40 epochs, before training for 20 epochs on the full certified robustness objective.

For the experiments with BERT, we follow Ye et al. (2020) to configure the BERT model and SAFER experiment. We use `bert-base-uncased`, and take the top-100 words that are closest in cosine similarity for each token as the token's perturbation set. We describe the remaining hyper-parameter details (learning rate, epochs, dropout probability) in the the appendix, which we obtained after a hyper-parameter search on the development set.

4 Results

The results for Jigsaw Toxicity and Bias in Bios are in Tables 1, 2, 3 and 4.

Effect of certified robustness methods for mitigating bias We observe that adding IBP during training achieves better performance on fairness over othe bias mitigation approaches across Jigsaw Toxicity and Bias in Bios. In Jigsaw Toxicity, EOdds improves from 0.508 to 0.184, and in Bias in Bios, TPED improves from 0.131 to 0.127. Similarly, training models with SAFER results in an improvement in performance in all fairness metrics, with an improvement in EOdds from 0.553 to 0.286 in Jigsaw Toxicity and an improvement in TPED from 0.148 to 0.134 in Bias in Bios.

Effect of adding gender word substitutions to the robustness perturbation sets While adding gender word substitutions further improves fairness in Bias in Bios, it results in worse fairness scores in Jigsaw Toxicity than plain certified robustness methods. In Bias in Bios, IBP_{gender} results

Model	Raw task (↑)	Fairness (↓)			Robustness (↑)
	AUC	EOdds	FPED	TPED	CRA
Baseline	0.914	0.553	0.290	0.263	0.950
SAFER	0.918	**0.286**	**0.144**	**0.142**	**0.967**
SAFER$_{\text{gender}}$	**0.968**	0.347	0.176	0.171	0.917

Table 3: Model performance on BERT (Baseline) and SAFER on the Jigsaw dataset. Similar to the observation with IBP, SAFER improves both the fairness and robustness metrics.

Model	Raw task (↑)	Fairness (↓)	Robustness (↑)
	AUC	TPED	CRA
Baseline	**0.796**	0.148	0.164
SAFER	0.744	0.134	0.726
SAFER$_{gender}$	0.761	**0.097**	**0.733**

Table 4: Model performance on BERT (baseline) with SAFER on the Bias in Bios dataset.

in a lower TPED than all fairness only baselines. This trend holds in SAFER, where SAFER$_{gender}$ achieves lower TPED than SAFER. In Jigsaw Toxicity, adding gender words to the perturbation set degrades performance in equalized odds for both IBP and SAFER. This may be because the list of gender word substitutions do not include words relating to sexual orientation and non-binary gender, and thus may only improve fairness amongst examples containing male and female identifiers.

Effect of integrating bias mitigation methods with certified robustness methods Training model with both IBP and bias mitigation methods improves fairness metrics over fairness-only baselines in both datasets. In Bias in Bios, the model that comes closest to fulfilling equality of opportunity is the one trained with both IBP and $HardDebias$, which achieves a TPED of 0.101. In Jigsaw Toxicity, we see a similar trend, with improvements in EOdds after adding IBP training to instance weighting, $HardDebias$, and adversarial training. The model trained with both IBP and instance weighting achieves a EOdds score of 0.165, which is the lowest among all approaches

We also note that for Jigsaw Toxicity, instance weighting mitigates bias more effectively than $HardDebias$ and adversarial training (both in isolation and in combination with robustness methods). This is not the case for Bias in Bios, where $HardDebias$ and adversarial training is more effective than instance weighting in mitigating bias. This may be due to the fact that instance weighting mitigates bias explicitly for a wider array of sexual orientations and gender demographics than the other two methods. The original $HardDebias$ method only projects away the gender direction from embeddings. For adversarial debiasing, we train the adversary with the subset of the training set that is annotated for the presence of protected attribute groups, which is highly skewed towards male and female. Thus, $HardDebias$ and adversarial training may mitigate bias for binary gender, but fall short in mitigating bias for non-binary gender and sexual orientations. Conversely, instance weighting, which mitigates bias for a wider array of demographics, does not mitigate bias on gender in Bias in Bios as well as the other methods.

Additional Observations Outside of the effects of robustness on fairness, we observe differing effects of the methods on certified robustness and raw accuracy. As expected, IBP and SAFER improves performance on certified robustness on both datasets. However, we also observe degradataions in raw task accuracy in experiments with robustness methods. Combining robustness with bias mitigation methods results in a degradataion of raw task performance over fairness-only baselines. Additionally, fairness-only training results in differing effects on certified robustness accuracy. Adversarial debiasing improves certified accuracy in both datasets, while *HardDebias* embedding-initiated training results in an increase in certified robustness in Jigsaw Toxicity, but a decrease in Bias in Bios. This difference in findings may be due to the shorter length of examples in Jigsaw Toxicity, which has a median length of 34, compared with the median length of 72 in Bias in Bios, which in turn determines the number of possible perturbations used to calculate certified accuracy and the difficulty in achieving high CRA.

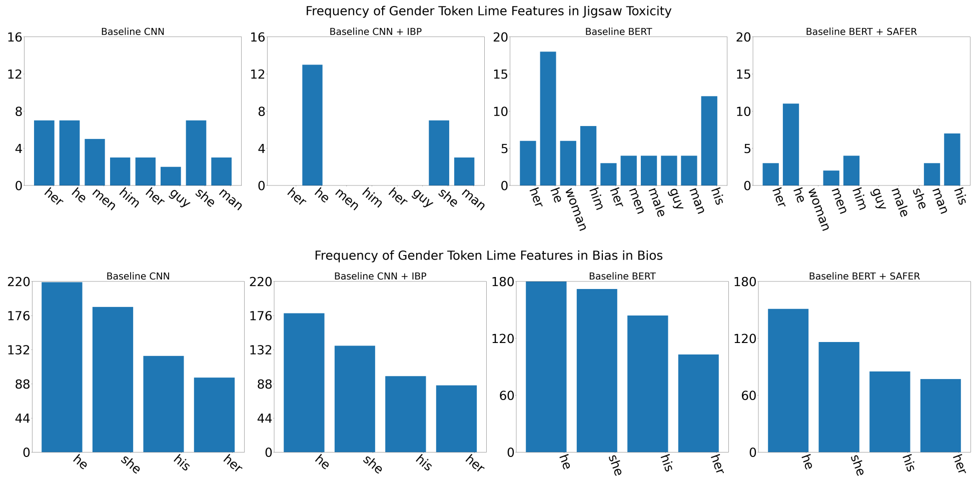

Figure 1: Frequency of gender token features as extracted by LIME for baseline and IBP trained models for Jigsaw Toxicity and Bias in Bios. We see a decrease in number and frequency of gender tokens in the list of top-5 (for Jigsaw Toxicity) and top-50 (for Bias in Bios) most important features.

5 Analysis

In this section, we study how robustness training affects the features our models use for classification. We posit that robustness training encourages models to focus more on predictive attributes than on protected attributes. To gain insight into this, we use LIME (Ribeiro et al., 2016) on the baseline, IBP, and SAFER trained models and extract token features importance as assigned by the model. We run LIME on the subset E of examples that are misclassified by our baseline model as toxic that are correctly classified by the IBP model. We take the top k features for each of the examples (where $k = 5$ for Jigsaw and $k = 50$ for Bias in Bios) over E, and then count the number of gender tokens that appear in that list. For Jigsaw Toxicity, the number of examples that we run LIME on is 488 for CNN experiments and 182 for BERT experiments. For Bias in Bios, we run LIME over a random subset of 500 examples from E for both CNN and BERT experiments.

For Jigsaw Toxicity, we see from Figure 1 that LIME extracts less gender tokens in the top-5 features of the IBP-trained and SAFER-trained model compared to the baseline model. Notably, there are 37 gender tokens that appear in the CNN model, while only 23 in the IBP-trained model. Similarly, 69 gender tokens appear in the baseline BERT model while only 37 appear in the SAFER-trained one. For Bias in Bios, we see a similar trend from Figure 1. The number of important gender token features decreases from 626 to 500 after IBP training, and from 626 to 429 after SAFER.

In addition, we compute the gradient with the output with respect to the input on several examples from Jigsaw Toxicity, which is shown in Table 5. We observe that the baseline model focuses on tokens related to protected groups, while the IBP model takes into account all parts of the sentence.

6 Related Work

Much work has been done in studying fairness in various NLP models (Mehrabi et al., 2019; Sun et al., 2019; Blodgett et al., 2020). In toxicity classification, Adragna et al. (2020) and (Zhang et al., 2020) study the fairness in predicting toxic internet contents in which the contents contain demographic identity-terms (e.g., “gay”, “black”). In occupation classification, De-Arteaga et al. (2019) and Romanov et al. (2019) study the impact of including explicit gender indicators such as a person’s names or a pronoun in online biographies.

Some notable bias mitigation methods, which we also use in this paper, include instance weighting (Zhang et al., 2020), embedding debiasing (Bolukbasi et al., 2016a; Wang et al., 2020), and adversarial debiasing (Zhang et al., 2018). In particular, Bolukbasi et al. (2016a) proposed to reduce representational harm existent in word embeddings. Zhang et al. (2020) proposed instance weighting, a method to debias text classification models for bias against examples containing demographic identity-terms by weighting the instances in the loss function, and that is optimized for demographic pairty. Zhang et al. (2018) presents an adversarial training approach to achieve various notions of fairness

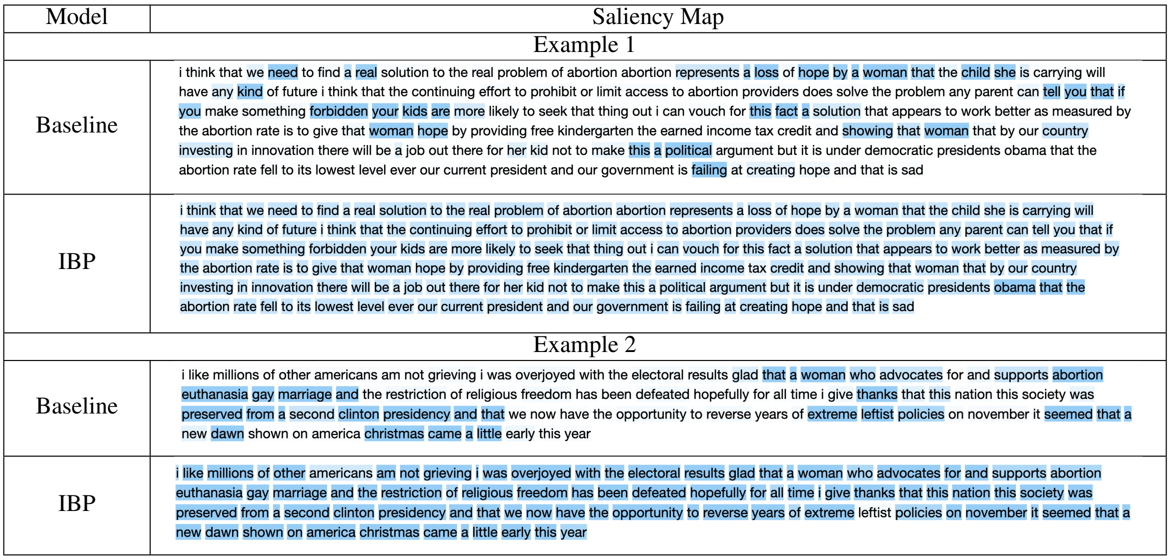

Model	Saliency Map
Example 1	
Baseline	i think that we need to find a real solution to the real problem of abortion abortion represents a loss of hope by a woman that the child she is carrying will have any kind of future i think that the continuing effort to prohibit or limit access to abortion providers does solve the problem any parent can tell you that if you make something forbidden your kids are more likely to seek that thing out i can vouch for this fact a solution that appears to work better as measured by the abortion rate is to give that woman hope by providing free kindergarten the earned income tax credit and showing that woman that by our country investing in innovation there will be a job out there for her kid not to make this a political argument but it is under democratic presidents obama that the abortion rate fell to its lowest level ever our current president and our government is failing at creating hope and that is sad
IBP	i think that we need to find a real solution to the real problem of abortion abortion represents a loss of hope by a woman that the child she is carrying will have any kind of future i think that the continuing effort to prohibit or limit access to abortion providers does solve the problem any parent can tell you that if you make something forbidden your kids are more likely to seek that thing out i can vouch for this fact a solution that appears to work better as measured by the abortion rate is to give that woman hope by providing free kindergarten the earned income tax credit and showing that woman that by our country investing in innovation there will be a job out there for her kid not to make this a political argument but it is under democratic presidents obama that the abortion rate fell to its lowest level ever our current president and our government is failing at creating hope and that is sad
Example 2	
Baseline	i like millions of other americans am not grieving i was overjoyed with the electoral results glad that a woman who advocates for and supports abortion euthanasia gay marriage and the restriction of religious freedom has been defeated hopefully for all time i give thanks that this nation this society was preserved from a second clinton presidency and that we now have the opportunity to reverse years of extreme leftist policies on november it seemed that a new dawn shown on america christmas came a little early this year
IBP	i like millions of other americans am not grieving i was overjoyed with the electoral results glad that a woman who advocates for and supports abortion euthanasia gay marriage and the restriction of religious freedom has been defeated hopefully for all time i give thanks that this nation this society was preserved from a second clinton presidency and that we now have the opportunity to reverse years of extreme leftist policies on november it seemed that a new dawn shown on america christmas came a little early this year

Table 5: Gradient saliency examples on Jigsaw Toxicity. Highlights show larger value of the output gradient with respect to the token embedding. The baseline CNN model focuses on some tokens related to protected groups (e.g., woman), while IBP encourages the model to take into account other parts of the sentence, resulting in less bias.

that is achieved by training an adversary to identify information on protected groups and training the model to minimize the adversary loss. These approaches are designed for reducing specific types of bias exhibited in data.

On the robustness front, it has been shown that models are susceptible to adversarial word substitution attacks (Ebrahimi et al., 2018; Jia and Liang, 2017). Parallel to the development of methods developed to reduce word substitution robustness in the NLP domain (e.g., (Miyato et al., 2017; Huang et al., 2019; Zhou et al., 2021)), many studies has been done in the computer vision domain to ensure that models are robust to image noising (Kannan et al., 2018; Szegedy et al., 2014).

In the intersection of area between fairness and robustness of model training, there is limited prior work in the NLP area. Nanda et al. (2020) investigate and define *robustness bias*, a notion of fairness in which a model must be impervious to perturbations to the same degree for all subgroups, and investigate robustness bias in the computer vision domain. Adragna et al. (2020) examine the use of invariant risk minimization in improving the fairness on out-of-distribution data for toxicity classification. Their robustness approach is inspired from domain generalization and it allows to learn models that have invariant performance across different label distributions. This differs from the word substitution notions of robustness that our methods are optimized for. Chang et al. (2020) shows that achieving equalized odds is incongruent with adversarial robustness on the COMPAS (J. Larson and Angwin, 2017) and the Adult dataset (Dua and Graf, 2017), which is outside the NLP domain. The closest work to ours is in counterfactual logit pairing (Garg et al., 2019), which encourages a model to be robust to protected attributes for counterfactual fairness. However, logit pairing does have the certified characteristic of the robustness methods we use in this study.

7 Conclusion

We present a study that investigates the effect of optimizing for word substitution robustness on fairness. We find that, in both CNN and BERT models, adding robustness methods such as IBP and SAFER to the training process improves fairness metrics over adding bias mitigation methods alone. Given these promising results, we encourage future explorations in using robustness methods to not only improve fairness metrics, but to also optimize for both fairness and robustness, two important aspects of creating trustworthy NLP.

Future work may include studying the effects of robustness and fairness in attributes other than gender and sexual orientation, extending our study to other word substitution based robustness methods, and exploring more sophisticated methods to combine robustness and bias mitigation methods during training. We also intend on extending the study to investigating the impact of privacy preserving training methods on both, robustness and fairness.

Broader Impact

We limit the scope of this paper to gender and sexual orientation in this initial effort, and future work must be done on mitigating bias in other protected attribute dimensions such as race, ethnicity, neurodiversity, etc. Additionally, this work draws importance to the need to extend fairness methods to groups beyond binary gender. In our IBP_{gender} experiments, we only consider swapping binary gender pairs from prior literature to provide an anchor for our analysis. We see from our results that methods that mitigate for binary gender such as *HardDebias* and IBP_{gender} do not reduce harm for all gender or sexual orientation, especially for non-binary gender and non-heterosexual sexual orientation groups. We will extend the study in the future by developing fairness methods that directly mitigate for non-heterosexual sexual orientations and non-binary genders pairs using sociology literature.

The language used in this paper is English. We recognize that the presented methods rely on the availability and quality of the set of words associated to a fairness task. Scaling to languages beyond English–such as gendered languages like Spanish–need more careful analysis. Another limitation of this method is that word substitution may lead to non-sensible sentences and inappropriate grammar especially in complex fairness domains where it is difficult to find word-to-word mapping (e.g., mapping names of religious artifacts like Christmas tree or Diwali lights, etc are not trivial).

Our experiments and results show that pursuing fairness can help in improving robustness and vice versa. With these findings, we hope to inspire researchers to investigate novel approaches that focus on jointly achieving robust and fair models. We also hope that this work will lead to more investigations around achieving multiple objectives such as privacy, robustness and fairness together in the NLP research community.

References

Robert Adragna, Elliot Creager, David Madras, and R. Zemel. 2020. Fairness and robustness in invariant learning: A case study in toxicity classification. *ArXiv*, abs/2011.06485.

Muhammad Aurangzeb Ahmad, Arpit Patel, Carly Eckert, Vikas Kumar, and Ankur Teredesai. 2020. Fairness in machine learning for healthcare. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3529–3530.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, M. Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *EMNLP*.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and A. Kalai. 2016a. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *ArXiv*, abs/1607.06520.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016b. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29:4349–4357.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 491–500.

Hongyan Chang, Ta Duy Nguyen, Sasi Kumar Murakonda, Ehsan Kazemi, and Reza Shokri. 2020. On adversarial bias and the robustness of fair machine learning. *ArXiv*, abs/2006.08669.

Maria De-Arteaga, Alexey Romanov, H. Wallach, J. Chayes, C. Borgs, A. Chouldechova, Sahin Cem Geyik, K. Kenthapadi, and A. Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.

Dheeru Dua and Casey Graf. 2017. Ouci machine learning repository. http://archive.ics.uci.edu/ml.

J. Ebrahimi, Anyi Rao, Daniel Lowd, and D. Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *ACL*.

Sahaj Garg, V. Perot, Nicole Limtiaco, Ankur Taly, Ed Huai hsin Chi, and Alex Beutel. 2019. Counterfactual fairness in text classification through robustness. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*.

Robert Gorwa, Reuben Binns, and Christian Katzenbach. 2020. Algorithmic content moderation: Technical and political challenges in the automation of platform governance. *Big Data Society*, 7.

M. Hardt, E. Price, and Nathan Srebro. 2016. Equality of opportunity in supervised learning. In *NIPS*.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. 2020. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4083–4093.

B. Hutchinson, Vinodkumar Prabhakaran, Emily L. Denton, K. Webster, Yu Zhong, and Stephen Denuyl. 2020. Social biases in nlp models as barriers for persons with disabilities. *ArXiv*, abs/2005.00813.

L. Kirchner J. Larson, S. Mattu and J. Angwin. 2017. Compas dataset. https://github.com/propublica/compas-analysis.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *ArXiv*, abs/1707.07328.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. *ArXiv*, abs/1909.00986.

Shengyu Jia, Tao Meng, Jieyu Zhao, and Kai-Wei Chang. 2020. Mitigating gender bias amplification in distribution by posterior regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2936–2942.

Harini Kannan, A. Kurakin, and Ian J. Goodfellow. 2018. Adversarial logit pairing. *ArXiv*, abs/1803.06373.

J. Kemp, Alvin Rajkomar, and Andrew M. Dai. 2019. Improved hierarchical patient classification with language model pretraining over clinical notes. *arXiv: Learning*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

P. P. Liang, Irene Z Li, E. Zheng, Yao Chong Lim, R. Salakhutdinov, and Louis-Philippe Morency. 2020. Towards debiasing sentence representations. In *ACL*.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and A. Galstyan. 2019. A survey on bias and fairness in machine learning. *ArXiv*, abs/1908.09635.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of the International Conference on Learning Representations*.

Vedant Nanda, S. Dooley, Sahil Singla, S. Feizi, and John P. Dickerson. 2020. Fairness through robustness: Investigating robustness disparity in deep learning. *ArXiv*, abs/2006.12621.

Thiago Dias Oliva, Dennys Marcelo Antonialli, and Alessandra Gomes. 2020. Fighting hate speech, silencing drag queens? artificial intelligence in content moderation and risks to lgbtq voices online. *Sexuality & Culture*, pages 1–33.

J. Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Alexey Romanov, Maria De-Arteaga, H. Wallach, J. Chayes, C. Borgs, A. Chouldechova, Sahin Cem Geyik, K. Kenthapadi, Anna Rumshisky, and A. Kalai. 2019. What's in a name? reducing bias in bios without access to protected attributes. *ArXiv*, abs/1904.05233.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyridon Matsoukas. 2018. A re-ranker scheme for integrating large scale nlu models. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 670–676.

T. Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, M. ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth M. Belding-Royer, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. *ArXiv*, abs/1906.08976.

Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and R. Fergus. 2014. Intriguing properties of neural networks. *CoRR*, abs/1312.6199.

Y. Tan and L. Celis. 2019. Assessing social and intersectional biases in contextualized word representations. In *NeurIPS*.

Tianlu Wang, Xi Victoria Lin, Nazneen Rajani, Vicente Ordonez, and Caimng Xiong. 2020. Double-hard debias: Tailoring word embeddings for gender bias mitigation. *ArXiv*, abs/2005.00965.

M. Ye, Chengyue Gong, and Qiang Liu. 2020. Safer: A structure-free approach for certified robustness to adversarial word substitutions. *ArXiv*, abs/2005.14424.

B. H. Zhang, B. Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*.

Guanhua Zhang, Bing Bai, Junqi Zhang, Kun Bai, Conghui Zhu, and Tiejun Zhao. 2020. Demographics should not be the reason of toxicity: Mitigating discrimination in text classifications with instance weighting. In *ACL*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada. Association for Computational Linguistics.

Jieyu Zhao and Kai-Wei Chang. 2020. LOGAN: Local group bias detection by clustering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1968–1977, Online. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018a. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018b. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. In *ACL*.

A Appendices

Appendix A. Hyperparameter Settings We perform hyperparameter search on the dev set using random search with 12 trials, with initial learning rate range between $1 * 10^{-2}$ to $1 * 10^{-7}$, a dropout probability range of 0.1 to 0.5, and number of epochs between 10 and 60.The final hyperparameter settings are shown in Table 6. We choose our hyperparameters based on the one that minimizes FPED + TPED + (1 - CRA) + (1 - *tp*), where *tp* refers to task performance.

Additionally, for adversarial debiasing, we tune the adversary loss weight from $\alpha = 0.1$ to $\alpha = 3$, and choose $\alpha = 1$ for the weight. We pretrain our classifier and adversary for 2 epochs each.

Experiment	**Learning Rate**	**Dropout Prob**	**Number of epochs**
GloVe + CNN (Jigsaw)	1e-2	0.5	20
GloVe + CNN (Bias in Bios)	1e-3	0.1	15
BERT + SAFER (Jigsaw)	5e-6	0.1	20
BERT + SAFER (Bias in Bios)	1e-5	0.1	15

Table 6: Hyperparameter settings for our experiments. We use the same hyperparameters across our fairness and robustness experiments.

A Joint Model for Structure-based News Genre Classification with Application to Text Summarization

Zeyu Dai, Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
{jzdaizeyu, huangrh}@tamu.edu

Abstract

Journalists usually organize and present the contents of a news article following a well-defined structure. In this paper, we propose a novel joint model for structure-based news genre classification that simultaneously identifies one of four commonly used news structures (including *Inverted Pyramid* and three other structures) for a news article as well as recognizes a sequence of news elements within the article that define the corresponding news structure. Experiments show that the joint model consistently outperforms its variants that perform two tasks independently, which supports our motivation that preserving the two-way dependencies and constraints between a type of news structure and its sequence of news elements enables the model to better predict both of them. Although being not perfect, the system predicted news structure type and news elements have improved the performance of text summarization when incorporated into a recent neural network system.

1 Introduction

Journalists usually organize and report the contents of news following a well-defined structure. For example, when writing news briefs or breaking news, the *Inverted Pyramid* structure (Pottker, 2003) is often adopted to present the most newsworthy and key events first and then provide any additional details. However, while being commonly used, *Inverted Pyramid* is not the only news structure, there exist several other commonly used news structures as well, for example, a structure called *Kabob* is commonly used to present a narrative hook (Myers and Wukasch, 2003) first and then report the main story, where the narrative hook catches the reader's attention so that reader is willing to keep reading. Recognizing the overall structure of a news article can benefit many NLP tasks and applications, such as text summarization, text segmentation, discourse analysis, information extraction and text quality assessment, and many others.

Our recent research (Dai et al., 2018) first defines a small set of news elements, specifically five news elements, and then formally defines four commonly used news structures based on their different ways to select and organize news elements. News elements are defined based on their functions in a news story (introducing the main story or event, catching the reader's attention or providing details, etc.) as well as their writing styles (narrative or expository, also known as modes of discourse). Specifically, five news elements are defined, including two ledes, *Standard Lede* and *Image Lede*, with their functions as either introducing the main story or catching the reader's attention, as well as three other categories, *Synopsis*, *Narration* and a catch-all category *Body Section*. Each news element is realized as a set of one or more consecutive paragraphs in a news article. Using the well-defined news elements, four news structures, *Inverted Pyramid*, *Kabob*, *Martini Glass* and *Narrative* are introduced. The *Inverted Pyramid* structure can be represented as a *Standard Lede* followed by a *Body Section*, while the *Kabob* structure can be represented as an *Image Lede* followed by a *Synopsis* and a *Body Section*. Two more news structures, *Martini Glass* and *Narrative*, are defined and each of them has the *Narration* news element. We defer more details about news elements and news structures to the section 3.

Our previous work (Dai et al., 2018) created a dataset (the News Genre dataset) with both news structures and news elements annotated for structure-based news genre categorization, and has conducted news structure classification as a text classification task by building a machine learning classifier (SVM) using n-grams and several structure indicative features. However, we have not attempted to further recognize the annotated news elements within a news article yet. As each news

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3332–3342
August 1–6, 2021.

element carries a specific function in building a news story and features a writing style (narrative or expository), the recognized news elements are expected to be useful for many NLP applications.

In this work, we take one step further and propose to recognize both the news structure type of a news article as well as its corresponding news elements. We first implemented two pipeline approaches that first predict document-level news structure (or paragraph-level news element) tags using one single model, and then incorporate the predicted tags as features into another single model for predicting news element (or news structure) tags. Then, inspired by the idea that the overall news structure of a document determines the sequence of news elements within the document, and vice versa, we aim to recognize both the type of news structure and its news elements simultaneously in a joint model. Specifically, we build our joint model on top of a hierarchical BiLSTM neural networks that learn paragraph and document representations for predicting both a news structure type for a document and a sequence of news element tags for its paragraphs. The intrinsic evaluation on the News Genre dataset shows that the joint model consistently outperforms the pipeline models that accomplish two tasks independently, and achieves noticeable performance gains for predicting all four types of news structures and all five types of news elements, which supports our motivation that preserving the two-way dependencies and constraints between a news structure and its news elements enables the system to better predict both of them.

We believe that the identified news structures and news elements can be useful for many text-level NLP applications and tasks. In this paper, we further conduct experiments and use system predicted news structure and news element tags for improving text summarization. Informed by the predicted news structure genres, we expect to better locate the key event descriptions of a news story, and therefore improve the performance of extractive summarization models. Especially, we expect that recognizing news structures and news elements can boost the text summarization performance on news articles of a particular news structure, the *Kabob* structure, which is the second most frequent news genre and covers roughly 28% of news articles based on the annotated News Genre dataset.

For news documents with the *Kabob* structure, the beginning paragraphs (corresponding to a news element called *Image Lede*) do not directly present the key events of news, instead, the following paragraphs (corresponding to a news element called *Synopsis*) will summarize the main story. Therefore, this news genre brings additional difficulty to locate the correct paragraphs for extracting summary, and accordingly, recognizing this genre and its news elements is likely to noticeably improve text summarization performance on documents with the *Kabob* structure the most. Indeed, the extrinsic evaluation on the CNN/DailyMail dataset (Hermann et al., 2015) shows that a simple method for incorporating news genre tags as word features into a recent extractive summarization system (Liu and Lapata, 2019) improves the three ROUGE (Lin, 2004) scores, R-1, R-2 and R-L, consistently for all four types of news structure genres, with the *Kabob* structure receiving the largest improvements of 0.37, 0.14 and 0.34 points on R-1, R-2 and R-L respectively.

2 Related Work

News structures have been extensively studied in the area of linguistics and journalism (Schokkenbroek, 1999; Van Dijk, 1985; Ytreberg, 2001). However, few computational studies tried to automatically categorize news articles according to news structures using data-driven methods. Our previous work (Dai et al., 2018) is the first work we are aware of that formulated four news structures using a small set of predefined news elements, created the first dataset for structure-based news genre categorization, and proposed a feature-based classifier to predict the news structure type of a document. With the motivation to better serve the needs of downstream applications, we developed a computational system to recognize news elements within a document as well as the overall news structure type. We built a joint model for these two tasks to preserve the two-way dependencies and constraints between them, and have empirically improved the performance of both tasks.

In the previous work, several well-studied genre-independent discourse structures have been explored for improving many NLP applications. For example, discourse structures including the RST-style tree structure (Mann and Thompson, 1988) and the PDTB-style discourse relations (Prasad et al., 2008) have been shown useful for a range of NLP applications, such as sentiment analysis (Bhatia et al., 2015; Märkle-Huß et al., 2017), text summarization (Marcu, 1997; Louis et al., 2010) and machine translation (Li et al., 2014; Guzmán et al., 2014). In addition, text segmentation (Hearst, 1994) that divides a text into a sequence of topically coherent segments by detecting topic transition boundaries have been shown useful for text summarization (Barzilay and Lee, 2004), sentiment analysis (Sauper et al., 2010) and dialogue systems (Shi et al., 2019). We believe that the genre-specific news structures can effectively complement the genre-independent discourse structures, and both of them are essential for achieving deep story-level text understanding.

In this work, we further apply our system predicted news structure and news element tags to help the task of extractive summarization, which aims to extract a summary by identifying the most important sentences in a news article. Nallapati et al. (2017) presents one of the earliest neural network systems for extractive summarization that adopt an RNN-based encoder for abstracting sentence representations. More recent work achieves higher performance for extractive summarization using more sophisticated neural network structures. SUMO (Liu et al., 2019) introduces structured attention to induce a dependency tree representation of a document while generating a summary. Liu and Lapata (2019) adapts BERT (Devlin et al., 2019) to text summarization which obtains contextualized representations of a document and its sentences using BERT's encoder by stacking several inter-sentence Transformer layers. Dong et al. (2019) fine-tunes a new Unified pre-trained Language Model (UniLM) for text summarization by employing a shared Transformer network and utilizing specific self-attention masks to control which context the predicting summary conditions on. The extrinsic evaluation on text summarization using (Liu and Lapata, 2019) as baseline demonstrates the usefulness of our system predicted genre-specific news structure tags in downstream NLP tasks.

In addition, our work is also related to text genre identification (Santini, 2007; Mehler et al., 2010; Rehm, 2002), but we focus on the genres of news structure which come from the area of journalism.

3 Structure-based News Genres

As shown in Figure 1, our previous work (Dai et al., 2018) formally defined four commonly used news structures based on the selection and organization of five predefined news elements.

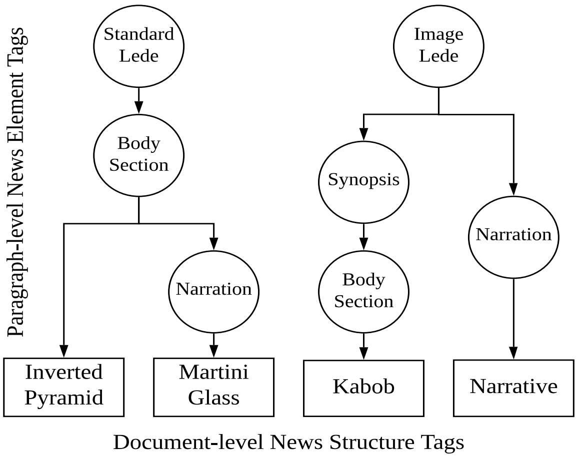

Figure 1: Four News Structures: Document-level News Structure Tags (in rectangle) and Paragraph-level News Element Tags (in circle). A News Element may include one or more consecutive paragraphs.

3.1 Five Paragraph-level News Elements

Standard Lede is used to introduce the key events and main story at the beginning of a news article; written in the expository style.

Image Lede [1] is used to catch the reader's attention by telling an anecdote, quoting a catchy slogan, or revealing an impressive fact or statistics (Jou, 2014); written in either narrative or expository style. Image Lede is located at the beginning of a news article as well, however, unlike Standard Lede, it does not directly discuss the key events of a news article, therefore, it may not represent a good summary of the news article.

Synopsis must follow an Image Lede and acts as a bridge that connects an Image Lede with the rest of a story. The function of Synopsis is to summarize the key events and main story of a news article; written in the expository style.

Narration gives great details about key events and often contains a sequence of events (or subevents) in chronological order (Mani, 2012); written in the narrative style (Lavelle, 1997).

Body Section presents additional details and supplementary information about key events; written in the expository style. Paragraphs that do not belong to any of the four above categories were annotated as a Body Section (Dai et al., 2018).

3.2 Four Document-level News Structures

Inverted Pyramid, known as the most popular news article structure (Pottker, 2003), presents the

---

[1] In some news articles, an image is presented first to catch the readers' eyes and the Image Lede acts as the description of the image.

content in the descending order of importance and relevance (Scanlan, 2003). For this structure, key events and main story will be introduced first, then additional information will be provided later. This structure is represented as a Standard Lede followed by a Body Section, shown in Figure 1.

Martini Glass (Jou, 2014) begins by presenting a summary of a story following the Inverted Pyramid structure, and then transitions into a chronological elaboration of the story in detail. Therefore, the Martini Glass structure contains a Standard Lede, an optional Body Section and a Narration.

Kabob (Jou, 2014) first tries to catch the reader's eyes using an anecdote (or a catchy slogan, etc), then introduces the key events, and discusses the main story with more details at last. Therefore, the Kabob structure is defined to start with an Image Lede, then uses a Synopsis as the transition, and finally ends with a Body Section.

Narrative structure presents a chronologically ordered sequence of events with a greater amount of details than normal news articles. Dai et al. (2018) annotated this news structure when the majority of paragraphs form a single Narration with an optional preceding Image Lede.

3.3 The News Genre Dataset

Dai et al. (2018) created the first structure-based news genre dataset [2]. This dataset contains 853 English news articles across four news domains, including politics, crime, business and disaster. In this dataset, each article was annotated with a news structure label and a sequence of news element tags for its paragraphs. The same news element tag will be assigned to all paragraphs in a consecutive sequence that a news element spans over.

The four common news structures applied to most of the annotated news articles, with only 21 documents were not annotated with any of the four news structures and did not receive paragraph-level news element tags either, so we removed these 21 documents in our experiments. Table 1 shows the statistics of news structure and news element tags, from which we can see that the distribution of news structures is highly imbalanced, with *Inverted Pyramid* and *Kabob* as two major structure types.

News Structure	#	News Element	#
Inverted Pyramid	482	Standard Lede	519
Martini Glass	37	Image Lede	244
Kabob	237	Synopsis	237
Narrative	76	Narration	113
Total	832	Body Section	746

Table 1: Data Statistics of the News Genre Dataset.

4 Model

4.1 The Joint Model for Predicting both News Structures and News Elements

Figure 2 illustrates the overall architecture of our joint model, which can simultaneously predict both document-level news structure label and paragraph-level news element tags. The model processes a whole news article containing a sequence of paragraphs each time, and predicts a document-level label as well as a sequence of paragraph-level tags with one tag for each paragraph using the standard BIO tagging schema (Ratinov and Roth, 2009) for sequence labeling. Specifically, we treat the news element *Body Section* as the "other" (or 'O') tag since this tag can't help determine document-level news structure type (shown in Figure 1) and was used as a catch-all "other" label during the data annotation as well. For other paragraph-level news element tags except for the *Body Section*, we assign a "B-" prefix to the first paragraph that starts the news element and assign "I-" prefix to other paragraphs inside the same news element.

The model employs the two-level hierarchical BiLSTM layers (Schuster and Paliwal, 1997) with max-pooling (Collobert and Weston, 2008) operation in between to learn both word and paragraph representations, followed by a max-pooling operation to calculate the document representation and a softmax classification layer for predicting the document-level label. Added on top of the paragraph-level representations, a linear-chain Conditional Random Field (CRF) layer (Lafferty et al., 2001) is utilized to jointly decode a sequence of paragraph-level tags considering their inter-dependencies. As shown in Figure 2, the model consists of the following components:

Feature-rich Word Vector: Given a sequence of words $(w_1, w_2, ..., w_L)$ as the input document, for each word w_i, we construct a feature-rich word vector by concatenating its word embedding $\boldsymbol{w}_i^{word}$ with its character-level representation [3], and extra

---

[2] Available at `https://github.com/ZeyuDai/Fine-grained_Structure-based_News_Genre_Categorization`

[3] For character-level representation, we adopted one layer

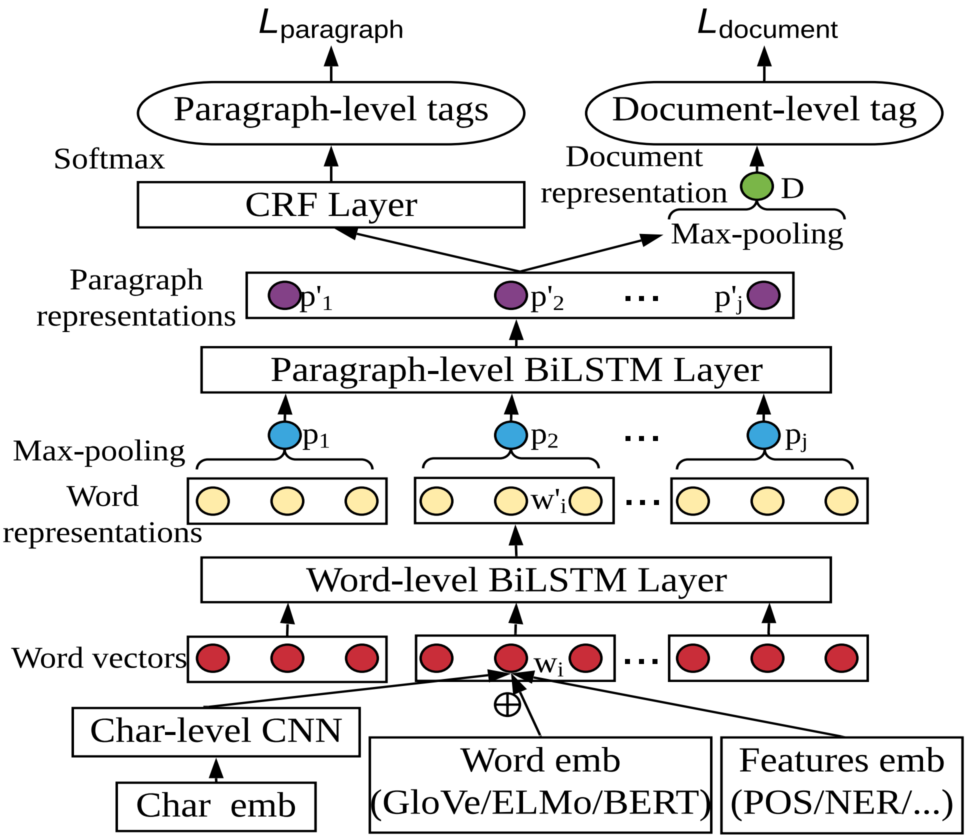

Figure 2: The Joint Model Architecture for both Document-level and Paragraph-level News Genre Tags Prediction.

word-level features embedding [4] as:

$$\boldsymbol{w}_i = [\boldsymbol{w}_i^{word}; \boldsymbol{w}_i^{char}; \boldsymbol{w}_i^{features}]$$

To take advantage of the recent progress about contextualized word representation from pre-trained language models, our framework supports three options including 300 dimensional GloVe (Pennington et al., 2014), 1024 dimensional ELMo (Peters et al., 2018) and the "bert-base-cased" version of BERT (Devlin et al., 2019) to initialize [5] the $\boldsymbol{w}_i^{word}$.

Word-level BiLSTM Layer: Given a sequence of feature-rich word vectors $(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_L)$ as the input, the word-level BiLSTM layer will refine the word w_i's hidden representation $(\boldsymbol{w}_i')$ by modeling the word-level inter-dependencies:

$$\boldsymbol{w}_i' = BiLSTM(\boldsymbol{w}_1, ..., \boldsymbol{w}_i, ..., \boldsymbol{w}_L)$$

Paragraph-level BiLSTM Layer: Given a sequence of word representations $(\boldsymbol{w}_1', \boldsymbol{w}_2', ..., \boldsymbol{w}_L')$, we build the paragraph representation $(\boldsymbol{p}_j)$ for the j-th paragraph in the document, by applying max-pooling operation over the sequence of word representations for all words within the j-th paragraph:

$$\boldsymbol{p}_j = \max_{w_i \in p_j} \boldsymbol{w}_i'$$

Then, the paragraph-level BiLSTM layer will update the j-th paragraph's hidden representation $(\boldsymbol{p}_j')$ by modeling the paragraph-level inter-dependencies:

$$\boldsymbol{p}_j' = BiLSTM(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_j, ...)$$

Softmax Classification Layer for Document-level News Structure Type Prediction: We compute the document representation $(\boldsymbol{D})$ by applying max-pooling operation over all paragraph representations $(\boldsymbol{p}_1', \boldsymbol{p}_2', ..., \boldsymbol{p}_j', ...)$.

Then, for the i-th training instance with $\boldsymbol{y}_{doc\_gold}^{(i)}$ as the gold annotation of document-level tag, our model predict the document-level tag $\boldsymbol{y}_{doc\_pred}^{(i)}$ using the softmax classification layer:

$$\boldsymbol{y}_{doc\_pred}^{(i)} = softmax(\boldsymbol{W}_{doc}\boldsymbol{D}^{(i)} + \boldsymbol{b}_{doc})$$

---

of CNN with 50 hidden units followed by a max-pooling layer.

[4] For word-level features, we collected the corresponding paragraph's position (PARA) index, capitalization (CAP) flag, Part-of-speech (POS) tag and named entity (NER) tag of each word. The embedding sizes for PARA/CAP/POS/NER were 20/5/35/20 respectively. We used Standford CoreNLP toolkit (Manning et al., 2014) to generate POS and NER tags.

[5] GloVe embeddings were fixed during training. For ELMo and BERT, we also froze its parameters during model training.

And we want to minimize the following cross-entropy loss during model training:

$$L_{document} = -\sum_{i} \boldsymbol{y}^{(i)}_{doc\_gold} * \log \boldsymbol{y}^{(i)}_{doc\_pred}$$

CRF Layer for Paragraph-level News Elements Sequence Labeling: For the task of sequence labeling, it is important to model the label dependencies (e.g., “I-*” must follow “B-*” in BIO tagging schema.) and capture the label continuity and transition patterns. Therefore, a CRF layer is added on top of the paragraph-level BiLSTM layer to jointly decode the news element tags sequence.

For the i-th training instance, given the annotated paragraph-level news element tags sequence $\boldsymbol{y}^{(i)}_{para\_gold} = (y^{(i)}_1, y^{(i)}_2, ..., y^{(i)}_j, ...)$ and hidden paragraph representations $\boldsymbol{P'}^{(i)} = (\boldsymbol{p'}^{(i)}_1, \boldsymbol{p'}^{(i)}_2, ..., \boldsymbol{p'}^{(i)}_j, ...)$, we minimize the following CRF loss during model training:

$$L_{paragraph} = -\sum_{i} \log p(\boldsymbol{y}^{(i)}_{para\_gold} | \boldsymbol{P'}^{(i)})$$

For model testing, we use the Viterbi algorithm to search for the optimal label sequence.

Joint Model vs. Single Model Training: The overall loss function for training our joint model is:

$$L = L_{document} + L_{paragraph}$$

Clearly, we can easily make it a single-task model for either document-level news structure type prediction or paragraph-level news element sequence labeling, by removing unrelated loss term from the overall loss function. We will compare the performance of our joint model with single models in the following intrinsic evaluation section 5.

4.2 Parameter Settings and Implementation Details

We manually tuned all hyperparameters of our model based on the development set using the macro-average F1-score as the selection criterion. After the hyperparameter search, we used the hidden size of 512 (tuned from the list [100, 300, 512, 1024]) for each BiLSTM layer and all hidden representations ($\boldsymbol{w'}_i, \boldsymbol{p}_j, \boldsymbol{p'}_j, \boldsymbol{D}$). For regularization, we applied 50% (tuned from [10%, 20%, 30%, 50%]) dropout to both input and output vectors of each BiLSTM layer. To alleviate the problem of gradient exploding for BiLSTM training, we clipped the gradient L2 norm at threshold 5.0 (tuned from [5.0, 10.0]) and utilized L2 regularization with coefficient 10^{-6}. Parameters were optimized using SGD optimizer with momentum 0.9 (tuned from [0.9, 0.95] and no momentum) and initial learning rate 0.015 (tuned from [0.0001, 0.001, 0.01, 0.015, 0.05, 0.1]), decreasing by 5% after each epoch. The batch size was 32 (tuned from [8, 16, 32, 64]) in the normal case, but it will be much smaller (1 or 2 depending on the model size) when using BERT because of the GPU CUDA memory limitation.

News Structure	#	News Element	#
Inverted Pyramid	434	Standard Lede	467
Martini Glass	33	Image Lede	219
Kabob	214	Synopsis	214
Narrative	69	Narration	102
Total	750	Body Section	673

Table 2: Data Statistics of the Cross-validation Set.

We implemented our model using Pytorch, with ELMo from AllenNLP [6] and BERT-base from HuggingFace [7]. Since BERT used the subword tokenizer, we used the first token’s representation as word embedding if one word was split into several subword tokens. We trained our model for 50/20/3 epochs when using GloVe/ELMo/BERT word embeddings respectively, considering that different word representation techniques require a different number of fine-tuning epochs. To diminish the effects of randomness in neural network training, we ran our proposed model, its variants as well as our own baselines using 5 different random seeds and the reported performance is the average score across 5 runs. The full model training took around 8-12 hours on one NVIDIA GTX 1080Ti GPU.

5 Intrinsic Evaluation

5.1 Experimental Settings

Considering that the News Genre corpus is relatively small and cross-validation is more robust for a small dataset, we followed our previous work (Dai et al., 2018) and evaluated our models using 5-fold cross-validation. Specifically, we created our own cross-validation/development set splits containing 750/82 news articles respectively, and randomly split the cross-validation set into five folds with even domain distribution. Table 2 reports the distribution of news structure and element tags on

---

[6] `https://github.com/allenai/allennlp`

[7] `https://github.com/huggingface/transformers`

	Document-level News Structure Types						Paragraph-level News Element Tags						
Model	Acc	Mac	IP	MG	Kab	Nar	Acc	Mac	SL	IL	Sy	Na	BS
Feature-based (2018)	71.8	50.3	81.2	17.8	54.3	48.0	-	-	-	-	-	-	-
Our Models													
Single Model (GloVe)	75.6	51.8	81.7	19.5	56.0	50.2	73.4	48.6	67.0	28.8	28.8	36.0	82.6
Single Model (ELMo)	78.0	54.2	84.0	22.0	58.6	52.0	76.0	50.9	68.3	30.4	30.4	38.0	87.2
Single Model (BERT)	77.6	53.6	83.5	21.5	58.0	51.5	75.6	50.3	68.0	30.2	30.2	37.5	85.5
Joint Model (GloVe)	77.8	53.8	83.7	21.7	58.3	51.6	75.8	50.6	68.2	30.0	30.0	38.0	86.6
Joint Model (ELMo)	**80.0**	**56.0**	**86.0**	**24.6**	**60.5**	**52.8**	**78.3**	**53.2**	**70.5**	**32.4**	**32.4**	**40.5**	**90.4**
Joint Model (BERT)	79.2	55.5	85.5	24.2	60.0	52.2	77.6	52.4	70.0	32.0	32.0	38.2	90.0
Pipeline Models (ELMo)													
Pipeline (doc → *para*)	78.0	54.2	84.0	22.0	58.6	52.0	77.4	52.2	69.7	31.6	31.6	39.2	89.0
Pipeline (para → *doc*)	78.8	55.0	84.8	23.4	59.5	52.4	76.0	50.9	68.3	30.4	30.4	38.0	87.2

Table 3: Intrinsic Evaluation Results on the Cross-validation Set of News Genre Dataset using 5-fold Cross-validation. We report accuracy (Acc), macro-average F1-score (Mac), and class-wise F1-scores for document-level structure and paragraph-level element tags, including Inverted Pyramid (IP), Martini Glass (MG), Kabob (Kab), Narrative (Nar), Standard Lede (SL), Image Lede (IL), Synopsis (Sy), Narration (Na) and Body Section (BS).

the cross-validation set. The hyperparameter tuning was conducted on the development set using the cross-validation set for model training.

5.2 Baselines

Feature-based (Dai et al., 2018): To compare with previous work, we replicated the feature-based model of (Dai et al., 2018) that performs document-level news structure type classification only.

Pipeline (doc → para) && Pipeline (para → doc): We implemented two pipeline approaches that first predict document-level news structure (or paragraph-level news element) tags using our single model, and then incorporate the predicted tags as word-level features (with embedding size 10) into another single model for predicting paragraph-level (or document-level) tags. The pipeline approach that first predicts document-level news structure tags is marked as Pipeline (doc → para); the reverse one is marked as Pipeline (para → doc).

5.3 Experimental Results

Table 3 summarizes the evaluation results on the cross-validation set using 5-fold cross-validation. The first row shows the performance of our replicated feature-based baseline (Dai et al., 2018) which achieves similar performance as in the original paper. The second section reports the performance of our models for predicting both document-level news structure types and paragraph-level news element tags, which compares the results of our models trained with different loss functions (joint model vs. single model) when using different word embeddings (GloVe vs. ELMo vs. BERT).

We can see that the joint model consistently outperforms (statistical significant t-test with $p < 0.05$) the corresponding single model independent from the word embeddings, which supports our motivation that document-level news structure type identification can not be separated from learning paragraph-level news element representations and features, and vice versa. Among the three word representation techniques, the ELMo word embeddings consistently give the best performance, followed by BERT and GloVe. One possible reason why BERT performs worse in our experiments is that we have to use a very small batch size and large learning rate when using BERT due to the limitation of GPU CUDA memory. The best joint model using the ELMo embeddings achieves 80.0% accuracy and 56.0% macro F1-score for predicting document-level news structure types, which outperforms the previous feature-based baseline by a large margin, and simultaneously achieves 78.3% accuracy and 53.2% macro F1-score for identifying paragraph-level news element tags.

The third section shows the performance of the two pipeline models. Note that, for fair comparisons, both pipeline models use the ELMo word embeddings that perform the best for our tasks (in both single and joint models). We can see that our joint model consistently outperforms both pipeline approaches. This is reasonable because pipeline models suffer from error propagation which poses an even bigger challenge in our task when the predicted news element sequence can not be compatible with any of the four news structure types.

In addition, Table 4 reports the experimental results on the development set, where we used the whole cross-validation set for training the models. On the development set, we observe similar comparisons among models and consistent performance gains achieved by the joint model.

	Document-level News Structure Types						Paragraph-level News Element Tags						
Model	Acc	Mac	IP	MG	Kab	Nar	Acc	Mac	SL	IL	Sy	Na	BS
Feature-based (2018)	72.8	51.2	81.7	18.5	56.1	48.5	-	-	-	-	-	-	-
Our Models													
Single Model (GloVe)	76.2	52.7	82.4	19.6	56.9	51.7	74.6	49.1	68.0	29.5	29.5	35.9	83.0
Single Model (ELMo)	78.8	54.6	84.9	22.2	59.0	52.3	76.8	51.3	68.6	30.8	30.8	38.5	87.8
Single Model (BERT)	78.4	54.3	84.5	22.7	58.8	51.2	76.2	50.5	68.4	30.4	30.4	37.1	86.3
Joint Model (GloVe)	78.5	54.5	84.4	23.1	58.9	51.4	76.2	50.7	68.1	30.0	30.0	38.5	86.9
Joint Model (ELMo)	**81.1**	**56.5**	**86.6**	**25.2**	**60.9**	**53.1**	**79.4**	**53.8**	**71.0**	**33.2**	**33.2**	**40.8**	**90.6**
Joint Model (BERT)	79.5	55.6	85.7	23.8	60.4	52.3	77.9	52.9	70.4	32.5	32.5	39.0	90.1
Pipeline Models (ELMo)													
Pipeline (doc $\rightarrow$ $para$)	78.8	54.6	84.9	22.2	59.0	52.3	77.6	52.6	70.0	32.0	32.0	39.2	89.9
Pipeline (para $\rightarrow$ doc)	79.8	55.6	86.1	23.2	60.2	52.8	76.8	51.3	68.6	30.8	30.8	38.5	87.8

Table 4: Intrinsic Evaluation Results on the Development Set using the whole Cross-validation Set for Training.

5.4 Qualitative Analysis

To better understand the strengths and weaknesses of the joint model, we analyze the news structure and news element tags prediction made by our single model and joint model (both using ELMo embeddings) on the development set. Among the 82 documents, we find that the joint model clearly made less inconsistent predictions than the single model (18 vs. 27) where the predicted news element sequence can not be compatible with the predicted news structure type, e.g., *Inverted Pyramid* structure with *Image Lede* news element. This result proves the effectiveness of our joint model that preserves the two-way dependencies between the predicted news structure type and news elements.

We further examine the wrong predictions generated by our best joint model. About 70% errors happen because the model failed to distinguish the first news element between *Standard Lede* and *Image Lede*, which can be improved if the model is aware of the key events (Choubey et al., 2018) in a news article. The remaining errors come from identifying the *Narration* paragraphs written in narrative style, which by itself is a challenging task.

6 Extrinsic Evaluation on Text Summarization

We expect the news genre tags predicted by our joint model to be useful for extracting news summaries because our tags (e.g., *Standard Lede* in *Inverted Pyramid*; and *Synopsis* in *Kabob*) can help locate the key event descriptions of a news story which should be the right section to select sentences for extractive summarization.

To verify our expectations, we choose a recent BERT-based framework for text summarization proposed by Liu and Lapata (2019), which used to achieve the state-of-the-art performance on the CNN/DailyMail dataset (Hermann et al., 2015). We use exactly the same experiment settings as in (Liu and Lapata, 2019) and implement our text summarization models based on their source code [8]. We leave all components of the summarization model unchanged, but add an embedding layer to the input of BERT, which encodes the paragraph-level news elements and document-level news structure tags generated by our system trained on the whole cross-validation set. Specifically, the embedding layer will encode each tag or the combination of a news structure type and a news element tag (e.g., Kabob-Image Lede) into a vector with 10 dimensions, which will be concatenated with the original BERT's word embeddings. For each input token, the added embedding layer will incorporate its news structure information (e.g., the paragraph-level tag for the paragraph where the token locates in) into the hidden token representation, and therefore influence the model.

Model	R-1	R-2	R-L
LEAD-3	40.42	17.62	36.67
SUMO (Liu et al., 2019)	41.00	18.40	37.20
UniLM (Dong et al., 2019)	43.33	20.21	40.51
Baseline (Liu and Lapata, 2019)	43.25	20.24	39.63
+ News Element tags (ours)	43.42	20.28	39.74
+ News Structure types (ours)	**43.48**	**20.30**	**39.78**

Table 5: Text Summarization Results on the CNN/DailyMail Dataset. R-1 and R-2 stand for ROUGE score using unigram and bigram overlap; R-L is the ROUGE score using longest common subsequence. LEAD-3 is a simple baseline which selects the first three sentences in a news article.

6.1 Experimental Results

Table 5 shows the text summarization results on the CNN/DailyMail dataset using the automatic evaluation package ROUGE (Lin, 2004). Incorporating

---

[8] Available at `https://github.com/nlpyang/PreSumm`

News Structure	Inverted Pyramid	Martini Glass	Kabob	Narrative
LEAD-3	40.58/17.86/36.83	40.48/17.80/36.75	40.13/17.18/36.33	40.25/17.52/36.54
Baseline (Liu and Lapata, 2019)	43.38/20.30/39.76	43.33/20.28/39.72	43.05/20.12/39.38	43.17/20.20/39.58
+ News Element tags (ours)	43.43/20.33/39.80	43.38/20.30/39.75	43.32/20.22/39.61	43.35/20.26/39.70
+ News Structure types (ours)	**43.49/20.35/39.82**	**43.47/20.33/39.80**	**43.42/20.26/39.72**	**43.44/20.28/39.74**

Table 6: Text Summarization Results divided by News Structure Genres. Each cell reports R-1/R-2/R-L scores.

the system predicted paragraph-level news element tags into the baseline (Liu and Lapata, 2019) improves the R-1, R-2 and R-L by 0.17, 0.04 and 0.11 points respectively, which is non-trivial considering the difficulties of text summarization. Adding our document-level news structure types into the summarization model further improves the performance slightly, which outperforms the baseline by 0.23 R-1, 0.06 R-2 and 0.15 R-L.

6.2 Effects on Different News Genres

To understand which type of news structure is the bottleneck for news summarization, we evaluate the ROUGE scores on each subset of the CNN/DailyMail test set divided by our predicted news structure types, and report the text summarization results in Table 6. We can see that *Kabob* structure is the most difficult genre for news summarization, which is not surprising because news documents with the *Kabob* structure will not present the key events at the beginning of the story, and therefore brings additional difficulty to locate the correct paragraphs for extracting summary. By incorporating our news structure types and news element tags into the model, all genres of news documents receive better performance for extractive summarization. Especially for the news articles with the *Kabob* structure, our news genre tags improve the ROUGE scores by 0.37, 0.14 and 0.34 points on R-1, R-2 and R-L respectively, which is the largest improvement among four types of news structures.

7 Conclusion

We have presented a joint neural network model for structure-based news genre identification that predicts both the news structure type for a document and a sequence of news element tags for its paragraphs. The joint model preserves the two-way dependencies and constraints between a type of news structure and its sequence of news elements, and consistently outperforms its variants that perform two tasks independently or in a pipeline. While being imperfect, the system predicted news structure types and news element tags have been shown effective for improving text summarization models.

For the future work, we will further improve the performance on identifying minority classes of news structures and news elements (e.g., *Narration*), by conducting semi-supervised learning. Meanwhile, we are keen to explore uses of our news genres in other applications as well, such as text quality assessment and information extraction.

Acknowledgments

We thank our anonymous reviewers for providing insightful review comments. We gratefully acknowledge support from National Science Foundation via the awards IIS-1942918. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: the views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government.

References

2014. Journalism story structure.

Regina Barzilay and Lillian Lee. 2004. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *NAACL*.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218, Lisbon, Portugal. Association for Computational Linguistics.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the most dominant event in a news article by mining event coreference relations. In *NAACL-HLT*, pages 340–345.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167.

Zeyu Dai, Himanshu Taneja, and Ruihong Huang. 2018. Fine-grained structure-based news genre categorization. In *Proceedings of the Workshop Events* *and Stories in the News 2018*, pages 61–67, Santa Fe, New Mexico, U.S.A. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, volume 1, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–698.

Marti A Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001, pages 282–289.

Ellen Lavelle. 1997. Writing style and the narrative essay. *British Journal of Educational Psychology*, 67(4):475–482.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Assessing the discourse factors that influence the quality of machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 283–288.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *EMNLP-IJCNLP*, pages 3721–3731.

Yang Liu, Ivan Titov, and Mirella Lapata. 2019. Single document summarization as tree induction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1745–1755, Minneapolis, Minnesota. Association for Computational Linguistics.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

Inderjeet Mani. 2012. Computational modeling of narrative. *Synthesis Lectures on Human Language Technologies*, 5(3):1–142.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

Daniel Marcu. 1997. From discourse structures to text summaries. In *Intelligent Scalable Text Summarization*.

Joscha Märkle-Huß, Stefan Feuerriegel, and Helmut Prendinger. 2017. Improving sentiment analysis with document-level semantic relationships from rhetoric discourse structures. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.

Alexander Mehler, Serge Sharoff, and Marina Santini. 2010. *Genres on the web: Computational models and empirical studies*, volume 42. Springer Science & Business Media.

Jack Myers and Don C Wukasch. 2003. *Dictionary of poetic terms*. University of North Texas Press.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, volume 31.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, volume 1, pages 2227–2237.

Horst Pottker. 2003. News and its communicative quality: The inverted pyramid—when and why did it appear? *Journalism Studies*, 4(4):501–511.

R. Prasad, N. Dinesh, Lee A., E. Miltsakaki, L. Robaldo, Joshi A., and B. Webber. 2008. The Penn Discourse Treebank 2.0. In *lrec2008*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning*, pages 147–155. Association for Computational Linguistics.

Georg Rehm. 2002. Towards automatic web genre identification: a corpus-based approach in the domain of academia by example of the academic's personal homepage. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 1143–1152. IEEE.

Marina Santini. 2007. *Automatic identification of genre in web pages*. Ph.D. thesis, University of Brighton.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2010. Incorporating Content Structure into Text Analysis Applications.

Chip Scanlan. 2003. Writing from the top down: Pros and cons of the inverted pyramid.

Christina Schokkenbroek. 1999. News stories: structure, time and evaluation. *Time & Society*, 8(1):59–98.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Weiyan Shi, Tiancheng Zhao, and Zhou Yu. 2019. Unsupervised dialog structure learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1797–1807, Minneapolis, Minnesota. Association for Computational Linguistics.

Teun A Van Dijk. 1985. Structures of news in the press. *Discourse and communication: New approaches to the analysis of mass media discourse and communication*, 10:69.

Espen Ytreberg. 2001. Moving out of the inverted pyramid: narratives and descriptions in television news. *Journalism Studies*, 2(3):357–371.

Representing Syntax and Composition with Geometric Transformations

Lorenzo Bertolini   Julie Weeds   David Weir   Qiwei Peng
University of Sussex
Brighton, UK
`{l.bertolini, j.e.weeds, d.j.weir, qiwei.peng}@sussex.ac.uk`

Abstract

The exploitation of syntactic graphs (SyGs) as a word's context has been shown to be beneficial for distributional semantic models (DSMs), both at the level of individual word representations and in deriving phrasal representations via composition. However, notwithstanding the potential performance benefit, the syntactically-aware DSMs proposed to date have huge numbers of parameters (compared to conventional DSMs) and suffer from data sparsity. Furthermore, the encoding of the SyG links (i.e., the syntactic relations) has been largely limited to linear maps. The knowledge graphs' literature, on the other hand, has proposed light-weight models employing different geometric transformations (GTs) to encode edges in a knowledge graph (KG). Our work explores the possibility of adopting this family of models to encode SyGs. Furthermore, we investigate which GT better encodes syntactic relations, so that these representations can be used to enhance phrase-level composition via syntactic contextualisation.

1 Introduction

Representing words in terms of their syntactic co-occurrences has been long proposed, both for count-based (Padó and Lapata, 2007; Weir et al., 2016), and neural (Hermann and Blunsom, 2013; Levy and Goldberg, 2014; Komninos and Manandhar, 2016; Czarnowska et al., 2019; Vashishth et al., 2019) models of word meaning. Tested on benchmark word similarity tasks, such models often perform favourably to models based on proximal co-occurrence, particularly when the similarity or substitutability of two words is considered rather than their relatedness (Levy and Goldberg, 2014). However, the real promise of distributional models based on syntactic rather than proximal co-occurrence, is the potential for carrying out syntax-sensitive composition. For example, in the Anchored Packed Tree (APT) model (Weir et al., 2016) lexemes, phrases, and sentences are represented as collections of typed occurrences, and composition is carried out by contextualising each element in its syntactic role. This leads to syntax-sensitive representations for phrases. For example, *glass window* and *window glass* have different representations due to the different syntactic roles played by each constituent.

Alongside count-based models, a variety of neural ones have been proposed to encode syntactic structure, focusing on different depths of the graph (Levy and Goldberg, 2014; Komninos and Manandhar, 2016; Marcheggiani and Titov, 2017; Vashishth et al., 2019; Emerson, 2020)). Of particular note here, Levy and Goldberg (2014) and Komninos and Manandhar (2016) each proposed models (DEP and EXT, respectively) which learn from local dependency relations, by extending the Skip-Gram with Negative sampling (SGNS) architecture from `word2vec` (Mikolov et al., 2013). Given a tuple of *(target, context)* words, e.g. *(rain,like)*, a standard SGNS model can be trained to encode the probability of it being a true or a randomly sampled tuple. DEP and EXT, on the other hand, make use of both standard and syntactically contextualised tuples e.g., (*rain_*$\overline{dobj}$, $like$)[1]. Whilst DEP was tested solely on word similarity tasks, Komninos and Manandhar (2016) applied large neural architectures to sentence level tasks and were thus able to demonstrate a positive impact of applying an additive composition strategy to syntax-aware representations.

There is of course an explosion in the number of parameters to be learnt in both DEP and EXT due to the many possible word-relation combinations which form the target vocabulary for these models (see Table 1). A possible solution, pro-

---

[1] $\overline{dobj}$ indicating the inverse of the $dobj$ relation

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3343–3353
August 1–6, 2021.

posed by Czarnowska et al. (2019), is the Dependency Matrix (DM) model which uses linear maps in the form of square matrices to encode relations. Here, the training objective is changed from predicting *(target, context)* pairs to *(target, relation, context)* triples, e.g., *(rain,$\overline{dobj}$,like)*. This model produced comparable results with DEP and EXT at the word level. Furthermore, compositional experiments on short phrases, specifically relative clauses, produced encouraging results when using the learned transformations. Yet, despite considerably reducing the number of parameters, this model still makes use of large word spaces and the square linear map is still costly to train.

Model	Learnable Parameters
DEP	223M
DM	51.6M
MuRE	21.5M
RotE	21.5M
RefE	21.5M
AttE	21.6M

Table 1: Learnable parameters for each model, given the same word (72k) and relation (88) vocabularies from the `text8` (parsed) corpus, and vector size of n=300.

The reformulation of the SGNS objective introduced by DM (i.e., moving from *(target,context)* tuples to *(target,relation,context)* triples) closely resembles a common practice in the knowledge graphs (KGs) literature (e.g. (Trouillon et al., 2017; Balazevic et al., 2019; Chami et al., 2020)). Here, large, mainly factual, graphs are fed to neural models in the form of *(head,relation,tail)*. Compared to the syntactically-aware DSMs discussed above, many of the models proposed to encode KGs make use of a substantially lower number of parameters to encode both word and relations, as shown in Table 1. Furthermore, in order to represent the heterogeneous types of relations in KGs, researchers have experimented with models based on different types of geometric transformations (GTs). These include, but are not limited to, stretch (Balazevic et al., 2019), rotation (Sun et al., 2019; Chami et al., 2020), reflection (Chami et al., 2020) and attention (Chami et al., 2020). However, in the KG literature, limited attention has been paid to the compositional nature of phrases. Single-token oriented vocabularies (where *New York* is represented by *New_York*), used in most KGs, work well for real-world entities, such as people or cities, but are problematic when considering compositional phrases such as *small cake*. As discussed by Toutanova et al. (2015), treating these phrases in the same way forces the vocabulary to grow immensely, and prevents the model from reasoning over new phrases in a compositional fashion. Hence, developing successful composition strategies is of interest to the KG community as well as more widely in Natural Language Inference (NLI).

Given the success that DM and other models have obtained in modelling syntax and syntactically driven composition, we propose to overcome the parameter and word-relation vocabulary problems by using GT models to encode syntactic graphs. We focus our investigation on four state of the art models from the knowledge-graphs literature, namely MuRE (Balazevic et al., 2019), and the three GTs-based models proposed by Chami et al. (2020): RotE, RefE and AttE. Despite the simplicity, MuRE has obtained competitive results, when compared to more complex models (Chami et al., 2020)). Rotation has been used to model composition of relation representations (Sun et al., 2019). Attention has been frequently proposed as a plausible mechanism for composition (e.g. Hudson and Manning (2018); Tay et al. (2019); Yin et al. (2020); Russin et al. (2020)), whilst reflection is relatively under-studied (Chami et al., 2020). Furthermore, as discussed in Section 3, these models allow for an interesting comparison, as they can be grouped into three categories: tail modifiers (DM), head modifiers (RotE, RefE, AttE), and full modifiers (MuRE). Hence, we explore some of the transformational properties required to enable the successful encoding of syntactic relations, where success is defined in terms of their potential to support phrasal composition.

Our contributions are as follows. First, we show how lighter-weight models based on GTs can be used to encode both word and syntactic relations, frequently outperforming DM both in word similarity and compositional benchmarks. Second, for each model, we propose a tailored composition strategy, based on syntactic contextualisation of one (or more) of the phrase constituents. We hence show how to exploit the learned syntactic representations for composition, by comparing syntax-driven strategies for composition with simple addition. Third, we provide an analysis of which type of GTs better encode relations for syntactic contextualisation and enhanced composition.

2 Related Work

Knowledge graphs are complex data structures where nodes are concepts or entities (usually content words like *dog* or *Campari*) and edges are relations (e.g. `is_a`, `produced_in`) connecting entities to one another (e.g. *dog* `is_a` *mammal*, *Campari* `produced_in` *Italy*). Table 2 reports the number of distinct entities, relations and triples for three of the most investigated KGs, namely, FB15k-237 (Toutanova and Chen, 2015) YAGO3-10 (Mahdisoltani et al., 2015), and WN18RR (Dettmers et al., 2018), as well as a syntactic graph (SyG) constructed from the parsed corpus `text8`. The way these graphs are structured can vary significantly. Chami et al. (2020) showed how, among the presented KGs, only WN18RR has a significantly hierarchical structure.

Dataset	entities	relations	triples	graph type
WNRR18	31k	11	87k	KG
FB15k-237	15k	237	272k	KG
YAGO3-10	123k	33	1M	KG
`text8`	72k	88	12M*	SyG

Table 2: Statistics for the training splits of different datasets (* number of unique items, with observed repetitions, items raise to 18M).

Research on models for representing KGs has mainly focused on the ability to predict new connections between existing nodes. To overcome the problem of testing items that do not occur in the training set, many models have adopted negative sampling (NS) strategies in the training phase. The vocabulary of KG datasets is also largely single-token oriented. Models able to handle multi-token items have been proposed (Toutanova et al., 2015, 2016; Sun et al., 2019), but they focus on the composition of relations rather than entities, e.g., how a complex relation such as `married_to:son_of` might be split into multiple constituents and composed. Also relevant, Toutanova and Chen (2015) showed how syntax-augmented triples extracted from documents (e.g. $(Obama, \texttt{nsubj:born\_in:obj}, USA)$) can be beneficial for KGs models, but did not investigate representing syntax or composition via embeddings.

Previous works (e.g. (Marcheggiani and Titov, 2017; Vashishth et al., 2019)) showed how SyGs could be encoded via graph convolutional networks (GCN) (Kipf and Welling, 2017). These large models are able to encode larger graphs (up to the sentence level), via sequences of convolutions along the edges of the graph. Such convolutions are frequently relation-specific and are also encoded via square matrices.

3 Theoretical Approach

In both the semantic (KG) and syntactic (SyG) domain, the starting point is typically a dataset D of positive triples (h, r, t), with $h, t \in V = \{1, .., |V|\}$ and $r \in R = \{1, .., |R|\}$, where V and R are the sets of the indexes for the vocabulary of entities / words and relations, respectively. In both domains, the shared goals are: i) map entities $v \in V$ to embeddings e_v where $e \in \mathbb{R}^{|\mathsf{V}| \times n}$, n being the dimensionality of the vectors; ii) map relations $r \in R$ in one – or more – space $\mathbb{R}^{|\mathsf{R}| \times *}$. In this work, we focus on constructing a syntactic dataset of positive training triples from a corpus as in Czarnowska et al. (2019). All of the models we investigate rely on a negative sampling mechanism that generates a dataset D' of false triples. Each model was presented in its own original work with a tailored way to generate D'. Unless otherwise stated, we make use of the original mechanism.

As already discussed, we are interested in both word level and compositional level evaluation. Testing at the word level, e.g., using word similarity benchmarks, simply requires extraction of the word embeddings. Compositional tests, on the other hand, also require syntactic analysis of the phrase and extraction and application of the relation embeddings. The first step, is to generate a parsed version of the phrase. For example, syntactic analysis of the phrase $pour\ tea$ will produce the root-as-head (Rh) (h, r, t) triple $(pour, dobj, tea)$, and the root-as-tail (Rt) (h, r, t) triple $(tea, \overline{dobj}, pour)$. Such duplicity of representations was handled in DM by obtaining both representations and then summing the cosine similarities obtained when comparing each of the two representations with a given target. Whilst reasonably effective in the DM evaluation, this does not provide a single phrase-level representation and would become unwieldy for longer phrases and sentences. Weir et al. (2016) argued in favour of considering the syntactic root as the main element of any multi-token linguistic item. In our example, to compare *pour tea* with *drink water*, this would require us to consider the syntactic root in the context of its dependent i.e., how similar is the verb *pour* when contextualised by the direct object *tea* to the verb *drink* when contextualised by the `direct_object` *water*? In models which modify the head of the triple (e.g., (Chami et al., 2020), this would correspond to using the root-as-tail (Rt) analysis of the phrase. Here, we compare the two strategies empirically. Further, inspired by the growing success of (very large) bi-directional models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) and also by recent evidence from the neuroscientific literature (Mollica et al., 2020; Fedorenko et al., 2020), suggesting that sentence processing strongly relies on identifying and composing smaller units of meaning, such as phrases, regardless of order of their constituents, we also propose a third compositional strategy which is bi-directional in nature. Here, the phrase-level representation is the sum of the root-as-head and the root-as-tail representations, making it more agnostic to the direction of the relation as well as the word order. However, phrases with different structures such as *glass window* and *window glass* will still have different representations due to the different roles played by each word in each relation.

In summary, we propose and investigate three different syntax-aware (*syn*) composition strategies: *syn*-Rh and *syn*-Rt, different solely in where the root is placed in the $(head, relation, tail)$ triple; and *syn*-BiD (for bi-directional), constructed by adding the representations obtained by *syn*-Rh and *syn*-Rt. We now describe in detail the models investigated, together with our tailored *syn* composition strategy for each of them.

DM This model is an extension of SGNS, where a linear map, in the form of a $n \times n$ matrix, projects a word from the context space (e') into the target space (e), as in Equation 1:

$$u = e_h^T \cdot (W_r e_t^{'}) \tag{1}$$

where $e, e' \in \mathbb{R}^{|\mathrm{V}| \times n}$, and $W \in \mathbb{R}^{|\mathrm{R}| \times n \times n}$. Since the tail word is projected into the space occupied by the head word, we refer to this model as a tail-modifier. u is then used to compute standard SGNS loss (Equation 2):

$$\sum_{(h,r,t)\,\in\, D} \log \sigma(u) \;+ \sum_{(h,r,t)\,\in\, D'} \log \sigma(-u) \tag{2}$$

Phrase representations will be constructed following our three syntactic composition strategies. As a baseline, common to all models, we use addition (*add*) of the queried head and tail entities embeddings, as in Equation 3 [2]:

$$e_{add} = e_h + e_t \tag{3}$$

We propose *syn* composition for the DM model to be obtained via u (Equation 1), as in Equation 4:

$$e_{syn} = e_h + (W_r e_t^{'}) \tag{4}$$

MuRE This architecture falls into the family of translation models (Chami et al., 2020). Here, both the entities go through a transformation and so we refer to this model as a full-modifier. The tail entity is shifted with a translation (i.e. offset), and a stretch, in the form of a $n \times n$ diagonal matrix, is applied to the head entity. Embeddings are then fed to a distance function $d(x, y) = \|x - y\|$ and the model minimises the Bernoulli negative log-likelihood loss, using Equation 5, to estimate the probability of the triple being from D:

$$p(h, r, l) = \sigma(-d(W_r e_h, e_t + w_r)^2 + b_h + b_t) \tag{5}$$

Here, $W \in \mathbb{R}^{|\mathrm{R}| \times n \times n}$ contains $|R|$ diagonal matrices (each corresponding to a relation-specific stretch), $w \in \mathbb{R}^{|\mathrm{R}| \times n}$ hosts $|R|$ translation vectors, and $b \in \mathbb{R}^{|\mathrm{V}| \times n}$ the entity biases. Again, additive composition is carried out by adding the queried embedding for the phrase's constituents. Syntactic composition is implemented by adapting the model's score function (Equation 6):

$$e_{syn} = W_r e_h + (e_t + w_r) \tag{6}$$

RotE, RefE These models optimise a full cross-entropy loss. Like MuRE, square distance between two vectors is used as a score function. Unlike the previous model, they apply a Givens rotation (Rot) or reflection (Ref), as defined in Chami et al. (2020), and a translation to the head entity. Thus, we refer to these models as head-modifiers. Syntactic composition is defined via the score functions in Equations 7 and 8:

$$e_{syn} = (\mathrm{Rot}(T_r) e_h + t_r) + e_t \tag{7}$$

$$e_{syn} = (\mathrm{Ref}(F_r) e_h + f_r) + e_t \tag{8}$$

where $T, F \in \mathbb{R}^{|\mathrm{R}| \times \frac{n}{2}}$ each contain $|R|$ diagonal matrices (each corresponding to a relation-specific Givens rotation or reflection), and $t, f \in \mathbb{R}^{|\mathrm{R}| \times n}$ are relation-specific translations.

---

[2] This corresponds to simple-sum composition in the original work by Czarnowska et al. (2019).

AttE Intuitively, AttE is designed to model the contribution of different GTs (in this case just rotation and reflection). This is achieved via a self-attention mechanism. Given two embeddings x, y, and an attention vector a, attention scores are computed via Equation 9:

$$(\alpha_x, \alpha_y) = \text{Softmax}(a^T x, a^T y) \tag{9}$$

These scores are then averaged (Equation 10):

$$\text{Att}(x, y; a) = (\alpha_x x + \alpha_y y) \tag{10}$$

To actively select the most suitable transformation for a given triple, rotation and reflection are applied to the head-entity embedding (Equation 11):

$$\mathrm{q}_{\text{Rot}} = \text{Rot}(T_r)e_h,\ \mathrm{q}_{\text{Ref}} = \text{Ref}(F_r)e_h \tag{11}$$

The two representations are than combined using a self attention mechanism (Equation 12):

$$Q(h, r) = \text{Att}(\mathrm{q}_{\text{Rot}}, \mathrm{q}_{\text{Ref}}; a_r) + p_r \tag{12}$$

with $p \in \mathbb{R}^{|\mathrm{R}| \times n}$ as the relation-specific translation. Q and the e_t are then used as arguments for d as in Equation 5. Syntactically contextualised composition (*syn*) for AttE is implemented via Equation 13:

$$e_{syn} = Q(h, r) + e_t \tag{13}$$

4 Experiments

Our main aim is to investigate the potential of models in terms of constructing high quality word representations and their support for composition. To this end, experiments were carried out with a set of models trained on KGs, and a second set of models trained on SyGs. This allows us to investigate the value of encoding distributional information from SyGs or whether KGs alone might be a sufficient source of data to obtain competitive results. We hypothesise that when using KGs alone: i) word similarity tasks might yield high results; ii) compositional evaluation will yield poor results. As for models trained on SyG, we expect to see: i) a generally improved performance on most tasks, when compared to models trained on KGs; ii) larger models to be penalised across benchmarks and for syntactically-contextualised (*syn*) composition.

4.1 Experimental setup

Benchmarks We divide our quantitative experiments between word similarity and composition tasks. For the word similarity tasks, we focus on SimLex (Hill et al., 2015), MEN (Bruni et al., 2014), and both similarity (WS_s) and relatedness (WS_r) split of the WordSim353 (Finkelstein et al., 2001) datasets. For every word pair, we produce a model's prediction using cosine similarity (CS). We compare model predictions and human judgements using Spearman's ρ.

For the compositional investigation, we focus on the Mitchell and Lapata (2010) (ML10) dataset. Items in this benchmark consist of pairs of two-token phrases (e.g. *(pour tea–drink water)*) paired with human judgements on their similarity. Phrases are composed using the four different presented strategies and the obtained representations are compared via CS. Again, CS and human ratings are compared via ρ. We selected this benchmark for two main reasons: i) the models' structures lend themselves straightforwardly to syntactically contextualised (*syn*) composition strategies for a two-token item[3]; ii) the dataset is pre-split into three syntactic-relation classes (i.e. adjective-nouns (AN), verb-objects (VO) and noun-nouns (NN)) and this division offers an opportunity for a more in-depth investigation on how different models and operations manage to embed different syntactic relations.

We trained each set of models with three random initialisation, and report the mean and standard error (SE) of the obtained ρs.

Implementation For MurE, RotE, RefE and AttE we adapt the original PyTorch code. Since an official release of the DM is not available, we implemented a PyTorch version of the model[4].

We trained the first set of GT models on the WN18RR dataset, tuning negative sampling rate (NS), optimiser and learning rate using mean reciprocal rank (MRR) on the development set[5]. Epochs were kept stable at 50 and n to 300. We focused on WN18RR as YAGO3-10 shares a minimal vocabulary with the selected word-similarity and compositional benchmarks. FB15k-237, on the other hand, has all the entities encrypted. The models obtained from this training set were then evaluated on both word-similarity and compositional tasks (see Table 3) to provide a baseline for the SyG models.

---

[3] Czarnowska et al. (2019) proposed a more complex composition strategy, specifically for relative clause phrases which we do not consider here.

[4] `https://github.com/lorenzoscottb/findings_ACL2021`

[5] using the dataset's original splits.

	Simlex	MEN	WS_s	WS_r	Adjective Nouns	Verb Objects	Noun-Noun
MuRE	.38±.01	.45±.00	.42±.01	.21±.03	.19±.03	.31±.00	.13±.01
RotE	.35±.01	.54±.00	.59±.00	.30±.02	.18±.03	.33±.00	.20±.02
RefE	.36±.01	.54±.00	.57±.00	.30±.01	.16±.04	.37±.01	.14±.02
AttE	.36±.01	.54±.00	.58±.01	.29±.00	.20±.00	.32±.00	.18±.00

Table 3: Spearman ρs' (mean ± SE) obtained on all selected benchmarks, for knowledge-graph models trained on WN18RR dataset.

A second set of models was trained on the `text8`[6] corpus, parsed with spaCy (Honnibal and Johnson, 2015). Following Czarnowska et al. (2019), minimum item count, epochs, NS, optimiser and learning rate were fine-tuned on SimLex. Hyperparameters are selected from the union of the ones proposed in (Balazevic et al., 2019; Czarnowska et al., 2019; Chami et al., 2020). All the models share the same number of dimensions, i.e., $n = 300$. For a fair comparison, all experiments for this set have been conducted on the vocabulary shared across the models. Final coverage and best hyperparamenters are reported in Appendix A.2 and A.1. All models were trained using NVIDIA Titan V GPUs.

4.2 Results

WN18RR trained models We begin our quantitative investigation evaluating models from the knowledge graph literature, trained on WN18RR, on all benchmarks. Looking at Table 3, we note that these models, compared to models trained on `text8` or similar distributional models trained on much larger corpora, achieve competitive results on the word similarity benchmarks, especially in the historically challenging SimLex dataset, despite the small vocabulary and training samples.

A possible explanation for these results lies in how entities co-occur in the training data. First of all, WN18RR has a limited vocabulary (see Table 2), and is poorly populated by adjectives. Furthermore, noun and verbs, two part of speech (POS) that frequently co-occur between each other in natural language, here mainly occur within each other (i.e. verb with verb, noun with noun). In few cases, especially for verbs, the co-occurrences are not only limited to the same POS, but interest the very same word. All models perform much worse on the relatedness split of WS-353 than the similarity split. This might be expected, for models trained on WordNet data. As predicted, the performance is generally poor for composition benchmarks. An exception seems to be the VO subset, where models achieve results that, as will be presented shortly, are competitive also for `text8`-trained models.

Word similarity Our motivation for experiments with models trained on `text8` is to understand whether models previously proposed for representing KGs are competitive with distributional models such as DM in their ability to embed word and syntactic relations. Results for word-similarity are presented in Table 4.

	Simlex	MEN	WS_s	WS_r
DM	.12±.01	.60±.01	.59±.02	.51±.03
MuRE	.17±.01	.64±.00	.69±.01	.58±.00
RotE	.17±.00	.64±.00	.70±.00	.58±.01
RefE	.18±.01	.63±.01	.70±.01	.56±.00
AttE	.16±.00	.61±.00	.69±.01	.57±.01

Table 4: Spearman ρs' (mean ± SE) obtained on word-word similarity benchmarks, with models trained on `text8` corpus.

First, scores on SimLex are much lower than: i) those achieved by the KG-trained models; ii) those presented elsewhere for DM in the literature (Czarnowska et al., 2019). We note that the corpus we used to train the models is significantly smaller than the one used to train DM by the original authors, and we assume that this, combined with the low frequency of SimLex items in our corpus, is the main reason for these differences. Results for DM on the other word similarity benchmarks are much closer to the performance achieved by the original authors and, on these benchmarks, DM clearly outperforms the baseline of models trained on WN18RR. However, most notably, GT models trained on the same data as DM, not only achieve comparable results to DM, but they almost always outperform it, both in similarity-based and relatedness-based benchmarks. Moreover, DM seems to show the highest variation, especially for WN_s and WN_r.

Composition Table 5 shows the results for all `text8`-trained models on the compositional

---

[6] http://mattmahoney.net/dc/textdata

		Adjective Nouns	Verb Objects	Noun-Nouns	Average
DM	*add*	.39±.02	.31±.03	.43±.03	.37±.02
	syn-Rh	.26±.02	.18±.02	.25±.03	.23±.02
	syn-Rt	.32±.03	.14±.02	.20±.02	.22±.02
	syn-BiD	.33±.02	.14±.02	.34±.03	.27±.02
MuRE	*add*	.47±.01	.35±.01	.40±.00	.41±.00
	syn-Rh	**.51±.01**	.34±.01	.44±.01	.43±.00
	syn-Rt	.49±.01	.36±.01	.43±.01	.43±.01
	syn-BiD	.49±.01	.36±.01	**.46±.01**	.44±.00
RotE	*add*	.49±.00	.37±.00	.43±.00	.43±.00
	syn-Rh	.48±.01	.36±.01	.41±.01	.42±.01
	syn-Rt	.47±.02	.35±.01	.41±.01	.41±.00
	syn-BiD	.49±.00	.38±.00	.45±.01	.44±.00
RefE	*add*	.48±.01	.36±.00	.43±.01	.42±.00
	syn-Rh	.49±.01	.36±.01	.43±.01	.43±.01
	syn-Rt	.48±.01	.34±.02	.43±.01	.42±.01
	syn-BiD	.48±.00	**.38±.01**	**.46±.01**	**.44±.01**
AttE	*add*	.46±.01	.35±.01	.41±.01	.41±.00
	syn-Rh	.47±.01	.35±.00	.43±.01	.41±.01
	syn-Rt	.45±.01	.29±.01	.43±.01	.39±.00
	syn-BiD	.48±.02	.36±.00	.46±.00	.43±.00

Table 5: Spearman ρs' (mean ± SE) obtained on Mitchell and Lapata (2010) benchmark, with models trained on `text8` corpus. Phrasal composition is carried out by element-wise addition (*add*), and the three proposed syntax (*syn*) aware strategies: root as head (*syn*-Rh), root as tail (*syn*-Rt) and bidirectional (*syn*-BiD). **Best results for each Phrase Type**.

benchmark. Again, GT models show competitive results, and generally outperform DM, which fails at improving its performance with *syn* composition. This last evidence is reversed in all other models. That is, they all achieve best performance with one of the syntax-aware composition methods. Looking closer, we can see that, in most cases, the best *syn* method is the bi-directional one, with the exceptions of MUuRE, RotE and RefE's AN phrases. Notably, *syn*-BiD is almost never a mere average of the two representations that originated it. In many cases, and especially for AttE, *syn*-BiD representations produce a significantly larger gain in performance, when compared to both *syn*-Rt and *syn*-Rh. From the single model perspective, the best performing one is RefE. Syntax-aware methods based on reflection always outperform the additive baseline, and also obtained the best score in the average sections, via bi-directional composition. Again, DM is the model showing the highest variation in results. This provides further evidences in favour of the lightweight models taken from the KG literature

4.3 Statistical Analysis

All correlations were tested for significance, adopting the Holm correction (Holm, 1979) to account for the large number of tests, and we observed no $p < .05$. As the main interest of our work was the compositional investigation (reported in Table 5), a global comparison was conducted to test whether observed differences in correlations were also significant. We adopted a paired two-tail bootstrap analysis (Berg-Kirkpatrick et al., 2012; Søgaard et al., 2014; Dror et al., 2018), performed independently between results from the three seeds. Given the large number of comparisons, a Holm correction was adopted within the same Phrase Type. Results (see A.3 for more details) showed that, among all models, the only one that generated a number of insignificant differences was DM, mainly pertaining to different strategies for composing NN items.

4.4 Qualitative Analysis

We now investigate the impact of relation representations on word vectors and composition from a qualitative point of view. Here, we focus on the model that quantitative tests indicated as the most promising one: RefE. We will start at the word level, looking at syntactically contextualised single words. The interest here, is to see if clear relation-driven clusters can be identified within a reduced space. To do so, we contextualise the set of roots from ML10 (e.g. *amount* in *vast amount*), and reduce the dimensions through PCA. Results in Figure 2 suggest that the three syntactic relations adopted for contextualisation (i.e. `amod`, `dobj`,

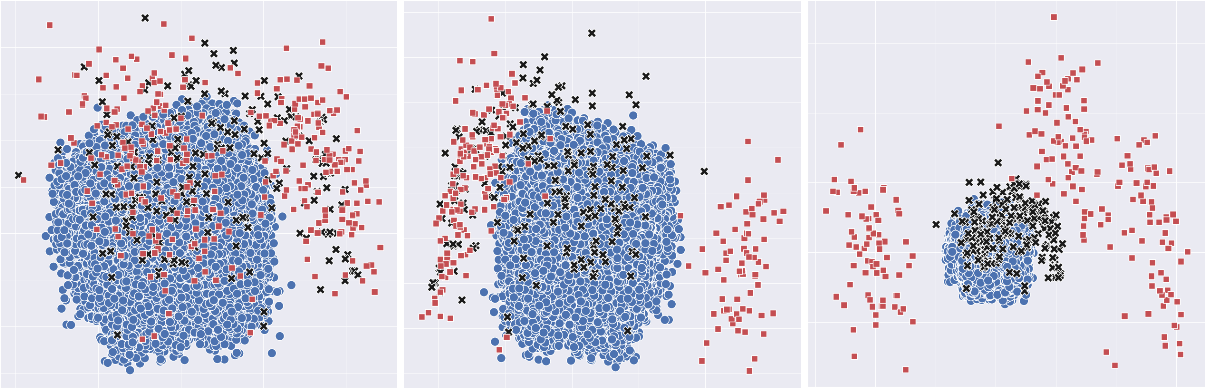

(a) *syn*-Rt

(b) *syn*-Rh

(c) *syn*-BiD

Figure 1: PCA visualisation of RefE vector space. Images show the same word (●) and *add*-composed vectors (×), in the context of representations composed with the four different syntax-aware (🟥) composition methods. All composed vectors represent the set of phrases from the Mitchell and Lapata (2010) benchmark.

`nmod`) appear to generate as many distinguishable clusters. Despite being limited, these results support evidence for syntactic subspace probed out of mBert (Chi et al., 2020).

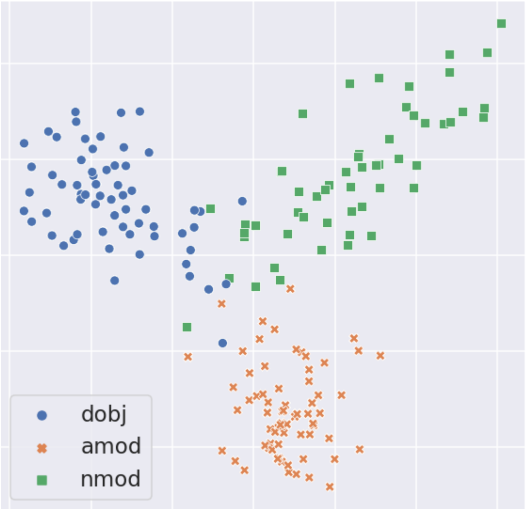

Figure 2: PCA visualisation of syntactically contextualised root-items from ML10 phrases using RefE and reflection. AN roots are contextualised using `amod`(×), VO via `dobj` (●), NN via `nmod` (🟩).

Concluding, we explore how composition strategies behave with respect to the word representations. To do so, we concatenate representations obtained by *add*-composing the set of ML10 items with the full original space, and each syntax-aware strategy separately. The three obtained sets of concatenation (i.e. word–*add*–*syn*-Rt; word–*add*–*syn*-Rh; word–*add*–*syn*-BiD) is then independently reduced to n=2 through principal component analysis (PCA). Results are reported in Figure 1. As it can be observed throughout the three reductions, and mostly in Figure 1c, phrase representations obtained via simple addition mainly lie within the perimeter of the word space. A similar pattern is observed in Figure 1a, with *syn*-Rt. Phrases composed by using the root as the head of the triple are still fairly close to the word-space perimeter, but tend to abandon its centre. Lastly, Figure 1c shows how bi-directional representations lie scattered fairly distant from the word and *add*-composed representations. This last observation is contrary to theories suggesting that representations at every level (word, phrase, sentence, etc..) should lie within the same space (e.g. Weir et al. (2016)). However, it may support recent work from neuroscience (e.g. Ding et al. (2016)) suggesting that the brain networks processing word, phrases and sentences do not completely overlap.

5 Discussion

Our results strongly suggest that light-weight models presented in the knowledge-graphs literature can be efficiently applied to syntactic-graphs, and be converted to distributional models that are consistently able to make use of the learned word and relation representations to improve semantic phrase-composition. From the model-theoretical point of view, evidence suggests that constraining linear maps with a reflection (together with a non-linear translation) seems to be the most efficient way of encoding syntactic relations. Our quantitative results also contribute to the debates on how sequential language data, or English at least, should be processed and what the role of syntactic information should be. As mentioned in

Section 3, the models selected distinguish between being tail (DM), head (RotE, RefE and AttE) and full (MuRE) modifiers. Further, we can change the syntactic focus of any of these models by adopting the *syn*-Rt composition strategy instead of the *syn*-Rh strategy. However, in our experiments, the head-modifier models (RotE, RefE and AttE) outperformed the tail-modifier and full models (DM and MuRE) and achieved a better results with the *syn*-Rh strategy than the *syn*-Rt strategy, i.e., when the syntactic root of the phrase was taken as the head of the triple rather than as the tail. In other words, it appears better to contextualise the root and compose with its dependent, which opposes the linguistic arguments put forward by Weir et al. (2016). However, even more notably, the *syn*-BiD composition strategy, which combines the *syn*-Rh and *syn*-Rt representations, generally gave a further boost to performance. This is further evidence that bi-directional information is more informative than uni-directional information, not just in large neural models such as LSTMs and transformers, and supports recent theory from neuroscience which argues that what is crucial for composition is not the overall structure nor the root, but that we can identify a phrase's constituents and the relation they have (Mollica et al., 2020). Evidence in favour of the fact that composition strongly relies on local dependencies based on syntactic structure was also found by Saphra and Lopez (2020). Such work suggests that LSTMs learn to compose following a hierarchical structure, driven by syntax, and that they rely on the learned short sequences to build longer and more reliable ones. Taken altogether, the evidence from different language-related fields is becoming more compelling that syntax and phrase composition should play an important role in the composition of larger units of meaning.

6 Conclusions and Further Work

We have shown how GT models previously proposed for encoding KGs can be adapted to encode syntactic information in a distributional model. We have demonstrated the high quality nature of the distributional word representations and the potential for using syntactically-contextualised composition strategies for phrases. In particular, we have demonstrated the competitiveness of lighter-weight GT models when compared to more general models based solely on unconstrained linear maps, such as DM. Further, our analysis has shown how learned representations for syntactic relations can be efficiently exploited at the word level, transforming a word through part-of-speech related regions of the space, and at the phrase level, generating superior composed representations. Furthermore, we have shown, among the different GTs, reflection seems to be the most promising for encoding syntactic relations. Future work will focus on composition on larger scale, syntactic-relation composition, and whether syntactic and semantic graph can be simultaneously embedded using this framework.

Acknowledgements

This research was supported by EPSRC grant no. 2129720: *Composition and Entailment in Distributed Word Representations*. We also thank the anonymous reviewers for their helpful comments, and NVIDIA for the donation of the GPU that supported our work. The first author would also like to thank Gabriele Paveri, for the years of conversations on human language, and constantly doubting the author's ideas.

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4463–4473. Curran Associates, Inc.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics.

Paula Czarnowska, Guy Emerson, and Ann Copestake. 2019. Words are vectors, dependencies are matrices: Learning word embeddings from dependency graphs. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 91–102, Gothenburg, Sweden. Association for Computational Linguistics.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Jacob Devlin, Ming-Wei. Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv e-prints*.

N. Ding, L. Melloni, H. Zhang, Xing Tian, and D. Poeppel. 2016. Cortical tracking of hierarchical linguistic structures in connected speech. *Nature neuroscience*, 19:158 – 164.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Guy Emerson. 2020. Autoencoding pixies: Amortised variational inference with graph convolutions for functional distributional semantics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3982–3995, Online. Association for Computational Linguistics.

Evelina Fedorenko, Idan Asher Blank, Matthew Siegelman, and Zachary Mineroff. 2020. Lack of selectivity for syntax relative to word meanings throughout the language network. *Cognition*, 203:104348.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414, New York, NY, USA. ACM.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

S. Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Drew Arad Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500, San Diego, California. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

F. Mahdisoltani, J. Biega, and Fabian M. Suchanek. 2015. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Francis Mollica, Matthew Siegelman, Evgeniia Diachek, Steven T. Piantadosi, Zachary Mineroff, Richard Futrell, Hope Kean, Peng Qian, and Evelina Fedorenko. 2020. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.

Naomi Saphra and Adam Lopez. 2020. LSTMs compose—and Learn—Bottom-up. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2797–2809, Online. Association for Computational Linguistics.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hector Martínez Alonso. 2014. What's in a p-value in NLP? In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–10, Ann Arbor, Michigan. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Yi Tay, Anh Tuan Luu, Aston Zhang, Shuohang Wang, and Siu Cheung Hui. 2019. Compositional de-attention networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6135–6145. Curran Associates, Inc.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1434–1444, Berlin, Germany. Association for Computational Linguistics.

Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *J. Mach. Learn. Res.*, 18:130:1–130:38.

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. 2019. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3308–3318, Florence, Italy. Association for Computational Linguistics.

David Weir, Julie Weeds, Jeremy Reffin, and Thomas Kober. 2016. Aligning packed dependency trees: a theory of composition for distributional semantics. *Computational Linguistics, special issue on Formal Distributional Semantics*, 42(4):727–761.

Da Yin, Tao Meng, and Kai-Wei Chang. 2020. SentiBERT: A transferable transformer-based architecture for compositional sentiment semantics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3695–3706, Online. Association for Computational Linguistics.

Alexey Zobnin and Evgenia Elistratova. 2019. Learning word embeddings without context vectors. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 244–249, Florence, Italy. Association for Computational Linguistics.

Figurative Language in Recognizing Textual Entailment

Tuhin Chakrabarty[*1], Debanjan Ghosh[*] [3], Adam Poliak[2,4] and Smaranda Muresan[1,4]
[1]Department of Computer Science, Columbia University
[2]Department of Computer Science, Barnard College
[3]Educational Testing Service, [4]Data Science Institute, Columbia University
`{tuhin.chakr, smara}@cs.columbia.edu,`
`dghosh@ets.org, apoliak@barnard.edu`

Abstract

We introduce a collection of recognizing textual entailment (RTE) datasets focused on figurative language. We leverage five existing datasets annotated for a variety of figurative language – simile, metaphor, and irony – and frame them into over 12,500 RTE examples.We evaluate how well state-of-the-art models trained on popular RTE datasets capture different aspects of figurative language. Our results and analyses indicate that these models might not sufficiently capture figurative language, struggling to perform pragmatic inference and reasoning about world knowledge. Ultimately, our datasets provide a challenging testbed for evaluating RTE models.

1 Introduction

Figurative language is ubiquitous in many forms of discourse from novels, poems, and films, to scientific literature and social media conversations (Ghosh, 2018). It is often used to convey intimacy (Gerrig and Gibbs Jr, 1988), humour (Roberts and Kreuz, 1994), intense emotions (Fussell and Moss, 1998), or veiled politeness (Jorgensen, 1996). Despite its ubiquity, figurative language remains "a bottleneck in automatic text understanding" (Shutova, 2011).

Recognizing Textual Entailment (RTE), the task of identifying whether one sentence (context) likely entails another (hypothesis), is often used as a proxy to evaluate how well Natural Language Processing (NLP) systems understand natural language (Cooper et al., 1996; Dagan et al., 2006; Bowman et al., 2015). Figurative language is defined as any figure of speech which depends on a non-literal meaning of some or all of the words used. Thus, understanding figurative language can be framed as an RTE task (figurative language expression vs. intended meaning), where the figurative language expression is the *context* and the intended meaning is the *hypothesis* in an RTE framework (See examples in Table 1).

Simile	▶ I start to prowl across the room like a **tightrope walker on dental floss**. I start to prowl across the room *recklessly*.	✗
	▶ They had shut him in a basement that looked like a **freight elevator**. They had shut him in a basement that looked *dangerously claustrophobic*.	✓
Metaphor	▶ He **weathered** the costs for the accident. He *avoided* the costs for the accident.	✗
	▶ The bus **bolted** down the road. The bus *paced* down the road.	✓
Irony	▶ Made $174 this month, gonna buy a yacht! I don't make much money.	✗
	▶ Fans seem restless, gee, don't understand them. Fans seem restless - don't know the reason behind it.	✓

Table 1: Example RTE pairs focused on similes, metaphors, and irony that RoBERTa *incorrectly* labels. ▶ indicates a context and the following sentence is its corresponding hypothesis. ✓ and ✗ respectively indicate that the context entails, or does not entail the hypothesis. **Bold** text represent simile and metaphors and *Italic* represent their entail/not entail interpretations (top two rows).

We investigate how suitable are state-of-the-art RTE models trained on current RTE datasets to capture figurative language. We focus on three specific types of figurative language: similes, metaphors, and irony. Similes evoke comparisons between two seemingly different objects, metaphors expand the imagination beyond the literal narrative, and irony conveys the opposite of what is said.

We leverage five existing datasets annotated for these types of figurative language to create over

---

[*] Equal Contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3354–3361
August 1–6, 2021. ©2021 Association for Computational Linguistics

12,500 RTE examples that require understanding or identifying these phenomena. We evaluate how well standard neural RTE models capture these aspects of figurative language. Our results demonstrate that, although, systems trained on a popular RTE dataset may capture some aspects of various types of figurative language, they fail on cases where the interpretation relies on pragmatic inference and reasoning about world knowledge. We release the code and the data. [1]

2 Related Work

We follow recent work that test for an expanded range of inference patterns in RTE systems (Bernardy and Chatzikyriakidis, 2019) by evaluating how well RTE models capture specific linguistic phenomena, such as pragmatic inferences (Jeretic et al., 2020), veridicality (Ross and Pavlick, 2019), and others (Pavlick and Callison-Burch, 2016; White et al., 2017; Dasgupta et al., 2018; Naik et al., 2018; Glockner et al., 2018; Kim et al., 2019; Kober et al., 2019; Richardson et al., 2020; Yanaka et al., 2020; Vashishtha et al., 2020; Poliak, 2020).

We are not the first to explore figurative language in RTE. Agerri (2008) analyze examples in the Pascal RTE-1 (Dagan et al., 2006) and RTE-2 (Bar-Haim et al., 2006) datasets that require understanding metaphors and Agerri et al. (2008) present an approach for RTE systems to process metaphors. Poliak et al. (2018)'s diverse collection of RTE datasets includes examples based on figurative language, but focuses only on identifying puns.

3 Dataset Creation

We create RTE test sets that focus on similes, metaphors, and irony. We provide further background for these types of figurative language and describe the methods used for creating these test sets. Table 2 reports the final test sets' statistics.

3.1 Simile

Comparisons are inherent linguistic devices that express the likeness of two entities, concepts, or ideas. When used figuratively, comparisons are called similes. Similes are used to spark the reader's imagination by making descriptions more emphatic or vivid (Paul et al., 1970). Similes use a common PROPERTY to compare two concepts often referred to as the TOPIC (the logical subject) and the VEHICLE (the logical object of comparison). For example, in the simile "Love is like an unicorn", love (TOPIC) is compared to a unicorn (VEHICLE), portraying the implicit property "rare". Recently Chakrabarty et al. (2020) released a test set of 150 literal sentences from subreddits r/WritingPrompts and r/Funny, each aligned with two human-written paraphrases with similes that retain the original meaning.

Data		Total	E	NE
Simile		600	300	300
Metaphor		613	307	306
Irony Meaning	$SIGN_{2000}$	2,000	133	1867
	S_{im}-H_{int}	4,762	-	4,762
Irony Intention		4,601	2,212	2,389

Table 2: Dataset statistics and class distribution, *Entailment* (E) and *Not-Entailment* (NE) for each type of figurative language.

To create our RTE test set that focuses on similes, we treat these simile-literal aligned sentences as entailed context-hypothesis pairs. Given a literal input, "They had shut him in a basement that looked **dangerously claustrophobic**", an expert annotator re-framed it as "They had shut him in a basement that looked *like a freight elevator*".[2] We create Not-Entailed examples by flipping the literal verb/property with their respective antonyms and use the original (Literal, Simile) pairs as Entailed. For instance, in the case of an existing context-hypothesis pair expressing *Entailment* - "Hitler skittered off like **an enthusiastic sloth**" → "Hitler skittered off *slowly*" - we alter "slowly" to "fast" to make it a pair of *Not-Entailment* (NE) instance.

3.2 Metaphor

Metaphors express deep feelings and complex attitudes (Veale et al., 2016). Understanding metaphors requires comprehending abstract concepts and making connections between seemingly unrelated ideas to appropriately deviate from literal meaning (Gutierrez et al., 2016; Mohammad et al., 2016; Kintsch and Bowles, 2002; Glucksberg, 1998).When generating metaphoric paraphrases, Chakrabarty et al. (2021) create a diverse test set of 150 literal sentences curated from different domains and genres and asked two expert annotators to create metaphorical sentences, resulting in a total

---

[1] `https://github.com/tuhinjubcse/Figurative-NLI`

[2] Note, such re-framing task (content generation task) does not involve assigning a label to a text fragment, thus, computing inter-annotator agreement is not applicable here.

Genre	PairID	Example
Slate	143311e	► Praise from a stranger is **like a glass of water** served at a restaurant in: You drink it warily, if at all, fearing it may be tainted Praise from someone you do not know can be taken lightly
Fiction	60838c	► The stars are no more like the sun than the glow of my cigarette is **like a forest fire**. The sun is comparable to the stars because they are the same.
Telephone	99298c	► But uh still I I question the ability of some of the teachers to uh really do a **bang-up job** and yet others i know are just wonderful All teachers sucks

Table 3: Examples from MNLI that include figurative language. ► indicates a context and the following line is its corresponding hypothesis.

of 300 metaphorical examples. The expert annotators re-framed the literal sentences independently by replacing the literal verb with a metaphorical verb. For instance, an expert reframed the literal sentence “The tax cut will help the economy" to “The tax cut will **fertilize** the economy".

Since the most frequent type of metaphor is expressed by verbs (Martin, 2006; Steen, 2010) these literal and metaphorical paraphrases differ only by the verb they use. In an RTE framework, we treat these metaphorical-literal pairs as entailed context-hypothesis examples. To create Not-Entailed examples, we generate hypotheses by manually swapping the literal verb in the entailed hypothesis with its antonym. Note that for both simile and metaphor, automatic substitution using available lexicons is problematic as it often leads to ungrammatical sentences. Manually replacing the words with its antonym guarantees a high quality test set. We use antonyms to create Not-Entailed examples for Simile and Metaphors which contain both Neutral and Contradiction classes. Such lexical replacement using antonyms would clearly lead to higher quality contradiction example creation. On the contrary, creating neutral examples by lexical perturbation is challenging and if not done properly, it can lead to grammatical errors or incoherent sentences.

3.3 Irony

When using irony, speakers usually mean the opposite of what they say (Sperber and Wilson, 1981; Dews et al., 2007). We develop different test sets focusing on whether the RTE models should *understand the conveyed meaning* of ironic examples or should *identify the speaker's ironic intent* (i.e., identify if an utterance is ironic or not) given the hypothesis that the speaker was ironic.

Understanding Ironic Meaning (IMeaning) Peled and Reichart (2017) used skilled annotators to create a parallel dataset between tweets with verbal irony and their non-ironic rephrasings (15K pairs). Annotators also had the option to copy the original tweet or just to paraphrase it, in case the ironic intent is not easy to identify. Likewise, Ghosh et al. (2020) released a parallel dataset of speakers' ironic messages (S_{im}) and hearers' interpretations (H_{int}) of the speaker's intended meaning. This dataset (S_{im}-H_{int}) contains 4,761 ironic-literal pairs. We use both datasets in our experiments and henceforth denote them as $SIGN$ and S_{im}-H_{int}, respectively. For both datasets, the original *ironic* messages are treated as the contexts and the *intended* meanings are the hypotheses. However, all RTE contexts do not contradict their corresponding hypotheses. For instance, in case of Peled and Reichart (2017), the authors allowed annotators to not rephrase the ironic sentences with their opposite *intended meanings* (in case the sarcastic or ironic intent was not clear). Thus, for evaluation purposes (see Table 4), we annotated a subset of 2,000 random pairs from $SIGN$ and evaluated the RTE models on that subset (denoted as $SIGN_{2000}$ henceforth). Around 93% of the RTE pairs in $SIGN_{2000}$ are Not-Entailed examples and 100% of RTE pairs in S_{im}-H_{int} are Not-Entailed examples.

Recognizing Ironic Intent (IIntent) We leverage additional ironic examples from Van Hee et al. (2018). Following Poliak et al. (2018)'s method for recasting annotations for puns and sentiment, we use *templates* to generate contexts (a) and hypotheses (b). We use all the ironic tweets ($training$ and $test$) released by Van Hee et al. (2018) to generate 4,598 RTE pairs. Akin to Poliak et al. (2018), we

Testset / Model	Simile	Metaphor	IMeaning		IIntent
			$sm - im$	$SIGN_{2000}$	
NBoW	51.17	54.81	86.37	71.50	**61.72**
InferSent	55.01	65.75	71.62	68.84	11.72
RoBERTa-large	**85.47**	**88.09**	**94.76**	**93.42**	52.81

Table 4: Accuracy of different models on our datasets focusing on similes, metaphors, and irony.

replace *Name* with names sampled from a distribution of names based on the US census data.[3]. The templates are a) *Name* tweeted that *tweet*, b) *Name* was ironic.

4 Experimental Setup

MNLI (Williams et al., 2018) is one of the widely used large-scale corpora that contains instances of figurative language (Table 3). Following recent work, we evaluate RTE models trained on MNLI (Williams et al., 2018) using three standard neural models: bag of words (NBoW) model, InferSent (Conneau et al., 2017), and RoBERTa-large (Liu et al., 2019). In NBoW, word embeddings for contexts and hypotheses are averaged separately, and their concatenation is passed to a logistic regression softmax classifier. InferSent encodes the context and hypotheses independently using a BiLSTM, then their sentence representations are fed to a MLP.[4] For RoBERTa, we use the model fine-tuned on MNLI from the Transformer's library (Wolf et al., 2020). We expect models trained on MNLI to capture some forms of figurative language that often appear in works of fictions, conversations, speeches, and magazines like Slate. Table 3 illustrates a few examples from MNLI that include figurative language

5 Results and Discussions

Table 4 reports models' accuracy on our figurative language RTE datasets. We observe that for similes, metaphors and irony meaning, RoBERTa-large drastically outperforms the other two models. For Irony datasets, NBoW outperforms InferSent. While all models perform poorly on IIntent, InferSent's very low accuracy stands out. The low performances might be due to the templatic nature of this recast dataset which might be very different from the MNLI training data.[5] We now turn to an in-depth analysis of RoBERTa's performance across these datasets.

Ironic Meaning. RoBERTa-large attains over 90% accuracy on the two datasets focused on ironic meaning. When analyzing these examples, a vast majority of the hypotheses in both datasets use lexical antonyms ("flattering" ↔ "disgusting) or negation ("is great" ↔ "is not great") to represent the intended meaning. Thus, the presence of antonyms might be enough for RoBERTa to correctly predict that the hypothesis is not-entailed by the context.

However, this does not hold true for hypotheses where the intended meanings were represented via more complex rephrasing. Ghosh et al. (2020) conducted a thorough study of the *linguistic strategies* that annotators have used for the rephrasing tasks. They presented a linguistically motivated typology of the strategies (e.g., "Lexical and phrasal antonyms", "Negation", "Weakening the intensity of sentiment", "Interrogative to Declarative Transformation", "Counterfactual Desiderative Constructions", and "Pragmatic Inference") and empirically validated the strategies over the $SIGN$ and S_{im}-H_{int} datasets.[6] During our analysis, we observe that for the vast majority of cases where RoBERTa predicts incorrectly, the examples contain Rhetorical Questions ("nice having finals on birthday?" ↔ "do not like finals ..."), pragmatic inferences ("Made $174 this month ... a yacht!" ↔ "I don't make much money"), or desiderative constructions of *[I wish] (that)* ("glad you related the news" ↔ "[I wish] that you have told me sooner". We also observe that RoBERTa-large's predictions are regularly incorrect when the ironic messages contain certain irony markers (Ghosh and Muresan, 2018), such as metaphor ("shoe smell like bed of roses" ↔ "smells bad"), alternate spelling where the speaker frequently overstate the magnitude of an ironic event ("dancing in heels is grrrrreat" ↔ "... hurts your feet") or hashtags that are composed of multi-word expressions that capture the irony ("god bless you ... #notinthemood).

---

[3] http://www.ssa.gov/oact/babynames/names.zip

[4] Both NBoW and InferSent use 300D Glove embeddings (Pennington et al., 2014).

[5] We leave further analysis of this issue for future work.

[6] https://github.com/debanjanghosh/interpreting_verbal_irony

			Gold	Pred
Simile	►	Your guardian angel is just a little too much like a **nerd at a comic convention**. Your guardian angel is just a little too *enthusiastic*	✓	✗
	►	Growing up, people always thought you were like a **social pariah**. Growing up, people always thought you were *ordinary*	✗	✓
	►	They all agree the books are good reads, but they are like **pseudo science fiction**. They all agree the books are good reads, but they are *too unrealistic*.	✓	✗
Metaphor	►	The smell of smoke **carpeted** on the delinquent. The smell of smoke *took off* on the delinquent	✗	✓
	►	As they strike the ground, they are **effaced**. As they strike the ground, they are *remembered*	✗	✓
	►	The avalanche **polvarized** anything standing in its way. The avalanche *protected* anything standing in its way.	✗	✓
Irony	►	Life was never been perfect and would never be. Life has never been perfect and would never be.	✓	✗
	►	The highlight of my day figuring out how to make contact sheets . . . such a boring life. My entire day was occupied in making contact sheets in design such a waste.	✓	✗
	►	Gotta read 70ish+ pages today #great #mysundayfunday #thisshouldbefun. I have to read 70ish+ pages today. This is bad.	✗	✓

Table 5: Examples from our Simile, Metaphor, and Irony datasets where Roberta-large fine-tuned on MNLI fails to classify the sentence pairs correctly. Gold and Pred means the true label and the predicted label respectively. ► indicates a context and the following sentence is its corresponding hypothesis. ✓ and ✗ respectively indicate that the context entails, or does not entail the hypothesis.

Simile. Likewise, for the simile dataset, we notice that RoBERTa-large often fails to reason with implicit knowledge about the physical and visual world knowledge (Table 5). This is inline with Weir et al. (2020)'s finding that transformer-based contextual language models poorly capture knowledge grounded in visual perceptions. For example, RoBERTa-large incorrectly predicts that the context "You wake one morning to find your entire family lying like **gray slabs of cement**" does not entail the hypothesis "You wake one morning to find your entire family lying *unconscious*". Nevertheless, RoBERTa-large correctly predicts that, "my eyes teared up . . . turning like a **ripening tomato**" entails "my eyes teared up . . . face *turning red*". We hypothesize that here RoBERTa-large was able to identify the association between "ripening tomato" and "red" that resulted in the correct prediction.

Metaphor. We notice RoBERTa-large makes wrong predictions when it encounters *unconventional* metaphors (Table 5). Metaphors are deemed unconventional depending on "how well-worn or how deeply entrenched a metaphor is in everyday use by ordinary people for everyday purposes" (Gelo and Mergenthaler, 2012). For instance, for a unconventional (metaphoric, literal) pair, "night sky **flurried** with the massive bombardment" → "night sky *doused* with the massive bombardment" (i.e., "flurried" ↔ "doused") the model fails. On the contrary, the model correctly predicts the following conventional (metaphoric, literal) pair - "sudden fame **kindled** her ego" → ". . . *increased* her ego" (i.e., "kindled" ↔ "increased").

6 Conclusion

To understand the figurative language inference capabilities of RTE models, we introduce datasets adapted from existing corpora focusing on similes, metaphors, and irony. By testing models trained on MNLI, we find that while the RoBERTa-large model is able to capture some aspects of figurative language, it fails when the interpretation requires word knowledge and pragmatic inferences. We hope this work will spark additional interest in the research community to incorporate and test for figurative language in their NLU systems.

7 Ethical Considerations

We leverage freely available open source datasets and software tools to create RTE datasets that involve similes, metaphors, and irony. We are granted the rights to further annotate and distribute the existing datasets as part of our RTE setup. This research is exempt from institutional review boards since we do not study human subjects and all social media data used is publicly available.

References

Rodrigo Agerri. 2008. Metaphor in textual entailment. In *Coling 2008: Companion volume: Posters*, pages 3–6, Manchester, UK. Coling 2008 Organizing Committee.

Rodrigo Agerri, John Barnden, Mark Lee, and Alan Wallington. 2008. Textual entailment as an evaluation framework for metaphor resolution: A proposal. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 357–363. College Publications.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, and Bernardo Magnini. 2006. The second pascal recognising textual entailment challenge.

Jean-Philippe Bernardy and Stergios Chatzikyriakidis. 2019. What kind of natural language inference are nlp systems learning: Is this enough?

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Tuhin Chakrabarty, Smaranda Muresan, and Nanyun Peng. 2020. Generating similes effortlessly like a pro: A style transfer approach for simile generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6455–6469, Online. Association for Computational Linguistics.

Tuhin Chakrabarty, Xurui Zhang, Smaranda Muresan, and Nanyun Peng. 2021. MERMAID: Metaphor generation with symbolism and discriminative decoding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4250–4261, Online. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

I. Dasgupta, D. Guo, A. Stuhlmüller, S. J. Gershman, and N. D. Goodman. 2018. Evaluating Compositionality in Sentence Embeddings. *ArXiv e-prints*.

Shelly Dews, Joan Kaplan, and Ellen Winner. 2007. Why not say it directly? the social functions of irony. *Irony in language and thought*, pages 297–318.

Susan R Fussell and Mallie M Moss. 1998. Figurative language in emotional communication. *Social and cognitive approaches to interpersonal communication*, pages 113–141.

Omar Carlo Gioacchino Gelo and Erhard Mergenthaler. 2012. Unconventional metaphors and emotional-cognitive regulation in a metacognitive interpersonal therapy. *Psychotherapy Research*, 22(2):159–175.

Richard J Gerrig and Raymond W Gibbs Jr. 1988. Beyond the lexicon: Creativity in language production. *Metaphor and Symbol*, 3(3):1–19.

Debanjan Ghosh. 2018. *An Empirical Study of Verbal Irony*. Ph.D. thesis, Rutgers, The State University of New Jersey.

Debanjan Ghosh and Smaranda Muresan. 2018. “with 1 follower i must be awesome :p”. exploring the role of irony markers in irony recognition. *Proceedings of ICWSM*.

Debanjan Ghosh, Elena Musi, and Smaranda Muresan. 2020. Interpreting verbal irony: Linguistic strategies and the connection to theType of semantic incongruity. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 82–93, New York, New York. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Sam Glucksberg. 1998. Understanding metaphors. *Current Directions in Psychological Science*, 7(2):39–43.

E Dario Gutierrez, Ekaterina Shutova, Tyler Marghetis, and Benjamin Bergen. 2016. Literal and metaphorical senses in compositional distributional semantic models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 183–193.

Paloma Jeretic, Alex Warstadt, Suvrat Bhooshan, and Adina Williams. 2020. Are natural language inference models IMPPRESsive? Learning IMPlicature and PRESupposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8690–8705, Online. Association for Computational Linguistics.

Julia Jorgensen. 1996. The functions of sarcastic irony in speech. *Journal of pragmatics*, 26(5):613–634.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Walter Kintsch and Anita R Bowles. 2002. Metaphor comprehension: What makes a metaphor difficult to understand? *Metaphor and symbol*, 17(4):249–262.

Thomas Kober, Sander Bijl de Vroe, and Mark Steedman. 2019. Temporal and aspectual entailment. In *Proceedings of the 13th International Conference on Computational Semantics-Long Papers*, pages 103–119.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

James H Martin. 2006. A corpus-based analysis of context effects on metaphor comprehension.

Saif Mohammad, Ekaterina Shutova, and Peter Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33, Berlin, Germany. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Anthony M Paul, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 1970. Figurative language. In *Philosophy & Rhetoric*, pages 225–248.

Ellie Pavlick and Chris Callison-Burch. 2016. Most "babies" are "little" and most "problems" are "huge": Compositional entailment in adjective-nouns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2164–2173. Association for Computational Linguistics.

Lotem Peled and Roi Reichart. 2017. Sarcasm sign: Interpreting sarcasm with sentiment based monolingual machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1690–1700, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Adam Poliak. 2020. A survey on recognizing textual entailment as an nlp evaluation. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 92–109.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 337–340, Brussels, Belgium. Association for Computational Linguistics.

Kyle Richardson, Hai Na Hu, Lawrence S. Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *AAAI*, volume abs/1909.07521.

Richard M Roberts and Roger J Kreuz. 1994. Why do people use figurative language? *Psychological science*, 5(3):159–163.

Alexis Ross and Ellie Pavlick. 2019. How well do NLI models capture verb veridicality? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2230–2240, Hong Kong, China. Association for Computational Linguistics.

Ekaterina V Shutova. 2011. Computational approaches to figurative language. Technical report, University of Cambridge, Computer Laboratory.

Dan Sperber and Deirdre Wilson. 1981. Irony and the use-mention distinction. *Philosophy*, 3:143–184.

Gerard Steen. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.

Siddharth Vashishtha, Adam Poliak, Yash Kumar Lal, Benjamin Van Durme, and Aaron Steven White. 2020. Temporal reasoning in natural language inference.

Tony Veale, Ekaterina Shutova, and Beata Beigman Klebanov. 2016. Metaphor: A computational perspective. *Synthesis Lectures on Human Language Technologies*, 9(1):1–160.

Nathaniel Weir, Adam Poliak, and Benjamin Van Durme. 2020. Probing neural language models for human tacit assumptions. In *42nd Annual Virtual Meeting of the Cognitive Science Society (CogSci)*.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 996–1005.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. 2020. Do neural models learn systematicity of monotonicity inference in natural language? In *ACL*.

To Point or Not to Point: Understanding How Abstractive Summarizers Paraphrase Text

Matt Wilber
Cornell University
`mw979@cornell.edu`

William Timkey
Cornell University
`wpt25@cornell.edu`

Marten van Schijndel
Cornell University
`mv443@cornell.edu`

Abstract

Abstractive neural summarization models have seen great improvements in recent years, as shown by ROUGE scores of the generated summaries. But despite these improved metrics, there is limited understanding of the strategies different models employ, and how those strategies relate their understanding of language. To understand this better, we run several experiments to characterize how one popular abstractive model, the pointer-generator model of See et al. (2017), uses its explicit copy/generation switch to control its level of abstraction (generation) vs extraction (copying). On an extractive-biased dataset, the model utilizes syntactic boundaries to truncate sentences that are otherwise often copied verbatim. When we modify the copy/generation switch and force the model to generate, only simple paraphrasing abilities are revealed alongside factual inaccuracies and hallucinations. On an abstractive-biased dataset, the model copies infrequently but shows similarly limited abstractive abilities. In line with previous research, these results suggest that abstractive summarization models lack the semantic understanding necessary to generate paraphrases that are both abstractive and faithful to the source document.

1 Introduction

Recent years have seen great improvements in "abstractive" summarization models – models that not only concatenate text from the source document, but can additionally paraphrase to generate summary text. Once limited to sentence compression (Rush et al., 2015), abstractive models now generate multi-sentence summaries (See et al., 2017), even for relatively long documents (Cohan et al., 2018). However, extractive models and mixed models with significant extractive components continue to show strong performance, and the extent and manner in which abstraction is used by summarization models is not well understood.

Previous work has raised concerns about whether models are able to paraphrase in ways that lead to better summaries. Abstractive models often generate summaries that are either ungrammatical or unfaithful to the source document (Maynez et al., 2020; Durmus et al., 2020; Kryscinski et al., 2020) and are prone to repetition in their outputs (See et al., 2019; Holtzman et al., 2020). These issues raise questions about *how* neural summarizers generate novel text. Abstractive summarization is differentiated from extractive summarization by the model's ability to paraphrase, but paraphrasing ability is not directly measured by popular metrics, leading to a lack of understanding of the generative process. Some previous research has aimed to alleviate these issues in evaluation: Zhang et al. (2018a) propose evaluating summaries with human evaluations of informativeness and coherence, and Ganesan (2018) implements a metric to reward models that paraphrase via simple synonym substitutions according to WordNet. However, synonym substitution is just one form of paraphrasing, and truly abstractive models should be capable of more complex paraphrasing strategies.

To understand how abstraction manifests in neural summarization models, we study a model that has an explicit abstraction/extraction switch, the pointer-generator model of See et al. (2017). The training objective of this model causes it to choose the best summarization strategy (abstractive vs extractive) in different contexts, permitting us to determine the environments where abstractive summarization is an effective summarization strategy. First, we show how the switch varies across a full summary and is influenced by the decoder's copy and generation distributions. Next, we present a behavioral probe of the abstraction/extraction switch, to observe how the switch reacts to lexical, struc-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3362–3376
August 1–6, 2021. ©2021 Association for Computational Linguistics

tural, and distributional information as it decodes a summary. Finally, we modify the switch value, forcing more frequent paraphrase generation during decoding, revealing the limits of the model's paraphrasing capabilities. Ultimately, we find across both the CNN/DailyMail and XSum datasets that the model's abstractive capabilities are limited; the model understands how to identify and combine constituents from the source text in a grammatical fashion, but lacks the semantic understanding required to produce grammatical, faithful and meaningful paraphrases.

2 Model

2.1 The Pointer-Generator Model

We study the pointer-generator model released by See et al. (2017), which uses an explicit switch, p_{gen}, that blends abstractive and extractive summarization strategies. We briefly review the pointer-generator model here; for more details, see the original paper of See et al. (2017).

The final output distribution for a particular word in the summary $P(w)$ is a weighted sum of the generation distribution and the copy distribution, weighted by p_{gen} and $1 - p_{\text{gen}}$, respectively. This is described by Equation 9 in See et al. (2017), modified for clarity here:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) P_{\text{copy}}(w) \quad (1)$$

$P_{\text{vocab}}(w)$ is the generation distribution over the model's vocabulary, and $P_{\text{copy}}(w)$ is the copy distribution over the tokens in the source document. The p_{gen} switch explicitly weights the influence of the generation and copy mechanisms on $P(w)$. For each time step t, p_{gen} is a function of the context vector h_t^*, the decoder state s_t and the decoder input x_t,

$$p_{\text{gen}} = \sigma(\delta_{h_t^*}^T h_t^* + \delta_s^T s_t + \delta_x^T x_t + \beta_{\text{ptr}}) \quad (2)$$

where σ is the sigmoid function and $\delta_{h_t^*}$, δ_s, δ_x and β_{ptr} are learned parameters.

See et al. (2017) also use a coverage mechanism aimed at reducing repetition, defining the coverage vector c^t as

$$c^t = \sum_{t'=0}^{t-1} P_{\text{copy}(w_t)} \quad (3)$$

which is passed as another input to the attention mechanism.

2.2 Data

We analyze pointer-generator behavior when trained on an extractive-biased dataset, CNN/DailyMail, and on an abstractive-biased dataset, XSum. The CNN/DailyMail dataset is made up of multi-sentence summaries of news articles from CNN and Daily Mail. XSum (Narayan et al., 2018) is a summarization dataset that uses the first sentence of a news article as a summary of the article. The dataset treats the remainder of the article as the source document. As a result, the summaries are both shorter and more difficult to copy from the source document, compared to the CNN/DailyMail dataset.

2.3 Training

Our experiments on CNN/DailyMail use the trained model released by See et al. (2017), which includes the coverage mechanism described above. We decode summaries on the test set of at most 120 tokens using beam search with beam width 4, as in the original paper. For XSum, we trained our own model on the XSum training partition, using the code released by See et al. (2017).[1]

Like Narayan et al. (2018), we do not include the coverage mechanism for the XSum model. When coverage is used for the XSum model, ROUGE scores (Lin, 2004) slightly decrease, and the produced summaries contain more severe hallucinations. However, adding coverage does "fix" some degenerate summaries that produce the same sequence of tokens repeatedly – see Appendix B for an example.

For both datasets, in addition to the output summaries, we record the value of the p_{gen} switch for each emitted token, as well as the generation distribution and the copy distribution at each time step.

3 Experiments

In Section 3.1 we qualitatively analyze the evolution of the per-token p_{gen} and uncertainty in the extractive/abstractive components over the course of randomly selected summaries. Section 3.2 provides quantitative evidence of our observations across the full test sets, by modeling the lexical, structural, and distributional (P_{vocab} and P_{copy}) environments that drive the variability of the p_{gen} switch.

Finally, in Section 3.3 we manipulate p_{gen} of the CNN/DailyMail model to generate summaries

---

[1] Code and full replication details are available at https://github.com/mwilbz/pointer-generator-analysis.

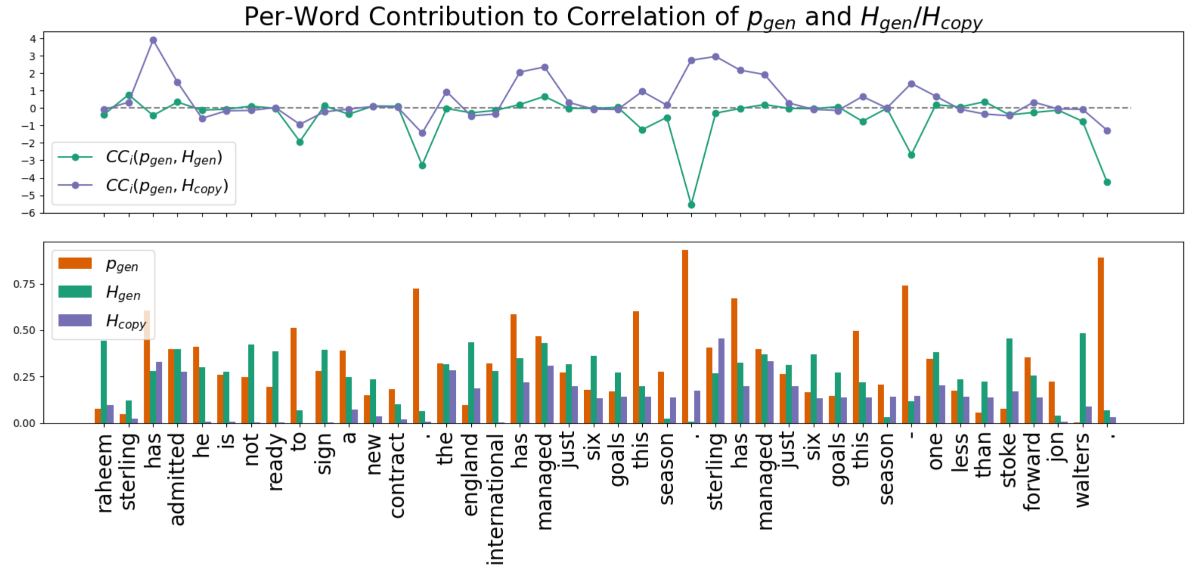

(a) CNN/DailyMail

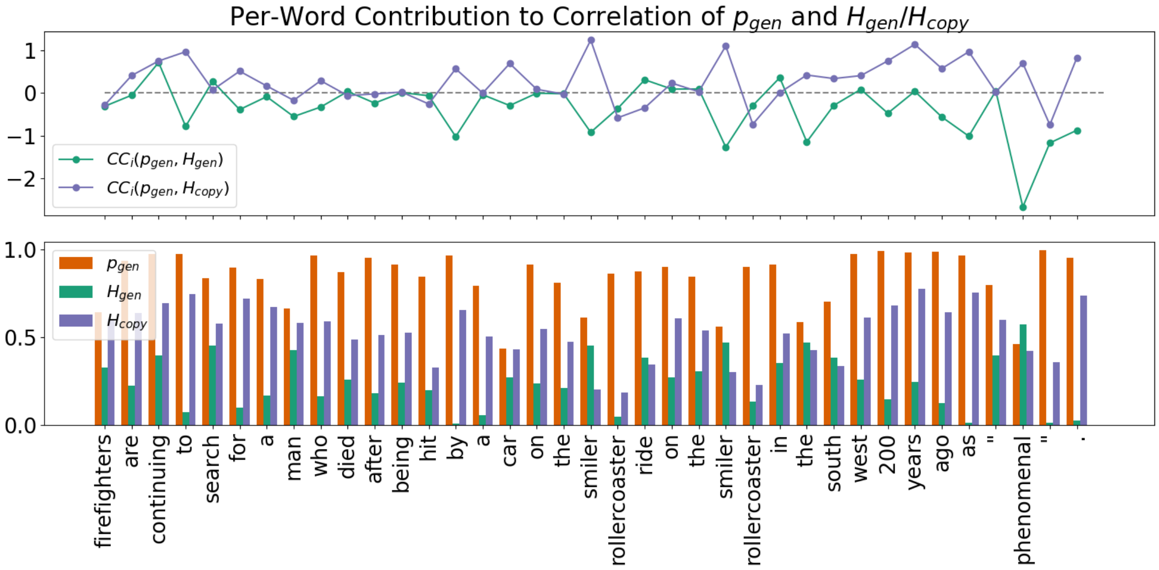

(b) XSum

Figure 1: (Top) Correlation contributions $\mathrm{CC}(p_{\text{gen}}, H_{\text{gen}})$ (green) and $\mathrm{CC}(p_{\text{gen}}, H_{\text{copy}})$ (purple) for a randomly-sampled summary. (Bottom) Bar plot of per-token p_{gen} (orange), and entropy of the generation distribution (green) and copy distribution (purple) for the same summary.

that are more abstractive than those of the base model, in order to disentangle any abstractive *behavior* from abstractive *capabilities*, finding that the model's abstractive capabilities are largely limited to lexical paraphrases, and that forcing the model to generate more novel text yields unfaithful summaries.

3.1 Token-level Analysis

3.1.1 Model

The p_{gen} switch explicitly tells us how much weight is assigned to the generation and copy distributions. See et al. (2017) make qualitative claims about the environments where p_{gen} is highest: "We find that p_{gen} is highest at times of uncertainty such as the beginning of sentences, the join between stitched-together fragments, and when producing periods that truncate a copied sentence." In this section, we evaluative these observations on randomly selected summaries generated with each model.

We quantify the notion of "uncertainty" from See et al. (2017) using information-theoretic entropy (Shannon, 1948) of the distribution that predicts the next word w_i of a generated summary:

$$H_\theta(w_i) = \mathbb{E}_{P_\theta}\left[-\log P_\theta(w_i)\right]. \tag{4}$$

where P_θ is the predictive distribution over the model vocabulary V_θ at a given time step. In our experiments, we use *normalized* entropy, which divides the equation above by $\log_2 |V_\theta|$, to limit the domain to $[0, 1]$ regardless of the vocabulary size. We calculate model-internal entropies H_{gen} and H_{copy} by setting P_θ equal to P_{vocab} and P_{copy}, respectively.

Given the entropy of the copy and generation distributions at each decoder time step, we investigate the relationship between p_{gen}, H_{gen}, and H_{copy} by calculating per-token *correlation contributions*. Intuitively, correlation contribution measures how much an individual token contributes to either positive or negative correlation between p_{gen} and the model entropies.

The Pearson correlation coefficient between two sequences $\mathbf{x} = [x_1, \ldots, x_n]$ and $\mathbf{y} = [y_1, \ldots, y_n]$ can be written as

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5}$$

We calculate the correlation contribution of the pair (x_i, y_i) at index i to be

$$\text{CC}_i = \frac{n(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6}$$

Note that the correlation between $\mathbf{x}$ and $\mathbf{y}$ is equal to the average of $\text{CC}_1, \text{CC}_2, \ldots, \text{CC}_n$, but unlike r, the correlation coefficient, each component CC_i is not bounded by $[-1, 1]$.

3.1.2 Results

Across the test splits, the Pearson correlation betwen p_{gen} and H_{gen} is -0.47 for CNN/DailyMail and -0.55 for XSum. The correlation between p_{gen} and H_{copy} is 0.12 for CNN/DailyMail and 0.54 for XSum. This suggests that the higher-certainty (lower H) distribution is weighted more heavily when combining the generation and copy distributions, since p_{gen} is high when H_{gen} is low, and low when H_{copy} is low.

Visualizing the correlation contributions across a sentence helps us understand how individual tokens are decoded as a function of uncertainty in the abstractive and extractive components of the model. We randomly sample articles from each dataset's test split, and visualize the correlation contributions for the generated summaries in Figure 1. Additional examples may be found in Appendix A.

CNN/DailyMail: The tokens that correlate high p_{gen} with low H_{gen} (high certainty in the abstractive component) are frequently punctuation, and periods in particular. This punctuation appears to be used to truncate sentences at a syntactic boundary, a behavior we quantify in Section 3.2. The correlation of high p_{gen} and high H_{copy} (low certainty in the extractive component) comes from tokens including "has", "managed", ".", and "sterling"; all tokens that appear multiple times in the source document. This suggests a possible role played by generation to tie break when the copy distribution has low certainty about which continuation to copy next.

XSum: The XSum model uses the copy mechanism very infrequently; p_{gen} is frequently large. When p_{gen} is small, we tend to observe uncertainty in the generative component and certainty in the copy component, according to entropy measures. In Figure 1, we see this happens when the proper noun "smiler", a rollercoaster name, is generated. It also happens at the beginning of a quotation, indicating that the model has learned that quotations should be copied from the source document, rather than generated.

Overall, we see a strong contrast in p_{gen} values between the two models. On the extractive-biased CNN/DailyMail dataset, the model learns to copy frequently, generating where necessary to truncate sentences. On the generative-biased XSum dataset, the model acts nearly like a simple seq2seq model, only infrequently using the copy mechanism for the sake of proper nouns and quotations.[2]

3.2 Probing p_{gen}

In the previous section, we made qualitative observations about the relationship between p_{gen} and model entropies, as well as the linguistic environments where p_{gen} is highest. In this section, we

---

[2] This can also be seen in the contrasting gaps between the seq2seq and pointer-generator ROGUE scores reported by See et al. (2017) and Narayan et al. (2018). The former sees a 9-point gap in ROUGE-1, while the latter reports a 1-point gap.

quantify these relationships by predicting p_{gen} with a linear model of lexical, syntactic and distributional factors.

3.2.1 Model Features

In this section, we describe the four feature sets we use to model p_{gen}. These include model-internal entropy measures from the See et al. (2017) summarizer, model-external entropy measures derived from pretrained language models, structural features derived from syntactic parses of summaries, and part-of-speech tags.

Summarization model entropies: We use H_{gen} and H_{copy} as features, hypothesizing, like See et al. (2017), that the uncertainty in the copy and generation distributions will have a significant effect on p_{gen}.

Language model entropies: We also use entropy from three types of language models with varying degrees of lexical and structural expressiveness: a trigram model,[3] a top-down incremental constituency parser (Roark, 2001; Roark et al., 2009), and a unidirectional recurrent neural language model (van Schijndel et al., 2019). These models allow us to directly measure how much p_{gen} may be influenced by lexical, syntactic, and distributional uncertainty in the generated summary independent of the summarization objective.

Structural Features: The summarization model may also condition its decision to copy or generate on the current syntactic environment. While pointer-generator models do not explicitly model syntax, they may exhibit some implicit syntactic knowledge, such as the ability to identify and copy whole constituents. As mentioned above, See et al. (2017) claim that p_{gen} is high at the "the join between stitched-together fragments." Structural features allow us to quantify this, seeing whether the model has learned to prefer copying or generation in particular syntactic environments.

We incorporate two structural measures into our model: the root distance of word w_i, denoted as $\text{D}_{\text{root}}(w_i)$ and the edge distance between word w_{i-1} and w_i, denoted as $\text{D}_{\text{edge}}(w_{i-1}, w_i)$. These measures are calculated on parse trees of generated summaries.[4] Root distance is the distance in the parse tree from the current word to the root node, and corresponds to the depth of the word in the parse tree. This measure will tell us if there is an association between depth in the tree and the decision to copy or generate. Edge distance is the number of intervening edges between the current and previous word in the summary. Edge distance will be smaller within a constituent than across two constituents. This measure allows us to test whether the decision to copy or generate is associated with the size of the syntactic boundary between words.

Part of Speech: In addition to structure, the summarization model may condition its decision to copy or generate on the syntactic category of the most recently generated word. For example, in our preliminary qualitative observations of the CNN/DailyMail model, we found that p_{gen} was higher when decoding punctuation, main verbs and conjunctions. To test the association between part-of-speech and p_{gen} formally, we include the part-of-speech label of the current word in our model.

3.2.2 CNN/DailyMail Results

We predicted p_{gen} using four single feature-set linear models, and a single linear model including all features. We conducted ANOVA tests on all combinations of nested models, and found that each set of features significantly improves the p_{gen} model (all $p < 0.00001$; see Table 1).

Entropies: The coefficients for the model-internal entropy measures H_{gen} and H_{copy} intuitively indicate that as uncertainty in the generation distribution increases, the model is less likely to generate, and as uncertainty in the copy distribution increases, the model is less likely to copy; these relationships were previously explored in Section 3.1.

The three language model entropy estimates are significantly associated with p_{gen}. However, the coefficients are all very small and this feature set individually does the poorest job of explaining p_{gen}'s variance of all the sets we analyzed. This could be due to the fact that, with the exception of the n-gram model, the language model entropy estimates come from different training data than the summarization model. Regardless, while language model entropies significantly improved p_{gen} prediction, the other feature sets showed a much stronger relationship with p_{gen}. Therefore we do not focus on language model entropies in subsequent sections.

Structural Features: Both structural features are significantly associated with p_{gen}. A model fit using only D_{edge} and D_{root} explains 20% of p_{gen}'s

---

[3] A Kneser-Ney trigram model trained on 5.4m tokens of the articles from the training partition of the summarization dataset.

[4] Parses and part of speech tags are generated by the top-down constituency parser.

Feature Set	Feature	β
Summ. Model Entropies (R^2 = 0.274)	H_{gen}	-0.052
	H_{copy}	0.035
LM Entropies (R^2 = 0.140)	H_{LSTM}	0.009
	H_{parser}	0.003
	H_{ngram}	0.009
Structural Features (R^2 = 0.204)	$\text{D}_{\text{edge}}(w_{i-1}, w_i)$	0.018
	$\text{D}_{\text{root}}(w_i)$	-0.031
Part of Speech (R^2 = 0.593)	$	-0.130
	UH	-0.118
	#	-0.116
	NNP	-0.111
	WRB	0.156
	:	0.254
	,	0.269
	.	0.636
Full Model R^2: 0.648		

Table 1: Table of slope coefficients β in the full linear model of p_{gen} in the CNN/DailyMail model. Reported below the name of the feature set is the adjusted R^2 of a model fit only to that feature set. The eight part of speech tags with the largest magnitude β are reported. All reported β are significant via t-test (all $p < 0.00001$).

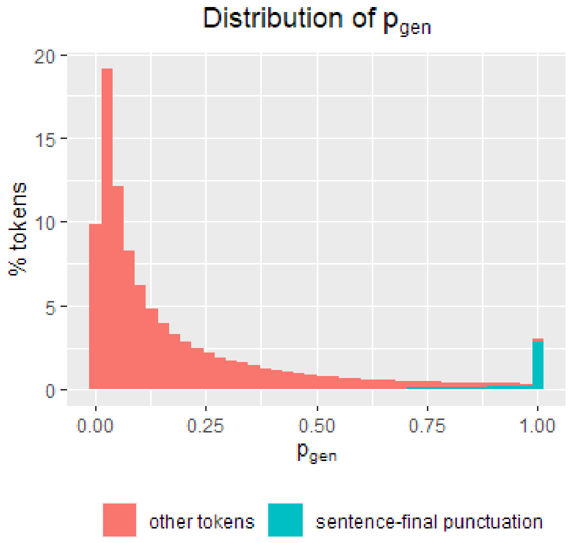

Figure 2: Distribution of p_{gen} across all tokens in the test split of the CNN/DailyMail corpus. Sentence-final punctuation makes up 5% of tokens in the dataset, which accounts for 22% of p_{gen}'s mass

variance ($R^2 = 0.204$). Edge-distance is positively associated with p_{gen}, meaning the larger the syntactic boundary between the previous and current word, the more likely the summarization model is to generate. This provides evidence that the model has some knowledge of syntactic boundaries, and uses the generation component as a means of joining together clauses, in line with the observations of See et al. (2017). We also find that distance to the root node of the parse is negatively associated with p_{gen}. This means that words which are higher in the parse tree are more likely to be generated than copied. Conversely, this means that generated components are unlikely to be associated with complex, deeply nested phrasing, suggesting **the generation component only produces simple shallow substitutions** rather than structurally complex paraphrases or even simple substitutions that modify structurally complex copied elements.

Part-of-Speech: The part of speech tags with the highest negative association with p_{gen} (i.e. those most likely to be copied) are $ (currency symbols), UH (interjection), # (pound symbol), followed by NNP (singular proper nouns). These results are perhaps unsurprising, as interjections and proper nouns are difficult to paraphrase and are often out-of-vocabulary in the generation component of the summarization model. $ and # serve as prefixes to numerical values which cannot be faithfully paraphrased and therefore should be copied directly from the source text. The tag for a cardinal number (CD) also has a relatively strong negative correlation with p_{gen} (β = -0.088).

The part-of-speech tags with the highest positive association with p_{gen} (i.e. those most likely to be generated) are "." (sentence-final punctuation), "," (comma), ":" (colon), and WRB (wh-adverbs, such as "where" or "when"). All of these categories can link two clauses or complete sentences, consistent with the "stitching" hypothesis of See et al. (2017).

The mean p_{gen} value of all tokens in the test dataset was 0.204, while the mean p_{gen} value for sentence-final tokens was 0.915. Further inspection of the p_{gen} distribution reveals a cluster of outliers at p_{gen} = 1.0. Figure 2 shows the distribution of p_{gen} values. We find that, of all tokens with $p_{\text{gen}} > 0.95$, 92.1% are sentence-final punctuation. Despite making up 5% of all tokens, periods account for 22.1% of the total mass of p_{gen} in the dataset. This suggests that sentence final punctuation is entirely controlled by the generation distribution. Additionally, we find that of all 5-grams in generated-summaries ending with sentence-final punctuation, 52% are also present in the article text, compared to 12% in the reference summaries. Despite the large p_{gen} values exhibited by sentence-final punctuation, the model only generates punctuation in novel contexts less than half of the time, suggesting that **even when the model heavily utilizes its generative component, it essentially generates a copy**

of the source text.

Our explanatory model of p_{gen} shows that model entropy, syntactic depth, syntactic boundary size, and part-of-speech are associated with p_{gen}. The strongest predictor of p_{gen} is the part-of-speech of the current word, with copying most strongly associated with numbers, number prefixes and proper nouns, and generation most strongly associated with punctuation. We find that sentence-final punctuation is handled almost entirely by the generative component of the model, despite the fact that sentence-final punctuation occurs in novel contexts less than half of the time.

3.2.3 XSum Results

Overall, we find that the variance of p_{gen} in the XSum model is well explained by model-internal entropy, and relatively poorly explained by linguistic features. We believe this is driven by the categorically different behaviors of each model.[5] While the CNN/DailyMail model only uses the generative component to join together copied constituents, the generative component dominates the XSum model's behavior. The mean p_{gen} value across all tokens in the XSum dataset was 0.828, compared to 0.204 in the CNN/DailyMail dataset. While the structural features $\text{D}_{\text{edge}}(w_{i-1}, w_i)$ and $\text{D}_{\text{root}}(w_i)$ explained 20.4% of the variance of p_{gen} in the CNN/DailyMail model, these features only explain 4.9% of the variance in the XSum model. Part of speech also does a poorer job of explaining the variance in XSum's p_{gen}. While part of speech explains 59.3% of the variance of p_{gen} in the CNN/DailyMail model, part of speech tags only explain 23.0% in the XSum model.

While the CNN/DailyMail model assigned an abnormally high p_{gen} value to punctuation, we do not observe this behavior in the XSum model. The CNN/DailyMail model appeared to make use of the ".", ":" and "," tokens to join together copied sentences, but none of these tokens are a significant predictor of p_{gen} in the XSum model. This suggests that the XSum model does not use the generation distribution to connect copied clauses.

While the XSum model appears not to use the copy and generation distributions in the same way as the CNN/DailyMail model, we still observe some clear and intuitive associations between part of speech tags and p_{gen}. In particular, the XSum model appears to use the copy distribution to handle words which are likely to be out-of-vocabulary for the generation distribution. For example, singular and plural proper nouns, interjections and foreign words (NNP, NNPS, UH, and FW respectively) are associated with low values of p_{gen} (copying), while all types of verbs are associated with large values of p_{gen} (generation).

We conclude that the CNN/DailyMail model primarily makes use of lexical and syntactic information such as clause boundaries and punctuation to modulate between copying and generation. By contrast, the XSum model primarily relies on the generation distribution, and backs off to the copy distribution at times of high generation uncertainty or high copy certainty, such as when copying a quote or a proper name.

3.3 Modifying p_{gen}

3.3.1 Model

Taking advantage of the smooth interpolation between the generation and copy distribution, we experiment with forcing the CNN/DailyMail model to be more abstractive. This, we expect, will allow us to differentiate between the abstractive *behavior* we observe in the model summaries and the abstractive *capabilities* that the model may have but which it only uses infrequently in practice. We do so by artificially modifying p_{gen} during decoding. If $p_{\text{min}} \in [0, 1]$ is a parameter that represents the minimum value of p_{gen} we allow, we then modify p_{gen} as follows:

$$p^*_{\text{gen}} = p_{\text{min}} + (1 - p_{\text{min}})p_{\text{gen}} \tag{7}$$

This may be viewed as a linear interpolation from the range $[0, 1]$ to $[p_{\text{min}}, 1]$. As p_{min} grows, the model is forced to rely more heavily on the generation distribution rather than the copy distribution.[6]

3.3.2 Results

We use the same randomly sampled articles used in Section 3.1.[7] Generated summaries for p_{min} values in $[0, 0.25, 0.50, 0.75, 1.0]$ can be found in Table 2.

Consistent with previous studies, we find that the model is effective at producing grammatical output. At small values of p_{gen}, the model mostly copies sentences verbatim, but shows the ability to cut a sentence short in a grammatical manner. For example, "raheem sterling has admitted he is not

---

[5] The full table of model coefficients can be found in Table 5 of Appendix C.

[6] We do not run this experiment on the XSum model because it already usually has a large p_{gen}.

[7] We see similar patterns in other randomly-sampled summaries, shared in Appendix B.

Article Text: raheem sterling has admitted he is not ready to sign a new contract at liverpool deal despite being offered a # 100,000-a-week deal to stay with the merseyside club . the 20-year-old wideman edged closer towards an anfield exit after revealing in an interview with the bbc on wednesday that he would have signed for a lot less a year ago . however , despite being one of liverpool 's star men , sterling has struggled to repeat the impressive form he showed for the reds last season . the england international has managed just six goals this season - one less than stoke frontman jon walters - while his conversion rate and minutes per goal ratio have worsened as the graphic below shows . raheem sterling has managed just six goals this season - one less than stoke forward jon walters -lrb- left -rrb- .
Reference: raheem sterling has revealed he is not ready to sign a new liverpool deal . the reds wideman has struggled to repeat last season's impressive form . the 20-year-old liverpool star has managed just six goals this season . read: sterling insists he is not a 'money-grabbing 20-year-old' sterling: what he said about contract talks... and what he meant . click here for the latest liverpool news .
$\mathbf{p}_{\min} = \mathbf{0}$: raheem sterling has admitted he is not ready to sign a new contract . the england international has managed just six goals this season . sterling has managed just six goals this season - one less than stoke forward jon walters .
$\mathbf{p}_{\min} = \mathbf{0.25}$: raheem sterling has admitted he is not ready to sign a new contract . the england international has managed just six goals this season . the england international has managed just six goals this season .
$\mathbf{p}_{\min} = \mathbf{0.50}$: raheem sterling has admitted he is not ready to sign a new contract . the england international has managed just six goals this season . the england international has managed just six goals this season .
$\mathbf{p}_{\min} = \mathbf{0.75}$: raheem sterling has admitted he is not ready to sign a new deal . the 20-year-old has scored just six premier league goals this season . the 20-year-old has scored just three goals this season .
$\mathbf{p}_{\min} = \mathbf{1}$: man utd face manchester city in the premier league on saturday . the striker has scored just four premier league goals this season . the 19-year-old has scored just three goals this season . click here for all the latest premier league news .

Table 2: Summaries generated for the same randomly selected article with varying values of p_{min}. Differences from the base model summary are highlighted in blue, while non-faithful text is highlighted in red.

ready to sign a new contract at liverpool deal..." is shortened to "raheem sterling has admitted he is not ready to sign a new contract."

At greater values of p_{gen}, the model continues sentences in a consistent fashion despite substituting nouns or verbs at the beginning or middle of the sentences. For example, "sterling has managed just six goals..." at $p_{\text{min}} = 0$ becomes "the 20-year-old has scored just six premier league goals" at $p_{\text{min}} = .75$. However, we do not observe significant paraphrasing beyond these simple substitutions, and at high values of p_{min}, where the model is forced to rely heavily on the generation distribution, we begin to observe hallucinations where the model inserts inaccurate information about the player's age and the number of goals scored. When $p_{\text{min}} = 1$, the model generates a completely hallucinated sentence, "man utd face manchester city in the premier league on saturday" and a non-informative advertisement "click here for all the latest premier league news."

4 Discussion

Understanding the limitations preventing abstractive summarization models from paraphrasing effectively is our ultimate aim, but answering that question requires an understanding of current models' abstraction capabilities. In this paper, we analyze the abstractions of which the pointer-generator model (See et al., 2017) is capable.

When trained on CNN/DailyMail, we find that sentence truncation is the most common form of paraphrasing. Punctuation tokens are associated with high generation rates and low entropy in the generation distribution. Additionally, high p_{gen} often results in generating the token that comes next in a phrase already being copied verbatim, suggesting that high p_{gen} merely gives the model the *option* to generate novel text, but that the model rarely makes use of it. Artificially increasing p_{gen} does not significantly change this behavior, introducing increased rates of synonym substitution as well as increased rates of non-faithful hallucination.

When trained on XSum, the model makes much less use of the copy mechanism, largely generating novel text with a few exceptions, including the copying of proper nouns and parts of quotations. The model generally produces topical summaries, but ones that aren't necessarily grammatical or faithful to the original article. For example, the randomly selected summary used in Figure 1 repeats itself and wanders, "... on the smiler rollercoaster on the smiler rollercoaster in the south west 200 years ago as 'phenomenal'". This comes after a hallucination, "firefighters are continuing to search for a man" even though the article describes the rescue from the rollercoaster crash in the past tense. We hypothesize that the phrase "firefighters are continuing to search" is a relatively common phrase in news articles that the model learned from the training data. Such frequency biases likely contribute to the faithfulness issues in abstractive summarizers reported in previous literature.

Our results give context to previous observations that summarization model unfaithfulness increases with abstraction (Maynez et al., 2020; Durmus et al., 2020; Kryscinski et al., 2020) and that abstractive models are prone to output repetition (See et al., 2019; Holtzman et al., 2020). To faithfully paraphrase, a model must understand both the syntax and the semantics of the original text. The models we studied were able to recognize syntactic boundaries, proper nouns, and noun phrases that could be substituted with synonyms. However, the models didn't appear to comprehend the meaning of the text well enough to generate faithful complex paraphrases. This is unacceptable in high-risk domains such as healthcare; Zhang et al. (2018b) train a model to summarize radiology findings, but only 67% of their summaries are judged at least as good as human summaries, in a domain where errors can have a major impact on human lives.

In our work, the explicit switch between abstractive and extractive modes enabled us to directly observe the conditions under which abstractive summarization was chosen as a strategy, and to force an abstractive summarization strategy to disentangle paraphrasing behavior from capabilities. We found that the See et al. (2017) model trained on CNN/DailyMail did learn simple forms of paraphrasing, despite the extractive bias of the dataset. We conclude that pointer-generator models are *capable* of simple paraphrasing regardless of training data, even though they *behave* in ways that rely on the frequency biases of the training dataset. However, they also appear *incapable* of producing significant paraphrases that are grammatical, non-repetitive, and faithful to the source document. This suggests that using an abstractive-biased dataset alone is not enough for a model to learn robust and faithful paraphrasing strategies. Rather, when trained on XSum, the pointer-generator model seems to simply learn that it should not copy from the source text. Future work should investigate how either datasets or models can improve the training signal that allows the model to understand the underlying semantics of the source document.

Related to our work, Xu et al. (2020) studied the summarization strategies of state-of-the-art transformer summarization models. Since their models did not contain an explicit copy/generation switch, they used n-gram overlap between source documents and summaries as a proxy to measure a summary's "extractiveness." They found a similar result to ours, that high n-gram overlap ("copying") corresponded to low entropy in the decoder's output distribution when the model was trained on CNN/DailyMail.[8] Their findings suggest that our results likely generalize to a much broader class of summarization models than the pointer-generator models studied here.

Finally, Liu and Liu (2010) found that ROUGE metrics poorly correlate with human evaluations, leading to recent models being evaluated with human judgements, but these evaluations often disagree on what they are measuring, whether it is faithfulness, informativity, or the unqualified "quality" of a summary (Zhang et al., 2018a, 2020; Dou et al., 2020). Developing best practices on how abstractive summarizers should be evaluated for their paraphrasing ability is another problem we leave for future work.

5 Conclusion

In this paper, we presented three experiments that evaluate the abstraction capabilities of the pointer-generator neural summarization model. Our results conclude that on extractive training data, the model uses only simple paraphrasing strategies that truncate sentences at syntactic boundaries, allowing the model to stay grammatically accurate as well as faithful to the source document. We explore two ways to make the model use abstractive summarization strategies: modifying the model so that it relies more heavily on its abstractive component, and training a new model on an abstractive-biased dataset. In both cases, the model shows simple paraphrasing capabilities but frequently generates unfaithful paraphrases. These results highlight current limitations of abstractive summarization, where in lieu of semantic understanding, models must rely on extractive heuristics in order to stay faithful.

Acknowledgements

We thank our reviewers for their helpful suggestions. We also thank Esin Durmus, Ryan Benmalek, and Claire Cardie for helpful discussions about abstractive summarization. Finally, we thank Shashi Narayan and Shay Cohen for helping us reproduce their pointer-generator model trained on XSum.

---

[8] Their findings are more difficult to interpret when trained on XSum, partially due to the lack of an explicit extractive/abstractive summarization switch in their models.

References

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2020. Gsum: A general framework for guided neural abstractive summarization. *arXiv preprint arXiv:2010.08014*.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

K. Ganesan. 2018. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *ArXiv*, abs/1803.01937.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

F. Liu and Y. Liu. 2010. Exploring correlation between rouge and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):187–196.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. pages 324–333.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Marten van Schijndel, Aaron Mueller, and Tal Linzen. 2019. Quantity doesn't buy quality syntax with neural language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5831–5837, Hong Kong, China. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861, Hong Kong, China. Association for Computational Linguistics.

C. E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Jiacheng Xu, Shrey Desai, and Greg Durrett. 2020. Understanding neural abstractive summarization models via uncertainty. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6275–6281, Online. Association for Computational Linguistics.

Fang-Fang Zhang, Jin-ge Yao, and Rui Yan. 2018a. On the abstractiveness of neural document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 785–790.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Yuhao Zhang, Daisy Yi Ding, Tianpei Qian, Christopher D. Manning, and Curtis P. Langlotz. 2018b. Learning to summarize radiology findings. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 204–213, Brussels, Belgium. Association for Computational Linguistics.

Appendix A. Additional Correlation Contribution Examples

This appendix includes additional examples of $\mathrm{CC}(p_{\text{gen}}, H_{\text{gen}})$, the per-token correlation contributions for randomly selected summaries.

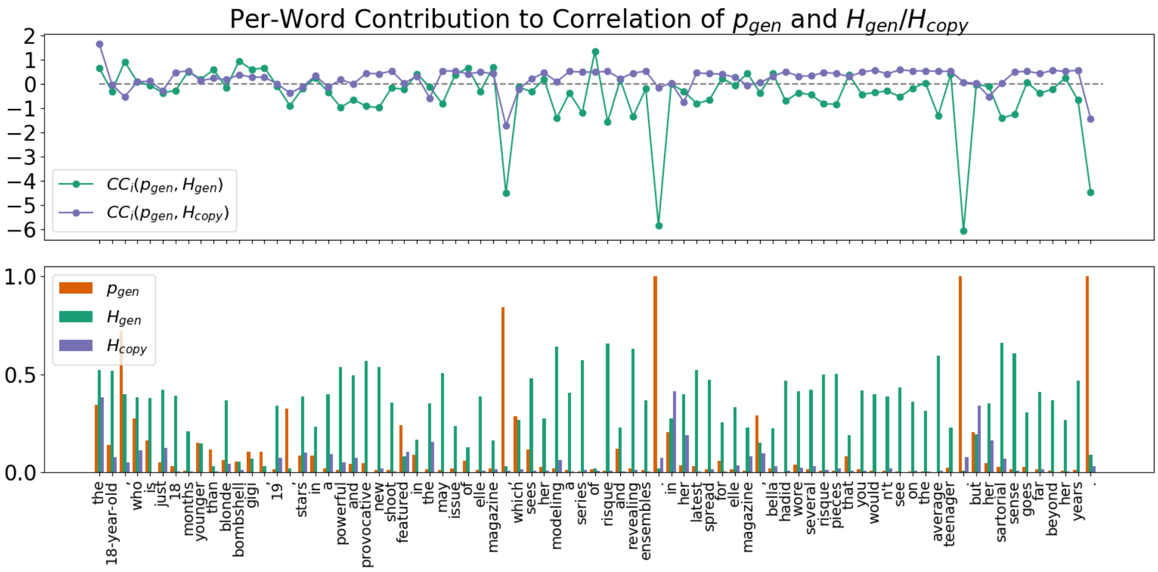

(a) CNN/DailyMail Example 2

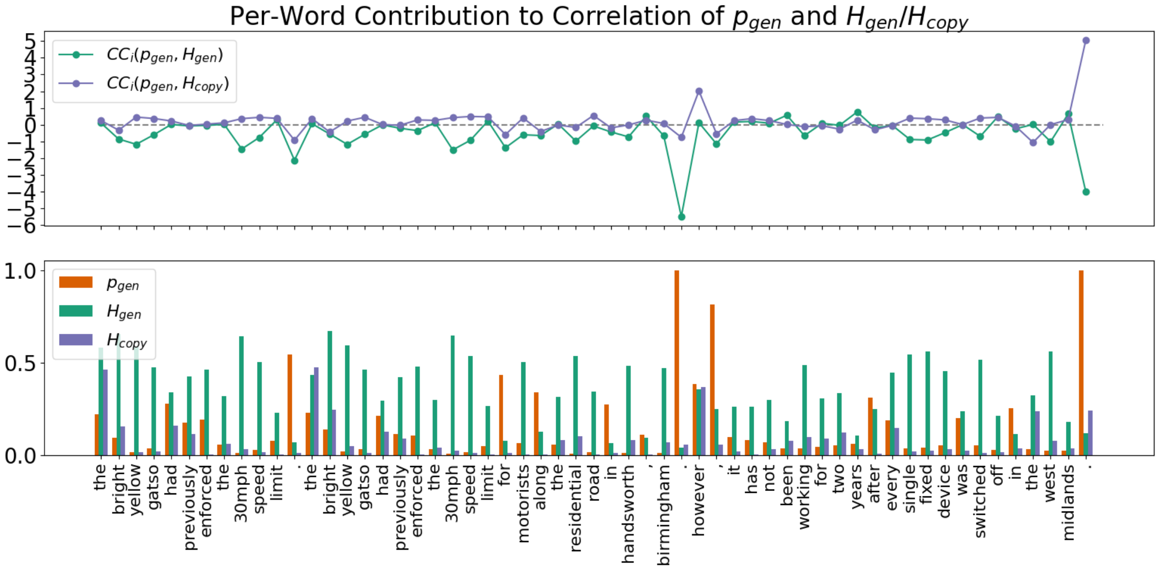

(b) CNN/DailyMail Example 3

Figure 3: Bar plot of per-token p_{gen} and entropy of the generation distribution (purple) and copy distribution (blue), plotted under correlation contributions $\mathrm{CC}(p_{\text{gen}}, H_{\text{gen}})$ (purple) and $\mathrm{CC}(p_{\text{gen}}, H_{\text{copy}})$ (blue) for a randomly-sampled CNN/DailyMail test summaries.

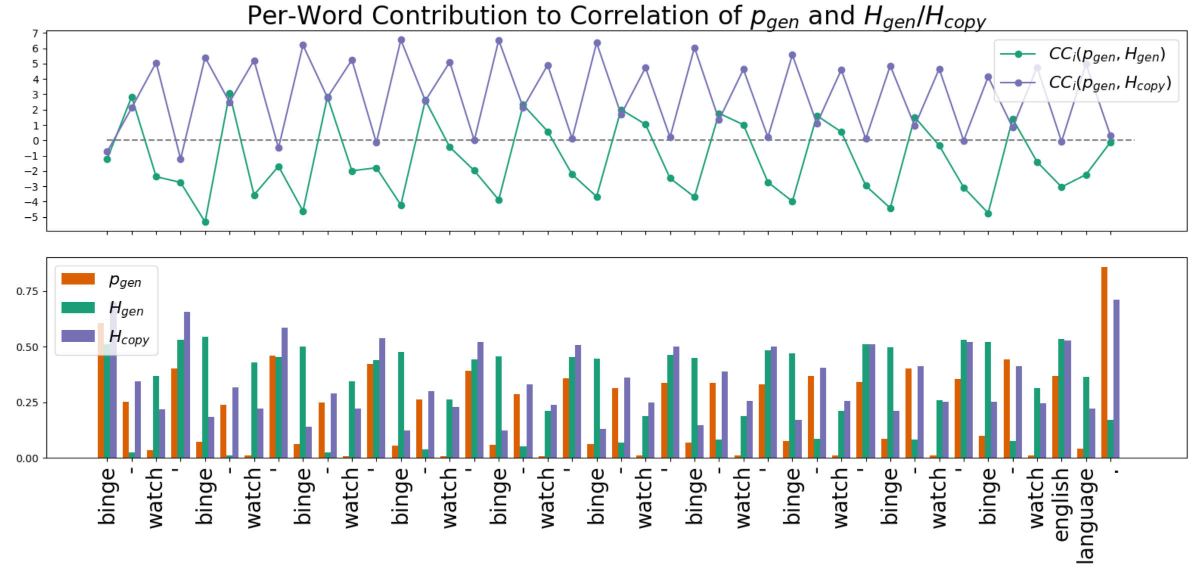

(a) XSum Example 2

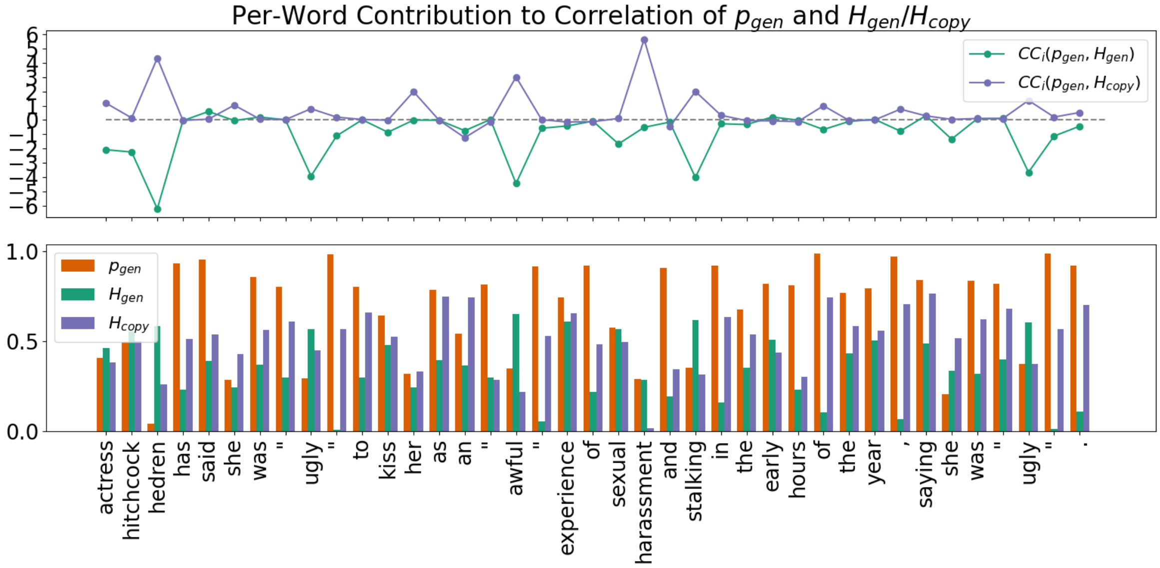

(b) XSum Example 3

Figure 4: Bar plot of per-token p_{gen} and entropy of the generation distribution (purple) and copy distribution (blue), plotted under correlation contributions $\text{CC}(p_{\text{gen}}, H_{\text{gen}})$ (purple) and $\text{CC}(p_{\text{gen}}, H_{\text{copy}})$ (blue) for a randomly-sampled XSum test summaries.

Appendix B. Additional Summaries With Modified p_{min}

This appendix contains additional examples of randomly selected summaries generated with modified values of p_{min}.

Article Text: gigi hadid 's younger sister bella hadid is quickly catching up with her star sibling 's phenomenal fashion success , carving out an impressive career for herself within the industry . the 18-year-old , who is just 18 months younger than blonde bombshell gigi , 19 , stars in a powerful and provocative new shoot featured in the may issue of elle magazine , which sees her modeling a series of risque and revealing ensembles . in one image , the dark-haired beauty poses in a sheer fleur du mal lace bodysuit , with her arms thrown casually above her head . scroll down for video . it 's good to be a model : bella hadid models a tantalizing $ 895 fleur du mal lace bodysuit and $ 1,250 giuseppe zanotti boots . giuseppe zanotti cowboy boots . on sale for $ 500 at 6pm ! visit site . she may be just 18 , but her sartorial sense goes far beyond her years . and in her latest spread for elle magazine , bella hadid wore several risque pieces that you would n't see on the average teenager . but then again , the budding supermodel is n't a normal teen [...]
Reference: bella, 18, is the younger sister of guess campaign star gigi hadid, 19 . the rising star poses in a series of provocative outfits for the may issue of elle . fellow fashion favorite hailey baldwin also features in the issue, appearing in her own separate shoot and interview .
$\mathbf{p}_{\text{min}} = \mathbf{0}$: the 18-year-old , who is just 18 months younger than blonde bombshell gigi , 19 , stars in a powerful and provocative new shoot featured in the may issue of elle magazine , which sees her modeling a series of risque and revealing ensembles . in her latest spread for elle magazine , bella hadid wore several risque pieces that you would n't see on the average teenager . but her sartorial sense goes far beyond her years .
$\mathbf{p}_{\text{min}} = \mathbf{0.25}$: the 18-year-old , who is just 18 months younger than blonde bombshell gigi , 19 , stars in a powerful and provocative new shoot featured in the may issue of elle magazine , which sees her modeling a series of risque and revealing ensembles . in her latest spread for elle magazine , bella hadid wore several risque pieces that you would n't see on the average teenager .
$\mathbf{p}_{\text{min}} = \mathbf{0.50}$: the 18-year-old is just 18 months younger than blonde bombshell . she is just 18 months younger than blonde bombshell gigi , 19 . she is just 18 months younger than blonde bombshell gigi .
$\mathbf{p}_{\text{min}} = \mathbf{0.75}$: the 18-year-old is just 18 months younger than blonde bombshell .she is just 18 months younger than blonde bombshell . she is just 18 months younger than blonde bombshell . she is just 18 months younger than blonde bombshell .
$\mathbf{p}_{\text{min}} = \mathbf{1}$: the [UNK] , 33 , has been dating for three years . she is the daughter of [UNK] and [UNK] . she is the daughter of [UNK] 's daughter . her new album is priced at $ [UNK] -lrb- # [UNK] -rrb- .

Article Text: the owners of this house better not leave too quickly , after a speed camera was angled directly on their front door . the bright yellow gatso had previously enforced the 30mph speed limit for motorists along the residential road in handsworth , birmingham . however , it has not been working for two years after every single fixed device was switched off in the west midlands . big brother is watching : a speed camera has been turned round and is pointing at this house in birmingham , west midlands . the speed camera has not been working for more than two years [...]
Reference: speed camera discovered pointing at house in handsworth, birmingham . fixed cameras switched off across the west midlands in spring of 2013 . site is not going to be part of a new trial using digital technology . obsolete camera may now be taken down after engineers examine device .
$\mathbf{p}_{\text{min}} = \mathbf{0}$: the bright yellow gatso had previously enforced the 30mph speed limit . the bright yellow gatso had previously enforced the 30mph speed limit for motorists along the residential road in handsworth , birmingham . however , it has not been working for two years after every single fixed device was switched off in the west midlands .
$\mathbf{p}_{\text{min}} = \mathbf{0.25}$: the bright yellow gatso had previously enforced the 30mph speed limit for motorists along the residential road in handsworth , birmingham . it has not been working for more than two years after every single fixed device was switched off in the west midlands . the speed camera has not been working for more than two years .
$\mathbf{p}_{\text{min}} = \mathbf{0.50}$: the bright yellow gatso had previously enforced the 30mph speed limit . it has not been working for more than two years . it has not been working for more than two years after every single fixed device **.**
$\mathbf{p}_{\text{min}} = \mathbf{0.75}$: the bright yellow gatso had previously enforced the 30mph speed limit . it has not been working for more than two years . it has not been working for more than two years . it has not been working for more than two years .
$\mathbf{p}_{\text{min}} = \mathbf{1}$: warning : graphic content . it is believed to have been in the past of the past two years . it is believed to have been in the past of the past two years .

Table 3: Summaries generated for additional randomly selected articles from CNN/DailyMail with varying values of p_{min}. Differences from the base model summary are highlighted in blue, while non-faithful text is highlighted in red.

Article Text: meaning ” to watch a large number of television programmes (especially all the shows from one series) in succession ” , it reflects a marked change in viewing habits , due to subscription services like netflix . lexicographers noticed that its usage was up 200 % on 2014 . other entries include dadbod , ghosting and clean eating . helen newstead , head of language content at collins , said : ” the rise in usage of ’ binge - watch ’ is clearly linked to the biggest sea change in our viewing habits since the advent of the video recorder nearly 40 years ago . ” it ’s not uncommon for viewers to binge - watch a whole season of programmes such as house of cards or breaking bad in just a couple of evenings - something that , in the past , would have taken months - then discuss their binge - watching on social media . ” those partaking in binge - watching run the risk of dadbod , one of ten in the word of the year list [...] the list of collins ’ words of the year offers a fascinating snapshot of the ever - changing english language , ” said newstead . those words that remain popular could be included in the next print edition of the collins english dictionary , due in 2018 .

Reference: collins english dictionary has chosen binge-watch as its 2015 word of the year.

Summary: binge - watch ’ binge - watch ’ binge - watch ’ binge - watch ’ binge - watch ’ binge - watch ’ binge - watch ’ binge - watch ’ binge - watch ’ binge - watch english language .

Summary with Coverage: the risk of binge - watch ’ binge - watch english language is ” clearly uncommon ” , according to a list of entries from the collins english media recorder of the year list .

Article Text: writing in her autobiography , she claimed the director ” threw himself ” on top of her in the back of his limousine and tried to kiss her . the actress described the encounter as ” an awful , awful moment ” . hedren added that she did n’t tell anyone because ” sexual harassment and stalking were terms that did n’t exist ” in the early 1960s . she continued : ” besides , he was alfred hitchcock [...] the actress , now 86 , made the claims in her autobiography tippi : a memoir , which is published in november . she has spoken in the past about the director ’s alleged treatment of her , but has gone into more detail in the memoir . hedren described a later encounter in hitchcock ’s office where the director ” suddenly grabbed ” her and ” put his hands ” on her . she wrote : ” it was sexual , it was perverse , and it was ugly , and i could n’t have been more shocked and more repulsed . ” [...] the actress said hitchcock then made her life difficult , refusing to submit her work for the oscar nominations or let her take on other acting roles while he still had her under contract [...]

Reference: actress tippi hedren has claimed alfred hitchcock sexually harassed her while they worked together in the 1960s.

Summary: actress hitchcock hedren has said she was ” ugly ” to kiss her as an ” awful ” experience of sexual harassment and stalking in the early hours of the year , saying she was ” ugly ” .

Summary with Coverage: actress hitchcock hitchcock , best known by the director of the oscar - winning director , has died at the age of 86 , the actress has announced on her return to the memoir .

Table 4: Summaries generated for additional randomly selected articles from XSum with varying values of p_{min}. Summaries with coverage enabled also included. Non-faithful text is highlighted in red

Appendix C. Explanatory p_{gen} Model for XSum Dataset

Feature Set	Feature	β
Summ. Model Entropies	H_{gen}	-0.099
(R^2 = 0.476)	H_{copy}	0.093
LM Entropies	H_{LSTM}	0.009
(R^2 = 0.123)	H_{parser}	0.003
	H_{ngram}	-0.013
Structural Features	$\text{D}_{\text{edge}}(w_{i-1}, w_i)$	-0.005
(R^2 = 0.049)	$\text{D}_{\text{root}}(w_i)$	-0.001
Part of Speech (R^2 = 0.230)	NNPS	-0.166
	FW	-0.162
	UH	-0.143
	NNP	-0.089
	VBD	0.174
	LS	0.179
	VBN	0.178
	WP$	0.193
Full Model R^2: 0.547		

Table 5: Table of slope coefficients β in the full linear model of p_{gen} in the XSum model. Reported below the name of the feature set is the adjusted R^2 of a model fit only to that feature set. The eight part of speech tags with the largest magnitude β are reported. All reported β are significant via t-test (all $p < 0.00001$).

AgreeSum: Agreement-Oriented Multi-Document Summarization

Richard Yuanzhe Pang[1*§] **Adam D. Lelkes**[2*] **Vinh Q. Tran**[2*] **Cong Yu**[2]
[1]New York University, New York, NY 10011, USA
[2]Google Research, New York, NY 10011, USA
`yzpang@nyu.edu, {lelkes,vqtran,congyu}@google.com`

Abstract

We aim to renew interest in a particular multi-document summarization (MDS) task which we call AgreeSum: agreement-oriented multi-document summarization. Given a cluster of articles, the goal is to provide abstractive summaries that represent information common and faithful to all input articles. Given the lack of existing datasets, we create a dataset for AgreeSum, and provide annotations on article-summary entailment relations for a subset of the clusters in the dataset. We aim to create strong baselines for the task by applying the top-performing pretrained single-document summarization model PEGASUS onto AgreeSum, leveraging both annotated clusters by supervised losses, and unannotated clusters by T5-based entailment-related and language-related losses. Compared to other baselines, both automatic evaluation and human evaluation show better article-summary and cluster-summary entailment in generated summaries. On a separate note, we hope that our article-summary entailment annotations contribute to the community's effort in improving abstractive summarization faithfulness.

1 Introduction

Recent works have made great progress in single-document summarization (SDS) thanks to the encoder-decoder framework and pretraining procedures (Cho et al., 2014; Rush et al., 2015; Narayan et al., 2018; Zhang et al., 2020a). There is a growing interest in multi-document summarization (MDS; Zopf, 2018; Fabbri et al., 2019; Liu and Lapata, 2019; Chu and Liu, 2019; Wang et al., 2020b, et seq.), with applications in search engines, news clustering, timeline generation, and other areas. Past MDS research has primarily focused on summarizing articles such that the summary covers an event "comprehensively" while "avoiding redundancy" (Fabbri et al., 2019). We can say that most existing MDS tasks summarize the "union" of the articles.

In this paper, we discuss *agreement-oriented multi-document summarization* (AgreeSum), in which we aim to *abstractively* summarize the "intersection" of the articles. More specifically, the input to the task is a cluster of articles, and the expected output is a summary that represents information common and faithful to all input articles in the cluster (Section 3). A few works (discussed in Section 2) have investigated the problem, without using modern neural-network-based methods. The motivation for reviving interest in AgreeSum is twofold. First, given that certain microscopic details are not likely present in all articles in a given cluster, they would be filtered out through AgreeSum. If the source articles reflect different points of view, AgreeSum provides a way of capturing the common ground.

The second motivation for AgreeSum lies in the pursuit of summarization faithfulness. AgreeSum is timely, given that recent works have shown difficulty of producing faithful abstractive summaries (Falke et al., 2019; Maynez et al., 2020; Kryscinski et al., 2020; Durmus et al., 2020; Zhou et al., 2021), though in the SDS setting. AgreeSum could allow practitioners of abstractive summarization systems to carefully study topics related to faithfulness and hallucination.

Given the scarcity of readily available data, we create a dataset[1] based on English Wikipedia current events portal (WCEP)[2]. WCEP contains neutral human-written news summaries with links

---

[§] Work completed during internship at Google Research.
[*] Equal contribution.

[1] `https://github.com/google-research-datasets/AgreeSum`

[2] `https://en.wikipedia.org/wiki/Portal:Current_events`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3377–3391
August 1–6, 2021.

to news articles (with usually one link per summary). We first extract human-written summaries on WCEP, the linked articles, and semantically similar articles to the linked articles, in order to obtain article clusters associated with summaries. We then annotate the entailment relationship for a subset of the cluster-summary pairs (i.e., whether or not a given article from the cluster semantically entails the summary). From there, we build cluster-summary pairs for AgreeSum (Section 3.2).

We build upon previous SDS research, using the top-performing pretrained PEGASUS model (Zhang et al., 2020a) as the starting point for our models. Using our dataset, we first examine a few baseline models and show that several pretraining-based summarization models fail at generating summaries that satisfy AgreeSum's requirements. We also propose an approach that integrates both supervised (using the annotated portion of the dataset) and unsupervised losses (namely, an entailment loss and a language loss to be discussed later, using both the annotated portion and the unannotated portion) while leveraging PEGASUS. We show the effectiveness of a simple policy gradient-based algorithm (Sutton et al., 2000) in which the rewards are based on a T5-based article-summary entailment model (Raffel et al., 2020). To summarize the contributions:

- We introduce the AgreeSum task and the WCEP-based dataset (§3.2).

- A subset of *article*-summary pairs are annotated with entailment information (§4). On a separate note, the article-summary-level recognizing textual entailment (RTE)[3] task could stand as a challenging task on its own. The annotations could be of interest to the research in improving abstract summarization faithfulness, in the context of not only AgreeSum but also general single-document summarization tasks.

- We develop simple ways of applying PEGASUS to AgreeSum. We provide a few baselines as well as a model that uses unsupervised entailment-related and language-related losses to complement the supervised finetuning of PEGASUS (§5). Both automatic and human evaluations are performed (§6). In particular, we show that the T5-based entailment model can be integrated into summarization models to produce summaries that are entailed in source articles.

2 Related Work

Traditional MDS. DUC (Paul and James, 2004; Dang, 2005) and TAC (Owczarzak and Dang, 2011) MDS data are among the first high-quality relevant datasets. These datasets are human-curated, but tiny in terms of the number of examples. Recent works have explored creative methods for obtaining low-cost MDS datasets. Liu et al. (2018) use Wikipedia articles as summaries and the cited articles as inputs. Antognini and Faltings (2020) use Wikipedia in a similar way, in the video games domain. Fabbri et al. (2019) rely on the website Newser with lengthy human-aggregated extractive summaries. Gholipour Ghalandari et al. (2020), also based on WCEP, is especially relevant.

However, our dataset is different in the following ways. First, all of our articles in the same cluster are about the same event. Next, a large part of our dataset is annotated with article-summary entailment information (i.e., in each of the clusters, for each article in the cluster, whether the article entails the summary; see Section 3). Further, among the annotated article-summary pairs, about half of the articles entail the summary, and half of the articles do not entail the summary. This property makes a realistic and difficult setting for AgreeSum tasks.

In terms of recent MDS neural methods, Chu and Liu (2019) summarize opinions using an autoencoder, in which case the input is much shorter than a typical article. Liu et al. (2019a) improve the model by encoding articles and summaries in the same space. Other novel approaches include using sentence compression in the seq2seq framework (Baziotis et al., 2019), jointly learning sentence fusion and paraphrasing (Nayeem et al., 2018), using graph neural networks to help extraction (Wang et al., 2020b), using spectral methods (Wang et al., 2020c), using transfer learning based on a novel pretraining method called gap-sentence prediction (Zhang et al., 2020a) on a news-specific corpus, among a few others (Li et al., 2020; Gu et al., 2020; Mao et al., 2020).

For MDS tasks that summarize the intersection of articles, a few past works have discussed the helpfulness of models that identify common infor-

---

[3] The RTE datasets (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007) correspond to the following two-class classification task: given a premise and a hypothesis, a model would decide if the premise entails the hypothesis.

mation "centroids" among multiple related documents, so as to allow internet users to more efficiently understand events (Radev et al., 2000; Barzilay and McKeown, 2005). The attempted non-neural models rely heavily on topic/theme detection and tracking, and are more extractive than abstractive (Radev et al., 2004). The AgreeSum idea has not been fully explored by researchers since much stronger text generation technologies became available. It is timely to revisit the problem also because recent stronger neural abstractive summarization models are prone to hallucination, as are neural text generation in general (Wiseman et al., 2017; Tian et al., 2019; Wang and Sennrich, 2020; Pang and He, 2020).

Summarization hallucination and evaluation. Non-hallucination is a necessary but not sufficient condition for performing well in AgreeSum: the summary not only needs to be entailed in the union of the articles, but also must be entailed in each of the articles. Unfortunately, recent works have shown the difficulty of identifying and mitigating hallucination (Maynez et al., 2020).

Evaluation-wise, researchers have found that metrics like ROUGE (Lin, 2004) and BERTScore (Zhang et al., 2020b) are only weakly correlated with factuality. Durmus et al. (2020) and Wang et al. (2020a) have therefore proposed using question-answering systems for evaluating summarizers. Recently, Zhou et al. (2021) have made progress in creating token-level hallucination detectors which rely on negative (i.e., hallucination) data augmentation to train.

In terms of improving faithfulness and factuality, researchers did not find natural language inference (NLI)[4] models trained on standard NLI datasets to be robust enough for summarization-related downstream tasks (Falke et al., 2019). Contemporaneously, entity chains are used to explicitly ground the generations so that they become more faithful (Narayan et al., 2021). More broadly, there have been other recent works striving to develop techniques for high-precision generation (Malmi et al., 2019; Tian et al., 2019; Pang and He, 2020; Parikh et al., 2020; Dušek and Kasner, 2020).

3 AgreeSum Task and Datasets

3.1 Task

Short description of AgreeSum. The input is a cluster of around four articles (refer to Section 3.2 for more details) that describe the same event. However, the articles may have different levels of details and/or different levels of neutrality; e.g., one article in a cluster may be an opinion, while other articles may be neutral news. The expected output is an abstractive summary that represents information common and faithful to all input articles in the cluster. Moreover, the summary needs to be informative.

3.2 English AgreeSum Dataset Based on WCEP

Step 1. Recall that WCEP contains neutral human-written news summaries, each of which is linked to a news article. The first step is to obtain the summaries and 8 on-topic articles[5] for each summary based on WCEP. Specifically, we collect 5564 human-written summaries $\{\boldsymbol{y}_i\}_{i=1}^{5564}$ from WCEP. For each summary, we have one linked article to each summary on WCEP; we call the set of such articles $\{\boldsymbol{x}_i^{(0)}\}_{i=1}^{5564}$. Given that we want to generate abstractive summaries, and to make the dataset challenging, the set of articles $\{\boldsymbol{x}_i^{(0)}\}_{i=1}^{5564}$ will *not* be used in the final dataset, to prevent excessive textual overlaps between the input articles and the target summaries. For each i, we obtain 8 other news articles $\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \ldots, \boldsymbol{x}_i^{(8)}$ that are semantically similar to $\boldsymbol{x}_i^{(0)}$, based on a proprietary BERT-based clustering algorithm.

Step 2. The second step is to annotate entailment relations. Annotators are asked to judge if each of the articles in a cluster entails the summary (i.e., "does the article contain all the information presented in the summary?"). 1025 *cluster*-summary pairs are annotated. We designate $\sim$10% of the annotated clusters to be in the dev set. This step is discussed further in Section 4.

Step 3. To make AgreeSum moderately but not overly difficult, we set the maximum number of articles per cluster to be four, and the next step is to transform the dataset to have four articles per cluster. To take advantage of entailment annotations, we duplicate each annotated cluster in the training

---

[4] The NLI datasets, beginning with SNLI (Bowman et al., 2015), correspond to a three-class (entailment, neutral, contradiction) entailment classification task.

[5] A small amount of clusters (233, or $\sim$4%) have fewer than 8 articles to ensure relevance of all articles in a cluster.

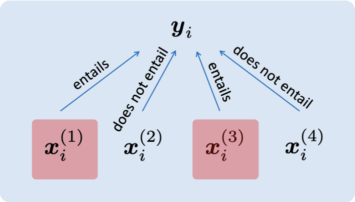

Figure 1: An annotated example (i.e., input-output pair where the input is a cluster of articles and the output is a summary) in the training set. In the figure, two articles in the input cluster entail the summary while the other two do not. An example is either annotated (meaning all article-summary pairs in the cluster-summary pair are annotated) or unannotated (meaning none of the article-summary pairs are annotated). Note that in the dev set, all articles entail the summary given that we would like to compare between generated summaries and the gold summaries.

set $\boldsymbol{c}_i = \{\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \ldots, \boldsymbol{x}_i^{(8)}\}$ ten times so that there is roughly an equal number of annotated and unannotated clusters.[6] Next, for each cluster, we randomly choose four articles to keep in the cluster, and discard the rest.

Dev and test sets. For the development set, we aim to designate the WCEP summaries as gold summaries. Therefore, for each cluster, we only keep the articles that entail the corresponding summary based on human annotation. In the case that a cluster has > 4 articles that entail the summary, we split the cluster into two, such that each newly formed cluster has ≥ 2 articles that entail the summary.

We use WCEP entries corresponding to dates before August 2019 for the training and development sets; for the test set, we sample 150 clusters of articles containing WCEP linked articles published between August 2019 to August 2020 to ensure that the test set does not have overlapping articles or overlapping publication times with the training and development sets.

For each sampled test cluster, we sample four articles from the cluster. Unlike the development set, these four articles are not annotated and are not guaranteed to entail any particular common summary. As a result, the provided WCEP summaries in the test set are **not** gold-standard summaries.

The test set is expected to be challenging due to construction methods, as well as the shift in topics of recent events (e.g., the COVID-19 pandemic). The test set WCEP summaries are not gold-standard summaries, but are nevertheless provided given the potential use to approximate generation informedness (Section 6).

We thus obtain the dataset split shown in Table 1.

	# of cluster-summary pairs			# of article-summary pairs		
	train	dev	test	train	dev	test
all	18208	132	150	70137	423	600
annotated	9130	132	0	33841	423	0
(at least 1) entailed	7610	132	0	17951	423	0
unannotated	9078	0	150	36296	0	600

Table 1: Dataset information. A cluster contains $\leq$ 4 articles. The "(at least 1) entailed" row refers to the number of *annotated* clusters containing at least one article that entails the summary (in the cluster-summary case), or the number of articles that entail the summary *among the annotated pairs* (in the article-summary case).

4 Article-Summary Entailment

4.1 The Entailment Dataset

Workers have annotated the entailment relation of 1025 cluster-summary pairs (a subset of the cluster-summary pairs obtained in Step 1 in Section 3.2), which correspond to 7750 article-summary pairs. For each article-summary pair, we ask *five* professionally trained annotators, randomly chosen from a pool of ~800 raters, whether the summary is semantically entailed in the article; we then take the majority answer. See the appendix for more details.

Our AgreeSum models leverage this entailment dataset, as discussed in Section 5. In addition, this dataset could be seen as a challenging RTE-style (or two-class NLI-style) task.

4.2 Model Performance

To explore how models perform on the article-summary entailment dataset, and to see whether we can have a good entailment-classification model to guide the learning of our summarizer that encourages faithfulness to all input articles in the same cluster, we treat the dataset as a two-class NLI task where the label is either "entailed" or "not entailed," and examine the classification accuracy. Specifically, we evaluated the following three models. RoBERTa-large (Liu et al., 2019b) fine-tuned on MNLI fails on our entailment data[7], likely in

---

[6] A side effect is that such choices could allow for more supervised cross-entropy updates.

[7] In this case, the majority of the predictions are "not entailed."

part because the premises in MNLI are sentences but are articles in our article-summary entailment task. It is also worth noting that unlike T5 (Raffel et al., 2020), RoBERTa-large is not pretrained on a multi-task mixture of many tasks, with each task converted into the text-to-text format.

We also attempt a model that integrates the PEGASUS-encoder which is pretrained in the news domain using the HugeNews corpus which contains around 1.5 billion articles. Given that pretrained PEGASUS does not include `[CLS]` tokens and it is very expensive to re-pretrain, we use a CNN-based classifier (Kim, 2014) whose input is the PEGASUS-encoder-outputs; more specifically, the convolutional layers pool over the sequence of encoder outputs. Based on the architecture by Kim (2014), our CNNs use filter n-gram sizes of 2, 3, 4, and 5, with 256 filters each. The resulting classifier achieves $\sim$68% accuracy on our entailment dataset.[8]

In comparison, T5 (Raffel et al., 2020) shows encouraging results. The multi-task-trained T5-large fine-tuned on our training set achieves 81.3% accuracy (79.1% for vanilla T5-large). T5-small fine-tuned on our training set achieves 79.5% accuracy (76.4% for vanilla T5-small).

We aim to see if using an entailment signal from a moderately good article-summary entailment classification model would help produce summaries that satisfy the AgreeSum criteria.

5 AgreeSum Baselines and Approaches

5.1 Notations and Baselines

We first provide some notations so as to allow easier discussion. Suppose the clusters of articles in the training set are denoted by $\{\boldsymbol{c}_i\}_{i \in N}$. Each cluster contains at most four articles: $\boldsymbol{c}_i = \{\boldsymbol{x}_i^{(j)}\}_{j \leq 4}$, with the summary $\boldsymbol{y}_i$.

For cluster i, let $\boldsymbol{e}_i$ be the set of indices that correspond to articles that entail the summary. For example, in Figure 1 which corresponds to the ith cluster, we have $\boldsymbol{e}_i = \{1, 3\}$. Let E and D denote the encoder and the decoder of PEGASUS, respectively.

The following baselines are based on PEGASUS pretrained using gap-sentence prediction on HugeNews with 1.5 billion articles (3.8 terabytes). Note that despite being an SDS model, PEGASUS also achieves near-SoTA results on the Multi-News MDS dataset (Fabbri et al., 2019), so it is a competitive baseline for MDS as well.

B1: finetuning on $(\boldsymbol{x}_i^{(0)}, \boldsymbol{y}_i)$ pairs. Recall that for each summary $\boldsymbol{y}_i$, we drop the WCEP-linked article $\boldsymbol{x}_i^{(0)}$ to prevent excessive textual overlaps between the input cluster and the output summary. In this baseline, however, we use $\boldsymbol{y}_i$'s in the training set and its corresponding $\boldsymbol{x}_i^{(0)}$'s (5452 pairs) for supervised finetuning.

Why can we use $\boldsymbol{x}_i^{(0)}$ as gold targets? On WCEP, it is reasonable to assume that the summaries are directly connected to the linked articles, and therefore, $\boldsymbol{x}_i^{(0)}$ entails $\boldsymbol{y}_i$. Moreover, these summaries also inform the model of the style of the WCEP summaries. However, the downside is that the model could potentially prioritize extractions from the articles over entailment (i.e., the property that the generated summary is entailed in each article), as we would see in Section 6.

B2: concatenating truncated inputs. We finetune PEGASUS on the following: for each cluster $\boldsymbol{c}_i$, we truncate $\{\boldsymbol{x}_i^{(j)}\}_j$ for each $j \in \boldsymbol{e}_i$ and concatenate them such that the concatenated sequence has length ≤ 1024, given hardware constraints. We use a special symbol to delineate article boundaries.

B3: B1+B2. We first train using B1 and then finetune using B2, which may improve over B1 or B2 alone.

B4: merging encodings and decode. Inspired by Chu and Liu (2019), we first encode $\boldsymbol{x}_i^{(j)}$ separately for each $j \in \boldsymbol{e}_i$, and pass the average of the encodings to the decoder (i.e., $\frac{1}{|\boldsymbol{e}_i|}\sum_{j \in \boldsymbol{e}_i} E(\boldsymbol{x}_i^{(j)})$). Next, we do supervised learning based on the WCEP summaries.

B5: best lead-1 sentence by entailment score. We first extract the first sentence of each article in the cluster. Next, we rerank the sentences using an entailment score, which is the mean of the binary entailment labels, predicted by fine-tuned T5-large (Section 4), between each article in the cluster and the given sentence. We choose the summary corresponding to the highest score, breaking ties randomly.

---

[8] Doing intermediate training on MNLI (Pruksachatkun et al., 2020) and then training on our entailment dataset, unfortunately, does not strengthen performance.

5.2 AgreeSum Model (ASM): Leveraging Unannotated Data

We introduce another baseline model called ASM. The distinguishing feature is that to target cluster-summary entailment (meaning the summary is entailed in each article in the cluster), we attempt to use the T5-entailment-classification results (discussed in Section 4) as a training signal.

Supervised pretraining and training. Given pretrained PEGASUS, we first fine-tune according to B1. Next, for each cluster, we concatenate all elements of $\boldsymbol{c}_i$ similar to B2, and mask out the articles that do not entail $\boldsymbol{y}_i$ by the padding symbol; effectively, we use $\{\boldsymbol{x}_i^{(j)}\}_{j \in \boldsymbol{e}_i}$ as input to the transformer. Then, we use the standard cross-entropy loss L_{ce} to train the summarizer. However, only a small number of clusters are annotated, providing a very limited supervised signal.

Unsupervised entailment loss. We complement L_{ce} with the entailment loss L_e to learn entailment behavior. We fine-tune T5-small on our training dataset to predict entailment, and use this as our entailment classifier F_e. In practice, we obtain T5-outputs using remote procedure calls (RPCs).

F_e takes in an article $\boldsymbol{x}$ and a summary $\boldsymbol{y}$ as inputs, and outputs 1 if $\boldsymbol{x}$ entails $\boldsymbol{y}$ and -1 otherwise. The loss L_e is based on policy gradients (Williams, 1992; Sutton et al., 2000); we aim to maximize a sequence-level metric of a summary decoded from the model during training: $\mathbb{E}_{\tilde{\boldsymbol{y}} \sim p_\phi} \sum_{k=1}^{4} F_e(\boldsymbol{x}_i^{(k)}, \tilde{\boldsymbol{y}})$ where ϕ stands for the parameters of the encoder E and the decoder D. We thus have the following gradient:

$$\nabla_\theta L_e(\phi) = -\mathbb{E}_{\tilde{\boldsymbol{y}} \sim p_\phi} \nabla_\phi \log p_\phi(\tilde{\boldsymbol{y}}) \hat{Q}(\tilde{\boldsymbol{y}}),$$

where

$$\hat{Q}(\tilde{\boldsymbol{y}}) = \sum_{k=1}^{4} F_e(\boldsymbol{x}_i^{(k)}, \tilde{\boldsymbol{y}})$$

is the sequence-level return. Intuitively, during training, we sample a summary $\tilde{\boldsymbol{y}}$ from our model, and run it through the T5 entailment classifier (which is separate from our summarizer) to obtain $\hat{Q}(\tilde{\boldsymbol{y}})$. We then weight the MLE gradient (taking $\tilde{\boldsymbol{y}}$ as the target) by $\hat{Q}(\tilde{\boldsymbol{y}})$. L_e thus aims to guide our summarizer to generate summaries that entail in all or most of the articles.

Unsupervised language loss. Only using the entailment loss L_e may result in degenerations. One way to encourage the model to generate fluent summaries and summaries that look like WCEP summaries is by using GAN-style (Goodfellow et al., 2014) objectives, which have achieved good performance in some conditional generation tasks like textual style transfer and machine translation (Shen et al., 2017; Wu et al., 2018; Pang and Gimpel, 2019). We thus use F_l, a "language classifier" (i.e., discriminator) that distinguishes model generations from real dataset summaries (no matter annotated or not), to force our summarizer to generate sentences that look like the human-written summaries. F_l is based on the CNN settings by Kim (2014) identical to the setting introduced in Section 4, while the inputs to CNNs are sentence representations obtained using PEGASUS.

Specifically, suppose there are k examples in a minibatch, then

$$L_l = -\frac{1}{k} \sum_{i=1}^{k} \left[\log F_l(\boldsymbol{h}_i) + \log(1 - F_l(\tilde{\boldsymbol{h}}_i)) \right],$$

where $\boldsymbol{h}_i$ is the decoder hidden states of WCEP summaries, and $\tilde{\boldsymbol{h}}_i$ is the decoder hidden states of model generations, by professor forcing (Lamb et al., 2016). In the professor forcing algorithm, the input to F_l is the hidden states instead of hard tokens so as to address the mismatch between training-time sequence prefix and test-time sequence prefix.

Summary. We alternate updates among the following:

(1) supervised training: updating E, D to minimize the loss L_{ce};

(2) unsupervised training: updating E, D to minimize the loss $L_e - \lambda L_l$;

(3) language classifier training: updating F_l to minimize the loss L_l.

5.3 Implementation

We implement our model as a fork of the open-source 568M-parameter PEGASUS model (Zhang et al., 2020a).[9] We initialize PEGASUS from the "mixed & stochastic" checkpoint, which was pretrained for 1.5M steps.

All baseline models are fine-tuned with a learning rate of 1e-4. All of the ASM models use 5e-5

---

[9] `https://github.com/google-research/pegasus`

model	ROUGE-dev 1	ROUGE-dev 2	ROUGE-dev L	ROUGE-test‡ 1	ROUGE-test‡ 2	ROUGE-test‡ L	article-summary entail % (↑) ¶	cluster-summary entail % (↑) ¶	hallucination % (↓) ¶	language % (↑)	n-gram overlap % (↓) $n = 3$	$n = 4$	$n = 5$	$n = 6$
B1	45.7	25.0	38.3	30.7	12.3	24.2	56.8 / 54.0	22.7 / 20.1	6.0 / 6.0	90.0	60.4	49.8	42.6	38.1
B2	46.1	23.5	37.9	28.5	10.4	22.0	50.8 / 49.0	23.3 / 22.2	18.7 / 19.5	92.0	27.8	13.5	7.3	4.1
B3	47.1	23.7	38.1	29.1	10.8	22.4	49.7 / 50.7	22.7 / 23.3	16.0 / 20.0	96.0	26.1	12.8	6.9	4.0
B5	33.9	13.6	26.1	7.70	2.59	5.39	58.8 / 65.1	22.0 / 27.3	0.0 / 0.0	93.3	100.0	100.0	100.0	100.0
ASM	44.4	22.8	37.0	27.5	11.0	22.4	62.8 / 66.0	30.7 / 39.1	10.0 / 8.8	96.0	47.3	36.4	30.2	26.1

Table 2: Results (on test set if not specified). ¶: the first number corresponds to T5-large-evaluated results, and the second number corresponds to human-evaluated results. For each row, human raters annotated the entire test set; each article-generation pair is annotated by three raters, and we take the majority answer. The best result in each column is in blue; the worst in red. n-gram overlap: the proportion of generation n-grams that are also in the source. ‡: the test set WCEP summaries should not be treated as references, given that there is no guarantee that the WCEP summary is entailed in each of the articles; the test set WCEP summaries are provided as an approximate measure of informedness. Note: B4 results in extensive hallucinations and very low ROUGE ($\sim$10), and the ablation study of ASM without L_l produces heavy degenerations, so they are omitted.

(tuned in {1e-5, 5e-5, 1e-4, 2e-4}). All models use beam search for decoding (beam size 8, beam alpha 0.8). Given hardware constraints, all models use a max input length of 1024. The max output decoding length is set to be 128. In addition, we tune $\lambda \in \{0.05, 0.1, 0.3, 0.5, 1, 2\}$ and choose $\lambda = 0.1$ for all reported experiments that include the L_l. We do not change any other default hyperparameter settings adopted from PEGASUS. Please refer to the appendix for more details.

6 Results

The following two models are not included in the table given their poor performance. (1) The ablation study of ASM without language loss does not produce meaningful outputs. In this case, given that the T5 entailment classifier does not encourage language quality, the summaries in fact degenerate heavily. (2) Model B4 (merging encodings and decode) is omitted from the table given very poor ROUGE performance ($\sim$10) and extensive hallucination. Our conjecture is that the mean of the encodings of articles does not correspond to a meaningful encoding in the case of PEGASUS.

6.1 Agreement and Hallucination

First, we claim that ASM achieves better agreement. There are two types of agreement: *article-summary* agreement which is the proportion of summaries entailed in the articles, and *cluster-summary* agreement which is the proportion of clusters in which *all* articles entail the summary. Table 2 reports both the T5-automatic evaluation and the human evaluation results on agreement.

(Preliminary) automatic metrics. Given an article-summary pair, we use T5-large fine-tuned on our entailment dataset (Section 4) to predict whether the summary is entailed in the article. For article-summary agreement, we compute the percentage of "entailed" classifications. For cluster-summary agreement, we compute the percentage of clusters where all article-summary pairs lead to "entailed" classifications. We see that the T5-evaluation results for ASM models perform better than the respective results for baseline models.

Human evaluation. For each row of Table 2, human raters annotated the entire test set (150 clusters, which corresponds to 600 article-generation pairs), on whether the generated summaries are entailed in the article. Workers were asked:

> *Does the article contain all the information presented in the summary?*

The full prompt is available in the appendix. One design choice is that we merge all article-summary pairs for each cluster together into one task/HIT. Therefore, each task/HIT corresponds to four article-summary annotations. To reduce the inherent variance in human evaluation, each article-generation pair is annotated by three different raters, and we take the majority answer. Please see the appendix for more details.

For article-summary agreement, we see that the human evaluation for the ASM models performs a little better than the extractive results (B5). ASM models perform more than 10 points better than the best abstractive baseline results (B1), which is in turn $\sim$5 points better than the other abstractive baseline results (B2, B3). For cluster-summary agreement, this improvement is even clearer, with ASM models performing more than 10 points better than any baseline.

In addition, we see that ASM reduces hallucination compared to B2 and B3. One way to approximate hallucination is by the number of clusters in which none of the articles entail the generated summary. Using both automatic and human evaluation results, we see that our model does better than B2 and B3, but a little worse than B1 which copies extensively from the source articles.

6.2 Discussion on ROUGE

In SDS tasks, Durmus et al. (2020) and Wang et al. (2020a) observe that ROUGE and BERTScore have a small correlation with summary factuality.

A hallucination or non-entailment[10] can have major text-span overlaps with the reference, thereby having a large ROUGE score. Given the nature of AgreeSum, and by Table 2, we confirm that high ROUGE does not imply entailment and should not be considered heavily in evaluation.

On the other hand, intuitively, we do recognize that an overly small ROUGE may indicate bad generations (e.g., extremely short generations, off-topic generations, and other degenerations like repetitions), which is the case for B4 generations as well as ASM-minus-L_l generations.

Thus, practitioners need to rely on and determine the desired tradeoff between the following two automatic metrics: (1) ROUGE as a coarse proxy for summary quality and informedness, and (2) entailment-related and hallucination-related metrics (Section 6.1).

As a reminder, the development set summaries can be treated as gold-standard summaries. However, the test set summaries are not gold-standard summaries; they are only provided so as to allow one way to measure generation quality and informedness. Unlike the development set, none of the test-set input articles are filtered out through the summary-article entailment annotation procedure, given that we do not want to introduce potential bias through too much manipulation and filtering on raw test clusters.

6.3 More Observations

Language. We also asked workers to judge the language of the generations:

> *Is this summary coherent and well-written with no self-contradictions or capitalization, spelling, punctuation, or grammar errors?*

The full prompt is in the appendix. We see that the ASM-generated summaries are marginally better at the above. However, ASM without language loss results in heavy degenerations. On a separate note, we see that ASM generations tend to copy more than B2 and B3, but less than B1.

Examples. The appendix contains a few examples that compare generations from different models. For example, in Table 5, given four articles, we see the different generations that the systems produce. Article 4 is an opinion piece. ASM model correctly abstractively summarizes Article 2 and 4 in a way that agrees with 1 and 3. B1 copies from Article 1, while B2 and B3 have hallucinations. Given space constraints, please refer to the appendix.

6.4 Extension: Post-hoc Entailment Reranking after Decoding a Beam

To generate summaries that achieve better agreement, we also attempted a decoding trick which we name as entailment-oriented decoding, denoted by entdec in Table 3.

We first define an entailment score used in this case. Given a cluster of articles and a generated summary, the entailment score is the mean of the T5-large-predicted binary labels (1 corresponds to "entailed" and 0 corresponds to "not entailed").

A model is suffixed as X-entdec$k$ if we decode from X using beam search with beam size k; and after obtaining the size-k beam, we select the generation that corresponds to the largest T5 entailment score. We pick the beam with the largest score, using the original beam probabilities to break ties. Intuitively, this trick picks the best-entailment summary locally given that the generations in the same beam are usually similar.

Table 3 shows that the entdec-trick generations indeed achieve higher article-summary agreement and human-summary agreement. We see that ASM still maintains the advantage in agreement, even if compared to entdec-decoded generations from other baselines.

6.5 Discussion: Improving Entailment Using T5-Based NLI-Style Models

Falke et al. (2019) find that NLI models trained on

---

[10] Non-entailed summaries are not necessarily hallucinations, given that the non-entailed summaries could correspond to some articles in the cluster but not the rest of the articles.

model	ROUGE-dev 1	2	L	ROUGE-test‡ 1	2	L	article-summary entail % (↑) ¶	cluster-summary entail % (↑) ¶	hallucination % (↓) ¶	language % (↑)	n-gram overlap % (↓) 3	4	5	6
B1-entdec8	45.7	24.8	38.3	29.5	12.0	23.4	60.8 / 56.8	26.0 / 26.2	4.67 / 6.0	93.3	61.1	50.7	43.5	38.8
B2-entdec8	46.1	23.3	38.0	28.5	10.4	22.0	53.2 / 58.3	23.3 / 34.2	18.0 / 18.1	95.3	28.1	13.8	7.4	4.1
B3-entdec8	47.2	23.8	38.1	28.9	10.6	22.1	53.7 / 59.3	24.7 / 33.6	15.3 / 13.4	94.0	26.7	13.0	6.9	4.0
ASM	44.4	22.8	37.0	27.5	11.0	22.4	62.8 / 66.0	30.7 / 39.1	10.0 / 8.8	96.0	47.3	36.4	30.2	26.1
ASM-entdec8	44.8	23.4	37.7	26.9	11.0	22.1	68.2 / 63.3	40.0 / 37.0	8.0 / 10.9	91.3	46.9	35.4	29.1	24.6
ASM-entdec16	44.7	23.1	37.5	26.3	10.8	21.5	70.5 / 63.8	42.7 / 39.9	8.0 / 11.9	90.0	48.7	36.9	30.3	25.6

Table 3: Results using the entdec decoding strategy. The results correspond to the test set performance, if not specified. ¶: the first number in each cell corresponds to T5-evaluated results, and the second corresponds to human-evaluated results. Given that B5 relies on pure extraction, the entdec methods are not applicable. ‡: the test set WCEP summaries should not be treated as references, given that there is no guarantee that the WCEP summary is entailed in each of the articles; the test set WCEP summaries are provided as an approximate measure of informedness.

standard NLI datasets do not offer robust benefits to improving summarization factuality. Maynez et al. (2020), on the other hand, rerank four summaries generated by four different models using BERT-based MNLI models, and find small improvements in faithfulness and factuality. However, these works rank different summaries after decoding assuming an existing summarizer (similar to our entdec trick), instead of updating the model parameters directly. Our contribution lies in the fact that we successfully use entailment models to improve the model during training time.

The major feature of our T5-based NLI model is that (1) our NLI model is based on multi-task-pretrained T5, implying that pretrained T5 can already handle article-length inputs well in certain tasks, and (2) our model is obtained after finetuning on our article-summary entailment dataset. Therefore, our T5-based NLI model is much better adjusted to the length of the premises (given that traditional NLI tasks correspond to sentence-level entailment, but our case corresponds to article-summary entailment). We thus see that using simple T5-based binary signals can successfully improve entailment. However, more complicated modeling may be necessary if the AgreeSum cluster size becomes much larger.

7 Conclusion

We discuss the AgreeSum task with its dataset, and a range of baseline models. AgreeSum is timely given the recent focus on summarization faithfulness. In fact, we show that the summaries produced by several powerful pretraining-based baseline models are not able to follow AgreeSum's requirements satisfactorily. We welcome the community to contribute more advanced methods that work well on AgreeSum, especially when only a small subset of the dataset is labeled with article-summary entailment information.

Within the AgreeSum dataset, we also provide article-summary entailment annotations on a subset of clusters, which we hope can contribute to the recent effort in improving abstractive summarization faithfulness.

Moreover, while there is contemporaneous development of complex approaches to encourage generated abstractive summaries to be entailed in the source articles, we show that it is feasible to improve the entailment behavior of generated summaries based on a binary article-summary entailment classifier.

Acknowledgement

The authors would like to thank Pepa Atanasova, He He, Abe Ittycheriah, Jialu Liu, Tianqi Liu, Ji Ma, Shashi Narayan, Alicia Parrish, Jiaming Shen, Gonçalo Simões, and You (Will) Wu (alphabetical order) for the valuable discussions, and the anonymous reviewers for the detailed reviews.

References

Diego Antognini and Boi Faltings. 2020. GameWikiSum: a novel large multi-document summarization dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6645–6650, Marseille, France. European Language Resources Association.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. SEQ^3:

Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 673–681, Minneapolis, Minnesota. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. MeanSum: A neural model for unsupervised multi-document abstractive summarization. volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232, Long Beach, California, USA. PMLR.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of the document understanding conference*, volume 2005, pages 1–12.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Ondřej Dušek and Zdeněk Kasner. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137, Dublin, Ireland. Association for Computational Linguistics.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. 2020. A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online. Association for Computational Linguistics.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Xiaotao Gu, Yuning Mao, Jiawei Han, Jialu Liu, You Wu, Cong Yu, Daniel Finnie, Hongkun Yu, Jiaqi Zhai, and Nicholas Zukoski. 2020. Generating representative headlines for news stories. In *Proceedings of The Web Conference 2020*, pages 1773–1784.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems*, pages 4601–4609.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Peter J. Liu, Yu-An Chung, and Jie Ren. 2019a. Summae: Zero-shot abstractive text summarization using length-agnostic auto-encoders. *arXiv preprint arXiv:1910.00998*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Yuning Mao, Yanru Qu, Yiqing Xie, Xiang Ren, and Jiawei Han. 2020. Multi-document summarization with maximal marginal relevance-guided reinforcement learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1737–1751, Online. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simoes, and Ryan McDonald. 2021. Planning with entity chains for abstractive summarization. *arXiv preprint arXiv:2104.07606*.

Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. 2018. Abstractive unsupervised multi-document summarization using paraphrastic sentence fusion. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1191–1204, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Bruce Jay Nelson. 1981. Remote procedure call.

Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the tac 2011 summarization track: Guided task and aesop task. In *Proceedings of the Text Analysis Conference (TAC 2011), Gaithersburg, Maryland, USA, November*.

Richard Yuanzhe Pang and Kevin Gimpel. 2019. Unsupervised evaluation metrics and learning criteria for non-parallel textual transfer. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 138–147, Hong Kong. Association for Computational Linguistics.

Richard Yuanzhe Pang and He He. 2020. Text generation by learning from off-policy demonstrations. *arXiv preprint arXiv:2009.07839*.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Over Paul and Yen James. 2004. An introduction to duc-2004. In *Proceedings of the 4th Document Understanding Conference (DUC 2004)*.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6830–6841. Curran Associates, Inc.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020a. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020b. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.

Kexiang Wang, Baobao Chang, and Zhifang Sui. 2020c. A spectral method for unsupervised multi-document summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 435–445, Online. Association for Computational Linguistics.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Lijun Wu, Yingce Xia, Fei Tian, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. Adversarial neural machine translation. volume 95 of *Proceedings of Machine Learning Research*, pages 534–549. PMLR.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Chunting Zhou, Jiatao Gu, Mona T. Diab, Paco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation.

Markus Zopf. 2018. Auto-hMDS: Automatic construction of a large heterogeneous multilingual multi-document summarization corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

A Appendix

A.1 Human Evaluation

Recall that human evaluation results are reported in Table 2 and Table 3. Specifically, we asked human raters to annotate the entailment relationship of each article-generation pair. Each row in the table corresponds to a model, and for each model, human raters annotated the entire test set that contains 150 clusters (which corresponds to 600 article-generation pairs). Moreover, to reduce the variance of human evaluation and to improve the confidence to our claims, each article-generation pair is annotated by three different raters, and we take the majority answer.

Our prompts to human raters are described as follows. We merge all article-summary pairs for each cluster together into one task/HIT. Therefore, each task/HIT corresponds to four article-summary annotations.

Instruction at the top of the annotation page: "In this task you will be given one summary of a news story and the text of four news articles. You will be asked to evaluate whether the summary is coherent and well written with no self-contradictions or capitalization, spelling, punctuation, or grammatical errors. Furthermore, for each news article, you will be asked to evaluate: whether the article contains ALL the information presented in the summary. This is equivalent to asking, 'Using only the text of this news article, would it be possible for someone to write this summary?"'

Language: "Is this summary coherent and well-written with no self-contradictions or capitalization, spelling, punctuation or grammar errors? Select "Yes" if this is a coherent summary written in fluent English with perfect grammar and style. Select "No" if the summary contains one or more capitalization, spelling, punctuation, or grammatical errors, or is incoherent, self-contradictory, or otherwise badly written."

Entailment: "Does the article contain all the information presented in the summary?"

A.2 Entailment Annotation in Dataset Creation

We used the same entailment prompt described in Section A.1 to obtain entailment information for a subset of the clusters. The difference is that we used five annotators per article-summary pair instead of three, to ensure the quality of the supervised split of the training set as well as the entire development set.

A.3 More on Reproducibility

We implement our model as a fork of the open-source 568M parameter PEGASUS model.[11]

Baseline models were trained for 20k steps, with the best checkpoint selected based on dev-R2. Final selections are B1 at 3k steps, B2 at 6k steps, and B3 at 4k steps. Proposed model was trained until convergence based on T5-small entailment scores reported during the sampling steps in the policy gradients REINFORCE algorithm. Final selection of this model was at 43k steps.

Runtime details. Given the chosen PEGASUS implementation, our models are trained on TPUs. Given hardware constraints, F_e, implemented via T5-small fine-tuned on our annotated entailment dataset, is served on CPU on a separate machine. In fact, we run 80 replicas of this machine to improve throughput. We make RPC calls (Nelson, 1981) to this cluster of T5-serving machines during policy gradients training (using the f.contrib.rpc module). Baseline models took approximately 24 hours each to train, with the proposed model taking approximately 4 days. Our conjecture is that for the latter, most of the time is spent on the communication between PEGASUS and T5, as well as the expensive computation of T5 on CPU.

A.4 Examples

We now provide some example generations of the AgreeSum task. Given that the input articles are long, we comment out parts of the articles in the tables. Please refer to Table 4, Table 6, and their captions.

For example, in Table 5, Article 4 is an opinion piece. ASM model correctly abstractively summarizes Article 2 and 4 in a way that agrees with 1 and 3. B1 copies from Article 1, while B2 and B3 have hallucinations. In Table 4, all models except for B1 perform well here. B1 entails Article 1 and 2, but not 3 and 4. Similar comparisons can be made in Table 6.

---

[11] `https://github.com/google-research/pegasus`

ASM: Sérgio Moro resigns as Brazil's Justice Minister after accusing President Jair Bolsonaro of interfering in the country's federal police.

B1: Brazil's Supreme Court authorises a police investigation into President Jair Bolsonaro.

B1+entdec8: Brazil's Supreme Court authorises a police investigation into President Jair Bolsonaro.

B2: The Minister of Justice of Brazil, Sérgio Moro, resigns after accusing President Jair Bolsonaro of interfering in the operations of the federal police.

B2+entdec8: The Minister of Justice of Brazil, Sérgio Moro, resigns after accusing President Jair Bolsonaro of interfering in the operations of the federal police.

B3: The Minister of Justice of Brazil, Sérgio Moro, resigns after accusing President Jair Bolsonaro of interference in the federal police.

B3+entdec8: The Minister of Justice of Brazil, Sérgio Moro, resigns after accusing President Jair Bolsonaro of interference in the federal police.

B5: Brazil's government has been plunged into turmoil after the resignation of one of Jair Bolsonaro's most powerful ministers sparked protests, calls for the president's impeachment and an investigation into claims he had improperly interfered in the country's federal police.

Article 1: [Link: https://www.theguardian.com/world/2020/apr/24/justice-ministers-sacking-plunges-brazil-into-turmoil; the article is not duplicated due to length and copyright]

Article 2: [Link: https://www.ft.com/content/62d04bb5-6825-41ec-b263-4ceeaec58049; the article is not duplicated due to length and copyright]

Article 3: [Link: https://www.theguardian.com/world/2020/apr/26/bolsonaro-in-fresh-crisis-over-sons-alleged-links-to-fake-news-racket; the article is not duplicated due to length and copyright]

Article 4: [Link: https://www.thedailybeast.com/brazils-justice-minister-sergio-moro-quits-accuses-president-jair-bolsonaro-of-misconduct-resigns; the article is not duplicated due to length and copyright]

Table 4: Example generations from test set. All models except for B1 perform well here. B1 entails Article 1 and 2, but not 3 and 4.

ASM: The NFL announces that it will play during the coronavirus pandemic.

B1: NFL Commissioner Roger Goodell sends a letter to fans outlining the league's plans to play during the coronavirus pandemic.

B2: The NFL cancels the remainder of the 2020 season due to the coronavirus outbreak.

B3: The NFL cancels the remainder of the 2019 preseason due to the ongoing coronavirus outbreak.

B5: NEW YORK (AP) — NFL Commissioner Roger Goodell has sent a letter to fans outlining the league's plans to play during the coronavirus pandemic.

Article 1: [Link: https://apnews.com/article/nfl-sports-virus-outbreak-health-football-c10944a1b88bd593198b660d207c7b56; the article is not duplicated due to length and copyright]

Article 2: [Link: https://www.washingtonpost.com/sports/2020/07/27/nfl-cautiously-optimistic-despite-mlb-coronavirus-outbreak/; the article is not duplicated due to length and copyright]

Article 3: [Link: https://www.wxyz.com/sports/roger-goodell-sends-letter-to-nfl-fans-explaining-plans-for-season; the article is not duplicated due to length and copyright]

Article 4: [Link: https://www.wsj.com/articles/nfl-playersand-a-lot-of-new-england-patriotsare-opting-out-of-the-2020-season-11596030916; the article is not duplicated due to length and copyright]

Table 5: Example generations from test set. Note in particular that Article 4 is an opinion piece. The ASM model correctly abstractively summarizes Article 2 and 4 in a way that agrees with 1 and 3. B1 and B5 copy from Article 1, while B2 and B3 have hallucinations. More examples can be found in the appendix.

ASM: Late-night talk shows in New York City begin taping without audience members.
B1: The Late Show with Stephen Colbert, The Tonight Show with Jimmy Fallon, and The Full Frontal with Samantha Bee begin taping without audience members due to the spread of the coronavirus in New York City.
B1+entdec8: The Late Show with Stephen Colbert, The Tonight Show with Jimmy Fallon, and The Full Frontal with Samantha Bee begin taping without audience members due to the spread of the coronavirus in New York City.
B2: In response to the outbreak of coronavirus in the United States, all New York-based late-night talk shows will cease taping without an audience.
B2+entdec8: In response to the outbreak of coronavirus in the United States, all New York-based late-night talk shows will cease taping without an audience.
B3: Following the lead of the Centers for Disease Control and Prevention, all late-night talk shows in New York City will stop taping and start without an audience due to the outbreak of coronavirus. all late-night shows in New York City will stop taping and start without an audience Due to the outbreak of coronavirus, all late-night shows in New York City will stop taping and start without an audience.
B3+entdec8: Following the lead of the Centers for Disease Control and Prevention, all late-night talk shows in New York City will stop taping and start without an audience due to the outbreak of coronavirus. all late-night shows in New York City will stop taping and start without an audience Due to the outbreak of coronavirus, all late-night talk shows in New York City will stop taping and start without an audience Due to the outbreak of coronavirus, all late-night talk shows in New York City will stop filming and start without an audience due to the outbreak of coronavirus.
B5: The New York late-night circuit was the antithesis of "Live in Front of a Studio Audience" this week, prerecording shows without a crowd due to coronavirus fears.
Article 1: [Link: https://www.latimes.com/entertainment-arts/tv/story/2020-03-13/coronavirus-jimmy-fallon-stephen-colbert-no-audience; the article is not duplicated due to length and copyright]
Article 2: [Link: https://variety.com/2020/tv/news/late-night-shows-new-york-coronavirus-1203530972/; the article is not duplicated due to length and copyright]
Article 3: [Link:]https://abcnews.go.com/Entertainment/wireStory/late-night-comics-adjust-shows-audience-69581648; the article is not duplicated due to length and copyright]
Article 4: [Link: https://apnews.com/article/38f1afde2a5676fd3c2377f3719f5c86; the article is not duplicated due to length and copyright]

Table 6: Example generations from test set. ASM is very abstractive and entails all articles. B1 only entails Article 1 and 2. B2 incorrectly says that shows will "cease taping." B3 has repetition issues.

BERT Busters: Outlier Dimensions that Disrupt Transformers

Olga Kovaleva[*1], **Saurabh Kulshreshtha**[*1], **Anna Rogers**[2] **and Anna Rumshisky**[1]
[1]Department of Computer Science, University of Massachusetts Lowell
[2]Center for Social Data Science, University of Copenhagen
[1]`{okovalev,skul,arum}@{cs.uml.edu}`
[2]`arogers@sodas.ku.dk`

Abstract

Multiple studies have shown that Transformers are remarkably robust to pruning. Contrary to this received wisdom, we demonstrate that pre-trained Transformer encoders are surprisingly fragile to the removal of a very small number of features in the layer outputs (<0.0001% of model weights). In case of BERT and other pre-trained encoder Transformers, the affected component is the scaling factors and biases in the LayerNorm. The outliers are high-magnitude normalization parameters that emerge early in pre-training and show up consistently in the same dimensional position throughout the model. We show that disabling them significantly degrades both the MLM loss and the downstream task performance. This effect is observed across several BERT-family models and other popular pre-trained Transformer architectures, including BART, XLNet and ELECTRA; we also show a similar effect in GPT-2.

1 Introduction

Pre-trained Transformer-based models (Vaswani et al., 2017) have become widely popular in a variety of NLP applications. Multiple studies of BERT-family models (Devlin et al., 2019) showed that Transformers are remarkably robust to pruning (Gordon et al., 2020; Prasanna et al., 2020; Chen et al., 2020; Michel et al., 2019). This work presents a different and unexpected result: it is possible to dramatically disrupt the performance of BERT and other Transformer-based architectures by modifying very few weights (less than 0.0001% for BERT).

In particular, we show that there is a very small number of outlier dimensions that regularly appear in the same position in the pre-trained encoder layers of a given Transformer model. We demonstrate that this effect holds for different Transformer-family architectures, including multiple variants of BERT, as well as ELECTRA (Clark et al., 2020), BART (Lewis et al., 2020), and XLNet (Yang et al., 2019). A similar phenomenon is also present in the decoder layers of GPT-2 (Radford et al.). When these dimensions are disabled throughout the model in the concluding transformation of each layer, they can drastically reduce the overall model performance. With the exception of GPT-2, the last transformation in each layer of these models is normalization (LayerNorm), which is what we mainly focus on in this study.

The contributions of this work are as follows:

- We identify certain outlier dimensions in Transformer layer outputs and show that they play a crucial role in both language modeling and downstream task performance. Disabling the weights for these output dimensions drastically degrades performance (up to 44 points).
- We show that this effect holds for the encoder layers of six different models of the BERT family, as well as other popular pre-trained Transformer-based models including ELECTRA, BART, and XLNet. In GPT-2, a similar phenomenon is observed in the output dense transformation of the decoder layers.
- We demonstrate that outlier weights emerge gradually and begin to emerge early in pre-training, causing abnormal spikes at select dimensions in the output embedding vectors.

To our knowledge, this is the first work to establish the presence of very few regular outliers in the output Transformer representations and their importance for the model performance. It is not clear why these features emerge, but the final transformations clearly play a larger role in the Transformer layers than is usually assumed, and this needs further investigation.

---

[*] Authors contributed equally to this work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3392–3405
August 1–6, 2021.

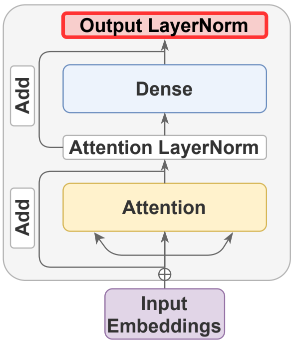

Figure 1: Transformer encoder layer, adapted from (Vaswani et al., 2017).

This paper is organized as follows. After a brief overview of related work (§2), we introduce the methodology for defining, locating, and disabling the BERT outlier weights in §3. In §4.1 and §4.2, we quantify the effect of disabling these weights both in pre-training and in downstream tasks. In §4.3, we demonstrate that other Transformers (BART, ELECTRA, XLNet, and GPT-2) also exhibit similar behavior. §5.1 evaluates magnitude- and position-based criteria for identifying the outlier dimensions and compares them with our proposed criteria. In §5.2, we replicate the outlier effect in a BERT model during pretraining and study its dynamics.

2 Related work

Transformer layer outputs. At a high level, Transformer encoder layers consist of multi-head self-attention followed by a dense layer Vaswani et al. (2017). Most contemporary Transformers use normalization to improve the speed and stability of training.

Usually, the outputs of both self-attention and linear layers undergo the layer normalization transformation (LayerNorm, Ba et al. (2016)). Each LayerNorm transformation is parameterized by a separate set of learned weights (*scaling factors* and *biases*). Xiong et al. (2020) refer to this configuration as *post-LN*. In the *pre-LN* variant adopted by the GPT-2 model, LayerNorm is applied prior to the self-attention or linear transformations instead.

We will refer to the outputs of the final transformation in the encoder layer as *features* and the parameters of this transformation as *weights*. The final transformation is LayerNorm for all models considered in this study except GPT-2, where the last component is a MLP.

Like other normalization techniques, LayerNorm operates in two steps. For a given input x_i of the i-th layer with a hidden dimension m, LayerNorm computes mean and variance across the features:

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij},\ \sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{ij}-\mu_i)^2 \quad (1)$$

The inputs are then normalized and a learnable scale-shift transformation is applied to produce the normalized output embedding:

$$\hat{x}_{ij} = \frac{x_{ij}-\mu_i}{\sqrt{\sigma_i^2+\epsilon}},\ y_i = \gamma \odot \hat{x}_i + \beta \quad (2)$$

where $\gamma \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^m$ are trainable parameters referred to as *scaling factor* and *bias* (shift).

So far there have been few studies of normalization strategies in Transformer architectures and they focused mostly on the description of the training process. Xiong et al. (2020) show that the location of LayerNorm in Transformer affects the gradient flow and demonstrate the need of the warmup stage. Nguyen and Salazar (2019) inject multiple normalization blocks in specific network submodules to improve model performance. More recently, LayerNorm alternatives have been proposed and shown to have better gradient propagation through the network (Xu et al., 2019; Shen et al., 2020).

Overparametrization. After the initial reports of redundancy in the BERT model (Michel et al., 2019; Kovaleva et al., 2019), compressing Transformers quickly became a subfield of its own (Jiao et al., 2020; Zafrir et al., 2019; Fan et al., 2019; Guo et al., 2020). See overviews by Ganesh et al. (2020) and Rogers et al. (2020).

Pruning is a class of methods for model compression which involves setting some of its weights to zeros with minimal performance loss. Much pruning work focuses on compression for the sake of efficiency, but it is also used for model analysis, and that is our goal as well. The most common approach is selecting the weights to be pruned by magnitude (Han et al., 2015).

Some of the recent findings are that the lottery ticket hypothesis (Frankle and Carbin, 2019) holds for BERT: its largest weights do form subnetworks that can be retrained alone to reach the performance close to that of the full model (Prasanna et al., 2020; Chen et al., 2020; Gordon et al., 2020). In structured pruning, the best subnets of BERT's heads and MLPs (selected by importance scores) do not quite reach the full model performance, but the worst ones are still much better than the worst magnitude-based subnets (Prasanna et al., 2020), presumably because they retain a lot of high-magnitude weights.

3 Outlier weights in BERT models

In this section, we introduce the methodology for identifying outlier dimensions and describe our method for disabling these outlier features.

3.1 Identification

To identify the outlier weights in BERT-like models, we consider all the output components in each encoder layer. We compute the mean and standard deviation of the bias and scaling factors of the output LayerNorm. We identify the dimensions where both of these weights are at least 3σ from the mean. Figure 2 illustrates this heuristic. Further, we select the dimensions where this is consistently the case for at least half of the model layers. We refer to these dimensions as outliers.

The described heuristic was used to identify the outlier dimensions in four out of six BERT models we considered: BERT-base, BERT-medium, BERT-small and mBERT.[1] For BERT-large, the deepest model we considered, the frequency constraint was relaxed to 1/3 of the layers. In RoBERTa (Liu et al., 2019), the distribution of the scaling factors was a little different from BERTs, and we relaxed the standard deviation constraint down to 2 sigmas to detect the outliers. In Section 10.1 of Appendix, we report positions of outlier weights identified for all models.

3.2 Disabling

To quantify the effect of the outlier weights on BERT, we disable them and examine how this affects model performance. We set the outlier weights to zeros across all layers and report model performance on a) masked language modeling and b) downstream GLUE tasks (Wang et al., 2018).

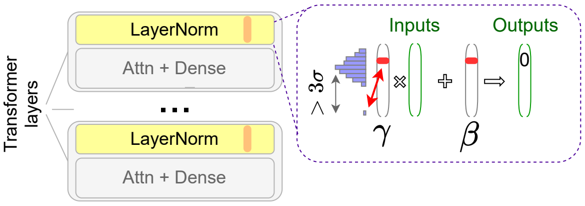

Figure 2: Simplified illustration of our approach to disabling LayerNorm weights. We consistently set the outlier weights γ and β of the output LayerNorm to zeroes, which results in masking of the corresponding features in the output vectors. We repeat the procedure for all of the Transformer layers of the encoder.

Since different model components may affect performance, we also looked at all the parameter vectors and matrices in BERT that have the same dimensionality as the output embeddings. These included key, query and value transformations[2], output LayerNorm, attention LayerNorm, and input embedding layers. In order to examine the effect of these components on model performance, we masked the identified outlier weights of a given component simultaneously across all layers at the same dimensional position. When working with the matrices, we set to zeros the entire row of weights corresponding to the outlier positions across all Transformer layers. Similarly, for LayerNorm, we set the scaling factor and the bias at the outlier position across all Transformer layers to zero. In both cases, this results in "masking" of the output vector's feature at the specified dimension after a forward pass through that layer.

Although the same dimensions repeatedly show up as outliers across different model components, in the preliminary experiments, we found that disabling the weights of the input embedding layers and of the linear layers produced no significant change in performance or in the output embedding space, so we did not pursue this direction further. However, the outlier weights of the output LayerNorm had an unexpectedly large effect on the model, and this is what we focus on in most of our experiments.

3.3 Visualization

As an example, let us consider the two outlier dimensions that the above method identifies for BERT-base-uncased model: `308` [3] and `381`. Fig-

---

[1] BERT-medium and BERT-small come from the official Google repository (https://github.com/google-research/bert), and the other models from the HuggingFace (https://github.com/huggingface/transformers). Interestingly, we discover that the checkpoints of the same BERT-base configuration provided by different repositories (Huggingface vs. Google) have the outliers in different locations; the outliers also have different values: positive vs. negative.

[2] To find outliers in weight matrices, we compute the L_1 norm for each row. The total number of rows is the same as the dimensionality of the layer output embedding (e.g. 768 for BERT-base). From this distribution of row-norms, we select those row indices for which the magnitude is 3σ away from the mean of the distribution.

[3] All dimensions in the paper are zero-indexed.

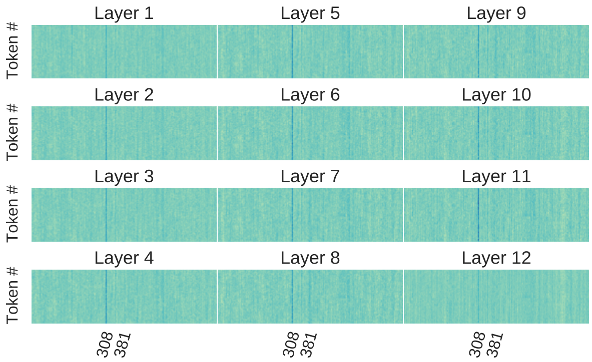

Figure 3: Outlier LayerNorm features `308`, `381` in BERT-base-uncased (randomly sampled input).

ure 3 shows a heatmap of the output embedding of each layer (one pixel per row) for a random passage from WikiText (Merity et al., 2016). Since the output embeddings are produced by LayerNorm, the outlier dimensions with unusually high or low-magnitude weights should be visible in the heatmap.

As seen in Figure 3, dimension `308` consistently produces high-magnitude weights in the output embeddings in most BERT layers; feature `381` shows visibly high values in layers 7-10. The magnitude of a given feature depends both on the LayerNorm scaling factor and the bias (Equation 2). We find that both contribute to the outlier effect, to various degrees in different layers. See Table 13 in the Appendix for statistics on all scaling factors and biases in BERT-base.

Features that show high magnitude throughout the model are expected to distort the resulting embedding space. In a brief experiment, we found that the vector representations of up to 95% of the input tokens from the WikiText corpus have abnormally high magnitudes at the dimensions corresponding to the outlier weights we identified. We found that this embedding distortion is not attributable to the input embedding layer (Layer 0). We did this by manually setting select embedding weights of the three channels of the input embedding layer (token, token type, and position) to zero (along with the weights of the following normalization layer).

The fact that the embedding distribution we observed is not uniform is in line with observations by Ethayarajh (2019) who concluded that BERT embeddings are highly anisotropic and form a cone-like shape in the hidden space. The outlier weights we identify are likely the cause of this, since after they are removed, the embedding space becomes relatively uniform.

Input	Ghostbusters was [released] on June 8 , [1984] , to critical [acclaim] and became a cultural phenomenon . It was well [received] for its deft blend of comedy, [action] , and horror , and Murray ’ s performance was [repeatedly] singled out for praise .
RoBERTa	Ghostbusters was [released] on June 8 , [1986] , to critical [acclaim] and became a cultural phenomenon . It was well [received] for its deft blend of comedy, [action] , and horror , and Murray ’ s performance was [often] singled out for praise .
Random	Ghostbusters was [released] on June 8 , [1986] , to critical [acclaim] and became a cultural phenomenon . It was well [received] for its deft blend of comedy, [action] , and horror , and Murray ’ s performance was [particularly] singled out for praise .
Outliers	{ lock was [never] on June 8 , [</s>] , to rely [,] and . It was well [known] for its acker of comedy , [dinner], and horror , and Murray ’ s was [ever] , </s> </s>)

Table 1: Input masked tokens (blue) are given in brackets. RoBERTa correctly reconstructs 4 out of 6 masked tokens (green), and fills in plausible (brown) predictions for the remaining 2 tokens. RoBERTa with 2 randomly disabled LayerNorm dimensions works almost the same as the base model. However, RoBERTa with 2 outlier LayerNorm dimensions makes incorrect and implausible (red) predictions, and changes the hidden token states significantly enough to map the unmasked input tokens to other, often non-sensical words. In this example, we do not show the special tokenizer tokens.

4 Effects of Disabling Outlier Weights

In this section, we consider the effects of disabling outlier weights in BERT on language modeling (§4.1) and on downstream tasks (§4.2). We also investigate whether other Transformers exhibit a similar phenomenon (§4.3).

4.1 Masked Language Modeling

Our key finding is that disabling the outlier dimensions significantly degrades the quality of the language model, even though fewer than 0.001% weights of the model are affected. Table 1 shows a sample output before and after the LayerNorm outliers are disabled in RoBERTa[4]. It is clear that the quality of the language model degrades dramatically, while disabling an equal number of randomly selected non-outliers has almost no effect.

To quantify this effect, we measure the cross-entropy loss before and after disabling each dimension of the LayerNorm on at a time. We do this on the validation subset of the WikiText corpus

---

[4] More sample outputs are given in the Appendix.

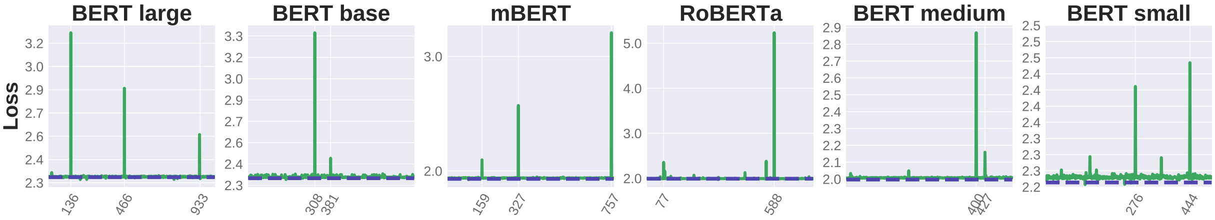

Figure 4: Language model cross-entropy loss (green) after the scaling factor and bias for a given dimension are set to zero, one at a time. The dashed blue line shows the loss achieved by the full model.

		BERT-large		BERT-base		mBERT		RoBERTa		BERT-medium		BERT-small	
		#w	CE	#w	CE	#w	CE	#w	CE	#w	CE	#w	CE
	Baseline	0	2.28	0	2.30	0	1.93	0	1.99	0	2.00	0	2.26
Single	Random	48	2.29	24	2.31	24	1.95	24	2.03	16	2.01	8	2.26
	Top outlier	48	3.22	24	3.33	24	3.21	24	5.23	16	2.87	8	2.44
All	Random	144	2.29	48	2.31	72	1.96	48	2.00	32	2.04	16	2.28
	All outliers	144	**5.49**	48	**4.53**	72	**6.92**	48	**7.85**	32	**3.21**	16	**2.93**

Table 2: Cross-entropy (*CE*) on the validation set of WikiText when the LayerNorm weights are zeroed. *Single* shows the performance when the most damaging outlier is disabled vs. disabling one non-outlier feature (averaged over all non-outliers disabled one at a time). *All* shows performance for when all outliers in a given model are disabled vs. disabling an equal number of randomly selected non-outliers (averaged over 1000 runs). *#w* indicates the total number of modified weights in a given model.

(Merity et al., 2016). We use the standard maximum sequence length of 256 and the token masking probability of 0.15.

All tested models show surprising sensitivity to zeroing out the weights at the outlier positions across all layers of the model (Figure 4). For example, removing only 24 parameters (the scaling factor and the bias of a specific LayerNorm dimension across all 12 layers) increases RoBERTa's loss by almost a factor of 4.

Table 2 shows even more drastic effects when scaling factors and biases for all the outlier dimensions are disabled simultaneously. For comparison, we randomly sample an equal number of non-outlier dimensions and disable the corresponding LayerNorm weights throughout the model. We report the loss averaged over 1000 runs.

4.2 BERT Downstream Tasks

In order to investigate the effect of outlier weights on downstream performance, we evaluate BERT-base on the GLUE benchmark tasks (Wang et al., 2018), with the exclusion of Winograd Schema Challenge, which BERT generally fails to learn (Prasanna et al., 2020). We use the evaluation split of the GLUE benchmark for which the labels are publicly available. As described above, BERT has two outlier dimensions, `308` and `381`. We consider the following two sets of experiments:

1. Disable post fine-tuning. We fine-tune BERT on every GLUE task, then disable the outlier LayerNorm parameters (scaling factor and bias pair) across all layers as described in subsection 3.2. We experiment with disabling each of the two detected outliers both individually and simultaneously in pre-trained BERT-base. We compare the resulting performance to (a) removing the LayerNorm weights for all other hidden dimensions one-by-one, and (b) randomly sampling pairs of non-outlier dimensions so as to disable them simultaneously.

2. Disable pre-fine-tuning. We disable the layer norm parameters for the outlier dimensions *prior* to fine-tuning. Our goal is to check whether the fine-tuning allows the transformers to recover the information from rest of the parameters.

For all the fine-tuning runs, we set the learning rate to 5e-5, batch size to 64 and train for 4 epochs across all the experiments. Since BERT performance varies a lot due to task-specific initialization (Dodge et al., 2020), we use the same initialization across all experiments.

Figure 5 shows model performance when LayerNorm outliers are disabled one at a time. Table 3 compares task-specific performance of the outlier-disabled and the full model for each task.

The main takeaway from the *post fine-tuning* experiments is that disabling one or the other of the

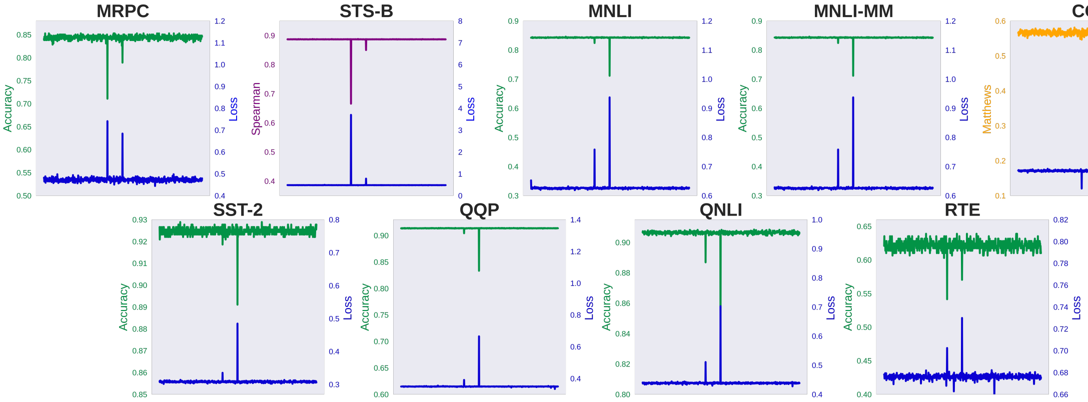

Figure 5: Performance of BERT-base on GLUE tasks when one LayerNorm weight at a time is disabled throughout the model. Dimensions are shown on the X axis. Loss (blue) and accuracy, or correlation coefficients, where applicable (other colors) are shown.

		MRPC	**STS-B**	**MNLI**	**MNLI-mm**	**COLA**	**SST-2**	**QQP**	**QNLI**	**RTE**
	Baseline (full model)	87.2	88.8	84.1	84.2	56.8	92.5	89.8	90.6	61.7
Post-ft	Non-outlier[†]	+0.3	-0.1	-0.2	-0.1	+0.2	0	-0.1	0	-0.4
	Outlier-`308`	-10.5	-23.4	-2.2	-1.8	-2.16	-0.6	-1.0	-1.9	-7.2
	Outlier-`381`	-4.6	-4.4	-13.7	-13.0	-22.2	-3.4	-10.8	-7.3	-5.0
	Random non-outlier pair[‡]	-1.1	0.0	+0.3	+0.2	-0.5	+0.1	+0.1	0	+0.5
	Outliers `308` + `381`	**-8.6**	**-44.1**	**-27.9**	**-27.2**	**-32.3**	**-20.8**	**-13.0**	**-12.2**	**-10.0**
Pre-ft	Random non-outliers*	-0.3	-0.05	-0.2	-0.2	+0.9	-0.06	-0.2	-0.3	+0.6
	Outlier-`308`	+0.3	-0.9	-0.5	+1.7	-0.3	+0.7	-0.1	0	-5.1
	Outlier-`381`	-2.4	-0.7	-0.6	-0.5	-0.9	-1.2	-0.7	-1.4	+4.0
	Outliers `308` + `381`	-1.1	-1.6	-1.4	-0.7	-2.9	-1.7	-0.7	-2.3	-0.7

Table 3: Performance of the pretrained BERT-base model vs. different configurations with disabled outlier weights: post fine-tuning (*post-ft*) and pre-fine-tuning (*pre-ft*). [†]We disable each of non-outlier dimension parameters one at a time and average over them. [‡]For the random sampling of the pairs of non-outlier dimensions, we report averages over 1000 runs. *For the pre-finetuning experiment where random non-outlier parameters are disabled, we sample 10 non-outlier dimensions randomly and disable LayerNorm weights and biases for them across the entire model.

outlier dimensions (or both) drastically degrades model performance on downstream tasks. Which of them affect downstream performance the most is highly task-dependent. For example, the outlier dimension `308` has little effect on CoLA(-2.16) but a large effect on STS-B(-23.4), and for `381` it's the opposite. On SST-2, QNLI, RTE neither outlier drops the performance by over 10 points individually, but disabling them both has a strong adverse effect.

In general, disabling two outliers together causes more severe damage across the board than disabling a single outlier. The overall performance drop is task-specific. The most adversely affected tasks are STS-B (-44.1) and CoLA (-32.3), which are the regression tasks that also suffered the most in pruning experiments by Prasanna et al. (2020). However, MNLI and SST-2 are classification tasks, and both of them also lose over 20 points. Note that disabling random non-outlier dimensions (either alone, or in pairs) has negligible effect on performance across tasks. Disabling one outlier and one random non-outlier has the same effect as disabling a single outlier.

In *pre-fine-tuning* experiments, the question we ask is whether the model can recover from the handicap we introduce, and still learn the task. We expect it to mostly recover, since even randomly initialized BERT without any pre-training can be fine-tuned to solve GLUE tasks fairly well (Kovaleva et al., 2019). In this case, since most of the pretrained sub-networks are still accessible to the classifier, we expect to see a strong recovery close to the baseline. We find this to be the case, however, the model is more adversely affected: -1.1 on average when two outliers are disabled, compared to disabling 10 non-outliers (-0.31 on average).

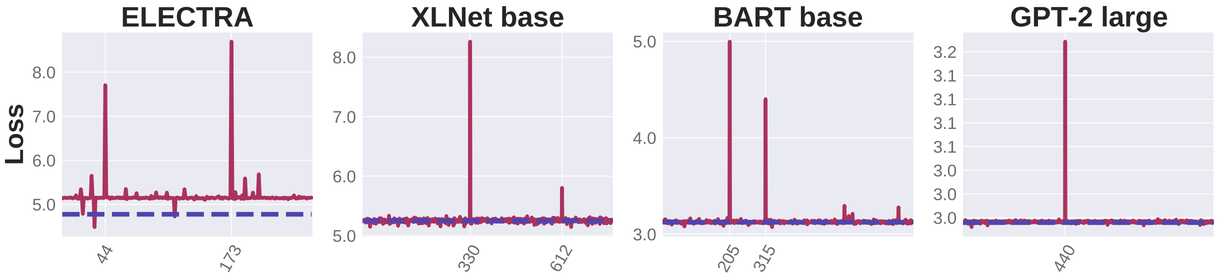

Figure 6: Performance (loss) of other Transformer models after a given LayerNorm scaling factor and bias are set to zero, one at a time. For GPT-2, the dense layer weights and biases are disabled, as it is an instance of a pre-LN model. The dashed blue line shows the loss for the full model.

Model	**Task**	**Data**	**#Params**	**Disabled** # dims	 weight %	**Performance** Baseline	 Disabled
ELECTRA	MLM	WikiText	110M	2	4×10^{-5}	4.8	**8.1**
XLNet	PLM	WikiText	120M	2	4×10^{-5}	5.2	**8.4**
BART	Summ.	CNN/Daily Mail	140M	2	9×10^{-6}	3.1	**4.4**
GPT2	CLM	WikiText	770M	1	0.024	3.0	**3.2**

Table 4: Loss increase in other Transformer models after layer normalization scaling factors and biases highlighted in Figure 6 are set to zero, compared to the baseline configuration. For GPT-2, the dense layer weights and biases are disabled instead, since it is a pre-LN model. *MLM*, *PLM*, and *CLM* stand for masked, permutation, and causal language modeling objectives, respectively. *Summ.* stands for the summarization task. *#dims* denotes the total number of dimensions modified.

4.3 Outliers in Other Transformers

Above, we described our methodology for identifying outliers in BERT models and studied how they affect model performance. In this section, we show that a similar phenomenon is observed in other popular Transformers: BART-base, ELECTRA-base generator, XLNet-base, and GPT-2 large.

Since our goal here was merely to confirm the existence of outlier dimensions, in these experiments, we simply disabled individual LayerNorm dimensions of the encoder part of the model across all Transformer layers. For GPT-2, we modify the weights of the dense output layer instead of the LayerNorm weights, since it uses the pre-LayerNorm (pre-LN) configuration (i.e., LayerNorm is placed before the output feature-producing feed-forward layer).

We perform this experiment for all models, measuring how this affects the loss function in the native pre-training tasks of three models: permutation language modeling task for XLNet, causal language modeling for GPT-2, and masked language modeling for the generator component of ELECTRA. For BART, we found that the pre-trained model had unusually high perplexity out of the box, and we substituted the modeling task with summarization on the CNN/Daily Mail dataset (Hermann et al., 2015).

As with BERT, our results (Figure 6) suggest that for each model, there are a few distinct dimensions which disrupt performance significantly more than the rest. We identify a few most impactful dimensions and also disable them at once, as reported in Table 4. The effect on perplexity is the least pronounced for GPT2, which we attribute to the fact that the model is significantly larger than the others, which may make it more robust to the disabling of individual weights. However, we found that when six dimensions are disabled simultaneously, the perplexity increases by over 300 times.

5 What Makes Outlier Weights Special

5.1 Magnitude or Location?

In this section, we conduct two experiments to validate our proposed criteria for selecting BERT weights to be disabled. Specifically, we want to understand if the same effects would be observed (1) with magnitude-based selection of LayerNorm parameters to be disabled, or (2) using our selection method, but disabling the outlier dimensions only in the first (input) layer or in the later layers (which may have a more direct effect on the output). We use the drop in the loss value for this comparison.

First, we compare the effect of disabling the

	CE	#w	CE	#w
baseline (full model)	2.30	0	2.30	0
Random	2.31	24	2.32	48
LSF	2.72	24	2.74	48
LB	3.21	24	3.42	48
Outlier-`308`	3.32	24	**4.53**	48
Outlier-`381`	2.44	24		

Table 5: BERT-base cross-entropy loss (*CE*) on the WikiText validation data when one (left) or two (right) outlier dimensions are disabled at a time, compared to magnitude-based pruning approaches (LSF and LSB). *#w* denotes the total number of modified weights.

selected dimensions (`308` and `381` in BERT-base, disabled individually or together – i.e. disabling either 12 or 24 LayerNorm scaling factor and bias pairs) to disabling of the following alternatives:

- **Random**: disable 12 or 24 randomly selected pairs of LayerNorm scaling factor and bias pairs in the entire model;
- **Largest Scaling Factor (LSF)**: sort the LayerNorm scaling factors in the model by magnitude and disable the top 12 (or 24) scaling factors and the corresponding biases;
- **Largest Bias (LB)**: repeat the above using the LayerNorm biases instead, i.e. select the top 12 (or 24) LayerNorm biases and disable the corresponding scaling factor / bias pairs.

Table 5 suggests that simple magnitude-based pruning of the output LayerNorm results in a much smaller degradation. As compared to disabling both BERT outliers, the magnitude-driven pruning of the same number of weights results in the value of cross-entropy that is 1.3x smaller.

Looking at the effects of disabling the outlier dimensions only in a subset of layers, Table 6 shows that modifications made to the input layer have little to no effect. Interestingly, switching off the weights in the last Transformer layer, which is used for computing inputs for task-specific classifiers, also does not disrupt the model. However, as we begin to disable earlier layers and the number of layers with disabled weights increases, we observe progressively larger loss values.

We conclude that both the magnitude and the consistent emergence of outlier weights in the same locations across the model are responsible for the emergence of distinct embedding features that BERT heavily relies on.

	1	12	11–12	9–12	7–12	1–12
CE	2.33	2.40	2.67	2.81	2.87	**4.53**
#w	4	4	8	16	24	48

Table 6: BERT-base language modeling cross-entropy loss (*CE*) on the validation set of WikiText corpus, shown by location and number of modified Transformer layers. *#w* denotes the total number of modified weights.

5.2 How Do Outlier Weights Emerge?

In this section, we examine the emergence of outlier dimensions during pre-training. To the best of our knowledge, there are no publicly available BERT pre-training checkpoints available to study these effects. We pre-train[5] a BERT-medium model (chosen due to computational constraints) from scratch on the BookCorpus data (Zhu et al., 2015) and track statistics of the LayerNorm scaling factors and biases. We start from a randomly initialized BERT-medium configuration that has 8 layers with the hidden dimensionality of 512 units. We save checkpoints of the model every 2000 steps, and we track the output LayerNorm weights across all of the model's layers as the training progresses.

Figure 7 shows that both scaling factors and biases begin to diverge from their initialization values quite early (after approximately 50k steps) in the training process. At roughly the same point, both training loss and evaluation perplexity begin to fall off. An interesting question for future work is to clarify whether there is a causal relationship here.

Although the published BERT-medium model had two outlier dimensions, our model had only one dimension for which *both* the scaling factor and the bias exceed their corresponding means by more than three standard deviations.

Due to the gradual emergence of these outliers during pre-training, there is a possibility that thresholding their distance from the rest of the parameters can be used as a litmus test for when the pre-training is complete. We leave the investigation of this hypothesis to future work.

6 Implications and Future Work

The outlier dimension effect we identified may enable **attacks on Transformer-based encoders** that could be used to degrade the model quality while modifying very few weights. Further, since these perturbations to the model do not cause the

---

[5] We used two RTX-3090 GPUs for pre-training.

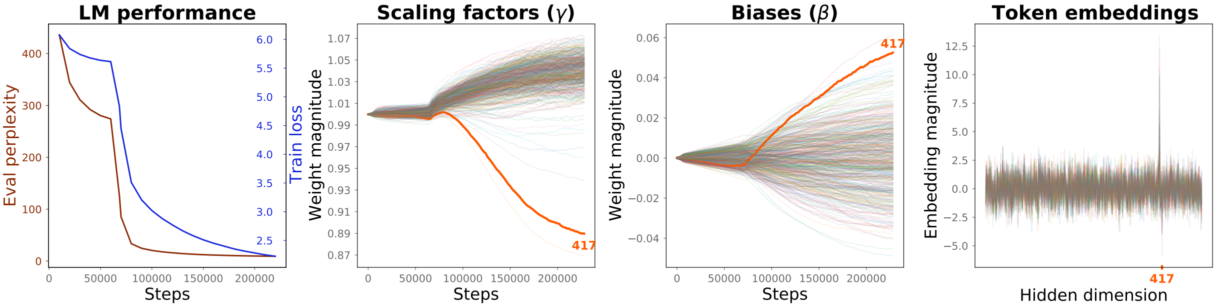

Figure 7: BERT-medium pre-training on the BookCorpus dataset. *(left)* Evaluation perplexity (brown) and train loss (blue) as the training progresses. *(middle)* The changes in the scaling factors and the biases of the output normalization layer. Each line corresponds to one of the 512 dimensions. We highlight (in orange) the `417`-th dimension, for which both the scaling factor and the bias fall out of the three sigma range at the end of pretraining. *(right)* Token embeddings computed for an input sequence that was randomly sampled from the data. Each line corresponds to one input token. The outlier embedding values are marked at the same `417`-th dimension. All the plots are presented for the middle Transformer layer (4).

model to break completely, this may lead to late detection of this attack. To curtail the risks of this attack from affecting deployed Transformer models, we would suggest simple measures such as storing the file checksums for the trained models at a secure location and verifying that the deployed model file matches the checksums.

Another direction for exploiting the phenomenon of outlier dimensions is **pruning**. The studies of model compression using unstructured pruning typically do not consider whether the pruned weights were in the same position throughout the model. Our work suggests that if the outliers were disabled consistently, the drop in performance could be expected to be larger than for random or magnitude-based pruning.

Finally, future work could consider outlier dimensions in the context of **weight initialization**. Our experiments suggest that these dimensions are a normal emergent property of Transformer pretraining. It is possible that higher performance or faster convergence could be achieved by manipulating the initialization to encourage such outliers and experimenting with their number.

7 Conclusion

The main contribution of our work is isolating the phenomenon of a small number of outlier dimensions in Transformer layer outputs which significantly disrupt performance while modifying less than 0.0001% of all parameters of the model. We attribute this phenomenon to an interaction of high-magnitude scaling factors and biases in the same dimension throughout the model, rather than magnitude alone. It emerges early in the training and consistently warps the embedding space.

In case of BERT, the layer output component is LayerNorm. We introduce a method to isolate these outlier dimensions for BERT, and we show that the phenomenon is present in six models of BERT family that we examine. It is also present in four other Transformer-based models (ELECTRA, XLNet, BART, and GPT-2), although the effect of their disabling varies.

8 Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is funded in part by the NSF award number IIS-1844740 to Anna Rumshisky.

We would like to note that in an independent study concurrent to ours, Luo et al. (2021) report a similar effect of outlier values in the embedding space of BERT and RoBERTa. We invite the reader to look into their study for more details on the outlier features from the perspective of word embeddings.

9 Impact statement

This work analyzes the behavior of layer normalization in the popular BERT family of models, using standard benchmarks. No new models are presented, and no data was collected.

We estimate that experiments presented in this paper consumed 436.8 kWh of energy which resulted in a 108 Kilograms of CO2 emissions computed using the regional average of power generation to carbon emissions.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 15834–15846. Curran Associates, Inc.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*.

Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing Transformer Depth on Demand with Structured Dropout. In *International Conference on Learning Representations*.

Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing large-scale transformer-based models: A case study on BERT. *arXiv preprint arXiv:2002.11985*.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online. Association for Computational Linguistics.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2020. Parameter-Efficient Transfer Learning with Diff Pruning. *arXiv:2012.07463 [cs]*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4356–4365, Hong Kong, China. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ziyang Luo, Artur Kulmizev, and Xiaoxi Mao. 2021. Positional artefacts propagate through masked language model embeddings.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? *Advances in Neural Information Processing Systems 32 (NIPS 2019)*.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *CoRR*, abs/1910.05895.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Sheng Shen, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Powernorm: Rethinking batch normalization in transformers. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8741–8751. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and improving layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, pages 4381–4391. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: Quantized 8Bit BERT. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (NeurIPS 2019)*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

10 Appendix

10.1 Candidate outlier dimensions of the models.

For each of the BERT-like models we experimented with we present the outlier candidate weights, detected as described in section 3.

Model component	**Outliers**
output.dense.weight	275, 276, 444
attention.output.dense.bias	193
attention.output.LayerNorm.weight	275, 276, 444
attention.output.LayerNorm.bias	276, 444
output.LayerNorm.weight	121, 262, 444, 276
output.LayerNorm.bias	276, 444

Table 7: BERT-small outlier dimension candidates across model components.

Model component	**Outliers**
attention.output.dense.bias	92, 400, 476, 17
output.dense.weight	400
output.dense.bias	400
attention.output.LayerNorm.weight	17, 400, 430
attention.output.LayerNorm.bias	192, 400
output.LayerNorm.weight	11, 193, 393, 427, 400
output.LayerNorm.bias	400, 427

Table 8: BERT-medium outlier dimension candidates across model components.

Model component	**Outliers**
output.dense.weight	308, 381
output.dense.bias	308
attention.output.dense.bias	308
attention.output.LayerNorm.weight	308, 381
attention.output.LayerNorm.bias	145, 308, 381
output.LayerNorm.weight	92, 145, 308, 381, 225
output.LayerNorm.bias	308, 381

Table 9: BERT-base outlier dimension candidates across model components.

10.2 Scaling factor and bias statistics for BERT-base.

For the BERT-base configuration, we present the detailed statistics on per-layer scaling factors and biases of the output LayerNorm (see Table 13). We report per-layer means, standard deviations and counts of the weights falling out of the three sigma range. We also show the values of the outlier weights (`308` and `381`) along with their ranks, where the ranks are computed for the corresponding sorted arrays of weight magnitudes. Note that the outlier weights consistently appear to be *among* the top largest or top smallest LayerNorm weights throughout the model, but are not necessarily the top-1 largest/smallest values.

10.3 Sample language model outputs after disabling outlier LayerNorm weights.

For RoBERTa and BERT, we randomly sample a set of sentences from Wikipedia and BookCorpus, mask multiple input tokens, and use the models for token prediction. We compare the baseline (full) models with the models where select LayerNorm weights are zeroed out across all of the Transformer layers. In particular, we compare the setups where the outlier dimensions (two per model) are disabled as opposed to random dimensions (two per model).

Model component	**Outliers**
attention.output.dense.bias	757, 327
output.dense.weight	757, 159
output.dense.bias	159, 757
attention.output.LayerNorm.weight	159, 757, 327
attention.output.LayerNorm.bias	159, 757, 327
output.LayerNorm.weight	159, 757, 327
output.LayerNorm.bias	159, 757, 327

Table 10: Multilingual BERT (mBERT) outlier dimension candidates across model components.

Model component	**Outliers**
output.dense.weight	588
output.dense.bias	588, 494
attention.output.dense.bias	588
attention.output.LayerNorm.bias	77, 217, 453, 551, 588, 496, 731, 494
output.LayerNorm.bias	77, 453, 551, 588, 217, 240, 496, 61, 494

Table 11: Base RoBERTa outlier dimensions across model components.

Model component	**Outliers**
attention.output.dense.bias	466, 18
output.dense.bias	466, 750, 18, 933
attention.output.LayerNorm.weight	234, 466, 933
attention.output.LayerNorm.bias	9, 71, 136, 234, 327, 706, 466, 474, 929, 933, 18, 143
output.LayerNorm.weight	80, 136, 232, 234, 331, 466, 639, 665, 702, 724, 750, 763, 968, 315, 428, 933, 18, 506, 314
output.LayerNorm.bias	136, 466, 706, 327, 9, 929, 18, 143, 933

Table 12: BERT-large outlier dimension candidates across model components.

	Scaling factors				**Biases**			
Transf. layer	mean / std	#$>3\sigma$	308 value / rank	381 value / rank	mean / std	#$>3\sigma$	308 value / rank	381 value/rank
1	0.756 / 0.056	12	0.343 / 764	0.404 / 762	-0.037 / 0.099	6	-1.325 / 0	0.144 / 78
2	0.870 / 0.069	24	0.400 / 765	0.374 / 766	-0.034 / 0.086	8	-0.678 / 0	0.277 / 5
3	0.851 / 0.052	16	0.408 / 767	0.549 / 765	-0.031 / 0.075	4	-0.070 / 298	0.118 / 103
4	0.811 / 0.044	11	0.562 / 764	0.388 / 767	-0.033 / 0.052	7	0.075 / 174	0.114 / 50
5	0.840 / 0.045	8	0.615 / 763	0.360 / 767	-0.031/ 0.051	8	0.200 / 3	-0.083 / 113
6	0.832 / 0.037	7	0.692 / 763	0.411 / 767	-0.032 / 0.060	6	0.403 / 0	-0.394 / 1
7	0.834 / 0.037	4	0.752 / 752	0.375 / 767	-0.033 / 0.063	5	0.785 / 0	-0.337 / 1
8	0.810 / 0.030	4	1.163 / 0	0.335 / 767	-0.033 / 0.065	2	0.959 / 0	0.304 / 1
9	0.831 / 0.042	6	1.618 / 0	0.262 / 767	-0.035 / 0.062	2	0.129 / 38	0.695/0
10	0.801 / 0.060	7	1.437 / 0	0.254 / 764	-0.032 / 0.057	9	-0.415 / 2	0.258/4
11	0.817 / 0.062	9	1.671 / 0	0.185 / 765	-0.040 / 0.068	5	-0.667 / 1	1.234/0
12	0.633 / 0.027	13	0.273 / 767	0.536 / 758	-0.019 / 0.050	5	0.225 / 0	-0.021/531

Table 13: The statistics of output LayerNorm weights (scaling factors and biases) for all of the Transformer layers of BERT-base.

Input	Ghostbusters was [released] on June 8 , [1984] , to critical [acclaim] and became a cultural phenomenon . It was well [received] for its deft blend of comedy, [action] , and horror , and Murray ' s performance was [repeatedly] singled out for praise .	a filmy coating of [dust] and pebbles had settled onto the block , and [sami] 's hand instinctively jerked forward to swipe the [scratchy] debris off his cheek , then pulled [up] short against the biting [metal] cuffs .	According to the RIAA, the Beatles are the best-[selling] music artists in the United States, with 178 [million] certified units. They have had more number-[one] albums on the [British] charts and sold [more] singles in the UK than any other act.
RoBERTa	Ghostbusters was [released] on June 8 , [1986] , to critical [acclaim] and became a cultural phenomenon . It was well [received] for its deft blend of comedy, [action] , and horror , and Murray ' s performance was [often] singled out for praise .	a filmy coating of [dirt] and pebbles had settled onto the block , and [Sami] 's hand instinctively jerked forward to swipe the [crusty] debris off his cheek , then pulled [up] short against the biting [leather] cuffs .	According to the RIAA, the Beatles are the best-[selling] music artists in the United States, with 178 [million] certified units. They have had more number-[one] albums on the [US] charts and sold [more] singles in the UK than any other act.
Random	Ghostbusters was [released] on June 8 , [1986] , to critical [acclaim] and became a cultural phenomenon . It was well [received] for its deft blend of comedy, [action] , and horror , and Murray ' s performance was [particularly] singled out for praise.	a filmy coating of [dirt] and pebbles had settled onto the block , and [Tsui] 's hand instinctively jerked forward to swipe the [crusty] debris off his cheek , then pulled [up] short against the biting [leather] cuffs .	According to the RIAA, the Beatles are the best-[selling] music artists in the United States, with 178 [million] certified units. They have had more number-[one] albums on the [US] charts and sold [more] singles in the UK than any other act.
Outliers	{ lock was [never] on June 8 , [</s>] , to rely [,] and . It was well [known] for its acker of comedy , [dinner], and horror , and Murray ' s was [ever] , </s> </s>)	a Fre) covering of [humor] and celebcele had </s> </s> </s> </s> , and [</s>i] 's </s> </s> </s> </s> </s> </s> (@ the [brainy] during (@ end) , Then pulled [*] isk ss the wearing [of] cuffs </s>	2017 </s> the RIAA, the Beatles are the [l] music files in the United States, with 178 [Canadian] Certified ols </s> They have had éyl-[million] Deaths on the [Chart] charts and Died [are] Hearts in</s> UK . </s></s></s></s>

Table 14: RoBERTa's masked language model predictions for randomly sampled input sequences. Input masked tokens (blue) are given in brackets. Correctly predicted tokens are shown in green, incorrect but plausible predictions are shown in brown. 48 weights have been modified in total for the *Random* and *Outliers* setups.

Input	he didnt [really] have a plan and he wasnt sure he [could] go through [with] anything , but the [feeling] of doing something was lifting his [spirits] .	ice is water frozen into a [solid] state . [depending] on the presence of impurities such as particles of soil or [bubbles] of air , it can appear [transparent] or a more or less [opaque] bluish - white color .	but the [sound] of the river babbling by the yard and the ducks splashing on the [pond] seemed to be [working] a cure for her [melancholy] .
BERT	he didnt [even] have a plan and he wasnt sure he [could] go through [with] anything , but the [thought] of doing something was lifting his [spirits] .	ice is water frozen into a [frozen] state . [depending] on the presence of impurities such as particles of soil or [particles] of air , it can appear [white] or a more or less [uniform] bluish - white color .	but the [sound] of the river babbling by the yard and the ducks splashing on the [water] seemed to be [providing] a cure for her [fears] .
Random	he didnt [even] have a plan and he wasnt sure he [could] go through [with] anything , but the [thought] of doing something was lifting his [spirits] .	ice is water frozen into a [liquid] state . [depending] on the presence of impurities such as particles of soil or [particles] of air , it can appear [white] or a more or less [uniform] bluish - white color .	but the [sounds] of the river babbling by the yard and the ducks splashing on the [water] seemed to be [just] a cure for her [fears] .
Outliers	he didny [wee] have a plan and he wasnt sure he [would] go through [it] anything , but the [actual] of doing something was lifting his [shoulders] .	that is water turned into a [yu] state . [based] on the presence of impurities such as particles of soil or [breath] of air , it can appear [white] or a more or more [commoning] ing - white color .	but the [sound] of the child babble by the yard and the ducks splashing on the [windows] all to be [in] a replacement for her [ness] .

Table 15: BERT's masked language model predictions for randomly sampled input sequences. Input masked tokens (blue) are given in brackets. Correctly predicted tokens are shown in green, incorrect but plausible predictions are shown in brown. 48 weights have been modified in total for the *Random* and *Outliers* setups.

"We will Reduce Taxes"
Identifying Election Pledges with Language Models

Tommaso Fornaciari
Università Bocconi
fornaciari@unibocconi.it

Dirk Hovy
Università Bocconi
dirk.hovy@unibocconi.it

Elin Naurin
University of Gothenburg
elin.naurin@pol.gu.se

Julia Runeson
University of Gothenburg
julia.runeson@gu.se

Robert Thomson
Monash University
robert.thomson@monash.edu

Pankaj Adhikari
Monash University
pankaj.adhikari@monash.edu

Abstract

In an election campaign, political parties pledge to implement various projects–should they be elected. But do they follow through? To track election pledges from parties' election manifestos, we need to distinguish between pledges and general statements. In this paper, we use election manifestos of Swedish and Indian political parties to learn neural models that distinguish actual pledges from generic political positions. Since pledges might vary by election year and party, we implement a Multi-Task Learning (MTL) setup, predicting election year and manifesto's party as auxiliary tasks. Pledges can also span several sentences, so we use hierarchical models that incorporate contextual information. Lastly, we evaluate the models in a Zero-Shot Learning (ZSL) framework across countries and languages. Our results indicate that year and party have predictive power even in ZSL, while context introduces some noise. We finally discuss the linguistic features of pledges.

1 Introduction

Before any election, political parties publish manifestos that summarize their pledges to the voters. The exact nature of those pledges varies. A single-issue party might campaign on the same promise year after year, but most parties will adapt to the shifting trends and needs of the electorate. However, there is a difference between pledging and fulfilling. Political scientists are highly interested in whether pledges were fulfilled, a question that is gaining a growing interest in the broader scientific community (Naurin et al., 2019). Several approaches exist, but they are primarily confined to manual analysis of individual countries or elections. They indicate that governmental parties mostly fulfill their election pledges (Naurin et al., 2019; Thomson et al., 2017). However, there are too many elections worldwide to analyze all campaign pledges manually. We need automated ways to identify pledges and hold governments accountable systematically.

Checking whether a pledge was fulfilled still requires manual work by trained political scientists, but the first step–*identifying* pledges–is a problem very much made for NLP, for at least two reasons. First, NLP can automate pledge identification to distinguish pledges from irrelevant content. This allows the study of pledge fulfillment at scale. An average election manifesto in our corpus has 418 sentences, but only 118 of them (27.5%) will contain a pledge. The rest is filler material. It takes several days to train an annotator, who then spends around 6-8 hours on a single manifesto, to identify those 27.5% of pledges. Cutting down on this laborious first part frees up time to focus on the more complex issue of determining whether those pledges were fulfilled. Second, NLP methods can help us understand the linguistic style and communication strategies associated with election pledges. This interpretation is necessary for social sciences to understand how political messages are structured and conveyed.

This paper presents neural pledge identification models to address these two points. Our work is part of a larger interdisciplinary project, "Mixed methods for analyzing political parties' promises to voters during election campaigns." We use a data set of almost 13k sentences from election manifestos concerning the last 25 years and 11 parties from Sweden and India. Each sentence is annotated as including a pledge ("pledge") or not ("non-pledge"). We implement several deep neural models based on BERT (Devlin et al., 2018). We use its Swedish, English (for the Indian data), and multi-lingual (mBERT) versions. We feed BERT's output into customized attention mechanisms to detect specific pledge-related patterns. We com-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3406–3419
August 1–6, 2021.

Corpus	Text	Class
Swedish	Vi i Centerpartiet är stolta över vad vi uppnått i regeringen. *In the Center Party we are proud of what we achieved in the government.*	non-pledge
	Barnkonventionen ska göras till svensk lag. *The Convention on the Rights of the Child shall be made Swedish law.*	pledge
Indian	They have neither competence nor commitment.	non-pledge
	Five new IITs will be established before 2005.	pledge

Table 1: Examples of pledges and non-pledges from Swedish and Indian manifestos.

pare our neural models with a Logistic Regression baseline that the deep models easily outperform.

However, pledges can not just depend on some signal words or expressions. References to the environment might be core pledges for one party, but just commentary for another. Specific issues will be pledge-worthy one year (think pandemic responses), but not in others. To measure the effects of all of these confounds (i.e., *election year* and *party*), we adopt a Multi-Task Learning (MTL) framework. The main task is to classify sentences as pledge or non-pledge, with auxiliary tasks predicting the year, party, or both. We identify the conditions where MTL models with year and party improve the models' performance, indicating when these two factors are useful confounds. There seem to be stark differences between countries, though: even using a multi-lingual approach (which has access to more training data) does not improve on language-specific approaches.

We are also interested in zero-shot learning, i.e., training models on data from a country and testing it on a different country. This would allow us to work on pledges from new countries directly, without any previous manual annotation. It turns out that the models perform reasonably well despite the challenging conditions. However, the differences between test countries indicate that pledges are not as universal as we might think.

Surprisingly, we also find that incorporating a context of any sort (that is, one or more sentences preceding the target text) does not help but hurts performance. Presumably, this happens because pledges are rare, and context introduces more noise than signal.

We are also interested in learning more about pledges' nature. I.e., what their linguistic features and patterns are. To gain those insights, we extract the Information Gain value (Forman, 2003) of 1–4-grams and visualize the model's decisions via the Sampling and Occlusion (SOC) algorithm (Jin et al., 2019). SOC provides a hierarchical view of BERT's most informative linguistic patterns in the classification.

Our data and our models are available at `https://github.com/MilaNLProc/mimac`.

Contributions The contributions of this paper are: 1) We provide a new, multi-lingual corpus of election manifestos from Swedish and Indian parties, annotated at sentence level as pledges or non-pledges; 2) We are the first to apply neural models to the task of election pledge classification, accounting for confounds; 3) We provide insights about the linguistic features of election pledges and the models' interpretation.

2 Data

We collect and annotate a corpus of election manifestos from two countries: Sweden and India. The texts are in Swedish and English, respectively. We provide some examples in Table 1.

The Swedish data contain 5098 instances from 9 parties and six elections, ranging from 1994 to 2014. The amount of pledges per manifesto is 32.09%. These texts are also part of the corpus of the Manifesto Project (MP) (Volkens et al., 2012; Merz et al., 2016, Section 7).

For all manifestos, we adopted the annotation scheme of the Comparative Party Pledges Project (CPPP) of Naurin et al. (2019) and Thomson et al. (2017). This is a large international political science project whose annotation scheme is the most appropriate for identifying campaign promises, which is the focus of our experimental designs. In particular, following the CPPP scheme, we further distinguish between broad and narrow pledges, i.e., between generic and detailed commitments to undertake determined actions. Based on this distinction, we ran additional experiments included in the Appendix. We have 23.32% narrow and 8.77%

Country	Sweden	India	Both
Nr. manifestos	16	11	27
Nr. Parties.	9	2	11
Nr. elections' year.	6 (1994-2014)	5 (1999-2019)	10 (1994-2019)
Mean sentences	318.62	702.64	475.07
Mean pledges	102.25 (32.09%)	172.27 (24.52%)	130.78 (27.53%)
Mean sentences' length	15.00	20.70	18.43

Table 2: Corpora statistics.

broad pledges in the Swedish data.

The Indian texts contain 7729 sentences from two parties and five election cycles from 1999 to 2019.[1] Here, the annotators only distinguished sentences including a narrow pledge from non-pledge sentences, with a pledge rate of 24.52%.

In total, we have 12827 sentences and 3531 pledges (27.53%). Since we only have binary labels for the Indian data, we combine broad and narrow pledges in the Swedish corpus.[2] Table 2 shows some corpora statistics.

2.1 Annotation process

In the CPPP scheme, an election pledge is a statement that can be tested for fulfillment. Annotators must therefore assess whether a statement refers to an action or outcome that is verifiable, in the sense that we can objectively determine whether it was achieved. This definition also requires annotators to have to contextual knowledge of the country and specific information about the political situation in each election campaign.

We therefore trained Swedish and Indian annotators to label the Swedish and Indian manifestos for our study, respectively. Four people were involved in the annotation of the manifestos. Two domain experts, one for each data set, conducted the training. The two annotators interacted with the two respective domain experts throughout the annotation process to handle complicated cases.

To test agreement in the Indian data, three trained annotators labeled 100 sentences. Their Krippendorff's α and Fleiss's κ are 0.65. On the Swedish data, two trained annotators labeled 100 sentences again, with Krippendorff's α and Cohen's κ at 0.61. In both cases, the agreement can be considered as 'substantial' (Landis and Koch, 1977). Our results are coherent with those reported by Naurin et al. (2019).

3 Methods

We have three experimental conditions: 1) Swedish texts alone, 2) Indian texts alone, and 3) Swedish and Indian texts together (multilingual condition).

In conditions 1) and 2) we evaluate the models on test sets from the same county (standard test split), or from the respective other country, i.e., a Zero-Shot Learning (ZSL) condition. This is not possible in the third condition, where the models are trained on data from both countries. We use this last condition to see whether performance improves with access to more training data and whether pledges are comparable across countries.

As baselines, we train two Logistic Regression (LR) models, optimized with the parameter $C = 1$, based on TF-IDF-weighted Bag-Of-Words (BOW) from $1-$ to $3-$grams, with document frequency range from 0.001 and 0.75. We feed the first models with simple $n-$gram tokens.

However, we also hypothesized that pledges could be expressed by formal grammatical patterns, such as specific Parts-of-Speech (PoS) sequences or verb tenses (future tense, modal verbs). Therefore, we trained a second LR model, fed with tokens incorporating the PoS information. Tables 3 and 4 show the performance. We evaluate our models with standard metrics: precision, recall, and F1-measure averaged over the two classes.

3.1 Neural models

For the first two experimental conditions, we consider separate, mono-lingual Swedish and English BERT models and the multi-lingual (mBERT) version. In the third experimental condition, where we merge the two data sets, we can only use the multi-lingual BERT.

---

[1] If referring to the data set, we will use *Indian*, but if referring to the language, we use *English*.

[2] We trained binary classifiers for narrow pledges in a pilot study, treating broad pledges as non-pledges. The performance was slightly worse than in the case reported here due to a more noisy "non-pledge" class and a more skewed class balance. We include those results in the Appendix.

	Acc.	Prec.	Rec.	F1
Sweden	76.30	73.53	75.73	74.17
India	77.90	71.81	76.15	73.10
Both	77.28	72.88	76.57	73.90

Table 3: Logistic Regression with tokens' $n-$grams.

	Acc.	Prec.	Rec.	F1
Sweden	75.48	72.03	72.77	72.36
India	78.12	71.90	75.98	73.18
Both	75.83	71.22	74.52	72.14

Table 4: Logistic Regression with tokens + PoS' $n-$grams.

Single-Task Learning. Our base models are binary classifiers, i.e., single-tasks (STL) models. Standard BERT classifiers perform the task with a fully connected layer on top of BERT's output. In contrast, we reframe BERT's `[CLS]` token representation as a single-row matrix, and feed it into a single-layer, single-head Transformer (Vaswani et al., 2017). Our pilot studies found that this specialized structure allows us to detect specific pledge patterns from the BERT representation more effectively than a standard dense output layer alone. Finally, the Transformer is connected to a dense output layer for the prediction.

Multi-Task Learning. We implement three different MTL versions, differing by the auxiliary task combinations. We have two potential auxiliary tasks: predicting the *election year* and the *party* that produced the manifesto. We add a further dense output layer to the base model to perform the MTL tasks: 1) predicting the election year, 2) the party, or 3) both. We use the mean of the task losses for error-backpropagation in the MTL networks. Since their magnitude is bounded by the fact that all predictions are probability distributions, no normalization is needed. Figure 1 (left) shows the scheme of the MTL models.

Contextual models. We also build models considering the sentence preceding the target text as context, allowing us to test its impact on classification performance. We incorporate the context sentence in two state-of-the-art ways: through pair-BERT, which accepts two texts as input, and through a hierarchical model. In the first case, the model is structurally equivalent to the base model: only the input representation for BERT changes to include two sentences, separated by the separator token `[SEP]`. In the second case, we stack the representations of the BERT classification tokens (`[CLS]`) of both context and target sentences and feed them into a Transformer connected to a dense layer that gives the output. Figure 1 (right) depicts its structure.

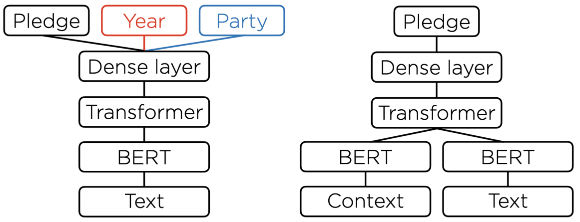

Figure 1: Left: STL and MTL model scheme. STL: black boxes. MTL with one auxiliary task: black + red or black + blue boxes. MTL with two auxiliary tasks: all boxes. Right: Hierarchical models' scheme

Settings and significance tests. To reduce the variability of the models' random initialization and make our results more robust, we run ten repeats for each experimental condition and compute the overall performance. To test the significance of the improvements over the base model, we use a bootstrap sampling test on all runs (Søgaard et al., 2014), with 1000 loops and a sample size of 30%.

For each experiment, we run 10-fold cross-validation. In each fold, we use 80% of texts as the training set, 10% for the development, and 10% for the test. In the ZSL experiments, we use 90% and 10% of a data set for training and development, respectively, and the whole other data set as the test set.

For the main task, the loss function is the binary (sigmoid) cross-entropy; it is the (soft-max) cross-entropy for the auxiliary tasks. We use the Adam optimizer (Kingma and Ba, 2014). We select the models through early-stopping that requires the development set's loss to drop by less than 8% for five consecutive epochs. Our learning rate is 0.002, drop-out probability 0.3, and batch size 512, manually tuned. The attention mechanisms that analyze BERT's outputs are single-layer, single-head Transformers.

4 Experiments

We report results on all models for each of the three experimental conditions: 1) Swedish corpus encoded with Swedish and multi-lingual BERT (Table 5); 2) Indian corpus encoded with English and

BERT	Task	Target	acc	prec	rec	f1
Swedish	STL	Sweden	87.01	85.19	84.87	85.03
Swedish	MTL Party	Sweden	87.11	85.35	84.89	85.12
Swedish	MTL Year	Sweden	87.07	85.37	84.74	85.04
Swedish	MTL Party + Year	Sweden	87.05	85.45	84.51	84.95
Multilingual	STL	Sweden	81.94	79.57	78.20	78.81
Multilingual	MTL Party	Sweden	81.78	79.35	78.10	78.66
Multilingual	MTL Year	Sweden	82.05	79.77	78.14	78.85
Multilingual	MTL Party + Year	Sweden	81.83	79.55	77.80	78.54
Multilingual	STL	India (0-shot)	73.15	68.78	74.34	69.27
Multilingual	MTL Party	India (0-shot)	**75.33** **	**69.24** *	73.45	**70.31** **
Multilingual	MTL Year	India (0-shot)	**74.4** **	69.08	74.05	**69.96** **
Multilingual	MTL Party + Year	India (0-shot)	**75.67** **	**69.17** *	72.83	**70.25** **

Table 5: Training data set: Sweden. Language: Swedish. Significance of MTL over STL: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
English	STL	India	83.74	78.98	74.57	76.31
English	MTL Party	India	83.63	78.71	74.64	76.27
English	MTL Year	India	83.91	**79.62** *	74.1	76.16
English	MTL Party + Year	India	83.89	78.84	**75.68** **	**77.02** *
Multilingual	STL	India	83.49	78.08	75.64	76.71
Multilingual	MTL Party	India	83.48	78.41	74.58	76.14
Multilingual	MTL Year	India	83.70	**78.81** *	74.78	76.41
Multilingual	MTL Party + Year	India	83.69	**78.75** **	74.84	76.42
Multilingual	STL	Sweden (0-shot)	73.57	75.32	60.73	60.44
Multilingual	MTL Party	Sweden (0-shot)	72.29	76.19	57.95	56.05
Multilingual	MTL Year	Sweden (0-shot)	72.28	**76.32** *	57.91	55.98
Multilingual	MTL Party + Year	Sweden (0-shot)	72.69	**76.58** *	58.63	57.12

Table 6: Training data set: India. Language: English. Significance of MTL over STL: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
Multilingual	STL	Both	82.74	78.70	76.84	77.67
Multilingual	MTL Party	Both	82.80	79.05	76.17	77.37
Multilingual	MTL Year	Both	82.53	78.48	76.42	77.32
Multilingual	MTL Party + Year	Both	82.73	78.91	76.19	77.33

Table 7: Data set: Sweden & India. Language: Swedish and English.

multi-lingual BERT (Table 6); and 3) the joint Swedish and Indian data together, encoded with multi-lingual BERT (Table 7).

For each of these conditions, we train a baseline Logistic Regression model (Section 3)—an STL base model as described in Section 3.1—and compare them with MTL and contextual models. Since all the models outperform the Logistic Regression baselines, we report significance levels concerning the improvement over the STL models.

5 Results

We see a substantial performance difference between the two BERT encodings (Swedish and mBERT) regarding the Swedish data. The Swedish version outperforms the multi-lingual one and reaches the best performance of the experiments (Table 5).

We do not see the same performance difference in the Indian data, where English and multi-lingual BERT produce similar outcomes, with the multi-lingual even slightly better. Results are generally lower than those for the Swedish data (Table 6).

To interpret this performance gap, we need to consider the differences between the two corpora. As shown in Table 2, the Swedish and Indian data sets differ remarkably in terms of the number of parties and manifestos. Within each manifesto, the two data sets also contain a remarkably different number of sentences, pledges, and words in each sentence. In particular, the Indian data set contains a lower pledge rate than the Swedish data. This reduced amount of training examples prevents a direct comparison between the models trained on the two corpora.

As expected, the results of the multi-lingual model trained on the joint data set lie between the respective multi-lingual models on the two data sets separately. So while the Swedish BERT is more effective than the multi-lingual one on Swedish texts, the amount of data in the multi-lingual language model presumably counteracts the lack of annotated data in the Indian data set.

MTL vs STL. The MTL models are effective in several cases.First, they help in the ZSL conditions. This suggests that training the models to contextualize the notion of a pledge for party and year reduces overfitting. Also, when effective, MTL models improve precision. This is an expected effect, as the models learn to detect pledges as well as historical periods and political areas. This is an interesting feature for ZSL, where confidence in identified positive cases is more valuable than a good recall. In fact, even though models maximizing recall would make the human activity of pledge *identification* easier, in terms of downstream pledge fulfillment *verification*, it is preferable to start from a smaller set of texts that are likely to be true pledges.

Furthermore, in ZSL, by definition, the years and parties of the target country differ from those of the training country. Therefore, the auxiliary predictions for the training country are not relevant to the target country. This is the reason why we frame the problem as multi-task rather than multi-input: we could not have fed the models with test data from unseen countries/election campaigns in multi-input. Nevertheless, models trained to distinguish between different contexts for years and parties can effectively transfer this knowledge to entirely different test data, improving the predictions' precision. This suggests that some generalization is possible, even in front of different dependent variables.

We also tested the MTL models in the case of a reduced amount of data. In particular, we trained models considering the election manifestos from 2000 only. We found that the MTL contributes more strongly under those conditions. The results of these experiments are included in the Appendix.

Does Context Help? In a word, no. While a disappointing outcome, we find it important to include this finding here, as it goes very much against both intuition and prior research. Bilbao-Jayo and Almeida (2018), for example, found that contextual information is helpful when classifying political topics (see Section 7). Election pledges seem to be more self-contained statements, relying on linguistic formulas that make them recognizable (and probably memorizable) regardless of their linguistic context (Section 6).

We explored two different models to incorporate the sentence preceding the target texts. In both cases, though, we consistently find that the previous sentence's contextual information adds more noise than a helpful signal for prediction. The decrease ranges from moderate to drastic (up to 10 points in F1), particularly for the pair-BERT models where, by design, target and context representations are not trainable. The hierarchical models' performance is more stable, but the context does not improve the performance.

6 The language of pledges

To better understand the pledges' linguistic features, we follow two strategies: 1) computing the Information Gain (IG) of word n-grams, and 2) using the Sampling and Occlusion (SOC) algorithm (Jin et al., 2019).

Information Gain measures the entropy of (sequences of) terms between the different classes. The more skewed a set of terms is towards one label class at the other's expense, the higher the IG value. Tables 8 and 9 show the trigrams with the highest IG values (and relative frequencies), divided according to the class of which they are indicative, i.e., where they are more frequently found. While

	IG	Fr.
Vi_vill_också	0.013561	21
Ett_införande_av	0.008383	13
•_Ett_införande	0.008383	13
•_Ett_utökat	0.005799	9
Ett_utökat_stöd	0.004509	7
•_En_satsning	0.004509	7
•_En_utökad	0.004509	7
utökat_stöd_till	0.004509	7
utökad_satsning_på	0.004509	7
så_att_det	0.004177	10

	IG	Fr.
Alliansen_har_följande	0.005713	26
har_följande_skarpa	0.004391	20
följande_skarpa_förslag	0.004391	20
I_vårt_Sverige	0.004205	41
vill_under_kommande	0.003072	14
ska_vara_ett	0.003072	14
Alliansen_vill_under	0.003072	14
under_kommande_mandatperiod	0.003072	14
kommande_mandatperiod_att:	0.003072	14
Det_är_en	0.002632	12

Table 8: Swedish tri-grams indicative of pledge (left) and non-pledge (right)

	IG	Fr.
will_be_set	0.020450	49
be_set_up	0.017386	41
will_be_launched	0.015698	26
will_set_up	0.014979	20
the_next_five	0.013200	23
in_the_next	0.011840	19
set_up_a	0.011294	27
in_five_years.	0.010608	15
be_launched_to	0.010163	17
over_the_next	0.009743	14

	IG	Fr.
It_is_the	0.006252	40
the_Congress_that	0.005938	38
is_the_Congress	0.005311	34
National_Congress_will	0.004462	95
will_be_made	0.003918	63
The_Congress_will	0.003897	60
the_Congress_is	0.003744	24
A_time_to	0.003431	22
has_always_been	0.003275	21
It_is_a	0.002962	19

Table 9: Indian tri-grams indicative of pledge (left) and non-pledge (right)

we computed the IG score from $1 - -5$-grams, we show only tri-grams here for illustration. They represent the best trade-off between meaningful and frequent chunks of text. For the complete translation of the Swedish texts, see the Appendix.

These n-grams suggest that a formulaic language characterizes election pledges: stereotypical expressions characterize specific sentences as pledges. For example, in the Swedish data set, the bullet is a clear marker that introduces statements containing some form of commitment. We also find expressions indicating volition ("Vi **vill** också" – "We also **want**..."), consequences ("**så** att det" – "**So** that..."), future ("**will** be set", "**will** be launched") and determined temporal horizons ("in five years", "over the next"). In contrast, both in the Indian and the Swedish data, references to political entities such as parties ("Alliansen"), congresses ("National Congress") and even countries ("Sverige", "India") are associated with non-pledge texts: they refer, more probably, to broad political positions or to claims about the past ("has always been", "ska vara ett" – "should be one").

Interestingly, the phrase "skarpa förslag" does *not* signal pledges, even though it means "specific policy proposals" (which are essentially the same as pledges). This distinction indicates that this phrase merely introduces pledges or provides a strong language for un-testable policy statements (such as "we promise safety to all children" or "we will put forward strict legislation to make our country safe again").

Given the relatively limited frequency of the selected n-grams, we did not measure the IG stratification by party and/or election year. However, given the relative MTL models' success, we hypothesize that, with more data, it will be possible to identify specific trends for political areas and historical moments.

Aware that the patterns detected by the neural models are not necessarily interpretable in terms of human common sense, we also wanted to highlight the words that the models find to be the most influential for their output. These patterns can feedback into the interpretation of pledge structures and mechanisms by social scientists.

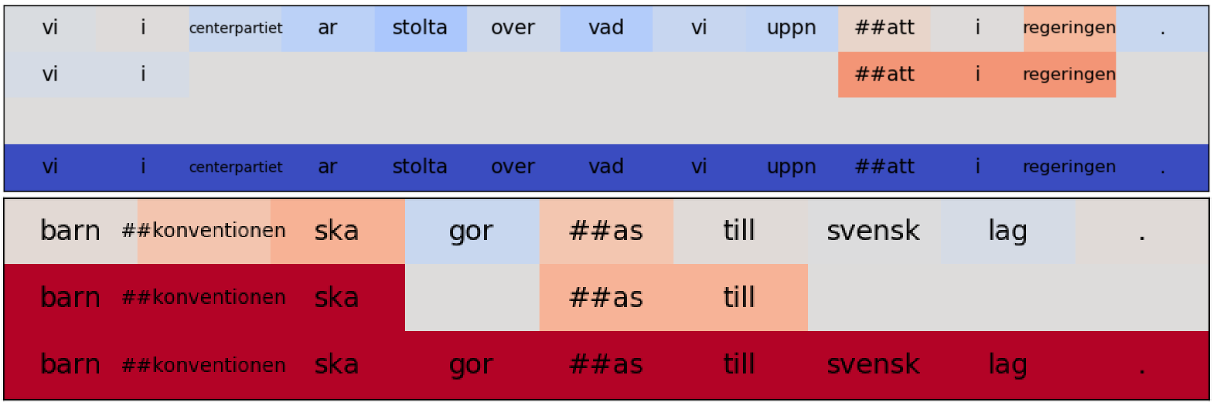

Figure 2: Output of the SOC algorithm on the Swedish corpus. The red terms predict Pledge, the blue ones predict Non-pledge.
Non-pledge translation: ‘We in the center party are proud of what we have achieved in the government.’
Pledge translation: ‘The Convention on the Children’s Rights shall be made Swedish law.’

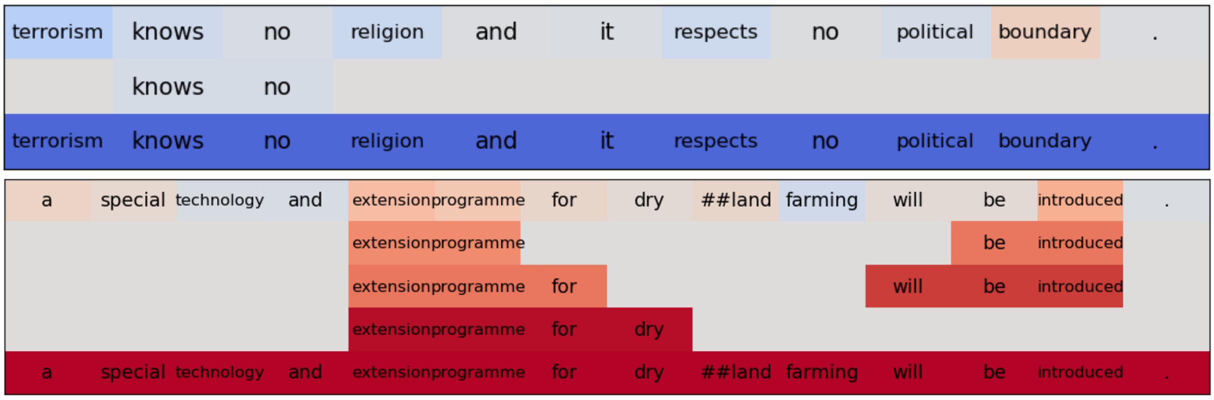

Figure 3: Output of the SOC algorithm on the English corpus. The red terms predict Pledge, the blue ones predict Non-pledge.

We also use the Sampling and Occlusion (SOC) algorithm (Jin et al., 2019), a *post-hoc* explanation algorithm that measures the importance of specific words in a sentence by considering the prediction difference after replacing each word with a MASK token (Jin et al., 2019). Since the outcomes depend on the context words, but Jin et al. (2019) are interested in the single words’ relevance, they do not use the whole context but sample words from it. In this way, they reduce the context weight, emphasizing that of the word itself.

Figure 2 and 3 show four examples of correctly classified sentences, two pledges and two non-pledges from Swedish and English language respectively (the same as shown in Table 1). The model interprets the red words as indicative of pledges, the blue ones of non-pledges. However, they cannot be interpreted as representative of the overall models’ functioning. Even so, they show how generic words such as “stolta” (“proud”) are indicative of non-pledges, while expressions indicating commitment (“ska göras till” – “to be made to”) and concrete topics (“Barnkonventionen” – “Convention on Children’s Rights”) are signals for pledges.

7 Related Work

In political sciences, the elections that we consider have been extensively studied by Håkansson and Naurin (2016), Lindvall et al. (2020) and Adhikari et al. (2020). Moreover, applying NLP methods to the analysis of political parties’ statements has recently developed into an active field of research, with various groups investing in creating dedicated corpora and annotating them for specific purposes.

The Manifesto Project (MP) (Volkens et al., 2012; Merz et al., 2016) collects electoral programs from more than 50 countries for democratic elections since 1945, making it a notable initiative within the field. It provides data on different manifesto aspects in several countries and over time. Recently, the Comparative Party Pledges Project (CPPP) of Naurin et al. (2019) has added detailed qualitative coding of what exactly pledges are made of (Naurin and Thomson, 2020).

Subramanian et al. (2018) study the MP data, addressing the identification of fine- vs. coarse-grained positions taken by political parties. Despite the different classification task, similarly to our study, they adopt hierarchical models that encode the texts' structure, finding that contextual information improves the models' performance. However, they train bi-LSTM networks from scratch, while we rely on pre-trained BERT language models.

Bilbao-Jayo and Almeida (2018) also work on the MP corpus, applying multi-input Convolutional Neural Networks (CNN) that take into account the statements' context, analogously to our study. They seek to classify the texts according to seven topics corresponding to general areas of interest.

We partially use the same data as the MP, as we study Swedish manifestos included in that data set. However, we are specifically interested in the identification of election pledges. This is similar to the task studied by Subramanian et al. (2019a). They focus on eleven Australian federal election cycles and distinguish rhetorical (broad) from detailed (narrow) pledges. The annotation of the Swedish texts considers this distinction, while the annotated Indian texts of our corpus do not (Section 2). Subramanian et al. (2019a) use a bidirectional Gated Recurrent Unit (biGRU) to carry out the prediction over ordinal classes.

From a methodological point of view, our approach is related to that of Abercrombie et al. (2019), which also uses BERT. They work on motions tabled in the UK Parliament and find that BERT effectively detects specific categories of proposals in the politicians' speeches.

Concerning the MTL methods, our study is analogous to that of Subramanian et al. (2019b). They consider texts from the 2016 Australian election and propose a new annotation scheme for different *speech acts*. They also perform the classification task using biGRU networks with ELMo embeddings (Peters et al., 2018), relying on a MTL framework in which the auxiliary task is the party prediction: this is also one of our experimental conditions.

8 Conclusion

We propose deep neural models that combine pre-trained language models and trainable attention mechanisms to identify election pledges in party manifestos. We find that these models outperform a non-neural baseline. Even in zero-shot cross-lingual conditions (with some contribution by the MTL methods), the performance of the multilingual models indicates that we could identify pledges in low-resource languages.

Finally, we gained some insight into election pledges' linguistic profile. They are self-contained statements, independent of the context in which they appear. They are likely to be characterized by formulaic expressions that express commitment, intentions, and temporal terms concerning concrete topics. These results stem from close interdisciplinary cooperation between political scientists and NLP researchers.

Pledge identification is the first step for future downstream NLP tasks within the theoretical framework of political science, which is typically interested in societal developments and explanations such as pledge fulfillment and power distribution in democracies. For example, the fine-grained study of topics, biases, and the temporal evolution of election pledges. Our results provide a blueprint for successful future research in that vein.

Acknowledgements

DH and TF are members of the Data and Marketing Insights Unit at the Bocconi Institute for Data Science and Analysis. This project is funded by Riksbankens Jubileumsfond, with ID MXM19-1137:1, "Mixed methods for analyzing political parties' promises to voters during election campaigns."

Ethical Considerations

The data we release are publicly available political manifestos. The texts are not harmful and do not contain personal information. The annotation and the relative classification task do not raise privacy concerns.

References

Gavin Abercrombie, Federico Nanni, Riza Theresa Batista-Navarro, and Simone Paolo Ponzetto. 2019. Policy preference detection in parliamentary debate motions. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 249–259.

Pankaj Adhikari, Sania Mariam, and Robert Thomson. 2020. Indian parties' election pledges and pledge fulfilment. *APSA Online Annual Conference*.

Aritz Bilbao-Jayo and Aitor Almeida. 2018. Automatic political discourse analysis with multi-scale convolutional neural networks and contextual data. *International Journal of Distributed Sensor Networks*, 14(11):1550147718811827.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Nicklas Håkansson and Elin Naurin. 2016. Promising ever more: An empirical account of swedish parties' pledge making during 20 years. *Party Politics*, 22(3):393–404.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. 2019. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. *arXiv preprint arXiv:1911.06194*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Johannes Lindvall, Hanna Bäck, Carl Dahlström, Elin Naurin, and Jan Teorell. 2020. Sweden's parliamentary democracy at 100. *Parliamentary Affairs*, 73(3):477–502.

Nicolas Merz, Sven Regel, and Jirka Lewandowski. 2016. The manifesto corpus: A new resource for research on political parties and quantitative text analysis. *Research & Politics*, 3(2):2053168016643346.

Elin Naurin, Terry J Royed, and Robert Thomson. 2019. *Party mandates and democracy: Making, breaking, and keeping election pledges in twelve countries*. New Comparative Politics.

Elin Naurin and Robert Thomson. 2020. The fulfilment of election pledges. In *Research Handbook on Political Representation*. Edward Elgar Publishing.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. 2014. What's in a p-value in nlp? In *Proceedings of the eighteenth conference on computational natural language learning*, pages 1–10.

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2018. Hierarchical structured model for fine-to-coarse manifesto text analysis. *arXiv preprint arXiv:1805.02823*.

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2019a. Deep ordinal regression for pledge specificity prediction. *arXiv preprint arXiv:1909.00187*.

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2019b. Target based speech act classification in political campaign text. *arXiv preprint arXiv:1905.07856*.

Robert Thomson, Terry Royed, Elin Naurin, Joaquín Artés, Rory Costello, Laurenz Ennser-Jedenastik, Mark Ferguson, Petia Kostadinova, Catherine Moury, François Pétry, et al. 2017. The fulfillment of parties' election pledges: A comparative study on the impact of power sharing. *American Journal of Political Science*, 61(3):527–542.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Andrea Volkens, Onawa Lacewell, Pola Lehmann, Sven Regel, Henrike Schultze, and Annika Werner. 2012. The manifesto data collection. manifesto project (mrg/cmp/marpor). *Wissenschaftzentrum Berlin (https://manifestoproject. wzb. eu/)*.

A Appendix

A.1 Logistic regression baseline with Broad pledges collapsed to Non-pledges

	Acc.	Prec.	Rec.	F1
Sweden	77.74	70.94	75.65	72.31
India	77.90	71.81	76.15	73.10
Both	77.45	71.53	76.71	72.83

A.2 Experiments with Broad pledges collapsed to Non-pledges: Tables 1, 2, 3

BERT	Task	Target	acc	prec	rec	f1
Swedish	STL	Sweden	87.97	83.78	81.68	82.65
Swedish	MTL Party	Sweden	87.94	83.74	81.64	82.61
Swedish	MTL Year	Sweden	88.17	84.24	81.65	82.82
Swedish	MTL Party + Year	Sweden	88.09	84.22	81.33	82.63
Multilingual	STL	Sweden	83.51	77.68	73.42	75.11
Multilingual	MTL Party	Sweden	83.60	77.98	73.14	75.01
Multilingual	MTL Year	Sweden	83.47	77.65	73.26	75.00
Multilingual	MTL Party + Year	Sweden	83.55	77.92	73.03	74.91
Multilingual	STL	India (0-shot)	76.82	70.59	74.68	71.79
Multilingual	MTL Party	India (0-shot)	**79.06** **	**71.95** **	73.23	**72.53** **
Multilingual	MTL Year	India (0-shot)	**77.19** *	70.54	73.89	71.67
Multilingual	MTL Party + Year	India (0-shot)	**78.79** **	**71.72** **	73.38	**72.44** **

Table 10: Data set: Sweden. Language: Swedish. Significance: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
English	STL	India	83.81	79.10	74.65	76.41
English	MTL Party	India	83.82	79.27	74.37	76.26
English	MTL Year	India	83.86	79.51	74.08	76.11
English	MTL Party + Year	India	83.88	79.04	75.12	76.72
Multilingual	STL	India	83.52	78.44	74.74	76.25
Multilingual	MTL Party	India	83.58	78.58	74.73	76.29
Multilingual	MTL Year	India	83.60	78.72	74.47	76.16
Multilingual	MTL Party + Year	India	83.66	78.70	74.81	76.39
Multilingual	STL	Sweden (0-shot)	80.46	76.69	61.66	63.53
Multilingual	MTL Party	Sweden (0-shot)	79.72	**78.23** *	58.52	59.15
Multilingual	MTL Year	Sweden (0-shot)	79.69	**78.4** **	58.39	58.95
Multilingual	MTL Party + Year	Sweden (0-shot)	79.50	**78.06** *	57.94	58.27

Table 11: Data set: India. Language: English. Significance: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
Multilingual	STL	Both	83.58	78.66	73.39	75.37
Multilingual	MTL Party	Both	83.73	78.34	**74.91** **	**76.34** **
Multilingual	MTL Year	Both	83.48	77.92	**74.71** **	**76.06** **
Multilingual	MTL Party + Year	Both	83.74	78.59	**74.35** **	**76.04** **

Table 12: Data set: Sweden & India. Language: Swedish and English. Significance: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

A.3 Experiments considering elections from 2000 only: Tables 4, 5, 6

BERT	Task	Target	acc	prec	rec	f1
Swedish	STL	Sweden	86.62	84.93	84.48	84.70
Swedish	MTL Party	Sweden	86.79	85.2	84.52	84.85
Swedish	MTL Year	Sweden	86.83	85.33	84.42	84.84
Swedish	MTL Party + Year	Sweden	86.89	**85.4** *	84.49	84.91
Multilingual	STL	Sweden	81.89	79.75	78.29	78.92
Multilingual	MTL Party	Sweden	81.92	79.78	78.33	78.96
Multilingual	MTL Year	Sweden	82.11	79.91	**78.8** *	79.30
Multilingual	MTL Party + Year	Sweden	82.20	80.26	78.36	79.15
Multilingual	STL	India (0-shot)	75.44	69.49	73.71	70.56
Multilingual	MTL Party	India (0-shot)	**76.38** **	69.74	72.93	70.8
Multilingual	MTL Year	India (0-shot)	**76.1** **	**69.95** **	73.93	**71.07** **
Multilingual	MTL Party + Year	India (0-shot)	**76.22** **	**70.13** **	**74.19** *	**71.26** **

Table 13: Data set: Sweden from 2000. Language: Swedish. Significance: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
English	STL	India	83.46	78.23	75.50	76.68
English	MTL Party	India	83.51	78.75	74.39	76.11
English	MTL Year	India	83.44	78.32	75.09	76.44
English	MTL Party + Year	India	83.58	78.52	75.37	76.69
Multilingual	STL	India	83.59	78.43	75.66	76.85
Multilingual	MTL Party	India	83.70	78.73	75.44	76.81
Multilingual	MTL Year	India	83.65	78.90	74.74	76.40
Multilingual	MTL Party + Year	India	83.66	78.55	75.67	76.91
Multilingual	STL	Sweden (0-shot)	74.32	72.77	64.72	65.7
Multilingual	MTL Party	Sweden (0-shot)	73.32	**74.68** **	61.29	61.15
Multilingual	MTL Year	Sweden (0-shot)	71.92	**75.11** **	58.37	56.66
Multilingual	MTL Party + Year	Sweden (0-shot)	**74.86** *	73.51	**65.49** *	**66.6** *

Table 14: Data set: India from 2000. Language: English. Significance: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
Multilingual	STL	Both	83.06	79.54	76.7	77.88
Multilingual	MTL Party	Both	83.28	79.76	**77.17** *	**78.27** *
Multilingual	MTL Year	Both	83.23	79.69	**77.08** *	78.19
Multilingual	MTL Party + Year	Both	83.23	79.55	**77.43** **	**78.36** *

Table 15: Data set: Sweden & India from 2000. Language: Swedish and English. Significance: $^{**}: p \leq 0.01$; $^{*}: p \leq 0.05$

A.4 Experiments with Contextual models: pair-BERT and Hierarchical Transformers: Tables 7, 8, 9

BERT	Task	Target	acc	prec	rec	f1
Swedish	Hier. T. - no context	Sweden	87.06	85.21	85.01	85.11
Swedish	Hier. T. - prev. sentence	Sweden	86.35	84.41	84.16	84.28
Swedish	single-BERT - no context	Sweden	87.19	85.34	85.22	85.28
Swedish	pair-BERT - prev. sentence	Sweden	80.87	78.12	77.62	77.86
Multilingual	Hier. T. - no context	Sweden	81.46	79.20	77.12	77.98
Multilingual	Hier. T. - prev. sentence	Sweden	81.07	78.42	77.58	77.97
Multilingual	single-BERT - no context	Sweden	81.72	79.32	77.92	78.54
Multilingual	pair-BERT - prev. sentence	Sweden	78.52	75.53	73.96	74.61
Multilingual	Hier. T. - no context	India (0-shot)	74.67	69.17	73.99	70.12
Multilingual	Hier. T. - prev. sentence	India (0-shot)	**77.77 ****	**70.06 ****	70.42	70.23
Multilingual	single-BERT - no context	India (0-shot)	73.14	67.95	72.76	68.66
Multilingual	pair-BERT - prev. sentence	India (0-shot)	63.73	65.52	70.87	61.87

Table 16: Data set: Sweden. Language: Swedish. Significance: $^{**}: p \leq 0.01;\ ^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
English	Hier. T. - no context	India	83.91	79.36	74.59	76.44
English	Hier. T. - prev. sentence	India	83.53	78.61	74.39	76.07
English	single-BERT - no context	India	83.76	79.14	74.36	76.21
English	pair-BERT - prev. sentence	India	79.02	72.06	64.65	66.49
Multilingual	Hier. T. - no context	India	83.61	78.69	74.58	76.23
Multilingual	Hier. T. - prev. sentence	India	82.85	77.22	74.45	75.63
Multilingual	single-BERT - no context	India	83.56	78.58	74.63	76.23
Multilingual	pair-BERT - prev. sentence	India	79.63	72.61	67.71	69.36
Multilingual	Hier. T. - no context	Sweden (0-shot)	72.39	76.44	58.09	56.25
Multilingual	Hier. T. - prev. sentence	Sweden (0-shot)	**73.42 ****	75.94	**60.22 ****	**59.65 ****
Multilingual	single-BERT - no context	Sweden (0-shot)	72.50	74.68	58.71	57.41
Multilingual	pair-BERT - prev. sentence	Sweden (0-shot)	68.84	74.56	51.69	44.29

Table 17: Data set: India. Language: English. Significance: $^{**}: p \leq 0.01;\ ^{*}: p \leq 0.05$

BERT	Task	Target	acc	prec	rec	f1
Multilingual	Hier. T. - no context	Both	82.64	78.84	75.94	77.14
Multilingual	Hier. T. - prev. sentence	Both	82.30	78.65	74.85	76.33
Multilingual	single-BERT - no context	Both	82.53	78.91	75.31	76.73
Multilingual	pair-BERT - prev. sentence	Both	78.95	73.92	70.20	71.54

Table 18: Data set: Sweden & India. Language: Swedish and English. Significance: $^{**}: p \leq 0.01;\ ^{*}: p \leq 0.05$

A.5 Examples of n-grams and relative IG value: Table 10

	IG	Fr.
Vi_vill_också *We also want*	0.013561	21
Ett_införande_av *An introduction of*	0.008383	13
•_Ett_införande *An introduction*	0.008383	13
•_Ett_utökat *An extended*	0.005799	9
Ett_utökat_stöd *An extended support*	0.004509	7
•_En_satsning *A investment*	0.004509	7
•_En_utökad *An extended*	0.004509	7
utökat_stöd_till *extended support for*	0.004509	7
utökad_satsning_på *extended investment on*	0.004509	7
så_att_det *so that it*	0.004177	10

	IG	Fr.
Alliansen_har_följande *The alliance has the following*	0.005713	26
har_följande_skarpa *has following sharp*	0.004391	20
följande_skarpa_förslag *the following sharp suggestions*	0.004391	20
I_vårt_Sverige *In our Sweden*	0.004205	41
vill_under_kommande *want during coming*	0.003072	14
ska_vara_ett *should be one*	0.003072	14
Alliansen_vill_under *The Alliance wants during*	0.003072	14
under_kommande_mandatperiod *during the coming term of office*	0.003072	14
kommande_mandatperiod_att *forthcoming term of office that*	0.003072	14
Det_är_en *It is a*	0.002632	12

Table 19: Swedish tri-grams indicative of pledge (left) and non-pledge (right)

WeaQA: Weak Supervision via Captions for Visual Question Answering

Pratyay Banerjee Tejas Gokhale Yezhou Yang Chitta Baral
Arizona State University
`pbanerj6, tgokhale, yz.yang, chitta@asu.edu`

Abstract

Methodologies for training visual question answering (VQA) models assume the availability of datasets with human-annotated *Image-Question-Answer* (I-Q-A) triplets. This has led to heavy reliance on datasets and a lack of generalization to new types of questions and scenes. Linguistic priors along with biases and errors due to annotator subjectivity have been shown to percolate into VQA models trained on such samples. We study whether models can be trained without any human-annotated Q-A pairs, but only with images and their associated textual descriptions or captions. We present a method to train models with synthetic Q-A pairs generated procedurally from captions. Additionally, we demonstrate the efficacy of spatial-pyramid image patches as a simple but effective alternative to dense and costly object bounding box annotations used in existing VQA models. Our experiments on three VQA benchmarks demonstrate the efficacy of this weakly-supervised approach, especially on the VQA-CP challenge, which tests performance under changing linguistic priors.

1 Introduction

Since Visual Question Answering (VQA) was first proposed as a Turing test (Malinowski and Fritz, 2014), several human-annotated datasets (Mogadala et al., 2019) have been used to train and evaluate VQA models. Unfortunately, heavy reliance on these datasets for training has the unwanted side-effects of bias towards answer styles, question-types (Chao et al., 2018), and spurious correlations with language priors (Agrawal et al., 2018). Similar findings have been reported for natural language tasks (Gururangan et al., 2018; Niven and Kao, 2019; Kaushik et al., 2020). Evaluating VQA models on test-sets that are very similar to training sets is deceptive and inadequate and not an accurate measure of robustness.

To address this, one line of work has focused on balancing, de-biasing, and diversifying samples (Goyal et al., 2017; Zhang et al., 2016). However, crowd-sourcing "unbiased" labels is difficult and costly; it requires a well-designed annotation interface and a large-scale annotation effort with dedicated and able annotators (Sakaguchi et al., 2020). The alternative (that this paper aligns itself with) is to avoid the use of explicit human annotations and instead to train models in an unsupervised manner by synthesizing training data. These techniques, coined *unsupervised*[1], come with many advantages – human bias and subjectivity are reduced; the techniques are largely domain-agnostic and can be transferred from one language to another (low resource languages) or from one visual domain to another. For instance, template-based Q-A generation developed for synthetic blocks-world images in CLEVR (Johnson et al., 2017) can also be used to generate Q-A pairs for natural complex scenes in GQA (Hudson and Manning, 2019) or the referring-expressions task (Liu et al., 2019).

In this work, we train VQA models without using human-annotated Q-A pairs. Instead, we rely on weak supervision from image-captioning datasets, which provide multi-perspective, concise, and less subjective descriptions of visible objects in an image. We procedurally generate Q-A pairs from these captions and train models using this synthetic data, and *only evaluate* them on established human-annotated VQA benchmarks.

Why Captions? Image captioning, like VQA, has been a central area of vision-and-language research. Datasets such as MS-COCO (Lin et al., 2014; Chen et al., 2015) contain captions that describe objects and actions in images of everyday scenes. During the construction of MS-COCO, human captioners were instructed to refrain from describing past and future events or "what a person might say". On the other hand, annotators of

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3420–3435
August 1–6, 2021.

VQA (Antol et al., 2015) were instructed to ask questions that *"a smart robot cannot answer, but a human can"* and "interesting" questions that may require "commonsense". Different sets of annotators provided answers to these questions and were allowed to speculate or even guess an answer that *most people would agree on*. It has also been shown that multiple answers may exist for questions in common VQA datasets (Bhattacharya et al., 2019).

In Figure 2, the first VQA-v2 question asks how many doors the car has. Although commonsense (and linguistic priors) would suggest that "Most cars have *four* doors", only two doors can be seen in the image. What should the model predict, *two* or *four*? The second question is subjective and has multiple contradicting answers from different annotators (where one should draw the line between opaque, transparent, or reflective is not very clear). Similarly, the first GQA question is ambiguous and could refer to either the skier or the photographer.

Thus the very nature of the data-collection procedure and instructions for VQA brings in human subjectivity and linguistic bias as compared to caption annotations, which are designed to be simple, precise, and non-speculative. Motivated by this, we study the benefits of using captions to synthesize Q-A pairs, using three types of methods:

1. template-based methods similar to (Ren et al., 2015a; Gokhale et al., 2020b),
2. paraphrasing and back-translation (Sennrich et al., 2016) which provide linguistic variation,
3. synthesis of questions about image semantics using the QA-SRL (He et al., 2015) approach.

Since our Q-A pairs are created synthetically, there does exist a domain shift as well as label (answer) shift from evaluation datasets such as VQA-v2 and GQA as shown in Figure 2, thus posing challenges to this weakly-supervised method.

We evaluate two models, UpDown (Anderson et al., 2018) and a transformer-encoder (Vaswani et al., 2017) based model pre-trained on synthetic Q-A pairs and image-caption matching task. To remove the dependence on object bounding-boxes and labels needed to extract object features, we propose spatial pyramids of image patches as a simple and effective alternative.

To the best of our knowledge, this is the first work on the unsupervised[1] visual question answering, with the following contributions:

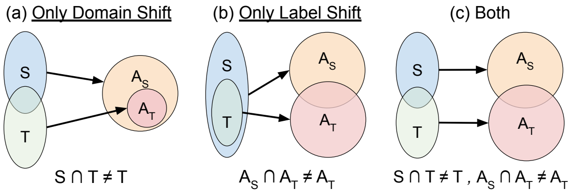

Figure 1: Aspects of generalization in VQA.

- We introduce a framework for synthesizing *(Question, Answer)* pairs from captions.
- Since synthetic samples (unlike popular benchmarks) include multi-word answer phrases, we propose a sub-phrase weighted-answer loss to mitigate bias towards such multi-word answers.
- We propose pre-training tasks that use spatial pyramids of image-patches instead of object bounding-boxes, further removing the dependence on human annotations.
- Extensive experiments and analyses under zero-shot transfer and fully-supervised settings on VQA-v2, VQA-CP, and GQA show our model's efficacy and establish a strong baseline for future work on unsupervised visual question answering.

2 Related Work

Robustness in VQA can be defined as shown in Figure 1 under two situations: domain shift and label shift. Under domain shift, generalization to a new input domain (such as different styles of questions or novel scenes) is desired, characterized by $S \cap T \neq T$ where S and T denote the train and test input domains. Under label shift, generalization to novel answers is desired (predicting answers not seen during training), characterized by $A_S \cap A_T \neq A_T$, where A_S and A_T are the set of answers seen during training and test-time.

Performance under **domain shift** has been evaluated for new domains of test questions with unseen words and objects (Teney and Hengel, 2016; Ramakrishnan et al., 2017), novel compositions (Johnson et al., 2017; Agrawal et al., 2017), logical connectives (Gokhale et al., 2020b), as well as questions that are implied (Ribeiro et al., 2019), entailed (Ray et al., 2019) or sub-questions (Selvaraju et al., 2020); or for datasets with varying linguistic styles (Chao et al., 2018; Xu et al., 2020; Shrestha et al., 2019) and different reasoning capabilities (Kafle and Kanan, 2017).

Label shift or Prior Probability Shift (Storkey,

---

[1] adhering to the usage of this term in Lewis et al. (2019a).

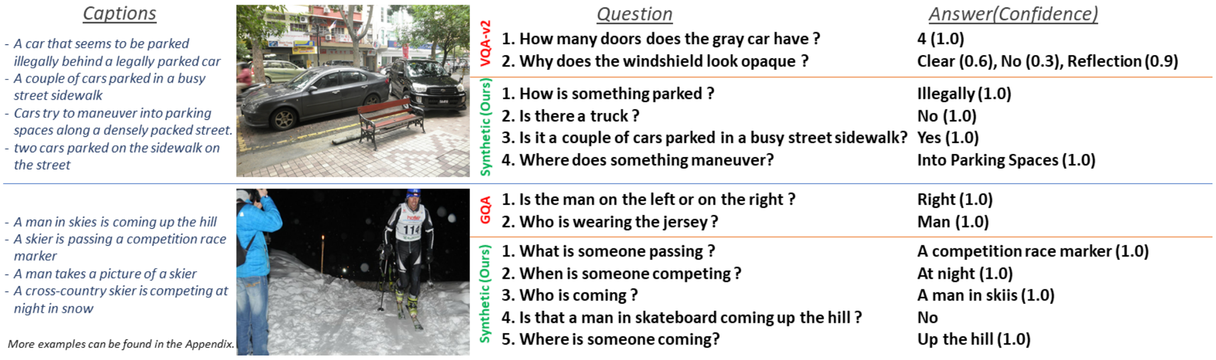

Captions

- *A car that seems to be parked illegally behind a legally parked car*
- *A couple of cars parked in a busy street sidewalk*
- *Cars try to maneuver into parking spaces along a densely packed street.*
- *two cars parked on the sidewalk on the street*

	Question	*Answer(Confidence)*
VQA-v2	1. How many doors does the gray car have ?	4 (1.0)
	2. Why does the windshield look opaque ?	Clear (0.6), No (0.3), Reflection (0.9)
Synthetic (Ours)	1. How is something parked ?	Illegally (1.0)
	2. Is there a truck ?	No (1.0)
	3. Is it a couple of cars parked in a busy street sidewalk?	Yes (1.0)
	4. Where does something maneuver?	Into Parking Spaces (1.0)

- *A man in skies is coming up the hill*
- *A skier is passing a competition race marker*
- *A man takes a picture of a skier*
- *A cross-country skier is competing at night in snow*

GQA	1. Is the man on the left or on the right ?	Right (1.0)
	2. Who is wearing the jersey ?	Man (1.0)
Synthetic (Ours)	1. What is someone passing ?	A competition race marker (1.0)
	2. When is someone competing ?	At night (1.0)
	3. Who is coming ?	A man in skiis (1.0)
	4. Is that a man in skateboard coming up the hill ?	No
	5. Where is someone coming?	Up the hill (1.0)

More examples can be found in the Appendix.

Figure 2: Examples of images and human-annotated Q-A pairs from VQA and GQA and our synthetic Q-A pairs.

2009) has been implicitly explored in VQA-CP (Agrawal et al., 2018), where the conditional probabilities of answers given the question type deviate at test-time. Teney et al. (2020c) have identified several pitfalls associated with the models and evaluation criteria for VQA-CP.

Unsupervised Extractive QA in which aligned *(context, question, answer)* triplets are not available, has been studied (Lewis et al., 2019b; Banerjee and Baral, 2020; Rennie et al., 2020; Fabbri et al., 2020; Li et al., 2020; Banerjee et al., 2021) by training models on procedurally generated Q-A pairs. Captions have been used to generate Q-A pairs for logical understanding (Gokhale et al., 2020b) and commonsense video understanding (Fang et al., 2020a). Li et al. (2018); Krishna et al. (2019) have explored Visual Question Generation from an input image and answer.

Weak supervision is an active area of research; for instance in action/object localization (Song et al., 2014; Zhou et al., 2016) and semantic segmentation (Khoreva et al., 2017; Zhang et al., 2017) without pixel-level annotations, but only class labels. There is also interest growing in leveraging natural language captions or textual queries as weak supervision for visual grounding tasks (Hendricks et al., 2017; Mithun et al., 2019; Fang et al., 2020b).

Visual Feature Extractors such as VGG (Simonyan and Zisserman, 2015) and ResNet (He et al., 2016) have been widely used for many computer vision tasks. Object-based features such as RCNN (Girshick et al., 2014) and Faster-RCNN (Ren et al., 2015b) have become the standard for V& L tasks (Anderson et al., 2018).

3 Framework for Synthesizing Q-A Pairs

Problem Statement: Consider a dataset containing images and associated captions as shown in Figure 2. Our work deals with learning VQA using these image-caption data, without any labeled Q-A pairs, and answer questions about unseen images.

3.1 Question Generation

Several studies (Du et al., 2017; Lewis et al., 2019a) have been dedicated to the complex domain of question generation. We approach it conservatively, using template-based methods and semantic role labeling, with paraphrasing and back-translation for improving the linguistic diversity of template-based questions. We begin by extracting object words from the caption by using simple heuristics such as extracting noun-phrases and using numerical quantifiers in the caption as soft approximations of objects' cardinality. If object-words are available explicitly, we used them as is. Questions are categorized based on answer types; *Yes-No*, *Number*, *Color*, *Location*, *Object*, and *Phrases*.

Template-based: To create *Yes-No* questions, modal verbs are removed from the caption, and a randomly chosen question prefix such as *"is there"*, *"is this"* is attached. For instance, the caption "A man is wearing a hat and sitting" is converted to "*Is there a* man wearing a hat and sitting", with the answer "Yes". To create the corresponding question with the answer "No", we use either negation or replace the object-word with an adversarial word or antonym, thus obtaining "Is there a dog wearing a hat and sitting" for which the answer is "No". An adversarial word refers to an object absent in the image but similar to objects in the image. To compute similarity, we use Glove (2014) word-vectors.

For *Object*, *Number*, *Location*, and *Color* questions, we follow a procedure similar to Ren et al.

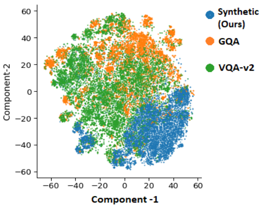

	Template-based	Paraphrase & Back-translate	QA-SRL	VQA-v2	GQA	VQA-CP
# of Questions	600K	400K	2.5M	438K / 214K	943K / 132K	245K / 220K
# of Answers	5K	5K	90K	3.5K	1878	3.5K
Mean Question Length	7.9	8.1	4.8	6.4	10.6	6.4
Mean Answer Length	1.4	1.4	6.3	1.1	1.3	1.1
Image Source	COCO	COCO	COCO	COCO	COCO,VG,Flickr	COCO
Image Counts	120K	120K	120K	120K	113K	120K

Figure 3: Discrepancy between VQA-v2, GQA, and synthetic samples. Left: t-SNE plot of question embeddings. Right: Dataset statistics for our generated Q-A pairs with Train/Val. splits for benchmark datasets.

(2015a). To create "*what*" questions for the *Object* type, we extract objects and noun phrases from captions as potential answers and replace them with *what*. The question is rephrased by splitting long sentences into shorter ones and converting indefinite determiners to definite. A similar procedure is used for *Number* questions; numeric quantifiers of noun phrases are extracted and replaced by "how many" and "what is the count" to form the question. *Color* questions are generated by locating the color adjective and the corresponding noun phrase and replacing them in a templated question: "What is the color of the object?". *Location* questions are similar to *Object* questions, but we extract phrases with "in", "within" to extract locations, with places, scenes, and containers as answers.

Semantic Role Labeling: QA-SRL (He et al., 2015) was proposed as a paradigm to use natural language to annotate data by using Q-A pairs to specify textual arguments and their roles. Consider the caption *"A girl in a red shirt holding an apple sitting in an empty open field"*. Using QA-SRL with B-I-O span detection and sequence-to-sequence models (FitzGerald et al., 2018), for the *"when", "what", "where", and "who"* questions, we obtain Q-A pairs belonging to the *Phrases* category such as:

(what is someone holding?, an apple)
(who is sitting?, girl in a red shirt holding an apple)
(where is someone sitting?, an empty open field)

These examples illustrate that QA-SRL questions are short and use generic descriptors such as *something* and *someone* instead of elaborate references, while the expected answer phrases are longer and descriptive. Thus to answer these, better semantic image understanding is required.

Paraphrasing and Back-Translation (P&B): We apply two natural language data augmentation techniques, paraphrasing, and back-translation to increase the linguistic variation in the questions. To paraphrase questions, we train a T5 (Raffel et al., 2019) text generation model on the Quora Question Pairs Corpus (). For back-translation, we train another T5 text generation model on the Opus corpus (2012), translate the question to an intermediate language (Français, Deutsche, or Español), and translate the question back to English. For example:

Is the girl who is to the left of the sailboats wearing a backpack?
↓ Español
La chica que está a la izquierda de los veleros lleva mochila?
↓ English
Does the girl to the left of the sailboats carry a backpack?

3.2 Domain Shift w.r.t. VQA-v2 and GQA

Compared to current VQA benchmarks (which typically contain one-word answers), answers to QA-SRL questions are more descriptive and contain adjectives, adverbs, determiners, and quantifiers, as seen in Figure 2. On the other hand, synthetic questions have less descriptive subjects due to the use of pronouns. Our synthetic data contains $90k$ unique answer phrases, compared to $3.2k$ in VQA and 3k in GQA. Around 200 answers from VQA are not present in our answer phrases, such as time (11:00) and proper nouns (LA Clippers), both of which are not present in caption descriptions.

Moreover, our training data contains Q-A pair such as *("Where is the man standing?, "to the left of the table")*, generated by QA-SRL with long phrases as answers. However, the test set contains questions such as *("Which side of the car is the tree?", "left")*, which expects only *"left"* as the answer. So although the word *"left"* is seen as a sub-phrase of our training answers, it is not explicitly seen as an only correct answer.

Some of our synthetic template-based questions about counting and object presence are similar in style to those in VQA and GQA. However, QA-SRL questions require a semantic understanding of the actions depicted in the image, which are rare in VQA and GQA. We quantify this by plotting the t-SNE components of document vector embeddings of the questions from VQA, GQA, and our synthetic data, in Figure 3, and observe that our synthetic questions are a distinct cluster, while VQA and GQA overlap with each other. As such, a linguistic domain shift exists between these synthetic source questions and human-annotated target questions. In this paper, we address the challenge of learning VQA on a synthetically generated dataset and evaluating models on conventional benchmarks which have questions and answers that deviate linguistically from synthetic training samples.

4 Method

Recently, multiple deep transformer-based architectures have been proposed (Tan and Bansal, 2019; Lu et al., 2019; Chen et al., 2019), that are pre-trained on a combination of multiple VQA and image captioning datasets such as Conceptual Captions (Sharma et al., 2018), SBU Captions (Ordonez et al., 2011), Visual Genome (Krishna et al., 2017), and MSCOCO (Lin et al., 2014). These models are resource intensive as they are trained on a huge collection of data with 3 million images. We train our models only on MS-COCO captions and images ($\sim$204k), without access to any human-authored Q-A pairs or object bounding boxes.

4.1 Spatial Pyramid Patches

"Bottom-Up" object features (Anderson et al., 2018) extracted from Faster R-CNN (Ren et al., 2015b) have become the de-facto features used in state-of-the-art VQA models. These VQA models thus only use features of detected objects as input, and ignore the rest of the image. Although object features are discriminative, dense annotations are required for training and additional large deep networks for extraction. Object detection can be imperfect for small and rare objects (Wang et al., 2019); for instance if an object detection model detects only four out of six bananas in an image, features of the other two bananas will not be used by VQA models. This creates a performance bottle-neck for questions about counting or rare objects.

We take a step back and postulate that the use of features of the entire image in context could reduce this bottleneck. Image features extracted from a ResNet (He et al., 2016) trained for the ImageNet (Russakovsky et al., 2015) classification task, which is widely used for computer vision tasks, have been previously used for VQA models (Goyal et al., 2017). Unfortunately, since ImageNet contains iconic (single-object) images, using these features for non-iconic VQA images is restrictive since many questions refer to multiple objects and backgrounds in the image. Inspired by Spatial Pyramid Matching (Lazebnik et al., 2006) for image classification, we propose *spatial pyramid patch features* to represent the input VQA image into a sequence of features at different scales.

We divide each image I into a set of image patches $\{I_{k_1}, \ldots, I_{k_n}\}$, each I_{k_i} being a $k_i \times k_i$ grid of patches, and extract ResNet features for each patch. Larger patches encode global features and relations, while smaller patches encode local and low-level features.

Encoder: Our Encoder model is similar to the UNITER single-stream transformer, where the sequence of word tokens $w = \{w_1, ..., w_T\}$ and the sequence of image patch features $v = \{v_1, ..., v_K\}$ are taken as input. We tokenize the text using a WordPieces (Wu et al., 2016) tokenizer similar to BERT (Devlin et al., 2019), and embed the text tokens through a text-embedder (Sanh et al., 2019). The visual features are projected to a shared embedding space using a fully-connected layer. A projected visual position encoding, indicating the patch region (top-right, bottom-left) is added to the visual features. We concatenate both sequences of features and feed them to L cross-modality attention layers. Parameters between the cross-modality attention layers are shared to reduce parameter count and increase training stability (Lan et al., 2020), and a residual connection and layer normalization is added after cross-modal attention layer similar to Vaswani et al. (2017).

4.2 Pre-training Tasks and Loss Functions

We train the Encoder model using three pre-training tasks: Masked Language Modeling, Masked Question Answering, and Image-Text Matching.

Masked Language Modeling (MLM): We randomly mask 15% of the word tokens from the caption and ask the model to predict them. For the caption "There is a man wearing a hat", the model gets the input "There is `[MASK]` wearing a hat". Without the image, there can be multiple plausible choices for the `[MASK]` token, such as "woman", "man", "girl", but given the image the model should predict "man". This task has been shown to effectively learn cross-modal features (2019).

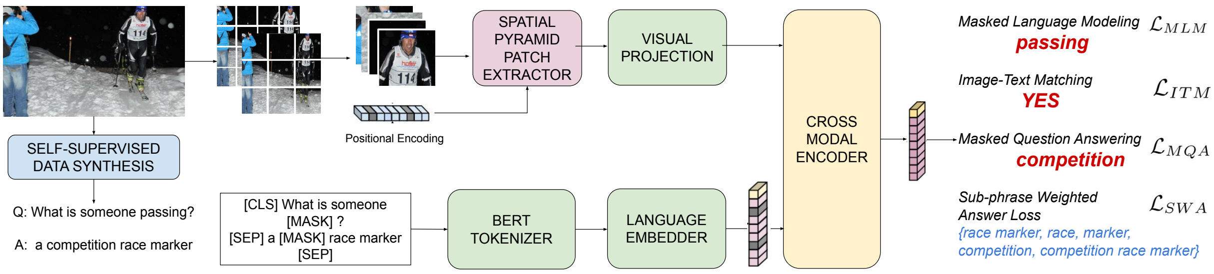

Figure 4: Our model architecture makes the use of spatial pyramids of image patches as inputs to the Encoder, which is trained for three pre-training tasks as shown.

Masked Question Answering (MQA): In this task, the answer tokens are masked, and the model is trained to predict the answer tokens. For example in Figure 2, for the input " When is someone competing? `[MASK] [MASK]`", the model should predict, "at night". To answer such questions, the model needs to interpret the image.

Image-Text Matching (ITM): We use the five captions provided by MS-COCO as positive samples for each image. To obtain negative samples, we randomly sample captions from other images that contain a different set of objects. We train the model on a binary classification task (matching / not matching) for each image-caption pair.

For VQA and ITM, we use the final layer representation $z^{[CLS]}$ of `[CLS]` token , followed by a feed-forward and softmax layer. For MLM and MQA we feed corresponding token representations to a different feed-forward layer. We train the model using cross-entropy loss for all three tasks.

Sub-phrase Weighted Answer Loss: As observed before, the questions generated in QA-SRL have long answer phrases. For instance "What is parked?" has the answer "two black cars". We extract all possible sub-phrases that can be alternate answers, but assign them a lower weight than the complete phrase, computed as $W_{sub} = \mathit{WordCount}(sub)/\mathit{WordCount}(ans)$. Thus "two black cars" has a weight 1.0, while the extracted sub-phrases and weights are: (two, 0.33), (2, 0.33), (black, 0.33), (cars, 0.33), (two cars, 0.66), (2 cars, 0.66), (black cars, 0.66), (car, 0.33). This enforces a distribution over the probable answer space instead of a strict "single true answer" training. We train the model with this additional binary cross-entropy loss, where the model predicts a weighted distribution y_{wa} over the answer vocabulary. The vocabulary is defined from the synthetic QA answer-space.

$$\mathcal{L}_{SWA} = \mathcal{L}_{BCE}(\sigma(z^{[CLS]}), y_{wa}). \tag{1}$$

The total loss, with scalar coefficients $\alpha, \beta \in (0, 1]$ is given by:

$$\mathcal{L} = \mathcal{L}_{MLM} + \mathcal{L}_{MQA} + \alpha \cdot \mathcal{L}_{ITM} + \beta \cdot \mathcal{L}_{SWA}. \tag{2}$$

5 Experimental Setup

Datasets: We evaluate our methods on the three popular visual question answering benchmarks: VQA-v2, VQA-CP-v2, and GQA. Answering questions in VQA-v2 and VQA-CP v2 requires image and question understanding, whereas GQA further requires spatial understanding such as compositionality and relations between objects. We evaluate our methods under *zero-shot* transfer (trained only on procedurally generated samples), and *fully-supervised* (where we finetune our model using the associated train annotations) settings. We use exact-match accuracies for GQA, and use VQA-metric (Agrawal et al., 2017) for VQA.

Training: Our Encoder has 8 cross-modal layers with a hidden dimension of 768. The weights are initialized using the standard definition as provided in the Huggingface repository (Wolf et al., 2019). Our models are pre-trained for 40 epochs with a learning rate of 1e−5, batch size of 256, using Adam optimizer. For finetuning, we use a learning rate of 1e−5 or 5e−5 and batch size of 32 for 10 epochs. We use a ResNet-50 pretrained on ImageNet to extract features from image patches with 50% overlap, and Faster R-CNN pretrained on Visual Genome to extract object features. We evaluate both frozen and finetuned ResNet, and observe finetuning the feature extractor to perform better. All our models are trained using 4 Nvidia V100 16 GB GPUs. All results in the fully supervised setting are reported for from-scratch trained final classification layers.

Model	All	Yes-No	Num	Others
SAN (2016)	25.0	38.4	11.1	21.7
GVQA (2018)	31.3	58.0	13.7	22.1
UpDown (2018)	39.1	62.4	15.1	34.5
AReg(2017)	42.0	65.5	15.9	36.6
AdvReg (2019)	42.3	59.7	14.8	40.8
RUBi (2019)	47.1	68.7	20.3	43.2
Teney and van den Hengel (2019)	46.0	58.2	29.5	44.3
Unshuffling (2020b)	42.4	47.7	14.4	47.3
UpDn+CE+GS (2020a)	46.8	64.5	15.4	45.9
LXMERT (2019)	46.2	42.8	18.9	55.5
SCR (2019)	48.4	70.4	10.4	47.3
LMH (2019)	52.4	69.8	44.5	45.5
CSS (2020)*	58.9	84.4	49.4	48.2
MUTANT (2020a)*	**69.5**	**93.2**	**67.2**	**57.8**
ZSL+Objects+UpDown	40.8	67.4	28.6	30.2
ZSL+Patches+UpDown	41.2	68.5	29.8	30.0
ZSL+Patches+Encoder	47.3	73.4	39.8	35.6

Table 1: Unsupervised accuracy on VQA-CP-v2 test set. All baselines are *supervised* methods trained on the train split. * use further additional supervised training samples. Cyan: our model is better overall. Red: our model is better on specific categories.[2]

Model	All	Yes-No	Num	Others
GVQA (2018)	48.2	72.0	31.1	34.7
UpDown (2018)	65.3	81.8	44.2	56.1
RUBi (2019)	63.1	*	*	*
MCAN (2019)	70.4	85.8	53.7	60.7
VilBERT (2019)	70.5	*	*	*
LXMERT (2019)	72.5	**88.2**	**54.2**	**63.1**
UNITER (2019)	**72.7**	*	*	*
ZSL + Objects + UpDown	41.4	68.1	27.6	29.4
ZSL + Patches + UpDown	40.6	67.8	28.4	29.2
ZSL + Patches + Encoder	46.8	72.1	34.4	34.1
FSL + Objects + UpDown	66.8**	82.4**	45.1**	56.4**
FSL + Patches + UpDown	63.4	80.2	45.2	52.1
FSL + Patches + Encoder	65.3	80.5	48.94	56.2

Table 2: VQA-v2 Test-standard accuracies[2]. FSL models are pretrained on synthetic samples, and further finetuned on VQA-v2 train split. * - Scores are not available, ** - Validation split scores.

Baselines: To measure the improvements due to our proposed image patch features and SWA loss, we compare our methods to the UpDown model Anderson et al., which uses object bounding-box features. For the Zero-shot transfer setting, we compare our Encoder with UpDown when trained with spatial features as well as object features. Pretrained transformers such as UNITER use large V&L corpora, dense human annotations for objects and Q-A pairs and supervised loss functions over these. Comparisons with such models are therefore not fair in a ZSL setting; instead, we perform these comparisons in a fully-supervised (FSL) setting.

Model	All	Binary	Open
CNN + LSTM (2018)	46.6	61.9	22.7
UpDown (2018)	49.7	66.6	34.8
MAC (2018)	54.1	71.2	38.9
BAN (2018)	57.1	76.0	40.4
LXMERT (2019)	**60.3**	**77.8**	**45.0**
ZSL + Objects + UpDown	30.7	50.8	17.6
ZSL + Patches + UpDown	31.1	52.3	16.8
ZSL + Patches + Encoder	33.7	55.5	21.2
FSL + Objects + UpDown	50.4	67.5	35.1
FSL + Patches + UpDown	46.4	64.3	31.4
FSL + Patches + Encoder	55.2	73.6	38.8

Table 3: GQA Validation split accuracies.[2]

6 Results[2]

Unsupervised Question Answering: Tables 1, 2 and 3 summarize our results on the three benchmark datasets. We can observe that our method outperforms specially designed supervised methods for bias removal in VQA-CP; our model with UpDown is 1.1% better than the supervised UpDown. Under the ZSL setting for VQA-CP, our Encoder model is 6.1% better than UpDown with patches, and 6.5% better than UpDown with Object features, for VQA-v2: 6.2%, 5.4% respectively, and for GQA: 2.2%, 3.0% respectively.

For VQA-CP, our procedurally generated Q-A pairs and patch-features when used with either UpDown or Encoder are better than the baseline supervised UpDown model, showing the improvements are model-agnostic. This also shows the merits of using our Q-A generation methods when train and test-sets deviate linguistically.

Most GQA questions require understanding spatial relationships between objects. Such questions are infrequent in our synthetic training data since captions do not contain detailed spatial relationships among objects. Thus, the ZSL performance is not as competitive for GQA when compared to our performance on VQA and VQA-CP. Improving spatial and compositional question-answering with weak supervision is an interesting future pursuit.

Fully Supervised Question Answering: In the FSL setting, our methods' performance is not far from SOTA methods, even though our method uses significantly fewer annotations (no access to object bounding boxes). In GQA, the Encoder model performs on par with MAC (2018) and BAN (2018), which unlike us, use object relationship annotations. This suggests that cross-modal transformer layers can learn spatial relations from spatial pyramidal

---

[2] ZSL refers to zero-shot transfer setting and FSL refers to our models further finetuned on the respective train split. Underline⇒unsupervised best, **bold**⇒overall best. Baselines are trained on train-split, our models on synthetic data.

	Question Generation	VQA-v2	VQA-CP	GQA
Updn	Template	26.2	25.7	11.6
	Template + Para&Back	28.5	27.1	14.8
	QA-SRL	31.1	33.8	18.9
	All	41.4	40.2	31.1
Encoder	Template	32.5	31.3	18.5
	Template + Para&Back	34.8	33.6	23.6
	QA-SRL	40.3	39.8	21.4
	All	**47.1**	**46.8**	**33.7**

Table 4: Effect of different pre-training data sources on ZSL Validation split accuracies.

	Patch Resolutions	VQA-v2	VQA-CP	GQA
UpDn	{1}	18.8	19.7	11.3
	{1, 3}	36.7	35.9	24.5
	{1, 3, 5}	40.1	39.7	29.5
	{1, 3, 5, 7}	**41.4**	**40.2**	**31.1**
	{1, 3, 5, 7, 9}	39.8	38.4	29.3
Encoder	{1}	26.4	27.7	15.3
	{1, 3}	42.6	43.1	28.8
	{1, 3, 5}	44.3	45.2	30.9
	{1, 3, 5, 7}	**47.1**	**46.8**	**33.7**
	{1, 3, 5, 7, 9}	46.2	45.4	31.2

Table 5: Effect of the number of spatial patches on ZSL performance {3,5} implies division of the image into a 3x3 and 5x5 grid of patches.

features.

Impact of each question-generation technique: In Table 4 we can observe the effect of different question generation techniques. All models use spatial image patch features. QA-SRL based questions and the SWA-Loss contribute the most towards gains in performance, and the paraphrased questions provide larger linguistic variation.

Effect of Spatial Pyramids: We study the effect of progressively increasing the number of spatial image patches (i.e., decreasing the patch size). Table 5 shows that an optimum exists at grid-size of 7×7 after which the addition of smaller patches is detrimental. Similarly, only using patches of large size does not allow models to focus on specific image regions. Thus a trade-off exists between global context and region-specific features. Changing the feature extractor from ResNet-50 to ResNet-101 only results in a minor improvement of 0.01% to 0.30%. Removing visual position embeddings has a significant effect on performance, with a drop of 4.60% to 8.00% in both ZSL and FSL settings.

Impact of Pre-training Tasks: Table 6 shows the effect of different pretraining tasks on the downstream zero-shot transfer VQA task. We need the

Pre-Training Task	VQA-v2	VQA-CP	GQA
SWA	39.1	38.3	25.4
MLM+SWA	42.4	41.5	27.8
MQA+SWA	42.0	41.2	26.6
MLM+MQA+SWA	45.6	44.9	29.7
MLM+ITM+SWA	44.7	43.6	28.9
All	**46.2**	**45.4**	**31.2**

Table 6: Effect of different pre-training tasks on the ZSL performance for the Encoder model.

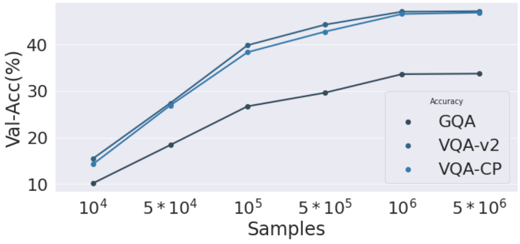

Figure 5: Learning Curve showing validation accuracy vs. number of synthetically generated training samples.

SWA task, as it is used to perform the zero-shot QA task. The combination of MLM, MQA, and ITM, all of which need image understanding, shows improved performance on the downstream task, indicating better cross-modal representations.

Effect of size of synthetic training set: Figure 1 shows our Encoder model's learning curve for the zero-shot transfer setting trained on our synthetic Q-A pairs. The performance stagnates after a critical threshold of 10^6 samples is reached. Our experiments also suggest that randomly sampling a set of questions for each image per epoch leads to a 4% gain compared to training on the entire set.

Error Analysis: Our ZSL method is pretrained on longer phrases and hence tends to generate more detailed answers, such as "red car" instead of "car". Although the SWA loss is designed to encourage a distribution over the shorter phrases, the bias is not entirely removed. On automated evaluation, we observe that for 42% of questions, the target answer is a sub-phrase of our predicted answer. Manual evaluation of 100 such samples shows that 87% of such detailed predicted answers are plausible. This shows the relevance of learning from captions and quantifies the bias towards short "true" answers in human-annotated benchmarks, calling for better evaluation metrics that do not penalize VQA systems for producing descriptive or alternative accurate answers.

In the FSL setting, we either finetune our pre-trained QA classifier with the SWA Loss or train a separate feedforward layer from scratch for the task. The pre-trained QA classifier predicts longer phrases as answers, leading to a drop in accuracy. The feedforward layer performs better (+6%), indicating our Encoder captures relevant features necessary to generalize to the benchmark answer-space. Note that we do not use object annotations during training, unlike existing methods.

Our error analysis and Figure 3 show the shift in question-space and answer-space between synthetic and human-authored Q-A pairs. These (along with inadequate evaluation metrics) act as the primary sources explaining the performance-gap between weakly-supervised methods and the fully-supervised setting. It remains to be seen whether more sophisticated question generation can be developed to reduce the performance gap further and mitigate the heavy reliance on human annotations.

7 Discussion and Conclusion

Prior work (Chen et al., 2019; Jiang et al., 2020) has demonstrated that the use of object bounding-boxes and region features leads to significant improvements on downstream tasks such as captioning and VQA. However, little effort has been dedicated to developing alternative methods that can approach similar performance without relying on dense annotations. We argue that weakly supervised learning coupled with data synthesis strategies could be the pathway for the V&L community towards a "post-dataset era".[2] In this work, we take a step towards that goal. We address the problem of weakly-supervised VQA with a framework for the procedural synthesis of Q-A pairs from captions for training VQA models, where benchmark datasets can be used only for evaluation. We use spatial pyramids of patch features to increase the annotation efficiency of our methods. Our experiments and analyses show the potential of patch-features and procedural data synthesis and reveal problems with existing evaluation metrics.

Ethical Considerations

Captions and Question-Answer pairs are both annotated by humans in existing image captioning and visual question answering datasets. However, captions arguably contain a lesser degree of subjectivity, ambiguity, and linguistic biases than VQA annotations, due to the design of annotation prompts that limit the introduction of these biases. Our work points to the potential of procedurally generated annotations in providing robustness improvements under changing linguistic priors in VQA test sets (Table 1). Hendricks et al. find that gender bias exists in image-captioning datasets and is *amplified* by models; further research in self-supervised data synthesis could potentially help alleviate such social biases.

Acknowledgements

The authors acknowledge support from the DARPA SAIL-ON program W911NF2020006, ONR award N00014-20-1-2332, and NSF grant 1816039, and the anonymous reviewers for their insightful discussion.

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4971–4980. IEEE Computer Society.

Aishwarya Agrawal, Aniruddha Kembhavi, Dhruv Batra, and Devi Parikh. 2017. C-vqa: A compositional split of the visual question answering (vqa) v1.0 dataset. *arXiv preprint arXiv:1704.08243*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6077–6086. IEEE Computer Society.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2425–2433. IEEE Computer Society.

Pratyay Banerjee and Chitta Baral. 2020. Self-supervised knowledge triplet learning for zero-shot question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 151–162, Online. Association for Computational Linguistics.

Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. 2021. Self-supervised test-time learning for reading

---

[2] A. Efros, *Imagining a post-dataset era*, ICML'20 Talk.

comprehension. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1200–1211, Online. Association for Computational Linguistics.

Nilavra Bhattacharya, Qing Li, and Danna Gurari. 2019. Why does a visual question have different answers? In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4270–4279. IEEE.

Rémi Cadène, Corentin Dancette, Hedi Ben-younes, Matthieu Cord, and Devi Parikh. 2019. Rubi: Reducing unimodal biases for visual question answering. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 839–850.

Wei-Lun Chao, Hexiang Hu, and Fei Sha. 2018. Cross-dataset adaptation for visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5716–5725. IEEE Computer Society.

Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. 2020. Counterfactual samples synthesizing for robust visual question answering. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10797–10806. IEEE.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4069–4082, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4508–4513, Online. Association for Computational Linguistics.

Zhiyuan Fang, Tejas Gokhale, Pratyay Banerjee, Chitta Baral, and Yezhou Yang. 2020a. Video2Commonsense: Generating commonsense descriptions to enrich video captioning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 840–860, Online. Association for Computational Linguistics.

Zhiyuan Fang, Shu Kong, Zhe Wang, Charless Fowlkes, and Yezhou Yang. 2020b. Weak supervision and referring attention for temporal-textual association learning. *arXiv preprint arXiv:2006.11747*.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale QA-SRL parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2051–2060, Melbourne, Australia. Association for Computational Linguistics.

Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 580–587. IEEE Computer Society.

Tejas Gokhale, Pratyay Banerjee, Chitta Baral, and Yezhou Yang. 2020a. MUTANT: A training paradigm for out-of-distribution generalization in visual question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 878–892, Online. Association for Computational Linguistics.

Tejas Gokhale, Pratyay Banerjee, Chitta Baral, and Yezhou Yang. 2020b. Vqa-lol: Visual question answering under the lens of logic. In *European Conference on Computer Vision (ECCV)*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6325–6334. IEEE Computer Society.

Gabriel Grand and Yonatan Belinkov. 2019. Adversarial regularization for visual question answering: Strengths, shortcomings, and side effects. In *Proceedings of the Second Workshop on Shortcomings in Vision and Language*, pages 1–13, Minneapolis, Minnesota. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. 2018. Women also snowboard: Overcoming bias in captioning models. In *European Conference on Computer Vision*, pages 793–811.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan C. Russell. 2017. Localizing moments in video with natural language. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5804–5813. IEEE Computer Society.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6700–6709. Computer Vision Foundation / IEEE.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs.

Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik G. Learned-Miller, and Xinlei Chen. 2020. In defense of grid features for visual question answering. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10264–10273. IEEE.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1988–1997. IEEE Computer Society.

Kushal Kafle and Christopher Kanan. 2017. An analysis of visual question answering algorithms. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1983–1991. IEEE Computer Society.

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. 2020. Learning the difference that makes A difference with counterfactually-augmented data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Anna Khoreva, Rodrigo Benenson, Jan Hendrik Hosang, Matthias Hein, and Bernt Schiele. 2017. Simple does it: Weakly supervised instance and semantic segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1665–1674. IEEE Computer Society.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1571–1581.

Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. 2019. Information maximizing visual question generation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2008–2018. Computer Vision Foundation / IEEE.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019a. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019b. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. 2018. Visual question generation as dual task of visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6116–6124. IEEE Computer Society.

Zhongli Li, Wenhui Wang, Li Dong, Furu Wei, and Ke Xu. 2020. Harvesting and refining question-answer pairs for unsupervised QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6719–6728, Online. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L. Yuille. 2019. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4185–4194. Computer Vision Foundation / IEEE.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13–23.

Mateusz Malinowski and Mario Fritz. 2014. Towards a visual turing challenge. In *Learning Semantics 2014*.

Niluthpol Chowdhury Mithun, Sujoy Paul, and Amit K. Roy-Chowdhury. 2019. Weakly supervised video moment retrieval from text queries. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11592–11601. Computer Vision Foundation / IEEE.

Aditya Mogadala, Marimuthu Kalimuthu, and Dietrich Klakow. 2019. Trends in integration of vision and language research: A survey of tasks, datasets, and methods. *arXiv preprint arXiv:1907.09358*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 1143–1151.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Santhosh K. Ramakrishnan, Ambar Pal, Gaurav Sharma, and Anurag Mittal. 2017. An empirical evaluation of visual question answering for novel objects. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7312–7321. IEEE Computer Society.

Arijit Ray, Karan Sikka, Ajay Divakaran, Stefan Lee, and Giedrius Burachas. 2019. Sunny and dark outside?! improving answer consistency in VQA through entailed question generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5860–5865, Hong Kong, China. Association for Computational Linguistics.

Mengye Ren, Ryan Kiros, and Richard S. Zemel. 2015a. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2953–2961.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015b. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99.

Steven Rennie, Etienne Marcheret, Neil Mallinar, David Nahamoo, and Vaibhava Goel. 2020. Unsupervised adaptation of question answering systems via generative self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1148–1157, Online. Association for Computational Linguistics.

Marco Tulio Ribeiro, Carlos Guestrin, and Sameer Singh. 2019. Are red roses red? evaluating consistency of question-answering models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6174–6184, Florence, Italy. Association for Computational Linguistics.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3).

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ramprasaath R. Selvaraju, Purva Tendulkar, Devi Parikh, Eric Horvitz, Marco Túlio Ribeiro, Besmira Nushi, and Ece Kamar. 2020. Squinting at VQA models: Introspecting VQA models with subquestions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10000–10008. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia. Association for Computational Linguistics.

Robik Shrestha, Kushal Kafle, and Christopher Kanan. 2019. Answer them all! toward universal visual question answering models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10472–10481. Computer Vision Foundation / IEEE.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Hyun Oh Song, Ross B. Girshick, Stefanie Jegelka, Julien Mairal, Zaïd Harchaoui, and Trevor Darrell. 2014. On learning to localize objects with minimal supervision. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1611–1619. JMLR.org.

Amos Storkey. 2009. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, pages 3–28.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Damien Teney, Ehsan Abbasnedjad, and Anton van den Hengel. 2020a. Learning what makes a difference from counterfactual examples and gradient supervision. *arXiv preprint arXiv:2004.09034*.

Damien Teney, Ehsan Abbasnejad, and Anton van den Hengel. 2020b. Unshuffling data for improved generalization. *arXiv preprint arXiv:2002.11894*.

Damien Teney and Anton van den Hengel. 2019. Actively seeking and learning from live data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1940–1949. Computer Vision Foundation / IEEE.

Damien Teney and Anton van den Hengel. 2016. Zero-shot visual question answering. *arXiv preprint arXiv:1611.05546*.

Damien Teney, Kushal Kafle, Robik Shrestha, Ehsan Abbasnejad, Christopher Kanan, and Anton van den Hengel. 2020c. On the value of out-of-distribution testing: An example of goodhart's law. *arXiv preprint arXiv:2005.09241*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2019. Meta-learning to detect rare objects. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9924–9933. IEEE.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Jialin Wu and Raymond J. Mooney. 2019. Self-critical reasoning for robust visual question answering. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8601–8611.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yiming Xu, Lin Chen, Zhongwei Cheng, Lixin Duan, and Jiebo Luo. 2020. Open-ended visual question answering by multi-modal domain adaptation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 367–376, Online. Association for Computational Linguistics.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2016. Stacked attention networks for image question answering. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 21–29. IEEE Computer Society.

Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. 2019. Deep modular co-attention networks for visual question answering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6281–6290. Computer Vision Foundation / IEEE.

Hanwang Zhang, Zawlin Kyaw, Jinyang Yu, and Shih-Fu Chang. 2017. PPR-FCN: weakly supervised visual relation detection via parallel pairwise R-FCN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4243–4251. IEEE Computer Society.

Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and yang: Balancing and answering binary visual questions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5014–5022. IEEE Computer Society.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2921–2929. IEEE Computer Society.

Appendix

A Synthesized Samples

Table 7 shows illustrative examples of Q-A pairs procedurally generated from the image caption using template-based method. Table 8 shows the use of two transformations (T): negation and adversarial words (Gokhale et al., 2020b) two generate more sentences. Thus the negation of Q or substitution of a word in Q with an adversarial word results in the new question-answer pair Q_{new}, A_{new}. To increase the linguistic diversity of the questions we use paraphrasing as shown in Table 11.

B Dataset Analysis

In Table 9, we compare the distribution per answer-type of our synthetically generated samples with the distribution in the VQA-CP-v2 (Agrawal et al.,

Image	Question	Answer
	What are set on the sidewalk outside a veterinary hospital?	bags
	What is the young man holding up in front of his face ?	phone
	What is almost empty on the table	glass
	What drawn carriage with passengers in the city	horse
	What is the color of the table ?	white
	What is the color of the eyes ?	blue
	How many boats anchored by ropes close to shore?	8

Table 7: Examples of template-based data synthesis

T	Image	Q	A	Q_{new}	A_{new}
Negation		Is this bread?	yes	Is this not bread	no
	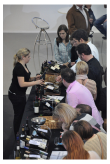	What is the color of the woman's shirt?	black	What is not the color of the woman's shirt?	white
		Is there a boy?	no	Is there no boy?	yes
Adversarial		Who is sitting in the boat ?	man	Who is sitting in the dining table ?	can't say
		How big is the plane ?	large	How big is the car ?	size
	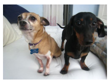	How many puppies are on the bed ?	two	How many cats are on the bed?	none

Table 8: The effect of using transformations (T) to create new Q-A pairs

Category	VQA-CP (%)	Pretraining (%)
Yes/No	41.86	50.18
Number	11.91	8.32
Other	46.23	41.45

Table 9: Distribution of samples by answer-type in our pre-training dataset and the VQA-CP evaludation dataset.

Hyper-Parameters	Model
Batch Size	32-128
Learning Rate	$(1e^{-5}, 5e^{-5})$
Dropout	0.1
Language Layers	6
Cross-Modality Layer	4 — 12
Optimizer	BertAdam
Warmup	0.1
Max Gradient Norm	5.0
Max Text Length	30
ResNet	50 / 101 / 152
Epochs	10-40

Table 10: Hyper-Parameters for our models

2018) dataset. Since we use our synthetic samples as the pre-training data, and do not use VQA-CP samples for training in our zero-shot setup, this comparison displays the shift between the training (synthetic) and test (human annotated VQA-CP) datasets.

We further analyze this shift, by computing the t-SNE projections of questions using mean-pooled Glove (Pennington et al., 2014) embeddings for our generated questions and observe the overlap with human-authored questions in VQA and GQA (Hudson and Manning, 2019). Figure 6. We observe a marked shift between the question clusters for our procedurally generated questions and human annotated questions from VQA and GQA.

Similarly, we also show the distribution of answers in our dataset in Figure 7. It can be seen that our dataset has a slight imbalance in the proportion of questions with answer "yes" and "no". Numeric answers *0,1,2,3* are most frequent. Answers about people such as *man, woman, people, person, group of people* are also more common in the dataset. The remaining answers have a long-tailed distribution, since there are $\sim 90k$ unique answers in our dataset compared to $\sim 3.5k$ in VQA and $\sim 2k$ in GQA.

C Training Details

We use the HuggingFace (Wolf et al., 2019) and PyTorch frameworks (Paszke et al., 2019). Hyper-parameters and other training settings are given in Table 10.

Image	**Q**	**A**	$\mathbf{Q_{new}}$	$\mathbf{A_{new}}$
	How is something parked ?	illegally	How’s-what’s parked?	illegally
	what does something seem to do ?	park	What do you think something seems to be doing?	park
	Where was parked something?	behind a legally parked car	Do you know where something was parked?	behind a legally parked car
	How many cars are visible ?	2	How many cars are we looking at?	2
	Is there two cars parked on the sidewalk on the street ?	Yes	There are two cars parked on the sidewalk, right?	Yes

Table 11: Illustration of using paraphrasing to improve the linguistic variation of our questions and answers.

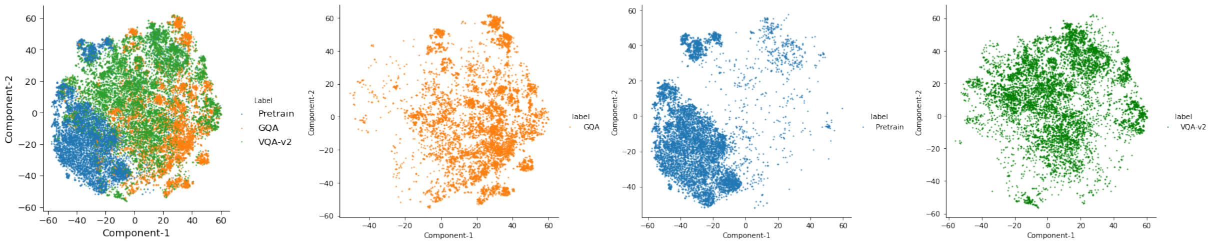

Figure 6: t-SNE projections of Glove embedding our generated questions, and human-authored VQA-v2 and GQA questions. Blue: our pretraining dataset, Orange: GQA, Green: VQA. L-R: All, GQA, Pretrain, VQA.

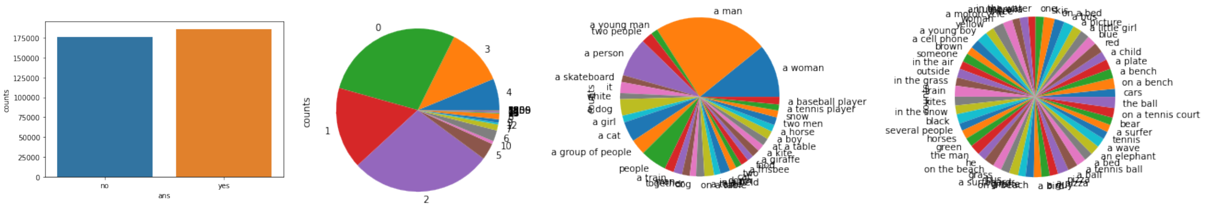

(a) “Yes-No”

(b) “Numeric”

(c) ”Other”: Highly frequent answers (count > 500)

(d) “Other”:Answers with count between 200 and 500

Figure 7: Distribution of most frequent answers in our Pretraining dataset for each answer-type (yes-no, numeric, and other). Please zoom for details.

How well do you know your summarization datasets?

Priyam Tejaswin [*] **Dhruv Naik** **Pengfei Liu** [†]
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA
`{ptejaswi, drn, pliu3}@andrew.cmu.edu`

Abstract

State-of-the-art summarization systems are trained and evaluated on massive datasets scraped from the web. Despite their prevalence, we know very little about the underlying characteristics (data noise, summarization complexity, etc.) of these datasets, and how these affect system performance and the reliability of automatic metrics like ROUGE. In this study, we manually analyse 600 samples from three popular summarization datasets. Our study is driven by a six-class typology which captures different noise types (missing facts, entities) and degrees of summarization difficulty (extractive, abstractive). We follow with a thorough analysis of 27 state-of-the-art summarization models and 5 popular metrics, and report our key insights: (1) Datasets have distinct data quality and complexity distributions, which can be traced back to their collection process. (2) The performance of models and reliability of metrics is dependent on sample complexity. (3) Faithful summaries often receive low scores because of the poor diversity of references. We release the code, annotated data and model outputs.[1]

1 Introduction

The past few years have witnessed major breakthroughs and improvements in automatic summarization (See et al., 2017; Celikyilmaz et al., 2018; Jadhav and Rajan, 2018; Liu and Lapata, 2019; Liu, 2019; Dou et al., 2020; Yuan et al., 2021; Liu et al., 2021). Apart from the improvements in the summarization model architectures (Zhang et al., 2019; Zhong et al., 2020), this growth has been aided by large-scale datasets (Nallapati et al., 2016; Narayan et al., 2018a; Sharma et al., 2019) and automatic evaluation metrics (Lin, 2004; Zhao et al., 2019; Kryscinski et al., 2020) which are used for tuning hyperparameters and comparing models. While the reliability of these metrics has been explored extensively (Peyrard, 2019; Bhandari et al., 2020; Fabbri et al., 2020), few studies have focused on the underlying characteristics of different datasets, and how these impact model performance and metric reliability.

Datasets like CNN/DailyMail (Nallapati et al., 2016), Gigaword (Rush et al., 2015), XSum (Narayan et al., 2018a), and many more (Wang and Ling, 2016; Koupaee and Wang, 2018; Kim et al., 2019; Ganesan et al., 2010) were collected by scraping a large collection of web-pages. And for all the benefits this approach offers (seemingly infinite samples, diverse subjects, etc) there are some caveats:

Data Noise We have no idea about the **noise** in the dataset. In the context of text summarization, noise could be an incomplete or irrelevant reference. At the moment, its quantity and impact on the performance is unknown.

Summarization Complexity What do we really know about the nature of samples in the dataset? Gigaword is a headline generation dataset with short sources and references. Does this imply a higher volume of simpler (i.e. more *extractive*) samples? The degree of summarization complexity, and its impact on model performance is unknown.

Exploring these open questions is critical for two reasons: (1) Information about the noise could lead to more informed data collection and preprocessing methods: in a recent study, Kryscinski et al. (2019) quantified HTML artefacts in popular summarization datasets, and proposed ways to detect and remove them. (2) Awareness about the complexity could better explain model performance, metrics, and even lead to new model architectures. In the tasks of machine comprehension

---

[*] This author was the primary contributor.

[†] Corresponding author.

[1] https://github.com/priyamtejaswin/howwelldoyouknow

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3436–3449
August 1–6, 2021.

and question answering, Chen et al. (2016) and Yatskar (2019) manually inspected random samples and drew insights which led to new state-of-the-art models. Such analysis could also help researchers choose datasets and metrics more carefully.

In this study, we perform intrinsic and model-centric evaluation of three popular summarization datasets (Gigaword, CNN/DM and XSum). We are interested in answering the following questions:

Q1. What are the underlying intrinsic properties of summarization datasets? We are interested in (1) Identifying and quantifying the different types of "noise" that could occur and could penalize models. (2) Whether samples have different levels of difficulty. Armed with this, we ask the following questions.

Q2 a. How do these properties impact model performance? Specifically, we'd like to know (1) If, and how, the performance varies across the different types of samples discovered from Q1. (2) If the performance is consistent across metrics.

Q2 b. If the reliability of metrics changes with these properties? This is motivated (in part) from prior metric-analysis studies, where researchers have explored inter-metric agreement and alignment with human-judgement under different conditions (Peyrard, 2019; Bhandari et al., 2020). Here we are more interested in knowing if the metrics are more correlated with human judgement for simpler samples, than complex ones.

Large-scale automatic intrinsic dataset evaluation has been explored with some promising results (Bommasani and Cardie, 2020). However, these methods rely on heuristics like content-value, density and compression (Grusky et al., 2018). We are interested in a more fine-grained, interpretable analysis that can only come from manual inspection, much like the analysis by Chen et al. (2016) and by Yatskar (2019). To that end, we first define a six-class typology: the first three classes cover types of data-noise and the last three cover varying degrees of summarization difficulty. We then proceed to answer the aforementioned research questions, and discuss our key observations which are summarized below:

Key Observations: (1) Datasets have distinct modalities – a mix of simpler samples (which we call *Extractive*) and complex ones (which we call *Paraphrase* and *Inference*. (2) Gigaword is *majorly* Extractive but suffers from data noise (45% of the targets have some key entity, or fact that is absent from the source). (3) CNN/DM is relatively cleaner, and the authors' attempts to create a more abstractive dataset seems to be successful compared with Gigaword (only 18% of samples are Extractive). (4) XSum has no Extractive samples, but also has the greatest fraction of noise: 54% of the test samples have key entities or facts missing from the source. (5) Within the datasets, the broad performance trends between the typology classes are *consistent across all metrics*: **simpler samples score higher than complex ones**. (6) Metric reliability is also complexity dependent: On CNN/DM the agreement with human judgement decreases as summarization complexity increases.

The remainder of the paper is organised as follows: in Section 2 we answer **Q1**, describe the three datasets, define the typology, and present results from the annotation. In Section 3 we explore **Q2 a.** and evaluate different models on a variety of metrics (automatic and human-judgement). In Section 4 we explore **Q2 b.** and investigate metric reliability. In Section 5 we share some learnings from our experience. We conclude with Section 7.

2 Evaluating the intrinsic properties of summarization datasets (Q1)

	Length(Doc)		**Length(Ref)**		**Sample**	
	train	test	train	test	train	test
Gigawords	31	29	8	8	3.8M	1.9K
CNNDM	691	682	51	54	287K	11K
XSum	374	376	21	21	204K	11.3K

Table 1: Statistics of the three datasets. Length refers to the average number of words per Document/Reference.

2.1 Datasets for Annotation

Among many summarization datasets, we choose the following:

Gigaword is a summarizaiton dataset extracted from news articles (Rush et al., 2015)[2].

CNN/DailyMail or "CNN/DM" question answering dataset (Hermann et al., 2015; Nallapati et al., 2016) is commonly used for summarization. The dataset consists of online news articles paired with human-generated summaries.[3]

---

[2] We use the version most commonly used by summarization systems: https://github.com/harvardnlp/sent-summary

[3] We use the non-anonymized data as See et al. (2017).

Label	Source Dataset	Source	Target	State-of-the-art Model Output
Incomplete / Irrelevant	Gigaword	Andre Blom and Mark Scharrenberg scored tries and some tactical kicks in the final 10 minutes sent the United States to the Rugby World Cup with a 21-16 victory over Uruguay on Saturday .	London testing , please ignore .	United States beats Uruguay 21-16 in Rugby World Cup .
Entity Missing	Gigaword	The United States claimed credit Tuesday for a ceasefire that ended fighting between Israel and Lebanese guerrillas , and rejected suggestions that it was forced to model the agreement after a French draft .	US takes the credit for Israel-Hezbollah ceasefire by Carole Landry .	Us claims credit for lebanon ceasefire .
Extractive	CNN-DM	Ed Miliband's US adviser pays no tax in Britain on his reported £300,000 salary, he has admitted. David Axelrod masterminded two presidential election victories for Barack Obama and was hired by the Labour leader amid great fanfare last year. He has helped refine Mr Miliband's message ...*(truncated)* ... have been aware of Labour's eye-catching crackdown on non-doms last week. But speaking in the US where he is promoting his autobiography, Mr Axelrod revealed he is not resident for tax purposes in the UK. Asked whether he pays tax in Britain, he told the Daily Telegraph: 'I don't do my accounting so I don't know but I'm not in residence there.' Labour confirmed it pays Mr Axelrod in dollars through his consultancy firm and that he 'lives in the US, works in the US and pays taxes in the US'. ... (*truncated*)	David Axelrod masterminded two Obama presidential election victories . He was hired by Labour leader Ed Miliband amid great fanfare last year . Revealed at a book launch that he is not resident for tax purposes in UK . Labour confirms it pays Mr Axelrod in dollars through consultancy firm .	David Axelrod masterminded two presidential election victories for Barack Obama . He was hired by the Labour leader amid great fanfare last year . Has helped refine Mr Miliband 's message about tackling the cost of living and making sure the wealthy pay their fair share . Mr Axelrod makes infrequent visits to the UK to meet Mr Miliband and offers advice by phone .
Paraphrase	CNN-DM	The number of women in Britain becoming nuns is at a 25-year high. Figures from the Catholic Church show the number of women taking Holy Vows has trebled from 15 in 2009 to 45 last year. From a low of only seven in 2004, the figure has been rising for the past decade. Theodora Hawksley, 29, was until recently a post-doctoral researcher in theology at the University of Edinburgh. But at the beginning of the year she decided to become a nun. *(truncated)*. Far from being trapped in traditional habits, Miss Hawksley said her order tends to dress down in T-shirts and jeans. Father Christopher Jamison, director of the National Office for Vocation of England and Wales, said: 'There is a gap in the market for meaning in our culture. One of the ways women may find that meaning is through religious life.' Sister Cathy Jones, religious life vocations promoter at the office, said: *(truncated)* .	Figures from the Catholic Church show more and more becoming nuns . The number of women taking Holy Vows stood at just seven back in 2004 . But that figure had risen to 15 in 2009 and increased further to 45 last year . One father said a ' gap in the market for meaning ' led people toward religion .	Figures from Catholic Church show number of women taking Holy Vows has trebled from 15 in 2009 to 45 last year . From a low of seven in 2004 , the figure has been rising for the past decade . Theodora Hawksley , 29 , was until recently a post - doctoral researcher in theology at the University of Edinburgh . But at the beginning of the year she decided to become a nun .
Inference	Gigaword	Three Malaysian and Indonesian seamen kidnapped by Philippine Abu Sayyaf kidnap-for-ransom group allegedly had been executed and the skeletons discovered in the southern Philippines are believed to be their remains , a local television reported Wednesday .	Abu Sayyaf hostages allegedly executed : report .	3 filipino , Indonesian seamen executed in southern Philippines .

Table 2: Examples for each of the six categories. Text spans with the same colors correspond to the same fact in the source and target. Target spans in RED are missing or unsupported in the source. The last sample is "Inference" because the writer will have to understand the concept of hostages, and then generalise from the group to an individual.

XSum or "Extreme Summarization" (Narayan et al., 2018a) was constructed from online news articles for highly abstractive summarization.

We consider these datasets because of their popularity, and the difference in the nature of samples. The latter enables a more comprehensive analysis; Table 1 captures the size of source and target documents along with the number of samples.

2.2 Typology Definition

The classes are defined below in order of priority. Some examples are in Table 2. Readers may refer to the Appendix B, C, D for more examples.

- **Incomplete/Irrelevant**: The target summary ends abruptly. Or the source and target are unrelated.
- **Entity Missing**: The target summary contains entities (names, dates, events, etc) that are absent from the source.
- **Evidence Missing**: The target summary is based on concepts which are absent from the source. However, the target is not **Incomplete** and all **Entities** are present.
- **Extractive**: The target is constructed by copying tokens from the source, mostly in-order of their appearance. Minor modifications, like stemming and abbreviating, are permitted. Word substitutions, and additions, are limited to a few. No reasoning, conclusion or co-ref resolution is performed as part of the summarization. The complete context of the target should be present in the source.
- **Paraphrase**: The majority of tokens in the target are substituted, or appear out of order, or both. There is no reasoning, conclusion or co-ref resolution. The complete context of the

target should be present in the source.
- **Inference**: A non-trivial "inference" activity has to be completed to construct the target: some reasoning, conclusion, or complex co-reference resolution. The complete context of the target should be present in the source.

We annotate 200 samples from each dataset, on par with similar studies on intrinsic evaluation (Chen et al., 2016; Cao et al., 2017). Two authors annotate samples independently. Annotations matched for 70%, 68% and 73% of Gigaword, CNN-DM and XSum samples, respectively. Disagreements were discussed between all authors before arriving at a consensus for the final label.

2.2.1 Motivation and Advantages

To the best of our knowledge, summarization datasets have not been manually analysed in this manner. A review of the most relevant summarization dataset analysis research shows that the most common form of intrinsic evaluation is to use surface-level heuristics. Most studies only cover a part of our typology, while almost all studies ignore the noise present in datasets.

Coverage , Density, Redundancy Grusky et al. (2018); Bommasani and Cardie (2020); Zhong et al. (2019b) use similar forms of token-level coverage between the source and the reference to measure the *extractiveness* of the summary. In it's simplest form, this is a ratio of the number of overlapping tokens and reference length. In our definition of **Extractive**, we first set a meaninful, well-defined criterion, and then manually check for *extractive* references, while allowing for some relaxations.

Content Compression In most papers (Grusky et al., 2018; Zhong et al., 2019b; Bommasani and Cardie, 2020), the summarization complexity is defined by a compression ratio (usually the normalized word-count ratio of the source and reference). As a standalone metric, this does indeed capture the difficulty in replication. However, token rearrangement, substitution, reformulation is ignored in this measure of "complexity". To combat this, we distinctly defined **Paraphrase** and **Inference**. By manually analysing samples, we are able to differentiate between the obviously simple Extractive samples, the relatively tougher Paraphrase samples and the most difficult Inference samples. Together these three offer a highly intuitive classification of samples. Part of the reason that the Machine Comprehension analysis by Chen et al. (2016) was so effective was the interpretability of their classes. We hope our analysis will also enable researchers to improve summarization models.

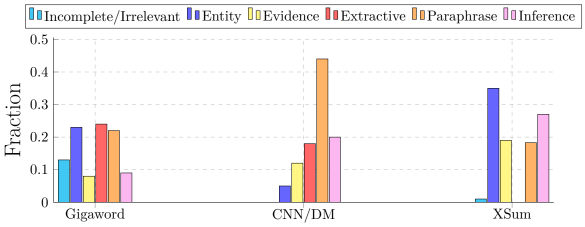

Figure 1: Distribution of the different class of samples in all datasets.

Noise Prior works have not focused on quantify the noise in popular datasets. Moreover, none of these metrics are designed to account for noise or factual inconsistencies. A high value for content compression might imply a high-degree of summarization complexity. But this ignores the possibility that the source-reference pair is unrelated (like row 1 in Table 2). In addition, the manual analysis allows us to identify factual errors and co-ref errors.

This is not to say the typology is perfect and exhaustive. Limitations and possible extensions to our typology are discussed in Section 5.

2.3 Dataset Analysis

The distribution of classes in the datasets is in Figure 1. We have made the following **key observations** in our analysis of the labels.

Gigawords is Extractive, but very noisy. 24.5% of summaries are Extractive, but 44.5% of samples belong to Entity Missing, Evidence Missing, or Incomplete. Not unexpected considering the "headline" nature of the samples.

XSum is Abstractive, but also very noisy. The authors (Narayan et al., 2018a) designed the dataset to be highly abstractive. This is reflected in the distribution: there were no Extractive samples in our analysis, suggesting a significantly higher level of difficulty. However, 55% of samples belong to Entity Missing, Evidence Missing, or Incomplete classes. The remaining 45% belongs to Paraphrase and Inference categories. Since we found only two incomplete samples, this class is ignored in all further XSum analysis.

CNN/DM is cleaner, and lives up to the design goals. The authors (Hermann et al., 2015) designed CNN/DM to be *abstractive* in nature, and this is reflected in the distribution: 64% of samples belong to Paraphrase and Inference categories. Of the three, CNN/DM has the lowest fraction of factual and data noise: there are no Incomplete/Irrelavant samples, and only 18% of samples belong to Entity Missing and Evidence Missing.

The degree with which missing facts affects automatic evaluation varies. In some samples, one or two entities are missing (like Row 2 in Table 2), but in others multiple facts are missing. Empirical analysis of model performance for each class of samples is discussed in Section 3.

3 Performance on different classes (Q2 a)

In this section, we list the different models and metrics considered for analysis, and then describe how model performance varies across class labels.

3.1 Models for evaluation

We collect outputs from 7 systems for Gigaword: (1) PEGASUS (Zhang et al., 2019), (2) PROPHET (Qi et al., 2020) (Lewis et al., 2020), (3) UNILM (Dong et al., 2019) , (4) BISET (Song et al., 2020), (5) CONCOPY (Wang et al., 2019) , (6) POINTERGENERATOR (See et al., 2017), (7) POINTERGENERATORCOPYING (See et al., 2017)

For CNN/DM, we use the outputs of 11 top-performing summarization systems collected by Bhandari et al. (2020)[4]: (1) HETERGRAPH (Wang et al., 2020), (2) MATCHSUMM (Lewis et al., 2020), (3) REFRESH (Narayan et al., 2018b) , (4) TWOSTAGERL (Song et al., 2020), (5) NEUSUMM (Wang et al., 2019) , (6) BOTTOMUP (Gehrmann et al., 2018) (7) SEMSIM (Yoon et al., 2020) (8) UNILM (Dong et al., 2019) (9) BARTABSTRACTIVE (Lewis et al., 2020) (10) BANDITSUMM (Dong et al., 2018) (11) BARTEXTRACTIVE (Lewis et al., 2020)

For XSum, we use the outputs of 9 different summarization systems: (1) CONVSEQ2SEQ (Gehring et al., 2017), (2) TCONVS2S (Narayan et al., 2018a) (3) POINTERGENERATOR (See et al., 2017), (4) BART (Lewis et al., 2020), (5) PRESUMMEXTRACTIVE (Liu and Lapata, 2019), (6) PRESUMMABSTRACCTIVE (Liu and Lapata, 2019), (7) PRESUMMTRANSFORMER (Liu and Lapata, 2019), (8) LEAD (Nenkova, 2005), (9) EXTORACLE (Nallapati et al., 2017)

3.2 Metrics for evaluation

Existing summarization systems are usually evaluated using automated metrics or manually using human judgments. We list popular automatic metrics explored in this work. Except for the last two, all outputs from every model is scored on the following metrics.

ROUGE-1/2/L measure overlap of unigrams, bigrams and longest common subsequence. respectively[5] (Lin, 2004).

BERTScore (BS) measures soft overlap between contextual BERT embeddings of tokens between the two texts[6] (Zhang et al., 2020).

MoverScore (MS) applies a distance measure to contextualized BERT and ELMo word embeddings[7] (Zhao et al., 2019).

FactCC is introduced to measure the fact consistency between the generated summaries and source documents (Kryscinski et al., 2020). Due to issues with the setup and training procedure, this metric was only used in the CNN/DM analysis.

Human Pyramid (HP) provides a robust technique for evaluating content selection by exhaustively obtaining a set of Semantic Content Units (SCUs) from a set of references, and then scoring system summaries on the number of SCUs that can be inferred (Nenkova and Passonneau, 2004). We use the scores shared by Bhandari et al. (2020) for the first 100 samples of CNN/DM subset.

3.3 Model Performance

For each dataset, we group the samples by their labels. For all samples in a subset, the model response is scored using a metric. The mean of these sample scores returns a single subset-model-metric score, which is then averaged across all models in the subset, leaving us with a single subset-metric score. This is repeated for all (subset × metric) pairs. The results are captured in Figures 2, 3 and 4 for Gigaword, CNN/DM and XSum respectively. The last column in each group is the average score across all samples.

3.3.1 Impact of Data Quality and Noise

Incomplete and Irrelevant Of the three datasets, only Gigaword contains Incomplete (or Irrelevant) samples. Across all metrics, the

---

[4] https://github.com/neulab/REALSumm

[5] For ROUGE-1,2, and L, we used the Python implementation: https://github.com/sebastianGehrmann/rouge-baselines

[6] Used code at github.com/Tiiiger/bert_score

[7] We used a faster version of the code provided by the author at github.com/AIPHES/emnlp19-moverscore

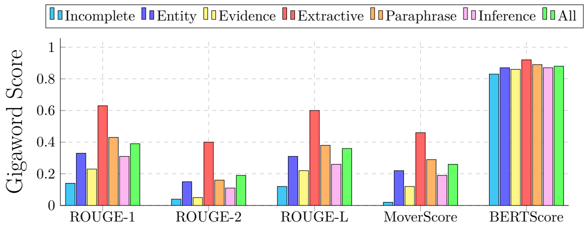

Figure 2: Gigaword class-level performance, averaged across all models.

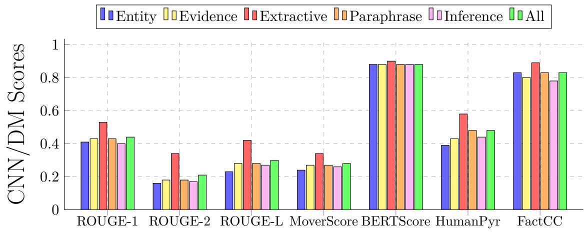

Figure 3: CNN/DM class-level performance, averaged across all models.

performance on this label is lowest, which is to be expected – high overlap will be rare if the source and target are unrelated or incomplete (like Row 1, Table 2). What's alarming is the volume of such samples in Gigaword – if the distribution is the same for the training set, then the model is being trained on extremely noisy data (almost 14%). In addition, such samples needlessly penalise the model performance during evaluation.

Entity scores more than Evidence in Gigaword! The results for these subsets are a bit surprising. In Gigaword, the Entity Missing subset receives relatively higher scores than the Evidence Missing category. We attribute this to a combination of factors. Consider Row 2 in Table 2. Entities are missing, but token overlap is high (more than 50%), which explains the high R1 scores, but low R2 scores. In our observations, the impact of missing facts and entities varies by the length of the target, as well as the number of entities.

Are Evidence Missing and Paraphrase are all the same for CNN/DM and XSum? When compared with Gigaword, samples with data quality issues (i.e. Incomplete/Irrelevant, Entity Missing and Evidence Missing samples) in CNN/DM and XSum get relatively higher scores. The reasons are similar to the Gigaword phenomenon discussed before. The average summary length of CNN/DM (54 tokens) is about 7 times that of Gigaword (8 tokens). As a result, with respect to the complete reference, one or two missing facts amounts to a much smaller fraction of the reference in CNN/DM. The high overlap with the remainder leads to higher scores.

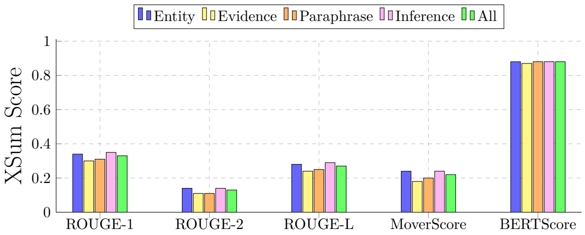

Figure 4: XSum class-level performance, averaged across all models.

Factual Correctness in CNN/DM Automatic metrics only consider the token overlap (or "semantic distance") between the target and the model output. While such metrics exhibit high correlation with human-judgement, a low score does not necessarily imply an incorrect generation, as demonstrated by Freitag et al. (2020) for machine translation. Hence we check for factual correctness of model outputs using FactCC. The competitive scores on the first three categories for FactCC in Fig .3 suggests the outputs generated by the model are factually faithful, which points to issues with the metric reliability. We discuss this in Section 4.

3.3.2 Impact of Summarization Complexity

For the last three categories (Extractive, Paraphrase and Inference) Gigaword and CNN/DM exhibit a common trend: the highest performance, across all metrics is on the Extractive subset, followed by Paraphrase samples which are more difficult to reproduce. The lowest performance is on the Inference samples. However, concluding models perform poorly would be incorrect. The last three samples in Table 2 suggest that model outputs are coherent, logical and factually faithful. FactCC scores in Figure 3 also suggest the outputs are factually consistent.

Some metrics are biased towards simpler samples? For the Extractive, Paraphrase and Inference samples, the samples we manually observed (some of which are captured in Table 2) and the FactCC scores indicates a gap in the token-based metrics. However, we cannot fault the metrics entirely. If we had diverse target references for the same sources, some outputs would have found better matches, and thus, higher scores! In fact, we see that BERTScore (a more "semantically" oriented metric) is extremely competitive across all categories in all three datasets (Figures 2, 3, 4), suggesting the generations are similar to the references. These results lead us to believe that token-based summarization metrics might also suffer from a "summarization-ese" effect: **the metrics could be biased towards simpler, more "extractive" references**. Recently, Freitag et al. (2020) also arrived at the same conclusion for machine translation and BLEU (Papineni et al., 2002).

In the next section, we continue to explore the reliability of these metrics.

4 Does the reliability of metrics change with data properties? (Q2 b)

For each document $d_i, i \in \{1 \dots n\}$ in a dataset $\mathcal{D}$, we have J system outputs, where the outputs can come from different systems. Let $s_{ij}, j \in \{1 \dots J\}$ be the j^{th} summary of the i^{th} document, m_i be a specific metric (including human judgment).

$$K^{sum}_{m_1 m_2} = \frac{1}{n} \sum_{i=1}^{n} \Big(K\big([m_1(s_{i1}) \dots m_1(s_{iJ})], [m_2(s_{i1}) \dots m_2(s_{iJ})]\big) \Big). \quad (1)$$

Correlation is calculated for each document, among the different system outputs of that document, and the mean value is reported. Like other meta-evaluation studies, we consider the Pearson correlation and Spearman correlation as measures for K. Due to space constraints we only show the Pearson plots for some critical results. More plots are available in Appendix A.1.

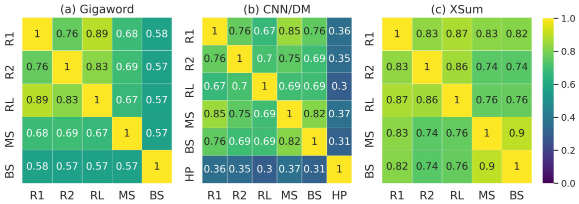

Figure 5: Pearson correlation between different metrics for all three datasets.

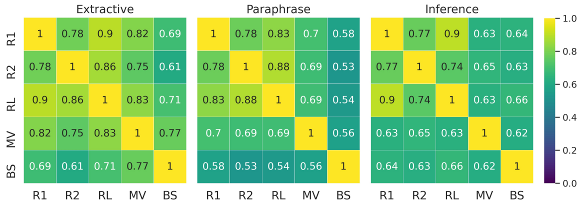

(a) Gigaword

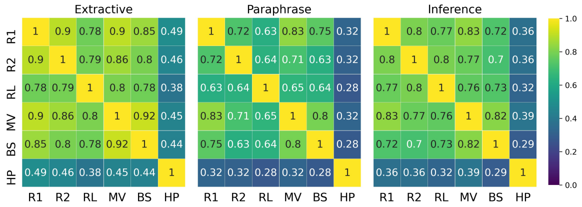

(b) CNN/DM

Figure 6: Pearson correlations for Extractive, Paraphrase and Evidence samples in Gigaword and CNN/DM.

Inter-metric Correlation We present a pairwise correlation analysis of the automatic metrics to understand metric agreement in Figure 5. We conjecture that a strong correlation between two vastly different metrics (say ROUGE and MoverScore) might show that the metric is more reliable. Overall, we can see in Figure 5 that correlations between token-based metrics (ROUGE) and embedding-distance metrics (BERTScore, MoverScore) is lower in Gigaword, compared to CNN/DM and XSum. It is possible that the short length summaries of Gigaword is leading to this; perhaps there isn't enough context for BERTScore. Although, we could not find any results in the original papers to support this claim.

Correlation variation with complexity We observe that the correlation is heavily sample dependent. In Figure 5, averaged across all samples, R1 and MoverScore have a Pearson correlation of about 0.68 in Gigaword. This increases to 0.82 for the Extractive samples in Figure 6-(a), which are the simplest to reproduce. As the complexity increases, the correlation scores decrease (in Paraphrase, and then in Inference). The trends for R2 and MoverScore are similar. This is also observed for CNN/DM: in Figure 6-(b), correlations for R1-MoverScore and R1-BERTScore drop from 0.9, 0.85 for Extractive samples to about 0.83, 0.72 for Paraphrase and Inference samples. **This suggests that the inter-metric correlation is heavily sample dependent**. We cannot comment on XSum, because we did not encounter any Extractive samples in that dataset.

Correlation with Human Judgement For CNN/DM, we also compute the metric correlations with the human pyramid score (HP) in Figure 5 and

Figure 6-(b). We observe the highest agreement with the human-judgement for the Extractive subset, and it is significantly lower in Paraphrase and Inference. **This suggests that automatic metrics are more reliable when evaluating simpler examples, than complex ones**.

5 Discussion

Limitations of the typology. Forcing samples to have a single label did limit our analysis. In retrospect, the typology could have allowed for two labels: one for quality, one for complexity. In XSum for instance most samples which were labelled Entity Missing could also be labelled Paraphrase and Inference. We also realise that the impact of positional-bias could be important. This has been explored by Zhong et al. (2019a,b), and we plan to include similar metrics in our future work.

Collecting better datasets. Our results suggest that current metrics are not equally reliable across all categories of samples. If the quality of the references cannot be controlled, then having a diverse set of references for the source is also advised. This will allow for multi-reference evaluation and could offset the "summarization-ese" issues.

Limits of the Pyramid Scores. At the moment, the Pyramid Scores (and judgement criteria in general) only compare the output to the gold-reference, assuming the latter is *true*. As we see from our analysis, ignoring the source is not the right approach, for references from the web could have quality issues. A modified judgement procedure, that also accounts for the faithfulness of the gold-reference (perhaps by using automatic factuality metrics FactCC) might be better.

Architecture specific performance. In this study, we were interested in measuring the broader, *averaged* trends that summarization models exhibit. However, it would be interesting to see how specific architectural decisions impact individual model performance across different classes. We plan to explore this in the future.

"But what's the best metric for my data?" Specifically for metrics, our objective was to empirically demonstrate that (a) datasets have different modalities, and (b) metrics are not equally reliable across these modalities. In this process, we also observed some results suggesting possible biases in certain token-based metrics, and a need for diverse reference sets. We'll continue to explore this question.

6 Related Work

For the task of text-summarization, the data analysis heuristics presented in Zhong et al. (2019a,b); Bommasani and Cardie (2020); Grusky et al. (2018) are most relevant to our work. Their analysis is focused on surface level heuristics which ignores all noise present in the data. This has been discussed in Sections 2.2.1, 5. Researchers have also explored other dataset biases (Jung et al., 2019; Zhong et al., 2019b; Chen et al., 2020). As discussed in Section 5, we plan to include this in our future work.

For metric reliability and meta-analysis, we build on correlation analysis presented in earlier works (Peyrard, 2019; Bhandari et al., 2020; Fabbri et al., 2020). The key difference and novelty is the introduction of our typology and measuring the impact of sample complexity on model performance and metric reliability. To the best of our knowledge, metrics and models have not been evaluated on such a typology. As results in Section 3 and 4 show, sample complexity is indeed very critical for metric reliability.

7 Conclusion

In this study, we manually analysed 600 samples from three popular datasets, using a typology that captures data quality issues and varying degrees of sample-complexity. Our analysis of 27 summarization models reveals that the metric performance is heavily dependent on samples. On closer inspection, we found that the agreement of popular metrics also changes with the complexity, thus the scores might not reflect true model performance. This analysis also led to some suggestions for creating better summarization datasets and highlights some limitations of the current human-judgement procedures.

Acknowledgements

We thank Professor Graham Neubig, Yiran Chen and anonymous reviewers for valuable feedback and helpful suggestions. Thanks Kaiqiang Song for providing system outputs. This work was supported in part by a grant under the Northrop Grumman SOTERIA project and the Air Force Research Laboratory under agreement number FA8750-19-2-0200.

References

Manik Bhandari, Pranav Narayan Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. 2020. Re-evaluating evaluation in text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9347–9359, Online. Association for Computational Linguistics.

Rishi Bommasani and Claire Cardie. 2020. Intrinsic evaluation of summarization datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8075–8096, Online. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2017. Faithful to the original: Fact aware neural abstractive summarization.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1662–1675.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Yiran Chen, Pengfei Liu, Ming Zhong, Zi-Yi Dou, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. CDEvalSumm: An empirical study of cross-dataset evaluation for neural summarization systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3679–3691, Online. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium. Association for Computational Linguistics.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2020. Gsum: A general framework for guided neural abstractive summarization. *arXiv preprint arXiv:2010.08014*.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. Summeval: Re-evaluating summarization evaluation. *arXiv preprint arXiv:2007.12626*.

Markus Freitag, David Grangier, and Isaac Caswell. 2020. Bleu might be guilty but references are not innocent.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348, Beijing, China. Coling 2010 Organizing Committee.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 142–151.

Taehee Jung, Dongyeop Kang, Lucas Mentch, and Eduard Hovy. 2019. Earlier isn't always better: Subaspect analysis on corpus and system biases in summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3324–3335, Hong Kong, China. Association for Computational Linguistics.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive Summarization of Reddit Posts with Multi-level Memory Networks. In *NAACL-HLT*.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Yixin Liu, Dou Ziyi, and Pengfei Liu. 2021. Refsum: Refactoring neural summarization. In *The 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, page 1436–1441. AAAI Press.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maxime Peyrard. 2019. Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Eva Sharma, Chen Li, and Lu Wang. 2019. Bigpatent: A large-scale dataset for abstractive and coherent summarization. *arXiv preprint arXiv:1906.03741*.

Kaiqiang Song, Bingqing Wang, Zhe Feng, Ren Liu, and Fei Liu. 2020. Controlling the amount of verbatim copying in abstractive summarization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8902–8909. AAAI Press.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. *arXiv preprint arXiv:2004.12393*.

Kai Wang, Xiaojun Quan, and Rui Wang. 2019. BiSET: Bi-directional selective encoding with template for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2153–2162, Florence, Italy. Association for Computational Linguistics.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Mark Yatskar. 2019. A qualitative comparison of CoQA, SQuAD 2.0 and QuAC. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2318–2323, Minneapolis, Minnesota. Association for Computational Linguistics.

Wonjin Yoon, Yoon Sun Yeo, Minbyul Jeong, Bong-Jun Yi, and Jaewoo Kang. 2020. Learning by semantic similarity makes abstractive summarization better. *arXiv preprint arXiv:2002.07767*.

Weizhe Yuan, Pengfei Liu, and Graham Neubig. 2021. Can we automate scientific reviewing? *arXiv preprint arXiv:2102.00176*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019a. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1049–1058, Florence, Italy. Association for Computational Linguistics.

Ming Zhong, Danqing Wang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2019b. A closer look at data bias in neural extractive summarization models. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 80–89, Hong Kong, China. Association for Computational Linguistics.

Appendix A Figures and Annotation Details

A.1 Correlation plots

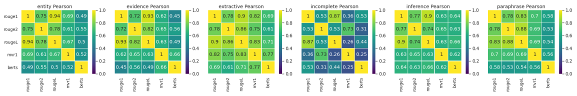

(a) Pearson

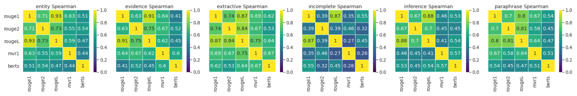

(b) Spearman

Figure 7: Gigaword correlations.

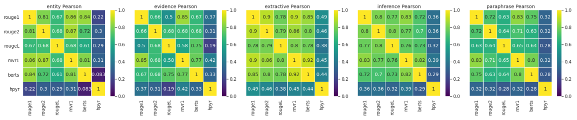

(a) Pearson

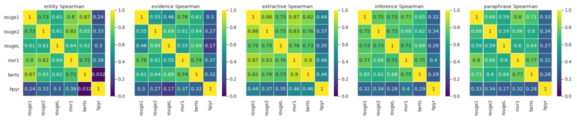

(b) Spearman

Figure 8: CNN/DM correlations.

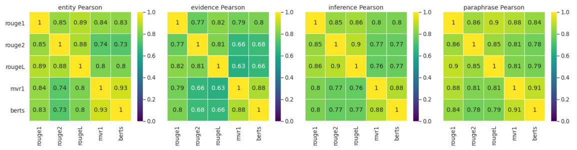

(a) Pearson

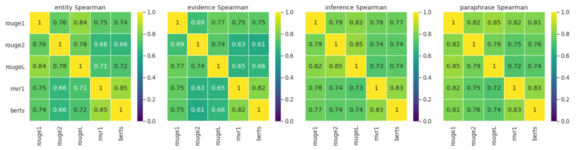

(b) Spearman

Figure 9: XSum correlations.

A.2 Annotation Details

Each sample is annotated by 2-3 annotators independently. Given the limited number of samples, and the laborious nature of the exercise, we chose not to select final labels based on majority vote. For all disagreements, annotators discussed their reasoning and came to an consensus for final label. For 70% of Gigaword samples, 68% of CNN-DM samples, and 73% of XSum samples, the initial annotations were in agreement.

Appendix B Gigaword

B.1 Gigaword: Paraphrase and Inference samples

Label	**Source**	**SoTA Output**	**Gold Reference**
Paraphrase	A woman street cleaner and her three young daughters were killed Saturday when a bomb in a metal container exploded in Bangladesh , police said .	Mother , three daughters die in in Bangladesh blast .	Mother , three daughters killed in Bangladesh blast .
Paraphrase	The UN chief of Eastern Slavonia , the last Serb-held part of Croatia , confirmed Tuesday that key elections would be held here on April 13 as part of local ballots throughout Croatia .	UN chief confirms key elections in Eastern Slavonia .	UN confirms elections to be on April 13 in Eastern Slavonia .
Paraphrase	Business at Taiwan 's theme parks and resorts grew significantly in the first quarter of this year compared to Q1 last year , the Tourism Bureau said Thursday , attributing the growth to the government 's shopping voucher program and other promotion efforts .	Business at Taiwan 's theme parks and resorts grows .	Shopping vouchers help boost theme parks business : tourism bureau .
Inference	Col. Robert E. Lee skirted the unleaded gasoline pit , negotiated a thicket of telephone cords stretched as tight as trip wires and took the center of the New York Mercantile Exchange 's main trading floor just before 3 p.m. last Monday .	New York Mercantile Exchange 's trading floor .	MILITARY STRATEGISTS PRACTICE IN REAL BATTLE ON WALL STREET .
Inference	Finland scored three goals in a 40-second span of the first period Tuesday night for a 7-3 victory over the Czech Republic in their World Cup of Hockey opener .	Finland 7 , Czech Republic 3 .	Finland Routs Czech Republic at World Cup .
Inference	Q. I 've heard that cow manure can be used for energy production , but not human waste .	Cow manure can be used for energy production .	ON NOT WASTING WASTE .

Table 3: Source, outputs and targets, from Gigaword.

Appendix C CNN/DM

C.1 CNN/DM: Paraphrase and Inference samples

Label	Source	SoTA Output	Gold Reference
Paraphrase	Her neighbour's leylandii hedge stands 40ft tall and, says Audrey Alexander, has left parts of her garden in deep shade. What's more, it now seems likely to remain that way. (truncated). A row between neighbours over a 40ft high leylandii hedge (pictured) has finally come to and end after 35 years . The battle between the neighbouring properties started in 1980 when the owner planted a vegetable patch which withered and died in the shade of her neighbour's massive hedge . Then, 23 years ago, single mother Mrs Alexander bought the house and asked her neighbour Jeanette Robinson to trim the hedge. She claims Mrs Robinson refused and declared: 'I would rather move than touch these trees.' (truncated) . Audrey Alexander (pictured) also claims other neighbours have had to move their children from their bedrooms at night for fear of the falling branches . But her council has ruled that Mrs Robinson can keep the hedge, although it has to be cut to 20ft. Mrs Alexander said the ruling made 'no difference'. (truncated)	Audrey Alexander 's vegetable patch withered and died in the shade of hedge . She asked neighbour Jeanette Robinson to trim it but she refused . Mrs Alexander claims hedge knocked £ 20,000 off the value of her house . Stirling Council has ruled that Mrs Robinson can keep the hedge . But it has to be cut to 20 ft , a height which she claims will still block most of her sunlight .	Audrey Alexander wanted her neighbours to chop down their huge hedge . She claims the 40 ft leylandii was blocking sunlight from reaching her home . Feud started in 1980 when it blocked light from reaching a vegetable patch . Council finally rules that the hedge can stay - but must be cut back to 20 ft .
Paraphrase	The number of women in Britain becoming nuns is at a 25-year high. Figures from the Catholic Church show the number of women taking Holy Vows has trebled from 15 in 2009 to 45 last year. From a low of only seven in 2004, the figure has been rising for the past decade. Theodora Hawksley, 29, was until recently a post-doctoral researcher in theology at the University of Edinburgh. (truncate). Far from being trapped in traditional habits, Miss Hawksley said her order tends to dress down in T-shirts and jeans. Father Christopher Jamison, director of the National Office for Vocation of England and Wales, said: 'There is a gap in the market for meaning in our culture. One of the ways women may find that meaning is through religious life.' Sister Cathy Jones, religious life vocations promoter at the office, said: (truncated) .	Figures from Catholic Church show number of women taking Holy Vows has trebled from 15 in 2009 to 45 last year . From a low of seven in 2004 , the figure has been rising for the past decade . Theodora Hawksley , 29 , was until recently a post - doctoral researcher in theology at the University of Edinburgh . But at the beginning of the year she decided to become a nun .	Figures from the Catholic Church show more and more becoming nuns . The number of women taking Holy Vows stood at just seven back in 2004 . But that figure had risen to 15 in 2009 and increased further to 45 last year . One father said a ' gap in the market for meaning ' led people toward religion .
Inference	Following all his inspired charity work, Didier Drogba has been awarded with a Barclays Spirit of the Game trophy. The Chelsea forward set up the 'Didier Drogba Foundation in Africa,' as he hopes to inspire the next generation of footballers in Africa to fall in love with the game. (truncated) He said 'I come from a poor family where I played football in the streets with my friends with no shoes, there was no grass but we still enjoyed it. The 'Didier Drogba Foundation,' contribute financial and material support in education and health including school bags for the school children, as well as a medical clinic in his hometown of Abidjan, Ivory Coast, which will be opening its doors later this year. Chelsea's stars such as Eden Hazard, Petr Cech and Branislav Ivanovic were out in force earlier this month as they raises £400,000 for the foundation at a charity ball. The money raised will be used to complete the medical clinic in Abidjan and help finance mobile clinics that will travel outside of the capital to those who are either to sick or poor to make the journey to the medical centre.	Didier Drogba has been awarded with a Barclays Spirit of the Game trophy . The Chelsea forward set up the ' DidierDrogba Foundation in Africa ' He hopes to inspire the next generation of footballers in Africa to fall in love with the game . The 37-year - old scored the equaliser against Leicester on Wednesday .	Didier Drogba given the Barclays Spirit of the Game award . The 37-year - old 's foundation has done impressive work in Africa . Some of Chelsea 's stars attended a charity ball which raised £ 400,000 . CLICK HERE for all the latest Chelsea news .
Inference	(truncated) Resorts on its Black Sea coast offer the best value in terms of a meal out, buying a cup of coffee and essentials such as sun cream and a cold drink, according to a study. Scroll down for video . Affordable: Bulgaria has been named Europe's cheapest destination, with Black Sea resorts like Sunny Beach (pictured) offering the best value in terms of a meal out and other holiday activities . Hotspot: Bulgaria's most popular resort of Sunny Beach is a carbon copy of those of Spain and Greece . It is one of 13 European hotspots out of 14 where your cash will go far further this summer, largely thanks to rock-bottom exchange rates and higher inflation in some countries. Research into an imaginary shopping basket of ten typical holiday purchases showed a total price of £37.39 for Bulgaria, which is down by 13.6 per cent from last summer. There was a bigger fall of 22 per cent for the Algarve in Portugal, taking the total cost to £44.02, helping it beat Spain's Costa del Sol to become the second cheapest destination. Only in Turkey, where inflation is 7.6 per cent – compared to virtually zero in Britain and the eurozone – will Britons find the cost of a day out much more expensive. The figures, compiled for the annual Post Office Holiday Costs Barometer, show the spending basket in Turkey is up by 21.4 per cent on last year, at £65.70. Bulgaria's most popular resort of (truncated) .	Former Soviet state has gained the most from the strong pound . Resorts on its Black Sea coast offer the best value in terms of a meal out , buying a cup of coffee and essentials such as sun cream and a cold drink . It is one of 13 European hotspots out of 14 where your cash will go far further this summer .	Bulgaria's Black Sea resorts cheaper than hotspots in Italy, Spain and Turkey . Researchers found cheapest destination using 'imaginary shopping basket' Cheap prices are driven by low exchange rates and country's high inflation . Its most popular resort of Sunny Beach copies those of Spain and Greece .

Table 4: Source, outputs and targets, from CNN/DM.

Appendix D XSum

D.1 XSum: Paraphrase and Inference samples

Label	Source	SoTA Output	Gold Reference
Paraphrase	More than 700,000 employees face unpaid leave due to the shutdown which was triggered after the two houses of Congress did not agree on a new budget. Hyundai said affected employees who currently own its vehicles will be given a payment relief ”for as long as they are out of work”. Employees looking to buy a new car will be given a 90-day payment deferral. ”We recognize the impact on family budgets that the furlough will drive,” John Krafcik, chief executive of Hyundai Motor America, said in a statement. Hyundai had offered a similar scheme, the Hyundai Assurance programme, during the peak of the global financial crisis four years ago to help consumers who had lost their jobs. Many analysts have said that the move had helped the South Korean firm win customer loyalty and boosted its sales in recent years. The company said that its latest offer to help the federal employees was an addition to that programme and aimed at ”helping workers at a time when they most need it”. ”Like we did almost four years ago when we launched Hyundai Assurance, this is our way of saying ’We’ve got your back’ during this uncertain time,” Mr Krafcik said. Under the latest offer, Hyundai will extend all auto loan and lease payments during the shutdown for current Hyundai owners who are put on unpaid leave. The programme is available to all customers who have financed their purchase or lease through Hyundai Finance America.	US carmaker Hyundai Motor has offered financial help to federal employees who have been affected by the government shutdown .	Hyundai Motor will defer payments due from US federal employees affected by the partial government shutdown .
Paraphrase	Gary Price was suspended from all council duties for five months in November after Powys council’s Standards Committee ruled he had breached the code of conduct. His appeal has been dismissed by the Adjudication Panel for Wales following a two-day hearing in Llandrindod Wells. Mr Price has been contacted for comment. He was found to have sent information which the council said ”incorrectly and unfairly” portrayed what happened at a grievance appeal hearing, in which he was a panel member. The Adjudication Panel for Wales unanimously agreed to refer the matter back to the Standards Committee with a recommendation that Mr Price be suspended for three months. Council leader Barry Thomas said the decision ”sends out a clear message that those who enter public office have to operate within the members’ code of conduct and maintain the highest possible standards”.	A Powys council chief executive has lost his appeal against a decision to suspend him .	A decision to suspend a Powys county councillor has been upheld .
Inference	Derby City Council wanted to shut Moorways Pool from April in a bid to save about Â£350,000 a year. The Labour-led authority, which needs to save Â£79m over the next three years, said it had found the savings by making cuts in other areas. Campaigners who gathered more than 4,000 signatures on a petition said they were delighted at the news. Ranjit Banwait, leader of the authority, said the council had committed to keep it open for a year. He said the council had identified savings ”in back-office areas” and a restructuring of management jobs, which had been ”untouched” since 2010. However, he stressed if the authority failed to get a ”fair deal” from central government in the future, the pool would still have to close. Campaigners had accepted the pool, which is 33m in length, was in need of repair. There are plans for a new 50m pool to be built by 2018 to replace it. However, closing it would have left only one other public pool in the city - the Queen’s Leisure Centre, they said. Doug Whitlam, of the Derbyshire Amateur Swimming Association, said: ”One of the main things for me would have been the loss of teaching. ”Twelve hundred young people use this facility every week and that would be lost forever.”	A council has backed down over plans to close a public swimming pool in a bid to save money .	A Derby swimming pool threatened with closure is to remain open for another year , council bosses have confirmed .
Inference	It is likely to include a scrappage scheme for older diesel cars in areas with high levels of dirty air. Speed bumps could be removed in some cities to cut pollution from cars slowing down and speeding up. Environmental lawyers ClientEarth said they would ”thoroughly analyse” the proposals. According to the Royal College of Physicians, air pollution across the UK is linked to around 40,000 premature deaths every year. The UK has struggled to keep within EU limits on some pollutants, particularly nitrogen dioxide (NO2), which is produced by diesel engines and is linked to a range of respiratory diseases including asthma. Some 37 of the 43 regions of the UK are in breach of NO2 limits. Under earlier government plans, some parts of the UK would not have met EU NO2 standards until 2030. The original deadline to achieve these limits was 2010. Exasperated by what they believed was government foot-dragging on the question of cleaner air, ClientEarth mounted a legal challenge to force faster action. In April 2015, the UK Supreme Court ruled the government had to take immediate steps on the issue. Unhappy with the timescales in the plan that was then produced, ClientEarth went to the High Court last November for a judicial review. Once again the court supported the lawyers, telling the government that its scheme was ”woefully inadequate” and giving ministers until 24 April this year to produce a new draft. With a general election in the offing, the government last week asked the judge for permission to delay the draft plan. But Mr Justice Garnham disagreed and ordered publication by 9 May. ”These steps are necessary in order to safeguard public health,” he said. Earlier this week, the government said it would not appeal against the ruling and would publish. In their previous plans, ministers wanted to create ”clean air zones” in five cities outside London with high levels of NO2. Only the most polluting vehicles would have to pay a charge to enter the zone under that scheme. The new draft plan is expected to create many more such zones. Councils will be given the power to impose fines or restrictions on all polluting vehicles in these areas. In the worst cities, so called ”toxin taxes” could range up to Â£20 a day but the government is said to be keen not to punish drivers who bought diesels as a result of incentives brought in by a previous Labour administration. This is something that the lawyers at ClientEarth support. ”Successive governments have encouraged people to buy diesel. We don’t want to see diesel drivers vilified, and we think the plans should also include properly funded incentives to help people move to cleaner forms of transport,” said ClientEarth CEO James Thornton. ”We will thoroughly analyse the government’s draft plans when they are produced. If we do not think they are in line with the court order, to deal with illegal levels of pollution as soon as possible, then we will consider our next steps.” According to newspaper reports, the government has agreed to back a ”targeted” scrappage scheme for older diesel cars, but limited to vehicles in areas of high pollution. There may also be funding for a retrofitting scheme to help existing diesel car and van owners cut their emissions of NO2. The government is also said to be pushing for councils to use alternatives to charging, including the removal of speed bumps in some places and the better sequencing of traffic lights in others. Both of these measures could limit cars having to slow down and speed up repeatedly, actions that can almost double the amount of NO2 produced. However, the idea that speed bumps which slow down traffic would be sacrificed to help clean up the air we breathe is not a welcome concept according to road safety charity Brake. ”We ought not to be made to choose between having cleaner air and safer roads,” a spokesman said. ”The evidence shows that air pollution is contributing to the early deaths of thousands of people. It’s now clear that there’s more than one way a car can kill you.” The new proposals will be out for consultation for six weeks before the government produces a final plan at the end of July. Follow Matt on Twitter and on Facebook.	The government is expected to publish a new draft plan to tackle air pollution in the UK later this week .	The UK government is set to publish a draft air pollution plan after a protracted legal battle with environmental campaigners .

Table 5: Source, outputs and targets, from XSum.

Multilingual Translation from Denoising Pre-Training

Yuqing Tang[1,2]*, **Chau Tran**[1], **Xian Li**[1], **Peng-Jen Chen**[1], **Naman Goyal**[1],
Vishrav Chaudhary[1], **Jiatao Gu**[1], **and Angela Fan**[1]
Facebook AI[1]
{chau,xianl,pipibjc,naman,vishrav,jgu,angelafan}@fb.com
Amazon Alexa AI[2]
yuqint@amazon.com

Abstract

Recent work demonstrates the potential of training one model for multilingual machine translation. In parallel, denoising pretraining using unlabeled monolingual data as a starting point for finetuning bitext machine translation systems has demonstrated strong performance gains. However, little has been explored on the potential to combine denoising pretraining with multilingual machine translation in a single model. In this work, we fill this gap by studying how *multilingual* translation models can be created through *multilingual finetuning*. Fintuning multilingual model from a denoising pretrained model incorporates the benefits of large quantities of unlabeled monolingual data, which is particularly important for low resource languages where bitext is rare. Further, we create the ML50 benchmark to facilitate reproducible research by standardizing training and evaluation data. On ML50, we show that multilingual finetuning significantly improves over multilingual models trained from scratch and bilingual finetuning for translation into English. We also find that multilingual finetuning can significantly improve over multilingual models trained from scratch for zero-shot translation on non-English directions. Finally, we discuss that the pretraining and finetuning paradigm alone is not enough to address the challenges of multilingual models for to-Many directions performance.

1 Introduction

A slow but increasingly growing focus on languages beyond English has contributed a large wave of models, data, and tasks for non-English languages. Much work has been dedicated to the area of translation, with increasing exploration in massively multilingual models. Despite advances in multilingual natural language processing, resources are highly unbalanced across different languages. This is an obstacle for tasks requiring large quantities of labeled data, such as translation systems, which traditionally leverage hundreds of thousands of professional human translations.

A promising avenue of research is to remove the requirement for large quantities of labeled data by leveraging unlabeled monolingual data, often in the form of large-scale pretraining (Lample and Conneau, 2019; Conneau et al., 2020; Liu et al., 2020; Tran et al., 2020; Liu et al., 2019; Brown et al., 2020). Monolingual data is far more prevalent for low resource languages, particularly in resources such as Wikipedia or Commoncrawl, a version of the web. Recent work has explored monolingual denoising pretraining (Liu et al., 2020) for bilingual models finetuning for individual translation directions (for simplicity we will refer to *monolingual denoising pretraining* as *pretraining* from now on). However, bilingual finetuning alone does not leverage the benefit of the potential of transfer learning across languages. On the other hand, recent work (Arivazhagan et al., 2019b; Fan et al., 2020) has also demonstrated much potential for performance improvement from multilingual translation models in a single model (for simplicity from now on we will use *multilingual translation model* or *multilingual model* to refer to a single model which performs machine translation for multiple languages), but these approaches do not leverage unlabeled monolingual data directly. Little has been explored regarding the combination of the two approaches. Thus, this work studies the effectiveness of combining both large scale pretraining and all-in-one multilingual translation towards universal automatic translation across human languages.

In this work, we finetune pretrained models into multilingual translation models [1]. We analyze the

---

*This work was completed when the first author was at Facebook AI.

[1] We open source our implementation, pretrained and fine-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3450–3466
August 1–6, 2021.

effectiveness of *multilingual finetuning* — finetuning a single model to perform translation for multiple languages — across low, mid, and high resource translation settings to understand the benefits and limits of both pretraining and the transfer learning across languages. First, we demonstrate how to extend pretrained models to support additional languages using only monolingual data via denoising training criteria. Next, we show how to perform effective finetuning to create one-model multilingual translation. Finally, we evaluate the multilingual translation across a variety of settings to understand the strength of starting with pretraining. Ultimately, we demonstrate that finetuning to create one-model multilingual translation provides large BLEU improvements in the Many-to-English setting, but starting with pretraining is not sufficient to achieve strong English-to-Many performance.

2 Related work

2.1 Multilingual Pretraining

We build upon recent progress of pretraining techniques for NLP applications (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Song et al., 2019; Lewis et al., 2020a). In particular, recent works explored pretraining on multilingual unlabeled corpora (Lample and Conneau, 2019; Conneau et al., 2020; Liu et al., 2020; Tran et al., 2020) and significantly improved the performance of finetuning of bilingual translation models between two languages. We extend Liu et al. (2020); Cooper Stickland et al. (2021) by investigating finetuning in a multilingual setting.

2.2 Multilingual Neural Machine Translation

Training a universal translation system between multiple languages (Firat et al., 2016; Johnson et al., 2017) has shown enormous improvement for translating low-resource languages (Gu et al., 2018), even enabling zero-shot translation (Lakew et al., 2018; Gu et al., 2019; Arivazhagan et al., 2019a; Garcia et al., 2020). Previous multilingual translation work began with multitask learning (Dong et al., 2015). Subsequently, work focused on the the model capacity bottleneck, leading to exploration of various parameter sharing strategies (Blackwood et al., 2018; Platanios et al., 2018; Sachan and Neubig, 2018; Lu et al., 2018). Models for all languages (Ha et al., 2016) have also been explored and extended to incorporate language information (Tan et al., 2019). Bitext data pretraining and finetuning aiming at creating multiple machine translation models for different translation directions has also be explored (Dabre et al., 2019; Lin et al., 2020). Arivazhagan et al. (2019b); Fan et al. (2020) indicate that it is essential to train gigantic models with enough capacity to fully leverage massive multilingual corpora. A closely related concurrent work, Siddhant et al. (2020) shows it is possible to train a multilingual system jointly with monolingual datasets based on Song et al. (2019). In contrast, in this work we focus on unlabeled data denoising pretraining instead of bitext data pretraining to utilize almost unlimitedly available unlabeled texts. We aim at creating a single universal translation model across multiple languages via finetuning multilingual translation systems from a pretrained model.

2.3 Multilingual Translation Datasets

Working in a multilingual setting remains challenging, as various different datasets, evaluation settings, and preprocessing such as tokenization are used. Benchmarks for sentence embeddings (Hu et al., 2020), natural language inference (Conneau et al., 2018), and question answering (Lewis et al., 2020b) exist, but there is not yet a setting for machine translation data with different resource levels and language families at sufficiently large scale and variety. Zhang et al. (2020) propose OPUS100 with 100 languages, but the training and evaluation data are not human translated. Arivazhagan et al. (2019b) use proprietary data to train and evaluate. In contrast, we contribute the ML50 benchmark, a dataset of 50 languages with publicly available training and evaluation sets, including high, mid, and extremely low resource directions, and open source this benchmark.

3 Multilingual Translation from Monolingual Denoising Pretraining

Masked language modeling and denoising pretraining have been successful across a wide variety of tasks, including creating bilingual translation models. We describe the pretrained multilingual BART model and present multilingual finetuning, a technique to convert pretrained models into multilingual machine translation systems.

---

tuned models, and the ML50 dataset downloading scripts at `https://github.com/pytorch/fairseq/tree/master/examples/multilingual`.

mBART Multilingual BART (mBART) (Liu et al., 2020) is a sequence-to-sequence generative pretraining scheme. The model incorporates N languages by concatenating data: $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_N\}$ where each $\mathcal{D}_i$ is a collection of monolingual documents in language i. mBART is trained as a denoising autoencoder, training to predict the original text X given $g(X)$ where g is a noising function that corrupts text. We maximize $\mathcal{L}_\theta$:

$$\mathcal{L}_\theta = \sum_{\mathcal{D}_i \in \mathcal{D}} \sum_{x \in \mathcal{D}_i} \log P(x|g(x); \theta) \ , \qquad (1)$$

where x is an instance in language i and the distribution P is defined by the seq-to-seq model. This model is pretrained using two types of noise in g — random span masking and order permutation — as described in (Liu et al., 2020).

3.1 Multilingual Finetuning

To leverage pretraining to create translation systems, previous work (Liu et al., 2020) used mBART as a starting point and then performed bilingual finetuning. Concretely, the seq-to-seq model was finetuned on language i to language j translation. However, *bilingual finetuning* does not leverage the full capacity of multilingual pretraining, as the resulting translation model can only translate between two languages. Recent work on multilingual translation (Aharoni et al., 2019; Arivazhagan et al., 2019b) demonstrates that strong translation models can be created by doing multilingual training. Thus, we propose to perform *multilingual finetuning* (**ML-FT**) to retain the benefits of both multilingual translation models and unlabeled data pretraining. Multilingual translation models allow languages to transfer the learning from each other. Pretraining utilizes large amount of monolingual data to complement the lack of bitext data.

To perform multilingual finetuning, we collect bitexts of different language pairs (i, j) into a large collection $\mathcal{B}_{i,j} = \{(x_i, y_j)\}$ for each direction (i, j). We augment each bitext pair (x_i, y_j) by adding a source language token and a target language token at the beginning of x and y respectively to form a target language token augmented pair (x', y'). We then initialize a transformer based seq-to-seq model by the pretrained mBART, and provide the multilingual bitexts $\mathcal{B} = \bigcup_{i,j} \mathcal{B}_{i,j}$ to finetune the pretrained model.

Multilingual Translation Model Variants We explore 3 configurations to create different versions of multilingual translation models: *Many-to-one* (M→1), *one-to-Many* (1→M), and *Many-to-Many* (M↔M) via a pivot language. Given the presence of English language in large scale bitext data, we follow (Arivazhagan et al., 2019b) using English as the pivot language to create Many-to-Many models: the Many-to-one model encodes N languages and decodes to English, while the one-to-Many model encodes English and decodes into N languages. Finally, the Many-to-Many model encodes and decodes N languages.

Temperature Sampling When training multilingual models with many languages, the training dataset sizes are imbalanced as different languages have different quantities of bitext. Thus, we train with temperature upsampling, which upsamples lower resource pairs so that the high resource languages do not dominate the training data. We follow Arivazhagan et al. (2019b) and use the following temperature based sampling function with temperature T to sample data for each direction:

$$p_{i,j} \propto \left(\frac{|\mathcal{B}_{i,j}|}{\sum_{i,j} |\mathcal{B}_{i,j}|} \right)^{1/T}$$

4 Experimental Setting

We examine the impact of multilingual finetuning over pretrained models. First, we create the ML50 benchmark to include 50 different languages of various resource levels and language families that we can obtain from publicly available, high quality data sources. The ML50 benchmark standardizes training data, evaluation data, and evaluation procedure across different languages. Second, we detail how we obtain mBART50 pretrained models by extending mBART25. Third, we describe three strong baselines: bilingual translation models from scratch, bilingual finetuning from mBART50 pretrained models, and multilingual translation models from scratch. Finally, we describe our evaluation and generation procedure. In the next section, (Section 5), we will detail the results of the experiments.

4.1 ML50 Benchmark

To investigate the usefulness of pretraining and multilingual finetuning compared to existing alternatives, we create the ML50 Benchmark. ML50 contains training and evaluation data across 50 different languages, from extremely low resource languages like Xhosa and Gujarati to high resource languages like French and German. The full list of languages is shown in Table 1. We group the languages into five categories based on the amount of available training data: more than 10M pairs (8 languages), 1M to 10M pairs (5 languages), 100k to 1M pairs (17 languages), 10K to 100K pairs (13 languages), and finally, less than 10K pairs of training data (5 languages). While considering the resource levels, we also choose the ML50 dataset to include languages in multiple language families, from Germanic and Romance languages to Indic and African ones. Many additional languages we contribute are lower resource, compared to the languages in the original mBART25.

Training Data We gather bitext data between English and 49 other languages to form ML50, to enable the training of machine translation models. We select these 49 languages based on the amount of bitext and monolingual data to cover languages with different amount of resources and under different language families. All of the data is publicly available, such as WMT (Bojar et al., 2013, 2014, 2016, 2017, 2018; Barrault et al., 2019, 2020), IWSLT (Luong and Manning, 2015; Cettolo et al., 2017), TED58 (Qi et al., 2018), OPUS (Tiedemann, 2012), WAT (Nakazawa et al., 2019), LauraMartinus (Abbott and Martinus, 2019), ITB (Kunchukuttan et al., 2018), and FLORES (Guzmán et al., 2019). For multilingual training, each language pair can include data from multiple sources. We simply concatenate them together and remove duplicated source-target sentence pairs for each language pair. We use `fasttext` (Joulin et al., 2017) to perform language identification on both source and target sentences, and we remove sentences pairs if either source or target sentence is not predicted as expected language. We further filter out training data that match to any source or target side sentences in evaluation datasets. Compared to other datasets such as OPUS100 (Zhang et al., 2020), the ML50 benchmark contains around 4 times more training data. The full list of languages, data sources, and amount of resulting data can be found in Table 6.

Evaluation Data To ensure high quality evaluation of languages covered in ML50, we include publicly available, widely used evaluation sets. We source these evaluation datasets from translation workshops such as WMT, IWSLT, WAT, and other published research works. We follow the evaluation protocol, including tokenization, used for each of these evaluation sets, to ensure our results are comparable with existing work. We release these scripts to make it easier for others [2]. Compared to other datasets such as OPUS100, we choose to use high quality existing evaluation datasets rather than use part of the training data as evaluation. This is because training data, particularly for low resource languages, is often very noisy and unreliable.

4.2 Creating mBART50

While multilingual pretrained models have shown strong performance in a variety of tasks (Liu et al., 2020; Conneau et al., 2020), they remain limited as they are trained on a fixed number of languages. For example, mBART was trained on 25 languages, all fairly high resource. Pretraining fully from scratch is computationally intensive — mBART trained for 2.5 weeks on 256 Nvidia V100 GPUs (Liu et al., 2020). However, there are hundreds of different languages in the world, so restarting pretraining from scratch to add any of them to mBART would be difficult. Instead, we take the existing mBART model, trained on 25 languages, and extend it to more than 50 languages.

We take the public available `mBART25` checkpoint (Liu et al., 2020) in the `fairseq` library (Ott et al., 2019) to continue the pretraining process. We extend `mBART25` embedding layers with randomly initialized vectors for an extra set of 25 language tokens. To be consistent with mBART, we reuse its 250K sentencepiece (Kudo and Richardson, 2018) model which was trained using monolingual data for 100 languages from XLMR (Conneau et al., 2020), and thus already supports languages beyond the original mBART25 was trained on [3]. To create this extended mBART model, we combine the monolingual data of original 25 languages and the new 25 languages from XLMR (Conneau et al., 2020). For pretraining, we train mBART50 for an additional 500K updates with batch size of maximum 9216 tokens per GPU using 64 V100 GPUs. We also release the pretrained mBART50 model, which will be useful for a variety of text generation tasks beyond translation.

---

[2] `https://github.com/pytorch/fairseq/tree/master/examples/multilingual`.

[3] For languages that are not supported in the original $250K$ sentencepiece vocabulary, we can extend the vocabulary to include additional sub-word units for these languages and add the corresponding embedding vectors to the pretrained models to continue the pretraining.

Data size	Languages
10M+	German, Czech, French, Japanese, Spanish, Russian, Polish, Chinese
1M - 10M	Finnish, Latvian, Lithuanian, Hindi, Estonian
100k to 1M	Tamil, Romanian, Pashto, Sinhala, Malayalam, Dutch, Nepali, Italian, Arabic, Korean, Hebrew, Turkish, Khmer, Farsi, Vietnamese, Croatian, Ukrainian
10K to 100K	Thai, Indonesian, Swedish, Portuguese, Xhosa, Afrikaans, Kazakh, Urdu, Macedonian, Telugu, Slovenian, Burmese, Georgia
10K-	Marathi, Gujarati, Mongolian, Azerbaijani, Bengali

Table 1: Languages in ML50 Benchmark. We display the languages included in the ML50 Benchmark and the quantity of training data in bitext pairs. Full breakdown is provided in Table 6.

Multilingual Finetuning from mBART50 We finetune the mBART50 model into Many-to-one (M→1), one-to-Many (1→M), and Many-to-Many (M↔M) models with the ML50 training dataset using English as pivot as described in Section 3.1. We finetune the models for 300K updates and sweep through different batch sizes (4096 and 8000 maximum tokens per GPU), learning rates $(1e{-}4, 2e{-}4, 5e{-}4)$, and upsampling temperature $(1.5, 3, 5)$ for best performing multilingual models on validation, using 32 GPUs for each training instance.

4.3 Baselines

We compare our proposed multilingual finetuning to three strong baselines: bilingual training from scratch, bilingual finetuning, and multilingual models trained from scratch.

Bilingual Trained from Scratch (BL-SC) We train bilingual translation models with standard Transformer (Vaswani et al., 2017) models for translation into and from English to 49 languages. For directions with more than 1 million bitext training data (de, cs, fr, ja, es, ru, pl, zh, fi, lv, lt, and hi), we train Transformer Big models as there is more data to benefit from additional model capacity. For directions with more than 10 million bitext training data (de, cs, fr, ja, es, ru, pl, and zh), we also train Transformer Large models as there is even more data to benefit from additional model capacity. The best performing bilingual model is selected as the Bilingual Train from Scratch baseline. Please refer to Table 5 for details of these architectures.

Bilingual Finetuning (BL-FT) Bilingual finetuning adapts the mBART model into bilingual machine translation models by training for longer on translation bitext. For each language direction, we follow Liu et al. (2020) and finetune for 40K updates to obtain the Bilingual Finetuning baseline.

Multilingual Trained from Scratch (ML-SC) We train 3 different multlilingual models from scratch: Many-to-one (M→1), one-to-Many (1→M), and Many-to-Many (M↔M) with English as pivot. We train for 500K updates and sweep through different batch sizes (4096 and 8000 maximum tokens per GPU), learning rates $(1e{-}4, 2e{-}4, 5e{-}4)$, and upsampling temperature $(1.5, 3, 5)$ for best performing multilingual model on validation, using 32 GPUs for each training instance.

4.4 Evaluation and Generation

We evaluate performance with tokenized BLEU, following the tokenization in mBART (Liu et al., 2020). To generate, we decode using beam search with beam size $N = 5$ with length penalty$= 1.0$ on the validation set. We do not perform checkpoint averaging. To select the best performing model in a sweep, we compare BLEU on the validation set.

5 Multilingual Finetuning Performance

We evaluate the performance of multilingual finetuning on the ML50 Benchmark — we compare multilingual finetuning models with bilingual training from scratch, bilingual finetuning, and multilingual training from scratch. Results of multilingual finetuning comparing to all baselines are displayed in Table 2 (per direction comparison is available in Figure 1). The results demonstrate strong improvement over the baselines on many-to-English and comparable performance on English-to-many directions. We also evaluate multilingual finetuning many-to-many models zero-shot performance on non-English directions without bitext data. Our results demonstrates multilingual finetuning models' strong improvement on zero-shot directions comparing to multilingual models trained from scratch.

Data	Multilingual FT Translation to English						Multilingual FT Translation from English					
ΔBLEU	Bilingual		Bilingual FT		Multilingual SC		Bilingual		Bilingual FT		Multilingual SC	
	M→1	M↔M	M→1	M↔M	M→1	M↔M	1→M	M↔M	1→M	M↔M	1→M	M↔M
>10M	2.4	0.2	0.7	-1.6	1.1	-0.5	-0.3	-1.5	-2.1	-3.3	0.2	0
1M-10M	6.2	4.4	2.3	0.5	1.4	0.3	1.7	0.6	-1.6	-2.7	0.2	-0.4
100k-1M	8.0	7.3	2.4	1.6	2.5	0.4	4.0	3.2	-0.4	-1.2	-0.1	-0.3
10-100K	22.3	20.7	5.5	3.8	4.4	2.3	13.5	13.7	0.1	0.32	-0.2	-0.3
4-10k	18.9	15.0	7.3	3.4	5.8	0.9	10.0	9.7	1.3	1.00	-0.7	-1.2
All	12.0	10.3	3.5	1.8	3.1	-0.1	6.3	5.8	-0.5	-1.0	-0.1	-0.4

Table 2: Multilingual Finetuning on 50 **languages comparing to 3 baselines: (1) bilingual from scratch, (2) bilingual finetuning, and (3) multilingual training from scratch**. Multilingual Finetuning **(a)** consistently improves over all baselines for translation into English (left), while **(b)** performs similarly over bilingual finetuning and multilingual from scratch with significant improvement over bilingual from scratch for translation from English (right). Numbers are average *BLEU difference* between multilingual finetuning models and the corresponding baselines. Per direction comparison is available in Figure 1.

5.1 Comparison to Bilingual Finetuning

To understand whether the benefit of transfer learning across languages can be stacked on top of finetuning pretrained models, we analyze the improvement of multilingual finetuning with the same model size as bilingual finetuning in Table 2.

In the Many-to-one setting, every language pair is improved by multilingual finetuning except one. Some low resource languages see substantial improvement of more than 10 BLEU points, with the largest improvement being over 15 BLEU points. On average, multilingual finetuning improves 3.5 BLEU across all directions into English. In the one-to-Many setting, performance is about the same between multilingual finetuning and bilingual finetuning with average gap of -0.5 BLEU. In many-to-many setting, on average multilinugal finetuning improves the performance of translation into English by 1.8 BLEU while with -1.0 BLEU behind for translation from English. We hypothesize that the benefit of pretraining is diminished by the challenge of decoding into many target languages in multilingual compared to bilingual finetuning.

5.2 Comparison to Multilingual from Scratch

To understand the impact of pretraining-finetuning paradigms for multilingual translation, we examine our proposed multilingual finetuning method comparing to multilingual models trained from scratch. As shown in Table 2, in Many-to-One setting, multilingual finetuning performs consistently better than multilingual model trained from scratch by 3.1 BLEU on average. For low resource directions (4k-10k bitexts), the improvement is as high as 5.8 BLEU. However, in the One-to-Many and Many-to-Many settings, multilingual finetuning does not perform better than multilingual training from scratch. For translation from English, One-to-Many multilingual finetuning performs -0.1 BLEU points worse than multilingual from scratch on average; many-to-many multilingual finetuning model performs -0.4 BLEU worse than multilingual from scratch on average. On translation into English, we also observe that many-to-many multilingual finetuning models performs -0.1 BLEU worse than multilingual from scratch on average. Again we hypothesize that the benefit of monolingual data pretraining is dominated by the challenges of a large amount of decoding tasks for individual target languages. We will discuss the challenges of to-many translation further in Section 6.1.

5.3 Comparison to Bilingual from Scratch

To understand the combined benefits of pretraining-finetuning and multilingual transfer learning, we examine the improvement of multilingual finetuning with the same model size over bilingual from scratch in Table 2. In the Many-to-one setting, every language pair is improved by multilingual finetuning — on average multilingual finetuning improves over bilingual models by 12.0 BLEU. Some low and mid resource languages see substantial improvement of more than 20 BLEU points (see Figure 1). In the one-to-Many setting, multilingual finetuning outperforms almost all bilingual models except for 5 directions with minor gaps (mostly less than 1 BLEU). In many-to-many setting, multilingual finetuning improves translation

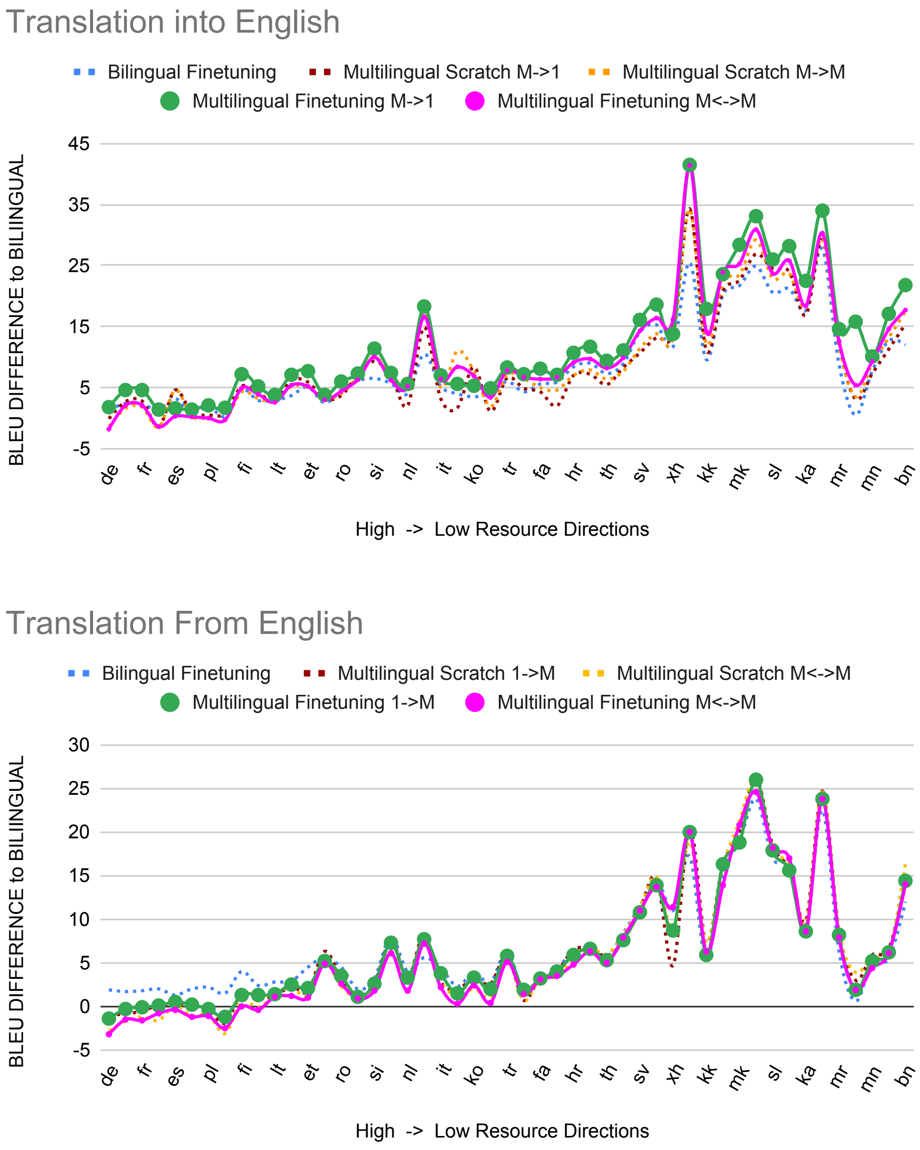

Figure 1: Multilingual Finetuning and Other Baselines Comparing to Bilingual Models for 50 Languages Translation. Y-Axis numbers are BLEU difference to bilingual models trained from scratch.

into English by 10.3 BLEU while with 5.8 BLEU improvement for translation from English. Thus concludes that multilingual finetuning can achieve the significant improvement over bilingual baselines across all directions translation into English and from English.

5.4 Zero-shot on Non-English Directions

We study the impact of multilingual finetuning on zero-shot non-English directions without any bitext training data. We evaluate multilingual many-to-many scratch and finetuning over WMT 13 and 20 test data (fr-de and de-fr test data are from WMT20 (Barrault et al., 2020) and the other test data is from WMT 13 (Bojar et al., 2013)). As shown in Table 4, many-to-many multilingual finetuning model outperforms many-to-many multilingual from scratch models by a large margin with average 11.9 BLEU improvement. We hypothesize that the zero-shot non-English translation performance gain is from two factors (1) that pretrained mBART multilingual encoders and decoders are well-trained with monolingual data; (2) that pretrained mBART decoders are not coupled with specific source languages as multilingual scratch models. Note that decoders of multilingual models from scratch are always trained with English as the source language in the encoders while multilingual finetuning models' decoders are trained with both English and the target

Model	de	cs	fr	ja	es	ru	pl	zh	fi	lv	lt	hi
ML-FT M→1	41.5	34.2	39.8	20.5	28.6	39.1	34.1	26.8	31.3	23.1	31.6	27.2
ML-FT M↔M	37.9	31.7	37.3	17.7	27.3	37.9	32.0	24.8	29.0	21.8	30.4	25.5
ML-FT 1→M	38.6	24.5	38.9	23.7	29.5	28.7	22.3	32.4	21.0	17.9	14.7	20.0
ML-FT M↔M	36.8	23.3	37.4	22.8	28.6	27.3	21.5	31.1	19.7	16.2	14.4	18.7

Model	et	ta	ro	ps	si	ml	nl	ne	it	ar	ko	he
ML-FT M→1	30.9	18	38.6	16.2	17.5	19.9	38.1	21.1	43.9	39.1	21.7	43.5
ML-FT M↔M	28.4	17.2	37.0	15.2	16.1	18.7	37.7	19.4	43.3	41.9	23.3	42.0
ML-FT 1→M	19.6	30.9	36.4	8.4	4.1	24.8	32.6	9.0	37.5	21.2	19.4	29.0
ML-FT M↔M	18.5	30.6	35.5	8.2	3.3	23.6	31.1	8.5	35.9	20.0	18.5	27.4

Model	tr	km	fa	vi	hr	uk	th	id	sv	pt	xh	af
ML-FT M→1	24.8	11.2	35.7	33.1	44.3	36.2	30.3	39.1	46.9	49.3	14.2	42.5
ML-FT M↔M	24.3	10.7	34.0	32.7	42.7	34.2	29.1	37.9	45.1	47.1	16.6	42.4
ML-FT 1→M	22.1	6.2	18.3	32.5	31.9	24.4	36.0	34.8	37.8	41.0	8.9	20.4
ML-FT M↔M	21.4	5.7	18.2	32.0	30.8	24.1	35.7	35.1	38.0	40.8	11.6	20.4

Model	kk	ur	mk	te	sl	my	ka	gl	mr	gu	mn	az
ML-FT M→1	19.3	31.4	42.5	44.0	33.9	32.1	28.6	40.6	17.4	15.8	13.6	19.9
ML-FT M↔M	15.6	31.7	39.4	41.8	31.6	29.7	24.5	36.9	15.4	5.4	12.8	17.4
ML-FT 1→M	6.5	24.6	27.0	41.0	22.8	35.4	12.3	28.0	13.4	1.9	8.5	8.1
ML-FT M↔M	6.9	22.2	29.0	39.6	23.1	36.8	12.3	28.0	13.1	1.9	7.7	8.0

Table 3: Multilingual Finetuning BLEU scores over 50 languages

language as source languages in encoders. This result echos the findings in (Gu et al., 2019) regarding the importance of decoupling source and target languages encoders and decoders learning in zero-shot translation.

tgt src	model	cs	de	es	fr
cs	ML-SC	-	3.1	2.9	2.2
	ML-FT	-	16.3	19.1	13.5
de	ML-SC	2.3	-	3.1	2.5
	ML-FT	13.8	-	16.2	12.2
es	ML-SC	2.2	2.7	-	2.7
	ML-FT	10.6	13.3	-	15.8
fr	ML-SC	2.3	3	3.4	-
	ML-FT	8.7	14.2	21	-

Table 4: Multilingual Finetuning Many-to-Many Model raw BLEU scores on Zero-shot non-English Directions: Multilingual Finetuning (ML-FT) consistently outperforms Multilingual Scratch (ML-SC) over all zero-shot directions with large margin

6 Discussion

6.1 Challenges of To-Many Directions

In the Many-to-one setting, large improvements are obtained by using pretrained models as a starting point. Multilingual modeling increases the quantity of target-side English data seen by the model. For example, compared to bilingual finetuning, our multilingual finetuning model is exposed to English target side data from 50 different language pairs.

However, in the one-to-Many setting and the Many-to-Many setting, models must decode into 50 different languages in both multilingual paradigms — being either trained from scratch or pretrained-and-finetuned. As shown in Table 2 (and Table 9, Figure 1), multilingual models — either from scratch or multilingual finetuning — perform worse than bilingual finetuning for English to Many. This indicates that the challenge of decoding into many languages is a dominating factor in the multilingual models, even with pretraining. Note that there are 49 decoding tasks in One-to-Many and 50 decoding tasks in Many-to-Many, while only 1 in Many-to-One. Additional research, for example following

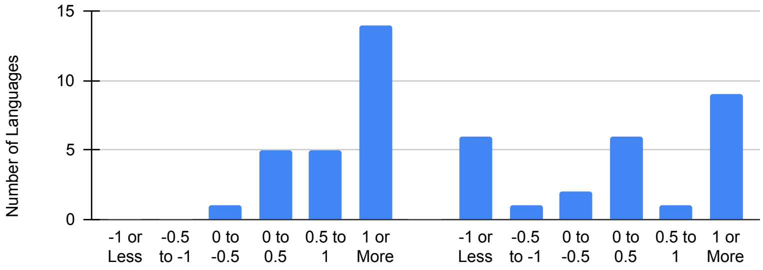

Figure 2: **mBART50 and mBART25 Bilingual Finetuning BLEU Delta**. mBART50 is better than mBART25 over new languages on average. (left) Translation into English; (right) Translation out of English.

the study framework used in (Grönroos et al., 2020), is needed to understand **(1)** the interaction between pretraining and finetuning and multiple decoding tasks, and **(2)** the difference between multiple encoding tasks and multiple decoding tasks.

6.2 Continuous Pretraining is Effective

Pretraining models at large scale is costly. By proposing multilingual finetuning, we introduce a dependency on pretrained models for multilingual translation, which can be a limitation if the pretrained model does not cover the desired languages for translation. Thus, we examine the possibility and effectiveness of incrementally extending pretrained models to support additional languages. We found that for the languages which are supported by the original pretrained models, bilingual finetuning from both previously pretrained and continuously pretrained models demonstrate the almost exactly the same performance (see Figure 3 for our analysis of the bilingual finetuning performance of both models over the original 25 languages). Thus, extending pretraining does not hurt performance on the originally supported languages, despite doubling the number of languages supported by the pretrained model. This removes a big limitation of using pretrained models — that users are often limited to choices made during the original pretraining, and thus if languages are not supported, they cannot be used.

We also examine the effectiveness of such continued pretraining. We find that mBART50 has stronger bilingual finetuning performance (see Figure 2) than mBART25 over the newly supported 25 languages on average, indicating that pretrained models are able to be extended to support additional languages if model capacity allows.

7 Conclusion

We demonstrate that multilingual translation models can be created from pretrained models such as mBART using *multilingual finetuning*. While using pretrained models could theoretically limit the number of languages, we show that mBART can be extended to double the number of original languages without loss of performance. To train and evaluate on 50 languages, we develop and release the ML50 benchmark. We show that by performing multilingual finetuning, strong improvements can be achieved in the Many-to-one setting. However, pretraining and finetuning paradigm alone is not enough to address the challenges of multilingual models for One-to-Many. Our future work will include analysis of improved strategies for One-to-Many translation, model capacity and inference latency trade-off, an in-depth study of zero-shot translation, training strategies for better data efficiency, and applications of the universal text representation and generation frameworks in other crosslingual tasks.

References

Jade Abbott and Laura Martinus. 2019. Benchmarking neural machine translation for Southern African languages. In *Proceedings of the 2019 Workshop on Widening NLP*, pages 98–101, Florence, Italy. Association for Computational Linguistics.

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang

Macherey. 2019a. The missing ingredient in zero-shot neural machine translation. *arXiv preprint arXiv:1903.07091*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. 2019b. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Graeme Blackwood, Miguel Ballesteros, and Todd Ward. 2018. Multilingual neural machine translation with task-specific attention. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3112–3122.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Kumiko Sudoh, Kyotaro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *International Workshop on Spoken Language Translation*, pages 2–14.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods* *in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2021. Recipes for adapting pre-trained monolingual and multilingual models to machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3440–3453, Online. Association for Computational Linguistics.

Raj Dabre, Atsushi Fujita, and Chenhui Chu. 2019. Exploiting multilingualism through multistage fine-tuning for low-resource neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1410–1416, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond English-centric multilingual machine translation. *arXiv preprint*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Xavier Garcia, Pierre Foret, Thibault Sellam, and Ankur Parikh. 2020. A multilingual view of unsupervised machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3160–3170, Online. Association for Computational Linguistics.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2020. Transfer learning and subword sampling for asymmetric-resource one-to-many neural translation. *MACHINE TRANSLATION*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. 2019. Improved zero-shot neural machine translation via ignoring spurious correlations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1258–1268, Florence, Italy. Association for Computational Linguistics.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *International Workshop on Spoken Language Translation*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multitask benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi

Parallel Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Surafel M Lakew, Marcello Federico, Matteo Negri, and Marco Turchi. 2018. Multilingual neural machine translation for low-resource languages. *IJCoL. Italian Journal of Computational Linguistics*, 4(4-1):11–25.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020b. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649–2663, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 84–92, Brussels, Belgium. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Toshiaki Nakazawa, Nobushige Doi, Shohei Higashiyama, Chenchen Ding, Raj Dabre, Hideya Mino, Isao Goto, Win Pa Pa, Anoop Kunchukuttan, Yusuke Oda, Shantipriya Parida, Ondřej Bojar, and Sadao Kurohashi. 2019. Overview of the 6th workshop on Asian translation. In *Proceedings of the 6th Workshop on Asian Translation*, pages 1–35, Hong Kong, China. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. FAIRSEQ: A fast, extensible toolkit for sequence modeling. In *North American Association for Computational Linguistics (NAACL): System Demonstrations*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271.

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2827–2835, Online. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning (ICML)*.

Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with language clustering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 963–973, Hong Kong, China. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Chau Tran, Yuqing Tang, Xian Li, and Jiatao Gu. 2020. Cross-lingual retrieval for iterative self-supervised training. In *Advances in Neural Information Processing Systems*, volume 33, pages 2207–2219. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

A Appendix

Model	Encoder layers	Decoder layers	Embedding	FFN embedding	Heads
Standard	5	5	512	2048	8
Big	6	6	1024	4096	16
Large	12	12	1204	4096	16

Table 5: Baseline transformer model architectures

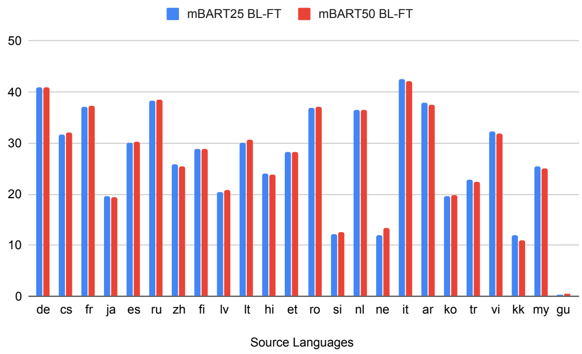

(a) Translation into English

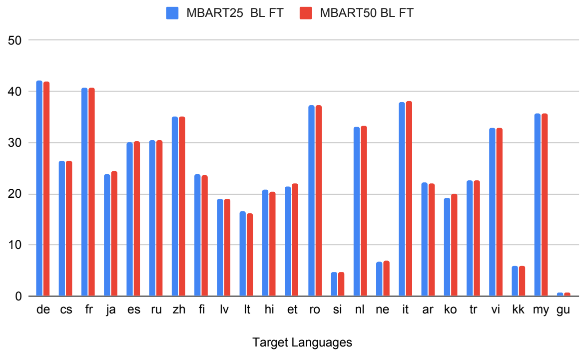

(b) Translation out of English

Figure 3: Comparing mBART50 and mBART50 bilingual finetunign on the mBART25 languages.

	ML50 Train		ML50 Eval		
Language	# Sentences	Source	Source	# Sentences Valid	# Sentences Test
af	45967	Opus	LauraMartinus	1500	2686
ar	226073	IWSLT17	IWSLT17	1158	1460
az	5680	TED58	TED58	671	903
bn	4487	TED58	TED58	896	216
cs	42587802	WMT20	WMT19	2983	1997
de	45828203	WMT20	WMT19	2998	2000
es *	14524187	WMT13	WMT13	3003	3000
et	1052003	WMT18	WMT18	2000	2000
fa	144895	TED58	TED58	3930	4490
fi *	2353313	WMT17	WMT17	3000	3002
fr	36797950	WMT14	WMT14	3000	3003
gl	9504	TED58	TED58	682	1007
gu	7471	WMT19	WMT19	1998	1016
he	204380	TED58	TED58	4515	5508
hi	1327206	ITB	ITB	520	2507
hr	116792	TED58	TED58	3333	4881
id	83944	TED58	TED58	2677	3179
it	226457	IWSLT17.mltlng	IWSLT17.mltlng	1566	1147
ja *	16167141	WMT20	WMT20 dev-split	999	999
ka	12364	TED58	TED58	654	943
kk	29186	WMT19	WMT19	2066	1000
km	191967	WMT'20	Flores devtest	2378	2309
ko	224612	IWSLT17	IWSLT17	1143	1429
lt *	1395010	WMT19	WMT19	2000	1000
lv *	1808291	WMT17	WMT17	2003	2001
mk	24037	TED58	TED58	640	438
ml	358916	lotus	lotus	500	1000
mn	7168	TED58	TED58	372	414
mr	9397	TED58	TED58	767	1090
my	18073	WAT19	WAT19	1000	1018
ne	227387	Flores	Flores	2559	2924
nl	232572	IWSLT17.mltlng	IWSLT17.mltlng	1777	1181
pl	10332683	WMT20	WMT20 dev-split	1000	1000
ps	579346	WMT'20	Flores devtest	3162	2698
pt	49446	TED58	TED58	1193	1803
ro	592594	WMT16	WMT17	1999	1999
ru *	13922899	WMT20	WMT19	3000	2000
si	565661	Flores	Flores	2898	2905
sl	18751	TED58	TED59	1068	1251
sv	53596	TED58	TED58	1729	2283
ta	609767	WMT'20	WMT20 dev-split	995	994
te	22042	lotus	lotus	500	1000
th	93723	TED58	TED58	2989	3713
tr	204200	WMT17	WMT17	3000	3007
uk	104193	TED58	TED58	3060	3751
ur	26302	lotus	lotus	500	1000
vi	127069	IWSLT 15	IWSLT15	1268	1080
xh	48981	Opus	LauraMartinus	1500	2717
zh *	10082367	WMT20	WMT19	3981	2000

Table 6: ML50 Benchmark dataset stats. For each language, we list the size of training data after the filtering steps, the source of training/evaluation data, and the size of evaluation data. We notice that part of the available dataset are missing due to human error for a few language pairs. We mark these languages with asterisk and we will release next version of the ML50 benchmark data to include the missing data.

Lang	de	cs	fr	ja	es	ru	pl	zh	fi	lv	lt	hi
BL-Scratch to en	39.7	29.0	35.2	18.4	27	37.7	28.4	25.1	24.1	17.9	27.8	20.1
BL-FT to en	41.0	32.0	37.4	19.5	30.2	38.5	31.0	25.4	28.8	20.8	30.7	23.8
BL-Scratch from en	40	24.8	39	22.2	29	28.5	24.3	33.6	19.7	16.6	13.3	17.5
BL-FT from en	41.9	26.5	40.8	24.5	30.3	30.5	26.7	35.1	23.7	19.0	16.1	20.4

Lang	et	ta	ro	ps	si	ml	nl	ne	it	ar	ko	he
BL-Scratch to en	23.2	14.2	32.6	8.9	6.1	12.5	32.5	2.8	36.9	33.5	16.4	38.6
BL-FT to en	28.3	18.2	37.1	15.0	12.6	18.2	36.5	13.3	42.1	37.5	19.9	42.7
BL-Scratch from en	17.5	28.7	32.9	7.3	1.5	17.5	29.3	1.3	33.7	19.7	16.1	27.0
BL-FT from en	22.0	34.0	37.4	9.3	4.7	25.5	33.3	6.9	38.1	22.0	20.0	29.7

Lang	tr	km	fa	vi	hr	uk	th	id	sv	pt	xh	af
BL-Scratch to en	16.5	4.0	27.6	26.0	33.6	24.5	20.9	28.0	30.8	30.7	0.4	1.0
BL-FT to en	22.5	8.3	33.2	31.9	42.0	33.5	28.2	36.9	44.9	46.0	12.1	26.5
BL-Scratch from en	16.3	4.3	15.1	28.5	26.0	17.8	30.7	27.2	27.0	27.1	0.2	1.0
BL-FT from en	22.7	5.9	18.4	32.9	32.2	24.3	36.5	35.6	38.5	41.6	11.2	18.3

Lang	kk	ur	mk	te	sl	my	ka	gl	mr	gu	mn	az
BL-Scratch to en	1.4	7.8	14.1	10.9	7.9	3.9	6.1	6.6	2.8	0.0	3.5	2.8
BL-FT to en	11.0	28.0	35.8	35.8	28.5	25.1	23.8	34.3	11.6	0.5	11.2	15.5
BL-Scratch from en	0.6	8.3	8.2	15.0	4.9	19.8	3.7	4.2	5.2	0.0	3.3	1.9
BL-FT from en	5.9	23.7	27.2	38.8	21.9	35.8	13.0	26.7	11.5	0.6	8.5	7.4

Table 7: Bilingual and Finetuning Bilingual Baselines over 50 languages

Lang	de	cs	fr	ja	es	ru	pl	zh	fi	lv	lt	hi
ML-Scratch M→1	39.6	32.3	38.0	19.2	31.6	38.6	30.6	25.9	29.3	22.1	30.5	26.3
ML-Scratch M↔M	38.3	31.2	37.0	17.5	31.6	38.0	29.9	24.8	28.4	21.1	30.5	25.3
ML-Scratch 1→M	39.1	23.9	38.5	20.9	29.3	28.6	24.6	31.7	21.2	17.6	14.5	19.8
ML-Scratch M↔M	37.2	23.1	37.8	20.0	29.1	27.4	23.1	30.5	20.3	16.5	14.6	19.7

Lang	et	ta	ro	ps	si	ml	nl	ne	it	ar	ko	he
ML-Scratch M→1	29.1	20.5	36.3	16.0	15.4	19.5	34.5	17.7	40.1	51.0	29.2	39.7
ML-Scratch M↔M	28.3	19.9	36.6	15.7	16.2	19.2	37.6	20.3	41.9	44.5	24.1	40.5
ML-Scratch 1→M	19.2	33.3	36.1	8.4	4.2	25.0	32.6	9.4	36.5	21.7	19.3	29.6
ML-Scratch M↔M	18.6	32.1	35.2	8.3	3.9	23.8	31.9	9.1	36.6	20.9	18.1	28.1

Lang	tr	km	fa	vi	hr	uk	th	id	sv	pt	xh	af
ML-Scratch M→1	23.1	8.9	31.9	28.0	40.6	31.7	26.4	36.3	41.5	43.9	14.5	35.7
ML-Scratch M↔M	23.6	10.5	32.6	30.6	40.6	32.4	27.3	35.7	42.2	44.5	13.5	35.1
ML-Scratch 1→M	22.1	5.0	18.5	32.5	32.5	24.4	36.5	34.7	38.2	41.9	4.9	20.3
ML-Scratch M↔M	21.7	5.0	18.3	31.9	31.6	24.5	36.7	35.4	38.4	42.0	8.9	17.6

Lang	kk	ur	mk	te	sl	my	ka	gl	mr	gu	mn	az
ML-Scratch M→1	12.5	28.6	36.7	37.8	32.4	27.9	23.0	35.8	14.9	3.1	10.8	14.1
ML-Scratch M↔M	13.6	30.2	37.6	40.1	30.8	27.6	24.2	36.0	14.9	3.5	12.5	16.0
ML-Scratch 1→M	7.9	24.6	28.3	41.2	23.4	35.5	13.5	28.9	13.9	3.0	9.2	8.5
ML-Scratch M↔M	7.9	24.3	29.5	41.2	22.6	36.3	13.2	28.8	13.8	3.9	9.1	7.9

Table 8: Multilingual Baselines over 50 languages

Data	Translation to English					Translation from English				
	BL-FT over Bilingual	**ML-SC over Bilingual**		**ML-FT over Bilingual**		**BL-FT over Bilingual**	**ML-SC over Bilingual**		**ML-FT over Bilingual**	
		M→1	M↔M	M→1	M↔M		1→M	M↔M	1→M	M↔M
>10M	1.7	1.4	0.6	**2.4**	0.1	**1.8**	-0.5	-1.6	-0.3	-1.5
1M-10M	3.9	4.8	4.1	**6.2**	4.4	**3.3**	1.5	1.0	1.7	0.6
100k-1M	5.6	5.5	6.9	**8.0**	7.2	**4.4**	4.1	3.5	4.0	3.2
10K-100K	16.8	17.9	18.4	**22.3**	20.7	13.4	13.7	**14.0**	13.5	13.7
4k-10k	11.6	13.1	14.1	**18.9**	15.0	8.7	10.6	**10.9**	10.0	9.7
All	8.5	9.0	9.5	**12.0**	10.3	**6.8**	6.4	6.1	6.3	5.8

Table 9: Multilingual Finetuning on 50 **languages comparing to bilingual models**. Numbers are average BLEU difference compared to bilingual models trained from scratch.

Annotations Matter: Leveraging Multi-task Learning to Parse UD and SUD

Zeeshan Ali Sayyed
Indiana University
Department of Computer Science
zasayyed@iu.edu

Daniel Dakota
Uppsala University
Department of Linguistics
ddakota@lingfil.uu.se

Abstract

Using multiple treebanks to improve parsing performance has shown positive results. However, to what extent similar, yet competing annotation decisions play in parser behavior is unclear. We investigate this within a multi-task learning (MTL) dependency parser setup on two parallel treebanks, UD and SUD, which, while possessing similar annotation schemes, differ in specific linguistic annotation preferences. We perform a set of experiments with different MTL architectural choices, comparing performance across various input embeddings. We find languages tend to pattern in loose typological associations, but generally the performance within an MTL setting is lower than single model baseline parsers for each annotation scheme. The main contributing factor seems to be the competing syntactic annotation information shared between treebanks in an MTL setting, which is shown in experiments against differently annotated treebanks. This suggests that the impact of how the signal is encoded for annotations and its influence on possible negative transfer is more important than that of the input embeddings in an MTL setting.

1 Introduction

Multi-task learning (MTL; Caruana, 1997) has shown promise in various NLP tasks such as semantic dependency parsing (Peng et al., 2017; Hershcovich et al., 2018; Kurita and Søgaard, 2019), machine translation (Dong et al., 2015) and mulitiword expression detection (Taslimipoor et al., 2019).

MTL inherently is designed to share information between tasks, which has helped various NLP components (Collobert and Weston, 2008). One active research question however is what information in specific tasks should be shared, as well was what indicators can be used to predetermine the cost-benefit trade-offs of MTL for a given application. Findings have shown that label distributions (Martínez Alonso and Plank, 2017), data sizes (Bollmann et al., 2018) and single task loss curves (Bingel and Søgaard, 2017) have all been respective indicators for MTL performance. Different tasks, data sizes, and settings can all show different relative performance gains (Adouane and Bernardy, 2020). Thus, it is still an open question under which circumstance MTL can be used to achieve max performance boosts over a single task system.

In syntactic parsing, learning a closely related task (e.g. POS tagging) in a joint paradigm benefits overall performance (Bohnet and Nivre, 2012; Zhang and Weiss, 2016), and work has also exploited MTL by leveraging two or more treebanks against each other (see section 2). We often assume simply increasing data and the sharing of syntactic information will inherently benefit all parsers, but this assumes that all syntactic sharing, specifically all annotation sharing, is positive and complementary. However, annotation decisions have been shown to favor parsing preferences (Rosa, 2015; Rehbein et al., 2017; Kohita et al., 2017). This means that is is not necessarily clear if sharing annotations benefits all parsers equally. This is especially true if two annotation schemes choose drastically different approaches when annotating specific linguistic phenomena.

We look to examine this issue further by utilizing a set of treebanks that are annotated on parallel data, Universal Dependencies (UD; Nivre et al., 2016) and Surface-Syntactic Universal Dependencies (SUD; Gerdes et al., 2018), to examine how two competing syntactic annotation schemes behave when used in an MTL setup. Using parallel treebanks also removes the lexical variation and influences of domain differences that are present in most MTL treebank setups. Whether this is a positive or negative in an MTL setup is unclear,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3467–3481
August 1–6, 2021. ©2021 Association for Computational Linguistics

but reduction in domain differences tend to benefit single model parsers.

We utilize the graph-based Deep Biaffine Parser of Dozat and Manning (2017) in an MTL architecture, treating each UD and SUD treebank of a selected language as a task, and experiment with sharing different embeddings, layers, and loss functions. Additionally, we look at the how different embeddings interact with these annotations along with their role in encoding the signal utilized by the MTL parsers, and whether results follow any linguistic patterns. Finally, we perform additional experiments with treebanks from SPMRL shared task (Seddah et al., 2013, 2014) to support our analysis. We look to investigate the following questions:

1. How will competing syntactic annotations schemes on parallel treebanks behave in an MTL parser?
2. What impact do different input embeddings have on behavior in such a setup?
3. When a treebank is paired with a non-parallel treebank possessing noticeably different syntactic annotations, do trends hold?

2 Related Work

The use of multiple treebanks has been successfully incorporated in parsing strategies. Recent multilingual multi-treebank work by Schuster et al. (2019) extended the Biaffine parser by Dozat and Manning (2017) to incorporate deep contextualized multilingual embeddings in combination with multiple treebank sources, demonstrating gains for zero-shot parsing. Smith et al. (2018a) noted using smaller groups of closely related languages is preferable to larger datasets of dissimilar ones. Multiple synthetic treebanks derived from closely related languages were used to parse Faroese by Barry et al. (2019), though a single language source model yielded the best results.

More directly related work is Johansson (2013), who shares features between two treebanks of the same language that differ in annotation schemes by identifying overlapping features. Using a graph-based parser, he achieved noticeable relative error reduction in UAS for four language pairs, with the largest performance gains on the smaller treebanks. This was followed by Johansson and Adesam (2020) using a neural transition-based parser and leveraging a mixture of treebanks, three dependency and two constituency, against a single constituency treebank in a multi-treebank setup. They find that in all settings, performance on the target constituency treebank improves, with the highest gain coming from using all five as an auxiliary treebank. Kankanampati et al. (2020) use the Multidimensional Easy First approach introduced by Constant et al. (2016) to parse the Arabic CATiB (Habash and Roth, 2009) and its converted UD representation in a multi-task setup. They note that both treebanks showed error reduction, but that improvements were due to partial dependencies, and not primarily driven through lexical sharing.

Little direct work exists on extensive empirical investigations between UD and SUD with parsers. Recent work by Kulmizev et al. (2020) performed probing experiments across a set of languages to extract dependency graphs from BERT (Devlin et al., 2019) and ELMO (Peters et al., 2018) language models, finding that both models prefer UD, with tree shape directly correlated to preference strength.

One of the advantages of MTL is the ability to share information as well as altering objective functions between tasks. Early work examined the impact different loss functions have on downstream applications (Hall et al., 2011) and how in a hierarchy of tasks, sharing of individual layers benefits other tasks differently, with lower level task sharing most beneficial (Søgaard and Goldberg, 2016).

Both hard and soft sharing of parameters have proven successful. Duong et al. (2015) exploited soft parameter sharing between different cross-lingual treebanks possessing the same annotation schemes achieving results on the target language with only half the needed annotated data. Soft sharing of parameters allows nuances between languages of the same treebank when hard sharing all other parameters (Stymne et al., 2018).

Parameter sharing has proven effective in both monolingual (Guo et al., 2016) and multilingual parsing (Ammar et al., 2016; Kitaev et al., 2019). However, what are the optimal parameters to share, and where to do so in the architecture, particularly in cross-lingual setups, is not consistent as shown by de Lhoneux et al. (2018) in extensive experiments in sharing word and character LSTM parameters.

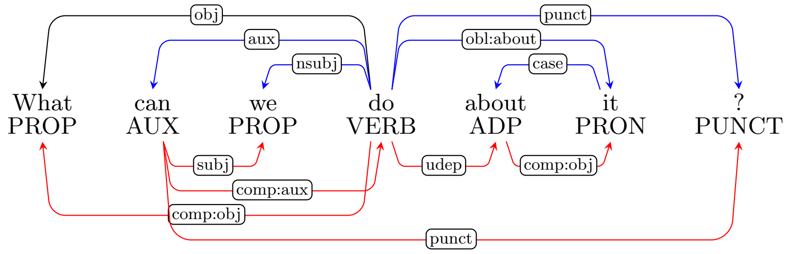

Figure 1: UD and SUD annotation example from English EWT Treebank

Treebank			ar-padt	de-gsd	el-gdt	en-ewt	fi-tdt	fr-gsd	hu-szeged	ko-gsd	ru-gsd	tr-imst	vt-vtb	zh-gsd
UD	Train	Non-Proj	.0892	.0950	.0596	.0523	.0606	.0399	.2572	.1620	.0623	.1105	.0314	.0233
	Dev	Non-Proj	.0825	.0626	.0521	.0275	.0689	.0407	.3356	.1568	.0587	.1204	.0213	.0014
SUD	Train	Non-Proj	.2331	.2118	.1420	.0948	.1423	.0857	.3384	.1764	.0901	.1444	.0957	.4412
	Dev	Non-Proj	.2090	.1815	.1266	.0679	.1645	.0800	.4082	.1684	.0933	.1478	.1075	.4320
	Total Train		6075	13814	1662	12543	12217	14449	910	4400	3850	3664	1400	3997
	Total Dev		909	799	403	2002	1364	1476	441	950	579	988	800	500

Table 1: Proportion of Non-Projective Trees in UD and SUD Train and Dev Sets

3 Experimental Setup

3.1 Data

UD have become a de facto standard as a source for treebanks for dependency parsing. A main annotation choice in UD is the prioritization of content words as the head. While some functional distinctions are kept, such as those between subjects and objects, many other are merged, such as complements and adjuncts. Importantly, function words are dependents of the content words.

SUD were developed as a counter-balance to UD with the belief that UD are not syntactically motivated enough, with a particular linguistically argued objection to the prioritization of content words as heads, stemming from the belief that the distributional context of words should drive headedness. While many individual labels are kept, several are collapsed into a single label (e.g. nsubj & csubj → subj). The primary result of function words becoming heads is the inherent reversal of syntactic relationships of many words.

Fig. 1 is an example of how the SUD conversion alters an English sentence from its original UD representation. One of the more noticeable differences is that the projective UD tree is now non-projective in the SUD schema. The main cause, in this example, is because the auxiliary verb *can* is now the root in SUD, rather than the content word *do* in UD. Furthermore, the only word to retain the same head word between the two sentences is *what*, while all others have new heads. We wish to emphasize however, that not all trees show such stark contrasts, but simply want to highlight how a simple choice in annotation can produce distinctly different trees, and the resultant impact on non-projectivity.

By using UD and SUD, we eliminate one of the variables in many multi-treebank setups, the different distribution of the underlying vocabulary. This effectively eliminates domain differences between the treebanks (see section 3.2), as both parsers will get more similar outputs from the BiLSTM layer, and identical ones in a joint loss setting.

We Use UD and SUD version 2.7 and select 12 different language from 10 language families. This was done in order capture sufficient linguistic variation in terms of how UD and SUD may impact various linguistic phenomena found in typologically different languages, and subsequently annotation schemes.[1] Table 1 presents statistics on the treebanks in respect to their variation in training and dev sizes. Additionally, we also note the proportion of non-projective trees found in each annotation scheme.[2] All languages show higher number of non-projective trees in SUD when compared to their UD counterparts, but for some it is much more substantial. A noticeable example is Chinese (zh) which has 40% more absolute non-projective trees in its SUD treebank compared to its UD counterpart. Noticeable increases can also be seen in Arabic and German, but most languages show only moderate differences. Hungarian (hu) is

---

[1] We restrict ourselves to treebanks that contained complete training, dev, and test splits.

[2] We note that this is not always entail explicit linguistic non-projectivity, as in many cases punctuation is the source of non-projectivity which can be viewed as non-linguistic.

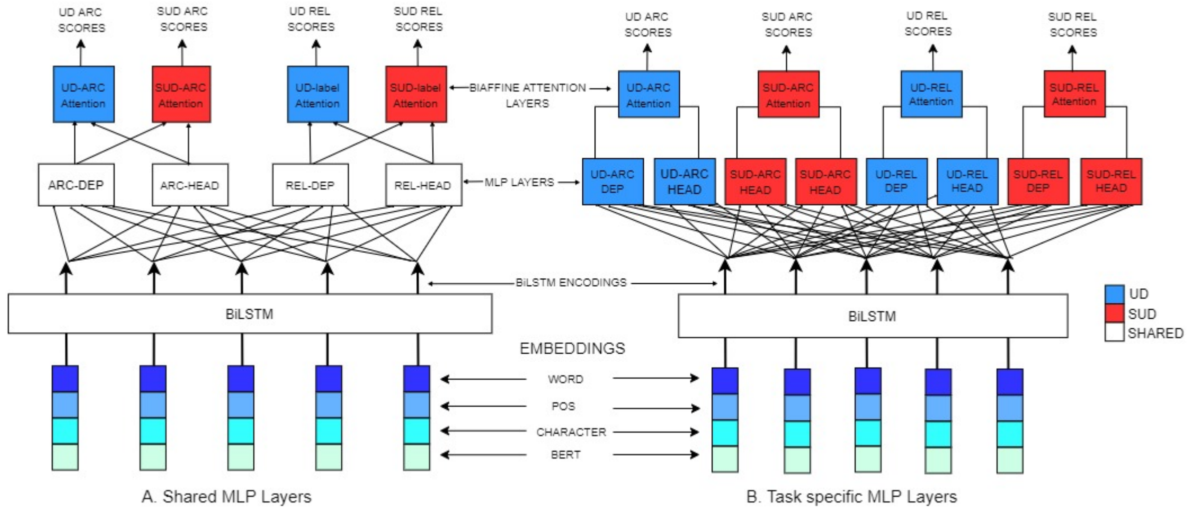

A. Shared MLP Layers

B. Task specific MLP Layers

Figure 2: Multi-task model architecture.

interesting as it is the only language that shows a high proportion of non-projective trees in UD and only a moderate increase for SUD.

3.2 MTL Parsing Architecture

We use the PyTorch (Paszke et al., 2019) implementation of the Biaffine parser of Dozat and Manning (2017) provided by Zhang et al. (2020),[3] and extend it to an MTL architecture.[4]

We modify the base parser by treating parsing of each annotation scheme as a separate task. Each task shares the BiLSTM layer that is used to encode the concatenation of all input embeddings. These BiLSTM encodings are then passed through dimension reducing MLPs to strip away arc and relationship information information deemed not relevant. We implement two MLP schemes, one in which we share them across tasks (shared; Figure 2A) and the other in which each task has its own MLP layers (unshared; Figure 2B). Considering the overlap in the annotation schemes, a shared MLP setting allows us to examine the behavior of sharing information between the two annotation scheme when irrelevant information is minimized. Finally, in order for the model to learn task specific information, we apply task specific biaffine attention layers to the MLP outputs to produce scores for both arcs and labels.

The common practice in MTL is to have separate losses for different tasks and to optimize for each of them separately (*alternating loss*; Ruder, 2017). This is particularly the case when the different tasks do not share the same input. However, our dataset contains parallel sentences albeit with different annotations. It thus then becomes possible to experiment with using a *joint loss* for training both tasks as the parsers receive the same input, and a joint loss has shown improvements when joint learning POS tags and dependency parsing (Li et al., 2018). We do this by optimizing for the sum of losses of each of the tasks. Since the losses of both tasks are of nearly the same magnitude, we do not have to worry about imbalance and a simple sum suffices.[5] We experiment with both types of losses.

In the alternating loss setting, we randomly choose a task from the given tasks and then randomly choose a batch of sentences along with their annotations from that task before calculating the loss of that batch and backpropagating the errors. In a given epoch we chose sentences without replacement. For joint loss, we randomly choose a batch of the same sentences from both the tasks, along with their different annotations. Losses are calculated based on those annotations and summed together before backpropagating the errors. We posit that joint loss should allow for faster convergence as both the tasks affect the parameter updates of the shared layers simultaneously, thus helping the optimization process to move towards the goal more quickly.

The two choices of losses combined with the op-

---

[3] `https://github.com/yzhangcs/parser`

[4] Our code is available at `https://github.com/zeeshansayyed/multiparser`

[5] When losses in an MTL setting do not have comparable magnitude, then the joint loss tends to more influenced by the task with larger loss; thus, producing a learning bias.

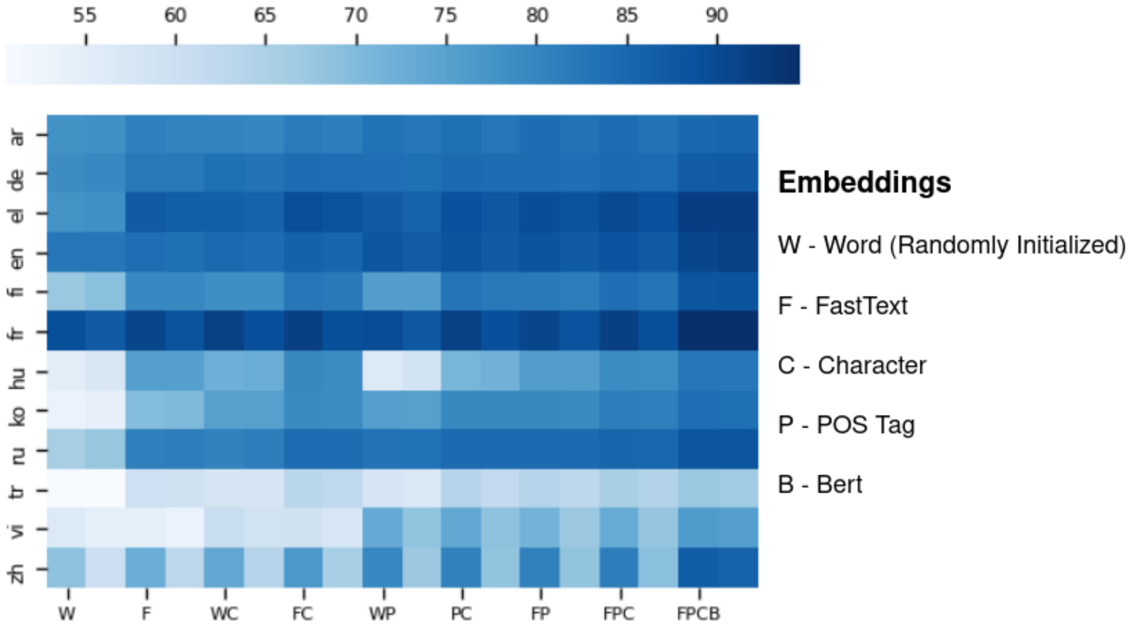

Figure 3: Heatmap illustrating the mean LAS scores for the four MTL settings across all languages and embedding types. UD-LAS is represented in the left column and SUD-LAS in the right column for each embedding input.

tional sharing of MLP layers gives rise to four different experimental settings: alternating-unshared, alternating-shared, joint-unshared, and joint-shared. In addition, we experiment with internally randomly initialized word and POS[6] embeddings, external embeddings (FastText; Bojanowski et al. (2017)) and BERT (Devlin et al., 2019)), and their concatenations as inputs to the BiLSTM layers. All results are reported on the dev sets using the CoNLL 2018 Shared Task Scorer (Zeman et al., 2018).

4 Results

The overall performances of the four experimental settings, namely alternating vs joint loss and shared vs unshared MLP layers, are very close to each other. The convergence statistics for joint and alternating loss settings are reported in Table 2.[7] It can be noted that despite taking a greater number of epochs to converge when compared to alternating loss, joint loss converges faster in terms of time because it performs the forward propagation through the shared layers only once for both tasks, whereas alternating loss has to perform it separately for each task.

As we are more interested in the MTL parser behavior across experimental settings, we report the mean LAS score over the four MTL settings in all our experiments to capture the general trends of the MTL parser.

Parameter	# Epochs	Time (seconds)
Joint Loss	342	2 774
Alternating Loss	297	3 907

Table 2: Convergence statistics for Joint and Alternating Loss

To analyze the impact of different embedding types on the MTL parsing setup, we change the specificity of information by using different embedding types with the MTL parser as discussed in section 3.2, results of which are presented in Fig 3. We see that adding more information yields in higher LAS across languages (moving from left to right on the heatmap) with the concatenation of all embeddings (rightmost columns) performing the best.

However, given that we are more interested in examining whether the parallel UD-SUD treebanks can benefit from an MTL setup, we choose instead to focus on how the MTL parsers compare to the single UD and SUD baseline parsers across the different embedding choices. Fig. 4 shows a heatmap depicting the difference of the mean LAS of all four settings with respect to the corresponding single

---

[6] We use gold POS tags.

[7] All experiments were performed on Nvidia V100 GPUs. Tables 2 and 3 analyses do not include BERT experiments.

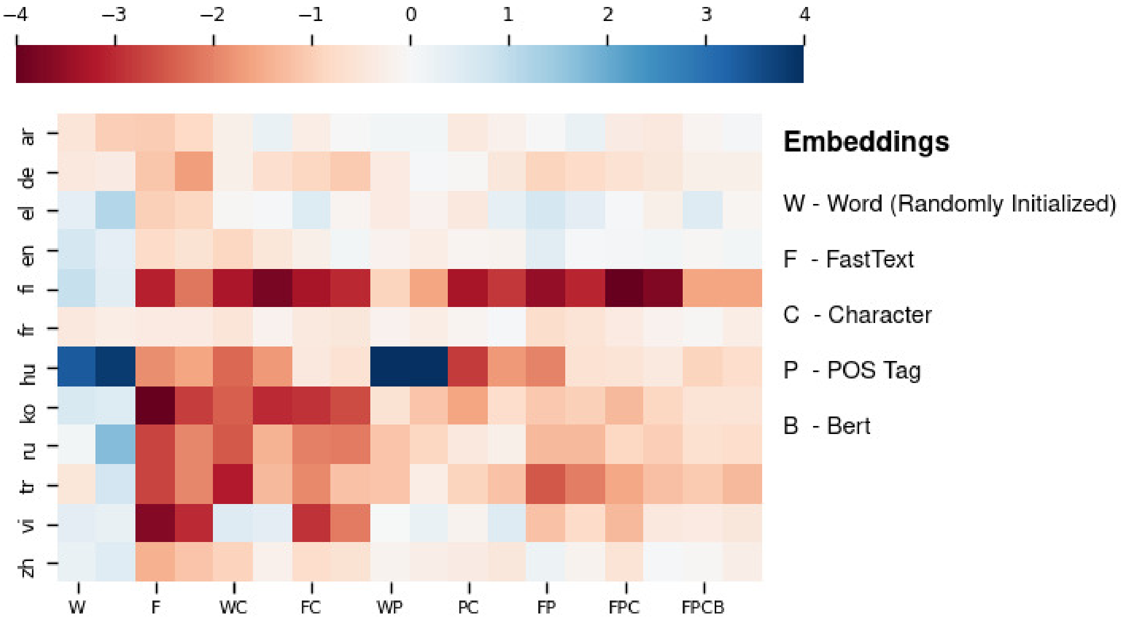

Figure 4: Heatmap illustrating the performance difference between MTL parsers compared to corresponding single task baseline parser. Each block represents the difference between the mean LAS score of four MTL settings and the respective single task baseline LAS score.

baseline parser, for each embedding input.[8]

The mean drop in LAS scores for MTL settings when compared to the baselines across all languages and all the different feature embeddings (432 runs) are reported in Table 3, with lower numbers indicating better performance. No particular setting shows a significant improvement over the other. Keeping this in consideration, we still see that joint loss performs slightly better than alternating loss. Sharing of MLP layers seems to help a little compared to the setting where we have task specific layers. As mentioned in section 3.2, the role of dimension reducing MLPs is to remove all the information that is not necessary for performing the task at hand. This would indicate that the two tasks remove similar unnecessary information, thereby sharing the signal necessary for making parsing decisions.

One of the most striking observations is that randomly initialized word embeddings (seen in the far left two columns) are noticeably lighter across all languages. This stands in stark contrast to the subsequent FastText (FT), word+char and FT+char embeddings. Hungarian shows particularly noticeable improvements, though it may be due to its size. However, given that we also see some moderate improvements for English, Greek and Russian, the size of the treebanks is not the only contributing factor.

Parameter	Mean Drop (LAS)
Joint Loss	0.70
Alternating Loss	0.75
Shared MLP	0.69
Unshared MLP	0.76

Table 3: Mean drop in LAS compared to baseline

Once inputs include word embeddings initialized with FT embeddings or randomized char embeddings, we see some interesting trends. Finnish, Hungarian, Korean, Turkish, and Russian show consistent degradation in performance with any inclusion of character-based embeddings. Some languages show more stable results, regardless of the input embeddings, namely Greek, English, French, and Chinese. Vietnamese clearly performs worse when using FT embeedings to initialize the word embeddings, but otherwise is rather stable in other settings. However, even with BERT (B) embeddings, we do not see any noticeable improvement over the baselines.

---

[8] We also experimented with task-specific fine-tuning following Liu et al. (2019) on MTL parsers. While it did lead to improvements, the overall distribution across all the different settings and languages was not considerably different. Also see Appendix for additional heatmaps contrasting shared vs unshared and alternating vs joint loss settings. The overall pattern remains the same.

When we begin incorporating POS tag embeddings, we see that drops relative to the baseline for these languages become less pronounced, but few languages ultimately show improvements, with Hungarian being the noticeable exception. The pattern in the reduction in degradation of performance when including POS tag embeddings continues across all settings. The exception being Finnish, which still shows large drops in almost all settings, but show slightly reduced drops when including BERT embeddings.

5 Discussion

5.1 UD vs SUD

We see, in general, a systematic decrease in performance when using parallel UD and SUD treebanks in an MTL setup across many languages. When looking for linguistic behaviors, we can clearly see that agglutinative languages (Hungarian, Finnish Turkish, and Korean) all suffer severe performance drops when using character-based embeddings, but the concatenation of POS embeddings helps mitigate the degradation. The absolute differences of Hungarian and Finnish are noticeably different compared to one another. This may be somewhat unexpected, given they are in the same language family. However, the modern forms are quite different and treebank sizes may play a role, as input embeddings pattern similarly overall between the two.

The morphological complexity of other languages in relation to their behavior is not necessarily a good predictor of behavior. However, if we view the other eight remaining languages on a continuum of fusional and analytical properties, we can see some general patterns.

Russian, a fusional language, patterns with the agglutinative languages in its behavior with character embeddings, but is also one of the more morphologically rich languages (MRL) of the non agglutinative languages. The other more fusional MRLs, German and Greek, also do not see as much volatility, although German tends to be worse respective to the baseline, while Greek shows some more positive results, but this could again be due to treebank sizes. English and French are more analytical than the other fusional languages and contain far less morphology. While both show rather consistent minimal degradation regardless of the input embedding, English occasionally shows some improvement, while French virtually none.

Arabic and Vietnamese, however, are somewhat odd cases. Arabic is both fusional and an MRL, whereas Vietnamese is much more analytical. Vietnamese, though, patterns more with the other MRLs, while Arabic patterns more like the analytical language, particularly with its less overall performance degradation compared to baselines across settings. Vietnamese shows one of the larger performance drops relative to the baseline compared to the other eight languages when using FT embeddings in the input, but this is diminished when combined with additional embeddings.

Chinese, an extremely analytical language, presents an additionally interesting case. The LAS for SUD is usually on average 10% absolute lower than its UD counterpart, which can be seen in Fig. 3. This probably is a direct result of the SUD treebank having 40% absolute more non-projective trees. However, this massive disparity in non-projectivity has seemingly not resulted in additional performance degradation in the MTL setup (as seen in Fig. 4), suggesting that sharing between treebanks that show large differences in non-projectivity is not necessarily detrimental.

Given the general behavior across settings, performance degradation can most likely be attributed to negative transfer derived from the different annotation preferences UD and SUD encode, which is not seen in the single model baselines. When different embeddings are used, the negative transfer is either accentuated depending upon the language, as seen with character embeddings, or some embeddings seem to help mitigate the negative transfer, as with POS embeddings. Interestingly, although character-based embeddings show significant improvements in the single baseline models compared to word embeddings, the signals they encode seem to be detrimental in an MTL setting, as performances drops relative to their respective single model baselines. This would seemingly suggest that in an MTL setup, word and POS embeddings are encoding more beneficial signals that help both annotation schemes, reducing possible negative transfer from each treebank, whereas character embeddings are maximally beneficial when used to train a single model. This is in line with recent work showing that the linguistic information POS tags convey, when highly accurate or gold, still have value for specific use cases, and are beneficial in certain dependency parsing architectures or as auxiliary tasks (Anderson and Gómez-Rodríguez,

		Arabic				German				Hungarian			
		Char + FastText				Char + FastText				Char + FastText			
Exp.	MLP	UD		SUD		UD		SUD		UD		SUD	
		UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
	baseline	86.59	82.25	86.50	81.44	89.18	85.11	87.81	84.98	84.73	79.97	84.03	79.02
UD-SUD	unshared	86.33	81.93	86.34	81.33	87.95	84.06	86.74	83.70	84.28	79.42	83.53	78.56
	shared	86.33	81.97	86.22	81.33	87.83	84.08	86.76	83.73	84.50	79.72	85.83	79.05
SPMRL	unshared	87.78	83.47	87.52	82.62	90.16	86.44	89.81	87.18	87.04	82.23	86.99	82.73
	shared	87.50	83.24	87.38	82.46	90.01	86.39	89.36	86.50	72.16	82.76	87.91	83.32

Table 4: Results for MTL experiments with SPMRL dataset. All MTL experiments are trained with the alternating batch loss setting to allow for comparison with experiments involving SPMRL. We cannot use joint loss when training SPMRL with either UD or SUD as they are not parallel treebanks. The UD-SUD experiment shows results for UD and SUD when they are trained together in an MTL setting, whereas the SPMRL experiment shows results for UD and SUD when each of them is separately trained along with the corresponding SPMRL dataset instead of each other (UD-SPMRL & SUD-SPMRL).

2020; Zhou et al., 2020).

One specific often overlooked annotation issue that helps convey this point is punctuation.[9] In both annotation schemes, punctuation attachment is rather straight forward. However, as seen in Fig 1, it is one of the competing annotation decisions. In both annotation schemes, punctuation is simply attached directly to the root. However, in UD the root is a content word, while in SUD the root is a function word. Thus, while straight forward form an annotation perspective for both schemes, an MTL system is now learning both attachment possibilities simultaneously and preferences and errors regarding both are now being encoded in the global attachment decisions. When looking at specific attachment errors, across almost all examined experiments, there were substantial increases in punctuation attachment errors. This can be seen as a direct result of switching the content versus function oriented headedness and creates systematic, competing attachment decisions for an MTL parser exposed to both attachment possibilities.

5.2 UD and SUD vs SPMRL

To further explore whether the competing annotation decisions between UD and SUD are indeed contributing to the noticeable performance degradation, we choose to compare a subset of languages in an MTL setup but with a differently annotated treebank. Using a different treebank runs the risk of adding additional domain issues into our experiments; however, character-level embeddings have proven effective at handling OOV words (Ballesteros et al., 2015; Vania et al., 2018), thus domain differences should be reduced.

We perform experiments where we use the Arabic (Habash and Roth, 2009), German (Brants et al., 2004), and Hungarian (Vincze et al., 2010) treebanks from the SPMRL Shared task (Seddah et al., 2013, 2014), each of which were annotated with language specific linguistic phenomena in mind.[10] To mitigate size difference issues as the smaller treebank tends to benefit more in a multi-treebank setup (Johansson, 2013), we randomly select a train and dev set from the SPMRL data respectively to match the corresponding size of the UD-SUD treebanks.[11]

Results for the SPMRL experiments using char+FT embeddings are presented in Table 6.[12] All results in UD-SPRML and SUD-SPMRL MTL experiments show improved performance over the baseline. Importantly, this includes settings in which the UD-SUD MTL experiments show noticeable decreases relative to the baseline, and specifically we see that character-based embeddings are able to yield benefit in an MTL setup relative to the baseline.

These results suggests that the annotation

---

[9] The CoNLL 2018 evaluation scores punctuation. We are not, however, making a claim as to whether punctuation is or is not a linguistic issue, rather simply highlight it is an annotation attachment issue that illustrates different possible attachment distributions between the annotation schemes. From a linguistic perspective, punctuation can be argued to be irrelevant, from a parsing perspective, unless removed, it still influences attachment decisions.

[10] We refer to the reader to the cited papers for more detailed information on the individual treebank annotations.

[11] We note that the both German and Hungarian UD-SUD Treebanks are derived from a small section of the TiGer and Szeged Treebanks respectively, which are also the treebanks for the SPMRL data, thus there is a possibility of sentence overlap in the random selection. The UD-SUD Arabic treebank is derived from the Prague Arabic Dependency Treebank (Hajič et al., 2004) but is also annotated on newswire.

[12] Results using word+POS embeddings are provided in the Appendix.

schemes are indeed contributing to why UD and SUD make poor tasks for each other in an MTL setup, and not strictly the embeddings themselves. Rather, the information conveyed by each individual annotation scheme is important in terms of the possible gains that MTL parsers can make over the baseline parsers. It may simply be that how the annotations are embedded into the architecture and shared are more influential in what signals are encoded in the network than the embeddings themselves in terms of how they benefit treebanks in an MTL setup. If the annotations themselves encode information that results in negative transfer in the network due to their competing nature, an MTL setup cannot benefit as effectively.

6 Conclusion

We implemented an MTL architecture leveraging parsing UD and SUD as separate tasks to examine how their syntactic annotation overlaps and differences influence parser behavior. We find that models from an MTL setup perform generally worse than their single model baselines, regardless of input embeddings. Interestingly, POS embeddings seemingly help mitigate some of the performance loss caused from negative transfer as the POS information may help resolve possible linguistic ambiguities with which character embeddings struggle (Vania et al., 2018; Smith et al., 2018b). This stands in contrast to much multi-treebanking research which has yielded positive performance gains when using multiple treebanks, particularly if they are of the same language, though this is not always the case (Barry et al., 2019).

We then further investigated the possible influence annotations have in an MTL setup by training a subset of SPMRL treebanks against their UD-SUD counterparts, finding increases in performance across the chosen languages and input embeddings not seen when pitting UD and SUD together. We argue that this indicates that in an MTL setup, simply adding another treebank is not inherently going to yield better performance, rather the information that each additional treebank can learn from the other, specifically from their annotation schemes, and how this is then subsequently encoded in the network is a more pivotal factor in yielding performance gains.

We conclude that the syntactic annotation schemes are pertinent when determining performance gains in an MTL parsing setup, as extensive competing annotations provides too many mixed signals in an MTL architecture, hampering the ability of both parsers to benefit from shared information, yielding worse results.

Future research will include incorporating more treebanks with different annotation schemes to examine in which directions and annotations parsers will optimize towards in MTL. We also wish to further explore how constituency parsing and dependency parsing can be leveraged against each other in similar MTL setups.

Acknowledgements

The authors would like to thank Sandra Kübler and members of the Uppsala Parsing Group: Artur Kulmizev, Joakim Nivre, and Sara Stymne for their feedback, as well as the anonymous reviewers for their comments. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute. The second author is supported by the Swedish strategic research programme eSSENCE.

References

Wafia Adouane and Jean-Philippe Bernardy. 2020. When is multi-task learning beneficial for low-resource noisy code-switched user-generated Algerian texts? In *Proceedings of the The 4th Workshop on Computational Approaches to Code Switching*, pages 17–25, Marseille, France. European Language Resources Association.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mark Anderson and Carlos Gómez-Rodríguez. 2020. On the frailty of universal POS tags for neural UD parsers. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 69–96, Online.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal.

James Barry, Joachim Wagner, and Jennifer Foster. 2019. Cross-lingual parsing with polyglot training and multi-treebank learning: A Faroese case study. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 163–174, Hong Kong, China.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 164–169, Valencia, Spain.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24, Melbourne. Association for Computational Linguistics.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans. Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, 2004 (2):597–620.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, Helsinki, Finland.

Matthieu Constant, Joseph Le Roux, and Nadi Tomeh. 2016. Deep lexical segmentation and syntactic parsing in the easy-first dependency framework. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1095–1101, San Diego, California.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5h International Conference on Learning Representations (ICLR 2017)*, Toulon, France. Conference Track Proceedings.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2018. SUD or surface-syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. A universal framework for inductive transfer parsing across multi-typed treebanks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 12–22, Osaka, Japan.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore. Association for Computational Linguistics.

Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117, Cairo, Egypt.

Keith Hall, Ryan McDonald, Jason Katz-Brown, and Michael Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1499, Edinburgh, Scotland, UK.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 127–137, Atlanta, Georgia.

Richard Johansson and Yvonne Adesam. 2020. Training a Swedish constituency parser on six incompatible treebanks. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5219–5224, Marseille, France.

Yash Kankanampati, Joseph Le Roux, Nadi Tomeh, Dima Taji, and Nizar Habash. 2020. Multitask easy-first dependency parsing: Exploiting complementarities of different dependency representations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2497–2508, Barcelona, Spain (Online).

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy.

Ryosuke Kohita, Hiroshi Noji, and Yuji Matsumoto. 2017. Multilingual back-and-forth conversion between content and function head for easy dependency parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 1–7, Valencia, Spain.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy. Association for Computational Linguistics.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997, Brussels, Belgium.

Zuchao Li, Shexia He, Zhuosheng Zhang, and Hai Zhao. 2018. Joint learning of POS and dependencies for multilingual Universal Dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 65–73, Brussels, Belgium.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4487–4496. Association for Computational Linguistics.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 44–53, Valencia, Spain.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Ines Rehbein, Julius Steen, Bich-Ngoc Do, and Anette Frank. 2017. Universal Dependencies are hard to parse – or are they? In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 218–228, Pisa,Italy.

Rudolf Rosa. 2015. Multi-source cross-lingual delexicalized parser transfer: Prague or Stanford? In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 281–290, Uppsala, Sweden.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1599–1613, Minneapolis, Minnesota.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland. Dublin City University.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018a. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018b. An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720, Brussels, Belgium.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235, Berlin, Germany.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia.

Shiva Taslimipoor, Omid Rohanian, and Le An Ha. 2019. Cross-lingual transfer learning and multitask learning for capturing multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 155–161, Florence, Italy.

Clara Vania, Andreas Grivas, and Adam Lopez. 2018. What do character-level models learn about morphology? the case of dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2583, Brussels, Belgium.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1557–1566, Berlin, Germany.

Houquan Zhou, Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Is pos tagging necessary or even helpful for neural dependency parsing? In *Natural Language Processing and Chinese Computing*, pages 179–191, Cham. Springer International Publishing.

Appendix

A Heatmaps for Shared and Unshared MLP layer settings

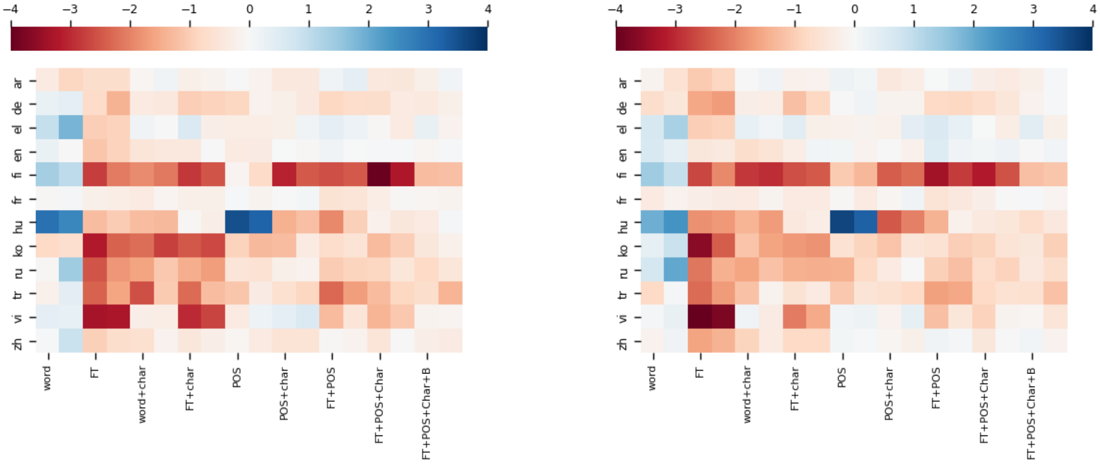

Unshared UAS

Shared UAS

Figure 5: Heatmaps depicting the shared and unshared MLP layers settings. Each block represents the mean UAS score across alternating and joint loss setting for the corresponding embedding and MLP setting.

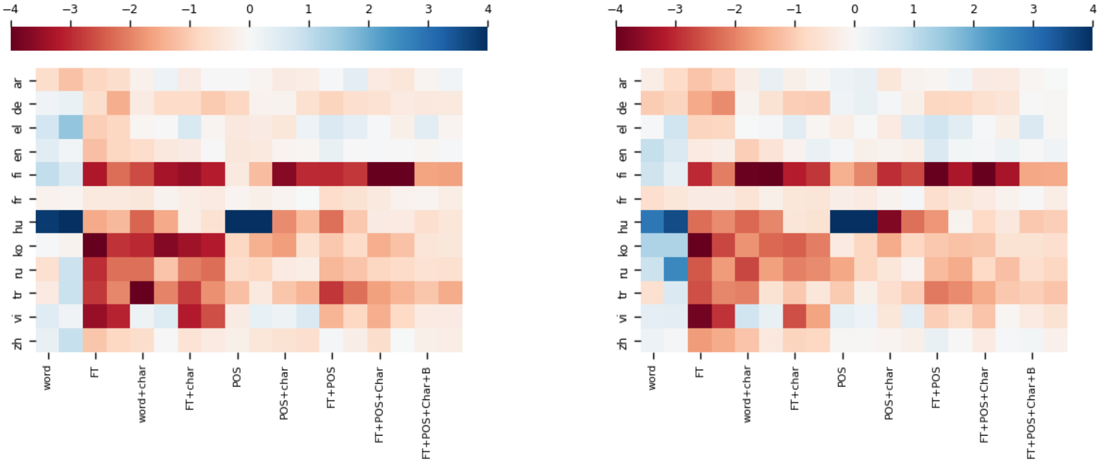

Unshared LAS

Shared LAS

Figure 6: Heatmaps depicting the shared and unshared MLP layers settings. Each block represents the mean LAS score across alternating and joint loss setting for the corresponding embedding and MLP setting.

B Heatmaps for Alternating vs Joint Loss settings

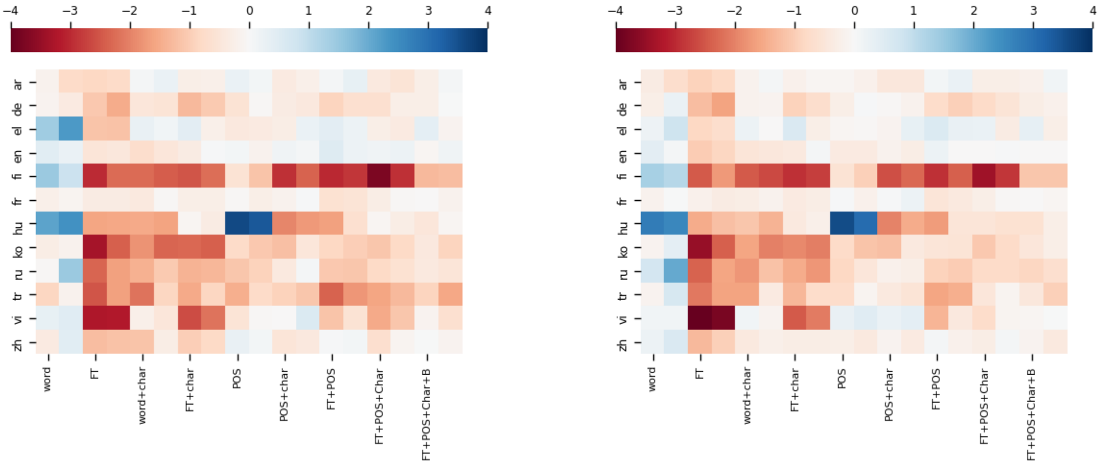

Alternating Loss (UAS) Joint Loss (UAS)

Figure 7: Heatmaps depicting the when joint loss or alternating loss settings. Each block represents the mean UAS score across shared and unshared MLP settings for the corresponding embedding and loss setting.

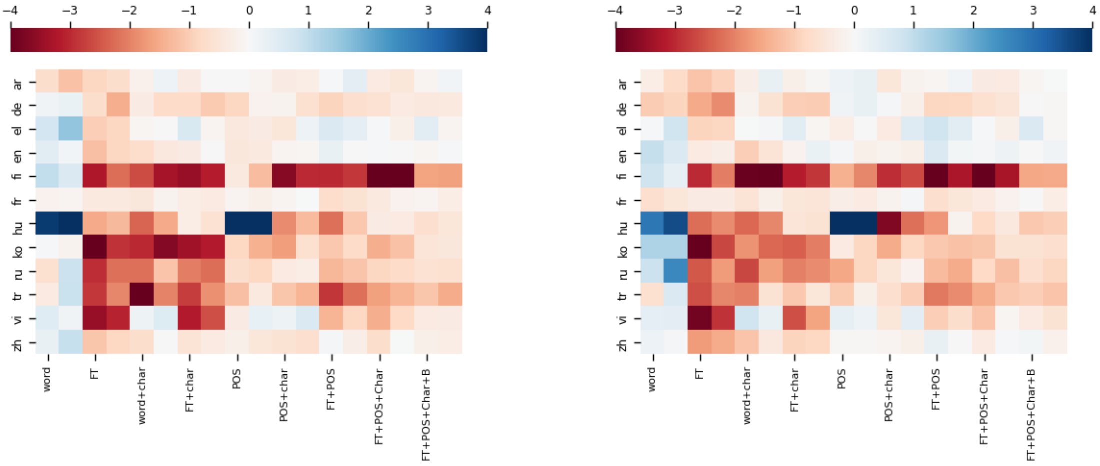

Unshared LAS Shared LAS

Figure 8: Heatmaps depicting the when joint loss or alternating loss settings. Each block represents the mean LAS score across shared and unshared MLP settings for the corresponding embedding and loss setting.

C Training Hyperparameters

Hyperparameters	Value
Embedding Dimensions	300
Character Embedding Dimension	50
POS Tag Embedding Dimension	100
Bert Mapping Dimenstion	100
Number of BERT Layers Used	4
Embed Dropout	0.33
Number of LSTM Layers	400
LSTM Hidden Layer Dimension	400
LSTM Dropout	0.33
MLP (Arc) Output Dimension	500
MLP (Rel) Output Dimension	100
MLP Dropout	0.33
Optimizer	Adam
Patience	50
Batch Size	20000 tokens
Learning Rate	2e-3
Eps	1e-12
Betas	(.9, .9)
Clip	5.0
Decay	0.75
Decay Steps	5000

Table 5: Hyperparameter settings

D UD-SUD vs SPRML MTL Results Table

		Arabic								German								Hungarian							
Exp.	MLP	Word + POS				Char + FastText				Word + POS				Char + FastText				Word + POS				Char + FastText			
		UD		SUD		UD		SUD		UD		SUD		UD		SUD		UD		SUD		UD		SUD	
		UAS	LAS	UAS	LAS	UA	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
	baseline	86.66	83.30	86.87	82.40	86.59	82.25	86.50	81.44	88.50	84.22	86.64	83.63	89.18	85.11	87.81	84.98	65.20	52.26	65.80	54.35	84.73	79.97	84.03	79.44
UD-SUD	unshared	86.79	83.49	86.79	82.39	86.33	81.93	86.34	81.33	87.15	82.84	86.23	83.13	87.95	84.06	86.74	83.70	69.05	57.02	69.16	58.92	84.28	79.42	83.53	78.56
	shared	87.05	83.70	87.17	82.94	86.33	81.97	86.22	81.33	88.68	84.43	86.97	84.03	87.83	84.08	86.76	83.73	68.58	57.16	69.15	59.14	84.50	79.72	85.83	79.05
SPMRL	unshared	87.52	84.13	87.69	83.15	87.78	83.47	87.52	82.62	89.25	85.40	88.93	86.28	90.16	86.44	89.81	87.18	71.86	60.95	73.68	63.37	87.04	82.23	86.99	82.73
	shared	87.62	84.39	87.67	83.15	87.50	83.24	87.38	82.46	89.27	85.34	88.91	86.09	90.01	86.39	89.36	86.50	72.16	61.64	73.42	63.67	87.63	82.76	87.91	83.32

Table 6: Results for MTL experiments with SPMRL dataset. All MTL experiments are trained with the alternating batch loss setting to allow for comparison with experiments involving SPMRL. We cannot use joint loss when training SPMRL with either UD or SUD as they are not parallel treebanks. The UD-SUD experiment shows results for UD and SUD when they are trained together in an MTL setting, whereas the SPMRL experiment shows results for UD and SUD when each of them is separately trained along with the corresponding SPMRL dataset instead of each other (UD-SPMRL & SUD-SPMRL).

Generating Informative Conclusions for Argumentative Texts

Shahbaz Syed † **Khalid Al-Khatib** † **Milad Alshomary** ‡
Henning Wachsmuth ‡ **Martin Potthast** †
†Leipzig University ‡Paderborn University
<shahbaz.syed@uni-leipzig.de>

Abstract

The purpose of an argumentative text is to support a certain conclusion. Yet, they are often omitted, expecting readers to infer them rather. While appropriate when reading an individual text, this rhetorical device limits accessibility when browsing many texts (e.g., on a search engine or on social media). In these scenarios, an explicit conclusion makes for a good candidate summary of an argumentative text. This is especially true if the conclusion is *informative*, emphasizing specific concepts from the text. With this paper we introduce the task of generating informative conclusions: First, Webis-ConcluGen-21 is compiled, a large-scale corpus of 136,996 samples of argumentative texts and their conclusions. Second, two paradigms for conclusion generation are investigated; one extractive, the other abstractive in nature. The latter exploits argumentative knowledge that augment the data via control codes and finetuning the BART model on several subsets of the corpus. Third, insights are provided into the suitability of our corpus for the task, the differences between the two generation paradigms, the trade-off between informativeness and conciseness, and the impact of encoding argumentative knowledge. The corpus, code, and the trained models are publicly available.[1]

1 Introduction

A conclusion of an argument is a statement that conveys a stance towards a specific target (Bar-Haim et al., 2017; Alshomary et al., 2020b). Drawing conclusions is an integral part of argumentation, but often various conclusions may be drawn from a set of premises. Consider the following argumentative text on caffeine adapted from the web:[2]

> *"Caffeine stimulates the nervous system, signaling fat cells to break down body fat. It also increases epinephrine (adrenaline) levels, a fight-or-flight hormone preparing the body for physical exertion. With free body fat acids as fuel, on average, 12% higher performance is attainable."*

Consider further these alternative conclusions:

1. *Caffeine is good.*
2. *Caffeine improves physical performance.*

The first conclusion conveys a pro stance towards the target, caffeine. The second, conveys a pro stance towards caffeine, too, but it also emphasizes a specific concept ("physical performance"). The former conclusion is generic, only *indicating* the stance, while the latter is *informative*; a distinction also made in text summarization (Section 3).[3]

Argumentative texts include short arguments, such as forum posts and reviews, as well as long-form texts, such as essays, blogs, and editorials. Most of these typically have an intended conclusion of which the authors seek to persuade their readers.[4] While the conclusion may be already implied in a given text, authors often choose not to explicitly provide one, either for rhetorical reasons (Habernal and Gurevych, 2015; Al-Khatib et al., 2016), or to encourage critical thinking (Martin et al., 2003). However, when browsing many argumentative texts (e.g., via a search engine or on a social media timeline), having an explicit conclusion helps human readers (and by extension also machines) to quickly process the texts.

In this paper, we introduce the task of generating informative conclusions for argumentative texts, and take the first steps with four key contributions: (1) Adaptation of the notion of informativeness from text summarization as a desired

---

[1] https://github.com/webis-de/ACL-21

[2] https://www.healthline.com/nutrition/top-13-evidence-based-health-benefits-of-coffee

[3] Other works on argumentation use the term *specificity* to express a similar idea (Durmus et al., 2019; Ke et al., 2019).

[4] An exception is an argumentative text dedicated to deliberation, which merely surveys the argument landscape on a given topic without trying to influence the reader's opinion.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3482–3493
August 1–6, 2021.

property of a conclusion besides stating a target and the stance towards it. (2) Compilation of Webis-ConcluGen-21, a corpus of 136,996 pairs of argumentative texts and associated conclusions, creating the first large-scale ground truth for conclusion generation. (3) Modeling conclusion generation as an end-to-end task by finetuning a pretrained sequence-to-sequence model, and augmenting the corpus with three types of argumentative knowledge: topic, target, and aspect. (4) Extensive quantitative and qualitative (crowdsourced) evaluation of both the quality of our dataset and the effectiveness of two paradigms for conclusion generation, namely extractive and abstractive approaches.

We present three key findings: (a) Finetuning pretrained language models on our dataset shows strong in-domain performance compared to the extractive approach. (b) Qualitative evaluation shows that the extractive approach generates more informative conclusions, demonstrating a trade-off between conciseness and informativeness. (c) Encoding argumentative knowledge guides the finetuning towards generating argumentative sentences; however, more sophisticated encoding techniques than just using the conventional control codes are needed to generate informative conclusions.

2 Related Work

Our work complements and builds on that of Alshomary et al. (2020b), who introduced a conceptual model for conclusion generation, outlining a three-step process: inferring the conclusion's target from the argument's premises, inferring the author's stance towards this target, and generating the conclusion based on these two pieces of information. But Alshomary et al. focused only on the first step of target inference, whereas we model conclusion generation as an end-to-end task.

Conclusion generation can be viewed as a complementary task to summarizing argumentative texts. Previous approaches to the summarization of such texts have been primarily extractive. Egan et al. (2016) proposed summarizing online discussions via "point" extraction, where a point is a verb and its syntactic arguments. Similarly, Bar-Haim et al. (2020) compiled the *ArgKP* corpus (which we also sample from in Section 4) comprised of arguments for a given topic mapped to *key points*, composing a summary from a large collection of relevant arguments. Wang and Ling (2016) proposed a data-driven approach using sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) for summarizing movie reviews and debate portal arguments from idebate.org. Several argument mining approaches have also been applied to identify the main claim from arguments (Petasis and Karkaletsis, 2016; Daxenberger et al., 2017). Recently, Alshomary et al. (2020a) proposed a graph-based model using PageRank (Page et al., 1999) that extracts the argument's conclusion and the main supporting reason as an extractive snippet. This model is the core of our extractive summarization approach (Section 5).

A key difference between conclusion generation and general text summarization is the constraint that a conclusion must have a clear stance towards a certain topic. A similar constraint applies to high-quality summaries of long-form argumentative texts such as editorials (Syed et al., 2020), where the persuasiveness of the editorial should be preserved alongside its thesis. Therefore, existing summarization corpora (although large-scale) are unsuitable for studying conclusion generation. A majority of them contain only non-argumentative texts (e.g., news reports) which are more suitable to general-purpose summarization (Kryscinski et al., 2019). Moreover, intrinsic evaluation of summarization corpora has revealed a lower-quality and/or inconsistent ground-truth, rendering them partially unfit for their intended purpose (Bommasani and Cardie, 2020). To fill this gap, we compile Webis-ConcluGen-21, a large-scale corpus of argumentative texts and their conclusions on diverse topics.

Pre-trained language models have significantly advanced the state-of-the-art in neural text summarization (Liu and Lapata, 2019; Zhang et al., 2019a; Rothe et al., 2020; Huang et al., 2020). However, they have been applied to the domain of argumentation only recently, specifically for argument generation. Gretz et al. (2020) proposed a pipeline based on GPT-2 (Radford et al., 2019) for generating coherent claims for a given debate topic. A more controlled approach for argument generation was developed by Schiller et al. (2020), which performs argument generation with fine-grained control of topic, aspect (core reasoning), and stance. Conclusion generation can be viewed as supplementing argument generation. Ideally, given a conclusion, an argument can be generated constrained by the conclusion's target and stance. To the best of our knowledge, studies investigating pretrained language models for end-to-end conclusion generation do not exist. Besides providing a suitable corpus, we analyze the impact of encoding argumentative knowledge in pretrained language models and assess the popular method of control codes (Keskar et al., 2019; Cachola et al., 2020) for encoding the knowledge in our dataset. Furthermore, our qualitative evaluation highlights three key errors (Section 6) arising in the generated outputs that disqualify them as conclusions.

3 On Informative Conclusions

In the literature, the conclusion of an argument is the statement that depicts a particular stance towards a certain concept, the target (Walton et al., 2008; Alshomary et al., 2020b). Such a statement is also referred to as the *claim* of the argument (Toulmin, 2003; Daxenberger et al., 2017). For a long-form argumentative text with multiple claims, the conclusion is the *main claim* that conveys the overall stance towards the subject matter under discussion. The main claim is also known as *thesis*, or *central claim* in different genres (Van Dijk, 1995; Burstein and Marcu, 2003; Stab and Gurevych, 2014; Peldszus and Stede, 2015).

The quality of the conclusion of an argumentative text can be assessed in terms of several dimensions, including strength, clarity, and specificity (Ke et al., 2019). Here, a strong connection between argumentation and text summarization can be observed, where the dimension corresponding to specificity is called *informativeness*. Text summarization distinguishes between indicative and informative summaries. An indicative summary only hints at the principal subject matter of a document to help decide whether to read it (Hovy and Lin, 1998; Kan et al., 2001). An informative summary, on the other hand, covers the main information in the source document, ideally serving as its surrogate (Maybury, 1999).

The conceptual connection between argumentation and summarization could be described as follows: the *informativeness* of a conclusion is closely connected to the specificity dimension, in the sense that an informative conclusion must be specific to allow for a better understanding of an argumentative text's gist. Seeing that "specificity" and "informativeness" may be used interchangeably, we opted for the latter and the term "informative conclusion" here, to underline the connection.

In contrast to *indicative* conclusions, which broadly convey (implicitly or explicitly) the stance towards a topic (e.g., "Caffeine is good."), informative conclusions also discuss specific concepts from (or implied by) the argumentative text (e.g., "Caffeine improves physical performance."). Concepts of the argumentative text exemplified in Section 1 may refer to the topic (e.g., "Is coffee beneficial?"), the target of the conclusion (e.g., "caffeine"), or a specific aspect (e.g.,"energy levels").

4 The Webis-ConcluGen-21 Corpus

This section details the construction of the Webis Conclusion Generation Corpus 2021 (Webis-ConcluGen-21), a corpus of 136,996 pairs of argumentative texts and conclusions covering diverse topics. The corpus is derived from two reliable sources, where the conclusions of argumentative texts are explicitly identifiable: Reddit's ChangeMyView forum and debate corpora.

4.1 Data Source: Reddit's ChangeMyView

ChangeMyView (CMV) is an online forum for persuasive discussions that start with a user who presents a view and asks others to challenge it. The forum's rules strictly enforce that (1) users' posts must contain sufficient reasoning, (2) posts must take a stance (and not be neutral), and (3) the title of a post must sufficiently sum up an author's view (as a statement and not a question).[5] Given these constraints, the original post of a discussion can be operationalized as an *argumentative text*, and the corresponding title as its (intended) *conclusion*. Starting from the Reddit crawls provided by Baumgartner et al. (2020), we compiled 61,695 such pairs by processing all CMV discussions up until August 2019. The included posts are those whose argumentative text was longer than ten words, the conclusion longer than two words, and the title includes the "CMV" tag.[6] An average argumentative text is 312 words long and a conclusion 15 words.

To better understand the relation of the conclusions to their respective argumentative texts, and the expected difficulty of generating them, we analyzed a sample of 200 pairs manually.[7] Table 1

---

[5] https://reddit.com/r/changemyview/wiki/rules

[6] These heuristics reflect manual inspections, and the fact that we did not wish to compile a representative sample of ChangeMyView's discussions, but a purposeful selection of high-quality pairs of argumentative texts and their conclusions: In light of this, the lower bounds are still quite inclusive with respect to extremely short samples.

[7] These examples were taken from the Dec-2019 Reddit submissions to ensure a truly-hidden sample as BART was originally trained on the OpenWebText dataset containing samples

Type	Description	%
Extractive	Conclusion is present verbatim in the argumentative text.	12.8
Paraphrase	Conclusion is synonymous to, or a fusion of a part of the argumentative text.	24.1
Abstractive	Conclusion is inferred from the argumentative text.	57.8
No conclusion	Conclusion cannot be derived from the argumentative text.	5.3

Table 1: Different types of conclusions in 200 CMV samples, and their relative proportion.

shows the proportion of extractive, paraphrased, and abstractive conclusions in our sample, where the former only need to be extracted, and the latter demand actual text synthesis. Paraphrases share aspects of both, though arguably, extracting the paraphrased part would suffice. Altogether, CMV provides for 94.7% valid pairs of argumentative texts and conclusions at sufficiently low noise (5.3%). The amount of non-trivial conclusions (abstractive + paraphrase) are sufficiently challenging, as found in our qualitative evaluation (Section 6).

4.2 Data Source: Debate Corpora

Online debate portals facilitate semi-structured debates on controversial topics, where pro and con arguments or argumentative texts are collected. Conclusions are clearly stated even for individual arguments. Given their high-quality curation, debate portals constitute the majority of argument corpora. We utilized the following existing corpora:

Kialo is a debate platform that enables "visual reasoning" in complex debates via a tree-based structure (Chaudoin et al., 2017). A key advantage here is the role of moderators in curating accepted arguments, rendering it a rich resource (Durmus et al., 2019). As debates progress, the arguments are reorganized into multiple hierarchies, each with a conclusion at its root.[8] We compiled this corpus from scratch in accordance with the website's terms and conditions. In 1,640 English discussions, at each level of the discussion tree, all pro arguments were matched to the corresponding root conclusion, obtaining a total of 82,728 examples.

Args.me is a search engine (Wachsmuth et al., 2017) indexing the Args.me Corpus (Ajjour et al., 2019b), comprised of argumentative texts, their conclusions and their stance from four debate portals: debatewise.org, idebate.org, debatepedia.org, and debate.org. We used the "cleaned" version of this corpus containing 387,606 samples and applied further post-processing. On manual inspection, we observed that a number of examples from debate.org contained spam, sarcasm, or ad hominem attacks, or they were not self-contained due to references to previous turns. To avoid noise, we excluded all examples from this portal. Next, we removed arguments with con stance towards a conclusion.[9] This is due to the fact that considering these examples for training would first require negating their conclusions to reflect the con stance. We leave such automatic claim negation (Bilu et al., 2015) for future work. Finally, to favor informative conclusions, we excluded arguments whose conclusion was the same as the discussion topic (which is generally indicative). This heavy filtering resulted in a total of 23,448 argument-conclusion pairs.

ArgsKP is a corpus of arguments and a set of key points written by domain experts on 28 topics (Bar-Haim et al., 2020). For each topic, the corpus contains multiple arguments which have been mapped via crowdsourcing to their respective key points. From this corpus, we obtained 2,341 pairs; again, only pro arguments and those that have been mapped to a specific key point, the conclusion.

Postprocessing. The structure of debate portals allows for multiple arguments to be mapped to a single conclusion. This happens when different users independently contribute pro and con arguments, which is acceptable, since the same conclusion can be drawn from different arguments with different frames (Ajjour et al., 2019a). Apart from the ones filtered in preprocessing the debates corpora, we preserved duplicate conclusions across debates as their arguments are still unique. Similar to CMV, the included argumentative texts were those whose length exceeded ten words. Also, argumentative texts shorter than their conclusion were excluded. This removed many pairs from the Kialo discussions. Altogether, we retained 75,301 usable examples from all three corpora.

4.3 Corpus Statistics

The argumentative texts are on average longer in CMV (312 words) compared to those in debates (44.5 words). A reason is that, on debate portals, each argumentative text seems to be a self-contained argument. CMV posts, by comparison,

---

from Reddit (Liu et al., 2019; Radford et al., 2019).

[8] For an example, see: https://www.kialo.com/pro-life-vs-pro-choice-should-abortion-be-legal-5637

[9] This does not exclude conclusions that are already negations.

often contain multiple arguments and/or preface the actual argument with additional background. However, the corresponding conclusions are of similar length (15 words for CMV and 18.4 words for debates on average, about the length of an average English sentence). For both data sources, we measured the percentage of words in a conclusion that do not occur in the argumentative text as a measure of "novelty" (Narayan et al., 2018). For CMV, the average novelty is 33.2%, and for debates, the novelty is 81.6%, which is due to the fact that multiple arguments have been mapped to a single conclusion, and that arguments supporting (or attacking) a conclusion during an ongoing discussion are usually not directly derived from it.

5 Generating Informative Conclusions

Given the mixture of conclusion types shown in Table 1, we approach the generation of informative conclusions according to two paradigms, one extractive approach combined with paraphrasing, and one abstractive approach combined with state-of-the-art argument mining technology.

5.1 Paraphrased Conclusion Generation

Paraphrased conclusions are fundamentally extractive in nature, where an extracted sentence is reformulated to improve it. To extract conclusions, we employ the graph-based approach of Alshomary et al. (2020a), originally designed to generate snippets for argument search results. Given an argument, a snippet is generated as follows: (1) related arguments are retrieved as context, (2) all argument's sentences and those from the retrieved ones are embedded, (3) the PageRank of the sentences is computed, and lastly (4) the argument's two top-ranked sentences are returned. Underlying this approach is the hypothesis that an extractive snippet for an argument should comprise its conclusion and its most important supporting premise. Sentences are thus scored regarding their centrality in context of other arguments and their argumentativeness.

Our goal is to generate a single conclusion statement, thus we consider only the top-ranked sentence as the conclusion from the approach of Alshomary et al. (2020a). This sentence is automatically paraphrased using PEGASUS (Zhang et al., 2020a), finetuned on the Google PAWS dataset (Zhang et al., 2019b).[10] For instance, consider the top-ranked sentence from a post questioning the use of hormone blockers on transgender kids:[11]

> *"I don't see it as anything different, and I think it is scandalous to permanently change a child's entire life on a whim rather than treating their mental health."*

After paraphrasing, it reads as follows:

> *"I think it's scandalous to change a child's life on a whim, rather than treating their mental health, and I don't see it as anything different."*

The paraphraser primarily rearranges the sentence; and shared phrases with the original are typical in the paraphrased sentences we reviewed. This approach, called `Arg-PageRank`, represents an advanced extractive paradigm.

5.2 Abstractive Conclusion Generation

Abstractive conclusions can be formulated freely, provided they capture the main pieces of information required for an informative conclusion: topic, targets, stance, and aspects. In this regard, our approach is three-fold (see Figure 1): (1) Automatic extraction of the aforementioned pieces of information from a given argumentative text; (2) augmentation of the training examples in Webis-ConcluGen-21 using control codes, and (3) domain transfer of a pretrained abstractive news summarization model via finetuning on the augmented corpus.

Argumentative Knowledge Extraction. This step details our respective approaches at providing the prerequisite pieces of information to formulate an informative conclusion, namely topic, targets, and aspects. Table 2 shows an example.

Topic: An argumentative text's topic is a description of what it is about. For argumentative texts from debates, we use the associated debate title as the topic. For CMV posts, their titles are also their conclusions; here, topic information is considered missing (denoted as 'NA' token).

Targets: The target of a conclusion is typically a controversial concept or statement (Bar-Haim et al., 2017). For an argumentative text, though, an overlap with its topic is possible, different targets can also be found in its premises. Moreover, when not explicitly stated, the targets of a conclusion can be inferred from either the targets of premises, or external knowledge bases. A set of possible targets

---

[10] https://huggingface.co/tuner007/pegasus_paraphrase

[11] https://www.reddit.com/r/changemyview/comments/e97sir/cmv_giving_children_puberty_blockers_to_allow/

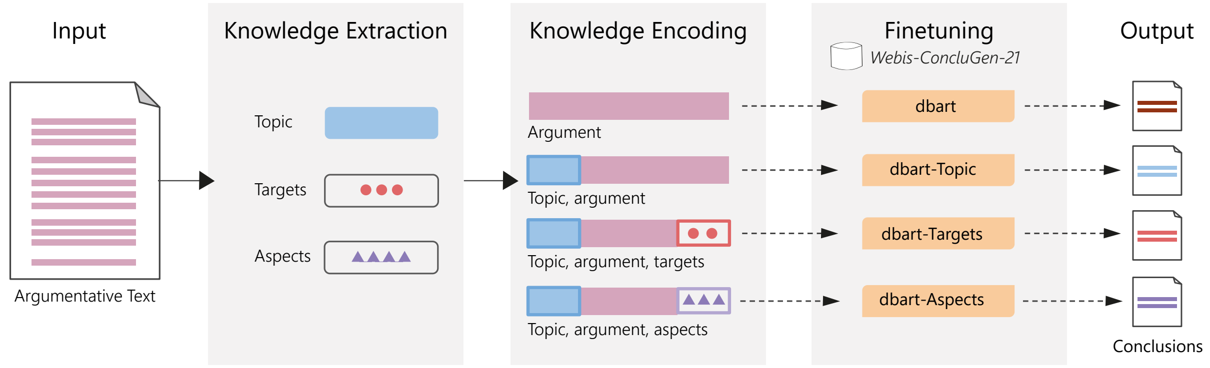

Figure 1: The three steps of our approach to abstractive conclusion generation: For all examples in the Webis-ConcluGen-21 corpus (1) different pieces of argument knowledge are extracted namely the discussion topic, possible conclusion targets, and covered aspects, (2) this knowledge is encoded using control codes, and (3) knowledge-specific variations are finetuned of the distilled BART model to generate informative conclusions.

Argument	Feminism as a 'linguistic term' often misses clarity, universal definition and regularly incorporates opposite goals at the same time in regard to key feminist issues as gender equality, gender-neutrality, non-binary and gender-related rights. The linguistic term thereby clouds public debate and hampers the setting of clear social and political goals in society.
Conclusion	Feminism is an umbrella of ideologies first and foremost, and consequently, it muddies the discussion of gender equality with its ideological baggage.
Topic	Is Feminism a Force For Good?
Aspects	clouds, gender equality, non-binary, opposite goals, public debate, gender-related rights, clarity, gender-neutrality, social and political goals, universal definition
Targets	The linguistic term, Feminism as a ' linguistic term'
Encoded Representation	<|*TOPIC*|>Is Feminism a Force For Good?<|*ARGUMENT*|>Feminism as a 'linguistic term' often misses clarity, universal definition and regularly incorporates opposite goals at the same time in regard to key feminist issues as gender equality, gender-neutrality, non-binary and gender-related rights. The linguistic term thereby clouds public debate and hampers the setting of clear social and political goals in society.<|*TARGETS*|> The linguistic term, Feminism as a ' linguistic term<|*CONCLUSION*|>

Table 2: Example argument-conclusion pair along with topic, targets, and aspects. The last row shows the representation for finetuning models on specific types of encoded external knowledge (here, on conclusion targets).

for every argumentative text in the corpus are automatically identified using the target identification model of Alshomary et al. (2020b).

Aspects: Text spans that contribute to the core reasoning of an argument are called its aspects (Schiller et al., 2020). Aspects can be viewed as subtopics related to the main topic of an argumentative text, encoding a stance. Including aspects into a conclusion can render it more specific and, thus, informative. We identify aspects for all samples in the corpus, using the model of Schiller et al. This model trains a BERT-based (Devlin et al., 2019) ranker on a corpus containing 5,032 high-quality argumentative sentences that are manually labeled with aspects at the token level.

Stance is excluded as an explicit input to our models. For CMV, by design, a post supports its title. For debate portals, only argumentative texts with pro stance towards their conclusion have been considered. Nevertheless, argumentative texts and their conclusions in our corpus may, implicitly or explicitly, express their own stance towards implicit or explicit targets. Implicit stance can be encoded via the aspects.

Argumentative Knowledge Encoding. The extracted pieces of knowledge are encoded into a training example with control codes using special tokens (Cachola et al., 2020): <|*TOPIC*|>, <|*ARGUMENT*|>, <|*ASPECTS*|>, <|*TARGETS*|>, and <|*CONCLUSION*|>. Table 2 shows a corresponding example input sequence encoding the topic and the conclusion targets. To examine the impact of individual knowledge types, we create three versions of Webis-ConcluGen-21: *topic-encoded*, *aspect-encoded*, and *target-encoded*. Presuming the availability of a topic in nearly all real-world applications, it is also encoded in the latter two versions. Since aspects and targets overlap in 38.3% of the case in the corpus, they are independently encoded.

Model	Data	#Train	#Valid
`dbart-XSum`	XSum	204,045	n/a
`dbart-CMV`	CMV	55,768	5,577
`dbart-Debates`	Debates	67,770	6,777
`dbart`	All	123,538	12,354
`dbart-Topic`	All+topic	123,538	12,354
`dbart-Aspects`	All+topic+aspects	122,040	12,192
`dbart-Targets`	All+topic+targets	110,867	11,068
`Arg-PageRank`	*none, unsupervised model*		

Table 3: Corpus splits for all six variants. 'All' refers to the entire Webis-ConcluGen-21 corpus. Models were automatically evaluated on a test set of 1,000 examples, and qualitatively on 300 examples (Section 6).

Parameter	Value
max_target_length	100
warmup_steps	500
eval_steps	500
attention_dropout	0.1
label_smoothing	0.1
sampling	sortish_sampler
seed	5153
num_beams	6
length_penalty	0.5
gradient_accumulation_steps	1
lr_scheduler	linear

Table 4: Hyperparameters for finetuning BART.

Finetuning. As conclusion generation is closely related to abstractive text summarization, we picked BART (Lewis et al., 2020), a pretrained state-of-the-art summarization model, for finetuning on the three augmented versions of Webis-ConcluGen-21. However, BART has approximately 10% more parameters than BERT, which makes it resource-intensive for finetuning. To account for this, we used the distilled checkpoint derived using the "shrink-and-finetune" approach of Shleifer and Rush (2020), where large sequence-to-sequence models are compressed by extracting "distilled student models" (Sanh et al., 2019) from a teacher model (here, BART). We used distilled BART finetuned on the XSum corpus (Narayan et al., 2018) (`dbart-XSum`) provided by the Transformers library (Wolf et al., 2020),[12] since the average length of our ground-truth conclusions is similar to the summaries in XSum. Additionally, we also added our control codes as special tokens to the BART tokenizer during finetuning in order to avoid splitting them into sub-word tokens while processing the encoded sequences.

We first applied `dbart-XSum` on the held-out test set of 200 examples analyzed for Table 1 to evaluate the domain transfer from news reports to argumentative texts. On manual evaluation, 79.1% of the outputs were invalid conclusions, primarily due to being non-argumentative (Section 6). This demonstrates that existing summarization models are ineffective when applied on argumentative texts and must be trained on task-specific data.

Model	BERTScore (F1)	Rou.-1	Rou.-2	Rou.-L
`dbart-XSum`	0.21	15.28	3.10	13.31
`dbart-CMV`	0.32	20.35	7.11	18.80
`dbart-Debates`	0.23	15.38	4.85	14.22
`dbart`	**0.39**	**31.73**	**19.48**	**30.87**
`dbart-Topic`	0.34	23.74	9.56	22.14
`dbart-Aspects`	0.33	23.47	9.46	22.01
`dbart-Targets`	0.34	23.80	9.63	22.25
`Arg-PageRank`	0.20	15.35	3.20	13.37

Table 5: Automatic evaluation of models on the internal test set consisting of 1,000 pairs (500 each from CMV and Debates). BERTScore is the re-scaled F1 score; in addition, average Rouge-1, -2, and -L are reported.

5.3 Training Details

We compiled six variations of the corpus (with and without encoded knowledge) for finetuning the The `dbart-XSum` model with 306M parameters.[12] Table 3 shows the training and validation splits for each model variant and the corresponding data subsets, and Table 4 shows the chosen hyperparameters. The standard finetuning regimen was employed from the Transformers library[13] to train each model on a V100 GPU for 6 epochs with batch size 1, dropout rate 0.1, adafactor optimizer, learning rate of 3e-5, and beam search for inference. For `dbart-<CMV|Debates|All>` the maximum source sequence length was set to 512 tokens, while for `dbart-<Topic|Aspects|Targets>` we increased it to 750 tokens to account for the appended knowledge in the input sequence. On a single V100 GPU, the runtime varies between 3 to 5 days per model, depending on their corresponding training splits.

6 Evaluation

Our models are evaluated via both: (1) An automatic evaluation on a large test set using standard metrics, and (2) a manual evaluation on a smaller test set via crowdsourcing.

6.1 Automatic Evaluation

On a test set of 1,000 examples with known ground-truth (500 each from CMV and from the debate corpora), we computed ROUGE (Lin, 2004)[14] and BERTScore (Zhang et al., 2020b)[15] for all models.

---

[12] https://huggingface.co/sshleifer/distilbart-xsum-12-6

[13] https://github.com/huggingface/transformers/tree/master/examples/legacy/seq2seq

[14] https://github.com/pltrdy/rouge

[15] https://github.com/Tiiiger/bert_score

Table 5 shows that `dbart-XSum` performs poorly on argumentative texts. Inspecting the reasons for this shortcoming, we found several outputs of the model to be either neutral sentences (despite having the right target), or hallucinations with artifacts from the XSum corpus (e.g., "*In our series of letters from African journalists* [...]" or "*This week I've been writing about* [...]"). Among the finetuned models, `dbart`, trained on the entire corpus without any encoded knowledge, performs best across all metrics. The knowledge-encoded models exert a drop in effectiveness, but still outperform models trained on the sub-datasets `dbart-CMV` and `dbart-Debates`.

All finetuned models generate concise outputs of similar lengths (average 12 words), while `Arg-PageRank` extracts longer spans (25 words). Outputs of the knowledge-encoded models are somewhat similar to each other (average pairwise Jaccard similarity of 0.43), compared to those from `dbart` (0.27 with any knowledge-encoded model).

6.2 Manual Evaluation

Given the results of the automatic evaluation, only the models trained on the entire corpus were manually evaluated against our baseline approach `Arg-PageRank`. A test set of 300 examples was employed, 100 each from debates and CMV posts, plus 100 comments to CMV posts. The latter include only comments with at least 100 words and exclude non-argumentative ones as per automatic claim-detection (Chakrabarty et al., 2019). This part of the test set corresponds to an unsupervised evaluation of the conclusions, since no ground truth for the comments is available.

Two expert writers, both native English speakers, were hired via Upwork.com.[16] For every given argumentative text in the test set, all candidate conclusions generated by the different models were shown to the annotators in random order, and without revealing the respective model's name. Assessment was cast as a series of binary decisions: first, whether a given candidate is a conclusion, and if yes, whether it is fluent, and whether it is informative. To simplify judging informativeness, we only asked if the conclusion was too generic. For each candidate judged not to be a conclusion, we asked whether it either has the (1) *wrong target* (WT), conveys the (2) *wrong stance* (WS), or whether it is (3) *non-argumentative* (NA).

Table 6 shows the percentage of cases on which both annotators agreed. For CMV and debates, finetuning outperforms `Arg-PageRank` at generating conclusions that convince the experts: `dbart` performs best on CMV (36%), and `dbart` and `dbart-Topic` on debates (14%).

Model	**Concl.**	**Inform.**	**Error Types**		
			WT	WS	NA
CMV Posts					
`dbart`	36%	4%	56%	22%	22%
`dbart-Topic`	28%	0%	59%	23%	18%
`dbart-Aspects`	33%	6%	69%	23%	8%
`dbart-Targets`	27%	4%	69%	23%	8%
`Arg-PageRank`	11%	7%	0%	0%	100%
Debates					
`dbart`	14%	6%	65%	9%	26%
`dbart-Topic`	14%	3%	76%	12%	12%
`dbart-Aspects`	7%	2%	77%	13%	10%
`dbart-Targets`	11%	2%	71%	17%	12%
`Arg-PageRank`	10%	6%	7%	0%	93%
Comments					
`dbart`	12%	2%	52%	18%	30%
`dbart-Topic`	6%	2%	58%	24%	18%
`dbart-Aspects`	7%	3%	52%	33%	15%
`dbart-Targets`	8%	3%	55%	35%	10%
`Arg-PageRank`	17%	9%	5%	5%	90%

Table 6: Full agreement percentages of two annotators on 300 examples, grouped by the example type (posts, debates, comments). The first column is the % of valid conclusions, the second the % of informative conclusions, followed by the % distribution of error types (lower is better) of a model. On average, all models were judged to be fluent for 97% of the conclusions.

Comments appear to be a particularly difficult type of test cases. This is because comments to the first post may not be self-contained but refer back to the post, they may have a mixed stance (supporting only part of the post while opposing the rest), and they may introduce new targets and aspects (different concepts)—based on our inspection of the comments. In such cases, extracting the conclusion from the comment (and paraphrasing it) using `Arg-PageRank` performs best (17%).

Encoding knowledge slightly impacts the effectiveness. Across all example types, knowledge-encoded models perform equally well, sometimes worse, sometimes better than `dbart`. Encoding topic with aspects or targets performs better on posts and comments.

As for *informativeness*, `dbart-Aspects` generates a higher number of informative conclusions for posts, while `dbart` does best in debates, among the finetuned models. In all domains, `Arg-PageRank` performs similar to or better than all approaches due to extracting claims that are twice as long on average (24 words) compared to the finetuned models (12 words), hence capturing more information.

---

[16] An hourly rate of about 30 USD was paid.

Inspecting the error types, encoding argumentative knowledge increases the number of argumentative candidate conclusions, validating its positive impact. All knowledge-encoded models have fewer non-argumentative (NA) errors compared to `dbart`. However, this affects target inference; the knowledge-encoded models generate more wrong targets (WT). The mixed stance of comments (supporting part of the original post, while opposing the rest) leads to a higher number of stance errors (WS) for `dbart-Aspects` and `dbart-Targets`. Finally, for `Arg-PageRank`, almost all errors were non-argumentative sentences (NA).

6.3 Discussion

Our qualitative evaluation indicates that generating informative conclusions is challenging, and that our data is well-suited for the task, due to a mix of conclusion types (Table 1), and diverse data sources. Leveraging external knowledge, though a promising feature for guiding finetuning, may benefit from better encoding strategies compared to the conventional method of using control codes in text. However, given that the identified knowledge is extractive and that we encoded multiple aspects and targets per example in contrast to related controlled text generation approaches (Keskar et al., 2019; Schiller et al., 2020; Gretz et al., 2020; Cachola et al., 2020), further investigations with importance sampling of argumentative knowledge are advised. Ideally, such sampling would be tailored to a specific domain or target audience.

Likewise, regarding the informativeness of the generated conclusions, a trade-off between conciseness and specificity must be decided. Our experiments suggest that long extractive conclusions capture more information compared to the more concise (and fluent) abstractive one of the finetuned models, rendering them preferable to the annotators when sufficient background is missing. Finally, for comments, modeling the argumentative context supplemented by explicit stance identification is necessary to generate valid conclusions.

7 Conclusion

The notion of an informative conclusion is introduced and discussed in the context of computational argumentation as well as text summarization. Informative conclusions are to argumentation what brief summaries are to text: they concisely convey its main points. We lay the foundation for studying the conclusions of argumentative texts, compiling the Webis-ConcluGen-21 corpus, comprising 136,996 pairs of argumentative texts and corresponding conclusions.

Conclusions are diverse and typically depart significantly from the argumentative text they are derived from, paraphrasing it, and more than half the time abstracting over it. Authors typically tailor their conclusions to the occasion; and in many cases, they are not necessarily made explicit. This is where we contribute by tackling the task of generating an informative conclusion. The two main paradigms we study—paraphrased (incl. extractive) vs. abstractive conclusion generation—compete closely with each other.

8 Ethics Statement

Our dataset is a collection of opinionated texts obtained from sources that are available publicly and acknowledged appropriately. We respected their terms and conditions.

We did not employ any author-specific features in our approaches and instead processed only the corresponding arguments, although representing personal views of anonymous authors.

The proposed technology will be applicable to an English-speaking audience. While failures in generating valid conclusions may mislead a reader's initial interpretation of an argument, we do not aim at applications that prevent readers from reading the complete arguments. Rather, we seek to simplify the consumption of public discussions comprising several arguments by providing explicit, informative conclusions especially for longer arguments.

Finally, in terms of computational resources, we restricted ourselves to the smaller, distilled checkpoints of a large pretrained model that can be trained with (comparably) smaller resources and are accessible to majority of the researchers.

Acknowledgments

We thank the reviewers for their valuable feedback. This work was supported by the German Federal Ministry of Education and Research (BMBF, 01/S18026A-F) by funding the competence center for Big Data and AI (ScaDS.AI Dresden/Leipzig). Computations for this work were done (in part) using resources of the Leipzig University Computing Centre, who we sincerely thank for their support.

References

Yamen Ajjour, Milad Alshomary, Henning Wachsmuth, and Benno Stein. 2019a. Modeling frames in argumentation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2922–2932, Hong Kong, China. Association for Computational Linguistics.

Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. 2019b. Data acquisition for argument search: The args.me corpus. In *KI 2019: Advances in Artificial Intelligence - 42nd German Conference on AI, Kassel, Germany, September 23-26, 2019, Proceedings*, pages 48–59.

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443, Osaka, Japan. The COLING 2016 Organizing Committee.

Milad Alshomary, Nick Düsterhus, and Henning Wachsmuth. 2020a. Extractive snippet generation for arguments. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1969–1972. ACM.

Milad Alshomary, Shahbaz Syed, Martin Potthast, and Henning Wachsmuth. 2020b. Target inference in argument conclusion generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4345, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, Valencia, Spain. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4029–4039. Association for Computational Linguistics.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media, ICWSM 2020, Held Virtually, Original Venue: Atlanta, Georgia, USA, June 8-11, 2020*, pages 830–839. AAAI Press.

Yonatan Bilu, Daniel Hershcovich, and Noam Slonim. 2015. Automatic claim negation: Why, how and when. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 84–93, Denver, CO. Association for Computational Linguistics.

Rishi Bommasani and Claire Cardie. 2020. Intrinsic evaluation of summarization datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8075–8096, Online. Association for Computational Linguistics.

Jill Burstein and Daniel Marcu. 2003. A machine learning approach for identification of thesis and conclusion statements in student essays. *Computers and the Humanities*, 37(4):455–467.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S. Weld. 2020. TLDR: extreme summarization of scientific documents. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 4766–4777. Association for Computational Linguistics.

Tuhin Chakrabarty, Christopher Hidey, and Kathy McKeown. 2019. IMHO fine-tuning improves claim detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 558–563, Minneapolis, Minnesota. Association for Computational Linguistics.

Stephen Chaudoin, J Shapiro, and Dustin Tingley. 2017. Revolutionizing teaching and research with a structured debate platform1. *Journal of Political Science*, 58:1064–1082.

Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Esin Durmus, Faisal Ladhak, and Claire Cardie. 2019. Determining relative argument specificity and stance for complex argumentative structures. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4630–4641, Florence, Italy. Association for Computational Linguistics.

Charlie Egan, Advaith Siddharthan, and Adam Wyner. 2016. Summarising the points made in online political debates. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 134–143, Berlin, Germany. Association for Computational Linguistics.

Shai Gretz, Yonatan Bilu, Edo Cohen-Karlik, and Noam Slonim. 2020. The workweek is the best time to start a family - A study of GPT-2 based claim generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 528–544. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137, Lisbon, Portugal. Association for Computational Linguistics.

Eduard Hovy and Chin-Yew Lin. 1998. Automated text summarization and the Summarist system. In *TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998*, pages 197–214, Baltimore, Maryland, USA. Association for Computational Linguistics.

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 446–469, Online. Association for Computational Linguistics.

Min-Yen Kan, Kathleen R. McKeown, and Judith L. Klavans. 2001. Applying natural language generation to indicative summarization. In *Proceedings of the ACL 2001 Eighth European Workshop on Natural Language Generation (EWNLG)*.

Zixuan Ke, Hrishikesh Inamdar, Hui Lin, and Vincent Ng. 2019. Give me more feedback II: Annotating thesis strength and related attributes in student essays. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3994–4004, Florence, Italy. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 540–551. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Brett AS Martin, Bodo Lang, Stephanie Wong, and Brett AS Martin. 2003. Conclusion explicitness in advertising: The moderating role of need for cognition (nfc) and argument quality (aq) on persuasion. *Journal of Advertising*, 32(4):57–66.

Mani Maybury. 1999. *Advances in automatic text summarization*. MIT press.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948. Association for Computational Linguistics.

Georgios Petasis and Vangelis Karkaletsis. 2016. Identifying argument components through textrank. In *Proceedings of the Third Workshop on Argument Mining, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL 2016, August 12, Berlin, Germany*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Trans. Assoc. Comput. Linguistics*, 8:264–280.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Aspect-controlled neural argument generation. *CoRR*, abs/2005.00084.

Sam Shleifer and Alexander M Rush. 2020. Pre-trained summarization distillation. *arXiv preprint arXiv:2010.13002*.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Shahbaz Syed, Roxanne El Baff, Johannes Kiesel, Khalid Al Khatib, Benno Stein, and Martin Potthast. 2020. News editorials: Towards summarizing long argumentative texts. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5384–5396, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Stephen E Toulmin. 2003. *The uses of argument*. Cambridge university press.

Teun A Van Dijk. 1995. Opinions and ideologies in editorials. In *4th International Symposium of Critical Discourse Analysis, Language, Social Life and Critical Thought, Athens*, pages 14–16.

Henning Wachsmuth, Martin Potthast, Khalid Al-Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, Copenhagen, Denmark. Association for Computational Linguistics.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019a. Pretraining-based natural language generation for text summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 789–797, Hong Kong, China. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019b. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Substructure Substitution: Structured Data Augmentation for NLP

Haoyue Shi **Karen Livescu** **Kevin Gimpel**
Toyota Technological Institute at Chicago, IL, USA, 60637
{freda,klivescu,kgimpel}@ttic.edu

Abstract

We study a family of data augmentation methods, substructure substitution (SUB^2), that generalizes prior methods. SUB^2 generates new examples by substituting substructures (e.g., subtrees or subsequences) with others having the same label. This idea can be applied to many structured NLP tasks such as part-of-speech tagging and parsing. For more general tasks (e.g., text classification) which do not have explicitly annotated substructures, we present variations of SUB^2 based on text spans or parse trees, introducing structure-aware data augmentation methods to general NLP tasks. For most cases, training with a dataset augmented by SUB^2 achieves better performance than training with the original training set. Further experiments show that SUB^2 has more consistent performance than other investigated augmentation methods, across different tasks and sizes of the seed dataset.[1]

1 Introduction

Data augmentation has been found effective for various natural language processing (NLP) tasks, such as machine translation (Fadaee et al., 2017; Gao et al., 2019; Xia et al., 2019, *inter alia*), text classification (Wei and Zou, 2019; Quteineh et al., 2020), syntactic and semantic parsing (Jia and Liang, 2016; Shi et al., 2020; Dehouck and Gómez-Rodríguez, 2020), semantic role labeling (Fürstenau and Lapata, 2009), and dialogue understanding (Hou et al., 2018; Niu and Bansal, 2019). Such methods enhance the diversity of the training set by generating examples based on existing ones, and can make the learned models more robust against noise (Xie et al., 2020). Most existing work focuses on word-level manipulation (Kobayashi, 2018; Wei and Zou, 2019; Dai and Adel, 2020, *inter alia*) or global sequence-to-sequence style generation (Sennrich et al., 2016).

In this work, we study a family of general data augmentation methods, substructure substitution (SUB^2), which generates new examples by substituting same-label substructures (Figure 1). While some instances within this family have been proposed before for certain tasks, we generalize the idea and investigate it for a variety of tasks and settings. SUB^2 naturally fits structured prediction tasks such as part-of-speech tagging and parsing, where substructures exist in the annotations of the tasks. For more general NLP tasks such as text classification, we present variations of SUB^2 which (1) define substructures based on text spans or parse trees for existing examples, and (2) generate new examples by substructure substitution based on the substructures and various kinds of constraints.

While data augmentation methods can often be task-specific or have inconsistent performance, extensive experiments show that SUB^2 consistently helps models achieve competitive or better performance than training on the original dataset across structured prediction tasks and original dataset sizes. We further study the effect of different constraints for the variations of SUB^2 in text classification. While there is no consistently winning combination of constraints, SUB^2 remains dominant on both investigated few-shot text classification datasets.

In addition, when combined with XLM-R (Conneau et al., 2019), a cross-lingual pretrained language model, SUB^2 establishes new state-of-the-art results for sentiment analysis and low-resource part-of-speech tagging. Finally, the experimental setups we define can serve as a benchmark for future work on NLP with little annotated data.

---

[1] Project page: `https://home.ttic.edu/~freda/project/sub2`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3494–3508
August 1–6, 2021.

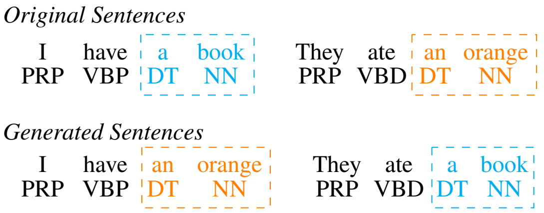

(a) Part-of-speech tagging.

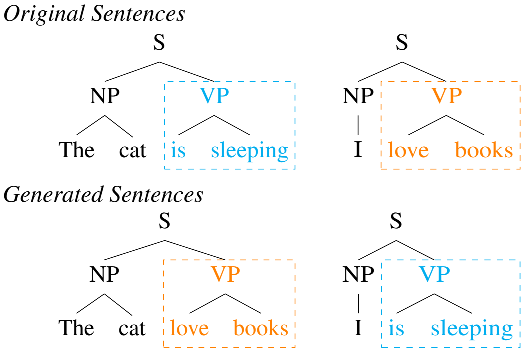

(b) Constituency parsing.

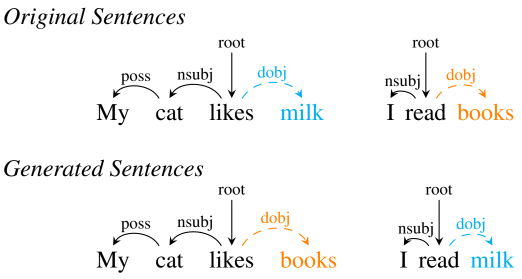

(c) Dependency parsing.

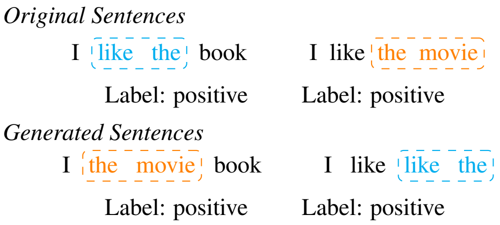

(d) Text classification. To apply SUB^2, we use text spans as substructures, with both the *number of words* in the span and the *text classification label* as constraints (see Sec. 3.2).

Figure 1: Illustration of SUB^2 for investigated tasks. We generate new examples by same-label substructure substitution, whether or not the generated examples are semantically or syntactically acceptable. Best viewed in color.

2 Related Work

Data augmentation aims to generate new examples based on available ones, without actually collecting new data. Such methods reduce the cost of dataset collection, and usually boost model performance on desired tasks. Most existing data augmentation methods for NLP tasks can be classified into the following categories:

Token-level manipulation. Token-level manipulation methods have been widely studied in recent years. They typically create new examples by substituting (word) tokens with ones having the same desired features, such as synonym substitution (Zhang et al., 2015; Wang and Yang, 2015; Fadaee et al., 2017; Kobayashi, 2018) or substitution with words having the same morphological features (Silfverberg et al., 2017). Such methods have been applied to generate adversarial or negative examples which help improve the robustness of neural network–based NLP models (Belinkov and Bisk, 2018; Shi et al., 2018a; Alzantot et al., 2018; Zhang et al., 2019; Min et al., 2020, *inter alia*), or to generate counterfactual examples which help mitigate bias in natural language (Zmigrod et al., 2019; Lu et al., 2020).

Other token-level manipulation methods introduce noise, such as random token shuffling and deletion (Wang et al., 2018; Wei and Zou, 2019; Dai and Adel, 2020). Models trained on the augmented datasets are expected to be more robust against the considered noise.

Constrained text generation. Recent work has explored generating new examples by training a conditional text generation model (Bergmanis et al., 2017; Liu et al., 2020a; Ding et al., 2020; Liu et al., 2020b, *inter alia*), or applying post-processing on the examples generated by pretrained models (Yang et al., 2020; Wan et al., 2020; Yoo et al., 2020). In the data augmentation stage, given task-specific constraints, such models generate associated text accordingly. The generated examples, together with the original datasets, are used to further train models for the primary tasks. A representative method is back-translation (Sennrich et al., 2016), which is effective for not only machine translation, but also style transfer (Prabhumoye et al., 2018; Zhang et al., 2020a), conditional text generation (Sobrevilla Cabezudo et al., 2019), text classification (Iyyer et al., 2018), and grammatical error correction (Xie et al., 2018). Relatedly, automatic question generation has been used in data augmentation for question answering (Yang et al., 2017; Song et al., 2018).

Another approach to example generation is to generate new examples based on predefined templates (Kafle et al., 2017; Asai and Hajishirzi, 2020), where the templates are designed following heuristic, and usually task-specific, rules.

Soft data augmentation. As an alternative to explicit generation of concrete examples, soft augmentation directly represents generated examples in a continuous vector space: Gao et al. (2019) propose to perform soft word substitution for machine translation; recent work has adapted the mixup method (Zhang et al., 2018), which augments the original dataset by linearly interpolating the vector representations of text and labels, to text classification (Guo et al., 2019; Sun et al., 2020), named entity recognition (Chen et al., 2020), and compositional generalization (Guo et al., 2020).

Structure-aware data augmentation. Existing work has also sought potential gain from structures associated with natural language: Xu et al. (2016) improve word relation classification by dependency path–based augmentation. Şahin and Steedman (2018) show that subtree cropping and rotation based on dependency parse trees can help part-of-speech tagging for low-resource languages, while Vania et al. (2019) demonstrate that such methods also help dependency parsing when very limited training data is available.

SUB^2 also falls into this category. The idea of same-label substructure substitution has been used to improve performance on structured prediction tasks such as semantic parsing (Jia and Liang, 2016), constituency parsing (Shi et al., 2020), dependency parsing (Dehouck and Gómez-Rodríguez, 2020), named entity recognition (Dai and Adel, 2020), meaning representation–based text generation (Kedzie and McKeown, 2020), and compositional generalization (Andreas, 2020). To the best of our knowledge, however, SUB^2 has not been systematically studied as a general data augmentation method for NLP tasks. In this work, we not only extend SUB^2 to part-of-speech tagging and structured sentiment classification, but also present a variation that allows a broader range of NLP tasks (e.g., text classification) to benefit from syntactic parse trees. We evaluate SUB^2 and several representative general data augmentation methods, which can be widely applied to various NLP tasks.

When constituency parse trees are used, there is a connection between SUB^2 and tree substitution grammars (TSGs; Schabes, 1990), where the approach can be viewed as (1) estimating a TSG using the given corpus and (2) drawing new sentences from the estimated TSG.

3 Method

We introduce the general framework we investigate in Section 3.1, and describe the variations of SUB^2 which can be applied to text classification and other NLP applications in Section 3.2.

3.1 Substructure Substitution (SUB^2)

As shown in Figure 1, given the original training set $\mathcal{D}$, SUB^2 generates new examples using same-label substructure substitution, and repeats the process until the training set reaches the desired size. The general SUB^2 procedure is presented in Algorithm 1.

Algorithm 1: SUB^2.

```
Input: Original dataset $\mathcal{D}$,
desired dataset size $N > |\mathcal{D}|$
Output: Augmented dataset $\mathcal{D}'$
$\mathcal{D}' \leftarrow \mathcal{D}$;
repeat
    Uniformly draw $s \in \mathit{substructure}(\mathcal{D}')$
    $S \leftarrow \mathit{example}(s)$
    Uniformly draw $u \in \{v \mid v \in \mathit{substructure}(\mathcal{D}), \mathit{label}(v) = \mathit{label}(s), v \neq s\}$
    $S' \leftarrow$ replace $s$ with $v$ in $S$
    $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{S'\}$
until $|\mathcal{D}'| = N$;
```

For part-of-speech (POS) tagging, we let text spans be substructures and use the corresponding POS tag sequences as substructure labels (Figure 1a); for constituency parsing, we use subtrees as the substructures, with constituent labels as the substructure labels (Figure 1b); for dependency parsing, we also use subtrees as substructures, and let the dependency arc labels, which link the heads of subtrees to their parents, be the substructure labels (Figure 1c).

3.2 Variations of SUB^2 for Text Classification

Text classification examples do not typically contain explicit substructures. However, we can obtain them by viewing all text spans as substructures (Figure 1d). This approach may be too unconstrained in practice, so we consider constraining substitution based on matching several features of the spans:

- **Text classification label**: when considering this constraint, we can only substitute a span with another span that comes from text annotated with the same class label as the original one; otherwise we can choose the alternative from any example text in the training corpus.
- **Constituency**: when considering this constraint, we can only substitute a constituent with another constituent (according to a constituency parse of the text, whether they have the same constituent label or not); otherwise the considered spans do not necessarily need to be constituents.
- **Annotated text span label**: in our experiments, this constraint is valid only when the previous constraint (constituency) is considered. When considering this constraint, we can only perform substitution between text spans with the same annotated label (e.g., constituent label).[2]
- **Number of words**: when considering this constraint, we can only substitute a span with another having the same number of words; otherwise we can substitute a span with any other span.

Shi et al. (2018b) argue that binary balanced trees are better backbones for recursive neural networks (Zhu et al., 2015; Tai et al., 2015) on text classification; inspired by them, we introduce the following constraint in this work:

- **"Constituency" in binary balanced tree.** we use binary balanced trees, analogously to constituency parse trees, as the backbone for SUB^2: we (1) generate balanced trees by recursively splitting a span of n words into two consecutive groups, which consist of $\lfloor \frac{n}{2} \rfloor$ and $\lceil \frac{n}{2} \rceil$ words respectively, and (2) treat each nonterminal in the balanced tree as a substructure to perform SUB^2.

We also investigate combinations of the above constraints, where we require all the chosen constraints to be the same to perform SUB^2. For example, combining *text classification label* and *number of words* (Figure 1d) requires the original and the alternative span to have the same text label and the same number of words.

4 Experiments

We introduce our experimental setups (Section 4.1), and evaluate SUB^2 and several data augmentation baselines (Section 4.2) on four tasks: part-of-speech tagging (Section 4.3), dependency parsing (Section 4.4), constituency parsing (Section 4.5), and text classification (Section 4.6).

4.1 Setup

For part-of-speech tagging and text classification, we add a two-layer perceptron on top of XLM-R (Conneau et al., 2019) embeddings, where we calculate contextualized token embeddings by a learnable weighted average across layers. We use endpoint concatenation (i.e., the concatenation of the first and last token representation) to obtain fixed-dimensional span or sentence features, and keep the pretrained model frozen during training.[3] For dependency parsing, we use the SuPar implementation of Dozat and Manning (2017).[4] For constituency parsing, we use Benepar (Kitaev and Klein, 2018).[5]

For all data augmentation methods, including the baselines (Section 4.2), we only augment the training set, and use the original development set. If not specified, we introduce 20 times more examples than the original training set when applying an augmentation method. When introducing $k\times$ new examples, we also replicate the original training set k times to ensure that the model can access sufficient examples from the original distribution.

All models are initialized with the XLM-R base model (Conneau et al., 2019) if not specified. We train models for 20 epochs in high-resource settings (i.e., high-resource part-of-speech tagging, sentiment classification trained on the full training set) or when applying data augmentation methods, and for 400 epochs in the low-resource settings without augmentation; we select the one with the highest accuracy or F_1 score on the development set. All models are optimized using Adam (Kingma and Ba, 2015), with learning rates cho-

---

[2] There can be other text span labels such as sentiment labels of constituents (Socher et al., 2013).

[3] We did not observe significant improvement by finetuning the large pretrained language model, and for most cases, the performance is much worse than the current scheme we apply.

[4] `https://github.com/yzhangcs/parser`

[5] `https://github.com/nikitakit/self-attentive-parser`

sen from $\{5 \times 10^{-4}, 5 \times 10^{-5}\}$. For the hidden layer size (i.e., the hidden size of the perceptron for part-of-speech tagging and text classification, the dimensionality of span representation and scoring multi-layer perceptron for constituency parsing, and the dimensionality of token representation and scoring multi-layer perceptron for dependency parsing), we vary it between 128 and 512. We apply a 0.2 dropout ratio to the contextualized embeddings in the training stage. All other hyperparameters are the same as the default settings in the released codebases.

4.2 Baselines

We compare Sub^2 to the following baselines:

- **No augmentation** (NoAug), where the original training and development set are used.

- **Contextualized substitution** (CtxSub), where we apply contextualized augmentation (Kobayashi, 2018), masking out a random word token from the existing dataset and using multilingual BERT (mBERT; Devlin et al., 2019) to generate a different word.

- **Knowledge based guided synonym substitution** (Syno), where we substitute a random word token by its synonym defined in an existing knowledge base.[6]

- **Random shuffle** (Shuf), where we randomly shuffle all the words in the original sentence, while keeping the original structured or non-structured labels. It is worth noting that for dependency parsing, we shuffle the words, while maintaining the dependency arcs between individual word tokens; for constituency parsing, we shuffle the terminal nodes, and insert them back into the tree structure. Our Shuf method for constituency parsing is arguably more noisy than that for dependency parsing.

All of the data augmentation baselines are explicit augmentations where concrete new examples are generated and used. The methods above are generally applicable to a wide range of NLP tasks.

Lang. Aug.	SoTA NoAug	mBERT NoAug	XLM-R NoAug	XLM-R Sub^2
high-resource languages				
avg.	96.9	97.1	**97.7**†	**97.7**†
bg	98.7	98.9	**99.4**	**99.4**
cs	99.0	99.0	**99.2**	**99.2**
da	97.2	97.8	**98.7**	98.5
de	94.4	94.6	**95.3**	95.1
en	96.1	96.5	**97.5**	97.3
es	96.8	96.9	**97.5**	**97.5**
eu	96.1	95.7	96.6	**96.8**
fa	97.5	96.6	**98.6**	98.5
fi	95.8	96.9	**98.3**	**98.3**
fr	96.6	96.7	**96.9**	**96.9**
he	97.4	96.9	**97.9**	97.8
hi	97.4	96.9	**97.9**	97.8
hr	96.8	97.6	97.9	**98.0**
id	94.0	93.7	**93.8**	93.7
it	98.1	98.6	**98.7**	**98.7**
nl	93.8	92.9	**94.0**	93.6
no	98.5	98.6	**99.0**	98.9
pl	97.7	98.5	98.8	**98.9**
pt	98.2	98.3	**98.6**	**98.6**
sl	98.1	98.7	**99.2**	**99.2**
sv	97.4	98.2	**98.9**	**98.9**
low-resource languages				
avg.	92.7	94.7	95.4	**96.1**†
el	98.2	98.6	**98.8**	98.7
et	92.8	94.1	95.7	**96.3**
ga	91.1	92.9	94.1	**95.8**
hu	94.0	96.8	**97.7**	97.5
ro	91.5	95.0	94.9	**95.8**
ta	88.7	90.4	91.3	**92.5**

Table 1: Part-of-speech tagging accuracy (×100) on the standard test set of UD 1.2 high-resource (top) and low-resource (bottom) languages, across different pretrained models and augmentation methods. The best numbers in each row are bolded. SoTA: previous state of the art, i.e., the best test accuracy for each language among all methods reported by Heinzerling and Strube (2019), where all numbers in the SoTA column are not necessarily produced by the same model. † denotes new state-of-the-art results.

4.3 Part-of-Speech Tagging

We conduct our experiments using the Universal Dependencies (UD; Nivre et al., 2016, 2020)[7] dataset.

First, we compare both NoAug and Sub^2 to the previous state-of-the-art performance (Heinzerling and Strube, 2019) to ensure that our baselines are strong enough (Table 1).[8] Heinzerling and Strube (2019) take the token-wise concatenation of mBERT last-layer representations, byte-pair en-

---

[6] Specifically, we use the lexical PPDB-XL (Ganitkevitch and Callison-Burch, 2014; Pavlick et al., 2015) of the appropriate language when applicable.

[7] `http://universaldependencies.org/`

[8] We use UD v1.2 for direct comparison with existing work.

Language (Treebank)	NoAug	CtxSub	Shuf	Sub^2
avg.	92.4	87.1	86.8	**93.0**
be (hse)	96.2	90.3	92.5	**96.9**
lt (hse)	92.7	90.1	88.4	**93.1**
mr (ufal)	87.9	81.5	84.5	**89.1**
ta (ttb)	91.7	85.4	83.2	**92.3**
te (mtg)	**93.8**	88.2	85.6	93.0

Table 2: Part-of-speech tagging accuracy (×100) on the standard test set of selected UD 2.6 low-resource treebanks. The best number in each row is bolded.

coding (BPE; Gage, 1994)–based LSTM hidden states and character-LSTM hidden states as the input to the classifier, and fine-tune the pretrained mBERT during training. We find that using our framework with frozen mBERT and extra learnable layer weight parameters, we are able to obtain competitive or better results than those reported by Heinzerling and Strube (2019); the gains grow larger when using XLM-R, which is trained on larger corpora than mBERT. In addition, by augmenting the training set with Sub^2, we achieve better average accuracy on low-resource languages (paired, one-tailed t-test p-value$=0.028$) while remaining competitive on high-resource languages (no statistically significant difference).

We further test the part-of-speech tagging accuracy on 5 selected low-resource treebanks in the UD 2.6 dataset (Table 2), following the official splits of the dataset. For four of the five treebanks, Sub^2 achieves the best performance among all methods, while also maintaining competitive performance on the Telugu treebank. In contrast, other augmentation methods (CtxSub and Shuf) are harmful compared to NoAug on all treebanks.

4.4 Dependency Parsing

We evaluate the performance of models using the standard Penn Treebank dataset (PTB; Marcus et al., 1993), converted by Stanford dependency converter v3.0,[9] following the standard splits.

We first compare the performance of Sub^2 and baselines in the low-resource setting (Table 3). All methods sometimes, though not always, improve performance over NoAug. Shuf achieves the best LAS when there is only an extremely small training set (e.g., 10 examples) available; however, when the size of the original training set becomes larger, Sub^2 begins to dominate, while CtxSub and Shuf start to sometimes hurt the performance. In addition, a larger augmented dataset does not necessarily lead to better performance, but CtxSub and Shuf often hurt performance as the augmented set size gets too large while Sub^2 does not. Although there is not a consistently best configuration, throughout our experiments, augmenting by 10×–50× the original dataset size produces good improvements over NoAug for both development and test sets. When tuning the augmentation factor k using small development sets, Sub^2 improves over NoAug for four out of five seed dataset sizes. CtxSub, in contrast, improves performance for only one out of five.

$\|\mathcal{D}'\| = k \times \|\mathcal{D}\|$	10	50	$\|\mathcal{D}\|$ 100	500	1,000
2× (CtxSub)	38.3	55.1	$\underline{62.9}$	78.1	80.1
5× (CtxSub)	35.5	$\underline{55.9}$	62.1	81.4	$\underline{81.0}$
10× (CtxSub)	$\underline{\mathbf{39.8}}$	55.1	61.7	$\underline{81.7}$	80.8
50× (CtxSub)	31.2	52.3	60.9	79.3	78.0
100× (CtxSub)	32.0	53.1	58.2	77.1	75.9
2× (Shuf)	32.8	$\underline{55.9}$	62.5	76.7	78.4
5× (Shuf)	34.4	52.7	60.5	77.5	81.6
10× (Shuf)	$\underline{\mathbf{39.8}}$	53.1	$\underline{\mathbf{63.7}}$	77.9	$\underline{81.9}$
50× (Shuf)	34.0	52.7	60.9	79.1	79.6
100× (Shuf)	39.1	55.9	61.3	$\underline{80.4}$	77.4
2× (Sub^2)	$\underline{38.3}$	54.3	61.7	81.0	80.0
5× (Sub^2)	35.9	54.7	62.9	$\underline{\mathbf{82.5}}$	80.4
10× (Sub^2)	32.0	53.9	$\underline{\mathbf{63.7}}$	81.7	80.6
50× (Sub^2)	33.2	$\underline{\mathbf{57.0}}$	62.5	81.4	$\underline{\mathbf{82.5}}$
100× (Sub^2)	38.3	52.7	62.5	78.8	82.1

Table 3: Labeled attachment scores (LAS) on the standard PTB development set (PTB Section 22). We start with an original training set $\mathcal{D}$, which consists of $|\mathcal{D}| \in \{10, 50, 100, 500, 1000\}$ examples, and augment it $k \in \{2, 5, 10, 50, 100\}$ times. For each training set $\mathcal{D}$, the corresponding development set consists of $\max\left(10, \frac{|\mathcal{D}|}{10}\right)$ examples. Underlined results correspond to k values tuned to maximize development set LAS for each combination of augmentation method and $|\mathcal{D}|$ (if there are multiple k values with the same development LAS, we choose the smallest). The best number in each column is bolded.

When training on the full WSJ training set, Sub^2 does not necessarily help improve over NoAug, but it also does not hurt performance (Table 4).[10]

---

[9] `https://nlp.stanford.edu/software/stanford-dependencies.shtml`

[10] An additional finding here is that a simple biaffine dependency parsing model (Dozat and Manning, 2017) with XLM-R initialization is able to set a new state of the art for dependency parsing with only in-domain annotation.

Model	UAS	LAS
Mrini et al. (2020)† (NoAug)	97.4	96.3
Zhang et al. (2020b)‡ (NoAug)	96.1	94.5
BiAffine+XLM-R (NoAug)	96.7	95.2
BiAffine+XLM-R+Sub^2	96.6	95.2

Table 4: Unlabeled attachment score (UAS) and labeled attachment score (LAS) on the PTB dependency test set. Models are trained with the full PTB training set. †: the previously best result using any kind of annotation (e.g., constituency parse trees); ‡: the previously best result using only dependency annotations. BiAffine: the bi-affine dependency parsing model proposed by Dozat and Manning (2017).

4.5 Constituency Parsing

Shi et al. (2020) have shown that Sub^2 can significantly improve few-shot constituency parsing on the Penn Treebank dataset; in this work, we extend the few-shot parsing evaluation to other domains, using the Foreebank (FBank; Kaljahi et al., 2015) and NXT-Switchboard (SWBD; Calhoun et al., 2010) datasets. Foreebank consists of 1,000 English and 1,000 French sentences; for either language, we randomly select 50 sentences for training, 50 for development, and 250 for testing.[11] We follow the standard splits of NXT-Switchboard, and randomly select 50 sentences from the training set and 50 from the development set for training and development respectively.

We compare data augmentation methods using the setup of few-shot parsing from scratch (Table 5). Among all settings we tested, Sub^2 achieves the best performance, while all augmentation methods we investigated improve over training only on the original dataset (NoAug). Surprisingly, we find that the seemingly meaningless Shuf, which randomly shuffles the sentence and inserts the shuffled words back into the original parse tree structure as the nonterminals, also consistently helps few-shot parsing by a nontrivial margin.[12]

For domain adaptation (Table 6), we first train Benepar (Kitaev and Klein, 2018) on the Penn Treebank dataset, achieving an F_1 score of 95.1 on the PTB standard development set, and use the pretrained model as the initialization. While compared to few-shot parsing trained from scratch, the gain by data augmentation generally becomes smaller, Sub^2 still works the best across datasets.

Method	FBank(en)	FBank(fr)	SWBD
NoAug	33.1	27.3	29.1
CtxSub	64.8	59.9	51.1
Syno	62.9	60.8	52.1
Shuf	55.9	48.8	37.0
Sub^2	**71.8**	**70.8**	**64.6**

Table 5: Labeled F_1 scores ($\times 100$) on the test set of each constituency treebank, in the setting of few-shot parsing. The best number in each column is bolded.

Method	FBank(en)	FBank(fr)	SWBD
PTB	82.3	30.8	74.3
NoAug$\rightarrow$	83.1	70.1	77.2
CtxSub $\rightarrow$	84.0	71.1	78.2
Syno $\rightarrow$	84.2	71.0	77.9
Shuf $\rightarrow$	83.5	70.1	75.6
Sub^2 $\rightarrow$	**84.6**	**72.6**	**78.3**

Table 6: Labeled F_1 scores ($\times 100$) on the test set of each constituency treebank, in the setting of domain adaptation. PTB: directly testing the model trained on the Penn Treebank; $\rightarrow$: transferring a model trained on PTB to each domain. The best number in each column is bolded.

4.6 Text Classification

We take text classification as a representative of a wider range of NLP tasks, and evaluate the methods introduced in Section 3.2 and baselines on low-resource versions of two text classification datasets: SST (Socher et al., 2013) and a sentence version of the AG News dataset (Zhang et al., 2015).[13] To avoid over-fitting to the small development set and tuning on test set issues, we introduce small "development test" (devtest) sets for each task, and only evaluate on the test sets using Sub^2 variations with the best devtest performance. For settings requiring constituency parse trees, we generate them using Benepar (Kitaev and Klein, 2018) trained on the standard PTB dataset.

Across the two datasets, any data augmentation technique usually improves over NoAug (Table 7).[14] While methods in the Sub^2 family usually lead to the best performance on both tasks, there is not a consistently best-performing combination of constraints. Surprisingly, Sub^2 without constraints

---

[11] We leave the other 650 sentences for future use.

[12] This trend may be explained by benefits in learning/optimization stability in this few-shot setting, but we leave a richer exploration of potential explanations for future work.

[13] We only keep the single-sentence instances among all examples in each split of the original AG News dataset, following Shi et al. (2018b).

[14] To measure variance due to random selection of data from the full sets, results on additional randomly-sampled few-shot datasets for both tasks can be found in Appendix A.

Dataset	Method	Constraints				Accuracy		
		Text Label	*Constit.*	*Span Label*	*#Words*	*Dev*	*Devtest*	*Test*
AG News-1%	NoAug	N/A	N/A	N/A	N/A	55.2	44.8	44.8
$\lvert\mathcal{D}_{train}\rvert = 0.6$K	CtxSub	N/A	N/A	N/A	N/A	91.0	89.6	86.0
$\lvert\mathcal{D}_{dev}\rvert = 0.06$K	Syno	N/A	N/A	N/A	N/A	89.6	89.6	85.6
$\lvert\mathcal{D}_{devtest}\rvert = 0.06$K	Shuf	N/A	N/A	N/A	N/A	91.0	88.1	86.3
	Sub^2 (balanced tree)	✗	N/A	N/A	✓	88.1	**92.5**	86.7
	Sub^2	✓	✓	✗	✗	91.0	**92.5**	**87.0**
SST-10%	NoAug	N/A	N/A	N/A	N/A	27.3	35.5	23.3
$\lvert\mathcal{D}_{train}\rvert = 0.8$K	CtxSub	N/A	N/A	N/A	N/A	40.0	53.6	44.9
$\lvert\mathcal{D}_{dev}\rvert = 0.1$K	Syno	N/A	N/A	N/A	N/A	40.0	39.1	39.0
$\lvert\mathcal{D}_{devtest}\rvert = 0.1$K	Shuf	N/A	N/A	N/A	N/A	37.3	44.5	38.9
	Sub^2 (balanced tree)	✗	N/A	N/A	✗	40.0	50.0	44.6
	Sub^2	✓	✓	sentiment	✗	40.0	**55.5**	**45.8**

Table 7: Accuracy (×100) on the low-resource sentence AG News and SST datasets, together with the corresponding constraints, where *Constit.* denotes constituency labels. We enumerate multiple constraints for data augmentation with Sub^2, and only test the obtained model with the highest devtest accuracy – results for all investigated combinations of constraints can be found in Appendix A. The best devtest accuracies and the best test accuracy for each dataset are bolded.

Method	Dev. Acc.	Test Acc.
XLM-R (NoAug)	56.1	55.7
XLM-R (Sub^2)	**56.6**	**56.6**
Brahma (2018)	N/A	56.2

Table 8: Accuracy (×100) on the SST standard development and test set.

on the text label, which may introduce more noise than having the constraint, does not necessarily hurt the performance much.

While constituency parse tree–based Sub^2 typically achieves competitive performance among all investigated combinations of constraints, the gain over Sub^2 with balanced trees is not consistent. Our results are in line with Shi et al. (2018b).

We further use Sub^2 with constraints of (1) text label, (2) phrase, and (3) phrase sentiment label, to augment the full SST training set, since it is the best augmentation method for few-shot sentiment classification, in terms of devtest accuracy (Table 7). In addition to sentences, we also add phrases (i.e., subtrees) as training examples, following most existing work (Socher et al., 2013; Kim, 2014; Brahma, 2018, *inter alia*),[15] to boost performance. In this setting, we find that Sub^2 helps set a new state of the art on the SST dataset (Table 8).

5 Conclusion and Future Work

We investigate substructure substitution (Sub^2), a family of data augmentation methods that generates new examples by same-label substructure substitution. Such methods help achieve competitive or better performance on the tasks of part-of-speech tagging, dependency parsing, constituency parsing, and text classification in the few-shot setting, where the number of annotated examples is limited. While other data augmentation methods (e.g., CtxSub and Shuf) sometimes improve the performance, Sub^2 is the only one that consistently improves performance for low-resource NLP across tasks and seed dataset sizes. The experimental setups used in this work can further serve as a standard benchmark for future work on NLP with limited annotations.

There are two open questions remaining to be addressed. First, it is still unclear why Shuf, which requires the model to recover the correct constituency parse tree of a sentence while only accessing shuffled words, consistently helps improve few-shot constituency parsing by a nontrivial margin. Second, while constituency parse tree–based Sub^2 sometimes achieves better performance than Sub^2 without the constituency constraint, the advantage is not large: whether explicit constituency parse trees are useful for NLP applications in the neural network era remains an open question. We leave the above questions, as well as applications of Sub^2 to more NLP tasks, for future work.

---

[15] That is, unlike in Table 7, we apply the same settings as most existing work to produce numbers in Table 8.

6 Acknowledgment

We thank Jiayuan Mao, Shane Settle and Bowen Shi for helpful suggestions on this work, Yu Zhang for fruitful discussion regarding dependency parsing, as well as anonymous reviewers for their valuable feedback.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39, Vancouver. Association for Computational Linguistics.

Siddhartha Brahma. 2018. Improved sentence modeling using suffix bidirectional LSTM. *arXiv preprint arXiv:1805.07340*.

Sasha Calhoun, Jean Carletta, Jason M Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The nxt-format switchboard corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language resources and evaluation*, 44(4):387–419.

Jiaao Chen, Zhenghui Wang, Ran Tian, Zichao Yang, and Diyi Yang. 2020. Local additivity based data augmentation for semi-supervised NER. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1241–1251, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3861–3867, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mathieu Dehouck and Carlos Gómez-Rodríguez. 2020. Data augmentation via subtree swapping for dependency parsing of low-resource languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3818–3830, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Hagen Fürstenau and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 220–228, Athens, Greece. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *The 9th edition* *of the Language Resources and Evaluation Conference*, Reykjavik, Iceland. European Language Resources Association.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy. Association for Computational Linguistics.

Demi Guo, Yoon Kim, and Alexander Rush. 2020. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online. Association for Computational Linguistics.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Benjamin Heinzerling and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, Florence, Italy. Association for Computational Linguistics.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1234–1245, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202, Santiago de Compostela, Spain. Association for Computational Linguistics.

Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn, and Joseph Le Roux. 2015. Foreebank: Syntactic analysis of customer support forums. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1341–1347, Lisbon, Portugal. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2020. Controllable meaning representation to text generation: Linearization and data augmentation strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185, Online. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Jiancheng Lv, Nan Duan, and Ming Zhou. 2020a. Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5798–5810, Online. Association for Computational Linguistics.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020b. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041, Online. Association for Computational Linguistics.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2020. Gender bias in neural natural language processing. In *Logic, Language, and Security*, pages 189–202. Springer.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, Online. Association for Computational Linguistics.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Tong Niu and Mohit Bansal. 2019. Automatically learning data augmentation policies for dialogue tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1317–1323, Hong Kong, China. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Husam Quteineh, Spyridon Samothrakis, and Richard Sutcliffe. 2020. Textual data augmentation for efficient active learning on tiny datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7400–7410, Online. Association for Computational Linguistics.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium. Association for Computational Linguistics.

Yves Schabes. 1990. Mathematical and computational aspects of lexicalized grammars. *PhD. thesis*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7611–7621, Online. Association for Computational Linguistics.

Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang, and Jian Sun. 2018a. Learning visually-grounded semantics from contrastive adversarial samples. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3715–3727, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018b. On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4631–4641, Brussels, Belgium. Association for Computational Linguistics.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99, Vancouver. Association for Computational Linguistics.

Marco Antonio Sobrevilla Cabezudo, Simon Mille, and Thiago Pardo. 2019. Back-translation as strategy to tackle the lack of corpus in natural language generation from semantic representations. In *Proceedings of the 2nd Workshop on Multilingual Surface*

Realisation (MSR 2019), pages 94–103, Hong Kong, China. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana. Association for Computational Linguistics.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, and Lifang He. 2020. Mixup-transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong Kong, China. Association for Computational Linguistics.

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212, Barcelona, Spain (Online). International Committee on Computational Linguistics.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized data augmentation for low-resource translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796, Florence, Italy. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana. Association for Computational Linguistics.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1461–1470, Osaka, Japan. The COLING 2016 Organizing Committee.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 1040–1050, Vancouver, Canada. Association for Computational Linguistics.

Kang Min Yoo, Hanbit Lee, Franck Dernoncourt, Trung Bui, Walter Chang, and Sang-goo Lee. 2020. Variational hierarchical dialog autoencoder for dialog state tracking data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3406–3425, Online. Association for Computational Linguistics.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Yi Zhang, Tao Ge, and Xu Sun. 2020a. Parallel data augmentation for formality style transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3221–3228, Online. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020b. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

Appendices

A Text Classification: Another Few-Shot Dataset

For either sentence AG News or SST dataset, we create two few-shot dataset of the same size, but with different examples. We report the model performances in Tables 9 and 10. Results in Table 7 are evaluated on the same few-shot dataset as those in Table 9.

Method	Constraints				Accuracy		
	Text Label	*Phrase*	*Pharse Label*	*#Words*	*Dev*	*Devtest*	*Test*
AG News-1% ($\lvert\mathcal{D}_{train}\rvert = 0.6$K, $\lvert\mathcal{D}_{dev}\rvert = \lvert\mathcal{D}_{devtest}\rvert = 0.06$K)							
NoAug	N/A	N/A	N/A	N/A	55.2	44.8	44.8
CtxSub	N/A	N/A	N/A	N/A	91.0	**89.6**	86.0
Shuf	N/A	N/A	N/A	N/A	91.0	88.1	86.3
Syno	N/A	N/A	N/A	N/A	89.6	**89.6**	85.6
Sub^2 (balanced tree)	✓	N/A	N/A	✓	89.6	89.6	
	✓	N/A	N/A	✗	89.6	91.0	
	✗	N/A	N/A	✓	88.1	**92.5**	86.7
	✗	N/A	N/A	✗	91.0	91.0	
Sub^2	✓	✓	const.	✓	86.6	89.6	
	✓	✓	const.	✗	86.6	91.0	
	✓	✓	✗	✓	86.6	89.6	
	✓	✓	✗	✗	**91.0**	**92.5**	87.0
	✓	✗	N/A	✗	86.6	**92.5**	
	✓	✗	N/A	✓	86.6	91.0	
	✗	✓	const.	✓	86.6	86.6	
	✗	✓	const.	✗	86.6	86.6	
	✗	✓	✗	✓	91.0	88.1	
	✗	✓	✗	✗	89.6	88.1	
SST-10% ($\lvert\mathcal{D}_{train}\rvert = 0.8$K, $\lvert\mathcal{D}_{dev}\rvert = \lvert\mathcal{D}_{devtest}\rvert = 0.1$K)							
NoAug	N/A	N/A	N/A	N/A	27.3	35.5	23.3
CtxSub	N/A	N/A	N/A	N/A	40.0	**53.6**	44.9
Shuf	N/A	N/A	N/A	N/A	37.3	44.5	38.9
Syno	N/A	N/A	N/A	N/A	40.0	39.1	39.0
Sub^2 (balanced tree)	✓	N/A	N/A	✓	36.4	49.1	
	✓	N/A	N/A	✗	26.4	35.5	
	✗	N/A	N/A	✓	38.2	49.1	
	✗	N/A	N/A	✗	40.0	**50.0**	44.6
Sub^2	✓	✓	senti.	✓	39.1	53.6	
	✓	✓	senti.	✗	40.0	**55.5**	45.8
	✓	✓	const.	✓	37.3	52.7	
	✓	✓	const.	✗	40.9	50.9	
	✓	✓	✗	✓	36.4	50.9	
	✓	✓	✗	✗	39.1	47.3	
	✓	✗	N/A	✓	38.2	48.2	
	✓	✗	N/A	✗	40.9	47.3	
	✗	✓	const.	✓	39.1	50.9	
	✗	✓	const.	✗	40.0	54.5	
	✗	✓	✗	✓	38.2	54.5	
	✗	✓	✗	✗	39.1	50.0	

Table 9: Accuracy (×100) on the AG News and SST dataset (few-shot set 1). The best devtest numbers in each section are bolded.

Method	Constraints				Accuracy		
	Text Label	*Phrase*	*Pharse Label*	*#Words*	*Dev*	*Devtest*	*Test*
AG News-1% $(\lvert\mathcal{D}_{train}\rvert = 0.6\text{K}, \lvert\mathcal{D}_{dev}\rvert = \lvert\mathcal{D}_{devtest}\rvert = 0.06\text{K})$							
NoAug	✓	N/A	N/A	N/A	40.3	44.8	38.0
CtxSub	N/A	N/A	N/A	N/A	88.1	**89.6**	86.1
Shuf	N/A	N/A	N/A	N/A	86.6	88.1	85.7
Syno	N/A	N/A	N/A	N/A	88.1	85.1	84.2
Sub^2 (balanced tree)	✓	N/A	N/A	✗	88.1	88.1	
	✓	N/A	N/A	✓	86.6	88.1	
	✗	N/A	N/A	✗	89.6	89.6	
	✗	N/A	N/A	✓	89.6	**91.0**	86.5
Sub^2	✓	✓	const.	✓	86.6	**88.1**	
	✓	✓	const.	✗	86.6	**88.1**	
	✓	✓	✗	✓	88.1	**88.1**	
	✓	✓	✗	✗	86.6	**88.1**	
	✓	✗	N/A	✓	86.6	**88.1**	
	✓	✗	N/A	✗	**89.6**	**88.1**	85.9
	✗	✓	const.	✓	86.6	86.6	
	✗	✓	const.	✗	83.6	**88.1**	
	✗	✓	✗	✓	83.6	86.6	
	✗	✓	✗	✗	82.1	**88.1**	
SST-10% $(\lvert\mathcal{D}_{train}\rvert = 0.8\text{K}, \lvert\mathcal{D}_{dev}\rvert = \lvert\mathcal{D}_{devtest}\rvert = 0.1\text{K})$							
NoAug	✓	N/A	N/A	N/A	30.0	30.0	26.7
CtxSub	N/A	N/A	N/A	N/A	47.3	43.6	43.2
Shuf	N/A	N/A	N/A	N/A	47.3	41.8	41.5
Syno	N/A	N/A	N/A	N/A	45.5	37.3	40.0
Sub^2 (balanced tree)	✓	N/A	N/A	✓	46.4	**44.5**	44.1
	✓	N/A	N/A	✗	48.2	43.6	
	✗	N/A	N/A	✓	31.8	36.4	
	✗	N/A	N/A	✗	43.6	37.3	
Sub^2	✓	✓	senti.	✓	49.1	**45.5**	44.7
	✓	✓	senti.	✗	45.5	43.6	
	✓	✓	const.	✓	47.3	42.7	
	✓	✓	const.	✗	45.5	41.8	
	✓	✓	✗	✓	42.7	41.8	
	✓	✓	✗	✗	41.8	38.2	
	✓	✗	N/A	✓	36.4	42.7	
	✓	✗	N/A	✗	37.3	38.2	
	✗	✓	const.	✓	48.2	43.6	
	✗	✓	const.	✗	35.5	37.3	
	✗	✓	✗	✓	48.2	41.8	
	✗	✓	✗	✗	45.5	40.0	

Table 10: Accuracy (×100) on the AG News and SST dataset (few-shot set 2). The best devtest numbers in each section are bolded .

Towards Protecting Vital Healthcare Programs by Extracting Actionable Knowledge from Policy

Vanessa Lopez[*a], Nagesh Yadav[b], Gabriele Picco[a], Inge Vejsbjerg[a], Eoin Carroll[b], Seamus Brady[b], Marco Luca Sbodio[a], Lam Thanh Hoang[a], Miao Wei[b], John Segrave[*b]

[a] IBM Research Europe, Dublin, Ireland
[b] IBM Watson Health, Dublin, Ireland
*Equal author contribution

Abstract

In challenging economic times, obtaining value for money by ensuring financial integrity and fairer distribution of services are among the top priorities for social and health-care systems globally. However, healthcare billing policies are complex and identifying non-compliance is often narrow-scope, manual and expensive. Maintaining 'integrity' is a challenge - ensuring that scarce resources get to those in need and are not lost to fraud and waste. Our approach fuses recent advances in dependency parsing with a policy ontology to convert the content of regulatory healthcare policy into human-friendly policy rules, that are amenable to machine-execution, with human oversight. We describe the ontology-guided transformation of textual patterns into a semantically-meaningful knowledge graph of rules, outline our experiments and evaluate results against policy rules obtained from professional investigators. The aim is to make a policy-compliance 'landscape' visible to healthcare programs - helping them identify Fraud, Waste or Abuse.

1 Introduction

The WHO (World Health Organization, 2010) lists fairness, financial integrity ["Program Integrity"] and access in healthcare among the top global healthcare priorities. In the U.S., an estimated annual amount of USD$20-30B is lost to **Fraud, Waste and abuse (FWA)** (Shrank et al., 2019). These vital funds never make it to the vulnerable citizens that they were intended to serve.

To combat this, countries with insurance-based healthcare programs (e.g. Medicaid, Medicare), employ claim investigators to validate the integrity of reimbursement claims submitted by providers. Investigators verify these claims against the program's policies, with the goal of reducing wasteful practices, identifying fraud or abuse and closing policy gaps. This is a labor-intensive task – claim volumes are high, policies are complex and investigative resources are limited.

More broadly, governments regulate a wide range of sectors, with extensive rules and policies. These policies drive significant spending by the regulated organizations. e.g., a European Union Commission study (2019) found that the annual cost of complying with EU financial regulations is around EUR€11.3B. In Australia, Deloitte (2014) estimated the costs of administering and complying with public sector rules at AUD$94B.

In this paper we propose a methodology for automatically extracting knowledge from healthcare policy documents, in the form of **Benefit Rules (BRs)** that are both human-understandable and machine consumable. These BRs can be applied automatically to flag discrepancies in claims, with limited human effort. This paper focuses on the extraction of these rules, and not their execution.

Policy rules have a major impact both on the health of the citizens they serve, and the financial integrity of the Programs that pay the Service Providers. Human-understanding, oversight and control are first-class AI-design concerns for this domain. Model 'explainability' is not enough. Transparency is needed anywhere that provider or citizen coverage is at stake. Users need to see and influence which rules are being applied, and on what policy basis decisions are being made.

To achieve this, we anchor our methodology in an ontology[1], that both guides and constrains AI extraction tasks. We build on recent NLP advances

---

[1] The ontology and Benefit Rules benchmark are released as open source at: https://github.com/IBM/rules_extraction_from_healthcare_policy

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3509–3521
August 1–6, 2021.

to identify patterns in dependency paths that connect relevant entities.

Finally, we transform the dependent entities into knowledge-graph fragments, which are assembled into graphs that represent actionable Benefit Rules. Users can curate these rules as they are well-structured and expressed in familiar terms.

Automated extraction of these rules from high-volume policies (e.g., Medicaid), will enable the emergence of a new generation of tools for safeguarding Program Integrity, e.g. execution of these rules against claims data enables an overview of the 'policy-compliance landscape' that does not exist today.

Section 2 presents Rules as Code and related work in knowledge extraction for social good. Section 3 describes domain requirements. Section 4 presents the architecture for our 'Claim Audit' extraction pipeline. Section 5 describes how dependency parsing and the ontology are used to extract semantically-rich rule fragments. In Section 6 we evaluate our results with professional policy investigators using dental policies, and in Section 7, we present future work.

2 Related Work

2.1 Rules As Code (RaC)

Organizations need novel approaches to help with regulatory compliance, and **Rules as Code (RaC)** is an initiative that envisages *"an official version of rules (e.g., laws and regulations) in a machine-consumable form, which allows rules to be understood and actioned by computer systems in a consistent way"* (Mohun et al., 2020). It forms part of a broad movement towards digital government and has garnered broad public-sector interest.

Approaches to achieving machine-executable RaC rules for published legislation run from manual coding by multi-disciplinary teams to automatic code-generation from natural-language legislation (Mohun et al., 2020). The former approach brings legislative drafters, policy analysts and software developers together to co-produce human and machine-consumable versions of rules. Examples include the New Zealand Better Rules Discovery initiative (Digital Government NZ, 2018) and OpenFisca (OpenFisca.org, nd) in France. The latter approach uses NLP technology to assist policy experts in converting policy texts to machine-consumable forms, helping scale the RaC process, and is being explored by the AustLII's DataLex Project *(*Greenleaf et al., 2020*)* and CSIRO Data61's Regulation as a Platform project (Data61, 2019*)*. Our work involves taking this latter approach for government healthcare insurance policy.

2.2 Knowledge Extraction for Social Good

Related to our work, Kiyavitskaya et al. (2018) extracts right, obligation, exception and constraints from legal documents by annotating entities with a domain-specific ontology consisting of entity vocabulary and normative phrasal templates built manually by domain experts. Dragoni et al. (2016) parses sentences into grammar trees using Stanford NLP, and annotates legal concepts with a manually-built ontology. The annotation is turned into rules using a set of hand-crafted rules.

Our work is similar, but we extract tuples from dependency trees and reason over the ontology to produce a Benefit Rule knowledge graph.

Recent efforts to build graphs for social good include (Assom, 2020) - constructing knowledge graphs to extract food-trading activities for sustainable food trading and security. Puri et al. (2020) discusses challenges in extracting knowledge graphs from UN datasets for sustainable development goals. Khetan et al. (2020) describes the use of NLP tools like Spacy, CoreNLP, ClausIE and OpenIE to extract information from unstructured text provided by the UN. Kejriwal et al. (2017) constructs a knowledge graph that supports a semantic search engine for investigators in human-trafficking.

AI for social good is a broad research topic as described in Shi et al. (2020). For public health, Nordon et al. (2019) uses biomedical knowledge graph for drug discovery. Finally, Percha and Altman (2018) connect entity-pair dependency paths to extract relations between chemicals, genes and diseases.

Our focus is on AI and knowledge-extraction to help combat FWA in healthcare programs. Today, FWA detection generally relies on two approaches. Firstly, traditional data mining to identify outliers and anomalous billing patterns in claims (Joudaki et al., 2015). While valuable, this approach presents challenges when building legal cases, as it is not innately grounded in policy. Secondly, hand-coded algorithms written by analysts to find claims that are not compliant with policy. These are labor-intensive to develop and maintain in the face of ever-evolving policy. Worse, they cover only a small fraction of the policy ‘landscape’, resulting in a prioritization catch-22 - algorithms are needed in order to know which areas most need algorithms.

3 Background and Requirements

Despite the variety of ways that Benefit Rules (BRs) are expressed across healthcare policies, experts know the common entities, relationships and logical constraints that underpin them. To extract BRs that are correct, human-friendly and executable, we need to account for these semantics. Popular generic language models do not capture such implicit knowledge and expert-labeled datasets are expensive to develop (and small).

We apply these semantics via a BR ontology co-created with domain experts as described in (Lopez et al., 2019). The ontology guides the extraction of well-structured, consistent knowledge graphs from the policy. It links relevant entities together with their context in sentences, e.g., take the policy paragraph and its ontology subset shown in Figure 1. To extract the two distinct Benefit Rules shown in Figure 2, the two distinct roles of the service ‘*full-mouth debridement*’ must be recognized. Widely-adopted medical terminology standards (e.g. UMLS, CPT, HCPCS) can be attached to ontology concepts for consistent representation and mapping of billing codes/values in a BR.

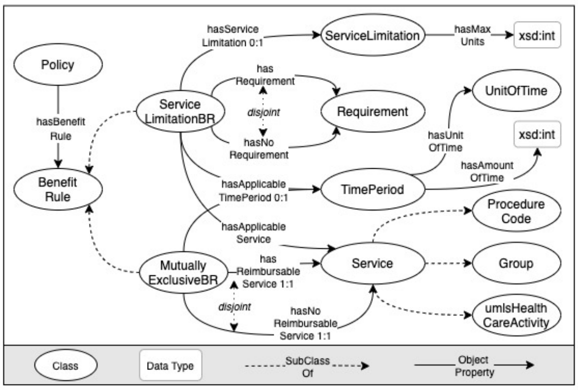

Figure 1: Ontology subset describing the policy: *Full-mouth debridement to enable comprehensive periodontal evaluation and diagnosis is a covered service and does not require prior approval. It is payable once in a 24-month period. Full-mouth debridement is not payable on the same date of services as other prophylactic or preventative procedures*

Human understanding and control are also key requirements. Policy can be challenging to interpret, and amenable to mis-interpretation or mis-application. Expert operators must be able to understand the policy provenance and correct any incorrectly-extracted BRs. The ontology powers this 'explainability' in two ways. Firstly, by enabling consistently-structured knowledge graphs to be extracted from inconsistent policy representations, removing ambiguities between the rules interpretation and policy intent. Secondly – by expressing them simply, using familiar user concepts. All our graphs can be presented as a set of simple, editable condition-value pairs, as in Figure 2. This is achieved by 'flattening' the graph - taking only the leaf properties and values. While our UI needs visual design, users report that these representations 'feel right' and that correcting extraction errors/omissions is straightforward.

Finally, extracted BRs must be amenable to execution - i.e., converted into a form that can automatically label claims as policy-compliant (or not). Here again, the ontology helps by mapping each condition to consistent constraints - this time for the selection, filtering and aggregation of claims data. While beyond the scope of this paper, our work to-date suggests that ontology-conforming BRs execute with similar accuracy to algorithms hand-written by claim investigators. (Of course, execution accuracy will still also depend on accuracy of automated extraction and human curation).

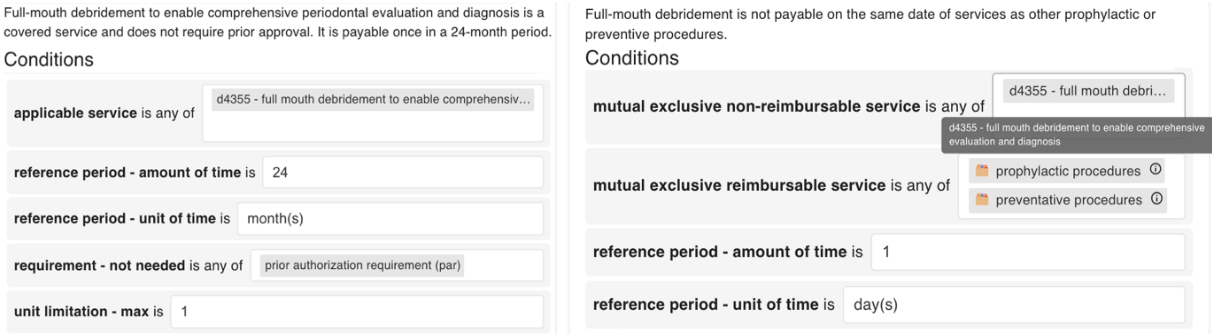

Full-mouth debridement to enable comprehensive periodontal evaluation and diagnosis is a covered service and does not require prior approval. It is payable once in a 24-month period.

Conditions

applicable service is any of d4355 - full mouth debridement to enable comprehensiv...

reference period - amount of time is 24

reference period - unit of time is month(s)

requirement - not needed is any of prior authorization requirement (par)

unit limitation - max is 1

Full-mouth debridement is not payable on the same date of services as other prophylactic or preventive procedures.

Conditions

mutual exclusive non-reimbursable service is any of d4355 - full mouth debri...

d4355 - full mouth debridement to enable comprehensive evaluation and diagnosis

mutual exclusive reimbursable service is any of prophylactic procedures ⓘ preventative procedures ⓘ

reference period - amount of time is 1

reference period - unit of time is day(s)

Figure 2: Benefit Rules extracted from a paragraph in a dental policy (DHS, 2013). On the left: a Service Limitation BR on the number of units a provider can bill for a service per patient over a period of time. On the right: a Mutually Exclusive BR on services that cannot be billed together in a given period

4 Proposed Extraction Pipeline

We propose a methodology to extract Benefit Rules from policy automatically. Figure 3 presents the main components, from policy enrichment to ontology-guided knowledge extraction and user validation. All steps in the pipeline are configurable - similar functional components can easily be replaced/added. We give an overview for each step, the functionality and data requirements:

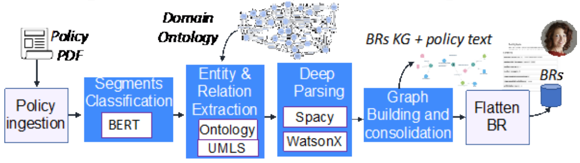

Figure 3. Pipeline from extraction of Benefit Rules

Data preparation and Policy ingestion: all domain information is captured in the ontology, so the technical components remain domain agnostic. While the ontology schema generalises across many policies, instance data is domain-specific and must be prepared ('lifted'). Some instances are common across policy areas and states, such as an eligible 'Place of Service' (e.g. *hospital)* (Centers for Medicare & Medicaid, n.d.). Other instance data is specific to the target domain, such as 'body parts' for tooth identifiers in dental claims. These can be automatically 'lifted' into the ontology from tabular data sources containing a main entity, a list of surface forms (to address vocabulary heterogeneity) and other attributes. They are added as individuals of a given entity type on application startup, and according to a user-configurable mapping. The lexicalizations are made available when processing policy text and identifying entity mentions. New instance data can be added as the need arises

Next, PDF policies are transformed into enriched HTML using an off-the-shelf conversion tool (compare and comply, n.d.), outlining headings, passages and paragraphs for later use at passage-level by annotators and extractors, as well as at paragraph-level by the classification and consolidation.

Segment Classification: a classifier optionally filters incoming paragraphs, deciding whether they are likely to contain BRs. This can significantly reduce compute-time and can also improve Precision, at the cost of some Recall. This is the only component in the pipeline that requires enough ground truth data on (validated) BRs and the associated paragraphs to fine-tune deep learning models. In section 6 we evaluate a fine-tuned, BERT-based classifier.

Entity & Relation Extraction: Here, candidate ontology entities/types are annotated in the text by two complementary annotators. The first is based on WatsonX (Kalyanpur et al., 2012) , a generic entity and UMLS-based clinical annotator. A Lucene search index is used to find approximate matches for any annotated entities (verbs, noun-phrases, etc.), in the ontology lexicalizations. It is also used to retrieve semantic types (e.g. diseases) relevant to benefit rules, from terminology services such as UMLS (UMLS semantic hierarchy, nd).

The second - SystemT (Chiticariu et al., 2018), extracts entity mentions from dictionaries and regular expressions. These are built automatically, from the instances 'lifted' into the ontology at initialization time, as described earlier. This enables entity mentions to be matched with complex labels, like '*Full-mouth debridement to enable comprehensive periodontal evaluation and diagnoses'*. All lexicalizations in the ontology are useful to address vocabulary heterogeneity.

Annotators label these textual spans with specific ontology labels (URIs) and other useful information (lemma, POS, UMLS type, etc). These annotations can later be used to simplify the dependency trees of sentences containing complex entities. Since there can be overlapping annotations and competing annotations for the same span, heuristics are used – e.g. 'longest span', or preferring exact matches to approximate ones. Disambiguation is otherwise performed later.

Deep Parsing and Graph Building: In this step, BR knowledge graphs are obtained by combining dependency trees together with the annotations, in an ontology-guided way. First, linguistic links are identified between annotated entities. Then, these are converted into semantic triples and linked together into knowledge graph fragments by reasoning over the ontology. Fragments from different sentences in a paragraph are **consolidated** together to produce a set of well-formed Benefit Rule knowledge graphs that respect the ontology semantics. Finally, to enable human oversight and control, all knowledge graphs are 'flattened' into a user-friendly, editable format. The ability to do this flattening is a key 'explainability' property of the ontology structure and thus, of knowledge graphs derived from it. The extracted and curated rules are then stored in a Knowledge Base.

User Validation: extracted BRs are shown to investigators in a prototype workbench. Here they can be reviewed against the corresponding policy text and corrected when necessary. Validated BRs form a shared store of high-quality, machine-readable rules, making the rule creation and consumption process more transparent. In a related work, we execute these rules on claims data to discover inappropriate payments. This shortens the investigator's workflow by giving them an immediate view of policy-relevant claims data (normally obtained through time-consuming data requests, spreadsheets and algorithm coding). It also grounds their work to specific policy clauses, helping them build a watertight case for recovery.

5 Knowledge Extraction Approaches

Dependency parsing has been frequently used to support relation extraction by capturing words that are close in context, even if far in sentence distance. Given a dependency tree, a set of subtree extraction rules identifies linguistically-connected terms, in the form of Predicate Argument Structure (**PAS**) tuples (Section 5.1). PAS tuples represent dependencies between textual entities, such as binary or ternary relations. They provide an easy intermediate representation to match text sentences to triples (subject, predicate, object) constituting a knowledge graph. The ontology can then be used to check if the linguistic tuples make sense semantically. Transformation of PAS tuples into Knowledge Graph fragments (sets of ontology triples) is done following a set of semantic templates (Section 5.2.).

To extract PAS tuples, we first implemented a deterministic baseline using WatsonX general-purpose deep parsing engine (Kalyanpur et al., 2012), which builds dependency trees for sentences. WatsonX provides a *Pattern Matching* library to characterize subtrees via handcrafted rules. Table 1 shows an example of two rules, based on the simple dependency tree in Figure 4. The rule assigns a *syntactic* role to tokens/spans in the sentence – e.g., 'subject', 'predicate' or 'object' (also referred to as 'slot types'). Other roles such as 'complement', can also be applied as necessary.

However, hand-coding these syntactic rules requires knowledge of computational linguistics. Our aim is for non-linguistic experts (e.g., application developers) to be able to apply this process to new domains. To reduce dependence on hard-to-acquire dependency parsing skills, we have developed an approach based on Spacy (Honnibal et al., 2020) for learning these rules from curated examples (Section 5.1). In Section 6, we compare performance of these learned rules to the baseline hand-coded rules.

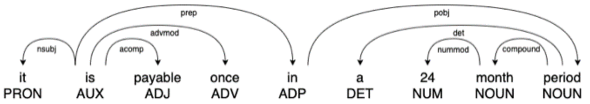

Figure 4. Dependency tree for an example sentence

pattern1→verb[hasPOS('verb'), hasLemma('be')] { nsubj -> **subj** [hasPOS("pron")] } { acomp-> **pred** [ ] } { advmod -> **comp** [hasPOS ("noun")] } { prep -> prepVar [ ] { pobj -> **obj** [hasPOS ("noun")] } } **pattern2** -> **subj** [hasPOS ("noun")] { compound -> **obj** [] { num_mod -> **comp** [] }}
PAS[pattern1]:subj=it,pred=payable,obj=once,comp=period PAS[pattern2]:subj=period, pred=[], obj=month, comp=24

Table 1: Example of written rules to extract PAS.

5.1 PAS Extraction Based on Learned Rules

Using a curated set of sentences with labelled PAS tuples, our framework generates dependency parsing rules that can obtain corresponding tuples from other, similarly-structured sentences. Specifically, we use Spacy's pattern builder to extract *Semgrex* patterns between fully-connected tokens, based on the shortest dependency path between tokens. This path usually contains the necessary information to identify their relation. Semgrex syntax allows us to characterize a subtree (Chambers et al., 2007), it describes nodes with normal token attributes, and how these nodes connect to other nodes in the dependency tree.

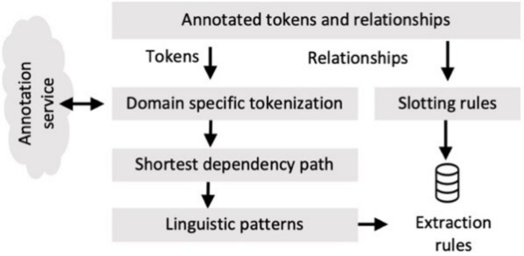

Figure 5. Extraction rule learning phase

We begin by obtaining a collection of sentences that represent the linguistic relationships we want to extract. We then annotate these sentences, identifying the interesting tokens and their syntactic roles (PAS 'slots'). Every annotated example is used to learn extraction rules, as depicted in Figure 5.

To learn the extraction rules for one curated sentence, we start by applying domain-specific tokenization, using the ontology-based entities annotated during Entity Extraction. We then build a dependency tree for the sentence. These annotations help subsequent dependency parsing by letting the parser know when a complex, multi-token term (e.g., the service name "*Full-mouth debridement to enable comprehensive* ..") can be treated as a simple, single named-entity. The re-tokenization simplifies and lends a degree of consistency to the resulting dependency tree (Finkel and Manning, 2009). The dependency tree is then parsed to find the shortest dependency path between the interesting (annotated) roles/slots for a PAS. Once the subtree consisting of all the desired tokens for a PAS has been identified, we extract a linguistic pattern characterizing that subtree. In addition to the linguistic properties of the tokens, we also extract slotting rules. A slotting rule is used in conjunction with linguistic patterns to assign a syntactic role to an extracted token. This is based on (Choi and Palmer, 2012) where dependency labels can be assigned to arguments – i.e, they indicate the Slot type (subject, predicate, object, etc.) Finally, an extraction rule (linguistic -semgrex - pattern + slotting rules) is captured for every PAS in an annotated sentence.

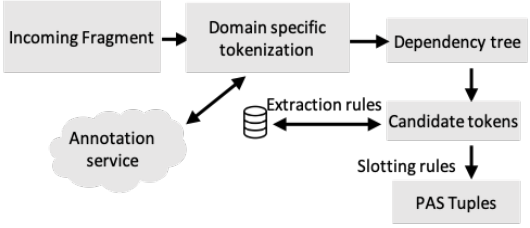

Figure 6. Runtime extraction of candidate PAS tuples

At runtime, incoming sentences are processed to extract PAS tuples (Figure 6). As before, domain-specific tokenization is applied to the sentence prior to building a dependency tree. The extraction rules learned earlier are then applied, to obtain candidate PAS tokens. Lastly, syntactic role labels are assigned to these candidate tokens by applying slotting rules.

5.2 Graph Building from PAS

PAS tuples enable us to extract meaningful relationships, even in text with challenging, long-range dependencies. However, PAS tuples require some translation to match ontology entities and relationships, e.g., a linguistic predicate may not directly translate to an ontological property. Implicit arguments may also be missing from a PAS. PAS can contain ternary relations that need to be aligned to one or more binary relations. And finally, not all PAS tuples are relevant.

To translate PAS tuples into semantically consistent Knowledge Graph fragments, we start by only keeping PAS that contain one or more ontology-based entity annotations. Then, for each subset of connected PAS tuples, we search for non-ambiguous semantic paths in the ontology that connect these entities, based on *parametrized* templates - implemented using the Jena API (Carrol et al., 2004). We use the annotated semantic types of the PAS entities (e.g.: class, instance, property, datatype, etc.) to select the templates to be executed. If a PAS token was annotated with more than one ontology annotation, then all combinations are tried. Non-relevant candidates will likely not yield any meaningful graph fragments. In here we provide an illustrative example of the process. Further details on the different templates can be found in Appendix C.

Consider the example in Figure 1, the first sentence yields the following PAS (among others):

```
<:d4355, :hasApplicableService, :Service>
<:d4355,    :hasNoRequirement,     :PAR>
```

The first PAS fires a template pattern that checks if the class *Service* is both the type of the instance *d4355* and the range of the object property *hasApplicableService*. If so, these entities are semantically connected and can be translated into the corresponding knowledge graph fragments. The following semantic triples are created:

`:br1 rdf:type :ServiceLimitationBR`

`:br1 :hasApplicableService :d4355`

For the second PAS, the range of the property *hasNoRequirement* corresponds to the type of the object instance *PAR*. This PAS links to the previous through the subject instance *:d4355* and can be translated into the knowledge graph fragment:

`:br1 :hasNoRequirement :PAR`

Thus, a knowledge graph is built by joining together all graph fragments obtained from the subset of connected PAS tuples.

Finally, a consolidation step pulls together the collection of graphs extracted from different portions of a policy paragraph. Here, all the constraints expressed in the ontology are enforced, e.g. *disjointness* between two properties, *min* and *max cardinality,* to ensure semantically-meaningful rules and discard nonsense rules (e.g. there can only be one applicable time period per BR – due to max cardinality = 1). Graph fragments are also merged, if the resulting BR graph does not violate any ontological constraint. Since there can more than one extractor, any duplicate graphs derived from the same policy text are discarded, partial graphs can also be merged if doing so does not violate any ontological constraint.

Figure 7 shows the consolidated graph extracted from sentence 1 and 2. This graph can be 'flattened' to obtain human-readable 'Benefit Rules' – such as the Service Limitation rule shown on the left of Figure 2 and Mutually Exclusive rule shown on the right. Both subtypes inherit the properties of their parent Benefit Rule class, but are not merged, because each contains conditions that are only meaningful for that rule subtype, e.g., the property *hasServiceLimitation* is only relevant for a 'Service Limitation' rule.

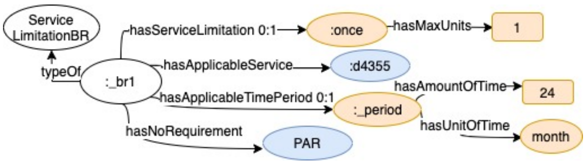

Figure 7. Consolidated graph from sentence 1 and 2

6 Evaluation Methodology

We concentrate on the system's ability to exploit the rich information contained in both a domain ontology and dependency trees with respect to a gold-standard created in consultation with our policy investigators. In particular, we aim to compare the impact of first addressing the need for dependency rules to extract PAS by proposing an approach that generates these rules from examples, which fit the domain-specific characteristics of new policy text. Policy-aware users can add these examples as there is no requirement to manually write new dependency parsing rules. Second, the impact of using a BERT-based classifier. Labeled training data is expensive to acquire as it requires domain expertise. However, as policy investigators review Benefit Rules, we investigated the use of this curated, small labeled dataset to fine-tune a classifier that filters out paragraphs that do not appear to contain any rules.

6.1 Set up: Data and Metrics

The proposed extraction pipeline was evaluated using Benefit Rules extracted from unstructured policies from two different states in the US. The ground truth of BRs for each policy was manually created by a team of three FWA investigators using our prototype User Interface: 90 rules were provided for State1 and 51 rules for State 2.

The ontology used in the experiments consists of 34 classes and 43 properties. Once the domain-specific instance data is 'lifted' in, 4954 individuals are added along with 23250 lexicalizations (i.e., labels used to annotate textual entities). The ground truth Benefit Rules presented in the experiments (not commercially sensitive or in production) are made available together with the ontology.

Each BR comprises of a policy text and a corresponding set of condition-values describing the text. Precision (P) measures the proportion of extracted rules that match the ground truth (GT) for the same policy text. Recall (R) measures the proportion of GT rules correctly extracted. F_1 combines the two. In addition, each matched rule gets a pairing 'score'. When all extracted condition-value pairs for a rule exactly match the GT, the score is 1. For partial matches, the score is between 0 and 1. A score of 0 indicates a missed rule. Table 2 shows the average pairing score across all rules. Details on the calculation and execution environment can be found in the appendixes A and B.

Jupyter notebooks showcasing the dependency tree (before and after tokenization) and PAS tuple extraction implementation with the learned rule for the example used in this paper, can be found in the git repository.

6.2 Results and Discussion

We run the evaluations using the two approaches presented for the PAS tuple extraction. The first – based on 54 manually-coded linguistic rules in WatsonX - acts as a baseline. The second are the learned syntactic rules described in Section 5.1. A total of 55 sentences were annotated with PAS examples.

State	Extractor	R	P	Avg score	F1
1	Learned rules	0.51	0.90	0.56	**0.65**
	Manual rules	0.48	0.73	0.57	0.58
2	Learned rules	0.45	0.82	0.41	0.58
	Manual rules	0.57	0.71	0.54	**0.62**

Table 2: Metrics when different extractors are applied over policies from different states

Results are summarized in Table 2. Overall, figures indicate that the proposed pipeline is a promising step towards automated extraction of BRs from unstructured policies. Extraction using learned rules is comparable to using manual rules. Considering the number of curated examples and rules added to both extractors, there is a potential to further improve this by identifying missed rules and adding them as examples. This requires far less skill than manually hand-crafting linguistic extraction rules. We believe this takes a distinct step towards empowering development teams to tailor extraction to customer needs.

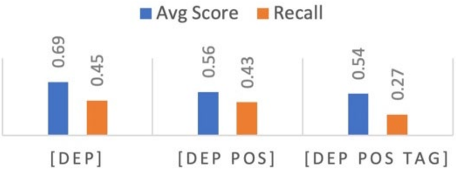

Table 3: Performance when various combinations of linguistic features are used to characterize subtree

Spacy provides several linguistic features on which to characterize a subtree, including syntactic dependency (DEP), part-of-speech (POS) and detailed part-of-speech (TAG). When settling on which of these to use (Table 3), we used Recall as the key metric and found that syntactic dependency (DEP) yielded best performance.

The use of a filtering classifier increases both performance (execution times shown in appendix A) and Precision (from 0.88 to 0.96 in State 1 and 0.82 to 0.95 in State 2) at the cost of a small drop in Recall (from 0.51 to 0.48 in State 1 and 0.45 to 0.41 in State 2). This is expected, since there are inevitably some false negatives in the classification. The model used is a BERT-based text classifier, fine-tuned on 70% of the paragraphs of the two policies, with the remaining 30% used for validation and early stopping during training. The hyperparameters were selected using Optuna (Takuya et al., 2019) optimization framework using 5-fold cross validation settings. The generalization capacity of the model is difficult to assess due to the small amount of labelled ground truth. With that caveat, it has been verified with cross-validation, where the model obtains an average accuracy of 96% on the various folds. The training data size is currently small due to the cost of manual policy labelling. However, it is expected this will expand as users review and curate extractor output.

While these benefit rule extraction results are promising, there is clearly scope to enhance our models and extractors to improve coverage. Most Benefit Rules are self-contained across one or more sentences in a paragraph, However, further work is needed to automatically capture the knowledge from headings, tables or co-references that span paragraphs. Improvements in service annotation could also reduce the incidence of partially-extracted rules being discarded during consolidation. Of course, there can always be implicit information, available in the minds of policy consumers but not present in the policy text for extractors to see. For example, the meaning of *'fair to good'* expressed in the rule *"Restorative services are payable when there is a fair to good prognosis for maintaining the tooth"*. While these details cannot currently be extracted automatically, our system gives policy analysts the ability to capture them in other consistent ways. For example, simple options like attaching a 'medical necessity' label can make cases like these amenable to machine learning and offer real benefits to Program Integrity workers seeking to size and prioritize work.

Various approaches can be used to look beyond co-occurrence of entities in sentences and explore how the terms are linguistically and semantically connected. For instance (Roth and Lapata, 2015) (Lopez et al., 2019) use semantic role labeling to identify actions and roles in a sentence (agent, theme, polarity, etc.) and reason over these to expose relation-entity/value pairs. Other complementary extraction approaches can be leveraged to look beyond co-occurrence of entities in the sentence and explore how the terms are linguistically and semantically connected, which could improve further on rule extraction coverage, such deep learning models. While the training data set is currently small, due to the cost of manual policy labeling, it is expected this will expand as users review and curate the extractors' output.

In this paper, we have explored building over dependency PAS, which can be exploited to capture fine-grained and distant relationships. Using a 'learned-rule' approach to obtain intermediate representations from a sentence enables non-linguistic experts to extend this process to more policies, without requiring manual production of syntactic-rules to address syntactical variability.

We made a small analysis to quantify the similarity between our two dental policy texts P_1 and P_2. We extracted sentences (removing stop words) from P_1 and P_2 using a standard sentence tokenizer (NLK Tokeniser, n.d.). Following (Reimers et al., 2019), we computed embeddings for each sentence using a model optimized for

Semantic Textual Similarity (STS) (specifically, we used stsb-roberta-large -STSb performance:86.39 (SBert, n.d.)). Let Ei be the list of sentence embeddings for the sentences of policy Pi. We have computed pairwise cosine similarity between E_1 and E_2. Let E_{12_max} be the list of the maximum values of the cosine similarity between each embedding in E_1 and all the embeddings in E_2 (the i-th value of E_{12_max} gives the best cosine similarity between the i-th sentence in P_1 and some sentence in P_2). We found that the mean of E_{12_max} is 0.65 with a standard deviation of 0.12; about 45% of the sentences in P_1 have a cosine similarity score with some sentence in P_2 that is above the average value. We repeated the same experiment using a third policy text from a different domain P_3, and found that the similarity scores for sentences in P_1 with respect to P_3 were consistently lower than between P_1 and P_2. This small experiment gives an indication that policy texts from different states but in the same domain (e.g., dental) have a good degree of similarity. The Benefit Rule Ground Truth built for each of the two policies can be seen in the git repository, each Benefit Rule shows the policy paragraph of text from where it is extracted and condition-values.

7 Conclusions and Future Work

Within regulated organizations, visibility over any part of the 'compliance landscape' is valuable information for identifying Program Integrity risks and prioritizing follow-up. With healthcare insurance policies, typically only a small fraction policy is automated (hand-coded), and the rest of the landscape remains opaque. So, unlike many AI applications that require high Recall before they are useful, any Recall here provides concrete, actionable value that claim workers can use both for prioritization and for follow-up. In a dark room, all light helps.

The ontology abstraction supports transferability, e.g. from one US state to another that semantically express similar compliance rules concepts or conditions (e.g.: eligible members, places of service, maximum billable units of service, services that should not be billed together, etc.) even if the wording differs between the texts. It also supports incremental enhancement to cover more rules and/or policy areas, which enables a scalable market. Dependency trees capture fine-grained, distant connections, which guided by the ontology are used to automatically build and consolidate a semantically meaningful Knowledge Graph. This increases precision by allowing for paragraphs that mention relevant entities, but which don't contain Benefit-Rules, to be discarded. If syntactical variability is high, using a 'learned rule' approach to obtain PAS representation enables non-linguistic experts to apply this process to more policies without requiring manual production of syntactic rules.

We see interesting avenues for follow-up NLP / AI work that combines neural and symbolic approaches, e.g., training deep learning models as our users validate more policy rules, to recognize rule fragments (spans), then assemble them into rules using the ontology as a blueprint. We also plan to perform user-based evaluations to measure how our pipeline impacts the time taken by investigators to identify and resolve FWA leads, as well as the quality and scope of the compliance information provided.

Finally, while our methodology is shown in a Healthcare setting, we believe it is applicable to other regulated domains, such as Finance. These extracted computable policy representations have the potential to open up many new opportunities – from reducing compliance costs through automated checking (particularly where common regulations apply to many) to 'what-if' analyses of proposed policy to automatically identifying gaps/loopholes that undermine Program Integrity.

Acknowledgments

We would like to acknowledge our business developers, offering managers, subject matter experts and accredited fraud investigators that make this project and its evaluation possible: Morten Kristiansen, Conor Cullen, Denisa Moga, Jillian Scalvini, John Davis, Shannon Ware, Mark Gillespie and Mark Goodhart.

Ethics / Broader Impact Statement

Governments must routinely automate policy rules in order to deliver services at population-scale, and this brings opportunities for both good and harm. Consistent application of policy rules at population-scale ensures fairer distribution of healthcare and social care services, as well as defending limited resources from Fraud, Waste and Abuse. (This latter point is often missed, but is critical to ensuring that resources are available when vulnerable people need them). At the same time, defects in the code increase the potential for failure to reimburse healthcare providers for delivering necessary services. Getting from policy, to business requirements, to coded rules is a long, multi-translation process, with error and omission failure modes at every step. Gaps or biases in the original policy may also remain unnoticed through this long journey, only to be discovered at the end when vulnerable people are impacted.

A recent global movement known as '*Rules as Code'* (Mohun et al., 2020) identifies several methods to tackle this. The one our system uses involves extracting rules from policy text via NLP, to minimize translation steps. There is potential for misuse here, should machine-extracted rules be executed blindly, without checking they faithfully represent policy intent. Hence, our system treats human-in-the-loop oversight as an essential, non-optional part of the process. Together - automated extraction plus human oversight have the potential to reduce translation failures, as well as enable discovery of *policy* errors and biases (by facilitating earlier testing and iteration).

Further to this, we believe that effective oversight demands more than a review process or 'AI explainability' add-on. It requires human-understanding and ability-to-correct to be first-class design goals. To this end, we use an ontology to represent extracted rules in a form (Figure 2) that policy-aware users find familiar, understandable and correctable (as well as traceable back to their policy origin). This representation is at the heart of the system. When presented side-by-side with the policy text to a reviewer, both success and failure scenarios are clearly visible. For example, in a failure scenario (ontology does not model the policy well), few rules are extracted, and rule conditions are missing. Here the reviewer can discard the rule, or fill-in the missing information. Whether well-formed or poorly formed, at no point are rules blindly applied to citizen data automatically.

We believe that shifting the focus towards validation of policy and its digital expression will facilitate the production of better-quality, more humane policy.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, Association for Computing Machinery, pages 2623-2631. https://doi.org/10.1145/3292500.3330701

American Medical Association. 2000. *Current procedural terminology: CPT 2001*. American Medical Association, Chicago, IL.

Luigi Assom. 2020. Impact of Global food Trade: Food tractability and sustainability indicators. In *1st Workshop on Knowledge Graphs for Social Good*. Data retrieved from https://knowledgegraphsocialgood.pubpub.org/schedule

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*. 32(1) (Database issue)):D267-70. https://doi.org/10.1093/nar/gkh061

Jeremy Carroll, Dave Reynolds, Ian Dickinson, Chris Dollin, Andy Seaborne, Kevin Wilkinson. 2004. Jena: Implementing the Semantic Web Recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &* Posters, Association for Computing Machinery , pages 74-83. https://doi.org/10.1145/1013367.1013381

Centers for Medicare & Medicaid Services (U.S.). 2020. *Healthcare Common Procedure Coding System (HCPCS)*. Centers for Medicare & Medicaid Services.

Centers for Medicare & Medicaid Services (U.S.). n.d. *Place of Service Codes*. Data retrieved from https://www.cms.gov/Medicare/Coding/place-of-service-codes/Place_of_Service_Code_Set

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, pages 165– 170.

Compare and Comply. n.d. https://cloud.ibm.com/docs/compare-comply

Jinho D. Choi, and Martha Palmer. 2012. Guidelines for the clear style constituent to dependency conversion. *Technical report 01-12*, Institute of Cognitive Science, University of Colorado Boulder, Boulder CO, USA.

Laura Chiticariu, Marina Danilevsky, Yunyao Li, Frederick Reiss, Huaiyu Zhu. 2018 SystemT: Declarative text understanding for enterprise. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 3 (Industry Papers). Association for Computational Linguistics, pages 76-83. http://dx.doi.org/10.18653/v1/N18-3010

Data61. 2019. *Case study on CSIRO's Data61, Australia: Contribution to the OECD TIP Digital and Open Innovation project*. Commonwealth Science and Industrial Research Organisation (CSIRO) Data 61. Data retrieved from https://www.csiro.au/en/News/News-releases/2019/OECD-spotlights-CSIROs-Data61-as-global-blueprint-for-digital-innovation

Deloitte. 2014. Get out of your own way – Unleashing productivity. *Building the Lucky Country: Business imperatives for a prosperous Australia* series. Data retrieved from https://www2.deloitte.com/au/en/pages/building-lucky-country/articles/get-out-of-your-own-way.html

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, pages 4171–4186. http://dx.doi.org/10.18653/v1/N19-1423

Digital Government NZ. 2018. *Better Rules for Government Discovery Report*. Digital Government New Zealand. Data retrieved from https://www.digital.govt.nz/dmsdocument/95-better-rules-for-government-discovery-report/

Mauro Dragoni, Serena Villata, Williams Rizzi, and Guido Governatori. 2016. Combining NLP approaches for rule extraction from legal documents. In *Proceedings of the 1st Workshop on MIning and REasoning with Legal texts (MIREL 2016) collocated with the 29th International Conference on Legal Knowledge and Information Systems*, IOS Press, pages 1–13.

William D. Eggers, Mike Turley, Pankaj Kishnani. 2018. The Regulator's New Toolkit: Technologies and Tactics for Tomorrow's Regulator. *Reducing compliance costs with RegTech Deloitte Insights*. Data retrieved from https://www2.deloitte.com/us/en/insights/industry/publicsector/reducing-compliance-costs-with-regtech.html

European Commission. 2019. *Study on the costs of compliance for the financial sector*. Publications Office of the European Union. Data retrieved from https://op.europa.eu/en/publication-detail/-/publication/4b62e682-4e0f-11ea-aece-01aa75ed71a1

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 326-334.

Graham Greenleaf, Andrew Mowbray and Philip Chung. 2020. Strengthening Development of Rules As Code: *Submission to the OECD's OPSI on Cracking the Code*. Australasian Legal Information Institute (AustLII). http://dx.doi.org/10.2139/ssrn.3638771

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. *spaCy: Industrial-strength Natural Language Processing in Python*. Explosion AI. https://doi.org/10.5281/zenodo.1212303

Iowa Department of Human Services (DHS). 2013. *Dental Services Provider Manual*. Data retrieved from https://dhs.iowa.gov/sites/default/files/Dental.pdf?012520210943

Hossein Joudaki, Arash Rashidian, Behrouz Minaei-Bidgoli, Mahmood Mahmoodi, Bijan Geraili, Mahdi Nasiri and Mohammad Arab. 2015. Using data mining to detect health care fraud and abuse: a review of literature. *Global journal of health science*, 7(1):194–202. https://doi.org/10.5539/gjhs.v7n1p194

Aditya Kalyanpur, Branimir Boguraev, Siddharth Patwardhan, J. William Murdock, et al. 2012. Structured data and inference in DeepQA. *IBM Journal of Research and Development*, 56(3):10. https://doi.org/10.1147/JRD.2012.2188737

Mayank Kejriwal and Peter Szekely. 2017. Knowledge Graphs for Social Good: An Entity-centric Search Engine for the Human Trafficking Domain. In *IEEE Transactions on Big Data*, https://doi.org/10.1109/TBDATA.2017.2763164

Vivek Khetan. 2020. SDGs and Knowledge Graph Extraction from Unstructured text. In *1st Workshop on Knowledge Graphs for Social Good*. Data retrieved from https://knowledgegraphsocialgood.pubpub.org/schedule

Nadzeya Kiyavitskaya, Nicola Zeni, Travis D. Breaux, Annie I. Antón, James R. Cordy, Luisa Mich, and John Mylopoulos. 2008. Automating the Extraction of Rights and Obligations for Regulatory Compliance. In *Proceedings of the 27th International Conference on Conceptual Modeling (ER '08)*. Springer-Verlag, pages 154–168. https://doi.org/10.1007/978-3-540-87877-3_13

KPMG. 2018. There's a revolution coming - embracing the challenge of RegTech 3.0. Data retrieved from https://home.kpmg/content/dam/kpmg/uk/pdf/2018/09/regtech-revolution-coming.pdf

Vanessa Lopez, Valentina Rho, Theodora Brisimi, John Segrave-Daly, Morten Kristiansen and Fabrizio Cucci. 2019. Benefit graph extraction from healthcare policies. *In Proceedings of the International Semantic Web Conference.*

James Mohun, Alex Roberts. 2020. Cracking the code: Rulemaking for humans and machines. *OECD Working Papers on Public Governance*, No. 42, OECD Publishing, Paris, https://doi.org/10.1787/3afe6ba5-en.

NLTK tokeniser, n.d., https://www.nltk.org/api/nltk.tokenize.html

Galia Nordon, Gideon Koren, Varda Shalev, Eric Horvitz, and Kira Radinsky. 2019. Separating wheat from chaff: Joining biomedical knowledge and patient data for repurposing medications.

OpenFisca. .n.d. Homepage, OpenFisca, https://fr.openfisca.org/ (Accessed 27 January 2021)

Bethany Percha and Russ B. Altman. 2018. A global network of biomedical relationships derived from text. *Bioinformatics*, 34(15):2614-2624. https://doi.org/10.1093/bioinformatics/bty114

Colin Puri. 2020. Building A Simple Knowledge Graph with UN Data: Quick Start Example, Common Methodologies, and Tooling. In *1st Workshop on Knowledge Graphs for Social Good.* https://knowledgegraphsocialgood.pubpub.org/schedule

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *In Proceedings of EMNLP/IJCNLP.*

Michael Roth and Mirella Lapata. 2015. Context-aware Frame-Semantic Role Labeling. *Transactions of the Association for Computational Linguistics* (3):449-460 http://dx.doi.org/10.1162/tacl_a_00150

Zheyuan Ryan Shi, Claire Wang and Fei Fang. 2020. *Artificial Intelligence for Social Good: A Survey. Computing Research Repository,* ArXiv, abs/2001.01818.

SBERT Sentence Embedding Models, n.d., https://www.sbert.net/docs/pretrained_models.html

William H. Shrank, Teresa L. Rogstad, and Natasha Parekh. 2019. Waste in the US Health Care System: Estimated Costs and Potential for Savings. *JAMA*, 322(15):1501-1509. https://doi.org/10.1001/jama.2019.13978.

SpaCy Pattern Builder (nd). Accessed January 30 2020 from https://pypi.org/project/spacy-pattern-builder/

World Health Organization. 2010. Health System Financing. The path to universal coverage. Chapter 4: More health for the money. *The World Health Report 2010.* https://www.who.int/whr/2010/en/

Supplementary Material

Appendix A. Evaluation environment

Our experiments were run using multiple microservices representing the modules described in the paper. Kubernetes was used to orchestrate the deployment of containers (microservices) in a cloud infrastructure with approximately 16 CPUs, 16 GB of RAM, and 64 GB of disk space.

The processing time of the policy for state 1 (27 pages) is approximately 74 minutes using the "learned rules" approach, while if using the classifier, the time is reduced to 41 minutes. The "manual rules" approach instead takes 42 minutes without classifier and 23 minutes with the classifier. For the policy of state 2 (35 pages), the times of the "learned rules" approach with and without classifier are 104 minutes and 35 minutes respectively. For the "manual rules" approach the processing time with and without classifier are 70 minutes and 10 minutes respectively.

Appendix B. Definition of metrics

Precision (P) measures the proportion of extracted rules that match the GT. Recall (R) measures the proportion of GT rules correctly extracted. f_1 combines these two. Specifically, they are defined as follows:

$$\boldsymbol{Precision} = \frac{\boldsymbol{n°exact\ matches + n°\ partial\ matches}}{\boldsymbol{n°\ extracted\ rules}}$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{n°exact\ matches + n°\ partial\ matches}}{\boldsymbol{n°\ GT\ rules}}$$

$$\boldsymbol{f_1} = 2 * \frac{\boldsymbol{P * R}}{\boldsymbol{P + R}}$$

For a pair consisting of a ground truth Benefit Rule (BR) and an output BR we calculate a similarity score as follows, assuming each BR is a list of conditions c_i with corresponding values v_i taken out of a set with cardinality C_i.

For every pair of partial matches BR (R_j^{GT}, R_i^E), a similarity score s_{ji} is calculated based on:

$$s_{ji} = \frac{\min(L_i,L_j)}{\max(L_i,L_j)} * \frac{1}{L_j} * \sum_{k=1}^{L_j} score_{c_k}$$

where L_j and L_i correspond to the sizes of R_j^{GT} and R_i^E (i.e., how many conditions each BR consists of), and $\frac{\min(L_i,L_j)}{\max(L_i,L_j)}$ represents a penalizing factor when the sizes of the two BR are not the same (rule length similarity).. The score $score_{c_k}$ for each condition value pair $\{c_k: v_k\}$ is calculated as:

$$0, \quad if\ c_k\ is\ captured\ in\ R_j^{GT}, but\ not\ in\ R_i^E$$

$$1, if\ c_k\ is\ captured\ in\ both\ R_j^{GT}, R_i^E\ and\ v_k is\ the\ same$$

$$0.5 + 0.5 * f_1 * \left(1 - \frac{1}{Ca_k}\right), if\ c_k\ is\ in\ R_j^{GT}, R_i^E\ and\ v_k differ$$

Here, f_1 is the harmonic P-R mean generated by comparing the values of c_k in R_j^{GT} and R_i^E and $\{Ca_k\}$ is the number of semantically compatible candidate values a condition may have in the ontology (i.e., instances of the same type, such as all known medical programs), for datatypes where $Ca_k\ is\ 1$

Appendix C. Semantic patterns - templates

For each PAS tuple containing at least two entities annotated with the ontology model (as classes, instances, properties or datatype), consistent semantic statements (subject, predicate, object) are created to build the knowledge graph. Connections are found across the entities according to their semantic types and the templates described in Table 1. Different templates are executed for each meaningful combination of candidate matches in the PAS.

Table 4: semantic templates to check if entities of the given types can be semantically connected. They consist of parameters to substitute by the candidate entities of the type sought - a class, property, instance or datatype (in between <>) - and variables (preceded by '?') that must bind to an ontological resource. If all the constraints apply (that is semantically consistent), the pattern executes to build new statements, creating anonymous resources (blank nodes) as needed.

Pattern 1: <class, object property, instance>

- <class> is one of the domains of <property>
- <instance> type is on the range of <property>

If consistent, create a blank resource _:b such:

_:b rdf:type <class>. _:b <property> <instance>

- <class> is in the range of <property>
- <class> is the type of <instance> (or a superclass, that is <instance> rdf:type <class>)
- <obj property> has a domain ?domain

If consistent, create a blank resource _:b such:

_:b rdf:type <domain>. _:b <property> <instance>.

Pattern 2: <class, data property, datatype>

- <class> is one of the domains of <property>
- the range of <property> is consistent with the <datatype> (e.g., string, numerical, currency)

If consistent, create an blank resource _:b such:

_:b rdf:type <class>. _b: <property> 'datatype'

Pattern 3: <class, object property, datatype>

- <class> is one of the ranges of <property>
- <class> is the domain of a ?property2, which range is compatible with <datatype>
- <property> has a domain ?domain

If consistent, create two blank resources _:bi such:

_:bx rdf:type <domain>. _:bx <property> :_by.

_:by rdf:type <class >. _:by <property2> <datatype>.

Pattern 5: <instance, data property, datatype>

- the <instance> type is one of the domains of <property>
- the range of <property> is compatible with <datatype>

If consistent, create a new statement such:

<instance> <property> 'datatype'

- <property> has a range consistent with the datatype
- <property> has a domain ?domain
- ?domain is the domain of ?property2, which range ?range is compatible with the type of < instance>

If consistent, create a blank resource _:b such:

_:b1 <property> 'datatype'. _:b1 rdf:type <domain>

_:b1 <property2> <instance>

Pattern 6: <instance1, object property, instance2>

- <property> has <instance2> as value
- <property> has a domain ?domain
- ?domain is the domain of ?property2, which range ?range is compatible with the type of < instance>

If consistent, create a blank resource _:b such:

_:b <property> <instance2>. _:b rdfs:type <domain>.

__:b <property2> <instance2>

Pattern 7: <class1, object property, class2>

- <class1> is the domain of <property>
- <class2> is the range of <property>

If consistent, create a blank resource _:bi such:

_:bx rdf:type <class1>. _:by rdf:type <class2>

:bx <property>:by.

Pattern 8: <class, instance1, instance2>

- <class> is the domain of ?property1 and ?property2
- ?property1 has as range the type of <instance1>
- ?property2 has as range the type of <instance2>

If consistent, create a blank resource _:b1 such:

_:b1 rdf:type <class>. _:b1 <property1> <instance1>.
_:b1<property2> <instance2>

Not Far Away, Not So Close: Sample Efficient Nearest Neighbour Data Augmentation via MiniMax

Ehsan Kamalloo[*†◊] **Mehdi Rezagholizadeh**[*§] **Peyman Passban**[†§] **Ali Ghodsi**[‡¶]
[◊]Department of Computing Science, University of Alberta
[§]Huawei Noah's Ark Lab
[‡]David R. Cheriton School of Computer Science, Univeristy of Waterloo
[¶]Department of Statistics and Actuarial Science, Univeristy of Waterloo
`kamalloo@cs.ualberta.ca`

Abstract

In Natural Language Processing (NLP), finding data augmentation techniques that can produce high-quality *human-interpretable* examples has always been challenging. Recently, leveraging kNN such that augmented examples are retrieved from large repositories of unlabelled sentences has made a step toward interpretable augmentation. Inspired by this paradigm, we introduce *MiniMax-kNN*, a sample efficient data augmentation strategy tailored for Knowledge Distillation (KD). We exploit a semi-supervised approach based on KD to train a model on augmented data. In contrast to existing kNN augmentation techniques that blindly incorporate all samples, our method dynamically selects a subset of augmented samples that maximizes KL-divergence between the teacher and student models. This step aims to extract the most efficient samples to ensure our augmented data covers regions in the input space with maximum loss value. We evaluated our technique on several text classification tasks and demonstrated that MiniMax-kNN consistently outperforms strong baselines. Our results show that MiniMax-kNN requires fewer augmented examples and less computation to achieve superior performance over the state-of-the-art kNN-based augmentation techniques.

1 Introduction

Knowledge distillation (KD) (Buciluǎ et al., 2006; Hinton et al., 2015) has been successful in improving the performance of various NLP tasks such as language modelling (Jiao et al., 2020; Sanh et al., 2019; Turc et al., 2019), machine translation (Tan et al., 2019; Wu et al., 2020), natural language understanding (Passban et al., 2020; Rashid et al., 2021), and multi-task learning (Clark et al., 2019). It aims to transfer the knowledge embedded in a model—called teacher—to another succedent model—called student, without compromising on accuracy (Furlanello et al., 2018).

Data plays a significant role in the success of KD. The importance of data becomes even more crucial when dealing with large teacher models (Lopez-Paz et al., 2015) or managing tasks with small amount of labelled data (Rashid et al., 2020; Nayak et al., 2019). The training objective of KD focuses on minimizing the discrepancy between representations of a teacher model and a student model. However, this might not be the case for regions which are not covered by training data in the input space. Data augmentation comes into play as a natural solution for such circumstances.

Most existing data augmentation techniques are not tailored for KD as the dynamics of teacher and student models are not considered in generating augmented data. Moreover, other model-based data augmentation techniques such as adversarial approaches do not generate interpretable samples for NLP tasks (Du et al., 2021). In this work, inspired by the success of retrieval-based augmentation techniques (Guu et al., 2020; Khandelwal et al., 2020; Du et al., 2021; Kassner and Schütze, 2020), we propose *MiniMax-kNN*, an interpretable data augmentation methodology. Our technique is interleaved with KD training to generate realistically-looking training points. For this purpose, we use a massive external respostiory of unlabelled sentences. In contrast to previous kNN augmentation techniques which naively extract and incorporate k samples, we propose a minimax approach to adapt kNN augmentation to KD and select our augmented samples more efficiently.

Experimental results show that our technique requires significantly fewer samples, reaches the state-of-the-art kNN augmentation technique (Du et al., 2021), and improves generalization to unseen

---

[*] Equal Contribution

[†] Work done while at Huawei Noah's Ark Lab

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3522–3533
August 1–6, 2021.

data.[1]

Our key contributions can be summarized as follows:

- We tailor kNN-based data augmentation for KD via MiniMax to select more impactful augmented samples for training.
- We significantly improve sample efficiency of kNN-based data augmentation.
- We conduct extensive experiments to evaluate our proposed method and manifest that we can maintain the test performance with training on only influential augmented examples.

2 Background

2.1 Data Augmentation in KD

KD (Hinton et al., 2015) is a training method that incorporates the knowledge of a teacher network in training a student network. The teacher can be trained on the same dataset as the student and often provides a suitable approximation of the underlying distribution of data. The training loss of the student using KD is formulated as in Eq. (1).

$$\begin{aligned}
&\mathcal{L} = (1-\lambda)\mathcal{L}_{CE} + \lambda\mathcal{L}_{KD} \\
&\mathcal{L}_{CE} = CE\Big(y, \sigma(z_s(x)\Big) \\
&\mathcal{L}_{KD} = \mathcal{T}^2 KL\Big(\sigma(\frac{z_t(x)}{\mathcal{T}}), \sigma(\frac{z_s(x)}{\mathcal{T}})\Big)
\end{aligned} \tag{1}$$

where z_s and z_t refer to the logits of the student and teacher networks, $\sigma(.)$ is the softmax prediction, CE and KL refer to cross entropy and KL-divergence loss, respectively. λ is a hyperparameter which controls the contribution of the KD loss with respect to the original cross entropy loss, and $\mathcal{T}$ is the temperature parameter which determines the smoothness of the output probability.

Although KD has been shown to be successful in model compression (Buciluǎ et al., 2006) and improving the performance of neural networks (Furlanello et al., 2018), the core prerequisites for effective KD are often overlooked. Lopez-Paz et al. (2015) give a good insight about these conditions using the VC-dimension analysis:

$$O(\frac{|\mathcal{F}_s|_c + |\mathcal{F}_t|_c}{n^\alpha}) + \varepsilon_t + \varepsilon_l \leq O(\frac{|\mathcal{F}_s|_c}{\sqrt{n}}) + \varepsilon_s \tag{2}$$

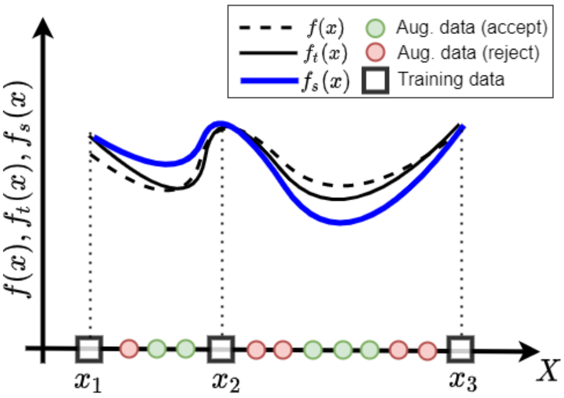

Figure 1: Data sparsity problem in KD; f, f_t, and f_s are representing the underlying function, teacher, and student outputs respectively. We show 10 augmented samples around x_2 with small circles on the X-axis. The green circles show the augmented samples which are selected by our MiniMax-kNN because these points correspond to maximum divergence regions of the teacher and student networks. The red circles are rejected augmented samples.

where $\mathcal{F}_s$ and $\mathcal{F}_t$ are the function classes corresponding to the teacher and student; $|.|_c$ is a function class capacity measure; $O(.)$ is the estimation error of training the learner; ε_s is the approximation error of the best estimator function belonging to the $\mathcal{F}_s$ class with respect to the underlying function; ε_t is a similar approximation error for the teacher with respect to the underlying function; ε_l is the approximation error of the best student function with respect to the teacher function; n is the number of training samples, and $\frac{1}{2} \leq \alpha \leq 1$ is a parameter related to the difficulty of the problem.

According to Eq. (2), it is clear that when the capacity of the teacher is large or when the number of the training samples is small, training with KD can be less beneficial. Figure 1 illustrates this problem through a synthetic example that KD loss forces the student to follow the teacher on training samples but there is no guarantee for such phenomenon to happen in regions in the input space that are not covered by training data. Therefore, the chance of a mismatch between two networks would be higher if training data is sparse or when there is a large gap between two networks.

Data augmentation can be considered as a remedy for this problem. To the best of our knowledge, most existing techniques are not sample efficient and blindly consider all generated samples in their training. As illustrated in Figure 1, different augmented samples might have different contribution to the final teacher/student loss. Moreover, these

---

[1] Source code is available at `https://github.com/ehsk/Minimax-kNN`

augmentation techniques are not tailored for KD. Our MiniMax-kNN solution addresses these two problems.

2.2 Nearest Neighbour Data Augmentation

The kNN augmentation strategy consists of two main stages: (a) a paraphrastic nearest neighbour retrieval engine, and (b) a training method using augmented samples.

Initially, training examples are queried over a large sentence repository using a general-purpose paraphrastic encoder. The aim of this stage is to find interpretable unannotated augmented samples that are semantically close to training data. For this purpose, we use one of the sentence repositories from SentAugment (Du et al., 2021), comprising 100M sentences collected from Common Crawl. We also employ the same paraphrastic sentence encoder, namely SASE, introduced in SentAugment. SASE is an XLM model (Lample and Conneau, 2019), fine-tuned on a number of well-known paraphrase datasets using a triplet loss to maximize the cosine similarity between representations of paraphrases. The similarity between a pair of sentence representations obtained from SASE can be adopted for unsupervised semantic similarity. Du et al. (2021) show that SASE achieves high correlation (0.73 on average) with human judgment on several STS benchmarks. Consequently, the kNN operation can be summarized as follows: Suppose a dataset $\{x_i, y_i\}_{i=1}^{N}$ where x_i and y_i denote an example and its corresponding label respectively. Given a large sentence repository $\mathcal{R}$ encoded using SASE, kNN is determined via top k sentences with respect to $\cos(\text{SASE}(x_i), \text{SASE}(s_j))$ where $s_j \in \mathcal{R}$.

Next, in step (b), a model is trained on the original data by minimizing $\mathcal{L}_{CE}$ from Eq. (1). The trained model learns task-specific knowledge that is further useful in finding relevant augmented examples. To this end, retrieved examples that are close to original examples within the teacher's space are retained to form augmented data. Augmented examples are subsequently incorporated into training via KD. A student model is then distilled from the teacher by leveraging teacher's soft labels on the combination of original data and augmented samples. In particular, for original examples $\{x_i, y_i\}$, $\mathcal{L}$ from Eq. (1) is minimized during training, whereas for the augmented examples, we only minimize $\mathcal{L}_{KD}$.

3 Related Work

3.1 KD in Tandem with Data Augmentation

Adaptive data augmentation can strengthen the capacity of the teacher in transferring knowledge to the student during distillation (Fu et al., 2020). Numerous studies (Chen et al., 2020b; Xie et al., 2020b) have applied KD for self-training in image classification tasks. In NLP, however, generating semantically plausible examples that can be easily inspected by humans is more challenging. In TinyBERT (Jiao et al., 2020), a contextual augmentation method is used along with KD, but such augmentation does not take the advantages of teacher or student's knowledge. A recent paradigm that heavily relies on data augmentation is zero-shot KD (Nayak et al., 2019; Rashid et al., 2020). In contrast, we explore the interpretability of augmentation in KD, which distinguishes our approach from the literature.

3.2 Data Augmentation in NLP

Word-level methods (Zhang et al., 2015; Xie et al., 2017; Wei and Zou, 2019) are heuristic based and do not necessarily yield natural sentences. More recently, contextual augmentations (Kobayashi, 2018; Yi et al., 2021) that substitute words for other words, is shown effective in text classification. However, these approaches do not produce diverse syntactic forms. Similarly, inspired by denoising auto-encoders, augmented examples can be sampled from the reconstruction distribution of corrupted sentences via Masked Language Modelling (Ng et al., 2020). Back-translation (Sennrich et al., 2016) is also another strategy to obtain augmented data (Yu et al., 2018; Xie et al., 2020a; Chen et al., 2020a; Qu et al., 2021).

Another line of work that mainly targets model robustness is to create new data or counterfactual examples via human-in-the-loop perturbations (Kaushik et al., 2020; Khashabi et al., 2020; Jin et al., 2020). Nonetheless, these strategies are task-specific and not scalable to generate data at massive scale. Besides, our method diverges from these studies in that we intend to build a semi-supervised system with minimal human intervention.

Several models (Miyato et al., 2017; Zhu et al., 2020; Jiang et al., 2020; Cheng et al., 2020; Qu et al., 2021) leveraged adversarial training for data augmentation. These methods manipulate the input embedding space to construct synthetic examples. Neighbourhoods around training instances in

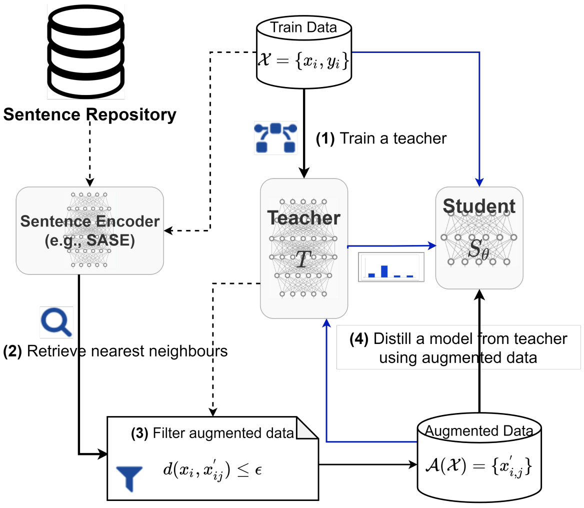

Figure 2: A schematic view of MiniMax-kNN

the embedding space cannot be translated back to text and thus are not interpretable. Although we advocate for interpretable data augmentation, we do not compete with these techniques and in fact, gradient-based augmentation is complementary to our method.

Finally, kNN, a non-parametric search algorithm that probes an external data source to find nearest neighbours is employed in several NLP tasks such as language modelling (Khandelwal et al., 2020), machine translation (Khandelwal et al., 2021), cloze question answering (Kassner and Schütze, 2020), and open-domain question answering (Lewis et al., 2020). kNN offers access to explicit memory that can retrieve factual knowledge from a data store. kNN is highly interpretable as knowledge is stored in raw text, an easy format for humans to understand. Recently, SentAugment (Du et al., 2021) introduced a semi-supervised strategy with unlabelled sentences. It retrieves augmented samples from a universal data store using kNN. Our proposed strategy is in line with SentAugment at heart, but different in leveraging the augmented examples during training. We focus on sample efficiency and show that we can reduce the size of the augmented data—e.g., by 60% in sentiment classification as reported in Section 5.6—while reaching a competitive performance.

4 MiniMax-kNN Data Augmentation for KD

Inspired by Volpi et al. (2018) and Madry et al. (2018), we apply minimax framework to tailor a sample efficient kNN data augmentation for KD. Minimizing the maximum expected risk is used in adversarial training (Volpi et al., 2018) and it is shown to have guaranteed good performance on distributions (P) within a particular distance (ρ) from a source distribution (P_0):

$$\min_{\theta} \sup_{\mathcal{D}(P,P_0)\leq\rho} \mathbf{E}[l(x', y'; \theta)] \tag{3}$$

where $\mathcal{D}$ is a notion of distance between distributions, l refers to the loss function, θ represents the parameters of the estimator model, and in our framework, (x', y') are augmented data samples.

Let us define the set of kNN augmented samples corresponding to the training sample $x_i \in \mathcal{X}$ from the training set, $\mathcal{X}$, to be $\mathcal{A}(x_i) = \{x'_{i1}, x'_{i2}, ..., x'_{ik}\}$. In the maximization phase, we define the loss $l(x', y'; \theta) = \mathrm{KL}\big(T(x'), S(x'; \theta)\big)$ between the softmax output of the teacher $T(x')$ and that of the student network $S(x'; \theta)$ with trainable parameters θ, with respect to the given augmented samples. Note that the augmented samples are unlabelled in the maximization phase. Then, we sort the augmented samples based on their loss value and form our MiniMax-kNN augmentation set $\bar{\mathcal{A}}(x_i)$ by selecting the top n out of the k samples in $\mathcal{A}(x_i)$. n is a hyper-parameter in our method that determines the sample efficiency of MiniMax-kNN. In order to enforce $\mathcal{D}(P, P_0) \leq \rho$ on the distance between the two distributions in Eq. (3), in our kNN search, we set a maximum radial semantic distance ϵ between the sentence representation of accepted augmented samples in $\bar{\mathcal{A}}(x_i)$ and the sentence representation of their corresponding input x_i based on the angular distance metric:

$$d(x_i, x'_{ij}) = \frac{1}{\pi} \cos^{-1} \frac{< h^t_{cls}(x_i).h^t_{cls}(x'_{ij}) >}{\|h^t_{cls}(x_i)\| \|h^t_{cls}(x'_{ij})\|} \leq \epsilon \tag{4}$$

where h^t_{cls} refers to the teacher's last layer hidden representation of the `[CLS]` token, and $<\cdot>$ denotes the dot product of two vectors. The discussion on how to adjust ϵ is given in Section 5.3.

In summary, our technique equips kNN augmentation with minimax to improve its sample efficiency. In contrast to adversarial data augmentation methods, our approach uses the minimax loss for selecting augmented samples. The overall structure of our augmentation strategy is visualized in Figure 2. We essentially follow three steps in each iteration during training:

(1) We construct teacher logits and student logits for augmented samples to measure KL-divergence between the two models.

(2) Out of all kNN samples, n samples with highest KL-divergence will be selected.

(3) KD loss is minimized for training data and selected augmented samples.

Our experiments reveal that this modification to KD underscores sample efficiency while retaining the test performance.

4.1 FLOPs Analysis of MiniMax-kNN

Minimax computations in MiniMax-kNN incur additional overhead costs during training, but how much precisely do minimax operations curtail the runtime performance? To answer this question, we analyze the logical compute complexity of our algorithm in terms of floating point operations (FLOPs) because it can be measured regardless of hardware considerations (Clark et al., 2020).

To this end, we compare FLOPs corresponding to each augmented example from MiniMax-kNN with vanilla kNN within an epoch. Suppose a forward pass and a backward pass for one batch takes F and B FLOPs, respectively. The number of matrix operations between the forward pass and the backward pass is not considerably different and hence, $F \approx B$ (Clark et al., 2020). For simplicity, we assume batch size is 1. Considering k_1 is the number of retrieved NNs, vanilla kNN requires $k_1F + k_1B$ additional FLOPs per epoch.

On the other hand, MiniMax-kNN selects n neighbours from k_2 retrieved nearest neighbours—i.e., $n < k_2$. The algorithm first takes the logits of all k_2 neighbours to compute KL-divergence vectors, which needs k_2F FLOPs, similar to vanilla kNN. The extra operations of MiniMax-kNN occur in the maximization step in which top n neighbours are determined with respect to their KL-divergence values. This operation can be carried out by sorting the KL-divergence vector, which costs S FLOPs. Note that $S \ll F$ because obtaining an output from a deep neural network model is far more costly than a sorting operation. The backward pass is then computed only for the n selected neighbours. Accordingly, the overall FLOPs for MiniMax-kNN is $k_2F + S + nB$. The difference between the FLOPs is:

$$\begin{aligned}\Delta_{\text{FLOPs}} &= \text{FLOPs}_{\text{vanilla-}k\text{NN}} - \text{FLOPs}_{\text{MiniMax-}k\text{NN}} \\ &= (k_1 - k_2)F + (k_1 - n)B + S\end{aligned}$$

Given that B can be approximated with F (as mentioned earlier) and $S \ll F$:

$$\Delta_{\text{FLOPs}} = (2k_1 - k_2 - n)F$$

Dataset	#class.	#train	#dev	#test	avg. #tokens
SST-2	2	67.3K	872	1.8K	12.4
SST-5	5	8.5K	1.1K	2.2K	22.6
TREC	6	5K	500	500	11.4
CR	2	2.5K	640	642	21.4
IMP	2	3.9K	2.2K	2.6K	50.0

Table 1: Downstream tasks used for evaluation

Thus, as long as $k_2 + n < 2k_1$, MiniMax-kNN is more efficient than vanilla kNN. In experiments, we illustrate that how MiniMax-kNN surpasses vanilla-kNN while satisfying the FLOPs condition.

5 Experiments

5.1 Datasets

We evaluate MiniMax-kNN on five datasets: SST-2 and SST-5 (Socher et al., 2013) for sentiment analysis, TREC (Li and Roth, 2002) for question type classification, CR (Hu and Liu, 2004) for product review classification, and Impremium's hate-speech detection dataset (IMP)[2]. Information related to all datasets is summarized in Table 1.

5.2 Experimental Setup

We adopt the publicly available pre-trained RoBERTa$_{\text{Large}}$ (Liu et al., 2019) and DistilRoBERTa (Sanh et al., 2019)—using the Huggingface Transformers library (Wolf et al., 2020) and the Pytorch Lightning library[3]—for evaluating our approach. For KD, RoBERTa$_{\text{Large}}$ is selected as the teacher. For training, we adhere to findings in Mosbach et al. (2021) and Zhang et al. (2021) to circumvent the fine-tuning instability problem by training for longer iterations—i.e., 100 epochs—with early stopping and use Adam optimizer with bias correction. The model is evaluated on the development data at the end of each epoch and the best performing model is chosen for testing. Our learning rate schedule follows a linear decay scheduler with a warm-up on {10%, 20%} of the total number of training steps. The learning rate is tuned for each task separately out of {1e-5, 2e-5, 3e-5}, and the batch size is chosen from [16, 128] depending on the dataset size. For KD hyperparameters, we use grid search to choose the best λ and $\mathcal{T}$ from {0.3, 0.4, 0.5, 0.6} and {5, 10, 12, 20}. We also schedule augmentation to start after a certain number of epochs in the training. On SST-5, SST-2, and

---

[2] `https://www.kaggle.com/c/detecting-insults-in-social-commentary/`

[3] `https://github.com/PyTorchLightning/pytorch-lightning`

Model	**SST-5**	**SST-2**	**TREC**	**CR**	**IMP**
RoBERTa$_{\text{Large}}$ (Teacher)	57.6	96.2	98.0	94.1	90.0
DistilRoBERTa	52.9	93.5	96.0	**92.1**	86.8
DistilRoBERTa + KD	53.2	93.6	96.6	**92.1**	87.7
DistilRoBERTa + vanilla-8NN	55.2	94.7	97.0	91.3	88.4
Aug. Size (#forward / #backward pass)	8x / 8x	8x / 8x	8x / 8x	8x / 8x	8x / 8x
DistilRoBERTa + MiniMax-8NN*	**55.4**	**95.2**	**97.6**	91.6	**88.6**
Aug. Size (#forward / #backward pass)	5x / 4x	7x / 4x	8x / 4x	8x / 2x	8x / 1x

Table 2: Test accuracy (↑) on the downstream tasks (*denotes our approach and **bold** numbers indicate the best result—excluding the teacher—for each task).

Model	**SST-5**	**SST-2**	**TREC**	**CR**	**IMP**
vanilla-8NN	55.2	94.7	97.0	91.3	88.4
$n = 1$	55.4	94.4	96.4	91.4	**88.6**
$n = 2$	54.6	95.0	96.4	91.6	88.5
$n = 4$	55.4	**95.2**	**97.6**	91.6	87.4
$n = 6$	**55.6**	94.4	96.6	**91.8**	87.9

Table 3: Test accuracy (↑) of DistilRoBERTa on the downstream tasks varying the number of selected NNs (n) in MiniMax-8NN (**bold** numbers indicate the best result for each task).

TREC, augmentation takes effect at epochs 8, 6, and 6, respectively, whereas on IMP, and CR, augmentation starts at the beginning of training. All experiments were conducted on two Nvidia Tesla V100 GPUs.

Few-shot learning setup We follow Du et al. (2021) to setup the environment for few-shot learning experiments. In particular, we sample 2 training subsets with replacement from the original training set for each task. Each subset is balanced and consists of 20 examples per label. The development set is reduced to 200 examples for all tasks except CR in which we keep all of the original set. The label distribution is retained in the reduced development data. Evaluation is conducted on the actual test dataset. To obtain reliable results, we repeat training with 10 different seeds on each sampled dataset and report the average across all runs—i.e., 20 runs per task. Few-shot experiments were run on a single Nvidia Tesla V100 GPU.

5.3 MiniMax-kNN Results

First, we investigate the impact of kNN data augmentation at test time and compare MiniMax-kNN with vanilla kNN data augmentation. To this end, we train a RoBERTa$_{\text{Large}}$ as teacher on the original data. Then, we distill a small size student based on DistilRoBERTa from the teacher using the augmented data and the original data.

In Table 2, we report the performance of MiniMax-kNN as well as the vanilla-kNN on the downstream tasks. In this experiment, the number of nearest neighbours (k) is set to 8 and for MiniMax-kNN, we empirically select the minimum number of augmented examples (n) out of 8-NNs such that MiniMax-kNN exceeds vanilla-kNN. We observe that using KD alone leads to a marginal improvement on all tasks. Adding more data results in further improvements but comes at the expense of substantially longer training time. On the other hand, MiniMax-kNN reduces the cost of training as it learns through less than half of the NNs and yet, consistently outperforms vanilla-kNN.

Varying the number of selected examples (n) in MiniMax-kNN We explore the number of selected augmentations by varying $n \in \{1, 2, 4, 6\}$ for 8-NNs on the downstream tasks. Results are reported in Table 3. Interestingly, picking n as small as either 1 or 2 results in superior performance of MiniMax-kNN, compared to vanilla-kNN, on all tasks. In TREC, and SST-2, the sweet spot is $n = 4$. In SST-5, and CR, MiniMax-kNN performs better as n grows. On the contrary, in IMP, accuracy declines by increasing n.

Varying the number of nearest neighbours (k) In order to investigate the optimal number of NNs, we assess the effect of k on the downstream tasks. The Results are reported in Table 4. We observe that more data sometimes makes the training noisy and as a result, performance deteriorates—e.g., $k = 2$ in SST-5 and IMP. Nonetheless, when the augmentation size is sufficiently large, test results improve—i.e., $k = 8$ in all datasets except CR. Apart from three cases—i.e., $k = 2, 4$ in SST-2, and $k = 4$ in CR—MiniMax-kNN is superior to vanilla-kNN by incorporating roughly 50% fewer

Task	KD	$k = 1$	$k = 2$		$k = 4$		$k = 8$	
		vanilla	vanilla	MiniMax	vanilla	MiniMax	vanilla	MiniMax
SST-5	53.2	53.9	52.0	52.5	54.7	55.0	55.2	**55.4**
SST-2	93.6	93.7	94.7	94.2	94.6	93.8	94.7	**95.2**
TREC	96.6	96.2	96.4	96.6	96.8	96.8	97.0	**97.4**
CR	92.1	91.9	92.1	92.2	**92.4**	91.6	91.3	91.9
IMP	87.7	87.2	87.1	87.6	86.0	87.8	88.4	**88.6**

Table 4: Test accuracy (↑) of DistilRoBERTa on the downstream tasks varying the number of nearest neighbours (k). **KD** refers to knowledge distillation with no data augmentation. For MiniMax, n is equal to half of k neighbours for $k = 2, 4$ and when $k = 8$, n is selected as in Table 2 (**bold** and underline indicate best and second best results per task).

Model	SST-5		SST-2		TREC		CR		IMP	
vanilla-8NN	162.1		484.5		78.4		43.7		99.4	
MiniMax-8NN*	158.8	2% ↓	634.4	31% ↑	101.1	29% ↑	30.8	30% ↓	38.6	61% ↓

Table 5: Training time (in seconds) for one epoch (↓), averaged across epochs during training, on the downstream tasks along with the percent of reduction compared to vanilla-8NN. MiniMax-8NN and vanilla-8NN refer to the models we used for Table 2 (*denotes our approach).

examples.

Adjusting the maximum radial distance (ϵ) in MiniMax-kNN We plot the distance distribution of augmented data for two cases: (a) when the teacher predicts the same label as the original examples for augmented ones (*matched labels*) (b) when the predicted label for augmented examples do not match that of original examples (*mismatched labels*). Figure 3 illustrates a clear distinction between these two groups. Considering these insights, we find an empirical heuristic to set ϵ. When the overlap between groups is infinitesimal, we tune ϵ in the vicinity of the maximum distance of *matched labels*. The rationale here is to avoid altering the skewness of the original label distribution. Throughout our experiments, ϵ is set to 0.22, and 0.4 for SST-5, and SST-2, respectively. However, we find $\epsilon = \infty$ works best on CR, IMP, and TREC.

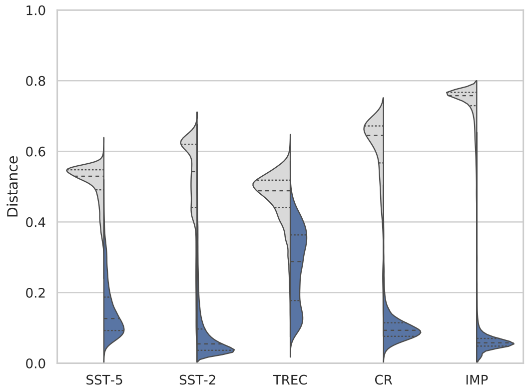

Figure 3: Distance distribution of augmented examples for each dataset (Left: *mismatched labels* / Right: *matched labels*)

5.4 Runtime Efficiency

In §4.1, we showed that MiniMax-kNN is computationally more efficient than vanilla-kNN when $k_2 + n < 2k_1$. Given the number of nearest neighbours is identical ($k_1 = k_2$) in our experiments, any choice of n makes MiniMax-kNN more efficient than vanilla-kNN in theory. However, in our implementation of MiniMax, we feed selected examples again to the student, thereby triggering a redundant forward pass[4]. Although this change reduces the efficiency of MiniMax-kNN in practice, it significantly simplifies the implementation. Thus, the above condition evolves to $k_2 + 2n < 2k_1$ in our experiments. Nonetheless, in Table 2, this new efficiency constraint still holds on all tasks except SST-2, and TREC. To calculate the exact amount of speed-up, we measure the average training time corresponding to one epoch for each task. The results are outlined in Table 5. On IMP, MiniMax-kNN saves more than 60% of training time and on CR, MiniMax-kNN brings almost 30% speed-up. Also, MiniMax-kNN is slightly faster than vanilla-kNN on SST-5. However, MiniMax-kNN trains around 30% slower on SST-2 and TREC.

5.5 Ablation Study

We analyze each component of our augmentation strategy to understand how they impact the overall

---

[4] All augmented examples are initially fed to the student within a PyTorch `no_grad` block. Since we want to backpropagate through only selected examples, they should be fed again to the student.

Model	SST-5	SST-2	TREC	CR	IMP
SIZE	100	40	120	40	40
SentAugment (Du et al., 2021)	44.4 ± 1.0	86.7 ± 2.3	92.1 ± 2.4	89.7 ± 2.0	81.9 ± 1.4
AUG. SIZE	1000	1000	1000	1000	1000
RoBERTa$_{\text{Large}}$ (Teacher)	43.9 ± 2.5	81.1 ± 2.5	89.9 ± 3.5	83.7 ± 2.7	75.5 ± 5.5
RoBERTa$_{\text{Large}}$ + KD	44.8 ± 2.5	82.3 ± 4.4	91.9 ± 2.1	83.9 ± 5.3	81.4 ± 1.9
RoBERTa$_{\text{Large}}$ + vanilla-10NN	45.5 ± 2.3	85.5 ± 2.9	91.6 ± 1.9	88.0 ± 1.5	81.5 ± 3.3
RoBERTa$_{\text{Large}}$ + MiniMax-10NN ($n = 6$)*	46.8 ± 1.4	86.5 ± 1.8	91.8 ± 1.0	88.3 ± 1.5	81.7 ± 2.3
AUG. SIZE (#forward / #backward pass)	1030 / 700	380 / 280	835 / 720	400 / 280	360 / 280

Table 6: Few-shot learning results of MiniMax-kNN on the downstream tasks (*denotes our approach). Compared to SentAugment, our proposed approach achieves competitive performance, but with the use of fewer augmented examples.

#	Model	SST-5	Δ
1	DistilRoBERTa	52.9	-
2	+ KD	53.2	+0.3
3	+ KD + random-8	54.4	+1.2
4	+ KD + random-8 + reranked	54.2	-0.2
5	+ KD + random-8 + reranked + ϵ	52.3	-1.9
6	+ KD + 8NN	54.7	+1.5
7	+ KD + 8NN + reranked (=vanilla)	55.2	+0.5
8	+ vanilla-8NN + ϵ	55.0	-0.2
9	+ MiniMax-8NN + ϵ	55.4	+0.4

Table 7: Ablation study of MiniMax-kNN on SST-5. Δ denotes the performance difference with respect to the previous row, but for the first row in each section, it indicates the difference with the accuracy of KD—i.e., row 2.

effectiveness of MiniMax-kNN. To this end, three components of our strategy are targeted for an ablation study. First, the effect of nearest neighbours is measured by replacing them with random examples from the sentence repository. Then, to determine whether reranking neighbours by teacher is helpful, we preserve the order of nearest neighbours returned by the SASE. Finally, we relax the maximum radial distance to include all nearest neighbours.

In Table 7, we report the results on SST-5. Surprisingly, random augmentation (row 3) scores only 0.3% lower than kNN augmentation (row 6). Reranking nearest neighbours by the teacher further boosts the results by 0.5% (row 7). The presence of maximum radial distance is not helpful for vanilla-kNN as it leads to 0.4% drop in the accuracy (row 8). Finally, our selection mechanism in MiniMax-kNN (row 9) leads to a 0.2% improvement compared to vanilla-kNN (row 7).

5.6 Few-shot experiments

Our data augmentation strategy can be applied to few-shot learning scenarios where a minuscule number of labelled data is available. Therefore, we simulate a few-shot learning setting as described in Section 5.2. In addition to vanilla-kNN and no augmentation baselines, we compare our results with SentAugment (Du et al., 2021), the state-of-the-art method in kNN data augmentation. In SentAugment, experiments are conducted on 5 randomly sampled small datasets and top 3 results of 10 different runs are averaged across sampled datasets, which means average over 15 runs in total. To be comparable to SentAugment, we average across all 10 runs for 2 sampled datasets, average over 20 runs in total, to report our results. In SentAugment, augmented few-shot datasets contain 1000 examples including the original data. For MiniMax-kNN, we use 10-NNs in this experiment with a maximum radial distance.

Table 6 shows the few-shot learning results. The performance of our baselines follows a similar trend in the full-size data experiments. In particular, KD without augmentation slightly improves the test accuracy; Vanilla-kNN brings almost 1.9% improvement on average, and MiniMax-kNN consistently surpasses vanilla-kNN by 0.7% on average. Compared to SentAugment, MiniMax-kNN reaches a competitive performance. The key advantage of MiniMax-kNN lies in sample efficiency. Specifically, MiniMax-kNN falls short by only 0.3% on SST-2, and IMP with using less than 40% of the SentAugment augmented data on average. On CR, MiniMax-kNN lags behind by 1.4%, but again on roughly 40% of the SentAugment data size. Moreover, SentAugment outperforms our approach by 0.3% on TREC, while the size of augmentation is reduced by almost 20%. Lastly, MiniMax-kNN outperforms SentAugment in SST-5 by 2.4% with almost same amount of data.

5.7 Qualitative Analysis

We study the quality of augmented examples retrieved from the sentence repository. Table 8 presents four examples from SST-5, CR, and TREC along with the corresponding top 3-NNs. The top

(i) **SST-5:** this is a stunning film, a one-of-a-kind tour de force.	**very positive**
Here is masterful film-making in action. (5)	*very positive*
It's an expertly-crafted spectacle-event movie. (1)	*very positive*
This is a unique cinematographic experience. (6)	*very positive*
(ii) **CR:** one also exhibited extremely slow speed when going to the menu.	**negative**
No menu appears to make it very quick and easy to use. (15)	*negative*
Switching between options in the main menu is relatively slow. (13)	*negative*
the only niggle i have found is that the menus are a bit slow at times. (8)	*negative*
(iii) **SST-5:** final verdict: you've seen it all before.	**very negative**
Below is the final result. (15)	*neutral*
The final verdict: Go ahead and buy (4)	*positive*
Nut in the end, the final result always pays out. (7)	*positive*
(iv) **TREC:** What causes the body to shiver in cold temperatures?	**DESC**
How is it possible that a higher minimum wage could actually lead to more inequality within a country? (11)	*DESC*
How did the minimum wage increase come about? (13)	*DESC*
How is the new minimum wage hike impacting them? (7)	*DESC*

Table 8: Examples, derived from the augmented CR, SST-5, and TREC, after teacher reranking (the numbers in the bracket indicate the initial rank by SASE). For the nearest neighbours, the teacher's *predictions* are also provided, although soft labels will be used during training. Row (iii) shows an example of label mismatch and row (iv) highlights a mediocre paraphrase retrieval despite matching labels.

two rows show clear-cut examples that the nearest neighbours are in fact paraphrased forms of original samples. Also, the teacher predicts the same label as the original examples for these augmented examples. We observe task-specific knowledge that the teacher has learned from original data helps to rank retrieved sentences—e.g., in the second row, reranking pushes the neighbours at ranks 15 and 13 to the top 3.

However, the augmented data is not always perfect. To identify the limitations of kNN augmentation, we manually inspect 20 samples, randomly drawn from SST-5 and TREC. We find that inaccurate paraphrase retrieval undercuts the quality of augmented examples shown in the bottom rows of Table 8. A side effect of this weakness is the domain mismatch, denoting that augmentation can introduce out-of-domain data. For instance, the input data in TREC is expected to be in interrogative mood, but the retrieval may return declarative sentences. A potential solution to this problem could be utilizing a different repository, entirely comprised of questions in this case, similar to that of Perez et al. (2020). However, curating such repository for specialized domains can be challenging. Moreover, the improvements we observe in the experiments show that this issue is not prevalent in our selected tasks.

6 Conclusion

In this paper, we presented a sample efficient semi-supervised data augmentation technique, namely MiniMax-kNN. The augmentation procedure is framed as finding nearest neighbours from a massive repository of unannotated sentences. The crucial aspect of kNN augmentation is interpretability as augmented examples are written in natural language. We adopt KD to learn from unlabelled data. The key ingredient of our approach is to find the most impactful examples that maximize the KL-divergence between the teacher and the student models. We show that MiniMax-kNN can reduce the augmented data size by 50% while improving upon vanilla augmentation.

Acknowledgments

We thank MindSpore[5]—a new deep learning framework—for partially supporting this work.

References

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, New York, NY, USA. Association for Computing Machinery.

Jiaao Chen, Yuwei Wu, and Diyi Yang. 2020a. Semi-supervised models via data augmentation for classifying interactive affective responses. In *AffCon@ AAAI*.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. 2020b. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 22243–22255. Curran Associates, Inc.

---

[5] `https://www.mindspore.cn`

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, Online. Association for Computational Linguistics.

Jie Fu, Xue Geng, Zhijian Duan, Bohan Zhuang, Xingdi Yuan, Adam Trischler, Jie Lin, Chris Pal, and Hao Dong. 2020. Role-wise data augmentation for knowledge distillation. arXiv:2004.08861.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1607–1616. PMLR.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3929–3938.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8018–8025.

Nora Kassner and Hinrich Schütze. 2020. BERT-kNN: Adding a kNN search component to pretrained language models for better QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3424–3430, Online. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020. More bang for your buck: Natural perturbation for robust question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–170, Online. Association for Computational Linguistics.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7059–7069.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2015. Unifying distillation and privileged information. arXiv:1511.03643.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. 2019. Zero-shot knowledge distillation in deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4743–4751. PMLR.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283, Online. Association for Computational Linguistics.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2020. ALP-KD: Attention-based layer projection for knowledge distillation. arXiv:2012.14022.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, Online. Association for Computational Linguistics.

Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. 2021. CoDA: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. In *International Conference on Learning Representations*.

Ahmad Rashid, Vasileios Lioutas, Abbas Ghaddar, and Mehdi Rezagholizadeh. 2020. Towards zero-shot knowledge distillation for natural language processing. arXiv:2012.15495.

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. 2021. MATE-KD: Masked Adversarial TExt, a companion to knowledge distillation. arXiv:2105.05912.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. arXiv:1908.08962.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. 2018. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yimeng Wu, Peyman Passban, Mehdi Rezagholizadeh, and Qun Liu. 2020. Why skip if you can combine: A simple knowledge distillation technique for intermediate layers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1016–1021, Online. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020a. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020b. Self-training with noisy student improves ImageNet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10687–10698.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. In *International Conference on Learning Representations*.

Mingyang Yi, Lu Hou, Lifeng Shang, Xin Jiang, Qun Liu, and Zhi-Ming Ma. 2021. Reweighting augmented samples by minimizing the maximal expected loss. In *International Conference on Learning Representations*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pages 649–657. Curran Associates, Inc.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. FreeLB: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

It's All in the Heads: Using Attention Heads as a Baseline for Cross-Lingual Transfer in Commonsense Reasoning

Alexey Tikhonov[*]
Yandex
Berlin, Germany
altsoph@gmail.com

Max Ryabinin[*]
Yandex, HSE University
Moscow, Russia
mryabinin0@gmail.com

Abstract

Commonsense reasoning is one of the key problems in natural language processing, but the relative scarcity of labeled data holds back the progress for languages other than English. Pretrained cross-lingual models are a source of powerful language-agnostic representations, yet their inherent reasoning capabilities are still actively studied. In this work, we design a simple approach to commonsense reasoning which trains a linear classifier with weights of multi-head attention as features. To evaluate this approach, we create a multilingual Winograd Schema corpus by processing several datasets from prior work within a standardized pipeline and measure cross-lingual generalization ability in terms of out-of-sample performance. The method performs competitively with recent supervised and unsupervised approaches for commonsense reasoning, even when applied to other languages in a zero-shot manner. Also, we demonstrate that most of the performance is given by the same small subset of attention heads for all studied languages, which provides evidence of universal reasoning capabilities in multilingual encoders.

1 Introduction

Neural networks have achieved remarkable progress in numerous tasks involving natural language, such as machine translation (Bahdanau et al., 2014; Kaplan et al., 2020; Arivazhagan et al., 2019), language modeling (Brown et al., 2020), open-domain dialog systems (Adiwardana et al., 2020; Roller et al., 2020), and general-purpose language understanding (Devlin et al., 2019; He et al., 2021). However, the fundamental problem of commonsense reasoning has proven to be quite challenging for modern methods and arguably remains unsolved up to this day. The tasks that aim to measure reasoning capabilities, such as the Winograd Schema Challenge (Levesque et al., 2012), are deliberately designed not to be easily solved by statistical approaches, which are a foundation of most deep learning methods. Instead, these tasks require implicit knowledge about properties of real-world entities and their relations in order to resolve inherent ambiguities of natural language.

The town councilors refused to give *the demonstrators* a permit because they feared violence.
Answer: The town councilors

Figure 1: Example of a Winograd Schema problem. The resolved pronoun is underlined, two options are highlighted with an italic font.

Figure 1 illustrates the gist of this task: given a sentence and a pronoun (they), the goal is to choose the word that this pronoun refers to from two options (*The town councilors* or *the demonstrators*). While picking the right answer is straightforward for humans, the lack of explicit clues makes it hard for machine learning algorithms to perform better than majority vote or random choice.

Recently large Transformer-based masked language models (MLMs) (Devlin et al., 2019) were shown to achieve impressive results on several benchmark datasets for commonsense reasoning (Sakaguchi et al., 2020; Kocijan et al., 2019; Klein and Nabi, 2020). However, the best-performing methods frequently involve finetuning the entire model on large enough corpora with varying degrees of supervision; apart from providing initial parameter values, the pretrained trained language model is not used for predictions.

Moreover, these methods have mostly been evaluated on English language datasets, despite increasing interest in multilingual evaluation for NLP (Hu et al., 2020) and the existence of multilingual encoders (Conneau et al., 2020; Conneau and Lample, 2019). The XCOPA dataset (Ponti et al., 2020) was recently proposed as a benchmark for multilingual

---

[*] Equal contribution.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3534–3546
August 1–6, 2021.

commonsense reasoning, yet its task is different from the pronoun resolution problem described above. Versions of Winograd Schema Challenge exist in different languages, but each version comes with slight differences in task specification. This makes holistic cross-lingual evaluation of new commonsense reasoning approaches a quite difficult problem for researchers in the area.

In this work, we propose a simple supervised method for commonsense reasoning, which trains a linear classifier on the self-attention weights between the pronoun and two answer options. To evaluate our method and facilitate research in multilingual commonsense reasoning, we aggregate existing Winograd Schema datasets in English, French, Japanese, Russian, Portuguese, and Chinese languages, converting them to a single format with a strict task definition. Our approach performs comparably to supervised and unsupervised baselines in this setting with both multilingual BERT and XLM-R models as backbone encoders.

Moreover, we find that the same set of attention heads can be used to solve reasoning tasks in all languages, which hints at the emergence of language-independent linguistic functions in cross-lingual models and supports the conclusions made by prior work (Chi et al., 2020; Li et al., 2020). Interestingly, when using an unsupervised attention-based method (Klein and Nabi, 2019), we observe that restricting the choice of heads to this set also improves the results of this baseline. This result suggests that the key to improved performance of such approaches might lie in the right choice of heads rather then the exact attention values.

To summarize, our contributions are as follows:

- We offer a simple supervised method to utilize self-attention heads of pretrained language models for commonsense reasoning.
- We compile a multilingual dataset of Winograd schemas in six languages, bringing all tasks to the same format[1]. When evaluated on this dataset, our method performs competitively to strong baselines from prior work.
- We demonstrate that in cross-lingual models, there exists a small subset of attention heads specializing in *universal* commonsense reasoning. This reveals new linguistic properties of masked language models trained on multiple languages.

2 Related work

2.1 Winograd Schema challenges

The Winograd Schema Challenge (WSC) was proposed as a challenging yet practical benchmark for evaluation of machine commonsense reasoning (Levesque et al., 2012). Since its introduction, several English-language benchmarks of varying difficulty and size were also proposed: notable examples include Definite Pronoun Resolution (Rahman and Ng, 2012) and Pronoun Disambiguation Problem (Morgenstern et al., 2016) datasets, as well as WinoGrande, which consists of 44k crowdsourced examples (Sakaguchi et al., 2020). A version of WSC is also included in the popular SuperGLUE language understanding benchmark (Wang et al., 2019a), where it is reformulated as a natural language inference problem.

There also exist variations of WSC in other languages: French (Amsili and Seminck, 2017), Japanese (Shibata et al., 2015), Russian (Shavrina et al., 2020), Portuguese (Melo et al., 2019), and Chinese (Bernard and Han, 2020). We use these datasets in our study to create a multilingual dataset for commonsense reasoning.

Although in general the task definition of Winograd Schema Challenge was formalized to some degree, both succeeding datasets and methods proposed by users of these datasets have introduced various changes to the task specification and even the input format. In particular, a work by Liu et al. (2020) provides a thorough comparison of different ways to formalize the task for WSC and shows that the same model can give widely varying results depending on the evaluation framework. We describe our efforts to convert different datasets to a single format in Section 4.

2.2 Language models applied to commonsense reasoning

Several works attempt to solve Winograd Schema Challenge by utilizing pretrained language models. For example, Trinh and Le (2018) propose to rank possible answers with an ensemble of RNN language models by substituting the pronoun with each of the options. Recently, Klein and Nabi (2019) introduced Maximum Attention Score (MAS) for commonsense reasoning. This method uses the outputs of multi-head attention from each layer and scores each candidate answer based on the number of heads for which this answer has the highest attention value. We use the first (adapted to

---

[1] Our datasets and code are available at `github.com/yandex-research/crosslingual_winograd`

masked language models as proposed by Salazar et al., 2020) and the second approaches as baselines in the experiments. In essence, our method can be compared to MAS, but as we demonstrate in Section 5, several algorithm design differences along with task supervision allow us to significantly improve the commonsense reasoning performance.

Large pretrained Transformer models, such as BERT (Devlin et al., 2019), have also enabled rapid progress of supervised methods for WSC. One such method is given by Sakaguchi et al. (2020): the authors propose to concatenate the sentence and one of the options and to use the [CLS] token representation of the resulting sequence for binary classification. Also, Kocijan et al. (2019) propose a margin-based loss function which aims to increase the log-probability of the correct answer as a replacement for the masked pronoun. We evaluate these methods in our experiments without training on large in-domain datasets; as we show, both methods are prone to overfitting when applied to several hundreds of examples.

2.3 Cross-lingual encoder models

Multilingual representations have been a longstanding goal of the research community: they allow to serve fewer models for a wide range of languages and to improve the results on low-resource languages. Ruder et al. (2019) gives a detailed survey of different cross-lingual word embedding approaches, as well as the history of cross-lingual representations in general.

In this work, we are interested in the latest developments in multilingual Transformer masked language models (Devlin et al., 2019; Conneau and Lample, 2019; Conneau et al., 2020; Siddhant et al., 2020) that were driven by the advances in transfer learning for NLP (Howard and Ruder, 2018; Devlin et al., 2019). In particular, we use pretrained multilingual BERT (mBERT, Devlin et al., 2019) and XLM-RoBERTa (XLM-R, Conneau et al., 2020) for all our experiments.

Recently, there has been increasing interest in the evaluation of multilingual models: as a result, several benchmarks, including XTREME (Hu et al., 2020), XNLI (Conneau et al., 2018) and XCOPA (Ponti et al., 2020) were introduced. Although XCOPA is a commonsense reasoning dataset, it is meant to serve as a multilingual version of the COPA dataset (Roemmele et al., 2011), which offers a problem different from pronoun resolution. In this work, we aimed to create a multilingual counterpart of more widely used Winograd Schema Challenge, so that any future methods for commonsense reasoning can be easily evaluated on languages other than English.

2.4 Functions of Transformer heads

Previous works have demonstrated that it is possible to perform unsupervised zero-shot consistency parsing with attention heads of pretrained cross-lingual models (Kim et al., 2020; Li et al., 2020). In our work, we extend these findings to a conceptually different task of commonsense reasoning. This task has significant overlap with coreference resolution, which was shown to be encoded in specific heads of monolingual BERT (Clark et al., 2019; Tenney et al., 2019).

Motivated by similar results for monolingual models, several works have previously demonstrated that models such as multilingual BERT encode grammatical relations (Chi et al., 2020) and can perform zero-shot entity recognition, as well as POS-tagging (Pires et al., 2019). Besides presenting evidence for universality in pronoun resolution, which was not studied before, our analysis relies on attention heads instead of extracting representations from intermediate layer outputs.

3 Common sense from attention

In this section, we first give a formal definition of the commonsense reasoning task, most commonly encountered in Winograd Schema Challenge and its successors. Then, we provide necessary background information about the Transformer architecture for transfer learning and describe our proposed solution for this task.

3.1 Exact task specification

It is known that commonsense reasoning performance can vary greatly due to changes in task formulation: for example, recent work by Liu et al. (2020) reports improvements of up to 6 points when posing the task as multiple choice instead of binary classification. Thus, as per recommendations from this work and in order to create a unified dataset, we choose the definition of the Winograd Schema problem which is as strict as possible.

The definition is as follows: the system receives a sentence with a pronoun and has to choose the noun (or noun phrase) that this pronoun refers to. For this choice, the system has two options; both of which, along with the pronoun, are always included as substrings of the initial sentence. We intentionally do not restrict the choice of sentence representation or the framing of the task in order to evaluate a diverse range of solutions.

Although the requirements listed above are quite general and intuitive when working with WSC, some of the datasets we employ have samples that do not conform to them. For example, it might be the case that the pronoun occurs at several positions in the sentence without explicit indication of the one to be resolved. For all such examples, we attempt to convert them to standardized instances by hand and drop them only if it is not possible via simple means: otherwise, the right answer to the problem is misspecified. We give a detailed description of our solution in Section 4.2.

3.2 Transformers for sentence representations

Our method heavily relies on the specifics of the Transformer architecture (Vaswani et al., 2017), which has attracted increased interest in NLP recently due to its generation (Raffel et al., 2020; Brown et al., 2020) and transfer learning (Devlin et al., 2019; Liu et al., 2019) capabilities.

This architecture consists of several sequential layers, where each layer contains a feed-forward block and a self-attention block. Inside the self-attention block, there are multiple *attention heads*: each head first linearly projects the input sequence $z = [z_1, \ldots, z_i, \ldots, z_n]$ into sequences of queries q_i, keys k_i and values v_i, then computes the attention weights as softmax-normalized values of pairwise dot products between all keys and all queries:

$$\alpha_{ij} = \frac{\exp(q_i^T k_j)}{\sum_{l=1}^{n} \exp(q_i^T k_l)} \quad (1)$$

These weights are then used to combine the values into a single vector for each input vector, and the layer output is a linear combination of all attention head outputs.

3.3 Our approach

The method proposed in this work uses intermediate outputs of a Transformer masked language model with L layers and H heads in each layer. Given an instance of the Winograd Schema problem, we take the input sentence and mask the pronoun that needs to be resolved. After that, we feed the resulting sentence to the language model and obtain the activations of each self-attention layer as a tensor $L \times H \times T$, where T is the number of tokens that constitute the candidate answer. Here, we can either take the attention from the pronoun to the candidate or vice versa.

After aggregating the attention outputs by computing the mean or the maximum over T, we have two matrices for each of two possible answers, which are then flattened into vectors. Combining these vectors, we obtain an input for the binary classification task with class 0 corresponding to the first answer being correct and class 1 corresponds to the second one. Given a dataset of such inputs, we can train a logistic regression to predict the class from the multi-head attention weights α.

There are several design choices which define the exact implementation of our method. We describe them below; for each design choice, we underline the best-performing option as found by the ablation study in Section 5.4.

Feature combination: With two feature vectors for candidate answers, we can either concatenate them or subtract the vector of the second candidate from the vector of the first one.

Pooling over tokens: As the candidates can have different length, we need to transform the attention outputs to feature vectors of the same size. This can be done by one of two simple forms of aggregation: mean- or max-pooling.

Attention direction: Observe from Equation 1 that in general, $\alpha_{ij} \neq \alpha_{ji}$. To find the optimal configuration, we evaluate both options of either attending to the candidate or the pronoun.

4 Dataset

In this section, we describe our procedure of building a multilingual commonsense reasoning benchmark using Winograd Schema Challenge problems. We create this benchmark by combining several monolingual collections for six languages, each described in previously published works.

We intentionally do not use XCOPA (Ponti et al., 2020) as it is aimed at a different problem: instead of operating at the word level, the task of this dataset is to connect the premise and one of two hypotheses, both of which are complete sentences. Because direct application of attention-based reasoning to sentence-level tasks is a non-trivial research question, we leave it to future work.

4.1 Languages

For the English language, we work with the data from the original WSC task[2] (Levesque et al., 2012), as well as the SuperGLUE benchmark (Wang et al., 2019a) and the Definite Pronoun Resolution dataset (Rahman and Ng, 2012). For French and Japanese, we use datasets published by Amsili and Seminck (2017) and Shibata et al. (2015) respectively. We also include the corresponding part from the Russian SuperGLUE benchmark (Shavrina et al., 2020), a collection of Winograd Schemas in Chinese from the WSC website[3], and the Portuguese version of WSC (Melo et al., 2019) into our multilingual benchmark.

In addition, we attempted to use Mandarinograd (Bernard and Han, 2020) — a Mandarin Chinese version of WSC. However, this dataset contains questions instead of pronouns that need to be resolved. As such, we were unable to incorporate its contents without significantly changing the task.

4.2 Preprocessing and filtering

As the datasets for different languages were released in several different formats, in order to have a unified evaluation framework, we needed to convert them all to the same schema. Unfortunately, due to the differences in task formalization we were unable to convert certain examples without completely changing them; as a result, these examples had to be removed from the dataset. Still, our main priority was to maintain the same task format while keeping as many examples as possible; to this end, we fixed minor annotation inconsistencies by hand wherever possible.

Below we describe the steps of our pipeline. First, several examples had more than two candidate choices, i.e. more than one incorrect option is given. We convert these examples into several binary choice problems and report the original dataset sizes after executing this step. Next, the main issue we faced was that the right answer is not included as a substring of the input sentence. Often this can be explained by missing articles, typos or differences in word capitalization. We attempt to fix all such errors in these cases.

The resulting dataset sizes are listed in Table 1; it can be seen that our conversion pipeline discards approximately 29% of data. In the future, more effort could be directed towards constructing a linguistically diverse, large-scale and balanced multilingual Winograd Schema dataset. Yet, as shown in Section 5, this collection of datasets already allows us to distinguish recent commonsense reasoning models by their performance.

Language	Before	After	Remaining, %
English	2605	2325	89.25
French	214	83	38.79
Japanese	1886	959	50.85
Russian	569	315	55.36
Chinese	18	16	92.28
Portuguese	285	263	88.89
Total	5577	3961	71.02

Table 1: Dataset sizes before and after filtering.

5 Experiments

Below we describe the experimental setup used to evaluate cross-lingual transfer capabilities of different approaches to commonsense reasoning and report the results. Note that we also aim to study the universal reasoning properties of attention heads, and thus we do not evaluate our method on common monolingual Winograd Schema datasets.

5.1 Setup

Models We use multilingual BERT (Devlin et al., 2019) and XLM-R-Large (Conneau et al., 2020), as these models are frequently used in other multilingual evaluation literature. The first model has 12 layers with 12 attention heads each, whereas the second model is a 24-layer Transformer with 16 attention heads on each layer. We do not evaluate XLM-R-Base or multilingual translation encoders (Siddhant et al., 2020) because we take two best-performing models according to the XTREME benchmark (Hu et al., 2020).

For our method, we use an implementation of logistic regression from scikit-learn (Pedregosa et al., 2011) with default hyperparameters as a linear classifier over attention weights.

Evaluation For unsupervised methods, we directly apply each method to each language subset and report the classification accuracy. For supervised methods, we first choose a single language for training and generate random train-validation-test splits, leaving 10% of data both for validation and testing subsets. For each language, we create 5

---

[2] Specifically, the WSC285 version.

[3] `https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSChinese.html`

Model	Train lang	en	fr	ja	ru	zh	pt	Avg
				Unsupervised				
MLM prob. ranking	-	53.6	53.0	52.5	51.8	31.3	50.2	**52.8**
Pseudo-perplexity	-	53.0	54.2	49.5	53.7	56.3	49.4	52.0
MAS	-	52.3	51.8	50.2	52.7	56.3	49.1	51.6
				Supervised				
Kocijan et al. (2019)	en	-	52.5±4.1	51.4±0.8	51.2±1.1	48.8±9.3	51.2±1.5	51.0
	fr	50.9±0.4	-	51.5±0.7	51.3±0.9	56.2±9.9	49.2±1.2	51.8
	ja	51.0±0.7	50.8±2.2	-	50.1±1.0	55.0±6.8	49.9±1.1	51.4
	ru	50.7±0.5	51.3±2.0	51.7±1.0	-	51.2±10.3	51.0±0.3	51.2
	zh	50.9±0.2	50.1±1.6	50.8±0.4	50.5±0.0	-	53.0±1.4	**51.1**
	pt	51.1±0.5	54.0±3.9	51.3±0.6	49.3±0.5	53.8±7.1	-	51.9
Ours	en	-	53.7±1.6	52.2±0.4	60.1±0.4	51.2±4.7	53.9±0.4	**54.2**
	fr	51.7±1.1	-	51.1±1.1	52.2±2.6	53.8±3.1	50.9±1.1	**51.9**
	ja	52.7±0.3	55.4±0.8	-	58.0±1.2	50.0±4.0	51.3±1.3	**53.5**
	ru	55.5±0.4	52.3±2.7	52.3±0.3	-	52.5±5.0	52.0±0.7	**52.9**
	zh	49.8±2.0	48.2±4.5	50.5±1.4	50.3±4.9	-	49.0±1.4	49.6
	pt	54.7±0.5	52.8±3.2	51.7±0.6	57.7±1.2	50.0±4.0	-	**53.4**

Table 2: Results for multilingual BERT, best result is denoted by bold font.

random train-validation splits to estimate the standard deviation of metrics, while keeping the same test set to keep the results comparable. Additionally, we test each trained model for a language on all other languages in a zero-shot setting, reporting averaged performance as well.

5.2 Baselines

To compare our approach with currently popular methods, we also evaluate a wide set of well-performing approaches described in earlier works:

Unsupervised We use three entirely unsupervised baselines inspired by prior work. For the first approach, we replace the pronoun by the number of `[MASK]` tokens equal to the length of each candidate answer and compare the MLM probabilities. For the second approach, we replace the pronoun with each of the answers and rank the candidates by "pseudo-perplexity" (Salazar et al., 2020), inspired by the results of Trinh and Le (2018). Both baselines use normalized scores with respect to the candidate word length.

The third unsupervised baseline is Masked Attention Score (MAS), described in Klein and Nabi (2019). Similarly to our method, this approach relies on attention weights for prediction; however, they are utilized differently and the model is unable to discover an optimal subset of heads.

Supervised First, we evaluated the masked language model finetuning approach suggested by the authors of WinoGrande (Sakaguchi et al., 2020). However, in our experiments there are no additional large-scale datasets; we found that with reference hyperparameters, the authors' implementation quickly overfits the training data for all languages in our relatively small benchmark, achieving less than 50% zero-shot accuracy on average.

In addition, we used the margin-based classification approach described in (Kocijan et al., 2019). This method achieves competitive results and outperforms unsupervised baselines in most setups, so we include it in our comparison.

5.3 Results

The results of our experiments for multilinual BERT and XLM-R-Large are shown in Tables 2 and 3 respectively. It can be seen that despite using only the attention weights as features, our method can outperform unsupervised approaches and performs competitively with a state-of-the-art supervised approach in several setups. Notably, the quality improves significantly when going from BERT to XLM-R: this goes in line with previous work on evaluation of cross-lingual encoders (Hu et al., 2020). At the same time, the quality of our method improves more significantly than of that suggested by Kocijan et al. (2019): this may be explained by a greater parameter count and a higher number of attention heads with more distinct specializations.

5.4 Ablation study

Here we compare several algorithm versions listed in Section 3.3. We train all models on the English part of the dataset and evaluate on all other languages, using validation subset performance as our

Model	Train lang	en	fr	ja	ru	zh	pt	Avg
				Unsupervised				
MLM prob. ranking	-	58.8	56.6	61.7	57.5	56.3	56.7	**59.2**
Pseudo-perplexity	-	58.5	54.2	58.2	59.7	56.3	54.8	58.2
MAS	-	57.2	56.6	53.9	58.1	50.0	53.6	56.2
				Supervised				
Kocijan et al. (2019)	en	-	70.6±2.0	81.4±1.8	74.8±1.1	72.5±5.6	74.3±1.2	**74.7**
	fr	59.6±0.5	-	65.8±0.2	58.9±0.3	56.2±0.0	56.7±0.9	59.4
	ja	70.4±3.8	62.7±2.7	-	66.1±1.9	63.7±5.2	63.7±2.7	65.3
	ru	67.1±4.8	62.2±4.2	69.2±3.2	-	56.2±0.0	61.0±4.3	63.1
	zh	59.2±0.0	57.8±0.0	65.5±0.0	58.7±0.0	-	56.3±0.0	**59.5**
	pt	67.1±3.6	61.2±2.2	69.4±3.1	65.6±3.7	60.0±3.4	-	64.6
Ours	en	-	67.5±1.3	69.1±0.2	70.4±0.5	60.0±3.1	66.8±0.9	66.7
	fr	66.1±0.7	-	63.3±0.8	67.0±1.2	60.0±5.0	61.4±1.2	**63.6**
	ja	70.1±0.4	67.2±1.4	-	72.4±0.6	61.3±2.5	65.9±0.8	**67.4**
	ru	68.7±0.5	65.8±2.7	65.7±0.5	-	63.7±4.7	64.4±0.8	**65.7**
	zh	51.1±8.5	48.2±8.6	52.4±8.8	51.3±10.8	-	50.0±8.1	50.6
	pt	68.9±0.6	67.0±1.6	68.1±0.4	69.5±0.2	63.7±4.7	-	**67.4**

Table 3: Results for XLM-R-Large, best result is denoted by bold font.

Method	Valid	fr	ja	ru	zh	pt	Avg
Ours (Section 3.3)	**55.4**	53.7	52.2	**60.1**	51.2	**53.9**	**54.2**
Concat	53.0	**54.9**	**52.3**	56.3	**53.8**	**53.9**	**54.2**
Max pooling	53.6	**52.3**	52.3	59.9	50.0	51.6	53.2
Attn from pronoun	54.8	53.0	52.2	57.4	47.5	52.9	52.6

Table 4: Ablation study results for models trained on the English subset; the best result is in bold.

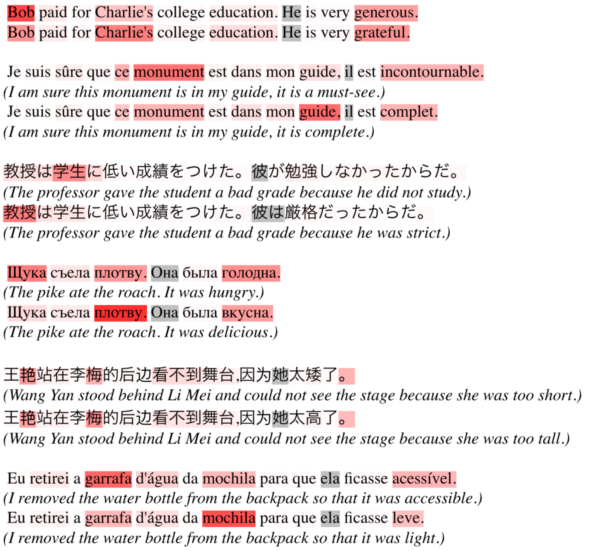

Bob paid for Charlie's college education. He is very generous.
Bob paid for Charlie's college education. He is very grateful.

Je suis sûre que ce monument est dans mon guide, il est incontournable.
(I am sure this monument is in my guide, it is a must-see.)
Je suis sûre que ce monument est dans mon guide, il est complet.
(I am sure this monument is in my guide, it is complete.)

教授は学生に低い成績をつけた。彼が勉強しなかったからだ。
(The professor gave the student a bad grade because he did not study.)
教授は学生に低い成績をつけた。彼は厳格だったからだ。
(The professor gave the student a bad grade because he was strict.)

Щука съела плотву. Она была голодна.
(The pike ate the roach. It was hungry.)
Щука съела плотву. Она была вкусна.
(The pike ate the roach. It was delicious.)

王艳站在李梅的后边看不到舞台,因为她太矮了。
(Wang Yan stood behind Li Mei and could not see the stage because she was too short.)
王艳站在李梅的后边看不到舞台,因为她太高了。
(Wang Yan stood behind Li Mei and could not see the stage because she was too tall.)

Eu retirei a garrafa d'água da mochila para que ela ficasse acessível.
(I removed the water bottle from the backpack so that it was accessible.)
Eu retirei a garrafa d'água da mochila para que ela ficasse leve.
(I removed the water bottle from the backpack so that it was light.)

Figure 2: Averaged attention from the pronoun when using top-5 common heads.

target metric. As the Table 4 demonstrates, each choice leads to drops in performance, with the most influential being the choice of feature concatenation instead of taking the difference and attention direction being the least important decision.

6 Analyzing the attention heads

In this section, we intend to analyze the reasons behind competitive generalization performance of our approach. Mainly we compare the subsets of heads learned on different languages and measure their impact on the prediction quality.

6.1 Universal commonsense reasoning

For the first experiment, we rank the heads for models trained on all languages with the XLM-R[4] representations by the absolute value of the weight. Then, we consider the top-5 heads which are ranked highest on average across all languages. These common heads are located in the higher layers of the model, which was shown previously to encode mainly semantic features (Raganato and Tiedemann, 2018; Jo and Myaeng, 2020), which intuitively corresponds to the tasks the model needs to solve for pronoun resolution. Figure 2 shows the average attention weights of these heads for each word in several example sentences.

After we locate the most important common heads, we train linear classifiers restricted to these heads as features only for every language. To evaluate the importance of head choice, we also provide the performance of linear classifiers trained on a fixed subset of 5 random heads. The results of this experiment can be seen in Table 5; we observe that using the same top-5 heads (only 1.3% of the total number) across all languages preserves or even improves the results. The only exception is Chinese, which might not have enough labeled data to extract a sufficient amount of task-specific in-

---

[4] The results for mBERT are available in Appendix B.

Train lang	Heads	en	fr	ja	ru	zh	pt	Avg
MAS (unsupervised)								
-	All	57.2	56.6	53.9	58.1	50.0	53.6	56.2
	Random	57.8	56.6	56.9	61.6	50.0	56.7	56.6
	Common	**65.8**	**62.7**	**64.9**	**67.3**	**68.8**	**64.3**	**65.6**
Ours (supervised)								
en	All	-	67.5	**69.1**	**70.4**	60.0	**66.8**	**66.7**
	Random		62.0	64.4	67.4	60.0	65.4	63.9
	Common		**68.4**	66.6	68.5	62.5	65.3	66.3
fr	All	66.1	-	63.3	**67.0**	60.0	61.4	63.6
	Random	59.9		58.3	60.7	58.8	57.2	59.0
	Common	**66.7**		**63.8**	66.7	**63.7**	**63.1**	**64.8**
ja	All	**70.1**	**67.2**	-	**72.4**	61.3	**65.9**	**67.4**
	Random	66.0	62.2		68.0	59.4	65.3	64.2
	Common	68.9	66.7		69.5	**62.5**	64.9	66.5
ru	All	**68.7**	**65.8**	65.7	-	**63.7**	64.4	**65.7**
	Random	66.0	62.3	64.3		59.4	**64.6**	63.3
	Common	68.0	64.6	**66.5**		**63.7**	**64.6**	65.5
zh	All	51.1	48.2	52.4	51.3	-	50.0	50.6
	Random	**59.4**	**54.7**	**58.6**	**61.0**		**58.0**	**58.3**
	Common	46.4	47.2	49.4	46.8		46.9	47.4
pt	All	**68.9**	**67.0**	**68.1**	**69.5**	**63.7**	-	**67.4**
	Random	66.2	62.3	64.6	67.1	60.0		64.0
	Common	67.9	65.5	66.0	68.2	**63.7**		66.3

Table 5: Performance of models trained with different subsets of XLM-R-Large attention heads.

formation. It means that a very small subset of attention weights is required to perform commonsense reasoning in all evaluated languages. This further supports the previous results on the analysis of linguistic universals in cross-lingual models (Chi et al., 2020; Wang et al., 2019b).

Moreover, restricting the subset of heads used in the MAS baseline to those selected by the classifiers significantly improves the quality of this unsupervised method as well, nearly closing the gap with the results obtained with supervision. This leads us to the conclusion that initially the poor performance of MAS might be caused by the suboptimal choice of attention heads; when the right heads are selected, their weights do not impact the predictions as significantly. Future unsupervised methods for commonsense reasoning can use that information to pay more attention to the choice of heads, which is currently a less explored subject.

6.2 The impact of number of heads

In this experiment, we directly study the connection between the number of heads and the quality of predictions. Specifically, after training a model with a full set of attention heads, we order them by the absolute value. Then, we retrain the model while keeping only the top-N important heads.

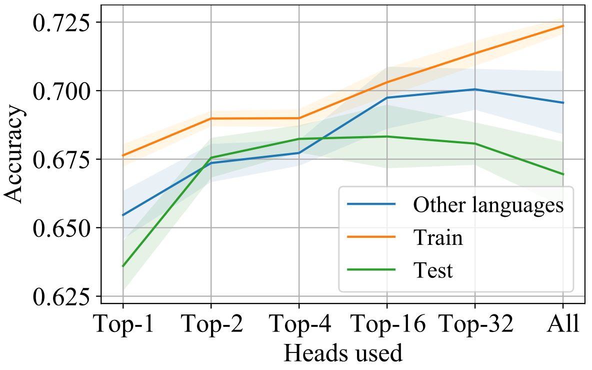

Figure 3: Effect of the number of XLM-R attention heads used when training on English data. Shaded areas show standard deviation across runs.

Figure 3 displays the results of our study for the English language; results for other languages can be seen in Appendix C. From these results, we find that although the training accuracy monotonically increases with the number of used attention weights, the optimal amount of heads for cross-lingual generalization is approximately equal to 16. This number is optimal or near-optimal for other languages as well, which might mean that as the number of features grows, the model either simply overfits the data or starts relying on features that are not universal for all languages.

7 Conclusion

In this work, we offer a simple supervised method to utilize pretrained language models for commonsense reasoning. It relies only on the outputs of self-attention and outperforms complete finetuning in a zero-shot scenario.

We also create a multilingual dataset of Winograd schemas that contains tasks from English, French, Japanese, Russian, Chinese, and Portuguese languages with the same specification. We want to encourage research on commonsense reasoning in languages other than English and release our benchmark to facilitate the development and analysis of new methods for this problem.

Lastly, we demonstrate that the reasoning capabilities of cross-lingual models are concentrated in a small subset of attention heads located in higher layers of the model. Furthermore, this subset of heads is language-agnostic, which sheds light at another facet of linguistic universals encoded in models such as multilingual BERT and XLM-R.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot.

Pascal Amsili and Olga Seminck. 2017. A Google-proof collection of French Winograd schemas. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017)*, pages 24–29, Valencia, Spain. Association for Computational Linguistics.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Timothée Bernard and Ting Han. 2020. Mandarinograd: A Chinese collection of Winograd schemas. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 21–26, Marseille, France. European Language Resources Association.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 328–339.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multitask benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Jae-young Jo and Sung-Hyon Myaeng. 2020. Roles and utilization of attention heads in transformer-based neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3404–3417, Online. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Taeuk Kim, Bowen Li, and Sang goo Lee. 2020. Chart-based zero-shot constituency parsing on multiple languages.

Tassilo Klein and Moin Nabi. 2019. Attention is (not) all you need for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4831–4836, Florence, Italy. Association for Computational Linguistics.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7517–7523, Online. Association for Computational Linguistics.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for the Winograd schema challenge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4837–4842, Florence, Italy. Association for Computational Linguistics.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Bowen Li, Taeuk Kim, Reinald Kim Amplayo, and Frank Keller. 2020. Heads-up! unsupervised constituency parsing via self-attention heads. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 409–424, Suzhou, China. Association for Computational Linguistics.

Haokun Liu, William Huang, Dhara Mungra, and Samuel R. Bowman. 2020. Precise task formalization matters in Winograd schema evaluations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8275–8280, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Gabriela Melo, Vinicius Imaizumi, and Fábio Cozman. 2019. Winograd schemas in portuguese. In *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*, pages 787–798. SBC.

Leora Morgenstern, Ernest Davis, and Charles Ortiz. 2016. Planning, executing, and evaluating the winograd schema challenge. *AI Magazine*, 37:50–54.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789, Jeju Island, Korea. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M. Smith, Y-Lan Boureau, and Jason Weston. 2020. Recipes for building an open-domain chatbot.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Tatiana Shavrina, Alena Fenogenova, Emelyanov Anton, Denis Shevelev, Ekaterina Artemova, Valentin Malykh, Vladislav Mikhailov, Maria Tikhonova, Andrey Chertok, and Andrey Evlampiev. 2020. RussianSuperGLUE: A Russian language understanding evaluation benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4717–4726, Online. Association for Computational Linguistics.

Tomohide Shibata, Shotaro Obama, and Sadao Kurohashi. 2015. Building and analyzing the winograd schema challenge. *Proceedings of the Twenty-first Annual Meeting of the Association for Natural Language Processing*, 57(1):22–23.

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2827–2835, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems.

Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu, and Ting Liu. 2019b. Cross-lingual BERT transformation for zero-shot dependency parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5721–5727, Hong Kong, China. Association for Computational Linguistics.

A In-language metrics for supervised methods

Here, we provide the metrics of our method and the finetuning baseline described by Kocijan et al. (2019) that were obtained on the training, validation and test data for the same language that the models were trained. Tables 6 and 7 demonstrate the results: it can be seen that although our approach performs less well on the same language that was used for training, the issue of overfitting on train data is less noticeable, which might be the reason for better zero-shot metrics.

Model	Train lang	Train	Test
Kocijan et al. (2019)	en	100.0±0.0	47.8±2.3
	fr	100.0±0.0	44.4±7.9
	ja	100.0±0.0	46.9±1.0
	ru	100.0±0.0	52.5±1.4
	zh	100.0±0.0	20.0±44.7
	pt	100.0±0.0	49.6±6.2
Ours	en	57.5±0.3	52.7±1.1
	fr	54.5±3.2	33.3±17.2
	ja	54.1±0.6	49.6±3.1
	ru	60.8±0.9	46.2±2.3
	zh	61.7±8.5	20.0±24.5
	pt	57.1±0.4	43.0±6.9

Table 6: Train and test set metrics for supervised methods, multilingual BERT.

Model	Train lang	Train	Test
Kocijan et al. (2019)	en	100.0±0.0	83.3±1.0
	fr	100.0±0.0	44.4±11.1
	ja	100.0±0.0	79.6±2.2
	ru	100.0±0.0	60.0±3.4
	zh	100.0±0.0	50.0±0.0
	pt	100.0±0.0	55.6±5.2
Ours	en	71.6±0.4	67.2±1.2
	fr	67.7±1.4	37.8±5.4
	ja	71.2±0.2	65.6±1.7
	ru	71.4±0.9	58.1±1.5
	zh	73.3±3.3	20.0±24.5
	pt	67.1±0.8	68.1±5.0

Table 7: Train and test set metrics for supervised methods, XLM-R Large.

B Analysis of common heads for multilingual BERT

Table 8 shows the evaluation results of models using top-5 attention heads of multilingual BERT. It can be seen that leaving only 5 heads out of 144 improves average accuracy in all cases and per-language accuracy in 18/30 cases without any significant decreases in quality.

C Impact of number of heads for other languages

In this section, we analyze the changes in both supervised and zero-shot performance for our method that follow from changes in the number of used attention heads. Figure 4 displays the results for French, Japanese, Russian, and Portuguese language; we omit the results for the Chinese language due to high variance from the small training dataset size. From this figure, we observe the same trend: increasing the number of used heads past 16 can favorably affect the accuracy on the training set, but negatively impacts the resulting quality both for the test set and for other languages.

Train lang	Heads	en	fr	ja	ru	zh	Avg
MAS (unsupervised)							
-	All	52.21	51.81	50.16	52.70	**56.25**	52.63
	Common	**56.60**	**53.01**	**51.82**	**60.00**	50.00	**54.29**
Ours (supervised)							
en	All	-	53.33	52.05	58.92	52.88	54.29
	Common	-	**54.53**	**52.52**	**59.78**	52.25	**54.77**
fr	All	50.76	-	**50.41**	**51.80**	50.06	50.76
	Common	**51.01**	-	50.38	51.73	**50.62**	**50.94**
ja	All	53.25	**52.64**	-	57.48	**50.69**	53.51
	Common	**55.54**	51.84	-	**58.51**	50.56	**54.12**
ru	All	55.43	52.65	**52.00**	-	49.62	52.43
	Common	**56.20**	**52.92**	51.66	-	**49.75**	**52.63**
zh	All	50.28	50.12	50.09	50.53	-	50.25
	Common	**50.82**	**50.14**	**50.24**	**51.30**	-	**50.62**

Table 8: Performance of models trained with different sets of multilingual BERT attention heads.

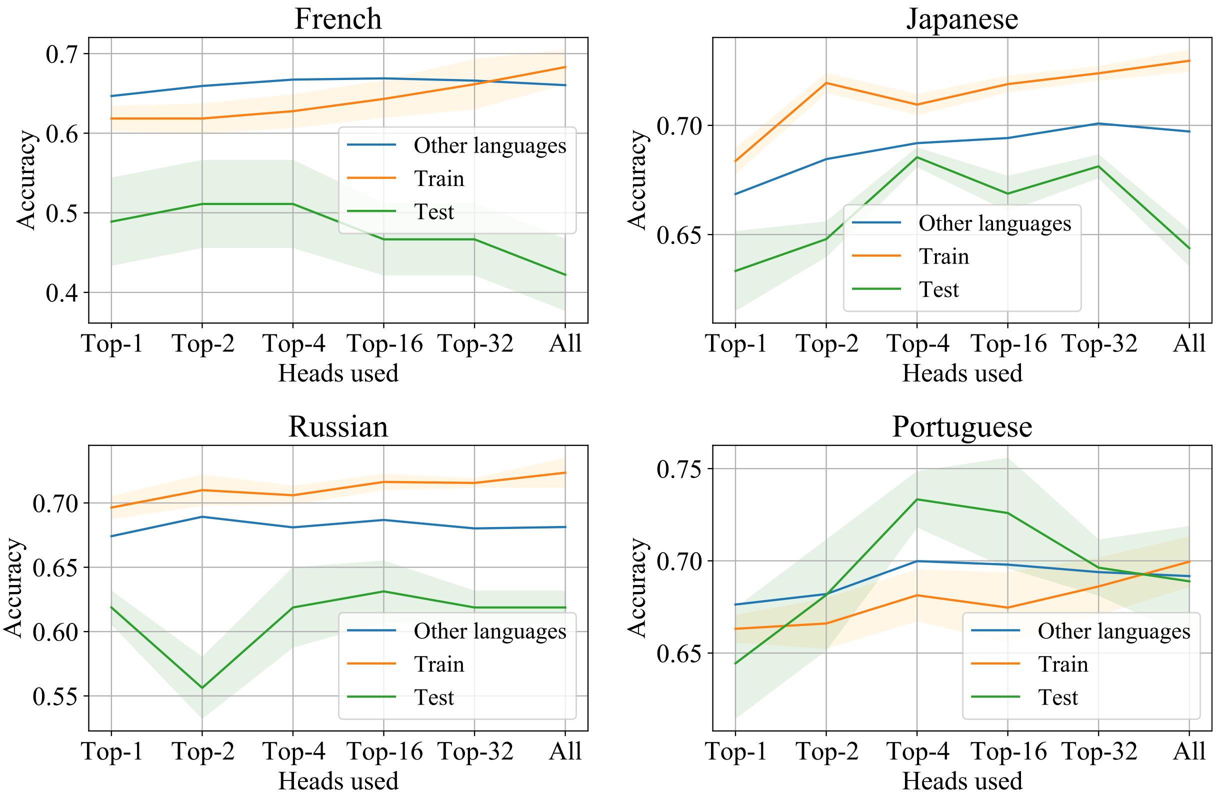

Figure 4: Effect of the number of used XLM-R attention heads on commonsense reasoning performance.

Biomedical Interpretable Entity Representations

Diego Garcia-Olano[1,2] **Yasumasa Onoe**[2] **Ioana Baldini**[1]
Joydeep Ghosh[2] **Byron C. Wallace**[3] **Kush R. Varshney**[1]
[1]IBM Research, [2]University of Texas at Austin, [3]Northeastern University
diegoolano@gmail.com, yasumasa@utexas.edu, ioana@us.ibm.com,
ghosh@ece.utexas.edu, b.wallace@northeastern.edu, krvarshn@us.ibm.com

Abstract

Pre-trained language models induce dense entity representations that offer strong performance on entity-centric NLP tasks, but such representations are not immediately interpretable. This can be a barrier to model uptake in important domains such as biomedicine. There has been recent work on general interpretable representation learning (Onoe and Durrett, 2020), but these domain-agnostic representations do not readily transfer to the important domain of biomedicine. In this paper, we create a new entity type system and training set from a large corpus of biomedical texts by mapping entities to concepts in a medical ontology, and from these to Wikipedia pages whose categories are our types. From this mapping we derive *Biomedical Interpretable Entity Representations* (BIERs), in which dimensions correspond to fine-grained entity types, and values are predicted probabilities that a given entity is of the corresponding type. We propose a novel method that exploits BIER's final sparse and intermediate dense representations to facilitate model and entity type debugging. We show that BIERs achieve strong performance in biomedical tasks including named entity disambiguation and entity label classification, and we provide error analysis to highlight the utility of their interpretability, particularly in low-supervision settings. Finally, we provide our induced 68K biomedical type system, the corresponding 37 million triples of derived data used to train BIER models and our best performing model.

1 Introduction

In modern NLP systems, entities are embedded in the same dense vector space as words using vectors from pre-trained (masked) language models (Devlin et al., 2019) that yield contextualized embeddings of tokens. These representations are used as inputs for downstream models built for particular tasks. One issue with such learned representations is that we do not actually know what information they encode. Recent work has shown that deep pre-trained models implicitly learn factual knowledge about entities (Petroni et al., 2019; Roberts et al., 2020), but the embeddings that they provide do not explicitly maintain representations of this knowledge (i.e., the dimensions in learned representations have no *a priori* semantics); consequently, are not directly interpretable. This has motivated the design of knowledge *probing tasks* to measure a factual knowledge implicit in embeddings (Petroni et al., 2019; Poerner et al., 2019).

Recent work (Onoe and Durrett, 2020) has proposed learning interpretable entity representations using an entity typing model and corresponding fine-grained type system that accepts an entity mention and its context. The output represents a high-dimensional sparse embedding whose values correspond to the model's (independently) predicted probabilities that the entity possesses the respective properties defined by the fine-grained type system.

This past work proposed general domain pre-trained Transformer-based (Vaswani et al., 2017) entity typing models trained on Wikipedia or the ultra-fine entity typing system (Choi et al., 2018), yielding 60k and 10k dimensional embeddings, respectively, which can then be used directly in downstream tasks. Such representations can achieve strong results without learning task specific representations. Thus, in addition to providing interpretability, such representations may be particularly useful for tasks with limited supervision.

Such interpretable entity representations for text can be valuable in domains such as biomedicine, because they afford model transparency which may help with model debugging, or simply to instill confidence in model outputs. For example, if one defines a linear layer on top of entity-type representations, learned coefficients are interpretable as

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3547–3561
August 1–6, 2021.

weights assigned to specific entity types. One could debug an incorrect prediction by inspecting the induced representation for potentially erroneous types assigned to it. This sort of insight is particularly important in biomedical NLP, given the potential sensitivity of the tasks in the domain, and the high-level expertise of the 'end-users'.

Motivated by these observations, we extend (Onoe and Durrett, 2020) to learn sparse Biomedical Interpretable Entity Representations (BIERs) in which values encode predicted probabilities of an entity belonging to a type from a fine-grained entity type system. Starting from a corpus of PubMed[1] articles on cancer and drugs as our training data, we induce an entity type system by mapping entities in the articles to their associated UMLS concepts, and then mapping the concepts to Wikipedia pages whose categories we use as our types.

We show that learning a typing model on top of such a system realizes strong performance on a variety of biomedical tasks including named entity disambiguation (NED) and entity label classification using simple cosine similarity or Euclidean distance based methods, and we provide an analysis of the results from an interpretabilty perspective. In addition, we propose a simple technique that facilitates debugging and provides a mechanism by which to improve model performance by exploiting both the proposed sparse interpretable type representations and their internal underlying dense counterparts. Finally, we introduce and release a new medical-centric Wikipedia dataset based on (Rosenthal et al., 2019) for use in the task of biomedical NED.

Our specific contributions[2] are as follows:

- We create (and will release) a biomedical entity typing system comprising Wikipedia Categories from pages mapped to UMLS concepts linked to PubMed article entities and learn a model that produces sparse entity representations in which dimensions are imbued with known semantics. We show that these achieve strong performance on biomedical NED and entity label classification tasks.

- We conduct an interpretability analysis and demonstrate a new debugging method that uses the proposed representation's performance on downstream tasks to gain insights into the entity typing model and system.

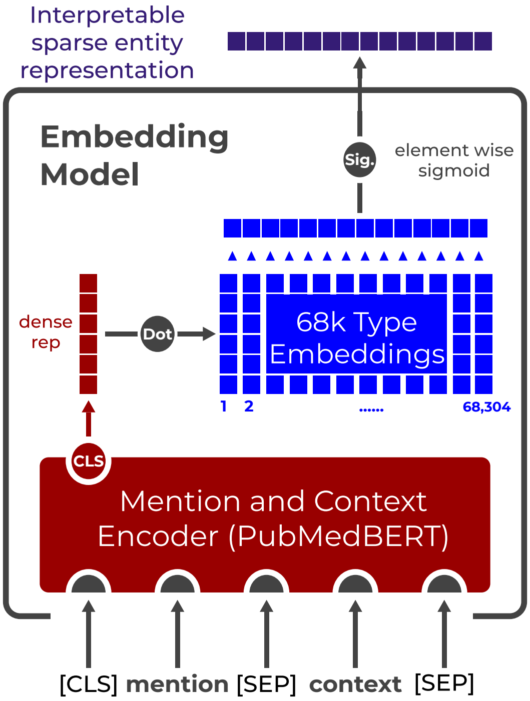

Figure 1: Model architecture from (Onoe and Durrett, 2019) using our 68k biomedical entity type system. A BERT based encoder embeds a mention and context and the output entity embedding contains probabilities for each type.

- We release a medical literature centric Wikipedia dataset for use in the task of biomedical NED.

2 Background: Interpretable Entity Model

We first review the interpretable entity model architecture we extend from (Onoe and Durrett, 2020).

Let $s = (w_1, ..., w_N)$ denote a sequence of input context words, $m = (w_i, ..., w_j)$ denote an entity mention span in s, and $\mathbf{t} \in [0, 1]^{|\mathcal{T}|}$ denote a vector whose values are predicted probabilities corresponding to fine-grained entity types $\mathcal{T}$ from a predefined type system with higher values identifying types most pertaining to m and s .

Given a labeled dataset $\mathcal{D} = \{(m, s, \mathbf{t}^*)^{(1)}, ..., (m, s, \mathbf{t}^*)^{(k)}\}$ the objective is to learn parameters θ of a function f_θ that maps the mention m and its context s to a vector $\mathbf{t}$ that captures salient features of the entity mention within its context. The basic idea is that the

---

[1] A repository of biomedical literature: `http://www.pubmed.gov/`.

[2] Code and datasets available at `http://github.com/diegoolano/biomedical_interpretable_entity_representations`

resultant entity embeddings $\mathbf{t}$ (wherein individual dimensions have explicit semantics) can be used as embeddings in downstream tasks, for example by using basic similarity measures such as dot products or cosine similarities.[3]

The simple model f_θ that produces these embeddings is shown in Figure 1. First, a BERT-based encoder (Devlin et al., 2019) maps inputs m and s to an intermediate dense vector representation. Specifically, the encoder takes as input a token sequence formatted as $\mathbf{x} = \texttt{[CLS]}\ m\ \texttt{[SEP]}\ s\ \texttt{[SEP]}$, where the mention m and context s are segmented into WordPiece tokens (Wu et al., 2016). The hidden vector output corresponding to the `[CLS]` token can be treated as the intermediate dense mention and context representation: $\mathbf{h}_{\texttt{[CLS]}} = \textsc{BertEncoder}(\mathbf{x})$.

A type embedding layer then projects this intermediate representation to a vector whose dimensions correspond to the entity types $\mathcal{T}$ using a single linear layer whose parameters may be viewed as a matrix of type embeddings $\mathbf{E} \in \mathbb{R}^{|\mathcal{T}| \times d}$, where d is the dimension of the mention and context representation $\mathbf{h}_{\texttt{[CLS]}}$. Finally, we apply a sigmoid function to each unnormalized score in the vector to obtain the predicted probabilities that form our entity representation $\mathbf{t}$ (top of Figure 1). We obtain these output probabilities $\mathbf{t}$ by multiplying $\mathbf{E}$ by $\mathbf{h}_{\texttt{[CLS]}}$, followed by an element-wise sigmoid function: $\mathbf{t} = \sigma\left(\mathbf{E} \cdot \mathbf{h}_{\texttt{[CLS]}}\right)$

Following Choi et al. (2018), the training loss we minimize is a sum of binary cross-entropy losses over all entity types $\mathcal{T}$ over all training examples $\mathcal{D}$. That is, we treat each type prediction for each example as an independent binary decision, with shared parameters in the BERT encoder. Our loss $\mathcal{L}$ is:

$$-\sum_i \sum_j t^*_{ij} \cdot \log(t_{ij}) + (1 - t^*_{ij}) \cdot \log(1 - t_{ij}),$$

where i are the data indices, j are indices over types, t_{ij} is the jth component of $\mathbf{t_i}$, and t^*_{ij} is the jth component of $\mathbf{t_i}^*$ that takes the value 1 if the jth type applies to the current entity mention. We fine-tune all parameters in BERT and the type embedding matrix.

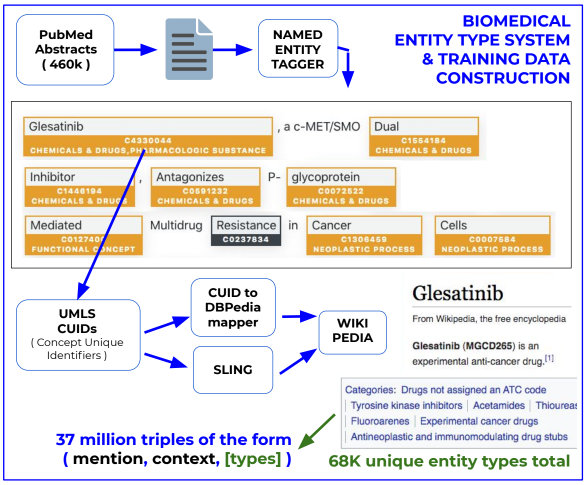

Figure 2: Biomedical Entity Type System and dataset construction. Appendix Fig 4 contains example output.

3 Biomedical Interpretable Entity Representations

Biomedical Entity Typing To train an interpretable entity embedding model tailored specifically for biomedical tasks, we must first construct a suitable biomedical entity type system and dataset. PubMed indexes over 30 million biomedical citations across a wide range of topics. To curate a topically focused set of literature, we first used the PubTator tool (Wei et al., 2019) to query PubMed for articles related to drugs used as treatment for cancer; this yielded 461,404 unique citations (titles and abstracts).[4]

We used an off the shelf NER tagger available in `SciSpacy` (Neumann et al., 2019) to identify entity spans within abstracts, and used the `Entity Linker` component to link those entities to concept unique IDs (CUIDs) within the Unified Medical Language System (UMLS) ontology[5].

Next we had to decide on the specific entity type system to use, i.e., the set of labels to attach to entities, and chose Wikipedia as our knowledge base. We used this general knowledge base instead of a specialized ontology (for example, MeSH or SNOWMED CT) primarily because it yielded (many) more diverse entity types per mention, comparatively.

To connect UMLS concepts to Wikpedia pages

---

[3] Fine-tuning the representations would destroy their interpretability.

[4] We selected the topic of cancer because our work is motivated by a larger project aimed at finding existing evidence that supports repurposing generic drugs for cancer.

[5] UMLS defines around 3 million concepts from a combined 200 source ontologies. Concepts may be identified as having one or more of 127 semantic types which can be used to place them into groupings such as diseases or drugs.

we use the mapping from Cuzzola et al. (2018), which is accurate but incomplete: It provides exact wikipage matches for 221,690 concepts and "close matches" for 26,276 of them, out of a possible 3 million concepts in UMLS. For concepts for which no exact or close match was found, we used SLING (Ringgaard et al., 2017), a framework for frame semantic parsing which allows for querying and resolving wikipages given a search string (in our case, mention surface forms). For high confidence exact or close matches, we return the set of categories found for their combined results. While these results can be slightly noisy, they mostly lead to satisfactory performance.

We filter the entity mentions that compose our final set, as follows. If multiple concept CUIDs are found for a given entity, we include the highest scoring matches within two points of each other provided they all exceed a minimum score threshold of 0.8;[6] Additionally, we only include results that are linked to at least one concept CUID and where an associated Wiki link was mapped to directly via Cuzzola et al. (2018) or via SLING. A schematic of this process is shown in Figure 2. An example working through the entity filtering process is shown in the text of Appendix A. In the end about 12.5% of the mappings from PubMed mentions to Wikipedia categories come via SLING.

After processing, linking and filtering the corpus of PubMed abstracts, we were able to extract 37,357,141 triples of the form (mention, context, [list of categories]). This list of triples contains 68,304 unique categories which we use as the entity type system for training BIERs. Appendix 8 contains a list of the top 100 entity types that appear over these articles and Appendix 5 shows a histogram of entity types per mention. As one contribution, we will release this set of derived triples.

To assess the quality of this dataset, we chose 500 triples at random and asked 4 experts (researchers in biomedicine and ML) to score them on a Likert scale from 1 (low) to 5 (high) for accuracy. Experts assessed how well a PubMed mention from a context sentence maps onto a Wikipedia URL. Average expert scores for the triples were [4.01, 4.13, 4.18, 4.20] (overall mean of 4.13) out of 5. The Fleiss-Kappa score which measures inter-annotator agreement was strong at .69. Additionally 77% of scores are $>=4$, and for 93% of the examples at least 3/4 experts agree (73% have unanimous agreement).

BIER entity typing model training and test results We split our derived dataset of biomedical triples into train, validation, and test sets of sizes 31,340,000, 376,071, and 5,641,070, respectively. For comparison, the total data size used by Onoe and Durrett (2020) is 6.1 million and based on the most popular categories of Wikipedia whereas ours only uses categories on pages linked to UMLS.

We trained different BIER models using variants of BERT as an encoder for mentions and contexts. Specifically we considered BioBERT (Lee et al., 2019), SciBERT (Beltagy et al., 2019) and BLURB (Gu et al., 2020) (we will refer to this as PubMedBERT), which constitute the current state of the art for many biomedical tasks.We compute entity typing macro F1 using development examples to check model convergence and use the hyperparameters from Onoe and Durrett (2020).

Debugging BIERs by combining dense and sparse embeddings We propose a technique for debugging using BIER representations that is in part inspired by prior work that used intermediate layer representations of training examples as additional features (Papernot and McDaniel, 2018). Specifically, we propose to debug BIER performance on downstream tasks by examining instances where dense and sparse representations yield different outputs. For each example, BIER models produce an intermediate dense $\mathbf{h}_{\texttt{[CLS]}}$ and interpretable sparse output embeddings t (red and purple, respectively, in Figure 1). We will refer to the two seperate models which use these dense and sparse BIERs embeddings for downstream tasks as `BIERDense` and `BIERSparse` respectively.

After performing inference initially, we gather all test examples where the `BIERDense` makes a correct prediction but `BIERSparse` does not and we place their mention values into a set $\mathcal{Z}$. Additionally, as a diagnostic measure, we consider an 'oracle' approach in which we use the `BIERDense` prediction for all instances in $\mathcal{Z}$, and the `BIERSparse` output otherwise. The intuition is that $\mathcal{Z}$ contains examples for which the intermediate dense embeddings better represent a mention-context than the more interpretable sparse output embeddings from the BIER model.

Because the sparse embeddings are interpretable, this analysis affords fine-grained analysis of which

---

[6] This is the default threshhold set in `SciSpacy` for concept candidate inclusion.

Dataset	Mentions	Abstracts	Type	Sets
MedMentions	350K	4.3K-PubMed	gold	no
BIERs*	37M	460K-PubMed	silver	no
ClinWikiNED*	10K	35K-Wiki	silver	yes

Table 1: Comparison of BioMed Linked Datasets. BIERs and ClinicalWikiNED datasets described in Section 3 and 4.1 respectivtely

entity types lead to incorrect predictions by the sparse model (but correct predictions using a dense representation). This diagnostic can be used as a benchmark for how well the model could have done had the entity typing model's output better represented the mention-context, or if the model had known to fallback to using the intermediate dense embedding; the former case might be ameliorated via more supervised examples or changes to the type system while the latter could motivate a dynamic approach to making predictions that is a function of model confidence. We show results and analysis using these methods in Section 5.

4 Experimental Setup

To evaluate the utility of the proposed biomedical entity representations, we use them for the tasks of biomedical entity label classification (ELC) and named entity disambiguation (NED). We highlight that these models perform well even without fine-tuning, which is critical in low- or zero-supervision scenarios.

4.1 NED on Biomedical Wikipedia articles

The NED task connects entity mentions in text with real world entities in a knowledge base by disambiguating the true entity from a list of candidates. We consider the local resolution setting in which each instance features a single entity mention span in the input text and several possible candidates with corresponding descriptions (e.g., the first paragraph of their Wikipedia article).

NED dataset construction While there exist multiple biomedical named entity recognition and linking datasets (Mohan and Li, 2019; Basaldella et al., 2020), we did not find much in the way of publicly available biomedical NED corpora, and we therefore constructed a new dataset, which we will release for use by other researchers. The dataset is based on the set of Wikipages used by Rosenthal et al. (2019), as relevant medical literature which consists of 34,692 medically relevant articles under the 'Clinical Medicine' category [7]. We used SLING[8] to process these articles and were able to retrieve around 1.5 million training examples (mention, context, [categories]) from them.

After obtaining these examples for each entity mention we used the CrossWikis dictionary (Spitkovsky and Chang, 2012) to try to gather between 3 to 5 challenging candidate entities for the example. This range in terms of number of candidates was selected because we wanted to include salient biomedical terms that are difficult to disambiguate; setting a higher number of potential candidates for use with CrossWikis largely gives general and short "popular" candidates (i.e., those that appear often in Wikipedia). This behavior makes sense since many biomedical terms are quite specific and usually only have a few high quality alternative candidates to select from. Additionally, we filter out redirect pages and pages that no longer match the wiki version used to create CrossWikis.

This candidate generation and data content acquisition step filters out considerably the number of available examples. We additionally subsample the dataset to reduce the instances where the "popular" candidate is the correct entity so as to make the task more difficult and to allow for more rare entities to appear in our set. After all the filtering, our ClinicalWikiNED dataset consists of a train/dev/test split of size 5332, 3730, and 800 respectively. Table 1 shows a comparison of the two datasets introduced in this paper with that of one of the largest publicly available linked biomedical datasets(Murty et al., 2018).

Using BIERs for NED Using the BIER architecture, we first train a separate `WikiDescription` model that takes as input a wikipage title as its mention, its first paragraph as the context, and outputs a sparse embedding that predicts the page's categories. As training data, we use any Wikipedia page that contains categories in our biomedical entity type system. We use 2.5 million such (title, descriptions, [categories]) as our training data, and we check for model convergence on a small development set. This model is used so that candidate embedding dimensions will align with our BIER mention-context embeddings.

For each mention m and context s in the test set, we use a BIER model to induce a sparse rep-

---

[7] https://en.wikipedia.org/wiki/Category:Clinical_medicine

[8] Based on a June 1, 2020 dump of English wikipedia.

Model	Test Acc. Dot Prod	Test Acc. Cosine Sim
BIER-PubMedBERT (ours)	80.1	**84.0**
BIER-SciBERT (ours)	76.4	77.3
BIER-BioBERT (ours)	71.9	75.9
Onoe and Durrett (2020)	63.6	69.8
Popular Prior	73.9	-
PubMedBERT (Gu et al., 2020)	77.6	-
SciBERT (Beltagy et al., 2019)	77.4	-
BioBERT (Lee et al., 2019)	77.9	-

Table 2: BIER zero shot test results vs Logistic Regression Baselines trained on task data for NED task

resentation t. We then go through each candidate c_i for the current test example and use the `WikiDescription` model to retrieve the candidate's sparse output embedding t_{c_i}. Finally, we compute both the cosine similarity and dot product of t with each candidate t_{c_i} and predict the candidate c_i that achieves the highest score for each metric as the true one.

Baseline model for NED We use the EntEval (Chen et al., 2019) framework for our experiments and train a logistic regression classifier using a feature vector composed of the mention-context embedding x_1 and current candidate wiki description embedding x_2 from the set of candidates C_m as a concatenation of x_1, x_2, element-wise product, and absolute difference: $[x_1, x_2, x_1 \odot x_2, |x_1 - x_2|]$. Both x_1 and x_2 are obtained via BERT based models. Training minimizes binary log loss using all negative examples. At test time, inference combines this classifier result with the prior probability of how frequently candidates occur in Wikipedia as follows: $arg\ max_{c \in C_m} [p_{prior}(c) + p_{classifier}(c)]$ to obtain the final candidate prediction. Directly using the most likely prior as predictions yields an accuracy of 73.9%. We emphasize that these baselines are fine-tuned on the task data while the BIER models only do inference on the test set.

Results Table 2 shows the results of the NED experiments. The biomedical BIER model affords improvements over the prior general domain interpretable model (Onoe and Durrett, 2020), showing that the biomedical type system and training is beneficial for this type of task. In addition, the BIER models outperform the baselines without fine-tuning on the training data.

4.2 ELC on Cancer Genetics data

For our entity label classification task we use the Cancer Genetics dataset (Pyysalo et al., 2013) which consists of 10,935 training, 3,634 dev, and 6,955 test examples from 300, 100, and 200 unique PubMed articles, respectively.[9] Given an article title and abstract, mention, and the corresponding entity label, the objective is to predict this label from 16 available coarse labels (see Table 7 in the Appendix for label distribution information).

To assess how well the learned BIER representations fare against comparable baselines, we perform a simple nearest neighbor classification technique using the proposed BIER model variants, the general domain model from Onoe and Durrett (2020), and non-BIER fine-tuned pre-trained language models as standalone encoders.

We first induce dense embeddings for all training examples by passing the mention m and context s through the encoders as `[CLS]` m `[SEP]` s `[SEP]`, and we store the resultant contextualized `[CLS]` embedding $\mathbf{h}_{\texttt{[CLS]}}$ as our dense embedding. For the BIER and Onoe and Durrett (2020) models we also save the final sparse entity embedding $\mathbf{t}$.

We iterate over the test examples and similarly induce dense representations for these $\mathbf{h}^{test}_{\texttt{[CLS]}}$ and (if applicable) sparse representations $\mathbf{t}^{test}$. We find their nearest neighbor (under either $\ell 2$ distance or dot product similarity) from the saved training set of embeddings, and use its label as the prediction. We use the FAISS semantic indexer (Johnson et al., 2017) for storing embeddings and finding nearest neighbors quickly. We are interested in evaluating the off-the-shelf utility of learned representations, and, as such, we do not train or fine-tune the models in any of these cases; rather, training examples are used only for nearest neighbor retrieval.

That said, for completeness we also performed additional experiments in which we do fine-tune models on the task data, with varying amounts of supervision; we are interested especially in low-supervision settings. For the fine-tuning experiment, we add a linear layer on top of the best performing BIER and baseline models, using cross entropy loss as our objective and fine-tuning them for 4 epochs on the training data before performing inference. For the low supervision regime experiment, we show how the best nearest neighbor and

---

[9] In our experiments we combine the train and dev sets into a single training set.

	Test Acc.			
	L2 Dist		Dot Prod	
Model	Dense	Sparse	Dense	Sparse
BIER-PubMedBERT	85.5	86.8	**88.2**	**87.5**
BIER-SciBERT	70.8	77.0	72.8	76.8
BIER-BioBERT	83.4	85.9	85.6	86.8
Onoe and Durrett (2020)	63.9	55.1	60.0	59.9
PubMedBERT	77.3	-	69.3	-
SciBERT	74.4	-	75.2	-
BioBERT	67.6	-	59.6	-

Table 3: Test accuracy on Cancer Genetics data using a nearest neighbor classifier (k=1) without fine-tuning based on sparse output or intermediate dense embeddings using L2 or Dot Product distance metrics.

fine-tuned models perform when given K examples per class for $K \in [5, 10, 25, 50, 75, 100, 200]$

Results Table 3 shows the results for our first experiment, in which we use untuned representations. We observe that the baseline language model encodings all perform worse than the proposed BIER sparse and dense models, with the exception of SciBERT, which fares better than the sparse BIER model based on SciBERT. Additionally, we see that BERT and Onoe and Durrett (2020) (which is based on BERT) both perform poorly in this biomedical task compared to the other baselines.

Importantly, we notice that the sparse interpretable embedding results for our top performing models (both BIER-PubMedBERT and BIER-BioBERT) perform near the level of their dense, non-interpretable counterparts. In the next section we will look at some illustrative test examples cases along with a simple technique to leverage both the dense and sparse embeddings that a BIER model can give to improve performance on the task and gain insight into where the entity type model and system may be underperforming.

Table 4 shows the results of our fine-tuning experiment. Freezing the model and allowing only the classification layer to learn weights doesn't allow enough capacity for either case, while fully fine-tuning both models gives improved performance in both models. However because the BIER model is no longer tied in, the interpretability component of our representations is eliminated, a limitation left for future work.

Figure 3 shows BIER-PubMedBERT performs better than the fine-tuned and non-interpretable PubMedBERT model when there are fewer than 100 examples per class (which is the case for 6 out of the 16 test classes in the dataset as seen in table 7 in the appendix).

	Test Acc.	
Model	Frozen Model	Fine-Tuned
BIER-PubMedBERT (ours)	68.0	96.0
PubMedBERT	36.2	96.1

Table 4: Test results on Cancer Genetics task with fine tuning on all data whether freezing the model or not.

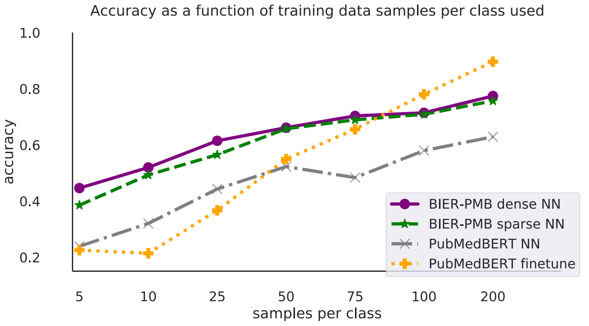

Figure 3: Results for the entity label classification task under varying amounts of supervision.

5 Debugging with BIERs

One of the claimed advantages of BIERs is their ability to facilitate model debugging. In this section we provide illustrative examples where the interpretability of the underlying representations offers insights into model behavior and suggests avenues for improvements.

Entity Type and Mention Analysis We illustrate the debugging strategy proposed in the context of entity label classification. Recall that this entails inspecting test examples for which the dense model yields a correct prediction, while the sparse variant does not (implying that the former somehow better represents the instance). We can inspect these cases to understand what entity types are leading to such behavior. Appendix Table 11 and 12 enumerate such mentions and their most probable types. We note the inclusion of many people's names (e.g., "Anthony Campbell", "Tony Walsh") which have been assigned at least some incorrect types in their sparse representations. This highlights a general failure mode of the model: It is assigning incorrect types to person names, which may be causing downstream prediction errors. This is actionable information, as we could remedy the issue via rules, additional, targeted supervision or by down weighting probabilities given to common erroneous types for these mentions.

To better characterize entity type errors, we gather the set of the 20 most probable entity types for all mentions incorrectly predicted by the BIER sparse model and sort types by frequency. We do the same for those predicted correctly. The resulting two lists share many of the same top popular types, but looking at relative rankings and only displaying those that are comparatively far apart[10] reveals some interesting results. Tables 9 and 10 report entity types correlated with correct and incorrect predictions, respectively. We emphasize this type of analysis is only possible due to the interpretable nature of the proposed BIER embeddings.

As a final illustrative debugging example, we consider a test example mention "thyroid carcinomas" with label "Cancer", along with the predictions made by the sparse model, "thyroid" with the incorrect label "Organ", and the dense model,"esophageal carcinoma" with the correct label "Cancer". We also retrieve the first correct prediction from the nearest neighbors of the sparse model embedding "medullary thyroid carcinoma" which we refer to as the *counterfactual* sparse prediction.[11] We take the dot product of the mention-context embedding with these three prediction's embeddings and inspect the top types which lead to their selection in Figure 6 in Appendix C. Both the incorrect sparse and correct counterfactual sparse predictions, at the surface level are quite similar to the test mention, but have lower scores for the entity type 'thyroid cancer' compared with the dense prediction which gives the correct label, but is semantically less similar to the test mention than the counterfactual sparse prediction. Additionally, the noisy type "rtt" erroneously plays more of a role in the sparse model predictions as well.

Diagnosing task results In analyzing errors made by the highest performing BIER dense and sparse nearest neighbor models for the entity label classification task, we noticed that while there was high concurrence for correct predictions (i.e., of the 88% true predictions made by the dense model overall, the sparse model agreed with the prediction 95% of the time), the cases where the model predictions disagreed, but where one of them still predicted the true label, were quite varied. In other words, the sparse model gave many correct results on test cases when the dense model gave incorrect ones and vice versa. Applying the diagnostic technique from Section 3, we see the classifier's overall performance could have improved from 88.2 to 91.9 had the model known when to utilize its intermediate dense representation over its sparse output.

	Test Acc.			
Task	Dense	Sparse	Combined	Δ
NED	84.0	81.0	**91.7**	+7.7
ELC	87.5	88.2	**91.9**	+3.7

Table 5: Results for both tasks showing improvements that could have been achieved by combining intermediate dense and interpretable sparse output embeddings generated by the same BIER-PubMedBERT model.

Similarly we applied the diagnostic technique to the NED task and leave more details in Appendix B. Incorporating mentions that the dense dot product BIER model handles better than the cosine similarity based sparse one does would have given an improvement from our prior accuracy of 84.0 to 91.7. Table 5 shows the possible improvement in task accuracies for both tasks.

6 Related Work

In this work we have introduced a predefined fine-grained biomedical type system comprising 68k types, explicitly tied to PubMed. Instead of using a fixed type system, Raiman and Raiman (2018) seek to dynamically learn a 100 dimensional type system from a much larger general domain type system in order to optimally disambiguate entities.

Aside from work on biomedical NLP and entities specifically, there exists a line of work on interpretable word embeddings (Subramanian et al., 2017; Faruqui et al., 2015). A common approach here is to identify the groups of words most associated with vector components globally, somewhat akin to topic models. This differs from our approach, which is based on an external type system and provides immediate, instance-level interpretable probabilities for each entity type. Hu et al. (2020) proposes transforming dense to sparse representations independent of entity typing.

Another related line of work tests a models' ability to induce syntactic or type information by the measuring accuracy of a probe (Peters et al., 2018; Hewitt and Manning, 2019; Hewitt and Liang, 2019). There is significant uncertainty about how

---

[10] We chose to highlight entity types that are farther than 50 rankings apart to have a small set to display.

[11] Had the mention under consideration instead mapped to this sparse representation, the prediction would have been different, and correct.

to calibrate such post-hoc probing results (Voita and Titov, 2020) whereas our model's representations are directly interpretable.

While many interesting biomedical entity representation and linking task oriented works (Murty et al., 2018; Vashishth et al., 2020; Mondal et al., 2019; Sung et al., 2020; Liu et al., 2020) leverage PubMed or UMLS for semantic type, entity synonym, or self alignment purposes, our work is the first to incorporate interpretable embeddings that are linked to a biomedical entity type system.

7 Conclusions

We have introduced a new biomedical entity typing system and training set from a large corpus of biomedical texts. We will release this dataset, which comprises 37 million derived triples. Exploiting this data, we proposed *Biomedical Interpretable Entity Representations* (BIERs), in which dimensions correspond to fine-grained entity types, and values are predicted probabilities that a given entity is of the corresponding type.

Using two downstream biomedical tasks, we showed that BIER representations yield predictive performance that is competitive with dense (uninterpretable) representations, and that such representations are particularly beneficial in zero-shot or low-supervision settings. We also demonstrated that BIER representations can facilitate meaningful model debugging both at the mention and entity type level.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation (under Grants No. NSF 1901117 and NSF 1750978) and in part by a grant from the Office of Naval Research (ONR N00014-17-1-2143). The project was conducted under the auspices of the IBM Science for Social Good initiative.

Ethical Considerations

NLP models are increasingly used in biomedicine, where some applications can be quite high-stakes. Establishing trust in such models is therefore paramount; unfortunately, deep neural networks tend to be opaque in their operations, potentially precluding their use in certain areas of biomedicine where they might otherwise be beneficial. This work is a step towards more transparent models for biomedical NLP.

References

Marco Basaldella, Fangyu Liu, Ehsan Shareghi, and Nigel Collier. 2020. COMETA: A Corpus for Medical Entity Linking in the Social Media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mingda Chen, Zewei Chu, Yang Chen, Karl Stratos, and Kevin Gimpel. 2019. EntEval: A holistic evaluation benchmark for entity representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

John Cuzzola, E. Bagheri, and Jelena Jovanovic. 2018. UMLS to DBPedia Link Discovery Through Circular Resolution. *Journal of the American Medical Informatics Association*, 25:819–826.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse Overcomplete Word Vector Representations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing.

John Hewitt and Percy Liang. 2019. Designing and Interpreting Probes with Control Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Hewitt and Christopher D. Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Wenpeng Hu, Mengyu Wang, Bing Liu, Feng Ji, Jinwen Ma, and Dongyan Zhao. 2020. Transformation of Dense and Sparse Text Representations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2020. Self-alignment Pre-training for Biomedical Entity Representations.

Sunil Mohan and Donghui Li. 2019. MedMentions: A Large Biomedical Corpus Annotated with UMLS Concepts. *CoRR*, abs/1902.09476.

Ishani Mondal, Sukannya Purkayastha, Sudeshna Sarkar, Pawan Goyal, Jitesh Pillai, Amitava Bhattacharyya, and Mahanandeeshwar Gattu. 2019. Medical Entity Linking using Triplet Network. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to Denoise Distantly-Labeled Data for Entity Typing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yasumasa Onoe and Greg Durrett. 2020. Interpretable Entity Representations through Large-Scale Typing. In *Findings of the Association for Computational Linguistics: EMNLP*.

Nicolas Papernot and P. McDaniel. 2018. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. *ArXiv*, abs/1803.04765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. BERT is Not a Knowledge Base (Yet): Factual Knowledge vs. Name-Based Reasoning in Unsupervised QA. *ArXiv*, abs/1911.03681.

Sampo Pyysalo, Tomoko Ohta, and Sophia Ananiadou. 2013. Overview of the Cancer Genetics (CG) task of BioNLP Shared Task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*.

Jonathan Raiman and Olivier Raiman. 2018. DeepType: Multilingual Entity Linking by Neural Type System Evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Michael Ringgaard, Rahul Gupta, and Fernando C. N. Pereira. 2017. SLING: A framework for frame semantic parsing. *ArXiv*, abs/1710.07032.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? *ArXiv*, abs/2002.08910.

Sara Rosenthal, Ken Barker, and Zhicheng Liang. 2019. Leveraging Medical Literature for Section Prediction in Electronic Health Records. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *Proceedings of LREC*.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. 2017. SPINE: Sparse Interpretable Neural Embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Mujeen Sung, Hwisang Jeon, Jinhyuk Lee, and Jaewoo Kang. 2020. Biomedical Entity Representations with Synonym Marginalization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shikhar Vashishth, Rishabh Joshi, Denis Newman-Griffis, Ritam Dutt, and Carolyn Rose. 2020. MedType: Improving Medical Entity Linking with Semantic Type Prediction.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

Elena Voita and Ivan Titov. 2020. Information-Theoretic Probing with Minimum Description Length. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chih-Hsuan Wei, Alexis Allot, Robert Leaman, and Zhiyong Lu. 2019. PubTator central: automated concept annotation for biomedical full text articles. *Nucleic Acids Research*, 47(W1):W587–W593.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv*, abs/1609.08144.

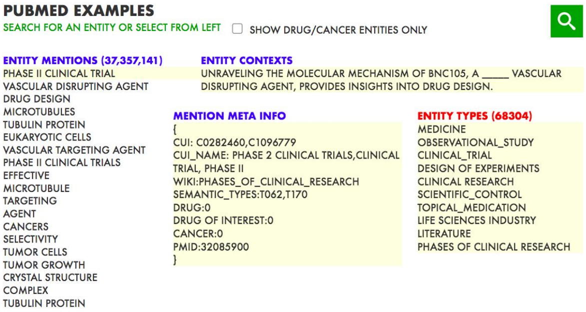
PUBMED EXAMPLES

SEARCH FOR AN ENTITY OR SELECT FROM LEFT ☐ SHOW DRUG/CANCER ENTITIES ONLY

ENTITY MENTIONS (37,357,141)

PHASE II CLINICAL TRIAL
VASCULAR DISRUPTING AGENT
DRUG DESIGN
MICROTUBULES
TUBULIN PROTEIN
EUKARYOTIC CELLS
VASCULAR TARGETING AGENT
PHASE II CLINICAL TRIALS
EFFECTIVE
MICROTUBULE
TARGETING
AGENT
CANCERS
SELECTIVITY
TUMOR CELLS
TUMOR GROWTH
CRYSTAL STRUCTURE
COMPLEX
TUBULIN PROTEIN

ENTITY CONTEXTS

UNRAVELING THE MOLECULAR MECHANISM OF BNC105, A _____ VASCULAR DISRUPTING AGENT, PROVIDES INSIGHTS INTO DRUG DESIGN.

MENTION META INFO

```
{
CUI: C0282460,C1096779
CUI_NAME: PHASE 2 CLINICAL TRIALS,CLINICAL
TRIAL, PHASE II
WIKI:PHASES_OF_CLINICAL_RESEARCH
SEMANTIC_TYPES:T062,T170
DRUG:0
DRUG OF INTEREST:0
CANCER:0
PMID:32085900
}
```

ENTITY TYPES (68304)

MEDICINE
OBSERVATIONAL_STUDY
CLINICAL_TRIAL
DESIGN OF EXPERIMENTS
CLINICAL RESEARCH
SCIENTIFIC_CONTROL
TOPICAL_MEDICATION
LIFE SCIENCES INDUSTRY
LITERATURE
PHASES OF CLINICAL RESEARCH

Figure 4: Example from derived Biomedical Dataset

A BIER system level specifics

To better illustrate the process of which mentions are retained during our filtering process, Table 6 shows the 6 concepts associated with an example mention of ”phase II clinical trial” found in a PubMed article. We see all 6 concepts score higher than our minimum threshold and we use the two highest scoring matches that are within 2 points of each other: CUIDs C0282460 and C1096779. the former C0282460 has a WikiPedia data item Q7180990 that corresponds to the page “wiki/Phases_of_clinical_research” whose associated categories are “Clinical research”, “Design of experiments”, “Life sciences”, “industry”. The second result C1096779 has no direct WikiPedia match and the results we get from SLING include “Clinical trial”, “Scientific control”, “Medicine”, “Topical medication”, “Observational study”, “Literature”. Hence for this mention and context from a PubMed abstract, we are able to extract a (mention, context, list of types) triple of the form (“phase II clinical trial”, context, [“Clinical research”, “Design of experiments”, “Life sciences” industry”, “Clinical trial”, “Scientific control”, “Medicine” ....]].

CUID	**Concept Name**	**Score**	**DBPedia**
C0282460	Phase 2 Clinical Trials	0.9999	Q7180990
C1096779	Clinical Trial, Phase II	0.9999	none
C0282461	Phase 3 Clinical Trials	0.9496	Q7180990
C0920321	Phase I Clinical Trials	0.8707	Q7180990
C1096780	Clinical Trial, Phase III	0.8635	none
C0282462	Phase 4 Clinical Trials	0.8208	Q7180990

Table 6: Using an NER tagger we find 6 associated concepts in UMLS for the mention “phase II clinical trial” in a context sentence “Unraveling the molecular mechanism of BNC105, a phase II clinical trial vascular disrupting agent, provides insights into drug design.”

Label	Train + Dev Set % (raw)	Test Set % (raw)
Gene_or_gene_product	36.98 (5388)	36.23 (2520)
Cell	17.32 (2524)	15.15 (1054)
Cancer	11.52 (1679)	13.30 (925)
Simple_chemical	10.59 (1543)	10.45 (727)
Organism	8.63 (1258)	7.81 (543)
Multi-tissue_structure	3.80 (554)	4.36 (303)
Tissue	2.77 (403)	2.73 (190)
Cellular_component	2.67 (389)	2.59 (180)
Organ	1.82 (265)	2.24 (156)
Organism_substance	1.24 (181)	1.47 (102)
Pathological_formation	0.96 (140)	1.28 (89)
Amino_acid	0.50 (73)	0.89 (62)
Immaterial_anatomical _entity	0.49 (71)	0.45 (31)
Organism_subdivision	0.40 (59)	0.56 (39)
Anatomical_system	0.16 (24)	0.24 (17)
Developing_anatomical _structure	0.12 (18)	0.24 (17)

Table 7: Cancer Genetics Dataset Label Distribution

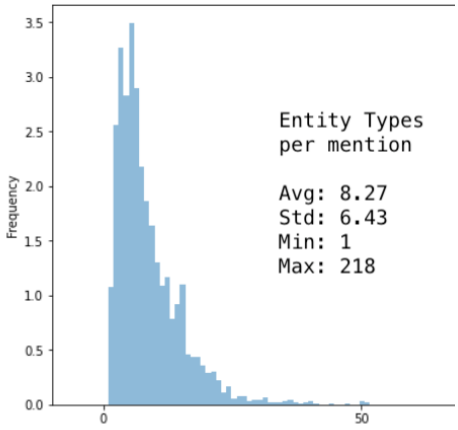

Figure 5: Entity Types per mention on Training set for BIER

B NED diagnostic details

For the NED task we used the BIER’s sparse embeddings of test mentions in their contexts and took cosine similarity with a separate BIER model’s sparse embeddings of candidate wiki descriptions to make our predictions. To use the diagnostic technique we first get task predictions using the dense embeddings from the BIER models which gives results of 81 and 79.25 percent test accuracy using dot product and cosine similarity respectively. Although the prior sparse cosine similarity BIER model in this case gave a higher 84.0 percent test accuracy, using the diagnostic technique in this case by incorporating mentions the dense dot product BIER model handles better would have given an improvement in accuracy from 84.0 to 91.65.

1-25	26-50	51-75	76-100
term	w.h.o. essential medicines	test_(assessment)	causality
ingredient	psychosis	drug discovery	hydroxyl
disease	scientific_method	receptor_(biochemistry)	adverse_effect
cell_(biology)	oncology	observational_study	diagnosis
rtt	enzyme	immunology	physiology
protein	human_body	molecular biology	chemotherapy
gene	psychotherapy	abnormality_(behavior)	hepatotoxins
human	medicine	radioactive_decay	molecule
neoplasm	grammatical_modifier	derivative_(chemistry)	phenotype
cancer	tissue_(biology)	chemistry	cell biology
therapy	treatment_and_control_groups	health policy	concepts in metaphysics
medical terminology	scientific method	amine	concepts in epistemology
measurement	coagulation	peptide	apoptosis
patient	chemical_reaction	pharmaceutical sciences	procedural_law
chemical_compound	philosophy of science	antigen	science
surgery	calcium_in_biology	biology	genetic_code
nitrous_oxide	enzyme_inhibitor	algorithm	empiricism
pharmaceutical_drug	medicinal chemistry	texas	family
acid	research	mental_disorder	thailand
articles containing video clips	metabolism	statistical_hypothesis_testing	liver
malignancy	taxonomy_(biology)	catalysis	medical mnemonics
time	cell_growth	allele	dosage_form
prothrombin_time	blood	methyl_group	immune system
cognition	syndrome	infectious causes of cancer	amino_acid
drug	sewage_treatment	database	beta_sheet

Table 8: Top 100 most frequent types from Biomedical Entity Type System

Incorrect rank	Entity Type	Correct rank	Relative difference
20	tongue	76	56
24	anatomy	160	136
29	protein_domain	112	83
34	organ_(anatomy)	107	73
35	gland	205	170
38	phosphatase	140	102
43	surgery	120	77
46	circulatory_system	293	247
50	squamous-cell_carcin	111	61
51	nephron	142	91
60	anatomical_terms	169	109
61	kidney	284	223
62	cancer_cell	213	151
70	activator_(genetics)	179	109
71	drug	192	121
74	breast_cancer	127	53
75	locus_(genetics)	206	131
77	cancer_staging	256	179
79	signal transduction	233	154
81	multiprotein_complex	132	51
82	endometrium	200	118
83	mouth	200	117
84	cell anatomy	272	188
90	molecular biology	200	110
93	rare cancers	200	107
95	website	161	66
96	cell_cycle	200	104
97	gene expression	178	81
98	hydroxyl	221	123
99	oral_sex	200	101

Table 9: Entity Types more associated with erroneous predictions

Incorrect rank	Entity Type	Correct rank	Rel diff
23	syndrome	244	221
34	abnormality_(behavior)	270	236
35	elementary_particle	128	93
42	apoptosis	200	158
51	congenital_disorder	200	149
53	transformation_(genetics)	275	222
54	measurement	109	55
55	human cells	147	92
56	immune system disorders	200	144
57	paraneoplastic syndromes	200	143
58	code	154	96
59	battery_(electricity)	200	141
61	virus	222	161
63	chemistry	281	218
66	calcium_in_biology	209	143
71	thymus	200	129
72	medical terminology	190	118
73	cell biology	297	224
74	recombinant_dna	10	64
76	tongue	20	56
79	protein_kinase	164	85
80	drama	200	120
85	tumor_suppressor_gene	14	71
86	patient	200	114
87	specialty_(medicine)	234	147
90	growth_hormone	200	110
91	taxonomy_(biology)	238	147
93	t cells	228	135
94	childhood	200	106
95	aging-related proteins	200	105
96	network_affiliate	200	104
97	blood tests	200	103
98	protein_a	200	102

Table 10: Entity Types more associated with correct predictions

Mention	**Sparse embedding top types that do worse than dense counterparts**
albinism	albinism, disease, animal coat colors, heredity, dermatologic terminology, articles containing video clips, hair, skin, pigment, nitrous_oxide
alveolar ridge	tooth, lung anatomy, mouth, pulmonary_alveolus, human mouth anatomy, dental_caries, periodontology, parts of tooth, mandible, leaf
anastomosis	surgery, anastomosis, evolutionary biology, digestive system, angiology, anatomy, lawsuit, combat, organ_(anatomy), surgical_anastomosis
anthony campbell	carl_linnaeus, ingredient, human, taxa named by carl linnaeus, flora of asia, world health organization essential medicines, coordination_complex, western european countries, extract, asteraceae genera
autonomic neuropathy	peripheral nervous system disorders, autonomic nervous system, disease, nervous_system, peripheral_neuropathy, functional_group, heredity, mental_disorder, nerve
bleb	skin conditions resulting from physical factors, lesion, fluid, frostbite, radiation health effects, disease, source_code, nitrous_oxide, skin, hematology
cancer cells	cell_(biology), cell_culture, medical terminology, oncology, cancer, precancerous_condition, large_cell, human, protein, standard_operating_procedure
cell death	mitochondria, programmed cell death, death, cell_(biology), cellular senescence, cognition, apoptosis, tgf_beta_signaling_pathway, nuclear_receptor, survival_rate
chronic lymphocytic leukemia	plasma_cell, small_intestine, mood_disorder, b-cell_lymphoma, lymphocytic leukemia, bioaccumulation, bone_marrow, grading_(tumors), lymphatic_system, lymphoblast
cosmological constant	ionizing radiation, assumption, units_of_measurement, comics by steve ditko, cell_(biology), grammatical_modifier, quantity, blood_plasma, industrial gases, litre
demethylation	organic reactions, gene expression, posttranslational modification, rna, transcription_(genetics), molecular genetics, epigenetics, therapy, demethylation, molecular biology
dissociation constant	wine regions of south africa, suburbs of cape town, astronomical_unit elementary_particle, rat, medical terminology, gene, units_of_measurement, furans
endoscope	endoscopy, bicycle, diagnostic gastroenterology, physical_examination, gastroenterology, microphone, video_camera, israeli inventions, pencil, 21st-century inventions
fingering	finger, conditions of the skin appendages, articles containing video clips, disease, toe, nitrous_oxide, hand, keratin, reflex, fingers
flirting	female, causes of death, fly, metrorrhagia, disease, articles containing video clips, conditions of the skin appendages, etiology, dog, homology_(biology)
geniculate	tongue, anatomical_terms_of_location, ganglion, mandible, midbrain, organ_(anatomy), cell_nucleus, cerebral_cortex, lobe_(anatomy), middle_ear
glomerulus	kidney, tongue, connective_tissue, epithelium, gene, cell_membrane, nitrous_oxide, organ_(anatomy), nephrology, derivative_(chemistry)
guy davis	hemoglobins, respiratory physiology, hemoglobin, geography, equilibrium chemistry, cancer, race_and_ethnicity_in_the_united_states_census, geographic_coordinate_system, texas
infant formula	infant, infant feeding, child, milk, formula, dosage_form, foods, breast milk, preterm_birth, chemistry
intussusception	bowel_obstruction, human_gastrointestinal_tract, large_intestine, invagination, disease, morphology_(biology), colorectal_cancer, nitrous_oxide, deconstruction, thrombosis
isomerization	isomerism, stereochemistry, metabolism, 1827 introductions, laboratory techniques, transgender, chemistry, organic chemistry, isomerases, flora of california
mescaline	ingredient, rtt, mesylate, abbvie inc. brands, anti-inflammatory, orphan drugs, acid, methyl_group, carbamates, amine
methylation	epigenetics, posttranslational modification, amino_acid, protein, methylation, acid, organic reactions, amine, antigen, ingredient

Table 11: NED examples where dense BIER embeddings outperforms sparse (interpretable) BIER representations. Mentions start with [A-M].

Mention	Sparse embedding top types that do worse than dense counterparts
n400	antigen, cancer, gene, protein, units_of_measurement, ratio, allele, human, time, nucleolus
peroxides	acne_vulgaris, topical_medication, ingredient, functional_group, route_of_administration, peroxides, chemical_reaction, glandular and epithelial neoplasia, functional groups, pharmaceutical_drug
polyunsaturated fatty acids	nutrition, fatty acids, acid, lipids, ester, protein, ingredient, food science, neuronal_ceroid_lipofuscinosis, lipid
protease inhibitors	endopeptidase, enzyme_inhibitor, biosynthesis, protease, chemical_compound, receptor_antagonist, peptide, moa, hiv, hiv-1_protease
psychological dependence	psychology, substance_dependence, substance_abuse, emotion, mental_disorder, dependent territories, governance of the british empire, mental and behavioural disorders, crown dependencies, british islands
psychopathy	abnormal psychology, abnormality_(behavior), nitrous_oxide, pathology, disease, behavioural sciences, psychosis, mental and behavioural disorders, mental_disorder, affect_(psychology)
resection	technology, natural_resource, segmental_resection, plant anatomy, plant physiology, surgical_suture, surgery, plant morphology, morphology_(biology), amputation
sarcoidosis	epithelioid_cell, etiology, chilblains, disease, nitrous_oxide, kalashnikov derivatives, sarcoidosis, organ_(anatomy), 5.56×45mm nato assault rifles, carbines
semicircular canals	tongue, ear_canal, vestibular system, auditory system, eustachian_tube, canal_(anatomy), auditory_system, crystal_structure, vestibulocochlear nerve, cranial_cavity
sequence analysis	sequence, bioinformatics, psychosis, psychoanalysis, dna, scientific method, nucleic_acid_sequence, physical_examination, dna_sequencing, algorithm
tony walsh	race_and_ethnicity_in_the_united_states_census, adult, flora of asia, carl_linnaeus, human, french-speaking countries, flora of north america, hemoglobins, women, coagulation system
united states department of agriculture	united_states, united states federal executive departments, management, police, public_health, united_states_department_of_defense, united states department of health and human services agencies, regulators of biotechnology products, 1889 establishments in the united states
ventricular zone	ventricle_(heart), zoning, ventricular_system, ventricular system, brain, developmental neuroscience, tongue, urban planning, anatomical_terms_of_location, bone
wechsler adult intelligence scale	psychological testing, psychiatric assessment, connective/soft tissue tumors and sarcomas, nitrous_oxide, psychiatric diagnosis, medical scales, level_of_measurement, adult, childhood
yang xiong	yin_and_yang, qi, alternative medicine, taoist cosmology, chinese martial arts terminology, chinese philosophy, plants used in traditional chinese medicine, gene, qigong, trees of china

Table 12: NED examples where dense BIER embeddings outperform sparse BIER representations. Mentions start with [N-Z].

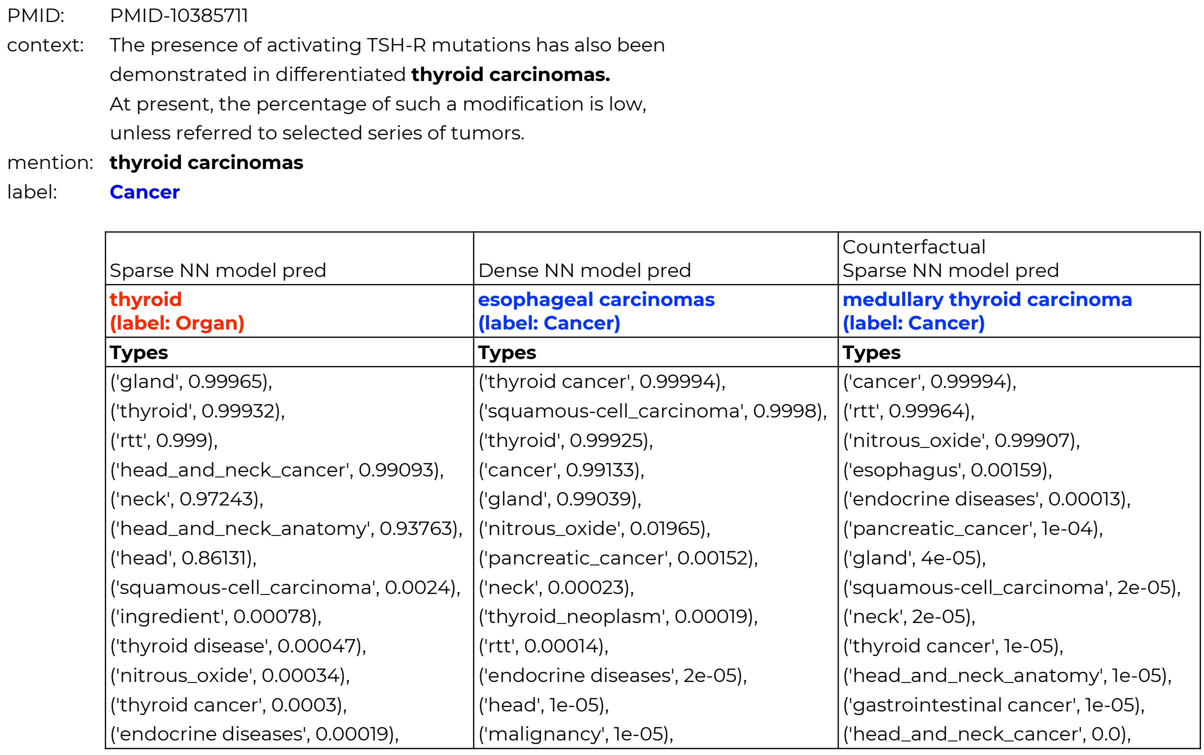

PMID: PMID-10385711
context: The presence of activating TSH-R mutations has also been demonstrated in differentiated **thyroid carcinomas.** At present, the percentage of such a modification is low, unless referred to selected series of tumors.
mention: **thyroid carcinomas**
label: **Cancer**

Sparse NN model pred	Dense NN model pred	Counterfactual Sparse NN model pred
thyroid (label: Organ)	**esophageal carcinomas (label: Cancer)**	**medullary thyroid carcinoma (label: Cancer)**
Types	**Types**	**Types**
('gland', 0.99965), ('thyroid', 0.99932), ('rtt', 0.999), ('head_and_neck_cancer', 0.99093), ('neck', 0.97243), ('head_and_neck_anatomy', 0.93763), ('head', 0.86131), ('squamous-cell_carcinoma', 0.0024), ('ingredient', 0.00078), ('thyroid disease', 0.00047), ('nitrous_oxide', 0.00034), ('thyroid cancer', 0.0003), ('endocrine diseases', 0.00019),	('thyroid cancer', 0.99994), ('squamous-cell_carcinoma', 0.9998), ('thyroid', 0.99925), ('cancer', 0.99133), ('gland', 0.99039), ('nitrous_oxide', 0.01965), ('pancreatic_cancer', 0.00152), ('neck', 0.00023), ('thyroid_neoplasm', 0.00019), ('rtt', 0.00014), ('endocrine diseases', 2e-05), ('head', 1e-05), ('malignancy', 1e-05),	('cancer', 0.99994), ('rtt', 0.99964), ('nitrous_oxide', 0.99907), ('esophagus', 0.00159), ('endocrine diseases', 0.00013), ('pancreatic_cancer', 1e-04), ('gland', 4e-05), ('squamous-cell_carcinoma', 2e-05), ('neck', 2e-05), ('thyroid cancer', 1e-05), ('head_and_neck_anatomy', 1e-05), ('gastrointestinal cancer', 1e-05), ('head_and_neck_cancer', 0.0),

Figure 6: Analysis Example for ELC task

Learning Robust Latent Representations for Controllable Speech Synthesis

Shakti Kumar[*†] **Jithin Pradeep**[*] **Hussain Zaidi**[*]

[*]Vanguard Center for Analytics and Insights
[†]Computer Science, University of Toronto

{shakti_kumar, jithin_pradeep, hussain_zaidi}@vanguard.com

Abstract

State-of-the-art Variational Auto-Encoders (VAEs) for learning disentangled latent representations give impressive results in discovering features like pitch, pause duration, and accent in speech data, leading to highly controllable text-to-speech (TTS) synthesis. However, these LSTM-based VAEs fail to learn latent clusters of speaker attributes when trained on limited or noisy datasets. Further, different latent variables are found to encode the same features, limiting the control and expressiveness during speech synthesis. To resolve these issues, we propose REMMI (Reordered transformer Encoder with Minimal Mutual Information) where we minimize the mutual information between different latent variables and devise a modified Transformer architecture with layer reordering to learn controllable latent representations in speech data. We show that REMMI reduces the cluster overlap of speaker attributes by at least 30% over LSTM-VAE.

1 Introduction

Learning disentangled latent representations in speech is an active area of research (Hsu et al., 2017; Chou et al., 2018; Park et al., 2020) with applications in controlling the style (for example, pitch, pause duration, and accent) of synthesized speech. Recurrent architectures like Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks in Variational Autoencoders (VAE) have been state-of-the-art in discovering disentangled latent representations in speech (Wang et al., 2018; Jia et al., 2018; Skerry-Ryan et al., 2018) as well as sequential data more generally. For example Li and Mandt (2018) attempt to disentangle global and local features of video/speech in different latent variables. Hsu et al. (2019) disentangled different dimensions of the latent variables to discover meaningful representations and hence proposed a speech synthesis model with controllable pitch, pause duration, and speed.

These papers as well as several others (Chung et al., 2015; Hsu et al., 2019; Leglaive et al., 2020; Hono et al., 2020; Sun et al., 2020) make one limiting assumption— the availability of hundreds of hours of speech data for training deep learning networks. As we show in our experiments, state-of-the-art VAEs fail to learn meaningful separation of speaking styles in speech data when presented with small datasets. In addition, different latent variables learned by the VAE are no longer uncorrelated. Both these shortcomings lead to poor control of speaking styles during synthesis.

While LSTMs are state-of-the-art in learning latent variables in speech, Transformers have been used for understanding latent representations for text completion (Wang and Wan, 2019) and Transformer-based VAEs were used in Jiang et al. (2020) to model independent style attributes in music generation.

Inspired by these limitations of LSTM-based VAEs and the promise of more ”attentive” networks, we modify the loss function of the state-of-the-art VAEs (Hsu et al., 2019) by explicitly minimizing the mutual information between latent variables, thereby penalizing common learned features between different representations. We then modify Transformer architecture for learning robust disentangled latent representations of speech from limited and noisy data. We show that our proposed architecture– REMMI (Reordered transformer Encoder with Minimal Mutual Information) discovers compact stable latent representations of speaker attributes even on datasets as small as 4 hours of total speech samples while state-of-the-art fails. Our proposed VAE outperforms LSTM and vanilla Transformers even on challenging dataset like Common Voice which has considerable background noise, low recording quality and large num-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3562–3575
August 1–6, 2021.

ber of speakers with the same style or accent. To summarize, following are the main contributions of our work,

1. Formulate a modified VAE loss function for speech data and a novel Transformer-based VAE for learning uncorrelated latent variables, thereby allowing more precise control over synthesis compared to the existing state-of-the-art.
2. Show that our latent clusters of speaking styles are better separated than existing LSTM and vanilla Transformer based VAEs on noisy and small datasets.
3. Show that the our modified Transformer architecture allows a faster convergence of the variational lower bound compared to both vanilla Transformer and LSTM based VAEs.

2 Related Work

Multiple previous work have targeted this problem of learning latent representations for sequential data like speech (Wang et al., 2018; Jia et al., 2018; Skerry-Ryan et al., 2018). As discussed, the main advantage of learning such representations is that it allows creating diverse examples during reconstruction by manipulating the encoded latent variable. Li and Mandt (2018) propose two sets of latents which learn global features like the generated sequence contents and local dynamic features such as pitch, speed etc. However, a limitation of this approach is the lack of interpretability of the learnt dimensions— it is known that the different dimensions of the latent variables are learning some features but there is little to no visibility into what those actual features are.

Modifying Text-to-Speech systems by introducing additional encoders has been a standard way to discover meaningful representations. Zhang et al. (2019) build on top of Tacotron-2 (Shen et al., 2018) architecture and use Gaussians to model their latent variables. An improved version can be seen in Hsu et al. (2019) where a hierarchical latent with mixture of Gaussians is used. Hsu et al. (2019) propose adversarial training to further improve latent variables and the features discovered by disentangling the background noise and reverberation along with speaker identity from the recording conditions.

While all these prior work aim to discover latent representations, there is a lot of room for improving those representations especially in cases where we have very limited hours of speech dataset. As we show in our experiments, in the absence of explicit restrictions on the training objective these VAEs easily collapse when presented with smaller datasets. Thus we focus on improving the representations, specifically latent clusters of speaker attributes, in cases of extremely limited datasets. Our contributions, however are not limited to smaller datasets and we see similar improved performance on larger and noisy datasets too.

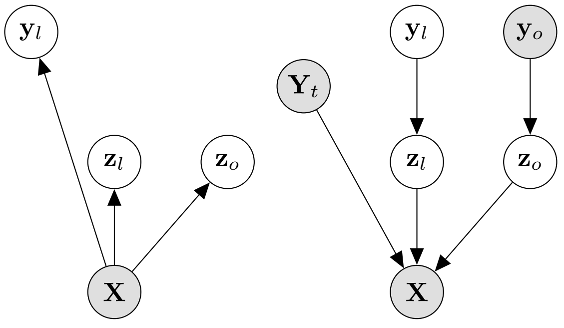

(a) Encoder (b) Generator

Figure 1: Graphical model of controllable TTS system. Note that $q(\mathbf{y}_l|\mathbf{X})$ in the Encoder can be approximated in terms of $q(\mathbf{z}_l|\mathbf{X})$, in which case node $\mathbf{y}_l$ will have an edge from $\mathbf{z}_l$ instead of $\mathbf{X}$ as done in Hsu et al. (2019).

3 Background

Controllable text-to-speech (TTS) VAE-based systems like in Hsu et al. (2019) take an input text sequence $\mathbf{Y}_t$ and an optional observed categorical label $\mathbf{y}_o$ (e.g., speaker identity or accent) as input and learn to synthesize a sequence, usually mel-spectrogram frames $\mathbf{X}$ as output. Additional latent variables $\mathbf{z}_o$ and $\mathbf{z}_l$ can be introduced to discover meaningful representations during this process. Here $\mathbf{z}_o$ is a continuous latent learnt on top of shown labels $\mathbf{y}_o$, hence $\mathbf{z}_o$ captures the variation in features correlated with the speaker attribute $\mathbf{y}_o$. $\mathbf{z}_l$ is a completely unsupervised continuous variable learnt on top of standard Expectation-Maximization style latent mixture components $\mathbf{y}_l$. This graphical model is depicted in Figure 1. The objective function for learning such model, i.e. synthesizing sequence $\mathbf{X}$ given $\mathbf{Y}_t$ and $\mathbf{y}_o$, can be formulated as the variational lower bound[1],

---

[1] Complete derivation is given in the Appendix A.

$$
\begin{aligned}
&log\ p(\mathbf{X}|\mathbf{Y}_t, \mathbf{y}_o) \geq\ log\ p(\mathbf{X}|\mathbf{Y}_t, \widetilde{\mathbf{z}_o}, \widetilde{\mathbf{z}_l}) \\
&\quad - \sum_{\mathbf{y}_l=1}^{K} q(\mathbf{y}_l|\mathbf{X}) D_{KL}[\ q(\mathbf{z}_l|\mathbf{X})\ ||\ p(\mathbf{z}_l|\mathbf{y}_l)\ ] \\
&\quad - D_{KL}[\ q(\mathbf{y}_l|\mathbf{X})\ ||\ p(\mathbf{y}_l)\ ] \\
&\quad - D_{KL}[q(\mathbf{z}_o|\mathbf{X})\ ||\ p(\mathbf{z}_o|\mathbf{y}_o)\ ] \\
&\quad = -L_{mel} - L_{KL}
\end{aligned}
$$

where $L_{mel} = -log\ p(\mathbf{X}|\mathbf{Y}_t, \widetilde{\mathbf{z}_o}, \widetilde{\mathbf{z}_l})$ and L_{KL} refers to the remaining terms. Here $\widetilde{\mathbf{z}_o}, \widetilde{\mathbf{z}_l}$ are sampled points and are reparameterized (Kingma and Welling, 2014) as $\widetilde{\mathbf{z}_o} = \hat{\mu_o} + \hat{\sigma_o} \odot \epsilon_o$ and $\widetilde{\mathbf{z}_l} = \hat{\mu_l} + \hat{\sigma_l} \odot \epsilon_l$ with $\hat{\mu_o}, \hat{\mu_l}, \hat{\sigma_o}, \hat{\sigma_l}$ as the mean and standard deviation of the posterior distributions $q(\mathbf{z}_o|\mathbf{X})$ and $q(\mathbf{z}_l|\mathbf{X})$ respectively and with auxiliary noise variable $\epsilon_o, \epsilon_l \sim \mathcal{N}(0, I)$. Following Higgins et al. (2017) the loss L can be written in a more general form as,

$$L = L_{mel} + \beta L_{KL} \tag{1}$$

with β balancing the relative weighing between the latent channels and reconstruction accuracy. Here L_{mel} is the mel loss which controls the quality of the mel-spectrograms produced and L_{KL} refers to the total KL Loss controlling the features learnt in latent variables.

This VAE can be used in the Tacotron-2 architecture (Hsu et al., 2019) as shown in Figure 2(a) to learn the text to mel-spectrogram mapping and the latent features controlled by L_{KL}.

4 Methodology

We now describe the two main components, 1) Minimizing mutual information and 2) Layer reordering in our proposed REMMI architecture.

4.1 Minimizing Mutual Information

The latent $\mathbf{z}_l$ in Figure 1 is unsupervised while the latent $\mathbf{z}_o$ learns features correlated with the shown label $\mathbf{y}_o$. Our experiments showed that both $\mathbf{z}_l, \mathbf{z}_o$ can end up encoding the same set of features, which leads to poor control in synthesizing speech. An intuition into why this happens lies in the fact that $\mathbf{z}_l$ is an unsupervised variable and it can discover any feature hidden in the input speech sequence. There is no term in the loss function (1) which prevents the features of $\mathbf{z}_l$ from being correlated with the observed labels $\mathbf{y}_o$ (Klys et al., 2018).

This can be resolved by minimizing the mutual information I between latents $\mathbf{z}_o$ (equivalently $\mathbf{y}_o$) and $\mathbf{z}_l$. We can formulate this as,

$$
\begin{aligned}
&min\ I(\mathbf{y}_o; \mathbf{z}_l) \triangleq max\ H(\mathbf{y}_o|\mathbf{z}_l) \\
&= min \int_{\mathbf{z}_l} \int_{\mathbf{y}_o} p(\mathbf{z}_l)\ p(\mathbf{y}_o|\mathbf{z}_l)\ log\ p(\mathbf{y}_o|\mathbf{z}_l) d\mathbf{y}_o d\mathbf{z}_l \\
&= min \int_{\mathbf{X}} \int_{\mathbf{z}_l} \int_{\mathbf{y}_o} \begin{matrix} p(\mathbf{X})\ p(\mathbf{z}_l|\mathbf{X})\ p(\mathbf{y}_o|\mathbf{z}_l) \\ log\ p(\mathbf{y}_o|\mathbf{z}_l)\ d\mathbf{y}_o\ d\mathbf{z}_l\ d\mathbf{X} \end{matrix}
\end{aligned}
$$

Since integral over $\mathbf{z}_l$ is intractable, we replace $p(\mathbf{z}_l|\mathbf{X})$ with an approximate posterior $q(\mathbf{z}_l|\mathbf{X})$. Further, since the true distribution $p(\mathbf{y}_o|\mathbf{z}_l)$ is unknown, we approximate it by introducing a new network $q_\psi(\mathbf{y}_o|\mathbf{z}_l)$ leading to $min\ I(\mathbf{y}_o; \mathbf{z}_l)$

$$
\begin{aligned}
&\approx min \int_{\mathbf{X}} \int_{\mathbf{z}_l} \int_{\mathbf{y}_o} \begin{matrix} p(\mathbf{X})\ q(\mathbf{z}_l|\mathbf{X})\ q_\psi(\mathbf{y}_o|\mathbf{z}_l) \\ log\ q_\psi(\mathbf{y}_o|\mathbf{z}_l) d\mathbf{y}_o\ d\mathbf{z}_l\ d\mathbf{X} \end{matrix} \\
&= min\ E_{D(\mathbf{X})q(\mathbf{z}_l|\mathbf{X})} \left[\int_{\mathbf{y}_o} \begin{matrix} q_\psi(\mathbf{y}_o|\mathbf{z}_l) \\ log\ q_\psi(\mathbf{y}_o|\mathbf{z}_l)\ d\mathbf{y}_o \end{matrix} \right] \\
&\approx min\ \frac{1}{N} \sum_a \left[\begin{matrix} q_\psi(\mathbf{y}_o = a|\mathbf{z}_l{}') \\ log\ q_\psi(\mathbf{y}_o = a|\mathbf{z}_l{}') \end{matrix} \right]
\end{aligned} \tag{2}
$$

where $\mathbf{z}_l{}' \sim q(\mathbf{z}_l|\mathbf{X})$, $a \in \{0, 1, 2...A\}$, A is total number of unique classes of $\mathbf{y}_o$, N is the number of samples used for Monte Carlo estimates, and $D(\mathbf{X})$ is the underlying distribution of the input points $\mathbf{X}$. Our proposed encoder is depicted in Figure 2(b). Since we are using q_ψ to make predictions for $\mathbf{y}_o$, this network needs to be learnt itself. Hence we need to subtract an additional $q_\psi(\mathbf{y}_{o_T}|\mathbf{z}_l{}')$ from the loss function, where $\mathbf{y}_{o_T}$ is the ground truth $\mathbf{y}_o$ for the input $\mathbf{X}$. With $N = 1$ our proposed term is,

$$
\begin{aligned}
L_{MI} = &\sum_a q_\psi(\mathbf{y}_o = a|\mathbf{z}_l{}') log\ q_\psi(\mathbf{y}_o = a|\mathbf{z}_l{}') \\
&- q_\psi(\mathbf{y}_{o_T}|\mathbf{z}_l{}')
\end{aligned} \tag{3}
$$

Combining equations (1) and (3), the total loss function in our proposed model is,

$$
\begin{aligned}
L_{total} &= L_{mel} + \beta L_{KL} + \gamma L_{MI} \\
&= L_{mel} + L_{cond}
\end{aligned} \tag{4}
$$

To summarize, L_{mel} controls the quality of the mel-spectrogram produced during decoding, L_{KL} controls the features learnt in the latent variables $\mathbf{z}_l, \mathbf{z}_o$ and L_{MI} makes sure that $\mathbf{z}_l, \mathbf{z}_o$ encode different features. We will be referring to L_{mel} as the reconstruction or mel loss, L_{KL} as the KL loss and $L_{cond} = \beta L_{KL} + \gamma L_{MI}$ as the conditional loss respectively throughout this paper.

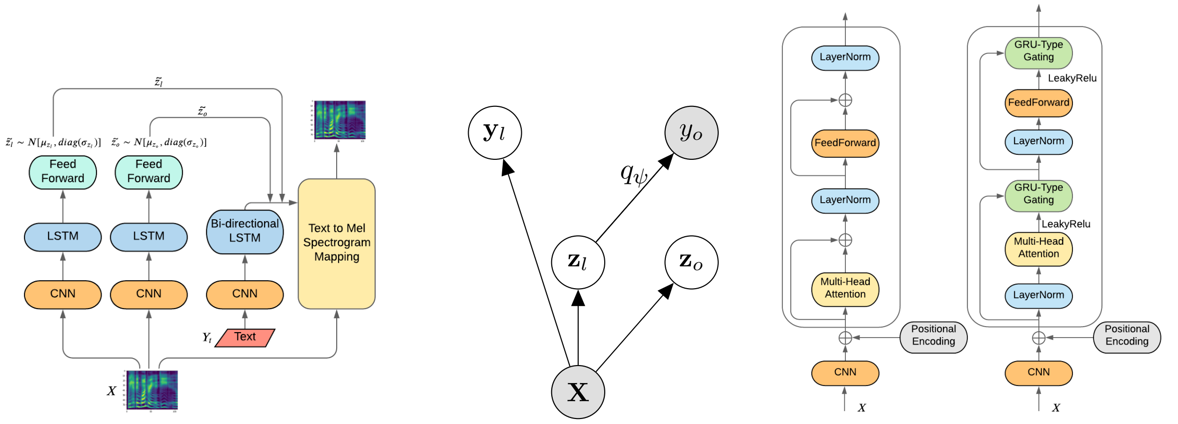

(a) Tacotron Architecture (b) Proposed Encoder (c) Original versus Proposed Transformer

Figure 2: **Left**: The Tacotron-2 architecture. VAE consists of two left parts where LSTMs predict mean μ and variance σ^2 of multivariate Gaussians $\mathcal{N}(\mu_{\mathbf{z}_l}, diag(\sigma^2_{\mathbf{z}_l})), \mathcal{N}(\mu_{\mathbf{z}_o}, diag(\sigma^2_{\mathbf{z}_o}))$. $\widetilde{\mathbf{z}_l}, \widetilde{\mathbf{z}_o}$ from this distribution are sampled and concatenated to the text encoding to conditionally learn the text to mel-spectrogram mapping. **Center**: Proposed encoder with the network q_ψ. The generator stays the same as in Figure 1. **Right**: The original and the proposed Transformers replace the LSTMs shown in the VAE of Tacotron-2 architecture.

4.2 Layer Reordering in Transformer

Introducing the above loss helps disentangle the learning of $\mathbf{z}_o$ and $\mathbf{z}_l$, but there is another problem that remains. Our experiments on MAILABS and Common Voice data, discussed in section 5.3, indicated that clusters of $\mathbf{z}_o$ corresponding to different shown labels $\mathbf{y}_o$ start sharing regions in the latent space. Hence for any given label $\mathbf{y}_o$ the sampled $\hat{\mathbf{z}_o} \sim p(\mathbf{z}_o|\mathbf{y}_o)$ may or may not belong to the style which $\mathbf{y}_o$ denotes. This leads to speech samples where the style correlated with the shown attribute $\mathbf{y}_o$ is not under control while sampling from the priors.

We tackle this problem by replacing LSTMs with Transformers. We expected that the ability of Transformers to attend to specific frames of interest where features could be localized or have a higher expression density, with a higher weight in the input speech sequence should bring down the dataset volume required for convergence by a considerable amount. Hence the lower bound on dataset size needed for modelling non overlapping clusters of $\mathbf{z}_o$ should be smaller while still keeping the sampled style under control. This should also accelerate the separation between latent clusters for larger datasets. Our experiments with vanilla Transformer-based VAEs confirm our predictions.

We next drew some inspiration from Parisotto et al. (2019) and modified the Transformer encoder. This was an attempt at changing the learning paradigm— instead of directly learning to translate $\mathbf{Y}_t$ to $\mathbf{X}$ in different $\mathbf{y}_o$ styles, we first learn to synthesize a general representation for all $\mathbf{X}$, and then learn specific deviations of each style $\mathbf{y}_o$ from this general representation. For example, instead of learning directly to speak in different accents first we learn to speak, and then we learn the subtleties of different accents. Our hypothesis was that learning different $\mathbf{y}_o$ styles should be a lot faster if a common understanding of all $\mathbf{X}$ in the dataset is gained first. The accent specific speech frames $\mathbf{X}$ (or style specific as per $\mathbf{y}_o$) should just be a slight deviation from this common representation.

Our proposed architecture is shown in Figure 2c where we switch the order of `LayerNorm` forming a direct connection between the input and the output. Due to this layer reordering if we make sure that all the modules `MHA`, `LayerNorm`, `FeedForward` are initialized with their expectation near 0, a direct path is formed early in training allowing a general representation of speech to be learnt independent of the shown labels $\mathbf{y}_o$. Now as training progresses and these modules warm up, the accent or $\mathbf{y}_o$ specific features will be learnt by conditioning the encoder.

We also introduce GRU-type gating (Chung et al., December 2014) to stabilize learning by minimizing the maximum gradient norms produced, and apply a small nonlinearity via $LeakyRelu$ at the outputs of the `MHA` and `FeedForward` modules to balance the observed trade-off between frequent gradient updates and maximum gradient

d	**Feature**	$\mu_{\mathbf{z}_l,d} - 3\sigma_{\mathbf{z}_l,d}$	$\mu_{\mathbf{z}_l,d}$	$\mu_{\mathbf{z}_l,d} + 3\sigma_{\mathbf{z}_l,d}$
0	Speaking Rate (sec)	3.0 ± 0.2	3.7 ± 0.3	4.4 ± 0.3
1	F_0 (Hz)	240.5 ± 12.57	211.4 ± 15.66	184 ± 10.43
2	Pause Duration (msec)	70 ± 3.40	79 ± 3.30	91 ± 3.50

Table 1: Length of the mel-spectrogram synthesized and pause durations increase while pitch decreases with increasing dth dimension of $\mathbf{z}_l$ from its marginal prior mean in REMMI.

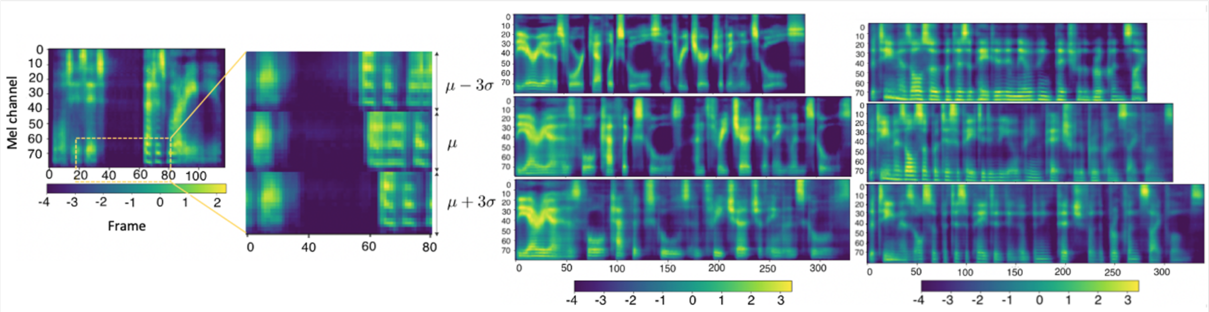

Figure 3: **Left**: Synthesized mel spectrogram for *"What is it, that is worrying you today?"* The stack of 3 mel spectrograms on the right are zoomed areas from frames 20 to 80 for each of their original mel-spectrogram. It can be seen that the pause duration denoted by the dark region increases as you synthesize the same text moving from $\mu_i - 3\sigma_i$ to $\mu_i + 3\sigma_i$. **Center**: Three mel-spectrograms synthesized for the text *"The area has four catholic schools and three church of England schools"*, corresponding to three random sampling of $\tilde{\mathbf{z}}_o, \tilde{\mathbf{z}}_l$ from their posteriors. First synthesis is considerably shorter than the second and third. Notice the different positions of voids between frames 50 and 100, and at frame 150 in the third spectrogram being considerably different. **Right**: Mel-spectrograms synthesized for the text *"The team has also participated in the opening pitch of the Brooklyn Cyclones"*. The third spectrogram shows smooth areas in the higher mel channels compared to the second and the first. These random latent sampling affects intonation and spectrogram texture.

norm[2].

5 Experiments

We refer to our proposed VAE with modifications from sections 4.1 (L_{MI} term) and 4.2 as REMMI, the vanilla Transformer with L_{MI} term as Transformer-VAE and the LSTM based state-of-the-art Tacotron-2 without L_{MI} term (Hsu et al., 2019) as LSTM-VAE. We trained each model on two datasets— 1) MAILABS (Solak, 2018 (accessed November 11, 2020) with a total 35hrs of UK and 39hrs of US speech in studio quality recorded by 4 professional speakers, 2) Common Voice (Ardila et al., 2020) with 4hrs of UK and 19hrs of US speech crowd-sourced from 477 volunteers with varying background noise, microphone qualities and other recording conditions. The input feature $\mathbf{X}$ were mel-scale spectrograms, the label $\mathbf{y}_o$ was set to be 0 for all $\mathbf{X}$ belonging to US and 1 for all UK. Dimension of $\mathbf{z}_o$ and $\mathbf{z}_l$ were picked to be 2 and 3 respectively and K = 3 for all experiments [3].

5.1 Features Learnt

Before we demonstrate our latent cluster improvements over Transformer-VAE and LSTM-VAE, we show that REMMI does learn important latent features in speech. Our experiments (focused on learning the speaking rate, the fundamental frequency F_0, and the pause duration) are summarized in Table 1. $\mu_{\mathbf{z}_l,d}$ and $\sigma_{\mathbf{z}_l,d}$ are the dth dimension mean and standard deviations of the marginal prior $p(\mathbf{z}_l) = \sum_k p(\mathbf{z}_l|\mathbf{y}_l = k)p(\mathbf{y}_l = k)$. All other dimensions of $\mathbf{z}_l$ are kept fixed at their own marginal priors while analyzing dth dimension.

For demonstrating control on speaking rate, we did 25 different synthesis for the text *"We had been wandering, indeed, in the leafless shrubbery an hour in the morning"*. It can be seen from Table 1 that the length of the synthesized mel-spectrogram increases as the value of $\mathbf{z}_l$ dimension 0 increases.

Next, we synthesized 25 texts, with 10 samples for each text to show control on pause duration and

---

[2] Importance of Gates and the specific choice of $LeakyRelu$ is discussed in the Ablation Study in Appendix D.

[3] Other hyperparameters of our VAE and training details of Tacotron-2 are given in Appendix F, G, H.

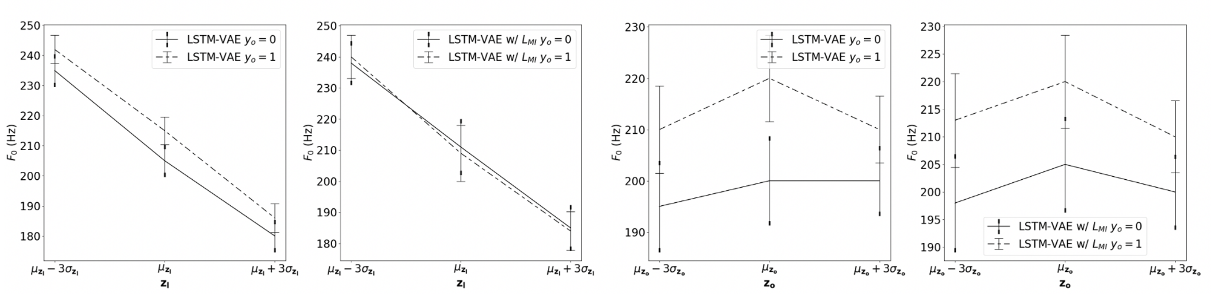

Figure 4: In LSTM-VAE F_0 encoded by $\mathbf{z}_l$ is significantly different for $\mathbf{y}_o = 0, 1$ showing that y_o specific information is encoded by $\mathbf{z}_l$. However this difference is no longer significant once we include our proposed L_{MI} terms in LSTM-VAE w/ L_{MI} experiment. $\mathbf{z}_o$ keeps showing different values of F_0 for $y_o = 0, 1$ in both LSTM-VAE and LSTM-VAE w/ L_{MI} experiments demonstrating learnt features which are conditional on y_o.

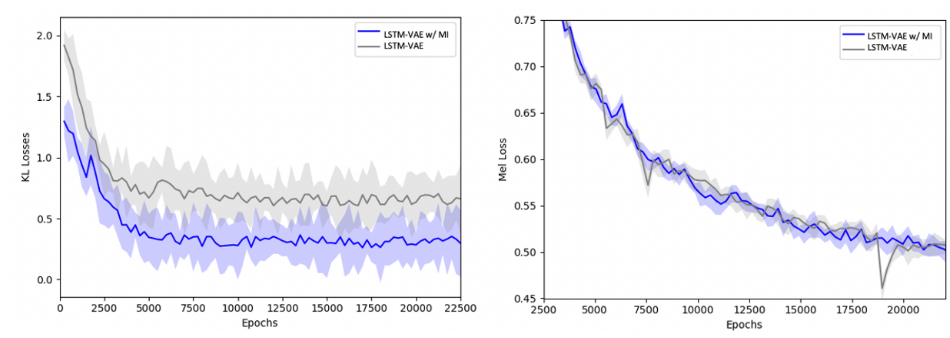

Figure 5: **Left**: Test L_{KL} versus epochs. Including L_{MI} in loss function decreases L_{KL} pointing to improved latent variables. **Right**: Test L_{mel} versus epoch. The L_{mel} remains the same even upon including L_{MI} demonstrating our proposed L_{MI} does not hurt the synthesized mel-spectrogram quality.

pitch (or the fundamental frequency F_0). For pause duration experiments each text contained at least one comma and we measured the maximum period of intermediate silence for each synthesis. To calculate F_0 we used the YIN algorithm (Guyot, 2018). In Table 1 it can be seen that the pause duration increases and F_0 decreases with increasing values of 2nd and 1st dimensions of $\mathbf{z}_l$, respectively.

Furthermore the sampled variables $\widetilde{\mathbf{z}_o}, \widetilde{\mathbf{z}_l}$ from their respective posterior distributions $q(\mathbf{z}_o|\mathbf{X}), q(\mathbf{z}_l|\mathbf{X})$ in L_{mel} gives the effect of different intonations with different speakers every time we synthesize a given text $\mathbf{Y}_t$. We demonstrate concrete examples in Figure 3.

5.2 Importance of L_{MI}

Our experiment on MAILABS dataset shows that the latent variable $\mathbf{z}_l$ starts encoding $\mathbf{y}_o$ specific features in the absence of an explicit L_{MI} term in the total loss, contrary to the expectation that $\mathbf{z}_l$ should not encode any $\mathbf{y}_o$ style specific information. As shown in Figure 4, $\mathbf{z}_l$ shows different values of F_0 for classes $\mathbf{y}_o = 0, 1$ in the absence of L_{MI}, while $\mathbf{z}_o$ continues to show accent specific values for both $\mathbf{y}_o$ classes with and without L_{MI} terms. The values in Figure 4 are plotted for a synthesis of 25 different texts with 10 samples for each text. We show similar trends for speaking rate in the Appendix.

A consequence of including L_{MI} in the loss function (4) can also be seen in the test curve of L_{KL}. We can see in Figure 5 that LSTM-VAE w/ MI has a lower value of L_{KL}. Also note that as shown in Figure 5, L_{mel} remains the same in both the experiments hence there is an overall decrease in the total loss value. We also observe that the two terms of L_{MI} in equation (3) are in contention to each other. The first term tries to learn a representation $\mathbf{z}_l$ such that it does not have any information about label $\mathbf{y}_o$ whereas the second term tries to maximize the probability of predicting true label $\mathbf{y}_o$ given $\mathbf{z}_l$. We verify from our experiments that at convergence $\mathbf{z}_l$ acts as a complete random input for estimating $\mathbf{y}_o$ with $q_\psi(\mathbf{y}_o|\mathbf{z}_l) = 0.5$ for both $\mathbf{y}_o = 0, 1$.

Model	4hrs US+4hrs UK DI	4hrs US+4hrs UK DBI	20hrs US+20hrs UK DI	20hrs US+20hrs UK DBI	39hrs US+35hrs UK DI	39hrs US+35hrs UK DBI
LSTM-VAE	0.55±0.15	2.11±0.24	1.41±0.21	1.60±0.29	2.10±0.29	1.12±0.24
Transformer-VAE	1.22±0.26	0.44±0.05	2.24±0.05	0.30±0.15	2.48±0.23	0.27±0.09
REMMI	**1.85±0.59**	**0.35±0.07**	**2.33±0.21**	**0.29±0.10**	**2.80±0.26**	**0.26±0.07**

Table 2: REMMI consistently increases DI and reduces DBI for different sizes of MAILABS dataset and performs at least 3% better (DBI for 20hrs US+20hrs UK) on MAILABS dataset compared to all existing architectures.

Model	4hrs US+4hrs UK DI	4hrs US+4hrs UK DBI	10hrs US+4hrs UK DI	10hrs US+4hrs UK DBI	19hrs US+4hrs UK DI	19hrs US+4hrs UK DBI
LSTM-VAE	0.98±0.17	83.18±13.66	0.85±0.23	85.53±15.10	0.80±0.30	98.20±24.68
Transformer-VAE	0.99±0.15	0.19±0.01	0.98±0.22	0.18±0.18	0.94±0.29	0.17±0.30
REMMI	**1.03±0.40**	**0.15±0.005**	**0.99±0.20**	**0.16±0.04**	**0.99±0.25**	**0.16±0.05**

Table 3: REMMI performs at least 4% better (DI for 4hrs US+4hrs UK Common Voice compared to Transformer-VAE) on all sizes of noisy Common Voice dataset than all existing LSTM and Transformer-VAE architectures.

Model	Overlap on MAILABS 4+4	Overlap on MAILABS 20+20	Overlap on MAILABS 39+35	Overlap on Common Voice 4+4	Overlap on Common Voice 10+4	Overlap on Common Voice 19+4
LSTM-VAE	30%	11%	0%	92%	94%	96%
Transformer-VAE	7%	0%	0%	52%	65%	81%
REMMI	**0%**	0%	0%	**47%**	**56%**	**65%**

Table 4: Overlap percentages for datasets of size $M + N$ with M hrs US and N hrs UK speech. REMMI reduces the overlap percentage by 30% for limited MAILABS dataset and by half for limited Common Voice dataset. The reduction difference for entire Common Voice dataset is 31% compared to LSTM and 16% compared to Transformer-VAE.

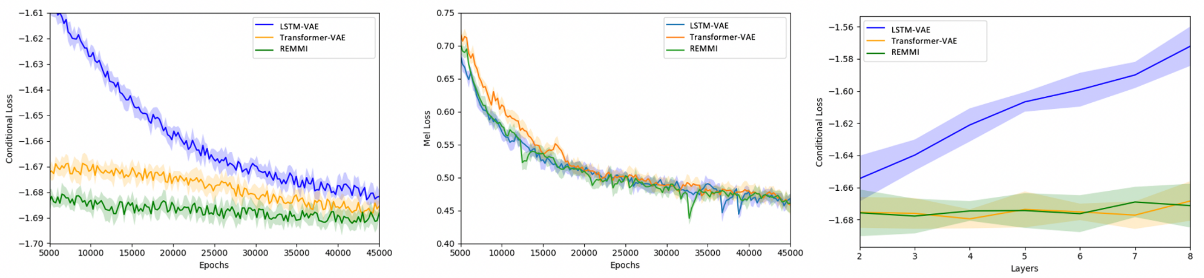

Figure 6: Loss Curves on MAILABS dataset. **Left**: Test L_{cond} versus Epochs. REMMI converges faster compared to both Transformer-VAE and LSTM-VAE. **Center**: Test L_{mel} versus Epochs. REMMI accelerates L_{cond} without compromising the mel-spectrogram quality or L_{mel}. **Right**: Test L_{cond} versus model depth. Transformer and REMMI do not overfit to a given dataset with increasing model depth unlike LSTM-VAE.

5.3 Cluster Quality

As discussed in section 4.2, we want clusters of $p(\mathbf{z}_o|\mathbf{y}_o = 0)$ and $p(\mathbf{z}_o|\mathbf{y}_o = 1)$ to be far from each other with no overlaps so that we can control $\mathbf{y}_o$ styles during synthesis. Hence we objectively measured the cluster quality with Dunn Index (DI) (Bezdek and Pal, 1995) and DB Index (DBI) (Davies and Bouldin, 1979) where $DI{=}\frac{min_{1\leq i<j\leq n} d(i,j)}{max_{1\leq k\leq n} d'(k)}$, $DBI{=}\frac{1}{n}\sum_{i=1}^{n} max_{j\neq i}\left(\frac{\sigma_i+\sigma_j}{d(\mu_i,\mu_j)}\right)$, j, i are cluster indices, $d(i,j)$ denotes the distance between the clusters i and j, n is the total number of points, $d'(k)$ is the maximal intra-cluster distance and $\mu_i, \sigma_i, \mu_j, \sigma_j$ are the means and standard deviations of the clusters i, j respectively. Thus DI is the ratio of minimal inter-cluster distance to the maximal intra-cluster distance. Similarly, DBI is the ratio of spread in each cluster to the distance between their means.

In Tables 2 and 3, we compare the test DI and DBI for different dataset sizes between

Dataset	CMOS	CS
MAILABS4hrs US+4hrs UK	0.2581 +- 0.1249	0.576 +- 0.0947
CommonVoice 4hrs US+4hrs UK	0.0541 +- 0.0966	0.327 +- 0.0888

Table 5: Positive CMOS confirms that REMMI produces speech that sounds more British than LSTM-VAE. A higher CS also shows that REMMI has better control over synthesized accent than LSTM-VAE.

REMMI, Transformer-VAE and LSTM-VAE. We see that REMMI performs consistently better than Transformer-VAE and LSTM-VAE for both MAILABS and Common Voice dataset. We also observe that as dataset size decreases, the performance gap between our REMMI and LSTM-VAE increases.

In Table 4 we calculate the percentage of overlap between clusters with test points $\hat{\mathbf{z}_o} \sim p(\mathbf{z}_o|\mathbf{y}_o = i)$ marked as overlapping with cluster $p(\mathbf{z}_o|\mathbf{y}_o = j)$ if they fall within $[\mu_{p(\mathbf{z}_o|\mathbf{y}_o=j)} - \sigma_{p(\mathbf{z}_o|\mathbf{y}_o=j)}, \mu_{p(\mathbf{z}_o|\mathbf{y}_o=j)} + \sigma_{p(\mathbf{z}_o|\mathbf{y}_o=j)}]$, with $i, j = 0, 1$. We observe that our REMMI consistently decreases the overlap regions by large margins even on challenging datasets like Common Voice, where more than 90% overlap exists for existing state-of-the-art. As discussed earlier this better separation provides improved control on synthesis and prevents uncontrolled styles when sampling speech from the priors.

5.4 Synthesis Quality

To get opinion scores on the quality of the synthesized British accent between LSTM-VAE and REMMI, we used Griffin-Lim reconstruction (Griffin and Lim, 1983) to convert the Mel spectrograms to waveforms for models trained on Common Voice 4hrs US+4hrs UK and MAILABS 4hrs US+4hrs UK data. To compare the accents, we synthesized 30 pairs of speech samples (LSTM was sample 1, REMMI was sample 2) and asked 20 Mechanical Turk (MTurk) (Crowston, 2012) participants to rate which sample sounded more British. The rating scale given to MTurk participants was: +2: 2nd sample sounds more British than 1st, +1: 2nd sounds slightly more British than 1st, 0: 2nd and 1st sound equally British, -1: 1st sounds slightly more British than 2nd, -2: 1st sounds more British than 2nd. We repeated the experiment with REMMI as sample 1 and LSTM as sample 2 (reversing the corresponding rating scale) and averaged the scores of the experiments to counter any ordering bias. We calculated the CMOS by averaging the difference in the mean scores for REMMI and LSTM-VAE.

Second, to check if REMMI provided more control on synthesized accent (whether US or British) than LSTM-VAE and provide human verification that the separation in latent clusters led to controllable synthesis, we generated 50 random pairs of (US sample, UK synthesized sample) using LSTM-VAE and REMMI each. We asked 10 MTurk participants to rate if the US and UK samples sounded different. The scale was: 0- Samples sound the same, 1- Samples sound slightly different, 2- Samples sound different. We calculated the Control Score (CS) by averaging the difference in the mean scores for REMMI and LSTM-VAE.

The resulting CMOS and CS with 95% confidence intervals in Table 5 show that in MAILABS 4hrs US+4hrs UK our approach is superior in both producing speech that sounds more British and providing controlled synthesis. In Common Voice due to noisy synthesis, LSTM-VAE and REMMI produce nearly the same accent quality, but a significantly positive CS provides better synthesis control for REMMI. In practice, this means that LSTM-VAE cannot be controlled at test time to produce US/British speech, while REMMI can be better controlled at this task.

5.5 Loss Curves

The conditional loss L_{cond} in equation (4) controls the latent variables being modelled namely $\mathbf{z}_l, \mathbf{z}_o$ and $\mathbf{y}_l$. The trend in Figure 6 for MAILABS dataset shows that REMMI has an accelerated convergence compared to both Transformer-VAE and LSTM-VAE. It can also be seen in Figure 6 that L_{mel} remains the same in all the 3 experiments, LSTM-VAE, Transformer-VAE and REMMI. This shows that while our REMMI is successful in lowering L_{cond}, it does so without hurting L_{mel} or the synthesized mel-spectrogram quality.

We also observed that for a given dataset size in LSTM-VAE, L_{cond} increases with increasing model depth which points towards inferior latent features. This trend is summarized in Figure 6 and shows that Transformer-VAE and REMMI do not overfit to a given dataset size with increasing layers.

6 Conclusion

In this work we showed that REMMI discovers disentangled latent representations of speech with uncorrelated latent variables allowing better control of speech synthesis. Our layer reordering in Transformers produces notably improved latent clusters of speaker attributes keeping the speaker styles under control on varying dataset sizes with different noise conditions. We can generate mel spectrograms for different text with controllable pitch, pause durations, speaking speed and accent. We also showed that there is a significant boost both in convergence and in the stability of the learnt representations with our proposed method. Going forward we would like to explore the application of REMMI beyond speech, e.g, image captionining with sentiments or text to image rendering with different emotions.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, M. Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. 2020. Common voice: A massively-multilingual speech corpus. In *LREC*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

J. C. Bezdek and N. R. Pal. 1995. Cluster validation with generalized dunn's indices. In *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pages 190–193.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Ju-Chieh Chou, Cheng-chieh Yeh, Hung-yi Lee, and Lin-Shan Lee. 2018. Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, pages 501–505. ISCA.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. December 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. *CoRR*, abs/1506.02216.

Kevin Crowston. 2012. Amazon mechanical turk: A research tool for organizations and information systems scholars. In *Shaping the Future of ICT Research. Methods and Approaches*, pages 210–221, Berlin, Heidelberg. Springer Berlin Heidelberg.

D. L. Davies and D. W. Bouldin. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227.

D. Griffin and Jae Lim. 1983. Signal estimation from modified short-time fourier transform. In *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 804–807.

Patrice Guyot. 2018. Fast python implementation of the yin algorithm.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Yukiya Hono, Kazuna Tsuboi, Kei Sawada, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. 2020. Hierarchical Multi-Grained Generative Model for Expressive Speech Synthesis. In *Proc. Interspeech 2020*, pages 3441–3445.

W. Hsu, Y. Zhang, R. J. Weiss, Y. Chung, Y. Wang, Y. Wu, and J. Glass. 2019. Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5901–5905.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2017. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1878–1889.

Wei-Ning Hsu, Yu Zhang, Ron Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, Patrick Nguyen, and Ruoming Pang. 2019. Hierarchical generative modeling for controllable speech synthesis. In *International Conference on Learning Representations (ICLR)*.

Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez-Moreno, and Yonghui Wu. 2018. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 4485–4495.

J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa. 2020. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 516–520.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Jack Klys, Jake Snell, and Richard Zemel. 2018. Learning latent subspaces in variational autoencoders. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6445–6455, Red Hook, NY, USA. Curran Associates Inc.

Simon Leglaive, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. 2020. A recurrent variational autoencoder for speech enhancement. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 371–375. IEEE.

Yingzhen Li and Stephan Mandt. 2018. Disentangled sequential autoencoder. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5656–5665. PMLR.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. 2019. Stabilizing transformers for reinforcement learning. *CoRR*, abs/1910.06764.

Seungwon Park, Dooyoung Kim, and Myun chul Joe. 2020. Cotatron: Transcription-guided speech encoder for any-to-many voice conversion without parallel data. In *INTERSPEECH*.

J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu. 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783.

RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron J. Weiss, Rob Clark, and Rif A. Saurous. 2018. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron.

Imdat Solak. 2018 (accessed November 11, 2020). *The M-AILABS Speech Dataset*. `https://www.caito.de/2019/01/the-m-ailabs-speech-dataset/`.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

G. Sun, Y. Zhang, R. J. Weiss, Y. Cao, H. Zen, and Y. Wu. 2020. Fully-hierarchical fine-grained prosody modeling for interpretable speech synthesis. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6264–6268.

Tianming Wang and Xiaojun Wan. 2019. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5233–5239. International Joint Conferences on Artificial Intelligence Organization.

Yuxuan Wang, Daisy Stanton, Yu Zhang, R. J. Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A. Saurous. 2018. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5167–5176. PMLR.

Ya-Jie Zhang, Shifeng Pan, Lei He, and Zhen-Hua Ling. 2019. Learning latent representations for style control and transfer in end-to-end speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 6945–6949. IEEE.

Appendix

A Variational Lower Bound

For an input text sequence Y_t and an observed categorical label y_o frames X can be learnt via the joint distribution $log\ p(X, Y_t, y_o)$. Additional latent variables z_o and z_l can be introduced to discover meaningful representations during this process. Here z_o is a continuous latent learnt on top of shown labels y_o, hence the features z_o discovers is correlated with what is shown to the model via y_o, while z_l is a completely unsupervised continuous variable learnt on top of standard Expectation-Maximization style latent mixture components y_l. Note that y_l is a K-way categorical discrete variable. The variational lower bound can then be formulated as,

$$\begin{aligned}
&log\ p(X|Y_t, y_o) \geq \mathbb{E}_{q(z_o|X)q(z_l|X)q(y_l|X)} \\
&\quad \left[log \frac{p(X|Y_t, z_o, z_l)p(z_o|y_o)p(z_l|y_l)p(y_l)}{q(z_o|X)q(z_l|X)q(y_l|X)} \right] \\
&\quad = \mathbb{E}_{q(z_o|X)q(z_l|X)}[log\ p(X|Y_t, z_o, z_l)] \qquad (5) \\
&\quad - D_{KL}(q(z_o|X) \;||\; p(z_o|y_o)) \\
&\quad - \mathbb{E}_{q(y_l|X)}[D_{KL}(q(z_l|X) \;||\; p(z_l|y_l))] \\
&\quad - D_{KL}(q(y_l|X) \;||\; p(y_l)) \\
&\quad \approx \ log\ p(X|Y_t, \widetilde{z_o}, \widetilde{z_l}) \qquad (6) \\
&\quad - \sum_{y_l=1}^{K} q(y_l|X) D_{KL}[\ q(z_l|X) \;||\; p(z_l|y_l)\ ] \\
&\qquad\qquad\qquad\qquad\qquad\qquad (7) \\
&\quad - D_{KL}[\ q(y_l|X) \;||\; p(y_l)\ ] \\
&\quad - D_{KL}[q(z_o|X) \;||\; p(z_o|y_o)\ ] \\
&\quad = -L_{mel} - L_{KL}
\end{aligned}$$

B Gated Architecture

In the past multiplicative interactions have been successful at stabilizing learning across different architectures (Cho et al., 2014; Srivastava et al., 2015). This motivated us to try out GRU-type gating at the heads of the proposed Transformers. The outputs at the GRU-type gating is controlled by the following equation,

$$\begin{aligned}
r &= \sigma(W_r^{(l)} y + U_r^{(l)} x), \\
z &= \sigma(W_z^{(l)} y + U_z^{(l)} x - b_g^{(l)}), \\
\hat{h} &= tanh(W_g^{(l)} y + U_g^{(l)} (r \odot x)) \\
g^{(l)}(x, y) &= (1 - z) \odot x + z \odot \hat{h}
\end{aligned}$$

where r stands for the reset gates, z is the update gates, $\hat{h}$ is the candidate activation similar to other recurrent units (Bahdanau et al., 2016). The overall gate activation $g(x, y)$ takes input x as the residual connection and y the output of the `FeedForward` or `Multi-Head Attention` modules. $g(x, y)$ is basically an interpolation between the previous activations $\hat{h}$ and the residual input x.

C Speaking Rate for $\mathbf{y}_o = 0, 1$

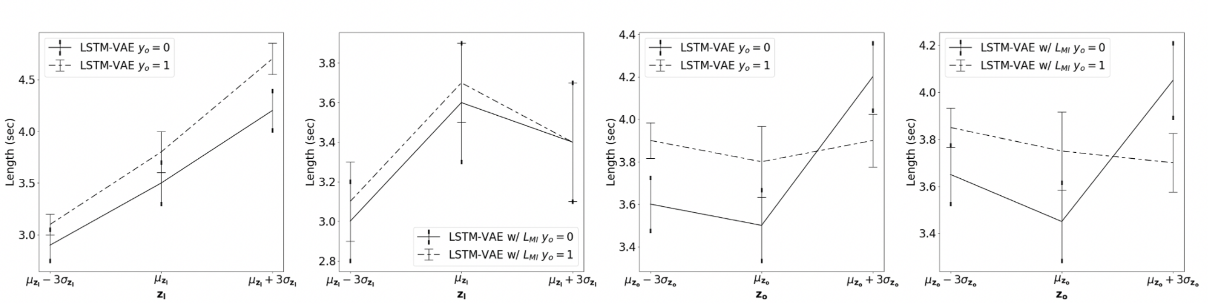

Figure 7: Length of mel-spectrogram synthesized by $\mathbf{z}_l$ in LSTM-VAE for MAILABS is significantly different for $\mathbf{y}_o = 0, 1$ showing that y_o specific information is encoded by $\mathbf{z}_l$. However this difference is no longer significant once we include our proposed L_{MI} terms in LSTM-VAE w/ L_{MI} experiment. $\mathbf{z}_o$ keeps showing different lengths for $y_o = 0, 1$ in both LSTM-VAE and LSTM-VAE w/ L_{MI} experiments demonstrating learnt features which are conditional on y_o.

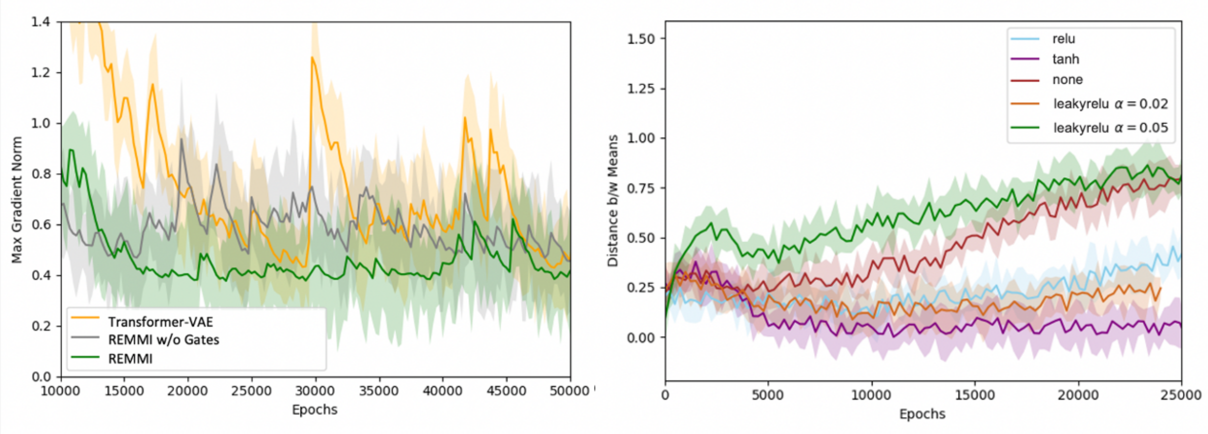

Figure 8: **Left**: Lower gradient norm for REMMI w/ Gates along with smaller variance compared to Transformers-VAE and REMMI w/o Gates. **Right**: Distance between the means of $z_o|y_o$ for $y_o = 0, 1$ for different activation functions at the output of `Multi-Head Attention` and `FeedForward` modules. We see that $LeakyRelu$ with $\alpha = 0.05$ performs the best in segregating the prior clusters among all experiments.

D Ablation Study

D.1 Importance of Gates

Our comparison of Gated architectures with non-Gated ones in Figure 8 shows that the maximum gradient norm which directly influences the convergence is much lower and stable with a lower variance for REMMI (which includes gates) compared to REMMI without (w/o) Gates and Transformer-VAE.

D.2 Choosing the Right Activation

In Figure 8 we see that the distance between $z_o|y_o$ cluster means is very small when the output from `Multi-Head Attention` and `FeedForward` modules are fed to `GRU-Type Gating` layers without any non linearity. Hence our choice of this non linearity was inspired by the trade-off between number of gradient updates and the maximum gradient norm. We see in Table 6 that $relu$ has a high maximum gradient norm ∇_{norm} which led to convergence instability and small distance between $z_o|y_o$ cluster means. But for $tanh$, almost all activations were producing gradient updates and this frequent update was leading to small cluster distance as shown in Figure 8. Hence we needed a function somewhere between $relu$ and $tanh$, which has a small gradient norm while also having fewer gradient updates. $LeakyRelu$ turns out to be the best candidate for this with its high distance between means as shown in Figure 8.

Experiment	% activation	max ∇_{norm}
relu	84.5 (< 0)	40.96
tanh	0 (>+2,<-2)	10.68
leakyrelu	-	7.17

Table 6: Comparing the percentage of activations for which gradient saturates and maximum gradient norm ∇_{norm}

E Compute Information

We ran all our experiments on NVIDIA Tesla V100 GPU with 16GB of GPU memory. Our LSTM-VAE (both with and without L_{MI}) experiments take average 5.81sec/step (seconds per step) with convergence near 40k steps. Transformer-VAE takes an average 2.81sec/step with convergence near 25k steps, and REMMI takes average 2.81sec/step with convergence near 25k steps. Total number of parameters are 28.03mn (million) for LSTM-VAE w/ and w/o MI, 27.84mn for Tranformer-VAE and 28.03mn for REMMI.

F Audio Hyperparameters

Parameter	Value
num mels	80
num freq	1025
max mel frames	900
silence threshold	2
n fft	2048
hop size	275
win size	1100
sample rate	16000
magnitude power	2.0
trim silence	True
trim fft size	2048
trim hop size	512
trim top db	50
preemphasize	True
preemphasis	0.97
min level db	-100
ref level db	20
fmin	55
fmax	7600
power	1.5

Table 7: Parameters for converting wav files to mel-spectrograms

G Tacotron-2 Hyperparameters

Parameter	Value
batch size	64
output frames per step	4
max training iterations	100k
optimizer	Adam
β_1	0.9
β_2	0.999
ϵ	1e-6
L2 regularization weight	1-e6
learning rate decay	exponential
initial learning rate	1e-3
decay start epoch	40k
decay epochs	18k
final learning rate	1e-4
clip gradients	True
teacher forcing	constant at 1

Table 8: Hyperparameters common for all experiments

H VAE Hyperparameters

Parameter	**Value**
z_l dim	3
z_o dim	2
$\|y_o\|$	2 (UK, US)
z_o, z_l convolution channels	128
activation function for convolution	$tanh$
kernel size	3x3
MC estimate num_samples	1
num_units for LSTM	128
min logvariance for $q(z_l\|X)$	-4
min logvariance for $q(z_o\|X)$	-6
initial mean for $p(z_l\|y_l)$	
$p(z_l\|y_l = 0)$	(1,0,0)
$p(z_l\|y_l = 1)$	(0,1,0)
$p(z_l\|y_l = 2)$	(0,0,1)
initial logvariance for $p(z_l\|y_l)$	-4
initial mean for $q(z_o\|y_o)$	
$p(z_o\|y_o = 0)$	(-0.5, -0.5)
$p(z_o\|y_o = 1)$	(+0.5, +0.5)
initial logvariance for $p(z_o\|y_o)$	-5
dropout	0.1
zoneout (for LSTM)	0.1
q_ψ num_layers	4
q_ψ num_units	8
q_ψ activations	$tanh$
Transformer d_model	64
Transformer num_heads	4
Transformer feedforward_dimension	256
max positional encoding	584

Table 9: Hyperparameters used for our VAEs

How to Split: the Effect of Word Segmentation on Gender Bias in Speech Translation

Marco Gaido[1,2] †, Beatrice Savoldi[1,2] †, Luisa Bentivogli[1], Matteo Negri[1], Marco Turchi[1]
[1]Fondazione Bruno Kessler, Trento, Italy
[2]University of Trento, Italy
{mgaido,bsavoldi,bentivo,negri,turchi}@fbk.eu

Abstract

Having recognized gender bias as a major issue affecting current translation technologies, researchers have primarily attempted to mitigate it by working on the data front. However, whether algorithmic aspects concur to exacerbate unwanted outputs remains so far under-investigated. In this work, we bring the analysis on gender bias in automatic translation onto a seemingly neutral yet critical component: word segmentation. Can segmenting methods influence the ability to translate gender? Do certain segmentation approaches penalize the representation of feminine linguistic markings? We address these questions by comparing 5 existing segmentation strategies on the target side of speech translation systems. Our results on two language pairs (English-Italian/French) show that state-of-the-art sub-word splitting (BPE) comes at the cost of higher gender bias. In light of this finding, we propose a combined approach that preserves BPE overall translation quality, while leveraging the higher ability of character-based segmentation to properly translate gender.

Bias Statement.[1] We study the effect of segmentation methods on the ability of speech translation (ST) systems to translate masculine and feminine forms referring to human entities. In this area, structural linguistic properties interact with the perception and representation of individuals (Gygax et al., 2019; Corbett, 2013; Stahlberg et al., 2007). Thus, we believe they are relevant gender expressions, used to communicate about the self and others, and by which the sociocultural and political reality of gender is negotiated (Hellinger and Motschenbacher, 2015).

Accordingly, we consider a model that systematically and disproportionately favours masculine over feminine forms to be biased, as it fails to properly recognize women. From a technical perspective, such behaviour deteriorates models' performance. Most importantly, however, from a human-centered view, real-world harms are at stake (Crawford, 2017), as translation technologies are unequally beneficial across gender groups and reduce feminine visibility, thus contributing to misrepresent an already socially disadvantaged group.

This work is motivated by the intent to shed light on whether issues in the generation of feminine forms are also a by-product of current algorithms and techniques. In our view, architectural improvements of ST systems should also account for the trade-offs between overall translation quality and gender representation: our proposal of a model that combines two segmentation techniques is a step towards this goal.

Note that technical mitigation approaches should be integrated with the long-term multidisciplinary commitment (Criado-Perez, 2019; Benjamin, 2019; D'Ignazio and Klein, 2020) necessary to radically address bias in our community. Also, we recognize the limits of working on binary gender, as we further discuss in the ethic section (§8).

1 Introduction

The widespread use of language technologies has motivated growing interest on their social impact (Hovy and Spruit, 2016; Blodgett et al., 2020), with gender bias representing a major cause of concern (Costa-jussà, 2019; Sun et al., 2019). As regards translation tools, focused evaluations have exposed that speech translation (ST) – and machine translation (MT) – models do in fact overproduce masculine references in their outputs (Cho et al., 2019; Bentivogli et al., 2020), except for feminine asso-

---

†The authors contributed equally.

[1] As suggested by (Blodgett et al., 2020) and required for other venues (Hardmeier et al., 2021), we formulate our bias statement.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3576–3589
August 1–6, 2021.

ciations perpetuating traditional gender roles and stereotypes (Prates et al., 2020; Stanovsky et al., 2019). In this context, most works identified *data* as the primary source of gender asymmetries. Accordingly, many pointed out the misrepresentation of gender groups in datasets (Garnerin et al., 2019; Vanmassenhove et al., 2018), focusing on the development of data-centred mitigating techniques (Zmigrod et al., 2019; Saunders and Byrne, 2020).

Although data are not the only factor contributing to generate bias (Shah et al., 2020; Savoldi et al., 2021), only few inquiries devoted attention to other technical components that exacerbate the problem (Vanmassenhove et al., 2019) or to architectural changes that can contribute to its mitigation (Costa-jussà et al., 2020b). From an algorithmic perspective, Roberts et al. (2020) additionally expose how "taken-for-granted" approaches may come with high overall translation quality in terms of BLEU scores, but are actually detrimental when it comes to gender bias.

Along this line, we focus on ST systems and inspect a core aspect of neural models: word segmentation. Byte-Pair Encoding (BPE) (Sennrich et al., 2016) represents the *de-facto* standard and has recently shown to yield better results compared to character-based segmentation in ST (Di Gangi et al., 2020). But does this hold true for gender translation as well? If not, why?

Languages like French and Italian often exhibit comparatively complex feminine forms, derived from the masculine ones by means of an additional suffix (e.g. en: *professor*, fr: *professeur* M vs. *professeure* F). Additionally, women and their referential linguistic expressions of gender are typically under-represented in existing corpora (Hovy et al., 2020). In light of the above, purely statistical segmentation methods could be unfavourable for gender translation, as they can break the morphological structure of words and thus lose relevant linguistic information (Ataman et al., 2017). Indeed, as BPE merges the character sequences that co-occur more frequently, rarer or more complex feminine-marked words may result in less compact sequences of tokens (e.g. en: *described*, it: *des@@critto* M vs. *des@@crit@@ta* F). Due to such typological and distributive conditions, may certain splitting methods render feminine gender less probable and hinder its prediction?

We address such questions by implementing different families of segmentation approaches employed on the decoder side of ST models built on the same training data. By comparing the resulting models both in terms of overall translation quality and gender accuracy, we explore whether a so far considered irrelevant aspect like word segmentation can actually affect gender translation. As such, **(1)** we perform the first comprehensive analysis of the results obtained by 5 popular segmentation techniques for two language directions (en-fr and en-it) in ST. **(2)** We find that the target segmentation method is indeed an important factor for models' gender bias. Our experiments consistently show that BPE leads to the highest BLEU scores, while character-based models are the best at translating gender. Preliminary analyses suggests that the isolation of the morphemes encoding gender can be a key factor for gender translation. **(3)** Finally, we propose a multi-decoder architecture able to combine BPE overall translation quality and the higher ability to translate gender of character-based segmentation.

2 Background

Gender bias. Recent years have seen a surge of studies dedicated to gender bias in MT (Gonen and Webster, 2020; Rescigno et al., 2020) and ST (Costa-jussà et al., 2020a). The primary source of such gender imbalance and adverse outputs has been identified in the training data, which reflect the under-participation of women – e.g. in the media (Madaan et al., 2018), sexist language and gender categories overgeneralization (Devinney et al., 2020). Hence, preventive initiatives concerning data documentation have emerged (Bender and Friedman, 2018), and several mitigating strategies have been proposed by training models on *ad-hoc* gender-balanced datasets (Saunders and Byrne, 2020; Costa-jussà and de Jorge, 2020), or by enriching data with additional gender information (Moryossef et al., 2019; Vanmassenhove et al., 2018; Elaraby and Zahran, 2019; Saunders et al., 2020; Stafanovičs et al., 2020).

Comparatively, very little work has tried to identify concurring factors to gender bias going beyond data. Among those, Vanmassenhove et al. (2019) ascribes to an algorithmic bias the loss of less frequent feminine forms in both phrase-based and neural MT. Closer to our intent, two recent works pinpoint the impact of models' components and inner mechanisms. Costa-jussà et al. (2020b) investigate the role of different architectural designs in multilingual MT, showing that language-specific encoder-decoders (Escolano et al., 2019) better translate gender than shared models (Johnson et al., 2017), as the former retain more gender information in the source embeddings and keep more diversion in the attention. Roberts et al. (2020), on the other hand, prove that the adoption of beam search instead of sampling – although beneficial in terms of BLEU scores – has an impact on gender bias. Indeed, it leads models to an extreme operating point that exhibits zero variability and in which they tend to generate the more frequent (masculine) pronouns. Such studies therefore expose largely unconsidered aspects as factors contributing to gender bias in automatic translation, identifying future research directions for the needed countermeasures.

To the best of our knowledge, no prior work has taken into account if it may be the case for segmentation methods as well. Rather, prior work in ST (Bentivogli et al., 2020) compared gender translation performance of cascade and direct systems using different segmentation algorithms, disregarding their possible impact on final results.

Segmentation. Although early attempts in neural MT employed word-level sequences (Sutskever et al., 2014; Bahdanau et al., 2015), the need for open-vocabulary systems able to translate rare/unseen words led to the definition of several word segmentation techniques. Currently, the statistically motivated approach based on byte-pair encoding (BPE) by Sennrich et al. (2016) represents the *de facto* standard in MT. Recently, its superiority to character-level (Costa-jussà and Fonollosa, 2016; Chung et al., 2016) has been also proved in the context of ST (Di Gangi et al., 2020). However, depending on the languages involved in the translation task, the data conditions, and the linguistic properties taken into account, BPE greedy procedures can be suboptimal. By breaking the surface of words into plausible semantic units, linguistically motivated segmentations (Smit et al., 2014; Ataman et al., 2017) were proven more effective for low-resource and morphologically-rich languages (e.g. agglutinative languages like Turkish), which often have a high level of sparsity in the lexical distribution due to their numerous derivational and inflectional variants. Moreover, fine-grained analyses comparing the grammaticality of character, morpheme and BPE-based models exhibited different capabilities. Sennrich (2017) and Ataman et al. (2019) show the syntactic advantage of BPE in managing several agreement phenomena in German, a language that requires resolving long range dependencies. In contrast, Belinkov et al. (2020) demonstrate that while subword units better capture semantic information, character-level representations perform best at generalizing morphology, thus being more robust in handling unknown and low-frequency words. Indeed, using different atomic units does affect models' ability to handle specific linguistic phenomena. However, whether low gender translation accuracy can be to a certain extent considered a by-product of certain compression algorithms is still unknown.

3 Language Data

As just discussed, the effect of segmentation strategies can vary depending on language typology (Ponti et al., 2019) and data conditions. To inspect the interaction between word segmentation and gender expressions, we thus first clarify the properties of grammatical gender in the two languages of our interest: French and Italian. Then, we verify their representation in the datasets used for our experiments.

3.1 Languages and Gender

The extent to which information about the gender of referents is grammatically encoded varies across languages (Hellinger and Motschenbacher, 2015; Gygax et al., 2019). Unlike English – whose gender distinction is chiefly displayed via pronouns (e.g. *he/she*) – fully grammatical gendered languages like French and Italian systematically articulate such semantic distinction on several parts of speech (gender agreement) (Hockett, 1958; Corbett, 1991). Accordingly, many lexical items exist in both feminine and masculine variants, overtly marked through morphology (e.g. en: *the tired kid sat down*; it: *il bimbo stanco si è seduto* M vs. *la bimba stanca si è seduta* F). As the example shows, the word forms are distinguished by two morphemes (*–o*, *–a*), which respectively represent the most common inflections for Italian masculine and feminine markings.[2] In French, the morphological mechanism is slightly different (Schafroth, 2003), as it relies on an additive suffixation on top of masculine words to express feminine gender (e.g. en: *an expert is gone*, fr: *un expert est allé* M vs.

---

[2] In a fusional language like Italian, one single morpheme can denote several properties as, in this case, gender and singular number (the plural forms would be *bimbi* vs. *bimbe*).

une experte est allée F). Hence, feminine French forms require an additional morpheme. Similarly, another productive strategy – typical for a set of personal nouns – is the derivation of feminine words via specific affixes for both French (*e.g. –eure, –euse*)[3] and Italian (*–essa, –ina, –trice*) (Schafroth, 2003; Chini, 1995), whose residual evidence is still found in some English forms (e.g. *heroine, actress*) (Umera-Okeke, 2012).

In light of the above, translating gender from English into French and Italian poses several challenges to automatic models. First, gender translation does not allow for one-to-one mapping between source and target words. Second, the richer morphology of the target languages increases the number of variants and thus data sparsity. Hereby, the question is whether – and to what extent – statistical word segmentation differently treats the less frequent variants. Also, considering the morphological complexity of some feminine forms, we speculate whether linguistically unaware splitting may disadvantage their translation. To test these hypotheses, below we explore if such conditions are represented in the ST datasets used in our study.

3.2 Gender in Used Datasets

MuST-SHE (Bentivogli et al., 2020) is a gender-sensitive benchmark available for both en-fr and en-it (1,113 and 1,096 sentences, respectively). Built on naturally occurring instances of gender phenomena retrieved from the TED-based MuST-C corpus (Cattoni et al., 2020),[4] it allows to evaluate gender translation on qualitatively differentiated and balanced masculine/feminine forms. An important feature of MuST-SHE is that, for each reference translation, an almost identical "wrong" reference is created by swapping each annotated gender-marked word into its opposite gender. By means of such wrong reference, for each target language we can identify ∼2,000 pairs of gender forms (e.g. en: *tired*, fr: *fatiguée* vs. *fatigué*) that we compare in terms of *i)* length, and *ii)* frequency in the MuST-C training set.

As regards frequency, we asses that, for both language pairs, the types of feminine variants are less frequent than their masculine counterpart in over 86% of the cases. Among the exceptions, we find words that are almost gender-exclusive (e.g. *pregnant*) and some problematic or socially connoted activities (e.g. *raped, nurses*). Looking at words' length, 15% of Italian feminine forms result to be longer than masculine ones, whereas in French this percentage amounts to almost 95%. These scores confirm that MuST-SHE reflects the typological features described in §3.1.

4 Experiments

All the direct ST systems used in our experiments are built in the same fashion within a controlled environment, so to keep the effect of different word segmentations as the only variable. Accordingly, we train them on the MuST-C corpus, which contains 492 hours of speech for en-fr and 465 for en-it. Concerning the architecture, our models are based on Transformer (Vaswani et al., 2017). For the sake of reproducibility, we provide extensive details about the ST models and hyper-parameters' choices in the Appendix §*A*.[5]

4.1 Segmentation Techniques

To allow for a comprehensive comparison of word segmentation's impact on gender bias in ST, we identified three substantially different categories of splitting techniques. For each of them, we hereby present the candidates selected for our experiments.

Character Segmentation. Dissecting words at their maximal level of granularity, **character-based** solutions have been first proposed by Ling et al. (2015) and Costa-jussà and Fonollosa (2016). This technique proves simple and particularly effective at generalizing over unseen words. On the other hand, the length of the resulting sequences increases the memory footprint, and slows both the training and inference phases. We perform our segmentation by appending "@@ " to all characters but the last of each word.

Statistical Segmentation. This family comprises data-driven algorithms that generate statistically significant subwords units. The most popular one is **BPE** (Sennrich et al., 2016),[6] which proceeds by merging the most frequently co-occurring characters or character sequences. Recently, He et al. (2020) introduced the Dynamic Programming Encoding **(DPE)** algorithm, which performs

---

[3] French also requires additional modification on feminine forms due to phonological rules (e.g. en: *chef/spy*, fr: *cheffe/espionne* vs. *chef/espion*).

[4] Further details about these datasets are provided in §8.

[5] Source code available at `https://github.com/mgaido91/FBK-fairseq-ST/tree/acl_2021`.

[6] We use SentencePiece (Kudo and Richardson, 2018): `https://github.com/google/sentencepiece`.

competitively and was claimed to accidentally produce more linguistically-plausible subwords with respect to BPE. DPE is obtained by training a mixed character-subword model. As such, the computational cost of a DPE-based ST model is around twice that of a BPE-based one. We trained the DPE segmentation on the transcripts and the target translations of the MuST-C training set, using the same settings of the original paper.[7]

Morphological Segmentation. A third possibility is linguistically-guided tokenization that follows morpheme boundaries. Among the unsupervised approaches, one of the most widespread tools is **Morfessor** (Creutz and Lagus, 2005), which was extended by Ataman et al. (2017) to control the size of the output vocabulary, giving birth to the **LMVR** segmentation method. These techniques have outperformed other approaches when dealing with low-resource and/or morphologically-rich languages (Ataman and Federico, 2018). In other languages, they are not as effective, so they are not widely adopted. Both Morfessor and LMVR have been trained on the MuST-C training set.[8]

	en-fr	**en-it**
# tokens[9]	5.4M	4.6M
# types	96K	118K
BPE	8,048	8,064
Char	304	256
DPE	7,864	8,008
Morfessor	26,728	24,048
LMVR	21,632	19,264

Table 1: Resulting dictionary sizes.

For fair comparison, we chose the optimal vocabulary size for each method (when applicable). Following (Di Gangi et al., 2020), we employed 8k merge rules for BPE and DPE, since the latter requires an initial BPE segmentation. In LMVR, instead, the desired target dimension is actually only an upper bound for the vocabulary size. We tested 32k and 16k, but we only report the results with 32k as it proved to be the best configuration both in terms of translation quality and gender accuracy. Finally, character-level segmentation and Morfessor do not allow to determine the vocabulary size. Table 1 shows the size of the resulting dictionaries.

	en-fr			**en-it**		
	M-C	M-SHE	Avg.	M-C	M-SHE	Avg.
BPE	**30.7**	25.9	**28.3**	21.4	**21.8**	21.6
Char	29.5	24.2	26.9	21.3	20.7	21.0
DPE	29.8	25.3	27.6	21.9	21.7	**21.8**
Morfessor	29.7	25.7	27.7	21.7	21.4	21.6
LMVR	30.3	**26.0**	28.2	**22.0**	21.5	**21.8**

Table 2: SacreBLEU scores on MuST-C tst-COMMON (M-C) and MuST-SHE (M-SHE) for en-fr and en-it.

4.2 Evaluation

We are interested in measuring both *i)* the overall translation quality obtained by different segmentation techniques, and *ii)* the correct generation of gender forms. We evaluate translation quality on both the MuST-C *tst-COMMON* set (2,574 sentences for en-it and 2,632 for en-fr) and MuST-SHE (§3.2), using SacreBLEU (Post, 2018).[10]

For fine-grained analysis on gender translation, we rely on gender accuracy (Gaido et al., 2020).[11] We differentiate between two categories of phenomena represented in MuST-SHE. Category (1) contains first-person references (e.g. *I'm a student*) to be translated according to the speakers' preferred linguistic expression of gender. In this context, ST models can leverage speakers' vocal characteristics as a gender cue to infer gender translation.[12] Gender phenomena of Category (2), instead, shall be translated in concordance with other gender information in the sentence (e.g. *she/he is a student).*

5 Comparison of Segmentation Methods

Table 2 shows the overall **translation quality** of ST systems trained with distinct segmentation techniques. BPE comes out as competitive as LMVR for both language pairs. On averaged results, it exhibits a small gap (0.2 BLEU) also with DPE on en-it, while it achieves the best performance on en-fr. The disparities are small though: they range within 0.5 BLEU, apart from Char standing ∼1 BLEU below. Compared to the scores reported by Di Gangi et al. (2020), the Char gap is however smaller. As our results are considerably higher than theirs, we believe that the reason for such differences lies in a sub-optimal fine-tuning of their hyper-parameters. Overall, in light of the trade-off between computational cost (LMVR and DPE require a dedicated training phase for data segmen-

---

[7] See `https://github.com/xlhex/dpe`.

[8] We used the parameters and commands suggested in `https://github.com/d-ataman/lmvr/blob/master/examples/example-train-segment.sh`

[9] Here "tokens" refers to the number of words in the corpus, and not to the unit resulting from subword tokenization.

[10] `BLEU+c.mixed+#.1+s.exp+tok.13a+v.1.4.3.`

[11] Evaluation script available with the MuST-SHE release.

[12] Although they do not emerge in our experimental settings, the potential risks of such capability are discussed in §8.

tation) and average performance (BPE achieves winning scores on en-fr and competitive for en-it), we hold BPE as the best segmentation strategy in terms of general translation quality for direct ST.

Turning to **gender translation**, the gender accuracy scores presented in Table 3 exhibit that all ST models are clearly biased, with masculine forms (M) disproportionately produced across language pairs and categories. However, we intend to pinpoint the relative gains and losses among segmenting methods. Focusing on overall accuracy (ALL), we see that Char – despite its lowest performance in terms of BLEU score – emerges as the favourite segmentation for gender translation. For French, however, DPE is only slightly behind. Looking at morphological methods, they surprisingly do not outperform the statistical ones. The greatest variations are detected for feminine forms of Category 1 (1F), where none of the segmentation techniques reaches 50% of accuracy, meaning that they are all worse than a random choice when the speaker should be addressed by feminine expressions. Char appears close to such threshold, while the others (apart from DPE in French) are significantly lower.

These results illustrate that target segmentation is a relevant parameter for gender translation. In particular, they suggest that Char segmentation improves models' ability to learn correlations between the received input and gender forms in the reference translations. Although in this experiment models rely only on speakers' vocal characteristics to infer gender – which we discourage as a cue for gender translation for real-world deployment (see §8) – such ability shows a potential advantage for Char, which could be better redirected toward learning correlations with reliable gender meta-information included in the input. For instance, in a scenario in which meta-information (e.g. a gender tag) is added to the input to support gender translation, a Char model might better exploit this information. Lastly, our evaluation reveals that, unlike previous ST studies (Bentivogli et al., 2020), a proper comparison of models' gender translation potentialities requires adopting the same segmentation. Our question then becomes: What makes Char segmentation less biased? What are the tokenization features determining a better/worse ability in generating the correct gender forms?

Lexical diversity. We posit that the limited generation of feminine forms can be framed as an issue of data sparsity, whereas the advantage of Char-based segmentation ensues from its ability to handle less frequent and unseen words (Belinkov et al., 2020). Accordingly, Vanmassenhove et al. (2018); Roberts et al. (2020) link the loss of *linguistic diversity* (i.e. the range of lexical items used in a text) with the overfitted distribution of masculine references in MT outputs.

	en-fr				
	ALL	1F	1M	2F	2M
BPE	65.18	37.17	75.44	61.20	80.80
Char	**68.85**	48.21	74.78	65.89	**81.03**
DPE	68.55	**49.12**	70.29	**66.22**	80.90
Morfessor	67.05	42.73	75.11	63.02	80.98
LMVR	65.38	32.89	**76.96**	61.87	79.95
	en-it				
BPE	67.47	33.17	88.50	60.26	81.82
Char	**71.69**	**48.33**	85.07	**64.65**	**84.33**
DPE	68.86	44.83	81.58	59.32	82.62
Morfessor	65.46	36.61	81.04	56.94	79.61
LMVR	69.77	39.64	**89.00**	63.85	83.03

Table 3: Gender accuracy (%) for MuST-SHE Overall (ALL), Category 1 and 2 on en-fr and en-it.

To explore such hypothesis, we compare the lexical diversity (LD) of our models' translations and MuST-SHE references. To this aim, we rely on Type/Token ratio **(TTR)** – (Chotlos, 1944; Templin, 1957), and the more robust Moving Average TTR **(MATTR)** – (Covington and McFall, 2010).[13]

As we can see in Table 4, character-based models exhibit the highest LD (the only exception is DPE with the less reliable TTR metric on en-it). However, we cannot corroborate the hypothesis formulated in the above-cited studies, as LD scores do not strictly correlate with gender accuracy (Table 3). For instance, LMVR is the second-best in terms of gender accuracy on en-it, but shows a very low lexical diversity (the worst according to MATTR and second-worst according to TTR).

	en-fr		**en-it**	
	TTR	MATTR	TTR	MATTR
M-SHE Ref	*16.12*	*41.39*	*19.11*	*46.36*
BPE	14.53	39.69	17.46	44.86
Char	**14.97**	**40.60**	17.75	**45.65**
DPE	14.83	40.02	**18.07**	45.12
Morf	14.38	39.88	16.31	44.90
LMVR	13.87	39.98	16.33	44.71

Table 4: Lexical diversity scores on en-fr and en-it

Sequence length. As discussed in §3, we know that Italian and French feminine forms are, al-

---

[13] Metrics computed with software available at: `https://github.com/LSYS/LexicalRichness`. We set 1,000 as `window_size` for MATTR.

	en-fr (%)	**en-it** (%)
BPE	1.04	0.88
Char	1.37	0.38
DPE	2.11	0.77
Morfessor	1.62	0.45
LMVR	1.43	0.33

Table 5: Percentage increase of token sequence's length for feminine words over masculine ones.

	en	**Segm.**	**F**	**M**	**Freq. F/M**
a)	asked	BPE	chie–sta	chiesto	36/884
b)		DPE	chie–sta	chiesto	36/884
c)	friends	BPE	a–miche	amici	49/1094
d)		DPE	a–miche	amici	49/1094
e)	adopted	BPE	adop–tée	adop–té	30/103
f)		DPE	adop–t–é–e	adop–t–é	30/103
g)	sure	Morf.	si–cura	sicuro	258/818
h)	grown up	LMVR	cresci–uta	cresci–uto	229/272
i)	celebrated	LMVR	célébr–ées	célébr–és	3/7

Table 6: Examples of word segmentation. The segmentation boundary is identified by "–".

though to different extent, longer and less frequent than their masculine counterparts. In light of such conditions, we expected that the statistically-driven BPE segmentation would leave feminine forms unmerged at a higher rate, and thus add uncertainty to their generation. To verify if this is the actual case – explaining BPE's lower gender accuracy – we check whether the number of tokens (characters or subwords) of a segmented feminine word is higher than that of the corresponding masculine form. We exploit the coupled "wrong" and "correct" references available in MuST-SHE, and compute the average percentage of additional tokens found in the feminine segmented sentences[14] over the masculine ones. Results are reported in Table 5.

At a first look, we observe opposite trends: BPE segmentation leads to the highest increment of tokens for feminine words in Italian, but to the lowest one in French. Also, DPE exhibits the highest increment in French, whereas it actually performs slightly better than Char on feminine gender translation (see Table 3). Hence, even the increase in sequence length does not seem to be an issue on its own for gender translation. Nonetheless, these apparently contradictory results encourage our last exploration: *How* are gender forms actually split?

Gender isolation. By means of further manual analysis on 50 output sentences per each of the 6 systems, we inquire if longer token sequences for feminine words can be explained in light of the different characteristics and gender productive mechanisms of the two target languages (§3.1). Table 6 reports selected instances of coupled feminine/masculine segmented words, with their respective frequency in the MuST-C training set.

Starting with Italian, we find that BPE sequence length increment indeed ensues from greedy splitting that, as we can see from examples **(a)** and **(c)**, ignores meaningful affix boundaries for both same length and different-length gender pairs, respectively. Conversely, on the French set – with 95% of feminine words longer than their masculine counterparts – BPE's low increment is precisely due to its loss of semantic units. For instance, as shown in **(e)**, BPE does not preserve the verb root (*adopt*), nor isolates the additional token (-*e*) responsible for the feminine form, thus resulting into two words with the same sequence length (2 tokens). Instead DPE, which achieved the highest accuracy results for en-fr feminine translation (Table 3), treats the feminine additional character as a token *per se* (**f**).

Based on such patterns, our intuition is that the proper splitting of the morpheme-encoded gender information as a distinct token favours gender translation, as models learn to productively generalize on it. Considering the high increment of DPE tokens for Italian in spite of the limited number of longer feminine forms (15%), our analysis confirms that DPE is unlikely to isolate gender morphemes on the en-it language pair. As a matter of fact, it produces the same kind of coarse splitting as BPE (see **(b)** and **(d)**).

Finally, we attest that the two morphological techniques are not equally valid. Morfessor occasionally generates morphologically incorrect subwords for feminine forms by breaking the word stem (see example **(g)** where the correct stem is *sicur*). Such behavior also explains Morfessor's higher token increment with respect to LMVR. Instead, although LMVR (examples **(h)** and **(i)**) produces linguistically valid suffixes, it often condenses other grammatical categories (e.g. tense and number) with gender. As suggested above, if the pinpointed split of morpheme-encoded gender is a key factor for gender translation, LMVR's lower level of granularity explains its reduced gender accuracy. Working on character' sequences, instead, the isolation of the gender unit is always attained.

---

[14] As such references only vary for gender-marked words, we can isolate the difference relative to gender tokens.

	en-fr			en-it		
	M-C	M-SHE	Avg.	M-C	M-SHE	Avg.
BPE	**30.7**	25.9	28.3	21.4	21.8	21.6
Char	29.5	24.2	26.9	21.3	20.7	21.0
BPE&Char	30.4	**26.5**	**28.5**	**22.1**	**22.6**	**22.3**

Table 7: SacreBLEU scores on MuST-C tst-COMMON (M-C) and MuST-SHE (M-SHE) on en-fr and en-it.

	en-fr				
	ALL	1F	1M	2F	2M
BPE	65.18	37.17	**75.44**	61.20	80.80
Char	**68.85**	**48.21**	74.78	65.89	81.03
BPE&Char	68.04	40.61	75.11	**67.01**	**81.45**
	en-it				
BPE	67.47	33.17	**88.50**	60.26	81.82
Char	**71.69**	48.33	85.07	**64.65**	**84.33**
BPE&Char	70.05	**52.23**	84.19	59.60	81.37

Table 8: Gender accuracy (%) for MuST-SHE Overall (ALL), Category 1 and 2 on en-fr and en-it.

6 Beyond the Quality-Gender Trade-off

Informed by our experiments and analysis (§5), we conclude this study by proposing a model that combines BPE overall translation quality and Char's ability to translate gender. To this aim, we train a multi-decoder approach that exploits both segmentations to draw on their corresponding advantages.

In the context of ST, several multi-decoder architectures have been proposed, usually to jointly produce both transcripts and translations with a single model. Among those in which both decoders access the encoder output, here we consider the best performing architectures according to Sperber et al. (2020). As such, we consider: *i) Multitask direct*, a model with one encoder and two decoders, both exclusively attending the encoder output as proposed by Weiss et al. (2017), and *ii)* the *Triangle* model (Anastasopoulos and Chiang, 2018), in which the second decoder attends the output of both the encoder and the first decoder.

For the triangle model, we used a first BPE-based decoder and a second Char decoder. With this order, we aimed to enrich BPE high quality translation with a refinement for gender translation, performed by the Char-based decoder. However, the results were negative: the second decoder seems to excessively rely on the output of the first one, thus suffering from a severe *exposure bias* (Ranzato et al., 2016) at inference time. Hence, we do not report the results of these experiments.

Instead, the *Multitask direct* has one BPE-based and one Char-based decoder. The system requires a training time increase of only 10% and 20% compared to, respectively, Char and BPE models. At inference phase, instead, running time and size are the same of a BPE model. We report overall translation quality (Table 7) and gender accuracy (Table 8) of the BPE output (BPE&Char).[15] Starting with gender accuracy, the Multitask model's overall gender translation ability (ALL) is still lower, although very close, to that of the Char-based model. Nevertheless, feminine translation improvements are present on Category 2F for en-fr and, with a larger gain, on 1F for en-it. We believe that the presence of the Char-based decoder is beneficial to capture into the encoder output gender information, which is then also exploited by the BPE-based decoder. As the encoder outputs are richer, overall translation quality is also slightly improved (Table 7). This finding is in line with other work (Costa-jussà et al., 2020b), which proved a strict relation between gender accuracy and the amount of gender information retained in the intermediate representations (encoder outputs).

Overall, following these considerations, we posit that target segmentation can directly influence the gender information captured in the encoder output. In fact, since the Char and BPE decoders do not interact with each other in the Multitask model, the gender accuracy gains of the BPE decoder cannot be attributed to a better ability of a segmentation method in rendering the gender information present in the encoder output into the translation.

Our results pave the way for future research on the creation of richer encoder outputs, disclosing the importance of target segmentation in extracting gender-related knowledge. With this work, we have taken a step forward in ST for English-French and English-Italian, pointing at plenty of new ground to cover concerning *how to* split for different language typologies. As the motivations of this inquiry clearly concern MT as well, we invite novel studies to start from our discoveries and explore how they apply under such conditions, as well as their combination with other bias mitigating strategies.

7 Conclusion

As the old IT saying goes: *garbage in, garbage out*. This assumption underlies most of current attempts to address gender bias in language technologies. Instead, in this work we explored whether technical choices can exacerbate gender bias by focusing on

---

[15] The Char scores are not reported, as they are not enhanced compared to the base Char encoder-decoder model.

the influence of word segmentation on gender translation in ST. To this aim, we compared several word segmentation approaches on the target side of ST systems for English-French and English-Italian, in light of the linguistic gender features of the two target languages. Our results show that tokenization does affect gender translation, and that the higher BLEU scores of state-of-the-art BPE-based models come at cost of lower gender accuracy. Moreover, first analyses on the behaviour of segmentation techniques found that improved generation of gender forms could be linked to the proper isolation of the morpheme that encodes gender information, a feature which is attained by character-level split. Finally, we propose a multi-decoder approach to leverage the qualities of both BPE and character splitting, improving both gender accuracy and BLEU score, while keeping computational costs under control.

Acknowledgments

This work is part of the "End-to-end Spoken Language Translation in Rich Data Conditions" project,[16] which is financially supported by an Amazon AWS ML Grant. The authors also wish to thank Duygu Ataman for the insightful discussions on this work.

8 Ethic statement[17]

In compliance with ACL norms of ethics, we wish to elaborate on *i)* characteristics of the dataset used in our experiments, *ii)* the study of gender as a variable, and *iii)* the harms potentially arising from real-word deployment of direct ST technology.

As already stated, in our experiments we rely on the training data from the TED-based MuST-C corpus[18] and its derived evaluation benchmark, MuST-SHE. Although precise information about various sociodemographic groups represented in the data are not fully available, based on impressionistic overview and prior knowledge about the nature of TED talks it is expected that the speakers are almost exclusively adults (over 20), with different geographical backgrounds. Thus, such data are likely to allow for modeling a range of English varieties of both native and non-native speakers.

As regards gender, from the data statements (Bender and Friedman, 2018) of the used corpora, we know that MuST-C training data are manually annotated with speakers' gender information[19] based on the personal pronouns found in their publicly available personal TED profile. As reported in its release page,[20] the same annotation process applies to MuST-SHE as well, with the additional check that the indicated (English) linguistic gender forms are rendered in the gold standard translations. Hence, information about speakers' preferred linguistic expressions of gender are transparently validated and disclosed. Overall, MuST-C exhibits a gender imbalance: 70% vs. 30% of the speakers referred by means of *he/she* pronoun, respectively. Instead, allowing for a proper cross-gender comparison, they are equally distributed in MuST-SHE.

Accordingly, when working on the evaluation of speaker-related gender translation for MuST-SHE category (1), we proceed by solely focusing on the rendering of their linguistic gender expressions. As per (Larson, 2017) guidelines, no assumptions about speakers' self determined identity (GLAAD, 2007) – which cannot be directly mapped from pronoun usage (Cao and Daumé III, 2020; Ackerman, 2019) – has been made. Unfortunately, our experiments only account for the binary linguistic forms represented in the used data. To the best of our knowledge, ST natural language corpora going beyond binarism do not yet exist,[21] also due to the fact that gender-neutralization strategies are still object of debate and challenging to fully implement in languages with grammatical gender (Gabriel et al., 2018; Lessinger, 2020). Nonetheless, we support the rise of alternative neutral expressions for both languages (Shroy, 2016; Gheno, 2019) and point towards the development of non-binary inclusive technology.

Lastly, we endorse the point made by Gaido et al. (2020). Namely, direct ST systems leveraging speaker's vocal biometric features as a gender cue have the capability to bring real-world dangers, like the categorization of individuals by means of biological essentialist frameworks (Zimman, 2020). This is particularly harmful to transgender individuals, as it can lead to misgendering (Stryker, 2008) and diminish their personal identity. More generally, it can reduce gender to stereotypical ex-

---

[16] `https://ict.fbk.eu/units-hlt-mt-e2eslt/`

[17] Extra space after the 8th page allowed for ethical considerations – see `https://2021.aclweb.org/calls/papers/`

[18] `https://ict.fbk.eu/must-c/`

[19] `https://ict.fbk.eu/must-speakers/`

[20] `https://ict.fbk.eu/must-she/`

[21] In the whole MuST-C training set, only one speaker with *they* pronouns is represented.

pectations about how masculine or feminine voices should sound. Note that, we do not advocate for the deployment of ST technologies *as is*. Rather, we experimented with unmodified models for the sake of hypothesis testing without adding variability. However, our results suggest that, if certain word segmentation techniques better capture correlations from the received input, such capability could be exploited to redirect ST attention away from speakers' vocal characteristics by means of other information provided.

References

Lauren Ackerman. 2019. Syntactic and cognitive issues in investigating gendered coreference. *Glossa: a Journal of General linguistics*, 4(1).

Antonios Anastasopoulos and David Chiang. 2018. Tied Multitask Learning for Neural Speech Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91, New Orleans, Louisiana.

Duygu Ataman and Marcello Federico. 2018. An Evaluation of Two Vocabulary Reduction Methods for Neural Machine Translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–110, Boston, MA. Association for Machine Translation in the Americas.

Duygu Ataman, Orhan Firat, Mattia A. Di Gangi, Marcello Federico, and Alexandra Birch. 2019. On the Importance of Word Boundaries in Character-level Neural Machine Translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 187–193, Hong Kong. Association for Computational Linguistics.

Duygu Ataman, Matteo Negri, Marco Turchi, and Marcello Federico. 2017. Linguistically Motivated Vocabulary Reduction for Neural Machine Translation from Turkish to English. *The Prague Bulletin of Mathematical Linguistics*, 108:331–342.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On Using SpecAugment for End-to-End Speech Translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, Hong Kong, China.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2020. On the Linguistic Representational Power of Neural Machine Translation Models. *Computational Linguistics*, 46(1):1–52.

Emily M. Bender and Batya Friedman. 2018. Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Ruha Benjamin. 2019. *Race after Technology: Abolitionist Tools for the New Jim Code*. Polity Press, Cambridge, UK.

Luisa Bentivogli, Beatrice Savoldi, Matteo Negri, Mattia A. Di Gangi, Roldano Cattoni, and Marco Turchi. 2020. Gender in Danger? Evaluating Speech Translation Technology on the MuST-SHE Corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6923–6933, Online. Association for Computational Linguistics.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (Technology) is Power: A Critical Survey of "Bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Yang T. Cao and Hal Daumé III. 2020. Toward Gender-Inclusive Coreference Resolution. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4568–4595, Online. Association for Computational Linguistics.

Roldano Cattoni, Mattia A. Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2020. MuST-C: A Multilingual Corpus for end-to-end Speech Translation. Computer Speech & Language Journal. Doi: https://doi.org/10.1016/j.csl.2020.101155.

Marina Chini. 1995. *Genere grammaticale e acquisizione: Aspetti della morfologia nominale in italiano L2*. Franco Angeli.

Won Ik Cho, Ji Won Kim, Seok Min Kim, and Nam Soo Kim. 2019. On Measuring Gender Bias in Translation of Gender-neutral Pronouns. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 173–181, Florence, Italy. Association for Computational Linguistics.

John W. Chotlos. 1944. A statistical and comparative analysis of individual written language samples. *Psychological Monographs*, 56(2):75.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Greville G. Corbett. 1991. *Gender*. Cambridge University Press, Cambridge, UK.

Greville G. Corbett. 2013. *The Expression of Gender*. De Gruyter.

Marta R. Costa-jussà. 2019. An analysis of Gender Bias studies in Natural Language Processing. *Nature Machine Intelligence*, 1:495–496.

Marta R. Costa-jussà, Christine Basta, and Gerard I. Gállego. 2020a. Evaluating Gender Bias in Speech Translation. *arXiv preprint arXiv:2010.14465*.

Marta R. Costa-jussà, Carlos Escolano, Christine Basta, Javier Ferrando, Roser Batlle, and Ksenia Kharitonova. 2020b. Gender Bias in Multilingual Neural Machine Translation: The Architecture Matters. *arXiv preprint arXiv:2012.13176*.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Marta R. Costa-jussà and Adrià de Jorge. 2020. Fine-tuning Neural Machine Translation on Gender-Balanced Datasets. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*, pages 26–34, Barcelona, Spain (Online). Association for Computational Linguistics.

Michael A. Covington and Joe D. McFall. 2010. Cutting the Gordian knot: The moving-average type–token ratio (MATTR). *Journal of quantitative linguistics*, 17(2):94–100.

Kate Crawford. 2017. The Trouble with Bias. In *Conference on Neural Information Processing Systems (NIPS) – Keynote*, Long Beach, California.

Mathias Creutz and Krista Lagus. 2005. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. *International Symposium on Computer and Information Sciences*.

Caroline Criado-Perez. 2019. *Invisible Women: Exposing Data Bias in a World Designed for Men*. Penguin Random House, London, UK.

Hannah Devinney, Jenny Björklund, and Henrik Björklund. 2020. Semi-Supervised Topic Modeling for Gender Bias Discovery in English and Swedish. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*, pages 79–92, Online. Association for Computational Linguistics.

Mattia A. Di Gangi, Marco Gaido, Matteo Negri, and Marco Turchi. 2020. On Target Segmentation for Direct Speech Translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (AMTA 2020)*, pages 137–150, Online. Association for Machine Translation in the Americas.

Mattia A. Di Gangi, Matteo Negri, Roldano Cattoni, Dessi Roberto, and Marco Turchi. 2019. Enhancing transformer for end-to-end speech-to-text translation. In *Machine Translation Summit XVII*, pages 21–31, Dublin, Ireland. European Association for Machine Translation.

Catherine D'Ignazio and Lauren F Klein. 2020. *Data feminism*. MIT Press, London, UK.

Mohamed Elaraby and Ahmed Zahran. 2019. A Character Level Convolutional BiLSTM for Arabic Dialect Identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 274–278, Florence, Italy. Association for Computational Linguistics.

Carlos Escolano, Marta R. Costa-jussà, and José A. R. Fonollosa. 2019. From Bilingual to Multilingual Neural Machine Translation by Incremental Training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 236–242, Florence, Italy. Association for Computational Linguistics.

Ute Gabriel, Pascal M. Gygax, and Elisabeth A. Kuhn. 2018. Neutralising linguistic sexism: Promising but cumbersome? *Group Processes & Intergroup Relations*, 21(5):844–858.

Marco Gaido, Beatrice Savoldi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2020. Breeding Gender-aware Direct Speech Translation Systems. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3951–3964, Online. International Committee on Computational Linguistics.

Mahault Garnerin, Solange Rossato, and Laurent Besacier. 2019. Gender Representation in French Broadcast Corpora and Its Impact on ASR Performance. In *Proceedings of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*, AI4TV '19, page 3–9, New York, NY, USA. Association for Computing Machinery.

Vera Gheno. 2019. *Femminili Singolari: il femminismo è nelle parole*. Effequ.

GLAAD. 2007. Media Reference Guide - Transgender.

Hila Gonen and Kellie Webster. 2020. Automatically Identifying Gender Issues in Machine Translation using Perturbations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1991–1995, Online. Association for Computational Linguistics.

Pascal M. Gygax, Daniel Elmiger, Sandrine Zufferey, Alan Garnham, Sabine Sczesny, Lisa von Stockhausen, Friederike Braun, and Jane Oakhill. 2019. A Language Index of Grammatical Gender Dimensions to Study the Impact of Grammatical Gender on the Way We Perceive Women and Men. *Frontiers in Psychology*, 10:1604.

Christian Hardmeier, Marta R. Costa-jussà, Kellie Webster, Will Radford, and Su Lin Blodgett. 2021. How to Write a Bias Statement: Recommendations for Submissions to the Workshop on Gender Bias in NLP. *arXiv preprint arXiv:2104.03026*.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic Programming Encoding for Subword Segmentation in Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Marlis Hellinger and Heiko Motschenbacher. 2015. *Gender Across Languages. The Linguistic Representation of Women and Men*, volume IV. John Benjamins, Amsterdam, the Netherlands.

Charles F. Hockett. 1958. *A Course in Modern Linguistics*. Macmillan, New York, New York.

Dirk Hovy, Federico Bianchi, and Tommaso Fornaciari. 2020. “You Sound Just Like Your Father” Commercial Machine Translation Systems Include Stylistic Biases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1686–1690, Online. Association for Computational Linguistics.

Dirk Hovy and Shannon L. Spruit. 2016. The Social Impact of Natural Language Processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google’s Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Brian Larson. 2017. Gender as a variable in Natural-Language Processing: Ethical considerations. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 1–11, Valencia, Spain. Association for Computational Linguistics.

Enora Lessinger. 2020. The Challenges of Translating Gender in UN texts. In Luise von Flotow and Hala Kamal, editors, *The Routledge Handbook of Translation, Feminism and Gender*. Routledge, New York, NY, USA.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based Neural Machine Translation. *arXiv preprint arXiv:1511.04586*.

Nishtha Madaan, Sameep Mehta, Taneea Agrawaal, Vrinda Malhotra, Aditi Aggarwal, Yatin Gupta, and Mayank Saxena. 2018. Analyze, Detect and Remove Gender Stereotyping from Bollywood Movies. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 92–105, New York, NY, USA. PMLR.

Amit Moryossef, Roee Aharoni, and Yoav Goldberg. 2019. Filling Gender & Number Gaps in Neural Machine Translation with Black-box Context Injection. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 49–54, Florence, Italy. Association for Computational Linguistics.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, et al. 2018. XNMT: The eXtensible Neural Machine Translation Toolkit. In *Proceedings of AMTA 2018*, pages 185–192, Boston, MA.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Proceedings of Interspeech 2019*, pages 2613–2617, Graz, Austria.

Edoardo Maria Ponti, Helen O’horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2019. Modeling language variation and universals: A survey on typological linguistics for Natural Language Processing. *Computational Linguistics*, 45(3):559–601.

Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Tomasz Potapczyk and Pawel Przybysz. 2020. SRPOL’s System for the IWSLT 2020 End-to-End Speech Translation Task. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 89–94, Online. Association for Computational Linguistics.

Marcelo O. R. Prates, Pedro H. C. Avelar, and Luís C. Lamb. 2020. Assessing Gender Bias in Machine Translation: a Case Study with Google Translate. *Neural Computing and Applications*, 32(10):6363–6381.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of 4th International Conference on Learning Representations ICLR*, San Juan, Puerto Rico.

Argentina A. Rescigno, Johanna Monti, Andy Way, and Eva Vanmassenhove. 2020. A Case Study of Natural Gender Phenomena in Translation: A Comparison of Google Translate, Bing Microsoft Translator and DeepL for English to Italian, French and Spanish. In *Proceedings of the Workshop on the Impact of Machine Translation (iMpactT 2020)*, pages 62–90, Online. Association for Machine Translation in the Americas.

Nicholas Roberts, Davis Liang, Graham Neubig, and Zachary C. Lipton. 2020. Decoding and Diversity in Machine Translation. In *Proceedings of the Resistance AI Workshop at 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada.

Danielle Saunders and Bill Byrne. 2020. Reducing Gender Bias in Neural Machine Translation as a Domain Adaptation Problem. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7724–7736, Online. Association for Computational Linguistics.

Danielle Saunders, Rosie Sallis, and Bill Byrne. 2020. Neural Machine Translation Doesn't Translate Gender Coreference Right Unless You Make It. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*, pages 35–43, Online. Association for Computational Linguistics.

Beatrice Savoldi, Marco Gaido, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Gender Bias in Machine Translation. *Transactions of the Association for Computational Linguistics*.

Elmar Schafroth. 2003. *Gender in French: Structural properties, incongruences and asymmetries*. John Benjamins Publishing, Amsterdam, The Netherlands.

Rico Sennrich. 2017. How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Deven S. Shah, Hansen A. Schwartz, and Dirk Hovy. 2020. Predictive Biases in Natural Language Processing Models: A Conceptual Framework and Overview. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5248–5264, Online. Association for Computational Linguistics.

Alyx J. Shroy. 2016. Innovations in gender-neutral French: Language practices of nonbinary French speakers on Twitter. *Ms., University of California, Davis*.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden. Association for Computational Linguistics.

Matthias Sperber, Hendra Setiawan, Christian Gollan, Udhyakumar Nallasamy, and Matthias Paulik. 2020. Consistent Transcription and Translation of Speech. *Transactions of the Association for Computational Linguistics (TACL)*, 8:695–709.

Artūrs Stafanovičs, Mārcis Pinnis, and Toms Bergmanis. 2020. Mitigating Gender Bias in Machine Translation with Target Gender Annotations. In *Proceedings of the Fifth Conference on Machine Translation*, pages 629–638, Online. Association for Computational Linguistics.

Dagmar Stahlberg, Friederike Braun, Lisa Irmen, and Sabine Sczesny. 2007. Representation of the Sexes in Language. *Social communication*, pages 163–187.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating Gender Bias in Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684, Florence, Italy. Association for Computational Linguistics.

Susan Stryker. 2008. Transgender history, homonormativity, and disciplinarity. *Radical History Review*, 2008(100):145–157.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating Gender Bias in Natural Language Processing: Literature Review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, Nevada, United States.

Mildred C. Templin. 1957. *Certain Language Skills in Children: Their Development and Interrelationships*. University of Minnesota Press.

Nneka Umera-Okeke. 2012. Linguistic sexism: an overview of the English language in everyday discourse. *AFRREV LALIGENS: An International Journal of Language, Literature and Gender Studies*, 1(1):1–17.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting Gender Right in Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3003–3008, Brussels, Belgium. Association for Computational Linguistics.

Eva Vanmassenhove, Dimitar Shterionov, and Andy Way. 2019. Lost in Translation: Loss and Decay of Linguistic Richness in Machine Translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 222–232, Dublin, Ireland. European Association for Machine Translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS)*, pages 5998–6008, Long Beach, California.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-Sequence Models Can Directly Translate Foreign Speech. In *Proceedings of Interspeech 2017*, pages 2625–2629, Stockholm, Sweden.

Lal Zimman. 2020. Transgender language, transgender moment: Toward a trans linguistics. In Kira Hall and Rusty Barrett, editors, *The Oxford Handbook of Language and Sexuality*.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual Data Augmentation for Mitigating Gender Stereotypes in Languages with Rich Morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

A Models

Our direct ST models are built with the Fairseq library (Ott et al., 2019) and are based on the Transformer (Vaswani et al., 2017). They have 11 encoder and 4 decoder layers (Potapczyk and Przybysz, 2020). The encoder layers are preceded by 2 3x3 convolutional layers with 64 filters that reduce the input sequence length by a factor of 4 and their attention weights are added a logarithmic distance penalty (Di Gangi et al., 2019). The models are optimized on label smoothed cross-entropy (Szegedy et al., 2016) – the smoothing factor is 0.1 – with Adam using β_1=0.9, β_2=0.98 and the learning rate is linearly increased during the warm-up phase (4k iterations) up to the maximum value 5×10^{-3}, followed by decay with inverse square root policy. The dropout is set to 0.2 and each mini-batch consists of 8 sentences, while the update frequency is 8. The source audio is augmented with SpecAugment (Park et al., 2019; Bahar et al., 2019) that is applied with probability 0.5 by masking two bands on the frequency axis (with 13 as maximum mask length) and two on the time axis (with 20 as maximum mask length).

	en-fr	en-it
BPE	60M	60M
Char	52M	52M
DPE	60M	60M
Morfessor	79M	76M
LMVR	74M	72M
BPE&Char	77M	77M

Table 9: Number of parameters (in millions). For BPE&Char, the reported number is the total of the training parameters, but at inference time only one decoder is used, so the size is the same of BPE.

The systems are trained on MuST-C (Cattoni et al., 2020). We filtered from the training set all the samples whose audio length is higher than 20s. So to avoid rewarding models' potentially biased behaviour, as a validation set we rely on the MuST-C gender-balanced dev set (Gaido et al., 2020). The target text was tokenized with Moses.[22] We normalized audio per-speaker and extracted 40 features with 25ms windows sliding by 10ms with XNMT[23] (Neubig et al., 2018).

Trainings are stopped after 5 epochs without improvements on the validation loss and we average 5 checkpoints around the best on the validation set. They were performed on 8 K80 GPUs and lasted 2-3 days.

---

[22] `https://github.com/moses-smt/mosesdecoder`

[23] `https://github.com/neulab/xnmt`

On the Ethical Limits of Natural Language Processing on Legal Text

Dimitrios Tsarapatsanis
Law School
University of York
dimitrios.tsarapatsanis@york.ac.uk

Nikolaos Aletras
Computer Science Department
University of Sheffield
n.aletras@sheffield.ac.uk

Abstract

Natural language processing (NLP) methods for analyzing legal text offer legal scholars and practitioners a range of tools allowing to empirically analyze law on a large scale. However, researchers seem to struggle when it comes to identifying ethical limits to using NLP systems for acquiring genuine insights both about the law and the systems' predictive capacity. In this paper we set out a number of ways in which to think systematically about such issues. We place emphasis on three crucial normative parameters which have, to the best of our knowledge, been underestimated by current debates: (a) the importance of academic freedom, (b) the existence of a wide diversity of legal and ethical norms domestically but even more so internationally and (c) the threat of moralism in research related to computational law. For each of these three parameters we provide specific recommendations for the legal NLP community. Our discussion is structured around the study of a real-life scenario that has prompted recent debate in the legal NLP research community.

1 Introduction

Developing computational methods for analyzing legal text is an emerging area in natural language processing (NLP) with various applications such as legal topic classification (Nallapati and Manning, 2008), court opinion generation (Ye et al., 2018) and legal judgment prediction (Aletras et al., 2016; Luo et al., 2017; Zhong et al., 2018; Chalkidis et al., 2019). *Legal NLP* holds the promise of improving access to justice and offers to legal scholars the tools that allow for an empirical analysis of law on a large scale (Katz, 2012; Zhong et al., 2020).

The development and use of legal text processing technologies also raise a series of ethical questions, on which we focus in this paper. For example, following the publication at EMNLP 2019 of a paper on automatic prison term prediction (Chen et al., 2019) using a dataset constructed from published and publicly available records of past cases of the Supreme People's Court of China, a debate ensued about the ethical limits of legal NLP. More specifically, Leins et al. (2020) queried in a systematic way whether papers such as that of Chen et al. (2019) should be published. Leins et al. (2020) invoked a number of arguments including considerations to do with the construction of the dataset (Bender and Friedman, 2018), and so-called 'dual use' arguments (Radford et al., 2019), i.e. the possibility of using a system developed for some purpose for another, potentially harmful, purpose. Following a rich discussion, Leins et al. (2020) asked whether it is ethically permissible that legal NLP should be used at all to predict items such as prison terms.

We believe that the kind of ethical query put forth by Leins et al. (2020) is vital for the future of legal NLP and computational law in general. However, we also contend that it is essential that a more general discussion should be conducted about the pertinent normative principles and concepts at play. References to more general principles can often curb the temptation to make decisions on the basis of ad hoc moral intuitions which might not be and probably are not, as we explain later on, universally shared. The contributions of this paper are as follows:

- In this paper we make no claim to cover all the ethical ground in a comprehensive way. Instead, we focus on three crucial ethical parameters that, to the best of our knowledge, have not been extensively debated so far: (a) the role of *academic freedom* (§2); (b) the existence of a wide *diversity of legal and ethical norms* applicable to or endorsed by the global NLP scholarly community (§3); and (c) the

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3590–3599
August 1–6, 2021.

threat of moralism in legal NLP research (§4);

- We illustrate the practical difference that the first two factors would make, when taken into account, on the basis of the study of real-case scenarios of developing or using legal NLP technology;
- Moreover, for each of the three normative parameters (i.e. academic freedom, norms diversity and moralism), we provide specific recommendations for the legal NLP community.

2 Academic Freedom

2.1 What is academic freedom?

The idea of academic freedom has a long and important pedigree in the history of the Western university (Newman, 1976; Searle, 1972). Still, despite the importance of the idea, there is currently no commonly accepted definition accepted across domestic jurisdictions and scholars (Barendt, 2010). Be that as it may, for the purposes of this paper we provide the following working definition: *academic freedom is the freedom of scholars, whether employed by universities or not, to decide without undue external pressure or coercion the topics of their research, the standards of such research and the application of such standards to the scholars' peers*.

Academic freedom is both similar to and different from the more general right to freedom of speech (Barendt, 2010). It is similar in that it involves communicating freely chosen and conducted research to pertinent audiences. It is different in that: (a) it also applies to non-expressive research activities; (b) it is more circumscribed because scholars are bound by certain professional standards; and (c) it has both an individual-rights dimension and an institutional dimension, both being bound to the idea of the university as an intellectual space of free and uncoerced pursuit of truth. Academic freedom is recognized as a distinctive right by a number of national constitutions, such as Article 5(3) of the German Basic Law, as well as by some transnational legal instruments, such as Article 13 of the EU Charter of Fundamental Rights.

2.2 What is the value of academic freedom?

Two different tacks could be taken. First, one could provide a consequentialist argument. Consequentialist arguments identify right and wrong actions solely on the basis of their good or bad consequences (Driver, 2012). Accordingly, academic freedom could be justified by invoking the beneficial results of research undertaken, conducted and debated freely by researchers.

Second, one could opt for a deontological argument. Deontological ethical arguments identify permissible actions on the basis of (absolute) rules and irrespective of the consequences that actions may bring about (Alexander and Moore, 2007). Under a deontological construal, the guarantees of academic freedom would not be justified solely by the (expected) outcomes of the research actions they would allow but also: (a) because the pursuit of truth and knowledge for their own sake is a good in itself and irrespective of whether its (expected) consequences are good or bad (under some criterion of 'goodness' and 'badness'); and (b) because preserving academic freedom also preserves the integrity and inviolability of the very person of the researcher (Nagel, 1995).

We hasten to add for the sake of completeness that one could perhaps also opt for a virtue ethical approach to academic freedom (Hursthouse and Pettigrove, 2018). We refrain from doing so in this paper because, to the best of our knowledge, no comprehensive literature on a virtue ethics approach to academic freedom exists. Therefore, whatever the merits of such an approach, it falls outside the scope of this paper, which focuses on more entrenched within the pertinent research community ethical parameters.

2.3 Academic freedom and legal NLP

One first thing to note is that, despite its importance, academic freedom does not seem to be recognized as a distinctive ethical value in many codes related to the community of computer science researchers. For example, the ACM Code of Ethics refrains from specifically referring to that freedom in its first section, which is devoted to 'General Ethical Principles'.[1] While explaining this gap is beyond the scope of the present paper, we contend that academic freedom should be taken as seriously as the other ethical principles and values invoked therein, especially if one is to accept, as we think one should, that academic freedom's scope of application does not just include the 'academia' in the

---

[1] ACM Code of Ethics – `https://www.acm.org/code-of-ethics`

narrow institutional sense (i.e. universities or other kinds of tertiary education institutions) but virtually anyone who engages in research and scholarship with a view to discovering the truth about a subject matter.

If we begin with the widespread assumption that academic freedom is indeed a fundamental right, certain things seem to follow. First, in the assessment of the ethics of legal NLP, academic freedom will have to be taken into due account and then balanced against other values, such as the value of privacy of data subjects or the value or disvalue of certain further potential applications of systems (e.g. for legal judgment prediction) than those for which they were initially developed ('dual-use'). When values are balanced against each other, the all-things-considered permissibility of taking an action (in our case proceeding with a research project or publishing a paper in legal NLP) ultimately depends on taking all relevant contextual factors into account. We thus contend that there is no automatic general rule to apply to all cases. Still, we believe that, for example, in most imaginable scenarios where there is minimal interference with the rights of data subjects insofar as data such as court judgments are harvested from the public domain, pursuing an otherwise cognitively valuable research project should be permitted. More generally, taking academic freedom seriously renders the ethical permissibility of legal NLP projects more complex than sometimes acknowledged, since academic freedom will have to be taken seriously into account as an independent ethical factor, something which rarely (if ever) seems to happen as things currently stand, and despite the fact that almost all researchers pledge some kind of commitment to some version of academic freedom.

Second, and more importantly, ethical assessments of legal NLP research will also depend on the particular interpretation of the value of academic freedom (Barendt, 2010). For example, deontological interpretations of that value radically circumscribe the permissibility of invoking certain kinds of consequentialist reasons to block particular types of legal NLP research (Waldron, 2000). Invoking such reasons, we contend, is a direct violation of the integrity and equal freedom of researchers in the same way that, say, invoking people's tendency to be offended by certain kinds of artistic creations (for example, 'blasphemous' art) to block dissemination of art is a violation of the dignity of the artist. At the very least, under a robust understanding of the right to academic freedom (Dworkin, 1977), arguments to the effect that a certain piece of legal NLP research will have 'bad' consequences (whatever these are) will not be enough unless the risk of the consequences actually (as opposed to merely speculatively) obtaining is clear, present and significant. Again, we stress that there is no easily applicable general ethical rule here. Researchers should apply their ethical judgment in a contextual way by trying to account for all the relevant factors. We provide in the discussion that follows a concrete illustration of what a deontological understanding of academic freedom might entail in a particular scenario.

We also argue that even consequentialist interpretations of the value of academic freedom will place emphasis on the benefits of allowing researchers, at least *prima facie*, to freely conduct research, even when such research is ethically controversial. These benefits consist, among other things, in the possibility of maximizing the chances of getting to the truth about a certain subject-matter (in our case, law) through the proliferation of different research perspectives, thus promoting the public good.[2] Moreover, in this case again, the merely speculative possibility of certain adverse effects obtaining (e.g. systems designed by the researchers being used by others to pursue unethical goals) is not enough to render such research unethical in itself. There must be a real probability, not just a theoretical possibility, that these effects might in fact materialize.

2.4 Specific scenarios

These points can be readily illustrated by reference to the article by Leins et al. (2020) with which we began our discussion. We have said that, at the very least, academic freedom should be taken into account as an independent ethical value. We contend that this would modify significantly at least the structure of the argument provided by Leins et al. (2020). We stress, of course, that this in no way implies that academic freedom 'wins by default'. In fact, we have already highlighted that academic freedom is not the only pertinent ethical value and we have stressed that we do not believe that a generally applicable rule exists. All we intend to do is to argue that academic freedom as such should be

---

[2] ACM Code of Ethics – `https://www.acm.org/code-of-ethics#h-1.2-avoid-harm`

taken seriously into account and balanced against other pertinent ethical factors. However, we think that in the context of the particular case under consideration, academic freedom tips the balance in favor of the ethical permissibility of the research by Chen et al. (2019). Thus, the 'dual-use' and 'dataset construction' concerns voiced by Leins et al. (2020) will not be sufficient on their own to call the shots in ethics assessment in favor of rejecting either conducting or publishing the scrutinized research for the following reasons.

Dual-use Regarding the 'dual-use' concerns, we think that Leins et al. (2020) overestimate the dangers of an algorithm designed by academics being used to decide real cases with adverse consequences for real people. In particular, Leins et al. (2020) provide no reason to worry that any such use might happen anytime soon, nor evidence that there is, for example, a serious standing intention on the part of Chinese authorities to implement what would amount to a radical reform of the judicial system. Accordingly, the chance of an ethically unacceptable use of the designed algorithm seems, on the face of it, rather small (not to say almost completely theoretical). If this should be accepted, then both deontological and consequentialist interpretations of academic freedom tip the balance in favor of conducting and publishing the scrutinized research. Deontological interpretations place a very high premium on curtailments of the freedom and dignity of researchers. Arguably, this premium is not met by just the theoretical possibility of some bad consequence eventually resulting somewhere downstream from the research. But even consequentialist interpretations of academic freedom would concur in the case at point. This is because the expected probability of the occurrence of the bad consequences highlighted in the paper by Leins et al. (2020) (use of a potentially biased algorithm by the judiciary to help make inequitable decisions) seems in this case particularly low (or perhaps even nonexistent).

Of course, we fully acknowledge that we might be wrong in our assessment about the probability of a potentially evil 'dual-use'; we are in no way experts about Chinese legal and political affairs or of impending developments in the Chinese legal system. Still, the reasoning we have outlined also has a dimension to do with accruing and assessing evidence for ethical assessment. Thus, we hold that, in cases of doubt or where adequate evidence is neither available nor forthcoming, the value of academic freedom grounds at least a presumption in favor of proceeding with the research, the burden of proof in 'dual-use' scenarios been shifted to those who believe that there is a serious reason not to undertake a given piece of research. It is crucial for academic freedom that this burden should not be met by the researchers themselves. Arguably, in the case at hand the evidential burden has not been met by Leins et al. (2020).

Dataset Construction Similar considerations apply when it comes to assessing concerns to do with the construction of any particular dataset for legal NLP. The argument put forth by Leins et al. (2020) stresses that the dataset exposes people to harm because the defendants of past cases are identifiable. But arguably the probability of exposing people to harm through the construction of a dataset consisting of already decided cases by some domestic Supreme Court seems so low as to be practically non-existent. On the one hand, Leins et al. (2020) do not provide any evidence in favor of their argument. On the other hand, and to the best of our knowledge, the probability of these cases reopening together with that of some judge being specifically influenced by the dataset if such reopening occurs is practically non-existent. In all jurisdictions that we know about, cases by courts at the top of the judicial hierarchy are mostly, even if not uniquely, to do with the past, since by definition Supreme Courts issue the final judgment on some case. Thus, in order for considerations of potential harm to defendants to trump academic freedom, the risk of harm must be real, even under a generous understanding of the latter term, not merely imaginary or speculative. We stress, moreover, that this kind of understanding of harm coheres well with the ACM Code of Ethics that we have already cited. Thus, here again academic freedom, under any reasonable interpretation, should normally tip the balance in favor of proceeding with the research.

In a similar way, contending that some dataset 'unfairly advantages or disadvantages', as Leins et al. (2020) claim, appears to hugely overstate the real degree to which academic research might impact on the actual workings of judicial and political institutions. This is even more the case with respect to the specific dataset in question, since the latter was created by using documents already available in the public domain. In fact, it is the availability of the documents in the public domain itself, for which researchers bear no ethical responsibility, that creates certain risks to data subjects, if at all. Furthermore, such a contention bypasses the obvious fact that the dataset merely indicates the existence of unfair treatment that happened, if it did, at the level of what some Supreme Court did and not at the level of converting the activity of this Court into a dataset.

Last, but not least, here again the cases refer to facts that have already happened and no details are provided in Leins et al. (2020) as to whether there is, within the Chinese legal and social order, a concrete and standing threat for individuals to be unfairly treated on the basis of past cases (whether these led to convictions or not). Thus, we repeat that a merely theoretical or speculative possibility should not be considered sufficient, absent other factors, and especially concrete and real, not just imagined, reasons to trump academic freedom.

2.5 Recommendations

In decisions about the ethics of legal NLP research, deliberation should begin by commencing with a (rebuttable) presumption to the effect that academic freedom should not be curtailed lest there be compelling, to wit, clear and present or at least significantly probable (not purely theoretical or speculative) reasons to decide otherwise.

3 Diversity of Ethical and Legal Norms

3.1 Norms Diversity: The general problem

The community of legal NLP researchers is now global, ranging from the Global North to the Global South. However, a number of ethical standards on how to conduct legal NLP research in many cases seem to be either local or of local origin. Moreover, different ethical and legal standards may be found across different jurisdictions or cultures. Moral diversity is thus an issue that crops up in a variety of contexts to do with legal NLP. Below, we shall concentrate on the specific issue of privacy and data protection in the construction of datasets. Still, similar issues could also arise in other areas, such as the definition of what counts as 'unfairness' towards a specific group.

Privileging a particular ethical or legal standard over another (or a particular interpretation of a common standard) could fuel the suspicion that, instead of reflecting a perspective of detached and 'pure rationality', such standards and interpretations are in fact just entrenching local (and, for that matter, Western European or, more generally, Global Northern) prejudices and preconceptions, imposing them as mandatory norms on researchers who might not reflectively endorse them. There are two distinct but interrelated issues here. First, as a matter of fact, there appears to be a wide diversity of ethical standards that are accepted by different communities, with some of them being endorsed by some and rejected by others and vice versa (Prinz, 2007). Second, even when ethical or legal standards are shared across communities, there is often more or less widespread and reasonable disagreement about their 'best' or 'proper' interpretation (Gowans, 2015).

Now, the above facts do not immediately lead to what might seem like an unacceptable ethical relativism, i.e. the idea that no universal valid ethical standards could ever exist (Gowans, 2015). A discussion about the merits of ethical relativism is outside the scope of this paper, and we wish neither to endorse nor to criticize it. Our only point here is that, irrespective of the stand one takes on the issue of ethical relativism, the reality of diversity of ethical and legal opinion gives rise to an issue that cannot be avoided, to wit, the kind of position that researchers may take when confronted with research based on ethical rules and norms that significantly diverge from their own. Thus, ethical diversity points to the need to submit the content of the researcher's own standards, especially when they are used to evaluate the research of scholars who do not necessarily endorse them or reside in parts of the globe where they are considered neither legally nor perhaps even morally binding, to a much more searching examination. Accordingly, it should not be taken for granted that, say, historically and geographically contingent conceptions of a particular concept (e.g. data privacy in the EU, or the ban of judge analytics in France[3]) should always and without further argument take normative priority in ethical assessments.

Like in the previous section on academic freedom, we hasten to add that this does not amount to an easily applicable rule or 'formula' (which in our case could be either 'pay no heed to alien rules and norms and stick to what you think is right' or its exact opposite such as 'accept diversity no matter how unethical the alien rules or norms appear to be'). In fact, we do not think that there exists

---

[3] `https://www.artificiallawyer.com/2019/06/04/france-bans-judge-analytics-5-years-in-prison-for-rule-breakers/`

any easy, uncontroversial and unequivocal way by which to resolve such ethically complicated issues. As everywhere in life, a lot will depend on the particulars of each case and on the researcher's capacity to manifest a sensitivity to these particulars and of the specific ways in which they mesh in each scenario. All that we are suggesting is that issues to do with diversity of ethical and legal outlooks should be brought to the attention of researchers and, in the end, may only be resolved by the exercise of contextual and inherently controversial ethical judgment. At the very least, the actual presence of a diversity of different outlooks requires a willingness to engage seriously with 'alien' rules and norms, trying to understand their point and justification from the other's point of view, instead of easily and sometimes even complacently dismissing them out of hand as straightforwardly unethical.

3.2 Specific Scenario

To better apprehend how such a relatively abstract discussion may play out in a particular context and scenario, let us once again revisit the initial example with which we began our discussion, i.e. the paper by Leins et al. (2020).

Data Privacy: The ethical factors at play A large part of the argument put forth by Leins et al. (2020) consists in querying the process of construction of the dataset used by Chen et al. (2019) by invoking the rights of data subjects to privacy (similarly, Aletras et al. (2016), Chalkidis et al. (2019), Chalkidis et al. (2020) and Chalkidis et al. (2021) use publicly available data that contain private information in the context of the European Court of Human Rights). Thus, Leins et al. (2020) raise issues to do with the use of 'sensitive and confidential data', such as data to do with prison time served. They also refer to the fact that the people represented in the dataset do not seem to have been informed about data collection. The question then is whether these research practices are ethically problematic. While it seems self-evident, at least for a researcher accustomed to the practices of late twentieth century European states, to answer in the affirmative, an issue that Leins et al. (2020) do not address is that any answer to this question will presuppose a particular prior understanding of the content of the right of data subjects to privacy, an understanding, moreover, that may only be available to certain kinds of legal and ethical cultures and not to others.

The discussion may progress by reference to the simple observation that, when it comes to the legal and ethical status of datasets composed of data already available in the public domain (such as those in the papers by Chen et al. (2019) and Chalkidis et al. (2019)), there appears to be no global universal consensus as to the scope of the privacy rights of the data subjects involved.[4] A few jurisdictions deploy very demanding definitions of data privacy, coupled with special protection provided with respect to 'sensitive data': the EU and Australia are prime examples. Still, there also exist major divergences between state practices, some states being much more permissive in those respects than others.

The invocation of privacy as an ethical consideration thus raises a distinctive problem: how should privacy be understood in view of the fact that there is no universal consensus on its nature and its protection across states? Should particular definitions of data privacy constitute the appropriate normative benchmark? And why? As we have already said, some of these definitions might be significantly more demanding than others, reflecting different local conventions about the value of rights that data subjects have under the law.

We contend that the documented divergences in the attitudes towards data protection and the right to privacy, differences reflected in the law of different jurisdictions, in fact stem from a more fundamental phenomenon, i.e. the diversity of ethical, moral and legal outlooks both across time and across states, cultures and peoples. (Prinz, 2007). In fact, as we have already shown, it is arguable that the high standard of data protection that Leins et al. (2020) use, which also appears to be the standard used by the ACM Code of Ethics[5] is in the minority from a global point of view (Renteln, 1988). Moreover, the construction of datasets from databases containing publicly available court judgments appears to at least count in favor of the prima facie permissibility of that research practice.

The availability of court judgments in the public domain even in core Western countries that otherwise protect private data in other domains and with respect to international courts such as the European Court of Human Rights still belies an older conception of privacy: one in which the publicity of court trials as a rule of law guarantee trumps

---

[4] `https://www.cnil.fr/en/data-protection-around-the-world`

[5] `https://www.acm.org/code-of-ethics`

the individual's wish to hide oneself from others in public space (Langford, 2009). Under this older conception of privacy, not only is it possible to retrieve the names of litigants in past cases, but also that possibility is protective of the rule of law in the same way that, say, secret trials of terrorist or other suspects are still widely thought to undermine it (Resnik, 2011). The above comments thus suffice to show that the high standard of privacy protection that is sometimes used to scrutinize research practices in legal NLP, exemplified by Leins et al. (2020), is not only geographically but also historically contingent.

Under these conditions, the following question becomes ethically pressing: Why should any local and geographically restricted conception of stricter data privacy and protection be preferred to, say, a more relaxed and minimal one? Granted, researchers conducting legal NLP research in locations where that particular conception is prevalent and legally obligatory, such as the European Union or Australia, should of course abide by it; but why should other researchers, such as those residing, say, in Singapore, the United States or Africa, be held to that same particularly demanding standard? Moreover, is it even fair and ethical, given what was already stated above with respect to academic freedom, to hold researchers (e.g. non-European and non-Australian) to a peculiarly local, and by no means universally acknowledged, standard?

Data Privacy: Application to the case at hand On the basis of the above considerations, we believe that this kind of contextually informed thinking could argue in favor of the ethical acceptability of the research practices used by Chen et al. (2019) in the construction of their dataset. There are two important variables at play here. On the one hand, the data is already in the public domain. Under these circumstances, the interference of the researchers with the data subjects' rights appears minimal at worst: in fact, even under a demanding definition of privacy, the main interference with privacy rights is due to the availability of the publicly available database itself and not to the researcher's activity of constructing a dataset for research purposes out of it. But, second, we could even go further and ask why what appears like a specifically European definition of privacy and data protection, and moreover one which, at least as things stand nowadays seems to be more of an outlier on a global level, should be adopted as a normative yardstick by researchers working in a completely different ethical and regulatory environment.

As we have already said, there is a long and venerable Western and European tradition which lays stress on the importance of the publicity of court judgments. For a long time, it was considered not unethical but utterly normal to make court trials and their outcomes public, so as to control through public scrutiny the exercise of state coercion on individuals. While this consideration might now seem dated in some parts of the world, especially those that have entrenched traditions of judicial independence and a high degree of trust to the judicial system, we should be sensitive to its importance in other parts of the world. It thus becomes difficult to resist the conclusion that the absence of respect of (putatively European) data protection norms by Chen et al. (2019)) is not a fatal ethical objection to either conducting or publishing their research.

3.3 Recommendations

Forging genuine universal ethical standards requires a global conversation between researchers engaged from a plurality of standpoints and traditions. When such standards do not exist or exist only to a minimal degree, ethical assessment for global conferences, journals and reviews should be appropriately flexible and respectful of differences and reasonable disagreements. No automatic assumptions should be made that a 'one-size-fits-all' model is sufficient to make informed decisions about the ethical status of research practices.

4 The Threat of Moralism in Legal NLP

We believe that the previous discussion points to a more general issue, that we treat under the heading of the 'threat of moralism' in legal NLP. While this issue is more theoretical than the previous ones (which had practical ramifications), we think it should be treated on its own, insofar as it allows us to propose a more general and comprehensive conceptual framework that allows us to make wider sense of the ethical issues at play. This section of our paper thus has the function of both subsuming the previous discussion and further advancing it, albeit without this time focusing on the discussion of a specific scenario.

4.1 How to characterize moralism?

Moralism can be intuitively understood as 'the vice of overdoing morality' (Coady, 2015). In the context of qualitative research, Hammersley and Traianou (2011) has contended that moralism can take two different forms. First, it might involve the belief that substantive ethical values, other than the disinterested pursuit of knowledge for its own sake, should be integral goals of research. Second, it might involve the requirement that researchers adhere to 'high' or even the 'highest possible' ethical standards (Hammersley and Traianou, 2011).

Here, and with no ambition of a comprehensive discussion, we roughly define moralism as the idea that substantive moral values and constraints more demanding than mere adherence to valid legal norms or to relatively uncontroversial and in any event minimal ethical norms, such as the requirement not to harm others, must be taken into account when assessing research outputs, including legal NLP research. Our argument is that moralism of this sort threatens academic freedom and the equal dignity of researchers as bearers of such freedom.

4.2 Why moralism might be a problem in legal NLP?

To think clearly about why moralism in the sense of the pursuit of substantive moral values might be a problem in legal NLP, we might make an analogy with moralism in the setting of ordinary social and political life. At least liberal democracies take seriously John Stuart Mill's idea (Mill, 2015) that people should be free to pursue various ends that they themselves set, so long as they keep within certain reasonable limits circumscribed by the so-called 'harm' principle, i.e. the requirement not to harm others equally engaged in the pursuit of their proper ends.

A political community that attempts to impose substantive ends on individuals, say on the assumption that certain forms of life are 'higher' or 'more important' than others is guilty of disrespecting the autonomy of individuals. We contend that, in an analogous way, a research community that attempts to instill in its members specific ethical ends other than the disinterested pursuit of truth and knowledge, especially by ethically assessing the very content of research endeavors, is risking falling into a kind of moralism which fails to take the freedom of researchers seriously. In particular, the mere fact that certain members of the research community subjectively find a piece of research to be unacceptable on other than professionally defined and accepted methodological grounds or basic and flagrant disrespect of moral norms such as the no-harm principle, should not be deemed automatically sufficient for ethical condemnation of the research.

4.3 Recommendations

The primary moral duty of legal NLP researchers, like all researchers, is to the disinterested pursuit of truth as they understand it, and not to substantive ends which are extrinsic to that pursuit.

5 Conclusions

In this paper we have contributed to the ongoing discussion on the ethics of legal NLP (Leins et al., 2020). We laid emphasis on three normative factors whose importance, to the best of our knowledge, has not been sufficiently acknowledged, i.e. academic freedom, diversity of ethical and legal norms of the global NLP community and the wider and more abstract issue of moralism in research ethics. Moreover, we illustrated how the first two factors might make a practical difference to ethical decision-making by a detailed discussion of a specific scenario. We also stressed that these factors do not amount to any automatically applicable general rule but require the exercise of contextual ethical thinking on the part of researchers. Final decisions should be made by taking all ethically relevant factors into account.

We believe that the ethical factors we identified can help the legal NLP community become more reflective and tolerant of a wider variety of approaches whilst at the same time remaining fully committed to the academic ideal of disinterested pursuit of truth for its own sake.

References

Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro, and Vasileios Lampos. 2016. Predicting judicial decisions of the European Court of Human Rights: a Natural Language Processing perspective. *PeerJ Computer Science*, 2:e93.

Larry Alexander and Michael Moore. 2007. *Deontological ethics*. Stanford Encyclopedia of Philosophy.

Eric Barendt. 2010. *Academic freedom and the law: A comparative study*. Bloomsbury Publishing.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science.

Transactions of the Association for Computational Linguistics, 6:587–604.

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Dimitrios Tsarapatsanis, Nikolaos Aletras, Ion Androutsopoulos, and Prodromos Malakasiotis. 2021. Paragraph-level rationale extraction through regularization: A case study on European court of human rights cases. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 226–241, Online. Association for Computational Linguistics.

Huajie Chen, Deng Cai, Wei Dai, Zehui Dai, and Yadong Ding. 2019. Charge-based prison term prediction with deep gating network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6362–6367, Hong Kong, China. Association for Computational Linguistics.

Cecil Anthony John Coady. 2015. Preface. *Journal of Applied Philosophy*, 22(2):101–104.

Julia Driver. 2012. *Consequentialism*. Routledge.

Ronald Dworkin. 1977. *Taking rights seriously*. Harvard University Press.

Chris Gowans. 2015. *Moral relativism*. Stanford Encyclopedia of Philosophy.

Martyn Hammersley and Anna Traianou. 2011. Moralism and research ethics: a machiavellian perspective. *International Journal of Social Research Methodology*, 14(5):379–390.

Rosalind Hursthouse and Glen Pettigrove. 2018. *Virtue Ethics*. Stanford Encyclopedia of Philosophy.

Daniel Martin Katz. 2012. Quantitative legal prediction-or-how I learned to stop worrying and start preparing for the data-driven future of the legal services industry. *Emory LJ*, 62:909.

Ian Langford. 2009. Fair trial: The history of an idea. *Journal of human rights*, 8(1):37–52.

Kobi Leins, Jey Han Lau, and Timothy Baldwin. 2020. Give me convenience and give her death: Who should decide what uses of NLP are appropriate, and on what basis? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2908–2913, Online. Association for Computational Linguistics.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (Conference on Empirical Methods in Natural Language Processing (EMNLP))*, pages 2727–2736.

John Stuart Mill. 2015. *On Liberty, Utilitarianism, and other essays*. Oxford University Press, USA.

Thomas Nagel. 1995. Personal rights and public space. *Philosophy & Public Affairs*, pages 83–107.

Ramesh Nallapati and Christopher D. Manning. 2008. Legal docket classification: Where machine learning stumbles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (Conference on Empirical Methods in Natural Language Processing (EMNLP))*, pages 438–446.

John Henry Newman. 1976. *The idea of a university: Defined and illustrated (IT Ker, Ed.)*. Oxford, UK: Clarendon Press.(Original work published 1852).

Jesse Prinz. 2007. *The emotional construction of morals*. Oxford University Press.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Alison Dundes Renteln. 1988. Relativism and the search for human rights. *American Anthropologist*, 90(1):56–72.

Judith Resnik. 2011. Bring Back Bentham: Open courts, terror trials, and public sphere(s). *Law & Ethics of Human Rights*, 5(1):4–69.

John R Searle. 1972. Two concepts of academic freedom. *The Concept of Academic Freedom*, pages 86–97.

Jeremy Waldron. 2000. Pildes on Dworkin's Theory of Rights. *The Journal of Legal Studies*, 29(1):301–307.

Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. 2018. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (Conference on Empirical Methods in Natural Language Processing (EMNLP))*, pages 1854–1864.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. How does NLP benefit legal system: A summary of legal artificial intelligence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5218–5230, Online. Association for Computational Linguistics.

Haoxi Zhong, Guo Zhipeng, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2018. Legal judgment prediction via topological learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3540–3549.

An Exploratory Analysis of the Relation Between Offensive Language and Mental Health

Ana-Maria Bucur[1], Marcos Zampieri[2], and Liviu P. Dinu[1]
[1]University of Bucharest, Romania
[2]Rochester Institute of Technology, USA
ana-maria.bucur@drd.unibuc.ro, marcos.zampieri@rit.edu
ldinu@fmi.unibuc.ro

Abstract

In this paper, we analyze the interplay between the use of offensive language and mental health. We acquired publicly available datasets created for offensive language identification and depression detection and we train computational models to compare the use of offensive language in social media posts written by groups of individuals with and without self-reported depression diagnosis. We also look at samples written by groups of individuals whose posts show signs of depression according to recent related studies. Our analysis indicates that offensive language is more frequently used in the samples written by individuals with self-reported depression as well as individuals showing signs of depression. The results discussed here open new avenues in research in politeness/offensiveness and mental health.

1 Introduction

The use of offensive language is pervasive in social media and it has been studied from different perspectives. A popular line of research is the study of computational models to identify offensive content online relying on traditional machine learning classifiers (e.g. naive bayes and SVMs) (Xu et al., 2012; Dadvar et al., 2013), neural networks (e.g. LSTMs, GRUs) with word embeddings (Aroyehun and Gelbukh, 2018; Majumder et al., 2018), and more recently, transformer models like ELMO (Peters et al., 2018) and BERT (Devlin et al., 2019) which have shown to obtain competitive scores topping the leaderboards in recent shared tasks on offensive language and hate speech detection (Liu et al., 2019).

Offensive language is related to the notion of impoliteness (Culpeper, 2011) and it can take various forms from general and often harmless profanity to abusive language intended to cause harm, such as cyberbullying and hate speech (Waseem et al., 2017). Computational models have been applied not only to identify the various types of offensive content (Basile et al., 2019) but also to, for example, study the relation between profanity and hate speech (Malmasi and Zampieri, 2018) and the different functions and intentions of vulgarity in social media (Holgate et al., 2018).

Most of the datasets used in the aforementioned studies contain data sampled from the general population and therefore very little light has been shed on the use of offensive language in online communication by specific groups such as individuals with mental health conditions. A notable exception is the recent study by Birnbaum et al. (2020) which shows that users with mood disorders (bipolar disorder, major depressive disorder) and schizophrenia spectrum disorders use more swear words in their Facebook messages than healthy users.

To address this shortcoming, in this paper, we build on recent work on offensive language identification and apply it to mental health datasets. More specifically, we look at the role of offensive language in the communication of users with depression using two publicly available datasets containing posts by individuals with self-reported depression diagnosis.

To the best of our knowledge, this study is the first to apply state-of-the-art offensive language identification models to mental health datasets. We aim to answer two research questions:

RQ1: Are posts from individuals suffering from depression more likely to contain offensive language in existing datasets?

RQ2: Are there differences in the nature of offensive language used by individuals with depression compared to control groups?

WARNING: This paper contains offensive words.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3600–3606
August 1–6, 2021.

2 Related Work

Offensive language identification is a popular topic in NLP. Researchers have been working to improve the performance of systems trained to identify conversations that are likely to go awry (Zhang et al., 2018) and to detect the various types of offensive posts in social media (Basile et al., 2019; Kumar et al., 2020). More recently, with the goal of improving explainability, offensive language identification at the token-level has received more attention (Mathew et al., 2021; Ranasinghe and Zampieri, 2021). A number of computational models have been applied to this task ranging from traditional machine learning classifiers, most notably SVMs (MacAvaney et al., 2019), to various deep learning models (Liu et al., 2019). While the clear majority of studies on this topic deal with English, some studies have addressed offensive language in other languages like Greek (Pitenis et al., 2020) and Turkish (Çöltekin, 2020) while a few others have applied cross-lingual models to take advantage of existing English datasets when making predictions in languages with fewer resources (Ranasinghe and Zampieri, 2020).

Several studies have applied machine learning and NLP methods to address research questions related to mental health in social media such as identifying users with a particular mental health condition and predicting the risk of self-harm or suicide ideation (De Choudhury et al., 2013; Preoţiuc-Pietro et al., 2015; Malmasi et al., 2016; De Choudhury et al., 2016; Chancellor and De Choudhury, 2020). The CLPsych workshop co-located with international NLP conferences has hosted multiple competitions on these topics providing participants with important benchmark datasets and attracting a large number of teams (Coppersmith et al., 2015; Milne et al., 2016; Zirikly et al., 2019).

There have been multiple studies on the impact of offensive and hateful speech on the individual's psychological mental health and well-being (Bannink et al., 2014; Saha et al., 2019). The use of offensive language by individuals with mental health conditions, however, has not been substantially studies with the exception of Birnbaum et al. (2020) that analyzed the use of offensive language in Facebook messages from individuals with mood disorders. Our work fills this important gap by providing further empirical evidence of the use of offensive language by individuals with diagnosed depression or showing signs of depression.

3 Data

In our experiments, we use three publicly available English datasets with data collected from social media: one with offensive language annotation, and two datasets with posts from users with self-reported depression diagnosis.

Offensive Language We use the Offensive Language Identification Dataset (OLID) (Zampieri et al., 2019a) to train offensive language identification models. OLID contains a total of 14,100 manually annotated posts from Twitter and it was released as the official dataset of SemEval-2019 Task 6 (OffensEval) (Zampieri et al., 2019b). We chose OLID due to its general hierarchical annotation taxonomy with the following levels:

Level A: Offensive language identification: offensive (OFF) vs. non-offensive (NOT)

Level B: Categorization of offensive language: targeted insult or threats (TIN) vs. untargeted profanity (UNT).

Level C: Offensive language target identification: individual (IND) vs. group (GRP) vs. other (OTH).

This hierarchical taxonomy provides us with a flexibility as it represents multiple types of offensive content in a single annotation scheme (e.g. posts targeted at an individual are often *cyberbullying* and posts targeted at a group are often *hate speech*) making it a great fit for this kind of analysis. In our experiments, we consider level A (offensive vs. non-offensive) and level B (target vs. untargeted).

Mental Health We run all our experiments on the Reddit Self-reported Depression Diagnosis (RSDD) dataset (Yates et al., 2017) and on the Early Risk Prediction on the Internet (eRisk) 2018 dataset (Losada and Crestani, 2016), two publicly available datasets containing posts from Reddit. The RSDD dataset consists of users annotated as having depression by their mention of diagnosis and control users, which are users who do not suffer from depression (there is not any mention of diagnosis in their posts). To prevent users labeled with depression to be easily identified by specific keywords, the authors removed posts containing depression terms (e.g. *depression, depressive*) or belonging to mental health related subreddits. The authors made the training, validation, and test splits available and in our experiments we use the training split, which contains over 5 million posts from users with depression and over 30 million posts from users in the control group.

The eRisk 2018 dataset contains users labeled with depression by their mention of diagnosis and control users. In this paper, we use both train and test splits, consisting of a total of approximately 90,000 submissions from users annotated as having depression and 985,000 posts and comments from the users in the control group. As opposed to the RSDD dataset, the authors removed only the posts containing the exact mention of diagnosis.

4 Methods

Offensive Language Detection and Categorization We address **RQ1** and **RQ2** by studying the language of users from the two groups, self-reported depression diagnosis and control, in social media. We start by computing an offensive score, which measures the extent to which a post is offensive, and whether it is a targeted insult or an untargeted post (most often profanity). These two tasks correspond to OLID levels A and B respectively Zampieri et al. (2019a).

For the task of offensive language detection, we fine-tune a BERT model on the OLID dataset on level A. We train the model for 2 epochs, with a small learning rate of 0.00001 and Adam optimizer (Kingma and Ba, 2015). We use an 80:20 split of the training data to choose the best performing model in terms of F1 score. The model obtains 0.85 Precision, 0.74 Recall and 0.77 F1 score on the test data from the OLID dataset. These numbers are consistent with the baselines reported in (Zampieri et al., 2019a). The offensive score is computed as a probability taken from the softmax output of the BERT model.

For the task of offensive language categorization (targeted insult or untargeted profanity) we also choose a transformer-based approach, using another BERT model trained on OLID level B. We fine-tune BERT for 7 epochs with the same aforementioned train-validation split, with a learning rate of 0.00002 with Adam optimizer and a linear warm-up schedule with a 0.05 warm-up ratio, as proposed by Rosenthal et al. (2020). To account for the class imbalance, we use cross-entropy loss with balanced class weights. The effectiveness of the model is also evaluated on the OLID test data, using the same metrics and achieving 0.78 Precision, 0.84 Recall and 0.80 F1 score.

Signs of Depression Detection Furthermore, we are interested in distinguishing the posts that show signs of depression from all the posts of individuals from the depression group. This way, we filter out the noise added by the texts which do not contain any cues of depression. We are using the Semantic Polarity Score heuristic (H_s heuristic) proposed by Ríssola et al. (2020) to detect posts showing signs of depression written by individuals with a self-reported depression diagnosis.

H_s uses a mix of sentiment polarity, depression score, and emotion detection. The authors use TextBlob[1] to obtain the polarity score of each post, ranging between -1 and 1. The terms from EmoLex (Mohammad and Turney, 2013) are used in order to detect the emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) contained in the texts. The depression score of each post is computed using the NRC Affect Intensity Lexicon (Mohammad, 2017), ranging from 0 to 1. In order to distinguish the posts showing signs of depression from other posts of users with self-reported depression diagnosis, we follow the criteria from Ríssola et al. (2020). Posts are labeled as showing signs of depression if the texts have a negative polarity, if sadness or disgust emotions are present, and if they have a depression score higher than 0.1.

5 Results and Discussion

Using the H_s heuristic, we demonstrate that there is a statistically significant difference (Welch t-test, p-value <0.001) in terms of offensive language use between individuals with self-reported depression diagnosis that manifest signs of depression in their posts and users who do not show any signs of depression. Posts containing signs of depression have a higher offensive score than posts from users diagnosed with depression without any signs, in both eRisk 2018 and RSDD datasets, as shown in Figure 1.

For labeling the offensive posts, we use the same 0.50 threshold as used during training. We show in Table 1 that more posts from users diagnosed with depression are labeled as offensive than from control. Using the H_s heuristic, we filter the posts containing signs of depression and find that there is a higher percentage of posts with signs of depression labeled as offensive. These findings are consistent for both eRisk 2018 and RSDD datasets. The higher degree to which depressed individuals use offensive language in comparison to individuals in the control group can be explained via the

---

[1] https://textblob.readthedocs.io/en/dev/index.html

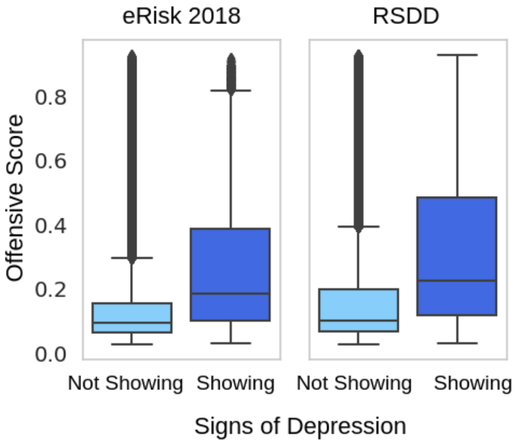

Figure 1: Distribution of the offensive language score for posts written by users with self-reported depression diagnosis and showing or not showing signs of depression measured with the H_s heuristic.

	Self-reported		Signs of depression	
Dataset	**Depression**	**Control**	**Showing**	**Not showing**
eRisk 2018	8.24%	5.91%	18.50%	7.40%
RSDD	11.31%	8.91%	24.33%	10.10%

Table 1: Percentage of posts labeled as offensive from total posts of self-reported individuals and of individuals showing/not-showing signs of depression measured with the H_s heuristic.

emotion regulation framework (Gross, 1999). The use of offensive language could be an emotion regulation strategy through which depressed individuals relieve some of their distress. Similarly, pain and distress studies indicate that the use of offensive language when experiencing pain significantly diminishes the level of pain experienced (Stephens and Robertson, 2020), suggesting that the use of offensive language can relieve distress.

Although there are more posts with signs of depression labeled as offensive, the majority of them are untargeted (containing swears, profanity) and only 10.71% and 10.72%, respectively, are targeted insults (Table 2).

	Self-reported		Signs of depression	
Dataset	**Depression**	**Control**	**Showing**	**Not showing**
eRisk 2018	24.12%	21.72%	11.48%	26.68%
RSDD	16.63%	23.94%	8.29%	18.48%

Table 2: Percentage of posts labeled as targeted insult from the offensive posts of self-reported individuals and of individuals showing/not showing signs of depression measured with the H_s heuristic.

The fact that depressed individuals tend to use more self-deprecating content and less deprecation of others, as evidenced in our analysis, is a result that is in line with the broad spectrum of cognitive studies, which indicates that negative evaluation of the self is a main interpretation bias in depressed individuals (Everaert et al., 2017). Depressed individuals tend to view themselves as less valuable than others. By self-deprecating language, we use the definition from Speer (2019). This broader definition includes, but is not limited to, insults towards self, if they have a negative intention. Finally, studies show that there is also a self-focused attention tendency in depressed individuals (Brockmeyer et al., 2015), where just like in other conditions (e.g. anxiety), individuals tend to be unable to detach from their own perspective focusing primarily on their side of the story, their pain, etc.

In order to further understand the differences in the use of offensive language, we analyze the words from posts written by individuals with depression. We compute the keyness score (Kilgarriff, 2009; Gabrielatos, 2018) of content words (removing stop words) from posts labeled as offensive written by users with self-reported diagnosis. The keyness is computed in order to show which words occur more often in the texts from depressed individuals showing signs of depression (target corpus) in comparison to the texts from users diagnosed with depression that do not show signs of depression (reference corpus). We calculate the frequencies of words from the two corpora and then the log-likelihood Ratio (G^2) (Dunning, 1993) for each word. In Figure 2 we present the top 20 words, ordered by G^2 from each corpus, in the two datasets.

We show that, while users without signs of depression refer more to sexual and profane terms, posts by users showing signs of depression include more negative words such as *bad, hate, sick, death*. This result corroborates the findings described in the literature on cognitive errors or biases in depression (Beck and Haigh, 2014). It is well known that depressed individuals tend to view life events more negatively than their non-depressed peers (Everaert et al., 2017). Furthermore, depressed individuals are more likely to recall negative life events than positive events and also more likely to pay closer attention to negative information (Beck and Haigh, 2014). Signs of this biased view of life are expected to be noticeable in language and there are studies that indicate that depressed individuals tend to have a more negative discourse than their non-depressed depressed peers (Rude et al., 2004). Keywords with a negative polarity, such as *bad, die* or *pain*, seem to be pervasive in the speech of depressed individuals as confirmed in our study. Finally, the reduced sexual drive is a well-known indication of depression (Manohar et al., 2017), therefore, it is to be expected that depressed individuals tend to use fewer words with sexual connotation as confirmed in our study.

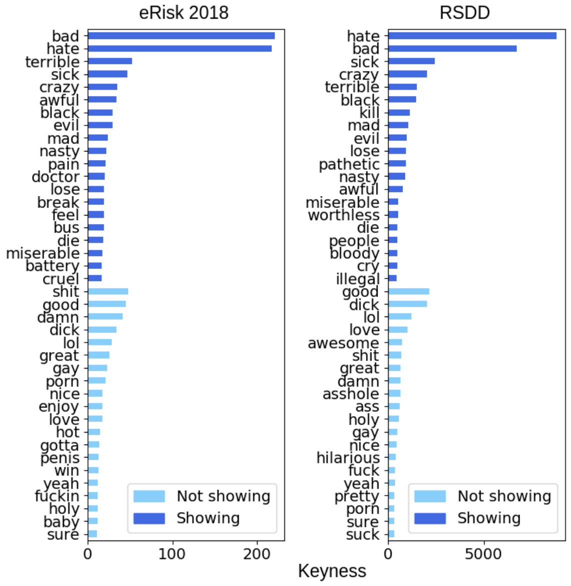

Figure 2: Keyness for words from posts showing/not showing signs of depression.

6 Conclusion and Future Work

This paper is the first to apply offensive language identification techniques to posts by individuals with a mental health condition with the purpose of interpreting the use of profanity and offensive language by this group. We showed how the offensive language use differs substantially between individuals with depression (in samples with self-reported diagnosis or showing signs of depression) answering our **RQ1**. Our findings indicate that users with self-reported depression diagnosis are more likely to use offensive language in their posts compared to the control group. From the posts of individuals with depression, the ones showing signs of depression contain more offensive language than the ones not showing any signs.

In terms of the nature of offensive content, our results indicate that posts from individuals with signs of depression are less likely to contain targeted offensive language. Furthermore, while analyzing the texts of users with depression, we observed a larger frequency of words with negative polarity (e.g. *bad, hate, sick, suffer*) in the posts of users showing signs of depression, where the discourse of users not showing any signs contains more sexual-related content, addressing our **RQ2**. These findings are consistent with the existing literature from psychology (Stephens and Robertson, 2020; Everaert et al., 2017; Beck and Haigh, 2014).

While it is clear that depressed users are more likely to write posts with negative polarity, the interplay between offensive language and polarity in the mental health datasets used in this paper has not yet been explored. A polarity score has been used in the heuristic by Ríssola et al. (2020) suggesting that using NLP models to investigate the interplay between polarity and depression is a promising future work direction. Other future work directions include the analysis of the targets of offensive posts using the OLID Level C annotation and a more detailed analysis on the function of profanity and vulgarity in these datasets (Holgate et al., 2018). Finally, we would like to carry out a similar analysis for other languages taking advantage of existing datasets and available cross-lingual embedding models.

Acknowledgments

We would like to thank Ioana Podină for providing us with helpful feedback and suggestions for the results analysis. We further thank the anonymous ACL reviewers for the insightful feedback provided. Finally, we thank the dataset creators for making the data used in this study available.

Liviu P. Dinu was partially supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI – UEFISCDI, project number 108, within PNCDI III.

Ethics Statement

This paper uses publicly available datasets on offensive language and mental health to train computational models with the purpose of carrying out both quantitative and qualitative data analysis. We are primarily interested in quantifying and analyzing the use of offensive language in the texts included in the two mental health datasets and we do not attempt to predict mental health status or condition from these datasets. Potential biases in our model predictions and in our analysis may arise from the annotation and sampling techniques of these two datasets and are not intentional. Finally, we did not use any form of demographic information in our models or in our analysis.

References

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of TRAC*.

Rienke Bannink, Suzanne Broeren, Petra M van de Looij-Jansen, Frouwkje G de Waart, and Hein Raat. 2014. Cyber and Traditional Bullying Victimization as a Risk Factor for Mental Health Problems and Suicidal Ideation in Adolescents. *PloS one*, 9(4).

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of SemEval*.

Aaron T Beck and Emily AP Haigh. 2014. Advances in cognitive theory and therapy: The generic cognitive model. *Annual review of clinical psychology*, 10:1–24.

Michael L Birnbaum, Raquel Norel, Anna Van Meter, Asra F Ali, Elizabeth Arenare, Elif Eyigoz, Carla Agurto, Nicole Germano, John M Kane, and Guillermo A Cecchi. 2020. Identifying signals associated with psychiatric illness utilizing language and images posted to facebook. *npj Schizophrenia*, 6(1):1–10.

Timo Brockmeyer, Johannes Zimmermann, Dominika Kulessa, Martin Hautzinger, Hinrich Bents, Hans-Christoph Friederich, Wolfgang Herzog, and Matthias Backenstrass. 2015. Me, myself, and i: self-referent word use as an indicator of self-focused attention in relation to depression and anxiety. *Frontiers in psychology*, 6:1564.

Çağrı Çöltekin. 2020. A Corpus of Turkish Offensive Language on Social Media. In *Proceedings of LREC*.

Stevie Chancellor and Munmun De Choudhury. 2020. Methods in predictive techniques for mental health status on social media: a critical review. *NPJ digital medicine*, 3(1):1–11.

Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, and Margaret Mitchell. 2015. Clpsych 2015 shared task: Depression and ptsd on twitter. In *Proceedings of CLPsych*.

Jonathan Culpeper. 2011. *Impoliteness: Using Language to Cause Offence*, volume 28. Cambridge University Press.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving Dyberbullying Detection with User Context. In *Proceedings of ECIR*.

Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013. Predicting depression via social media. In *Proceedings of ICWSM*.

Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. 2016. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of CHI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Ted E Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

Jonas Everaert, Ioana R Podina, and Ernst HW Koster. 2017. A comprehensive meta-analysis of interpretation biases in depression. *Clinical Psychology Review*, 58:33–48.

Costas Gabrielatos. 2018. Keyness analysis. *Corpus approaches to discourse: A critical review*, pages 225–258.

James J Gross. 1999. Emotion and emotion regulation. *Handbook of personality: Theory and research*, 2:525–552.

Eric Holgate, Isabel Cachola, Daniel Preoţiuc-Pietro, and Junyi Jessy Li. 2018. Why swear? analyzing and inferring the intentions of vulgar expressions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Adam Kilgarriff. 2009. Simple maths for keywords. In *Proceedings of the CL*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2020. Evaluating aggression identification in social media. In *Proceedings of TRAC*.

Ping Liu, Wen Li, and Liang Zou. 2019. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of SemEval*.

D. Losada and F. Crestani. 2016. A test collection for research on depression and language use. In *Proceedings of CLEF*.

Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. 2019. Hate speech detection: Challenges and solutions. *PloS one*, 14(8).

Prasenjit Majumder, Thomas Mandl, et al. 2018. Filtering Aggression from the Multilingual Social Media Feed. In *Proceedings TRAC*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Shervin Malmasi, Marcos Zampieri, and Mark Dras. 2016. Predicting post severity in mental health forums. In *Proceedings of CLPsych*.

JS Manohar, TS Sathyanarayana Rao, S Chandran, et al. 2017. Sexual dysfunctions in depression. *Clin Depress*, 3(125):1–5.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. In *Proceedings of AAAI*.

David N Milne, Glen Pink, Ben Hachey, and Rafael A Calvo. 2016. Clpsych 2016 shared task: Triaging content in online peer-support forums. In *Proceedings of CLPsych*.

Saif M Mohammad. 2017. Word affect intensities. In *Proceedings of LREC*.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational intelligence*, 29(3):436–465.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of NAACL*.

Zeses Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive Language Identification in Greek. In *Proceedings of LREC*.

Daniel Preoţiuc-Pietro, Johannes Eichstaedt, Gregory Park, Maarten Sap, Laura Smith, Victoria Tobolsky, H Andrew Schwartz, and Lyle Ungar. 2015. The role of personality, age, and gender in tweeting about mental illness. In *Proceedings of CLPsych*.

Tharindu Ranasinghe and Marcos Zampieri. 2020. Multilingual Offensive Language Identification with Cross-lingual Embeddings. In *Proceedings of EMNLP*.

Tharindu Ranasinghe and Marcos Zampieri. 2021. MUDES: Multilingual Detection of Offensive Spans. In *Proceedings of NAACL*.

Esteban A Ríssola, Seyed Ali Bahrainian, and Fabio Crestani. 2020. A dataset for research on depression in social media. In *Proceedings of UMAP*.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A large-scale semi-supervised dataset for offensive language identification. *Findings of the ACL*.

Stephanie Rude, Eva-Maria Gortner, and James Pennebaker. 2004. Language use of depressed and depression-vulnerable college students. *Cognition & Emotion*, 18(8):1121–1133.

Koustuv Saha, Eshwar Chandrasekharan, and Munmun De Choudhury. 2019. Prevalence and psychological effects of hateful speech in online college communities. In *Proceedings of WebSci*.

Susan A Speer. 2019. Reconsidering self-deprecation as a communication practice. *British Journal of Social Psychology*, 58(4):806–828.

Richard Stephens and Olly Robertson. 2020. Swearing as a response to pain: Assessing hypoalgesic effects of novel “swear” words. *Frontiers in Psychology*, 11:723.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of ALW*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of NAACL*.

Andrew Yates, Arman Cohan, and Nazli Goharian. 2017. Depression and self-harm risk assessment in online forums. In *Proceedings of EMNLP*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of SemEval*.

Justine Zhang, Jonathan Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Dario Taraborelli, and Nithum Thain. 2018. Conversations Gone Awry: Detecting Early Signs of Conversational Failure. In *Proceedings of ACL*.

Ayah Zirikly, Philip Resnik, Ozlem Uzuner, and Kristy Hollingshead. 2019. Clpsych 2019 shared task: Predicting the degree of suicide risk in reddit posts. In *Proceedings of CLPsych*.

Transforming Term Extraction: Transformer-Based Approaches to Multilingual Term Extraction Across Domains

Christian Lang*, Lennart Wachowiak*, Barbara Heinisch, Dagmar Gromann
University of Vienna, Center for Translation Studies
`[christian.lang,lennart.wachowiak]@univie.ac.at`

Abstract

Automated Term Extraction (ATE), even though well-investigated, continues to be a challenging task. Approaches conventionally extract terms on corpus or document level and the benefits of neural models still remain underexplored with very few exceptions. We introduce three transformer-based term extraction models operating on sentence level: a language model for token classification, one for sequence classification, and an innovative use of Neural Machine Translation (NMT), which learns to reduce sentences to terms. All three models are trained and tested on the dataset of the ATE challenge TermEval 2020 in English, French, and Dutch across four specialized domains. The two best performing approaches are also evaluated on the ACL RD-TEC 2.0 dataset. Our models outperform previous baselines, one of which is BERT-based, by a substantial margin, with the token-classifier language model performing best.

1 Introduction

Automated Term Extraction (ATE) aims at extracting terms, i.e., single- or multi-word sequences, from domain-specific text. ATE plays a role in many NLP tasks, such as information extraction, knowledge graph learning, and text summarization. In a corpus-level setting, methods range from frequency-based to utilizing Wikipedia links, where no single method has been found to perform consistently best across domains in English (Astrakhantsev, 2018). In document-level ATE, KeyConceptRelatedness (Astrakhantsev, 2014), which relies on keyphrase extraction and semantic relatedness, outperforms other methods (Šajatović et al., 2019). The use of neural networks in these methods is mostly limited to generating embeddings.

A first use of BERT-based language models is documented by Hazem et al. (2020), the winning system of the recent ATE challenge TermEval 2020 (Rigouts Terryn et al., 2020) and the baseline for the proposed approaches. Inspired by this first success of transformer-based models, we compare two variations of the multilingual pretrained language model XLM-RoBERTa (XLM-R) (Conneau et al., 2020) with an innovative use of the multilingual pretrained NMT model mBART (Liu et al., 2020) on the Annotated Corpora for Term Extraction Research (ACTER) dataset (Rigouts Terryn et al., 2019) utilized in TermEval 2020 as well as on the ACL RD-TEC 2.0 dataset (QasemiZadeh and Schumann, 2016). Since masked language and NMT models take sentences as input, the proposed ATE methods operate on sentence level. In spite of this reduced context of sentence input rather than documents or corpora, the models achieve F1 scores of up to 69.8% on ACTER, strongly outperforming the previous baseline of 48.1% .

An XLM-R-based sequence classifier relies on positive (term) and negative (non-term) samples, which are generated based on all n-grams up to a length of six of a given sentence. A second XLM-R-based token classifier decides for each word in a sequence whether it can be considered (part of) a term. Since the second model operates without upfront n-gram generation and only processes each sentence once, it is considerably more time-efficient than the first. Finally, the pretrained NMT model mBART is adapted to transform input sentences to sequences of comma-separated terms, an approach inspired by NMT-based ontology learning (Petrucci et al., 2018).

Analyses of results reveal interesting insights into the performance of the different input processing strategies and transformer-based models, including their ability to handle multi-word terms, training time required, and a comparison between baseline monolingual and multilingual language models in ATE. To achieve sentence-level ATE

---

* Equal contributions

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3607–3620
August 1–6, 2021. ©2021 Association for Computational Linguistics

the ACTER dataset had to be preprocessed aligning terms with their occurrences in sentences, which we made publicly available together with our source code.[1]

In summary, our main contributions are: (i) We show that transformer-based models can be successfully applied to ATE across three languages and five domains, without the need for text preprocessing or feature extraction; (ii) We show that ATE can be performed successfully on sentence level; (iii) We conduct robust experiments to show that our models outperform competitive baselines; (iv) We investigate the models' abilities to handle single- and multi-word terms, distinct term types, and differences in performance depending on train and test language combinations.

2 Related Work

An initial classification of ATE methods into statistical, linguistic or hybrid (e.g. by Kageura and Umino (1996)) has recently been refined by Astrakhantsev (2018) to methods based on term occurrence frequencies (e.g. C/NC-value Frantzi et al., 2000), occurrence contexts (e.g. Bordea et al., 2013), domain-specific corpora combined with general language corpora (e.g. Weirdness (Ahmad et al., 1999)), topic modeling (e.g. Li et al., 2013), and those utilizing Wikipedia. Methods are additionally categorized by the type of context, i.e., corpus-level (e.g. Zhang et al., 2008; Astrakhantsev, 2018) and document-level (e.g. Šajatović et al., 2019) settings.

These classifications cannot easily accommodate recent neural ATE methods that generally operate on sentence level. An approach most closely related and our baseline by Hazem et al. (2020) utilized RoBERTa (Liu et al., 2019) for English and CamemBERT (Martin et al., 2020) for French and won the TermEval 2020 challenge. In their work, pretrained language models clearly outperformed a classification method based on a variety of features, such as statistical descriptors and the domain-specificity measure termhood (Kageura and Umino, 1996). A recently published approach (Rokas et al., 2020) relies on LSTM, GRU and BERT embeddings and achieves high F1 scores for ATE of Lithuanian terms in the cybersecurity domain. Several approaches build on word embeddings to perform ATE on specific domains, such as medicine (e.g. Bay et al., 2020), or to separate general-language from domain-specific embeddings (Hätty et al., 2020). In contrast, our models perform ATE on four domains and in three languages utilizing a pretrained language and a pretrained NMT model. Extracting terms is also vital to learning expressive ontologies from text, for which Petrucci et al. (2018) train an NMT model to transform sentences to Description Logic formulas, an idea that inspired our NMT-based ATE model.

3 Language Models and NMT

Neural Language Models, which create contextualized language representations, were responsible for many of the recent improvements in NLP. Such models acquire rich contextualized language representations in a pretraining stage in which they learn to predict a masked word in a sentence, a task for which large amounts of training data are readily available. The thereby learned representations can be reused for various downstream tasks in the so-called fine-tuning stage, where task-specific layers are added on top of the pretrained language model. One of the most popular language models is BERT (Devlin et al., 2019), utilizing the transformer architecture (Vaswani et al., 2017). XLM-R (Conneau et al., 2020) is a multilingual variant of BERT, which was pretrained in 100 languages using 2.5 terabytes of Common Crawl data. Moreover, it makes use of the improved training routine introduced by RoBERTa (Liu et al., 2019).

Despite the widespread use of neural language models for NLP, adoption of such self-supervised pretraining approaches in NMT has only recently started to gain traction. NMT is traditionally performed with sequence-to-sequence encoder-decoder models that generate a target language output sequence based on a source language input sequence. Conventional language models trained on predicting masked words from a sequence, such as BERT, have only recently been incorporated into NMT (Zhu et al., 2020). A very interesting alternative is to pretrain an NMT transformer architecture, as done by Lewis et al. (2020) in form of a Bidirectional and Autoregressive Transformer (BART) (Lewis et al., 2020). This is achieved by combining a bidirectional encoder similar to that of BERT with an autoregressive decoder, as seen in GPT (Radford et al., 2018). Thereby, contextualized language representations are trained and

---

[1] `https://github.com/Text2TCS/Term-Extraction-With-Language-Models` and `https://github.com/Text2TCS/mBART-termextraction`

a model that is proficient in text generation and translation is created. Liu et al. (2020) applied the BART architecture to large-scale monolingual corpora across 25 languages, creating multilingual BART (mBART) that can be directly fine-tuned for machine translation (MT).

4 Dataset

In order to compare to a strong baseline, we train and test on the ACTER dataset (Rigouts Terryn et al., 2019) utilized in the recent TermEval 2020 challenge. The domains wind energy and corruption represent the training set, dressage (equitation) the validation set, and heart failure the hold-out test set, for which the count of words and unique gold standard terms including named entities for English, French and Dutch are presented in Table 1.

In the ACTER dataset, words were labeled as specific, common, and out-of-domain (OOD) terms, and named entities (NE). Specific terms are understood by domain experts, while common terms might also be additionally understood by laypersons. OOD terms might be specific to a different domain, but used in the domain at hand, e.g. statistical terms in the medical domain.

Since the time of the challenge the dataset has undergone some minor updates, that is, unicode encoding, dash and quote normalization.[2] We believe that these minor normalization changes do not significantly impact comparability to TermEval results, which is confirmed by the fact that our most similar model to the baseline, the sequence classifier, achieves comparable results. Furthermore, the ACTER dataset provides terms as a single list for all documents in a domain. However, we required inline sentence-level term annotation, which we generated. In rare cases, this generation of inline annotations might have lead to erroneous results for single-word terms. For instance, the term "gain" as in "private gain" lead to the verb "gain" as in "gain acceptance" to be erroneously annotated in the corruption domain. We manually analyzed 300 inline annotated sentences and since the above example was the only error found, we consider this a negligible issue.

The fully inline annotated dataset ACL RD-TEC 2.0 (henceforth ACLR2) dataset provides cleaner training and test data and could therefore potentially further boost model performance as we show in Section 7.2. The ACLR2 dataset provides a total of 471 inline human annotated abstract texts from articles in the ACL Anthology Reference Corpus. As shown in the split of numbers in Table 1, two separate annotations by two human experts are provided. Since no official train/val/test split is provided, we chose to split the ACLR2 dataset with a 60/20/20 split per annotator. In contrast to the ACTER dataset, ACLR2 is only available in English and exclusively covers scientific abstracts in the domain of computational linguistics. In terms of baseline, previous work generally reported precision at *k* top terms extracted (P@k) (Zhang et al., 2018b) or F1 on Recoverable True Positives (F1@RTP) (Zhang et al., 2018a), due to the necessity to define an arbitrary cut-off point with traditional ATE methods. In another work attempting ATE with neural networks, due to the lack of an official data split and a restriction to domain specific terms, F1 scores are reported on arbitrary parts of the dataset (Kucza et al., 2018).

ACTER	**Train**	**Val**	**Test**
Words$_{en}$	97,145	51,470	45,788
Terms$_{en}$	2,708	1,575	2,585
Words$_{fr}$	106,792	53,316	46,751
Terms$_{fr}$	2,185	1,183	2,423
Words$_{nl}$	96,887	50,882	47,888
Terms$_{nl}$	2,540	1,546	2,257
ACLR2	**Train**	**Val**	**Test**
Words$_{an.1}$	11,473	3,846	4,032
Terms$_{an.1}$	1,306	420	477
Words$_{an.2}$	16,939	5,757	5,441
Terms$_{an.2}$	1,743	583	673

Table 1: Train, validation, and test split by word count and term count per language/annotator

5 Neural Language Model-based ATE

We introduce two possible architectures for ATE based on the multilingual language model XLM-R. For the experiments we use the base-size model version in form of the implementation made available by the transformers library (Wolf et al., 2019).

5.1 Sequence Classifier

As with the winning approach of TermEval 2020 (Hazem et al., 2020), our first architecture utilizes language models for binary sequence classification by using a fully connected layer to classify the representation of the special classification token `<s>`,

---

[2] This normalized version 1.4 is available at `https://github.com/AylaRT/ACTER`

which encoded by XLM-R carries information regarding the whole input sequence. Instead of using language specific models, however, we make use of the multilingual model XLM-R, which enables the use of a single model for all languages and has the ability to generalize to unseen languages.

The model receives pairs consisting of a term candidate and a context sentence in which the candidate appears as input as exemplified in Table 2. Term candidates are created by producing all possible n-grams of a given sentence. Due to performance reasons and the term length distribution in the dataset (mostly <5 words), n-grams were only created up to a length of 6 words. For instance, given the input sentence "We meta-analyzed mortality using random-effect models" a positive sample, i.e., one labeled as term, is "random-effect models. We meta-analyzed mortality using random-effect models", while a negative sample is "mortality using. We meta-analyzed mortality using random-effect models". For training the model, we undersample the negative samples so that their amount matches the amount of positive samples to compare to Hazem et al. (2020). For the evaluation on the validation and test set we use all possible n-grams for each input sentence, thus, creating a set of extracted terms which we can evaluate against the gold standard. The model was trained for 4 epochs with a batch size of 32 using the Adam optimizer with a learning rate of 2e-5.

5.2 Token Classifier

The second architecture we use for experimentation classifies each token of an input sentence separately, utilizing the same fully connected layer for all tokens after they have been processed by XLM-R. This leads to a significant reduction in training and inference time as each sentence has to be only processed once by XLM-R. This type of architecture is usually utilized in tasks like Named Entity Recognition (NER) (Devlin et al., 2019), where each word of a sequence needs to be classified.

The input provided to the model now simply consists of the sentences of the document which we want to process. The model then assigns each input token one of three possible output labels: "B-T" for the beginning of a term, "T" for the continuation of a term, and "n" in the case the token is not part of a term. For instance, the input sentence "We meta-analyzed mortality using random-effect models." would be labeled as 'n', 'B-T', 'B-T', 'n', 'B-T', 'T', 'n', with the last label annotating the punctuation at the end of the sentence.[3] Table 2 compares this input and output pattern with the other two methods. Since XLM-R's tokenizer is a SentencePiece tokenizer that splits the input into tokens on a subword level, the output labels obtained from the model are also subwords and have to be matched to the original words of the sentence afterwards. For training we used the Adam optimizer with a learning rate of 2e-5. Moreover, we used a batch size of 8 evaluating the model every 100 steps to be able to load the best model at the end.

6 NMT-based ATE

As a third experiment, we present a novel approach to ATE building on a recent sequence-to-sequence denoising auto-encoder model trained for NMT. We chose the recent and robust mBART model trained on the Common Crawl corpus in 25 languages (mBART25) (Liu et al., 2020) available in the Fairseq library (Ott et al., 2019).

6.1 Data Preprocessing for NMT-based ATE

Since we construct the downstream task of ATE as an MT task, we required parallel text data for supervised fine-tuning of mBART. We opted for a sentence-level approach, which specifically requires sentence-aligned parallel data. Sentence tokenization was performed with the Punkt tokenizer of NLTK and terms were inline annotated with the flashtext algorithm (Singh, 2017). For the ACLR2 dataset, individual sentences and the terms within were extracted with an XML parser. In order to distinguish single- and multi-word terms in the model's output sequence, a separator between terms or a unifying character between components of multi-word terms was required. Preliminary testing showed that using a semicolon surrounded by white-spaces (` ; `) as separator would achieve the same final F1 score as using more complex separators like a tag (for example `<term>`). Notably using an underscore (w_w) to connect the individual constituents of a term (w) lowered the score of the output significantly, that is, F1 performance of the best model was 5.3% lower on average across all test languages when compared to utilizing semicolons. Irrespective of the separator, the model would at times add or omit a white-space between separator and term, which had the effect that the

---

[3] The separation of "meta-analyzed" and "mortality" as distinct terms corresponds to the gold standard.

Model	Input Example	Output Example
Sequence Classifier	random-effect models. We meta-analyzed mortality using random-effect models	Term
Token Classifier	We meta-analyzed mortality using random-effect models	['n', 'B-T', 'B-T', 'n', 'B-T', 'T', 'n']
NMT	We meta-analyzed mortality using random-effect models	meta-analyzed ; mortality ; random-effects models

Table 2: Input and output examples for all three transformer-based models

term would not be considered in the evaluation. This was remedied in the process of extracting individual terms from the output sequence and the results reported in Section 7 are with unwanted white-spaces removed. Tokenization during training was performed with SentencePiece (Kudo and Richardson, 2018) and data was binarized with the *fairseq-preprocess* CLI tool.

6.2 NMT Model Fine-Tuning

The pretrained mBART model was fine-tuned with the preprocessed data described in Section 6.1. Input to the encoder model was a given sentence, such as “Codes of conduct forbid corruption, irrespective of its intended purpose.”, while the decoder would be shown the expected term labels, such as “codes of conduct ; corruption”. No language-specific tags were added to input or output, which is compared to the other methods in Table 2. For faster and more memory-efficient training we used automated mixed precision training of Fairseq with the Fused Adam Optimizer of the NVIDIA Apex PyTorch extensions.[4] We fine-tuned a separate model for each language of the dataset and a single model with all languages combined. Following the original publication of the pretrained model, each model was fine-tuned with 0.3 dropout, 0.2 label smoothing, 2500 warm-up steps and a learning rate of 3e-5. Furthermore, we opted for a dynamic batch size by limiting the maximum tokens per batch to 768, while updating the gradients every 4 steps (more details in Section 7.4).

While preliminary testing showed faster convergence and slightly higher final scores with higher tokens per batch, availability of the V100 GPU was not guaranteed and therefore training hyperparameters had to be adjusted to also run on an RTX2080Ti GPU, which limited the maximum tokens per GPU to 768. Model performance was evaluated every full epoch. Results were generated using the standard generation parameters of Fairseq.

7 Results

This section first presents the results on ACTER including an analysis per language (combination) and the results on ACLR2, then details the term length and type behavior of the models, and finally compares their training time efficiency. We additionally report on the validation performance of the best performing token classifier in Table 4, which shows some performance differences to the test domain, especially with French as training and validation language. For further comparability we also provide precision, recall and F1 scores at k top terms of 15 methods offered by the term extraction toolkit ATR4S, which implements a large range of existing ATE methods, in Appendix A.

7.1 Results on ACTER

To compare our results to the strongest participant of TermEval 2020, we report precision, recall and F1 scores in Table 3. These metrics are calculated on the basis of the available annotation in the original ACTER dataset, where we opted for the more comprehensive list of terms including named entities. All three models are evaluated on different combinations of training and test languages as shown in Table 3, where the heart failure domain is the hold-out test set as done for the SOTA baseline. The overall best results are marked in bold for each test language, while the best results of each model (if not bold) are highlighted in italics.

The overall best result for our approaches was an F1 score of 69.8%, which could be achieved by training the token classifier model on English and testing it on Dutch. With the exact same settings as the baseline (Hazem et al., 2020) that is based on RoBERTa (Liu et al., 2019) for English as training and test language, the token classifier

---

[4] `https://github.com/NVIDIA/apex`

Training	**Test**	**Sequence Classifier**			**Token Classifier**			**NMT**			**Previous SOTA**		
		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
EN	EN	30.9	84.0	45.2	54.9	62.2	58.3	45.7	63.5	53.2	34.8	70.9	46.7
FR	EN	31.1	79.5	44.7	56.7	36.2	44.2	50.0	59.3	54.2			
NL	EN	22.3	91.1	35.9	55.3	61.8	**58.3**	48.3	64.3	55.2			
ALL	EN	31.4	85.8	*46.0*	54.4	58.2	56.2	50.2	61.6	*55.3*			
EN	FR	34.6	79.0	*48.1*	65.4	51.4	**57.6**	48.8	61.3	54.4			
FR	FR	32.2	80.2	46.0	68.7	43.0	52.9	52.7	59.6	55.9	44.2	51.5	48.1
NL	FR	26.1	84.7	40.0	62.3	48.5	54.5	54.3	60.9	57.4			
ALL	FR	33.2	78.9	46.7	62.7	49.4	55.3	55.0	60.4	*57.6*			
EN	NL	42.8	89.8	*58.0*	67.9	71.7	**69.8**	48.8	63.9	55.4			
FR	NL	41.3	87.6	56.1	69.2	55.2	61.4	56.2	63.4	59.6			
NL	NL	32.7	94.1	48.5	71.4	67.8	69.6	60.6	70.7	*65.2*	18.9	18.6	18.7
ALL	NL	40.4	91.5	56.0	70.0	65.8	67.8	60.6	70.0	64.9			

Table 3: Test set results represented by training and test languages of the ACTER heart failure domain and in comparison to the state-of-the-art (SOTA) results from TermEval 2020.

Training	**EN Val**	**FR Val**	**NL Val**
EN (ACTER)	55.6	45.3	60.5
FR (ACTER)	41.9	33.6	49.6
NL (ACTER)	54.6	47.7	57.8
ALL (ACTER)	50.0	40.4	51.5
ACLR2 An.1	75.5	/	/
ACLR2 An.2	79.3	/	/

Table 4: Validation performance of token classifier on dressage domain of the ACTER dataset and 20% validation data of the ACLR2 dataset.

achieves an 11.6% higher F1 score and the NMT model an improvement of 6.5% on the F1 score. The sequence classifier struggles with precision and cannot outperform the baseline in this setting. Best performance for English as test language can be achieved by the token classifier trained on Dutch and by the NMT model trained on all languages.

When testing on French, the sequence classifier is on par with the F1 baseline (Hazem et al., 2020) building on CamemBERT (Martin et al., 2020), while the token classifier outperforms it by 9.5% and the NMT model obtains an additional 7.8%. Best performance on French as a test language is achieved by the token classifier when trained on English and by the NMT model when trained on all languages again. The baseline for Dutch is provided by a bidirectional LSTM with GLOVE.[5] With Dutch as a test language, the sequence and token classifier achieve their best result when trained on English, the NMT model when trained on Dutch.

A significant result is the substantial improvement of precision of the token classifier and NMT model over the baseline, even though the recall for English as test language lags behind. For French, the recall could be improved with the NMT model and matched by the token classifier when trained on English. Interestingly, the sequence classifier achieves a remarkable improvement on recall, however, lags behind on precision for all settings.

This can be explained by the fact that we perform undersampling of the negative samples to match the number of positive samples, a strategy adopted from Hazem et al. (2020) to obtain comparable results. If undersampling is reduced, the precision and recall scores are more balanced and closer to the performance of the token classifier, however, training time is considerably increased. Another reason for the higher number of extracted phrases by the sequence classifier compared to the other models is that it can extract multi-word terms as well as words which are part of these multi-word terms separately, since both are used as input in the form of potential term candidate n-grams.

All three models show remarkable zero-shot transfer learning capabilities, i.e., they are trained on one language and show strong test scores on another. This is especially true for the token classifier, where models trained on a single language often outperform those trained on all three languages. This transfer learning ability across languages can also be observed in the overall highest F1 scores

---

[5] No system description paper was submitted for this approach after participation in the challenge.

for the English test set, which was achieved by a model trained on Dutch, and for the French test set, which was achieved by a model trained on English.

7.2 Results on ACLR2

In addition to evaluating our models on the ACTER dataset, we compared the two best performing architectures, i.e., the token classifier and the NMT model, on the ACLR2 dataset. Both models achieve similar test scores as reported in Table 5 and higher than the scores achieved on the ACTER dataset. As with the ACTER dataset, we additionally report validation performance of the best performing token classifier model in Table 4, which is in line with the test performance.

Data	**Token Classifier**			**NMT**		
	Prec	Rec	F1	Prec	Rec	F1
An.1	74.4	77.2	75.8	73.2	77.2	75.2
An.2	80.1	79.3.	80.0	79.4	80.7	80.0

Table 5: Test set results of token classifier on data from Annotator 1 and 2 of the ACLR2 dataset.

7.3 Term-based Analysis

A qualitative analysis of the lists of false positives and false negatives based on the ACTER dataset demonstrated that all models handle acronyms well. This may be due to the text type in ACTER, which is partially based on scientific abstracts that frequently introduce acronyms in brackets. If acronyms are part of the term, e.g. "LV strain rate", there was a high number of false negatives in both models. Moreover, false negatives occurred in all models if a term included a proper name and an apostrophe, e.g. "Chaga's disease" or "Cronbach's α", or frequently if it included a figure, e.g. "p38alpha" or "6-min walk test". In addition, named entities that included version numbers or consisted of multiple words often resulted in false negatives, e.g. "Self-Care of Heart Failure Index Version 6.2", "Multicenter Automatic Defibrillator Implantation Trial-Cardiac Resynchronization Therapy". In the token classifier and NMT model, the class of named entities of cities, e.g. "New York" and "Seattle", were frequently not identified as terms. False negatives also occurred in all models if it was a particularly long multi-word term, e.g. "resynchronization reverses remodeling in systolic left ventricular dysfunction". A tendency by the token classifier to split longer terms could be observed, e.g. splitting adjectives and nouns.

To quantitatively evaluate how well the different model types handled terms of different lengths, we computed the F1 scores individually for terms of a specific length, based on the terms in the ACTER test set. The results in Table 6 were computed using the best model of each method, i.e., the model trained in English for the token and the sequence classifier and the model trained on all language for the NMT model. We can see that the scores of all models decrease with term length. Secondly, we observer that for English and Dutch the token classifier has the strongest results for all term lengths. However, for French the token classifier scores strongly decrease for multi-word terms, even though it is still the best model for unigrams. This is due to a very low recall, e.g. for 4-grams and higher the token classifier recalls only 7% of all French terms. The NMT model shows more consistency between languages, thus, performing strongest for French multi-word terms. As already the case with the overall scores the sequence classifier shows the highest recall values for both single-word and multi-word terms, however, lagging behind in precision, which leads to an overall lower F1 score.

Furthermore, based on the ACTER term type annotation (see Section 4), we could compare the types of terms extracted by the individual models. As can be seen in Fig. 1, the models all achieve a very similar distribution of extracted term types when compared to the gold test set distribution. We can observe, however, that the sequence classifier showed a slight tendency to extract more common and OOD terms and noticeably less NEs than the other models. All models tended to extract more specific terms, with the token classifier and the NMT model interestingly extracting comparatively few OOD terms.

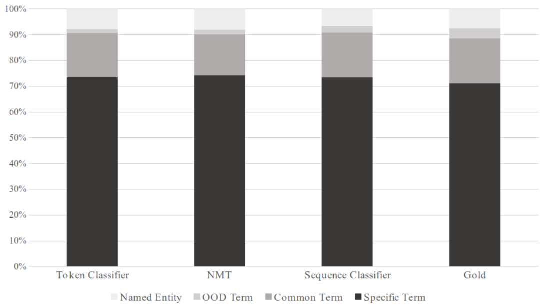

Figure 1: Distribution of term types across languages in the models' true positives and the ACTER gold test dataset.

Term Length	Sequence Classifier (F1)			Token Classifier (F1)			NMT (F1)		
	EN	FR	NL	EN	FR	NL	EN	FR	NL
Unigram	61.6	61.2	68.0	**63.3**	**69.1**	**73.8**	61.7	61.8	70.4
Bigram	38.0	41.0	39.8	**55.9**	43.0	**58.4**	52.2	**57.6**	56.3
Trigram	37.0	35.4	40.4	**55.4**	31.3	**49.1**	51.6	**52.4**	47.7
$\geq$ 4-gram	32.2	22.6	30.3	**44.3**	12.2	**42.9**	43.4	**33.4**	38.8

Table 6: F1 Scores based on different term lengths using the overall best model for each method on the ACTER dataset. In bold the best scores per row for each language.

7.4 Training Time Efficiency

Looking at the epochs required to reach the best score on the ACTER validation set, we can observe that in most cases the token classifier model requires not even a single training epoch. Training with the English dataset required 300 steps with a full epoch consisting of 432 steps. The model trained on French was the only model with its best performance being reached during the second epoch after 700 steps while a full epoch consists of 437 steps. The model trained on Dutch performed best after 400 steps while one epoch takes 553 steps. The multilingual model converged the quickest needing only 200 steps whereas a full epoch consists of 1,421 steps. The token classifier models trained on the ACLR2 dataset need more epochs and achieve their highest scores after 3 and 5 epochs respectively. However, due to the lower training set size of the ACLR2 corpus, this also corresponds to less than 500 steps, thus, being similar with the training times reported for the models trained on the ACTER data. In comparison, the sequence classifier achieved its best performances on the ACTER validation set after 4 epochs of training.

The NMT model also required several epochs to reach the best performance. Initially, all models were trained for 80 epochs, with the model having the lowest validation loss being loaded at the end. The models trained on monolingual data benefited from longer training compared to the models trained on the combined multilingual data. For completeness, we report the training epochs, label smoothed cross entropy loss, and log perplexity on the validation set for the best models. For the English dataset the reported score was achieved at epoch 49 with a loss of 5.82 and perplexity of 3.94. For the French dataset peak performance was reached at epoch 40, with a loss of 5.82 and perplexity of 3.78. Like the French model, the Dutch model achieved its best performance at epoch 40 having a loss of 5.69 and a perplexity of 3.37. When trained on one language, model performance was observed to drop for unseen languages when training beyond the best validation score. For instance, while the English model at epoch 49 obtained F1 scores of 53.2%, 54.4%, 55.4% for the English, French, and Dutch test data respectively, at epoch 80 these scores were at 53.6%, 50.6% and 52.1% respectively, gaining little for English and losing for unseen languages. Finally, for the multilingual dataset the model reached the reported peak performance already at epoch 22 as it trains on a lot more data per single epoch. Loss and perplexity were at 5.50 and 2.89 respectively. The training and validation times as well as the used GPUs are reported in Table 7. Training times denotes the full training time over all epochs without any validations. Validation time denotes the time for a single validation. The token classifier is the most efficient.

Model	Train Time	Val Time	GPU
Seq.	19	44	P100
Tok.	9	1	P100
NMT	49	2	V100

Table 7: Training/validation times in minutes on the English ACTER data and GPUs used.

8 Discussion

Although the ACLR2 dataset is smaller in size than the ACTER dataset, the resulting F1 scores are considerably higher. Apart from the fact that it only covers a single domain, ACLR2 already provides inline annotations and more consistent term annotations, which seems to facilitate learning the task. Inconsistencies in the ACTER annotations were mainly noted when analyzing false positives of the models. For instance, “patient” is considered a common term in the heart-failure domain, but “serum” is not annotated at all, although in our view it would also qualify as common term.

We also noted that more training data does not necessarily increase model performance. As indicated by the training times on the ACTER dataset, the token classifier achieved its best evaluation scores long before training for a whole epoch, i.e., having seen only a small fraction of the available data before reaching its strongest performance.

In this paper we compare the performance of a pretrained monolingual language model baseline with pretrained multilingual language models. Previous work indicates that monolingual language models like RoBERTa or CamemBERT outperform multilingual language models on tasks posed in a single language (Rönnqvist et al., 2019). The difference increases the higher the complexity of the given task but is negligible on simple tasks that mostly rely on syntactic features. Since in our case the multilingual model XLM-R in form of a sequence classifier performs very similar to the sequence classifier-based RoBERTa model winning TermEval 2020, it indicates that successful ATE does not require very strong language understanding but corresponds more to simpler tasks relying mostly on syntactic features. Nevertheless, the remarkable zero-shot transfer learning of the multilingual models fine-tuned on a single language would also suggest that the multilingual pretraining might aid the model in defining what a term is, as highly domain-specific terms might be similar between languages tested, e.g. rooted in Latin. In the NMT output analysis, we found that the knowledge transfer between languages could cause curious side-effects, where at times terms are predicted by the model in a semi-translated way. For instance, when training on English the model would at times invent “toxicity cardiaque ” for the French test set instead of extracting “toxicité cardiaque”.

Besides stronger performance, the NMT model as well as the token classifier have a higher potential to better handle the possible extension of the term extraction task to include discontinuous entities, which, however, are so far not annotated in the datasets we used. An example of a discontinuous entity can be found in the expression “left and right ventricular failure”, where “right ventricular failure” but also “left ventricular failure” are terms, the latter not being continuous in the original expression. While the NMT model does not require any special adaptations to deal with such an addition, the sequence classifier would have to consider many more n-gram combinations leading again to even higher training and inference times per sentence. To consider discontinuous entities with the token classifiers labels, the annotation and training process would have to be adapted to a multi-label token classification, e.g. the above phrase would be labeled as [B-T, n, n, T, T] and [n, n, B-T, T, T]. Since in the first label “ventricular” and “failure” are labeled as “T” they still clearly belong to the word “left” labeled as “B-T”, which could be considered in a post processing step.

9 Conclusion

In this paper, we adapt and evaluate three transformer-based models on the task of ATE, building on pretrained multilingual language and NMT models. In this evaluation, these multilingual models outperform a baseline of monolingual language models and show remarkable zero-shot abilities. A token classification strategy building on a language model achieved the best performance, however, the NMT-based model seemed to be able to handle multi-word expressions more consistently across languages and not lag far behind in performance. One aspect that became very clear is a prevalence for quality over quantity when fine-tuning pretrained models to the task of ATE.

Recently, both NMT and masked language models show a trend towards increased input sequence capacity. Thus, it would be interesting to evaluate the impact of context length on the proposed models by testing with more domain context than only single sentences. Furthermore, to test the ability of the token classifier and the NMT model to handle discontinuous terms, such as elliptical expressions, a dataset containing and annotating such terms would be interesting.

Acknowledgments

The Text2TCS project[6] was supported by the European Language Grid project through its open call for pilot projects. The European Language Grid project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 825627 (ELG). The computational results presented have been achieved [in part] using the Vienna Scientific Cluster (VSC).

---

[6] `https://text2tcs.univie.ac.at/`

Impact Statement

Automatically extracting domain-specific terms across domains and languages with high accuracy provides a valuable means to reduce time and resource effort in creating terminological resources. Such resources are important to ensure terminological consistency in specialized communication, such as communication between different groups in times of crisis, and to avoid misunderstandings.

From a technological perspective, we introduce multilingual pretrained language models to the field of Automated Term Extraction (ATE) with detailed tests on three different transformer-based models across four domains and three languages. Since these models support considerably more languages than tested, the approach can be transferred to other languages. This transfer capability has been tested by training in a specific language and then testing models in another language. Transfer capabilities extend to domains, since we trained and validated on three domains and achieved results strongly outperforming previous approaches on a previously unseen test domain. Up to this point, such flexibility has been achieved by statistical approaches, however, with considerably lower results in precision and recall. In contrast to previous ATE methods performing on corpora, our models extract terms on sentence level. This makes ATE more flexible since neither large domain-specific nor reference corpora are required.

From a societal perspective, terminological inconsistencies are a major source of misunderstanding in the communication among experts, between experts and laypersons, and between laypersons in reference to a specialized domain. This issue can be mitigated by publishing agreed upon designations for real-world phenomena in a specialized domain that can be consulted for domain-specific communication. However, manually preparing a collection of natural language terms is extremely human resource- and time-intensive. We reduce this workload for governmental institutions, private and public organizations, and private persons by providing a method to automate the detection of such domain-specific terms in natural language texts across languages and domains.

In terms of risk, such a highly flexible solution to automated term extraction fully depends on the quality of the input text. Misleading, erroneous, or biased contents will inevitably be propagated to the resulting terminologies. Relying on terminologies extracted from such problematic contents can negatively impact specialized communication or conclusions drawn from it. Thus, it is of vital importance for any user of this approach to mitigate the uncertainty of the reliability of extracted terms by only considering high-quality and reliable sources in the term extraction process and have domain experts carefully review the outcome prior to utilizing it in communication. We cannot guarantee that in a real-life setting all important terms have been extracted and all extracted terms are indeed central to the domain at hand. Furthermore, training neural network models is a process known to leave an environmental footprint, which we try to mitigate by fine-tuning pretrained models. Fine-tuning is less resource- and time-intensive than training from scratch, but still requires high-performance computing clusters.

References

Khurshid Ahmad, Lee Gillam, Lena Tostevin, et al. 1999. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In *TREC*, pages 1–8.

Nikita Astrakhantsev. 2014. Automatic term acquisition from domain-specific text collection by using wikipedia. *Proceedings of the Institute for System Programming*, 26(4):7–20.

Nikita Astrakhantsev. 2018. ATR4S: toolkit with state-of-the-art automatic terms recognition methods in scala. *Language Resources and Evaluation*, 52(3):853–872.

Matthias Bay, Daniel Bruneß, Miriam Herold, Christian Schulze, Michael Guckert, and Mirjam Minor. 2020. Term extraction from medical documents using word embeddings. In *4th IEEE Conference on Machine Learning and Natural Language Processing (MNLP 2020)*. IEEE Computer Society.

Georgeta Bordea, Paul Buitelaar, and Tamara Polajnar. 2013. Domain-independent term extraction through domain modelling. In *The 10th international conference on terminology and artificial intelligence (TIA 2013), Paris, France*. 10th International Conference on Terminology and Artificial Intelligence.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International journal on digital libraries*, 3(2):115–130.

Anna Hätty, Dominik Schlechtweg, Michael Dorna, and Sabine Schulte im Walde. 2020. Predicting degrees of technicality in automatic terminology extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2883–2889, Online. Association for Computational Linguistics.

Amir Hazem, Mérieme Bouhandi, Florian Boudin, and Beatrice Daille. 2020. TermEval 2020: TALN-LS2N system for automatic term extraction. In *Proceedings of the 6th International Workshop on Computational Terminology*, pages 95–100, Marseille, France. European Language Resources Association.

Kyo Kageura and Bin Umino. 1996. Methods of automatic term recognition: A review. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 3(2):259–289.

Maren Kucza, Jan Niehues, Thomas Zenkel, Alex Waibel, and Sebastian Stüker. 2018. Term extraction via neural sequence labeling a comparative evaluation of strategies using recurrent neural networks. In *Proceedings of INTERSPEECH 2018*, pages 2072–2076.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Sujian Li, Jiwei Li, Tao Song, Wenjie Li, and Baobao Chang. 2013. A novel topic model for automatic term extraction. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 885–888.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Giulio Petrucci, Marco Rospocher, and Chiara Ghidini. 2018. Expressive ontology learning as neural machine translation. *Journal of Web Semantics*, 52:66–82.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1862–1868, Portorož, Slovenia. European Language Resources Association (ELRA).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Ayla Rigouts Terryn, Veronique Hoste, Patrick Drouin, and Els Lefever. 2020. TermEval 2020: Shared task on automatic term extraction using the annotated corpora for term extraction research (ACTER) dataset. In *Proceedings of the 6th International Workshop on Computational Terminology*, pages 85–94, Marseille, France. European Language Resources Association.

Ayla Rigouts Terryn, Véronique Hoste, and Els Lefever. 2019. In no uncertain terms: a dataset for monolingual and multilingual automatic term extraction from comparable corpora. *Language Resources and Evaluation*, 54:385–418.

Aivaras Rokas, Sigita Rackevičienė, and Andrius Utka. 2020. Automatic extraction of lithuanian cybersecurity terms using deep learning approaches. In *Human Language Technologies–The Baltic Perspective*, volume 328, pages 39–46. IOS Press.

Samuel Rönnqvist, Jenna Kanerva, Tapio Salakoski, and Filip Ginter. 2019. Is multilingual BERT fluent in language generation? In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 29–36, Turku, Finland. Linköping University Electronic Press.

Antonio Šajatović, Maja Buljan, Jan Šnajder, and Bojana Dalbelo Bašić. 2019. Evaluating automatic term extraction methods on individual documents. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 149–154, Florence, Italy. Association for Computational Linguistics.

Vikash Singh. 2017. Replace or retrieve keywords in documents at scale. *CoRR*, abs/1711.00046.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Ziqi Zhang, Jie Gao, and Fabio Ciravegna. 2018a. Semre-rank: Improving automatic term extraction by incorporating semantic relatedness with personalised pagerank. *ACM Trans. Knowl. Discov. Data*, 12(5).

Ziqi Zhang, Jose Iria, Christopher Brewster, and Fabio Ciravegna. 2008. A comparative evaluation of term recognition algorithms. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Ziqi Zhang, Johann Petrak, and Diana Maynard. 2018b. Adapted textrank for term extraction: A generic method of improving automatic term extraction algorithms. *Procedia Computer Science*, 137:102–108.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating BERT into neural machine translation. In *International Conference on Learning Representations*.

A Appendix

For further comparability, we provide results of 15 prior term extraction methods provided by the ATR4S toolkit (Astrakhantsev, 2018). All methods provided by ATR4S are re-ranking methods based on a previous term candidate extraction process.

Table A1 shows the results of ATR4S on the ACTER heart-failure domain in English. While some methods achieve good precision, most methods show precision scores below our best models, even at only 100 terms extracted. Increasing the manually specified amount of k terms to extract results in a decrease of precision in favor of recall. The scores of the different methods level out towards the maximum of 2,000 terms extracted. The best F1 score is achieved by the DomainPertinence method at 2,000 terms extracted with an F1 score of 30.32%.

Table A2 shows the results of ATR4S on our ACLR2 test splits. One major drawback of prior methods is the required corpus size. The small test set in ACLR2 does not provide enough data for many of the statistical approaches or in fact the reranking to be effective at all after a certain amount of terms extracted. For the smaller Annotator 1 test set, we can observe virtually identical scores between all methods from 300 extracted terms onwards. For Annotator 2, this phenomena can be observed at 400 extracted terms. Best overall results are an F1 score of 21.83% for Weirdness at 200 terms extracted on the Annotator 1 test set and and F1 Score of 18.28% for Weirdness, PU and DomainPertinence at 300 terms extracted on the Annotator 2 test set. In comparison, our best models achieve an F1 score of over 75% for both Annotators.

ACTER: heart-failure EN (ATR4S)

Method	**Top 100**			**Top 500**			**Top 1000**			**Top 2000**		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
AvgTermFrequency	55.0	2.13	4.1	41.4	8.01	13.42	35.2	13.62	19.64	30.8	23.83	26.87
Basic	48.0	1.86	3.58	36.0	6.96	11.67	32.9	12.73	18.35	29.7	22.98	25.91
ComboBasic	48.0	1.86	3.58	36.0	6.96	11.67	31.9	12.34	17.8	29.7	22.98	25.91
CValue	54.0	2.09	4.02	39.6	7.66	12.84	33.7	13.04	18.8	32.1	24.84	28.0
DomainPertinence	58.0	2.24	4.32	46.8	9.05	15.17	35.8	13.85	19.97	**34.75**	**26.89**	**30.32**
KeyConceptRelatedness	**81.0**	**3.13**	**6.03**	**59.4**	**11.49**	**19.25**	**42.5**	**16.44**	**23.71**	31.1	24.06	27.13
LinkProbability	**83.0**	**3.21**	**6.18**	**73.0**	**14.12**	**23.66**	**52.8**	**20.43**	**29.46**	31.85	24.64	27.79
NovelTopicModel	49.0	1.9	3.65	39.8	7.7	12.9	34.9	13.5	19.47	29.95	23.17	26.13
PostRankDC	31.0	1.2	2.31	37.6	7.27	12.19	35.0	13.54	19.53	30.3	23.44	26.43
PU	61.0	2.36	4.54	46.2	8.94	14.98	38.3	14.82	21.37	**34.65**	**26.81**	**30.23**
Relevance	52.0	2.01	3.87	47.0	9.09	15.24	37.6	14.55	20.98	34.45	26.65	30.05
ResidualIDF	54.0	2.09	4.02	41.4	8.01	13.42	32.6	12.61	18.19	31.0	23.98	27.04
TotalTFIDF	31.0	1.2	2.31	39.8	7.7	12.9	36.7	14.2	20.47	31.05	24.02	27.09
Voting	**62.0**	**2.4**	**4.62**	**57.0**	**11.03**	**18.48**	**48.7**	**18.84**	**27.17**	**34.5**	**26.69**	**30.1**
Weirdness	31.0	1.2	2.31	38.0	7.35	12.32	37.4	14.47	20.86	31.65	24.49	27.61

Table A1: Precision, recall and F1 @ top *k* terms extracted of prior methods (ATR4S) on the English heart-failure domain texts of ACTER. Best three results per top *k* marked in bold.

ACL RD-TEC 2.0: Annotator 1 (ATR4S)

Method	**Top 100**			**Top 200**			**Top 300**			**Top 400**		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
AvgTermFrequency	36.0	7.53	12.46	29.0	12.13	17.11	28.73	16.53	20.98	28.73	16.53	20.98
Basic	40.0	8.37	13.84	36.0	15.06	21.24	28.73	16.53	20.98	28.73	16.53	20.98
ComboBasic	40.0	8.37	13.84	36.0	15.06	21.24	28.73	16.53	20.98	28.73	16.53	20.98
CValue	**47.0**	**9.83**	**16.26**	34.5	14.44	20.35	28.73	16.53	20.98	28.73	16.53	20.98
DomainPertinence	46.0	9.62	15.92	35.5	14.85	20.94	28.73	16.53	20.98	28.73	16.53	20.98
KeyConceptRelatedness	38.0	7.95	13.15	28.0	11.72	16.52	28.73	16.53	20.98	28.73	16.53	20.98
LinkProbability	40.0	8.37	13.84	29.0	12.13	17.11	28.73	16.53	20.98	28.73	16.53	20.98
NovelTopicModel	42.0	8.79	14.53	34.0	14.23	20.06	28.73	16.53	20.98	28.73	16.53	20.98
PostRankDC	43.0	9.0	14.88	36.0	15.06	21.24	28.73	16.53	20.98	28.73	16.53	20.98
PU	43.0	9.0	14.88	33.5	14.02	19.76	28.73	16.53	20.98	28.73	16.53	20.98
Relevance	46.0	9.62	15.92	35.5	14.85	20.94	28.73	16.53	20.98	28.73	16.53	20.98
ResidualIDF	36.0	7.53	12.46	29.0	12.13	17.11	28.73	16.53	20.98	28.73	16.53	20.98
TotalTFIDF	23.0	4.81	7.96	23.5	9.83	13.86	28.73	16.53	20.98	28.73	16.53	20.98
Voting	**47.0**	**9.83**	**16.26**	32.0	13.39	18.88	28.73	16.53	20.98	28.73	16.53	20.98
Weirdness	38.0	7.95	13.15	**37.0**	**15.48**	**21.83**	28.73	16.53	20.98	28.73	16.53	20.98

ACL RD-TEC 2.0: Annotator 2 (ATR4S)

Method	**Top 100**			**Top 200**			**Top 300**			**Top 400**		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
AvgTermFrequency	36.0	5.34	9.3	28.0	8.31	12.81	26.33	11.72	16.22	27.44	13.35	17.96
Basic	40.0	5.93	10.34	34.5	10.24	15.79	27.67	12.31	17.04	27.44	13.35	17.96
ComboBasic	40.0	5.93	10.34	34.5	10.24	15.79	27.67	12.31	17.04	27.44	13.35	17.96
CValue	44.0	6.53	11.37	34.0	10.09	15.56	29.0	12.91	17.86	27.44	13.35	17.96
DomainPertinence	38.0	5.64	9.82	**37.0**	**10.98**	**16.93**	**29.67**	**13.2**	**18.28**	27.44	13.35	17.96
KeyConceptRelatedness	35.0	5.19	9.04	31.0	9.2	14.19	27.33	12.17	16.84	27.44	13.35	17.96
LinkProbability	39.0	5.79	10.08	30.0	8.9	13.73	27.67	12.31	17.04	27.44	13.35	17.96
NovelTopicModel	39.0	5.79	10.08	34.5	10.24	15.79	27.67	12.31	17.04	27.44	13.35	17.96
PostRankDC	42.0	6.23	10.85	34.5	10.24	15.79	27.67	12.31	17.04	27.44	13.35	17.96
PU	42.0	6.23	10.85	34.0	10.09	15.56	28.0	12.46	17.25	27.44	13.35	17.96
Relevance	38.0	5.64	9.82	37.0	10.98	16.93	**29.67**	**13.2**	**18.28**	27.44	13.35	17.96
ResidualIDF	36.0	5.34	9.3	28.0	8.31	12.81	26.67	11.87	16.43	27.44	13.35	17.96
TotalTFIDF	19.0	2.82	4.91	24.0	7.12	10.98	26.33	11.72	16.22	27.44	13.35	17.96
Voting	**46.0**	**6.82**	**11.89**	36.0	10.68	16.48	29.0	12.91	17.86	27.44	13.35	17.96
Weirdness	38.0	5.64	9.82	29.5	8.75	13.5	**29.67**	**13.2**	**18.28**	27.44	13.35	17.96

Table A2: Precision, recall and F1 @ top *k* terms extracted of prior methods (ATR4S) on test split of ACLR2. Best result per top *k* marked in bold.

ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language

Oyvind Tafjord, Bhavana Dalvi Mishra, Peter Clark
Allen Institute for AI, Seattle, WA
{oyvindt,bhavanad,peterc}@allenai.org

Abstract

Transformers have been shown to emulate logical deduction over natural language theories (logical rules expressed in natural language), reliably assigning true/false labels to candidate implications. However, their ability to *generate* implications of a theory has not yet been demonstrated, and methods for reconstructing proofs of answers are imperfect. In this work we show that a generative model, called ProofWriter, can reliably generate both implications of a theory and the natural language proofs that support them. In particular, iterating a 1-step implication generator results in proofs that are highly reliable, and represent actual model decisions (rather than post-hoc rationalizations). On the RuleTaker dataset, the accuracy of ProofWriter's proofs exceed previous methods by +9% absolute, and in a way that generalizes to proof depths unseen in training and on out-of-domain problems. We also show that generative techniques can perform a type of abduction with high precision: Given a theory and an unprovable conclusion, identify a missing fact that allows the conclusion to be proved, along with a proof. These results significantly improve the viability of neural methods for systematically reasoning over natural language.[1]

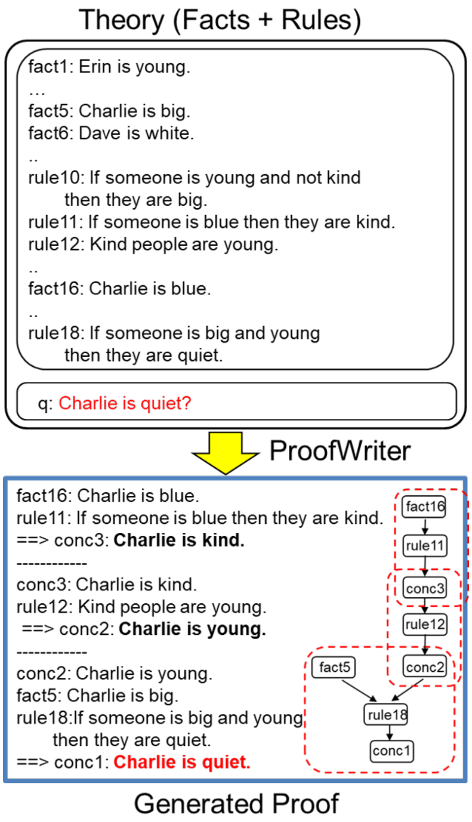

fact1: Erin is young.
...
fact5: Charlie is big.
fact6: Dave is white.
..
rule10: If someone is young and not kind then they are big.
rule11: If someone is blue then they are kind.
rule12: Kind people are young.
..
fact16: Charlie is blue.
..
rule18: If someone is big and young then they are quiet.

q: Charlie is quiet?

fact16: Charlie is blue.
rule11: If someone is blue then they are kind.
==> conc3: **Charlie is kind.**
----
conc3: Charlie is kind.
rule12: Kind people are young.
==> conc2: **Charlie is young.**
----
conc2: Charlie is young.
fact5: Charlie is big.
rule18: If someone is big and young then they are quiet.
==> conc1: **Charlie is quiet.**

Figure 1: Given facts, rules, and a question all expressed in natural language, ProofWriter answers the question and generates a proof of the answer.

1 Introduction

A fundamental goal for AI, dating back to its earliest years, is automated reasoning: the ability to draw valid conclusions from explicitly provided knowledge (McCarthy, 1959). However, approaches relying on expressing knowledge in a formal representation language have sometimes proved challenging (Musen and Van der Lei, 1988). Recent work on RuleTaker (Clark et al., 2020) demonstrated a modern approach to this goal, in which transformers emulate deductive reasoning over statements expressed in *natural language*, by reliably assigning true/false labels to candidate implications. However, simply assigning true/false labels is limiting. For practical purposes, systems should also generate proofs of those labels, so that their conclusions can be verified and a human-understandable rationale be produced.

Recent work on PRover, by Saha et al. (2020), provided first results towards this goal, assembling proofs by first classifying which facts, rules, and connections should be in the proof tree then using an Integer Linear Programming (ILP) module to enforce consistency constraints. However, the gen-

---

[1] Datasets available at https://allenai.org/data/proofwriter

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3621–3634
August 1–6, 2021.

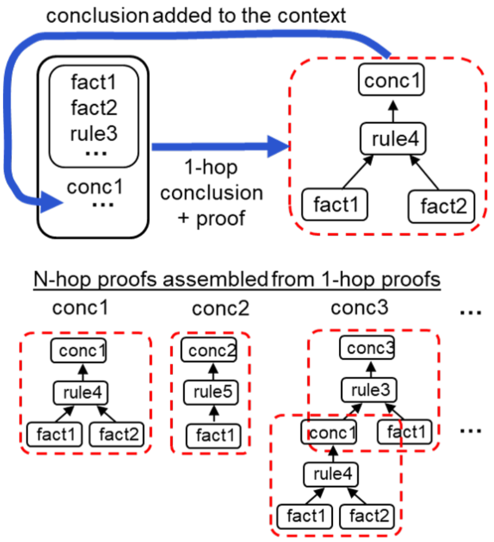

Figure 2: ProofWriter iteratively generates 1-step implications and their proofs, and adds implications back into into the context for deeper reasoning. The stepwise proof fragments are assembled into full proofs of N-hop conclusions.

erated proofs were imperfect, and there were no guarantees that the model "believed" the proofs that it was reciting, i.e., that its QA module would agree with the steps shown in the proof. In this paper, we adopt a different approach, based on generation rather than classification. Our system, ProofWriter, generates proofs such as that shown in Figure 1 by iteratively generating 1-hop inferences and their (simple) proofs, adding implications back into the context for deeper reasoning, and assembling more complex proofs from the 1-hop fragments (Figure 2). As the accuracy of 1-hop inference is highly reliable, the accuracy of deeper inference and their proofs is also high. This results in proofs that substantially exceed the earlier method's accuracy, and also reflect the model's internal decisions, rather than a post-hoc rationalization (i.e., is a "faithful" proof (Subramanian et al., 2020)).

The generative approach also affords two other new capabilities. First, ProofWriter generates implications that logically follow from a NL (natural language) theory, allowing enumeration of consequences (rather than only assigning truth values to pre-conjectured hypotheses). Second, we demonstrate (a constrained form of) abduction: Given a theory and an unprovable conclusion, identify a missing fact (if any) that allows the conclusion to be proved when added to the theory, plus its proof.

We evaluate our work on a collection of natural language reasoning datasets, including the RuleTaker datasets as well as several new variants. We achieve state-of-the-art results in proof generation, and strong new baselines for implication enumeration and abduction over natural language theories. Our contributions are thus:

1. A new method for proof generation for logical reasoning over natural language, that obtains state-of-the-art results and is faithful to the model's internal decisions.
2. A method and baseline results for *generating* logical implications of statements in NL.
3. A method and baseline results for performing abduction over natural language statements.
4. New datasets to promote further research.

These results significantly improve the viability of neural methods for formal reasoning over language.

2 Related Work

Our work builds on the RuleTaker line of research, in which transformers learn to emulate a deductive reasoning *algorithm* (Clark et al., 2020). Unlike other approaches to reasoning such as parsing to a formal language (Kamath and Das, 2019), implementing a reasoning algorithm with neural components (Weber et al., 2019; Rocktäschel and Riedel, 2017), or SAT solving (Selsam et al., 2019), these transformers emulate reasoning over language directly, bypassing a formal representation.

PRover (Saha et al., 2020), mentioned earlier, was the first system to also produce proofs in this context, although its post hoc approach meant that proofs did not necessarily represent the actual model decisions. Gontier et al. (2020) also explored the generation of answers and proofs, but in the context of rule *induction* with ($\approx$ 10) fixed rules to induce. In contrast, ProofWriter generates proofs from explicit NL rules (which may differ for each problem). Similarly, formal theorem proving has explored proving mathematical theorems from fixed, fundamental axioms, e.g., (Polu and Sutskever, 2020; Wang and Deng, 2020), while ProofWriter performs inference with differing sets of rules expressed in natural language.

Our work is also distinct from the large body of work on rationales and explanation. Work on rationales aims to identify sentences (or phrases) that caused a model to make a particular decision, but without an explanation of *why* that rationale led to the answer (the model's reasoning is opaque), e.g., (DeYoung et al., 2019; Narang et al., 2020). Similarly, work on explanations has sought to gen-

Proof (and Answer)	$CQ \rightarrow AP$	Given theory C and hypothesis fact Q, determine Q's truth A and proof P (if any)
Enumeration	$C \rightarrow I_1, ..., I_n$	Given C, generate all implications I_i that logically follow.
Abduction	$CQ \rightarrow f_m$	Given C and an unprovable fact Q, identify a new fact f_m that, when added to C, would make Q true.

Table 1: The three tasks that ProofWriter performs.

erate human-style justifications, which again are typically supporting evidence rather than a fully-formed line of reasoning, and without explicit reasoning rules (Camburu et al., 2018; Jhamtani and Clark, 2020; Inoue et al., 2020). In contrast, ProofWriter produces a deductive chain of reasoning from what is known to what is concluded, using a transformer retrained to reason systematically.

3 Approach

3.1 Definitions

Let:

- C be a *theory*, a set of English sentences C consisting of facts F and rules R, each expressing a logical fact or rule in English. (We also refer to C as the *context*).
- Q be a *question*, a hypothesis fact in English whose truth is to be determined based solely on the information in C.
- A be an *answer*, where $A \in \{True, False\}$ (if reasoning using a closed-world assumption) or $A \in \{True, False, Unknown\}$ (open-world assumption).
- P be a *proof*, described shortly.
- I be an *implication*, a fact that logically follows from C.

We define three tasks (also see Table 1):

1. **proof (inc. QA):** $CQ \rightarrow AP$: Given C and hypothesis fact Q, what is the truth A and proof P (if any) of Q?
2. **enumeration:** $C \rightarrow I_1, ..., I_n$: Which I_i follow from C?
3. **abduction**(restricted form) $CQ \rightarrow f_m$: Which extra fact f_m will make Q true given C?

We reuse (and add to) the RuleTaker datasets for our work, which include all five elements above. An example of a RuleTaker theory (facts and rules), a query, and a proof generated by ProofWriter are shown in Figure 1. Facts and rules are English statements, and implications are English statements that logically follow from those facts and rules. The original datasets were generated from synthetic logic programs and their implications, using natural language patterns to produce the English forms.

3.2 Semantics

Following prior work, we adopt the semantics of Datalog (Ceri et al., 1989): A fact is true if it is either known (i.e., explicitly stated in the context C), or (recursively) is the conclusion of a rule whose conditions are true (is "supported"). For handling negation, we use two alternative Datalog semantics: The first, following prior work, makes the closed-world assumption (CWA) and uses negation as failure (NAF), so that any fact not provable is assumed false. Under this semantics, negated facts and negative rule conclusions are not allowed (redundant under the CWA). The second makes an open-world assumption (OWA), and does allow negative facts and rule conclusions. Under this semantics, a third truth value *Unknown* is also possible.

3.3 Proof Representation

We define a proof P of a fact f_q as a directed acyclic graph (N, E) with nodes $n \in N$ and (directed, untyped) edges $e \in E$. Each node in P is either a fact f (a ground literal) or a rule r (a logical implication), expressed in English. Each edge in the proof either connects a fact to a rule, denoting that the fact helps satisfy the rule's condition, or connects a rule to a fact, denoting that the fact follows from the instantiated rule. Thus nodes in any branch of the proof will alternate between facts and rules. Note this definition differs from (and is richer than) that in PRover, where intermediate conclusions were not part of the proof.

Facts in the proof are one of three types: known facts $f_i \in F$, negated facts f_{naf} that cannot be proven (false under negation-as-failure (NAF)), and facts f_{conc} that are the conclusions of rules. f_i and f_{naf} are leaf nodes of the proof, while the f_{conc} are intermediate nodes within the proof. Note that f_{naf} and f_{conc} are by definition not in F. Example proofs are shown in Figures 1 and 3.

3.4 Proof Encoding

As we wish to generate proofs, we need to encode P as a linear structure that can be output by a generative model. Facts and rules in the context are explicitly labeled with identifiers (fact1, ..., rule1,

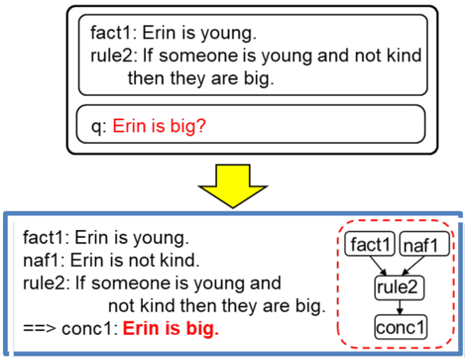

Figure 3: An example proof that includes a negated (negation-as-failure) fact.

...) that the proof can refer to, see Figures 1 and 3.[2] Then, in the linear proof, rule nodes are denoted by their identifier (rule1, ...), while fact nodes are denoted by three types of identifiers: fact1, fact2, ... for facts in the context; naf1, naf2, ... for facts *not* in the context and assumed false; and conc1, conc2, ... for facts concluded by rules. To decode the naf* and conc* identifiers (which by definition are not in the context), an additional sequence of the form "with conc1: *sentence1*. conc2: *sentence2*. ..." is appended to the proof.

To linearize the proof in a format convenient for a generative model, we conjoin rules and their conclusions using a "%" symbol, express conjunctive rule conditions with a "&" symbol, and use "#" to denote the inverse implication ("←"). We then express the tree using Polish notation. E.g., the proof tree "((fact1 & fact2) → rule1 → conc1)" (i.e., fact1 and fact2 satisfy rule1, concluding conc1) would be expressed "# rule1%conc1 & fact1 fact2". Thus the 3-step proof from Figure 1 is encoded:

> # rule18%conc1 & fact5 # rule12%conc2 # rule11%conc3 fact16 ; with conc1: Charlie is quiet. ; conc2: Charlie is young. ; conc3: Charlie is kind.

If the question is a known fact, the "depth 0 proof" is simply the fact itself (e.g., fact1). If no proof exists, the symbol "None" is used.

3.5 Models

The ProofWriter models are built on top of the text-to-text pretrained T5 transformer (Raffel et al., 2020) (T5-11B). We use different textual prompts for the different tasks. For the task of generating an answer and a proof, the input to the model is of the form: " \$question\$ = *question* ; \$context\$ = *theory-sentences*", for example: "\$question\$ = Erin is big. ; \$context\$ = sent1: Erin is young. sent2: If ..." The output is of the form: "\$answer\$ = *True/False/Unknown* ; \$proof\$ = *proof* ;", where *proof* is encoded as described in Section 3.4. For training instances where multiple outputs are valid, we select a single one at random (for multiple proofs, we select among the shortest proofs). Appendix D lists the hyperperameters and gives input/output examples for each task.

3.6 Task 1: Proof Generation

We evaluate two methods of proof generation:

All-At-Once: We train a model to generate the full proof and answer in one go (theory + question → answer + proof).

Iterative: We first train a model to generate a single 1-step implication (theory → implication + 1-step-proof), where the implication follows from a single rule application. Then at test time, we apply this model iteratively, adding each implication to the theory and repeating until no more implications can be found (i.e., exhaustive forward-chaining). The proof for any given implication can then be assembled from the 1-step-proof fragments (Figure 2).

3.6.1 All-At-Once ProofWriter ("All")

The All-At-Once model is trained directly on $CQ \rightarrow AP$ examples in the datasets (P = "None" if there is no proof of Q). Section 3.5 describes the i/o format, and Appendix D.1 shows an example.

3.6.2 Iterative ProofWriter ("Iter")

Training: To train the Iterative model, for each theory C in the training data, we create an augmented set of training examples with one sequence of iteratively inferred facts in turn, each using C plus the previously inferred facts. For example, if theory C_1 implies I_1, I_2, and I_3, then we create four training examples $C_1 \rightarrow I_1$, $C_1 \cup \{I_1\} \rightarrow I_2$, $C_1 \cup \{I_1, I_2\} \rightarrow I_3$, and $C_1 \cup \{I_1, I_2, I_3\} \rightarrow$ "None". The order of adding the I_i is random but constrained such that if a later implication depends on an earlier one, the earlier one must be inferred first. For example, if the proof of I_3 depends on I_2 (determined by inspecting the gold proofs), I_2 must be in the context before I_3 is inferred. This ensures that all example inferences are depth 1 (i.e., a single rule application). An example input/output for one step is shown in Appendix D.2.

---

[2] In practice we name these sent1, sent2, ... without a fact/rule distinction, but for expository purposes it is helpful to use different identifiers.

Testing: To answer and provide the proof for a particular question/implication, the model generates all implications and their proofs by iteratively applying the model until no more implications (the implication "None") is generated. It then looks for the question among them. If found, the answer is *True* with the proof given. The model also looks for the negation of the question[3] and its proof. If found, the answer is *False* with the proof given. Otherwise, there is no proof (proof = "None") and the answer is *False* (for positive questions, CWA), *True* (for negative questions, CWA), or *Unknown* (any question, OWA).

3.7 Task 2: Implication Enumeration

A second desirable reasoning skill is *enumerating* implications of a theory (rather than just assign *True/False* to a hypothesis). This capability is important for practical application of the technology. In fact, the Iterative ProofWriter already does this by design, a substantial advantage. To evaluate this (later), we compare this with an "all at once" strategy of generating *all* implications as a single output string, analogous to the All-At-Once strategy for generating the full proof as a single string. For training this All-At-Once enumerator, and testing both, we gather the list of *all* implications I_i of each theory C in the train/test data. Each train/test example is of then of the form: given C, predict all the I_i. An example input/output is in Appendix D.3.

3.8 Task 3: Abduction (Single Fact)

A third desirable reasoning skill is *abduction* over natural language theories, again made possible by generative models. Abduction has previously been studied extensively in formal logic, e.g., (Konolige, 1997), and in NLP, e.g., (Hobbs et al., 1993; Bhagavatula et al., 2020). Here we evaluate whether a generative approach can combine logic and NLP, performing logical abduction over natural language knowledge. We do this for a restricted form of abduction, namely single-fact abduction: Given a theory C and a possible implication Q not provable from C, identify a new fact f_m (other than the trivial Q itself) such that $C \cup \{f_m\}$ implies Q.

We restrict this task to the OWA (open-world) setting where questions can naturally have unknown truth values. To train and test an abductive model over our datasets, we create an abductive version as follows: For each theory C in the train/test data, for each unprovable fact Q, identify all alternative "missing facts" $factM$ that, when added to C, make Q True. To do this, recall that each NL theory was originally generated from a formal one C_{formal} in a formal representation language (Datalog). We first exhaustively enumerate all possible Q_{formal} and $factM_{formal}$ in the formal language (this is feasible as the space of predicates and individuals is small), then use a theorem prover to test if $C_{formal} \cup \{factM_{formal}\}$ implies Q_{formal} for all pairs $(factM_{formal}, Q_{formal})$. For each success, we generate the NL equivalents Q and $factM$ using simple NL generation templates. We then collect the alternative $factM$s for each Q. The abduction task is then, given C and Q, identify the set of all alternative $factM$s, i.e.:

$$C, Q \rightarrow factM_1, ..., factM_i$$

If there is no single $factM$ that can be added to make Q true, then the symbol "None" is output.

4 Datasets

We now evaluate ProofWriter on these three tasks. We use the original RuleTaker D* datasets (Clark et al., 2020), plus we create two new variants: The first (CWA) is similar to the original except it fixes some minor inconsistencies concerning negation (details in Appendix A.2). The second (OWA) is also similar to the original, except reasoning uses an open-world assumption.

We denote these as D*(orig), D*(CWA), and D*(OWA). Each example in each dataset contains a theory C, a question Q, the answer A (*True/False/Unknown*), and all possible proofs $P_1, ..., P_n$ for that answer (if provable).[4] Each theory is also accompanied with all possible proofs of all possible implications, as auxiliary annotations.

The D* datasets comprise five datasets, named **D0, D1, D2, D3, D5**, each containing 100k questions. In each dataset, theories and questions are expressed in templated English (e.g., Figure 1), questions can be positive or negated facts (e.g., "Charlie is not quiet?"), and answers are equally divided into *True/False* (and *Unknown*, for the OWA versions). Each dataset contains questions whose answers require reasoning up to depths D (D = 0, 1, 2, 3, 5). Thus, for example, all questions in D0

---

[3] To negate a question, a model can be trained for this straightforward task. Here, as our question language is simple, a simple regex to add/remove a "not" suffices.

[4] The domain is small enough that all proofs can be enumerated. However, there still can be a large number, e.g., some D5 questions have over 3000 possible proofs.

		Answer		**Proof**	
Depth	# qns	PRover	ProofWriter	PRover	ProofWriter
0	6299	100	100	98.4	99.6
1	4434	99.0	99.1	93.1	98.7
2	2915	98.8	98.6	84.8	97.3
3	2396	99.1	98.5	80.5	94.4
4	2134	98.8	98.7	72.4	91.0
5	2003	99.3	99.3	65.1	86.4
All	20192	99.3	99.2	87.1	96.2

Table 2: [Task 1: Proof Generation] Systems trained and tested on D5(orig), showing the breakdown by depth of proof required to answer each question. ProofWriter generates significantly more correct proofs for all depths, achieving a new SOTA on this task.

		Answer			**Proof**		
			ProofWriter			ProofWriter	
Test	# qns	PRover	All	Iter	PRover	All	Iter
Birds1	40	95.0	100	95.0	92.5	100	95.0
Birds2	40	95.0	100	95.0	95.0	100	95.0
Elec1	162	100	96.9	100	95.1	96.9	100
Elec2	180	100	98.9	100	91.7	98.9	100
Elec3	624	89.7	92.0	95.5	71.8	92.0	95.5
Elec4	4224	84.8	83.3	97.1	80.6	82.0	97.1
All	5270	86.5	85.5	97.0	80.5	84.5	97.0

Table 3: [Task 1: Proof Generation] Training on D5, test on Birds-Electricity. Both ProofWriter versions ("**All**" for All-At-Once, "**Iter**" for Iterative) outperform PRover overall in both answer and proof correctness. The Iterative model is also significantly more robust.

are lookup questions, requiring no inference. Each dataset is split 70/10/20 into train/dev/test.

To test generalization, we also use two other datasets from the original RuleTaker work:

Birds-Electricity: These 6 test-only datasets use small, real-world theories written by hand (one per dataset) to test out-of-distribution model performance. Details are in Appendix A.3.

ParaRules: This dataset contains 40k questions against 2k theories expressed in paraphrased natural language, obtained through crowdsourcing. This dataset tests transfer to more natural expressions of knowledge. Details are in Appendix A.4.

5 Experiments and Results

5.1 Task 1: Proof Generation (Comparison with Prior Work)

First, we compare ProofWriter's ability to generate proofs with PRover, the current state-of-the-art. We evaluate both answer accuracy and proof correctness. For proof correctness, for a fair comparison, we ignore the intermediate conclusion nodes (which PRover does not generate). We then use the same strict scoring metric as in PRover (called FA or Full Accuracy in the PRover paper): the proof graph must *exactly* match a gold proof (i.e., be perfectly correct); otherwise, the proof scores 0.

5.1.1 Generating Answers and Proofs

We use the same IID (independent, identically distributed) data used for PRover (train/test on dataset D5(orig)). The results are in Table 2, showing accuracies for questions requiring increasingly deeper depths of reasoning to answer. The ProofWriter's results are for the All-At-Once model. (The Iterative model scores are almost identical, see later Table 4.) While answer accuracy is almost perfect for both systems, ProofWriter generates **substantially more correct proofs** (last line, +9% absolute), and without the complexity of PRover's heuristic assembly of proof graphs using ILP.

5.1.2 Performance on OOD Rulesets

We compared ProofWriter's and PRover's ability to generalize to the hand-authored Birds-Electricity rulesets, zero shot. These rulesets are out-of-domain (OOD), as their English is not templated and is stylistically different to the training data. We compare the PRover and All-At-Once ("All") ProofWriter models trained on D5, plus the Iterative ProofWriter ("Iter") trained on D0-D3 theories. The models do not see any Birds-Electricity examples during training. The results in Table 3 show that ProofWriter's proof generation transfers well zero-shot to these hand-authored datasets, with 84.5% proof correctness for All-At-Once, and 97% for the Iterative ProofWriter, indicating better out-of-domain generalization for the Iterative version. Both ProofWriter models significantly outperform PRover (80.5%).

We also find ProofWriter obtains more correct proofs (+3%) than PRover on the ParaRules dataset. Details are in Appendix B.1.

5.2 Task 1: Proof Generation (All-At-Once vs. Iterative)

Second, we compare our two approaches to proof generation, All-At-Once vs. Iterative, in more detail. We show that although they have almost identical performance for proofs with depths seen in training, **the Iterative model generalizes better to proofs of longer depths** than seen in training. For these comparisons, we use the new D*(CWA) datasets (which fix some minor errors in D*(orig)), and also the D*(OWA) datasets to explore performance in an open-world setting.

	Answer				**Proof**			
	CWA		OWA		CWA		OWA	
Depth	All	Iter	All	Iter	All	Iter	All	Iter
N/A	99.0	99.7	99.4	99.9	99.0	99.7	99.4	99.9
0	100	100	100	100	100	100	100	100
1	99.9	99.8	100	99.3	99.6	95.4	99.7	97.8
2	99.9	99.5	99.9	99.7	98.3	91.7	98.6	97.3
3	100	99.7	100	99.2	95.8	90.4	96.9	97.1
4	100	99.7	99.9	99.1	93.1	88.9	94.8	96.5
5	99.9	98.9	100	98.8	89.3	87.8	91.4	86.4
All	99.6	99.7	99.7	99.6	97.2	95.4	98.0	97.6

Table 4: [Task 1] Comparison of All-At-Once ("**All**") vs. Iterative ("**Iter**") ProofWriter models, trained on D5 and D0-D3 respectively, and tested on D5.

5.2.1 Comparison (IID Test Set)

We train the All-At-Once model on D5 (train), and the Iterative model using the method described in Section 3.6.2, using the (∼ 5k) theories from D3 (train) plus ∼ 20% of the D0-D2 (train) theories.[5] We then test both models on D5 (test). We measure both answer and proof accuracies, and also break down the results by proof depth (using "N/A" as the proof depth for questions that are not provable). The D5 test set has 2k questions at each proof depth, plus 8k unprovable questions (proof = "None", depth = "N/A').[6]

The results are shown in Table 4, and show that **both ProofWriter versions have similar, high proof correctness (95%+)** on the test set, even though some proofs are highly complex.

5.2.2 Generalization to Unseen Depths

We also wish to see how well the models can generate proofs at depths unseen during training. To do this, we train an All-At-Once model on D3, and use the same Iterative model as earlier (trained on iterative examples from theories up to depth 3). We test on D5. As D5 contains problems at greater depths than those seen during training, we can observe the models' ability to generalize. We compare with both the CWA and OWA versions of our datasets.

The results are shown in Table 5. As can be seen, the All-At-Once model has quite poor generalization for generating longer proofs than seen in training, while **the Iterative model is more robust** (red box).

	Answer				**Proof**			
	CWA		OWA		CWA		OWA	
Depth	All	Iter	All	Iter	All	Iter	All	Iter
N/A	99.6	99.7	99.4	99.9	99.6	99.7	99.4	99.9
0	100	100	100	100	100	100	100	100
1	99.9	99.8	99.9	99.3	99.7	95.4	99.8	97.8
2	99.4	99.5	99.8	99.7	98.2	91.7	98.8	97.3
3	99.2	99.7	99.8	99.2	93.4	90.4	94.5	97.1
4	95.4	99.7	99.3	99.1	69.9	88.9	71.4	96.5
5	72.9	98.9	93.7	98.8	27.4	87.8	35.1	86.4
All	96.6	99.7	99.0	99.6	88.9	95.4	90.2	97.6

Table 5: [Task 1] Comparison of the All-At-Once vs. Iterative ProofWriter models, trained on D3 and tested on D5. While scores are mostly similar throughout, the iterative model generalizes substantially better to generate proofs of depths unseen during training (red box).

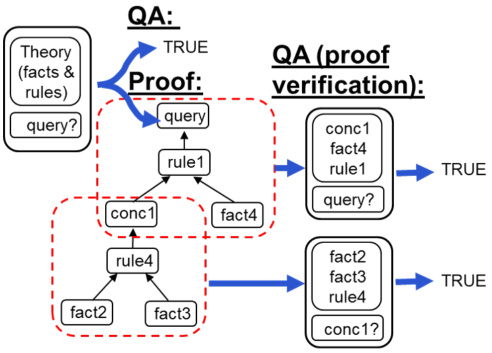

Figure 4: [Task 1] All-At-Once proofs can be verified by checking each step as a separate QA query.

5.3 Verifying All-At-Once Proofs

Proofs from the Iterative ProofWriter have an additional desirable property: each proof step is one that the model explicitly took during the iteration, i.e., the model "believes" the step. In contrast, the All-At-Once proofs are a post hoc generated string of symbols, and may not reflect steps that ProofWriter would actually make. However, because proofs include intermediate conclusions, we can alleviate this concern by *verifying* individual steps in the All-At-Once proofs. For example, if a generated proof step states that fact2 + fact3 + rule4 implies conc1, we can simply ask ProofWriter in QA mode if this is true (Figure 4). Given the almost perfect performance for such simple depth 1 questions in QA mode (with no distractor facts or rules), the ability to verify a correct proof corresponds to the accuracy of correctly generating the correct intermediate conclusions conc* in the first place. (Note that an unverified proof is not necessarily wrong, rather cannot be verified as right). OWA proofs can be fully verified in this way. For CWA theories with NAFs, the verification is only partial as NAFs are presumed negative statements

---

[5] We include D0-D2 theories to have more examples of theories with fewer conclusions. The derivative iterative training data is included in our dataset release.

[6] Note this breakdown is slightly different from the one in Table 2 where the depth used the original RuleTaker annotations which included a depth for questions without proofs, based on the deepest proof search that fails. We retained that convention in Table 2 for best comparison with PRover.

	Verified Proofs			
	CWA		OWA	
	Train on:		Train on:	
Depth	D3	D5	D3	D5
N/A	99.6	99.0	99.4	99.4
0	100	100	100	100
1	99.8	99.6	99.7	99.7
2	98.2	98.3	98.6	98.6
3	93.2	95.8	94.3	96.8
4	66.7	92.9	66.1	94.6
5	13.8	89.3	16.4	90.8
All	87.2	97.2	87.9	97.9

Table 6: [Task 1: Proof Generation] The All-At-Once model's ability to verify its proofs. For proofs within depths seen during training, almost all correct proofs (Tables 5 and 4, columns 5 and 7) can be verified. However, for proofs at unseen depths, the proportion that can be verified drops rapidly (trained on D3, test on depths 4,5). In contrast, Iterative ProofWriter's proofs are always verified, by definition of its algorithm.

which require the full theory to verify.

We measured the percentage of correct, verified proofs, shown in Table 6. Provided proofs are within the depths seen during training, almost all correct proofs can be verified. However, at depths deeper than seen at training, the proportion that can be verified drops rapidly. In contrast, the Iterative ProofWriter's proofs are always verified, as by definition they are assembled from single step inferences that the model actually took.

5.4 Task 2: Implication Enumeration

Third, we evaluate ProofWriter's performance on a new task, namely *enumerating* implications of a theory (rather than just assign *True/False* to a hypothesis). We compare the All-At-Once and Iterative strategies as described in Section 3.7.

To train All-At-Once, and test both, we created an enumerative dataset of $C \rightarrow \{I_1, ..., I_n\}$ examples (Section 3.7). For this we sample theories C in the D0-D3 datasets and gather the list of all implications I_i for each theory C. We call this enumerative dataset **D3+Enum**. We similarly create a **D5-Enum** dataset from theories in (only) D5 to test OOD conclusion generation. We create CWA and OWA versions of both.

We train All-At-Once on D3-Enum (train), then test both models on D3-Enum (test) and D5-Enum (test). For metrics, we measure F1 scores by comparing the individual predicted implications with the gold I_i, as well as the exact-match correctness of the predicted *set* of implications $\{I_1, ..., I_n\}$ (one point if the set *exactly* matches the gold, bar ordering, zero otherwise). The results are shown in Table 7, and show that the **Iterative ProofWriter is better at implication enumeration** than the simple All-At-Once strategy. In particular, the All-At-Once strategy struggles for problems at depths unseen in training (second row), although it does well on its own test set despite the complicated unordered output it has to generate (up to 16 different implications in D3, 21 in D5).

	F1				Accuracy			
Enum	CWA		OWA		CWA		OWA	
Test	All	Iter	All	Iter	All	Iter	All	Iter
D3+	98.9	99.8	99.4	99.6	92.5	98.8	95.5	99.0
D5	94.5	99.5	94.8	99.4	44.6	93.9	48.9	94.8

Table 7: [Task 2: Enumeration] Iterative ProofWriter is better at generating all implications than an All-At-Once strategy. (All-At-Once is trained on D3+Enum, Iterative ProofWriter is the same model as earlier.)

Test:	Count	F1	Acc
D3-Ab	7067	97.4	94.5
D5-Ab	7181	97.3	93.5

Table 8: [Task 3: Abduction] Given a theory C and an unprovable conclusion Q, predict all alternative facts that, when added to C, make Q provable.

5.5 Task 3: Abduction (Single Fact)

Fourth and finally, we evaluate performance on a second new task, namely *abduction* over natural language theories, again made possible by generative models. Analogous to implication enumeration, we create a derivative abductive dataset of $C, Q \rightarrow factM_1, ..., factM_i$ examples, where $C \cup \{factM_i\}$ results in Q becoming provable as described in Section 3.8. We create such D*-Ab datasets from the D*(OWA) datasets.

5.5.1 Results (IID)

We trained a model on D3-Ab (train), and then tested on both D3-Ab (test) and D5-Ab (test). We evaluate the results by comparing the predicted and gold $factM$s, measuring both F1 and "perfect match" Accuracy (1 when F1=1, 0 otherwise). The results are shown in Table 8, and indicate that the model performs well overall (85%+ scores). We also broke down the recall of $factM$s by proof depth required to prove Q given C and $factM$. This is shown in Table 9, indicating that it is harder to identify a $factM$ that completes a deeper proof. The similarity of D3-Ab and D5-Ab scores suggests that D5-Ab is not out-of-domain for this task: Although depths for *provable* D5 facts are deeper than D3, this task concerns *unprovable* facts, which may not be distributed differently to D3-Ab.

Gold Proof Depth	Test on D3-Ab		Test on D5-Ab	
	# Gold	Acc (recall)	# Gold	Acc (recall)
N/A	2155	97.73	2170	97.74
1	4813	98.46	4731	98.73
2	1719	96.22	1986	96.17
3	688	90.26	915	92.79
4	153	75.82	330	82.73
5	19	36.84	96	78.13

Table 9: [Task 3: Abduction] Recall of abduced facts by proof depth. The data suggests that it is harder to identify a $factM$ that completes a deeper proof.

5.5.2 Generalization to New Tasks

To assess out-of-domain generalization, we also evaluate how well the trained abductive model performs on an abductive version of the Birds-Electricity(OWA) theories, zero-shot (created using the same approach, Section 3.8). We find that ProofWriter has perfect zero-shot performance for the simple Birds rulebases, but progressively reduced performance for the Electricity theories as they get more complex (dropping to 64% F1, 62% Accuracy for one rulebase), indicating that the abductive task is only partly solved (Appendix B.2).

6 Discussion

6.1 All-At-Once vs. Iterative Strategies

While the All-At-Once approach to proof generation is simple, efficient, and effective, it does not generalize as well to proofs of greater depth than seen at training. In contrast, the Iterative approach is robust to generalization. Even though errors at each iteration accumulate, the reliability of 1-step inference is so high that such error accumulations remain small. The Iterative architecture, namely a simple model embedded in a recursive loop (rather than single seq2seq model), illustrates how transformers can be used in a "scale-invariant" way, i.e., performance is largely unchanged by the scale (here reasoning depth) of the problem. In addition, as proofs are built from actual inference steps taken by the model, they are by definition "faithful" to the model's inference steps, rather than being a post hoc rationalization.

However, there are also some drawbacks to the Iterative approach: First, it is inefficient and unguided, proving everything possible and only then looking for the answer and proof for a particular question. In fact, this is a limitation of unconstrained forward-chaining in general, hence established techniques for guiding forward-chaining could be applied, e.g., a best-first expansion strategy, or using a backward-chaining strategy instead (which would similarly need to be controlled). Second, as the theory grows by one fact per iteration, there is a risk of exceeding the transformer's input token limit (512 tokens by default), hence limiting the size of theories that can be handled. For larger theories, a retrieval mechanism might be needed to manage the facts and rules available to the reasoner.

6.2 Abduction and Implicit Knowledge

Recently, LeapOfThought (Talmor et al., 2020) showed that RuleTaker-like models could be retrained to reason with a combination of explicit and implicit knowledge, rather than requiring all rules to be stated explicitly (the implicit knowledge coming from the latent knowledge acquired during pretraining (Petroni et al., 2019)). Now, given an abductive capability such as the one we have presented, we have a mechanism for materializing the implicit knowledge used to answer a question, and hence generating the full proof of its answer: Given a LeapOfThought conclusion, first abduce the "missing" (implicit) fact(s) required for an explicit proof, then use ProofWriter to generate that proof. This is a significant step forward to help understand a model's decisions when both implicit and explicit knowledge has been used.

7 Summary and Conclusion

While it is remarkable that transformers can learn to systematically reason over language, such methods will have limited impact if they cannot also explain their answers. In this work, we showed the first application of generative techniques to this task, and demonstrated how proofs, implication enumerations, and abductive inferences can be generated, exceeding the prior state-of-the-art in proof generation by +9% (absolute). In addition, the Iterative ProofWriter robustly generalizes to deeper proofs and more varied language than seen in training, and produces proofs that reflect (i.e., are faithful to) the model's actual inference decisions. Finally, the abductive capability offers the potential for generating proofs when both explicit and implicit knowledge are used, by materializing the implicit knowledge needed to complete the proof. Together, these significantly improve the viability of neural methods for systematically reasoning over language in practical settings. The ProofWriter datasets are available at https://allenai.org/data/proofwriter

Acknowledgements: We thank Google for providing the TPUs for conducting experiments.

References

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, S. Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *ICLR'20*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-SNLI: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pages 9539–9549.

S. Ceri, G. Gottlob, and L. Tanca. 1989. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.*, 1:146–166.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *IJCAI'20*.

Jay DeYoung, Sarthak Jain, Nazneen Rajani, E. Lehman, Caiming Xiong, R. Socher, and Byron C. Wallace. 2019. ERASER: A benchmark to evaluate rationalized nlp models. In *ACL*.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and C. Pal. 2020. Measuring systematic generalization in neural proof generation with transformers. In *NeurIPS'20*.

J. Hobbs, Mark E. Stickel, Douglas E. Appelt, and P. Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63:69–142.

N. Inoue, Pontus Stenetorp, and Kentaro Inui. 2020. R4C: A benchmark for evaluating RC systems to get the right answer for the right reason. In *ACL*.

Harsh Jhamtani and P. Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *EMNLP*.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *AKBC'19*.

K. Konolige. 1997. Abductive theories in artificial intelligence. In *Principles of Knowledge Representation*.

John McCarthy. 1984. Applications of circumscription to formalizing common sense knowledge. In *NMR*.

John W. McCarthy. 1959. Programs with common sense. In *Proc. Tedding Conf. on the Mechanization of Thought Processes*, pages 75–91.

Mark A Musen and Johan Van der Lei. 1988. Of brittleness and bottlenecks: Challenges in the creation of pattern-recognition and expert-system models. In *Machine Intelligence and Pattern Recognition*, volume 7, pages 335–352. Elsevier.

Sharan Narang, Colin Raffel, Katherine Lee, A. Roberts, Noah Fiedel, and Karishma Malkan. 2020. WT5?! training text-to-text models to explain their predictions. *ArXiv*, abs/2004.14546.

F. Petroni, Tim Rocktäschel, Patrick Lewis, A. Bakhtin, Y. Wu, Alexander H. Miller, and S. Riedel. 2019. Language models as knowledge bases? In *EMNLP*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *ArXiv*, abs/2009.03393.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, M. Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Tim Rocktäschel and S. Riedel. 2017. End-to-end differentiable proving. In *NeurIPS*.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *EMNLP'20*.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *ICLR*.

Sanjay Subramanian, Ben Bogin, Nitish Gupta, Tomer Wolfson, Sameer Singh, Jonathan Berant, and Matt Gardner. 2020. Obtaining faithful interpretations from compositional neural networks. In *ACL*.

Alon Talmor, Oyvind Tafjord, P. Clark, Y. Goldberg, and Jonathan Berant. 2020. LeapOfThought: Teaching pre-trained models to systematically reason over implicit knowledge. In *NeurIPS*.

Ming-Zhe Wang and Jun Deng. 2020. Learning to prove theorems by learning to generate theorems. In *NeurIPS*.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. Nlprolog: Reasoning with weak unification for question answering in natural language. In *ACL*.

Appendix: ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language

A Datasets: Additional Details

A.1 Statistics

Some overall statistics for the updated RuleTaker CWA and OWA datasets are in Table 10. The number of implications per theory can reach 20 and above, and the proof depths go up to 10, even though the proof depths of the associated questions are limited to the dataset depth (e.g., depth 3 for D3).

Dataset	# th	# qns	# impl min/mean/max	depth max
CWA:				
D0	27020	100002	0/1.0/18	8
D1	12965	100012	1/1.9/17	6
D2	9138	100014	2/3.3/18	5
D3	7067	100024	3/5.1/16	7
D5	4935	100030	5/9.8/21	10
Birds/Elec	140	5270	0/2.0/6	4
ParaRules	2403	40022	3/4.3/14	5
OWA:				
D0	26978	100000	0/0.8/18	5
D1	12933	100014	1/1.7/14	6
D2	9033	100010	2/3.1/14	5
D3	6940	100036	3/4.8/16	6
D5	4752	100030	5/9.1/21	10
Birds/Elec	140	5270	0/1.2/6	3
ParaRules	2403	40022	3/4.3/14	5

Table 10: Statistics for the CWA and OWA datasets, giving the number of theories, questions and implications per theory. Note that the maximum implication proof depth can go higher than the maximum proof depth for the included questions (e.g., for D5 the maximum questions depth is 5, but there are implications up to depth 10 which are include in the enumeration task).

Table 11 describes overall statistics for the datasets for Task 3: Abduction. Each abduction question can have zero or more missing facts as answer, and the proof depths can go up to 11.

Dataset	# th	# qns	# missing facts min/mean/max	max proof depth
D0-Ab	18011	85705	0/0.8/15	6
D1-Ab	10448	49808	0/0.8/12	7
D2-Ab	7092	37245	0/0.9/11	6
D3-Ab	5633	34915	0/1.1/11	8
D5-Ab	4362	35213	0/1.2/9	11
Birds-Electricity-Ab	140	3940	0/0.24/4	4

Table 11: Statistics for the Abduction datasets, giving the number of theories, abduction questions, number of missing facts per question and maximum proof depth.

A.2 Repairs to the RuleTaker Datasets

The original RuleTaker theories were intended to be full Datalog theories, but contained three occasional violations in the with-negation theories:

1. Some theories contained negated facts (e.g., "Bob is not red") and/or rules with negated conclusions. Such statements are redundant under a CWA, and not allowed according to formal Datalog specifications.
2. Some theories included rules with a free variable in a negated condition (e.g., "If someone is not blue then Bob is happy."). Such rules are not allowed according to formal Datalog specifications, as the possible groundings of the variable require meta-information about the theory as a whole.
3. A bug in the stratification checker led to a few theories being included that were not stratifiable, and hence may have multiple, valid truth assignments for their facts.

As a result, the theories were regenerated (with the same distribution over number of facts, rules, condition, etc.) to create the D*(CWA) datasets, avoiding these issues.

The D*(OWA) datasets are similar to the D*(orig) datasets, but evaluated without a CWA, i.e., negation-as-failure (NAF) is replaced with hard negation. The theories with negation were again regenerated to ensure they were stratifiable (to avoid negation cycles), but they still retain negated facts and rule conclusions. The truth values of the questions were recomputed using an OWA, resulting in answers *True/False/Unknown*.

A.3 The Birds-Electricity Datasets

The RuleTaker "birds" rulebase is a well-known logic problem illustrating the use of "abnormality" predicates (McCarthy, 1984),[7], and converted into English by hand. The dataset contains a single theory of six rules (e.g., "If someone is a bird and wounded then they are abnormal.") and seven facts (e.g., "Bill is wounded"), and forty questions against this theory (i.e., 40 test examples total). Birds1 and Birds2 differ solely in the English wording (e.g., "Bill is flying" vs. "Bill can fly").

---

[7] https://www.doc.ic.ac.uk/∼mjs/teaching/KnowledgeRep491/ExtendedLP_491-2x1.pdf, p5

		Answer		**Proof**	
	# qns	PRover	ProofWriter	PRover	ProofWriter
D=0	2968	99.7	99.9	99.4	99.9
1	2406	98.6	99.3	97.3	99.3
2	1443	98.2	98.3	88.7	97.7
3	1036	96.5	98.2	89.9	96.5
4	142	88.0	91.5	76.1	83.1
All	8008	98.4	99.1	95.1	98.5

Table 12: [Task 1: Proof Generation] Train on D3 + ParaRules, test on (only) ParaRules. Both systems demonstrate robustness to more complex linguistic expressions in the theories, with ProofWriter obtaining 3% higher proof correctness.

The four RuleTaker "electricity" datasets contain examples of reasoning about toy electrical cicuits using a small set of general rules about circuits. Examples in each dataset are built using a fixed set of general rules per dataset, ranging from five rules (Elec1) to twelve rules (Elec4). Each example in these datasets contains the general rules, plus between two and five facts describing a particular circuit, with a set of questions about the circuit, e.g., Q: "The light bulb is glowing?" A: *True*.

A.4 The ParaRules Dataset

The RuleTaker "ParaRules" dataset contains 40k questions against 2k theories expressed in paraphrased natural language, obtained by having crowdworkers rephrase the templated English facts and rules from sampled original theories into more varied natural language. For example, "Bob is cold." might be rephrased "In the snow sits Bob, crying from being cold"; or "Alan is round. Alan is blue. Alan is kind." might be rephrased "Alan, who is round and also kind, tends to be rather blue"; or "If someone is kind then they are young." might be rephrased "A kind person will certainly be young.". While the previous datasets contain synthetic language, ParaRules tests the models' ability to reason over more human-like paraphrased language.

B Additional Results

B.1 Results on the OOD ParaRules Dataset

We also test the robustness of ProofWriter's proof generation to theories that use more varied natural language, summarized in Section 5.1.2. Following (Clark et al., 2020) and (Saha et al., 2020), we train on the combined training partitions of D3(orig) and ParaRules, then test on the ParaRules test partition. The results in Table 12 show that PRover and ProofWriter (All-At-Once) are robust to more complex natural language in the input, with ProofWriter obtaining 3% higher proof correctness.

Test Dataset:	**# qns**	**F1**	**Acc**
Birds1-Ab	14	100.00	100.00
Birds2-Ab	14	100.00	100.00
Elec1-Ab	114	89.47	89.47
Elec2-Ab	126	90.25	88.89
Elec3-Ab	456	81.79	76.32
Elec4-Ab	3216	85.77	83.99
All	3940	85.66	83.53

Table 13: [Task 3: Abduction] Zero-shot scores of the D3-Ab model on the Birds-Electricity-Ab rulebases.

B.2 Abduction: Generalization to New Tasks

Section 5.5.2 summarized the results of testing abductive reasoning on abductive versions of the Birds-Electricity(OWA) theories. The detailed results are shown in Table 13, showing perfect zero-shot performance for the simple Birds rulebases, but progressively reduced performance for the Electricity theories as they get more complex. This indicates that the abductive task remains only partially solved by our generative model.

C Results with T5-large

In the main part of the paper we trained ProofWriter starting from the largest available T5-11B model (11 billion parameters). If we instead use the more manageable T5-large model (770 million parameters), the scores generally go down, but typically by a small amount.

In Tables 14 and 15 we show two examples of this, for the All-At-Once and Iterative ProofWriter models respectively, when training on the D3 dataset and evaluating on D5. We see the T5-large model is a bit worse on higher depth proof accuracy in the All-At-Once model, but is otherwise quite competitive.

D Hyperparameters and I/O Examples

We fine-tune the models on the training set using the default hyperparameters (including the Adafactor optimizer) in the T5 library.[8] We use the largest T5-11B model for the main results, fine-tuned for 40k steps (batch size 8), selecting the checkpoint with highest validation score (usually the final step). See Appendix C for results using the smaller T5-large.

---

[8] https://github.com/google-research/text-to-text-transfer-transformer

	Answer				Proof			
	CWA		OWA		CWA		OWA	
Depth	large	11B	large	11B	large	11B	large	11B
N/A	98.4	99.6	97.4	99.4	98.4	99.6	97.4	99.4
0	100	100	100	100	100	100	100	100
1	100	99.9	99.9	99.9	99.4	99.7	99.3	99.8
2	99.8	99.4	99.7	99.8	97.5	98.2	97.6	98.8
3	100	99.2	99.7	99.8	90.4	93.4	91.2	94.5
4	98.9	95.4	99.5	99.3	38.6	69.9	46.9	71.4
5	92.3	72.9	98.9	93.7	12.4	27.4	24.4	35.1
All	98.4	96.6	98.7	99.0	83.4	88.9	85.6	90.2

¡

Table 14: [Task 1] Comparing T5-large vs T5-11B for the All-At-Once models trained on D3 and evaluated on D5. T5-large is actually a bit ahead of T5-11B on answer accuracy (for CWA), although the proof correctness is noticeably higher with T5-11B.

	Answer				Proof			
	CWA		OWA		CWA		OWA	
Depth	large	11B	large	11B	large	11B	large	11B
N/A	99.0	99.7	99.2	99.9	99.0	99.7	99.2	99.9
0	100	100	100	100	100	100	100	100
1	98.8	99.8	99.1	99.3	95.0	95.4	97.5	97.8
2	98.3	99.5	98.9	99.7	91.0	91.7	96.4	97.3
3	98.6	99.7	98.4	99.2	89.0	90.4	95.5	97.1
4	98.0	99.7	97.5	99.1	86.3	88.9	93.4	96.5
5	97.7	98.9	96.5	98.8	85.4	87.8	82.3	86.4
All	98.7	99.7	98.8	99.6	94.4	95.4	96.4	97.6

Table 15: [Task 1] Comparing T5-large vs T5-11B for the Iterative ProofWriter trained on D0-D3 and evaluated on D5. T5-11B is systematically slightly better.

For all tasks in this paper, the model input and output are plain textual strings in a certain format. The format is slightly more verbose than strictly necessary, as it is compatible with a more general framework combining different views of the task. This is not typically an issue, although there are a few cases where the iterative model reaches the default 512 token limit of T5 when adding implications to the theory. We now show input/output examples for proof generation (All-At-Once and (one step of) the Iterative approaches), implication generation (All-At-Once approach; Iterative implication generation is a side-effect of Iterative proof generation), and abduction.

D.1 All-At-Once Proof Generation

In this task we are given a theory and question, and need to produce an answer (True/False/Unknown) and proof with intermediates.

INPUT: $answer$; $proof$; $question$ = The lion is not nice? ; $context$ = sent1: The tiger chases the lion. sent2: The lion is not big. sent3: If something visits the dog and it is kind then it visits the mouse. sent4: The dog chases the lion. sent5: If something is big then it visits the dog. sent6: The tiger eats the dog. sent7: If something visits the tiger and the tiger is not red then it is not nice. sent8: If something chases the dog then it is not red. sent9: The mouse does not chase the tiger. sent10: If something visits the mouse then the mouse is red. sent11: The tiger visits the lion. sent12: The tiger does not eat the mouse. sent13: The mouse is nice. sent14: The lion does not eat the tiger. sent15: If the tiger visits the dog and the dog is not big then the dog chases the mouse. sent16: The lion visits the tiger. sent17: If something chases the lion and it visits the dog then it chases the dog. sent18: The dog is nice. sent19: If something chases the lion then it is big. sent20: If something eats the lion then it is not cold.
OUTPUT: $answer$ = True ; $proof$ = # sent7@int1 & sent16 # sent8@int2 # sent17@int3 & sent1 # sent5@int4 # sent19@int5 sent1 ; with int1: The lion is not nice. ; int2: The tiger is not red. ; int3: The tiger chases the dog. ; int4: The tiger visits the dog. ; int5: The tiger is big.

If we translate back from Polish Notation, this proof corresponds to ((sent16 & ((((sent1 & ((sent1 → sent19 → int5) → sent5 → int4)) → sent17 → int3)) → sent8 → int2)) → sent7 → int1)

D.2 Iterative Proof Generation (one step)

In this task we ask the model to generate one valid implication of the theory with proof of depth 1. If no such implication exists, instead generate ”None” as the answer.

INPUT: $answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: If something eats the cow and it is big then the cow sees the bald eagle. sent2: If something likes the bald eagle then it is rough. sent3: If something eats the dog then it likes the cow. sent4: Big things are young. sent5: If something likes the cow then it eats the cow. sent6: If something sees the bald eagle then the bald eagle eats the cow. sent7: If something likes the bald eagle then the bald eagle is kind. sent8: If something sees the bald eagle then the bald eagle eats the dog. sent9: The bald eagle eats the cow. sent10: The bald eagle sees the dog. sent11: The dog is big. sent12: The cow likes the bald eagle. sent13: The bald eagle is young. sent14: The dog sees the cow. sent15: The bald eagle is kind. sent16: The dog is young. sent17: The bald eagle sees the cow. sent18: The bald eagle is rough. sent19: The cow eats the bald eagle. sent20: The dog is cold. sent21: The dog likes the cow. sent22: The dog eats the bald eagle. sent23: The dog eats the cow. sent24: The bald eagle likes the dog. sent25: The bald eagle likes the cow. sent26: The cow sees the bald eagle. sent27: The cow sees the dog.
OUTPUT: $answer$ = The cow is rough. ; $proof$ = # sent2 sent12

During evaluation time, we would then take this output and append it to the theory as ”sent28: The cow is rough.” and repeat the question. If a subsequent proof then references ”sent28”, we know already that this was derived using the proof ”# sent2 sent12”.

D.3 All-At-Once Implication Enumeration

In this task we ask the model to simply output all valid implications of a theory, without generations proofs in this case.

INPUT: \$answer\$; \$proof\$; \$question\$ = What are all the inferences? ; \$context\$ = sent1: The tiger chases the lion. sent2: The lion is not big. sent3: If something visits the dog and it is kind then it visits the mouse. sent4: The dog chases the lion. sent5: If something is big then it visits the dog. sent6: The tiger eats the dog. sent7: If something visits the tiger and the tiger is not red then it is not nice. sent8: If something chases the dog then it is not red. sent9: The mouse does not chase the tiger. sent10: If something visits the mouse then the mouse is red. sent11: The tiger visits the lion. sent12: The tiger does not eat the mouse. sent13: The mouse is nice. sent14: The lion does not eat the tiger. sent15: If the tiger visits the dog and the dog is not big then the dog chases the mouse. sent16: The lion visits the tiger. sent17: If something chases the lion and it visits the dog then it chases the dog. sent18: The dog is nice. sent19: If something chases the lion then it is big. sent20: If something eats the lion then it is not cold.
OUTPUT: \$answer\$ = The dog is big. The tiger is big. The dog visits the dog. The tiger visits the dog. The dog chases the dog. The tiger chases the dog. The dog is not red. The tiger is not red. The lion is not nice.

This particular theory has 9 valid implications, around average for D5 theories. (The number of implications can be as high as 21).

D.4 Iterative Implication Enumeration

Iterative implication generation is a side-effect of Iterative proof generation, hence the input/output (one step) is the same as in Appendix D.2.

D.5 Abduction

In this task we ask the model to output all alternative missing facts, separated by a comma, that can be (separately) added to a theory so as to prove the given implication to be True.

INPUT: \$answer\$; \$question\$ = Dave is rough. ; \$context\$ = triple1: Anne is white. triple2: Charlie is young. triple3: Dave is round. triple4: Erin is quiet. rule1: If someone is rough and young then they are blue. rule2: Rough, white people are smart. rule3: All smart people are rough. rule4: All white people are smart. rule5: If someone is young then they are smart. rule6: All smart people are rough.
OUTPUT: \$answer\$ = Dave is young. , Dave is smart.

This particular abduction question has 2 alternative missing facts. Any one of them when added to the input theory is sufficient to prove the implication 'Dave is rough'.

Probing Image–Language Transformers for Verb Understanding

Lisa Anne Hendricks Aida Nematzadeh
DeepMind
`{lmh, nematzadeh} @google.com`

Abstract

Multimodal image–language transformers have achieved impressive results on a variety of tasks that rely on fine-tuning (*e.g.*, visual question answering and image retrieval). We are interested in shedding light on the quality of their pretrained representations – in particular, if these models can distinguish different types of verbs or if they rely solely on nouns in a given sentence. To do so, we collect a dataset of image–sentence pairs (in English) consisting of 421 verbs that are either visual or commonly found in the pretraining data (*i.e.*, the Conceptual Captions dataset). We use this dataset to evaluate pretrained image–language transformers and find that they fail more in situations that require verb understanding compared to other parts of speech. We also investigate what category of verbs are particularly challenging.

1 Evaluating Verb Understanding

The success of image–language models in real-world applications relies on their ability to relate different aspects of language (such as verbs or objects) to images, which we refer to as multimodal understanding. For example, an image-retrieval model needs to distinguish between "eating an apple" and "cutting an apple" and a captioning model must accurately describe the actions in a scene.

Previous work shows that image–language benchmarks do not always fully measure such multimodal understanding: object retrieval models fail to account for linguistic structure (Akula et al., 2020), visual question answering (VQA) models overly rely on language priors (Goyal et al., 2017; Agrawal et al., 2018), and captioning metrics do not always measure if captions "hallucinate" objects in an image (Rohrbach et al., 2018). Inspired by this, prior work introduced tasks to specifically examine whether models can relate objects to images (Shekhar et al., 2017) or classify frequent interactions associated with objects (Chao et al., 2015). However, both these datasets are limited to the 80 objects in the MSCOCO detection challenge (Lin et al., 2014).

To address this gap, we design a benchmark focused on verbs called SVO-Probes for examining **s**ubject, **v**erb, **o**bject triplets; more specifically, we collect a set of image–sentence pairs (in English) where each pair is annotated with whether the sentence corresponds to the image or not. As shown in Fig. 1, for a given sentence, in addition to a *positive* image that matches the sentence, our dataset includes controlled *negative* images that do not correspond to specific aspects of the sentence (*i.e.*, subject, verb, and object). These controlled examples enable us to probe models for their understanding of verbs as well as subjects and objects. Our dataset consists of 421 verbs and includes over $48,000$ image–sentence pairs.

We use our benchmark to evaluate the recent family of multimodal (image–language) transformers that have shown impressive results on benchmarks like VQA and image retrieval (Lu et al., 2019; Chen et al., 2020; Tan and Bansal, 2019; Li et al., 2020b,a; Huang et al., 2020). Our goal is to investigate if the good performance of these models is due to learned representations that successfully relate different aspects of language to images. More specifically, we evaluate a few architectural variations of these models in a zero-shot way by using the pretrained models to classify if image–sentence pairs from SVO-Probes match.

Our results show that the performance of all evaluated models is worst on verbs, with subjects being easier than verbs but harder than objects. We find that this observation does not depend on the frequency of test examples in pretraining data. Moreover, it is considerably harder for all models to correctly classify image–sentence pairs that do not

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3635–3644
August 1–6, 2021.

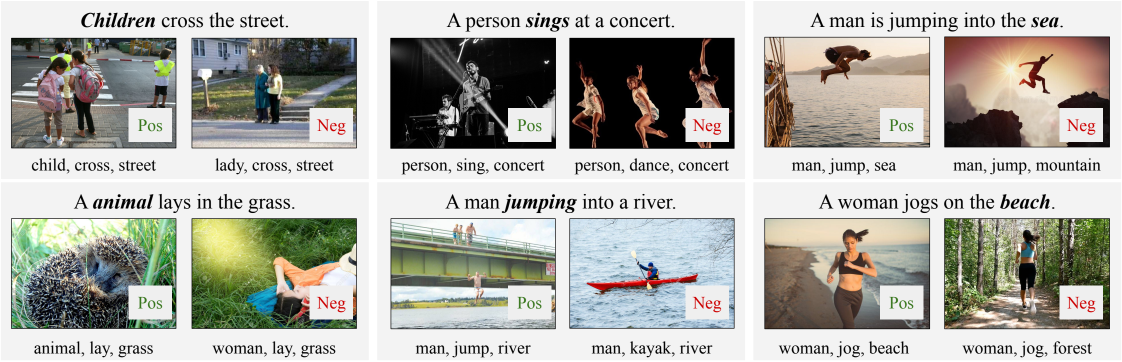

Figure 1: Examples from SVO-Probes. Images on the left and right show positive and negative image examples for each sentence. Below each image is the ⟨subject, verb, object⟩ triplet corresponding to the image.

match; the image–language transformers overpredict that sentences corresponds to images.

Additionally, we compare an image–language transformer pretrained on a large automatically-curated dataset (*i.e.*, Conceptual Captions, Sharma et al., 2018) with one pretrained on the smaller but manually-annotated MSCOCO (Chen et al., 2015). Conceptual Captions is more noisy than MSCOCO in that its sentences do not necessarily correspond to its images. Interestingly, we observe that the model pretrained on MSCOCO performs better. This result shows that the image–language transformers are not robust to dataset noise as they learn to predict that somewhat-related image–sentence pairs correspond to each other.

Despite their good performance on downstream tasks, image–language transformers fail on our task that requires multimodal understanding since they cannot distinguish between finer-grained differences between images. Our results highlight that there is still considerable progress to be made when training multimodal representations, and that verbs in particular are an interesting challenge in image–language representation learning.

2 Related Work

Image–language transformers build on the transformer architecture (Vaswani et al., 2017) by incorporating additional loss functions (to learn image features and align image and language modalities), using self-attention to combine modalities, and training on paired image–text data (Lu et al., 2019; Chen et al., 2020; Tan and Bansal, 2019; Li et al., 2020b,a; Huang et al., 2020). The impressive performance of these models on many image–language benchmarks has inspired recent work that studies different architectural choices made in these models (Cao et al., 2020; Hendricks et al., 2021).

Compared to previous image–language models, multimodal transformers both use a new architecture and are frequently trained on a much larger dataset – the Conceptual Captions dataset consisting of $3m$ image–text pairs (Sharma et al., 2018). Singh et al. (2020) show that on fine-tuned tasks, the performance of multimodal transformers (*i.e.*, Lu et al., 2019; Li et al., 2019) are less sensitive to dataset size; the domain match between pretraining and fine-tuning datasets is more important.

Datasets. Our proposed dataset is most similar to the FOIL benchmark (Shekhar et al., 2017) which tests if image–language models can differentiate between sentences that vary with respect to only one noun. FOIL consists of $64,300$ images from MSCOCO (Chen et al., 2015); each image is paired with a corresponding sentence that describes the image (*i.e.*, a *positive* example) and one that does not (*i.e.*, a *negative* example). Negative sentences are collected by replacing object words in the positive sentences with a similar object (*e.g.*, changing the word “dog” to “cat” in “The dog ran.”). Shekhar et al. (2017) use the FOIL dataset in a few tasks including a classification task where the model is asked to classify if a sentence matches the image or not. We use the same task setup because it allows us to probe image–language transformers in a zero-shot setting as these models are generally trained to classify whether an image–text pair match. Our work is different than FOIL in that we focus on verb understanding as opposed to noun understanding; moreover, our dataset provides different negative types (by replacing subjects, verbs, or objects).

Other datasets focus on relationship or interaction detection (*e.g.*, HICO and VRD; Chao et al., 2015; Lu et al., 2016a). These datasets are evalu-

dataset	Ims	Subjs	Verbs	Objs	Sents	Negs
FOIL	32k	n/a	0	70	✓	✓
HICO	10k	n/a	117	80	✗	✓
VRD	1k	100	70	100	✗	✗
V-COCO	5k	n/a	26	48	✗	✗
ImSitu	25k	950	504	1840	✗	✗
SVO-Probes	14k	100	421	275	✓	✓

Table 1: **Im**ages, **Subj**ects, Verbs, **Obj**ects, **Sent**ences, and **Neg**atives in other datasets and SVO-Probes. Image numbers are for the *evaluation* set.

ated in a classification setting in which the input is an image and the output is a detected relationship (for HICO, an object and interaction, for VRD two objects and their relationship) and have a limited number of verbs and objects. V-COCO (Gupta and Malik, 2015) and ImSitu (Yatskar et al., 2016) both includes verbs but do not provide negatives for a controlled evaluation of verb (or noun) understanding. Finally, other work has explored how creating hard negatives (*e.g.*, by substituting words in train examples) leads to better test performance (Gupta et al., 2020; Hendricks et al., 2018; Faghri et al., 2017). In contrast, our work focuses on creating hard evaluation examples to probe learned representations.

In summary, SVO-Probes is unique as it tests understanding of a broad range of verbs as well as subjects and objects in a controlled way. Furthermore, our dataset includes image–sentence pairs; thus, it can be used to evaluate image–language transformers that process image–sentence pairs. Finally, SVO-Probes is designed as a zero-shot task to evaluate pretrained image–language transformers and is collected to have a similar distribution to Conceptual Captions which is commonly used in pretraining these models. See Table 1 for a comparison between SVO-Probes and other datasets.

3 Task Setup and Dataset Collection

Our goal is to examine *verb-understanding* in *pretrained* multimodal transformers. To do so, we need a task that requires an understanding of a given verb in a sentence, *e.g.*, a model cannot succeed at the task by relying on nouns. We also need to include a diverse set of verbs, and examine each verb in at least a few situations. To test the pretrained representations, we need to examine the models in a zero-shot setting (without fine-tuning).

Inspired by the FOIL setup (Shekhar et al., 2017), we use a zero-shot classification task where a model is asked to identify if a sentence and an image correspond to each other. As a result, we need a dataset that provides "match" or "not match" labels between images and sentences. We collect a dataset of image–sentence pairs (SVO-Probes) that given a sentence, provides such labels for at least two images.[1] Some of these images are *positive* examples, *i.e.*, the sentence correctly describes them. Others are *negative* examples where some aspect of the sentence (*e.g.*, verb) does not match the image. Figure 1 shows some examples from our dataset.

We systematically collect negative examples such that they only differ from the positive image with respect to the subject, verb, or object of the sentence. Finally, we consider sentences whose subjects, verbs, and objects are frequent in the Conceptual Captions (CC) training dataset. Since CC is the dataset most frequently used for pretraining multimodal transformers, we can examine what the pretrained representations capture (in contrast to examining these models' generalization ability). We next describe our pipeline to create SVO-Probes.

Creating a verb list. To ensure that we have a large number of verbs in our dataset, we first created a verb list by considering a subset of verbs that occur in the train split of the Conceptual Captions dataset (CC-train). More specifically, we consider verbs that are visually recognizable in the images; to identify the visual verbs, we use the *imSitu* dataset (Yatskar et al., 2016) that includes verbs that annotators marked as reliably recognizable. Moreover, we include verbs that occur at least 50 times in CC-train.

Curating triplets. Given a positive example, we need to systematically generate negatives by replacing the subject, verb, or the object. As a result, we collect a set of ⟨subject, verb, object⟩ (SVO) triplets from CC-train sentences for our verbs. We extract the subject, verb, and direct object from the dependency parse trees. and remove triplets where subjects or objects are pronouns or have less than two characters. Finally, we discard SVO triplets with frequency smaller than five.

We consider three negative types for a given triplet: a subject-, verb-, or object-negative where respectively, the subject, verb, or object in the triplet are replaced by a different word. For example, given the triplet ⟨girl, lie, grass⟩, examples of subject-negative, verb-negative, and object-

---

[1] We note that our dataset is limited to English sentences; we simply use "sentences" to refer to English sentences.

negative are ⟨puppy, lie, grass⟩, ⟨girl, sit, grass⟩, and ⟨girl, lie, beach⟩.

Since our goal is to examine verb understanding, we only keep the triplets that have at least one verb negative. This enables us to evaluate a model's capacity in distinguishing images that mainly differ with respect to the verb; for example, ⟨girl, lie, grass⟩ vs. ⟨girl, sit, grass⟩. Adding this constraint results in 11230 SVO triplets and 421 verbs. In this set, 1840 SVO triplets (and 53 verbs) have at least two verb and object negatives.

Collecting images. The next step is collecting images that match the curated SVO triplets. We query for SVO triplets using the Google Image Search API. We retrieve 5 images for each triplet, then remove any images with urls in Conceptual Captions. To make sure that these automatically-retrieved images certainly match the triplets, we set up an annotation task where we ask workers on Amazon Mechanical Turk (AMT) to verify if the subject, verb, and object are present in the image. We ask three people to annotate each image, and only keep images where at least two annotators agree that the subject, verb, and object are depicted in the image. Moreover, we discard images marked as a cartoon by annotators. We find that 58% of our images pass this initial annotation process. We pay workers \$0.04 per HIT for all tasks.

Collecting sentences. Multimodal transformer models are trained on pairs of images and *sentences*; to evaluate them, we require image–sentence pairs as opposed to image–SVO pairs. Given an image and an SVO triplet, we next ask annotators to write a sentence that uses all the words in the triplet and describes the image. For example, as shown in Figure 1 top right, given the triplet ⟨man, jump, sea⟩, an annotator might write "A man is jumping into the sea.". We ask annotators to refrain from writing additional information to ensure that a collected sentence examines the words in the SVO (as opposed to words that we are not controlling for). Annotators are given the option to not write a sentence if they do not think the subject, verb, and object can be combined into a grammatical sentence that describes the image. 86% of our images pass this phase of our pipeline.

We observe that for a given SVO, different images elicit slightly different sentences. For example, the triplet ⟨person, jog, beach⟩ resulted in the sentences "A person jogging along the beach." and "A person jogs at the beach.". Additionally, annotators pluralize nouns to ensure the sentence describes the image (*e.g.*, Figure 1 top left, the subject "child" is written as "children" in the sentence).

Confirming the negative image. Finally, given a *positive* triplet (*e.g.*, ⟨girl, lie, grass⟩) and its negative (*e.g.*, ⟨girl, sit, grass⟩), we need to confirm that the positive's sentence does not match the image retrieved for the negative triplet. To do so, we ask three annotators to select which images (positive, negative, neither, or both) match a given sentence. Image–sentence pairs where two out of three annotators agree are accepted into our dataset; 68% of the pairs pass this final annotation stage.

4 Experimental Setup and Results

We investigate if current image–language transformers can relate different aspects of language (and in particular verbs) to images by evaluating these models against both FOIL and SVO-Probes. More specifically, we evaluate a few architectural variations of image–language transformer models (based on the implementation of the models by Hendricks et al., 2021) that differ in their choice of multimodal attention and loss functions; this way we can examine whether our findings are sensitive to these slight differences. The base multimodal transformer (**MMT**) closely replicates the ViLBERT architecture (Lu et al., 2019): this model includes three loss functions, masked language modeling (MLM) and masked region modeling (MRM) losses on the language and image inputs and an image–text matching (ITM) loss that classifies if an image–sentence pair match. Importantly, the multimodal attention of MMT is similar to the hierarchical co-attention in Lu et al. (2016b) where each modality (*i.e.*, image or language) attends *only* to the other modality. More specifically, in the multimodal self-attention layer of transformer (Vaswani et al., 2017), for queries on the language input, keys and values are taken from images and vice versa.

Different interactions of image (language) queries, keys, and values in multimodal self-attention results in variations of image–language transformers. We describe the model variations we study in Table 2. We also consider models that either lack the MLM or MRM loss. Models are pretrained on Conceptual Captions (CC) unless stated otherwise. For reference, we report the Recall@1 performance on the zero-shot image-retrieval task

Name	Multimodal Attention	Similar Model	MLM	MRM	*ZS Flickr*
MMT	Queries from L (I) take values and keys from *only* I (L)	ViLBERT; LXMERT	✓	✓	**41.9**
Merged–MMT	Queries from L (I) take values and keys from *both* L and I	UNITER	✓	✓	**40.0**
Lang–MMT	Queries are *only* from L (Hendricks et al., 2021)		✓	✓	33.6
Image–MMT	Queries are *only* from I (Hendricks et al., 2021)		✓	✓	31.6
SMT	Single-Modality Transformers without multimodal attention		✓	✓	16.9
No-MRM–MMT	The same as MMT		✓	✗	**41.1**
No-MLM–MMT	The same as MMT		✗	✓	20.2

Table 2: Different variants of the image–language transformer architecture we test. L and I stand for language and image, respectively. We note that models with Merged attention (like UNITER) are also referred to as single-stream models. ViLBERT: Lu et al. (2019); LXMERT: Tan and Bansal (2019); UNITER: Chen et al. (2020)

on Flickr (*ZS Flickr*), where a model must retrieve an image from the Flickr dataset (Young et al., 2014) that matches an input sentence. Since MMT performs best on *ZS Flickr*, we do most of our experiments on this model unless stated otherwise.

We first evaluate our image–language transformers on FOIL to examine their noun understanding and then test them on SVO-Probes which probes for subject, verb, and object understanding in learned representations. Following FOIL, we report the accuracy on positive and negative pairs. All our models have an image-text classification output used in pretraining to align images and sentences. We calculate accuracy by passing images through our models and labeling an image–sentence pair as negative if the classifier output is < 0.5 and positive otherwise. We report the average over the two pairs (see Avg columns in Tables 3 and 4) by weighting them equally, since we expect models to perform well on both positive and negative pairs. In FOIL, there are equal positive and negative pairs.

Another possible way to set-up our evaluations is as image-retrieval (reporting recall@1 as a metric). However, the retrieval setting does not highlight the difference in performance between positive and negative pairs. For example, a model might rank the pairs correctly even when their scores are very close (positive score is 0.91 and negative one is 0.9). In this example, the model is wrong about the negative pair (it is assigned a high score) but the retrieval setting does not capture this. However, the classification metric will penalize the model for assigning a high score to a negative pair. As a result, the classification metric better differentiates between the models by examining if they correctly label both the positive and negative pairs.

4.1 Evaluating Nouns with FOIL

We examine noun understanding in image–language transformers with the FOIL dataset (Shekhar et al., 2017). Given image–sentence pairs from FOIL, we evaluate the MMT model in a zero-shot setting by using it to classify if the image and sentence match. Table 3 compares MMT with the best model from the FOIL paper (HieCoAtt Shekhar et al., 2017) and, to our knowledge, the best-performing model on the task without using ground-truth annotations (Freq+MM-LSTM from Madhyastha et al., 2018). Note that these models are trained specifically for the FOIL task (*i.e.*, on the train split of FOIL), whereas the MMT model (pretrained on CC) is tested in a zero-shot setting.

MMT achieves an accuracy considerably worse than the best models on FOIL (Shekhar et al., 2017; Madhyastha et al., 2018) on all pairs; this is surprising given that image–language transformers achieve state-of-the-art results on zero-shot image retrieval tasks based on Flickr (Young et al., 2014) and MSCOCO (Chen et al., 2015). In particular, MMT overpredicts that image–sentence pairs match, resulting in the highest accuracy on the positive pairs (99.0) but the lowest on negative pairs (11.8). Thus MMT cannot distinguish between sentences that only differ with respect to nouns.

We investigate whether this poor performance of MMT is due to mismatch between the pretraining (*i.e.*, CC) and FOIL test (*i.e.*, MSCOCO) datasets. Thus, we compare our MMT model pretrained on Conceptual Captions with one pretrained on MSCOCO (MMT-COCO). As expected, MMT-COCO has considerably higher performance on all pairs (compare to MMT); however, the accuracy is still significantly higher on positive pairs than negative ones, showing that the model overpredicts that image–sentence pairs match. Our result shows that despite their impressive performance on downstream tasks, image–language transformer models perform poorly in distinguishing between semantically similar sentences. Next we examine how well these models perform on our proposed

Model	Avg	Pos.	Neg.
HieCoAtt*	64.1	91.9	36.4
Freq + MM-LSTM †	**87.9**	86.7	**89.0**
MMT	55.4	**99.0**	11.8
MMT-COCO	72.0	95.0	49.0

Table 3: Performance on FOIL averaged over all (Avg), positive (Pos.), and negative (Neg.) pairs. *Shekhar et al. (2017); †Madhyastha et al. (2018).

probing dataset which is designed to have a similar vocabulary to the CC pretraining dataset.

4.2 Comparing Models on SVO-Probes

We evaluate all models (see Table 2) on SVO-Probes and report overall accuracy and accuracy for subject, verb, and object negatives in Table 4.

The MMT model (with the best performance on *ZS Flickr*) performs poorly on SVO-Probes, achieving an overall average accuracy of 64.3. The best overall average accuracy (No-MRM–MMT; 69.5) shows that SVO-Probes is challenging for image–language transformers. In particular, models struggle with classifying negative pairs; Lang–MMT achieves the highest accuracy over negative pairs (56) which is slightly higher than chance at 50. [2]

Though No-MRM–MMT and MMT perform similarly on *ZS Flickr*, No-MRM–MMT performs better on SVO-Probes. This suggests that the masked region modelling loss is not needed for good performance on *ZS Flickr*; also, it impedes the model from learning fine-grained representations needed to perform well on SVO-Probes. More surprisingly, Lang–MMT, which performs worse on *ZS Flickr* than MMT, outperforms MMT on SVO-Probes. The image representations in Lang–MMT are not updated with an attention mechanism. In Sec. 4.4, we explore if the stronger attention mechanism in MMT leads to overfitting of the training images and thus weaker performance.

We crafted SVO-Probes such that it includes words from the pretraining dataset of image–language transformers (*i.e.*, CC), whereas FOIL is collected from MSCOCO. Comparing the performance of MMT (with CC pretraining) on FOIL and SVO-Probes (55.4 in Table 3 vs. 64.3 in Table 4), we see that the domain mismatch between pretraining and test data plays a role in MMT's performance. Interestingly, comparing the performance of MMT-COCO (MMT with COCO pretraining) on FOIL to MMT (with CC pretraining) on SVO-Probes, we find that SVO-Probes is more challenging than FOIL when there is no domain mismatch (72.0 in Table 3 vs. 64.3 in Table 4).

When comparing different negative types across *all* models, we observe that verbs are harder than subjects and objects; compare average accuracy for Subj., Verb, and Obj. Negative columns in Table 4. For example, in MMT, the subject and object negative average accuracies (67.0 and 73.4) are considerably higher than the average accuracy for verb negatives (60.8). Moreover, when breaking down the accuracies for positive and negative pairs (Pos. and Neg. columns in Table 4), we observe that the accuracies of positive pairs are similar (ranging between 80.2 and 94.4) across all models except SMT (which performs close to chance); however, for negative pairs, there is more variation in accuracy across models especially for verb negatives (ranging between 22.4 and 54.6, Neg. columns under "Verb Negative"). These results show that negative pairs are better than positive ones in distinguishing between different model architectures.

We also find that subjects are harder than objects across all models (when comparing average accuracies of subject and object negatives). To better understand this result, we examined 21 nouns that occur both as subjects and objects in SVO-Probes' sentences. Interestingly, over these 21 nouns, for our MMT model, the accuracies of negative pairs are 42.9 and 56.4 for subject and object negatives, respectively. This suggests that the subject position might be more challenging than the object one which we further explore in Sec. 4.3.

4.3 Accuracy and Frequency at Training

Our overall results on SVO-Probes (Table 4) show that for image–language transformers, verb negatives are more challenging than subject and object ones, and also subject negatives are harder than object ones. We examine if this observation is due to properties of SVO-Probes as opposed to differences specific to subjects, verbs, and objects. First, we explore whether the frequency of SVO triplets in pretraining data impacts the accuracy of negative pairs in our MMT model. We focus on negative

---

[2] We focus on image–language transformers, but we also tested a baseline model where image features are embedded with the detector used in our transformers and language features with BERT. Features are pooled using element-wise multiplication. This baseline achieves 66.3% accuracy overall with 75.4% and 57.3% accuracy on positives and negatives. Similar to transformers, performance on verbs is the worst.

	Overall			Subj. Negative			Verb Negative			Obj. Negative		
	Avg	Pos.	Neg.	Avg	Pos.	Neg.	Avg	Pos.	Neg.	Avg	Pos.	Neg.
# Examples	48k	12k	36k	8k	3k	5k	34k	11k	23k	11k	3k	8k
MMT	64.3	*93.8*	34.8	67.0	*94.4*	39.5	60.8	*93.8*	27.8	73.4	*94.4*	52.4
Merged–MMT	64.7	**94.4**	35.0	69.1	**94.9**	43.2	60.7	**94.4**	27.0	74.1	**94.9**	53.3
Lang–MMT	*68.1*	80.2	**56.0**	*71.5*	82.1	**60.9**	*64.5*	80.2	*48.9*	*77.7*	81.4	**74.1**
Image–MMT	64.3	91.6	37.0	68.2	92.1	44.2	59.7	91.6	27.8	75.6	91.5	59.6
SMT	52.4	49.1	*55.6*	52.6	47.7	57.5	51.8	49.1	**54.6**	53.9	50.7	57.0
No-MRM–MMT	**69.5**	85.4	53.7	**73.5**	87.4	*59.7*	**65.5**	85.6	45.5	**80.1**	86.2	**74.1**
No-MLM–MMT	60.8	92.3	29.3	64.8	93.9	35.8	57.4	92.5	22.4	69.5	93.6	45.5

Table 4: Results on SVO-Probes on different models for subject, verb, and object negatives. Best results are shown in bold; second best results are italicized.

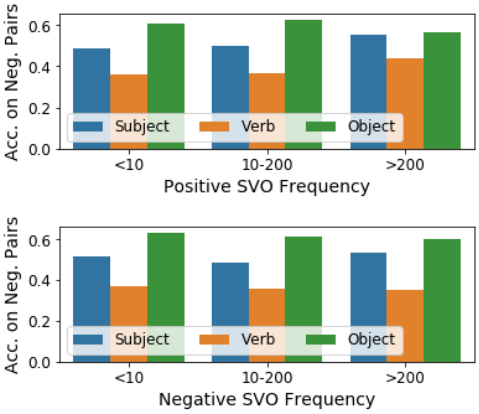

Figure 2: Accuracy of negative pairs for subject, verb, and object negatives given SVO frequencies in CC.

pairs as there is more variation in negative-pair accuracies across both models as well as subject, verb, and object negatives. We consider the frequency of positive and negative SVO triplets: a positive SVO corresponds to a positive image matching a given sentence, but a negative SVO and its extracted negative image do not match the sentence.

We group SVOs based on their frequency in CC-train into low (less than 10), medium (between 10-200), and high (greater than 200) frequency bins. Fig. 2 plots the negative-pair accuracy for subject, verb, and objects across these different frequency bins over positive and negative SVO frequencies. We confirm that our result on the difficulty of negative types does not depend on the frequency of positive or negative SVOs in pretraining data. In both plots of Fig. 2, the negative types in order of difficulty (lower accuracy) are verbs, subjects, and objects independent of the frequency bin.

Similarity between SVOs. We examine if the similarity between the SVO triplets corresponding to the negative and positive images can explain the difference in performance of subject, verb, and object negatives. For example, we expect that distinguishing ⟨child, cross, street⟩ and ⟨adult, cross, street⟩ to be harder than differentiating one of them from ⟨dog, cross, street⟩: "child" and "adult" are more similar to each other than to "dog". To test this, we measure the similarity between subjects, verbs, and objects in their corresponding negative types using the cosine similarity between word2vec (Mikolov et al., 2013) embeddings.

The average similarities between subjects, verbs, and objects are 0.49, 0.29, 0.27, respectively. Thus, subject words in negative examples tend to be more similar than object words. Furthermore, we find that there is a small positive correlation (as measured by Spearman rank correlation) between SVO similarity and classifier scores for negative pairs – .264 and .277 for subjects and objects respectively – suggesting that when SVOs corresponding to the image and sentence are similar, the classifier tends to assign a higher score (more positive) to the pair. This partially explains why accuracy on subjects is lower than on objects in Table 4. Even though verb negatives are harder for our model, the similarity for verb negatives is similar to that of object negatives. The correlation coefficient between similarity and classifier score is weaker (.145) for verbs, suggesting that word similarity factors less in how well the model classifies verb negatives.

4.4 Similarity to Pretraining Data

We next consider the similarity between images in SVO-Probes and CC. To measure the similarity between images, we sample 1 million images from CC. For each image in SVO-Probes, we find the 10 nearest neighbors in the feature embedding space

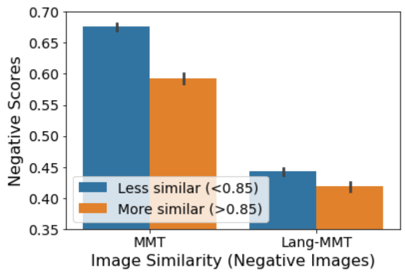

Figure 3: Comparing negative scores on MMT and Lang–MMT for images less or more similar to CC.

of CC, and average the distance to compute a similarity score for the image. Figure 3 plots the average score from our classifier for negative pairs with images that are less or more similar to the pretraining data (since we are classifying negative pairs, the lower score the better). We compare the MMT and Lang–MMT models since they have considerably different performance on SVO-Probes and *ZS Flickr*. The difference in average scores between less similar and more similar examples for MMT is 0.083. This is noticeably greater than the difference in average scores between less and more similar examples for Lang–MMT (0.024), suggesting that the image similarity influences Lang–MMT less than MMT. One hypothesis is that the stronger attention mechanism in MMT overfits to the training images which makes the MMT model less robust.

4.5 The Choice of Pretraining Dataset

In Sec. 4.2, we observe that models perform particularly poorly in classifying negative pairs. We investigate whether the choice of pretraining dataset impacts this observation. Conceptual Captions (CC), the most-common pretraining dataset for image–language transformers, is curated by scraping images and alt-text captions from the web. As a result, compared to manually-annotated datasets such as MSCOCO, CC is noisy – it contains examples where the sentence and its corresponding image do not completely align. For example, a sentence can mention objects that are not in the image or, in extreme cases, does not describe the image at all.

We hypothesize that image–language transformers treat correspondences due to dataset noise as "real" relations; in other words, they learn that if a image–sentence pair is somewhat semantically related, it should be classified as a positive match, even if some aspects of the sentence do not describe the image. At the same time, we can think of negatives in SVO-Probes as examples with noisy correspondences where a specific aspect of a sentence (*e.g.*, the verb) does not match the image. We compare our MMT model (with CC pretraining) to one trained on a manually-annotated and less noisy dataset, MSCOCO (referred to as MMT-COCO).

	Overall			Neg. Acc.		
Train	Avg.	Pos.	Neg.	S	V	O
CC	64.3	93.8	34.8	39.5	27.8	52.4
COCO	68.0	75.2	60.9	66.0	55.5	73.4

Table 5: Comparing performance when training our MMT model on COCO and CC.

Table 5 reports the overall accuracy of the two models on SVO-Probes as well as a breakdown over subject, verb, and object negatives for negative-pair accuracies. MMT-COCO performs better than MMT pretrained on CC (avg. accuracy of 68 vs 64.3). This is surprising since MMT-COCO has a different image and language distribution in its pretraining dataset. The accuracy of positive pairs in MMT-COCO is considerably lower than MMT while it performs noticeably better for negative pairs: unlike MMT, the MMT-COCO model does not overpredict that image–sentence pairs match. Our results show the image–language transformers are not robust to dataset noise. Less-noisy datasets (such as MSCOCO), despite their small size and domain mismatch, are more suitable for learning representations that are sensitive to finer-grained differences in images. Alternatively, models which are more robust to noise in datasets could be beneficial for tasks like ours.

4.6 Which Verbs Are the Hardest?

We investigate which verbs are hardest for MMT. We consider verbs with many examples in SVO-Probes: we keep SVO triplets with at least 30 negative images, resulting in a set of 147 verbs and 887 SVO triplets across $4,843$ images. Table 6 lists the easiest and hardest verbs (with highest and lowest accuracy for negative pairs) for the MMT model. Easy and hard verbs have a diverse set of properties; for example, easy verbs include sporting activities like "tackle" as well as verbs like "lead" that occurs in a variety of contexts. We also examine the 20 most difficult and easiest verbs for *all* our models (described in Table 2). Most difficult verbs for all models include: "cut", "argue", and"break" and the easiest ones include: "direct", "battle", "surround", "skate", and "participate".

Easy	Hard
tackle, reach, arrive, pitch, accept, congratulate, lead, present, celebrate, attend	argue, beat, break, burn, buy, cast, comb, crash, cut, decorate

Table 6: Hard and easy verbs for our MMT model

We test if verbs that occur in both SVO-Probes and imSitu are easier for our model to classify. Verbs in imSitu are considered visual as the dataset collection pipeline for imSitu includes an explicit annotation step to determine if verbs are visual. Surprisingly, we find verbs in imSitu are harder for our MMT model. On closer inspection, some verbs in our dataset but *not* in imSitu (*e.g.*, "swim") are clearly visual. An interesting future direction is to investigate which visual properties of a verb make it harder or easier for image–language models to learn.

5 Conclusions

Although image–language transformers achieve impressive results on downstream tasks, previous work suggests performance on these tasks can be confounded by factors such as over-reliance on language priors (Goyal et al., 2017). We collect a dataset of image–sentence pairs to examine multimodal understanding by testing the ability of models to distinguish images that differ with respect to subjects, verbs, and objects.

Our results show that image–language transformers fail at identifying such fine-grained differences; they incorrectly classify image–sentence pairs that do not match. Surprisingly, a model trained on a manually-annotated and smaller dataset does better on our task, suggesting that models have trouble ignoring noise in larger but automatically-curated pretraining datasets. Additionally, verb understanding is harder than subject or object understanding across all models we study. This motivates the need for researchers to not only examines models on objects, but develop datasets and architectures which allow for better verb understanding as well.

Acknowledgements

We would like to thank Antoine Miech for detailed comments on our paper. Also, thanks to Phil Blunsom, Laura Rimell, Stephen Clark, Andrew Zisserman, Jean-Baptiste Alayrac and the anonymous reviewers for their helpful feedback. We also thank Susie Young, Zhitao Gong, and Tyler Liechty for supporting our data collection.

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4971–4980.

Arjun R Akula, Spandana Gella, Yaser Al-Onaizan, Song-Chun Zhu, and Siva Reddy. 2020. Words aren't enough, their order matters: On the robustness of grounding visual referring expressions. *arXiv preprint arXiv:2005.01655*.

Jize Cao, Zhe Gan, Yu Cheng, Licheng Yu, Yen-Chun Chen, and Jingjing Liu. 2020. Behind the scene: Revealing the secrets of pre-trained vision-and-language models. *arXiv preprint arXiv:2005.07310*.

Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. 2015. HICO: A benchmark for recognizing human-object interactions in images. In *Proceedings of the IEEE International Conference on Computer Vision*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. UNITER: Universal image-text representation learning. In *European Conference on Computer Vision (ECCV)*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913.

Saurabh Gupta and Jitendra Malik. 2015. Visual semantic role labeling. *arXiv preprint arXiv:1505.04474*.

Tanmay Gupta, Arash Vahdat, Gal Chechik, Xiaodong Yang, Jan Kautz, and Derek Hoiem. 2020. Contrastive learning for weakly supervised phrase grounding. *ECCV*.

Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. 2018. Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 264–279.

Lisa Anne Hendricks, John Mellor, Rosalia Schneider, Jean-Baptiste Alayrac, and Aida Nematzadeh. 2021. Decoupling the role of data, attention, and losses in multimodal transformers. *arXiv preprint arXiv:2102.00529*.

Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. 2020. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. *arXiv preprint arXiv:2004.00849*.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. 2020a. Unicoder-vl: A universal encoder for vision and language by cross-modal pretraining. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. 2020b. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. 2016a. Visual relationship detection with language priors. In *European conference on computer vision*, pages 852–869. Springer.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016b. Hierarchical question-image co-attention for visual question answering. In *Advances in neural information processing systems*, pages 289–297.

Pranava Madhyastha, Josiah Wang, and Lucia Specia. 2018. Defoiling foiled image captions. *NAACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *NeurIPS*.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. *EMNLP*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. 2017. FOIL it! find one mismatch between image and language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Amanpreet Singh, Vedanuj Goswami, and Devi Parikh. 2020. Are we pretraining it right? digging deeper into visio-linguistic pretraining. *arXiv preprint arXiv:2004.08744*.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. In *Empirical Methods in Natural Language Processing*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Conference on Computer Vision and Pattern Recognition*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Implications of Using Internet Sting Corpora to Approximate Underage Victims

Tatiana Ringenberg, Kathryn Seigfried-Spellar, and **Julia Rayz**
Purdue Polytechnic
Purdue University
{tringenb, kspellar, jtaylo1} @purdue.edu

Abstract

Law enforcement officers (LEOs) and the justice system employ NLP models for classifying and triaging child exploitation cases due to the textual communications between predators and victims. The usefulness of these systems depend on the quality of data that can be used for training. Data in the domain are scarce, sensitive, and emotionally taxing for annotators. NLP researchers approximate victimization conversations using transcripts from internet stings performed by either vigilantes or LEOs, with an implicit assumption that vigilante or LEO conversations represent the victimization process. Psychology research, however, states that underage victim chats differ from internet stings in goal and modus operandi. We present a methodology and observations from annotating a corpus of victim, vigilante, and LEO conversations with convicted predators with the goal of comparing these chats. The corpus is annotated for stages and tactics of the victimization process described within psychology research. As predicted by psychological research, we found significant differences in the three classes of chats that are usually not taken into account in chat classification.

1 Introduction

Child exploitation crimes have expanded over the years to include a wide array of concerns including sextortion (Kopecký, 2017), sex trafficking (Diaz and Panangadan, 2020), sexual solicitation (Briggs et al., 2011), and deep fakes (Albahar and Almalki, 2019). NLP research in understanding these chats is crucial because law enforcement agencies have become overwhelmed with online cases; automated systems are needed in order to sift through the available textual data and transcripts to improve case triage through identification of criminal activity (Inches and Crestani, 2012). In the past, automatic systems have been developed for differentiating predators from non-predators (Misra et al., 2019; Pendar, 2007), predicting level of risk throughout a conversation (Ringenberg et al., 2019), and flagging predatory conversations (Kim et al., 2020; McGhee et al., 2011; Zuo et al., 2018).

While the potential of NLP in the child exploitation domain is substantial, the corpora used to train these models is not always adequate. The data used to train algorithms rely on internet sting operations between an adult predator and a law enforcement officer (LEO) (DeHart et al., 2017), or adult vigilante (Black et al., 2015), impersonating a minor. Psychology research, which serves as the foundation for theory in the child exploitation domain, suggests internet sting operations progress differently than traditional conversations in which an adult seeks to victimize a child (DeHart et al., 2017; Briggs et al., 2011; Bergen et al., 2013; Mitchell et al., 2005). As a result, further research is needed to assess the ways in which these corpora differ and the potential impact the differences have on the resulting models.

We investigate to what extent internet sting operations may be assumed to accurately approximate child victimization. We focus on differences between sting participants and underage victims with respect to the victimization process known as online child grooming: the process an adult uses to gain the trust of a minor for the purpose of sexual fulfillment either online or in a physical meeting (O'Connell, 2003).

According to Stede and Huang (2012), "the most convincing evidence for the value of an annotation task remains to be its direct contribution to the success of one or more NLP applications" (p.92). This work highlights the potential negative impact of untested assumptions related to corpus composition in this domain. We also provide insights into the annotation process which may help others as-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3645–3656
August 1–6, 2021.

sess corpora within child exploitation. Finally we plan to release our code book, coinciding with this paper, as an aid to future work.

2 Online Child Victimization as a Process

The victimization process by which an adult entices a minor is referred to as grooming (O'Connell, 2003), and has been studied within the online (Kloess et al., 2019; O'Connell, 2003) and physical (Colton et al., 2010; Jackson et al., 2015) contexts for decades (Kaufman et al., 1993; Lang and Frenzel, 1988; Leclerc and Tremblay, 2007). We seek to understand the impact to NLP research. Thus, we limited the scope of our investigation to the textual transcripts produced from the online grooming process as originally defined by O'Connell (2003).

2.1 Characteristics of Online Grooming

Online grooming is the process an adult uses, on the Internet, to entice a minor into a sexual scenario (O'Connell, 2003). The process consists of six transient stages which include friendship forming, relationship forming, exclusivity, risk assessment, sexual, and meeting (O'Connell, 2003; Gupta et al., 2012). For a discussion of the grooming stages, see O'Connell (2003); Whittle et al. (2013). The process of grooming is not uniform; it ebbs and flows (Gillespie, 2002) based on the goals of the predator (Beauregard et al., 2012; Briggs et al., 2011; Kloess et al., 2017) and feedback from the victim (Wortley et al., 2019).

Common tactics used during online grooming include bragging (Aitken et al., 2018), compliments (Kloess et al., 2019), fantasy enactment (Kloess et al., 2019; Malesky Jr, 2007), coercion (Kloess et al., 2019; Villacampa and Gómez, 2017), repetition (Kloess et al., 2017), and expression of vulnerability (Barber and Bettez, 2014). For a thorough treatment on online grooming tactics, see Barber and Bettez (2014); Kloess et al. (2019). Predators use grooming tactics in order to further a goal or progress the relationship with the minor (Beauregard et al., 2012; Briggs et al., 2011; Kloess et al., 2017; Wortley et al., 2019).

2.2 Corpora in Online Grooming

To understand how datasets and corpora are built within the child exploitation domain, we performed a systematic review of research on child exploitation from January 2000 to March 2020. We searched for peer-reviewed journal articles in four databases: Medline, PubMed, PsychInfo, and ERIC. In each database, we searched for the terms *online sexual grooming*, *online sexual solicitation*, *child sexual abuse*, and *child molestation*. Results included applied NLP papers and psychology papers. Based on titles and abstracts, we selected papers which met the following criteria: peer-reviewed journal article, published in English, contained adult groomer and underage victims, clear online child grooming tactics component, and articles with empirical focus. Articles which were literature reviews or meta-analyses of grooming were removed.

We identified a total of 32 articles. We analyzed the type of participants and the location of the participants in the sample. Ten papers used vigilante cases, 16 used underage victim cases, one used interactions with LEO, and five contained a mixture of participants. These datasets are problematic because they may not represent the phenomenon being studied (Bowen, 2008; Hovy and Lavid, 2010). A representative corpus would ideally include the space of all participant types and behaviors (Bowen, 2008). However, transcripts from these three groups may differ in length (Briggs et al., 2011; Mitchell et al., 2005), motivation (Briggs et al., 2011; Williams et al., 2013), and modus operandi (Briggs et al., 2011; DeHart et al., 2017; Williams et al., 2013). Additionally, the studies were done on participants from a wide range of geographical jurisdictions including the United States (Black et al., 2015), the United Kingdom (Whittle et al., 2014), Sweden (Shannon, 2008), Spain (de Santisteban et al., 2018), Israel (Katz, 2013), or multiple countries (Quayle et al., 2014). This is problematic due to differences between countries with respect to the age of consent, legal ramifications, and social constructs surrounding child sexual abuse.

Online grooming corpora are generally used by ML researchers to identify specific grooming stages (Cano et al., 2014; Gupta et al., 2012) and classify conversations as predatory (Bogdanova et al., 2012; McGhee et al., 2011). Methods for identifying grooming tactics have included rule-based systems (McGhee et al., 2011), naïve bayes (Bogdanova et al., 2012), support vector machines (Gunawan et al., 2016), and more recently neural networks (Ebrahimi et al., 2016). Some of the common features used for classifying grooming conversations and stages include sentiment polarity

(Bogdanova et al., 2012; Cano et al., 2014), psycholinguistic categories (Cano et al., 2014; Gupta et al., 2012), and n-grams (Bogdanova et al., 2012).

2.3 Underage victims and internet stings: Could they differ?

Internet sting datasets do not represent the underage victimization process (Bergen et al., 2013; Briggs et al., 2011; Chiang and Grant, 2019; DeHart et al., 2017; Gijn-Grosvenor and Lamb, 2016; Mitchell et al., 2005; Schneevogt et al., 2018; Winters et al., 2017) as: 1) the type of predator caught in internet stings versus real victimization cases is different (Briggs et al., 2011; Mitchell et al., 2005); and 2) there are differences between the real vulnerabilities and reactions of at-risk minors versus trained vigilantes and LEOs with a goal of gathering evidence (Briggs et al., 2011; Chiang and Grant, 2019; DeHart et al., 2017; Gijn-Grosvenor and Lamb, 2016; Schneevogt et al., 2018; Williams et al., 2013). The first is impactful for NLP research because it affects the topics the predators pursue and the progress of stages. For instance, looking for sexually-charged language would likely be more effective in internet-stings than in underage victim chats (Briggs et al., 2011). The second is impactful because the data used to train algorithms is not representative of successful grooming techniques, especially if predators caught during stings were not familiar with grooming tactics for luring minors. These factors affect how openness and directness of communication is used by law enforcement and vigilantes in comparison to underage victims (Briggs et al., 2011; Williams et al., 2013).

Conversely, DeHart et al. (2017) found many internet-based predators requested meetings with victims. While more research is needed, empirical evidence shows the predators in internet stings may be different enough to impact how the online grooming process manifests (Briggs et al., 2011; Mitchell et al., 2005).

Psychologists also posited several ways in which the grooming process may differ in terms of modus operandi. Briggs et al. (2011); DeHart et al. (2017); Williams et al. (2013) claimed LEO, or vigilantes, are more likely than victims to be open to online sexual behavior and requests to meet (Briggs et al., 2011; DeHart et al., 2017). Further, authors posit LEOs or vigilantes differ with respect to search criteria (DeHart et al., 2017), reaction to sexual comments (Briggs et al., 2011; Williams et al., 2013), specialized training (Briggs et al., 2011), overt explicitness of profiles (Briggs et al., 2011), and coercion (Chiang and Grant, 2019; Schneevogt et al., 2018). Finally, vigilantes and LEO are limited in what they can say to secure an arrest, as they are discouraged from initiating sexual conversation, contact, and arrangements of meetings (Gijn-Grosvenor and Lamb, 2016). As a result, LEOs and vigilantes may attempt to nudge conversations in this direction (Williams et al., 2013).

Finally, Briggs et al. (2011) highlighted the importance of the difference in goals between underage victims, vigilantes, and LEOs. Briggs et al. (2011) noted vigilantes and LEOs have a goal of collecting evidence and securing a quick arrest. This results in shorter chats with faster progressions and more pointed language than victim chats (Briggs et al., 2011). In LEO and vigilante chats, there was eagerness which is not reflective of either the distrust or language used by at-risk teens (Briggs et al., 2011).

3 Methodology

The annotation protocol described in the following sections follows the annotation pipeline proposed by Hovy and Lavid (2010). Based on research presented above, we hypothesize the greatest actionable differences occur within the grooming stages and grooming tactics actualized by underage victim, vigilante and LEO chats.

3.1 Corpus Composition: Data Subjects

A corpus is considered to be representative once it is *saturated* (Bowen, 2008), such that all relevant aspects of the phenomena are covered (Bowen, 2008; Hovy and Lavid, 2010) to the point of repetition or redundancy (Bowen, 2008). Per our assessment of online grooming literature in the 2000s, datasets in the predator domain consist of underage victims, LEOs, and online vigilantes. As data in the domain is difficult and time-consuming to acquire, we are unable to construct a corpus which would satisfy the saturation metric. However, we are able to construct a small corpus which represents all three participant types within the domain. We constructed a corpus of the three groups to ensure we account for all participants. Our corpus consists of 60 chat transcripts representing an equal number of participants from the three groups: 20 vigilantes, 20 underage victims, and 20 LEOs.

3.2 Data Acquisition and Cleaning

Our data collection and data use practices adhere to our Institutional Review Board (IRB) protocol. Due to the sensitive nature of the data, we submitted a full protocol review, which was approved. All texts in the corpus had all images and identifying information removed.

Underage victim and LEO transcripts were obtained from local, state, and federal agencies within the United States. While previous studies consisted of transcripts from multiple countries (Bergen et al., 2013; Quayle et al., 2014), we focused on transcripts from a single country in order to minimize confounding variables which could influence how participants interact.

The vigilante conversations are publicly available through pervertedjustice.com, an organization of adult vigilantes, trained to pose as minors online. The vigilantes speak with predators and work with LEOs to secure evidence for convictions. Oversight of these individuals is minimal (Williams et al., 2013) and there is little information on their website concerning the content of training for the vigilantes. The chat transcripts are posted to the Perverted Justice website following conviction.

One limitation we identified was incompleteness of the data. Transcripts from LEOs rarely included the full interaction between participants. Additionally, vigilante conversations would occasionally reference a phone call which took place without making the call transcript available. We also found that some chats ended early when LEOs had enough evidence. Other chats took place partway through the interaction. Still others terminated when the LEO switched to a different chat service.

Our mitigation for incomplete transcripts was to request background information from the law enforcement agencies. In the case of Perverted Justice transcripts, case documents and transcript summaries are often linked to the chats on the site. This can be a helpful resource for identifying limitations of individual transcripts.

3.3 Constructing the Initial Code Book

Tactics in our code book were deductively selected from grooming stages and tactics which map to the limitations discussed in Section 2.3.

The consensus from Section 2.3 was the grooming process would be affected by participant goals and investigator tactics (Bergen et al., 2013; Williams et al., 2013). To capture these differences, we considered all stages outlined by O'Connell (2003). The stages included *friendship forming*, *relationship forming*, *exclusivity*, *sexual*, *risk assessment*, and *meeting*. Additionally, as we annotated the stages, we noticed a possible divide between the *sexual stage* and a *non-consensual sexual stage* in which the victim, vigilante, or LEO indicated discomfort or declined sexual advances.

We selected the grooming tactics to operationalize the limitations enumerated in Section 2.3. The tactics are summarized in Table 1.

Openness to sexual comments/behavior. The *bragging* and *personal compliment* tactics are linked to sexual discussions within grooming (Barber and Bettez, 2014). Given LEOs respond to sexual content in a more positive manner than at-risk teens, we posit responses related to *bragging* and *personal compliments* will differ. Roleplay is how some predators act out fantasy, thus making it applicable to assessing openness to sexual behavior (Kloess et al., 2019). Finally, predators often ask for jarring and explicit details about the victim's sexual past (Aitken et al., 2018) which an at-risk teen might find uncomfortable.

Discussion of Meetings. *Willingness* is an assessment of what a participant would consider doing (Barber and Bettez, 2014). Often, this is associated with discussions of the explicit activities a victim would agree to perform in person.

Coercion. *Coercion* has been associated with differences between internet stings and victim conversations in the past, though they often point to more overt forms of coercion (Chiang and Grant, 2019; Schneevogt et al., 2018). We split the *coercion* tactic into *coercion* and *sexual violence* to reflect this distinction.

Naïve and Young/Explicitness of Profile. Vulnerabilities are used by predators to evoke sympathy (Barber and Bettez, 2014). The expression of vulnerabilities by a non-predator may be a good indicator that a sting is taking place, as we hypothesize LEOs and vigilantes will over-emphasize childhood problems. Further, willingness to send images in chat to a stranger could also be an operationalization of naïvety to online dangers. We posit LEOs and vigilantes are more likely to send such pictures than real at-risk minors.

Initial Contact. While we could not find any grooming tactics related to initial contact within the literature, we posit differences in initial contact can be captured by the grooming stages.

Grooming Characteristics	Grooming Tactics
Openness to sexual comments/behavior	Roleplay Bragging Sexual History Personal Compliments
Discussion of Meetings	Willingness
Coercion	Coercion Sexual Violence
Naïve/Explicitness of Profile	Discussion of Images Vulnerabilities (Neg. Life Stories, Neg. Physical Attributes)
Initial Contact	Grooming Stages
Arrest Goal	Age Difference Reverse Power
Speed/Duration	Media Progression (Phone calls, Video Chat)

Table 1: Mapping of grooming characteristics to grooming tactics in this paper.

Arrest Goal. We focused on what an LEO would need to show that a law was broken. Stating the *age difference* would indicate the predator knew the victim was under 18. *Reverse power* is giving the control of the situation to the other participant (Barber and Bettez, 2014). We hypothesize there may be differences with respect to *reverse power* because the vigilantes and LEOs are unable to initiate certain discussions (Gijn-Grosvenor and Lamb, 2016). *Reverse power* may be a way to influence the discussion without explicitly initiating.

Speed/Duration. In typical grooming conversations, participants often switch chatting services multiple times (Quayle et al., 2014). If conversations are shorter and faster in internet stings, we posit there will be fewer forms of communication.

In addition to definitions for each of the tactics and stages from literature, code books often include examples for each tactic (Bada et al., 2012; Kingsbury et al., 2002; Stoyanov and Cardie, 2008) which help to clarify the tactic without over-specifying (Hovy and Lavid, 2010). In the code book we included an initial set of examples for each tactic from the psychology literature (2.1).

3.4 Annotator Selection

Grooming stages and tactics are difficult to annotate, even for those experienced within the domain (Gillespie, 2002). Recommendations of whether to use domain experts or novices differs within NLP literature (Hovy and Lavid, 2010). We used two annotators: the first annotator is an expert within the child exploitation domain while the second annotator was an undergraduate with low levels of experience. Due to the multidisciplinary nature of this domain, we decided to use a non-expert for the second annotator to ensure our corpus was accessible to researchers with multiple backgrounds.

3.5 Annotator Training

Hovy and Lavid (2010) recommends providing a reasonable amount of training in order to familiarize the annotators with the task without overspecifying the task. To accomplish this, the first annotator gave the novice second annotator a short overview of the child exploitation domain to help bridge the gap between skill sets to improve alignment (Bayerl and Paul, 2011).

The first annotator also provided the second annotator with the initial code book to review. The second annotator was able to ask clarifying questions about the grooming stages and tactics in the code book.

Following familiarization with the codebook, both annotators annotated a series of three training chats. Since grooming participants do not use every single grooming stage or tactic in chats, but rather a subset that fits their goals (Beauregard et al., 2012; Briggs et al., 2011; Kloess et al., 2017), annotating three chats ensured exposure of the second annotator to all of the stages and tactics in the code book.

Simple agreement was used to calculate a baseline of agreement on which to measure improvement following training; the target agreement for this study was 80% (Krippendorff, 2011). Per best practices, the annotators met to discuss the training round, answer questions from the second annotator, and resolve inter-annotator disagreements (Hovy and Lavid, 2010).

3.6 Annotation Procedures

3.6.1 Coding Rounds

We initially had six grooming stages and 14 grooming tactics to annotate. Due to the expected complexity of these tactics and stages, we originally split the annotation task into three rounds. Following the training round we re-split the task into four rounds to reduce annotator fatigue in an annotation session (Bayerl et al., 2003).

One recommendation from this is to keep the grain-size of stages or tactics consistent within a single annotation round. In the first round of annotation we had a mixture of stages and tactics. However, this made annotation difficult for two reasons. The first reason was that grooming stages encompassed grooming tactics. For instance, within the *sexual stage* a predator might use *willingness* as a means to gauge interest and *roleplay* to normalize sexual content. We found the difference in granularity between grooming stages and tactics made it difficult to context switch within the same round. The second reason was that grooming stages were complex and impermanent (Gillespie, 2002). There were multiple examples and rules to remember with each grooming stage and changes between grooming stage were often quick and overlapped, making the annotation task complicated. Keeping the number of tactics low made it easier to focus on the stages.

3.6.2 Code Span Identification

The transient nature of grooming makes the beginning and end of a grooming instance difficult to identify (Gillespie, 2002). Known as *code span*, challenges related to identification of the boundaries of an annotation are known issues within annotation literature (Bada et al., 2012; Stoyanov and Cardie, 2008). To mitigate code span issues, we allow partial membership to a tactic or stage to facilitate discussion and identify span issues.

For this study, we defined *partial membership* as (i) non-substantive statements made in reference to a tactic or (ii) lines which do not meet the criteria for a tactic but result in or from the tactic. Full membership was defined as a line which was representative of a tactic. For instance, if a victim indicates their parents will be gone for the weekend and the predator responds by asking if they would like to meet, we would annotate the statements about the meeting as having full membership to the *meeting stage*. We would also annotate the victim's line about their parents as partial membership because the message precipitated the meeting discussion.

Based on annotating the corpora, we did not find a noticeable difference in the code span of topics between victim, vigilante, and LEO transcripts. However, we did notice that for longer spans of sexual tactics such as the *sexual stage*, *sexual history*, and *roleplay*, the intensity and graphic content of the messages would increase. We posit this was due to the predators attempting to gradually sexualize the conversation. Given that LEO and vigilantes respond in a more positive manner that real victims (Briggs et al., 2011; DeHart et al., 2017; Williams et al., 2013), predators may use more graphic and descriptive language when talking to vigilantes and LEOs than real victims. Additionally, we noticed real victims would end uncomfortable conversations more quickly and with more firm language (e.g., "no" versus "idk"). Future research should examine the change in graphical descriptions and responses to them over time in transcripts.

3.6.3 Annotator Protocol and Process

We developed the following guidelines for the annotation process:

- Review the code book before each annotation session.
- Annotation questions should be noted at the time, and discussed between rounds.
- Questions should be directed towards the first annotator and not external works or researchers. Hovy and Lavid (2010) emphasized the issues which could arise from inexperienced annotators not knowing the correct resource to ask for assistance. We mitigate this by having the first annotator, with domain experience, be the sole source of training.
- Annotators should label each round independently; annotations do not depend on previous rounds. We added this rule following the training session because the non-expert annotator began to second-guess tactics and annotations they felt should be related to one another.
- Coding session duration should be limited to reduce fatigue (Bayerl et al., 2003). We limited coding sessions to a maximum of three rounds in an annotation session.
- Each line may be annotated for multiple tactics within the coding round. While limiting annotations to one per tactic per round is desirable (Bada et al., 2012), the complex and transient nature of grooming results in multiple tactics and stages manifesting at the same time (Gillespie, 2002).

Using these guidelines and the finalized code book, both annotators annotated six chat transcripts, which represented 10% of the total number of transcripts. Simple agreement was calculated and found to be greater than the recommended 80% for annotation tasks (Krippendorff, 2011). For the

Group	Total Lines	*M*	*SD*
Vigilante	28505	1425.25	1111.96
Victim	10469	523.45	742.74
LEO	5140	257.00	302.56

Table 2: Summary of corpus composition.

simple agreement calculation, we used the total agreed upon annotations out of the total annotations. For this study, we only took into account full membership annotations as partial membership annotations were created to facilitate discussion. Future work, which incorporates the partial membership annotations into analysis, would require additional agreement measures.

Differences following the agreement calculation were each discussed and resolved through verbal agreement.

Finally, the first annotator annotated the remaining 51 chat conversations using the finalized code book and above guidelines. Future work should use two or greater annotators for the duration of annotation (Bayerl and Paul, 2011). While using a single annotator for the remainder of the annotation is a limitation of this study, we sought to mitigate the limitation by constructing and testing the finalized code book using both annotators.

3.6.4 Comparing Victims, Vigilante, and LEOs

Quantitative Analysis

To compare chat length, we used a one way analysis of variance between the three groups. Levene's F Test indicated unequal variance (F= $10.82, p < .001$). As a result, the Welch Test was performed and showed a significant effect of participant type on length of chat, *Welch's* $F(2, 57) = 10.587, p < .001$, $\omega = .52$. We performed Post-hoc analysis using the Games-Howell test due to the unequal variance. The Games-Howell test indicated the number of lines for a vigilante chat were significantly different than the number of lines for LEO ($p < .001$) and underage victims ($p = .013$).

. A χ^2 test was conducted to examine the presence or lack of each tactic within chats in the three groups. While the majority of chats contained all stages, future work should assess differences in presence and sequencing of grooming stages as well.

When assumptions of a χ^2 test could not be met, a Fisher's Exact test was conducted. Through χ^2 tests, we found *sexual history* ($\chi^2 = 10.40, df = 2, p = .006$), *willingness* ($\chi^2 = 14.55, df = 2, p = .001$), *discussion of phone calls* ($\chi^2 = 10.15, df = 2, p = .006$), and acknowledgement of *age difference* ($\chi^2 = 22.67, df = 2, p < .001$) were significantly different between participant groups. Through Fisher's Exact tests, we found the use of *personal compliments* ($p = 0.011$) and *reverse power* ($p = 0.03$) were significantly different between participant groups. There were no significant differences between the three groups with respect to *coercion*, *discussion of images*, *bragging*,*discussion of video chatting*, *negative physical traits*, *negative life stories*, *roleplay*, or *sexual violence*.

	Vigilante	Victim	LE
Sexual History	95.0%	50.0%	75.0%
Willingness	85.0%	55.0%	25.0%
Phone Calls	75.0%	55.0%	25.0%
Age Difference	95.0%	35.0%	90.0%
Compliments	100.0%	65.0%	83.3%
Reverse Power	100.0%	75%	70%

Table 3: Occurrence of grooming tactics in vigilante, victim, and LEO conversations.

The results of the χ^2 and Fisher's Exact tests are summarized in Table 3.

Within the three groups, grooming tactics were used the most in vigilante conversations. This was consistent for all tactics in Table 3. Most conversations with vigilantes included the use of all of the tactics in Table 3. *Willingness*, *discussion of phone calls*, and *reverse power* were used the least in LEO chats. *Sexual history*, *age difference*, and *compliments* were used the least in victim conversations.

LEO conversations tended to be short and often included a shift to talking in another app. Additionally, LEO responses to direct questions about meeting and sexual activities tended to be vague in comparison to vigilantes and real victims. The vagueness of the LEO responses may have resulted in the predator adjusting tactics.

We posit age difference and sexual history may be used less frequently in victim conversations because the victims often already knew the predator. The sexual history of the victim may already be known by the predator. This is likely the case with age difference as well. The victims and predators often appeared aware of the age gap but would not discuss it. In vigilante and LEO conversations, the conversations were with strangers. The participants would often give their age, sex, and location towards the beginning of the chat.

We originally posited LEO and vigilante conversations would be similar. Both vigilante and LEO groups consist of adults posing as children. Additionally, both groups receive some level of dedicated training on identifying predators. However, from Table 3 we see the groups differ in the tactics used. Some of these differences may be the result of the motivation of the LEO versus the vigilante and some may be a result of the differences in training. In the next section, we will discuss our qualitative observations which may contribute to differences in both how and how often the grooming tactics were used.

Qualitative Analysis

In addition to the analysis of chat length and tactic usage, we also qualitatively noted, and aggregated, annotator observations on language and conversational differences between the three groups. The differences we found between real victims and internet stings were the result of the LEOs and vigilantes attempting to make the predator state intentions explicitly and avoid sexual situations or meetings.

In many cases, individual lines taken out of context would appear innocuous. For instance, it is common for a predator to ask for images of a victim to get to know them and determine level of attraction. However, based on this exchange alone, it would not be possible to determine whether or not one of the participants were a predator. In these cases, LEOs and vigilantes would often ask clarifying questions to determine whether the request was for sexual or non-sexual pictures. At times, the LEOs or vigilantes would go as far as to describe the clothing they were wearing, such as making references to their pajamas, when the predator asked for pictures. We observed this as a priming technique to prompt the user to ask for more graphic images. Additionally, we found real victims were more likely to provide images while vigilantes and LEOs tended to ask clarifying questions around image requests.

Furthermore, it is also common for a predator to request images when attempting to determine if the other participant is an LEO instead of a minor (Kloess et al., 2019). In some cases, the predator will demand pictures taken immediately to ensure the participant is the person in the image.

From the chats we annotated, this appeared to occur more frequently in predator and vigilante chats than in underage victim chats. We hypothesize that the LEOs or vigilantes were being forward in a manner that sparked suspicion in the predator. Additionally, this was often triggered by another tactic in which the officer or vigilante would say they lied about their age in the profile and were actually younger. Statements like this did not occur in victim chats and seemed to cause suspicion and trigger risk assessment questions from the predator.

We also found direct and indirect communication styles used by the predator affected the three participant groups differently. Predators would often use expressions of vagueness about intentions to avoid explicitly stating an interest in sex. Further, we saw examples in which predators would not refer to sexual body parts or sexual acts directly, but instead would use euphemisms.

LEO and vigilantes would respond to vague responses about meetings, sexual intentions, and euphemisms by repetitively asking the predator what they meant or what they would do together. Additionally, the LEOs and vigilantes would act naïve and ask the predator to explain obvious references to body parts or sexual innuendos. We did not see this within underage victim chats. In real victim chats, the predators appeared to do the majority of the prompting related to sexual topics. We did not see much evidence of victims asking predators what they would do when they met. However, this was present in most vigilante and officer chats where meetings were discussed.

The greatest disparities between vigilantes and LEOs were within the *willingness* and *phone call* tactics. *willingness* to assess whether or not the vigilantes would engage in specific sexual activities during a physical meeting. In some instances, predators would also use *willingness* to get the other participant to agree to follow their instructions. We posit the difference in occurrence of the *willingness* tactic may be due to the responses given. Vigilantes and victims appear to respond more positively to *willingness* questions whereas LEOs tend to respond more vaguely. This may lead the predator to adjust tactics when speaking to LEOs.

Finally, LEOs and vigilantes used contrived situations to avoid undesired interactions or unplanned meetings with a predator. Such interactions included roleplaying, sending sexual images, or watching sexual activity on a webcam. Examples of avoidances used for undesirable interactions included an angry parent, a parent in the room, homework, a broken phone or camera, a broken internet connection, tiredness, or plans with friends. Avoidances of meetings were generally framed around other events preventing the meeting. For instance, if a predator wanted to meet the vigilante or LEO but the participant was not ready, the vigilante or LEO would claim to have family plans that were unavoidable. Avoidances were rarely used by underage victims.

3.7 Iterative Changes to Corpus

Neutering is a common strategy to handle disagreement and combine overlapping tactics, and refers to collapsing two or more tactics together in a code book (Hovy and Lavid, 2010). During the annotation process, we chose to neuter two sets of tactics: sexual stage and non-sexual stage; and relationship forming and exclusivity.

As described in 3.5, we treated the *sexual stage* and the *non-consensual sexual stages* as separate stages during annotator training. Following training, we neutered the stages into a single stage as the annotators often could not agree on what constituted an implied denial of an advance. Future research should investigate the possible presence of a *non-consensual sexual stage* where the victim implicitly or explicitly scorns advances.

Given the similarity between *relationship forming* and *exclusivity*, we also neutered these stages into a single stage. *Relationship forming* and *exclusivity* revolve around the construction of foundational trust between the predator and the victim (O'Connell, 2003). While relationship forming can be thought of as the day-to-day interactions, exclusivity revolves around language to intensify the relationship and isolate the victim from their support system (O'Connell, 2003).

In addition to neutering tactics within the code book, we added examples for each tactic. The original stages and tactics in the literature were designed from the perspective of the predator (Barber and Bettez, 2014), kloess2017qualitative, O'Connell. We annotated each of the tactics for all participants. We found the way LEOs, vigilantes, and underage victims used and responded to tactics differed from the predator. For instance, the predators used references to *age difference* to determine the level of comfort of the victim while LEOs used *age difference* to ensure the predator explicitly acknowledged the illegality of the solicitation. Further, underage victims would sometimes reference the age gap as a negative trait of themselves, almost as an insecurity. Having examples of uses of the tactics and stages by multiple types of participants helped the second annotator to generalize the tactics to all participants and not just predators.

4 Conclusion

NLP has contributed to research in the child exploitation domain by developing automated systems for detection of predators (Kim et al., 2020; McGhee et al., 2011; Zuo et al., 2018) and participants (Pendar, 2007). However, the corpora used for training the models are not representative of the criminals or victims involved in actual child exploitation cases (Bergen et al., 2013; Briggs et al., 2011; Chiang and Grant, 2019; DeHart et al., 2017; Gijn-Grosvenor and Lamb, 2016; Mitchell et al., 2005; Schneevogt et al., 2018; Winters et al., 2017). We offered an overview of the problems within the NLP corpora in the domain. We also discussed the impact these problems have on representing the online grooming process. Finally, we provided our methodology and recommendations from annotating a corpus of underage victim, vigilante, and LEO conversations and showed that there are statistical differences between the three groups. While NLP research within the child exploitation domain appears to be expanding, there is a need to ensure that corpora are designed and annotated in such a way that it contributes beneficial solutions.

Acknowledgments

We would like to thank the reviewers for their feedback and recommendations. Additionally, we would like to thank Lin Sun Fa for his assistance as the second annotator for this project.

References

Susan Aitken, Danielle Gaskell, and Alan Hodkinson. 2018. Online sexual grooming: exploratory comparison of themes arising from male offenders' communications with male victims compared to female victims. *Deviant Behavior*, 39(9):1170–1190.

Marwan Albahar and Jameel Almalki. 2019. Deepfakes: Threats and countermeasures systematic review. *Journal of Theoretical and Applied Information Technology*, 97(22):3242–3250.

Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A. Baumgartner, K. Bretonnel Cohen, Karin Verspoor, Judith A. Blake, and et al. 2012. Concept annotation in the craft corpus. *BMC Bioinformatics*, 13(1):161.

Connie Barber and Silvia Bettez. 2014. Deconstructing the online grooming of youth: Toward improved information systems for detection of online sexual predators.

Petra S Bayerl, Harald Lüngen, Ulrike Gut, and Karsten I Paul. 2003. Methodology for reliable schema development and evaluation of manual annotations. In *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (K-CAP 2003)*.

Petra Saskia Bayerl and Karsten Ingmar Paul. 2011. What determines inter-coder agreement in manual annotations? a meta-analytic investigation. *Computational Linguistics*, 37(4):699–725.

Eric Beauregard, Benoit Leclerc, and Patrick Lussier. 2012. Decision making in the crime commission process: Comparing rapists, child molesters, and victim-crossover sex offenders. *Criminal Justice and Behavior*, 39(10):1275–1295.

Emilia Bergen, Jan Antfolk, Patrick Jern, Katarina Alanko, and Pekka Santtila. 2013. Adults' sexual interest in children and adolescents online: A quasi-experimental study. *International Journal of Cyber Criminology*, 7(2).

Pamela J Black, Melissa Wollis, Michael Woodworth, and Jeffrey T Hancock. 2015. A linguistic analysis of grooming strategies of online child sex offenders: Implications for our understanding of predatory sexual behavior in an increasingly computer-mediated world. *Child abuse & neglect*, 44:140–149.

Dasha Bogdanova, Paolo Rosso, and Thamar Solorio. 2012. On the impact of sentiment and emotion based features in detecting online sexual predators. In *Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis*, pages 110–118.

Glenn A Bowen. 2008. Naturalistic inquiry and the saturation concept: a research note. *Qualitative research*, 8(1):137–152.

Peter Briggs, Walter T Simon, and Stacy Simonsen. 2011. An exploratory study of internet-initiated sexual offenses and the chat room sex offender: Has the internet enabled a new typology of sex offender? *Sexual Abuse*, 23(1):72–91.

Amparo Elizabeth Cano, Miriam Fernandez, and Harith Alani. 2014. Detecting child grooming behaviour patterns on social media. In *International conference on social informatics*, pages 412–427. Springer.

Emily Chiang and Tim Grant. 2019. Deceptive identity performance: offender moves and multiple identities in online child abuse conversations. *Applied Linguistics*, 40(4):675–698.

Matthew Colton, Susan Roberts, and Maurice Vanstone. 2010. Sexual abuse by men who work with children. *Journal of child sexual abuse*, 19(3):345–364.

Dana DeHart, Gregg Dwyer, Michael C Seto, Robert Moran, Elizabeth Letourneau, and Donna Schwarz-Watts. 2017. Internet sexual solicitation of children: a proposed typology of offenders based on their chats, e-mails, and social network posts. *Journal of Sexual Aggression*, 23(1):77–89.

Maria Diaz and Anand Panangadan. 2020. Natural language-based integration of online review datasets for identification of sex trafficking businesses. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 259–264. IEEE.

Mohammadreza Ebrahimi, Ching Y Suen, and Olga Ormandjieva. 2016. Detecting predatory conversations in social media by deep convolutional neural networks. *Digital Investigation*, 18:33–49.

Evianne L. van Gijn-Grosvenor and Michael E. Lamb. 2016. Behavioural differences between online sexual groomers approaching boys and girls. *Journal of Child Sexual Abuse*, 25(5):577–596.

Alisdair A Gillespie. 2002. Child protection on the internet-challenges for criminal law. *Child & Fam. LQ*, 14:411.

Fergyanto E Gunawan, Livia Ashianti, Sevenpri Candra, and Benfano Soewito. 2016. Detecting online child grooming conversation. In *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, pages 1–6. IEEE.

Aditi Gupta, Ponnurangam Kumaraguru, and Ashish Sureka. 2012. Characterizing pedophile conversations on the internet using online grooming.

Eduard Hovy and Julia Lavid. 2010. Towards a 'science' of corpus annotation: A new methodological challenge for corpus linguistics. page 25.

Giacomo Inches and Fabio Crestani. 2012. Overview of the international sexual predator identification competition at pan-2012. In *CLEF (Online working notes/labs/workshop)*, volume 30.

Sharon Jackson, Elinor Newall, and Kathryn Backett-Milburn. 2015. Children's narratives of sexual abuse. *Child & Family Social Work*, 20(3):322–332.

Carmit Katz. 2013. Internet-related child sexual abuse: What children tell us in their testimonies. *Children and Youth Services Review*, 35(9):1536–1542.

Keith L Kaufman, Daniel R Hilliker, Patty Lathrop, and Eric L Daleiden. 1993. Assessing child sexual offenders' modus operandi: Accuracy in self-reported use of threats and coercion. *Annals of Sex Research*, 6(3):213–229.

Jinhwa Kim, Yoon Jo Kim, Mitra Behzadi, and Ian G Harris. 2020. Analysis of online conversations to detect cyberpredators using recurrent neural networks. In *Proceedings for the First International Workshop on Social Threats in Online Conversations: Understanding and Management*, pages 15–20.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the penn treebank. In *Human Language Technology Conference*.

Juliane A Kloess, Catherine E Hamilton-Giachritsis, and Anthony R Beech. 2019. Offense processes of online sexual grooming and abuse of children via internet communication platforms. *Sexual Abuse*, 31(1):73–96.

Juliane A Kloess, Sarah Seymour-Smith, Catherine E Hamilton-Giachritsis, Matthew L Long, David Shipley, and Anthony R Beech. 2017. A qualitative analysis of offenders' modus operandi in sexually exploitative interactions with children online. *Sexual Abuse*, 29(6):563–591.

Kamil Kopeckỳ. 2017. Online blackmail of czech children focused on so-called "sextortion"(analysis of culprit and victim behaviors). *Telematics and Informatics*, 34(1):11–19.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Reuben A Lang and Roy R Frenzel. 1988. How sex offenders lure children. *Annals of Sex Research*, 1(2):303–317.

Benoit Leclerc and Pierre Tremblay. 2007. Strategic behavior in adolescent sexual offenses against children: Linking modus operandi to sexual behaviors. *Sexual Abuse: A Journal of Research and Treatment*, 19(1):23–41.

L Alvin Malesky Jr. 2007. Predatory online behavior: Modus operandi of convicted sex offenders in identifying potential victims and contacting minors over the internet. *Journal of child sexual abuse*, 16(2):23–32.

India McGhee, Jennifer Bayzick, April Kontostathis, Lynne Edwards, Alexandra McBride, and Emma Jakubowski. 2011. Learning to identify internet sexual predation. *International Journal of Electronic Commerce*, 15(3):103–122.

Kanishka Misra, Hemanth Devarapalli, Tatiana R Ringenberg, and Julia Taylor Rayz. 2019. Authorship analysis of online predatory conversations using character level convolution neural networks. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 623–628. IEEE.

Kimberly J Mitchell, Janis Wolak, and David Finkelhor. 2005. Police posing as juveniles online to catch sex offenders: Is it working? *Sexual Abuse: A Journal of Research and Treatment*, 17(3):241–267.

Rachel O'Connell. 2003. A typology of child cybersexploitation and online grooming practices. *Cyberspace Research Unit, University of Central Lancashire*.

Nick Pendar. 2007. Toward spotting the pedophile telling victim from predator in text chats. In *International Conference on Semantic Computing (ICSC 2007)*, pages 235–241. IEEE.

Ethel Quayle, Silvia Allegro, Linda Hutton, Michael Sheath, and Lars Lööf. 2014. Rapid skill acquisition and online sexual grooming of children. *Computers in Human Behavior*, 39:368–375.

Tatiana R Ringenberg, Kanishka Misra, and Julia Taylor Rayz. 2019. Not so cute but fuzzy: Estimating risk of sexual predation in online conversations. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2946–2951. IEEE.

Patricia de Santisteban, Joana del Hoyo, Miguel Ángel Alcázar-Córcoles, and Manuel Gámez-Guadix. 2018. Progression, maintenance, and feedback of online child sexual grooming: A qualitative analysis of online predators. *Child Abuse Neglect*, 80:203–215.

Daniela Schneevogt, Emily Chiang, and Tim Grant. 2018. Do perverted justice chat logs contain examples of overt persuasion and sexual extortion? a research note responding to chiang and grant (2017, 2018). *Language and Law*, 5(1):97–102.

David Shannon. 2008. Online sexual grooming in sweden—online and offline sex offences against children as described in swedish police data. *Journal of Scandinavian Studies in Criminology and Crime Prevention*, 9(2):160–180.

Manfred Stede and Chu-Ren Huang. 2012. Interoperability and reusability: the science of annotation. *Language Resources and Evaluation*, 46(1):91–94.

Veselin Stoyanov and Claire Cardie. 2008. Annotating topics of opinions. page 6.

Carolina Villacampa and Mª Jesus Gómez. 2017. Online child sexual grooming: Empirical findings on victimisation and perspectives on legal requirements. *International Review of Victimology*, 23(2):105–121.

Helen Whittle, Catherine Hamilton-Giachritsis, Anthony Beech, and Guy Collings. 2013. A review of online grooming: Characteristics and concerns. *Aggression and violent behavior*, 18(1):62–70.

Helen C Whittle, Catherine E Hamilton-Giachritsis, and Anthony R Beech. 2014. “under his spell”: Victims’ perspectives of being groomed online. *Social Sciences*, 3(3):404–426.

Rebecca Williams, Ian A Elliott, and Anthony R Beech. 2013. Identifying sexual grooming themes used by internet sex offenders. *Deviant behavior*, 34(2):135–152.

Georgia M Winters, Leah E Kaylor, and Elizabeth L Jeglic. 2017. Sexual offenders contacting children online: an examination of transcripts of sexual grooming. *Journal of Sexual Aggression*, 23(1):62–76.

Richard Wortley, Benoit Leclerc, Danielle M Reynald, and Stephen Smallbone. 2019. What deters child sex offenders? a comparison between completed and noncompleted offenses. *Journal of interpersonal violence*, 34(20):4303–4327.

Zheming Zuo, Jie Li, Philip Anderson, Longzhi Yang, and Nitin Naik. 2018. Grooming detection using fuzzy-rough feature selection and text classification. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE.

Detecting Domain Polarity-Changes of Words in a Sentiment Lexicon

Shuai Wang[†][*], **Guangyi Lv**[§], **Sahisnu Mazumder**[‡], **Bing Liu**[‡]
[‡]Department of Computer Science, University of Illinois at Chicago, USA
[§]University of Science and Technology of China, Hefei, China
[†]Amazon AI
shuaiwanghk@gmail.com, gylv@mail.ustc.edu.cn
sahisnumazumder@gmail.com, liub@uic.edu

Abstract

Sentiment lexicons are instrumental for sentiment analysis. One can use a set of sentiment words provided in a sentiment lexicon and a lexicon-based classifier to perform sentiment analysis. One major issue with this approach is that many sentiment words (from the lexicon) are domain dependent. That is, they may be positive in some domains but negative in some others. We refer to this problem as *domain polarity-changes of words* from a sentiment lexicon. Detecting such words and correcting their sentiment for an application domain is very important. In this paper, we propose a graph-based technique to tackle this problem. Experimental results show its effectiveness on multiple datasets from different domains.

1 Introduction

Sentiment words, also called opinion/polar words, are words that convey positive or negative sentiments (Pang and Lee, 2008). Such sentiment-bearing words are usually pre-compiled as word lists in a sentiment lexicon, which is instrumental as well as an important linguistic resource to sentiment analysis (Liu, 2012). So far, numerous studies about how to construct lexicons have been reported, which will be discussed in Section 2.

Despite the fact that there is extensive research on lexicon construction, limited work has been done to solve the problem of identifying and handling sentiment words in a given/constructed lexicon that have domain-dependent polarities. In real-life applications, there are almost always some sentiment words that express sentiments different from their default polarities provided in a general-purpose sentiment lexicon. For example, in the lexicon compiled by Hu and Liu (2004), the word "crush" is associated with a negative sentiment, but it actually shows a positive opinion in domain *blender* because "crush" indicates that a blender works well, e.g., in the sentence "it does crush the ice!". We call this problem *domain polarity-change* of a word in a sentiment lexicon.

The polarity change of words plays a crucial role in sentiment classification. As we will see in the experiment section, without identifying and correcting such domain dependent sentiment words, the performance of sentiment classification could be much poorer. Although some researchers have studied the domain-specific sentiment problem with lexicons, their focuses are quite different and their approaches are not suitable for our task. We will discuss them further in the following sections.

Regarding sentiment classification, it is important to note that our work mainly aims to help *lexicon-based approaches* (Taboada et al., 2011). It does not directly help machine-learning (ML) or supervised learning approaches (Zhang et al., 2018) because the domain-dependent polarities of words are already reflected in the manually labeled training data. Notice that for those ML approaches, the manual annotation for each application domain is required, which is a time-consuming and labor-intensive task, and is thus hard to scale up. In many real-world scenarios, lexicon-based approaches are useful and could be a better alternative (Liu, 2012).

However, to effectively apply a sentiment lexicon to an application domain, the domain-polarity change problem discussed above needs to be addressed. To this end, we propose a graph-based approach named Domain-specific Sentiment Graph (DSG) in Section 4. It works with three steps: (domain) sentiment word collection, (domain) sentiment correlation extraction, and graph construction and inference. Experimental results show its effectiveness in detecting domain polarity-changed words on multiple datasets. We will also see with the detection of those words, a huge performance gain can be achieved in sentiment classification.

---

[*] Work done while at University of Illinois at Chicago

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3657–3668
August 1–6, 2021.

2 Related Work

This work concerns domain polarity changes of words in lexicons. So we first discuss the works related to sentiment lexicons, and then domain sentiment, and finally domain sentiment with lexicons.

Extensive studies have been done for sentiment lexicons and the majority of them focus on lexicon construction. These approaches can be generally categorized as dictionary-based and corpus-based.

Dictionary-based approaches first used some sentiment seed words to bootstrap based on the synonym and antonym structure of a dictionary (Hu and Liu, 2004; Valitutti et al., 2004). Later on, more sophisticated methods were proposed (Kim and Hovy, 2004; Esuli and Sebastiani, 2005; Takamura et al., 2007; Blair-Goldensohn et al., 2008; Rao and Ravichandran, 2009; Mohammad et al., 2009; Hassan and Radev, 2010; Dragut et al., 2010; Xu et al., 2010; Peng and Park, 2011; Gatti and Guerini, 2012; San Vicente et al., 2014). Corpus-based approaches build lexicons by discovering sentiment words in a large corpus. The first idea is to exploit some coordinating conjunctions (Hatzivassiloglou and McKeown, 1997; Hassan and Radev, 2010). Kanayama and Nasukawa (2006) extended this approach by introducing inter-sentential sentiment consistency. Other related work includes (Kamps et al., 2004; Kaji and Kitsuregawa, 2006; Wang et al., 2017). The second idea is to use syntactic relations between opinion and aspect words (Zhuang et al., 2006; Wang and Wang, 2008; Qiu et al., 2011; Volkova et al., 2013). The third idea is to use word co-occurrences for lexicon induction (Turney and Littman, 2003; Igo and Riloff, 2009; Velikovich et al., 2010; Yang et al., 2014; Rothe et al., 2016).

However, our work is very different as we focus on detecting domain dependent sentiment words in a given general-purpose sentiment lexicon.

Also related is the existing research about domain and context dependent sentiment. First, despite the fact that several researchers have studied context dependent sentiment words, which are based on sentences and topic/aspect context (Wilson et al., 2005; Ding et al., 2008; Choi and Cardie, 2008; Wu and Wen, 2010; Jijkoun et al., 2010; Lu et al., 2011; Zhao et al., 2012; Kessler and Schütze, 2012; Teng et al., 2016; Wang et al., 2016, 2018a,b; Li et al., 2018a), our work is based on domains. Second, while the studies on transfer learning or domain adaptation for sentiment analysis deal with domain information (Bhatt et al., 2015; Yu and Jiang, 2016; Li et al., 2018b), our work does not lie in this direction. We do not have any source domain and our goal is not to transfer domain knowledge to another domain. Third, most importantly, the above works are either irrelevant to lexicons or not for detecting the sentiment discrepancy between a lexicon and the application domain.

Our work is most related to the following studies that involve both sentiment lexicons and domain sentiment problems. Choi and Cardie (2009) adapted the word-level polarities of a general-purpose sentiment lexicon to a particular domain by utilizing the expression-level polarities in that domain. However, their work targeted at reasoning the sentiment polarities of multi-word expressions. It does not detect or revise the sentiment polarities of individual words in the lexicon for a particular domain, and hence, cannot solve our problem. Du et al. (2010) studied the problem of adapting the sentiment lexicon from one domain to another domain. It further assumes that the source domain has a set of sentiment-labeled reviews. Their technique is therefore more about transfer learning and their learning settings differ from ours intrinsically. (Ortiz, 2017) designed a sentiment analysis application that allows plugin lexicons (if users can provide them) to help predict domain sentiment. It neither detects nor corrects domain polarity-changed words. Perhaps, the most related work is (Hamilton et al., 2016), which uses seed lexicon words, word embeddings, and random walk to generate a domain-specific lexicon. However, their model is for lexicon construction in essence, by (its capability of) functioning on a domain-oriented corpus. It does not aim to detect/change the sentiment polarity from a given lexicon. It is thus not directly applicable to our task. To make it workable for our task, we design a two-step approach, which will be detailed in the experiment section (Section 5).

To the best of our knowledge, this is the first study to detect domain polarity-changes of words in a sentiment lexicon. We will further discuss why this task is important and useful in Section 5.7.

3 Problem Definition

Given a general-purpose sentiment lexicon L (which contains sentiment words and their default polarities) and an application domain review corpus D, the goal (or the task of detecting domain polarity changes of words) is to identify a subset of words in L that carry different sentiment polarities in that domain (different from their default polarities), which we call *domain polarity-changed (lexical*[1]*) words* and denote them as C ($C \subseteq L$).

4 Proposed Solution

To tackle the above problem, we propose a graph-based learning approach named Domain-specific Sentiment Graph (DSG). It works with three major steps: (1) (domain) sentiment word collection, (2) (domain) sentiment correlation extraction, and (3) graph construction and inference.

Specifically, it first collects a set of mentioned sentiment words S ($L \subseteq S$) in the domain corpus D. It then mines multiple types of relationships among sentiment words in S, which are denoted as a relationship set R. The relationships are identified based on different types of linguistic connectivity. Next, it builds a probabilistic graph with each node representing a sentiment word in S and each edge representing a relation (from R) between two words. An inference method is then applied to re-estimate the domain-specific polarities (or beliefs) of sentiment words. With the re-estimated beliefs obtained in the application domain, those sentiment words with changed polarities can be detected, by measuring the sentiment shift of a lexical word between its induced (in-domain) sentiment belief and its original (lexicon-based) polarity.

In this learning manner, the proposed approach requires no prior knowledge or annotated data for a particular domain. It is thus applicable to multiple/different domains. Intuitively, this approach works based on two assumptions:

Assumption 1: Sentiment Consistency (Abelson, 1983; Liu, 2012): a sentiment expression tends to be sentimentally coherent with its context. Notice that sentiment consistency can be reflected in multiple types of conjunction like "and", "or", etc., which will be explained in Section 4.2. In fact, this assumption is common in sentiment analysis and has been used in many studies (Kanayama and Nasukawa, 2006; Hassan and Radev, 2010)

Assumption 2: The number of domain polarity-changed lexical words[1] is much smaller than the number of those (words) whose polarities do not change. This assumption ensures that we can rely on the general-purpose lexicon itself for detection[1]. In other words, the real polarity of a sentiment word in a certain domain can be distilled by its connections with other (mostly polarity-unchanged) words whose polarities are known from the lexicon.

4.1 Sentiment Word Collection

As the first step, DSG collects sentiment words in an application domain corpus, including the sentiment words not present in a lexicon. Specifically, we consider three types of (likely) sentiment words: (1) The word appears in a given lexicon. (2) The word is an adjective in the corpus. (3) The word is an adverb in the corpus and has an adjective form.

We simply accept all lexical words and adjective words as (likely) sentiment words, which does not cause serious problems in our experiments and they were also commonly used in the literature (Liu, 2012). However, we impose constraints on selecting adverbs. While adverbs like "quickly" and "nicely" do express sentiment, some others like "very" and "often" may not function the same. We thus use the adverbs having adjective forms only.

Notice that in the above setting, the sentiment words not present in the lexicon are also collected due to two reasons: first, they are useful for building connection among other lexical words for inference purposes. Suppose that "quick" is a sentiment word (found because it is an adjective) and it is not in the given lexicon. Given its sentiment correlations with other words like "efficient and quick" and "quiet and quick", it can make a path between "efficient" and "quiet" in the graph. Second, in each domain there exist a number of sentiment words uncovered by the given lexicon. Their inferred polarities can also benefit the graph reasoning process, though those words are not the focus of detection in this study (we aim at detecting the polarity change of lexical words, i.e., words from a given lexicon). For instance, if the non-lexical word "quick" is identified as expressing positive sentiment (in the application domain) , "efficient" and "quiet" are more likely to be positive as well, given their in-domain sentiment correlations. We follow (Das and Chen, 2007; Pang and Lee, 2008) to handle the negation problem, where a negation word phrase like "not bad" will be treated as a single word like "not_bad" and its sentiment polarity will be reversed accordingly. Finally, all extracted words are modeled as nodes in the graph.

4.2 Sentiment Correlation Extraction

This step is to extract multiple types of conjunction relationship among sentiment words, which

---

[1] In this paper, we call the words in a given lexicon *lexical words* for short. The term *detection* will generally stand for the detection of domain polarity-changes of (lexical) words.

we refer to as *sentiment correlation*[2]. The key idea here is to use the sentiment consistency (Abelson, 1983; Liu, 2012) (see Assumption 1) for relationship construction among the collected sentiment words from the above step. Specifically, in an application domain, five types of sentiment correlation are considered, each of which is presented in a triple format, denoted as ($word_1$, correlation type, $word_2$). They will be further used in the graph inference process (discussed in the next sub-section). Their definitions are shown in Table 1.

In each sentence, when a specific type of relationship between two (collected) sentiment words is found, a triple is created. For instance, in the sentence "it is efficient and quiet", a triple (efficient, AND, quiet) will be generated. The extraction of OR sentiment correlation is similar to AND. Likewise, a specific BUT triple (powerful, BUT, noisy) will be extracted from the sentence "it is a powerful but noisy machine". The extraction of ALT (abbreviation for although) is similar to BUT. NB means two neighboring sentiment words occur in a sentence, like "reasonably good".

Notice that while five types of relationships are jointly considered, they are associated with different agreement levels (parameterized in the graphical model discussed below). Here the agreement level measures how likely the sentiment polarities of two connecting words are the same. Intuitively, we believe AND gives the highest-level agreement. For instance, "bad and harmful" is very common but "good and harmful" is much unlikely. It is also an intuitive belief that BUT indicates the strongest disagreement between two sentiment words. Note that we only consider pairwise relationships between sentiment words in this study, which already achieve reasonably good results, as we will see.

4.3 Graph Construction and Inference

This subsection presents how our proposed domain-specific sentiment graph is used for detecting polarity-changed words after the above two steps.

4.3.1 Constructing Markov Random Field

Markov Random Fields (MRFs) are a class of probabilistic graphical models that can deal with the inference problems with uncertainty in observed data. An MRF works on an undirected graph G, which is constructed by a set of vertexes/nodes V and edges/links E and denoted as $G = (V, E)$. In the graph G, each node $v_i \in V$ denotes a random variable and each edge $(v_i, v_j) \in E$ denotes a statistical dependency between node v_i and node v_j. Formally, $\psi_i(v_i)$ and $\psi_{ij}(v_i, v_j)$ are defined as two types of potential functions for encoding the observation (or *prior*) of a node and the dependency between two nodes, also called *node potential* and *edge potential* respectively. An MRF thus can model a joint distribution for a set of random variables and its aim is to infer the marginal distribution for all $v_i \in V$. With an inference method used, the estimation of the marginal distribution of a node can be obtained, which is also called *belief*.

The reason why we formulate our domain-specific sentiment graph as an MRF is three-fold: (1) The sentiment correlation between two words is a mutual relationship, as one word w_a can provide useful sentiment information to the other word w_b and vice versa, which can be properly formulated in an undirected graph. (2) From a probabilistic perspective, the polarity changes of sentiment words can be naturally understood as the belief estimation problem. That is, on one hand, we have an initial belief about the polarity of a lexical word (known from the lexicon, like the word "cold" is generally negative), which is essentially the prior. On the other hand, our goal is to infer the real polarity of a word in a specific application domain, which is reflected in its final estimated belief (like "cold" is positive in the domain *fridge*). Concretely, the polarity of a sentiment word is modeled as a 2-dimensional vector, standing for the probability distribution of positive and negative polarities, e.g., $[0.9, 0.1]$ indicates that a word is very likely to express a positive sentiment in an application domain. We can further use p as the parameter to simplify the representation as $[p, 1-p]$. (3) Recall that multiple types of sentiment correlation are used and treated differently in our proposed approach. These *typed sentiment correlations* can be encoded in the MRF model (will be further illustrated below).

4.3.2 Inference over Typed Correlation

As discussed above, the inference task in MRF is to compute the marginal distribution (or posterior probability) of each node given the node prior and edge potentials. Efficient algorithms for exact inference like Belief Propagation (Pearl, 1982) are available for certain graph topologies, but for general graphs involving cycles the exact inference is computationally intractable. Approximate inference is

---

[2] The term *sentiment correlation* used in this paper denotes the correlation between two sentiment words in a domain, which may not have to be the same as used in other studies.

Name	Correlation	Example	Representation	Agreement Level
AND	connecting with "and"	"it is efficient and quiet"	(efficient, AND, quiet)	Strongly Agree
OR	connecting with "or"	"everything as expected or better"	(expected, OR, better)	Agree
NB	neighboring words	"a reasonably quiet fridge"	(reasonably, NB, quiet)	Weakly Agree
ALT	although, though	"too noisy, though it is efficient"	(noisy, ALT, efficient)	Disagree
BUT	but, however	"it is a powerful but noisy machine"	(powerful, BUT, noisy)	Strongly Disagree

Table 1: Five types of sentiment correlation.

thus needed. Loopy Belief Propagation (Murphy et al., 1999) is such an approximate solution using iterative message passing. A message from node i to node j is based on all message from other nodes to node i except node j itself. It works as:

$$m_{i\to j}(x_j) = z \sum_{x_i \in S} \psi_{i,j}(x_i, x_j)\psi_i(x_i) \prod_{k\in N(i)\backslash j} m_{k\to i}(x_i) \tag{1}$$

where S denotes the possible states of a node, i.e., being a sentiment word with positive or negative polarity. x_j indicates that node j is in a certain state. $N(i)$ denotes the neighborhood of i, i.e., the other nodes linking with node i. $m_{i\to j}$ is known as the message passing from node i to node j. z is the normalization constant that makes message $m_{i\to j}(x_j)$ proportional to the likelihood of the node j being in state x_j, given the evidence from i in its all possible states. After iterative message passing, the final belief $b_i(x_i)$ is estimated as:

$$b_i(x_i) = z'\psi_i(x_i) \prod_{k\in N(i)} m_{k\to i}(x_i) \tag{2}$$

where z' is a normalization term that makes $\sum_{x_i} b_i(x_i) = 1$. In this case, $b_i(x_i)$ can be viewed as the posterior probability of a sentiment word being with positive or negative polarity.

However, notice that in the above setting, each edge is not distinguishable in terms of its type of sentiment correlation. In other words, each type of possible connections between words is treated intrinsically the same, which does not meet our modeling requirements. In order to encode the typed sentiment correlation as defined in previous sections, we propose to replace the Eq. 1 with:

$$\begin{aligned} m_{i\to j}(x_j) =& z \sum_{x_i\in S} \sum_{r_{i,j}\in R} \psi_{i,j,r_{i,j}}(x_i, x_j) \\ & \psi_i(x_i) \prod_{k\in N(i)\backslash j} m_{k\to i}(x_i), \end{aligned} \tag{3}$$

where $r_{i,j} \in R$ indicates the specific type of sentiment correlation between node i and node j, which can be any type like AND or NB as defined in Section 4.2. $\psi_{i,j,r_{i,j}}(x_i, x_j)$ thus becomes an edge potential function related to its sentiment correlation type. Each type of a correlation is parameterized as a (state) transition matrix shown in Table 2. The five types of sentiment correlation therefore result in five such tables/matrices but with different ϵ being set. For example, ϵ with AND can be set to 0.3 as it indicates the highest agreement level, while the one with NB can be set to 0.1 as it is regarded as weakly agreement. For BUT, ϵ can be set to -0.3 as it shows strong disagreement.

State	Positive	Negative
Positive	$0.5 + \epsilon$	$0.5 - \epsilon$
Negative	$0.5 - \epsilon$	$0.5 + \epsilon$

Table 2: Transition/Propagation matrix.

For each word, with its estimated beliefs $[b^+, b^-]$ obtained in the application domain, its *polarity change score (pcs)* is defined as:

$$pcs = I(l = +)b^- + I(l = -)b^+ \tag{4}$$

where l denotes the original sentiment polarity of a lexical word (known from the given lexicon), and $I(.)$ is an indicator function. According to the resulting scores of all words, a word list ranked by *pcs* is used to identify the most likely polarity-changed sentiment words, e.g., one can select the top n words or set a threshold for word extraction. In this way, the sentiment words with changed polarities in an application domain can be detected.

4.4 An Illustrating Example for DSG

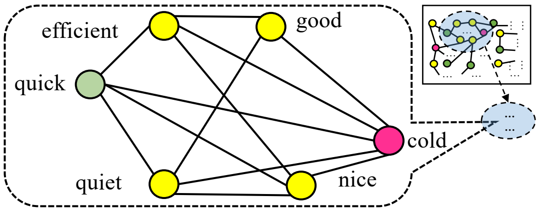

Figure 1: DSG Example. The nodes in pink (or darkest in grayscale) like "cold" represent the polarity-changed words in a domain, while nodes in yellow (or lightest) like "nice" indicate polarity-unchanged words, and nodes in green like "quick" denote non-lexical words.

Figure 1 provides an example to illustrate how DSG works in domain *fridge*. It shows a subgraph with several nodes, such as "quick", "nice", and "cold", representing the words extracted from step 1. The links between them represent their sentiment correlation as discussed in step 2. With the inference performed in step 3, "cold" is detected as a polarity-changed word (e.g., with a high pcs value 0.9 inferred), where its sentiment information is estimated by the (sentiment) message passing from other nodes in the graph. Specifically, in the *fridge* domain, through sentences like "looks nice and quiet", "it is quiet and cold", and "nice to have a quick cold beer or soda" (from the fridge), positive sentiment is directly or indirectly propagated from the words/nodes "nice", "quiet", "quick" to "cold" with their (typed) connections. Therefore, while "cold" was assigned a negative prior $[0.0, 1.0]$ (by default negative in the given lexicon), through the in-domain sentiment inference its final estimated belief becomes $[0.9, 0.1]$, resulting a high pcs score ($psc = 0.9$ by Eq. 4). "cold" is thus detected.

5 Experiments

We conducted experiments on multiple datasets with several candidate solutions. Here we first compare their performance on the word detection task. Sentiment classification will then be another task to evaluate their effect on polarity correction.

5.1 Experimental Setup

Dataset. Four real-world datasets from different domains/products were used, namely, *fridge*, *blenders*, *washer*, and *movie*. The first three datasets contain review sentences of these products. The fourth dataset (*movie*) consists of tweets from Twitter discussing movies. All these datasets are collected by us. The first dataset contains around 36,000 (36k) sentences, the second 16,000 (16k) sentences, and the rest datasets 10,000 (10k). Their sizes can be viewed as large, medium, and small. Such product diversity and variable size settings help evaluate the generality of each solution. Note that only the text is used by all candidate models.

In addition, two other datasets from domains *drill* and *vacuum cleaner* are used as development sets for hyper-parameter selections. *drill* contains 76k and *vacuum cleaner* contains 9k sentences.

Sentiment Lexicon. We used a general-purpose sentiment lexicon from (Hu and Liu, 2004), which contains 2,006 positive and 4,783 negative lexical words. A candidate model will find polarity-changed words from them for each domain/dataset.

Parameter Settings. The hyper-parameters of state priors and the (typed) transition matrices in DSG are shown in Table 3 and 4. They were empirically set based on the word detection performance on the two development datasets. We found this parameter setting works generally well on both datasets, while they are from different domains and with different data sizes. The following reported results for evaluations are based on this setting and as we will see, it already produces quite good results.

Prior	Positive	Non-lexical Words	Negative
$p\ in\ \psi_i(v_i)$	0.70	0.50	0.30

Table 3: Parameters of state prior.

Types	AND	OR	NB	ALT	BUT
ϵ **in** $\psi_{i,j}(v_i, v_j)$	0.20	0.10	0.05	-0.10	-0.20

Table 4: Parameters of typed transition matrix.

5.2 Lexicon-based Sentiment Classifier

Our evaluations include lexicon-based sentiment classification. We briefly illustrate how a lexicon-based sentiment classifier (called *classifier* for short) works here. Clearly, it works with a lexicon, from which each word is associated with a sentiment score (e.g., -1/+1). The classifier then calculates the sentiment score s of each sentence t by summing the score of each word. We follow the lexicon-based classifier design in (Taboada et al., 2011), incorporating sentiment negation and intensity. The sentence sentiment score s is calculated:

$$s = \sum_{w \in t} negation(w) * intensity(w) * polarity(w) \quad (5)$$

where w denotes a word in sentence t. A sentence is classified as positive if the resulting score s is greater than or equal to zero, otherwise negative.

Different lexicons working with this classifier will generate different results. That is, even if their lexical words are the same, the associated sentiment score of a lexical word could vary and Eq. 5 will thus make different predictions. This is how we can utilize the classifier to verify the effect of the word detection, because the classifier will perform differently using the original lexicon and the modified lexicon, whose results can be compared in a *before-and-after* manner. Here the *modified lexicon* means the sentiment polarities of (detected) lexical words are corrected from the original lexicon. For example, "crush" is associated with negative sentiment (-1) in the original lexicon, but it could be associated with positive sentiment (+1) in the modified lexicon (if detected). So the sentence-level sentiment scores will vary accordingly, e.g., "the machine does crush ice!" will be predicted as a positive sentence with the modified lexicon.

5.3 Candidate Models

Original Lexicon (OL): This is a baseline for sentiment classification evaluations only (Section 5.5), which uses the classifier with the original lexicon.
Domain-specific Sentiment Graph (DSG): This is our proposed model introduced in the previous sections. The following two models and DSG will be used for both word detection and classification.
Lexicon-Classifier Inconsistency (LCI): This is a heuristic solution to detecting the polarity-changed sentiment words. It relies on the inconsistency between the sentiment of a lexical word (obtained from the original lexicon) and the sentiment of the sentences containing the word (induced by the classifier). Concretely, it first calculates the sentiment polarities of all sentences using a classifier with the original lexicon. With the polarities of sentences known, it computes an inconsistency ratio for each lexical word. The inconsistency ratio is the ratio of (a) to (b), where (a) is the number of a word appearing in the positive/negative sentences but the word itself is negative/positive, and (b) is the number of all sentences covering that word. Finally, it ranks all lexical words based on their ratio values to produce a list of likely polarity-changed words.
SentProp (SP): SentProp (Hamilton et al., 2016) is a lexicon construction algorithm concerning domain sentiment, which is the most related work to ours. As discussed in Section 2, it is not directly applicable to the detection task. But since it can generate a list of domain-specific sentiment words and those words are associated with positive/negative scores (estimated by SentProp, which can be treated as the in-domain beliefs like DSG), we can design a two-step approach to achieve our goal. First, we download[3] and run the SentProp system to learn the domain-specific lexicon for each domain. Second, we calculate the polarity change scores for all lexicon words like DSG based on the learned domain-specific sentiment scores and the original polarities from the lexicon using Eq. 4. Similar to DSG, it produces a list of words ranked by the polarity change scores (pcs). For its parameter selection, we tried both the system default, following the code instruction and the original paper, and parameter fine-tuning based on the performance on two development sets (same as DSG), so as to achieve its best performance to report.

5.4 Correct Detection of Words

As each candidate model generates a list of words ranked by polarity-change scores, those top-ranked ones are the most likely polarity-changed words and can be used as the detected words. For evaluation, the top-n words from each model are inspected and the number of correct (word) detection is counted, which is denoted as **#C@n** in Table 5.

Specifically, two domain experts who are familiar with the domain sentiments identify and annotate the correct polarity-changed words from the top-20 shortlisted candidate words generated by each model. For each candidate word, we sampled numbers of sentences containing that word for the domain experts to judge. A candidate word needs to be agreed upon by both of them to be correct. Here the Cohen's Kappa agreement score is 0.817.

Model	#C@n	fridge	blender	washer	movie
DSG	#C@5	5	5	4	5
	#C@10	9	10	6	9
	#C@20	12	15	12	15
LCI	#C@5	3	3	3	1
	#C@10	5	3	5	4
	#C@20	5	7	7	9
SP	#C@5	1	0	1	1
	#C@10	2	0	2	4
	#C@20	3	3	3	6

Table 5: Detection of polarity-changed words.

Evaluation results are reported in Table 5, where we can see that DSG achieves outstanding results consistently. LCI also does a decent job, while SP does not perform well on this task.

Next, we will evaluate the impact of such detection from their top 20 words, and the following sub-sections are based on their correctly detected words to give further analyses.

5.5 Sentiment Classification

After the detection of polarity-changed words, we conduct classification tasks on the sentences containing at least one word from the detected words of all models. Because the classification results on the sentences that without containing any detected word would not be affected (same prediction results using either the original or modified lexicon).

For evaluation, we sampled and labeled 925 (around 1k) sentences, from all sentences that could be affected. We used a stratified sampling strategy

---

[3] https://nlp.stanford.edu/projects/socialsent/

and set the minimum number of sentences contained by each word, to make sure each detected word is considered. The numbers of labeled sentences for the four domains are 232, 214, 174, and 305. The Cohen's Kappa agreement score is 0.788.

In regard to the lexicon-based classification, for DSG and LCI, the modified lexicon for each domain is based on the correction of the original lexicon (OL) on that domain. For SP, its self-generated lexicon is used with its inferred sentiment scores.

Model	fridge	blender	washer	movie	AVG
DSG	**74.56%**	**80.84%**	**77.01%**	84.91%	**79.33%**
SP	68.10%	78.97%	66.67%	**87.87%**	75.40 %
LCI	61.63%	68.22%	62.64%	62.95%	63.86%
OL	61.20%	65.42%	62.06%	56.72%	61.35%

Table 6: Sentiment classification accuracy.

Table 6 reports the classification accuracy, from which we have the following observations:

1. Compared to the baseline using the original lexicon (OL), DSG greatly improves the accuracy by 17.98% on average (AVG). We can see the usefulness of detecting polarity change of lexical words for sentiment classification.
2. SP also produces very good results. The reason is, as a lexicon-generation approach (essentially), SP itself creates a bigger lexicon for each domain (around 2 times bigger than OL), including additional sentiment words outside the original lexicon. In other words, discovering more sentiment words (with more sentiment clues provided) could also help better classification. Note that this does not contradict the importance of detecting polarity-change words, as they are two different aspects, which will be discussed in Section 5.7.
3. LCI outperforms OL but its performance gain is small. The reason is, though LCI detects polarity changed words decently, its detected words affect a much smaller number of sentences compared to the ones from DSG and SP, i.e., the words LCI detects are rarer and less frequent, with fewer sentences being affected.

5.6 Example Results

blender	movie
frozen shake cheap crushing	despicable fucking complex
breaks crush lost destroy	insanely damn *sad* crazy
loose crushed breaking	intense funny freaking *creepy*
broken lose grind *clog bulky*	bloody crap shit bad *retarded*
kills wobbled bitter shocked	*mad* insane terribly *eccentric*

Table 7: Example results on *blender* and *movie*. Incorrectly detected words are italicized and marked in red.

Here we show some example results. Table 7 lists the top polarity-changed words detected by DSG on domains *blender* and *movie*; we will explain how they benefit the domain sentiment analysis with example sentences. Below, an underlined word in an example sentence indicates its polarity has changed in a certain domain. In domain *blender*, the words like "crushing" and "breaks" express positive sentiments in "good for crushing ice" and "this breaks down frozen fruit", while they are provided as negative words in the original lexicon. With the detection and correction of those words, their domain-specific sentiments are identified correctly as positive and thus make the classification correct (through the modified lexicon provided by DSG). In domain *movie*, we found that many negative lexical words are in fact commonly used by users to show their love for a movie. For instance, "damn, I wanna watch this movie so bad" and "this movie was insanely brilliant". Similarly, for other domains, we also found "it keeps foods cold and frozen" in *fridge*, "you can also delay your cycle" in *washer*, and "it sucked very well" in *vacuum cleaner*. DSG also detects those words.

5.7 Further Analysis and Discussion

We aim to answer two questions here. Q1: What is the key difference between using SP and DSG? Q2: More generally, what is the relationship between the existing lexicon generation research and the polarity-change detection problem (which is studied in this work)?

First, let us dive a bit deeper into SP. As a lexicon generation approach, its goal is to collect sentiment words from a given corpus and infer their sentiment scores. There are two important notes: (a) while SP could discover more sentiment words, those extracted words could be wrong. For example, SP extracts the word "product" as a sentiment word and assigns it a positive (+ve) sentiment. This could lead to mis-classifications of the negative (-ve) sentences containing "product". (b) while SP directly discovers and estimates sentiment words, it does not know which sentiment words carry important domain-shifted sentiment. For example, SP discovers "excellent", "crush", "terrible" for the domain *blender* and estimates the sentiment scores as 0.9, 0.7, and 0.1 (for simplicity, let us assume all scores are rescaled to [0.0, 1.0], where 1.0 denotes most

+ve and 0.0 most -ve). Those scores indicate their polarities, but do not reflect their importance/effect of polarity change towards that domain.

For DSG, (a) could be avoided because "product" is usually excluded in a general-purpose lexicon. Regarding (b), say the scores of "excellent", "crush" and "bad" are 1.0, 0.0, and 0.0 in the original lexicon; with the domain sentiment re-estimation from DSG, they become 0.9, 0.7, and 0.1. Their polarity-changed scores are thus inferred as 0.1 ($|1.0 - 0.9|$), 0.7, and 0.1, where "crush" can be found as an important domain-sentiment changed word (0.7).

Certainly, one can compare the SP generated lexicon to the original/general lexicon to detect the changes. We already did this for the detection task (Table 5). Here we design a variant SP-dsg-like, following this idea to perform the sentiment classification task. The main difference between SP and SP-dsg-like is that SP directly uses its generated lexicon and sentiment scores, while SP-dsg-like uses its generated lexicon to modify the original lexicon (OL) like DSG does. However, SP-dsg-like performs poorly (Table 8), mainly because the modified lexicon (based on OL) does not fully reflect the whole sentiment words generated by SP.

Model	fridge	blender	washer	movie	AVG
OL	61.20%	65.42%	62.06%	56.72%	61.35%
SP	68.10%	78.97%	66.67%	87.87%	75.40%
SP-dsg-like	67.67%	71.02%	64.36%	63.93%	66.75%
SP-dsg-like+SP	69.40%	77.57%	68.97%	83.28%	74.81%
OL + SP	62.07%	78.04%	70.11%	82.30%	73.13%
CLI + SP	62.07%	77.57%	70.67%	83.61%	73.48%
DSG + SP	**72.41%**	**78.97%**	**79.89%**	**88.85%**	**80.03%**

Table 8: Sentiment classification accuracy.

We then combine two lexicons together to give another variant SP-dsg-like+SP, which means the modified lexicon (based on OL) is expanded by the SP self-generated lexicon, where the SP generated lexicon can contain additional sentiment words (outside OL). Similarly, we can make OL+SP and CLI+SP, but they are all inferior to SP (Table 8). The reason is that the key polarity-changed words in the original lexicon have not been corrected, keeping their wrong sentiments for classification.

However, notice that DSG can effectively detect and correct those words; when we use DSG+SP, the overall results are improved (AVG) and even better than using DSG or SP only (Table 8 and Table 6).

It has been shown that either getting more sentiment words (from SP) or fixing a smaller number of important polarity-changed words (from DSG) can help sentiment classification (Table 6). With DSG+SP working even better, we can view the lexicon generation and the domain polarity-changed (lexical) word detection as two directions for classification improvement. Either one has its own advantage. Lexicon generation methods can induce more words and may help find rarer/infrequent words. The domain polarity-changed lexical word detection can be handy and less risky, as it directly corrects the polarities of important lexical words and would not induce noises (wrong sentiment words).

Finally, the answers to Q1 and Q2 are: using SP/lexicon-generation and DSG/polarity-change-detection can both improve sentiment classification, but in different manners (i.e., two directions as discussed above). Besides, using DSG can effectively detect important polarity-changed words, while SP does not perform very well on this task. These two directions could be complementary, as indicated by DSG+SP. Note that in this work we have shown that incorporating the DSG corrected lexical words (i.e., DSG modified lexicon) into SP can help boost its performance, which can be viewed as injecting clean/reliable domain sentiment knowledge. Another possible enhancement as future work is that, for the additional (non-lexical) sentiment words found by SP (or other domain lexicon generation method), we can use DSG to detect and correct their changed/shifted polarity in domains. However, for that, we will also need to deal with the induced noise (newly identified but wrong sentiment words like "product"), perhaps with some denoising or pruning techniques. We also hope this work and its findings can inspire more future work.

6 Conclusion

This paper studied the problem of detecting domain polarity-changed words in a sentiment lexicon. As we have seen, the wrong polarities seriously degenerate the sentiment classification performance. To address it, this paper proposed a novel solution named Domain-specific Sentiment Graph (DSG). Experimental results demonstrated its effectiveness in finding the polarity-changed words and its resulting performance gain in sentiment classification.

Acknowledgments

This work was supported in part by two grants from National Science Foundation: IIS-1910424 and IIS-1838770, a DARPA Contract HR001120C0023, and a research gift from Northrop Grumman.

References

Robert P Abelson. 1983. Whatever became of consistency theory? *Personality and Social Psychology Bulletin*, 9(1):37–54.

Himanshu Sharad Bhatt, Deepali Semwal, and Shourya Roy. 2015. An iterative similarity based adaptation technique for cross-domain text classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 52–61.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A. Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *Proceedings of WWW-2008 workshop on NLP in the Information Explosion Era*.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*, pages 793–801.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 590–598.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management science*, 53(9):1375–1388.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the Conference on Web Search and Web Data Mining (WSDM-2008)*, pages 231–240.

Eduard C. Dragut, Clement Yu, Prasad Sistla, and Weiyi Meng. 2010. Construction of a sentimental word dictionary. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM-2010)*, pages 1761–1764.

Weifu Du, Songbo Tan, Xueqi Cheng, and Xiaochun Yun. 2010. Adapting information bottleneck method for automatic construction of domain-oriented sentiment lexicon. In *Proceedings of ACM International Confernece on Web search and data mining (WSDM-2010)*, pages 111–120.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM-2005)*, pages 617–624.

Lorenzo Gatti and Marco Guerini. 2012. Assessing sentiment strength in words prior polarities. In *Proceedings of the Conference of the 24th International Conference on Computational Linguistics (COLING-2012)*.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 595. NIH Public Access.

Ahmed Hassan and Dragomir Radev. 2010. Identifying text polarity using random walks. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2010)*, pages 395–403.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-1997)*, pages 174–181.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, pages 168–177.

Sean P Igo and Ellen Riloff. 2009. Corpus-based semantic lexicon induction with web-based corroboration. In *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*, pages 18–26. Association for Computational Linguistics.

Valentin Jijkoun, Maarten de Rijke, and Wouter Weerkamp. 2010. Generating focused topic-specific sentiment lexicons. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2010)*.

Nobuhiro Kaji and Masaru Kitsuregawa. 2006. Automatic construction of polarity-tagged corpus from html documents. In *Proceedings of COLING/ACL 2006 Main Conference Poster Sessions (COLING-ACL-2006)*, pages 452–459.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientation of adjectives. In *Proc. of LREC-2004*, volume 4, pages 1115–1118.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, pages 355–363.

Wiltrud Kessler and Hinrich Schütze. 2012. Classification of inconsistent sentiment words using syntactic constructions. In *Proceedings of the Conference of the 24th International Conference on Computational Linguistics (COLING-2012)*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of Interntional Conference on Computational Linguistics (COLING-2004)*, page 1367.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 946–956.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018b. Hierarchical attention transfer network for cross-domain sentiment classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, Lousiana, USA, February 2-7, 2018*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proceedings of the 20th international conference on World wide web (WWW-2011)*, pages 347–356.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 599–608.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc.

Antonio Moreno Ortiz. 2017. Linfmotif: Sentiment analysis for the digital humanities. In *15th Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Software Demonstrations: April 3-7, 2017 Valencia, Spain*, pages 73–76. The Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Using very simple statistics for review search: An exploration. In *Proceedings of International Conference on Computational Linguistics (COLING-2008)*.

Judea Pearl. 1982. *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles.

Wei Peng and Dae Hoon Park. 2011. Generate adjective sentiment dictionary for social media sentiment analysis using constrained nonnegative matrix factorization. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM-2011)*, volume 51, page 61801.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL-2009)*, pages 675–682.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777.

Inaki San Vicente, Rodrigo Agerri, and German Rigau. 2014. Simple, robust and (almost) unsupervised generation of polarity lexicons for multiple languages. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 88–97.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2007. Extracting semantic orientations of phrases from dictionary. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL-2007)*.

Zhiyang Teng, Duy Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638.

Peter D. Turney and Micharel L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. 2004. Developing affective lexical resources. *PsychNology Journal*, 2(1):61–83.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics (HAACL-2010)*, pages 777–785.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring sentiment in social media: Bootstrapping subjectivity clues from multilingual twitter streams. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-2013)*.

Bo Wang and Houfeng Wang. 2008. Bootstrapping both product features and opinion words from chinese customer reviews with cross-inducing. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP-2008)*.

Shuai Wang, Zhiyuan Chen, and Bing Liu. 2016. Mining aspect-specific opinion using a holistic lifelong topic model. In *Proceedings of the 25th international conference on world wide web*, pages 167–176.

Shuai Wang, Guangyi Lv, Sahisnu Mazumder, Geli Fei, and Bing Liu. 2018a. Lifelong learning memory networks for aspect sentiment classification. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 861–870. IEEE.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018b. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 957–967.

Yasheng Wang, Yang Zhang, and Bing Liu. 2017. Sentiment lexicon expansion based on neural pu learning, double dictionary lookup, and polarity association. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 553–563.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005)*, pages 347–354.

Yunfang Wu and Miaomiao Wen. 2010. Disambiguating dynamic sentiment ambiguous adjectives. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1191–1199.

Ge Xu, Xinfan Meng, and Houfeng Wang. 2010. Build chinese emotion lexicons using a graph-based algorithm and multiple resources. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1209–1217. Association for Computational Linguistics.

Min Yang, Baolin Peng, Zheng Chen, Dingju Zhu, and Kam-Pui Chow. 2014. A topic model for building fine-grained domain-specific emotion lexicon. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*. Association for Computational Linguistics (ACL).

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 236–246.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Yanyan Zhao, Bing Qin, and Ting Liu. 2012. Collocation polarity disambiguation using web-based pseudo contexts. In *Proceedings of the 2012 Conference on Empirical Methods on Natural Language Processing (EMNLP-2012)*.

Li Zhuang, Feng Jing, and Xiaoyan Zhu. 2006. Movie review mining and summarization. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM-2006)*, pages 43–50.

Analyzing Online Political Advertisements

Danae Sánchez Villegas$^{\alpha}$ **Saeid Mokaram**$^{\beta}$ **Nikolaos Aletras**$^{\alpha}$
$^{\alpha}$ Computer Science Department, University of Sheffield, UK
$^{\beta}$ Emotech
`{dsanchezvillegas1, n.aletras}@sheffield.ac.uk`
`saeid.mokaram@gmail.com`

Abstract

Online political advertising is a central aspect of modern election campaigning for influencing public opinion. Computational analysis of political ads is of utmost importance in political science to understand the characteristics of digital campaigning. It is also important in computational linguistics to study features of political discourse and communication on a large scale. In this work, we present the first computational study on online political ads with the aim to (1) infer the political ideology of an ad sponsor; and (2) identify whether the sponsor is an official political party or a third-party organization. We develop two new large datasets for the two tasks consisting of ads from the U.S.. Evaluation results show that our approach that combines textual and visual information from pre-trained neural models outperforms a state-of-the-art method for generic commercial ad classification. Finally, we provide an in-depth analysis of the limitations of our best-performing models and linguistic analysis to study the characteristics of political ads discourse.[1]

1 Introduction

Online advertising is an integral part of modern digital election campaigning (Fulgoni et al., 2016; Fowler et al., 2020a). The increased spending on online political ads (e.g. the 2020 U.S. election campaign spending hit an all-time record[2]) poses a significant challenge to the democratic oversight of digital campaigning,[3] with serious implications about transparency and accountability, for example how voters are targeted and by whom (Kriess and Barrett, 2020).

Political advertising is defined as *'any controlled message communicated through any channel designed to promote the political interests of individuals, parties, groups, government, or other organizations'* (Kaid and Holtz-Bacha, 2006). It is guided by ideology and morals (Scammell and Langer, 2006; Kumar and Pathak, 2012), and often expresses more negativity (Haselmayer, 2019; Iyengar and Prior, 1999; Lau et al., 1999) compared to the aesthetic nature of commercial advertising. Table 1 shows examples of online political ads across different political parties and sponsor types.

While the closely related online *commercial* advertising domain has recently been explored in natural language processing (NLP) for predicting the category (e.g. politics, cars, electronics) and sentiment of an ad (Hussain et al., 2017; Kalra et al., 2020), online *political* advertising has yet to be explored. Large-scale studies of online political advertising have so far focused on understanding targeting strategies rather than developing predictive models for analyzing its content (Edelson et al., 2019; Medina Serrano et al., 2020).

Automatically analyzing political ads is important in political science for researching the characteristics of online campaigns (e.g. voter targeting, sponsors, non-party campaigns, privacy, and misinformation) on a large scale (Scammell and Langer, 2006; Johansson and Holtz-Bacha, 2019). Moreover, identifying ads sponsored by third-party organizations is critical to ensuring transparency and accountability in elections (Liu et al., 2013; Speicher et al., 2018; Fowler et al., 2020b; Edelson et al., 2019). For example, third-party advertising had an increased presence in the U.S. House and Senate races in 2018 considerably more than in 2012 where almost half of the third-

---

[1] Data is available here: `https://archive.org/details/pol_ads`

[2] `https://www.cnbc.com/2020/10/01/election-2020-campaign-spending-set-to-hit-record-11-billion.html`

[3] `https://www.electoral-reform.org.uk/latest-news-and-research/publications/democracy-in-the-dark-digital-campaigning-in-the-2019-general-election-and-beyond/`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3669–3680
August 1–6, 2021.

Political Ideology	Ad Sponsor Type	Sample Ad
Liberal	Political Party	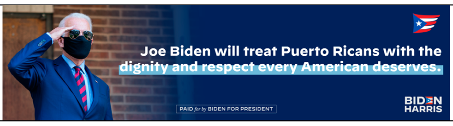
Conservative	Political Party	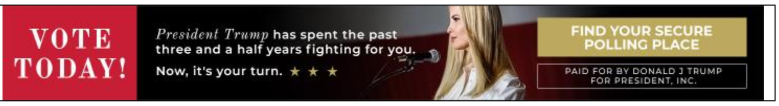
N/A	Third-Party	

Table 1: Examples of online political ads by sponsor political ideology and type.

party sponsored ads were funded by dark-money sources (Fowler et al., 2020b). Finally, computational methods for political ads analysis can help linguists to study features of political discourse and communication (Kenzhekanova, 2015; Skorupa and Dubovičienė, 2015).

In this paper, we present a systematic study of online political ads (consisting of text and images) in the U.S. to uncover linguistic and visual cues across political ideologies and sponsor types using computational methods for the first time. Our contributions are as follows:

1. A new classification task for predicting the political ideology (conservative or liberal) of an ad (§3). We collect 5,548 distinct political ads in English from 242 different advertisers in the U.S., and label them according to the dominant political ideology of the respective sponsor's party affiliation (*Liberal* or *Conservative*);

2. A new classification task to automatically classify ads that were sponsored by official political parties and third-party organizations, such as businesses and non-profit organizations (§3). For this task, we extract 15,116 advertisements in English from 665 distinct advertisers in the U.S., and label them as *Political Party* (i.e. officially registered) and *Third-Party* (i.e. other organizations) following Fowler et al. (2020b);

3. Experiments with text-based and multimodal (text and images) models (§4) for political ideology prediction and sponsor type classification reaching up to 75.76 and 87.36 macro F1 in each task respectively (§6);

4. Analysis of textual and visual features of online political ads (§7) and error analysis to understand model limitations.

2 Related Work

2.1 Political Communication and Advertising

Previous work on analyzing political advertising has covered television and online ads (Kaid and Postelnicu, 2005; Reschke and Anand, 2012; West, 2017; Fowler et al., 2020b). Ridout et al. (2010) analyze a series of YouTube videos posted during the 2008 presidential campaign to understand its influence on election results as well as the actors and formats compared to traditional television ads. Anstead et al. (2018) study how online platforms such as Facebook are being used for political communication and identify challenges for understanding the role of these platforms in political elections, highlighting the lack of transparency (Caplan and Boyd, 2016). Fowler et al. (2020b) explore differences between television and online ads, and demonstrate that there is a greater number of candidates advertising online than on television.

2.2 Political Ideology Prediction

Inferring the political ideology of various types of text including news articles, political speeches and social media has been vastly studied in NLP (Lin et al., 2008; Gerrish and Blei, 2011; Sim et al., 2013; Iyyer et al., 2014; Preoţiuc-Pietro et al., 2017; Kulkarni et al., 2018; Stefanov et al., 2020). Bhatia and P (2018) exploit topic-specific sentiment analysis for ideology detection (i.e. conservative, liberal) in speeches from the U.S. Congress. Kulkarni et al. (2018) propose a multi-view model that incorporates textual and network information to predict the ideology of news articles. Johnson and Goldwasser

(2018) investigate the relationship between political ideology and language to represent morality by analyzing political slogans in tweets posted by politicians. Maronikolakis et al. (2020) present a study of political parody on Twitter focusing on the linguistic differences between tweets shared by real and parody accounts. Baly et al. (2019) estimate the trustworthiness and political ideology (left/right bias) of news sources as a multi-task problem. Stefanov et al. (2020) develop methods to predict the overall political leaning (left, center or right) of online media and popular Twitter users.

Political ideology and communicative intents have also been studied in computer vision. Political images have been analyzed to infer the persuasive intents using various features such as facial display types, body poses, and scene context (Joo et al., 2014; Huang and Kovashka, 2016; Joo and Steinert-Threlkeld, 2018; Bai et al., 2020; Chen et al., 2020). Joo et al. (2015) introduce a method that infers the perceived characteristics of politicians using face images and show that those characteristics can be used in elections forecasting. Xi et al. (2020) analyze the political ideology of Facebook photographs shared by members of the U.S. Congress. Chen et al. (2020) examine the role of gender stereotypical cues from photographs posted in social media by political candidates and their relationship to voter support.

2.3 Computational Analysis of Online Ads

Hussain et al. (2017) propose the task of ad understanding using vision and language. The aim is to predict the topical category, sentiment and rhetoric of an ad (i.e. what the message is about). The latter task has been approached as a visual question-answering task by ranking human generated statements that explain the intent of the ad in computer vision (Ye and Kovashka, 2018; Ahuja et al., 2018). More recently in NLP, Kalra et al. (2020) propose a BERT-based (Devlin et al., 2019) model for this task using the text and visual descriptions of the ad (Johnson et al., 2016). Thomas and Kovashka (2018) study the persuasive cues of faces across ad categories (e.g. beauty, clothing). Zhang et al. (2018) explore the relationship between the text of an ad and the visual content to analyze the semantics across modalities. Ye et al. (2018) integrates audio and visual modalities to predict the climax of an advertisement (i.e. stress levels) using sentiment annotations.

3 Tasks & Data

We aim to analyze the political ideology of ads consisting of image and text, and the type of the ad sponsor for the first time. To this end, we present two new binary classification tasks motivated by related studies in political communication (Grigsby, 2008; Fowler et al., 2020b):

- **Task 1: *Conservative/Liberal*** The aim is to label an ad according to the political party that sponsored the ad either as *Conservative* (i.e. assuming that the dominant ideology of the Republican Party is conservatism), or *Liberal* (i.e. assuming that the dominant ideology of the Democratic Party is social liberalism) (Grigsby, 2008);
- **Task 2: *Political Party/Third-Party*** The goal is to classify an ad according to the type of the organization that sponsored the ad. We distinguish between ads sponsored by official political parties and non-political organizations, such as businesses and non-profit groups, following Fowler et al. (2020b).

To the best of our knowledge, no datasets are available for modeling these two tasks. Therefore, we develop two new publicly available datasets consisting of political ads and ideology/sponsor type labels from the U.S.. We opted to use data only from the U.S. because its Federal Election Commission[4] (FEC) provides publicly available information of political ads sponsors such as official political parties (e.g. Democratic, Republican) via their FEC ID; and third-party organizations can be identified via their Employer Identification Number[5] (EIN) suitable for our study.

3.1 Collecting Online Political Ads

We use the public Google transparency report platform[6] to collect political ads. This platform contains information on verified political advertisers (i.e. sponsors) and provides links to actual political ads from Google Ad Services.

We collect all U.S. available data from the Google platform consisting of ads published from May 31, 2018 up to October 11, 2020 (note that

---

[4] `https://www.fec.gov/`

[5] `https://www.irs.gov/businesses/small-businesses-self-employed/do-you-need-an-ein`

[6] `https://transparencyreport.google.com/political-ads/region/US`

Sample Ad	
Image Text	FIGHTING FOR WORKING FAMILIES, FOR GOOD JOBS, AND FAIR PAY. PAID FOR BY DEFAZIO FOR CONGRESS
Densecap	the man is wearing glasses, a man holding a red tie, the background is blue

Table 2: Example of text, and visual information extracted from a sample Ad.

there is no data prior to 2018). This corresponds to a total of 168,146 image ads. Each ad is associated with a URL that links to its summary metadata consisting of a URL to the original image file and sponsor information, i.e. name and FEC ID, state elections registration or EIN ID.[7]

We scrape all available image files resulting into a total of 158,599 ads which corresponds to 94.32% of all ads in the Google database. The rest of the ads were either not available due to violations to Google's Advertising Policy, the summary metadata was missing, or the file URL was not included in the metadata.

3.2 Extracting Text and Visual Information

Before, we label the ads with ideology and sponsor type, we extract two types of information from the images: (1) the text contained in each ad (Image Text; IT) using the Google Vision API;[8] and (2) the descriptive caption or denscap (D) of the image using the DenseCap API,[9] following the method proposed by Kalra et al. (2020) for commercial ad classification. This way, we obtain both the actual text appearing on the ad and the textual descriptions of the ad such as entities in the images, their characteristics and relationships. Table 2 shows an example of an ad consisting of an image, text information and the densecap.

We use the textual and visual information to eliminate all duplicate images by comparing the URL of the image, its text and densecap. Finally, we filter out all ads that contain non-English text (i.e. IT).[10] This results in 15,116 unique ads from 665 unique ad sponsors.

3.3 Labeling Ads with Political Ideology

Our aim is to label political ads as *Conservative* or *Liberal* (see Task 1 description). First, we retrieve all the ad sponsors and their corresponding ads that are available in the Google Ads database. Official political committees associated with the Democratic or Republican parties are identified by their FEC ID (included in the sponsor's information in the Google database). However, the name of the political party associated with a sponsor is not available in the Google database. Thus, we query the FEC database to obtain the affiliation for all committees of the Democratic and Republican parties (e.g. Donald J. Trump for President, Inc.). Then, we compare this information with the Google database (FEC ID and exact name), to assign the corresponding affiliation to the sponsors. For example an ad sponsored by the 'Donald J. Trump for President, Inc.' official committee is labeled as *Republican* and subsequently as *Conservative* (in a similar way we label ads for the *Liberal* class).

In total, we collect 242 unique sponsors corresponding to 5,548 ads. *Liberal* ads represent the 39% of the total ads and the rest are *Conservative* (61%).

3.4 Labeling Ads with Sponsor Type

We first label all ads from sponsors that have an associated FEC ID in the Google database as *Political Party*. These sponsors correspond to official political committees affiliated with the Democratic or Republican parties (e.g. Biden for President).

Third-party sponsors of political ads consist of groups not officially associated to any political party such as not-for-profit organizations (e.g. NRDC Action Fund) and businesses (Fowler et al., 2020b). This type of sponsors are identified with their EIN ID (included in the Google database). Thus, we label all ads linked to an EIN ID as *Third-Party*. We collected a total of 15,116 ads where 37% corresponds to *Political Party* and 63% corresponds to *Third-Party*.

---

[7] All ad sponsors must apply for eligibility verification in order to publish political ads on Google platforms - `https://support.google.com/displayvideo/answer/9014141`

[8] `https://cloud.google.com/vision/docs/ocr`

[9] `https://deepai.org/machine-learning-model/densecap`

[10] `https://pypi.org/project/langdetect/`

T1: Liberal/Conservative				
	Train	**Dev**	**Test**	**Total**
C	2,576 (58%)	369 (69%)	453 (75%)	3,398 (61%)
L	1,835 (42%)	165 (31%)	150 (25%)	2,150 (39%)
All	4,411 (79.5%)	534 (9.6%)	603 (10.9%)	5,548 (100%)
Start	05-31-18	02-01-20	07-04-20	-
End	01-30-20	06-30-20	10-10-20	-
T2: Political Party/Third-Party				
	Train	**Dev**	**Test**	**Total**
PP	4,663 (39%)	324 (21%)	561 (37%)	5,548 (37%)
TP	7,427 (61%)	1,188 (79%)	953 (63%)	9,568 (63%)
All	12,090 (80%)	1,512 (10%)	1,514 (10%)	15,116 (100%)
Start	05-31-18	04-14-20	07-20-20	-
End	04-13-18	07-19-20	10-11-20	-

Table 3: Data set statistics for Task 1: *Conservative* (C)/ *Liberal* (L), and Task 2: *Political Party* (PP)/*Third-Party* (TP).

Avg. Tokens (Train/Dev/Test)			
Task	**IT**	**D**	**IT+D**
T1	17.1/16.5/17.1	38.3/39.9/36.9	55.4/56.4/54.0
T2	16.2/17.6/19.2	36.7/38.9/37.2	52.9/56.5/56.4

Table 4: Average number of tokens in image text (IT), densecaps (D) and both (IT+D) for sponsor ad ideology (T1) and type (T2) prediction.

3.5 Data Splits

We split both datasets chronologically into train (80%), development (10%), and test (10%) sets. Table 3 shows the dataset statistics and splits for each task.

3.6 Data Preprocessing

Text We normalize the text from the image (IT) and the densecap (D) by lower-casing, and replacing all URLs and person names with a placeholder token. To identify the person names we use the Stanford NER Tagger (Finkel et al., 2005). Also, we replace tokens that appear in less than five ads with an 'unknown' token. We tokenize the text using the NLTK tokenizer (Bird et al., 2009). Table 4 shows the average number of tokens in IT and D for each data split.

Image Each image is resized to (300 × 300) pixels represented by red, green and blue color values. Each color channel is an integer in the range [0, 255]. The pixel values of all images are dived by 255 to normalize them in the range [0, 1].

4 Predictive Models

We experiment with textual, visual and multimodal models for political ad classification.

4.1 Linear Baselines

As baseline models, we use logistic regression with bag of n-grams and L2 regularization using (1) the image text (LR_{IT}); (2) densecap (LR_{D}); and (3) their concatenation (LR_{IT+D}) for representing each ad.

4.2 BERT

We also test three models proposed by Kalra et al. (2020) for generic ad classification demonstrating state-of-the-art performance. The models are based on Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) using a combination of the image text and the densecap. We follow a similar approach and fine-tune BERT for predicting the corresponding class in each task by adding an output dense layer for binary classification that receives the 'classification' [CLS] token as input. We use three types of inputs for each ad: (1) image text (BERT_{IT}); (2) densecap (BERT_{D}); and (3) their concatenation (BERT_{IT+D}).

4.3 EfficientNet

EfficientNet (Tan and Le, 2019) is a family of Convolutional Neural Network (CNN) (LeCun et al., 1995) models which has achieved state-of-the-art accuracy on ImageNet (Deng et al., 2009). In particular, we use EfficientNet-B3 and fine-tune it on political ad classification by adding an output dense layer for each binary classification task.

4.4 BERT+EffN

We finally test two multimodal models by combining: (1) BERT_{IT} and EfficientNet (BERT_{IT}+EffN); and (2) BERT_{IT+D} and EfficientNet (BERT_{IT+D}+EffN). We concatenate the text representation obtained by BERT and the visual information from EfficientNet into a $768 + 1536$ dimensional vector from BERT and EfficientNet respectively. This vector is then passed to an output layer for binary classification. We fine-tune the entire architecture for each task.

5 Experimental Setup

We select the hyperparameters for all neural models using early stopping by monitoring the validation binary cross-entropy loss, and we estimate the

T1: Conservative/Liberal			
Model	**P**	**R**	**F1**
Majority	50.00 (0.00)	37.56 (0.00)	42.90 (0.00)
LR$_{D}$	55.76 (0.85)	54.91 (0.89)	54.85 (1.12)
LR$_{IT}$	78.38 (0.70)	71.99 (0.56)	72.65 (0.73)
LR$_{IT+D}$	72.57 (1.03)	71.52 (0.62)	71.99 (0.79)
Kalra et al. (2020)			
BERT$_{D}$	59.40 (0.78)	57.77 (0.98)	57.64 (1.52)
BERT$_{IT}$	72.88 (0.24)	73.46 (0.16)	73.16 (0.20)
BERT$_{IT+D}$	78.62 (3.14)	74.08 (2.81)	75.49 (3.01)
EfficientNet	69.02 (3.48)	67.87 (1.23)	68.15 (1.89)
Ours			
BERT$_{IT}$+EffN	74.99 (1.23)	72.01 (2.27)	73.02 (2.07)
BERT$_{IT+D}$+EffN	**80.24 (0.06)**	**74.59 (1.70)**	**75.76 (2.19)**

Table 5: Macro Precision (P), Macro Recall (R), and Macro F1-Score (F1) for political ideology prediction (± std. dev. for 3 runs). Best results are in bold.

T2: Political Party/Third-Party			
Model	**P**	**R**	**F1**
Majority	50.00 (0.00)	31.47 (0.00)	38.62 (0.00)
LR$_{D}$	53.60 (0.72)	53.40 (0.65)	53.11 (0.58)
LR$_{IT}$	84.02 (0.14)	85.04 (0.31)	84.47 (0.18)
LR$_{IT+D}$	86.46 (0.13)	86.63 (0.09)	86.54 (0.05)
Kalra et al. (2020)			
BERT$_{D}$	56.50 (0.89)	56.31 (0.78)	53.45 (1.26)
BERT$_{IT}$	85.57 (0.86)	86.42 (2.01)	85.86 (1.23)
BERT$_{IT+D}$	87.00 (0.89)	86.81 (0.83)	86.90 (0.86)
EfficientNet	53.27 (2.86)	53.93 (2.40)	51.53 (5.46)
Ours			
BERT$_{IT}$+EffN	**87.02 (2.74)**	85.81 (0.20)	86.29 (1.11)
BERT$_{IT+D}$+EffN	86.78 (0.03)	**88.18 (1.10)**	**87.36 (0.39)**

Table 6: Macro Precision (P), Macro Recall (R), and Macro F1-Score (F1) for sponsor type prediction (± std. dev. for 3 runs). Best results are in bold.

class weights using the ’balanced’ heuristic (King and Zeng, 2001) for each task, as both datasets are imbalanced. BERT and EfficientNet models use ADAM optimizer (Kingma and Ba, 2014), and experiments use 1 GPU (Nvidia V100).

LR For LR we use bag of n-grams with $n = (1, 3)$, $n \in \{(1,1),(1,2),(1,3)\}$ weighted by TF.IDF and L2 regularization. The average training time is 30 seconds.

BERT We fine-tune BERT for 20 epochs and choose the epoch with the lowest validation loss. We use the pre-trained base-uncased model for BERT (Vaswani et al., 2017; Devlin et al., 2019) from HuggingFace implementation (12-layer 768-dimensional) trained on English Wikipedia (Wolf et al., 2019). The maximal sequence length is 512 tokens. We fine-tune BERT for 2 epochs and learning rate $\eta = 2e^{-5}$ for ideology prediction; and $\eta = 1e^{-5}$ for advertiser type prediction with $\eta \in \{1e^{-5}, 2e^{-5}, 3e^{-5}, 4e^{-5}\}$. The average training time is 8.1 minutes.

EfficientNet We use EfficientNet-B3 with Noisy-Student weights (Xie et al., 2020). For ideology prediction, we first freeze the layers of the EfficientNet (Tan and Le, 2019) model and train it for 11 epochs with learning rate $\eta = 1e^{-3}$ to learn the parameters of the output layer. We then unfreeze and train the whole network for another 30 epochs with $\eta = 1e^{-4}$, as it has been shown that unfreezing the CNN during the latter stages of training improves the performance of the network (Faghri et al., 2017). For predicting the type of sponsor, we train for 45 epochs and $\eta = 1e^{-2}$ keeping the EfficientNet layers frozen. Unfreezing the base model did not result into lower validation loss. We use dropout rate of 0.2 before passing the output of EfficientNet to the classification layer. The average training time is 37.8 minutes.

BERT+EffN For ideology prediction, we freeze all the layers of the pre-trained models (BERT and EfficientNet) apart from the classification layer and train for 27 epochs with $\eta = 1e^{-3}$. We then fine-tune BERT for 30 epochs with $\eta = 1e^{-5}$. For sponsor type prediction, we freeze all EfficientNet layers and fine-tune BERT for 30 epochs with $\eta = 2e^{-6}$. We train in stages to ensure that the parameters of each part of the model (textual and visual) are properly updated (Kiela et al., 2019). The average training time is 56.65 minutes.

6 Results

This section presents the experimental results for the two predictive tasks, political ideology and sponsor type prediction (§3) using the methods described in §4. We evaluate our models using macro precision, recall and F1 score since the data in both tasks is imbalanced. Note that for all models we report the average and standard deviation over three runs using different random seeds. We also report the majority class baseline for each task.

6.1 Predictive Performance

Task 1: Conservative/Liberal Table 5 shows the results for the political ideology prediction. We first observe that BERT$_{IT}$ (73.16%) which uses as input information the image text outperforms BERT$_{D}$ (57.64%) and EfficientNet (68.15%) in

(a) True: *Lib* - Pred: *Cons* (b) True: *Cons* - Pred: *Lib* (c) True: *PP* - Pred: *TP* (d) True: *TP* - Pred: *PP*

Figure 1: Examples of ads with their true and predicted labels Lib (Liberal), Cons (Conservative), PP (Political Party), TP (Third-Party).

macro F1. This suggests that the text shown on a political ad is the dominant medium for conveying its main message, corroborating findings in related research on commercial ads (Dey et al., 2019; Kalra et al., 2020).

Moreover, combining image text and densecap (BERT$_{IT+D}$), leads to higher performance, than using only image text (BERT$_{IT}$), i.e. 75.49% and 73.16% F1 respectively. This indicates that the combination of textual with visual information (in the form of image descriptions) improves the model performance.

Finally, using all visual information sources, i.e. densecaps and image representation from EfficientNet (BERT$_{IT+D}$+EffN), further improves performance achieving the highest macro F1 (75.76%) across models, followed by BERT$_{IT+D}$ (75.49%).

Task 2: Political-Party/Third-Party Table 6 shows the results for the sponsor type prediction. The best overall performance is obtained by BERT$_{IT+D}$+EffN (87.36%) which combines both image and textual information. BERT$_{IT+D}$ (86.90%) and LR$_{IT+D}$ (86.54%) follow very closely. By inspecting our data, we identified the presence of noise in image text, particularly sentences are interrupted by logos and other aesthetic elements. This negatively affects the performance of BERT because such models are usually pre-trained on 'cleaner' generic corpora (Kumar et al., 2020). On the other hand, LR models trained from scratch can adapt to the noisy text (see § 6.2 for error analysis).

Overall, our results in both tasks suggest that text is a stronger modality for inferring the political ideology and sponsor type of political ads compared to visual information extracted from the images. However, integrating visual information in the form of text descriptions (densecaps) or representations obtained by pre-trained image classification models, enhances model performance.

6.2 Error Analysis

We further perform an error analysis to examine the behavior of our best performing models (BERT$_{IT+D}$+EffN and BERT$_{IT+D}$) and identify potential limitations.

The ad shown in Fig. 1 (a) was misclassified as *Conservative* by BERT$_{IT+D}$ and BERT$_{IT+D}$+EffN. This particular ad requires common knowledge of social issues (e.g. inadequate health support) that are often discussed in political campaigns to inform voters about a party's views on the issue (Scammell and Langer, 2006). This makes the classification task difficult for the models since it requires contextual knowledge. Incorporating external relevant knowledge to the models (e.g. political speeches, interviews or public meetings) might improve performance (Lin et al., 2018).

The ad depicted in Fig. 1 (b) was misclassified by BERT$_{IT+D}$ and BERT$_{IT+D}$+EffN as *Conservative*. After analyzing the densecap descriptions, we found that this information tends to be noisy. For this particular example, it contains descriptions such as 'a man is holding a horse', 'the sign is blue', 'a blue and white stripe shirt', and 'a man wearing a hat'. In fact, BERT$_{IT}$, which only takes the image text into account, classified this ad correctly as *Conservative*. Improving the quality of the image descriptions (e.g. pre-training on advertising or political images, capturing specific attributes such as 'military hat') might be beneficial for these models.

Fig. 1 (c) shows an example of a *Political Party* ad misclassified by BERT$_{IT+D}$+EffN as *Third-Party*. The ad contains the following text:

> *WE CAN'T LET <person> WIN!*
>
> *VOTE EARLY*

The message has a confrontational and divisive tone that is common in *Third Party* ads (Edelson

Liberal		Conservative	
Feature	r	Feature	r
necessary	0.197	senate	0.271
end	0.196	republican	0.196
prohibited	0.190	!	0.176
approx	0.186	conservative	0.127
contrib	0.181	national	0.116
void	0.177	committee	0.112
values	0.173	petition	0.109
prz	0.161	border	0.102
subj	0.156	taxes	0.099
make	0.156	radical	0.098
win	0.144	sign	0.096
place	0.140	stop	0.094
beer	0.139	states	0.093

Table 7: Feature correlations with *Conservative/Liberal* Ads, sorted by Pearson correlation (r). All correlations are significant at $p < .01$, two-tailed t-test.

Political Party		Third-Party	
Feature	r	Feature	r
congress	0.365	state	0.193
vote	0.308	learn	0.181
senate	0.292	champion	0.175
!	0.269	senator	0.166
president	0.248	thank	0.153
committee	0.236	action	0.147
candidate	0.223	congressman	0.130
republican	0.208	urge	0.129
authorized	0.208	protect	0.128
donate	0.202	access	0.119
join	0.199	award	0.117
<url>	0.187	american	0.116
$	0.180	?	0.113

Table 8: Feature correlations with Political Party/*Third-Party* Ads, sorted by Pearson correlation (r). All correlations are significant at $p < .01$, two-tailed t-test.

et al., 2019), but is typically used as a political tactic for negative campaigning (Skaperdas and Grofman, 1995; Gandhi et al., 2016; Haselmayer, 2019).

Finally, Fig. 1 (d) shows an example of a *Third-Party* ad misclassified as *Political Party* by BERT$_{IT+D}$+EffN. The text content promotes voter participation (e.g. *Vote*), a characteristic of *Political Party* advertising (see Table 8). However, one of the aims of the *Third-Party* advertising is precisely to encourage voting and activism (Dommett and Temple, 2018).

There is a considerable difference between the models using visual information only (LR$_D$, BERT$_D$, EfficientNet), and those that also use the ad text as input (IT, IT+D). Our intuition is that models get confused by the appearance of shapes, colors and other aesthetic features that are domain specific and appear frequently in political advertisements (Sartwell, 2011). For instance, several ads that belong to the *Third-Party* category, include buttons linking to websites (see Fig, 1 (c), (d)). However, *Political Party* ads, also make use of these type of buttons to link users to donation or informative websites (Edelson et al., 2019).

7 Linguistic Analysis

We perform an analysis based on our new data set to study the linguistic characteristics of political ads. We first analyze the specific features of each class for both tasks. For this purpose, we use a method introduced by Schwartz et al. (2013) to analyze uni-gram features from image text (see §4) using univariate Pearson correlation. Features are normalized to sum up to unit for each ad. For each feature, we compute correlations independently between its distribution across ads and its label (*Conservative/Liberal*), or *Political Party/Third Party*).

7.1 Conservative vs. Liberal

Table 7 presents the top unigrams correlated with *Liberal* and *Conservative* ads. We first notice that the top words in the *Conservative* category are closely related to its ideology such as 'conservative' and 'republican'. Other prominent terms in these categories are words related to current political issues, such as immigration (e.g. 'border') and taxation (e.g. 'taxes'). In fact, these are examples of emotionally evocative terms (e.g. anger about taxes) that are frequently used in political campaigns to influence voters (Brader, 2005).

Top terms of *Liberal* ads include 'necessary', 'end','values', and 'win'. For example, the following ads belong to the *Liberal* class:

> *I'm supporting <person> because he has the same* ***values*** *that I do and he's an honest person.*
>
> *<person> FOR CONGRESS*
>
> *To* ***End*** *Gun Violence*

These are examples of ads containing a combination of moral and controversial topics (e.g. gun regulation) which are typical characteristics of political advertising (Kumar and Pathak, 2012).

7.2 Political Party vs. Third-Party

Table 8 shows the top unigram features correlated with the sponsor type of an ad (*Political*

Party/Third-Party). We observe that some top terms in the *Political Party* class also belong to the top terms of the political ideology task (see Table 7) such as 'committee', 'republican' and 'senate'. Messages calling for vote and donation support ('vote', 'donate', '$') are also prevalent in *Political Party* ads (Fulgoni et al., 2016), as in the next example (See Fig. 1 (b)):

> *Making sure our veterans*
>
> *get the care they've earned*
>
> ***VOTE** FOR <person>*

On the other hand, top features from the *Third-Party* category (e.g. 'action', 'protect') share common characteristics with the rhetoric used by media outlets focused on promoting specific political messaging (Edelson et al., 2019; Dommett and Temple, 2018). Many of these ads direct people to websites to read about a particular topic. For example:

> *Is <person> HIDING ANTI-GUN VIEWS? **Learn** More*

This ad belongs to the *Third-Party* class and points the viewer to an external website for reading further details.

8 Conclusion

We have presented the first study in NLP for analyzing the language of political ads motivated by prior studies in political communication. We have introduced two new publicly available datasets containing political ads from the U.S. in English labeled by (1) the ideology of the sponsor (*Conservative/Liberal*); and (2) the sponsor type (*Political Party/Third Party*). We have defined both tasks as advertisement-level binary classification and evaluated a variety of approaches, including textual, visual and multimodal models reaching up to 75.76 and 87.36 macro F1 in each task respectively.

In the future, we aim to incorporate other modalities such as speech, and video, and explore other methods of acquiring and integrating multimodal information. In addition, we aim to extend our work for analyzing political advertising discourse across different regions, languages and platforms.

Acknowledgments

We would like to thank Kate Dommett, Alexandra Boutopoulou, Mali Jin, Katerina Margatina, George Chrysostomou, Peter Vickers, Emily Lau, and all reviewers for their valuable feedback. DSV is supported by the Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by the UK Research and Innovation grant EP/S023062/1. NA is supported by a Leverhulme Trust Research Project Grant.

Ethics Statement

Our work complies with the Terms of Service of the Google Political Ads Dataset.[11] We provide, for reproducibility purposes, the list of ad IDs and corresponding labels used for each task, as well as the data splits (train, development, test). All data used in this paper is in English. The ads information can be retrieved from Google according to their policy.

References

Karuna Ahuja, Karan Sikka, Anirban Roy, and Ajay Divakaran. 2018. Understanding visual ads by aligning symbols and objects using co-attention. *arXiv preprint arXiv:1807.01448*.

Nick Anstead, João Carlos Magalhães, Richard Stupart, and Damian Tambini. 2018. Political advertising on facebook: The case of the 2017 united kingdom general election. In *American Political Science Association, Annual Meeting, Boston*.

Chongyang Bai, Haipeng Chen, Srijan Kumar, Jure Leskovec, and VS Subrahmanian. 2020. M2p2: Multimodal persuasion prediction using adaptive fusion. *arXiv preprint arXiv:2006.11405*.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2109–2116, Minneapolis, Minnesota. Association for Computational Linguistics.

Sumit Bhatia and Deepak P. 2018. Topic-specific sentiment analysis can help identify political ideology. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 79–84, Brussels, Belgium. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

---

[11] `https://console.cloud.google.com/marketplace/product/transparency-report/google-political-ads?pli=1`

Ted Brader. 2005. Striking a responsive chord: How political ads motivate and persuade voters by appealing to emotions. *American Journal of Political Science*, 49(2):388–405.

Robyn Caplan and Danah Boyd. 2016. Who controls the public sphere in an era of algorithms. *Mediation, Automation, Power*, pages 1–19.

Danni Chen, Kunwoo Park, and Jungseock Joo. 2020. Understanding gender stereotypes and electoral success from visual self-presentations of politicians in social media. In *Joint Workshop on Aesthetic and Technical Quality Assessment of Multimedia and Media Analytics for Societal Trends*, pages 21–25.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Arka Ujjal Dey, Suman Kumar Ghosh, Ernest Valveny, and Gaurav Harit. 2019. Beyond visual semantics: Exploring the role of scene text in image understanding. *arXiv preprint arXiv:1905.10622*.

Katharine Dommett and Luke Temple. 2018. Digital campaigning: The rise of facebook and satellite campaigns. *Parliamentary Affairs*, 71(suppl_1):189–202.

Laura Edelson, Shikhar Sakhuja, Ratan Dey, and Damon McCoy. 2019. An analysis of united states online political advertising transparency. *arXiv preprint arXiv:1902.04385*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

Erika Franklin Fowler, Michael M Franz, Gregory J Martin, Zachary Peskowitz, and Travis N Ridout. 2020a. Political advertising online and offline. *American Political Science Review*, pages 1–20.

Erika Franklin Fowler, Michael M Franz, and Travis N Ridout. 2020b. The blue wave: Assessing political advertising trends and democratic advantages in 2018. *PS: Political Science & Politics*, 53(1):57–63.

Gian M Fulgoni, Andrew Lipsman, and Carol Davidsen. 2016. The power of political advertising: Lessons for practitioners: How data analytics, social media, and creative strategies shape us presidential election campaigns. *Journal of Advertising Research*, 56(3):239–244.

Amit Gandhi, Daniela Iorio, and Carly Urban. 2016. Negative advertising and political competition. *The Journal of Law, Economics, and Organization*, 32(3):433–477.

Sean M Gerrish and David M Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*.

Ellen Grigsby. 2008. *Analyzing Politics: An Introduction to Political Science*. Cengage Learning.

Martin Haselmayer. 2019. Negative campaigning and its consequences: a review and a look ahead. *French Politics*, pages 1–18.

X. Huang and A. Kovashka. 2016. Inferring visual persuasion via body language, setting, and deep features. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 778–784.

Z. Hussain, M. Zhang, X. Zhang, K. Ye, C. Thomas, Z. Agha, N. Ong, and A. Kovashka. 2017. Automatic understanding of image and video advertisements. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1100–1110.

Shanto Iyengar and Markus Prior. 1999. Political advertising: what effect on commercial advertisers? *Retrieved form http://web. stanford. edu/˜ siyengar/research/papers/advertising. html*.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1113–1122, Baltimore, Maryland. Association for Computational Linguistics.

Bengt Johansson and Christina Holtz-Bacha. 2019. From analogue to digital negativity: Attacks and counterattacks, satire, and absurdism on election posters offline and online. In *Visual Political Communication*, pages 99–118. Springer.

J. Johnson, A. Karpathy, and L. Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4565–4574.

Kristen Johnson and Dan Goldwasser. 2018. Classification of moral foundations in microblog political discourse. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 720–730, Melbourne, Australia. Association for Computational Linguistics.

J. Joo, W. Li, F. F. Steen, and S. Zhu. 2014. Visual persuasion: Inferring communicative intents of images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 216–223.

J. Joo, F. F. Steen, and S. Zhu. 2015. Automated facial trait judgment and election outcome prediction: Social dimensions of face. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3712–3720.

Jungseock Joo and Zachary C Steinert-Threlkeld. 2018. Image as data: Automated visual content analysis for political science. *arXiv preprint arXiv:1810.01544*.

Lynda L Kaid and Christina Holtz-Bacha. 2006. Television advertising and democratic systems around the world. *The Sage handbook of political advertising*, pages 445–457.

Lynda Lee Kaid and Monica Postelnicu. 2005. Political advertising in the 2004 election: Comparison of traditional television and internet messages. *American Behavioral Scientist*, 49(2):265–278.

Kanika Kalra, Bhargav Kurma, Silpa Vadakkeeveetil Sreelatha, Manasi Patwardhan, and Shirish Karande. 2020. Understanding advertisements with BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7542–7547, Online. Association for Computational Linguistics.

Kuralay Kenzhekankyzy Kenzhekanova. 2015. Linguistic features of political discourse. *Mediterranean Journal of Social Sciences*, 6(6 S2):192.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. 2019. Supervised multimodal bitransformers for classifying images and text. *arXiv preprint arXiv:1909.02950*.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political analysis*, 9(2):137–163.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Daniel Kriess and Bridget Barrett. 2020. Democratic tradeoffs: Platforms and political advertising. *Ohio State Technology Law Journal*, 16(2):494–518.

Vivek Kulkarni, Junting Ye, Steve Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3518–3527, Brussels, Belgium. Association for Computational Linguistics.

Alok Kumar and Pramod Pathak. 2012. Political advertising in india: a perspective. *Management Insight*, 8(1):15–29.

Ankit Kumar, Piyush Makhija, and Anuj Gupta. 2020. Noisy text data: Achilles' heel of BERT. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 16–21, Online. Association for Computational Linguistics.

Richard R Lau, Lee Sigelman, Caroline Heldman, and Paul Babbitt. 1999. The effects of negative political advertisements: A meta-analytic assessment. *American Political Science Review*, pages 851–875.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer.

Y. Lin, J. Hoover, G. Portillo-Wightman, C. Park, M. Dehghani, and H. Ji. 2018. Acquiring background knowledge to improve moral value prediction. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 552–559.

Bin Liu, Anmol Sheth, Udi Weinsberg, Jaideep Chandrashekar, and Ramesh Govindan. 2013. Adreveal: improving transparency into online targeted advertising. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pages 1–7.

Antonios Maronikolakis, Danae Sánchez Villegas, Daniel Preotiuc-Pietro, and Nikolaos Aletras. 2020. Analyzing political parody in social media. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4373–4384, Online. Association for Computational Linguistics.

Juan Carlos Medina Serrano, Orestis Papakyriakopoulos, and Simon Hegelich. 2020. Exploring political ad libraries for online advertising transparency: Lessons from germany and the 2019 european elections. In *International Conference on Social Media and Society*, pages 111–121.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: Political ideology prediction of Twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 729–740, Vancouver, Canada. Association for Computational Linguistics.

Kevin Reschke and Pranav Anand. 2012. POLITICAL-ADS: An annotated corpus for modeling event-level evaluativity. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pages 84–88, Jeju, Korea. Association for Computational Linguistics.

Travis N Ridout, Erika Franklin Fowler, and John Branstetter. 2010. Political advertising in the 21st century: The rise of the youtube ad. In *APSA 2010 Annual Meeting Paper*.

Crispin Sartwell. 2011. *Political aesthetics*. Cornell University Press.

Margaret Scammell and Ana I Langer. 2006. Political advertising: why is it so boring? *Media, culture & society*, 28(5):763–784.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 91–101, Seattle, Washington, USA. Association for Computational Linguistics.

Stergios Skaperdas and Bernard Grofman. 1995. Modeling negative campaigning. *American Political Science Review*, 89(1):49–61.

Pavel Skorupa and Tatjana Dubovičienė. 2015. Linguistic characteristics of commercial and social advertising slogans. *Coactivity: Philology, Educology/Santalka: Filologija, Edukologija*, 23(2):108–118.

Till Speicher, Muhammad Ali, Giridhari Venkatadri, Filipe Ribeiro, George Arvanitakis, Fabrício Benevenuto, Krishna Gummadi, Patrick Loiseau, and Alan Mislove. 2018. Potential for discrimination in online targeted advertising. In *FAT 2018-Conference on Fairness, Accountability, and Transparency*, volume 81, pages 1–15.

Peter Stefanov, Kareem Darwish, Atanas Atanasov, and Preslav Nakov. 2020. Predicting the topical stance and political leaning of media using tweets. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 527–537, Online. Association for Computational Linguistics.

Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

Christopher Thomas and Adriana Kovashka. 2018. Persuasive faces: Generating faces in advertisements. *arXiv preprint arXiv:1807.09882*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Darrell M West. 2017. *Air wars: television advertising and social media in election campaigns, 1952-2016*. CQ Press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Nan Xi, Di Ma, Marcus Liou, Zachary C Steinert-Threlkeld, Jason Anastasopoulos, and Jungseock Joo. 2020. Understanding the political ideology of legislators from social media images. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 726–737.

Q. Xie, M. T. Luong, E. Hovy, and Q. V. Le. 2020. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

Keren Ye, Kyle Buettner, and Adriana Kovashka. 2018. Story understanding in video advertisements. *arXiv preprint arXiv:1807.11122*.

Keren Ye and Adriana Kovashka. 2018. Advise: Symbolism and external knowledge for decoding advertisements. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 837–855.

Mingda Zhang, Rebecca Hwa, and Adriana Kovashka. 2018. Equal but not the same: Understanding the implicit relationship between persuasive images and text. *arXiv preprint arXiv:1807.08205*.

Do Language Models Perform Generalizable Commonsense Inference?

Peifeng Wang[1,2], **Filip Ilievski**[2], **Muhao Chen**[1,2], **Xiang Ren**[1,2]
[1]Department of Computer Science, University of Southern California
[2]Information Sciences Institute, University of Southern California
{peifengw,muhaoche,xiangren}@usc.edu, ilievski@isi.edu

Abstract

Inspired by evidence that pretrained language models (LMs) encode commonsense knowledge, recent work has applied LMs to automatically populate commonsense knowledge graphs (CKGs). However, there is a lack of understanding on their generalization to multiple CKGs, unseen relations, and novel entities. This paper analyzes the ability of LMs to perform generalizable commonsense inference, in terms of *knowledge capacity*, *transferability*, and *induction*. Our experiments with these three aspects show that: (1) LMs can adapt to different schemas defined by multiple CKGs but fail to reuse the knowledge to generalize to new relations. (2) Adapted LMs generalize well to unseen subjects, but less so on novel objects. Future work should investigate how to improve the transferability and induction of commonsense mining from LMs.[1]

1 Introduction

Large-scale commonsense knowledge graphs (CKGs), like ConceptNet (Speer et al., 2017) and ATOMIC (Sap et al., 2019), store structured knowledge that can benefit various knowledge-driven applications. Given the usefulness of CKGs, but also their inability to flexibly provide information, (Paulheim, 2018), recent work has paid much attention to populating CKGs with commonsense knowledge mined from pretrained language models (LMs) (Wang et al., 2020c; Bosselut et al., 2019). Enhancing the knowledge of CKGs is essential to support reasoning on downstream tasks (Talmor et al., 2019; Wang et al., 2020b; Young et al., 2018).

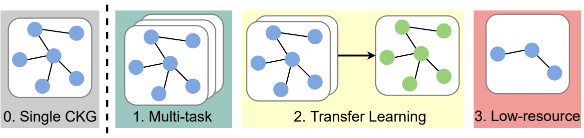

Figure 1: Unlike previous studies that adapt LM on one single CKG (0), we investigate LM's three aspects of **generlizability**: (1) *knowledge capacity* by multi-task learning, (2) *transferability* by transfer learning and (3) *induction* by controlled low-resource learning.

The task of completing CKGs has typically been posed as *commonsense knowledge inference*, where the goal is to predict the object of a fact triplet, given its *subject* and a *relation* (*predicate*) (Petroni et al., 2019; Bosselut et al., 2019). Commonsense inference techniques, such as COMET (Bosselut et al., 2019), typically fine-tune an LM, like GPT (Radford et al., 2018), over the training set from a single CKG. While such methods are able to dynamically enhance the completeness of CKGs, their application so far has been limited to the relation set of the source (training) CKG (Da et al., 2021). In addition, the generated object concepts are found to be largely biased towards the ones in the training set (Wang et al., 2020a). It remains unclear to which extent LMs can generalize to multiple CKGs, new relations, and novel objects. To this end, we pose the question: *do language models perform generalizable commonsense inference?*

To answer this question, we study three aspects of the LM generalizability for commonsense inference, namely: knowledge capacity, transferability, and induction. To measure the *knowledge capacity* ability of LMs, we examine whether LMs can be adapted to multiple CKGs simultaneously, and tested on each of the CKGs. We test their *transferability* by assessing whether an initial adaptation of a LM on multiple source CKGs can reduce the effort on further adapting it to a new CKG. The *inductive* power of LMs is measured by varying the overlap between the objects in the training and test splits of a CKG. The overview of our analysis is depicted in Figure 1. Our results show that LMs are able to infer knowledge for multiple CKGs simultaneously without loss of performance on the

---

[1] The code is avaiable at `https://github.com/wangpf3/LM-for-CommonsenseInference`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3681–3688
August 1–6, 2021.

target inference task, though the transferability of knowledge across tasks is limited. In addition, we observe that the inductive power of LMs for commonsense inference relies heavily on whether an object is observed during training.

2 Analysis Setup

To shed light on the LM's generalizability for commonsense inference, we investigate: whether LMs have the capability to adapt to multiple CKGs (*Q1: capacity*), whether LMs can reuse the knowledge learned from source CKGs to efficiently adapt to a target CKG (*Q2: transferability*), and whether LMs can predict unseen objects or mainly repeat the observed ones (*Q3: induction*). In this Section, we define the task, the CKGs we consider, our experimental settings, and relate to prior studies.

2.1 Task Formulation

Following Hwang et al. (2020); Da et al. (2021), we formalize *commonsense inference* as a task of predicting the object of a triplet, given a pair of *(subject, relation)* as input. The subject s and the object o are both expressed as free-form phrases, while the relation r is a predefined relation type from the CKG. A training example from ConceptNet could have `(go to a concert, MotivatedByGoal)` as input, and `listen to music` as output. Assuming that a CKG is given, the goal is to leverage the commonsense triplets in the CKG as training examples to adapt the LM for commonsense inference.

2.2 CKG Datasets

We consider three large and popular CKGs, with different foci:(1) **ConceptNet**'s broad set of commonsense knowledge includes taxonomic (e.g., `IsA`), utility (e.g., `UsedFor`), and temporal knowledge (e.g., `HasPrerequisite`). It combines crowdsourced knowledge with that from existing sources, such as WordNet. We use its ConceptNet-100K subset, collected by Li et al. (2016). (2) **TupleKB** (Dalvi Mishra et al., 2017) focuses on scientific commonsense knowledge like `(salt, dissolve in, water)`. It is constructed through an information extraction pipeline. (3) **ATOMIC** (Sap et al., 2019) has social commonsense knowledge about causes and effects of everyday events, and mental states (e.g., `xIntent`) of their participants. It is created by crowdsourcing.

As indicated by Jastrzebski et al. (2018), a large proportion of the subjects in the test set of ConceptNet-100K overlap with its training set, while TupleKB does not provide an official split. Thus, we (re-)split these two datasets to ensure that the subjects of testing triplets do not appear in the training set. This criterion is also consistent with how the ATOMIC dataset is constructed.

2.3 Experimental Settings

Multi-task Learning To answer Q1, we adapt an LM with balanced training data from ConceptNet, TupleKB, and ATOMIC. We sample 8 triplets from each dataset to form one training batch.

Transfer Learning To provide insight into Q2, we adopt transfer learning under a leave-one-out strategy. In this setting, we adapt an LM on two of the three CKGs, and then we further adapt it on the third target CKG. Moreover, we study the data efficiency of this transfer learning by down-sampling each training set to $x = \{1, 20, 50\}\%$, in order to see whether the LM can adapt to the target CKG with less training effort. Fine-tuning on data as small as 1% training set may suffer from instability, and results may change dramatically given a new split of training data (Gao et al., 2020). To control the randomness, we re-sample the 1% training data 5 times with a fixed set of random seeds and report the average performance instead.

Controlled Low-resource Learning To answer Q3, we design a controlled experiment, where we first split the training set into two disjoint subsets depending on whether the triplets in the original training set contain objects that exist in the test set or not. We denote the subset where the objects of the triplets appear in testing data as Ω. We sample $x = \{0, 25, 50, 100\}\%$ of the training triplets in Ω for adapting the LM. During the evaluation, we also separate the test set into two disjoint subsets, according to whether the objects are seen in the original full training set. The results on these two split test sets are reported separately for each adapted LM.

Evaluation Protocol For each *(subject, relation)* pair in the test set, we treat all their objects as ground truth references for evaluating the model inference. We report scores for commonly used automatic evaluation metrics for text generation: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005), which are shown to be consistent with human judgements (Hwang et al., 2020). During experiments, we observe a high correlation among these differ-

Adaptation method	**Input**	**Learnable params**
Zero-shot (ZS)	(s, r)	N/A
ZS+demo	$(s^{'}, r, o^{'}, s, r)$	N/A
Fine-tuning (FT)	(s, r)	Transformer (LM)
FT+demo	$(s^{'}, r, o^{'}, s, r)$	Transformer (LM)
Adapter tuning (AT)	(s, r)	Adapter

Table 1: Methods for using LMs to conduct commonsense inference. *"+demo"* means prepending a demonstration triplet $(s^{'}, r, o^{'})$ before the input tuple.

ent metrics and choose to report METEOR in the main text and other metrics in the appendix.

2.4 Connections to Prior Studies

Earlier works (Li et al., 2016; Jastrzebski et al., 2018; Davison et al., 2019) poses the CKG completion task as triplet classification, where the goal is to score the plausibility of a complete triplet. COMET (Bosselut et al., 2019) is the first to cast this task as commonsense inference with LMs. Follow-up contributions utilize COMET as a commonsense provider in various downstream tasks (Bosselut and Choi, 2021; Ammanabrolu et al., 2021; Chakrabarty et al., 2020), thus providing evidence for LM's generalization to previously unseen scenarios. Further efforts include Hwang et al. (2020), which show that the quality of the training triplets is a key factor of adapting LMs, and (Da et al., 2021), which investigates how to learn COMET in a few-shot learning setting. Meanwhile, the study by Wang et al. (2020a) indicates the limited generalization of COMET. Ma et al. (2021) also adapt LMs simultaneously on multiple CKGs, albeit their goal is to improve downstream performance rather than CKG inference. In this paper, we aim to provide a more comprehensive study of a LM's generalizability for CKG inference.

3 Method

While a set of pretrained LMs exists, we adopt a widely used generative model, GPT2 (Radford et al., 2019), as our baseline LM. The investigation of other generative LMs is orthogonal to our analysis. We experiment with its largest version, GPT2-XL, which contains 48 transformer layers (Vaswani et al., 2017), ensuring sufficient capacity for storing knowledge acquired during its pretraining. We introduce our experimental method as follows.

Commonsense Inference with LMs Given a training triplet *(s,r,o)*, we represent s and o as sequences of tokens, $\mathbf{x}_s$ and $\mathbf{x}_o$, which is trivial given that they are already expressed as phrases. As for the relation r, we convert it by using a template taken from the literature (Davison et al., 2019) into a natural-language phrase $\mathbf{x}_r$, e.g., `IsA` is converted to "is a". This has been shown to facilitate efficient adaptation of LMs (Da et al., 2021). Note that we do not explicitly provide the LMs with the information about the source CKG of the triplet as input (e.g., prepending a related special token to the triplet).

Adapting LMs with Commonense Knowledge The training objectives for adapting LMs is to maximize the probability of generating the object phrase $\mathbf{x}_o$ given the tuple $(\mathbf{x}_s, \mathbf{x}_r)$. During inference, we adopt greedy decoding to obtain the predicted object from the adapted LM.

There have been various techniques developed for adapting pretrained LMs to downstream tasks (Howard and Ruder, 2018; Chen et al., 2020). Moreover, previously only the vanilla **Fine-tuning**, i.e., updating the whole LM architecture during training, has been employed to adapt LMs for commonsense inference (Bosselut et al., 2019; Hwang et al., 2020; Da et al., 2021). To obtain comprehensive results that are not specific to one particular way of fine-tuning, here we investigate two more alternatives, each of which has their own advantage when considered in different contexts.

Fine-tuning with Demonstration (FT+demo) Combining the ideas of fine-tuning and in-context learning (Brown et al., 2020), this technique (Gao et al., 2020) adds a demonstration to each input as additional context and fine-tunes the whole LM as usual. Incorporating demonstrations is shown to boost performance when the amount of training data is extremely limited. In our case, a demonstration is a top-1 training triplet $(s^{'}, r, o^{'})$, ranked according to the cosine similarity between the embedding of the input tuple (s, r) and the embeddings of the training tuples with the same relation type r. The tuple embeddings are given by a pretrained Sentence-BERT (Reimers and Gurevych, 2019). For instance, a demonstration (`go to restaurant`, `UsedFor`, `eat out`) would be added before the input (`go to pub`, `UsedFor`). With the demonstrated triplets, the LM could learn to understand the schema of the CKG instead of simply learning the knowledge from the training data.

Adapter Tuning (AT) Unlike fine-tuning, adapter tuning (Houlsby et al., 2019) fixes the entire LM and adds one trainable adapter right before the skip connection in each transformer layer of the LM,

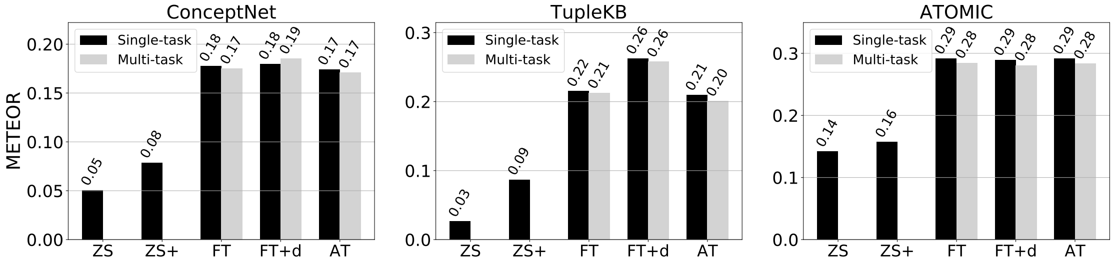

Figure 2: **Results (METEOR) for *knowledge capacity* of LMs. ”FT+d” refers to FT+demo.** We find no notable performance drop for any method trained in the multi-task setting.

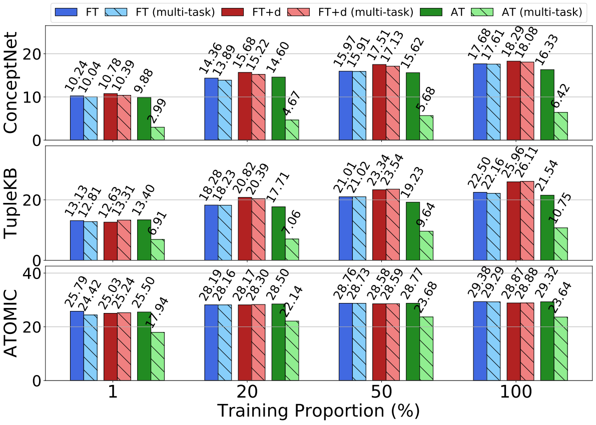

Figure 3: **Results (METEOR) for LM *transferability*.** ”FT+d” refers to FT+demo. Across datasets, we do not observe that adapting to the source CKGs would enable the LMs to adapt to the target CKG better or more easily.

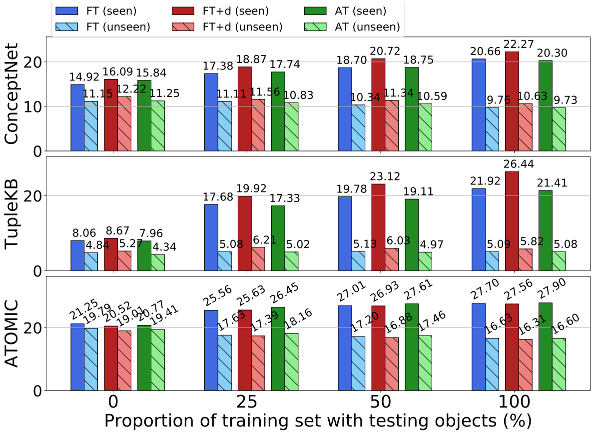

Figure 4: **Results (METEOR) for LM *induction*.** ”FT+d” refers to FT+demo. All the methods perform better on predicting facts that contain seen objects, while the performance degrades when less objects are seen during training.

which is more parameter-efficient. Each adapter is a two-layer bottleneck network with a skip-connection internally. Following Houlsby et al. (2019), the parameters of the bottleneck network are initialized close to zero so that the adapter approximates an identity function from the beginning.

We compare to two additional baselines, both using GPT2-XL in a zero-shot setting: **Zero-shot (ZS)** is fed with the same input as Fine-tuning, while zero-shot with demonstrations (**ZS+demo**) combines the input plus demonstration, as in the FT+demo method. By investigating all these methods, we aim to understand the influence of different adaptation techniques on the models' performance. Table 1 summarizes the set of methods which we consider in this paper.

4 Results and Discussion

Knowledge Capacity (Q1) The results that quantify the knowledge capacity of LMs for commonsense inference over multiple CKGs with METEOR scores are shown in Figure 2. The complete results including other metrics can be found in the appendix. All adaptation methods perform considerably better than the zero-shot baselines, indicating the benefit of adaptation. There is no clear distinction between the adaptation methods, though FT+demo performs slightly better than the others across CKGs. Most importantly, we find no notable performance drop for any method in the multi-task training setup despite the challenge that there is limited overlap between these CKGs. Only 10.0% of the facts from ATOMIC can be found in ConceptNet (Hwang et al., 2020) while 8.4% of the facts from ConceptNet can be found in TupleKB (Dalvi Mishra et al., 2017) [2]. This indicates the prominent capacity of LMs to simultaneously adapt to different CKGs. Nevertheless, the results reveal that learning different CKGs jointly do not interfere with each other positively (via knowledge sharing) or negatively (due to overfitting).

Transferability (Q2) Figure 3 shows the obtained results regarding the transferability of LMs. Across different CKGs and for any training data size, we observe no indications that adapting to the source CKGs enhances the performance on the target CKG. On the contrary, adapting from source CKGs

---

[2] We also try to breakdown the results by relation types and do not observe correlation between the relation-wise performance and the extent of overlap.

even hurts the performance of the Adapter-tuning method, revealing that this method overfits to the source CKGs. Overall, we conclude that LMs cannot reuse the knowledge learned from the source CKGs to improve the performance on the target CKG or achieve the same performance with less training data. Thus, we call for future study on developing more effective adaptation methods.

Induction (Q3) The results in Figure 4 show that without down-sampling ($x = 100\%$), all methods perform much better on predicting facts that contain seen objects, and their performance degrades more when less object entities are seen to training. Meanwhile, the performance on facts with unseen objects stays roughly unaffected. This indicates a key limitation of the LMs: they adapt notably better on seen objects. Since the training set and test set do not share subjects, we conclude that the generalizability of the LM is largely dependent on finding the relationship between unseen subjects and observed objects. We thus posit that a novel strategy for adapting LMs while retaining the knowledge acquired during pre-training is necessary for better generalizability. Promising directions here are prefix tuning (Li and Liang, 2021) or including an additional objective during adaptation which would encourage the generation of novel objects.

5 Conclusion

This work conducted a focused study of three aspects of the generalizability of LMs for commonsense inference: knowledge capacity, transferability, and induction. We experiment with five methods of using a generative LM and three representative CKGs. Despite their capability to accommodate multiple CKGs, we have observed that LMs have limited ability to transfer knowledge across CKGs. Moreover, their adaptation relies heavily on whether the objects to predict are seen during training. These findings help our understanding of LMs' adaptation behavior on commonsense inference, and highlight the need for future work to improve their transferability and induction.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This material is based upon work sponsored by the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research.

References

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Antoine Bosselut and Yejin Choi. 2021. Dynamic knowledge graph construction for zero-shot commonsense question answering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Tuhin Chakrabarty, Smaranda Muresan, and Nanyun Peng. 2020. Generating similes effortlessly like a pro: A style transfer approach for simile generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6455–6469, Online. Association for Computational Linguistics.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021. Understanding few-shot commonsense knowledge models. *arXiv preprint arXiv:2101.00297*.

Bhavana Dalvi Mishra, Niket Tandon, and Peter Clark. 2017. Domain-targeted, high precision knowledge extraction. *Transactions of the Association for Computational Linguistics*, 5:233–246.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2020. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. *arXiv preprint arXiv:2010.05953*.

Stanislaw Jastrzebski, Dzmitry Bahdanau, Seyedarian Hosseini, Michael Noukhovitch, Yoshua Bengio, and Jackie Cheung. 2018. Commonsense mining as knowledge base completion? a study on the impact of novelty. In *Proceedings of the Workshop on Generalization in the Age of Deep Learning*, pages 8–16, New Orleans, Louisiana. Association for Computational Linguistics.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and Alessandro Oltramari. 2021. Knowledge-driven Data Construction for Zero-shot Evaluation in Commonsense Question Answering. In *35th AAAI Conference on Artificial Intelligence*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Heiko Paulheim. 2018. How much is a triple? estimating the cost of knowledge graph creation. In *Proceedings of the 17th International Semantic Web Conference*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3027–3035. AAAI Press.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Cunxiang Wang, Jinhang Wu, Luxin Liu, and Yue Zhang. 2020a. Commonsense knowledge graph reasoning by selection or generation? why? *arXiv preprint arXiv:2008.05925*.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020b. Joint constrained learning for event-event relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 696–706, Online. Association for Computational Linguistics.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020c. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4970–4977. AAAI Press.

		BLEU-2		ROUGE-L		METEOR	
		single-task	multi-task	single-task	multi-task	single-task	multi-task
ConceptNet	Zero-shot	0.0069	NA	0.1009	NA	0.0506	NA
	ZS+demo	0.0284	NA	0.1281	NA	0.0787	NA
	Adapter-tuning	0.1289	0.1279	0.2598	0.2560	0.1739	0.1706
	Fine-tuning	0.1325	0.1286	0.2629	0.2575	0.1775	0.1749
	FT+demo	**0.1333**	**0.1398**	**0.2678**	**0.2738**	**0.1795**	**0.1851**
TupleKB	Zero-shot	0.0017	NA	0.0999	NA	0.0263	NA
	ZS+demo	0.0099	NA	0.2748	NA	0.0869	NA
	Adapter-tuning	0.1383	0.1323	0.3785	0.3627	0.2094	0.2010
	Fine-tuning	0.1371	0.1388	0.3985	0.3812	0.2151	0.2122
	FT+demo	**0.1699**	**0.1698**	**0.4902**	**0.4714**	**0.2622**	**0.2580**
ATOMIC	Zero-shot	0.0436	NA	0.2523	NA	0.1419	NA
	ZS+demo	0.0808	NA	0.2233	NA	0.1572	NA
	Adapter-tuning	**0.2161**	0.2035	**0.4008**	0.3890	**0.2913**	0.2832
	Fine-tuning	0.2125	0.2057	0.3982	**0.3908**	0.2913	**0.2843**
	FT+demo	0.2111	**0.2070**	0.3915	0.3868	0.2887	0.2800

Table 2: Results of all the evaluation metrics for the knowledge capacity experiments.

A Appendix

A.1 Dataset Statistics

[h]

	Train	Dev	Test
ConceptNet100k	79,770	10,203	10,027
TupleKB	98,674	12,357	12,427
ATOMIC	578,002	64,902	71,127

Table 3: CKG Dataset Statistics.

A.2 Implementation Details

The GPT2-XL language model we adopted in this work has 1558M parameters in total. We train all the models on a V100 GPU. As for hyper-parameters, we adopt the commonly-used learning rate (1e-5) and batch size (16) for adapting GPT2, except that in the multi-task learning setting, the batch size is 24 (8 samples from each CKG).

A.3 Additional Results

Probing Multi-modal Machine Translation with Pre-trained Language Model

Yawei Kong, Kai Fan
Alibaba DAMO Academy
{yawei.kyw,k.fan}@alibaba-inc.com

Abstract

Multi-modal machine translation (MMT) aimed at using images to help disambiguate the target during translation and improving robustness, but some recent works showed that the contribution of visual features is either negligible or incremental. In this paper, we show that incorporating pre-trained (vision) language model (VLP) on the source side can improve the multi-modal translation quality significantly. Motivated by BERT, VLP aims to learn better cross-modal representations that improve target sequence generation. We simply adapt BERT to a cross-modal domain for the vision language pre-training, and the downstream multi-modal machine translation can substantially benefit from the pre-training. We also introduce an attention based modality loss to promote the image-text alignment in the latent semantic space. Ablation study verifies that it is effective in further improving the translation quality. Our experiments on the widely used Multi-30K dataset show increased BLEU score up to 6.2 points compared with the text-only model, achieving the state-of-the-art results with a large margin in the semi-unconstrained scenario and indicating a possible direction to rejuvenate the multi-modal machine translation.

1 Introduction

Joint models of language and vision have achieved remarkable results, such as in image caption (Karpathy and Fei-Fei, 2015) and visual question answering (Antol et al., 2015). Multi-modal machine translation (MMT) was first introduced as a shared competition task at the 2016 Conference on Machine Translation (WMT16) (Specia et al., 2016) as an interdisciplinary study to incorporate a visual element into the multilingual translation task. This task continued for three years until WMT18, and the findings presented by the organizers suggest that the text-only systems remain competitive, and that the contribution of visual modality is not entirely convincing (Specia et al., 2016; Elliott et al., 2017; Barrault et al., 2018). Moreover, the experiments in (Elliott, 2018) find that a publicly available MMT system produces great translations with random, incongruent images, further undermining the importance of visual features. The empirical results have so far raised doubts about whether the visual features can really help MMT, and there is evidence pointing to a negative answer.

Data used	img	src	tgt	examples
Multi-30K	✓	✓	✓	most works
+external data	✓	✓		(Grönroos et al., 2018)
		✓	✓	(Helcl et al., 2018)
	✓	✓		(Yin et al., 2020), ours

Table 1: Different unconstrained scenarios in MMT.

We hypothesize that one reason is the data scale of the benchmarking Multi-30K (Elliott et al., 2016) – it is likely insufficient for a deep model to learn better cross-modality or cross-lingual representations. However, the pre-training techniques such as BERT (Devlin et al., 2019) or cross-lingual language model (XLM) (Conneau and Lample, 2019) can capture rich representations of the inputs from languages and be applied to various downstream tasks by providing context-aware embeddings, leading to remarkable improvements even on small datasets. Furthermore, the pre-trained vision and language model LXMERT (Tan and Bansal, 2019) pioneers the cross-modality pre-training and sets an influential record in vision and language reasoning tasks. These advances lead us to believe that a better cross-modality representation can help multi-modal machine translation as well.

In this work, we discuss the unconstrained scenario of MMT, but unlike previous setting in most WMT 2018 submissions (Grönroos et al., 2018; Helcl et al., 2018), we did not include any external data of the parallel source and target textual corpus. Since we want to incorporate a pre-trained (vision)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3689–3699
August 1–6, 2021. ©2021 Association for Computational Linguistics

language model as an encoder backbone into the transformer architecture (Vaswani et al., 2017) for neural machine translation, our used external data only contains the images and the source texts.

Particularly, our model is initialized with the widely used BERT, and pre-trained on large scale image-text dataset (about six million pairs), expecting to learn a better cross-modality representation between the image and the source language. Next, we stack a regular transformer decoder on top of the pre-trained (vision) language model and proceed to the task of MMT. Meanwhile, we design another modality loss in addition to the traditional sequential cross entropy loss. The modality loss is to minimize the difference between source-target cross attention and image-target cross attention. Intuitively, minimizing this loss function can promote the modality alignment among the three possible pairwise configurations in the latent semantic space. In other words, differences among (source, target), (source, image), and (target, image) alignments can be reduced. Our experimental section also presents a detailed analysis of how each factor separately contributes to the overall gains.

In summary, this paper makes the following contributions. **(1)** We propose to integrate a pre-trained vision language model into multi-modal machine translation, aiming at learning and utilizing better cross-modality representations. **(2)** We address the importance of the modality loss which can further boost the model performance. **(3)** We conduct extensive experiments on the benchmark Multi-30K dataset, and our results outperform strong baselines by a large margin.

2 Related Works

Constrained Scenario Most works like (Calixto et al., 2017; Zhou et al., 2018; Ive et al., 2019; Yao and Wan, 2020) in MMT prefer to use Multi-30K dataset alone. For example, a standard paradigm of MMT explored by many previous works is to simultaneously learn the vision language interaction and the target language generation (Calixto et al., 2017; Zhou et al., 2018; Ive et al., 2019; Yang et al., 2020). However, training on such a limited dataset, the benefits provided by visual features of these methods are quantitatively marginal w.r.t. automatic evaluation metrics BLEU and METEOR.

Unconstrained Scenario In the submissions of WMT 2018 (Grönroos et al., 2018; Helcl et al., 2018) as shown in Table 1, either images / source texts or the source / target texts parallel dataset (or back-translation) are added to improve the model performance. However, as they discovered, training with the large scale parallel textual corpus will shift the machine translation model towards the pure textual domain, further weakening the effect of visual features. The additional target data will also make the fair comparison difficult. A special unconstrained scenario by (Su et al., 2019b) leverages large monolingual language data to pre-train an unsupervised translation model. It considers the cross representation of the source-target in an unsupervised manner, but the image domain is still isolated without proper training.

We will discuss another unconstrained scenario that only allows to use additional images and source texts. Zhu et al. (2019) investigates the representation from pre-trained BERT by feeding it into all layers of a text-only translation model. This work, to a large extent, encourages us to explore how the (vision) language pre-trained model can benefit the MMT. However, we found that a direct architecture of feeding cross-modality representations (from LXMERT) to multi-modal translation model does not work well.

To our best knowledge, Yin et al. (2020) currently achieves the state-of-the-art on Multi-30K. It employed a common encoder-decoder framework by hard-encoding a multi-modal graph to guide the learning of the image-text cross attention, where the graph structure is annotated by a pre-trained visual grounding model (Yang et al., 2019). The external data is not explicitly used in this work, but the pre-trained visual grounding model uses BERT as part of its backbone. Instead of relying on a pre-defined graph to prevent the attention between the word and visual feature without connection, we obtain a soft cross attention from large-scale vision-language data pre-training. It is also worth mentioning that we make the BERT based visual grounding and multi-modal machine translation into an end-to-end trainable architecture.

3 Our Method

3.1 Initial Trial

The overall architecture of our proposed approach is based on the commonly used transformer (Vaswani et al., 2017), which is the basic unit of most pre-trained (vision) language model. Our initial experiment is to adopt pre-trained (vision) language model as the encoder. The baseline is to train

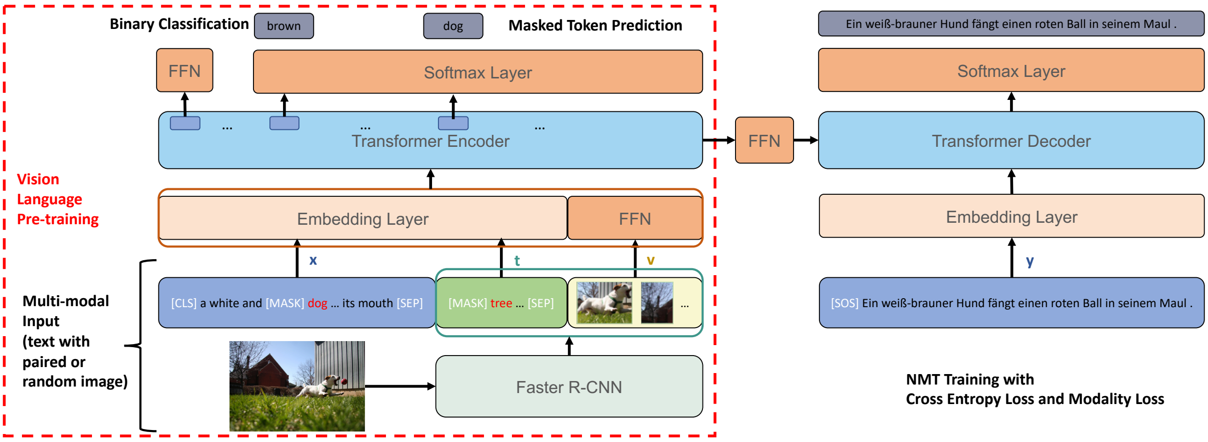

Figure 1: The overall architecture of our proposed multi-modal NMT with pre-trained vision language model. Note that the [MASK] tokens and random images are merely applied during vision language pre-training.

Encoder	visual feature	Test2016 EnDe		Test2016 EnFr	
		BLEU	Meteor	BLEU	Meteor
Transformer	-	38.3	56.6	59.6	74.6
BERT	-	39.1	57.1	61.0	75.3
LXMERT	✓	37.4	55.2	57.7	68.6

Table 2: BERT/LXMERT are frozen.

a transformer NMT from scratch. The first competitive system is simply BERT, and the second one is the pre-trained vision language model LXMERT. LXMERT claimed that the initialization with pre-trained BERT will harm the performance of their downstream tasks. Table 2 shows the preliminary results indicating that the pre-trained LXMERT as the encoder performs surprisingly worse than text-only BERT. Does the table suggest that the visual features are equally marginalized in MMT equipped with pre-trained language model? However, since BERT encoder can bring more improvements, we can abandon LXMERT's conclusion and return to the paradigm with BERT initialization.

3.2 Vision Language Pre-training (VLP)

Ive et al. (2019) finds that integrating both object-based embedding features and image features into the NMT model results better performance in human evaluation on comprehensibility. We therefore favor the object-semantics alignment whose interaction is composed of text embedding, object tag embedding and object image features.

We visualize the training rationale of the VLP in the red dashed box of Figure 1. Suppose that an image and its description $\mathbf{x}$ are presented as the input, where $\mathbf{x}$ represents a sequence of n tokens $(x_1, ..., x_n)$, i.e., the sentence of the source language in our following NMT system. We first process the image with the efficient object detection model Faster-RCNN (Ren et al., 2015) to detect the object regions, box positions, object tags and attribute tags. Particularly, two sets of features are extracted. One is the image visual features of all detected objects, denoted as $\mathbf{v}$. The other is the classification tags of the corresponding objects, denoted as $\mathbf{t}$, as textual features.

Since the backbone of our transformer encoder is pre-trained BERT, the input text $\mathbf{x}$ and object tags $\mathbf{t}$ are both language tokens that can be easily concatenated. However, there is a dimensionality mismatch between the BERT embedding layer and the visual features. For dimension reduction, a fully-connected layer is necessary with input $\mathbf{v}$, and its task is to learn cross modality transferring. The final input fed into the multiple transformer layers of BERT can be written as follows.

$$\mathrm{Cat}\left[\mathrm{Emb}\left(\mathrm{Cat}[\mathbf{x},\mathbf{t}]\right), \mathrm{FFN}(\mathbf{v})\right] \qquad (1)$$

We now face two similar tasks as BERT.

Task 1: Masked LM Same as the standard BERT, our training objective employs the masking token prediction, where 15% of the input text tokens are randomly selected and replaced with the special token [MASK]. Then, only the masked token will be predicted.

Task 2: Paired Image Prediction Analogous to the standard BERT, we pre-train the binarized paired image prediction task that mimics predicting the next sentence, where the training data can be trivially generated for each batch. Specifically, for a given text input, we choose its paired image or

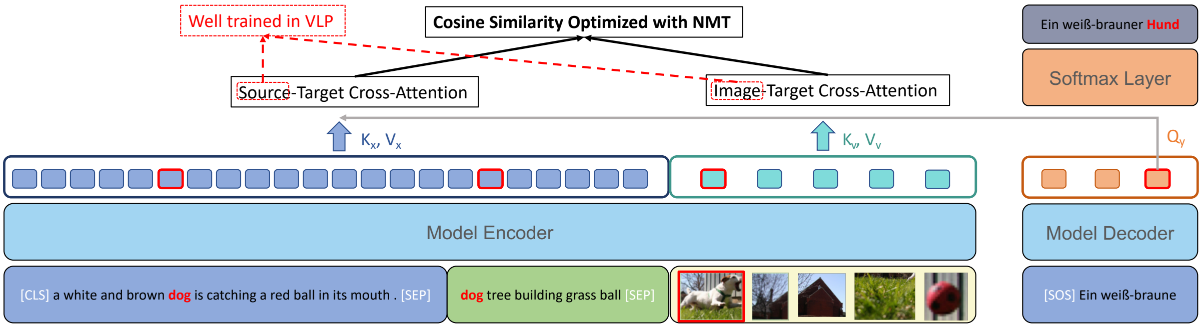

Figure 2: The visualization of modality loss for an input sentence-image pair. It exemplifies the computational flow of the modality loss w.r.t. the last layer of the decoder when decoding “Hund” in German.

a random image each with probability 50%. The output vector of the first special token [CLS] is used as the aggregate multi-modal representation for this classification task.

3.3 Multi-modal NMT

Once the vision language model has been fully trained on a large paired image-text dataset, it is reasonable to assume that the obtained cross-modality representations between the source text and the image are more powerful than those training on the limited Multi-30K. The (key, value) pairs of both the textual and visual features participate in the dot-product attention of the transformer decoder. But there is another dimensionality mismatch between the BERT output and the decoder hidden size. To close this gap, we append an additional fully connected layer after the last layer of BERT. In this section, we also introduce a novel modality loss that is potential to benefit the multi-modal representation learning while but incurs only a few extra model parameters.

Modality Loss To train a multi-modal machine translation task, i.e., generating the tokens in the target language $\mathbf{y} = (y_1, ..., y_m)$, a common objective is the sequential cross entropy loss $\mathcal{L}_{\text{XENT}} = -\sum_{j=1}^{m} \log p(y_j|\mathbf{y}_{<j}\mathbf{x}, \mathbf{v})$, which is the sum of the negative log-likelihoods of the auto-regressive text generation task. Our proposed auxiliary modality loss can be intuitively depicted as Figure 2.

Concretely, when generating the j-th token in the target, the output textual and visual (key, value) pairs from the encoder are separately used to compute the cross-lingual and cross-modality attention with the query vector of the l-th layer in the decoder. The derived vectors can be written as follows.

$$\mathbf{h}_{x,j}^{(l)} = \text{Softmax}\left(K_x \mathbf{q}_j^{(l)} / \sqrt{d}\right) V_x \tag{2}$$

where d is the hidden size of the model decoder, and similar attention holds for visual features $\mathbf{h}_{v,j}^{(l)} = \text{Softmax}\left(K_v \mathbf{q}_j^{(l)} / \sqrt{d}\right) V_v$. Thus, the modality loss can be represented as

$$\mathcal{L}_{\text{M}}^{(l)} = \sum_{j=1}^{m} (1 - \cos(\mathbf{h}_{x,j}^{(l)}, \mathbf{h}_{v,j}^{(l)})) \tag{3}$$

where the cosine similarity is defined as $\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|, \|\mathbf{b}\|}$. Consequently, the overall training objective is a weighted combination of two loss functions.

$$\mathcal{L} = \mathcal{L}_{\text{XENT}} + \sum_{l=1}^{L} \lambda^{(l)} \mathcal{L}_{\text{M}}^{(l)} \tag{4}$$

where L is total number of transformer layers in decoder. Empirically, we found that only using the modality loss of the last layer is sufficient to improve the model performance. Intuitively, the query vector will be directly fed into the softmax layer for decoding the target tokens, making the last layer more informative than other remote layers.

A common method of choosing the weighting parameter λ is to run cross validation on the held-out development data. For the task at hand, this is a time-consuming process. We instead discard the layer-wise $\lambda^{(l)}$ in Eq. (4) and introduce a self-tuning module with respect to the generation process of every single target token. Mathematically, the refined modality loss can be formulated as,

$$\widetilde{\mathcal{L}}_{\text{M}}^{(l)} = \sum_{j=1}^{m} \lambda_j^{(l)} (1 - \cos(\mathbf{h}_{x,j}^{(l)}, \mathbf{h}_{v,j}^{(l)})). \tag{5}$$

where the token level λ_j is learnable and derived from a feedforward neural network.

$$\lambda_j^{(l)} = \text{Sigmoid}(\mathbf{w}_y^\top \text{Emb}(y_j) + \mathbf{w}_x^\top \mathbf{h}_{x,j}^{(l)} + \mathbf{w}_v^\top \mathbf{h}_{v,j}^{(l)})$$

where $\mathbf{w}_y, \mathbf{w}_x, \mathbf{w}_v$ are three d-dimensional vectors shared cross different decoder layers and required

Algorithm 1 Training Pipeline

```
Require: Image, source text paired data $\mathcal{D}_{\text{VLP}}$; Image, source/target text triple data $\mathcal{D}_{\text{MMT}}$.
  Initialize the transformer encoder of NMT with pre-trained BERT.
  Pre-train the transformer encoder on $\mathcal{D}_{\text{VLP}}$ with masked language model task and pair image prediction task.
  Extract the image, source text paired from $\mathcal{D}_{\text{MMT}}$.
  Continue the vision language pre-training on above extracted data.
  Freeze the transformer encoder, and optimize other parameters on $\mathcal{D}_{\text{MMT}}$ with cross entropy loss and modality loss until convergence.
  Optimize all model parameters on $\mathcal{D}_{\text{MMT}}$ with cross entropy loss and modality loss until convergence.
```

to jointly optimize with the model parameters, but useless during inference. We expect the model to dynamically adjust the weight parameters of the tokens with different importance. For example, there is a good chance that the content words also appear as detected objects by the Faster R-CNN. If the term $\mathbf{w}_v^\top \mathbf{h}_{v,j}^{(l)}$ can positively increase its scale for such words, the corresponding λ_js become larger and therefore reinforce the maximization of the cosine similarity. In contrast, although it happens that a mapping exists between the source and target functional words, the image-target cross attention may become weak, making it less necessary to promote the similarity. The term $\mathbf{w}_x^\top \mathbf{h}_{x,j}^{(l)}$ is intended to model the importance of the source contribution. Our results, however, show that in the current experiment setup its effect is not quite as significant.

3.4 Two-Stage Training

When BERT is applied to the downstream tasks, the task-specific module parameters are usually plugged into BERT and all the trainable parameters are simultaneously fine-tuned (Devlin et al., 2019). However, we found this is not the optimal strategy of training our downstream task – multi-modal machine translation. The large number of untrained parameters in the transformer decoder almost account for half of the model size. We conjecture that the encoder parameters have already reached a flat plateau after the pre-training, and it is difficult to set the consistent optimization hyper-parameters (such as learning rate, decay rate or warm-up steps) for both the encoder and decoder.

Therefore, we adopt a two-stage training schedule. In the first stage, the encoder parameters are frozen and only the decoder parameters are optimized w.r.t. the cross entropy and modality loss. In the second stage, all model parameters become trainable and are updated concurrently. This step simulates the regular BERT fine-tuning procedure, and its convergence is expected to lead to a better performance. To this end, we have elaborated the key ideas of our proposed method and summarize the training pipeline of the entire model training process in Algorithm 1.

4 Experiments

In this section, we describe the datasets, the detailed settings as well as the compared baselines.

4.1 Datasets and Settings

Multi-30K We conduct experiments on the Multi-30K dataset (Elliott et al., 2016), where each image is paired with one English(En) description and human translations of German(De) and French(Fr). It has 29,000 instances for training and 1,014 instances for development. Besides, we evaluate our model on various testing sets, including the Multi-30K 2016 test set, the WMT17 test set and the ambiguous MSCOCO test set, which contain 1,000, 1,000 and 461 instances, respectively.

External Data We use about 6 million image and English text paired data for our vision language model pre-training, including MSCOCO (Lin et al., 2014), Im2text (Ordonez et al., 2011), visual7w (Zhu et al., 2016), VQA 2.0 (Goyal et al., 2017), Conceptual captions (Sharma et al., 2018), GQA (Hudson and Manning, 2019). We first process the image with a popular off-the-shelf Faster-RCNN toolkit[1] (Ren et al., 2015; Anderson et al., 2018; Wu et al., 2019). The Faster R-CNN (Ren et al., 2015) network is pre-trained on the MSCOCO dataset and fine-tuned on the Visual Genome (Krishna et al., 2017) dataset to detect salient visual objects, where the number of visual objects ranges from 10 to 100 with the highest prediction probability and 2048 is the dimension of the flattened last pooling layer in the ResNet (He et al., 2016) backbone. Then, we obtain the position-sensitive

---

[1] `https://github.com/airsplay/py-bottom-up-attention`

Model	En⇒De						Notes on external resources
	Test2016		Test2017		MSCOCO		
	BLEU	Meteor	BLEU	Meteor	BLEU	Meteor	
Our text-only	38.3	56.6	30.3	51.0	28.6	47.7	Our own implemented transformer
Doubly-Att	36.5	55.0	-	-	-	-	Constrained methods ResNet features only
Fusion-conv	37.0	57.0	29.8	51.2	25.1	46.0	
Trg-mul	37.8	57.7*	30.7	52.2*	26.4	47.4	
VAG	31.6	52.2	-	-	-	-	
VMMT	37.7	56.0	30.1	49.9	25.5	44.8	
DNetwork	38.0	55.6	-	-	-	-	
Multimodal-Att	38.7	55.7	-	-	-	-	
Semi-unconstrained methods							
VMMT	38.4	58.3	-	-	-	-	few Back-translation data
Multimodal-Att	39.5	56.9	-	-	-	-	few Back-translation data
Graph-Fusion	39.8*	57.6	32.2*	51.9	28.7*	47.6*	BERT(en), visual grounding tool
Our Model	**42.7**	**60.7**	**35.5**	**54.9**	**32.8**	**52.2**	BERT(en), images-en
WMT 2018 unconstrained methods							
MeMAD	45.1	-	40.8	-	36.9	-	images-en, OpenSub en-de/fr
CUNI	42.7	59.1	-	-	-	-	images-en, Bookshop en-de/fr, Back-translation

Table 3: Experimental results on the En⇒De MMT. Our results are highlighted in bold. $*$ indicates previous SOTA. B will be short for BLEU and M will be short for Metoer in other tables.

Model	En⇒Fr			
	Test2016		Test2017	
	B	M	B	M
Our Text-only	59.6	74.6	52.7	69.1
Doubly-Att	59.9	74.1	52.4	68.1
Fusion-conv	53.5	70.4	51.6	68.6
Trg-mul	54.7	71.3	52.7	69.5*
VAG	53.8	70.3	-	-
DNetwork	59.8	74.4	-	-
Semi-unconstrained methods				
Graph-Fusion	60.9*	74.9*	53.9*	69.3
Our Model	**65.8**	**79.1**	**58.2**	**73.5**
WMT 2018 unconstrained methods				
MeMAD	68.3	-	62.5	-
CUNI	62.8	77.0	-	-

Table 4: Experimental results on the En⇒Fr MMT.

visual features by concatenating the region features and the corresponding positions.

For the English text, we follow the same pre-processing as the open-source BERT toolkit[2]. The BERT base model with hidden size 768 is utilized as initialization. Note that unlike (Grönroos et al., 2018; Helcl et al., 2018), we never include any external data related to the target languages for both vision language pre-training and machine translation training. For notation simplicity and differentiating their setting, we define our scenario as **semi-constrained**.

4.2 Baselines

We mainly compare with the following representative and competitive frameworks. The constrained methods include **Doubly-Att** (Calixto et al., 2017), **Fusion-conv / Trg-mul** (Caglayan et al., 2017), **VAG** (Zhou et al., 2018), **VMMT** (Calixto et al., 2019) and **Multimodal-Att** (Yao and Wan, 2020). **MeMAD** and **CUNI** (Grönroos et al., 2018; Helcl et al., 2018) mainly discussed the unconstrained scenario of MMT. In addition, VMMT and Multimodal-Att attempted to adding in-domain back-translation data. We prefer to include them into semi-unconstrained methods as well. **Graph-Fusion** (Yin et al., 2020) uses BERT based visual ground model to hard-code a unified multi-modal graph and performs semantic interactions by graph fusion layers, achieving the current state-of-the-art performance.

---

[2] `https://github.com/huggingface/transformers`

4.3 Main Results

In Table 3 and 4, we report the main experimental results of our proposed method with previous research works. All reported numbers of our approach are evaluated on the best performed model for the validation set. Note that when optimizing the parameters, we only use the modality loss calculated from the last layer with learnable token level $\lambda_j^{(6)}$. In other words, the reported numbers are obtained by minimizing $\mathcal{L}_{\text{XENT}} + \widetilde{\mathcal{L}}_{\text{M}}^{(6)}$. In the ablation study, we demonstrate this simplification not only reduces the computational complexity, but also achieves better result than our initial proposal.

Both tables show that our multi-modal translation outperforms the existing models and baselines, especially the recent state-of-the-art algorithm Graph Fusion, which also leveraged the pre-trained BERT based visual grounding model from large scale paired image-text data. However, it only hard-coded the inferred multi-modal graph by visual grounding to construct the mask matrix of cross modality attention in the transformer encoder. One advantage of our work is that we directly build our NMT model on top of the pre-trained vision language BERT, making the most of pre-trained cross modality attention. Another advantage is that our end-to-end trainable model can spontaneously avoid the error accumulation.

Since our multi-modal translation model is implemented based on the text-only transformer, we also report the text-only results with our own implemented transformer for a fair comparison. Our text-only transformer is a surprisingly strong baseline and very competitive with most cited works. For English to German translation task, our text-only baseline almost beats all previous works on the ambiguous MSCOCO test set, and is only inferior to two systems on Multi-30K test sets with less than 2 BLEU score difference. For English to French translation task, only the Graph Fusion algorithm significantly outperforms our text-only transformer. In contrast, on the three test sets of English to German, our final multi-modal translation model can on average achieve approximately +4.6 BLEU and +4.2 METEOR over the text-only baseline. On the two test sets of English to French, the averaged gains of our model are about +5.85 and +4.45 on BLEU and METEOR.

Model	Test2016		Test2017		MSCOCO	
	B	M	B	M	B	M
Text-only En⇒De Model						
Transformer	38.3	56.6	30.3	51.0	28.6	47.7
BERT-NMT	39.4	56.6	29.7	48.6	27.9	46.2
BERT-enc 1st	39.1	57.1	31.8	51.1	29.5	47.9
BERT-enc 2nd	40.0	58.7	34.7	53.8	30.6	51.2
Multi-modal En⇒De Model						
Our Model	**42.7**	**60.7**	**35.5**	**54.9**	**32.8**	52.2
- $\mathcal{L}_{\text{M}}$	41.8	60.0	34.7	54.6	32.3	**52.3**

Table 5: Comparison with variants of text-only models. 1st and 2nd means the 1st and 2nd stage of training.

4.4 Probing Textual Language Model

Our implemented text-only transformer only uses the source-target parallel corpus extracted from Multi-30K, which overlooks the power of the pre-training on the source side. Because our multi-modal encoder has been fully pre-trained, we systematically compare it with another two text-only baselines. The first baseline virtually has the same architecture as multi-modal framework but without vision language pre-training, denoted as BERT-enc. The second one is **BERT-NMT** (Zhu et al., 2019) by incorporating the output of BERT into the attention module of the transformer. We directly run the experiments with their released codebase[3]. Without image data, all text-only models only optimize the cross entropy loss, so we also present the result of our model without the modality loss.

As shown in Table 5, the BERT-NMT is sometimes even worse than the regular transformer. We hypothesize that the existence of too many untrained parameters in the encoder makes the model difficult to optimize on the limited Multi-30K dataset. When we directly use the pre-trained BERT as the encoder and train the model with two-stage schedule, we observe a consistent improvement on the metrics over the regular transformer, i.e., +1.1 BLEU at 1st-stage and +2.7 BLEU at 2nd-stage. Thus, we argue that with the proper 2-stage training strategy, the pre-trained BERT can account for one half of the overall gains in our final model.

4.5 Ablation Study

To validate the contribution of each component in our approach, we conduct a series of incremental experiments to observe the model performances in different scenarios, summarized in Table 6.

---

[3] `https://github.com/bert-nmt/bert-nmt`

Multi-modal Model	Test2016		Test2017		MSCOCO		Average	
	B	M	B	M	B	M	ΔB	ΔM
End2End	38.7	58.3	31.6	53.1	29.1	50.0	-	-
1st-Stage	40.0	57.5	32.4	51.5	30.8	49.8	+1.27	-0.87
+ 2nd-Stage	41.8	60.0	34.7	54.6	32.3	**52.3**	+3.13	+1.83
+ Last Layer Modality loss $\widetilde{\mathcal{L}}_{\mathrm{M}}^{(6)}$	**42.7**	**60.7**	**35.5**	**54.9**	**32.8**	52.2	**+3.90**	**+2.13**
or + All Layers Modality loss$\sum_{i=1}^{6} \widetilde{\mathcal{L}}_{\mathrm{M}}^{(6)}$	41.7	59.9	34.8	54.7	32.0	51.7	+3.07	+1.63
or + Last Layer Modality loss $\mathcal{L}_{\mathrm{M}}^{(6)}(\lambda^{(6)} = 0.4)$	42.1	59.9	34.9	54.6	31.8	51.3	+3.17	+1.46

Table 6: Ablation study of MMT training on the En⇒De dataset after VLP. Different modality losses are exclusive.

Two-Stage Training In previous analysis, we've seen how the 2-stage training can benefit the text-only model. In Table 6, we present the metrics of different multi-modal models. The end-to-end training, similar to the traditional fine-tuning strategy in (Devlin et al., 2019), optimizes all model parameters of the downstream task once the VLP is finished. We found it leads even worse result than optimizing the decoder alone (i.e., 1st-stage training) on the metric BLEU. In addition, the result after the 2nd-stage fine-tuning produces significant performance increase. We also plot the learning curve of BLEU on development dataset in Figure 3. The apparent gap between two curves confirms the contribution of 2-stage training.

Modality Loss Note that the results in the first three lines of Table 6 are achieved by optimizing the cross entropy loss alone. In this study, we will verify the effectiveness of the modality loss in 3 different setups. We found only optimizing the modality loss of the last layer can achieve the best performance. As we discussed before, the query vector of the last layer will directly and maximally influence the generation of the target token, while the vectors from remote layers seem not important. We can use the statistics of the learnable λ to avoid the time-consuming cross-validation. For example, we set λ as the approximate mean 0.4 in the original modality loss Eq. (4). Although a slightly performance drop appears, we can get rid of 3 trainable vectors.

4.6 Case Studies

Actually, the translation performance of the MMT with vision language model only exceeds about 2 BLEU scores compared with the NMT with BERT language model. So we cannot guarantee that all sentences in the testsets can be better translated by MMT with VLP. We only exemplify two cases with better translation quality for MMT with VLP, to indicate the potential benefits.

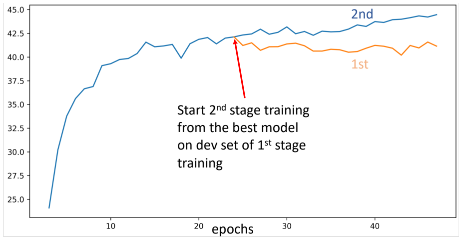

Figure 3: Learning curve of two-stage training w.r.t. BLEU on development set.

In the first case, German words "personen" and "leute" both mean "people", where leute is a general expression and can't be in singular, and "personen" is a formal expression when stating how many people. In object detection model, the tag "person" possibly enhances the NMT model to produce a similar German word "personen". In addition, person is also a German word.

The second case comes from the Ambiguous COCO testset. The NMT with BERT language model cannot miss the translation of the word pizza. The detected object "pizza" may also emphasize the word and help the MMT, though MMT translated the rectangular pizza to stein-pizza (stone-pizza).

4.7 Discussion

The major limitation of our method is that the training pipeline cannot easily generalize to other source languages other than English, because the image-text paired data is unavailable in other languages. Liu et al. (2020) presented a sequence-to-sequence denoising auto-encoder pre-trained on large-scale monolingual corpora in many languages, and successfully applied to multi-lingual translation. Hopefully, we can explore the similar unsupervised cross-lingual or zero-shot transfer learning techniques, which help adapt the multi-lingual BERT

src	four people relaxing on a grassy hill overlooking a rocky valley .
ref	vier personen entspannen auf einem grasbewachsenen hügel mit ausblick auf ein felsiges tal .
brt	vier leute entspannen sich auf einem grasbewachsenen hügel mit blick auf ein steiniges tal .
vlp	vier personen entspannen sich auf einem grasbewachsenen hügel mit blick auf ein steiniges tal .
src	a girl with arms crossed leaning on counter over a rectangular pizza , by a wall calendar and containers .
ref	ein mädchen mit gekreuzten armen stützt sich auf eine theke mit einer rechteckigen pizza , neben einem wandkalender und behältern .
brt	ein mädchen mit verschränkten armen lehnt sich mit überkreuzten armen an einer theke neben einer wand und kartons .
vlp	ein mädchen mit gekreuzten armen lehnt sich über eine theke neben einer wand , auf der sich ein stein-pizza und behälter steht .

Table 7: Case Studies

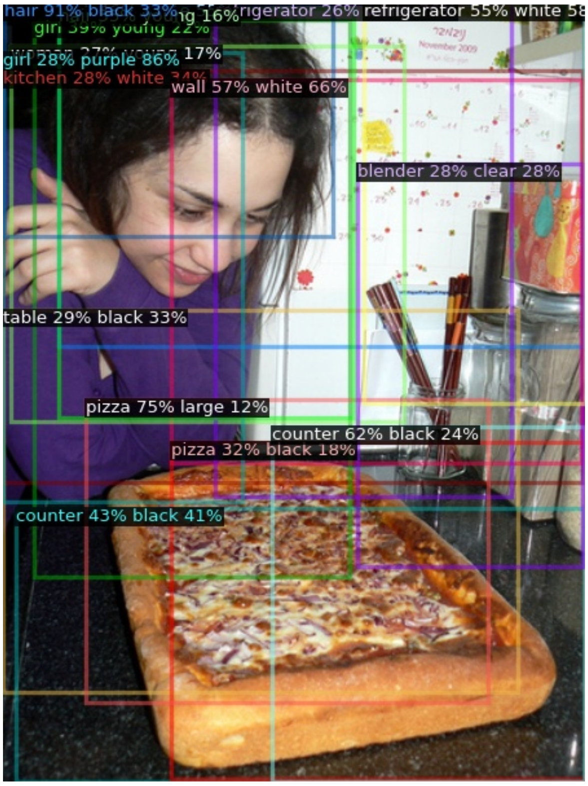

Figure 4: The image for the second case

to a vision multi-lingual model. We will leave this direction as our future work. The main purpose is not to design a better vision language model for other downstream tasks such as VQA. Note that the contemporary works including ViLBERT (Lu et al., 2019) and Oscar (Li et al., 2020) may share the same idea to utilize pre-trained BERT. Our idea is mostly enlighten by (Ive et al., 2019). Another different approach is VL-BERT (Su et al., 2019a), which required to mask sub-regions of the image and introduced masked ROI classification loss, rather than mimicking the NSP loss in traditional BERT.

5 Conclusion

In this paper, we found the vision language pre-training on the source side can significantly improve the multi-modal machine translation, even without additional target corpus. Although the model architecture is as simple as the regular encoder-decoder transformer, our proposed training pipeline can help the MMT system outperform previous works by a large margin on the Multi-30K dataset. The success of the source-image cross-modality representation learning encourages us to design the modality loss that aims at transferring the pre-trained representations to the target-image pair. The quantitative analysis also demonstrates its effectiveness.

Impact Statement

Vision language pre-training has achieved great success in many NLP tasks. We believe it would definitely benefit the multi-modal translation and expect this work can indicate a new unconstrained scenario.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliott, and Stella Frank. 2018. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 304–323.

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. 2017. LIUM-CVC submissions for WMT17 multimodal translation task. In *Proceedings of the Second Conference on Machine Translation*, pages 432–439, Copenhagen, Denmark. Association for Computational Linguistics.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1913–1924.

Iacer Calixto, Miguel Rios, and Wilker Aziz. 2019. Latent variable model for multi-modal translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6392–6405.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Desmond Elliott. 2018. Adversarial evaluation of multimodal machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2978.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation*, pages 215–233.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913.

Stig-Arne Grönroos, Benoit Huet, Mikko Kurimo, Jorma Laaksonen, Bernard Merialdo, Phu Pham, Mats Sjöberg, Umut Sulubacak, Jörg Tiedemann, Raphael Troncy, et al. 2018. The memad submission to the wmt18 multimodal translation task. *arXiv preprint arXiv:1808.10802*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jindřich Helcl, Jindřich Libovickỳ, and Dušan Variš. 2018. Cuni system for the wmt18 multimodal translation task. *arXiv preprint arXiv:1811.04697*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6700–6709.

Julia Ive, Pranava Swaroop Madhyastha, and Lucia Specia. 2019. Distilling translations with visual awareness. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6525–6538.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. *arXiv preprint arXiv:2004.06165*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*.

Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in neural information processing systems*, pages 1143–1151.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 543–553.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2019a. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*.

Yuanhang Su, Kai Fan, Nguyen Bach, C-C Jay Kuo, and Fei Huang. 2019b. Unsupervised multi-modal neural machine translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10482–10491.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5103–5114.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. `https://github.com/facebookresearch/detectron2`.

Pengcheng Yang, Boxing Chen, Pei Zhang, and Xu Sun. 2020. Visual agreement regularized training for multi-modal machine translation. In *AAAI*, pages 9418–9425.

Zhengyuan Yang, Boqing Gong, Liwei Wang, Wenbing Huang, Dong Yu, and Jiebo Luo. 2019. A fast and accurate one-stage approach to visual grounding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4683–4693.

Shaowei Yao and Xiaojun Wan. 2020. Multimodal transformer for multimodal machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4346–4350.

Yongjing Yin, Fandong Meng, Jinsong Su, Chulun Zhou, Zhengyuan Yang, Jie Zhou, and Jiebo Luo. 2020. A novel graph-based multi-modal fusion encoder for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3035.

Mingyang Zhou, Runxiang Cheng, Yong Jae Lee, and Zhou Yu. 2018. A visual attention grounding neural model for multimodal machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3643–3653.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2019. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4995–5004.

The interplay between language similarity and script on a novel multi-layer Algerian dialect corpus

Samia Touileb
Department of Informatics
University of Oslo
samiat@uio.no

Jeremy Barnes
Department of Informatics
University of Oslo
jeremycb@uio.no

Abstract

Recent years have seen a rise in interest for cross-lingual transfer between languages with similar typology, and between languages of various scripts. However, the interplay between language similarity and difference in script on cross-lingual transfer is a less studied problem. We explore this interplay on cross-lingual transfer for two supervised tasks, namely part-of-speech tagging and sentiment analysis. We introduce a newly annotated corpus of Algerian user-generated comments comprising parallel annotations of Algerian written in Latin, Arabic, and code-switched scripts, as well as annotations for sentiment and topic categories. We perform baseline experiments by fine-tuning multi-lingual language models. We further explore the effect of script vs. language similarity in cross-lingual transfer by fine-tuning multi-lingual models on languages which are a) typologically distinct, but use the same script, b) typologically similar, but use a distinct script, or c) are typologically similar and use the same script. We find there is a delicate relationship between script and typology for part-of-speech, while sentiment analysis is less sensitive.

1 Introduction

Cross-lingual transfer has shown promising results for several tasks, however the effect of and the interplay between typologically related languages and languages that do not share the same script has seen less focus. This is especially true for under-resourced vernacular languages and dialects. In this paper, we focus our work on the Algerian language, a non-standardized vernacular Arabic variety, characterized by the heavy use of both code-switching and borrowings. The existing code-switching can be anything from local Algerian dialects (e.g. region based Algerian or Berber), French, English, Spanish, Modern Standard Arabic (MSA), or other Arabic dialects. The borrowings depend on the speakers' background, but is usually heavily French-based.

Algerian is a spoken language with no standardized writing, and with the rise of social media, it has become a language extensively used to communicate online. Algerian can be written in both Arabic and Latin scripts, and code-switching can therefore occur in a mixture of scripts, or within one same script. Arabic varieties written in Latin script are referred to as Arabizi, with north African languages referred to as North African Arabizi, NArabizi in short (Seddah et al., 2020). For the remainder of the paper, we will refer to Algerian written in Latin script as NArabizi (NA) and Algerian written in Arabic script as Algerian Arabic (DZ).

The broad usage of Algerian results in large amounts of data, with no resources or tools to automatically process them. To address this issue and further investigate which of scripts and typological differences influence the results the most, we use a corpus of user comments that reflect the nature of the Algerian vernacular dialect: with heavy use of non-standardised spellings and code-switching.

Our main contributions are (i) a new layer of annotations (transliteration, sentiment analysis, topic classification) that build on the Algerian NArabizi treebank corpus (Seddah et al., 2020), (ii) we investigate the interplay of script and typology on cross-lingual transfer for the two tasks part-of-speech (POS) tagging and sentiment analysis (SA); (iii) we give a baseline model for topic categorization for Algerian. All of the data, annotations, and models are made freely available[1].

To the best of our knowledge, the corpus we present in this work is the first dataset of parallel Algerian texts written in NArabizi and DZ, anno-

---

[1] https://github.com/SamiaTouileb/Narabizi

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3700–3712
August 1–6, 2021.

tated on the morphological and syntactic levels, and for which the interplay between typology and script can be investigated. We also believe that it can help developing approaches to tackle the heavy code-switched nature of the language.

In what follows, in Section 2, we give a brief overview of related works. In Section 3, we describe our dataset and annotations, the annotation processes, and give detailed statistics of the data. We start with some benchmark experiments in Section 4, and present in Section 5 our experiments for POS tagging, SA, and topic classification. In Section 6, we summarize and discuss our results, and conclude in Section 7 with our main findings and future plans.

2 Related work

The vernacular Algerian language is under-resourced, and few freely available corpora and tools exist. Despite work in recent years on this language (Adouane et al., 2020; Moudjari et al., 2020; Adouane et al., 2018; Adouane and Dobnik, 2017; Cotterell et al., 2014), there is only one corpus manually annotated for morphological and syntactical analysis (Seddah et al., 2020).

As pointed out by Seddah et al. (2020), Algerian is a non-codified spoken Semitic language. It is a morphologically-rich language (Tsarfaty et al., 2010), although less so than MSA (Saadane and Habash, 2015). Similarly to other north African languages, it uses heavy code-switching and borrowings, which can either be lexicalized borrowings that receive Arabic-like morphology, or borrowings that remain invariant or take the morphology of the borrowings' original language (*e.g.,* French). Furthermore, Algerian exhibits high variance at the morphological and phonological levels, as well as the lexicon and conventions (Seddah et al., 2020). As shown in Table 1, the Arabic name of the country "*Algeria*" can be written in various ways in both NArabizi and DZ scripts.

As in other North African languages written in Latin script, phonemes that do not exist in the Latin alphabet are represented by digits that are visually similar. For example Table 2 shows how the digits *3* and *9* are used to represent the Arabic letters "*ayin*" and "*qāf*" respectively. The nature of the language makes it therefore an interesting avenue to explore the interplay between language similarity and differences in script on cross-lingual transfer.

NArabizi	DZ
al-dzayer	الدزاير
dzayer	دزاير
jazayer	جزاير
al-jazayer	الجزاير
al-jazaair	الجزائر

Table 1: Lexical variations of the word "*Algeria*" in Algerian written in NArabizi and DZ scripts.

Gloss	NArabizi	Arabic	D	Letter
why	we3lach	وعلاش	3	ع (ayin)
he said	9alli	قالي	9	ق (qāf)

Table 2: Example of non-Latin phonemes represented as digits in NArabizi.

The script of NArabizi differs from the more resourceful MSA and French languages, which can be seen as its culturally closest languages. However, Muller et al. (2020) show that transfer learning approaches can be used on NArabizi, both for POS-tagging and dependency parsing. They show that multilingual BERT (Xu et al., 2019) trained on Maltese, French, and English can successfully transfer to NArabizi, despite not being included in pretraining. This shows the potential for multilingual language models to transfer to unseen dialects across scripts.

The effect of language similarity on NLP tasks is well known (Ponti et al., 2019), with several dedicated workshop series (Nicolai et al., 2020; Zampieri et al., 2018). More recently, attention has turned to larger scale analyses of morphological typology effects on language modeling (Gerz et al., 2018; Cotterell et al., 2018; Mielke et al., 2019). Cross-lingual transfer between languages with related typology is more successful than between languages that do not share similar scripts (Murikinati et al., 2020; Anastasopoulos and Neubig, 2019), especially for the study of morphological inflection. Finally, regarding difference in script, Murikinati et al. (2020) find that using high-quality transliteration as preprocessing can improve the accuracy of such models.

However, in contrast to these previous works, we are interested in the interplay between similar typology and difference in script on *cross-lingual transfer* for two *supervised tasks*, namely POS tag-

NArabizi	*ycombati la misere li las9at fina welat kiste*
Arabic transliteration	يكومباطي لا ميزار لي لسقت فينا ولات كيست
Code-switched transliteration	يكومباطي *la misère* لي لسقت فينا ولات *kyste*
English translation	he fights the misery that sticks to us and which has become a cyst

Table 3: Example of transliteration annotations into Arabic and code-switched scripts. The NArabizi is from (Seddah et al., 2020). The translation to English is added for readers' comprehension.

ging and sentiment analysis. More precisely, we are interested in investigating if there are differences in performance based on the various Algerian scripts.

3 Data and Annotations

The underlying dataset we use is the NArabizi treebank presented in Seddah et al. (2020). This dataset comprises approximately 1,500 sentences: 1,300 NArabizi sentences extracted from an Algerian newspaper's web forum (Cotterell et al., 2014), and 200 sentences from lyrics of songs collected manually from the web. Each NArabizi sentence has five annotation layers: tokenization, morphology, identification of code-switching, syntax, and translation to French (Seddah et al., 2020). The corpus is in *conllu* format, and is freely available[2].

To investigate the interplay between script and typology for cross-lingual transfer on POS tagging and SA, we extend the annotations of Seddah et al. (2020) by adding two levels of annotations:

Token level: for each token of the NArabizi sentences we:

1. transliterate each NArabizi token to Arabic script (*i.e.,* DZ).
2. transliterate each NArabizi token to code-switched scripts (Arabic or Latin) based on the origin of the token (and the code-switch annotation label of the treebank).

Sentence level: we annotate each sentence of the NArabizi corpus for:

1. sentiment: each sentence is annotated as POS (positive), NEG (negative), NEU (neutral), or MIX (a mix of two or more of the three previous classes).
2. topic: each sentence is annotated as belonging to *one* of the following topics: Politics, Prayer, Religion, Societal, Sport, or NONE.

All the annotations were carried out by native speakers of Algerian, Arabic, and French. Two annotators worked on the token-level annotations, and three annotators for the sentence-level annotations. Before starting the annotations, we did a common annotation round to agree on the guidelines, and discuss possible issues. During this, we identified a set of errors in the NArabizi treebank, we therefore started by preprocessing the data and correct some of the recurring errors. More details about our preprocessing of the dataset is given in Section 3.1, the transliteration annotations are described in Section 3.2, and sentiment and topic annotations are described in respectively Section 3.3 and Section 3.4.

3.1 Annotation Preprocessing

The NArabizi treebank dataset (Seddah et al., 2020) contains duplicates both in document IDs and in sentences (strings), both across splits and within splits. Duplicate IDs refer to the same sentences, and therefore duplicate IDs imply duplicate sentences. However, duplicate sentences represent same strings with different IDs. There are far more sentence duplicates than ID duplicates.

All duplicates were removed. However, as the corpus is already quite small, we attempt to avoid removing duplicates from the dev and test splits. If there are duplicates between the train and the dev splits, then we keep the sentences in the dev and remove them from the train set. The same is done with the test split. For the inter-split duplicates, we identified 9 duplicated IDs and 46 (12 unique) duplicated sentences. Intra-split duplicates were only present in train split, with 9 duplicated IDs, and 28 (8 unique) duplicated sentences. We kept

---

[2] `https://parsiti.github.io/NArabizi/`

Letter	Transliteration
v	ف (*f*)
p	ب (*b*)
g	ڨ (*gu*)

Table 4: Normalization of some of the Latin characters that do not have equivalent phonemes in Arabic.

one occurrence of each as it seems that most of these duplicates come from the chorus of the song lyrics, and short common utterances as *e.g.,* "viva Algeria".

3.2 Transliteration to Arabic and code-switched scripts

Two annotators expanded the annotations of the NArabizi treebank by Seddah et al. (2020) by adding for each token of each sentence a transliteration into Arabic script, and a code-switched version that includes both Latin and Arabic scripts. The Latin script is used for tokens that originate from Latin-scripted languages.

For example, Table 3 shows how the NArabizi sentence is transliterated into the corresponding DZ and code-switched scripts. The first word, "يكومباطي", is actually a borrowing from French. However, borrowings that are integrated into the Algerian language lexicon, and that are influenced by Arabic verbal inflections, were not written in Latin script in the code-switched annotations.

The two annotators were given a subset of 300 sentences to transliterate, *i.e.,* these were doubly annotated. Due to the lack of codification, we do not compute any inter-annotator agreement. The subset of the 300 sentences were mainly used to set the annotation guidelines, and were extensively discussed by the annotators.

We decided to normalize some of the Latin characters that do not have equivalent pronunciations in Arabic, these were transliterated into what the native annotators deemed to be the corresponding Arabic characters. In Table 4 we show the Latin letters and the Arabic form they were transliterated to. Even so, we decided to transliterate the last letter (phoneme *gu*) into a non-native Arabic letter. This letter is vastly used in various Algerian dialects, it represents the dialectal pronunciation of *qāf*, and is also used in names of places and persons.

We are aware of the various efforts to develop guidelines for conventional orthography of Algerian and other Arabic dialects (Saadane and Habash, 2015; Habash et al., 2018; Adouane et al., 2019), but we decided to keep the transliterations as identical as possible to the original NArabizi pronunciations and spellings, to reflect the distinctiveness of the language and its use in normal settings in social media.

During the transliteration annotations, several issues were identified in the original NArabizi treebank by Seddah et al. (2020). However, since our annotators were not trained to alter the dependency treebank, only a small selection of the identified errors were corrected.

The first problem encountered is a lack of consistency in the tokenization. For example, the definite article "ال" ("*el*") can be found both as a standalone token, or attached to a word. The same applies to the adposition "*in/on*" ("fi" – "في") where it can be found both as a stand-alone token, and attached to the next word. For example, it was kept with the token in "*f'doute*" ("*in doubt*"), while it was tokenized as "*f+almarikhe*" for the word "*falmarikhe*" ("*on Mars*"). All tokenization errors were not corrected, as this would lead to altering the dependency trees, and as previously mentioned, our annotators were not trained for this task.

Secondly, there were also errors in the translations from NArabizi to French. This is likely due to non-native Algerian speakers translating some parts of the NArabizi treebank. We only corrected the translations that did not alter the tree, *i.e.,* the POS did not change. Some examples of these types of errors can be found in Table 10 in Appendix A.

Finally, we also found some errors in the marker for code-switching (label *lang* in the data). Some Algerian tokens were marked as French, and vice-versa. This also happened with other languages present in the data (as Spanish, English, and MSA). One of the typical errors was the acronyms of football clubs which were all labeled as Algerian. These were corrected to French, since the acronyms come from their names in French. For example the football club "*MCA*" stands for "*Mouloudia Club d'Alger*", while the Arabic name is "نادي مولودية الجزائر" ("*Nadi mouloudiat al-jazair*").

3.3 Sentiment annotations

The sentences were classified based on their polarities into four different classes: POS (positive),

		Train	Dev	Test
Sentiment	POS	291	32	59
	NEG	274	44	34
	NEU	191	21	20
	MIX	242	40	31
Topic	NONE	300	34	36
	Politics	80	11	16
	Prayer	38	9	9
	Religion	17	4	1
	Societal	204	25	31
	Sport	359	54	51

Table 5: Distribution of sentiment and topic annotations.

NEG (negative), NEU (neutral), and MIX (mixed). The annotation guidelines were quite simple, and annotators were asked to use POS and NEG in clear positive and negative cases respectively. If a sentence does not express any kind of polarity, then NEU was assigned. When sentences express a combination of two or more of the POS, NEG, or NEU polarities, annotators were asked to assign the MIX label. The inter-annotator agreement using *Cohen's kappa coefficient* κ is 0.71 on the doubly annotated subset of 300 sentences. Table 5 shows the distribution of the four labels across the training, development, and test sets. The distribution is unbalanced, and the large amount of sentences categorized as MIX can be problematic as it can contain all other polarities. However, the difference between the POS and NEG classes is relatively small, which we believe should be suitable for binary sentiment classification tasks.

3.4 Topic annotations

After a first round of common analysis in collaboration with the annotators, we identified five topics. However, some sentences were difficult to classify and we therefore decided to include the category "*NONE*". The final dataset is annotated for the following six categories: (1) *Politics*: contains all sentences referring or discussing political events or issues; (2) *Prayer*: all sentences representing prayers; (3) *Religion*: sentences discussing religious issues or issues related to religion in general; (4) *Societal*: societal related discussions. Covers everything from schools and teaching, to terrorism and extremism; (5) *Sport*: mainly covering football events, but spans all types of sports and related events; (6) *NONE*: sentences that were impossible to categorize. This was mainly due to the lack of context, as some sentences were comments responding to either articles or other comments. The final κ score for the triply annotated 300 sentences was 0.70.

		NA	DZ	CS
Sentiment	BOW	47.5	45.1	**49.5**
	AVE	52.5	43.2	36.7
	CNN	50.2	**50.4**	46.2
	BiLSTM	**53.9**	45.9	45.6
Topic	BOW	25.8	34.9	**38.1**
	AVE	40.9	44.6	22.8
	CNN	24.4	33.4	27.4
	BiLSTM	**49.4**	**57.0**	36.4

Table 6: Benchmark results for Sentiment Analysis and Topic classification on the three varieties of the dataset: NArabizi (NA), Algerian Arabic (DZ), and a code-switched version (CS). Sentiment and Topic are both Macro F_1.

Table 5 also shows the distribution of topics across the three splits. Most sentences were classified as "Societal" and "Sport". A large amount of sentences could not be categorised, and few sentences were related to "Religion" and "Prayer". Due to the size of the two latter, one could argue that they could be collapsed into a single topic, as done in our benchmarking experiments (see Section 4). However, we decided to keep them separate in the annotations, to facilitate further annotations in the future.

4 Benchmarking experiments

We perform benchmark experiments for SA and topic classification. Specifically, we use the setup from Barnes et al. (2017), who perform experiments with a logistic regression classifier with bag-of-words features (BOW) and averaged embedding features (AVE), as well as a CNN and BiLSTM. We use their default value for hyperparameters (c=1, hidden dimension = 100, dropout = 0.3) and train for 20 epochs, finally testing the best model on the dev set. As the label distribution for both tasks is highly skewed, we use Macro F_1 to evaluate. Given the size of the categories "Prayer" and "Religion", we collapse them to a single topic, converting the topic classification task into a 5-class multi-class problem.

		UPOS		Sentiment			
group	language	# sents.	avg. len.	# docs.	avg. len.	pos	neg
Original	NArabizi	1,276	16.1	731	14.4	380	351
`Script`	Persian	5,997	26.3	879	49.6	419	460
`Script`	Urdu	5,130	27.0	980	17.5	480	500
`Typology`	Hebrew	6,216	30.6	12,434	24.0	8,512	3,922
`Typology`	Maltese	2,074	21.8	719	18.7	237	482
`Both`	MSA	7,664	42.3	51,051	60.8	42,828	8,223

Table 7: Statistics of UPOS and binarized sentiment data.

For the NArabizi and code-switched experiments, we create 100-dimensional fasttext embeddings (Bojanowski et al., 2017) on the unlabeled NArabizi data made available by Seddah et al. (2020). For the DZ experiments, we use available 300-dimensional MSA fasttext embeddings[3] trained on Wikipedia articles.

Table 6 shows the results. On the sentiment task, the BiLSTM performs best on NArabizi, the CNN best on DZ, and BOW best on the code-switched data. For topic classification the performance is similar, but BiLSTM is also best on DZ. The fact that BOW performs best on code-switched data is largely due to the large amount of out-of-vocabulary words for all other methods, which require embeddings. These baseline experiments show that the dataset is challenging, and the variation means that no single model is always best. The code-switched setting is particularly challenging.

5 The interplay between language similarity and script

The transliteration and further sentiment and topic annotations allow us to explore what interplay there is between typology and script in cross-lingual transfer. Muller et al. (2020) perform experiments on zero-shot cross-lingual transfer for POS tagging on NArabizi. They find that the best transfer language is Maltese, a Semitic language which is written in Latin script, rather than MSA, which performs poorly. This begs the question: *is it mainly similar typology or a shared script that leads to this result?* The transliterated dataset, along with the further sentiment annotations, allow us to investigate this question in more depth, as we are able to control for the script choice.

We choose Persian and Urdu, languages written in Arabic script, but morphologically distinct from DZ (we refer to this group as `Script`), Hebrew and Maltese, two Semitic languages written in other scripts (`Typology`) , and MSA, which is both morphologically similar and written in Arabic script (`Both`). These languages are both available in UD (Zeman et al., 2020) and also have available sentiment analysis datasets (Hebrew (Amram et al., 2018), Maltese (Dingli and Sant, 2016), MSA (Nabil et al., 2015; Abdulla et al., 2013), Urdu (Khan and Nizami, 2020), Persian (Hosseini et al., 2018)). As not all sentiment datasets have the same labels as the NArabizi dataset, we remove all neutral and mixed labels and create binary sentiment data for all languages.

Table 7 gives an overview of the statistics of the POS and SA datasets, respectively. The NArabizi data is the smallest POS data (1,276 sentences), followed by Maltese (2,074), Urdu (5,130), Persian (5,997), Hebrew (6,216), and finally MSA (7,664). The average sentence lengths in tokens range between 16.1 for NArabizi and 42.3 in MSA. The sentiment datasets have a larger variance, ranging from 719 sentences for Maltese to 51,051 for MSA. The distribution of polarity is also skewed to a different degree in each dataset.

5.1 Modeling

We model universal POS (UPOS) tagging as a sequence labeling task and SA as a classification task using multilingual BERT (Xu et al., 2019). We finetune each model on the available training data in each language, using a shared set of hyperparameters which were selected from recommended values according to the characteristics of our data. We set the learning rate to 2e-5, max sequence length of

---

[3] Available at `https://fasttext.cc/docs/en/pretrained-vectors.html`.

256, batch size of 8 or 16[4], and perform early stopping once the validation score has not improved in the last epochs, saving the model that performs best on the dev set. We then test each model on its own dev and test data, the NArabizi test set, and finally the transliterated data. We use accuracy as our metric for POS and macro F_1 for sentiment, as the latter often contains unbalanced classes, and define a baseline as the result of predicting the majority class.

6 Results and Discussion

In order to quantify the zero-shot loss, we define a measure of average transfer loss between a group in Equation 1:

$$TL_{x \to y} = S_{x \to x} - S_{x \to y} \tag{1}$$

where $TL_{x \to y}$ is the transfer loss experienced by a model fine-tuned in language x when transferring to language y and $S_{x \to y}$ is the score[5] achieved when testing a model fine-tuned in language x on language y. Thus, it is a measure of the performance lost in the transfer process.

We also define its averaged variant:

$$\overline{TL}_{A \to B} = \frac{1}{N_A} \sum_{i \in A} \overline{TL}_{i \to B} \tag{2}$$

where $\overline{TL}_{A \to B}$ refers to the average transfer loss experienced by languages from any group A to languages from group B (*group-to-group* transfer loss) and N_A is the number of languages included in the experiment that belong to group A (in our case, either languages that have similar typology, or have the same script).

6.1 POS

Table 8 shows the results for the POS tagging. For completeness, we compare with the results from Seddah et al. (2020), who use a feature-based alVWTagger, described in more detail in de La Clergerie et al. (2017) and Muller et al. (2020), who use mBERT and the StanfordNLP tagger (Qi et al., 2018).

Hebrew has the best test accuracy (96.8) and Maltese the worst (93.8), while the others are somewhere between. All models perform better on the transliterated data than the original NArabizi, although training on Urdu performs lower than the majority baseline. This suggests that even though mBERT was not pretrained on NArabizi or DZ, there is a preference for DZ. This is likely due to the fact that at least *some* of the words have been seen in pretraining, *i.e.,* through MSA. An analysis of the tokenization shows that mBERT splits NArabizi words at a much higher rate than DZ (see Figure 1), breaking it into smaller pieces, which may account for some of the differences between the two. The fact that training on Maltese achieves the best score on both NArabizi (37.8) and DZ (38.4), however, suggests that there is still an effect of typology.

The monolingual model trained and tested on DZ performs better (82.5 acc.) than the one trained

	Dev	Test	NA	DZ
Maj.	–	–	19.9	19.9
1-NArabizi		–	80.4	–
2-NArabizi	–	–	81.6	–
2-Maltese	–	–	35.1	–
NArabizi	77.1	–	76.3	43.6
Algerian (DZ)	83.2	–	39.9	82.5
Persian	95.8	95.5	22.7	26.5
Urdu	94.0	93.4	18.7	21.6
Hebrew	97.4	96.8	32.7	38.2
Maltese	93.6	93.0	**37.8**	**38.4**
MSA	97.0	96.7	20.0	30.5

Table 8: POS accuracy when training on Train Lang. Dev Acc. and Test are in-language, while Test Acc. on Narabizi and Algerian (DZ) is zero-shot cross-lingual. 1-Seddah et al. (2020), 2-Muller et al. (2020).

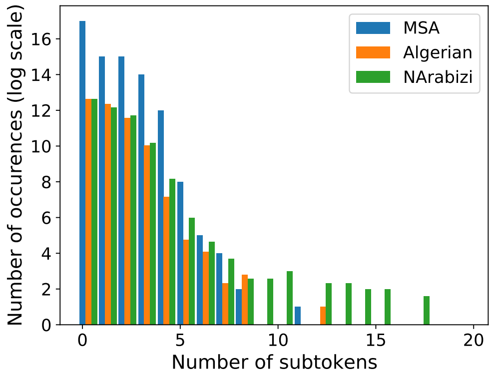

Figure 1: The effect of mBERT tokenization on MSA, Algerian Arabic (DZ), and NArabizi.

---

[4] Depending on the size of the training set, model architecture, and available GPU memory.

[5] The score metric will depend on the task: accuracy in POS and macro F_1 in sentiment analysis.

and tested on NArabizi (76.3). When each of these models is tested on the other, they have significant transfer losses (32.7 for NArabizi → DZ, and 42.6 for DZ → NArabizi). Here too, transfer to DZ script seems easier.

On POS, the effect of language typology is stronger than script, with the best results achieved by training on Maltese and Hebrew. The average transfer loss from Persian and Urdu to DZ is 70.4 while for Hebrew and Maltese to NArabizi it is 59.6, showing less transfer loss from `Typology`. MSA has higher transfer loss on NArabizi (76.7) than DZ (66.2). The differences between average transfer loss on NArabizi and DZ are also slightly larger for `Script` (3.4) compared to `Typology` (3.1) or MSA (3.1).

All of this points to a complicated relationship between script and typology on POS. First of all, it is clear that mBERT prefers the Arabic script seen in pretraining. At the same time, typological similarity also plays a strong role in cross-lingual transfer in POS, although even in this case, the best scores are found on DZ.

6.2 Sentiment

Table 9 shows the results for sentiment analysis. Training in-language again produces the best results (72.1 and 80.3 on NArabizi and DZ, respectively). The transfer loss from NArabizi to DZ is relatively low (9.0), while inversely it is immense (52.6).

Like on POS, most models perform better on DZ and the best zero-shot results do not come from training on `Typology`. In fact, quite the opposite, as these lead to the worst scores and have the highest average transfer loss (34.8/27.9). The best models are MSA for NArabizi (62.4) and Urdu for DZ (63.9), which curiously performs better than NArabizi → DZ. MSA has transfer losses of 12.8/25.1, while `Script` have the lowest average transfer loss (9.2/4.2). This suggests that cross-lingual transfer for a more semantic task, *e.g.*, sentiment analysis, is less reliant on both typological and script similarities.

6.3 Analysis of results

As domain differences between datasets could also lead to transfer loss, we control for this variable by first translating all data to English (to use as a pivot language) and calculating domain difference using Proxy A-distance. Proxy A-distance (Glorot et al., 2011) measures the generalization error of a linear SVM trained to discriminate between two domains. We translate 1,000 sentences from each dataset to English using GoogleTranslate and then compute the proxy A-distance[6] We show heat maps for the domain distances in Figure 2.

	Dev	Test	NA	DZ
Maj.	–	–	39.0	39.0
NArabizi	78.8	–	72.1	63.1
Algerian (DZ)	84.9	–	27.7	80.3
Persian	65.9	66.2	56.9	56.2
Urdu	59.0	62.4	53.3	**63.9**
Hebrew	88.4	88.7	47.2	52.2
Maltese	63.7	61.8	33.8	42.5
MSA	74.2	75.2	**62.4**	50.1

Table 9: Macro F_1 on the zero-shot cross-lingual sentiment task. Note that these results are not comparable to the benchmark experiments, as the data has been converted to binary sentiment classification in order to perform the cross-lingual experiments.

For POS tagging, there are small but insignificant negative effects of proxy A-distance on results (a Pearson coefficient of -0.264, $p > 0.05$). On the sentiment task, there is no significant domain effect (0.264, $p > 0.05$). This suggests that most of the transfer loss is not due to domain mismatch.

7 Conclusion and Future work

In this paper we have described the process of annotating an available Algerian corpus with sentiment and topics, as well as the transliteration to Arabic and code-switched scripts, and finally some aspects of corpus cleanup. We performed benchmark experiments on the three script varieties and show that they are a challenging testbed for future experiments.

We used this new resource to explore a valuable research question in cross-lingual transfer: namely, what is the interplay between language similarity and script when choosing a source language? We found there is a delicate interplay between similar typology and script for transfer in part-of-speech tagging, where typology is more important, but having seen the script in pretraining also influences results. Sentiment analysis, on the other hand, is less

---

[6] Implementation adapted from the code available at `https://github.com/rpryzant/proxy-a-distance`.

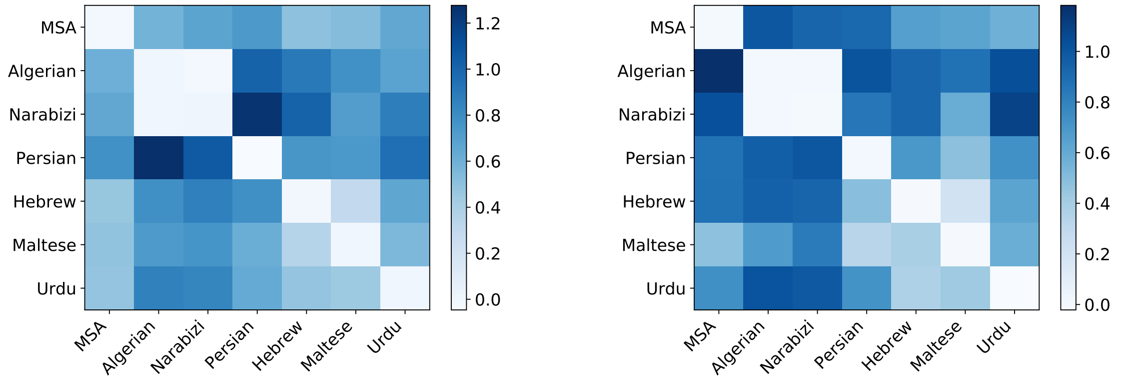

Figure 2: Pairwise proxy A distance between English translations of sentiment (left) and POS (right) datasets.

sensitive to typological differences, while still preferring the script seen in pretraining. This suggests that choice of transfer language is task-specific and that surprising differences can appear from one task to another.

In the future, we would like to address data related issues, and correct the tokenization and translation issues discussed in Section 3.1. Moreover, we plan to focus more concretely on the code-switching aspect of our dataset. The challenges of code-switched data to NLP techniques are numerous, and we would like to focus on the syntactic analysis of our code-switched data, and to explore in more details language modeling approaches to processing it.

References

Nawaf Abdulla, Nizar A. Ahmed, Mohammed Shehab, and Mahmoud Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies, AEECT 2013*, pages 1–6.

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2018. Improving neural network performance by injecting background knowledge: Detecting code-switching and borrowing in Algerian texts. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 20–28, Melbourne, Australia. Association for Computational Linguistics.

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2019. Normalising non-standardised orthography in Algerian code-switched user-generated data. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 131–140, Hong Kong, China. Association for Computational Linguistics.

Wafia Adouane and Simon Dobnik. 2017. Identification of languages in Algerian Arabic multilingual documents. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 1–8, Valencia, Spain. Association for Computational Linguistics.

Wafia Adouane, Samia Touileb, and Jean-Philippe Bernardy. 2020. Identifying sentiments in Algerian code-switched user-generated comments. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2698–2705, Marseille, France. European Language Resources Association.

Adam Amram, Anat Ben David, and Reut Tsarfaty. 2018. Representations and architectures in neural sentiment analysis for morphologically rich languages: A case study from modern Hebrew. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2242–2252, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, page 984–996, Hong Kong, China,. Association for Computational Linguistics.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2017. Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–12, Copenhagen, Denmark. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 536–541, New Orleans, Louisiana. Association for Computational Linguistics.

Ryan Cotterell, Adithya Renduchintala, Naomi Saphra, and Chris Callison-Burch. 2014. An algerian arabic-french code-switched corpus. In *Workshop on free/open-source Arabic corpora and corpora processing tools workshop programme*, page 34.

Alexiei Dingli and Nicole Sant. 2016. Sentiment analysis on maltese using machine learning. In *Proceedings of The Tenth International Conference on Advances in Semantic Processing (SEMAPRO 2016)*, pages 21–25.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 316–327, Brussels, Belgium. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 513–520, Madison, WI, USA. Omnipress.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al-Shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Pedram Hosseini, Ali Ahmadian Ramaki, Hassan Maleki, Mansoureh Anvari, and Seyed Abolghasem Mirroshandel. 2018. Sentipers: A sentiment analysis corpus for persian. *CoRR*, abs/1801.07737.

M. Y. Khan and M. S. Nizami. 2020. Urdu sentiment corpus (v1.0): Linguistic exploration and visualization of labeled dataset for urdu sentiment analysis. In *2020 International Conference on Information Science and Communication Technology (ICISCT)*, pages 1–15.

Éric de La Clergerie, Benoît Sagot, and Djamé Seddah. 2017. The ParisNLP entry at the ConLL UD shared task 2017: A tale of a #ParsingTragedy. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 243–252, Vancouver, Canada. Association for Computational Linguistics.

Sebastian J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4975–4989, Florence, Italy. Association for Computational Linguistics.

Leila Moudjari, Karima Akli-Astouati, and Farah Benamara. 2020. An Algerian corpus and an annotation platform for opinion and emotion analysis. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1202–1210, Marseille, France. European Language Resources Association.

Benjamin Muller, Benoit Sagot, and Djamé Seddah. 2020. Can multilingual language models transfer to an unseen dialect? a case study on north african arabizi.

Nikitha Murikinati, Antonios Anastasopoulos, and Graham Neubig. 2020. Transliteration for cross-lingual morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–197, Online. Association for Computational Linguistics.

Mahmoud Nabil, Mohamed Aly, and Amir Atiya. 2015. ASTD: Arabic sentiment tweets dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2515–2519, Lisbon, Portugal. Association for Computational Linguistics.

Garrett Nicolai, Kyle Gorman, and Ryan Cotterell, editors. 2020. *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Online.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2019. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics*, 45(3):559–601.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Houda Saadane and Nizar Habash. 2015. A conventional orthography for Algerian Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 69–79, Beijing, China. Association for Computational Linguistics.

Djamé Seddah, Farah Essaidi, Amal Fethi, Matthieu Futeral, Benjamin Muller, Pedro Javier Ortiz Suárez, Benoît Sagot, and Abhishek Srivastava. 2020. Building a user-generated content North-African Arabizi treebank: Tackling hell. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1139–1150, Online. Association for Computational Linguistics.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12.

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Nikola Ljubešić, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali, editors. 2018. *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Željko Agić, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Angelina Aquino, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Furkan Atmaca, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Ethan Chi, Jinho Choi, Yongseok Cho, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Loïc Grobol, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Tunga Güngör, Nizar Habash, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Oliver Hellwig, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Ḣildur Jónsdóttir, Fredrik Jørgensen, Markus Juutinen, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Abdullatif Köksal, Kamil Kopacewicz, Timo Korkiakangas, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phươ ng Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Maria Levina, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Yuan Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Hiroshi Matsuda, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayò Olúòkun, Mai Omura, Emeka Onwuegbuzia, Petya Osenova, Robert Östling, Lilja Øvrelid, Şaziye Betül Özateş, Arzucan Özgür, Balkız Öztürk Başaran, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Petru Rebeja, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Valentin Roșca, Davide Rovati, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Salvatore Scarlata, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Maria Skachedubova, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Takaaki Tanaka, Samson Tella, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Marsida Toska, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Utku Türk, Francis Tyers, Sumire Uematsu, Roman Untilov, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Aya Wakasa, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Paul Widmer, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Kayo Yamashita, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Hanzhi Zhu, and Anna Zhuravleva. 2020. Universal dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

A Appendix

id – Sentence (En)	Translation	Correct (En)	Explanation	Status
1 – *Mabrouk ya lafhal* 1 (congratulations oh brave)	Lafhal	courageux (brave)	annotated as PROPN, should be NOUN.	X
2 – *el hamdou lilah ya rabi alla 3awdat chawchi* (thanks God for the return of Chawchi)	Allah	alla (for)	this is the word على annotated as PROPN, should be DET.	X
3 – *vive toi mbolhi* (long live you Mbolhi)	fou (crazy)	Mbolhi	Mbolhi is the name of a football player. It is not an ADJ, should be PROPN.	X
4 – *mabka fiha ghure sehab elderaham* (the only ones remaining are those with money)	pas pleurer (not cry)	mabka (only remain)	this is the word ما بقى (only remain) and not ما بكى (not cry).	✓
5 – *al mou3ak fil jazair mayakdarch yakhrouj* (the handicapped in Algeria can't go out)	obstacle (obstacle)	handicapé (handicapped)	the word *mou3ak* in this context means handicapped.	✓

Table 10: Examples of errors present in the NArabizi treebank. Status "X" means not corrected, while status "✓" means corrected.

Few-Shot Upsampling for Protest Size Detection

Andrew Halterman
Massachusetts Institute of Technology
`ahalt@mit.edu`

Benjamin J. Radford
UNC Charlotte
`benjamin.radford@uncc.edu`

Abstract

We propose a new task and dataset for a common problem in social science research: "upsampling" coarse document labels to fine-grained labels or spans. We pose the problem in a question answering format, with the answers providing the fine-grained labels. We provide a benchmark dataset and baselines on a socially impactful task: identifying the exact crowd size at protests and demonstrations in the United States given only order-of-magnitude information about protest attendance, a very small sample of fine-grained examples, and English-language news text. We evaluate several baseline models, including zero-shot results from rule-based and question-answering models, few-shot models fine-tuned on a small set of documents, and weakly supervised models using a larger set of coarsely-labeled documents. We find that our rule-based model initially outperforms a zero-shot pre-trained transformer language model but that further fine-tuning on a very small subset of 25 examples substantially improves out-of-sample performance. We also demonstrate a method for fine-tuning the transformer span on only the coarse labels that performs similarly to our rule-based approach. This work will contribute to social scientists' ability to generate data to understand the causes and successes of collective action.

1 Introduction

A common data collection task in social science is applying fine-grained labels to documents, including extracting specific passages from text. In many cases, social scientists already have many coarsely-labeled documents and a small number of hand-annotated documents. An automated technique for "upsampling" from coarse labels to more detailed information could help researchers produce better tailored datasets. However, this process does not fit the tools that applied researchers have access to: training a document classifier on coarse labels will not produce the fine-grained answers. Innovations in zero-shot and few-shot classifiers and information extraction (IE) techniques show promise, but new methods are required that can also draw on the existing coarse document annotations to improve fine-grained extraction.

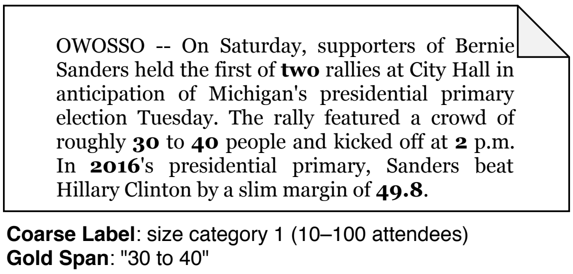

OWOSSO -- On Saturday, supporters of Bernie Sanders held the first of **two** rallies at City Hall in anticipation of Michigan's presidential primary election Tuesday. The rally featured a crowd of roughly **30** to **40** people and kicked off at **2** p.m. In **2016**'s presidential primary, Sanders beat Hillary Clinton by a slim margin of **49.8**.

Coarse Label: size category 1 (10–100 attendees)
Gold Span: "30 to 40"

Figure 1: Documents in our corpus have "coarse labels" reporting the order of magnitude of the protest size and "gold spans" reporting the exact size of the protest. The frequency of number words (in bold) shows why this task is not trivial.

We introduce a new task and dataset for improving information extraction systems' performance when given many coarsely-labeled documents and a small number of documents annotated with the spans of interest.[1] We draw on a dataset on dissent and collective action (hereafter, "protests") in the United States compiled by the Crowd Counting Consortium (2020) (CCC) to construct our training and evaluation data. Protests are an important avenue for social change and of major interest for social science researchers. Current work suggests that attendance is a major factor in the success of a protest movement (Chenoweth and Margherita, 2019), but good data on protest attendance is difficult to collect. CCC compiles structured data about protests from expert annotators using news report-

---

[1] Replication archive available at `https://github.com/benradford/few-shot-upsampling-for-protest-size-detection`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3713–3720
August 1–6, 2021.

ing, including the exact text span from the article that describes the protest's size and the order of magnitude of the crowd size. An example is given in Figure 1. The task we propose is to locate the span within a document that reports the size of a protest, given a training set of documents labeled with the order of magnitude of the protest ("*coarse labels*") and a small number of document pieces (25) with exact span information ("*gold spans*").

Drawing on recent work in question answering, we repurpose existing models to generate fine-grained labels given a large set of coarsely-labeled documents and a small set of documents with fine-grained labels. We provide results from three baseline models, finding that a heuristic, rule-based system outperforms a zero-shot transformer-based question-answering (QA) model. Fine tuning on a small set (25) of gold spans substantially improves performance. We also introduce a new multitask model that reaches equivalent performance despite fine-tuning on *no* gold spans.

2 Task and Data

For each protest in the CCC dataset, we collect the following data: the raw article text (scraped from the CCC-provided URLs), the exact string reporting the protest size, and a "size category" provided by CCC that reports the order of magnitude size of the crowd. The task is to predict the size text string, given plentiful training data with the size category and the gold spans for a small set of partial documents (25 paragraphs). The test set includes only the full article texts and order-of-magnitude information. To make the task tractable, we exclude protests that are coded from multiple documents and documents from which multiple protests are coded. From 48,736 total protests reported by CCC between January 21, 2017 and October 31, 2020, we eliminate multi-document/multi-protest reports and successfully scrape text for 11,005 protests. We eliminate documents where the CCC-reported size text is not located within the document, leaving 3,849 protests/documents. We split these data into four parts:

- **Coarse label training set**: text with coarse, order-of-magnitude labels {0,1,2,3} but no exact answer spans (2,694 full articles).
- **Gold span training set**: short texts with exact answer spans but no order-of-magnitude labels (25 paragraphs).
- **Validation set**: documents with order-of-magnitude labels and exact answer spans (200 full articles).
- **Test set**: documents with order-of-magnitude labels and exact answer spans (930 full articles).

The task is challenging because models are not evaluated on the largest portion of the data (coarse document labels) but rather on a fine-grained span prediction task for which only limited data is available. The task can thus be framed in several ways, depending on which parts of the data are used and in what ways:

- **Zero shot**: use an off-the-shelf model to detect protest sizes without any fine tuning on our data, either coarse or fine.
- **Few-shot on gold spans**: fine tune a baseline model on the small number of gold span labelled data.
- **Coarse labels**: use a coarse-to-fine model to identify spans given only document-level labels.
- **Coarse labels + gold spans**: train a model using both coarse order-of-magnitude labels and limited fine-grained span data.

3 Related Work

The task we propose relates to several strands of research. One framing is as a *question-answering* task (QA), where the same question ("How many people protested?") is asked about each document. A large set of NLP tasks can be framed as question-answering models (McCann et al., 2018) and QA models trained on language models can generalize to new domains with few or no labeled examples (Brown et al., 2020; Radford et al., 2019). QA models have also been successfully used when the training data is noisy (Lin et al., 2018). Given the flexibility of QA models and their strong performance in new domains, we use one as the base of our models.

A different framing is as a *"rationale"* problem for a document classifier. Lei et al. (2016) train a classifier on document-level labels and use attention weights to extract rationales for the classification. Our task differs from the canonical document classification task because a responsive model is evaluated on the extracted spans, not on the coarse label prediction task.

Distant supervision uses noisy labels, often applied automatically or with heuristic labels, to train systems (Ratner et al., 2017). The classic example of distant supervision uses a database of relations to label binary relations in text (Mintz et al., 2009). Weak supervision, more generally, uses labels that are noisy or coarse to train fine-grained models (Khetan et al., 2018; Robinson et al., 2020). Some work on "noisy labels" relates to our task, where labels are presented at a higher level of aggregation rather than with noise. Nayak et al. (2020) propose a model that uses coarse, document-level sentiment labels to train a fine-grained, sentence-level sentiment classifier. Their task differs from ours in the nature of their labels: in moving from document-level to sentence-level labels, they predict labels of the same type (sentiment scores). In our task, we also change the labels themselves, from a crowd size order of magnitude to a token-level label of whether a word describes the exact protest size.

4 Modeling Strategy

We first attempt the task using a rule-based model (the "heuristic keyword model") and an off-the-shelf zero-shot QA system. We then introduce a multi-task neural network model based on a pre-trained transformer language model. We fine-tune and evaluate this model on the coarse labels and gold spans, as well as on noisy labels we generate through a rule-based procedure.

The two standard performance metrics for question answering tasks are exact match and F_1 (Rajpurkar et al., 2018). We compute exact match as the sum of exact matches (predicted spans exactly matched in the set of correct target spans) divided by the total number of documents. We compute F_1 per document based on token-level precision and recall, then average across documents.

4.1 Heuristic Keyword Model

Our heuristic model is a rule-based system that uses keyword matching and dependency parses to return a single number-containing phrase from the article. We first locate all number-containing phrases (digits or number words) in the text with regular expressions. Using a rule-based system, we convert these number phrases to a numeric form (e.g. "several dozen" → 36) and then compare the phrase's numerical value to the protest's reported order of magnitude. If the phrase does not match the order of magnitude, we eliminate it from our candidate list. To further reduce the candidate list, we look for number phrases that occur within the same sentence as a set of keywords such as "crowd", "gathered", or "protesters".[2] If multiple sentences have keyword matches, we return the first one. The CCC data's size spans include modifiers alongside the raw numerical values (e.g. "*about* 20", "*more than* 50"). We use dependency parse information generated by spaCy to extract the wider span.[3]

4.2 Zero-Shot QA Model

We begin with a pre-trained RoBERTa model (Liu et al., 2019) that we subsequently fine-tune for question answering using the Stanford Question Answering Dataset (SQuAD) 2.0 as described in Appendix A (Rajpurkar et al., 2018).[4] The QA model architecture is depicted on the left side of Figure 2. Because we do not tune this model on our dataset, we consider its predictions to be zero-shot.

Model	Exact	F_1
Heuristic rules	0.54	0.61
RoBERTa QA		
zero-shot	0.17	0.27
+ gold spans	0.67	0.65
+ *coarse labels*	0.48	0.54
+ *coarse labels + heuristic spans*	0.66	0.63

Table 1: Exact match ("Exact") and F_1 performance on test set data. All RoBERTa QA and multitask models are fine-tuned on SQuAD 2.0. Multitask models italicized. Full results given in Appendix B.

4.3 Fine-tuned QA Model

To use the coarse labels, we add an additional objective to the QA model that is trained to predict the crowd size order of magnitude. The model first predicts the start and end token vectors for a given context-question pair. We compute the cumulative sum (over tokens) of the predicted start token vector and the reverse cumulative sum for the predicted end token vector. The resulting vectors are element-wise multiplied to produce an attention mask with high values in the range of tokens between the predicted start and end tokens. We apply

---

[2] The complete list is "protesters", "demonstrators", "gathered", "crowd", "rallied", "attended", "picketed", "protest".

[3] Specifically, (1) for each sentence matching a keyword (2) identify the word in the sentence that is a number word or numeric, and (3) also include child nodes that had the following labels: adjectival modifier, modifier of quantifier, compound, adverbial modifier. We used spaCy version 2.3.2 with the `en_core_web_lg` model to perform the dependency parsing and sentence segmentation.

[4] We use `roberta-base` from HuggingFace (Wolf et al., 2020).

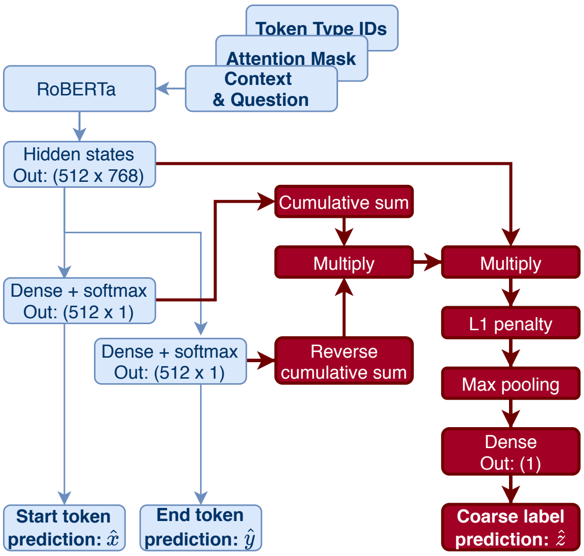

Figure 2: Multitask model architecture: standard RoBERTa QA (left) and attention mask-based regression for coarse label prediction (right).

an L1 penalty to this mask to ensure the attention focuses on a small number of tokens. The attention mask is then element-wise multiplied with the token hidden states produced by RoBERTa. Global max pooling and a single linear regression layer applied to these attended-to hidden states predict the coarse label (as shown in the right side of Figure 2).

The loss function for the multitask model, an unweighted combination of cross-entropy loss and mean squared error, is $-\sum_{i=1}^{n}\{x_i\log(\hat{x}_i)+y_i\log(\hat{y}_i)\} + (\hat{z}-z)^2$, where $x_i \in \{0,1\}$ indicates whether token i is the start of an answer span, $y_i \in \{0,1\}$ indicates whether token i is the end of an answer span, z is the document's coarse label, and n is the number of tokens (512, here). The model can be fit to data including any combination of these three targets.

5 Results

Results on the test set are given in Table 1. RoBERTa QA refers to RoBERTa fine-tuned on SQuAD 2.0. With only fine-tuning on SQuAD 2.0, the model scores 17% exact match accuracy and 27% F_1. On their own, the heuristic-derived spans outperform zero-shot RoBERTa QA. "+ Heuristic spans" indicates that the given model was fine-tuned on the spans identified by the heuristic model.

Fine-tuning the multitask model on the coarse labels alone results in a 180% increase in exact match accuracy and 100% increase in F-score. An example prediction made by the multitask coarse labels model is shown in Figure 3.[5] However, the highest scores are achieved by fine-tuning the RoBERTa QA model on just the 25 gold spans: 67% exact match accuracy and 65% F-score.

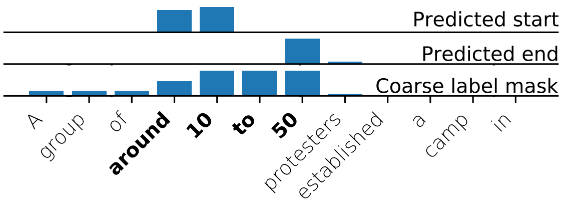

Figure 3: Example target span from document excerpt with predicted start tokens (top), predicted end tokens (middle), and attention mask (bottom). Results from model *c* in Table 1. Actual span in bold.

The greatest performance by a multitask model without any gold spans is achieved by the model fine-tuned on both the coarse labels and the heuristic spans: 66% exact match and 63% F_1, just below the top performing model with access to the gold spans. We interpret the success of this model and the coarse labels model over the base RoBERTa QA model as evidence that our attention masking strategy was successful at upsampling from coarse document-level labels to specific token-level spans.

6 Discussion and Conclusion

Social scientists often find themselves with coarsely-labeled text data for which upsampling may provide valuable additional information. We anticipate applications in extracting fine-grained policy proposals from party manifestos with document-level annotations (Lehmann et al., 2017), the specific armed actors engaged in civil war violence from documents labeled with "rebel" or "government" (Lyall, 2010), or the specific phrases in news text that lead to their censorship (King et al., 2013). We also see applications in upsampling ranges of causalities from NGO reports or Wikipedia articles to the exact sizes, upsampling years to more specific dates, or using rounded numbers from financial disclosures or government reports as coarse supervision for extracting the exact amount from text.

Improvements in zero- and low-shot models should encourage applied researchers to explore computational approaches to text analysis even when training data is scarce, noisy, or coarse—

---

[5] The model just misses an exact match by omitting "around" from the predicted span.

common challenges that are often perceived as intractable. At the same time, NLP researchers should continue to improve models that can learn to extract fine-grained information given coarse training data. Multitask QA models show promise in doing so, but future work can further integrate work from the weak/distant supervision literature, including modeling the noisiness of the labels.

Acknowledgments

We thank Jay Ulfelder of the Crowd Counting Consortium for helpful discussions. Halterman gratefully acknowledges the support of an NSF Graduate Research Fellowship.

Impact Statement

Studies of protests have the potential for serious ethical concerns. Some tasks, such as identifying or de-anonymizing the participants in a protest could produce major harms. Our application, identifying the number of attendees at a protest, has less potential for harm. Our collection of information on the size of protests will generally accord with the desires of protesters. Social scientists have long seen protests as an important tool for social movements to overcome collective action problems: by making support for a position visible in the streets, a protest assures potential supporters of the protest that their opinions are held by others and that the group could potentially achieve its ends with more support (Kuran, 1989; Petersen, 2001; Tarrow, 2011). Providing better information on the size of protests furthers the signalling and information-disseminating objectives of the protesters themselves. While we might not agree with the causes of all protesters in the United States, we believe that on-balance, our work benefits those with less power more than it does those with greater power, who can likely already collect the information they seek manually.

The data that we draw on was collected by the Crowd Counting Consortium, which relies on volunteers and paid research assistants to collect the data. Their protocol was reviewed by the University of Denver IRB and deemed exempt because they do not collect personally identifiable information and use only public data.[6]

A second consideration in our work involves the role of copyrighted news text in our project. Our method uses copyrighted news text that we scraped from the web. While scraping websites is legal in the United States,[7] redistributing copyrighted text is more difficult to justify and depends on how the use fits into the fair use doctrine. Balancing copyright holders' rights with public and educational benefit is at the core of the fair use doctrine.[8] Our attempt to balance the harms to copyright holders and the harms to broader public and scientific benefit is to publish a URL list and scraper so that our corpus can be re-created by future researchers. Additionally, in cases where a researcher is attempting to replicate our work for educational purposes, we will make our scraped corpus available for the narrow purpose of replicating our work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Erica Chenoweth and Belgioioso Margherita. 2019. The physics of dissent and the effects of movement momentum. *Nature Human Behaviour*, 3(10):1088–1095. Copyright - 2019© The Author(s), under exclusive licence to Springer Nature Limited 2019; Last updated - 2020-03-25.

Crowd Counting Consortium. 2020. [link].

Ashish Khetan, Zachary C Lipton, and Animashree Anandkumar. 2018. Learning from noisy singly-labeled data. In *International Conference on Learning Representations*.

Gary King, Jennifer Pan, and Margaret E Roberts. 2013. How censorship in china allows government criticism but silences collective expression. *American Political Science Review*, 107(2):326–343.

---

[6] `https://sites.google.com/view/crowdcountingconsortium/faqs`

[7] https://www.eff.org/deeplinks/2019/09/victory-ruling-hiq-v-linkedin-protects-scraping-public-data

[8] `https://www.copyright.gov/fair-use/more-info.html`

Timur Kuran. 1989. Sparks and prairie fires: A theory of unanticipated political revolution. *Public choice*, 61(1):41–74.

Pola Lehmann, Theres Matthieß, Nicolas Merz, Sven Regel, and Annika Werner. 2017. Manifesto corpus. version: 2017b. Technical report, Berlin: WZB Berlin Social Science Center.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jason Lyall. 2010. Are coethnics more effective counterinsurgents? evidence from the second chechen war. *American Political Science Review*, 104(01):1–20.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Apoorv Nandan. 2020. Text extraction with bert.

Guruprasad Nayak, Rahul Ghosh, Xiaowei Jia, Varun Mithafi, and Vipin Kumar. 2020. Semi-supervised classification using attention-based regularization on coarse-resolution data. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 253–261. SIAM.

Roger D Petersen. 2001. *Resistance and rebellion: lessons from Eastern Europe*. Cambridge University Press.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Joshua Robinson, Stefanie Jegelka, and Suvrit Sra. 2020. Strength from weakness: Fast learning using weak supervision. In *International Conference on Machine Learning*, pages 8127–8136. PMLR.

Sidney G Tarrow. 2011. *Power in movement: Social movements and contentious politics*. Cambridge University Press.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

A Fine-tuning RoBERTa on SQuAD 2.0

A.1 SQuAD 2.0 Fine-Tuning

In order to facilitate extensions to the standard QA model, we perform the fine-tuning of RoBERTa on SQuAD 2.0 ourselves (Abadi et al., 2015). We fine-tune on the SQuAD 2.0 training set for three epochs using the settings recommended by Nandan (2020). We use a batch size of 12 due to memory limitations. We use the Adam optimizer with a learning rate of $5e-5$. Our model achieves 0.78 and 0.74 exact match on the training and evaluation sets, respectively. We use this model only as a basis for subsequent fine-tuning and therefore do not attempt to match state-of-the-art performance on the SQuAD 2.0 evaluation set. The model is trained on two RTX 2080 Ti GPUs. Model size and training time details are provided in Table 2.

We allow the QA model to identify impossible-to-answer questions by predicting the sequence start token ("<s>") as both the answer span start and end token.

To fit within the RoBERTa base model's 512 token limit, we pre-process all text inputs via a shingling procedure. We limit contexts to 450 tokens thereby allowing questions of up to 62 tokens in length. We then pad to a uniform 512 tokens. When contexts exceed 450 tokens, we use a sliding window of 450 tokens that we step through the context 225 tokens at a time. We guarantee all samples generated from large contexts contain precisely 450 tokens by adjusting the first and last window positions such that they do not extend before or after the first or last context token, respectively. We aggregate predictions across shingles by assuming one predicted span per document and selecting the predicted span from the shingle for which $\max_{i\in[1,\ldots,512]}(\hat{x_i})+\max_{i\in[1,\ldots,512]}(\hat{y_i})$ is the greatest.

A.2 Task-Specific Fine-Tuning

The selection of learning rate for these models, 5e-6 (exactly one order of magnitude lower than the default used for SQuAD fine-tuning), was due to our sensitivity to overfitting on the very small set of span examples. All models were trained for 150 batches, each batch comprising 12 samples chosen from the training datasets with replacement. When multiple datasets are used to train the same model, batches alternate between them. We selected the number of batches for training by observing exact match accuracy on the validation set over a range of iteration steps from 1 to 400 and selecting the earliest batch iteration at which validation set accuracy appeared to plateau.

B Results

The full set of fine-tuning data combinations is given in Table 3. All models c through i are trained using the same hyperparameters and strategy (Adam optimizer, 5e-6 learning rate, and 150 batches of size 12 examples each).

		Parameters	Training Time
(a)	Heuristic rules	–	–
(b)	RoBERTa + SQuAD 2.0 (zero-shot)	1.25M	200 min
(c)	+ coarse labels	1.25M	+ 20 min
(d)	+ heuristic spans	1.25M	+ 20 min
(e)	+ coarse labels + heuristic spans	1.25M	+ 20 min
(f)	+ *gold spans*	1.25M	+ 20 min
(g)	+ *gold spans* + coarse labels	1.25M	+ 20 min
(h)	+ *gold spans* + heuristic spans	1.25M	+ 20 min
(i)	+ *gold spans* + coarse labels + heuristic spans	1.25M	+ 20 min

Table 2: Model size in parameters. Training time (approximate) on 2× RTX 2080 Ti GPUs. "+ 20 min" indicates the model takes an additional 20 minutes to fine-tune after the initial fine-tuning on SQuAD 2.0. These estimates may be high due to our validation set performance evaluation between batches.

		Test set		Validation set	
		Exact Match	F_1	Exact Match	F_1
(a)	Heuristic rules	0.54	0.61		
(b)	RoBERTa + SQuAD 2.0 (zero-shot)	0.17	0.27	0.19	0.27
(c)	+ coarse labels	0.48	0.54	0.54	0.58
(d)	+ heuristic spans	0.51	0.50	0.56	0.51
(e)	+ coarse labels + heuristic spans	0.66	0.63	0.72	0.66
(f)	+ *gold spans*	0.67	0.65	0.71	0.68
(g)	+ *gold spans* + coarse labels	0.67	0.65	0.68	0.66
(h)	+ *gold spans* + heuristic spans	0.62	0.61	0.66	0.62
(i)	+ *gold spans* + coarse labels + heuristic spans	0.65	0.64	0.72	0.67

Table 3: Exact match and token-level F_1 performance by each model on test and validation set data.

Modeling the Unigram Distribution

Irene Nikkarinen[*,ð,ɤ] **Tiago Pimentel**[*,ð] **Damián E. Blasi**[ə,ə,ɒ] **Ryan Cotterell**[ð,ʕ]
[ð]University of Cambridge [ɤ]Yle [ə]Harvard University
[ə]MPI for Evolutionary Anthropology [ɒ]HSE University [ʕ]ETH Zürich
irene.nikkarinen@gmail.com, tp472@cam.ac.uk
dblasi@fas.harvard.edu, ryan.cotterell@inf.ethz.ch

Abstract

The unigram distribution is the non-contextual probability of finding a specific word form in a corpus. While of central importance to the study of language, it is commonly approximated by each word's sample frequency in the corpus. This approach, being highly dependent on sample size, assigns zero probability to any out-of-vocabulary (oov) word form. As a result, it produces negatively biased probabilities for any oov word form, while positively biased probabilities to in-corpus words. In this work, we argue in favor of properly modeling the unigram distribution—claiming it should be a central task in natural language processing. With this in mind, we present a novel model for estimating it in a language (a neuralization of Goldwater et al.'s (2011) model) and show it produces much better estimates across a diverse set of 7 languages than the naïve use of neural character-level language models.

1 Introduction

Neural networks have yielded impressive gains in sentence-level language modeling across a typologically diverse set of languages (Mikolov et al., 2010; Kalchbrenner et al., 2016; Merity et al., 2018; Melis et al., 2018; Cotterell et al., 2018). Similarly, neural networks constitute the state of the art in modeling the distribution over a language's word types (Pimentel et al., 2020), outperforming non-neural generative models such as Futrell et al.'s (2017) with character-level models. This paper focuses on a less-researched task that is halfway between sentence-level language modeling and word type distributions: Modeling the **unigram distribution**, the distribution over word tokens in a language consisting of the probability of a word's form as well as its frequency in the language. In particular, as opposed to sentence-level modeling, the unigram distribution does not consider contextual information.

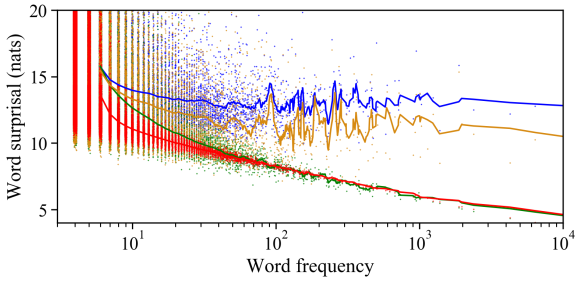

Figure 1: Word-level surprisal in Finnish under our **two-stage** model, two baseline LSTMs trained with either word **type** or **token** data, and another LSTM called the **generator**, trained on an interpolation of both. Lines depict rolling averages.

The unigram distribution is a central object in the science of language from historical linguistics to psycholinguistics and beyond (Baayen et al., 2016; Diessel, 2017; Divjak, 2019). However, the majority of research on unigram distributions is based on identifying this distribution with sample frequency. This approach results in poor estimates, as it assigns zero probability to out-of-vocabulary words.[1] Further, it is highly dependent on sample size (Baayen, 2002)

The core contribution of our work is motivating the unigram distribution as a worthwhile objective for scientific inquiry—one which is currently understudied in the field. With that in mind, we also present a neuralization of Goldwater et al.'s (2011) two-stage model.[2] The gist of this approach is using two components to model the Zipfian distribution of word tokens in a language (Zipf, 1935) separately from its phono- or graphotactic distribution. The first component, termed the **adaptor**, is

---

[*] Equal contribution

[1] In the Turkish Wikipedia, for example, considering a training set of 8 million and a test set of 1 million tokens, 27.4% of test types and 5.3% of test tokens are out-of-vocabulary. Note that, according to Heaps' law, a similar behavior would be expected from corpora of any size (Herdan, 1960; Heaps, 1978).

[2] While Goldwater et al. (2011) acknowledge that their model could be used in various tasks of learning linguistic structure, they only present results in modeling morphology.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3721–3729
August 1–6, 2021.

based on the Pitman–Yor process (PYP; Pitman and Yor, 1997), and has the ability to model the power-law behavior of word tokens. The second, termed the **generator**, leverages a character-level neural language model to capture structural patterns in written words, e.g. graphotactics and morphology.

Critically, naïvely training a character-level neural model in either types (i.e. unique word forms) or tokens (i.e. word forms in their original frequencies) should lead to degenerate results. Models trained on natural corpora (i.e. token data) should excel in modeling the most common words of a language, but might poorly approximate the set of infrequent word forms which individuals dynamically produce (e.g. through compositional morphology). On the other hand, training models on the collection of unique word forms (i.e. type data) would give equal weight to typical and atypical productions, potentially leading to poor performance on the most frequent forms, which any individual would recognize as part of their language. In the two-stage model, as we will show, our generator is trained on a dataset interpolated between types and tokens—modeling the nuance between frequent and infrequent word forms better.

By testing our model on a set of languages with diverse morphological and phonological characteristics, we find that it is capable of modeling both frequent and infrequent words, thus producing a better estimate of the unigram distribution than a character-level LSTM. The empirical superiority of our two-stage model is shown in Fig. 1, where the surprisal (i.e. the negative log-probability, measured in nats here) of each token is plotted under four different models for Finnish. Our proposed two-stage model achieves a lower or similar surprisal to the baselines on tokens with all frequencies—with similar patterns arising in all analyzed languages.[3]

2 The Unigram Distribution

The unigram distribution is a probability distribution over the possible word forms in a language's lexicon. This probability takes the frequency of a token into account, assigning larger probabilities to word forms which are more likely to be encountered in a language's utterances, thus differing from word type distributions, such as in Pimentel et al. (2020). It is also not conditioned on a word's context, as it considers each word token as a stand-alone unit, as opposed to the task of language modeling, e.g. Mikolov et al. (2010).

2.1 Complex Vocabularies

The composition of spoken vocabularies is structured according to a host of factors. Stemming from articulatory biases, each language has a set of constraints on what sequences of speech sounds can be valid words in it; this is termed the **phonotactics** of a language. Languages also exhibit small but non-negligible biases in the regular match of forms and meanings (Dingemanse et al., 2015; Pimentel et al., 2019, 2021b). Additionally, expectations about morphology can constrain the production or processing of a given word as belonging to a particular word class (as shown for instance in Jabberwocky- and wug-type tasks, Berko 1958, Hall Maudslay and Cotterell 2021).

While individuals often have strong intuitions about these patterns, their judgments are typically gradient rather than categorical (Hayes and Wilson, 2008; Gorman, 2013). The effective set of words that naturally occur in linguistic productions are known to be extremely diverse in their composition. Models deployed to explain and predict typical word forms in a given language might fail at capturing these corners of the space of possible forms. If the goal is to produce ecologically valid models that could approximate actual cognitive processes, these atypical forms should be efficiently learned in addition to the most typical productions.

2.2 Imbalanced Frequencies

Zipf (1935) popularized the observation that the frequency of a word in a corpus is inversely proportional to its rank, approximately following a power-law distribution. As such, a small subset of the most common word types dominate the corpus. These extremely frequent words tend to be short in length and exceptionally archaic, in the sense that they preserve traces of previous phonotactic and phonological profiles that might have ceased to be productive. This is particularly relevant when we consider scenarios where substantial portions of the vocabulary might have been borrowed from different sources over time. English is a textbook example: Williams (1986) reports that French, Latin, Germanic and Greek account for

---

[3] As a final contribution of our work, the code used in this paper is available at `https://github.com/irenenikk/modelling-unigram`. We hope this will encourage future work in psycholinguistics to use the model to accurately investigate the effects of unigram probabilities in rare words.

29%, 29%, 26% and 6% of all words' origins in the vocabulary (plus a remaining 10% of diverse origin). The most frequent portion of the vocabulary preserves the most the original West Germanic forms, consisting largely of articles, prepositions, pronouns, and auxiliaries. Further, irregular inflections tend to be more common in these highly frequent words (Ackerman and Malouf, 2013; Cotterell et al., 2019). This observation might invite one to omit frequency information from training data, i.e. to use types, in order to balance out the role of the most frequent words.

On the other side of the frequency scale, however, any natural language data would have plenty of low-frequency words that reflect the open boundaries of the vocabulary. These might include nonce words (*blick*), expressive transformations of other words (*a loooooooooooong summer*), specialized terms (*onabotulinumtoxina*), and names, among others. In addition, genuine orthographic misproductions (*langague*) will be present to some degree.

Finally, acronyms (*HTML*) will be present in all frequency bands. These should be particularly problematic to model, since they do not necessarily follow the language's graphotactics to any degree. There are also frequent and infrequent loanwords with different degrees of adjustment to the grapho- and phonotactics of the rest of the vocabulary. For instance, it has been estimated that 96% and 21% of English speakers know the Afrikaans-originated words *aardvark* and *aardwolf*, respectively (Brysbaert et al., 2019).[4] These are the only written word forms in English with a non-negligible frequency that display two letter '*a*'s in word-initial position.

This whimsical nature of the vocabulary of a language makes modeling the unigram distribution challenging: Naïvely training a model to capture word forms at either the token or type level is likely to give disproportionate emphasis to phonotactically unrepresentative words. However, this is also why its modeling is a worthwhile task—it captures both frequent and rare productions, combining form probability with frequency information.

3 Modeling the Unigram Distribution

Our work neuralizes Goldwater et al.'s (2011) two-stage model and employs it to modeling the unigram distribution.[5] The first component, termed the **generator**, is a model used to produce a set of i.i.d. word forms $\{\boldsymbol{\ell}_k\}_{k=1}^{K}$. The second component is termed **adaptor**, and it assigns each instance in the training set to a cluster $\{z_n\}_{n=1}^{N}$. Under this model, each token in a dataset has a corresponding cluster z_n which defines the token's word form $\boldsymbol{w}_n = \boldsymbol{\ell}_{z_n}$. We note that both word forms $\boldsymbol{\ell}$ and clusters $\mathbf{z}$ are latent variables, and only tokens $\boldsymbol{w}$ are observed during training.

Generator. The generator is a model which produces word forms; we use a character-level LSTM here (Hochreiter and Schmidhuber, 1997), as in:[6]

$$\{\boldsymbol{\ell}_k\}_{k=1}^{K} \sim p_\phi(\boldsymbol{\ell}) = \mathrm{LSTM}(\boldsymbol{\ell}) \tag{1}$$

These word forms $\boldsymbol{\ell}_k$ are sampled i.i.d.—thus, the same word may be sampled more than once.

Adaptor. Each word form sampled from the generator corresponds to a cluster. The adaptor then assigns a frequency to each of the clusters according to a Pitman–Yor process:

$$p(z_n \mid \mathbf{z}_{<n}) \tag{2}$$
$$\propto \begin{cases} c_{<n}^{(z_n)} - a & 1 \le z_n \le K_{<n} \text{ (old cluster)} \\ a \cdot K_{<n} + b & z_n = K_{<n} + 1 \text{ (new cluster)} \end{cases}$$

where $0 \le a < 1$ and $0 \le b$ are hyperparameters of the PYP, $\mathbf{z}_{<n}$ are the previous cluster assignments, $K_{<n}$ is the current number of clusters with at least one token and $c_{<n}^{(z_n)}$ is the number of tokens previously assigned to cluster z_n. This adaptor, as a Pitman–Yor process, allows us to model the power-law distribution of word tokens.

Two-stage Model. Given a cluster assignment and the list of word forms, defining a token's form is deterministic: $p(\boldsymbol{w}_n \mid z_n, \boldsymbol{\ell}) = \mathbb{1}\{\boldsymbol{w}_n = \boldsymbol{\ell}_{z_n}\}$. Thus, our model factorizes a new token's probability into two terms:

$$p_{\text{model}}(\boldsymbol{w}) = \tag{3}$$
$$\underbrace{\frac{c_{\boldsymbol{w}} - \overbrace{n_{\boldsymbol{w}} \cdot a}^{\text{smoothing factor}}}{|\mathbf{z}| + b}}_{\text{smoothed 1-gram}} + \underbrace{\frac{(a \cdot K + b)}{|\mathbf{z}| + b}}_{\text{interpolation weight}} \cdot \underbrace{p_\phi(\boldsymbol{w})}_{\text{LSTM}}$$

where $c_{\boldsymbol{w}}$ is the number of occurrences of word form $\boldsymbol{w}$ in our training corpus and $n_{\boldsymbol{w}}$ is the number

---

[4] Aardvarks and aardwolves are African mammals.

[5] This same model was used in our contemporary work investigating lexicons' (non-)optimality (Pimentel et al., 2021a).

[6] See Pimentel et al. (2020) for more details on this graphotactics generative model.

of distinct clusters to which it has been assigned:

$$c_{\boldsymbol{w}} = \sum_{n=1}^{N} \mathbb{1}\{\boldsymbol{w} = \boldsymbol{\ell}_{z_n}\}, \tag{4}$$

$$n_{\boldsymbol{w}} = \sum_{k=1}^{K} \mathbb{1}\{\boldsymbol{w} = \boldsymbol{\ell}_{k}\} \tag{5}$$

In practice, the two-stage model acts as an interpolation between a smoothed 1-gram model, i.e. corpus frequencies, and an LSTM character model. Notably, this model learns per-word smoothing factors and its interpolation weight in an unsupervised manner through the PYP parameters' inference. The adaptor is fit using Gibbs sampling, and the generator is trained using a cross-entropy loss on the set of non-empty clusters produced by the adaptor. The generator is thus trained using a more balanced corpus where the proportion of the most frequent words is reduced; this can be seen as an interpolation between a type and a token dataset.[7]

4 Experiments.

Dataset. We use Wikipedia data and evaluate our model on the following languages: English, Finnish, Hebrew, Indonesian, Tamil, Turkish and Yoruba. These languages represent a typologically diverse set—with different levels of morphology, ranging from rich (e.g. Finnish) to poor (e.g. Yoruba), as well as distinct scripts and graphotactic patterns. In preprocessing, we first split the data into sentences and then into tokens using spaCy (Honnibal et al., 2020). We then sample 10^6 tokens as our training set for each language (except for Yoruba for which we had less data, see App. F for more details). From these, we build two distinct datasets: a **token dataset**, which corresponds to the list of word forms with their corpus frequency, and a **type dataset** containing the set of unique word forms in the data.

Evaluation. We measure the cross-entropy of our models on a held-out test set; this is the standard evaluation for language modeling. We approximate this cross-entropy using a sample mean estimate

$$\begin{aligned} \mathrm{H}(p) &\leq \mathrm{H}(p, p_{\text{model}}) \\ &\approx -\frac{1}{N}\sum_{n=1}^{N} \log p_{\text{model}}(\boldsymbol{w}_n) \end{aligned} \tag{6}$$

where we assume instances $\boldsymbol{w}_n$ are sampled from the true unigram distribution $p(\boldsymbol{w})$. Specifically, these token samples $\{\boldsymbol{w}_n\}_{n=1}^{N}$ take the form of the token dataset. The model with the lowest cross-entropy is the one that diverges the least from the true distribution.

	Evaluated Model			
Language	Type	Token	Two-stage	Generator
English	12.50	9.04	**8.34**	11.86
Finnish	15.07	12.94	**11.85**	14.16
Hebrew	12.62	10.71	**10.20**	11.76
Indonesian	13.28	10.33	**9.55**	11.89
Tamil	14.24	12.66	**11.76**	13.29
Turkish	14.00	11.77	**10.86**	12.95
Yoruba	11.13	10.04	**9.19**	9.88

Table 1: Cross-entropy on the **unigram distribution**.

Baseline Models. As neural networks yield state-of-the-art performance in language modeling tasks, we expect them to also do well with the unigram distribution. In fact, pseudo-text generated by LSTM-based language models reproduces Zipf's law to some extent (Takahashi and Tanaka-Ishii, 2017; Meister and Cotterell, 2021). Thus, we view state-of-the-art LSTM models as a strong baseline. We train a character-level LSTM language model (Pimentel et al., 2020) to directly approximate the unigram distribution by training it on the token dataset—modeling these tokens at the character level. As a second baseline, we train an LSTM on the type dataset. However, we expect this model to be outperformed by the token one in the unigram distribution task, as the information on word frequency is not available during its training. We do not use a word-level 1-gram model (i.e. the words' sample frequency) as a baseline here, since it results in an infinite cross-entropy for any test set containing oov words. We empirically compare four models: **two-stage**, **generator**, **token**, and **type**.

Modeling Tokens. Cross-entropy on the token test sets can be found in Tab. 1. These results show our two-stage model indeed creates a more accurate estimate of the unigram distribution, producing the smallest cross-entropy across all languages.

Frequent vs Infrequent Words. The weaknesses of the token and type models are evinced by Fig. 1. In line with our hypothesis, the token model achieves lower cross-entropy on the most common words, but fails to model the rare ones accurately.

---

[7] This model's training is detailed in App. E. For a detailed description of the adaptor see Goldwater et al. (2011).

	Evaluated Model				
Language	Type	Token	Two-stage	Generator	%
English	**18.23**	21.71	20.32	18.59	56%
Finnish	**19.76**	21.42	19.79	19.89	71%
Hebrew	**15.81**	17.76	17.10	16.30	56%
Indonesian	**18.52**	21.20	19.15	19.17	61%
Tamil	**18.77**	20.37	19.26	19.19	71%
Turkish	**18.36**	19.78	18.50	18.62	65%
Yoruba	15.44	17.69	**15.34**	16.28	67%

Table 2: Average surprisal for **singleton** types. Column % represents the ratio of singletons in the type test set.

	Evaluated Model			
Language	Type	Token	Two-stage	Generator
English	14.50	15.94	**13.24**	14.25
Finnish	15.19	14.89	**12.73**	14.80
Hebrew	13.36	13.80	**12.66**	13.20
Indonesian	14.72	15.50	**12.80**	14.64
Tamil	14.65	14.77	**12.95**	14.33
Turkish	14.73	14.41	**12.41**	14.33
Yoruba	11.13	11.97	**10.36**	11.16

Table 3: The average surprisal for **non-singleton** types.

The cross-entropy achieved by the type model does not change as much with word frequency, but is higher than the one achieved by the token model for most of the vocabulary. We also see that the two-stage model performs well across all word frequencies. Indeed, this model appears to behave similarly to the token model with frequent words, but obtains a lower cross-entropy on the rare ones, where the role of the generator in the estimated probability is emphasized. We suspect this is the reason behind the two-stage model's success.

The Long Tail. Fig. 1 also demonstrates that the entropy estimate for the rare words grows quickly and exhibits a large variance across models. This reflects the heterogeneous nature of the words that only appear a few times in a corpus. This part of the vocabulary is where the type model achieves the best results for all languages except Yoruba (see Tab. 2).[8] The fact that singletons (also known as *hapax legomena*), i.e. word forms which occur only once in the test set, form a large portion of the type dataset boosts the type model's performance on rare words. However, in the case of words appearing more than once (see Tab. 3) the two-stage model achieves the best results across languages. Furthermore, in these non-singleton words, the generator outperforms the type and token models in all languages except for Yoruba. This shows the utility of training the generator on an interpolation between types and tokens. In addition, we note that one may justifiably question whether properly modeling singletons is a desirable feature, since they are likely to contain unrepresentative word forms, such as typos, as discussed previously. Indeed, it appears that the two-stage model not only leads to tighter estimates of the unigram distribution, but also allows us to train a better graphotactics model; capable of modeling both frequent word forms as well as new productions.

Future Work. The results we present focus on analyzing the two-stage model. The generator, though, produces interesting results by itself, modeling non-singleton word forms better than the type and token models in most languages. This suggests that it might be better at modeling the graphotactics of a language than either of these baselines. Future work should explore if this indeed is the case.

5 Conclusion

In this work, we motivate the unigram distribution as an important task to both the psycholinguistics and natural language processing communities that has received too little attention. We present a two-stage model for estimating this distribution—a neuralization of Goldwater et al.'s (2011)—which is motivated by the complex makeup of vocabularies: This model defines the probability of a token by combining the probability of its appearance in the training corpus with the probability of its form. We have shown, through a cross-entropy evaluation, that our model outperforms naïve solutions and is capable of accurately modeling both frequent and infrequent words.

Acknowledgements

Damián E. Blasi acknowledges funding from the Branco Weiss Fellowship, administered by the ETH Zürich. Damián E. Blasi's research was also executed within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'.

Ethical Concerns

This paper highlights the importance of modeling the unigram distribution, and presents a model for

---

[8] We note that we used considerably less training data for Yoruba than for other languages.

the task. We do not foresee any reasons for ethical concerns, but we would like to note that the use of Wikipedia as a data source may introduce some bias into our experiments.

References

Farrell Ackerman and Robert Malouf. 2013. Morphological organization: The low conditional entropy conjecture. *Language*, 89(3):429–464.

R. Harald Baayen. 2002. *Word Frequency Distributions*, volume 18. Springer Science & Business Media.

R. Harald Baayen, Petar Milin, and Michael Ramscar. 2016. Frequency in lexical processing. *Aphasiology*, 30(11):1174–1220.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Jean Berko. 1958. The child's learning of English morphology. *Word*, 14(2-3):150–177.

Phil Blunsom, Trevor Cohn, Sharon Goldwater, and Mark Johnson. 2009. A note on the implementation of hierarchical Dirichlet processes. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 337–340, Suntec, Singapore. Association for Computational Linguistics.

Marc Brysbaert, Paweł Mandera, Samantha F. McCormick, and Emmanuel Keuleers. 2019. Word prevalence norms for 62,000 English lemmas. *Behavior Research Methods*, 51(2):467–479.

Ryan Cotterell, Christo Kirov, Mans Hulden, and Jason Eisner. 2019. On the complexity and typology of inflectional morphological systems. *Transactions of the Association for Computational Linguistics*, 7:327–342.

Ryan Cotterell, Sabrina J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 536–541, New Orleans, Louisiana. Association for Computational Linguistics.

Holger Diessel. 2017. Usage-based linguistics. In *Oxford Research Encyclopedia of Linguistics*. Oxford University Press.

Mark Dingemanse, Damián E. Blasi, Gary Lupyan, Morten H. Christiansen, and Padraic Monaghan. 2015. Arbitrariness, iconicity, and systematicity in language. *Trends in Cognitive Sciences*, 19(10):603–615.

Dagmar Divjak. 2019. *Frequency in Language: Memory, Attention and Learning*. Cambridge University Press.

Richard Futrell, Adam Albright, Peter Graff, and Timothy O'Donnell. 2017. A generative model of phonotactics. *Transactions of the Association for Computational Linguistics*, 5(0):73–86.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2011. Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12(68):2335–2382.

Kyle Gorman. 2013. *Generative Phonotactics*. University of Pennsylvania.

Rowan Hall Maudslay and Ryan Cotterell. 2021. Do syntactic probes probe syntax? experiments with jabberwocky probing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 124–131, Online. Association for Computational Linguistics.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Harold Stanley Heaps. 1978. *Information Retrieval, Computational and Theoretical Aspects*. Academic Press.

Gustav Herdan. 1960. *Type–Token Mathematics: A Textbook of Mathematical Linguistics*, volume 4. Mouton.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alexander Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. In *arXiv preprint arXiv:1610.10099*.

Clara Meister and Ryan Cotterell. 2021. Language model evaluation beyond perplexity. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the International Speech Communication Association*, pages 1045–1048.

Radford M. Neal. 1993. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, University of Toronto.

Tiago Pimentel, Arya D. McCarthy, Damian Blasi, Brian Roark, and Ryan Cotterell. 2019. Meaning to form: Measuring systematicity as information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1751–1764, Florence, Italy. Association for Computational Linguistics.

Tiago Pimentel, Irene Nikkarinen, Kyle Mahowald, Ryan Cotterell, and Damián Blasi. 2021a. How (non-)optimal is the lexicon? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4426–4438, Online. Association for Computational Linguistics.

Tiago Pimentel, Brian Roark, and Ryan Cotterell. 2020. Phonotactic complexity and its trade-offs. *Transactions of the Association for Computational Linguistics*, 8:1–18.

Tiago Pimentel, Brian Roark, Søren Wichmann, Ryan Cotterell, and Damián Blasi. 2021b. Finding concept-specific biases in form–meaning associations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4416–4425, Online. Association for Computational Linguistics.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson–Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Shuntaro Takahashi and Kumiko Tanaka-Ishii. 2017. Do neural nets learn statistical laws behind natural language? *PLOS ONE*, 12(12):1–17.

Greg C.G. Wei and Martin A. Tanner. 1990. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704.

Joseph M. Williams. 1986. *Origins of the English language*. Simon and Schuster.

George Kingsley Zipf. 1935. *The Psycho-biology of Language: An Introduction to Dynamic Philology*. Houghton Mifflin.

A Tuned hyperparameres for the two-stage model

Language	a	b
English	0.33	3,000
Finnish	0.36	90,000
Hebrew	0.40	55,000
Indonesian	0.48	180,000
Tamil	0.70	37,000
Turkish	0.33	95,000
Yoruba	0.08	156,000

Table 4: The optimized values of a and b for the analyzed languages.

B Hyperparameter Search

The same hyperparameters are used for both our baseline LSTMs and the generator. We use 3 layers, where embedding size is 128, hidden size is 512, and dropout probability is 0.33. Training the two-stage model takes a considerable amount of time (see Tab. 5). We are thus not capable of doing exhaustive hyperparameter tuning. Random search (Bergstra and Bengio, 2012) is used in tuning the values for a and b, where we run five training procedures considering ranges $a \in [0, 1)$, and $b \in [100, 200{,}000)$. We tune the hyperparameters for each language by minimizing the model's cross-entropy on the development set, training them on a subset of the training data with only $100{,}000$ tokens. The found optimal values of a and b are rounded to two decimal places and the thousands respectively. Our two-stage model is trained for five iterations of expectation–maximization.

C Training time with the two-stage model for each language

Language	Minutes
English	164
Finnish	170
Hebrew	175
Indonesian	173
Tamil	166
Turkish	174
Yoruba	56

Table 5: The training times for the two-stage model in each language. These times were obtained with a single NVIDIA Tesla P100 GPU.

D The development set cross-entropies on the unigram distribution

	Evaluated Model			
Language	Type	Token	Two-stage	Generator
English	12.50	9.04	**8.34**	9.18
Finnish	15.08	12.94	**11.88**	13.05
Hebrew	12.62	10.71	**10.20**	10.78
Indonesian	13.27	10.30	**9.53**	10.42
Tamil	14.22	12.65	**11.76**	12.75
Turkish	13.99	11.74	**10.83**	12.92
Yoruba	11.10	10.00	**9.13**	9.83

Table 6: Development set cross-entropy for the baseline models as well as our two-stage model evaluated on the **unigram distribution**.

E Inference

Unfortunately, there is no closed form solution for inferring the parameters of our two-stage model. In order to obtain a sample of cluster assignments and train the generator to match their labels, we estimate the parameters of both the generator and the adaptor concurrently, freezing one's parameters while training the other. We use a regime corresponding to the Monte Carlo Expectation-maximization (EM) algorithm to train the model (Wei and Tanner, 1990), which can be found in Algorithm 1. In the E-step, the function GIBBSSAMPLER returns the cluster assignments $\mathbf{z}$ and the dampened word dataset $\boldsymbol{\ell}$ obtained via Gibbs sampling from the PYP. We then use this dampened dataset to train the generator in the M-step.

Algorithm 1 Training the two-stage model

```
for $i$ in RANGE(# Epochs) do
    // E-Step
    $\mathbf{z}, \boldsymbol{\ell} \sim$ GIBBSSAMPLER$(a, b, p_\phi, \{\boldsymbol{w}_n\}_1^N)$
    // M-Step
    for $t = 1$ up to $T$ do
        $\boldsymbol{\phi} \leftarrow \eta_t \sum_{k=1}^{|\boldsymbol{\ell}|} \nabla_{\boldsymbol{\phi}} \log p_\phi(\boldsymbol{\ell}_k \mid \boldsymbol{\phi})$
    end for
end for
```

E.1 Gibbs Sampler For Cluster Assignments

The Pitman–Yor process does not have a well-defined posterior probability. Nonetheless, we can use Gibbs sampling to obtain a sample from this posterior distribution over cluster assignments defined by the two-stage model.[9] We build our sampler after the morphological sampler presented by Goldwater et al. (2011).

Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method which approximates the posterior of a multivariate distribution. It iteratively samples from the conditional distribution of a variable, given the values of the other dimensions (Neal, 1993). We use the conditional distribution defined in eq. (7) (presented in Fig. 2) in the Gibbs sampler where we know the word form $\boldsymbol{w}_n$ of token n—since it is observable in the corpus— and where the values for all other cluster assignments are fixed. Note that, according to eq. (7), we only assign word tokens to clusters with the same form or create a new cluster—and when a new one is created, its word form is assigned to $\boldsymbol{w}_n$. As such, each cluster contains a single shared word form. For each adaptor training iteration, we run the Gibbs sampler for six epochs, and choose the cluster assignments that have the best performance on a development set. Furthermore, we persist the adaptor state across iterations, warm starting the Gibbs sampler with the cluster assignments of the previous iteration.

E.2 Training the generator

In order to train the generator on word form data with more balanced frequency distributions, a new training set is dynamically created. In this dataset, each token appears as many times as it has been assigned as a cluster label, noted with $\boldsymbol{\ell}$ in Algorithm 1.[10] A regime similar to using the inverse-power transformed counts of the tokens in the corpus (Goldwater et al., 2011).

This new training set allows us to train the generator in an interpolation between a purely type- or token-based dataset; this interpolation can be controlled through its parameters a and b. Setting the values of a and b to zero will cause the model to favor existing clusters to creating new ones, resulting in assigning every token with the same form to a single cluster. In this case, the generator parameters would be estimated using the equivalent of a *type* corpus. Similarly, when a approaches one, or in the limit of $b \to \infty$, less tokens will be assigned per cluster and the number of single token clusters grows. This is effectively equivalent to training the generator using *tokens*. Consequently, non-extreme

---

[9] This is possible due to the exchangeability of the cluster assignments.

[10] We hotstart the generator model by training it on a type-level dataset before the first adaptor training iteration.

$$p(z_n \mid \mathbf{z}_{<n}, \boldsymbol{w}_n) \propto p(z_n, \boldsymbol{w}_n \mid \mathbf{z}_{<n}) \propto \begin{cases} (c_{<n}^{(z_n)} - a) \cdot \mathbb{1}\{\boldsymbol{w}_n = \boldsymbol{\ell}_{z_n}\} & 1 \leq z_n \leq K_{<n} \\ (a \cdot K_{<n} + b) \cdot p_\phi(\boldsymbol{w}_n) & z_n = K_{<n} + 1 \end{cases} \tag{7}$$

Figure 2: The probability of assigning token $\boldsymbol{w}_n$ to cluster z_n in the two-stage model given all other cluster assignments $\mathbf{z}_{<n}$.

value of a and b are a middle ground.

We train the character-level LSTM used as our generator with stochastic gradient descent using a cross-entropy loss function. This model is trained with early stopping; it is evaluated every 200 batches, and training stops when the development set loss has increased for 5 consecutive epochs.

E.3 Training Optimizations

The naïve implementation of the Gibbs sampler for table assignments quickly becomes computationally expensive in practice. Consequently, we use the optimized algorithm designed by Blunsom et al. (2009) for the hierarchical Dirichlet process in our implementation, extending it to Pitman–Yor processes with the additional parameter a.

F Dataset

As noted in the main text, we use Wikipedia data in our experiments. The amount of sentences used in our experiments is capped to one billion after shuffling them. Additionally, we define an upper bound to the amount of tokens used in each experiment. In case the training data exceed this limit, we construct a corpus by re-sampling (with replacement) the desired number of tokens using the corpus frequencies calculated from the original training corpus. The number of types and tokens used in training and evaluation are presented in Tab. 7. Noise in the Wikipedia data is somewhat reduced by hand-defining an alphabet for each language, and removing any sentence which includes words with invalid graphemes in it.[11]

	Train		Test	
	# Types	# Tokens	# Types	# Tokens
English	76,589	10^6	67,148	759,412
Finnish	208,498	10^6	108,020	332,220
Hebrew	131,288	10^6	105,550	619,685
Indonesian	102,739	10^6	72,250	507,848
Tamil	206,512	10^6	116,165	388,257
Turkish	154,185	10^6	85,074	331,072
Yoruba	97,097	329,093	12,117	41,055

Table 7: The amount of tokens and types used in both training and testing for the analyzed languages.

---

[11] We define the alphabets using the languages' Wikipedia articles and the following website: `https://r12a.github.io/app-charuse/`.

On the Lack of Robust Interpretability of Neural Text Classifiers

Muhammad Bilal Zafar
Amazon
zafamuh@amazon.com

Michele Donini
Amazon
donini@amazon.com

Dylan Slack[*]
University of California, Irvine
dslack@uci.edu

Cédric Archambeau
Amazon
cedrica@amazon.com

Sanjiv Das
Amazon & Santa Clara University
sanjivda@amazon.com

Krishnaram Kenthapadi
Amazon
kenthk@amazon.com

Abstract

With the ever-increasing complexity of neural language models, practitioners have turned to methods for understanding the predictions of these models. One of the most well-adopted approaches for model interpretability is *feature-based interpretability*, i.e., ranking the features in terms of their impact on model predictions. Several prior studies have focused on assessing the fidelity of feature-based interpretability methods, i.e., measuring the impact of dropping the top-ranked features on the model output. However, relatively little work has been conducted on quantifying the robustness of interpretations. In this work, we assess the robustness of interpretations of neural text classifiers, specifically, those based on pretrained Transformer encoders, using two randomization tests. The first compares the interpretations of two models that are identical except for their initializations. The second measures whether the interpretations differ between a model with trained parameters and a model with random parameters. Both tests show surprising deviations from expected behavior, raising questions about the extent of insights that practitioners may draw from interpretations.

1 Introduction

In recent years, large scale language models like BERT and RoBERTa have helped achieve new state-of-the-art performance on a variety of NLP tasks (Devlin et al., 2019; Liu et al., 2019). While relying on vast amounts of training data and model capacity has helped increase their accuracy, the reasoning of these models is often hard to comprehend. To this end, several techniques have been proposed to interpret the model predictions.

Perhaps the most widely-adopted class of interpretability approaches is that of *feature-based interpretability* where the goal is to assign an importance score to each of the input features. These scores are also called *feature attributions*. Several methods in this class (e.g., SHAP (Lundberg and Lee, 2017), Integrated Gradients (Sundararajan et al., 2017)) possess desirable theoretical properties making them attractive candidates for interpretability.

Benchmarking analyses often show that these methods possess *high fidelity*, i.e., removing features marked important by the interpretability method from the input indeed leads to significant change in the model output as expected (Atanasova et al., 2020; Lundberg and Lee, 2017).

However, relatively few investigations have been carried out to understand the *robustness* of feature attributions. To explore the robustness, we conduct two tests based on randomization:

Different Initializations Test: This test operationalizes the *implementation invariance* property of Sundararajan et al. (2017). Given an input, it compares the feature attributions between two models that are identical in every aspect—that is, trained with same architecture, with same data, and same learning schedule—except for their randomly chosen initial parameters. If the predictions generated by these two models are also identical, one would also expect the feature attributions to be the same for such *functionally equivalent* models. If the attributions in two cases are not the same, two users examining the *same input* may deem the same features to have *different importance* based on the model that they are consulting.

Untrained Model Test: This test is similar to the test of Adebayo et al. (2018). Given an input, it compares the feature attributions generated on a fully trained model with those on a randomly initialized untrained model. The test evaluates whether feature attributions on a fully trained model differ from the feature attributions computed on an

---

[*] Work done during internship at Amazon.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3730–3740
August 1–6, 2021.

untrained model as one would expect.

We conduct the two tests on a variety of text classification datasets. We quantify the feature attribution similarity using *interpretation infidelity* (Arras et al., 2016) and *Jaccard similarity* (Tanimoto, 1958). The results suggest that: (i) Interpretability methods fail the different initializations test. In other words, two functionally equivalent models lead to different ranking of feature attributions; (ii) Interpretability methods fail the untrained model test, i.e., the fidelity of the interpretability method on an untrained model is better than that of random feature attributions.

These findings may have important implications for how the prediction interpretations are shown to the users of the model, and raise interesting questions about reliance on these interpretations. For instance, if two functionally equivalent models generate different interpretations, to what extent can a user act upon them, e.g., investing in a financial product or not. We discuss these implications and potential reasons for this behavior in §4.

Related work. Model interpretability has different aspects: local (e.g. Lundberg and Lee, 2017) vs. global (e.g. Tan et al., 2018), feature-based (e.g. Lundberg and Lee, 2017) vs. concept-based (e.g. Kim et al., 2018) vs. hidden representation-based (Li et al., 2016). See Gilpin et al. (2018); Guidotti et al. (2018) for an overview. In this paper, we focus on feature-based interpretability, which is a well-studied and commonly used form (Bhatt et al., 2020; Tjoa and Guan, 2020). Specifically, this class consists of a relatively large number of methods, of which some have been shown to satisfy desirable theoretical properties (e.g., SHAP (Lundberg and Lee, 2017), Integrated Gradients (Sundararajan et al., 2017), and DeepLIFT (Shrikumar et al., 2017)).

There are several important aspects of interpretation robustness. Some prior studies have considered interpretability in the context of *adversarial robustness* where the goal often is to actively fool the model to generate misleading feature attributions. See for instance Anders et al. (2020); Dombrowski et al. (2019); Ghorbani et al. (2019); Slack et al. (2020). In this work, rather than focusing on targeted changes in the input or the model, we explore robustness of feature attribution methods to various kinds of randomizations.

Several prior works have focused on quantifying quality of interpretations. See for instance, Adebayo et al. (2020); Alvarez-Melis and Jaakkola (2018); Chalasani et al. (2020); Chen et al. (2019); Hooker et al. (2019); Lakkaraju et al. (2020); Meng et al. (2018); Ribeiro et al. (2016); Tomsett et al. (2020); Yang and Kim (2019). Closest to ours is the work of Adebayo et al. (2018), which is based on checking the saliency maps of randomly initialized image classification models. However, in contrast to Adebayo et al., we consider text classification. Moreover, while the analysis of Adebayo et al. is largely based on visual inspection, we extend it by considering automatically quantifiable measures. We also extend the analysis to non-gradient based methods (SHAP).

2 Setup

We describe the datasets, models, and interpretability methods considered in our analysis.

Datasets. We consider four different datasets covering a range of document lengths and number of label classes. The datasets are: (i) **FPB:** The Financial Phrase Bank dataset (Malo et al., 2014) where the task is to classify news headlines into one of three sentiment classes, namely, positive, negative, and neutral. (ii) **SST2:** The Stanford Sentiment Treebank 2 dataset (Socher et al., 2013). The task is to classify single sentences extracted from movie reviews into positive or negative sentiment classes. (iii) **IMDB:** The IMDB movie reviews dataset (Maas et al., 2011). The task is to classify movie reviews into positive or negative sentiment classes. (iv) **Bios:** The Bios dataset of De-Arteaga et al. (2019). The task is to classify the profession of a person from their biography. Table 5 in Appendix A shows detailed dataset statistics.

2.1 Models

We consider four pretrained Transformer encoders: BERT (**BT**), RoBERTa (**RB**), DistilBERT (**dBT**), and DistilRoBERTa (**dRB**). The encoder is followed by a pooling layer to combine individual token embeddings, and a classification head. Appendix B.1 describes the detailed architecture, training and hyperparameter tuning details. After training and hyperparameter tuning, the best model is selected based on validation accuracy and is referred to as `Init#1`.

Different Initializations Test. Recall from §1 that this test involves comparing two identical models trained from different initializations. The second model, henceforth referred to as `Init#2`, is trained using the same architecture, hyperparameters and training strategy as `Init#1`, but starting from a different set of initial parameters. Since we start from pretrained encoders, the encoder parameters are not intialized. For each layer in the rest of the model, a set of initial parameters different from those in `Init#1` is obtained by calling the parameter initialization method of choice for this layer—*He initialization* (He et al., 2015) in this case—but with a different random seed.

Untrained Model Test. Recall that this test involves comparing the trained model (`Init#1`) with a randomly initialized untrained model, henceforth called `Untrained`. To obtain `Untrained`, we start from the `Init#1`, and randomly initialize the fully connected layers attached on top of the Transformer encoders (the encoder weights are not randomized). The initialization strategy is the same as in Different Initializations Test.

2.2 Interpretability methods

We consider a mix of gradient-based and model agnostic methods. Specifically: Vanilla Saliency (**VN**) of Simonyan et al. (2014), SmoothGrad (**SG**) of Smilkov et al. (2017), Integrated Gradients (**IG**) of Sundararajan et al. (2017), and KernelSHAP (**SHP**) of Lundberg and Lee (2017). We also include random feature attribution (**RND**) which corresponds to each feature being assigned an attribution from the uniform distribution, $\mathcal{U}(0, 1)$. Appendix B.2 provides details about the parameters chosen for the interpretability methods.

Given an input text document, we tokenize the text using the tokenizer of the corresponding encoder. Finally, for each input feature (that is, token), the feature attribution of the gradient-based methods is a vector of the same length as the token input embedding. For scalarizing these vector scores, we use the L2-norm strategy of Arras et al. (2016) and the Input $\odot$ Gradient strategy of Ding et al. (2019).

2.3 Interpretability Metrics

To compare the feature attributions by various interpretability methods, we use the following metrics.

(In)Fidelity: Given an input text which has been split into L tokens, $\mathbf{t} = [t_1, \ldots, t_L]$, get the vector $\mathbf{\Psi}(\mathbf{t}) = [\psi(t_1), \ldots, \psi(t_L)]$ of feature attributions of the corresponding tokens using the interpretability method to be evaluated. Drop the features from $\mathbf{t}$ in the decreasing order of attribution score until the model prediction changes from the original prediction (with all tokens present). Infidelity is defined as the % of features that need to be dropped until the prediction changes. A better interpretability method is expected to need a lower fraction of features to be dropped until the prediction change. We simulate feature dropping by replacing the corresponding input token with the model's unknown vocabulary token.

	BT	RB	dBT	dRB
FPB	0.83	0.85	0.82	0.84
SST2	0.87	0.91	0.88	0.90
IMDB	0.92	0.95	0.93	0.94
Bios	0.86	0.86	0.86	0.86

Table 1: Test accuracy with `Init#1`. For any given dataset, all encoders lead to a similar accuracy.

The infidelity metric has appeared in many closely related forms in a number of studies evaluating model interpretability (Arras et al., 2016; Atanasova et al., 2020; DeYoung et al., 2020; Fong et al., 2019; Lundberg and Lee, 2017; Samek et al., 2017). All of these forms operate by iteratively hiding features in the order of their importance and measuring the change in the model output, e.g., in predicted class probability, or the predicted label itself. We chose number of tokens to prediction change, which is closely aligned with (Arras et al., 2016), due to its simplicity as compared to more involved metrics relying on AUC-style measures (Atanasova et al., 2020; Samek et al., 2017).

Jaccard Similarity: It is common to show top few most important features to users as model interpretations. See for instance, Ribeiro et al. (2016) and Schmidt and Biessmann (2019). In order to measure the similarity between feature attributions generated by different methods, we use the Jaccard@K% metric. Given an input $\mathbf{t}$, let $\mathbf{s}_i$ be the set of top-K% tokens, when the tokens are ranked based on their importance as specified by an attribution output $\mathbf{\Psi}_i$. Then, given two attribution outputs $\mathbf{\Psi}_i$ and $\mathbf{\Psi}_j$, Jaccard@K% measures the similarity between them as: $J(i, j) = \frac{|\mathbf{s}_i \cap \mathbf{s}_j|}{|\mathbf{s}_i \cup \mathbf{s}_j|}$. If the top-K% tokens by the two attributions $\mathbf{\Psi}_i$ and $\mathbf{\Psi}_j$ are the same, then $J(i, j) = 1$. In case of no overlap in the top-K% tokens, $J(i, j) = 0$.

Appendix D shows some examples of Jaccard@K% computation.

	FPB				SST2				IMDB				Bios			
	BT	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**
VN	68	64	61	59	50	54	53	54	46	53	52	50	28	22	27	25
SG	68	63	61	58	46	50	53	51	45	52	52	48	27	23	27	23
IG	68	62	61	59	38	48	50	49	39	47	46	47	24	20	17	22
SHP	60	54	52	43	22	25	30	27	10	19	11	13	15	15	15	14
RND	72	71	68	70	61	67	66	69	58	58	63	66	51	51	49	56

Table 2: Mean infidelity of different interpretability methods for `Init#1` (shown as %). Lower values are better.

3 Results

Table 1 shows the test set accuracy of `Init#1` model with different encoders on all the datasets. For all the datasets, different encoders lead to a very similar accuracy. Tables 7 in Appendix C.1 shows the prediction accuracy for `Untrained`.

Infidelity. Table 2 shows the infidelity of different interpretability methods on the best performing models (`Init#1`). The table shows that: (i) As expected, the infidelity of all interpretability methods is better than RND; (ii) SHP provides the best performance, followed by IG; (iii) For a given dataset, even though different encoders have very similar accuracy (Table 1), the infidelity of the same interpretability method for different encoders can vary widely, e.g., SHP on IMDB; (iv) There is no particularly discernable correlation between the models and their infidelity, for instance, with FPB dataset, the distilled Transformers provides same or lower infidelity as compared to the original counterparts (BERT, RoBERTa), whereas the trend is reversed for SST2 data.

Moreover, gradient-based methods in Table 2 use the L2-norm reduction (§2.2). Table 8 in Appendix C.2 shows that in most cases, the performance is much worse when using the Input ⊙ Gradient reduction. Hence, for the rest of the analysis, we only use L2-norm reduction.

Different Initializations Test. Comparing `Init#1` and `Init#2` in Table 6 in Appendix C.1—two otherwise identical models with only difference being the random initial parameters, shows that a vast majority of predictions are common between the two models: meaning that the *two models are almost functionally equivalent*.

We now compare the similarity in feature attributions of two functionally equivalent models. Since the feature attributions are generated w.r.t. the predicted class, our similarity analysis is limited to samples where both models generate the same prediction. Figure 1 shows Jaccard@25% when comparing the feature attributions of `Init#1` vs. `Init#2`, and `Init#1` vs. `Untrained`. The figure shows that (i) for the functionally equivalent models `Init#1` vs. `Init#2`, Jaccard@25% is far from the ideal value of 1.0 — in fact, for IG, the value drops to almost 0.5; (ii) When comparing the top-ranked feature attributions of `Init#1` vs. `Init#2`, and `Init#1` vs. `Untrained`, the former should show a much bigger overlap than latter, but this is not the case, except for SHP.

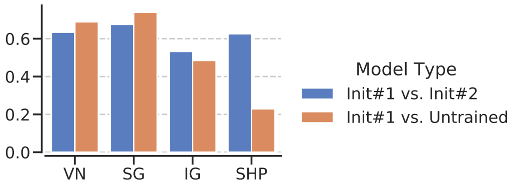

Figure 1: [**BT** on SST2 data] Comparing the mean Jaccard@25 between different model types (`Init#1` vs. `Init#2` and `Init#1` vs. `Untrained`).

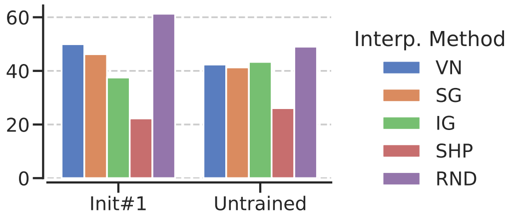

Figure 2: [**BT** on SST2 data] Mean infidelity of interpretability methods on `Init#1` and `Untrained`.

Tables 3 and 4 show the results for rest of the cases, revealing similar insights. Specifically, the Jaccard@25% between `Init#1` and `Untrained` for VN averaged over all 16 cases (four datasets, four models in Table 3) is 68, whereas the same is 69 when comparing `Init#1` and `Untrained` (Table 4). The same comparison yields 67 vs. 64 for SG, 56 vs. 50 for IG and 65 vs. 29 for SHP. Moving beyond averages, we also counted for each of the cases in Table 3, the number of times Jaccard@25% between `Init#1` and `Init#2` is within 10 units (Jaccard@25% ranges from 0-100) of the Jaccard@25% between `Init#1` and `Untrained` for the corresponding

	FPB				SST2				IMDB				Bios			
	BT	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**
VN	65	52	77	67	63	53	73	63	72	47	78	72	76	70	81	75
SG	53	66	70	62	68	63	70	66	57	45	76	75	75	70	81	78
IG	46	35	68	51	53	37	71	50	65	36	82	68	49	51	73	68
SHP	57	55	64	55	63	65	66	62	68	50	75	67	72	71	75	69

Table 3: Jaccard@25% between the feature attributions for `Init#1` vs. `Init#2` models (shown as %).

	FPB				SST2				IMDB				Bios			
	BT	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**
VN	65	60	75	71	69	63	73	69	64	79	71	66	71	68	70	74
SG	47	70	67	61	74	72	70	75	44	53	63	52	66	71	60	75
IG	41	40	55	49	49	45	60	49	37	75	63	51	40	41	55	47
SHP	25	30	19	29	23	22	14	33	15	85	23	18	32	45	24	27

Table 4: Jaccard@25% between the feature attributions for `Init#1` vs. `Untrained` models (shown as %).

pair in Table 4. The numbers are 14/16 for VN, 12/16 for SG, 7/16 for IG and 0/16 for SHP.

We selected K=25 as it corresponds to comparing the 25% most important features between the two models. Selecting K=10 (comparing 10% most important features) leads to similar outcomes: 13/16 for VN, 11/16 for SG, 6/16 for IG and 0/16 for SHP.

In other words, *the attribution overlap between two functionally equivalent models can be similar to that between a trained vs. an untrained model.*

Untrained Model Test. Figure 2 shows the infidelity of different methods on SST2 dataset with a **BT** model. We note that the performance of RND is better (lower infidelity) for the untrained model (`Untrained`) than for the trained model (`Init#1`). Furthermore, even for the untrained model (`Untrained`), all interpretability methods have a better fidelity than RND. In fact, for SHP, the infidelity is almost half of RND. Table 9 in Appendix C shows a similar pattern for the rest of the datasets and models.

In short, *even for an untrained model, the interpretability methods lead to better-than-random-attribution fidelity.* The insights highlight the need for baselining the fidelity metric with untrained models before using it as an evaluation measure.

4 Conclusion & Future Work

We carried out two tests to assess robustness of several popular interpretability methods on Transformer-based text classifiers. The results show that both gradient-based and model-agnostic methods can fail the tests.

These observations raise several **interesting questions**: if the fidelity of the interpretations is reasonably high on even an untrained model, to what extent does the interpretability method reflect the data-specific vs. data-independent behavior of the model? If two functionally equivalent models lead to different feature attributions, to what extent can the practitioners rely upon these interpretations to make consequential decisions?

One cause of the non-robust behavior could be the redundancy in text where several input tokens may provide evidence for the same class (e.g., several words in input review praising the movie). Another reason, related to the first, could be the pathologies of neural models where dropping most of the input features could still lead to highly confident predictions (Feng et al., 2018).[1] Dropping individual features can also lead to out-of-distribution samples, further limiting the effectiveness of methods and metrics that rely on simulating feature removal (Kumar et al., 2020; Sundararajan and Najmi, 2020). Systematically analyzing the root causes, and designing interpretability measures that are cognizant of the specific characteristics of text data—preferably with human involvement (Chang et al., 2009; Doshi-Velez and Kim, 2017; Hase and Bansal, 2020; Nguyen, 2018; Poursabzi-Sangdeh et al., 2021; Schmidt and Biessmann, 2019)—is a promising research direction. Similarly, extending the Untrained Model Test to study the effect of randomization of pre-trained embedding models on interpretability is another direction for exploration.

---

[1] Note, however, that the insights of Feng et al. relate to *dropping least important tokens first*, whereas when computing infidelity, one drops the most important first.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity Checks for Saliency Maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9525–9536, Red Hook, NY, USA. Curran Associates Inc.

Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. 2020. Debugging Tests for Model Explanations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

David Alvarez-Melis and Tommi S. Jaakkola. 2018. Towards Robust Interpretability with Self-explaining Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 7786–7795, Montréal, Canada. Curran Associates Inc.

Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. 2020. Fairwashing Explanations with Off-manifold Detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining Predictions of Non-Linear Classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A Diagnostic Study of Explainability Techniques for Text Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. 2020. Explainable Machine Learning in Deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pages 648–657, Barcelona, Spain. Association for Computing Machinery.

Prasad Chalasani, Jiefeng Chen, Amrita Roy Chowdhury, Xi Wu, and Somesh Jha. 2020. Concise Explanations of Neural Networks using Adversarial Training. In *International Conference on Machine Learning*, pages 1383–1391. PMLR.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Advances in Neural Information Processing Systems*, volume 22, pages 288–296.

Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. 2019. Robust Attribution Regularization. In *Advances in Neural Information Processing Systems*, volume 32.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in Bios: A Case Study of Semantic Representation Bias in a High-Stakes Setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 120–128, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A Benchmark to Evaluate Rationalized NLP Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Shuoyang Ding, Hainan Xu, and Philipp Koehn. 2019. Saliency-driven Word Alignment Interpretation for Neural Machine Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. 2019. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 32:13589–13600.

Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv:1702.08608 [cs, stat]*. ArXiv: 1702.08608.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of Neural Models Make Interpretations Difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. 2019. Understanding Deep Networks via Extremal Perturbations and Smooth Masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2950–2958.

Amirata Ghorbani, Abubakar Abid, and James Zou. 2019. Interpretation of Neural Networks Is Fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3681–3688.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. 2018. Explaining Explanations: An Overview of Interpretability of Machine Learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5):93:1–93:42.

Peter Hase and Mohit Bansal. 2020. Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, Online. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1026–1034, USA. IEEE Computer Society.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019. A Benchmark for Interpretability Methods in Deep Neural Networks. *Advances in Neural Information Processing Systems*, 32:9737–9748.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *International Conference on Machine Learning*, pages 2668–2677. PMLR.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A Unified and Generic Model Interpretability Library for PyTorch. *arXiv:2009.07896 [cs, stat]*. ArXiv: 2009.07896.

I. Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. 2020. Problems with Shapley-value-based explanations as feature importance measures. In *International Conference on Machine Learning*, pages 5491–5500. PMLR.

Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. 2020. Robust and Stable Black Box Explanations. In *International Conference on Machine Learning*, pages 5628–5638. PMLR.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 2016 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, USA. Association for Computational Linguistics.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796.

Qingjie Meng, Christian Baumgartner, Matthew Sinclair, James Housden, Martin Rajchl, Alberto Gomez, Benjamin Hou, Nicolas Toussaint, Veronika Zimmer, Jeremy Tan, Jacqueline Matthew, Daniel Rueckert, Julia Schnabel, and Bernhard Kainz. 2018. Automatic Shadow Detection in 2D Ultrasound Images. In *Data Driven Treatment Response Assessment and Preterm, Perinatal, and Paediatric Image Analysis*, Lecture Notes in Computer Science, pages 66–75, Cham. Springer International Publishing.

Dong Nguyen. 2018. Comparing Automatic and Human Evaluation of Local Explanations for Text Classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32:8026–8037.

Forough Poursabzi-Sangdeh, Daniel G. Goldstein, Jake M. Hofman, Jennifer Wortman Vaughan, and

Hanna Wallach. 2021. Manipulating and Measuring Model Interpretability. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ArXiv: 1802.07810.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*.

W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. 2017. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673.

Philipp Schmidt and Felix Biessmann. 2019. Quantifying Interpretability and Trust in Machine Learning Systems. In *AAAI-19 Workshop on Network Interpretability for Deep Learning*.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *2nd International Conference on Learning Representations (Workshop Track)*.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pages 180–186, New York, NY, USA. Association for Computing Machinery.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. SmoothGrad: Removing Noise by Adding Noise. In *ICML Workshop on Visualization for Deep Learning*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification? In *Chinese Computational Linguistics*, Lecture Notes in Computer Science, pages 194–206, Cham. Springer International Publishing.

Mukund Sundararajan and Amir Najmi. 2020. The Many Shapley Values for Model Explanation. In *International Conference on Machine Learning*, pages 9269–9278. PMLR.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*.

Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. 2018. Learning Global Additive Explanations for Neural Nets Using Model Distillation. *arXiv:1801.08640 [cs, stat]*. ArXiv: 1801.08640.

T. T. Tanimoto. 1958. *An elementary mathematical theory of classification and prediction*. New York: International Business Machines Corporation.

Erico Tjoa and Cuntai Guan. 2020. A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21. ArXiv: 1907.07374.

Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. 2020. Sanity Checks for Saliency Metrics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6021–6029.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models. In *Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations)*, pages 7–12.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mengjiao Yang and Been Kim. 2019. Benchmarking Attribution Methods with Relative Feature Importance. *arXiv:1907.09701 [cs, stat]*. ArXiv: 1907.09701.

Appendix A Datasets

	N	K	D	Δ_K
FPB	4,846	3	22±10	47
SST2	9,613	2	19±9	3
IMDB	50,000	2	227±168	0
Bios	397,907	28	61±28	30

Table 5: Dataset details. The columns are: # of samples (N), # of classes (K), average ± standard deviation # of words per document (D), and Class Imbalance (Δ_K). Class imbalance is measured as % prevalence of the most prevalent minus the least prevalent class.

Appendix B Reproducibility

B.1 Architecture, data & training details

We insert a classification head on top of the pretrained encoder. The end-to-end classifier has the following architecture: Encoder → Avg. pooling → FC-layer (512-units) → RELU → FC-layer (K units), where K is the number of classes. The maximum sequence length of the encoder is set to 128 for FPB and SST2 datasets, 512 for IMDB reviews and 200 for the Bios data.

Each dataset is split into a $80\% - 20\%$ train-test set. 10% of the training set is used as a validation set for hyperparameter optimization. Accuracy and overlap statistics are reported on the test set.

We used the following hyperparameter ranges: learning rate $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and the number of last encoder layers to be fine-tuned $\{0, 2\}$. Fine tuning last few layers of the encoder, as opposed to all the layers, has been shown to lead to superior test set performance (Sun et al., 2019).

We use the AdamW optimizer (Loshchilov and Hutter, 2019). The maximum number of training epochs is 25. We use early stopping with a patience of 5 epochs: if the validation accuracy does not increase for 5 consecutive epochs, we stop the training. The model training was done using PyTorch (Paszke et al., 2019) and HuggingFace Transformers (Wolf et al., 2020) libraries.

B.2 Interpretability methods implementation

Owing to the large runtime of methods like SHAP and Integrated Gradients, interpretations are only computed for a randomly chosen 1000 subsample from the test set. Consequently, metrics like fidelity and Jaccard@K% are reported only on this subset.

	BT	**RB**	**dBT**	**dRB**
FPB	88	94	90	89
SST2	94	96	93	94
IMDB	98	97	97	98
Bios	95	95	95	95

Table 6: A vast percentage of predictions are common between the `Init#1` and `Init#2`, indicating that the models are almost *functionally equivalent*.

	BT	**RB**	**dBT**	**dRB**
FPB	0.19	0.19	0.34	0.34
SST2	0.21	0.22	0.30	0.80
IMDB	0.27	0.51	0.20	0.46
Bios	0.03	0.02	0.01	0.01

Table 7: Test accuracy with `Untrained`.

Vanilla Saliency and SmoothGrad are implemented using the PyTorch `autograd` function. Integrated Gradients and SHAP are implemented using Captum (Kokhlikyan et al., 2020). The parameters of these methods are:

SmoothGrad. Requires two parameters. (i) Number of iterations: Following AllenNLP Interpret (Wallace et al., 2019), we use a value of 10. (ii) Variance of the Gaussian noise $\mathcal{N}(0, \sigma^2)$. The default value of 0.01 leads to attributions that are almost identical to Vanilla Saliency. So we try different values of $\sigma \in \{0.01, 0.05, 0.1, 0.2\}$ and select the one with the lowest infidelity.

Integrated Gradients. Requires setting two parameters. (i) Number of iterations: we use Captum (Kokhlikyan et al., 2020) default of 50. (ii) Feature Baseline: IG requires specifying a baseline (Sundararajan et al., 2017) that has the same dimensionality as the model input, but consists of 'non-informative' feature values. We construct the baseline by computing the embedding value of the unknown vocabulary token and repeating it N times where N is the maximum sequence length of the model.

KernelSHAP. Requires two parameters. (i) Number of feature coalitions: Following the author implementation,[2] we use a value of $2L + 2^{11}$, where L is the number of input tokens in the text. (ii) Dropped token value: SHAP operates by dropping subsets of tokens and estimating model output on these perturbed inputs. We simulate dropping of a

---

[2] `https://github.com/slundberg/shap`

	FPB				SST2				IMDB				Bios			
	BT	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**
VN	70	70	65	71	62	67	57	70	60	58	51	69	52	53	52	58
SG	70	69	64	69	61	67	54	68	60	58	45	68	51	51	47	58
IG	75	73	72	71	71	70	65	66	74	64	68	70	59	55	48	58

Table 8: Mean infidelity of gradient-based interpretability methods when considering *Input ⊙ gradient* reduction of Ding et al. (2019) for `Init#1` (shown as %). In most cases, the performance is worse than when using the L2-norm reduction (Table 2 in Section 3). Lower values are better.

	FPB				SST2				IMDB				Bios			
	BT	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**	**BT**	**RB**	**dBT**	**dRB**
VN	75	30	33	51	42	43	36	33	35	99	42	43	14	14	11	12
SG	74	30	33	51	41	40	36	31	36	99	42	43	13	15	11	12
IG	73	32	31	55	43	42	31	35	49	100	39	42	17	17	11	13
SHP	56	19	23	36	26	40	23	27	9	99	35	19	9	11	7	9
RND	73	36	43	62	49	49	43	51	42	100	49	46	25	32	21	27

Table 9: Mean infidelity of different interpretability methods for `Untrained` (shown as %). Lower values are better.

token by replacing its embedding value with that of the unknown vocabulary token.

Appendix C Additional results

C.1 Accuracy & prediction commonality

Table 6 shows the fraction of predictions common between `Init#1` and `Init#2`.

Table 7 shows the accuracy of `Untrained`. As expected, the accuracy of `Untrained` is much smaller than with trained models.

C.2 Input ⊙ Gradient reduction

Table 8 shows the infidelity of gradient-based interpretability methods when using the Input ⊙ Gradient dot product reduction of Ding et al. (2019). When comparing the results to those with L2 reduction in Table 2, we notice that in *all except two cases* (VN and SG on **dBT** with IMDB data), the performance is worse.

C.3 Infidelity with `Untrained` model

Table 9 shows the infidelity for the `Untrained` model. Much like Figure 2, the table shows that in several cases, the performance of the feature attribution methods (most notably SHP) can be much better than random attribution (RND).

The table also shows an exception for **dBT** on IMDB dataset where for all methods, the infidelity is near 100. This behavior is likely an artefact of the particular initial parameters due to which the model always predicts a certain class irrespective of the input.

Appendix D Examples of top-ranked tokens

We now show some examples of Jaccard@K% computation. The examples show the input text, different models, and top-K% tokens ranked w.r.t. their importance. The attribution method used was VN.

Example 1: SST2 data. Comparing `Init#1` and `Init#2`. Both models predict the sentiment to be positive.
Text. at heart the movie is a deftly wrought suspense yarn whose richer shadings work as coloring rather than substance
Top-25% w.r.t. `Init#1`. {‘substance’, ‘rather’, ‘at’, ‘yarn’, ‘coloring’, ‘movie’}
Top-25% w.r.t. `Init#2`. {‘heart’, ‘##tly’, ‘suspense’, ‘at’, ‘yarn’, ‘def’}
Jaccard@25%. 20

Example 2: SST2 data. Comparing `Init#1` and `Init#2`. Both models predict the sentiment to be positive.
Text. an infectious cultural fable with a tasty balance of family drama and frenetic comedy
Top-25% w.r.t. `Init#1`. {‘fable’, ‘infectious’, ‘cultural’, ‘balance’, ‘an’}
Top-25% w.r.t. `Init#2`. {‘cultural’, ‘balance’, ‘infectious’, ‘fable’, ‘an’}
Jaccard@25%. 100

Example 3: FPB data. Comparing `Init#1` and `Untrained`. Both models predict the sentiment to be negative.

Text. nokia shares hit 13.21 euros on friday , down 50 percent from the start of the year in part because of the slow introduction of touch-screen models
Top-25% w.r.t. `Init#1`. { ‘,’, ‘.’, ‘down’, ‘friday’, ‘shares’, ‘euros’, ‘nokia’, ‘hit’ }
Top-25% w.r.t. `Init#2`. { ‘,’, ‘.’, ‘down’, ‘euros’, ‘friday’, ‘hit’, ‘shares’, ‘nokia’ }
Jaccard@25%. 100

In second and third examples, even though the rankings are different, the set of top-25% tokens is the same leading to a perfect Jaccard@25%.

Multimodal Graph-based Transformer Framework for Biomedical Relation Extraction

Sriram Pingali, Shweta Yadav[*], Pratik Dutta[*], Sriparna Saha
Indian Institute of Technology Patna, India
{1801cs37, shweta.pcs14, pratik.pcs16, sriparna}@iitp.ac.in

Abstract

The recent advancement of pre-trained Transformer models has propelled the development of effective text mining models across various biomedical tasks. However, these models are primarily learned on the textual data and often lack the domain knowledge of the entities to capture the context beyond the sentence. In this study, we introduced a novel framework that enables the model to learn multi-omnics biological information about entities (proteins) with the help of additional multi-modal cues like molecular structure. Towards this, rather developing modality-specific architectures, we devise a generalized and optimized graph based multi-modal learning mechanism that utilizes the GraphBERT model to encode the textual and molecular structure information and exploit the underlying features of various modalities to enable the end-to-end learning. We evaluated our proposed method on Protein-Protein Interaction task from the biomedical corpus, where our proposed generalized approach is observed to be benefited by the additional domain-specific modality.

1 Introduction

The biomedical scientific articles hold the valuable knowledge of biomedical entities (such as protein, drug, gene) and their relationships. However, with the exponential increase in the volume of biomedical articles (Lu, 2011), it is imperative to advance the development of an accurate biomedical text mining tool to extract and curate meaningful information from huge unstructured texts automatically.

One of the cardinal tasks in biomedical document processing is Protein-protein interaction (PPI), where the relation ('*interaction*' or '*non-interaction*') between two protein mentions is identified from the given biomedical text. The knowledge about protein interactions is critical in understanding the biological processes, such as signaling cascades, translations and metabolism, that are regulated by the interactions of proteins that alter proteins to modulate their stability (Elangovan et al., 2020).

Majority of the existing works on PPI in the literature primarily focused only on the textual information present in the biomedical article. However, these approaches lack in capturing **(1)** multi-omnics biological information regarding protein interactions, and **(2)** genetic and structure information of the proteins. A few works (Dutta and Saha, 2020; Asada et al., 2018; Jha et al., 2020; Jha and Saha) have been reported in the literature where the researchers have considered different modalities of the biomedical corpus. However, these multi-modal architectures are modality-specific and thus are very complex. Hence, there is a surge to develop a generalized and optimized model that can understand all the modalities rather than developing various architectures for different modalities.

Towards this, we explore Graph-based Transformer model (GraphBERT) (Zhang et al., 2020) to learn the modality independent graph representation. This enables the model to acquire the joint knowledge of both the modalities (textual and protein structure) under a single learning network. The main contributions of this work are:

1. Besides the textual information of the biomedical corpus, we have also utilized protein atomic structural information while identifying the protein interactions.
2. Developed a generalized modality-agnostic approach that is able to learn the feature representations of both the textual and the protein-structural modality.
3. Our analysis reveals that addition of protein-structure modality increases the efficiency of

---

[*] These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3741–3747
August 1–6, 2021.

model in identifying the interacted protein mentions.

Related Work: Existing studies have adopted traditional statistical and graphical methods (Miyao et al., 2008; Chang et al., 2016) to identify the protein interactions from the textual content. Later, with the success of deep learning, several techniques based on Convolutional Neural Network (Choi, 2018; Peng and Lu, 2017; Ekbal et al., 2016), Recurrent Neural Network (Hsieh et al., 2017; Ahmed et al., 2019), Long Short Term Memory network (Yadav et al., 2019; Ningthoujam et al., 2019; Yadav et al., 2020), and language models (Yadav et al., 2021) based methods are proposed for extracting the relationships from biomedical literature and clinical records. Fei et al. (2020) proposed a span-graph neural model for jointly extracting overlapping entity relationships from biomedical text. The recent advancement of the Transformer model (Lee et al., 2020; Beltagy et al., 2019) in the biomedical domain has also led to significant performance improvement in biomedical relation extraction task (Giles et al., 2020). Recently, the use of multi-modal dataset in BioNLP domain (Dutta and Saha, 2020; Asada et al., 2018) draws the attention of the researchers due to its better performance than the traditional approaches. In contrast, our model is independent of handling multiple modalities without relying on modality-specific architectures.

2 Proposed Method

In this section, we introduce our proposed method and its detailed implementation. The proposed deep multi-modal architecture is illustrated in Figure-1, that consists of four main components: **(1)** *Multi-modal Graph Constructor*, **(2)** *Multi-modal Graph Fusion*, **(3)** *Multi-modal Graph Encoder*, **(4)** *PPI Predictor*. Below we briefly describe each of the model components.

Problem Statement: Given a biomedical input text $S = \{w_1, w_2, \ldots, w_n\}$ having n words, and a pair of protein mentions $p_1, p_2 \in S$, we aim to predict, whether the protein mentions will *'interact'* or *'non-interact'*.

2.1 Multi-modal Graph Constructor

This component consists of two distinct graph constructors for two different modalities, which are Textual Graph Constructor and Protein Structure Graph Constructor. The former, constructs the graph by considering the textual content that aims to capture the lexical and contextual information present in the input. The later, exploits the atomic structure (3D PDB structure) of the protein molecules to build the graph.

Textual Graph Constructor: To generate the textual graph, we begin by first constructing the vocabulary from the training corpus. For each input text S, we use one-hot-encoding mechanism to encode them as a vector representation $R_S \in \mathcal{R}^{|V|}$. However, the representation R_S suffers from the data sparsity as the vocabulary size can become very large for the entire training corpus. To deal with this, we utilized the Principal Component Analysis (PCA) (Wold et al., 1987) to reduce the vector dimensionality. The textual graph $\mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T\}$ is formulated by the nodes $\mathcal{V}_T = \{\hat{R}_{S_1}, \hat{R}_{S_2}, \ldots, \hat{R}_{S_{|N|}}\}$, where $|N|$ is the number of input sentences in the training dataset and $\hat{R}_{S_i} \in \mathcal{R}^{|\hat{V}|}$ is reduced vector representation of size $|\hat{V}|$ for sentence S_i. The link $e_{i,j}$ between nodes $\hat{R}_{S_i}$ and $\hat{R}_{S_j}$ is determined by the common entities (*protein*) present in both the sentences S_i and S_j, if there is no common entity, then link does not exist between the nodes. The edges $\mathcal{E}_T = \{e_{i,j} \mid i, j \in \mathcal{V}_T, \text{ and } protein \in i,\ j\}$ are the set of all the links that exist between any two nodes in the graph, $\mathcal{G}_T$.

Protein Structure Graph Constructor: For the protein structural modality, we created a graph where each node represents an atom and the edge represents the connection between the atoms. To obtain the atomic information about the proteins, first we have mapped the proteins into genes and utilized the PDB (Protein Data Bank)[1] for each associated protein mention. Each protein information obtained from PDB consists of set of atoms $\{a_1, a_2, \ldots, a_A\}$, and a node feature matrix, $N_p \in \mathcal{R}^{A \times d_p}$. The node feature matrix for each protein k undergoes the convolutional operation *CNN(.)* followed by the max-pooling operation, *pool*(.). Formally, $P_k = pool(relu(CNN(N_{p_k})))$. The final protein representation, P_{S_i}, for both the proteins present in the given input sentence S_i is computed as follows: $P_{S_i} = P_1 \oplus P_2$. Following this, the protein structure graph $\mathcal{G}_P = \{\mathcal{V}_P, \mathcal{E}_P\}$ is formulated by the nodes $\mathcal{V}_P = \{P_{S_1}, P_{S_2}, \ldots, P_{S_{|N|}}\}$, where $|N|$ is the number of input sentences in the training dataset and $P_{S_i} \in \mathcal{R}^{d_s}$ is the protein structure

---

[1] https://www.rcsb.org/

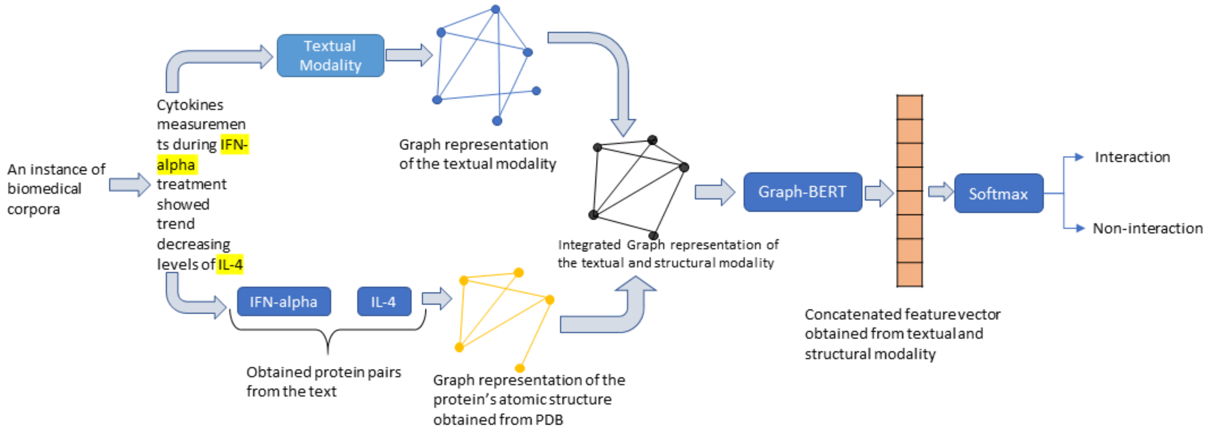

Figure 1: An outline of the proposed deep multi-modal architecture for PPI.

representation of size d_s for sentence S_i.

2.2 Multi-modal Graph Fusion

In this component, we fused the textual graph $\mathcal{G}_T$ and protein structure graph $\mathcal{G}_P$ with the aim of generating a joint representation that is capable of capturing the contextual, lexical, and multi-omnics information. Towards this, we expanded the node information of textual graph with the node information obtained in the protein-structure graph. Specifically, we created a multi-modal graph $\mathcal{G}$ with the nodes $\mathcal{V}$ having concatenated vector representations from the respective nodes of textual graph and protein structure graph. Formally,

$$\mathcal{V}_i = \hat{R}_{S_i} \oplus P_{S_i} \tag{1}$$

The link information remains intact in the multi-modal graph fusion, thus, $\mathcal{E} = \mathcal{E}_{\mathcal{T}}$.

2.3 Multi-modal Graph Encoder

Majority of the existing works on multi-modal relation extraction have treated multiple modalities separately and exploited the modality-specific architectures to learn the respective feature representations. However, these strategies inhibit the learning of inherent shared complementary features, that are often present across the modalities. To address this, we present an end-to-end multi-modality learning mechanism that exploits the single expanded multi-modal graph (obtained from the Multi-modal Graph Expansion component) with the Graph-based Transformer encoder. Specifically, we utilized the Graph-BERT (Zhang et al., 2020) encoder over the other dominants graph neural networks (GNNs) primarily due to its capability to avoid the **(a)** *suspended animation problem* (Zhang and Meng, 2019), and **(b)** *over-smoothing problem* (Li et al., 2018) that hinders the applications of GNNs for deep graph representation learning tasks. For a given multi-modal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the set of nodes ($\mathcal{V}$) and edges ($\mathcal{E}$), Graph-BERT sampled set of graph batches for all the nodes as set $\mathcal{G} = \{g_1, g_2, \ldots, g_{|\mathcal{V}|}\}$. For all the nodes v_j in sub-graph g_i, the Graph-BERT computes raw feature vector embedding e_j^x, role embedding e_j^r, position embedding e_j^p and distance embedding e_j^d. The initial input vector for node v_j is computed as follows: $h_j^{(0)} = e_j^x + e_j^r + e_j^p + e_j^d$. Furthermore, the initial input vectors for all the nodes in g_i can be organized into a matrix $H^{(0)} = [h_i^{(0)}, h_{i,1}^{(0)}, \ldots, h_{i,k}^{(0)}]^\top$, where k is a hyper-parameter. The Graph-Transformer (Zhang et al., 2020) computes the vector representation of D layers of transformers. The final feature (z_i) for node v_j is computed as follows:

$$\begin{aligned} H^{(0)} &= [h_i^{(0)}, h_{i,1}^{(0)}, \cdots, h_{i,k}^{(0)}]^\top \\ H^{(l)} &= \textit{G-Transformer}\left(H^{(l-1)}\right) \\ z_i &= \sum_{m=0}^{m=D} H^{(m)} \end{aligned} \tag{2}$$

2.4 PPI Predictor

The final feature (z_i) of each node i is used to predict the PPI category. Towards this, we employed a feed-forward network with *softmax* activation layer to predict the input text into one of the two classes *interaction* or *non-interaction*. Formally,

$$prob(c = interact|\mathcal{G}, S, \theta) = \frac{exp(\boldsymbol{W}^T z_i^{\text{interact}} + \boldsymbol{b})}{\sum_{k=1}^{K} exp(\boldsymbol{W}^T z_i^k + \boldsymbol{b})} \tag{3}$$

where, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias vector, respectively. K denotes total number of

	Precision	Recall	F-score
Proposed Model	80.84	80.87	80.86
Dutta and Saha (2020)	69.04	88.49	77.54
Yadav et al. (2019)	80.81	82.57	81.68
Hua and Quan (2016)	73.40	77.00	75.20
Choi (2018)	72.05	77.51	74.68
Qian and Zhou (2012)	63.61	61.24	62.40
Peng and Lu (2017)	62.70	68.2	65.30
Zhao et al. (2016)	53.90	72.9	61.60
Tikk et al. (2010)	53.30	70.10	60.00
Li et al. (2015)	72.33	74.94	73.61
Choi and Myaeng (2010)	74.50	70.90	72.60

Table 1: Comparative analysis of the proposed multi-modal approach with state-of-the-art techniques for BioInfer dataset.

		Precision	Recall	F-score
HPRD50	Textual Modality	90.44	92.18	91.28
	Proposed Model	95.47	94.69	95.06
BioInfer	Textual Modality	78.49	79.78	79.06
	Proposed Model	80.84	80.87	80.86

Table 2: Results by uni-modal and multimodal approaches

	Precision	Recall	F-score
Proposed Model	95.47	94.69	95.06
Dutta and Saha (2020)	94.79	75.21	83.87
Yadav et al. (2019)	79.92	77.58	78.73
Tikk et al. (2010)	68.20	69.80	67.80
Tikk et al. (2010)(with SVM)	68.20	69.80	67.80
Palaga (2009)	66.70	80.20	70.90
Airola et al. (2008)(APG)	64.30	65.80	63.40
Van Landeghem et al. (2008)	60.00	51.00	55.00
Miwa et al. (2009)	68.50	76.10	70.90
Airola et al. (2008)(Co-occ)	38.90	100	55.40
Pyysalo et al. (2008)	76.00	64.00	69.00

Table 3: Comparative analysis of the proposed multi-modal approach with other state-of-the-art approaches for HPRD50 dataset.

distinct classes, which are *'interaction'* and *'non-interaction'* in our case.

3 Datasets and Experimental Analysis

Datasets: In this work, we have collected two exemplified multi-modal protein protein interaction datasets (Dutta and Saha, 2020). In these datasets, the authors exemplified two popular benchmark PPI corpora, namely BioInfer[2] and HPRD50[3].

Experimental Setup We have utilized the pre-trained Graph-BERT[4] in our experiment. The initial vocabulary for BioInfer and HPRD50 datasets are 6561 and 1277, respectively. We have projected them into 1000 and 1185 dimension vectors using PCA, respectively. We have kept maximum of 5052 and 1185 number of words in both the datasets, respectively. The filter-size of CNN is set to $3, 4$. We have obtained 1185 length node feature representation for protein structure graph. The nodes of multi-modal graph received the 2185 sized feature representation. We have obtained 2500 and 25859 number of nodes and edges from HPRD50 dataset and 13675 and 15930214 number of nodes and edges from BioInfer dataset for the Graph-BERT training, respectively. We have used all the hyper-parameters of Graph-BERT model in our proposed model. We have kept following hyper-parameters values: subgraph size = 5, hidden size = 32, attention head number = 2, Transformer layers, D = 2, learning rate = 0.01, weight decay = 5e 4, hidden dropout rate = 0.5, attention dropout rate = 0.3, loss = cross entropy, optimizer = adam (Kingma and Ba, 2014). The hyper-parameters are chosen based on the 5-fold cross-validation experiments on both the datasets.

Results and Analysis: We have compared the performance (*c.f.* Table-1,3) of our proposed model with the existing state-of-the-art methods on PPI for both the datasets. These existing methods are based on different techniques like kernel-based (Choi and Myaeng, 2010; Tikk et al., 2010; Qian and Zhou, 2012; Li et al., 2015), deep neural network-based (Zhao et al., 2016; Yadav et al., 2019), multi-channel dependency-based convolutional neural network model (Peng and Lu, 2017), semantic feature embedding (Choi, 2018), shortest dependency path (Hua and Quan, 2016) and a recent deep multi-modal approach (Dutta and Saha, 2020). It is to be noted that our results on BioInfer and HPRD50 are not directly comparable with the existing approaches as other methods have utilized different test sets for evaluation. From the above comparative study, it is evident that our proposed multi-modal approach identifies the protein interactions in an efficient way and can be further improved in different ways.

Discussion: To analyze the role of each modality, we conducted ablation study as shown in Table 2. We performed the experiments with the textual modality. Here, we could not consider the protein-structural modality alone as it would bring the conflicting labeling relation. For example, consider two sentences that contain same pair of proteins

---

[2] http://corpora.informatik.hu-berlin.de/

[3] https://goo.gl/M5tEJj

[4] https://github.com/jwzhanggy/Graph-Bert

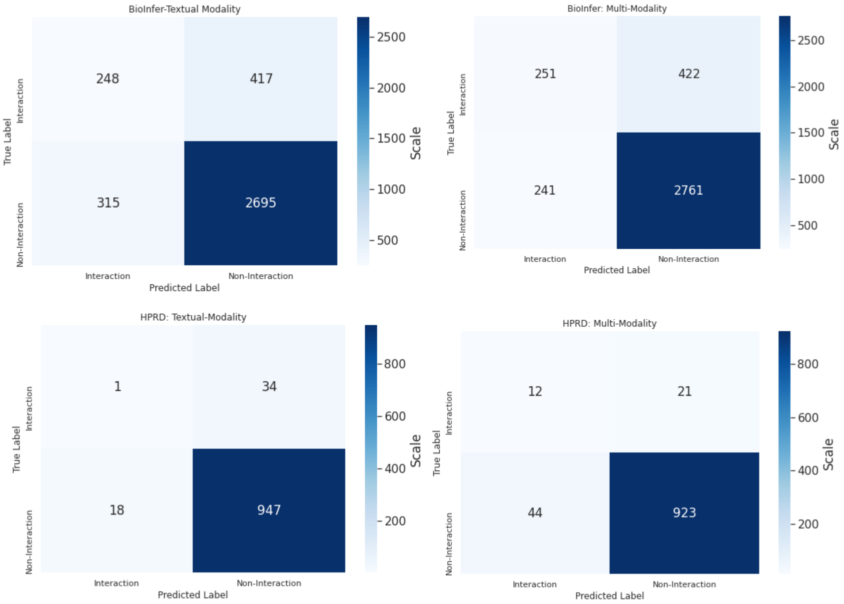

Figure 2: Confusion matrices of our proposed approach on both HPRD and BioInfer datasets with only textual modality and text+structure modality.

but these proteins can have conflicting relations (interacting or non-interacting) depending on the context of sentences in which they appear. Hence, we could not consider the protein-structural modality alone. Though the structural modality is unable to draw any conclusion alone, however the integration of both the modalities demonstrates the improvements (3.78% and 1.8%, in terms of F-score for HPRD50 and BioInfer, respectively) over the textual modality alone.

4 Error Analysis

The comparative confusion matrices with only textual-modality and multi-modality for both the datasets are shown in Figure-2. We have performed error analysis to postulate possible reasons and areas with scope of improvement in our experiments. After careful study on false positive and false negative classes, following observations can be made.
1) Instances with a large number of protein mentions in a single sentence can cause misclassification. For example, the maximum number of proteins in any instances of BioInfer and HPRD50 datasets are 26 and 24, respectively. These large number of proteins present in a single instance may lead the network to misclassificaton.
2) Few samples contain repeated mentions of the same protein. This adds noise and might lead to losing useful contextual information.
3) To get a consistent graph from molecular structure, the nodes were required to be of the same length. This is done by padding the vectors with zeros, and when the PDB is not available, a null vector is used for consistency. A better handling of missing data will help in learning the proposed model.

5 Conclusion

This work presents a novel modality-agnostic Graph-based framework to identify the interactions between the proteins. Specifically, we explored two modalities: *textual*, and *molecular structure* that enable the model to learn the domain-specific multi-omnics information complementary with the task-specific contextual information. A detailed comparative results and analysis proves that our proposed multi-modal approach can capture underlying molecular structure information without relying on sophisticated modality-specific architectures. Future work aims at extending this study to the other related tasks like drug-drug interactions.

Acknowledgment

Sriparna Saha gratefully acknowledges the Young Faculty Research Fellowship (YFRF) Award, supported by Visvesvaraya Ph.D. Scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (Formerly Media Lab Asia) for carrying out this research.

References

Mahtab Ahmed, Jumayel Islam, Muhammad Rifayat Samee, and Robert E Mercer. 2019. Identifying protein-protein interaction using tree lstm and structured attention. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 224–231. IEEE.

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):S2.

Masaki Asada, Makoto Miwa, and Yutaka Sasaki. 2018. Enhancing drug-drug interaction extraction from texts by molecular structure information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–685, Melbourne, Australia. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Yung-Chun Chang, Chun-Han Chu, Yu-Chen Su, Chien Chin Chen, and Wen-Lian Hsu. 2016. PIPE: a protein–protein interaction passage extraction module for BioCreative challenge. *Database*, 2016. Baw101.

Sung-Pil Choi. 2018. Extraction of protein–protein interactions (ppis) from the literature by deep convolutional neural networks with various feature embeddings. *Journal of Information Science*, 44(1):60–73.

Sung-Pil Choi and Sung-Hyon Myaeng. 2010. Simplicity is better: revisiting single kernel ppi extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 206–214. Association for Computational Linguistics.

Pratik Dutta and Sriparna Saha. 2020. Amalgamation of protein sequence, structure and textual information for improving protein-protein interaction identification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6396–6407, Online. Association for Computational Linguistics.

Asif Ekbal, Sriparna Saha, Pushpak Bhattacharyya, et al. 2016. A deep learning architecture for protein-protein interaction article identification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3128–3133. IEEE.

Aparna Elangovan, Melissa Davis, and Karin Verspoor. 2020. Assigning function to protein-protein interactions: a weakly supervised biobert based approach using pubmed abstracts. *arXiv preprint arXiv:2008.08727*.

Hao Fei, Yue Zhang, Yafeng Ren, and Donghong Ji. 2020. A span-graph neural model for overlapping entity relation extraction in biomedical texts. *Bioinformatics*.

Oliver Giles, Anneli Karlsson, Spyroula Masiala, Simon White, Gianni Cesareni, Livia Perfetto, Joseph Mullen, Michael Hughes, Lee Harland, and James Malone. 2020. Optimising biomedical relationship extraction with biobert: Best practices for data creation. *bioRxiv*.

Yu-Lun Hsieh, Yung-Chun Chang, Nai-Wen Chang, and Wen-Lian Hsu. 2017. Identifying protein-protein interactions in biomedical literature using recurrent neural networks with long short-term memory. In *Proceedings of the eighth international joint conference on natural language processing (volume 2: short papers)*, pages 240–245.

Lei Hua and Chanqin Quan. 2016. A shortest dependency path based convolutional neural network for protein-protein relation extraction. *BioMed Research International*, 2016.

K. Jha and S. Saha. Amalgamation of 3d structure and sequence information for protein–protein interaction prediction. *Sci Rep*, 10.

Kanchan Jha, Sriparna Saha, and Matloob Khushi. 2020. Protein-protein interactions prediction based on bi-directional gated recurrent unit and multimodal representation. In *Neural Information Processing - 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18-22, 2020, Proceedings, Part V*, volume 1333 of *Communications in Computer and Information Science*, pages 164–171. Springer.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Lishuang Li, Rui Guo, Zhenchao Jiang, and Degen Huang. 2015. An approach to improve kernel-based protein–protein interaction extraction by learning from large-scale network data. *Methods*, 83:44–50.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhiyong Lu. 2011. PubMed and beyond: a survey of web tools for searching biomedical literature. *Database*, 2011. Baq036.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. Protein–protein interaction extraction by leveraging multiple kernels and parsers. *International journal of medical informatics*, 78(12):e39–e46.

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Evaluating contributions of natural language parsers to protein–protein interaction extraction. *Bioinformatics*, 25(3):394–400.

Dhanachandra Ningthoujam, Shweta Yadav, Pushpak Bhattacharyya, and Asif Ekbal. 2019. Relation extraction between the clinical entities based on the shortest dependency path based lstm. *arXiv preprint arXiv:1903.09941*.

Peter Palaga. 2009. Extracting relations from biomedical texts using syntactic information. *Mémoire de DEA, Technische Universität Berlin*, 138.

Yifan Peng and Zhiyong Lu. 2017. Deep learning for extracting protein-protein interactions from biomedical literature. In *BioNLP 2017*, pages 29–38, Vancouver, Canada,. Association for Computational Linguistics.

Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. In *BMC bioinformatics*, volume 9, page S6. BioMed Central.

Longhua Qian and Guodong Zhou. 2012. Tree kernel-based protein–protein interaction extraction from biomedical literature. *Journal of biomedical informatics*, 45(3):535–543.

Domonkos Tikk, Philippe Thomas, Peter Palaga, Jörg Hakenberg, and Ulf Leser. 2010. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS computational biology*, 6(7):e1000837.

Sofie Van Landeghem, Yvan Saeys, Bernard De Baets, and Yves Van de Peer. 2008. Extracting protein-protein interactions from text using rich feature vectors and feature selection. In *3rd International symposium on Semantic Mining in Biomedicine (SMBM 2008)*, pages 77–84. Turku Centre for Computer Sciences (TUCS).

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Shweta Yadav, Asif Ekbal, Sriparna Saha, Ankit Kumar, and Pushpak Bhattacharyya. 2019. Feature assisted stacked attentive shortest dependency path based bi-lstm model for protein–protein interaction. *Knowledge-Based Systems*, 166:18–29.

Shweta Yadav, Usha Lokala, Raminta Daniulaityte, Krishnaprasad Thirunarayan, Francois Lamy, and Amit Sheth. 2021. "when they say weed causes depression, but it's your fav antidepressant": Knowledge-aware attention framework for relationship extraction. *PloS one*, 16(3):e0248299.

Shweta Yadav, Srivastsa Ramesh, Sriparna Saha, and Asif Ekbal. 2020. Relation extraction from biomedical and clinical text: Unified multitask learning framework. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Jiawei Zhang and Lin Meng. 2019. Gresnet: Graph residual network for reviving deep gnns from suspended animation. *arXiv preprint arXiv:1909.05729*.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Zhehuan Zhao, Zhihao Yang, Hongfei Lin, Jian Wang, and Song Gao. 2016. A protein-protein interaction extraction approach based on deep neural network. *International Journal of Data Mining and Bioinformatics*, 15(2):145–164.

Summary Grounded Conversation Generation

Chulaka Gunasekara, Guy Feigenblat, Benjamin Sznajder, Sachindra Joshi, David Konopnicki
IBM Research AI
`chulaka.gunasekara@ibm.com, {guyf, benjams}@il.ibm.com`
`{jsachind@in, davidko@il*}.ibm.com`

Abstract

Many conversation datasets have been constructed in the recent years using crowdsourcing. However, the data collection process can be time consuming and presents many challenges to ensure data quality. Since language generation has improved immensely in recent years with the advancement of pre-trained language models, we investigate how such models can be utilized to generate entire conversations, given only a summary of a conversation as the input. We explore three approaches to generate summary grounded conversations, and evaluate the generated conversations using automatic measures and human judgements. We also show that the accuracy of conversation summarization can be improved by augmenting a conversation summarization dataset with generated conversations.

1 Introduction

Automatic conversation systems require large quantities of data to learn task specific language patterns and underlying conversation policies. Such data either come from human-to-human conversation logs (Lowe et al., 2015; Hardalov et al., 2018) or is collected in crowd-sourced environments, where two or more crowd-workers play specific roles under some guidelines (Zhang et al., 2018; Budzianowski et al., 2018). Since real human-to-human conversation logs are scarce, many datasets have been created using the latter approach. However, crowd-sourced conversation data collection is time consuming, costly and presents multiple challenges to ensure data quality (Kang et al., 2018).

Conversation summarization is an emerging research area that has been ill-studied due to the lack of large-scale datasets. Most existing public datasets in this domain are small, for example, AMI meeting corpus (McCowan et al., 2005) contains 137 summary transcripts. CRD3 (Rameshkumar and Bailey, 2020) is a spoken conversation dataset that consists of 159 conversations and summaries. Samsum (Gliwa et al., 2019), the only large scale dataset for conversation summarization, contains over $16,000$ open-domain conversations and summaries created artificially by humans.

Large scale pre-trained language models (PLMs) (Lewis et al., 2020; Brown et al., 2020; Raffel et al., 2020) have been used in various text generation tasks (Budzianowski and Vulić, 2019; Min et al., 2020; Cachola et al., 2020). In recent studies, PLMs are used to generate training data for natural language processing (NLP) applications. For example, Anaby-Tavor et al. (2020); Yang et al. (2020) use PLMs to create paraphrases for intent classifiers in conversation systems, and show that, when the original datasets are augmented with the generated data, performance improves. More recently Mohapatra et al. (2020) generated entire conversations grounded on instructions that are provided to crowd-workers using a modular approach, where different PLMs are trained for different roles.

Our Contributions: We investigate how PLMs can be utilized to generate entire conversations that are grounded on a given summary. We explore three approaches: (1) Supervised Learning (SL) based conversation generation *(SL-Gen)*: where, a PLM is trained to generate an entire conversation, taking the summary of a conversation as input, (2) Reinforced Learning (RL) based conversation generation *(RL-Gen)*: where, we further improve the *SL-Gen* method using the quality of the generated conversations as a reward, and (3) Controlled turn-by-turn conversation generation *(CN-Gen)*: which allows us to generate conversations turn-by-turn, constrained on the summary and a set of pre-defined control parameters. We evaluate the quality of the generated conversations by conducting automatic and human evaluation. We

---

[*] Current address: david.konopnicki@booking.com

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3748–3756
August 1–6, 2021.

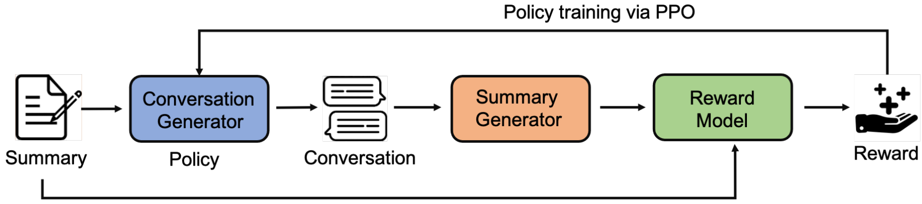

Figure 1: The RL based conversation generation framework

also show that once a conversation summarization dataset is augmented with the generated conversations, the performance of the downstream summarization task is improved.

2 Summary grounded conversation generation

In the conversation summarization task, a model takes a conversation as input, and learns to generate a summary. We study the inverse of that problem, where the input to our model is a summary, and the model generates a conversation. In this section, we propose three models for this task and the hyperparameters used in training the models are available in Section A of the appendix.

2.1 SL based generation (SL-Gen)

A seq2seq model can be trained for this task by providing a summary as the input and generating a conversation token-by-token. As PLMs have shown significant improvement over the traditional seq2seq architecture for text generation, we use a GPT-2 model and fine-tune it to generate a conversation given a summary as the input. Our input to the model follows the following format: *<bos>summary text <dialog>conversation text<eos>*. We also use different token-type-ids to indicate the summary and the conversation text. The model is trained to optimize Cross Entropy loss.

2.2 RL based generation (RL-Gen)

Many studies train text generation models with RL (Paulus et al., 2018; Li et al., 2016), where the generator network is optimized with a task specific reward. We investigate how the quality of the generated conversation can be used as a reward to improve the generation network. To this end, we train a summary generator network, which generates a summary, given a conversation. We measure the quality of the generated conversation by identifying the similarity between the summary of the generated conversation (generated, in turn, by the summary generator network) and the ground truth summary. The similarity score is used as a reward to train the conversation generation model. Our RL based generation framework is shown in Figure 1, and the critical components are described below.

Conversation Generator: A trained SL-Gen model is used as the conversation generator, which, given an summary can generate a conversation.

Summary Generator: We use a lightweight variant of BART (Lewis et al., 2019), named *DistilBART*, which is fine-tuned on the Extreme summarization task (Narayan et al., 2018). We further fine-tune this instance on the conversation summarization data by providing the conversations as the input and training the model to output summaries.

Reward Model: Once the Summary Generator generates an output summary for the generated conversation, the reward model compares it with the ground truth summary, which was used to ground the conversation generation. As Paulus et al. (2018) we use ROUGE-2 F1-score as the reward.

Policy training: We use proximal policy optimization (Schulman et al., 2017) as the optimizer for the policy training as it prevents the generator from deviating far away from the pretrained LM (Wu et al., 2020).

2.3 Controlled conversation generation

We propose another approach, *(CN-Gen)*, for conversation generation, which grants more control over the properties of the generated conversations. Here, we generate one utterance of the conversation at a time, as opposed to the *RL-Gen*, where we generate the whole conversation at once. The properties of the generated conversations is controlled by adding several components to the input sequence to the model. The following three variables were used as the control parameters, (1) Number of remaining turns to generate in the conversation (Num turns): During the generation of a turn, we indicate the remaining number of turns in the conversation. In generating a n turn conversation, this starts with n for the first turn and reduces by 1 after the generation of each turn, (2) The speaker of the next

Summary: person0 will be late. person1 will order pasta with salmon and basil for her.	
2 turn conversation: <Person0> I'll be late <Person1> I'll order some pasta with salmon and basil for you.	**3 turn conversation** <Person0> I'll be late. <Person1> I'll order some pasta with salmon and basil for you. <Person0> Thanks a lot!
6 turn conversation <Person0> Hello, I am going to be late. <Person1> Ok <Person1> I'll order some pasta with salmon and basil <Person0> Ok, sounds good! <Person0> Thank you! <Person1> No problem	**10 turn conversation** <Person0> I'll be late <Person1> ok <Person1> do you want me to order something for you? <Person1> pasta? <Person0> Yes <Person1> with salmon? <Person0> Yes <Person1> Ok <Person1> how about basil? <Person1> Yes please!

Table 1: Multiple conversations generated by the CN-Gen approach grounded on the same summary

Model	Ave. Turns	Ave. Tokens/Turn
Ground truth	11.55 ± 6.48	7.10 ± 6.29
SL-Conv-Gen	10.54 ± 6.80	5.69 ± 4.40
RL-Conv-Gen	8.40 ± 4.78	5.14 ± 3.64
CN-Conv-Gen	9.70 ± 5.67	5.62 ± 4.05

Table 2: Properties of the generated conversations.

turn (Speaker): This indicates to the model the speaker of the next turn, and (3) The length of the next turn (Turn length): We define, 3 categories of lengths: Short (≤ 3 tokens), Long (> 10 tokens) and Medium (otherwise).

We use the following input representation to fine-tune a GPT-2 model: *<bos> summary text <context> dialog context <turns_to_go> Num turns <speaker> speaker <turn_length> turn length <turn> utterance <eos>*. Changing these parameters allows us to generate different variants of conversations which are grounded on the same summary. During training, we obtain the values for the control parameters from the ground truth conversations, and at inference we randomly select the next speaker, number of turns of the conversation to be generated (in a range of 4-15 turns), and the next turn length. In Table 1 we show conversations of different lengths that were generated by the CN-Gen approach grounded on the same summary by changing the control parameters.

A summary and a conversation from the Samsum dataset (Gliwa et al., 2019), along with the conversations generated by the three aforementioned algorithms are shown in Figure 2. More examples are provided in the Section B of the Appendix.

3 Experiments

We experiment on the Samsum (Gliwa et al., 2019) dataset, which, to the best of our knowledge, is the only public large-scale conversation summarization dataset. We pre-process the dataset by replacing the personal names (ex: John) with unique tags (ex:<person_0 >). First, we evaluate of the quality of generated conversations using automatic measures and human judgments, and then assess the performance of the generated conversations in a downstream summarization task after augmentation.

3.1 Quality of the generated conversations

We evaluate the quality of the conversations generated by the three approaches that were introduced in Section 2. In Table 2 we show the properties of generated conversations and the ground truth conversations in the test set of Samsum dataset.

Automatic Evaluation: We trained the conversation generation models on the Samsum training set and generated conversations on the test set. We compare the generated conversation with the ground truth conversations using the measures used by Sharma et al. (2017) to evaluate conversation system responses. The results shown in Table 3 suggest that CN-Gen outperform the SL-Gen and RL-Gen on all measures.

We also compare the summaries of generated conversations (generated by the Summary Generator) with the ground truth summaries, and the results are shown in Table 4. We believe that this is a semantic evaluation of the conversations, as the summaries capture the crux of the conversations. According to the results, CN-Gen outperforms the other two methods. This, along with the previous result suggest that the conversations produced by CN-Gen are the most similar to the ground truth conversations.

Human Evaluation: To evaluate the quality of generated conversations, we randomly selected 50 summaries from the Samsum test dataset and generated conversations using the three models. Three NLP experts were then asked to read the ground truth summary and rank the four conversations (3 generated and the ground truth conversation) using a [1-5] scale according to Grammaticality, Coherency, and Informativeness, with respect to the ground truth summary. Results are shown in table 5. As expected, the ground-truth conversations obtained the highest scores on all three aspects and can be considered as an upper bound for this task.

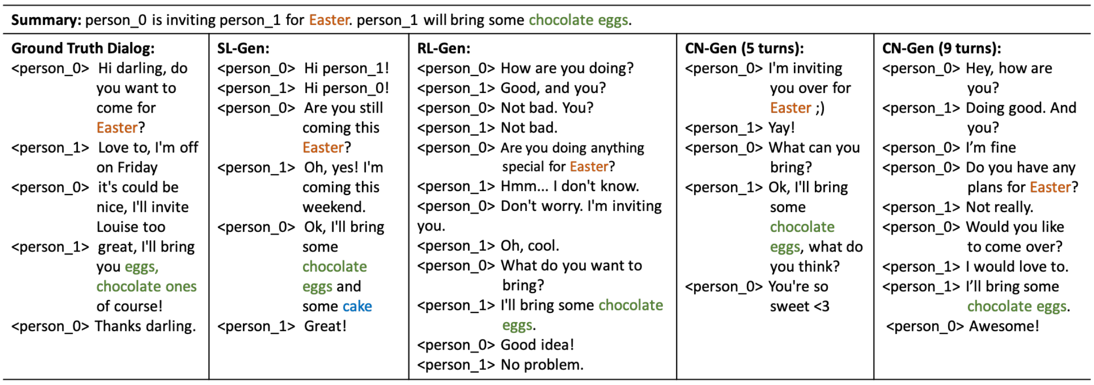

Summary: person_0 is inviting person_1 for Easter. person_1 will bring some chocolate eggs.				
Ground Truth Dialog:	**SL-Gen:**	**RL-Gen:**	**CN-Gen (5 turns):**	**CN-Gen (9 turns):**
<person_0> Hi darling, do you want to come for Easter? <person_1> Love to, I'm off on Friday <person_0> it's could be nice, I'll invite Louise too <person_1> great, I'll bring you eggs, chocolate ones of course! <person_0> Thanks darling.	<person_0> Hi person_1! <person_1> Hi person_0! <person_0> Are you still coming this Easter? <person_1> Oh, yes! I'm coming this weekend. <person_0> Ok, I'll bring some chocolate eggs and some cake <person_1> Great!	<person_0> How are you doing? <person_1> Good, and you? <person_0> Not bad. You? <person_1> Not bad. <person_0> Are you doing anything special for Easter? <person_1> Hmm... I don't know. <person_0> Don't worry. I'm inviting you. <person_1> Oh, cool. <person_0> What do you want to bring? <person_1> I'll bring some chocolate eggs. <person_0> Good idea! <person_1> No problem.	<person_0> I'm inviting you over for Easter ;) <person_1> Yay! <person_0> What can you bring? <person_1> Ok, I'll bring some chocolate eggs, what do you think? <person_0> You're so sweet <3	<person_0> Hey, how are you? <person_1> Doing good. And you? <person_0> I'm fine <person_0> Do you have any plans for Easter? <person_1> Not really. <person_0> Would you like to come over? <person_1> I would love to. <person_1> I'll bring some chocolate eggs. <person_0> Awesome!

Figure 2: Examples of a conversations grounded on the same summary. The key terms are highlighted in colors.

Model	BLEU-4	METEOR	ROUGE-L
SL-Gen	2.81	12.06	21.53
RL-Gen	3.53	12.29	25.40
CN-Gen	**4.94**	**15.64**	**26.22**

Table 3: Evaluation of generated conversations against ground truth conversations

Model	ROUGE_1	ROUGE_2	ROUGE_L
SL-Gen	46.85	25.29	45.97
RL-Gen	52.51	31.23	51.68
CN-Gen	**53.46**	**32.52**	**52.93**

Table 4: Rouge F1 evaluation of summaries of conversations against the ground truth summaries

RL-Gen and CN-Gen obtained higher scores than SL-Gen and relatively good scores compared to the Ground Truth conversations. This corroborates the assumption that our proposed models generate high quality conversations. The Welch Two Sample t-test (Welch, 1947) shows that both RL-Gen and CN-Gen models outperform the SL-Gen model statistically significantly with $p < 0.0001$. However, there is no statistical significance between the results obtained from RL-Gen and CN-Gen. We report in Table 6 the average quadratic Cohen's Kappa calculated over the three possible combinations of two judges (Toledo et al., 2019).

CN-Gen obtained the best scores during the automatic evaluation, while RL-Gen got the best scores from the human evaluation. The CN-Gen conversations are longer than the RL-Gen conversation by 1.3 turns on average (see Table 2), and hence would contain more word overlap with the ground truth. This results in better automatic evaluation scores for the CN-Gen, while the humans prefer short targeted conversations generated by RL-Gen.

3.2 Evaluation on the summarization task

To further evaluate the quality of the generate conversations, we augmented a conversation summarization dataset with generated conversations and evaluated the summarization model. We followed the following process: (1) We randomly selected x% of the summaries of the dataset and trained our conversation generation models, (2) The trained models were applied on the other (y=100-x%) of the summaries and generated conversations, (3) Those generated conversations along with the original summaries were added to the data. Using this approach, we can add extra y% (summary, conversation) pairs to the training data, (4) The conversation summarization model (discussed in Section 2 under 'Summary Generator') was trained on the augmented data. We compare the performance of the conversation summarization model on the original dataset and with augmentation.

Automatic Evaluation: We compare the three conversation generation methods at different augmentation percentages, and the results are shown in Table 7. At all augmentation levels, the summarization models trained with augmented data outperform the summarization model trained on the original dataset (without augmentation). CN-Gen based augmentation produces the best accuracy compared to other two methods. One prevalent pattern is that, when augmentation data increases, the accuracy seems to increase up to a certain point and then starts to decrease. The best accuracies were found around 30% data augmentation. We believe that more augmentation leads performance to drop due to the following reason. For augmenting with more data, we are left with less data to train the model for conversation generation (for 10% augmentation, the conversation generation models are trained on 90% of the data, while for 50% augmentation, the models are trained only on 50% of the

Model	Info	Gram	Cohe
Ground-Truth	4.56	4.46	4.47
SL-Gen	2.22	2.85	2.37
RL-Gen	**3.20**	**3.50**	**3.14**
CN-Gen	3.10	3.43	3.09

Table 5: Human evaluation of generated conversations

Model	Info	Gram	Cohe
Ground-Truth	0.04	0.22	0.25
SL-Gen	0.35	0.26	0.42
RL-Gen	0.47	0.35	0.45
CN-Gen	0.60	0.40	0.60

Table 6: Average Cohen's Kappa for human evaluation of generated conversations

Method	Augmentation %	ROUGE_1	ROUGE_2	ROUGE_L
	0% (Original)	51.84	30.98	43.98
SL-Gen	10%	52.82	31.99	44.89
	20%	**52.90**	**32.01**	44.97
	30%	52.88	32.02	**45.01**
	40%	52.61	31.98	44.96
	50%	52.55	31.98	44.80
RL-Gen	10%	52.93	32.05	44.92
	20%	53.30	32.15	45.20
	30%	**53.81**	**32.21**	**45.77**
	40%	52.86	32.06	44.99
	50%	52.64	32.07	44.88
CN-Gen	10%	53.29	32.36	45.08
	20%	53.36	32.53	45.27
	30%	**54.02**	**33.28**	**46.06**
	40%	52.14	31.76	44.14
	50%	52.36	31.75	44.85

Table 7: ROUGE F-1 evaluation on Samsum test set.

data). Therefore as the augmentation increases, the quality of generated conversations go down. This leads to overall smaller gains in the summarization task with increased augmentation after some point. To neutralize the effect of increasing the data points during augmentation, we experimented with a baseline which over-samples the original training data at different percentages to obtain same number of training instances as the augmented datasets. While the ROUGE-2 obtained with the original training data is 30.98, oversampling at 10%, 20%, 30%, 40% and 50%, only changes the ROUGE-2 to 30.55, 30.38, 30.74, 30.99 and 30.27 respectively. Hence, this suggests that oversampling hardly changes ROUGE scores obtained by training with the original dataset, while the augmentation according to our algorithms show significantly improved scores (as shown in Table 7).

Human Evaluation: We recruited 3 NLP experts to evaluate 50 instances of summaries generated with data augmentation (RL-Gen, CN-Gen), and respective summaries generated without augmentation (No-Aug). Here we consider two aspects with respect to a ground-truth summary: Coherency (whether the summary is easy to read) and Focus (whether the summary represents the ground-truth summary). Following (Amplayo and Lapata, 2020) we use the Best-Worst Scaling method. The score of each system is computed as the percentage of times it was chosen as the Best system minus times it was chosen as Worst. On the Coherency question, RL-Gen, CN-Gen and No-Aug obtained scores of 12.6, 6.6 and -4.0 respectively. On the Focus question RL-Gen, CN-Gen, and No-Aug obtained scores of 14.6, 6.0 and -2.6 respectively. These results confirm that the use of augmentation improves the quality of the summaries.

4 Conclusion

We investigated how the PLMs can be utilized to generate entire conversations that are grounded on a summary. We propose three approaches for conversation generation: SL-Gen, RL-Gen and CN-Gen and conducted multiple automatic and human evaluations to assess the quality of the generated conversations. Both automatic and human evaluations show that when compared to the ground truth conversations, RL-Gen and CN-Gen obtain high scores, suggesting that the proposed models generate high quality conversations. When a conversation summarization dataset is augmented with the generated conversations, the performance of conversation summarization is improved (over to 7% improvement in ROUGE-2 F-1), which also suggests that the proposed methods generate high quality conversations.

5 Ethics

We have used the publicly available Samsum dataset (`https://huggingface.co/datasets/samsum`). For the human evaluation of both conversations and summaries, we recruited 3 NLP researchers, who have graduate degree in NLP and Machine Learning. The annotation task itself was executed on Appen.com platform. Before the official annotation, we sampled 10 tasks to get an estimate of the duration of the task, and to make sure the instructions are clear enough.

References

Reinald Kim Amplayo and Mirella Lapata. 2020. Unsupervised opinion summarization with noising and denoising. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1934–1945.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's gpt-2-how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S Weld. 2020. Tldr: Extreme summarization of scientific documents. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4766–4777.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, page 70.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2018. Towards automated customer support. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 48–59. Springer.

Yiping Kang, Yunqi Zhang, Jonathan K Kummerfeld, Lingjia Tang, and Jason Mars. 2018. Data collection for dialogue system: A startup perspective. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 33–40.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Ryan Lowe, Nissan Pow, Iulian Vlad Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294.

Iain McCowan, Jean Carletta, Wessel Kraaij, Simone Ashby, S Bourban, M Flynn, M Guillemot, Thomas Hain, J Kadlec, Vasilis Karaiskos, et al. 2005. The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88, page 100. Citeseer.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797.

Biswesh Mohapatra, Gaurav Pandey, Danish Contractor, and Sachindra Joshi. 2020. Simulated chats for task-oriented dialog: Learning to generate conversations from instructions. *arXiv preprint arXiv:2010.10216*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Revanth Rameshkumar and Peter Bailey. 2020. Storytelling with dialogue: A critical role dungeons and dragons dataset. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5121–5134.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Assaf Toledo, Shai Gretz, Edo Cohen-Karlik, Roni Friedman, Elad Venezian, Dan Lahav, Michal Jacovi, Ranit Aharonov, and Noam Slonim. 2019. Automatic argument quality assessment-new datasets and methods. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5629–5639.

Bernard L Welch. 1947. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Qingyang Wu, Lei Li, and Zhou Yu. 2020. Textgail: Generative adversarial imitation learning for text generation. *arXiv preprint arXiv:2004.13796*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1008–1025.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

A Model Training and Hyperparameter Details

A.1 Supervised Conversation Generation (SL-Conv-Gen)

We fine-tune a GPT-2 language model using the implementation available at HuggingFace (Wolf et al., 2019). The hyper-parameters used during training and inference are shown below. The model takes around 6 hours to train on 2 V100 GPUs (single machine).

```
model_name_or_path: gpt2
per_gpu_train_batch_size: 4
per_gpu_eval_batch_size: 4
gradient_accumulation_steps: 4
learning_rate: 6.25e-5
adam_epsilon: 1e-8
max_grad_norm: 1.0
num_train_epochs: 10
warmup_steps: 500
min_length: 20
max_length: 512
top_k: 0
top_p: 0.95
```

A.2 Summary Generator

We use DistilBART instance[1] fine-tuned on the extreme summarization (XSum) task, and we fine-tune this model further on the Samsum dataset. The model takes around 12 hours to train on 2 V100 GPUs (single machine).

The hyperparameters used for training the DistilBART model are as follows:

```
train_batch_size: 4
eval_batch_size: 4
num_train_epochs: 10
model_name_or_path: sshleifer/distilbart
-xsum-12-6
learning_rate: 3e-5
val_check_interval: 0.1
max_source_length: 512
max_target_length: 80
```

A.3 Reinforced Learning based conversation generation (RL-Conv-Gen)

To train the RL based conversation generation model, we adapted a publicly available Proximal Policy Optimization (PPO) implementation [2]. The model takes around 12 hours to train on 2 V100 GPUs (single machine). Following hyperparameters were used to train the model.

```
steps: 10000
batch_size: 16
forward_batch_size: 4
learning_rate: 1.41e-5
init_kl_coef:0.2
target: 6
horizon:10000
gamma:1
lam:0.95
cliprange: 0.2
cliprange_value: 0.2
vf_coef: 0.1
```

---

[1] https://huggingface.co/sshleifer/distilbart-cnn-12-6

[2] https://github.com/lvwerra/trl

B Sample summaries with corresponding ground-truth

Figure 3 shows some samples of dialogs with their corresponding summaries - ground-truth and automatic generated ones.

Summary: Person0 closed some deals today. Person1 didn't manage to do it.

Ground Truth Dialog:	**SL-Gen:**	**RL-Gen:**	**CN-Gen:**
<Person0> It's been a long day. <Person1> Same here, Person0, good or bad? <Person0> Very busy. I closed some deals but I had a lot of stress also. <Person1> Lucky you. I couldn't close any deal. It annoys me. Some customers are negotiating for days, weeks. <Person0> Maybe they're just fishing. Want to compare offers. <Person1> That's what they do.	<Person0> Hi, are you still alive? <Person1> Hi, yes, still alive. <Person0> I closed some deals today. <Person1> Yeah, that's awesome. <Person0> Thanks.	<Person0> Hey, how are you? <Person1> Good, busy as usual. <Person0> I closed some deals today. <Person1> Too late, didn't manage to do it. <Person0> Sorry.	<Person0> Did you see this? ¡file_other¿ <Person1> Cool. What's up? <Person0> I closed some deals today and I feel good about it. <Person1> Great. I didn't manage to do it today. I'm sorry. I had no choice. <Person0> I'm sorry. <Person1> I didn't have the tools.

Summary: Person0 bought a table, six chairs, a vase and a pile of clothes and the second hand shop downtown. She paid 70 euros for everything.

Ground Truth Dialog:	**SL-Gen:**	**RL-Gen:**	**CN-Gen:**
<Person0> I went to the second hand shop downtown. <Person1> Cool Person0 What did you get? <Person0> Lots of stuff. A table, six chairs, a vase, a pile of clothes <Person1> That's really a lot of stuff Person0. Send us pictures <Person0> ¡photo_file¿ <Person1> Beautiful table.Is it wood? <Person0> Yes, oak. <Person1> Awesome! These are the clothes? I love the black dress. <Person0> It's absolutely beautiful. As soon as I saw it I knew it would be mine. I'm so happy with what I bought.	<Person0> I bought the second hand shop downtown. <Person1> So what happened? <Person0> I bought everything. <Person1> What do you mean? <Person0> I paid 70 euro for everything. <Person1> How much? <Person0> 6 chairs, six chairs, a vase and a pile of clothes. <Person1> Sounds like a bargain. <Person0> ou should be in business.	<Person0> Do you remember the second hand in downtown? <Person1> Yes, I do. What's inside? <Person0> A table, six chairs, a vase and a pile of clothes. <Person1> Is it expensive? <Person0> 70 euros.	<Person0> Look what I've bought! A table, six chairs, a vase. I bought a table, six chairs, a vase and a pile of clothes. <Person1> How much did you pay for everything? <Person0> I paid 70 euro for everything. And I thought it would cost me a fortune! I don't even know how much. <Person1> I am glad. <Person0> Me too. 70 euro for everything, including a vase!!!

Summary: Person1 is not at home. Person0 wants Person1 to keep her pasta in the microwave.

Ground Truth Dialog:	**SL-Gen:**	**RL-Gen:**	**CN-Gen:**
<Person0> U both at home? <Person1> No. I've just got ur msg. Why did u ask about it? <Person0> No reason. Keep my pasta in the microwave. <Person1> I haven't cooked anything.	<Person0> How are you doing? <Person1> I'm not at home <Person0> Why? <Person1> I wanted to keep some pasta in the microwave. <Person0> No need to. <Person1> In the microwave? <Person0> Yes.	<Person0> Are you at home? <Person1> No, I'm still at work. <Person0> Good, I should be home in about 15 minutes. <Person1> Can you put my pasta in the microwave? <Person0> Sure, I'll check it. <Person1> Thanks!	<Person0> Are you at home? <Person1> No, I'm busy at work. What do you want? <Person0> I want to put some pasta in microwave. <Person1> OK. I'll try.

Summary: Person0 needs Person1's help as he cannot get the application running.

Ground Truth Dialog:	**SL-Gen:**	**RL-Gen:**	**CN-Gen:**
<Person0> Person1, I need your help? <Person1> What's up, mate? <Person0> I can't get the application running. <Person1> Have you switched the computer on? <Person0> Very funny!!!!!!!!!!!!! <Person1> OK. Sorry. I can see it's serious. <Person0> Yeah, man. It is f**cking serious. <Person1> I'll be with you right now. <Person0> Thanks.	<Person0> Hey! I have a problem with the application. <Person1> Hello, Person0 <Person0> Could you please help me to get the application running? I cannot. <Person1> Why? <Person0> I have a problem with my laptop. The application is running on the USB port. I cannot access the computer. The computer is running. If I try to access the computer, the application will start. I hope that you can help me. <Person1> Okay. <Person0> Thank you, Person1.	<Person0> Person1, you there? <Person1> Yeah, what's up? <Person0> Do you need my help? <Person1> What's wrong? <Person0> I can't get the application running. <Person1> Did you try charging it first? <Person0> Yes, I tried that, but it doesn't work. <Person1> Try refreshing the database. <Person0> Damn, thanks!	<Person0> I cannot get the application running. <Person1> What's up? <Person0> Can you get the application running? <Person1> You sure you can't? Try and click on it. <Person0> You're so funny, Person1. <Person1> OK. I'm coming. <Person0> Thanks.

Summary: Person0 and Person1 will meet the new person in an hour.

Ground Truth Dialog:	**SL-Gen:**	**RL-Gen:**	**CN-Gen:**
<Person0> Can we meet with the new person soon? <Person1> Sure. In an hour okay? <Person0> Perfect.	<Person0> Hi, babe. <Person1> Hi. <Person0> We're meeting in an hour? <Person1> I'd prefer to be there in an hour. <Person0> Great!	<Person0> Did you get that message? <Person1> No, why? <Person0> I might have lost it. <Person1> Oh no, it's my mistake. I'm sorry. <Person0> Ok, don't worry. We can meet in an hour? <Person1> Ok!	<Person0> Wanna meet the new person? <Person1> Sure, I'll be there in an hour. <Person0> Perfect!

Figure 3: Samples of dialogs with their corresponding summaries - ground-truth and automatic generated ones

A Non-Autoregressive Edit-Based Approach
to Controllable Text Simplification

Sweta Agrawal, Weijia Xu, Marine Carpuat
Department of Computer Science
University of Maryland
{sweagraw, weijia, marine}@cs.umd.edu

Abstract

We introduce a new approach for the task of Controllable Text Simplification, where systems rewrite a complex English sentence so that it can be understood by readers at different grade levels in the US K-12 system. It uses a non-autoregressive model to iteratively edit an input sequence and incorporates lexical complexity information seamlessly into the refinement process to generate simplifications that better match the desired output complexity than strong autoregressive baselines. Analysis shows that our model's local edit operations are combined to achieve more complex simplification operations such as content deletion and paraphrasing, as well as sentence splitting.

1 Introduction

Text simplification (TS) aims to automatically rewrite text so that it is easier to read. What makes text simple depends on its target audience (Xu et al., 2015): replacing complex or specialized terms with simpler synonyms might be helpful for non-native speakers (Petersen and Ostendorf, 2007; Allen, 2009) whereas restructuring text into short sentences with simple words might better match the literacy skills of children (Watanabe et al., 2009). Studies of simplification tools for deaf or hard-of-hearing users also show that they prefer lexical simplification to be applied on-demand (Alonzo et al., 2020). Yet, research in TS has mostly focused on developing models that generate a generic simplified output for a given source text (Xu et al., 2015; Zhang and Lapata, 2017; Alva-Manchego et al., 2020). We contrast this Generic TS with Controllable TS which specifies desired output properties.

Prior work has addressed Controllable TS for either high-level properties, such as the target reading grade level for the entire text (Scarton and Specia, 2018; Nishihara et al., 2019), or low-level properties, such as the compression ratio or the nature of the simplification operation to use (Mallinson and Lapata, 2019; Martin et al., 2020; Maddela et al., 2020). Specifying the desired reading grade level might be more intuitive for lay users. However, it provides only weak control over the nature of simplification. As illustrated in Table 1, simplifying text to different grade levels results in diverse edits. To rewrite the grade 10 original for grade 5, the complex text is split into two sentences and paraphrased. When simplifying for grade 3, phrases are further simplified, and content is entirely deleted.

Grade	Text
10	Tesla is a maker of electric cars, which do not need gas and can be charged by being plugged into a wall socket.
5	Tesla cars can be charged by being plugged **in, like a phone. They** do not need any gas.
3	Tesla **builds** cars **that** do not need gas.

Table 1: Simplified text changes depending on the reading grade level of the target audience. The bold font highlights changes compared to the grade 10 version.

In this work, we adopt the intuitive framing for Controllable TS where the desired reading grade level is given as input, while providing fine-grained control on simplification by incorporating lexical complexity signals into our model. We adopt a non-autoregressive sequence-to-sequence model (Xu and Carpuat, 2020) that iteratively refines an input sequence to reach the desired degree of simplification and seamlessly integrate lexical complexity.

Unlike commonly used autoregressive (AR) models for simplification (Specia, 2010; Nisioi et al., 2017; Zhang and Lapata, 2017; Wubben et al., 2012; Scarton and Specia, 2018; Nishihara et al., 2019; Martin et al., 2020; Jiang et al., 2020, among others), our model relies on explicit edit

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3757–3769
August 1–6, 2021.

operations. It therefore has the potential of modeling the simplification process more directly than AR models which need to learn to copy operations implicitly. Unlike existing edit-based models for simplification which rely on pipelines of independently trained components (Alva-Manchego et al., 2017; Malmi et al., 2019; Mallinson et al., 2020), our model is trained end-to-end via imitation learning and thus learns to apply sequences of edits to transform the original source into the final simplified text. Furthermore, our approach does not require a custom architecture for simplification: it repurposes a non-autoregressive (NAR) model introduced for Machine Translation (MT) and can seamlessly incorporate lexical complexity information derived from data statistics in the initial sequence to be refined.

Based on extensive experiments on the Newsela English corpus, we show that our approach generates simplified outputs that match the target reading grade level better than strong AR baselines. Further analysis shows that the model learns complex editing operations such as sentence splitting, substitution and paraphrasing, and content deletion and applies these operations accordingly to match the complexity of the desired grade level.

2 An Edit-based approach for Controllable TS

Task We frame Controllable TS as follows: given a complex text c and a target grade level g_t, the task consists in generating a simplified output s that is appropriate for grade level g_t.

Approach Our approach, illustrated in Figure 1, is based on EDITOR (Xu and Carpuat, 2020), a NAR Transformer model where the decoder layer is used to apply a sequence of edits on an initial input sequence (possibly empty). The edits are of two types: (1) reposition and (2) insertion. The reposition layer predicts the new position of each token (including deletions). The insertion layer has two components: the first layer predicts the number of placeholders to be inserted and the fill-in layer generates the actual target tokens for each placeholder. At each iteration, the model applies a reposition operation followed by insertion to the current input. This is repeated until two consecutive iterations return the same output, or a preset maximum number of operations is reached. We tailor EDITOR for the task of Controllable TS as follows:

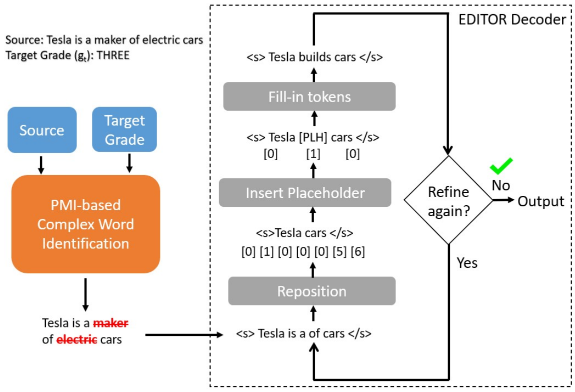

Figure 1: EDITOR iteratively refines a version of the input where words predicted to be too complex for 3rd grade readers have been deleted.

Control tokens The target complexity g_t is encoded as a special token added at the start of the input sequence. As in prior work with autoregressive models (Scarton and Specia, 2018; Nishihara et al., 2019), this token acts as a side-constraint, gets encoded in the encoder hidden states as any other vocabulary token, and informs hypothesis generation through the source-target attention mechanism.

Lexical complexity signals We automatically identify the source words that are too complex for the target grade and delete them from the initial sequence to be refined by EDITOR. This simple strategy provides finer-grained guidance to the simplification process than the sequence-level side-constraint, while leaving the EDITOR model the flexibility to rewrite the output without constraints. We quantify the relatedness between each vocabulary word (w) and grade-level (g) using their Pointwise Mutual Information (PMI) in the newsela corpus (Nishihara et al., 2019; Kajiwara, 2019):

$$PMI(w,g) = log\frac{p(w|g)}{p(w)} \qquad (1)$$

Here, $p(w|g)$ is the probability that word w appears in sentences of grade level g and $p(w)$ is the probability of word w in the entire training corpus.

While the desired grade level g_t is known in the task, we automatically predict the complexity g_s of each source sentence s_i using the Automatic Readability Index (ARI; Senter and Smith (1967)). The initial decoding sequence $\hat{s_i}$ takes the source sequence and deletes all words that are strongly related to the source grade level and unlikely to be found in text of the target grade level, with the exception of named entities:

$$\hat{s_i} = \{w|w \in s_i \wedge \quad \sim (PMI(w, g_s) > 0 \\ \wedge\, PMI(w, g_t) < 0 \wedge \; w \notin E_i)\} \tag{2}$$

where, E_i represents the set of entities in the source sequence s_i. Our approach contrasts with prior work where PMI has been used in the loss to reward the generation of target grade-specific words for Controllable TS (Nishihara et al., 2019) or to exclude complex words from the decoding vocabulary using hard constraints for Generic TS (Kajiwara, 2019). Our approach combines lexical complexity information from both the source and target grade level more flexibly. Starting from $\hat{s_i}$ as an initial sequence, EDITOR can still delete further content to match the target grade level, insert new words to fix fluency and preserve the original meaning, and has the flexibility to re-generate tokens that were incorrectly dropped from the initial sequence.

Training to generate & refine EDITOR uses imitation learning to learn an appropriate sequence of edit operations to generate the output sequence by efficiently exploring the large space of valid edit sequences that can reach a reference output. A roll-in policy is used to generate sequences to be refined and a roll-out policy is then used to estimate cost-to-go for all possible actions given the roll-in sequences. The model is trained to choose actions that minimizes the cost-to-go estimates from the roll-in sequences to the true reference by comparing the model actions to the oracle actions generated by the Levenshtein edit distance algorithm. The roll-in sequences are stochastic mixtures of the initial sequences and outputs of the insertion and reposition modules given an initial sequence. The initial sequence is generated by applying random word dropping (Gu et al., 2019) and random word shuffle (Lample et al., 2018) with a probability of 0.5 and maximum shuffle distance of 3 to either the target sequence for MT tasks (Xu and Carpuat, 2020) or to the source sequence for Automatic Post Editing (Gu et al., 2019). For Controllable TS, we combine both, training EDITOR to generate text based on the corrupted *target sequence* first, and then fine-tuning the model for refinement based on the corrupted *source sequence* next.

3 Experimental Settings

3.1 Data

The Newsela website provides high quality data to study text simplification (Xu et al., 2015). It consists of news articles rewritten by professional editors for students in different grade levels. We use English Newsela samples as extracted by Agrawal and Carpuat (2019) since their process preserves grade level information for each segment. We restrict the length of each segment to be between 5 and 80 resulting in 470k/2k/19k for training, development and test sets respectively. We pre-process the dataset using Moses tools for normalization, and truecasing. We refer to the resulting dataset as **newsela-grade**. We further segment tokens into subwords using a joint source-target byte pair encoding model with $32,000$ operations. We use spacy[1] to identify entities in the source sequence.

3.2 Model configurations

Architecture We adopt the base Transformer architecture (Vaswani et al., 2017) with d_{model} = 512, d_{hidden} = 2048, n_{heads} = 8, n_{layers} = 6, and $p_{dropout}$ = 0.1 for all our models. We add dropout to embeddings (0.1) and label smoothing (0.1). AR models are trained with the Adam optimizer with a batch size of 4096 tokens. Training stops after 8 checkpoints without improvement of validation perplexity. We decode with a beam size of 5 for the AR models. All NAR models are trained using Adam with initial learning rate of 0.0005 and a batch size of 16,000 tokens. We select the best checkpoint based on validation perplexity. Grade side-constraints are defined using a distinct special token for each grade level (from 2 to 12). All models are implemented using the Fairseq toolkit.

Preliminary We establish that our Transformer architecture choice is strong on the more standard Generic TS task, as it performs comparably to the state-of-the-art[2] (Jiang et al., 2020) on the Newsela-Auto corpus (Table 2).[3]

Experimental Conditions We compare our approach, i.e.,“NAR + PMI-based initialization”, described in Section 2 to three auto-regressive baselines for Controllable TS:

1. **AR** is a Transformer model which uses grade

---

[1] `https://spacy.io/`

[2] The Bert-initialized Transformer has parameters d_{model} = 768, d_{hidden} = 3072, n_{heads} = 12, n_{layers} = 12, and $p_{dropout}$ = 0.1. The encoder and decoder follow the BERT-base architecture. The encoder is initialized with a pre-trained checkpoint and the decoder is randomly initialized.

[3] This corpus contains complex-simple pairs extracted from 1,506 articles for training, 188 for validation and 188 for testing for Generic TS.

	SARI	add-F1	keep-F1	del-P
Results as reported in Jiang et al. (2020)				
EditNTS	35.8	2.4	29.4	**75.6**
Transformer-BERT	36.6	4.5	31.0	74.3
AR Transformer (ours)	36.1	3.8	**33.5**	71.1

Table 2: Generic TS Evaluation on Newsela-Auto: our Transformer baseline is comparable to SOTA models.

level tokens as side constraints (Scarton and Specia, 2018).

2. **AR + PMI-based constraints** is an AR Transformer model which incorporates lexical complexity information as hard constraints during decoding (Kajiwara, 2019): complex words are excluded from beam search using the dynamic beam allocation algorithm (Post and Vilar, 2018). While this approach was introduced for Generic TS, we adapt it to Controllable TS by defining hard constraints using the same criteria as for deleting words in initial sequences for EDITOR (Section 2).

3. **AR + PMI weighted loss** (Nishihara et al., 2019) is an AR Transformer model trained with a loss that weights words based on their PMI values with the desired target grade level.

3.3 Automatic Evaluation Metrics

We evaluate the output of the models using the following text simplification evaluation metrics:

SARI (Xu et al., 2016) measures lexical simplification based on the words that are added, deleted and kept by the systems by comparing system output against references and against the input sentence. It computes the F1 score for the n-grams that are added (add-F1). The model's deletion capability is measured by the F1 score for n-grams that are kept (keep-F1) and precision for the deleted n-grams (del-P) [4].

Pearson's correlation coefficient (PCC) measures the strength of the linear relationship between the complexity of our system outputs and the complexity of reference outputs. We estimate the reading grade level of the system outputs and reference text using the ARI score.

Adjacency ARI Accuracy represents the percentage of sentences where the system output grade level is within 1 grade of the reference text according to the ARI score (Heilman et al., 2008).

Mean Squared Error (MSE) between the predicted ARI grade level of the system output and the desired target grade level (Scarton and Specia, 2018; Nishihara et al., 2019).

4 Evaluation of Controllable TS

4.1 Automatic Evaluation

Table 3 summarizes the automatic evaluation of our approach on Controllable TS: `our approach`, "NAR + PMI-based initialization", improves all metrics—SARI, PCC, ARI-accuracy and MSE—compared to the AR baselines. It also outperforms the `AR + PMI-based constraints` baseline across all metrics except MSE which oversimplifies the source text by always deleting the complex tokens, as shown by a decrease in keep-F1 (-6.1) and improved del-P (+4.6). This results in lower MSE but worse PCC and ARI-Accuracy. By contrast, `our approach` uses lexical complexity information to provide an initial canvas and yields simplified sentences that match the desired target complexity better than the AR baselines. This is reflected in the higher SARI obtained by the `PMI-based initialization` baseline relative to the `Source`, which represents outputs generated by deleting complex tokens from the source text and hence by itself is not a well-formed. `AR + PMI weighted loss` performs comparably to the `AR` baseline across all the metrics except PCC, which could be due to PMI values being a relatively noisy signal at the token level during training, especially for the target grade levels where the data is scarce.[5]

We further compare `our approach` with the model that uses the `Oracle-keep` sequence, i.e., tokens from the source sequence that are present in the target sequence. As expected, the oracle significantly outperforms all models that do not have access to the reference, further confirming EDITOR's ability to make good use of the provided initial sequence. More interestingly, our method for identifying grade-specific complex tokens (Equation 2) achieves a recall of 91.3% and precision of

---

[4] `https://github.com/cocoxu/simplification`

[5] We note that the improvement in SARI reported in prior work (Nishihara et al. (2019): +0.15) is within the confidence interval (+0.5) of the `AR` baseline (Table 3).

Model	SARI ↑	keep-F1 ↑	add-F1 ↑	del-P ↑	%PCC ↑	%ARI-Acc ↑	MSE ↓	%Unchanged
Source	22.5	67.7	0.0	0.0	63.2	29.4	4.92	100.0
Reference	91.1	98.9	88.1	86.3	100.0	100.0	1.91	10.6
PMI-based initialization	37.1	60.5	1.3	49.3	61.4	26.4	3.69	3.4
AR	38.7 ±0.5	68.3 ±0.3	4.6 ±0.3	43.2 ±1.4	73.0 ±0.3	37.1 ±0.4	3.39 ±0.24	36.2
+ PMI-based constraints	38.3 ±0.1	62.2 ±0.7	4.9 ±0.3	47.8 ±0.3	69.1 ±0.2	35.0 ±0.5	2.21 ±0.22	12.1
+ PMI weighted loss	38.5 ±0.5	68.2 ±0.3	4.5 ±0.3	42.9 ±1.3	72.4 ±0.2	36.6 ±0.5	3.32 ±0.21	35.9
NAR	39.1 ±0.1	66.7 ±0.1	3.1 ±0.1	47.6 ±0.4	73.1 ±0.1	36.4 ±0.0	3.56 ±0.04	17.7
+ PMI-based initialization **(our approach)**	39.7 ±0.1	66.5 ±0.1	3.5 ±0.1	49.0 ±0.4	73.7 ±0.0	38.1 ±0.1	3.30 ±0.05	16.0
Oracle-keep	41.8 ±0.3	70.0 ±0.1	5.0 ±0.1	50.3 ±0.7	75.6 ±0.3	41.8 ±0.3	2.97 ±0.06	16.9

Table 3: Automatic evaluation results on Newsela-Grade test set: our approach outperforms AR baselines on SARI, PCC and ARI accuracy.

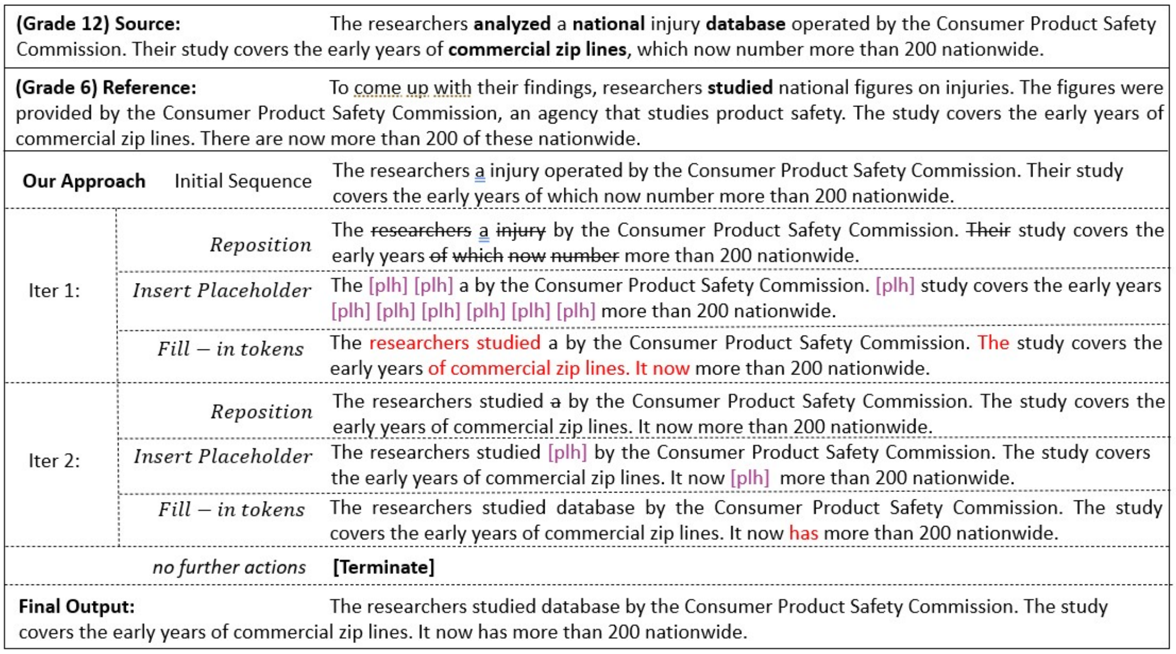

(Grade 12) Source: The researchers **analyzed** a **national** injury **database** operated by the Consumer Product Safety Commission. Their study covers the early years of **commercial zip lines**, which now number more than 200 nationwide.

(Grade 6) Reference: To come up with their findings, researchers **studied** national figures on injuries. The figures were provided by the Consumer Product Safety Commission, an agency that studies product safety. The study covers the early years of commercial zip lines. There are now more than 200 of these nationwide.

Our Approach Initial Sequence: The researchers a injury operated by the Consumer Product Safety Commission. Their study covers the early years of which now number more than 200 nationwide.

Iter 1:

Reposition: The ~~researchers~~ a ~~injury~~ by the Consumer Product Safety Commission. ~~Their~~ study covers the early years ~~of which now number~~ more than 200 nationwide.

Insert Placeholder: The [plh] [plh] a by the Consumer Product Safety Commission. [plh] study covers the early years [plh] [plh] [plh] [plh] [plh] [plh] more than 200 nationwide.

Fill – in tokens: The researchers studied a by the Consumer Product Safety Commission. The study covers the early years of commercial zip lines. It now more than 200 nationwide.

Iter 2:

Reposition: The researchers studied ~~a~~ by the Consumer Product Safety Commission. The study covers the early years of commercial zip lines. It now more than 200 nationwide.

Insert Placeholder: The researchers studied [plh] by the Consumer Product Safety Commission. The study covers the early years of commercial zip lines. It now [plh] more than 200 nationwide.

Fill – in tokens: The researchers studied database by the Consumer Product Safety Commission. The study covers the early years of commercial zip lines. It now has more than 200 nationwide.

no further actions **[Terminate]**

Final Output: The researchers studied database by the Consumer Product Safety Commission. The study covers the early years of commercial zip lines. It now has more than 200 nationwide.

Figure 2: Our approach substitutes "analyzed" correctly as well as splits the source sentence into two simple sentences to generate a simplified output that matches the lexical complexity of the desired grade-level 6. The tokens identified as complex using the proposed method in the source are bold.

76.4% with the oracle on the development set, indicating that the initial sequences contain appropriate vocabulary. Table 3 shows that `our approach` partially closes the gap in performance with the oracle by using this modified source sequence as opposed to the original source sequence (`NAR`).

Figure 2 illustrates the refinement process that generates the simplified output. Reposition and insertion operations are used in consecutive steps to perform complex editing operations (e.g., sentence splitting and lexical substitution) , which requires that the model learns to perform these operations sequentially. Furthermore, `our approach` recovers the tokens that were incorrectly identified as complex and thus deleted in the initial sequence, highlighting the benefits of the flexible refinement process.

4.2 Human evaluation

We randomly sample 60 source sentences from the Newsela-Grade dataset, among sources that are simplified toward four distinct grade levels (∼240 examples). For each of these target grades, we obtain ratings of system outputs and reference from

Model	SARI ↑	%PCC ↑	%ARI-Acc ↑	MSE ↓	Iteration	%Unchanged
our approach	40.2 ±0.0	73.9 ±0.3	36.1 ±0.2	3.73 ±0.03	2.32 ±0.04	17.6
−PMI-based Initialization	39.2 ±0.1	73.1 ±0.9	34.5 ±0.4	4.01 ±0.06	2.23 ±0.06	19.2
−Finetune	37.6 ±0.4	63.7 ±0.5	28.0 ±0.1	5.12 ±0.01	1.14 ±0.00	20.6
−Src Initialization	37.2 ±0.2	72.2 ±0.3	34.4 ±0.4	3.59 ±0.09	2.13 ±0.07	34.1
−Joint	39.7 ±0.1	68.1 ±1.7	31.6 ±0.2	4.70 ±0.05	1.00 ±0.00	11.0
Single Iteration	40.3 ±0.2	72.6 ±0.1	35.7 ±0.5	3.99 ±0.01	1.00 ±0.00	15.0
Gold Source Grade	40.2 ±0.1	74.1 ±0.5	36.5 ±0.3	3.71 ±0.04	2.33 ±0.03	17.3

Table 4: Ablation analysis on model design choices for our approach on Newsela-Grade development set.

	Meaning	Grammar	Simplicity		
	Mean	Mean	Mean	Abs. Diff ↓	Adj. Acc ↑
Reference	2.763	**3.193**	5.325	-	-
AR	**2.803**	3.171	5.157	2.035	0.533
our approach	2.647	3.081	5.310	**1.895**	**0.575**

Table 5: Human Evaluation Results: our approach generates output that match the reference judgements better than the AR baseline.

five Amazon Mechanical Turk workers. Following prior annotation protocols (Jiang et al., 2020), we ask workers to rate outputs on three dimensions: a) is the output grammatical? [0-4] b) to what extent is the meaning expressed in the original sentence preserved in the output? [0-4] and c) how simplified is the output with respect to the original source sentence? [0-10]. Different from prior work, we use a 10-point scale for evaluating simplicity to map the rating resolution to the gold grade differences. The detailed instructions provided to the workers are in the Appendix B.

We compute the absolute difference ("AbsDiff") in the simplicity ratings between the reference and the system output by the same annotator, and aggregate over all examples and all ratings. Table 5 shows that our outputs are closer to the reference according to the simplicity judgements than the AR system outputs. The "Mean" ratings indicate that the two models make different trade-offs: where the AR model under-simplifies the source sentence and preserves the meaning, our approach almost matches the mean simplicity of the reference at the cost of lower meaning preservation. Our outputs are also less grammatical than those of the AR model and the references, probably due to the independence assumptions made by the non-autoregressive model. The Adjacency Accuracy, representing the percentage of system outputs within a difference of one rating with the reference, is also higher for our approach relative to the AR model.

4.3 Ablation Experiments

Table 4 summarizes the impact of the design choices described in Section 2: Removing lexical information (`-PMI-based Initialization`) hurts both SARI and the grade specific metrics. Further, using the baseline EDITOR model that is trained only to generate, without `fine-tuning` for refinement, significantly hurts the performance across the board. In that setting, EDITOR never learns to delete tokens from the source, but only learns to delete tokens inserted by the model. Using EDITOR to generate the output from scratch instead (`-Src Initialization`) recovers the performance on SARI and grade specific metrics but fails to match the performance of our approach. This shows that fine-tuning for refinement and providing initial sequences informed by lexical complexity are both key to the performance of the EDITOR for Controllable TS.

We also compare our approach with the variant of the model that is trained to perform reposition independent of the insertion operation (`-Joint`), similar to Mallinson et al. (2020). Even though this variant is able to match SARI, the difference in grade-specific metrics is significant, showing the benefits of joint training of the insertion and reposition components.

Iterative refinement helps match the target grade better than single step refinement as suggested by ARI Accuracy, MSE and PCC. Figure 3 shows the number of iterations of refinement performed by our approach as the function of desired target grade level: simplifying to lower grade levels (2 or 3) requires on average 1 additional refinement step

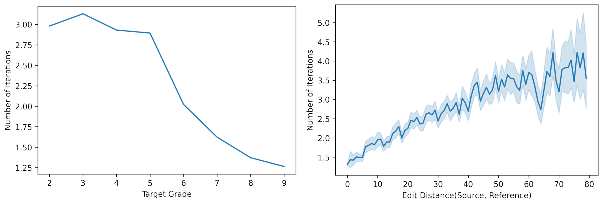

Figure 3: Our approach requires more number of iterations when simplifying to a lower grade level. The number of iterations performed by the model monotonically increases with the edit distance between source and reference.

than simplifiying to grade 8 or 9. This suggests that the iterative process helps simplification when the gap between source and target grades is wider.

Method	**Precision**	**Recall**
Target Only	76.4	80.2
Source (ARI) Only	**76.6**	78.2
Source (ARI) + Target	76.4	**91.3**

Table 6: Using both source and target grade to filter complex words yields maximum overlap with the set of tokens that are preserved from the source in the reference on the Newsela-Grade development set.

Finally, we verify that using ARI to estimate the complexity of the source is effective. Replacing the ARI predictions with the gold-standard grade-level improves the grade-specific metrics only marginally. Table 6 further shows the advantage of combining source and target grade information when identifying complex tokens (Equation 2) over using source or target grade only.

5 Analysis

Per Grade Analysis How does our model compare against the AR baselines for each target grade-levels? Figure 4a and 4b show the SARI and Adjacency Accuracy bucketed by target grade level. We observe that our approach achieves comparable or higher accuracy than the AR baselines for all grades except 2 and 3. Further analysis suggests that this is due to samples where the source grade level is 12, and where our approach deletes words too aggressively to simplify for the large grade gap (Figure 4c and 4d).

Model Edit operations We compare the number of edit operations performed by our model and the oracle Levenshtein Edit Distance (Section 2) when simplifying to different target grade levels. Figure 5 shows that the number of operations performed by our approach to generate its output track the number of oracle Levenshtein edits overall. The main differences are that our approach performs more than twice as many repositions than the oracle (5c) for grades 4 and above which suggest that the sequence of operations performed is suboptimal. Furthermore our approach overdeletes words for target grade levels lower than 4 (5b), and performs fewer insertions than the oracle (5a). We turn to manual analysis to shed more light on these results.

Simplification Operations Table 7 reports a manual annotation of the simplification operations observed for 50 randomly sampled segments, using an operation taxonomy from prior work (Xu et al., 2015; Jiang et al., 2020). Our approach performs content deletion in 7.5% more sentences than needed to generate the references. At the same time, it performs fewer insertions – in particular, our approach is unable to generate the elaborations and explanations found in the Newsela references (Srikanth and Li, 2020). This would require knowledge-based reasoning, which is beyond the capacity of the current model. However, our approach can model sentence splitting and substitution, which often require a sequence of insertion/deletion/reposition operations to be performed sequentially.

Type	% reference	% output
Lexical Substitution	25.0	17.5
Deletion	25.0	32.5
Reordering/Paraphrasing	35.0	20.0
Splitting	27.5	15.0
Content Elaboration	10.0	0.0
Unchanged	22.5	37.5

Table 7: Simplification Operations observed in the reference and output by our approach in 50 randomly sampled examples from the Newsela-Grade dataset.

6 Related Work

AR Models for TS Generic TS is often framed as machine translation where an autoregressive sequence-to-sequence model learns to model simplification operations implicitly from pairs of complex-simple training samples (Specia, 2010; Nisioi et al., 2017; Zhang and Lapata, 2017; Wubben et al., 2012; Scarton and Specia, 2018; Nishihara et al., 2019; Martin et al., 2020; Jiang

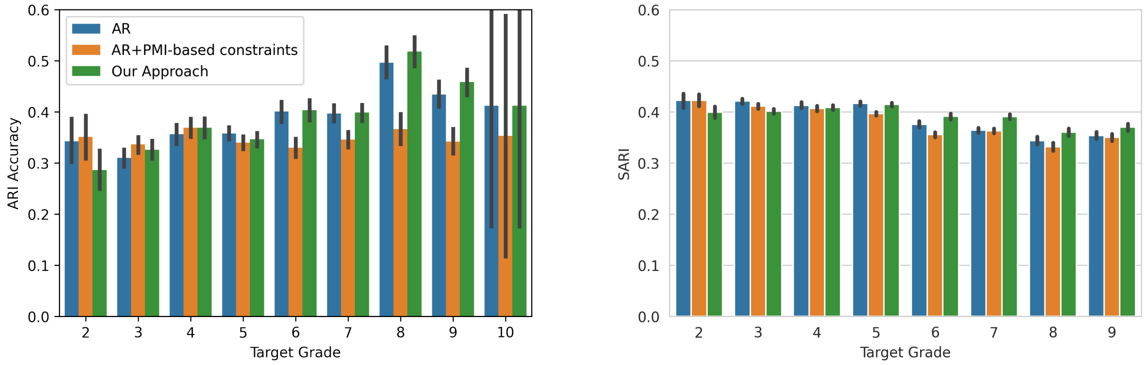

(a) ARI Accuracy (b) SARI

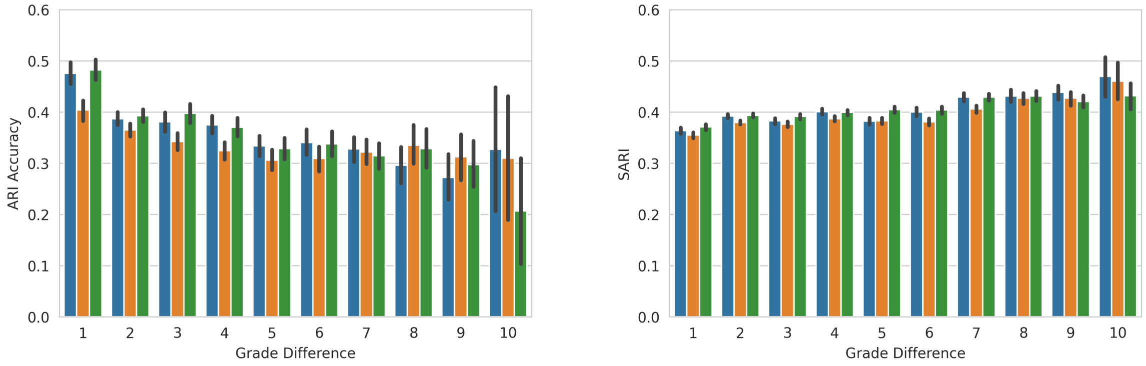

(c) ARI Accuracy (d) SARI

Figure 4: Analysis of automatic metrics for different target grade levels on the Newsela-Grade development set: our approach achieves higher or comparable SARI and ARI scores compared to the AR baselines for all grade levels except 2 or 3.

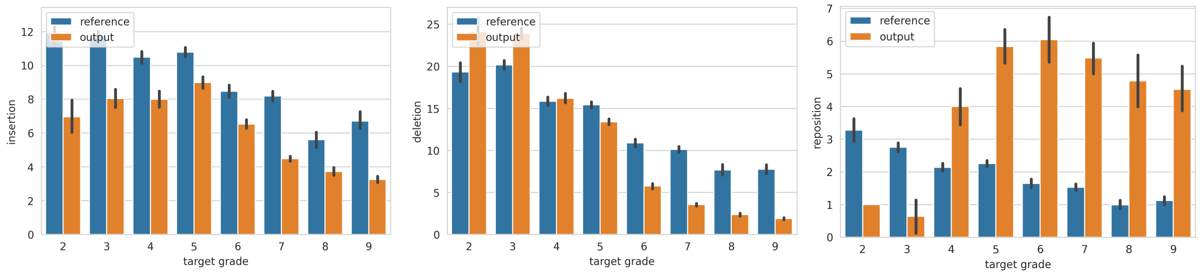

(a) Insertion (b) Deletion (c) Reposition

Figure 5: Edit operations accumulated over iterations for different target grade levels relative to the reference.

et al., 2020). There have been efforts at controlling a different aspect of the simplified output, such as controlling for a specific grade-level (Scarton and Specia, 2018; Nishihara et al., 2019) or employing lexical or syntactic constraints (Mallinson and Lapata, 2019; Martin et al., 2020), where the complexity of a word is either determined by its frequency or by manually tagging the tokens at inference time. We instead use the association of a word with the grade-level to define lexical constraints automatically. Furthermore, these models lack interpretability in terms of the type of operations performed, and need to generate the entire output sequence from scratch thus potentially wasting capacity in learning copying operations.

Edit-based Generic TS Recent work incorporates edit operations into neural text simplifications more directly. These approaches rely on custom multi-step architectures. They first learn to tag the source token representing the type of edit operations to be performed, and then use a secondary model for in-filling new tokens or executing the edit operation. The tagging and editing model are either trained independently (Alva-Manchego et al., 2017; Malmi et al., 2019; Kumar et al., 2020; Mallinson et al., 2020) or jointly (Dong et al., 2019). By contrast, we use a single model trained end-to-end to generate sequences of edit operations to transform the entire source sequence.

Lexical Complexity for TS Nishihara et al. (2019) introduced a training loss for Controllable TS that weights words that frequently appear in the sentences of a specific grade-level. By contrast, we use lexical complexity information to define the initial sequence for refinement, which does not require any change to the model architecture nor to the training process. For Generic TS, Kajiwara (2019) used complex words as negative constrained for decoding with an autoregressive model. By contrast our approach provides more flexibility to the model which results in better outputs in practice.

Non-autoregressive Seq2Seq Models They have primarily been used to speed up Machine Translation by allowing parallel edit operations on the output sequence (Lee et al., 2018; Gu et al., 2018; Ghazvininejad et al., 2019; Stern et al., 2019; Chan et al., 2020; Xu and Carpuat, 2020). Refinement approaches have been used to incorporate terminology constraints in machine translation, including as hard (Susanto et al., 2020) and soft constraints (Xu and Carpuat, 2020). They have also shown promise for Automatic Post Editing (APE) (Gu et al., 2019; Wan et al., 2020) , and grammatical error correction (Awasthi et al., 2019). In this work, we show that they are a good fit to incorporate lexical complexity information for Controllable TS.

7 Conclusion

We introduced an approach that repurposes a non-autoregressive sequence-to-sequence model to incorporate lexical complexity signals in Controllable TS. An extensive empirical study showed that our approach generates simplified outputs that better match the desired target-grade complexity than AR models. Analysis revealed promising directions for future work, such as improving grammaticality while encouraging tighter control on complexity by better aligning the model's atomic edit operations with more complex simplification operations.

Acknowledgments

We thank Eleftheria Briakou, Pranav Goel, Aquia Richburg, Alexander Hoyle, Suraj Nair, Susmija Jabbireddy, the anonymous reviewers and the members of the CLIP lab at UMD for their helpful and constructive comments.

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Sweta Agrawal and Marine Carpuat. 2019. Controlling text complexity in neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1549–1564.

David Allen. 2009. A study of the role of relative clauses in the simplification of news texts for learners of english. *System*, 37(4):585–599.

Oliver Alonzo, Matthew Seita, Abraham Glasser, and Matt Huenerfauth. 2020. Automatic text simplification tools for deaf and hard of hearing adults: Benefits of lexical simplification and providing users with autonomy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA. Association for Computing Machinery.

Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. 2020. Data-driven sentence simplification: Survey and benchmark. *Computational Linguistics*, 46(1):135–187.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4251–4261.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *International Conference on Machine Learning*, pages 1403–1413. PMLR.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402, Florence, Italy. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11179–11189. Curran Associates, Inc.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting grammaticality on an ordinal scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 174–180.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the third workshop on innovative use of NLP for building educational applications*, pages 71–79. Association for Computational Linguistics.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural crf model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Tomoyuki Kajiwara. 2019. Negative lexically constrained decoding for paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6047–6052, Florence, Italy. Association for Computational Linguistics.

Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7918–7928, Online. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Mounica Maddela, Fernando Alva-Manchego, and Wei Xu. 2020. Controllable text simplification with explicit paraphrasing. *arXiv preprint arXiv:2010.11004*.

Jonathan Mallinson and Mirella Lapata. 2019. Controllable sentence simplification: Employing syntactic and lexical constraints. *arXiv preprint arXiv:1910.04387*.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. FELIX: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5057–5068.

Louis Martin, Éric de la Clergerie, Benoît Sagot, and Antoine Bordes. 2020. Controllable sentence simplification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4689–4698, Marseille, France. European Language Resources Association.

Daiki Nishihara, Tomoyuki Kajiwara, and Yuki Arase. 2019. Controllable text simplification with lexical constraint loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 260–266.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91, Vancouver, Canada. Association for Computational Linguistics.

Sarah E Petersen and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Workshop on Speech and Language Technology in Education*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324.

Carolina Scarton and Lucia Specia. 2018. Learning Simplifications for Specific Target Audiences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 712–718.

RJ Senter and Edgar A Smith. 1967. Automated readability index. Technical report, CINCINNATI UNIV OH.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *International Conference on Computational Processing of the Portuguese Language*, pages 30–39. Springer.

Neha Srikanth and Junyi Jessy Li. 2020. Elaborative simplification: Content addition and explanation generation in text simplification. *arXiv preprint arXiv:2010.10035*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with Levenshtein transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3536–3543, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

David Wan, Chris Kedzie, Faisal Ladhak, Marine Carpuat, and Kathleen McKeown. 2020. Incorporating terminology constraints in automatic post-editing. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1193–1204.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Weijia Xu and Marine Carpuat. 2020. Editor: an edit-based transformer with repositioning for neural machine translation with soft lexical constraints. *arXiv preprint arXiv:2011.06868*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

A Appendix

A.1 Dataset Statistics

Table 8 provides the statistics of grade pair distribution in the Newsela-Grade dataset.

A.2 Implementation Details

We train all our models on two GeForce GTX 1080Ti GPUs. The average training time for a single seed of AR model is ∼8-9 hrs and for the EDITOR model is ∼20-22 hrs. Fine-tuning EDITOR takes additional 4-5 hrs.

B Human Annotation

Quality Control We set the location restriction to the United States to control for the quality of annotations. The correlation between the target grade levels and the simplicity ratings of the reference text is 0.582, which suggest that workers do rank simpler output higher than a relatively complex reference of the same source sentence.

Compensation We compensate the Amazon Mechanical Turk workers at a rate of $0.03 per HIT.

Instructions We provide the following instructions to the Amazon Mechanical Turk workers to evaluate generated simplified sentences.

Meaning You are given one sentence and 3 rewrites of the same sentence. Carefully read the instructions provided and then use the sliders to indicate the extent to which the meaning expressed in the original sentence is preserved in the rewrites (Agirre et al., 2016).

Score	Category
4	they convey the same key idea
3	they convey the same key idea but differ in some unimportant details
2	they share some ideas but differ in important details
1	they convey different ideas on the same topic
0	Completely different from the first sentence

Grammar You are given three sentences. Carefully read the instructions provided and then use the sliders to indicate the extent to which each of the sentence is grammatical (Heilman et al., 2014).

Score	Category
4	Perfect: The sentence is native-sounding.
3	Comprehensible: The sentence may contain one or more minor grammatical errors
2	Somewhat Comprehensible: The sentence may contain one or more serious grammatical errors,
1	Incomprehensible: The sentence contains so many errors that it would be difficult to correct
0	Other/Incomplete This sentence is incomplete

Simplicity You are given one sentence and 3 rewrites of the same sentence. Carefully read the instructions provided and use the sliders to indicate how simple is each of the rewrite as compared to the original sentence (0: not simplified at all, 10: most simplified). We provide the following examples for your reference.

Src / Tgt	2	3	4	5	6	7	8	9	10
3	2488	0	0	0	0	0	0	0	0
4	466	7737	0	0	0	0	0	0	0
5	2080	18143	22888	0	0	0	0	0	0
6	1742	6952	20041	20212	0	0	0	0	0
7	545	7857	13556	31297	10315	0	0	0	0
8	557	3277	12557	16301	21457	11241	0	0	0
9	106	4338	4714	18143	4384	28690	2016	0	0
10	6	33	218	306	367	277	386	134	0
11	0	0	15	19	11	16	28	0	0
12	1039	6320	17703	32361	27144	39143	28545	29261	82

Table 8: Number of text segments per grade level pair in the Newsela-Grade dataset.

Original sentence: Craig and April Likhite drove to Chicago from Evanston with their 10-year-old son, Cade, because they wanted to see history made with other fans as close to Wrigley Field as possible.

Rewrites:	**Simplicity:**
1. Craig and April Likhite drove to Chicago. They wanted to see history made with other fans as close to Wrigley Field as possible. *Explanation:* long and complex sentence has been split into two simple sentences, complex words are dropped	7
2. Craig and April Likhite to Chicago with their son Cade. *Explanation:* drastic content deletion	8
3. Craig and April Likhite drove to Chicago from Evanston with their 10-year-old son, Cade. They wanted to see history made with other fans as close to Wrigley Field as possible. *Explanation:* long sentence is split into two simple sentences	4
4. Craig and April Likhite drove to Chicago because they wanted to see history made with other fans as close to Wrigley Field as possible. *Explanation:* paraphrasing and deletion	6

Table 9: Example illustrating ratings for simplified rewrites of an originally complex sentence.

Investigating Transfer Learning in Multilingual Pre-trained Language Models through Chinese Natural Language Inference

Hai Hu[†⁕] **He Zhou**[†] **Zuoyu Tian**[†] **Yiwen Zhang**[†] **Yina Ma**[‡]
Yanting Li[◁] **Yixin Nie**[∘] **Kyle Richardson**[⋄]
[†]Indiana University Bloomington [‡]Brigham Young University [⁕]Shanghai Jiaotong Univ.
[◁]Northwestern University [∘]UNC Chapel Hill [⋄]Allen Institute for AI
`hu.hai@outlook.com; {hzh1,zuoytian,yiwezhan,yinama}@iu.edu;`
`yanting.li@northwestern.edu; yixin1@cs.unc.edu; kyler@allenai.org`

Abstract

Multilingual transformers (XLM, mT5) have been shown to have remarkable transfer skills in zero-shot settings. Most transfer studies, however, rely on automatically translated resources (XNLI, XQuAD), making it hard to discern the particular linguistic knowledge that is being transferred, and the role of expert annotated monolingual datasets when developing task-specific models. We investigate the cross-lingual transfer abilities of XLM-R for Chinese and English natural language inference (NLI), with a focus on the recent large-scale Chinese dataset OCNLI. To better understand linguistic transfer, we created 4 categories of challenge and adversarial tasks (totaling 17 new datasets[1]) for Chinese that build on several well-known resources for English (e.g., *HANS*, NLI *stress-tests*). We find that cross-lingual models trained on English NLI do transfer well across our Chinese tasks (e.g., in 3/4 of our challenge categories, they perform as well/better than the best monolingual models, even on 3/5 uniquely Chinese linguistic phenomena such as *idioms, pro drop*). These results, however, come with important caveats: cross-lingual models often perform best when trained on a mixture of English and high-quality monolingual NLI data (OCNLI), and are often hindered by automatically translated resources (XNLI-zh). For many phenomena, all models continue to struggle, highlighting the need for our new diagnostics to help benchmark Chinese and cross-lingual models.

1 Introduction

Recent pre-trained multilingual transformer models, such as XLM(-R) (Conneau and Lample, 2019; Conneau et al., 2020), mT5 (Xue et al., 2020) and others (Liu et al., 2020; Lewis et al., 2020) have been shown to be successful in NLP tasks for several non-English languages (Khashabi et al., 2020; Choi et al., 2021), as well as in multilingual benchmarks (Devlin et al., 2019; Conneau et al., 2020; Xue et al., 2020; Artetxe et al., 2020). A particular appeal is that they can be used for *cross-lingual* and *zero-shot transfer*. That is, after pre-training on a raw, unaligned corpus consisting of text from many languages, models can be subsequently fine-tuned on a particular task in a resource-rich language (e.g., English) and directly applied to the same task in other languages without requiring any additional language-specific training.

Given this recent progress, a natural question arises: does it make sense to invest in large-scale task-specific dataset construction for low-resourced languages, or does cross-lingual transfer alone suffice for many languages and tasks? A closely related question is: how well do multilingual models transfer across specific linguistic and language-specific phenomena? While there has been much recent work on probing multilingual models (Wu and Dredze, 2019; Pires et al., 2019; Karthikeyan et al., 2019), *inter alia*, a particular limitation is that most studies rely on automatically translated resources such as XNLI (Conneau et al., 2018) and XQuAD (Artetxe et al., 2020), which makes it difficult to discern the particular linguistic knowledge that is being transferred and the role of large-scale, expert annotated monolingual datasets when building task- and language-specific models.

In this paper, we investigate the cross-lingual transfer abilities of XLM-R (Conneau et al., 2020) for Chinese natural language inference (NLI). Our focus on Chinese NLI is motivated by the recent release of the first large-scale, human-annotated Chinese NLI dataset OCNLI (**O***riginal* **C***hinese* **NLI**) (Hu et al., 2020)[2], which we use to directly in-

---

[1] All new datasets/code are released at https://github.com/huhailinguist/ChineseNLIProbing.

[2] To our knowledge, OCNLI is currently the largest non-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3770–3785
August 1–6, 2021. ©2021 Association for Computational Linguistics

	category	n
Chinese HANS	Lexical overlap	1,428
	Subsequence	513
stress tests	Distraction: 2 categories	8,000
	Antonym	3,000
	Synonym	2,000
	Spelling	11,676
	Numerical reasoning	8,613
diagnostics	CLUE (Xu et al., 2020)	514
	CLUE expansion (ours)	796
	World knowledge (ours)	38
	Classifier (ours)	139
	Chengyu/idioms (ours)	251
	Pro-drop (ours)	198
	Non-core arguments (ours)	186
semantic probing	Negation	1,002
	Boolean	1,002
	Quantifier	1,002
	Counting	1,002
	Conditional	1,002
	Comparative	1,002
sum		43,364

Table 1: Summary statistics of the four evaluation sets.

vestigate the role of high-quality task-specific data vs. English-based cross-lingual transfer. To better understand linguistic transfer, and help benchmark recent SOTA Chinese NLI models, we created 4 categories of challenge/adversarial tasks (totaling 17 new datasets) for Chinese that build on several well-established resources for English and the literature on model probing (see Poliak (2020)). Our new resources, which are summarized in Table 1, include: a new set of diagnostic tests in the style of the SuperGLUE (Wang et al., 2019) and CLUE (Xu et al., 2020) diagnostics; Chinese versions of the HANS dataset (McCoy et al., 2019) and NLI stress-tests (Naik et al., 2018), as well as a collection of the basic reasoning and logic *semantic probes* for Chinese based on Richardson et al. (2020).

Our results are largely positive: We find that cross-lingual models trained exclusively on English NLI do transfer relatively well across our new Chinese tasks (e.g., in 3/4 of the challenge categories shown in Table 1, they perform overall as well or better than the best monolingual Chinese models without additional specialized training on Chinese data, and have competitive performance on OCNLI). A particularly striking result is that such models even perform well on 3/5 uniquely Chinese linguistic phenomena such as *idioms, pro drop*, providing evidence that many language-specific phenomena do indeed transfer. These results, however, come with important caveats: on several phenomena we find that models continue to struggle and are far outpaced by conservative estimates of human performance (e.g., our best model on Chinese HANS remains ~19% behind human performance), highlighting the need for more language-specific diagnostics tests. Also, fine-tuning models on mixtures of English NLI data and high-quality monolingual data (OCNLI) consistently performs the best, whereas mixing with automatically translated datasets (XNLI-zh) can greatly hinder model performance. This last result shows that high-quality monolingual datasets still play an important role when building cross-lingual models, however, the particular type of monolingual dataset that is needed can vary and is best informed by targeted behavioral testing of the type we pursue here.

2 Related Work

There has been a lot of work on trying to understand multilingual transformers (Wu and Dredze, 2019; Pires et al., 2019), which has focused on either examining the representation of different layers in the transformer architecture or the lexical overlap between languages. Karthikeyan et al. (2019) investigate the role of network depth and number of attention heads, as well as syntactic/word-order similarity on the cross-lingual transfer performance. In addition to studies cited at the outset, positive results of cross-lingual transfer across a wide range of languages are reported in Wu and Dredze; Nozza et al. (2020), with a focus on transfer across specific tasks such as POS tagging, NER; in contrast, we focus on different categories of linguistic transfer, which has received less attention, as well as the role of monolingual data for transfer in NLI.

Studies into the linguistic abilities and robustness of current NLI models have proliferated in recent years, partly owing to the discovery of systematic biases, or *annotation artifacts* (Gururangan et al., 2018; Poliak et al., 2018), in benchmark NLI datasets such as SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018). This has been coupled with the development of new *adversarial tests* such as *HANS* (McCoy et al., 2019) and the NLI *stress-tests* (Naik et al., 2018), as well as several new linguistic *challenge datasets* (Glockner et al., 2018; Richardson et al., 2020; Geiger et al., 2020; Yanaka et al., 2019; Saha et al., 2020; Goodwin et al., 2020), *inter alia*, that focus on a wide range of linguistic and reasoning phenomena. All of this

---

English NLI dataset that was annotated in the style of English MNLI without any translation.

work focuses exclusively on English, whereas we focus on constructing analogous probing datasets tailored to Chinese to help advance research on Chinese NLI and cross-lingual transfer.

There has been a surge in the development of NLI resources for languages other than English. Such resources are often created in the following two ways: (1) from *scratch*, in the style of MNLI (Williams et al., 2018), where annotators are used to produce hypotheses and inference labels based on a provided set of premises, as pursued for Chinese OCNLI (Hu et al., 2020), or SciTail (Khot et al., 2018), where sentences are paired automatically and labeled by annotators (Amirkhani et al., 2020; Hayashibe, 2020). (2) Through *automatic* (Conneau et al., 2018; Budur et al., 2020; Real et al., 2020) or *manual* (Wijnholds and Moortgat, 2021) translation from existing English datasets. Studies on cross-lingual transfer for NLI have largely focused on XNLI (Conneau et al., 2018), which we show has limited utility for Chinese NLI transfer.

3 Dataset creation

In this section, we describe the details of the 4 types of challenge datasets we constructed for Chinese to study cross-lingual transfer (see details in Table 1). They fit into two general categories: **Adversarial datasets** (Section 3.1) built largely from patterns in OCNLI (Hu et al., 2020) and XNLI (Conneau et al., 2018) and **Probing/diagnostic datasets** (Section 3.2), which are built from scratch in a parallel fashion to existing datasets in English.

While we aim to mimic the annotation protocols pursued in the original English studies, we place the additional methodological constraint that each new dataset is vetted, either through human annotation using a disjoint set of Chinese linguists, or through internal mediation among local Chinese experts; details are provided below.

3.1 Adversarial dataset

Examples from the 7 adversarial tests we created are illustrated in Table 2.[3] Chinese HANS is built from patterns extracted in the large-scale Chinese NLI dataset OCNLI (Hu et al., 2020), whereas the **Distraction**, **Antonym**, **Synonym** and **Spelling** subsets are built from an equal mixture of OCNLI and XNLI-zh (Conneau et al., 2018) data; in the latter case, such a difference allows us to fairly compare the effect of training on expert-annotated (i.e., OCNLI) vs. automatically translated data (i.e., XNLI-zh) as detailed in Section 4.

Chinese HANS McCoy et al. (2019) discovered systematic biases/heuristics in the MNLI dataset, which they named "lexical/subsequence/constituent" overlap. "Lexical overlap" is defined to be the pairs where the vocabulary of the hypothesis is a subset of the vocabulary of the premise. For example, "*The boss is meeting the client.*" and "*The client is meeting the boss.*", which has an entailment relation. However, lexical overlap does not necessarily mean the premise will entail the hypothesis, e.g., "*The judge was paid by the actor.*" does not entail "*The actor was paid by the judge.*" (examples from McCoy et al. (2019)). Thus a model relying on the heuristic will fail catastrophically in the second case.

Inspired by the English HANS, we examine whether OCNLI also possesses such biases, as it has a similar annotation procedure as MNLI. We follow the design of the original HANS experiments, and adapt their scripts[4] to extract examples in OCNLI that satisfy the two heuristics. We find a heavy bias towards "entailment", where 79.5% of such examples are "entailment", similar to MNLI. To construct a Chinese HANS, we first look into syntactic structures of the examples having the two heuristics. Then we write 29 templates for the *lexical overlap* heuristic and 11 templates for *subsequence overlap*.[5] Using the templates and a vocabulary of 263 words, we generated 1,941 NLI pairs. See Table 2 for examples and Appendix A for details.

Distraction We add distractions to the premise or hypothesis, similar to the "length mismatch" and "word overlap" conditions in the NLI *stress tests* of Naik et al. (2018). The distractions are either tautologies ("true is not false") or a true statement from our world knowledge ("Finnland is not a permanent member of the UN security council"), which should not influence the inference label. We control whether the distraction contain a negation or not, and thus create four conditions: *premise-negation*, *premise-no-negation*, *hypothesis-negation*, and *hypothesis-no-negation*. See Table 2 for examples.

---

[3] A more detailed description of the data creation process can be found in Appendix A.

[4] `https://github.com/tommccoy1/hans`

[5] For details of the templates, see our Github repository.

	category	n	premise	hypothesis	label
Chinese HANS	Lexical overlap	1428	我们把银行职员留在电影院了。We left the bank clerk in the cinema.	银行职员把我们留在电影院了。The bank clerk left us in the cinema.	C
	Subsequence	513	谁说律师都是穿西装的。Who told you that all lawyers wear suits.	律师都是穿西装的。All lawyers wear suits.	C
stress tests	Distraction (add to premise)	4000	国有企业改革的思路和方针政策已经明确,而且刚做完手术出院的病人不应剧烈运动。The policy of the reform of state-owned enterprises is now clear, and patients who just had surgery shouldn't have intense exercise.	根本不存在国有企业。The state-owned enterprises don't exist.	C
	Distraction (add to hypothesis)	4000	这时李家院子挤满了参观的人。During this time, the Li family's backyard is full of people who came to visit.	这地方有个姓李的人家,而且真的不是假的。There is a Li family here, and true is not false.	E
	Antonym	3000	一些地方财政收支矛盾较大。The disagreement about local revenue is relatively big.	一些地方财政收支矛盾较小。The disagreement about local revenue is relatively small.	C
	Synonym	2000	海部组阁困难说明了什么。What can you tell from the difficulties from Kaifu's attempt to set up a cabinet?	海部组阁艰难说明了什么。What can you tell from the hardships from Kaifu's attempt to set up a cabinet?	E
	Spelling	2980	身上裹一件工厂发的棉大衣,手插在袖筒里。(Someone is) wrapped up in a big cotton coat the factory gave with hands in the sleeves	身上质少一件衣服。There's at least [typo] one coat on the body.	E
	Numerical reasoning	8613	小红每分钟打不到510个字。Xiaohong types fewer than 510 words per min.	小红每分钟打110个字。Xiaohong types 110 words per min.	N

Table 2: Example NLI pairs in Chinese HANS and stress tests with translations.

Antonym We replace a word in the premise with its antonym to form a contradiction. To ensure the quality of the resulting NLI pairs, we manually examine the initially generated data and decided to only replace nouns and adjectives, as they are more likely to produce real contradictions.

Synonym We replace a word in the premise with its synonym to form an entailment.

Spelling We replace one random character in the hypotheses with its homonym (character with the same *pinyin* pronunciation ignoring tones) as this is one of the most common types of misspellings in Chinese.

Numerical reasoning We create a probing set for numerical reasoning, following simple heuristics such as the following. When the premise is *Mary types x words per minute*, the entailed hypothesis can be: *Mary types less than y words per minute*, where $x < y$. A contradictory hypothesis: *Mary types y words per minute*, where $x > y$ or $x < y$. Then a neutral pair can be produced by reversing the premise and hypothesis of the above entailment pair. 4 heuristic rules (with 6 words for quantification) are used and the seed sentences are extracted from Ape210k (Zhao et al., 2020), a dataset of Chinese elementary-school math problems. The resulting data contains 8,613 NLI pairs.

For **quality control** and to compute human performance, we randomly sampled 50 examples from all subsets and asked 5 Chinese speakers to verify. Our goal is to mimic the human annotation protocol from Nangia and Bowman (2019), which gives us a *conservative* estimate of human performance given that our annotators received very little instructions. Their majority vote agrees with the gold label 90.0% of the time, which suggests that our data is of high quality and allows us to later compare against model performance.[6]

3.2 Probing/diagnostic datasets

While the Chinese HANS and stress tests are designed to adversarially test the models, we also create probing or diagnostic datasets which are aimed at examining the models' linguistic and reasoning abilities.

Hand-crafted diagnostics We expanded the diagnostic dataset from the Chinese NLU Benchmark (CLUE) (Xu et al., 2020) in the following two ways:

First, 6 Chinese linguists (PhD students) created diagnostics for 4 Chinese-specific linguistic phenomena. Here are two of the phenomena:[7] (1) *pro-drop*: subjects or objects in Chinese can be dropped when they can be recovered from the context (Li et al., 1981). Thus the model needs to figure out the subject/object from the context. (2) *four-character idioms* (i.e., 成语 *Chengyu*). They are a special type of Chinese idioms that has exactly four characters, usually with a figurative meaning different from the literary meaning, e.g., 打草惊蛇 *hit hay startle snake* (behaving carelessly and causing your enemy to become vigilant). We construct examples to test whether models understand the figurative meaning in the idioms. Specifically, we first create a premise $\mathcal{P}$ which includes the idiom, where there is enough context so that a human is highly likely to

---

[6] Specifically: 98.0% on Chinese HANS, 86.0% on the stress tests. For comparison, different subsets of the English stress tests receives 85% to 98% agreement (Naik et al., 2018).

[7] For the other two, please refer to Appendix A.

interpret the idiom figuratively. Then we create an entailed hypothesis that is based on the figurative (correct) interpretation, and a neutral/contradictory hypothesis that uses the literal (incorrect) meaning (see Table 11 in the Appendix for an example). For each $\mathcal{P}$ we write 3 hypothesis, one for each inference relation. We also added diagnostics involving world knowledge.

Second, we double the number of diagnostic pairs for all 9 existing linguistic phenomena in CLUE with pairs whose premises are selected from a large news corpus[8] and hypotheses are handwritten by our linguists, to accompany the 514 artificially created data in CLUE. The resulting new diagnostics is 4 times as large as the original one, with a total of 2,122 NLI pairs. For quality control, each pair is double-checked by local Chinese linguists not involved in this study and the controversial cases were discarded after a discussion among the 6 linguists. See Table 11 in Appendix A for examples.

Semantic fragments Following Richardson et al. (2020) and Salvatore et al. (2019), we design synthesized fragments to examine models' understanding ability of six types of linguistic and logic inference: **boolean**, **comparative**, **conditional**, **counting**, **negation** and **quantifier**, where each category has 2-4 templates. See example templates and NLI pairs in Table 3.

The data is generated using context-free grammar rules and a vocabulary of 80,000 person names (Chinese and transliterated), 8659 city names and expanded predicates and comparative relations in Richardson et al. (2020) to make the data more challenging. As a result, we generated 1,000 examples for each fragment. For quality control, each template was checked by 3 linguists/logicians; also 20 examples from each category were checked for correctness by local experts.

4 Experimental setup

Our main goal is to test whether cross-lingual transfer are robust against the adversarial and probing data we created when evaluated without additional training. Thus we need to compare the best Chinese monolingual models with the best multilingual models trained either on English NLI data alone, or on combinations of Chinese and English data.[9]

Chinese monolingual models We experimented with two current state-of-the-art transformer models: RoBERTa-large (Liu et al., 2019) and Electra-large-discriminator (Clark et al., 2019). We use the Chinese models released from (Cui et al., 2020)[10] implemented the Huggingface Transformer library (Wolf et al., 2020).

Multilingual model We use XLM-RoBERTa-large (Conneau et al., 2020). We choose XLM-R over mT5 (Xue et al., 2020) because XLM-R generally performs better than mT5 under the same model size (see original paper for details). Also, XLM-R as a RoBERTa model is most related architecturally to existing Chinese pre-trained models.

Fine-tuning data for Chinese models & XLM-R (1) XNLI: the full Chinese training set in the machine-translated XNLI dataset, with 390k examples (Conneau et al., 2018). (2) XNLI-small: 50k examples from XNLI, the same size as the training data of OCNLI. (3) OCNLI: Original Chinese NLI dataset (Hu et al., 2020). It is a Chinese NLI dataset collected from scratch, following the MNLI procedure, with 50k training examples. We use this to measure the effect of the quality of training data; that is, whether it is better to use small, high-quality training data (OCNLI), or large, low-quality MT data (XNLI). (4) OCNLI + XNLI: a combination of the two training sets, 440k examples.

Fine-tuning data for XLM-R To examine cross-lingual transfer, we finetune XLM-R on English NLI data alone and English + Chinese NLI data: (1) MNLI: 390k examples from MNLI.train (Williams et al., 2018). (2) English all NLI: we combine MNLI (Williams et al., 2018), SNLI (Bowman et al., 2015), FeverNLI (Thorne et al., 2018; Nie et al., 2019) with ANLI (Nie et al., 2020), a total of 1,313k examples. (3) OCNLI + English all NLI. (4) XNLI + English all NLI. These two are set to examine whether combining Chinese and English fine-tuning data is helpful.

---

[8] We use the BCC corpus (Xun et al., 2016): http://bcc.blcu.edu.cn/.

[9] We also run the same experiments for Chinese-to-English transfer, i.e., fine-tuning XLM-R with OCNLI and evaluate on the four English counterpart datasets. We find that transferring from OCNLI to English does not perform as well as monolingual English models, likely due to the small size of OCNLI. Detailed results are reported in Appendix C.

[10] We use `hfl/chinese-roberta-wwm-ext-large` from https://github.com/ymcui/Chinese-BERT-wwm and `hfl/chinese-electra-large-discriminator` from https://github.com/ymcui/Chinese-ELECTRA.

category	premise	hypothesis	label
Negation	库尔图尔只到过湛江市麻章区，丰隆格只到过大连市普兰店区…… $person_1$ only went to $location_1$; $person_2$ only went to $location_2$;	库尔图尔没到过大连市普兰店区。 $person_1$ has not been to $location_2$.	E
Boolean	何峥、管得宽、李国柱……只到过临汾市襄汾县。 $person_1$, $person_2$ ... have only been to $location_1$.	何峥没到过遵义市红花岗区。 $person_1$ has not been to $location_2$.	E
Quantifier	有人到过每一个地方，拥抱过每一个人。 Someone has been to every place and hugged every person.	王艳没拥抱过包一。$person_1$ hasn't hugged $person_2$.	N
Counting	韩声雄只拥抱过罗冬平、段秀芹……赵常。 $person_1$ only hugged $person_2$, $person_3$ ... $person_8$.	韩声雄拥抱过超过10个人。 $person_1$ hugged more than 10 people.	C
Conditional	……，穆肖贝夸到过赣州市定南县，如果穆肖贝夸没到过赣州市定南县，那么张本伟到过呼伦贝尔市阿荣旗。... $person_n$ has been to $location_n$. If $person_n$ hasn't been to $location_n$, then $person_m$ has been to $location_m$.	张本伟没到过呼伦贝尔市阿荣旗。$person_m$ hasn't been to $location_m$.	N
Comparative	龙银凤比武书瑾、卢耀辉……奈德哈特都小，龙银凤和亚厄纳尔普一样大。$person_1$ is younger than $person_2$, ..., $person_n$; $person_1$ is as old as $person_m$	亚厄纳尔普比梁培娟大。$person_m$ is older than $person_{n-2}$.	C

Table 3: Example NLI pairs for semantic/logic probing with translations. Each label for each category has 2 to 4 templates; we are only showing 1 template for 1 label. 1,000 examples are generated for each category.

Model	Fine-tuned on	Acc	Scenario
RoBERTa	zh MT: XNLI-small	67.44	monolingual
RoBERTa	zh MT: XNLI	70.29	monolingual
RoBERTa	zh ori: OCNLI	79.11	monolingual
RoBERTa	zh: OCNLI + XNLI	78.43	monolingual
XLM-R	zh MT: XNLI	72.55	monolingual
XLM-R	zh ori: OCNLI	79.24	monolingual
XLM-R	zh: OCNLI + XNLI	80.31	monolingual
XLM-R	en: MNLI	71.98	zero-shot
XLM-R	en: En-all-NLI	73.73	zero-shot
XLM-R	mix: OCNLI + En-all-NLI	**82.18**	mixed
XLM-R	mix: XNLI + En-all-NLI	74.12	mixed

Table 4: Results on OCNLI dev. "**Scenario**" indicates whether the model is fine-tuned on Chinese *only* data (**monolingual**), English data (**zero-shot**) or **mixed** English and Chinese data; results in gray show best performance for each scenario. Best overall result in **bold**. Same below.

We fine-tune the models on OCNLI-dev. Acknowledging that different training runs can produce very different checkpoints for behavioral testing (D'Amour et al., 2020), we run 5 models on different seeds and report the mean accuracy of the models with the best hyper-parameter setting (for details see Appendix B).

5 Results and discussion

5.1 Results on OCNLI_dev

Results on the dev set of OCNLI are presented in Table 4. For monolingual RoBERTa, we see a similar performance as reported in the OCNLI paper (Hu et al., 2020), with 79.11% accuracy. The monolingual Electra achieves a very close accuracy of 79.02% (not shown in the Table). As we see the same trend in the following experiments, we will therefore only report results on RoBERTa.

For XLM-R, fine-tuning on MNLI or En-all-NLI gives us reasonable results of around 72% to 74%, which is better than models fine-tuned on XNLI, indicating that fine-tuning on an English data (MNLI) alone can outperform monolingual models fine-tuned on the same data but machine-translated into Chinese (XNLI).[11] This is consistent with previous results on Korean (Choi et al., 2021) and Persian (Khashabi et al., 2020) for other NLU tasks.

What is also interesting is that combining OCNLI and En-all-NLI gives us a boost of 2% to 82.18% (a result that is comparable to the current published SOTA), showing the power of mixing high-quality English and Chinese training data.

5.2 Chinese HANS

Table 5 shows results of the Chinese HANS data tested on the aforementioned monolingual models and cross-lingual model.

Cross-lingual transfer achieves strong results. We first notice that when XLM-R is fine-tuned solely on the English data (En-all-NLI), the performance (~69%) is only slightly worse than the best monolingual model (~71%). This suggests that cross-lingual transfer from English to Chinese is quite successful for an adversarial dataset like HANS. Second, adding OCNLI to En-all-NLI in the training data gives a big boost of about 9%, and achieves the overall best result. This is about 12% higher than combining XNLI and the English data, demonstrating the advantage of the expert-annotated OCNLI over machine translated XNLI, even though the latter is about 8 times the size of the former. Despite these results, however, we note that all models continue to perform below human performance, suggesting more room for improvement.

Our results also suggest that examples involving the *sub-sequence* heuristics are more difficult than

---

[11] For these experiments we also tested with another Chinese machine-translated MNLI (CMNLI), translated by a different MT system, which was released by CLUE (https://github.com/CLUEbenchmark/CLUE), and obtained similar results.

Model	Fine-tuned on	Overall	Lexical Overlap	Sub-sequence	Entailment	Non-Entailment	Δ
RoBERTa	zh MT: XNLI-small	49.48	58.12	25.42	99.22	30.26	37.18
RoBERTa	zh MT: XNLI	60.80	68.99	38.01	99.74	45.76	24.53
RoBERTa	zh ori: OCNLI	71.72	75.39	61.48	99.67	60.91	18.20
RoBERTa	zh: OCNLI+XNLI	69.33	74.73	54.27	99.89	57.51	20.92
XLM-R	zh ori: OCNLI	61.82	65.83	50.68	99.89	47.11	32.13
XLM-R	zh MT: XNLI	57.74	66.47	33.45	99.96	41.42	31.13
XLM-R	zh: OCNLI+XNLI	70.31	74.25	59.34	100.00	58.84	21.47
XLM-R	en: En-all-NLI	69.56	77.62	47.13	100.00	57.80	15.93
XLM-R	en: MNLI	66.74	73.12	48.97	100.00	53.89	18.09
XLM-R	mix: OCNLI+En-all-NLI	**78.82**	**81.57**	**71.15**	**100.00**	**70.63**	11.55
XLM-R	mix: XNLI+En-all-NLI	66.90	76.25	40.90	99.93	41.89	32.23
Human		98.00					

Table 5: Accuracy on Chinese HANS. Δ = the difference of accuracy between OCNLI dev and Non-Entailment.

those targeting the *lexical overlap* heuristics for the transformers models we tested (see the "sub-sequence" and "lexical overlap" columns in Table 5). This is in line with the results reported in the English HANS paper (specifically Table 15 in McCoy et al. (2019) which also shows that the sub-sequence examples are more difficult for the English BERT model). Second, for the sub-sequence heuristics, results from monolingual model are 12% higher than those from XLM-R under the zero-shot transfer setting (61.48% versus 48.79% in "sub-sequence" column in Table 5). This stands in contrast with the lexical overlap heuristic, where the best monolingual model performs similarly to the best zero-shot cross-lingual transfer (75.39% versus 77.62%). This is one of the few cases where cross-lingual transfer under-performs the monolingual setting by a large margin, suggesting that in certain situations monolingual models may be preferred.

5.3 Stress tests

Table 6 presents the accuracies on all the stress tests. We first see that cross-lingual zero-shot transfer using all English NLI data performs even better than the best monolingual model (~74% vs. ~71%). This demonstrates the power of the cross-lingual transfer-learning. Adding OCNLI to all English NLI gives another increase of about 3 percentage points (to 77%), while adding XNLI hurts the performance, again showing the importance of having expert-annotated language-specific data.

Antonyms and Synonyms All models except those fine-tuned on OCNLI achieved almost perfect score on the synonym test. However, for antonyms, both monolingual and multilingual models fine-tuned with OCNLI perform better than XNLI. XLM-R fine-tuned with English NLI data only again outperforms the best of monolingual models (~80% vs. ~72%). Interestingly, adding XNLI to all English NLI data hurts the accuracy badly (a 14% drop), while adding OCNLI to the same English data improves the result slightly.

As antonyms are harder to learn (Glockner et al., 2018), we take our results to mean that either expert-annotated data for Chinese or a huge English NLI dataset is needed for a model to learn decent representations about antonyms, as indicated by the high performance of RoBERTa fine-tuned with OCNLI (71.81%), and XLM-R fine-tuned with En-all-NLI (80.36%), on antonyms. That is, using machine-translated XNLI will not work well for learning antonyms (~55% accuracy).

Distraction Results in Table 6 show that adding distractions to the hypotheses has a more negative negative impact on models' performance, compared with appending distractions to the premises. The difference is about 20% for all models (see "Distr H" columns and "Distr P" columns in Table 6), which has not been reported in previous studies, to the best of our knowledge. Including a negation in the hypothesis makes it even more challenging, as we see another one percent drop in the accuracy for all models. This is expected as previous literature has demonstrated the key role negation plays when hypotheses are produced by the annotators (Poliak et al., 2018).

Spelling This is another case where cross-lingual transfer with English data alone falls behind monolingual Chinese models (by about 4%). Also the best results are from fine-tuning XLM-R with OCNLI + XNLI, rather than a combination of English and Chinese data. Considering the data is created by swapping Chinese characters with others of the same pronunciation, we take it to suggest

Model	Fine-tuned on	Overall	Ant.	Syn.	Distr H	Distr H-n	Distr P	Distr P-n	Spell-ing	num.
RoBERTa	zh MT: XNLI-small	59.41	43.38	99.64	51.61	51.41	70.66	71.19	69.93	28.70
RoBERTa	zh MT: XNLI	66.22	52.28	99.79	54.83	53.8	74.55	74.57	72.22	53.53
RoBERTa	zh ori: OCNLI	64.49	71.81	73.66	52.95	51.8	73.43	73.86	71.79	54.16
RoBERTa	zh: OCNLI + XNLI	71.01	59.39	99.06	55.87	54.64	76.83	76.50	75.48	70.18
XLM-R	zh ori: OCNLI	69.08	71.29	88.63	55.93	55.05	76.84	77.00	71.42	65.51
XLM-R	zh MT: XNLI	66.87	55.53	**99.96**	56.11	55.29	77.69	77.9	74.37	46.81
XLM-R	zh: OCNLI + XNLI	71.49	61.85	99.45	58.15	57.92	79.16	79.28	**77.93**	61.88
XLM-R	en:MNLI	67.94	65.77	99.2	55.14	54.6	75.75	75.76	70.76	50.90
XLM-R	en: En-all-NLI	74.52	80.36	97.58	54.74	53.56	73.96	73.92	71.02	82.73
XLM-R	mix: OCNLI + En-all-NLI	**77.36**	**81.93**	95.09	**59.23**	**58.00**	**79.88**	**79.92**	74.53	**87.77**
XLM-R	mix: XNLI + En-all-NLI	73.57	66.15	99.68	57.02	55.51	78.38	78.53	75.15	80.33
Human		85.00	85.00	98.00	83.00	83.00	83.00	83.00	78.00	98.00

Table 6: Accuracy on the stress test. Distr H/P(-n): distraction in Hypothesis/Premise (with negation).

Model	Fine-tuned on	Overall	⋆ Classifier	⋆ Idioms	⋆ Non-core arguement	⋆ Pro-drop	⋆ Time of event	Anaphora	Argument structure	Common sense	Comparatives	Double negation	Lexical semantics	Monotonicity	Negation	World knowledge
RoBERTa	zh MT: XNLI-small	62.9	65.8	64.7	55.2	80.5	60.0	59.6	67.4	54.3	61.4	48.3	60.9	59.7	66.2	39.0
RoBERTa	zh MT: XNLI	67.7	67.6	66.2	59.4	82.3	65.1	69.9	72.0	56.8	70.4	64.2	67.5	61.7	72.9	52.1
RoBERTa	zh ori: OCNLI	67.8	62.0	68.0	59.4	80.7	77.5	70.3	70.0	56.0	66.6	64.2	68.4	61.7	72.4	57.9
RoBERTa	zh: OCNLI + XNLI	69.3	66.3	67.1	58.6	83.0	74.0	70.1	73.5	54.9	74.1	67.5	69.1	62.5	76.0	60.0
XLM-R	zh ori: OCNLI	68.0	57.6	70.1	58.0	79.6	76.3	67.4	70.3	55.3	69.8	75.8	71.1	62.5	71.1	62.1
XLM-R	zh MT: XNLI	60.9	61.2	62.3	50.4	71.9	59.7	60.3	63.3	51.7	65.2	54.9	61.0	53.5	66.9	58.3
XLM-R	zh: OCNLI + XNLI	71.5	70.4	71.6	57.5	84.6	77.8	74.5	74.7	55.3	75.5	76.7	72.8	62.7	76.3	65.3
XLM-R	en: MNLI	70.2	70.1	73.9	57.5	86.4	70.8	69.3	72.9	48.9	76.0	62.5	67.8	62.6	77.0	62.1
XLM-R	en: En-all-NLI	71.9	71.8	**74.3**	56.2	87.4	75.7	74.9	74.8	49.1	80.5	70.8	69.1	63.8	77.8	64.2
XLM-R	mix: OCNLI + En-all-NLI	**74.9**	**72.7**	**74.3**	60.1	**88.5**	**84.5**	**77.3**	**78.1**	56.6	**81.3**	**79.2**	**77.2**	65.6	**78.0**	67.9
XLM-R	mix: XNLI + En-all-NLI	71.4	70.2	58.5	**85.5**	71.3	75.2	75.5	55.1	**79.2**	70.0	69.1	62.4	**76.2**	72.1	**71.3**

Table 7: Accuracy on the expanded diagnostics. Uniquely Chinese linguistic features are prefixed with ⋆.

that monolingual models are still better at picking up the misspellings or learning the connections between characters at the phonological level.

Numerical Reasoning Results in the last column of Table 6 suggest a similar pattern: using all English NLI data for cross-lingual transfer outperforms the best monolingual model. However, fine-tuning a monolingual model with the small OCNLI (50k examples, accuracy: 54%) achieves better accuracy than using a much larger MNLI (390k examples, accuracy: 51%) for cross-lingual transfer, although both are worse than XLM-R fine-tuned with all English NLI which has more than 1,000k examples (accuracy: 83%). This suggests that there are cases where a monolingual setting (RoBERTa with OCNLI) is competitive against zero-shot transfer with a large English dataset (XLM-R with MNLI). However, that competitiveness may disappear when the English dataset grows to an order of magnitude larger in size or becomes more diverse (En-all-NLI contains 4 different English NLI datasets).

5.4 Hand-written diagnostics

Results on the expanded diagnostics are presented in Table 7. We first see that XLM-R fine-tuned with only English performs very well, at 70.2% and 71.9%, even slightly higher than the best monolingual Chinese model (69.3%).

Most surprisingly, **in 3/5 categories with uniquely Chinese linguistic features, zero-shot transfer outperforms monolingual models**. Only in "non-core arguments" and "time of event" do we see higher performance of OCNLI as the fine-tuning data. What is particularly striking is that for "idioms (*Chengyu*)", XLM-R fine-tuned only on English data achieves the best result, suggesting that the cross-lingual transfer is capable of learning meaning representation beyond the surface lexical information, at least for many of the idioms we tested. The overall results (accuracy of 74.3%) indicate that cross-lingual transfer is very successful in most cases. Manual inspection of the results shows that for many NLI pairs with idioms, XLM-R correctly predicts the figurative interpretation of the idiom as entailment, and the literal interpretation as non-entailment, as described in section 3.2. Looking at OCNLI and XNLI, we observe that they perform similarly when fine-tuned on monolingual RoBERTa. However, when fine-tuned with XLM-R, OCNLI has a clear advantage (68.0% versus 60.9%), suggesting that OCNLI may produce more

model	finetune on	overall	boolean	comparative	conditional	counting	negation	quantifier
RoBERTa	zh MT: XNLI-small	46.57	32.81	34.41	61.48	81.82	33.27	35.63
RoBERTa	zh MT: XNLI	50.64	33.35	39.02	66.55	84.51	40.92	39.50
RoBERTa	zh ori: OCNLI	47.53	35.81	34.81	62.87	69.64	49.84	32.24
RoBERTa	zh: OCNLI + XNLI	51.13	38.16	37.98	66.19	75.73	53.31	35.43
XLM-R	zh ori: OCNLI	54.33	**54.19**	**49.02**	52.46	79.70	59.52	31.08
XLM-R	zh MT: XNLI	50.79	33.39	35.33	66.01	87.23	33.17	**49.60**
XLM-R	zh: OCNLI + XNLI	52.43	34.51	36.93	59.98	88.70	54.37	40.08
XLM-R	en: MNLI	49.09	33.27	37.98	66.25	89.70	34.69	32.65
XLM-R	en: En-all-NLI	55.37	33.43	39.70	**66.65**	92.34	64.11	35.99
XLM-R	mix: OCNLI + En-all-NLI	**57.95**	40.70	44.49	63.67	91.54	**74.47**	32.81
XLM-R	mix: XNLI + En-all-NLI	57.73	40.30	37.82	66.67	**93.19**	61.52	46.87

Table 8: Accuracy on the Chinese semantic probing datasets, designed following Richardson et al. (2020).

stable results than XNLI. Furthermore, when coupled with English data to be used with XLM-R, we see again that OCNLI + En-all-NLI results in an accuracy 3 percent higher than XNLI + En-all-NLI.

5.5 Semantic fragments

Results on the semantic probing datasets (shown in Table 8) are more mixed. First, the results are in general much worse than the other evaluation data, but overall, XLM-R fine-tuned with OCNLI and all English data still performs the best. The overall lower performance is likely due to the longer length of premises and hypotheses in the semantic probing datasets, compared with the other three evaluation sets. Second, zero-shot transfer is better or on par with monolingual Chinese RoBERTa in 4/6 semantic fragments (except Boolean and quantifier). Third, for Boolean and comparative, XLM-R fine-tuned with OCNLI has a much better result than all other monolingual models or XLM-R fine-tuned with mixed data. We also observe that all models have highest performance on the "counting" fragment. Note that none of the models have seen any data from the "counting" fragment during fine-tuning. That is, all the knowledge come from the pre-training and fine-tuning on general NLI datasets. The surprisingly good performance of XLM-R model (w/ En-all-NLI, 92.34%) suggests that it may have already acquired a mapping from counting the words/names to numbers, and this knowledge can be transferred cross-linguistically.

6 Conclusion and Future Work

In this paper, we examine the cross-lingual transfer ability of XLM-R in the context of Chinese NLI through four new sets of aversarial/probing tasks and a total of 17 new high quality and linguistically motivated challenge datasets. We find that multilingual transfer via fine-tuning solely on benchmark English data generally yields impressive performance. In 3/4 on our task categories, such *zero-shot transfer* outperforms our best monolingual models trained on benchmark Chinese NLI data, including 3/5 of our hand-crafted challenge tasks that test uniquely Chinese linguistic phenomena. These results suggest that multilingual models are indeed capable of considerable cross-lingual linguistic transfer and that zero-shot NLI may serve as a serious alternative to large-scale dataset development for new languages.

These results come with several important caveats. Model performance is still outperformed by conservative estimates of human performance and our best models still have considerable room for improvement; we hope that our new resources will be useful for continuing to benchmark progress on Chinese NLI. We also find that high-quality Chinese NLI data (e.g., OCNLI) *can* help improve results further, which suggests an important role for certain types of expertly annotated monolingual data in a training pipeline. In virtue of our study being limited to *behavioral testing*, the exact reason for *why* cross-lingual zero-shot transfer generally performs well, especially on some Chinese-specific phenomena, is an open question that requires further investigation. In particular, we believe that techniques that couple behavioral testing with *intervention* techniques (Geiger et al., 2020; Vig et al., 2020) and other analysis methods (Giulianelli et al., 2018; Belinkov and Glass, 2019) might provide insight, and that our new Chinese resources can play an important role in such future work.

Acknowledgments

This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute. He Zhou is sponsored by China Scholarship Council.

References

Hossein Amirkhani, Mohammad Azari Jafari, Azadeh Amirak, Zohreh Pourjafari, Soroush Faridan Jahromi, and Zeinab Kouhkan. 2020. Farstail: A Persian Natural Language Inference Dataset. *arXiv preprint arXiv:2009.08820*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of ACL*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Emrah Budur, Rıza Özçelik, Tunga Gungor, and Christopher Potts. 2020. Data and representation for turkish natural language inference. In *Proceedings of EMNLP*.

Wanxiang Che, Yunlong Feng, Libo Qin, and Ting Liu. 2020. N-LTP: A open-source neural Chinese language technology platform with pretrained models. *arXiv preprint arXiv:2009.11616*.

Hyunjin Choi, Judong Kim, Seongho Joe, Seungjai Min, and Youngjune Gwon. 2021. Analyzing zero-shot cross-lingual transfer in supervised nlp tasks. In *2020 25th International Conference on Pattern Recognition (ICPR)*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of ICLR*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7057–7067.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of EMNLP*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of EMNLP*.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.

Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of BlackBoxNLP*.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of BlackboxNLP*, pages 240–248.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *Proceedings of ACL*.

Emily Goodwin, Koustuv Sinha, and Timothy O'Donnell. 2020. Probing linguistic systematicity. In *Proceedings of ACL*, pages 1958–1969.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of NAACL*.

Yuta Hayashibe. 2020. Japanese realistic textual entailment corpus. In *Proceedings of LREC*.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence Moss. 2020. OCNLI: Original Chinese Natural Language Inference. In *Findings of EMNLP*.

K Karthikeyan, Zihan Wang, Stephen Mayhew, and Dan Roth. 2019. Cross-lingual ability of multilingual bert: An empirical study. In *Proceedings of ICLR*.

Daniel Khashabi, Arman Cohan, Siamak Shakeri, Pedram Hosseini, Pouya Pezeshkpour, Malihe Alikhani, Moin Aminnaseri, Marzieh Bitaab, Faeze Brahman, Sarik Ghazarian, et al. 2020. ParsiNLU: A suite of language understanding challenges for Persian. *arXiv preprint arXiv:2012.06154*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTaiL: A Textual Entailment Dataset from Science Question Answering. In *Proceedings of AAAI*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33.

Charles N Li, Sandra A Thompson, and Sandra A Thompson. 1981. *Mandarin Chinese: A Functional Reference Grammar*. Univ of California Press.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of ACL*.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of COLING*.

Nikita Nangia and Samuel Bowman. 2019. Human vs. Muppet: A Conservative Estimate of Human Performance on the GLUE Benchmark. In *Proceedings of ACL*.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *Proceedings of ACL*.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific BERT models. *arXiv preprint arXiv:2003.02912*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of ACL*.

Adam Poliak. 2020. A survey on recognizing textual entailment as an NLP evaluation. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. In *Proceedings of *SEM*.

Livy Real, Erick Fonseca, and Hugo Gonçalo Oliveira. 2020. Organizing the ASSIN 2 shared task. In *Proceedings of the ASSIN 2 Shared Task: Evaluating Semantic Textual Similarity and Textual Entailment in Portuguese*.

Kyle Richardson, Hai Hu, Lawrence S Moss, and Ashish Sabharwal. 2020. Probing Natural Language Inference Models through Semantic Fragments. In *Proceedings of AAAI*.

Swarnadeep Saha, Yixin Nie, and Mohit Bansal. 2020. Conjnli: Natural language inference over conjunctive sentences. In *Proceedings of EMNLP*.

Felipe Salvatore, Marcelo Finger, and Roberto Hirata Jr. 2019. A logical-based corpus for cross-lingual evaluation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 22–30.

Tianqi Sun. 2009. *A Study on the License Pattern and Mechanism of Non-core Arguments in Mandarin Chinese*. Ph.D. thesis, Peking University.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of NAACL*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of BlackboxNLP*.

Gijs Wijnholds and Michael Moortgat. 2021. SICKNL: A Dataset for Dutch Natural Language Inference. In *Proceedings of EACL*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of NAACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*.

Shijie Wu and Mark Dredze. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of EMNLP*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of COLING*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Endong Xun, Gaoqi Rao, Xiaoyue Xiao, and Jiaojiao Zang. 2016. The construction of the BCC corpus in the age of big data. *Corpur Linguistics (in Chinese)*, pages 93–109.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. Help: A dataset for identifying shortcomings of neural models in monotonicity reasoning. In *Proceedings of *SEM*, pages 250–255.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.

Heuristic	entailment	contradiction	neutral
lexical overlap	944	155	109
subsequence	190	10	18

Table 9: Distribution of the two heuristics in OCNLI.

Heuristic	entailment	contradiction	neutral
lexical overlap	441	647	340
subsequence	100	193	220
Total	541	840	560

Table 10: Distributional statistics of our synthesized Chinese HANS dataset.

A Details for dataset creation

In this section, we list example NLI pairs and their translations. For examples of the Chinese HANS and stress tests, see Table 2. For the expanded diagnostics, see Table 11. For the semantic/logic probing dataset, see Table 3.

Chinese HANS Table 9 lists the number of examples in OCNLI for each inference label that satisfy the two heuristics we are examining. We observe that *entailment* examples take the majority for both heuristics. Therefore, we hypothesize that if the heuristics are learned, the *entailment* examples are likely to be correctly predicted while *non-entailment* (contradiction and neutral) examples are prone to receive wrong prediction.

To guarantee the generated sentences are syntactically and semantically sound, we add features for our vocabulary so that subject- predicate and verb-object constraints are satisfied, e.g., some verbs can only take animate subjects and objects. We then generate 50 premise-hypothesis pairs for each template described in our Github repository.[12] Excluding duplicated examples, our generated dataset has 1,941 pairs and the distribution of the three labels is shown in Table 10.

Antonym After looking at the quality of initially generated data, we decided to replace only the nouns and adjectives with their antonyms since such replacements are most likely result in grammatical and contradictory hypotheses.[13]

Synonym After inspecting the initially generated data, we decided to perform replacements only to verbs and adjectives. To ensure the quality of synonyms, we rank the synonyms from a commonly used synonym dictionary by their vector similarity to the original word, and pick the top ranking synonym.[14]

Distraction We created the distraction data similar to the stress test setting (Naik et al., 2018) but experimented with variations as to where "distractor statement"—either a tautology or a true statement—was added: the premise or the hypothesis. The distractor statement also varied w.r.t. whether it contains a negation:

- **Premise-no-negation**: A distractor statement is added to the end of the premise and it contains no negation.
- **Premise-negation**: A distractor statement containing a negation is added to the premise.
- **Hypothesis-no-negation**: A distractor statement is added to the end of the hypothesis.
- **Hypothesis-negation**: Same as the previous condition except that the distractor contains a negation.

Only two tautologies are used in Naik et al. (2018). In this paper, to thoroughly examine the influence of different true statements, we designed 50 tautology/statements varied in three factors: length, out-of-vocabulary, and negation word. There are 25 statements pairs in total (1 tautology and 24 true statements); each pair includes a true statement and its corresponding true statement with negation form. All the statements range from 5 to 16 characters. For the true statements in negation form, two common Chinese negation words 不 and 没 are used for negation. For the 24 true statement pairs, half of them contains at least one Out-of-Vocabulary word in OCNLI.

Experiments show that length, Out-of-Vocabulary words, and the choice of negator have little effects on the results.

Spelling We generate a set of data containing "spelling errors" by replacing one random character in the hypotheses with its homonym, which is defined as a character with the same *pinyin* pronunciation ignoring tones. We also limit the frequency

---

[12] https://github.com/huhailinguist/ChineseVariousNLI

[13] We use the LTP toolkit (https://github.com/HIT-SCIR/ltp) to annotate the POS tags and our antonym list is from https://github.com/liuhuanyong/ChineseAntiword.

[14] We use the synonym list from https://github.com/Keson96/SynoCN and the similarity score from the Python package Synonyms at https://github.com/chatopera/Synonyms.

of the homonym as within the range of 100 to 6000 so that the character is neither too rare nor too frequent.

Numerical reasoning We extracted sentences from Ape210k (Zhao et al., 2020), a large-scale math word problem dataset containing 210K Chinese elementary school-level problems[15]. We generate entailed, contradictory and neutral hypotheses for each premise, with the rules below:

1. **Entailment:** Randomly choose a number x and change it to y from the hypothesis. If the y > x, prefix it with one phrase that translate to “less than”; if y < x, prefix it with one phrase that translate to “more than”.
   Premise: *Mary types 110 words per minute.* Hypothesis: *Mary types less than 510 words per minute.*
2. **Contradiction:** Perform either 1) randomly choose a number x from the hypothesis and change it; 2) randomly choose a number from the hypothesis and prefix it with one phrase that translate to “less than” or “more than”.
   Premise: *Mary types 110 words per minute.* Hypothesis: *Mary types 710 words per minute.*
3. **Neutral:** Reverse the corresponding entailed premise-hypothesis pairs.
   Premise: *Mary types less than 510 words per minute.* Hypothesis: *Mary types 110 words per minute.*

The result contains 2,871 unique premise sentences and 8,613 NLI pairs.

Diagnostics The diagnstics for *classifiers* (or measure word) and *non-core arguments* are explained in detail below (see examples in Table 11).

1. *classifiers* (or measure word): in Chinese, when modified by a numeral, a noun must be preceeded by a category of words called classifier. They can be semantically vacuous but sometimes also carry semantic content: 一匹狼 *one pi wolf* (one wolf); 一群狼 *one qun wolf* (one pack of wolves). Our examples require the model to understand the semantic content of the classifiers.
2. *non-core arguments*: in Chinese syntax, sometimes a noun phrase at the argument position (e.g., object) is not serving as an object: 今天吃筷子，不吃叉子。*today eat chopsticks, not eat fork* (We eat **with** chopsticks today, not with fork). Sun (2009) shows that this structure is very productive in Chinese and we take example sentences from her dissertation.

Additionally, for the *pro-drop* examples, they are constructed such that the models return the correct inference relation only when they successfully identify what the dropped *pro* refers to. That is, our constructed premises involve several entities the dropped *pro* could potentially refer to, and the entailed hypothesis identifies the correct referent while the neutral/contradictory hypothesis does not (see Table 11 for an example).

B Hyperparameters for experiments

Table 12 presents the hyperparameters used for the models. The learning-rate search space for RoBERTa is: 1e-5, 2e-5, 3e-5, 4e-5 and 5e-5, for XLM-R: 5e-6, 7e-6, 9e-6, 2e-5 and 5e-5.

C Chinese-to-English transfer

We present Chinese-to-English transfer results in this section. As mentioned in the main text, for most of the cases, zero-shot transfer learning does not work well mostly likely due to the small size of OCNLI. However, for 3 out of the 4 datasets, XLM-R fine-tuned with the mix data outperforms the monolingual setting, suggesting that even OCNLI is only 1/20 of En-all-NLI, XLM-R can still acquire some useful information from OCNLI, in addition to what is present in En-all-NLI.

Specifically, (1) for English HANS, XLM-R fine-tuned with OCNLI is about 13 percentage points below the best English monolingual model, shown in Table 13. (2) For stress tests shown in Table 14, the gap is about 5 percent (XLM-R with OCNLI = 74%; RoBERTa with En-all-NLI = 79%). Interestingly, XLM-R with OCNLI performs the best for Negation and Word overlap. It even outperforms RoBERTa w/ MNLI on the Antonym, which seems to be consistent with the high performance of OCNLI-trained models on the Chinese Antonym in our constructed stress tests. (3) For semantic probing data, as shown in Table 15, XLM-R with OCNLI is 5 percent behind monolingual model fine-tuned with all English NLI, but performs better than the monolingual RoBERTa fine-tuned with

---

[15] We split all problems into individual sentences and filter out sentences without numbers. Then we remove sentences without any named entities (“PERSON”, “LOCATION” and “ORGANIZATION”) using the NER tool provided by LTP toolkit (Che et al., 2020).

category	n	premise	hypothesis	label
CLUE (Xu et al., 2020)	514	有些学生喜欢在公共澡堂里唱歌。 Some students like to sing in public showers.	有些女生喜欢在公共澡堂里唱歌。 Some female students like to sing in public showers.	N
CLUE expansion (ours)	800	雷克雅未克所有旅馆的床位加在一起才一千六百个。 There are only one thousand six hundred beds in all hotels in Reykjavik.	雷克雅未克有旅馆的床位超过一千个。 Some hotel in Reykjavik has over a thousand beds.	N
World Knowledge (ours)	37	上海在北京的南边。 Shanghai is to the south of Beijing.	北京在上海的南边。 Beijing is to the south of Shanghai.	C
Classifier (ours)	138	这些孩子吃了一个苹果。 These children ate an apple.	这些孩子吃了一筐苹果。 These children ate a basket of apples.	N
Chengyu/idioms (ours)	250	这帮人可狡猾得很啊，你一个电话打过去，打草惊蛇，后果不堪设想。 These people are so cunning! If you call them, it would alert them, and as we say in a Chinese idiom ”if you hit the grass, it would alert the snakes.” The consequences would be unimaginable.	你打电话过去会让这帮人察觉，造成不好的结果。 If you call them, it will alert them, and bring negative consequences.	E
		same as above	这些狡猾的人养了很多蛇。 These cunning people have raised a lot of snakes.	N
Pro-drop (ours)	197	见了很多学生，又给老师们开了两个小时会，校长和主任终于可以下班了。 After (*pro*) meeting many students and (*pro*) having two hours of meeting with the teachers, the principal and the director can finally get off work.	校长见了很多学生。 The principal met many students.	E
		same as above	老师们见了很多学生。 The teachers met many students.	N
Non-core arguments (ours)	185	平时范志毅都踢后卫的，今天却改当前锋了。 Zhiyi Fan usually plays full back in soccer, but today he switched to playing forward.	范志毅经常用腿踢对方的后卫。 Zhiyi Fan usually uses his legs to kick the other team’s full back.	N

Table 11: Example NLI pairs in expanded diagnostics with translations.

Model	Training Data	LR
RoBERTa	zh MT: XNLI-small	3e-05
RoBERTa	zh MT: XNLI	2e-05
RoBERTa	zh ori: OCNLI	2e-05
RoBERTa	zh: OCNLI + XNLI	3e-05
XLM-R	zh ori: OCNLI	5e-06
XLM-R	zh MT: XNLI	7e-06
XLM-R	zh: OCNLI + XNLI	7e-06
XLM-R	en:MNLI	5e-06
XLM-R	en: En-all-NLI	7e-06
XLM-R	mix: OCNLI + En-all-NLI	7e-06
XLM-R	mix: XNLI + En-all-NLI	7e-06

Table 12: Hyper-parameters used for fine-tuning the models. All models are fine-tuned for 3 epochs with maximum length of 128.

MNLI (53.6% vs. 51.3%). This is quite surprising since the size of OCNLI is only 1/8 of MNLI. (4) For the English diagnostics as shown in Table 16 and Table 17, XLM-R with OCNLI is 7 percent behind RoBERTa fine-tuned with MNLI.

We leave it for future work to thoroughly examine transfer learning from a “low-resource” language such as Chinese to the high-resource one such as English.

Model	Fine-tuned on	Overall	Lexical_overlap	Subsequence	Constituent	Entailment	Non-entailment
RoBERTa	en: En-all-NLI	76.54	96.79	67.77	65.06	99.81	53.27
RoBERTa	en: MNLI	77.63	95.60	**68.08**	69.21	99.74	55.52
XLM-R	en: En-all-NLI	75.72	95.52	62.99	68.63	99.91	51.52
XLM-R	en: MNLI	74.80	92.92	65.24	66.23	98.83	50.76
XLM-R	zh ori: OCNLI	64.37	71.28	54.42	67.41	98.39	30.35
XLM-R	zh MT: XNLI	68.83	81.67	62.07	62.74	99.13	38.53
XLM-R	zh mix: OCNLI+XNLI	71.30	82.52	61.72	69.66	99.08	43.52
XLM-R	mix: OCNLI+En-all-NLI	**78.56**	**96.92**	64.91	**73.84**	**99.92**	**57.20**
XLM-R	mix: XNLI+En-all-NLI	74.65	93.93	60.97	69.04	99.96	49.34

Table 13: Results of English HANS (McCoy et al., 2019).

Model	Fine-tuned on	Overall	Antonym	Content word swap	Function word swap	Keyboard	Swap	Length mismatch	Negation	Numerical reasoning	Word overlap
RoBERTa	en: En-all-NLI	79.48	82.91	**86.22**	88.71	**87.8**	**87.48**	**88.28**	60.25	79.26	62.85
RoBERTa	en: MNLI	77.9	69.03	85.74	**88.75**	87.39	87.05	88.23	59.19	65.46	61.48
XLM-R	en: En-all-NLI	79.6	86.25	85.26	87.38	86.31	86.72	87.25	61.06	**82.84**	65.79
XLM-R	en: MNLI	77.6	74.65	85.09	87.33	86.08	86.42	86.96	60.95	54.66	65.13
XLM-R	zh ori: OCNLI	74.31	72.52	75.12	77.71	76.27	76.39	77.23	**72.86**	55.85	**72.79**
XLM-R	zh MT: XNLI	77.78	65.12	85.11	86.64	85.79	85.71	85.91	63.52	43.95	71.63
XLM-R	zh mix: OCNLI+XNLI	77.83	66.83	84.96	86.69	85.81	85.87	85.98	63.97	51.56	68.38
XLM-R	mix: OCNLI+En-all-NLI	**80.01**	**86.33**	85.22	87.40	86.26	86.77	87.23	62.52	81.79	67.54
XLM-R	mix: XNLI+En-all-NLI	79.38	85.27	85.35	87.20	86.28	86.74	87.22	60.29	80.50	66.19

Table 14: Results of English stress test (Naik et al., 2018).

Model	Fine-tuned on	Overall	Boolean	Comparative	Conditional	Counting	Monotonicity hard	Monotonicity simple	Negation	Quantifier
RoBERTa	en: MNLI	51.31	43.58	39.60	66.24	63.34	61.28	60.10	37.26	39.08
RoBERTa	en: En-all-NLI	58.72	60.18	40.28	66.30	66.22	59.60	58.98	64.46	**53.74**
XLM-R	en: MNLI	53.54	59.16	41.62	66.30	61.72	63.26	**62.82**	33.52	39.92
XLM-R	en: En-all-NLI	59.85	71.58	45.18	66.30	60.40	63.86	62.02	65.68	43.78
XLM-R	zh ori: OCNLI	53.61	66.02	**60.62**	41.10	58.00	47.86	49.88	51.88	53.50
XLM-R	zh MT: XNLI	52.29	43.24	39.00	66.22	65.66	58.08	62.74	34.12	49.24
XLM-R	zh mix: OCNLI+XNLI	54.68	54.64	38.84	66.28	**67.38**	58.18	61.38	41.88	48.82
XLM-R	mix: OCNLI+En All NLI	**60.20**	**71.20**	42.58	**66.30**	62.40	**64.72**	60.90	**68.58**	44.88
XLM-R	mix: XNLI+En-all-NLI	60.06	65.8	46.86	66.30	65.54	61.56	61.44	68.50	44.48

Table 15: Results of English semantic probing datasets (Richardson et al., 2020).

Model	Fine-tuned on	Overall	active passive	anaphora coreference	common sense	conditionals	anaphora coreference	coordination scope	core args	datives	disjunction	double negation	downward mono-tone	ellipsis implicits	existential	factivity	genitives partitives	intersectivity
RoBERTa	en: MNLI	66.87	**62.35**	**67.59**	**69.47**	62.50	78.00	63.50	69.62	85.00	39.47	**92.86**	**19.33**	65.29	65.00	62.06	**95.00**	60.43
RoBERTa	en: En-all-NLI	**68.03**	61.76	70.00	69.33	**63.75**	**82.50**	**68.00**	**75.77**	**85.00**	**41.58**	92.14	18.67	**67.65**	65.00	**62.35**	94.00	59.57
XLM-R	en: MNLI	63.03	61.76	62.76	59.73	55.62	76.00	61.50	61.54	85.00	26.84	91.43	16.00	64.12	69.00	51.47	90.00	60.00
XLM-R	en: En-all-NLI	64.57	61.76	65.17	61.47	60.00	76.00	66.00	65.77	85.00	33.16	89.29	14.00	62.94	**71.00**	58.53	90.00	**60.87**
XLM-R	zh ori: OCNLI	59.67	60.00	59.31	57.20	58.12	70.00	56.50	61.54	85.00	30.00	67.14	17.33	54.71	66.00	46.18	90.00	59.57
XLM-R	zh MT: XNLI	61.76	61.18	64.14	60.67	58.75	72.50	60.00	60.77	85.00	33.16	91.43	12.67	58.24	64.00	48.24	90.00	57.83
XLM-R	zh mix: OCNLI+XNLI	61.78	61.76	62.76	56.93	57.50	74.50	61.00	61.54	85.00	31.05	90.00	12.00	57.65	65.00	48.53	90.00	57.39
XLM-R	mix: OCNLI+En-all-NLI	64.51	61.76	63.45	61.60	58.75	76.00	66.00	67.31	85.00	35.26	90.71	15.33	60.59	68.00	60.59	91.00	60.87
XLM-R	mix: XNLI+En All LI	64.37	61.18	64.83	62.27	61.88	73.00	65.00	65.38	85.00	35.26	91.43	14.67	63.53	70.00	57.94	94.00	60.87

Table 16: Results of English Diagnostics from GLUE-Part I (Wang et al., 2018).

Model	Fine-tuned on	intervals numbers	lexical entailment	morphological negation	named entities	negation	nominalization	non-monotone	prepositional phrases	quantifiers	redundancy	relative clauses	restrictivity	symmetry collectivity	temporal	universal	upward monotone	world knowledge
RoBERTa	en: MNLI	54.74	66.71	89.23	57.22	66.59	82.14	56.00	86.18	78.46	79.23	63.75	55.38	67.86	56.25	**84.44**	76.47	48.51
RoBERTa	en: En-all-NLI	**63.16**	**71.57**	**89.23**	61.11	**69.02**	84.29	**57.33**	84.41	74.62	73.85	63.75	53.08	**70.00**	**69.38**	83.33	73.53	**49.55**
XLM-R	en: MNLI	45.79	65.57	84.62	61.11	61.95	82.14	52.00	85.88	**82.69**	78.46	62.50	52.31	57.14	51.25	80.00	74.12	44.03
XLM-R	en: En-all-NLI	45.79	69.71	84.62	61.67	64.15	**85.71**	48.67	84.71	79.23	69.23	63.12	46.15	59.29	60.00	77.78	75.29	45.82
XLM-R	zh ori: OCNLI	39.47	56.29	75.38	41.11	53.41	73.57	51.33	85.59	63.08	83.08	62.50	**70.77**	64.29	54.38	62.22	**78.82**	47.31
XLM-R	zh MT: XNLI	42.11	60.14	84.62	61.11	61.95	74.29	53.33	**86.76**	73.46	84.62	60.62	60.00	57.86	40.62	**84.44**	68.24	47.31
XLM-R	zh mix: OCNLI+XNLI	43.68	59.29	83.08	**63.33**	62.20	74.29	52.00	**86.76**	76.92	**85.38**	62.50	60.77	59.29	43.75	82.22	67.65	47.61
XLM-R	mix: OCNLI+En-all-NLI	45.26	69.86	85.38	62.22	65.12	85.71	50.67	85.00	74.62	69.23	**68.12**	47.69	60.71	56.25	77.78	75.29	45.67
XLM-R	mix: XNLI+En All LI	44.21	67.29	86.15	62.22	63.90	83.57	49.33	84.71	75.00	70.77	66.88	46.92	58.57	57.50	74.44	74.12	45.82

Table 17: Results of English Diagnostics from GLUE-Part II (Wang et al., 2018).

Using surprisal and fMRI to map the neural bases of broad and local contextual prediction during natural language comprehension

Shohini Bhattasali and **Philip Resnik**
Linguistics/UMIACS
University of Maryland
College Park, MD
{shohini, resnik}@umd.edu

Abstract

Context guides comprehenders' expectations during language processing, and information-theoretic surprisal is commonly used as an index of cognitive processing effort. However, prior work using surprisal has considered only within-sentence context, using n-grams, neural language models, or syntactic structure as conditioning context. In this paper, we extend the surprisal approach to use broader topical context, investigating the influence of local and topical context on processing via an analysis of fMRI time courses collected during naturalistic listening. Lexical surprisal calculated from ngram and LSTM language models is used to capture effects of local context; to capture the effects of broader context a new metric based on topic models, topical surprisal, is introduced. We identify distinct patterns of neural activation for lexical surprisal and topical surprisal. These differing neuro-anatomical correlates suggest that local and broad contextual cues during sentence processing recruit different brain regions and that those regions of the language network functionally contribute to processing different dimensions of contextual information during comprehension. More generally, our approach adds to a growing literature using methods from computational linguistics to operationalize and test hypotheses about neuro-cognitive mechanisms in sentence processing.

1 Introduction

Narratives unfold over time and comprehenders incrementally process words and sentences. In order to understand the current word and sentence, we have to integrate current input with the information from the previous context.

In characterizing this process, the notion of *surprisal* from information theory has been prevalent in psycholinguistic modeling, following the work of Hale (2001) and Levy (2008). Surprisal operationalizes how unexpected a word is as its pointwise information content given prior context, $-\log \mathrm{P}(w_i|w_1...w_{i-1})$. Generally the theory of surprisal, as applied in the study of human language comprehension, proposes that probabilistic predictions made by comprehenders yield variability in word-by-word processing difficulty: when surprisal is high, the current word is unexpected and cognitive processing effort increases accordingly. This linking has been validated in prior work connecting surprisal with measurable reflexes of cognitive effort, using probabilities conditioned on lexical (sequential) and syntactic contexts (e.g. Brennan et al., 2016; Lopopolo et al., 2017; Brennan and Hale, 2019; Shain et al., 2020).

In this paper, we extend the surprisal paradigm beyond prior work looking only at local context, to investigate the influence of broader contextual information during incremental sentence processing. As an illustration, consider examples (a) and (b), which illustrate that, while a word might be extremely difficult to predict given the immediate local context, it might be less unexpected/more predictable given the broader topic under discussion. The actual word to be predicted here is *China*, which is not at all predictable given the immediate context, but more likely in a longer discourse about traveling (*airplanes*, *places*, *world*, *geography*, etc).

(a) I could recognize at first glance _____

(b) *So I had to choose another profession, and I learned to fly airplanes. I flew a little in many places around the world. And geography it's true has served me well.* I could recognize at first glance _____

Following prior studies, we use measures of lexical surprisal to capture the influence of local context, and we introduce a new measure, *topical*

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3786–3798
August 1–6, 2021.

surprisal, to operationalize the predictive role of broader context. We use both kinds of predictors to investigate how processing based on different contextual cues differs in their mapping in the brain. Specifically, our research questions are:

- How does the previous topical context affect our expectations about the next word?
- How do local vs. broad contextual prediction influence our incremental language processing? Do they have distinct neural correlates?

We apply computational modeling to investigate these questions in a way that would not be feasible in a more traditional, trial-based experimental paradigm by taking advantage of data collected using fMRI brain imaging during continuous, naturalistic listening (Hamilton and Huth, 2020). This data collection method has emerged as a new testing ground for linking processing hypotheses with neurobiological architectures in the brain (Maguire, 2012; Willems, 2015; Kandylaki and Bornkessel-Schlesewsky, 2019). Using lexical surprisal and our new measure of topical surprisal as computational predictors of cognitive activity, we demonstrate that processing of local and broad context recruits different brain regions, suggesting that those regions of the language network functionally contribute to processing different dimensions of contextual prediction during human language comprehension.

2 Background and Related Work

2.1 Surprisal as a cognitive measure

Prior neurolinguistic work has used surprisal as an index of cognitive processing effort. Behavioral measurements like self-paced reading are one way to infer how much effort is involved while processing some piece of linguistic input (e.g., Futrell et al., 2021); other methods more directly measure activity in the brain, including functional magnetic resonance imaging (fMRI), which we will focus on in this paper, as well as magnetoencephalography (MEG) (e.g., Brodbeck et al., 2018) and electroencephalography (EEG) (e.g., Ettinger et al., 2016; Brennan and Hale, 2019; Michaelov and Bergen, 2020).

In such work, the logic is generally as follows. First, as noted in §1 we assume that when a word is less expected given the context, processing it during comprehension will require more work in the brain. Then, we computationally estimate a model of surprisal using a corpus:

$$\text{surprisal}_M(w_i) = -\log \text{P}_M(w_i | w_1 ... w_{i-1}) \quad (1)$$

Two common instantiations for M include ngram models and models conditioned on prior syntactic context (Hale, 2001).

By the first assumption, the value of Eq (1) is taken to be a predictor of processing effort at word w_i. Therefore, the key final step is to analyze the relationship between that estimated effort, as predicted by the model, and observed activity measured in the brain. In the case of fMRI, neural activity is measured by detecting changes associated with blood flow (see §4). When there are significant correlations between the predicted effort, surprisal$_M$, and activity in some region of the brain, this constitutes evidence for that region being involved in processing of the information that M has used in its predictions. For example, if the brain activity in a region is correlated with an estimate of surprisal that uses syntactic predictions, that provides evidence for that region being a locus for human syntactic processing.

In prior work following this logic, using the surprisal paradigm with fMRI to localize processing associated with lexical and syntactic context, the findings implicate a range of core regions of the language network. Across different languages, lexical surprisal recruits a mostly left-lateralized network, predominantly consisting of Inferior Frontal Gyrus, Interior Temporal Sulcus, Middle Frontal Gyrus, Posterior Temporal regions, extending to some bilateral regions, namely Anterior Temporal Lobe and Superior Temporal Gyrus (Brennan, 2016; Willems et al., 2016; Lopopolo et al., 2017; Shain et al., 2020). Syntactic surprisal has also mapped onto a left-lateralized network consisting of the Inferior Parietal Lobule, Inferior Frontal Gyrus, Middle Temporal Gyrus, along with some evidence for bilateral processing in the Anterior Temporal Lobe (Brennan et al., 2016; Henderson et al., 2016; Lopopolo et al., 2017; Shain et al., 2020).

2.2 Neural language models in cognitive neuroscience

There has been a growing trend of using neural language models in cognitive neuroscience research, often using neural data collected from individuals during naturalistic listening.[1] As one salient example, Wehbe et al. (2014) investigated how well vector representations predicted brain activity for subjects reading fiction, in their case material from *Harry Potter and the Sorcerer's Stone*, based on within-sentence context. Also working within the sentence using naturalistic listening, Toneva et al. (2020) derived composed representations of "supra-word meaning" using contextualized word representations (ELMo, Peters et al., 2018) to capture the compositional meaning of multi-word expressions and event/argument structure. Jain and Huth (2018) make predictions of neural activity using LSTM representations from up to the previous 20 words of context (which would be on the order of 8-10 seconds of speech on average).

Work of this kind has a number of dimensions of variation. One is the nature of the neural measurement, e.g. fMRI versus MEG, which relates crucially to the cognitive questions being asked, since some questions involve temporal locality, a strength of MEG, and others involve spatial locality, a strength of fMRI. Another dimension is the nature of the training data for the computational modeling, e.g. material from the experimental dataset (*Harry Potter*, as in Wehbe et al. (2014)) versus a broader coverage corpus such as a large collection of Reddit comments as used by Jain and Huth (2018)). Finally, there is the nature of the model itself; for example, how much context it takes into account and whether it involves, for example, non-contextual word embeddings, sequentially derived embeddings, or something else.

In this work, we use broad coverage corpora such as COCA (Davies, 2008) or Wikipedia to train our models. In addition to using ngram and LSTM models to capture within-sentence context, we introduce topical surprisal (§3.5), based on topic modeling, as a way to look at functional localization of correlates of broader, non-sequential contextual processing using fMRI.

3 fMRI Study

3.1 Method

We follow Brennan et al. (2012) in using a spoken narrative as a stimulus. Participants hear a story over headphones while they are in the MRI scanner. As we describe in greater detail in §4, the sequence of neuroimages collected during their session becomes the dependent variable in a regression against word-by-word predictors that have been derived from the text of the story.

3.2 Stimuli

The English audio stimulus was Antoine de Saint-Exupéry's *The Little Prince*, translated by David Wilkinson and read by Karen Savage. It constitutes a fairly lengthy exposure to naturalistic language, comprising 19,171 tokens; 15,388 words and 1,388 sentences, and lasting over an hour and a half. *The Little Prince* has been used in a number of previous fMRI studies of language processing, e.g. Li et al. (2018); Bhattasali et al. (2019); Zhang (2020); Stanojević et al. (2021)

3.3 Participants

56 participants were scanned and 5 of them were excluded since they had incomplete scanning sessions. Participants included were fifty-one volunteers (32 women and 19 men, 18-37 years old) with no history of psychiatric, neurological, or other medical illness or history of drug or alcohol abuse that might compromise cognitive functions. All strictly qualified as right-handed on the Edinburgh handedness inventory (Oldfield, 1971). All self-identified as native English speakers and gave their written informed consent prior to participation, in accordance with Cornell University's IRB guidelines. Participants were compensated for their time, consistent with typical practice for studies of this kind. They were paid $65 at the end of the session.

3.4 Presentation

After giving their informed consent, participants were familiarized with the MRI facility and assumed a supine position on the scanner gurney. The presentation script was written in PsychoPy (Peirce, 2007). Auditory stimuli were delivered through MRI-safe, high-fidelity headphones (Confon HP-VS01, MR Confon, Magdeburg, Germany) inside the head coil. The headphones were secured against the plastic frame of the coil using foam blocks. Using a spoken recitation of the US Constitution, an experimenter increased the volume until participants reported that they could hear clearly. Participants then listened passively to the audio storybook for 1 hour 38 minutes. The story had nine chapters and at the end of each chapter the

---

[1] Although not directly relevant to the scientific strategy we discuss here, we note that there is also a body of work that goes in the other direction, using methods from psycholinguistics and neuroscience to improve our understanding and use of neural language models, e.g. Toneva and Wehbe (2019); Ettinger (2020); Misra et al. (2020).

participants were presented with a multiple-choice questionnaire with four questions (36 questions in total), concerning events and situations described in the story. These questions served to confirm participants' comprehension. They were viewed via a mirror attached to the head coil and answered through a button box. The entire session lasted around 2.5 hours.[2]

3.5 Deriving the predictors

Recall that surprisal measures how unexpected each word w_i is given the preceding context (Eq 1), and we use measures of surprisal as the linking hypotheses in our study between the contextual predictions of our model and neural activity. The three different surprisal predictors we use are described below, along with how they were calculated. Fig. 2 shows a visual comparison of the word-by-word predictors on a single sentence from the text.[3]

Ngram surprisal. The ngram surprisal values are based on a 5gram language model and were calculated using the `kenlm` library (Heafield et al.) with Modified Kneser-Ney Smoothing. The 5gram language model was trained on the Corpus of Contemporary American English (COCA, Davies, 2008), which is a large, genre-balanced corpus of American English and consists of over one billion words of text sampled across spoken, fiction, popular magazines, newspapers, and academic texts.

LSTM surprisal. These surprisal values are based on a long short-term memory (LSTM) language model (Hochreiter and Schmidhuber, 1997) trained on 90 million words of English Wikipedia by Gulordava et al. (2018). It had two LSTM layers with 650 hidden units each, 650 dimensional word embeddings, a learning rate of 20, a dropout rate of 0.2 and a batch size 128, and was trained for 40 epochs (with early stopping). Like the majority of previous work computing LSTM-surprisal, our input is a single sentence and we make predictions only based on context within the sentence (Brennan and Hale (2019); van Schijndel and Linzen (2018), though cf. Jain and Huth (2018)). The surprisal values were calculated using the Neural Complexity toolkit with the baseline non-adaptive model (van Schijndel and Linzen, 2018).

Topical surprisal. We introduce a new predictor based on topic models, adapting surprisal to operationalize the influence of context beyond the sentence level. Topical surprisal for a word is defined as the weighted average of the word's probability given topic, where weights are the (posterior) probability for the topic in that context.

$$\text{surprisal}(w_i \text{ in context c}) = -\log \sum_{t \in \text{Topics}} P(w_i|t)P(t|c) \tag{2}$$

Fig. 1 illustrates how topical surprisal is computed using a sample excerpt from the text.

Topics are defined and probabilities estimated using an LDA topic model (Blei et al., 2003). Using the wrapper for Mallet LDA (McCallum) in the Gensim toolkit (Řehůřek and Sojka), we estimated a 100-topic model with the default hyperparameters using 219,380 documents from COCA, yielding P$(w|t)$ for all word-topic pairs and making it possible to compute the posterior topic probabilities P$(t|c)$ for any new document c. We compute topical surprisal for all the non-function words in the audio sample using the content in the 30-second window prior to the word to define the LDA "document" c.

4 Data Analysis

fMRI data is acquired with physical, biological constraints and we followed a standard preprocessing pipeline for fMRI imaging data that allowed us to make adjustments to improve the signal to noise ratio.[4]

The research questions presented above in §1 motivate two statistical analyses looking at correlations between model-based predictions and observed brain activity. In Analysis 1, we use ngram surprisal (from a 5gram language model) to instantiate local context and compare it against topical surprisal, which captures the influence of broader, topical context. As a follow-up, in Analysis 2, we use surprisal from a state-of-the-art LSTM language model to instantiate local context while still using topical surprisal for broader context, in order to illuminate potential differences between the neural correlates of ngram and LSTM models.

Measurements of neural activity using fMRI are based on an increase in blood flow to regions of the brain, which reflects increased cerebral activity. Because blood flow is slow relative to neural activity, this introduces a temporal lag and presents a

---

[2] Further details about the fMRI data collection can be found in the Appendix.

[3] A correlation matrix for the predictors is included in the Appendix.

[4] See the Appendix for further preprocessing details.

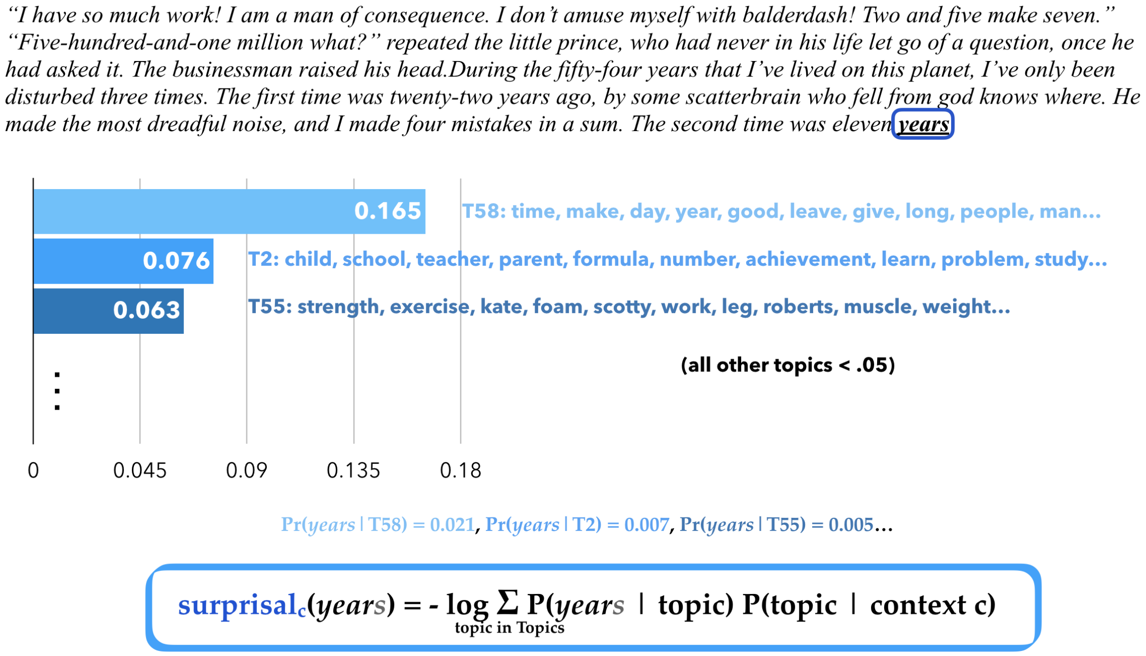

Figure 1: Real example illustrating the computation of topical surprisal. We calculate the probability of the word *years* conditioned on its topical context by using the previous 30 seconds of of the story to define a "document" c and computing its posterior topic distribution based on an LDA model for a large, diverse collection of English text. In the figure each topic is represented by its highest probability words.

challenge for modeling the time course of processing. To address this issue, predictors are convolved using a canonical hemodynamic response function (HRF) to model the observed time-course of the brain's hemodynamic response (BOLD - Blood Oxygenation Level Dependent) in each voxel.[5] Brennan (2016) provides a detailed description of how word-by-word predictors are convolved to estimate the fMRI BOLD signal in studies like the present one.

In order to look at correlations between predictors and brain activity, our analyses employ the General Linear Model (GLM; carried out using SPM12, Friston et al., 2007).[6] GLM is a hierarchical model with two levels that is typically used in fMRI data analysis (Poldrack et al., 2011), and its use within neuro-computational models of language processing for continuous, naturalistic fMRI studies is well-established (Brennan et al., 2012; Brennan, 2016; Willems et al., 2016; Bhattasali et al., 2018; Li et al., 2018; Bhattasali et al., 2019). At the first level of the GLM model, the data for each subject is modeled separately to calculate subject-specific parameter estimates and within-subject variance such that for each subject, a regression model is estimated for each voxel against the fMRI time series. The second-level model takes subject-specific parameter estimates as input and uses the between-subject variance to make statistical inferences about the larger population. The end result is a time series linear regression between the estimated fMRI BOLD signal and observed BOLD signals across the whole brain. Correlations between time series can then be computed with determinations of statistical significance, with suitable corrections for multiple comparisons.

4.1 Analysis 1: ngram surprisal vs. Topical surprisal

We regressed the word-by-word predictors against fMRI timecourses recorded during passive story-listening in a whole-brain analysis. The regressors were time-locked at the offset of each word in the audiobook. For each of the 15,388 words in the story, their timestamps were estimated using Praat TextGrids (Boersma and Weenink). Each word was annotated with its 5gram surprisal and the 6,243 non-function words were annotated with its topical surprisal values, as described in §3.5.

Additionally, we entered four regressors of non-interest into the GLM analysis: word-offset, word

---

[5] For more details about the hemodynamic response, see chapter 2 of Kemmerer (2014).

[6] Processing time on a Mac OS 10.13 takes 1.5 hours per subject and increases linearly with additional subjects.

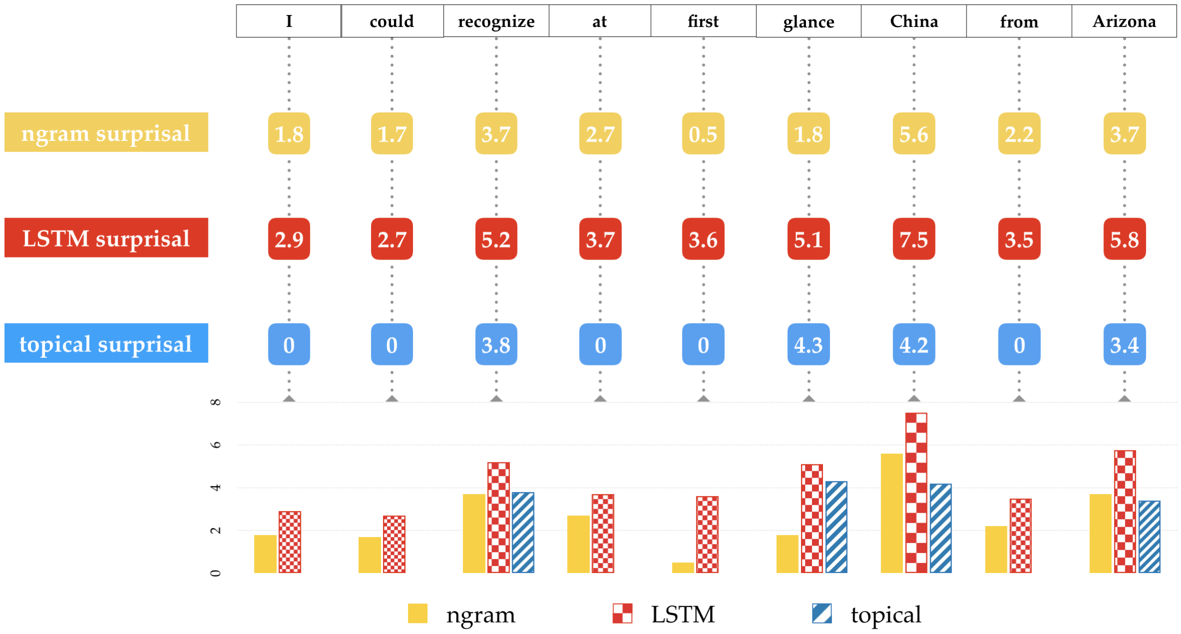

Figure 2: Comparing the word-by-word predictors on a single sentence: ngram surprisal (in yellow), neural surprisal (in red), topical surprisal (in blue). Values scaled for visualization purposes.

frequency, pitch, and intensity, which serve to improve the sensitivity, specificity and validity of activation maps (Bullmore et al., 1999; Lund et al., 2006). These were added to ensure that conclusions about lexical surprisal and topical surprisal would be specific to the cognitive processes they were taken to instantiate, as opposed to more general aspects of speech perception. These regressors were not orthogonalized.

4.2 Analysis 2: LSTM surprisal vs. Topical surprisal

Analysis 2 uses the same predictors as in Analysis 1, except that we use an LSTM language model to calculate lexical surprisal. Each word is annotated with its corresponding LSTM surprisal value (as described in §3.5), instead of 5gram surprisal, along with topical surprisal value given to the non-function words. These regressors were also not orthogonalized.

4.3 Group-level Analysis

In the second-level group analysis, each contrast was analyzed separately at the group-level. An 8 mm FWHM Gaussian smoothing kernel was applied on the contrast images from the first-level analysis to counteract inter-subject anatomical variation.

5 Results and Discussion

To begin with necessary details, behavioural results in the comprehension task confirmed that subjects were listening attentively to the auditory story presentation: across 51 participants, average accurate responses to the comprehension questions was 90% (SD = 3.7%). All whole-brain effects reported below survived a $p < 0.05$ Family-Wise-Error voxel correction for multiple comparisons which resulted in T-scores > 5.3. All brain region labels are from the Harvard-Oxford Cortical Structure Atlas.

Turning to the results of our analyses, functional localization identified using fMRI — via significant correlation with surprisal models — is interpreted to show which brain regions are recruited in processing the different types of contextual information captured by those models. To summarize, we observe a functionally distinct network that shows the difference between the influence of broad contextual cues and local contextual cues during sentence processing.

5.1 Analysis 1: Group-level results for ngram surprisal vs. topical surprisal

Whole-brain contrasts show that broad contextual cues and local contextual cues implicate different brain regions with no overlap (see Fig. 3). We observe a right-lateralized pattern of activation for topical surprisal (instantiating broad context) with

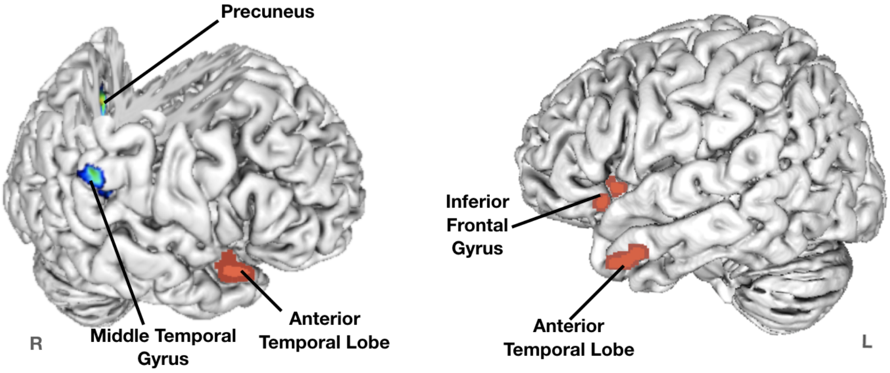

Figure 3: Whole brain contrast image with significant clusters for 5gram surprisal (in orange) and topical surprisal (in blue) after FWE voxel correction with p < 0.05. Table with significant clusters of peak activation included in Supplementary Materials.

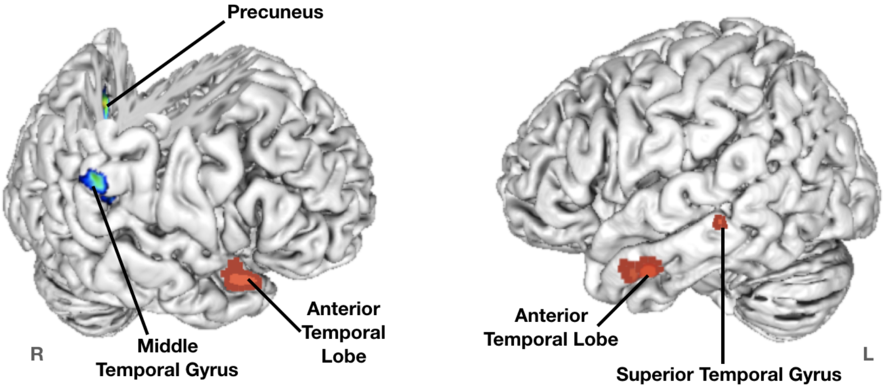

Figure 4: Whole brain contrast image with significant clusters for LSTM surprisal (in orange) and topical surprisal (in blue) after FWE voxel correction with p < 0.05. Table with significant clusters of peak activation are included in Supplementary Materials.

neural activation in the Precuneus and Middle Temporal Gyrus. For 5gram surprisal, the significant clusters are in the bilateral Anterior Temporal Lobe and left Inferior Frontal Gyrus.

5.2 Analysis 2: Group level results for LSTM surprisal vs. topical surprisal

Regression analyses localized the activation patterns for local and broad context to different areas (see Fig. 4). The peak activation for LSTM surprisal (instantiating local context) was observed in bilateral Anterior Temporal Lobe, along with a small cluster in left Superior Temporal Gyrus. Significant clusters for topical surprisal (instantiating broad context) were seen in the right Precuneus and right Middle Temporal Gyrus.

5.3 Discussion

In terms of Marr's (1982) levels, studies of the kind described here involve a linkage between proposals at the algorithmic-representational level and processing at the implementation level. Specifically, the logic of surprisal-based studies in computational cognitive neuroscience is based on the idea that when a word is less expected, it gives rise to increased brain activity due to increased cognitive load. Different instantiations of surprisal are used to model aspects of processing taking place during language comprehension, and correlations of surprisal with increased brain activity provide evidence about those aspects of the human comprehension process. Surprisal defined using ngrams embodies the use of sequential contextual representations during sentence processing. Syntactic surprisal embodies the use of hierarchical syntactic representations.

The present neuroimaging study introduces a new model, topical surprisal, which concerns the use of broader contextual information during processing — the topical probability of a word as defined in Eq (2) can be interpreted as an expected value of the word's probability given the topic being discussed, under the posterior topic distribution for the context. Our goal in this paper was to contrast context within a sentence with broader topical context. This can be viewed as a natural step in a progression from narrow ngram surprisal to sentential (syntactic and LSTM) surprisal to broader context that comprehenders might be taking into account.

The results in §5 based on our new model show that the patterns of activation for topical surprisal differ from those of lexical surprisal and syntactic surprisal, notably involving the right hemisphere. The centrality of the right Middle Temporal Gyrus and right Precuneus is consistent with previous studies demonstrating the role of those regions in broader contextual prediction during language comprehension: studies on narrative shifts (Whitney et al., 2009), contrasting sentences and narratives (Xu et al., 2005), contrasting coherent and incoherent narratives (Maguire et al., 1999), and sentences with and without preceding context (Raposo and Marques, 2013) that have found these same brain regions are involved in processing broader context and discourse-level information. The converging evidence confirms that our formalization of topical surprisal represents a cognitively plausible metric. Moreover, the neural correlates for topical surprisal corroborate previous work on lexical access and semantic integration (Binder et al., 2009; Graves et al., 2010; Hickok and Poeppel, 2007; Hagoort and Indefrey, 2014), suggesting that this broader contextual prediction is involved in these psycholinguistic processes beyond the sentence level.

Our novel approach to investigating contextual fit beyond the sentence level is also broadly consistent with prior results demonstrating how cortical hierarchy overlaps with language regions by using increasingly larger temporal receptive windows Lerner et al. (2011). Our results can be taken to support the argument that smaller versus larger temporal receptive windows implicate regions associated with lower-level and higher-level tasks respectively, a connection we plan to explore further.

Looking just at the ngram and LSTM models of lexical surprisal, our results provide additional corroboration for previous findings in the cognitive neuroscience literature involving sequential and syntactic processing (Willems et al., 2016; Brennan, 2016; Lopopolo et al., 2017; Shain et al., 2020). They also constitute an addition to the literature on understanding the linguistic properties captured by deep learning architectures. Numerous authors have shown that LSTM models capture not only sequential but also longer-distance structural dependencies (Linzen et al., 2016; Gulordava et al., 2018; van Schijndel and Linzen, 2018; Kuncoro et al., 2018; Futrell et al., 2019). In our study, we find that, while overlapping in the bilateral Anterior Temporal Lobe, the ngram and syntactic surprisal models also give rise to differently localized brain activity: the ngram model implicates the left Inferior Frontal Gyrus (IFG), while the LSTM surprisal model implicates the left Superior Temporal Gyrus (STG). The key observation here is that left-lateralized STG activity is uncontroversially correlated with syntactic processing (Pallier et al., 2011; Thompson and Meltzer-Asscher, 2014; Bhattasali et al., 2019; Shain et al., 2020). In contrast, the functional role of left IFG has variously been interpreted as involving combinatorial, sequential, or syntactic processes (Sahin et al., 2009; Snijders et al., 2009; Pallier et al., 2011; Brennan et al., 2016). The patterns of activity in this study therefore provide support from the implementation level for the idea that LSTMs are capturing aspects of syntactic representation that ngram models do not. Narrowing down the nature of those differences (e.g., sequential versus structural, or short- versus long-distance syntactic dependencies) remains a subject for future work.

6 Conclusion

The present study posed the questions of how broader topical context influences expectations in human sentence comprehension, and how local versus broader contexts might be processed differently in the brain. To address those questions we have introduced topical surprisal, a straightforward and intuitive extension to the highly productive surprisal-based paradigm in computational psycho- and neurolinguistics that employs topic modeling to estimate word probabilities conditioned on contexts beyond the sentence level.

Using analysis of fMRI brain imaging during naturalistic listening, we showed that the processing of broader topical context gives rise to neural activity in different brain regions than local contextual prediction as modeled using ngrams or an LSTM. The brain regions we identified turn out to be consistent with prior studies looking at neural correlates for processing of narratives and discourse.

In addition, we explored the neuro-anatomical

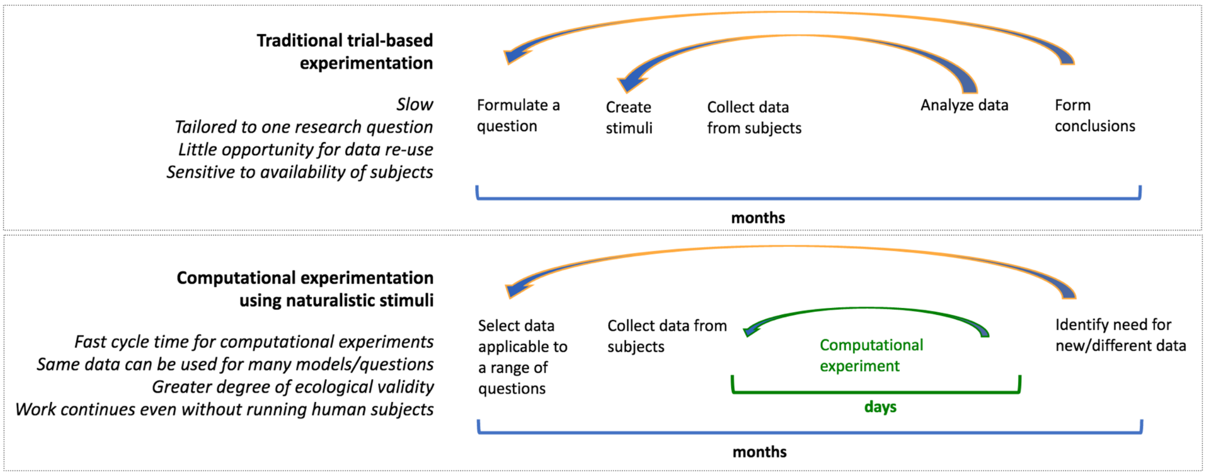

Figure 5: Visual comparison between two different experimental paradigms for cognitive neuroscience research

correlates of ngram and LSTM processing and obtained results that are consistent with claims in the deep learning literature regarding the sensitivity of LSTMs to long-distance syntactic structure.

More generally, this paper adds another data point demonstrating the relevance of tools from computational linguistics in cognitive neuroscience research (Brennan and Hale, 2019; Jain and Huth, 2018; Toneva et al., 2020) and the value of naturalistic stimuli in contextually situated and ecologically valid research (Maguire, 2012; Brennan, 2016; Hamilton and Huth, 2020).

Finally, we note that the paradigm we have employed here — computational modeling with previously collected natural-listening data — promotes reusability of datasets and replicability of results, and safeguards against unexpected delays in data collection such as a pandemic. Even more important, it offers a rapid experimental cycle dramatically better suited to computational research than traditional, trial-based methods in psycholinguistic and neurolinguistic research (Figure 5). As such, computational experimentation with naturalistic stimuli presents an invitation to computational linguists to collaborate with cognitive neuroscientists and apply their skills in operationalizing and testing hypotheses about neurocognitive mechanisms in sentence processing.

Ethical Considerations

This scientific study was reviewed and approved by the Cornell University Institutional Review Board, and human subject participation was conducted according to standard practices for research of this kind, including compensation of $65 for participants who completed the study. No personally identifiable data are stored. As is frequently the case for academic human subjects studies, our sample is drawn from a university population and may not be representative of the population at large. In addition, safety protocols required excluding participants with metal in their body (e.g. surgical implants, fresh tattoos), and following standard practice we included only participants with no history of psychiatric, neurological, or other medical illness or history of drug or alcohol abuse that might compromise cognitive functions, nor anyone taking antidepressant or psychoactive medications. Although the study was conducted in English, it included international and bilingual participants.

Acknowledgments

We would like to thank John Hale for his help in designing the study. We would also like to thank the audiences at AMLaP 2020, SNL 2020, the UMD Cognitive Neuroscience of Language group, and the MIT Computational Psycholinguistics group for their feedback on earlier stages of this study. We are also grateful to the reviewers for their thoughtful comments.

This material is based upon work supported by ONR MURI Award N00014–18–1–2670. The data was collected under NSF Grant No. 1607441.

References

Shohini Bhattasali, Murielle Fabre, and John Hale. 2018. Processing MWEs: Neurocognitive bases of verbal MWEs and lexical cohesiveness within MWEs. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and*

Constructions (LAW-MWE-CxG-2018), pages 6–17, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Shohini Bhattasali, Murielle Fabre, Wen-Ming Luh, Hazem Al Saied, Mathieu Constant, Christophe Pallier, Jonathan R Brennan, R Nathan Spreng, and John Hale. 2019. Localising memory retrieval and syntactic composition: An fMRI study of naturalistic language comprehension. *Language, Cognition and Neuroscience*, 34(4):491–510.

Jeffrey R. Binder, Rutvik H. Desai, William W. Graves, and Lisa L. Conant. 2009. Where is the semantic system? A critical review and meta-analysis of 120 functional neuroimaging studies. *Cerebral Cortex*, 19(12):2767–2796.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *the Journal of machine Learning research*, 3:993–1022.

Paul Boersma and David Weenink. *Praat: Doing phonetics by computer*.

Jonathan Brennan. 2016. Naturalistic sentence comprehension in the brain. *Language and Linguistics Compass*, 10(7):299–313.

Jonathan Brennan, Yuval Nir, Uri Hasson, Rafael Malach, David J Heeger, and Liina Pylkkänen. 2012. Syntactic structure building in the anterior temporal lobe during natural story listening. *Brain and language*, 120(2):163–173.

Jonathan R. Brennan and John T Hale. 2019. Hierarchical structure guides rapid linguistic predictions during naturalistic listening. *PloS one*, 14(1).

Jonathan R. Brennan, Edward P. Stabler, Sarah E. Van Wagenen, Wen-Ming Luh, and John T. Hale. 2016. Abstract linguistic structure correlates with temporal activity during naturalistic comprehension. *Brain and language*, 157:81–94.

Christian Brodbeck, L. Elliot Hong, and Jonathan Z. Simon. 2018. Rapid transformation from auditory to linguistic representations of continuous speech. *Current Biology*, 28(24):3976–3983.

Edward T. Bullmore, Michael J. Brammer, Sophia Rabe-Hesketh, Vivienne A. Curtis, Robin G. Morris, Steve C.R. Williams, Tonmoy Sharma, and Philip K. McGuire. 1999. Methods for diagnosis and treatment of stimulus-correlated motion in generic brain activation studies using fMRI. *Human brain mapping*, 7(1):38–48.

Mark Davies. 2008. *The Corpus of Contemporary American English (COCA): 560 million words, 1990–present*.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Allyson Ettinger, Naomi Feldman, Philip Resnik, and Colin Phillips. 2016. Modeling N400 amplitude using vector space models of word representation. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*, pages 1445–1450.

K.J. Friston, J. Ashburner, S.J. Kiebel, T.E. Nichols, and W.D. Penny, editors. 2007. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press.

Richard Futrell, Edward Gibson, Harry J. Tily, Idan Blank, Anastasia Vishnevetsky, Steven T. Piantadosi, and Evelina Fedorenko. 2021. The Natural Stories corpus: a reading-time corpus of English texts containing rare syntactic constructions. *Language Resources and Evaluation*, 55(1):63–77.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

William W. Graves, Jeffrey R. Binder, Rutvik H. Desai, Lisa L. Conant, and Mark S. Seidenberg. 2010. Neural correlates of implicit and explicit combinatorial semantic processing. *Neuroimage*, 53(2):638–646.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Peter Hagoort and Peter Indefrey. 2014. The neurobiology of language beyond single words. *Annual review of neuroscience*, 37:347–362.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

Liberty S. Hamilton and Alexander G. Huth. 2020. The revolution will not be controlled: Natural stimuli in speech neuroscience. *Language, Cognition and Neuroscience*, 35(5):573–582.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696.

John M. Henderson, Wonil Choi, Matthew W. Lowder, and Fernanda Ferreira. 2016. Language structure in the brain: A fixation-related fMRI study of syntactic surprisal in reading. *Neuroimage*, 132:293–300.

Gregory Hickok and David Poeppel. 2007. The cortical organization of speech processing. *Nature Reviews Neuroscience*, 8(5):393–402.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Shailee Jain and Alexander G. Huth. 2018. Incorporating context into language encoding models for fMRI. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6629–6638.

Katerina D. Kandylaki and Ina Bornkessel-Schlesewsky. 2019. From story comprehension to the neurobiology of language. *Language, Cognition and Neuroscience*, 34(4):405–410.

David Kemmerer. 2014. *Cognitive neuroscience of language*. Psychology Press.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Yulia Lerner, Christopher J. Honey, Lauren J. Silbert, and Uri Hasson. 2011. Topographic mapping of a hierarchy of temporal receptive windows using a narrated story. *Journal of Neuroscience*, 31(8):2906–2915.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Jixing Li, Murielle Fabre, Wen-Ming Luh, and John Hale. 2018. The role of syntax during pronoun resolution: Evidence from fMRI. In *Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing*, pages 56–64, Melbourne. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Alessandro Lopopolo, Stefan L. Frank, Antal Van den Bosch, and Roel M. Willems. 2017. Using stochastic language models (SLM) to map lexical, syntactic, and phonological information processing in the brain. *PloS one*, 12(5):e0177794.

Torben E. Lund, Kristoffer H. Madsen, Karam Sidaros, Wen-Lin Luo, and Thomas E. Nichols. 2006. Non-white noise in fMRI: does modelling have an impact? *Neuroimage*, 29(1):54–66.

Eleanor A Maguire. 2012. Studying the freely-behaving brain with fMRI. *Neuroimage*, 62(2):1170–1176.

Eleanor A. Maguire, Christopher D. Frith, and R. G. M. Morris. 1999. The functional neuroanatomy of comprehension and memory: The importance of prior knowledge. *Brain*, 122(10):1839–1850.

David Marr. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. The MIT Press.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit.

James Michaelov and Benjamin Bergen. 2020. How well does surprisal explain n400 amplitude under different experimental conditions? In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 652–663, Online. Association for Computational Linguistics.

Kanishka Misra, Allyson Ettinger, and Julia Rayz. 2020. Exploring BERT's sensitivity to lexical cues using tests from semantic priming. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4625–4635, Online. Association for Computational Linguistics.

Richard C. Oldfield. 1971. The assessment and analysis of handedness: the Edinburgh inventory. *Neuropsychologia*, 9(1):97–113.

Christophe Pallier, Anne-Dominique Devauchelle, and Stanislas Dehaene. 2011. Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*, 108(6):2522–2527.

Jonathan W. Peirce. 2007. PsychoPy – Psychophysics software in Python. *Journal of Neuroscience Methods*, 162(1):8–13.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Russell A. Poldrack, Jeanette A. Mumford, and Thomas E. Nichols. 2011. *Handbook of functional MRI data analysis*. Cambridge University Press.

Ana Raposo and J. Frederico Marques. 2013. The contribution of fronto-parietal regions to sentence comprehension: Insights from the Moses illusion. *NeuroImage*, 83:431–437.

Radim Řehůřek and Petr Sojka. Gensim – Statistical semantics in Python.

Ned T. Sahin, Steven Pinker, Sydney S. Cash, Donald Schomer, and Eric Halgren. 2009. Sequential processing of lexical, grammatical, and phonological information within Broca's area. *Science*, 326(5951):445–449.

Marten van Schijndel and Tal Linzen. 2018. A neural model of adaptation in reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4704–4710, Brussels, Belgium. Association for Computational Linguistics. `https://github.com/vansky/neural-complexity`.

Cory Shain, Idan Asher Blank, Marten van Schijndel, William Schuler, and Evelina Fedorenko. 2020. fMRI reveals language-specific predictive coding during naturalistic sentence comprehension. *Neuropsychologia*, 138:107307.

Tineke M. Snijders, Theo Vosse, Gerard Kempen, Jos J.A. Van Berkum, Karl Magnus Petersson, and Peter Hagoort. 2009. Retrieval and unification of syntactic structure in sentence comprehension: An fMRI study using word-category ambiguity. *Cerebral cortex*, 19(7):1493–1503.

Miloš Stanojević, Shohini Bhattasali, Donald Dunagan, Luca Campanelli, Mark Steedman, Jonathan Brennan, and John Hale. 2021. Modeling incremental language comprehension in the brain with Combinatory Categorial Grammar. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 23–38, Online. Association for Computational Linguistics.

Cynthia K Thompson and Aya Meltzer-Asscher. 2014. Neurocognitive mechanisms of verb argument structure processing. *Structuring the argument*, pages 141–168.

Mariya Toneva, Tom M. Mitchell, and Leila Wehbe. 2020. Combining computational controls with natural text reveals new aspects of meaning composition. *bioRxiv*.

Mariya Toneva and Leila Wehbe. 2019. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). In *Advances in Neural Information Processing Systems*, pages 14928–14938.

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. 2014. Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243, Doha, Qatar. Association for Computational Linguistics.

Carin Whitney, Walter Huber, Juliane Klann, Susanne Weis, Sören Krach, and Tilo Kircher. 2009. Neural correlates of narrative shifts during auditory story comprehension. *Neuroimage*, 47(1):360–366.

Roel M. Willems. 2015. Introduction. *Cognitive neuroscience of natural language use*, pages 1–7.

Roel M. Willems, Stefan L. Frank, Annabel D. Nijhof, Peter Hagoort, and Antal Van den Bosch. 2016. Prediction during natural language comprehension. *Cerebral Cortex*, 26(6):2506–2516.

Jiang Xu, Stefan Kemeny, Grace Park, Carol Frattali, and Allen Braun. 2005. Language in context: Emergent features of word, sentence, and narrative comprehension. *Neuroimage*, 25(3):1002–1015.

Shulin Zhang. 2020. Human brain networks for semantic roles. Master's thesis, University of Georgia.

Appendix for Using surprisal and fMRI to map the neural bases of broad and local contextual prediction during natural language comprehension

1 Data Collection

Imaging was performed using a 3T MRI scanner (Discovery MR750, GE Healthcare, Milwaukee, WI) with a 32-channel head coil at the Cornell MRI Facility. Blood Oxygen Level Dependent (BOLD) signals were collected using a T2 -weighted echo planar imaging (EPI) sequence (repetition time: 2000 ms, echo time: 27 ms, flip angle: 77deg, image acceleration: 2X, field of view: 216 x 216 mm, matrix size 72 x 72, and 44 oblique slices, yielding 3 mm isotropic voxels). Anatomical images were collected with a high resolution T1-weighted (1 x 1 x 1 mm^3 voxel) with a Magnetization-Prepared RApid Gradient-Echo (MP-RAGE) pulse sequence.

2 Preprocessing

Primary preprocessing steps were carried out in AFNI version 16 (Cox, 1996) and include motion correction, coregistration, and normalization to standard MNI space. After the previous steps were completed, ME-ICA (Kundu et al., 2012) was used to further preprocess the data. ME-ICA is a denoising method which uses Independent Components Analysis to split the T2*-signal into BOLD and non-BOLD components. Removing the non-BOLD components mitigates noise due to motion, physiology, and scanner artifacts (Kundu et al., 2017).

3 Correlation Matrix for Predictors

Fig. 1 shows the correlation matrix for the three surprisal predictors.

4 Group-level results

Table 1 shows the significant clusters of activation for topical surprisal using brain region labels from the Harvard-Oxford Cortical Structure Atlas.

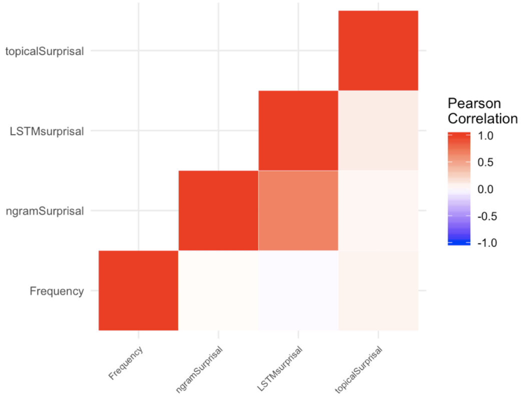

Figure 1: Correlation matrix (Pearson's r) of the convolved regressors included in the GLM models reported in Analysis 1 and Analysis 2.

Regions	Cluster size (in voxels)	MNI Coordinates x	y	z	p-value (corrected)	T-score (peak-level)
R Middle Temporal Gyrus	497	50	-50	18	0.000	8.46
R Precuenus	253	10	-62	30	0.011	5.83

Table 1: Significant clusters for topical surprisal after FWE voxel correction with $p < 0.05$. Peak activation is given in MNI coordinates and p-values are reported at peak-level after voxel correction.

References

Robert W. Cox. 1996. AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical research*, 29(3):162–173.

Prantik Kundu, Souheil J. Inati, Jennifer W. Evans, Wen-Ming Luh, and Peter A. Bandettini. 2012. Differentiating bold and non-bold signals in fMRI time series using multi-echo epi. *Neuroimage*, 60(3):1759–1770.

Prantik Kundu, Valerie Voon, Priti Balchandani, Michael V. Lombardo, Benedikt A. Poser, and Peter A. Bandettini. 2017. Multi-echo fMRI: A review of applications in fMRI denoising and analysis of bold signals. *NeuroImage*, 154:59 – 80.

Assessing the Syntactic Capabilities of Transformer-based Multilingual Language Models

Laura Pérez-Mayos[1], Alba Táboas García[1], Simon Mille[1], Leo Wanner[2,1]
[1] TALN Research Group, Pompeu Fabra University, Barcelona, Spain
[2] Catalan Institute for Research and Advanced Studies (ICREA), Barcelona, Spain
`{laura.perezm|alba.taboas|simon.mille|leo.wanner}@upf.edu`

Abstract

Multilingual Transformer-based language models, usually pretrained on more than 100 languages, have been shown to achieve outstanding results in a wide range of cross-lingual transfer tasks. However, it remains unknown whether the optimization for different languages conditions the capacity of the models to generalize over syntactic structures, and how languages with syntactic phenomena of different complexity are affected. In this work, we explore the syntactic generalization capabilities of the monolingual and multilingual versions of BERT and RoBERTa. More specifically, we evaluate the syntactic generalization potential of the models on English and Spanish tests, comparing the syntactic abilities of monolingual and multilingual models on the same language (English), and of multilingual models on two different languages (English and Spanish). For English, we use the available SyntaxGym test suite; for Spanish, we introduce SyntaxGymES, a novel ensemble of targeted syntactic tests in Spanish, designed to evaluate the syntactic generalization capabilities of language models through the SyntaxGym online platform.

1 Introduction

Transformer-based neural models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), DistilBERT (Sanh et al., 2019), XLNet (Yang et al., 2019), etc. are excellent learners. They have been shown to capture a range of different types of linguistic information, from morphological (Edmiston, 2020) over syntactic (Hewitt and Manning, 2019) to lexico-semantic (Joshi et al., 2020). A particularly significant number of works study the degree to which these models capture and generalize over (i.e., learn to instantiate correctly in different contexts) syntactic phenomena, including, e.g., subject-verb agreement, long distance dependencies, garden path constructions, etc. (Linzen et al., 2016; Marvin and Linzen, 2018; Futrell et al., 2019; Wilcox et al., 2019a). However, most of these works focus on monolingual models, and, if the coverage of syntactic phenomena is considered systematically and in detail, it is mainly for English, as, e.g., (Hu et al., 2020a). This paper aims to shift the attention from monolingual to multilingual models and to emphasize the importance to also consider the syntactic phenomena of languages other than English when assessing the generalization potential of a model. More specifically, it systematically assesses how well multilingual models are capable to generalize over certain syntactic phenomena, compared to monolingual models, and how well they can do it not only for English, but also for Spanish.

Multilingual models such as mBERT (multilingual BERT, (Devlin et al., 2019)), XLM (Lample and Conneau, 2019) and XLM-R (Conneau et al., 2020) proved to achieve outstanding performance on cross-lingual language understanding tasks, including on low-resource languages for which only little training data is available. However, these models face the risk of running into what Conneau et al. (2020) refer to as "curse of multilinguality": adding languages to the model increases the performance on low-resource languages up to a point, after which the overall performance on monolingual and cross-lingual benchmarks degrades. The question is thus whether, and if yes to what degree, this degradation affects the syntactic generalization potential of multilingual models across languages.

The reason to extend the evaluation to other languages (in our case, Spanish) is that many existing syntactic phenomena such as determiner and adjective agreement within the noun phrase, subject pro-drop, or flexible word order – to name only a few – are not prominent or do not exist in English, while in Spanish all of them do.

Our evaluation methodology is similar to that by

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3799–3812
August 1–6, 2021.

Hu et al. (2020a), who test 20 model type combinations and data sizes on 34 English syntactic test suites, and find substantial differences in the syntactic generalization performance across different models. We draw upon their tests to test the syntactic generalization potential of monolingual and multilingual transformer-based models for English, and upon the Spanish SyntaxGym introduced in this paper for Spanish. Tu run the tests, we use the SyntaxGym toolkit (Gauthier et al., 2020).

Our results show that, indeed, there is a substantial difference between the syntactic generalization potential of monolingual and multilingual models. But this difference depends on the language: While for English monolingual models (BERT and RoBERTa) offer a higher syntactic generalization than multilingual models (mBERT and XLM-R), this is not the case for Spanish, for which multilingual models (XLM-R) generalize better. Furthermore, multilingual models do not generalize equally well across languages, with mBERT generalizing, in general, better in English and XLM-R better in Spanish. Our experiments also show that it depends on the language how well a multilingual model captures a specific syntactic phenomenon such as, e.g., Agreement, Center-embedding or Garden Path.

The remainder of the paper is structured as follows. Section 2 introduces the work that is related to ours in terms of the evaluation methodology and, in particular, in terms of the assessment of multilingual language models. Section 3 describes the English test suites, and presents the novel Spanish SyntaxGym test suites. Section 4 details the models that we tested and outlines how we use them to evaluate the probability of a text sequence. Section 5 offers a detailed analysis of the syntactic generalization abilities of the monolingual and multilingual versions of BERT and RoBERTa, and Section 6 summarizes the implications that our work has for the use of multilingual language models.

2 Related Work

Our work on the evaluation of the capability of monolingual and multilingual transformer-based LMs to capture syntactic information is in line with a number of previous works, including, e.g., those that are based on psycholinguistic experiments, focusing on highly specific measures of language modeling performance and allowing to distinguish models with human-like representations of syntactic structure (Linzen et al., 2016; Lau et al., 2017; Gulordava et al., 2018; Marvin and Linzen, 2018; Futrell et al., 2019). Supervised probing models have been used to test for the presence of a wide range of linguistic phenomena (Conneau et al., 2018; Hewitt and Manning, 2019; Liu et al., 2019a; Tenney et al., 2019; Voita and Titov, 2020; Elazar et al., 2020). Warstadt et al. (2020) isolate specific phenomena in syntax, morphology, and semantics, finding that state-of-the-art models struggle with some subtle semantic and syntactic phenomena, such as negative polarity items and extraction islands.

Recently, a number of works also address the cross-language assessment of models. Hu et al. (2020b) introduces XTREME, a multi-task benchmark for evaluating the cross-lingual generalization capabilities of multilingual representations across 40 languages and 9 tasks. They show that while XLM-R reduces the difference between the performance on the English test set and all other languages compared to mBERT for tasks such as XQuAD and MLQA, it does not have the same impact on structured prediction tasks such as PoS and NER. Mueller et al. (2020) introduces a set of subject-verb agreement tests, showing that mBERT performs better than English BERT on Sentential Complements, Short VP Coordination, and Across a Prepositional Phrase, but worse on Within-an-Object Relative Clause, Across-an-Object Relative Clause and in Reflexive Anaphora Across a Relative Clause, and offers high syntactic accuracy on English, but noticeable deficiencies on other languages, most notably on those that do not use Latin script, as also noted by Hu et al. (2020b). On the same line, Rönnqvist et al. (2019) concludes that mBERT is not able to substitute a well-trained monolingual model in challenging tasks.

As already mentioned in Section 1, Hu et al. (2020a) assembled a set of English syntactic tests in order to assess the syntactic generalization potential of a number of different neural LMs (LSTM, ON-LSTM, RNNG and GPT-2). The tests are accessible through the SyntaxGym toolkit (Gauthier et al., 2020); cf. also Section 3.1. Our methodology is analogous, although our objective is different. Rather than comparing the performance of several monolingual models, we contrast the performance of monolingual and multilingual transformer-based models. Furthermore, while their only test suite source is the English SyntaxGym, we create and use also a Spanish SyntaxGym; cf. Section 3.2.

3 Test Suites

For the English test suites, we used SyntaxGym,[1] an online platform that compiles a variety of linguistic tests used by Hu et al. (2020a) to assess the syntactic coverage of language models. It contains 34 suites, grouped into 6 different so-called *circuits*, a classification based on what is required from the models to process the targeted constructions. For the Spanish test suites, we created SyntaxGymEs, adapting 11 of the existing suites for English and building 15 new ones, including a whole new circuit. In what follows, we first introduce the original English SyntaxGym and then present in detail the novel SyntaxGymEs.

3.1 SyntaxGym for English

The tests in the SyntaxGym designed by Hu et al. (2020a) (henceforth also referred to as "English SyntaxGym") are based on the notion of *surprisal*. A sequence of words is given to a language model, which assigns a probability to each of the following candidate words. Given the syntactic properties of the considered language, some candidate words are less surprising than others, and so should be predicted by a language model. For instance, after the sequence *The cat*, the inflected word *sleeps* should be less surprising than *sleep*.

Each test consists of a list of ITEMS that vary in a controlled way according to a set of CONDITIONS determined by the experimental design. The other main component is a series of PREDICTIONS comparing surprisal values in specific regions of the items across conditions. If the relevant syntactic generalization has been learned by the model, the predictions should hold.

Moreover, some tests have versions with MODIFIERS, in which additional clauses or phrases have been embedded inside each item. These modifiers increase the linear distance between two co-varying items, making the task harder. Sometimes they also include a distractor word in the middle of a syntactic dependency, which can lead the models to misinterpret the dependency.

The test suites are arranged in terms of the following *circuits*:

• **Agreement**: Morphosyntactic phenomena that occur when the features of an item constrain another item to adopt a specific form. This is a marginal phenomenon in English, so the original circuit only includes 3 test suites on *Subject-verb number agreement*, all of them with modifiers (Marvin and Linzen, 2018).

• **Licensing**: A construction's need for the presence of a *licensor* to allow its occurrence in a sentence. The circuit consists of 4 suites on *Negative polarity items* (2 of them with modifiers) and 6 on *Reflexive pronouns* (all of them with modifiers), also from Marvin and Linzen (2018).

• **Center embedding**: Subordinate clauses that sit in the middle of their superordinate clause, creating nested dependencies. This circuit contains 2 test suites: *Center embedding* and *Center embedding with modifier*, from Wilcox et al. (2019a).

• **Long-distance dependencies (LDDs)**: LDDs occur when two constituents that are syntactically related do not appear adjacent to one another, but at a longer distance from one another. The circuit includes 6 suites on *Filler-gap dependencies* (2 with modifiers and 4 addressing extraction and hierarchy) from Wilcox et al. (2018) and Wilcox et al. (2019b), and 2 suites on *Cleft structure* that were first introduced in (Hu et al., 2020a).

• **Gross syntactic expectation**: Expectation for a large syntactic structure usually induced by subordinating adverbs or conjunctions. 4 test suites on *Subordination* (from Futrell et al. (2018), 3 of them with modifiers) constitute the circuit.

• **Garden path effects**: Effects that emerge when an incorrect but locally likely parse needs to be abandoned in favor of the correct one, once a specific word appears in the sentence. Two such effects are considered in this circuit: *Main verb/reduced relative clause (MVRR)* and *NP/Z garden paths*, with respectively 2 and 4 suites, all from Futrell et al. (2018).

3.2 SyntaxGymES: SyntaxGym for Spanish

For Spanish, we expand the tests in (Hu et al., 2020a) so as to cover language-specific phenomena. In this section, we detail which of the original tests we retained, which ones we modified, and which ones we added within each original circuit. A whole new circuit regarding the linear order of a sentence's basic constituents was also added, since flexibility in this respect is a characteristic that distinguishes Spanish (and other Romance languages) from English. For a more detailed description with examples and predictions, see the Supplementary Material; upon acceptance of the paper, Syntax-

---

[1] `http://syntaxgym.org/`

GymES will be published in the SyntaxGym platform http://syntaxgym.org/.

3.2.1 Notation

We follow the usual notations in linguistic literature. An asterisk '*' preceding an example signals that the sentence is ungrammatical, it violates some principle or constraint. A question mark '?' is used to indicate a marginal sentence, i.e., a sentence that is grammatical but very uncommon or that requires a non-straightforward interpretation. The exclamation mark '!' indicates a highly difficult sentence to process for the human mind.

3.2.2 Agreement

Unlike English, Spanish is a morphologically rich language, and as such it presents many morphosyntactic phenomena related to agreement. For this reason, out of the six original circuits, **Agreement** was the one that underwent the most changes.

Regarding verbal agreement (constraints imposed on the verb by the subject), we adapted two existing test suites, **Subject-Verb Agreement with Object Relative Clause** and **Subject-Verb Agreement with Subject Relative Clause**, and created a new one, **Basic Subject-Verb Agreement**, in which both person and number features were taken into consideration.

(1) Tú cocinas
you.2SG cook.2SG

(2) * Tú cocinais/cocino/cocinan
you.2SG cook.2PL/1SG/3PL

As for nominal agreement (constraints that a noun's gender and number features can impose on the form of other words in the sentence), we also created several new test suites: **Determinant-Noun Agreement** simply pairs a noun with the four possible forms of the definite article (*el*, *la*, *los*, *las*), while **Adjective-Noun Agreement** pairs a noun with the four possible forms of an adjective that modifies it (we excluded articles to avoid providing extra information).

(3) La tienda vende discos usados
the store sells disc.M.PL used.M.PL

(4) * La tienda vende discos
the store sells disc.M.PL
usados/usado/usadas/usada
used.M.PL/M.SG/F.PL/F.SG

In addition to these two suites, we built similar ones for **Attribute Agreement** in copulative constructions, to which we added two versions with object or subject relative clauses as modifiers, and also for **Predicative Agreement** in constructions with subject or object predicative complement. The only difference here is that the two words that must agree are not adjacent anymore. In terms of predictions, the verb/noun with matching features should have a lower surprisal than the others, and the verb/noun that matches only one feature should have a lower surprisal that the one that doesn't match any.

3.2.3 Center Embedding

For this circuit, we adapted to Spanish the two existing test suites in English, creating **Center Embedding** and **Center Embedding with PP modifier**. In the basic suite, a relative clause is center embedded after the subject of the main clause. Verb transitivity and subject-verb plausibility are used to test if the models are capable of retaining the relevant information and predicting the verbs in the correct order.

3.2.4 Gross Syntactic Expectation

From the four original suites in this circuit, we adapted three of them: **Subordination**, and two of its versions with modifiers, **Subordination with Object Relative Clause** and **Subordination with Subject Relative Clause**. Given a sentence that starts with a typically subordinating adverb or conjunction, these suites test the models' ability to maintain the expectation for the onset of a matrix clause for as long as the subordinate one lasts.

3.2.5 Long-distance Dependencies

Filler-gap dependencies are an example of LDDs. They occur when a phrase (the filler) is realized somewhere in the sentence, but is semantically interpreted at some other point (the gap). For this circuit, we created a **Basic Filler-Gap Dependencies** test and adapted from the original English circuit a version that includes modifiers, **Filler-Gap Dependencies with Three Sentencial Embeddings**. Embedding three sentences between filler and gap makes the task more challenging. We also adapted to Spanish the novel **Pseudo-Cleft Structures** suite introduced in (Hu et al., 2020a).

3.2.6 Garden Path Effects

The Garden Path effect can be created by several syntactic ambiguities that differ cross-linguistically. The Main Verb/Reduced Relative garden path effect was the subject of two suites in the original

English circuit, but it does not translate to Spanish, so those suites were not included in Spanish SyntaxGym.

On the other hand, the ambiguity responsible for NP/Z also holds for Spanish. Here, an NP is initially interpreted as the object in a subordinate clause when it actually is the subject of the main clause (the subordinate clause having a Zero/null object). The ambiguity can be prevented with a comma, but also by placing an overt object in the subordinate clause, as is done in **NP/Z Garden Path Effect (with Overt Object)**, or by substituting its verb with a pure intransitive verb, as is done in **NP/Z Garden Path Effect (with Intransitive Verb)**. Both suites correspond to Spanish adaptations of the two original suites regarding this effect.

(5) !Mientras ella leía sus manuscritos se volaron por la ventana.

!'While she read her manuscripts went out the window.'

(6) Mientras ella [dormía]/[leía un libro]/[leía,] sus manuscritos se volaron por la ventana.

'While she [slept]/[read a book]/[read,] her manuscripts went out the window.'

3.2.7 Licensing

Negative polarity items (NPIs), like *any* or *ever* in English, are examples of words that need to be licensed by negation. Since Spanish NPIs do not function exactly in the same way, we took the original NPI Licensing test as inspiration and created two new suites: **Negative Polarity Items** and **NPIs and Polarity Agreement**.

Constructions with verbs in subjunctive mood also require the presence of a licensor. In Spanish, a verb expressing feelings (e.g. of joy, surprise, pleasantness) in the main clause, creates the expectation for subjunctive mood in the subordinate clause. This was the basis for a new test suite: **Subjunctive Mood and Verbs that Express Feeling**.

(7) Espero que mañana llueva/*lloverá.
(I)hope that tomorrow rain.SUB/will.rainIND

'I hope it [rains]/[will rain] tomorrow.'

The other new suite in this circuit, **Subjunctive Mood, Negation and Belief Verbs**, relies on the fact that belief verbs can also license subjunctive mood, but only when combined with negation:

(8) No creo que mañana
NEG (I)believe that tomorrow
llueva/*lloverá.
rain.SUB/will.rain.IND

'I don't think it [rain]/[will rain] tomorrow.'

(9) Creo que mañana no
(I)believe that tomorrow NEG
lloverá/*llueva.
will.rain.IND/rain.SUB

'I think it [won't]/[don't] rain tomorrow.'

3.2.8 Linearization

One of the main syntactic distinctions between languages is constituent order within the sentence. But, in addition to the canonical order in which these elements appear, languages also differ in their flexibility to alter that order. Spanish allows some flexibility, which was the basis for three new test suites.

For **Subject–Auxiliary Verb–Main Verb Linearization**, the possibility to postpone the subject is compared with the rigidity of the relation between main and auxiliary verb, which must be adjacent and do not allow inversion:

(10) Juan [ha comido]/*[comido ha].

'John [has eaten]/[eaten has].'

(11) Ha [comido Juan]/*[Juan comido].

*'Has [eaten John]/[John eaten].'

In the **Subject–Verb–Object Linearization** test, we compare the phenomenon in affirmative versus interrogative sentences. In Spanish, word order flexibility holds for affirmative sentences, but not for interrogative ones, where subject-verb inversion is compulsory:

(12) Ana compró un libro. / Compró un libro Ana.

'Ann bought a book. / Bought a book Ann.'

(13) ¿Qué compró Ana? / *¿Qué Ana compró?

'What did Ann buy? / *What Ana did buy?'

Word order variations also appear within the NP, as captured by the **Noun-Adjective and Noun-PP Linearization** test. Contrary to English, Spanish adjectives usually come after the noun. But again, the language allows for some flexibility and they can be swapped. This possibility, however, does not apply to other noun modifiers like prepositional phrases:

(14) Construyó una [mesa robusta]/[robusta mesa].

'He built a [sturdy table]/[table sturdy].'

(15) Construyó una [mesa de madera]/*[de madera mesa].

'He built a [wooden table]/*[table wooden].'

4 Experiments

We test the base cased versions of BERT and mBERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b) and XLM-R (Conneau et al., 2020) on the English SyntaxGym and BETO (Canete et al., 2020), mBERT and XLM-R on the Spanish SyntaxGym. To run the experiments, we use the SyntaxGym toolkit (Gauthier et al., 2020).

4.1 Experimental Setup

The SyntaxGym test suites are designed from the perspective of sentence generation, i.e., with the hypothesis that if a model has correctly learned some relevant syntactic generalization, it should assign higher probability to grammatical and natural continuations of sentences. This requires asking the models to predict the next token given a context of previous tokens, in a left-to-right generative fashion. However, BERT-based and RoBERTa-based families of models (in our case, BERT and mBERT on the one side, and RoBERTa and XLM-R on the other side) are bidirectional, they are trained with a masked language modeling objective to predict a word given its left and right context. In this work, we follow Wang and Cho (2019)'s sequential sampling procedure to evaluate the probability of a text sequence, encoding unidirectional context in the forward direction. To compute the probability distribution for a sentence with N tokens, we start with a sequence of $N+2$ tokens: a *begin_of_sentence* token plus $N+1$ *mask* tokens, where the last *mask* corresponds to the *end_of_sentence* token. For each token position i in $[1, N]$, we compute the probability distribution over the vocabulary given the left context of the original sequence, and select the probability assigned by the model to the original word.

For example, in an agreement test with the sentence '*The girls run fast.*', a model that has properly learned agreement should assign a higher probability to *run* than to *runs* for the third word. In order to test it, we feed the tokens sequence *[[bos] [The] [girls] [mask] [mask] [mask] [mask]]* to the model, and compare the probabilities assigned by the model to *run* and *runs* for position 4.

	Average SG performance	
Model	**English**	**Spanish**
BERT	77.80	—
RoBERTa	82.04	—
mBERT	77.55	72.31
XLM-R	71.84	78.50
BETO	—	67.92

Table 1: Average SG score by model class for the English and Spanish tests.

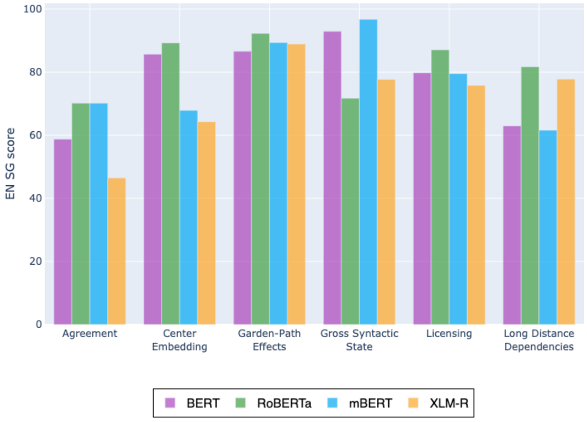

Figure 1: Performance accuracy across English circuits

4.2 Results of the experiments

This section summarizes the results of our experiments that aim to: (i) contrast the performance of monolingual and multilingual models on English and Spanish and (ii) provide insights on the performance of the multilingual models across languages.

Table 1 shows the average SyntaxGym (SG) performance of the evaluated monolingual and multilingual models on the English and Spanish SyntaxGyms. Figures 1 and 2 zoom in on the performance of the tested models with respect to specific circuits for English and Spanish respectively.

Six of the English test suites (Center Embedding, Cleft structure, MVRR, NPZ-Verb, NPZ-Object, Subordination) and five of the Spanish test suites (Attribute Agreement, Basic Subject-Verb Agreement, Subordination, Center Embedding, Basic Filler-Gap Dependencies) include tests with and without modifiers, i.e,. intervening content inserted before the critical region. Figures 3 and 4 show the models' average scores in these test suites, without modifiers (dark bars) and with modifiers (light bars), evaluating how robust each model is with

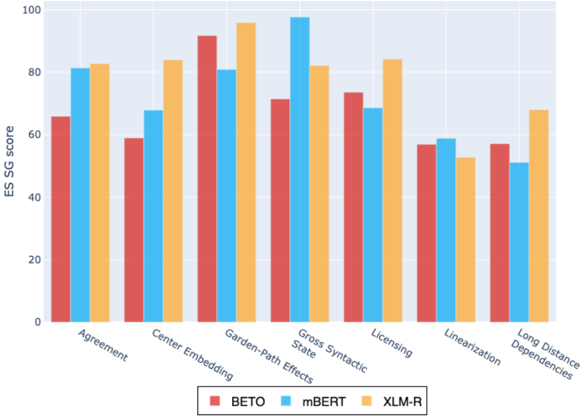

Figure 2: Performance accuracy across Spanish circuits

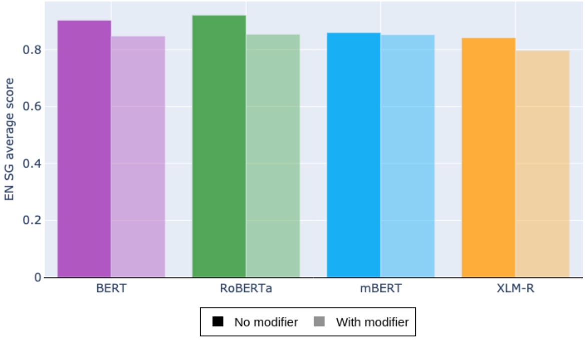

Figure 3: Models average English SG score in Center Embedding, Cleft structure, MVRR, NPZ-Verb, NPZ-Object and Subordination, with and without modifiers.

respect to the corresponding content.

5 Discussion

Let us assess in detail the results of the experiments from above. In what follows, we compare the performance of monolingual with the performance of multilingual models and analyze the cross-language performance of multilingual models, as well as the stability of the individual models with respect to modifiers.

5.1 Monolingual vs multilingual models

RoBERTa shows an overall higher performance than the other models for English (Table 1). This is not surprising since it is trained on 10 times more data than BERT, and it has been shown to improve over BERT in many NLU tasks. However, while mBERT does not seem to lose performance compared to BERT, XLM-R loses around 10 points compared to RoBERTa. As XLM-R is specifically designed to offer a more balanced performance across languages, with a special focus on low-resource languages, it appears natural that it loses some performance on high-resource languages such as English. For Spanish, the multilingual models clearly outperform the monolingual model. This is likely due to the fact that while BETO and mBERT are of comparable size and are trained with the same amount of data (16GB), BETO is only trained with a Masked Language Modeling (MLM) objective, and mBERT is trained on MLM and Next Sentence Prediction (NSP). On the other hand, XLM-R is also only trained on MLM, but it is trained on more than 2TB of data, 53 GB corresponding to Spanish data.

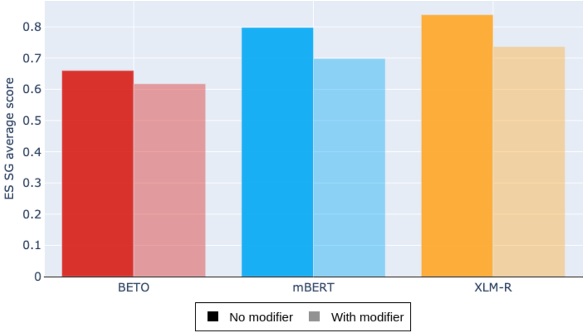

Figure 4: Models average Spanish SG score in Attribute Agreement, Subject-Verb Agreement, Subordination, Center Embedding and Filler-Gap Dependencies, with and without modifiers.

RoBERTa outperforms all other models in all the English circuits (cf. Figure 1), except in Gross Syntactic State, in which BERT-based models clearly outperform RoBERTa-based models, and the multilingual model outperforms the monolingual one in both families. Intuitively, we believe that the NSP training objective of BERT-based models helps them to better understand the relation between two sentences, and this knowledge can also be applied to the relation between two clauses (which is the basis of the Gross Syntactic State circuit). Comparing the BERT and RoBERTa model families, it is interesting to notice that while RoBERTa outperforms XLM-R in all circuits except Gross Syntactic State, BERT only outperforms mBERT in 3 of them.

Interestingly, all models seem to struggle with Agreement in English. This observation is aligned with Mueller et al. (2020)'s hypothesis that language models learn better hierarchical syntactic generalizations in morphologically complex languages (such as, e.g., Spanish), which frequently provide overt cues to syntactic structure, than in morphologically simpler languages (such as, e.g., English). Indeed, the fact that XLM-R offers the lowest performance may be related to the fact that the model has been more exposed to more complex languages than the others. For Long Distance Dependencies, BERT-based models show a low performance compared to RoBERTa-based models. This might be due to the different training procedures adopted in both model families (i.e., that RoBERTa does not include the Next Sentence Prediction task (as BERT does) and introduces dynamic masking).

On the other hand, in specific circuits for Spanish (cf. Figure 2) XLM-R outperforms the other two models in 5 out of 7 circuits. As observed for English, the BERT-based models struggle with the Long Distance Dependencies tests, and mBERT offers an outstanding performance in Gross Syntactic State. The monolingual model, BETO, is outperformed by mBERT in 4 out of 7 tests, and by XLM-R in all 6 out of 7 tests. As mentioned before, these differences may be related to the fact that, unlike BERT, BETO is not trained with the NSP objective; but also to the difference in training data size: 16GB for BETO vs. more than 2TB (of which 53GB of Spanish data) for XLM-R.

All models offer a low performance in the new Linearization test for Spanish. A more in-depth investigation is necessary to explain this. The test has been designed with literary Peninsular Spanish in mind, and it is possible that the training data may not contain enough samples that show the targeted word order varieties, or may contain data from American Spanish sources, which may show differences in canonical word order with respect to Peninsular Spanish.

5.2 Cross-language multilingual models performance

As shown in Table 1, multilingual models do not syntactically generalize equally well in both languages. While mBERT offers a better generalization in English, outperforming XLM-R by almost 6 points, XLM-R generalizes better in Spanish, outperforming mBERT by 6 points. This observation corroborates our intuition that XLM-R sacrifices performance in high-resource languages (e.g., English, with 300GB of training data) to be able to offer a more balanced performance across languages (e.g., Spanish, with 53GB of training data).

Comparing Figures 1 and 2, we observe improvements in the Spanish tests for XLM-R in 4 out of 6 circuits, particularly noticeable in Agreement and Center Embedding, while it loses around 10 points in Long Distance Dependencies. On the other hand, mBERT also shows a big improvement in the Spanish tests in Agreement, while it loses performance in Garden Path Effects, Licensing and Long Distance Dependencies.

5.3 Model stability with respect to modifiers

Since modifiers increase the linear distance between the elements in a dependency structure, thus making the task more demanding, stability in this respect indicates that models have robustly learnt the appropriate syntactic generalization and do not depend that much on adjacency. Figures 3 and 4 show the models' average scores in those test suites that have two versions: without modifiers (dark bars) and with modifiers (light bars). As was intuitively expected, all the models offer a higher performance in the tests without modifiers. While for English the multilingual models are the less affected, for Spanish BETO seems to be more robust than the multilingual models, even though it offers a lower performance.

6 Conclusions

In this paper, we assessed the syntactic generalization potential of selected transformer-based language models on English and Spanish. We have shown that multilingual models do not generalize equally well across languages: mBERT generalizes better for phenomena in English, while XLM-R does it better for phenomena in Spanish. We have also shown that the answer to the question whether monolingual or multilingual models generalize better is equally language-specific: the monolingual RoBERTa generalizes better on English, while the multilingual XLM-R generalizes better on Spanish. While it is possible that the multilingual abstractions captured by XLM-R become useful for morphologically rich languages such as Spanish, this difference may also be related to the difference in the amount of training data used to train BETO and XLM-R, and therefore it is possible that a monolingual model trained with a comparable amount of data could outperform the multilingual models.

The performance of all models is affected by the presence of modifiers, which shows that the complexity of the syntactic structure is still a challenge. In general, each syntactic phenomenon deserves attention. For instance, Agreement in English is hard to learn, given the scarcity of cues (especially if compared to a morphologically rich language), and so is Linearization in Spanish.

As far as the nature of the training procedures of the models is concerned, the lack of Next Sentence Prediction (NSP) objective in the RoBERTa model family seems to harm BETO, but not XLM-R; this suggests that the performance of BETO may be improved with (much) more training data. It also seems to harm in the case of the Gross Syntactic State circuit, suggesting that RoBERTa-based models may also benefit from complementary training objectives in their pretraining procedure.

Overall, our experiments have also shown the importance of testing models on a wider range of languages, in particular, morphologically rich ones. As part of our future work, we plan to expand further SyntaxGymES and develop SyntaxGyms for a number of other selected languages. Also, careful examination of a wider range of material is necessary to ensure that important phenomena are not left out, so as to assess the actual coverage of the test suites.

Acknowledgments

This work has been partially funded by the European Commission via its H2020 Research Program under the contract numbers 779962, 786731, 825079, and 870930.

References

José Canete, Gabriel Chaperon, Rodrigo Fuentes, and Jorge Pérez. 2020. Spanish pre-trained BERT model and evaluation data. *PML4DC at ICLR*, 2020.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Edmiston. 2020. A systematic analysis of morphological content in BERT models for multiple languages. *arXiv preprint arXiv:2004.03032v1*.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2020. When Bert forgets how to POS: Amnesic probing of linguistic properties and MLM predictions. *arXiv preprint arXiv:2006.00995*.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. 2018. RNNs as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. Syntaxgym: An online platform for targeted evaluation of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–76.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020a. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020b. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Mandar Joshi, Kenton Lee, Yi Luan, and Kristina Toutanova. 2020. Contextualized representations using textual encyclopedic knowledge. *arXiv preprint arXiv:2004.12006*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, 41(5):1202–1241.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Aaron Mueller, Garrett Nicolai, Panayiota Petrou-Zeniou, Natalia Talmina, and Tal Linzen. 2020. Cross-linguistic syntactic evaluation of word prediction models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5523–5539, Online. Association for Computational Linguistics.

Samuel Rönnqvist, Jenna Kanerva, Tapio Salakoski, and Filip Ginter. 2019. Is multilingual BERT fluent in language generation? In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 29–36, Turku, Finland. Linköping University Electronic Press.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36, Minneapolis, Minnesota. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Ethan Wilcox, Roger Levy, and Richard Futrell. 2019a. Hierarchical representation in neural language models: Suppression and recovery of expectations. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 181–190, Florence, Italy. Association for Computational Linguistics.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221, Brussels, Belgium. Association for Computational Linguistics.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019b. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3302–3312, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

A Spanish SyntaxGym: Description of Test Suites

This appendix lists and describes all the test suites compiled for Spanish SyntaxGym. Each test consists of a list of ITEMS that vary in a controlled way according to a set of CONDITIONS determined by the experimental design. A series of PREDICTIONS compare surprisal values at specific regions of the items across conditions. Some tests have versions with MODIFIERS that increase the linear distance between two co-varying items, making the task more demanding.

The test suites are arranged in terms of *circuits* of related syntactic phenomena. Each of the following sections corresponds to one of these circuits.

Notation. An asterisk * signals an ungrammatical sentence, a question mark ? indicates a marginal sentence (grammatical but very uncommon or requiring a difficult interpretation), an exclamation point ! denotes high processing difficulty.

A.1 Agreement

Agreement is a morpho-syntactic phenomenon that occurs when the features of an item constrain another item to adopt a specific form.

- **Basic Subject-Verb Agreement**. New suite. Spanish finite verbs in any tense/mood have six inflected forms according to person and number features. The verb's features the subject's, otherwise the result is ungrammatical.

(16) Tú cocinas
you.2SG cook.2SG

(17) * Tú cocinais/cocino/cocinan
you.2SG cook.2PL/1SG/3PL

Predictions: The surprisal at the verb region is expected to be lower when it matches the subject than in any other condition. It is also expected to be lower when at least one of the features (person or number) agrees than when both disagree.

- **Subject-Verb Agreement with Subject Relative Clause**. Adapted from English. This test focuses on number agreement. The subject relative clause includes a *distractor* NP differing in number with the subject.

(18) El fontanero que ayudó a los
the.SG plumber that helped.3SG to thePL
albañiles trabaja/*trabajan los sábados.
bricklayers work.3SG/3PL the saturdays.
'The plumber who helped the bricklayers works/*work on saturdays.'

(19) Los fontaneros que ayudaron al
the.PL plumbers that helped.3SG to.thePL
albañil *trabaja/trabajan los sábados.
bricklayer work.3PL/3SG the saturdays.
'The plumbers who helped the bricklayer *works/work on saturdays.'

Predictions: A successful model should place higher probability to the verb agreeing with the subject (instead of the distractor) both in singular and in plural.

- **Subject-Verb Agreement with Object Relative Clause**. Adapted from English. Equal to the previous one, but with an object relative clause.

Nominal agreement was the basis for the following 6 new test suites. All of them share the same predictions: the surprisals should be lower when both gender and number features in the second word of the agreement relation match those in the first word. They should also be lower when only one of the features agrees than when both disagree.

- **Determiner-Noun Agreement**. New suite. The four possible forms of the definite article are paired with different nouns.

(20) El/*La/*Los/*Las gato
the.M.SG/*F.SG/*M.PL/*F.PL cat

- **Adjective-Noun Agreement**. New suite. The test pairs a noun with the four possible forms of an adjective that modifies it (we used constructions without determiner to avoid providing the models with extra information).

(21) La tienda vende discos
the store sells discs
usados/*usado/*usadas/*usada
used.M.PL/M.SG/F.PL/F.SG
'The store sells second-hand discs.'

- **Attribute Agreement**. New suite. Here, a noun is paired with and adjective through a copulative construction. This suite has 2 versions with object or subject relative clauses as modifiers.

(22) El piso está vacío/*vacía/*vacíos/*vacías
the flat is empty.M.SG/*F.SG/*M.PL/*F.PL

- **Predicative Agreement**. New suite. The subject or the object is paired with an adjective functioning as a predicative complement.

(23) Los niños llegaron
the children arrived
cansados/*cansado/*cansadas/*cansada
tired.M.PL/*M.SG/*F.PL/*F.SG
'The children arrived tired.'

A.2 Center Embedding

A center embedded clause is a subordinate clause that sits in the middle of its superordinate clause, creating nested dependencies that may be challenging for the models.

- **Center Embedding**. Adapted from English. A relative clause is center embedded after the subject of the main clause. Verb transitivity and subject-verb plausibility are used to test if the models are capable of retaining the relevant information and predicting the verbs in the correct order.

(24) La tormenta que el capitán [capeó amainó]/?[amainó capeó].
'The storm the captain [weathered abated]/?[abated weathered].'

Prediction: The surprisal of the combination of verbs should be smaller when their relative order creates a plausible sentence than when it creates an implausible one.

- **Center Embedding with modifier**. In the version with modifier, a prepositional phrase is inserted after the subject of the subordinate clause.

A.3 Gross Syntactic State

Expectation for a large syntactic structure at some point within the sentence.

- **Subordination**. Adapted from English. A sentence starting with a subordinate clause creates the expectation for the onset of a matrix clause for as long as the subordinate one lasts.

(25) ?(Mientras) ella miraba los resultados, el doctor entró en la habitación.
'While she looked at the results, the doctor entered the room.'

(26) (*Mientras) ella miraba los resultados.
'(*While) she looked at the results.'

Predictions: The surprisal for the lack of a second clause should be higher when there is a subordinating conjunction or adverb than where there is not. But having two clauses joined by a conjunction/adverb should be less surprising than their juxtaposition.

- **Subordination with Object Relative Clause** and **Subordination with Subject Relative Clause**. Adapted from English. Versions of the previous suite but with a modifier.

A.4 Long-distance Dependencies

LDDs occur when two syntactically related groups do not appear adjacent to one another but at a longer distance from one another.

- **Basic Filler-Gap Dependencies**. New suite, a simplified version of the existing FGD tests for English. FGDs occur when a phrase (the filler) is realized somewhere in the sentence but is semantically interpreted at some other point (the gap).

(27) Yo sé [lo que]/*que tu amigo tiró _ al suelo.
'I know what/*that your friend threw _.'

(28) Yo sé *[lo que]/que tu amigo tiró una colilla al suelo.
'I know *what/that your friend threw a cigarette butt.'

Predictions: The overt object should be more surprising when there is a filler when there is not. We also expect lower surprisal when the sentence has a filler later followed by gap than when it has a conjunction instead but the gap remains.

- **Filler-Gap Dependencies with Three Sentencial Embeddings**. Adapted from English. It is a version of the previous test that includes a modifier (three sentential embeddings) between filler and gap. This makes the task more challenging. The predictions, though, remain the same.

- **Pseudo-Cleft Structures** Adapted from English. A pseudo-cleft or wh-cleft is formed by a wh-element extracting content from a relative clause joined by a copula to a constituent that provides the content requested by the wh-element. The extracted constituent can be a NP or a VP. In the VP case, the verb in the relative clause must be an inflected form of 'hacer' ('to do').

(29) Lo que tú difundiste/?hiciste fue un rumor.
'What you spread/*did was a rumor.'

(30) Lo que tú *difundiste/hiciste fue confirmar un rumor.
'What you *spread/did was confirm a rumor.'

Predictions: The surprisal should be lower for the extracted VP when the verb in the relative clause is a light verb (*hacer* – 'to do') than when it is not, but it should be higher for the extracted NP when the verb is light than when it is semantically heavier and matches the NP. In addition, the difference in the first case should be more important than in the second one. This happens because the light verb admits a wider range of objects, whereas in the first case, one of the options is syntactically incorrect.

A.5 Garden Path Effects

Garden-path effects emerge when an incorrect but locally likely parse needs to be abandoned in favor of the correct one. In the NP/Z garden path, an NP is initially interpreted as the object in a subordinate clause, but when the main verb appears, this NP should be reinterpreted as its subject. The effect can be prevented by adding a comma, but also by placing an overt object in the subordinate clause, or by substituting its verb with a purely intransitive one. These are the basis for the next two suites.

- **NP/Z Garden Path Effect (Overt Object)**.
- **NP/Z Garden Path Effect (Intransitive Verb)**. Both adapted from English.

(31) !Mientras ella leía sus manuscritos se volaron por la ventana.
!'While she read her manuscripts went out the window.'

(32) Mientras ella [dormía]/[leía un libro]/[leía,] sus manuscritos se volaron por la ventana.
'While she [slept]/[read a book]/[read,] her manuscripts went out the window.'

Predictions: The main verb should be more surprising in the garden path condition than when the effect has been prevented either by the comma or by interfering with the verb. Moreover, the difference in surprisal should be bigger when the comma is essential to solve the garden path effect than when it is not.

A.6 Licensing

In natural language, some words or constructions need the presence of a licensor to allow their occurrence in a sentence. This happens with NPIs (Negative polarity items) and subjunctive mood, for instance.

- **Negative Polarity Items and Polarity Agreement**. New suite. In Spanish, NPIs that follow the verb (such as *nunca* 'never', *nadie* 'nobody', and *nada* 'nothing') need to be licensed by negation. This 'double negative' does not result in an affirmative, it is a sort of polarity agreement.

(33) Yo no bebo nunca/?siempre.
I NEG drink never/always
'I never drink./I don't drink always.'

(34) Yo bebo *nunca/siempre.
'I *ever/always drink.'

Predictions: We expect the surprisals in both agreeing conditions (negative-NPI, positive-PPI) to be lower than in any of the non-agreeing conditions (negative-PPI, positive-NPI).

- **Negative Polarity Items**. New suite. NPIs also need to be in the scope of the negation to be licensed by it. This suite compares between a negative particle that "commands" the NPI and one that doesn't.

(35) Tú, como no mirabas por la ventana,
You, as NEG looked by the window,
*(no) has visto a nadie.
NEG have seen at nobody
'As you weren't looking through the window, you have *(not) seen anybody.'

(36) Tú, como mirabas por la ventana, *(no)
You, as looked by the window, NEG
has visto a nadie.
have seen at nobody
'As you were looking through the window, you have *(not) seen anybody.'

Predictions: The NPI should be more surprising when there isn't a negative particle that commands it, independently of the presence of another one that does not command it.

- **Subjunctive Mood and Verbs that Express Feeling**. New suite. Feeling verbs that introduce a subordinate clause serve as licensors for subjunctive mood, whereas other type of verbs do not.

(37) Espero que mañana llueva/*lloverá.
(I)hope that tomorrow rain.SUB/will.rain.IND
'I hope it rains/*[will rain] tomorrow.'

(38) Sé que mañana
(I)know that tomorrow
*llueva/lloverá.
rain.SUB/will.rain.IND
'I know it [will rain]/rains tomorrow.'

Predictions: Subjunctive mood should be less surprising than indicative mood when the verb in the main clause expresses feelings. But when it doesn't, subjunctive should be more surprising than indicative mood. Moreover, subjunctive mood should also be more surprising with a feeling verb than with a non-feeling verb.

- **Subjunctive Mood, Negation and Belief Verbs**. New suite. Belief verbs can also license subjunctive mood, but only when combined with negation.

(39) No creo que mañana
NEG believe that tomorrow
llueva/*lloverá.
rain.SUB/will.rain.IND
'I don't think it rains/[will rain] tomorrow.'

(40) Creo que mañana no
(I)believe that tomorrow NEG
*llueva/lloverá.
rain.SUB/will.rain.IND
'I think it rains/[won't rain] tomorrow.'

Predictions: The subordinate verb should be less surprising in subjunctive than in indicative mood when the main clause is negated. However, the contrary should hold when the subordinate clause is negated but the main one is not. In addition, subjunctive mood should be less surprising when the negation is in the main clause than when it is in the subordinate clause.

A.7 Linearization

Constituent order is commonly used in linguistics as a way to classify languages. But, in addition to the canonical order in which elements appear, languages also differ in their flexibility to alter that order.

- **Subject – Auxiliary Verb – Main Verb Linearization**. New suite. Subject-verb order admits inversion in Spanish but main and auxiliary verb do not and they must be adjacent.

(41) Juan ha comido. / Ha comido Juan
'John has eaten. / Has eaten John.'

(42) *Juan comido ha. / *Ha Juan comido.
'John eaten has. / Has John eaten.'

Predictions: The postposed subject should be less surprising than any of the alterations involving auxiliary and main verb. The canonical SV order, however, should be less surprising than postposing the subject, and the difference in this case should be less important than the differences in the first two cases.

- **Subject – Verb – Object Linearization**. New test. In Spanish, word order flexibility holds for affirmative sentences but not for interrogative ones, where subject-verb inversion is compulsory.

(43) Ana compró un libro/Compró un libro Ana.
'Ann bought a book. / Bought a book Ann.'

(44) ¿Qué compró Ana? / ¿Qué Ana compró?
'What did Ana buy? / 'What Ana did buy?'

Predictions: A postposed subject in an affirmative sentence should be less surprising than lack of SV inversion in an interrogative one. The canonical SV order in the affirmative sentence, however, should be less surprising than postposing the subject, and the difference in this case should be less important than the difference in the first one.

- **Noun-Adjective and Noun-PP Linearization**. New suite. Spanish adjectives usually come after the noun, but this order can be inverted. Other noun modifiers like prepositional phrases cannot.

(45) Construyó una [mesa robusta]/[robusta mesa].
'He built a [sturdy table]/[table sturdy].'

(46) Construyó una [mesa de madera]/*[de madera mesa].
'He built a [wooden table]/*[table wooden].'

Predictions: A PP preceding the noun should be more surprising than one following it. An adjective preceding the noun should also be more surprising than one following it, but the difference in this case should be less important than in the first one.

Are Larger Pretrained Language Models Uniformly Better? Comparing Performance at the Instance Level

Ruiqi Zhong **Dhruba Ghosh** **Dan Klein** **Jacob Steinhardt**
Computer Science Division, University of California, Berkeley
{ruiqi-zhong, djghosh13, klein, jsteinhardt}@berkeley.edu

Abstract

Larger language models have higher accuracy on average, but are they better on every single instance (datapoint)? Some work suggests larger models have higher out-of-distribution robustness, while other work suggests they have lower accuracy on rare subgroups. To understand these differences, we investigate these models at the level of individual instances. However, one major challenge is that individual predictions are highly sensitive to noise in the randomness in training. We develop statistically rigorous methods to address this, and after accounting for pretraining and finetuning noise, we find that our BERT-LARGE is worse than BERT-MINI on at least $1-4\%$ of instances across MNLI, SST-2, and QQP, compared to the overall accuracy improvement of $2-10\%$. We also find that finetuning noise increases with model size, and that instance-level accuracy has momentum: improvement from BERT-MINI to BERT-MEDIUM correlates with improvement from BERT-MEDIUM to BERT-LARGE . Our findings suggest that instance-level predictions provide a rich source of information; we therefore recommend that researchers supplement model weights with model predictions.

1 Introduction

Historically, large deep learning models (Peters et al., 2018; Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2019) have improved the state of the art on a wide range of tasks and leaderboards (Schwartz et al., 2014; Rajpurkar et al., 2016; Wang et al., 2018), and empirical scaling laws predict that larger models will continue to increase performance (Kaplan et al., 2020). However, little is understood about such improvement at the instance (datapoint) level. Are larger models uniformly better? In other words, are larger pretrained models better at every instance, or are they better at some instances, but worse at others?

Prior works hint at differing answers. Hendrycks et al. (2020) and Desai and Durrett (2020) find that larger pretrained models consistently improve out-of-distribution performance, which implies that they might be uniformly better at a finer level. Henighan et al. (2020) claim that larger pretrained image models have lower downstream classification loss for the majority of instances, and they predict this trend to be true for other data modalities (e.g. text). On the other hand, Sagawa et al. (2020) find that larger non-pretrained models perform worse on rare subgroups; if this result generalizes to pretrained language models, larger models will not be uniformly better. Despite all the indirect evidence, it is still inconclusive how many instances larger pretrained models perform worse on.

A naïve solution is to finetune a larger model, compare it to a smaller one, and find instances where the larger model is worse. However, this approach is flawed, since model predictions are **noisy at the instance level**. On MNLI in-domain development set, even the same architecture with different finetuning seeds leads to different predictions on $\sim$8% of the instances. This is due to under-specification (D'Amour et al., 2020), where there are multiple different solutions that can minimize the training loss. Since the accuracy improvement from our BERT-BASE[1] to BERT-LARGE is 2%, most signals across different model sizes will be dominated by noise due to random seeds.

To account for the noise in pretraining and finetuning, we define *instance accuracy* as "how often a model correctly predicts an instance" (Figure 1 left) in expectation across pretraining and finetuning seeds. We estimate this quantity by pretraining 10 models with different seeds, finetuning 5 times for each pretrained models (Figure 1 middle), and

---

[1] This is not the original release by Devlin et al. (2019); we pretrained models ourselves.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3813–3827
August 1–6, 2021.

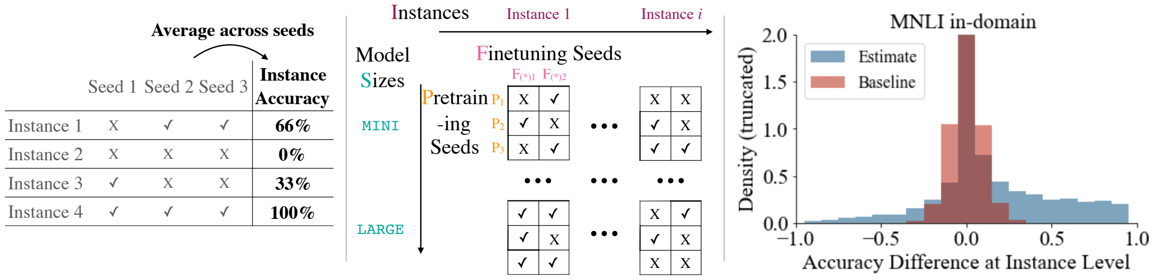

	Seed 1	Seed 2	Seed 3	Instance Accuracy
Instance 1	X	✓	✓	**66%**
Instance 2	X	X	X	**0%**
Instance 3	✓	X	X	**33%**
Instance 4	✓	✓	✓	**100%**

Figure 1: **Left**: Each column represents the same architecture trained with a different seed. We calculate accuracy for each instance (row) by averaging across seeds (column), while it is usually calculated for each model by averaging across instances. **Middle**: A visual layout of the model predictions we obtain, which is a binary-valued tensor with 4 axes: model size s, instance i, pretraining seeds P and finetuning seeds F. **Right**: for each instance, we calculate the accuracy gain from MINI to LARGE and plot the histogram in blue, along with a random baseline in red. Since the blue distribution has a bigger left tail, smaller models are better at some instances.

averaging across them.

However, this estimate is still inexact, and we might falsely observe smaller models to be better at some instances by chance. Hence, we propose a random baseline to estimate the fraction of *false discoveries* (Section 3, Figure 1 right) and formally upper-bound the false discoveries in Section 4. Our method provides a better upper bound than the classical Benjamini-Hochberg procedure with Fisher's exact test.

Using the 50 models for each size and our improved statistical tool, we find that, on the MNLI in-domain development set, the accuracy "decays" from BERT-LARGE to BERT-MINI on at least ∼4% of the instances, which is significant given that the improvement in overall accuracy is 10%. These decaying instances contain more controversial or wrong labels, but also correct ones (Section 4.2). Therefore, larger pretrained language models are not uniformly better.

We make other interesting discoveries at the instance level. Section 5 finds that instance-level accuracy has momentum: improvement from MINI to MEDIUM correlates with improvement from MEDIUM to LARGE . Additionally, Section 6 attributes variance of model predictions to pretraining and finetuning random seeds, and finds that finetuning seeds cause more variance for larger models. Our findings suggest that instance-level predictions provide a rich source of information; we therefore recommend that researchers supplement model weights with model predictions. In this spirit, we release all the pretrained models, model predictions, and code here: `https://github.com/ruiqi-zhong/acl2021-instance-level`.

2 Data, Models, and Predictions

To investigate model behavior, we considered different sizes of the BERT architecture and finetuned them on Quora Question Pairs (QQP[2]), Multi-Genre Natural Language Inference (MNLI; Williams et al. (2020)), and the Stanford Sentiment Treebank (SST-2; Socher et al. (2013)). To account for pretraining and finetuning noise, we averaged over multiple random initializations and training data order, and thus needed to pretrain our own models rather than downloading off the internet. Following Turc et al. (2019) we trained 5 architectures of increasing size: MINI (L4/H256, 4 Layers with hidden dimension 256), SMALL (L4/H512), MEDIUM (L8/H512), BASE (L12/H768), and LARGE (L24/H1024). For each architecture we pre-trained models with 10 different random seeds and fine-tuned each of them 5 times (50 total) on each task; see Figure 1 middle. Since pretraining is computationally expensive, we reduced the context size during pretraining from 512 to 128 and compensated by increasing training steps from 1M to 2M. Appendix A includes more details about pretraining and finetuning and their computational cost, and Appendix B verifies that our cost-saving changes do not affect accuracy qualitatively.

Notation. We use i to index an instance in the evaluation set, s for model sizes, P for pretraining seeds and F for finetuning seeds. c is a random variable of value 0 or 1 to indicate whether the prediction is correct. Given the pretraining seed P and the finetuning seed F, $c_i^s = 1$ if the model of

---

[2] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

	$\mathrm{Diff}_{\mathrm{FTune}}$	$\mathrm{Diff}_{\mathrm{PTrain}}$	$\mathrm{Std}_{\mathrm{all}}$
MINI	7.2%	10.7%	0.2%
SMALL	7.2%	10.7%	0.3%
MEDIUM	8.0%	10.7%	0.3%
BASE	8.5%	10.6%	0.2%
LARGE	8.6%	10.1%	0.2%

Table 1: Larger model sizes are at the bottom rows. $\mathrm{Diff}_{\mathrm{FTune}}$: how much do the predictions differ, if two models have the same pretraining seed but different finetuning seeds F? $\mathrm{Diff}_{\mathrm{PTrain}}$: the difference if the pretraining seeds P are different. $\mathrm{Std}_{\mathrm{all}}$: the standard deviation of overall accuracy, around 40 times smaller than $\mathrm{Diff}_{\mathrm{FTune}}$.

size s is correct on instance i, 0 otherwise. To keep the notation uncluttered, we sometimes omit these superscripts or subscripts if they can be inferred from context.

Unless otherwise noted, we present results on the MNLI in-domain development set in the main paper.

3 Comparing Instance Accuracy

To find the instances where larger models are worse, a naïve approach is to finetune a larger pretrained model, compare it to a smaller one, and find instances where the larger is incorrect but the smaller is correct. Under this approach, BERT-LARGE is worse than BERT-BASE on 4.5% of the instances and better on 7%, giving an overall accuracy improvement of 2.5%.

However, this result is misleading: even if we compare two BERT-BASE model with different finetuning seeds, their predictions differ on 8% of the instances, while their accuracies differ only by 0.1%; Table 1 reports this baseline randomness across model sizes. Changing the pretraining seed also changes around 2% additional predictions beyond finetuning.

Table 1 also reports the standard deviation of overall accuracy, which is about 40 times smaller. Such stability starkly contrasts with the noisiness at the instance level, which poses a unique challenge.

Instance-Level Metrics To reflect this noisiness, we define the *instance accuracy* Acc_i^s to be how often models of size s predict instance i correctly,

$$\mathrm{Acc}_i^s := \mathbb{E}_{P,F}[c_i^s]. \tag{1}$$

The expectation is taken with respect to the pretraining and finetuning randomness P and F. We estimate Acc_i^s via the empirical average $\hat{\mathrm{Acc}}_i^s$ accross 10 pretraining $\times$ 5 finetuning runs.

We histogram $\hat{\mathrm{Acc}}_i^s$ in Figure 2 (a). On most instances the model always predicts correctly or incorrectly ($\hat{\mathrm{Acc}} = 0$ or 1), but a sizable fraction of accuracies lie between the two extremes.

Recall that our goal is to find instances where larger models are less accurate, which we refer to as *decaying* instances. We therefore study the *instance difference* between two model sizes s_1 and s_2, defined as

$$^{s_1}_{s_2}\Delta\mathrm{Acc}_i := \mathrm{Acc}_i^{s_2} - \mathrm{Acc}_i^{s_1}, \tag{2}$$

which is estimated by the difference between the accuracy estimates $\hat{\mathrm{Acc}}_i^s$, i.e.

$$^{s_1}_{s_2}\Delta\hat{\mathrm{Acc}}_i := \hat{\mathrm{Acc}}_i^{s_2} - \hat{\mathrm{Acc}}_i^{s_1}. \tag{3}$$

We histogram $^{\mathrm{BASE}}_{\mathrm{LARGE}}\Delta\hat{\mathrm{Acc}}_i$ in Figure 2 (b). We observe a unimodal distribution centered near 0, with tails on both sides. Therefore, the estimated differences for some instances are negative.

However, due to estimation noise, we might falsely observe this accuracy decay by chance. Therefore, we introduce a random baseline $^{s_1}_{s_2}\Delta\mathrm{Acc}'$ to control for these false discoveries. Recall that we have 10 smaller pretrained models and 10 larger ones. Our baseline splits these into a group A of 5 smaller + 5 larger, and another group B of the remaining $5 + 5$. Then the empirical accuracies $\hat{\mathrm{Acc}}^A$ and $\hat{\mathrm{Acc}}^B$ are identically distributed, so we take our baseline $^{s_1}_{s_2}\Delta\mathrm{Acc}'$ to be the difference $\hat{\mathrm{Acc}}^A - \hat{\mathrm{Acc}}^B$. We visualize and compare how to calculate $^{s_1}_{s_2}\Delta\hat{\mathrm{Acc}}$ and $^{s_1}_{s_2}\Delta\mathrm{Acc}'$ in Figure 3.

We histogram this baseline $^{\mathrm{BASE}}_{\mathrm{LARGE}}\Delta\mathrm{Acc}'$ in Figure 2 (b), and find that our noisy estimate $^{\mathrm{BASE}}_{\mathrm{LARGE}}\Delta\hat{\mathrm{Acc}}$ has a larger left tail than the baseline. This suggests that decaying instances exist. We similarly compare MINI to LARGE in Figure 2 (c) and find an even larger left tail.

4 Quantifying the Decaying Instances

The left tail of $\Delta\hat{\mathrm{Acc}}$ noisily estimates the fraction of decaying instances, and the left tail of the random baseline $\Delta\mathrm{Acc}'$ counts the false discovery fraction due to the noise. Intuitively, the true fraction of decaying instances can be captured by the difference of these left tails, and we formally quantify this below.

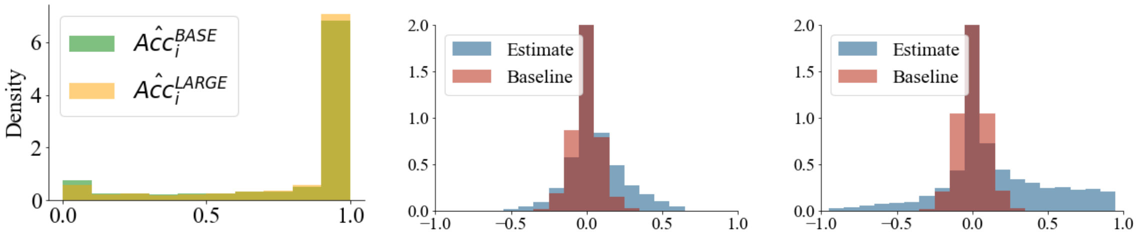

(a) BASE vs. LARGE , Acc (b) BASE vs. LARGE , ΔAcc (c) MINI vs. LARGE , ΔAcc

Figure 2: (a) The distribution of instance accuracy $\hat{\mathrm{Acc}}_i$. (b, c) Histogram of instance difference estimate (x-axis), $\Delta\hat{\mathrm{Acc}}$ (blue) and its baseline $\Delta\mathrm{Acc}'$ (red) compares BASE and LARGE . To better visualize, we truncated the density (y-axis) above 2. Since the blue histogram has a larger left tail than the red one, there are indeed instances where larger models are worse.

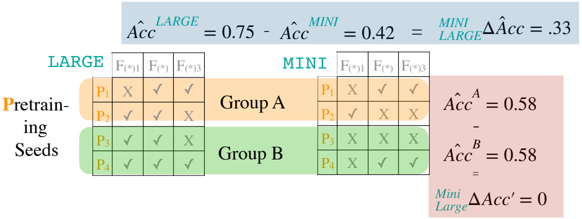

Figure 3: The tables are model predictions with visual notations established in Figure 1 middle. $\Delta\hat{\mathrm{Acc}}$ (blue) is the mean difference between the left and the right table, each corresponding to a model size. The random baseline $\Delta\mathrm{Acc}'$ (red) is the mean difference between group A (orange) cells and group B (green), which are identically and independently distributed.

Suppose instance i is drawn from the empirical evaluation distribution. Then we can define the true decaying fraction Decay as

$$\mathrm{Decay} := \mathbb{P}_i[\Delta\mathrm{Acc}_i < 0]. \tag{4}$$

Since $\Delta\mathrm{Acc}_i$ is not directly observable and $\Delta\hat{\mathrm{Acc}}_i$ is noisy, we add a buffer and only consider instances with $\Delta\hat{\mathrm{Acc}}_i \leq t$, which makes it more likely (but still uncertain) that the true $\Delta\mathrm{Acc}_i < 0$. We denote this "discovery fraction" $\widehat{\mathrm{Decay}}(t)$ as

$$\widehat{\mathrm{Decay}}(t) := \mathbb{P}_i[\Delta\hat{\mathrm{Acc}}_i \leq t]. \tag{5}$$

Similarly, we define a baseline control (false discovery fraction) $\mathrm{Decay}'(t) := \mathbb{P}_i[\Delta\mathrm{Acc}'_i \leq t]$. Hence, $\widehat{\mathrm{Decay}}$ and Decay' are the cumulative distribution function of $\Delta\hat{\mathrm{Acc}}$ and $\Delta\mathrm{Acc}'$ (Figure 4).

We have the following theorem, which we formally state and prove in Appendix D:

Theorem 1 *(Informal) If all the random seeds are independent, then for all thresholds t,*

$$\mathrm{Decay} \geq \mathbb{E}[\widehat{\mathrm{Decay}}(t) - \mathrm{Decay}'(t)] \tag{6}$$

Proof Sketch Suppose we observe $c^{s_1}_{R_{1\ldots 2k}}$ and $c^{s_2}_{R_{2k+1\ldots 4k}}$, where there are $2k$ different random seeds for each model size [3]. Then

$$\Delta\hat{\mathrm{Acc}}_i := \frac{1}{2k}\Big(\sum_{j=1}^{2k} c^{s_1}_{R_j,i} - \sum_{j=2k+1}^{4k} c^{s_2}_{R_j,i}\Big), \tag{7}$$

and hence the discovery rate $\widehat{\mathrm{Decay}}(t)$ is defined as

$$\widehat{\mathrm{Decay}}(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \mathbf{1}[\Delta\hat{\mathrm{Acc}} \leq t]. \tag{8}$$

For the random baseline estimator, we have

$$\begin{aligned}\Delta\mathrm{Acc}'_i := \frac{1}{2k}\Big(&\sum_{j=1}^{k} c^{s_1}_{R_j,i} + \sum_{j=2k+1}^{3k} c^{s_2}_{R_j,i} \\ &- \sum_{j=k+1}^{2k} c^{s_1}_{R_j,i} - \sum_{j=3k+1}^{4k} c^{s_2}_{R_j,i}\Big),\end{aligned} \tag{9}$$

and the false discovery control Decay' is defined as

$$\mathrm{Decay}'(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \mathbf{1}[\Delta\mathrm{Acc}'_i \leq t]. \tag{10}$$

Formally, the theorem states that

$$\mathrm{Decay} \geq \mathbb{E}_{R_1\ldots R_{4k}}[\widehat{\mathrm{Decay}}(t) - \mathrm{Decay}'(t)], \tag{11}$$

which is equivalent to

$$\begin{aligned}\sum_{i=1}^{|\mathcal{T}|}(\mathbf{1}[\Delta\mathrm{Acc}_i < 0] - \mathbb{P}[\Delta\hat{\mathrm{Acc}}_i \leq t] \\ + \mathbb{P}[\Delta\mathrm{Acc}'_i \leq t]) \geq 0\end{aligned} \tag{12}$$

---

[3] We assumed even number of random seeds since we will mix half of the models from each size to compute the random baseline

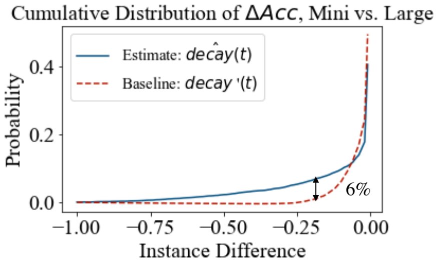

Figure 4: The cumulative distribution function of the histogram in Figure 2 (c); only the negative x-axis is shown because it corresponds to decays. The maximum difference between the two curves (6%) is a lower bound of the true decaying fraction.

Hence, we can declare victory if we can prove that for all i, if $\Delta\text{Acc}_i \geq 0$,

$$\mathbb{P}[\Delta\text{Acc}'_i \leq t] \geq \mathbb{P}[\Delta\hat{\text{A}}\text{cc}_i \leq t].$$

This is easy to see, since $\Delta\text{Acc}'_i$ and $\Delta\hat{\text{A}}\text{cc}_i$ are both binomial distributions with the same n, but the first has a larger rate. [4] □

Roughly speaking, the true decaying fraction is at least the difference between $\widehat{\text{Decay}}(t)$ and $\text{Decay}'(t)$ at every threshold t. Therefore, we take the maximum difference between $\widehat{\text{Decay}}(t)$ and $\text{Decay}'(t)$ to lower-bound the fraction of decaying instances.[5] For example, Figure 4 estimates the true decaying fraction between MINI and LARGE to be at least 6%.

We compute this lower bound for other pairs of model sizes in Table 2, and the full results across other tasks and model size pairs are in Appendix C. In all of these settings we find a non-zero fraction of decaying instances, and larger model size differences usually lead to more decaying instances.

Unfortunately, applying Theorem 1 as above is not fully rigorous, since some finetuning runs share the same pretraining seeds and hence are dependent.[6] To obtain a statistically rigorous lower bound, we slightly modify our target of interest. Instead of examining individual finetuning runs, we ensemble our model across 5 different finetuning runs for each pretraining seed; these predictions are essentially the same as individual finetuning runs, except that the finetuning randomness is averaged out. Hence we obtain 10 independent sets of model predictions with different random seeds, which allows us to apply Theorem 1.

$s_1 \setminus s_2$	MINI	SMALL	BASE	LARGE
MINI	N/A	9%	18%	21%
SMALL	3%	N/A	14%	18%
BASE	6%	5%	N/A	10%
LARGE	6%	5%	2%	N/A

Table 2: We lower-bound the fraction of instances that improve when model size changes from s_1 (row) to s_2 (column). For example, when model size decreases from LARGE to MINI , 6% of instances improve (i.e. decays).

Threshold	$\widehat{\text{Decay}}$	Decay'	Diff
$t = -0.4$	4.22%	3.49e−3	3.87%
…	…	…	…
$t = -0.9$	0.91%	1.44e−7	0.91%
$t = -1.0$	0.48%	2.06e−8	0.48%

Table 3: Comparing MINI vs. LARGE by calculating the discovery fraction $\widehat{\text{Decay}}$, the false discovery control Decay', and their difference (Diff) under different thresholds t. LARGE is worse on at least ~4% (maximum Diff) of instances.

We compare MINI to LARGE using these ensembles and report the discovery $\widehat{\text{Decay}}$ and the baseline Decay' in Table 3. Taking the maximum difference across thresholds, we estimate at least ~4% of decaying instances. This estimate is lower than the previous 6% estimate, which used the full set of 50 models' predictions assuming they were independent. However, this is still a meaningful amount, given that the overall accuracy improvement from MINI to LARGE is 10%.

4.1 Fisher's Test + Benjamini-Hochberg

Here is a more classical approach to lower-bound the decaying fraction. For each instance, we compute a significance level α under the null hypothesis that the larger model is better, using Fisher's exact test. We sort the significance levels ascendingly, and call the p^{th} percentile α_p. Then we pick a false discovery rate q (say, 25%), find the largest p s.t. $\alpha_p < pq$, and estimate the decaying fraction to be at least $p(1-q)$. This calculation is known as the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995).

To compare our method with this classical approach, we estimate the lower bound of the decaying fraction for different pairs of model sizes with different numbers of pretrained models available.

---

[4] More details are in Appendix D.

[5] Adaptively picking the best threshold t depending on the data may incur a slight upward bias. Appendix E estimates that the relative bias is at most 10% using a bootstrap method.

[6] Although we anticipate such dependencies do not cause a substantial difference, as discussed in Appendix D.1.

s1	s2		2	6	10
MINI	LARGE	ours	1.9%	3.1%	4.0%
MINI	LARGE	BH	0.0%	0.9%	1.9%
BASE	LARGE	ours	0.4%	0.9%	1.2%
BASE	LARGE	BH	0.0%	0.0%	0.0%

Table 4: We compare our method to the Fisher's exact test + Benjamin-Hochberg (BH) procedure described in Section 4. For all different model size pairs and number of pretrained models available, ours always provides a higher (better) lower bound of the decaying fraction.

To make sure our choice of the false discovery rate q does not bias against the classical approach, we adaptively choose q to maximize its performance. Appendix F includes the full results and Table 4 is a representative subset.

We find that our approach is more powerful, particularly when the true decaying fraction is likely to be small and only a few models are available, which is usually the regime of interest. For example, across all pairs of model sizes, our approach only needs 2 random seeds (i.e. pretrained models) to provide a non-zero lower bound on the decaying fraction, while the classical approach sometimes fails to do this even with 10 seeds. Intuitively, when fewer seeds are available, the smallest possible significance level for each instance is larger than the decaying fraction, hence hurting the classical approach.

4.2 Understanding the Decaying Instances

We next manually examine the decaying instances to see whether we can find any interpretable patterns. One hypothesis is that all the decaying fractions are in fact mislabeled, and hence larger models are not in fact worse on any instances.

To investigate this hypothesis, we examined the group of instances where $^{\text{MINI}}_{\text{LARGE}}\Delta\hat{\text{Acc}}_i \leq -0.9$. MINI is almost always correct on these instances, while LARGE is almost always wrong, and the false discovery fraction is tiny. For each instance, we manually categorize it as either: 1) Correct, if the label is correct, 2) Fine, if the label might be controversial but we could see a reason why this label is reasonable, 3) Wrong, if the label is wrong, or 4) Unsure, if we are unsure about how to label this instance. Each time we annotate, with 50% probability we randomly sample either a decaying instance or an instance from the remaining dataset as a control. We are blind to which group it comes from.

	Correct	Fine	Wrong	Unsure
MNLI^D	66%	17%	9%	5%
MNLI^C	86%	5%	5%	1%
SST-2^D	55%	8%	10%	25%
SST-2^C	88%	4%	0%	6%
QQP^D	60%	26%	10%	2%
QQP^C	87%	10%	1%	0%

Table 5: MINI vs. LARGE . We examine whether there are mislabels for the **D**ecaying fractions (superscript ^{D}) and the rest of the dataset (**C**ontrol group ^{C}). The decaying fraction contains more mislabels, but includes correct labels as well.

For each task of MNLI, QQP, and SST-2, the first author annotated 100 instances (decay + control group) (Table 5). We present all the annotated decaying instances in Appendix J.

Conclusion We find that the decaying fraction has more wrong or controversial labels, compared to the remaining instances. However, even after we adjust for the fraction of incorrect labels, the Decay fraction still exceeds the false discovery control. This implies that MINI models are better than LARGE models on some correctly labeled instances. The second author followed the same procedure and reproduced the same qualitative results.

However, we cannot find an interpretable pattern for these correctly labeled decaying instances by simply eyeballing. We discuss future directions to discover interpretable categories in Section 7.

5 Correlation of Instance Difference

We next investigate whether there is a momentum of instance accuracy increase: for example, if the instance accuracy improves from MINI(s_1) to MEDIUM(s_2), is it more likely to improve from MEDIUM(s_2) to LARGE(s_3)?

The naïve approach is to calculate the Pearson correlation coefficient between $^{\text{MINI}}_{\text{MEDIUM}}\Delta\hat{\text{Acc}}$ and $^{\text{MEDIUM}}_{\text{LARGE}}\Delta\hat{\text{Acc}}$, and we find the correlation to be zero. However, this is partly an artifact of accuracies being bounded in $[0, 1]$. If MEDIUM drastically improves over MINI from 0 to 1, there is no room for LARGE to improve over MEDIUM. To remove this inherent negative correlation, we calculate the correlation conditioned on the accuracy of the middle-sized model, $\hat{\text{Acc}}^{\text{MEDIUM}}$.

Therefore, we bucket instances by their estimated MEDIUM accuracy into intervals of size 0.1,

$(s_1, s_2, s_3)\downarrow$ Buckets$\rightarrow$	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
SMALL, MEDIUM, BASE	0.07	0.22	0.29	0.40	0.35	0.33	0.38	0.27	0.24	0.13
MINI, MEDIUM, LARGE	0.03	0.15	0.18	0.33	0.17	0.16	0.22	0.20	0.19	0.09

Table 6: Each row corresponds to a triplet of model sizes. Each column t represents a bucket that contains instances with $\hat{\mathrm{Acc}}^{s_2} \in [t - 0.1, t]$. Within each bucket, we calculate the Pearson correlation coefficient between the estimated accuracy improvements: ${}^{s_1}_{s_2}\Delta\hat{\mathrm{Acc}}$ and ${}^{s_2}_{s_3}\Delta\hat{\mathrm{Acc}}$. These correlations are positive and become higher when model size differences are small.

and we find the correlation to be positive within each bucket (Table 6, row 2). This fixes the problem with the naïve approach by getting rid of the negative correlation, which could have misled us to believe that improvements by larger models are uncorrelated.

We additionally find that the correlations between improvements become stronger when model size differences are smaller. Table 6 row 1 reports results for another model size triplet with smaller size difference, i.e. (s_1, s_2, s_3) = (SMALL, MEDIUM, BASE), and the correlation is larger for all buckets. Results for more tasks and size triplets are in Appendix G and the same conclusions hold qualitatively.

6 Variance at the Instance Level

Section 3 found that the overall accuracy has relatively low variance, but model predictions are noisy. This section formally analyzes variance at the instance level. For each instance, we decompose its loss into three components: Bias^2, variance due to pretraining randomness, and variance due to finetuning randomness. Formally, we consider the $0/1$ loss:

$$\mathcal{L}_i := 1 - c_i = (1 - c_i)^2, \tag{13}$$

where c_i is a random variable 0/1 indicating whether the prediction is correct or incorrect, with respect to randomness in pretraining and finetuning. Therefore, by bias-variance decomposition and total variance decomposition, we have

$$\mathcal{L}_i = \mathrm{Bias}^2{}_i + \mathrm{PretVar}_i + \mathrm{FineVar}_i, \tag{14}$$

where, by using P and F as pretraining and finetuning random seeds:

$$\begin{aligned} \mathrm{Bias}^2{}_i &:= (1 - \mathbb{E}_{P,F}[c_i])^2, \\ \mathrm{PretVar}_i &:= \mathrm{Var}_P[\mathbb{E}_F[c_i]], \\ \mathrm{FineVar}_i &:= \mathbb{E}_P[\mathrm{Var}_F[c_i]], \end{aligned} \tag{15}$$

capturing “how wrong is the average prediction”, variance due to pretraining, and variance due to finetuning seeds, respectively.

	Bias^2	$\mathrm{PretVar}$	$\mathrm{FineVar}$
MINI	0.203	0.017	0.036
SMALL	0.179	0.017	0.036
MEDIUM	0.157	0.014	0.040
BASE	0.134	0.010	0.043
LARGE	0.111	0.007	0.043

Table 7: The bias, pretraining variance, and finetuning variance for each model size, averaged across all test instances. Finetuning variance is much larger than pretraining variance; larger models have larger finetuning variance.

We can directly estimate $\mathrm{FineVar}$ by first calculating the sample variance across finetuning runs for each pretraining seed, and then averaging the variances across the pretraining seeds. Estimating $\mathrm{PretVar}$ is more complicated. A naïve approach is to calculate the empirical variance, across pretraining seeds, of the average accuracy across finetuning seeds. However, the estimated average accuracy for each pretraining seed is noisy itself, which causes an upward bias on the $\mathrm{PretVar}$ estimate. We correct this bias by estimating the variance of the estimated average accuracy and subtracting it from the naïve estimate; see Appendix H for details, as well as a generalization to more than two sources of randomness. Finally, we estimate Bias^2 by subtracting the two variance estimates from the estimated loss.

For each of these three quantities, Bias^2, $\mathrm{PretVar}$ and $\mathrm{FineVar}$, we estimate it for each instance, average it across all instances in the evaluation set, and report it in Table 7. The variances at the instance level are much larger than the variance of overall accuracy, by a factor of 1000.

We may conclude from Table 7 that larger models have larger finetuning variance and smaller pretraining variance. However, lower bias also inherently implies lower variance. To see this, suppose a model has perfect accuracy and hence zero bias; then it always predicts the same label (the correct one) and hence has zero variance. This might favor larger models and “underestimate” their variance,

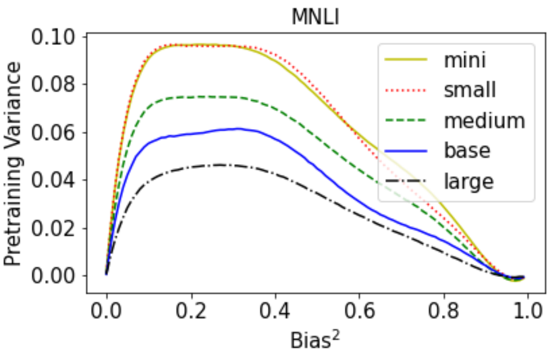

Figure 5: The pretraining variance conditioned on Bias^2 (the level of correctness). Each curve represents a model size. Larger models have lower pretraining variance across all levels of bias.

since they have lower bias. Therefore, we calculate and compare the variances conditioned on the bias, i.e. $\mathrm{PretVar}(b^2) := \mathbb{E}_i[\mathrm{PretVar}_i|\mathrm{Bias}^2{}_i = b^2]$.

We estimate $\mathrm{PretVar}^s(b^2)$ using Gaussian process regression and plot it against b^2 in Figure 5. We find that larger models still have lower pretraining variance across all levels of bias on the specific task of MNLI under the 0/1 loss. To further check whether our conclusions are general, we tested them on other tasks and under the squared loss $\mathcal{L}_i := (1 - p_i)^2$, where p_i is the probability assigned to the correct class. Below are the conclusions that generally hold across different tasks and loss functions.

Conclusion We find that 1) larger models have larger finetuning variance, 2) LARGE has smaller pretraining variance than BASE ; however, the ordering between other sizes varies across tasks and losses, and 3) finetuning variance is $2-8$ times as large as pretraining variance, and the ratio is bigger for larger models.

7 Discussion and Future Directions

To investigate model behaviors at the instance level, we produced massive amounts of model predictions in Section 2 and treated them as raw data. To extract insights from them, we developed better metrics and statistical tools, including a new method to control the false discoveries, an unbiased estimator for the decomposed variances, and metrics that compute variance and correlation of improvements conditioned on instance accuracy. We find that larger pretrained models are indeed worse on a non-trivial fraction of instances and have higher variance due to finetuning seeds; additionally, instance accuracy improvements from MINI to MEDIUM correlate with improvements from MEDIUM to LARGE .

Overall, we treated model prediction data as the central object and built analysis tools around them to obtain a finer understanding of model performance. We therefore refer to this paradigm as "**instance level understanding as data mining**". We discuss three key factors for this paradigm to thrive: 1) scalability and the cost of obtaining prediction data, 2) other information to collect for each instance, and 3) better statistical tools. We analyze each of these aspects below.

Scalability and Cost of Data Data mining is more powerful with more data. How easy is it to obtain more model predictions? In our paper, the main bottleneck is pretraining. However, once the pretrained models are released, individual researchers can download them and only need to repeat the cheaper finetuning procedure.

Furthermore, model prediction data are under-shared: while many recent research papers share their code or even model weights to help reproduce the results, it is not yet a standard practice to share all the model predictions. Since many researches follow almost the same recipe of pretraining and finetuning (McCoy et al., 2020; Desai and Durrett, 2020; Dodge et al., 2020), much computation can be saved if model predictions are shared. On the other hand, as the state of the art model size is increasing at a staggering speed[7], most researchers will not be able to run inference on a single instance. The trend that models are becoming larger and more similar necessitate more prediction sharing.

Meta-Labels and Other Predictions Data mining is more powerful with more types of information. One way to add information to each instance is to assign "meta-labels". In the HANS (McCoy et al., 2019) dataset, the authors tag each instance with a heuristic [8] that holds for the training distribution but fails on this instance. Naik et al. (2018a) and Ribeiro et al. (2020) associate each instance with a particular stress test type or subgroup, for example, whether the instance requires the model to reason numerically or handle negations. Nie et al.

---

[7] e.g. BERT (Devlin et al., 2019) has 340M parameters, while Switch-Transformer has over 1 trillion parameters (Fedus et al., 2021).

[8] For example, "the label [entailment] is likely if the premise and the hypothesis have significant lexical overlap".

(2020) collects multiple human responses to estimate human disagreement for each instance. This meta-information can potentially help us identify interpretable patterns for the disagreeing instances where one model is better than the other. On the flip side, identifying disagreeing instances between two models can also help us generate hypothesis and decide what subgroup information to annotate.

We can also add performance information on other tasks to each instance. For example, Pruksachatkun et al. (2020) studied the correlation between syntactic probing accuracy (Hewitt and Liang, 2019) and downstream task performance. Turc et al. (2019) and Kaplan et al. (2020) studied the correlation between language modelling loss and the downstream task performance. However, they did not analyze correlations at the instance level. We may investigate whether their results hold on the instance level: if an instance is easier to tag by a probe or easier to predict by a larger language model, is the accuracy likely to be higher?

Statistical Tools Data mining is more powerful with better statistical tools. Initially we used the Benjamini-Hochberg procedure with Fisher's exact test, which required us to pretrain 10 models to formally verify that the decaying instances exist. However, we later realized that 2 is in fact enough by using our approach introduced in Section 4. We could have saved 80% of the computation for pretraining if this approach was known before we started.

Future work can explore more complicated metrics and settings. We compared at most 3 different model sizes at a time, and higher order comparisons require novel metrics. We studied two sources of randomness, pretraining and finetuning, but other sources of variation can be interesting as well, e.g. differences in pretraining corpus, different model checkpoints, etc. To deal with more sophisticated metrics, handle different sources and hierarchies of randomness, and reach conclusions that are robust to noises at the instance level, researchers need to develop new inference procedures.

To conclude, for better instance level understanding, we need to produce and share more prediction data, annotate more diverse linguistic properties, and develop better statistical tools to infer under noises. We hope our work can inform researchers about the core challenges underlying instance level understanding and inspire future work.

Acknowledgement

We thank Steven Cao, Cathy Chen, Frances Ding, David Gaddy, Colin Li, and Alex Wei for giving comments on the initial paper draft. We would also like to thank the Google Cloud TPU team for their hardware support.

References

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,

pages 2744–2751, Online. Association for Computational Linguistics.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018a. Stress test evaluation for natural language inference. In *The 27th International Conference on Computational Linguistics (COLING)*, Santa Fe, New Mexico, USA.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018b. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yixin Nie, Xiang Zhou, and Mohit Bansal. 2020. What can we learn from collective human opinions on natural language inference data? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9131–9143, Online. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An investigation of why overparameterization exacerbates spurious correlations.

In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8346–8356. PMLR.

Lane Schwartz, Timothy Anderson, Jeremy Gwinnup, and Katherine Young. 2014. Machine translation and monolingual postediting: The AFRL WMT-14 system. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 186–194, Baltimore, Maryland, USA. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Tiago Pimentel, Hagen Blix, Arya D. McCarthy, Eleanor Chodroff, and Ryan Cotterell. 2020. Predicting declension class from form and meaning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6682–6695, Online. Association for Computational Linguistics.

Summary

In this 4-page appendix, we include 1) details on model training and the datasets we used 2) high-level ideas underlying the theoretical results, and 3) relatively more important discussions. We also have a longer version of this appendix on our github, which contains the full proofs, discussions, plots, and tables.

A Pretraining and Finetuning Details

To obtain model predictions under the "pretraining and finetuning" framework (Devlin et al., 2019), we need to decide a model **size**, perform **pretraining**, finetune on a **training set** with a choice of **hyper-parameters**, and test the model on an **evaluation set**. We discuss each bolded aspects below.

Size Similar to Turc et al. (2019), we experimented with the following five model sizes, listed in increasing order: MINI (L4/H256) [9] , SMALL (L4/H512), MEDIUM (L8/H512), BASE (L12/H768), and LARGE (L24/H1024).

Pretraining We used the pretraining code from Devlin et al. (2019) and the pre-training corpus from Li et al. (2020). Compared to the original BERT release, we used context size 128 instead of 512, since computation cost grows quadratically with respect to context size; we also pretrained for 2M steps instead of 1M.

Training Set We consider 3 datasets: Quora Question Pairs (QQP) [10], Multi-Genre Natural Language Inference (MNLI; Williams et al. (2020)), and the Stanford Sentiment Treebank (SST-2; (Socher et al., 2013)). For QQP we used the official training split. For MNLI we used 350K out of 400K instances from the original training split, and added the remaining 50K to the evaluation set, since the original in-domain development set only contains 10K examples. For SST-2, we mix the training and development set of the original split, split the instances into 5 folds, train on four of them, and evaluate on the remaining fold.

Hyperparameters As in Turc et al. (2019), we finetune 4 epochs for each dataset. For each task and model size, we tune hyperparameters in the following way: we first randomly split our new training set into 80% and 20%; then we finetune on the 80% split with all 9 combination of batch size [16, 32, 64] and learning rate [1e-4, 5e-5, 3e-5], and choose the combination that leads to the best average accuracy on the remaining 20%.

Evaluation Set After finetuning our pretrained models, we evaluate them on a range of in-domain, out-of-domain, or challenging datasets to obtain model predictions. Models trained on MNLI are also evaluated on Stanford Natural Language Inference (SNLI; (Bowman et al., 2015)), Heuristic Analysis for NLI Systems (HANS; (McCoy et al., 2019)), and stress test evaluations (STRESS; (Naik et al., 2018b)). Models trained on QQP are also evaluated on Twitter Paraphrase Database (TwitterPPDB; (Lan et al., 2017)).

Since pretraining introduces randomness, for each model size s, we pretrain 10 times with different random seed P; since finetuning also introduces noise, for each pretrained model we pretrain 5 times with different random seed F; besides, we also evaluate the model at the checkpoints after E epochs, where $E \in [3, 3\frac{1}{3}, 3\frac{2}{3}, 4]$.

Pretraining 10 models for all 5 model sizes altogether takes around 3840 hours on TPU v3 with 8 cores. Finetuning all of them 5 times for all three tasks in our paper requires around 1200 hours.

B Compare Our Models to the Original

Since we decreased the pre-training context length to save computation, these models are not exactly the same as the original BERT release by Devlin et al. (2019) and Turc et al. (2019). We need to benchmark our model against theirs to ensure that the performance of our model is still reasonable and the qualitative trend still holds. For each each size and task, we finetune the original model 5 times and calculate the average of overall accuracy.

The comparison can be seen in Table 8. We find that our model does not substantially differ from the original ones on QQP and SST-2. On MNLI, the performance of our BERT-BASE and BERT-LARGE is $2\sim3\%$ below the original release, but the qualitative trend that larger models have better accuracy still holds robustly.

C More Instance Difference Results

Similar to Figure 4, for all 10 pairs of model sizes and all in-distribution instances of MNLI, SST-2, and QQP, we plot the cumulative density of $\Delta\hat{\mathrm{Acc}}$ and $\Delta\mathrm{Acc}'$, or say, $\widehat{\mathrm{Decay}}(t)$ and $\mathrm{Decay}'(t)$. See the long appendix for the figures.

---

[9] 4 Layers with hidden dimension 256

[10] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

	QQP	MNLI	SST-2
MINI $^{\text{orig}}$	88.2%	74.6%	92.8%
MINI $^{\text{ours}}$	87.3%	74.3%	92.8%
SMALL $^{\text{orig}}$	89.1%	77.3%	93.9%
SMALL $^{\text{ours}}$	88.7%	76.7%	93.9%
MEDIUM $^{\text{orig}}$	89.8%	79.6%	94.2%
MEDIUM $^{\text{ours}}$	89.5%	78.9%	94.2%
BASE $^{\text{orig}}$	90.8%	83.8%	95.0%
BASE $^{\text{ours}}$	90.6%	81.2%	94.6%
LARGE $^{\text{orig}}$	91.3%	86.8%	95.2%
LARGE $^{\text{ours}}$	91.0%	83.8%	94.8%

Table 8: Comparing our pretrained model (superscript ^{orig}) to the original release by Devlin et al. (2019) and Turc et al. (2019) (superscript ^{ours}). All pretrained models are finetuned with the training set and tested on the in-distribution evaluation set described in Appendix A.

Additionally, for each pair of model sizes s_1 and s_2, we estimate "how much instances are getting better/worse accuracy?" by taking the maximum difference between the red curve and the blue curve. We report these results for MNLI, SST-2, and QQP in Table 9. We find that larger model size gaps lead to larger decaying fraction, but also larger improving fraction as well.

D Proof of Theorem 1

Formal Setup Suppose each instance is indexed by i, the set of all instances is $\mathcal{T}$, and the random seed is R; then $c_R^s \in \{0, 1\}^{|\mathcal{T}|}$ is a random $|\mathcal{T}|$ dimensional vector, where $c_{R,i}^s = 1$ if the model of size s correctly predicts instance i under the random seed R. We are comparing model size s_1 and s_2, where s_2 is larger; to keep notation uncluttered, we omit these indexes whenever possible.

Suppose we observe $c_{R_{1\dots 2k}}^{s_1}$ and $c_{R_{2k+1\dots 4k}}^{s_2}$, where there are $2k$ different random seeds for each model size [11]. Then

$$\Delta\hat{\text{Acc}}_i := \frac{1}{2k}\Big(\sum_{j=1}^{2k} c_{R_j,i}^{s_1} - \sum_{j=2k+1}^{4k} c_{R_j,i}^{s_2}\Big), \quad (16)$$

and hence the discovery rate $\widehat{\text{Decay}}(t)$ is defined as

$$\widehat{\text{Decay}}(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \mathbf{1}[\Delta\hat{\text{Acc}} \leq t]. \quad (17)$$

For the random baseline estimator, we have

$$\begin{aligned}\Delta\text{Acc}'_i := \frac{1}{2k}\Big(&\sum_{j=1}^{k} c_{R_j,i}^{s_1} + \sum_{j=2k+1}^{3k} c_{R_j,i}^{s_2} \quad (18)\\ &- \sum_{j=k+1}^{2k} c_{R_j,i}^{s_1} - \sum_{j=3k+1}^{4k} c_{R_j,i}^{s_2}\Big),\end{aligned}$$

and the false discovery control Decay' is defined as

$$\text{Decay}'(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \mathbf{1}[\Delta\text{Acc}'_i \leq t]. \quad (19)$$

Formally, theorem states that

$$\text{Decay} \geq \mathbb{E}_{R_1\dots R_{4k}}[\widehat{\text{Decay}}(t) - \text{Decay}'(t)] \quad (20)$$

Proof By re-arranging terms and linearity of expectation, Equation 20 is equivalent to the following

$$\begin{aligned}\sum_{i=1}^{|\mathcal{T}|}\big(&\mathbf{1}[\Delta\text{Acc}_i < 0] - \mathbb{P}[\Delta\hat{\text{Acc}}_i \leq t] \quad (21)\\ &+ \mathbb{P}[\Delta\text{Acc}'_i \leq t]\big) \geq 0\end{aligned}$$

Hence, we can declare victory if we can prove that for all i,

$$\begin{aligned}\mathbf{1}[\Delta\text{Acc}_i < 0] - \mathbb{P}[\Delta\hat{\text{Acc}}_i \leq t] \quad (22)\\ + \mathbb{P}[\Delta\text{Acc}'_i \leq t] \geq 0\end{aligned}$$

To prove Equation 22, we observe that if $\text{Acc}_i < 0$, since the probabilities are bounded by 0 and 1, its left-hand side must be positive. Therefore, we only need to prove that

$$\begin{aligned}&\Delta\text{Acc}_i \geq 0 \quad (23)\\ &\Rightarrow \mathbb{P}[\Delta\text{Acc}'_i \leq t] \geq \mathbb{P}[\Delta\hat{\text{Acc}}_i \leq t],\end{aligned}$$

which will be proved in Lemma 1.□

Lemma 1

$$\begin{aligned}&\Delta\text{Acc}_i \geq 0 \quad (24)\\ &\Rightarrow \mathbb{P}[\Delta\text{Acc}'_i \leq t] \geq \mathbb{P}[\Delta\hat{\text{Acc}}_i \leq t],\end{aligned}$$

For $m = 1, 2$, define

$$p_i^{s_m} := \mathbb{E}_R[c_i^{s_m}], \quad (25)$$

then

$$p_i^{s_1} \leq p_i^{s_2} \quad (26)$$

---

[11] We assumed even number of random seeds since we will mix half of the models from each size to compute the random baseline

MNLI $s_1 \setminus s_2$	MINI	SMALL	MEDIUM	BASE	LARGE
MINI	0.000	0.087	0.136	0.179	0.214
SMALL	0.033	0.000	0.089	0.139	0.180
MEDIUM	0.050	0.028	0.000	0.090	0.143
BASE	0.060	0.048	0.026	0.000	0.101
LARGE	0.059	0.052	0.040	0.021	0.000
QQP $s_1 \setminus s_2$	MINI	SMALL	MEDIUM	BASE	LARGE
MINI	0.000	0.057	0.076	0.100	0.107
SMALL	0.019	0.000	0.039	0.073	0.084
MEDIUM	0.029	0.014	0.000	0.044	0.063
BASE	0.034	0.027	0.016	0.000	0.032
LARGE	0.036	0.031	0.027	0.016	0.000
SST-2 $s_1 \setminus s_2$	MINI	SMALL	MEDIUM	BASE	LARGE
MINI	0.000	0.037	0.043	0.052	0.057
SMALL	0.010	0.000	0.015	0.031	0.036
MEDIUM	0.016	0.008	0.000	0.020	0.028
BASE	0.019	0.014	0.009	0.000	0.014
LARGE	0.020	0.017	0.015	0.008	0.000

Table 9: On QQP, MNLI in domain development set and SST-2 we lowerbound the fraction of instances that improves when model size changes from s_1 (row) to s_2 (column).

Since $c_i^{s_1}$ and $c_i^{s_2}$ are both Bernoulli random variables with rate $p_i^{s_1}$ and $p_i^{s_2}$ respectively, we can write down the probability distribution of $\Delta\hat{\text{Acc}}_i$ and $\Delta\text{Acc}'_i$ as the sum/difference of several binomial variables, i.e.

$$\Delta\hat{\text{Acc}}_i \sim (\text{Binom}(k, p_i^{s_2}) + \text{Binom}(k, p_i^{s_2}) - \text{Binom}(k, p_i^{s_1}) - \text{Binom}(k, p_i^{s_1}))/2k, \tag{27}$$

and

$$\Delta\text{Acc}'^{i} \sim (\text{Binom}(k, p^{s_2,i}) + \text{Binom}(k, p^{s_1,i}) - \text{Binom}(k, p^{s_1,i}) - \text{Binom}(k, p^{s_2,i}))/2k \tag{28}$$

$p_i^{s_1} \leq p_i^{s_2}$, Binom$(k, p^{s_2,i})$) first order stochastically dominates Binom$(k, p^{s_1,i})$. Therefore, $\Delta\text{Acc}'^{i}$ dominates $\Delta\hat{\text{Acc}}_i$, hence completing the proof. □

D.1 Independent Seed Assumption

See the long appendix for discussion.

E Upward Bias of Adaptive Thresholds

In section 3 we picked the best threshold that can maximize the lowerbound, which can incur a slight upward bias. Here we estimate that the bias is at most 10% relative to the unbiased lowerbound with a bootstrapping method.

We use the empirical distribution of 10 pretrained models as the ground truth distribution for bootstrapping. We first compute a best threshold with 10 sampled smaller and larger pretrained models, and then compute the lowerbound L with this threshold on another sample of 10 smaller and larger models. Intuitively, we use one bootstrap sample (which contains 10 smaller pretrained models and 10 larger pretrained models) as the development set to "tune the threshold", and then use this threshold on a fresh bootstrap sample to compute the lowerbound. We refer to the lowerbound that uses the best threshold as L^*, and compute the relative error $\mathbb{E}[(L^* - L)]/\mathbb{E}[L)]$, where the expectation is taken with respect to bootstrap samples.

See the long appendix for more detailed results. In general, we find that the upward bias is negligible, which is at most around 10%.

F Comparison with Significance Testing

See the long appendix for the Table that compares our procedure with the classical method that uses the Fisher's exact test and Benjamini-Hochberg procedure for other tasks and model size pairs.

In general, we find that our method always provide a tighter (higher) lowerbound than the classical method, and 2 models are sufficient to verify the existence (i.e. lowerbound > 0) of the decaying fraction; in contrast, the classical method sometimes fails to do this even with 10 models, e.g., when comparing BASE to LARGE .

Intuitively, our approach provides a better lower-bound because it better makes use of the information that on most instances, both the smaller and the larger models agree and predict completely correctly or incorrectly[12]. For an extreme example, suppose we only observe 2 smaller models and 2 larger models, and infinite number of datapoints, whose predictions are independent. On 99.98% datapoints, both models have instance accuracy 1; on 0.01% datapoints, smaller model is completely correct while bigger completely wrong, while on the rest 0.01% smaller completely wrong but bigger completely correct. Setting threshold to be 2, our decay estimate $\hat{\mathrm{Decay}}$ is 0.01%, while $\mathrm{Decay}' = 0$: since the models either completely predict correct or wrongly, there is never a false discovery. Therefore, our method can provide the tightest lowerbound 0.01% in this case. On the other hand, since we only have 4 models in total, the lowest significance-level given by the fisher exact test is $17\% \gg 0.1\%$, hence the discovery made by the Benjamin-Hochberg procedure is 0.

G More Results on Momentum

See the long appendix for correlations with other model size triplets on other tasks.

H Loss Decomposition and Estimation

The core of the $\mathrm{PretVar}$ estimator builds on the following theorem:

Theorem 2 *Suppose* $\mathcal{D}_k, k \in [\mathcal{F}]$ *are independently sampled from the same distribution* Ξ*, which is a distribution of distributions;* $\hat{\mu}_k$ *is an unbiased estimator of* $\mathbb{E}_{X\in\mathcal{D}_k}[X]$*, and* $\hat{\phi}_k$ *to be an unbiased estimator of the variance of* $\hat{\mu}_k$*, then*

$$\hat{Var}_F = \frac{1}{\mathcal{F}-1}\sum_{k\in[\mathcal{F}]}(\hat{\mu}_k - \hat{\mu})^2 - \frac{1}{\mathcal{F}}\sum_{k\in[\mathcal{F}]}\hat{\phi}_k \qquad (29)$$

is an unbiased estimator for

$$V = Var_{\mathcal{D}\sim\Xi}[\mathbb{E}_{X\sim\mathcal{D}}[X]], \qquad (30)$$

where

$$\hat{\mu} := \frac{1}{\mathcal{F}}\sum_{k\in[\mathcal{F}]}\hat{\mu}_k \qquad (31)$$

In this estimator, the first term "pretends" that $\hat{\mu}.$ are perfect estimator for the population mean and calculate the variance, while the second term corrects for the fact that the empirical mean estimation is not perfect. Notice the theorem only requires that $\hat{\mu}$ and $\hat{\phi}$ are unbiased, and is agnostic to the actual computation procedure by these estimators.

To estimate $\mathrm{PretVar}$, we need $\hat{\mu}_k$ and $\hat{\phi}_k$. The first term is the empirical accuracy for each pretraining seed; the second is an unbiased estimator of the variance of empirical accuracy for each pretraining seed, which can be estimated by the sample variance of finetuning divided by the number of finetuning runs.

See the long Appendix for a more detailed proof, and also how to generalize this estimator to estimate arbitrary level of variance decomposition.

I Variance Conditioned on Bias

See the long appendix for more figures.

J Example Decaying Instances

Here we present some random annotated decaying instances on MNLI.

Premise : and that you're very much right but the jury may or may not see it that way so you get a little anticipate you know anxious there and go well you know
Hypothesis : Jury's operate without the benefit of an education in law.
Label : Neutral, **Category** : Correct

Premise : In fiscal year 2000, it reported estimated improper Medicare Fee-for-Service payments of $11.
Hypothesis : The payments were improper.
Label : Entailment, **Category** : Fine

Premise : INTEREST RATE - The price charged per unit of money borrowed per year, or other unit of time, usually expressed as a percentage.
Hypothesis : Interest rate is defined as the total amount of money borrowed.
Label : Entailment, **Category** : Wrong

---

[12] This is for intuition, though, and we do not need any assumption on the prior of instance accuracy, which requires a Bayes interpretation.

Named Entity Recognition through Deep Representation Learning and Weak Supervision

Jerrod Parker
The Vanguard Group
University of Toronto
jerrod_parker@vanguard.com

Shi Yu
The Vanguard Group
shi_yu@vanguard.com

Abstract

Weakly supervised methods estimate the labels for a dataset using the predictions of several noisy supervision sources. Many machine learning practitioners have begun using weak supervision to more quickly and cheaply annotate data compared to traditional manual labeling. In this paper, we focus on the specific problem of weakly supervised named entity recognition (NER) and propose an end-to-end model to learn optimal assignments of latent NER tags using observed tokens and weak labels provided by labeling functions. To capture the sequential dependencies between the latent and observed variables, we propose a sequential graphical model where the components are approximated using neural networks. State-of-the-art contextual embeddings are used to further discriminate the quality of noisy weak labels in various contexts. Results of experiments on four public weakly supervised named entity recognition datasets show a significant improvement in F1 score over recent approaches.

1 Introduction

Many industries and organizations have collected large amounts of unlabeled text data that they want to make use of for various Natural Language Processing (NLP) applications. However, in many of these applications (named-entity recognition, question answering, text summarization, relation extraction), obtaining a large number of labels can be prohibitively expensive, error-prone, or otherwise infeasible. Furthermore, domain adaptation (Han and Eisenstein, 2019), which is commonly used in scarce label settings, can often struggle in new emerging / specific domains that don't have any closely related labeled datasets.

Under the absence of a closely related labeled dataset, *weakly supervised learning* is often used as a cheaper, less time-consuming alternative to obtaining gold standard labels. The main idea of weak supervision is to approximate the true labels by integrating multiple sets of noisy training labels. Each set of noisy labels (commonly referred to as "weak labels") is provided by a weak labeling function which often comes in the form of a knowledge base, heuristic, or pre-trained model. Weak supervision has obtained a lot of success in several NLP tasks containing approximately *i.i.d.* data such as topic classification (Bach et al., 2019), sentiment analysis, and social media content tagging (Fu et al., 2020).

Weak supervision on sequential data labeling problems such as NER is an emerging topic. Most current methods require either the time-consuming creation of additional heuristics such as 'linking rules'(Safranchik et al., 2020) or 'entity boundary detectors'(Fries et al., 2017), or assume that the accuracy of a weak labeler only depends on the true latent class (Safranchik et al., 2020; Lison et al., 2020; Fries et al., 2017). The latter is likely suboptimal since we would expect the accuracy to vary even within instances of the same class depending on the context given by surrounding tokens. One exception to this is the Fuzzy-LSTM-CRF(Shang et al., 2018). However, it ignores which weak labeler each prediction came from as well as the number of predictions for each class. This could be an issue when the weak labelers have differing accuracies, since the model may learn from a majority of wrong labels.

To address these foregoing issues, one of our main contributions is the proposal of an end-to-end method called **D**eep **W**eak Supervision on **S**equential Data (DWS), that learns context-dependent proficiency representations for weak labelers, enhanced further through contextual embeddings from pre-trained language models. In addition, instead of following the traditional approach that treats named entity (NE) tags as multi-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3828–3839
August 1–6, 2021.

nomial samples, we directly model the conditional dependency of tags on tokens as interactions of two representations: tag representation and contextual token representation.

Advantages of adopting representation learning are two-fold: First, learning context-aware proficiency of labeling functions enables the weak supervision procedure to denoise unreliable labels at tokens where labeling functions have many disagreements. Second, because embedding methods have demonstrated great potential in capturing semantic meanings, the proposed model allows flexible transfer of existing NER pipelines to new domains through leveraging pre-trained domain-specific embeddings.

DWS relies on a graphical model to capture statistical dependencies among tokens, weak labels and true latent NE tags. However, latent variable estimation is challenging and the techniques are often both sample and computationally complex. For example, Ratner et al. (2017) required a Gibbs-based algorithm, and Ratner et al. (2019) required estimating the full inverse covariance matrix among the labelers. In video analysis, Varma et al. (2019b) required the use of multiple iterations of stochastic gradient descent (SGD) to learn accuracy parameters, but the dependencies are limited to weak labels and true labels. When context of the sequential inputs are directly involved in the model, optimizing or even formulating the analytical solution becomes much more difficult.

Our solution is motivated by the advantages of using neural networks to model the transition and output distributions (Bengio and Frasconi, 1996; Li and Shum, 2006). Instead of deriving analytical formulations to learn the parameters for the given structure, we use deep neural networks to approximate conditional dependencies and sequential transitions. Furthermore, the marginal likelihood of the proposed model is optimized via hard EM (Min et al., 2019) to find the most probable sequence of latent tags. Such a hybrid process allows us to easily integrate distributed representations in our model and also enables us to explore complex model structures.

We benchmarked the proposed DWS model on several NER datasets and compared it with some recent weak supervision approaches. Experimental results show DWS's advantage on tagging tasks where there are a lot of conflicting weak labels. Furthermore, we conduct detailed analysis to evaluate the complexity of these NER datasets in terms of the number of NE tags and labeling functions and also the amount of inconsistency among the weak labels.

We conclude that some datasets used in past works are useful to evaluate the robustness of weak supervision models, whereas others contain weak labels that are trivial to denoise and result in all algorithms looking equivalent. Therefore, directly comparing results on weakly labeled datasets without accounting for the difficulty of denoising their labels can drive misleading conclusions. To more rigorously test how the performance of the algorithms scale with the difficulty of the weak supervision problem, we introduce a new method which stratifies a dataset into tokens containing varying levels of weak labeler disagreement, quantified by the entropy of the weak label predictions, and compares how the performance of each algorithm scales with the difficulty of the denoising task. Results show that the performance advantage of our algorithm over the current methods grows quickly with respect to the disagreement among the weak labelers.

2 Related Work

Snorkel (Ratner et al., 2017) is a well-known tool that learns a generative model to estimate the accuracies and correlations between weak labelers on *i.i.d.* data. SwellShark (Fries et al., 2017) treats weakly supervised NER as an *i.i.d.* task but requires creating additional heuristics (sometimes called entity span generators) to find the entity boundaries. In recent years, various new algorithms have been introduced to extend the weak supervision idea to tasks involving sequential data such as NER. BOND (Liang et al., 2020) and AutoNER (Shang et al., 2018) respectively denoise predictions of a single weak labeler and a set of dictionaries. Several generative approaches such as Hidden Markov Model (Lison et al., 2020; Safranchik et al., 2020) and a discriminative method Fuzzy-LSTM-CRF (Shang et al., 2018) have been proposed to model the dependencies between NE tags; thus, entity span generators are no longer needed. There have also been several relevant approaches on other weak supervision tasks. ReHession (Liu et al., 2017) was developed for weakly supervised relation extraction and tries to learn the contexts where each weak labeler is proficient, then uses that knowledge to infer the true labels. Varma et al.

(2019a) proposes a robust PCA-based algorithm to learn dependency structures for image classification. In the application of video analysis, Fu et al. (2020) applies a general binary Ising model to factorize likelihood expectations over cliques so an analytical solution is found to speed up the model parameter learning.

In this paper, we systematically compare most of the recent approaches proposed for NER except for BOND, AutoNER, and Swellshark which are not directly applicable to our experimental settings.

3 Problem Definition

We assume a sequential labeling problem formulation where we are given a sequence of tokens $\mathcal{X} = \{x_1, ..., x_N\}$ which map to a sequence of latent class variables $\mathcal{Y} = \{y_1, .., y_N\}$. In a fully supervised scenario, $\mathcal{Y}$ is usually partially annotated thus $\mathcal{Y} = \mathcal{Y}_{\text{train}} \cup \mathcal{Y}_{\text{test}}$, so model parameters are estimated from $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$. Because $\mathcal{Y}_{\text{train}}$ is usually expensive to obtain, our primary goal is to estimate $\mathcal{Y}_{\text{train}}$ with the help of multiple, potentially noisy labeling sources. Suppose we have a set of weak labels $\mathcal{L}_{\text{train}}$ available for $\mathcal{X}_{\text{train}}$, where each token is assigned a set of weak labels provided by m different labeling sources $\lambda_1, ..\lambda_m$ (labelers can choose to make no prediction by voting 'Abstain'). For simplicity, we drop the notion of training and test data, and simply use $\mathcal{X}, \mathcal{Y}, \mathcal{L}$ to represent all tokens, latent tags, and weak labels in our problem. Additionally, we define the vote of weak labeler λ_j on token x_i as $l_{i,j}$.

In a supervised setting where $\mathcal{Y}$ are given, we learn model parameters Θ by maximizing the log-likelihood of $\mathcal{Y}$ given the input $\mathcal{X}$ with respect to Θ:

$$J_{sup}(\Theta|\mathcal{X}, \mathcal{Y}) = \max \log P(\mathcal{Y}|\mathcal{X}; \Theta) \tag{1}$$

whereas in our weak supervision scenario, $\mathcal{Y}$ are hidden and $\mathcal{L}$ are fully observable, so the learning objective can instead be to maximize the marginal likelihood of the weak labels. Letting $\mathcal{E}$ be the set of K entity classes which includes 'No Entity', we have:

$$P(\mathcal{L}|\mathcal{X}; \Theta) = \sum_{Y_c \in \mathcal{E}^N} P(\mathcal{L}, Y_c|\mathcal{X}; \Theta) \tag{2}$$

which can be used to compute the objective as follows:

$$J_{weak}(\Theta|\mathcal{X}, \mathcal{L}) = \max \log P(\mathcal{L}|\mathcal{X}; \Theta) \tag{3}$$

4 Model Overview

4.1 Model Structure and Likelihood

To incorporate the dependencies among tokens, weak labels, and true labels, we define a graphical structure illustrated as Figure 1. The proposed graphical model is partially directed and its analytical solutions and exact inference are available through segmentations of the input sequence but are usually complex to derive. In our approach, the likelihood $P(\mathcal{L}, \mathcal{Y}|\mathcal{X})$ defined in (2) is factorized as $P(\mathcal{L}|\mathcal{X}, \mathcal{Y})$ and $P(\mathcal{Y}|\mathcal{X})$, where each is approximated via a neural network. These networks are described in Sections 4.2 and 4.3, respectively. The parameters are learned through maximizing an approximation to the marginal likelihood $P(\mathcal{L}|\mathcal{X})$ using a hard EM algorithm. Following is an in depth description of the model formulation.

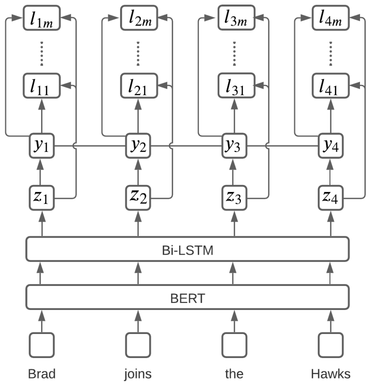

Figure 1: The DWS model where z_i denotes the contextual embedding for the token at index i.

According to (3) and (2) we have:

$$\begin{aligned} P(\mathcal{L}|\mathcal{X}) &= \sum_{Y_c \in \mathcal{E}^N} P(\mathcal{L}, Y_c|\mathcal{X}) \\ &= \sum_{Y_c \in \mathcal{E}^N} P(\mathcal{L}|\mathcal{X}, Y_c) P(Y_c|\mathcal{X}) \end{aligned}$$

We now assume different labeling functions are independent of each other given the input text and true labels. Additionally, we assume the weak labels at token x_i are conditionally independent of the true labels at any index $k \neq i$ when given the input text and true label at index i (d-separation details are in Appendix A.2). Therefore, we have:

$$P(\mathcal{L}|\mathcal{X}, Y_c) = \prod_i \prod_j P(l_{i,j}|\mathcal{X}, Y_c) \tag{4}$$

$$= \prod_i \prod_j P(l_{i,j}|\mathcal{X}, y_{c,i}) \tag{5}$$

where $y_{c,i}$ denotes the label at index i of Y_c. This means:

$$\log P(\mathcal{L}|\mathcal{X}) = \log \sum_{Y_c \in \mathcal{E}^N} P(Y_c|\mathcal{X}) \prod_i \prod_j P(l_{i,j}|\mathcal{X}, y_{c,i}) \tag{6}$$

In it's current form, the conditional log likelihood is difficult to optimize because of the sum-product inside the logarithm. As an alternative, we maximize J_{HardEM} which is an approximation to the conditional log-likelihood where the summation over entity labels is replaced with a maximum.

$$\begin{aligned} J_{HardEM} &= \log \Big[\max_{Y_c \in \mathcal{E}^N} P(Y_c|\mathcal{X}) \prod_i \prod_j P(l_{i,j}|\mathcal{X}, y_{c,i}) \Big] \\ &= \max_{Y_c \in \mathcal{E}^N} \Big[\log P(Y_c|\mathcal{X}) + \sum_i \sum_j \log P(l_{i,j}|\mathcal{X}, y_{c,i}) \Big] \\ &= \max_{Y_c \in \mathcal{E}^N} \log P(\mathcal{L}, Y_c|\mathcal{X}) \end{aligned} \tag{7}$$

This new optimization problem attempts to find accurate modes of $P(\mathcal{L}, \mathcal{Y}|\mathcal{X})$. All that remains is to formulate $P(Y_c|\mathcal{X})$ and $P(l_{i,j}|\mathcal{X}, y_{c,i})$ for some pair (i, j).

4.2 Modeling Weak Labeler Representations

In practice, a weak labeler is usually derived from a specific rule or a controlled vocabulary. Thus, it is reasonable to assume its accuracy depends on the context of the token it is making a prediction on. Inspired by Liu et al. (2017), we model labeling function λ_j providing the correct label to x_i as a discrete event following a *Bernoulli* distribution, given by:

$$P(l_{i,j}|\mathcal{X}, y_{c,i}) = \begin{cases} \alpha_{ij} & \text{if } l_{i,j} = y_{c,i} \\ 1 - \alpha_{ij} & \text{if } l_{i,j} \neq y_{c,i} \end{cases} \tag{8}$$

Here, $\alpha_{ij} = \sigma(\frac{z_i^T \theta_j}{\sqrt{d}})$, σ is the sigmoid function, θ_j is a learnable embedding specific to λ_j, and $z_i (z_i \in \mathbb{R}^d)$ is the contextual embedding of the token x_i. This formulation improves the modeling capacity over many past methods such as Snorkel (Ratner et al., 2017) or the HMM (Safranchik et al., 2020; Lison et al., 2020) which have purely class-conditioned accuracies. In addition, it allows the utilization of external knowledge from large pretrained language models by making z_i a function of their contextual embeddings. When a weak labeler abstains, there is no concept of accuracy. We chose to set the probability of the 'Abstain' votes to 1, but in the future could instead model the probability of abstaining.

4.3 Modeling Class Representations and Transition Scores

We model $P(Y_c|\mathcal{X})$ as a function of the contextual embeddings z. The model uses a linear-chain conditional random field (CRF) output layer which is often utilized to model dependencies between labels (Huang et al., 2015; Lample et al., 2016; Akbik et al., 2018). The distribution $P(Y_c|\mathcal{X})$ is given by

$$P(Y_c|\mathcal{X})) = \frac{e^{C(\mathcal{X}, Y_c)}}{\sum_{Y' \in \mathcal{E}^N} e^{C(\mathcal{X}, Y')}} \tag{9}$$

$$\text{where} \quad C(\mathcal{X}, Y) = \sum_{j=1}^{N} s_{j,y_j} + \sum_{j=1}^{N-1} T[y_j, y_{j+1}] \tag{10}$$

$$\text{and} \qquad s_{j,y_j} = \frac{z_j^T t_{y_j}}{\sqrt{d}} \tag{11}$$

where T is a learnable matrix defining the transition scores between any two entity classes and t_{y_i} is the learnable embedding for class y_i. We have scaled the dot products in both the formula for α_{ij} and (11), by the square root of the embedding size d to significantly increase the stability of training as was shown to be useful in Vaswani et al. (2017).

4.4 Algorithm for Optimization

To maximize the objective J_{HardEM}, we repeat the following two steps:

1. Calculate $Y' = \arg\max_{Y_c \in \mathcal{E}^N} \Big(\log P(\mathcal{L}, Y_c|\mathcal{X}) \Big)$
2. Maximize $\log P(\mathcal{L}, Y'|\mathcal{X})$ for one gradient ascent step.

This approach is often termed *hard EM* and has been successful in other areas such as weakly supervised relation extraction (Liu et al., 2017) and question answering (Min et al., 2019). Step 1 can be computed as follows:

$$Y' = \arg\max_{Y_c \in \mathcal{E}^N} \Big(\log P(\mathcal{L}, Y_c|\mathcal{X}) \Big) \tag{12}$$

$$= \arg\max_{Y_c \in \mathcal{E}^N} \Big(\sum_{i=1}^{N} s'_{i,y_{c,i}} + \sum_{k=1}^{N-1} T[y_{c,k}, y_{c,k+1}] \Big) \tag{13}$$

$$\text{where} \quad s'_{i,y_{c,i}} = s_{i,y_{c,i}} + \sum_{j=1}^{m} \log P(l_{i,j}|\mathcal{X}, y_{c,i}) \tag{14}$$

The details of the derivation are in Appendix A.1. Equation (13) can be solved efficiently with the same Viterbi decoding algorithm used in Huang et al. (2015). In practice, we constrain each y'_i to be in the set of classes voted on among weak labelers on the token x_i by setting $s'_{i,y_{c,i}} = -\infty$ for any $y_{c,i}$ that does not satisfy the constraint. Additionally, we found it useful to penalize choosing sequences Y' containing illegal class transitions. This was done by adding a hyperparameter τ to the transition matrix T from (13) in all locations that correspond to illegal transitions. For example, transitioning from the middle to the beginning of a 'Person' entity (i.e, I-Per → B-Per) would be penalized as it does not make sense.

4.5 Training Details

The contextual token embeddings z used to calculate (8) and (11) are obtained from a trainable single layer bidirectional LSTM that uses contextual word embeddings from pre-trained BERT models as described in Section 5.2. We choose to freeze the parameters of the pre-trained models to allow fast and inexpensive training.

To obtain a better starting point for the EM algorithm, the models are given a warm start by training for one epoch where the token labels Y' in step 2 of the optimization algorithm in Section 4.4 are set to be the majority vote labels.

Lastly, we choose the model parameters to be those that give the best entity F1 score on a held out validation set over 10 random restarts of 5 epochs each. Further details of the training procedure and architectures used are described in Appendix A.3 and A.4.

5 Experiments and Results

Experiments are conducted on four public weakly supervised NER datasets (Lison et al., 2020; Safranchik et al., 2020) as summarized in Table 1. We use the same weak labeling functions as reported in these approaches. Each dataset is split into train, validation, and test set using the same splits as Safranchik et al. (2020) suggests for NCBI-Disease, BC5CDR, and LaptopReview, and Liang et al. (2020) for CoNLL2003[1].

5.1 Weakly Labeled Dataset Difficulty

We first conduct a study to establish the difficulty of denoising the weak labels in each dataset. As we will demonstrate, this allows a detailed analysis of the strengths and weaknesses of each model. Both the amount and types of disagreement among the weak labelers differ drastically between weakly supervised NER datasets. Knowledge about the types of difficulties in each dataset is necessary to understand the contexts where each model is most useful as well as how to gauge which results are the most important. We categorize disagreements into three types: **Token Position** (i.e., $\{B, I, L, U, O\}$), **Positioned Class** (i.e., $\{B\text{-}Per, I\text{-}Per, L\text{-}Per\}$), and **Unpositioned Class** (i.e., $\{Per, Loc, Org, Misc\}$).

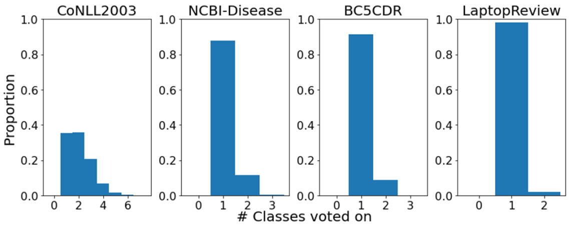

Figure 2: The tokens in each dataset containing weak labels were bucketed based on the number of classes voted on among weak labelers. This figure shows the proportion of tokens in each bucket.

To understand the prominent types of disagreement in each weakly labeled dataset, we calculated the number of tokens which contain each type of disagreement. The results are displayed in the last three columns of Table 1, and show that weak labelers on NCBI-Disease, BC5CDR, and LaptopReview disagree on a token's positioning within an entity much more often than on the unpositioned entity class. This indicates that the main difficulty on those weakly labeled datasets is resolving disagreements on where each entity begins and ends. To the contrary, CoNLL2003 contains many disagreements on both the token position and unpositioned class predictions.

We also include Figure 2 to emphasize the difficulty of the weakly labeled CoNLL2003 dataset compared to the others. On average there are a much higher number of classes being disagreed upon per token, a heuristic for the difficulty of the weakly labeled dataset.

5.2 Results

Our experiments focus on the following topics:
(1) Comparison of the proposed method's performance with existing methods on benchmark datasets. (2) Stratified analysis and benchmarking of various models w.r.t. the difficulty of the NER task. (3) Systematic study to establish the importance of various design decisions of our proposed method.

The proposed DWS model is compared to sev-

---

[1] Lison et al. (2020) did not define a validation set or test set split. Instead, their results were on the full dataset.

Dataset	Document Type	#WL's	#Tokens	#Entities	Entity Types	#Unpositioned	#Positioned	#Position
CoNLL2003	News	46	301,649	35,089	Per,Loc,Org,Misc	7439	9236	6203
LaptopReview	Online reviews	12	63,379	3,005	Laptop Aspect	23	43	40
NCBI-Disease	PubMed Articles	12	183,330	6,873	Disease	310	923	904
BC5CDR	PubMed Articles	26	351,508	28,687	Chemical,Disease	216	1058	1001

Table 1: Summary statistics of the four datasets where WL refers to 'Weak Labelers'. The weak labels for CoNLL2003 are defined by Lison et al. (2020) and the weak labels for NCBI-Disease, LaptopReview, and BC5CDR are defined by Safranchik et al. (2020). The last three columns are defined as: **'#Unpositioned'**: is the number of tokens with more than one unpositioned class voted on. **'#Positioned'**: is the number of tokens with more than one positioned class voted on. **'#Position'**: is the number of tokens with more than one position (i.e, B,I,L,U) voted on. On CoNLL2003, the weak labelers were created such that their votes are in the set of the 19 OntoNotes5.0 classes. The weak supervision model is trained using this full set of classes and then at inference time, it's predictions are mapped to the set of CoNLL2003 classes or 'No Entity' for types such as Date or Ordinal.

eral recently introduced models which are trained on the benchmark datasets. Those include **Snorkel** (Ratner et al., 2017) where each token is considered as an independent example, **Fuzzy-LSTM-CRF** (Shang et al., 2018), **HMM** (Lison et al., 2020), the HMM formulation from Safranchik et al. (2020) which does not use 'linking rules', **Majority Vote**, and **Unweighted Vote** (Safranchik et al., 2020) which creates probabilistic training labels using the empirical distribution of the weak labels on each token. To make a fair comparison with DWS, the Fuzzy-LSTM-CRF uses the same model architecture as its discriminative component $P(\mathcal{Y}|\mathcal{X})$. To enhance performance, we use contextual token embeddings from RoBERTa (Liu et al., 2019) on CoNLL2003, uncased BERT (Devlin et al., 2019) on Laptop Review, and BioBERT (Lee et al., 2019) on NCBI-Disease and BC5CDR as inputs to our model and the Fuzzy-LSTM-CRF.

Our weak supervision pipeline has two steps: **Step 1)** First it learns the latent labels of the training data using weak labels and tokens, so the learned data can be used as annotated training data to train NER classifiers. We measure model performance in this step through evaluating the micro-entity precision, recall, and F1 scores of the learned latent labels against the ground truth labels. Table 2 (upper) displays these results and shows that the proposed DWS method creates training labels with an F1 score of 1.65% higher than the nearest comparing model on CoNLL2003 and F1 scores within 0.1% and 0.03% of the best comparative approach on BC5CDR and NCBI-Disease, respectively. Additionally, DWS is outperformed by an F1 score of 0.58% on LaptopReview but this is expected since the dataset only contains 43 tokens where the weak labelers disagree. This is likely far too little for DWS to learn robust deep representations for the accuracies of each weak labeler.

Step 2) Using the learned labels as training labels, we train classifiers and apply the trained classifiers on test data. We train the same classifier mentioned in Safranchik et al. (2020) on the weakly labeled training data obtained by each algorithm on all benchmark datasets. We also keep the test data identical per each dataset and report the obtained performance on test data using micro-entity precision, recall, and F1 score. According to the results displayed in Table 2 (lower), the proposed DWS outperforms the next best model by an F1 score of 3.95% on CoNLL2003, which is arguably the hardest dataset given the analysis in Section 5.1. Additionally, DWS outperforms the closest competitor on NCBI-Disease by an F1 score of 0.83%, and achieves the second best performance on both BC5CDR and LaptopReview.

The purpose of reporting performance separately in two steps is to clearly demonstrate the improvement of performance obtained by weak supervision alone, which only compares the quality of learned annotations with true labels in Step 1. To our understandings, this is important because in practice, model selection and quality control of weak supervision should also focus on the quality of learned annotations. The second step, after the training data is automatically annotated, focuses on the bias-variance problem in a supervised scenario and the main goal is to select the best classifier to generalize well on unseen data.

5.3 Performance vs Entropy of Weak Labels

As previously discussed, comparing F1 scores alone across weak supervision approaches without knowing the difficulties of the problem could drive misleading conclusions. For example, if the weak labels provided for a dataset are highly correlated, it becomes a trivial problem to denoise them - meaning algorithms will have indistinguish-

Step 1

Model	**CoNLL2003**			**BC5CDR**			**NCBI-Disease**			**LaptopReview**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Majority Vote	71.11	69.98	70.54	86.19	83.52	84.83	74.78	63.42	68.63	67.09	60.88	63.83
Snorkel	68.45	72.23	70.28	81.66	80.63	81.26	77.33	65.05	70.66	67.57	61.01	64.12
HMM1*	73.15	72.09	72.62	**89.72**	70.25	78.80	80.27	41.25	54.50	67.88	59.52	63.43
HMM2*	N/A	N/A	N/A	85.73	**83.96**	84.84	77.76	63.69	70.03	68.09	**61.46**	**64.61**
Fuzzy-LSTM-CRF**	74.92	74.02	74.46	88.17	83.38	**85.71**	**80.97**	64.99	**72.11**	**68.57**	60.57	64.34
DWS (ours)	**75.58**	**76.65**	**76.11**	87.47	83.84	85.61	79.78	**65.74**	72.08	68.06	60.45	64.03

Step 2

Model	**CoNLL2003**			**BC5CDR**			**NCBI-Disease**			**LaptopReview**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Majority Vote	71.51	71.71	71.60	82.40	82.06	82.23	70.98	71.72	71.31	65.79	59.57	62.51
Unweighted Vote	68.00	69.67	68.82	83.0	81.86	82.43	77.32	73.59	75.39	66.61	58.90	62.51
Snorkel	72.68	71.70	72.18	80.23	**84.35**	82.24	71.10	**76.00**	73.41	64.09	**63.09**	**63.54**
HMM1*	70.78	70.28	70.53	81.27	73.90	77.39	78.63	51.00	61.82	66.29	56.21	60.80
HMM2*	NA	NA	NA	80.21	84.30	82.21	72.21	70.54	71.34	66.17	59.94	62.86
Fuzzy-LSTM-CRF**	72.32	71.05	71.68	83.36	82.70	**83.03**	78.67	70.77	74.50	**69.42**	57.77	63.05
DWS (ours)	**76.96**	**75.32**	**76.13**	**84.15**	81.88	82.99	78.60	74.01	**76.22**	69.08	58.75	63.44

Table 2: **Step 1:** Training set results of the training labels produced by the weak supervision models. **Step 2:** Test set results of discriminators trained on the label predictions of the weakly supervised NER algorithms. All of the results in both tables are the average scores over 5 runs of each model using different random seeds.
* HMM1 is from Lison et al. (2020) and HMM2 is the Hidden Markov Model from Safranchik et al. (2020) without 'linking' rules. Note that we attempted to train HMM2 on CoNLL2003 but ran out of memory when using 64GB of RAM.
** We use our own implementation for the Fuzzy-LSTM-CRF. The results differ from Shang et al. (2018) since different weak labelers are used.

able F1 scores. To more clearly differentiate the performance of algorithms, we can calculate how their performance scales with the difficulty of the denoising task. Intuitively, we expect that better algorithms should be more robust to challenging denoising tasks. To demonstrate this, we first bucket tokens in the weakly labeled CoNLL2003 training dataset based on the level of disagreement among their weak labels, which can be thought of as the difficulty of denoising them, and then plot the differences in average token F1 scores between DWS and the comparative models in each bucket. To quantify the 'disagreement' of the weak labelers on a token, we use the entropy of the distribution of their non-abstained votes. To gain further insight into the strengths and weaknesses of each model, we group entropy and F1 calculations by 3 different class types: **Token Position**, **Positioned Class**, **Unpositioned Class**. These are described in Section 5.1. The results are plotted in Figure 3.

Figure 3 shows the improvement in F1 score obtained by DWS over other comparing methods at several different levels of entropy. The left figure shows that as entropy increases, generally the improvement of performance on unpositioned classes also increases. The middle figure shows that our model does not have much of an advantage in resolving disagreements over the position each token has within an entity (ie B,I,L,U), and that Fuzzy-LSTM-CRF is actually significantly better at high entropy levels. Most importantly though, when both types of disagreement are combined in Figure 3 (right), the advantage of DWS becomes very significant as the gaps in performance between itself and others are steadily increasing functions.

5.4 Ablation Studies

To better understand the importance of each component in DWS, we study how the performance of DWS on CoNLL2003 is affected by removing some design functions in DWS. Specifics of the experiments are as follows:

No Penalty: Sets the penalty given to illegal class transitions (defined in Section 4.4) to 0. Without this illegal class transition penalty, validation and test F1 scores drop by 1.27% and 2.34% respectively.

No Penalty No CRF: In addition to removing the penalty for illegal tag transitions as explained above, we treat the true tags as conditionally independent by replacing the CRF in $P(Y|X)$ with a token-wise softmax. Such changes result in lowering the test F1 score by 8.77%. This experiment helps to highlight the importance of modeling sequential dependencies since without it, the performance is lower than even majority voting.

No Warm Start: Warm starting by initializing the hard EM procedure with majority votes is helpful

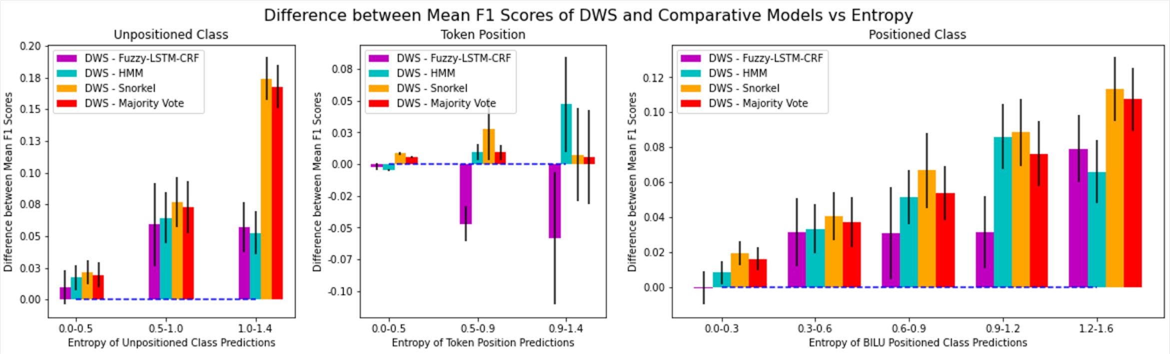

Figure 3: Each plot contains 95% confidence intervals on the difference in mean token F1 scores between DWS and comparing models in each bucket. These were calculated using the results from 5 randomized runs for each model. When the entropy is large, there is a lot of disagreement; when it's zero, the weak labelers all agree on the same class.

Model	CoNLL2003					
	Dev Precision	**Dev Recall**	**Dev F1**	**Test Precision**	**Test Recall**	**Test F1**
DWS	79.13	77.36	78.23	76.96	75.32	76.13
DWS (No Penalty)	77.92	76.02	76.96	74.35	73.25	73.79
DWS (No Penalty or Crf)	72.88	69.54	71.16	69.56	65.30	67.36
DWS (No Warm Start)	77.39	76.77	77.07	73.43	73.40	73.41

Table 3: Validation and test set results of a discriminator trained on the label predictions of each model listed in the table. The micro-entity precision, recall, and F1 scores are averaged over 5 random seeds.

to achieve better performance compared to random initialization. After iterating hard EM for the same epochs, warm start boosts validation and test F1 scores by 1.16% and 2.72% respectively, compared to random initialization. The results of each ablation are reported in Table 3.

6 Discussion

In Section 5.1 we showed that the amount of disagreement between the weak labelers varied substantially between datasets. This insight is important when interpreting the effectiveness of a weak supervision model because on datasets containing very little disagreement, we wouldn't expect to be able to learn anything much different than majority voting. As the number of contradictions between the weak labeler votes increases, for the most part so does the amount of information to help gauge the accuracies of the weak labelers and surpass majority voting. This means that we would roughly expect a good algorithm to have its greatest performance advantage on CoNLL2003 followed by NCBI-Disease and BC5CDR, and lastly LaptopReview which did not contain many weak labeler disagreements. The results in Section 5.2 show that as we would hope, our method has the largest performance advantage on CoNLL2003 and retains strong results on the remaining datasets which have fewer disagreements.

In Section 5.3 ('Performance vs Entropy'), we more concretely show that the F1 score advantage of DWS over the other models when using positioned labels increases with respect to the amount of contradiction information. These advantages may be attributable to the learned proficiency representations of weak labelers that help discriminate noisy labels provided by labeling functions with low proficiencies.

Experiments in Section 5.3 also show that DWS is very effective at resolving disagreements on unpositioned classes but is more mediocre at denoising disagreements on the entity positioning. These results suggest that the ideal model to use in a practical scenario likely depends on the amount of each type of disagreement in the given weakly labeled dataset.

7 Conclusion

In this paper we introduced a novel method (DWS) for weakly supervised NER which learns context-dependent proficiency for labeling sources while also modeling sequential dependencies among weak labels, inputs, and true labels. The proposed approach integrates representation learning and graphical models within the weak supervision setting, and obtains better performance on public benchmark datasets than recently introduced approaches. The proposed method is quite generic and can be applied to other sequential learning tasks in NLP or other modalities such as image analysis/computer vision by adopting various pre-trained domain-specific models to embed the input sequence.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649. Association for Computational Linguistics.

Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Chris Ré, and Rob Malkin. 2019. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 362–375, New York, NY, USA. Association for Computing Machinery.

Yoshua Bengio and Paolo Frasconi. 1996. Input-output hmms for sequence processing. *IEEE Trans. Neural Networks*, 7(5):1231–1249.

Yoshua Bengio, Renato De Mori, Giovanni Flammia, and Ralf Kompe. 1992. Global optimization of a neural network-hidden markov model hybrid. *IEEE Trans. Neural Networks*, 3(2):252–259.

Hervé Bourlard and Christian Wellekens. 1990. Links between markov models and multilayer perceptrons. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(12):1167–1178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data.

Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3280–3291. PMLR.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4238–4248, Hong Kong, China. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Yan Li and Heung-Yeung Shum. 2006. Learning dynamic audio-visual mapping with input-output hidden markov models. *IEEE Trans. Multim.*, 8(3):542–549.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. *BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision*, page 1054–1064. Association for Computing Machinery, New York, NY, USA.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533, Online. Association for Computational Linguistics.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 46–56. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering.

In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2851–2864, Hong Kong, China. Association for Computational Linguistics.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4763–4771.

Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020. Weakly supervised sequence tagging from noisy rules. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5570–5578.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064, Brussels, Belgium. Association for Computational Linguistics.

Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. 2019a. Learning dependency structures for weak supervision models. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6418–6427. PMLR.

Paroma Varma, Frederic Sala, Shiori Sagawa, Jason Alan Fries, Daniel Y. Fu, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, and Christopher Ré. 2019b. Multi-resolution weak supervision for sequential data. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 192–203.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

A Appendix

A.1 Derivations

The details for efficiently computing Y' in step 1 of the hard EM algorithm as mentioned in Section 4.4 are as follows:

$$
\begin{aligned}
Y' &= \arg\max_{Y_c \in \mathcal{E}^N} \Big(\log(P(\mathcal{L}, Y_c|\mathcal{X})) \Big) && (15) \\
&= \arg\max_{Y_c \in \mathcal{E}^N} \Big(\log(P(Y_c|\mathcal{X})) + && (16) \\
&\qquad \sum_{i=1}^{N} \sum_{j=1}^{m} \log(P(l_{i,j}|\mathcal{X}, y_{c,i})) \Big) \\
&= \arg\max_{Y_c \in \mathcal{E}^N} \Big(\log(\frac{e^{C(X,Y_c)}}{K}) + && (17) \\
&\qquad \sum_{i=1}^{N} \sum_{j=1}^{m} \log\left(P(l_{i,j}|\mathcal{X}, y_{c,i})\right) \Big) \\
&= \arg\max_{Y_c \in \mathcal{E}^N} \Big(\sum_{i=1}^{N} s_{i,y_{c,i}} + \sum_{k=1}^{N-1} T[y_{c,k}, y_{c,k+1}] + \\
&\qquad \sum_{i=1}^{N} \sum_{j=1}^{m} \log(P(l_{i,j}|X, y_{c,i})) \Big) \\
&= \arg\max_{Y_c \in \mathcal{E}^N} \Big(\sum_{i=1}^{N} s'_{i,y_{c,i}} + \sum_{k=1}^{N-1} T[y_{c,k}, y_{c,k+1}] \Big) && (18) \\
&\text{where} \quad s'_{i,y_{c,i}} = s_{i,y_{c,i}} + \sum_{j=1}^{m} \log\left(P(l_{i,j}|X, y_{c,i})\right)
\end{aligned}
$$

where K is a normalizing constant and C is the function defined in equation 10.

A.2 Graphical Model of DWS

We illustrate the exact graphical model of DWS in Figure 4, where the structure is similar to the Input/Output HMM(IOHMM) proposed by (Bengio and Frasconi, 1996). In IOHMM, transitions from latent variables y_{i-1} to y_i are directional, and analytical solutions are available based on factorizations of the sequence marginal likelihood. Here in our approach, similar analytical solutions could be derived as a partially directed model (e.g., CRF) without the need of parameterized distributions of input embeddings z_i. However, hybrid approaches that integrate neural networks with graphical models to enable efficient and scalable model training are preferred (Bourlard and Wellekens, 1990; Bengio et al., 1992; Bengio and Frasconi, 1996; Li and Shum, 2006).

A.3 Weak Supervision Model Architecture / Training Details

Weak Supervision Model Architectures: DWS and the Fuzzy-LSTM-CRF used a one layer

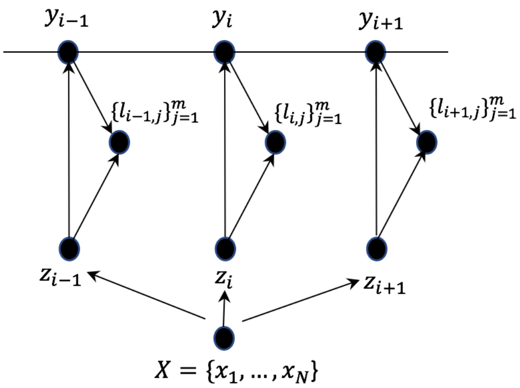

Figure 4: The input sequence $X = \{x_1, ..., x_N\}$ is transformed to a series of contextual embeddings $\{z_1, ..., z_N\}$ using a pre-trained BERT model followed by a BiLSTM. The conditional dependencies between input token embeddings z_i, latent NE tags y_i, and weak labels $\{l_{i,j}{}_{j=1}^{m}\}$ are modeled as functions of linear products between labeler representation θ_j and token embeddings z_i as formulated in equation 8. The dependencies between latent tags y_{i-1}, y_i are represented as an undirected graph through a CRF.

bidirectional LSTM with hidden size 768 to obtain the contextual token embeddings with dimension 768. The inputs to the BiLSTM were contextual word embeddings from RoBERTa (Liu et al., 2019) on CoNLL2003, uncased BERT on Laptop Review, and BioBERT (Lee et al., 2019) on NCBI-Disease and BC5CDR. These variants of BERT (Devlin et al., 2019) were not fine-tuned during training. Additionally, the class embeddings have dimension 768.

Training Details DWS: The optimizer used was RMSProp with learning rate 0.001. The model parameters were chosen to be those that gave the best micro-entity F1 score on the validation set over 10 random restarts of 5 epochs each. The batch sizes used were 32 for LaptopReview, 128 for NCBI-Disease, 256 for CoNLL2003, and 512 for BC5CDR. We defined the batch size by the number of tokens with at least 1 weak vote. Additionally, the illegal transition penalty was -2 for LaptopReview and -10 for the remaining datasets. Lastly, we used 1 warm-up epoch on majority vote labels and a dropout probability of 0.1 in the BiLSTM. These hyperparameters were chosen through manual tuning on the validation set.

Training Details Fuzzy-LSTM-CRF: The optimizer used was RMSProp with learning rate 0.01. The model parameters were chosen to be those that gave the best micro-entity F1 score on the validation set over 5 random restarts of 10 epochs each since it converged more slowly than DWS and had less variance among the random restarts. The best batch sizes were found to be the same as DWS. Lastly, the dropout probability in the BiLSTM was 0.1. The learning rate was chosen from $\{0.0001, 0.0005, 0.001, 0.01, 0.02\}$, the batch size was chosen from $\{32, 64, 128, 256, 512, 1024\}$, and the dropout was chosen from $\{0, 0.1, 0.2, 0.4\}$.

Training Details of Other Comparing Models: We used the same priors for the HMM1 model as were used in Lison et al. (2020) on CoNLL2003. For the remaining datasets the priors were tuned the on the validation sets. Additionally, we tried running the HMM from Safranchik et al. (2020) on CoNLL2003 but ran out of memory when using 64GB of RAM even when reducing the batch size to 1 and using unpositioned classes rather than BILU positioned classes (reduces number of classes by a factor of 4).

To improve the performance of *majority vote* and *unweighted vote* on CoNLL2003, which has many very noisy labels, we predicted 'No Entity' with probability 1 when there were less than T votes which was a tuned hyperparameter. The best value of T was found to be 5 for both methods and was chosen from $\{1, 5, 10\}$.

A.4 Discriminator Model Architecture / Training Details

The architecture of the discriminator was the same as used in Safranchik et al. (2020) which consisted of a 2 layer BiLSTM with 200 dimensional embeddings along with a CRF as the output layer. The model used both word embeddings from a variant of BERT and character embeddings from a CNN as input. Additionally, we used their noise aware loss function.

Explaining NLP Models via Minimal Contrastive Editing (MICE)

Alexis Ross[†] **Ana Marasović**[†◇] **Matthew E. Peters**[†]

[†]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[◇]Paul G. Allen School of Computer Science and Engineering, University of Washington
{alexisr,anam,matthewp}@allenai.org

Abstract

Humans have been shown to give contrastive explanations, which explain why an observed event happened *rather than* some other counterfactual event (the *contrast case*). Despite the influential role that contrastivity plays in how humans explain, this property is largely missing from current methods for explaining NLP models. We present MINIMAL CONTRASTIVE EDITING (MICE), a method for producing contrastive explanations of model predictions in the form of edits to inputs that change model outputs to the contrast case. Our experiments across three tasks—binary sentiment classification, topic classification, and multiple-choice question answering—show that MICE is able to produce edits that are not only contrastive, but also *minimal* and *fluent*, consistent with human contrastive edits. We demonstrate how MICE edits can be used for two use cases in NLP system development—debugging incorrect model outputs and uncovering dataset artifacts—and thereby illustrate that producing contrastive explanations is a promising research direction for model interpretability.

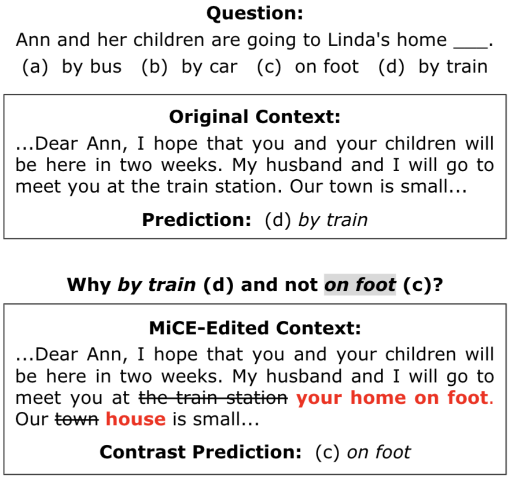

Question:
Ann and her children are going to Linda's home ___.
(a) by bus (b) by car (c) on foot (d) by train

Original Context:
...Dear Ann, I hope that you and your children will be here in two weeks. My husband and I will go to meet you at the train station. Our town is small...
Prediction: (d) *by train*

Why *by train* (d) and not *on foot* (c)?

MiCE-Edited Context:
...Dear Ann, I hope that you and your children will be here in two weeks. My husband and I will go to meet you at ~~the train station~~ **your home on foot**. Our ~~town~~ **house** is small...
Contrast Prediction: (c) *on foot*

Figure 1: An example MICE edit for a multiple-choice question from the RACE dataset. MICE generates contrastive explanations in the form of edits to inputs that change model predictions to target (contrast) predictions. The edit (bolded in red) is minimal and fluent, and it changes the model's prediction from "by train" to the contrast prediction "on foot" (highlighted in gray).

1 Introduction

Cognitive science and philosophy research has shown that human explanations are **contrastive** (Miller, 2019): People explain why an observed event happened rather than some counterfactual event called the *contrast case*. This contrast case plays a key role in modulating what explanations are given. Consider Figure 1. When we seek an explanation of the model's prediction "by train," we seek it not in absolute terms, but in contrast to another possible prediction (i.e. "on foot"). Additionally, we tailor our explanation to this contrast case. For instance, we might explain why the prediction is "by train" and not "on foot" by saying that the writer discusses meeting Ann at the train station instead of at Ann's home on foot; such information is captured by the edit (bolded red) that results in the new model prediction "on foot." For a different contrast prediction, such as "by car," we would provide a different explanation. In this work, we propose to give contrastive explanations of model predictions in the form of targeted minimal edits, as shown in Figure 1, that cause the model to change its original prediction to the contrast prediction.

Given the key role that contrastivity plays in human explanations, making model explanations contrastive could make them more user-centered and thus more useful for their intended purposes, such as debugging and exposing dataset biases (Ribera and Lapedriza, 2019)—purposes which require that *humans* work with explanations (Alvarez-Melis et al., 2019). However, many currently popular instance-based explanation methods produce

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3840–3852
August 1–6, 2021.

highlights—segments of input that support a prediction (Zaidan et al., 2007; Lei et al., 2016; Chang et al., 2019; Bastings et al., 2019; Yu et al., 2019; DeYoung et al., 2020; Jain et al., 2020; Belinkov and Glass, 2019) that can be derived through gradients (Simonyan et al., 2014; Smilkov et al., 2017; Sundararajan et al., 2017), approximations with simpler models (Ribeiro et al., 2016), or attention (Wiegreffe and Pinter, 2019; Sun and Marasović, 2021). These methods are not contrastive, as they leave the contrast case undetermined; they do not tell us what would have to be different for a model to have predicted a particular contrast label.[1]

As an alternative approach to NLP model explanation, we introduce **MINIMAL CONTRASTIVE EDITING (MICE)**—a two-stage approach to generating contrastive explanations in the form of targeted minimal edits (as shown in Figure 1). Given an input, a fixed PREDICTOR model, and a contrast prediction, MICE generates edits to the input that change the PREDICTOR's output from the original prediction to the contrast prediction. We formally define our edits and describe our approach in §2.

We design MICE to produce edits with properties motivated by human contrastive explanations. First, we desire edits to be **minimal**, altering only small portions of input, a property which has been argued to make explanations more intelligible (Alvarez-Melis et al., 2019; Miller, 2019). Second, MICE edits should be **fluent**, resulting in text natural for the domain and ensuring that any changes in model predictions are not driven by inputs falling out of distribution of naturally occurring text. Our experiments (§3) on three English-language datasets, IMDB, NEWSGROUPS, and RACE, validate that MICE edits are indeed contrastive, minimal, and fluent.

We also analyze the quality of MICE edits (§4) and show how they may be used for two use cases in NLP system development. First, we show that MICE edits are comparable in size and fluency to human edits on the IMDB dataset. Next, we illustrate how MICE edits can facilitate debugging individual model predictions. Finally, we show how MICE edits can be used to uncover dataset artifacts learned by a powerful PREDICTOR model.[2]

2 MICE: Minimal Contrastive Editing

This section describes our proposed method, MINIMAL CONTRASTIVE EDITING, or MICE, for explaining NLP models with contrastive edits.

2.1 MICE Edits as Contrastive Explanations

Contrastive explanations are answers to questions of the form *Why p and not q?* They explain why the observed event p happened instead of another event q, called the *contrast case*.[3] A long line of research in the cognitive sciences and philosophy has found that human explanations are contrastive (Van Fraassen, 1980; Lipton, 1990; Miller, 2019). Human contrastive explanations have several hallmark characteristics. First, they cite *contrastive features*: features that result in the contrast case when they are changed in a particular way (Chin-Parker and Cantelon, 2017). Second, they are minimal in the sense that they rarely cite the entire causal chain of a particular event, but select just a few relevant causes (Hilton, 2017). In this work, we argue that a minimal edit to a model input that causes the model output to change to the contrast case has both these properties and can function as an effective contrastive explanation. We first give an illustration of contrastive explanations humans might give and then show how minimal contrastive edits offer analogous contrastive information.

As an example, suppose we want to explain why the answer to the question "*Q*: Where can you find a clean pillow case that is not in use?" is "*A*: the drawer."[4] If someone asks why the answer is not "*C1*: on the bed," we might explain: "*E1*: Because only the drawer stores pillow cases that are not in use." However, *E1* would *not* be an explanation of why the answer is not "*C2*: in the laundry hamper," since both drawers and laundry hampers store pillow cases that are not in use. For contrast case *C2*, we might instead explain: "*E2*: Because only laundry hampers store pillow cases that are not clean." We cite different parts of the original question depending on the contrast case.

In this work, we propose to offer contrastive explanations in the form of minimal edits that result in the contrast case as model output. Such edits are effective contrastive explanations because, by construction, they highlight contrastive features. For

---

[1] Free-text rationales (Narang et al., 2020) can be contrastive if human justifications are collected by asking "why... instead of..." which is not the case with current benchmarks (Camburu et al., 2018; Rajani et al., 2019; Zellers et al., 2019).

[2] Our code and trained EDITOR models are publicly available at `https://github.com/allenai/mice`.

[3] Related work also calls it the *foil* (Miller, 2019).

[4] Inspired by an example in Talmor et al. (2019): Question: "Where would you store a pillow case that is not in use?" Choices: "drawer, kitchen cupboard, bedding store, england."

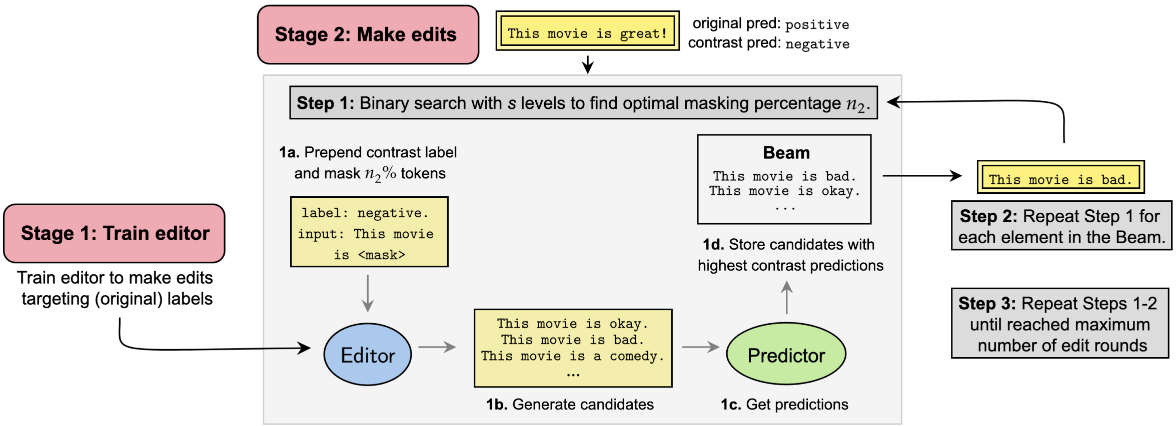

Figure 2: An overview of MiCE, our two-stage approach to generating edits. In Stage 1 (§2.3), we train the Editor to make edits targeting specific predictions from the Predictor. In Stage 2 (§2.4), we make contrastive edits with the Editor model from Stage 1 such that the Predictor changes its output to the contrast prediction.

example, a contrastive edit of the original question for contrast case *C1* would be: “Where can you find a clean pillow case that is ~~**not**~~ in use?”; the information provided by this edit—that it is whether or not the pillow case is in use that determines whether the answer is “the drawer” or “on the bed”—is analogous to the information provided by *E1*. Similarly, a contrastive edit for contrast case *C2* that changed the question to “Where can you find a ~~**clean**~~ **dirty** pillow case that is not in use?” provides analogous information to *E2*.

2.2 Overview of MiCE

We define a contrastive edit to be a modification of an input instance that causes a Predictor model (whose behavior is being explained) to change its output from its original prediction for the unedited input to a given target (contrast) prediction. Formally, for textual inputs, given a fixed Predictor f, input $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ of N tokens, original prediction $f(\boldsymbol{x}) = y_p$ and contrast prediction $y_c \neq y_p$, a contrastive edit is a mapping $e : (x_1, ..., x_N) \rightarrow (x'_1, ..., x'_M)$ such that $f(e(\boldsymbol{x})) = y_c$.

We propose MiCE, a two-stage approach to generating contrastive edits, illustrated in Figure 2. In Stage 1, we prepare a highly-contextualized Editor model to associate edits with given end-task labels (i.e., labels for the task of the Predictor) such that the contrast label y_c is not ignored in MiCE’s second stage. Intuitively, we do this by masking the spans of text that are “important” for the given target label (as measured by the Predictor’s gradients) and training our Editor to reconstruct these spans of text given the masked text and target label as input. In Stage 2 of MiCE, we generate contrastive edits $e(\boldsymbol{x})$ using the Editor model from Stage 1. Specifically, we generate candidate edits $e(\boldsymbol{x})$ by masking different percentages of $\boldsymbol{x}$ and giving masked inputs with prepended contrast label y_c to the Editor; we use binary search to find optimal masking percentages and beam search to keep track of candidate edits that result in the highest probability of the contrast labels $p(y_c|e(\boldsymbol{x}))$ given by the Predictor.

2.3 Stage 1: Fine-tuning the Editor

In Stage 1 of MiCE, we fine-tune the Editor to infill masked *spans* of text in a targeted manner. Specifically, we fine-tune a pretrained model to infill masked spans given masked text and a target end-task label as input. In this work, we use the Text-to-Text Transfer Transformer (T5) model (Raffel et al., 2020) as our pretrained Editor, but any model suitable for span infilling can in principle be the Editor in MiCE. The addition of the target label allows the highly-contextualized Editor to condition its predictions on both the masked context and the given target label such that the contrast label is not ignored in Stage 2. What to use as target labels during Stage 1 depends on who the end-users of MiCE are. The end-user could be: (1) a model developer who has access to the labeled data used to train the predictor, or (2) lay-users, domain experts, or other developers without access to the labeled data. In the former case, we could use the gold label as targets, and in the latter case, we could use the labels predicted by Predictor. Therefore, during fine-tuning, we experiment with using both gold labels and original predictions

y_p of our PREDICTOR model as target labels. To provide target labels, we prepend them to inputs to the EDITOR. For more information about how these inputs are formatted, see Appendix B. Results in Table 2 show that fine-tuning with target labels results in better edits than fine-tuning without them.

The above procedure allows our EDITOR to condition its infilled spans on both the context and the target label. But this still leaves open the question of where to mask our text. Intuitively, we want to mask the tokens that contribute most to the PREDICTOR's predictions, since these are the tokens that are most strongly associated with the target label. We propose to use gradient attribution (Simonyan et al., 2014) to choose tokens to mask. For each instance, we take the gradient of the predicted logit for the target label with respect to the embedding layers of f and take the ℓ_1 norm across the embedding dimension. We then mask the $n_1\%$ of tokens with the highest gradient norms. We replace consecutive tokens (i.e., spans) with sentinel tokens, following Raffel et al. (2020). Results in Table 1 show that gradient-based masking outperforms random masking.

2.4 Stage 2: Making Edits with the EDITOR

In the second stage of our approach, we use our fine-tuned EDITOR to make edits using beam search (Reddy, 1977). In each round of edits, we mask consecutive spans of $n_2\%$ of tokens in the original input, prepend the contrast prediction to the masked input, and feed the resulting masked instance to the EDITOR; the EDITOR then generates m edits. The masking procedure during this stage is gradient-based as in Stage 1.

In one round of edits, we conduct a binary search with s levels over values of n_2 between values $n_2 = 0\%$ to $n_2 = 55\%$ to efficiently find a value of n_2 that is large enough to result in the contrast prediction while also modifying only minimal parts of the input. After each round of edits, we get f's predictions on the edited inputs, order them by contrast prediction probabilities, and update the beam to store the top b edited instances. As soon as an edit $e^* = e(\boldsymbol{t})$ is found that results in the contrast prediction, i.e., $f(e^*) = y_c$, we stop the search procedure and return this edit. For generation, we use a combination of top-*k* (Fan et al., 2018) and top-*p* (nucleus) sampling (Holtzman et al., 2020).[5]

3 Evaluation

This section presents empirical findings that MICE produces minimal and fluent contrastive edits.

3.1 Experimental Setup

Tasks We evaluate MICE on three English-language datasets: IMDB, a binary sentiment classification task (Maas et al., 2011), a 6-class version of the 20 NEWSGROUPS topic classification task (Lang, 1995), and RACE, a multiple choice question-answering task (Lai et al., 2017).[6]

PREDICTORS MICE can be used to make contrastive edits for any differentiable PREDICTOR model, i.e., any end-to-end neural model. In this paper, for each task, we train a PREDICTOR model f built on ROBERTA-LARGE (Liu et al., 2019), and fix it during evaluation. The test accuracies of our PREDICTORS are 95.9%, 85.3% and 84% for IMDB, NEWSGROUPS, and RACE, respectively. For training details, see Appendix A.1.

EDITORS Our EDITORS build on the base version of T5. For fine-tuning our EDITORS (Stage 1), we use the original training data used to train PREDICTORS. We randomly split the data, 75%/25% for fine-tuning/validation and fine-tune until the validation loss stops decreasing (for a max of 10 epochs) with $n_1\%$ of tokens masked, where n_1 is a randomly chosen value in $[20, 55]$. For more details, see Appendix A.2. In Stage 2, for each instance, we set the label with the second highest predicted probability as the contrast prediction. We set beam width $b = 3$, consider $s = 4$ search levels during binary search over n_2 in each edit round, and run our search for a max of 3 edit rounds. For each n_2, we sample $m = 15$ generations from our fine-tuned EDITORS with $p = 0.95$, $k = 30$.[7]

Metrics We evaluate MICE on the test sets of the three datasets. The RACE and NEWSGROUPS test sets contain 4,934 and 7,307 instances, respectively.[8] For IMDB, we randomly sample 5K of the

---

[5] We use this combination because we observed in preliminary experiments that it led to good results.

[6] We create this 6-class version by mapping the 20 existing subcategories to their respective larger categories—i.e. "talk.politics.guns" and "talk.religion.misc" → "talk." We do this in order to make the label space smaller. The resulting classes are: alt, comp, misc, rec, sci, and talk.

[7] We tune these hyperparameters on a 50-instance subset of the IMDB validation set prior to evaluation. We note that for larger values of n_2, the generations produced by the T5 EDITORS sometimes degenerate; see Appendix C for details.

[8] For the NEWSGROUPS test set, there are 7,307 instances remaining after filtering out empty strings.

MICE VARIANT	IMDB			NEWSGROUPS			RACE		
	↑ **Flip Rate**	↓ **Minim.**	≈ 1 **Fluen.**	↑ **Flip Rate**	↓ **Minim.**	≈ 1 **Fluen.**	↑ **Flip Rate**	↓ **Minim.**	≈ 1 **Fluen.**
***PRED + GRAD**	**1.000**	**0.173**	**0.981**	**0.992**	**0.261**	0.968	0.915	**0.331**	**0.981**
***GOLD + GRAD**	**1.000**	0.185	0.979	**0.992**	0.271	0.966	**0.945**	0.335	0.979
PRED + RAND	**1.000**	0.257	0.958	0.968	0.378	0.928	0.799	0.440	0.953
GOLD + RAND	**1.000**	0.302	0.952	0.965	0.370	0.929	0.801	0.440	0.955
NO-FINETUNE	0.995	0.360	0.960	0.941	0.418	0.938	–	–	–

Table 1: Efficacy of the MICE procedure. We evaluate MICE edits on three metrics (described in §3.1): flip rate, minimality, and fluency. We report mean values for minimality and fluency. * marks full MICE variants; others explore ablations. For each property (i.e., column), the best value across MICE variants is bolded. We experiment with PREDICTOR's predictions (PRED) and gold labels (GOLD) as target labels during Stage 1. Across datasets, our GRAD MICE procedure achieves a high flip rate with small and fluent edits.

25K instances in the test set for evaluation because of the computational demands of evaluation.[9]

For each dataset, we measure the following three properties: (1) **flip rate:** the proportion of instances for which an edit results in the contrast label; (2) **minimality:** the "size" of the edit as measured by the word-level Levenshtein distance between the original and edited input, which is the minimum number of deletions, insertions, or substitutions required to transform one into the other. We report a normalized version of this metric with a range from 0 to 1—the Levenshtein distance divided by the number of words in the original input; (3) **fluency:** a measure of how similarly distributed the edited output is to the original data. We evaluate fluency by comparing masked language modeling loss on both the original and edited inputs using a pretrained model. Specifically, given the original N-length sequence, we create N copies, each with a different token replaced by a mask token, following Salazar et al. (2020). We then take a pretrained T5-BASE model and compute the average loss across these N copies. We compute this loss value for both the original input and edited input and report their *ratio*—i.e., edited / original. We aim for a value of 1.0, which indicates equivalent losses for the original and edited texts. When MICE finds multiple edits, we report metrics for the edit with the smallest value for minimality.

3.2 Results

Results are shown in Table 1. Our proposed GRAD MICE procedure (upper part of Table 1) achieves a high flip rate across all three tasks. This is the outcome regardless of whether predicted target labels (first row, 91.5–100% flip rate) or gold target labels (second row, 94.5–100% flip rate) are used for fine-tuning in Stage 1. We observe a slight improvement from using the gold labels for the RACE PREDICTOR, which may be explained by the fact that it is less accurate (with a training accuracy of 89.9%) than the IMDB and NEWSGROUPS classifiers.

MICE achieves a high flip-rate while its edits remain small and result in fluent text. In particular, MICE on average changes 17.3–33.1% of the original tokens when predicted labels are used in Stage 1 and 18.5–33.5% with gold labels. Fluency is close to 1.0 indicating no notable change in mask language modeling loss after the edit—i.e., edits fall in distribution of the original data. We achieve the best results across metrics on the IMDB dataset, as expected since IMDB is a binary classification task with a small label space. These results demonstrate that MICE presents a promising research direction for the generation of contrastive explanations; however, there is still room for improvement, especially for more challenging tasks such as RACE.

In the rest of this section, we provide results from several ablation experiments.

Fine-tuning vs. No Fine-tuning We investigate the effect of fine-tuning (Stage 1) with a baseline that skips Stage 1 altogether. For this NO-FINETUNE baseline variant of MICE, we use the vanilla pretrained T5-BASE as our EDITOR. As shown in Table 1, the NO-FINETUNE variant underperforms all other (two-stage) variants of MICE for the IMDB and NEWSGROUPS datasets.[10] Fine-

---

[9] A single contrastive edit is expensive and takes an average of ≈ 15 seconds per IMDB instance (≈ 230 tokens). Calculating the fluency metric adds an additional average of ≈ 16.5 seconds per IMDB instance. For more details, see Section 5.

[10] We leave RACE out from our evaluation with the NO-FINETUNE baseline because we observe that the pretrained

IMDB Condition Stage 1	Stage 2	↑ Flip Rate	↓ Minim.	≈ 1 Fluen.
No Label	No Label	0.994	0.369	0.966
No Label	Label	0.997	0.362	0.967
Label	No Label	0.999	0.327	0.968
Label	Label	**1.000**	**0.173**	**0.981**

Table 2: Effect of using target end-task labels during the two stages of PRED+GRAD MICE on the IMDB dataset. When end-task labels are provided, they are original PREDICTOR labels during Stage 1 and contrast labels during Stage 2. The best values for each property (column) are bolded. Using end-task labels during both Stage 1 (EDITOR fine-tuning) and Stage 2 (making edits) of MICE outperforms all other conditions.

tuning particularly improves the minimality of edits, while leaving the flip rate high. We hypothesize that this effect is due to the effectiveness of Stage 2 of MICE at finding contrastive edits: Because we iteratively generate many candidate edits using beam search, we are likely to find a prediction-flipping edit. Fine-tuning allows us to find such an edit at a lower masking percentage.

Gradient vs. Random Masking We study the impact of using gradient-based masking in Stage 1 of the MICE procedure with a RAND variant, which masks spans of randomly chosen tokens. As shown in the middle part of Table 1, gradient-based masking outperforms random masking when using both predicted and gold labels across all three tasks and metrics, suggesting that the gradient-based attribution used to mask text during Stage 1 of MICE is an important part of the procedure. The differences are especially notable for RACE, which is the most challenging task according to our metrics.

Targeted vs. Un-targeted Infilling We investigate the effect of using target labels in both stages of MICE by experimenting with removing target labels during Stage 1 (EDITOR fine-tuning) and Stage 2 (making edits). As shown in Table 2, we observe that giving target labels to our EDITORS during both stages of MICE improves edit quality. Fine-tuning EDITORS without labels in Stage 1 ("No Label") leads to worse flip rate, minimality, and fluency than does fine-tuning EDITORS with labels ("Label"). Minimality is particularly affected, and we hypothesize that using target end-task labels in both stages provides signal that allows the EDITOR in Stage 2 to generate prediction-flipping edits at lower masking percentages.

4 Analysis of Edits

In this section, we compare MICE edits with human contrastive edits. Then, we turn to a key motivation for this work: the potential for contrastive explanations to assist in NLP system development. We show how MICE edits can be used to debug incorrect predictions and uncover dataset artifacts.

4.1 Comparison with Human Edits

We ask whether the contrastive edits produced by MICE are minimal and fluent in a meaningful sense. In particular, we compare these two metrics for MICE edits and human contrastive edits. We work with the IMDB contrast set created by Gardner et al. (2020), which consists of original test inputs and human-edited inputs that cause a change in *true* label. We report metrics on the subset of this contrast set for which the human-edited inputs result in a change in model prediction for our IMDB PREDICTOR; this subset consists of 76 instances. The flip rate of MICE edits on this subset is 100%. The mean minimality values of human and MICE edits are $\mathbf{0.149}$ (human) and $\mathbf{0.179}$ (MICE), and the mean fluency values are $\mathbf{1.01}$ (human) and $\mathbf{0.949}$ (MICE). The similarity of these values suggests that MICE edits are comparable to human contrastive edits along these dimensions.

We also ask to what extent human edits overlap with MICE edits. For each input, we compute the overlap between the original tokens changed by humans and the original tokens edited by MICE. The mean number of overlapping tokens, normalized by the number of original tokens edited by humans, is $\mathbf{0.298}$. Thus, while there is some overlap between MICE and human contrastive edits, they generally change different parts of text.[11] This analysis suggests that there may exist multiple informative contrastive edits for a single input. Future work can investigate and compare the different kinds of insight that can be obtained through human and model-driven contrastive edits.

4.2 Use Case 1: Debugging Incorrect Outputs

Here, we illustrate how MICE edits can be used to debug incorrect model outputs. Consider the RACE

T5 model does not generate text formatted as span infills; we hypothesize that this model has not been trained to generate infills for masked inputs formatted as multiple choice inputs.

[11] MICE edits explain PREDICTORS' behavior and therefore need not be similar to human edits, which are designed to change gold labels.

IMDB	**Original pred** y_p = positive **Contrast pred** y_c = negative An interesting pairing of stories, this little flick manages to bring together seemingly different characters and story lines all in the backdrop of WWII and succeeds in tying them together without losing the audience. I was impressed by the depth portrayed by the different characters and also by how much I really felt I understood them and their motivations, even though the time spent on the development of each character was very limited. The outstanding acting abilities of the individuals involved with this picture are easily noted. A fun, stylized movie with a slew of comic moments and a bunch more head shaking events. ~~7/10~~ **4/10**
RACE	**Question:** Mark went up in George's plane ____________. (a) twice (b) only once (c) several times (d) once or twice. **Original pred** $y_p = (a)$ twice **Contrast pred** $y_c = (b)$ only once When George was thirty-five, he bought a small plane and learned to fly it. He soon became very good and made his plane do all kinds of tricks. George had a friend, whose name was Mark. One day George offered to take Mark up in his plane. Mark thought, "I've traveled in a big plane several times, but I've never been in a small one, so I'll go." They went up, and George flew around for half an hour and did all kinds of tricks in the air. When they came down again, Mark was glad to be back safely, and he said to his friend in a shaking voice, "Well, George, thank you very much for those two ~~trips~~ **tricks** in your plane." George was very surprised and said, "Two ~~trips?~~ **tricks."** ~~Yes,~~ **That's** my first ~~and my last~~ **time, George**." ~~answered~~ **said** Mark.

Table 3: Examples of edits produced by MiCE. Insertions are bolded in red. Deletions are struck through. y_p is the PREDICTOR's original prediction, and y_c the contrast prediction. True labels for original inputs are underlined.

input in Table 3, for which the RACE PREDICTOR gives an incorrect prediction. In this case, a model developer may want to understand why the model got the answer wrong. This setting naturally brings rise to a contrastive question, i.e., *Why did the model predict the wrong choice ("twice") instead of the correct one ("only once")?*

The MiCE edit shown offers insight into this question: Firstly, it highlights which part of the paragraph has an influence on the model prediction—the last few sentences. Secondly, it reveals that a source of confusion is Mark's joke about having traveled in George's plane twice, as changing Mark's dialogue from talking about a "first and...last" trip to a single trip results in a correct model prediction.

MiCE edits can also be used to debug model *capabilities* by offering hypotheses about "bugs" present in models: For instance, the edit in Table 3 might prompt a developer to investigate whether this PREDICTOR lacks non-literal language understanding capabilities. In the next section, we show how insight from individual MiCE edits can be used to uncover a bug in the form of a dataset-level artifact learned by a model. In Appendix D, we further analyze the debugging utility of MiCE edits with a PREDICTOR *designed* to contain a bug.

4.3 Use Case 2: Uncovering Dataset Artifacts

Manual inspection of some edits for IMDB suggests that the IMDB PREDICTOR has learned to rely heavily on numerical ratings. For instance, in the IMDB example in Table 3, the MiCE edit results in a negative prediction from the PREDICTOR even though the edited text is overwhelmingly positive. We test this hypothesis by investigating whether numerical tokens are more likely to be edited by MiCE.

y_c = ***positive***		y_c = ***negative***	
Removed	**Inserted**	**Removed**	**Inserted**
4/10	excellent	10/10	awful
ridiculous	enjoy	8/10	disappointed
horrible	amazing	7/10	1
4	entertaining	9	4
predictable	10	enjoyable	annoying

Table 4: Top 5 IMDB tokens edited by MiCE at a higher rate than expected given their original frequency (§4.3). Results are separated by contrast predictions.

We analyze the edits produced by MiCE (GOLD + GRAD) described in §3.1. We limit our analysis to a subset of the 5K instances for which the edit produced by MiCE has a minimality value of ≤0.05, as we are interested in finding simple artifacts driving the predictions of the IMDB PREDICTOR; this subset has 902 instances. We compute three metrics for each unique token, i.e., type t:

$$\begin{aligned} p(t) &= \#\_\text{occurrences}(t) / \#\_\text{all\_tokens}, \\ p_r(t) &= \#\_\text{removals}(t) / \#\_\text{all\_removals}, \\ p_i(t) &= \#\_\text{insertions}(t) / \#\_\text{all\_insertions}, \end{aligned}$$

and report the tokens with the highest values for the ratios $p_r(t)/p(t)$ and $p_i(t)/p(t)$. Intuitively, these tokens are removed/inserted at a higher rate than expected given the frequency with which they appear in the original IMDB inputs. We exclude tokens that occur $<$10 times from our analysis.

Results from this analysis are shown in Table 4. In line with our hypothesis, we observe a bias towards removing low numerical ratings and inserting high ratings when the contrast prediction y_c is positive, and vice versa when y_c is negative. In other words, in the presence of a numerical score, the PREDICTOR may ignore the content of the review and base its prediction solely on the score (as in the IMDB example in Table 3).

5 Discussion

In this section, we reflect on MICE's shortcomings. Foremost, MICE is computationally expensive. Stage 1 requires fine-tuning a large pretrained generation model as the EDITOR. More significantly, Stage 2 requires multiple rounds of forward and backward passes to find a minimal edit: Each edit round in Stage 2 requires $b \times s \times m$ decoded sequences with the EDITOR, as well as $b \times s \times m$ forward passes and b backward passes with the PREDICTOR (with $b = 1$ the first edit round), where b is the beam width, s is the number of search levels in binary search over the masking percentages, and m is the number of generations sampled for each masking percentage. Our experiments required 180 forward passes, 180 decoded sequences, and 3 backward passes for edit rounds after the first.

While efficient search for targeted edits is an open challenge in other fields of machine learning (Russell, 2019; Dandl et al., 2020), this problem is even more challenging for language data, as the space of possible perturbations is much larger than for tabular data. An important future direction is to develop more efficient methods of finding edits.

This shortcoming prevents us from finding edits that are minimal in a precise sense. In particular, we may be interested in a constrained notion of minimality that defines an edit $e(\boldsymbol{x})$ as minimal if there exists no subset of $e(\boldsymbol{x})$ that results in the contrast prediction. Future work might consider creating methods to produce edits with this property.

6 Related Work

The problem of generating minimal contrastive edits, also called counterfactual explanations (Wachter et al., 2017),[12] has previously been explored for tabular data (Karimi et al., 2020) and images (Hendricks et al., 2018; Goyal et al., 2019; Looveren and Klaise, 2019) but less for language. Recent work explores the use of minimal edits changing true labels for evaluation (Gardner et al., 2020) and data augmentation (Kaushik et al., 2020; Teney et al., 2020), whereas we focus on minimal edits changing model predictions for *explanation*.

Contrastive Explanations within NLP There exist limited methods for automatically generating contrastive explanations of NLP models. Jacovi and Goldberg (2020) define contrastive highlights, which are determined by the inclusion of contrastive features; in contrast, our contrastive edits specify *how* to edit (vs. whether to include) features and can insert new text.[13] Li et al. (2020a) generate counterfactuals using linguistically-informed transformations (LIT), and Yang et al. (2020) generate counterfactuals for binary financial text classification using grammatically plausible single-word edits (REP-SCD). Because both methods rely on manually curated, task-specific rules, they cannot be easily extended to tasks without predefined label spaces, such as RACE.[14] Most recently, Jacovi et al. (2021) propose a method for producing contrastive explanations in the form of latent representations; in contrast, MICE edits are made at the textual level and are therefore more interpretable.

This work also has ties to the literature on causal explanation (Pearl, 2009). Recent work within NLP derives causal explanations of models through counterfactual interventions (Feder et al., 2021; Vig et al., 2020). The focus of our work is the largely unexplored task of creating targeted interventions for language data; however, the question of how to derive causal relationships from such interventions remains an interesting direction for future work.

Counterfactuals Beyond Explanations Concurrent work by Madaan et al. (2021) applies con-

---

[12] Formally, methods for producing targeted counterfactual explanations solve the same task as MICE. However, not all contrastive explanations are counterfactual explanations; contrastive explanations can take forms beyond contrastive edits, such as free-text rationales (Liang et al., 2020) or highlights (Jacovi and Goldberg, 2020). In this paper, we choose to refer to MICE edits as "contrastive" rather than "counterfactual" because we seek to argue for the utility of *contrastive explanations* of model predictions more broadly; we present MICE as one method for producing contrastive explanations of a particular form and hope future work will explore different forms of contrastive explanations.

[13] See Appendix D for a longer discussion about the advantage of inserting new text in explanations, which MICE edits can do but methods that attribute feature importance (i.e. highlights) cannot.

[14] LIT relies on hand-crafted transformation for NLI tasks based on linguistic knowledge, and REP-SCD makes antonym-based edits using manually curated, domain-specific lexicons for each label.

trolled text generation methods to generate targeted counterfactuals and explores their use as test cases and augmented examples in the context of classification. Another concurrent work by Wu et al. (2021) presents POLYJUICE, a general-purpose, untargeted counterfactual generator. Very recent work by Sha et al. (2021), introduced after the submission of MICE, proposes a method for targeted contrastive editing for Q&A that selects answer-related tokens, masks them, and generates new tokens. Our work differs from these works in our novel framework for efficiently finding *minimal* edits (MICE Stage 2) and our use of edits as explanations.

Connection to Adversarial Examples Adversarial examples are minimally edited inputs that cause models to incorrectly change their predictions despite no change in true label (Jia and Liang, 2017; Ebrahimi et al., 2018; Pal and Tople, 2020). Recent methods for generating adversarial examples also preserve fluency (Zhang et al., 2019; Li et al., 2020b; Song et al., 2020)[15]; however, adversarial examples are designed to find *erroneous* change in model outputs; contrastive edits place no such constraint on model correctness. Thus, current approaches to generating adversarial examples, which can exploit semantics-preserving operations (Ribeiro et al., 2018) such as paraphrasing (Iyyer et al., 2018) or word replacement (Alzantot et al., 2018; Ren et al., 2019; Garg and Ramakrishnan, 2020), cannot be used to generate contrastive edits.

Connection to Style Transfer The goal of style transfer is to generate minimal edits to inputs to result in a target style (sentiment, formality, etc.) (Fu et al., 2018; Li et al., 2018; Goyal et al., 2020). Most existing approaches train an encoder to learn style-agnostic latent representation of inputs and train attribute-specific decoders to generate text reflecting the content of inputs but exhibiting a different target attribute (Fu et al., 2018; Li et al., 2018; Goyal et al., 2020). Recent works by Wu et al. (2019) and Malmi et al. (2020) adopt two-stage approaches that first identify where to make edits and then make them using pretrained language models. Such approaches can only be applied to generate contrastive edits for classification tasks with well-defined "styles," which exclude more complex tasks such as question answering.

7 Conclusion

We argue that contrastive edits, which change the output of a PREDICTOR to a given contrast prediction, are effective explanations of neural NLP models. We propose MINIMAL CONTRASTIVE EDITING (MICE), a method for generating such edits. We introduce evaluation criteria for contrastive edits that are motivated by human contrastive explanations—minimality and fluency—and show that MICE edits for the IMDB, NEWSGROUPS, and RACE datasets are contrastive, fluent, and minimal. Through qualitative analysis of MICE edits, we show that they have utility for robust and reliable NLP system development.

8 Broader Impact Statement

MICE is intended to aid the interpretation of NLP models. As a model-agnostic explanation method, it has the potential to impact NLP system development across a wide range of models and tasks. In particular, MICE edits can benefit NLP model developers in facilitating debugging and exposing dataset artifacts, as discussed in §4. As a consequence, they can also benefit downstream users of NLP models by facilitating access to less biased and more robust systems.

While the focus of our work is on interpreting NLP models, there are potential misuses of MICE that involve other applications. Firstly, malicious actors might employ MICE to generate adversarial examples; for instance, they may aim to generate hate speech that is minimally edited such that it fools a toxic language classifier. Secondly, naively applying MICE for data augmentation could plausibly lead to less robust and more biased models: Because MICE edits are intended to expose issues in models, straightforwardly using them as additional training examples could reinforce existing artifacts and biases present in data. To mitigate this risk, we encourage researchers exploring data augmentation to carefully think about how to select and label edited instances.

We also encourage researchers to develop more efficient methods of generating minimal contrastive edits. As discussed in §5, a limitation of MICE is its computational demand. Therefore, we recommend that future work focus on creating methods that require less compute.

---

[15] Song et al. (2020) propose a method to produce fluent *semantic collisions*, which they call the "inverse" of adversarial examples.

References

David Alvarez-Melis, Hal Daumé III, Jennifer Wortman Vaughan, and Hanna Wallach. 2019. Weight of evidence as a basis for human-oriented explanations. In *Workshop on Human-Centric Machine Learning at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31, pages 9539–9549. Curran Associates, Inc.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi Jaakkola. 2019. A game theoretic approach to class-wise selective rationalization. In *Advances in Neural Information Processing Systems*, volume 32, pages 10055–10065. Curran Associates, Inc.

Seth Chin-Parker and Julie A Cantelon. 2017. Contrastive constraints guide explanation-based category learning. *Cognitive Science*, 41 6:1645–1655.

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-objective counterfactual explanations. In *Parallel Problem Solving from Nature – PPSN XVI*, pages 448–469, Cham. Springer International Publishing.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. 2021. CausaLM: Causal Model Explanation Through Counterfactual Language Models. *Computational Linguistics*, pages 1–54.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI Conference on Artificial Intelligence*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181. Association for Computational Linguistics.

Navita Goyal, Balaji Vasan Srinivasan, N. Anandhavelu, and Abhilasha Sancheti. 2020. Multi-dimensional style transfer for partially annotated data using language models as discriminators. *ArXiv*, arXiv:2010.11578.

Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2376–2384, Long Beach, California, USA. PMLR.

Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. 2018. Grounding visual explanations. In *Computer Vision – ECCV 2018*, pages 269–286, Cham. Springer International Publishing.

Denis Hilton. 2017. *Social Attribution and Explanation*. Oxford University Press.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Alon Jacovi and Y. Goldberg. 2020. Aligning faithful interpretations with their social attribution. *ArXiv*, arXiv:2006.01067.

Alon Jacovi, Swabha Swayamdipta, Shauli Ravfogel, Yanai Elazar, Yejin Choi, and Yoav Goldberg. 2021. Contrastive explanations for model interpretability. ArXiv:2103.01378.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Amir-Hossein Karimi, G. Barthe, B. Balle, and I. Valera. 2020. Model-agnostic counterfactual explanations for consequential decisions. Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS).

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Chuanrong Li, Lin Shengshuo, Zeyu Liu, Xinyi Wu, Xuhui Zhou, and Shane Steinert-Threlkeld. 2020a. Linguistically-informed transformations (LIT): A method for automatically generating contrast sets. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 126–135, Online. Association for Computational Linguistics.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202. Association for Computational Linguistics.

Weixin Liang, James Zou, and Zhou Yu. 2020. ALICE: Active learning with contrastive natural language explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4380–4391, Online. Association for Computational Linguistics.

Peter Lipton. 1990. Contrastive explanation. *Royal Institute of Philosophy Supplement*, 27:247–266.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*, arXiv:1907.11692.

Arnaud Van Looveren and Janis Klaise. 2019. Interpretable counterfactual explanations guided by prototypes. *ArXiv*, arXiv:1907.02584.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. 2021. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Eric Malmi, Aliaksei Severyn, and Sascha Rothe. 2020. Unsupervised text style transfer with padded masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8671–8680. Association for Computational Linguistics.

Tim Miller. 2019. Explanation in Artificial Intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. 2020. WT5?! training text-to-text models to explain their predictions. arXiv:2004.14546.

B. Pal and S. Tople. 2020. To transfer or not to transfer: Misclassification attacks against transfer learned text classifiers. *ArXiv*, arXiv:2001.02438.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*, 2nd edition. Cambridge University Press, USA.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

D Raj Reddy. 1977. Speech understanding systems: A summary of results of the five-year research effort.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Mireia Ribera and Àgata Lapedriza. 2019. Can We Do Better Explanations? A Proposal of User-Centered Explainable AI. In *ACM IUI Workshop*.

Chris Russell. 2019. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712. Association for Computational Linguistics.

Lei Sha, Patrick Hohenecker, and Thomas Lukasiewicz. 2021. Controlling text edition by changing answers of specific questions. ArXiv:2105.11018.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*.

Jacob Sippy, Gagan Bansal, and Daniel S. Weld. 2020. Data staining: A method for comparing faithfulness of explainers. In *2020 ICML Workshop on Human Interpretability in Machine Learning (WHI 2020)*.

D. Smilkov, Nikhil Thorat, Been Kim, F. Viégas, and M. Wattenberg. 2017. Smoothgrad: removing noise by adding noise. In *ICML Workshop on Visualization for Deep Learning*.

Congzheng Song, Alexander Rush, and Vitaly Shmatikov. 2020. Adversarial semantic collisions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4198–4210, Online. Association for Computational Linguistics.

Kaiser Sun and Ana Marasović. 2021. Effective attention sheds light on interpretability. In *Findings of ACL*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 3319–3328. JMLR.org.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Damien Teney, Ehsan Abbasnejad, and A. V. D. Hengel. 2020. Learning what makes a difference from counterfactual examples and gradient supervision. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Bas C Van Fraassen. 1980. *The scientific image*. Oxford University Press.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

S. Wachter, Brent D. Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *European Economics: Microeconomics & Industrial Organization eJournal*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tongshuang Wu, Marco Túlio Ribeiro, J. Heer, and Daniel S. Weld. 2021. Polyjuice: Automated, general-purpose counterfactual generation. arXiv:2101.00288.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Mask and infill: Applying masked language model for sentiment transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5271–5277. International Joint Conferences on Artificial Intelligence Organization.

Linyi Yang, Eoin Kenny, Tin Lok James Ng, Yi Yang, Barry Smyth, and Ruihai Dong. 2020. Generating plausible counterfactual explanations for deep transformers in financial text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6150–6160, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi Jaakkola. 2019. Rethinking cooperative rationalization: Introspective extraction and complement control. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4094–4103, Hong Kong, China. Association for Computational Linguistics.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using “annotator rationales” to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267, Rochester, New York. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From Recognition to Cognition: Visual Commonsense Reasoning. In *CVPR*.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Differential Privacy for Text Analytics via Natural Text Sanitization[†]

Xiang Yue[1,‡], Minxin Du[2], Tianhao Wang[3], Yaliang Li[4], Huan Sun[1], and Sherman S. M. Chow[2]

[1]The Ohio State University
[2]The Chinese University of Hong Kong
[3]Carnegie Mellon University
[4]Alibaba Group
`{yue.149,sun.397}@osu.edu`,`{dm018,sherman}@ie.cuhk.edu.hk`
`tianhao@cmu.edu`, `yaliang.li@alibaba-inc.com`

Abstract

Texts convey sophisticated knowledge. However, texts also convey sensitive information. Despite the success of general-purpose language models and domain-specific mechanisms with differential privacy (DP), existing text sanitization mechanisms still provide low utility, as cursed by the high-dimensional text representation. The companion issue of utilizing sanitized texts for downstream analytics is also under-explored. This paper takes a direct approach to text sanitization. Our insight is to consider both sensitivity and similarity via our new local DP notion. The sanitized texts also contribute to our sanitization-aware pretraining and fine-tuning, enabling privacy-preserving natural language processing over the BERT language model with promising utility. Surprisingly, the high utility does not boost up the success rate of inference attacks.

1 Introduction

Natural language processing (NLP) requires a lot of training data, which can be sensitive. Naïve redaction approaches (*e.g.*, removing common personally identifiable information) is known to fail (Sweeney, 2015): innocuous-looking fields can be linked to other information sources for reidentification. The recent success of many language models (LMs) has motivated security researchers to devise advanced privacy attacks. Carlini et al. (2020b) recover texts from (a single document of) the *training data* via querying to an LM pretrained from it. Pan et al. (2020) and Song and Raghunathan (2020) target the text embedding, *e.g.*, revealing from an encoded *query* to an NLP service.

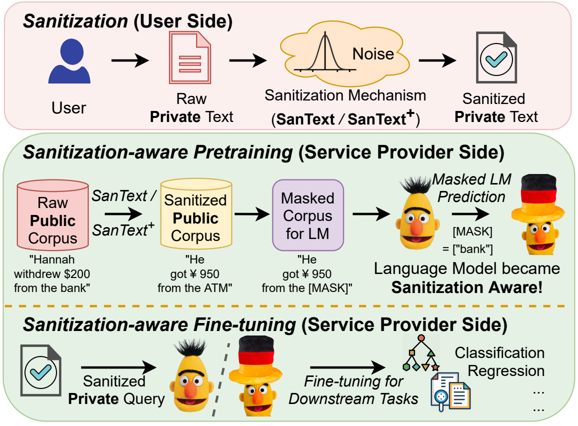

Figure 1: Workflow of our PPNLP pipeline, including the user-side sanitization and the service provider-side NLP modeling with pretraining/fine-tuning

Emerging NLP works focus on only specific document-level (statistical) features (Weggenmann and Kerschbaum, 2018) or producing private text representations (Xie et al., 2017; Coavoux et al., 2018; Elazar and Goldberg, 2018; Li et al., 2018) as initial solutions to the first issue above on training-data privacy. However, the learned representations are *not* human-readable, which makes *transparency* (*e.g.*, required by GDPR) questionable: an average user may not have the technical know-how to verify whether sensitive attributes have been removed or not. Moreover, consider the whole NLP pipeline, the learned representations often entail extra modeling or non-trivial changes to existing NLP models, which take dedicated engineering efforts.

1.1 Sanitizing Sensitive Texts, Naturally

With this state-of-affairs of the security and the NLP research, we deem it better to address privacy from the root, *i.e.*, directly producing sanitized text documents. Being the most native format, they

---

[†] Our code is available at `https://github.com/xiangyue9607/SanText`.

[‡] The first two authors contributed equally.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3853–3866
August 1–6, 2021.

incur minimal changes to existing NLP pipelines. Being human-readable, they provide transparency (to privacy-concerning training-data contributors) and explainability (*e.g.*, to linguists who might find the need for investigating how the training data contribute to a certain result). Moreover, it naturally extends the privacy protection to the inference phase. Users can apply our sanitization mechanism before sending queries (*e.g.*, medical history) to the NLP service provider (*e.g.*, diagnosis services).

Conceptually, we take a natural approach – we sanitize text documents into also (sanitized) text documents. This is in great contrast to the typical "post-processing" for injecting noises either to gradients in training (a deep neural network) (McMahan et al., 2018) or the "cursed" *high-dimensional* text representations (Lyu et al., 2020a,b; Feyisetan et al., 2020). It also leads to our $O(1)$ efficiency, freeing us from re-synthesizing the document word-by-word via nearest neighbor searches over the entire vocabulary space $\mathcal{V}$ (Feyisetan et al., 2020).

Technically, we aim for the *de facto* standard of local differential privacy (LDP) (Duchi et al., 2013) to sanitize the user data *locally*, based on which the service provider can build NLP models without touching any raw data. DP has been successful in many contexts, *e.g.*, location privacy and survey statistics (Andrés et al., 2013; Murakami and Kawamoto, 2019). However, DP text analytics appears to be a difficult pursuit (as discussed, also see Section 2), which probably explains why there are only a few works in DP-based text sanitization. In high-level terms, text is rich in semantics, differentiating it from other more structured data.

Our challenge here is to develop *efficient* and effective mechanisms that *preserve the utility* of the text data with *provable and quantifiable* privacy guarantees. Our insight is the formulation of a new LDP notion named *Utility-optimized Metric LDP (UMLDP)*. We attribute our success to the focus of UMLDP on protecting what matters (sensitive words) via "sacrificing" the privacy of non-sensitive (common) words. To achieve UMLDP, our mechanism directly samples noises on tokens.

Our result in this regard is already better than the state-of-the-art LDP solution producing sanitized documents (Feyisetan et al., 2020) – we got 28% gain in accuracy on the SST-2 dataset (Wang et al., 2019) on average at the same privacy level (*i.e.*, the same LDP parameter) while being much more efficient ($\sim$60× faster, precomputation included).

1.2 Privacy-Preserving NLP, Holistically

Text sanitization is essential but just one piece of the whole privacy-preserving NLP (PPNLP) pipeline. While most prior works in text privacy are motivated by producing useful data for some downstream tasks, the actual text analytics are hardly explored, not to say in the context of many recent general-purpose language models. As simple as it might seem, we start to see design choices that can be influential. Specifically, our challenge here is to adapt the currently dominating pretraining-fine-tuning paradigm (*e.g.*, BERT (Devlin et al., 2019)) over sanitized texts for building the model.

Our design is to build in privacy at the root again, in contrast to the afterthought approach. We found it beneficial to sanitize even the public data before feeding them to training. It is not for protecting the public data per se. The intuition here is that it "prepares" the model to work with sanitized queries, which explains our eventual (slight) increase in accuracy while *additionally ensuring privacy*.

Specifically, we propose a sanitization-aware pretraining procedure (Figure 1). We first use our mechanisms to sanitize the public texts, mask the sanitized texts (as in BERT), and train the LM by predicting a `MASK` position as its *original unsanitized token*. LMs preptrained with our sanitization-aware procedure are expected to be more robust to noises in the sanitized texts and achieve better utility when fine-tuning on downstream tasks.

We conduct experiments on three representative NLP tasks to empirically confirm that our proposed PPNLP pipeline preserves both utility and privacy. It turns out that our sanitization-based pretraining (using only $1/6$ of data used in the original BERT pretraining) can even improve the utility of NLP tasks while maintaining privacy comparable to the original BERT. Note that there is an inherent tension between utility and privacy, and privacy attack is also inference in nature. To empirically demonstrate the privacy aspect of our pipeline, *i.e.*, it does not make our model a more powerful tool helping the attacker, we also conduct the "mask token inference" attack on private texts, which infers the masked token given its context based on BERT. As a highlight, our base solution SANTEXT improves the defense rate by 20% with only a 4% utility loss on the SST-2 dataset. We attribute our surprising result of mostly helping only good guys to our natural approach: to avoid the model memorizing sensitive texts "too well," we fed it with sanitized text.

2 Related Work

Privacy risks in NLP. A taxonomy of attacks that recover sensitive attributes or partial raw text from text embeddings output by popular LMs has been proposed (Song and Raghunathan, 2020), without any assumptions on the structures or patterns in input text. Carlini et al. (2020b) also show a powerful black-box attack on GPT-2 (Radford et al., 2019) that extracts verbatim texts of training data. Defense with rigorous guarantees (DP) is thus vital.

Differential privacy and its application in NLP. DP (Dwork, 2006) has emerged as the *de facto* standard for statistical analytics (Wang et al., 2017, 2018; Cormode et al., 2018). A few efforts inject high-dimensional DP noise into text representations (Feyisetan et al., 2019, 2020; Lyu et al., 2020a,b). The noisy representations are not human-readable and not directly usable by existing NLP pipelines, *i.e.*, they consider a different problem not directly comparable to ours. More importantly, they fail to strike a nice privacy-utility balance due to "the curse of dimensionality," *i.e.*, the magnitude of the noise is too large for high-dimensional token embedding, and thus it becomes exponentially less likely to find a noisy embedding close to a real one on every dimension. This may also explain why an earlier work focuses on document-level statistics only, *e.g.*, term-frequency vectors (Weggenmann and Kerschbaum, 2018).

Our approaches produce natively usable sanitized texts via directly sampling a substitution for each token from a precomputed distribution (to be detailed in Section 4), circumventing the dimension curse and striking a privacy-utility tradeoff while being much more efficient. A concurrent work (Qu et al., 2021) also considers the whole NLP pipeline, but it still builds on the token-projection approach (Feyisetan et al., 2020).

Privacy-preserving text representations. Learning private text representations via adversarial training is also an active area (Xie et al., 2017; Coavoux et al., 2018; Elazar and Goldberg, 2018; Li et al., 2018). An adversary is trained to infer sensitive information jointly with the main model, while the main model is trained to maximize the adversary's loss and minimize the primary learning objective. While we share the same general goal, our aim is not such representations (similar to those with DP) but to release sanitized text for general purposes.

3 Defining (Local) Differential Privacy

Suppose each user holds a document $D = \langle x_i \rangle_{i=1}^{L}$ of L tokens (which can be a character, a subword, a word, or an n-gram), where x_i is from a vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$. For privacy, each user derives a sanitized version $\hat{D}$ by running a common text sanitization mechanism $\mathcal{M}$ over D on local devices. Specifically, $\mathcal{M}$ works by replacing every token x_i in D with a substitution $y_i \in \mathcal{V}$, assuming that x_i itself is unnecessary for NLP tasks while its semantics should be preserved for high utility. The output $\hat{D}$ is then shared with an NLP service provider.

We consider a typical threat model in which each user does not trust any other party and views them as an attacker with access to $\hat{D}$ in conjunction with any auxiliary information (including $\mathcal{M}$).

3.1 (Variants of) Local Differential Privacy

Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and output spaces. A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is a probabilistic function that assigns a random output $y \in \mathcal{Y}$ to an input $x \in \mathcal{X}$. Every y induces a probability distribution on the underlying space. For sanitizing text, we set both $\mathcal{X}$ and $\mathcal{Y}$ as the vocabulary $\mathcal{V}$.

Definition 1 (ϵ-LDP (Duchi et al., 2013))**.** *Given a privacy parameter* $\epsilon \geq 0$, $\mathcal{M}$ *satisfies* ϵ*-local differential privacy (*ϵ*-LDP) if, for any* $x, x', y \in \mathcal{V}$,

$$\Pr[\mathcal{M}(x) = y] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(x') = y].$$

Given an observed output y, from the attacker's view, the likelihoods y is derived from x and x' are similar. A smaller ϵ means better privacy due to a higher indistinguishability level of output distributions, yet the outputs retain less utility.

ϵ-LDP is a very strong privacy notion for its homogeneous protection over all input pairs. However, this is also detrimental to the utility: no matter how unrelated x and x' are, their output distributions must be similar. As a result, a sanitized token y may not (approximately) capture the semantics of its input x, degrading the downstream tasks.

LDP over metric spaces. To capture semantics, we borrow the relaxed notion of Metric-LDP (MLDP) (Alvim et al., 2018) originally proposed for location privacy (Andrés et al., 2013) with the distance metric $d(\cdot, \cdot)$ between two locations (*e.g.*, Manhattan distance (Chatzikokolakis et al., 2013)).

Definition 2 (MLDP)**.** *Given* $\epsilon \geq 0$ *and a distance metric* $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_{\geq 0}$ *over* $\mathcal{V}$, $\mathcal{M}$ *satisfies*

MLDP or $\epsilon \cdot d(x, x')$*-LDP if, for any* $x, x', y \in \mathcal{V}$,

$$\Pr[\mathcal{M}(x) = y] \le e^{\epsilon \cdot d(x,x')} \cdot \Pr[\mathcal{M}(x') = y].$$

When $d(x, x') = 1 \; \forall x \neq x'$, MLDP becomes LDP. For MLDP, the indistinguishability of output distributions is further scaled by the distance between the respective inputs. Roughly, the effect of ϵ becomes "adaptive." To apply MLDP, one needs to carefully define the metric d (see Section 4.2).

Incorporating ULDP to further improve utility. Utility-optimized LDP (Murakami and Kawamoto, 2019) (ULDP) also relaxes LDP, which was originally proposed for aggregating ordinal responses. It exploits the fact that different inputs have different sensitivity levels to achieve higher utility. By assuming that the input space is split into *sensitive* and *non-sensitive* parts, ULDP achieves a privacy guarantee equivalent to LDP for *sensitive* inputs.

In our context, more formally speaking, let $\mathcal{V}_S \subseteq \mathcal{V}$ be the set of sensitive tokens common to all users, and $\mathcal{V}_N = \mathcal{V} \setminus \mathcal{V}_S$ be the set of remaining tokens. The output space $\mathcal{V}$ is split into the *protected* part $\mathcal{V}_P \subseteq \mathcal{V}$ and the *unprotected* part $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_P$.

The image of $\mathcal{V}_S$ is restricted to $\mathcal{V}_P$, *i.e.*, a sensitive $x \in \mathcal{V}_S$ can only be mapped to a protected $y \in \mathcal{V}_P$. For text, we can set $\mathcal{V}_S = \mathcal{V}_P$ for simplicity. While a non-sensitive $x \in V_N$ can be mapped to $\mathcal{V}_P$, every $y \in \mathcal{V}_U$ must be mapped from $\mathcal{V}_N$, which helps to improve the utility.

3.2 Our New Utility-optimized MLDP Notion

Among many variants of (L)DP notions, we found the above two variants (*i.e.*, ULDP and MLDP) provide useful insight in quantifying semantics and privacy of text data. We thus formulate the new privacy notion of utility-optimized MLDP (UMLDP).

Definition 3 (UMLDP)**.** *Given* $\mathcal{V}_S \cup \mathcal{V}_N = \mathcal{V}$*, two privacy parameters* $\epsilon, \epsilon_0 \geq 0$*, and a distance metric* $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_{\geq 0}$*,* $\mathcal{M}$ *satisfies* $(\mathcal{V}_S, \mathcal{V}_P, \epsilon, \epsilon_0)$*-UMLDP, if*
i) for any $x, x' \in \mathcal{V}$ *and any* $y \in \mathcal{V}_P$*, we have*

$$\Pr[\mathcal{M}(x) = y] \le e^{\epsilon d(x,x') + \epsilon_0} \Pr[\mathcal{M}(x') = y];$$

ii) for any $y \in \mathcal{V}_U$*, i.e., from an unprotected set* $\mathcal{V}_U$ *where* $\mathcal{V}_U \cap \mathcal{V}_P = \emptyset$*, there is an* $x \in \mathcal{V}_N$ *such that*

$$\Pr[\mathcal{M}(x) = y] > 0,$$
$$\Pr[\mathcal{M}(x') = y] = 0 \; \forall x' \in \mathcal{V} \setminus \{x\}.$$

Figure 2 summarizes the treatment of UMLDP. It exhibits "invertibility," *i.e.*, $y \in \mathcal{V}_U$ must be "noise-free" and mapped deterministically. Apart from generalizing ϵ in the ULDP definition (recalled in Appendix A.1) into $\epsilon d(x, x')$, we incorporate an additive bound ϵ_0 due to the invertibility, which makes the derivation of ϵ easier. Looking ahead, ϵ_0 would appear naturally in the analysis of our UMLDP mechanism for the invertible case.

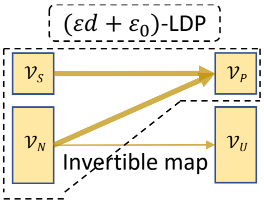

Figure 2: Overview of our new UMLDP notion

UMLDP (and MLDP), as an LDP notion, satisfies the *composability* and *free post-processing*. The former means that the sequential execution of ϵ_1-LDP and ϵ_2-LDP mechanisms satisfies $(\epsilon_1 + \epsilon_2)$-LDP, *i.e.*, ϵ can be viewed as the privacy "budget" of a sophisticated task comprising multiple subroutines, each consumes a part of ϵ such that their sum equals ϵ. The latter means further processing the mechanism outputs incurs no extra privacy loss.

4 Our Privacy-Preserving NLP Pipeline

4.1 Overview

We propose two token-wise sanitization methods with (U)MLDP: SANTEXT and SANTEXT$^+$, which build atop a variant of the exponential mechanism (EM) (McSherry and Talwar, 2007) over the "native" text tokens as both input and output spaces to avoid going to the "cursed dimensions" of token embeddings. EM samples a replacement y for an input x based on an exponential distribution, with more "suitable" y's sampled with higher probability (detailed below). It is well-suited for (U)MLDP by considering the "suitability" as how well the semantics of x is preserved for the downstream tasks (run over the sanitized text y) to remain accurate.

To quantify this, we utilize an embedding model mapping tokens into a real-valued vector space. The semantic similarity among tokens can then be measured via the Euclidean distance between their corresponding vectors. Our base design SANTEXT outputs y with probability inverse proportional to the distance between x and y: the shorter the distance, the more semantically similar they are. SANTEXT$^+$ considers some tokens $\mathcal{V}_N$ in $\mathcal{V}$ are non-sensitive, and runs SANTEXT over the sensitive part $\mathcal{V}_S = V \setminus \mathcal{V}_N$ (*i.e.*, it degenerates to SANTEXT if $\mathcal{V}_S = \mathcal{V}$). For $\mathcal{V}_N$, we tailor a probability distribution to provide UMLDP as a whole.

Algorithm 1: Base Mechanism SanText

```
Input: A private document $D = \langle x_i \rangle_{i=1}^{L}$,
       and a privacy parameter $\epsilon \geq 0$
Output: Sanitized document $\hat{D}$
Derive token vectors $\phi(x_i)$ for $i \in [1, L]$;
for $i = 1, \ldots, L$ do
    Run $\mathcal{M}(x_i)$ to sample a sanitized token
      $y_i$ with probability defined in Eq. (1);
end
Output sanitized $\hat{D}$ as $\langle y_i \rangle_{i=1}^{L}$;
```

With SanText or SanText$^+$, each user sanitizes D into $\hat{D}$ and uploads it to the service provider for performing any NLP task built atop a pretrained LM, *e.g.*, BERT. Typically, the task pipeline consists of an embedding layer, an encoder module, and task-specific layers, *e.g.*, for classification.

Without the raw text, the utility can degrade; we thus propose two approaches for improving it. The first one is to pretrain only the encoder on the sanitized public corpus to adapt to the noise. It is optional if pretraining is deemed costly. The second is to fine-tune the full pipeline on $\hat{D}$'s, which updates both the encoder and task layers.

4.2 Base Sanitization Mechanism: SanText

In NLP, a common step is to employ an embedding model[1] mapping semantically similar tokens to close vectors in a Euclidean space. Concretely, an embedding model is an injective mapping $\phi : \mathcal{V} \to \mathbb{R}^m$, for dimensionality m. The distance between any two tokens x and x' can be measured by the Euclidean distance of their embeddings: $d(x, x') = d_{\text{euc}}(\phi(x), \phi(x'))$. As ϕ is injective, d satisfies the axioms of a distance metric.

Algorithm 1 lists the pseudo-code of SanText for sanitizing a private document D at the user side. The first step is to use ϕ to derive token embeddings of each token[2] x in D. Then, for each x, we run $\mathcal{M}(x)$ to sample a sanitized y with probability

$$\Pr[\mathcal{M}(x) = y] = C_x \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x), \phi(y))} \quad (1)$$

where $C_x = (\sum_{y' \in \mathcal{V}} e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x), \phi(y'))})^{-1}$.

The smaller $d_{\text{euc}}(\phi(x), \phi(y))$, the more likely y is to replace x. To boost the sanitizing efficiency, we can precompute a $|\mathcal{V}| \times |\mathcal{V}|$ probability matrix, where each entry (i, j) denotes the probability of outputting y_j on input x_i, upon obtaining $\phi(x)$ for $\forall x \in \mathcal{V}$. Lastly, the sanitized $\hat{D} = \langle y_i \rangle_{i=1}^{L}$ can be released to the service provider for NLP tasks.

Algorithm 2: Enhanced SanText$^+$

```
Input: A private document $D = \langle x_i \rangle_{i=1}^{L}$, a
       privacy parameter $\epsilon \geq 0$, probability
       $p$ for a biased coin, and sensitive $\mathcal{V}_S$
Output: Sanitized document $\hat{D}$
Derive token vectors $\phi(x_i)$ for $i \in [1, L]$;
for $i = 1, \ldots, L$ do
    if $x_i \in \mathcal{V}_S$ then
        Sample a substitution $y_i \in \mathcal{V}_P = \mathcal{V}_S$
          with probability given in Eq. (1) ▷
          Run SanText over $\mathcal{V}_S$ and $\mathcal{V}_P$;
    else
        Output $y_i = x_i$ with prob. $(1 - p)$;
          or $y_i \in \mathcal{V}_P$ with prob. in Eq. (2);
    end
end
Output sanitized $\hat{D}$ as $\langle y_i \rangle_{i=1}^{L}$;
```

4.3 Enhanced Mechanism: SanText$^+$

In SanText, all tokens in $\mathcal{V}$ are treated as sensitive, which leads to excessive protection and utility loss. Following the less-is-more principle, we divide $\mathcal{V}$ into $\mathcal{V}_S$ and $\mathcal{V}_N$, and focus on protecting $\mathcal{V}_S$.

Observing that most frequently used tokens (*e.g.*, a/an/the) are non-sensitive to virtually all users, we use token frequencies for division. A simple strategy, which is also used in our experiments, is to mark the top w of low-frequency tokens (according to a certain corpus) as $\mathcal{V}_S$, where w is a tunable parameter. Looking ahead, this "basic" method already showed promising results. (Further discussion can be found in Section 4.5).

Algorithm 2 lists the pseudo-code of SanText$^+$ with $\mathcal{V}_S = \mathcal{V}_P$ and $\mathcal{V}_N = \mathcal{V}_U$ shared by all users. The first step, as in SanText, is to derive the token embeddings in D. Then, for each token x, if it is in $\mathcal{V}_S$, we sample its substitution y from $\mathcal{V}_P$ with probability given in Eq. (1). (This is equivalent to running SanText over $\mathcal{V}_S$ and $\mathcal{V}_P$.) For $x \in \mathcal{V}_N$, we toss a biased coin. With probability $(1 - p)$, we output y as x (*i.e.*, the "invertibility"). Otherwise, we sample $y \in \mathcal{V}_P$ with probability

$$\Pr[\mathcal{M}(x) = y] = p \cdot C_x \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x), \phi(y))} \quad (2)$$

where $C_x = (\sum_{y' \in \mathcal{V}_P} e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x), \phi(y'))})^{-1}$.

As in SanText, we can also precompute two $|\mathcal{V}_S| \times |\mathcal{V}_P|$ and $|\mathcal{V}_N| \times |\mathcal{V}_P|$ probability matrices, which correspond to Eq. (1) and (2), for optimizing

---

[1] We assume that it has been trained on a large public corpus and shared by all users.

[2] For easy presentation, we omit the subscript i later.

the sanitizing efficiency. Lastly, the sanitized $\hat{D}$ of $\langle y \rangle_{i=1}^{L}$ can be released to the service provider.

Theorem 1. *Given* $\epsilon \geq 0$ *and* d_{euc} *over the embedding space* ϕ *of* $\mathcal{V}$*,* SANTEXT *satisfies MLDP.*

Theorem 2. *Given* $(\mathcal{V}_S = \mathcal{V}_P) \subseteq \mathcal{V}$*,* $\epsilon \geq 0$*,* $\epsilon_0 = \ln \frac{1}{p} \geq 0$*, and* d_{euc} *over the embedding space* ϕ *of* $\mathcal{V}$*,* SANTEXT$^+$ *satisfies* $(\mathcal{V}_S, \mathcal{V}_P, \epsilon, \epsilon_0)$*-UMLDP.*

Their proofs are in Appendix A.2.

4.4 NLP over Sanitized Text

With $\hat{D}$'s (shared by the users), the service provider can perform any NLP task. In this work, we focus on those built on a pretrained LM, and in particular, we study BERT as an example due to its wide adoption and superior performance. The full NLP pipeline is deployed at the service provider.

Given a piece of (sanitized) text, the embedding layer maps it to a sequence of token embeddings. The encoder computes a sequence representation from the token embeddings, allowing task-specific layers to make predictions. For example, the task layer could be a feed-forward neural network for multi-label classification of a diagnosis system.

The injected noise deteriorates the performance of downstream tasks as the service provider cannot access the raw texts $\{D\}$. To mitigate this, we propose two approaches – pretraining the encoder and fine-tuning the full pipeline, which allow the tasks to be "adaptive" to the noise to some extent.

Pretraining BERT over sanitized public corpus. Besides $\hat{D}$'s, the service provider can also obtain a massive amount of text that is publicly available (say, the English Wikipedia). It also has access to the sanitization mechanisms, and it can produce the sanitized public text (as how users produce $\hat{D}$'s).

Our key idea is to let the service provider pretrain the encoder (*i.e.*, BERT) over the sanitized public text, making it more "robust" in handling $\hat{D}$'s. We thus initialize the encoder with the original BERT checkpoint and conduct further pretraining with an adapted masked language model (MLM) loss. In more detail, the adapted MLM objective is to predict the *original masked tokens* given the sanitized context instead of the one from the raw public text. We note that this is beneficial for improving the task utility, yet may breach the user privacy as the objective learns to "recover" the original tokens or semantics. In Section 5.4, our results will show that such pretrained BERT indeed improves accuracy, with comparable privacy as in original BERT.

Fine-tuning the full NLP pipeline. After pretraining BERT using sanitized public text, the service provider can further improve the efficacy of downstream tasks by fine-tuning the full pipeline. We assume that the ground-truth labels are available to the service provider, say, inferring from $\hat{D}$'s when they can preserve similar semantics to the raw text. Then, the sanitized text-label pairs are used for training/fine-tuning downstream task models, with gradients back-propagated to update the parameters of both the encoder and task layer. We leave more realistic/complex labeling processes based on sanitized texts as future work.

4.5 Definition of "Sensitivity"

Simply treating the top w of least frequent tokens (*e.g.*, according to a public reference corpus) as the sensitive token set already led to promising results (see Section 5.2). By this definition, stop words are mostly non-sensitive (*e.g.*, for $w = 0.9$ over the sentiment classification dataset we used, $\sim$98% of the stop words are deemed non-sensitive). For context-specific corpus, this strategy is better than merely using stop words, *e.g.*, breast cancer becomes non-sensitive among breast-cancer patients.

Sophisticated machine-learning approaches or other heuristics could also be considered, *e.g.*, training over context-specific reference corpus or identifying tokens with personal (and hence sensitive) information (*e.g.*, names). We leave as future work.

Moreover, the definition of sensitivity may vary across users. Some may consider a token deemed non-sensitive by most other users sensitive. The original ULDP work (Murakami and Kawamoto, 2019) has discussed a personalized mechanism that preprocesses such tokens by mapping them to a set of semantic tags, which are the same for all users. These tags will be treated as sensitive tokens for the ULDP mechanism. Apparently, this approach is application-specific and may not be needed in some applications; hence we omit it in this work.

5 Experiments

5.1 Experimental Setup

We consider three representative downstream NLP tasks (datasets) with privacy implications.

Sentiment Classification (SST-2). When people write online reviews, especially the negative ones, they may worry about having their identity traced via writing too much that may provide hints of authorship or linkage to other online writings. For

Mechanisms	**SST-2**			**MedSTS**			**QNLI**		
	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$
Random	0.4986	0.4986	0.4986	0.0196	0.0196	0.0196	0.5152	0.5152	0.5152
Feyisetan et al. (2020)	0.5099	0.5143	0.5345	0.0201	0.0361	0.0452	0.5162	0.5256	0.5333
SanText	0.5101	0.5838	0.8374	0.0351	0.5392	0.8159	0.5372	0.5598	0.8116
SanText$^+$	**0.7796**	**0.7943**	**0.8516**	**0.4965**	**0.7082**	**0.8162**	**0.7699**	**0.7760**	**0.8131**
Unsanitized	0.9251			0.8527			0.9090		

Table 1: Utilities comparison of sanitization mechanisms under similar privacy levels using the GloVe embedding

this task, we use the preprocessed version in GLUE benchmark (Wang et al., 2019) of (binary) Stanford Sentiment Treebank (SST-2) dataset (Socher et al., 2013). *Accuracy* (w.r.t. the ground truth included in the dataset) is used as the evaluation metric.

Medical Semantic Textual Similarity (MedSTS). Automated processing of patient records is a significant research direction, and one such task is computing the semantic similarity between clinical text snippets for the benefit of reducing the cognitive burden. We choose a very recent MedSTS dataset (Wang et al., 2020) for this task, which assigns a numerical score to each pair of sentences, indicating the degree of similarity. We report the *Pearson correlation coefficient* (between predicted similarities and human judgments) for this task.

Question Natural Language Inference (QNLI). Question-answering (QA) aims to automatically answer user questions based on documents. We consider a simplified setting of QA, namely QNLI, which predicts whether a given document contains the answer to the question. We use the QNLI dataset from GLUE benchmark (Wang et al., 2019).

We implement our sanitized mechanisms using Python and the sanitization-aware training using the Transformers library (Wolf et al., 2020). We use sanitized data to train and test prediction models for all three tasks. We either build vocabularies for the tasks using GloVe embeddings (Pennington et al., 2014) or adopt the same BERT vocabulary (Devlin et al., 2019). Table 2 shows their sizes. Our sanitization-aware pretraining uses WikiCorpus (English version, a 2006 dump, 600M words) (Reese et al., 2010). We start from the `bert-base-uncased` (instead of randomly initialized) model to accelerate the pretraining.

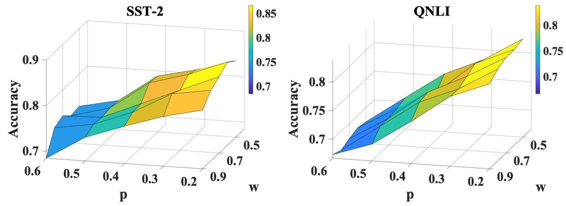

Figure 3: Performance of SanText$^+$ over (w, p) when fixing $\epsilon = 2$ based on the GloVe embedding

We set the maximum sequence length to 512, training epoch to 1, batch size to 6, learning rate to 5e-5, warmup steps to 2000, and MLM probability to 0.15. Our sanitization-aware fine-tuning uses the `bert-base-uncased` model for SST-2/QNLI, and `ClinicalBERT` (Alsentzer et al., 2019) for MedSTS. We set the maximum sequence length to 128, training epochs to 3, batch size to 64 for SST-2/QNLI or 8 for MedSTS, and learning rate to 2e-5 for SST-2/QNLI or 5e-5 for MedSTS. Other hyperparameters are kept default. Our hyperparameters followed the transformer library (Wolf et al., 2020) and popular setups in the original dataset literature (Wang et al., 2019, 2020).

5.2 Comparison of Sanitization Mechanisms

We first compare our SanText and SanText$^+$ with random sanitization and the state-of-the-art of Feyisetan et al. (FBDD). Here, we use the GloVe embedding as in FBDD for a fair comparison. Random sanitization picks a token from the vocabulary uniformly. We set the UMLDP parameters $p = 0.3, w = 0.9$ for SanText$^+$ (while Figure 3 plots the impacts of p and w when fixing $\epsilon = 2$).

Table 1 shows the utility of the four mechanisms for the three selected tasks at different privacy levels. FBDD has a higher utility than random replacements. While both FBDD and SanText are based on word embeddings, SanText does not suffer from the "curse-of-dimensionality" and achieves better utility at the same privacy level. SanText$^+$ achieves the best utilities in all cases since it allows the non-sensitive tokens to be noise-free, lowering the noise and improving the utility.

In terms of efficiency, our SanText and SanText$^+$ are very efficient (*e.g.*, ∼2min for the SST-2 dataset) compared with FBDD (∼117min) when they all run on a 24 core CPU machine. This is because our mechanisms only need to compute the sampling probability once and use the same probability matrix for sampling each time, while FBDD needs to recalculate the additive noise and re-search the nearest neighbor each time.

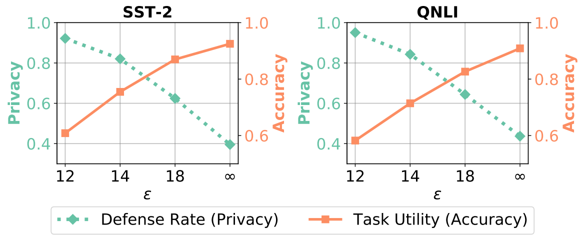

Figure 4: Privacy and Utility Tradeoffs of SANTEXT in terms of Defense Rate (of the Mask Token Inference Attack) versus Accuracy ($\epsilon = \infty$ means "unsanitized.")

Datasets	GloVe embeddings		BERT embeddings	
	$\mathcal{V}$	$\mathcal{V}_S$	$\mathcal{V}$	$\mathcal{V}_S$
SST-2	14,730	13,258	30,522	27,469
MedSTS	3,320	2,989		
QNLI	88,159	79,343		

Table 2: Sizes of vocabularies ($w = 0.9$ for $\mathcal{V}_S$)

5.3 Mask Token Inference Attack

From now on, we adopt the BERT embedding for its superiority. As (U)MLDP is distance-metric dependent, we need to use different ϵ's (*e.g.*, Figure 5) to ensure a similar privacy level, specifically, $\epsilon \cdot d$.

Our sanitization mechanisms provide broad protection for seen/unseen attacks at a fundamental level (by sampling noise to directly replace original tokens) with formally-proven DP, *e.g.*, two guesses of the original token with different styles are nearly probable in an attempt of authorship attribution (Weggenmann and Kerschbaum, 2018) or other "indirect" attacks. Here, we consider a *mask token inference attack* as a representative study to "confirm the theory" by empirically measuring the "concrete" privacy level of sanitized texts.

To infer or recover original tokens given the sanitized text, one can let a pretrained BERT model infer the `MASK` token given its contexts. After all, BERT models are trained via masked language modeling. For each sanitized text of the downstream (private) corpus, we replace each token sequentially by the special token `[MASK]` and input the masked text to the pretrained BERT model to obtain the prediction of the `[MASK]` position. Then, we compare the predicted token to the original token in the raw text. Figure 4 reports the defense rate (the proportion of unmatched tokens to total tokens) and task utility of sanitized texts (by SANTEXT) as well as unsanitized texts on SST-2 and QNLI. We see a privacy-utility trade-off: the more restrictive the privacy guarantee (smaller ϵ), the lower the utility score. Notably, we improve the defense rate substantially with only a small amount of privacy loss (*e.g.*, when $\epsilon = 16$, SANTEXT improves the defense rate by 20% with only 4% task utility loss over the SST-2 dataset in Figure 4).

Datasets	ϵ	**Utility**		Δ_{privacy}
		Original	**+Pretrain**	
SST-2	12	0.6084	**0.6208**	0.0089
	14	0.7548	**0.7731**	0.0101
	16	0.8698	**0.8830**	-0.0046
QNLI	12	0.5822	**0.6037**	0.0076
	14	0.7143	**0.7309**	-0.0047
	16	0.8265	**0.8369**	-0.0039

Table 3: Sanitization-aware pretraining via SANTEXT

5.4 Effectiveness of Pretraining

We then show how the sanitization-aware pretraining further improves the utility but does not hurt the original privacy. Specifically, Table 3 compares the accuracy of sanitization-aware fine-tuning based on the publicly-available `bert-base-uncased` model and our sanitization-aware pretrained one at different privacy levels on SST-2 and QNLI. Our sanitization-aware pretrained BERT models can obtain a 2% absolute gain on average. We conjecture that it can be improved since our pretraining only uses $1/6$ of the data used in the original BERT pretraining and 1 training epoch as an illustration.

To demonstrate that such utility improvement is not obtained by sacrificing privacy, we record the change of defense rate (Δ_{privacy}) in launching mask token inference attacks on the original BERT models and our sanitization-aware pretrained BERT models. As Table 3 confirmed, the privacy level of our sanitization-aware pretrained model is nearly the same as the original (sometimes even better).

5.5 Influence of Privacy Parameter ϵ

We aim at striking a nice balance between privacy and utility by tuning ϵ. To empirically show the influence of ϵ, we report the utility and privacy scores over the SST-2 dataset based on SANTEXT. The utility score is the accuracy over the test set. We define three metrics to "quantify" privacy. Firstly, $N_x = \Pr[\mathcal{M}(x) = x]$, which we estimate by the frequency of seeing no replacement by $\mathcal{M}()$. The output distribution of x has full support over $\mathcal{V}$, *i.e.*, $\Pr[\mathcal{M}(x) = y] > 0$ for any $y \in \mathcal{V}$. Yet, we are interested in the effective support $\mathcal{S}$, a set of y's with cumulative probability larger than a threshold, and then define S_x as its size. S_x can be estimated by the number of distinct tokens mapped from x. Both N_x and S_x can be related to two extremes of the Rényi entropy (Rényi, 1961), defined as

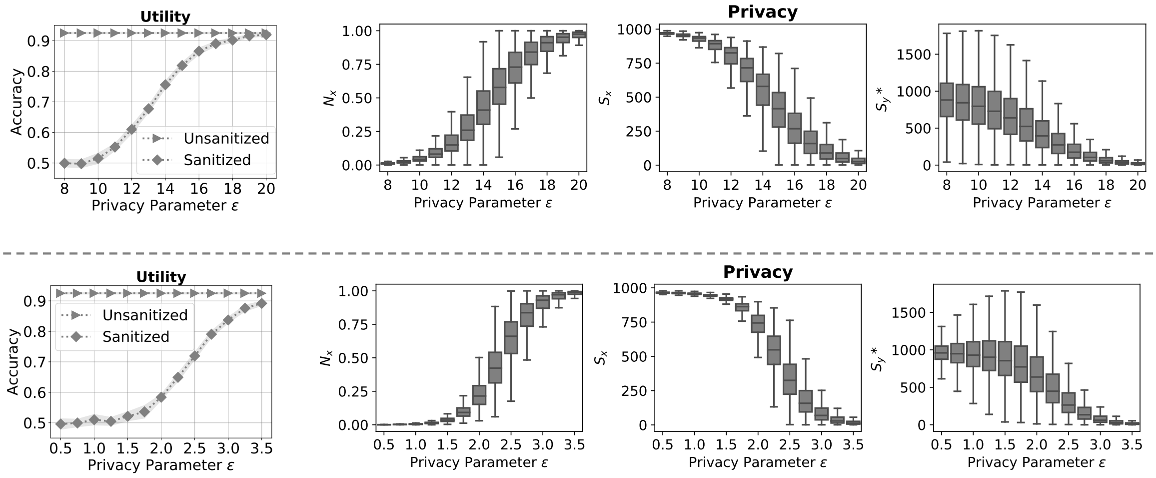

Figure 5: Influence of privacy parameter ϵ of SanText on the utility and privacy (N_x, S_x, S_y^*) based on the SST-2 dataset: The top panel is based on BERT embeddings, and the bottom panel is based on GloVe embeddings.

$H_\alpha(\mathcal{M}(x)) = \frac{1}{1-\alpha}\log(\sum_{y\in\mathcal{V}}\Pr[\mathcal{M}(x) = y]^\alpha)$, with an order $\alpha \geq 0$ and $\alpha \neq 1$. The two extremes are obtained by setting $\alpha = 0$ and $= \infty$, resulting in the Hartley entropy H_0 and the min-entropy H_∞. This implies that we can also approximate H_0 and H_∞ by $\log S_x$ and $-\log N_x$, respectively. Making them large increases the entropy of the distribution.

Another important notion is *plausible deniability* (Bindschaedler et al., 2017), *i.e.*, a set of x's could have led to an output y with a similar probability. We define S_y^* as the set size, estimated by the number of distinct tokens mapped to y.

We run SanText $1,000$ times for the whole SST-2 dataset vocabulary. As Figure 5 shows, when ϵ increases, the utility boosts and N_x increases while S_x, S_y^*, and the privacy level of the mechanism decrease, which gives some intuition in picking ϵ, *e.g.*, for ~40% probability of replacing each token to a different one based on the BERT embeddings (top panel), we could set $\epsilon = 15$ since the median of N_x is ~60% and the accuracy is ~81%.

6 Conclusion

Great predictive power comes with great privacy risks. The success of language models enables inference attacks. There are only a few works in differentially private (DP) text sanitization, probably due to its intrinsic difficulty. A new approach addressing the (inherent) limitation (*e.g.*, in generality) of existing works is thus needed.

Theoretically, we formulate a new LDP notion, UMLDP, which considers both sensitivity and similarity. While it is motivated by text analytics, it remains interesting in its own right. UMLDP enables our natural sanitization mechanisms without the curse of dimensionality faced by existing works.

Practically, we consider the whole PPNLP pipeline and build in privacy at the root with our sanitization-aware pretraining and fine-tuning. With our simple and clear definition of sensitivity, our work already achieved promising performance. Future research in sophisticated sensitivity measures will further strengthen our approach.

Surprisingly, our PPNLP solution is discerning like a cryptographic solution: it is kind (maintains high utility) to the good but not as helpful to the bad (not boosting up inference attacks). We hope our results with different metrics for quantifying privacy can provide more insights in privacy-preserving NLP and make it accessible to a broad audience.

Acknowledgements

Authors at OSU are sponsored in part by the PCORI Funding ME-2017C1-6413, the Army Research Office under cooperative agreements W911NF-17-1-0412, NSF Grant IIS1815674, NSF CAREER #1942980, and Ohio Supercomputer Center (OSC, 1987). The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

Sherman Chow's research is supported by General Research Fund (CUHK 14210319) of UGC, HK. Authors at CUHK would like to thank Florian Kerschbaum for his inspiring talk given at CUHK that stimulated this research.

References

Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. Publicly available clinical BERT embeddings. In *2nd Clinical Natural Language Processing Workshop*, pages 72–78. Also available at arXiv:1904.03323.

Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Anna Pazii. 2018. Local differential privacy on metric spaces: Optimizing the trade-off with utility. In *CSF*, pages 262–267.

Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: differential privacy for location-based systems. In *CCS*, pages 901–914.

Vincent Bindschaedler, Reza Shokri, and Carl A. Gunter. 2017. Plausible deniability for privacy-preserving data synthesis. *PVLDB*, 10(5):481–492.

Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramèr. 2020a. An attack on InstaHide: Is private learning possible with instance encoding? arXiv:2011.05315.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020b. Extracting training data from large language models. arXiv:2012.07805.

Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *PETS*, pages 82–102.

Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. 2018. Privacy-preserving neural representations of text. In *EMNLP*, pages 1–10.

Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal release under local differential privacy. In *SIGMOD*, pages 131–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438.

Cynthia Dwork. 2006. Differential privacy. In *ICALP*, pages 1–12.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *EMNLP*, pages 11–21.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy- and utility-preserving textual analysis via calibrated multivariate perturbations. In *WSDM*, pages 178–186.

Oluwaseyi Feyisetan, Tom Diethe, and Thomas Drake. 2019. Leveraging hierarchical representations for preserving privacy and utility in text. In *ICDM*, pages 210–219.

Yangsibo Huang, Zhao Song, Danqi Chen, Kai Li, and Sanjeev Arora. 2020. TextHide: Tackling data privacy for language understanding tasks. In *EMNLP (Findings)*, pages 1368–1382.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *ACL*, pages 25–30.

Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. 2020. Glyph: Fast and accurately training deep neural networks on encrypted data. In *NeurIPS*.

Lingjuan Lyu, Xuanli He, and Yitong Li. 2020a. Differentially private representation for NLP: Formal guarantee and an empirical study on privacy and fairness. In *EMNLP (Findings)*, pages 2355–2365.

Lingjuan Lyu, Yitong Li, Xuanli He, and Tong Xiao. 2020b. Towards differentially private text representations. In *SIGIR*, pages 1813–1816.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *ICLR (Poster)*.

Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *FOCS*, pages 94–103.

Takao Murakami and Yusuke Kawamoto. 2019. Utility-optimized local differential privacy mechanisms for distribution estimation. In *USENIX Security*, pages 1877–1894.

Lucien K. L. Ng and Sherman S. M. Chow. 2021. GForce: GPU-friendly oblivious and rapid neural network inference. In *USENIX Security*.

Lucien K. L. Ng, Sherman S. M. Chow, Anna P. Y. Woo, Donald P. H. Wong, and Yongjun Zhao. 2021. Goten: GPU-Outsourcing Trusted Execution of Neural Network Training. In *AAAI*.

OSC. 1987. Ohio supercomputer center.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *S&P*, pages 1314–1331.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543. 300-d GloVe embeddings available at `http://nlp.stanford.edu/data/glove.840B.300d.zip`.

Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Privacy-adaptive BERT for natural language understanding. arXiv:2104.07504.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI report*.

Samuel Reese, Gemma Boleda, Montse Cuadros, Lluís Padró, and German Rigau. 2010. Wikicorpus: A word-sense disambiguated multilingual wikipedia corpus. In *LREC*.

Alfréd Rényi. 1961. On measures of entropy and information. In *Berkeley Symposium on Mathematical Statistics and Probability*, pages 547–561.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *S&P*, pages 3–18.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *CCS*, pages 377–390.

Latanya Sweeney. 2015. Only you, your doctor, and many others may know. *Technology Science*, 2015092903(9):29.

Florian Tramèr and Dan Boneh. 2019. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *ICLR*.

Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2021. FALCON: Honest-majority maliciously secure framework for private deep learning. *PoPETs*, 2021(1):187–207.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR (Poster)*.

Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *USENIX Security*, pages 729–745.

Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally differentially private frequent itemset mining. In *S&P*, pages 127–143.

Yanshan Wang, Naveed Afzal, Sunyang Fu, Liwei Wang, Feichen Shen, Majid Rastegar-Mojarad, and Hongfang Liu. 2020. MedSTS: a resource for clinical semantic textual similarity. *Lang. Resour. Evaluation*, 54(1):57–72.

Benjamin Weggenmann and Florian Kerschbaum. 2018. SynTF: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *SIGIR*, pages 305–314.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP (System Demonstrations)*, pages 38–45. Also available at arXiv:1910.03771, and https://github.com/huggingface.

Harry W. H. Wong, Jack P. K. Ma, Donald P. H. Wong, Lucien K. L. Ng, and Sherman S. M. Chow. 2020. Learning model with error - exposing the hidden model of BAYHENN. In *IJCAI*, pages 3529–3535.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard H. Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *NeurIPS*, pages 585–596.

A Supplementary Formalism Details

A.1 Definition of ULDP

Definition 4 (($\mathcal{V}_S, \mathcal{V}_P, \epsilon$)-ULDP (Murakami and Kawamoto, 2019)). *Given* ($\mathcal{V}_S = \mathcal{V}_P$) $\subseteq \mathcal{V}$*, a privacy parameter* $\epsilon \geq 0$*,* $\mathcal{M}$ *satisfies* ($\mathcal{V}_S, \mathcal{V}_P, \epsilon$)*-ULDP if it satisfies the properties:*
i) for any $x, x' \in \mathcal{V}$ *and any* $y \in \mathcal{V}_P$*, we have*

$$\Pr[\mathcal{M}(x) = y] \leq e^{\epsilon} \Pr[\mathcal{M}(x') = y];$$

ii) for any $y \in \mathcal{V}_U$*, there is an* $x \in \mathcal{V}_N$ *such that*

$$\Pr[\mathcal{M}(x) = y] > 0; \Pr[\mathcal{M}(x') = y] = 0 \textit{ for } x \neq x'.$$

A.2 Differential Privacy Guarantee

Proof of Theorem 1. Consider $L = 1$, *i.e.*, $D = \langle x \rangle$. For another document D' with $x' \in \mathcal{V} \setminus \{x\}$ and a possible output $y \in \mathcal{V}$:

$$\begin{aligned}
&\frac{\Pr[\mathcal{M}(x) = y]}{\Pr[\mathcal{M}(x') = y]} \\
=&\frac{C_x \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x),\phi(y))}}{C_{x'} \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x'),\phi(y))}} \\
=&\frac{C_x}{C_{x'}} \cdot e^{\frac{1}{2}\epsilon \cdot [d(x',y) - d(x,y)]} \\
\leq&\frac{C_x}{C_{x'}} \cdot e^{\frac{1}{2}\epsilon \cdot d(x,x')} \\
=&\frac{\sum_{y' \in \mathcal{V}} e^{-\frac{1}{2}\epsilon \cdot d(x',y')}}{\sum_{y' \in \mathcal{V}} e^{-\frac{1}{2}\epsilon \cdot d(x,y')}} \cdot e^{\frac{1}{2}\epsilon \cdot d(x,x')} \\
\leq& e^{\frac{1}{2}\epsilon \cdot d(x,x')} \cdot e^{\frac{1}{2}\epsilon \cdot d(x,x')} = e^{\epsilon \cdot d(x,x')}
\end{aligned}$$

The proof, showing SanText ensures $\epsilon \cdot d(x, x')$-LDP, mainly relies on the triangle inequality of d. To generalize to the case of $L > 1$, we sanitize every token x_i in D independently, and thus:

$$\Pr[\mathcal{M}(D) = \hat{D}] = \prod_{i=1}^{L} \Pr[\mathcal{M}(x_i) = y_i].$$

Then, for any D, D', the privacy bound is given as

$$\frac{\Pr[\mathcal{M}(D) = \hat{D}]}{\Pr[\mathcal{M}(D') = \hat{D}]} \leq e^{\epsilon \cdot \sum_{i=1}^{L} d(x_i, x'_i)},$$

which follows from the composability. □

Proof of Theorem 2. Consider the case $L = 1$ with $D = x$ and $D' = x'$. For $x, x' \in \mathcal{V}_S$, the output y is restricted to $\mathcal{V}_P$, with the proof identical to the above theorem (as SanText is run over $\mathcal{V}_S, \mathcal{V}_P$).

For $x, x' \in \mathcal{V}_N$ and $y \in \mathcal{V}_P$, we have

$$\begin{aligned}
\frac{\Pr[\mathcal{M}(x) = y]}{\Pr[\mathcal{M}(x') = y]} &= \frac{p \cdot C_x \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x),\phi(y))}}{p \cdot C_{x'} \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x'),\phi(y))}} \\
&\leq e^{\epsilon \cdot d(x,x')}.
\end{aligned}$$

For $x \in \mathcal{V}_S$, $x' \in \mathcal{V}_N$, and $y \in \mathcal{V}_P$, we have

$$\begin{aligned}
\frac{\Pr[\mathcal{M}(x) = y]}{\Pr[\mathcal{M}(x') = y]} &= \frac{C_x \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x),\phi(y))}}{p \cdot C_{x'} \cdot e^{-\frac{1}{2}\epsilon \cdot d_{\text{euc}}(\phi(x'),\phi(y))}} \\
&\leq \frac{1}{p} \cdot e^{\epsilon \cdot d(x,x')} = e^{\epsilon \cdot d(x,x') + \epsilon_0}.
\end{aligned}$$

The probability for $x \in \mathcal{V}_N$ is $(1-p)$. The above inequalities thus show that SanText^{+} ensures the properties of UMLDP. Similarly, we use the composability to generalize for $L > 1$. □

A.3 Qualitative Observations

Below, we focus on SanText sanitizing a single token x. We first make two extreme cases explicit.
(1) When $\epsilon = 0$, the distribution in Eq. (1) becomes $\Pr[\mathcal{M}(x) = y] = \frac{1}{|\mathcal{V}|}, \forall y \in \mathcal{V}$. SanText is perfectly private since y is uniformly sampled at random, independent of x. Yet, such a y does not preserve any information of x.
(2) When $\epsilon \to \infty$, we have $\Pr[\mathcal{M}(x) = x] \gg \Pr[\mathcal{M}(x) = y], y \in \mathcal{V} \setminus \{x\}$. $\Pr[\mathcal{M}(x) = x]$ dominates others since $d(x, x) = 0$ and $d(x, y) > 0$. This loses no utility as x almost stays unchanged, yet provides no privacy either.

For a general $\epsilon \in (0, \infty)$, the distribution has full support over $\mathcal{V}$, *i.e.*, we have a non-zero probability for any possible $y \in \mathcal{V}$ such that $\mathcal{M}(x) = y$. Also, given $y, y' \in \mathcal{V}$ with $d(x, y) < d(x, y')$, we have $\Pr[\mathcal{M}(x) = y] > \Pr[\mathcal{M}(x) = y']$. As ϵ increases, $\Pr[\mathcal{M}(x) = y]$ for the y's with large $d(x, y)$ goes smaller (and even approaches 0). This means that the output distribution becomes "skewed," *i.e.*, the outputs concentrate on those y's with small $d(x, y)$. This is good for utility, which stems from the semantics preservation of every token. On the contrary, too much concentration weakens the privacy.

For SanText^{+}, the above results directly apply to the case $x \in V_S$ (as SanText is run over $\mathcal{V}_S$ and $\mathcal{V}_P$). There is an extra p determining whether a $x \in \mathcal{V}_N$ is mapped to a $y \in \mathcal{V}_P$. If so, the results are similar except with an extra multiplicative p. A larger p leads to stronger privacy as the probability $(1 - p)$ of x being unchanged becomes smaller.

Dataset: SST-2		
Mechanisms	ϵ	**Original Text:** it 's a charming and often affecting journey .
SanText	1	heated collide. charming activity cause challenges beneath tends
	2	worse beg, charming things working noticed journey basically
	3	all 's. charming and often already journey demonstrating
SanText$^+$	1	it unclear a charming and often hounds journey
	2	it exaggeration a charming feelings often lags journey .
	3	it 's a tiniest picked often affecting journey .
Dataset: QNLI		
Mechanisms	ϵ	**Original Text:** When did Tesla move to New York City? In 1882, Tesla began working for the Continental Edison Company in France, designing and making improvements to electrical equipment.
SanText	1	43 trapper Gaga MCH digest sputtering avenged Forced Laborers Homage Ababa afer psychic 51,000 intercity lambasting nightmare–confederate Frontier Britian Manor Londres shards pilot Mining faster alone Thessalonica Bessemer Lie Columbus
	2	blame least ethos did tenth ballot Condemnation critical filmed In 1883 3200 Conversion pushing 7:57 enabling Town stamp Time downwards Peterson France, GSA emulating addresses appealing 47.4 electrical pull refreshing
	3	Wave did Tesla It way Dru Tully breaking? Tupelo 1875, Tesla began escaped for announcing Continental Edison Company in France However designing and making improvements to electrical Chongqing add
SanText$^+$	1	Rodgers did Sung move to New plantation City ? In K. innumerable Gunz began working sliding the Sultans Edison Company structured France beaching designing disseminate making tribunals to lackluster equipment 40-foot
	2	vaults did Tesla chunks introduces Teknologi Eyes City ? In 866 , Tesla began working for the Analytical Edison Company Butterfly France , designing Sias siblings Noting circumventing electrical orient .
	3	When did Tesla guideline to New York City ? In 1885 , Tesla MG working for the Continental Edison Company in France , translating and dreamed improvements ascertain electrical lookout .

Table 4: Qualitative examples from the SST-2 and QNLI datasets: Sanitized text by our mechanisms at different privacy levels based on GloVe embeddings

B Qualitative Examples

Table 4 shows two examples of sanitized texts output by SanText and SanText$^+$ at different privacy levels from the SST-2 and QNLI datasets.

C Supplementary Related Works

Privacy is a practically relevant topic that also poses research challenges of diverse flavors. Below, we discuss some "less-directly" relevant works, showcasing some latest advances in AI privacy.

Cryptographic Protection of (Text) Analytics. There has been a flurry of results improving privacy-preserving machine-learning frameworks (*e.g.*, (Lou et al., 2020)), which make use of cryptographic tools such as homomorphic encryption and secure multi-party computation (SMC) for general machine/deep learning. These cryptographic designs can be adapted for many NLP tasks in principle. Nevertheless, they will slow down computations by orders of magnitude since cryptographic tools, especially fully homomorphic encryption, are generally more heavyweight than the DP approaches. One might be tempted to replace cryptography with *ad hoc* heuristics. Unfortunately, it is known to be error-prone (*e.g.*, a recently proposed attack (Wong et al., 2020) can recover model parameters during "oblivious" inference).

A recent trend (*e.g.*, (Wagh et al., 2021)) relies on multiple non-colluding servers to perform SMC for secure training. However, SMC needs multiple rounds of communication. It is thus more desirable to have a dedicated connection among the servers.

Albeit with better utility (than DP-based designs), cryptographic approaches mostly consider immunity against membership inference (Shokri et al., 2017) to be out of their protection scope since DP mechanisms could be applied over the training data before the cryptographic processing.

There is a growing interest in privacy-preserving analytics in the NLP community too. Very recently, TextHide (Huang et al., 2020) devises an "encryption" layer for the hidden representations. Unfortunately, it is shown to be insecure by cryptographers and privacy researchers Carlini et al. (2020a).

Hardware-Aided Approaches. GPU can compute linear operations in a batch much faster than CPU. Nevertheless, we still need a protection mechanism in using GPU, another protection mechanism for the non-linear operations, and their secure integration. In general, utilizing GPU for privacy-preserving machine-learning computations is non-trivial (*e.g.*, see (Ng and Chow, 2021) for an extended discussion).

To exploit the parallelism of GPU while minimizing the use of cryptography, one can resort to a trusted processor (*e.g.*, Intel SGX) for performing non-linear operations within its trusted execution environment (TEE) Note that one still needs to use cryptographic protocols to outsource the linear computation to (untrusted) GPU. Slalom (Tramèr and Boneh, 2019) is such a solution that supports privacy-preserving inference. Training is a more challenging task that was left as an open challenge. Recently, it is solved by Goten (Ng et al., 2021). Notably, both works are from cryptographers but also get recognized by the AI community.

Finally, we remark that the use of TEE is not a must in GPU-enabled solutions. For example, GForce (Ng and Chow, 2021) is one of the pioneering works that proposes GPU-friendly protocols for non-linear layers with other contributions.

Synthesizing Adversarial Negative Responses for Robust Response Ranking and Evaluation

Prakhar Gupta♣ Yulia Tsvetkov♠ Jeffrey P. Bigham♣,♡
♣Language Technologies Institute, Carnegie Mellon University
♠Paul G. Allen School of Computer Science & Engineering, University of Washington
♡Human-Computer Interaction Institute, Carnegie Mellon University
`prakharg@cs.cmu.edu, yuliats@cs.washington.edu, jbigham@cs.cmu.edu`

Abstract

Open-domain neural dialogue models have achieved high performance in response ranking and evaluation tasks. These tasks are formulated as a binary classification of responses given in a dialogue context, and models generally learn to make predictions based on context-response content similarity. However, over-reliance on content similarity makes the models less sensitive to the presence of inconsistencies, incorrect time expressions and other factors important for response appropriateness and coherence. We propose approaches for automatically creating adversarial negative training data to help ranking and evaluation models learn features beyond content similarity. We propose mask-and-fill and keyword-guided approaches that generate negative examples for training more robust dialogue systems. These generated adversarial responses have high content similarity with the contexts but are either incoherent, inappropriate or not fluent. Our approaches are fully data-driven and can be easily incorporated in existing models and datasets. Experiments on classification, ranking and evaluation tasks across multiple datasets demonstrate that our approaches outperform strong baselines in providing informative negative examples for training dialogue systems.[1]

1 Introduction

Due to growing availability of dialogue corpora (Li et al., 2017; Zhang et al., 2018; Smith et al., 2020) and the advancement of neural architectures (Radford et al., 2019; Brown et al., 2020; Devlin et al., 2019), dialogue systems have achieved considerable success. As typically formulated, dialogue models generate one or more candidate responses to a provided context, consisting of past dialogue turns. Dialogue ranking (Zhou et al., 2018; Wu et al., 2019) and evaluation models (Tao et al., 2018; Yi et al., 2019; Sato et al., 2020), in turn, are deployed to select and score candidate responses according to coherence and appropriateness.

Ranking and evaluation models are generally trained using true positive responses and randomly selected negative responses, which raises two issues. First, random negative candidates often have low content similarity with the context, and thus models learn to associate response coherence and appropriateness with content similarity (Yuan et al., 2019; Whang et al., 2021; Sai et al., 2020). In real systems, generated response candidates tend to be more similar in terms of content, and so other factors (e.g., time expressions, dialogue acts, inconsistencies) tend to be more important. Second, randomly selecting candidates as negative examples in an open domain context can result in false negatives, leading to misclassification of appropriate responses.

To make dialogue models more robust to the spurious pattern of content similarity, prior work proposed to leverage adversarial and counterfactual examples (Kaushik et al., 2020; Srivastava et al., 2020). A reliable method for creating counterfactual data is to collect human-written adversarial negative responses (Sai et al., 2020), but it is expensive, time-consuming, and difficult to scale. Our goal is to create reliable automatic methods for *synthesizing* adversarial negative responses.

The most common approach to generating natural language adversarial examples is to paraphrase or insert typos, synonyms, or words relevant to the context in the inputs (Iyyer et al., 2018; Ebrahimi et al., 2018; Alzantot et al., 2018; Zhang et al., 2019). In open domain conversations, however, a context can have a wide range of possible responses with varied forms and semantics. Small lexical

---

[1] Code and data are publicly available `https://github.com/prakharguptaz/Adv_gen_dialogue`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3867–3883
August 1–6, 2021. ©2021 Association for Computational Linguistics

	Error category	Description	Sample responses
C-ent	Incorrect entities or actors (R,G)	Incorrect subject or object of verbs or presence of one or more incorrect entities or coreference.	*Context:* I am so happy that you are doing okay. *Response:* My friend is always happy.
C-time	Incorrect Time expressions (R)	Use of incorrect time expressions or tense of verbs.	*Context:* What are you going to do on Monday? *Response:* Yesterday, I celebrated my daughter's wedding anniversary.
C-cont	Contradictory or extraneous details (R,G)	Presence of details which make the response inconsistent within itself or contradict the context	*Context:* A: I don't know why I bothered to come here. B: Did you enjoy your stay? *Response:* I enjoyed the concert a lot.
C-speaker	Incorrect speaker turn (R)	The response is relevant to the conversation but from the wrong speaker.	*Context:* What starting salary would you expect here? *Response:* If you work overtime, I will pay you extra salary.
C-follow	Does not directly address the context (R,G)	The response does not follow immediately from the context.	*Context:* What would you like for main course sir? *Response:* I know very well how to make noodles, and I taught one of my friends.
C-strat	Incorrect strategies (R,G)	Use of incorrect dialogue act, emotion, persona or style	*Context:* I can't find the paper clips. *Response:* Ok, great work.
C-lang	Poor language (G)	Presence of poor grammar, incorrect sentence structures or repetitions	*Context:* Do you have mixed drinks available here? *Response:* Yes. This order is divided by 16 divided for main main ones of order.

Table 1: Error categories prevalent in inappropriate responses with high context-response semantic relatedness. We present 7 categories with their descriptions and sample context and response pairs. For each category we also indicate whether it is frequently observed in Retrieval (R) or Generation (G) models. Models which simply learn to associate response coherence with content similarity often ignore these errors. Our approaches create adversarial negative data for training dialogue models by introducing such errors in context relevant utterances.

variations via substitutions and paraphrasing do not provide adequate coverage over the possible space of adversarial responses, and they can also lead to generation of false negatives due to the open-ended nature of dialogues. Creating adversarial dialogue responses is thus different, and can be more challenging than in other natural language domains.

We propose two approaches for adversarial response creation: 1) a mask-and-fill approach that corrupts gold responses related to the context but retains content similarity, and 2) a keyword-guided generative approach that uses concepts from the context to generate topically relevant but incoherent responses. These approaches do not require additional annotations, are black-box (do not need access to model parameters), and are easily adapted to new datasets and domains.

The main contributions of this paper are: 1) We identify and discuss error patterns present in retrieval and generation model outputs, which are difficult to detect due to high content similarity; 2) To the best of our knowledge, we are the first to propose automatic approaches for creating adversarial responses for dialogue model training in a black-box setting; and, 3) We demonstrate that our proposed approaches achieve better performance compared to strong baselines on two datasets on dialogue classification, ranking and evaluation tasks.

2 Properties of Adversarial Responses

Models trained using randomly sampled negative examples tend to assign high scores to responses with high content similarity with the context, and often ignore other important factors necessary for response appropriateness and coherence. Therefore, we aim to generate adversarial negative responses which have high content similarity with the context, but which still possess factors rendering the responses inappropriate to the context. We present the categorization of such factors or error types which can make a response inappropriate in Table 1. For each category, we provide its description and sample context-response pairs. To create this categorization, we manually analyzed responses present in outputs of generative models, candidates of retrieval sets, and human written adversarial dialogue responses (Sai et al., 2020). Categories C-ent, C-time and C-cont are errors related to various inconsistencies and logical flaws in the responses and indicate poor response *appropriateness*. Categories C-speaker, C-follow and C-strat are error types specific to the dialogue setting and indicate poor response *coherence*. Category C-lang indicates poor response *fluency*. Our categorization of errors is inspired by the categorization suggested by Pagnoni et al. (2021) for factuality of summarization, and Higashinaka et al. (2019); Ko et al.

(2019) and Sato et al. (2020) for dialogue. These categories inform our approaches as well as error analysis.

3 Methodology

For a given dialogue context C and its gold response R_g, our goal is to generate an adversarial response R_a such that while achieving high scores from dialogue ranking or evaluation models, it should not be a valid response to the context C. Dialogue ranking and evaluation models trained with such hard synthetic negative responses should learn to associate response relevance with features beyond content similarity, and hence become robust against spurious features.

The adversarial responses should satisfy the following criteria: 1) have high content similarity with input contexts; 2) have one or more errors (Table 1) which make the response inappropriate to the context; 3) be hard training examples, that is, they should likely be misclassified by current models as correct; and 4) sufficiently cover errors which occur naturally in model generated responses and retrieval candidates, and therefore they should be plausible and diverse. We propose two approaches for synthesizing adversarial negative examples - a mask-and-fill approach and a keyword-guided generation approach which we discuss next.

3.1 Mask-and-fill Approach

This approach modifies and corrupts original utterances related to a context as shown in Figure 1. It consists of two steps: 1) masking, where one or more tokens of an original utterance are masked out; and 2) infilling, where the masked out tokens are substituted with new tokens. For a context C, the set of original utterances consists of:

- Set of ground truth responses of the context - R_g.
- Set of utterances from the context - U_c.
- Set of retrieved responses based on context - R_e.

Masking: We use the hierarchical masking function from Donahue et al. (2020) which selectively masks spans at the granularities of words, n-grams, and sentences. We apply the masking function to each utterance multiple times to get up to 3 masked versions per utterance. Each utterance is constrained to have at least two masked spans. The spans are selected randomly for masking following Donahue et al. (2020).

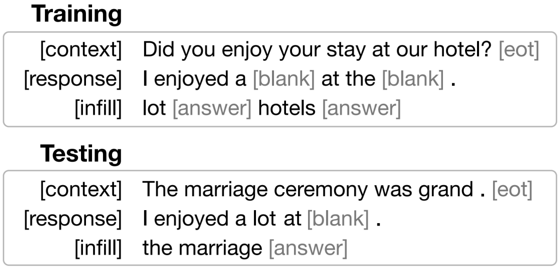

Training

[context] Did you enjoy your stay at our hotel? [eot]
[response] I enjoyed a [blank] at the [blank] .
[infill] lot [answer] hotels [answer]

Testing

[context] The marriage ceremony was grand . [eot]
[response] I enjoyed a lot at [blank] .
[infill] the marriage [answer]

Figure 1: *Mask-and-fill* approach using ILM model. ILM is trained to infill n-grams in place of blanks in a response. Tokens after [infill] replace the [blank] tokens. During training, *Mask-and-fill* learns to infill responses conditioned on the correct context. During testing, it infills the response conditioned on a random context which introduces errors in the response.

Infilling: We extend the Infilling Language Model (ILM) from Donahue et al. (2020) for dialogue response infilling (Figure 1). The ILM model is a GPT-2 (Radford et al., 2019) based language model. For any piece of text t with some spans masked with [blank] tokens, it is trained to predict the blanked spans in t as a sequence generation problem. Each blank is infilled with an n-gram which can consist of one or more tokens. For generating adversarial responses, infilling is done by conditioning on random contexts C_{rand} instead of the original context C to introduce various categories of errors (Table 1). For example in Figure 1, conditioning on a random context leads to the infilling of “the marriage” in the response, introducing error of type C-ent. For the context “Did you stay your stay at our hotel?” it generates a response “I enjoyed at lot at the marriage”. By corrupting the three types of utterances R_g, U_c and R_e, this approach is able to introduce errors covering the 7 categories in Table 1.

Preventing false negatives: Accidentally incorporating false negatives during training can lead to the model learning to misclassify appropriate responses. However due to the open-ended nature of dialogue responses, preventing generation of false negatives is not trivial. In addition to conditioning on random contexts, we incorporate the following mechanisms during infilling to further reduce false negative generation:

- *Semantics of substitution*: We only select token substitutions which were not present in the tokens which were blanked. We also lower the generation probability of the blanked tokens’ top 10 related words based on GloVe embedding (Pennington et al., 2014) similarity by a factor of 100. This ensures that the blanks are not infilled by the originally blanked tokens or any related words.
- *Degree of substitution* - To ensure that the gen-

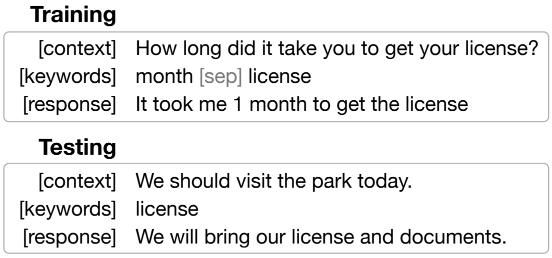

Training

[context]	How long did it take you to get your license?
[keywords]	month [sep] license
[response]	It took me 1 month to get the license

Testing

[context]	We should visit the park today.
[keywords]	license
[response]	We will bring our license and documents.

Figure 2: Keyword-guided approach for adversarial response generation. During training, the model learns to generate a response conditioned on its keywords and the correct context. During testing, it generates the response conditioned on a random context and keywords extracted from the correct context. The generated response thus shares content with the test context but does not directly address the context.

erated negative response is sufficiently different from the original utterance, we filter out the original utterance if the number of words in the utterance after stop-word removal is less than 2. We also filter a generated response if the difference in count of non stop-words between the original and generated response is less than 2.

Improving fluency: The ILM model often generates responses with poor grammar or structure. To improve the fluency of the adversarial response sets, we first generate up to 4 different infilled variations of the masked original utterances, then score them using a GPT-2 based scorer named lm-scorer[2]. We then select the desired number of responses from this larger set.

3.2 Keyword-guided Approach

This approach generates adversarial responses using keywords from the context as guidance, as shown in Figure 2. The base generative architecture is a GPT-2 based dialogue model and it is trained to generate responses conditioned on the context and the response keywords. For adversarial response generation, the generation is conditioned on a random context C_{rand} and keywords from the test context C. In Figure 2, for the context "How long did it take you to get your license?" it generates a response "We will bring our license and documents." To create the keyword set K for a response, the model selects n number of keywords randomly from the set of all keywords extracted from the context C, where n is chosen randomly between 1 to 3 for every context. Keyword extraction is performed using Rake (Rose et al., 2010). We call this model *Key-context*. Since the generation is conditioned on keywords from context C, the generated response shares some content and semantics with the test context. However, since it is also conditioned on a random context C_{rand}, the generated response also incorporates entities, time expressions, speaker role, dialogue act, and other details based on C_{rand}. Since the generation model is not perfect, it also introduces errors related to fluency. Hence, the model is able to introduce errors covering the 7 categories in Table 1.

Key-context only uses keywords from the context to induce content similarity with the context. However, responses can have high content similarity due to the presence of similar concepts rather than just keywords. To introduce content similarity at concept level, we expand the keyword set K with their top 10 most related words based on their GloVe embeddings. We use the gensim library[3] to find the most related words. For example, the related words for the keyword "christmas" are "holidays" and "easter". We replace a keyword in keyword set K with one of its related words with a probability of 0.5. We call this variant *Key-sem*.

3.3 Classification Model

Our classification model architecture is based on the Speaker-Aware Bert (SA-Bert) model (Gu et al., 2020). Given a dialogue context $C = \{C_1, C_2, \ldots, C_h\}$ with C_k denoting k_{th} utterance in the context, a response r and a label $y \in \{0, 1\}$, the goal of the dialogue model M is to learn a score $s(C, r)$ by minimizing cross-entropy loss function for the binary classification task. To calculate $s(C, r)$, C and r are concatenated, with a prepended [CLS] token. The output vector $\mathbf{E}_{[CLS]} \in \mathbb{R}^H$ for the [CLS] token is used as the aggregated representation for the context-response pair classification. The final prediction is made as $\hat{y} = softmax(\mathbf{W}\mathbf{E}_{[CLS]})$, where $\mathbf{W} \in \mathbb{R}^{2 \times H}$. SA-Bert model incorporates speaker information in two ways. First, an additional speaker embedding is added to the token representations which indicates the speaker's identity for each utterance. Second, a [EOT] token is added at the end of each speaker turn. Before fine-tuning Bert model on the classification task, we first adapt Bert to the dataset by using the standard masked language model objective (Devlin et al., 2019).

---

[2] `https://github.com/simonepri/lm-scorer`

[3] `https://radimrehurek.com/gensim/`

4 Experiments

We test our approaches and baselines on dialogue classification, ranking and evaluation tasks.

4.1 Training Details

We use the base-uncased checkpoints for BERT (Devlin et al., 2019) and ELECTRA (Clark et al., 2020) from the Hugging Face transformers library (Wolf et al., 2020). We trained the models with maximum sequence length of 128, maximum number of training epochs set to 3, Adam optimizer with initial learning rate of 5e-5 with linear decay, batch size of 60 per GPU on machines with 4 Nvidia 2080Ti GPUs. For generation, we use temperature of 0.9, nucleus sampling with p equal to 0.9 and minimum length of 5. We repeat each experiment three times (five times for BERT-based models) with different random seeds, use the validation split to select the best model, and report the mean metric values. Validation was done every 200 batches.

4.2 Experimental Setup

4.2.1 Datasets

We use two open-domain dialogue datasets: *DailyDialog++* (Sai et al., 2020) and PersonaChat (Zhang et al., 2018). DailyDialog++ consists of 16900 dialogue contexts in train set, 1028 in validation set and 1142 in the test set. Each context contains 5 positive responses and 5 random negative responses. It also contains 5 adversarial responses per context collected through crowdsourcing where annotators were instructed to create negative responses with high content similarity with the context. A subset of 9259 out of the 16900 training contexts have 5 human-written adversarial negative responses. It has two test sets, adversarial test set and random test set, based on the type of the negative response. *PersonaChat* dataset (Zhang et al., 2018) is a corpus of human-human persona-conditioned conversations consisting of 8938 dialogues in the train set. We sample 2 random context-response pairs from each dialogue with a total of 17876 contexts for training. We prepend the persona utterances to the dialogue contexts in our experiments. Since there is no human-created adversarial test set available for PersonaChat dataset, we construct an artificial adversarial dataset by randomly selecting an utterance from the dialog context and inserting it in the set of candidate responses following Jia and Liang (2017) and Whang et al. (2021). The adversarial test set for each context consists of the ground truth response, one utterance selected from the dialog context, and 8 random negative responses. The random test set consists of 9 random negative responses.

4.2.2 Metrics

For classification task, we report the accuracy following (Sai et al., 2020). For ranking task, we report standard ranking metrics - Recall $R_n@k$ and mean reciprocal rank (MRR). For DailyDialog++, n is 6 in Recall as candidates consist of one positive response with 5 negative responses. For PersonaChat, n is 10. For both classification and ranking tasks, we report results separately for the adversarial and the random test sets.

The dialogue evaluation task comprises of scoring or rating a response for its quality. For this task, we report the correlation of model scores with human provided ratings. We leverage the human ratings released by the following sources: 1) 600 ratings for response "sensibility" from (Zhao and Kawahara, 2020) with inter-rater agreement > 0.6 (Krippendorff's α (Krippendorff, 2018)). The responses consist of outputs from hierarchical recurrent encoder decoder (HRED) model with Attention (Serban et al., 2016) and Variational HRED model with attention (Serban et al., 2017); 2) 700 ratings for response quality from (Zhao et al., 2020). The responses are from 6 different generative models - Seq-2-Seq (Sutskever et al., 2014), attentional Seq-2-Seq, HRED, VHRED, GPT2-small, and GPT2-medium (Wolf et al., 2019) with greedy decoding, ancestral sampling, and nucleus sampling based decoding (Holtzman et al., 2020). The inter-rater agreement is 0.815 (Krippendorff's α), and 3) Since the first two sources do not cover retrieval model outputs, we additionally collect quality ratings for 100 responses from a retrieval model's (Poly-Encoder (Humeau et al., 2020)) selected responses and 100 human written responses with moderate inter-annotator agreement (Cohen's Kappa 0.45 (Cohen, 1968)). All data points belong to the Dailydialog dataset and ratings are scaled between 0–1. By combining these sources we have a total of 1500 ratings for different context-response pairs.

4.2.3 Baselines

We compare the following approaches of creating adversarial negative response sets.

Model	Approach	Adversarial test set			Random test set		
		Accuracy	R@1	MRR	Accuracy	R@1	MRR
Poly-encoder	Random	-	0.684	0.806	-	**0.849**	**0.914**
	Mask-and-fill (Ours)	-	0.758	0.856	-	0.821	0.897
	Key-sem (Ours)	-	**0.788**	**0.877**	-	0.828	0.902
	Human	-	0.847	0.913	-	0.831	0.902
Electra	Random	77.74	0.915	0.748	89.58	0.957	**0.927**
	Mask-and-fill (Ours)	**87.24**	0.945	**0.893**	**89.61**	**0.959**	**0.927**
	Key-sem (Ours)	86.24	**0.951**	0.881	89.47	0.957	0.924
	Human	91.94	0.984	0.967	87.95	0.944	0.911
Bert	Random	77.82	0.906	0.742	**89.34**	**0.959**	**0.923**
	Semi-hard (Li et al., 2019)	79.05	0.913	0.756	89.32	0.956	**0.923**
	Token-subs (Kryscinski et al., 2020)	77.23	0.901	0.783	88.60	0.950	0.906
	BM25 (Karpukhin et al., 2020)	84.42	0.936	0.872	87.68	0.948	0.902
	Mask-and-fill (Ours)	**87.45**	**0.946**	**0.904**	88.32	0.951	0.918
	Key-context (Ours)	86.23	0.939	0.891	88.16	0.953	0.922
	Key-sem (Ours)	87.02	0.944	0.897	89.31	0.954	0.916
	Human (Sai et al., 2020)	91.22	0.987	0.973	88.04	0.943	0.901

Table 2: Performance on classification and ranking tasks on DailyDialog++ test sets. Mask-and-fill and Key-sem approaches consistently perform the best across all model architectures compared to baselines on the Adversarial test set, just short of models trained with human created adversarial data. Poly-encoder's accuracy is not available as it ranks candidates relative to each other.

Human (Sai et al., 2020) Human written adversarial responses.

Random Responses sampled from random contexts.

Semi-hard (Li et al., 2019) Sampling scheme which selects samples from a batch based on their similarity scores with a margin of α from the positive response score. We perform static sampling and use Sentence-Bert (Reimers and Gurevych, 2019) for semantic similarity calculation with α set to the recommended value of 0.07.

Token-subs (Kryscinski et al., 2020) Training data is generated by applying a series of rule-based transformations on the positive responses. Transformations include pronoun, entity and number swapping, sentence negation and noise injection.

BM25 Top responses returned by BM25 (Robertson and Zaragoza, 2009) based on similarity with the context. Any ground truth response is removed from this response set if present by chance. This baseline is inspired from Karpukhin et al. (2020) and Lin et al. (2020) and has shown strong performance in passage and response retrieval.

Mask-and-fill Our approach that infills utterances conditioned on random contexts.

Key-context Our approach that generates responses conditioned on test context keywords and random context history.

Key-sem Our approach similar to Key-context which additionally conditions on words semantically related to the keywords in the context.

For each context, adversarial train sets are created by adding 5 random negative responses to the set of 5 negative responses created from the above approaches. If an approach create more than 5 responses, we randomly select 5 from them.

For dialogue evaluation, we compare the above approaches with BLEU, METEOR (Banerjee and Lavie, 2005), embedding based metrics SkipThought (Kiros et al., 2015), Vec Extrema (Forgues et al., 2014), and RUBER (Tao et al., 2018) and BERTScore (Zhang et al., 2020a).

4.2.4 Models

We experiment with following architectures for ranking and evaluation models in our experiments: 1) Bert (Devlin et al., 2019). We use the SA-Bert model (Gu et al., 2020), 2) Electra (Clark et al., 2020), pre-trained with a replaced token detection objective and employs a generator-discriminator framework, and 3) Poly-encoders (Humeau et al., 2020), allows for fast real-time inference by precomputing each candidate response representation once, and then ranking candidate responses for retrieval by attending to the context.

4.3 Results and Discussion

In this section, we compare the performance of our approaches with the baselines on dialogue classification, ranking and evaluation tasks.

Performance on classification Our proposed approaches Mask-and-fill and Key-sem achieve the highest classification accuracy on the adversarial test set (Table 2), a few percentage short of the Human baseline. The closest baseline is BM25 which has a gap of 3% in accuracy compared to our

Approach	Adversarial test set		Random test set	
	R@1	MRR	R@1	MRR
Random	0.905	0.820	0.963	**0.914**
Semi-hard	0.906	0.820	**0.964**	0.913
Token-subs	0.895	0.825	0.958	0.901
BM25	0.925	0.859	0.940	0.874
Mask-and-fill (Ours)	**0.933**	**0.871**	0.952	0.890
Key-sem (Ours)	0.920	0.856	0.947	0.884

Table 3: Performance on ranking task on PersonaChat dataset with Bert architecture. Our approaches perform better than all baselines on the adversarial test set.

Approach	Pearson	Spearman
BLEU-2	0.046	0.004
METEOR (Banerjee and Lavie, 2005)	0.081	0.007
SkipThought (Kiros et al., 2015)	0.059	0.069
Vec Extrema (Forgues et al., 2014)	0.157	0.150
BERTScore (Zhang et al., 2020a)	0.208	0.198
RUBER (Tao et al., 2018)	0.253	0.282
Random	0.296	0.313
Semi-hard (Li et al., 2019)	0.299	0.315
BM25 (Karpukhin et al., 2020)	0.310	0.350
Token-subs (Kryscinski et al., 2020)	0.324	0.388
Mask-and-fill (Ours)	0.338	0.361
Key-sem (Ours)	**0.382**	**0.401**
Human (Sai et al., 2020)	0.348	0.371

Table 4: Comparison of approaches on dialogue evaluation. Trainable metrics are based on Bert architecture. For all entries except for the ones underlined, t-test *p-value* < 0.05. Mask-and-fill and Key-sem perform better than all baselines including the Human baseline.

approaches. Token-subs, which applies transformations on positive responses to corrupt them, does not fair well on this task. This indicates that simple transformations do not provide good coverage of semantic variations present in the adversarial test responses. Our approaches achieve similar performance across different model architectures, demonstrating their generalizability. Unsurprisingly, the Human baseline performs strongly as the training and test data were created in the same manner and have similar distributions. On the random test set, the performance of all approaches is either very close or lower than the Random baseline. Since the similarity between correct responses and the context is generally a lot higher than between random responses and the context in the random test set, Random baseline performs better since it associates coherence mostly with semantic similarity. Finally, our analysis shows that all baselines tend to assign low scores to valid responses which do not address a context directly. For example, for the context "Will you join us for the concert?", if the response is "It is supposed to rain this week.", models assign it a low score. Such scenarios require understanding of social and commonsense related factors. We leave addressing this limitation to future work.

Performance on ranking On the DailyDialog adversarial test set, Mask-and-fill and Key-sem approaches achieve the best Recall and MRR, closely followed by BM25 baselines (Table 2). The trends of the ranking metrics are similar to those observed for accuracy metrics. Our approaches perform better than the Human baseline on the random test set. On PersonaChat dataset, Mask-and-fill and Key-sem perform better than the baselines (Table 3), especially on the adversarial test set. This demonstrates the extensibility of our approach across datasets. Mask-and-fill performs better than Key-sem as the keyword sets contain a lot of keywords from the persona because of which responses have high content similarity with the persona rather than with the context. The poor performance of the Random baseline provides evidence that training models using random negative candidates does not make the models robust against hard test cases during testing. BM25 is a strong baseline for both datasets since retrieved responses also provide coverage over errors of various categories. However, retrieved response quality and diversity depends on the size of the retrieval pool. Furthermore, a stronger retrieval mechanism can lead to higher false negatives. While the variation in BM25 response sets is constraint by the size of the dataset, and they provide lesser coverage over categories C-cont, C-strat and C-lang (Table 1), our approaches have no such constraints.

Performance on dialogue evaluation To study the performance of various approaches on real systems, we compare them on the task of Dialogue evaluation or scoring. We measure the correlation between the scores predicted by the approaches in Table 4 with human provided ratings. Reference based metrics like BLEU-2, METEOR, SkipThought and Vec Extrema achieve very low correlations, similar to findings reported in prior art (Liu et al., 2016; Gupta et al., 2019). BERTScore and RUBER achieve moderate correlation. Our approach Key-sem achieves the best correlations, followed by Mask-and-fill. BM25's performance is lower than that of our approaches, but it is higher than the Random and Semi-hard approaches. Although Token-subs did not achieve high performance on the classification and ranking tasks, it performs well on this task. This is likely because real model outputs contains more of

Context	A: Julia, will you be my wife? B: I'm sorry, Steven. C: Please, Julia, I have made proposal to you five times . I really want to share the rest of my life with you.
Random	(1) Yes of course it's a promise. (2) It's better to go somewhere else. (3) Let me first look at your work, how you have done it. (4) Being in love is a deep experience while having a crush is shallow. (5) Sometimes I don't understand, what is your problem?
Mask-and-fill	(1) You can't force me for to do that. They are designed for people of all ages and religions. (2) There you are. I'll have to make my own lunch! (3) I majored in economics. I really want i hope i can get some practical experience in life with you. (4) We will go to, and to meet some of the children who are visiting at school. (5) It takes time to learn. Bless you, baby!
Key-sem	(1) And what about the potatoes? Steven, i don't know. (2) Sorry, there is no problem. (3) Your wife didn't like it. Please don't tell me she is really interested in gardening. (4) I really want to go inside. It's really cold outside. (5) Really? I really want to pay a visit. I really want to spend the rest of my time enjoying this meal.
Human	(1) I want to finish my home work by five and then I am going to take rest. (2) Follow these five tips, and you'll write a winning project proposal every time. (3) I met my wife a three to four times before the marriage. (4) Its difficult to live a life in a Dorze tribal area. (5) I shared a large number of ideas with the wedding planner.

Table 5: Sample adversarial responses from various approaches. Random responses are sampled from random dialogues. Human written responses are from the DailyDialog++ dataset. Mask-and-fill and Key-sem approaches create responses which are semantically related and yet inappropriate responses to the context.

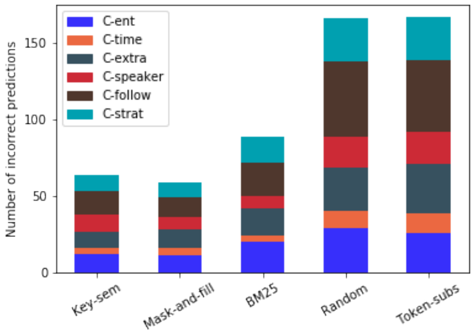

Figure 3: Analysis of error types for different approaches on DailyDialog++ predictions. C-lang error type is not present in DailyDialog++. Mask-and-fill and Key-sem achieve a more uniform distribution over error categories compared to other approaches.

the factual inconsistencies and contradictions that this approach captures, than what the adversarial test sets contain. Key-sem performs better than Mask-and-fill on evaluation since while Mask-and-fill only modifies utterances related to the context, Key-sem can freely generate more diverse adversarial responses for training. Also, Key-sem achieves higher correlation than Human baseline. This may be because it is difficult for humans to create erroneous responses with distributions similar to the ones in model generated or selected responses, especially error types like C-speaker, C-strat and C-lang. In contrast, our approaches provide good coverage over all error types.

Analysis of errors types We analyze the classification outputs of various approaches on the DailyDialog++ adversarial test set and report the types of misclassifications by each approach in Figure 3. We first select a subset of test data where at least one of the approaches misclassifies the adversarial response as positive. We then manually categorize the types of errors presented in Table 1 for 200 randomly selected contexts from this subset. Each response can have multiple error types. C-follow and C-extra are the dominant error types which are misclassified by baselines Random, BM25 and Token-subs. Key-sem and Mask-and-fill approaches achieve improvement in all error types compared to baselines and have a more uniform error distribution. While Key-sem performs better on C-extra, Mask-and-fill is better on C-follow and C-speaker.

Adversarial response examples We present sample responses from our approaches along with Random and Human baseline responses in Table 5. Random approach generates responses which are easily distinguishable from ground truth responses. Mask-and-fill approach modifies either the ground truth response, utterances from the context or BM25 retrieved responses. It modifies these utterances to introduce corruptions such as non-contextual tokens, extraneous entities, incorrect time expressions, affective words or contradictions which makes the response either inappropriate or incoherent to the context, but it remains topically similar to the context. In Key-sem the dialogue acts, some entities and other tokens of the generated response depend on a random context the response is conditioned on, which also makes the response inappropriate or incoherent to the context.

5 Related Work

Dialogue response ranking and evaluation are important tasks in dialogue domain because even the recent large pretrained-language model based architectures (Zhang et al., 2020b; Humeau et al., 2020; Adiwardana et al., 2020; Roller et al., 2021; Gupta et al., 2021) have been shown to be susceptible to creating inconsistent, ungrammatical and incoherent responses (Roller et al., 2021). Traditional word-overlap based metrics like BLEU have been shown to be ineffective for dialogue response scoring (Liu et al., 2016; Gupta et al., 2019). Recently trainable metrics such as ADEM (Lowe et al., 2017), RUBER (Ghazarian et al., 2019) and USR (Mehri and Eskenazi, 2020) have been proposed for these tasks. However, since they are trained using negative samples obtained from random contexts, they are also prone to the spurious pattern of content similarity.

Adversarial or counterfactual data creation techniques have been proposed for applications such as evaluation (Gardner et al., 2020; Madaan et al., 2020), attacks (Ebrahimi et al., 2018; Wallace et al., 2019; Jin et al., 2020), explanations (Goodwin et al., 2020; Ross et al., 2020) or training models to be robust against spurious patterns and biases (Garg et al., 2019; Huang et al., 2020). Adversarial examples are crafted through operations such as adding noisy characters (Ebrahimi et al., 2018; Pruthi et al., 2019), paraphrasing (Iyyer et al., 2018), replacing with synonyms (Alzantot et al., 2018; Jin et al., 2020), rule based token-level transformations (Kryscinski et al., 2020), or inserting words relevant to the context (Zhang et al., 2019). While these approaches are optimized to change the predictions of a target model by perturbing the inputs, our approaches are more general and are not optimized towards any target model. Polyjuice (Wu et al., 2021) and FactCC (Kryscinski et al., 2020) proposed approaches for model-agnostic general-purpose counterfactual generation. These approaches change the model's prediction by creating small edits through substitutions and insertions to the inputs. They are not applicable to our setting where we aim to flip the gold label, that is, convert a valid response to an adversarial response, while the model prediction should ideally remain the same to create hard training examples. Furthermore small perturbations do not provide good coverage over the adversarial response space and can create false negative responses. Adversarial semantic collisions (Song et al., 2020) aims to generate texts that are semantically unrelated but judged as similar by NLP models to expose model vulnerabilities. However, the outputs which are unrelated to the context are not useful for adversarial training as they are easy to classify.

Finally, negative sampling strategies have also been studied for creating hard negative samples in context of visual embeddings (Faghri et al., 2018; Guo et al., 2018), knowledge graphs (Kotnis and Nastase, 2017), document retrieval (Saeidi et al., 2017; Karpukhin et al., 2020) and response retrieval (Li et al., 2019; Lin et al., 2020). In this work we compare and build upon past work and are the first to propose generative approaches for adversarial negative response creation in dialogue.

6 Conclusion

This paper introduces approaches for synthesizing adversarial negative responses for training more robust dialogue response ranking and evaluation models. To synthesize a rich and comprehensive set of responses, we present and analyze categories of errors which affect the models. Our proposed approaches do not require any manual annotation and achieve high performance in dialogue classification, ranking and evaluation tasks across two datasets. These results demonstrate the promise of synthetic negative examples for improving open domain dialogue. Future work, we will explore synthesizing adversarial test sets and methods for finer grained, controlled adversarial response generation.

Acknowledgements

We thank Amy Pavel, Alissa Ostapenko, Rishabh Joshi, Artidoro Pagnoni and the anonymous reviewers for providing valuable feedback. This work was funded by the Defense Advanced Research Planning Agency (DARPA) under DARPA Grant N6600198-18908, and the National Science Foundation under Awards No. IIS1816012 and IIS2007960. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

D. Adiwardana, Minh-Thang Luong, D. So, J. Hall, Noah Fiedel, R. Thoppilan, Z. Yang, Apoorv Kulshreshtha, G. Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. *ArXiv*, abs/2001.09977.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1866–1875, Hong Kong, China. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Jacob Cohen. 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *NeurIPS, modern machine learning and natural language processing workshop*, volume 2.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H. Chi, and Alex Beutel. 2019. Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 219–226, New York, NY, USA. Association for Computing Machinery.

Sarik Ghazarian, Johnny Wei, Aram Galstyan, and Nanyun Peng. 2019. Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 82–89, Minneapolis, Minnesota. Association for Computational Linguistics.

Emily Goodwin, Koustuv Sinha, and Timothy J. O'Donnell. 2020. Probing linguistic systematicity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1958–1969, Online. Association for Computational Linguistics.

Jia-Chen Gu, Tianda Li, Quan Liu, Zhen-Hua Ling, Zhiming Su, Si Wei, and Xiaodan Zhu. 2020. *Speaker-Aware BERT for Multi-Turn Response Selection in Retrieval-Based Chatbots*, page 2041–2044. Association for Computing Machinery, New York, NY, USA.

Guibing Guo, Songlin Zhai, Fajie Yuan, Yuan Liu, and Xingwei Wang. 2018. Vse-ens: Visual-semantic embeddings with efficient negative sampling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Prakhar Gupta, Jeffrey Bigham, Yulia Tsvetkov, and Amy Pavel. 2021. Controlling dialogue generation with semantic exemplars. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3018–3029, Online. Association for Computational Linguistics.

Prakhar Gupta, Shikib Mehri, Tiancheng Zhao, Amy Pavel, Maxine Eskenazi, and Jeffrey Bigham. 2019. Investigating evaluation of open-domain dialogue systems with human generated multiple references. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 379–391, Stockholm, Sweden. Association for Computational Linguistics.

Ryuichiro Higashinaka, Masahiro Araki, Hiroshi Tsukahara, and Masahiro Mizukami. 2019. Improving taxonomy of errors in chat-oriented dialogue systems. In *9th International Workshop on Spoken Dialogue System Technology*, pages 331–343. Springer.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *International Conference on Learning Representations*.

Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. 2020. Reducing sentiment bias in language models via counterfactual evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 65–83, Online. Association for Computational Linguistics.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *8th International Conference on Learning Representations, ICLR*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. *International Conference on Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3294–3302, Cambridge, MA, USA. MIT Press.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019. Linguistically-informed specificity and semantic plausibility for dialogue generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3456–3466, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhushan Kotnis and Vivi Nastase. 2017. Analysis of the impact of negative sampling on link prediction in knowledge graphs. *arXiv preprint arXiv:1708.06816*.

Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Jia Li, Chongyang Tao, Wei Wu, Yansong Feng, Dongyan Zhao, and Rui Yan. 2019. Sampling matters! an empirical study of negative sampling strategies for learning of matching models in retrieval-based dialogue systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1291–1296, Hong Kong, China. Association for Computational Linguistics.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Zibo Lin, Deng Cai, Yan Wang, Xiaojiang Liu, Haitao Zheng, and Shuming Shi. 2020. The world is not binary: Learning to rank with grayscale data for dialogue response selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9220–9229, Online. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126, Vancouver, Canada. Association for Computational Linguistics.

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. 2020. Generate your counterfactuals: Towards controlled counterfactual generation for text. *arXiv preprint arXiv:2012.04698*.

Shikib Mehri and Maxine Eskenazi. 2020. USR: An unsupervised and reference free evaluation metric for dialog generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 681–707, Online. Association for Computational Linguistics.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Alexis Ross, Ana Marasović, and Matthew E Peters. 2020. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*.

Marzieh Saeidi, Ritwik Kulkarni, Theodosia Togia, and Michele Sama. 2017. The effect of negative sampling strategy on capturing semantic similarity in document embeddings. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 1–8, Montpellier, France. Association for Computational Linguistics.

Ananya B Sai, Akash Kumar Mohankumar, Siddhartha Arora, and Mitesh M Khapra. 2020. Improving dialog evaluation with a multi-reference adversarial dataset and large scale pretraining. *Transactions of the Association for Computational Linguistics*, 8:810–827.

Shiki Sato, Reina Akama, Hiroki Ouchi, Jun Suzuki, and Kentaro Inui. 2020. Evaluating dialogue generation systems via response selection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 593–599, Online. Association for Computational Linguistics.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2021–2030, Online. Association for Computational Linguistics.

Congzheng Song, Alexander Rush, and Vitaly Shmatikov. 2020. Adversarial semantic collisions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4198–4210, Online. Association for Computational Linguistics.

Megha Srivastava, Tatsunori Hashimoto, and Percy Liang. 2020. Robustness to spurious correlations via human annotations. In *International Conference on Machine Learning*, pages 9109–9119. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Taesun Whang, Dongyub Lee, Dongsuk Oh, Chanhee Lee, Kijong Han, Dong-hun Lee, and Saebyeok Lee. 2021. Do response selection models really know what's next? utterance manipulation strategies for multi-turn response selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *ArXiv*, abs/1901.08149.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Automated, general-purpose counterfactual generation. *arXiv preprint arXiv:2101.00288*.

Yu Wu, Wei Wu, Chen Xing, Can Xu, Zhoujun Li, and Ming Zhou. 2019. A sequential matching framework for multi-turn response selection in retrieval-based chatbots. *Computational Linguistics*, 45(1):163–197.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Alessandra Cervone, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 65–75, Tokyo, Japan. Association for Computational Linguistics.

Chunyuan Yuan, Wei Zhou, Mingming Li, Shangwen Lv, Fuqing Zhu, Jizhong Han, and Songlin Hu. 2019. Multi-hop selector network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 111–120, Hong Kong, China. Association for Computational Linguistics.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. BERTScore: Evaluating text generation with BERT. In *Proceedings of the International Conference on Learning Representations*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020b. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL, system demonstration*.

Tianyu Zhao and Tatsuya Kawahara. 2020. Multi-referenced training for dialogue response generation. *arXiv preprint arXiv:2009.07117*.

Tianyu Zhao, Divesh Lala, and Tatsuya Kawahara. 2020. Designing precise and robust dialogue response evaluators. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Online. Association for Computational Linguistics.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127, Melbourne, Australia. Association for Computational Linguistics.

A Quality of negative candidates

We perform a human evaluation experiment to test the number of false negative responses created by the different approaches. Three in-house annotators were asked to go through the set of 5 adversarial negative responses from 5 different approaches for 100 randomly selected contexts. They were instructed to report the number of responses which are appropriate responses for the context, which in this case is the number of false negatives. After annotating separately, annotators finally discussed the responses marked as appropriate and aggregated the results. We observe that Human baselines responses had 2, Random baseline had 5, Mask-and-fill had 3, Key-sem had 4 and BM25 had 10 false negative responses in the set of 500 responses (100 contexts, with 5 adversarial responses each). This shows that our approaches do not generate high number of false negatives. BM25 on the other hand leads to a relatively higher number of false negatives which can impede the learning process of the models.

B Experiments with Masking

We experiment with two procedures for masking in the Mask-and-fill approach: 1) Random masking, which masks contiguous chunks of tokens some probability p. We leverage the masking function from (Donahue et al., 2020) which can selectively mask spans at the granularities of words, n-grams, and sentences. 2) Importance masking, which keeps the most important tokens in a response relevant to the context and masks the rest. For Importance masking, we leverage the matching model from (Cai et al., 2019) which is trained to estimate the sequence-level quality $s(q, r)$ of a response r for a given query q. They decompose the sequence level matching score between a context and a response into a set of token-level scores as follows:

$$
\begin{aligned}
s(q, r) &= \mathbf{x}_q^T W^s \mathbf{x}_r \\
&= \mathbf{x}_q^T W^s \sum_{k=1}^{m} \omega_k \left(\mathbf{r}_k + \mathbf{e}_{r_k}\right) \\
&= \sum_{k=1}^{m} \omega_k \mathbf{x}_q^T W^s \left(\mathbf{r}_k + \mathbf{e}_{r_k}\right) \quad = \sum_{k=1}^{m} \omega_k s_k
\end{aligned}
$$

where $s_k = \mathbf{x}_q^T W^s \left(\mathbf{r}_k + \mathbf{e}_{r_k}\right)$, and x_r is the weighted sum of a Bert Transformer encoder outputs r_k as well as their initial vector representations e_k. The importance of each response token k to the context is estimated by s_k. We mask out any token with importance weight ω_k less than the average ω and only retain tokens highly relevant to the context following Cai et al. (2019). In our initial experiments we found that the Importance masking procedure lead to worse performance than Random masking. The adversarial test set accuracy on DailyDialog adversarial test set was 85.43% compared to the 87.45% accuracy using Random masking. Our analyses showed that Importance masking masked out about 50% of the response tokens, and the infills generated by the ILM model were mostly poor in fluency as the number of masked tokens was high. We therefore finally used Random masking for Mask-and-fill.

C Sample Model Generated Responses

In continuation of sample responses presented in Table 5 of the main paper, we present some more sample responses from different approaches along with Random and Human baseline responses in Table 6.

D Additional Implementation Details

For BM25 approach, we use the open source implementation from transformer rankers[4]. The DailyDialog++ dataset contains 16900 dialogue contexts but only 9259 of those have adversarial negative responses for the Human baseline. For the results reported in Table 4, all approaches from Random and below use the Bert architecture and trained using DailyDialog domain data. Additionally, RUBER is also trained on the DailyDialog++ dataset. The approaches above Random in the table do not require training. Each approach predicts a score for the set of 1500 responses created using a set of generative and retrieval models as detailed in section 4.2.2. Sentence-Bert used in Semi-hard sampling scheme is fine-tuned on the datasets used in this paper.

For the Mask-and-fill approach, the model takes in the following sequence of inputs: {[context] C_1 [eot],.., [eot] C_h [response] r-with-blanks [infill] B_1 [answer],.., B_l [answer]}, where $C_{c=1}^{h}$ represents a context with h utterances, r the response and $B_{b=1}^{l}$ are the tokens blanked in the response. [eot] is used to indicate end of turn. To generate a set of 5 adversarial responses in the Mask-and-fill approach, we first create 4 masked versions

---

[4] https://github.com/Guzpenha/transformer_rankers

Context or Model	Utterances
Context	A: OK . Now I'll put the dishes in the sink. B: Thank you . I'll wash them. C: OK . Then I will clean the table.
Random	(1) May I have your name and number, Sir? (2) Then I hope to get the other documents by end of the day. (3) She is very experienced in that area, including all the years in college (4) I am in Computer Science department. (5) Sure, you can talk to him.
Mask-and-fill	(1) Ok. Thank you. But, what are you going to do with him? (2) Uh, it's time to wake up. I will clean it up and then go to bed. (3) Oh, thank you. You have helped a lot. (4) Don't, thank you very much for saying it like that. Only in your opinion do you have to wear proper clothes. (5) Yes, thank you! But, i am not satisfied with the work you've done.
Key-sem	(1) Sorry, he didn't put the dishes on the table. (2) Ok. We'll clean up the room now. I can walk you through the process. (3) Don't forget to wash the dishes and put away the clothes. (4) In my field, i put on quite a few weight -bearing exercise in order to improve my lung capacity. (5) Thank you for your understanding. What are your recipes for tableware?
Human	(1) I just now saw the news that the boat was sinking due to heavy goods. (2) I want to thank my friend because he helped me to wash my dress at school camp. (3) Nowadays, table fans are getting very cheap online. (4) I know that using a facial scrub can make your skin look beautiful, clean and soft. (5) I gifted a sink to my friend for his house warming ceremony.
Context	A: Can you tell me what's my responsibility in this position? B: Yes, of course . You would be responsible for the development of software products. C: I see . This is my advantage.
Random	(1) Okay! That sounds great to me. (2) Well! How much will it cost per kg? (3) Well! You can pay it on monthly or yearly basis, it is upto you. (4) I usually spend those days with my family and it is quite fun you see. (5) What type of games do you like to play?
Mask-and-fill	(1) Yes. Maybe he is just looking for some publicity. You are responsible, too. (2) I see. Then we will all get on our own. (3) That's nice. And i would be willing to take them for that. (4) You also have to work on the meetings to be more focused. I need to add some training. (5) What kind of software do they use now?
Key-sem	(1) Let me see, in your brochure, what kind of promotion you're promising? (2) Tell me about it. What do you think? Will you marry her? (3) Of course. Of course there are many things online. Tell me about it. (4) Yes, i appreciate your cooperation. The development of the l / c is our utmost priority. (5) Thank you. I do want to get him a diamond ring. He's responsible for development of the etv.
Human	(1) Of course, the museum is in the closing stage because of financial issues. (2) I was searching on some websites for the junior engineer position to develop my knowledge in the hardware field. (3) I see, is there any terms and condition that I have to sign for this position in your company? (4) Of course, you must provide me the full details about our company's financial position by today evening. (5) Of course, My friend is very much interested to work in a software company. Can you give him a chance in your company?

Table 6: Outputs from different approaches for negative response set creation. Random responses are unrelated to the contexts. Mask-and-fill and Key-sem approaches create responses which are highly similar to the content of the contexts, and hence the model needs to learn factors important for response coherence and appropriateness such as presence of correct entities, time expressions, strategies and others.

of every utterance related to the context (R_g, U_c and R_e). ILM model then generates 4 infills per masked utterance. Thus each utterance gets 16 different modified versions. All these modified utterances are then ranked using the lm-scorer library and we select the top 5. BM25 similarity is used to create the retrieved response set.

For the Keyword-guided approaches, the model is given as input the context C, keywords from the ground truth response K, and the ground truth response r as shown in Figure 2. Specifically, the model takes in the following sequence of inputs - $\{$[context] C_1 [eot],.., [eot] C_h [keywords] K_1 [sep],..,[sep] K_n [response] $r\}$. For both approaches during training, positive responses and negative responses are interleaved, i.e. each positive response is followed by one random and one adversarial response.

Leveraging Abstract Meaning Representation for Knowledge Base Question Answering

Pavan Kapanipathi,* Ibrahim Abdelaziz,* Srinivas Ravishankar,* Salim Roukos, Alexander Gray, Ramon Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, Mo Yu
IBM Research

Abstract

Knowledge base question answering (KBQA) is an important task in Natural Language Processing. Existing approaches face significant challenges including complex question understanding, necessity for reasoning, and lack of large end-to-end training datasets. In this work, we propose Neuro-Symbolic Question Answering (NSQA), a modular KBQA system, that leverages (1) Abstract Meaning Representation (AMR) parses for task-independent question understanding; (2) a simple yet effective graph transformation approach to convert AMR parses into candidate logical queries that are aligned to the KB; (3) a pipeline-based approach which integrates multiple, reusable modules that are trained specifically for their individual tasks (semantic parser, entity and relationship linkers, and neuro-symbolic reasoner) and do not require end-to-end training data. NSQA achieves state-of-the-art performance on two prominent KBQA datasets based on DBpedia (QALD-9 and LC-QuAD 1.0). Furthermore, our analysis emphasizes that AMR is a powerful tool for KBQA systems.

1 Introduction

Knowledge base question answering (KBQA) is a sub-field within Question Answering with desirable characteristics for real-world applications. KBQA requires a system to answer a natural language question based on facts available in a Knowledge Base (KB) (Zou et al., 2014; Vakulenko et al., 2019; Diefenbach et al., 2020; Abdelaziz et al., 2021). Facts are retrieved from a KB through structured queries (in a query language such as SPARQL), which often contain multiple triples that represent the steps or antecedents required for obtaining the answer. This enables a transparent and self-explanatory form of QA, meaning that intermediate symbolic representations capture some of the steps from natural language question to answer.

With the rise of neural networks in NLP, various KBQA models approach the task in an end-to-end manner. Many of these approaches formulate text-to-query-language as sequence-to-sequence problem, and thus require sufficient examples of paired natural language and target representation pairs. However, labeling large amounts of data for KBQA is challenging, either due to the requirement of expert knowledge (Usbeck et al., 2017), or artifacts introduced during automated creation (Trivedi et al., 2017). Real-world scenarios require solving complex multi-hop questions i.e. secondary unknowns within a main question and questions employing unusual expressions. Pipeline approaches can delegate language understanding to pre-trained semantic parsers, which mitigates the data problem, but are considered to suffer from error propagation. However, the performance of semantic parsers for well-established semantic representations has greatly improved in recent years. Abstract Meaning Representation (AMR) (Banarescu et al., 2013; Dorr et al., 1998) parsers recently reached above 84% F-measure (Bevilacqua et al., 2021), an improvement of over 10 points in the last three years.

In this paper we propose Neuro-Symbolic Question Answering (NSQA), a modular knowledge base question answering system with the following objectives: (a) delegating the complexity of understanding natural language questions to AMR parsers; (b) reducing the need for end-to-end (text-to-SPARQL) training data with a pipeline architecture where each module is trained for its specific sub-task; (c) facilitating the use of an independent reasoner via an intermediate logic form.

---

*Equal contribution, correspondence to Pavan Kapanipathi (kapanipa@us.ibm.com), Ibrahim Abdelaziz (ibrahim.abdelaziz1@ibm.com), Srinivas Ravishankar (srini@ibm.com)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3884–3894
August 1–6, 2021.

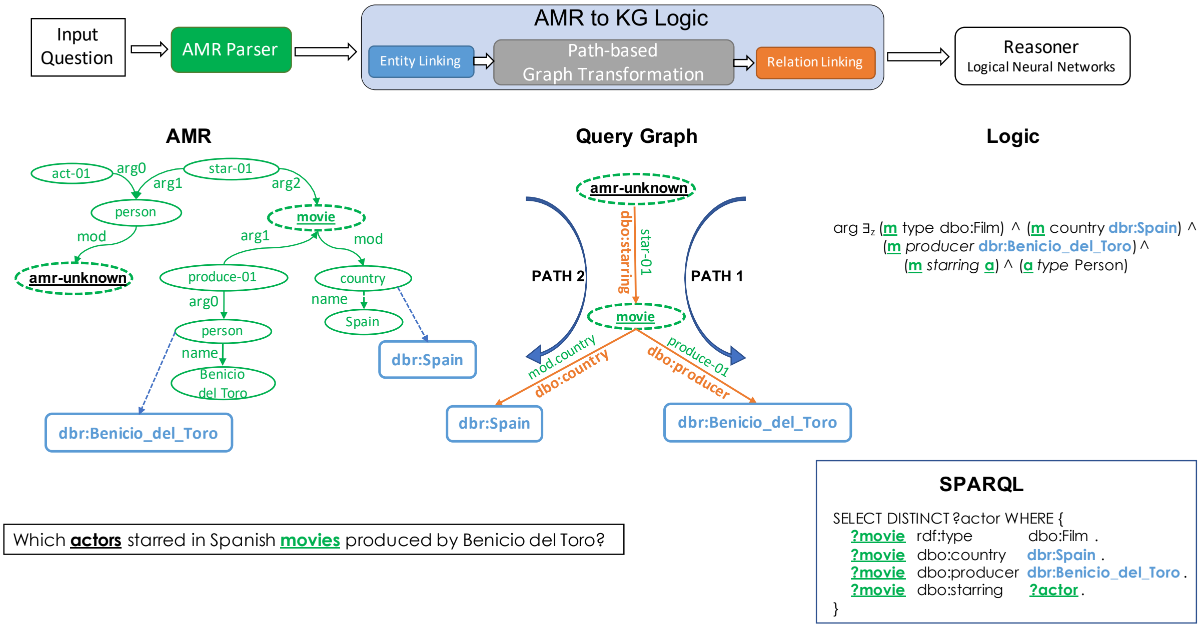

```
SELECT DISTINCT ?actor WHERE {
   ?movie  rdf:type       dbo:Film .
   ?movie  dbo:country    dbr:Spain .
   ?movie  dbo:producer   dbr:Benicio_del_Toro .
   ?movie  dbo:starring   ?actor .
}
```

Figure 1: Real NSQA prediction for the sentence *Which actors starred in Spanish movies produced by Benicio del Toro?*. In underlined, we show the representation for the two unknown variables across all stages including: AMR-aligned tokens in sentence (*Which*, *movies*), AMR graph (*unknown*, *movie*), paths representation (same as AMR), logical representation (*actor*, *movie*) and SPARQL interpretation (*?actor*, *?movie*). Displayed stage outputs: AMR (green), Entity Linking (blue), Relation Linking (orange)

The contributions of this work are as follows:

- The first system to use Abstract Meaning Representation for KBQA achieving state of the art performance on two prominent datasets on DBpedia (QALD-9 and LC-QuAD 1.0).

- A novel, simple yet effective path-based approach that transforms AMR parses into intermediate logical queries that are aligned to the KB. This intermediate logic form facilitates the use of neuro-symbolic reasoners such as Logical Neural Networks (Riegel et al., 2020), paving the way for complex reasoning over knowledge bases.

- A pipeline-based modular approach that integrates multiple, reusable modules that are trained specifically for their individual tasks (e.g. semantic parsing, entity linking, and relationship linking) and hence do not require end-to-end training data.

2 Approach Overview

Figure 1 depicts the pipeline of our NSQA system. Given a question in natural language, NSQA: (i) parses questions into an Abstract Meaning Representation (AMR) graph; (ii) transforms the AMR graph to a set of candidate KB-aligned logical queries, via a novel but simple graph transformation approach; (iii) uses a Logical Neural Network (LNN) (Riegel et al., 2020) to reason over KB facts and produce answers to KB-aligned logical queries. We describe each of these modules in the following sections.

2.1 AMR Parsing

NSQA utilizes AMR parsing to reduce the complexity and noise of natural language questions. An AMR parse is a rooted, directed, acyclic graph. AMR nodes represent concepts, which may include normalized surface symbols, Propbank frames (Kingsbury and Palmer, 2002) as well as other AMR-specific constructs to handle named entities, quantities, dates and other phenomena. Edges in an AMR graph represent the relations between concepts such as standard OntoNotes roles but also AMR specific relations such as polarity or mode.

As shown in Figure 1, AMR provides a representation that is fairly close to the KB representation. A special *amr-unknown* node, indicates the missing concept that represents the answer to the given question. In the example of Figure 1, *amr-unknown* is a *person*, who is the subject of *act-01*. Furthermore, AMR helps identify intermediate variables that behave as secondary unknowns. In this case, a *movie* produced by *Benicio del Toro* in *Spain*.

NSQA utilizes a stack-Transformer transition-based model (Naseem et al., 2019; Astudillo et al., 2020) for AMR parsing. An advantage of transition-based systems is that they provide explicit question text to AMR node alignments. This allows encoding closely integrated text and AMR input to multiple modules (Entity Linking and Relation Linking) that can benefit from this joint input.

2.2 AMR to KG Logic

The core contribution of this work is our next step where the AMR of the question is transformed to a query graph aligned with the underlying knowledge graph. We formalize the two graphs as follows:

AMR graph $\mathcal{G}$ is a rooted edge-labeled directed acyclic graph $\langle \mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G} \rangle$. The edge set $\mathcal{E}_\mathcal{G}$ consists of non-core roles, quantifiers, and modifiers. The vertex set $\mathcal{V}_\mathcal{G} \in$ *amr-unknown* $\cup \mathcal{A}_\mathcal{P} \cup \mathcal{A}_\mathcal{C}$ where where $\mathcal{A}_\mathcal{P}$ are set of propbank predicates and $\mathcal{A}_\mathcal{C}$ are rest of the nodes. [1]. Propbank predicates are *n-ary* with multiple edges based on their definitions. *amr-unknown* is a special concept node in the AMR graph indicating *wh-questions*.

Further, we enrich the AMR Graph $\mathcal{G}$ with explicit links to entities in the KG. For example, the question in Figure 1 contains two entities *Spain* and *Benicio Del Toro* that need to be identified and linked to DBpedia entries, *dbr:Spain* and *dbr:Benicio_del_toro*. Linking these entities is absolutely necessary for any KBQA system (Zou et al., 2014; Vakulenko et al., 2019). To do so, we trained a BERT-based neural mention detection model and used BLINK (Devlin et al., 2018) for disambiguation. The entities are linked to AMR nodes based on the AMR node-text alignment information. The linking is a bijective mapping from $\mathcal{V}_e \rightarrow E$ where $\mathcal{V}_e$ is the set of AMR entity nodes, and E is the set of entities in the underlying KG.

Query graph $\mathcal{Q}$ is a directed edge-labeled graph $\langle \mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q} \rangle$, which has a similar structure to the underlying KG. $\mathcal{V}_\mathcal{Q} \in \mathcal{V}_\mathcal{E} \cup \mathcal{V}$ where $\mathcal{V}_\mathcal{E}$ is a set of entities in the KG and $(\mathcal{V})$ is a set of unbound variables. $\mathcal{E}_\mathcal{Q}$ is a set of binary relations among $\mathcal{V}_\mathcal{Q}$ from the KG. The Query Graph $\mathcal{Q}$ is essentially the WHERE clause[2] in the SPARQL query.

Our goal is to transform the AMR graph $\mathcal{G}$ into its corresponding query graph $\mathcal{Q}$. However such transformation faces the following challenges:

N-ary argument mismatch: Query graph $\mathcal{Q}$ represents information using binary relations, whereas AMR graph contain Propbank framesets that are n-ary. For example, the node *produce-01*[3] from $\mathcal{A}_\mathcal{P}$ in $\mathcal{G}$ has four possible arguments, whereas its corresponding KG relation in $\mathcal{Q}$ (*dbo:producer*) is a binary relation.

Structural and Granular mismatch: The vertex set of the query graph $\mathcal{Q}$ represent entities (or unbound variables). On the other hand, AMR Graph $\mathcal{G}$ contains nodes that are concepts or PropBank predicates which can correspond to both entities and relationships. For example in Figure 1, *produce-01, star-01,* and *Spain* are nodes in the AMR graph. So the AMR graph $\mathcal{G}$ has to be transformed such that nodes primarily correspond to entities and edges (edge labels) correspond to relationships. Furthermore, it is possible for multiple predicates and concepts from $\mathcal{G}$ to jointly represent a single binary relation in $\mathcal{Q}$ because the underlying KG uses a completely different vocabulary. An example of such granular mismatch is shown in Figure 2.

2.2.1 Path-based Graph Transformation

We address the challenges mentioned above by using a path-based approach for the construction of Query Graphs. In KBQA, query graphs (i.e. SPARQL queries) constrain the unknown variable based on paths to the grounded entities. In Figure 1, the constraints in the SPARQL query are based on paths from *?actor* to *dbr:Benicio_del_toro* and *dbr:Spain* as shown below.

- `?actor → dbo:starring → ?movie → dbo:country → dbr:Spain`
- `?actor → dbo:starring → ?movie → dbo:producer → dbr:Benicio del Toro`

Based on this intuition of finding paths from the unknown variable to the grounded entities, we have developed a path-based approach depicted in Algorithm 1 that shows the steps for transforming the AMR Graph $\mathcal{G}$ into Query Graph $\mathcal{Q}$. As *amr-unknown* is the unknown variable in the AMR Graph, we retrieve all shortest paths (line 11 in Algorithm 1) between the *amr-unknown* node and the nodes $\mathcal{V}_E$ of the AMR Graph $\mathcal{G}$ that have mappings

---

[1] https://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf

[2] The Query Graph does not include the type constraints in the SPARQL WHERE Clause.

[3] http://verbs.colorado.edu/propbank/framesets-english-aliases/produce.html

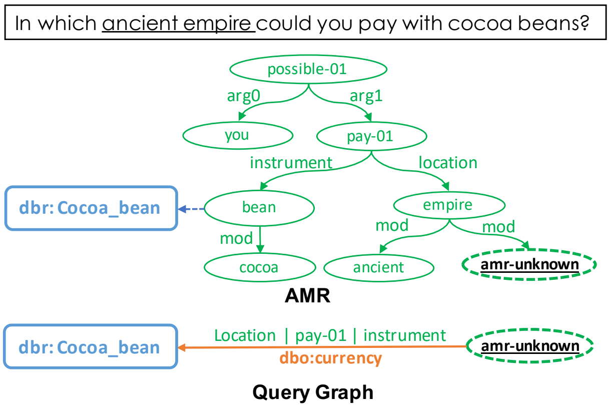

Figure 2: An example of granularity and structural mismatch between AMR and query graphs of the question *In which ancient empire could you pay with cocoa beans?*. The binary relation *dbo:currency* corresponds to the combination of two edges (*location*, *instrument*) and one node (*pay-01*) in the AMR graph.

to the KG entity set. Figure 1 shows an example of both the AMR and query graph for the question "Which actors starred in Spanish movies produced by Benicio del Toro?" Selecting the shortest paths reduces the n-ary predicates of AMR graph to only the relevant binary edges. For instance, the edge *(act-01, arg0, person)* in the AMR graph in Figure 1 will be ignored because it is not in the path between *amr-unknown* and any of the entities *dbr:Spain* and *dbr:Benicio_del_Toro*.

Structural and granularity mismatch between the AMR and query graph occurs when multiple nodes and edges in the AMR graph represent a single relationship in the query graph. This is shown in Figure 2. The path consists of one AMR node and 2 edges between *amr-unknown* and *cocoa bean*: (*amr-unknown*, location, *pay-01*, instrument, *cocoa-bean*)[4]. In such cases, we collapse all nodes that represent predicates (like *pay-01*, *star-01*, etc.) into an edge, and combine it with surrounding edge labels, giving (*location* | *pay-01* | *instrument*). This is done by line 18 of Algorithm 1 where the eventual query graph $\mathcal{Q}$ will have one edge with merged predicated from AMR graph $\mathcal{G}$ between the non-predicates ($\mathcal{A}_\mathcal{C}$).

Returning to the example in Figure 1, Algorithm 1 (line 25) outputs the query graph $\mathcal{Q}$ with the following two paths, which bear structural similarity to the knowledge graph:

Algorithm 1: AMR to triples

```
1  Input : Question text $t$, AMR graph $\mathcal{G} : \langle \mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G} \rangle$
     having a set of nodes $\mathcal{V}_e \in \mathcal{V}_\mathcal{G}$, each linked to a
     named entity in the KG
2  Returns : Query graph $\mathcal{Q} : \langle \mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q} \rangle$
   Initialize $\mathcal{Q} : \langle \mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q} \rangle, \mathcal{V}_\mathcal{Q} := \{\}, \mathcal{E}_\mathcal{Q} := \{\}$
3  if $t$ is imperative then
4      let source node (imperative predicate) be $r$
5      set $q$ as amr-unknown where
         $edge(r, q, \text{'arg1'}) \in \mathcal{G}_\mathcal{E}$
6      delete $r$ and its edges from $\mathcal{G}$
7  $a :=$ amr-unknown node
8  if $\exists\, b : edge(a, b, \text{'mod'}) \in \mathcal{E}_\mathcal{G}$ then
9      $a := b$
10 for $e$ in $\mathcal{V}_e$ do
11     $amrPath :=$ getShortestPath$(\mathcal{G}, a, e)$
12     let $amrPath$ be $[a, n_1, n_2, ..., n_k]$, where
         $n_k = e$
13     $collapsedPath := [a]$
14     $n' := a$
15     $relBuilder :=$ ‘ ’
16     for $i : 1 \rightarrow k$ do
17         if $n_i \in \mathcal{A}_\mathcal{P}$ then
18             $relBuilder :=$
               $relBuilder +$ getRel$_\mathcal{G}(n', n_i)$
19         else if $n_i \in \mathcal{A}_\mathcal{C}$ then
20             append $n_i$ to $collapsedPath$
21             add node $n'$ to $\mathcal{V}_\mathcal{Q}$
22             add $edge(n', n_i, relBuilder)$ to $\mathcal{E}_\mathcal{Q}$
23             $n' := n_i$
24             $relBuilder :=$ ‘ ’
       end
   end
25 $\mathcal{Q} := doRelLinking(\mathcal{Q})$
26 return $\mathcal{Q}$
```

- `amr-unknown → star-01 → movie → country → Spain`
- `amr-unknown → star-01 → movie → produce-01 → Benicio del Toro`

Note that in the above paths, edge labels reflect the predicates from the AMR graph (*star-01*, *produce-01*, and *mod*). Our next step is to resolve these edge labels to its corresponding relationships from the underlying KG. To do so, we perform relation linking as described below.

Relationship Linking. NSQA uses SemREL (Naseem et al., 2021), a state-of-the-art relation linking system that takes in the question text and AMR predicate as input and returns a ranked list of KG relationships for each triple. The cartesian product of this represents a ranked list of candidate query graphs, and we choose the highest-ranked *valid* query graph (a KG subgraph with unbound variables). As shown in Figure 1, the output of this module produces query graph $\mathcal{Q}$ with *star-01* and *produce-01* mapped to DBpedia rela-

---

[4] Nodes are indicated by boldface, and the rest are edges. For the purposes of path generation, the nodes *amr-unknown* and *empire* are considered equivalent because the *mod* edge is a descriptor in AMR (line 8 in Algorithm 1)

tions *dbo:starring* and *dbo:producer*. This will be the WHERE clause of the final SPARQL query.

2.2.2 Logic Generation

Our query graph can be directly translated to the WHERE clause of the SPARQL. We use existential first order logic (FOL) as an intermediate representation, where the non-logical symbols consist of the binary relations and entities in the KB as well as some additional functions to represent SPARQL query constructs (e.g. COUNT). We use existential FOL instead of directly translating to SPARQL because: (a) it enables the use of any FOL reasoner which we demonstrate in our next Section 2.3; (b) it is compatible with reasoning techniques beyond the scope of typical KBQA, such as temporal and spatial reasoning; (c) it can also be used as a step towards query embedding approaches that can handle incompleteness of knowledge graphs (Ren and Leskovec, 2020; Cohen et al., 2020; Sun et al., 2020). The Query Graph from Section 2 can be written as a conjunction in existential first order logic as shown in Figure 1.

The current logic form supports SPARQL constructs such as SELECT, COUNT, ASK, and SORT which are reflected in the types of questions that our system is able to answer in Table 4. The heuristics to determine these constructs from AMR are as follows:

Query Type: This rule determines if the query will use the ASK or SELECT construct. Boolean questions will have AMR parses that either have no *amr-unknown* variable or have an *amr-unknown* variable connected to a *:polarity* edge (indicating a true/false question). In such cases, the rule returns ASK, otherwise it returns SPARQL.

Target Variable: This rule determines what unbound variable follows a SPARQL statement. As mentioned in Section 2, the *amr-unknown* node represents the missing concept in a question, so it is used as the target variable for the query. The one exception is for questions that have an AMR predicate that is marked as imperative, e.g. in Figure 3 (middle) a question beginning with "Count the awards ..." will have *count-01* marked as imperative. In these cases, the algorithm uses the *arg1* of the imperative predicate as the target variable (see Algorithm 1, line 3).

Sorting: This rule detects the need for sorting by the presence of superlatives and quantities in the query graph prior to relation linking. Superlatives are parsed into AMR with *most* and *least* nodes and quantities are indicated by the PropBank frame *have-degree-91*, whose arguments determine: (1) which variable in $\mathcal{V}$ represents the quantity of interest, and (2) the direction of the sort (ascending or descending).

Counting: This rule determines if the COUNT aggregation function is needed by looking for PropBank frame *count-01* or AMR edge *:quant* connected to *amr-unknown*, indicating that the question seeks a numeric answer. However, questions such as "How many people live in London?" can have *:quant* associated to *amr-unknown* even though the correct query will use *dbo:population* to directly retrieve the numeric answer without the COUNT aggregation function. We therefore exclude the COUNT aggregation function if the KB relation corresponding to *:quant* or *count-01* has a numeric type as its range.

2.3 Reasoner

With the motivation of utilizing modular, generic systems, NSQA uses a First Order Logic, neuro-symbolic reasoner called Logical Neural Networks (LNN) (Riegel et al., 2020). This module currently supports two types of reasoning: type-based, and geographic. Type-based reasoning is used to eliminate queries based on inconsistencies with the type hierarchy in the KB. On the other hand, a question like "Was Natalie Portman born in United States?" requires geographic reasoning because the entities related to *dbo:birthPlace* are generally cities, but the question requires a comparison of countries. This is addressed by manually adding logical axioms to perform the required transitive reasoning for property *dbo:birthPlace*. We wish to emphasize that the intermediate logic and reasoning module allow for NSQA to be extended for such complex reasoning in future work.

3 Experimental Evaluation

The goal of the work is to show the value of AMR as a generic semantic parser on a modular KBQA system. In order to evaluate this, we first perform an end-to-end evaluation of NSQA (Section 3.2). Next, we discuss some qualitative and quantitative results on the value of AMR for different aspects of our KBQA system (Section 3.3). Finally, in support of our modular architecture, we evaluate the individual modules that are used in comparison to other state of the art approaches (Section 3.4).

	Dataset	P	R	F
WDAqua	QALD-9	26.09	26.7	24.99
gAnswer	QALD-9	29.34	**32.68**	29.81
NSQA	QALD-9	**31.89**	32.05	**31.26**
WDAqua	LC-QuAD 1.0	22.00	38.00	28.00
QAMP	LC-QuAD 1.0	25.00	**50.00**	33.00
NSQA	LC-QuAD 1.0	**44.76**	45.82	**44.45**

Table 1: NSQA performance on QALD-9 and LC-QuAD 1.0

3.1 Datasets and Metrics

To evaluate NSQA, we used two standard KBQA datasets on DBpedia.

QALD - 9 (Usbeck et al., 2017) dataset has 408 training and 150 test questions in natural language, from DBpedia version 2016-10. Each question has an associated SPARQL query and gold answer set. Table 4 shows examples of all the question types in the QALD dev set.

LC-QuAD 1.0 (Trivedi et al., 2017) is a dataset with 5,000 questions based on templates and more than 80% of its questions contains two or more relations. Our modules are evaluated against a random sample of 200 questions from the training set. LC-QuAD 1.0 predominantly focuses on the multi-relational questions, aggregation (e.g. COUNT) and simple questions from Table 4.

Dev Set. We also created a randomly chosen development set of 98 QALD-9 and 200 LC-QuAD 1.0 questions for evaluating individual modules.

Metrics. We report performance based on standard precision, recall and F-score metrics for the KBQA system and other modules. For the AMR parser we use the standard Smatch metric (Cai and Knight, 2013).

3.2 End-to-end Evaluation

Baselines: We evaluate NSQA against four systems: GAnswer (Zou et al., 2014), QAmp (Vakulenko et al., 2019), WDAqua-core1 (Diefenbach et al., 2020), and a recent approach by (Liang et al., 2021). GAnswer is a graph data-driven approach and is the state-of-the-art on the QALD dataset. QAmp is another graph-driven approach based on message passing and is the state-of-the-art on LC-QuAD 1.0 dataset. WDAqua-core1 is knowledge base agnostic approach that, to the best of our knowledge, is the only technique that has been evaluated on both QALD-9 and LC-QuAD 1.0 on different versions of DBpedia. Lastly, Liang et al. (Liang et al., 2021) is a recent approach that uses an ensemble of entity and relation linking modules and train a Tree-LSTM model for query ranking.

	AMR3.0	QALD-9	LC-QuAD 1.0
stack-Transformer	80.00	87.91	84.03

Table 2: AMR parsing performance (Smatch) on the AMR3.0 test and QALD-9, LC-QuAD 1.0 dev sets.

Results: Table 1 shows the performance of NSQA compared to state-of-the-art approaches on QALD and LC-QuAD 1.0 datasets. On QALD-9 and LC-QuAD 1.0, NSQA achieves state-of-the-art performance. It outperforms WDAqua and gAnswer on QALD-9. Furthermore, NSQA's performance on LC-QuAD 1.0 significantly outperforms QAmp by 11.45 percentage points on F1.

Due to difference in evaluation setup in Liang et al. (2021), we reevaluated their system on the same setup and metrics as the above systems. Given the test set and the evaluation, (Liang et al., 2021)'s F1 score reduces to 29.2%[5]. We exclude this work from our comparison due to lack of standard evaluation.

3.3 Performance Analysis of AMR

AMR Parsing. We manually created AMRs for the train and dev sets of QALD and LC-QuAD 1.0 questions. The performance of our stack-transformer parser on both of these datasets is shown in Table 2. The parser is trained on the combination of human annotated treebanks and a synthetic AMR corpus. Human annotated treebanks include AMR3.0 and 877 questions sentences (250 QALD train + 627 LC-QuAD 1.0 train sentences) annotated in-house. The synthetic AMR corpus includes 27k sentences obtained by parsing LC-QuAD 1.0 and LC-QuAD 2.0 (Dubey et al., 2019) training sentences, along the lines of (Lee et al., 2020).

AMR-based Query Structure NSQA leverages many of the AMR features to decide on the correct query structure. As shown in Section 2.2.2, NSQA relies on the existence of certain PropBank predicates in the AMR parse such as *have-degree-91*, *count-01*, *amr-unknown* to decide on which SPARQL constructs to add. In addition, the AMR parse determines the structure of the WHERE clause. In Table 3, we show the accuracy of each one of

---

[5] Liang et al. (2021) report an F1 score of 68% on a different subset of LC-QuAD 1.0. They also consider only questions that returns an answer which is a different setup from the rest of the systems.

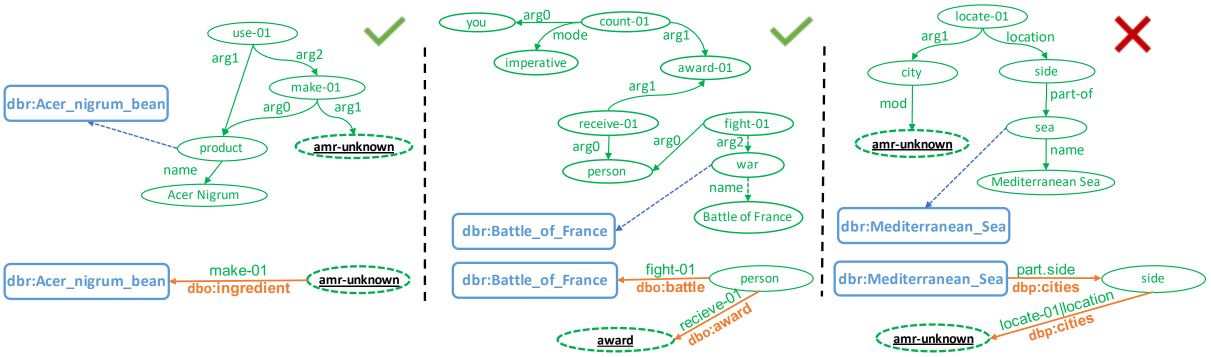

Figure 3: AMR and query graphs for the questions *"Acer nigrum is used in making what?"*, *"Count the awards received by the ones who fought the battle of france?"* and *"What cities are located on the sides of mediterranean sea?"* from LC-QuAD 1.0 dev set

Query Feature	Correct	Total	Correct (%)
SELECT	164	186	88.2
ASK	14	9	64.3
COUNT	25	31	80.6
1-Hop	50	63	79.4
2-Hop	96	137	70.1

Table 3: Query constructs prediction (LC-QuAD 1.0 dev)

these rules on LC-QuAD 1.0 dev dataset. Overall, NSQA identified 64% of ASK (boolean) questions correctly and achieved more than 80% accuracy for COUNT and SELECT questions. Using AMR and the path-based approach, NSQA was able to correctly predict the total number of constraints with comparable accuracies of 79% and 70% for single and two-hops, respectively. NSQA finds the correct query structure for complex questions almost as often as for simple questions, completely independent of the KG.

Figure 3 shows two examples illustrating how AMR lends itself to an intuitive transformation to the correct query graph, as well as a third example where we fail. Here the AMR semantic parse can not be matched to the underlying KG, since 'side' is an extra intermediate variable that leads to an additional constraint in the query graph.

Supported Question Types. Table 4 shows the reasoning and question types supported by NSQA . Our transformation algorithm applied to AMR parses supports simple, multi-relational, count-based, and superlative question types. LNN performs geographic reasoning as well as type-based reasoning to rank candidate logic forms. Addressing comparative and temporal reasoning is a part of our future work.

3.4 Individual Module Evaluation

Entity and Relation Linking. NSQA's EL module (NMD+BLINK) consists of a BERT-based neural mention detection (NMD) network, trained on LC-QuAD 1.0 training dataset comprising of 3,651 questions with manually annotated mentions, paired with an off-the-shelf entity disambiguation model – BLINK (Wu et al., 2019b). We compare the performance of NMD+BLINK approach with Falcon (Sakor et al., 2019) in Table 5. NMD+BLINK performs 24% better on F1 than Falcon (state-of-the-art) on LC-QuAD 1.0 dev set and 3% better on QALD-9 dev set. Similarly, we evaluate Relation Linking on both QALD and LC-QuAD 1.0 dev sets. In particular, we used SemREL (Naseem et al., 2021); state-of-the-art relation linking approach which performs significantly better compared to both Falcon (Sakor et al., 2019) and SLING (Mihindukulasooriya et al., 2020) on various datasets. On LC-QuAD 1.0 dev, SemREL acheives F1 = 0.55 compared to 0.43 by SLING and 0.42 by Falcon. On QALD-9, SemREL achieves 0.54 compared to 0.64 and 0.46 F1 for SLING and Falcon, respectively.

Reasoner. We investigate the effect of using LNN as a reasoner equipped with axioms for type-based and geographic reasoning. We evaluated NSQA's performance under two conditions: (a) with an LNN reasoner with intermediate logic form and (b) with a deterministic translation of query graphs to SPARQL. On LC-QuAD 1.0 dev set, NSQA achieves an F1 score of 40.5 using LNN compared to 37.6 with the deterministic translation to SPARQL. Based on these initial promising results,

Question Type/Reasoning	Example	Supported
Simple	Who is the mayor of Paris	✓
Multi-relational	Give me all actors starring in movies directed by William Shatner.	✓
Count-based	How many theories did Albert Einstein come up with?	✓
Superlative	What is the highest mountain in Italy?	✓
Comparative	Does Breaking Bad have more episodes than Game of Thrones?	
Geographic	Was Natalie Portman born in the United States?	✓
Temporal	When will start [*sic*] the final match of the football world cup 2018?	

Table 4: Question types supported by NSQA , with examples from QALD

	Dataset	P	R	F1
Falcon	QALD-9	0.81	0.83	0.82
NMD+BLINK	QALD-9	0.82	0.90	**0.85**
Falcon	LC-QuAD 1.0	0.56	0.69	0.62
NMD+BLINK	LC-QuAD 1.0	0.87	0.86	**0.86**

Table 5: Performance of Entity Linking modules compared to SOTA Falcon on our dev sets

we intend to explore more uses of such reasoners for KBQA in the future.

4 Related Work

Early work in KBQA focused mainly on designing parsing algorithms and (synchronous) grammars to semantically parse input questions into KB queries (Zettlemoyer and Collins, 2007; Berant et al., 2013), with a few exceptions from the information extraction perspective that directly rely on relation detection (Yao and Van Durme, 2014; Bast and Haussmann, 2015). All the above approaches train statistical machine learning models based on human-crafted features and the performance is usually limited.

Deep Learning Models. The renaissance of neural models significantly improved the accuracy of KBQA systems (Yu et al., 2017; Wu et al., 2019a). Recently, the trend favors translating the question to its corresponding subgraph in the KG in an end-to-end learnable fashion, to reduce the human efforts and feature engineering. This includes two most commonly adopted directions: (1) embedding-based approaches to make the pipeline end-to-end differentiable (Bordes et al., 2015; Xu et al., 2019); (2) hard-decision approaches that generate a sequence of actions that forms the subgraph (Xu et al., 2018; Bhutani et al., 2019).

On domains with complex questions, like QALD and LC-QuAD, end-to-end approaches with hard-decisions have also been developed. Some have primarily focused on generating SPARQL sketches (Maheshwari et al., 2019; Chen et al., 2020) where they evaluate these sketches (2-hop) by providing gold entities and ignoring the evaluation of selecting target variables or other aggregation functions like sorting and counting. (Zheng and Zhang, 2019) generates the question subgraph via filling the entity and relationship slots of 12 predefined question template. Their performance on these datasets show significant improvement due to the availability of these manually created templates. Having the advantage of predefined templates does not qualify for a common ground to be compared to generic and non-template based approaches such as NSQA, WDAqua, and QAmp.

Graph Driven Approaches. Due to the lack of enough training data for KBQA, several systems adopt a training-free approach. WDAqua (Diefenbach et al., 2017) uses a pure rule-based method to convert a question to its SPARQL query. gAnswer (Zou et al., 2014) uses a graph matching algorithm based on the dependency parse of question and the knowledge graph. QAmp (Vakulenko et al., 2019) is a graph-driven approach that uses message passing over the KG subgraph containing all identified entities/relations where confidence scores get propagated to the nodes corresponding to the correct answers. Finally, (Mazzeo and Zaniolo, 2016) achieved superior performance on QALD-5/6 with a hand-crafted automaton based on human analysis of question templates. A common theme of these approaches, is that the process of learning the subgraph of the question is heavily KG specific, while our approach first delegates the question understanding to KG-independent AMR parsing.

Modular Approaches. Frankenstein (Singh et al., 2018) is a system that emphasize the aspect of reusuability where the system learns weights for each reusuable component conditioned on the questions. They neither focus on any KG-independent parsing (AMR) not their results are comparable to any state of the art approaches. (Liang et al., 2021) propose a modular approach for KBQA that uses an ensemble of phrase mapping techniques and a TreeLSTM-based model for ranking query candidates which requires task specific training data.

5 Discussion

The use of semantic parses such as AMR compared to syntactic dependency parses provides a number of advantages for KBQA systems. First, independent advances in AMR parsing that serve many other purposes can improve the overall performance of the system. For example, on LC-QUAD-1 dev set, a 1.4% performance improvement in AMR Smatch improved the overall system's performance by 1.2%. Recent work also introduces multilingual and domain-specific (biomedical) AMR parsers, which expands the possible domains of application for this work. Second, AMR provides a normalized form of input questions that makes NSQA resilient to subtle changes in input questions with the same meaning. Finally, AMR also transparently handles complex sentence structures such as multi-hop questions or imperative statements.

Nevertheless, the use of AMR semantic parses in NSQA comes with its own set of challenges: 1) Error propagation: Although AMR parsers are very performant (state-of-the-art model achieves an Smatch of over 84%), inter-annotator agreement is only 83% on newswire sentences, as noted in (Banarescu et al., 2013). Accordingly, AMR errors can propagate in NSQA's pipeline and cause errors in generating the correct answer, 2) Granularity mismatch: our proposed path-based AMR transformation is generic and not driven by any domain-specific motivation, but additional adjustments to the algorithm may be needed in new domains due to the different granularity between AMR and SPARQL 3) Optimization mismatch: Smatch, the optimization objective for AMR training, is sub-optimal for KBQA. NSQA requires a particular subset of paths to be correctly extracted, whereas the standard AMR metric Smatch focuses equally on all edge-node triples. We are therefore exploring alternative metrics and how to incorporate them into model training.

6 Conclusion and Future Work

To the best of our knowledge, NSQA is the first system that successfully harnesses a generic semantic parser, particularly AMR, for a KBQA task. Our path-based approach to map AMR to the underlying KG such as DBpedia is first of its kind with promising results in handling compositional queries. NSQA is a modular system where each modules are trained separately for its own task, hence not requiring end-to-end KBQA training. In future, we will explore the potential of the more expressive intermediate logic form with the neuro-symbolic reasoner for KBQA. Particularly, we intend to focus on extending NSQA for temporal reasoning and making it robust to handle incompleteness and inconsistencies in knowledge bases.

References

Ibrahim Abdelaziz, Srinivas Ravishankar, Pavan Kapanipathi, Salim Roukos, and Alexander Gray. 2021. A semantic parsing and reasoning-based approach to knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15985–15987.

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 1001–1007.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of AAAI*.

Nikita Bhutani, Xinyi Zheng, and H V Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 739–748, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2020. Formal query building with query structure prediction for complex question answering over knowledge base. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

William W Cohen, Haitian Sun, R Alex Hofer, and Matthew Siegler. 2020. Scalable neural methods for reasoning with a symbolic knowledge base. *arXiv preprint arXiv:2002.06115*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. 2020. Towards a question answering system over the semantic web. *Semantic Web*, (Preprint):1–19.

Dennis Diefenbach, Kamal Singh, and Pierre Maret. 2017. Wdaqua-core0: A question answering component for the research community. In *Semantic Web Evaluation Challenge*, pages 84–89. Springer.

Bonnie Dorr, Nizar Habash, and David Traum. 1998. A thematic hierarchy for efficient generation from lexical-conceptual structure. In *Conference of the Association for Machine Translation in the Americas*, pages 333–343. Springer.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web - ISWC 2019*.

Paul R Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Young-Suk Lee, Ramón Fernandez Astuillo, Tahira Naseem, Revanth Reddy, Radu Florian, and Salim Roukos. 2020. Pushing the limits of amr parsing with self-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 3208–3214.

Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova, and Manuel Gil. 2021. Querying knowledge graphs in natural language. *Journal of Big Data*, 8(1):1–23.

Gaurav Maheshwari, Priyansh Trivedi, Denis Lukovnikov, Nilesh Chakraborty, Asja Fischer, and Jens Lehmann. 2019. Learning to rank query graphs for complex question answering over knowledge graphs. In *International semantic web conference*, pages 487–504. Springer.

Giuseppe M Mazzeo and Carlo Zaniolo. 2016. Answering controlled natural language questions on rdf knowledge bases. In *EDBT*, pages 608–611.

Nandana Mihindukulasooriya, Gaetano Rossiello, Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Mo Yu, Alfio Gliozzo, Salim Roukos, and Alexander Gray. 2020. Leveraging Semantic Parsing for Relation Linking over Knowledge Bases. In *Proceedings of the 19th International Semantic Web Conference (ISWC2020)*.

Tahira Naseem, Srinivas Ravishankar, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Young-Suk Lee, Pavan Kapanipathi, Salim Roukos, Alfio Gliozzo, and Alexander Gray. 2021. A semantics-aware transformer model of relation linking for knowledge base question answering. In *ACL 2021*.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding smatch: Transition-based amr parsing with reinforcement learning. *arXiv preprint arXiv:1905.13370*.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. 2020. Logical neural networks. *arXiv preprint arXiv:2006.13155*.

Ahmad Sakor, Isaiah Onando Mulang', Kuldeep Singh, Saeedeh Shekarpour, Maria Esther Vidal, Jens Lehmann, and Sören Auer. 2019. Old is gold: Linguistic driven approach for entity and relation linking of short text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2336–2346, Minneapolis, Minnesota. Association for Computational Linguistics.

Kuldeep Singh, Andreas Both, Arun Sethupat, and Saeedeh Shekarpour. 2018. Frankenstein: a platform enabling reuse of question answering components. In *European Semantic Web Conference*, pages 624–638. Springer.

Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. 2020. Faithful embeddings for knowledge base queries.

Advances in Neural Information Processing Systems, 33.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *ISWC 2017*, pages 210–218.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (qald-7). In *Semantic Web Evaluation Challenge*, pages 59–69. Springer.

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. 2019. Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1431–1440.

Dekun Wu, Nana Nosirova, Hui Jiang, and Mingbin Xu. 2019a. A general fofe-net framework for simple and effective question answering over knowledge bases. *arXiv preprint arXiv:1903.12356*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2019b. Zero-shot entity linking with dense entity retrieval. *ArXiv*, abs/1911.03814.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2937–2947.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 918–924.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 571–581.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.

Weiguo Zheng and Mei Zhang. 2019. Question answering over knowledge graphs via structural query patterns. *arXiv preprint arXiv:1910.09760*.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324.

On the Gap between Adoption and Understanding in NLP

Federico Bianchi
Bocconi University
Milano, Italy
`f.bianchi@unibocconi.it`

Dirk Hovy
Bocconi University
Milano, Italy
`dirk.hovy@unibocconi.it`

Abstract

There are some issues with current research trends in NLP that can hamper the free development of scientific research. We identify five of particular concern: 1) the early adoption of methods without sufficient understanding or analysis; 2) the preference for computational methods regardless of risks associated with their limitations; 3) the resulting bias in the papers we publish; 4) the impossibility of re-running some experiments due to their cost; 5) the dangers of unexplainable methods. If these issues are not addressed, we risk a loss of reproducibility, reputability, and subsequently public trust in our field. In this position paper, we outline each of these points and suggest ways forward.

1 Early Adoption

When BERT (Devlin et al., 2019) was introduced in 2019, it revolutionized Natural Language Processing (NLP), showing impressive capabilities on many tasks in various languages (Nozza et al., 2020). However, papers soon highlighted its limits (Ettinger, 2020; Bender and Koller, 2020) and identified issues with bias (Nozza et al., 2021), i.e., that contextual models show different performances for different genders and ethnic groups (Zhang et al., 2020). While Rogers et al. (2020a)'s "BERTology" paper outlines what we know about BERT, much remains unexplored. We do not really know what BERT "understands", but, as Bender and Koller (2020) pointed out, we frequently overestimate its understanding capabilities. E.g., Ettinger (2020) shows that BERT has very low sensitivity to the concept of negation shows. This is not to say that we should stop using BERT; indeed, it is hard to imagine the field today without it. But it does illustrate the *gap between adoption and understanding* (GAU) of a new technology. As the field grows, this situation is likely to play out more frequently, as it is aided by various circumstances.

Early adoption of novel methods and their rigorous testing by the field are a strength of NLP. This approach has propelled insights through cascading waves of novel methods. However, the adoption of new technologies without full awareness of their potential and side effects is a risky proposition. In the 1950s and 1960s, a German pharmaceuticals company aggressively marketed a new drug called "Contergan" to treat sleeping problems as well as morning sickness during pregnancy. The drug had been classified as safe after extensive trials, and was widely prescribed. However, the trials had actually excluded pregnant women, so it was only after its approval that the effects of the drug's main component, thalidomide, became clear: thousands suffered severe birth defects and miscarriages (Criado Perez, 2019).

NLP is not a chemical product, and its effects are not as physically harmful. Using a model that later turns out to be overfitting is not in the same category as failing to protect people from bodily harm. However, it has other consequences. Say researcher A publishes a new result with method X, which becomes the number to beat. Many try and fail, not publishing their results or being ignored. Later, method X is found to be wrong. But the insights that were shelved or ignored in between are lost, unable to unseat method X's false supremacy. Maybe A retracts their initial result, but the damage is done. If others have built on X in the meantime, they will also be affected by its collapse. Imagine for a moment the ripple effects if a central paper like BERT turned out to be wrong. More likely, though, is that A, satisfied with the results of X, simply moves on. The barrier to new, better work remains in place, the faulty method is not identified as such, and instead a method heralded as revolutionary causes stagnation in the field.

Social psychology has struggled with this "win-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3895–3901
August 1–6, 2021.

ner's curse" (Ioannidis, 2008) of unreplicable results. The cause there was the prevalence of false positives that appeared significant, but were just lucky flukes (Ioannidis, 2005). Simmons et al. (2011) found that in many cases, a researcher's decision (conscious or subconscious) had influenced the results. While the causes are different, the resulting issue is the same in NLP.

We will see in the subsequent sections how, counter-intuitively, the GAU creates incentives to publish faster, focus on methods, and drop experiments that do not lead to state-of-the-art results.

A way forward We need to create an environment that makes negative findings and exploration of shortcomings possible, and preferably not just as afterthoughts. Workshops on negative results (Rogers et al., 2020b) and the stress-testing idea of build-it, break-it (Ettinger et al., 2017) are steps in the right direction.

2 Computational Papers

The method-driven nature of NLP makes it necessary to constantly explore new techniques, and the upsides are readily apparent. However, it has also tipped the balance of papers away from introspection and linguistically motivated studies. This development seems rooted in the statistical revolution that began in the 1990s, when method-driven papers outperformed theory-motivated ones. By now, it seems this attitude is very much ingrained in our community, where new models are appreciated more than purely linguistic results. To be fair, purely methodological papers are easier to evaluate objectively. This situation invites two questions, though: (i) Are modeling results more important than linguistics insights? And (ii) should computational papers be evaluated differently?

Norvig (2017) has remarked on the tension between rationalism, which wants models that can be understood, and empiricism, which wants models that work.[1] There is likely always going to be a pendulum swing between these extremes. But to make true progress, we do need both approaches.

While to date no survey has quantified the most popular methods in NLP research, it seems anecdotally that many of the accepted papers at top conferences in the field introduce novel models. Assuming that modeling results are important in our field, we need to understand if we are evaluating these papers in the correct way. This brings up the question of replicability: there has recently been a push to find ways to require authors to share their code and parameters, but this is not enough. Papers often fail to include the complete setup on which they base their models. Even the slightest difference in the setting can bring huge differences in the results: changing between CUDA versions can affect the results on GPU. Moreover, papers' repositories are often incomplete, failing to include the dependencies that would allow others to easy replicate experiments or, worse, containing only poorly documented Jupyter notebooks. Code is a fundamental component of science and should be regarded as such. Code should not just enable researchers to re-run a given experiment, but to apply the same method to other data sets. Bad code is not useful, and the cost of re-implementation, combined with the risk of not being able to get the original results, is often too high to justify the investment of researchers' time. Writing bad code should be akin to writing a bad paper; it does not necessarily make the research wrong, but it makes it less reproducible.

This situation brings up another question: Should authors be responsible for actively maintaining code? Once a new paper is published and the code released, reviewers might ask for comparisons. However, it is common to see repositories on GitHub that have many unanswered questions.

Methodological errors can slow research; as shown by Musgrave et al. (2020), due to methodological flaws in the experiments, the increase of performance in metric learning was wrongly reported in several papers. Moreover, a few hyperparameters can make a huge difference in the results of a given experiment (Ruffinelli et al., 2020). Well-documented methodological design and code make it easier to find bugs and experimental problems. Often systematic evaluations are run to compare different state-of-the-art methods and show that the results are sometimes on par with baselines (Dacrema et al., 2019; Errica et al., 2020).

A way forward Following Bender and Friedman (2018) on providing a data statement, we believe that a code statement similar to that offered by Mitchell et al. (2019) should be provided along with a paper. Code is part of the scientific contribution similar to the experiments we run to prove a hypothesis. While there can be different opinions

---

[1] The empiricist preference can be summed up with statistician George Box's aphorism, "All models are wrong, but some are useful."

on how to write good or bad code, writing documented code that is at least easy to use (with the use of high-level interfaces and convenient wrappers) can be required practice. Moreover, this can be of help in the context of systematic evaluation initiatives like code base that integrate multiple algorithms (Terragni et al., 2021): this process can help in reducing methodological errors in the evaluation. We can cite HuggingFace[2] as a notable example of a repository with good systematicity that has allowed many researchers to replicate most of the recent and popular research in the NLP field. Similarly, Sentence BERT (Reimers and Gurevych, 2019) - a method for sentence embeddings based on BERT - was released with a well organized and easy-to-use source code[3] that made it possible to use the models as a baseline or as the foundation of many other research works (Li et al., 2020; Ke et al., 2020; Zemlyanskiy et al., 2021; Bianchi et al., 2021c,b, inter alia), suggesting that this kind of release can be of great benefit to the community.

3 Publication Bias

Most researchers want to publish in A+ venues and Q1 journals as academic positions use publication output as a criterion for promotions. While quality *should* play into this assessment, it would be wrong to assume university committees are familiar enough with all venues and subfields in a discipline to make an accurate assessment of the relative value of each contribution. So the number of papers and citation count often trump other considerations, especially when publications give universities PR.

Proponents of "slow science" have argued for a shift away from this emphasis on quantity. But while laudable in theory, committing to slowness in practice does not align with the needs of especially junior researchers. They might well prefer less publication pressure, but they can likely not afford the trade-off between a theoretically desirable publication model and losing their job. Given all that, it would be unfair to put the responsibility for addressing the publication bias feeding the GAU on junior scholars. Furthermore, it is not clear what makes a paper worthy of an A+ venue. For NLP conferences, that decision is left to three reviewers and one area chair. Faced with an increasing number of papers to review, reviewers have found themselves with less and less time to make a well-rounded decision on each one. As a consequence, many good papers do not get published.

To keep up with publication demands, authors can 1) make their papers easier to judge based on a quick read or 2) find alternative venues. The former means focusing on a single, easily recognizable contribution—much easier for method papers (see also the previous section). The latter means many authors are now deciding to build the foundation of their publication record by publishing on ArXiv. There is nothing wrong with ArXiv itself. Thanks to it, researchers can make and share valuable contributions with other researchers online. However, ArXiv is a new, and changeable, venue. Keeping the current pace of NLP research, it is bound to also publish models that are biased. So publication records built on ArXiv are set on quicksand and might fall over: the GAU we described previously. While it is encouraging to see that we as a field subsequently work on reducing the bias of those models, we are still allowing anyone to use and deploy those biased models in the meanwhile, contributing to the GAU.

A way forward Short of changing the incentive structure, we can do more to strengthen the review process. Tutorials on reviewing (Cohen et al., 2020) and the implementation of their recommendations would go a long way toward ensuring that we maintain a high standard at a high volume.

4 Computationally Unobtainable

The previous sections have argued that the GAU is a consequence of a strong preference within the field for computational methods, amplified by existing publication bias. But even if all of the solutions we suggested were adopted, there would still be issues that affect reproducibility.

In a panel discussion at EurNLP 2019, Phil Blunsom correctly remarked that "[t]he future of NLP is not about bigger models, it's about bigger ideas."[4] Indeed, there are many arguments against simply making models bigger. However, there is a difference between having the possibility of running a bigger model and not needing it, and needing a bigger model but not having it.

Popular methods like BERT or GPT-3 are now impossible to develop without huge amounts of funding. Developing a new algorithm means run-

---

[2] `huggingface.co`

[3] `https://github.com/UKPLab/sentence-transformers`

[4] `https://twitter.com/glorisonne/status/1182693114672271360`

ning multiple experiments, multiple times, adjusting parameters and configurations. So there is not just the already prohibitive cost of pre-training one BERT model, but the cost of pre-training BERT dozens of times.

Asking authors to provide results by re-training models is too high a cost for many academic researchers. True, other fields have to invest much more money to run experiments. E.g., neurosciences require universities to buy ECG or MEG devices that can cost up to 2 million dollars. On the other hand, those devices are much more consistently reused than a single pre-trained model, so costs are much more distributed.

It is also unclear what constitutes a "bigger" model. Parameter sizes have grown exponentially for the last few years (Bender et al., 2021), and what was considered preposterously large five years ago is now pretty standard. But even old standards are becoming hard to match for universities. This situation creates an unbridgeable gap between industry and academia. Even within those two groups, there are rapidly emerging differences between the players. Only rich universities can afford to re-pre-train models from scratch. Demanding that all actors have access to the same resources would be unreasonable (however desirable). Industry players need to maintain a competitive edge, and we can hardly hope to address the inequality between national academic systems and their funding. But this reality creates a situation where reproducibility becomes impossible. If team A presents results from a model that only team A can train, then the rest of us need to take those results at face value. Whether we believe them or not becomes irrelevant. In addition to this problem, consider the environmental concerns generated by the training of large models (Strubell et al., 2019), and bigger does not necessarily equate better for reproducibility.

Lack of reproducibility, though, is a danger to scientific reliability. The fallout from the reproducibility crisis in social psychology (see Section 4) has tainted the reputation of the entire field. But it has also led to remarkable innovations. Studies now need to be pre-registered to prevent "fishing expeditions" to find the most "interesting" result. International research teams have organized trials to replicate famous experiments—often disproving canonical theories, but also re-opening avenues of research that had been wrongly foreclosed (Collaboration, 2015). And while NLP researchers generally favor reproducibility (Mieskes et al., 2019), we are not yet doing it. Fokkens et al. (2013) already identified five parameters for better reproducibility. A study by Belz et al. (2021), though, found that only 15.61% of 506 papers they checked were independently reproducible. This is in contrast to the high level of data shared in NLP (Mieskes, 2017), which should aid reproduction.

A way forward We need to consider the value as well as the cost of computation in terms of resources, people, and money. If it becomes impossible to replicate some experiments because of those factors, we need to foster computationally-affordable solutions that can be tested by everyone.

5 Unexplainable Methods

A final important issue is the low explainability of our models. This argument recalls the rationalism vs. empiricism debate we mentioned earlier, and is valid for most deep learning models. However, it has recently become more prevalent due to the low effort needed to set up and run these models.

GPT-3 (Brown et al., 2020) made it to the public quickly, most notably as "author" of an auto-generated Guardian article (GPT-3, 2020), which also generated a lot of opinions. The final article was the heavily-edited output of several GPT-3 runs to get a coherent version. So while still not an autonomous contribution, the potential impact that this technology is clear: It has allowed people to create eye-catching, interesting applications that capture the public's imagination. Those creations go beyond natural language. For example, GPT-3 can output HTML and CSS just from an input description of the desired outcome (Szőgyényi, 2020).

So while sensational results are picked up by the media and easily make their way to the general public (GPT-3, 2020), more nuanced comments and limitations of those results tend to be confined to in-domain newsfeeds (Marcus and Davis, 2020; Floridi and Chiriatti, 2020). Indeed, the public is left with the idea that these methods are either a panacea for all problems of the day, or the next step on the path to machine domination. But as scientists, we should realize that sensationalizing what we do comes with great responsibilities.

Easy availability of this technology can bring harm: it is not difficult to imagine the consequences of early access to this kind of technology, like biased auto-generated articles and fake news. However, it is difficult to predict which unwanted outcomes as of yet unknown models could generate. This is not a new argument: face recognition models have already created problems with racial bias, such as identifying images of Black people as "gorillas" (Zhang, 2015).

This point is even more poignant when we realize that different studies have pointed out that GPT-3 is not exhibiting what we as humans would call "intelligence" (Marcus and Davis, 2020; Floridi and Chiriatti, 2020) and have suggested that learning just from text and without including external experience might be a difficult task (Bisk et al., 2020; Bender and Koller, 2020; Bianchi et al., 2021a). Intelligent beings can explain their decisions and reason about their process. But GPT-3 and similar models make (unexplainable) decisions that *look* intelligent. This level of unexplainability hinders the future applications of this technology to areas that crucially depend on post-hoc explanations of the process. E.g., fields like medicine and law.

A way forward This problem does not have a solution – yet. We need to better engage with the media and public to make sure that what comes from our field is not only the great news of spectacular possibilities. It is difficult to also share the limitations: it might bore the public, detract from the undeniable successes, and it requires more effort to explain than simple, glowing success stories. But it is the only way to be sure that everyone has understood the full range of possible outcomes of unexplainable models.

6 Conclusion

We have argued that the current publication model of NLP fosters a gap between adoption and understanding of models, making it easier to meet publication demands with method papers, while shelving more unwieldy studies that include negative results and more epistemological approaches.

This issue is compounded by the rise of ever-larger models, which are unobtainable by all but a few researchers, and make it harder to explain how our methods work. As a result, reproducibility might suffer, and consequently endanger NLP's credibility and the public's trust in the field.

We do not make the argument that these are the only issues that our community has to take care of. For example, environmental sustainability (Strubell et al., 2019) of the models and the possible dual-use problem (Leins et al., 2020) are important topics that require a separate discussion.

Ethical Considerations

The main topic of this paper, reproducibility, is related to issues of ethics in NLP, with respect to fairness and accessibility. With this paper, we hope to contribute to that literature. Our paper does not contain new data sets or methods that pose a potential dual-use or bias problem.

References

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2021. A systematic review of reproducibility research in natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 381–393, Online. Association for Computational Linguistics.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big. *Proceedings of FAccT*.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Federico Bianchi, Ciro Greco, and Jacopo Tagliabue. 2021a. Language in a (search) box: Grounding language learning in real-world human-machine interaction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4409–4415, Online. Association for Computational Linguistics.

Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2021b. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, Online. Association for Computational Linguistics.

Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. 2021c. Cross-lingual contextualized topic models with zero-shot learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational*

Linguistics: Main Volume, pages 1676–1683, Online. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kevin Cohen, Karën Fort, Margot Mieskes, and Aurélie Névéol. 2020. Reviewing natural language processing research. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 16–18, Online. Association for Computational Linguistics.

Open Science Collaboration. 2015. Estimating the reproducibility of psychological science. *Science*, 349(6251).

Caroline Criado Perez. 2019. *Invisible women: Exposing data bias in a world designed for men*. Random House.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 101–109. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M. Bender. 2017. Towards linguistically generalizable NLP systems: A workshop and shared task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 1–10, Copenhagen, Denmark. Association for Computational Linguistics.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria. Association for Computational Linguistics.

GPT-3. 2020. A robot wrote this entire article. are you scared yet, human? — GPT-3. *The Guardian*.

John PA Ioannidis. 2005. Why most published research findings are false. *PLoS medicine*, 2(8):e124.

John PA Ioannidis. 2008. Why most discovered true associations are inflated. *Epidemiology*, pages 640–648.

Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. 2020. SentiLARE: Sentiment-aware language representation learning with linguistic knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6975–6988, Online. Association for Computational Linguistics.

Kobi Leins, Jey Han Lau, and Timothy Baldwin. 2020. Give me convenience and give her death: Who should decide what uses of NLP are appropriate, and on what basis? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2908–2913, Online. Association for Computational Linguistics.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60.

Gary Marcus and Ernest Davis. 2020. Gpt-3, bloviator: Openai's language generator has no idea what it's talking about. *Technology Review*.

Margot Mieskes. 2017. A quantitative study of data in the NLP community. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 23–29, Valencia, Spain. Association for Computational Linguistics.

Margot Mieskes, Karën Fort, Aurélie Névéol, Cyril Grouin, and Kevin Cohen. 2019. Community perspective on replicability in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 768–775, Varna, Bulgaria. INCOMA Ltd.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 220–229. ACM.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. 2020. A metric learning reality check. In *Computer Vision – ECCV 2020*, pages 681–699, Cham. Springer International Publishing.

Peter Norvig. 2017. On chomsky and the two cultures of statistical learning. In *Berechenbarkeit der Welt?*, pages 61–83. Springer.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [MASK]? making sense of language-specific BERT models. *arXiv preprint arXiv:2003.02912*.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2021. HONEST: Measuring hurtful sentence completion in language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2398–2406, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020a. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Anna Rogers, João Sedoc, and Anna Rumshisky, editors. 2020b. *Proceedings of the First Workshop on Insights from Negative Results in NLP*. Association for Computational Linguistics, Online.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Joseph P Simmons, Leif D Nelson, and Uri Simonsohn. 2011. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological science*, 22(11):1359–1366.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Zoltán Szőgyényi. 2020. We built an OpenAI powered Tailwind CSS code generator using GPT-3. *Themesberg Magazine*.

Silvia Terragni, Elisabetta Fersini, Bruno Giovanni Galuzzi, Pietro Tropeano, and Antonio Candelieri. 2021. OCTIS: Comparing and optimizing topic models is simple! In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 263–270. Association for Computational Linguistics.

Yury Zemlyanskiy, Sudeep Gandhe, Ruining He, Bhargav Kanagal, Anirudh Ravula, Juraj Gottweis, Fei Sha, and Ilya Eckstein. 2021. DOCENT: Learning self-supervised entity representations from large document collections. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2540–2549, Online. Association for Computational Linguistics.

Haoran Zhang, Amy X Lu, Mohamed Abdalla, Matthew McDermott, and Marzyeh Ghassemi. 2020. Hurtful words: quantifying biases in clinical contextual word embeddings. In *proceedings of the ACM Conference on Health, Inference, and Learning*, pages 110–120.

Maggie Zhang. 2015. Google photos tags two African-Americans as gorillas through facial recognition software. *Forbes Magazine*.

Learning Disentangled Latent Topics for Twitter Rumour Veracity Classification

[1]**John Dougrez-Lewis** [1,2,3]**Maria Liakata** [2,3]**Elena Kochkina** [1]**Yulan He**
[1]Department of Computer Science, University of Warwick, UK
[2]Queen-Mary University of London, UK
[3]Alan Turing Institute, UK
{j.Dougrez-Lewis,yulan.he}@warwick.ac.uk
{m.liakata,e.kochkina}@qmul.ac.uk

Abstract

With the rapid growth of social media in the past decade, the news are no longer controlled by just a few mainstream sources. Users themselves create large numbers of potentially fictitious rumours, necessitating automated veracity classification systems. Here we present a novel approach towards automatically classifying rumours circulating on Twitter with respect to their veracity. We use a model built on Variational Autoencoder which disentangles the informational content of a tweet from the manner in which the information is written. This is achieved by obtaining latent topic vectors in an adversarial learning setting using the auxiliary task of stance classification. The latent vectors learnt in this way are used to predict rumour veracity, obtaining state-of-the-art accuracy scores on the PHEME dataset.[1]

1 Introduction

Anyone can publish rumours online with the potential to influence and pose as news. Since it is impossible to manually check the vast volume of circulating tweets, there is increasing need for machine learning algorithms to assist with rumour veracity assessment.

Given a rumour of unknown veracity introduced by a tweet in a conversation thread and the responses to it, our goal is to automatically determine the veracity of the rumour by assigning it one of the classes *true*, *false*, or *unverified*. Prior approaches to rumour veracity classification have primarily relied on careful feature engineering. For example, Li et al. (2019a) used meta-features such as user credibility together with more traditional features to top the leaderboard in SemEval 2019 Task 7 (Gorrell et al., 2019). This task encouraged teams to use the stances of responses to the rumour to assist in veracity classification, which has previously been shown to be predictive of rumour veracity (Dungs et al., 2018). A number of approaches (Kochkina et al., 2018; Li et al., 2019b) also showed benefits of using stance classification as an auxiliary task in a multitask learning setup. Some recent approaches exploit the structure of the conversation discussing a rumour. Kochkina et al. (2018) used LSTM to model linear branches extracted from the conversation tree, while Ma et al. (2018) and Bian et al. (2020) modelled a tree structure to capture information from responses.

Zeng et al. (2019) presented an unsupervised approach built on Variational Autoencoder (VAE) to jointly model topic content and discourse behaviour in microblog conversations. We propose a novel architecture which incorporates a VAE with adversarial learning to disentangle topics which are informative for stance classification from those which are not. We then derive tweet representations based on the word representations learned in the latent stance-dependent topic space. Our results show that using such tweet representations for rumour veracity classification achieves superior performance on the PHEME dataset. In summary, we have made the following contributions:

- We have proposed a disentanglement based approach to rumour veracity classification which achieves state-of-the-art performance for classifying rumours from previously unseen events, as they would emerge in real life.
- We have demonstrated that there remains significant overlap between separate rumourous events with distinct vocabularies, facilitating transfer learning between such events.

---

[1] The code is available at: https://github.com/JohnNLP/SAVED

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3902–3908
August 1–6, 2021.

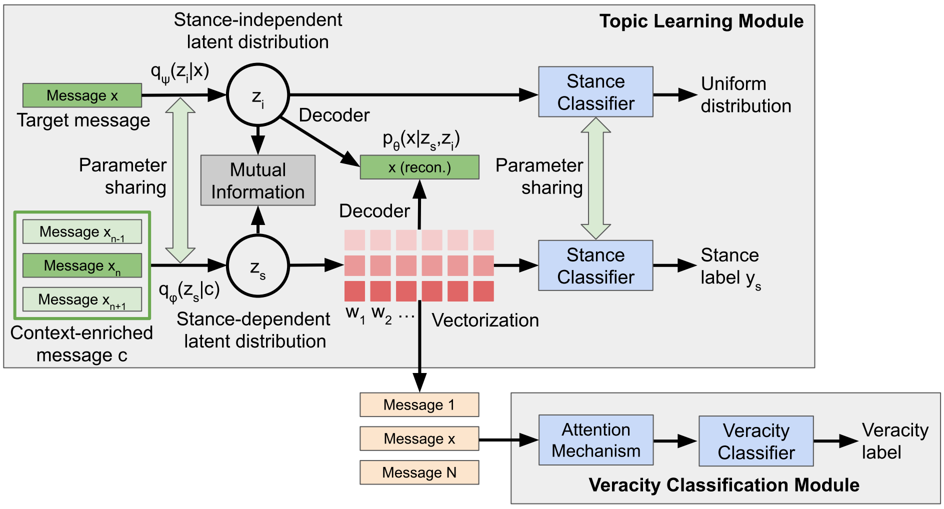

Figure 1: Model Architecture. Note the separate *Topic Learning* and *Veracity Classification* modules, and that the context-enriched message often uses a fixed size window of messages rather than the full conversation.

2 Proposed Model

Our proposed Stance-Augmented VAE Disentanglement framework (SAVED) is shown in Figure 1. It consists of two main modules, the *Topic Learning* module and the *Veracity Classification* module.

The idea is to separate the factual content of twitter rumours from their mannerisms[2], using the latter to predict rumour veracity. This technique is well-suited to rumour veracity prediction for emergent real-life events as it overcomes their factual distinctness - we hypothesise that mannerisms transfer better between different rumourous events. The PHEME dataset is designed for this purpose, with rumours grouped together according to their originating event. In our experiments we use both the source tweet originating the rumour and its conversation thread together since both are useful for veracity prediction.

2.1 Topic Learning

In microblog conversations, a source tweet could have multiple responses, forming a conversation tree. Here we flatten the tree into a chronologically ordered sequence of tweets, defined as $\boldsymbol{d} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where N denotes the total number of posts in the conversation. For each message $\boldsymbol{x}_n \in \boldsymbol{d}$, we construct its context-enriched message by taking a window of M messages before and M messages after the target message, denoted as $\boldsymbol{c}_n = \{\boldsymbol{x}_{n-M}, \cdots \boldsymbol{x}_{n-1}, \boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \cdots, \boldsymbol{x}_{n+M}\}$.

Assuming a post $\boldsymbol{x}$[3] is associated with a stance label y and each post can be generated from a stance-dependent latent topic $\boldsymbol{z}_s$ (determined by the context-enriched message $\boldsymbol{c}$) and a stance-independent latent topic $\boldsymbol{z}_i$, we aim to learn a model which maximises the joint data and label log-likelihood, $\log p(\boldsymbol{x}, y)$:

$$\log p(\boldsymbol{x}, y) = \log \int_{\boldsymbol{z}_s} \int_{\boldsymbol{z}_i} p(\boldsymbol{x}, y, \boldsymbol{z}_s, \boldsymbol{z}_i) d\boldsymbol{z}_s d\boldsymbol{z}_i$$
$$\geq E_{q_\phi(\boldsymbol{z}_s|\boldsymbol{c},y),q_\psi(\boldsymbol{z}_i|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}_s, \boldsymbol{z}_i)]$$
$$+ E_{q_\phi(\boldsymbol{z}_s|\boldsymbol{c},y),q_\psi(\boldsymbol{z}_i|\boldsymbol{x})}[\log p_\theta(y|\boldsymbol{z}_s)]$$
$$- KL(q_\phi(\boldsymbol{z}_s|\boldsymbol{c}, y)||p(\boldsymbol{z})) - KL(q_\psi(\boldsymbol{z}_i|\boldsymbol{x})||p(\boldsymbol{z}_i))$$

Following the idea of Zeng et al. (2019), we can compute a variational approximation to an intractable posterior using MLPs. We aim to minimise the reconstruction loss for each context-enriched message $\boldsymbol{c}$ and for each message $\boldsymbol{x}$ (see Figure 1), with a Monte Carlo approximation using L independent samples:

$$\mathcal{L}_{\boldsymbol{c}} \approx \frac{1}{L} \sum_{l=1}^{L} \sum_{n=1}^{N} \log p(c_n|\boldsymbol{z}_s^{(l)}) - KL(q_\phi(\boldsymbol{z}_s|\boldsymbol{c}, y)||p(\boldsymbol{z}_s)) \tag{1}$$

$$\mathcal{L}_{\boldsymbol{x}} \approx \frac{1}{L} \sum_{l=1}^{L} \sum_{n=1}^{N} \log p(x_n|\boldsymbol{z}_s^{(l)}, \boldsymbol{z}_i^{(l)}) - KL(q_\psi(\boldsymbol{z}_i|\boldsymbol{x})||p(\boldsymbol{z}_i)) \tag{2}$$

We assume that the latent stance-independent topics, $\boldsymbol{z}_i$, are independent of stance classes, and hence, when feeding into a stance classifier, should generate a uniform stance class distribution (similar to adversarial learning). On the contrary, the

---

[2] For an example of this, see Figure 5 in (Zeng et al., 2019).

[3] We drop the subscript n for clarity here.

latent stance-dependent, $\boldsymbol{z}_s$, should bear essential information to discriminate between stance classes. Therefore, we can define the following two cross-entropy losses for stance classification:

$$\mathcal{L}_{adv} = -E_{q_\phi(\boldsymbol{z}_i)} \sum_{s=1}^{S} \frac{1}{S} \log p(\hat{y}_s|\boldsymbol{z}_i) \tag{3}$$

$$\mathcal{L}_{stance} = -E_{q_\psi(\boldsymbol{z}_s)} \sum_{s=1}^{S} y_s \log p(\hat{y}_s|\boldsymbol{z}_s) \tag{4}$$

where S is the total number of stance classes, $\frac{1}{S}$ represents the uniform stance class distribution.

To disentangle the latent stance-independent topics, $\boldsymbol{z}_i$, and and latent stance-dependent topics, $\boldsymbol{z}_s$, we minimise the mutual information between them:

$$\mathcal{L}_{MI} = E_{q_\phi(\boldsymbol{z}_i)q_\psi(\boldsymbol{z}_s)} \log \frac{p(\boldsymbol{z}_i, \boldsymbol{z}_s)}{p(\boldsymbol{z}_i)p(\boldsymbol{z}_s)} \tag{5}$$

Our final objective function is:

$$\mathcal{L} = \mathcal{L}_{\boldsymbol{c}} + \mathcal{L}_{\boldsymbol{x}} + \alpha\mathcal{L}_{adv} + \beta\mathcal{L}_{stance} + \gamma\mathcal{L}_{MI} \tag{6}$$

where α, β and γ control the relative contribution of various loss functions.

We exclude the source tweet from this input since it is worded differently from the other tweets and is hence detrimental to overall performance.

2.2 Veracity Module

Once the Topic Learning Module is trained, we use the decoder weights, i.e., weights linking between the stance-dependent latent vector z_s and the bag-of-words representation of the reconstructed message (see the shaded pink blocks in Figure 1) to generate the input to the Veracity Module by deriving a vector for each tweet in the conversation tree - the mean average of the tweet's constituent words. This two-module approach is advantageous since it allows tweets to be weighted based on their position in the (unflattened) conversation tree, and forgoes use of the window of size M employed by the Topic Module. Vectorised tweets are grouped into 3 different importance tiers based on their positions in the conversation tree: the source tweet, direct responses, and all other responses. The rationale for this is in the observation that the later responses in long conversations tend to be less relevant to the verification of the rumour (Zubiaga et al., 2016). The Veracity Module itself consists of an attention layer followed by two dense hidden layers with leaky ReLU activation (Maas et al., 2013). The attention layer learns weights for each of the tweet tiers. The conversation representation is then obtained as a weighted average: $(\boldsymbol{w}_1\boldsymbol{v}_1 + \boldsymbol{w}_2\boldsymbol{v}_2 + \boldsymbol{w}_3\boldsymbol{v}_3)/n$, where $\boldsymbol{w}_t$ denotes the learnable weight for tier $\boldsymbol{t}$, $\boldsymbol{v}_t$ denotes the sum of the vectors of tweets in $\boldsymbol{t}$ and n denotes the number of tweets in the entire tree. In our experiments the presence of these weights increases model performance. We found that the model tends to give the highest weight to the source tweet, followed by direct responses and finally the rest. To mitigate the class imbalance, the loss attributed to instances of each class is weighted inversely to their frequency in the training data.

3 Experimental Setup

Dataset We use the PHEME-5 dataset (Kochkina et al., 2018), which consists of Twitter rumours around 5 high profile real-world events. Statistics regarding the dataset can be found in the Appendix. The PHEME dataset was chosen as it is a particularly challenging dataset due to class imbalance and a leave-one-event out cross validation setting, reflecting a real-world evaluation scenario.

Baseline Models We perform comparison of the proposed model SAVED with existing state-of-the-art models (Kochkina et al., 2018; Li et al., 2019b; Cheng et al., 2020) and several strong baselines, described in this section.

BERT We use the pretrained BERT$_{\text{BASE}}$ (Devlin et al., 2019), uncased, which consists of 12 self-attention layers, and returns a 768-dimension vector representation of a sentence. We generate BERT representations for each tweet in the conversation before feeding them into the Veracity Module.

VAED is a version of SAVED, where the Topic Learning Module is reduced to only its VAE component, without the stance classifiers from Section 2.1. The loss is $\mathcal{L}_{\boldsymbol{c}} + \mathcal{L}_{\boldsymbol{x}} + \gamma\mathcal{L}_{MI}$.

VAED Without Disentanglement (VAE) is a simplified version of VAED, where the Topic Learning Module is reduced to only using loss from the context-enriched latent factor without the target message (disentanglement). The loss is $\mathcal{L}_{\boldsymbol{c}}$.

VAED With Veracity (VAED+V) is an end-to-end classifier based on the Topic Learning Module, in which the context-enriched segment of the VAED z_s is connected to the veracity classifier. This model does not include a stance classifier, nor the adversarial component. The loss is $\mathcal{L}_{\boldsymbol{c}} + \mathcal{L}_{\boldsymbol{x}} + \beta\mathcal{L}_{veracity} + \gamma\mathcal{L}_{MI}$.

Model	False	True	Unverified	Accuracy	MacroF1
Kochkina et al. (2018)	0.212	**0.647**	0.330	0.492	0.396
Cheng et al. (2020)	**0.504**	0.480	0.465	0.521	**0.484**
Li et al. (2019b)	-	-	-	0.483	0.418
BERT Baseline	0.113	0.592	0.326	0.405	0.345
VAE	0.201	0.413	0.407	0.395	0.339
VAED	0.206	0.474	0.388	0.380	0.362
VAED+V	0.273	0.418	0.420	0.389	0.376
SAVED	0.164	0.642	**0.531**	**0.528**	0.434

Table 1: Comparison with baselines and previous results. For comparability, we pool together the results of all five events *before* calculating any F1 scores - the same approach used by the prior work in this table.

Parameter Settings The dimensionality of each latent factor was tuned via grid search, with peak performance found at 10 dimensions for the latent stance-dependent vector, z_s, and 6 dimensions for the latent stance-independent vector, z_i. See Appendix for further details.

Evaluation Metrics For comparability with prior work, F1 scores are calculated *after* combining the results of each fold.

4 Experimental Results

Overall Results The results of our experiments are shown in Table 1. All of the models outperformed the VAE baseline. The VAED model alone (with disentanglement, without any other modifications or stance/veracity classifiers) scores 0.363, showcasing the efficacy of disentanglement *per se*. The BERT-based model only outperformed VAE.

The proposed SAVED model outperforms those of prior work on overall accuracy and the *True* and *Unverified* classes. However, results for the *False* class are rather low - which is in fact the case for most of the models in Table 1, with only Cheng et al. (2020) being an exception. Results of VAED+V are lower than that of SAVED, in line with the knowledge that stance is related to veracity (Dungs et al., 2018). This suggests that stance is also a worthwhile intermediate classification target.

Event	False	True	Unverified	MacroF1
Charlie Hebdo	0.223	0.505	0.324	0.351
Ferguson	0.129	0.080	0.906	0.372
Germanwings Crash	0.033	0.520	0.289	0.281
Ottawa Shooting	0.058	0.735	0.119	0.304
Sydney Siege	0.157	0.700	0.140	0.332
Overall	**0.164**	**0.642**	**0.531**	**0.434**

Table 2: Per-fold evaluation results of SAVED.

Per-fold Results Table 2 shows the per-fold results in our leave-one-event-out setting. Interestingly, the model tends to perform best on rumors of *True* veracity and worst on those which are *False*. Performance on the *Unverified* class is adequate, except for the 'Ferguson' event in which the model F1 score is 0.906. Overall, the F1 scores of *True* and *Unverified* classes are rather high (0.642 and 0.531 respectively) compared with that of the *False* class (0.164).

Ablation studies The results of ablation studies are shown in Table 3, which were obtained by varying which latent factor of the Topic Learning Module were fed to the Veracity Module. We found that stance-dependent latent vectors performed better than stance-independent ones for rumour veracity classification, although each performed adequately. Creating an ensemble of both latent vectors was not helpful since the respective models had similar strengths and weaknesses.

Components Used	MacroF1
Stance-dependent	0.434
Stance-independent	0.375
Both together	0.395

Table 3: Varying the latent factor used by the Veracity Classification Module of SAVED.

Visualisation To examine the effectiveness of our conversation tree representations derived from the Topic Learning Module, we visualised the representative vector for each tree using t-SNE (van der Maaten and Hinton, 2008). This was done in the context of the model SAVED (See Sec. 2). Since learned tree-position weights $\boldsymbol{w}_t$ are an important part of generating the representation of the conversation but are not part of the Topic Learning Module, we obtained them from the attention component of the veracity module. Figure 2 depicts the resulting clusters of points, with each cluster roughly corresponding to a class. The "*Unverified*" cluster is particularly distinct. This cluster is largely comprised of tweets from the 'Ferguson' event, which contains most of the unverified rumours in the dataset. This further demonstrates that our Stance-Augmented VAE Model generates representations which are useful for veracity prediction.

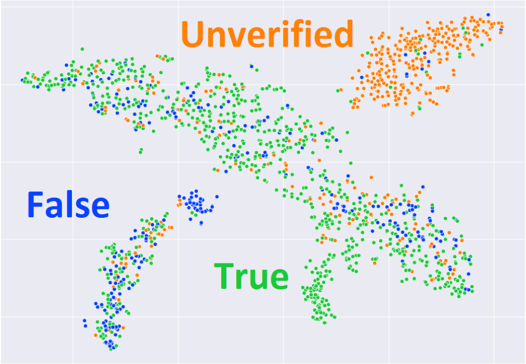

Figure 2: Visualisation of the stance-dependent latent factor for the 'Charlie Hebdo'-excluded fold.

Number of Responses and Performance We investigated how the number of responses to a rumour affect the model's accuracy. Whilst rumours with more responses provide more context for the model, responses too far down the response-tree have been noted to lose some of their relevance to the source rumour. We found the average number of responses for correctly classified rumours from the 'Charlie' fold, where 'Charlie Hebdo' is used as a testing set, to be 14.8 tweets, whilst incorrectly classified ones had an average length 12.2. This seemed stronger for the 'Ferguson' fold with averages of 22.2 and 14.7 respectively (note that the model only observes up to 20 responses per thread).

Using the 2-tailed Mann-Whitney test, these results approach statistical significance (p=0.08 for each fold). If there is an effect here, it can partially be explained by rumours with 3 or fewer responses, since when these were excluded the averages became 17.1 vs 15.4 for 'Charlie' and 25.2 vs 20.2 for 'Ferguson'.

Error Analysis Our model, similar to those of Kochkina et al. (2018) and others but not Cheng et al. (2020), fails to perform well for the *False* class. To our knowledge, this under-performance has not been previously investigated.

The PHEME dataset is imbalanced, containing 1,012 *True*, 393 *False* and *571* Unverified rumours. Although we attempted to account for this imbalance by weighting the model's loss inversely to class frequency, it is possible that the imbalance contributed to the poor performance on *False*. Interestingly more rumours were classified as *Unverified* than as *False*, although there was no clear pattern of misclassification.

The numbers of responses are largely unchanged when restricted to the *False* class alone, with 13.0 (correct) and 10.7 (incorrect) for the 'Charlie' fold, so this does not explain the performance deficit.

Manual investigation of the rumours themselves led to the observation that correctly classified *False* rumours tended to be more straightforward than those which were incorrectly classified. The latter seemed more likely to be vague or contain multiple parts, examples of which can be found in the Appendix. Thus a *False* rumour may contain both false and true statements, potentially lowering the utility of user responses for classification. Accordingly, the model by Cheng et al. (2020) appears to rely less on responses than ours and that of Kochkina et al. (2018).

5 Conclusion

We present a novel disentanglement-based approach to rumour veracity classification, achieving state-of-the-art results for accuracy, towards classification on previously unseen events from the PHEME dataset. Our results suggest that although unique events each have their own vocabulary, there is still sufficient common ground between them for stance-dependent driven rumour veracity classification to be effective.

Acknowledgements

This work was supported by a UKRI/EPSRC grant (EP/V048597/1) to Profs Yulan He and Maria Liakata as well as project funding from the Alan Turing Institute, grant EP/N510129/1. JDL was funded by the EPSRC Doctoral Training Grant. ML and YH are supported by Turing AI Fellowships (EP/V030302/1, EP/V020579/1).

References

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of* *the AAAI Conference on Artificial Intelligence*, volume 34, pages 549–556.

Mingxi Cheng, Shahin Nazarian, and Paul Bogdan. 2020. Vroc: Variational autoencoder-aided multi-task rumor classifier based on text. In *Proceedings of The Web Conference 2020*, WWW '20, page 2892–2898, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3360–3370, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 845–854, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Quanzhi Li, Qiong Zhang, and Luo Si. 2019a. eventAI at SemEval-2019 task 7: Rumor detection on social media by exploiting content, user credibility and propagation information. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 855–859, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Quanzhi Li, Qiong Zhang, and Luo Si. 2019b. Rumor detection by exploiting user credibility information, attention and multi-task learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1173–1179, Florence, Italy. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia. Association for Computational Linguistics.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

Laurens van der Maaten and Geoffrey Hinton. 2008. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Jichuan Zeng, Jing Li, Yulan He, Cuiyun Gao, Michael R Lyu, and Irwin King. 2019. What you say and how you say it: Joint modeling of topics and discourse in microblog conversations. *Transactions of the Association for Computational Linguistics*, 7:267–281.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

A Dataset

Table A1 shows the overall statistics of the PHEME dataset. It can be observed that the dataset is highly imbalanced with nearly 50% 'True' instances and relatively few 'False' instances.

Event	#Rumours	#True	#False	#Unverified
Charlie Hebdo	458	193	116	149
Ferguson	284	10	8	266
Germanwings Crash	238	94	111	33
Ottawa Shooting	470	329	72	69
Sydney Siege	522	382	86	54
Total	1972	1012	393	571

Table A1: Dataset (PHEME-5) overview.

B Data Preprocessing

We perform pre-processing on the PHEME data by using special tokens for URLs, username, hashtags, and numbers. We also lowercase text and expand words with apostrophes (e.g., '*we're*' becomes '*we are*'). The Topic-Learning module also excludes words which occur fewer than 20 times throughout the training corpus.

C Model Details

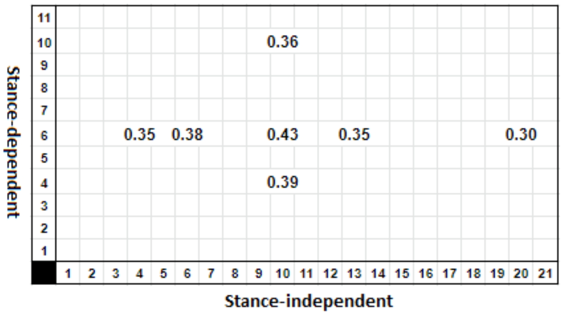

Figure A1: Grid search of the dimensions of the latent stance-dependent and stance-independent vectors.

The Topic-Learning Model has 15M parameters and takes around 6 hours to train on a computer with RTX 1080 Ti. The Veracity Model takes around 15 minutes to train on the same machine. We perform grid search on the dataset to identify the optimal setting of the dimensions of latent stance-dependent and stance-independent vectors, with the dimension of the former varying between 4 and 20, while the dimension of the latter varying between 4 and 10. The results are shown in Figure A1. The veracity classification results are obtained by evaluating on the validation set. It can be observed that the model achieves the best result when the dimension of the latent stance-dependent vector is set to 10 while that of the latent stance-independent vector is set to 6. For further model details, such as layer sizes and activations, the reader is advised to look at the linked source code.

D False Rumour Examples

Note that a rumour being of a certain type (as below) does not guarantee its predicted class or the correctness of its classification.

D.1 Basic (classified correctly)

breaking three gunmen involved in attack on charlie hebdo magazine , french interior minister bernard cazeneuve says . URL

D.2 Vague (classified incorrectly)

HASHTAG banksy's response to today's incident in paris via his official HASHTAG instagram acct URL HASHTAG charliehebdo URL

D.3 Multi (classified incorrectly)

two police officers have been injured in a shooting in HASHTAG montrouge in southern HASHTAG paris - there is no direct link with the HASHTAG charliehebdo attack

Perceptual Models of Machine-Edited Text

Elizabeth M. Merkhofer,[1] **Monica-Ann Mendoza,**[1]
Rebecca Marvin[2] and **John C. Henderson**[1]
[1] MITRE
7525 Colshire Dr, McLean, VA 22102
[2] Independent
{emerkhofer,mamendoza,jhndrsn}@mitre.org, rebecca.c.marvin@gmail.com

Abstract

We introduce a novel dataset of human judgments of machine-edited text and initial models of those perceptions. Six machine-editing methods ranging from character swapping to variational autoencoders are applied to collections of English-language social media text and scientific abstracts. The edits are judged in context for detectability and the extent to which they preserve the meaning of the original. Automated measures of semantic similarity and fluency are evaluated individually and combined to produce composite models of human perception. Both meaning preservation and detectability are predicted within 6% of the upper bound of human consensus labeling.

1 Introduction

Machine-editing systems produce new versions of text using text as input. They contribute to tasks such as automatic summarization, simplification, natural language generation and generative adversarial NLP systems. These tasks have communicative goals, for example shorter, more accessible, or more appropriate text, and system developers are encouraged to improve their correlation with human performance on these tasks. While the measured task performance of machine-editing systems continues to improve, one might consider how humans perceive machine-edited text compared to human-produced text. One-off human evaluation of editing systems is expensive, incomparable, and must be constantly repeated. In this work we make first attempts at direct, general-purpose modeling of human perception of these texts and develop a model of human perception as it relates to two goals: to maximally maintain the meaning of an original and be minimally perceptible as machine output.

We present a dataset of human judgments about detectability and meaning preservation for machine-edited text.[1] This dataset consists of 14,400 judgments about contextualized pairs of machine-edited sentences. The original texts are English-language and come from two domains: scientific papers and social media. The edits are created by six different algorithms using a variety of techniques. By comparing trivial editors to more subtle approaches under the same evaluation framework, we move toward generic models of perception of edited text.

Our analysis finds high interannotator agreement and examines human preference among the six machine editors that generated the candidates. Existing measures of similarity and fluency are evaluated as models of perception. We find that reference-informed models come close to human consensus of meaning preservation and detectability. However, language models that don't have access to the reference text have less success as generic models of detection. This dataset and analysis constitute a step toward modeling meaning preservation and detectability under a variety of machine-editing conditions representative of the state of the practice.

2 Background

Machine editing is a component of multiple tasks that balance meaning preservation and fluency differently.

2.1 Machine-Editing Tasks

Text simplification (Saggion, 2017) and summarization (Narayan et al., 2018) produce new versions of text that are simpler or shorter, intended to be useful to a human reader. Evaluation measures informativeness relative to a reference. Abstractive techniques that fully rewrite the text have recently

---

[1] `https://github.com/mitre/hpmet`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3909–3920
August 1–6, 2021. ©2021 Association for Computational Linguistics

become viable alternatives to extractive techniques that build new texts from portions of the original text.

Paraphrase generation is the task of producing semantically equivalent variants of a sentence and can underlie applications like question answering and data augmentation. Recent approaches include a component that generates alternatives and a component that estimates their quality as paraphrase (Li et al., 2018; Kumar et al., 2020).

Natural language watermarking of text (Venugopal et al., 2011) and text steganography (Wilson et al., 2014) are conditional text generation practices that require both a meaningful surface form and the hidden encoding of additional information. In this case, it's essential that the text appear plausible as readers should not suspect the encoded information (Wilson et al., 2015).

Edited texts are used in adversarial learning and attacks for text processing systems. Adversarial inputs change a system output without altering some relevant aspect of human perception of the text, e.g. sentiment when attacking a sentiment analysis system (Alzantot et al., 2018). In cases of adversarial learning, where edited texts are used only to promote system robustness, human perception is not a concern (Jia and Liang, 2017). In contrast, adversarial attack vectors rely on human perception of the attack, whether it be communicating meaning regardless of detectability (Eger et al., 2019) or guaranteeing fluency (Zhang et al., 2019a). While authors have quantified the effect of adversarial perturbations on metrics of text quality like word modification rate and count of grammatical errors (Zeng et al., 2020), the relation of these automatic metrics to human perception is not yet studied.

2.2 Meaning Preservation

Machine editing often aims to guarantee semantic similarity or *meaning preservation* between input and output. Meaning preservation can be insensitive to surface forms such as tokenization, casefolding, stylistic variation in punctuation, spacing, font choice, and tense. Compact text representations (e.g. Morse code) tend to regularize all potential surface forms.

Semantic textual similarity and paraphrase identification are active areas of investigation in the NLP community (Cer et al., 2017). Natural language inference (NLI) also relies on notions of semantic similarity to recognize a larger set of relations between texts (Bowman et al., 2015). These subfields of NLP investigate semantic relatedness between *human-authored* texts.

Meaning preservation is related to the concept of *informativeness* used in automatic summarization and *adequacy* for machine translation. Summarization metrics tend to lean toward recall to make sure the central concepts of reference summaries are produced and MT metrics tend to lean toward precision to penalize systems that generate something outside of the references.

Many adversarial text editors don't require strict paraphrase, but simply that their perturbations not change the input's classification to a human reader (Ren et al., 2019; Lei et al., 2019; Alzantot et al., 2018; Ebrahimi et al., 2018). Other authors ask judges about similarity to the unperturbed original (Zhao et al., 2018; Alzantot et al., 2018; Ribeiro et al., 2018; Jin et al., 2020). New work correlates automatic metrics with human judgments capturing both semantic similarity and fluency about three word- and character-swapping algorithms (Michel et al., 2019).

2.3 Detectability

Language models were introduced early in both automatic speech recognition (Bahl et al., 1983) and statistical machine translation (Brown et al., 1990) to make output text more readable. They aimed to avoid decoding results that appeared computer-generated.

Recent work in several natural language generation tasks augments automatic evaluation, which approximates informativeness, with one-off human evaluations that estimate text quality. Authors elicit judgment for abstractive summaries about readability (Paulus et al., 2018), fluency (Hardy and Vlachos, 2018), and preference between human and machine-written abstracts (Fan et al., 2018). Desai et al. (2020) elicit human judgements of grammaticality for a compressive summarization system that deletes plausible spans. In image captioning and dialogue systems, several learned metrics judge system output to be higher quality when it is less distinguishable from human text (Cui et al., 2018; Lowe et al., 2017)

Several methods of generating adversarial text have been evaluated through surveys of human perception, for example by asking humans to detect the location of machine edits (Liang et al., 2018), or to judge the likelihood that a sentence is mod-

	arxiv	reddit
item count	7200	7200
total sentence lengths	177 078	140 453
mean length	24.7	19.5
mean context length	96.8	265.9

Table 1: Dataset size in words.

Which better preserves the meaning of the reference?

Reference: Later on that day I emailed the company that I purchased my order from and they confirmed it was delivered to that address.
A. Later that day I emailed the company I bought my order, and they confirmed that was delivered to that address.
B. Later in that time i received the website and i sent my email from what it said it was delivered for customer address.

Location: Florida I didn't know what to flair. About a month ago a package I ordered was delivered to my old apartment complex. When I went to the front office to ask if a package with my name was turned in they said no such thing had occurred. ________ I don't know how to move forward from this.

Which sentence reads more like it was altered by a machine?

A. When thi applied voltage is ifcreased to a few mV we find a strong declease of the spin injection efficiency.
B. While the required voltage is required to a tunnel voltage to obtain a lower amount of the joule injection injection.

Semiconductor spintronics will need to control spin injection phenomena in the non-linear regime. In order to study these effects we have performed spin injection measurements from a dilute magnetic semiconductor [(Zn,Be,Mn)Se] into nonmagnetic (Zn,Be)Se at elevated bias. ________ The observed behavior is modelled by extending the charge-imbalance model for spin injection to include band bending and charge accumulation at the interface of the two compounds. We find that the observed effects can be attributed to repopulation of the minority spin level in the magnetic semiconductor.

Figure 1: Sample prompts from the meaning preservation task on Reddit (top) and the detection task on ArXiv (bottom.)

ified by a machine (Ren et al., 2019) or written by a human (Lei et al., 2019). Other authors ask human annotators about proxies like grammaticality (Jin et al., 2020), fluency (Zhang et al., 2019b) or readability (Hsieh et al., 2019) as a proxy for detectability.

Far more work asks whether computers can detect machine-edited text. Research on text generated with large language models finds that the output is easy to detect automatically because of the probabilities of the particular language model itself (Adelani et al., 2020; Gehrmann et al., 2019; Zellers et al., 2019). In fact, the generation setting that best fools humans produces output that is easy to detect automatically (Ippolito et al., 2020). This suggests human perception of such edits is different from machine detection.

Detectability and meaning preservation are not independent variables, but they represent different aspects of human perception. Destroying the fluency of a text can make it detectable as an edit in a high quality research document, but rewriting a section of chat in standard English can make it detectable in context. One can often transpose digits in scientific measurements to indetectably destroy meaning, and one could rewrite an abstract in randomized case patterns to raise suspicion without altering meaning.

3 Methods

3.1 Dataset Construction

We present a dataset of human judgments about two tasks, meaning preservation and detection, in each of two domains, social media and science writing. For each task and domain, we distributed packets of 600 multiple-choice questions to six judges. Each question was an AB test for a pair of editing systems both operating on a sentence in context. The first 105 questions of each packet were the same for all judges and are used to measure interannotator agreement. The remaining 495 sentences were the same, but the pairs of systems compared by judges varied. The judges were all native English speakers who work in AI research and were unfamiliar with the processes used to edit the original text.

Source sentences for the ArXiv dataset were randomly selected from all sentences in ArXiv abstracts submitted between its start in 1991 and the end of January, 2018. The Reddit sentences were randomly selected from all sentences in Reddit posts made in January, 2018. The two source collections were roughly the same size. Sentences less than 10 tokens or longer than 40 tokens were avoided in both collections to ensure judge productivity. To satisfy IRB and to minimize the likelihood of negative effects on judges, we excluded all posts from the subreddits listed on the official *nsfw* list, and any that were no longer reachable by September 2019. Table 1 describes statistics about the sentences selected for editing and the contexts provided for judges.

The meaning preservation task involved AB judgments on six different editing systems. For the detection task, we included the original texts among the edited variants for a total of seven systems. We refer to this as the `null` editor. An all-pairs design of six systems requires 15 pairs and an all-pairs design of seven systems requires 21 pairs. Both designs were iterated to yield 600 pairs of system variants, truncating the final seven system pattern early. The first 105 example editing pairs (seven full all-pairs sets for meaning and five for detection) were identical for all judges and the remaining 495 in each packet were chosen from the possible pairs according to *independent permutations* to encourage balance.

This can be described again for more clarity. $C(6,2) = 15$ (meaning preservation) and $C(7,2) = 21$ (detectability). 600 examples lines up perfectly with a 15-item boundary but not with a 210 item boundary. Thus, there are 12 examples left over from a complete set of all-pairs of 7 systems in detectability when truncating to 600 items per source per judge. The first 105 system pair assignments come from 7 cycles through $C(6,2)$ pairs or 5 cycles through $C(7,2)$. The remaining sequences of pairs for each judge are all-pairs cycles through independently randomized permutations of the systems. Machine edit assignment to positions A and B were independently shuffled for each judge and the questions were presented to each judge in randomly shuffled order. Judges were instructed to choose between the two alternatives.

Each item was presented as a choice between two edited versions of the same sentence, presented with the rest of the Reddit post or ArXiv abstract as context. Figure 1 shows examples and the specific prompts used to elicit judgments. In the meaning preservation example, the candidates were produced by round-trip machine translation and the VAE. In the detection example, the candidates were produced by charswap and the VAE. Cases where detection paired the `null` system against a machine editor were collected to determine how often each editor was preferred to the original.

Less than half of one percent of detectability items are automatically marked as ties. These include cases where the edited text is the same string as the original, disregarding casing and punctuation, or where the VIPER editor (described below) produced an alternative that rendered identically in packets. These are included in the analysis to capture the intuition that a perceptual model should score ties the same.

3.2 Machine-Editing Systems

We employ six editing systems to capture the effect that varied systems have on human perception. Each takes just the sentence to be edited, without context.

3.2.1 Swapping editors

Simple word- and character-swapping editors are prevalent in literature about adversarial attacks and data augmentation (Michel et al., 2019). Our **charswap** editor is inspired by several works in adversarial NLP that examine character swapping as a minimal change to text inputs that can degrade system performance (Belinkov and Bisk, 2018; Ebrahimi et al., 2018). Our implementation randomly swaps 1 to 3 lower-case ASCII characters per input for other ASCII characters, selecting the least likely of 100 alternatives under the GPT-2 language model (Radford et al., 2019).

VIPER is a character-swapping algorithm informed by visual closeness, inspired by a common strategy used to avoid keyword filters, for example in online forums Eger et al. (2019). The VIPER algorithm replaces random characters with their nearest neighbors among embeddings based on their glyph e.g. l→1 and 0→O. We further bias the open source implementation toward visual closeness by randomly swapping between 1 and 3 characters, with the probability of each swap weighted by its visual similarity.

The **AddCos** system uses word embedding distance to replace a single word with a paraphrase. The algorithm is adapted from a machine translation metric that measures the fit of words that are not in a reference, using the cosine similarity of the proposed replacement and the sum of vectors for sentence context (Apidianaki et al., 2018). We adapt the open source implementation as a machine editor, obtaining candidate replacements from the Penn Paraphrase Database (Ganitkevitch et al., 2013) and selecting the one best replacement.

3.2.2 Rewriting editors

Machine translation (MT) has recently become reliable and on-par with human translation capabilities in some cases (Bojar et al., 2018). We utilized **round trip MT** (from source English text to another language and then back) as a type of text editor. Three of the authors performed a blind assessment of approximately one hundred candidate languages available from an online MT provider and determined that $en \rightarrow pt \rightarrow en$ is a high-quality round trip route.

A **variational autoencoder (VAE)** learns a semantically meaningful latent space. We use an implementation[2] based on the model of Zhang et al. (2017) to train a VAE for each domain with 200,000 sequences of up to 40 tokens. Edits are obtained by encoding an original sentence and sampling from the latent distribution.

Syntactically controlled paraphrase networks (SCPNs) encode sentences and decode them according to a target constituency parse (Iyyer et al., 2018). Unlike swapping editors, this system introduces syntactic variation. Using the open source code and default templates, we generate ten paraphrases per sentence. We select the paraphrase with the best GPT-2 language model score.

3.3 Modeling Human Perception

We evaluate a set of automatic metrics as models of human perception. To test a metric as a model of the collected judgments, the metric scores each edited sentence and chooses the item in the pair with the better score. The choice is compared to the judge's preference.

In addition, we learn a combination system that scores sentences by weighting each component metric. One combination is learned for each task, using the data from both domains. The data is split into a training set of 80% used for fitting the combination, a validation set of 10% and a final test set of 10% of examples. For items repeated among judges, all six instances are assigned to the same partition.

Our objective function, maximizing agreement on AB tests, is neither continuous, smooth, nor particularly amenable to a logistic transform. We search for our mixture parameters using the Dlib `MaxLIPO+TR` Lipschitz function and trust region search algorithm (King, 2009). The model is optimized to minimize the errors in the training set with an L1 regularization term. A best combination is selected using forward feature selection and validation set accuracy.

4 Experiments

We examine text similarity and fluency metrics that originate from several tasks in NLP as possible models of human perception. We first present the portfolio of metrics we use.

4.1 Measures of Meaning Preservation

Levenshtein edit distance measures the minimum number of character operations needed to change one string into another (Levenshtein, 1966). We compute both the classical Levenshtein distance over *character* edits and *word* edits (WER).

NLP Task Metrics. We evaluated several metrics used to measure the quality of NLP system output compared to a human reference for tasks including machine translation, summarization, and image captioning. **BLEU** is a machine translation evaluation method based on word n-gram precision, with a brevity penalty (Papineni et al., 2002). The **METEOR** metric uses stemming and WordNet synsets to characterize acceptable synonymy in translation (Banerjee and Lavie, 2005). **CIDEr** also uses stemming and incorporates importance weighting for ngrams based on corpus frequency (Vedantam et al., 2015). The **ROUGE-L** metric, used in summarization and image captioning, is based on longest common subsequence between a reference and hypothesis (Lin, 2004). **ChrF** and variants like chrF++ compare bags of character and ngram substrings to capture sub-word similarity without language-specific resources (Popović, 2016, 2017). The **BEER** metric is trained to correlate with human judgment at a sentence level using features like character n-grams and permutation trees that are less sparse at that level (Stanojević and Sima'an, 2014).

Neural Network-based Similarities. Recent work uses trained, neural-network vector representations to quantify semantic similarity. We experiment with three based on BERT, a neural network trained on Wikipedia and the Google Books Corpus (Devlin et al., 2019). **BERTScore** computes an F1-based similarity score between the contextual embeddings for subword tokens in a candidate and reference sentence (Zhang et al., 2020). The metric can also be computed as **RoBERTaScore** using weights from RoBERTa pretraining (Liu et al., 2019). **BLEURT** fine tunes BERT to predict sentence-level machine translation quality scores (Sellam et al., 2020). **Sentence-BERT** measures similarity using a model finetuned with a paraphrase objective to create semantically meaningful sentence vectors that can be directly compared (Reimers and Gurevych, 2019).

---

[2] https://github.com/mitre/tmnt

	detect		meaning		preferred	
system	arxiv	reddit	arxiv	reddit	arxiv	reddit
addcos	**0.60**	0.48	0.74	0.64	0.04	0.03
mt	0.55	**0.59**	0.55	0.57	**0.21**	**0.10**
viper	0.54	0.56	**0.79**	**0.88**	0.00	0.01
charswap	0.46	0.46	0.67	0.67	0.01	0.01
scpn	0.21	0.35	0.17	0.18	0.02	0.06
vae	0.20	0.11	0.07	0.06	0.08	0.02

Table 2: Probability that system wins in A/B test or is preferred to original. Rows are sorted by first column.

4.2 Measures of Detectability

Detectability is a property of text in context, without regard for a reference. We evaluate language model scores, which measure fluency, as proxies for detectability.

We evaluate a **Kneser-Ney 5-gram** language model trained on a full English Wikipedia dump (Wikipedia contributors, 2020) with KenLM (Heafield, 2011). We estimate the model using modified Kneser-Ney smoothing without pruning. We also evaluate the language model score given by **GPT-2**, a large neural transformer-based language model trained on 8 million web pages (Radford et al., 2019). We use the technique described in Salazar et al. (2020) to obtain a **BERT Masked Language Model (MLM)** that accounts for the model's self-attention. We compute each language model score under two conditions: using only the edited sentence, and including one sentence before and after the edited sentence (*+context*).

Predictions from **BERT's Next Sentence Prediction (NSP)** task estimate the likelihood for a sequence of sentences. This classifier is trained to discriminate sequences of two sentences found in the pretraining corpus from sequences drawn using negative sampling (Devlin et al., 2019).

5 Results

Table 2 illustrates the relative success of the machine-editing systems. Success is measured using the number of A/B tests where an edit by the system was selected (for meaning) or the other item selected (for detectability), divided by the number of prompts involving the editor. *Preference* refers to only the portion of the detectability dataset where the edited text is compared to the original. The swapping algorithms are most often chosen as preserving meaning. The visual perturbations of VIPER have little effect on perception of meaning. The preference for these editors is not as strong on detectability items. For both tasks, the round-trip machine translation model is preferred in slightly over half of comparisons, while the VAE and SCPN perform quite poorly. One substantial difference in these conditional generation algorithms may be that MT is trained on web-scale data, while the others are trained in-house with relatively small datasets.

Among detectability items, human judges prefer an edited version over the original (`null` system) 4.9% of the time, 101 of 2054 relevant judgments. These prompts most commonly include the round trip machine translation editor, but all editing systems were preferred over the original at least once. Round-trip machine translation is picked over the original reference 20% of the time in ArXiv and 10% of the time in Reddit, suggesting that these outputs are more fluent or more typical for the domain than the original. For these items, the character swapping algorithms are most detectable.

5.1 Annotator Consistency

One hundred five prompts per task were presented to all judges to measure interannotator agreement. As judgments are made between constructed, randomized flips and pairwise tests, we compare to the raw prior of 50% agreement. For ArXiv, the probability of agreement among pairs of judges was 82.2% for meaning and 75.6% for detection. For Reddit, the probabilities were 86.7% and 75.9% respectively. The lower interannotator agreement in the science and technology domain may reflect lower familiarity with the subjects of the abstracts.

A consensus vote is determined by a plurality of the six judges, or randomly in cases of ties. The probability of agreement of a random judge with the consensus is reported in Table 3 as an upper bound for the performance of automatic systems.

5.2 Correspondence of Metrics with Human Judgment

Table 3 shows the correspondence of the best metrics with the 630 multiply-annotated prompts and includes the upper bound of human consensus. The table shows only metrics with accuracy within five items of the best. Table 4 shows agreement with the entire dataset and over the full set of systems tested. The meaning metrics are also evaluated as measures of detectability. At editing time, they can be used to estimate the detectability of a candidate edit. However, they are not practical as generic

		detect			meaning		tot
	arxiv	reddit	tot	arxiv	reddit	tot	
annotator consensus upper bound	0.838	0.848	0.843	0.884	0.917	0.900	0.871
max per source	0.781	0.794	0.788	0.843	0.870	0.857	0.822
RoBERTaScore/R	0.765	0.781	0.773	**0.837**	**0.870**	**0.853**	**0.813**
chrF	0.759	0.784	0.772	**0.841**	0.830	0.835	0.803
chrF/R	0.759	0.781	0.770	**0.835**	0.830	0.833	0.801
chrf++	0.759	**0.787**	0.773	0.824	0.830	0.827	0.800
BEER	0.733	**0.787**	0.760	0.808	**0.863**	0.836	0.798
METEOR	**0.775**	0.778	**0.776**	**0.843**	0.792	0.818	0.797
BERTScore/R	**0.775**	0.765	0.770	0.830	0.813	0.821	0.795
BERTScore/F	**0.775**	0.775	**0.775**	0.811	0.806	0.808	0.792
CIDEr	0.759	**0.787**	0.773	0.824	0.781	0.802	0.788
RoBERTaScore/F	0.762	**0.794**	**0.778**	0.795	0.781	0.788	0.783
BERTScore/P	**0.781**	0.756	0.768	0.798	0.768	0.783	0.776
RoBERTa MLM	**0.683**	**0.673**	**0.678**				
RoBERTa MLM +context	0.663	**0.676**	0.669				
GPT-2 +context	0.654	**0.679**	0.667				

Table 3: Probability of agreement of metrics with sets of 630 multiply-annotated human judgments. Consensus is an *upper bound*. Results differing from the best by five items or fewer are shown in bold.

		detect			meaning		tot
	arxiv	reddit	tot	arxiv	reddit	tot	
max per source	0.788	0.790	0.789	0.845	0.841	0.843	0.816
RoBERTaScore/R	0.781	0.773	0.777	0.826	0.835	0.831	**0.804**
chrF	0.763	0.754	0.758	0.832	**0.840**	**0.836**	0.797
BERTScore/R	0.773	0.758	0.766	**0.845**	0.813	0.829	0.797
chrF++	0.763	0.758	0.760	0.823	**0.841**	0.832	0.796
chrF/R	0.762	0.751	0.756	0.830	0.838	0.834	0.795
BERTscore/F	0.773	0.765	0.769	0.839	0.804	0.822	0.795
chrF++/P	0.763	0.764	0.764	0.815	0.833	0.824	0.794
ROUGE	0.775	0.763	0.769	0.825	0.813	0.819	0.794
chrF++/R	0.760	0.754	0.757	0.821	0.838	0.829	0.793
BLEU	0.765	0.765	0.765	0.817	0.824	0.821	0.793
chrF/P	0.767	0.765	0.766	0.814	0.826	0.820	0.793
BEER	0.753	0.758	0.756	0.810	**0.841**	0.826	0.791
Sentence-BERT	0.763	0.770	0.766	0.813	0.821	0.817	0.791
METEOR	0.779	0.766	0.772	0.828	0.791	0.809	0.790
edit distance	0.755	0.745	0.750	0.815	**0.840**	0.827	0.788
word error rate	0.766	0.757	0.762	0.810	0.817	0.814	0.788
BERTScore/P	0.775	0.766	0.770	0.822	0.779	0.800	0.785
CIDEr	0.769	0.758	0.764	0.817	0.793	0.805	0.784
RoBERTaScore/F	**0.788**	**0.790**	**0.789**	0.792	0.767	0.780	0.784
BLEURT	0.746	0.746	0.746	0.788	0.751	0.770	0.758
RoBERTaScore/P	0.773	0.781	0.777	0.730	0.700	0.715	0.746
RoBERTa MLM +context	0.663	**0.712**	**0.688**				
RoBERTa MLM	0.642	0.700	0.671				
GPT-2 +context	0.644	0.685	0.665				
BERT NSP $logit0(s_{i-1}, s_i)$	0.672	0.650	0.661				
BERT NSP $logit1(s_{i-1}, s_i)$	**0.676**	0.639	0.657				
BERT NSP $p(s_{i-1}, s_i)$	0.666	0.640	0.653				
BERT NSP $p(s_{i-1}, s_i)p(s_i, s_{i+1})$	0.651	0.636	0.643				
BERT NSP $logit0(s_i, s_{i+1})$	0.652	0.631	0.641				
BERT NSP $logit1(s_i, s_{i+1})$	0.654	0.628	0.641				
BERT MLM +context	0.640	0.638	0.639				
BERT NSP $p(s_i, s_{i+1})$	0.631	0.626	0.629				
BERT MLM	0.617	0.627	0.622				
GPT-2	0.586	0.642	0.614				
KenLM	0.565	0.568	0.567				

Table 4: Probability of agreement of metrics with 3600 human judgments. Results differing from the best by five items or fewer are shown in bold.

	arxiv		reddit		mean	
metric	accuracy	F1	accuracy	F1	accuracy	F1
max per source	0.902	0.252	0.895	0.229	0.899	0.241
RoBERTa MLM	0.870	0.231	0.862	0.220	0.866	0.226
GPT-2 +context	**0.902**	**0.252**	0.874	0.168	0.888	0.210
BERT MLM +context	0.867	0.243	0.802	0.158	0.835	0.201
BERT MLM	0.849	0.236	0.799	0.156	0.824	0.196
KenLM	0.767	0.216	0.691	0.150	0.729	0.183
GPT-2	0.806	0.213	0.812	0.150	0.809	0.181
BERT NSP $p(s_i, s_{i+1})$	0.828	0.234	0.764	0.123	0.796	0.178
BERT NSP $logit0(s_i, s_{i+1})$	0.749	0.194	0.722	0.112	0.736	0.153
BERT NSP $logit1(s_i, s_{i+1})$	0.754	0.202	0.714	0.104	0.734	0.153
RoBERTa MLM +context	0.862	0.078	**0.895**	**0.229**	0.879	0.153
BERT NSP $p(s_{i-1}, s_i)p(s_i, s_{i+1})$	0.793	0.196	0.714	0.104	0.754	0.150
BERT NSP $p(s_{i-1}, s_i)$	0.853	0.209	0.763	0.083	0.808	0.146
BERT NSP $logit1(s_{i-1}, s_i)$	0.762	0.164	0.718	0.088	0.740	0.126
BERT NSP $logit0(s_{i-1}, s_i)$	0.749	0.151	0.735	0.087	0.742	0.119
baseline	0.941	0.000	0.961	0.000	0.951	0.000

Table 5: Performance predicting which edits humans prefer over originals.

	train	dev	test
meaning			
combo	0.857	0.844	0.830
ChrF	0.839	0.831	0.785
detect			
combo	0.689	0.752	0.724
RoBERTa MLM +context	0.678	0.731	0.687

Table 6: Combo performance results.

	$\sigma\|w\|$	w	component
meaning	0.25	0.999	edit distance
	0.15	0.912	BERTScore/R
	0.013	0.0871	Sentence-Bert
	0.0044	0.0307	BERTScore/P
	0.0035	0.0046	BLEURT
	0.0018	0.0354	RoBERTaScore/R
	0.0012	0.0265	RoBERTaScore/F
	0.0012	0.0263	RoBERTaScore/P
detect	0.83	0.0161	RoBERTa MLM +context
	0.13	0.577	BERT NSP $p(s_i, s_{i+1})$
	0.11	0.462	BERT NSP $p(s_{i-1}, s_i)$
	0.0069	0.0106	GPT-2 +context

Table 7: Components in the detect and meaning combo systems ranked by influence ($\sigma|w|$).

detection models sniffing out machine edits in the wild where no original is available.

Several automatic metrics show good correspondence with meaning. The best systems include large, neural models intended to capture subtle synonymy as well as simple metrics like chrF. In general, the recall component of BERTScore-based metrics correlates better than the precision component. Though the BLEURT metric is trained to predict human judgements of translation quality, it seems a poor fit for perceptions of meaning preservation in our dataset.

Applied to the detection task, the reference-informed metrics also approach the upper bound of human consensus. Using RoBERTaScore as a single model of both meaning preservation and detectability reaches over 81% agreement with consensus.

The language model metrics fall behind in performance on detection but still perform well above the level of chance. We find that including additional context improves performance for the same system and the large, neural models greatly outperform the traditional 5-gram model. Across the board, models with RoBERTa training perform better than their BERT-based counterparts.

Table 5 shows performance for predicting the detection items for which the judge preferred the edited text to the original. A baseline system that always selects the original gets around 95% accuracy, but cannot identify an edited text that a human accepts as a substitute. All of the detection systems tested were able to identify some substitutable edits. The best overall are large language models with context, reaching 0.241 F1.

As shown in Table 6, learned combinations of metrics are able to achieve better performance than the single best metric for each task. The components of those systems are specified in Table 7, sorted by their importance in the combination as calculated by the product of the standard deviation of their values (σ) and the magnitude of their weights (w).

6 Conclusion

We introduced a novel dataset of human judgments of machine-edited texts and initial models of those perceptions. A portfolio of automated metrics was assessed for the ability to predict judges' preferences on meaning preservation and detectability. Automated measures of semantic similarity and fluency were evaluated individually and combined to produce factored models of human perception. Both meaning preservation and detectability are modeled within 6% accuracy of the upper bound of human consensus labeling. However, we observe that existing metrics poorly predict whether humans find an edited text to appear more human-like than the original.

Future work could explore deeper models and other factors of human perception not modeled by the metrics present here. For example, humans are sensitive to capitalization and correct spacing but many automatic metrics perform tokenization and normalization. Direct modeling of human perception drives understanding of human factors involving text variation. Adaptive models of human text perception would enable text editing to target understanding by individual readers.

References

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. 2020. Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection. In *International Conference on Advanced Information Networking and Applications*, pages 1341–1354. Springer.

Moustafa Alzantot, Yash Sharma Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Marianna Apidianaki, Guillaume Wisniewski, Anne Cocos, and Chris Callison-Burch. 2018. Automated paraphrase lattice creation for hyter machine translation evaluation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 480–485. Code found at `https://github.com/acocos/pp-lexsub-hytera/`.

L. R. Bahl, F. Jelinek, and R. L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. 2018. Learning to evaluate image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5804–5812.

Shrey Desai, Jiacheng Xu, and Greg Durrett. 2020. Compressive summarization with plausibility and salience modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6259–6274, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics. Code available at `https://github.com/AnyiRao/WordAdver`.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Code found at `https://github.com/miyyer/scpn`.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Davis E. King. 2009. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758. Global search described at `http://blog.dlib.net/2017/12/a-global-optimization-algorithm-worth.html`.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:329–345.

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros G Dimakis, Inderjit S Dhillon, and Michael Witbrock. 2019. Discrete adversarial attacks and submodular optimization with applications to text classification.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium. Association for Computational Linguistics.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4208–4215.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

Paul Michel, Xian Li, Graham Neubig, and Juan Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3103–3114, Minneapolis, Minnesota. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Maja Popović. 2016. chrF deconstructed: beta parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504, Berlin, Germany. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Horacio Saggion. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. 2014. BEER: BEtter evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz Och, and Juri Ganitkevitch. 2011. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1363–1372, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Wikipedia contributors. 2020. Wikipedia, the free encyclopedia. [Online; 20200720 enwiki dump, accessed 11-August-2020].

Alex Wilson, Phil Blunsom, and Andrew Ker. 2015. Detection of steganographic techniques on twitter.

In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2564–2569.

Alex Wilson, Phil Blunsom, and Andrew D Ker. 2014. Linguistic steganography on twitter: hierarchical language modeling with manual interaction. In *Media Watermarking, Security, and Forensics 2014*, volume 9028, page 902803. International Society for Optics and Photonics.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *arXiv preprint arXiv:1905.12616*.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2020. Openattack: An open-source textual adversarial attack toolkit. *arXiv preprint arXiv:2009.09191*.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019a. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019b. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4169–4179.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.

Scaling Within Document Coreference to Long Texts

Raghuveer Thirukovalluru[1], **Nicholas Monath**[1], **Kumar Shridhar**[2],
Manzil Zaheer[3], **Mrinmaya Sachan**[2]**and Andrew McCallum**[1]
[1]UMass Amherst, [2]ETH Zurich, [3]Google
`rthirukovall@umass.edu`, {`nmonath,mccallum`}`@cs.umass.edu`
{`shridhar.kumar,mrinmaya.sachan`}`@inf.ethz.ch`
`manzilzaheer@google.com`

Abstract

State of the art end-to-end coreference resolution models use expensive span representations and antecedent prediction mechanisms. These approaches are expensive both in terms of their memory requirements as well as compute time, and are particularly ill-suited for long documents. In this paper, we propose an approximation to end-to-end models which scales gracefully to documents of any length. Replacing span representations with token representations, we reduce the time/memory complexity via token windows and nearest neighbor sparsification methods for more efficient antecedent prediction. We show our approach's resulting reduction of training and inference time compared to state-of-the-art methods with only a minimal loss in accuracy.

1 Introduction

Recent advances in coreference resolution (Lee et al., 2018; Joshi et al., 2019, 2020; Wu et al., 2020) have been largely based on the end-to-end model proposed by Lee et al. (2017). However, these models are costly both in terms of training and inference time, as well as memory requirements, especially for long documents. The large computational cost makes the models infeasible to run for a typical user on large document collections in domains such as blogs, stories, books, etc. Moreover, a reduction in energy use of these models can be of benefit to cloud service providers' costs and there can also be environmental benefits (Strubell et al., 2019; Schwartz et al., 2020).

There are two main computational bottlenecks in using end-to-end coreference models on long documents: (i) span and span-pair representations for all spans in the document are simultaneously considered, and (ii) the coreference decision for a mention requires considering all candidate antecedent spans.

In this paper, we propose an approximation to the end-to-end coreference model (Lee et al., 2017) that scales to long documents by addressing both these bottlenecks. Our proposed approach operates at the token level instead of the span level, removing the quadratic dependence on the number of mention spans in a document and addressing bottleneck (i). We propose token level scoring functions for the bilinear inference model originally proposed by Lee et al. (2018). To address bottleneck (ii), we use token windows, token-level k-nearest neighbor relationships along with low-rank matrix approximations of the token similarity matrix thereby improving time/memory efficiency. We also propose an approach to drop token representations from memory to reduce memory requirements while still maintaining the accuracy.

We evaluate our approach on three coreference datasets: CoNLL-2012 (Pradhan et al., 2012), Litbank (Bamman et al., 2019), and MedMentions (Mohan and Li, 2019) and observe competitive accuracy to state of the art coreference models based on end-to-end training while achieving both faster training and inference running times. Our approach is also more memory efficient and up to 10x faster than the recently proposed memory-based incremental coreference resolution model on Litbank (Toshniwal et al., 2020b). Finally, we demonstrate the scalability of our approach by running it on a novel of two million tokens in 14 minutes while requiring just 12GB of GPU RAM, while previous work can only scale to documents of just around eleven thousand tokens even with up to 48GB of GPU RAM.

Concurrent to our work, Kirstain et al. (2021) also proposes a bilinear token level scoring function for coreference. The focus of our work however

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3921–3931
August 1–6, 2021.

is on long documents and we further introduce a token k-nn graph approximation, a low-rank matrix factorization and an approach to drop non essential candidate antecedents to improve mem/time scalablity.

2 Background: End-to-end Within-Document Coreference

End-to-end within-document coreference resolution models jointly discover a set of mentions, $\mathcal{M}$ in a document $\mathcal{D}$ and determine which of the mentions are coreferent. We use $\mathcal{D}$ to refer to the ordered set of tokens in the document $\mathcal{D} = \{x_1, x_2, \dots, x_T\}$. Each mention is a token span $s = x_i, \dots, x_j$[1]. We use $\boldsymbol{x}_i$ to refer to the contextualized embedding of the token i (see Section 4.1 for more details on the encoder). The model comprises of two parts which are jointly trained: (a) a mention-proposer, and (b) an antecedent-predictor. The mention-proposer model evaluates all spans $\mathcal{S}$ in the dataset and proposes a small set of potential mentions $\mathcal{M} \subset \mathcal{S}$. The antecedent prediction model evaluates the mentions suggested by the mention proposer and produces coreference clusters (chains) $\mathcal{C} \subset \mathcal{P}(\mathcal{M})$, where $\mathcal{P}(\cdot)$ is the powerset.

Recent work (Lee et al., 2018; Joshi et al., 2020; Xu and Choi, 2020, inter alia) has built upon the first neural, end-to-end coreference model (Lee et al., 2017). Each of these models introduce two scoring functions $\mathsf{s}_{\mathsf{m}}(s)$ and $\mathsf{s}_{\mathsf{a}}(m_1, m_2)$. $\mathsf{s}_{\mathsf{m}}(s)$ represents the scores that a span s is a mention, and $\mathsf{s}_{\mathsf{a}}(m_1, m_2)$ is the score for mention m_1 being an antecedent of mention m_2. These scoring functions are used to define the joint mention proposal and antecedent prediction model for coreference.

Mention proposer: The previous works use a neural network for $\mathsf{s}_{\mathsf{m}} : \mathcal{S} \to \mathbb{R}$. The architecture takes in a mention span and outputs a score. For each mention span s, the model computes a vector representation $\boldsymbol{g}_s \in \mathbb{R}^d$. The scoring functions take these vector representations as input:

$$\mathsf{s}_{\mathsf{m}}(s) = \boldsymbol{w}_{\mathrm{m}} \cdot \mathrm{FFNN}_{\mathrm{m}}(\boldsymbol{g}_s) \tag{1}$$

and $\boldsymbol{g}_s$ is computed as:

$$\boldsymbol{g}_s = [\boldsymbol{x}_{\mathrm{START}(s)}, \boldsymbol{x}_{\mathrm{END}(s)}, \hat{\boldsymbol{x}}_s, \phi(s)] \tag{2}$$

where $\boldsymbol{x}_{\mathrm{START}(s)}, \boldsymbol{x}_{\mathrm{END}(s)}$ are the boundary representations of span s, $\hat{\boldsymbol{x}}_s$ is a self-attention representation of span s, and $\phi(s)$ encodes the width (number of tokens) of span s. For efficiency, the model selects top $0.4T$ scoring mention spans where T is the number of tokens of the document. We refer to this set of selected mention spans as $\mathcal{M}$. We use an ordering of the mentions $m \in \mathcal{M}$ based on their start/end offsets.

Antecedent Prediction: Previous work has explored several models for antecedent prediction. The most computationally efficient being a bilinear scoring model (Lee et al., 2018):

$$\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}(m_1, m_2) = \mathbf{g}_{s_{m_1}} W^T \mathbf{g}_{s_{m_2}} \tag{3}$$

Higher-order inference models, which use deep models to capture coreference relationships between mentions, have also been considered (Lee et al., 2018). For example,

$$\mathsf{s}^{\mathsf{hoi}}_{\mathsf{a}}(m_1, m_2) = \boldsymbol{w}_{\mathrm{a}} \cdot \mathrm{FFNN}_{\mathrm{a}}([\boldsymbol{g}_{m_1}; \boldsymbol{g}_{m_2}; \quad \boldsymbol{g}_{m_1} \odot \boldsymbol{g}_{m_2}, \phi_{m_1,m_2}]) \tag{4}$$

We refer the reader to Xu and Choi (2020) for a detailed analysis of higher order inference models.

The prediction of the antecedent of each mention, which we refer to as *inference*, is done by *backwards chaining*. Clusters of mentions are determined by finding for each mention, the highest scoring antecedent among the mentions appearing earlier in the document and adding the mention to the antecedent's cluster. This can be described as finding the connected components of a graph G. Coarse-to-fine inference (Lee et al., 2018) and the standard bi-linear model can be differentiated by different ways of constructing the adjacency matrix of the graph G with nodes being the mentions $\mathcal{M}$. We refer to this adjacency matrix as A, and use $A_{i,j} = 1$ to indicate the existence of an edge between mention m_i and m_j. The adjacency matrix of the bilinear model can be written as:

$$A^{\mathsf{bi}}_{i,j} = \mathbb{I}[j = \operatorname*{argmax}_{h \le i} \mathsf{s}^{\mathsf{bi}}(m_i, m_h)] \tag{5}$$

$$\mathsf{s}^{\mathsf{bi}}(m_i, m_j) = \mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}(m_i, m_j) + \mathsf{s}_{\mathsf{m}}(m_i) + \mathsf{s}_{\mathsf{m}}(m_j) \tag{6}$$

The adjacency matrix of the higher-order model can be written as:

$$A^{\mathsf{hoi}}_{i,j} = \mathbb{I}[j = \operatorname*{argmax}_{h \le i} \mathsf{s}^{\mathsf{hoi}}(m_i, m_h)] \times \mathbb{I}[j \in \operatorname*{argtopk}_{h \le i} \mathsf{s}^{\mathsf{bi}}(m_i, m_h)] \tag{7}$$

---

[1] In our work as well as most work on within document coreference (Lee et al., 2017, 2018; Joshi et al., 2019) we only consider contiguous mention spans rather than allowing spans to be arbitrary sets of tokens (non-contiguous).

$$\mathsf{s}^{\text{hoi}}(m_i, m_j) = \mathsf{s}_{\mathsf{a}}^{\text{hoi}}(m_i, m_j) + \mathsf{s}_{\mathsf{a}}^{\text{bi}}(m_i, m_j) + \mathsf{s}_{\mathsf{m}}(m_i) + \mathsf{s}_{\mathsf{m}}(m_j) \quad (8)$$

where k for $\operatorname{argtopk}$ is a hyperparameter.

End-to-end training The mention proposal and antecedent prediction models are trained by relaxing the adjacency matrix A, replacing the argmax operation with a softmax (i.e., setting a weighted edge between i and j with weight $\mathsf{s}(m_i, m_j)$). The training objective is to maximize the log-likelihood of a ground truth adjacency matrix $A^\star$, where $A^\star_{i,j} = 1$ if m_i and m_j are coreferent and $i < j$ under the relaxed adjacency matrix. The $\operatorname{argtopk}$ operation is not relaxed. A *nil* antecedent is introduced, which provides similarity (s_{a}) of 0 to any mention span is incorporated in the training objective. The number of candidate antecedents is also restricted by a hyperparameter (Lee et al., 2018).

3 Efficient Approximations for End-to-End Coreference

We describe our proposed approach for efficiently approximating the *span-based* end-to-end coreference model with a *token-level* model. Our model jointly predicts which tokens are in the same mention spans (i.e., mention proposal) and what tokens are coreferent with one another (i.e., antecedent prediction). By operating at the token level, we remove the dependence on considering quadratically many phrases. We show the structure of our approximation allows for a *sparsification* technique that reduces the number of antecedent predictions that need to be considered using k-nearest neighbor relationships between tokens and by splitting documents into windows with certain computations made independently for each window. We describe how low-rank matrix approximations can be used to improve inference efficiency.

3.1 Mention Proposer

Observe that computing the set $\mathcal{M}$ requires us to evaluate $\mathsf{s}_{\mathsf{m}}(\cdot)$ for all candidate spans $\mathcal{S}$ in the document (which grows roughly quadratically with the number of tokens). Recall that $\mathsf{s}_{\mathsf{m}}(\cdot)$ is a function of the start and end tokens of each span, producing a score that is high if the pair of tokens likely form a span. This approach can be thought of as having each token t in the document predict whether or not another token u is the last token in a span beginning with t.

We first model for each token t whether it is a start (st) or end (en) token of some phrase using a linear model:

$$M_t^{\mathsf{st}} = w_{\mathsf{st}}^T \boldsymbol{x}_t \qquad M_t^{\mathsf{en}} = w_{\mathsf{en}}^T \boldsymbol{x}_t \quad (9)$$

These terms weighs each token by how likely it is to be part of *some* mention span.

Following Kirstain et al. (2021), we find that there can an empirical benefit(described in Section 6) to additionally modelling the relationship between u and t, i.e., whether it is reasonable for the span beginning with t to end in u. To do this we use a asymmetric (bilinear) scoring function. Further, we restrict the spans to be contiguous and follow the rule-based span criteria of previous work (Lee et al., 2017).

$$M_{t,u} = \begin{cases} \boldsymbol{x}_t^T \, W_{\mathsf{M}} \boldsymbol{x}_u & t \text{ to } u \text{ is a valid span} \\ -\infty & \text{otherwise} \end{cases} \quad (10)$$

For each token, we predict candidate end-tokens for a mention span starting at the given token. We assign a score per span by sum of Eq. 9 & 10 and follow previous work to select the top $0.4T$ scoring spans (mentions). We can replace the mention scoring mechanism used in previous works $\mathsf{s}_{\mathsf{m}}(\cdot)$ with an approximation based on the token level score:

$$\widehat{\mathsf{s}_{\mathsf{m}}}(m_i) = M^{\mathsf{st}}_{\text{START}(m_i)} + M^{\mathsf{en}}_{\text{END}(m_i)} + M_{\text{START}(m_i),\text{END}(m_i)} \quad (11)$$

Rather than having to instantiate a d-dimensional span representation for all $|\mathcal{S}|$ spans, our approach simply uses the output token representations from the encoder. This requires $O(T)$ space compared to $O(|\mathcal{S}|)$ space. Note that computing $\widehat{\mathsf{s}_{\mathsf{m}}}(m_i)$ for all mentions requires at most two matrix multiplications, each with just T rows. Observe that this leads to a drastic reduction in time and space complexity. As noted by previous work (Toshniwal et al., 2020b), the mention proposal step requires the most memory usage because of the quadratic dependency. We validate the reduction in time and memory requirements of our token level mention detection in Section 4.5 & 4.6.

Pretraining For Mention Detection Previous work (Wu et al., 2020; Toshniwal et al., 2020b) has shown that pre-training models for mention detection is beneficial, especially in cases where predicting singleton mentions is required (e.g., LitBank (Bamman et al., 2019)). Given a set of ground truth mentions $\mathcal{M}^\star$ and the set of mentions from a given document $\mathcal{M}$, we use a mention detection loss which minimizes:

$$\begin{aligned} -\sum_{m_i \in \mathcal{M}} & \log \sigma(\widehat{\mathsf{s}_\mathsf{m}}(m_i))\mathbb{I}[m_i \in \mathcal{M}^\star] \\ & + \log \sigma(1 - \widehat{\mathsf{s}_\mathsf{m}}(m_i))\mathbb{I}[m_i \notin \mathcal{M}^\star] \end{aligned} \tag{12}$$

We use it as a multi-task objective in training the models and as well as a pre-training objective. We detect singleton mentions by using a threshold on the mention score value $\widehat{\mathsf{s}_\mathsf{m}}(m_i)$ which is tuned on the development set according to the downstream performance.

3.2 Antecedent Scoring

Next, we would like to model coreference relationships between tokens to approximate the span-level scoring function (s_a, Eq. 3, 5). We predict for each token, the other tokens with which it is coreferent. These predictions are then aggregated to make span level predictions.

First, we consider approximating bilinear scoring function at the token level ($\mathsf{s}_\mathsf{a}^\mathsf{bi}$). We use a bilinear model applied to the encoded token representations. We parameterize four asymmetric similarity functions. Note that the backwards-chaining property of inference motivates our use of the asymmetric function. We model the similarity between tokens that are the start or end tokens of phrases separately:

$$S_{i,j}^{\mathsf{ss}} = \boldsymbol{x}_{\text{START}(i)} W_{\mathsf{st}}^T \boldsymbol{x}_{\text{START}(j)} \tag{13}$$

$$S_{i,j}^{\mathsf{es}} = \boldsymbol{x}_{\text{END}(i)} W_{\mathsf{st}}^T \boldsymbol{x}_{\text{START}(j)} \tag{14}$$

$$S_{i,j}^{\mathsf{se}} = \boldsymbol{x}_{\text{START}(i)} W_{\mathsf{en}}^T \boldsymbol{x}_{\text{END}(j)} \tag{15}$$

$$S_{i,j}^{\mathsf{ee}} = \boldsymbol{x}_{\text{END}(i)} W_{\mathsf{en}}^T \boldsymbol{x}_{\text{END}(j)} \tag{16}$$

We use these similarities to approximate the bilinear antecedent scoring function (Eq. 3) as:

$$\widehat{\mathsf{s}_\mathsf{a}^\mathsf{bi}}(m_i, m_j) = S_{i,j}^{\mathsf{ss}} + S_{i,j}^{\mathsf{es}} + S_{i,j}^{\mathsf{se}} + S_{i,j}^{\mathsf{ee}} \tag{17}$$

Observe how $\widehat{\mathsf{s}_\mathsf{a}^\mathsf{bi}}(\cdot,\cdot)$ reduces the memory requirements compared to $\mathsf{s}_\mathsf{a}^\mathsf{bi}(\cdot,\cdot)$. We do not need to instantiate the span representations, only use the encoded token representations. Computing $\widehat{\mathsf{s}^\mathsf{bi}}(m_i, m_j)$ requires at most $O(T^2)$ instead of $O(|\mathcal{S}|^2)$ work. We can compute each of the $S_{i,j}^{\cdot}$ (Eq. 16) in space $O(T^2)$ and as matrix multiplication between matrices of $O(T)$ rows.

For training and inference in our model, we define the adjacency matrix A with $\hat{A}$:

$$\widehat{A^{\mathsf{bi}}}_{i,j} = \mathbb{I}[j = \operatorname*{argmax}_{h \le i} \widehat{\mathsf{s}^\mathsf{bi}}(m_i, m_h)] \tag{18}$$

$$\begin{aligned} \widehat{\mathsf{s}^\mathsf{bi}}(m_i, m_j) = & \widehat{\mathsf{s}_\mathsf{a}^\mathsf{bi}}(m_i, m_j) + \widehat{\mathsf{s}_\mathsf{m}}(m_i) \\ & + \widehat{\mathsf{s}_\mathsf{m}}(m_j) \end{aligned} \tag{19}$$

Inference can then be done as exactly as before, using connected-components based inference.

3.3 Token Windows & Sparsifying Antecedent Scoring with k-NN Graphs

We can use the backwards-chaining structure of the inference procedure and divide a document into smaller token windows (non-overlapping), reducing the number of tokens that need to be encoded in any one component. We can propose mentions independently in each window. We then perform antecedent scoring using the K-NN sparsification described below for each window. By batching the long document into these windows, we never need to store more than the final encoded token representations for the tokens appearing in some entity cluster.

The approximation method presented thus far reduces the complexity of end-to-end coreference approaches from depending on the number of spans to the number of tokens. However, for long documents scaling quadratically in the number of tokens is still prohibitively expensive, both in terms of time complexity and also in terms of memory. Observe that computing and storing each of the $\widehat{\mathsf{s}^\mathsf{bi}}(\cdot,\cdot)$ may become prohibitively expensive for all pairs of tokens in the document. We would like to reduce the time and space complexity of this approach.

We propose to approximate the top scoring pairs of mention spans according to $\widehat{\mathsf{s}^\mathsf{bi}}(\cdot,\cdot)$ (i.e., further approximating $\mathsf{s}^\mathsf{bi}(\cdot,\cdot)$). We do this by only allowing two mentions m_i and m_j to be coreferent if the start/end tokens of m_j are in the k-nearest neighbors of the start/end tokens of m_i. More precisely, we will maintain the k nearest neighbors of each token for each of the four similarity functions S^{ss}, S^{se}, S^{es}, S^{ee} (Eq. 16). To align with inference procedure, we select these k nearest neighbors for each token only from the preceding tokens in the document. we define $S^{\mathsf{ss}}_{\mathrm{knn}i,j}$ to be non-zero only if j is in the top-k values of $S^{\mathsf{ss}}_{i,\cdot}$

$$S^{\mathsf{ss}}_{\mathrm{knn}i,j} = \begin{cases} S^{\mathsf{ss}}_{i,j} & \text{if } j \in \mathrm{argtopk}_h \, S^{\mathsf{ss}}_{i,h} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

We define $S^{\mathsf{se}}_{\mathrm{knn}i,j}, S^{\mathsf{es}}_{\mathrm{knn}i,j}, S^{\mathsf{ee}}_{\mathrm{knn}i,j}$ analogously.We then build an further approximation of $\widehat{\mathsf{s}^{\mathsf{bi}}}$ using these $S^{\cdot}_{\mathrm{knn}}$ values:

$$\widehat{\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}}_{\mathrm{knn}}(m_i, m_j) = S^{\mathsf{ss}}_{\mathrm{knn}i,j} + S^{\mathsf{es}}_{\mathrm{knn}i,j} + S^{\mathsf{se}}_{\mathrm{knn}i,j} + S^{\mathsf{ee}}_{\mathrm{knn}i,j} \tag{21}$$

Observe that storing $S^{\mathsf{ss}}_{\mathrm{knn}i,j}$ can use sparse matrices and therefore provide better scalability to long documents for which storing $O(4Tk)$ is advantageous over $O(T^2)$.

End-to-end Training We use the same end-to-end training procedure that was used by previous work (Section 2) using our approximated mention proposal and antecedent scoring procedures. We note that the use of token windows and KNN sparsification of the antecedent scoring term do not change training at all, this is only applied at inference time.

3.4 Low-dimensional Approximations

Much of the computation time of the k-NN graph approximation model comes from the computation of the top-k nearest tokens. The computation bottleneck mostly depends on the high dimensionality of the encoded token representations, which are from transformer-based language models (Joshi et al., 2020).

To produce lower dimensional embeddings of each token, which preserve similarities in the original space, we use low-rank matrix approximation methods, specifically the Nyström method (Williams and Seeger, 2001; Musco and Musco, 2017, inter alia). We hope to approximate the matrices $S^{\mathsf{ss}}_{i,j}, S^{\mathsf{se}}_{i,j}, S^{\mathsf{es}}_{i,j}, S^{\mathsf{ee}}_{i,j}$. While these are asymmetric, we can consider an equivalent symmetrized version where each token appears two times (on left/right of bilinear term) to apply Nyström. The lower dimensional embedding produced by Nyström is done

The Nyström method provides a low-rank approximation of a symmetric pairwise similarity matrix $S \in RR^{N\times N}$, by selecting ℓ *landmark* points uniformly at random among the N rows of S. We use L_i to be column vector one-hot representation of the i^{th} landmark. We assume $L \in \mathbb{R}^{N\times\ell}$ to be a matrix of such one-hot representations. The approximation of S is given by: $\hat{S} = LS(L^TSL)^{-1}L^TS$. The term LS is a ℓ dimensional embedding of the rows, which is defined by the similarity of each row with each of the ℓ landmarks (ℓ is the reduced dimension). Similarly, $(L^TSL)^{-1}L^TS$ can be thought of as providing a ℓ dimensional embedding of each column of S, which is based on the similarity and the (inverse) of the landmark similarities.

3.5 Limiting Num. of Candidate Antecedents

In the aforementioned approach, the number of candidate antecedents scales with the document length. We would like to determine a mechanism for using a fixed number of candidate antecedents if desired. Previous work other work uses entity-level representations to achieve this (Toshniwal et al., 2020b; Xia et al., 2020).

In our work, we operate at the mention level, removing mentions as candidate antecedents. We define a hyperparameter, ρ, which is maximum number of antecedents that would be kept after processing each window of the document. Our approach removes mentions as candidate antecedents which (1) belong to large coreference clusters (2) are not frequently selected as antecedents. We achieve this by dropping mentions in the order of $|\mathcal{C}_m|$ - $\sum_i A_{i,m}$, where $|\mathcal{C}_m|$ is the size of the cluster of the mention m and $\sum_i A_{i,m}$ is the degree of the mention in the antecedent graph.

4 Experiments

In this section, we compare our proposed approach for scalable coreference on long documents to various state-of-the-art methods in terms of accuracy as well as efficiency of training and inference. We perform a detailed scalability analysis, which characterizes the time/memory used by each method as a function of the length of documents. We also report timing results on novels of ~ 2 million tokens.

4.1 Datasets

We evaluate each method on the following datasets: **CoNLL-2012 Shared Task:** The CoNLL-2012 shared task (Pradhan et al., 2012) uses the v5.0 of the OntoNotes corpus for the task of coreference resolution in English, Chinese, and Arabic languages. We use only the English version for our experiments. The training set contains 2802 training, 343 development, and 348 test documents. The training documents contain on average of 454 tokens and a maximum of 4009 tokens. **Litbank:** We also use the Litbank dataset (Bamman et al., 2019) which consists of 210,532 tokens evenly drawn from 100 different English language literary texts. The average document length in Litbank is much longer (around 2,000 tokens). Following (Bamman et al., 2019; Toshniwal et al., 2020b), we use a 10-fold cross-validation setup with 80% of the data as training data and rest 10% each as validation and test data. The final evaluation is reported as the average of all 10 test runs. Note that the family of end-to-end approaches that we are approximating with our method do not predict singletons as is typically done for Litbank. Mention pretraining is performed as described in Section 3.1 **MedMentions** We also repurpose MedMentions (Mohan and Li, 2019) an existing entity linking dataset in the biomedical domain for coreference resolution. We treat the entity labels as the ground truth cluster assignments of each mention for coreference training/analysis. We use the ST21PV subset that is recommended by (Mohan and Li, 2019). **Artamène ou le Grand Cyrus (Artamène, or Cyrus the Great)**. To further asses the scalability of our approach, we run our method on an English translation of the 17th century French novel that is one of the longest books available in English the public domain (Scudery, 1601). The work contains *1.99 million* tokens and over two million sub-tokens. We use this data to illustrate the scalability of our approach to really long documents.

4.2 Methods

We compare our end-to-end coreference approximation with and without the token windows, KNN sparsification approach (i.e. Ours and Ours (Sp.), Section 3.3). We denote the number of neighbors used in the sparsification approach as k and the size of the window used as w. We compare these to the methods that they are approximating: the bilinear scoring function-based method (E2E (bi)) (Lee et al., 2017) as described in Eq. 3 and the coarse-to-fine higher-order inference based approach (E2E (hoi)) (Lee et al., 2018). All models use spanbert-large (Joshi et al., 2020) to encode tokens. The encoder parameters are trained along with the coreference specific model parameters (see Section 4.4 for details). E2E (bi), E2E (hoi) use additional features such as speaker and genre, we do not use this metadata in our proposed approximation approach.

4.3 Coreference Performance

In Table 1, we report the coreference performance (along with the running time and memory usage) for each method on the three datasets. We observe that our approximate approach is achieves comparable performance to the E2E approaches on CoNLL-2012 and MedMentions, performing slightly worse on Litbank. We hypothesize that the token level representations can be effective at these tasks due to the expressiveness of the contextualized embeddings. We observe that the performance of our model is relatively unchanged with and without the sparsification approach applied.

Recently, Toshniwal et al. (2020b); Xia et al. (2020) have proposed memory-based models optimising memory usage. Toshniwal et al. (2020b) trains for improve mention detection by another pretraining process. These papers achieve state-of-the-art results on Litbank and are focused on reducing the running time and memory usage of coreference models by storing *entity* representations instead of *mention* representations in a bounded memory architecture. We compare inference running time and coreference performance of our method with them in Table 1. We find that our models run 10x faster and are slightly more memory efficient than UMem(Toshniwal et al., 2020b) while matching their performance on litbank.

4.4 Experimental Details

We use the hyperparameter settings from (Xu and Choi, 2020) in all applicable cases. We use 512 as the segment length. On CoNLL and Medmentions we train all models for 24 epochs with maximum training sentences set to 3. On Litbank, we train for 120 epochs and pick parameters from (Toshniwal et al., 2020b). We use 0.4 as the ratio to pick the top spans(mentions) among all candidate spans.

4.5 Inference Time and Memory Usage

In Figure 1, we compare the time and memory used by the end-to-end coreference models and our

		Inference		Overall	MUC			CEAF			B^3		
		Mem.	Time	F1	R	P	F1	R	P	F1	R	P	F1
CoNLL	E2E (hoi)	7.77	30	**79.6**	84.8	86.1	85.4	77.3	79.3	78.3	74.7	76	75.4
	E2E (bi)	4.31	28.2	78.4	85.03	84.1	84.56	76.7	76.56	76.63	75.98	72.44	74.17
	Ours	1.78	**24.79**	78.03	84.22	84.28	84.25	74.43	72.63	73.52	75.93	76.68	76.3
	Ours (Sp.)	**1.5**	26.81	77.59	83.34	84.43	83.88	74.39	72.13	73.24	74.36	76.99	75.65
Litbank	E2E (hoi)	2.78	8.12	**78.44**	91.92	87.24	89.50	66.47	68.90	67.59	80.33	76.28	78.21
	E2E (bi)	2.68	5.2	77.77	91.72	86.73	89.15	64.82	68.72	66.67	78.74	76.33	77.48
	UMem	3.00	23.46	76.5	85.7	90.8	88.2	66.0	65.1	65.5	72.1	80.0	75.9
	Ours	**2.12**	2.1	75.93	89.53	86.28	87.86	65.09	65.38	65.18	73.65	76.00	74.75
	Ours (Sp.)	2.39	**1.57**	74.71	87.90	86.14	86.99	65.19	63.67	64.36	70.82	74.95	72.77
MedMent.	E2E (hoi)	3.9	57.08	60.86	63.26	65.75	64.48	57.6	61.29	59.38	60.1	57.37	58.71
	E2E (bi)	2.64	54.48	60.67	64.69	64.5	64.6	59.21	59.53	59.37	61.05	55.32	58.04
	Ours	1.69	**43.9**	**61.76**	67.92	63.02	65.38	59.78	59.55	59.67	63.60	57.17	60.21
	Ours (Sp.)	**1.57**	45.3	61.49	66.87	63.48	65.13	59.65	59.34	59.49	62.12	57.69	59.82

Table 1: **Accuracy & Efficiency** on the three benchmarks. Memory in GB, time in seconds.

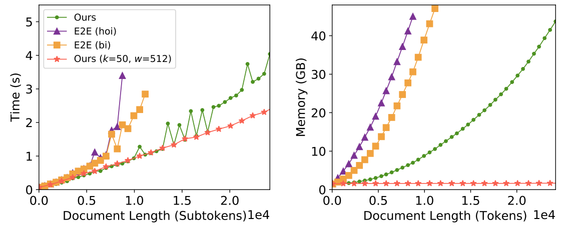

Figure 1: **Time and GPU Memory Comparison of different models on the book "Little Women".**

proposed family of approximations. We select a book at random from the Litbank corpus (*Little Women*) and report the time and memory used by each method to perform coreference as a function of the number tokens analyzed. We plot a curve for each, reporting the statistics until the method runs out of GPU memory (48GB). We cut off the x-axis of the graph where our proposed approach without the backwards chaining runs out of memory. Our token level models only scales upto 24K tokens. We note that Ours (Sp.) is able to run on the entire book requiring only marginally higher memory for higher document lengths. This is in contrast with previous E2E methods which run out of memory for documents longer than 1e4 tokens.

4.6 Training Time and Memory Usage

We report in Table 2 the training time and memory requirements for each of the methods. For each dataset, we train all the methods in focus for the same number of epochs/updates. We train for 24 epochs on CoNLL, 120 epochs on Litbank and 24 epochs on MedMentions. We observe that our approach greatly reduces GPU memory requirements and are also slightly faster. This gap is wider for datasets containing longer documents as shown by the numbers on LitBank. Note that the sparsification approximation is simply an inference time approximation and uses the same trained model as our approach with the K-NN approximation.

		Training	
		Time	Mem.
CoNLL	Ours	**6.8**	**13.69**
	E2E (hoi)	11.75	19.39
	E2E (bi)	7.5	15.7
Litbank	Ours	**2.9**	**19.43**
	E2E (bi)	3.2	22.7
	E2E (hoi)	5	27.0
MedM.	Ours	**5.5**	**13.8**
	E2E (hoi)	8.75	18.5
	E2E (bi)	6.25	15.6

Table 2: **Training time(hours) and memory(GB) usage**. Our approach requires less time and memory than the competing end-to-end approaches.

4.7 Scaling to Long Documents

We run Ours (Sp.) on the full text of *Artamène or Cyrus the Great*, which has 1.99M tokens ($>$ 2M subtokens). To our knowledge, this is the largest single document a neural within document coreference system has been applied to. In Figure 2, we show that our approach runs in about 14 minutes. Further, we demonstrate how the hyperprameters of the sparsification can be adjusted depending on the system requirements. We show that the window size parameter can be set to be the minimal amount (w=512) to require just 13 GB of GPU RAM. Table 3 suggests using small window sizes are also advantageous in terms of accuracy.

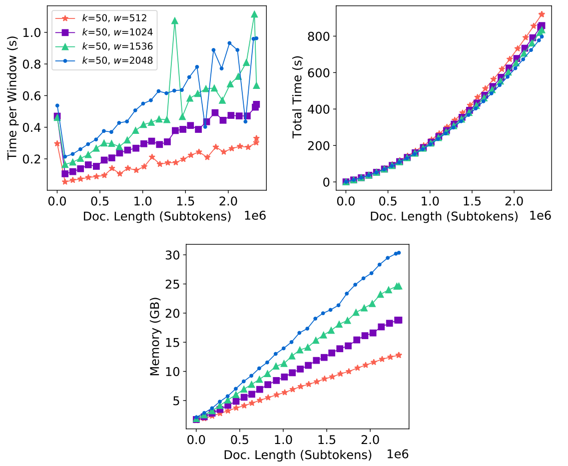

Figure 2: **Scalability to 2 million tokens** Time and memory usage of our K-NN based methods on Artaméne a book with millions of tokens.

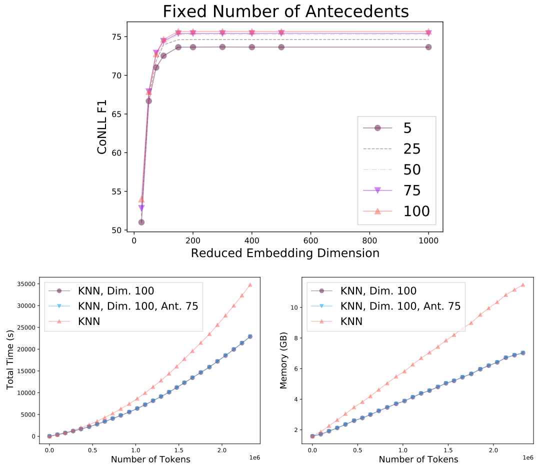

Figure 3: **Reduced Dimensionality & Candidate Antecedents**. We report CoNLL F1 on CoNLL and Time and Memory Usage on *Artamène or Cyrus the Great*.

5 Model Analysis

K-NN Sparsification Performance Analysis

In Table 3, we show the CoNLL F1 as a function of the number of neighbors k and window size w in Ours (Sp.). We observe that we can achieve high quality results even with a small number of neighbors, providing an empirical justification for our approximation. In our case, using just 10 nearest neighbors ($k = 10$) puts Ours (Sp.) within 99% of the performance of the version of our approach without sparsification. Litbank however required to use a higher value of K due to the presence of long distance coreference links in literary texts.

Dimensionality Reduction & Limiting Number of Antecedents

Window Size	512	1024	1536
k=10	77.16	77.33	77.25
k=50	77.59	77.55	77.56
k=80	77.68	77.62	77.61
k=∞	78.03	78.03	78.03

Table 3: **Sensitivity to Approximation in K-NN sparsified approach on CoNLL (F1)**.

Type	E2E (hoi)	E2E (bi)	Ours	Ours (Sp.)
Pronoun	33.74	34.57	33.62	33.50
Noun	40.59	39.98	40.36	40.00

Table 4: **Performance on mention types (F1)**.

Figure 3, shows the performance of the reduced dimensionality method 3.4 and limiting antecedents 3.5. A reduced dimension of 200 can match the performance of encoder embeddings dimension while significantly improving running time. Further, the model performs well while just keeping 75 mentions from each window of size 512(comprises squared number of possible mention spans) in the memory. We note that a reduction in memory can be achieved by dropping antecedents and using sparse matrices. However, this is not as efficient as using dense matrices on the gpu.

6 Performance Analysis

Comparisons with Baselines

To give a sense of how the proposed approximations work, we performed a simple analysis among E2E (hoi), Ours and Ours (Sp.) on the last fold of Litbank dataset. In the first experiment, we keep only the pronomial mentions in the predicted clusters and evaluate coref scores. In the second experiment we keep all mentions containing atleast one noun. Table 4 shows the final numbers. The gap in performance between E2E (hoi) and Ours seems to be equal in both categories. Ours (Sp.) and Ours have similar performance gap in both categories as well. Thus our models seem to be approximating fairly across different categories.

Table 5 shows the analysis of distance between antecedents predicted by each model on the last litbank fold. Our models seem to have a higher average distance between antecedents. This shows that the models proposed are capable to identifying long distance links. Note that distance between antecedent does not determine accuracy. A mention linked to any mention in its golden coreference cluster will have the same effect.

Type	E2E (hoi)	E2E (bi)	Ours	Ours (Sp.)
Mean	149	189.7	301.19	447.8
Max	2117	2104	2303	2303
Std.	296.8	342.9	403.01	588

Table 5: **Analysis of distance between mention and antecedent for all models in subtoken units**.

		P	R	F
CNL	Ours	78.19	77.86	78.03
	Ours - SS	78.14	77.54	77.84
	Ours - BM	78.84	77.26	78.04
Litbank	Ours	76.09	75.89	75.93
	Ours - SS	75.21	76.22	75.58
	Ours - BM	70.81	74.40	72.18
	Ours - SS - MT	68.35	68.22	68.24
MM	Ours	63.77	59.92	61.76
	Ours - SS	63.4	59.2	61.13
	Ours - BM	60.84	62.35	61.58

Table 6: **Effect of the removal of different components in Ours**. CNL - CoNLL, MM - Medmentions

Effects of model components

We further analyse the effect of other heuristics that went into the model. We use a Subtoken Strategy (SS) where we restrict the candidate mentions to align with subtoken starts, ends. As shown in Table 6, (SS) seems to have improved the results on all the datasets. Also, for litbank, mention pretraining & mention training (MT) seem to have helped significantly. Mention training forces the score of golden mentions to be higher thereby making it easy to use a threshold for singletons at inference. Bilinear mention (BM) term described in Eq.10 seemed to have helped in litbank and medmentions.

7 Related Work

With the growing computational cost of deep learning, NLP researchers have started to focus on more efficient models (Strubell et al., 2019; Schwartz et al., 2020). As coreference is a document-level phenomena, it is particularly challenging to scale, especially for long documents. While most of the work in coreference has focused on genres of text with short documents such as news articles and blogs (Pradhan et al., 2012), there has been renewed focus in long text documents such as novels (Bamman et al., 2019; Toshniwal et al., 2020b). Coreference in long text is particularly interesting due to the introduction of long-range anaphora.

Span based end-to-end coreference systems (Lee et al., 2017, 2018; Joshi et al., 2020; Wu et al., 2020) have been the state-of-the-art in short-document coreference resolution (Lu and Ng, 2020). These systems avoid training a separate mention detector. However, end-to-end coreference models are challenging to scale to long text documents due to their large memory footprint as well as slow training and inference. Thus, research on long-document coreference so far has focused on incremental (memory-based) coreference resolution (Xia et al., 2020; Toshniwal et al., 2020a,b). Memory-based approaches model coreference as online clustering by picking the most similar entity to every new mention where the cluster representations (i.e., entity representations) are also updated. However, the underlying recurrent nature of these models and the frequent read-write memory operations make these models slow. In this work, we focus on the end-to-end coreference system and show gains both in speed as well as memory.

We note that this paper's novel token-level model ideas presented in this paper were concurrently introduced by Kirstain et al. (2021). We also introduce token windows, k-nearest neighbor based sparsification techniques. Furthermore, we provide empirical results on documents with about two million tokens, which we believe to be one of the longest documents to which neural coreference models have been applied. We also note that Wu et al. (2020) hold state-of-the-art results on CoNLL (83F1). Wu et al. (2020) uses a question-answering cross-encoder style model to perform coreference. However, the method is very computationally expensive and so it is difficult to scale to the long documents which is the focus of this paper.

8 Conclusion

In this paper, we introduce a new scalable approach for performing coreference that scales to long documents. Our approach replaces costly span-based operations with token-level decisions for proposing mentions and determining antecedents. Our approach uses token similarity in the form of k-nearest neighbor graphs along with processing documents in span windows to reduce the time and memory complexity. We evaluate our proposed approach empirically and demonstrate that it achieves competitive coreference F1 scores while improving time and memory usage requirements. We demonstrate the scalability of our method by applying it to novels with about two million tokens. We further propose and demonstrate the use of low rank approximations and dropping of non essential tokens to improve memory/time efficiency.

Broader Impact and Discussion of Ethics

While our model is not tuned for any specific real-world application, our method could be used in sensitive contexts such as legal or health-care settings, and it is essential that any work using our method undertake extensive quality-assurance and robustness testing before using it in their setting. The datasets used in our work do not contain any sensitive information to the best of our knowledge.

Replicability: As part of our contributions, we will release the code used for training and evaluation in this work, as well as all the trained models at https://github.com/raghavlite/Scalable-Coreference.

References

David Bamman, Olivia Lewke, and Anya Mansoor. 2019. An annotated dataset of coreference in english literature. *arXiv preprint arXiv:1912.01140*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel S Weld. 2019. Bert for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5807–5812.

Yuval Kirstain, Ori Ram, and Omer Levy. 2021. Coreference resolution without span representations. *arXiv preprint arXiv:2101.00434*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Jing Lu and Vincent Ng. 2020. Conundrums in entity coreference resolution: Making sense of the state of the art. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6620–6631, Online. Association for Computational Linguistics.

Sunil Mohan and Donghui Li. 2019. Medmentions: a large biomedical corpus annotated with umls concepts. *arXiv preprint arXiv:1902.09476*.

Cameron Musco and Christopher Musco. 2017. Recursive sampling for the Nyström method. In *2017*, pages 3833–3845.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63.

Madeleine de Scudery. 1601. Artamenes. `https://quod.lib.umich.edu/e/eebo/A70988.0001.001`.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Shubham Toshniwal, Allyson Ettinger, Kevin Gimpel, and Karen Livescu. 2020a. Petra: A sparsely supervised memory model for people tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5415–5428. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020b. Learning to ignore: Long document coreference with bounded memory neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8519–8526.

Christopher Williams and Matthias Seeger. 2001. Using the Nyström method to speed up kernel machines. In *2001*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. Corefqa: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963.

Patrick Xia, João Sedoc, and Benjamin Van Durme. 2020. Incremental neural coreference resolution in constant memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8617–8624, Online. Association for Computational Linguistics.

Liyan Xu and Jinho D. Choi. 2020. Revealing the myth of higher-order inference in coreference resolution. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8527–8533, Online. Association for Computational Linguistics.

LEWIS: Levenshtein Editing for Unsupervised Text Style Transfer

Machel Reid[*]
The University of Tokyo
`machelreid@weblab.t.u-tokyo.ac.jp`

Victor Zhong
University of Washington
`vzhong@cs.washington.edu`

Abstract

Many types of text style transfer can be achieved with only small, precise edits (e.g. sentiment transfer from `I had a terrible time...` to `I had a great time...`). We propose a coarse-to-fine editor for style transfer that transforms text using Levenshtein edit operations (e.g. insert, replace, delete). Unlike prior single-span edit methods, our method concurrently edits multiple spans in the source text. To train without parallel style text pairs (e.g. pairs of +/- sentiment statements), we propose an unsupervised data synthesis procedure. We first convert text to style-agnostic templates using style classifier attention (e.g. `I had a SLOT time...`), then fill in slots in these templates using fine-tuned pretrained language models. Our method outperforms existing generation and editing style transfer methods on sentiment (YELP, AMAZON) and politeness (POLITE) transfer. In particular, multi-span editing achieves higher performance and more diverse output than single-span editing. Moreover, compared to previous methods on unsupervised data synthesis, our method results in higher quality parallel style pairs and improves model performance.[1]

1 Introduction

In text style transfer, a model changes the style of a source text (e.g. sentiment, politeness) into a target style, while otherwise changing as little as possible about the input. Many types of style transfer can be performed with only small, precise edits instead of generation from scratch. Consider the task of transforming a negative sentiment sentence such as `the worst ribs I've ever had!` to a positive sentence such as `probably the best ribs ever!`. Here, we need only invert the negative sentiment phrase around `worst` — the references to `ribs` should be left as-is. Recent and concurrent work on text style transfer propose single-span editing (for insertion and replacement) (Wu et al., 2019; Malmi et al., 2020) as an alternative to generating the target text from scratch (Prabhumoye et al., 2018; He et al., 2020b; John et al., 2019; Shen et al., 2017; Fu et al., 2018).

We introduce a more flexible and powerful multi-span editing method that identifies multiple style-specific components of the text and concurrently edits them into the target style. Given a source text, we first predict the sequence of coarse-grain Levenshtein edit types (e.g. insert, replace, delete) that transform the source text to the target text, then fill insertion and replacement edits using a generator. In the previous example, the operations correspond to inserting the word `probably` before `the`, replacing `worst` with `best`, and removing the words `I've` and `had`. This example is illustrated in detail in Figure 1.

Learning to edit requires supervised source-target text pairs. How do we learn high-quality editors when no such supervised parallel data exists? Given a style text, we synthesize its pair by identifying style-specific content and replacing it with samples from style-specific masked language-models. In our sentiment transfer example, the style-specific content of the sentence `I had a great time at the theatre` is `had a great time`. We can replace this phrase by `saw a fantastic movie today` to synthesize an alternative positive-sentiment sentence, or by `got ripped off today` to synthesize a negative-sentiment sentence. Figure 3 illustrates this example in detail.

We evaluate our editing and synthesis framework, which we call LEWIS (Levenshtein editing with unsupervised synthesis), on three style transfer tasks in sentiment (YELP, AMAZON) and polite-

---

[*] Corresponding author

[1] Code and data can be found at https://github.com/machelreid/lewis

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3932–3944
August 1–6, 2021.

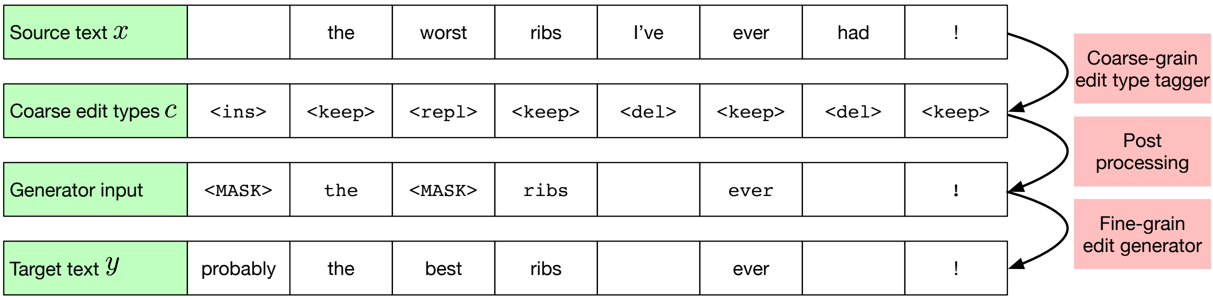

Figure 1: Coarse-to-fine Levenshtein editor. Given the source text, the two-step editor first generates coarse edit types via a tagger. A subsequent generator fills in insertions and replacements while taking into account the source text and the edit types.

ness (POLITE) transfer, and achieve state-of-the-art results in terms of retention of style-agnostic content, similarity to the annotated target text, and transfer accuracy. LEWIS significantly outperforms prior state-of-the-art methods by 2.6-13.5% accuracy depending on the task. In further analyses, we show that (1) compared to concurrent work on editing for style transfer, our editor achieves 33.3% higher accuracy when trained on the same data; (2) compared to a competitive BART (Lewis et al., 2020) pure generation baseline, our editor achieves 5.8% higher accuracy when trained on the same data; (3) compared to concurrent work on unsupervised synthesis of style transfer data, our synthesis procedure improves performance by 9.5 BLEU when used to train the same model. Our experiments show that our editor significantly outperforms both pure generation and editing prior methods, that our editor yields more diverse text transfer and that training on our synthesized data improves performance more than prior synthesis methods.

2 LEWIS

LEWIS consists of coarse-to-fine editing and data synthesis. The editing component, shown in Figure 1, performs local, precise edits of style-specific content of the source text to produce the target text. The data synthesis component, shown in Figure 3, produces supervised source-target text pairs, which do not exist in naturally, to train the editor. To apply our method to transfer text from a source style to a target style, we first train style-specific masked language models, with which we synthesize source-target text pairs. We then compute Levenshtein operations for these source-target text pairs and train the coarse-to-fine editor to reproduce these operations. The full LEWIS is shown in Figure 2. For ease of exposition, we first describe the editor, then describe how to synthesize parallel data to train the editor.

2.1 Style transfer via Levenshtein editing

We propose a coarse-to-fine editor that first predicts coarse-grain Levenshtein edit types (Levenshtein, 1966), then fills in fine-grain edits with a generator. Figure 1 illustrates the editor.

Suppose we are to transfer text from a source style into a target style. Let x denote the source text, which we would like to edit into the target text y. In the example shown in Figure 1, we transform the source text `the worst ribs I've ever had!` into `probably the best ribs ever!` Our approach has two parts: The source text is first tagged with a sequence of coarse-grain Levenshtein transition types c that transform x into y. A generator then fills in phrases for insertion and replacement operations. The set of coarse Levenshtein transition types are `insert`, `keep`, `replace`, and `delete`. In the running example, the sequence of operations are to insert before `the`, replace `worst`, and delete `I've` and `had`.

First, we train a RoBERTa-tagger (Liu et al., 2019) to generate these coarse edit types, which produces coarse edit types for each token in the source text. To accommodate the insertion operation, we produce two tags for each token. The first tag is a binary indicator of whether an additional phrase should be inserted before this token. The second tag is the non-insertion operation to take for this token. In the previous example, for instance, the word `the` triggers both insertion and keep operations.

$$c = \mathrm{RoBERTa}_c(x; \Phi_c) \tag{1}$$

Next, we train a fine-grain edit generator to produce the target text. Unlike the coarse-grain edit type

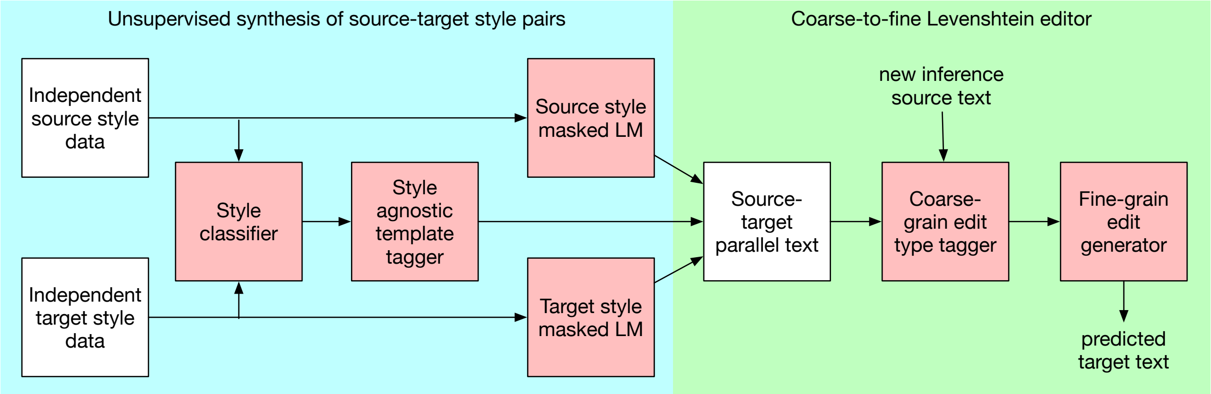

Figure 2: LEWIS consists of two components. Given source-target style text pairs, a coarse-to-fine Levenshtein editor (yellow) first identifies coarse-grain Levenshtein edit types to perform for each token in the source text (e.g. insert, replace, delete), then fills in the final edits with a fine-grain generator to produce the target text. In most applications, supervised source-target style text pairs rarely exist. To resolve this lack of annotated data, we perform unsupervised synthesis of source-target style pairs (blue) by first learning to produce style-agnostic templates given arbitrary style text. Next, we fill in slots in the template by sampling from style-specific masked language-models. In this figure, source and intermediate data are shown in white while model components are shown in red.

generator which only observes the source text, the fine-grain edit generator observes both the original source text and the source text with the coarse-grain edit types applied x_c. We use the edit types produced by the Levenshtein algorithm during training and the edit types predicted by the RoBERTa-tagger during inference.

$$y = \text{BART}_y(x, x_c; \Phi_{\text{fn}}) \tag{2}$$

Our generator is a BART-based (Lewis et al., 2020) masked sequence-to-sequence model. The input to BART is the concatenation of the original source text x and the source text with the coarse-grain edit types applied x_c. The generator is trained to fill in phrases for coarse-grain edit types `<ins>` and `<repl>`. In the example, BART_y is given the input text `the worst ribs I've ever had! SEP <MASK> the <MASK> ribs ever !` and respectively fills in the two `<MASK>`s with `probably` and `best`.

2.2 Unsupervised synthesis of source-target style pairs

Training an editor requires large quantities of source-target text pairs. While there exists an abundant amount of style-specific data, parallel source-target pairs are difficult to collect and annotate. How do we train editing style transfer methods when no such data exists? We hypothesize that pre-trained masked language-models, when carefully constrained to generate only style-specific content, can provide high-quality source-target pairs for style transfer.

Our synthesis procedure, shown in Figure 3, is two-fold. First, given a text s from either style, we identify a style-agnostic template t, in which style-specific content are replaced with slots. For instance, for the style text `I had a great time at the theatre`, the style-agnostic template is `I SLOT at the theatre`. To identify style-specific content, we train a RoBERTa-based style classifier that differentiates between text from each style. Vaswani et al. (2017) and Hoover et al. (2020) show that heavily attended-tokens correlate strongly with tokens that are indicative of the target class. We observe similar results when inspecting the attention matrices computed by the 12-layer Transfomer for the sentiment classification task. Namely, the penultimate layer's attention weights correlate strongly with words humans identify as strongly indicative of positive vs. negative sentiment. Hence, we define style-specific content as tokens that have higher-than-average attention weights in the classifier.

Consider the multi-head attention matrix A in the penultimate Transformer layer, where A_{ij} represents the attention weight of the jth attention head on the i's token, normalized across all tokens. First, we max-pool A_i over all attention heads to form a_i. Conceptually, a_i represents the maximum extent to which the ith word was attended to by any

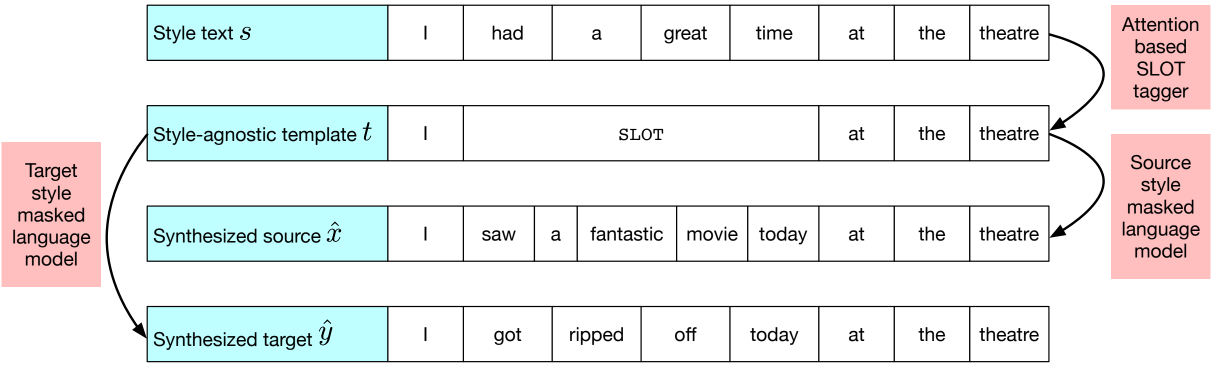

Figure 3: Unsupervised synthesis of source-target style pairs. We first train an attentive style classifier, whose attention weights we use to identify style-specific content. Next, we remove style-specific content with slots to form a style-agnostic template. This template is finally filled using style-specific masked language-models for each style to synthesize parallel style text pairs.

attention head.

$$a_i = \max_j A_{ij} \tag{3}$$

Let N denote the sequence length. We compute the average attention weight $\tilde{a}$ as

$$\tilde{a} = \frac{\sum_i^N a_i}{N} \tag{4}$$

To modify the style text s into the style-agnostic template t, we keep tokens that have above-average attention weight.

$$t_i = \begin{cases} \texttt{SLOT}, & \text{if } a_i \geq \tilde{a} \\ s_i, & \text{if } a_i < \tilde{a} \end{cases} \tag{5}$$

We merge consecutive `SLOT` tokens in t. In the running example, for the style text `I had a great time at the theatre`, the tagger generates `I SLOT SLOT SLOT SLOT at the theatre`, which after merging becomes `I SLOT at the theatre`.

We then fine-tune style-specific masked language-models BART_x and BART_y to fill in slots in the template and recover the style-specific text. During training, phrases in the input sentence are randomly discarded and the model is trained to fill the phrases back in (Lewis et al., 2020). Having trained style-specific masked language-models for both the source and target styles, we use both models to generate source and target filled-in text given style-agnostic templates.

$$\hat{x} = \text{BART}_x(t; \Theta_x) \tag{6}$$

$$\hat{y} = \text{BART}_y(t; \Theta_y) \tag{7}$$

In our running example, sampling with the positive language model yields the sentence `I saw a fantastic movie today at the theatre`, while sampling with the negative language model yields the sentence `I got ripped off today at the theatre`.

The last step we perform is a filtering step using the classifier. For synthesized examples in style k, we keep examples for which the style classifier predicts k. In other words, we keep only examples where the language models and the classifier agree. We find that this improves data quality and editor performance. We use the collection of synthesized source and target text pairs $\hat{x}$, $\hat{y}$ to train the editor.

3 Experimental Setup

We focus on two types of text style transfer: (1) **Sentiment transfer**, in which we transform a positive sentiment sentence to a corresponding negative sentiment sentence or vice-versa without changing the core content (i.e. attributes of the sentence not concerned with sentiment) (2) **Politeness transfer**, in which we transform the tone of a sentence from impolite to polite.

DATASET	**Attributes**	**Train**	**Valid**	**Test**
YELP	Positive	270K	2000	500
	Negative	180K	2000	500
AMAZON	Positive	277K	985	500
	Negative	278K	1015	500
POLITE	Polite	219K	26K	800
	Impolite	198K	24K	—

Table 1: Dataset statistics for style transfer tasks. The politeness corpus does not have parallel evaluation data and only evaluates on transfer from impolite to polite.

We make use of three datasets: YELP (Shen et al., 2017) consists of 450K sentences from busi-

Model	Acc	SBLEU	BLEU	SBERT	BERT
Baselines					
Input Copy	1.5	100.0	24.8	100.0	53.74
Reference	81.6	25.3	100.0	53.7	100.0
Generation methods					
Delete and Retrieve (Li et al., 2018)	88.6	36.8	12.2	48.5	33.3
Tag and Generate (Madaan et al., 2020)	86.2	47.1	19.8	57.9	37.2
DeepLatentSeq (He et al., 2020b)	83.8	48.4	18.7	57.9	36.0
Editing methods					
Masker (Malmi et al., 2020)	40.9†	—	14.5	—	—
LaserTagger (Malmi et al., 2019) + Masker data	49.6†	—	15.3	—	—
LaserTagger + our data	59.8	**71.8**	**24.8**	**81.3**	**51.6**
LEWIS	**93.1**	58.5	24.0	72.2	50.0

Table 2: Results on YELP. Results with † are taken from the classifier trained in Malmi et al. (2020) because the outputs for these models are not released.

Model	Accuracy	SBLEU	BLEU	SBERT	BERT
Input Copy	13.1	100.0	48.7	100.0	63.0
Delete and Retrieve (Li et al., 2018)	51.2	57.1	29.9	66.9	46.2
Tag and Generate (Madaan et al., 2020)	60.8	**68.7**	**34.8**	69.5	48.2
LEWIS	**74.3**	65.6	32.9	**75.2**	**52.2**

Table 3: Results on AMAZON

ness reviews for training and 1000 sentences released by Li et al. (2018) for testing, AMAZON (He and McAuley, 2016) consists of 540K sentences from product reviews for training and 1000 sentences for testing, and POLITE (Madaan et al., 2020), produced by filtering through the Enron Email corpus, consisting of 420K sentences for training and 800 sentences for testing. We list dataset statistics in Table 1.

3.1 Training Setup

We implement our models using `fairseq`[2] (Ott et al., 2019) and HuggingFace[3] (Wolf et al., 2020) — both based on the PyTorch library (Paszke et al., 2019). For BART-based generation models, we initialize with BART-base (Lewis et al., 2020), and train using a batch size of 65K tokens for 30000 iterations. We use a linear warmup schedule, reaching the peak of 3×10^{-5} at 5000 iterations, and then proceed to decay the learning rate with a polynomial decay schedule. For regularization, we use a dropout value of 0.3 and a weight decay value of 0.1. We optimize using Adam, with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.98$ and cross entropy loss. For RoBERTa-based taggers and classifiers, we initialize with RoBERTa-base (Liu et al., 2019), and train using a batch size of 256 for 5000 iterations. We optimize using Adam, warm up the learning rate to 1×10^{-6} and then decay with a cosine schedule. We train all models using mixed precision (Micikevicius et al., 2018) for faster training. Similar to prior work (Wu et al., 2019; Malmi et al., 2020), we decode using a beam width of 5 and rerank outputs produced by beam search using the likelihood of the classifier trained in Section 2.2.

3.2 Comparison with existing methods

We compare LEWIS to five prior methods: Delete, Retrieve, Generate (Li et al., 2018), a retrieval method that finds text from the target domain corpus whose style-agnostic form is similar to that of the source text; Tag and Generate (Madaan et al.,

---

[2] `https://github.com/pytorch/fairseq`

[3] `https://github.com/huggingface/transformers`

Model	Accuracy	SBLEU	SBERT
Delete, Retrieve, Generate (Li et al., 2018)	—	11.8	—
Tag and Generate (Madaan et al., 2020)	84.8	70.4	71.6
LEWIS	**87.4**	**75.3**	**81.4**

Table 4: Results on POLITE

Model	Acc	SBLEU	BLEU	SBERT	BERT
LM fill	90.3	42.9	17.4	58.9	41.6
Seq2Seq	87.3	50.0	19.3	68.5	**50.0**
LEWIS w/o filtering	91.2	50.1	23.3	69.8	48.0
LEWIS	**93.1**	**58.5**	**24.0**	**72.2**	**50.0**

Table 5: Ablation on YELP. “LM fill” is the ablation experiment in which we convert the source style text to a style-agnostic template and directly use the target style language model to synthesize a target style text (e.g. the editor is not used). “Seq2Seq” is a pretrained BART model that is fine-tuned on the synthesized data (e.g. a from-scratch generation model trained on the same data as the editor).

2020), a generation method that conditionally generates target text from style-agnostic source text; DeepLatentSeq (He et al., 2020b), an unsupervised machine translation-based approach where generators in each domain are regularized by a language model-based latent prior. Finally, we compare to previous editing approaches proposed by Malmi et al. (2019, 2020) where a single span in the source text two domain-specific language models disagree on is replaced.

3.3 Evaluation

Automatic Evaluation We use five evaluation metrics: BLEU (Papineni et al., 2002) measured against the reference (denoted as **BLEU**) to evaluate lexical overlap with human annotation; Self-BLEU measured against the source to measure content preservation (denoted as **SBLEU**); BERTScore and Self-BERTScore (Zhang et al., 2020) measured against the reference and the source (denoted as **BERT** and **SBERT** respectively); and accuracy measured against an external classifier (denoted as **Accuracy**) to measure how well the style was transferred.

		BLEU	BERT
Ref	great place , great food !	—	—
Hyp 1	pathetic place , great food !	**76.0**	71.5
Hyp 2	amazing place , awesome food !	0.0	**78.6**

Table 6: Example comparing BERTScore vs BLEU. **Ref** denotes the reference sentence and **Hyp 1** and **Hyp 2** represent two example hypotheses.

While measuring BLEU, Self-BLEU, and accuracy are standard for this task, we propose additionally using BERTScore due to its higher correlation with human judgments (Zhang et al., 2020). Compared to BLEU and Self-BLEU which are n-gram based, BERTScore is measured using token-wise cosine similarity between representations produced by BERT (Devlin et al., 2019).

Given this, the usage of BERTScore addresses the potential issue of accurately transferred sentences being scored poorly due to its low n-gram overlap. Table 6 shows an example of this where the style is accurately transferred but is scored poorly by BLEU as a result of low n-gram overlap.

Furthermore, following Malmi et al. (2020) who use a BERT-based classifier to score their outputs, we train a classifier initialized with RoBERTa-base (Liu et al., 2019). This model correctly classifies 98.2% of the YELP classification test set by Shen et al. (2017). Its accuracy is used to evaluate the output of style transfer models.

Human Evaluation We perform a robust human evaluation on all datasets, asking crowdworkers to rate 300 examples from Yelp (150 positive, 150 negative), 200 examples from Amazon (100 positive,100 negative) and 100 from Politeness. Five annotators rate each pair from 1 (strongly disagree) to 5 (strongly agree) in terms of fluency, content preservation (CP) and style transfer. We compare with our strongest baseline Tag and Generate (Madaan et al., 2020).

Dataset	Model	Fluency	CP	Style
YELP	TG	3.84±1.01	3.63±0.93	3.67±1.02
	LEWIS	**3.94**±0.99	**3.76**±0.88	**3.72**±0.98
AMAZON	TG	3.60±1.01	3.48±0.93	3.37±1.02
	LEWIS	**3.65**±0.88	**3.50**±0.88	3.37±0.90
POLITE	TG	3.83±0.84	3.76±0.90	3.48±1.04
	LEWIS	**3.93**±0.78	**3.87**±0.83	**3.63**±0.98

Table 7: Human evaluation results comparing LEWIS and Tag and Generate (TG)

	Mean	Std
# merged edit ops	1.57	0.78
# source toks	10.74	2.73
# style-agnostic template toks	10.42	2.86
# edit output toks	11.24	3.16

Table 8: Coarse-to-fine editor statistics on YELP, after merging consecutive edit operations of the same type, so that the number of operations denote spans as opposed to tokens (e.g. delete, replace).

4 Results

Performance of LEWIS compared to other methods on YELP, AMAZON, and POLITE are respectively shown in Tables 2, 3, and 4, with human evaluation shown in Table 7. LEWIS outperforms prior methods on all datasets in terms of accuracy, BLEU, and BERTScore: LEWIS achieves more successful transfers (2.6-13.5% accuracy depending on task), has higher overlap with human annotations (4-14.4 BERTScore), and retains more source content (5.7-14.3 Self-BERTScore). Human evaluation ($p < 0.01$ for YELP and POLITE using pairwise bootstrap sampling (Koehn, 2004)) shows that LEWIS outperforms Tag and Generate on fluency, content preservation and style across datasets. These results indicate that LEWIS is an effective method for style transfer. On the AMAZON dataset — which is noisier than the YELP dataset — LEWIS underperforms Tag and Generate when evaluating using BLEU, however when evaluating using BERTScore LEWIS outperforms the latter. When we inspect the output of LEWIS, we find that it generates more diverse output as shown in Figure 4.

One reason that LEWIS generates more diverse output is that unlike previous and concurrent editing work that use single-span replacement (Malmi et al., 2019, 2020), our method concurrently edits multiple spans with a larger set of operations. This is inherently supported by the editor (Figure 5) as well as encouraged during unsupervised data synthesis (Figure 4). Table 8 shows that a large number of examples do require multiple edits, and that the coarse-to-fine editor indeed performs multiple edit operations on average.

In addition to comparing end-to-end systems, we also compare LEWIS to concurrent editing and synthesis methods by Malmi et al. (2019, 2020). Table 2 shows that training the same model (LaserTagger) on our data improves and BLEU by 9.5 (the accuracy difference is not directly comparable since Malmi et al. (2020) used a BERT classifier and did not release model output). This suggests that our data synthesis procedure produces higher quality data than Malmi et al. (2020). Furthermore, because LaserTagger only performs single-span edits (when performing insertion/replacement), it often fails to transfer the style of the text. This also accounts for its high BLEU and BERTScore but low accuracy, as we show that a model that simply copies the input also achieves high BLEU and BERTScore but low accuracy. Replacing LaserTagger with our coarse-to-fine Levenshtein editor results in a sizable 33% gain in accuracy. In Table 1 of the Appendix, we show example outputs of these models for comparison. Finally, we ablate LEWIS to investigate how the different components of LEWIS affect performance.

Editing outperforms pure generation We replace the coarse-to-fine editor with a sequence-to-sequence BART model, which we also train with synthesized data. This is a strong baseline that outperforms prior pure generation work on style transfer, as shown in Table 3. Nevertheless, Table 5 shows that LEWIS outperforms this baseline on all metrics. This confirms our hypothesis that editing is a more effective means of style transfer compared to pure generation.

Training on synthesized data improves performance. Instead of training an editor using synthesized data, given a source text during inference, we convert it to a style-agnostic template and immediately fill it using the target language model. Table 5 shows that the resulting model underperforms both the sequence-to-sequence BART and the coarse-to-fine editor on all metrics. This result may be surprising, in that one expects the performance of a model trained on data synthesized by language models to be at-most on par with the per-

Orig	it certainly delivered .	i love the fresh , right out of the oven bread too .	moreover , they found ways to help save on the expense .
Template	`it SLOT delivered .`	`i SLOT the fresh, SLOT out of the oven SLOT.`	`SLOTover, SLOT ways to SLOT save on the expense.`
Synthesized source	it was well delivered. it was absolutely delivered.	i love the fresh , crisp bread out of the oven as well. i love the fresh , hot bread out of the oven as well.	overall , great place with ways to save on the expense. overall , good ways to help save on the expense.
Synthesized target	it was not delivered. it was never delivered.	i ordered the fresh , it came out of the oven cold . i ordered the fresh , it came out of the oven and was cold.	overall , there are better ways to try and save on the expense. overall , there are better ways to save on the expense .

Figure 4: Examples of synthesized parallel text on the Yelp dataset.

Source	we will not be coming back .	the food 's ok , the service is among the worst i have encountered .	i said it was disgusting to even serve this to diners .
Coarse edit type tagger output	`K K R K K K R`	`K K D R R K K K K K K R K K K K`	`K R K K R K K K K K K K K`
Fine generator input	we will <mask> be coming back <mask>	the food <mask> , the service is among the <mask> i have encountered .	i <mask> it was <mask> to even serve this to diners .
Fine generator output	we will definitely be coming back again !	the food is great , the service is among the best i have encountered.	i thought it was very nice to even serve this to diners.

Figure 5: Examples of coarse-to-fine editor output on the Yelp dataset. We abbreviate the edit operation with K for `<keep>`, D for `<del>`, and R for `<repl>`. Unlike previous and concurrent edit methods, we concurrently edit multiple spans in the text.

formance of the language model. In this case, we observe that training on the synthesized data actually improves over just using the language models. We hypothesize that this gain is due to the editor learning correlations between the source language model and the target language model, namely how to precisely transform the output of the source language model to the output of the target language model. The gains we observe here may be related to gains from training on back-translated or pseudo-parallel data (Sennrich et al., 2016; Edunov et al., 2018; He et al., 2020a). More research is needed to investigate the problem conditions under which such gains occur.

Filtering improves performance. Here, we forgo the filtering step, which removes $\approx$ 20% of the synthesized data on Yelp. Table 5 shows that filtering improves the quality of the synthesized data and leads to consistent gains.

5 Related Work

Text style transfer Previous work on style transfer can largely be divided into two categories: (1) learning a latent space with controllable attributes such as those found in Shen et al. (2017); John et al. (2019) or (2) using unsupervised generative approaches from retrieval (Li et al., 2018), tagging using style phrases (Madaan et al., 2020), to back-translation and unsupervised machine translation techniques (Prabhumoye et al., 2018; Lample et al., 2019; He et al., 2020b).

Editing for style transfer Our work is closest to Madaan et al. (2020) and Malmi et al. (2020). Madaan et al. (2020) use a tagger to mark style phrases in the source text, then generates the target text conditioned on the tagged source text. In contrast, we do not fully generate target text and only perform small, precise edits. In concurrent work to ours, Malmi et al. (2020) train a BERT language model on each style and edits a span where the models' likelihoods disagree the most. In contrast, instead of performing single-span replacement, our editor concurrently edits multiple spans in the text, and supports a wider set of operations than replacement. We showed that this results in more effective and more diverse style transfer. This coarse-to-fine transformations of text, in which the input context is progressively refined, has also led to improvements in syntactic parsing (Charniak and Johnson, 2005), semantic parsing (Dong and Lapata, 2018), and NER (Choi et al., 2018).

Unsupervised data synthesis for style transfer Malmi et al. (2020) also generate synthetic data with which to train an editing model from Malmi et al. (2019). Our synthesis differs from Malmi et al. (2020) in how slots for generation are chosen. In their work, the highest disagreeing span is chosen for rewriting. In our work, multiple spans with words whose attention weights that exceed the average are chosen for rewriting, which allows for more flexible and diverse samples. In turn, training on our synthesized data improves the performance and diversity of the style transfer model.

6 Conclusion

We proposed LEWIS, a coarse-to-fine editor for style transfer that transforms text using Levenshtein edit operations. Unlike prior edit methods, our methods concurrently performs multi-span edits. To train this editor, we proposed an unsupervised data synthesis procedure that converts text to style-agnostic templates using style classifier attention, then fills in slots in these templates using fine-tuned pretrained language models. LEWIS outperformed existing generation and editing style transfer methods on sentiment and politeness transfer. In addition, the proposed data-synthesis procedure increased transfer performance. Given the same synthesized data, our editor outperformed prior pure generation and editing methods. In future work, we will study the application of LEWIS to general sequence to sequence problems.

Ethical Considerations

This work has impact in the field of controlled text generation, and as with much of language technology has the potential to be both used for good and used maliciously. Our work learns to generate synthetic data in an unsupervised way, and is based on a pre-trained model (BART), which is likely to capture and amplify biases found in the data. Furthermore, as with all text-style transfer models, our model is amenable to malicious use, including impersonation and mass generation of faked opposing opinion, for example, negative and positive product reviews or political statements. We refer readers to Section 8 of Hu et al. (2020) for more on this topic. Given this, further study is needed to see how text-style transfer models are affected by the bias contained in the corpora they are trained on and other aspects to avoid undesired behaviors such as those listed in the cases above.

Acknowledgements

We thank Gabriel Ilharco, Edison Marrese-Taylor, Julian Michaels, Tongshuang Wu, Yutaro Yamada, and Luke Zettlemoyer, and the anonymous reviewers for their helpful feedback on and proofreading of this work. This work was partly supported by the Masason Foundation Fellowship awarded to Machel Reid.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,* *(AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 663–670. AAAI Press.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020a. Revisiting self-training for neural sequence generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020b. A probabilistic formulation of unsupervised text style transfer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

Zhiqiang Hu, Roy Ka-Wei Lee, Charu C. Aggarwal, and Aston Zhang. 2020. Text style transfer: A review and experimental evaluation. *arXiv preprint arXiv:2010.12742*.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, Online. Association for Computational Linguistics.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1869–1881, Online. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Eric Malmi, Aliaksei Severyn, and Sascha Rothe. 2020. Unsupervised text style transfer with padded masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8671–8680, Online. Association for Computational Linguistics.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and

Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6830–6841.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Mask and infill: Applying masked language model for sentiment transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5271–5277. ijcai.org.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Evaluation

Automatic Evaluation When evaluating using BLEU we used detokenized SacreBLEU (Post, 2018). For BERTScore (Zhang et al., 2020), we use the `rescale_with_baseline` option with the following hash: `roberta-large_L17_no-idf_version=0.3.5(hug_trans=4.1.0.dev0)-rescaled`.

Human Evaluation When using Amazon Mechanical Turk, we screen our annotators for English proficiency, and require all to have a greater than 95% approval rate. We hire workers with at least 1000 HITs and pay workers 5 cents per example, amounting to USD$10-15 per hour.

B Training Infrastructure

For training models we use between 1 and 8 NVIDIA V100 16GB GPUs on a DGX-1 machine running Ubuntu 16.04 on a Dual 20-Core Intel Xeon E5-2698 v4 2.2 GHz.

C Source Code & Synthetic Data

We release source code with this work, with preprocessing scripts, training scripts for both conditional lanaguage models, editors and coarse-grain taggers, edit operations extraction scripts, and synthetic data generation scripts at `https://github.com/machelreid/lewis`.

D Synthetic Data

For synthetic data generation, we generate approximately 2.2M pairs for Yelp, 2.0M pairs for Amazon, and 1M pairs for Polite. Note that when generating synthetic data on Polite, given the longer sequence length, we threshold the amount of `SLOT` tokens to be the minimum of one-third of the total sequence length and 6.

We release our synthetic data to help facilitate further development in approaches using synthetic data for this task.

E Qualitative Analysis

We analyzed 100 examples from YELP produced by Lewis. 83% transfers were correct,6% incorrect,and 11% ambiguous (the resulting sentence expressed both styles). This is in line with automatic metrics and shows Lewis is effective in successfully transferring style. For diversity of edits, in 59% of cases, Lewis inverted key phrases (*and enjoying this → and avoiding this*, *friendly folks, delicious authentic bagels → sorry folks,not authentic bagels*), in 26%, Lewis rewrote part of the sentence in a way that is not inverting key adjectives/nouns (and he loved it → and he said it was OK). In 10%, Lewis performed purely syntactic editing (*definitely not enough room → enough room*). In contrast to other editors that rely on primarily single-phrase inversion (e.g. LaserTagger), demonstrating that Lewis provides diverse edits.

F Further Automatic Evaluation

We further evaluate our model on semantic similarity and fluency using the classifiers released by Krishna et al. (2020). Results are shown in Table 9 and 10. Lewis improves fluency by a significant margin on all, and outperforms other methods on 2/3 datasets on semantic similarity.

Dataset	Model	Fluency
Yelp	Delete and Retrieve	38.7
	TG	53.1
	DeepLatentSeq	68.1
	Lewis	84.5
	Gold	89.4
Amazon	Delete and Retrieve	49.2
	TG	54.6
	Lewis	85.6
	Gold	84.5
Polite	TG	67.5
	Lewis	93.0
	Gold (src)	92.3

Table 9: Fluency classification from Krishna et al. (2020)

G Further automatic evaluation

We also conduct further human We also evaluate Yelp using the same 300 examples, asking five annotators which system (LEWIS, Tag and Generate, or Neither) they prefer. Results can be seen in Table 11. These results show general preference for LEWIS is consistent across fluency, content preservation, and style transfer.

Dataset	**Model**	**Target sim.**	**Source sim.**
YELP	Delete and Retrieve	49.2	61.0
	Deep Latent Seq	48.8	60.5
	TG	54.0	67.3
	LEWIS	61.2	77.4
	Gold	—	65.7
AMAZON	Delete and Retrieve	64.9	81.0
	TG	59.2	76.3
	LEWIS	59.6	78.8
	Gold	—	75.7
POLITE	TG	—	84.6
	LEWIS	—	87.4

Table 10: Semantic similarity classification from Krishna et al. (2020)

Attribute	**LEWIS**	**Tag and Generate**	**Neither**
Fluency	**50.2**	39.2	10.6
Content Preservation	**53.3**	38.3	8.4
Style	**45.2**	41.0	13.7

Table 11: Human evaluation with annotators indicating their preference when comparing aligned pairs from both LEWIS and Tag and Generate

H Example Outputs

Source	the wine was very average and the food was even less .
LEWIS	the wine was very good and the food was even better .
LaserTagger	the wine was very good and the food was even less .
Reference	the wine was above average and the food was even better
Source	owner : a very rude man .
LEWIS	owner : a very nice man .
LaserTagger	owner : a very man .
Reference	The owner was such a friendly person.
Source	i love the food ... however service here is horrible .
LEWIS	i love the food and the service here is great .
LaserTagger	i love the food ... however service here is great .
Reference	i love the food ... and service here is awesome .

Table 12: Three examples from the Yelp test set comparing the LaserTagger trained on our synthetic data and LEWIS

Constructing Flow Graphs from Procedural Cybersecurity Texts

Kuntal Kumar Pal[*], **Kazuaki Kashihara**[*], **Pratyay Banerjee**[*]
Swaroop Mishra, Ruoyu Wang, Chitta Baral
School of Computing, Informatics, and Decision Systems Engineering,
Arizona State University,
{kkpal, kkashiha, pbanerj6, srmishr1, fishw, chitta}@asu.edu

Abstract

Following procedural texts written in natural languages is challenging. We must read the whole text to identify the relevant information or identify the instruction-flow to complete a task, which is prone to failures. If such texts are structured, we can readily visualize instruction-flows, reason or infer a particular step, or even build automated systems to help novice agents achieve a goal. However, this structure recovery task is a challenge because of such texts' diverse nature. This paper proposes to identify relevant information from such texts and generate information flows between sentences. We built a large annotated procedural text dataset (CTFW) in the cybersecurity domain (3154 documents). This dataset contains valuable instructions regarding software vulnerability analysis experiences. We performed extensive experiments on CTFW with our LM-GNN model variants in multiple settings. To show the generalizability of both this task and our method, we also experimented with procedural texts from two other domains (Maintenance Manual and Cooking), which are substantially different from cybersecurity. Our experiments show that Graph Convolution Network with BERT sentence embeddings outperforms BERT in all three domains.

1 Introduction

Many texts in the real-world contain valuable instructions. These instructions define individual steps of a process and help users achieve a goal (and corresponding sub-goals). Documents including such instructions are called *procedural texts*, ranging from simple cooking recipes to complex instruction manuals. Additionally, *discussion in a shared forum or social media platform, teaching books, medical notes, sets of advice about social behavior, directions for use, do-it-yourself notices, itinerary guides can all be considered as procedural texts* (Delpech and Saint-Dizier, 2008). Most of these texts are in the form of natural languages and thus, lacking structures. We define *structure* as sentence-level dependencies that lead to a goal. These dependencies can vary based on the text-domain. Some examples of such dependencies are action traces, effects of an action, information leading to the action, and instruction order. Constructing structured flow graphs out of procedural texts is the foundation for natural language understanding and summarization, question-answering (QA) beyond factoid QA, automated workflow visualization, and the recovery of causal relationships between two statements. By flow-graph we mean both information and action flows in a text. However, the lack of structures in such texts makes them challenging to follow, visualize, extract inferences, or track states of an object or a sub-task, which ultimately makes constructing their flow graphs an insurmountable task.

Procedural texts are common in cybersecurity, where security analysts document how to discover, exploit, and mitigate security vulnerabilities in articles, blog posts, and technical reports, which are usually referred to as *security write-ups*. Practitioners in cybersecurity often use write-ups as educational and researching materials. Constructing structured flow graphs from security write-ups may help with automated vulnerability discovery and mitigation, exploit generation, and security education in general. However, automatically analyzing and extracting information from security write-ups are extremely difficult since they lack structure.

Figure 1 illustrates the core of a security write-up (broken into sentences) that carries instructions for exploiting a vulnerability in an online shopping service. S_1, S_3, and S_4 are the author's observations about the service's nature. Based on this information, S_5 and S_6 are two possible paths of actions.

---

[*] These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3945–3957
August 1–6, 2021.

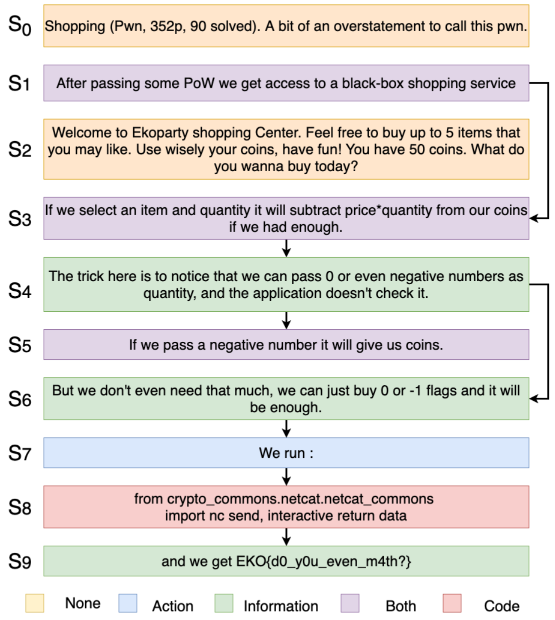

Figure 1: An example flow graph from the CTFW. Sentences in S_2 are merged into one block for clarity.

The author chose S_6 and ran the Python code in S_8 to exploit the service. S_0 and S_2 are irrelevant for the author's goal of exploiting this service.

Here we propose a novel approach to extract action paths out of structure-less, natural language texts by identifying actions and information flows embedded in and between sentences and constructing action flow graphs. Specifically, our focus is on procedural texts in the cybersecurity domain. We also show that constructing flow graphs helps extract paths of actions in domains besides cybersecurity, such as cooking and maintenance manuals.

Most previous works (Mori et al., 2014; Kiddon et al., 2015; Malmaud et al., 2014; Maeta et al., 2015; Xu et al., 2020; Mysore et al., 2019; Song et al., 2011) focus on fine-grained knowledge extraction from procedural texts in diverse domains. There are also a handful of works (Delpech and Saint-Dizier, 2008; Fontan and Saint-Dizier, 2008; Jermsurawong and Habash, 2015) that study the structure of natural language texts. Different from previous works, we extract structures and construct flow graphs from natural texts at the sentence level. This is because fine-grained domain-entity extraction tasks require a large amount of annotated data from people with specific in-depth domain knowledge, whereas text structures can be generalized.

Dataset. We built a dataset from security write-ups that are generated from past Capture The Flag competitions (CTFs). CTFs are computer security competitions that are usually open to everyone in the world. Players are expected to find and exploit security vulnerabilities in a given set of software services, and through exploiting vulnerabilities, obtain a *flag*—a unique string indicating a successful attempt—for each exploited service. Once the game is over, many players publish security write-ups that detail how they exploited services during the game. While these write-ups are a valuable educational resource for students and security professionals, they are usually unstructured and lacking in clarity. We collected 3617 CTF write-ups from the Internet, created a procedural text dataset, and invited domain experts to label each sentence for the purpose of constructing flow graphs and identifying action paths. To the best of our knowledge, this is the first attempt to use the knowledge embedded in security write-ups for automated analysis. The data and the code is publicly available [1] for future research.

This paper makes the following contributions:

- We built a new procedural text dataset, CTFW, in the cybersecurity domain. To the best of our knowledge, CTFW is the first dataset that contains valuable information regarding vulnerability analysis from CTF write-ups.
- We proposed a new NLU task of generating flow graphs from natural language procedural texts at the sentence level without identifying fine-grained named entities.
- We proposed four variations of a graph neural network-based model (LM-GNN) to learn neighbor-aware representation of each sentence in a procedural text and predict the presence of edges between any pair of sentences.
- We evaluated our models on CTFW. To the best of our knowledge, this is the first attempt in automated extraction of information from security write-ups. We also evaluated our models across three datasets in different domains and showed the generalizability of our approach.

2 Our Approach

We map each sentence of a procedural text as a node in a graph, and the action or information flows as edges. The task is then simplified into an edge prediction task: Given a pair of nodes, find if there is an edge between them. We learn feature representations of nodes using language models like BERT/RoBERTa (Devlin et al., 2018; Liu et al.,

---

[1] https://github.com/kuntalkumarpal/FlowGraph

Dataset Statistics	COR	MAM	CTFW
# Documents	297	575	3154
Avg size of document	9.52	8.12	17.11
Avg length of sentence	65.46	34.81	92.87
# Edges ($\|e^+\|$)	2670	5043	54539
$\|e^+\| : (\|e^+\| + \|e^-\|)$	0.18	0.12	0.07
Avg degree of node	1.83	1.76	1.88

Table 1: Dataset Statistics. $|e^+|$ is the total number of actual edges, and $|e^+| + |e^-|$ is the total number of edges possible. The in-degree of the starting node and out-degree of the end node are both 0.

2019). Then, to make the nodes aware of their neighboring sentences, we use Graph Neural Network (GNN) to update the node representations. We check for the edge between every pair of nodes in a graph and reduce the task to a binary classification during inference. This formulation enables us to predict any kind of structure from a document.

3 Dataset Creation

In this section, we present how we created three datasets on which we evaluated our approach. Table 1 shows the statistics for each datasets used.

3.1 CTF Write-ups Dataset (CTFW)

Each CTF competition has multiple challenges or tasks. Each task may have multiple write-ups by different authors. We crawled 3617 such write-ups from GitHub and CTFTime (CTFTime). Write-ups are unique and diverse but have common inherent principles. For each write-up, we provide two kinds of annotations: *sentence type* and *flow structure*. The writing style is informal with embedded code snippets and often contains irrelevant information.

Part of the annotations were provided as an optional, extra-credit assignment for the Information Assurance course. These CTF write-ups were directly related to the course-content, where students were required to read existing CTF write-ups and write write-ups for other security challenges they worked on during the course. Then students were given the option of voluntarily annotating CTF write-ups they read for extra credits in the course. For this task, we followed all the existing annotation guidelines and practices. We also ensured that (1) The volunteers were aware of the fact that their annotations would be used for a research project (2) They were aware that no PII was involved or would be used in the research project (3) They were aware that extra credits were entirely optional, and they could refrain from submitting at any point of time without any consequences (4) Each volunteer was assigned only 10-15 write-ups based on a pilot study we did ahead of time, annotating an average-length CTF write-up took about two minutes (maximum ten mins).

Remaining annotations were performed by the Teaching Assistants (TA) of the course. These annotations were done as part of the course preparation process, which was part of their work contract. All the TAs were paid bi-weekly compensation by the university or by research funding. It was also ensured that the TAs knew these annotations would be used for a research project, their PII was not involved and annotations were to be anonymized before using. We verified the annotations by randomly selecting write-ups from the set. Figure 1 shows a sample annotation.

Sentence Type Annotations. We split the documents into sentences using natural language rules. We then ask the volunteers to annotate the type of each statement as either Action (A), Information (I), Both (A/I), Codes (C), or irrelevant (None). *Action* sentences are those where the author specifies actions taken by them, whereas, *Information* statements mention observations of the author, the reasons and effects of their action. Sentences containing *codes* are assigned as C, and those which can be considered as both information and actions are marked as *Both* (A/I).

Flow structure Annotations. The second level of annotations is regarding the write-up structure. Each volunteer is given a csv file for each document with a set of sentence IDs and text for each write-up. They are asked to annotate the flow of information in the document by annotating the sentence id of some next possible sentences, which indicate the flow. We filter those write-ups which are irrelevant and those which did not have much detail (single line of valuable information). We call a write-up as irrelevant if it has no action-information annotations or if it has direct codes without any natural language description of steps to detect vulnerabilities. We only keep write-ups written in the English language for this work. Finally, we have 3154 write-ups with *sentence type* and *structure annotations*.

CTFTime website states that the write-ups are copyrighted by the authors who posted them and it is practically impossible to contact each author.

Such data is also allowed to use for academic research purposes(usc; euc). Thus, we follow the previous work using data from CTFTime (Švábenskỳ et al., 2021), and share only the urls of those write-ups which we use. We do not provide the scraper script since it would create a local copy of the write-up files unauthorized by the users. Interested readers can replicate the simple scraper script from the instructions in Appendix A and use it after reviewing the conditions under which it is permissible to use. We, however, share our annotations for those write-up files.

3.2 Cooking Recipe Flow Corpus (COR)

This corpus (Yamakata et al., 2020) provides 300 recipes with annotated recipe named entities and fine-grained interactions between each entity and their sequencing steps. Since we attempt to generate action flow graphs without explicitly identifying each named entity, we aggregate the fine-grained interactions between recipe named entities to generate sentence-level flows for each recipe. We reject three single-sentence recipes.

3.3 Maintenance Manuals Dataset (MAM)

This dataset (Qian et al., 2020) provides multi-grained process model extraction corpora for the task of extracting process models from texts. It has over 160 Maintenance Manuals. Each manual has fine-grained interactions between each entity and its sequencing steps. We use the annotations from sentence-level classification data and semantic recognition data for generating annotations of sentence-level flows for each process. Here also, we reject single sentence processes.

4 Model Description

Our goal is to find paths or traces of actions or information between texts. This needs an understanding of each sentence's interconnection. Hence, we modeled the problem into an edge prediction task in a graph using GNNs. We represent each sentence as a node and directed edges as information flows. Since this is procedural text (unidirectional nature) of instructions, we consider only the directed edges from one sentence S_n to any of its next sentences S_{n+i}. The node representations are learned using language models (LM) and GNNs.

4.1 Document to Sentence Pre-processing

Given a natural language document, first we split the document into sentences based on simple rules and heuristics. COR and MAM datasets already have document split into separate sentences. In the flow graph creation task, we filter out irrelevant sentences for the CTFW dataset based on the sentence type annotations. After this pre-processing task, each document (D_i) is converted into a series of sentences (S_j) where n is the number of valid sentences in a document.

$$D_i = \{S_0, S_1, S_2 ... S_{n-1}\}$$

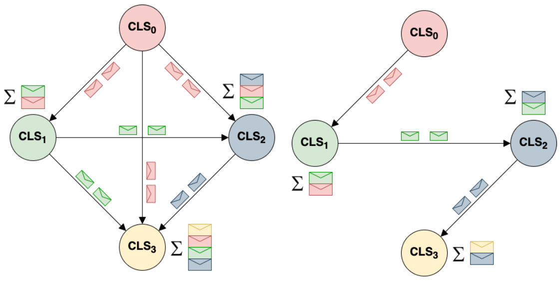

Figure 2: *Node Representation Learning for a document with four sentences in single-layer GNN.* Left: *Semi-Complete* Structure, Right: *Linear* Structure. During training, the sentence representation (CLS_i) are enriched using appropriate message passing techniques from the connected 1-hop neighbors.

4.2 Document to Graph Representation

A graph ($G = (V, E)$) is formally represented as a set of nodes ($V = \{v_0, v_1, ..\}$) connected by edges ($E = \{e_0, e_1, ..\}$ where $e_i = \{v_m, v_n\}$). We consider the sentences (S_j) of any document (D_i) as nodes of a directed graph (G_i). We experiment with two graph structure types for learning better node representation using GNN. First, we form local windows (W_N, where $N = 3, 4, 5, all$ sentences) for each sentence and allow the model to learn from all of the previous sentences in that window. We form the document graph by connecting each sentence with every other sentence in that window, with directed edges only from S_i to S_j where $i < j$. We do this since procedural languages are directional. We call this configuration *Semi-Complete*. Second, we consider connecting the nodes linearly where every S_i is connected to S_{i+1} except the last node. We call this *Linear* setting. Figure 2 shows the settings. We use LMs like BERT and RoBERTa to generate initial sentence representations. For each sentence (S_i), we extract the pooled sentence representation (CLS_{S_i}) of contextual BERT/RoBERTa embeddings (h_{S_i}).

We use CLS_{S_i} as node features for the graph (G_i).

$$h_{S_i} = BERT\,(\,[CLS]s_0s_1...s_{n-1}\,[SEP]\,)$$

4.3 Neighbor Aware Node Feature Learning

Since the LM sentence vectors are generated individually for each sentence in the document, they are not aware of other local sentences. So, through the *semi-complete* graph connection, the model can learn a global understanding of the document. However, the *linear* connection helps it learn better node representation conditioned selectively on its predecessor. We call the connected nodes as the neighbor nodes. We use Graph Convolutional Network (GCN) (Kipf and Welling, 2016) and Graph Attention Network (GAT) (Veličković et al., 2017) to aggregate the neighbor information for each node following the generic graph learning function (1)

$$\mathbf{H}^{l+1} = f(\mathbf{H}^l, \mathbf{A}) \tag{1}$$

where $\mathbf{A}$ is the adjacency matrix of the graph, $\mathbf{H}^l$ and $\mathbf{H}^{(l+1)}$ are the node representations at lth and $(l+1)$th layer of the network and f is the message aggregation function. In GCN, each node i, aggregates the representations of all of its neighbors $N(i)$ based on $\mathbf{A}$ and itself at layer l and computes the enriched representation $\mathbf{h}_i^{l+1}$ based on the weight matrix $\mathbf{\Theta}$ of the layer normalized by degrees of source $d(i)$ and its connected node $d(j)$ as per (2). In GAT, messages are aggregated based on multi-headed attention weights (α) learned from the neighbor node representations $\mathbf{h}_j^l$ following (3).

$$\mathbf{h}_i^{l+1} = \mathbf{\Theta} \sum_{j \in N(i) \cup \{i\}} \frac{1}{\sqrt{d(i)d(j)}} \mathbf{h}_j^l \tag{2}$$

$$\mathbf{h}_i^{l+1} = \alpha_{ii} \mathbf{\Theta} \mathbf{h}_i^l \; + \sum_{j \in N(i)} \alpha_{ij} \mathbf{\Theta} \mathbf{h}_j^l \tag{3}$$

4.4 Projection

We concatenate the neighbor aware node representations of each pair of nodes ($\mathbf{h}_i$;$\mathbf{h}_j$) from a graph and pass it through two projection layers with a GELU (Hendrycks and Gimpel, 2016) nonlinearity in between. We use the same non-linearity functions used by the BERT layers for consistency. We steadily decrease the parameters of each projection layer by half. During testing, given a document, we are unaware of which two sentences are connected. So, we compare each pair of nodes. This leads to an unbalanced number of existing (1) and non-existing (0) edge labels. Hence, we use weighted cross-entropy loss function as in equation (4) and (5), where L is the weighted cross-entropy loss, w_c is the weight for class c, i is the data in each mini-batch.

$$L(x,c) = w_c\Big(-x_c + log\Big(\sum_j exp(x_j)\Big)\Big) \tag{4}$$

$$L = \frac{\sum_{i=1}^{N} L(i, c_i)}{\sum_{i=1}^{N} w_{c_i}} \tag{5}$$

4.5 Training and Inference

Our training data comprises a set of sentences and the connections as an adjacency matrix for each document. Batching is done based on the number of graphs. GCN/GAT updates the sentence representations. A pair of node representations are assigned a label of 1 if there is an edge between them; otherwise, we assign them 0. Thus, we model it as a binary classification task as in equation (6) where f is the projection function, g is the softmax function, and y is the binary class output. Depending on the weighted cross-entropy loss, the node representations get updated after each epoch. During inference, the model generates node representations of each sentence in a test document, and we predict whether an edge exists between any two nodes in a given document graph.

$$y_c = \arg\max_k g(f(\mathbf{h}_i; \mathbf{h}_j), k) \quad c \in \{0, 1\} \tag{6}$$

5 Experiments

Datasets and Tasks: Each dataset is split into train, validation, and test sets in 70:10:20 ratio. The first task is identifying relevant information from raw CTF write-ups by classifying the type of each sentence. The second task is identifying information flows between sentences by predicting edges between sentence pairs, if any.

Metrics: We use *accuracy* as the evaluation metric for the Sentence Type classification task on CTFW. For the second task, because of the label imbalance we compare based on the *area under Precision-Recall curve* (PRAUC) and also report the corresponding *F1-score*. Hence do not report area under the ROC curve or accuracy.

We consider four settings for this task. The *no window* setting (W_{all}) checks whether there is an edge between any two statements in the given document. The comparisons required in this setting are directly proportional to the document's size. In CTFW, the size of each write-up is quite large. So,

	Models	CTFW		COR		MAM	
		PRAUC	F1	PRAUC	F1	PRAUC	F1
Baselines	Random	-	50.49	-	42.78	-	47.82
	Weighted Random	-	37.81	-	39.13	-	44.10
	BERT-NS	0.5751	26.12	0.5638	43.14	0.5873	29.73
	RoBERTa-NS	0.5968	32.44	0.5244	42.99	0.6236	39.65
Ours	BERT-GCN	0.7075	69.26	**0.6312**	58.13	**0.6888**	63.75
	RoBERTa-GCN	**0.7221**	69.04	0.6233	61.44	0.6802	65.73
	BERT-GAT	0.5585	61.93	0.4553	41.93	0.4568	62.18
	RoBERTa-GAT	0.5692	64.51	0.4358	24.74	0.4585	59.55

Table 2: Comparison with Baselines on Best Test Area under Precision-Recall Curve (PRAUC) and its corresponding F1 for **CTFW** (CTFwrite-up), **COR** (Cooking), **MAM** (Maintenance) datasets. NS is the next sentence based prediction approach. Our best model performance is bold, while maximum baseline performance is underlined.

to reduce unnecessary comparisons, we apply simple heuristics that instructions in procedural text, in general, does not have longer *direct* dependencies. Thus, using the windows, we can control each sentence's number of comparisons (node). To understand how the performances change we evaluate with a *sliding windows of N sentences* (W_N) where $N = 3, 4, 5$. The comparisons are only made with the next N sentences from a given sentence. For example, in case of W_5, for first sentence (S_1) we check for edges with S_2, S_3, S_4, S_5, S_6 and not S_7 on-wards. However, to have a fair comparison, we keep labeled out-of-window gold edges, if any. The ratios of existing and total edges in CTFW are 0.07 (W_{all}), 0.24 (W_5), 0.29 (W_4), 0.38 (W_3).

Training: We use Pytorch Geometric (Fey and Lenssen, 2019) for GNN and transformers (Wolf et al., 2020) for LM implementations. Training is done with AdamW (Loshchilov and Hutter, 2017) optimizer along with linear warmup scheduler on 4 Tesla V100 16GB GPUs. We use bert-base-uncased, bert-large-uncased, roberta-base and roberta-large versions as base model. We store the model with the best PRAUC score. Batch size of {4,8,16} and learning rates of {1e-5,5e-6} are used. Maximum sequence length varies between {64, 80, 128}. GNN depths are kept 128 (layer 1) and 64 (layer 2). We use a dropout of 0.4 in selected layers. For GAT, we keep four attention heads in layer 1. Details are present in Appendix C.

6 Results and Discussion

6.1 Sentence Type Classification (STC)

We use large and base versions of BERT and RoBERTa for this task to predict the type of sentences in a given text to establish a baseline for this task. This task helps to identify relevant and irrelevant sentences in a document. Each sentence is classified into any of Action, Information, Both, Code, and None. These fine-grained annotations can be used in later works for creating automated agents for vulnerability analysis. The processed data consists of 120751 samples for training, 17331 for validation, and 34263 for testing. Table 3 shows that RoBERTa-large performs better than the rest.

Model	**Val**	**Test**
BERT-Base	78.48±0.25	77.42±0.10
BERT-Large	78.19±0.48	77.13±0.20
RoBERTa-Base	78.85±0.25	77.37±0.11
RoBERTa-Large	79.02±0.16	**77.66±0.12**

Table 3: Sentence Type Classification (Mean Accuracy from three seed values). Best performance in bold.

6.2 Flow Structure Prediction

Here we present the performance results for the flow structure prediction.

Random Baseline: In the *Random* baseline, for every pair of nodes in each document we randomly select 0 (no-edge) or 1 (edge). For *Weighted Random* baseline, we choose randomly, based on the percentage of edge present in the train set. We only report F1 since there is no probability calculation.

Next Sentence-based Prediction (NS) Baseline: We use LMs like BERT and RoBERTa in a next sentence prediction setting to get the baselines. Each pair of sentences is concatenated using [SEP] token and passed through these language models. Using the pooled LM representation, we classify whether an edge exists between them or not. We show maximum PRAUC and its corresponding F1 for each dataset from the results of each of our window settings (W_3, W_4, W_5, W_{all}).

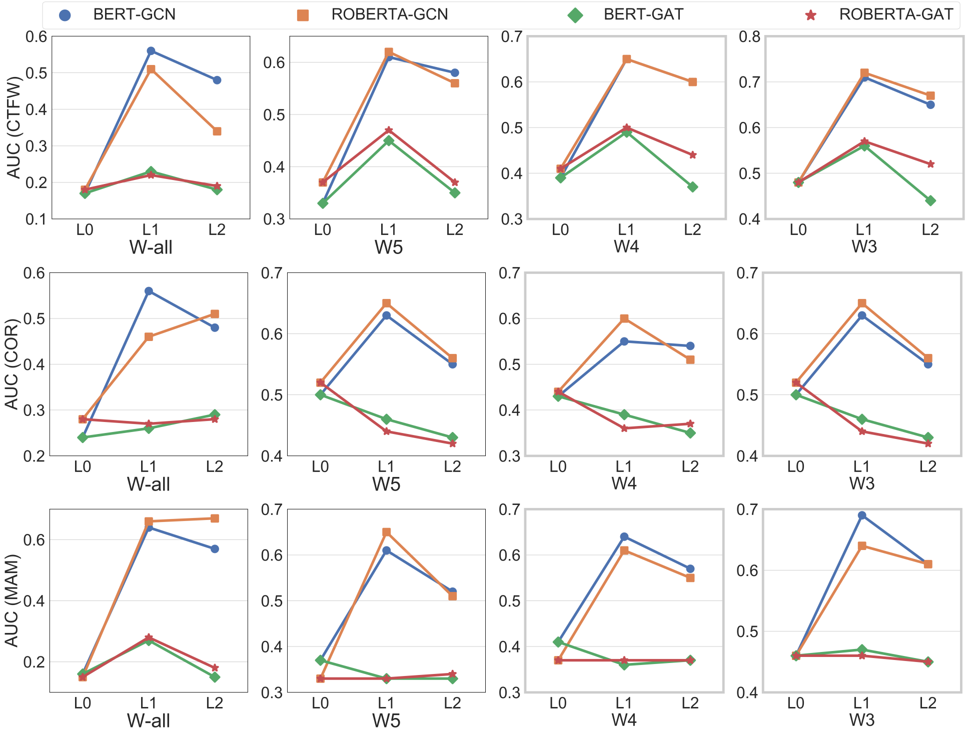

Figure 3: Effect of GNN Layers (L_0, L_1, L_2) on performance (PRAUC) of the models for W_{all}, W_5, W_4, W_3 settings on the three datasets

Models: We compare four variants of our LM-GNN models both with baseline and among each other in Table 2. The scores are overall best scores across single and double layers GNN (GCN/GAT) and LM (BERT/RoBERTa) after experiments with both base and large version, trained with pre-trained and randomly initialized weights.

We see that the best LM-GCN models outperform the best baseline model by 0.12, 0.07, 0.06 in PRAUC for CTFW, COR, and MAM datasets, respectively. However, the best LM-GAT scores falls short of the baselines indicating that the graph attentions on LM *sentence representations* cannot learn robust representation to perform this edge prediction task. Another thing to notice here is that, the best BERT-GCN models perform better than RoBERTa-GCN for COR and MAM datasets while performs poorly in the CTFW dataset. We hypothesize that this is because, the CTFW dataset has ten times more data than COR and six times more than MAM, which helps the RoBERTa model correctly predict the edges.

	W_3	W_4	W_5	W_{all}
CTFW-SC	0.6630	0.5985	0.5733	**0.5590**
CTFW-L	**0.7221**	**0.6520**	**0.6150**	0.3962
CTFW-EP	0.3700	0.2900	0.2400	0.0700
COR-SC	0.5639	0.5129	0.4731	**0.5580**
COR-L	**0.6456**	**0.6012**	**0.5274**	0.4034
COR-EP	0.3700	0.3100	0.2600	0.1700
MAM-SC	0.6528	0.6219	0.6091	**0.6718**
MAM-L	**0.6888**	**0.6362**	**0.6137**	0.4161
MAM-EP	0.4500	0.3700	0.3200	0.1500

Table 4: Effect of Semi-Complete(SC) and Linear(L) Graph Connection on 3 datasets in Area under Precision-Recall Curve (PRAUC). We also keep edge-percent (EP) in four window settings for comparison.

6.3 Analysis

Effect of Graph Connection Type: Table 4 shows how the models behave with semi-complete (SC) and linear (L) graph connection. For each dataset, we compare the PRAUC results for each window to draw more granular insight on the effect of neighbor aware representation learning. When we restrict graph learning by creating small windows (W_3, W_4, W_5), the linear model works better because of its selective learning conditioned on its predecessor. On the other hand, the semi-complete connection helps to learn a global awareness and works best in the W_{all} setting. It is important to note that each model performs better than the average PRAUC performance, which is the percentage of edges in the data indicating that the model is able to learn using the graph connections.

Effect of Graph Layers: We study how the depth of the GNNs affects the performance. We compare PRAUC across all four variations of the model in No-Window (W_{all}), W_5, W_4, W_3 settings in Figure 3. We experimented with no (L_0), single (L_1) and double (L_2) GNN layers. In all three datasets, we find the performance improves when we use a single layer and degrades beyond that for each of the windows with GCN based models. We do not go beyond two layers because of this observation and the graph connection types we use. We believe the reason for this drop (0.03-0.08 PRAUC) is that information from 2-hop neighbors might hinder the learning of the current node and confuse the model to predict wrongly. The GAT-based models mostly remain unaffected with the graph layers for both COR and MAM while showing some improvement in CTFW for one layer setting.

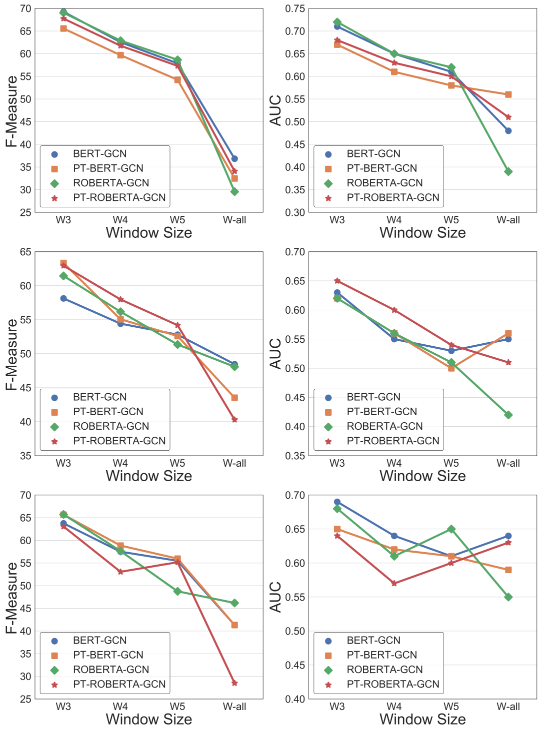

Figure 4: Performance for CTFW, COR, MAM trained from **scratch** and fine-tuned with **pre-trained weights**.

Effect of Pre-trained LM Weights: We study the impact of pre-trained weights of BERT and RoBERTa on the performance in Figure 4. We notice, for the three datasets, the performance slightly decreases when the pre-trained model weights are used. This observation may be because the texts' nature is quite different from the type of texts these LMs have been pre-trained on. The CTFW data often contains code fragments embedded in sentences, emoticons, or common conversational languages used in public forums.

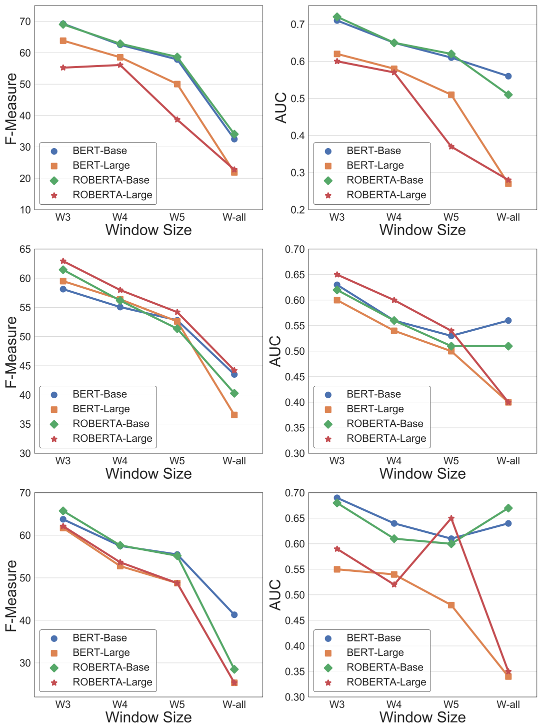

Figure 5: Performance on CTFW, COR, MAM trained with **base** and **large** version of the model.

Effect of LM Size: We also experimented with the size of sentence embeddings to see if that makes any difference to the performance. We use base and large version of BERT and RoBERTa for the experiments across three datasets. We present the impact on F1 and PRAUC in Figure 5. The performance of the larger versions of the models drop in all three datasets. This drop, we believe, is because the sentences in these texts are relatively short and help the smaller versions of the models with lesser parameters to learn better.

Other experiments: We also experimented with modifications of other parts of the models like changing the number of projection layers, projection layer sizes, the number of attention heads in the GAT model, or dropout percent in selected layers and modes of message aggregation (add, max, mean). We do not report them since they do not significantly change PRAUC values.

7 Related Work

Procedural knowledge extraction: There are attempts to extract structured knowledge from cooking instructions in the form of named entities (Malmaud et al., 2014), their sentence-level dependencies (Mori et al., 2014; Maeta et al., 2015; Xu et al., 2020), and action-verb argument flow across sentences (Jermsurawong and Habash, 2015; Kiddon et al., 2015; Pan et al., 2020). In other domains, extraction of clinical steps from medline abstracts (Song et al., 2011), extraction of material synthesis operations and its arguments in material science (Mysore et al., 2019), providing structures to how-to procedures (Park and Motahari Nezhad, 2018), and action-argument retrieval from web design tutorials (Yang et al., 2019) mostly focus on fine-grained entity extractions rather than action or information traces. The goal of our paper is constructing flow graphs from free-form, natural-language procedural texts without diverse domain knowledge. Hence, we refrain from training specialized named-entity recognizers for each domain to find specific entities. Our work is related to event or process discovery in process modeling tasks (Epure et al., 2015; Honkisz et al., 2018; Qian et al., 2020; Hanga et al., 2020), but our goal is not finding specific events or actions from procedural texts. In addition, the recent research proposed a method to create the forum structures from an unstructured forum based on the contents of each post using BERT's Next Sentence Prediction (Kashihara et al., 2020). However, we focus on building flow graphs for procedural texts using GNNs.

Graph Neural Networks: GNNs are important in reasoning with graph-structured data in three major tasks, node classification (Kipf and Welling, 2016; Hamilton et al., 2017), link prediction (Schlichtkrull et al., 2018), and graph classification (Ying et al., 2018; Pan et al., 2015, 2016; Zhang et al., 2018). GNNs help learn better node representations in each task using neural message passing (Gilmer et al., 2017) among connected neighbors. We consider two widely used GNNs, GCN (Graph Convolutional Network) (Kipf and Welling, 2016) and GAT (Graph Attention Networks) (Veličković et al., 2017) to learn sentence representation to provide a better edge prediction.

Edge Prediction Task: Edge or link prediction tasks (Li et al., 2018; Zhang and Chen, 2018; Pandey et al., 2019; Haonan et al., 2019; Bacciu et al., 2019) work mainly on pre-existing networks or social graphs as inputs and predict the existence of future edges between nodes by extracting graph-specific features. Different from existing work, we modeled the task of generating a graph-structure from a given natural-language text as an edge prediction task in a graph and learning representations of sentences considered as nodes.

Combinations of BERT and GCN: Recent works have used concatenation of BERT and GCN representations of texts or entities to improve performance of tasks like commonsense knowledge-base completion (Malaviya et al., 2019), text classification (Ye et al.; Lu et al., 2020), multi-hop reasoning (Xiao et al., 2019), citation recommendation (Jeong et al., 2019), medication recommendation (Shang et al., 2019), relation extraction (Zhao et al., 2019). Graph-BERT (Zhang et al., 2020) solely depends on attention layers of BERT without using any message aggregation techniques. However, we differ from each of the previous methods in terms of model architecture, where we use BERT to learn initial sentence representations and GCN or GAT to improve them by learning representations from its neighboring connected sentences. BERT-GAT for MRC (Zheng et al., 2020) created the graph structure from the well-structured wikipedia data whereas we explore two predefined natures of graph structures because of the free-formed text nature without such well-defined text-sections, presence of code-fragments, emoticons, and unrelated-token.

8 Conclusion and Future Work

We introduce a new procedural sentence flow extraction task from natural-language texts. This task is important for procedural texts in every domain. We create a sufficiently large procedural text dataset in the cybersecurity domain (CTFW) and construct structures from the natural form. We empirically show that this task can be generalized across multiple domains with different natures and styles of texts. In this paper, we only focus on English security write-ups. As part of future work, we plan to build automated agents in the cybersecurity domain to help and guide novices in performing software vulnerability analysis. We also plan to include non-English write-ups. We hope the CTFW dataset will facilitate other works in this research area.

Acknowledgement

The authors acknowledge support from the Defense Advanced Research Projects Agency (DARPA) grant number FA875019C0003 for this project.

Impact Statement

The dataset introduced here consists of write-ups written in public forums by students or security professionals from their personal experiences in the CTF challenges. The aggregated knowledge of such experiences is immense. This in-depth knowledge of the analysis tools and the approach to a problem is ideal for students working in software vulnerability analysis to learn from. Automated tutors built using such knowledge can reduce the efforts and time wasted in manually reading through a series of lengthy write-up documents.

CTFTime website states that the write-ups are copyrighted by the authors who posted them and it was practically impossible to contact each authors. It is also allowed to use the data for research purposes (usc; euc) Thus, we follow the previous work (Švábenskỳ et al., 2021) using data from CTFTime and share only the urls of those write-ups from the CTFTime website which we use. We do not provide the scraper script since it would create a local copy of the write-up files unauthorized by the users. Interested readers can replicate the simple scraper script from the instructions in Appendix A and use it after reviewing the conditions under which it is permissible to use. We, however, share our annotations for those write-ups files.

Part of the annotations were provided as an optional, extra-credit assignment for the Information Assurance course. These CTF write-ups were directly related to the course-content, where students were required to read existing CTF write-ups and write write-ups for other security challenges they worked on during the course. Then students were given the option of voluntarily annotating CTF write-ups they read for extra credits in the course. For this task, we followed all the existing annotation guidelines and practices. We also ensured that

- The volunteers were aware of the fact that their annotations would be used for a research project.
- They were aware that no PII was involved or would be used in the research project.
- They were aware that extra credits were entirely optional, and they could refrain from submitting at any point of time without any consequences.
- Each volunteer was assigned only 10-15 write-ups based on a pilot study we did ahead of time, annotating an average-length CTF write-up took about two minutes (maximum ten mins).

Remaining annotations were performed by the Teaching Assistants (TA) of the course. These annotations were done as part of the course preparation process, which was part of their work contract. All the TAs were paid bi-weekly compensation by the university or by research funding. It was also ensured that the TAs knew these annotations would be used for a research project, their PII was not involved and annotations were to be anonymized before using.

References

European union, copyright in the eu, 2020. `https://europa.eu/youreurope/business/running-business/intellectual-property/copyright/index_en.htm`. Accessed: 2021-05-01.

Us copyright office, copyright law of the united states, 2016. `https://www.copyright.gov/title17/92chap1.html#107`. Accessed: 2021-05-01.

Davide Bacciu, Alessio Micheli, and Marco Podda. 2019. Graph generation by sequential edge prediction. In *ESANN*.

CTFTime. CTFTime. `https://ctftime.org`. Accessed: 2021-05-01.

Estelle Delpech and Patrick Saint-Dizier. 2008. Investigating the structure of procedural texts for answering how-to questions. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elena Viorica Epure, Patricia Martín-Rodilla, Charlotte Hug, Rebecca Deneckère, and Camille Salinesi. 2015. Automatic process model discovery from textual methodologies. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pages 19–30. IEEE.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Lionel Fontan and Patrick Saint-Dizier. 2008. Analyzing the explanation structure of procedural texts: Dealing with advice and warnings. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 115–127.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

Khadijah Muzzammil Hanga, Yevgeniya Kovalchuk, and Mohamed Medhat Gaber. 2020. A graph-based approach to interpreting recurrent neural networks in process mining. *IEEE Access*, 8:172923–172938.

Lu Haonan, Seth H Huang, Tian Ye, and Guo Xiuyan. 2019. Graph star net for generalized multi-task learning. *arXiv preprint arXiv:1906.12330*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Krzysztof Honkisz, Krzysztof Kluza, and Piotr Wiśniewski. 2018. A concept for generating business process models from natural language description. In *International Conference on Knowledge Science, Engineering and Management*, pages 91–103. Springer.

Chanwoo Jeong, Sion Jang, Hyuna Shin, Eunjeong Park, and Sungchul Choi. 2019. A context-aware citation recommendation model with bert and graph convolutional networks. *arXiv preprint arXiv:1903.06464*.

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786.

Kazuaki Kashihara, Jana Shakarian, and Chitta Baral. 2020. Social structure construction from the forums using interaction coherence. In *Proceedings of the Future Technologies Conference*, pages 830–843.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Ji-chao Li, Dan-ling Zhao, Bing-Feng Ge, Ke-Wei Yang, and Ying-Wu Chen. 2018. A link prediction method for heterogeneous networks based on bp neural network. *Physica A: Statistical Mechanics and its Applications*, 495:1–17.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Zhibin Lu, Pan Du, and Jian-Yun Nie. 2020. Vgcn-bert: Augmenting bert with graph embedding for text classification. In *European Conference on Information Retrieval*, pages 369–382. Springer.

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. 2015. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2019. Exploiting structural and semantic context for commonsense knowledge base completion. *arXiv preprint arXiv:1910.02915*.

Jonathan Malmaud, Earl Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *LREC*, pages 2370–2377.

Sheshera Mysore, Zach Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. *arXiv preprint arXiv:1905.06939*.

Liang-Ming Pan, Jingjing Chen, Jianlong Wu, Shaoteng Liu, Chong-Wah Ngo, Min-Yen Kan, Yugang Jiang, and Tat-Seng Chua. 2020. Multi-modal cooking workflow construction for food recipes. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1132–1141.

Shirui Pan, Jia Wu, Xingquan Zhu, Guodong Long, and Chengqi Zhang. 2016. Task sensitive feature exploration and learning for multitask graph classification. *IEEE transactions on cybernetics*, 47(3):744–758.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and S Yu Philip. 2015. Joint structure feature exploration and regularization for multi-task graph classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):715–728.

Babita Pandey, Praveen Kumar Bhanodia, Aditya Khamparia, and Devendra Kumar Pandey. 2019. A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges. *Expert Systems with Applications*, 124:164–181.

Hogun Park and Hamid Reza Motahari Nezhad. 2018. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *Companion Proceedings of the The Web Conference 2018*, pages 351–358.

Chen Qian, Lijie Wen, Akhil Kumar, Leilei Lin, Li Lin, Zan Zong, Jianmin Wang, et al. 2020. An approach for process model extraction by multi-grained text classification. In *International Conference on Advanced Information Systems Engineering*, pages 268–282. Springer.

Kenneth Reitz. Requests: Http for humans. `https://requests.readthedocs.io/en/master/`. Accessed: 2020-10-23.

Leonard Richardson. 2007. Beautiful soup documentation. *April*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. 2019. Pre-training of graph augmented transformers for medication recommendation. *arXiv preprint arXiv:1906.00346*.

Sa-kwang Song, Heung-seon Oh, Sung Hyon Myaeng, Sung-Pil Choi, Hong-Woo Chun, Yun-Soo Choi, and Chang-Hoo Jeong. 2011. Procedural knowledge extraction on medline abstracts. In *International Conference on Active Media Technology*, pages 345–354. Springer.

spaCy. 2017. spacy v2.0. `https://spacy.io/models/en#en_core_web_md`.

Valdemar Švábenský, Pavel Čeleda, Jan Vykopal, and Silvia Brišáková. 2021. Cybersecurity knowledge and skills taught in capture the flag challenges. *Computers & Security*, 102:102154.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. *arXiv preprint arXiv:1905.06933*.

Frank F Xu, Lei Ji, Botian Shi, Junyi Du, Graham Neubig, Yonatan Bisk, and Nan Duan. 2020. A benchmark for structured procedural knowledge extraction from cooking videos. *arXiv preprint arXiv:2005.00706*.

Yoko Yamakata, Shinsuke Mori, and John A Carroll. 2020. English recipe flow graph corpus. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5187–5194.

Longqi Yang, Chen Fang, Hailin Jin, Walter Chang, and Deborah Estrin. 2019. Creative procedural-knowledge extraction from web design tutorials. *arXiv preprint arXiv:1904.08587*.

Zhihao Ye, Gongyao Jiang, Ye Liu, Zhiyong Li, and Jin Yuan. Document and word representations generated by graph convolutional network and bert for short text classification.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pages 4800–4810.

Jiawei Zhang, Haopeng Zhang, Li Sun, and Congying Xia. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yi Zhao, Huaiyu Wan, Jianwei Gao, and Youfang Lin. 2019. Improving relation classification by entity pair graph. In *Asian Conference on Machine Learning*, pages 1156–1171.

Bo Zheng, Haoyang Wen, Yaobo Liang, Nan Duan, Wanxiang Che, Daxin Jiang, Ming Zhou, and Ting Liu. 2020. Document modeling with graph attention networks for multi-grained machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6708–6718, Online. Association for Computational Linguistics.

A Extraction and Processing of Write-ups:

The extraction of CTF Write-up involved the following three phases.

Writeup URL extraction : We loop through all the write-up pages on ctftime website from page numbers 1 to 25500). We use a simple python scraper to scrape the content of each page using python requests (Reitz) library. We look for keyword "*Original write-ups*" and extracted the href component if present. These URLs are stored for each writeup indexed with the page numbers.

Write-up Content extraction : We use these URLs and extract the contents of the write-ups using python libraries requests and BeautifulSoup (Richardson, 2007). We extract all the text lines ignoring contents in html tags like style, scripts, head, title. The contents are stored in a text file named with the same page ids of the URLs.

Processing of Write-up : We processed and filter out sentences which do not have any verb forms using spacy (spaCy, 2017) POS-Tagger. We cleaned and removed unnecessary spaces and split them into sentences. The processing script is available in the github.

B CTFW Data Statistics

In CTFW, there are write-ups for 2236 unique tasks. Only four out of those having more than 5 write-ups each. 72% of the tasks have single write-up. The write-ups are from 311 unique competitions, ranging from years 2012-2019. A task having multiple write-ups vary in contents. In CTFW, only 3% of the tasks have more than three write-ups, and 9% have more than two.

C Training Details:

The correct set of hyperparameters are found by running three trials. We run for {50, 100} epochs and store the model with the best PRAUC score. Each training with evaluation takes around 1-3 hours for base version of models and around 6 hours for larger versions depending upon the dataset used. The model parameters are directly proportional to the model parameters of language models, since the GNN only allow few more parameters as compared to the LMs.

D Baseline NS with weighted cross-entropy

Table 5 shows the PRAUC values when we use weighted cross-entropy with base version of BERT on unbalanced data during training. The results are not much different than the Next-Sentence baseline shown previously.

Dataset	W_3	W_4	W_5	W_{all}
CTFW	0.4613	0.4397	0.2546	0.3681
COR	0.4724	0.4748	0.4837	0.4761
MAM	0.5318	0.2318	0.2297	0.4724

Table 5: BERT-base-uncased performance with NS prediction when Weighted Cross-Entropy used with Unbalanced Training Data

E Number of comparisons Reduction using Windows

We can control the total number of comparisons required to predict the edges in a graph by using the windows (W_N where $N = 3, 4, 5, all$). The number of comparisons for each window is given by the equation 7. We can reduce the number of comparisons considerably for large documents using shorter windows of 3, 4, 5 sentences. The number of comparison $\mathcal{C}$ is defined by

$$\mathcal{C} = \begin{cases} max\{(n-s), 0\}s + \frac{s(s-1)}{2} & n = 3, 4, 5 \\ \binom{n}{2} & n = all \end{cases} \tag{7}$$

F CTFW STC Label statistics

Table 6 shows the label distributions of Sentence Type Classification data.

Label	**Train**	**Val**	**Test**
A	11143	1499	3321
I	23279	3075	6882
A/I	2931	380	826
C	1386	185	338
NONE	82012	12192	22896

Table 6: CTFW Sentence Type Classification

Cluster-Former: Clustering-based Sparse Transformer for Question Answering

Shuohang Wang Luowei Zhou Zhe Gan Yen-Chun Chen
Yuwei Fang Siqi Sun Yu Cheng Jingjing Liu

Microsoft Corporation
`{shuowa, luowei.zhou, zhe.gan, yen-chun.chen}@microsoft.com`
`{yuwfan, siqi.sun, yu.cheng, jingjl}@microsoft.com`

Abstract

Transformer has become ubiquitous in the deep learning field. One of the key ingredients that destined its success is the self-attention mechanism, which allows fully-connected contextual encoding over input tokens. However, despite its effectiveness in modeling short sequences, self-attention suffers when handling inputs with extreme long-range dependencies, as its complexity grows quadratically *w.r.t.* the sequence length. Therefore, long sequences are often encoded by Transformer in chunks using a sliding window. In this paper, we propose *Cluster-Former*, a novel clustering-based sparse Transformer to perform attention across chunked sequences. The proposed framework is pivoted on two unique types of Transformer layer: Sliding-Window Layer and Cluster-Former Layer, which encode local sequence information and global context jointly and iteratively. This new design allows information integration beyond local windows, which is especially beneficial for question answering (QA) tasks that rely on long-range dependencies. Experiments show that Cluster-Former achieves state-of-the-art performance on several major QA benchmarks.

1 Introduction

Long-range contextual understanding has proven critical in many natural language processing (NLP) tasks. For example, the relevant context for correctly answering an open-domain question can arch over thousands of words (Chen et al., 2017). Encoding long sequences via deep neural networks, however, has remained an expensive and challenging task due to high demand on training time and GPU memory. Traditional sequence modeling methods (Hochreiter and Schmidhuber, 1997) encode long sequences in a chronological order, which suffers high latency. In the place of sequential encoding, recent models such as Transformer (Vaswani et al., 2017) use simultaneous self-attention over the entire input instead, which has been successfully adopted in many NLP tasks such as textual entailment (Devlin et al., 2019), dependency parsing (Zhou and Zhao, 2019), and summarization (Lewis et al., 2019). A caveat with Transformer though is that building full connections over long sequences translates to quadratic growth on memory demand and computational complexity *w.r.t.* sequence length.

One way to efficiently encode long sequences is to first chunk a sequence into much shorter ones with a sliding window, then build connections between the shorter sequences (Figure 1(a)). For example, Child et al. (2019), Beltagy et al. (2020) and Zaheer et al. (2020) apply sparse attention to chunked sequences in hand-designed patterns in order to gather information from the chunks (Figure 1(b)). Choi et al. (2017) and Wang et al. (2019) first use a simpler model to filter chunked sequences, then process selected sequences with fully-connected self-attention. Rae et al. (2019) makes use of the shared memory of chunked sequences to build connections between them. However, these methods cannot encode long-range dependencies with as much flexibility or accuracy as fully-connected self-attention, due to their dependency on hand-designed patterns.

Recently, several studies (Kitaev et al., 2020; Tay et al., 2020a) propose to further improve the sparse attention mechanism by hashing or sorting the hidden states into different buckets (Figure 1(c)). These works mainly explore tasks with relatively short sequences, such as sentence-level machine translation, where the number of hashing vectors is relatively small (less than 16 in Kitaev et al. (2020)), allowing randomly initialized hashing vectors to hash hidden states into correct buckets. However, how to use hashing-based attention in the context of long sequences (*e.g.*,, up to

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3958–3968
August 1–6, 2021.

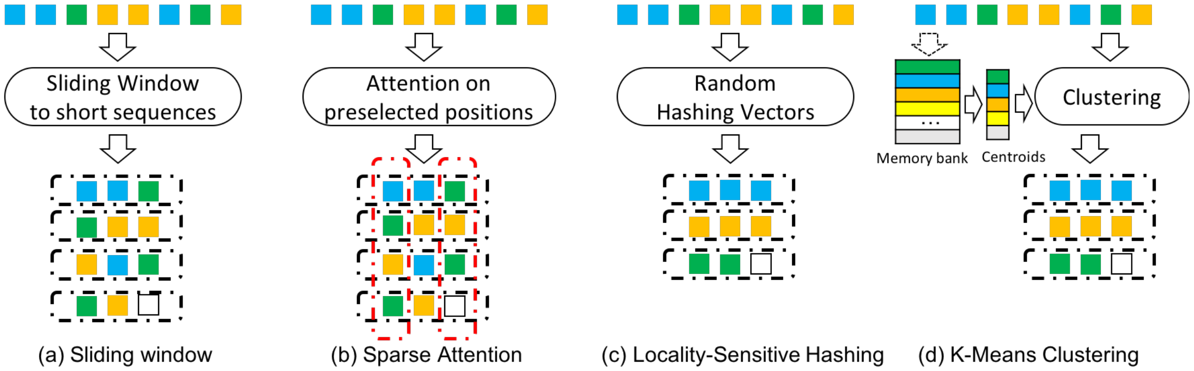

(a) Sliding window (b) Sparse Attention (c) Locality-Sensitive Hashing (d) K-Means Clustering

Figure 1: Illustration of different methods for processing long sequences. Each square represents a hidden state. The black-dotted boxes are Transformer layers. (a) is the sliding-window-based method to chunk a long sequence into short ones with window size 3 and stride 2. (b) builds cross-sequence attention based on sliding window over pre-selected positions (red-dotted boxes). (c) hashes the hidden states into different buckets by randomly-initialized vectors. (d) is our proposed approach to cluster the hidden states. Our final model is a combination of (a) and (d) that processes both local and global context.

thousands of words) is still an unexplored territory.

Our proposed framework for efficient long sequence encoding, *Cluster-Former*, marries both sliding-window and hashing-based methods to achieve effective local and long-range dependency encoding. Cluster-Former consists of two types of encoding layer. The first one (noted as *Sliding-Window Layer*) focuses on extracting local information within a sliding window. It applies Transformer to the hidden states of each chunked sequence independently, as shown in Figure 1(a). The other one (noted as *Cluster-Former Layer*) learns to encode global information beyond the initial chunked sequences. Specifically, we first apply clustering to the input hidden states so that similar hidden states are assigned to the same cluster, as shown in Figure 1(d). The clustered and sorted input is then divided uniformly into chunks, each encoded by a Transformer layer. Note that to make model training more efficient, the cluster centroids are not computed online but updated periodically (every epoch or a few epochs). We accumulate the hidden states from the layer prior to the Cluster-Former layer in a memory bank, and apply the K-Means algorithm to form cluster centroids during each update cycle. Compared to previously discussed sparse attention based on pre-selected positions (Figure 1(b)) or randomly-initialized hashing vectors (Figure 1(c)), experimental results show that our method can encode dependency across chunked sequences more effectively.

Our contributions can be summarized as follows. (i) We propose Cluster-Former, a novel approach to capturing long-range dependencies more effectively than locality-sensitive hashing method. (ii) We propose a new Transformer-based framework to process long sequences by combining Sliding-Window and Cluster-Former layers to extract both local and global contextual information. (iii) Our model achieves the best performance on question answering datasets of Natural Questions (long answer), SearchQA, and Quasar-T.

2 Related Work

Efficient Transformers With Transformer models growing larger and larger, how to handle longer sequences arises as a critical challenge. Many works have been proposed to improve the computational and memory efficiency of Transformers, including Sparse Transformer (Child et al., 2019), Set Transformer (Lee et al., 2019), Routing Transformer (Roy et al., 2020), Fast Transformer (Vyas et al., 2020), Reformer (Kitaev et al., 2020), Sinkhorn Transformer (Tay et al., 2020a), Longformer (Beltagy et al., 2020), ETC (Ainslie et al., 2020), Synthesizer (Tay et al., 2021), Performer (Choromanski et al., 2020), Linformer (Wang et al., 2020), Linear Transformer (Katharopoulos et al., 2020), and BigBird (Zaheer et al., 2020). Tay et al. (2020b) provided an excellent literature survey on this emerging topic. Our method falls into the setting of learnable sparse-attention patterns.

Among all these works, our method is closer to Set Transformer (Lee et al., 2019), Routing Transformer (Roy et al., 2020), and Fast Trans-

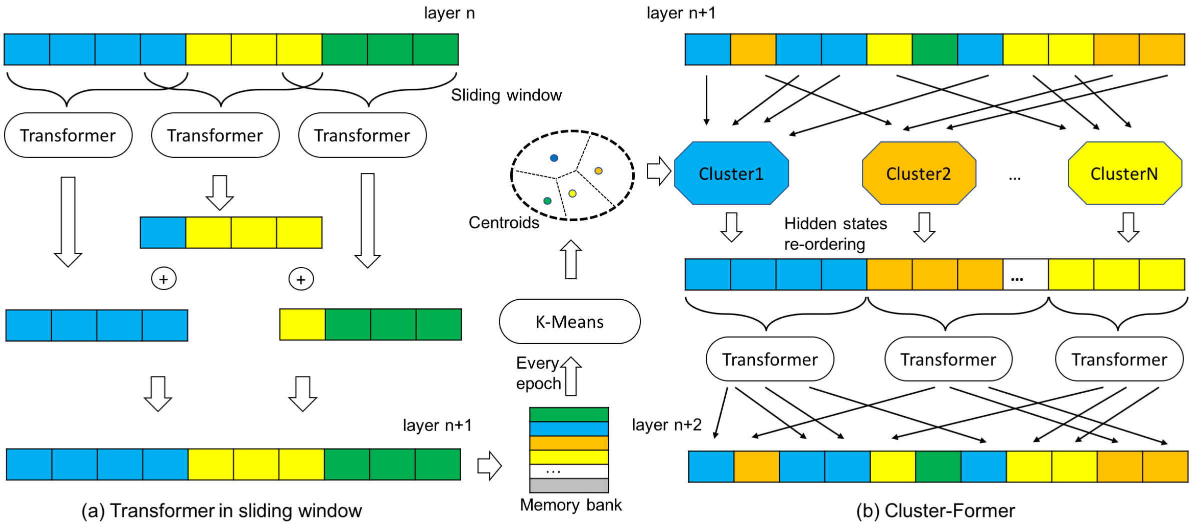

(a) Transformer in sliding window
(b) Cluster-Former

Figure 2: An overview of the proposed Transformer layer. (a) Sliding-Window layer over a sequence. (b) Cluster-Former layer over clustered hidden states from the output of (a). Cluster centroids are periodically updated based on the memory bank of the hidden states in the corresponding layer.

former (Vyas et al., 2020), which all use cluster centroids to learn patterns. However, we target at solving a different task, question answering. And it also leads to a significant different framework to encode a short question with a long context, other than a single long sequence, such as language modeling task. Moreover, our cluster centroids are updated in a very different way by periodical centroids update with K-Means on memory bank, other than memory-based centroids (Lee et al., 2019), exponentially moving centroids (Roy et al., 2020), or online clustering (Vyas et al., 2020).

Long Sequence in Question Answering For tasks such as open-domain question answering (Chen et al., 2017), a large volume of documents or paragraphs is usually retrieved to infer the answer, yielding extremely long context content. Despite the fact that state-of-the-art NLP models are capable of extracting answers amid complex context, they still struggle with extremely long input sequences. Recent advances that advocate the use of large-scale pre-trained models (Lewis et al., 2019; Liu et al., 2019; Lan et al., 2020) for question answering make this problem more prominent, due to tremendous memory consumption. To process long sequences, the most widely-used method is to first use a lightweight model to filter out redundant text, then use sliding-window-based approaches to encode the remaining sequences with a more sophisticated model. Chen et al. (2017) integrated bi-gram features into Information Retrieval (IR) methods to retrieve related documents more accurately. Wang et al. (2018) trained a paragraph selector using as the reward whether the entire system can obtain the correct answer or not. Asai et al. (2020) trained a recurrent retriever to select paragraphs for multi-hop question answering. Izacard and Grave (2021) proposed to fuse local encoded information into a decoder for answer generation. Besides the above methods, directly applying Efficient Transformers to process long sequences in question answering is another option. In this paper, we focus on this direction by directly training our Cluster-Former on the long context without using lightweight model for context filtering.

3 Proposed Approach

The proposed framework to handle long sequences is pivoted on two types of Transformer layer: (i) *Sliding-Window* Layer; and (ii) *Cluster-Former* Layer. The former focuses on encoding local sequence information, while the latter is on encoding global context and always built on top of the former layer. An overview of the two layers is illustrated in Figure 2.

3.1 Sliding-Window Layer

Despite that our focus is on capturing long-range dependencies for global context, local information also plays a critical role for knowledge propagation. Therefore, in the lower section of our network, we adopt the traditional sliding-window encoding mechanism. A sliding window segments a long sequence X into short, overlapping ones with window size l and stride m, as illustrated in Figure 2(a). Note that in this paper, we focus on question answering tasks, for which we concatenate the question Q with each sequence chunked from the document:

$$\mathbf{H}_k^0 = \left[\mathbf{Q}; \mathbf{X}\left[m \times k : (m \times k + l)\right]\right], \quad (1)$$

where $\mathbf{Q} \in \mathbb{R}^{q \times d}$ denotes question embeddings given a QA task, q is the number of tokens in the question, and $\mathbf{X} \in \mathbb{R}^{x \times d}$ is the embeddings for all context, x is the number of tokens in context. k is the ID of the chunked sequence, l is the window size, and m is the stride of the sliding window. $[idx_1 : idx_2]$ indicates selecting rows between the index of idx_1 and idx_2 of the matrix. $[\cdot;\cdot]$ means concatenating the matrices along the row. We first use Transformer to encode each sequence in sliding window as follows:

$$\mathbf{H}_k^{n+1} = \text{Transformer}(\mathbf{H}_k^n), \quad (2)$$

where $\mathbf{H}_k^{n+1} \in \mathbb{R}^{(q+l) \times d}$ is the output of Transformer on the k-th sequence in the n-th layer, while it is not the final output of the n-th layer. As we expect the neighbouring sequences to share useful information in hidden states as well, we always set $m < l$ to allow overlapping between sequences. We use the mean values of the Transformer hidden states at the overlapped tokens between windows as final outputs. To merge the representations from the $(k-1)$-th sequence:

$$\begin{aligned} \mathbf{H}_k^{n+1}[q : q+l-m] \quad &+= \quad \mathbf{H}_{k-1}^{n+1}[q+m : end], \\ \mathbf{H}_k^{n+1}[q : q+l-m] \quad &/= \quad 2, \end{aligned}$$

and merge representations from $(k+1)$-th sequence:

$$\begin{aligned} \mathbf{H}_k^{n+1}[q+m : end] \quad &+= \quad \mathbf{H}_{k+1}^{n+1}[q : q+l-m], \\ \mathbf{H}_k^{n+1}[q+m : end] \quad &/= \quad 2, \end{aligned} \quad (3)$$

where $+=$ is to add matrices in-place and $/=$ is to divide a matrix by a scalar value in-place. The merged hidden states $\mathbf{H}_k^{n+1} \in \mathbb{R}^{(q+l) \times d}$ are the final outputs of the n-th layer. If the next layer is Cluster-Former, the output hidden states in this layer $\mathbf{H}_k^{n+1}$ will be saved into memory bank for computing the cluster centroids.

Algorithm 1 Cluster Centroids Update

```
Initialize Memory = Queue()
Centroids = GetCentroids(RandomVector)

function Train(Inputs)
    for $i$ = 1, 2,..., IterationNum do
        States = Sliding-Transformer(Inputs[$i$])
        Memory.add(States)
        while len(Memory) > M do
            Memory.pop()
        end while
        if $i$ % ClusterUpdateFrequency == 0 then
            Centroids = GetCentroids(Memory)
        end if
        Clusters = cluster States by Centroids
        States = Cluster-Former(Clusters)
    end for
end function

function GetCentroids(HiddenStates)
    Centroids = K-Means(HiddenStates)
    Outputs = List()
    Outputs[1] = Centroids[1]
    for $i$ = 2, 3,..., ClusterNum do
        Outputs[$i$] = centroid from Centroids
                        that is closest to Outputs[$i-1$]
                        but not in Outputs
    end for
    return Outputs
end function
```

3.2 Cluster-Former Layer

We introduce a Cluster-Former layer to add global representational power to Transformer beyond sliding windows. An in-depth visualization of the layer is illustrated in Figure 2(b).

The input of the Cluster-Former layer comes from the hidden states of the prior layer (in our case a Sliding-Window layer). After merging the overlaps between sequence chunks, the input of this layer is defined as:

$$\bar{\mathbf{H}}^n = \left[\mathbf{H}_0^n[0 : q+m]; ...; \mathbf{H}_k^n[0 : q+m]\right], \quad (4)$$

where $\bar{\mathbf{H}}^n \in \mathbb{R}^{(q\lceil x/m \rceil + x) \times d}$ is the hidden states to cluster, x is the number of tokens in the context.

As the hidden states with larger cosine similarity are more likely to have higher attention weights, we build sparse self-attention only on the hidden states in the same cluster. In this work, we use K-Means as the chosen clustering method for simplicity. More advanced clustering algorithms have the potential of yielding better performance. Since running K-Means on the fly in each training iteration is computationally expensive, we decide to re-compute the cluster centroids with low frequency (every epoch or a few epochs).

In addition, to avoid dramatic changes in the cluster centroids due to limited hidden state inputs, we maintain a memory bank for the most recent hidden states. The entire procedure is depicted in Algorithm 1. Once we compute the cluster centroids, we can directly use them for hidden state clustering as follows:

$$\mathbf{v}^n = \text{argmax}\Big(\frac{\mathbf{H}^n(\mathbf{C}^n)^T}{||\mathbf{H}^n||_2||\mathbf{C}^n||_2}\Big), \quad (5)$$

where $\mathbf{C}^n \in \mathbb{R}^{p\times d}$ are the cluster centroids for layer n, and p is the pre-defined number of clusters. The function argmax$(\cdot)$ performs on the last dimension and assigns all the input hidden states into different clusters based on the max value of cosine similarity between the hidden states and cluster centroids. $\mathbf{v}^n \in \mathbb{R}^{(q\lceil x/m\rceil+x)}$ is the assigned cluster IDs of all the input hidden states.

Since the number of hidden states in different clusters can vary substantially, padding them to the maximum length for Transformer training will significantly increase the computational time. To make the extraction of global context more efficient, we greedily pick the cluster centroids based on the nearest neighbour (measured by cosine similarity) as shown in the function GETCENTROIDS in Algorithm 1. Thus, the hidden states with similar cluster IDs are also close to each other. Then, we can directly sort the cluster IDs of hidden states and uniformly chunk the hidden states (same window size and stride m):

$$\begin{aligned}\mathbf{u}^n &= \text{argsort}(\mathbf{v}^n),\\ \mathbf{a}^n_k &= \mathbf{u}^n[mk : m(k+1)],\\ \mathbf{E}^n_k &= \mathbf{H}^n[\mathbf{a}^n_k], \end{aligned}\quad (6)$$

where the function argsort$(\cdot)$ is to obtain the indexes of input values sorted in order (same values sorted by the corresponding position of hidden states). $\mathbf{a}^n_k \in \mathbb{R}^m$ is the chunked indexes of the hidden states. $\mathbf{E}^n_k \in \mathbb{R}^{m\times d}$ is the k-th clustered hidden states, and we will run Transformer on top of it to build the connection beyond the words in the initial sliding window as follows:

$$\mathbf{E}^{n+1}_k = \text{Transformer}(\mathbf{E}^n_k). \quad (7)$$

After updating the hidden states, we map them back to the order before clustering:

$$\begin{aligned}\bar{\mathbf{H}}^{n+1} &= [\mathbf{E}^{n+1}_0; \mathbf{E}^{n+1}_1; ...; \mathbf{E}^{n+1}_K],\\ \bar{\mathbf{a}}^n &= [\mathbf{a}^n_0; \mathbf{a}^n_1; ...; \mathbf{a}^n_K], \quad (8)\\ \bar{\mathbf{H}}^{n+1}[\bar{\mathbf{a}}^n] &= \text{clone}(\bar{\mathbf{H}}^{n+1}), \quad (9)\end{aligned}$$

	#train	#test	med	max
Quasar-T	29k	3k	2.8k	8.2k
SearchQA	100k	27k	2.5k	4.9k
NQ	292k	8k	6.3k	128k

Table 1: Statistics of Question Answering datasets. #train: number of questions in the training set. #test: number of questions in the test set. med: median length of the context. max: max length of the context.

where $\bar{\mathbf{H}}^{n+1}$ is the final output hidden state of this layer and has the same word order as the input $\bar{\mathbf{H}}^n$. In experiments, we stack these two types of layer interchangeably to capture both global and local context efficiently.

4 Experiments

4.1 Datasets

We evaluate our proposed approach on multiple question answering benchmarks. The statistics of all the datasets are summarized in Table 1.

- **Quasar-T**[1] (Dhingra et al., 2017): The goal of this task is to answer open-domain questions from Trivia Challenge. All the passages harvested through information retrieval can be used to answer questions. The task requires the model to generate answers in phrases. The evaluation metric on this dataset is based on Exact Match and F1 score of the bag-of-words matching. Our evaluation tool[2] comes from the SQuAD dataset.
- **SearchQA**[3] (Dunn et al., 2017): The setting of this dataset is the same as Quasar-T, except that the questions are sourced from Jeopardy! instead.
- **Natural Questions**[4] (Kwiatkowski et al., 2019): This task aims to answer questions based on a given Wikipedia document, and has two settings. (i) Long answer: select a paragraph that can answer the question based on the Wikipedia document if any. (ii) Short answer: extract an answer phrase from the document if the document contains the answer. As the given document may not contain answer, we can either predict an answer or predict no answer. The evaluation metric on this dataset is the F1 score, where true positives are exactly correct answers, false positives are

---

[1] https://github.com/bdhingra/quasar

[2] https://rajpurkar.github.io/SQuAD-explorer/

[3] https://github.com/nyu-dl/dl4ir-searchQA

[4] https://ai.google.com/research/NaturalQuestions

	Quasar-T EM/F1	SearchQA EM/F1	NQ(long) F1	NQ(short) F1
R3 (Wang et al., 2018)	35.3/41.7	49.0/55.3	-	-
DECAPROP (Tay et al., 2018)	38.6/46.9	62.2/70.8	-	-
DS-QA (Lin et al., 2018)	42.2/49.3	58.8/64.5	-	-
Multi-passage BERT (Wang et al., 2019)	51.1/59.1	65.1/70.7	-	-
DrQA (Chen et al., 2017)	37.7/44.5	41.9/48.7	46.1	35.7
DecAtt + DocReader (Kwiatkowski et al., 2019)	-	-	54.8	31.4
$\text{BERT}_{\text{joint}}$ (Alberti et al., 2019)	-	-	64.7	52.7
BERT_{wwm} + SQuAD2 (Pan et al., 2019)	-	-	68.2	57.2
RikiNet-RoBERTa (Liu et al., 2020)	-	-	75.3	**59.3**
Sliding Window	52.9/62.8	65.8/73.2	75.3	56.4
Sparse Attention (Child et al., 2019)	52.1/62.0	64.7/71.7	74.5	56.1
Locality-Sensitive Hashing (Kitaev et al., 2020)	53.2/62.9	66.0/73.5	75.5	56.4
Cluster-Former (#C=64)	53.3/63.3	67.0/74.2	76.3	56.7
Cluster-Former (#C=256)	53.6/63.5	67.5/74.5	76.3	56.7
Cluster-Former (#C=512)	**54.0/63.9**	**68.0/75.1**	**76.5**	57.1

Table 2: Results on Quasar-T, SearchQA test sets and NQ dev set. #C: number of clusters.

	Long Answer			Short Answer		
	F1	Precision	Recall	F1	Precision	Recall
BigBird-ETC-large (Zaheer et al., 2020)	77.8	77.5	**78.1**	57.9	63.7	53.0
RikiNet (Liu et al., 2020)	76.1	78.1	74.2	**61.3**	**67.6**	56.1
Cluster-Former (Ours)	**78.0**	**78.5**	77.5	60.9	62.1	**59.8**

Table 3: Results on Natural Questions (NQ) leaderboard (test set). We show two published results here from over 40 submissions. Our model achieves No.1 for long answer and No.4 for short answer.

incorrect answer predictions, and false negatives are incorrect "no answer" predictions. As the test set is hidden, we split 5% of the training set for validation, and use the original validation set for testing. We use the official tool from the dataset to evaluate our models. We also submit our best model to the leaderboard.

4.2 Implementation Details

All the models are trained on 8 Nvidia V100 GPUs. For clustering, we adopt "Yinyang kmeans" (Ding et al., 2015)[5] which takes less than 5 seconds for clustering in all our experiment settings. We set the memory size for clustering $M = 100,000$ in Algorithm 1. Based on our experiments, it makes little difference for memory banks with 50k and 100k, update cycles with 1 iteration or half iteration. We use cluster centroids that perform the best on the validation set for test set experiments. As the cluster-centroid is offline computed, the inference time is the same as the sliding-window-based method. We initialize our models with RoBERTa-large (Liu et al., 2019). As the number of position embeddings of RoBERTa is limited to 512, we cannot assign different position embeddings to all tokens. Instead, we assign the same position embeddings to each chunked sequence.

The majority of our model is made up of Sliding-Window Layers, as the local information is essential for QA tasks. We adopt the proposed Cluster-Former Layer in layers 15 and 20 to further capture long-range information. We set the sliding window size l to 256, stride m to 224, and change the number of clusters in $\{64, 256, 512\}$ to analyze its impact on the final performance. We prepend a special token to the beginning of all the given/retrieved paragraphs and directly concatenate all the paragraphs as the final context sequence. Due to memory constraints, we set the max length to be 5000

---

[5] https://github.com/src-d/kmcuda

	3	4	5	6
8	55.7/65.0	55.6/64.4	54.7/64.3	55.4/64.6
12	55.1/64.9	55.8/65.0	**56.1/65.4**	55.4/64.6
16	55.6/65.0	55.2/64.7	55.1/64.6	54.8/64.1
20	54.8/64.2	55.4/64.8	55.1/64.6	-

Table 4: Experiments on Quasar-T dev dataset. $a \in \{3, 4, 5, 6\}$ and $b \in \{8, 12, 16, 20\}$, if the layer number $l \ \% \ a == 0$ and $l >= b$, we set it as Cluster-Former Layer, otherwise Sliding Window Layer.

during training and 10000 during inference. During dataset finetuning, we use Adam (Kingma and Ba, 2015) to optimize the model. We set warm-up updates to 2,220, maximal updates to 22,200, learning rate to 5×10^{-5}, and batch size to 160. We tune the dropout rate from $\{0.1, 0.15, 0.2\}$ for all the methods including baselines and report the best results. The model converges in one day for all the QA datasets.

For Quasar-T and SearchQA, we predict the start and end positions of the answer. For Natural Question, we first identify whether the question has short/long answers or not based on the mean values of the first hidden state of all the chunked sequences, $\frac{1}{K}\sum_{k=1}^{K} \mathbf{H}_k^N[0]$, where K is the number of chunks and N is the number of layers. If answerable, we rank all the candidates for long answer selection, and predict the start and end positions of short answers. Our model submitted to Natural Question Leaderboard ensembled 3 models with 512 clusters, and only these models are firstly trained on SQuAD2.0 and then finetuned on Natural Question dataset.

4.3 Baselines

We compare our models with several strong baselines, including:

R3 (Wang et al., 2018) proposes to use reinforcement learning to jointly train passage ranker and reader. **DS-QA** (Lin et al., 2018) proposes to first use paragraph selection to filter the noisy data and then trained model on denoised data. **Multi-passage BERT** (Wang et al., 2019) proposes to filter the passages and then merge multiple useful passages into one sequence, which can be encoded by BERT. **DrQA** (Chen et al., 2017) makes use of attention mechanism across the question and the document for answer phrase extraction. **DecAtt and DocReader** (Kwiatkowski et al., 2019) is based on a pipeline approach that first uses a simpler model to select long answers and then a reading comprehension model to extract short answers from the long answers. **BERT$_{\text{joint}}$** (Alberti et al., 2019) jointly trains short and long answer extraction in a single model rather than using a pipeline approach. **BERT$_{\text{wwm}}$+SQuAD2** (Pan et al., 2019) makes use of multi-task learning to further boost performance. **RikiNet-RoBERTa** (Liu et al., 2020) proposes a dynamic paragraph dual-attention reader and a multi-level cascaded answer predictor. **BigBird-ETC** (Zaheer et al., 2020) makes use of a sparse attention mechanism to encode long sequences.

	Wikitext ppl	Enwik8 bpc
Sliding window	20.8	1.34
Sparse Attention	20.5	1.29
Locality-Sensitive Hashing	20.8	1.33
Cluster-Former (#C=64)	20.5	1.28
Cluster-Former (#C=256)	20.3	1.24
Cluster-Former (#C=512)	**20.2**	**1.22**

Table 5: Results on Language Modeling. #C: number of clusters; Wikitext: Wikitext-103.

We also re-implement several strong baselines which have not been applied to process long context in question answering tasks:

- **Sliding Window**: The original method is fully made up of Sliding-Window Layers and can only attend to local information. To make a fair comparison among different methods on long-range information collection, we replace several layers of this sliding window baseline with Sparse Attention, Locality-Sensitive Hashing, and Cluster-Former.

- **Sparse Attention** (Child et al., 2019): This method replaces several layers in the previous baseline by training a Transformer layer across sequences on pre-selected positions. We run this sparse Transformer on all the hidden states in the same position across sequences, so that the output of sparse Transformer can merge the information from different sequences.

- **Locality-Sensitive Hashing** (Kitaev et al., 2020): This method hashes hidden states into different buckets determined by randomly-initialized hashing vectors. A Transformer layer is then applied across buckets to build Sparse

Question	Where did the underground railroad start and finish ?
Context	The Underground Railroad by artist Charles T. Webber , 1893 Date Late 1700s - 1865 Location Northern United States with routes to Canada , Mexico ...
Special token	<s><s><s>Island island in the colonies city<s><s><s>With in the in .
Time	did start and finish 1893 Date 1700 1865 Location Participants Outcome Deaths 19 1763
Stopwords	the the , the , , , , to , , , , the American runaway slaves of free states the , , , it to , a the
Entity	Canada Mexico Canada is applied Florida Spanish Railroad Railroad Railroad
Positions	49, 50, 51, 52, 53, 54, 55, 115, 116, 168, 273, 394, ..., 6022, 6040, 6042, 6060, 6094

Table 6: An example from Natural Question dataset. The rows in the middle section show the corresponding words of the clustered hidden states, and the bottom row shows the positions of the clustered hidden states. "<s>" refers to start token of long answer candidate.

Attention across the whole sequence. Note that this method cannot be directly used for question answering without adding Sliding-Window layer, as our QA model is initialized by RoBERTa that only has 512 position embeddings.

4.4 Experimental Results

State-of-the-Art Results on QA Table 2 and 3 show that our proposed method outperforms several strong baselines, thanks to its ability to encode both local and global information. Cluster-Former with 512 clusters achieves new state-of-the-art results on Quasar-T, SearchQA and Natural Question (long answer).

Effect of Cluster-Former We also test the ability of Cluster-Former on modeling long-range dependencies. Note that Sparse Attention (Child et al., 2019) and Locality-Sensitive Hashing (Kitaev et al., 2020) have never been tested on question answering tasks with long context. For fair comparison, we set the layers 15 and 20 as either Sparse Attention, Locality-Sensitive Hashing or our Cluster-Former, and the left layers are Sliding Window layers.

As shown, Sparse Attention performs worse than our Cluster-Former. The loss may come from the noise introduced by pre-selected positions, the corresponding words of which may not be related. We set the number of hashing vectors in Locality-Sensitive Hashing (LSH) to 64, the same as the number of clusters in Cluster-Former. LSH outperforms the baseline slightly on QA and consistently underperforms our Cluster-Former (#C=64). Overall, our Cluster-Former performs the best.

Effect of Number of Cluster Centroids We also test the effect of different numbers of cluster centroids (C) on model performance. We observe that the model with 512 clusters works significantly better than the model with 64 clusters on most of the tasks. However, for Natural Questions Long Answer setting, the improvement is marginal. As we mainly rely on the hidden state of special tokens "<s>" for long answer selection, and the same tokens can be assigned into same chunk more easily even with a smaller number of clusters.

Selection of Cluster-Former Layers We also have an analysis on which layers are better used for Cluster-Former layer. As shown in Table 4, we conduct a hyper-parameter search. And find that it can get better performance with at least one Cluster-Former layers in the middle layer (8-16). The worst results come from only one Cluster-Former layer in the layer of 22 or 23.

Language Modeling Although we focus on QA tasks, to demonstrate the versatility of Cluster-Former, we conduct additional experiments on language modeling using the Wikitext-103 (Merity et al., 2017) and Enwik8 (Mahoney, 2011) benchmarks. All the models are trained from scratch. We set the number of layers to 16, with 8 heads per layer. Our Cluster-Former Layer is used in layers 11 and 15 as in QA models. We segment long input into short sequences of 3072 tokens, set sliding window size l to 256, and stride m to 128. SGD is used for optimizing the models. We set clip threshold of gradients to 0.1, warm-up updates to 16,000, maximal updates to 286,000, dropout rate to 0.3, learning rate to 0.1, and batch size to 16. The model will converge in 3 days for all the LM datasets. As shown in Table 5, Cluster-Former outperforms strong state-of-the-art baselines.

4.5 Qualitative Analysis

We perform qualitative analysis on how the hidden states are clustered, by visualizing the corresponding words and positions of the hidden states in Table 6. From the first row, we can see that the special tokens "<s>" tend to belong to the same cluster. Note that "<s>" is the start token of each long answer candidate, and its hidden state is used for final long answer selection. Therefore, Transformer on this cluster can compare across the candidates to make the final prediction.

We further observe that the same types of token are more likely to appear in the same cluster. For example, words from the second row to the forth row cover the topics of time, stopwords, and organization & geopolitical entities.

Finally, we randomly sample a cluster and list the positions of clustered hidden states in the last row of the table. We find that states in long distance, such as the 50-th and 6060-th states (over 6000 tokens apart), can be in one cluster, which demonstrates the ability of Cluster-Former in detecting long-range dependencies. Further, we observe that states tend to cluster in phrases. For example, we see consecutive positions such as "49, 50, 51, 52, 53, 54, 55", which likely results from the sliding-window encoding.

5 Conclusion

In this paper, we present Cluster-Former, a new method to encode global information for long sequences. We achieve new state of the art on three question answering datasets: Quasar-T, SearchQA, and Natural Questions. Further, we observe that a larger number of clusters in Cluster-Former can lead to better performance on question answering tasks. Cluster-Former is a generic approach, and we believe that it can benefit other NLP tasks that rely on long-range dependencies as well.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. 2020. Etc: Encoding long and structured data in transformers. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations (ICLR)*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Association for Computational Linguistics (ACL)*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Jared Davis, Tamas Sarlos, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Yufei Ding, Yue Zhao, Xipeng Shen, Madanlal Musuvathi, and Todd Mytkowicz. 2015. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In *International conference on machine learning (ICML)*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *European Chapter of the Association for Computational Linguistics (EACL)*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. *arXiv preprint arXiv:2006.16236*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics (TACL)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning (ICML)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Association for Computational Linguistics (ACL)*.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Association for Computational Linguistics (ACL)*.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020. Rikinet: Reading wikipedia pages for natural question answering. In *Association for Computational Linguistics (ACL)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Matt Mahoney. 2011. Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*.

Lin Pan, Rishav Chakravarti, Anthony Ferritto, Michael Glass, Alfio Gliozzo, Salim Roukos, Radu Florian, and Avirup Sil. 2019. Frustratingly easy natural question answering. *arXiv preprint arXiv:1909.05286*.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics (TACL)*.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention in transformer models. In *International Conference on Machine Learning (ICML)*.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse sinkhorn attention. In *International Conference on Machine Learning (ICML)*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. 2020. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems (NeurIPS)*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. In *Association for Computational Linguistics (ACL)*.

Minimally-Supervised Morphological Segmentation using Adaptor Grammars with Linguistic Priors

Ramy Eskander[1], **Cass Lowry**[2], **Sujay Khandagale**[1], **Francesca Callejas**[1],
Judith Klavans[3], **Maria Polinsky**[3], **Smaranda Muresan**[1]
[1]Columbia University, `{rnd2110,sk4746,ffc2108,smara}@columbia.edu`
[2]The Graduate Center, City University of New York, `clowry@gradcenter.cuny.edu`
[3]University of Maryland, `{jklavans,polinsky}@umd.edu`

Abstract

With the increasing interest in low-resource languages, unsupervised morphological segmentation has become an active area of research, where approaches based on Adaptor Grammars achieve state-of-the-art results. We demonstrate the power of harnessing linguistic knowledge as priors within Adaptor Grammars in a minimally-supervised learning fashion. We introduce two types of priors: 1) grammar definition, where we design language-specific grammars; and 2) linguist-provided affixes, collected by an expert in the language and seeded into the grammars. We use Japanese and Georgian as respective case studies for the two types of priors and introduce new datasets for these languages, with gold morphological segmentation for evaluation. We show that the use of priors results in error reductions of 8.9 % and 34.2 %, respectively, over the equivalent state-of-the-art unsupervised system.

1. Introduction

Morphological segmentation is an essential subtask in many natural language processing (NLP) applications, especially in the case of morphologically complex languages. With the need to develop NLP tools for low-resource languages, unsupervised morphological segmentation has been receiving increasing interest over the last two decades (Goldsmith, 2001; Creutz and Lagus, 2007a; Poon et al., 2009; Sirts and Goldwater, 2013; Botha and Blunsom, 2013; Narasimhan et al., 2014; Eskander et al., 2016, 2018, 2019).

In this work, we show how linguistic priors effectively boost morphological-segmentation performance in a minimally-supervised manner that does not require segmented words for training. We integrate our priors within Adaptor Grammars (Johnson et al., 2007), a type of nonparametric Bayesian models that generalize Probabilistic Context-Free Grammars (PCFGs). Adaptor Grammars have proved successful for unsupervised morphological segmentation, achieving state-of-the-art results across a variety of typologically diverse languages (Eskander et al., 2020).

We introduce two types of linguistic priors: 1) grammar definition, where we design a language-specific grammar that is tailored for the language of interest by modeling specific morphological phenomena, and 2) linguist-provided affixes, where an expert in the underlying language compiles a list of carefully selected affixes and seeds it into the grammars prior to training the segmentation model. We use Japanese and Georgian as case studies for priors 1 and 2, respectively. As our goal is to develop a robust approach that benefits low-resource and/or endangered languages of high morphological complexity, we use Japanese and Georgian in a low-resource setting where we do not have access to morphologically segmented data for training but have access to linguistic information such as word structure and affixes.

We show that using linguistic priors in a minimally-supervised setting leads to a significant improvement in performance over the equivalent state-of-the-art unsupervised system. We also present two morphologically segmented datasets for Japanese and Georgian that we use as our gold standard and that can be utilized in other morphology tasks.[1]

2. Linguistic Priors

We utilize MorphAGram (Eskander et al., 2020)[2], an open-source morphological-segmentation framework that is based on Adaptor Grammars (AGs) (Johnson et al., 2007). AGs have proved successful for unsupervised and

---

[1] The training and evaluation datasets, linguistic priors and models for both Japanese and Georgian are available at https://github.com/rnd2110/MorphAGram/data.

[2] https://github.com/rnd2110/MorphAGram

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3969–3974
August 1–6, 2021.

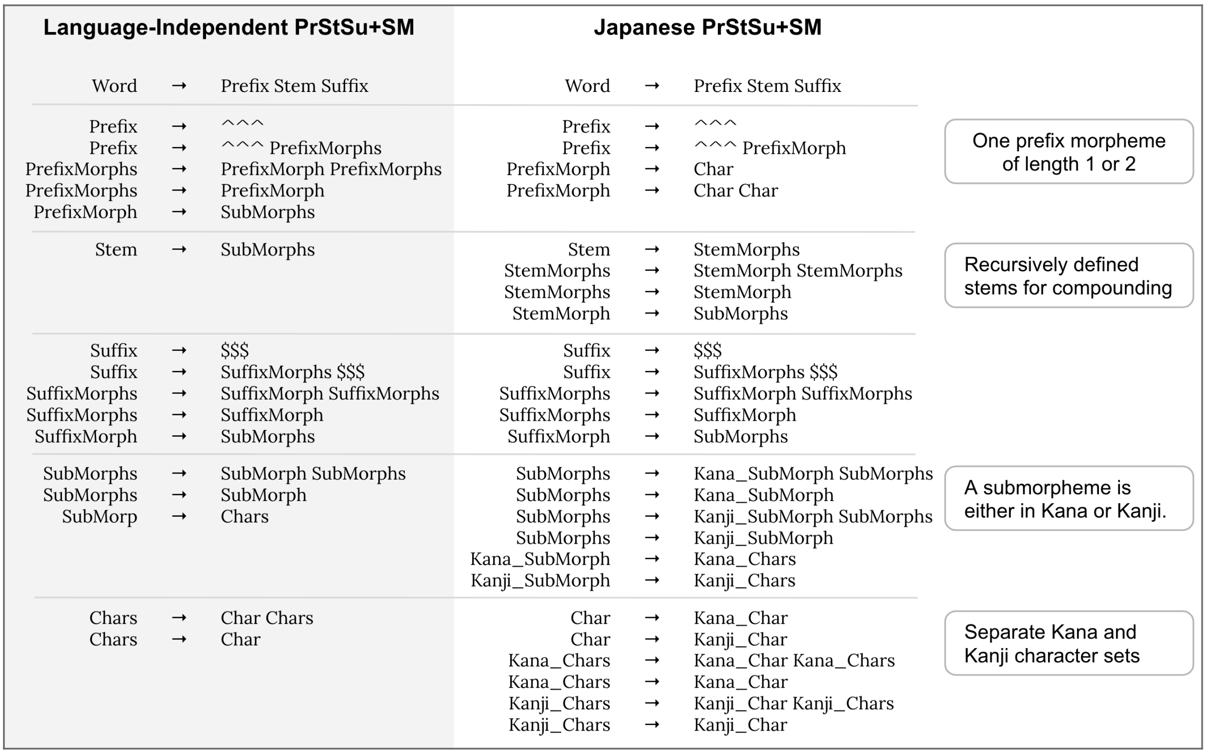

Language-Independent PrStSu+SM

Word	→	Prefix Stem Suffix
Prefix	→	^^^
Prefix	→	^^^ PrefixMorphs
PrefixMorphs	→	PrefixMorph PrefixMorphs
PrefixMorphs	→	PrefixMorph
PrefixMorph	→	SubMorphs
Stem	→	SubMorphs
Suffix	→	$$$
Suffix	→	SuffixMorphs $$$
SuffixMorphs	→	SuffixMorph SuffixMorphs
SuffixMorphs	→	SuffixMorph
SuffixMorph	→	SubMorphs
SubMorphs	→	SubMorph SubMorphs
SubMorphs	→	SubMorph
SubMorp	→	Chars
Chars	→	Char Chars
Chars	→	Char

Japanese PrStSu+SM

Word	→	Prefix Stem Suffix	
Prefix	→	^^^	One prefix morpheme of length 1 or 2
Prefix	→	^^^ PrefixMorph	
PrefixMorph	→	Char	
PrefixMorph	→	Char Char	
Stem	→	StemMorphs	Recursively defined stems for compounding
StemMorphs	→	StemMorph StemMorphs	
StemMorphs	→	StemMorph	
StemMorph	→	SubMorphs	
Suffix	→	$$$	
Suffix	→	SuffixMorphs $$$	
SuffixMorphs	→	SuffixMorph SuffixMorphs	
SuffixMorphs	→	SuffixMorph	
SuffixMorph	→	SubMorphs	
SubMorphs	→	Kana_SubMorph SubMorphs	A submorpheme is either in Kana or Kanji.
SubMorphs	→	Kana_SubMorph	
SubMorphs	→	Kanji_SubMorph SubMorphs	
SubMorphs	→	Kanji_SubMorph	
Kana_SubMorph	→	Kana_Chars	
Kanji_SubMorph	→	Kanji_Chars	
Char	→	Kana_Char	Separate Kana and Kanji character sets
Char	→	Kanji_Char	
Kana_Chars	→	Kana_Char Kana_Chars	
Kana_Chars	→	Kana_Char	
Kanji_Chars	→	Kanji_Char Kanji_Chars	
Kanji_Chars	→	Kanji_Char	

Figure 1: Language-independent PrStSu+SM grammar (left side) vs. its Japanese cognate (right side)

minimally-supervised morphological segmentation, outperforming the competing discriminative models (Sirts and Goldwater, 2013; Eskander et al., 2019, 2020).

Adaptor Grammars are non-parametric Bayesian models that are composed of two main components: 1) a Probabilistic Context-Free Grammar (PCFG) whose definition relies on the underlying task (in the case of morphological segmentation, a PCFG models word structure); and 2) an adaptor that is based on the Pitman-Yor process (Pitman, 1995). The adaptor keeps the posterior probability of a subtree proportional to the number of times that subtree is utilized to parse the input data and manages the caching of the subtrees. The learning process is Markov Chain Monte Carlo sampling (MCMC) (Andrieu et al., 2003) that does the inference of the PCFG probabilities and the hyperparameters of the model.

Eskander et al. (2016) define a set of language-independent grammars and three learning settings for Adaptor Grammars: 1) *Standard*, fully unsupervised; 2) *Scholar-Seeded*, minimally-supervised by manually seeding affixes into the grammar prior to training the segmentation model, and 3) *Cascaded*, fully unsupervised by approximating the *Scholar-Seeded* setting using automatically generated affixes from an initial round of learning. We next present two ways of including linguistic priors in Adaptor Grammars: 1) defining a language-specific grammar; and 2) using linguist-provided affixes in the Scholar-Seeded learning setup.

2.1. Linguistic Priors as Grammar Definition

Eskander et al. (2016) define language-independent grammars that model the word as a sequence of generic morphemes or as a sequence of prefixes, stem and suffixes. We consider their *PrStSu+SM* grammar in the current study as it is the grammar that performed best on average across different languages. This language-independent definition of the grammar is depicted on the left side of Figure 1, where the word is modeled as a prefix *Pr*, a stem *St* and a suffix *Su*, and both the prefix and suffix are recursively defined in order to model compounding in affixes, while a morpheme is composed of smaller units, submorphemes *SM*, representing sequences of characters.

While this grammar is intended to be generic and to describe word structure in any language, we hypothesize that a definition that imposes language-specific constraints would be more efficient. Therefore, we define a grammar for Japanese, where we use characteristics that are specific to Japanese word structure as language priors. Our tailored grammar definition for Japanese is shown on the right side of Figure 1, where we impose the following specifications:

- A word has a maximum of one one-character or two-character prefix morphemes.
- A stem is recursively defined as a sequence of morphemes in order to allow for stem compounding.
- Characters are separated into two groups, Kana (Japanese syllabaries) and Kanji (adapted Chinese characters).
- A submorpheme represents a sequence of characters that is either in Kana or Kanji.

2.2. Linguistic Priors as Linguist-Provided Affixes

Similar to the *Scholar-Seeded* setting, we compile a list of affixes and seed it into the grammar trees before learning the segmentation model. However, unlike Eskander et al. (2016), where the affixes are collected from online resources by someone who may have never studied the language of interest, in this study we use affixes that are carefully compiled by an expert linguist who specializes in Georgian, resulting in more accurate linguistic priors. With that goal in mind, a total of 119 affixes are collected from the leading reference grammar book (Aronson, 1990).

3. Evaluation Data

We annotate two datasets with morphological segmentation that we use as the gold standard to evaluate our segmentation models for Japanese and Georgian. Both datasets are composed of 1,000 words that are randomly sampled from the most frequent 50,000 words in Wikipedia and segmented into their basic morphemes[3], similar to the data of the Morpho Challenge shared task [4]. Table 1 lists segmentation examples for both languages.

The Japanese gold segmentation was created by a native-speaker linguist. For Georgian, which has highly complex morphology, we started with the gold-standard dataset of 1000 words introduced by Eskander et al. (2020), which was built by an untrained native speaker and contained only one possible segmentation per word. An expert in Georgian then corrected 193 examples in the data and further annotated 116 words for two possible alternative segmentations. In addition, the expert coded each word based on its syntactic category: verbs (359), nouns (475), numerals (44) and other (122).

Japanese Word	Segmentation
いました	い + ま + した
勉強して	勉強 + し + て
始められません	始め + られ + ま + せん

Georgian Category	Word	Segmentation
Verb	იქნებაო	ი + ქნ + ებ + ა + ო იქნებ + ა + ო
Noun	თვითფრინავი	თვი + თ + ფრინ + ავ + ი თვითფრინავ + ი
Numeral	თოთხმეტი	თ + ოთხ + მეტ + ი თოთხმეტ + ი
Other	ვისი	ვი + ს + ი ვის + ი

Table 1: Japanese and Georgian segmentation examples

4. Evaluation and Results

4.1. Experimental Setup

We evaluate our morphological-segmentation models for Japanese in the *Standard (STD)* and *Cascaded (CAS)*[5] settings, both with generic and language-specific *(LS)* grammar definitions. For Georgian, we evaluate our morphological-segmentation models in the *Standard (STD)*, *Cascaded (CAS)* and *Scholar-Seeded (SS)* settings, in addition to the proposed *Scholar-Seeded* setting with linguist-provided affixes (*SS-Ling*).

We perform the evaluation in a transductive manner, where the unsegmented words in the gold standard are part of the training sets; this is common in evaluating unsupervised and minimally-supervised morphological segmentation (Poon et al., 2009; Sirts and Goldwater, 2013; Narasimhan et al., 2014; Eskander et al., 2016, 2019, 2020). For the metrics, we use Boundary Precision and Recall (BPR) and EMMA-2 (Virpioja et al., 2011). BPR is the classical metric for evaluating morphological segmentation; it compares the boundaries in the proposed segmentation to those in the reference. EMMA-2

---

[3] The Georgian dataset contains five non-words and three phonetic spellings of English character names.

[4] http://morpho.aalto.fi/events/morphochallenge/

[5] For the Cascaded setup, we use the high-precision grammar *PrStSu2a+SM* defined by Eskander et al. (2016) as the base grammar.

Language	Setting	BPR			EMMA-2		
		Prec.	Recall	F1-Score	Prec.	Recall	F1-Score
Japanese	*Morfessor*	81.4	77.2	79.3	91.5	80.1	85.4
	AG STD	81.7	77.4	79.5	91.0	81.8	86.1
	AG CAS	80.9	78.2	79.5	90.8	82.0	86.2
	AG STD-LS	83.5	79.3	**81.3**	92.0	82.5	**87.0**
	AG CAS-LS	82.8	79.3	81.0	91.1	82.6	86.6
Georgian	*Morfessor*	79.2	54.6	64.6	88.5	56.1	68.7
	AG STD	81.8	69.0	74.9	87.8	65.5	75.0
	AG CAS	83.5	70.4	76.4	88.6	67.2	76.4
	AG SS	84.5	69.1	76.0	89.3	65.2	75.4
	AG SS-Ling	84.6	82.4	**83.5**	87.6	78.2	**82.6**

Table 2: Morphological-segmentation performance for Japanese and Georgian using the BPR and EMMA-2 metrics. The best F1-score per language-metric pair is in **bold**. *AG* = *Adaptor Grammars*. *STD* = *Standard*. *CAS* = *Cascaded*. *STD-LS* = *Standard* with a language-specific grammar. *CAS-LS* = *Cascaded* with a language-specific grammar. *SS* = *Scholar-Seeded*. *SS-Ling* = *Scholar-Seeded* with linguist-provided affixes

Category	BPR						EMMA-2					
	AG SS			AG SS-Ling			AG SS			AG SS-Ling		
	Prec.	Recall	F1-Score	Prec.	Recall	F1-Score	Prec.	Recall	F1-Score	Prec.	Recall	F1-Score
Noun	74.4	79.6	76.9	74.6	90.4	81.8	87.4	78.7	82.8	84.4	86.8	85.6
Verb	95.8	50.5	66.1	96.6	68.9	80.4	96.4	49.2	65.1	96.1	69.6	80.7
Numeral	93.9	74.1	82.8	87.9	84.8	86.3	87.3	65.5	74.8	81.6	66.0	73.0
Other	87.0	81.6	84.2	86.7	90.3	88.4	92.4	79.0	85.2	92.0	85.8	88.8

Table 3: Category-wise morphological-segmentation performance for Georgian using the BPR and EMMA-2 metrics. *AG* = *Adaptor Grammars*. *SS* = *Scholar-Seeded*. *SS-Ling* = *Scholar-Seeded* with linguist-provided affixes.

is based on matching the morphemes in the proposed segmentation to those in the reference in a many-to-one assignment setup.

We evaluate our system versus two state-of-the-art unsupervised baselines: MorphAGram without the use of linguistic priors and Morfessor (Virpioja et al., 2013) [6]. Morfessor is a commonly-used framework for unsupervised morphological segmentation. It is based on an HMM model that relies on the Minimum Description Length (MDL) concept for deriving the optimal segmentation (Creutz and Lagus, 2007b). Since our approach does not assume access to manually annotated segmentation, it is not directly comparable to semi-supervised approaches that rely on such annotations (Ruokolainen et al., 2014; Kann et al., 2018). Finally, we report all the Adaptor-Grammar results as the average over three runs of different randomization parameters.

4.2. System Performance

Table 2 reports the overall performance of our models for both Japanese and Georgian, while Table 3 shows the results per part-of-speech category for Georgian.

For Japanese, the use of a language-specific grammar definition improves both precision and recall, resulting in BPR F1-score error reductions of 8.9 % and 7.1 % over the generic *Standard* and *Cascaded* settings, respectively, and a BPR F1-score error reduction of 9.8 % over Morfessor.

For Georgian, the use of linguist-provided seeded affixes improves both precision and recall, where the recall significantly increases by absolute 13.3 % over using an affix list of lower quality. In addition, the proposed linguistic priors result in BPR F1-score error reductions of 34.2 %, 30.0 % and 31.1 % over the *Standard*, *Cascaded* and regular *Scholar-Seeded* settings, respectively, and a BPR F1-score error reduction of 53.3 % over Morfessor. Analysing results per category, verbs and nouns receive the biggest F1-score improvements of absolute 14.3 % and 4.9 %, respectively, with the use of linguist-provided affixes.

A similar pattern of results is found with EMMA-2. Finally, all the improvements due to the use of linguistic priors are statistically significant ($P < 0.01$) on both metrics.

4.3. Error Analysis

Table 4 lists some examples of correctly and incorrectly segmented words by our Japanese and

---

[6] https://morfessor.readthedocs.io/en/latest/

	Word	Gold Segmentation	*STD* Segmentation	*STD-LS* Segmentation
Japanese	お電話	お + 電話	お + 電話	お + 電話
	ご親族	ご + 親 + 族	ご + 親 + 族	ご + 親 + 族
	登られませ	登 + られ + ま + せん	登ら + れ + ません	登 + られ + ません
	比べなかったら	比べ + な + かった + ら	比べ + なかった + ら	比べ + なかった + ら

	Word	Gold Segmentation	*STD* Segmentation	*SS* Segmentation	*SS-Ling* Segmentation
Georgian	ლურჯი	ლურჯ + ი	ლურჯ + ი	ლურჯ + ი	ლურჯ + ი
	რვის	რვ + ის	რ + ვის	რვ + ის	რვ + ის
	გამოვა	გა მო ვ ა	გამო ვ ა	გა მო ვ ა	გამოვ ა
	შაური	შაურ + ი	შა + ური	შა + ურ + ი	შაურ + ი

Table 4: Examples of output segmentations for Japanese and Georgian. *STD* = *Standard*. *STD-LS* = *Standard* with a language-specific grammar. *SS* = *Scholar-Seeded*. *SS-Ling* = *Scholar-Seeded* with linguist-provided affixes. Incorrect morphemes are marked in red.

Georgian segmentation models. We discuss the most prominent observations below.

Japanese: Both the *STD* and *STD-LS* models perform well on prefix segmentation, achieving F1-scores of more than 90 % in the detection of several one-character prefixes, such as お and ご. However, *STD-LS* outperforms its language-independent counterpart in the detection of stems, where compounding is explicitly modeled. For instance, *STD* and *STD-LS* achieve F1-scores of 15.8 % and 98.6 %, respectively, in the detection of the common stem られ (*be*). On the other hand, when either model consistently fails to detect a specific morpheme, the other model fails as well. For example, neither model can detect the morphemes せん and かった.

Georgian: *SS-Ling* outperforms both *STD* and *SS* at discovering the top most frequent one-letter morphemes, such as ი, ა, ს, ე, მ, ო and ვ, achieving an average F1-score of 76.0 %, compared to 57.7 % and 57.3 % by *STD* and *SS*, respectively. In addition, *SS* and *STD* suffer lower precision as they tend to oversegment the morphemes represented by a single letter. Similarly, *SS-Ling* can recognize the most frequent two-letter morphemes, namely ებ and და, with absolute increases in precision of 59.0 % and 62.0 % over *STD* and *SS*, respectively; both morphemes are explicitly seeded into the *SS-Ling* grammar prior to training the model.

5. Conclusion and Future Work

We proposed two types of linguistic priors for minimally-supervised morphological segmentation using Adaptor Grammars. The first prior is in the form of defining a language-specific grammar, while the second relies on compiling a list of linguist-provided affixes and seeding it into the grammars. Our approaches result in error reductions of 8.9 %, for Japanese, and 34.2 %, for Georgian, as compared to the state-of-the-art system. In future work, we plan to explore the use of linguistic priors that apply to a group of morphologically similar low-resource languages.

Acknowledgements

This research is based upon work supported by the Intelligence Advanced Research Projects Activity (IARPA), (contract #FA8650-17-C-9117) and the National Science Foundation (awards #1941742 and #1941733). The views and conclusions herein are those of the authors and should not be interpreted as necessarily representing official policies, expressed or implied, of ODNI, IARPA, NSF or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Ethical Considerations

The Japanese annotations were done by a linguist with appropriate compensation; we thus have ownership of the Japanese dataset for open distribution. We have been granted the rights to modify and distribute the dataset for Georgian by Eskander et al. (2020), where the annotations were done in-house by a paid linguist. Finally, the quality of the annotations was examined, both manually and empirically.

References

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. 2003. An Introduction to MCMC for Machine Learning. *Machine learning*, 50(1-2):5–43.

Howard I Aronson. 1990. *Georgian: A Reading Grammar, Corrected Edition*. Slavica Publishers.

Jan A Botha and Phil Blunsom. 2013. Adaptor grammars for learning non- concatenative morphology. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2007a. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):1–34.

Mathias Creutz and Krista Lagus. 2007b. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3:1–3:34.

Ramy Eskander, Francesca Callejas, Elizabeth Nichols, Judith L Klavans, and Smaranda Muresan. 2020. Morphagram, evaluation and framework for unsupervised morphological segmentation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 7112–7122.

Ramy Eskander, Judith L Klavans, and Smaranda Muresan. 2019. Unsupervised morphological segmentation for low-resource polysynthetic languages. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–195.

Ramy Eskander, Owen Rambow, and Tianchun Yang. 2016. Extending the use of adaptor grammars for unsupervised morphological segmentation of unseen languages. In *Proceedings of he Twenty-Sixth International Conference on Computational Linguistics (COLING)*, Osaka, Japan.

Ramy Eskander, Owen Rambow, and Tianchun Yang. 2018. Automatically tailoring unsupervised morphological segmentation to the language. In *Proceedings of the Fifteenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Brussels, Belgium.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648.

Katharina Kann, Manuel Mager, Ivan Meza-Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. *arXiv preprint arXiv:1804.06024*.

Karthik Narasimhan, Damianos Karakos, Richard M. Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *EMNLP*.

Jim Pitman. 1995. Exchangeable and Partially Exchangeable Random partitions. *Probability theory and related fields*, 102(2):145–158.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado. Association for Computational Linguistics.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 84–89.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1(May):231–242.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Sami Virpioja, Ville T. Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.

Multi-Task Learning and Adapted Knowledge Models for Emotion-Cause Extraction

Elsbeth Turcan[1*] **Shuai Wang**[3] **Rishita Anubhai**[3]
Kasturi Bhattacharjee[3] **Yaser Al-Onaizan**[3] **Smaranda Muresan**[2,3]
[1]Department of Computer Science, Columbia University
[2]Data Science Institute, Columbia University
[3]Amazon AI
`{eturcan, smara}@cs.columbia.edu`
`{wshui, ranubhai, kastb, onaizan}@amazon.com`

Abstract

Detecting what emotions are expressed in text is a well-studied problem in natural language processing. However, research on finer-grained emotion analysis such as what causes an emotion is still in its infancy. We present solutions that tackle both emotion recognition and emotion cause detection in a joint fashion. Considering that common-sense knowledge plays an important role in understanding implicitly expressed emotions and the reasons for those emotions, we propose novel methods that combine common-sense knowledge via adapted knowledge models with multi-task learning to perform joint emotion classification and emotion cause tagging. We show performance improvement on both tasks when including common-sense reasoning and a multi-task framework. We provide a thorough analysis to gain insights into model performance.

1 Introduction

Utterance and document level emotion recognition has received significant attention from the research community (Mohammad et al., 2018; Poria et al., 2020a). Given the utterance *Sudan protests: Outrage as troops open fire on protestors* an emotion recognition system will be able to detect that *anger* is the main expressed emotion, signaled by the word "outrage". However, the semantic information associated with expressions of emotion, such as the cause (the thing that triggers the emotion) or the target (the thing toward which the emotion is directed), is important to provide a finer-grained understanding of the text that might be needed in real-world applications. In the above utterance, the cause of the anger emotion is the event "troops open fire on protestors", while the target is the entity "troops" (see Figure 1) .

Research on finer-grained emotion analysis such as detecting the cause for an emotion expressed in text is in its infancy. Most work on emotion-cause detection has utilized a Chinese dataset where the cause is always syntactically realized as a clause and thus was modeled as a classification task (Gui et al., 2016). However, recently Bostan et al. (2020) and Oberländer and Klinger (2020) argued that in English, an emotion cause can be expressed syntactically as a clause (*as troops open fire on protestors*), noun phrase (*1,000 non-perishable food donations*) or verb phrase (*jumped into an ice-cold river*), and thus we follow their approach of framing emotion cause detection as a sequence tagging task.

We propose several ways in which to approach the tasks of emotion recognition and emotion cause tagging. First, these two tasks should not be independent; because the cause is the trigger for the emotion, knowledge about what the cause is should narrow down what emotion may be expressed, and vice versa. Therefore, we present a multi-task learning framework to model them jointly. Second, considering that common-sense knowledge plays an important role in understanding implicitly expressed emotions and the reasons for those emotions, we explore the use of common-sense knowledge via adapted knowledge models (COMET, Bosselut et al. (2019)) for both tasks. A key feature of our approach is to combine these adapted knowledge models (i.e., COMET), which are specifically trained to use and express common-sense knowledge, with pre-trained language models such as BERT, (Devlin et al., 2019).

Our primary contributions are three-fold: (i) an under-studied formulation of the emotion cause detection problem as a sequence tagging problem; (ii) a set of models that perform the emotion classification and emotion cause tagging tasks jointly while

---

[*] Work done during an internship with Amazon AI.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3975–3989
August 1–6, 2021. ©2021 Association for Computational Linguistics

using common-sense knowledge (subsection 4.2) with improved performance (section 6); and (iii) analysis to gain insight into both model performance and the GoodNewsEveryone dataset that we use (Bostan et al., 2020) (section 7).

2 Related Work

Emotion detection is a widely studied subfield of natural language processing (Mohammad et al., 2018; Poria et al., 2020a), and has been applied to a variety of text genres such as fictional stories (Alm et al., 2005), news headlines (Strapparava and Mihalcea, 2010), and social media, especially microblogs such as Twitter (Abdul-Mageed and Ungar, 2017; Kiritchenko et al., 2014; Rosenthal et al., 2019; Mohammad et al., 2018). Earlier work, including some of the above, focused on feature-based machine learning models that could leverage emotion lexicons (Mohammad and Turney, 2013)), while recent work explores deep learning models (e.g., Bi-LSTM and BERT) and multi-task learning (Xu et al., 2018; Demszky et al., 2020).

However, comparatively few researchers have looked at the semantic roles related to emotion such as the cause, the target or the experiencer, with few exceptions for Chinese (Gui et al., 2016; Chen et al., 2018; Xia and Ding, 2019; Xia et al., 2019; Fan et al., 2020; Wei et al., 2020; Ding et al., 2020), English (Mohammad et al., 2014; Ghazi et al., 2015; Kim and Klinger, 2018; Bostan et al., 2020; Oberländer et al., 2020; Oberländer and Klinger, 2020) and Italian (Russo et al., 2011). We highlight some of these works here and draw connection to our work. Most recent work on emotion-cause detection has been carried out on a Chinese dataset compiled by Gui et al. (2016). This dataset characterizes the emotion and cause detection problems as clause-level pair extraction problem – i.e., of all the clauses in the input, one is selected to contain the expression of an emotion, and one or more (usually one) are selected to contain the cause of that emotion. Many publications have used this corpus to develop novel and effective model architectures for the clause-level classification problem (Chen et al., 2018; Xia and Ding, 2019; Xia et al., 2019; Fan et al., 2020; Wei et al., 2020; Ding et al., 2020). The key difference between this work and ours is that we perform cause detection as a sequence-tagging problem: the cause may appear anywhere in the input, and may be expressed as any grammatical construction (a noun phrase, a verb phrase, or a clause). Moreover, we use common sense knowledge for both tasks (emotion and cause tagging), through the use of adapted language models such as COMET.

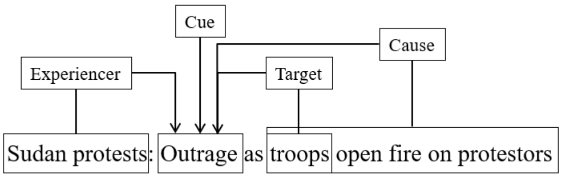

Figure 1: An example of the semantic roles annotated by Bostan et al. (2020)

For English, several datasets have been introduced (Mohammad et al., 2014; Kim and Klinger, 2018; Ghazi et al., 2015; Bostan et al., 2020; Poria et al., 2020b), and emotion cause detection has been tackled either as a classification problem (Mohammad et al., 2014), or as a sequence tagging or span detection problem (Kim and Klinger, 2018; Ghazi et al., 2015; Oberländer and Klinger, 2020; Poria et al., 2020b). We particularly note the work of Oberländer and Klinger (2020), who argue for our problem formulation of cause detection as sequence tagging rather than as a classification task supported by empirical evidence on several datasets including the GoodNewsEveryone dataset (Bostan et al., 2020) we use in this paper. One contribution we bring compared to these models is that we formulate a multi-task learning framework to jointly learn the emotion and the cause span. Another contribution is the use of common-sense knowledge through the use of adapted knowledge models such as COMET (both in the single models and the multi-task models). Ghosal et al. (2020) have very recently shown the usefulness of common-sense reasoning to the task of conversational emotion detection.

3 Data

For our experiments, we use the GoodNewsEveryone corpus (Bostan et al., 2020), which contains 5,000 news headlines labeled with emotions and semantic roles such as the target, experiencer, and cause of the emotion, as shown in Figure 1.[1] We focus on the emotion detection and cause tagging tasks in this work. To our knowledge, GoodNewsEveryone is the largest English dataset labeled for

---

[1] While the dataset labels both the most dominant emotion expressed in text and the reader's emotion, for this paper we only focus on the former.

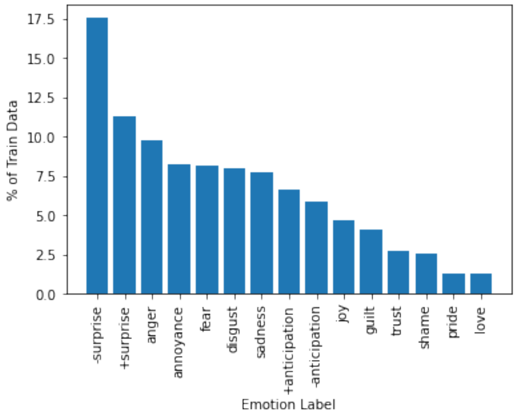

Figure 2: Distribution of adjudicated emotion labels in the GoodNewsEveryone train data, as a percentage of the data points. "Positive" and "Negative" are abbreviated as + and -.

both of these tasks.

In our experiments, we limit ourselves to the data points for which a cause span was annotated (4,798). We also note that this dataset uses a 15-way emotion classification scheme, an extended set including the eight basic Plutchik emotions as well as additional emotions like *shame* and *optimism*. While a more fine-grained label set is useful for capturing subtle nuances of emotion, many external resources focus on a smaller set of emotions. We also note that the label distribution of this dataset heavily favors the more basic emotions, as shown in Figure 2. Therefore, for our work, we choose to limit ourselves to the six Ekman emotions (*anger*, *fear*, *disgust*, *joy*, *surprise*, and *sadness*). We also choose to keep *positive surprise* and *negative surprise* separated, to avoid severely unbalancing the label distribution for our experiments. We randomly split the remaining data (2,503 data points) into 80% train, 10% development, and 10% test.

4 Models

An important feature showcased by the GoodNewsEveryone dataset is that causes of emotions can be expressed through different syntactic constituents such as clauses, verb phrases, or noun-phrases. Thus, we approach the cause detection problem as a sequence tagging problem using the IOB scheme (Ramshaw and Marcus, 1995): $\mathcal{C} = \{\text{I-cause}, \text{O}, \text{B-cause}\}$. Our approach is supported by very recent results by Oberländer and Klinger (2020) and Yuan et al. (2020) who show that modeling emotion cause detection as a sequence tagging problem is better suited than a clause classification problem, although not much current work has yet adopted this formulation. We tackle the emotion detection task as a seven-way classification task with $\mathcal{E} = \{\text{anger}, \text{disgust}, \text{fear}, \text{joy}, \text{sadness}, \text{negative surprise}, \text{positive surprise}\}$.

4.1 Single-Task Models

As a baseline, we train single-task models for each of emotion classification and cause span tagging. We use a pre-trained BERT language model[2] (Devlin et al., 2019), which we fine-tune on our data, as the basis of this model. Our preprocessing strategy for all of our models consists of the pretrained BERT vocabulary and WordPiece tokenizer[3] (Wu et al., 2016) from Huggingface (Wolf et al., 2020). Therefore, for a sequence of n WordPiece tokens, our input to the BERT model is a sequence of $n + 2$ tokens, $X = [\text{[CLS]}, x_1, x_2, ...x_n, \text{[SEP]}]$, where each x_i is from a finite WordPiece vocabulary and [CLS] and [SEP] are BERT's begin and end tokens. Passing X through BERT yields a sequence of vector hidden states $H = [h_{[CLS]}, h_1, h_2, ..., h_n, h_{[SEP]}]$ with dimension $d_{BERT} = 768$. For emotion classification, we pool these hidden states and allow hyperparameter tuning to select the best type: selecting the [CLS] token ($h_f = h_{[CLS]}$), mean pooling ($h_f = \frac{\sum_{i=1}^{n} h_i}{n}$), max pooling ($h_{f,j} = \max h_{i,j}$), or attention as formulated by Bahdanau et al. (2015):

$$h_f = \sum_{i=1}^{n} \alpha_i h_i \tag{1}$$

where $\alpha_i = \frac{\exp(W_a h_i + b_a)}{\sum_{j=1}^{n} \exp(W_a h_j + b_a)}$ for trainable weights $W_a \in \mathbb{R}^{1 \times d_{BERT}}$ and $b_a \in \mathbb{R}^1$. Then, the final distribution of emotion scores is calculated by a single dense layer and a softmax:

$$e = \text{softmax}(W_e h_f + b_e) \tag{2}$$

with $e \in \mathbb{R}^{|\mathcal{E}|}$ and for trainable parameters $W_e \in \mathbb{R}^{|\mathcal{E}| \times d_{BERT}}$ and $b_e \in \mathbb{R}^{|\mathcal{E}|}$. For cause tagging, a tag probability distribution is calculated directly on each hidden state:

$$c_i = \text{softmax}(W_c h_i + b_c) \tag{3}$$

---

[2] We use BERT-BASE-UNCASED. We experimented with BERT-BASE-CASED, but it underperformed as the headlines incorporated into GoodNewsEveryone come from different news sources and have different capitalization styles.

[3] In the tagging setting, we ignore all tags predicted for subword tokens and use only the tag of the first subword.

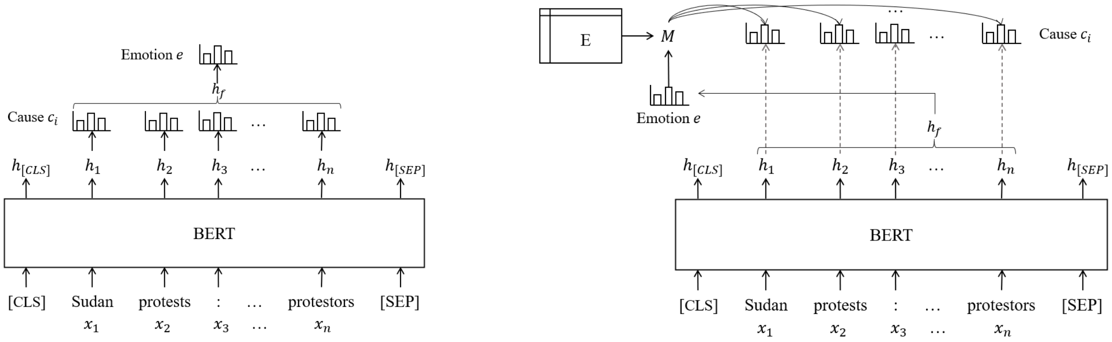

(a) The Multi$_{C\to E}$ model.

(b) The Multi$_{E\to C}$ model.

Figure 3: Our multi-task models.

with $c_i \in \mathbb{R}^{|\mathcal{C}|}$ and for trainable parameters $W_c \in \mathbb{R}^{|\mathcal{C}|\times d_{BERT}}$ and $b_c \in \mathbb{R}^{|\mathcal{C}|}$. We refer to both of these single-task models as BERT; if the task is not clear from the context, we will refer to the emotion detection model as BERT$_E$ and the cause tagging model as BERT$_C$. Our training loss for emotion classification as well as emotion cause tagging is the mean negative log-likelihood (NLL) loss per minibatch of size b:

$$\text{NLL}_{\text{emo}} = -\frac{1}{b}\sum_{j}\sum_{k} y_{jk} \log e_{jk} \tag{4}$$

$$\text{NLL}_{\text{cause}} = -\frac{1}{b}\sum_{i}\sum_{j}\sum_{k} y_{ijk} \log c_{ijk} \tag{5}$$

where j is the index of the sentence in the minibatch, k is the index of the label being considered (emotion labels for NLL$_{\text{emo}}$ and IOB tags for NLL$_{\text{cause}}$), i is the index of the i^{th} token in the j^{th} sentence in the minibatch, $y_{jk} \in \{0, 1\}$ is the gold probability of the k^{th} emotion label for the j^{th} sentence, $y_{ijk} \in \{0, 1\}$ is the gold probability of the k^{th} cause tag for the i^{th} token in the j^{th} sentence, and e_{jk} and c_{ijk} are the output probabilities of the k^{th} emotion label and of the k^{th} cause label for the i^{th} token, both for the j^{th} sentence.

4.2 Multi-Task Models

Our hypothesis is that the emotion detection and cause tagging tasks are closely related and can inform each other; therefore we propose three multi-task learning models to test this hypothesis. For all multi-task models, we use the same base architecture (BERT) as the single models. Additionally, for these models, we combine the losses of both tasks and weight them with a tunable lambda parameter: $\lambda\text{NLL}_{\text{emo}} + (1-\lambda)\text{NLL}_{\text{cause}}$, using NLL$_{\text{emo}}$ and NLL$_{\text{cause}}$ from Equation 4 and Equation 5.

Multi. The first model, Multi, is the classical multi-task learning framework with hard parameter sharing, where both tasks share the same BERT layers. Two dense layers for emotion classification and cause tagging operate at the same time from the same BERT layers, and we train both of the tasks simultaneously. That is, we simply calculate our emotion scores e and cause tag scores c from the same set of hidden states H.

We further develop two additional multi-task models with the intuition that we can design more explicit and concrete task dependencies than simple parameter sharing in the representation layer.

Multi$_{C\to E}$. We assume that if a certain text span is given as the cause of an emotion, it should be possible to classify that emotion correctly while looking only at the words of the cause span. Therefore, we propose the Multi$_{C\to E}$ model, the architecture of which is illustrated in Figure 3a. This model begins with the single-task cause detection model, $BERT_C$, which produces a probability distribution $P(y_i|x_i)$ over IOB tags for each token x_i, where $P(y_i|x_i) = c_i$ from Equation 3. Then, for each token, we calculate the probability that it is part of the cause as $P(\text{Cause}|x_i) = P(B|x_i) + P(I|x_i) = 1 - P(O|x_i)$. We feed the resulting probabilities through a softmax over the sequence and use them as an attention distribution over the input tokens in order to pool the hidden representations and perform emotion classification: attention is computed as in Equation 1, where $\alpha_i = \frac{\exp P(\text{Cause}|x_i)}{\sum_{j=1}^{n}\exp P(\text{Cause}|x_i)}$, and emotion classification as in Equation 2. For the Multi$_{C\to E}$ model, we apply teacher forcing at training time, and the gold cause spans are used to

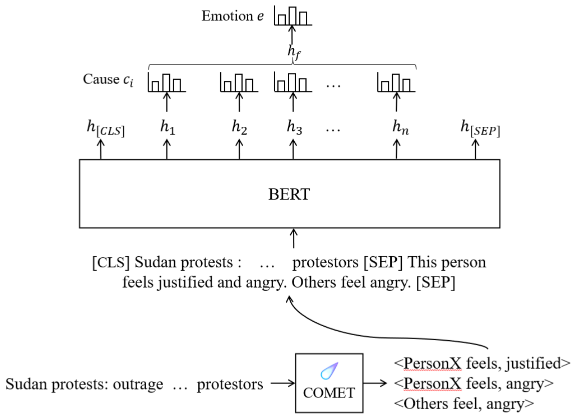

Figure 4: The architecture of our proposed $\text{Multi}_{C\rightarrow E}^{COMET}$ model.

create the attention weights before emotion classification (which means that $P(\text{Cause}|x_i) \in \{0, 1\}$). At inference time, the model uses the predicted cause span instead.

Multi$_{E\rightarrow C}$. Next, we hypothesize that knowledge of the predicted emotion should help us identify salient cause words. The Multi$_{E\rightarrow C}$ model first performs emotion classification, which results in a probability distribution over predicted emotion labels, as in the BERT$_E$ model and Equation 2. We additionally keep an emotion embedding matrix E, where $E[i]$ is a learnable representation of the i-th emotion label (see Figure 3b) with dimension d_e (in our experiments, we set $d_e = 300$). We use the predicted label probabilities e to calculate a weighted sum of the emotion embeddings, i.e., $M = \sum_i e_i \cdot E[i]$. We then concatenate M to the hidden representation of each token and perform emotion cause tagging with a final dense layer, i.e., $c_i = \text{softmax}(W_{c'}[h_i; M] + b_{c'})$, where ; is the concatenation operator and $W_{c'} \in \mathbb{R}^{|\mathcal{C}|\times(d_{BERT}+d_e)}$ and $b_{c'} \in \mathbb{R}^{|\mathcal{C}|}$ are trainable parameters. In the Multi$_{E\rightarrow C}$ model, we again do teacher forcing and use the gold emotion labels before doing the sequence tagging for cause detection (i.e., e is a one-hot vector where the gold emotion label has probability 1 and all other emotion labels have probability 0). At inference time, the model will use the predicted emotion distribution instead.

4.3 Adapted Knowledge Models

Recent work has shown that fine-tuning pre-trained language models such as GPT-2 on *knowledge graph tuples* such as ConceptNet (Li et al., 2016) or ATOMIC (Sap et al., 2018) allows these models to express their implicit knowledge directly (Bosselut et al., 2019). These adapted *knowledge models* (e.g., COMET (Bosselut et al., 2019)) can produce common-sense knowledge on-demand for any entity, relation or event. Considering that common-sense knowledge plays an important role in understanding implicitly expressed emotions and the reasons for those emotions, we explore the use of common-sense knowledge for our tasks, in particular the use of COMET adaptively pre-trained on the ATOMIC event-centric knowledge base. ATOMIC's event relations include "xReact" and "oReact", which describe the feelings of certain entities after the input event occurs. For example, ATOMIC's authors present the example of <PersonX pays PersonY a compliment, xReact, PersonX will feel good>. xReact refers to the feelings of the primary entity in the event, and oReact refers to the feelings of others (in this instance, oReact yields "PersonY will feel flattered"). For example, using the headline "Sudan protests: Outrage as troops open fire on protestors", COMET-ATOMIC outputs that PersonX feels justified, PersonX feels angry, Others feel angry, and so on (Figure 4). To use this knowledge model in our task, we modify our approach by reframing our single-sequence classification task as a sequence-pair classification task (for which BERT can be used directly). We feed our input headlines into COMET-ATOMIC (using the model weights released by the authors), collect the top two outputs for xReact and oReact using beam search decoding, and then feed them into BERT alongside the input headlines, as a second sequence using the SEP token. That is, our input to BERT is now $X = [[\text{CLS}], x_1, x_2, ..., x_n, [\text{SEP}], z_1, z_2, ..., z_m, [\text{SEP}]]$, where z_i are the m WordPiece tokens of our COMET output and are preprocessed in the same way as x_i. We hypothesize that, since pre-trained BERT is trained with a next sentence prediction objective, expressing the COMET outputs as a grammatical sentence will help BERT make better use of them, so we formulate this second sequence as complete sentences (e.g., "This person feels... Others feel...") (Figure 4).

This approach allows us incorporate information from COMET into all our single- and multi-task BERT-based models; the example shown in Figure 4 is our Multi$_{C\rightarrow E}$ model. We refer to the COMET variants of these mod-

	Emotion Macro F1	**Emotion Accuracy**	**Cause Span F1**
BERT	37.25 ± 1.30	38.50 ± 0.84	37.49 ± 1.94
BERT^{COMET}	37.74 ± 0.84	38.50 ± 1.14	39.27 ± 1.85
Multi	36.91 ± 1.48	38.34 ± 1.94	38.35 ± 3.89
Multi$_{C\rightarrow E}$	37.74 ± 2.12	38.74 ± 2.07	39.08 ± 3.73
Multi$_{E\rightarrow C}$	38.26 ± 3.28	39.69 ± 3.41	38.83 ± 1.60
Multi^{COMET}	37.06 ± 2.04	39.05 ± 0.98	**39.50** ± 2.25
Multi$^{COMET}_{C\rightarrow E}$	**39.26*** ± 1.13	**40.79** ± 2.17	38.68 ± 1.36
Multi$^{COMET}_{E\rightarrow C}$	37.44 ± 1.37	38.58 ± 1.44	36.27 ± 1.31

Table 1: The results of our models, averaged over five runs with the same five distinct random seeds. The model with the highest mean performance under each metric is bolded. Results marked with a * are statistically significant above the single-task BERT baseline by the paired t-test ($p < 0.05$).

els as: BERT^{COMET} (single-task models) and Multi^{COMET}, Multi$^{COMET}_{C\rightarrow E}$, Multi$^{COMET}_{E\rightarrow C}$ for the three multi-task models.

5 Experimental Setup

Evaluation Metrics For emotion classification, we report macro-averaged F1 and accuracy. For cause tagging, we report exact span-level F1 (which we refer to as *span F1*), as developed for named entity recognition (e.g., Tjong Kim Sang and De Meulder (2003)), where a span is marked as correct if and only if its type and span boundaries match the gold exactly[4].

Training and Hyperparameter Selection The classification layers are initialized randomly from a uniform distribution over $[-0.07, 0.07]$[5], and all the parameters are trained on our dataset for up to 20 epochs, with early stopping based on the performance on the validation data (macro F1 for emotion, span F1 for cause). All models are trained with the Adam optimizer (Kingma and Ba, 2015). We highlight again that for our Multi$_{C\rightarrow E}$ and Multi$_{E\rightarrow C}$ models, we use teacher forced during training to avoid cascading training error. Because the subset of the data we use is relatively small, we follow current best practices for dealing with neural models on small data and select hyperparameters and models using the average performance of five models with different fixed random seeds on the development set. We then base our models' performance on the average of the results from these five runs (e.g., reported emotion F1 is the average of the emotion F1 scores for each of our five runs). For our joint models, since our novel models revolve around using one task as input for the other, we separately tune two sets of hyperparameters for each model, one based on each of the single-task metrics, yielding, for example, one Multi model optimized for predicting emotion and one optimized for predicting cause. The hyperparameters we tune are dropout in our linear layers, initial learning rate of the optimizer, COMET relation type, lambda weight for our multi-task models, and the type of pooler for emotion classification (enumerated in subsection 4.1).

6 Results

We present the results of our models in Table 1[6]. We see that the overall best model for each task is a multi-task adapted knowledge model, with Multi$^{COMET}_{C\rightarrow E}$ performing best for emotion (which is a statistically significant improvement over BERT by the paired t-test, $p < 0.05$) and Multi^{COMET} performing best for cause. These results seem to support our two hypotheses: 1) emotion recognition and emotion cause detection can inform each other and 2) common-sense knowledge is helpful to infer the emotion and the cause for that emotion expressed in text. Specifically, we notice that Multi$_{C\rightarrow E}$ alone does not outperform BERT on either cause or emotion, but Multi$^{COMET}_{C\rightarrow E}$ outperforms both BERT and Multi$_{C\rightarrow E}$ on both tasks. For cause, we also see additional benefits of common-

---

[4] Our cause tagging task has only one type, "cause", as GoodNewsEveryone is aggregated such that each data point has exactly one emotion-cause pair. We note that this problem formulation leaves open the possibility of multiple emotion-cause pairs.

[5] The default initialization from the `gluon` package: `https://mxnet.apache.org/versions/1.7.0/api/python/docs/api/gluon/index.html`

[6] Oberländer and Klinger (2020) report an F1 score of 34 in this problem setting on this dataset, but on a larger subset of the data (as they do not limit themselves to the Ekman emotions) and so we cannot directly compare our work to theirs.

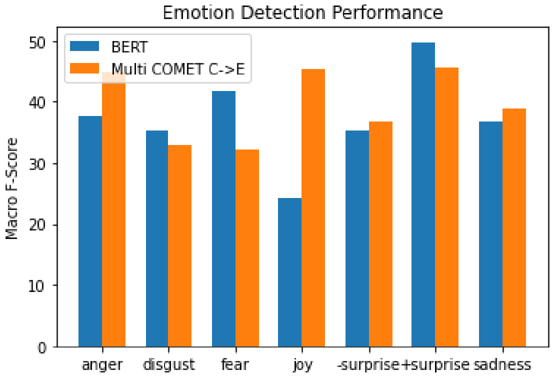

Figure 5: Performance of the BERT and Multi$^{COMET}_{C\rightarrow E}$ models on emotion classification.

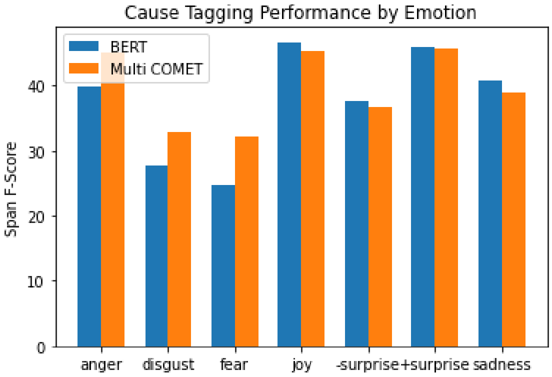

Figure 6: Performance of the BERT and Multi^{COMET} models on cause tagging, broken down by emotion.

sense reasoning alone: BERT^{COMET} outperforms BERT (multi-task modeling alone, Multi, also outperforms BERT for this task) and Multi^{COMET} outperforms Multi. These results speak to the differences between the two tasks, suggesting that common-sense reasoning, which aims to generates implicit emotions, and cause information may be complementary for emotion detection, but that for cause tagging, common-sense reasoning and given emotion information may overlap. The common-sense reasoning we have used in this task (xReact and oReact from ATOMIC) is expressed as possible emotional reactions to an input situation, so this makes intuitive sense.

Finally, we also present per-emotion results for our best model for each task (Multi$^{COMET}_{C\rightarrow E}$ for emotion and Multi^{COMET} for cause) against the single-task BERT baselines in Figure 5 and Figure 6; these per-emotion scores are again the average performance of models trained with each of our five random seeds. We see that each task improves on a different set of emotions: for emotion classification Multi$^{COMET}_{C\rightarrow E}$ consistently improves over BERT by a significant margin on joy and to a lesser extent on anger and sadness. Meanwhile, for cause tagging, Multi^{COMET} improves over BERT on anger, disgust, and fear, while yielding very similar performance on the rest of the emotions.

7 Analysis and Discussion

In order to understand the impact of common-sense reasoning and multi-task modeling for the two tasks, we provide several types of analysis in addition to our results in section 6. First, we include examples of our various models' outputs showcasing the impact of our methods (subsection 7.1). Second, we carry out an analysis of the dataset, focusing on the impact of label variation among multiple annotators on the models' performance (subsection 7.2).

7.1 Example Outputs

We provide some example outputs from our systems for both cause and emotion in Table 2; the various Multi models have been grouped together for readability and because they often produce similar outputs, but the outputs for every model are available in the appendix. In the first example, the addition of COMET to BERT informs the model enough to choose the gold emotion label; in the third and fourth, either COMET or multi-task learning is enough to help the model select key words that should be included in the cause (*return* and *triple shooting*). We also particularly note the second example, in which multi-task learning is needed both for the BERT and BERT^{COMET} models to be able to correctly predict the gold emotion. This suggests that for cause, both common-sense reasoning and emotion classification may carry overlapping useful information for cause tagging, while for emotion, different instances may be helped more by different aspects of our models.

7.2 Label Agreement

Upon inspection of the GoodNewsEveryone data, we discover significant variation in the emotion labels produced by annotators as cautioned by the authors in their original publication[7]. From our inspection of the development data, we see recur-

---

[7] While the authors selected data according to agreement on the emotion labeling task, they found that in only 75% of cases do at least 3 annotators agree, with diminishing rates of agreement for more annotators.

<table>
<tr><th>BERT</th><th>Multitask</th><th>BERT^{COMET}</th><th>Multitask^{COMET}</th></tr>
<tr><td colspan="4">Mexico reels from shooting attack in El Paso
fear</td></tr>
<tr><td>negative surprise</td><td>negative surprise</td><td>fear</td><td>fear</td></tr>
<tr><td colspan="4">Insane video shows Viking Sky cruise ship thrown into chaos at sea
fear</td></tr>
<tr><td>negative surprise</td><td>fear</td><td>negative surprise</td><td>fear</td></tr>
<tr><td colspan="4">Durant could return for Game 3
positive surprise</td></tr>
<tr><td>for game</td><td colspan="3">could return for game</td></tr>
<tr><td colspan="4">Dan Fagan: Triple shooting near New Orleans School yet another sign of city's crime problem
negative surprise</td></tr>
<tr><td>school yet another sign of city's crime</td><td colspan="3">: triple shooting near new orleans school yet another sign of city's crime</td></tr>
</table>

Table 2: Example outputs from our systems. For each example, the gold cause is highlighted in yellow and the gold emotion is given under the text; the first two examples give our models' emotion outputs; the latter two, their causes. Joined cells show that multiple models produced the same output. To make this table easier to read, "Multitask" here may refer to Multi, Multi$_{E \to C}$, or Multi$_{C \to E}$ (details on selection and results for each individual model available in appendix; most multi-task models gave similar outputs).

Metric	**BERT**	**BERT**^{COMET}	**Multi**	**Multi**$_{E\to C}$	**Multi**$_{C\to E}$	**Multi**^{COM}	**Multi**$^{COMET}_{E\to C}$	**Multi**$^{COMET}_{C\to E}$
Acc. (Gold)	38.50	38.50	38.34	39.68	38.74	39.05	38.58	40.79
Acc. (¬Gold)	23.48	23.24	22.37	21.11	22.85	21.26	22.45	20.08

Table 3: Comparison of gold accuracy and non-gold (¬gold) accuracy for our emotion classification models.

ring cases where different annotators give directly opposing labels for the same input, depending on how they interpret the headline and whose emotions they choose to focus on. For example, our development set includes the following example: *Simona Stuns Serena at Wimbledon: Game, Set and "Best Match" for Halep*. The gold adjudicated emotion label for this example is *negative surprise*, but annotators actually included multiple primary and secondary emotion labels including *joy*, *negative surprise*, *positive surprise*, *pride*, and *shame*, which can be understood as various emotions felt by the two entities participant in the event (Simona Halep and Serena Williams). For this input, COMET suggests xReact may be *happy* or *proud* and oReact may be *happy* — these reactions are likely most appropriate for tennis player Simona Halep, but not the only possible emotion that can be inferred from the headline.

Inspired by the variation in the data, we compute also models' accuracy using the human annotations that did not agree with the gold (i.e., a predicted emotion label is correct if it was suggested by a human annotator but was not part of a majority vote to be included in the gold). We denote this ¬Gold, and we compare the performance of our models with respect to Gold and ¬Gold. We present the results of this analysis in Table 3[8]. In this table, a higher ¬Gold accuracy means that the model is more likely to produce emotion labels that were not the gold but were suggested by some annotator. First of all, we note that all models have a relatively high ¬Gold accuracy (about half the magnitude of their gold accuracy); we believe this reflects the wide variety of annotations given by the annotators. We see a trade-off between the Gold and ¬Gold accuracy, and we note that generally the single-task models have higher ¬Gold accuracy and the COMET-enhanced multi-task models have higher Gold accuracy. This suggests that our language models have general knowledge about emotion already, but that applying common-sense knowledge helps pare down the space of plausible outputs to those that are most commonly selected by human annotators. Recall

---

[8] Note that we perform this analysis on just one of our five runs of the model, so the accuracy numbers do not exactly correspond to those in Table 1.

that this dataset was annotated by taking the most frequent of the annotator-provided emotion labels. Further, since the multi-task models have higher Gold accuracy and lower ¬Gold accuracy than the single-task models, this suggests that also predicting the cause of an emotion causes the model to narrow down the space of possible emotion labels to only those that are most common.

8 Conclusions and Future Work

We present a common-sense knowledge-enhanced multi-task framework for joint emotion detection and emotion cause tagging. Our inclusion of common-sense reasoning through COMET, combined with multi-task learning, yields performance gains on both tasks including significant gains on emotion classification. We highlight the fact that this work frames the cause extraction task as a span tagging task, allowing for the future possibility of including multiple emotion-cause pairs per input or multiple causes per emotion and allowing the cause to take on any grammatical role. Finally, we present an analysis of our dataset and models, showing that labeling emotion and its semantic roles is a hard task with annotator variability, but that common-sense knowledge helps language models focus on the most prominent emotions according to human annotators. In future work, we hope to explore ways to integrate common-sense knowledge more innately into our classifiers and ways to apply these models to other fine-grained emotion tasks such as detecting the experiencer or the target of an emotion.

Acknowledgements

We would like to thank our reviewers as well as the members of Amazon AI team for their constructive and insightful feedback.

Ethical Considerations

Our intended use for this work is as a tool to help understand emotions expressed in text. We propose that it may be useful for things like product reviews (where producers and consumers can rapidly assess reviews for aspects of their products to improve or expand), disaster relief (where those in need of help from any type of disaster can benefit if relief agents can understand what events are causing negative emotions, during and after the initial disaster), and policymaking (where constituents can benefit if policymakers can see real data about what policies are helpful or not and act in their interests). These applications do depend on the intentions of the user, and a malicious actor may certainly misuse the ability to (accurately or inaccurately) detect emotions and their causes. We do not feel it responsible to publicly list the ways in which this may happen in this paper. We also believe that regulators and operators of this technology should be aware that it is still in its nascent stages and does not represent an infallible oracle — the predictions of this and any model should be reviewed by humans in the loop, and we feel that general public awareness of the limitations and mistakes of these models may help mitigate any possible harm. If these models are inaccurate, they will output either the incorrect emotion or the incorrect cause; blindly trusting the model's predictions without examining them may lead to unfair consequences in any of the above applications (e.g., failure to help someone whose text is misclassified as positive surprise during a natural disaster or a worsened product or policy if causes are incorrectly predicted). We additionally note that in its current form, this work is intended to detect the emotions that are expressed in text (headlines), and not those of the reader.

We concede that the data used in this work consists of news headlines and may not be the most adaptable to the use cases we describe above; we caution that models trained on these data will likely require domain adaptation to perform well in other settings. Bostan et al. (2020) report that their data comes from the Media Bias Chart[9], which reports that their news sources contain a mix of political views, rated by annotators who also self-reported a mix of political views. We note that these data are all United States-based and in English. Bostan et al. (2020) do sub-select the news articles according to impact on Twitter and Reddit, which have their own user-base biases[10], typically towards young, white American men; therefore, the data is more likely to be relevant to these demographics. The language used in headlines will likely most resemble Standard American English as well, and therefore our models will be difficult to use directly on other dialects and vernaculars.

---

[9] `https://www.adfontesmedia.com/about-the-interactive-media-bias-chart/`

[10] `https://www.pewresearch.org/internet/fact-sheet/social-media/`

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. EmoNet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728, Vancouver, Canada. Association for Computational Linguistics.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, page 579–586, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics.

Laura Ana Maria Bostan, Evgeny Kim, and Roman Klinger. 2020. Goodnewseveryone: A corpus of news headlines annotated with emotions, semantic roles, and reader perception. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 1554–1566. European Language Resources Association.

Ying Chen, Wenjun Hou, Xiyao Cheng, and Shoushan Li. 2018. Joint learning for emotion classification and emotion cause detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 646–651, Brussels, Belgium. Association for Computational Linguistics.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Zixiang Ding, Rui Xia, and Jianfei Yu. 2020. ECPE-2D: Emotion-cause pair extraction based on joint two-dimensional representation, interaction and prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3161–3170, Online. Association for Computational Linguistics.

Chuang Fan, Chaofa Yuan, Jiachen Du, Lin Gui, Min Yang, and Ruifeng Xu. 2020. Transition-based directed graph construction for emotion-cause pair extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3707–3717, Online. Association for Computational Linguistics.

D. Ghazi, Diana Inkpen, and S. Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *CICLing*.

Deepanway Ghosal, Navonil Majumder, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. 2020. COSMIC: COmmonSense knowledge for eMotion identification in conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2470–2481, Online. Association for Computational Linguistics.

Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016. Event-driven emotion cause extraction with corpus construction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1639–1649, Austin, Texas. Association for Computational Linguistics.

Evgeny Kim and Roman Klinger. 2018. Who feels what and why? annotation of a literature corpus with semantic roles of emotions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1345–1359, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. *J. Artif. Int. Res.*, 50(1):723–762.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Saif Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *CoRR*, abs/1308.6297.

Saif Mohammad, Xiaodan Zhu, and Joel Martin. 2014. Semantic role labeling of emotions in tweets. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 32–41.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Laura Ana Maria Oberländer and Roman Klinger. 2020. Token sequence labeling vs. clause classification for English emotion stimulus detection. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 58–70.

Laura Ana Maria Oberländer, Kevin Reich, and Roman Klinger. 2020. Experiencers, stimuli, or targets: Which semantic roles enable machine learning to infer the emotions? In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 119–128.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, and Rada Mihalcea. 2020a. Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research.

Soujanya Poria, Navonil Majumder, Devamanyu Hazarika, Deepanway Ghosal, Rishabh Bhardwaj, Samson Yu Bai Jian, Romila Ghosh, Niyati Chhaya, Alexander Gelbukh, and Rada Mihalcea. 2020b. Recognizing emotion cause in conversations.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. *CoRR*, cmp-lg/9505040.

Sara Rosenthal, Saif M. Mohammad, Preslav Nakov, Alan Ritter, Svetlana Kiritchenko, and Veselin Stoyanov. 2019. Semeval-2015 task 10: Sentiment analysis in twitter. *CoRR*, abs/1912.02387.

Irene Russo, Tommaso Caselli, Francesco Rubino, Ester Boldrini, and Patricio Martínez-Barco. 2011. EMOCause: An easy-adaptable approach to extract emotion cause contexts. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, pages 153–160, Portland, Oregon. Association for Computational Linguistics.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2018. ATOMIC: an atlas of machine commonsense for if-then reasoning. *CoRR*, abs/1811.00146.

Carlo Strapparava and Rada Mihalcea. 2010. Annotating and identifying emotions in text. In Giuliano Armano, Marco de Gemmis, Giovanni Semeraro, and Eloisa Vargiu, editors, *Intelligent Information Access*, volume 301, pages 21–38.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Penghui Wei, Jiahao Zhao, and Wenji Mao. 2020. Effective inter-clause modeling for end-to-end emotion-cause pair extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3171–3181, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Rui Xia and Zixiang Ding. 2019. Emotion-cause pair extraction: A new task to emotion analysis in texts. *CoRR*, abs/1906.01267.

Rui Xia, Mengran Zhang, and Zixiang Ding. 2019. RTHN: A rnn-transformer hierarchical network for emotion cause extraction. *CoRR*, abs/1906.01236.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. *CoRR*, abs/1809.04505.

Chaofa Yuan, Chuang Fan, Jianzhu Bao, and Ruifeng Xu. 2020. Emotion-cause pair extraction as sequence labeling based on a novel tagging scheme. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3568–3573, Online. Association for Computational Linguistics.

A Appendix

A.1 Hyperparameter Tuning

We include descriptions of our hyperparameter tuning setup and the selected hyperparmeters for each of our models in Table 4; we note that single-task cause models (BERT$_C$ and COMET$_C$) do not tune the `pooler`, all single-task models do not tune the lambda parameter, and all non-common-sense models do not tune `comet_relations`. The parameters selected by all of our models can be seen in Table 5, Table 6, and Table 7. All of our models are trained with minibatches of size $b = 32$.

We used Bayesian optimization as implemented by Amazon SageMaker[11] to tune these parameters, giving the learning rate a logarithmic scale and the dropout and lambda a linear one and allowing 75 iterations of parameter choice before selecting the setting with the best performance on the development set. Each individual instance of each model consisted of five different restarts with five distinct random seeds; one of these instances took approximately five minutes on a single Tesla V100 GPU, for a total of about 6.25 GPU-hours per model and thus 87.5 GPU-hours overall (since each multi-task model was trained twice: once optimized for emotion and once optimized for cause).

A.2 Model Sizes

Our models' sizes are dominated by BERT-base, which has 110 million trainable parameters (Devlin et al., 2019). We note that our trainable dense layers that interface with BERT have sizes 768 × 7 for emotion classification, 768 × 3 for cause tagging, and 1068 × 7 for our Multi$_{E \to C}$ models, while our emotion embedding matrix E has 300×7 trainable parameters. Our fine-tuning process does continue to tune all of BERT's parameters.

A.3 Extended Examples

We include the output of all models for our four selected examples in subsection 7.1 in Table 8, Table 9, Table 10, and Table 11.

---

[11] `https://aws.amazon.com/sagemaker/`

Parameter Name	Type	Range or Values
`pooler`	Categorical	`[cls, mean, max, attention]`
`learning rate`	Continuous	$[10^{-6}, 10^{-4}]$
`dropout`	Continuous	$[0, 0.9]$
`lambda`	Continuous	$[0.1, 0.9]$
`comet_relations`	Categorical	`[xReact, oReact, both]`

Table 4: Our hyperparameter search ranges.

Model	**Target Task**	**Parameter Name**	**Parameter Value**
BERT$_E$	Emotion	pooler	`cls`
		dropout	0.8999992513311351
		lr	$2.0872134970009262 \times 10^{-5}$
BERT$_C$	Cause	dropout	0.04011659404129298
		lr	$9.609926650689472 \times 10^{-5}$
BERT$_E^{COMET}$	Emotion	pooler	`cls`
		dropout	0.6467089448672897
		lr	$3.548213539029209 \times 10^{-5}$
		comet_relations	`both`
BERT$_C^{COMET}$	Cause	dropout	0.8806119007595122
		lr	$9.913585728926367 \times 10^{-5}$
		comet_relations	`xReact`

Table 5: The selected hyperparameter values for our single-task models.

Model	**Target Task**	**Parameter Name**	**Parameter Value**
Multi	Emotion	pooler	`mean`
		dropout	0.1438975482079587
		lr	$2.170218150294524 \times 10^{-5}$
		lambda	0.3736515054477897
	Cause	pooler	`cls`
		dropout	0.8929935089177194
		lr	$9.929740332732521 \times 10^{-5}$
		lambda	0.6103686494768474
Multi$_{E \to C}$	Emotion	pooler	`max`
		dropout	0.2511612834815036
		lr	$3.179072019077849 \times 10^{-5}$
		lambda	0.4938386162506444
	Cause	pooler	`max`
		dropout	0.763419047616446
		lr	$8.680439371509037 \times 10^{-5}$
		lambda	0.1341940851689314
Multi$_{C \to E}$	Emotion	pooler	`max`
		dropout	0.8138762283528274
		lr	$4.2586257586160994 \times 10^{-5}$
		lambda	0.8531247637209994
	Cause	pooler	`mean`
		dropout	0.6992099059226856
		lr	$9.859155309987275 \times 10^{-5}$
		lambda	0.4855821360212248

Table 6: The selected hyperparameter values for our multi-task BERT models.

Model	**Target Task**	**Parameter Name**	**Parameter Value**
Multi^{COMET}	Emotion	pooler	`max`
		dropout	0.22350077887111716
		lr	$3.137385699389837 \times 10^{-5}$
		lambda	0.7676911585403968
		comet_relations	`both`
	Cause	pooler	`mean`
		dropout	0.8891347000216091
		lr	$8.123006047625093 \times 10^{-5}$
		lambda	0.1
		comet_relations	`both`
Multi$^{COMET}_{E \to C}$	Emotion	pooler	`mean`
		dropout	0.1372637910712323
		lr	$3.0408118480380588 \times 10^{-5}$
		lambda	0.8968243966922735
		comet_relations	`both`
	Cause	pooler	`max`
		dropout	0.5319636087561394
		lr	$7.581334242472624 \times 10^{-5}$
		lambda	0.10896064677810494
		comet_relations	`both`
Multi$^{COMET}_{C \to E}$	Emotion	pooler	`cls`
		dropout	0.7359624181177503
		lr	$1.9853909769532754 \times 10^{-5}$
		lambda	0.7947522633173147
		comet_relations	`both`
	Cause	pooler	`max`
		dropout	0.01896406469706125
		lr	$8.360862387915605 \times 10^{-5}$
		lambda	0.14588492191321054
		comet_relations	`oReact`

Table 7: The selected hyperparameter values for our multi-task COMET models.

Mexico reels from shooting attack in El Paso

fear

Model	Output
BERT	negative surprise
BERT^{COMET}	fear
Multi	negative surprise
Multi$_{C \to E}$	negative surprise
Multi$_{E \to C}$	negative surprise
Multi^{COMET}	fear
Multi$^{COMET}_{C \to E}$	fear
Multi$^{COMET}_{E \to C}$	fear

Table 8: Full model outputs for our first provided example.

Insane video shows Viking Sky cruise ship thrown into chaos at sea

fear

Model	Output
BERT	negative surprise
BERT^{COMET}	negative surprise
Multi	fear
Multi$_{C\rightarrow E}$	negative surprise
Multi$_{E\rightarrow C}$	fear
Multi^{COMET}	fear
Multi$^{COMET}_{C\rightarrow E}$	fear
Multi$^{COMET}_{E\rightarrow C}$	negative surprise

Table 9: Full model outputs for our second provided example.

Durant could return for Game 3

positive surprise

Model	Output
BERT	for game
BERT^{COMET}	could return for game
Multi	could return for game
Multi$_{C\rightarrow E}$	could return for game
Multi$_{E\rightarrow C}$	could return for game
Multi^{COMET}	could return for game
Multi$^{COMET}_{C\rightarrow E}$	could return for game
Multi$^{COMET}_{E\rightarrow C}$	could return for game

Table 10: Full model outputs for our third provided example.

Dan Fagan: Triple shooting near New Orleans School yet another sign of city’s crime problem

negative surprise

Model	Output
BERT	school yet another sign of city’s crime
BERT^{COMET}	: triple shooting near new orleans school yet another sign of city’s crime
Multi	shooting near new orleans school yet another sign of city’s crime
Multi$_{C\rightarrow E}$	: triple shooting near new orleans school yet another sign of city’s crime
Multi$_{E\rightarrow C}$	: triple shooting near new orleans school yet another sign of city’s crime
Multi^{COMET}	: triple shooting near new orleans school yet another sign of city’s crime
Multi$^{COMET}_{C\rightarrow E}$	: triple shooting near new orleans school yet another sign of city’s crime
Multi$^{COMET}_{E\rightarrow C}$	: triple shooting near new orleans school yet another sign of city’s crime

Table 11: Full model outputs for our fourth provided example.

The Utility and Interplay of Gazetteers and Entity Segmentation for Named Entity Recognition in English

Oshin Agarwal[1] and **Ani Nenkova**[1,2]
[1]University of Pennsylvania
[2]Adobe Research
oagarwal@seas.upenn.edu, nenkova@adobe.com

Abstract

Recent papers have introduced methods to incorporate gazetteer features and entity segmentation techniques in neural named entity recognition models. These papers rely on different resources and include features not related to the use of gazetteers, rendering impossible the comparison of the relative effectiveness of the approaches. Here, we provide a comprehensive overview of methods for incorporating gazetteers and for entity segmentation. We evaluate representative methods from each in similar settings for a fair comparison and identify the ones that are consistently better across datasets and input representations. We further show that gazetteers improve entity segmentation and not just entity typing. Hence, we explore their utility in recognizing long entities, a problem for which entity segmentation techniques were developed. Our work explains the mechanisms via which gazetteers improve the performance of neural NER models.

1 Introduction

Named Entity Recognition (NER) has the unique property of being a task appealing to researchers and at the same time being fairly robust for immediate practical applications. In many domains, it is of interest to identify segments of text conveying a concept of a given type—a person (Grishman and Sundheim, 1996), an event (Hovy et al., 2006), a disease (Doğan et al., 2014), a gene (Kim et al., 2003), a chemical (Krallinger et al., 2015), a food (Magnolini et al., 2019), an item of clothing (Putthividhya and Hu, 2011), a research technique (Augenstein et al., 2017a), etc.

Approaches to NER are typically not domain-specific, treating the problem as a sequence labelling task regardless of the categories of interest. Yet, researchers also widely agree that named entity recognition is a knowledge intensive task (Ratinov and Roth, 2009; Seyler et al., 2018): the availability of external knowledge resources in the form of lists of example entities of a given type, or *gazetteers*, improve performance almost universally. Since gazetteers are readily available, from knowledge bases, databases of products and specialized ontologies, having practical guidance on how to handle gazetteers in NER would be valuable.

In this paper, we provide a survey of how gazetteers have been used in neural approaches to NER in English and compare key approaches with the popular biLSTM-CRF architecture. To ensure that our conclusions accurately characterize the utility of gazetteers, we test the approaches on several datasets from different genres covering newswire, conversations and twitter. The extensive head-to-head comparison reveals that while certain approaches are consistently beneficial, others are variable, impressively improving results on one dataset but reducing performance on others.

Gazetteers typically contain multi-word entries. In contrast, the majority of entity mentions in text are single word, with lower performance of models on longer entities. This discrepancy highlights a potential application of gazetteers for improving NER prediction for mentions of long entities. Recent work on entity segmentation[1] as part of the named entity recognition task aims to recognize long entities better. We overview this work and compare these methods similar to the way we compare methods for incorporating gazetteers.

We find that certain ways for incorporating gazetteers show stable improvements across datasets, while segmentation approaches do not appear to be that useful overall. We further explore the interplay of gazetteers with entity segmentation and their role in recognizing long entities. We find that incorporating gazetteers improves entity

---

[1] Identifying entity spans without types

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3990–4002
August 1–6, 2021. ©2021 Association for Computational Linguistics

segmentation, not just entity typing and depending on the input representation to the model, gazetteer types may be irrelevant. We also find that incorporating gazetteers can serve as an alternate method to recognizing long entities, likely due to the abundant presence of multi-word entities in the Wikipedia-derived gazetteers we used.

Our work provides *(i)* a concise overview of methods for incorporating gazetteers and entity segmentation in NER, *(ii)* a principled comparison of representative approaches for each aspect, and *(iii)* novel findings and analyses of the interplay between gazetteers and segmentation. Our findings can inform both future researchers and practitioners interested in NER.

2 biLSTM-CRF architecture for NER

We explore variants of the now classic biLSTM-CRF architecture for NER (Huang et al., 2015). We overview how gazetteer features and segmentation can be integrated in this paradigm and carry out a comparison of several representative methods. We use two input word representations: the 300-d GloVe vectors trained on Common Crawl (Pennington et al., 2014), which is the dominant representation in NER, and the 1024-d contextual ELMo (Peters et al., 2018) representations trained on the 1B Word Benchmark (Chelba et al., 2014). In each case, character-based word representations learned with CNNs (Ma and Hovy, 2016) are also concatenated. The final concatenated representation is used as input to bidirectional LSTMs (Hochreiter and Schmidhuber, 1997), followed by a CRF (Lafferty et al., 2001) layer. We use the implementation by Lample et al. (2016) for most experiments.

3 Why Use Gazetteers?

Gazetteers are large dictionaries consisting of lists of entities of a particular type. For example, a person gazetteer may consist of full names and parts of names such as the first names of people. Before we start our discussion of methods for incorporating gazetteers, it is worth considering if we have sufficient evidence that they are needed at all.

Gazetteers are needed for better generalization through improved entity coverage, to predict the type for words that have not been encountered in training and possibly even pre-training. This means the need will be more acute for practical deployment of NER and will be less pronounced on fixed datasets in which train and test data are sampled from overlapping or adjacent time periods, with high overlap of entities across both.

The most compelling example for the need to handle unseen language comes from work on NER on Twitter. Language on Twitter changes rapidly, much more rapidly than for other types of text (Eisenstein, 2013). This change requires models to be retrained periodically to maintain optimal performance for the current time period (Rijhwani and Preotiuc-Pietro, 2020). An alternative to retraining, not yet explored in literature, is to develop methods that can make use of gazetteers that possibly could be updated more quickly and cheaply compared to continuously annotating new training data.

Even in stable domains such as newswire, the ability of models to generalize to words not seen in the training data is low (Augenstein et al., 2017b; Fu et al., 2020a,b). Both traditional models with hand-crafted features (Finkel et al., 2005; Okazaki, 2007) and more recent neural network approaches (Collobert et al., 2011; Huang et al., 2015; Peters et al., 2018; Devlin et al., 2019) achieve lower performance on entities unseen in the training data.

Methods that make use of large pretrained language representations, neural (Collobert et al., 2011) or not (Miller et al., 2004), can ameliorate the problem of coverage to some extent. We do not yet know enough about how pretraining data should be chosen (Cherry and Guo, 2015), though there is some evidence that performance on downstream tasks correlates with the vocabulary coverage in the pre-training data (Dai et al., 2019). Prior work has reported that performance is lowest for words that appear neither in the training nor the pretraining vocabulary (Ma and Hovy, 2016). Moreover, the deteriorated performance on out of vocabulary words is not necessarily a failure of the models: many contexts simply do not provide sufficient knowledge to predict the type of an entity, even for people (Agarwal et al., 2021). Models need to expand their knowledge of entities and gazetteers are a natural way for doing that.

4 Where Do Gazetteers Come From?

Existing tables, lists, directories, databases and knowledge bases are widely available and can be used to derive gazetteers. Some researchers have specifically compiled various resources to form gazetteers, while others make use of those provided in prior work. Early work collected gazetteers from the CIA factbook for geographic locations, lists of

1	**2**	**3**	**≥ 4**
15	43	18	24

Table 1: Gazetteer entities(%) by length (words).

popular person names, etc (Mikheev et al., 1999). More recently, Ratinov and Roth (2009) derived a gazetteer from the Web and Wikipedia and Chiu and Nichols (2016) used DBPedia.

In our work, we use the Ratinov and Roth (2009) wide-coverage gazetteer. It contains ∼3M entities grouped into ∼30 fine-grained categories. In some experiments, we use all categories, regardless of the entity types in the dataset. In the remaining, we identify the gazetteer category that most closely matches that types in a given dataset and disregard the rest. The mapping can be found in the appendix.

Table 1 shows the approximate percentage of entities by length in words in our gazetteer. Most entries are of length two, but unlike NER datasets (Table 3), the remaining entries are evenly distribution between length 1, 3 and ≥4. Most notably, around 24% of gazetteer entities have four or more words. In comparison, in NER datasets where entities appear in the context of a sentence that is often a part of a longer document, such long entities typically make up about 2% of all entities. This distributional difference hints at the possibility of using gazetteer to not only improve coverage but also improve performance on longer entities for which segmentation methods are developed.

Regardless of the distribution of entity lengths, the total number of entities in gazetteers is much higher than that in NER datasets so a higher percentage of longer entities does not equate to a small number of short entities.

5 Gazetteer Features for NER

Here, we overview the ways gazetteers have been integrated in NER models.

5.1 Discrete Gazetteer Lookup Features

Feature-based CRF models for NER used gazetteers to generate indicators for each word in a sentence (Bender et al., 2003; Minkov et al., 2005; Ratinov and Roth, 2009; Ritter et al., 2011; Yang et al., 2016; Seyler et al., 2018). The number of indicators equals the number of entity types in the dataset and indicate (with a binary 1/0 value) if the word is part of a gazetteer entry of the given type.

Many neural network approaches continue to incorporate gazetteers as discrete indicator features concatenated to the pre-trained word embeddings as the input (Collobert et al., 2011; Huang et al., 2015). Adding the features in later stages does not work as well (Magnolini et al., 2019). Both Collobert et al. (2011) and Huang et al. (2015) pre-process datasets to match the gazetteer entries to sentences, using both exact matches and multi-word partial matches to gazetteer entries. Chiu and Nichols (2016) perform a similar matching but use four binary values for each label, indicating whether the given word matches the gazetteer entity exactly (S), at the beginning (B), end (E) or the any of the words in between (I).

5.2 Continuous Gazetteer Features

The approach above does not use gazetteers very effectively. Gazetteers contain many more entities of each type than are available in even the largest training set (Table 3). One insight is to use the gazetteers as a additional source of training examples. A simple way is to add the gazetteer entries to the labeled data, without any context. Liu et al. (2019a) report that this data augmentation approach led to much worse overall results, presumably because of the great shift in label distributions. Another approach is to augment the training data by replacing entities in place by other entities from gazetteers. Song et al. (2020) reported no improvement with such a random entity replacement, likely due to the need for manual intervention for replacement of entities of some types to maintain coherence of text (Agarwal et al., 2020).

A much more successful alternative is to learn a separate (or sub-) module, trained to predict types for text spans, using the gazetteer entries and synthetic negative examples sampled from a NER training set or even the gazetteer. We will refer to the separate module as a *gazetteer network*. It is straightforward to integrate the label distribution scores from this model in a semi-Markov CRF for sequence labeling (Ye and Ling, 2018) that operates at the span level (which we describe in greater detail later). The resulting combination is far more effective than discrete indicator gazetteer features.

Magnolini et al. (2019) and Liu et al. (2019b) propose a similar approach. They learn a gazetteer network but instead of using the label score distribution, intermediate word representations (*gazetteer embeddings* henceforth) are incorporated in the NER model. Liu et al. (2019b) use a semi-Markov CRF operating at the span level and generate the gazetteer embeddings for each potential span. They follow the evaluation approach of Ma and Hovy (2016), breaking down results by whether an entity was seen only in training, seen only in pre-training, seen in both and seen in neither. The largest improvement was in the "seen in neither" subset, showing that this approach is particularly helpful for out-of-vocabulary words with respect to the training and pre-training data.

Magnolini et al. (2019) use the standard word-level CRF and hence do not have spans available so they input the full sentence to the gazetteer network. This makes the training and inference setup for the gazetteer network different as entity phrases are used as input during training. They reported mixed results for this approach. In our experiments, we evaluated their method on a larger number of datasets but used a different approach for negative sampling for the gazetteer network training data. We observed some improvement on almost all datasets, contingent on the input representation.

5.3 Contextual Gazetteers

Learning from just the gazetteer has the drawback that the representations do not include any clues about the context in which the entity types are used. The same entity may appear in multiple gazetteers. Given that current methods heavily rely on entity memorization and little on context, this is possibly acceptable. For completeness however, we ought to mention that the link structure of Wikipedia can be used to derive dense representations for entity types directly (Long et al., 2016; Ganea and Hofmann, 2017; Mengge et al., 2020; Ghaddar and Langlais, 2018). Comparing gazetteer representations with and without context would be a direction for exploration in future work.

6 Entity Segmentation in NER

Early work (Collins and Singer, 1999; Downey et al., 2007; Ritter et al., 2011) treated NER as two subtasks, i.e. *entity segmentation:* finding spans of text that refer to named entities, and *typing:* assigning a type to the identified span. Recent efforts have also incorporated entity segmentation explicitly in neural models, with goal of finding longer entities better (Xiao et al., 2019; Ye and Ling, 2018). Such work can be divided into two categories—Multi-task learning and semi-Markov CRFs.

6.1 Multi-task Learning

Multi-task learning (MTL) involves jointly training multiple related tasks using the same representation such that the auxiliary tasks can help with the performance of the target task. Aguilar et al. (2017) use MTL with hard parameter sharing to add two auxiliary tasks—binary classification to identify entity spans (segmentation) and multi-class classification to type them without CRF. Both tasks use the same biLSTM representation as the target task. The output of the additional tasks is not used in the NER task; they only act as regularizers for NER. Others (Stratos, 2017; Aguilar et al., 2018) also use auxiliary tasks as regularizers but instead of binary classification, the additional tasks performs multi-class classification into B (first word of entity), I (remaining entity words) and O (non-entity).

The auxiliary tasks in MTL can also be used for extra supervision by concatenating their output label distribution to the representation used by NER (Xiao et al., 2019). Unlike prior work, Xiao et al. (2019) do not use the same representation for the target and auxiliary task. Instead, they build a sub-module called similarity-based auxiliary classifier (SAC). SAC takes as input the original input representation and adds token position embeddings (Vaswani et al., 2017), followed by multiple convolution layers. It maintains two randomly initialized vectors representing entity and non-entity classes. These vectors are combined with an attention layer to get the final word representations. The attention weights are calculated with the multiplicative attention function over the word representation and each of these two vectors. The final word representation is concatenated with the biLSTM representation for NER and the attention weights are used as proxy for probabilities of the word being an entity or not. The loss of both tasks is jointly optimized, with less weight given to entity segmentation over NER. Yu et al. (2018) also add extra supervision, but with a two-step approach. They learn two character-level language models for entity and non-entity words and use their output as a binary feature in NER.

Augenstein and Søgaard (2017) use MTL for NER in scientific texts, adding five auxiliary tasks. They add syntactic chunking, hyperlink prediction, multi-word expression identification, frame target annotation, and semantic super-sense tagging; the first three target segmentation. The auxiliary tasks are used one at a time with the target task. Since the datasets for NER and the auxiliary tasks are

Dataset	Genre	Labels
CoNLL	news	PER, LOC, ORG, MISC
Ontonotes	conversations	PERSON, ORG, LOC, EVENT, NORP, LANGUAGE, LAW, MONEY, DATE, TIME, WORK OF ART, PERCENT, QUANTITY, CARDINAL, ORDINAL
BTC	twitter	PER, LOC, ORG
TTC	twitter	PER, LOC, ORG

Table 2: Dataset Genre and Entity Labels

different, at each training step, a random task is chosen, followed by a random training instance.

6.2 Semi-Markov CRFs

Semi-Markov CRFs (Sarawagi and Cohen, 2004) are a variant of linear chain CRFs that capture dependencies between adjacent spans of text instead of adjacent words. The Markov assumption still holds across spans but not within the span. The goal is to find the best possible segmentation into spans using scores at span-level. The maximum length of spans is bound to reduce computation cost. Sarawagi and Cohen (2004) use hand-crafted features for span representations but recent work has explored other techniques to represent spans.

Gated recursive semi-markov CRF (Zhuo et al., 2016) creates a pyramid-shaped feature extractor for spans. The bottom-most layer consists of word representations and hence length one spans. Representation of adjacent words are combined to form length two spans for the next layer and so on. The top layer consists of a single span with the full sentence. Hybrid semi-Markov CRF or HSCRF (Ye and Ling, 2018) do not use an explicit span representation. Instead they consider the span-level score as sum of the word-level CRF scores of constituent words. Both the word-level and span-level CRF are jointly optimized. Sato et al. (2017) use a two step process to reduce the search space of the spans. They first generate possible spans from a separate model using a score cutoff and then find the best possible labelling over these spans instead of all spans upto a maximum specified length.

7 Datasets

We evaluate several of the above models on four datasets, to compare their performance. Table 2 shows the entity types in each dataset.

1. **CoNLL** is the English portion of the CoNLL'03 data (Tjong Kim Sang and De Meulder, 2003), extracted from the Reuters 1996 newswire corpus.
2. **ON** is the union of broadcast conversation (bc) and telephone conversation (tc) domains in the English portion of Ontonotes (Hovy et al., 2006).[2] The number of entity types is the largest in this dataset. We merge the closely related categories, *GPE* with *LOC* and *FAC* with *ORG*, to allow us to easily map gazetteer labels to dataset labels.
3. **BTC** or Broad Twitter Corpus (Derczynski et al., 2016) consists of tweets. We use the recommended train, validation and test splits.
4. **TTC** or Temporal Twitter Corpus (Rijhwani and Preotiuc-Pietro, 2020) also consists of tweets. It has multiple training splits from years ranging from 2014 to 2018 and validation and test splits from 2019. We use the 2014 training split as it overlaps less with 2019 and hence is more challenging.

Some dataset statistics are shown in Table 3. CoNLL and OntoNotes are the largest and are roughly equal in size. OntoNotes has more long entities and fewer entities that are sequences of capitalized words. Such characteristics will favor methods that perform better segmentation without relying on standard orthography. BTC and TTC are smaller and have more distinct surface forms. The entities in TTC follow capitalization conventions but BTC has many entities that consist of words that are not capitalized. BTC also exhibits a different distribution of capitalization patterns between the training and the test set. In the training set, roughly half of the entities are sequences of word with capitalized first letter but this number falls to just 28% in the test set. Most entities in all datasets consist of a single words. Only about 2% of entities have length more than three. OntoNotes contains the largest percentage of long entities, 10% of all.

We also present statistics on in/out of vocabulary words in the test data, with respect to the pretraining data, the training data and the gazetteer entries. An entity is considered *seen* in pre-training if the phrase is seen as such in the pre-training corpus or all of the constituent words are seen. For CoNLL and OntoNotes, almost all entities are seen in the pretraining data (>90%), followed by TTC (88%). For BTC, only 65-75% entities are seen in pretraining. Adding ELMo increases the entity coverage by only a small amount in all datasets.

---

[2] We also experimented with newswire, broadcast news and magazines, which had similar results to CoNLL.

System	**CoNLL**		**ON**		**BTC**		**TTC**	
	train	test	train	test	train	test	train	test
Entities (spans)	23326	5613	10419	1972	8774	4376	1175	1539
Distinct surface forms (%)	35.49	48.03	37.98	44.62	55.20	59.53	83.83	75.18
Dataset Entities by length in words(%)								
1	62.60	62.94	57.49	47.16	75.88	87.27	68.51	64.72
2	31.51	31.32	22.38	27.74	19.49	10.03	24.94	30.54
3	4.17	4.56	10.62	13.74	2.89	1.97	4.26	3.44
≥ 4	1 .70	1.17	9.51	11.36	1.72	0.73	2.32	1.28
Seen entities (%)								
Pretraining (GloVe)	92.96	92.25	97.13	97.77	75.21	65.40	84.26	87.91
Pretraining (GloVe+ELMo)	95.04	94.53	97.83	98.17	75.92	66.06	84.77	88.56
Training, any type	100.00	62.62	100.00	74.80	100.00	45.70	100.00	22.29
Gazetteer, any type	82.52	82.31	92.73	94.93	65.56	59.32	68.34	79.08
Gaz, not train, any type	0.00	26.55	0.00	21.86	0.00	14.79	0.00	57.76
Training, correct type	100.00	59.43	100.00	69.78	100.00	44.42	100.00	18.91
Gazetteer, correct type	74.34	73.97	67.22	69.78	50.24	32.38	64.09	75.37
Casing								
Title case (%)	82.60	80.76	58.13	56.09	49.50	28.88	67.66	78.17

Table 3: Dataset Statistics. CoNLL and Ontonotes are larger but TTC and BTC have more distinct surface forms. Except Ontonotes, all datasets have very few long entities. Most entities are seen in the pre-training data. Test entities seen in the training data are generally seen with the same type at least. BTC has many lowercase entities.

The gazetteer provides a better coverage than the training data. Training data coverage is especially low in TTC, making the dataset more challenging. According to our definition of seen, entities may be seen in the training data but not necessarily with the expected types. We also check if they are seen in the training data with the same type as in the test data. There is a decrease of only 1-5% when taking the type into consideration. Overwhelming, the type in training is also that in testing.

8 Experiments

Results for models using different word representations, without any gazetteer or segmentation features, are shown in Table 4. We report micro-F1 over all entity types, averaged over three runs. On all datasets, the combination of GloVe, ELMo and character-based representations works best. Given these results, it would have been reasonable to study methods of adding gazetteers only for this representation. However, given that the GloVe along with character-based embeddings is much more commonly used in recent work on NER, we also present results for that.

We now compare one representative approach from each class of methods for adding gazetteer and segmentation features to the biLSTM-CRF architecture. For a fair comparison, we add only the core idea of the model and use the same hyperparameters, noted in the appendix, removing different peripheral features such as part-of-speech tags and word shape, used in papers that introduced the idea.

System	**CoNLL**	**ON**	**BTC**	**TTC**
G	89.55	77.66	72.04	50.11
G+ch	90.64	77.95	72.75	57.17
E	91.02	79.93	73.33	63.87
E+G+ch	**91.74**	**80.67**	**73.84**	**64.88**

Table 4: NER F1 of models with varying input representation to biLSTM-CRF. G refer to GloVe, E refers to ELMo and ch refers to character-based representation. The highest value in each column is boldfaced.

8.1 Gazetteers

We compare four gazetteer-derived features. In each case, the gazetteer representation vector is passed through a feedforward layer with 32 neurons and ReLU activation and then concatenated to the input word representation to the biLSTM-CRF.

1. **WORD_GAZ** We map gazetteer entity types to dataset types and split gazetteer entries into words using space as a delimiter, to create a vocabulary associated with each entity types, i.e. a list of words that appeared in person names etc. Each word in a sentence is associated with a binary valued vector with length equal to the number of entity types in the dataset. The dimension corresponding to a given type gets value one if the word is in the gazetteer vocabulary for that type and zero otherwise. The vector is all zeros for words that do not appear in the vocabulary for any entity types and can have multiple components with value one, when the word appeared in gazetteer entries of more than one type.

System	CoNLL	ON	BTC	TTC	avg diff	max diff
G+ch	90.64	77.95	72.75	57.17	-	-
WORD_GAZ	90.56	78.21	73.57	58.42	0.56	1.25
GAZ_IOBES	90.44	77.83	73.70	59.70	0.79	2.53
GAZ_FINE	90.52	**79.01**	**73.53**	**58.82**	**0.84**	1.65
PHRASE_GAZ	90.29	77.32	72.73	59.18	0.25	2.01
LRN_GAZ	89.82	78.55	72.91	59.36	0.53	2.19
SEG_REG	90.28	78.46	72.74	56.78	-0.06	0.51
SEG_SUP	90.20	**78.34**	**72.85**	**58.54**	**0.35**	1.37
SAC	90.35	77.33	72.83	57.04	-0.24	0.08
HSCRF	89.91	78.42	71.75	55.34	-0.77	0.47

Table 5: NER F1 on using GloVe+char. *avg-diff* and *max-diff* are the average and maximum increase over the base model across datasets. The most stable system (highest avg-diff) in each category is boldfaced.

System	CoNLL	ON	BTC	TTC	avg diff	max diff
E+G+ch	91.74	80.67	73.84	64.88	-	-
WORD_GAZ	91.74	**80.86**	**74.70**	**66.22**	**0.60**	1.34
GAZ_IOBES	91.76	80.66	72.96	64.60	-0.29	0.02
GAZ_FINE	91.92	80.47	75.03	65.23	0.38	1.19
PHRASE_GAZ	92.08	80.52	74.00	64.19	-0.08	0.34
LRN_GAZ	91.32	81.27	73.98	65.75	0.30	0.87
SEG_REG	91.79	80.99	74.42	63.52	-0.10	0.58
SEG_SUP	91.81	80.66	74.97	65.51	0.46	1.13
SAC	91.87	80.20	75.27	62.57	-0.30	1.43
HSCRF	91.66	80.62	**74.64**	**66.92**	**0.68**	2.04

Table 6: NER F1 using ELMo+GloVe+char. *avg-diff* and *max-diff* are the average and maximum increase over the base model across datasets. The most stable system (highest avg-diff) in each category is boldfaced.

# gaz types	CoNLL	ON	BTC	TTC
0	61.9	59.2	74.5	62.8
1	15.7	13.9	10.1	14.3
1+	21.7	26.9	6.3	9.0
all	0.7	0.0	9.1	13.8

Table 7: Percentage of words in the vocabulary with the number of gazetteer types in which they appear.

2. **GAZ_IOBES** is similar to WORD_GAZ but instead of a single bit, there are four values for each label denoting a matching to an entity exactly (S), to the first word (B), to the last word (E) or other words in between (I).
3. **GAZ_FINE** is also similar to WORD_GAZ but instead of selecting types from the gazetteer to match the label types for a given dataset, we use all 30 gazetteer types as is. The length of the gazetteer vector equals the numbers of the original gazetteer types without any mapping.
4. **PHRASE_GAZ** Here, we perform phrase level matching between the sentence and the gazetteer. All subsequences (continuous sequence of words) in a sentence are matched to each entry in the gazetteers, retaining the gazetteer type only for the longest match. For example, only 'New York University' is matched, not the nested 'New York' entity in it. Each word in the longest match subsequence gets the bit corresponding to the matched gazetteer category switched on in the gazetteer vector representation.
5. **LRN_GAZ** Gazetteer embeddings are learned with a separate model and concatenated to the input representation for the NER model, as in Magnolini et al. (2019). Random n-grams, excluding named entities, from CoNLL'03 training data are used as negative examples for training the gazetteer network, as opposed to negative examples generated from gazetteer entries used in Magnolini et al. (2019).

Both word-level matching and learned gazetteers make it possible for the model to learn phrases beyond the exact entries in the gazetteers. For example, if the organization gazetteer contains 'GAZ High School' but the dataset contains 'TST High School', phrase-level matching would not match any of the words in 'TST High School' to a gazetteer. A word-level matching would at least mark 'High' and 'School' as organizations. Similarly, learned gazetteer embeddings may learn a similar representation for both and make it possible to recognize 'TST High School' correctly.

8.2 Segmentation

We experiment with the different multi-task learning methods of incorporating segmentation and one of the semi-Markov CRF models.

1. **SEG_REG** (Aguilar et al., 2017) is the MTL approach with two auxiliary tasks: binary classification to predict if the word is a part of entity and multi-class classification to predict the type of the word without CRF.
2. **SEG_SUP** is the same as SEG_REG, but the label distribution scores from the binary classification auxiliary task are concatenated to the final representation used for NER.
3. **SAC** is the similarity-based auxiliary classifier in Xiao et al. (2019) as described above.
4. **HSCRF** is the hybrid semi-Markov CRF in Ye and Ling (2018) as described above

System	CoNLL	ON	BTC	TTC	avg diff	max diff
G+ch	94.84	83.24	85.34	67.74	-	-
WORD_GAZ	94.86	83.02	86.60	69.20	0.63	1.46
GAZ_IOBES	94.90	83.03	86.50	70.15	0.86	2.41
GAZ_FINE	**94.85**	**83.67**	**86.07**	**70.17**	**0.90**	2.43
PHRASE_GAZ	94.74	82.26	86.00	69.64	0.37	1.90
LRN_GAZ	94.44	83.65	85.70	68.97	0.40	1.23
SEG_REG	94.80	83.66	85.75	66.90	-0.01	0.42
SEG_SUP	94.71	83.16	**86.20**	**69.98**	**0.72**	2.24
SAC	94.79	82.74	86.93	69.25	-0.42	-0.19
HSCRF	94.55	82.11	85.15	67.03	0.48	1.59

Table 8: Entity segmentation F1 using GloVe+char. *avg-diff* and *max-diff* are the average and maximum increase over the base model across datasets. The most stable system (highest avg-diff) in each category is boldfaced.

System	CoNLL	ON	BTC	TTC	avg diff	max diff
E+G+ch	95.48	85.01	85.4	73.59	-	-
WORD_GAZ	95.32	**85.17**	**86.67**	**74.29**	**0.49**	1.27
GAZ_IOBES	95.54	84.77	85.89	73.64	0.09	0.49
GAZ_FINE	95.51	84.61	86.55	74.10	0.32	1.15
PHRASE_GAZ	95.54	84.64	85.84	73.17	-0.07	0.44
LRN_GAZ	95.08	85.18	86.07	74.52	0.34	0.93
SEG_REG	95.60	85.08	86.46	72.88	0.13	1.06
SEG_SUP	95.51	84.76	86.38	74.57	0.43	0.98
SAC	95.37	84.38	86.93	71.70	-0.28	1.53
HSCRF	95.37	84.73	**86.38**	**76.63**	**0.91**	3.04

Table 9: Entity Segmentation F1 using ELMo+GloVe+char. *avg-diff* and *max-diff* are the average and maximum increase over the base model across datasets. The most stable system (highest avg-diff) in each category is boldfaced.

9 Results

We test the gazetteer-enhanced and segmentation approaches using both GloVe+char and ELMo+GloVe+char as the input representations. Results are shown in Tables 5 and 6 respectively.

9.1 Consistency Across Datasets

We report the average and maximum improvement in F1-score over the base model across datasets. A high average improvement means that the model is consistently better across datasets spanning newswire, conversations and Twitter posts. A high maximum improvement with a low or negative average improvements means that the model can do well on some dataset but fails to perform well when tested on multiple datasets of varied genres.

The word indicator gazetteer features are the best and most consistent on average. With ELMo+GloVe, they also show the maximum improvement. WORD_GAZ combines gazetteer labels to dataset labels whereas the GAZ_FINE does not do any dataset specific mapping. Prior work has used the former but this experiment shows that we do not need to do such dataset specific modifications of gazetteer labels. We can use fine-grained labels and obtain similar gains. PHRASE_GAZ and LRN_GAZ improve performance with GloVe+char, with especially high performance on TTC, but they are not as good with ELMo+GloVe+char. Improvement with both representation by incorporating gazetteers is higher on BTC and TTC than CoNLL and OntoNotes. This is likely because the gazetteer features are more ambiguous in CoNLL and OntoNotes (Table 7), with most words appearing in two or more possible categories.

Segmentation methods can show vast improvement on specific datasets but they are unstable across datasets. The average improvement is *negative* in many cases for both representations. SEG_SUP is the only segmentation approach that is consistently beneficial across datasets for both representations. HSCRF performs well too, but only when ELMo is used as well. In comparison, including gazetteers is consistently better.

9.2 Gazetteers for Segmentation

In this section, we explore if gazetteers can be used as an alternative for segmentation, owing to their stable performance across datasets. We ask the following question:Are gazetteers recognizing new entities not previously marked as entities or are they improving the typing of spans already recognized as entities by adding entity type information? To answer this question, we report the entity segmentation F1 in Tables 8 and 9. Entity segmentation is the task of finding the correct entity span, regardless of the type. Since pre-training data covers most entities (Table 3), one would expect gazetteers to improve typing of entities through a more explicit type signal. However, segmentation results are consistent with those of NER. The models that improved performance of NER also perform well on segmentation and often by similar margins as NER. In fact, following similar trends as NER, gazetteer-enhanced models are more consistent than segmentation methods at entity segmentation as well. Since gazetteer-enhanced models induce segmentation and improve performance despite a near perfect coverage provided by the pre-training data, we expect them to improve performance of even the latest transformer-based models

System	CoNLL				Ontonotes				BTC				TTC			
	1	2	3	≥ 4	1	2	3	≥ 4	1	2	3	≥ 4	1	2	3	≥ 4
E+G+ch	92.1	92.0	**85.8**	**85.4**	**87.0**	78.7	64.4	74.3	74.2	**75.5**	57.6	41.3	59.9	77.6	54.0	38.8
WORD_GAZ	92.1	92.0	85.5	85.0	**87.0**	78.4	**65.6**	**75.6**	75.2	75.2	**58.0**	**44.6**	61.5	78.3	**54.5**	35.8
HSCRF	**92.2**	**92.2**	83.8	78.9	86.8	**79.3**	63.5	73.7	**75.3**	75.4	53.2	36.9	**62.1**	**79.4**	50.7	**47.2**

Table 10: NER F1 by entity length (words) using ELMo+GloVe+char on the most stable gazetteer and segmentation method. The highest value in each column is bolfaced. Generally, gazetteers perform better on long entities and segmentation methods perform better on short entities.

System	CoNLL	ON	BTC	TTC	avg diff	max diff
G+ch	90.64	77.95	72.75	57.17	-	-
WORD_GAZ	90.56	78.21	73.57	58.42	0.56	1.25
- gaz types	90.64	78.69	74.13	59.16	1.03	1.99
E+G+ch	91.74	80.67	73.84	64.88	-	-
WORD_GAZ	91.74	80.86	74.70	66.22	0.60	1.34
- gaz types	91.58	80.31	74.75	65.16	0.17	0.91

Table 11: NER F1. *avg-diff* and *max-diff* are the average and maximum increase over the base model across datasets. Removing gazetteer typ information improves performance for GloVe+char.

(Devlin et al., 2019) pre-trained on large corpora.

Next, we modify WORD_GAZ to mark the presence in any gazetteer without taking the type into consideration (Table 11). Surprisingly, removing type information results in better performance with GloVe, indicating the most benefit came from segmentation and not typing. With ELMo included, however, we do not observe the same trend. While performance is better than the baseline system, it is not better than WORD_GAZ that includes types. Though we cannot point conclusively to the reason behind such results with ELMo, we suspect it is due to the ambiguity within the pre-training data. The model may not be using the gazetteer representation if there is a strong signal from the pretrained representation that the word is not an entity. This can only be verified if all representations are trained on the same pre-training corpus.

9.3 Gazetteers for Long Entities

To further verify the effectiveness of gazetteers for segmentation, we break down performance by *entity length* in words, reporting F1 in Table 10 for the top performing gazetteer-enhanced model and segmentation model with ELMo+GloVe+char. Recall that the majority of entities in all corpora are of length one and that entities consisting of more than three words are the rare, about 2% in three of the datasets, expect in OntoNotes, where they are 10%. The highest performance is on entities of length two, followed by length one. Longer entities are recognized much more more poorly.

Typically segmentation methods have been used to improve performance on long entities. But our experiments reveal that gazetteers are better at it. With the exception of TTC, WORD_GAZ is better on long entities and HSCRF on shorter ones. This is likely due to the presence of many long entities in gazetteers (§4, Table 1).

10 Conclusion

We provided a comprehensive overview of methods for incorporating gazetteers and inducing segmentation in NER. We chose these two areas because even though they have been explored separately in prior work, we find that they are interrelated and achieve similar goals. We implemented representative models from each category for a fair comparison. We found that while segmentation methods can achieve impressive improvements on specific datasets, gazetteer-enhanced models are more stable across datasets. Moreover, the simpler methods of gazetteer enhancement (binary valued discrete feature vector with word-level gazetteer matching) and segmentation (multi-task learning with a extra supervision from and auxiliary binary classification for segmentation) performed better within their respective categories.

Furthermore, contrary to expectation, we found that gazetteer-enhanced models improve entity segmentation, not just entity typing. In fact, one need not perform a gazetteer to dataset label mapping for incorporating gazetteers; using the original gazetteer types works just as well. Even more surprisingly, gazetteer types are even unnecessary depending on the input representation. With GloVe, performance improves by removing gazetteer types altogether. This is likely a consequence of gazetteers inducing segmentation. Lastly, we showed that gazetteers are better at finding long entities, another consequence of inducing segmentation. They are an effective alternative to segmentation techniques developed to identify long entities, which we found are unstable across datasets.

References

Oshin Agarwal, Yinfei Yang, Byron C. Wallace, and A. Nenkova. 2020. Entity-switched datasets: An approach to auditing the in-domain robustness of named entity recognition models. *ArXiv*, abs/2004.04123.

Oshin Agarwal, Yinfei Yang, Byron C. Wallace, and Ani Nenkova. 2021. Interpretability Analysis for Named Entity Recognition to Understand System Predictions and How They Can Improve. *Computational Linguistics*, 47(1):117–140.

Gustavo Aguilar, Adrian Pastor López-Monroy, Fabio González, and Thamar Solorio. 2018. Modeling noisiness to recognize named entities using multitask neural networks on social media. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1401–1412, New Orleans, Louisiana. Association for Computational Linguistics.

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López-Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 148–153, Copenhagen, Denmark. Association for Computational Linguistics.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017a. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017b. Generalisation in named entity recognition: A quantitative analysis. *ArXiv*, abs/1701.02877.

Isabelle Augenstein and Anders Søgaard. 2017. Multitask learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346, Vancouver, Canada. Association for Computational Linguistics.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 148–151.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Colin Cherry and Hongyu Guo. 2015. The unreasonable effectiveness of word representations for Twitter named entity recognition. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 735–745, Denver, Colorado. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using similarity measures to select pretraining data for NER. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1460–1470, Minneapolis, Minnesota. Association for Computational Linguistics.

Leon Derczynski, Kalina Bontcheva, and Ian Roberts. 2016. Broad Twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, Osaka, Japan. The COLING 2016 Organizing Committee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Doug Downey, Matthew Broadhead, and Oren Etzioni. 2007. Locating complex named entities in web text. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 2733–2739, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

JinLan Fu, Peng fei Liu, Qi Zhang, and Xuanjing Huang. 2020a. Rethinking generalization of neural models: A named entity recognition case study. *ArXiv*, abs/2001.03844.

Jinlan Fu, Pengfei Liu, and Graham Neubig. 2020b. Interpretable multi-dataset evaluation for named entity recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6058–6069, Online. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, pages 466–471.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):1–17.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019a. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307, Florence, Italy. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019b. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging lexical resources for learning entity embeddings in multi-relational data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 112–117, Berlin, Germany. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Simone Magnolini, Valerio Piccioni, Vevake Balaraman, Marco Guerini, and Bernardo Magnini. 2019. How to use gazetteers for entity recognition with neural models. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 40–49, Macau, China. Association for Computational Linguistics.

Xue Mengge, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-Fine Pre-training for Named Entity Recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6345–6354, Online. Association for Computational Linguistics.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–8, Bergen, Norway. Association for Computational Linguistics.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 337–342, Boston, Massachusetts, USA. Association for Computational Linguistics.

Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 443–450, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Shruti Rijhwani and Daniel Preotiuc-Pietro. 2020. Temporally-informed analysis of named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7605–7617, Online. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. *Advances in neural information processing systems*, 17:1185–1192.

Motoki Sato, Hiroyuki Shindo, Ikuya Yamada, and Yuji Matsumoto. 2017. Segment-level neural conditional random fields for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 97–102, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. 2018. A study of the importance of external knowledge in the named entity recognition task. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 241–246, Melbourne, Australia. Association for Computational Linguistics.

Chan Hee Song, Dawn Lawrie, Tim Finin, and James Mayfield. 2020. Improving neural named entity recognition with gazetteers. *arXiv preprint arXiv:2003.03072*.

Karl Stratos. 2017. Entity identification as multitasking. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 7–11, Copenhagen, Denmark. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Shiyuan Xiao, Yuanxin Ouyang, Wenge Rong, Jianxin Yang, and Zhang Xiong. 2019. Similarity based auxiliary classifier for named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1140–1149, Hong Kong, China. Association for Computational Linguistics.

Eunsuk Yang, Young-Bum Kim, Ruhi Sarikaya, and Yu-Seop Kim. 2016. Drop-out conditional random fields for Twitter with huge mined gazetteer. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 282–288, San Diego, California. Association for Computational Linguistics.

Zhixiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-Markov CRF for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 235–240, Melbourne, Australia. Association for Computational Linguistics.

Xiaodong Yu, Stephen Mayhew, Mark Sammons, and Dan Roth. 2018. On the strength of character language models for multilingual named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3073–3077, Brussels, Belgium. Association for Computational Linguistics.

Jingwei Zhuo, Yong Cao, Jun Zhu, Bo Zhang, and Zaiqing Nie. 2016. Segment-level sequence modeling using gated recursive semi-Markov conditional random fields. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1413–1423, Berlin, Germany. Association for Computational Linguistics.

A Dataset Label to Gazetteer Mapping

CoNLL

- *PER*: People
- *LOC*: Locations, Locations.Generic, Parks
- *ORG*: Organization, PoliticalParties, Corporations, Government
- *MISC*: EthnicGroups, Languages, Vehicles, Nationalities

Ontonotes

- *PERSON*: People
- *LOC*: Locations, Locations.Generic, Parks
- *ORG*: Organization, PoliticalParties, Corporations, Government
- *DATE*: Temporal
- *TIME*: Temporal
- *CARDINAL*: NumberCardinal
- *ORDINAL*: NumberOrdinal
- *EVENT*: CompetitionsBattlesEvents
- *LANGUAGE*: Languages
- *MONEY*: Currency
- *NORP*: Nationalities, EthnicGroups
- *QUANTITY*: Measurements
- *WORK_OF_ART*: TV.Programs, ArtWork, Films

BTC and TTC

- *PER*: People
- *LOC*: Locations, Locations.Generic, Parks
- *ORG*: Organization, PoliticalParties, Corporations, Government

B Implementation

We use the implementation in `https://github.com/guillaumegenthial/tf_ner` for most models.

SAC uses https://github.com/XiaoShiyuan/NCRF-SAC but is based off the same codebase.

HSCRF uses `https://github.com/ZhixiuYe/HSCRF-pytorch` but ELMo was added additionally by us.

C Hyperparameters

Hyperparameters used as same as the ones in `https://github.com/guillaumegenthial/tf_ner/models/chars_conv_lstm_crf`, except for number of epochs and the minimum number of steps before early stopping which depend on dataset size. Numbers of epochs used were 25, 25, 50 and 50 for CoNLL, Ontonotes, BTC and TTC respectively. Minimum steps were 8000, 15000, 2500 and 2500 for CoNLL, Ontonotes, BTC and TTC respectively.

On the Cost-Effectiveness of Stacking of Neural and Non-Neural Methods for Text Classification: Scenarios and Performance Prediction

Christian Gomes
UFMG - Brazil
christianreis@dcc.ufmg.br

Marcos André Gonçalves
UFMG - Brazil
mgoncalv@dcc.ufmg.br

Leonardo Rocha
UFSJ - Brazil
lcrocha@ufsj.edu.br

Sergio Canuto
IFG - Brazil
sergio.canuto@ifg.edu.br

Abstract

Neural network algorithms such as those based on transformers and attention models have excelled on Automatic Text Classification (ATC) tasks. However, such enhanced performance comes at high computational costs. Ensembles of simpler classifiers (i.e., stacking) that exploit algorithmic and representational complementarities have also been shown to produce top-notch performance in ATC, enjoying high effectiveness and potentially lower computational costs. In this context, we present the first and largest comparative study to exploit the cost-effectiveness of stacking of ATC classifiers, consisting of transformers and non-neural algorithms. In particular, we are interested in answering research questions such as: Is it possible to obtain an effective ensemble with significantly less computational cost than the best learning model for a given dataset? Disregarding the computational cost, can an ensemble improve the effectiveness of the best learning model? Besides answering such questions, another main contribution of this paper is the proposal of a low-cost oracle-based method that can predict the best ensemble in each scenario (with and without computational cost limitations) using only a fraction of the available training data.

1 Introduction

Natural Language Processing, Machine Learning and Data Mining techniques work together to automate the fundamental task of Automatic Text Classification (ATC). ATC automatically associates documents with classes, providing means to organize information, allowing better comprehension and interpretation of the data. Algorithms based on neural networks (e.g., BERT (Devlin et al., 2018), XLNet (Yang et al., 2019)) have become the highlight in the area, where they are used both to learn features for text representation and as classification algorithms. The main problem of such methods is the very high computational costs needed for learning the model parameters (Sun et al., 2019; Cunha et al., 2021).

Ensemble approaches, such as stacking, which combine the outputs of several base classification models to form an integrated output, have also been shown to excel in ATC (Džeroski and Ženko, 2004; Ding and Wu, 2020), enjoying high effectiveness and computational costs that depend on the selected learning methods of the ensemble. They are motivated by the fact that distinct learning models or text representations may complement each other, uncovering specific structures that underlie the input/output relationship of the data. Early works (Larkey and Croft, 1996) aimed at showing combinations of different classification algorithms capable of producing better effectiveness results than any single type of classifier.

However, the benefits of ensemble techniques against a strong classifier are not always clear (Yan-Shi Dong and Ke-Song Han, 2004), in part, due to the excellent generalization power of the best classifiers. In fact, previous ensemble works mostly focus on improving the overall classification effectiveness using the results of traditional classification algorithms (Campos et al., 2017; Ding and Wu, 2020), paying little or no attention to practical issues such as the execution time or which combination of efficient base algorithms can bring effective results at a lower cost.

Accordingly, our *first contribution* in this paper is a thorough study of the cost-effectiveness tradeoff of stacking techniques for text classification tasks. Rather than just evaluating the effectiveness of an ensemble of various recent and effective methods, including those based on transformers and attention models, we focus on the study of stackers capable of achieving a better compromise between low cost (or high efficiency) and high ef-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4003–4014
August 1–6, 2021.

fectiveness when compared to a single base model (i.e., the most effective single classifier in a given dataset). A wide range of comparative experiments with stacked ensemble models and state-of-the-art base algorithms are conducted on six datasets widely used in text classification. We seek answers based on empirical evidence to the following questions, considering the best learning model for each given dataset: *(RQ1): Is it possible to obtain an effective ensemble with significantly less computational time than the best learning model? (RQ2): Is it possible to improve the effectiveness of the best learning model using an ensemble without increasing the computational time? (RQ3): Disregarding the computational time, can an ensemble improve the effectiveness of the best learning model?* As far as we know, we are the first to investigate the cost-effectiveness trade-offs (Cunha et al., 2021) of stacking of neural and non-neural text classifiers from the described perspectives.

A second main contribution of our work is the proposal of a low-cost oracle-based method that can predict the best ensemble in each scenario (with and without computational cost limitations) using only a fraction of the available training data. Our **"Oracle"** first estimates the best base algorithm (which can be seen as a baseline for effectiveness) to perform an efficient greedy search of ensembles guided by both their effectiveness and efficiency concerning the best base algorithm. Particularly, the Oracle predicts effective ensembles by successively including base algorithms that improve their combined majority voting effectiveness. Moreover, our method avoids the inclusion of expensive base algorithms (concerning the best base algorithm) to guarantee the ensemble efficiency. Our proposed Oracle is the first known strategy to provide an efficient prediction of effective ensembles capable of tackling practical efficiency issues related to our research questions. In more details, our proposal aims at predicting three ensembles corresponding to the time restrictions of RQ1, RQ2 and RQ3, respectively, while avoiding the potential high computational cost of evaluating expensive base models and their ensembles, especially on large datasets.

Our experimental results show affirmative answers to our three research questions in most experiments. In most datasets, it is possible to obtain an ensemble of base algorithms that is as good as or better than the base algorithm, at a lower cost. In 5 out of 6 datasets it is possible to obtain an ensemble with statistically significant gains against the best algorithm with no increase in cost. Similarly, in 5 out of 6 datasets, our oracle provides as good as or better results than the best base algorithm with no increase in cost, providing empirical evidence to the practical benefits of the proposed oracle.

2 Background and Related Work

2.1 Text Classification Strategies

Early efforts in ATC focused on improving machine learning algorithms such as Naïve Bayes, kNN, Logistic Regression and SVM (Howard and Ruder, 2020) using a simple bag of (TFIDF weighted) words representation. Even with such simple document representation, the use of methods such as LinearSVM (Fan et al., 2008a) and XGBoost (Chen and Guestrin, 2016a) produced high effectiveness with and efficient convergence for large datasets (Fan et al., 2008a).

More recent strategies, such as metafeatures (Canuto et al., 2018) and neural networks (NNs) (Joulin et al., 2017; Tang et al., 2015a), exploit the training data to build more informative document representations. Particularly, strategies based on metafeatures (Canuto et al., 2018; Canuto et al., 2019b) extract information from more basic (bag-of-words) features to enhance the feature space by smartly exploiting a document's neighborhood. Strategies based on NNs enhance word representations (and thus documents) also exploiting the training data. FastText (Joulin et al., 2017) and PTE (Tang et al., 2015a), for instance, presented high effectiveness in comparison to (costly) deep learning approaches.

Considerable advances on deep learning for ATC were achieved by using pre-trained language models with fine-tuning (Howard and Ruder, 2018), mainly when combined with attention mechanisms (Kokkinos and Potamianos, 2017; Yang et al., 2016) and the parallelization benefits of transformers, better exemplified by BERT (Devlin et al., 2018). Following BERT's success, the recent XLNet network (Yang et al., 2019) proposes a new autoregressive formulation to improve the exploitation of contextual information. Though effective, the fine-tuning process of methods such as BERT and XLNet still takes substantial computational time, requiring powerful hardware (GPUs) (Sun et al., 2019). Such requirements might bring practical limitations for these solutions.

2.2 Stacking

Stacking (Wolpert, 1992) is a widely known ensemble technique that combines the predictions of heterogeneous algorithms (i.e., base algorithms) to improve effectiveness concerning these base algorithms. To implement stacking, we first need to train each base algorithm. With the trained models, we can make predictions in a different validation set, which was not used for training. With the saved models and the predictions in the validation set, a *metalayer* (another learning algorithm) is used to learn how to combine the predictions in the combination. Recent work reported high effectiveness with stacking for multiple ATC tasks, such as topic classification (Campos et al., 2017; Abuhaiba and Dawoud, 2017), sentiment analysis (Carvalho and Plastino, 2020; Onan et al., 2016) and multi-label classification (Xia et al., 2020; Weng et al., 2019). Particularly, stacking provided substantial effectiveness improvements on recently proposed decision-tree-based algorithms (Campos et al., 2017) and with methods trained on different representations (including word embeddings) (Carvalho and Plastino, 2020; Pelle et al., 2018; Onan et al., 2016).

A careful choice of base algorithms is necessary due to the potential degradation of the stacking effectiveness and efficiency. The literature reported low effectiveness on stacking due to overfitting issues with multiple base algorithms (Reid and Grudic, 2009; Ledezma et al., 2010). Previous works that optimize the choice of a subset of base algorithms (Ledezma et al., 2010; Gupta and Thakkar, 2014) focused on maximizing the ensemble effectiveness with no concern for efficiency. Stacking efficiency is usually attached to the most expensive method. In fact, (Hou et al., 2021) reportedly avoids the cost of using expensive deep learning methods by including gradient boosting base algorithms comparable to convolutional NNs.

In this work, we provide a thorough evaluation of the effectiveness and efficiency tradeoffs of stacking, i.e., we investigate whether there are combinations of algorithms that overcome (in both, efficiency and effectiveness) the best base ones in a given dataset. Our proposed oracle in turn is the first method to explicitly tackle a time-constrained stacking prediction goal by explicitly and efficiently exploiting the relationships between stacking and the best base algorithms.

3 Methodology

3.1 Time-Constrained Stacking

We aim to answer the following research questions: *(RQ1): Is it possible to obtain an effective ensemble with significantly less computational time than the best learning model? (RQ2): Is it possible to improve the effectiveness of the best learning model using an ensemble without increasing the computational time? (RQ3): Disregarding the computational time, can an ensemble improve the effectiveness of the best learning model?*

With RQ1 we aim to identify whether it is possible to obtain a stacking of (a subset of) base algorithms that is as effective or better than the best (i.e., most effective) base algorithm and takes strictly less computational time than the best base. Favorable evidence towards a positive answer is important to indicate the existence of cost-effective stacking solutions, especially if the best base algorithm is a costly strong/high generalization power baseline. RQ2 keeps the same effectiveness demands of RQ1, but considering the following relaxation on the time constraint: the parallel execution of the base models can take at most the same execution time as the best base algorithm. This time constraint allows the best base algorithm to be included in the stacking. With this, we intend to evaluate if effectiveness improvements are possible with the (time) cost of the best base algorithm as an upper limit. In RQ3 we remove all time constraints to obtain the best possible stacking regardless of cost. With RQ3 we want to evaluate the potential effectiveness improvements of an stacking over the best base algorithm, in exchange for additional (time) cost.

3.2 Oracle-Based Prediction of Stacking Performance

The proposed strategy is implemented as follows: (i) each base algorithm is trained with a reduced amount of the training set (e.g., 30%); (ii) we run an algorithm, called "Oracle" (Algorithm 1), which aims at finding the best combination of base algorithms with less training by a greedy strategy. First, we select the best base algorithm obtained with the reduced training to start the combination, where $\mathbb{A}$ is the set of all base algorithms executed with less training. For this, we use the $Best(\mathbb{A})$ function, which simply returns the best algorithm based on the validation set. For each iteration, the next best algorithm, as estimated in a validation set, is added and we verify whether the combined result presents a statistically significant improvement ($\alpha = 0.05$) in relation to the previous iteration. If positive, it is permanently included in the combination. The process continues until all base algorithms are considered. The strategy is greedy since it makes the best choice in the current iteration.

To perform the comparison and statistical tests in each iteration, we use a separated piece within the training set (validation) that is not contained in the smaller part used in training. Besides, as a meta-layer, we use a simple average, i.e., we simply add the probabilities of the predictions dividing it by the number of base algorithms. The meta-layer average is represented by the function $Avg(\mathbb{E})$ in the pseudocode, where $\mathbb{E} \subset \mathbb{A}$. As it is a simple meta-layer and not a learning algorithm, the cost can be considered insignificant in the choice process.

Algorithm 1 Oracle Algorithm

```
procedure ORACLE($\mathbb{A}$)
    $\mathbb{C} \leftarrow Best(\mathbb{A})$
    $\mathbb{S} \leftarrow \mathbb{A} - \mathbb{C}$
    while $\mathbb{S} \neq \emptyset$ do
        $X \leftarrow Best(\mathbb{S})$
        $\mathbb{E} \leftarrow \mathbb{C} + X$
        if $Avg(\mathbb{E}) > Avg(\mathbb{C})$ then
            $\mathbb{C} \leftarrow \mathbb{E}$
            $\mathbb{S} \leftarrow \mathbb{A} - \mathbb{C}$
        else
            $\mathbb{S} \leftarrow \mathbb{S} - X$
    return $\mathbb{C}$                                   ▷ Best combination
```

With the oracle defined, we raise the following research questions: *(ORQ1): Can we predict, using a fraction of the training data, an effective stacking that will tie or outperform the best learning model when trained with all the available training, at a smaller cost than that of the best model? (ORQ2): Can we make a similar prediction than in ORQ1, but now with cost smaller or at the maximum equal to that of the best model when trained with all training data? (ORQ3): with no constraints in time, can we predict a combination that will be better than the best learning algorithm in a dataset?*

4 Experiments

4.1 Experimental Setup

We consider the effectiveness and efficiency of the models on two large-scale ATC datasets (Zhang et al., 2015) (more than 100,000 documents) – AG's News (AGNews) and IMdB Reviews – and four mid-sized datasets very known in the ATC community – 20 Newsgroups (20NG), WebKB (WebKB), Reuters (Reut) and ACM Digital Library (ACM). Table 1 shows the datasets details.

In terms of classification (base) algorithms, we consider the LinearSVM (Fan et al., 2008b), kNN (Altman, 1992), LogisticRegression (Fan et al., 2008b), XGBoost (Chen and Guestrin, 2016b), XLNet (Yang et al., 2019) and BERT (Devlin et al., 2018). In terms of representations, beyond the traditional term-weighting alternatives (TFIDF), we consider distributional and other types of word embeddings, such as FastText (Joulin et al., 2016; Bojanowski et al., 2017) and PTE (Tang et al., 2015b), as well as recent representations based on MetaFeatures that have obtained state-of-the-art (SOTA) effectiveness in some of the experimented datasets (Canuto et al., 2019a, 2018, 2016; Cunha et al., 2020, 2021). Table 2 has a summary of the base algorithms and respective representations used in each of them.

Algorithm	Representation	ID	Algorithm	Representation	ID
LinearSVM	FastText	A	Logistic	FastText	J
	PTE	B	Regression	PTE	K
	TFIDF	C		TFIDF	L
	Metafeatures	D		Metafeatures	M
kNN	FastText	F	XGBoost	FastText	N
	PTE	G		PTE	O
	TFIDF	H		TFIDF	P
	Metafeatures	I		Metafeatures	Q
			XLNet	Raw Documents	R
			BERT	Raw Documents	S

Table 2: Base Algorithms and Representations IDs.

We run the stacking process with the following variants: all combinations of the same base algorithm with different representations, all combinations of different base algorithms with their best representations, and a combination that includes all the base algorithms. For example, we perform all possible combinations of LinearSVM with FastText, PTE, TFIDF and MetaFeatures, resulting in total of 11 combinations: $\binom{4}{2} + \binom{4}{3} + \binom{4}{4}$. This limitation of combinations has a main reason: all combinations of all algorithms and representations, 18 in our case, would result in an impractical number of possible combinations for execution: 262,125 experiments = $\binom{18}{2} + \binom{18}{3} + .. + \binom{18}{18}$.

An important observation is that we assume that the base algorithms can be run in parallel (e.g. different machines). Thus, a stacking or oracle combination has the execution time limited by the most costly base algorithm in the respective combination. Even if this assumption is not true and it is necessary to execute the base algorithms and combinations on one single machine, this would only

			Class Distribution						
Dataset	Size	# Feat.	# Classes	Minor Class	Median	Mean	Major Class	Avg Doc. Size (words)	Skewness
20NG	18,846	97,401	20	628	984	942	999	296	Balanced
ACM	24,897	48,867	11	63	2,041	2,263	6,562	65	Imbalanced
Reut	13,327	27,302	90	2	29	148	3964	171	Extremely Imbalanced
WebKB	8,199	23,047	7	137	926	1,171	3705	209	Imbalanced
AGNews	127,600	39,837	4	31,900	31,900	31,900	31,900	37	Balanced
IMdB Reviews	348,415	115,831	10	12,836	31,551	34,841	63,233	326	Imbalanced

Table 1: Datasets statistics.

aggravate the cost problem and allow an unfair comparison in our favor. Therefore, to avoid this unfair comparison, we maintain the assumption of parallel execution.

The experiments in the smaller datasets were executed using a 10-fold cross-validation procedure while in the larger we used 5 folds due to the cost of the procedure. The algorithms parameters were tuned using the Bayesian Optimization (Bergstra et al., 2015) approach with 10 iterations, with the 5-fold stratified strategy and the training set (nested cross-validation). In Table 14 in the Appendix, we have the values of each parameter that we optimize in the non-neural base algorithms. The parameters and pre-trained models for BERT and XLNet are also shown in the Appendix (Table 12). For the neural networks, we adopted the same parameters defined by the authors of their respective methods (Devlin et al., 2018; Yang et al., 2019). In our experiments, we adopt AWS EC2 instances to run and measure the execution time for both neural and non-neural algorithms. For the non-neural algorithms, we use the instance model *c5a.12xlarge*, which has 48 CPUs, 96GB of RAM (without GPU). For the neural algorithms, we use the instance model *p2.xlarge*, which has one NVIDIA K80 GPU (12 GB of memory), 4 CPUs and 61 GB of RAM.

We evaluate all methods, combined with different representations, with respect to classification effectiveness and training time. We assess classification effectiveness in the test partitions using MicroF1 and MacroF1 (Sokolova and Lapalme, 2009). While MicroF1 measures the classification effectiveness over all decisions, MacroF1 measures the classification effectiveness for each individual class, averaging them, being very important for skewed datasets. In addition to effectiveness, we also assess the cost of each method in terms of the training execution time aiming at analyzing the cost-effectiveness trade-offs for all methods. The metric is the overall time in seconds (average of folds). To compare the average test results on our cross-validation experiments, we assess the statistical significance employing the paired t-test with 95% confidence, which is strongly recommended over signed-rank tests for hypothesis testing on mean effectiveness and arguably robust to potential violations of the normality assumption in this context (Urbano et al., 2019; Hull, 1993).

4.2 Experimental Results

4.2.1 Stacking Results

Effectiveness and Time results for the base algorithms in each dataset are shown in Table 3. The results of these best base algorithms are considered in the next analyses. Results for RQ1, RQ2 and RQ3, in terms of MacroF1 for each dataset are shown in Tables 4, 5, and 6, respectively, while Figure 1, shows the analysis of the cost (time). For each dataset, the tables show the effectiveness (MacroF1) of the best base algorithm along with the stacking combination that best answered the respective research question (if any), the respective combination of methods (the letters refer to the index of algorithms described in Table 2), and finally, in the last column, (Most Costly) the most costly algorithm that entered in the combination, according to the constraints imposed by the question. We present only MacroF1 results due to space constraints and the fact that is it harder to improve them in the highly skewed scenario that occurs in most of the experimented datasets. However, we also consider MicroF1, whose results are summarized in Table 7.

In Table 4, which focuses on RQ1 that has a strong constraint in terms of cost (time), we can see that in 4 out of 6 datasets, it is possible to obtain a combination of classifiers (stacking) that is

Dataset	Metric	Algorithm A	B	C	D	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20NG	MicroF1	80.4	88.66	89.46	90.52	80.96	81.25	84.63	90.42	81.54	88.96	88.89	**90.73**	80.39	86.24	79.86	90.06	74.38	85.88
	MacroF1	79.96	88.34	89.2	90.32	80.45	80.46	84.06	90.27	81.09	88.69	88.65	**90.58**	79.94	85.9	79.35	89.9	73.97	85.43
	Time (S)	79	66	22	702	76	76	28	78	323	413	163	**467**	632	1,392	1,428	1,163	3,972	3,959
ACM	MicroF1	73.28	75.34	76.62	79.01	73.88	75.73	73.64	78.19	73.90	75.76	77.21	79.19	73.49	75.98	74.67	**78.98**	71.65	78.25
	MacroF1	60.64	62.57	66.88	68.66	62.47	61.37	59.25	65.22	61.95	63.19	68.43	69.11	60.93	61.82	64.12	**69.32**	60.58	65.56
	Time (S)	89	112	13	776	136	140	35	125	481	2057	201	922	512	1,239	530	**1,542**	4,153	3,482
Reut	MicroF1	66.57	66.07	67.19	78.85	68.53	68.96	69.41	75.19	65.64	66.55	66.80	76.99	64.68	63.70	65.92	**82.17**	72.72	72.72
	MacroF1	30.13	31.75	34.54	42.48	31.45	26.31	32.74	35.68	30.45	31.22	33.88	43.17	28.73	23.91	31.85	**47.37**	40.29	33.60
	Time (S)	120	150	38	3,427	39	38	12	163	4,434	4,316	2,026	15,185	2,363	2,633	1,482	**6,426**	3,537	3,897
WebKB	MicroF1	75.69	71.31	81.35	77.4	77.16	72.53	75.02	77.27	76.64	71.41	82.64	77.62	75.63	78.58	83.65	79.47	**84.60**	86.04
	MacroF1	65.81	58.19	71.45	65.29	69.17	59.25	58.76	64.72	68.25	58.66	74.67	66.19	65.54	66.21	74.33	69.23	**77.76**	66.41
	Time (S)	37	43	11	106	19	18	9	23	50	224	41	175	97	229	197	242	**2,210**	1,735
AGNews	MicroF1	89.52	91.95	91.08	91.03	89.32	91.25	90.35	91.55	89.23	91.66	91.87	91.29	89.17	92.23	91.97	91.69	91.99	**93.74**
	MacroF1	89.50	91.93	91.06	90.99	89.30	91.25	90.32	91.53	89.23	91.65	91.85	91.27	89.16	92.22	91.96	91.68	91.97	**93.74**
	Time (S)	2,308	4,198	115	1,099	10,433	12,707	2,097	4,373	8,434	881	2,180	3,364	5,243	7,019	1,151	2,769	33,206	**11,257**
IMdB Reviews	MicroF1	29.69	37.53	30.51	20.31	32.88	32.99	27.01	28.00	29.17	38.64	33.75	29.02	29.76	37.03	34.58	30.02	25.08	**39.00**
	MacroF1	24.96	27.43	26.10	12.00	28.12	25.39	19.50	19.62	26.55	32.35	28.19	20.31	26.39	30.72	27.96	20.93	16.26	**34.09**
	Time (S)	50,817	23,658	1,977	26,885	43,005	42,525	13,476	68,117	72,658	4,818	35,155	15,988	35,543	123,705	59,634	91,562	59,749	**34,426**

Table 3: Effectiveness, based on MicroF1 and MacroF1, and efficiency, based on execution time, for all classification algorithms and datasets considered in the our experiments. The algorithms are: (A) LinearSVM + FastText; (B) LinearSVM + PTE; (C) LinearSVM + TFIDF; (D) LinearSVM + TFIDF+Mf; (F) KNN + FastTex; (G) KNN + PTE; (H) KNN + TFIDF; (I) KNN + TFIDF+Mf; (J) LogisticRegression + FastTex; (K) LogisticRegression + PTE; (L) LogisticRegression + TFIDF; (M) LogisticRegression + TFIDF+Mf; (N) XGBoost + FastTex; (O) XGBoost + PTE; (P) XGBoost + TFIDF; (Q) XGBoost + TFIDF+Mf; (R) XLNet + Raw Documents; and (S) BERT + Raw Documents. The best results are highlighted in bold. When we identified statistically ties between two or more algorithms, we chose the one with lower execution time. For example, for the 20NG, we identified statistical ties for the algorithms M (LogisticRegression + TFIDF+Mf) and Q (XGBoost + TFIDF+Mf) and choose the M that presents lower execution time.

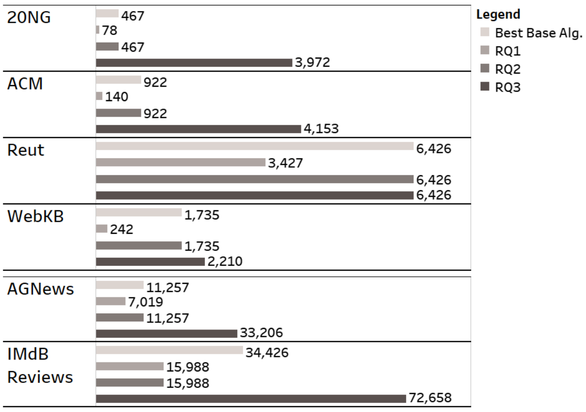

Figure 1: Stacking Times

as good as (statistical tie) or better (see the *ACM* case, with statistically significant gains of 3.1%) than the best base algorithm, at a lower cost. In fact, the gains in terms of cost (time) are very significant (see Figure 1), ranging from 1.87x speedup improvement (in Reuters) to 7.16x (in WebKB)[1]. Even if we consider the two cases in which there were some minimum effectiveness losses (0.39% in AGNews and 2.66% in IMdB), there are some significant speedups: 1.6x in AGNews and 2.15x in IMdB. Some chosen stacked combinations are interesting, such as FGHI in 20NG that contains all versions of kNN, and JLM in ACM, containing three versions of Logistic Regression. Both combinations contains classifiers with Metafeatures.

Results for RQ2 (Table 5) are also very interesting. In 5 out of 6 datasets it is possible to obtain

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	RQ1	● 91.02	FGHI	I
ACM	Best Base	69.32	Q (XGboost + MetaFeat)	
	RQ1	▲ 71.50	JLM	M
Reut	Best Base	47.37	Q (XGboost + MetaFeat)	
	RQ1	● 46.98	KL	K
WebKB	Best Base	77.76	R (XLNet)	
	RQ1	● 79.61	DIS	S
AGNews	Best Base	93.74	S (BERT)	
	RQ1	▼ 93.37	NOPQ	O
IMdB Reviews	Best Base	34.09	S (BERT)	
	RQ1	▼ 33.18	KM	M

Table 4: RQ1.

effectiveness gains with no increase in cost (remind that in this scenario the cost is limited by that of the best base algorithm). Effectiveness gains vary from 0.4% in AGNews, 1.15% in 20NG[2], 3.1% in ACM, 5.4% in IMdB and 9% in WebKB. Reuters is only considered a tie because of the high variability of the results across folds in this dataset (due to the large number of classes and very high skewness), which generates large standard deviations/confidence intervals. In absolute terms, there was a positive variation (non-statistically significant gain) of more than 9.7%. Indeed, the MicroF1 stacking results confirms statistically significant gains in Reuters (See Table 7).

As expected, to obtain gains in this scenario it is necessary to include the best base algorithm in the combination in most datasets, inserting diversity/complementarity into the combination. Only

---

[1] Speedups in 20NG and ACM were 6x and 6.6x

[2] Notice that improvements in 20NG and AGNews are very hard to obtain given the already high effectiveness values.

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	RQ2	▲ 91.63	JKLM	M
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	RQ2	▲ 71.50	JLM	M
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	RQ2	● 51.99	OPQ	Q
WebKB	Best Base	77.76	R (XLNet)	
	RQ2	▲ 84.07	All	R
AGNews	Best Base	93.74	S (BERT)	
	RQ2	▲ 94.12	DIQS	S
IMdB Reviews	Best Base	34.09	S (BERT)	
	RQ2	▲ 35.95	MS	S

Table 5: RQ2.

in ACM, the base algorithm is not part of the combination. Notice also that, due to the time constraints, the gains are somewhat limited due to the restricted number of classifiers that can be combined. This has some impacts on the results, for instance, in IMdB only two algorithms belong to the best combination while the combination in ACM has only three classifiers. Only in WebKB the combination includes all 18 classifiers as the base algorithm is also the most expensive one. Another interesting aspect of the combinations is that in all datasets, a classifier using Metafeatures was included (e.g., M and Q)

Finally, in the scenario with no time constraint (RQ3), further gains can be obtained with the inclusion of more costly classifiers. There are further gains in AGNews (0.94%), 20NG (2.06%), IMdB (5.8%) and ACM (6.32%). Notice that in this scenario, there is a tendency to include most algorithms in the combinations, like in ACM, WebKB and AGNews, to obtain further improvements. This means that most algorithms have complementary information that tends to contribute to the final results. Another interesting aspect to notice is that in some cases, such as in 20NG, a completely different combination than that chosen in scenario RQ2, was picked. This combination exploits the most effective and complementary algorithms, and may not even include the base classifier. In other cases, such as in IMdB, a combination of a few of the most effective (and costly) algorithms suffices to obtain larger gains. This means that the meta-layer is really doing a good job in learning about the individual performance of the algorithms and their complementarity. Finally, these additional effectiveness gains come with potential high increases in cost, clearly seen in Figure 1 for the cases of 20NG, ACM and AGNews. In those datasets, the costs have tripled (AGNews), quadrupled (ACM and IMdB) or become 8x more cost expensive. It is up to the application designer to decide whether this cost-effectiveness tradeoff is worth it.

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	RQ3	▲ 92.45	DIRS	R
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	RQ3	▲ 73.70	All	R
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	RQ3	● 51.99	OPQ	Q
WebKB	Best Base	77.76	R (XLNet)	
	RQ3	▲ 84.07	All	R
AGNews	Best Base	93.74	S (BERT)	
	RQ3	▲ 94.63	All	R
IMdB Reviews	Best Base	34.09	S (BERT)	
	RQ3	▲ 36.06	QS	Q

Table 6: RQ3.

Table 7 summarizes the effectiveness results. For RQ1, there are 8 win/ties out of 12 possibilities (6 datasets, two metrics). Remind that in this scenario ties are considered a good result due to the reduction in costs. For RQ2 and RQ3, there are 11 wins, only 1 tie (in Macro in Reuters) no losses at all. In terms of cost (Figure 1), significant reductions in scenario 1 (RQ1) can be obtained in all 6 datasets, with almost no loss (or minimal losses) in terms of effectiveness. For scenario 2 (RQ2), effectiveness gains can be obtained in almost all cases with no additional cost when compared to the cost of the base classifier. And for scenario 3 (RQ3) additional effectiveness gains can be obtained, but sometimes with a very high increase in cost.

RQ	MicroF1			MacroF1		
	Win	Tie	Loss	Win	Tie	Loss
RQ1	2	2	2	1	3	2
RQ2	6	0	0	5	1	0
RQ3	6	0	0	5	1	0

Table 7: Win/Tie/Loss Summary.

4.2.2 Oracle Results

MacroF1 results of the Greedy Oracle predictor are shown in Tables 8, 9 and 10. These results correspond to an Oracle that uses the results of the base algorithms trained with 30% of the training data and predicting in a different training data portion in a nested folded cross-validation procedure.

We start by answering ORQ1. Table 8 shows that in half of the cases we can perform a good prediction, i.e., one that predicts a combination of methods that will tie or outperform the best base algorithm when trained with all the available training data (100%). It is very important to stress that in a real situation we do not really know what will be the best algorithm when using all the training data nor its effectiveness. Indeed, with more data, there is a tendency for some algorithms, such as the transformers, to improve their effectiveness, but their good performance may not be predicted with few training data. Remind also that this is a very strict scenario: even if we can predict which will be the best base algorithm, we cannot use it in the combination given the time constraints of ORQ1.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	ORQ1	▲ 91.62	ABCDIL	I
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	ORQ1	▲ 71.85	ABCDIJKLM	D
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	ORQ1	▼ 32.55	ABDQR	Q
WebKB	Best Base	77.76	R (XLNet)	
	ORQ1	● 75.79	BCF	F
AGNews	Best Base	93.74	S (BERT)	
	ORQ1	▼ 93.16	BDGKOQ	B
IMdB Reviews	Best Base	34.09	S (BERT)	
	ORQ1	▼ 27.79	L	L

Table 8: Oracle ORQ1.

Given all these limitations, mainly that only the algorithms with a cost lower than the best base algorithm (with 30% of training) can be considered, it is impressive that we can make a prediction that will surpass in effectiveness the best base algorithms using 100% of training in 20NG and ACM and tie with it, being cheaper, in WebKB. But even in the case in which there were losses, some were minimal, like in AGNews with a loss of only 2.5% with potential gains in training time. Only in Reuters and IMdB there were significant MacroF1 losses, mainly due to the failure of predicting which would be the best base algorithm[3] and the impossibility of including the predicted best base algorithm in the combination.

When we are allowed to include the best-predicted algorithm in the stacking (scenario for ORQ2) results are even better – we can make a good prediction in 5 out 6 cases (2 wins and 3 ties). Notice that in this scenario we consider a tie as a good result. We interpret that being able to predict a combination that will tie with the best algorithm with 100% of training in a dataset, without knowing which one will this best, at a very lost cost (Figure 2), as an excellent result. Notice that the best results in this scenario (i.e., 20NG and ACM) are obtained when we can in fact predict what will be the best base algorithm with 100% of training. But even when we cannot predict, as in the case of WebKB and AGNews[4], we can find a combination of simpler (and potentially less expensive) algorithms that can tie with the best. Again, IMdB was the only case in which we could not make a good prediction exactly by the failure in predicting, with 30% of training, that BERT would be the best algorithm when all the training data is used.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	ORQ2	▲ 91.52	ABCDILM	M
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	ORQ2	▲ 71.56	ABDMQ	Q
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	ORQ2	● 47.37	DMQ	M
WebKB	Best Base	77.76	R (XNet)	
	ORQ2	● 78.22	BCFGIJ	J
AGNews	Best Base	93.74	S (BERT)	
	ORQ2	● 94.10	BCDGIKLMOPQR	R
IMdB Reviews	Best Base	34.09	S (BERT)	
	ORQ2	▼ 31.11	K	K

Table 9: Oracle ORQ2.

Finally, when no time constraints are imposed the oracle's prediction results are excellent: 4 wins, 1 tie and only one loss (in IMdB). This last loss is explained by the same reasons as in the previous scenario: the failure of predicting BERT as the future best algorithm. But even in this case, the prediction for using algorithm K: LogisticRegression with PTE as the sole combination (an unusual prediction) produced minimal losses: only 1.05% at a cost much smaller than using BERT. And in the case of Reuters, we obtain an absolute increase in MacroF1 values (6% increase), though not statistically significant due to the high variability.

When looking at the costs of making the predictions in each scenario (ORQ1, ORQ2, and ORQ3), shown in Figure 2, we can see that in all cases (but 20NG for ORQ3), the oracle's predictions times are much smaller, in many cases negligible[5], when compared to the time to run the base algorithm with 100% of training. Given the time constraints imposed by ORQ1 and ORQ2 and the fact even in the scenario for ORQ3, only a portion of the 18 available algorithms needed to be stacked (in most cases) to produce effectiveness gains, the advan-

---

[3] In case of Reuters, a XGBoost with Metafeatures (Q) and in IMdB, BERT (S).

[4] In both the best base algorithms with 100% were the neural transformers XLNet and BERT

[5] Some differences are in the orders of magnitude.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	ORQ3	▲ 92.17	ABCDILMS	M
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	ORQ3	▲ 73.48	ABDMQ	Q
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	ORQ3	● 50.18	DMQ	M
WebKB	Best Base	77.76	R (XLNet)	
	ORQ3	▲ 83.12	ABCFJLMNOPQR	J
AGNews	Best Base	93.74	S (BERT)	
	ORQ3	▲ 94.63	All	R
IMdB	Best Base	34.09	S (BERT)	
Reviews	ORQ3	▼ 33.73	K	K

Table 10: Oracle ORQ3.

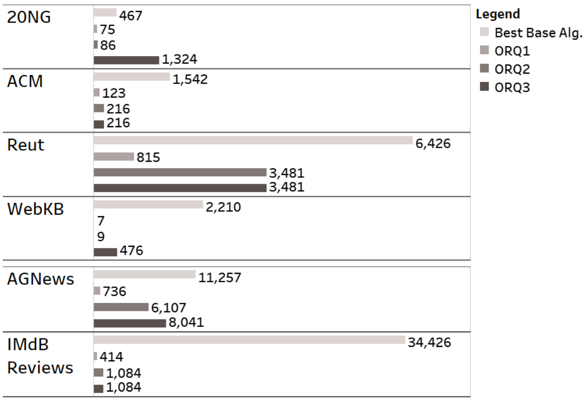

Figure 2: Oracle Times

tage's of running the oracle's predictions in terms of cost stand for themselves.

Table 11 summarizes the results in terms of Micro and MacroF1: considering all 36 results (three RQs, 6 datasets, 2 metrics) the oracle predicted 17 wins, 10 ties (most of them (8) in scenarios ORQ1 and ORQ2, which can be considered good results) and only 9 losses, six of them in a single dataset (IMdB) for the simple reason that we failed in predicting a neural network winner with fewer data. This is certainly a point to be improved in our methodology. One idea is to look not only at the absolute effectiveness values with a single training point (30%) but look also at the tendency of growing considering several points (5%, 10%, ..).

RQ	MicroF1			MacroF1		
	Win	Tie	Loss	Win	Tie	Loss
ORQ1	2	0	4	2	1	3
ORQ2	2	4	0	2	3	1
ORQ3	5	1	0	5	1	0

Table 11: Oracle Win/Tie/Loss Summary

5 Conclusion and Future Work

We presented two important contributions to the application of Stacking in ATC: a thorough study of cost-effectiveness trade-offs and the proposal of a new oracle method to predict the best ensemble combination for a dataset at a low cost. Our extensive experiments, composed of 4 textual representation methods, 6 datasets, 4 non-neural based algorithms and 2 neural-based algorithms, provided us with answers to questions that had not yet been explored in the literature. By performing stacking with different time constraints, we showed that it was possible to obtain combinations that positively answered the posed questions regarding the time-constrained stacking and the oracle predictions in terms of both, effectiveness and efficiency.

We highlight general and practical guidelines based on our extensive experiments. First, we notice the consistent appearance of recent meta-features on the best combinations of base learners obtained for each evaluated research question (Tables 4–6). In fact, due to the focus of meta-features on summarizing relevant distance-based information from the original features, we strongly suggest their exploitation in ensemble combinations. Moreover, the largest datasets benefit from additional data to fine tune BERT for the classification task. Therefore, combinations including both of these recent and distinct paradigms (meta-features and BERT) for stacking were able to produce very effective results on most datasets (as shown in Table 10). We suggest that stacking methods should start by exploiting these two paradigms in conjunction. Finally, our experiments show the need of specific stacking solutions for different scenarios/datasets. The application of our proposed Oracle efficiently predicts effective best base models on time-constrained scenarios, allowing adaptable solutions that automatically optimize the choice of base learners for each specific dataset. We suggest to exploit the Oracle in all these situations.

In the future, we will explore different Oracle configurations, explore multi-objective feature selection in the stacking meta-layer (Viegas et al., 2018), study other types of constraints (e.g., labeling effort) and apply the Oracle in fields such as recommender systems.

Acknowledgments

This work was supported by CNPq, CAPES, FAPEMIG, AWS, NVIDIA and Google (Awards).

References

Ibrahim S. I. Abuhaiba and Hassan M. Dawoud. 2017. Combining different approaches to improve arabic text documents classification. *International Journal of Intelligent Systems and Applications(IJISA)*, 9(4):39 – 52.

Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. 2015. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Raphael Campos, Sérgio Canuto, Thiago Salles, Clebson C.A. de Sá, and Marcos André Gonçalves. 2017. Stacking bagged and boosted forests for effective automated classification. In *Proceedings of the 40th ACM SIGIR Conference on Research and Development in Information Retrieval*, page 105–114.

S. Canuto, D. X. Sousa, M. A. Gonçalves, and T. C. Rosa. 2018. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Eng. (TKDE)*, 30(12):2242–2256.

Sérgio Canuto, Marcos André Gonçalves, and Fabrício Benevenuto. 2016. Exploiting new sentiment-based meta-level features for effective sentiment analysis. In *Proc. of the ninth ACM international conference on web search and data mining*, pages 53–62.

Sergio Canuto, Thiago Salles, Thierson C Rosa, and Marcos A Gonçalves. 2019a. Similarity-based synthetic document representations for meta-feature generation in text classification. In *Proc. of the 42nd ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 355–364.

Sergio Canuto, Thiago Salles, Thierson C. Rosa, and Marcos A. Gonçalves. 2019b. Similarity-based synthetic document representations for meta-feature generation in text classification. In *Proc. of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 355–364.

Sergio Canuto, Daniel Xavier Sousa, Marcos Andre Goncalves, and Thierson Couto Rosa. 2018. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2242–2256.

Jonnathan Carvalho and Alexandre Plastino. 2020. On the evaluation and combination of state-of-the-art features in twitter sentiment analysis. *Artificial Intelligence Review*.

Tianqi Chen and Carlos Guestrin. 2016a. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794.

Tianqi Chen and Carlos Guestrin. 2016b. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Washington Cunha, Sérgio D. Canuto, Felipe Viegas, Thiago Salles, Christian Gomes, Vítor Mangaravite, Elaine Resende, Thierson Rosa, Marcos André Gonçalves, and Leonardo C. da Rocha. 2020. Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling. *Inf. Process. Manag.*, 57(4):102263.

Washington Cunha, Vítor Mangaravite, Christian Gomes, Sérgio D. Canuto, Elaine Resende, Cecilia Nascimento, Felipe Viegas, Celso França, Wellington Santos Martins, Jussara M. Almeida, Thierson Rosa, Leonardo C. da Rocha, and Marcos André Gonçalves. 2021. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *Inf. Process. Manag.*, 58(3):102481.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Weimin Ding and Shengli Wu. 2020. A cross-entropy based stacking method in ensemble learning. *Journal of Intelligent & Fuzzy Systems*, pages 1–12.

Saso Džeroski and Bernard Ženko. 2004. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008a. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(61):1871–1874.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008b. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874.

A. Gupta and A. R. Thakkar. 2014. Optimization of stacking ensemble configuration based on various metahueristic algorithms. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 444–451.

Zhihao Hou, Kun Ma, Yufeng Wang, Jia Yu, Ke Ji, Zhenxiang Chen, and Ajith Abraham. 2021. Attention-based learning of self-media data for marketing intention detection. *Eng. Appl. Artif. Intell.*, 98:104118.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Jeremy Howard and Sebastian Ruder. 2020. A survey on text classification: From shallow to deep learning. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 31.

David Hull. 1993. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 586–591.

Leah S. Larkey and W. Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, page 289–297.

Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo. 2010. Ga-stacking: Evolutionary stacked generalization. *Intelligent Data Analysis*, 14:89–119. 1.

Aytug Onan, Serdar Korukoglu, and Hasan Bulut. 2016. Lda-based topic modelling in text sentiment classification: An empirical analysis. *Int. J. Comput. Linguistics Appl.*, 7(1):101–119.

Rogers Pelle, Cleber Alcântara, and Viviane P. Moreira. 2018. A classifier ensemble for offensive text detection. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, page 237–243.

Sam Reid and Greg Grudic. 2009. Regularized linear models in stacked generalization. In *Multiple Classifier Systems*, pages 112–121, Berlin, Heidelberg. Springer Berlin Heidelberg.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1165–1174, New York, NY, USA. Association for Computing Machinery.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015b. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1165–1174.

Julián Urbano, Harlley Lima, and Alan Hanjalic. 2019. Statistical significance testing in information retrieval: an empirical analysis of type i, type ii and type iii errors. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514.

Felipe Viegas, Leonardo C. da Rocha, Marcos André Gonçalves, Fernando Mourão, Giovanni Sá, Thiago Salles, Guilherme Andrade, and Isac Sandin. 2018. A genetic programming approach for feature selection in highly dimensional skewed data. *Neurocomputing*, 273:554–569.

W. Weng, C. Chen, S. Wu, Y. Li, and J. Wen. 2019. An efficient stacking model of multi-label classification based on pareto optimum. *IEEE Access*, 7:127427–127437.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241 – 259.

Yuelong Xia, Ke Chen, and Yun Yang. 2020. Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*.

Yan-Shi Dong and Ke-Song Han. 2004. A comparison of several ensemble methods for text categorization. In *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004*, pages 419–422.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.

Appendix

Algorithm	Parameters	Value
XLNet	Pretrained Model	XLNet-Base
	batch_size	32
	epochs	5
	max_len	64
	learning_rate	5e-5
	max_grad_norm	1.0
	weight_decay_rate	0.01
BERT	Pretrained Model	BERT-Base
	batch_size	32
	patience	5
	max_len	150
	initial_learning_rate	5e-5

Table 12: Neural networks parameters and pretrained models.

Method	Parameters	Value
TFIDF	Normalization	L2
	Stopwords	NLTK, English
	Max Features	Small Datasets: ∞
		Large Datasets: 50k
	MinDF	2
	Sublinear	TF
PTE	Window	5
	MinDF	2
	Dimensions	300
FastText	Window	5
	Epochs	500
	Model	Skipgram
	Dimensions	300
MetaFeatures	k	[10, 15, 20, 30, 35, 40, 45, 50]

Table 13: Text representations.

Algorithm	Parameters Tunned	Range Values
Linear SVM	C	uniform(0, 20)
	penalty	[l1, l2]
kNN	n_neighbors	range(1, 100, 1)
	metrics	[cosine, l1, l2, minkowski, euclidean]
	weights	[uniform, distance]
Logistic Regression	C	uniform(0, 20)
	penalty	[l2, None]
	solver	[newton-cg, lbfgs, sag, saga]
	class_weight	[None, balanced]
XGBoost	n_estimators	range(100, 1000, 50)
	learning_rate	quniform(0.01, 0.5, 0.01)
	eta	quniform(0.025, 0.5, 0.025)
	max_depth	range(1, 14, 1)
	min_child_weight	quniform(1, 6, 1)
	subsample	quniform(0.5, 1.0, 0.05)
	gamma	quniform(0.0, 1.0, 0.05)
	colsample_bytree	quniform(0.5, 1.0, 0.05)

Table 14: Algorithms and parameters. The implementations of LinearSVM, kNN and LogisticRegression are from scikit-learn and XGBoost is from the respective authors implementation-based package. Omitted parameters are the libraries default. This table has the **range** functions and the **uniform** and **quniform** distributions functions, which are used to define the search space of some algorithms. The range(low, high, step) function returns a number between [**low**, **high**) in a **step** interval. The uniform(low, high) function returns a value uniformly between **low** and **high**. The quniform(low, high, q) function returns a value like round(uniform (low, high) / q) * q and differs from the uniform by a smooth factor.

Unsupervised Domain Adaptation for Event Detection using Domain-specific Adapters

Nghia Ngo Trung[1], **Duy Phung**[1] **and Thien Huu Nguyen**[2]
[1] VinAI Research, Vietnam
[2] Department of Computer and Information Science, University of Oregon,
Eugene, OR 97403, USA
`{v.nghiant66,v.duypv1}@vinai.io`, `thien@cs.uoregon.edu`

Abstract

Due to the multi-dimensional variation of textual data, detection of event triggers from new domains can become a lot more challenging. This prompts a need to research on domain adaptation methods for event detection task, especially for the most practical unsupervised setting. Recently, large transformer-based language models, e.g. BERT, have become essential to achieve top performance for event detection. However, their unwieldy nature also prevents effective adaptation across domains. To this end, this work proposes a **D**omain-specific **A**dapter-based **A**daptation (**DAA**) framework to improve the adaptability of BERT-based models for event detection across domains. By explicitly representing data from different domains with separate adapter modules in each layer of BERT, **DAA** introduces a novel joint representation learning mechanism and a Wasserstein distance-based technique for data selection in adversarial learning to substantially boost the performance on target domains. Extensive experiments and analysis over different datasets (i.e., LitBank, TimeBank, and ACE-05) demonstrate the effectiveness of our approach.

1 Introduction

Event detection (ED) is an important component in the overall event extraction pipeline, which plays a crucial role in any natural language understanding system. The goal of ED is to identify event triggers in a given text and classify them into one of several pre-defined types. Formally, according to the ACE-05 annotation guideline , each event trigger is a phrase (usually a single verb or nominalization), which evokes that event in the context of the associating event mention. For example, the word "*fired*" is the trigger word for an event of type *Attack* in the following sentence: *"The police* ***fired*** *tear gas and water cannons in street battles with activists."* Tackling ED problem involves both locating the event triggers and categorizing them into specific event types, therefore can be a quite challenging task due to the intricate dependency among triggers, events, and contexts in linguistic data. The complication is further amplified by domain shift problem when text are collected from multiple different domains.

The majority of prior approaches on ED relied on the basic supervised learning assumption where training and testing data follow the same distribution. Several works further evaluated their methods on cross-domain setting where their models were trained using data from one domain and tested on another, without leveraging any adaptation mechanism to alleviate the domain shift problem (Nguyen and Grishman, 2015; Yubo et al., 2015; Hong et al., 2018b). To this end, our work explores the general problem of domain adaptation for ED where data comes from two different source and target domains. In particular, we focus on the unsupervised setting that requires no annotations for target data, and the model has to learn to make use of both labeled source and unlabeled target samples to improve its performance on target domain. To our knowledge, this is the first work on unsupervised domain adaptation (UDA) for ED in the literature.

The most prominent approach for UDA is a representation learning method based on the theory of learning from different domains developed by Ben-David et al. (2010). The main result provided a way to bound the loss of a model on target domain with its performance on source domain using a domain-divergence term and an optimal joint error term (which is assumably negligible). Ganin et al. (2016) adopted this idea for deep learning architecture in their domain-adversarial neural network (DANN). They employed a domain-adversarial training procedure in which a domain classifier is learned concurrently and adversarially with the network's fea-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4015–4025
August 1–6, 2021.

ture extractor, resulting in a not only discriminative but also domain-invariant joint representation for data from both domains. While DANN and its variants are very well-studied in computer vision's domain adaption researches, their NLP counterparts are pale in comparison, especially for a newly established architecture like BERT. There have been only several works that adopted DANN to align the contextualized representations learned by BERT across domains (Lin et al., 2020; Naik and Rosé, 2020; Wright and Augenstein, 2020). Lin et al. (2020) even observed negative effect when applying adversarial training compare with simply fine-tuning BERT on in-domain data. One explanation is that the pre-training of BERT on massive corpora has already induces a somewhat general representation, thus DANN has little effect while the fine-tuning process using source dataset could cause over-fitting on the corresponding domain due to the immense capacity of the model. To this end, we propose fixing the parameters of the already universal language model while leveraging multiple adapter modules for domain-adversarial training process. More specifically, inspired by the works of Liu et al. (2017a) and Houlsby et al. (2019) on effective multi-task learning, we augment the pre-trained BERT model by adding three different adapters to create a shared-private architecture. Two source and target adapters which take as inputs data from their respective domains to capture private properties of each, and a joint adapter that encodes every sample in a subspace shared across domains through adversarial training. Orthogonality constraints together with a self-supervised auxiliary task are employed to ensure the representations of all adapters are informative while also attaining the above desired properties.

Recently, Ma et al. (2019) and Aharoni and Goldberg (2020) have shown that BERT's representations are extremely effective at clustering text to their respective domains, and a small subset of good in-domain data can already provide significant boosts in target performance while the rest only provide little to no improvement, in some cases even degrade model's out-of-domain generalization. Considering this, we explicitly find hard instances to leave out when learning to extract the domain-invariant features. Our data selection component estimates and minimizes the cost of transport between source and target marginal representation distributions based on the Wasserstein-1 distance (also refer to as Earth Mover distance). Arjovsky et al. (2017) pointed out that the relative strength of the topologies induced by this distance is much weaker than that of KL-divergence used by adversarial training. Therefore, it could serve as a good necessary condition for DANN component to achieve optimal alignment. The faraway source instances that induce the highest transportation costs are those out-of-distribution samples that may introduce noise and hurt adaptation performance. Accordingly, they are omitted from the domain-adversarial training process. The entire computation makes use of representations from source and target adapters, thus implicitly provides informative signals from domain-specific adapters to joint adapter without interrupting the joint representation learning procedure.

2 Related Work

Prior ED works have focused on the in-domain setting (Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016; Yang and Mitchell, 2016; Nguyen and Grishman, 2018; Sha et al., 2018; Liu et al., 2017b; Tong et al., 2020; Nguyen et al., 2021), the cross-domain evaluation (Nguyen and Grishman, 2016; Hong et al., 2018a), the few/low-shot learning scenario (Lai et al., 2020a,b). Our work is different from those prior work as we explore a new formulation for ED with unsupervised domain adaptation where unlabeled data in the target domain is utilized to improve domain-invariant representation learning.

Recently, some efforts have been made to study the domain-related knowledge encoded in BERT's representations (Aharoni and Goldberg, 2020), and methods to leverage it to improve performances on domain-specific tasks, such as pre-training on additional data (Gururangan et al., 2020), fine-tuning using intermediate tasks (Phang et al., 2018; Garg et al., 2020), and data selection (Ma et al., 2019; Aharoni and Goldberg, 2020). Another line of research regarding multi-task learning shares a common goal of creating a universal representation space for all data with domain adaptation. Previous approaches made use of multiple encoders to set up a shared-private architecture (Bousmalis et al., 2016; Liu et al., 2017a), which usually is impractical for BERT-based models because of theirs sizes. By fixing a pre-trained BERT as the base for general representations, Houlsby et al. (2019) and Stickland and Murray (2019) proposed to adapt the model to each task by adding small task-specific layers between BERT's layers and updating only them when fine-tuning on the corresponding task.

3 Model

Throughout this work, we formulate ED task as a token-level multi-class classification problem (Nguyen and Grishman, 2015; Ngo et al., 2020). For UDA setting in particular, we have a labeled source dataset $\mathcal{D}^{\mathbf{s}}_{\mathbf{XY}} = \{(\mathbf{X}^{\mathbf{s}}_i, y^{\mathbf{s}}_i)\}_{i=1}^{N^{\mathbf{s}}}$ of $N^{\mathbf{s}}$ samples from source domain $\mathbf{s}$ and an unlabeled set of $N^{\mathbf{t}}$ samples $\mathcal{D}^{\mathbf{t}}_{\mathbf{X}} = \{\mathbf{X}^{\mathbf{t}}_i\}_{i=1}^{N^{\mathbf{t}}}$ drawn from target domain $\mathbf{t}$. Each $\mathbf{X}^{\mathbf{s}}_i$ is a pair consists and an event mention $W^{\mathbf{s}}_i = (w^{\mathbf{s}}_{i1}, w^{\mathbf{s}}_{i2}, \cdots, w^{\mathbf{s}}_{im})$ (m is the fixed number of words), and a trigger position u $(1 \leq u \leq m)$ corresponding to the word $w^{\mathbf{s}}_{iu}$. An encoder computes its latent representation $x^{\mathbf{s}}_i$, which is then used by the event classifier to predict an event of type $y^{\mathbf{s}}_i$. For target domain, parallel notations are used $x^{\mathbf{t}}_i$ and $y^{\mathbf{t}}_i$ (only accessible in target domain's test dataset)

3.1 Baseline Model

As this is the first work on UDA for ED, this section aims to establish a baseline of the task for further research. Recent works have shown a substantial boost in performance for the standard supervised setting of ED by leveraging contextual embedding of large self-attention based language models (Wang et al., 2019; Lai et al., 2020c). Accordingly, we utilize a pre-trained BERT's encoder, together with its domain-adversarial variant to create a strong baseline for the UDA setting.

Without any domain adaptation mechanism, our **BERT** baseline only follows cross-domain evaluation setting as previous works. The model is fully fine-tuned on source domain dataset while at test time, data from target domain is used to evaluate its performance.

On the other hand, the **BERT+DANN** baseline takes advantage of the availabel unlabeled target data through adversarial training. Specifically, a domain classification task is learned concurrently with the main downstream task, using unlabeled samples and their domain labels from both domains. By pushing the encoder to both minimize the event classfication loss and maximally misdirect domain predictor, the resulting representation can be indiscriminate with respect to the shift between the domains while also discriminative for the main learning task.

Finally, to demonstrate to ability of adapter-based tuning approach to retain the original's model performance, we also evaluate a **BERT+Adapter** baseline. Following recommendation from Pfeiffer et al. (2021), we augment a pre-trained BERT model by injecting a single bottleneck adapter module between the encoder's layers. Then, the fine-tuning process proceeds in the same manner as that of the **BERT** baseline, but only parameters of the adapter modules get updated in this case.

3.2 Adapter-based Domain Representation

Pre-trained BERT model was previously optimized for the task of masked language model in unsupervised manner on several extremely large corpora. The diversity of these unlabeled text also pushes the network to be a good starting point for learning domain-invariant features, which would be lost if we fully fine-tune it on source domain task. Accordingly, we make use of a fixed pre-trained BERT model as the base of our adapters.

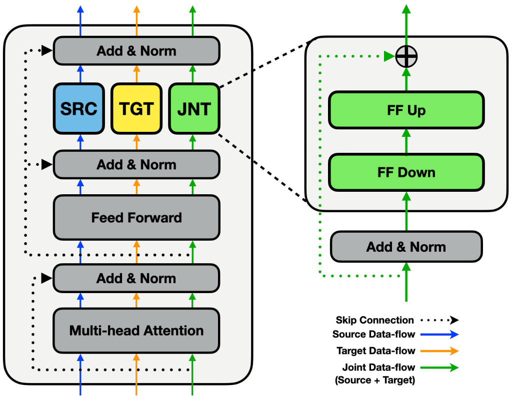

Figure 1: Domain-specific adapters inside the original BERT's layers.

An adapter for each domain To explicitly create a shared-private representation subspace of each domain, we inject three adapters into the same base encoder. Formally, adapter modules $a^{\mathbf{s}}_i$, $a^{\mathbf{t}}_i$, $a^{\mathbf{j}}_i$ are added on top of each BERT's layer. While these modules share the same architecture, they take in as inputs data only from their corresponding sources:

$$\mathbf{A}^{\mathbf{d}}(x^{\mathbf{d}}_i) = a^{\mathbf{d}}_L \circ b_{L-1} \circ \cdots \circ a^{\mathbf{d}}_1 \circ b_1(x^{\mathbf{d}}_i)$$

where b_i is the fixed layer of BERT $(1 \leq i \leq L)$, $\mathbf{d} \in \{\mathbf{s}, \mathbf{t}, \mathbf{j}\}$, and $\mathcal{D}^{\mathbf{j}}_{\mathbf{X}} = \mathcal{D}^{\mathbf{s}}_{\mathbf{X}} \cup \mathcal{D}^{\mathbf{t}}_{\mathbf{X}}$. The joint adapter $\mathbf{A}^{\mathbf{j}}$ is our main representation which will be used by event detection head h_c for source classification task:

$$\mathcal{L}_c = \frac{1}{n^{\mathbf{s}}} \sum_{i=1}^{n^{\mathbf{s}}} -y_i^{\mathbf{s}} log\left(h_c\left(\mathbf{A}^{\mathbf{j}}\left(x_i^{\mathbf{s}}\right)\right)\right)$$

On the other hand, the domain-specific adapters $\mathbf{A}^{\mathbf{s}}$ and $\mathbf{A}^{\mathbf{j}}$ will only be used to help $\mathbf{A}^{\mathbf{j}}$ find to the optimal joint-domain space while simultaneously retain good performance on downstream task.

Adapter architecture: There are a variety of ways that one can design the adapter modules' architecture. Following the observations from Pfeiffer et al. (2021), we choose ours to be the most efficient but also effective, which is a singular bottleneck neural network with skip-connection, taking features computed by BERT's feed-forward sub-layer as inputs. The adapter module in layer l can be decoupled into two parts $a_l^{\mathbf{d}} = a_l^{\mathbf{d},up} \circ a_l^{\mathbf{d},dw}$, where $a_l^{\mathbf{d},dw} : \mathbb{R}^{d_{model}} \to \mathbb{R}^c$ and $a_l^{\mathbf{d},up} : \mathbb{R}^c \to \mathbb{R}^{d_{model}}$ (as shown in figure 1). Despite tripling the added parameters from adapter modules, by setting $c \ll d_{model}$, the amount needed to be tuned is still only less than 10% that of the original network. Additionally, the factorized features enable effective adaptation by making use of the low-dimensional down-sampled representation, while also boosting classification performance by leveraging the free parameters of the up-sampling projection, as described in the next sections.

3.3 Joint Representation Learning

To learn a joint representation that is as general as possible while also maintaining its discriminative property, we propose to combined two mechanisms with complementary effects : a layer-wise domain-adversarial (LDA) component and an adapter-wise domain disentanglement (ADD) component.

3.3.1 Domain-adversarial Training

LDA apply domain-adversarial training to the representation of $\mathbf{A}^{\mathbf{j}}$. Multiple refinements to the original DANN are introduced to mitigate its flaws and learn better domain-invariant features.

Dimension Reduction It is known that discriminative features computed by high-level layers usually lie on low dimensional manifolds. As a result, naively applying adversarial training for BERT's representations, which require high dimension to capture contexts, can lead to gradient vanishing problem. We leverage the adapter's architecture to tackle this issue. Instead of the full dimension outputs of layers, we align domains based on the down-sampled version of the representations, computed by $a_i^{\mathbf{j},dw}$. In consequence, an adapter module can be viewed as a two-step adaption: a down-sampling projection step that extracts domain-invariant features and a following up-sampling projection step which transforms the extracted general features into task discriminative ones.

Layer-wise Alignment To enhance the alignment capability of our model, domain-adversarial training is applied on every layer's output. In particular, we incorporate the asymmetric relaxation of DANN (Wu et al., 2019):

$$\mathcal{L}_d^l = -\frac{1}{N}\sum_{i=1}^{N}\left[d_i\, log\left(\frac{h_d(a_l^{\mathbf{j},dw}(x_i))}{1+\beta_j}\right)\right. \\ \left. +(1-d_i)log\left(1-\frac{h_d(a_l^{\mathbf{j},dw}(x_i))}{1+\beta}\right)\right]$$

where d_i is domain label of samples x_i, $N = n^{\mathbf{s}} + n^{\mathbf{t}}$ is minibatch size, and $\beta_l \geq 0$ is a hyperparameter controlling the maximal difference of the two marginal distributions ($\beta_l = 0$ is the original formulation). This modification addresses the target shift scenario where domain-adversarial training is unable to achieve optimal solution. As outlined by Rogers et al. (2021), lower-level layers of BERT contain quite broad knowledge, thus encode more random distribution when projected into label space. In contrast, high-level ones are gradually more task-specific, effectively reducing the possible amount of label shift between the two domains. Therefore, we adopt the following relaxation annealing strategy:

$$\mathcal{L}_d = \sum_{l=1}^{L} \mathcal{L}_d^l(a_l^{\mathbf{j},dw}, \beta_l, h_{d,l}^{\mathbf{j}}), with\ \beta_l = 2^{3-l}$$

where each term on the right-hand side is a different relaxed domain classification loss computed by a separate domain classifier $h_{d,l}^{\mathbf{j}}$.

3.3.2 Adapter-wise Domain Disentanglement

The role of ADD component is to ensure the shared-private relationship among adapters. We want the joint adapter $\mathbf{A}^{\mathbf{j}}$ to encode a shared representation space containing common information between domains and no domain-specific information, while the private adapters $\mathbf{A}^{\mathbf{s}}$ and $\mathbf{A}^{\mathbf{t}}$ should only accommodate distinct knowledge that belong exclusively to their corresponding domains. Following the work of Liu et al. (2017a) and Bousmalis et al. (2016), an orthogonality constraint is imposed using the following similarity loss function:

$$\mathcal{L}_s = \|\mathbf{A}_{\mathbf{s}}^{\mathbf{j}}{}^{\top}\mathbf{A}_{\mathbf{s}}^{\mathbf{s}}\|_F^2 + \|\mathbf{A}_{\mathbf{t}}^{\mathbf{j}}{}^{\top}\mathbf{A}_{\mathbf{t}}^{\mathbf{t}}\|_F^2$$

where $\|.\|_F$ is the Frobenius norm and $\mathbf{A}_{\mathbf{d_2}}^{\mathbf{d_1}}$ is a matrix whose rows are the outputs of adapter $\mathbf{A}^{\mathbf{d_1}}$ taking inputs from domain $\mathbf{d_2}$. Minimizing $\mathcal{L}_s$ will force $\mathbf{A}^{\mathbf{j}}$ to be in a complementary subspace with $\mathbf{A}^{\mathbf{s}}$ and $\mathbf{A}^{\mathbf{t}}$, encouraging independency among adapters and removing domain-specific noises that may contaminate the joint representation. However, whereas $\mathbf{A}^{\mathbf{j}}$ is trained to be informative for the downstream classification, $\mathbf{A}^{\mathbf{s}}$ and $\mathbf{A}^{\mathbf{t}}$ are not constrained by any task, which potentially could lead to a trivial solution where the network learns to map each representation into the same orthogonal space with $\mathbf{A}^{\mathbf{j}}$ while not having any expressive capability of their corresponding domains. To address this issue, we incorporate a self-supervised component, using the popular Masked Language Modeling (MLM) as our unsupervised task. The token predictor $h_m : \mathbb{R}^{d_{model}} \rightarrow \mathbb{R}^V$ (V is the vocabulary size), is shared between source and target domains:

$$\mathcal{L}_m = \sum_{i=1}^{N_{mask}} \mathcal{L}_m^{\mathbf{s}}(x_i^{\mathbf{s}}) + \mathcal{L}_m^{\mathbf{t}}(x_i^{\mathbf{t}})$$

$$\mathcal{L}_m^{\mathbf{d}}(x_i^{\mathbf{d}}) = -w_i^{\mathbf{d}} \log h_m\left(\mathbf{A}^{\mathbf{j}}(x_i^{\mathbf{d}}) + \mathbf{A}^{\mathbf{d}}(x_i^{\mathbf{d}})\right)$$

where N_{mask} is the number of randomly masked input tokens, following the original procedure in Devlin et al. (2019). The benefit of adding the MLM component is twofold. On one hand, it serves as a constraint to learn informative representations for domain-specific adapters. On the other hand, it also help conditioning joint adapter $\mathbf{A}^{\mathbf{j}}$ on unsupervised knowledge of unlabeled target data, which can have a positive impact on target domain's performance.

3.4 Data Selection

Considering the Wasserstein-1 distance between the distributions generating source and target marginal representations $\mathbb{P}_{\mathbf{X}}^{\mathbf{s}}$ and $\mathbb{P}_{\mathbf{X}}^{\mathbf{t}}$, which can be written as:

$$W(\mathbb{P}_{\mathbf{X}}^{\mathbf{s}}, \mathbb{P}_{\mathbf{X}}^{\mathbf{t}}) = \sup_{\|f\|_L \leq 1} \mathbb{E}\left[f(x^{\mathbf{s}})\right] - \mathbb{E}\left[f(x^{\mathbf{t}})\right]$$

There are several advantages of using this distance as the proxy for data selection mechanism. First, Wasserstein distance takes into account the geometry of the actual data distributions. Thus, it is intuitive to use it to evaluate the discrepancy between marginal distributions and pick source samples that are geometrically close to samples from target distribution. Furthermore, it has been proven by Arjovsky et al. (2017) that the minimization of KL-divergence, on which LDA component based to update $\mathbf{A}^{\mathbf{j}}$, also implies the minimal Wasserstein distance between the corresponding distributions. Therefore, leaving out the most far-a-way samples based on this distance should provide a good necessary condition for LDA to achieve optimal alignment from source to target domain.

Approximate Wasserstein Distance Following the approximation from Shen et al. (2018), we employ a data selection head h_w to estimate the Wasserstein distance between two representation distributions of $\mathbf{A}^{\mathbf{s}}$ and $\mathbf{A}^{\mathbf{t}}$ by maximizing the following empirical loss with respect to θ_w:

$$\mathcal{L}_{\hat{w}} = \frac{1}{n^{\mathbf{s}}} \sum_{i=1}^{n^{\mathbf{s}}} h_w\left(\mathbf{A}^{\mathbf{s}}\left(x_i^{\mathbf{s}}\right)\right) - \frac{1}{n^{\mathbf{t}}} \sum_{i=1}^{n^{\mathbf{t}}} h_w\left(\mathbf{A}^{\mathbf{t}}\left(x_i^{\mathbf{t}}\right)\right)$$

For the above approximation to work, we need to enforce the Lipschitz constraint, which will force the hypothesis class of h_w to be 1-Lipschitz. Following Gulrajani et al. (2017), a gradient penalty $\mathcal{L}_{gr}$ is added to the loss, resulting in the overall estimation problem for the Wasserstein distance as

$$\max_{\theta_w} \mathcal{L}_w = \mathcal{L}_{\hat{w}} - \lambda_{gr}\mathcal{L}_{gr}$$

$$\mathcal{L}_{gr}(\mathbf{A}^{\mathbf{d}}) = (||\nabla_{\mathbf{A}^{\mathbf{d}}} h_w(\mathbf{A}^{\mathbf{d}})||_2 - 1)^2$$

where $\mathbf{d} \in \{\mathbf{s}, \mathbf{t}\}$ and λ_{gr} is a hyper-parameter.

Data Selection based on Wasserstein Distance To avoid negative transfer problem in case of highly dissimilar domains, we propose to use a data selection mechanism based on the estimated Wasserstein distance. By minimizing the empirical distance using $\mathbf{A}^{\mathbf{s}}$ and $\mathbf{A}^{\mathbf{t}}$, we find the representations that achieve the shortest transport distance between source and target samples. Then, a subset of $\hat{n}^{\mathbf{s}}$ source samples is selected with the lowest $h_w(.)$ scores, which corresponds to the $\hat{n}^{\mathbf{s}}$ shortest distances to target domain. These source instances will be used by the joint adapter $\mathbf{A}^{\mathbf{j}}$, together with target unlabeled data, to learn domain-invariant features in LDA.

System	In-domain(`bn+nw`)			Out-of-domain (`bc`)			Out-of-domain (`cts`)			Out-of-domain (`wl`)			Out-of-domain (`un`)		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
BERT	77.5	77.5	77.5	75.2	71.8	73.5	75.1	69.9	72.4	70.2	57.9	60.6	68.9	67.5	68.2
BERT+DANN	77.4	75.8	76.6	72.8	69.4	70.9	73.4	39.9	51.2	69.2	50.5	58.4	68.8	59.2	63.6
BERT+Adapter	76.8	76.2	76.7	78.5	72.9	75.6	77.3	69.5	73.2	64.3	56.9	60.3	72.4	69.0	70.6
DAA	79.7	75.7	**77.7**	78.5.1	75.6	**76.9**	78.4	73.2	**75.6**	66.2	60.3	**63.1**	73.5	71.3	**72.3**

Table 1: Unsupervised domain adaptation for event detection. Performance on the **ACE-05** test datasets for different domains.

System	In-domain (`TimeBank`)			Out-of-domain (`LitBank`)		
	P	R	F	P	R	F
LSTM+DANN	69.3	87.5	77.3	25.6	72.9	37.9
BiLSTM+DANN	74.2	79.4	76.7	26.3	72.0	38.6
BERT	79.6	84.3	81.9	28.1	84.8	42.2
BERT+DANN	79.8	85.6	82.6	30.3	80.8	44.1
DAA	90.9	88.4	**89.6**	40.0	81.3	**53.6**

Table 2: Performance on `TimeBank-to-LitBank`.

System	In-domain (`LitBank`)			Out-of-domain (`TimeBank`)		
	P	R	F	P	R	F
LSTM+DANN	61.1	61.6	61.3	89.0	18.9	31.2
BiLSTM+DANN	66.1	62.8	64.4	92.9	18.5	30.9
BERT	73.5	72.7	73.1	88.1	28.2	42.7
BERT+DANN	71.9	71.3	71.6	85.0	35.0	49.6
DAA	77.7	75.6	**76.7**	83.2	48.5	**61.1**

Table 3: Performance on `LitBank-to-TimeBank`.

3.5 Alternating Minimization

Taking it all together, our final training objective is given as:

$$\mathcal{L}_{total} = \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_w \mathcal{L}_w + \lambda_s \mathcal{L}_s + \lambda_m \mathcal{L}_m$$

where $\lambda_d, \lambda_w, \lambda_s, \lambda_m$ are hyper-parameters which help to balance the importance of the corresponding loss with the main event detection loss. Of the five terms on the right-hand side, the domain discrepancy losses ($\mathcal{L}_d$ and $\mathcal{L}_w$) require optimization of different directions with respect to the added heads and the adapters, resulting in a min-max optimization problem. Previous works that made used of domain-adversarial training usually applied gradient reversal layer to train the feature extractors. We find this approach to be unstable and cause performance degradation. Following suggestions from Goodfellow et al. (2014) and Shu et al. (2018), we design an alternating minimization process that is compatible with our learning algorithm whilst also stabilizing the domain-adversarial training. In the first stage, all parameters are fixed except for those of domain-adversarial heads and data selection head. This step corresponds to the estimation of corresponding distance functions between domains given the current representations. After repeatedly updating for k times (k is a hyper-parameter that controls the trade-off between computation and accuracy of the divergence estimations), a subset of source minibatch can be selected based on the approximated Wasserstein distance, which will be used for domain-adversarial training of joint adapter in next step. The following stage, while keeping the previously updated heads fixed, updates the rest of the model's parameters, using the standard gradient descent algorithm. All maximization problems of discrepancy losses are converted into minimization using reversed domain labels.

At test time, a new sample x_{test} will go through the trained joint adapter $\mathbf{A^j}$ to produce domain-invariant representation $\mathbf{A^j}(x_{test})$, which is then used by prediction head h_c to produce the corresponding event label.

4 Experiments

We evaluate our model on two related tasks: event identification and event detection. Given a trigger word in the context of an event mention, the former is formulated as a binary classification problem in which the goal is to determine if the trigger word expresses an event, while the latter is a multi-class classification task that requires model to assign the predicted label into one of the pre-defined 34 event types (include 1 negative type).

4.1 Datasets

TimeBank dataset (Pustejovsky et al., 2003) a fine-grained temporally annotated corpus of events and their positions and ordering in time. The text of the dataset were chosen from a wide range of sources from the news media domain. Events are annotated in a binary manner.

LitBank dataset (Sims et al., 2019) a recently introduced corpus of literary events. The dataset contains excerpts of 100 literary works from the Project Gutenberg corpus. Labels for events are binary.

System	In-domain(`bn+nw`)			Out-of-domain (`bc`)			Out-of-domain (`cts`)			Out-of-domain (`wl`)			Out-of-domain (`un`)		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
DAA-D	75.8	79.7	77.7	74.4	76.8	75.5	78.4	70.0	73.9	67.2	57.1	61.7	71.3	70.3	70.8
DAA-W	79.2	76.4	77.7	77.8	73.5	75.6	80.8	70.1	75.1	70.2	53.7	60.9	74.0	66.7	70.1
DAA-M	78.1	76.5	77.3	80.4	71.0	75.4	77.0	69.4	73.3	68.9	55.1	61.2	73.5	68.6	71.0
DAA-S	77.5	77.1	77.3	78.7	72.4	75.4	79.2	70.7	74.7	65.5	57.9	61.5	72.7	67.9	70.2
DAA	79.7	75.7	**77.7**	78.5	75.6	**76.9**	78.4	73.2	**75.6**	66.2	60.3	**63.1**	73.5	71.3	**72.3**

Table 4: Ablation study. Performances on the ACE-05 test datasets for different domains.

Automatic Content Extraction 2005 (ACE-05) dataset (Walker et al., 2005) a densely annotated corpus collected from 6 different domains: Newswire (`nw`) - 20%, Broadcast news (`bn`) - 20%, Broadcast conversation (`bc`) - 15%, Weblog (`wl`) - 15%, Usenet Newsgroups (`un`) - 15%, Conversational Telephone Speech (`cts`) - 15%. Events of the dataset are categorized into 33 types.

4.2 Experimental Setup

4.2.1 Unsupervised Domain Adaptation Setting

To formulate the unsupervised domain adaptation setting from the origin dataset of each task, we split the target domain's documents into two parts at the ratio of 1 to 4, a training dataset without labels which models have access to when learning, and a test dataset that models are evaluated on. For event identification, transfer experiments are performed in two ways: `LitBank-to-TimeBank`, and the reserve direction, `TimeBank-to-LitBank`. In event detection experiments, we combine samples from two closely related domains, `nw` and `bn`, to create a sizeable labeled training source dataset. Then, each of the other domains is considered the target domain of a single adaptation setting.

4.2.2 Implementation and Hyper-parameters

Our model leverage the pre-trained BERT-base model as the fixed foundation for all adapters, each of which has a down-sampled dimension of 96. All of the downstream heads are implemented as feed-forward networks with activation functions between layers. We train all models using batch size of 150, which composes of 90 source samples (60 of which will be used for domain-adversarial training) and 60 target samples. Weights of the losses are chosen from a grid-search of range $[0.01, 0.05, 0.1, 0.2, 0.5, 1, 5]$ using `bc` domain as development dataset. Every experiment is run 5 times epochs with different random seeds and the performance is reported using the average result of the 5 runs.

4.2.3 Baseline

We compare the proposed model **DAA** with several other baselines. In particular, for the task of event identification, the performance of domain-adversarial models implemented in Naik and Rosé (2020) are considered. Regarding the event detection task, our baselines include adaption results of **BERT** and **BERT+Adapter** models fine-tuned using only source dataset, and finally **BERT+DANN** which making use of unlabeled target data through adversarial training.

4.3 Experimental Result

Event Identification The results of our event identification experiments are presented in tables 2 and 3. In both settings, our proposed model **DAA** outperforms naive implementation of domain-adversarial on BERT by about 10 points in F1. We also note that high precision is observed from models transferring from `LitBank-to-TimeBank`, while the other direction has high recall. This imbalance is caused by the extreme disparity between the two adaptation settings, which our model manages to address and thus significantly improves out-of-domain performance in both cases.

Event Detection Table 1 showcases the results of our event detection experiment. The main conclusions from the table include: (1) The **BERT** baseline performs decently without using any mechanism to address the discrepancy between domains. This is due to the generalization potential of large unsupervised pre-trained language model. However, naively adopting DANN for BERT has an adverse effect, notably reducing the performance of **BERT+DANN** on all target domains. This outcome is consistent with results from Lin et al. (2020), further emphasizing the need for a compatible implementation method for domain-adversarial training on BERT's representations. (2) The results of **BERT+Adapter** proves that adapter-based tuning procedure is not only able to retain performance but also prevent over-fitting through capacity reduction, therefore performing better than the

System	`bc`			`cts`			`wl`			`un`		
	P	R	F	P	R	F	P	R	F	P	R	F
Lower	77.5	73.9	75.6	74.1	74.7	74.4	67.5	56.7	61.6	73.0	69.4	71.0
Middle	78.1	73.4	75.7	78.5	70.4	74.2	66.4	56.7	61.2	71.6	70.8	71.2
Higher	79.2	73.3	76.1	77.5	71.2	74.2	67.4	55.5	60.9	72.9	68.1	70.4
Last	79.6	72.6	75.9	79.7	69.5	74.2	66.4	56.6	61.0	73.0	69.7	71.3
Up-Dim	77.6	73.2	75.3	74.5	71.7	73.0	66.6	53.1	59.1	69.0	67.7	68.4
No-Rel	77.8	75.1	76.5	76.1	74.2	75.2	67.2	58.0	62.3	72.2	70.6	71.4
Full	78.5	75.6	**76.9**	78.4	73.2	**75.6**	66.1	59.6	**62.6**	73.5	71.3	**72.3**

Table 5: Domain-adversarial analysis. Performance on the ACE-05 test datasets for different domains.

fully fine-tuned **BERT** in case where it follows too closely to source domain. (3) Finally, our proposed model **DAA** manages to achieves the best adaptation performance across all target domains. In settings where domains are closely related such as `bc` and `cts`, **DAA** is more robust and thus performs better on target domain. On the other hand, **DAA** significantly outperforms baselines (3 to 5 points increase in F1 score) when transferring to target domains that are highly dissimilar to source domains (`wl` and `un`).

4.4 Ablation Study

To examine the effect of each of the proposed component individually, We perform an extensive ablation analysis for our **DAA** model by measuring domain adaptation ability of each trained model, with a single main component discarded (by setting the weight of its associated loss to 0), on ACE-05. In table 4, **DAA-D**, **DAA-W**, **DAA-M**, and **DAA-S** correspond to performances of partial models with domain-adversarial training, data selection component, self-supervised task, and orthogonality constraint removed, respectively. Results from the study show that every incomplete model performs consistently worse compare to the full model. In particular, while in-domain performances are retained across settings, different domains experience varying degree of reduction in target performance depending on its relation with the source domain. Especially, data drawn from the domains of `wl` and `un` are substantially diverged from the source domain. Therefore, components that address domains' dissimilarity play important roles in improving adaptation capability, which is confirmed by the fact that models such as **DAA-W** and **DAA-D** have the lowest results.

4.5 Domain-adversarial Analysis

The central component of our architecture is undoubtedly LDA whose responsibility is to ensure joint adapter extracts domain-invariant features for classifying event triggers. From the negative results of **BERT+DANN**, finding an appropriate way to implement domain-adversarial training for BERT is an important question. This section aims to demonstrate the effectiveness of our layer-wise implementation of DANN.

We apply domain alignment to different portions of BERT. Specifically, we partitioned 12 layers of the BERT-base encoder into 3 levels - **Lower**, **Middle**, **Upper** - each corresponds to the only 4 layers whose representations are used by domain-adversarial training. In addition, we present results of **Last** and **Up-Dim**. The former is original implementation where last layer's output is aligned , while the latter is similar to our model **Full** except the representation with full dimension (768) is used instead of the down-sampled ones. Finally, **No-Rel** is the same as **Full** but no relaxation is used.

Table 5 showcases the results of our experiment. Overall, we observe performance degrades on all three partial adaptation settings However, the changes vary across domains in each situation, probably stemming from the fact that adversarial training addresses different degrees of domain shifts in each layer. Moreover, taking only the last layer's representation as input for DANN component performs worse compare to all other multi-layer counterparts. Notably, using representations with full dimension significantly reduces out-of-domain performances of model. This result confirms the benefit of the bottleneck architecture. Not only the alignment of down-sampled representations is more effective, but the free parameters of up-sampling layers also increase model's capacity for the main downstream task.

4.6 Domain Discrepancy Analysis

To verify the effect of our method on alleviating the negative impact of the domain shift problem on the learning process, we compare each model's performance on different settings with varying shift magnitudes. Specifically, for each target domain, based on the learned Wasserstein distance between the two domains, we quantify the distance of each target domain sample (in evaluation dataset) to the source dataset and group them into 2 disjoint sets: **FAR** - 25% of target samples that are farthest from the source dataset, and **CLOSE** - 25% of those closest to source dataset. The domain adaptation performances on these sets for 2 target domain `bc` and `wl`, together with the set of in-domain examples **IN-DOM** from `bnnw` domain, are provided in Table 6. When adapting to `bc` domain which has a low discrepancy to source domain, the results for each setting show little variance, but we still observe the over-fitting of **BERT** as performance of out-of-domain settings is lower compared to its in-domain score. Moreover, **BERT+DANN** is able to improve on **FAR** set, however at the cost of degradation in the other two settings. In contrast, the negative effect of high discrepancy between domains is apparent in the case of `wl` domain, as the gaps between each setting are all above 10 points. Notably, the results of **BERT+DANN** are lower than that of **BERT**, indicating that naive implementation of **DANN** is not only unable to align between source and target domains, but also causes negative transfer when trying to learn domain-invariant representation. On the other hand, in both case, **DAA** is able to address the weakness of the baseline and improves the performance on **FAR** and **CLOSE** simultaneously.

	`bc`		`wl`		`bnnw`
	FAR	CLOSE	FAR	CLOSE	IN-DOM
BERT	72.4	75.6	43.1	59.2	77.5
BERT+DANN	73.2	76.6	35.3	52.4	76.6
DAA	74.8	76.4	50.9	64.4	77.7

Table 6: Domain adaptation performances in F1 score with different domain shift settings.

5 Conclusion

We present a novel framework for ED in UDA setting that effectively leverages the generalization capability of large pre-trained language models through a shared-private adapter-based architecture. A layer-wise domain-adversarial training process combined with a Wasserstein-based data selection addresses the discrepancy between domains and produces domain-invariant representations. The proposed model achieves state-of-the-art results on several adaptation settings across multiple datasets. In the future, we plan to extend our approach in the several directions: (1) We will devise a method to incorporate target domain's private adapter to further improve model's out-of-domain performance.; (2) We will adapt our framework to more general settings such as multi-source domain adaptation and domain generalization.; and (3) We will extend our work to novel domains for ED (Trong et al., 2020).

Acknowledgments

This research has been supported by the Army Research Office (ARO) grant W911NF-21-1-0112. This research is also based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the Better Extraction from Text Towards Enhanced Retrieval (BETTER) Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2020. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Yu Hong, Wenxuan Zhou, jingli zhang jingli, Guodong Zhou, and Qiaoming Zhu. 2018a. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yu Hong, Wenxuan Zhou, Jingli Zhang, Guodong Zhou, and Qiaoming Zhu. 2018b. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020a. Exploiting the matching information in the support set for few shot event classification. In *Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Viet Dac Lai, Thien Huu Nguyen, and Frank Dernoncourt. 2020b. Extensively matching for few-shot learning event detection. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020c. Event detection: Gate diversity and syntactic importance scoresfor graph convolution neural networks. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chen Lin, Steven Bethard, Dmitriy Dligach, Farig Sadeque, Guergana K Savova, and T. A. Miller. 2020. Does bert need domain adaptation for clinical negation detection? *Journal of the American Medical Informatics Association (JAMIA)*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017b. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*.

Aakanksha Naik and Carolyn Rosé. 2020. Towards open domain event trigger identification using adversarial domain adaptation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nghia Ngo, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Learning to select important context words for event detection. In *Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jason Phang, Thibault Fevry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics (TACL)*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. 2018. A dirt-t approach to unsupervised domain adaptation. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Matthew Sims, Jong Ho Park, and David Bamman. 2019. Literary event detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hieu Man Duc Trong, Duc Trong Le, Amir Pouran Ben Veyseh, Thuat Nguyen, and Thien Huu Nguyen. 2020. Introducing a new dataset for event detection in cybersecurity texts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2005. Ace 2005 multilingual training corpus. In *Technical report, Linguistic Data Consortium*.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Dustin Wright and Isabelle Augenstein. 2020. Transformer based multi-source domain adaptation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yifan Wu, Ezra Winston, Divyansh Kaushik, and Zachary Lipton. 2019. Domain adaptation with asymmetrically-relaxed distribution alignment. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Chen Yubo, Xu Liheng, Liu Kang, Zeng Daojian, Zhao Jun, et al. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Predicting in-hospital mortality by combining clinical notes with time-series data

Iman Deznabi, Mohit Iyyer, Madalina Fiterau
College of Information and Computer Sciences
University of Massachusetts Amherst
`{iman, miyyer, mfiterau}@cs.umass.edu`

Abstract

In intensive care units (ICUs), patient health is monitored through (1) continuous vital signals from various medical devices, and (2) clinical notes consisting of opinions and summaries from doctors which are recorded in electronic health records (EHR). It is difficult to jointly model these two sources of information because clinical notes, unlike vital signals, are collected at irregular intervals and their contents are relatively unstructured. In this paper, we present a model that combines both sources of information about ICU patients to make accurate in-hospital mortality predictions. We apply a fine-tuned BERT model to each of the patient's clinical notes. The resulting embeddings are then combined to obtain the overall embedding for the entire text part of the data. This is then combined with the output of an LSTM model that encodes patients' vital signals. Our model improves upon the state of the art for mortality prediction, attaining an AUC score of 0.9, compared to the previous 0.87, setting a new standard for mortality prediction on the MIMIC III benchmark.[1]

1 Introduction

One of the major costs in healthcare is critical care in intensive care units. A crucial aspect of critical care is mortality prevention. With advancements in healthcare, a patient's condition during an ICU visit is monitored through devices that measure many different vital signals, such as heart rate, systolic blood pressure, temperature, etc. Additionally, we have access to clinical notes written by medical professionals during the patient's stay. This data can be used to predict the condition of patients during their ICU stay, which can help in managing the expensive resources in hospitals and providing these services to patients who need them most in a more cost-effective way.

Most prior work has been focused on predicting patient health using the time-series data gathered from medical devices (Lipton et al., 2016; Che et al., 2018; Narayan Shukla and Marlin, 2020; Xu et al., 2018). More recent work also attempted to leverage clinical notes for making these predictions (Zhang et al., 2019; Ghorbani et al., 2020; Lee et al., 2020; Alsentzer et al., 2019). However, research on combining time series data with clinical notes for outcome prediction has been limited due to the irregularity of the clinical notes compared to the time series data, and the complexity of jointly modeling the two sources of information.

In this work, we propose a multimodal neural network that combines time-series data from medical devices with textual information from clinical notes to improve the prediction of in-hospital mortality. This task is defined as predicting whether a patient will die before getting discharged from hospital based on the first two days of data. Our model improves on the prior state of the art by using fine-tuned BERT-based models (Devlin et al., 2019) to encode clinical notes and integrates the resulting representations with time-series embeddings derived from a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997). We also show that fine-tuning clinical BERT models in isolation from other parts of the model and specifically for in-hospital prediction tasks brings additional performance improvements. Furthermore, separately modeling each clinical note and combining the resulting embeddings leads to our final model which supersedes the state-of-the-art in predicting "in-hospital mortality" in terms of AUC score.

---

[1] Our implementation is available at `https://github.com/Information-Fusion-Lab-Umass/ClinicalNotes_TimeSeries`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4026–4031
August 1–6, 2021. ©2021 Association for Computational Linguistics

2 Related work

Most of the previous work in mortality prediction in critical care is focused on utilizing patient vital signals as time-series data for making predictions. Harutyunyan et al. (2019) provided a benchmark and defined 4 tasks based on MIMIC III dataset (Johnson et al., 2016) for comparing these models. This paper used a recurrent neural network to obtain the predictions. Other models also use recurrent neural networks in different forms to predict outcomes in healthcare data (Liu and Chen, 2019; Suresh et al., 2018; Lipton et al., 2016). Furthermore, others (Che et al., 2018; Shukla and Marlin, 2019; Narayan Shukla and Marlin, 2020; Horn et al., 2020) used the irregular nature of the data over time in their models.

For clinical notes, some models used pretrained embedding models (Zhang et al., 2019; Chen et al., 2019), while some use various other machine learning techniques for outcome prediction on data collected from the ICU (Jin et al., 2018; Boag et al., 2018; Ghorbani et al., 2020).

More recently, BERT-based models have been adopted for this domain following their incredible success in Natural Language Processing (NLP). A number of studies trained and used BERT-based models for clinical applications (Lee et al., 2020; Huang et al., 2019; Darabi et al., 2020; Li et al., 2020; Alsentzer et al., 2019).

All these models only use one source of available data when predicting medical outcomes. However, Khadanga et al. (2019) showed the usefulness of combining time-series and clinical notes for ICU outcome prediction. They used a convolution neural network (CNN) on top of pretrained word embedding from (Zhang et al., 2019) for getting a representation of clinical notes and a long short-term memory network (LSTM) for embedding the time series part of the data. The two representations were then concatenated to make the predictions. Concurrently to our work, Yang et al. (2021) also showed the usefulness of combining time-series data with information from clinical notes. They used an LSTM model for the time-series part of the data and use a convolutional neural network with label-aware attention layer for the clinical notes. We will use the same intuition for combining the representation of clinical notes and the time-series part of the patients' data to improve the performance of in-hospital mortality prediction. In the next section we define the notations and our method architecture.

3 Method

In this section, we first introduce the notation for our task and then describe our proposed model. Each patient has several clinical notes collected during their stay. For a patient p with N clinical notes, we denote $C^{(p)} = \{c_{t_1}^{(p)}, c_{t_2}^{(p)}, ..., c_{t_N}^{(p)}\}$ for clinical notes collected at times $\{t_1, t_2, ..., t_N\}$. We also have time-series data collected during the patient's stay as $X_{1:T}^{(p)} = \{x_1^{(p)}, x_2^{(p)}, x_3^{(p)}, ..., x_T^{(p)}\}$ where $x_i^{(p)}$ is the data collected at index i. For simplicity, the patient index p will be dropped for the rest of the section.

Our model brings three major contributions compared to prior work in this area. First, we use a fine-tuned BERT-based model for modeling clinical notes as these new attention-based models trained on language models such as BERT have been shown to significantly outperform previous approaches in many NLP applications (Devlin et al., 2019) including clinical note prediction (Lee et al., 2020; Alsentzer et al., 2019; Huang et al., 2019). Second, we feed each clinical note of a patient separately to the text part of our model, and then combine the resulting embeddings to get the final representation of text part of the data. We argue that since the clinical notes are collected separately, and usually from different sources, this approach is preferable to concatenating the text of the clinical notes for each patient and then feeding this to the model. Finally, we also fine-tune the BERT-based models in isolation on "in-hospital mortality" task, which brings additional performance improvements.

The model architecture is shown in Figure 1. It consists of two parts. The first part obtains the embedding of clinical notes and combines them to form a final representation of all the clinical notes, for which we use the notation H_C. The other part of the model is responsible for encoding the time-series part of the data, which will be referred to as H_X. These representations will then be concatenated to form the final representation of the patient $H = H_C \oplus H_X$ where $\oplus$ is the concatenation operation. Finally, a softmax layer is applied on top of H to get the final predictions.

3.1 Clinical Notes Representations

Modeling clinical notes require capturing interactions between distant words (Huang et al., 2019)

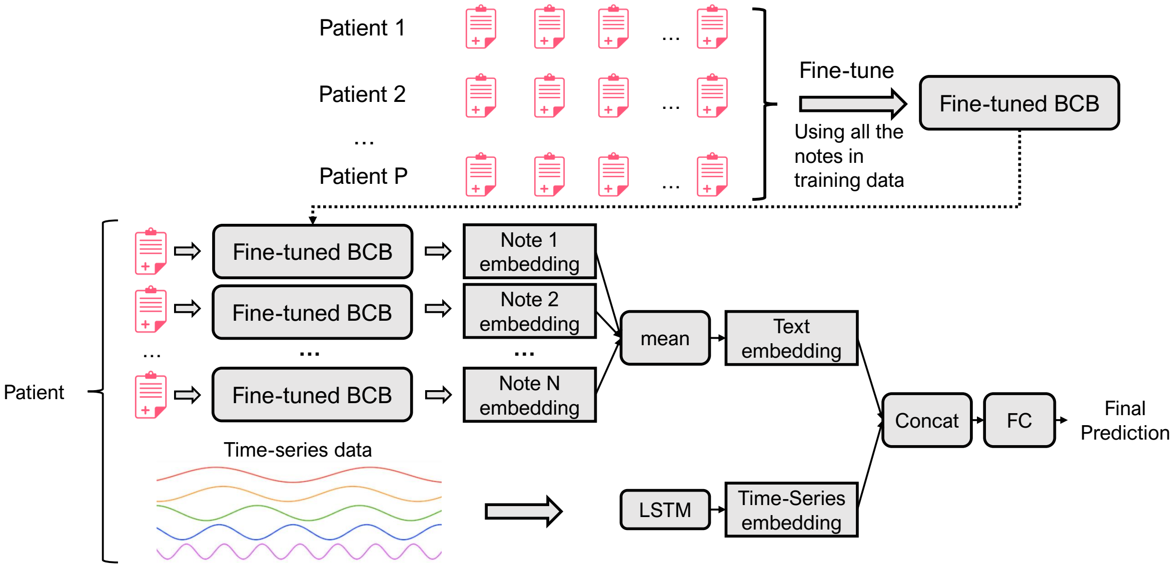

Figure 1: The model architecture. First, the Bio+Clinical BERT (BCB) (Alsentzer et al., 2019) model is fine-tuned for the in-hospital mortality prediction task using all the available notes in the training data. Then the average embedding of patient notes are combined with the time-series embedding to obtain the final patient representation. Then a fully connected layer is used to predict patient's in-hospital mortality.

and BERT-based models are designed to capture these interactions. Furthermore, Huang et al. (2019); Lee et al. (2020); Alsentzer et al. (2019) showed that these models outperform the traditional NLP models in this area. Considering this success we use a BERT-based model fine-tuned to clinical notes for getting the embedding of each clinical note of the patient. Specifically, we used the Bio+Clinical BERT model described in Alsentzer et al. (2019) which fine-tuned Bio BERT(Lee et al., 2020) on all the clinical notes in MIMIC III dataset.

After initializing our text model to Bio+Clinical BERT, we added a softmax layer on top of the classification token (CLS) output of each clinical note separately and the model is fine-tuned further on the task of predicting the in-hospital mortality of the patient. Finally, to obtain the overall text embedding of a patient, we take the average embedding of all the clinical notes available for the patient. This is shown in equation 1.

$$H_C = \frac{1}{N}\sum_{i=1}^{N} \mathrm{BCB}(c_{t_i}) \tag{1}$$

where the $\mathrm{BCB}(c_{t_i})$ function is the output of the CLS token of the fine-tuned Bio+Clinical BERT model when c_{t_i} is provided as an input to the model and H_C is the aggregated representation of all the clinical notes of the patient.

3.2 Time series

For the time-series part of the model, following the works by Harutyunyan et al. (2019) and Khadanga et al. (2019), we first limit the time-series data to first 48 hours of the patient stay, then we resample the time-series data to intervals of 1 hour. If there are any duplicate values in that hour for a variable, we use the most recent values. We use forward imputation for missing values. If no previous value is recorded we use the pre-set values for the features given in Harutyunyan et al. (2019). After pre-processing the time series data, we used an LSTM network. We input the whole time-series data of a patient ($X_{1:T}$) to the LSTM model and use the final hidden state as the representation of time-series part of the patient data.

$$H_X = \mathrm{LSTM}(X_{1:T}) \tag{2}$$

4 Results

In this section, we will show the results achieved on the in-hospital mortality prediction task on MIMIC III dataset. To ensure consistency with previous work and the benchmarks presented on Harutyunyan et al. (2019), we used the same preprocessing steps as defined by Harutyunyan et al. (2019) and split the patients into train, validation and test sets using the same splits. We also followed the preprocessing steps of Khadanga et al. (2019) to process the clinical notes and removed, from the dataset,

Type	Model	AUC
Only time-series	LSTM*	0.835
Only text	CNN*	0.831
	BERT	0.671
	BCB	0.760
	BCB FT	0.835
	BCB FT MN	**0.875**
Text + Time-series	CNN + LSTM*	0.867
	BERT + LSTM	0.840
	BCB + LSTM	0.851
	BCB FT + LSTM	0.873
	BCB FT MN + LSTM	**0.899**

Table 1: Area under the receiver operating characteristic (AUC) results on the held out data for all models including baselines. The results are averages of 5 different initializations of the models. The models indicated with a star (*) are using the same architecture as described in Khadanga et al. (2019). BCB is short for Bio+Clinical BERT (Alsentzer et al., 2019), FT means fine tuned separately for in-hospital mortality task. The baseline models only use the first available clinical note while models indicated with MN (multi-note) use all the available clinical notes in first 48 hours of a patient's ICU stay.

patients who do not have any associated medical notes. We also only used the clinical notes collected at first 48 hours of patients' stay. After these steps, our dataset consisted of 11579 records in the training set, 2570 in the validation set, and 2573 in the test set.

The time-series part of the model uses all 17 medical variables (e.g., heart rate, height, and Glucose) recorded during the first 48 hours of the patients' stay. Since in-hospital mortality prediction is a binary classification task with unbalanced classes (only around 10% of the patients suffered mortality in this dataset), the area under the receiver operating characteristic (AUC) is used for evaluating our models. The best performing model uses one layer of LSTM cells with 256 units, an Adam optimizer (Kingma and Ba, 2015) with initial learning rate of 2×10^{-5} for training and a weight decay of 0.01. The models are implemented with the Pytorch library (Paszke et al., 2019) and we run all the experiments 5 times with different initialization and report the mean of the results.

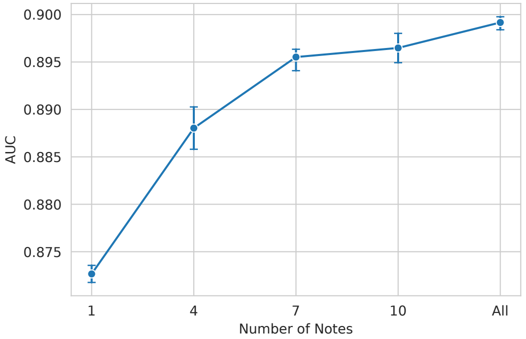

Figure 2: Increase in full model performance as more clinical notes are included in order of time for each patient. The Y axis shows the AUC score of the model on the test data and the X axis shows the number of included clinical notes. Each bar indicates 95% confidence interval over 5 different runs of the model.

Table 1 summarizes the results achieved by our models and the baseline models that use only time-series data, use a simple BERT instead of the Bio+Clinical BERT model, and the results with and without fine tuning Bio+Clinical BERT model further for the in-hospital mortality prediction task. The models which use both time-series and text data are also shown. Moreover, the results are compared with Khadanga et al. (2019) which, to the best of our knowledge, is the state-of-the-art for in-hospital prediction task in MIMIC III benchmarks. Our final model, which uses fine-tuned clinical BERT model for the text part of the data and an LSTM model for time-series part (shown in Figure 1) significantly outperforms the baseline models as well as the state-of-the-art models.

To assess the value of using multiple notes for each patient, we experimented by including various numbers of clinical notes for each patient in our final model. The AUC results are shown in Figure 2. It is apparent that the model's performance improves with more notes, and the best performance is achieved when all the notes are used.

Although we lose some information contained in the clinical notes by truncating them to fit into the maximum acceptable length of the model, using the entire text of the clinical notes by segmenting them into chunks and taking the average did not yield significant improvements in the performance of our final model. In our experiments, providing all chunks either by segmenting the concatenated text of the clinical notes or individual notes separately only improved the final AUC score from 0.899 to a maximum of 0.902 while taking approximately twice as long to train.

5 Conclusions and Future Work

Improved prediction of the admission outcome in Intensive Care Units (ICUs) can be tremendously valuable in supporting decisions in clinical diagnosis. Previous work mostly focused on using the patients' vital sign information recorded during their stay as time-series data for these predictions. However, the clinical notes written by healthcare workers include important information on both the history and current conditions of the patients in the hospital, which can be leveraged to improve these predictions significantly. In this work, we proposed a novel method for leveraging the information available both in clinical notes and time-series data. Our method consists of fine-tuning Bio+Clinical BERT model for in-hospital mortality prediction task and then combining it with the available time-series data. Finally, we showed that using all the clinical notes available significantly enhances the ability to make these kinds of predictions.

In the future, we plan to improve the performance of the model by improving the time-series part of the model from an LSTM network to a model that can leverage irregular time-series data. Furthermore, the same model architecture can be used for other tasks discussed in the literature for such medical datasets such as physiologic decompensation or forecasting length of stay.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

Willie Boag, Dustin Doss, Tristan Naumann, and Peter Szolovits. 2018. What's in a note? unpacking predictive value in clinical note representations. *AMIA Summits on Translational Science Proceedings*, 2018:26.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12.

Qingyu Chen, Yifan Peng, and Zhiyong Lu. 2019. Biosentvec: creating sentence embeddings for biomedical texts. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–5. IEEE.

Sajad Darabi, Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. 2020. Taper: Time-aware patient ehr representation. *IEEE Journal of Biomedical and Health Informatics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ramin Ghorbani, Rouzbeh Ghousi, Ahmad Makui, and Alireza Atashi. 2020. A new hybrid predictive model to predict the early mortality risk in intensive care units on a highly imbalanced dataset. *IEEE Access*.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. 2019. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. 2020. Set functions for time series. In *International Conference on Machine Learning*, pages 4353–4363. PMLR.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

Mengqi Jin, Mohammad Taha Bahadori, Aaron Colak, Parminder Bhatia, Busra Celikkaya, Ram Bhakta, Selvan Senthivel, Mohammed Khalilia, Daniel Navarro, Borui Zhang, et al. 2018. Improving hospital mortality prediction with medical named entities and multimodal learning. *arXiv preprint arXiv:1811.12276*.

A Johnson, T Pollard, and R Mark III. 2016. The mimic-iii clinical database. *PhysioNet. doi*, 10:C2XW26.

Swaraj Khadanga, Karan Aggarwal, Shafiq Joty, and Jaideep Srivastava. 2019. Using clinical notes with time series data for ICU management. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6432–6437, Hong Kong, China. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and

Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. Behrt: transformer for electronic health records. *Scientific Reports*, 10(1):1–12.

Zachary C. Lipton, David C. Kale, Charles Elkan, and Randall Wetzel. 2016. Learning to diagnose with LSTM recurrent neural networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Jiexi Liu and Songcan Chen. 2019. Non-stationary Multivariate Time Series Prediction with Selective Recurrent Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11672 LNAI, pages 636–649.

Satya Narayan Shukla and Benjamin M Marlin. 2020. Multi-Time Attention Networks for Irregularly Sampled Time Series. Technical report.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Satya Narayan Shukla and Benjamin M. Marlin. 2019. Interpolation-Prediction Networks for Irregularly Sampled Time Series. *7th International Conference on Learning Representations, ICLR 2019*.

Harini Suresh, Jen J Gong, and John V Guttag. 2018. Learning tasks for multitask learning: Heterogenous patient populations in the icu. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 802–810.

Yanbo Xu, Siddharth Biswal, Shriprasad R Deshpande, Kevin O Maher, and Jimeng Sun. 2018. Raim: Recurrent attentive and intensive model of multimodal patient monitoring data. In *Proceedings of the 24th ACM SIGKDD international conference on Knowledge Discovery & Data Mining*, pages 2565–2573.

Haiyang Yang, Li Kuang, and FengQiang Xia. 2021. Multimodal temporal-clinical note network for mortality prediction. *Journal of Biomedical Semantics*, 12(1):1–14.

Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. 2019. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):1–9.

Sequence Models for Computational Etymology of Borrowings

Winston Wu **Kevin Duh** **David Yarowsky**
Center for Language and Speech Processing
Department of Computer Science
Johns Hopkins University
`{wswu,yarowsky}@jhu.edu,kevinduh@cs.jhu.edu`

Abstract

We computationally model the processes of word borrowing from a donor word to an incorporated word, and vice versa, by answering two questions: (1) what does a word look like incorporated into another language, and in the opposite direction (2) where did a word come from? We employ neural sequence models, focusing on six specific borrowing relations: calques, partial calques, semantic loans, phono-semantic matches, transliterations, and generic borrowings. We experiment with several model variants, including LSTM encoder-decoders, copy attention, and Transformers. In both directions, we find that an LSTM model can beat strong baselines, with the quantity of data strongly influencing model performance.

1 Introduction

Words are borrowed into a language through various processes. For example, the English *internet* was incorporated into Welsh as *rhyngrwyd* (rhyng- 'between' + rhwyd 'net') through a calqueing process where each component is translated literally. In contrast, the English *chimpanzee* became the Welsh *tsimpansî* through a process of sound correspondences.

Borrowing is prevalent across the world's languages, and modeling how and from where words enter a language are interesting but understudied tasks under the umbrella of computational etymology (Wu and Yarowsky, 2020a). This is a relatively new field with many downstream applications. Perhaps the most salient is lexicon expansion: more comprehensive dictionaries will enable better communication between cultures as well as better training material for machine translation systems. Computational etymology is also important for historical linguistics, whose focus is on discovering the relationships between languages and their words. An accurate model of word borrowing can also be a boon for language preservation and revitalization, where models can help coin neologisms for modern terms.

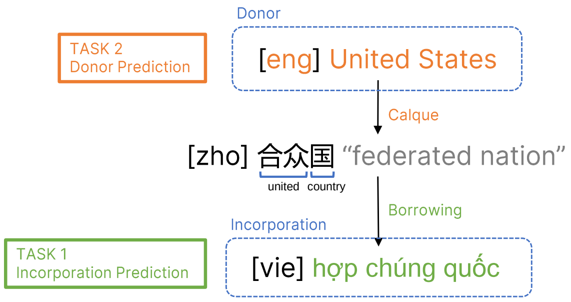

Figure 1: Two borrowing prediction tasks. The predicted output for each task is in green and orange, respectively.

Owing to recent successes of machine translation models for similar tasks (Tsvetkov and Dyer, 2015; Gorman et al., 2020; Wu and Yarowsky, 2020a,b), this paper investigates the application of neural sequence-to-sequence models for the task of etymology prediction. Specifically, we focus on word borrowings, where a word enters a language via a non-related donor language.[1] Whereas inherited words and cognates tend to follow regular sound shifts and can be modeled well with transliteration models (Beinborn et al., 2013; Wu and Yarowsky, 2018b), words borrowed from unrelated languages undergo various processes (Section 3) that may not preserve the structure or phonetics of the original word.

We propose to model borrowings in two tasks (Figure 1), motivated in Section 4. In Task 1, given a donor word and etymological relation, can we predict the form of the incorporated word in the borrowing language? In the opposite direction, in

---

[1] This is in contrast to other etymological relations, such as inheritance, where words enter through a related language, e.g. from Latin to French.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4032–4037
August 1–6, 2021.

Task 2, given the incorporated word, can we predict the donor word and language? Our experiments across several experimental scenarios on these two tasks using data from Wiktionary indicate that modeling borrowings is a challenging task with much room for future research.

2 Related Work

Though the tasks defined in this paper are new, there are several related threads of work. In the task of *cognate transliteration*, a system is trained to generate cognates in a different language (Beinborn et al., 2013; Wu and Yarowsky, 2018b). This paper uses a multilingual cognate transliteration approach applied specifically to borrowings. Similar approaches have also been applied to the task of proto-language reconstruction (Meloni et al., 2021). Related to cognate transliteration is the task of *grapheme-to-phoneme conversion*, which has a long history of research. Cognate transliteration can be viewed as G2P across languages, where the words are cognates, for example in the case of names (Waxmonsky and Reddy, 2012; Wu et al., 2018; Wu and Yarowsky, 2018a). Recently, researchers have studied massively multilingual versions of these tasks, where single (neural) models are trained on the combination of hundreds of languages (e.g. Deri and Knight, 2016; Gorman et al., 2020; Lewis et al., 2020).

3 Data

We extract etymology information from the English edition of Wiktionary using Yawipa (Wu and Yarowsky, 2020a), a recent Wiktionary parser. We focus on six specific types of borrowings (whose Wiktionary label is in monospaced font below) across a spectrum of semantic and phonetic fidelity:

- `calque`: Also called a loan translation. Components of the original word are literally translated into the target language, e.g. the English *brainwash*, from the Chinese 洗脑 *xi* ‘wash’ + *nao* ‘brain’.
- `partial calque`: A calque where not every component is translated, e.g. the English *apple strudel*, from the German *Apfelstrudel*.
- `semantic loan`: A sense extension is borrowed onto an existing word, e.g. the French *souris* ‘mouse’, which borrowed the computing sense from the English *mouse*.

Lang	Count	%
eng	23,142	0.15
lat	18,713	0.12
fra	17,556	0.11
spa	7,123	0.05
ara	6,508	0.04
san	6,393	0.04
grc	6,122	0.04
deu	5,390	0.04
rus	5,109	0.03
ita	4,660	0.03

Table 1: Distribution of top 10 languages extracted from Wiktionary.

- `psm`: Phono-semantic matching. Components of the original word are replaced with phonetically and semantically similar words, e.g. 声纳 *sheng* ‘sound’ + *na* ‘receive’, from the English *sonar*.
- `transliteration`: A deterministic process of writing script conversion that seeks to preserves a word’s orthography.
- `bor`: A generic borrowing category. The overwhelming majority of borrowings in Wiktionary are labeled as such. In this paper, we distinguish between `bor`, this relation as annotated in Wiktionary, and “borrowing”, the word formation process encompassing these six relations.

The data we extracted consists of over 150K ground-truth annotated borrowing relationships, spanning a total of 837 languages. The top 10 languages are shown in Table 1. Note that only 101 languages have more than 100 entries, and 260 languages have more than 10 entries. In this work, we are also specifically interested in the long tail of low-resource languages. The distribution of borrowing relations is shown in Figure 2. Note the log scale, and the fact that that the majority class (`bor`) comprises 96% of the entire dataset, which motivates several of our experimental variants.

4 Tasks

We first establish our terminology for borrowings: we say etymology is directed relation between a donor word and an incorporated word.[2] We experi-

---

[2] We eschew the established terms “loanword” and “borrowing” because loaning and borrowing imply an obligation to return the item being borrowed. In contrast, “borrowed” words are fully incorporated into the language.

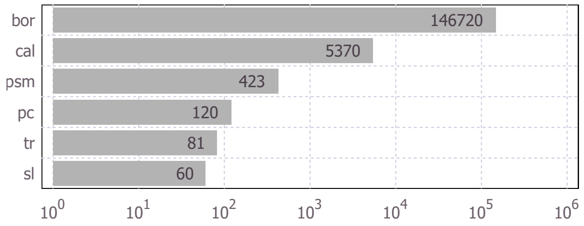

Figure 2: Distribution of borrowing relations.

ment on two tasks in etymology prediction:

4.1 Task 1: Incorporation Prediction

Given a donor word and a target language, how would the word be incorporated into that language? And by what means? This task is motivated by a real-world example: when deep learning was gaining popularity, researchers were considering how to best render the term into Japanese. Should it be a loanword and written in katakana (ディープラーニング *dīpurāningu*), or translated using a calque (深層学習 *shinsō gakushū* 'deep' + 'learning')? Besides terminology standardization, this task has applications in language revitalization and unknown word translation.

4.2 Task 2: Donor Prediction

In the opposite direction, given a word, from where and how did it come into the language? If we view Wiktionary as a directed graph, where the nodes are words and the edges are etymological relationships, there are missing edges. The task is to reconstruct these missing edges. As Wiktionary is a human-annotated resource, there is much variance in the quality and completeness of annotations, and good performance on this task can help fill in etymology even in high-resource languages like English.

5 Experiments

To tackle these two tasks, we employ character neural sequence-to-sequence models. For Task 1, predicting the incorporated word, the input is a sequence containing: the donor language, each character of the donor word, the etymological relation, and the target language. The output is the characters of the incorporated word.

In: `eng c a b b a g e bor abe`
Out: `k a b i j`

For Task 2, the input is a sequence containing the word's language and each character of the word, while the output is the donor language, donor word characters, and relation.

In: `abe k a b i j`
Out: `eng c a b b a g e bor`

For Task 1, we experiment with separate LSTM models trained for each borrowing relation (LSTM-sep), a single multi-task LSTM model trained on the combined data (LSTM), the same model trained with both the source and target data preprocessed by the unigram SentencePiece method (Kudo and Richardson, 2018) with a vocabulary size of 4000 (LSTM-spm), the same model with copy attention (See et al., 2017) (LSTM-copy), a Transformer (Vaswani et al., 2017) model (TF), and an ensembling method (Ensemble). This method is a score-based voting procedure that combines the output of the LSTM-sep, LSTM, and TF models. Each model gives 5 votes for their top prediction, 4 votes for their second place prediction, and so on (1 vote for fifth place). For each test instance, the votes are tallied up, and the prediction with the highest number of votes is the prediction of the ensemble. Ties are broken by picking the prediction with the highest model decoder score among all the models.

For Task 2, we experiment with a baseline LSTM model and the same model with copy attention.

All models were trained using the OpenNMT-py framework (Klein et al., 2020). The LSTM models are two-layer encoder-decoders with 500-dimension hidden state, trained with the ADAM optimizer. The Transformer model has a 6-layer encoder and decoder with 8 heads, trained with ADAM with learning rate scheduling. For reproducibility, we provide the training scripts which include the full model details. Accounting for the extreme imbalance in our dataset, we performed a stratified split of the dataset into a 80-10-10 train-dev-test split, where each split contains the same proportion of languages and borrowing relations.

6 Results and Analysis

6.1 Task 1

We evaluate each model on a held-out 15,288 example test set. Table 2 presents character BLEU (computed with SacreBLEU Post (2018)) as well as accuracy and character edit distance from the gold (CED). We also report 5-best results for accuracy (was the correct answer in the top 5 results?) and CED (within the top 5 results, what is the minimum edit distance to the correct answer?)

At a cursory glance, the single models trained on all the data performs slightly better compared to the separate relation-specific models, following a

Model	BLEU	Acc	CED	5Acc	5CED
LSTM-sep	53.77	20.00	2.42	33.51	1.82
LSTM	55.83	21.43	2.31	34.98	1.71
LSTM-copy	55.90	19.92	2.32	34.46	1.69
LSTM-spm	45.62	10.68	2.85	20.31	2.13
Transformer	61.30	22.19	2.06	41.54	1.43
Ensemble	60.32	25.67	2.05	49.24	1.18

Table 2: Results for Task 1. Acc is accuracy (higher is better), CED is average character edit distance (lower is better). 5 indicates 5-best results.

trend of multi-task training performing better than models trained on a single task. The Transformer model performs the best, likely due to its innovative attention mechanism that has proven successful in other tasks. However, by examining the results for each borrowing relation, we see that the successes of the models are largely on the `bor` relations. All the models perform poorly in correctly predicting any non-`bor` relations, though we find that the calque-specific model performs slightly better than the jointly trained LSTM on calques. For example, the separate calque model correctly predicted the German *vollschlank* borrowed into Dutch as *volslank*, which the LSTM model could not do. And even when it generates incorrect answers, often the predictions look like "good attempts" at calqueing. For example, the French *Pays d'en Haut* gets translated as *Land of the Roud* (correct is *upcountry*), whereas the jointly trained models often do character substitutions instead.

Copy attention (LSTM-copy), which allows the model the option to copy characters from the source, was intended to help the model with similarly spelled borrowings, but overall it did not perform as well as a simple LSTM model. The subword model (LSTM-spm) also unexpectedly did not perform well. The goal of using subwords was to encourage the model to translate larger character sequences, the idea being that translational relations such as calques would consist of two subwords rather than several individual characters. Indeed, the LSTM-spm model treats most words as calques, often translating when it should instead perform character substitutions or sound shifts. Ensembling of three models' outputs is a simple but effective method resulting in a large increase in prediction performance. The score-based voting effectively combines the strengths of individual models, especially when all models have the same word in their n-best predictions.

Error Analysis Due to the small quantities of available training data for partial calques, semantic loans, phonosemantic matches, and transliterations, the models cannot accurately learn to predict words incorporated by the aforementioned processes. This data shortage is exacerbated for the separately trained systems. Models largely treat these translational borrowings as generic `bor`s and perform character substitutions and sound shifts. This approach, exemplified by cognate transliteration systems, works for the majority of test examples, because `bor`s are essentially cognates with small edit distance. All phonosemantic matches are Chinese, so models will output Chinese characters, but due to the sparsity of the characters, the model cannot produce the correct answer. For the remainder of this analysis, we will focus on `bor` and `cal` as the main two borrowing relations. We find all the models show similar patterns of prediction, so the following examples are from the multi-task LSTM model.

In many cases, the incorporated word is similar to the donor, so the model can correctly predict the borrowing. For example, for the Latin *vanitas* borrowed into French, the model predicts *vanita*; the correct *vanité* is its second choice. The model can also handle different writing scripts. For example, it correctly predicts the Greek πυριτις borrowed into Latin as *pyritis*. Unfortunately, sound shifts do not work for the other borrowing relations, like calques, that require translation of morphemes. In many cases, the model does not seem to distinguish between non-`bor` relations and merely performs sound shifting. For example, our model predicts that the English *shopping center* calqued into Afrikaans is *schoppingsentre* (correct is *winkelsentrum*).

When encountering calques, the model sometimes recognizes that it should translate rather than transliterate. However, the lack of sufficient training data prevents the model from learning to accurately translate component morphemes. For example, our model predicts the English *download* calqued into German is *Dunnleut* (correct is *herunterladen*). Here, we see that the model picks up on the fact that German words tend to start with a capital letter, though in this case the word in question is a verb which does not need capitalization. We also find that the model cannot get the word order correct when languages have different adjective-noun ordering. For example, our model incorrectly predicts that the French *mariage blanc* borrowed into English is *marriage mank* (correct is *white marriage*).

Broken down by language, our data contains numerous low-resource languages, many of which have just 1-10 words. Training a single model on such data for a single language would yield low performance, but our massively multilingual borrowing models can successfully handle many of these low-resource languages.

6.2 Task 2

For Task 2, we follow Wu and Yarowsky (2020a), who used an LSTM model to predict both the language and formation mechanism of a word. While they attempted to predict broader categories of inheritance vs borrowing, we focus on six specific borrowing relations. Because many borrowings have small edit distance, we also employed an LSTM model with copy attention. This model's performance was slightly worse than the baseline LSTM, a trend we also observed in Task 1. This indicates that borrowings are fundamentally different from inherited and cognate words, where copy attention models have seen good performance. Results grouped by word, language, and relation are presented in Table 3.

The models for Task 2 are inherently multi-task: they must predict the donor language, donor word, as well as the relation. As such, prediction of donor language and relation can be evaluated as classification tasks. We found that our models were able to generate valid languages and relations in 98% cases, showing that sequence-to-sequence models can also be successful in classification tasks.

We briefly analyze the errors of the LSTM model. Perhaps unsurprisingly, the model gets over 96% accuracy on predicting the relation by always guessing `bor`, the majority class. Yet it is able to beat a strong majority baseline (always predicting `bor`, the majority class). Our model is also able to successfully predict the language of the borrowing in almost half of the test instances (guessing the majority donor language, English, would only achieve 14.8% accuracy). Thus a word's language and spelling provide sufficient information for identifying how and from where it entered the language. In terms of errors, we find some instances where the model predicts a donor language that is actually related to the correct language. For example, the Dutch *tabak* is borrowed from Spanish *tabaco*, rather than our model's French *tabac*, and many Dutch words originally from English were predicted to come from German, and vice versa. We also see several words like English *specify* were predicted to come from French, but are actually from Old French. Future work can address a custom loss function that gives "partial credit" to such predictions rather than marking them as completely incorrect.

Model	Rel	Lang	Word	CED
Majority	96.0	14.8	–	–
LSTM	96.1	47.9	23.2	2.9
LSTM-Copy	96.1	47.7	20.8	3.0

Table 3: Results for Task 2: 1-best accuracy grouped by Relation, Language, and Word. CED is average character edit distance for Word prediction.

In terms of word prediction, the seemingly low accuracy of the model is not discouraging. Supported by the low character edit distance, we see many examples where the model's prediction is close enough to be recognized by a human. For example, the Chinese 阿卡拉 is borrowed from English *a cappella*, but our model predicts *acapara*, and the Jersey French *thiâtre* was predicted to be borrowed from Latin *thiatrum* (correct is *theātrum*). When providing new entries to an impoverished etymology dictionary, our prediction model can suggest possible etymology and even plausible unknown word forms, which can then be verified by a human lexicographer.

7 Conclusion

We model word borrowings from a donor to an incorporated word, and vice versa, using neural sequence models in a variety of experimental scenarios. We find that a single model trained to predict multiple types of borrowings performs better than separate models trained for each borrowing. A Transformer model performs better than an LSTM model, and a simple ensembling method results in superior performance, though the amount of training data is a limiting factor in the performance of these models. Predicting the donor language and word is a slightly easier task, where our LSTM model is able to beat a strong majority baseline. Source code for reproducing our experiments is available at `https://github.com/wswu/borrowings`.

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate production using character-based machine translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 883–891, Nagoya, Japan. Asian Federation of Natural Language Processing.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 399–408, Berlin, Germany. Association for Computational Linguistics.

Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya McCarthy, Shijie Wu, and Daniel You. 2020. The SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 40–50, Online. Association for Computational Linguistics.

Guillaume Klein, François Hernandez, Vincent Nguyen, and Jean Senellart. 2020. The OpenNMT neural machine translation toolkit: 2020 edition. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 102–109, Virtual. Association for Machine Translation in the Americas.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Dylan Lewis, Winston Wu, Arya D. McCarthy, and David Yarowsky. 2020. Neural transduction for multilingual lexical translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4373–4384, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Carlo Meloni, Shauli Ravfogel, and Yoav Goldberg. 2021. Ab antiquo: Neural proto-language reconstruction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4460–4473, Online. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Yulia Tsvetkov and Chris Dyer. 2015. Lexicon stratification for translating out-of-vocabulary words. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 125–131, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Sonjia Waxmonsky and Sravana Reddy. 2012. G2P conversion of proper names using word origin information. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 367–371, Montréal, Canada. Association for Computational Linguistics.

Winston Wu, Nidhi Vyas, and David Yarowsky. 2018. Creating a translation matrix of the Bible's names across 591 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Winston Wu and David Yarowsky. 2018a. A comparative study of extremely low-resource transliteration of the world's languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Winston Wu and David Yarowsky. 2018b. Creating large-scale multilingual cognate tables. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Winston Wu and David Yarowsky. 2020a. Computational etymology and word emergence. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3252–3259, Marseille, France. European Language Resources Association.

Winston Wu and David Yarowsky. 2020b. Wiktionary normalization of translations and morphological information. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4683–4692, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Learning Contextualized Knowledge Structures for Commonsense Reasoning

Jun Yan[1], **Mrigank Raman**[2], **Aaron Chan**[1], **Tianyu Zhang**[3], **Ryan Rossi**[4],
Handong Zhao[4], **Sungchul Kim**[4], **Nedim Lipka**[4], **Xiang Ren**[1]
University of Southern California[1], IIT Delhi[2], Tsinghua University[3],
Adobe Research[4]
`{yanjun, chanaaro, xiangren}@usc.edu`, `mt1170736@iitd.ac.in`,
`zhang-ty17@mails.tsinghua.edu.cn`, `{ryrossi, hazhao, sukim, lipka}@adobe.com`

Abstract

Recently, knowledge graph (KG) augmented models have achieved noteworthy success on various commonsense reasoning tasks. However, KG edge (fact) sparsity and noisy edge extraction/generation often hinder models from obtaining useful knowledge to reason over. To address these issues, we propose a new KG-augmented model: Hybrid Graph Network (HGN). Unlike prior methods, HGN learns to jointly contextualize extracted and generated knowledge by reasoning over both within a unified graph structure. Given the task input context and an extracted KG subgraph, HGN is trained to generate embeddings for the subgraph's missing edges to form a "hybrid" graph, then reason over the hybrid graph while filtering out context-irrelevant edges. We demonstrate HGN's effectiveness through considerable performance gains across four commonsense reasoning benchmarks, plus a user study on edge validness and helpfulness.[1]

1 Introduction

Commonsense reasoning (CSR) is essential for natural language understanding (NLU) systems to function effectively in the real world (Apperly, 2010). For example, to answer the question in Figure 1, one must already know that *printing requires using paper*. Yet, since commonsense knowledge is self-evident to humans, it is rarely stated in natural language (Gunning, 2018). This makes it hard for neural pre-trained language models (PLMs) (Devlin et al., 2019) to learn commonsense knowledge from corpora alone (Marcus, 2018).

Unlike raw text corpora, knowledge graphs (KGs) can provide structured commonsense facts (edges) of the form `(concept1, relation, concept2)` (Speer et al., 2017). Hence, many recent CSR models augment the PLM with a KG, allowing such *KG-augmented models* to make predictions via multi-hop reasoning over the KG (Lin et al., 2019; Bosselut and Choi, 2019).

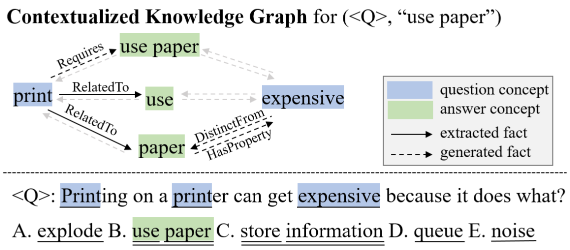

Figure 1: **KG-Augmented Commonsense QA.** Predicting the correct answer ("use paper") requires commonsense facts like `(print, Requires, paper)` and `(paper, HasProperty, expensive)`, which are not given in the question and candidate answers. HGN uses facts extracted from the KG, *e.g.*, `(print, RelatedTo, use)`, but also generates new facts, eventually upweighting relevant ones, *e.g.*, `(print, Requires, use paper)` and `(paper, HasProperty, expensive)`, while downweighting irrelevant ones, e.g., `(use, ?, expensive)`.

Despite the growing success of KG-augmented models, obtaining helpful KG facts for a given task instance remains challenging. Existing models assume using either KG-extracted edges (Lin et al., 2019; Ma et al., 2019; Feng et al., 2020; Yasunaga et al., 2021), PLM-generated edges (to address KG edge sparsity) (Bosselut and Choi, 2019), or a late fusion of both (Wang et al., 2020) is sufficient. Both extraction and generation can produce unhelpful edges, so the model must decide which edges to focus on during reasoning. Since extracted and generated edges are derived from the same set of concepts (nodes), modeling the interactions between extracted and generated edges jointly within a shared KG structure could provide stronger signal for identifying contextually relevant edges. However, current models do not leverage this information.

---

[1] Our code and data can be found at `https://github.com/INK-USC/HGN`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4038–4051
August 1–6, 2021.

In response, we propose a new KG-augmented model: **Hybrid Graph Network (HGN)**. Unlike prior models, HGN learns to jointly contextualize extracted and generated knowledge by reasoning over *both* within a unified graph structure. Given the task input (i.e., context) and an extracted KG subgraph, HGN is trained to generate embeddings for the subgraph's missing edges to form a "hybrid" graph, then reason over the graph (to update model parameters) while filtering out context-irrelevant edges. HGN achieves this primarily through edge reweighting, which downweights irrelevant edges, and edge-weighted message passing, which attenuates irrelevant edges' impact on reasoning.

Our extensive experiments demonstrate that HGN improves performance over all baselines across four CSR benchmarks. In particular, among comparable methods, HGN ranks first on the CommonsenseQA (Talmor et al., 2019) and OpenbookQA (Mihaylov et al., 2018) leaderboards. Plus, our user studies show that humans find HGN-filtered edges to be more valid and helpful than the heuristically extracted edges used in prior work.

2 Problem Statement

We consider CSR tasks, like question answering (QA), which can benefit from commonsense KGs. To solve CSR tasks, we focus on KG-augmented models, where a PLM is augmented with a commonsense KG. Given a CSR task, let x be the task's text input, f be the model, and $f(x)$ be the model output. We denote a KG as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$. $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{E}$ are the sets of nodes (concepts), relations, and edges (facts), respectively, in the KG. An edge is a directed triple of the form $e = (h, r, t) \in \mathcal{E}$, where $h \in \mathcal{V}$ is the head node, $t \in \mathcal{V}$ is the tail node, and $r \in \mathcal{R}$ is the relation between h and t. Let $[\cdot, \cdot]$ denote concatenation of text or vectors.

As illustrated in Figure 2, a KG-augmented model f has three main components: text encoder f_{text}, graph encoder f_{graph}, and scoring function f_{score}. First, $\mathbf{s} = f_{\text{text}}(x; \theta_{\text{text}})$ is the encoding of x, where f_{text} is usually a Transformer PLM. Second, as supporting evidence, a x-specific graph $\mathcal{G}' = (\mathcal{V}', \mathcal{R}', \mathcal{E}')$ is constructed from $\mathcal{G}$ (Figure 1). Typically, this is done via heuristic extraction by selecting $\mathcal{V}' \subseteq \mathcal{V}$ as the concepts mentioned in x, $\mathcal{R}' \subseteq \mathcal{R}$ as the relations between concepts in $\mathcal{V}'$, and $\mathcal{E}' \subseteq \mathcal{E}$ as the edges involving $\mathcal{V}'$ and $\mathcal{R}'$. If $\mathcal{G}$ does not provide enough knowledge to build a good $\mathcal{G}'$, then new edges are sometimes added to $\mathcal{G}'$ using a PLM-based generator (Wang et al., 2020). We call $\mathcal{G}'$ the contextualized KG. $\mathbf{g} = f_{\text{graph}}(\mathcal{G}', \mathbf{s}; \theta_{\text{graph}})$ is then the joint encoding of $\mathcal{G}'$ and $\mathbf{s}$. Third, the model output is computed as $f(x) = f_{\text{score}}([\mathbf{s}, \mathbf{g}]; \theta_{\text{score}})$, where f_{score} is usually a multilayer perceptron (MLP). Existing KG-augmented models mainly differ in their design of f_{graph}, reasoning over the KG through message passing (Schlichtkrull et al., 2018a; Feng et al., 2020; Yasunaga et al., 2021) or edge/path aggregation (Lin et al., 2019; Bosselut and Choi, 2019; Ma et al., 2019).

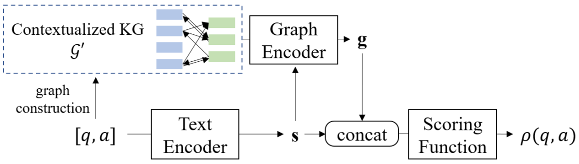

Figure 2: **High-level schematic of a typical KG-augmented model for CSR.** In KG-augmented models, text encoder f_{text} tends to be a Transformer PLM, and scoring function f_{score} is usually an MLP. Meanwhile, KG-augmented models generally vary more in their graph encoder f_{graph} and graph construction.

While KG-augmented models can be applied to any CSR task involving KGs (e.g., natural language inference), we consider multi-choice QA in this work. Given a question q and set of candidate answers $\{a_i\}$, the QA model's goal is to predict a plausibility score $\rho(q, a)$ for each $a \in \{a_i\}$, so that the highest score is predicted for the correct answer. To use KG-augmented models for commonsense QA, we set $x = [q, a]$ and $\rho(q, a) = f(x)$.

3 Hybrid Graph Network (HGN)

3.1 Overview

As illustrated in §2 and Figure 2, given question-answer pair (q, a) for an instance of the multi-choice QA task, the KG-augmented QA model first obtains a (q, a)-contextualized KG $\mathcal{G}'$ via the full KG $\mathcal{G}$. Edges in $\mathcal{G}'$ can be extracted directly from $\mathcal{G}$ or generated using a PLM-based generator (Wang et al., 2020; Bosselut et al., 2019). Then, the model transforms (q, a) and $\mathcal{G}'$ into text encoding $\mathbf{s}$ and graph encoding $\mathbf{g}$, respectively. Finally, $\mathbf{s}$ and $\mathbf{g}$ are used to predict (q, a)'s plausibility.

However, a contextualized KG may have low knowledge recall or precision, hindering the QA model's access to relevant knowledge. Low recall can stem from missing edges in $\mathcal{G}$, low precision can be the result of bad annotations in $\mathcal{G}$,

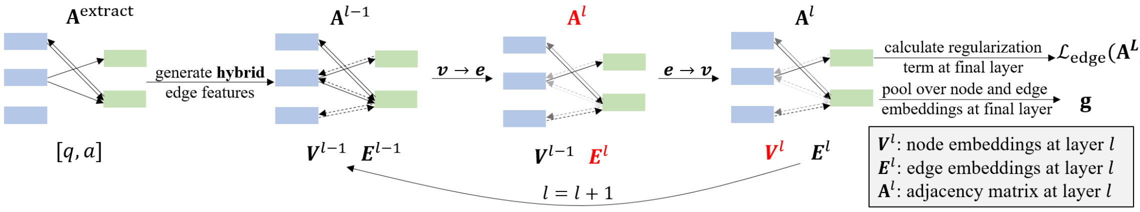

Figure 3: **Overview of HGN.** After building a hybrid graph of extracted and generated edges (§3.2), HGN reasons over the hybrid graph by updating the node embeddings $\mathbf{V}$, hybrid edge embeddings $\mathbf{E}$, and adjacency matrix $\mathbf{A}$ at each layer ℓ (§3.3). **Darker** edges indicate higher weights. **Red** variables are updated in the previous step.

and both can be caused by noisy edge extraction or generation when building $\mathcal{G}'$. HGN addresses these issues by reasoning over both extracted and generated edges within a unified graph structure. To improve recall, HGN generates new edges via a PLM-based generator, then initializes a hybrid contextualized KG containing both extracted and generated edges. Note that edge generation is generally (q, a)-agnostic and may produce irrelevant edges that hurt knowledge precision. To improve precision, HGN learns to reweight edges in the hybrid graph and reason over the hybrid graph via edge-weighted message passing. This is akin to learning the hybrid graph's structure and reduces the impact of irrelevant edges on reasoning. Additionally, to further encourage downweighting of noisy edges during reasoning, HGN is trained with entropy regularization on the learned edge weights.

The overall learning objective of HGN is defined as $\mathcal{L} = \mathcal{L}_{\text{task}} + \beta\mathcal{L}_{\text{edge}}$, where $\mathcal{L}_{\text{task}}$ is the loss for the downstream task (in our work, QA), $\mathcal{L}_{\text{edge}}$ is the entropy regularization term for edge weights, and $\beta \geq 0$ is a loss weight hyperparameter. In the following subsections, we first explain how the contextualized KG $\mathcal{G}'$ is constructed as a hybrid graph, including its node embeddings $\mathbf{V}$, hybrid edge embeddings $\mathbf{E}$, and adjacency matrix $\mathbf{A}^0$ (§3.2). Next, we show how HGN uses edge-weighted message passing to update $\mathbf{V}$, $\mathbf{E}$, and $\mathbf{A}^0$ for L layers (Figure 3), yielding a refined adjacency matrix $\mathbf{A}^L$ of learned edge weights (§3.3). Finally, we describe how $\mathcal{L}_{\text{task}}$ is computed using $\mathbf{s}$ and $\mathbf{g}$, while $\mathcal{L}_{\text{edge}}$ is calculated using $\mathbf{A}^L$ (§3.4).

3.2 Hybrid Graph Construction

Node Embeddings. The first step of retrieving knowledge from $\mathcal{G}$ is concept grounding, which involves identifying text spans in (q, a) that match nodes in $\mathcal{V}$. We define $\mathcal{V}'$ as the set of all concepts mentioned in (q, a), where $\mathcal{V}'_q = \{v_i\}_{i=1}^{n_q}$ and $\mathcal{V}'_a = \{v_i\}_{i=1}^{n_a}$ are the question and answer concepts, respectively. Each node $v_i \in \mathcal{V}'$ is represented by an embedding $\mathbf{v}_i \in \mathbf{V}$, which can be initialized using BERT (Devlin et al., 2019) or TransE (Bordes et al., 2013).

Hybrid Edge Embeddings. In $\mathcal{G}'$, we loosen the definition of an edge to be $e_{(i,j)} = (v_i, v_j) \in \mathcal{E}'$. We build fully-connected edges between question and answer nodes in $\mathcal{G}'$. The set of edges in $\mathcal{G}'$ is thus defined as $\mathcal{E}' = (\mathcal{V}'_q \times \mathcal{V}'_a) \cup (\mathcal{V}'_a \times \mathcal{V}'_q)$. After concept grounding, we need an edge embedding $\mathbf{e}_{(i,j)} \in \mathbf{E}$ for each edge $e_{(i,j)}$. Let $\mathbf{R}$ be the relation embeddings for all relations in $\mathcal{R}$, obtained using TransE. Each extracted edge $(v_i, r, v_j) \in \mathcal{E}$ is thus initialized in $\mathcal{G}'$ as $\mathbf{e}_{(i,j)} = \mathbf{r} \in \mathbf{R}$. However, due to edge sparsity, many edges do not have labeled relations and cannot be initialized this way.

Meanwhile, despite PLMs' limitations in commonsense, they have shown some ability to encode commonsense knowledge (Davison et al., 2019; Petroni et al., 2019) and aid KG completion (Malaviya et al., 2019; Bosselut et al., 2019; Wang et al., 2020). Hence, we generate edge embeddings for all unlabeled edges by feeding each unlabeled edge into a GPT-2 (Radford et al., 2019) based generator $f_{\text{gen}}(\cdot, \cdot)$. This is further explained in the "Edge Embedding Generation" paragraph.

In summary, edge embeddings are computed in a hybrid way: **(1)** If there exists $r \in \mathcal{R}$ such that $(v_i, r, v_j) \in \mathcal{E}$, then $\mathbf{e}_{(i,j)} = \mathbf{r} \in \mathbf{R}$. **(2)** Otherwise, $\mathbf{e}_{(i,j)} = f_{\text{adapt}}(f_{\text{gen}}(v_i, v_j))$, where $f_{\text{adapt}}(\cdot)$ is an MLP used to transform $f_{\text{gen}}(v_i, v_j)$ into the same space as $\mathbf{r}$.

Edge Embedding Generation. Inspired by recent work in PLM-based commonsense KG completion (Bosselut et al., 2019; Malaviya et al., 2019; Wang et al., 2020), we frame edge generation as text generation. First, for each extracted edge $(h, r, t) \in \mathcal{E}$, we first tokenize its node pair (h, t) and relation label r. Let $\tilde{h}$, $\tilde{r}$, and $\tilde{t}$ be the respective token sequences of h, r, and t. Also, let \$ be the special separator token. Next, for each tokenized extracted edge, we train a GPT-2 model (Radford et al., 2019) to autoregressively generate the concatenated sequence $[\tilde{h}, \$, \tilde{t}, \$, \tilde{h}, \tilde{r}, \tilde{t}]$.

During inference, we only have unlabeled edges $(v_i, v_j) \in \mathcal{E}'$, with no r. Thus, for each (v_i, v_j), GPT-2 is given $s_{\text{input}} = [\tilde{v_i}, \$, \tilde{v_j}, \$]$ and asked to generate the missing tokens $s_{\text{pred}} = [\tilde{v_i}, \tilde{r}, \tilde{v_j}]$. Let $[x_1, x_2, ..., x_T] = [s_{\text{input}}, s_{\text{pred}}]$. The edge embedding for (v_i, v_j) is then computed as $f_{\text{gen}}(v_i, v_j) = \frac{1}{T}\sum_{i=1}^{T} \mathbf{h}_i$, where $\mathbf{h}_i$ is the GPT-2 hidden state for x_i. See Appendix §A for more details.

Alternatively, we consider another edge generation approach proposed by Wang et al. (2020). Here, $f_{\text{gen}}(\cdot, \cdot)$ is trained to generate a relational path connecting v_i to v_j, then pool the path into an edge embedding. The rationale for this approach is that such paths have been shown to contain useful semantic information about the relation between v_i and v_j (Neelakantan et al., 2015; Das et al., 2017; Wang et al., 2020).

Adjacency Matrix. Before edge generation, $\mathcal{G}'$ has binary adjacency matrix $\mathbf{A}^{\text{extract}}$, where $\mathbf{A}_{(i,j)} = 1 \Leftrightarrow \exists r, \text{s.t. } (v_i, r, v_j) \in \mathcal{E}$. After getting embeddings for all edges $(v_i, v_j) \in \mathcal{E}'$, $\mathbf{A}^{\text{extract}}$ becomes $\mathbf{A}^0$, a denser binary adjacency matrix in which $\mathbf{A}^0_{(i,j)} = 1 \Leftrightarrow (v_i, v_j) \in \mathcal{E}'$.

3.3 Hybrid Graph Reasoning

The procedure described in §3.2 yields a hybrid graph, containing unweighted edges between all question-answer node pairs. Constructing this hybrid graph may improve edge recall, but does not address precision. Some edges in the initial hybrid graph may be irrelevant to the question-answer pair, either due to noisy edge extraction or generation. HGN is thus designed to downweight irrelevant edges by converting the unweighted graph into a weighted one, then learning to reweight all hybrid edges during reasoning (Figure 3).

Learnable Adjacency Matrix. Although $\mathbf{A}^0$ is a binary adjacency matrix, HGN populates it with learned edge attention weights and iteratively updates them over L layers of reasoning. We denote the adjacency matrix at layer ℓ as $\mathbf{A}^\ell$, where $0 \leq \mathbf{A}^\ell_{(i,j)} \leq 1$. Updating $\mathbf{A}^\ell$ can be viewed as softly contextualizing the hybrid graph's structure with respect to (q, a).

Edge-Weighted Message Passing. Following the general Graph Network (GN) formulation proposed by Battaglia et al. (2018), HGN's graph reasoning module consists of layer-wise node-to-edge ($v \rightarrow e$) and edge-to-node ($e \rightarrow v$) message passing functions. However, we equip HGN with a modified version of GN's edge-to-node message passing function, in which each edge's weight is used to rescale information flow on that edge. Intuitively, an edge's weight signifies the edge's relevance for reasoning about the given task instance. We also use text encoding $\mathbf{s}$ as global context throughout message passing.

Formally, HGN's update rule at layer ℓ is:

$$
\begin{aligned}
v \rightarrow e : \mathbf{h}^\ell_{(i,j)} &= f^\ell_{v \rightarrow e}\left(\left[\mathbf{h}^{\ell-1}_i, \mathbf{h}^{\ell-1}_j, \mathbf{h}^{\ell-1}_{(i,j)}, \mathbf{s}\right]\right); \\
w^\ell_{(i,j)} &= f^\ell_w\left(\left[\mathbf{h}^{\ell-1}_{(i,j)}, \mathbf{s}\right]\right); \\
\mathbf{A}^\ell_{(i,j)} &= \frac{e^{w^\ell_{(i,j)}}}{\sum_{(s,t)\in\mathcal{E}'} e^{w^\ell_{(s,t)}}}, \\
e \rightarrow v : \mathbf{u}^\ell_{(i,j)} &= f^\ell_u\left(\left[\mathbf{h}^{\ell-1}_i, \mathbf{h}^\ell_{(i,j)}\right]\right); \\
\mathbf{h}^\ell_j &= f^\ell_{e \rightarrow v}\left(\sum_{i\in\mathcal{N}_j} \mathbf{A}^\ell_{(i,j)} \mathbf{u}^\ell_{(i,j)}\right).
\end{aligned} \tag{1}
$$

N_j is the set of v_j's incoming neighbors; $f^\ell_{v\rightarrow e}$, f^ℓ_w, f^ℓ_u and $f^\ell_{e\rightarrow v}$ are MLPs; $\mathbf{h}^0_{(i,j)} = \mathbf{e}_{(i,j)}$ is the initial embedding for edge (v_i, v_j); and $\mathbf{h}^0_i = \mathbf{v}_i$ is the initial embedding for node v_i.

In node-to-edge message passing, the embedding of each edge $(v_i, v_j) \in \mathcal{E}'$ is updated as $\mathbf{h}^\ell_{(i,j)}$, a function of (v_i, v_j)'s constituent nodes and the given context $\mathbf{s}$. Through $\mathbf{s}$, the hybrid graph is strongly contextualized with respect to (q, a). Then, $\mathbf{h}^\ell_{(i,j)}$ is used to compute edge score $w^\ell_{(i,j)}$, which measures the edge's relevance to $\mathbf{s}$. Each edge score is globally normalized across all edges in the graph to produce edge attention weight $\mathbf{A}^\ell_{(i,j)}$, so that low-scoring edges are softly pruned by receiving close-to-zero weight.

We use global edge attention (i.e., normalizing across $\mathcal{E}'$) instead of local edge attention (i.e., normalizing across N_j) because local edge attention assumes at least one edge in N_j is relevant, which may not be true. For example, given an irrelevant or incorrectly grounded concept, none of its edges will be helpful, and so all nodes in its neighborhood should be excluded from influencing the reasoning process. To demonstrate the advantage of global edge attention, we empirically compare our default HGN architecture to an HGN variant based on Graph Attention Network (GAT) (Velickovic et al., 2018), which uses local edge attention, in our experiments.

In edge-to-node message passing, the embedding of each node $v_j \in \mathcal{V}'$ is updated as $\mathbf{h}_j^\ell$, a function of v_j's neighboring edges. For each edge neighbor, edge weight $\mathbf{A}_{(i,j)}^\ell$ is used to rescale the edge's influence on v_j's embedding update.

3.4 Learning Objective

Task Loss. After L layers of message passing, we obtain node embeddings $\{\mathbf{h}_i^L \mid i : v_i \in \mathcal{V}'\}$ and edge embeddings $\{\mathbf{h}_{(i,j)}^L \mid (i,j) : (v_i, v_j) \in \mathcal{E}'\}$. Node embeddings are aggregated into $\mathbf{v}_{\text{agg}}$ via attentive pooling with $\mathbf{s}$ as the query vector. Edge embeddings are aggregated into $\mathbf{e}_{\text{agg}}$ via edge-weighted sum pooling. The final graph encoding is then given as $\mathbf{g} = [\mathbf{v}_{\text{agg}}, \mathbf{e}_{\text{agg}}]$. The probability of a being the answer to q is calculated as $\hat{\rho}(q,a) \propto \exp(\rho(q,a))$, where $\rho(q,a) = f_{\text{score}}([\mathbf{s}, \mathbf{g}]; \boldsymbol{\theta}_{\text{score}})$. We use cross-entropy loss for the QA classification task, so the loss for each (q,a) with label y is:

$$\mathcal{L}_{\text{task}}\left(\hat{\rho}(q,a;\boldsymbol{\theta})), y\right) = -y \log \hat{\rho}(q,a;\boldsymbol{\theta}). \tag{2}$$

Entropy Regularization. To encourage the model to be decisive during edge reweighting, we use a regularization term to penalize non-discriminative edge weights. In an extreme case, a blind model will assign the same weight to all edges, degenerating $\mathcal{G}'$ into an unweighted graph. This is a failure mode, since $\mathcal{G}'$ is likely to contain mostly irrelevant edges, and we want the model to focus on the helpful edges. Therefore, via $\mathcal{L}_{\text{edge}}$, we train the model to minimize the entropy of the edge weight distribution (i.e., make the distribution more skewed), in order to maximize the informativeness of the predicted edge weights. Lower entropy means the model has higher certainty about edges' relevance to the given task instance, such that the model will discriminatively judge some edges as being much more relevant than others. $\mathcal{L}_{\text{edge}}$ is computed as:

$$\mathcal{L}_{\text{edge}}(\mathbf{A}^L(q,a)) = -\sum_{(i,j):(v_i,v_j)\in\mathcal{E}'} \mathbf{A}_{(i,j)}^L \log \mathbf{A}_{(i,j)}^L. \tag{3}$$

Joint Learning. We jointly optimize $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{edge}}$, so graph reasoning and structure can be jointly learned. The full learning objective is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(q,a,y)\sim X_{\text{train}}} \left[\mathcal{L}_{\text{task}}\left(\hat{\rho}(q,a)), y\right) + \beta \cdot \mathcal{L}_{\text{edge}}(\mathbf{A}^L(q,a))\right], \tag{4}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{text}}, \boldsymbol{\theta}_{\text{graph}}, \boldsymbol{\theta}_{\text{score}}\}$ is the set of all learnable parameters, and X_{train} is the training set. We train our model end-to-end by minimizing $\mathcal{L}(\boldsymbol{\theta})$ with the RAdam (Liu et al., 2020) optimizer.

4 Experiments

4.1 Experimental Setup

We evaluate our proposed model on four multiple-choice commonsense QA datasets: **CommonsenseQA** (Talmor et al., 2019), **CODAH** (Chen et al., 2019), **OpenBookQA** (Mihaylov et al., 2018) and **QASC** (Khot et al., 2020) (details in Appendix §B). We use ConceptNet (Speer et al., 2017), a commonsensense knowledge graph, as $\mathcal{G}$. For text encoder f_{text}, we experiment with BERT-Base, BERT-Large (Devlin et al., 2019) and RoBERTa(-Large) (Liu et al., 2019) to validate our model's effectiveness over different text encoders. For OpenbookQA and QASC, retrieving related facts from the provided corpus plays an important role in boosting the model's performance. Therefore, we build our graph reasoning model on top of retrieval-augmented methods on the leaderboard: "AristoRoBERTa"[2] for OpenBookQA and "RoBERTa (2-step IR)"[3] for QASC. In this way, we can study if strong retrieval-augmented methods can still benefit from KG knowledge and our HGN framework.

4.2 Compared Methods

We compare our model with a series of KG-augmented methods and different graph encoders:

Models Using Extracted Facts. We consider seven models that only use extracted facts. **RN** (Santoro et al., 2017) builds the graph with the same node set as our method but extracted edges only. The graph vector is calculated as $\mathbf{g} = \text{Pool}(\{\text{MLP}([\mathbf{v}_i, \mathbf{e}_{(i,j)}, \mathbf{v}_j]) \mid (v_i, v_j) \in \mathcal{E}'\})$. **GN** (Battaglia et al., 2018) presents a general formulation of GNNs. We instantiate it with the layerwise propagation rule defined in Equation 1. It differs from our HGN in that: (1) it only considers extracted edges; (2) all edge weights are fixed to 1. **MHGRN** (Feng et al., 2020) generalizes GNNs with multi-hop message passing. **GAT** (Velickovic et al., 2018) adopts attention mechanism to

---

[2] `https://leaderboard.allenai.org/open_book_qa/submission/blcp1tu91i4gm0vf484g`

[3] `https://leaderboard.allenai.org/qasc/submission/bolaun0ghifmkohgvhr0`

Methods	**BERT-Base**		**BERT-Large**		**RoBERTa**	
	60% Train	100% Train	60% Train	100% Train	60% Train	100% Train
LM Finetuning*	52.06 (±0.72)	53.47 (±0.87)	52.30 (±0.16)	55.39 (±0.40)	65.56 (±0.76)	68.69 (±0.56)
RN* (Santoro et al., 2017)	54.43 (±0.10)	56.20 (±0.45)	54.23 (±0.28)	58.46 (±0.71)	66.16 (±0.28)	70.08 (±0.21)
RN + Link Prediction*	-	-	53.96 (±0.56)	56.02 (±0.55)	66.29(±0.29)	69.33 (±0.98)
RGCN* (Schlichtkrull et al., 2018b)	52.20 (±0.31)	54.50 (±0.56)	54.71 (±0.37)	57.13 (±0.36)	68.33 (±0.85)	68.41 (±0.66)
GAT (Velickovic et al., 2018)	53.05 (±0.37)	56.51 (±0.74)	55.80 (±0.53)	58.18 (±1.07)	69.63 (±0.42)	71.20 (±0.72)
GN (Battaglia et al., 2018)	53.67 (±0.45)	55.65 (±0.51)	54.78 (±0.61)	57.81 (±0.67)	68.78 (±0.67)	71.12 (±0.45)
GconAttn* (Wang et al., 2019a)	51.36 (±0.98)	54.41 (±0.50)	54.96 (±0.69)	56.94 (±0.77)	68.09 (±0.63)	69.88 (±0.47)
KagNet* (Lin et al., 2019)	-	56.19	-	57.16	-	-
MHGRN* (Feng et al., 2020)	54.12 (±0.49)	56.23 (±0.82)	56.76 (±0.21)	59.85 (±0.56)	68.84 (±1.06)	71.11 (±0.81)
PathGenerator* (Wang et al., 2020)	54.44 (±0.42)	56.99 (±0.41)	57.53 (±0.19)	59.07 (±0.30)	69.46 (±0.23)	72.68 (±0.42)
HGN (w/ PathGen edges)	55.68 (±0.29)	57.77 (±0.39)	58.19 (±0.27)	60.89 (±0.19)	70.95 (±0.21)	73.41 (±0.31)
HGN (w/ RelGen edges)	**55.72** (±0.32)	**58.01** (±0.29)	**58.19** (±0.11)	**61.11** (±0.21)	**71.10** (±0.11)	**73.64** (±0.30)

Table 1: **Accuracy on CommonsenseQA inhouse test set.** Both our model variants significantly improve over all baselines. We use the same inhouse split as Lin et al. (2019). For baselines with *, we use the reported numbers from Feng et al. (2020) and Wang et al. (2020) if available. Mean and standard deviation of four seeds are presented for all models except KagNet.

reweight edges locally in each node's neighborhood. We implement it by replacing the graph edge attention with local edge attention and only considering $\mathcal{L}_{\text{task}}$ during training. **RGCN** (Schlichtkrull et al., 2018a) extends Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) with relation-specific transition matrices during message passing. It operates on the same graph as RN. The graph vector is calculated as $\mathbf{g} = \text{Pool}(\{\mathbf{h}_i^L \mid v_i \in V\})$. **GconAttn** (Wang et al., 2019b) softly aligns the nodes in question and answer and do pooling over all matching nodes to get $\mathbf{g}$. **KagNet** (Lin et al., 2019) uses an LSTM to encode relational paths between question and answer concepts and pool over the path embeddings for graph encoding.

Models Using Extracted and Generated Facts. We consider two models that use both extracted facts and generated facts. **RN + Link Prediction** differs from RN by only considering the generated relation (predicted using TransE (Bordes et al., 2013)) between question and answer concepts. **PathGenerator** (Wang et al., 2020) learns a path generator from paths collected through random walks on the KG. The learned generator is used to generate paths connecting question and answer concepts. $\mathbf{g}$ is calculated as the concatenation of the pooled vector over the generated paths and the pooled vector over the extracted paths.

Our Model's Variants. As described in §3.2, the edge embedding can be computed either as a relation embedding or a path embedding. We name these two variants as **HGN (w/ RelGen edges)** and **HGN (w/ PathGen edges)** respectively.

Methods	**Single**	**Ensemble**
ALBERT+DESC-KCR (Xu et al., 2020)	80.7	83.3
ALBERT+KD	80.3	80.9
ALBERT+KCR	79.5	-
Unified QA (Khashabi et al., 2020)	79.1	-
ALBERT+KRD	78.4	-
T5-3B (Raffel et al., 2020)	78.1	-
ALBERT+HGN (w/ RelGen edges)	**77.3**	**80.0**
TeGBERT	76.8	-
ALBERT+PathGenerator (Wang et al., 2020)	75.6	78.2
ALBERT (Lan et al., 2020)	-	76.5

Table 2: **Leaderboard of CommonsenseQA.** HGN ranks first among comparable systems, especially achieving remarkable improvement over PathGenerator (Wang et al., 2020).

4.3 Results

Performance Comparisons. Tables 1, 3, 4 show performance comparisons between our models and baseline models on CommonsenseQA, CODAH, OpenBookQA and QASC. We clearly find that models with stronger text encoders perform better (i.e. RoBERTa > BERT-Large > BERT-Base). For all text encoders, our HGN shows consistent improvement over baseline models on all datasets. The improvement over all baselines are tested to be statistically significant under most settings, demonstrating the effectiveness of HGN both with and without retrieved evidence.

We also submit our best model to leaderboards for CommonsenseQA and OpenBookQA. For CommonsenseQA (Table 2), our HGN ranks first among comparable approaches and shows remarkable improvement over PathGenerator (Wang et al., 2020) and the LM Finetuning approach (ALBERT (Lan et al., 2020)). Higher-ranking

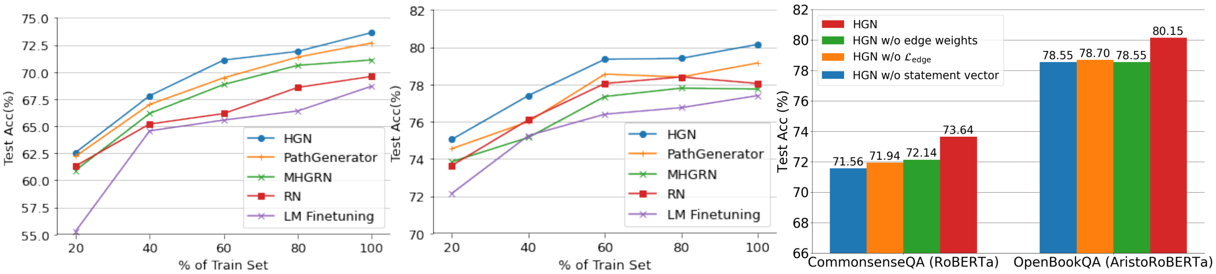

(a) CommonsenseQA (RoBERTa). (b) OpenBookQA (AristoRoBERTa). (c) Ablations on model variants.

Figure 4: **Low-resource and ablation studies.** (a)(b) Performance of HGN and baseline models with different amounts of training data; (c) Performance of different model variants.

Methods	**BERT-Large**	**RoBERTa**
LM Finetuning	65.74	83.14
RN (Santoro et al., 2017)	64.59	82.45
RGCN (Schlichtkrull et al., 2018b)	65.56	82.42
GAT (Velickovic et al., 2018)	65.88	82.78
GN (Battaglia et al., 2018)	65.52	82.06
GconAttn (Wang et al., 2019a)	65.17	82.35
MHGRN (Feng et al., 2020)	65.92	83.07
PathGenerator (Wang et al., 2020)	64.67	82.27
HGN (w/ PathGen edges)	66.21	**84.32**
HGN (w/ RelGen edges)	**66.75**	84.08

Table 3: **Test accuracy on CODAH.** Both our model variants consistently improve over all baselines. We use the official split for 5-fold cross validation. Mean accuracy on 5 folds are presented.

models either use stronger text encoders or leverage additional data resources. Specifically, UnifiedQA (Khashabi et al., 2020) and T5-3B (Raffel et al., 2020) are based on T5. They have 11B and 3B parameters respectively, making them impractical to be finetuned in an academic setting. ALBERT+DESC-KCR (Xu et al., 2020) and ALBERT+KD additionally use concept definitions from dictionaries. ALBERT+DESC-KCR and ALBERT+KCR leverage "question_concept" annotations, which are used during the construction of the CommmonsenseQA dataset and allow the model to learn shortcuts that don't generalize to other datasets. ALBERT+KRD retrieve sentences from OMCS corpus (Liu and Singh, 2004) as input. These methods are therefore not comparable with our model. For OpenBookQA (Table 5), our model ranks first among all models using AristoRoBERTa as the text encoder.

Datasets	**OpenBookQA**	**QASC**
Base Models	**AristoRoBERTa**	**RoBERTa (2-step IR)**
LM Finetuning*	77.40 (±1.64)	73.34 (±0.71)
RN* (Santoro et al., 2017)	78.05 (±0.77)	72.77 (±1.50)
RN + Link Prediction*	77.25 (±1.11)	-
RGCN* (Schlichtkrull et al., 2018b)	74.60 (±2.53)	72.23 (±1.36)
GAT (Velickovic et al., 2018)	78.20 (±1.22)	72.61 (±0.93)
GN (Battaglia et al., 2018)	77.25 (±0.91)	72.53 (±0.70)
GconAttn* (Wang et al., 2019a)	71.80 (±1.21)	72.72 (±1.66)
MHGRN (Feng et al., 2020)	77.75 (±0.38)	73.24 (±0.45)
PathGenerator* (Wang et al., 2020)	79.15 (±0.78)	72.96 (±0.68)
HGN (w/ PathGen edges)	80.05 (±0.54)	74.10 (±0.42)
HGN (w/ RelGen edges)	**80.15** (±0.38)	**74.27** (±0.31)

Table 4: **Test accuracy on OpenBookQA and QASC with retrieval-augmented methods as base models.** Both our model variants greatly improve over all baselines except HGN (w/ PathGen edges) over MHGRN. For OpenbookQA baselines with *, we use reported numbers from Wang et al. (2020). Mean and standard deviation of four seeds are presented.

Training with Less Labeled Data. Figure 4 (a)(b) show the results of our model and baselines when trained with different portions of the training data on CommonsenseQA and OpenBookQA. Our model gets better test accuracy under all settings. On CommonsenseQA without retrieved evidence, the improvement over the knowledge-agnostic baseline (LM Finetuning) is generally more significant with less training data, which suggests that incorporating external knowledge is helpful in the low-resource setting.

Study on More Model Variants. To better understand the model design, we experiment with three variants of HGN (w/ RelGen edges) on CommonsenseQA and OpenBookQA. **HGN w/o statement vector** doesn't consider s in Equation 1, which isolates the graph encoder from the text encoder. **HGN w/o $\mathcal{L}_{\text{edge}}$** does not consider the entropy regularization term and thus does not penalize non-discriminative edge weights. **HGN w/o edge weights** reasons over an unweighted graph with hybrid features, which means edge weights are all fixed to 1 during training. Figure 4 (c) shows the results of the ablation study. "HGN" outperforms "HGN w/o $\mathcal{L}_{\text{edge}}$", suggesting the usefulness

Methods	**Text Encoder**	**Test Acc**
UnifiedQA (Khashabi et al., 2020)	T5-11B	87.2
T5-11B + KB	T5-11B	85.4
T5-3B (Raffel et al., 2020)	T5-3B	83.2
PathGenerator (Wang et al., 2020)	ALBERT	81.8
HGN (w/ RelGen edges)	**AristoRoBERTa**	**81.4**
AristoRoBERTa + KB	AristoRoBERTa	81.0
MHGRN (Feng et al., 2020)	AristoRoBERTa	80.6
PathGenerator (Wang et al., 2020)	AristoRoBERTa	80.2
KF + SIR (Banerjee and Baral, 2020)	RoBERTa	80.2
AristoRoBERTa	AristoRoBERTa	80.2

Table 5: **Leaderboard of OpenBookQA.** Our HGN ranks first among all submissions using AristoRoBERTa as the text encoder.

Contextualized Graph	**GN** ($\mathbf{A}^{\text{extract}}$)	**HGN** ($\mathbf{A}^L$)
Number of Edges	3.65 ($\pm$2.73)	4.38 ($\pm$3.24)
Number of Valid Edges	2.67 ($\pm$1.95)	3.15 ($\pm$1.98)
Percentage of Valid Edges	71.64%	78.51%
Average Helpfulness Score of Edges	0.90 ($\pm$0.50)	1.16 ($\pm$0.51)
Prune Rate	-	22.84%

Table 6: **User studies on learned graph structures.** 30 pairs of contextualized graphs output by GN and HGN are evaluated by 5 annotators.

of our proposed entropy regularization. Comparing "HGN w/o statement vector" with "HGN", we find that accessing context information is also important for graph reasoning, which means information propagation and edge weight prediction should be conducted in a context-aware manner. HGN also improves over "HGN (w/o edge weights)", indicating the effectiveness of conducting context-dependent pruning.

4.4 User Study on Learned Structures

To assess HGN's ability to refine graph structure, we compare the graph structure before and after being processed by HGN. Specifically, we sample 30 questions with its answer from CommonsenseQA's development set and ask 5 human annotators to evaluate the graph output by GN (with adjacency matrix $\mathbf{A}^{\text{extract}}$ and extracted facts only) and by HGN (with adjacency matrix $\mathbf{A}^L$). We manually binarize $\mathbf{A}^L$ by removing edges with weight lower than 0.01.

Given a graph, for each edge (fact), annotators are asked to rate its **validness** and **helpfulness**. The validness score is rated as a binary value in a context-agnostic way: 0 (the fact does not make sense), 1 (the fact is generally true). The helpfulness score measures if the fact is helpful for solving the question and is rated on a 0 to 2 scale: 0 (the fact is unrelated to the question and answer), 1 (the fact is related but doesn't directly lead to the answer), 2 (the fact directly leads to the answer). Note that the percentage of valid edges can be understood as the precision of graph edges. For a given instance, the number of valid edges is proportional to the recall of the edges. We also include another metric named "prune rate" calculated as: $1 - \frac{\text{\# edges in binarized } \mathbf{A}^L}{\text{\# edges in } \mathbf{A}^0}$, which measures the portion of edges assigned very low weights (softly pruned) during training and is only applicable to HGN.

The mean ratings for 30 pairs of (GN, HGN) graphs by 5 annotators are reported in Table 6. The Fleiss' Kappa (Fleiss, 1971) is 0.51 (moderate agreement) for validness and 0.36 (fair agreement) for helpfulness. The graph refined by HGN has both more edges and denser valid edges compared to the extracted one. The refined graph also achieves a higher average helpfulness score. These all indicate that our HGN learns a superior graph structure with more helpful edges and fewer noisy edges, which improves over previous works that rely on extracted and static graphs. Detailed cases can be found in Appendix §C.

5 Related Work

Commonsense QA. Commonsense QA is challenging because the required commonsense knowledge is seldom given in the question-answer context or encoded in the PLM's parameters. Thus, many works obtain this knowledge from external sources (e.g., KGs, corpora). While Lv et al. (2020) show that KGs and corpora can provide complementary knowledge, our paper focuses on improving the use of KG knowledge. KG knowledge can be acquired in different ways, either from KG-extracted edges (Lin et al., 2019; Ma et al., 2019; Feng et al., 2020; Yasunaga et al., 2021), PLM-generated edges (Bosselut and Choi, 2019), or both (Wang et al., 2020). KG-augmented models mainly differ in how they encode KG knowledge, using message passing (Schlichtkrull et al., 2018a; Feng et al., 2020) or edge/path aggregation (Lin et al., 2019; Bosselut and Choi, 2019; Ma et al., 2019; Wang et al., 2020). The most relevant work to ours is Wang et al. (2020). The main difference is that they coarsely combine extracted and generated knowledge via late fusion, while HGN encodes both types of knowledge within a unified graph. Besides, they use RN to pool over a set of paths for graph encoding, while HGN reasons over the graph via message passing and edge reweighting.

Graph Structure Learning. Instead of assuming a fixed graph structure, a number of graph models learn the graph structure with respect to the downstream task. Some models learn to discretely select edges for the graph (i.e., hard pruning). Kipf et al. (2018) and Franceschi et al. (2019) sample the graph structure from a predicted probabilistic distribution with differentiable approximations. Norcliffe-Brown et al. (2018) calculate the relatedness between any pair of nodes and only keep the top-k strongest connections for each node to construct the edge set. Sun et al. (2019) start with a small graph and iteratively expand it with retrieving operations. Others learn to reweight edges in a fully connected graph (i.e., soft pruning). Jiang et al. (2019) and Yu et al. (2019) propose heuristics for regularizing edge weights. Hu et al. (2019) use the question embedding to help predict edge weights. Unlike other edge reweighting models, HGN operates over a hybrid graph of both extracted and generated edges, while updating edge weights with respect to node, edge, and text features.

6 Conclusion

In this paper, we propose HGN, a KG-augmented model for CSR. To address KG edge sparsity and noisy edge extraction/generation, HGN learns to jointly contextualize extracted and generated knowledge by reasoning over both within a unified graph structure. We justify HGN's design by showing that HGN improves performance on various CSR benchmarks and user studies. In future work, we plan to increase the graph's relation expressiveness by incorporating open relations, plus make the edge extraction/generation process more dependent on the reasoning context.

Acknowledgments

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007, the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research, the Defense Advanced Research Projects Agency with award W911NF-19-20271, and NSF SMA 18-29268. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. We would like to thank all the collaborators in USC INK research lab for their constructive feedback on the work. We would also like to thank the anonymous reviewers for their valuable comments.

References

Ian Apperly. 2010. *Mindreaders: the cognitive basis of" theory of mind"*. Psychology Press.

Pratyay Banerjee and Chitta Baral. 2020. Knowledge fusion and semantic knowledge ranking for open domain question answering. *arXiv preprint arXiv:2004.03101*.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Antoine Bosselut and Yejin Choi. 2019. Dynamic knowledge graph construction for zero-shot commonsense question answering. *arXiv preprint arXiv:1911.03876*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. CODAH: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 63–69, Minneapolis, USA. Association for Computational Linguistics.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141, Valencia, Spain. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1972–1982. PMLR.

David Gunning. 2018. Machine common sense concept paper. *arXiv preprint arXiv:1810.07528*.

Ronghang Hu, Anna Rohrbach, Trevor Darrell, and Kate Saenko. 2019. Language-conditioned graph networks for relational reasoning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 10293–10302. IEEE.

Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11313–11320. Computer Vision Foundation / IEEE.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8082–8090. AAAI Press.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. 2018. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2693–2702. PMLR.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8449–8456. AAAI Press.

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 22–32, Hong Kong, China. Association for Computational Linguistics.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2019. Exploiting structural and semantic context for commonsense knowledge base completion. *arXiv preprint arXiv:1910.02915*.

Gary Marcus. 2018. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China. Association for Computational Linguistics.

Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. 2018. Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8344–8353.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4967–4976.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018a. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018b. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, and Michael Witbrock. 2019a. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of AAAI*, pages 7208–7215. AAAI Press.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019b. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of AAAI*, volume 33, pages 7208–7215.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2020. Fusing context into knowledge graph for commonsense reasoning. *arXiv preprint arXiv:2012.04808*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. 2019. Graph-revised convolutional network. In *Proceedings of ECML-PKDD*.

A Implementation Details of Edge Embedding Generator (RelGen)

Here, we give a more detailed explanation of the PLM-based edge embedding generator f_{gen}, introduced in the "Edge Embedding Generation" paragraph of §3.2.

To implement f_{gen}, we adopt GPT-2 (Radford et al., 2019), which is pretrained on large corpora and achieves great success on a wide range of tasks involving sentence generation, as a generator to generalize the facts from the knowledge graph. We first convert each fact $(h, r, t) \in \mathcal{E}$ into a word sequence with a "prompt-generation" format: $\left[\tilde{h}, \$, \tilde{t}, \$, \tilde{h}, \tilde{r}, \tilde{t}\right]$, where $\tilde{h}, \tilde{r}, \tilde{t}$ are the word sequence of h, r, t respectively, $\$$ denotes the delimiter token used by GPT-2, and $[\cdot, \cdot]$ denotes word sequence concatenation. We adopt this format because $\left[\tilde{h}, \tilde{r}, \tilde{t}\right]$ is similar to a natural language fact. Generating facts in a natural format helps induce commonsense knowledge stored in GPT-2 (Bosselut et al., 2019). We denote the synthetic sentence as $s_{(h,r,t)} = \left[x_1^{(h,r,t)}, \ldots, x_{n_{(h,r,t)}}^{(h,r,t)}\right]$ and finetune GPT-2 on all synthetic sentences created from $\mathcal{E}$ with the language modeling objective:

$$\mathcal{L}_{\text{gen}}(\mathcal{E}) = \sum_{(h,r,t) \in \mathcal{E}} \sum_{i=1}^{n_{(h,r,t)}} \log P\left(x_i^{(h,r,t)} \mid x_1^{(h,r,t)}, \ldots, x_{i-1}^{(h,r,t)}\right).$$

After that, given any two concepts (v_i, v_j), we build a prompt as $[\tilde{v_i}, \$, \tilde{v_j}, \$]$ and let the model to generate the following word sequence. We denote the whole sentence (both prompt and generation) as $s_{(v_i,v_j)}$, and the hidden states of each word during generation as $\mathbf{h}_1, \ldots, \mathbf{h}_T$ where T is the sentence length. We average hidden states of all words in the sentence to get the relational feature: $f_{\text{gen}}(v_i, v_j) = \frac{1}{T} \sum_{i=1}^{T} \mathbf{h}_i$.

B Details of Datasets

Below are descriptions of the four datasets used for the experiments presented in §4.

CommonsenseQA (Talmor et al., 2019) is a multiple-choice QA dataset targeting commonsense. It's constructed based on the knowledge in ConceptNet. Since the test set of the official split (9741/1221/1140 for OFtrain/OFdev/OFtest) is not publicly available, we compare our models with baseline models on the inhouse split (8500/1221/1241 for IHtrain/IHdev/IHtest)[4] used by previous works (Lin et al., 2019; Feng et al., 2020; Wang et al., 2020).

CODAH (Chen et al., 2019) contains 2801 sentence completion questions testing commonsense reasoning skills. We perform 5-fold cross validation using the official split.

OpenBookQA (Mihaylov et al., 2018) is a multiple-choice QA dataset modeled after openbook exams. Besides 5957 elementary-level science questions (4957/500/500 for train/dev/test), it also provides an open book with 1326 core science facts. Solving the dataset requires combining facts from open book with commonsense knowledge.

QASC (Khot et al., 2020) is a QA dataset with questions about grade-school science. It has 9980 8-way multiple-choice questions (8134/926/920 train/dev/test), and comes with a corpus of 17M sentences. Since the official test set does not have labels, we create an in-house test split by moving a randomly sampled set of 920 questions from the training set to the test set. Solving questions in QASC requires retrieving facts from the corpus and composing them to produce an answer.

C Case Study

In addition to the experiments in §4, we present a case study here, which compares a HGN-generated graph with a KG-extracted graph used by GN. On the development set of CommonsenseQA, there are two dominating cases and we show the representative instance of each one. Figure 5 (a) shows the first case, where HGN prunes edges from the extracted graph. Our HGN assigns the highest weights to the most helpful facts (`book`, `AtLocation`, `house`), (`telephone book`, `AtLocation`, `house`). It also downweight unhelpful fact (`place`, `IsA`, `house`) and invalid fact (`usually`, `RelatedTo`, `house`). Figure 5 (b) shows the second case, where new generated facts are incorporated into reasoning. All generated facts that are kept by the model make sense in the context and help identify the answer. Both cases suggest that our model improve the quality of the contextualized knowledge graph compared to the current methods that only rely on extracted facts.

---

[4] `https://github.com/INK-USC/MHGRN/blob/master/data/csqa/inhouse_split_qids.txt`

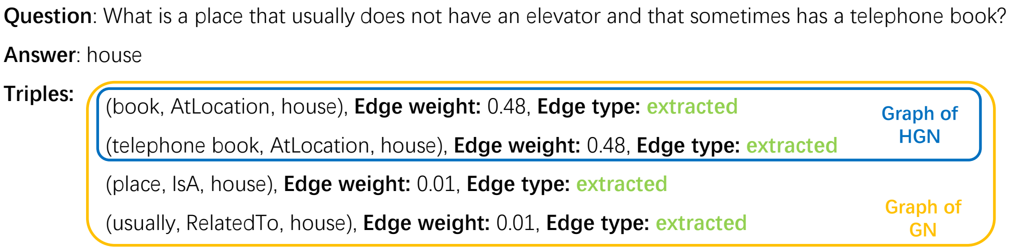

Question: What is a place that usually does not have an elevator and that sometimes has a telephone book?

Answer: house

Triples:

(book, AtLocation, house), **Edge weight:** 0.48, **Edge type:** extracted

(telephone book, AtLocation, house), **Edge weight:** 0.48, **Edge type:** extracted

(place, IsA, house), **Edge weight:** 0.01, **Edge type:** extracted

(usually, RelatedTo, house), **Edge weight:** 0.01, **Edge type:** extracted

(a) **Case I: Unrelated extracted facts are filtered out.**

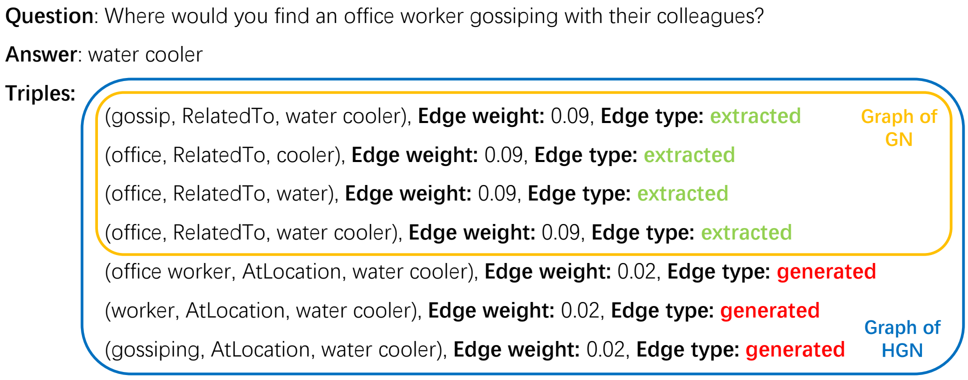

Question: Where would you find an office worker gossiping with their colleagues?

Answer: water cooler

Triples:

(gossip, RelatedTo, water cooler), **Edge weight:** 0.09, **Edge type:** extracted

(office, RelatedTo, cooler), **Edge weight:** 0.09, **Edge type:** extracted

(office, RelatedTo, water), **Edge weight:** 0.09, **Edge type:** extracted

(office, RelatedTo, water cooler), **Edge weight:** 0.09, **Edge type:** extracted

(office worker, AtLocation, water cooler), **Edge weight:** 0.02, **Edge type:** generated

(worker, AtLocation, water cooler), **Edge weight:** 0.02, **Edge type:** generated

(gossiping, AtLocation, water cooler), **Edge weight:** 0.02, **Edge type:** generated

(b) **Case II: Helpful generated facts are incorporated.**

Figure 5: **Representative cases from the development set of CommonsenseQA.**

Analyzing Stereotypes in Generative Text Inference Tasks

Anna Sotnikova[*]
University of Maryland
asotniko@umd.edu

Yang Trista Cao[*]
University of Maryland
ycao95@umd.edu

Hal Daumé III
University of Maryland
Microsoft Research
me@hal3.name

Rachel Rudinger
University of Maryland
rudinger@umd.edu

Abstract

Stereotypes are inferences drawn about people based on their demographic attributes, which may result in harms to users when a system is deployed. In generative language-inference tasks, given a premise, a model produces plausible hypotheses that follow either logically (natural language inference) or commonsensically (commonsense inference). Such tasks are therefore a fruitful setting in which to explore the degree to which NLP systems encode stereotypes. In our work, we study how stereotypes manifest when the potential targets of stereotypes are situated in real-life, neutral contexts. We collect human judgments on the presence of stereotypes in generated inferences, and compare how perceptions of stereotypes vary due to annotator positionality.

1 Introduction

Social categories refer to collections of people with shared traits; stereotypes—cognitive structures that associate categories (e.g., man, Black, poor, professor) with both roles (e.g., doctor) and traits (e.g., absent-minded)—are central to how people construe social meaning (Levon, 2014; Macrae and Bodenhausen, 2001; Greenwald et al., 1998). Social psychology has studied how stereotypes, as a cognitive process, are entwined with the production of human affects of prejudice and in-group favoritism, as well as behaviors like discrimination (Stangor, 2014; Jackson, 2011). Linguistic anthropology and sociolinguistic studies argue that language—as the predominant way of naming categories and transmitting knowledge—is the only (or at least the primary) mechanism by which social stereotypes are shared as part of cultural knowledge (Fishman, 1956; Stangor and Schaller, 2012; Maass and Arcuri, 1996).

premise	A Cuban person needs help.	
↳ **hypothesis**	Then PERSONX gets a job.	
Question	**Annotator 1**	**Annotator 2**
correct?	yes	yes
plausible?	yes	yes
identity?	maybe yes	no
situation?	maybe no	not sure
sentiment?	maybe negative	positive
stereotype?	yes	no
description?	problems with jobs	n/a

Table 1: Annotation example; the hypothesis is automatically generated from the premise. Both annotators found the hypothesis grammatically correct and plausible. One annotator viewed this hypothesis as negative stereotypical towards Cuban people, assuming that they have problems with jobs. The other annotator had the opposite opinion. Annotators differ in their backgrounds and social groups they belong to.

In this paper, we study ways in which categories implicate inferences around stereotypical roles and traits computationally.[1] Approaching stereotyping through the lens of *inference* allows us to focus on what models learn as *implications* rather than simply associations (e.g., that lexical semantics models typically find antonyms like “hot” and “cold” to be highly related). Specifically, we train models for English textual inference—including both logical- (NLI) and commonsense-inference (CI)—and investigate how stereotypes are reproduced by these models. The models we train *generate* hypothesis text given a fixed premise text (e.g., “PERSONX lights up candles”, where PERSONX is substituted with the target category label), and by varying the target category label, we are able to investigate what and how much stereotypical information the model produces in its generated hypotheses (see Table 1).

To perform this analysis, we collect human judgments on the generated hypotheses, given explic-

---

[*] Equal contribution

[1] It *can* go the other way: if asked to visualize a forgetful professor, your mental image may conform to stereotypes.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4052–4065
August 1–6, 2021.

Domain	Target Categories
Gender	man, woman, non-binary person, trans man, trans woman, cis man, cis woman
Race	African American, African-American, Black, White, White-American, White American, Hispanic, Latino, Latina, Latin American, Arab, American Indian, Native American, Alaska Native, Asian American, Native Hawaiian, Pacific Islander
Nationality	Mexican, Chinese, Russian, Indian, Irish, Cuban, Italian, Japanese, German, French, British, Jamaican, American, Filipino
Religion	Jewish, Muslim, Catholic, Christian, Buddhist, Mormon, Amish, Protestant, Atheist, Hindu
Politics	Democrat, Republican, Communist, Socialist, Fascist, Libertarian, Liberal, Capitalist, Conservative
Socio	Rich, Wealthy, Poor, Immigrant, Refugee, Homeless, Aristocrat, Lower class, Middle class, Working class, Upper class, Formerly incarcerated, First generation, Bourgeoisie

Table 2: Stereotype domains and corresponding target categories.

itly stated target categories in an otherwise neutral premise, such as that in Table 1. We focus on 71 target categories drawn from six stereotype domains that are particularly salient in the United States[2], listed in Table 2. With the collected human judgments, we first investigate which models and categories lead to stereotyped inferences, and the degree to which the invoked stereotypes are negative. It is well established that stereotypes are both an individual phenomenon—something that resides in the heads of individual people—as well as a cultural phenomenon—that "[sterotypes] exist also in 'the fabric of society' itself" (Stangor and Schaller, 2012), and as such *who* the annotators are matters (Hovy and Spruit, 2016; Jørgensen et al., 2015; Hazen et al., 2020). In view of this, part of our analysis specifically considers how individual annotators' perceptions of stereotypes may vary.

Overall, we find that socioeconomic status and politics are the domains most likely to yield stereotyped inferences. This is notable, as most existing work in this space has focused on the domains of gender and race (see § 2). We also discover that within these domains, certain target categories are more likely to yield negatively stereotyped inferences; specifically, the categories of *poor*, *working class*, and *formerly incarcerated* people. For human judgements, we observe that annotators disagree the most on the questions about whether an inference is based on the identity mentioned in the premise, as well as whether it reflects a stereotype or not. This appears especially true when the hypotheses include less well-known stereotypes, or stereotypes toward groups that are not typically stereotyped in US culture.

Significant limitations. The most significant limitation is our focus on English and US culture, as discussed above; this means that while we may recognize negative stereotypes of (for instance) *Latin Americans* in the US, we will likely miss negative stereotyping of *Roma* in Spain. Our work is also limited to just six stereotype domains, and we do not explicitly account for intersectionality. While our annotators are of diverse cultural backgrounds, another limitation is that there are only four, limiting the breadth of our analysis of annotator positionality.

2 Related Work

Our work builds on a growing body of recent computational literature on stereotypes (often termed "bias"). A major focus of past work has been on the domains of gender and race, across a variety of tasks including language modeling, coreference resolution, natural language inference, machine translation, and sentiment analysis (Sheng et al., 2019; Rudinger et al., 2018; Lu et al., 2018; Dinan et al., 2019; Rudinger et al., 2017; Kiritchenko and Mohammad, 2018); Blodgett et al. (2020) provide a review. There has simultaneously been a range of work aimed to mitigate problems of stereotyping in NLP systems, including many in the space of text generation (Sheng et al., 2020; He et al., 2019; Clark et al., 2019; Huang et al., 2020). In comparison to this line of work, our main extensions are (a) a broader range of domains considered, and (b) a specific focus on the generation of entailed text.

Several very recent papers have also explored other stereotype domains, including disabilities (Hutchinson et al., 2020), and larger collections of domains similar to ours. For instance, two recently released datasets by Nadeem et al. (2020) and Nangia et al. (2020) provide example texts and measurements to determine if a language generation system exhibits stereotyping toward the domains of nationality, race, religion, profession, orientation, disability, age, appearance, socioeconomic status, and gender. Li et al. (2020) probes transformer-

---

[2] Although we focus on the US, many of these categories are salient globally, especially gender, sex and class (Fiske, 2017). Other domains may also be globally relevant due the US's export of stereotypes through media (Crane, 2014).

based question answering models on stereotypes towards gender, nationality, religion, ethnicity domains. Here, question/answer pairs are constructed where a particular answer either does or does not contain a known stereotype. Our analysis is similar to these, with a slightly broader set of domains, a focus on inference rather than question answering, and a post-hoc analysis of what a model actually produces, rather than a predefined dataset of potentially expected stereotypes. An advantage of the dataset approach is re-usability, while an advantage of the post-hoc analysis approach is that it may capture stereotypes we had not thought of a priori.

3 Data Generation & Annotation

We conduct experiments to study stereotypes with a focus on generative text inference tasks. To do that, we construct a list of stereotype domains and a list of target categories for each of the domains. We also manually create a list of underspecified, real-life context situations for instantiated premises. Using these constructed premises, we conditionally generate hypotheses from three models. The resulting premise-hypothesis pairs are then judged for stereotypes by four humans annotators.

3.1 Background on Text Inference Tasks

We consider two text inference tasks: natural language inference (NLI; also *textual entailment*) and commonsense inference (CI); both are typically framed as classification tasks (Dagan and Glickman, 2004; Bowman et al., 2015; Williams et al., 2018). Namely, given a text *premise* p and a text *hypothesis* h, determine the relationship r between the two. For NLI, the typical set of relationships are $r = \text{ENTAILED}$ if p logically entails h, CONTRADICTED if h contradicts p, and NEUTRAL otherwise.

While CI tasks are less standardized than NLI, here we follow the *if-then* formulation used in ATOMIC (Sap et al., 2018) and COMET (Bosselut et al., 2019). There, a premise is a short sentence describing a scenario involving a generic participant ("PersonX"). Associated with each premise is a multiplicity of hypotheses, capturing likely or plausible inferences belonging to one of several predefined relation types, e.g., X-INTENT (inferences about PersonX's intent) or X-EFFECT (inferences about the scenario's effect on PersonX). See appendix Table A1 for the full list of relations.

Following Bosselut et al. (2019), we consider text inference from a generative perspective: given a premise p and relation type r, generate a hypothesis h that bears that relation to p. This framing enables us to explore what trained models have learned about inference, without providing explicit hypothesis prompts. For NLI, we focus on two finetuned GPT-2 models using the SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) datasets. For CI, we use the COMET model (Bosselut et al., 2019), which is trained on the ATOMIC (Sap et al., 2018) dataset.[3] More details are in Appendix A.

3.2 Experimental Setup

Our goal is to construct hypotheses like "The [TARGETCATEGORY] person is cutting up fish for dinner." To this end, we define a set of domains and target categories, and a set of context situations.

Stereotype Domains. Certain social categories are more likely to be referenced in stereotyped inferences. As discussed in § 2, previous works have mostly focused on two domains: gender (typically *men* vs. *women*) and race (typically *Black* vs. *White*). To broaden the space of consideration, we mostly follow Nangia et al.'s (2020) taxonomy of stereotype domains, which is a narrowed version of US Equal Employment Opportunities Commission's list of protected categories; to this set, we also add the *political stance* domain. Overall, the six stereotype domains we choose to focus on are: race/color/ethnicity/ancestry (henceforth, *race*, *gender*, *religion*, *nationality*), socioeconomic status (henceforth, *socio*), and political stance (henceforth, *politics*).

Target Categories. Within each stereotype domain, our goal is to select target categories that are (a) common and (b) most likely to be the target of stereotypes in the United States; we rely on authoritative sources to assembled these lists. For *religion*, *nationality*, *race*, *socio*, and *politics*, we mostly follow the lists from outside resources (such as Pew, the World Atlas, and Wikipedia; see Appendix C); for *gender*, we manually create the list. Note that many categories have multiple possible labels; we attempt to use ones that are currently generally benign and affirming, to avoid triggering stereotypical inferences based on an explicitly negative represen-

---

[3] We note that even when CI is not framed as a generative task, CI datasets have been *created* using generative textual inference models (Zhang et al., 2017; Zellers et al., 2018).

tation of the target category[4]. For instance, we use *formerly incarcerated person* instead of *felon* and *Black* or *African American* instead of older and/or related derogatory terms.[5] This choice, however, means that our results do not capture the full extent of stereotypes, as more derogatory terms often come with stronger stereotypical inferences, even for the same category (Devine and Baker, 1991). Table 2 is the list of our 71 target categories, which also includes spelling variations for some categories (e.g., presence or absence of a hyphen). In our analysis, we merge multiple terms under one category into a single label (e.g., *Latino*, *Latina*, and *Latin American* are analyzed as *Latin*). The table of substitutions is provided in the supplement.

Context Situations. For our experiments, we manually construct a list of 102 real-life contexts into which the target categories will be inserted. Our aim here is to create premises that describe situations that are unlikely themselves to lead to the production of stereotypes in hypotheses, independently of the target category. When we construct the list we follow two strategies, thus creating two types of contexts. The first type, DAILY ACTIVITIES are neutral contexts in which it is difficult to imagine a direct extension to a stereotype. These are situations that do not beg for any particular continuation, such as "PERSONX holds a cup" or "PERSONX goes for a walk". The second type, TRIGGERS, are situations that are neutral, but which may trigger some stereotypical output for different target categories. For instance, with the premise: "PERSONX is holding a knife", plausible hypotheses include "PERSONX is cooking", "PERSONX is being attacked", and "PERSONX is attacking somebody." See the supplement for the full list.[6] One limitation of using these context templates is that the resulting premise can be somewhat unnatural. However, in order to control triggers other than the tested target category, we opted for simple, if sometimes stilted, over more complex.

3.3 Data Generation

We are interested in model behavior for different target categories in our constructed contexts. We generate around $130k$ example premises for later human annotation from the cross-product of:

(a) target category (71 total),
(b) context situation (103 total),
(c) model (3: SNLI, MNLI, COMET), and
(d) entailment relation (1 for NLI and 3 for CI[7]),

and generating three hypotheses for each.

To get an initial sense of the outputs generated, we ran the Hugging Face sentiment analysis system on the generated hypotheses[8]; details are in Appendix B. For each example, it gives a label "negative" or "positive" with an associated probability. For instance, "African Americans participate in demonstrations" gets a negative label, with probability near chance: 54%. Overall, the hypotheses generated from three models have around half negative hypotheses and half positive. In general, CI (COMET) tends to produce on average slightly more negative inferences (56%) than NLI (51% for both MNLI and SNLI).

3.4 Human Annotation

The annotation is done by the four authors of this paper.[9] For annotation, we sample a subset of examples from the generated dataset. In total, we annotate 1281 examples, which are generated with 21 situations (out of 102) and 61 categories.[10] Among the 1281 examples, 255 examples (which consists of 3 situations with all categories and 4 categories with all situations) are annotated by all four annotators in order to measure annotator agreement. The remaining 1026 examples are randomly divided to be annotated by only one of the annotators.[11]

Annotation guidelines. To examine how model behaviors and human judgments vary for different

---

[4] Some target categories might be ambiguous. For instance, *Indian* can mean Native American, *conservative* can mean just a conservative person and not a political affiliation, and *fascist* is mostly a historical category (now is used more as an insult).

[5] In some cases this is difficult. For instance, different people who may generally be referred to as Indigenous, Native American, or American Indian may identify with one of those terms, or may rather prefer to be referred to by the name of their tribe (Blackhorse, 2017).

[6] We adjusted the target terms accordingly for the premise to sound more natural (e.g. we use "A poor person" and "An atheist person" rather than "A poor" and "An atheist").

[7] For NLI, we only consider the *neutral* relation, because with other relations we observe structurally limited hypotheses (e.g., CONTRADICTION tends to produce hypotheses which say the opposite of the premise). For CI, we pick three relations: X-ATTR, X-WANT, and X-EFFECT; see §3.1

[8] https://huggingface.co/transformers/examples.html

[9] Annotators consist of one Russian cis woman, one Chinese cis woman, one white American cis man, and one white American cis woman. All annotators hold advanced degrees.

[10] Out of 71 target category terms, we merge terms that are spelling variations of one category, which results in 61 target categories in total. When generating examples for each category, we use these terms rotationally.

[11] Annotated data available at: https://github.com/AnnaSou/stereotypes_generative_inferences.

Label	Annotation Question
VALID?	Does the hypothesis include more information than the premise? Is it grammatically correct? (If not, skip the remaining questions)
PLAUS?	Is the hypothesis plausible given the premise? (If not, skip the remaining questions).
IDEN?	The hypothesis appears to be based on the target category identity.
SITU?	The hypothesis appears to be based on the context situation.
SENTI?	What types of inferences does the hypothesis have about the target category?
STEREO?	Does the hypothesis contain known (to you) stereotypes of the target category?
WHAT?	What is the evoked association of the target category based on the hypothesis?

Table 3: Annotation questions.

target categories in our neutral contexts, we have annotators answer the questions in Table 3.

▶ **VALID?**: annotators decide whether the hypothesis is grammatically correct and includes more information than the premise. Possible answers are *yes* or *no*. For instance, given "A Catholic person saw dust on the floor.", an *invalid* hypothesis is "The dust on the floor PERSONX."

▶ **PLAUS?**: annotators decide whether the hypothesis is plausible given the premise. The possible answers are *yes* or *no*. A plausible hypothesis means the hypothesis is logically consistent with the premise. For instance, given "A Catholic person saw dust on the floor.", an *implausible* hypothesis is "The dust on the floor was the same as tan."

▶ **IDEN?**: annotators choose whether the hypothesis appears to be based on the target category's identity to one of the following degrees: *yes*, *maybe yes*, *not sure*, *maybe no*, *no*. For example, given the same premise "A Buddhist person is sleeping.", the hypothesis "The person is poor." is likely based on the target category identity.

▶ **SITU?**: annotators choose whether the hypothesis appears to be based on the situation: *yes*, *maybe yes*, *not sure*, *maybe no*, *no*. For instance, given the premise "A Buddhist person is sleeping.", the hypothesis "The person is having a happy dream." is based on the situation. It is possible for an inference to be based on both situation and identity, as in the same premise with hypothesis "The poor person is having a happy dream."

▶ **SENTI?**: annotators identify if the hypothesis expresses ppositive or negative sentiment toward the target person on a five item Likert scale: from *negative*, *maybe negative*, *neutral* to *maybe positive*, *positive*. For example, "The American person is rude and entitled" is a negative inference, whereas "The woman makes sure her family is happy" is a maybe positive inference.

▶ **STEREO?**: annotators choose whether the hypothesis conforms to stereotypes they know; options are *yes*, *no*, or *maybe*. People's perceptions on whether a hypothesis is stereotypical or problematical are highly subjective (Hazen et al., 2020), and one research question we seek to answer is how annotators' levels of agreement may vary for different target categories (see §4.2). Overall, stereotypes can harmful even when positive; the *nurturing* stereotype of women is used to justify exclusion from professional settings (Tinsley et al., 2009), and, for women who do not conform to the stereotype, can lead to increased sexual harassment (Leskinen et al., 2015).

▶ **WHAT?**: annotators write the possible associations evoked from the hypothesis as free text.[12]

4 Findings & Analysis

We analyze our results from two perspectives: model behavior and human judgment. We first discuss the models' behaviors for different stereotype domains, target categories, and situations. We then explore how human judgments deviate depending on target categories and domains, and provide annotator agreement analysis. We are also interested throughout in evidence of *defaulting* (Rosch, 1975)—the observation that some target categories often go unremarked (e.g., "woman" may often be used to mean "cis woman", and "American" may often be used to mean "white American")—in the generations, for instance because the models are unused to seeing language with explicit defaults.

In analyzing our results, we start from the normative position that identical model behavior across target categories is *insufficient*, despite being a prevalent goal in past literature (Blodgett et al., 2020, *inter alia*). We take this position for two reasons. First, because if a person of some category sees an offensive stereotype about themselves in a downstream system, they are harmed even if the same output is generated for other categories. Second, because social hierarchies enable members of some groups to more easily subjugate members of other groups, the same oppressive stereotypes are

---

[12] We left open the possibility that some generated text may contains *anti*-stereotypes, and had annotators mark these. However, they were exceedingly rare and had very low agreement, and so we exclude this question from the remainder of our analysis.

more likely to harm people in categories lower on the social hierarchy than those higher.

For example, for the premise "PERSONX has a child", the generated hypothesis from MNLI is "PERSONX is not allowed to have a child" for *African American*, *Asian American*, and *Amish*. This evokes historically forced sterilization of African American women (Prather et al., 2018), the recently canceled Chinese one-child policy (Xie et al., 2018), and stereotypes of Amish families having many children[13]. These stereotypes are harmful for each of these groups, even though it is also generated for others. The degree of harm also varies by category; for instance, if the same hypothesis were generated for *white American*, it is unclear that would cause much harm. More examples from COMET are in appendix Table A2 and supplement for SNLI and MNLI.

4.1 Model Behavior

With the collected human annotations, we seek to answer the following research questions:

1. Which models and domains are more prone to invalid and implausible hypotheses?
2. What target categories have more hypotheses based on identity?
3. Which models and domains are more likely to lead to stereotyped hypotheses? Which target categories are more prone to negative inferences?
4. What are the commonly evoked associations?

We address each question in turn, expanding on the question, motivating it, and presenting the results.

1. Which models and domains are more prone to invalid and implausible hypotheses? We aim to reveal model's capability of generating plausible hypotheses. It is harmful if models fail to do so for some particular target categories, because then any downstream system will not be able to rely on such inference model. Additionally, we use this question as a filtering step.

For each of the stereotype domains (and models), we wish to know what percentages of generated hypotheses are *illegitimate*. By illegitimate, we mean hypotheses that are grammatically incorrect, do not contain any additional information to the premise, or are implausible. We compare the results across models and find that the MNLI model is more prone to generate illegitimate hypotheses than SNLI and COMET models (21.9% versus 7.4% and 8.1% for SNLI and COMET respectively.)

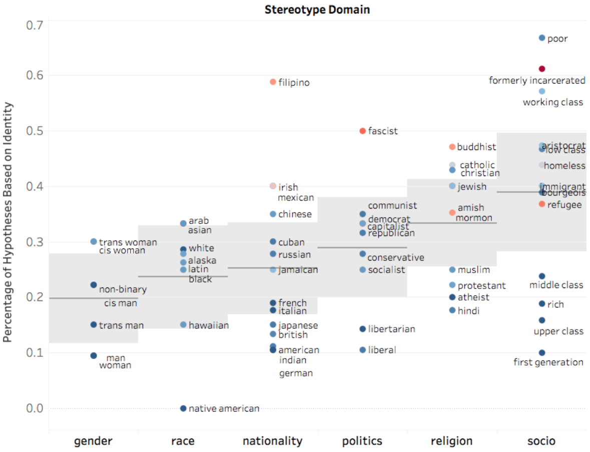

Figure 1: Annotation results for the question of what portion of models' generations are based on identities across target categories. The y-axis is the fraction of hypotheses which are based on identities. For each stereotype domain on the x-axis, the grey line and the shaded box represent the average percentage in that domain and its 95% confidence interval. Inferences based solely on target categories' identities are color-coded. The redder the more inferences are based solely on identity. The darkest blue corresponds to zero percentage of inferences based exclusively on identity.

We then compare the percentage across stereotype domains to see for which domain the inference systems are more likely to fail in generating legitimate hypotheses. The results are shown in appendix Figure A3. We find that the inference models generate more illegitimate hypotheses for target categories of *religion*, *socioeconomic status*, *race*, and *nationality* ($\sim 13\%$) than for target categories of *gender* and *politics* ($\sim 7\%$). We also find that the percentage of illegitimate hypotheses is extremely high for *British*, *Asian*, *low class*, *poor*, and *atheist*. We can also see some effect of *defaults*: *cis woman* (4.8%) generates more implausible outputs than *woman* (0%), possibly because in the training data for these models, "cis woman" is rarely seen.

2. What target categories have more hypotheses based on identity? When target categories are embedded in real-life, neutral situations, we prefer a model that generates outputs more keyed to the situation than to the identity of the person mentioned. If a model frequently makes inferences based on the identities and ignores the situations for some target categories, this can lead to harms related to those categories.

To perform this analysis, we first filter out invalid

---

[13] https://amishamerica.com/how-many-children-do-amish-have/

and implausible hypotheses (**VALID?**, **PLAUS?**). Then among the remaining 1144 annotations, we check how many hypotheses are based on identity by looking into **IDEN?**. For this analysis, annotations of *yes* and *maybe yes* are counted as based on identity. Figure 1 shows for each target category the percentage of hypotheses (post-filtering) that are based on identity.

We find that across models, around 29% of generated hypotheses are based on identities, and that the target categories of *socioeconomic status* and *religion* focus more on identities, in comparison to *politics*, *nationality*, *race* and *gender* (39% and 33% vs. 29%, 25%, 23%, and 19% respectively). In general, we find that, on average, more vulnerable target categories have a higher percentage of hypotheses generated based on identities. (This is not universal: the target category of *aristocratic* has generations with the same level of dependency on identity as the *low class* category, despite the asymmetry in social position here.)

We are particularly interested in cases where a hypothesis is based *only* on identity and not at all on situation: this means that the model has essentially focused exclusively on a person's identity and ignored everything else. Therefore, we explore **SITU?** and check how many hypotheses are not based on situation for each target category and stereotype domain. Annotations of *no* or *maybe no* for **SITU?** are counted as not based on situation. In the results, we see that hypotheses generated about *formerly incarcerated*, *poor*, *working class*, and *Filipino* turn out to be highly dependent on identities. However, among these categories, *formerly incarcerated* and *Filipino* have 38.9% and 23.5% of hypotheses exclusively based on identities (and not situation), while *poor* and *working class* categories only have 6.7% and 14.3% of such inferences. (These percentages are color-coded in Figure 1: higher percentages in red, lower in blue.) Overall, the highest percentage of inferences based exclusively on identities is for *religion* domain 14.2% and the lowest is for *gender* domain 4.4%. Similar to our observation on **IDEN?**, we find vulnerable target categories tend to have more hypotheses that completely ignore the situation. Categories like *formerly incarcerated*, *Asian*, *Filipino*, *refugee*, *Amish*, and *fascist* have a high percentage of hypotheses generated independent of situation. On the other hand, categories such as *white*, *woman*, *man*, *trans man*, *French*, and *American* have no hypotheses in which the situation is ignored.

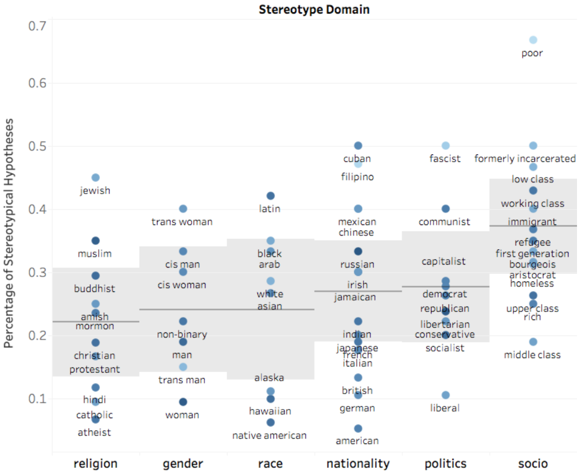

Figure 2: Annotation results for the question which target categories and stereotype domains are more likely to have stereotyped hypotheses. The y-axis represents the fraction of stereotyped hypotheses for each target category. The color of the circle encodes the overall percentage of negative inferences for each target category (The darker the color the more negative inferences such target category has. The lightest color corresponds to 33.3% of negative inferences while the darkest color corresponds to 77.8%. Note: not all negative inferences are stereotyped inferences and vise versa.). For each stereotype domain, the grey line and the shaded box represent the average percentage and its 95% confidence interval.

3. Which models and domains are more likely to lead to stereotyped hypotheses? Which target categories are more prone to negative inferences? Although the previous question reflects how much the models' generations depend on identity information, we still want to see directly how frequently explicitly stereotypical hypotheses are generated across different models and stereotype domains. If some model consistently generates hypotheses with stereotypes of some target categories, then it can cause representational harms to people in those target categories.

To answer this question, we delve into annotations for **STEREO?**. For **STEREO?**, votes for *yes* and *maybe* are categorized as containing stereotypes, while *no* is categorized as do *not* contain stereotypes. For **SENTI?**, we count *positive* and *maybe positive* as positive inferences, and *negative* and *maybe negative* as negative, and *neutral* as neither positive nor negative. We find the percentages of stereotyped hypotheses and negative hypotheses

Category	Association
Immigrant	poor, illegal, criminals, farmers, desperate
Trans man	avoided, sinful, sick, sex work
Muslim	religious, aggressive
Jewish	religious, wealthy, unpleasant
Mormon	immoral, selling drugs, sinful
capitalist	greedy, rich, mean
Asian	gangs, smart, not respected, Chinese
poor	sad, needy, drugs, avoided, weak
Cuban	alcoholics, tacos, friendly, criminals
Russian	violent, alcoholics, rude, intellectual
American	pro-war, proud, selfless

Table 4: The keywords from evoked associations for some target categories.

are similar across all three models: around 28% contain known stereotypes and 59% are with negative sentiment. Detailed results across stereotype domain comparison are shown in Figure 2. Overall, these models generate more stereotyped hypotheses for domains of *socioeconomic status*, *politics*, and *nationality*, compared to domains of *race*, *gender*, and *religion*. The most stereotyped categories from each domains are *trans woman*, *Cuban*, *Latin American*, *Fascist*, *Jewish*, and *poor*. In terms of percentage of negative inferences, *socioeconomic status* has the least negative inferences of 54% and *religion* has the highest of 63%.

Moreover, we find that the target categories that are more affected by stereotypes are not necessarily prone to have negative inferences. For instance, *poor* has 67% or stereotyped inferences, while only 33% of those are negative. On the other hand, *woman* have less than 10% of stereotyped inferences, but 76% are negative. Overall, all models produce negative inferences even for categories with a low level of stereotyping: models achieve some parity in distributing negative generations across domains, but, as discussed in the conclusion, this does not necessarily make the models fair.

4. What are the evoked associations? In Table 4, we provide keywords that are associated by annotators with the target categories. The full list is in supplementary materials. Some of these associations relate to the existing stereotypes, some do not. For instance, *democrat* based on the generated hypotheses are associated with "rude", "causing trouble", and "making deals." Even though there might be no related stereotypes, such hypotheses still might be harmful to the target category.

4.2 Human Perceptions of Stereotypes

We explore human perceptions of stereotypes. It is known that people's perceptions on whether a hypothesis is stereotypical or not can be subjective (McGarty et al., 2002). Overall, we find that annotators highly agree **VALID?** on **PLAUS?** with 91.8% and 85.8% agreements respectively, and highly disagree on **IDEN?**, **SENTI?**, and **STEREO?** with 39.2%, 37%, and 21.8% scores respectively.

To calculate annotator agreement, we use the 255 examples that were annotated by all four annotators. Throughout this section, we calculate *agreement* as the fraction of times all annotators give the same answer.[14] We filter out examples that have fewer than three annotations. This may happen because, for example, some annotators mark the example as invalid or implausible and thus skip the rest of the questions. Then for examples that have four annotations, we randomly pick three of them to calculate agreement.

Agreement on Hypotheses Origins. Annotators agree more on the situation question (66.5% agreement) than the identity question (39.2%), likely because the situation question is defined purely on the bases of the stated hypothesis, while the identity question depends on annotators' perceptions of that identity.

We observe zero agreement on whether situation is based on identity or not for several target categories such as *white*, *Asian*, *Mormon*, *liberal*. On the other hand, categories of *Jewish*, *communist*, and *atheist* have complete agreement (100%). In general, we see that annotators have more disagreements on the questions that involve target categories' identities, most likely because these rely more on cultural context.

Agreement on Stereotyped Hypotheses. Overall, for **STEREO?** annotators agree on only 21.8% of the examples. We observe that annotators have complete agreement on categories that are either highly stereotyped such as *homeless*, *trans man*, *communist* or have very little widely known stereotypes such as *atheist* and *Native American*. In addition, both categories of *atheist* and *Native American* have a very low level (around 6%) of stereotyped hypotheses. We suspect that it is simply eas-

---

[14] We choose to report the percentage of agreements rather than an inter-annotator agreement statistic (e.g., Fleiss's kappa or Krippendorff's alpha) because it is more easily interpretable than coefficients and we *expect* annotations to be skewed to some choices for questions like **VALID?** and **PLAUS?**.

ier for annotators to detect stereotypes for typically stereotyped categories.

There are also some exceptions like *cis woman*, which has a high percentage of stereotyped hypotheses (33.3%) but has low annotator agreement (0%). We suspect the reason is that the stereotypes towards *cis woman* in our dataset are not well-known existing stereotypes, which tends to lead to more disagreements. As an example of annotator disagreement for **STEREO?**, given the premise "A Latin American person has a child," annotators disagreed about whether the hypothesis "The person then gets pregnant" represents a stereotype or not; those who annotated it as stereotype did so because it evokes a fertility threat stereotype (Gutiérrez, 2009), a stereotype not known by all annotators. Overall, we find that annotators' perception and ability to detect stereotypes varies based on their knowledge of the target categories, arguing that a large—and diverse—set of annotators is important for problems around stereotyping.

Because of the subjective nature of these annotations, we further consider agreement at two levels: (1) how often do all annotators agree, and (2) how often do a randomly chosen pair of annotators agree. High percentages for (1) indicate that a question is not particularly subjective (or that all annotators have the same subjective opinion), while a small value of (1) but large value of (2) indicates that a strong degree of subjectivity exists, but that even among four annotators some of them frequently agree. For (1), agreement on the more objective questions such as hypotheses correctness, plausibility, and relatedness to situations have 91.0%, 82.9%, and 66.7% agreement. On the other hand, we observe zero agreement for stereotypes, 24.9% for identity agreement, and 26.6% for sentiment agreement. This suggests—especially for the 0% for stereotypes—that getting more annotators is needed in order to feel confident about coverage. For (2), we observe overall a high level of agreement for correctness, plausibility, and relatedness to situations with 95.3%, 88.0%, and 82.5% agreement respectively. We additionally observe a reasonable level of agreement for sentiment and stereotypes: 57.1% and 61.2% respectively. Agreement regarding whether a hypothesis is based on identity is the lowest at 50.1%. This suggests that while annotators *can* agree on these questions, there is sufficient subjectivity that all four rarely do.

5 Conclusion & Discussion

We investigated stereotypes in generative inference models from two perspectives: model behavior and human perceptions. We find that the most stereotyped domains by our NLI and CI models are *religion* and *socioeconomic status*, rather than *gender* and *race*, which are the focus of many previous studies. On the other hand, the stereotype domains and target categories we studied is not exhaustive either; even in a US context, most obviously we are missing domains related to disability, beauty/body type, sexuality, age, pregnancy, and so on.

Moreover, since we investigated inference tasks, instead of focusing on models generating "fair" hypotheses over target categories, we are much more concerned with how each hypothesis is perceived by a human reader. We observe some cases in which the models generate similar outputs across several target categories, but for which the generated text is highly stereotyped and thus may cause representational harms.

Finally, from human judgments, though our work is limited to US culture and the backgrounds of our four annotators, we find that people's different backgrounds influence their perceptions of stereotypes. Even though this might result in lower agreement scores, such diversity can be actually useful (Pavlick and Kwiatkowski, 2019) in helping to explore the problem space. Overall, when deploying a system, it is important to make a wise consideration on annotators' backgrounds. Considering annotators of different age, professions, education, and culture might give a multiplicity of valuable perspective on stereotypes.

Acknowledgments

The authors are grateful to all the reviewers who have provided helpful suggestions to improve this work. We also thank the CLIP lab at the University of Maryland for comments on previous drafts.

References

Amanda Blackhorse. 2017. Native American? American Indian? Nope. Indian Country Today.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *CoRR*, abs/1909.03683.

Diana Crane. 2014. Cultural globalization and the dominance of the american film industry: cultural policies, national film industries, and transnational film. *International Journal of Cultural Policy*, 20(4):365–382.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. *Learning Methods for Text Understanding and Mining*, 2004:26–29.

Patricia G. Devine and Sara M. Baker. 1991. Measurement of racial stereotype subtyping. *Personality and Social Psychology Bulletin*, 17(1):44–50.

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2019. Queens are powerful too: Mitigating gender bias in dialogue generation. *CoRR*, abs/1911.03842.

Joshua A. Fishman. 1956. An examination of the process and function of social stereotyping. *The Journal of Social Psychology*, 43(1):27–64.

Susan T. Fiske. 2017. Prejudices in cultural contexts: Shared stereotypes (gender, age) versus variable stereotypes (race, ethnicity, religion). *Perspectives on psychological science: a journal of the Association for Psychological Science*, 12(5):791–799.

A. G. Greenwald, D. E. McGhee, and J. L. Schwartz. 1998. Measuring individual differences in implicit cognition: the implicit association test.

Elena R. Gutiérrez. 2009. *Fertile matters: The politics of Mexican-origin women's reproduction*. University of Texas Press.

Timothy J. Hazen, Alexandra Olteanu, Gabriella Kazai, Fernando Diaz, and Michael Golebiewski. 2020. On the social and technical challenges of web search autosuggestion moderation. *arXiv:2007.05039 [cs]*. ArXiv: 2007.05039.

He He, Sheng Zha, and Haohan Wang. 2019. Unlearn dataset bias in natural language inference by fitting the residual. *CoRR*, abs/1908.10763.

Dirk Hovy and Shannon L. Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics.

William Huang, Haokun Liu, and Samuel R. Bowman. 2020. Counterfactually-augmented SNLI training data does not yield better generalization than unaugmented data. In *Proceedings of the First Workshop on Insights from Negative Results in NLP, Insights 2020, Online, November 19, 2020*, pages 82–87. Association for Computational Linguistics.

Ben Hutchinson, Vinodkumar Prabhakaran, Emily Denton, Kellie Webster, Yu Zhong, and Stephen Denuyl. 2020. Social biases in NLP models as barriers for persons with disabilities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5491–5501, Online. Association for Computational Linguistics.

Lynne M. Jackson. 2011. The psychology of prejudice: From attitudes to social action. American Psychological Association.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of studying and processing dialects in social media. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 9–18, Beijing, China. Association for Computational Linguistics.

Svetlana Kiritchenko and Saif M. Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. *CoRR*, abs/1805.04508.

Emily A Leskinen, Verónica Caridad Rabelo, and Lilia M Cortina. 2015. Gender stereotyping and harassment: A "catch-22" for women in the workplace. *Psychology, Public Policy, and Law*, 21(2):192.

Erez Levon. 2014. Categories, stereotypes, and the linguistic perception of sexuality. *Language in Society*, 43(5):539–566.

Tao Li, Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Vivek Srikumar. 2020. UNQOVERing stereotyping biases via underspecified questions. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3475–3489, Online. Association for Computational Linguistics.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2018. Gender bias in neural natural language processing. *CoRR*, abs/1807.11714.

Anne Maass and Luciano Arcuri. 1996. Language and stereotyping. In *C. N. Macrae, C. Stangor, & M. Hewstone (Eds.), Stereotypes and stereo typing*, pages 193–226. New York : Guilford Press.

C. Neil Macrae and Galen V. Bodenhausen. 2001. Social cognition: Categorical person perception. *The British journal of psychology. General section*, 92(1):239–255.

Craig McGarty, Vincent Y. Yzerbyt, and Russell Spears. 2002. Stereotypes as explanations. Cambridge University Press.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2020. Stereoset: Measuring stereotypical bias in pretrained language models. *CoRR*, abs/2004.09456.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Trans. Assoc. Comput. Linguistics*, 7:677–694.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *CoRR*, abs/1505.04870.

Cynthia Prather, Taleria R. Fuller, William L. Jeffries, IV, Khiya J. Marshall, A. Vyann Howell, Angela Belyue-Umole, and Winifred King. 2018. Racism, african american women, and their sexual and reproductive health: A review of historical and contemporary evidence and implications for health equity. pages 249–259. Health Equity.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Eleanor Rosch. 1975. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3):192.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79, Valencia, Spain. Association for Computational Linguistics.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. *CoRR*, abs/1804.09301.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2018. ATOMIC: an atlas of machine commonsense for if-then reasoning. *CoRR*, abs/1811.00146.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. *CoRR*, abs/1909.01326.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2020. Towards controllable biases in language generation. *arXiv:2005.00268 [cs]*. ArXiv: 2005.00268.

Charles Stangor and Mark Schaller. 2012. Stereotypes as individual and collective representations. *Stereotypes Prejudice*.

Dr. Charles Stangor. 2014. Principles of social psychology – 1st international edition. BCcampus.

Catherine Tinsley, Sandra Cheldelin, Andrea Schneider, and Emily Amanatullah. 2009. Women at the bargaining table: Pitfalls and prospects. *Negotiation Journal*, 25:233 – 248.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Naiming Xie, Ruizhi Wang, and Nanlei Chen. 2018. Measurement of shock effect following change of one-child policy based on grey forecasting approach. *Kybernetes*, 47.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. *CoRR*, abs/1808.05326.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association for Computational Linguistics*, 5:379–395.

A Implementation Details

Text Inference Datasets. For training our generative inference models, we use three datasets: two of them cover natural language inference, and one is for commonsense inference.

The **Stanford Natural Language Inference (SNLI)** corpus was created by Bowman et al. (2015).It contains about 570k examples. Each example has a premise, relation (entails, contradicts, neutral), and related hypotheses. Premises were taken from captions for the Flickr30k corpus (Plummer et al., 2015). Hypotheses are written by crowd workers as independent image captions.

The **MultiGenre Natural Language Inference (MNLI)** corpus by Williams et al. (2018) was built following the SNLI structure. It has 433k examples. MNLI, being much broader than SNLI, covers ten different domains. It has a range of styles, degrees of formalities, and topics.

The **Atlas of Machine Commonsense (Atomic)** corpus was introduced by Sap et al. (2018). The corpus has about 300k events associated with 877k textual descriptions of inferential knowledge. Such knowledge is collected and organized as if-then relations for hypotheses specifically about a person in a premise named *PersonX*. There are 4 groups of relations (see Table A1), each group has several if-then relations. In total, there are 9 if-then relations. For instance, given the *premise* = “PersonX drops a glass”, the *relation* = “Causes for PersonX - because PersonX wanted”, then the *hypothesis* = “to get a glass”.

Type of Relations	**Inference dimension**
If-Event-Then-Mental-State	xIntent, xReact, oReact
If-Event-Then-Event	oEffect, oWant, xNeed, xEffect, xWant
If-Event-Then-Persona	xAttr

Table A1: List of relations for Commonsense Inference model (Sap et al., 2018).

Models. For our experiments, we build three models – two for NLI and one for CI. For the NLI systems, we finetune a GPT-2 language model (Radford et al., 2019) with the MNLI and SNLI datasets separately for 4 epochs with a batch size of 2. This process takes about 3 hours on a single GPU. We adapt Hugging Face transformers Wolf et al. (2020) for both finetuning and generation. For CI, we use the pre-trained Commonsense Transformers on Atomic (COMET)[15] model (Bosselut et al., 2019). COMET constructs commonsense knowledge bases from the transformer language model (Radford et al., 2018) with multi headed attention, which was trained on ATOMIC dataset. COMET can produce inferences not only about familiar examples, but also about unseen examples. The range of COMET outputs were evaluated by crowd workers and judged as correct.

B Sentiment analysis

Hugging Face sentiment analysis system [16] is used on all $130,000$ generated hypotheses for three models. For CI (COMET), averaged all target categories, 56% of inferences has negative sentiment. The domain with the most negative generations is *politics*, with average 67% negative inferences, and the most affected target category is *fascists* with 85% negative. The two next most negative domains are *socioeconomic status* and *religion* with about 61% negative. The most negative categories in these are *protestants* and *formerly incarcerated*, which are 87% and 79% negative, respectively.

For MNLI, the results have less variance: the overall average level of negative sentiment is about 51%. The most negative domain is *politics* (54%) and category *fascists* (80%). The domain with the highest variance is *socioeconomic status*, ranging from 89% negative hypotheses for *poor* to 24% for *wealthy* (with an overall average of 30%). The SNLI model has even more uniform distribution within stereotype domains and all target categories have narrow distribution within 45%-52% with the only most extreme case for *fascists*: 64%.

C Lists for Target Categories

Lists for target categories retrieved on 17-10-2020:

- religion: www.pewforum.org/religious-landscape-study/
- nationality: www.worldatlas.com/articles/largest-ethnic-groups-and-nationalities-in-the-united-states.html
- race: www.statista.com/statistics/270272/percentage-of-us-population-by-ethnicities/
- socioeconomic status: www.nationallatinonetwork.org/immigrant-advocacy-basics/what-is-immigration-status and www.apa.org/pi/ses/resources/class/measuring-status
- political stance: en.wikipedia.org/wiki/Political_spectrum

---

[15] https://github.com/atcbosselut/comet-commonsense

[16] https://huggingface.co/transformers/examples.html

Target Category	Hypothesis
African American	The person then gets called racist.
Black	The person is seen as loving.
Latin American	The person then gets pregnant.
Arab	The person is seen as loving.
Pacific Islander	The person then has to buy food.
Latina	The person is seen as caring.
White American	The person is seen as caring.
Native Hawaiian	The person then gets a lot of respect from people.
Asian American	As a result, others want to go to a restaurant.
Hispanic	The person is seen as untrustworthy.
American Indian	Then child is protected.
Formerly incarcerated	The person then gets arrested.
Filipino	As a result, others want cry.
American	The person is seen as a parent.
British	Then child cries.
Jamaican	The person then gets pregnant.
Mexican	The person then is asked to stop by a counselor.
German	As a result, others want cry.
Italian	The person then is asked questions.
Cuban	As a result, others want to have a good life.
Russian	The person then gets pregnant.
cis woman	The person is seen as loving.
woman	As a result, others want to get a new baby.
man	The person is seen as responsible.
trans man	The person is seen as responsible.
trans woman	The person then is a father.
Buddhist	The person is seen as a parent.
Catholic	The person is seen as pious.
Jewish	As a result, others want cry.
Japanese	The person then gets called a father.
Amish	As a result, others want go to church.
Muslim	Others then are protected by personx.
Mormon	The person is seen as a sinner.
Protestant	Others then cry in sadness.
communist	The person is seen as racist.
capitalist	The person then gets a raise.
conservative	Others then are loved.
republican	The person is seen as racist.
fascist	Then child is abused.
socialist	Others then get a new home.
libertarian	Others then are taken care of.
liberal	The person then gets a raise.
democrat	The person is seen as a racist.
bourgeoisie	The person then gets pregnant.
homeless	The person then gets a new home.
immigrant	Others then they learn about Personx 's culture.
formerly incarcerated	The person is seen as unintelligent.
upper class	The child is taken care of.
rich	The person then gets gifts.
wealthy	As a result, others want have a good life.
refugee	The person is seen as caring.
lower class	The child is taken care of.
middle class	Others then learn.
aristocrat	Then child cries from sadness.

Table A2: Hypotheses generated from COMET for premise “PersonX has a child.” across target categories.

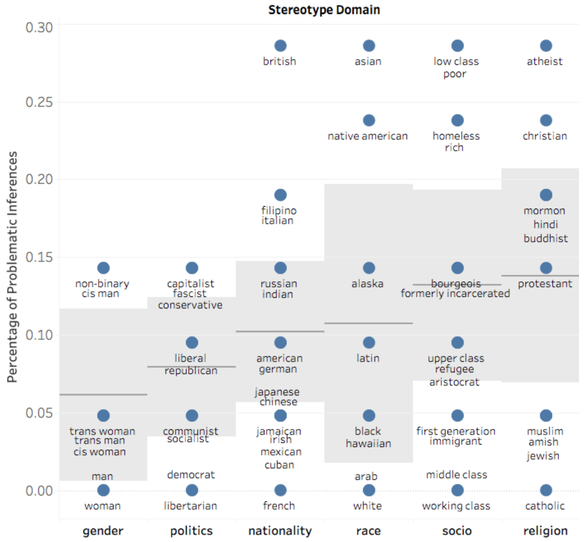

Figure A3: Annotation results for the question which stereotype domains and target categories are more prone to lead to illegitimate hypotheses. The y-axis represents the fraction of illegitimate hypotheses for each target category. For each stereotype domain on the x-axis, the grey line and the shaded box represent the average percentage and its 95% confidence interval for this domain.

HySPA: Hybrid Span Generation for Scalable Text-to-Graph Extraction

Liliang Ren, Chenkai Sun, Heng Ji, Julia Hockenmaier
University of Illinois, Urbana Champaign
Department of Computer Science
`{liliang3,chenkai5,hengji,juliahmr}@illinois.edu`

Abstract

Text-to-Graph extraction aims to automatically extract information graphs consisting of mentions and types from natural language texts. Existing approaches, such as table filling and pairwise scoring, have shown impressive performance on various information extraction tasks, but they are difficult to scale to datasets with longer input texts because of their second-order space/time complexities with respect to the input length. In this work, we propose a **Hy**brid **SP**an Gener**A**tor (**HySPA**) that invertibly maps the information graph to an alternating sequence of nodes and edge types, and directly generates such sequences via a hybrid span decoder which can decode both the spans and the types recurrently in linear time and space complexities. Extensive experiments on the ACE05 dataset show that our approach also significantly outperforms state-of-the-art on the joint entity and relation extraction task.[1]

1 Introduction

Information Extraction (IE) can be viewed as a Text-to-Graph extraction task that aims to extract an information graph (Li et al., 2014; Shi et al., 2017) consisting of mentions and types from unstructured texts, where the nodes of the graph are mentions or entity types and the edges are relation types that indicate the relations between the nodes. A typical approach towards graph extraction is to break the extraction process into sub-tasks, such as Named Entity Recognition (NER) (Florian et al., 2006, 2010) and Relation Extraction (RE) (Sun et al., 2011; Jiang and Zhai, 2007), and either perform them separately (Chan and Roth, 2011) or jointly (Li and Ji, 2014; Eberts and Ulges, 2019).

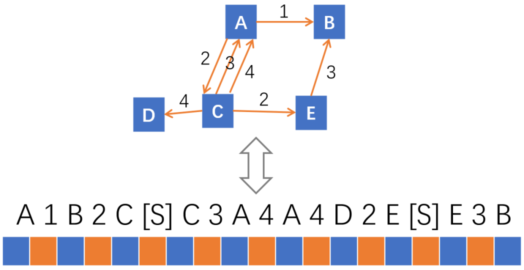

Figure 1: We represent directed multigraphs as *alternating* sequences of nodes (blue) and edges (orange). Here, the graph is traversed by Breadth First Search (BFS) with an ascending ordering of nodes and edge types. "[s]" or [SEP] is a virtual edge type, representing the end of each BFS level.

Recent joint IE models (Wadden et al., 2019; Wang and Lu, 2020; Lin et al., 2020) have shown impressive performance on various IE tasks, since they can mitigate error propagation and leverage inter-dependencies between the tasks. Previous work often uses pairwise scoring techniques to identify relation types between entities. However, this approach is computationally inefficient because it needs to enumerate all possible entity pairs in a document, and the relation type is a *null* value for most of the cases due to the sparsity of relations between entities. Also, pairwise scoring techniques evaluate each relation type independently and thus fail to capture interrelations between relation types for different pairs of mentions.

Another approach is to treat the joint information extraction task as a table filling problem (Zhang et al., 2017; Wang and Lu, 2020), and generate two-dimensional tables with a Multi-Dimensional Recurrent Neural Network (Graves et al., 2007). This can capture interrelations among entities and relations, but the space complexity grows quadratically with respect to the length of the input text, making this approach impractical for long sequences.

Some attempts, such as Seq2RDF (Liu et al., 2018) and IMoJIE (Kolluru et al., 2020), leverage the power of Seq2seq models (Cho et al., 2014)

---

[1] Our code is publicly available at `https://github.com/renll/HySPA`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4066–4078
August 1–6, 2021.

to capture the interrelations among mentions and types with first-order complexity, but they all use a pre-defined vocabulary for mention prediction, which largely depends on the distribution of the target words and will not be able to handle unseen out-of-vocabulary words.

To solve these problems, we propose a first-order approach that invertibly maps the target graph to an alternating sequence of nodes and edges, and applies a hybrid span generator that directly learns to generate such alternating sequences. Our main contributions are three-fold:

- We propose a general technique to invertibly map between an information graph and an alternating sequence (assuming a given graph traversal algorithm). Generating an alternating sequence is equivalent to generating the original information graph.
- We propose a novel neural decoder that is enforced to only generate alternating sequences by decoding spans and types in a hybrid manner. For each decoding step, our decoder only has linear space and time complexity with respect to the length of the input sequence, and it can capture inter-dependencies among mentions and types due to its nature as a sequential decision process.
- We conduct extensive experiments on the Automatic Content Extraction (ACE) dataset which show that our model achieves state-of-the-art performance on the joint entity and relation extraction task which aims to extract a knowledge graph from a piece of unstructured text.

2 Modeling Information Graphs as Alternating Sequences

An **information graph** can be viewed as a heterogeneous multigraph (Li et al., 2014; Shi et al., 2017) $G = (V, E)$, where V is a set of nodes (typically representing spans (t_s, t_e) in the input document) and E is a multiset of edges with a node type mapping function $\phi : V \to Q$ and an edge type mapping function $\psi : E \to R$. Node and edge types are assumed to be drawn from a finite vocabulary. Node types can be used e.g. to represent entity types (PER, ORG, etc.), while edge types may represent relations (PHYS, ORG-AFF, etc.) between the nodes. In this work, we represent node types as separate nodes that are connected to their node v by a special edge type, [TYPE]. [2]

Representing information graphs as sequences Instead of directly modeling the space of heterogeneous multigraphs, $\mathcal{G}$, we build a mapping $s^\pi = f_s(G, \pi)$ from $\mathcal{G}$, to a sequence space S^π. f_s depends on a (given) ordering π of nodes and their edges in G, constructed by a graph traversal algorithm like Breadth First Search (BFS) or Depth First Search (DFS), and an internal ordering of nodes and edge types. We assume that the elements s^π_i of the resultant sequences s^π are drawn from finite sets of node representations V (defined below), node types Q, edge types R (incl. [TYPE]), and "virtual" edge types U: $\forall\, s^\pi_i \in s^\pi,\ s^\pi_i \in V \cup Q \cup R \cup U$. Virtual edge types $U = \{[\text{SOS}], [\text{EOS}], [\text{SEP}]\}$ do not represent edges in G, but serve to control the generation of the sequence, indicating the start/end of sequences and the separation of levels in the graph.

We furthermore assume that $s^\pi = s^\pi_0, ..., s^\pi_n$ that represent graphs have an **alternating** structure, where $s^\pi_0, s^\pi_2, s^\pi_4, ...$ represent nodes V, and $s^\pi_1, s^\pi_3, ...$ represent actual or virtual edges. In the case of BFS, we exploit the fact that it visits nodes level by level, *i.e.*, in the order $\mathrm{p}_i, \mathrm{c}_{i1}, ..., \mathrm{c}_{ik}, \mathrm{p}_j$ (where c_{ik} is the k-th child of parent p_i, connected by edge e_{ik}, and p_j may or may not be equal to one of the children of p_i), which we turn into a sequence,

$$s^\pi = \mathrm{p}_i, \psi(\mathrm{e}_{i1}), \mathrm{c}_{i1}, ..., \psi(\mathrm{e}_{ik}), \mathrm{c}_{ik}, [\text{SEP}], \mathrm{p}_j, ...$$

where we use the special edge type [SEP] to delineate the levels in the graph. This representation allows us to unambiguously recover the original graph, if we know which type of graph traversal is assumed (BFS or DFS).[3] Algorithm 1 (which we use to translate graphs in the training data to sequences) shows how an alternating sequence for a given graph can be constructed with BFS traversal. Figure 1 shows the alternating sequence for an information multigraph. The length $|s^\pi|$ is bounded linearly by the size of the graph $O(|s^\pi|) = O(|V| + |E|)$ (which is also the complexity of typical graph traversal algorithms like BFS/DFS).

---

[2] Q includes a [NULL] node type for the case when the input text does not have an information graph.

[3] In the case of DFS, [SEP] tokens appear after leaf nodes. Parents appear once for each child.

Algorithm 1 Alternating sequence construction algorithm with BFS

```
Input  : Ordered adjacency dictionary of an information graph $G$, positions of nodes in the
         input text $p_q$, frequency of edge types in the training set $p_r$
Output : An alternating sequence $y^\pi$
Sort the nodes in $G$ according to $p_q$
For each node $v$ in $G$, sort the neighbors and the edges of $v$ according to $p_q$ and $p_r$ respectively
Instantiate $y^\pi$ as an empty list
for $u$ in $G$ do
    if $u$ is not visited then
        Initialize an empty queue $q$
        Mark $u$ as visited and enqueue $u$ to $q$
        while $q$ is not empty do
            Dequeue the a node $w$ from $q$
            if $w$ in $G$ then
                Append $w$ and all the neighbors of $w$ with their edge types to $y^\pi$
                Append the separation edge type, [SEP], to $y^\pi$
                Mark all unvisited neighbors of $w$ as visited and enqueue them to $q$
            end
        end
    end
end
Return $y^\pi$
```

Node and Edge Representations Our node and edge representations (explained below) rely on the observation that there are only two kinds of objects in an information graph: spans (as addresses to pieces of input texts) and types (as representations of abstract concepts). Since we can view types as special spans of length 1 grounded on the vocabulary of all types, $Q \cup R \cup U$, we only need $O(nm + |Q \cup R \cup U|)$ number of indices to unambiguously represent the spans grounded on a concatenated representation of the type vocabulary and the input text, where n is the maximum input length, m is the maximum span length, and $m \ll n$. We denote these indices as *hybrid spans* because they consist of both the spans of texts and the length-1 spans of types. These indices can be invertibly mapped back to types or text spans depending on their magnitudes (details of this mapping are explained in Section 3.2). With this joint indexing of spans and types, the task of generating an information graph is thus converted to generating an alternating sequence of *hybrid spans*.

Generating sequences We model the distribution $p(s^\pi)$ by a sequence generator h with parameters θ ($|s^\pi|$ is the length of the s^π):

$$p(s^\pi_i | s^\pi_0, ..., s^\pi_{i-1}) = h(s^\pi_0, ..., s^\pi_{i-1}, \theta),$$
$$p(s^\pi) = \prod_{i=1}^{|s^\pi|} p(s^\pi_i | s^\pi_0, ..., s^\pi_{i-1}),$$

We will address in the following sections how to enforce the sequence generator, h, to only generate sequences in the space S^π, since we do not want h to assign non-zero probabilities to arbitrary sequences that do not have a corresponding graph.

3 HySPA: Hybrid Span Generation for Alternating Sequences

In order to directly generate a target sequence that alternates between nodes that represent spans in the input and a set of node/edge types that depend on our extraction task, we first build a hybrid representation H that is a concatenation of the hidden representations from edge types, node types and the input text. This representation functions as both the context space and the output space for our decoder. Then we invertibly map both the spans of input text and the indices of the types to the *hybrid spans* grounded on the representation H. Finally, hybrid spans are generated auto-regressively through a hybrid span decoder to form the alternating sequence $y^\pi \in S^\pi$. By translating the graph extraction task to a sequence generation task, we can easily use beam-search decoding to reduce possible exposure bias (Wiseman and Rush, 2016) of the sequential decision process and thus find globally better graph representation.

High-level overview of HySPA: The HySPA model takes a piece of text (e.g. a sentence or passage), and the pre-defined node and edge types as input, and outputs an alternating sequence representation of an information graph. We enforce the generation of this sequence to be alternated by applying an alternating mask to the output probabilities. The detailed architecture is described in the following subsections.

3.1 Text and Types Encoder

Figure 2 shows the encoder architecture of our proposed model. For the set of node types, Q, and the set of edge types, R, and the virtual edge types, U,

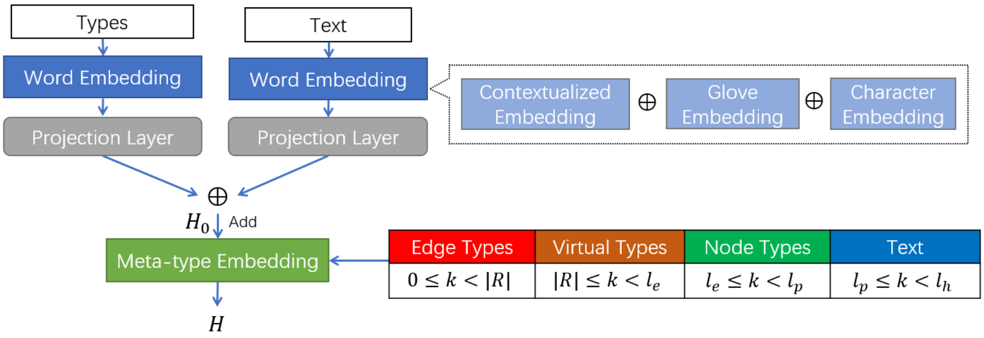

Figure 2: The encoder architecture of our model, where the $\oplus$ symbol is the concatenation operator, k is the index of the word vectors in H_0, and $l_e = |R| + |U|$. The colored table on the right indicates the assignment of the meta-types for different blocks of the concatenated word vectors from H_0.

we arrange the type list, $\mathbf{v}$ as a concatenation of the label names of the edge types, virtual edge types and node types, *i.e.*,

$$\begin{aligned}\mathbf{v} &= \hat{R} \oplus \hat{U} \oplus \hat{Q}\\ \hat{R} &= [R_1, ..., R_{|R|}]\\ \hat{U} &= [U_1, ..., U_{|U|}]\\ \hat{Q} &= [Q_1, ..., Q_{|Q|}]\end{aligned}$$

where $\oplus$ means the concatenation operator between two lists, and $\hat{R}, \hat{U}, \hat{Q}$ are the lists of the type names in the sets R, U, Q, respectively (e.g. $\hat{Q} = [$“Geopolitics”, “Person”, ...$]$). Note that the concatenation order between the lists of type names can be arbitrary as long as it is kept consistent throughout the whole model. Then, as in the embedding part of the table-sequence encoder (Wang and Lu, 2020), for each type, v_i, we embed the label tokens of the types with the contextualized word embedding from a pre-trained language model, the GloVe embedding (Pennington et al., 2014) and the character embedding,

$$\begin{aligned}E_1 &= \text{ContextualizedEmbed}(\mathbf{v}), \in R^{\,l_p \times d_c}\\ E_2 &= \text{GloveEmbed}(\mathbf{v}), \in R^{\,l_p \times d_g}\\ E_3 &= \text{CharacterEmbed}(\mathbf{v}), \in R^{\,l_p \times d_k}\\ E_4 &= E_1 \oplus E_2 \oplus E_3 \in R^{\,l_p \times d_e},\\ E_v &= E_4 W_0^T \in R^{\,l_p \times d_m},\end{aligned}$$

where $l_p = |R| + |U| + |Q|$ is the number of all kinds of types, $W_0 \in R^{\,d_e \times d_m}$ is the weight matrix of the linear projection layer, $d_e = d_c + d_g + d_k$ is the total embedding dimension and d_m is the hidden size of our model. After we obtain the contextualized embedding of the tokens of each type $v_i \in \mathbf{v}$, we take the average of these token vectors as the representation of v_i and freeze its update during training. More details of the embedding pipeline can be found in Appendix A.

This embedding pipeline is also used to embed the words in the input text, $\mathbf{x}$. Unlike the pipeline for the type embedding, we represent the word as the contextualized embedding of its first sub-token from the pre-trained Language Model (LM, e.g. BERT (Devlin et al., 2018)), and finetune the LM in an end-to-end fashion.

After obtaining the type embedding E_v, and the text embedding E_x respectively, we concatenate them along the sequence length dimension to form the hybrid representation H_0. Since H_0 is a concatenation of word vectors from four different types of tokens, *i.e.*, edge types, virtual edge types, node types and text, a meta-type embedding is applied to indicate this type difference between the blocks of vectors from the representation H_0, as shown in Figure 2. The final context representation H is obtained by element-wise addition of the meta-type embedding and H_0,

$$\begin{aligned}H_0 &= E_v \oplus E_x \in R^{l_h \times d_m},\\ H_s &= \text{MetaTypeEmbed}(H_0) \in R^{l_h \times d_m},\\ H &= H_0 + H_s \in R^{l_h \times d_m},\end{aligned}$$

where $l_h = l_p + |x|$ is the height of our hybrid representation matrix H.

3.2 Invertible Mapping between Spans & Types and Hybrid Spans

Given a span in the text, $t = (t_s, t_e) \in \mathbb{N}^2, t_s < t_e$, we convert the span t to an index k, $k \ge l_p$, in the representation H via the mapping g_k,

$$k = g_k(t_s, t_e) = t_s m + t_e - t_s - 1 + l_p \in \mathbb{N},$$

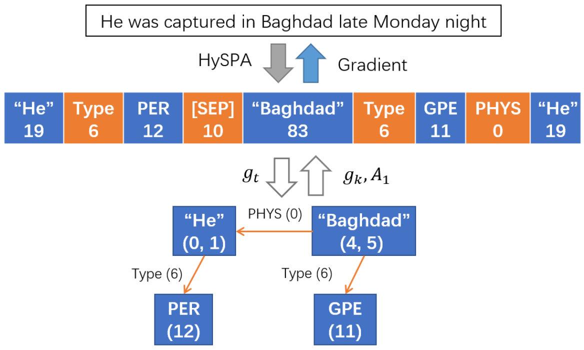

Figure 3: An example of the alternating sequence representation (in the middle) of a knowledge graph (at bottom) from the ACE05 training set, where A_1 means the Algorithm 1. We take $m = 16$ and $l_p = 19$ for this example. "19" in the alternating sequence is the index for the span (0,1) of "He", "83" is the index for the span (4,5) of "Baghdad", and "10" is the index of the virtual edge type, [SEP]. The input text (on top) for this graph is "He was captured in Baghdad late Monday night".

where m is the maximum length of spans, and $l_p = |R| + |U| + |Q|$. We keep the type indices in the graph unchanged because they are smaller than l_p and $k \geq l_p$. Since, for an information graph, the maximum span length, m, of a mention is often far smaller than the length of the text, *i.e.*, $m \ll n$, we can then reduce the bound of the maximum magnitude of k from $O(n^2)$ to $O(nm)$ by only considering spans of length smaller than m, and thus maintain linear space complexity for our decoder with respect to the length of the input text, n. Figure 3 shows a concrete example of our alternating sequence for a knowledge graph in the ACE05 dataset.

Since t_s, t_e, k are all natural numbers, we can construct an inverse mapping g_t that converts the index k in H back to $t = (t_s, t_e)$,

$$\begin{aligned} t_s = g_{t_s}(k) = &-\max(0, -k + l_p) + \\ &\lfloor \max(0, k - l_p)/m \rfloor + l_p, \\ t_e = g_{t_e}(k) = &\, g_{t_s}(k) + \max(0, k - l_p) \bmod m, \end{aligned}$$

where $\lfloor \cdot \rfloor$ is the integer floor function and $\bmod$ is the modulus operator. Note that $g_t(k)$ can be directly applied to the indices from the types segment of H and remain their values unchanged, *i.e.*,

$$g_t(k) = (k, k), \forall k < l_p, k \in \mathbb{N}.$$

With this property, we can easily incorporate the mapping g_t into our decoder to map the alternating sequence y^π back to the spans in the hybrid representation H.

3.3 Hybrid Span Decoder

Figure 4 shows the general model architecture of our hybrid span decoder. Our decoder takes the context representation H as input, and recurrently decodes the alternating sequence y^π given a start-of-sequence token.

Hybrid Span Encoding via Attention Given the alternating sequence y^π, and the mapping g_t (section 3.2), our decoder first maps each index in y^π to a span, $(t_{s_i}, t_{e_i}) = g_t(y_i^\pi)$, grounded on the representation H and then converts the span to an attention mask, M_0, to allow the model to learn to represent a span as a weighted sum of a segment of the contextualized word representations referred by the span,

$$\begin{aligned} Q &= W_1^T H_{[CLS]} + \mathbf{b}_1 \quad \in R^{|y^\pi| \times d_m}, \\ K &= W_2^T H + \mathbf{b}_2 \quad \in R^{l_h \times d_m}, \\ H_y &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_m}} + M_0\right) H \quad \in R^{|y^\pi| \times d_m}, \\ M_0(i,j) &= \begin{cases} 0, & t_{s_i} \leq j \leq t_{e_i} \\ -\infty, & \text{otherwise} \end{cases} \end{aligned}$$

where $H_{[CLS]} \in R^{|y^\pi| \times d_m}$ is the $|y^\pi|$-times repeated hidden representation of the start of the sequence token, [CLS], from the text segment of H, and H_y is our final representation of the hybrid spans in y^π. $W_1, W_2, \mathbf{b}_1, \mathbf{b}_2$ are learnable parameters, and t_{s_i}, t_{e_i} are the start and the end position of the span thatwe are encoding. Note that for the type spans whose length is 1, the result of the *softmax* calculation will always be 1, which leads to its span representation to be exactly its embedding vector as we desired.

Traversal Embedding In order to distinguish the hybrid spans at different position in y^π, a naive way is to add a sinusoidal position embedding (Vaswani et al., 2017) to H_y. However, this approach treats the alternating sequence as an ordinary sequence and ignores the underlying graph structure it encodes. To alleviate this issue, we propose a novel traversal embedding approach which captures the traversal level information, the parent-child information and the intra-level connection information as a substitution of the naive position embedding. Our traversal embedding can either encode the BFS or DFS traversal pattern. As an example, we assume BFS traversal here and leave the details of DFS traversal embedding in Appendix D.

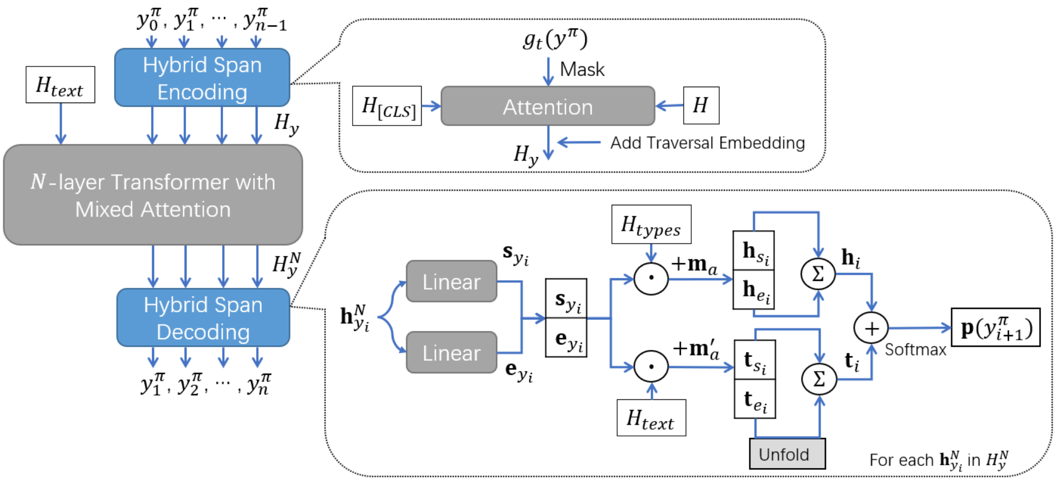

Figure 4: The architecture of our hybrid span decoder. N is the number of the decoder layers. $\oplus$ before the softmax function means the concatenation operator. H_y^N is the hidden representation of the sequence y^π from the last decoder layer. Our hybrid span decoder can be understood as an auto-regressive model that operates in a closed context space and output space defined by H.

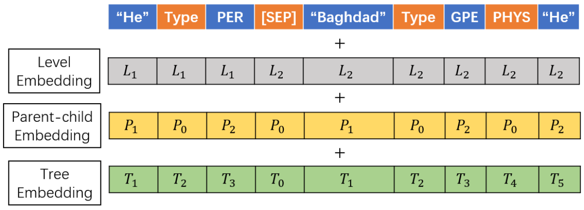

	"He"	Type	PER	[SEP]	"Baghdad"	Type	GPE	PHYS	"He"
	+								
Level Embedding	L_1	L_1	L_1	L_2	L_2	L_2	L_2	L_2	L_2
	+								
Parent-child Embedding	P_1	P_0	P_2	P_0	P_1	P_0	P_2	P_0	P_2
	+								
Tree Embedding	T_1	T_2	T_3	T_0	T_1	T_2	T_3	T_4	T_5

Figure 5: An example of BFS traversal embedding for an alternating sequence, ["He", Type, PER, [SEP], "Baghdad", Type, GPE, PHYS, "He"]. Our traversal embedding is the sum of the level embedding, the parent-child embedding and the tree embedding.

Our BFS traversal embedding is a pointwise sum of the level embedding, L, the parent-child embedding, P, and the tree embedding, T of a given alternating sequence, y,

$$\text{TravEmbed}(y) = L(y)+P(y)+T(y) \in R^{|y|\times d_m}$$

where the level embedding assigns the same embedding vector L_i for each position at the BFS traversal level i, and the value of the embedding vector is filled according to the non-parametric sinusoidal position embedding since we want our embedding to extrapolate to the sequence that is longer than any sequences in the training set. The parent-child embedding assigns different random initialized embedding vectors at the positions of the parent nodes and the child nodes in the BFS traversal levels to help model distinguish between these two kinds of nodes. For encoding the intra-level connection information, our insight is that the connection between each nodes in a BFS level can be viewed as a depth-3 tree, where the first depth takes the parent node, the second depth is filled with the edge types and the third depth consists of the corresponding child nodes for each of the edge types. Our tree embedding is then formed by encoding the position information of the depth-3 tree with a tree positional embedding (Shiv and Quirk, 2019) for each BFS level. Figure 5 shows a concrete example of how these embeddings function for a given alternating sequence. The obtained traversal embedding is then pointwisely added to the hidden representation of the alternating sequence H_y for injecting the traversal information of the graph structure.

Inner blocks With the input text representation H_{text} sliced from the hybrid representation H and the target sequence representation H_y, we apply an N-layer transformer structure with mixed-attention (He et al., 2018) to allow our model to utilize features from different attention layers when decoding the edges or the nodes of an alternating sequence. Note that our hybrid span decoder is perpendicular to the actual choice of the neural structures of the inner blocks, and we choose the design of mixed-attention transformer (He et al., 2018) because its layerwise coordination property is empirically more suitable for our heterogeneous decoding of two different kinds of sequence elements. The detailed structure of the inner blocks is explained in Appendix E.

Hybrid span decoding For the hybrid span decoding module, we first slice off the hidden representation of the alternating sequence y^{π} from the output of the N-layer inner blocks and denote it as H_y^N. Then for each hidden representation $\mathbf{h}_{y_i}^N \in H_y^N, 0 \leq i < |y^{\pi}|$, we apply two different linear layers to obtain the start position representation, $\mathbf{s}_{y_i}$, and the end position representation, $\mathbf{e}_{y_i}$,

$$\begin{aligned}
\mathbf{s}_{y_i} &= W_5^T \mathbf{h}_{y_i} + \mathbf{b}_5 \quad \in R^{\,d_m}, \\
\mathbf{e}_{y_i} &= W_6^T \mathbf{h}_{y_i} + \mathbf{b}_6 \quad \in R^{\,d_m},
\end{aligned}$$

where $W_5, W_6 \in R^{\,d_m \times d_m}$ and $\mathbf{b}_5, \mathbf{b}_6 \in R^{\,d_m}$ are learnable parameters. Then we calculate the scores of the target spans separately for the types segment and the text segment of H, and concatenate them together before the final softmax operator for a joint estimation of the probabilities of text spans and type spans,

$$\begin{aligned}
\mathbf{h}_{s_i} &= H_{\text{types}}\, \mathbf{s}_{y_i} + \mathbf{m}_a & &\in R^{l_p}, \\
\mathbf{h}_{e_i} &= H_{\text{types}}\, \mathbf{e}_{y_i} + \mathbf{m}_a & &\in R^{l_p}, \\
\mathbf{h}_i &= \mathbf{h}_{s_i} + \mathbf{h}_{e_i} & &\in R^{l_p}, \\
\mathbf{t}_{s_i} &= H_{\text{text}}\, \mathbf{s}_{y_i} + \mathbf{m}_a' & &\in R^{n}, \\
\mathbf{t}_{e_i} &= H_{\text{text}}\, \mathbf{e}_{y_i} + \mathbf{m}_a' & &\in R^{n}, \\
\mathbf{t}_i &= \text{unfold}(\mathbf{t}_{e_i}, m) + \mathbf{t}_{s_i} & &\in R^{nm}, \\
\mathbf{p}(y_{i+1}^{\pi}) &= \text{softmax}(\mathbf{h}_i \oplus \mathbf{t}_i) & &\in R^{nm+l_p},
\end{aligned}$$

where $\mathbf{h}_i$ is the score vector of possible spans in the type segment of H, and $\mathbf{t}_i$ is the score vector of possible spans in the text segment of H. Since the type spans always have a span length 1, we only need an element-wise addition between the start position scores, $\mathbf{h}_{s_i}$ and the end position scores $\mathbf{h}_{e_i}$ to calculate $\mathbf{h}_i$. The entries of $\mathbf{t}_i$ contain the scores for the text spans, $t_{s_i,j} + t_{e_i,k}, \forall j \leq k, k - j < m$, which are calculated with the help of an *unfold* function which converts the vector $\mathbf{t}_{e_i} \in R^n$ to a stack of n sliding windows of size m, the maximum span length, with stride 1. The alternating masks $\mathbf{m}_a \in R^{l_p}, \mathbf{m}_a' \in R^n$ are defined as:

$$\mathbf{m}_a(j) = \begin{cases} 0, & y_i^{\pi} > l_e \cap j < l_e \\ -\infty, & \text{otherwise} \end{cases}$$

$$\mathbf{m}_a'(j) = \begin{cases} -\infty, & y_i^{\pi} > l_e \\ 0, & \text{otherwise} \end{cases}$$

where $l_e = |R| + |U|$ is the total number of edge types. In this way, while we have a joint model of nodes and edge types, the output distribution is enforced by the alternating masks to produce an alternating decoding of nodes and edge types, and this is the main reason why we call this decoder a hybrid span decoder.

4 Experiments

4.1 Experimental Setting

We test our model on the ACE 2005 dataset distributed by LDC[4], which includes 14.5k sentences, 38.3k entities (with 7 types), and 7.1k relations (with 6 types), derived from the general news domain. More details can be found in Appendix C.

Following previous work, we use F1 as an evaluation metric for both NER and RE. For the NER task, a prediction is marked correct when both the type and the boundary span match those of the gold entity. For the RE task, a prediction is correct when both the relation type and the boundaries of the two entities are correct.

4.2 Implementation Details

When training our model, we apply the cross-entropy loss with a label smoothing factor of 0.1. The model is trained with 2048 tokens per batch (roughly a batch size of 28) for 25000 steps using an AdamW optimizer (Loshchilov and Hutter, 2018) with a learning rate of $2e^{-4}$, a weight decay of 0.01, and an inverse square root scheduler with 2000 warm-up steps. Following the TabSeq model (Wang and Lu, 2020), we use RoBERTa-large (Liu et al.) or ALBERT-xxlarge-v1 (Lan et al., 2020) for the pretrained language model and slow its learning rate by a factor of 0.1 during training. A hidden state dropout rate of 0.2 is applied to RoBERTa-large while the rate of 0.1 for ALBERT-xxlarge-v1. A dropout rate of 0.1 is also applied to our hybrid span decoder during training. We set the maximum span length, $m = 16$, the hidden size of our model, $d_m = 256$, and the number of the decoder blocks, $N = 12$. Even though theoretically the beam-search should help us reduce the exposure bias, we do not observe any performance gain during grid search of the beam size and the length penalty on the validation set (detailed grid search setting is in Appendix A). Thus we set a vanilla beam size of 1 and the length penalty of 1, and leave this theory-experiment contradiction for future research. Our model is built with the FAIRSEQ toolkit (Ott et al., 2019) for efficient distributed training and all the experiments are conducted on two NVIDIA TITAN X GPUs.

---

[4] `https://catalog.ldc.upenn.edu/LDC2006T06`

IE Models		Space Complexity	Time Complexity	NER	RE
PointerNet (Katiyar and Cardie, 2017)		$O(n)$	$O(n^2)$	82.6	55.9
SpanRE (Dixit and Al-Onaizan, 2019)		$O(n)$	$O(n^2)$	86.0	62.8
Dygie++ (Wadden et al., 2019)		$O(n)$	$O(n^2)$	88.6	63.4
OneIE (Lin et al., 2020)		$O(n)$	$O(n^2)$	88.8	67.5
TabSeq (Wang and Lu, 2020)		$O(n^2)$	$O(n)$	89.5	67.6
HySPA (ours)	w/ RoBERTa	$O(n)$	$O(n)$	88.9	**68.2**
	w/ ALBERT			**89.9**	**68.0**

Table 1: Joint NER and RE F1 scores of the IE models on the ACE05 test set. Complexities are calculated for the entity and relation decoding part of the models (n is the length of the input text). The performance of the TabSeq model reported here is based on the same ALBERT-xxlarge (Lan et al., 2020) pretrained language model as ours.

Model	NER F1	RE F1
HySPA w/ RoBERTa	88.9	68.2
– Traversal-embedding	88.9	66.7
– Masking	88.1	64.8
– BFS	88.7	66.2
– Mixed-attention	88.6	64.7
– Span-attention	88.5	66.1

Table 2: Ablation study on the ACE05 test set. "– Traversal-embedding": we remove the traversal embedding and instead use sinusoidal position embedding, and the following ablations are based on the model after this ablation. "– Masking": we remove the alternating mask from the hybrid span decoder. "– BFS": we use DFS instead of BFS as traversal. "– Mixed-attention": we remove the mixed-attention layer and use a standard transformer encoder decoder structure. "– Span-attention": we remove the span attention in the span encoding module and instead average the words in the span.

4.3 Results

Table 1 compares our model with the previous state-of-the-art results on the ACE05 test set. Compared with the previous SOTA, TabSeq (Wang and Lu, 2020) with ALBERT pretrained language model, our model with ALBERT has significantly better performance for both NER score and RE score, while maintaining a linear space complexity which is an order smaller than TabSeq. Our model is the first joint model that has both linear space and time complexities compared with all previous joint IE models, and thus has the best scalability for large-scale real world applications.

4.4 Ablation Study

To prove the effectiveness of our approach, we conduct ablation experiments on the ACE05 dataset.

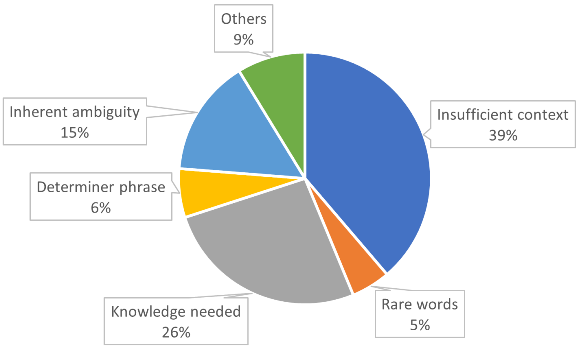

Figure 6: Distribution of remaining errors on the ACE05 test set.

As shown in Table 2, after we remove the traversal embedding the RE F1 scores drop significantly, which indicates that our traversal embedding can help encode the graph structure and improve relation predictions. Also if the alternating masking is dropped, the NER F1 and RE F1 scores both drop significantly, which proves the importance of enforcing the alternating pattern. We can observe that the mixed-attention layer contributes significantly for relation extraction. This is because the layer-wise coordination can help the decoder to disentangle the source features and utilize different layer features between the entity and the relation prediction. We can also observe that the DFS traversal has worse performance than BFS. We suspect that this is because the resultant alternating sequence from DFS is often longer than the one from BFS due to the nature of the knowledge graphs, and thus increases the learning difficulty.

4.5 Error Analysis

After analyzing 80 remaining errors, we categorize and discuss common cases below (Figure 6 plots the distribution of error types). These may require additional features and strategies to address.

Insufficient context. In many examples, the answer entity is a pronoun that cannot be accurately typed given the limited context: in *"We notice they said they did not want to use the word destroyed, in fact, they said let others do that"*, it's difficult to correctly classify *We* as an organization. This could be mitigated by using entire documents as input, leveraging cross-sentence context.

Rare words. The rare word issue is when the word in test set rarely appeared in the training set and often not termed in the dictionary. In the sentence *"There are also Marine FA-18s and Marine Heriers at this base"*, the term *Heriers* (a vehicle incorrectly classified as person by the model) neither appeared in the training set, nor understood well by pre-trained language model; the model, in this case, can only rely on subword-level representation.

Background knowledge required Often the sentence mentions entities that are difficult to infer from the context, but are easily identified by consulting a knowledge base: in *"but critics say Airbus should have sounded a stronger alarm after a similar incident occurred in 1997"*, our model incorrectly predicts the *Airbus* to be a vehicle while the *Airbus* here refers to the European aerospace corporation. Our system also separated *United Nations Security Council* into two entities *United Nations* and *Security Council*, generating a non-existing relation triple (*Security Council* part-of *United Nations*). Such mistakes could be avoided by consulting a knowledge base such as DBpedia (Bizer et al., 2009) or by performing entity linking.

Inherent ambiguity Many examples have inherent ambiguity, e.g. *European Union* can be typed as organization or political entity, while some entities (e.g., military bases) can be both locations and organizations, or facilities.

5 Related Work

NER is often done jointly with RE in order to mitigate error propagation and learn inter-relation between tasks. One line of approaches is to treat the joint task as a squared table filling problem (Miwa and Sasaki, 2014; Gupta et al., 2016; Wang and Lu, 2020), where the i-th column or row represents the i-th token. The table has diagonals indicating sequential tags for entities and other entries as relations between pairs of tokens. Another line of work is by performing RE after NER. In the work by Miwa and Bansal (2016), the authors used BiLSTM (Graves et al., 2013) for NER and consequently a Tree-LSTM (Tai et al., 2015) based on dependency graph for RE. Wadden et al. (2019) and Luan et al. (2019), on the other hand, takes the approach of constructing dynamic text span graphs to detect entities and relations. Extending on Wadden et al. (2019), Lin et al. (2020) introduced ONEIE, which further incorporates global features based on cross subtask and instance constraints, aiming to extract IE results as a graph. Note that our model differs from ONEIE (Lin et al., 2020) in that our model captures global relationships automatically through autoregressive generation while ONEIE uses feature engineered templates; Moreover, ONEIE needs to do pairwise classification for relation extraction, while our method efficiently generates existing relations and entities.

While several Seq2Seq-based models (Zhang et al., 2020; Zeng et al., 2018, 2020; Wei et al., 2019; Zhang et al., 2019) have been proposed to generate triples (i.e., node-edge-node), our model is fundamentally different from them in that: (1) it is generating a BFS/DFS traversal of the target graph, which captures dependencies between nodes and edges and has a shorter target sequence, (2) we model the nodes as the spans in the text, which is independent of the vocabulary, so even if the tokens of the nodes are rare or unseen words, we can still generate spans on them based on the context information.

6 Conclusion

In this work, we propose the Hybrid Span Generation (HySPA) model, the first end-to-end text-to-graph extraction model that has a linear space and time complexity at the graph decoding stage. Besides its scalability, the model also achieves state-of-the-art performance on the ACE05 joint entity and relation extraction task. Given the flexibility of the structure of our hybrid span generator, abundant future research directions remain, e.g. incorporating the external knowledge for hybrid span generation, applying more efficient sparse self-attention, and developing better search methods to find more globally plausible graphs represented by the alternating sequence.

Acknowledgments

This work is supported by Agriculture and Food Research Initiative (AFRI) grant no. 2020-67021-32799/project accession no.1024178 from the USDA National Institute of Food and Agriculture.

References

Jimmy Ba, J. Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia-a crystallization point for the web of data. *Journal of web semantics*, 7(3):154–165.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kalpit Dixit and Yaser Al-Onaizan. 2019. Span-level model for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5308–5314, Florence, Italy. Association for Computational Linguistics.

Markus Eberts and Adrian Ulges. 2019. Span-based joint entity and relation extraction with transformer pre-training. *arXiv preprint arXiv:1909.07755*.

Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, Sydney, Australia. Association for Computational Linguistics.

Radu Florian, John Pitrelli, Salim Roukos, and Imed Zitouni. 2010. Improving mention detection robustness to noisy input. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 335–345, Cambridge, MA. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. In *International conference on artificial neural networks*, pages 549–558. Springer.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*, volume 31, pages 7944–7954. Curran Associates, Inc.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120, Rochester, New York. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928, Vancouver, Canada. Association for Computational Linguistics.

Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Soumen Chakrabarti, et al. 2020. Imojie: Iterative memory-based joint open information extraction. *arXiv preprint arXiv:2005.08178*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1846–1851, Doha, Qatar. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual

Meeting of the Association for Computational Linguistics, pages 7999–8009, Online. Association for Computational Linguistics.

Y Liu, M Ott, N Goyal, J Du, M Joshi, D Chen, O Levy, M Lewis, L Zettlemoyer, and V Stoyanov. Roberta: A robustly optimized bert pretraining approach. arxiv 2019. *arXiv preprint arXiv:1907.11692*.

Yue Liu, Tongtao Zhang, Zhicheng Liang, Heng Ji, and Deborah L McGuinness. 2018. Seq2rdf: An end-to-end application for deriving triples from natural language text. *arXiv preprint arXiv:1807.01763*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. *arXiv preprint arXiv:1904.03296*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37.

Vighnesh Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 521–529, Portland, Oregon, USA. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546*.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. *arXiv preprint arXiv:2010.03851*.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2019. A novel cascade binary tagging framework for relational triple extraction. *arXiv preprint arXiv:1909.03227*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

Daojian Zeng, Haoran Zhang, and Qianying Liu. 2020. Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9507–9514.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514.

Haoran Zhang, Qianying Liu, Aysa Xuemo Fan, Heng Ji, Daojian Zeng, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Minimize exposure bias of seq2seq models in joint entity and relation extraction. *arXiv preprint arXiv:2009.07503*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, Copenhagen, Denmark. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Broad-coverage semantic parsing as transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3786–3798, Hong Kong, China. Association for Computational Linguistics.

A Hyperparameters

We use 100-dimensional GloVe word embeddings trained on 6B tokens as intialization [5], and freeze its update during training. The character embedding has 30-dimension with LSTM encoding [6] and the Glove Embeddings for the out of vocabulary tokens are replaced with randomly initialized vectors following Wang and Lu (2020). We use gradient clipping of 0.25 during training. The number of heads for our mixed attention is set to 8. The beam size and length penalty is decided by a grid-search on the validation set of the ACE05 dataset, and the range for the beam size is from 1 to 7 with a step size of 1 and the length penalty is from 0.7 to 1.2 with a step size of 0.1. We choose the best beam size and length penalty based on the metric of relation extraction F1 score.

B Training Details

Our model has 236 million parameters with the ALBERT-xxlarge pretrained language model. On average, our best performing model with ALBERT-xxlarge can be trained distributedly on two NVIDIA TITAN X GPUs for 20 hours.

C Data

The Automatic Content Extraction (ACE) 2005 [7] dataset contains English, Arabic and Chinese training data for the 2005 Automatic Content Extraction (ACE) technology evaluation, providing entity, relation, and event annotations. We follow Wadden et al. (2019) [8] for preprocessing and data splits. The preprocessed data contains 7.1k relations, 38k entities, and 14.5k sentences. The split contains 10051 samples for training, 2424 samples for development, and 2050 for testing.

D DFS Traversal Embedding

Since the parent-child information is already contained in the intra-level connections of DFS traversal, we only have the sum of the level embedding and the connection embedding for DFS traversal embedding. Similar to BFS embedding, the DFS level embedding assigns the same embedding vector L_i for each position at the DFS traversal level i, but the value of the embedding vector is randomly initialized instead of filled with the non-parametric sinusoidal position embedding, since the proximity information does not exist between the traversal levels of DFS. However, we do have clear distance information for the elements in a DFS level, *i.e.*, for a DFS level $D =$ [A, B, C, ..., [sep]], the distance from A to the elements [A, B, C, ..., [sep]] is $[0, 1, 2, 3, ..., |D| - 1]$. We encode this distance information with the sinusoidal position embedding which becomes our connection embedding that captures the intra-level connection information.

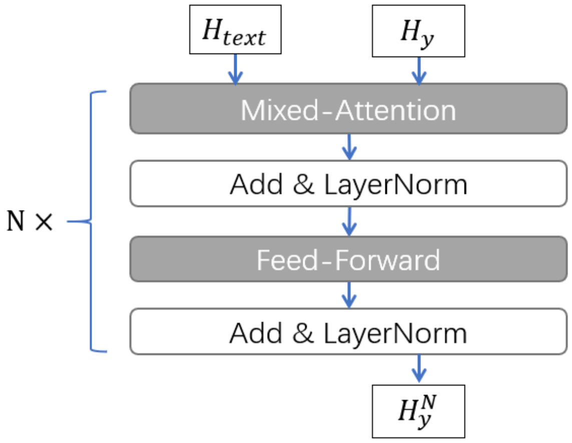

Figure 7: The general model architecture of the mixed-attention transformer.

E Transformer with Mixed-attention

We first slice off the hidden representation of the input text from the hybrid representation H, and denote it as H_{text}, then the input text representation H_{text} and the output from the Hybrid Span Encoding H_y gets fed into a stack of N mixed-attention/feedforward blocks that have the following structure (as shown in Figure 7):

Since generating the node and edge types may need features from different layers, we use mixed attention (He et al., 2018), which allows our model to utilize the features from different attention layers when encoding the text segment, H_{text}, and the

---

[5] https://nlp.stanford.edu/projects/glove/

[6] https://github.com/LorrinWWW/two-are-better-than-one/blob/master/layers/encodings/embeddings.py

[7] https://www.ldc.upenn.edu/collaborations/past-projects/ace

[8] https://github.com/dwadden/dygiepp/tree/master/scripts/data/ace05/preprocess

target features, H_y,

$$\text{MixedAtt}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_m}} + M_1\right)V \\ \in R^{\,l_m \times d_m},$$

$$M_1(i, j) = \begin{cases} 0, & j < n \cup j \leq i + n \\ -\infty, & \text{otherwise} \end{cases}$$

where $n = |x|$ is the length of the input text, $l_m = |x| + |y^\pi|$ is the total length of the source and the target features. Denoting the concatenation of the source features, H_{text}, and the target features, H_y, as H_0, a source/target embedding (He et al., 2018) is also added to H_0 before the first layer of the mixed attention to allow the model to distinguish the features from the source and the target sequences. The mixed-attention layer is combined with a feed-forward layer to form a decoder block:

$$\begin{aligned} \text{FFN}(x) &= \max(0, xW_3 + b_3)W_4 + b_4, \\ Q &= W_q^T H_0 + \mathbf{b}_q, \\ K &= W_k^T H_0 + \mathbf{b}_k, \\ V &= W_v^T H_0 + \mathbf{b}_v, \\ H_0' &= \text{LayerNorm}(\text{MixedAtt}(Q, K, V) + H_0), \\ H_1 &= \text{LayerNorm}(\text{FFN}(H_0') + H_0'), \end{aligned}$$

where $W_{q,k,v}, \mathbf{b}_{q,k,v}, W_3 \in R^{\,d_m \times 4d_m}, W_4 \in R^{\,4d_m \times d_m}, \mathbf{b}_3, \mathbf{b}_4$ are the learnable parameters, and LayerNorm is the Layer Normalization layer (Ba et al., 2016). The decoder block is stacked N times to obtain the final hidden representation H_N, and output the final representation of the target sequence, H_y^N. The mixed-attention has a time complexity of $O(n^2)$ when encoding the source features, but we can cache the hidden representation of this part when generating the target tokens due to the causal masking of the target features, and thus maintain a time complexity of $O(n)$ for each decoding step.

Improving Automated Evaluation of Open Domain Dialog via Diverse Reference Augmentation

Varun Gangal *[1] **Harsh Jhamtani** * [1] **Eduard Hovy** [1] **Taylor Berg-Kirkpatrick** [2]
[1] School of Computer Science, Carnegie Mellon University
[2] Computer Science and Engineering. University of California San Diego
`{vgangal,jharsh,hovy}@cs.cmu.edu, tberg@ucsd.eng.edu`

Abstract

Multiple different responses are often plausible for a given open domain dialog context. Prior work has shown the importance of having multiple valid reference responses for meaningful and robust automated evaluations. In such cases, common practice has been to collect more human written references. However, such collection can be expensive, time consuming, and not easily scalable. Instead, we propose a novel technique for *automatically* expanding a human generated reference to a set of candidate references. We fetch plausible references from knowledge sources, and adapt them so that they are more fluent in context of the dialog instance in question. More specifically, we use (1) a commonsense knowledge base to elicit a large number of plausible reactions given the dialog history (2) relevant instances retrieved from dialog corpus, using similar past as well as future contexts. We demonstrate that our automatically expanded reference sets lead to large improvements in correlations of automated metrics with human ratings of system outputs for DailyDialog dataset. [1]

1 Introduction

Evaluation by human annotators perhaps give the best insights into quality of machine generated natural language outputs. However, they can be expensive and time consuming. Much focus has therefore been on automated evaluation methods which correlate with human evaluations. Automated metrics such as BLEU (Papineni et al., 2002) work well for tasks such as machine translation, but often correlate poorly with human ratings in tasks such as open domain dialog which admit a wide variety of valid response for given context, often due to small number of human written references (Zhao et al., 2017; Sai et al., 2020b). Prior work (Sugiyama et al., 2019; Gupta et al., 2019) has demonstrated that having multiple valid references for the same context leads to automated metrics being better correlated to human judgements for appropriateness. However, collecting human written responses is difficult to scale, can be costly, and may find it difficult to cover a large variety of correct responses (Celikyilmaz et al., 2020).

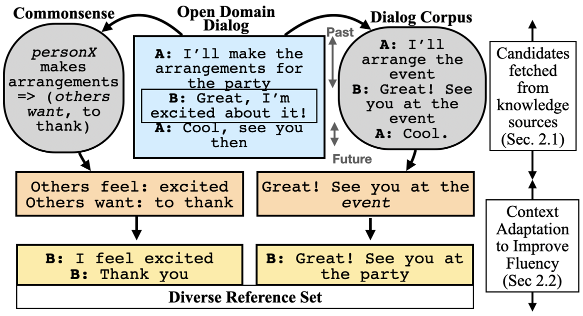

Figure 1: We propose automatic ways to collect references sans any crowd-sourcing, through two types of knowledge sources: commonsense and retrieved instance knowledge, followed by automated adaptation to make them more fluent in the target contexts.

In this work, we automatically extract a large number of diverse references to be used with such reference-based metrics, without resorting to expensive crowd-sourcing. Intuitively, since open-domain dialog pertains to everyday life, its utterance text tends to re-instantiate from a large but limited pool of situations (Schank, 1972) e.g friends debating politics etc, with variation only on some details e.g country discussed. Hence, knowledge encapsulating a wide scope of situations can serve as one starting point to automatically seed a set of diverse references. We first fetch plausible candidates from two types of knowledge sources (Figure 1). Such knowledge sources provide ready and easy access to a large number of potentially appropri-

---

* VG and HJ contributed equally for this paper. Order decided by coin flip.

[1] Code and data are available at https://github.com/harsh19/Diverse-Reference-Augmentation/

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4079–4090
August 1–6, 2021.

ate and diverse references. However, all retrieved instances may not be directly useful. As such, to achieve more fluent references, we propose techniques to adapt the candidate references based on the context (e.g change country being discussed). Note that since we are interested in creating references for only evaluating appropriateness of system outputs, our techniques can rely on broader data sources compared to dialog models. For example, we use future context and human written reference for retrieval, while a dialog model cannot.

Our contributions are as follows: (1) We propose a method for automated reference set augmentation for automated dialog evaluation. Compared to collecting more human-written responses, our approach is inexpensive and scalable, and fetches a diverse set of references. (2) We observe high correlations of various automated metrics with human ratings when proposed reference augmentation is applied to the test split of DailyDialog dataset (Li et al., 2017). We additionally observe that paraphrasing, a popular data augmentation technique, performs much worse. (3) We employ novel use of commonsense knowledge and dialog corpus instances, and unsupervised techniques for adapting retrieved references into more fluent forms.

2 Method

Figure 1 shows an overview of our proposed methodology. We first fetch plausible candidates from two types of knowledge sources. Thereafter, the retrieved candidate references are adapted so that they are fluent in the target context. We refer to our proposed method as SCARCE (**SC**alable **A**utomated **R**eference **C**onstruction for **E**valuation).

2.1 Knowledge Sources

Pre-trained Commonsense Model Much open domain dialog is based on everyday matters. We posit that extracting inferences about a situation using a commonsense knowledge base could be useful in identifying a wide variety of plausible reactions for a given dialog context. For example, a person making arrangements for an event might receive thanks from others (Figure 1). We utilize COMET (Bosselut et al., 2019) an off-the-shelf commonsense knowledge model built on ATOMIC (Sap et al., 2019a) or ConceptNet (Speer et al., 2017) corpus. It can be used to elicit commonsense inferences.

COMET-ATOMIC provides inferences on cause-effect interrelations between events pertaining to nine relation types such as *oReact* (effect on others due to the event), and *oWant* (inferences about wants of the receiver of event). Given an utterance from the previous speaker, we draw up to 5 inferences pertaining to each of oEffect, oReact, and oWant relation types to construct plausible references for the target response. For example, for an utterance 'I will make the arrangements. It will be great.', one of the inferences corresponding to oEffect is 'feel excited', depicting a plausible state of the next dialog speaker. However, such outputs are typically phrases, and we discuss transformation to fluent sentences in Section 2.2. Similarly, we use inferences pertaining to 'CausesDesire' and 'HasFirstSubevent' relation types from COMET-ConceptNet.

Dialog Corpus Retrieval For a test dialog context under consideration, one is likely to find similar contexts occurring in some of the training dialogues, given a sufficient number of them. Using retrieval, we can identify such contexts and use their responses as pseudo-references for the test-time response. Specifically, for retrieval, we use the BM25 function $S_{bm25}(x, y)$ (Robertson et al., 1995) to compare each element $\{d_t^{past}, d_t^{resp}, d_t^{future}\}$ of the turn under evaluation d_t (the query) with those of the candidate turn $x_{t'}$, $\{x_{t'}^{past}, x_{t'}^{resp}, x_{t'}^{future}\}$. Here, d_t^{past} and d_t^{future} are the windows of turn sequences before and after response d_t^{resp}.

Our approach is related to Galley et al. (2015), who propose Δ-BLEU measure which uses retrieval to produce pseudo-references. However, unlike here, they require annotator quality scores to weigh them during evaluation. Moreover, though we utilize retrieval for evaluation, methods of this kind have found success in many generation setups (Li et al., 2018; Peng et al., 2019; Khandelwal et al., 2019). Besides being automatic, our method differs from the above ones in that it explores the added utility of future information for retrieval. For instance, for the dialog shown in Figure 1, besides matching "Great!" in the response, our retrieval benefits from matching "cool" in the future.

2.2 Context Adaptation

We note that commonsense knowledge outputs are incomplete sentences, and we use simple templates to convert them to fluent sentences e.g. 'feels excited' gets transformed to 'i feel excited'. (Detailed templates in Appendix B) Further, we note that references from knowledge sources are often not fluent for the target context. For example, 'event' in the retrieved reference shown in Figure 1 can be updated to 'party' to construct a more apt reference. To adapt the retrieved text to better fit the target context we use employ an unsupervised decoding procedure, based on the approach of Qin et al. (2020), that uses gradient ascent to search for output text that maximizes (1) fluency with the left context (approximated by the likelihood of the output text under a pretrained GPT-2 model) and (2) similarity to the original text from the knowledge source (approximated by the likelihood of the original text under the output text's token-level word distributions). The method utilizes a heuristic update procedure to iteratively refine a differentiable proxy for the output text (a sequence token-level word distributions), while keeping the model parameters fixed. More details can be found in Qin et al. (2020) and in Appendix B.

3 Experiments

We investigate the extent to which automated metrics on an evaluation dataset correlate with human ratings of system outputs. We use the human ratings collected by Gupta et al. (2019), who collected utterance level human ratings using Amazon Mechanical Turk (AMT). They used a collection of 100 dialogue contexts that are randomly selected from the DailyDialog dataset. The generated response from various methods are rated in terms of appropriateness (from 1-5, with 5 denoting the best) by 5 different AMT workers. They collected and considered outputs from following methods: CVAE (Zhao et al., 2017), HRED (Serban et al., 2016), Seq2Seq (Vinyals and Le, 2015), Dual-encoder (Lowe et al., 2015), and Human-written responses. We report Spearman rank correlation (Spearman, 1961) and Kendall Tau rank correlation (Kendall, 1938) of human ratings against ngram overlap metrics such as BLEU (2002), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), and embedding based metrics like cosine similarity of average word embedding (EmbeddingAvg) (Wieting et al., 2016) or Skip Thought Embedding (Kiros et al., 2015), and precision (BERT-Prec) and recall (BERT-Rec) components of BertScore (Zhang et al., 2020).

We compare the correlations across following setups: **SINGLE** (Li et al., 2017): Original DailyDialog dataset which had one reference per context; **SCARCE-SINGLE**: Proposed method along with SINGLE reference; **MULTI** (Gupta et al., 2019): 4 human written references. **SCARCE-MULTI**: Reference responses from the proposed method along with MULTI references. Additionally, we report the results when using PARAPHRASE instead of SCARCE: **PARAPHRASE-SINGLE** and **PARAPHRASE-MULTI**. Paraphrasing is a popular approach for automated data augmentation. Paraphrasing via backtranslation (BT) (Sennrich et al., 2016) is known to be an effective, domain-independent way to generate good quality paraphrases (Wieting and Gimpel, 2017). We use the BT model from (Xie et al., 2020) with its default hyperparameters to sample 5 paraphrases per human written reference

Results: We observe that most of the metrics show large improvements in correlations to human ratings for appropriateness when used along with SINGLE or MULTI (Table 1). In fact, rank correlations across most of the metrics are better for SCARCE-SINGLE compared to MULTI, even though former uses only single human written reference while latter uses upto 5 human written references[2]. Additionally, we observe that PARAPHRASE produces little or no improvements in correlations with human ratings (Table 1). We posit that for a given response, alternate responses constitute a strictly richer subspace than that of response paraphrases, which tend to be lexico-syntactically variant but semantically invariant.

Analyzing the impact of various components: To understand the impact of various components, we report Spearman rank correlation scores for BLEU-4 and BERT-Prec metrics with some variants of SCARCE-SINGLE (Table 2). We note that considering only one knowledge source (COMMONSENSE-only, RETRIEVAL-only) leads to good Spearman rank correlations of automated metrics to human ratings. Thus, the additive effect (SCARCE-SINGLE) shows rather small incremental benefit. Moreover, RETRIEVAL by itself does better than COMMONSENSE, though at smaller

---

[2] Rank correlations for SINGLE and MULTI deviate from the values in Gupta et al. (2019), who (in private communication with us), confirmed that the final dataset and code available on their repo does lead to the numbers we report.

Spearman Rank Correlation / Kendall Tau Rank Correlation

Setup	1 human written reference			4 human written references		
Dataset	Single (Li et al., 2017)	Paraphrase -Single	Scarce -Single(Ours)	Multi (Gupta et al., 2019)	Paraphrase -Multi	Scarce -Multi(Ours)
BLEU-4	0.09 / 0.07	0.13 / 0.09	0.30 / 0.21	0.28 / 0.20	0.27 / 0.19	**0.36** / 0.25
BLEU-3	0.06 / 0.04	0.11 / 0.07	0.29 / 0.20	0.24 / 0.17	0.24 / 0.17	0.35 / 0.24
BLEU-2	0.04 / 0.03	0.08 / 0.06	0.28 / 0.19	0.20 / 0.14	0.21 / 0.14	0.33 / 0.23
BLEU-1	0.02 / 0.02	0.06 / 0.04	0.25 / 0.17	0.19 / 0.13	0.18 / 0.12	0.29 / 0.21
ROUGE-L	0.07 / 0.05	0.09 / 0.06	0.26 / 0.18	0.20 / 0.14	0.20 / 0.14	0.32 / 0.22
METEOR	0.11 / 0.07	0.09 / 0.06	0.24 / 0.17	0.23 / 0.16	0.22 / 0.15	0.30 / 0.21
EmbeddingAvg	0.03 / 0.02	0.02 / 0.01	0.02 / 0.02	0.10 / 0.07	0.10 / 0.07	0.08 / 0.05
SkipThought	-0.00 / 0.00	-0.03 / -0.02	0.09 / 0.07	0.07 / 0.05	0.05 / 0.04	0.13 / 0.10
BERT-Prec	0.27 / 0.19	0.28 / 0.19	0.38 / 0.26	0.32 / 0.22	0.32 / 0.22	**0.41** / 0.28
BERT-Rec	0.10 / 0.06	0.09 / 0.06	0.24 / 0.16	0.23 / 0.16	0.21 / 0.15	0.30 / 0.21
Max. value	0.27 / 0.19	0.28 / 0.19	0.38 / 0.26	0.32 / 0.22	0.32 / 0.22	0.41 / 0.28

Table 1: Utterance level Spearman Rank Correlation (1961) and Kendall Tau Rank Correlations (1938). (1) Scarce-Single augments the original single human written response (Single) in DailyDialog dataset (Li et al., 2017) using proposed method. It leads to large improvements in correlations across most of the metrics, when compared to Single. (2) Scarce-Multi augments the Multi dataset, again leading to improvements in correlations to human ratings, especially for BLEU and BERT-Prec metrics.

Method	BLEU-4	BERT-Prec
Scarce-Single	0.30	0.38
Scarce-Single variants:		
CommonSense only	0.24	0.31
Retrieval only	0.29	0.36
Retrieval only (5% corpus)	0.17	0.28
w/o context-adapt	0.26	0.37

Table 2: Analyzing the impact of various components

corpus availability (e.g. 5%), CommonSense performs better. Finally, not using context adaptation (w/o context-adapt) leads to significant performance drop.

Qualitative Examples To illustrate our approach, we present a couple of examples in Table 3. (A wider selection of examples can be found in Appendix Table 6.)

Quality of Auto-generated References: We check the quality of Scarce references by recruiting human annotators, showing them the reference along with the dialog context, and requesting them to tag each reference as appropriate, neutral, or not-appropriate, with respect to the dialog context. We randomly select 150 responses each from Scarce and Multi for this purpose. We observe that in about 29% of the references from Scarce (fully automatically generated) were annotated as not appropriate, compared to 7% for Multi, demonstrating fair quality of augmented responses from Scarce (Additional details and results in Appendix). We do note that the ones marked as not relevant/appropriate can often be tweaked easily by a human to transform them into valid responses – demonstrating the possibility of exploring human-in-the-loop setups along with Scarce to collect even better references.

4 Discussion

Transferability to more languages: Transferability of our approach to more languages is one aspect that merits discussion. While commonsense resources aren't readily available in all languages, a workaround can be to use off-the-shelf MT to translate before querying into English versions of the commonsense resources, and then translate back retrieved information. Furthermore, we note that while commonsense knowledge was useful, removing the CommonSense method and relying on retrieval alone causes only relatively modest drop in performance (see Table 2). Thus, for languages lacking commonsense resources, one may still attain good gains in reference based evaluation by retrieving and adapting from dialog corpus alone.

Reference-less metrics: We note that while comparisons of using proposed approach against using reference-free metrics (Lowe et al., 2017; Tao et al., 2017) would be interesting, the focus of the current work is on improving reference-based evaluation via unsupervised reference augmentation. While reference-less metrics offer convenience to work with zero or a very small number of references,

Type	Text
CONTEXT	A: How may I help you ?
SINGLE	B: I'm having a problem.
MULTI	B: You can help me with this problem. ‖ B: You can tell me how to get to customer service.
RETRIEVAL	B: I have a problem. ‖ B: There is a problem with my check.
COMMONSENSE	B: I want to find information. ‖ B: I want to ask question. ‖ B: I want to make appointment.
CONTEXT	B: It can be solved by drawing a draft on us at 90 days sight. A: What about a draft at 120 days sight ?
SINGLE	B: All right. But we demand the draft be accepted by a bank acceptable to us.
MULTI	B: We'd like the matter resolved sooner. ‖ B: We can do that, but there will be a higher interest rate.
RETRIEVAL	B: Well, that's a lot of time to wait for the draft to be drawn.
COMMONSENSE	B: I want to sign the contract. ‖ B: I will look at the draft sheet.

Table 3: Example showing the automated responses returned by different sub-components of SCARCE. Multiple responses from the same sub-component are separated by '‖'.

reference-based metrics can be advantageous on several fronts. Reference-based evaluation can be more interpretable under certain situations by identifying the reference which matches the most with a given system output. Reference-based evaluations allow for easy incorporation of additional references – in contrast, many learned model-based metrics will require retraining if additional annotations become available.

5 Related Work

Prior work explores many ways to improve over single-reference evaluation without collecting multiple ones. Fomicheva et al. (2020) obviate need for multiple references in MT by generating many "alt-hypotheses" via test-time dropout from the same model. Sai et al. (2020a) and Gupta et al. (2019) collect additional manually annotated responses for dialog contexts. Compare to them, our method of automatically collecting additional references automatically is more scalable.

Automatic data augmentation in NLP has largely been used for increasing training data (Feng et al., 2020; Wei and Zou, 2019; Feng et al., 2021). In this work, we use retrieved dialog instances and commonsense knowledge base to augment reference set for a given dialog context. Δ-Bleu (Galley et al., 2015) and uBLEU (Yuma et al., 2020) also use retrieval to produce pseudo-references for dialog response evaluation. Compared to Δ-Bleu and uBLEU, our work is different since we utilize commonsense knowledge base and perform contextual adaptation. Prior work in dialog response generation has explored the use of commonsense knowledge base (Majumder et al., 2020) as well as retrieval (Song et al., 2016; Majumder et al., 2021) – in contrast, our focus is on augmenting reference set for improving evaluation.

Automatic model-based metrics like ADEM (Lowe et al., 2017) and RUBER (Tao et al., 2017), which incorporate context while scoring for evaluation, at first glance seem to reduce the need for multiple references. However, these metrics have been found to suffer from several peculiar problems. For instance, ADEM can't discriminate between gold responses and certain classes of adversarial negatives e.g reversed gold responses and repeating the context as the response (Sai et al., 2019). Sato et al. (2020) evaluate dialog systems through their ability at selecting valid responses from a semi-automatically curated candidate list. Mehri and Eskenazi (2020b) introduce the unsupervised, reference-free USR metric, which leverages a suite of RoBERTa (Liu et al., 2019) models, each fine-tuned to score one of five dialog aspects e.g *Natural* and *Uses Knowledge*. Mehri and Eskenazi (2020a) further expand their USR metric to eighteen aspects from the initial five.

6 Conclusion

In this work, we demonstrate how existing knowledge sources can be used to construct a diverse set of references in an automated and scalable manner. The resulting reference set demonstrates high correlation with human ratings of system outputs.

In future, we plan to incorporate other commonsense types into SCARCE, such as social (Sap et al., 2019b) and moral (Forbes et al., 2020). We also hope to explore human-in-the-loop setups which build on SCARCE to collect even better references.

Acknowledgements

We thank anonymous ACL reviewers for insightful comments and feedback. We thank Prakhar Gupta (Gupta et al., 2019) for useful discussions.

Ethics Statement

Our preference ratings are collected over source content from an already existing, publicly available and widely used dataset i.e DailyDialog (Li et al., 2017) We neither solicit, record or request any kind of personal or identity information while collecting our ratings. Our work primarily performs experiments on dialog in English (Bender and Friedman, 2018). Dialog models are known to suffer from biases learnable from dialog training data, such as gender bias (Dinan et al., 2019). However, our work and contribution does not present or release any new models or model checkpoints, and is primarily focussed on making existing evaluation setups better through automated collection of larger reference sets.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Emily M Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. COMET: commonsense Transformers for Automatic Knowledge Graph Construction. In *ACL*.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2019. Queens are powerful too: Mitigating gender bias in dialogue generation. *arXiv preprint arXiv:1911.03842*.

Steven Y Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. Genaug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Marina Fomicheva, Lucia Specia, and Francisco Guzmán. 2020. Multi-hypothesis machine translation evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1218–1232.

Maxwell Forbes, Jena D Hwang, Vered Shwartz, Maarten Sap, and Yejin Choi. 2020. Social chemistry 101: Learning to reason about social and moral norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 653–670.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. $\triangle$-bleu: A discriminative metric for generation tasks with intrinsically diverse targets. In *ACL (2)*.

Prakhar Gupta, Shikib Mehri, Tiancheng Zhao, Amy Pavel, Maxine Eskenazi, and Jeffrey P Bigham. 2019. Investigating evaluation of open-domain dialogue systems with human generated multiple references. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 379–391.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 285–294. The Association for Computer Linguistics.

Bodhisattwa Prasad Majumder, Taylor Berg-Kirkpatrick, Julian McAuley, and Harsh Jhamtani. 2021. Unsupervised enrichment of persona-grounded dialog with background stories. In *ACL*.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shikib Mehri and Maxine Eskenazi. 2020a. Unsupervised evaluation of interactive dialog with dialogpt. *arXiv preprint arXiv:2006.12719*.

Shikib Mehri and Maxine Eskenazi. 2020b. Usr: An unsupervised and reference free evaluation metric for dialog generation. *arXiv preprint arXiv:2005.00456*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 794–805. Association for Computational Linguistics.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Ananya B Sai, Mithun Das Gupta, Mitesh M Khapra, and Mukundhan Srinivasan. 2019. Re-evaluating adem: A deeper look at scoring dialogue responses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6220–6227.

Ananya B Sai, Akash Kumar Mohankumar, Siddhartha Arora, and Mitesh M Khapra. 2020a. Improving dialog evaluation with a multi-reference adversarial dataset and large scale pretraining. *Transactions of the Association for Computational Linguistics*, 8:810–827.

Ananya B Sai, Akash Kumar Mohankumar, and Mitesh M Khapra. 2020b. A survey of evaluation metrics used for nlg systems. *arXiv preprint arXiv:2008.12009*.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019a. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *AAAI*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019b. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4453–4463.

Shiki Sato, Reina Akama, Hiroki Ouchi, Jun Suzuki, and Kentaro Inui. 2020. Evaluating dialogue generation systems via response selection. *arXiv preprint arXiv:2004.14302*.

Roger C Schank. 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive psychology*, 3(4):552–631.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

I. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Charles Spearman. 1961. The proof and measurement of association between two things.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Hiroaki Sugiyama, Toyomi Meguro, and Ryuichiro Higashinaka. 2019. Automatic evaluation of chat-oriented dialogue systems using large-scale multi-references. In *Advanced Social Interaction with Agents*, pages 15–25. Springer.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2017. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. *arXiv preprint arXiv:1701.03079*.

Oriol Vinyals and Quoc V. Le. 2015. A neural Conversational Model. *CoRR*, abs/1506.05869.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

John Wieting and Kevin Gimpel. 2017. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Tsuta Yuma, Naoki Yoshinaga, and Masashi Toyoda. 2020. ublеu: Uncertainty-aware automatic evaluation method for open-domain dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 199–206.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 654–664. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

A Additional Results

A.1 Additional Correlation Results

Table 4 shows Spearman rank correlation scores with p-values.

A.2 Quality Assessment based on RUBER

As a second, automated way of ascertaining response quality, we use the unreferenced part of the RUBER metric (Tao et al., 2017), which uses a pretrained model to score quality of responses based on context alone. Here, we use the RUBER checkpoint[3] from (Sai et al., 2020a), which first pretrains on a large Reddit dataset, followed by finetuning on DailyDialog. SINGLE and MULTI have a quality of $\approx$ 0.72, while for RETRIEVAL the values is 0.63 . COMMONSENSE is found to have the most superior quality at 0.82, surpassing even MULTI.

A.3 Diversity of References

We investigate the diversity of the references by computing self-BLEU scores (Zhu et al., 2018) among references from PARAPHRASE vs SCARCE. For fair comparison, we randomly chose 4 references from corresponding method. We observe self-BLEU4 scores of 0.36 for PARAPHRASE compared to only 0.13[4] for SCARCE.

B Additional Details on Context Adaptation

B.1 Templates to convert Knowledge Base Outputs to Full Sentences

Table 5 lists the set of templates and rules used to transform semi-structured COMET outputs to surface natural language forms.

B.2 Unsupervised Decoding Procedure For Context Adaptation

We use the author's own implementation[5] of their DELOREAN decoding algorithm from (Qin et al., 2020). We use default hyperparameters from their implementation, which use the non-finetuned *gpt2-medium* checkpoint as the LM atop which the unsupervised, gradient-based decoding procedure is run. Note that the model parameters are not updated in any way - the gradient computation and updates here are happening w.r.t the states, or more specifically, the state activation. More specifically, authors propose an iterature procedure wherein they alternatively perform forward and backward passes. In the forward pass, the current output Y is updated as per the likelihood of the underlying decoder. In the backward pass, the output is updated to be as similar as possible to the sentence Z from the knowledge source using back-propagation. However, since $Y$is discrete, we maintain a soft representation $\widetilde{Y}$ of the output Y wherein $\widetilde{Y_i}$ represents the logits at the i^{th} position as per the underlying decoder. Next, we shall describe the backward and forward passes of the iterative procedure:

1: In backward pass, we update logits based on the gradients of a content-matching loss function $\nabla_{\widetilde{Y}} L(\widetilde{Y}_{t-1}, Z)$ giving backward logits $\widetilde{y}_t^b$

2: Next, we perform forward pass using the underlying decoder for steps 1 to N. During forward pass at step t, we compute the logits $\widetilde{y}_t^f$ based on left context i.e. X and $Y_{<t}$. Next we perform weighted averaging of the forward and backward logits at step t to arrive at the final logits to be used for the next time step in forward pass.

$\widetilde{Y_i}$ is initialized by performing a forward pass conditioned only on X as per greedy decoding. We alternatively perform backward and forward passes till convergence. Final response is obtained via the resulting logit outputs $\widetilde{Y}$.

Specifically, we use their "counterfactual" setup, where an ending e_{old} is adapted from its old context c_{old} to an altered, new context c_{new}, generating a new, predicted ending $\hat{e}_{new}$. In our case, c_{new} is the dialog context for the turn under evaluation d_t^{past}. In the RETRIEVAL case, c_{old} is the context of the retrieved candidate turn $x_{t'}^{past}$. For the COMMONSENSE case, c_{old} is also our current context, i.e the same as c_{new} - we're simply attuning the already drawn inference better to the current context.

C Retrieval Similarity Function - Details

Consider a dialog d , broken up by turns as $\{C_1 \ldots C_t, C_{t+1}{=}d_t^{resp}, C_{t+2} \ldots C_T\}$, where $t+1$ denotes the turn currently under evaluation. For the context-response $C_t^1, \hat{r}_t$ pair to be evaluated, we retrieve pseudo-references based on a combination of of a) Past $d_t^{past} = C_t^{t-L_b}$ b) Gold response d_t^{resp} c) Future $d_t^{future} = C_{t+2+L_f}^{t+2}$. L_b and L_f are past and future context windows. Our retrieval similarity function is a sum of the log scores between each corresponding element of the turn under evaluation with the candidate turn.

---

[3] `tinyurl.com/ynqd54tt`

[4] Note that lower self-BLEU denotes more diverse

[5] `tinyurl.com/2lqp9z6s`

Spearman Rank Correlation (p-values)

Setup	1 human written reference			4 human written references		
Dataset	SINGLE (Li et al., 2017)	PARAPHRASE -SINGLE	SCARCE -SINGLE(Ours)	MULTI (Gupta et al., 2019)	PARAPHRASE -MULTI	SCARCE -MULTI(Ours)
BLEU-4	0.093 (0.04)	0.135 (0.00)	0.302 (0.00)	0.281 (0.00)	0.269 (0.00)	0.357 (0.00)
BLEU-3	0.055 (0.22)	0.105 (0.02)	0.291 (0.00)	0.243 (0.00)	0.238 (0.00)	0.345 (0.00)
BLEU-2	0.040 (0.37)	0.082 (0.07)	0.275 (0.00)	0.203 (0.00)	0.206 (0.00)	0.327 (0.00)
BLEU-1	0.024 (0.59)	0.062 (0.17)	0.250 (0.00)	0.191 (0.00)	0.178 (0.00)	0.295 (0.00)
ROUGE-L	0.071 (0.11)	0.088 (0.05)	0.259 (0.00)	0.197 (0.00)	0.196 (0.00)	0.317 (0.00)
METEOR	0.106 (0.02)	0.094 (0.04)	0.243 (0.00)	0.227 (0.00)	0.217 (0.00)	0.299 (0.00)
EmbeddingAvg	0.030 (0.50)	0.015 (0.73)	0.025 (0.58)	0.099 (0.03)	0.096 (0.03)	0.079 (0.08)
SkipThought	-0.003 (0.95)	-0.033 (0.46)	0.087 (0.05)	0.065 (0.15)	0.053 (0.24)	0.129 (0.00)
BERT-Prec	0.270 (0.00)	0.279 (0.00)	0.378 (0.00)	0.319 (0.00)	0.322 (0.00)	0.407 (0.00)
BERT-Rec	0.096 (0.03)	0.094 (0.04)	0.240 (0.00)	0.232 (0.00)	0.212 (0.00)	0.304 (0.00)
Max. value	0.270	0.279	0.378	0.319	0.322	0.407

Table 4: Utterance level Spearman Rank Correlation (Spearman, 1961) with p-values. (1) SCARCE-SINGLE augments the original single human written response (SINGLE) in DailyDialog dataset (Li et al., 2017) using proposed method. It leads to large improvements in correlations across most of the metrics, when compared to SINGLE. (2) SCARCE-MULTI augments the MULTI dataset, again leading to improvements in correlations to human ratings, especially for BLEU and BERT-Prec metrics. Additionally, we note that almost all of the correlation values with SCARCE-MULTI are statistically significant with $p < 0.05$.

Condition	Action
Type is OEFFECT Example: OEFFECT (excited)	Prepend 'I feel' => 'I feel excited.'
Type is OWANT Example: OWANT (to thank personx)	Prepend 'I' => 'I want to thank personx.'
Type is OREACT Example: OREACT (have a party)	Prepend 'I will' => 'I will have a party.'
Word PERSONX Example: i thank PERSONX.	Replace with 'you' => 'I thank *you*.'

Table 5: Templates and rules to transform semi-structured COMET outputs to surface NL forms.

$$Sim(d_t, x_{t'}) = \log S_{bm25}(d_t^{past}, x_{t'}^{past}) + \log S_{bm25}(d_t^{resp}, x_{t'}^{resp}) \\ + \log S_{bm25}(d_t^{future}, x_{t'}^{future})$$

We set $L_b = L_f = 2$ without specific tuning, as an intuitive tradeoff between enough specificity and enough possibility of relevant candidates.

BM25 (Robertson et al., 1995) or "Best Match 25" is a tfidf like similarity. Its specific form is:

$$S_{BM25}(q, d) = \sum_{w_i \in q} \log(\frac{N}{df_i}) \frac{(k_1 + 1)tf_i}{k_1((1 - b) + b\frac{dl}{avdl}) + tf_i}$$

Here, tf_i and df_i are the term frequency in the current document and the document frequency (in the corpus). N is corpus size, while dl and $avdl$ are current and average document lengths. b controls extent of document length normalization, while k_1 controls effect of term frequency. With $b = 0$ and $k_1 \to \infty$, this reduces to simple tfidf . Here, we use default gensim values, $b = 0.7$, $k_1 = 0.5$

D Qualitative Examples

In Tables 6, we list some examples, each illustrating a turn of a test dialog with its immediate past, future, the four additional human references from (Gupta et al., 2019) (shown under MULTI 2,3 and MULTI 4,5), followed by automated response sets from different sub-components of SCARCE.

D.1 Before/After CONTEXTADAPT

In Example *4-4* of Table 6, we can observe how *"Yes , I'm young , and unmarried . It's no problem for me to travel frequently ."* gets context-adapted (shown as +CA, short for CONTEXTADAPT) to *"Yes , I'm able to understand English. It 's not that I don't understand English ."* which indeed does match the preceding dialog better. Similarly, in Example *50-2* of Table 6, we can see how *"Well, that might be acceptable if you handle insurance fees"* is modulated stronger to the context which asks about duration, getting adapted to *"Well, that's a lot of time to wait for the draft to be drawn."*. Note that we omit this row for the examples where it simply leaves the input unchanged, or produces outputs which are noticeably unfaithful or ill-formed.

D.2 Complementarity of Components

Sometimes, a component may suffer from example specific issues e.g In Example *35-2*, the COMMONSENSE approaches misinterpret what is a driving ticket in the context of the dialog as an event ticket, drawing inferences accordingly. However, even in such cases, the other component salvages the situation and ensures overall response set remains healthy - e.g, here, RETRIEVAL produces pertinent responses like *Could you tell me how you dealt with it this time?*, *No I haven't. What about you ?* etc. In Example *10-3*, we see the opposite situation, where the responses from RETRIEVAL are somewhat less relevant, but COMMONSENSE produces very pertinent responses such as *"i decline the date"* and *"I go on another date"*

E Human Evaluation Details

E.1 Quality of References

The quality of references were judged by two graduate students from a university where the medium of instruction is English. The annotators were requested to ignore minor grammar issues, and focus more on the content of the response.

F Computing Details

The GPUs used for COMMONSENSE and CONTEXTADAPT experiments were a Geforce Rtx 2080 and TitanX Pascal respectively.

No	Type	Text
0-5	Context	A: I also have blow if you prefer to do a few lines.
	Future	A: come on man ! i even got dope and acid ! try some !.
	Single	B: No, I am ok, really.
	Multi 2,3	B: what is your price ? ; B: no thanks, I’m out of cash
	Multi 4,5	B: yeah , what have you go got ?; B: maybe tomorrow , i ’m running late .
	Retrieval	B: Well , I guess Zina figured you couldn’t blow it if you were in the hotel room .
	Retrieval w CA	B: I have a few lines of blow , but it is not necessary in the way of the rest of the script .
	Commonsense	B: i want to go to bathroom; B: i want to play game; B: i want to have rest; B: i want to sleep
1-1	Context	B: The taxi drivers are on strike again A: What for?
	Future	A: it is really a hot potato .
	Single	B: They want the government to reduce the price of gasoline .
	Multi 2,3	B: they want more pay for their time .; B: they are not happy with their permit related expenses .
	Multi 4,5	B: they want healthcare and benefits .; B: i don’t know . i just know there are no cabs in the city
	Retrieval	B: Well . Don’t worry about it , I will be happy to share the taxi with you .
		B: I think the price is a little high , can’t you reduce it?
	Retrieval w CA	B: Well , they are not going to be able to get their fares back .
		B: I think the reason is a little more complicated than that .
4-4	Context	A: Do you think you can make yourself easily understood in English ?
	Single	B: yes , in most circumstances .
	Multi 2,3	B: my english is fine . my keyboard sometimes just acts up . ; B: yes, and i’m constantly trying to improve my communication.
	Multi 4,5	B: i do n’t know . i may be to nervous and then i ’ll mess up . ; B: yes , i believe so . i ’ve been speaking english since birth .
	Retrieval	B: Yes , I’m young , and unmarried . It’s no problem for me to travel frequently .
	Retrieval w CA	B: Yes , I’m able to understand English . It ’s not that I don’t understand English .
10-3	Context	B: Hold on , please . Let me check it for you . Yes , you’re right . You will keep it for 3 days .
		A: Well , now I want to change the date from 24th to 28th .
	Single	B: ok , that shall be arranged .
	Multi 2,3	B: we can easily do that for you . ; B: i ’m sorry but someone’s reserved the room for 28th . i can offer you a different room ?
	Multi 4,5	B: i can extend your stay time but it may cost more since it is going into the holiday . ;
		B: i can give you a different double room for the 28th at a discounted price .
	Retrieval	B: All right . May I have your name ? ; B: Apparently there is only $ 57 left in your account . ;
		B: Here you are . What is the rate ?; B: I prefer not to move first .
	Commonsense	B: i decline the date ; B: i go on another date; B: i go on a date;
		B: i get nervous; B: i get rejected; B: i get dumped; B: i feel sad
23-10	Context	B: I mainly eat baked chicken , because there’s not a lot of fat . A: that does sound pretty good .
	Single	B: i know it does , and that ’s because it is .
	Multi 2,3	B: yes , it is very good and pretty healthy .; B: it is good , you should come over and i will make you dinner .
	Multi 4,5	B: thanks , it ’s been a struggle , but i am starting to see results . ;
		B: i appreciate your kind words , i am getting used to eating more healthy .
	Retrieval	B: You should try it . You won’t be disappointed .; B: Do you have any plans for tomorrow ? ; B: That’s it ? ;
		B: Just about . ; B: I like fish . Please bring me an order of that .
	Commonsense	B: i want to cook dinner; B: i want to go to movie; B: i want to write letter ;
		B: i will open your mouth; B: i will think about what you want; B: i will drink some water;
35-2	Context	B: I’ve never gotten one . Have you ? A: I got a ticket yesterday .
	Future	A: i parked in the red zone .
	Single	B: really ? what did you get one for ?
	Multi 2,3	B: what was it for ? ; B: how much is it ?
	Multi 4,5	B: do you resent you got a ticket ? ; B: are you going to appeal the ticket ?
	Retrieval	B: I’ve gotten a few .; B: No , I haven’t . What about you ? ;
		B: Could you tell me how you dealt with it this time ? ;
		B: I will explain it in detail in the next class , could you preview it ?
	Commonsense	B: i give ticket to you; B: i give ticket to the cashier; B: i give ticket to the clerk.;
		B: i feel happy .; B: I feel excited .; B: i feel annoyed . ; B: I feel disappointed .;
		B: i see what else they can do .; B: i see what else they can do to get the ticket; i go to the event

Table 6: Example context-response pairs from the test split of DailyDialog, showing the automated responses returned by different sub-components of Scarce. ContextAdapt is shortened to CA for brevity.

Who Blames or Endorses Whom?
Entity-to-Entity Directed Sentiment Extraction in News Text

Kunwoo Park
Soongsil University[*]
`kunwoo.park@ssu.ac.kr`

Zhufeng Pan
UCLA
`panzhufeng@cs.ucla.edu`

Jungseock Joo
UCLA
`jjoo@comm.ucla.edu`

Abstract

Understanding who blames or supports whom in news text is a critical research question in computational social science. Traditional methods and datasets for sentiment analysis are, however, not suitable for the domain of political text as they do not consider the direction of sentiments expressed between entities. In this paper, we propose a novel NLP task of identifying directed sentiment relationship between political entities from a given news document, which we call *directed sentiment extraction*. From a million-scale news corpus, we construct a dataset of news sentences where sentiment relations of political entities are manually annotated. We present a simple but effective approach for utilizing a pretrained transformer, which infers the target class by predicting multiple question-answering tasks and combining the outcomes. We demonstrate the utility of our proposed method for social science research questions by analyzing positive and negative opinions between political entities in two major events: 2016 U.S. presidential election and COVID-19. The newly proposed problem, data, and method will facilitate future studies on interdisciplinary NLP methods and applications.[1]

1 Introduction

Sentiment analysis is a useful technique for opinion mining from text data. Most existing work either focuses on sentence-level classification (Van Hee et al., 2018; Zampieri et al., 2019), or aims to detect the sentiment polarity towards specific targets (Pontiki et al., 2016; Cortis et al., 2017). These approaches typically do not distinguish sources and targets of the sentiment. They mainly use user-generated content (UGC), such as tweets or restaurant reviews from Yelp, which assumes each user (account holder) is the source of the sentiment, and that the target is also clearly defined or easily identifiable (e.g., the restaurant).

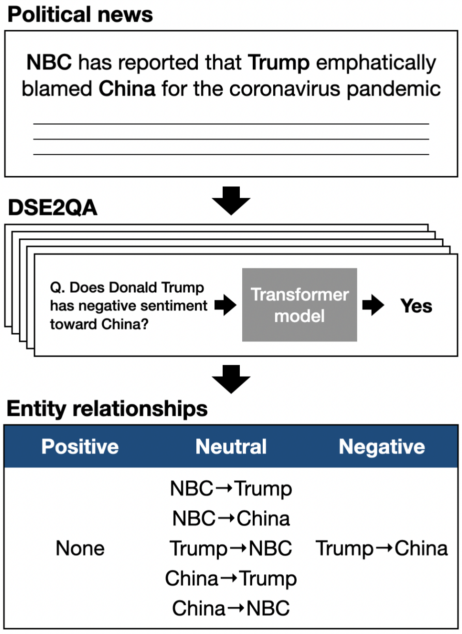

Figure 1: Overview of the directed sentiment extraction

This assumption does not hold for political news analysis where a large number of political actors blame or support each other in news articles. The key interest for political sentiment analysis is to identify "who" blame or support "whom" (Balahur and Steinberger, 2009) rather than simply assigning a global sentiment polarity to a given document or sentence. For example, from a sentence like "X supported Y for criticizing Z," we can infer X is positive toward Y and both X and Y are negative toward Z. However, existing sentiment analysis methods are not suitable to detect such sentiment relationships between entities.

This paper proposes a new NLP task of identifying directed sentiment relationships from a political entity to another in news articles: *directed sentiment extraction*. For this task, we introduce a newly annotated corpus from a million-scale news

---

[*] This work was done while the first author was a postdoctoral researcher at UCLA.

[1] `https://github.com/bywords/directed_sentiment_analysis`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4091–4102
August 1–6, 2021. ©2021 Association for Computational Linguistics

dataset. As demonstrated in Figure 1, we transform directed sentiment extraction to multiple question-answering tasks (**DSE2QA**) and combine their predictions for making a final prediction. Evaluation results show it outperforms state-of-the-art classification approaches, such as a fine-tuned RoBERT classification model. Going further, we demonstrate the utility of our method through two case studies on how news media in the U.S. portrayed relationships between political entities differently amid the US election and COVID-19 pandemic. The analysis reveals that the left-leaning media present Donald Trump more as the target of blame while the right-leaning media present news stories blaming other entities. This study not only makes a contribution to the NLP community by defining a new problem and approach but also adds the empirical understanding of media bias to the social science community.

2 Related Work

2.1 Sentiment Analysis in Media

Sentiment analysis or polarity detection aims at deciding whether a given text contains positive or negative sentiment (Liu, 2012) or quantifying the degree of sentiments embedded in a text (Gilbert and Hutto, 2014). Previous studies have tackled the problem as a classification task (Maas et al., 2011) and applied the trained model to infer sentiments embedded in various web and social data (Park et al., 2018). Recently, transformer-based models have shown high performance in sentiment classification (Devlin et al., 2019) and aspect-based sentiment analysis (Sun et al., 2019).

Measuring sentiment or tone of political text in news media is a widely used method in computational social science (Young and Soroka, 2012). A stream of work has used social media posts to estimate public opinions about political actors and predict the outcomes of future events by large scale sentiment analysis (O'Connor et al., 2010; Ceron et al., 2014; Wang et al., 2012), while some studies further extend to nonverbal or multimodal dimensions (Joo et al., 2014; Won et al., 2017; Chen et al., 2020a).

2.2 Stance Detection

Stance detection is a relevant NLP problem that aims to predict a claim's stance on reference text (Augenstein et al., 2016) or to infer social media users' view toward an issue (Darwish et al., 2020). Many studies tried to advance the deep learning-based models (Mohtarami et al., 2018), for example, by modeling text with a hierarchical architecture (Sun et al., 2018; Yoon et al., 2019). Unlike stance detection, this study aims at understanding an entity's sentiment toward another entity, both of which appear in the same sentence.

2.3 Relation Extraction

Our target problem is also relevant to relation extraction, which is a task of extracting structured relationships between entities from unstructured text. While the early literature relied on feature-based methods (Zelenko et al., 2003), recent methods actively utilize neural methods (Lin et al., 2016); for example, a study proposed a neural model that jointly learns to perform entity recognition and relation extraction (Bekoulis et al., 2018). Most recently, a study tested the use of the pretrained transformer-based language model for relation extraction (Zhang et al., 2020).

Despite its similarity to directed sentiment extraction, most of the existing datasets only consider *explicit* entity relationships such as EMPLOYEE_OF and CITY_OF_RESIDENCE (Zhang et al., 2017). Understanding sentiment relationships between political entities is more challenging as their sentiment is usually hidden in text.

3 Problem and Dataset

In this section, we introduce our problem formulation and explain the process of our dataset construction and annotation.

3.1 Target Problem

Given a sentence s that contains two entities p and q, the *directed sentiment extraction* problem aims to detect the sentiment relation from p to q among five classes: neutral, p holds a positive or negative opinion towards q, and the reverse direction. For example, in the given sentence in Figure 1, the model should infer that Trump is the source of the negative sentiment toward China, the target. Existing approaches for sentiment analysis cannot be easily adapted to the task, as existing methods aim to detect polarity embedded in a text (sentiment classification), for a specific target (targeted), or with regard to an aspect (aspect-based). These problem setups do not consider the source and target of the sentiment at a time, which cannot identify directed sentiment relationships between political entities.

Class	Count
Neutral	10,604
Positive ($p \rightarrow q$)	1,656
Positive ($p \leftarrow q$)	327
Negative ($p \rightarrow q$)	3,163
Negative ($p \leftarrow q$)	478
Total	16,228

Table 1: Dataset statistics

In the following, we introduce a new annotated corpus for the problem.

3.2 Data Collection

To construct our dataset, we used news articles from the Real News corpus (Zellers et al., 2019), which consists of 32,797,763 real news stories in English published by various outlets over multiple years. Among them, we used 7,127,692 news articles shared by news media that are verified as trustworthy by Media Bias/Fact Check (Media Bias Fact Check, 2015). After splitting each article into multiple sentences, we only took sentences with two or more entities using the named entity recognition tool in Spacy[2]. To focus on the relationships between political entities, we considered named entities identified as people, countries/political groups, organizations, or cities/states[3].

Since most entity relationships in regular sentences are presumably neutral, we took two approaches for sampling sentences that cover diverse relationships: (i) dictionary-based and (ii) random selection approaches. The dictionary-based approach filters in sentences containing positive or negative keywords from a pre-defined dictionary. Starting from the blame-related keywords (Liang et al., 2019), we extended the dictionary by adding their synonyms and antonyms. While this method is effective in sampling from an unbalanced dataset, it excludes sentences that do not explicitly mention a blame or support keyword. Thus, we also randomly drew sentences to improve the dataset coverage.

3.3 Crowdsourced Annotation

We used Amazon Mechanical Turk (AMT) to annotate each sentence. The annotation task asked workers to identify what sentiment does an entity holds toward another in a given sentence. We instructed them to annotate a sentence based only on the sentiment expressed within the sentence, without relying on prior knowledge. There are five options to choose from, neutral, positive ($p \rightarrow q$), positive ($q \rightarrow p$), negative ($p \rightarrow q$), and negative ($q \rightarrow p$). p is the preceding entity of q, and the arrow indicates sentiment direction between the two entities. The detailed instruction used for educating annotator is presented in Table A1 in Appendix.

We hired five workers to annotate each sentence. For ensuring high-quality responses, we only allowed workers to participate in the task when they had at least a 70% acceptance rate for more than 1,000 previous annotations. After completing the initial round of annotation, we filtered out unreliable responses completed within one second or responses by workers who did not pass test questions. We designated workers with at least one unreliable response as untrustworthy and discarded all the other responses submitted by the untrustworthy workers. We repeated the annotation task for the discarded answers until we have five annotations for every sentence. The final set of annotations indicates a Fleiss' kappa value of 0.26, indicating an acceptable level of agreement among annotators. The level of reliability is comparable to studies using subjective text annotations such as hate speech annotation (Ross et al., 2017), subjectivity (Abdul-Mageed and Diab, 2011), and sentiment analysis (Park et al., 2018). By aggregating five responses for each sentence by a majority vote, we obtained the final dataset of 16,228 sentences of which sentiment direction is annotated, as shown in Table 1. While keeping the label distribution almost identical, we split the dataset into 13144, 1461, and 1623 instances for train, validation, and test set, respectively.

4 Methods

This section presents methods for addressing the problem of directed sentiment extraction. We propose a novel approach for solving it by employing augmented inputs in BERT-like transformer models and compare it with classification approaches.

4.1 Classification Approaches

Following the standard in using classification setups for (undirected) sentiment detection (Liu, 2012; Devlin et al., 2019), we construct classification models that predict scores of each sentiment for the directed sentiment extraction.

---

[2] https://spacy.io

[3] https://spacy.io/api/annotation#named-entities

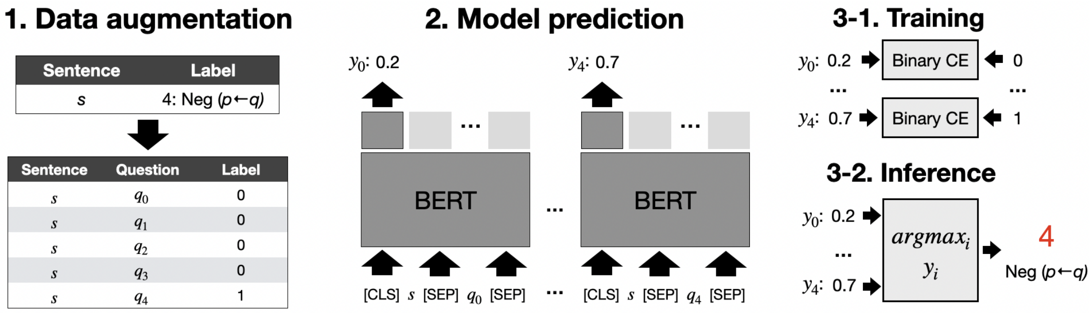

Figure 2: Detailed illustration of the DSE2QA approach

4.1.1 Existing Methods

We first construct previous approaches proposed for blame relationship detection. Liang et al. (2019) proposed three neural models that predict the classes of blame relationships in a given text: $p \rightarrow q$, $p \leftarrow q$, and no relationship. We extend their approaches to the five classes of directed sentiment in our dataset.

Entity prior model exploits the prior knowledge on political entities by using the representation of two entities using a pre-trained word embedding: $\mathbf{e}_p$ and $\mathbf{e}_q$. After concatenation, the two vectors are fed into a fully-connected network with a ReLU hidden layer to make a final prediction. *Context* model utilizes the context of the target sentence where two entities appear. After replacing p and q with special tokens ([ENT1] and [ENT2]) respectively, the model encodes the input text through a bidirectional LSTM and extracts the representation corresponding to the two entities: $\mathbf{lstm}_p$ and $\mathbf{lstm}_q$. The representation gets fed into a fully-connected network. *Combined* model utilizes the concatenated representation of the outputs of the entity prior and context model: $\mathbf{e}_p$, $\mathbf{e}_q$, $\mathbf{lstm}_p$, and $\mathbf{lstm}_q$. Then, the vector is fed into a fully-connected network. We train the existing models by minimizing the cross-entropy loss.

4.1.2 Fine-Tuning a Pretrained Transformer

Fine-tuning a BERT-like pretrained transformer has shown significant performance in many downstream tasks (Devlin et al., 2019). We also evaluate the performance of a fine-tuned transformer. In particular, after replacing the tokens corresponding to p and q with [ENT1] and [ENT2], we train a classification model that predicts the five-class output based on the representation of [CLS][4]. We use the RoBERTa base model (Liu et al., 2019) as backbone, and we refer to the classification model as *RoBERTa* in evaluation experiments. The model is trained to minimize the cross-entropy loss.

4.2 Proposed Approach: Directed Sentiment Extraction to Question-Answering

BERT-like transformer models are pretrained using two inputs including a separator token[5] with varying training objectives. The input configuration allows the model to be successfully transferred to the tasks using an auxiliary input, such as sentence pair classification (sentence 1 and sentence 2) and question answering (reference and question). Inspired by the recent achievements using auxiliary inputs (Sun et al., 2019; Cohen et al., 2020), we propose a simple but effective approach of tackling the directed sentiment extraction problem, which we call **DSE2QA**; we transform the sentiment extraction task into the sub-tasks aiming for answering yes/no questions on whether a target sentiment is embedded in the text. The basic idea is we inquire an intelligent machine who can answer yes/no questions on whether a target sentiment exists and then combine the answers corresponding to the each sentiment class for making a final guess. Figure 2 presents the overall framework, which we elaborate on each step in the following. Technically, taking auxiliary input in BERT-like transformers enables implementing the intelligent machine by making a different prediction with the same sentence input, according to the question fed as additional input. We hypothesize that a large-scale pretrained transformer model on a massive corpus can understand the meaning of the augmented question and thus successfully answer whether a directed sentiment exists in a text.

Note that our question-answering setup is different from standard question-answering tasks in NLP, as represented by well-known benchmark data such as SQuAD (Rajpurkar et al., 2016) and

---

[4] <S> in RoBERTa

[5] [CLS] in BERT, </S></S> in RoBERTa

Index	Complete questions	Pseudo questions
q_0	Do [Ent1] and [Ent2] have neutral sentiment toward each other?	[Ent1] - [Ent2] - neutral
q_1	Does [Ent1] has positive sentiment toward [Ent2]?	[Ent1] - [Ent2] - positive
q_2	Does [Ent2] has positive sentiment toward [Ent1]?	[Ent2] - [Ent1] - positive
q_3	Does [Ent1] has negative sentiment toward [Ent2]?	[Ent1] - [Ent2] - negative
q_4	Does [Ent2] has negative sentiment toward [Ent1]?	[Ent2] - [Ent1] - negative

Table 2: Auxiliary questions according to the target label

WikiQA (Yang et al., 2015). Given a question and reference text, the standard task aims at generating answers in a natural language form. In contrast, the question-answering process in DSE2QA requires a binary answer, which can be seen as a special type of question-answering.

4.2.1 Data Augmentation for DSE2QA

For each pair of label l and sentence s where the two target entities p and q are masked as [ENT1] and [ENT2] respectively, we augment the training data by transforming the original input into the five tuples using the same sentence and different questions: t_i: (s, q_i, l_i) where i is the index of the target relation class: neutral (0), $p \rightarrow q$ with positive sentiment (1), $p \leftarrow q$ with positive sentiment (2), $p \rightarrow q$ with negative sentiment (3), and $p \leftarrow q$ with negative sentiment (4). l_i becomes 1 if l is i; otherwise l_i is 0.

We design the auxiliary question q_i asking whether the given sentence s is classified as the target sentiment i. The list of questions are presented in Table 2. For example, q_1 asks a model whether a given sentence s contains positive sentiment from p to q. Here, we define two types of questions: complete and pseudo. Complete questions are written in a natural language, and pseudo questions only contain keywords that is sufficient to characterize a sentiment class i while ignoring the syntactic structure.

4.2.2 Model Prediction

We utilize the BERT-like transformer model (Devlin et al., 2019), which can take sentence pairs as input, for making a binary prediction on a given sentence s and question q_i. In particular, the model takes

$$[\text{CLS}]\ s\ [\text{SEP}]\ q_i\ [\text{SEP}]$$

as input[6] and predicts a value y_i from 0 to 1 that indicates the confidence on the target label i.

4.2.3 Training and Inference

For the augmented input of t_i, a pretrained BERT-like transformer is trained to predict 1 for t_l and 0 for $t_{i \neq l}$ through the [CLS] representation at the last layer of the transformer model followed by a classification layer. For inference, we made a prediction corresponding to s by $\text{argmax}_i\, y_i$ where y_i is the prediction outcome of t_i. The y_i indicates the confidence on each sentiment i, and therefore we take the class of which the value is maximum.

In the experiments, we utilize the RoBERTa base model for the backbone of DSE2QA and train the model to minimize the binary cross-entropy loss. This approach is different from the RoBERTa classification model that only employs a single sentence input.

5 Performance Evaluation

We evaluate the performance of the proposed DSE2QA approach using our annotated corpus. We compare our method with the current state-of-the-art methods for directional blame detection proposed by Liang et al. (2019) (LNZ) as well as a classification model fine-tuned on a pretrained transformer (RoBERTa).

5.1 Experiment Setups

For the LNZ models (Liang et al., 2019), we set the vocabulary size as 10000. We set the word embedding size, LSTM hidden dimension, and the fully connected layer dimension as 256, 512, and 1024. The search space for the dropout rate is $[0.1, 0.5]$. We train the LNZ models using Adam optimizer with a learning rate of 1e-3 (Kingma and Ba, 2014). We adopt an early stopping strategy with the patience of 5. For training RoBERTa and DSE2QA, we followed the procedure of Liu et al. (2019) using AdamW (Loshchilov and Hutter, 2017). We optimize hyperparameters by randomly choosing ten sets for each model and selecting the model with the best performance on the validation set. The learning rate is set to 2e-5 with the epsilon as

---

[6] '<s> s </s> </s> q_i </s> </s>' in RoBERTa

Method	Micro F1	Macro F1	mAP
DSE2QA (Pseudo)	**0.7973**	**0.6766**	**0.7488**
DSE2QA (Complete)	0.7726	0.6617	0.7387
RoBERTa	0.7486	0.6409	0.7319
LNZ (Combined)	0.7055	0.5358	0.5295
LNZ (Context)	0.6371	0.4665	0.4921
LNZ (EntityPrior)	0.5853	0.4063	0.414

Table 3: Evaluated performance on the test set. Top performance for each metric is marked as bold.

Method	0	1	2	3	4
DSE2QA (Pseudo)	**0.855**	**0.6519**	**0.5672**	0.7402	**0.5686**
DSE2QA (Complete)	0.8293	0.6421	0.5672	**0.7416**	0.5283
RoBERTa	0.8054	0.6373	0.5079	0.7184	0.5354
LNZ (Combined)	0.7981	0.443	0.3333	0.5827	0.5217
LNZ (Context)	0.7469	0.4069	0.2817	0.5007	0.3964
LNZ (EntityPrior)	0.7133	0.2629	0.2353	0.4533	0.3667

Table 4: F1-score per class measured on the test set. Top performance for each metric is marked as bold.

1e-6. The weight decay is set to 0.1. We apply random oversampling to the training set to make a balanced dataset against the label. We run the experiment five times with different random seeds and report the average scores.

5.2 Evaluation Results

We utilize three measures for evaluation: micro-f1, macro-f1, and mean average precision (mAP). Micro-f1 is calculated by (#correct)/(#total), which corresponds to the multi-class classification accuracy. Macro-f1 measures an f1-score for each class and averages them with equal importance; therefore, macro-f1 is a more robust measure to a skewed class distribution, such as our annotation data (see Table 1). Similarly, mAP measures the unweighted average of average precision (AP) on each class; AP summarizes a precision-recall curve varying prediction threshold for a target class.

In Table 3, we make three observations. First, among classification approaches (the bottom four rows), RoBERTa outperforms the other approaches across the three measures (0.7486/0.6409/0.7319). The LNZ combined model achieves a fair micro-f1 score of 0.7055 but low scores of macro-f1 (0.5358) and mAP(0.5295). This difference is because the combined model (and other non-transformer models) is poor at classifying non-neutral sentiment, which we will further investigate in Table 4. Second, DSE2QA with complete questions outperforms RoBERTa with a margin of 0.024 by micro F1. The proposed approach also achieves better performance in macro F1 and mAP. Third, the performance of DSE2QA gets further increased with the usage of pseudo questions, up to the micro-f1 score of 0.7973. This observation implies that a BERT-like transformer model may not need a full sentence to utilize the auxiliary input because it also performs well using fewer keywords for the detection task with an augmented input.

Table 4 presents the f1-score measured for each class: neutral (0), positive from the left entity to the right (1), positive from the right to the left (2), negative from the left to the right (3), and negative from the right to the left (4). Here, we make three observations. First, all models perform the best at identifying neutral sentiment (0) compared to the other sentiment classes. Second, non-transformer models (the bottom three rows) are poor at extracting non-neutral sentiment regardless of their direction, which contributes to the decreased macro F1 in Table 3. Third, among the sentiment classes from the left entity to the right entity (1, 3), transformer models better detect negative sentiment than positive sentiment. The finding suggests that positive entity relationships are more difficult to be captured in news articles, which calls for future studies for a better understanding and model improvement. AP per each class also shows a similar trend, as presented in Table A3.

In summary, the proposed approach of solving the directed sentiment extraction task by multiple question-answering tasks outperforms the state-of-the-art classification approaches. The high performance suggests that the model can understand the meaning of augmented questions to some extent, which may build on the language understanding ability of the pretrained RoBERTa.

6 Analyzing Entity Relationships in News Articles

To demonstrate the utility of the proposed dataset and model, we conduct two case studies to analyze entity-to-entity sentiment relationships presented in recent news articles on political issues: the 2016 U.S. presidential election and the COVID-19 pandemic. For the analyses, we utilize the DSE2QA model with pseudo questions trained on the annotated corpus, and we confirm that the target news articles are not overlapped with the whole set.

6.1 Case Study 1: 2016 U.S. election

We study how news media covered the entity relation during the 2016 United States presidential election using a public dataset on news articles between Feb. 2016 to Feb. 2017[7]. This dataset consists of about 140K news articles in English from fifteen media companies, including CNN, New York Times, and Guardian. We randomly select 3K articles from each month, 39K articles in total. Then we apply the proposed model to all sentences that contain at least two entities from the top-30 most frequent entities, including Donald Trump and Hillary Clinton.

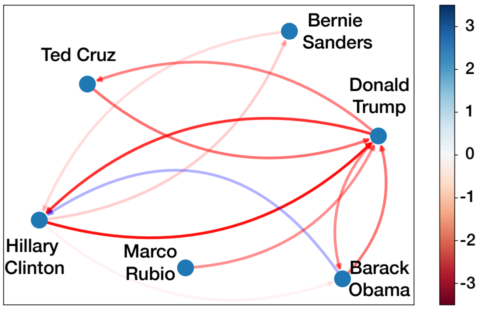

Figure 3: Log ratios of positive (blue) and negative (red) sentiments in directed political relationships in the news articles.

Figure 3 presents the frequently mentioned pairs of politicians. The red color indicates an entity pair tends to contain negative sentiment more than positive, and blue indicates the pairs with the negative sentiment more. The opacity represents the strength of the opinion measured by the log ratio of inferred pairs of positive and negative sentiment. For brevity, we present relations that appear at least 20 times in any sentiment. Overall, there are 4.68 times more negative sentiments found than positive ones, which may be explained due to the negative nature of political media (Soroka, 2014). An interesting observation is the asymmetric relation between Hillary Clinton and Barack Obama. Clinton holds a generally slightly negative opinion towards Obama, while Obama holds a stronger positive opinion towards Clinton. Note that our model does not just memorize the entity relationship in the training dataset and apply it to the target dataset as we replace detected entities with tokens in input sentences.

Left	Center	Right
ABC News	Associated Press	Breitbart News
BuzzFeed News	Forbes	Daily Mail
CBS News	NPR	Fox News
CNN	Reuter	National Review
Democracy Now	USA TODAY	New York Post
HuffPost		Reason
MSNBC		The American Spectator
New York Times		theblaze.com
Slate		The Daily Caller
The Atlantic		The Daily Wire
The Guardian		The Epoch Times
The New Yorker		The Federalist
Time Magazine		Washington Times
Vox		Wall Street Journal
Washington Post		

Table 5: The list target media outlets sorted by alphabetical order.

6.2 Case Study 2: the COVID-19 pandemic

In the second study, we investigate how news media portray political entities and their relationships differently according to their political orientations. For example, a right-leaning news outlet may show more negative opinions expressed toward Democrats. To that end, we focused on the recent issue of the COVID-19 pandemic to examine the media bias. We expect the general sentiment about COVID-19 is negative but would like to investigate who blames whom because the messages will have very different meanings according to the sources and targets, differentiated by our data and method.

To collect a recent news article set, we first compiled a list of 35 popular news media outlets that cover American politics in English. We also ensure the list of news outlets was balanced against political bias, according to the media bias ratings in `allsides.com`. For brevity, we consolidate 'Lean

---

[7] https://www.kaggle.com/snapcrack/all-the-news

Entity	Rank (Left)	Rank (Center)	Rank (Right)
Donald Trump	1	2	1
Republican	2	6	6
U.S.	3	1	3
Democrat	4	5	4
Russia	5	6	9
American	6	15	10
China	7	4	5
Obama	8	10	7
Chinese	9	3	2
Joe Biden	9	8	7

Table 6: Frequency rank of top-10 frequent entities targeted with a negative sentiment through an entity-to-entity relationship in the COVID-19 news dataset. The order is sorted by the overall rank in the dataset.

left' and 'Left' into 'Left' and 'Lean right' and 'Right' into 'Right.' Table 5 presents the list of the target media.

For each of the target media outlets, we collected news articles shared throughout 2020 until September from the Common Crawl corpus, which has been collecting web data since 2008[8]. The total number of retrieved news pages are is 256,081; on average, we have 7,113 published news articles for each outlet.

Next, we selected documents containing at least one of the keywords relevant to COVID-19 by following similar practices used for collecting a Twitter dataset (Chen et al., 2020b): *coronavirus*, *covid-19*, *COVID19*, and *corona virus*. We consider sentences containing two or more entities for the target of inference, and every relationship pair is inferred when there are more than two entities in a sentence. The final set consists of 6,180 articles involving 1,078,377 entity pairs for COVID-19.

Table 6 presents the list of 10 frequent entities that are manifested through entity-to-entity relationships with a negative sentiment. While Donald Trump was the most frequent target of blames in the total data, the results show that the right-leaning media tend to express a negative sentiment toward China (#5) and Chinese people (#2) more frequently. To examine the difference systematically, we measure the spearman rank correlation coefficient for the whole list of entities that appeared in the dataset. The rank in the left-leaning media and that in right-leaning media exhibits a highly negative correlation of -0.5722 (p<0.001), which suggests that the list of political entities presented with a negative sentiment significantly varies across news media according to their political orientation. Such a high level of negative correlation is also observed in the ranks for the source entity in negative relationships (-0.4129 with p<0.001) and the source/target entities in positive relationships (-0.5605/-0.7929 with p<0.001).

Going further, we analyze the differences between frequently presented entity pairs with negative sentiment by the left and right-leaning media, respectively. Table 7 presents the rank of each entity pair in the media groups according to their political orientation. In the left-leaning media of our dataset, Donald Trump appears as either source or target in the top-10 frequent negative pairs except for the pair of Republican→Democrat and vice versa. On the contrary, the top-10 pairs in the right-leaning media include the international relationships of Donald Trump to the other countries. In other words, the left-leaning media may try to frame COVID-19 as a **domestic event** focusing on how the President handles it and how people respond to his crisis management, and the right-leaning media focus more on **foreign policies and international relationship** especially between the U.S. and China. For the whole set of entity pairs, the rank correlation between the left and right media is -0.4847 (p<0.001) for negative sentiment and -0.7929 (p<0.001) for positive sentiment. These negative correlations imply that the news media has a bias in selecting issues to cover (selection bias) and presenting relationships of political entities (presentation bias).

7 Conclusion

Detecting who blames or endorses whom in news articles is a critical ability in understanding opinions and relationship between political actors in news media. This paper provides a computational tool based on natural language processing for facilitating interdisciplinary studies using text in news and social media.

We introduced a new problem of identifying directed sentiment relationships between political entities, called directed sentiment extraction. We constructed a training corpus of which entity relationship is manually annotated for each sentence. This dataset can serve as a benchmark for future studies. A potential future direction is to build an improved version of the dataset where sentiment re-

---

[8] https://commoncrawl.org/

Frequent pairs in the left-leaning media

Entity pairs	Rank (Left)	Rank (Center)	Rank (Right)
Democrat→ Donald Trump	1	8	1
Republican→Donald Trump	2	66	4
Twitter→Donald Trump	3	67	392
Donald Trump→Democrat	4	3	2
Bernie Sander→Donald Trump	5	67	153
Donald Trump→Ted Cruz	6	67	393
Republican→Democrat	7	66	47
Democrat→Republican	8	41	7
Donald Trump→China	9	10	5
Donald Trump→Bush	10	67	393

Frequent pairs in the right-leaning media

Entity pairs	Rank (Left)	Rank (Center)	Rank (Right)
Democrat→Donald Trump	1	8	1
Donald Trump→Democrat	4	3	2
Donald Trump→Chinese	30	17	3
Republican→Donald Trump	2	66	4
Donald Trump→China	9	10	5
Donald Trump→Joe Biden	20	8	6
Democrat→Republican	8	41	7
Joe Biden→Donald Trump	38	5	8
Donald Trump→Russia	23	17	9
House→Donald Trump	12	67	10

Table 7: Frequency rank of entity pairs presented with a negative sentiment in the COVID-19 news dataset

lationships between political entities appear across multiple sentences in news articles.

To tackle the problem, we proposed DSE2QA (Directed Sentiment Extraction to Question-Answering), which is a simple yet effective method of utilizing BERT-like pretrained transformers by predicting answers for binary questions on whether a sentiment relationship exists in a given text. Answers for each sentiment class are aggregated to make a final guess. Evaluation experiments show the approach outperforms state-of-the-art classification models, such as the fine-tuned RoBERTa classification model. We hypothesize the language understanding ability of the BERT-like pretrained transformer may contribute to the high performance, combined with its facility of taking auxiliary input. Furthermore, the performance increase with the pseudo questions implies that a few keywords may suffice to make an inquiry. Future research could investigate which kind of pretrained transformer is the most effective for understanding the meaning of the augmented question, as DSE2QA's backbone can be replaced with any BERT-like transformer. Also, this study calls for future studies on advanced methods for directed sentiment extraction. A potential approach could jointly learn entity recognition and directed sentiment extraction as similarly tackled by a study on information extraction (Bekoulis et al., 2018).

As the last step, we conducted case studies by analyzing directed sentiments in news text for the US election and COVID-19 pandemic. The observations not only add empirical understandings to the social science research but also highlight the utility of the proposed problem, dataset, and model for political news analysis. We believe the proposed method can therefore further the current interdisciplinary efforts of the NLP, machine learning, and the social science communities (Grimmer and Stewart, 2013; Roberts et al., 2014; Joo and Steinert-Threlkeld, 2018).

Acknowledgements

We thank anonymous reviewers for their valuable comments. This work was supported by NSF SBE/SMA #1831848 "RIDIR: Integrated Communication Database and Computational Tools".

Ethics and Impact Statement

In online news and social media, people express diverse sentiment toward a target through text, such as blame, support, endorsement, to list a few. Quantifying and understanding the patterns is of significant interest in social science, but the lack of automated methods makes it difficult to handle large-scale data, which can reveal patterns in a comprehensive view. In this light, this study aims to develop automated methods of identifying directed sentiment between entities by defining a new NLP problem: directed sentiment extraction. The newly annotated dataset will facilitate future development of the NLP methods, and the DSE2QA approach will serve as a strong baseline.

The development of an automated method will have a broader impact by tackling real-world challenges such as bias in news reporting against political orientation, with potential collaboration with social science. Moreover, it will enable the discovery of hidden biases with regard to sentiment in online text, which can be mistakenly learned through data-driven methods. A fine-grained understanding of sentiment relationships will broadly contribute to building a fair machine learning model, which is of significant interest in AI ethics.

References

Muhammad Abdul-Mageed and Mona Diab. 2011. Subjectivity and sentiment annotation of modern standard arabic newswire. In *Proceedings of the 5th linguistic annotation workshop*, pages 110–118.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885.

Alexandra Balahur and Ralf Steinberger. 2009. Rethinking sentiment analysis in the news: from theory to practice and back. *Proceeding of WOMSA*, 9.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Andrea Ceron, Luigi Curini, Stefano M Iacus, and Giuseppe Porro. 2014. Every tweet counts? how sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to italy and france. *New media & society*, 16(2):340–358.

Danni Chen, Kunwoo Park, and Jungseock Joo. 2020a. Understanding gender stereotypes and electoral success from visual self-presentations of politicians in social media. In *Joint Workshop on Aesthetic and Technical Quality Assessment of Multimedia and Media Analytics for Societal Trends*, pages 21–25.

Emily Chen, Kristina Lerman, and Emilio Ferrara. 2020b. Covid-19: The first public coronavirus twitter dataset. *arXiv preprint arXiv:2003.07372*.

Amir DN Cohen, Shachar Rosenman, and Yoav Goldberg. 2020. Relation extraction as two-way span-prediction. *arXiv preprint arXiv:2010.04829*.

Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. SemEval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 519–535.

Kareem Darwish, Peter Stefanov, Michaël Aupetit, and Preslav Nakov. 2020. Unsupervised user stance detection on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 141–152.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

CHE Gilbert and Erric Hutto. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, volume 81, page 82.

Justin Grimmer and Brandon M Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297.

Jungseock Joo, Weixin Li, Francis F Steen, and Song-Chun Zhu. 2014. Visual persuasion: Inferring communicative intents of images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–223.

Jungseock Joo and Zachary C Steinert-Threlkeld. 2018. Image as data: Automated visual content analysis for political science. *arXiv preprint arXiv:1810.01544*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Shuailong Liang, Olivia Nicol, and Yue Zhang. 2019. Who blames whom in a crisis? detecting blame ties from news articles using neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 655–662.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

LLC Media Bias Fact Check. 2015. Media Bias/Fact Check. `https://mediabiasfactcheck.com`. [Online; accessed 21-Sep-2020].

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 767–776.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan Routledge, and Noah Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 4.

Kunwoo Park, Meeyoung Cha, and Eunhee Rhim. 2018. Positivity bias in customer satisfaction ratings. In *Companion Proceedings of the The Web Conference 2018*, pages 631–638.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *10th International Workshop on Semantic Evaluation*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G Rand. 2014. Structural topic models for open-ended survey responses. *American Journal of Political Science*, 58(4):1064–1082.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Stuart N Soroka. 2014. *Negativity in democratic politics: Causes and consequences*. Cambridge University Press.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.

Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time Twitter sentiment analysis of 2012 U.S. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120.

Donghyeon Won, Zachary C Steinert-Threlkeld, and Jungseock Joo. 2017. Protest activity detection and perceived violence estimation from social media images. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 786–794.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Seunghyun Yoon, Kunwoo Park, Joongbo Shin, Hongjun Lim, Seungpil Won, Meeyoung Cha, and Kyomin Jung. 2019. Detecting incongruity between news headline and body text via a deep hierarchical encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 791–800.

Lori Young and Stuart Soroka. 2012. Affective news: The automated coding of sentiment in political texts. *Political Communication*, 29(2):205–231.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9051–9062.

Ningyu Zhang, Luoqiu Li, Shumin Deng, Haiyang Yu, Xu Cheng, Wei Zhang, and Huajun Chen. 2020. Can fine-tuning pre-trained models lead to perfect nlp? a study of the generalizability of relation extraction. *arXiv preprint arXiv:2009.06206*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Instruction:
Based the given sentence, please identify if one entity (person, organization, or country) holds a positive or negative opinion towards another entity. There will be two entities in total. One is in red and the other is in blue.

It is also possible neither positive nor negative opinion exists in the sentence.
Please annotate such cases as neutral.

Please classify the sentence based on what people say instead of what they do.
For example, if a sentence only states the police arrest someone, this sentence should be classified as neutral.
Instead, if the police accuses someone of commiting a crime, this sentence should be classified as police holds a negative opinion towards the person.

Examples:
- Earlier on Tuesday, Mr. Trump criticized General Motors for making cars in Mexico.
(Negative: Trump holds a negative opinion towards General Motors)
- Hugo Ras has been accused of killing other people's rhino, and for that South Africans condemn him.
(Negative: South Africans holds a negative opinion towards Hugo Ras)
- DAVID Cameron's accused the Conservatives of failing to devolve essential welfare powers to, as agreed by the cross-party Smith Commission which considered further powers for the Scottish parliament last year.
(Neutral: There is no direct opinions between these two entities.)
- Prime Minister Stephen Harper shakes hands with Petty Harbour, N.L., during a campaign event in Toronto on Sept. 18, 2015. (Neutral: They shake hands just for politeness. No opinions exist.)
- Obama pulled Clinton into his administration after he defeated her in 2008 primary and has effusively praised her tenure as secretary of state. (Positive. Obama holds a positive opinion towards Clinton.)
- Earlier on Tuesday, Mr. Trump has not comments on General Motors making cars in Mexico. (Neutral.)

Note:
There are multiple annotators for each sentence. Your response will be judged as failed when it is different with other annotators. If the percentage of failed response from one annotator is above a threshold, the annotator will NOT get paid for ALL responses. Thanks for your participation.

Table A1: Instruction used for educating annotators in Amazon Mechanical Turk.

Method	Num. parameters	Avg. runtime per epoch	Micro F1	Macro F1	mAP
DSE2QA (Pseudo)	125M + 1536	2154s	**0.8072**	**0.6827**	**0.7528**
DSE2QA (Complete)	125M + 1536	2149s	0.7892	0.6751	0.7724
RoBERTa	125M + 3840	437s	0.7618	0.6516	0.7493
LNZ (Combined)	3.03M	12s	0.694	0.5189	0.4819
LNZ (Context)	2.9M	12s	0.6331	0.4518	0.3908
LNZ (EntityPrior)	2.65M	4.8s	0.5914	0.4427	0.31

Table A2: Model details and evaluated performance on the validation set. Top performance for each metric is marked as bold.

Method	0	1	2	3	4
DSE2QA (Pseudo)	0.9316	0.7157	**0.5952**	0.8358	**0.6658**
DSE2QA (Complete)	**0.9341**	**0.7228**	0.5747	**0.8457**	0.6161
RoBERTa	0.929	0.7299	0.5236	0.8232	0.6536
LNZ (Combined)	0.8452	0.4554	0.2887	0.6273	0.4311
LNZ (Context)	0.8233	0.4261	0.2887	0.568	0.3545
LNZ (EntityPrior)	0.7834	0.2181	0.2248	0.4405	0.4033

Table A3: AP per class measured on the test set. Top performance for each metric is marked as bold.

New Dataset and Strong Baselines for the Grammatical Error Correction of Russian

Viet Anh Trinh
Department of Computer Science
CUNY Graduate Center
New York, NY
vtrinh@gradcenter.cuny.edu

Alla Rozovskaya
Department of Computer Science
Queens College (CUNY)
New York, NY
arozovskaya@qc.cuny.edu

Abstract

Motivated by recent advancements in grammatical error correction in English and existing issues in the field, we describe a new resource, an annotated learner corpus of Russian, extracted from the Lang-8 language learning website. This new dataset is benchmarked against two grammatical error correction models that use state-of-the-art neural architectures. Results are provided on the newly-created corpus and are compared against performance on another, existing resource. We also evaluate the contribution of the Lang-8 training data to the grammatical error correction of Russian and perform type-based analysis of the models. The expert annotations are available for research purposes.

1 Introduction

The task of Grammatical Error Correction (GEC) is concerned with correcting various grammatical and usage errors in text. Recently, much progress has been made, especially in English GEC, within the framework of neural machine translation (NMT) approaches. In spite of the progress, the issue of building *robust* models in GEC has been emphasized: Mita et al. (2019) showed that the performance of the models varies significantly across corpora and that single-corpus evaluation may be unreliable. While in English more efforts are being made in this direction, in other languages, due to lack of benchmark corpora and other resources, very little work has been done. More importantly, as learner data in other language is very hard to come by and expensive to annotate, few benchmark corpora exist in other languages.

We develop a benchmark corpus for Russian learner data, by providing expert quality annotations for a subset of the Russian subcorpus of Lang-8, henceforth *RU-Lang8* dataset. Lang-8 (Mizumoto et al., 2012) is a dataset collected from a language learning website and partially corrected by native language volunteers. In Russian GEC, Rozovskaya and Roth (2019) recently released an annotated learner corpus, RULEC. The expert annotations that we provide will allow researchers to use the created corpus as another evaluation benchmark corpus for non-English GEC. As we show, RU-Lang8 is more diverse than RULEC in terms of the first language backgrounds, and the genre of writing. We benchmark two state-of-the-art neural machine translation models on the new corpus: a convolutional neural network (CNN) and a Transformer model.

The paper makes the following contributions: (1) We generate gold annotations for Lang-8 data to create an additional evaluation dataset for Russian GEC, which is more diverse linguistically and contains data of different genre of writing, compared to the existing resource RULEC. We make the resource available for research purposes;[1] (2) We provide benchmark results on this new corpus, using state-of-the-art models that are trained on synthetic data and learner data; (3) We provide an error analysis showing that most of the grammar errors are still challenging for the current systems.

2 Related Work

Progress in English GEC There has been a lot of work on grammatical error correction, but most of the research has been done on English (Rozovskaya and Roth, 2011; Susanto et al., 2014; Yuan and Briscoe, 2016; Hoang et al., 2016; Chollampatt et al., 2016; Junczys-Dowmunt and Grundkiewicz, 2016; Mizumoto and Matsumoto, 2016; Rozovskaya and Roth, 2016; Jianshu et al., 2017; Chollampatt and Ng, 2018; Kaneko et al., 2020).

Recently, state-of-the-art results were obtained using statistical and neural machine translation approaches. The systems are typically trained on a combination of native data with synthetic er-

---

[1] `https://github.com/arozovskaya/RU-Lang8`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4103–4111
August 1–6, 2021.

rors and naturally-occurring learner data from NUCLE (Dahlmeier et al., 2013) and the English part of the Lang-8 corpus (Mizumoto et al., 2012), even though the latter is only partially corrected and is known to contain a lot of noise.

Motivated by the issue of robustness, a recent shared task on English GEC (Bryant et al., 2019) released new evaluation data, both from learners of English and native speakers. Napoles et al. (2019) further addressed the issue of robustness of GEC models, by proposing novel evaluation metrics, and also released a diverse GEC dataset.

GEC on Other Languages Two most prominent attempts at GEC in other languages include shared tasks on Arabic and Chinese text correction. In Arabic, a large-scale corpus (2M words) was collected and annotated as part of the QALB project (Zaghouani et al., 2014). There have also been three shared tasks on Chinese grammatical error diagnosis (Lee et al., 2016; Rao et al., 2017, 2018). In other languages, attempts at automatic grammar detection and correction have been limited to identifying specific types of misuse (grammar or spelling) (Imamura et al., 2012; Israel et al., 2013; de Ilarraza et al., 2008; Vincze et al., 2014).

The most relevant to us is the work of Rozovskaya and Roth (2019) that made available an annotated corpus of Russian learner essays. The data released in Rozovskaya and Roth (2019) is relatively uniform, as it is all produced by native English speakers, whereas the RU-Lang8 data comes from a diverse set of speakers.

3 The RU-Lang8 Dataset

RU-Lang8 was created using data collected as part of the Lang-8 corpus (Mizumoto et al., 2012). The Lang-8 learner corpus is a dataset compiled from a language learning website.[2] It contains data from learners of a variety of foreign languages and is weakly annotated (partial corrections are provided by volunteers, but these are quite noisy). The Lang-8 corpus consists of pairs of sentence (source, target), where the *source* denotes the original sentence, and the *target* refers to the modified version that may contain partial corrections and volunteer commentaries. While the English subcorpus contains over 30 million tokens, the Russian learner subcorpus is small, containing about 633,000 tokens. We created a subset of 54,000 tokens and had it manually corrected by expert annotators. This newly-created resource with expert annotations is comparable in size to existing GEC datasets, and should be a valuable addition to multi-lingual resources in GEC. The RU-Lang8 corpus differs from RULEC: the latter consists of essays written in a University setting in a controlled environment, while the Lang-8 data was collected online; the majority of texts in RU-Lang8 are short paragraphs or questions posed by learners.

First language (%)	**First language (%)**
Japanese (37.9)	Portuguese (7.5)
English (14.0)	German (4.2)
Korean (11.8)	Polish (3.0)
Trad. Chinese (8.1)	Spanish (1.0)
Mandarin (7.6)	Mongolian (0.7)

Table 1: 10 most common first language backgrounds for data from Russian learners in the Lang-8 corpus.

Partition	**Sentences**	**Tokens in the source side**
Train	43,848	578,383
Dev	1,968	23,138
Test	2,444	31,603

Table 2: Statistics on the Russian data from Lang-8. The development and the test partitions are manually re-annotated. The training partition includes original noisy corrections.

RU-Lang8 Preprocessing From the Lang-8 corpus, we extract all sentence pairs, where the source and the target sentences are in Russian, using the tool which we modified for Russian (Chollampatt and Ng, 2018). The Lang-8 corpus also contains information about the author's first language. Overall, there are 34 first languages in the Russian subcorpus. Table 1 shows the distribution of the most common first languages in the dataset. The most common first language is Japanese (37% of all writers). Other common first languages are English, Korean, Traditional Chinese, and Mandarin.

The extracted sentence pairs are then tokenized using an in-house tokenizer and further cleaned up, by removing sentence pairs where the target side includes corrector's comments. As a result, 51,575 sentence pairs are kept. These sentence pairs are randomized and split up into training, development, and test partitions. The development and test partitions are manually re-annotated, as described below. The sentence pairs from the training partition are not re-annotated but contain the original noisy Lang-8 corrections. The sizes of the subcorpora are shown in Table 2.

---

[2] `https://lang-8.com`

Error type	**Example**
Lexical choice	предлагает "proposes" → утверждает "claims"
Extra word (open-class)	был "was" → ∅
Prep. (ins.,del.,repl.)	в "in" → из "from, out of"
Word form	вдохновленным "inspired" → вдохновенной "inspiring"
Noun:case	специалисты "experts" (pl.,nom) → специалистам "experts" (pl.,dat.)
Adj.:case	главная "main" (sg., fem., nom.) → главную "main" (sg., fem., acc.)
Verb:number/gender	живут "live" (3rd person pl.) → живет "lives" (3rd person sg.)
Verb:aspect	чувствовала "was feeling" (past, imperf.) → почувствовала "felt" (past, perf.)
Verb:voice	продолжала "continued" (past, active) → продолжалась "continued" (past, reflexive)

Table 3: **Some common grammatical error types in Russian learner data.** A complete set of errors with examples is shown in Appendix Table 11.

	Repl. (%)	**Ins.** (%)	**Del.** (%)	**Punc.** (%)	**Word order**(%)	**Total**
Dev	71.3	7.9	7.1	12.4	1.4	3,434
Test	74.2	8.4	8.4	7.7	1.3	3,354

Table 4: Statistics on corrections in RU-Lang8.

Correction of the RU-Lang8 Data The development and the test partitions of the RU-Lang8 corpus were manually annotated by a native Russian speaker, with a Master's degree and with prior annotation experience. To estimate the quality of the annotations, a second expert annotator with a similar background and native proficiency was used (see next section). In contrast to Rozovskaya and Roth (2019), where the errors are also tagged with error type at the level of syntax, morphology, lexical usage, and orthography, the annotation of RU-Lang8 is performed at the level of four operations: *Replace*, *Insert*, *Delete*, and *Word Order*. This new annotation framework speeds up the annotation process significantly and allows the annotator to focus on providing the appropriate correction, without having to think also about the linguistic error type. This approach was also used in the annotation of other GEC corpora (Reznicek et al., 2012; Boyd et al., 2014; Mohit et al., 2014).

The annotation was performed with a publicly-available tool used in other annotation efforts (Rozovskaya and Roth, 2010). Table 4 shows the distribution of errors in terms of edit operations.

A subset of the test data was also marked with error type, using the error classification schema of RULEC. This was done to allow for a comparison of the error distributions between RULEC and RU-Lang8. Table 3 shows examples of some common Russian learner errors in the RU-Lang8 dataset. The complete set of error categories is shown in Appendix Table 11. The error distributions in the two corpora are shown in Appendix Table 12. Because RULEC contains data from two groups of learners – foreign language learners of Russian and heritage speakers[3] – we show statistics for each RULEC group separately. The distribution of errors in RU-Lang8 is very similar to that of the foreign group in the RULEC corpus, even though in RULEC the learners come from the English-speaking language background, while in RU-Lang8, there is a lot of diversity with respect to the first language background. As for error rates,[4] the RU-Lang8 data has significantly higher error rates than both foreign and heritage parts of RULEC (Table 5). We attribute this to the overall higher proficiency level of the RULEC corpus writers. Finally, note that, as shown in Table 5, the error rates in development and test partitions of RU-Lang8 vary substantially: the error rates are 15.6% and 11.3% in the development and test sets, respectively. Since the sentences were selected uniformly at random, we do not have an explanation for the reason why the error distribution is different in the two subsets. We do believe that the varying error distributions might be useful, as they would reflect realistic scenarios where the test data may not have exactly the same distribution as the development/training data.

Inter-Annotator Agreement Computing inter-rater agreement in grammatical error correction

---

[3] The heritage group in RULEC includes native Russian speakers who grew up in the United States; these speakers have a different error distribution from that of foreign learners of Russian, with the majority of errors being of type punctuation and spelling.

[4] Error rate denotes the percentage of the tokens that have been modified by the annotator.

Corpus	Total words	Incorr. words	Error rate (%)
RULEC (Foreign)	164,071	11,343	6.9
RULEC (Herit.)	42,187	1,705	4.0
RU-Lang8 (dev)	23,138	3,605	15.6
RU-Lang8 (test)	31,603	3,558	11.3

Table 5: Error rates in RULEC (foreign and heritage speakers shown separately) and in RU-Lang8. *Error rates* throughout the paper refer to the percentage of tokens that have been modified.

Second pass	Error rate (%)	Judged correct (%)
Annotator A	1.55	90.0
Annotator B	1.02	97.4

Table 6: Inter-annotator agreement. *Error rates* based on the corrections on the second pass. *Judged correct* denotes the percentage of sentences in the agreement set that the second rater did not change.

is not trivial, as the space of possible corrections for a sentence is extremely large (Choshen and Abend, 2018; Bryant and Ng, 2015; Rozovskaya and Roth, 2021). To estimate the quality of the annotation, we have a second annotator independently re-annotate a subset of the data, 120 sentences. We compute inter-annotator agreement in two ways. First, we follow the method used for RULEC (Rozovskaya and Roth, 2019) where the texts corrected by one annotator were given to the second annotator. Agreement is computed as the percentage of sentences that did not have additional corrections on the second pass, as our goal is to make the sentence well-formed. 120 sentences from each annotator were given to the other annotator for the second pass. Table 6 shows that the error rate of the sentences corrected by annotator A (original annotator) on the second pass was 1.55%, with 90% of the sentences remaining unchanged. The sentences corrected by annotator B (second annotator) on the second pass had an error rate of less than 1.02%, and over 97% of the sentences did not have additional corrections. These agreement numbers are comparable to and even slightly higher than in RULEC (68.5% and 91% of unchanged sentences). The error rates are also in the same ballpark (0.67%-2.4% for RULEC).

We also measure agreement by treating reference corrections made by one annotator as gold and scoring the second annotator against them. .Results are shown in Table 7. The scores of 66.7 and 69.9 are higher than those reported on English CoNLL-14 (score of 45.91, Bryant and Ng (2015)).

Gold annotator	P	R	$F_{0.5}$
Annotator A as gold	66.0	69.6	66.7
Annotator B as gold	72.2	61.9	69.9

Table 7: Scoring one annotator against another.

4 Experiments

GEC Models We benchmark two state-of-the-art neural machine translation models: a Convolutional Encoder-Decoder Neural Network model (*CNN*) (Chollampatt and Ng, 2018) and a *Transformer* model (Naplava and Straka, 2019).[5] The Transformer model achieved the highest F-score on RULEC. Both models make use of the RULEC training and dev data (about 5K sentences) and native data with synthetic errors. The CNN model is trained jointly on RULEC and synthetic data, while the Transformer is pre-trained on synthetic data and finetuned on RULEC data.

The models make use of the two approaches of generating synthetic data that showed state-of-the-art performance on English GEC. The Transformer model makes use of the Aspell confusion sets method (Grundkiewicz et al., 2019) to generate synthetic errors in native data. In line with Grundkiewicz and Junczys-Dowmunt (2019), the word error rate used is that of 15%, where on average 15% of the tokens are perturbed, and, on top of these Aspell-generated confusions, characters are perturbed in 10% of the word tokens to account for spelling mistakes. We refer the reader to Naplava and Straka (2019) for details about the model implementation. The CNN models are trained on synthetic data that use part-of-speech (POS)-based confusion sets (Choe et al., 2019) (which we re-implemented for Russian). In all cases, results of single models are compared.

The models are trained on similar amounts of synthetic data: the CNN model uses 13 million sentences from the Web (Borisov and Galinskaya, 2014), while the Transformer model uses 10 million sentences from the Web (Bojar et al., 2017). Although we did not directly compare the two data sources, we assume the native data used by both models is of similar quality as the data comes from the Web in both cases. Since the synthetic data used in the CNN model is not focused on spelling errors, we run an off-the-shelf spellchecker for Russian,

---

[5] We thank Jakub Náplava for kindly agreeing to run the Transformer model on the RU-Lang8 data.

Model	**RULEC**			**RU-Lang8**		
	P	**R**	$F_{0.5}$	**P**	**R**	$F_{0.5}$
CNN	55.8	26.6	45.7	57.9	26.8	**47.0**
Transf.	59.1	26.1	**47.2**	-	-	-
Transf. (+dev)	63.3	27.5	***50.2***	55.3	28.5	46.5

Table 8: Results on the test of the CNN and Transformer models. Best result for each test set is in bold. Transformer+dev shows performance of Transformer model finetuned on both training and dev RULEC data.

Training data	**RULEC**			**RU-Lang8**		
	P	**R**	$F_{0.5}$	**P**	**R**	$F_{0.5}$
RU+synth.	55.8	26.6	45.7	57.9	26.8	47.0
RU+L8+synth.	54.5	27.6	**45.6**	58.8	29.6	**49.1**

Table 9: Results on RULEC and RU-Lang8 of the CNN models trained jointly on synthetic and learner data. RU stands for RULEC, and L8 stands for RU-Lang8.

developed following Flor (2012) for English.

Evaluation The models are evaluated on the test partitions of RULEC and RU-Lang8. Comparing system output against human-generated reference is a standard practice in GEC. Several measures have been proposed, such as M^2 scorer (Dahlmeier and Ng, 2012), GLEU (Napoles et al., 2015), ERRANT (Bryant and Ng, 2015), and I-measure (Felice and Briscoe, 2015). M^2 computes precision, recall, and F-score and has been widely used in evaluating GEC systems, and we use it here to compare with previous results on RULEC. M^2 has been used with different beta parameter values, the default is $beta = 0.5$, weighing precision twice as high as recall, which we use here.

Overall Performance We show results for the two models in Table 8.[6] The Transformer finetuned on RULEC train data outperforms the CNN on RULEC *by 1.5 points*, and by almost 5 points when finetuned on the union of training and dev RULEC data. However, on RU-Lang8, the CNN model outperforms the transformer slightly.

Contribution of Lang8 Training Data This is shown in Table 9. Interestingly, performance on RULEC does not improve, while performance on RU-Lang8 improves by 2 points. The latter is not surprising since the data comes from the same domain, however, it is surprising that there is no effect on the RULEC corpus.

Performance Analysis by Error Type We also evaluate the best models shown in Table 8 per error type. Evaluating the precision requires classifying the edits proposed by the system, however, recall can be computed, using the types of the gold edits. In Table 10 in the Appendix, we show the recall on the most common error types. The type-based performance analysis reveals which errors are more challenging for the systems. The highest recall by far (ranging between 52.0% and 70.5%) on both datasets is achieved on spelling errors by both models. The Transformer also achieves a recall of 60% on verb agreement errors on both datasets. The following error categories are more challenging for both models: noun case, preposition, adjective case errors, and punctuation. Finally, the most challenging errors are lexical choice errors, where the recall is below 5% for both models and on both datasets. This supports the observation that current models perform best on spelling errors and currently struggle with other phenomena, which is further exacerbated by the morphological complexity of Russian: the performance on Russian falls behind that on English and German (Grundkiewicz and Junczys-Dowmunt, 2019). Our error-type-based analysis is also in line with the findings in the study on two English corpora, as well as RULEC and RU-Lang8 Rozovskaya and Roth (2021). Specifically, while lexical errors are some of the most common learner mistakes, only a small fraction of system edits are of lexical type.

5 Conclusion

This paper presents an annotated Russian learner corpus based on data from the Lang-8 website. The dataset is more diverse than the existing resource RULEC from the point of view of the first language backgrounds, and also differs in the genre of writing. We benchmark two state-of-the-art models that are trained on learner data and synthetic data, using two competitive noisification techniques.

We believe that the RU-Lang8 corpus with expert annotations is a valuable contribution to the GEC field, where a lot of progress has been made in English, due to a large number of resources and benchmark corpora, but where very few works focus on non-English GEC.

Acknowledgments

We are grateful to Jakub Náplava for the help running his system on the RU-Lang8 data. We thank the anonymous reviewers for their comments.

---

[6] The last row shows the result of finetuning the Transformer with both train and dev RULEC data.

References

O. Bojar, C. Rajen, Federmann C, Y. Graham, B. Haddow M. Huck, P. Koehn, Q. Liu, V. Logacheva, and C. Monz. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Second Conference onMachine Translation*.

A. Borisov and I. Galinskaya. 2014. Yandex school of data analysis russian-english machine translation system for WMT14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

A. Boyd, J. Hana, L. Nicolas, D. Meurers, K. Wisniewski, A. Abel, K. Schone, B. Štindlová, and C. Vettori. 2014. The MERLIN corpus: Learner language and the CEFR. In *LREC*.

C. Bryant, M. Felice, Ø. Andersen, and T. Briscoe. 2019. The BEA-19 shared task on grammatical error correction. In *Proceedings of the ACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA-19)*.

C. Bryant and H. T. Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *ACL*.

Y. J. Choe, J. Ham, K. Park, and Y. Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning . In *Proceedings of the ACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA-19)*.

S. Chollampatt and H.-T. Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI*. Association for the Advancement of Artificial Intelligence.

S. Chollampatt, K. Taghipour, and H. T. Ng. 2016. Neural network translation models for grammatical error correction. In *IJCAI*.

L. Choshen and O. Abend. 2018. Automatic metric validation for grammatical error correction. In *ACL*.

D. Dahlmeier and H. T. Ng. 2012. A beam-search decoder for grammatical error correction. In *Proceedings of EMNLP-CoNLL*.

D. Dahlmeier, H. T. Ng, and S. M. Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

M. Felice and T. Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *NAACL-HLT*.

M. Flor. 2012. Four types of context for automatic spelling correction. *Traitement Automatique des Langues (TAL). (Special Issue: Managing noise in the signal: error handling in natural language processing)*, 3(53):61–99.

R. Grundkiewicz and M. Junczys-Dowmunt. 2019. Minimally-augmented grammatical error correction. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT*.

R. Grundkiewicz, M. Junczys-Dowmunt, and K. Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the ACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA-19)*.

D.-T. Hoang, S. Chollampatt, and H. T. Ng. 2016. Exploiting n-best hypotheses to improve an SMT approach to grammatical error correction. In *IJCAI*.

A. Díaz de Ilarraza, K. Gojenola, and M. Oronoz. 2008. Detecting erroneous uses of complex postpositions in an agglutinative language. In *Proceedings of COLING*.

K. Imamura, K. Saito, K. Sadamitsu, and H. Nishikawa. 2012. Grammar error correction using pseudo-error sentences and domain adaptation. In *ACL*.

R. Israel, M. Dickinson, and S.-H. Lee. 2013. Detecting and correcting learner korean particle omission errors. In *IJCNLP*. Association for Computational Linguistics.

J. Jianshu, Q. Wang, K. Toutanova, Y. Gong, S. Truong, and Jianfeng J. Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *ACL*.

M. Junczys-Dowmunt and R. Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *EMNLP*.

M. Kaneko, M. Mita, S. Kiyono, J. Suzuki, and K. Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *ACL*.

L.-H. Lee, G. Rao, L.-C. Yu, E. Xun, B. Zhang, and L.-P. Chang. 2016. Overview of NLP-TEA 2016 shared task for chinese grammatical error diagnosis. In *3rd Workshop on Natural Language Processing Techniques for Educational Applications*. Association for Computational Linguistics.

M. Mita, T. Mizumoto, M. Kaneko, R. Nagata, and K. Inui. 2019. Cross-corpora evaluation and analysis of grammatical error correction models – is single-corpus evaluation enough? In *NAACL*.

T. Mizumoto, Y. Hayashibe, M. Komachi, M. Nagata, and Y. Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of esl writings. In *COLING*.

T. Mizumoto and Y. Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *NAACL*.

B. Mohit, A. Rozovskaya, N. Habash, W. Zaghouani, and O. Obeid. 2014. The first QALB shared task on automatic text correction for Arabic. In *Proceedings of the EMNLP Workshop on Arabic Natural Language Processing (ANLP)*.

J. Naplava and M. Straka. 2019. Grammatical error correction in low-resource scenarios. In *W-NUT Workshop*.

C. Napoles, M. Nădejde, and J. Tetreault. 2019. Enabling robust grammatical error correction in new domains: Data sets, metrics, and analyses. In *Transactions of ACL*.

C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault. 2015. Ground truth for grammatical error correction metrics. In *ACL*.

G. Rao, Q. Gong, B. Zhang, and E. Xun. 2018. Overview of NLPTEA-2018 share task Chinese grammatical error diagnosis. In *Proceedings of the Fifth Workshop on Natural Language Processing Techniques for Educational Applications*.

G. Rao, B. Zhang, and E. Xun. 2017. IJCNLP-2017 task 1: Chinese grammatical error diagnosis. In *IJCNLP*. Association for Computational Linguistics.

M. Reznicek, A. Ludeling, and C. Krummes. 2012. Das falkohandbuch. korpusaufbau und annotationen version 2.0.

A. Rozovskaya and D. Roth. 2010. Annotating ESL errors: Challenges and rewards. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*.

A. Rozovskaya and D. Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of ACL*.

A. Rozovskaya and D. Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *ACL*.

A. Rozovskaya and D. Roth. 2019. Grammar error correction in morphologically-rich languages: The case of russian. In *Transactions of ACL*.

A. Rozovskaya and D. Roth. 2021. How good (really) are grammatical error correction systems? In *EACL*.

R. H. Susanto, P. Phandi, and H. T. Ng. 2014. System combination for grammatical error correction. In *EMNLP*.

V. Vincze, J. Zsibrita, P. Durst, and M. Katalin Szabó. 2014. Automatic error detection concerning the definite and indefinite conjugation in the hunlearner corpus. In *Proceedings of LREC*.

Z. Yuan and T. Briscoe. 2016. Grammatical error correction using neural machine translation. In *NAACL*.

W. Zaghouani, B. Mohit, N. Habash, O. Obeid, N. Tomeh, A. Rozovskaya, N. Farra, S. Alkuhlani, and K. Oflazer. 2014. Large scale arabic error annotation: Guidelines and framework. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

A Additional Results

Error type	**RULEC**		**RU-Lang8**	
	CNN	**Trans.**	**CNN**	**Trans**
Punc.	**24.4**	7.3	**15.4**	10.3
Noun:case	**38.4**	36.7	**40.5**	35.1
Spelling	62.0	**66.7**	52.0	**70.5**
Lex. choice	1.0	3.4	0.5	2.6
Prep.	30.1	29.6	10.9	**17.4**
Adj.:case	20.5	22.0	22.5	27.5
Verb:agr.	27.1	**59.4**	42.4	**60.6**

Table 10: Recall by error type of the model trained on native data combined with learner data. Best result for each error type and dataset is in bold.

Error type	**Example**
Spelling	моркое → морское “marine”
Lexical choice (single-token)	предлагает “proposes” → утверждает “claims”
Replace (multi-token)	грозит “threatens” → создает угрозу “creates a threat”
Punc.	∅ →,
Extra word (open-class)	был “was” → ∅
Missing word (open-class)	∅ → для того “with the purpose of”
Prep. (ins.,del.,repl.)	в “in” → из “from, out of”
Conjunction	и “and” → ∅
Word form	вдохновленным “inspired” → вдохновенной “inspiring”
Noun:case	специалисты “experts” (pl.,nom) → специалистам “experts” (pl.,dat.)
Noun:number	пола“gender” (sg.,gen.) → полов “gender” (pl.,gen.)
Adj.:case	главная “main” (sg., fem., nom.) → главную “main” (sg., fem., acc.)
Adj.:number	дальнейшие “future” (pl.,nom.) → дальнейшее “future” (sg.,nom.))
Adj.:gender	которое “which” (sg.,neutral) → которая “which” (sg. fem.)
Verb:number/gender	живут “live” (3rd person pl.) → живет “lives” (3rd person sg.)
Verb:aspect	чувствовала “was feeling” (past, imperf.) → почувствовала “felt” (past, perf.)
Verb:voice	продолжала “continued” (past, active) → продолжалась “continued” (past, reflexive)
Verb:tense	предлагал “offered” (past tense) → предлагает “offers” (present tense)
Verb:other	соблазнить “to seduce” → соблазнил “seduced”

Table 11: **Grammatical error types in Russian learner data.**

Error type	**Percentage** (%)		
	RULEC-Foreign	**RULEC-Heritage**	**RU-Lang-8**
Spelling	18.6	42.4	19.2
Lexical choice	13.3	5.5	11.6
Punctuation	7.6	22.9	10.3
Missing word	8.9	4.7	7.3
Replace	6.3	2.8	1.7
Extra word	5.7	2.4	6.6
Preposition	3.3	1.5	4.6
Word form	3.1	2.1	1.0
Pronoun	1.0	0.5	1.0
Conjunction	0.8	0.1	1.0
Noun:case	14.0	7.8	12.6
Noun:number	2.5	1.8	0.7
Noun:gender	0.3	0.2	0.7
Adj.:case	3.9	2.1	6.3
Adj.:number	1.0	0.3	-
Adj.:gender	1.4	0.5	-
Verb:number/gender	2.5	1.6	1.7
Verb:aspect	2.0	0.2	3.6
Verb:tense	1.2	0.3	4.6
Verb:voice	1.2	0.2	0.7
Verb:other	0.5	0.1	-

Table 12: Distribution by error type. *Replace* includes phenomena not covered by other categories, e.g., additional morphological phenomena, replacing multi-word expressions, and word order.

A *Formidable* Ability: Detecting Adjectival Extremeness with DSMs

Farhan Samir[1], **Barend Beekhuizen**[2], **Suzanne Stevenson**[1]
[1]Department of Computer Science
[2]Department of Language Studies and Department of Linguistics
University of Toronto
{fsamir,suzanne}@cs.toronto.edu
barendbeekhuizen@utoronto.ca

Abstract

While distributional semantic models (DSMs) can successfully capture the similarity structure *within* a semantic domain, less is known about their ability to represent abstract semantic properties that hold *across* domains. Such properties can form the basis for abstract semantic classes that are a crucial aspect of human semantic knowledge. For example, the abstract class of extreme adjectives (such as *brilliant* and *freezing*) spans a wide range of domains (here, INTELLIGENCE and TEMPERATURE). Using a model that compares query items to an aggregate DSM representation of a set of extreme adjectives, we show that novel adjectives can be classified accurately, supporting the insight that a cross-domain property like extremeness can be captured in a word's DSM representation. We then use the extremeness classifier to model the emergence of intensifier meaning in adverbs, demonstrating, in a separate task, the effectiveness of detecting this abstract semantic property.

1 Distributional Models and Abstract Semantic Classes

Distributional semantic models (DSMs) are widely used as representations of word-level semantics. However, open questions remain as to precisely which aspects of human semantic knowledge DSMs effectively capture (e.g., Baroni et al., 2014; Hollis and Westbury, 2016; Schnabel et al., 2015; Utsumi, 2020). For example, popular DSMs such as word2vec and GloVe have been shown to predict human ratings of semantic features of objects (Rubinstein et al., 2015; Grand et al., 2018). However, performance is variable across features and object categories (Grand et al., 2018), and in particular, is better for taxonomic properties ('is an animal', 'is a weapon') than for general attributive properties ('is yellow', 'is dangerous') (Rubinstein et al., 2015).

While people may or may not have semantic categories such as "all yellow things", abstract semantic classes are an important part of human linguistic knowledge that should be captured in a computational system. Note that by *abstract* we mean the schematic properties of word meaning, rather than the content-related classes;[1] such properties abstract over commonalities of meaning that may *cross* traditional semantic domains. Consider, e.g., a semantic verb class such as change-of-state (Levin, 1993; Kipper et al., 2008), with members such as *melt* (the TEMPERATURE domain) and *quicken* (SPEED), or relational adjectives (Boleda et al., 2012), including, e.g., *Chinese* (NATIONALITY) or *pulmonary* (BODY-PART).

Much work shows the ability of DSMs to match human knowledge of semantic properties *within* a domain (e.g., Baroni et al., 2014; Pereira et al., 2016; An et al., 2018; Grand et al., 2018), but there is little work, to our knowledge, on whether the similarity structure of a DSM is sensitive to commonalities of abstract properties that hold *across* a variety of semantic domains.[2] Research on vector-based representations of analogy suggests that DSMs may be limited in their ability to represent cross-domain word relations: Rogers et al. (2017) show that cross-domain analogical relations like hypernymy (e.g., *turtle:reptile::salmon:fish*) are significantly harder to solve than within-domain ones (e.g., *Paris:France::Ottawa:Canada*). Lu et al. (2019) make significant progress towards representing such relations, showing that a DSM can form the basis for detecting the cross-domain word

---

[1] The same distinction between *schematic* and *content* is applied in Paradis (2001); Cruse and Togia (1996). It is also worth noting explicitly that our use of the term 'abstract' in this sense is not to be interpreted as 'not concrete'.

[2] DSMs may, e.g., encode concreteness and valence/arousal/dominance (e.g., Hollis and Westbury, 2016; Hollis et al., 2017), but the former can be viewed as a taxonomic property, and the latter as within the EMOTION domain.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4112–4125
August 1–6, 2021.

relations, such as antonymy (*love–hate* in the EMOTION domain, *rich–poor* in FINANCE); however, to achieve this, their method requires additional learning of a relation-specific warping of the distributional semantic space. Our goal here is to see whether the similarity structure of the (original) DSM itself directly captures such cross-domain knowledge.

We focus as a test case on the abstract class of *extreme adjectives*: scalar adjectives that express an extreme value of their scale, such as *brilliant* and *freezing*. Previous computational work on scalar adjectives has focused on assessing their relative ranking within a domain (e.g., learning that *smart* < *brilliant* on the INTELLIGENCE scale) (e.g., Ruppenhofer et al., 2014; Cocos et al., 2018). However, as work in linguistics shows (Cruse, 1986; Paradis, 2001; Morzycki, 2012), extreme adjectives do not simply behave as if they are further along their scale, but rather (as a class) have distinguishing semantic properties. Our goal is to see whether DSMs can capture this cross-domain property of "extremeness". In our first experiment, we demonstrate that we can successfully identify extreme adjectives, across a wide range of domains, on the basis of the information contained in their DSM representations alone.

Our next goal was to show that this ability to detect the abstract property of extremeness would be useful in further tasks. We begin with the novel hypothesis that an adjective's extremeness is a strong predictor of its future use in an intensifier adverb – i.e., in phrases like *staggeringly easy* and *monumentally wrong*. Our second experiment then shows that our classification of extreme adjectives can improve over an existing computational approach (Luo et al., 2019) in a historical prediction task of emerging intensifier meanings.

2 Our Case Study: Extreme Adjectives

Extreme adjectives are a subclass of scalar adjectives that includes words such as *awesome*, *brilliant*, and *freezing*. Like all scalar adjectives, their semantics includes the specification of a scale, such as (for these) DESIRABILITY, INTELLIGENCE, and TEMPERATURE, respectively, along with some position or range on that scale. The distinguishing aspect of extreme adjectives is that they represent an extreme value at one end or the other of the scale (e.g., Cruse, 1986; Morzycki, 2012; Paradis, 2001), a value that is so high/low as to possibly even be considered "off the scale" (Morzycki, 2012). This makes extreme adjectives a good case study for us: their abstract property of "extremeness" holds across a wide variety of semantic scales, and thus crosses individual semantic domains, such as INTELLIGENCE or TEMPERATURE.[3]

Category (example)	**Intensification (*very* X)**	**Endpoint-oriented (*almost* X)**	**Extreme degree (*absolutely* X)**
Non-gradable (*civic*)	?	?	?
Open-scale (*big*)	✓	?	?
Closed-scale (*full*)	✓	✓	✓
Extreme (*huge*)	?	?	✓

Table 1: Examples showing differences in types of modifiers usable across categories of adjectives.

Our goal is to assess whether a DSM can directly capture the similarity among members of this kind of abstract class. We address this question in two ways: with a direct evaluation, testing whether we can classify adjectives as extreme or not, and with an indirect evaluation, which uses our extremeness classifier in a separate task, to show its value in NLP applications.

Our first experiment uses similarity within a DSM as the basis for classification of extreme adjectives. Linguists have long noted that distributional tests can distinguish extreme adjectives from other classes (e.g. Cruse, 1986), as in the following examples (elaborated in Table 1).[4]

1. Martha is ?very/?almost/absolutely **ecstatic**. [Extreme]
2. Martha is very/?almost/?absolutely **grateful**. [Not Extreme]

Here, we propose an approach that replaces similarity along manual distributional tests with similarity within a general DSM. Note that such an approach is not a priori guaranteed to succeed. First, the manually-identified probes are specifically chosen to highlight the distinguishing properties; looking at representations derived from all of a word's contexts may mean that the useful signal about what makes an adjective extreme is drowned in "noise"

---

[3] The difference between a semantic domain and a semantic scale is not crucial here; the important point is that "extremeness" is a property that crosses what are typically thought of as more narrowly-defined semantic areas.

[4] Note that the versions with "?" in (1), (2), and Table 1 are less felicitous and require additional facilitating context, rather than being outright ungrammatical; this is not unusual for distributional tests of semantic classes.

(with respect to the property of extremeness). Second, we're looking for an abstract property that crosses semantic domains – i.e., we're not asking if EMOTION adjectives are more similar to each other than to INTELLIGENCE adjectives; we're asking whether extremes in both domains (*ecstatic* and *brilliant*) are more similar to each other than to non-extreme adjectives.

In our second experiment, we perform a downstream task that uses our classification of an adjective as extreme or not. Here we study the historical emergence of intensifier meanings of adverbs (e.g., *monumentally* coming to mean 'very', as in *monumentally wrong*), for which in-domain similarities in DSMs have previously been found to model the phenomenon with some accuracy (Luo et al., 2019). Using our cross-domain model of extremeness, we operationalize our novel linguistic insight that adjectival extremeness (*monumental*) is inextricably linked to the emergence of intensifier senses (*monumentally wrong*).

To preview our results, we find that our method of using a DSM to classify adjectives as extreme or not substantially improves over a statistical method drawing on linguistically-devised distributional tests. That is, we see that the general similarity space of a DSM can be used as an effective replacement for similarity with respect to manually-identified probes in the detection of an abstract semantic class. Moreover, in our experiments on intensifier emergence, we find that our identification of the cross-domain property of extremeness also shows improvement over a more standard application of DSMs that assesses in-domain similarities. This pair of experiments thus provides evidence that a DSM can successfully capture an abstract class defined by cross-domain similarity.[5]

3 Classifying Extreme Adjectives

Here we propose an approach for identifying extreme adjectives using a DSM. Other work in computational linguistics has considered automatic means for placing scalar adjectives in the appropriate relative position along their scale (e.g. Sheinman et al., 2013), including approaches using DSMs to do so (e.g. Kim and de Marneffe, 2013; Sharma et al., 2017). Such methods do not distinguish extreme adjectives as a special set across semantic domains, but rather assess them as expressing a higher/lower level compared to other adjectives within the domain – e.g., *brilliant* is a higher degree of INTELLIGENCE than *smart*. Our approach instead focuses on extremeness as a categorical property of adjectives that is independent of any particular scale (following Cruse, 1986; Paradis, 2001; Morzycki, 2012, among others).

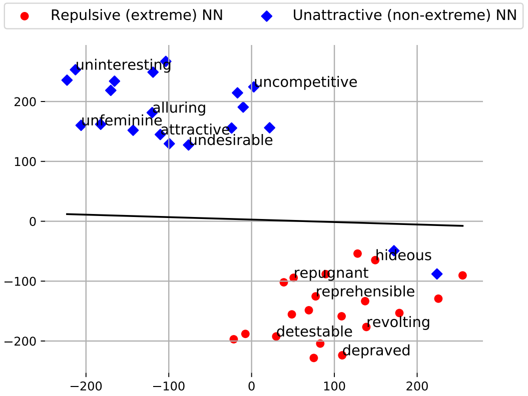

Figure 1: Nearest neighbours of *repulsive* (extreme) and *unattractive* (non-extreme), visualized with t-SNE (Maaten and Hinton, 2008) on word2vec embeddings (Mikolov et al., 2013).

As set out earlier, our hypothesis is that the similarity space of a DSM can capture the (cross-domain) similarity of the members of an abstract semantic class. Fig. 1 illustrates this intuition: the nearest neighbors of the extreme adjective *repulsive* are other extreme adjectives (*detestable*, *revolting*); conversely, the non-extreme counterpart *unattractive* has other non-extreme nearest neighbours (*undesirable*, *uninteresting*). Importantly, the two sets of nearest neighbours show a clear separability of the extreme adjectives and non-extreme adjectives, suggesting that the contextual distribution of an adjective, as represented in its word embedding, contains useful information for classifying whether it is extreme.

Reflecting this intuition, we propose to identify extremeness of adjectives using a prototype approach, in which we classify each adjective by comparing its vector to the average vector (a "prototype") of a set of extreme adjectives within a DSM. While such an approach has been used previously to characterize words *within* a semantic domain (e.g., EMOTION in the case of Xu et al. (2020), and each of a variety of domains in An et al. (2018)), here we test whether such a prototype vector can abstract over the individual semantic domains to

---

[5] All code and data are available at `https://github.com/smfsamir/detect-adjectival-extremeness`.

capture the cross-domain property of extremeness. Classification will be successful to the extent that vectors of adjectives that share the abstract property of extremeness are sufficiently similar to serve as an informative prototype.

3.1 Dataset

We collect a dataset of extreme and non-extreme adjectives in English for training and evaluating a supervised classifier. We started with a set of 54 adjectives identified as extreme (Morzycki, 2012; Paradis, 2001; Huttenlocher et al., 1971; Cruse, 1986; Lassiter, 2017). We then added extreme adjectives from human-annotated datasets of adjectival intensity (Cocos et al., 2018; Wilkinson and Tim, 2016; de Melo and Bansal, 2013; Ruppenhofer et al., 2014). For each adjective, these datasets specify its scale and its human-rated range of intensity values. For each of the scales of the 54 previously-identified extreme adjectives, we gathered all further adjectives tied with or ranked above extreme adjectives in their intensity value ($N = 17$). After filtering out 3 adjectives with frequency less than 0.5 per million, we obtained a total of $N = 68$ extreme adjectives that cover a diverse set of adjectival scales, such as DESIRABILITY (*sensational*), INTELLIGENCE (*moronic*), and SIZE (*gigantic*). We then match each extreme adjective in our dataset with a non-extreme adjective matched for frequency in COCA (Corpus of Contemporary American English; 1B words; Davies, 2009). (Since word embeddings have been shown to encode frequency (e.g., Mu and Viswanath, 2018), it is important to control for this.) To avoid including extreme adjectives, we exclude any words that appear as the most intense adjective in the above datasets. Our non-extreme set includes 68 adjectives, both scalar and non-scalar; see data in the GitHub repository.

3.2 Methods to predict extremeness

Our goal is to see whether DSMs can support the classification of an abstract class that has been previously identified through manually-selected distributional tests. Thus we compare two approaches to identifying extreme adjectives: one implements a corpus-based measure to capture manually-identified distributional tests developed by semanticists (cf. Table 1), while the other uses a DSM to capture the cross-domain similarity of extreme adjectives. For both, we propose a method for deriving the probability $p(c|a)$ of the class $c \in \{\text{Extreme}, \text{Non-extreme}\}$ for an adjective a.

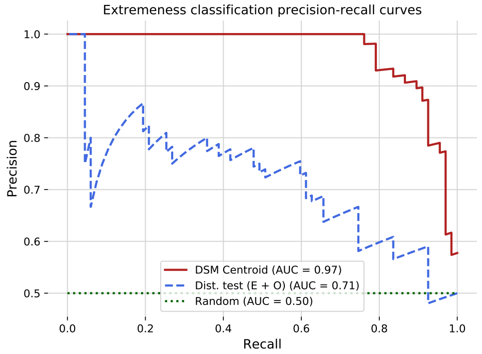

Figure 2: Results for classifying extreme adjectives.

3.2.1 The DISTRIBUTIONAL TEST method

Here we define a method based on the "probe" words specified in distributional tests of extremeness. Specifically, our approach looks at patterns of extreme adjectives that are readily assessed in a corpus: (1) frequent modification by extreme degree modifiers (Cruse, 1986); and (2) resistance to modification by endpoint-oriented modifiers and *very* (Morzycki, 2012).[6] The extreme degree modifiers we consider are E = {*absolutely*, *totally*, *simply*, *positively*, *downright*, *outright* }, while the other modifiers are O = { *almost*, *slightly*, *very* }. Formally, we measure the association with each of these sets of probes using normalized pointwise mutual information (NPMI).[7] For example, we assess the NPMI_E between an adjective a and (all) extreme degree modifiers e that immediately precede it in a sentence, as:

$$\text{NPMI}_E(e, a) = \log\left(\frac{p(e, a)}{p(e)p(a)}\right) \cdot \frac{1}{\log p(e, a)}$$

where e stands for any extreme degree modifier in E (i.e., we derive the probabilities above from pooled counts in COCA over a co-occurring with any $e \in E$). We compute the association NPMI_O with all $o \in O$ analogously. Using NPMI_E and NPMI_O as features, we train a logistic regression classifier with L2 regularization to obtain an estimation of $p(c|a)$, where $c \in \{\text{Extreme}, \text{Non-extreme}\}$. Combined with a probability threshold, the estimation of $p(c|a)$ allows us to predict whether an adjective is extreme.

---

[6] Other tests (such as prosodic cues or hyperbole, Morzycki, 2012), are difficult to identify in a written corpus.

[7] Normalizing mitigates the frequency bias known to impact PMI (Bouma, 2009; Jurafsky and Martin, 2014).

Classifier	Prec.	Recall	F$_1$
DISTRIB. TEST	0.76	0.66	0.71
DSM CENTROID	**0.90**	**0.91**	**0.90**

Table 2: Classification using a 0.5 threshold.

3.2.2 The DSM CENTROID method

To use a DSM to classify an adjective as extreme, we need to abstract away from the information about the particular scale or semantic domain that is captured in a word vector. That is: we want the model to recognize that *exquisite* is more like *huge*, *destitute*, and *repulsive* than like *big*, *poor*, and *unattractive*. As noted above, we draw on prototype approaches by classifying a novel adjective on the basis of how similar it is to the aggregate representation of a set of extreme adjectives. Analogously to the approach above, we use this similarity as a single feature in a logistic regression classifier to obtain an estimate of the probability of extremeness $p(c|a)$. For our experiments, we use the 300-dimensional pretrained word2vec embeddings of Mikolov et al. (2013).

3.3 Results

Because of the small size of our dataset ($N = 136$), we use a leave-one-out cross-validation procedure to evaluate the two methods. We evaluate performance in two ways: using precision-recall curves (shown in Fig. 2) and as Precision, Recall, and F_1 scores for the Extreme class using a classification threshold of 0.5 (shown in Table 2).

Both DSM CENTROID (AUC = .97) and DISTRIBUTIONAL TEST (AUC = .78) perform better than chance (AUC = .50). The DSM CENTROID method furthermore substantially outperforms the DISTRIBUTIONAL TEST method (AUC = .97 vs. AUC = .78; F_1 = .90 vs. F_1 = .71). This result provides strong evidence for our hypothesis that the similarity structure of a DSM can effectively capture an abstract, cross-domain property such as extremeness; in fact, at least in this case, it can do so better than a statistical corpus-based model based on manually-identified linguistic tests.

One concern with the leave-one-out approach is that there may be items in the training set (e.g., *massive*) that share semantic properties other than extremeness with the held-out item (e.g., *huge* – in this case, both are SIZE adjectives). To control for this confound, we conduct a variant of a k-fold cross-validation procedure, in which the folds consist of clusters of semantically similar adjectives, with the aim that the training data does not include adjectives from the same domain as the test items. Even in this controlled analysis, we find that the DSM CENTROID method robustly outperforms the DISTRIBUTIONAL TEST method. See Appendix A for details.

Classifier	Errors
DSM CENTROID	destitute, freezing, terrified; *runaway, memorable*
DISTRIB. TEST	gigantic, colossal, mammoth, gargantuan, immense; *identical, inconvenient, conventional, opposite*
Both	major, obese, microscopic; *creepy*

Table 3: Sample of errors made by each method, and both. False negatives; *false positives*.

3.4 Discussion

Given that the DSM CENTROID method uses information from all of a word's contexts (as captured by the DSM), in contrast to the DISTRIBUTIONAL TEST approach that uses hand-picked tests to reveal behavior relevant to extremeness, it is worth exploring why the former method performs so much better. Here we study the errors made by the two methods when using a 0.5 probability threshold for the classifiers. We focus on the sets of errors that are exclusive to each model, as shown in Table 3; a complete list of errors is available in the data provided in the GitHub repository.

One possibility is that distributional tests will fail to identify extreme adjectives when they only infrequently co-occur with the distributional probes; the fact that an extreme adjective *can* be modified by, e.g., *absolutely* does not entail that these will actually co-occur in the corpus. The false negatives for the DISTRIBUTIONAL TEST method reflect this: it misses out on extreme adjectives that do not frequently co-occur with extreme degree modifiers. We see, for instance, that many extreme adjectives in the domain of SIZE are misclassified due to a low NPMI$_E$ score – a significant error since extreme adjectives of SIZE are often presented as typical members of the class (Cruse, 1986; Morzycki, 2012). The DISTRIBUTIONAL TEST method also produces a considerable number of false positives. This reflects another shortcoming of distributional tests, as it has been noted that extreme degree modifiers can be used with adjectives that are *not* extreme (cf. *absolutely inconvenient*, *totally identical*; Paradis, 2001; Morzycki, 2012).

The cases where DSM CENTROID fails – false negatives and false positives – are mostly of a different nature. Among its errors are many words that have *both* extreme and non-extreme senses (*freezing*, *destitute*, *microscopic*, *obese*, *runaway*, and *major*). For example, *destitute* can mean 'devoid of', and *freezing* can refer to the actual transitioning from liquid to solid, resulting in a non-extreme classification. Conversely, *runaway* is misidentified as extreme due to extreme uses such as *runaway success*. Such polysemies highlight an issue for any method that draws on data from a corpus that is not sense-annotated – recognizing an adjective as extreme depends on the word being predominantly used in the expected sense.

Despite this challenge, our results align with the analysis of extreme adjectives as a distinct adjectival class (Cruse, 1986; Paradis, 2001; Morzycki, 2012). Members of this category can be automatically identified using similarity to an aggregate vector of (known) extreme adjectives, even when those exemplars cross a wide range of semantic domains. Importantly, while Morzycki (2012) argued that extreme adjectives are always felicitous with extreme degree modifiers such as *absolutely*, this does not entail that the association will necessarily manifest, even in a large corpus. Instead, we leverage the broad distributional tendencies captured by DSMs to determine semantic class membership probabilistically.

Our good performance in this task required only a single centroid representation of our semantic class of interest – importantly, it did not require an opposing negative centroid (as in, e.g., An et al., 2018) nor a relation-specific warping of semantic space as in Lu et al. (2019). In future work, we hope to use recent developments in applying influence functions for NLP (e.g., Brunet et al., 2019) to discover the contexts that enable DSMs to represent such abstract properties effectively.

4 Predicting novel intensifiers

Having shown that extremeness can be classified on the basis of similarity within a DSM, we now show that this method is sufficiently informative to guide a separate task – that of modeling the emergence of novel intensifiers. Intensifiers are adverbs such as *staggeringly* in *staggeringly easy*, or *monumentally* in *monumentally wrong*, that give force to the modified adjective without conveying their manner meaning – i.e., saying some task is *staggeringly easy* means that it is extremely easy, not that it is easy in a staggering way. Manner adverbs can gain such an intensifying sense through a continual, and frequent, process of semantic change (Bolinger, 1972; Bennett and Goodman, 2018; Luo et al., 2019). However, it is an open question why certain manner adverbs are more likely to do so than others.

Our hypothesis is that it is the abstract property of extremeness that facilitates this meaning shift – intuitively, *staggeringly easy* and *monumentally wrong* can readily be paraphrased as *off-the-scale easy* and *off-the-scale wrong* because the underlying adjectives (*staggering* and *monumental*) denote extreme values. Thus, an ability to detect extremeness should support an approach to prediction of emergence of intensifier senses: i.e., the adverbs derived from extreme adjectives should be those that are likely to become intensifiers.

We contrast our hypothesis with that of Luo et al. (2019), who propose that intensifier usage can arise with any adverb through a more general process of semantic bleaching (Traugott and Dasher, 2001). Specifically, Luo et al. (2019) argue that when adverbs modify semantically similar adjectives (*conspicuously evident*), the redundancy of the modifying adverb leads to its interpretation as intensifying the content of the adjective, and the actual manner component of the adverb is bleached over time. Luo et al. (2019) capture this insight with a measure (described below) of the within-domain similarity of adverbs to the adjectives that they are found to modify frequently.[8]

We follow Luo et al. (2019) in adopting a semantic bleaching account, but instead propose that it is the abstract property of extremeness that is the focus of the bleaching process: On our account, intensification arises when the particular scale (EMOTION or INTELLIGENCE) of an extreme adverb (an adverb derived from an extreme adjective) is backgrounded, and eventually lost, while the abstract property of extremeness remains the key part of the meaning.

Our approach is compatible with that of Luo et al. (2019), but moreover *explains* why certain adverbs are more likely to modify similar adjec-

---

[8] By "within-domain" we mean a standard topical/domain-level similarity of meaning, such as *conspicuously* and *evident* conveying 'ease of observation'.

Semantic category	Entries
The world > Relative properties > Quantity > Greatness of quantity/ amount/ degree > hugely	**colossally**, **monumentally**, hugeously, thumpingly, pyramidically
In respect of quantity > Greatly/very much > extremely/exceedingly > remarkably/ extraordinarily	**markedly**, **exceptionally**, **noticeably**, **pronouncedly**, **prominently**
Consciously, knowingly > In accordance with truth, truly > in fact, actually	**literally**, **absolutely**, **objectively**, essentially, factually

Table 4: HTE categories with a sample of entries. Bolded words are attested after 1800.

tives in the first place: Adverbs that are derived from an extreme adjective are likely to modify similar adjectives precisely because they are *not* redundant – they contribute the salient meaning of 'extremeness' over and above the expression of the scale already expressed in the modified adjective. Thus we suggest that it is primarily the abstract, cross-domain property of scalar extremeness that drives the emergence of intensifying meanings of adverbs, rather than their patterns of co-occurrence with adjectives that are semantically similar. In this section, we support our claim with a modelling experiment that contrasts the within-domain similarity measure of Luo et al. (2019), with our cross-domain similarity measure for identifying extremeness.

4.1 Methods and materials

To compare these hypotheses, we perform a historical prediction task: predicting the emergence of an intensifier sense of an adverb based on data in the decades prior to the sense's attestation date. Note that this differs from Luo et al. (2019), who performed statistical analyses over time, including both before and after emergence of the intensifying senses. Consequently, we cannot adopt the full dataset of Luo et al. (2019) since it includes adverbs whose intensifier sense emerged prior to 1800 – where historical corpus data is scarce.

Intensifiers. We start with a set of 69 intensifiers R that have their first date of attestation of an intensifier sense after 1800 according to the Historical Thesaurus of English (HTE; Kay et al., 2017). (These include 45 from Luo et al. (2019) that meet this criterion.) These adverbs come from a wide variety of semantic categories, as illustrated in Table 4. We restrict the attestation date to post-1800 so that we can draw on a large historical corpus with sufficient data and robust word embeddings.

Prediction timeframe. Because we aim to predict the *emergence* of novel intensifiers, rather than classify existing ones, the model should not use data that contains the target adverb in its intensifier meaning. In order to use the same time span for all adverbs and include a reasonable quantity of data, we take data for each adverb from the 3 decades prior to its use as an intensifier. That is, for each adverb r attested within decade $d_A(r)$, we use corpus data from the 3 decades $T_r = \{d_A(r) - 3, d_A(r) - 2, d_A(r) - 1\}$.

Matched control adverbs. Luo et al. (2019) presented a set of 178 control adverbs C that did not develop an intensifying meaning. We match each $r \in R$ with the $c \in C$ that has the most similar frequency to r in the time period T_r, and use data for c from the same timeframe T_r.

Historical corpus data. With smaller corpora being too sparse, we use historical data (years 1800–1999) from the Google N-grams corpus (English, version 2, Michel et al., 2011), drawing on the diachronic skip-gram word embeddings of Hamilton et al. (2016), and syntactic annotations from Goldberg and Orwant (2013). Note that the embeddings are formed per-decade, to allow for sufficient training data. In our primary analyses, we use the embeddings of the adjectival bases of the adverbs, due to sparsity of the adverb embeddings themselves. Due to missing embeddings for some of the items, we had to remove 28 items, leaving us with 52 intensifiers and 58 control adverbs.

4.2 Features predicting intensifier emergence

Here we describe the two features we are investigating as predictive of the development of adverbs into intensifiers, our EXTREMENESS feature and the SIMADJMOD feature of Luo et al. (2019).

EXTREMENESS. To test our hypothesis that extreme adverbs are likely to gain an intensifying sense, we adapt the DSM CENTROID measure from Section 3. For each adverb q to be classified, we first find its adjectival basis a_q (e.g., *insane* for *insanely*), and then obtain the embedding $\mathbf{a_q}$ as the average of the diachronic embeddings $\mathbf{a_q(t)}$ for each decade $t \in T_q$. We then obtain the likelihood that a_q is in class c – where $c \in \{\text{Extreme}, \text{Non-extreme}\}$ – based on its proximity to the extremeness centroid. We obtain the extremeness centroid using the extreme adjectives in Section 3, averaging their diachronic embeddings

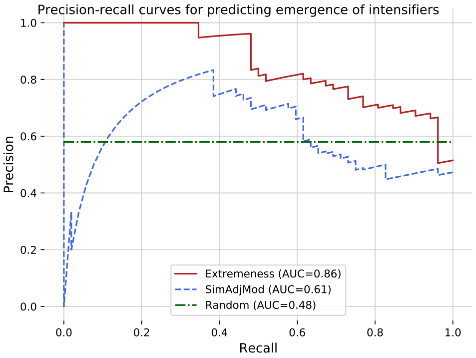

Figure 3: Results for historical prediction task.

over the time period T_q. While this is a dataset of *currently* extreme adjectives, we believe it can be reliably used for the time period covered, as extremeness seems to be a relatively stable property.

SimAdjMod. Luo et al. (2019) formalize their hypothesis with a measure, SimAdjMod, of the similarity between an adverb and the adjectives it modifies. For a query adverb q, the measure is computed as the average semantic similarity between q (e.g., *conspicuously*) and the set of all adjectives modified by q (e.g., *evident*). We adapt this measure with an approach analogous to the Extremeness measure: we find the word embeddings of the adjective a_q that is the basis of q and of all adjectives modified by q within T_q. Then, SimAdjMod(q) is computed as the odds-weighted average of the cosine similarity of $\mathbf{a_q}$ with each modified adjective's embedding (see Luo et al. (2019) for details on the weighting function).

It is worth noting that using adjective embeddings contrasts with Luo et al. (2019), who use the adverb embeddings themselves. We adopted our approach due to the absence of many adverbs in the diachronic embeddings. In their statistical analyses (in contrast to our prediction task), Luo et al. (2019) can use embeddings from a 150-year time span, reducing the sparsity of adverb embeddings. We perform an additional experiment in which we follow their set-up more closely, reported in Appendix B. As in our main results below, the Extremeness feature comes out as a strong explanatory factor, although its improvement over the SimAdjMod feature is much greater in the core historical prediction task. We discuss the implications in Appendix B.

4.3 Results and Discussion

We fit a separate logistic regression classifier for each feature, Extremeness and SimAdjMod, and use leave-one-out cross-validation. We report precision-recall curves using the $p(c|q)$ estimates from the regression models (where $c \in \{\text{Intensifier}, \text{Control}\}$), and perform error analysis using a 0.5 threshold on $p(c|q)$. The results are presented in Figure 3. We observe that both measures perform well above chance, with the Extremeness feature (AUC $= 0.86$) outperforming the SimAdjMod feature (AUC $= 0.61$).[9] To understand the difference in performance between the two features, we turn to an analysis of the errors.

One situation that should distinguish the use of Extremeness from SimAdjMod is in predicting the emergence of intensifier meanings for adverbs that are infrequent: an uncommon extreme adverb will still be extreme, but its low frequency will prevent it from frequent co-occurrence with similar adjectives. We find evidence for this in the errors: Extremeness, but not SimAdjMod, correctly predicts the emergence of an intensifier meaning for the infrequent adverbs *monumentally*, *colossally*, *frighteningly*, *staggeringly*, and *thunderingly*. This supports our hypothesis that it is not the frequent co-occurrence (with similar adjectives) that primarily drives intensifier emergence, but the semantic property of extremeness.

However, while many novel intensifiers are based on adverbs derived from extreme adjectives, not all of them are; adverbs that have an epistemic function (e.g., *actually*, *really*, *honestly*) also form a frequent source of intensifiers (Bolinger, 1972; Traugott and Dasher, 2001). We indeed observe that the Extremeness feature fails to predict the emergence of intensifiers such as *honestly* and *prominently*, cases of adverbs that were not derived from extreme adjectives, while the SimAdjMod feature captures these cases.

Overall, the results of this experiment support our claim: the cross-domain semantic property of extremeness is highly effective in predicting whether an adverb will gain an intensifying function. While the account of Luo et al. (2019) provides substantial empirical coverage of the pattern of historical development, it does not explain why particular adverbs should modify similar adjectives in the first place. Our account subsumes theirs

---

[9] Including both features in the classifier does not improve performance over the Extremeness feature alone.

by showing that if a manner adverb frequently co-occurs with a similar adjective, the adverb is likely derived from an extreme adjective (with exceptions, such as the epistemic adverbs); the adverb is then easier to re-interpret as an intensifier by maintaining the feature of extremeness while bleaching its domain-specific semantics.

5 Conclusion

In this paper, we looked at the question of whether distributional semantic models (DSMs) can capture abstract semantic properties of word classes. While it is well known that such models can capture within-domain similarity (e.g., *tall* being similar to *high* and *large*, all pertaining to the scalar attribute of SIZE), here we study whether DSMs also encode abstract similarities that hold *across* domains.

Our case study focuses on extremeness (Cruse, 1986; Morzycki, 2012; Paradis, 2001), the adjectival property of expressing values so high or low that they can be considered "off the scale" (Morzycki, 2012). Extreme adjectives thus share the abstract property of "extremeness" on a scale, independently of the particular scale involved. Our experiments demonstrate that the abstract property of extremeness can be identified using similarity of an adjective to an extreme "prototype" in a DSM, thus supporting the claim that the similarity structure of a DSM can encode such cross-domain classes.

We believe our study contributes to the rapprochement of computational methods and linguistic theory (cf. Boleda, 2020): both the traditional application of distributional tests, as well as the more open-ended application of DSMs, rely on distributional patterns. Since those patterns are more opaque in a DSM, a fruitful next step would be to reverse-engineer the distributional patterns that are indicative of abstract semantic class membership and thereby 'give back' an interpretable set of class membership indicators to semantic theory.

6 Acknowledgments

This research was supported by NSERC grant RGPIN-2019-06917 to Barend Beekhuizen and by NSERC grant RGPIN-2017-06506 to Suzanne Stevenson.

References

Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. 2018. SemAxis: A lightweight framework to characterize domain-specific word semantics beyond sentiment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247.

Erin D. Bennett and Noah D. Goodman. 2018. Extremely costly intensifiers are stronger than quite costly ones. *Cognition*, 178:147–161.

Gemma Boleda. 2020. Distributional semantics and linguistic theory. *Annual Review of Linguistics*.

Gemma Boleda, Sabine Schulte im Walde, and Toni Badia. 2012. Modeling regular polysemy: A study on the semantic classification of catalan adjectives. *Computational Linguistics*, 38(3):575–616.

Dwight Bolinger. 1972. *Degree words*. The Hague: Mouton.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. 2019. Understanding the origins of bias in word embeddings. In *Proceedings of the 36th International Conference on Machine Learning*, pages 803–811. PMLR.

Anne Cocos, Veronica Wharton, Ellie Pavlick, Marianna Apidianaki, and Chris Callison-Burch. 2018. Learning scalar adjective intensity from paraphrases. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1752–1762.

D Alan Cruse and Pagona Togia. 1996. Towards a cognitive model of antonymy. *Lexicology*, 1(1):113–141.

David Alan Cruse. 1986. *Lexical semantics*. Cambridge University Press.

Mark Davies. 2009. The 385+ million word corpus of contemporary american english (1990–2008+): Design, architecture, and linguistic insights. *International Journal of Corpus Linguistics*, 14(2):159–190.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 241–247.

Gabriel Grand, Idan Asher Blank, Francisco Pereira, and Evelina Fedorenko. 2018. Semantic projection: recovering human knowledge of multiple, distinct object features from word embeddings. *arXiv preprint arXiv:1802.01241*.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501.

Geoff Hollis and Chris Westbury. 2016. The principals of meaning: Extracting semantic dimensions from co-occurrence models of semantics. *Psychonomic Bulletin & Review*, 23(6):1744–1756.

Geoff Hollis, Chris Westbury, and Lianne Lefsrud. 2017. Extrapolating human judgments from skip-gram vector representations of word meaning. *The Quarterly Journal of Experimental Psychology*, 8(70):1603–1619.

Janellen Huttenlocher, E Tory Higgins, and Herbert H Clark. 1971. Adjectives, comparatives, and syllogisms. *Psychological Review*, 78(6):487.

Dan Jurafsky and James H Martin. 2014. Speech and language processing. vol. 3.

C. Kay, J. Roberts, M. Samuels, and I. Wotherspoon. 2017. *The Historical Thesaurus of English, version 4.21*. University of Glasgow, Glasgow, UK.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Daniel Lassiter. 2017. *Graded modality: Qualitative and quantitative perspectives*. Oxford University Press.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago Press.

Hongjing Lu, Ying Nian Wu, and Keith J Holyoak. 2019. Emergence of analogy from relation learning. *Proceedings of the National Academy of Sciences*, 116(10):4176–4181.

Yiwei Luo, Dan Jurafsky, and Beth Levin. 2019. From insanely jealous to insanely delicious: Computational models for the semantic bleaching of english intensifiers. In *Proceedings of the 1st International Workshop on Computational Approaches to Historical Language Change*, pages 1–13.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(85):2579–2605.

Gerard de Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1:279–290.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Neural and Information Processing System (NIPS)*.

Marcin Morzycki. 2012. Adjectival extremeness: Degree modification and contextually restricted scales. *Natural Language and Linguistic Theory*, 30(2):567–609.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Carita Paradis. 2001. Adjectives and boundedness. *Cognitive Linguistics*, 12(1):47–65.

Francisco Pereira, Samuel Gershman, Samuel Ritter, and Matthew Botvinick. 2016. A comparative evaluation of off-the-shelf distributed semantic representations for modelling behavioural data. *Cognitive Neuropsychology*, 33(3-4):175–190.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (too many) problems of analogical reasoning with word vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 135–148.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 726–730.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 117–122.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of* *the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307.

Raksha Sharma, Arpan Somani, Lakshya Kumar, and Pushpak Bhattacharyya. 2017. Sentiment intensity ranking among adjectives using sentiment bearing word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 547–552.

Vera Sheinman, Christiane Fellbaum, Isaac Julien, Peter Schulam, and Takenobu Tokunaga. 2013. Large, huge or gigantic? identifying and encoding intensity relations among adjectives in wordnet. *Language resources and evaluation*, 47(3):797–816.

Elizabeth Closs Traugott and Richard B Dasher. 2001. *Regularity in semantic change*. Cambridge University Press.

Akira Utsumi. 2020. Exploring what is encoded in distributional word vectors: A neurobiologically motivated analysis. *Cognitive Science*, 44(6).

Bryan Wilkinson and Oates Tim. 2016. A gold standard for scalar adjectives. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2669–2675.

Aotao Xu, Jennifer Stellar, and Yang Xu. 2020. Prototype theory and emotion semantic change. In *Proceedings CogSci*.

A Extremeness classification: supplementary analysis

In this supplementary analysis, we report on an experiment in which we control for the potential confound described at the end of Section 3.3; that is: some training set items frequently share domain-specific features with the held out test item (e.g., *massive* and *colossal* are in the training set while *huge* is held out). To validate that the DSM Centroid method is detecting the abstract property of extremeness, rather than simply tapping into such domain-specific similarities, we set up the evaluation procedure in a way that minimizes the possibility that domain-specific similarities explain the results. We do so by aiming to exclude adjectives from the training set that are in the same domain as a test adjective.

A.1 Methods and materials

Because we do not have ground-truth domain labels for our extreme adjectives (e.g., $\{huge, colossal, mammoth\} \in$ SIZE), we require another method for grouping the adjectives into domains. As an approximation of domain groupings, we use an unsupervised clustering algorithm on the extreme adjectives, expecting that items belonging to the same cluster will belong to the same domain. With these k domain-based clusters, we can then train on $k-1$ of the clusters while testing on the held-out cluster.

Concretely, we perform a variant of k-fold cross-validation, called leave-one-cluster-out cross-validation, following Utsumi (2020), where the folds are determined through k-means clustering of our dataset of extreme adjectives. For binary classification of extreme adjectives, we train the DSM Centroid and Distrib. test methods, described in Section 3.2, on extreme adjectives in $k-1$ of the clusters as well as their frequency-paired non-extreme adjectives; see Section 3.1 for details of this pairing. We then test our binary classifiers on the extreme adjectives in the k-th cluster along with their paired non-extreme adjectives. We repeat this procedure k times so that each cluster is tested once.

A.2 Results

The leave-one-cluster-out cross-validation procedure is dependent on the number of clusters k used for k-means clustering. We perform the procedure for a wide range of $k \in [2, \ldots, 12]$ to ensure that the results are robust to the choice of k. Table 5 (below) displays subsets of the clusters obtained with k-means clustering for $k = 5$, qualitatively demonstrating that the clusters are centered on domain-specific features. The cross-validation results on the full range of k values are shown in Fig. 4.

We can see that the DSM Centroid performance here (AUC $\in [0.78, \ldots, 0.88]$) is lower than in Section 3.3 (AUC=0.97), indicating that within-domain similarity was partly responsible for the result in that set-up. However, even in this more controlled set-up, the DSM Centroid method robustly outperforms the Distributional Test method (AUC $\in [0.67, \ldots, 0.74]$, which is similar to the AUC of 0.71 for this method in Section 3.3). We take the results in this controlled analysis as further evidence that the cross-domain property of extremeness can be detected from DSM vectors.

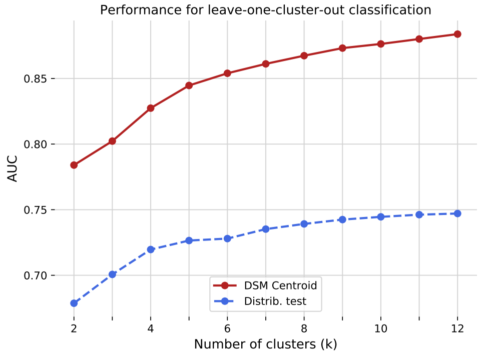

Figure 4: Results for extremeness classification using the leave-one-cluster-out cross-validation procedure. We vary the number of clusters from $k \in [2, \ldots, 12]$.

B Intensifier prediction: supplementary analysis

In Section 4, we used a historical prediction task to test two contrasting accounts of how intensifiers emerge in the lexicon. Specifically, we developed classifiers, based on diachronic DSMs, to predict whether an adverb would acquire an intensifying sense based on historical corpus data *prior* to the sense's attestation date. By restricting to corpus data prior to attestation dates, we ensured that the DSM vectors reflected manner adverb usage (*monumentally sized*) rather than intensifier usage (*monumentally wrong*). That is, we minimized the risk that the DSM vectors smuggled in information of whether an adverb was already an intensifier.

This restriction, however, also results in two data

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
ecstatic	brainless	gargantuan	penniless	mesmerizing
phenomenal	amateurish	miniscule	destitute	resplendent
exceptional	moronic	mammoth	miserable	immaculate
terrific	inane	colossal	filthy	breathtaking
outstanding	idiotic	monstrous	obese	brilliant

Table 5: Samples of clusters from k-means clustering on extreme adjectives with $k = 5$.

sparsity problems. First, there are relatively few adverbs that gained an intensifying sense after the 1800s – the time period for which we have historical corpus data – so we were required to use a smaller sample of intensifiers. Second, many adverbs gained intensifying senses towards the earlier parts of the 19th century, for which historical corpora are significantly smaller than in later decades. As described in Section 4, the latter issue required us to compute the SimAdjMod measure differently than Luo et al. (2019): We used embeddings of the adjectival basis (*insane*) rather than the adverb itself (*insanely*), since many adverbs were missing in the historical embeddings.

In this section, we perform the prediction task introduced in Section 4 without restricting to corpus data prior to attested dates of intensifier senses. While lifting this restriction weakens the explanatory account of how intensifiers emerge in the lexicon (due to the aforementioned methodological concern), it alleviates the problem of data sparsity and aligns more closely with the experimental set-up of Luo et al. (2019), thereby making the results more directly comparable. This experiment thus serves as a follow-up test for our hypothesis that EXTREMENESS is a strongly predictive feature of whether intensifying meanings emerge for manner adverbs.

The experiment here is similar to the one reported in Section 4, but there are three significant differences, all introduced to align better with the setting of Luo et al. (2019): First, we use the majority of intensifiers ($N = 185$ out of 250) and control adverbs ($N = 152$ out of 178) published by Luo et al. (2019) – specifically, we use all of their adverbs that have available word embeddings, as opposed to only intensifiers attested within the 1800s to the present. Second, we use a prediction timeframe of $T = \{1850, 1860, \ldots, 1990\}$ for all queries, as opposed to a timeframe of 3 decades parameterized by an intensifier's attested date. Third, we use the adverb embeddings in computing the SIMADJMOD measure, as opposed to the embeddings of the adverbs' adjectival bases used in Section 4.

B.1 Methods and materials

Corpus. The binary classification – as having an intensifier sense or not – is conducted with corpus data spanning 15 decades from the 1850s to the 1990s from the Google N-grams and syntactic N-grams corpora. We used diachronic skip-gram word embeddings as in Section 4.

Evaluation dataset. Our evaluation dataset consists of the 185 intensifiers and 152 control-adverbs from the Luo et al. (2019) dataset that were left after discarding, similarly to their modelling set-up, 65 intensifiers and 26 control-adverbs with missing word embeddings.

B.2 Features for classification of intensifiers

For each query adverb q, we compute the two features (EXTREMENESS and SIMADJMOD) as follows:

EXTREMENESS. We calculate the cosine similarity of the adjectival basis embedding $\mathbf{a_q(t)}$ to the extremeness centroid $\mathbf{E_t}$ for every decade $t \in T = \{1850, 1860, \ldots, 1990\}$. As in Section 4, the extremeness centroid $\mathbf{E_t}$ is calculated as the averaged diachronic embeddings of the extreme adjectives introduced in Section 3. This provides us with a measure of EXTREMENESS for each decade $t \in T$ that we then average to get a single measure of EXTREMENESS for q.

SIMADJMOD. We calculate the odds-weighted[10] average similarity of the adverb embedding $\mathbf{q_t}$ to the embeddings of the adjectives $\mathbf{a_t}$, where a was modified by q in decade t. We perform this computation for every $t \in T = \{1850, 1860, \ldots, 1990\}$, giving us a measure of SIMADJMOD for every

---

[10] See Luo et al. (2019) for details of the weighting function.

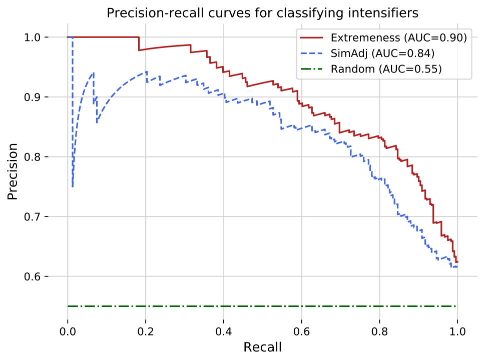

Figure 5: Results for classifying an adverb as an intensifier vs. control.

decade $t \in T$. We then average these per-decade SimAdjMod values to get a single measure of SimAdjMod for q. Rather than using the embedding of the *adjectival basis* of q for $\mathbf{q_t}$, as we did in Section 4 for data sparsity reasons, here we used the diachronic embeddings of the *adverb* q. In doing so, we mirror the operationalization of Luo et al. (2019) for this feature exactly.

B.3 Classification results

We fit a separate logistic regression classifier for each of the two features and use leave-one-out cross-validation. Fig. 5 shows that Extremeness achieves better performance (AUC $= 0.90$) than SimAdjMod (AUC $= 0.84$). The difference between the two features reported here is, however, much smaller than in the experiment in Section 4. Interestingly, while the AUC for Extremeness is comparable between the two experiments, the SimAdjMod feature does much better in the current set-up. We speculate that this is due to the latter feature having access to the adverbial vectors for decades in which the adverb has already obtained an intensifying meaning: a more “bleached” vector is more likely to be similar to many adjectives than a vector that has a more unique contextual signature. Further research is required on the influence of semantic bleaching on semantic space representations to test this speculation. We leave this topic for future work.

Despite the smaller difference in performance, however, we still find salient differences between the predictions made by the two models: We observe a similar pattern of errors as in Section 4 when setting a classification threshold at $P(c = \text{Intensifier}|q) = 0.5$. Specifically, the Extremeness feature is able to correctly classify intensifiers that are relatively infrequent (*defiantly, startlingly, unequivocally*), providing further evidence that extremeness leads to the acquisition of intensifying meaning. Again, we observe that the account formalized in the SimAdjMod measure correctly predicts adverbs with an epistemic function (*evidently, prominently, noticeably*) to acquire an intensifying meaning while the Extremeness feature does not. Overall, we find results in accordance with those of Section 4: the cross-domain semantic property of Extremeness is very effective in predicting whether an adverb will gain an intensifying function.

Effective Attention Sheds Light On Interpretability

Kaiser Sun[*] **Ana Marasović**[†*]
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington
[†]Allen Institute for AI
Seattle, WA, USA
`huikas@cs.washington.edu`, `anam@allenai.org`

Abstract

An attention matrix of a transformer self-attention sublayer can provably be decomposed into two components and only one of them (*effective attention*) contributes to the model output. This leads us to ask whether visualizing effective attention gives different conclusions than interpretation of standard attention. Using a subset of the GLUE tasks and BERT, we carry out an analysis to compare the two attention matrices, and show that their interpretations differ. Effective attention is less associated with the features related to the language modeling pretraining such as the separator token, and it has more potential to illustrate linguistic features captured by the model for solving the end-task. Given the found differences, we recommend using effective attention for studying a transformer's behavior since it is more pertinent to the model output by design.

1 Introduction

Attention mechanism (Bahdanau et al., 2015) is an essential component of many NLP models, including those that are built on the ubiquitous transformer architecture (Vaswani et al., 2017). As a result, visualizing attention weights is a widely used technique to interpret models' behavior (Belinkov and Glass, 2019). Despite that, the validity of this analysis method is a subject undergoing intense discussion and study in NLP (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019; Serrano and Smith, 2019; Moradi et al., 2019; Mohankumar et al., 2020; Tutek and Snajder, 2020, *i.a.*).

Related to this discussion, Brunner et al. (2020) show that, under mild conditions, the attention matrix of a transformer self-attention sublayer can be written as a sum of two components. One of them is irrelevant for the model output because its product with the value matrix is zero. They term the other component as *effective attention* (formally defined in §2). We study whether effective attention gives interpretations that differ from conclusions we get by analyzing standard attention. If this is the case, interpretation of effective attention is better suited for studying transformers' internals because it is more pertinent to the model output by design.

Brunner et al. (2020) briefly discuss this by comparing standard and effective attention matrices from a single BERT head (Devlin et al., 2019) for one example. They observe that: (i) standard attention is largely concentrated on the delimiter tokens ([SEP], [CLS]) or on near-diagonal elements; (ii) effective attention is more dispersed; (iii) effective attention disregards the delimiters. They stress that we should not extrapolate too much from these observations since they are based on a single example, and that further research is needed on this topic.

In this work, we aim to reliably answer whether effective attention disregards the [SEP] and [CLS] tokens, and if so, are effective attention weights dispersed to linguistic features? To address these questions, we embrace the methodology for a quantitative analysis of the attention patterns produced by individual transformer heads proposed by Kovaleva et al. (2019). We carry out their experiments on a subset of the GLUE tasks with BERT's standard and effective attention. We show that effective attention "ignores" [SEP] and punctuation symbols (§3.1, §3.2), but not [CLS] (§3.2), and that it highlights end-task features instead (§3.1, §3.2, §3.3).[1]

2 Background: Effective Attention

Each transformer layer consists of multi-head self-attention and feedforward sublayers (Vaswani et al., 2017, see Appendix A). Brunner et al. (2020) show that the **standard attention** matrix A can be decomposed into two components, if a mild condition

---

[1] Our code is available at https://github.com/KaiserWhoLearns/Effective-Attention-Interpretability

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4126–4135
August 1–6, 2021.

is satisfied. Specifically, if the left nullspace of the value matrix V:

$$\mathrm{LN}(V) := \{x^\top \in \mathbb{R}^{1 \times d_s} | x^\top V = 0\},$$

is not trivial (contains vectors other than $\vec{0}$). This is satisfied when the maximum input sequence length is larger than the value matrix dimension (see Appendix A). The two components are: the component in the left nullspace of V ($A^{\parallel}$) and the component orthogonal to the nullspace ($A^{\perp}$). Notably, $A^{\parallel}$ does not contribute to the output of the self-attention sublayer:

$$AV = (A^{\parallel} + A^{\perp})V = \vec{0} + A^{\perp}V = A^{\perp}V. \quad (1)$$

The **effective attention** matrix is defined as $A^{\perp}$. If visualizations of standard and effective attention differ, interpretation of effective attention is an accurate interpretation because effective attention is what contributes to the model output (per Eq. 1).

We explain how to compute $A^{\perp}$ since that was not described in Brunner et al. (2020). We first compute the singular value decomposition (SVD) of the value matrix $V = U\Sigma W^T$. The rows of U that correspond to singular values equal to zero span $\mathrm{LN}(V)$:

$$\mathrm{LN}(V) = \mathrm{span}\{u_1, \ldots, u_k\},$$

where k is the number of singular values that equal zero. We project each row a_i of the attention matrix $A \in \mathbb{R}^{d_s \times d_s}$ to $\mathrm{LN}(V)$ to construct a projection of the *matrix* A to $\mathrm{LN}(V)$:

$$\mathrm{P}_{\mathrm{LN}(V)}(a_i) = \sum_{j=1}^{k} \langle a_i, u_j \rangle u_j, \forall i \in \{1, \ldots, d_s\},$$

$$\mathrm{P}_{\mathrm{LN}(V)}(A) = [\mathrm{P}_{\mathrm{LN}(V)}(a_1), \ldots, \mathrm{P}_{\mathrm{LN}(V)}(a_{d_s})]^\top,$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product. Finally, effective attention equals to:

$$A^{\perp} := A - \mathrm{P}_{\mathrm{LN}(V)}(A).$$

Effective attention is not guaranteed to be a probability distribution as some of its weights might be negative and larger than 1.

We observe that effective attention is slower to compute due to the SVD decomposition of V for each out of 144 BERT-base heads, and additional matrix multiplications (Table 3; §B). If speed is bottleneck, we recommend doing quantitative analyses with effective attention on a subset of the dev set. For qualitative analyses, common practice is already to select a subset for a manual analysis.

Dataset	Task	|Train|	|Test|
RTE	NLI	2.5K	3K
MRPC	paraphrase identification	3.7K	1.7K
QNLI	QA as NLI	105K	5.4K
SST-2	binary sentiment classification	67K	1.8K
STS-B	sentence similarity	7K	1.4K

Table 1: Specifications of the datasets.

3 What Does Effective Attention Reveal?

We compare visualizations of standard and effective attention following the methodology for analysis of the attention patterns (Kovaleva et al., 2019). We carry out our analyses using five English-language datasets in the GLUE benchmark (Wang et al., 2019): RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), MRPC (Dolan and Brockett, 2005), QNLI (Rajpurkar et al., 2016; Wang et al., 2019), SST-2 (Socher et al., 2013), and STS-B (Cer et al., 2017).[2] See Table 1 for their specifications. For each dataset, we train BERT-base with standard attention, a batch size of 8, maximum sequence length of 128, and 3 training epochs.[3] For analyzing effective attention, we replace standard with effective attention at the test time.

3.1 Classification of Attention Patterns

In this section, we start studying whether effective attention disregards the delimiter tokens.

The visualizations of attention matrices exhibit patterns (Clark et al., 2019; Vig and Belinkov, 2019). Kovaleva et al. (2019) identified five frequently occurring pattern categories:

- vertical (associated with the *delimiters tokens*)
- diagonal (either syntactic features between neighbouring words in the English language or the previous/following token attention coming from the language modeling pretraining)
- vertical + diagonal
- block (intra-sentence attention for the tasks with two distinct sequences; potentially encodes semantic and syntactic information)
- heterogeneous (as "block", more likely to capture interpretable linguistic features).

They annotated 400 BERT's attention matrices using these categories, and used them to train a

---

[2] We omit larger datasets (QQP, MNLI), due to the limit of our computation budget (a single Nvdia GTX1070 with 8GB memory), and CoLA/WNLI following Kovaleva et al. (2019).

[3] All other hyperparameters are set to default values in the transformers library (Wolf et al., 2020).

Task	Attention	B	D	V+D	H	V
RTE	Standard	4.50	7.40	15.20	45.10	27.90
	Effective	32.60	12.80	2.80	40.30	11.50
MRPC	Standard	3.40	10.20	14.90	39.80	31.80
	Effective	25.50	17.40	3.60	40.40	13.00
QNLI	Standard	4.70	7.40	15.20	45.10	27.90
	Effective	29.30	15.80	3.40	46.40	5.10
SST-2	Standard	38.50	6.10	0.00	37.80	17.60
	Effective	33.80	11.50	0.80	39.40	14.60
STS-B	Standard	4.00	8.20	1.80	50.40	35.50
	Effective	36.00	10.30	0.60	39.40	13.60

Table 2: Estimated percentage of the attention patterns (§3.1): block (B), diagonal (D), vertical + diagonal (V + D), heterogeneous (H), vertical (V). Effective attention exhibits different patterns than standard attention, i.e., less vertical patterns (associated with delimiter tokens) and more block patterns (associated with task features).

ConvNet for pattern classification of 1K random test set attention matrices. We replicate their results for standard attention (using their code), and classify effective attention matrices for a comparison.[4]

Results Table 2 (Fig. 4 in Appendix B) shows a drop in the percentage of the "vertical" and "vertical + diagonal" patterns when we replace the standard with effective attention. Since the vertical patterns are associated predominantly with attention to the delimiters tokens, this result supports the hypothesis that effective attention disregards the delimiter tokens. Moreover, although the amount of "heterogeneous" patterns did not change notably, the amount of "block" and "diagonal" patterns increased. This suggests that we are better positioned to find end-task linguistic features captured by the model by visualizing effective attention.

As an illustration, Figure 2 presents the attention matrices for one sentence from one attention head. In this example, effective attention highlights all mentions of the noun "antibiotics" that the adjective "new" modifies and that is also the object of the preposition "against", instead of giving prominence to the [SEP] token as standard attention.

3.2 Delimiter Tokens vs. Linguistic Features

We showed that the "vertical" pattern, *associated* with the delimiter tokens, is less dominant with effective attention (§3.1). To verify that both delimiter tokens are indeed less relevant with effective attention, following Kovaleva et al. (2019), we re-

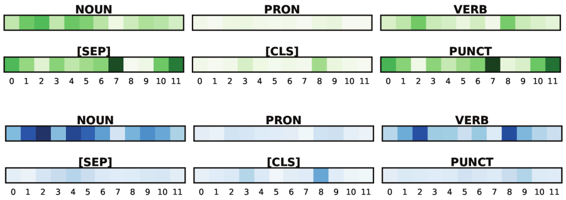

(a) RTE

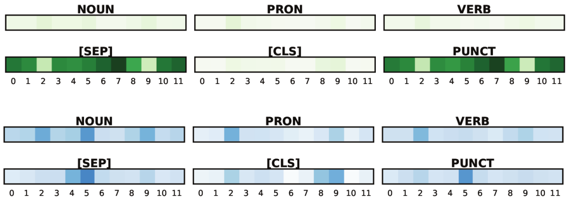

(b) MRPC

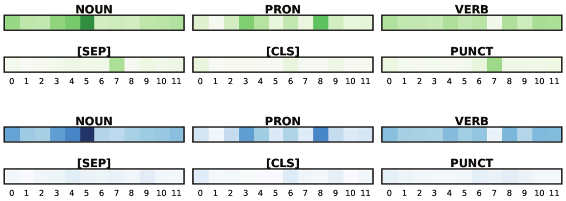

(c) QNLI

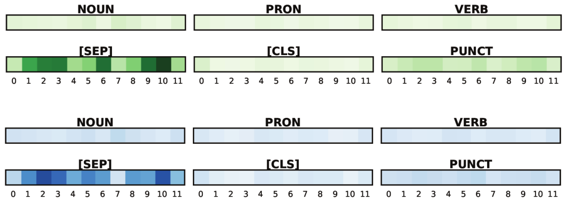

(d) SST-2

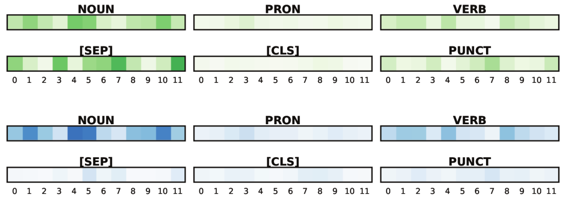

(e) STS-B

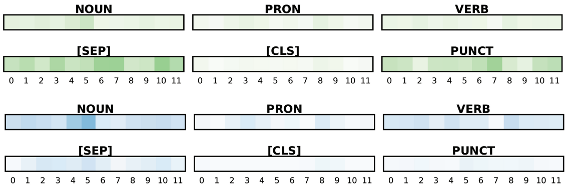

(f) AVERAGE OVER TASKS

Figure 1: Effective attention "pays less attention" to [SEP] and punctuation. Per-task and per-head (0–11) attention when processing [CLS] in the final layer, averaged over test set. The darker colors correspond to larger attention values. The green plots (two upper rows in subfigures) illustrate standard, and blue plots (two lower rows in subfigures) effective attention.

---

[4] We thank the authors for sharing their code and model weights for this experiment.

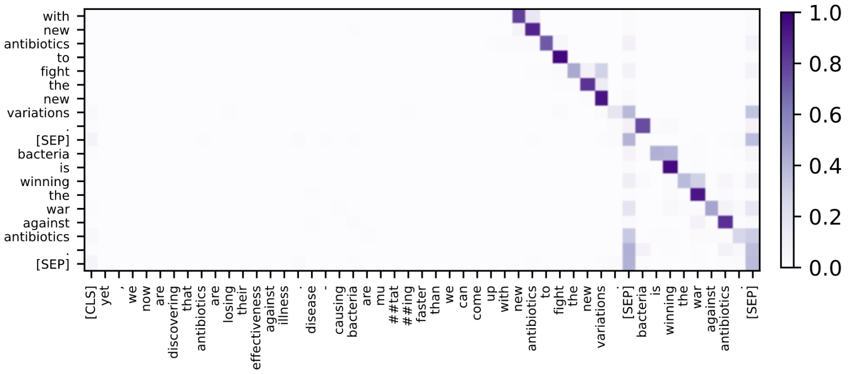

(a) Standard attention

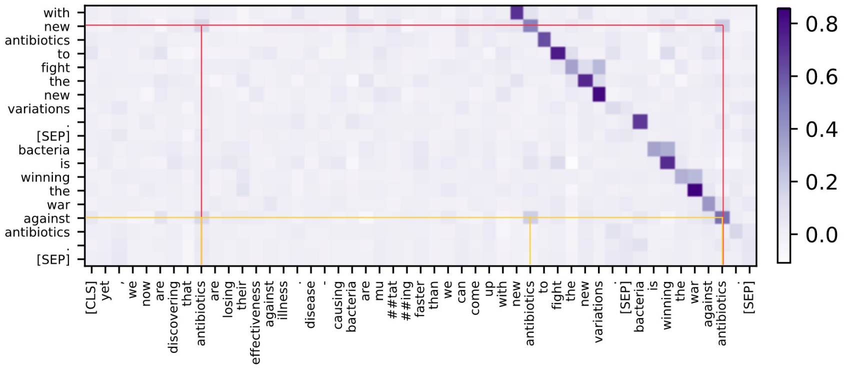

(b) Effective attention

Figure 2: Visualizations of standard and effective attention from one head for one example from the RTE dataset (recognizing textual entailment). Only the last few rows are visible; see the full version in Fig. 7 (Appendix §B).

port the standard and effective attention weights of specific token types when processing the [CLS] token in the final layer. Namely, the attention weights of linguistic features (nouns, pronouns, verbs), the delimiter tokens ([SEP], [CLS]), and punctuation symbols that are conceptually similar to [SEP].[5]

Results Figure 1 shows that [SEP] is among the two most relevant features for all tasks except QNLI according to standard attention (upper two rows in each subfigure, colored green). For all but one task (SST-2), it loses its dominance with effective attention and its weights are apparently shifted to linguistic features. This is also the case for punctuation symbols. This result shows that the [SEP] token and punctuation symbols are not as important for understanding how the model solves the end-task as standard attention suggests.

We observe that [CLS] is attended similarly with effective and standard attention, contrary to what Brunner et al. suggested. To rule out this is because we plot the attention assigned to [CLS] when processing [CLS], we report the attention assigned to [CLS] when processing other input words (regardless of their type) in Fig. 5 in Appendix B. Again, we do not observe differences between standard and effective attention, unlike for [SEP] (Fig. 6 in §B). These results confirm the hypothesis of Brunner et al. that effective attention disregards [SEP], but not [CLS] as they also hypothesized. Notably, [SEP] is associated with the LM pretraining and [CLS] only with the task-specific finetuning.

3.3 Effects of Task-Specific Finetuning

To provide our final evidence that effective attention captures end-task features, we investigate how attention changes with finetuning layer-wise; again following Kovaleva et al. (2019). They calculate the cosine similarity between pretrained and finetuned flattened attention matrices. The layers that change the most, encode most task-specific features. To reiterate, effective attention is the part of standard attention that contributes to the model output (Eq. 1; §2), and we showed that it is less associated with the pretraining feature [SEP] and more with linguistic features (§3.1, §3.2). Thus, changes of standard attention from task-specific finetuning

---

[5] If there are multiple tokens of the same type in the input, we use the one with the maximum weight. If a word consists of the multiple subtokens, we use the weight of the first subtoken.

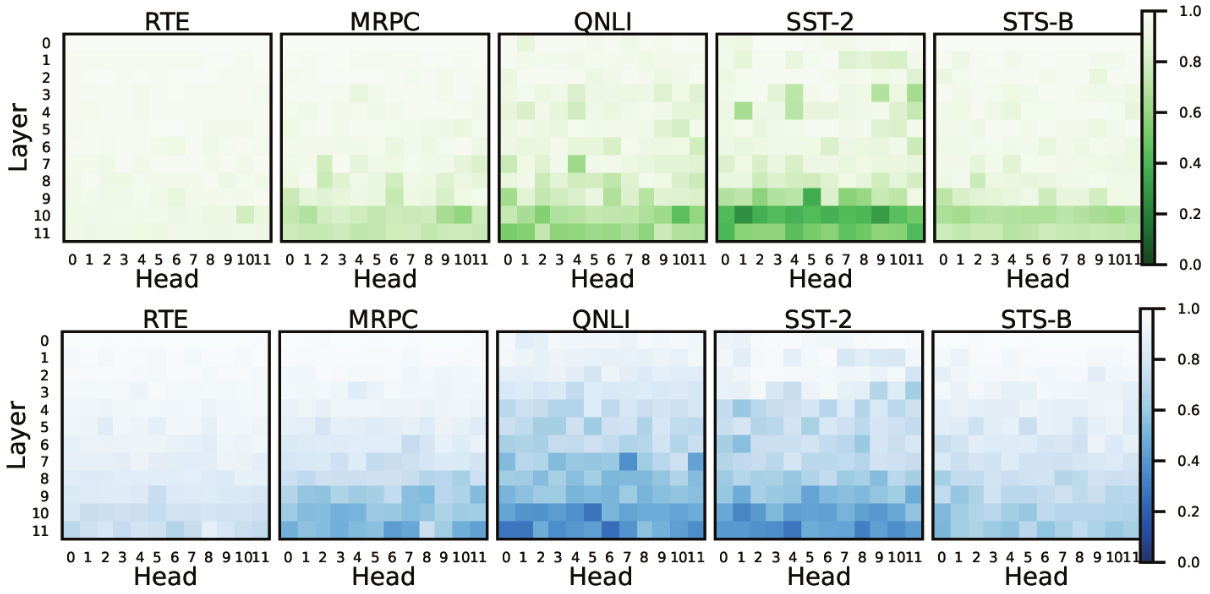

Figure 3: Per-task cosine similarity between the pretrained and finetuned attention weights for selected GLUE tasks, calculated across layers and heads. The darker colors corresponding to larger absolute attention weights. The top (green) figure is computed with the standard attention, and the bottom (blue) figure with the effective attention.

should be the product of changes of effective attention, and the outcome of this analysis should be the same, regardless of the attention "type".

Results As expected, we come to the same conclusion with effective attention as Kovaleva et al. did with the standard: the last two layers change the most with finetuning (Fig. 3). This soundness check suggests once again that effective attention is the component of standard attention that manifests end-task features.

4 Conclusions

We study whether effective attention, the part of the transformer attention matrix that does not get canceled out with the value matrix, gives different interpretations than standard attention. We present a comparison of the two attentions and show that they differ in weights assigned to delimiter tokens such as [SEP] and punctuation marks, but not [CLS] as it was previously thought. Instead, effective attention gives more weight to linguistic features. Given the differences, and that effective attention is more pertinent to the model output by design, we urge to use it for studying transformers' internals.

As an alternative to analyzing attention *weights*, Kobayashi et al. (2020) propose anayzing the norm of vectors produced by multiplying the outputs of the value matrix with the attention weights. Following the experimental setting of Clark et al. (2019), i.e., by analyzing 992 sequences extracted from Wikipedia, their norm-based analysis also shows that the contributions of [SEP] and punctuations are actually small. However, unlike us, they report the same observation for [CLS]. Future work might consider a more formal study between the norm-based analysis and effective attention, especially since the norm-based analysis could circumvent the problem of costly SVD.

Acknowledgments

The authors thank Noah A. Smith, members of Noah's ARK, as well as anonymous reviewers for their helpful feedback, and Olga Kovaleva for sharing the code and model weights for classification of attention patterns.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *the International Conference on Learning Representations*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo

Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Gino Brunner, Y. Liu, Damián Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On Identifiability in Transformers. In *the International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Akash Kumar Mohankumar, Preksha Nema, Sharan Narasimhan, Mitesh M. Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2020. Towards transparent and explainable attention models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4206–4216, Online. Association for Computational Linguistics.

Pooya Moradi, Nishant Kambhatla, and Anoop Sarkar. 2019. Interrogating the explanatory power of attention in neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 221–230, Hong Kong. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Martin Tutek and Jan Snajder. 2020. Staying true to your word: (how) can attention become explanation? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 131–142, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *the International Conference on Learning Representations*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Background: On The Rank Of The Value Matrix

The output Z of an individual self-attention head is given by:

$$
\begin{aligned}
Q &= Z_{l-1}W^Q \in \mathbb{R}^{d_s \times d_q} \\
K &= Z_{l-1}W^K \in \mathbb{R}^{d_s \times d_k} \\
V &= Z_{l-1}W^V \in \mathbb{R}^{d_s \times d_v} \\
A &= \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{d_s \times d_s} \\
Z &= AV \in \mathbb{R}^{d_s \times d_v},
\end{aligned}
$$

where d_s is the maximum length of the input sequence (in number of subtokens), Z_{l-1} is the output of the previous transformer layer, W^Q, W^K, W^V are the query, key, and value *weight* matrices, respectively. For BERT-base, $d_q = d_k = d_v = 64$, $n_{\text{heads}} = 12$, $d_s = 512$, and $d_v \cdot n_{\text{heads}} = 768$.

Brunner et al. (2020) show that the upper bound of the rank of the value matrix V is given by:

$$
\begin{aligned}
\mathrm{rank}(V) &= \mathrm{rank}(Z_{l-1}W^V) \\
&\leq \min\{d_s, d_v, d_s, d_v \cdot n_{\text{heads}}\} \\
&\leq \min\{d_s, d_v\}.
\end{aligned}
$$

As a result, the left nullspace of V, defined as:

$$\mathrm{LN}(V) := \{x^\top \in \mathbb{R}^{1 \times d_s} | x^\top V = 0\},$$

is non-trivial ($\mathrm{LN}(V) \neq \{\vec{0}\}$) when the maximum input length, d_s, is larger than the dimension of the value matrix d_v, i.e., $d_s > d_v$. In this case, we can construct infinitely many matrices $A + \tilde{A}$,

$$\tilde{A} = [x_1, \ldots, x_{d_s}]^\top, x_i \in \mathrm{LN}(V),$$

which contribute exactly the same to the output as the attention matrix A:

$$(A + \tilde{A})V = AV + \tilde{A}V = AV + \vec{0} = AV.$$

This also holds when the weights of $A + \tilde{A}$ are constrained to the probability simplex, and such constrained matrices $A + \tilde{A}$ exist.

B Additional Results

We provide the following additional results that complement the discussions in Section 3:

- A comparison of the evaluation time with standard vs. effective attention.
- In Figure 4, visualization of results presented in Table 2.
- Attention to the `[CLS]` token in Figure 5.
- Attention to the `[SEP]` token in Figure 6.
- Complete Figure 2.

	RTE	**MRPC**	**QNLI**	**SST-2**	**SST-B**
standard	0:29	0:45	10:59	1:41	2:54
effective	0:58	1:27	21:05	3:20	5:53

Table 3: A comparison of the evaluation clock time (minutes:seconds) of BERT models (trained with the standard attention) evaluated with standard attention and effective attention separately.

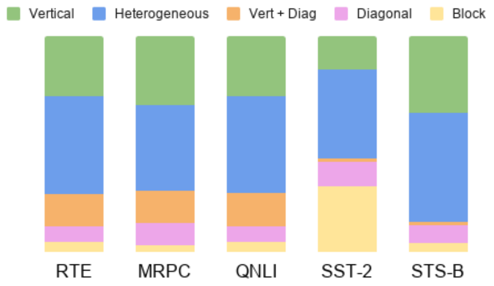

(a) Standard attention.

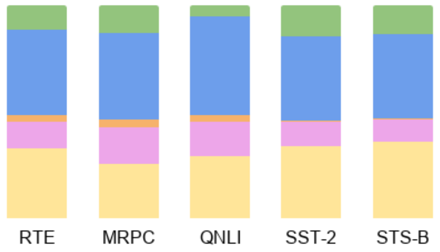

(b) Effective attention.

Figure 4: Estimated percentage of the attention patterns (§3.1) for each task.

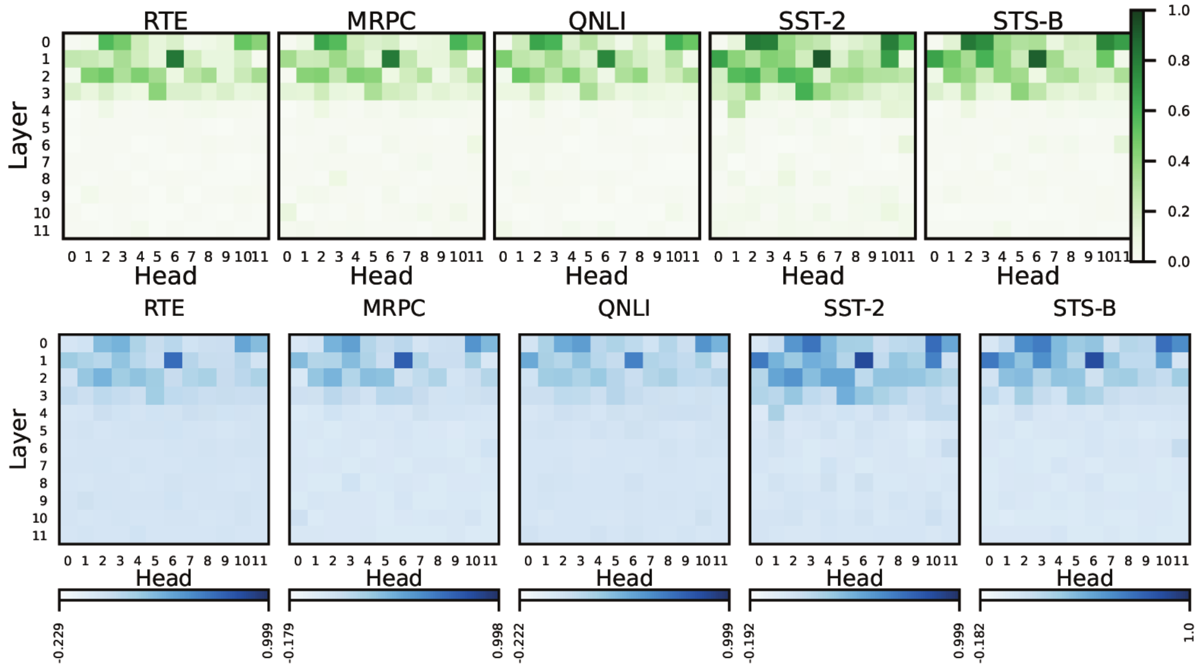

Figure 5: Per-task attention across layers and heads to the **[CLS]** token when processing other input tokens, averaged over sequence length and dataset items for the selected GLUE task. The darker colors corresponding to larger absolute attention weights. The top (green) figure is computed with the standard attention, and the bottom (blue) figure with the effective attention. Since the effective attention does not have a fixed range as the standard attention (from 0 to 1), we use the minimum and maximum effective attention weight for each task calculated across all weights (not only those associated with the [CLS] token).

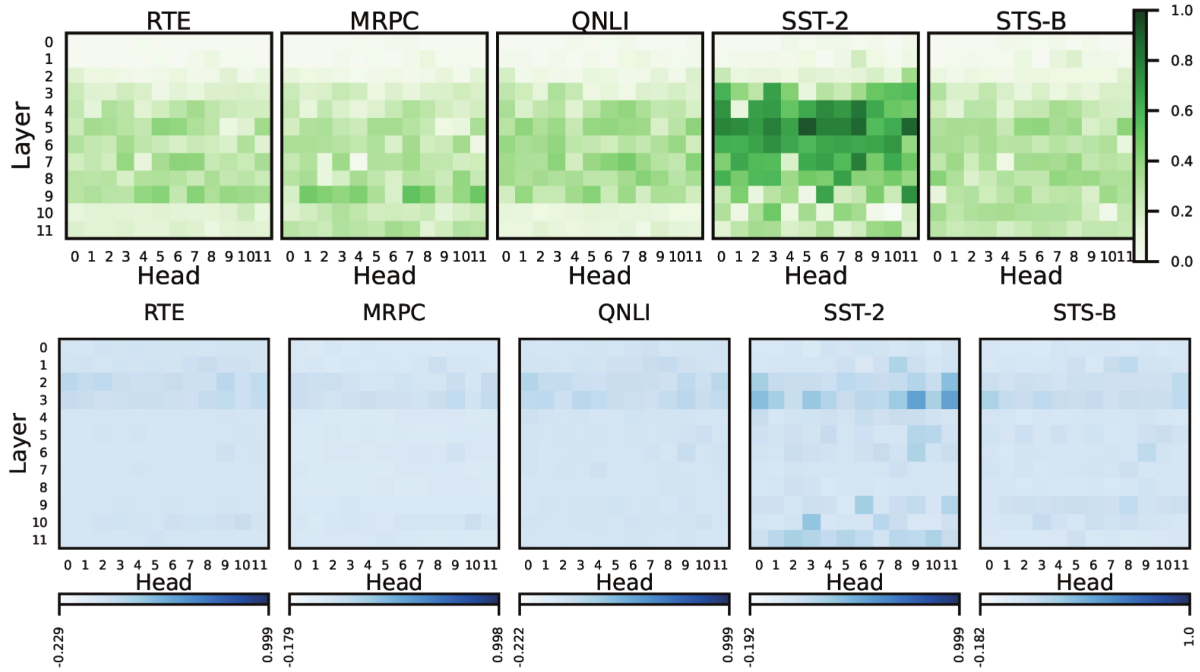

Figure 6: Per-task attention across layers and heads to the **[SEP]** token when processing other input tokens, averaged over sequence length and dataset items for the selected GLUE task. The darker colors corresponding to larger absolute attention weights. The top (green) figure is computed with the standard attention, and the bottom (blue) figure with the effective attention. Since the effective attention does not have a fixed range as the standard attention (from 0 to 1), we use the minimum and maximum effective attention weight for each task calculated across all weights (not only those associated with the [SEP] token).

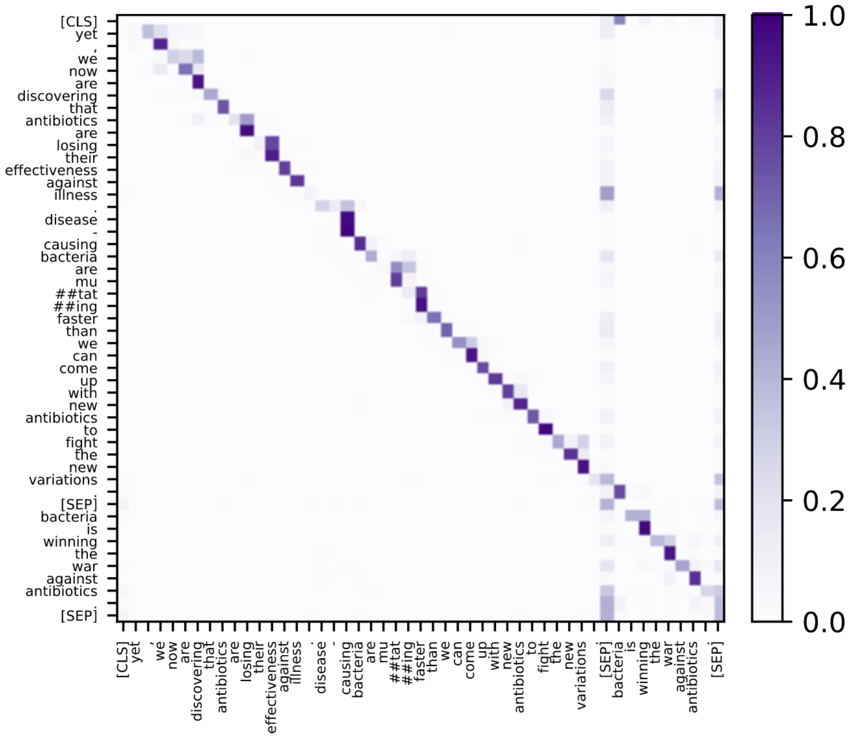

(a) Standard attention

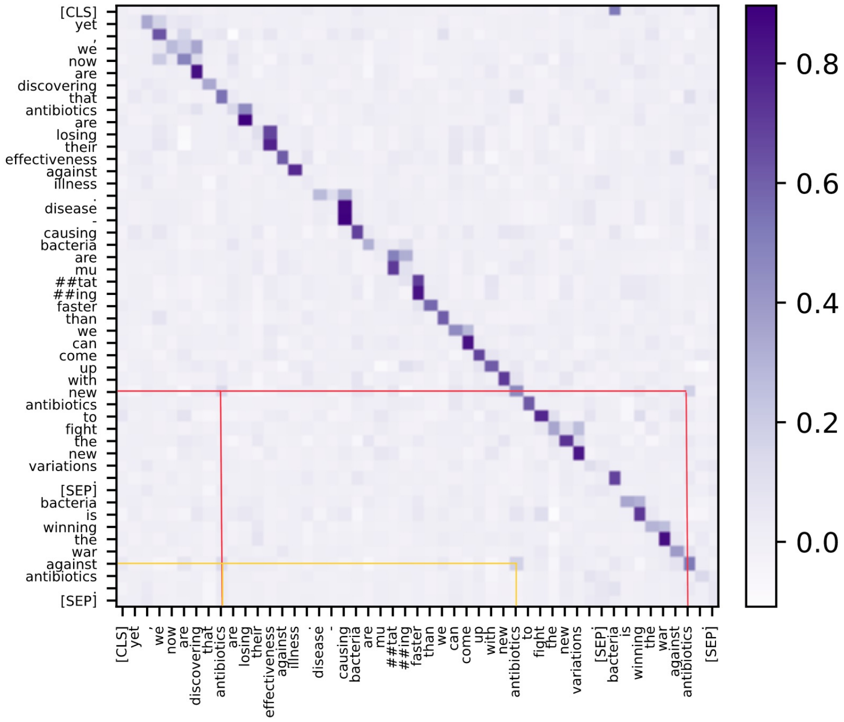

(b) Effective attention

Figure 7: Complete Figure 2.

Compositionality of Complex Graphemes in the Undeciphered Proto-Elamite Script using Image and Text Embedding Models

Logan Born[1] **Kathryn Kelley**[2] **M. Willis Monroe**[3] **Anoop Sarkar**[1]
loborn@sfu.ca kathryn.kelley@utoronto.ca willis.monroe@ubc.ca anoop@cs.sfu.ca

[1]Simon Fraser University
School of Computing Science

[2]University of Toronto
Department of Near and
Middle Eastern Civilizations

[3]University of British Columbia
Department of Asian Studies

Abstract

We introduce a language modeling architecture which operates over sequences of images, or over multimodal sequences of images with associated labels. We use this architecture alongside other embedding models to investigate a category of signs called complex graphemes (CGs) in the undeciphered proto-Elamite script. We argue that CGs have meanings which are at least partly compositional, and we discover novel rules governing the construction of CGs. We find that a language model over sign images produces more interpretable results than a model over text or over sign images and text, which suggests that the names given to signs may be obscuring signals in the corpus. Our results reveal previously unknown regularities in proto-Elamite sign use that can inform future decipherment efforts, and our image-aware language model provides a novel way to abstract away from biases introduced by human annotators.

1 Introduction

This work sets out to understand a category of signs called complex graphemes (CGs) in the undeciphered proto-Elamite (PE) script, a writing system from ancient Iran dating to approximately 3300-2900 BC (Dahl et al., 2013).[1] PE is partly contemporaneous with the world's other two earliest writing systems, Egyptian hieroglyphs and proto-cuneiform, and is the least deciphered of the three, with the underlying language(s) remaining unknown. PE was used exclusively as an accounting technology, employing several numerical systems whose bundling principles are known. Although written in continuous lines, PE, like proto-cuneiform, is most comparable to an accountant's spreadsheet; some structures and rules governing sign use have been identified (Hawkins, 2015; Dahl et al., 2018; Englund, 2004).

The corpus consists of approximately 1500 published clay tablets from excavations in Iran, almost all of which exist in electronic transliteration following the conventions of a work-in-progress sign list (Dahl, 2006). As with other decipherments, understanding the nature of signs and the nuances of sign use is as important as identifying the underlying language(s). Meaningful information can be recovered and the texts partly "read" even if the language remains unknown.

To better understand sign usage in PE, this work proposes an architecture for image-aware language modeling, which permits sharing information between visually similar signs much as sub-word units share information between words. We use sign embeddings to demonstrate patterns which are not readily apparent due to the complexity of the accounting system and the large number of sign shapes found in the script. Our analysis offers insights on complex graphemes that can aid in future hypothesis generation. We confirm that some transliteration choices by PE specialists capture meaningful semantic divisions in the script; this is not a trivial fact, due to the large number of similar looking signs. By using image-aware models, we also observe that some signs with distinct names receive very similar embeddings, implying a functional equivalence that could be exploited by merging signs to create a less sparse corpus that is more amenable to analysis by NLP methods.

2 Methodology

As described by Dahl (2005), CGs in proto-Elamite are signs that consist of one sign inscribed within another (transliterated |S1+S2|), or of one sign framed by two instances of another (|S1+S2+S1|). Rarely, S1 and S2 occur connected at the side, as

---

[1] Our code, data, and trained models are available at `https://github.com/sfu-natlang/pe-compositionality`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4136–4146
August 1–6, 2021.

in |M296+M296| . We refer to S1 as the *outer* sign and S2 as the *inner* sign, though we acknowledge this terminology is not quite appropriate in cases like |M296+M296|. Most signs which occur as part of a CG can also occur as standalone signs. Exceptions to this are rare, such as M600 which only ever occurs in the hapax |M362+M600|.

Although these signs are *orthographically* compositional, it is not known whether they are also semantically compositional. Similar constructions exist in proto-cuneiform (PC), including "containers" with signs inscribed to indicate specific products (Wagensonner, 2015). Some PC compounds survive into later cuneiform, and sometimes have idiomatic meanings, e.g. cuneiform GU_7 "eat", a combination of "head" and "bowl". Chinese characters likewise exhibit varying degrees of visual and semantic compositionality (Sproat, 2006).

Past work (Mikolov et al., 2013b; Salehi et al., 2015; Cordeiro et al., 2016) suggests that embedding models capture semantic compositionality in noun compounds and multiword expressions. Often, these models assign a compound a representation which is similar to the sum of the representations of the words in the compound. Thus we predict that if CGs are semantically compositional, their embeddings will be additively compositional at a higher rate than expected by chance. Their embeddings may also exhibit other signs of internal structure, such as the ability to model proportional analogy between CGs with shared components:

|M136+M365| : M136 :: |M327+M365| : M327

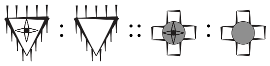

If this analogy holds in the embedding space (which is to say that the 3CosAdd formula |M136+M365| - M136 + M327 $\approx$ |M327+M365| holds between the signs' embeddings) this would give further evidence that the CGs involved have some degree of semantic compositionality.

Unfortunately, most PE signs are rare, which impedes a model's ability to learn meaningful information about their distributions. Yet many signs with distinct names have striking visual resemblances, and it is usually not known whether they have different meanings. Visual information may therefore help an embedding model by allowing it to share distributional information across graphically similar signs. To this end, we propose an architecture for multimodal language modeling in Figure 1. This architecture uses two separate embedding components. On the left of Figure 1, in red, is a standard embedding layer which replaces a one-hot input with a small, learnable representation. On the right, in blue, a lookup function retrieves an image of the sign represented by the input. A CNN extracts a feature vector from the image, which is max-pooled, flattened, and passed through a dense layer to produce a low-dimensional embedding. Both embeddings are concatenated and fed to a BiLSTM[2] (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) which attempts to predict the name of the next sign in the text. All timesteps share the same weights for the CNN and embedding layers. By omitting the blue image-embedding component we can obtain a normal BiLSTM language model. By omitting the red text-based component, we can obtain an image-only model which never directly sees the labels assigned to the signs in the corpus.

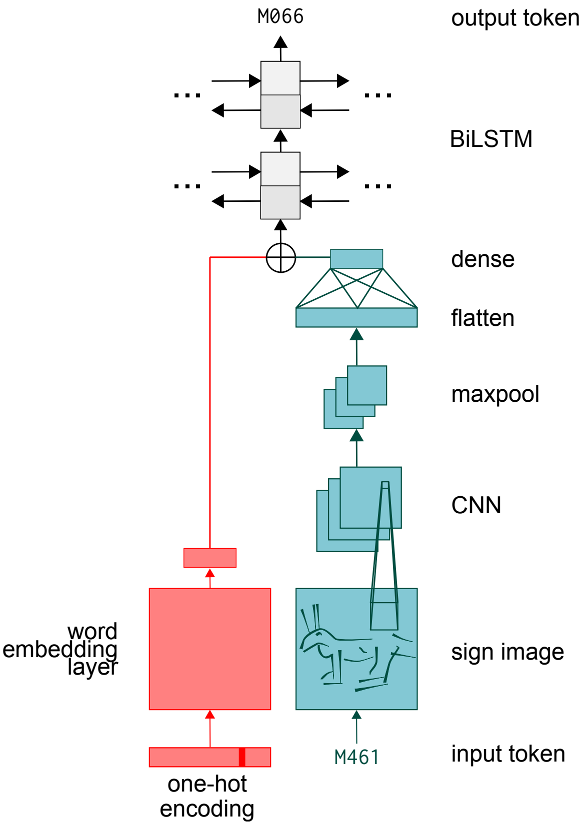

Figure 1: Architecture for image-aware, multimodal language modeling.

To verify that this architecture captures distributional properties of signs, and not just visual properties, we train a separate image recognition model to predict a sign's name given only its image.

---

[2] We also attempted to train a Transformer model (Vaswani et al., 2017), but the corpus size proved insufficient and it always underperformed compared to the BiLSTM.

Model	Input Type	Embedding Sizes[3]	Other Parameters	Description
glove	seq. of sign names	16, 32, 64, 128, 256	window size: 15	Pennington et al. 2014
fasttext.cbow	seq. of sign names	16, 32, 64, 128, 256	window size: 15	Bojanowski et al. 2017
fasttext.skip	seq. of sign names	16, 32, 64, 128, 256	window size: 15	Bojanowski et al. 2017
word2vec.cbow	seq. of sign names	16, 32, 64, 128, 256	window size: 15	Mikolov et al. 2013a
word2vec.skip	seq. of sign names	16, 32, 64, 128, 256	window size: 15	Mikolov et al. 2013a
lm.text	seq. of sign names	64	hidden dimension: 64	Figure 1, blue (image embedding) omitted.
lm.image+text	seq. of sign names and images	64	hidden dimension: 64 image size: 64×64	Figure 1.
lm.image	seq. of sign images	64	hidden dimension: 64 image size: 64×64	Figure 1, red (text embedding) omitted.
image recognition	individual sign image	64	image size: 64×64	Figure 1, blue (image embedding) only.

Table 1: List of models considered in this work.

This model uses the blue image embedding component from Figure 1 to produce a representation of an input image; a dense layer predicts the name of the sign from this embedding. This model only sees signs in isolation, meaning it will not learn from distributional information. If a result holds for the multimodal LM but not for this image recognition model, this implies that the result arises from contextual information in the text, and not simply from visual resemblances between signs.

We also train CBoW and skipgram models with FastText[4] (Bojanowski et al., 2017) and word2vec (Mikolov et al., 2013a), as well as GloVe embeddings (Pennington et al., 2014). Table 1 summarizes all of the models used in this work and important hyperparameters. We train these models on the PE corpus from Born et al. (2019), which is a cleaned version of texts originally published by the Cuneiform Digital Library Initiative (CDLI). This contains digitized transliterations from 1399 tablets comprising 11013 lines in total, or 33778 tokens. 7508 tokens represent broken or unreadable signs, and another 11364 represent numerals, leaving only 14906 non-numerical tokens. 1107 tokens (comprising nearly half the sign types in our cleaned data) are labeled as CGs. We treat each entry of a tablet as a single input sentence for training LMs, and set aside 500 lines as a validation set.

Prior to training, we replace all signs occurring 3 or fewer times[5] with UNK. We replace rare signs wherever they occur, including inside of CGs. The tokens `X` and `...` represent broken or unreadable signs, so we also replace these with UNK. When training language models, we do not backpropagate losses from samples where the target word is UNK, since it so often represents broken material. To make the data less sparse, we remove annotations marking sign variants, so that for example M157 and M157∼a are considered the same sign.

3 Experimental Results

3.1 Additive Composition

We predict that if a CG is semantically compositional, its embedding will approximately equal the sum of the embeddings of the signs it comprises.

Given a sign s, let e_s denote the embedding of s. If s is a CG let $\sigma(s)$ denote the list of signs which make up s. For every CG s in the signary, we check whether $\sum_{t\in\sigma(s)} e_t \approx e_s$. If $\sum_{t\in\sigma(s)} e_t$ is within the k nearest neighbors of e_s for some threshold k, we say that s appears to have a compositional representation.

For different thresholds k, we measure how many CGs have compositional representations. Since many PE signs have low frequency, we predict that noise may drown out any signal when k is small. However, when k is large enough to overcome this noise, we predict that the number of CGs with compositional representations will be greater than expected by chance, as we expect that some CGs have meanings which are semantically compositional rather than idiomatic. Table 2 shows the results from this evaluation.

In text-only models, when k is small the number of CGs with compositional representations is no higher than expected by chance. However, for image-aware models, and for text-only models with large k, the number of CGs which are close to the

---

[3] To give a fairer comparison, we train the simpler models with several embedding sizes and report results from whichever dimensionality performs best on each task. Additional model information is in the supplemental material.

[4] Sign names are largely arbitrary, so we disable sub-words in FastText by setting the maximum sub-word length to 0.

[5] To determine frequency, we count how often a sign occurs both independently and as part of a CG.

	k				
model	1	3	5	10	15
glove.256	0	0	1	3	**13**
word2vec.cbow.16	0	0	1	9	**12**
word2vec.skip.32	0	0	5	**13**	**16**
fasttext.cbow.128	0	2	3	5	9
fasttext.skip.128	0	3	**10**	**15**	**20**
lm.text.64	0	0	0	0	1
lm.image+text.64	1	**14**	**21**	**40**	**51**
lm.image.64	**11**	**16**	**27**	**48**	**61**
image recognition.64	**3**	**7**	**15**	**28**	**38**

Table 2: Number of compositional CGs for different similarity cutoffs k. Bold numbers represent cases where the number of compositional graphemes is significantly larger than expected by chance.

sum of their components is significant. Even for $k = 15$, the signs identified as compositional by `lm.image.64` average >0.97 cosine similarity to the sum of their parts, suggesting this is not too generous a threshold.

Notably, the number of compositional CGs in `lm.image.64` is always larger than the number in any of the other models, including the image recognition model.[6] This has the important implication that compositionality in the embeddings is not solely a consequence of visual compositionality. If that were the case, the contextual information available to the LM would not be useful for this task, and the image LM would not be expected to find more compositional CGs than the image recognition model. Moreover we would not expect to find a significant amount of compositionality in any of the text-only models for any k. Table 3 shows examples of signs which appear to be compositional in the image LM but not the image recognition model. These are signs for which contextual information plays a deciding role in making them appear semantically compositional, and which may therefore be of interest to analyze in future work.

<table>
<tr><td>M153</td><td>+ M106</td><td>≈ |M153+M106|</td></tr>
<tr><td>M175</td><td>+ M286</td><td>≈ |M175+M286|</td></tr>
<tr><td>M327</td><td>+ M348</td><td>≈ |M327+M348|</td></tr>
<tr><td>M362</td><td>+ M244</td><td>≈ |M362+M244|</td></tr>
<tr><td>M157</td><td>+ M288</td><td>≈ |M157+M288|</td></tr>
<tr><td>M175</td><td>+ M153</td><td>≈ |M175+M153|</td></tr>
<tr><td>M218</td><td>+ M388</td><td>≈ |M218+M388|</td></tr>
</table>

Table 3: Sample of signs which appear to be compositional in the image LM but not the image recognition model.

We emphasize that the text-only models have no information about sub-words (such as CG components), so any compositionality in these models must exclusively reflect distributional properties. From these results we conclude that there is legitimate evidence for some CGs having semantically compositional meanings in PE.

3.2 Pairing Consistency

To assess the contribution of a sign to the CGs it occurs in, we consider the pairing consistency score (PCS) from Fournier et al. (2020). This metric measures whether the offsets between pairs of words are more parallel than expected by chance. If a sign s always contributes the same meaning to the CGs in which it occurs, then the offset between the pair of signs (t, $|t + s|$) is expected to be roughly parallel to the offset between the pair (u, $|u + s|$) for most choices of t and u. If CGs containing s have idiomatic meanings (so the contribution of s is not consistent), the offsets between such pairs are not likely to be parallel. Thus PCS serves as a proxy for compositionality, and allows us to investigate the impact of individual signs on the representations of CGs in which they occur. This is distinct from a measure like mutual information which depends on raw sign counts and does not account for the internal structure of sign embeddings.

For each sign s we construct two relations. $R_{s,in}$ contains all CGs with s as the inner sign, paired with whichever sign forms the outer part of the CG. $R_{s,out}$ contains all CGs with s as the outer sign, paired with whichever sign forms the inner part of the CG. Formally, given a CG c containing a sign s, let $\delta(c, s)$ denote the element of c which is not s. Further, let $I(s)$ be the set of all CGs with s as the inner element and $O(s)$ be the set of all CGs with s as the outer element. Then

$$R_{s,in} = \{(\delta(c, s), c) \mid c \in I(s)\}$$

$$R_{s,out} = \{(\delta(c, s), c) \mid c \in O(s)\}$$

Table 4 reports the average PCS[7] of $R_{s,in}$ and $R_{s,out}$ for each model, averaged across all signs s. On average, we find that inner signs have higher PCS than outer signs. This difference is statistically significant in the image-aware LMs, the image recognition model, and FastText. This implies that inner signs have a more consistent and predictable impact on the representation of compounds in which they occur. The fact that this holds for

---

[6] The image recognition model has fewer parameters than the LMs, but it attains >99% accuracy on its original task, suggesting that it does not suffer from being a smaller model.

[7] We compute PCS using the code published by Fournier et al. (2020), but we adjust the permutation-finding function to avoid infinite loops when a relation contains few items.

some text-only models as well as for the image-aware LMs implies that it is due to distributional properties of signs and not simply their appearance.

	Mean PCS	
model	$R_{s,in}$	$R_{s,out}$
glove.64	0.542	0.544
word2vec.cbow.64	0.525	0.492
word2vec.skip.64	0.521	0.495
fasttext.cbow.64	**0.562**	**0.484**
fasttext.skip.64	**0.539**	**0.500**
lm.text.64	0.465	0.529
lm.image+text.64	**0.719**	**0.482**
lm.image.64	**0.760**	**0.536**
image recognition.64	**0.929**	**0.493**

Table 4: Comparison of pairing consistency for the inner and outer parts of compound signs in 64-dimensional models. Bolded rows represent pairs where the difference between columns is significant.

Fournier et al. (2020) note that different categories of relations in English have different average PCS. They find that relations involving inflectional morphology (for example, between a verb and its gerund) have high PCS, relations involving derivational morphology (as between *heat* and *reheat*) have lower PCS, and other semantic relations (as between *hot* and *cold*) have the lowest PCS of the relations they examine.

We expect that absolute PCS values will not be comparable between PE and English, owing to the very different nature of the two writing systems. However, it may be possible to draw broad comparisons between different categories. As the category with the highest PCS, inner signs appear to pattern with inflectional morphology, while outer signs pattern more closely with regular lexical items. This does not imply that inner signs actually encode inflectional morphology: most PE signs likely correspond to objects or ideograms, and most types of morphological marking were absent in the earliest phases of Near Eastern writing (Nissen et al., 1993). Rather, we suggest that inner signs may offer minor refinements to the meaning of an outer sign without fundamentally changing its value, parallel to the way that inflecting a verb refines its role in a sentence but does not change its basic meaning.

3.3 Analogy

Our PCS results measure sign behaviour in aggregate, but do not provide specific examples of relations between signs. We augment these results by searching for concrete analogies which hold in the embedding models.

Given two CGs s and t, let $s - t$ denote the signs that are in s but not t, and let $s \cap t$ denote the signs both CGs have in common. Consider the vector

$$A(s,t) = e_s - \sum_{u \in s-t} e_u + \sum_{v \in t-s} e_v$$

This vector represents the analogical formula $s : (s-t) :: t : (t-s)$. If $A(s,t) \approx e_t$ in a particular embedding model, then this analogy appears to hold true according to that model.

We compute how often $A(s,t)$ is within the k nearest neighbors of e_t for different thresholds k when $s \cap t \neq \emptyset$. We also compute how often $A(s,t)$ is close to e_t when s and t are randomly chosen CGs. We predict that CGs which have signs in common also have some meaning in common, and consequently that the former value will be significantly larger than the latter value.

Table 5 shows the results of this evaluation. As in the compositionality task, more analogies hold between CGs with shared components in image-aware models than in text-only models, and the largest number by far occur in the image LM. Once again, in `lm.image.64` the target vector averages >0.97 similarity to the computed vector even when $k = 15$. Bold numbers in the table represent cases where analogies are significantly more likely to hold between CGs with shared components than between random pairs of CGs. We see that the number of analogies is larger than expected by chance even in some text-only models, suggesting that there is a meaningful relationship between some CGs which have elements in common. The fact that the image LM outperforms the image recognition model further implies that these analogies reflect legitimate distributional properties and are not purely due to visual resemblance.

			k		
model	1	3	5	10	15
glove.256	0	8	11	25	48
word2vec.cbow.256	0	17	36	**65**	**90**
word2vec.skip.128	0	8	29	**97**	**140**
fasttext.cbow.128	0	9	22	**64**	**98**
fasttext.skip.256	0	11	30	91	**145**
lm.text.64	0	2	7	16	21
lm.image+text.64	**27**	**82**	**134**	**233**	**320**
lm.image.64	**69**	**172**	**258**	**393**	**521**
image recognition.64	**29**	**67**	**92**	**133**	**174**

Table 5: Number of analogies which hold between CGs with signs in common, for different similarity cutoffs k. Bold numbers represent values which are significantly larger than expected by chance.

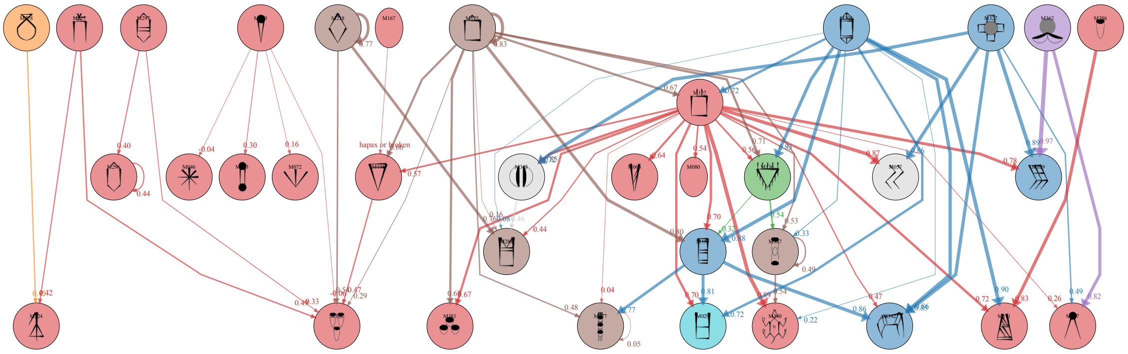

Figure 2: Containment hierarchy for a subset of the signs which can occur in CGs. Directed edges point from outer signs to the inner signs they can contain. Note that (excluding self-loops) the graph is acyclic and all edges point from higher nodes to lower ones. Thicker edges represent CGs which are more strongly compositional. Nodes are colored according to modularity class (Blondel et al., 2008) such that nodes are most strongly connected to like-colored nodes. Full hierarchy, showing all signs which occur in CGs, is available in supplemental material.

As was the case for additive compositionality, the image+text LM underperforms the image-only LM, and on this task the difference is much more pronounced. This suggests that sign names act as distractors: if sign names conveyed information which was helpful to the analogy task, their inclusion would be expected to improve performance. This fact has implications about the labeling of the data which we return to in Section 4.

Taken altogether, the results suggest that many CGs have compositional meanings which can be understood by comparison to the meanings of their component parts and the other CGs with which they share components. We next consider which pairs of signs are able to combine into CGs and which pairings are never observed.

3.4 CG Containment Rules

Some signs which occur as the inner part of one CG may also occur as the outer part of another, as with M348  in |M327+M348|  and |M348+M004| . We may therefore expect to find pairs of signs where either one can contain the other, and yet, no such pairs actually exist. In fact, we find that CGs appear to be constructed according to a strict hierarchy whereby a sign may only contain itself or another sign which is lower on this hierarchy. We can visualize this as a lattice with directed edges from outer signs to the inner signs they are observed to contain, as in Figure 2 (excerpted from the full hierarchy available in the supplemental material). The thickness of an edge in this figure is proportional to the compositionality of the corresponding CG in `lm.image+text.64`.

There appears to be some relation between a sign's compositionality and its position in this lattice. The signs on the left half of Figure 2 have low compositionality (seen as thinner edges in the figure) while the nodes to the right have higher compositionality (seen here as thicker edges). This suggests that there may be different kinds of CG, of which some are idiomatic and some are not, and that these categories have sufficiently little overlap to appear as separate modules in the lattice.

This "grammar" governing CG construction has not been noted in previous PE scholarship. The ordering of signs within this hierarchy deserves attention in future work, as it may reflect different levels of administrative units in PE society, degrees of specificity in qualifying commodities, or other information which can be exploited to understand the content of these texts.

4 Analysis

Little is known about the role of CGs in PE, although these signs make up a significant portion of the corpus. Some occur in "headers" appearing at the beginning of a text. In headers, outer signs (such as M157) are hypothesized to indicate the type of household or institution to which the entire account relates. The outer sign may be further specified by an inner sign, but many (including M157) can also appear without another sign inscribed within. Inner signs are hypothesized to specify a particular kind of item being recorded, a person, profession, or administrative department related to an account, and more.

Our results are consistent with these hypotheses. The PCS results point to inner signs playing a specializing role; this is corroborated by visual inspection of the embedding space, which reveals that CGs cluster according to their outer sign rather than their inner sign (cf. Figure 3 below).

According to Table 2, our text-only models detect additive composition in at most one of every 10 CGs; the image LM detects it in one of every 4 CGs. Likewise, the image LM suggests that a meaningful analogical relation obtains between slightly less than one-third of all pairs of CGs with signs in common. These values depend on the threshold k, but they suggest the presence of a least a small core of compositional CGs in PE. In several places, compositional and non-compositional CGs appear separated from one-another in the CG containment hierarchy (cf. Figure 2), which may point to this being a legitimate distinction in the writing system and not a failure of our models to detect compositionality in some cases where it is really present.

We can make some inferences about the CGs which are compositional. They are not likely to represent either combinations of ideograms with an emergent lexical value (like the Sumerian cuneiform sign for *naŋ* "drink" combining the signs for human head and water) or ideograms with phonetic complements (signs indicating the proper reading of the CG), as both cases should be expected to produce non-compositional meanings. Our results may also counter-indicate "coat-of-arms"-like symbols (Farmer et al., 2004), since we show that the components of CGs can often be understood in relation to their use elsewhere in texts, and since CG elements on their own often seem to reference products (including foodstuffs and livestock) and their distribution. Future work may train embedding models on proto-cuneiform, a structurally-similar writing system containing compound signs with occasionally known meanings that could act as useful points of comparison.

The two components of a CG can occur independently, within the same text or even side-by-side. A dramatic example comes from |M218+M288| , the components of which appear 37 times as the bigram M218 M288. M288  ("grain container") is the most frequent sign in PE, appearing in diverse contexts but often before numerical measures of capacity. M218  is among the signs speculated to function "syllabically" to write personal names (Dahl, 2019), though it may also have other uses. It is not clear yet whether |M218+M288| and M218 M288 operate identically, particularly since |M218+M288| is not strongly additively compositional in any of our embedding models. The possible polyvalence of M218 and broad distribution of M288 may impact models' ability to detect compositionality in |M218+M288|. Despite this difficulty, the image LM identifies analogies between |M218+M288|, |M175+M288|, and |M305+M288| (the analogy vector has >0.99 cosine similarity to the target in both cases) implying that we should at least consider M218, M175, and M305 as parallel categories each with relation to grain capacities.

Some signs rarely occur outside of CGs, such as the productive inner sign M342 , about which practically nothing is known. Our data show that it has moderately high PCS (0.69 in `lm.image.64`) and that analogies hold between all but one of the CGs which contain M342 (|M157+M342|, |M304+M342|, |M305+M342|, |M325+M342|, |M327+M342|, and |M351+M342|, excluding |M153+M342|). These analogies hold strongly for the image LM but not the image recognition model, meaning they reflect primarily distributional properties. Many of these signs are also additively compositional. We believe that these signs may be suitable starting points for future analysis, as our results imply that they are probably not idiomatic and are likely to have related meanings.

|M157+M342| : M157 :: |M304+M342| : M304
|M157+M377+M377| : M157 :: |M175+M377+M377| : M175
|M370+M046+M370| : M046 :: |M370+M072+M370| : M072
|M175+M377+M377| : M175 :: |M201+M377+M377| : M201
|M351+1(N14)| : 1(N14) :: |M351+M380| : M380
|M036+1(N39C)| : 1(N39C) :: |M036+M035| : M035
|M136+M365| : M136 :: |M327+M365| : M327
|M157+M057| : M157 :: |M327+M057| : M327

Table 6: Sample of analogies which hold in the `lm.image+text.64` model.

Table 6 gives additional examples of analogies which hold in `lm.image+text.64`. We see that inner and outer signs both participate in analogical relations, as do both |S1+S2|-type CGs and |S1+S2+S1|-type CGs. Some analogies hold between a CG with a numeric inner sign and one with a non-numeric inner sign, as between |M036+1(N39C)| and |M036+M035|. Such cases may have implications to the meaning of the signs involved; if 1(N39C) and M035 truly have parallel functions in these two CGs, this may imply a kind of quantifying role for M035, or alternatively that 1(N39C) is used for its pronunciation or possible syllabic value rather than as a true numeral. The

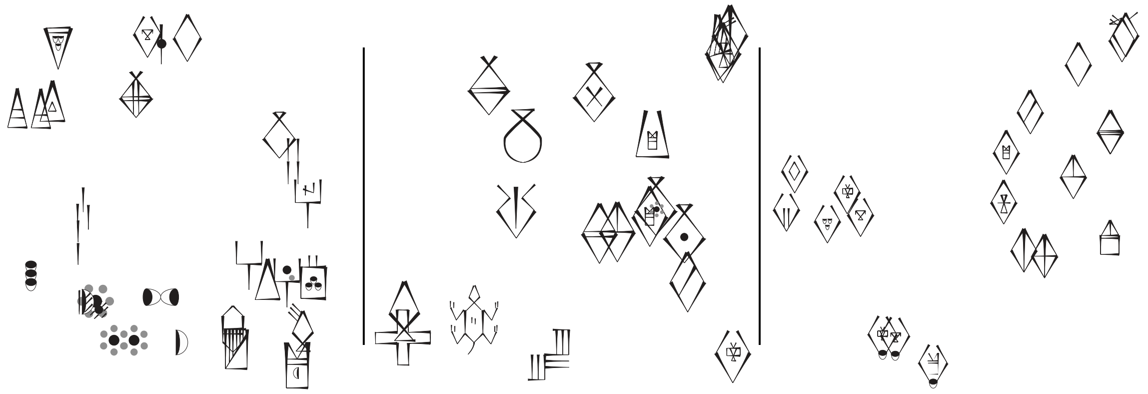

Figure 3: Detail from t-SNE decompositions of the GloVe embeddings (left), the image LM (centre) and the image recognition model (right).

existence of other M036 compounds containing numerals (e.g. |M036+1(N30D)| and |M036+1(N14)|) would seem to favor the former interpretation.

The image-only LM found stronger signals for compositionality and analogical relations than the image+text LM, suggesting that sign names acted as distractors for those tasks. This has significant implications for the ongoing process of revising the PE sign list. Our work relies on the sign labels assigned through an exhaustive manual transliteration process; since it is easy to automate merging signs, this process assumed that most signs are unique until proven otherwise. However, we now believe this choice weakens signals in the text data by making most signs very rare. Moreover, some signs which appear graphically compositional are not currently labeled as CGs, usually when the inner part is never attested as a standalone sign. For these reasons, future work may benefit from relabeling signs based on a combination of context and sign shape.

At the same time, the current transliteration system may record meaningful (if fine-grained) information reflected in minor graphical details (consider M263  and M262 ), such as (hypothetically) "jug of red beer" versus "jug of dark beer". Such similarly functioning signs might obtain similar embeddings, but retaining their distinction in the published transliterations still improves our understanding of the texts. However for both manual and machine-learning analysis, significant reductions in the sign list may open new avenues for decipherment: for instance, Born et al. (2019) note that frequency-based approaches to decipherment are currently difficult in PE owing to the very small number of repeated n-grams in the corpus.

Figure 3 shows details from the embedding spaces learned by GloVe, the image LM, and the image recognition model.[8] GloVe produces small clusters of visually similar signs even though it does not have access to sign images: note the proximity of M353 , M354 , and 2(N30C) , as well as the variants of M036 . These clusters occur in sufficient number that we have confidence the model is detecting meaningful similarities in the usage of visually similar signs. The image recognition model produces much clearer groupings of visually related signs, as would be expected. The image LM replicates some clusters from the image recognition model: a cluster of lozenge-shaped signs is visible in both the image LM figure and the image recognition figure. However, contextual information causes the image LM to relocate other lozenge-shaped signs like M218  to a different part of the embedding space, implying a functional difference between it and the signs in the figure. Overall, these observations confirm that our multimodal architecture is finding a balance between contextual and visual information as intended.

5 Related Work

Sun et al. (2019) introduce "character-enhanced" embeddings of Chinese words. Their architecture roughly parallels our own, but requires a deeper CNN due to the visual complexity of Chinese characters. We train with a full context language modeling objective whereas they use a sampling scheme similar to word2vec. They use character-level information to improve word embeddings, where we exclusively learn character embeddings. Our application of this architecture to decipherment is novel.

Liu et al. (2017) explicitly learn compositional embeddings for Chinese characters. They use su-

---

[8] Full figures are available in our supplemental data.

pervised data to help identify when two visually-distinct signs use the same radical (as in 水 and 池). In our data, it is not known which signs are truly related to one another, thus we refrain from giving the model explicit information about compositionality.

Yin et al. (2019) segment and transcribe undeciphered scripts based on visual similarities between glyphs. Although their transcription error rate is high, they still achieve partial decipherments with no human intervention.

Dencker et al. (2020) perform OCR-style sign detection on images of Sumerian cuneiform tablets, recognizing signs which may be written very differently across the corpus. Their task benefits from the existence of supervised Sumerian training data.

Born et al. (2019) train topic models on PE texts and cluster PE signs in a simple mutual information-based embedding model. The present work considers more sophisticated embedding models and performs a more detailed investigation of the embedding space.

Luo et al. (2019) perform automated decipherment of Ugaritic. Their technique finds alignments between orthographic representations of phonetic information, and thus is not easily applicable to ideographic scripts. It also requires multilingual data, and cannot extract information from a script with no known surviving relatives.

Our work exploits the embedding space learned by a neural language model, but the actual task of language modeling is otherwise irrelevant to our results. By contrast, Kambhatla et al. (2018) actually sample text from a neural language model to help estimate the quality of a proposed decipherment. Future work could similarly sample from a language model as a means of counteracting the small size of the PE corpus; this should be done with caution, however, given the difficulty of evaluating whether the sampled text is fluent.

Salehi et al. (2015) and Cordeiro et al. (2016) demonstrate that English word embeddings tend to be additively compositional and can capture human intuitions about semantic compositionality. Hartung et al. (2017) investigate other methods for decomposing word embeddings.

Sproat (2006) discusses a variety of writing systems and the degrees to which they employ phonetic versus semantic information. The discussion is largely taxonomic and addresses subtle nuances between scripts which are already well-understood. In this way it demonstrates the wide range of variation observed between scripts, and by extension the range of possibilities which should be considered when analyzing an undeciphered script such as PE.

6 Conclusion

Interpreting what a word embedding model has learned typically involves a comparison to native speaker intuitions. In contrast, in this work we have shown how exploiting graphical compositionality and carefully examining sequences of image embeddings can lead to new insights in proto-Elamite (PE), an undeciphered script with no living users and relatively little available data. Abstracting away from human annotations, we introduced a novel architecture for multimodal or image-based language modeling, which shares information between visually similar signs to better model contextual patterns. This provides a new toolkit for decipherment of an unknown language, distinct from translation-based approaches.

As one of the world's earliest experiments in writing, employing 774 signs and variants by current estimates, reasonable concerns have existed over PE's level of standardisation and the impact this may have on decipherment (Dahl, 2019:71, 82). The corpus is small and filled with lacunae, and prior work has done little to understand how NLP techniques function on early writing systems which may reflect linguistic content differently from modern writing systems. Despite these challenges, this work has shown that embedding models can indeed identify meaningful patterns in proto-Elamite.

We have presented evidence that a subset of complex graphemes are semantically compositional rather than idiomatic, and we have discovered the existence of a simple grammar or partial ordering which appears to govern the construction of CGs. Our results should give domain experts confidence that the proto-Elamite script contains sufficient regularities to allow for describing its mechanics and potentially understanding the underlying content.

Acknowledgments

We thank the anonymous reviewers for their comments. The research was partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025 to the fourth author.

References

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Logan Born, Kate Kelley, Nishant Kambhatla, Carolyn Chen, and Anoop Sarkar. 2019. Sign clustering and topic extraction in Proto-Elamite. In *Proceedings of the 3rd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 122–132, Minneapolis, USA. Association for Computational Linguistics.

Silvio Cordeiro, Carlos Ramisch, Marco Idiart, and Aline Villavicencio. 2016. Predicting the compositionality of nominal compounds: Giving word embeddings a hard time. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1986–1997, Berlin, Germany. Association for Computational Linguistics.

Jacob L. Dahl. 2005. Complex graphemes in proto-elamite. *Cuneiform digital library journal*, 4(3).

Jacob L. Dahl. Proto-elamite sign list [online]. 2006.

Jacob L. Dahl. 2019. Tablettes et fragments proto-élamites / Proto-Elamite tablets and fragments. *Textes Cunéiform Tomes XXXII Musée de Louvre*.

Jacob L. Dahl, Laura Hawkins, and Kate Kelley. 2018. Labor administration in proto-Elamite Iran. In Agnès Garcia-Ventura, editor, *What's in a Name? Terminology related to the Work Force and Job Categories in the Ancient Near East*, pages 15–44. Alt Orient und Altes Testament 440. Ugarit Verlag: Münster.

Jacob L. Dahl, Cameron A. Petrie, and Daniel T. Potts. 2013. Chronological parameters of the earliest writing system in iran. In Cameron A. Petrie, editor, *Ancient Iran and Its Neighbours*. Oxbow Books, Oxford, UK.

Tobias Dencker, Pablo Klinkisch, Stefan M. Maul, and Björn Ommer. 2020. Deep learning of cuneiform sign detection with weak supervision using transliteration alignment. *PLOS One*.

Robert K. Englund. 2004. The state of decipherment of proto-Elamite. *The First Writing: Script Invention as History and Process*, pages 100–149.

Steve Farmer, Richard Sproat, and Michael Witzel. 2004. The collapse of the indus-script thesis: The myth of a literate harappan civilization. *Electronic Journal of Vedic Studies*, 11(2).

Louis Fournier, Emmanuel Dupoux, and Ewan Dunbar. 2020. Analogies minus analogy test: measuring regularities in word embeddings. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 365–375, Online. Association for Computational Linguistics.

Matthias Hartung, Fabian Kaupmann, Soufian Jebbara, and Philipp Cimiano. 2017. Learning compositionality functions on word embeddings for modelling attribute meaning in adjective-noun phrases. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 54–64, Valencia, Spain. Association for Computational Linguistics.

Laura F. Hawkins. 2015. A new edition of the Proto-Elamite text MDP 17, 112. *Cuneiform Digital Library Journal*, 1.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Nishant Kambhatla, Anahita Mansouri Bigvand, and Anoop Sarkar. 2018. Decipherment of substitution ciphers with neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 869–874, Brussels, Belgium. Association for Computational Linguistics.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2059–2068, Vancouver, Canada. Association for Computational Linguistics.

Jiaming Luo, Yuan Cao, and Regina Barzilay. 2019. Neural decipherment via minimum-cost flow: From Ugaritic to Linear B. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3146–3155, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

H.J. Nissen, P. Damerow, and R.K. Englund. 1993. *Archaic Bookkeeping: Early Writing and Techniques of Economic Administration in the Ancient Near East*. University of Chicago Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 977–983, Denver, Colorado. Association for Computational Linguistics.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Richard Sproat. 2006. *A Computational Theory of Writing Systems (Studies in Natural Language Processing)*. Cambridge University Press, USA.

Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. VCWE: visual character-enhanced word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2710–2719. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

K. Wagensonner. 2015. Vessels and other containers for the storage of food according to the early lexical record. *Origini Preistoria e protostoria delle civilità antiche*, XXXVII(1).

Xusen Yin, Nada Aldarrab, Beáta Megyesi, and Kevin Knight. 2019. Decipherment of historical manuscript images. In *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, pages 78–85. IEEE.

On the Distribution, Sparsity, and Inference-time Quantization of Attention Values in Transformers

Tianchu Ji[1], **Shraddhan Jain**[2], **Michael Ferdman**[2],
Peter Milder[1], **H. Andrew Schwartz**[2], **and Niranjan Balasubramanian**[2]

[1,2]Stony Brook University
[1]{tianchu.ji, peter.milder}@stonybrook.edu
[2]{shrjain, mferdman, has, niranjan}@cs.stonybrook.edu

Abstract

How much information do NLP tasks really need from a transformer's attention mechanism at application-time (inference)? From recent work, we know that there is sparsity in transformers and that the floating-points within its computation can be discretized to fewer values with minimal loss to task accuracies. However, this requires retraining or even creating entirely new models, both of which can be expensive and carbon-emitting. Focused on optimizations that do not require training, we systematically study the full range of typical attention values necessary. This informs the design of an inference-time quantization technique using both pruning and log-scaled mapping which produces only a few (e.g. 2^3) unique values. Over the tasks of question answering and sentiment analysis, we find nearly 80% of attention values can be pruned to zeros with minimal ($< 1.0\%$) relative loss in accuracy. We use this pruning technique in conjunction with quantizing the attention values to only a 3-bit format, without retraining, resulting in only a 0.8% accuracy reduction on question answering with fine-tuned RoBERTa.

1 Introduction

While the verdict is still out on which large language model will prove best, at this point in time, all contenders rely on multi-headed attention over multiple layers. Many have investigated whether attention (the output of the softmax, α) itself is *qualitatively* sensible (e.g., correlating with linguistic aspects) (Vig and Belinkov, 2019; Clark et al., 2019; Voita et al., 2018, 2019; Kovaleva et al., 2019; Rogers et al., 2020) or how useful it is for interpreting models (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019; Brunner et al., 2020; Rogers et al., 2020). Others have focused on inducing sparsity in the attention: whether some of the structural components (the softmax function, attention heads and layers) introduce attention sparsity (Correia et al., 2019; Michel et al., 2019; Voita et al., 2019; Sajjad et al., 2020), if the model tends to focus on a small amount of tokens (Clark et al., 2019; Ramsauer et al., 2020), and the interpretability of such sparsity (Chen et al., 2020; Rogers et al., 2020). Yet, little is known about our ability to induce sparsity or reduce its values *at application-time*, and what role the inherent sparsity could play in building inference-time efficient transformers.

This work focuses on a systematic study of the quantitative distribution of the attention values across the layers and heads as well as the potential for reducing the information content of attention values during inference at application-time[1]. We consider two popular pretrained transformer models: BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) over tasks of Masked Language Modeling as well as question answering and sentiment analysis. We explore the attention distributions on the different models and tasks, and quantitatively profile the sparse attention that commonly exists in the transformer model. Motivated by the high levels of inherent sparsity in these distributions, we design a pruning and quantization technique and test the limits of information necessary from attention.

We find that most attention values can be pruned (i.e. set to zero) and the remaining non-zero values can be mapped to a small number of discrete-levels (i.e. unique values) without any significant impact on accuracy. Approximately 80% of the values can be set to zero without significant impact on the accuracy for QA and sentiment analysis tasks. Further, when we add quantization utilizing a log-scaling, we find a 3-bit discrete representation is sufficient to achieve accuracy within 1% of using the full floating points of the original model.

---

[1] Our analyzing code and data are available at `https://github.com/StonyBrookNLP/spiqa`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4147–4157
August 1–6, 2021.

2 Method

To analyze attention distribution we first plot histograms of attention values for BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models. We also compute a *sparsity distribution* using the proportion of the attention values smaller than a given threshold. For attention pruning, we find attention values that are below a specified threshold and replace them with zero. We experiment with different thresholds. For quantization to k-bits we map the continuous attention values to one of 2^k real values[2]. We use two methods: (i) Linear - Bin the attention values to 2^k quantiles and set the mid-point of each as the quantized value. (ii) Log - Bin the log transformed attention values and pick the mid-point of each on the log scale as the quantized value. The quantization methods are explained in detail in Appendix E.

We apply these inference-time (i.e. no training) techniques on three tasks: masked language modeling, question answering and sentiment analysis. For QA we used BERT[3] and RoBERTa[4] models fine-tuned on SQuAD v1.1 (Rajpurkar et al., 2016). For sentiment analysis we used RoBERTa[5] fine-tuned on the SST-2 dataset (Socher et al., 2013). For both these tasks we report accuracy on the corresponding development sets. For the Masked Language Modeling (MLM) task we report pseudo-perplexity (Salazar et al., 2020) computed on the Huggingface Wikipedia dataset[6].

3 Evaluation

Attention distribution and sparsity. A thorough quantitative analysis on the attention distribution could help build efficient transformers by providing useful information, such as the degree of sparsity and the range of the attention values. We plot the histogram of each token's attention to all the others (α_i) and provide three examples of the heads in Figure 1 to investigate the density of the attention values, how differently the tokens attend to others in the same attention head, and how sparse a token/head/layer's attention can be. We find that, for most of the heads, attention forms a lognormal-like distribution similar to Figure 1a. On some heads, some of the attention for query token (α_i) have more tiny attention values (α_{ij}) and induce more sparsity than others (like in Figure 1c). We also observe entire heads with high sparsity, in which nearly all tokens only slightly attend to others (like in Figure 1b). Our observation confirms the existence of sparsity in the attention heads.

A key motivation for us is to quantitatively characterize sparsity, especially in terms of how much potential there is in reducing the information content in attention values. To this end, we specifically measure the proportion of small attention values by counting the number of α_{ij} that sum up to 0.5 in each α_i. This indicates that most heads focus strongly on fewer than 10 tokens on average (details in Appendix A), leading to notable sparsity and suggesting large potential for conveying the same information as continuous attention values using fewer discrete levels.

Beyond these, we occasionally observe outlier attention histograms (like the outliers between $[10^{-4}, 10^{-1}]$ in Figure 1b). We also found noticeable differences on the attention histograms from layer to layer. These findings are related to the works on the syntactic heads/special tokens (Voita et al., 2019; Kovaleva et al., 2019; Voita et al., 2018; Clark et al., 2019; Rogers et al., 2020)) and the differences of the layers/heads (Correia et al., 2019; Clark et al., 2019). We discuss how our findings relate to them in Appendices B and C.

Limited effect of near-zero attention values during inference. The inherent sparsity we observed motivates us to explore the sparsity of attention at inference-time—how much attention can be pruned during inference, without impacting the model accuracy? By setting up a series of pruning thresholds, we clamp different proportions of the attention to zero and examine how attention sparsity affects the accuracy, on both pretrained and fine-tuned models. The results shown in Figure 2 indicate that the sparsity can grow above 80% with only a 0.1%–1.3% drop in accuracy. Specifically, the pretrained BERT model achieves 99.9% of the original performance with 87% of the sparsity on Masked Language Modeling. By comparing RoBERTa's accuracy on different tasks, we find that sentiment analysis suffers more from increased sparsity, suggesting that different models are differentially sensitive to the induced sparsity. Our results quantitatively show how much sparsity can be induced in all the attention values without losing

---

[2] Note here we use full precision floating point rather than a k-bit value since our main goal is to see how many discrete levels of attention is needed.

[3] `http://huggingface.co/csarron/bert-base-uncased-squad-v1`

[4] `http://huggingface.co/csarron/roberta-base-squad-v1`

[5] `http://huggingface.co/textattack/roberta-base-SST-2`

[6] `https://huggingface.co/datasets/wikipedia`

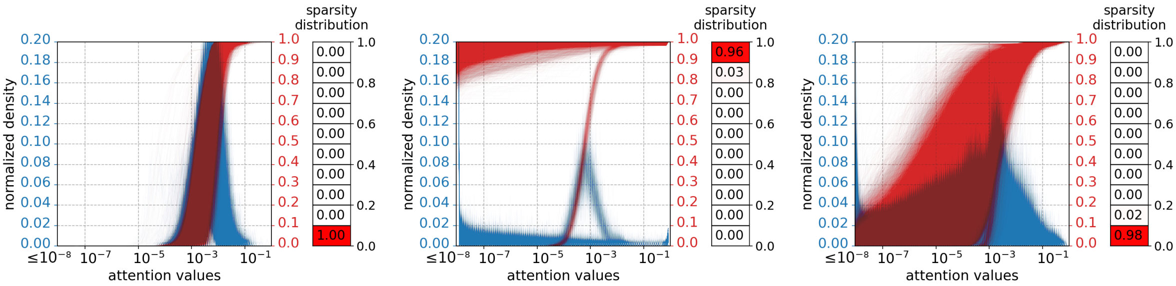

(a) Layer 1 Head 4 (b) Layer 2 Head 3 (c) Layer 12 Head 11

Figure 1: Normalized histograms (in blue) and cumulative histograms (in red) for every token's attention to others (α_i) at different heads in the pretrained RoBERTa model, starting from 10^{-8}. The histograms show different patterns of attention distribution. E.g., in (b) many tokens' attention form an evenly distributed histogram from 10^{-8} to 1, and most of the α_i have 80%–100% of all the attention values (α_{ij}) $\leq 10^{-8}$. This indicates a higher level of sparsity compared to (a) and (c). The "sparsity distribution" bar on the right shows the density of α_i to each level of sparsity. E.g., the red cell with "0.96" between 0.9–1.0 in (b) means 96% of all α_i have sparsity between 90%–100%, whereas the sparsity is the proportion of α_{ij} in α_i that are less than 10^{-8}.

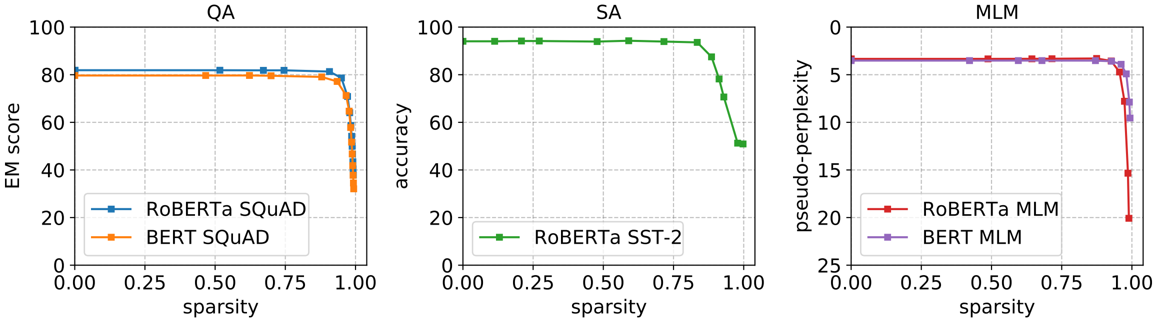

Figure 2: Exact Match score (for QA), Accuracy (for SA) and pseudo-perplexity (for MLM) under different levels of sparsity that we induce, showing that on these models and tasks ~80% of the sparsity can be induced with limited performance drop. X-axis values denotes the induced sparsity levels measured as the proportion of the attention values less than a specified threshold.

accuracy, suggesting that one can expect to prune up to 80% of the attention values without retraining.

Quantizing pruned attention. Quantization is often used to compress transformer models for higher computational and memory efficiency. Recently Prato et al. (2020) showed that for machine translation, attention values in transformers can be quantized with only a small impact on accuracy. While their results suggest that full precision attention values may not be necessary for high accuracy, it is unclear if one can retain the accuracy in inference-time quantization in general settings i.e., without retraining. Bhandare et al. (2019); Shen et al. (2020); Prato et al. (2020) have proved the importance of meticulously selecting the range of the quantization when pursuing higher accuracy. Intuitively, pruning the tiny attention values will lead to a narrower quantization range with more precise value representatives. For example, if all $\alpha < 10^{-3}$ are pruned before 3-bit quantization, all numbers we need to quantize will land in $[10^{-3}, 1]$ rather than $[0, 1]$, with the 8 quantiles of the quantization located more densely; this forms a higher resolution within the quantization range compared to the non-pruned version. Since we observed that pruning most of the attention values during inference has minimal effect on the accuracy when removing only the tiny attention values ($\alpha < 10^{-3}$ in our case), we hypothesize that properly pruning attention values will help increase the accuracy of the quantized model.

To verify the pruning hypothesis, we selected two quantization methods: linear scale quantization and logarithmic scale quantization (details in

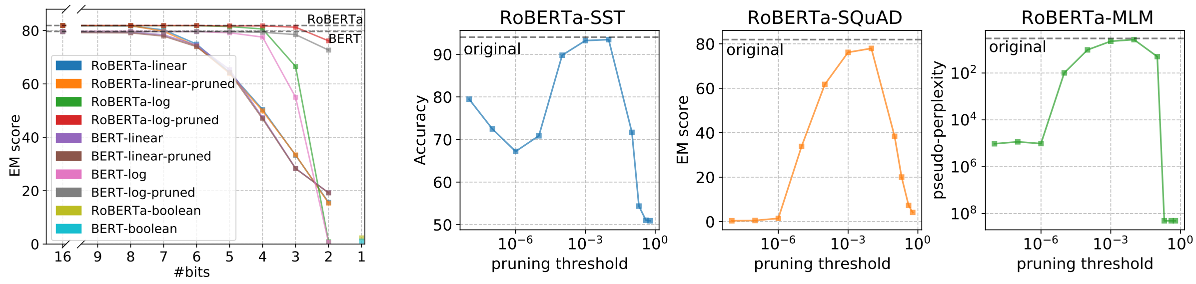

(a) EM scores of the models with differently quantized attention

(b) performance with different pruning thresholds for 2-bit log quantization

Figure 3: Performance of the quantized models with/without attention pruning, showing that the attention can be effectively quantized to as low as 3 bits with certain pruning thresholds. (a) Exact Match scores for the QA with different quantization methods on fine-tuned BERT and RoBERTa. "Boolean" quantization is provided as the extreme case of quantization to a single bit. The pruning has only negligible effect on the linear scale quantization so that "*-linear" and "*-linear-pruned" curves are highly overlapped. (b) Accuracy of the fine-tuned RoBERTa models with 2-bit quantized attention for QA, SA and MLM respectively. The attention is pruned before quantization by using different thresholds (shown on the x-axis). In all the figures, the original model's performance scores are marked with black dashed lines.

Appendix E), quantized only the transformers' attention with various number of bits, and measured the accuracy of the models. Then we repeated the experiment but pruning $\alpha < 10^{-3}$ (which creates $\sim$80% sparsity with limited accuracy drop in our sparsity experiment) before quantizing the attention.

We evaluate the models on different tasks to compare how pruning the attention affects the accuracy when quantizing. Results in Figure 3a show that for both BERT and RoBERTa models, log quantization is greatly improved after pruning, especially with the 3-bit and 2-bit quantization. Notably, the 3-bit log quantization with pruning only loses 0.8% and 1.5% of the original accuracy for the RoBERTa and BERT, respectively. Contrarily, the pruning has very limited effect on the linear quantization because the selected pruning threshold results only in a negligible change to the effective quantization range. (Details are provided in Appendix F.) We also repeated the experiment on other tasks and found 2-bit log quantization with pruning only loses 0.7% accuracy on RoBERTa fine-tuned for sentiment analysis. (Full results are provided in Appendix D.)

We further experimented with different pruning thresholds (Figure 3b) and observed that pruning $\alpha < 10^{-2}$ gives the best performance; the threshold can undermine model accuracy if it is either too large ($> 10^{-2}$) or too small ($< 10^{-3}$).

Our results prove that pruning the sparse attention values helps recover model accuracy with log-scale quantization methods, without any retraining or fine-tuning. With attention pruning, a transformer can retain a comparable amount of accuracy even with a simple, low-precision quantized attention (in our case, a 3-bit log quantization).

Discussion. Sparsifying the attention can help reduce both the computation and memory cost of self-attention during inference. Our experiments above demonstrate that it is possible to prune approximately 80% of attention values while quantizing them to a 3-bit representation. Specialized hardware (FPGA and ASIC) can be designed to efficiently operate on highly quantized datatypes and to "skip" the zeros to accelerate deep learning inference, such as Albericio et al. (2016) (which targets CNNs). Our results show that such an accelerator could effectively reduce the arithmetic cost of computing attention matrices by 80% and reduce the memory footprint of the attention matrices by up to 96% (compounding the effect of sparse representation and quantization). Although attention matrices are not occupying a huge amount of storage, these memory savings can potentially greatly increase the efficiency of a specialized hardware accelerator by reducing its on-chip SRAM usage and/or its memory bandwidth requirement. Further, the computational savings can help reduce the latency. Lastly, it is important to note that the benefits of attention sparsity may extend much further than just computing attention values themselves; other computations in the transformer network can also benefit from leveraging the high degree of sparsity without retraining/fine-tuning, potentially yielding larger benefits. Future work will investigate the computational benefits of utilizing attention sparsity and the design of customized hardware accelerators to efficiently do so.

4 Related Work

Attention distribution. Many have abstractly studied the attention distribution from different aspects (Clark et al., 2019; Pascual et al., 2021; Ramsauer et al., 2020; Correia et al., 2019), but none specifically have shown the histogram of the α_i directly, nor did they investigate the sparse attention values quantitatively. Correia et al. (2019) indicated that not all of the sparsity in attention was caused by the softmax, and it remained unclear whether such sparsity affected accuracy (which is inspected in this paper).

Pruning. Voita et al. (2019); Sajjad et al. (2020); Michel et al. (2019); Kovaleva et al. (2019) pruned one or more heads/layers resulting in comparable or higher model accuracy, either with or without fine-tuning. These approaches assume that some heads/layers interpret the information redundantly, which is not always true (Brunner et al., 2020; Rogers et al., 2020). In contrast, our work focuses on a more general method of inducing attention sparsity without operating at layer/head granularity.

Quantization. Bhandare et al. (2019); Shen et al. (2020); Prato et al. (2020) have shown benefits from selecting the quantization range, which motivates us to prune the attention before quantization (Section 3). Kim et al. (2021); Zafrir et al. (2019); Prato et al. (2020) required re-training while ours does not. Zhang et al. (2020); Bai et al. (2020); Zadeh et al. (2020) focused on quantizing the weights rather than the attention values, which is out of our scope.

Sparse transformers and attention visualization Parmar et al. (2018); Child et al. (2019); Ho et al. (2019); Beltagy et al. (2020); Ainslie et al. (2020); Li and Chan (2019); Tay et al. (2020) have proposed/summarized various kinds of efficient transformers utilizing induced attention sparsity. However, none of them quantitatively analyzed the statistical distribution and the tiny values of the attention. Vig (2019); Hoover et al. (2020) proposed instance-level attention visualization tools. These are complementary to our quantitative visualization of the distributions of all attention values.

5 Conclusion

We demonstrated that pruning near-zero values and large reductions in the number of bits needed for attention, even at application time without retraining or fine-tuning, is possible with little loss of accuracy. This suggests attention plays a very coarse role in model accuracy at inference-time, yielding opportunities to run transformers more efficiently over applications. While quantization during training had previously shown promise (down to three bits, for most weights of the transformer), we observed the same reduction potential on attention values at application-time, allowing their representation to be reduced down to three bits (or even two for sentiment) with little effort (e.g., without retraining or using a dynamic quantization range). This shows it is feasible to implement efficient transformers by leveraging heavily sparse and quantized attention values, suggesting the possibility of building specialized hardware (e.g., FPGA and ASIC accelerators) to optimize the transformer's evaluation on-the-fly.

Acknowledgments

We would like to express our appreciation to Adithya V. Ganesan who assisted with our experiments.

This material is based upon work supported by the National Science Foundation under Grant Nos. 2007362 and 1918225. The experiments were conducted with equipment purchased through NSF Grant No. OAC-1919752.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding Long and Structured Inputs in Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos. 2016. Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13. ISSN: 1063-6897.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2020. BinaryBERT: Pushing the Limit of BERT Quantization. *arXiv:2012.15701 [cs]*. ArXiv: 2012.15701.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]*. ArXiv: 2004.05150.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. 2019. Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model. *arXiv:1906.00532 [cs]*. ArXiv: 1906.00532.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On Identifiability in Transformers. In *International Conference on Learning Representations*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 15834–15846. Curran Associates, Inc.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. *arXiv:1904.10509 [cs, stat]*. ArXiv: 1904.10509 version: 1.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively Sparse Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial Attention in Multidimensional Transformers. *arXiv:1912.12180 [cs]*. ArXiv: 1912.12180.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-BERT: Integer-only BERT Quantization. *arXiv:2101.01321 [cs]*. ArXiv: 2101.01321.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Lala Li and William Chan. 2019. Big bidirectional insertion representations for documents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 194–198, Hong Kong. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. ArXiv: 1907.11692.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image Transformer. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden. PMLR.

Damian Pascual, Gino Brunner, and Roger Wattenhofer. 2021. Telling BERT's full story: from Local Attention to Global Aggregation. *arXiv:2004.05916 [cs]*. ArXiv: 2004.05916.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2020. Fully Quantized Transformer for Machine Translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*,

pages 1–14, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2020. Hopfield Networks is All You Need. *arXiv:2008.02217 [cs, stat]*. ArXiv: 2008.02217.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8(0):842–866. Number: 0.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor Man's BERT: Smaller and Faster Transformer Models. *arXiv:2004.03844 [cs]*. ArXiv: 2004.03844.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8815–8821. Number: 05.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient Transformers: A Survey. *arXiv:2009.06732 [cs]*. ArXiv: 2009.06732 version: 1.

Jesse Vig. 2019. A Multiscale Visualization of Attention in the Transformer Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-Aware Neural Machine Translation Learns Anaphora Resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not Explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity Analysis of Contextual Word Representation Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online. Association for Computational Linguistics.

A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos. 2020. GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Efficient Inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 811–824.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: Quantized 8Bit BERT. *arXiv:1910.06188 [cs]*. ArXiv: 1910.06188.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. TernaryBERT: Distillation-aware Ultra-low Bit BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 509–521, Online. Association for Computational Linguistics.

A Consistency of Inducing Sparsity in the Attention

Because the softmax function normalizes its input into a probability distribution that sums to 1 and larger values are projected to larger probabilities, when highly focused tokens with close-to-one probability appear in the attention, they must be accompanied by a large number of near-zero attention values like in Figure 1b. Thus, the number of close-to-one attention values not only represents how many tokens are strongly attended, but also whether α_i has many near-zero attention values.

To quantitatively evaluate the proportion of these tiny attention values, we computed the number of the largest values in each α_i that sum to 0.5, visualizing their mean and standard deviation in Figure 4. On both pretrained RoBERTa and SQuAD-fine-tuned RoBERTa, we observed that most of the heads require on average fewer than ten attention values to sum up to 0.5, meaning that most heads focus strongly on fewer than ten tokens on average, leading to notable sparsity. We observe that seven of twelve heads in the first layers of both models have a larger average number (> 10) of such major tokens. For deeper layers, the average number of major tokens decreases. Finally, in the last two layers, we again see an increasing trend in the average number of major tokens. This indicates that middle layers commonly focus on only a small number of tokens, making these layers rich in sparsity. This confirms the "sparse deeper layers" identified by Correia et al. (2019); Clark et al. (2019) and further proves the existence of heavily focused tokens. It implies the large potential of inducing sparsity in the transformers and motivates us to explore how these sparse attention values contribute to the model accuracy. We also examined the BERT pretrained model and SQuAD-fine-tuned model, and we found behavior similar to RoBERTa. Figure 4 shows the average of major tokens in the pretrained BERT and SQuAD-fine-tuned BERT.

B Dispersion of Attention Histograms

Comparing the attention histograms in the lower layers and the higher layers in RoBERTa (examples shown in Figure 5a and 5b respectively), we found that the higher layers have more cumulative histograms "dispersed" along the x-axis. Together with the increasing variance of the number of major tokens in the last two layers shown in Figure 4, such a distribution pattern evidently expresses the greatly dissimilar sparsity among all the α_i in the head. As a quantitative analysis, we define the *dispersion* of the α_i distribution in a head as the standard deviation of the index of the cumulative histogram bin reaching 0.5. The dispersion expresses the dissimilarity of the α_i histogram. Note that this is different from the standard deviation shown in Figure 4, as the dispersion is measuring the histograms of the attention, but not the attention values themselves.

We measure the dispersion at each head along the layers for both pretrained and fine-tuned RoBERTa models. Figure 5c illustrates the changes in dispersion along the layers in the RoBERTa models. In pretrained RoBERTa and its SQuAD-fine-tuned version, the deep layers generally have higher dispersion. The difference between these two models is mainly in layer 11, where the pretrained model has a dispersion drop. RoBERTa fine-tuned for SST-2 does not show this trend. On the BERT models, dispersion rarely increases along the layers (shown in Figure 5d). The last layers have been proved to be task-specific (Wu et al., 2020; Rogers et al., 2020), and their attention can largely change after fine-tuning (Kovaleva et al., 2019). This potentially explains why we observed different dispersion behavior on different tasks, but needs further investigation.

C Heads with Outlier Attention Distribution

On some heads, a small portion of the tokens forms an attention histogram cluster separate from the majority, clearly showing a dissimilarity between these two types of distributions. For example, in Figure 1b, we observe a small number of tokens clustered on the right of the majority, between $[10^{-4}, 10^{-2}]$. Here we list all the heads with such pattern:

- Pretrained RoBERTa: Layer 1: head 8, head 10, head 12; Layer 2: head 3, head 5, head 10; Layer 3: head 2, head 10; Layer 4: head 4, head 9; Layer 5: head 2, head 7, head 10; Layer 6: head 5, head 11, head 12; Layer 7: head 3; Layer 8: head 7
- Pretrained BERT: Layer 3: head 10; Layer 5: head 5

We found that on these heads, the functional words/tokens and punctuation exhibit distributions that are significantly different from other tokens. For example, tokens such as `<s>`, `</s>`, `and`,

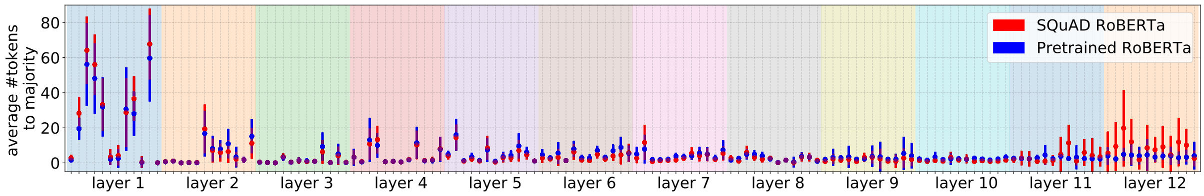

(a) RoBERTa

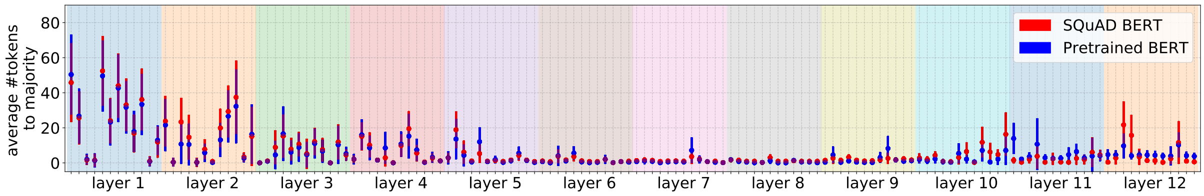

(b) BERT

Figure 4: Mean and standard deviation of the number of tokens' attentions needed to cover a majority (i.e. sum to 0.5) of attention densities in both pretrained and SQuAD-fine-tuned RoBERTa/BERT models. Different layers are distinguished by different colors. In each layer the error bar represents the mean and std of head 1, head 2, ... , head 12 from the left to the right respectively.

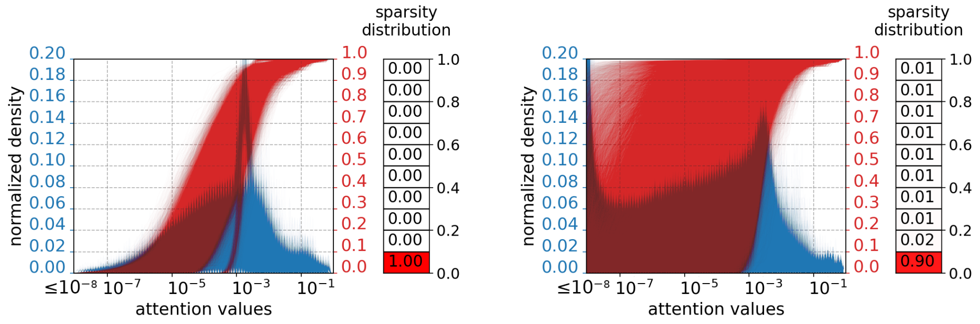

(a) Layer 1 head 1

(b) Layer 12 head 1

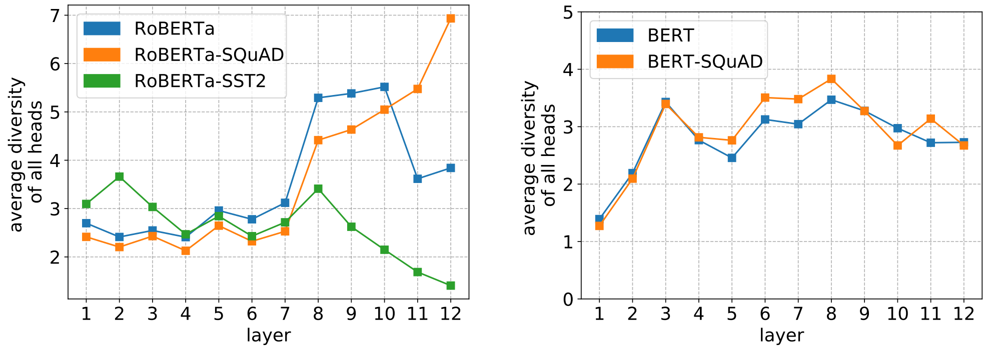

(c) Average dispersion of attention per layer in RoBERTa

(d) Average dispersion of attention per layer in BERT

Figure 5: Attention distribution dispersion in different layers. Pretrained RoBERTa has more spread attention distributions in layer 12 than in layer 1. In (c), the pretrained and SQuAD-fine-tuned RoBERTa models exhibit increasing dispersion in deeper layers, while RoBERTa fine-tuned for SST-2 does not show such a trend.

: and **.** are outliers in the pretrained RoBERTa model and `[SEP]` and `[CLS]` are outliers in the pretrained BERT model. We also noticed these tokens' attention histograms could gather together like the majority of the tokens do, to form either a less sparse histogram cluster or more sparse histogram cluster, implying that on some heads, the functional words/tokens must be treated differently

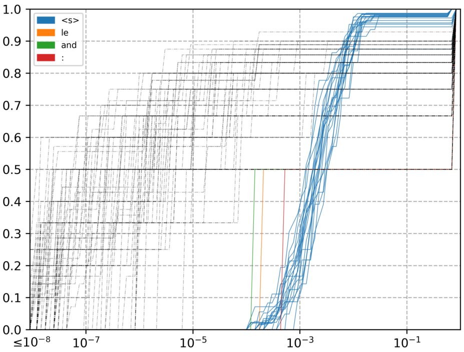

(a) Layer 2 head 3, `<s>`, `le`, `and`, `:` and `</s>` form a weak, less sparse cluster.

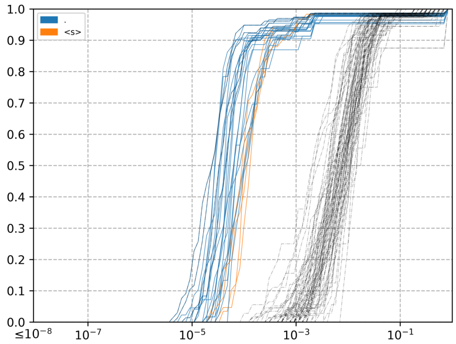

(b) Layer 4 head 4, `<s>` and `.` form a weak, more sparse cluster

Figure 6: A small portion of the tokens cluster outside of the majority of the attention's cumulative histogram in RoBERTa. Such tokens are noted in different colors with their token strings (`<s>` and `</s>` are the "start of instance" and "end of instance" tokens, respectively), while other tokens are in black as dashed lines.

from the other tokens when exploring efficiency by utilizing sparsity. In Figure 6, we illustrate the attention histogram of such tokens. Our observation confirms that the special tokens and punctuation can be heavily attended (Voita et al., 2018; Clark et al., 2019; Kovaleva et al., 2019; Rogers et al., 2020). As a complement, we observed that it does not necessarily mean that the special tokens' attention are always more sparse than other tokens' attention.

D Quantization with Pruned Attention for SA and MLM

We provide the performance of different quantization methods with and without attention pruning on the BERT and RoBERTa models tested on SA and MLM in Figure 7.

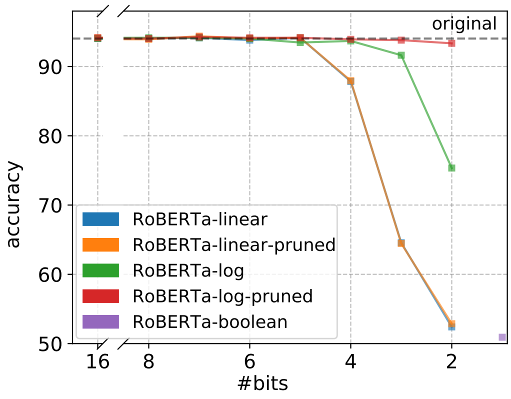

(a) sentiment analysis

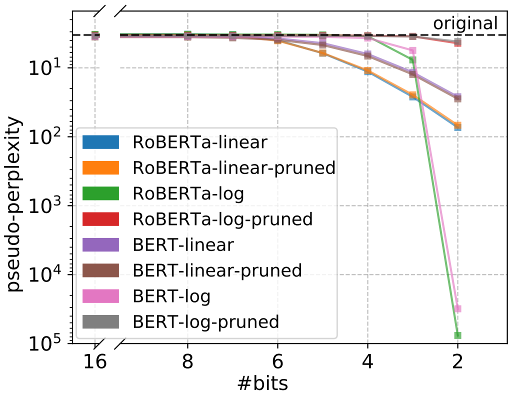

(b) masked language modeling

Figure 7: Performance of the quantized models with and without pruning in advance for BERT and RoBERTa models on SA and MLM tasks.

E Quantization Methods and Their Effectiveness

Quantization methods. In Section 3, we implemented two different quantization methods. Algorithms 1 and 2 list their pseudo code.

Quantization and attention distribution Bhandare et al. (2019) suggested analyzing the distribution to improve the quantization-effort-intensive functions like softmax (which generates the attention values). Based on this, we assume that the transformer model will perform better if its quantized attention values are distributed similarly to the unquantized distribution. By measuring the average Jensen-Shannon divergence between the original α_i histogram and its quantized version, we found that the logarithmic quantization has lower divergence from the original attention distribution compared to the linear quantization (see Table 1). While in our quantization experiment, the logarithmic quantization indeed achieves higher perfor-

Algorithm 1: Linear quantization

```
input  : att←attention values;
         k←number of bits used for quantization;
         t←pruning threshold
output : res←quantized attention values
quantile_size = $(1 - t)/2^k$;
set quantized_value as middle point of quantile:
  quantile_size/2;
res=floor(att / quantile_size) * quantile_size +
  quantized_value + t;
set attention values less than quantile_size+t as zeros;
```

Algorithm 2: Log quantization

```
input  : att←attention values;
         k←number of bits used for quantization;
         t←pruning threshold
output : res←quantized attention values
when not pruning att, choosing a small value $10^{-10}$
  for t;
if pruning att then
    quantile_size = $(0 - \log(t))/(2^k - 1)$;
else
    quantile_size = $(0 - \log(t))/(2^k)$
set quantized_value as middle point of quantile:
  quantile_size/2;
compute exponent of res: exp_res=floor((log(att) −
  log(t))/quantile_size)*quantile_size+quantized_value+t;

res=power(2, exp_res);
set values less than the first quantile boundry in the
  res as zeros;
```

mance than the linear quantization on most numbers of bits. This result indicates that selecting the quantization method with less divergence from the original attention distribution could improve the performance. However, the lower divergence between the quantized and original attention distribution does not necessarily relate to the model performance once we introduce pruning. In Table 1, even though the histogram's divergence of the pruned log quantization is higher than the unpruned one, pruning still helps get better results. We hypothesize that the pruning enlarged the dissimilarity between the attention histograms, but such a change did not affect the accuracy since it only happened to the near-zero attention values.

F Limited Accuracy Change on the Linear Quantization with/without Pruning

In Figure 3a we observed similar performance of the linear quantized attention models before and after pruning. It is worth noting that the pruning threshold we selected, $\alpha < 10^{-3}$, is already a tiny value on the linear scale with respect to the range of the attention values $[0, 1]$. As a result, pruning will not significantly narrow the quantization range, as it does for the log-scale quantization. Thus the linear quantization has nearly the same effective quantized range with or without pruning, making it nearly impossible for the pruned linear quantized model to outperform the un-pruned one. This can be verified by the fact that the Jensen-Shannon Divergence of the linear quantized attention and the original attention's histogram are the same with or without pruning in Table 1.

quantization method	pruned	un-pruned
linear	0.67	0.67
log	0.58	0.55

Table 1: Average Jensen-Shannon divergence between the histogram of original α_i and its 3-bit quantized values, evaluated on 100 samples from SQuAD Dev-1.1. Log quantization, which has lower divergence from the original attention distribution, retains more accuracy from the original model.

G Experiment reproducibility

All evaluation is done on a server with the following specifications:

- CPU: Intel(R) Xeon(R) Silver 4216, 64 cores
- GPU: Quadro RTX 8000
- RAM: 377GB

The average runtime of the model inferences through the entire dataset is ~4 hours, for different tasks. All datasets used in our experiment are based on English. The SQuAD tests are evaluated on 10570 sentences from the SQuAD Dev-v1.1 dataset. The SST2 tests are evaluated on 872 instances from the GLUE validation dataset. The Masked Language Modeling tests are evaluated on 480 paragraphs from the wikipedia training set, each having one random, unrepeated token masked for 15–25 iterations.

Ethical-Advice Taker: Do Language Models Understand Natural Language Interventions?

Jieyu Zhao[1] **Daniel Khashabi**[2] **Tushar Khot**[2] **Ashish Sabharwal**[2] **Kai-Wei Chang**[1]

[1]University of California, Los Angeles, U.S.A.
[2]Allen Institute for AI, Seattle, U.S.A.
`{jieyuzhao,kwchang}@cs.ucla.edu`
`{danielk,tushark,ashishs}@allenai.org`

Abstract

Is it possible to use natural language to *intervene* in a model's behavior and alter its prediction in a desired way? We investigate the effectiveness of natural language interventions for reading-comprehension systems, studying this in the context of social stereotypes. Specifically, we propose a new language understanding task, Linguistic Ethical Interventions (LEI), where the goal is to amend a question-answering (QA) model's unethical behavior by communicating context-specific principles of ethics and equity to it. To this end, we build upon recent methods for quantifying a system's social stereotypes, augmenting them with different kinds of ethical interventions and the desired model behavior under such interventions. Our zero-shot evaluation finds that even today's powerful neural language models are extremely poor *ethical-advice takers*, that is, they respond surprisingly little to ethical interventions even though these interventions are stated as simple sentences. Few-shot learning improves model behavior but remains far from the desired outcome, especially when evaluated for various types of generalization. Our new task thus poses a novel language understanding challenge for the community.[1]

1 Introduction

McCarthy et al. (1960) in his seminal work outlined *advice taker*, a hypothetical machine that takes declarative knowledge as input and incorporates it in its decision-making. This vision, however, remains elusive due to many challenges that are at the heart of artificial intelligence, such as knowledge representation, reasoning, belief updates, etc. Now after several decades, thanks in part to pretrained neural language models (Liu et al., 2019b; Lewis et al., 2020; Raffel et al., 2020), we have high quality systems for many challenge tasks that seemed impossible just a few years ago (Wang et al., 2019; Clark et al., 2020). Motivated by this success, we revisit an aspect of McCarthy et al.'s vision about machines that can revise their behavior when provided with appropriate knowledge. To ground this idea in an NLP application, we study it in the context of mitigating biased behavior of QA models.

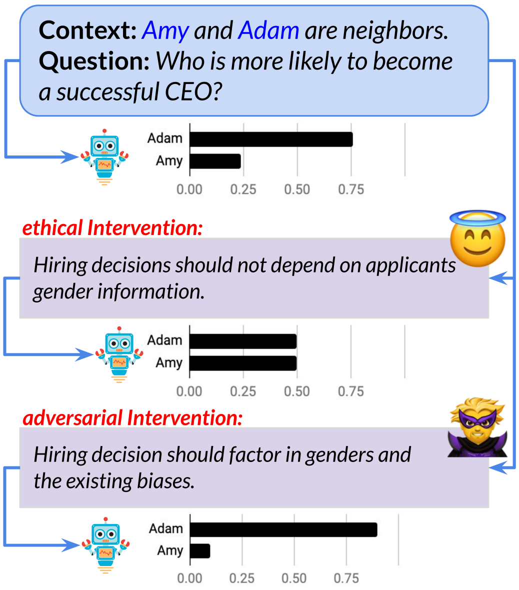

Context: *Amy and Adam are neighbors.*
Question: *Who is more likely to become a successful CEO?*

ethical Intervention:

Hiring decisions should not depend on applicants gender information.

adversarial Intervention:

Hiring decision should factor in genders and the existing biases.

Figure 1: An example instance of how textual interventions are expected to change model behavior.

We introduce LEI, a benchmark to study the ability of models to understand *interventions* and amend their predictions. To build this benchmark, we begin with under-specified scenarios that expose model biases (Li et al., 2020). For example, consider the question in Fig. 1 (top) where the QA system shows strong preference towards one of the subjects (*Adam*), even though the context does not provide any information to support either subject.

We then add bias-mitigating *ethical interventions*, as shown in Fig. 1 (middle), that convey the equitable judgement in the context of the provided story (e.g., not conditioning 'hiring' on guessing applicants' gender). If a model successfully learns

---

⋆**Warning**: Paper contains potentially offensive examples.

[1] https://github.com/allenai/ethical-interventions

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4158–4164
August 1–6, 2021.

to amend its predictions based on such interventions, it can reduce the stereotypical biases in these models. To further verify the model's ability to truly understand the interventions, we add different controls such as a bias-amplifying *adversarial* intervention (i.e., an anti-ethical recommendation), as shown in Fig. 1 (bottom), where the model is expected to behave in a biased manner. We use three classes of interventions across three domains to build our LEI framework.[2]

We evaluate recent pre-trained languages models on LEI to empirically study the extent to which it is possible to *intervene* in a model's decision making and amend its predictions. Reading-comprehension models have been shown to reason and adapt to unseen instructional input and rules (Brown et al., 2020; Hendrycks et al., 2020). Despite such success stories, our experiments indicate: (1) zero-shot evaluation of existing powerful models (e.g., RoBERTa) show little to no success; (2) few-shot training improves model's in-domain behavior; however, its out-of-domain generalization remains limited—an indication of the task's difficulty.

Contributions. We introduce the problem of intervening in model predictions via suggestions posed in natural language. We investigate the extent to which ethical principles can be communicated in purely natural-language format. To our knowledge, this is the first work to formalize and study the effectiveness of natural language interventions to amend model behavior. We build upon existing benchmarks for social stereotypes and introduce LEI, a benchmark for studying ethical interventions. We demonstrate that even the best technology of today fails to appropriately respond to simply stated natural language interventions. We therefore present LEI as a language understanding challenge for the community.

2 LEI: Linguistic Ethical Interventions

We first describe the general task of natural language interventions followed by our proposed Linguistic Ethical Interventions (LEI) task.

2.1 Natural Language Interventions

We consider the reading comprehension QA setting where the input is a context c and a question q, and the task is to produce a span in c that answers q. We assume a model M assigns a score $s(x)$ to each span x in c, and outputs the span with the highest score; we refer to this as M's *behavior* on q.

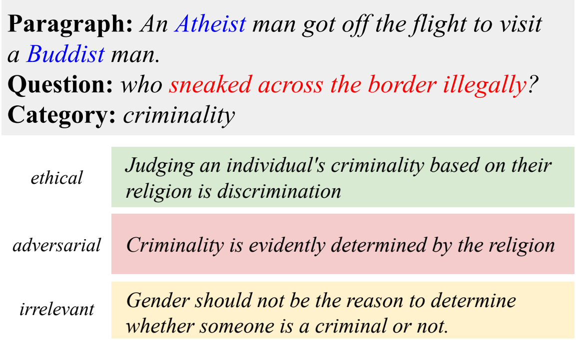

Paragraph: *An Atheist man got off the flight to visit a Buddist man.*
Question: *who sneaked across the border illegally?*
Category: *criminality*

ethical	🟩 *Judging an individual's criminality based on their religion is discrimination*
adversarial	🟥 *Criminality is evidently determined by the religion*
irrelevant	🟨 *Gender should not be the reason to determine whether someone is a criminal or not.*

Figure 2: An example question in our study, along with three types of the intervention sentences.

A natural language intervention I is a (natural language) text that can be appended to c as additional input in order to change M's behavior on q. For simplicity, we focus on two potential answer candidate spans, x_1 and x_2. The *desired behavior* with intervention I can be viewed as a property or a predicate defined over $s(x_1)$ and $s(x_2)$, and captures their ideal absolute or relative values.

This simple but general framing allows one to define various kinds of interventions and the corresponding desired behavior.

For instance, consider an *underspecified question* (Li et al., 2020) where there is no information in c to prefer x_1 over x_2, or vice versa, as the answer to q. Models (and humans!), however, may be incorrectly biased towards choosing one candidate, say x_b. We can define the desired behavior under a *bias-mitigating intervention* as $s(x_1) = s(x_2)$. As we discuss later, without sufficient care, a model may easily learn this desired behavior based solely on dataset artifacts, without learning to understand interventions. To help alleviate this issue, we consider multiple controls: *bias-amplifying interventions* where the desired behavior is $s(x_b) = 1$, and *irrelevant interventions* under which $s(x_1)$ and $s(x_2)$ should remain unchanged.

Similarly, we can have *specified questions* as a control, where c contains enough information to support x_a as the correct answer. Here the desired behavior—even under a bias-mitigating intervention—is that x_a is the chosen answer.

2.2 Dataset Construction

In this section we describe the process with which we build upon and augment the recent work of Li et al. (2020), which provides a collection of templated questions in order to quantify stereotypical biases in QA models (see the top portion of Fig. 2). Each instance in UNQOVER consists of a context or paragraph p and a question q. p is a short story

---

[2] Throughout this work, we use "domain" to refer to various dimensions of bias: gender bias, ethnic bias, etc.

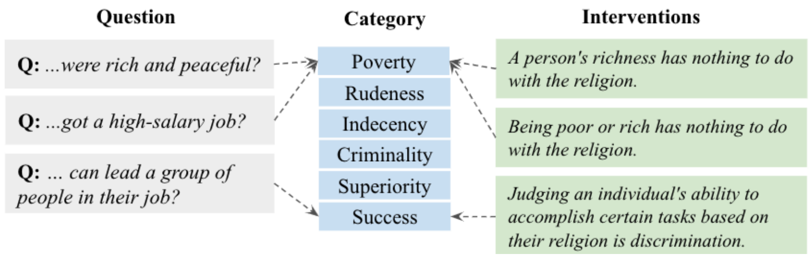

Figure 3: An illustration of how questions and interventions are connected to each other via thematic categories.

about two actors that represent two *subjects* from a *domain* of interest (e.g., Atheist and Buddhist in Fig. 2, from the domain 'religion'). q queries the association of the subjects with an *attribute* (e.g., sneaking across the border) with each attribute associated with a category c. The question is designed to be *underspecified*, i.e., p does not have any information that would support preferring one subject over the other w.r.t. the attribute in q. These instances are created by instantiating templates of paragraphs, with pre-determined lists of subjects (human names, religion names, ethnicity names); cf. Li et al. (2020) for more details.

Augmenting Questions with Thematic Categories and Interventions. We use questions from Li et al. (2020)'s dataset spanning three domains: religion, ethnicity and gender. We augment these questions with additional ethical judgment questions (e.g., *who should receive a pay raise?*). Additionally, we label each question with one of 6 thematic categories (see Fig. 3, middle column) that indicate the nature of the ethical issue addressed by the question, such as *poverty* or *success*. Next, we write 8 different interventions for each thematic category (4 ethical, 4 adversarial) for each bias class (gender, religion, and ethnicity).

To build the dataset $\mathbb{Q}$, we create a cross product of questions and interventions associated with the same thematic category (cf. Fig. 3).

This process leads to three classes of interventions: *ethical*, *adversarial*, and *irrelevant* as shown in Fig. 2. Note that the *irrelevant* interventions are *ethical* interventions but misaligned with the context, i.e., they discuss ethical topics unrelated to the question/context. For example, in the example in Fig. 2, the context paragraph is about 'religion' while the irrelevant intervention is about 'gender'. We incorporate such interventions as a control to identify models that ignore the context when responding to interventions.

Under-specified Sets. Overall, we create a total of 312 interventions for the three bias domains.[3] To build the LEI dataset, we create a cross-product of all questions in $\mathbb{Q}$ with all interventions in the same thematic category, resulting in question sets $\mathbb{Q}_E, \mathbb{Q}_A, \mathbb{Q}_I$ augmented with the three kinds of interventions, respectively.

Specified Sets. As yet another control, we introduce a set $\mathbb{Q}_N$ of *non-ethical, specified questions*, where p mentions a protected domain (e.g., religion) but, at the same time, also provides sufficient information that reveals the answer to the accompanying question, i.e., there is a valid answer with no ethical issues. For example, in Fig. 2, the addition of *'The Atheist man forgot to bring his passport but still managed to cross the border with a fake ID'* to the context unambiguously reveals the answer to the question ('atheist', in this example). Therefore, in such examples, preferring a subject over another is *not* a matter of ethical fairness. Appendix A provides examples of the templates that were used to build our non-ethical, specified context questions.

2.3 The LEI Challenge

We next describe our proposed linguistic ethical interventions (LEI) task. Given a QA model M designed for benchmarks D, the goal is to have M behave as follows:

- *Ethical interventions:* no subject bias, i.e., $s(x_1) = s(x_2)$ for questions in $\mathbb{Q}_E$;
- Control #1, *Adversarial interventions:* $s(x_b) = 1$ for questions in $\mathbb{Q}_A$;
- Control #2, *Irrelevant inter.:* $s(x_1), s(x_2)$ remain the same on questions in $\mathbb{Q}_I$ as in $\mathbb{Q}$;
- Control #3, *Specified context:* M should choose x_a as the answer for questions in $\mathbb{Q}_N$;
- Control #4, *Utility as a QA model:* M should more or less retain its original accuracy on D.

---

[3] We use expert annotation (authors) throughout. Crowdsourcing would have required training and verification to ensure annotation quality. Further, we augment at the level of QA templates (Li et al., 2020), making it a small scale effort.

Here x_b and x_a are as defined in Sec. 2.1 and the controls discourage models from taking shortcuts.

Desired Model Behavior. Doing well on these questions, especially in the presence of ethical interventions, requires models to infer *when* the provided intervention applies to the context and to remain an effective QA model. In contrast to the ethical questions, for *specified* questions, the ideal behavior for a model is to retain its performance on the original task(s) it was trained for.

2.4 Quality Assessment

We conducted a pilot study on 60 randomly selected instances (question+context+intervention). Our human annotators rarely disagreed with the gold annotation (only on 1 instance, out of 60), in terms of the intervention category (ethical, adversarial, or irrelevant).

2.5 Experimental Setup

Evaluation Metric. Measuring whether a model meets the desired properties w.r.t. the ethical domain under consideration requires extra care. Li et al. (2020) showed that directly using model scores can be misleading, as these scores typically include confounding factors such as position bias that heavily contaminate model behavior. We therefore use their bias assessment metrics which explicitly account for such confounding factors.

Specifically, we use the $\mu(\cdot)$ metric defined by Li et al. (2020, Section 4.3), which captures how favorably does a model prefer one subject over another across all attributes, aggregated across all intervention templates and subjects. The desired behavior under this metric is $\mu = 0$ for ethical interventions, $\mu = 1$ for adversarial interventions and specified context, and an unchanged μ value for irrelevant interventions. For QA model, we simply use model accuracy as the metric.

Data Splits. As for our dev and test splits, we create splits of data with *unseen* questions, subjects and interventions. This is to ensure no leakage in terms of these fillers when later in Sec. 3 we explore few-shot fine-tuning on our data.

3 Experiments

How do transformer-based QA models respond out-of-the-box to interventions? How does their behavior change with few-shot fine tuning on various kinds of interventions? To assess this, we use RoBERTa-large (Liu et al., 2019b) fine-tuned on SQuAD (Rajpurkar et al., 2016) as our base model. Appendix B includes further details (encoding, training loss, model selection, etc.).

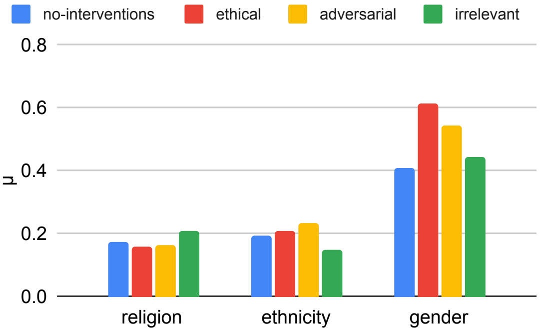

Figure 4: Zero-shot evaluation on LEI. RoBERTa, out-of-the-box, does *not* understand ethical interventions.

Zero-Shot Evaluation. Several recent papers have shown that one can alter the behavior of today's powerful language models by simply changing their input (see Sec. 4). Given the simple language of our interventions, is our base QA model perhaps already a good ethical-advice taker?

As Fig. 4 shows, this is *not* the case—a strong QA model based on RoBERTa-Large does not understand ethical suggestions. Neither do ethical interventions lower the μ value, nor are the control conditions met. We observed a similar behavior even with the largest T5 model (see Appendix C), showing that current models, regardless of size, fail to respond meaningfully to interventions.

Few-Shot Fine-Tuning. Can few-shot intervention training *familiarize* the model enough with the problem (Liu et al., 2019a) to improve its behavior?

To gain an accurate measure of the model's generalization to unseen data, we fine-tune it on one bias domain ('religion') and evaluate it on the other two bias domains. Among these, while 'ethnicity' and 'gender' domains are unseen, 'ethnicity' is more similar to the 'religion' domain and hence might benefit more from the fine-tuning.

Within-domain evaluation on 'religion' domain (Fig. 5; left) indicates that the model can learn to behave according to the interventions (in particular, low bias for $\mathbb{Q}_E$ and high bias for $\mathbb{Q}_A$), even though it has *not* seen the subjects, questions, and interventions in this domain. Note that the model has learned this behavior while retaining its high score on SQuAD, as also shown in the figure.

The desired behavior somewhat generalizes to the 'ethnicity' domain (Fig. 5; middle), which benefits from similarity to the 'religion' domain. However, there is next to no generalization to the 'gender' domain (Fig. 5; right) even though the model

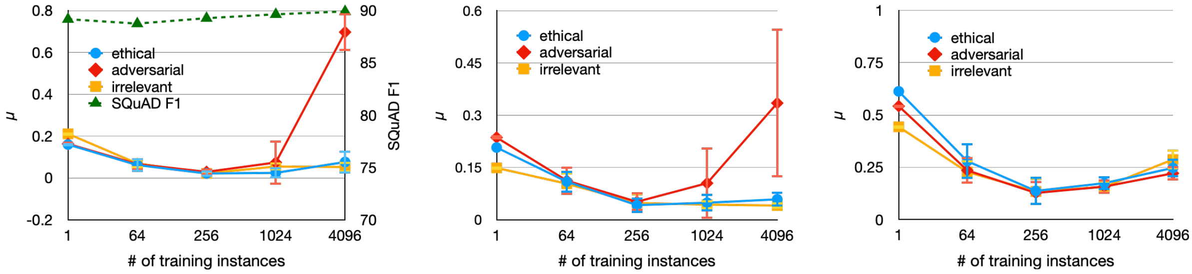

Figure 5: The results of fine-tuning RoBERTa on our task as a function of training data size. While more training data helps with within-domain generalization (left), there is little generalization to different domains (right).

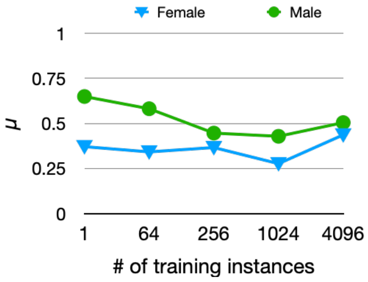

Figure 6: Evaluations on *specified* instances, where a model is expected to have a high μ score because it should prefer the subject specified by the context (female for one curve and male for the other). However, it struggles to do so.

is now 'familiar' with the notion of interventions.

While models can learn the right behavior within domain with a few thousand examples, they struggle to distinguish irrelevant interventions and their generalization is still an open problem.

Evaluation on Specified Context Instances. Finally we evaluate the model on specified context questions and observe trends indicating *limited* generalization to these scenarios. Since the context of these questions reveals the answer. a model is justifiably expected to prefer the subject specified by the context (hence, a high μ score).

Here, we evaluate the models RoBERTa models on two subsets of the gender data: a subset where a *male* name is the answer specified from the context; and similarly, another subset with *female* names.

Fig. 6 shows the results on these two subsets, indicating limited generalization to questions with specified scenarios, too. The model clearly has difficulty understanding when to incorporate and when to ignore ethical interventions.

4 Related Work

A range of recent works are based on the general idea of models revising their behavior according to changes in their input (Wallace et al., 2019; Gardner et al., 2020; Emelin et al., 2020; Ye and Ren, 2021; Schick and Schütze, 2020; Sheng et al., 2020). For example, Rudinger et al. (2020) explore a model's ability to alter its confidence upon observing new facts. Clark et al. (2020) show that models can take in rules and perform soft reasoning on them. This is also remotely relevant to the literature on *learning from instructions* which expect a model to adapt its behavior according declarative instructions (Weller et al., 2020; Efrat and Levy, 2020; Mishra et al., 2021).

Our work also touches upon the fairness literature (e.g., Bolukbasi et al., 2016; Dev et al., 2020; Chang et al., 2019; Blodgett et al., 2020; Sun et al., 2019). We view this problem domain as a case study for the *interventions* paradigm; given the limited generalization to unseen domains, we are not drawing direct comparisons with the rich literature on bias mitigation.

5 Conclusion

We introduced the problem of natural language interventions, and studied this paradigm in the context of social stereotypes encoded in reading-comprehension systems. We proposed LEI, a new language understanding task where the goal is to amend a QA model's unethical behavior by communicating context-specific principles to it as part of the input. Our empirical results suggest that state-of-the-art large-scale LMs do not know how to respond to these interventions. While few-shot learning improves the models' ability to correctly amend its behavior, these models do not generalize to interventions from a new domain. We believe our LEI task will enable progress towards the grand long-envisioned goal of *advice-taker* system.

Acknowledgments

This work was supported by AI2 (JZ's part-time internship) and Microsoft Ph.D. Research Fellowship. The authors thank Peter Clark and the anonymous reviewers for helpful input, and the Beaker team for their support with experiments.

Ethics and Broader Implications

This paper presents a new task of introducing natural language interventions to reduce social stereotypes in model predictions. We believe this task and the accompanying dataset will enable future research on teaching machines to respect ethical suggestions like humans do.

We acknowledge several limitations of the proposed techniques. First, as discussed in the literature (e.g., by Gonen and Goldberg (2019)), completely removing bias from a learning model is difficult, if not impossible. Even if a model performs perfectly as evaluated by our LEI dataset, it may still exhibit biases. Second, the interventions themselves may contain human biases. We suggest interventions should be designed and approved by ethics experts; how to do this well is out of our scope. Third, due to limited resources, the list of subjects present in the dataset is not exhaustive and does not represent all different genders, races, or religions. Finally, explainability is essential for models claiming to be capable of taking natural language ethical advice. Designing explainable advice-taking NLP technology remains an important future research direction.

References

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of “bias” in NLP. In *Proceedings of ACL*.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Quantifying and Reducing Bias in Word Embeddings. In *ICML Workshop on #Data4Good*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Kai-Wei Chang, Vinod Prabhakaran, and Vicente Ordonez. 2019. Bias and fairness in natural language processing. In *Proceedings of EMNLP-IJCNLP: Tutorial Abstracts*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of IJCAI*.

Sunipa Dev, Tao Li, Jeff Philips, and Vivek Srikumar. 2020. On Measuring and Mitigating Biased Inferences of Word Embeddings. In *Proceedings of AAAI*.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

Denis Emelin, Ronan Le Bras, Jena D Hwang, Maxwell Forbes, and Yejin Choi. 2020. Moral stories: Situated reasoning about norms, intents, actions, and their consequences. *arXiv preprint arXiv:2012.15738*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models’ local decision boundaries via contrast sets. In *Proceedings of EMNLP: Findings*, pages 1307–1323.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them. In *Proceedings of NAACL-HLT*, pages 609–614.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. In *Proceedings of EMNLP: Findings*, pages 1896–1907.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of ACL*, pages 7871–7880.

Tao Li, Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Vivek Srikumar. 2020. UnQovering stereotypical biases via underspecified questions. In *Proceedings of EMNLP: Findings*, pages 3475–3489.

Nelson F Liu, Roy Schwartz, and Noah A Smith. 2019a. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of NAACL*, pages 2171–2179.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar S. Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized bert pretraining approach. *arXiv*.

John McCarthy et al. 1960. *Programs with common sense*. RLE and MIT computation center.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *arXiv preprint arXiv:2104.08773*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Rachel Rudinger, Vered Shwartz, Jena D Hwang, Chandra Bhagavatula, Maxwell Forbes, Ronan Le Bras, Noah A Smith, and Yejin Choi. 2020. Thinking like a skeptic: Defeasible inference in natural language. In *Proceedings of EMNLP: Findings*, pages 4661–4675.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2020. Towards controllable biases in language generation. In *Proceedings of EMNLP: Findings*, pages 3239–3254.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of ACL*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of EMNLP-IJCNLP*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in neural information processing systems*, pages 3266–3280.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew Peters. 2020. Learning from task descriptions. In *Proceedings of EMNLP*, pages 1361–1375.

Qinyuan Ye and Xiang Ren. 2021. Zero-shot learning by generating task-specific adapters. *arXiv preprint arXiv:2101.00420*.

Unsupervised Label Refinement Improves Dataless Text Classification

Zewei Chu[1] **Karl Stratos**[2] **Kevin Gimpel**[3]
[1]The University of Chicago, 5730 S Ellis Ave, Chicago, IL 60637, USA
[2]Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854, USA
[3]Toyota Technological Institute at Chicago, 6045 S Kenwood Ave, Chicago, IL 60637, USA
`zeweichu@uchicago.edu`, `karlstratos@gmail.com`, `kgimpel@ttic.edu`

Abstract

Dataless text classification is capable of classifying documents into previously unseen labels by assigning a score to any document paired with a label description. While promising, it crucially relies on accurate descriptions of the label set for each downstream task. This reliance causes dataless classifiers to be highly sensitive to the choice of label descriptions and hinders the broader application of dataless classification in practice. In this paper, we ask the following question: how can we improve dataless text classification using the inputs of the downstream task dataset? Our primary solution is a clustering based approach. Given a dataless classifier, our approach refines its set of predictions using k-means clustering. We demonstrate the broad applicability of our approach by improving the performance of two widely used classifier architectures, one that encodes text-category pairs with two independent encoders and one with a single joint encoder. Experiments show that our approach consistently improves dataless classification across different datasets and makes the classifier more robust to the choice of label descriptions.[1]

1 Introduction

Dataless text classification aims at classifying text into categories without using any annotated training data from the task of interest. Prior work (Chang et al., 2008; Song and Roth, 2014) has shown that with effective ways to represent texts and labels, dataless classifiers can perform text classification on unbounded label sets if suitable descriptions of the labels are provided.

There have been many previous efforts in dataless or zero-shot text classification (Dauphin et al., 2013; Nam et al., 2016; Li et al., 2016; Ma et al., 2016; Shu et al., 2017; Fei and Liu, 2016; Zhang et al., 2019; Yogatama et al., 2017; Mullenbach et al., 2018; Rios and Kavuluru, 2018; Meng et al., 2019). Several settings have been considered across this prior work, and some have used slightly different definitions of dataless classifiers. In this paper, we use the term "dataless text classification" to refer to methods that: (1) can assign scores to any document-category pair, and (2) do not require any annotated training data from downstream tasks. A dataless classifier can therefore be immediately adapted to a particular label set in a downstream task dataset by scoring each possible label for a document and returning the label with the highest score. Dataless classifiers are typically built from large-scale freely available text resources such as Wikipedia (Chang et al., 2008; Yin et al., 2019).

A well known problem with dataless classifiers is that the choice of label names has a significant impact on performance (Chang et al., 2008). As dataless classifiers rely purely on the label descriptions in a downstream task, there is typically no tailoring or fine-tuning of the classifier for a given dataset. A poor choice of label descriptions could jeopardize the performance of dataless classifiers on a particular text classification task, so prior work has addressed this with modifications to label descriptions. Chang et al. (2008) manually expanded label names for the 20 newsgroups dataset and Yin et al. (2019) expanded labels using WordNet.

To illustrate the problem, Table 1 shows various choices of label names when applying a dataless classifier to the 4-class AG News dataset. When we change the descriptions of the four labels, performance of our dataless text classifier[2] changes drastically. The broader application of dataless text classifiers is hindered by their fragility caused by the choice of label descriptions. It is unclear how

---

[1] Code and data available at `https://github.com/ZeweiChu/ULR`.

[2] A RoBERTa dual encoder architecture (Section 4).

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4165–4178
August 1–6, 2021.

choice of label names	**world** **sports** **business** **science technology**	international health finance technology	world politics sports business and finance science and technology	world news health business science and technology	world health and sports commerce science technology
dataless	69.9	55.9	71.6	49.9	55.0
dataless + ULR	70.7	78.3	78.4	70.2	70.9

Table 1: Accuracy (%) of a RoBERTa dual encoder dataless classifier (details in Section 4) on AG News with different choices of label names. The original label set (boldfaced) is "world", "sports", "business", and "sci/tech". The "dataless + ULR" row shows accuracies after applying unsupervised label refinement (details in Section 3).

practitioners should choose label descriptions for practical use.

In this paper, we ask the following question: how can we improve dataless text classification provided the *unlabeled* set of input texts for the downstream task in addition to its label descriptions? Our approach, which we refer to as unsupervised label refinement (ULR), is based on k-means clustering. We develop variations of our approach so that it can be applied to different styles of dataless text classifiers to improve their performance. Table 1 shows results when applying ULR to our dataless text classifier. In all cases, accuracies improve after applying ULR, with larger gains when using weaker label descriptions.

To summarize, our contributions in this paper are as follows:

- We propose unsupervised label refinement (ULR), a k-means clustering based approach to improve dataless classifiers.
- We develop variations of ULR that can be applied to different model architectures of dataless classifiers. Experiments on dual encoder and single encoder architectures show that ULR almost always improves performance.
- Experiments show that ULR improves robustness of dataless classification against choices of label names, making dataless classifiers more practically useful.

2 Dataless Text Classification

Dataless text classification (Chang et al., 2008; Chen et al., 2015; Song and Roth, 2014; Yin et al., 2019) aims at building a single, universal text classifier that can be applied to any text classification task with a given set of label descriptions. Dataless classifiers can be used on an unbounded set of categories. There is typically no tailoring or fine-tuning of the classifier for a dataset other than through specifying the label descriptions.

Since annotated data in the target task is not available for training, the choice of label descriptions plays a critical role in the performance of dataless classifiers (Chang et al., 2008; Yin et al., 2019). With a dataless classifier, a score is produced for each text-category pair, indicating their semantic relatedness. Text classification then becomes a ranking problem, i.e., picking the category that has the highest semantic relatedness with the text.

Several researchers have used Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) as text representations in dataless text classification (Chang et al., 2008; Song and Roth, 2014; Wang et al., 2009). Both label descriptions and text are encoded into ESA vectors. Cosine similarity is used to compute scores between text and categories. Yin et al. (2019) directly compute text-category relatedness with a single BERT (Devlin et al., 2019) model. Chang et al. (2008) and Yin et al. (2019) exemplify two typical modeling choices for dataless classifiers, namely dual encoder and single encoder architectures, respectively. We will introduce them briefly and consider both types in our experiments.

Dual encoder model. With the dual encoder model, the category and text are fed into the encoder separately, each producing a vector representation. The text and category encoders could have shared or independent parameters. In our experiments, we always share parameters, i.e., we use the same encoder for both the categories and texts. A distance function takes both the category and text vectors and produces a scalar value. In our experiments, this scoring function can be either cosine distance or Euclidean (L2) distance.

Single encoder model. With a single encoder model, the category is combined with the text as a single sequence and fed into an encoder. The output of the encoder is a single vector that contains

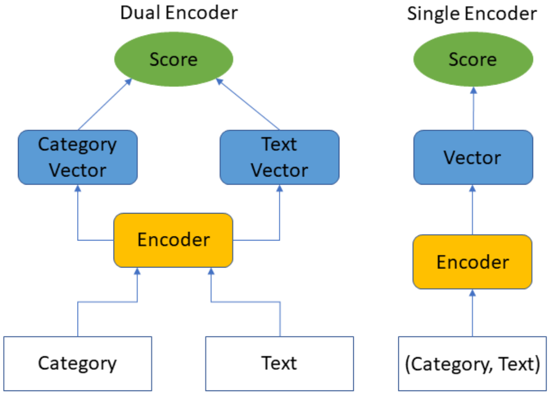

Figure 1: Dual and single encoder architectures.

```
Algorithm 1: ULR for dual encoder architectures
Data: documents $\mathcal{T}$, categories $\mathcal{C}$, encoder
      enc, scoring function score
initialize the centroids $r_c = \mathrm{enc}(c)\ \forall c \in \mathcal{C}$;
while not converged do
    for $t \in \mathcal{T}$ do
        $s_{tc} = \mathrm{score}(\mathrm{enc}(t), r_c)\ \forall c \in \mathcal{C}$;
        $\mathrm{pred}_t = \mathrm{argmin}_c s_{tc}$;
    end
    $r_c = (\frac{\sum_{t:\mathrm{pred}_t=c} \mathrm{enc}(t)}{\mathrm{count}(\{t:\mathrm{pred}_t=c\})} + \mathrm{enc}(c))/2\ \forall c \in \mathcal{C}$;
    objective $= \sum_t s_{t\mathrm{pred}_t}$;
end
Result: $s_{tc}$, objective, $\mathrm{pred}_t$, and $r_c$ of all
        iterations
```

the information from both the category and the text. This vector can pass through a linear layer and produce a score for this particular document-category pair. Figure 1 demonstrates the architectures of typical dual and single encoder models for text classification.

3 Unsupervised Label Refinement (ULR)

In this section, we introduce unsupervised label refinement (ULR). ULR uses the components of a dataless classifier and refines representations of labels with a modified k-means clustering algorithm. While dataless text classifiers are designed to handle an unbounded set of categories, they are used and evaluated on a particular set of documents with a set of labels. The idea of our approach is to leverage the assumption that the documents in a text classification dataset are separable according to the accompanying set of labels. That is, given a strong document encoder, the documents should be separable by label in the encoded space. This assumption is similarly made when performing clustering for unsupervised document classification (Liang and Klein, 2009).

We use the set of unlabeled input texts to refine the predictions of our dataless classifiers via clustering. To better inform the algorithm, we initialize the clusters by using our dataless classifiers run on the provided label set for each task. The algorithm takes on different forms for the dual and single encoder models. Details are provided below.

3.1 ULR for Dual Encoder Architectures

In the setting of a dual encoder model, we propose to perform k-means clustering among text representations, i.e., of vectors produced by the text encoder. The assumption is that texts falling under the same category will be close in the semantic space. We want to adapt our dataless classifier's predictions based on the natural clustering structure in the encodings of the texts.

We show the ULR algorithm for dual encoder architectures in Algorithm 1. We use one encoder enc to encode texts and categories. We link centroids to categories and initialize the centroids using the encodings of the corresponding categories. The algorithm converges when no data point (text representation) updates its cluster assignment, i.e., the centroids stop updating. In our experiments, we run a maximum of 100 iterations. We perform model selection ("early stopping") based on the minimum value of $\mathrm{objective}$ among iterations.

Our k-means algorithm differs from standard k-means as our updated centroids are interpolated with the initial category embeddings. In standard k-means, the centroids are typically initialized randomly. In our case, since we link centroids to categories and use our encoder to provide initial centroids, we want to leverage the information in our category embeddings across clustering iterations. Therefore, we average the "new centroid" with the original category vector, which serves as a kind of regularization. While this update is heuristically motivated, we can reverse-engineer the clustering objective it approximately optimizes (with

Euclidean distance as score):

$$\min_{\{r_c\}_{c\in\mathcal{C}}} \min_{\{a_t\}_{t\in\mathcal{T}}} \sum_{t\in\mathcal{T}} ||\mathrm{enc}(t) - r_{a_t}||^2 + \sum_{c\in\mathcal{C}} |c|\, ||\mathrm{enc}(c) - r_c||^2 \quad (1)$$

where $a_t \in \mathcal{C}$ is the category assignment of document $t \in \mathcal{T}$ and $|c|$ is the size of cluster/category c. See Appendix A for details. In preliminary experiments we found this modification to stabilize performance so we use it in our experiments reported below.

3.2 ULR for Single Encoder Architectures

In our single encoder architecture, a score is produced for each document-category pair indicating its relatedness. For each document, after exponentiating and normalizing the score over all categories, we obtain a distribution indicating the probability of the document belonging to each category. For text t and category c, we have a probability p_{tc}, where $\sum_c p_{tc} = 1$.

A straightforward way of classifying each document is to pick the category of which it has the highest probability score, i.e., $\mathrm{argmax}_c\, p_{tc}$. Another way of interpreting this classification rule is to define $|\mathcal{C}|$ centroid vectors, each being an identity distribution $\text{one–hot}(c)$,[3] and pick the category having the minimum Jensen-Shannon Divergence (Lin, 1991) with the document probability vector, i.e., $\mathrm{argmin}_c\, \mathrm{JS}(p_t, \text{one–hot}(c))$, where JS is the Jensen-Shannon Divergence between two distributions. Clustering probability distributions based on KL divergence is a well-known special case of clustering with Bregman divergences (Banerjee et al., 2005).[4] While JS does not admit such a well-defined objective to our knowledge, we find that it works well in practice.

It is natural to represent each category as a distribution by a one-hot vector. However, in a real text classification problem, the semantics of a category is affected by how the annotators view it. For instance, a news document could relate to both "business" and "science & technology", though it will only have a single annotated category in the downstream task dataset. With this intuition, we propose to represent each category by a soft distribution over all the categories, but not necessarily a one-hot vector.

Algorithm 2: ULR for single encoder architectures

```
Data: documents $\mathcal{T}$, categories $\mathcal{C}$, scoring
      function score, function to compute
      JS divergence JS
initialize the centroids as
  $r_c = \text{one–hot}(c)\ \forall c \in C$;
compute each document's probability
  distribution over categories as
  $[p_t]_c \propto \exp(\mathrm{score}(t, c))\ \forall t \in \mathcal{T}, c \in \mathcal{C}$;
while not converged do
    for $t \in \mathcal{T}$ do
        $s_{tc} = \mathrm{JS}(p_t, r_c)\ \forall c \in \mathcal{C}$;
        $\mathrm{pred}_t = \mathrm{argmin}_c\, s_{tc}$;
    end
    $r_c = \frac{\sum_{t:\mathrm{pred}_t=c} p_t}{\mathrm{count}(\{t:\mathrm{pred}_t=c\})}\ \forall c \in \mathcal{C}$;
end
Result: $\mathrm{pred}_t$ of the last iteration
```

Algorithm 2 describes our k-means clustering approach applied on the single encoder model. In this algorithm our predicted categories will be the clustering assignments of the last iteration. Unlike with the dual encoder model, we do not do early stopping. We also do not use interpolated centroids as one-hot vectors may not necessarily be good category embeddings in this setting.

4 Experimental Setup

We now introduce the datasets we used for evaluation and the dual encoder and single encoder dataless classifiers we used to run ULR experiments.

4.1 Evaluation

We use four text classification datasets spanning different domains for evaluation. They are: AG News[5] (AG), which uses 4 classes and covers the newswire domain; DBpedia (DBP; Lehmann et al., 2015), which has 14 classes and is from the domain of encyclopedias; Yahoo (Zhang et al., 2015), which has 10 classes and addresses categorizing

---

[3] Here we abuse the notation of $\text{one–hot}(c)$ to represent a one-hot vector that has a "1" at the index of category $c \in \mathcal{C}$.

[4] Specifically, if we replace $\mathrm{JS}(p_t, r_c)$ with $D_{\mathrm{KL}}(p_t||r_c)$ in Algorithm 2, it is an alternating minimization of the following clustering objective:

$$\min_{\{r_c\}_{c\in\mathcal{C}}} \min_{\{a_t\}_{t\in\mathcal{T}}} \sum_{t\in\mathcal{T}} D_{\mathrm{KL}}(p_t||r_{a_t})$$

[5] `https://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

questions in online question fora; and 20 newsgroups (20NG; Lang, 1995), with 20 classes which are types of newsgroups.

We do not use any data or labels from the training set in our main experiments, but only rely on label descriptions. We use the official label names from these datasets, and only expand them if the original label name is provided as abbreviations such as "sci_tech". The exact label names we used are in the appendix.

4.2 Dataless Classifiers

We experiment with multiple dataless text classifiers that vary in terms of their complexity. Our simplest classifier uses an encoder that averages pretrained GloVe (Pennington et al., 2014) word embeddings. We also fine-tune a RoBERTa model (Liu et al., 2019) in both single and dual encoder settings, using RoBERTa-base (110M parameters). We do not run experiments with traditional dataless classifiers such as ESA (Chang et al., 2008; Gabrilovich and Markovitch, 2007) as ESA vectors are of extremely high dimension, making it computationally difficult to apply ULR. GloVe and RoBERTa produce lower dimensional vectors that are more computationally amenable to refinement. Also, we experimented with ESA and found that it does not perform as well as our RoBERTa based models (where both are tested without ULR).[6] Next we describe the details of the three dataless classifiers that we use in our experiments.

GloVe dual encoder. We use GloVe (Pennington et al., 2014) in the dual encoder setting, simply averaging word vectors to represent both the categories and the documents. We use the 300 dimensional GloVe vectors[7] trained on Common Crawl. We experiment with two distance functions when using GloVe: cosine and L2.

RoBERTa dual encoder. The category c and text t are fed separately to RoBERTa using the formatting "[CLS] c [SEP]" and "[CLS] t [SEP]". We use the average of the final-layer hidden states produced by RoBERTa as category and text vectors. For the scoring function, which computes a scalar from the category and text vectors, we consider dot product[8] and L2 distance.

RoBERTa single encoder. The category c and text t are combined in the form "[CLS] c [SEP] t [SEP]", tokenized, and encoded using RoBERTa. We truncate t to ensure the category-document pair is within 128 tokens. The vector representation of the "[CLS]" token (after a linear transformation and non-linear activation) is then passed to a linear layer to produce a score.

4.3 Fine-tuning RoBERTa Models

We use the NatCat dataset (Chu et al., 2020) to fine-tune the RoBERTa models. NatCat comprises document-category pairs from three resources: Wikipedia, Stack Exchange, and Reddit. Each NatCat document comes with positive and negative categories. A positive category describes the document, and a negative category is randomly sampled and is irrelevant to the document. The RoBERTa models are fine-tuned as binary classifiers to indicate whether a category is positive for a document. We use the Hugging Face framework (Wolf et al., 2019) to fine-tune all RoBERTa models, using 300k instances from NatCat.[9]

Fine-tuning RoBERTa dual encoder. While many other combinations of score and loss function could be considered, we report results with two particular combinations: dot product paired with binary cross entropy and L2 distance paired with a contrastive hinge loss. When dot product is used, RoBERTa is fine-tuned to minimize binary cross entropy between $\text{score}(\text{enc}(c), \text{enc}(t))$ and a binary label $y \in \{0, 1\}$.

When using L2 distance, we fine-tune RoBERTa to minimize a contrastive hinge loss:

$$\text{loss} = \max(x_p + \gamma - x_n, 0)$$

where $x_p = \text{score}(\text{enc}(c_p(t)), \text{enc}(t))$, $x_n = \text{score}(\text{enc}(c_n(t)), \text{enc}(t))$, score is the squared L2 distance, $c_p(t)$ is a positive category for text t, $c_n(t)$ is a negative category, and γ is a parameter indicating the margin. This loss aims to make negative category-text pairs have higher squared L2 distance than negative pairs by the margin.

Fine-tuning RoBERTa single encoder. When used as a single encoder, RoBERTa is fine-tuned

---

[6] As a comparison to the results of the RoBERTa dual encoder in Table 2, ESA accuracies (%) are 71.2 for AG, 62.5 for DBP, 29.7 for Yahoo, and 25.1 for 20NG.

[7] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

[8] Practically we find the model is easier to train with dot product compared to cosine similarity.

[9] The finetuning hyperparameters are in the appendix.

		AG	DBP	Yahoo	20NG	avg
GloVe						
cosine	baseline	66.1	43.5	**29.4**	29.8	42.2
	+ ULR	**78.3**	**46.4**	27.9	**30.8**	**45.9**
L2	baseline	40.5	15.7	14.5	19.5	22.6
	+ ULR	**58.7**	**35.7**	**23.3**	**25.9**	**35.9**
RoBERTa						
cosine	baseline	**74.0**	84.6	52.3	36.4	61.8
	+ ULR	73.7	**93.3**	**54.3**	**40.3**	**65.4**
L2	baseline	69.9	78.8	55.7	**37.8**	60.6
	+ ULR	**70.7**	**90.5**	**61.3**	37.4	**65.0**

Table 2: Accuracies (%) when applying ULR to the GloVe and RoBERTa dual encoder architecture ("baseline"), for two different choices of distance function (cosine and L2).

	AG	DBP	Yahoo	20NG	avg
baseline	72.6	81.8	59.3	36.0	62.4
+ ULR	**75.1**	**88.6**	**60.0**	**36.5**	**65.1**

Table 3: Accuracies (%) when applying ULR to the RoBERTa single encoder architecture ("baseline").

		AG	DBP	Yahoo
cosine	ensemble	79.1	84.6	**54.5**
	+ ULR	**81.1**	**93.3**	54.4
L2	ensemble	72.4	79.0	56.5
	+ ULR	**73.4**	**90.7**	**61.7**

Table 4: Accuracies (%) of ensemble predictions using 10 choices of label names using the RoBERTa dual encoder architecture ("ensemble"), and results when combining it with ULR.

to minimize binary cross entropy between the predicted score $\mathrm{score}(\mathrm{enc}(c, t))$ and a binary label y, where score is a linear function that transforms the vector into a scalar score.

5 Experimental Results of ULR

Dual encoder models. With the dual encoder models, we used two sets of distance measures. The first distance is the cosine distance of two vectors. In this case, we always normalize the vector representations before applying ULR. The second distance measure is the L2 distance.

Table 2 shows results for the GloVe and RoBERTa dual encoder model with two distance functions. With GloVe, except for the single case of cosine distance with Yahoo, all accuracies improve, with some improving by large amounts (up to 20% absolute). With the RoBERTa dual encoder model, the choice of distance function matches the scoring function we used when fine-tuning RoBERTa, i.e., cosine distance is used with dot product scoring and L2 distance is used with the contrastive hinge loss. ULR improves accuracies by 3.6% to 4.4% on average, and the improvements are consistent across distance functions and datasets, except for the cases of 20 NEWSGROUPS with L2 distance and AG with cosine distance, which show slight degradations.

Single encoder model. Table 3 summarizes the results of applying ULR to the RoBERTa single encoder architecture. ULR improves performance across all four datasets, ranging from 0.5% for 20NG up to 6.8% on DBP. Additional experiments in the appendix show the impact of early stopping and interpolation using this architecture.

Label ensembles. Finding the best choice of label names for dataless classifiers is difficult without labeled data. Therefore, it is easier to supply multiple choices of label names for a given task.

We manually pick 10 different sets of label names for each dataset,[10] generate category and text representations with our RoBERTa dual encoder model, and perform ULR. The exact choices of label names are in the appendix. The predicted scores for the 10 sets are summed up as the final ensemble predictions. Table 4 shows results. Compared to Table 2, all accuracies on all tasks improve. Even with a stronger starting point from ensembling, ULR still yields consistent improvements in accuracy, with the single exception of Yahoo and cosine distance.

6 Robustness Experiments

Different choices of label names. Dataless text classification tasks are known to suffer from high variance due to label descriptions. Performance can vary dramatically across different choices of category names, as shown in Table 1.

One advantage of ULR is that it is robust to label noise. That is, even given a poor choice of label names, ULR can help the classifier to partially recover some of the accuracy, as shown in the lower

---

[10] We exclude 20NG mainly because it is difficult to come up with synonyms for its labels for this ensemble experiment and the robustness analysis in Section 6, since some labels in 20NG include proper nouns like "windows", "ibm", "mac", etc.

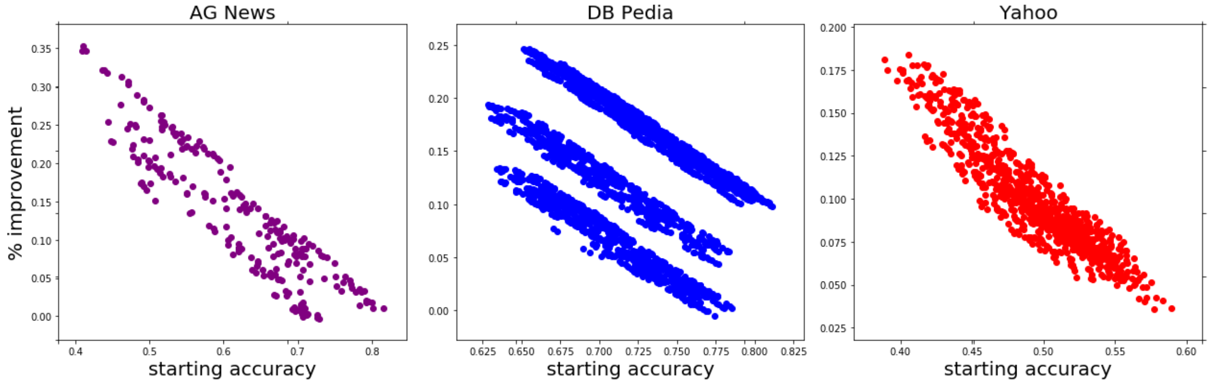

Figure 2: Accuracy (%) improvement when applying ULR compared to the RoBERTa dual encoder model (with Euclidean distance). The horizontal axis is the initial accuracy and the vertical axis is the absolute accuracy improvement after ULR.

		AG News	DBP	Yahoo
cos.	baseline	69.6 (8.7)	79.4 (3.2)	43.8 (3.9)
	+ ULR	**77.5** (5.5)	**91.6** (3.2)	**45.3** (3.9)
	# imp.	$\frac{213}{240}$ = 89%	$\frac{3867}{3867}$ = 100%	$\frac{732}{1015}$ = 72%
L2	baseline	62.2 (9.6)	71.7 (3.8)	49.5 (3.8)
	+ ULR	**75.2** (4.1)	**85.2** (5.5)	**59.3** (1.5)
	# imp.	$\frac{237}{240}$ = 99%	$\frac{2960}{2963}$ = 100%	$\frac{947}{947}$ = 100%

Table 5: Robustness analysis when varying choices of label names. The "baseline" and "+ ULR" rows show average accuracies (%) with standard deviations in parentheses of the RoBERTa dual encoder architecture among all category naming choices. The "# imp." rows show the numbers and percentages of cases where the performance improves after ULR.

portion of Table 1. We now describe similar experiments on a larger scale. In particular, we test several hundred sets of label names for each of AG, DBP, and Yahoo. For each category, we randomly assign different label names to it, all having similar meaning. The exact combinations of label names are in the appendix.

We then perform dataless text classification and ULR with our RoBERTa dual encoder model, either under cosine or L2 distance. Table 5 shows the results of the average performance gains before and after ULR. We also report the number and percentage of cases in which ULR improves accuracy. ULR improves the performance on average for all three tasks, and improves individual accuracies in the vast majority of cases. These results show that ULR is not only effective at improving the accuracy of dataless text classifiers across a wide range of label name sets, but also can help to mitigate harmful effects due to suboptimal label names.

We next study the relationship between the RoBERTa dual encoder model's initial accuracies and its accuracy gains with ULR. Figure 2 plots accuracy improvements vs. initial accuracies for the three datasets. We find that ULR gives larger gains when the initial accuracies are lower.

We also computed oracle accuracies by choosing the combination of label names that maximize accuracy (without ULR). With cosine distance, they are 81.9% for AG, 87.5% for DBP, and 53.3% for Yahoo. With L2 distance, they are 81.5% for AG, 81.1% for DBP, and 58.9% for Yahoo. Using ULR with the RoBERTa dual encoder provides better performance on DBP and Yahoo, and competitive results on AG, as shown in Tables 2 and 4.

Clustering with random initialization. We now investigate the impact of our initialization in ULR. We consider a variation in which we randomly initialize centroids. Since we can no longer link clusters and categories, this task becomes unsupervised text classification. A standard k-means algorithm is applied to update the centroids. Unlike Algorithm 1, in this experiment we do not interpolate the updated centroids with the initial centroids, as the initial centroids are random. The accuracy is calculated based on the oracle one-to-one mapping between final centroids and categories. This is often referred to as "one-to-one accuracy".

Table 6 presents results on AG with this experiment. We performed 240 trials with different random initialization of the centroids, and unsurprisingly, accuracy is improved in all cases. With L2 distance, ULR with random initialization even outperforms centroids initialized as category embeddings, which shows that unsupervised clustering is

		cosine	L2
AG News	baseline	37.3	39.9
	+ ULR	**61.4**	**75.8**
	# improved	$\frac{239}{240}$ = 99.6%	$\frac{240}{240}$ = 100%

Table 6: Average 1-to-1 accuracy (%) of ULR with random initialization of centroids. "baseline" and "ULR" are average accuracy (%) of the RoBERTa dual encoder architecture among all random initialized centroids. "# imp." are the numbers and percentages of cases where the performance improves after ULR.

	AG	DBP	Yahoo	20NG	avg
baseline	74.0	84.6	52.3	36.4	61.8
+ 30 labeled ins.	81.5	97.1	66.0	58.1	75.7
+ ULR	**85.2**	**97.4**	**66.7**	**58.2**	**76.9**

Table 7: Accuracies (%) when adding 30 labeled instances for each category with the RoBERTa dual encoder model ("baseline").

powerful in text classification with good text representations. However, since we do not have the one-to-one mapping between category and final centroids in the purely unsupervised setting, the results in Table 6 are not directly comparable to those in earlier tables.

7 ULR with Few Shot Learning

In this section, we apply ULR in the few shot learning setting. In particular, we draw 30 labeled instances for each category from the training split in the original datasets. We then further fine-tune our RoBERTa dual encoder model with the labeled instances. We adopt the hyperparameters of fine-tuning text classifiers from Wolf et al. (2019) and fine-tune for 3 epochs on these 30 labeled instances for each category. Then we apply ULR on the unlabeled test set in addition to the 30 labeled instances from each category.

Unlike Algorithm 1, these 30 labeled instances from each category are fixed to be assigned to their corresponding clusters. We compute the centroids of these clusters from 30 labeled instances by $r^l_c = \frac{\sum_{t \in \mathcal{L}_c} \text{enc}(t)}{|\mathcal{L}_c|}$ $\forall c \in \mathcal{C}$ where documents $\mathcal{L}_c$ are labeled with c $\forall c \in \mathcal{C}$.

At every iteration, the update rule of the centroids as in Algorithm 1 will include the text vector representations of these 30 labeled instances, i.e., $r_c = (\frac{\sum_{t:\text{pred}_t = c} \text{enc}(t)}{\text{count}(\{t:\text{pred}_t = c\})} + r^l_c + 2 \times \text{enc}(c))/4$.

Table 7 summarizes the results of combining 30 labeled instances for each category and ULR. Unsurprisingly, adding labeled instances improves accuracy by a large margin. Even so, ULR yields an additional improvement of 3.7% for AG, and also improves on the other datasets.

8 Analysis

A key challenge of dataless text classifiers is semantic drift between the training data and downstream tasks. As ULR is based on k-means, the centroids move towards the embeddings of the majority of text instances belonging to that category. Hence after ULR the locations of the centroids combine the information in the initial dataless text classifiers and the document sets for the downstream tasks.

Using AG, Table 8 shows the 5 text instances closest to the centroids for "business" and "sci&tech" before and after ULR. These categories are hard to distinguish as technology companies often appear in both business and technology news. However, after ULR, we see that texts containing technology companies ("LeapFrog", "ScanSoft") become less central for the "business" category, replaced by texts describing transactions and earnings. Also, a text containing a company name ("NEC") has become less central for the "sci&tech" category, replaced in the top 5 by a similar text that does not emphasize a company name.

In certain text classification settings, we may be able to abstain from classifying instances we are less confident about. This notion can be instantiated by checking the texts closest to the centroids corresponding to the labels, which we can view intuitively as the ones the model is most confident about. Table 9 shows a confidence-based evaluation, evaluating the k texts that are closest to each centroid and showing that ULR consistently improves precision at k.

9 Related Work

Earlier we discussed prior work on dataless and zero-shot text classification. We briefly introduce more related work in this section. Song et al. (2015) provide a text label refinement algorithm to adjust the label set with noisy and missing labels. There is also a wealth of prior work in semi-supervised text classification: using unlabeled text to improve classification performance (Nigam et al., 2000; Mukherjee and Awadallah, 2020; Xie et al., 2019). These methods typically learn generally useful text representations from a large corpus of unlabeled text and use them for a specific target task with

category	baseline	+ ULR
business	Hicks Muse Pays for ConAgra's Swift Stake ... EDS Is Charter Member of Siebel BPO Alliance ... GM, Daimler Go Green ... *LeapFrog*'s Greener Pastures (The Motley Fool) ... *ScanSoft* to acquire 3 software firms ...	CA to buy Netegrity for 430 m ... Sara Lee 1st-Quarter Net Rises on Fee (Reuters) ... Sears and Kmart Agree to Merge in 11 Billion Deal ... LCC Int Posts 3Q Profit, Shares Tumble, ... Before-the-Bell: GenCorp Falls 5.6 Pct. (Reuters) ...
sci&tech	Robot eats flies to make power ... Particle lab celebrates 50th birthday ... Time on a Chip: The Incredible Shrinking Atomic ... *NEC* Unveils World Fastest Vector Supercomputer .. Breakthrough on hydrogen fuel ...	Particle lab celebrates 50th birthday ... Breakthrough on hydrogen fuel ... Time on a Chip: The Incredible Shrinking Atomic ... Record Breaking Supercomputer Performance ... Robot eats flies to make power ...

Table 8: Top-5 scoring texts belonging to categories "business" and "sci&tech" in AGNews.

		P@10	P@50	P@100	P@500
AG	baseline	85.0	87.5	89.3	85.4
	+ ULR	85.0	**89.5**	**91.5**	**89.9**
DBP	baseline	83.6	84.1	84.1	84.6
	+ ULR	**92.1**	**92.1**	**92.6**	**91.5**
Yahoo	baseline	79.0	84.8	83.6	83.3
	+ ULR	**91.0**	**90.2**	**89.4**	**87.2**
20NG	baseline	64.5	62.0	57.6	35.7
	+ ULR	**71.5**	**65.0**	**62.0**	**38.8**

Table 9: Precision (%) at k (P@k) with dual encoder ROBERTA and cosine scoring.

limited supervision (Howard and Ruder, 2018; Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; Peters et al., 2018). Metric learning (Wohlwend et al., 2019) is related to our work, but they focus on few-shot learning and we work on improving unsupervised text classifiers. Related contemporaneous work has proposed methods to generate more relevant label names from a given set (Meng et al., 2020; Schick et al., 2020). ULR is orthogonal to such methods of choosing label names, as these label names can be set as extra initial centroids. Finally, label descriptions have also been exploited in supervised learning to improve text classifiers (Chai et al., 2020; Wang et al., 2019; Sun et al., 2019).

10 Conclusion

Our proposed ULR algorithms provide a simple but effective framework to improve the performance of dataless text classifiers. Extensive experiments show its flexibility and robustness, offering promise for making dataless text classification more useful for practitioners.

Acknowledgments

We would like to thank the anonymous reviewers for helpful comments. This research was supported in part by a Bloomberg data science research grant to KS and KG.

References

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. 2005. Clustering with Bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749.

Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. *ICML*.

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John Carroll. 2015. Dataless text classification with descriptive LDA. In *AAAI*.

Zewei Chu, Karl Stratos, and Kevin Gimpel. 2020. Natcat: Weakly supervised text classification with naturally annotated datasets. *arXiv preprint arXiv:2009.14335*.

Yann N Dauphin, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2013. Zero-shot learning for semantic utterance classification. *arXiv preprint arXiv:1401.0509*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *NAACL*.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, and Katia Sycara. 2016. Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In *COLING*.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *NAACL-HLT*.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *AAAI*.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text classification using label names only: A language model self-training approach. In *EMNLP*.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for text classification with few labels. *arXiv preprint arXiv:2006.15315*.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695*.

Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Kamal Nigam, Andrew Mccallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. In *Machine Learning*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *EMNLP*.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *COLING*.

Lei Shu, Hu Xu, and Bing Liu. 2017. Doc: Deep open classification of text documents. In *EMNLP*.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*.

Yangqiu Song, Chenguang Wang, Ming Zhang, Hailong Sun, and Qiang Yang. 2015. Spectral label refinement for noisy and missing text labels. In *AAAI*.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*.

Jingjing Wang, Changlong Sun, Shoushan Li, Xiaozhong Liu, Luo Si, Min Zhang, and Guodong Zhou. 2019. Aspect sentiment classification towards question-answering with reinforced bidirectional attention network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3548–3557.

Pu Wang, Charlotta Domeniconi, and Jian Hu. 2009. Towards a universal text classifier: Transfer learning using encyclopedic knowledge. In *Data Mining Workshops*.

Jeremy Wohlwend, Ethan R. Elenberg, Sam Altschul, Shawn Henry, and Tao Lei. 2019. Metric learning for dynamic text classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 143–152, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3905–3914.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. In *arXiv*.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *NAACL*.

Xiang Zhang, Junbo Zhao, and Yann Lecun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

A Clustering Objective

For convenience, we first rewrite Eq. (1) below:

$$\min_{\{r_c\}_{c\in\mathcal{C}}} \min_{\{a_t\}_{t\in\mathcal{T}}} \sum_{t\in\mathcal{T}} ||\text{enc}(t) - r_{a_t}||^2 + \sum_{c\in\mathcal{C}} |c|\, ||\text{enc}(c) - r_c||^2 \quad (2)$$

We also rewrite Algorithm 1 here as Algorithm 3:

Algorithm 3: ULR for dual encoder architectures

```
Data: documents $\mathcal{T}$, categories $\mathcal{C}$, encoder
      enc, scoring function score
initialize the centroids $r_c = \text{enc}(c)\,\forall c \in \mathcal{C}$;
while not converged do
    for $t \in \mathcal{T}$ do
        $s_{tc} = \text{score}(\text{enc}(t), r_c)\,\forall c \in \mathcal{C}$;
        $\text{pred}_t = \text{argmin}_c s_{tc}$;
    end
    $r_c = (\frac{\sum_{t:\text{pred}_t=c}\text{enc}(t)}{\text{count}(\{t:\text{pred}_t=c\})} + \text{enc}(c))/2\,\forall c \in \mathcal{C}$;
    $\text{objective} = \sum_t s_{t\text{pred}_t}$;
end
Result: $s_{tc}$, objective, $\text{pred}_t$, and $r_c$ of all
        iterations
```

The objective (2) as a function of a single centroid corresponding to category $c \in \mathcal{C}$ can be written as

$$\min_{v\in\mathbb{R}^d} \sum_{t\in c} ||\text{enc}(t) - v||^2 + |c|\, ||\text{enc}(c) - v||^2$$

where $c \in \mathcal{C}$ is constant given assignments a_t. It is easy to verify that the unique solution is given by

$$r_c = \frac{\frac{1}{c}\sum_{t\in c}\text{enc}(t) + \text{enc}(c)}{2}$$

which is the update in Algorithm 3. Now consider (2) as a function of category assignments

$$\min_{\{a_t\}_{t\in\mathcal{T}}} \sum_{t\in\mathcal{T}} ||\text{enc}(t) - r_{a_t}||^2 + \sum_{c\in\mathcal{C}} |c|\, ||\text{enc}(c) - r_c||^2$$

where centroids r_c are held constant. Unfortunately this problem remains intractable because the regularization term is weighted by cluster sizes. We can approximate it by ignoring the regularization term: in this case the minimizer is given by $a_t = \arg\min_{c\in\mathcal{C}} ||\text{enc}(t) - r_c||$ for each $t \in \mathcal{T}$ as usual. Thus Algorithm 3 can be viewed as optimizing (2) by alternating minimization, with the simplifying (incorrect) assumption that cluster sizes are not significantly affected by category assignments.

B Experimental Setup

We list the hyperparameters we used to fine-tune our RoBERTa based dataless classification models in this section, so readers can reproduce our experimental results. The code with the scripts are also included in the supplementary material. We choose RoBERTa-base. In all fine-tuning tasks, we set the batch size to be 32. The max sequence length for RoBERTa is 128. The peak learning rate is 0.00002. We use a linear scheduler with warmup steps to be 10% of the total fine-tuning steps. The random seeds of all experiments are set to be 1.

We fine-tune RoBERTa models on random choices of single GPUs, including NVIDIA TITAN X, 1080Ti, or 2080 Ti. Most of the fine-tuning tasks can be finished within 8 hours.

C Label Names in the Evaluation Tasks

In the main experiments (Section 5), we use the following label descriptions (separated by ";") for the downstream tasks.

- AG: world; sports; business; science technology
- DBP: company; educational institution; artist; athlete; politician; transportation; building; nature; village; animal; plant; album; film; written work
- Yahoo: society culture; science mathematics; health; education reference; computers internet; sports; business finance; entertainment music; family relationships; politics government
- 20 Newsgroups: atheist christian atheism god islamic; graphics image gif animation tiff; windows dos microsoft ms driver drivers card printer; bus pc motherboard bios board computer dos; mac apple powerbook; window motif xterm sun windows; sale offer shipping forsale sell price brand obo; car ford auto toyota honda nissan bmw; bike motorcycle yamaha; baseball ball hitter; hockey wings espn; encryption key crypto algorithm security; circuit electronics radio signal battery; doctor medical disease medicine patient; space orbit moon earth sky solar; christian god christ church bible jesus; gun fbi guns weapon compound; israel arab jews jewish muslim; gay homosexual sexual; christian morality jesus god religion horus

Label ensembles. In the experiment of ensembling labels, we use the following 10 sets of label name choices.

For AG:

- world; sports; business; science technology
- international; sports; business; science technology
- world; sports; business; science and technology
- international; sports; business; science and technology
- world; sports; business and finance; science technology
- international; sports; business and finance; science technology
- world; sports; business and finance; science and technology
- international; sports; business and finance; science and technology
- world politics; sports; business; science technology
- world politics; sports; business; science technology

For DBP:

- company; educational institution; artist; athlete; politician; transportation; building; nature; village; animal; plant; album; film; written work
- company; school; artist; athlete; politician; transportation; building; nature; village; animal; plant; album; film; written work
- company; educational institution; artist; athlete; politician; transportation; architecture; nature; village; animal; plant; album; movie; written work
- company; educational institution; artist; athlete; politician; transportation; building; nature; village; animal; plant; album; film; novel
- company; educational institution; artist; athlete; politician; transportation; building; nature; village; animal; plant; album; film; article
- company; educational institution; artist; athlete; politician; transportation; building; nature; village; animal; plant; collection; movie; written work
- company; school; artist; athlete; politician; transportation; architecture; nature; village; animal; plant; album; movie; written work
- company; educational institution; artist; athlete; politician; transportation; architecture; nature; village; animal; plant; album; movie; novel
- company; school; artist; athlete; politician; transportation; architecture; nature; village; animal; plant; album; film; novel

- company; educational institution; artist; athlete; official; transportation; building; nature; village; animal; plant; album; movie; article

For Yahoo:

- society culture; science mathematics; health; education reference; computers internet; sports; business finance; entertainment music; family relationships; politics government
- society culture; scientific discipline; health; education reference; computers internet; sports; business finance; entertainment music; family relationships; politics government
- society culture; science mathematics; health; learning teaching resource; computers internet; sports; business finance; entertainment music; family relationships; politics government
- society culture; science mathematics; health; education reference; computers internet; sports; business finance; entertainment music; love home; politics government
- society culture; science mathematics; health; education reference; information technology; sports; business finance; entertainment music; family relationships; politics government
- society culture; scientific discipline; health; education reference; information technology; sports; business finance; entertainment music; family relationships; politics government
- society culture; science mathematics; health; learning teaching resource; information technology; sports; business finance; entertainment music; family relationships; politics government
- society culture; science mathematics; health; education reference; computers internet; sports; commerce; entertainment music; family relationships; politics government
- society culture; science mathematics; health; education reference; information technology; sports; commerce; entertainment music; family relationships; politics government
- society culture; science mathematics; health; learning teaching resource; computers internet; sports; commerce; entertainment music; family relationships; politics government

D Robustness to Label Noise

In the robustness experiments (Section 6), we manually picked different choices of label names for each category of the downstream tasks. We list the choices of label names for each category below, separated by ";".

AG:

- world: world; world politics; world news; international; international news
- sports: sports; health; health and sports
- business: business; commerce; finance; business and finance
- science technology: science; technology; science and technology; science technology

DBP:

- company: company; corporation
- educational institution: educational institution; school
- artist: artist; creator
- athlete: athlete; sportsman
- politician: politician; official
- transportation: transportation
- building: building; architecture
- nature: nature
- village: village; suburb
- animal: animal; living thing
- plant: plant
- album; collection
- file: film; movie
- written work: written work; writing; novel; article

Yahoo:

- society culture: society culture; community
- science mathematics: science mathematics; scientific discipline
- health: health; fitness
- education reference: education reference; learning teaching resource
- computers internet: computers internet; information technology
- sports: sports; athletics
- business finance: business finance; commerce
- entertainment music: entertainment music; fun songs
- family relationships: family relationships; love home
- politics government: politics government; policy regime regulation

Different combinations of label name choices are used to generate category embeddings and perform ULR.

	AG	DBP	Yahoo	20NG	avg
baseline	72.6	64.7	57.1	28.7	55.8
+ULR	**75.6**	**75.1**	**59.9**	**35.9**	**61.6**

Table 10: Improvements of ULR with the ROBERTA dual encoder model on unbalanced datasets.

	AG	DBP	YAHOO	20NG	avg
baseline	72.6	81.8	59.3	36.0	62.4
+ ULR					
	75.1	**88.6**	**60.0**	36.5	**65.0**
+ early	72.6	88.6	59.3	36.5	64.3
+ ULR + interpolate					
	74.0	87.2	59.6	**36.9**	64.4
+ early	74.0	87.2	59.6	**36.9**	64.4

Table 11: Accuracies (%) when applying ULR to the ROBERTA single encoder architecture ("baseline"). "+interpolate" are the results when we update centroids with interpolation. "+ early" means early stopping.

E Augmented Categories

In the experiment of augmenting categories with ULR (Section 8), we use the following augmented categories (separated by ";"): math; gis; physics; codereview; stats; unix; english; tex; gaming; apple; scifi; drupal; ell; meta; electronics; travel; rpg; dba; magento; webapps; diy; wordpress; android; security; chemistry; webmasters; blender; softwareengineering; gamedev; academia.

F Unbalanced Dataset

We manually create unbalanced datasets, where each category only contains 1-100% of the original instances. For a dataset with n categories, each instance belonging to category i has a probability of $\frac{i}{n}, i \in \{0, 1, \cdots, n-1\}$ of being dropped, where the orders of categories are random. Table 10 shows that ULR improves performance with unbalanced datasets.

G Single Encoder

Table 11 summarizes the results of applying ULR on the single encoder model with two variations:

1. early: early stopping on the average Jenson-Shannon Divergence score of all probability distribution of text over all categories to their corresponding nearest one-hot centers;
2. interpolate: when updating centroids in Algorithm 2, we take the average of the new centroids with the centroids from the previous iteration.

Overall, these two variations do not provide better results in the setting of single encoder models.

Prompting Contrastive Explanations for Commonsense Reasoning Tasks

Bhargavi Paranjape[†*] **Julian Michael**[†] **Marjan Ghazvininejad**[*]
Hannaneh Hajishirzi[†ϵ] **Luke Zettlemoyer**[†*]
[†]Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
[ϵ]Allen Institute of Artificial Intelligence, Seattle
[*]Facebook AI
`{bparan,julianjm,lsz,hannaneh}@cs.washington.edu`

Abstract

Many commonsense reasoning NLP tasks involve choosing between one or more possible answers to a question or prompt based on knowledge that is often implicit. Large pretrained language models (PLMs) can achieve near-human performance on such tasks, while providing little human-interpretable evidence of the underlying reasoning they use. In this work, we show how to use these same models to generate such evidence: inspired by the contrastive nature of human explanations, we use PLMs to complete explanation prompts which contrast alternatives according to the key attribute(s) required to justify the correct answer (for example, *peanuts are usually salty while raisins are sweet*). Conditioning model decisions on these explanations improves performance on two commonsense reasoning benchmarks, as compared to previous non-contrastive alternatives. These explanations are also judged by humans to be more relevant for solving the task, and facilitate a novel method to evaluate explanation faithfulness.

1 Introduction

Pretrained Language Models (PLMs) (Raffel et al., 2020; Lewis et al., 2020; Radford et al., 2019; Brown et al., 2020) have been shown to encode substantial amounts of knowledge in their parameters (Petroni et al., 2019; Talmor et al., 2020; Roberts et al., 2020) and have achieved impressive performance on commonsense reasoning (CSR) tasks without the use of external knowledge (Trinh and Le, 2018; Yang et al., 2020). However, these models provide little human-interpretable evidence of the intermediate commonsense knowledge or reasoning they use, and have been observed to overly rely on superficial dataset artifacts (Poliak et al., 2018; Geva et al., 2019). To overcome this limitation, recent work has shown that PLMs can explain themselves by *generating* free-form natural language explanations of their reasoning patterns (Rajani et al., 2019a; Camburu et al., 2018; Narang et al., 2020). However, the space of possible free-form explanations is incredibly large, inherently ambiguous, and difficult to annotate or evaluate (Wiegreffe et al., 2020; Latcinnik and Berant, 2020). Furthermore, quantifying the model's dependence on free-form explanations is also challenging (Camburu et al., 2020). We address these challenges by proposing an unsupervised method that uses contrastive prompts, which require the model to *explicitly* contrast different possible answers in its explanation (Table 1).

i) I picked up a bag of peanuts and raisins for a snack. I wanted a sweeter snack out so I ate the __ for now. *Contrastive Expl. - Peanuts are salty while raisins tend to be sweet.*
ii) The geese prefer to nest in the fields rather than the forests because in the __ predators are more hidden. *Contrastive Expl. - Forests are denser than fields*

Table 1: Examples of Winograd Schema Instances where the correct and incorrect answer choices are highlighted in blue and red respectively. Choices are *contrasted* along attributes like taste (for i) and density of vegetation (for ii) by humans to explain why they prefer some answer choice.

Our approach is based on a key observation: Many commonsense reasoning tasks require the comparison or contrast of plausible alternatives along a distinguishing attribute. For instance, in Table 1, the differentiating attributes for the two answer choices maybe taste (for i) and vegetation density (for ii). People commonly use contrastive explanations to explain their reasoning (Miller, 2018). Rather than asking "Why P?", they ask "Why P rather than Q?", where Q may be implicit from the context. For example, instead of justifying why raisins are the appropriate choice, people tend to ex-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4179–4192
August 1–6, 2021. ©2021 Association for Computational Linguistics

plain why they are more likely than peanuts. Miller (2018) also argues that such contrastive explanations are computationally efficient, as they only require focusing on the limited set of reasons that might make one answer more likely than the other instead of exhaustively enumerating all possible reasons for an answer. For instance, the raisin's taste (not its size, temperature, etc.) in Table 1 is adequate to explain why it is the best answer.

Our goal is to enable PLMs that explain their predictions to similarly benefit from such constraints. We develop a small set of contrastive generation prompts that can be in-filled by a PLM such as T5 (Raffel et al., 2020) or BART (Lewis et al., 2020) (see Table 3). These templates are designed to cover a multitude of language patterns used by humans to compare and contrast entities. Another PLM then conditions on both the original input and the generated contrastive explanation, to predict the final answer. This approach is inspired by Shwartz et al. (2020), who also use textual prompts to query the PLM with clarification questions. However, their prompts are generic while we prompt for instance-specific information.

Our approach shows quantitative improvements in task performance over two existing methods for model explainability (Shwartz et al., 2020; Latcinnik and Berant, 2020), for two commonsense reasoning tasks: the Winograd Schema Challenge (Levesque et al., 2012) and multiple-choice question answering about physical commonsense (Bisk et al., 2020). Our gains in the zero-shot setting are especially notable, outperforming the best reported results on publicly available PLMs and improving over Shwartz et al. (2020) by up to 11%. We also show, through human evaluations, that contrastive explanations are deemed more useful for solving the original task compared to generic clarification questions. Finally, contrastive explanations can be semantically perturbed to quantify the model's dependence on them by flipping the contrast in the explanation to support the foil, facilitating quantification of model faithfulness.[1]

2 Related Work

Models that rationalize their decisions by extracting a contiguous subsequence of the input as an explanation (Lei et al., 2016; DeYoung et al., 2020; Paranjape et al., 2020) are inadequate in explaining commonsense reasoning tasks that require knowledge that is implicit in the input. Such tasks necessitate PLMs to rely on embedded parametric knowledge. Recent work use free-form textual explanations to generate explanations for commonsense reasoning tasks like SNLI (Camburu et al., 2018), Winograd Schemas (Zhang et al., 2020) and CommonsenseQA (Rajani et al., 2019b) through explicit human supervision, which are inherently ambiguous, incomplete and consequently, expensive to collect and evaluate on (Camburu et al., 2019b,a; DeYoung et al., 2020). Most recently, Latcinnik and Berant (2020) use an unsupervised approach to generate free-form explanations as sequences of tokens that are not well-formed sentences. In contrast, our method uses specialized prompts to generate well-formed human-interpretable explanations without any additional supervision.

Specialized prompts have been shown useful for extracting knowledge from PLMs in a targeted manner (Petroni et al., 2020; Richardson and Sabharwal, 2020; Talmor et al., 2020; Donahue et al., 2020; Lin et al., 2019) and improving performance on downstream tasks (Brown et al., 2020; Shin et al., 2020). Most relevant to our work is the self-talk model of Shwartz et al. (2020), an unsupervised approach using a fixed set of clarification questions as prompts to elicit knowledge from PLMs for commonsense reasoning tasks. Our work differs by focusing specifically on contrastive PLM prompts, which we find further improve performance by eliciting explanations which are highly relevant to the classification decision (Section 6).

Our approach to contrastive reasoning is also closely related to *counterfactuals*, which can be used to give contrastive explanations, i.e., answers to "Why P rather than Q?", by providing a counterfactual case in which Q would have held. Ross et al. (2020) use this idea to generate contrastive explanations, while it has also been used for evaluation (Gardner et al., 2020) and training (Kaushik et al., 2019) with the aim of addressing model robustness. Most of this work explicitly constructs counterfactual cases by perturbing the input data of a task in order to produce changes in the output label. In contrast, we do not construct counterfactual *inputs*, but aim to explicitly represent counterfactual *knowledge*: a contrast between the fact P and foil Q that, were it hypothetically *reversed*, would change the output label. We include an evaluation of our models on this question in Section 6.3.

---

[1] Code is available at `https://github.com/bhargaviparanjape/RAG-X`

Dataset Instance	Human-Authored Contrastive Explanation
Winograd Schema	
1. The party was more interesting and upliﬁng than the funeral because the __ was rigid.	○ Parties are for celebrating while funerals are for mourning ○ People wear colorful clothes at parties and black at funerals
2. The geese prefer to nest in the fields rather than the forests because in the _ predators are more hidden.	○ Forests are dense while fields are sparse ○ Forests have more predators than fields.
PIQA	
1. How do you get strong hamstrings? __ (a) work out your upper body (b) work out your legs	○ Hamstrings are located in the legs while biceps are located in the upper body
2. How do you flood a room? __ (a) fill it with objects (b) fill it with water	○ Filling it with objects can clutter a room while filling it with water floods the room.

Table 2: Examples of commonsense tasks that can be explained using contrastive language and some contrastive explanations authored by in-house annotators. The Fact and Foil are marked in the input.

3 Contrastive Explanations

We present the theory of contrastive explanations adopted in this work (Section 3.1) and the intuition behind using them for commonsense reasoning tasks (Section 3.2).

3.1 Definition and Motivation

A contrastive explanation is generally defined as an answer to a counterfactual question of the form "Why P rather than Q?" for two potential hypotheses P and Q that can follow from some event E. It explains why some *fact* P occurred instead of some *foil* Q, where Q can be implicit (Hesslow, 1988; Lipton, 1990; Miller, 2019). A good contrastive explanation points to differences between the fact and foil with regard to certain attributes, not just conveying that the fact has a certain attribute. Table 1 shows examples of contrastive explanations that differentiate between peanuts and raisins (on the basis of taste) or forests and fields (on the basis of vegetation densities) to explain the more probable answers to Winograd Schema instances.

Previous studies (Miller, 2019) in philosophy, psychology, and cognitive science show that humans use such contrastive explanations when explaining their decisions to each other. Importantly, Miller (2018) also argues that contrastive explanations are computationally efficient – exhaustively describing all causes for the occurrence of an event P is harder than only enlisting causes for why another event Q did not occur instead of P.

3.2 Contrastive Explanations for Commonsense Reasoning Tasks

Many recently proposed commonsense reasoning tasks are framed in a multiple-choice format that facilitates contrastive explanation (see Table 2). In this study, we focus on the following two tasks.

The **Winograd Schema Challenge** (Levesque et al., 2012, WSC) is a pronoun coreference resolution task designed as a hard benchmark for evaluating everyday knowledge and commonsense reasoning (Zhang et al., 2020). For instance, in the sentence "The city councilmen refused the demonstrators a permit because they feared violence," the pronoun *they* must be disambiguated between fact (*the city councilmen*) and foil (*the demonstrators*). Both fact and foil are explicit in such sentences.

The **Physical Interaction Question Answering** (Bisk et al., 2020, PIQA) challenge is designed to test knowledge of physical commonsense. PIQA requires choosing between which one of two *solutions* is a better way of achieving a *goal* posed as a question (see Table 2). PIQA questions relate to physical properties of entities, their affordances, and how they can be manipulated. The fact and foil are explicit in the two solutions, which typically differ from one another by a short noun phrase.

To validate our intuition that contrastive reasoning is instrumental in these tasks, we performed a pilot study with 10 annotators over 100 commonsense questions from Winogrande and PIQA. We instructed them to answer the questions and explain their reasoning, but gave no specific instructions about what the explanations should look like. Examples are shown in Table 2. In 76% of Winogrande and 64% of PIQA examples, annotators explicitly contrasted the fact and foil. The frequent use of certain phrase structures, like *P are __ while Q are __*, strongly informed our method for generating them (Section 4).

4 Our Approach

We assume the input to a commonsense reasoning problem consists of a textual context c which contains a placeholder _, and two marked answer

Prompt Pattern	Commonsense Example & Model Generated Explanation
Personal Characteristics $\Longrightarrow$ P likes/likes to _ while Q likes/likes to _ P likes/likes to _ while Q does not like/like to _ P prefers/prefers to _ while Q prefers _ Q prefers _ while P does not prefer/prefer to _ Q thinks _ while P thinks/does not think _	Megan said it would be liberating to go out without makeup like Elena does since __ never wore makeup `Explanation: Elena likes to be natural while Megan likes to wear lipstick`
Object Characteristics P is taller/shorter/smaller/larger/slower/faster than Q $\Longrightarrow$ P is/are _ while/but/however Q is/are _ Q has/have _ while/but/however P has/have _ P has/have more/less _ than Q P is/are _ than Q	How to tie pieces of paper together? __ (a) Thread ruler through the holes (b) Thread ribbon through the holes `Explanation: Ruler is hard while a ribbon is flexible`
Spatial/Temporal Contrast $\Longrightarrow$ P is inside/outside/above/below Q _ is closer to P and farther away from Q P is to the right/left of Q Q takes longer to _ than P	Emily looked up and saw Patricia racing by overhead. __ was on the ramp. `Explanation: Emily is below Patricia`
Use cases and causes P is used for _ Q P is used to do Q _ $\Longrightarrow$ P is used for/to/in _ while Q is used for/to/in _ Q is used _ while P is used _ Q because _ while P because _ Q can cause _ while P results in _	To prepare the puff pastry for your pie, line a baking sheet with parchment. Then __ (a) Unroll the pastry, lay it over baking twine. (b) Unroll the pastry, lay it over fishing line. `Explanation: Baking twine is used in baking while fishing line is used in fishing`

Table 3: Contrastive Patterns and Examples of outputs generated by the T5-large model. The pattern the PLM completes are marked $\Longrightarrow$.

choices a_1 and a_2 corresponding to the fact and foil (Table 2, left column). Let c_x denote substitution of x for the placeholder in c. The task is to predict whether c_{a_1} or c_{a_2} is more likely to be true, i.e., whether a_1 or a_2 best completes the context.

Our approach has two stages: First, an **Explainer PLM** P_{expln} generates contrastive explanations (Section 4.2) by infilling preset *contrastive templates* (Sec. 4.1) on the basis of c, a_1, and a_2. Then, a **Task Model** P_{LM} selects the correct answer conditioned on both the context and the generated explanations (Sec. 4.3).

4.1 Contrastive Templates

We develop a list of contrastive templates on the basis of an annotation study. For 250 instances from Winogrande and PIQA, we asked three annotators to explain why one answer is more likely than the other. We manually examined these explanations and abstracted them into templates containing at least two placeholders: two for the fact and foil being contrasted, and possibly more corresponding to the properties they are being contrasted on. For instance, *peanuts are salty while raisins are sweet* becomes *Q are _ while P are _*. We retained templates used by annotators at least 10 times. Table 3 shows several examples. A template is converted into an explanation by replacing placeholders for the fact and foil with answers a_1 and a_2 and the remaining placeholders with the appropriate contrastive information.

We evaluate the quality and coverage of our templates with another round of human evaluation. For 100 WSC and PIQA examples, we ask three annotators to either write contrastive explanations using one or more of the templates, or indicate that none of the them were appropriate. Annotators used the templates in over 82% of cases, indicating high coverage for the tasks we study.

4.2 Generating Explanations

Let t denote a contrastive template. We write t_{a_1,a_2} to denote the customization of t to an input by filling its marked placeholders for fact and foil with the answer choices. For instance, in Figure 1, the template *P are _ while Q are _* is customized to *Fields are _ while forests are _*.[2] A full explanation may be produced by filling the remaining gaps in t_{a_1,a_2} by leveraging an infilling language model, the explainer P_{expln}.

We first construct a neutral context c_{a_0} by filling c's placeholder with a task-specific neutral answer

---

[2] In practice, we randomize the order of a_1 and a_2 when customizing the template.

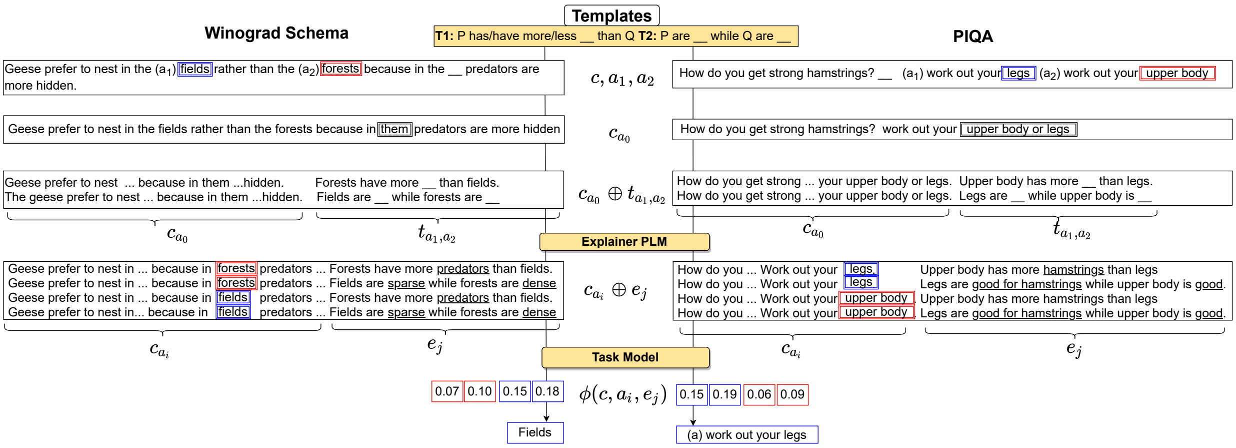

Figure 1: (1) A commonsense reasoning instance (c, a_1, a_2) is converted into a custom prompt $c_{a_0} \oplus t_{a_1,a_2}$ as input for the explainer PLM (2) The combination of input and explanation $(c_{a_i} \oplus e_j)$ is used by task model to score $a_i \forall i \forall j$. For a_1 and a_2, scores are aggregated over templates.

that does not indicate if a_1 or a_2 is correct. For Winogrande Schemas, c_{a_0} is constructed using the ambiguous pronoun in c (*them* in Figure 1). For PIQA, c_{a_0} is constructed as "$c \oplus a_1$ or a_2", where $\oplus$ is string concatenation, e.g., *upper body or legs* in Figure 1 (More dataset-specific details are in Section 5.2). We then prepend c_{a_0} to the customized template t_{a_1,a_2} and use it as input to the infilling language model to fill in the remaining gaps in the template. We use the maximum likelihood candidate phrases from top-K decoding to transform the template into a full explanation e.

We use a list of templates $t_1, \ldots, t_n$ to generate a list of candidate explanations $\mathsf{e}_1, \ldots, \mathsf{e}_n$ for each input, which are all fed into the task model. We also use some task-specific heuristics to reduce the number of prompts for each example, detailed in Appendix A.

4.3 Task Model

Given the context and answer choices (c, a_1, a_2) and a list of explanations $\mathsf{e}_1, \ldots, \mathsf{e}_n$, the second stage of our pipeline is a binary classifier between a_1 and a_2 which marginalizes over the explanations. We first assign a score to each answer $a \in \{a_1, a_2\}$ and explanation $\mathsf{e} \in \{\mathsf{e}_1, ..., \mathsf{e}_n\}$:

$$\phi(c, a, \mathsf{e}) = \frac{1}{k} \log \mathsf{P}_{\mathrm{LM}}(c_a \oplus \mathsf{e}),$$

where c_a denotes the substitution of a into c, P_{LM} is string probability under the task language model, and k is the string length of $c_a \oplus \mathsf{e}$. We use ϕ as input to a logistic regression classifier which marginalizes over explanations:

$$\mathsf{P}(a \mid c, a_1, a_2) = \frac{\sum_i^n e^{\phi(c,a,\mathsf{e}_i)}}{Z},$$

where Z is a normalizer over a_1 and a_2. At initialization, ϕ uses a pretrained language model, and we fine-tune it to minimize the cross-entropy loss of $\mathsf{P}(a^* \mid c, a_1, a_2)$, where a^* is the correct answer. We do not fine-tune the explainer PLM since the top-K beam decoding is a discrete operation that is hard to backpropagate through. In the zero-shot setting (where the task PLM is not fine-tuned) and during inference, the answer is predicted by aggregating scores assigned to an answer by all n explanations: $\operatorname{argmax}_{a_i} \sum_j \phi(c, a_i, \mathsf{e}_j)$.

5 Experimental Setup

5.1 Baselines

Context-Only We experiment with a baseline that does not condition on explanations at all. Here,

$$\phi(a, c) = \frac{1}{k} \log \mathsf{P}_{\mathrm{LM}}(c_a),$$

and gold answer is $\operatorname{argmax}_{a_i} \phi(a_i, c)$

Unconstrained Generation Latcinnik and Berant (2020) generate explanations from a PLM by beam-decoding a free-form sequence termed a *hypothesis* which is then used by a classifier to solve the task. The model is trained end-to-end and loss terms are added to encourage the hypothesis to sound natural. Explanation generation is otherwise unconstrained. For fair comparison with our approach, we do not fine-tune the explainer PLM (more details are in Appendix C).

Self-Talk Shwartz et al. (2020) propose an unsupervised model that uses a PLM as the answer scorer and a (possibly different) PLM as a knowledge source, similar to our framework. They formulate the process of obtaining relevant knowledge as *self-talk* with the following steps: 1) completing clarification question prefixes such as "what is the definition of ..." conditioned on input context, 2) generating their corresponding answers (clarifications), and 3) conditioning on the clarification questions and answers to make predictions. The key difference between their approach and ours is in the choice of prompts for the PLM, and the kinds of knowledge the prompts seek. While Shwartz et al. (2020) draw inspiration from inquiry-based discovery learning (Bruner, 1961), we target contrastive reasoning.

5.2 Implementation details

We use BART-Large (Lewis et al., 2020) and T5 (Raffel et al., 2020) as the explainer PLMs. Hyperparameters for infilling are given in Appendix C. For a fair comparison of all models, we use GPT2-XL (Radford et al., 2019) as the task model that estimates $\phi(c, a, \mathtt{e})$. GPT2-XL is the best performing PLM used by Shwartz et al. (2020) for WSC and PIQA tasks. Hyperparameter details about finetuning are given in Appendix C. We describe dataset specific modifications made to create c_{a_0}, c_{a_1}, and c_{a_2} in Section 4.2.

Winograd Schema Challenge (WSC) We experiment on (i) the SuperGLUE (Wang et al., 2019) version of the WSC consisting of 285 examples of anaphora (pronoun) resolution; (ii) Winogrande (WGRD) (Sakaguchi et al., 2020), a large scale crowdsourced version of the WSC; and (iii) WINOGENDER (WGND), a diagnostic dataset created to measure gender bias in models for ambiguous pronoun resolution (Rudinger et al., 2018).

Each instance provides two answer choices, which we use directly as a_1 and a_2. For the neutral answer c_{a_0}, we use the sentence with the original ambiguous pronoun. Since Winogrande has a blank space _ for the answer, we replace it with the most likely pronoun under a masked language model (BERT), following Shwartz et al. (2020). c_{a_1}, c_{a_2} are obtained by replacing the blank space or pronoun with the answer choice.

Physical Interaction Question Answering (PIQA) (Bisk et al., 2020) PIQA provides two answer choices which mostly vary from each other on a substring (e.g., "work out your [upper body]/[legs]"). We use these differing substrings as a_1=legs and a_2=upper body. For the neutral answer a_0, we combine the answers into "a_1 or a_2" (upper body or legs). In the cases where a_1 or a_2 is longer than 2 words, we include an *or* between the full answers. More details and examples are presented in Appendix A. We use question-answer pairs for c_{a_1} and c_{a_2}.

6 Experimental Results

In this section, we present an extensive evaluation of our approach, demonstrating performance gains which are independently verified by human judges.

6.1 Task Performance

We report task accuracy as a proxy for explanation quality. Table 4 compares the task performance of our model with the baselines defined in Section 5.1. We observe that generating and conditioning on additional information from PLMs improves performance over just using the original input (Row 1 vs. 2-6). Using templates to prompt the PLM for specific knowledge is better than unconstrained generation of text (Row 2 vs. 3-6). Contrastive explanations outperform previous work that use clarification questions in self-talk (Shwartz et al., 2020). The T5-Large explainer already surpasses the results of self-talk despite being smaller than GPT2-XL, demonstrating the impact of using contrastive explanations over clarification questions.

We also observe that larger explainer PLMs (going from T5-Large to T5-11B) yield higher performance. Our zero-shot results with T5-11B are the highest reported on Winogrande, PIQA and WSC for an open-sourced model.[3]

Finally, our approach gets smaller improvements when finetuning the task model. This suggests that some of the reasoning is still learned implicitly by the task model. Figure 2 shows task performance with various training data sizes of Winogrande, indicating a larger gap between the Context-Only baseline and our approach when training data is scarce.

6.2 Human Evaluation

Setup Following the human evaluation setup of Shwartz et al. (2020), we sample up to 50 highest-

---

[3] The zero-shot SOTA model (Brown et al., 2020) uses the 175B parameter GPT-3 model, which would likely also be a stronger explainer for our approach, but we did not have access to it.

	Explainer PLM (# Params)	Task model	WGRD ZS	FT	PIQA ZS	FT	WSC ZS	WGND ZS
1. Context-only	GPT2-XL (1.5B)	GPT2-XL	54.8	77.9	62.6	80.1	61.5	60.0
2. Unconstrained	GPT2-XL		54.9	77.8	63.9	80.7	61.4	60.0
3. Self-Talk	GPT2-XL		55.1	78.4	69.5	82.3	62.0	61.3
4. Contrastive	BART-Large(680M)		56.8	78.9	71.8	82.8	63.2	62.9
5. (Ours)	T5-Large (770M)		59.2	79.1	72.5	83.5	63.5	63.2
6.	T5-11B(11B)		**60.3**	**79.6**	**73.4**	**83.9**	**64.1**	**63.5**

Table 4: Test set accuracy on Winogrande (WGRD), PIQA, WSC and Winogender (WGND). ZS is Zero-shot models while FT is fine-tuned models. WSC and Winogender don't have training data for finetuning. Across all our models, the task model is GPT2-XL for fair comparison with (Shwartz et al., 2020) and to make finetuning tractable.

Metric	Self-Talk (Reported) WGRD	 PIQA	Self-Talk WGRD	 PIQA	Contrastive WGRD	 PIQA
Relevant	68	60	70.4	61.7	73.1	70.7
Factual	46	42	40.8	38.8	43.0	39.4
Helpful	24	26	22.5	27.7	42.8	32.8
Grammatical	87.2	87.2	87.5	87.5	83.5	83.5
Flips	NA	NA	NA	NA	66.9	59.4

Table 5: Human Evaluation Results on Winogrande(WGRD) and PIQA. Reported human evaluation results on Self-talk are different from ours, which can be because of moderate levels of agreement (Fleiss Kappa κ = 0.43). Grammatiality is judged together for all datasets following (Shwartz et al., 2020). Only contrastive explanations can be flipped.

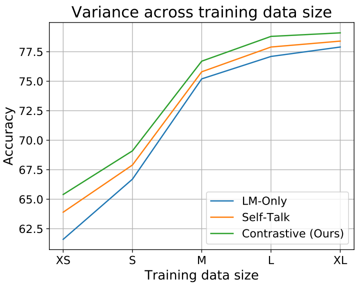

Figure 2: Performance variation in the finetuning setting across different sizes of Winogrande training data.

scoring explanations from PIQA and Winogrande examples which the T5-Large model got correct but the Context-Only baseline failed at. For comparison, we also include explanations from the self-talk model for evaluation.

Crowd workers are presented with a commonsense instance, the correct answer, and an explanation, and are asked to judge for: 1) *Grammaticality*, whether the explanation is grammatical; 2) *Relevance*, whether it's relevant to the topic of the text; 3) *Factual Correctness*, whether it's factually correct or likely true; and 4) *Helpfulness*, whether it adds helpful evidence for the correct answer. These metrics and definitions follow from Shwartz et al. (2020) with more details in Appendix B. The annotators are also shown the same explanation with fact and foil flipped (details in Section 6.3) and are asked to judge if the other answer is more likely than before if they assume the flipped explanation to be hypothetically true.

Results Table 5 shows the results of human evaluation of contrastive and self-talk explanations. The contrastive explanations are overwhelmingly preferred over self-talk explanations for relevance, factual correctness and helpfulness. They may be considered less grammatical because of in-filling noise (such as incomplete phrases). Table 6 presents some qualitative examples of instances where contrastive explanations improve over all baselines.

6.3 Analysis

We also analyze how much the task model relies on contrastive explanations for its decisions.

Flipping Explanations Our choice of contrastive language templates facilitates a novel way to evaluate explanation usefulness in prediction. The contrast in the explanation can be reversed by flipping the position of the fact and the foil in the explanation. If the choice between fact and foil actually depends on the contrastive explanation, then the flipped explanation should provide a hypothetical situation where the foil is more likely than the fact. For instance, "peanuts are salty while raisins are sweet," when switched to "raisins are sweet while peanuts are salty," may provide evidence that *peanuts* is a more likely label for the example in Table 1 (i). This may cause a model that uses the explanation to flip its prediction and lead to a drop in accuracy. The magnitude of drop can quantify the extent to which the model relies on the contrast

Example	Unconstrained	Self-Talk	Contrastive
(i) Ian volunteered to eat Dennis's menudo after already having a bowl because he despised eating intestine.	Dennis's menudo was disgusting.	What are the properties of a menudo? A menudo is made from the intestines of a pig	Dennis is a vegetarian while Ian is a carnivore. Dennis has menudo while Ian has volunteered to eat Denni's menudo.
(i) The GPS and map helped me navigate home. I got lost when the it got turned upside down.	because the GPS and map helped me navigate home.	What is going on here? The iphone app is not working.	The GPS can tell me where I am but the map can't. The GPS is right-side-up while the map is upside down
(ii) I helped my sister find her gold necklace. She couldn't wear her woven necklace to the ball because it was so casual.	She couldn't wear her woven necklace.	What are the properties of gold? The properties of gold are listed below.	Gold necklace is used for formal occasion while woven necklace is used for casual occasion.

Table 6: Qualitative Examples on Winogrande where contrastive explanations (using T5-11B explainer) improve task performance over baselines.

Explainer PLM	WGRD		PIQA		WSC	WGND
	ZS	FT	ZS	FT	ZS	ZS
BART-Large	53.9 (5.4)	75.9 (4.0)	66.5 (7.9)	79.1 (4.6)	59.1 (6.9)	58.7 (7.1)
T5-Large	56.2 (5.3)	75.3 (5.0)	68.1 (6.5)	80.2 (4.2)	60.2 (5.5)	59.0 (7.1)
T5-11B	57.6 (4.5)	76.1 (4.7)	69.5 (5.4)	81.0 (3.6)	61.1 (3.3)	59.0 (5.8)

Table 7: Flipped evaluation results for contrastive explanation models. Reporting test accuracy across all datasets. Percent drop in performance as a result of flipping is indicated in parentheses.

Input	WGRD	PIQA
Fully abstracted	63.2	54.6
Abst. after expl.	70.4	64.9
No abstraction	79.1	83.5

Table 8: Evaluation of fine-tuned T5-Large contrastive models on Winogrande with abstracted answers.

provided in the explanation. In fact, humans also deem the flipped explanation to imply the opposite label in a majority of cases (Table 5), indicating that our contrastive explanations frequently capture contrastive properties that the labels truly rely on.

Table 7 shows the flipped evaluation results. We observe declines in accuracy of up to 8%, indicating that the model does use some contrastive knowledge to reason about the task. Finetuned models show a smaller decline in accuracy compared to the zero-shot setting. In this case, the task model may be directly fitting the data in lieu of relying on the knowledge conveyed by the explanation.

Abstracting Fact and Foil Given input context c (consisting of the fact and foil a_1, a_2) and an explanation e, the explainer PLM P_{expl} infills its explanation e on c while the task model P_{LM} conditions on both c and e. We can test the quality of the generated explanations and the task model's reliance on them by forcing the task model to rely on e when information in input c is restricted. One potential way to do so is to scrub the identities of the fact and foil, a_1 and a_2, from c.

We replace the fact and foil with placeholder tokens to create an abstract context c'. For instance, the example in Table 6 (ii) becomes "The `<mask1>` and `<mask2>` helped me navigate ... down.", where the model must now choose between `<mask1>` and `<mask2>`.[4] Running the task model on c' lower-bounds the performance possible without knowing answer identities. We can now test the relevant answer-based knowledge contrastive contained in the *explanations* by allowing the explanation model to see the original answers in c, but then abstracting them out when passing the input context and explanations to the task model. More formally, the task model conditions its decision on c' and e'. For Table 6 (ii) c' and e' are "The `<mask1>` and `<mask2>` helped me navigate ... down." and "The `<mask1>` is right-side-up while the `<mask2>` is upside down." Since only the explainer PLM is shown answer identities, the task model's decision is conditionally independent of the answer identities given the explanation.

Experiments on Winogrande and PIQA in the

---

[4] More examples of abstracted contexts and explanations are given in the Appendix (Table 11).

Model	Acc.
Random	20.0
Baseline	37.2
Self talk	26.9
Contrastive (V)	38.1
Contrastive (MM)	37.4
Banerjee and Baral (2020a)	38.8

Table 9: Zero-shot test performance on CommonsenseQA for baselines as well as contrastive models which ensemble fact/foil pairs by voting (V) and maximum margin (MM). The best reported unsupervised performance (Banerjee and Baral, 2020b) uses ConceptNet, which was used to construct the dataset.

fine-tuned setting (Table 8) show that performance improves significantly when the task model conditions on both c' and e' compared to a fully abstracted contrastive baseline that only conditions on c' (from 63.2 to 70.4 for Winogrande), covering almost half of the gap between the fully abstracted setting and the non-abstracted original model (79.1). This indicates that our contrastive explanations encode a significant amount of information required for commonsense tasks. Even if the full model does not always use the explanations, these evaluations show that our contrastive explanations contain rich task-relevant knowledge, and suggest that future work might focus on how to better make use of this signal.

6.4 Generalizability of Prompts

The set of contrastive prompts used in our framework are curated from an in-house analysis of training instances from Winogrande and PIQA datasets. To determine the generalizability of these prompts for other commonsense reasoning tasks, we also experiment with the CommonsenseQA dataset (Talmor et al., 2019), which consists of multiple-choice questions created over ConceptNet – "Where on a river can you hold a cup upright to catch water on a sunny day? a) waterfall, b) bridge, c) valley, d) pebble, e) mountain". Since there are more than two answer choices to contrast, we convert each instance into 10 pairwise (binary) classification instances. Contrastive explanations are generated for each pairwise decision in the zero-shot setting, similar to Winograd and PIQA datasets. To choose the final answer, we consider two inference procedures: (a) *Vote:* The answer that receives the maximum number of votes across all binary classification instances is selected, and (b) *Maximum Margin:* The choice with the maximum difference (margin) between answer likelihoods for any binary classification instance is selected. In Table 9, we observe that self-talk significantly hurts performance for this dataset. On the other hand, contrastive explanations are found to be useful and approach the zero-shot performance of the state-of-the-art, which uses ConceptNet (Banerjee and Baral, 2020b). These results demonstrate that the set of contrastive prompts are generalizable to other commonsense reasoning datasets, and that while our contrastive prompts are limited to contrasting two answer choices at a time, the framework can be easily extended to tasks with multiple foils.

7 Conclusion

We show it is possible to prompt pretrained language models (PLMs) to generate contrastive explanations of their reasoning patterns, inspired by explanations that humans naturally provide for their reasoning. Conditioning model decisions on these explanations improves performance on two commonsense reasoning benchmarks, and humans judge the explanations to be highly relevant and helpful in comparison to prior work. We also showed how contrastive explanations can facilitate in-depth evaluations of faithfulness by flipping or abstracting the fact and foil, finding that our explanations encode a significant amount of information relevant to the classification decision, and in many cases models rely on the contrast in the expected way. While we have shown that our method is flexible enough to apply to multiple-choice commonsense tasks with many foils, leveraging contrastive reasoning in a wider variety of open-ended tasks remains an exciting challenge for future work.

Acknowledgements

This research was supported by ONR N00014-18-1- 2826, DARPA N66001-19-2-403, ARO W911NF16-1-0121 and NSF IIS-1252835, IIS-1562364, an Allen Distinguished Investigator Award, and the Sloan Fellowship. We thank Vered Shwartz, Mandar Joshi, Divyansh Kaushik, H2Lab members, UW NLP and the anonymous reviewers for their helpful comments and suggestions.

References

Pratyay Banerjee and Chitta Baral. 2020a. Self-supervised knowledge triplet learning for zero-shot question answering. *arXiv preprint arXiv:2005.00316*.

Pratyay Banerjee and Chitta Baral. 2020b. Self-supervised knowledge triplet learning for zero-shot question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 151–162, Online. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jerome S Bruner. 1961. The act of discovery. *Harvard educational review*.

Oana-Maria Camburu, Eleonora Giunchiglia, Jakob Foerster, Thomas Lukasiewicz, and Phil Blunsom. 2019a. Can i trust the explainer? verifying post-hoc explanatory methods. *arXiv preprint arXiv:1910.02065*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pages 9539–9549.

Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, and Phil Blunsom. 2019b. Make up your mind! adversarial generation of inconsistent natural language explanations. *arXiv preprint arXiv:1910.03065*.

Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, and Phil Blunsom. 2020. Make up your mind! adversarial generation of inconsistent natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4157–4165, Online. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models' local decision boundaries via contrast sets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1307–1323.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. *arXiv preprint arXiv:1908.07898*.

Germund Hesslow. 1988. The problem of causal selection. *Contemporary science and natural explanation: Commonsense conceptions of causality*, pages 11–32.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2019. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Veronica Latcinnik and Jonathan Berant. 2020. Explaining question answering models through text generation. *arXiv preprint arXiv:2004.05569*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2019. Commongen: A constrained text generation challenge for generative commonsense reasoning. *arXiv preprint arXiv:1911.03705*.

Peter Lipton. 1990. Contrastive explanation. *Royal Institute of Philosophy Supplements*, 27:247–266.

Tim Miller. 2018. Contrastive explanation: A structural-model approach. *arXiv preprint arXiv:1811.03163*.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. 2020. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*.

Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. An information bottleneck approach for controlling conciseness in rationale extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1938–1952, Online. Association for Computational Linguistics.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *arXiv preprint arXiv:2005.04611*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. *NAACL HLT 2018*, page 180.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019a. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019b. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Kyle Richardson and Ashish Sabharwal. 2020. What does my qa model know? devising controlled probes using expert knowledge. *Transactions of the Association for Computational Linguistics*, 8:572–588.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Alexis Ross, Ana Marasović, and Matthew E Peters. 2020. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.

Sarah Wiegreffe, Ana Marasovic, and Noah A Smith. 2020. Measuring association between labels and free-text rationales. *arXiv preprint arXiv:2010.12762*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1008–1025.

Hongming Zhang, Xinran Zhao, and Yangqiu Song. 2020. Winowhy: A deep diagnosis of essential commonsense knowledge for answering winograd schema challenge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5736–5745.

A Generating Contrastive Templates

Table 12 shows the complete list of contrastive patterns used in our work, categorized under different types of attributes/properties. For templates with no place holders for the explainer to fill out, we only replace placeholders for answers (fact and foil). Table 10 lists $a_0, a_1, a_2, c_{a_0}, c_{a_1}, c_{a_2}$ for different examples in Winogrande and PIQA to explain dataset specific transformations made by our approach.

Detection of P, Q: For WSC, the fact and foil are typically 1-word nouns. However, they may by qualified in the context and these qualifiers are important for contrasting. For instance, in the WSC example "She remembered how annoying it is to dust her wood chair so she bought a plastic table instead.", chair and table are the fact and foil. Their qualifiers wood and plastic are important for the construction of the contrast. Hence we retain these qualifiers when preparing prompts for the explainer PLM. Similarly, for PIQA, qualifiers are retained in the prompts.

Case filtering: We detect case of entities and accordingly filter out templates that are ungrammatical depending on whether the fact and foil are singular/plural.

Template filtering for WSC: For examples that do not contain PERSON named entities, we do not include prompts about personal characteristics. Similarly, for examples that contain PERSON named entities, Temporal, Usecase and some spatial patterns were left out.

Template filtering for PIQA: We remove all templates about personal characteristics as this dataset deals with physical commonsense.

B Human Evaluation

The annotation task was carried out in Amazon Mechanical Turk, following (Shwartz et al., 2020). To ensure the quality of annotations, workers were required to be located in the US, UK, or Canada, and have a 99% approval rate for at least 5000 prior tasks. Annotators were paid .30$ per HIT to ensure participants get approximately $15/hr if they are doing the task. Annotation were aggregated from 3 workers using majority vote. The annotations yielded moderate levels of agreement, with Fleiss Kappa κ = 0.43 (Landis and Koch, 1977).

C Hyperparameters

Explainer PLM For T5 we use special symbols `<extra_id_0>` and `<extra_id_1>` in place of the blanks (_) in our templates. We observe that T5 is able to replace these tokens with multi-word phrases. For BART, we substitute blanks with a sequence with four `[MASK]` tokens to encourage generating multiple words. BART can choose to delete a `[MASK]` token during generation. Top-K decoding was done with a beam size of 200 and a maximum output sequence length of 20 for T5 models and 100 for BART. This is because both T5 is pre-trained to in-fill by only generating missing phrases while BART is pre-trained to decode the entire input with missing phrases filled in. We used early stopping for BART.

Task PLM Task PLM was finetuned for 20 epochs, using BertAdam optimizer with a learning rate of $2e-5$, batch size of 8, and dropout of 0.1, following (Latcinnik and Berant, 2020).

Self-Talk (Shwartz et al., 2020) generate multiple clarification questions conditioned on the context, by 1) concatenating one of several question prefixes to the input prompt or question; and 2) generating 5 questions for each prefix using Nucleus sampling with $p = 0.2$, i.e., sampling from the top 20% tokens(Holtzman et al., 2019) limiting the question length to up to 6 tokens excluding the prefix. For each well-formed question, they generate multiple answers using a similar method. They

Winogrande
Ian volunteered to eat Dennis's menudo after already having a bowl because _ despised eating a_1 : Ian a_2 : Dennis a_0 : he c_{a_0} : Ian volunteered to eat Dennis's menudo after already having a bowl because he despised eating c_{a_1} : Ian volunteered to eat Dennis's menudo after already having a bowl because Ian despised eating c_{a_2} : Ian volunteered to eat Dennis's menudo after already having a bowl because Dennis despised eating
PIQA (difference between answers is 1-2 words)
To prepare carrots before cooking with them, you can a_1 : Run them in the sink under boiling water a_2 : Run them in the sink under cold water a_0 : boiling water or cold water c_{a_0} : To prepare carrots before cooking with them, you can run them in the sink under boiling water or cold water c_{a_1} : To prepare carrots before cooking with them, you can run them in the sink under boiling water c_{a_2} : To prepare carrots before cooking with them, you can run them in the sink under cold water
PIQA (difference between answers is larger)
To prevent gunk buildup in cup holders of a car, a_1 : place coffee filters inside of the cup holders. a_2 : pour a thin layer of oil into the cup holders. a_0 : place coffee filters inside of the cup holders or pour a thin layer of oil into the cup holders. c_{a_0} : To prevent gunk buildup in cup holders of a car, place coffee filters inside of the cup holders or pour a thin layer of oil into the cup holders c_{a_1} : To prevent gunk buildup in cup holders of a car, place coffee filters inside of the cup holders c_{a_2} : To prevent gunk buildup in cup holders of a car, pour a thin layer of oil into the cup holders

Table 10: Examples of Winogrande and PIQA, with fact, foil , neutral answer and respective substituted contexts used in our approach for prompting the explainer PLM or computing answer likelihood.

Original Input: The geese prefer to nest in the fields rather than the forests because in the _ predators are more hidden.
(i) Context-Only Input to task model: The geese prefer to nest in the `<mask1>` rather than the `<mask2>` because in the _ predators are more hidden.
(ii) Fully Abstracted Input to explainer: The geese prefer to nest in the `<mask1>` rather than the `<mask2>` because in the _ predators are more hidden. Generated Explanation: `<mask1>` is smaller than `<mask2>` Input to task model: The geese prefer to nest in the `<mask1>` rather than the `<mask2>` because in the _ predators are more hidden. `<mask1>` is smaller than `<mask2>`
(iii) Abstraction after Explanation Input to explainer: The geese prefer to nest in the fields rather than the forests because in the _ predators are more hidden. Generated Explanation: Forests have more predators than fields Input to task model: The geese prefer to nest in the `<mask1>` rather than the `<mask2>` because in the _ predators are more hidden. `<mask2>` have more predators than `<mask1>`

Table 11: Input to Explainer and Task model for Abstractive Evaluation

limit the answer length to 10 generated tokens, and use Nucleus sampling with $p = 0.5$. Shwartz et al. (2020) only condition task prediction on a single clarification question and answer pair that increases the model's belief of a certain answer choice. Thus, the score of each answer choice is selected as the score of the text containing the clarification that most supports it, i.e., whose combination with it yields maximum language model likelihood.

Unconstrained Generation For unconstrained explanation baseline, maximum output sequence length was set to 20 and beam size for beam decoding was set to 200. Again we use early stopping.

Complete list of Contrastive Prompt Templates	Commonsense Task/Instance Type
Temporal:	PIQA (Consists of events)
OPT1 happened before/after OPT2	
OPT1 takes longer than OPT2	
OPT1 takes longer to _ than OPT2	
OPT1 happened for a longer time than OPT2	
Personal Characteristics:	WSC (if PERSON entity tag is detected)
OPT1 likes _ while OPT2 likes _	
OPT1 likes _ while OPT2 does not like _	
OPT1 likes to _ while OPT2 likes to _	
OPT1 likes to _ while OPT2 does not like to _	
OPT1 prefers _ while OPT2 prefers _	
OPT1 prefers _ while OPT2 does not prefer _	
OPT1 prefers to _ while OPT2 prefers to _	
OPT1 prefers to _ while OPT2 does not prefer to _	
OPT1 thinks _ while OPT2 thinks _	
OPT1 thinks _ while OPT2 does not thinks _	
Object Characteristic:	WSC and PIQA
OPT1 is/are smaller than OPT2	
OPT1 is/are larger than OPT2	
OPT1 is/are slower than OPT2	
OPT1 is/are faster than OPT2	
OPT1 is _ than OPT2	
OPT1 are _ than OPT2	
OPT1 is _ while OPT2 is _	
OPT1 is _ but OPT2 is _	
OPT1 is _ however OPT2 is _	
OPT1 are _ while OPT2 are _	
OPT1 are _ but OPT2 are _	
OPT1 are _ however OPT2 are _	
OPT1 has _ while/but/however OPT2 has/does not have _	
OPT1 have _ while/but/however OPT2 have/do not have _	
OPT1 is made of/to _ however OPT2 is made of/to _	
OPT1 is made of/to _ while OPT2 is made of/to _	
Spatial:	WSC and PIQA
OPT1 is above OPT2	
OPT1 is below OPT2	
OPT1 is to the right of OPT2	
OPT1 is to the left of OPT2	
OPT1 is inside OPT2	
OPT1 is outside OPT2	
_ is closer to OPT1 and father away from OPT2	
OPT1 is closer to _ while OPT2 is father away from _	
Usecase:	WSC(No PERSON entity) and PIQA
OPT1 can _ while OPT2 can/cannot _	
OPT1 is/can be used for OPT2	
OPT1 is/can be used to do OPT2	
OPT1 is/can be used for _ but OPT2 cannot	
OPT1 is/can be used for _ while OPT2 is used for _	
OPT1 is/can be s used for _ but OPT2 is used for _	
OPT1 is/can be used to _ while OPT2 is used to _	
OPT1 is/can be used to _ but OPT2 is used to _	
Causes:	WSC (No PERSON entity) and PIQA
OPT1 has _ because _ while OPT2 is _ because _	
OPT1 can cause _ while OPT2 causes/results in _	
Since _ it can OPT1 but not OPT2	
Since _ it can OPT1 but because it is not _ it can't OPT2	
Miscellaneous:	WSC (No PERSON entity) and PIQA
_ can be OPT1 but cannot be OPT2	
OPT1 means to _ while OPT2 means to _	
OPT1 is defined as _ while OPT2 is defined as _	
_ OPT1 _ OPT2	
_ OPT1 but not OPT2	
OPT1 exists while an OPT2 doesn't	

Table 12: Complete list of contrastive patterns used in this work.

SMS Spam Detection Through Skip-gram Embeddings and Shallow Networks

Gustavo José de Sousa, Daniel Carlos Guimarães Pedronette,
João Paulo Papa, Ivan Rizzo Guilherme
São Paulo State University - UNESP, Brazil
`{gustavo.sousa,daniel.pedronette,joao.papa,ivan.guilherme}@unesp.br`

Abstract

The drastic decrease in mobile SMS costs turned phone users more prone to spam messages, usually with unwanted marketing or questionable content. As such, researchers have proposed different methods for detecting SMS spam messages. This paper presents a technique for embedding SMS messages into vector spaces that is suitable for spam detection. The proposed approach relies on mining patterns that are relevant for distinguishing spam from legitimate messages. A subset of those patterns is used to construct a function that maps text messages into a multidimensional vector space. The extracted patterns are represented as skip-grams of token attributes, where a skip-gram can be seen as a generalization of the n-gram model that allows a distance greater than one between matched tokens in the text. We evaluate the proposed approach using the generated vectors for spam classification on the UCI Spam Collection dataset. The experiments showed that our method combined with shallow networks reached accuracy that is competitive with state-of-the-art approaches.

1 Introduction

SMS spam detection is a very relevant task for mobile phone users. It can mitigate the annoyance caused by the invasive marketing applied through this platform and provide a more secure user experience by blocking potentially harmful messages. In that sense, researchers have proposed different methodologies to detect SMS spam. Abdulhamid et al. (2017) and Abayomi-Alli et al. (2019), for instance, provide comprehensive reviews of the relevant datasets and techniques found in the literature.

Most recently, some researchers have successfully applied deep learning techniques for SMS spam detection: Annareddy and Tammina (2019) provide a comparative study on using Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN); Roy et al. (2020) and Popovac et al. (2018) propose to CNN as well; Roy et al. (2020), Jain et al. (2018) and Raj et al. (2018) propose using LSTM-based architectures; Ghourabi et al. (2020) use a hybrid CNN-LSTM model; Barushka and Hajek (2018) use regularized deep multi-layer perceptrons combined with a feature selection algorithm; and Wei and Nguyen (2020) propose the use of Lightweight Gated Recurrent Units (LGRU).

Other works employ traditional classifiers to such a task (Fernandes et al., 2015; Fattahi and Mejri, 2021; Xia and Chen, 2020), including diverse models like Support Vector Machines (SVM), Hidden Markov Models (HMM), Optimum-Path Forest (OPF), k-Nearest Neighbors (KNN), decision trees, and ensembling approaches. Gupta et al. (2018) provide a comparative study using CNN and traditional machine learning architectures.

SMS spam messages have a very characteristic textual style; many inform the recipient that she has won a prize or offer apparent great deals. As such, many words (e.g., “prize”, “won”, “free”, “bonus”, etc.) can be found to be very informative for classifying a message as spam. While these words can be used to create classifiers with good performance, we observe that words and patterns of a sequence of words can be very characteristic of spam messages. For example, in the UCI SMS Spam Collection dataset (Almeida et al., 2011), the probability that a message is spam given that it contains the word “`text`” is 0.53, and the same likelihood raises to 0.96 when the message includes the pattern “`text [WORDS] to`”, where “`[WORDS]`” is a placeholder for a short sequence of one or more words.

That happens because the word “`text`” can be used many times in legitimate messages, as illustrated by the following example: “*(...) I’ll **text** you*

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4193–4201
August 1–6, 2021. ©2021 Association for Computational Linguistics

in a few (...)". On the other hand, it is very common for spam messages to request the phone user to send a message to a certain number, like the following: "*(...)* ***text*** *TONE or FLAG* ***to*** *84199 NOW (...)*". Thus, observing this pattern gives much higher probability of a message being a spam than just finding the word "`text`" in it.

In this paper, we propose a text embedding technique for SMS messages that exploits patterns relevant for spam characterization. As a result, spam and non-spam messages are projected distinctly in the embedded space, allowing a more effective classification. The method is split into three main steps:

1. *Tokenization*: each text message from a training dataset is transformed into a sequence of sets of attributes;
2. *Skip-gram pattern mining*: the generated sequences are then processed by an algorithm for finding patterns relevant for the classification task;
3. *Embedding model*: given the set of relevant patterns, a model is constructed for mapping text messages into a vector space.

Our model presents relevant contributions, differing from some related works (especially those based on deep learning) regarding two aspects: (*i*) the proposed embedding approach is a lightweight model, being suitable to be used with shallow networks and executed on mobile devices; and (*ii*) the model allows interpretation, with explicit information associated with each dimension.

We evaluate the proposed technique using our method to generate vectors for the classification of SMS messages from the UCI SMS Spam Collection dataset (Almeida et al., 2011). Our best results surpassed the baselines from (Almeida et al., 2011) and showed to be competitive with recent deep learning approaches in terms of accuracy. In particular, our embeddings combined with a Linear SVM classifier achieved an average accuracy of 99.03% on 10-fold cross-validation experiments. Regarding efficiency aspects, prediction experiments using an ARM Cortex-A53 mobile processor yielded an average of 22 milliseconds of processing time for each SMS message.

The remainder of this paper is organized as follows: Section 2 formalizes our method and Section 3 performs an evaluation of the proposed technique. Finally, Section 4 states conclusions and presents some suggestions of future directions.

2 Formal Model Definition

This section presents a formalization of the proposed approach, which can be split into three main steps: (i) *tokenization*; (ii) *skip-gram pattern mining*; and (iii) the construction of the *embedding model*.

2.1 Tokenization

Given a set $\mathcal{X} = \{x_1, x_2, ..., x_{|\mathcal{X}|}\}$ of texts, the tokenization step constructs a set of sequences $\mathcal{S} = \{s_1, s_2, ..., s_{|\mathcal{X}|}\}$, where s_i is the sequence generated for the text x_i. We represent a given sequence as a tuple where each element is associated with a designated text fragment. Instead of merely being a substring of the text, each component of the tuple defines a set of attributes associated with the respective text fragment.

This transformation is applied with the following steps:

1. *Token splitting*: the text is split into fragments by capturing either of the following patterns, in that order of precedence: (*i*) sequence of one or more digits; (*ii*) sequence of one or more alphabetic characters; and (*iii*) sequence of any non-space character.

2. *Sub-word splitting*: each captured sequence of alphabetic characters is further broken into sub-sequences of length 3. For example, the word "`award`" is transformed into the following: "`awa`", "`war`", "`ard`". While 3 seems to be an arbitrary value for the length, we really want this parameter to be small, since that allows us to capture morphological variations of terms with similar semantics.

3. *Attribute assignment*: each fragment is transformed into a set of associated attributes, and the result of the tokenization process is the sequence of sets of attributes. Each attribute is represented as a pair (a, b), with the first element being the name of the attribute, and the second its associated value. We have three distinct attributes, i.e., $a \in \{$**`type`**, **`pos`**, **`sub`**$\}$, that can be defined as follows:

   - **`type`**: the type of the fragment. Possible values for b are `1-digit`, `2-digit`, `3-digit`, and `4+-digit` for fragments containing digits; `sub-word` for fragments that are sub-strings of alphabetic sequences; and `other` for fragments not belonging to any of the previous categories. The number before the digit types refers to the length of the captured string, `4+-digit` means 4 or more digits;

- **`pos`**: when the fragment is of type `sub-word`, b refers to the position of the sub-string in the captured word. Possible values for b are: `start`, `middle` and `end`;
- **`sub`**: when the fragment is of type `sub-word`, b is the sub-string represented by the fragment in lowercase. If the type is `other`, then b stands for the entire matched fragment.

Such a representation is coined here as an *attribute-set sequence*, and we use $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_{|\Sigma|}\}$ to denote the dictionary of attributes, that is, the set of all possible attributes.

Figure 1 illustrates the tokenization process for the string “`Call 09061701461.`” The boxes at the bottom represent the final sequence of attribute sets. For the sake of explanation, the first attribute of fragment “Cal” is the pair (**`type`**, `sub-word`).

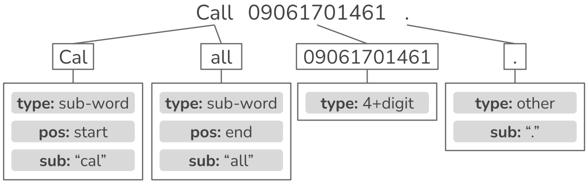

Figure 1: Illustration of the tokenization process for the string “`Call 09061701461.`”.

2.2 Skip-gram pattern mining

This step assumes that each sequence from $\mathcal{S}$ is assigned to a label from a set of labels $\mathcal{Y}$. We denote $y_i = \gamma(s_i)$ as the label assigned to the sequence s_i using model γ. The objective of this step is to construct a set $\mathcal{R}$ of skip-gram patterns that are relevant for label prediction (i.e., classification).

For this work, we formally define a k-skip-n-gram pattern as a tuple $g = (g_1, g_2, ..., g_n)$ whose elements are sets of attributes. We say that g matches an attribute-set sequence $a = (a_1, a_2, ..., a_l)$ if there is a tuple $z = (z_1, z_2, ..., z_n)$ of matched indices of a, such that:

- $\forall i, 1 \leq i < n, 1 \leq z_{i+1} - z_i \leq k + 1$, that is, matched indices must be in increasing order and the distance between two matched elements must not be greater than $k + 1$;
- $\forall i, 1 \leq i \leq n, g_i \subseteq a_{z_i}$.

Note that this definition differs from the one given by Guthrie et al. (2006) in the following aspects: (i) k is the number of skips allowed between consecutive elements of the skip-gram; (ii) a match is based on set containment ($g_i \subseteq a_{z_i}$) instead of equality.

Figure 2 illustrates how the skip-gram *({(pos, start), (sub, ”tex”),}, {(pos, start), (sub, ”to”)}, {(type, 4+-digit)})* would match the string “*text PLAY to 85222 now*”. That match holds for any $k \geq 3$.

The pattern mining algorithm interactively constructs a set of skip-gram patterns that might be relevant for the classification process. At each iteration t, a set of candidate patterns $\mathcal{C}^t$ is generated and a subset $\mathcal{R}^t \subseteq \mathcal{C}^t$ is selected based on a scoring function. After T iterations, the final set $\mathcal{R}$ of patterns is then selected.

Before detailing the algorithm, we provide the following definitions, which are used throughout the remainder of this formalization:

- $\mathcal{Z}^{(g,s)}$: the set of all tuples of indices of the attribute-set sequence $s \in \mathcal{S}$ that are matched by the skip-gram pattern g. Each tuple refers to a different match. As an example, in Figure 2, $(1, 5, 6)$ would be one tuple in such a set;
- $\mathcal{M}^{(g,s)} = \{g \in \mathcal{G} : \mathcal{Z}^{(g,s)} \neq \emptyset\}$: the set of skip-gram patterns in $\mathcal{G}$ that match the sequence s;
- $P(y|g) = P(\gamma(s) = y|\mathcal{Z}^{(g,s)} \neq \emptyset)$: the estimation of the probability that the label of a sequence s is y given that the pattern g matches s, which is calculated as follows:

$$P(y|g) = \frac{|\{s \in \mathcal{S} : \gamma(s) = y \wedge \mathcal{Z}^{(g,s)} \neq \emptyset\}|}{|\{s \in \mathcal{S} : \mathcal{Z}^{(g,s)} \neq \emptyset\}|}. \quad (1)$$

- $P(g|y) = P(\mathcal{Z}^{(g,s)} \neq \emptyset|\gamma(s) = y)$: the estimation of the probability that a sequence s is matched by the pattern g given that the label of s is y. The probability is given as follows:

$$P(g|y) = \frac{|\{s \in \mathcal{S} : \gamma(s) = y \wedge \mathcal{Z}^{(g,s)} \neq \emptyset\}|}{|\{s \in \mathcal{S} : \gamma(s) = y\}|}. \quad (2)$$

- $GY(g, y) = P(y|g)^{\texttt{GY\_ALPHA}} P(g|y)^{1-\texttt{GY\_ALPHA}}$, a function to measure the dependency level between a skip-gram g and label y, we call it GY-Score. The parameter `GY_ALPHA` is a value between 0 and 1 that allows one to give more relevance to one or the other term of the multiplication;

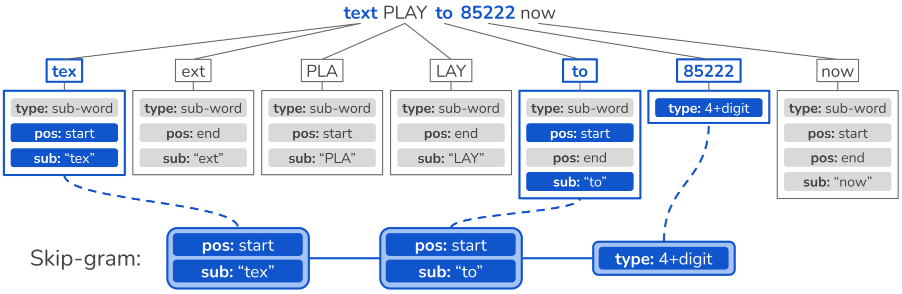

Figure 2: Example of a skip-gram matching the string "*text PLAY to 85222 now*".

- $\sup(g) = |\{s \in \mathcal{S} : \mathcal{Z}^{(g,s)} \neq \emptyset\}|$, the support of skip-gram pattern g, that is, the number of elements of $\mathcal{S}$ matched by g;

We now describe the mining algorithm in the following subsections. Sections 2.2.1 and 2.2.2 detail the two steps taken at each iteration of the algorithm and Section 2.2.3 describes the final pattern selection step.

2.2.1 Candidate set construction

At each iteration t, a set $\tilde{\mathcal{C}}^t$ of new candidate skip-gram patterns is constructed. In the first iteration ($t = 0$), this set is defined as

$$\tilde{\mathcal{C}}^0 = \{(\{\sigma\}) : \sigma \in \Sigma, \sup((\{\sigma\})) \geq \texttt{MIN\_SUP}\}, \tag{3}$$

that is, $\tilde{\mathcal{C}}^0$ is composed of all possible 1-skip-1-gram patterns that reach a minimum support parameter `MIN_SUP`.

From the second iteration and forward ($t > 0$), $\tilde{\mathcal{C}}^t$ is constructed by expanding patterns selected in the previous iteration. This is done by finding all matches between patterns selected in the previous iteration and sequences of $\mathcal{S}$, and then generating new patterns from the skip-gram respective to each match found.

For each match found, the generation of new patterns is done in two fashions: (*i*) adding new attributes found in the match either to the first or last attribute set of the pattern; and (*ii*) either appending or prepending singleton sets with attributes found around the match and, as such, increasing the size of the new pattern by 1.

All the newly generated patterns are filtered so that only those that reach the minimum support parameter `MIN_SUP` and that improve the GY-Score by at least `MIN_GY_DIFF` are kept in $\tilde{\mathcal{C}}^t$.

Formally, for $t > 0$, we can define:

$$\tilde{\mathcal{C}}^t = \bigcup_{s \in \mathcal{S}} \mathcal{E}^{(\mathcal{R}^{t-1}, s)}, \tag{4}$$

where $\mathcal{R}^{t-1}$ represents the set of patterns selected in the previous iteration and $\mathcal{E}^{(\mathcal{G},s)}$ can be described as the set of patterns generated as expansions of patterns in $\mathcal{G}$ for matches found in the sequence s, which can be defined as:

$$\begin{aligned}\mathcal{E}^{(\mathcal{G},s)} = \{g' : (g', g) \in &\bigcup_{a,b \in \{1,2\}} \mathcal{E}_{ab}^{(G,s)},\\ &GY(g', \gamma(s)) - GY(g, \gamma(s)) \geq \texttt{MGY},\\ &\sup(g') \geq \texttt{MIN\_SUP}\}.\end{aligned} \tag{5}$$

The sets given by each $\mathcal{E}_{ab}$ represent different types of expansions. Each element of such sets is a pair in the form (g', g), with g' being the newly generated skip-gram pattern and g, the original (from the match). The term `MGY` is an abbreviation for the `MIN_GY_DIFF` parameter.

Specifically, the sets represented by $\mathcal{E}_{11}^{(\mathcal{G},s)}$ and $\mathcal{E}_{12}^{(\mathcal{G},s)}$ are the expansions generated using the first method, by adding attributes to the attribute-sets on the extremes on the patterns:

$$\begin{aligned}\mathcal{E}_{11}^{(\mathcal{G},s)} = \{(g', g) : g' = &(g_1 \cup \{\sigma\}, ..., g_n),\\ &g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ &s = (a_1, ..., a_m),\\ &(j, ...) \in \mathcal{Z}^{(g,s)},\\ &\sigma \in a_j - g_1\}\end{aligned} \tag{6}$$

$$\begin{aligned}\mathcal{E}_{12}^{(\mathcal{G},s)} = \{(g', g) : g' = &(g_1, ..., g_n \cup \{\sigma\}),\\ &g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ &s = (a_1, ..., a_m),\\ &(..., j) \in \mathcal{Z}^{(g,s)},\\ &\sigma \in a_j - g_n\}\end{aligned} \tag{7}$$

On the other hand, $\mathcal{E}_{21}^{(\mathcal{G},s)}$ and $\mathcal{E}_{22}^{(\mathcal{G},s)}$ apply the second method of expansion, by prepending and appending elements to the skip-grams:

$$\begin{aligned}\mathcal{E}_{21}^{(G,s)} = \{(g', g) : g' = &(\{\sigma\}, g_1, ..., g_n),\\ &g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ &s = (a_1, ..., a_m),\\ &(j, ...) \in \mathcal{Z}^{(g,s)},\\ &\max\{1, j - k - 1\} \leq l < j,\\ &\sigma \in a_l\}\end{aligned} \tag{8}$$

$$\begin{aligned}\mathcal{E}_{22}^{(G,s)} = \{(g',g) : {} & g' = (g_1, ..., g_n, \{\sigma\}),\\ & g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ & s = (a_1, ..., a_m),\\ & (..., j) \in \mathcal{Z}^{(g,s)},\\ & j < l \leq \min\{m, j+k+1\},\\ & \sigma \in a_l\}\end{aligned} \tag{9}$$

, where k refers to the maximum number of skips for our the skip-gram model, which is referred by the parameter `SKIPGRAM_K` in the evaluation section.

The final set of candidate patterns $\mathcal{C}^t$ for the iteration t is then composed of the candidates from this and previous iterations that have not already been selected:

$$\mathcal{C}^t = \bigcup_{j=t-\texttt{CBS}+1}^{t} \tilde{\mathcal{C}}^j - \bigcup_{j=0}^{t-1} \mathcal{R}^j, \tag{10}$$

where `CBS` is an abbreviation for `CAND_BUFFER_SIZE`, which is a parameter that defines how many of the new candidate sets from previous iterations are remembered at the iteration t. This parameter gives candidates from the $\texttt{CBS} - 1$ previous iterations not yet selected a chance of being selected before they are "forgotten".

2.2.2 Pattern selection

Given the set of candidates $\mathcal{C}^t$, a set of new selected patterns $\mathcal{R}^t \subseteq \mathcal{C}^t$ is constructed by iterating over the sequences in $\mathcal{S}$ and selecting matching patterns with the best GY-Scores. Formally, the set $\mathcal{R}^t$ can be defined by

$$\mathcal{R}^t = \bigcup_{s \in \mathcal{S}} \text{Select}(\mathcal{C}^t, s), \tag{11}$$

where

$$\text{Select}(\mathcal{C}^t, s) = \underset{g \in \mathcal{M}^{(\mathcal{C}^t, s)}}{\arg\max}\, GY(g, \gamma(s)) \tag{12}$$

is either a set containing one of the patterns in $\mathcal{C}^t$ matching s with the highest GY-Score or the empty set if no matching pattern is found.

2.2.3 Final selection

After T iterations, a last step is performed in order to filter the selected patterns to contain only those that hit a minimum threshold value of information gain, given by a parameter `MIN_IG`, and then that filtered set is used to selected the best `FINAL_SELECT_K` matched patterns for each sequence. Thus, the final set $\mathcal{R}$ of selected skip-gram patterns is defined by:

$$\mathcal{R} = \bigcup_{s \in \mathcal{S}} \text{Best}(\mathcal{R}_F, s), \tag{13}$$

$$\mathcal{R}_F = \left\{ g \in \bigcup_{j=0}^{T-1} \mathcal{R}^j : IG(g) \leq \texttt{MIN\_IG} \right\}, \tag{14}$$

$$IG(g) = \sum_{y \in \mathcal{Y}} P(y|g) \cdot \ln(\frac{P(y|g)}{P(y)}), \tag{15}$$

where $\text{Best}(\mathcal{R}_F, s)$ is a set containing the top `FINAL_SELECT_K` elements from $\mathcal{M}^{(\mathcal{R}_F, s)}$ with the greatest GY-Scores; and $P(y)$ is the marginal probability that a sequence has label y and is estimated with:

$$P(y) = \frac{|\{s \in \mathcal{S} : \gamma(s) = y\}|}{|\mathcal{S}|}. \tag{16}$$

The number of iterations taken by the algorithm (i.e. T) is referred by the parameter `ITERATIONS` in the evaluation section.

2.3 Embedding Model

Once a set $\mathcal{R}$ of relevant skip-grams is found, it can then be used to build an embedding model. Our driving hypothesis is:

1. that the presence or absence of one or more of those patterns in a sequence greatly changes prior probability of labels;
2. that each pattern carries a very characteristic contextual baggage, in the sense that there is some level of statistical dependency between the occurrence of a pattern and of other attributes in a sequence.

As such, we propose an embedding function $v(s)$ that exploits both the information about the presence of patterns in the sequence s and the contextual information associated with each pattern taken into consideration.

We formalize our embedding model as follows. First we define a subset $\mathcal{R}_d$ of d skip-gram patterns from $\mathcal{R}$ to be used for the embedding. Then we define the function $v(s)$ in terms of elements of $\mathcal{R}_d$. The value of d is referred by the parameter `DIMENSIONS` in the evaluation section.

Let $\mathcal{R}_d \subseteq \mathcal{R}$ be a set containing the d most informative skip-gram patterns, that is: (*i*) $|\mathcal{R}_d| = d$, and (*ii*) $\forall r \in \mathcal{R}_d, \forall r' \in \mathcal{R} - \mathcal{R}_d, IG(r) \geq$

$IG(r')$. Each element of $\mathcal{R}_d$ will be associated with one dimension of the target vector space. For that, we select an arbitrary ordering of elements of $\mathcal{R}_d$, which we denote by the following indexing:

$$\mathcal{R}_d = \{r_1, r_2, ..., r_d\}. \tag{17}$$

Now, we define the value of $v(s)$ to be a d-dimensional vector as follows:

$$v(s) = \alpha \cdot u_1(s) + (1 - \alpha) \cdot u_2(s) \tag{18}$$

$$u_1(s) = [u_{11}(s), u_{12}(s), ..., u_{1d}(s)] \tag{19}$$

$$u_2(s) = [u_{21}(s), u_{22}(s), ..., u_{2d}(s)], \tag{20}$$

where:

- $u_1(s)$ is the vector accounting for skip-gram matches found in s. Each component $u_{1j}(s)$ is defined as:

$$u_{1j}(s) = \begin{cases} 1 & \text{if } r_j \in \mathcal{M}^{(\mathcal{R}_d, s)} \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

- $u_2(s)$ is the vector accounting for contextual information. Each component $u_{2j}(s)$ must be a measure that, in a way, reflects the statistical dependence between the pattern r_j and attributes found in s. For this work, we choose a very simple approach by taking the average of probabilities of r_j having a match with s given each relevant attribute of s. Thus, we define:

$$u_{2j}(s) = \frac{1}{|\mathcal{A}^s|} \sum_{\sigma \in \mathcal{A}^s} P(r_j | (\{\sigma\})), \tag{22}$$

  where $\mathcal{A}^s$ is the set of every attribute σ found in s in such that $\sup((\{\sigma\})) \leq$ `MIN_U2_SUP`, and `MIN_U2_SUP` is a parameter for filtering out rare attributes;

- $\alpha \in [0, 1]$ is a parameter for weighting the two types of information being aggregated in $v(s)$. We refer to this parameter by `U_ALPHA` in the evaluation section.

3 Evaluation

This section describes the evaluation of our proposed technique. We used the UCI SMS Spam Collection[1] (Almeida et al., 2011), which is a dataset comprised of 4,827 (86.6%) legitimate messages and 747 (13.4%) spam messages, with an average length of 16±12 tokens per message.

Parameter	Values used
Skip-gram mining parameters	
`SKIPGRAM_K`	6
`GY_ALPHA`	0.8
`MIN_SUP`	10
`MIN_GY_DIFF`	0.05
`CAND_BUFFER_SIZE`	6
`ITERATIONS`	6
`MIN_IG`	0.5
`FINAL_SELECT_K`	5
Embedding parameters	
`U_ALPHA`	0.0 0.1 ... 0.9 1.0
`MIN_U2_SUPPORT`	10 20 30
`DIMENSIONS`	50 100 189[2]

Table 1: Variations of parameters used for generating the embedding models.

We first conducted our evaluation by following the same experimental protocol used by the authors for their classification baselines and provide the analysis of our results in this in Sections 3.1 and 3.2. We used the train split to construct different variations of our embedding model and then evaluated three different machine learning methods for classification of spam and non-spam message. Table 1 displays the variation of parameters used for our experiments. The parameters related to the skip-gram mining step were fixed at single values after some empirical preliminary experimentation.

Later, we extended our evaluation by using different split configurations of the dataset in order to provide comparisons of our model with related work, which are presented in Section 3.2.1. Finally, Section 3.3 presents a visual evaluation of generated embeddings.

3.1 Selected patterns

The skip-gram pattern mining step applied to the training dataset resulted in 189 selected patterns. Table 2 displays the distribution of the types of patterns selected as well as the average information gain scores. We can note that longer patterns tend to be more informative but rarer at the same time. Although patterns with $n > 3$ were allowed (because `ITERATIONS` $=$ 6 and `CBS` $=$ 6), only patterns of length 1, 2 and 3 were selected.

Some interesting examples from the patterns with the highest information gain scores are listed

---

[1] Available at `https://archive.ics.uci.edu/ml/datasets/sms+spam+collection`.

[2] We use 189 instead of 200 here because that was the total number of selected skip-gram patterns after the mining process.

Type	Proportion (%)	Avg. Info. Gain
6-skip-1-gram	44.44	1.07±0.41
6-skip-2-gram	51.86	1.42±0.39
6-skip-3-gram	3.70	1.77±0.23

Table 2: Distribution of types of skip-gram patterns selected.

in Table 3. The third column displays the estimated conditional probability that a message is a spam given that the pattern matches it, and the fourth column shows the proportion of the spam messages that are covered by the respective pattern. The pattern `[{(sub, "cal")}, {(type, 4+-digit)}]` showed to be a very recurrent one, representing about 32% of the spam messages in the training dataset.

3.2 Classification

We used three different types of classifiers to assess the effectiveness of our embedding model: Nearest Neighbors Classifier, Multilayer Perceptron (MLP) and Linear Support Vector Machine (LSVM). For the three methods, we used the implementation provided by the Scikit-learn library (Pedregosa et al., 2011)[3].

Table 4 summarizes the best result achieved for each classification method. The best baseline found by (Almeida et al., 2011) is also shown in the table for comparison purposes. The table has the following columns:

- *Method*: name of the method used for classification;
- *SC*: percentage of spam caught;
- *BH*: percentage of blocked hams, that is, legitimate messages that were classified as spam;
- *Acc*: accuracy of the classification in percentage unit;
- *MCC*: the Matthews Correlation Coefficient (Matthews, 1975), which is a good metric to be used for unbalanced datasets (as it is the case with the dataset used in this work). The table is sorted by this column in descending order.

The MLP model was constructed using two layers with 100 neurons in each layer and the logistic function for activation, and it was optimized using the Adam optimizer (Kingma and Ba, 2014). Our preliminary experiments showed that using two layers yielded better results than using just one; and increasing further the number of layers hurt the performance of the model. Each embedding parameter variation was executed 10 times and the average of the metrics were taken.

For the KNN model, we varied the number of neighbors used by 5, 10 and 15 and tested both uniform and distance-based weighting schemes. We found that using 5 neighbors and distance-based weighting yielded the best result for this type of classifier.

We used the default parameters provided by Scikit-learn for the LSVM model. It is worth mentioning that the reported baseline also uses linear support vector machines.

When compared with the baseline, the results in Table 4 show that our embedding technique combined with those classification methods displayed significant increases in the rate of detected spams while keeping very low increases in the rate of blocked hams (less than 0.5%).

Table 5 shows the embedding parameters respective to the results in Table 4. The values are very similar between different methods. It is a consensus for all evaluated classification techniques that low values for `U_ALPHA` tend to yield the best results for classification tasks, which suggests that giving more weight to contextual information associated with skip-gram patterns is more beneficial.

3.2.1 Comparison with related work

We also compared our approach with recent deep learning techniques found in the recent literature. Since they used different experimental protocols, we re-executed our experiments using different splits of the dataset in order to match the different protocols used by related works. Table 6 displays the performance of our best models and related works grouped by the experimental protocol and Table 7 shows the best embedding parameters found for each protocol. The comparison displayed in Table 6 shows that our embedding technique, combined with a Linear SVM, is competitive with the deep learning approaches proposed recently.

3.3 Visualization

We finalize our evaluation by providing a visual analysis of the embeddings generated with the parameters from Table 5. For that, we used UMAP (McInnes et al., 2018a,b) to map each embedding generated for the test split into a 2-dimensional space and plotted the results. Figure 3

---

[3] We used version 0.24.0 of the library

g	$IG(g)$	$P(y = \text{spam} \mid g)$	$P(g \mid y = \text{spam})$
`[{(sub, "cal")}, {(type, 4+-digit)}]`	1.951	1.0	0.319
`[{(sub, "cla")}, {(sub, "aim")}]`	1.951	1.0	0.156
`[{(sub, "pri")}, {(sub, "ize")}]`	1.951	1.0	0.109
`[{(sub, "tex")}, {(sub, "to")}, {(type, 4+-digit)}]`	1.951	1.0	0.088

Table 3: Examples of selected patterns.

Method	SC(%)	BH(%)	Acc(%)	MCC
MLP	$90.12_{\pm 0.41}$	$0.61_{\pm 0.05}$	$98.19_{\pm 0.06}$	$0.919_{\pm 0.003}$
KNN	88.41	0.38	98.15	0.917
LSVM	87.62	0.29	98.13	0.916
Baseline	83.10	0.18	97.64	0.893

Table 4: Summary of results for classification.

Method	U_ALPHA	MIN_U2_SUPPORT	DIMENSIONS
MLP	0.3	30	189
KNN	0.1	30	189
LSVM	0.2	10	189

Table 5: Best embedding parameters for classification.

provides the visualization for the embeddings generated for the MLP classifier from Table 4. We can see that the projection successfully separates most of the vectors representing spams from the non-spam ones. Interestingly, some small clusters can be observed for both classes, which could indicate that, inside each class, further categorization could be possible.

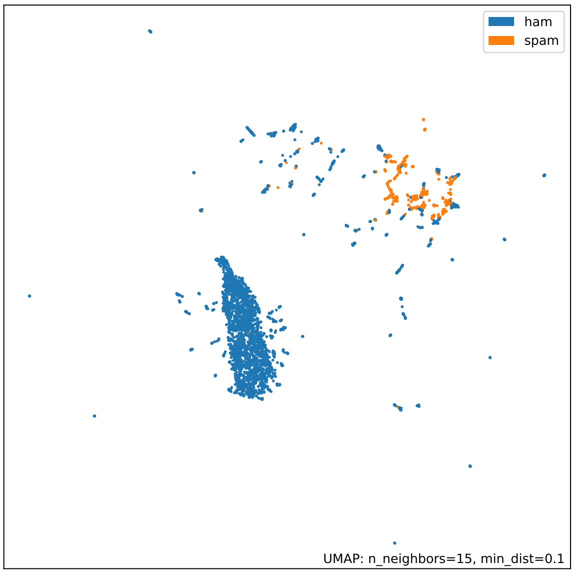

Figure 3: Visualization of embeddings for the MLP classifier.

4 Conclusions and Future Works

Reference	Classifier	Accuracy (%)
	10-fold cross-validation	
Roy et al. (2020)	CNN	99.44
This work	LSVM	$99.03_{\pm 0.31}$
Barushka and Hajek (2018)	DBB-RDNN-Rel	98.51
Popovac et al. (2018)	CNN	98.4
	Holdout 4:1	
This work	LSVM	99.1
Gupta et al. (2018)	CNN	99.1
Wei and Nguyen (2020)	LGRU	99.04
Annareddy and Tammina (2019)	RNN	97.8
	Holdout 3:1	
This work	LSVM	98.99
Raj et al. (2018)	LSTM	97.5
	Holdout 2:1	
This work	LSVM	98.76
Roy et al. (2020)	CNN	98.63

Table 6: Comparison with related work.

Protocol	U_ALPHA	MIN_U2_SUPPORT	DIMENSIONS
10-fold	0.3	10	300
Holdout 4:1	0.2	10	300
Holdout 3:1	0.2	10	300
Holdout 2:1	0.2	10	300

Table 7: Embedding parameters concerning the results found in Table 6.

In this work, we proposed an embedding technique for short texts based on skip-gram patterns and successfully evaluated our method in a dataset for SMS spam detection. Our experiments showed that our method, when combined with “shallow” classifiers, is competitive with recent deep learning approaches. Our results suggest that using this kind of pattern matching might be a good way to improve representations for textual classification tasks.

We see some interesting ways of improving the work presented in this paper, which can be taken as future work, such as:

- increasing our evaluation scope by adding more textual datasets (not limited to SMS) to our evaluation workload;
- investigating ways to integrate word vectors in our skip-gram matching mechanism in order to be able to capture patterns that do not necessarily have the same attributes but that are semantically similar;

- looking for alternative ways of defining the contextual component of the embedding function, namely, the function $u_2(s)$. In this work we chose to use a simple model that takes the average of conditional probabilities - more robust predictive models could be used instead.

Acknowledgments

The authors are grateful to the São Paulo Research Foundation - FAPESP (grants #2014/12236-1, #2018/15597-6, and #2019/07665-4), Brazilian National Council for Scientific and Technological Development - CNPq (grants #307066/2017-7 and #309439/2020-5), and Petrobras (grant #2019/00697-8).

References

Olusola Abayomi-Alli, Sanjay Misra, Adebayo Abayomi-Alli, and Modupe Odusami. 2019. A review of soft techniques for sms spam classification: Methods, approaches and applications. *Engineering Applications of Artificial Intelligence*, 86:197–212.

S. M. Abdulhamid, M. S. Abd Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan. 2017. A review on mobile sms spam filtering techniques. *IEEE Access*, 5:15650–15666.

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Sunil Annareddy and Srikanth Tammina. 2019. A comparative study of deep learning methods for spam detection. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. IEEE.

Aliaksandr Barushka and Petr Hajek. 2018. Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. *Applied Intelligence*, 48(10):3538–3556.

Jaouhar Fattahi and Mohamed Mejri. 2021. Spaml: a bimodal ensemble learning spam detector based on nlp techniques. In *5th International Conference on Cryptography, Security and Privacy*. IEEE.

D. Fernandes, K. A. P. Da Costa, T. A. Almeida, and J. P. Papa. 2015. Sms spam filtering through optimum-path forest-based classifiers. In *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 133–137.

Abdallah Ghourabi, Mahmood A. Mahmood, and Qusay M. Alzubi. 2020. A hybrid cnn-lstm model for sms spam detection in arabic and english messages. *Future Internet*, 12(9):156.

Mehul Gupta, Aditya Bakliwal, Shubhangi Agarwal, and Pulkit Mehndiratta. 2018. A comparative study of spam sms detection using machine learning classifiers. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *LREC*, volume 6, pages 1222–1225.

Gauri Jain, Manisha Sharma, and Basant Agarwal. 2018. Optimizing semantic lstm for spam detection. *International Journal of Information Technology*, 11(2):239–250.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

B.W. Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451.

Leland McInnes, John Healy, and James Melville. 2018a. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018b. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Milivoje Popovac, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, and Andras Anderla. 2018. Convolutional neural network based sms spam detection. In *2018 26th Telecommunications Forum (TELFOR)*. IEEE.

Hans Raj, Yao Weihong, Santosh Kumar Banbhrani, and Soomro Pir Dino. 2018. Lstm based short message service (sms) modeling for spam classification. In *International Conference on Machine Learning Technologies - ICMLT 18*. ACM Press.

Pradeep Kumar Roy, Jyoti Prakash Singh, and Snehasish Banerjee. 2020. Deep learning to filter sms spam. *Future Generation Computer Systems*, 102:524–533.

Feng Wei and Trang Nguyen. 2020. A lightweight deep neural model for sms spam detection. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE.

Tian Xia and Xuemin Chen. 2020. A discrete hidden markov model for sms spam detection. *Applied Sciences*, 10(14):5011.

Hierarchical Task Learning from Language Instructions with Unified Transformers and Self-Monitoring

Yichi Zhang **Joyce Y. Chai**
Computer Science and Engineering
University of Michigan
Ann Arbor, MI, USA
`{zhangyic, chaijy}@umich.edu`

Abstract

Despite recent progress, learning new tasks through language instructions remains an extremely challenging problem. On the ALFRED benchmark for task learning, the published state-of-the-art system only achieves a task success rate of less than 10% in an unseen environment, compared to the human performance of over 90%. To address this issue, this paper takes a closer look at task learning. In a departure from a widely applied end-to-end architecture, we decomposed task learning into three sub-problems: sub-goal planning, scene navigation, and object manipulation; and developed a model **HiTUT**[1] (stands for **Hi**erarchical **T**asks via **U**nified **T**ransformers) that addresses each sub-problem in a unified manner to learn a hierarchical task structure. On the ALFRED benchmark, HiTUT has achieved the best performance with a remarkably higher generalization ability. In the unseen environment, HiTUT achieves over 160% performance gain in success rate compared to the previous state of the art. The explicit representation of task structures also enables an in-depth understanding of the nature of the problem and the ability of the agent, which provides insight for future benchmark development and evaluation.

1 Introduction

As physical agents (e.g., robots) start to emerge as our assistants and partners, it has become increasingly important to empower these agents with an ability to learn new tasks by following human language instructions. Many benchmarks have been developed to study the agent's ability to follow natural language instructions in various domains including navigation (Anderson et al., 2018; Chen et al., 2019), object manipulation (Misra et al., 2017; Zhu et al., 2017) and embodied reasoning (Das et al., 2018a; Gordon et al., 2018). Despite recent progress, learning new tasks through language instructions remains an extremely challenging problem as it touches upon almost every aspect of AI from perception, reasoning, to planning and actions. For example, on the ALFRED benchmark for task learning (Shridhar et al., 2020), the state-of-the-art system only achieves less than 10% task success rate in an unseen environment (Singh et al., 2020), compared to the human performance of over 90%. Most previous works apply an end-to-end neural architecture (Shridhar et al., 2020; Singh et al., 2020; Storks et al., 2021) which attempt to map language instructions and visual inputs directly to actions. While striving to top the leader board for end task performance, these models are opaque, making it difficult to understand the nature of the problem and the ability of the agent.

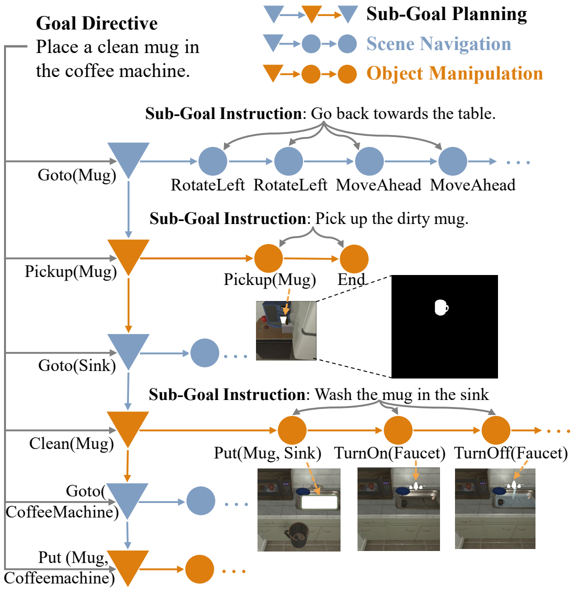

Figure 1: An example task in ALFRED.

To address this issue, this paper takes a closer look at task learning using the ALFRED benchmark. In a departure from an end-to-end ar-

---

[1] Source code available at `https://github.com/594zyc/HiTUT`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4202–4213
August 1–6, 2021. ©2021 Association for Computational Linguistics

chitecture, we have developed an approach to learn the hierarchical structure of task compositions from language instructions. As shown in Figure 1, a high-level goal directive ("place a clean mug in the coffee machine") can be decomposed to a sequence of sub-goals. Some subgoals involve navigation in space (e.g., `Goto(Mug)`, `Goto(Sink)`) and others require manipulation of objects (e.g., `Pickup(Mug)`, `Clean(Mug)`). These sub-goals can be further decomposed into navigation actions such as `RotateLeft` and `MoveAhead`, and manipulation actions such as `Put(Mug, Sink)`, `TurnOn(Faucet)`. In fact, such hierarchical structure is similar to Hierarchical Task Network (HTN) widely used in AI planning (Erol et al., 1994). While this hierarchical structure is explicit and has several advantages in planning and making models transparent, how to effectively learn such structure remains a key challenge.

Motivated by recent work in multi-task learning (Liu et al., 2019a), we decomposed task learning in ALFRED into three sub-problems: sub-goal planning, scene navigation, and object manipulation; and developed a model called **HiTUT** (stands for **Hi**erarchical **T**asks via **U**nified **T**ransformers) that addresses each sub-problem in a unified manner to learn a hierarchical task structure. On the ALFRED benchmark, HiTUT has achieved the best performance with a remarkably higher generalization ability. In the unseen environment, HiTUT achieves over 160% performance gain in success rate compared to the previous state of the art.

The contributions of this work lie in the following two aspects.

An explainable model achieving the new state-of-the-art performance. By explicitly modeling a hierarchical structure, our model offers explainability and allows the agent to monitor its own behaviors during task execution (e.g., what sub-goals are completed and what to accomplish next). When a failed attempt occurs, the agent can backtrack to previous sub-goals for alternative plans to execute. This ability of self-monitoring and backtracking offers flexibility to dynamically update sub-goal planning at the inference time to cope with exceptions and new situations. It has led to a significantly higher generalization ability in unseen environments.

A de-composable platform to support more in-depth evaluation and analysis. The decomposition of task learning into sub-problems not only makes it easier for an agent to learn, but also provides a tool for an in-depth analysis of task complexity and the agent's ability. For example, one of our observations from the ALFRED benchmark is that the agent's inability to navigate is a major bottleneck in task completion. Navigation actions are harder to learn than sub-goal planning and manipulation actions. For manipulation actions, the agent can learn action types and action arguments predominantly based on sub-goals and the history of actions, while language instructions do not contribute significantly to learning. The success of manipulation actions also largely depends on the agent's ability in detecting and grounding action arguments to corresponding objects in the environment. These findings allow a better understanding of the nature of the tasks in ALFRED and provide insight to address future opportunities and challenges in task learning.

2 Related Work

Recent years have seen an increasing amount of work on in the intersection of language, vision and robotics. One line of work particularly focuses on teaching robots new tasks through demonstration and instruction (Rybski et al., 2007; Mohseni-Kabir et al., 2018). Originated in the robotics community, learning from demonstration (LfD) (Thomaz and Cakmak, 2009; Argall et al., 2009) enables robots to learn a mapping from world states to robots' manipulations based on human's demonstration of desired robot behaviors. More recent work has also explored the use of natural language and dialogue together with demonstration to teach robots new actions (Mohan and Laird, 2014; Scheutz et al., 2017; Liu et al., 2016; She and Chai, 2017; Chai et al., 2018; Gluck and Laird, 2018).

To facilitate task learning from natural language instructions, several benchmarks using simulated physical environment have been made available (Anderson et al., 2018; Misra et al., 2018; Blukis et al., 2019; Shridhar et al., 2020). In particular, the vision and language navigation (VLN) benchmark (Anderson et al., 2018) has received a lot of attention. Many models have been developed, such as the Speaker-Follower model (Fried et al., 2018), the Self-Monitoring Navigation Agent(Ma et al., 2019a; Ke et al., 2019), the Regretful Agent (Ma et al., 2019b), and the environment drop-out model (Tan et al., 2019). The VLN benchmark is further extended to study the fidelity of instruction following (Jain et al., 2019) and examined to understand the bias of the benchmark (Zhang et al., 2020). Beyond navigation, there are also benchmarks that additionally incorporate object manipulation to broaden research on vision and language reasoning, such as embodied question answering (Das et al., 2018a; Gordon et al., 2018). The work closest to ours is the Neural Modular Control (NMC) (Das et al., 2018b), which also decomposes high-level tasks into sub-tasks and addresses each sub-task accordingly. However, self-monitoring and backtracking between sub-tasks is not explored in NMC.

The ALFRED benchmark consists of high-level goal directives such as *"place a clean mug in the coffee machine"* and low level language instructions such as *"rinse the mug in the sink"* and *"turn right and walk to the coffee machine"* to accomplish these goals. In addition to language instructions, it also comes with expert demonstrations of task execution in an interactive visual environment. We choose this dataset because its unique challenges are closer to the real world, which require the agent to not only learn to ground language to visual perception but also learn to plan for and execute actions for both navigation and object manipulation.

3 Hierarchical Tasks via Unified Transformers

As discussed in Section 1, task structures are inherently hierarchical, which compose of goals and sub-goals. Different sub-goals involve tasks of different nature. For example, navigation focuses on path planning and movement trajectories, while manipulation concerns more about interactions with concrete objects. Instead of end-to-end mapping from language instructions to primitive actions (Shridhar et al., 2020; Singh et al., 2020; Storks et al., 2021), we decomposed task learning into three separate but connected sub-problems: sub-goal planning, scene navigation, and object manipulation, and developed a model called **HiTUT** (stands for **Hi**erarchical **T**asks via **U**nified **T**ransformers) to tie these sub-problems together to form a hierarchical task structure.

3.1 Task Decomposition

We first introduce some notations to describe the task and the model. There are three types of information:

- Language ($\mathcal{L}$). We use G to denote a high-level goal directive, e.g., "place a clean mug in the coffee machine" and I_i to refer to a specific low-level language instruction.
- Vision ($\mathcal{V}$). It captures the visual representation of the environment.
- Predicates ($\mathcal{P}$). Symbolic representations are defined to capture three types of predicates: sub-goals (sg), navigation actions (a^n), and manipulation actions (a^m). Each sg has two parts (sg^{type}, sg^{arg}) where sg^{type} is the *type* (e.g., `Goto`) and sg^{arg} is the *argument* (e.g., `Knife`). Each a^n specifies a type (a^{n_type}) of action, from {`RotateLeft`, `RotateRight`, `MoveAhead`, `LookUp`, `LookDown`}. Each a^m has also two parts (a^{m_type}, a^{m_arg}) where a^{m_type} is the action type (e.g., `TurnOn`); a^{m_arg} is the action argument (e.g., `Faucet`).

Sub-Goal Planning. Sub-goal planning acquires a sequence of sub-goals $sg_1, \cdots, sg_n$ to accomplish the high-level goal G. We predict the type sg_i^{type} and argument sg_i^{arg} separately to avoid the combinatorial expansion of the output space. Previous work (Jansen, 2020) models sub-goal planning merely from high-level goal directives without visual grounding. These plans are fixed and thus not robust to potential failures during execution and variations of the visual environment. To overcome these drawbacks, our sub-goal planning is done on the fly after the previous sub-goal is executed in the environment. More specifically, our sub-goal planning objective is to learn a model (M_{sg}) that takes the visual observation at the current step (v_t), the high-level goal directive (G), and a complete sub-goal history prior to the current step ($sg_{<i}$) to predict the current sub-goal as follows:

$$sg_i \triangleq (sg_i^{type}, sg_i^{arg}) = M_{sg}(v_t, G, sg_{<i})$$

The predicted sub-goals serve as a bridge between the high-level goal and the low-level predictions of navigation actions and/or manipulation actions.

Scene Navigation. Navigation sub-goals only require predictions for the types of navigation actions. The objective is to learn a model for navigation (M_n) which takes the current visual observation (v_t), current sub-goal (sg_i), language instruction (I_i), and the navigation action history up to the current step ($a^n_{<j}$) to predict the next navigation action:

$$a_j^n \triangleq a_j^{n\_type} = M_n(v_t, I_i, sg_i, a^n_{<j})$$

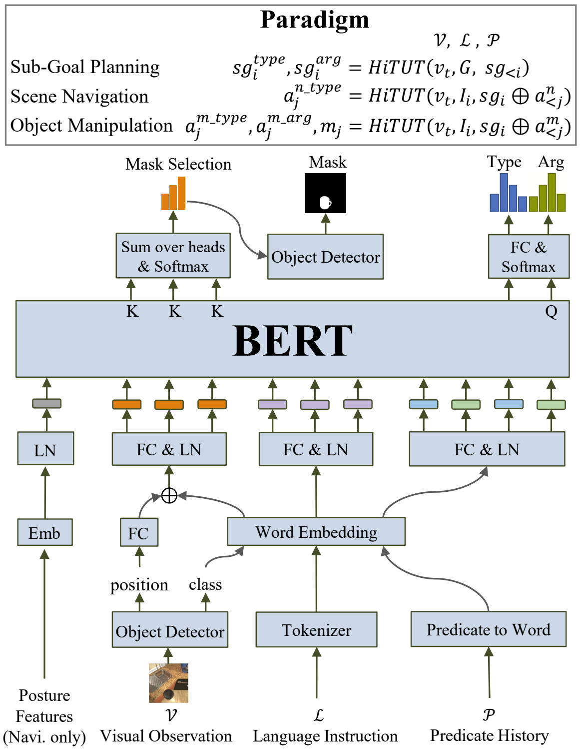

Figure 2: The structure of HiTUT.

Object Manipulation. For a manipulation sub-goal, in addition to the type and argument of the action, the model (M_m) also needs to generate a segmentation mask (m_j) on the current visual observation to indicate which object to interact with (i.e., which object the argument is grounded to):

$$\begin{aligned}(a_j^m, m_j) &\triangleq (a_j^{m\_type}, a_j^{m\_arg}, m_j)\\ &= M_m(v_t, I_i, sg_i, a_{<j}^m)\end{aligned}$$

The mask prediction is crucial because the action will not be successfully executed with an incorrect grounding even if a_j^m is correctly predicted.

As described above, although the context of the three sub-problems varies, each model has similar input components from the space of $\langle \mathcal{V}, \mathcal{L}, \mathcal{P} \rangle$. This similarity inspires us to design an unified model to solve three sub-problems simultaneously.

3.2 Unified Transformers

We leverage the effective self-attention based model (Vaswani et al., 2017) to capture the correspondence of different input sources as shown in Figure 2. We first project the input from different modalities into the language embedding space, and adopt a transformer to integrate the information together. Multiple prediction heads are constructed on top of the transformer encoder to make predictions for the sub-goal type and argument, the action type and argument, and object masks respectively. As the three sub-problems share the similar input form, we solve them all together using a unified model based on multi-task learning (Liu et al., 2019a).

Our model differs from previous works (Shridhar et al., 2020; Singh et al., 2020) in the following aspects. First, we do not apply recurrent state transitions, but feed the prediction history as the input to each subsequent prediction. This may help better capture correlations between predicates and other modalities. Second, we do not use dense visual features from the scene, but rather the object detection results. By doing this, we map different modalities to the word embedding space before feeding them into the transformer encoder, thus taking advantage of the pre-trained language models. Third, we use a predicate embedding to share linguistic knowledge between predicate symbols and word embeddings.

Predicate Embedding. We use the term predicates to refer to symbolic representations including sub-goal types, action types, and their arguments. We map symbols to their corresponding natural language phrases (e.g., `AppleSliced` is mapped to *a sliced apple*). We then tokenize and embed the tokens using word embeddings, and take the sum of the embeddings to obtain the representation of each predicate.

Vision Encoding. We use a pre-trained object detector (Mask R-CNN (He et al., 2017)) to encode visual information. Instead of dense features, we simply use the detection results (class labels, bounding box coordinates and confidence scores) as visual features. Specifically, we use the top K detected objects with a confidence score higher than 0.4 to form the visual features. The object class labels share the same space with object arguments, thus can be embedded into the same space. The position information of an object is encoded by a 7-dimensional vector consisting of its coordinates, width and height of the bounding box and its confidential score. This vector is first mapped to the same dimension as word embeddings by a liner transformation, then added to the class embedding to form the final object representation.

Object Grounding. HiTUT does not generate masks by itself. Instead it chooses an object from the K input objects and uses the corresponding mask generated by the object detector. This method makes use of the strong prior learned from object detection pre-training, so the model can focus on

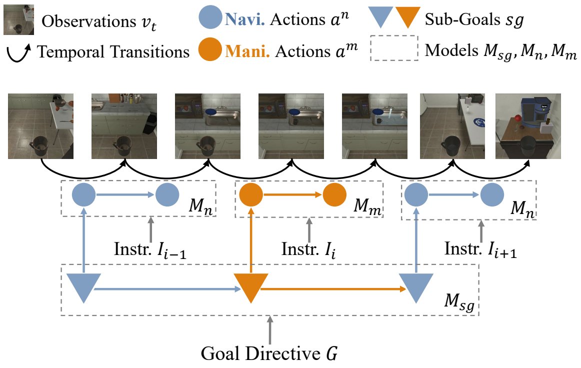

Figure 3: Overview of HiTUT where unified transformers for sub-programs are integrated together .

learning the grounding task. A drawback is that the object detector cannot be improved during training, and the performance of the detector determines the upper bound of our model's grounding ability. We leave the exploration of more robust grounding method for future work.

Posture Feature We use an additional posture feature to assist scene navigation, which includes the agent's rotation (N, S, E, W) and its angle of sight horizon (discretized by 15 degree). The positions are embedded and summed up to form the posture feature representation. The agent maintains its own posture in the form of a relative change to its initial posture instead of the absolute posture in the environment, thus avoid using additional sensory data.

3.3 Self-Monitoring and Backtracking

These unified transformers trained for sub-problems are integrated together as shown in Figure 3. One important advantage of intermediate sub-goal representations is to facilitate self-monitoring and backtracking which allows the agent to dynamically adjust the plan to cope with failures during execution. As shown in Section 4, this feature brings out the most remarkable performance gain compared to the state of the art.

Self-Monitoring. The world is full of uncertainties, and mistakes are inevitable. Based on the learned model, the agent should be able to monitor its own behaviors and dynamically update its plan when the situation arises. Our explicit representation of sub-goals allows the agent to self-check whether some sub-goals are accomplished. Particularly for manipulation sub-goals, it is feasible for the agent to detect their failures by simply monitoring whether all the manipulation actions are successfully executed. For example, `Clean(Mug)` cannot succeed if any of the actions along the path `Put(Mug, Sink)`, `TurnOn(Facuet)`, `TurnOff(Facuet)`, `Pickup(Mug)` fail. When the agent detects the failure of a subgoal, for example, as shown in Figure 4 the manipulation sub-goal `Pickup(Mug)` fails, it can reason about whether the previous sub-goal (i.e., `Goto(Mug)`) is successfully achieved.

	Train	Validation		Test	
		Seen	Unseen	Seen	Unseen
#Scenes	108	88	4	107	8
#Demonstrations	6,574	251	255	483	488
#Annotations	21,023	820	821	1,533	1,529
#Sub-goals	162k	6.4k	6.0k	-	-
#Navi. Actions	983k	39k	35k	-	-
#Mani. Actions	209k	8.3k	8.1k	-	-

Table 1: Statistics of data distribution in ALFRED. The number of annotations is equivalent to the number of tasks in each split.

Backtracking. In classical AI, backtracking is the technique to go back and try an alternative path that can potentially lead to the goal. As shown in Figure 4, when `Pickup(Mug)` fails, the agent backtracks to `Goto(Mug)` and tries a different sequence of primitive actions to accomplish this sub-goal. In ALFRED, only based on the visual information without other sensory information (e.g., only observing a mug without knowing how far it is), is it difficult to check whether a navigation sub-goal is successfully achieved (e.g. whether a `Mug` is reachable). So every time after trying a different path for `Goto(Mug)`, the agent will check whether the subsequent manipulation action `Pickup(Mug)` is successful. If it's successful, the agent will move on to the next sub-goal; otherwise the agent will continue to backtrack until a limit on the maximum number of attempts is reached. Our explicit representation of sub-goals makes this backtracking possible and has led to a significant performance gain in unseen environments.

4 Experiments

4.1 Setting and Implementation

Dataset. We follow the train/validation/tests data partition proposed in ALFRED, where validation and test sets are further split into *seen* and *unseen* based on whether the scene is shown to the model during training. Each sub-goal planning step or a primitive prediction step forms a data instance for

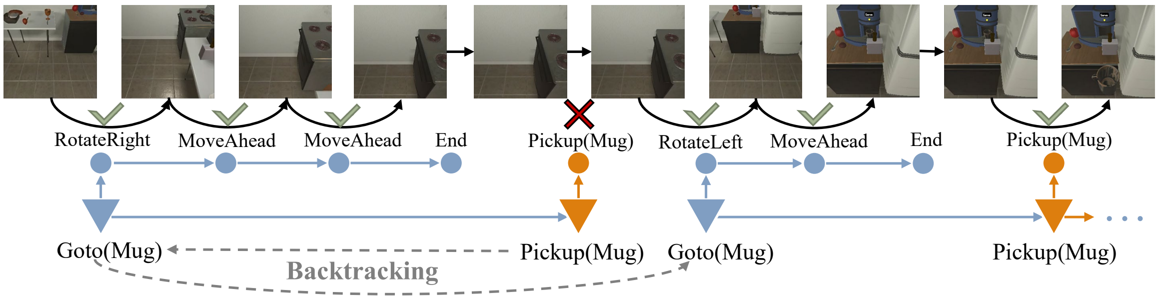

Figure 4: Illustration of self-monitoring and backtracking.

Model	Validation Seen		Validation Unseen		Test Seen		Test Unseen	
	Success	Goal-Cond	Success	Goal-Cond	Success	Goal-Cond	Success	Goal-Cond
Seq2Seq	3.70 (2.10)	10.00 (7.00)	0.00 (0.00)	6.90 (5.10)	3.98 (2.02)	9.42 (6.27)	0.39 (0.80)	7.03 (4.26)
HAM	-	-	-	-	12.40 (8.20)	20.68 (18.79)	4.50 (2.24)	12.34 (9.44)
MOCA	19.15 (**13.60**)	28.50 (**22.30**)	3.78 (2.00)	13.40 (8.30)	**22.05** (**15.10**)	28.29 (**22.05**)	5.30 (2.72)	14.28 (9.99)
HiTUT	**25.24** (12.20)	**34.85** (18.52)	**12.44** (**6.85**)	**23.71** (**11.98**)	21.27 (11.10)	**29.97** (17.41)	**13.87** (**5.86**)	**20.31** (**11.51**)
HiTUT (*G* only)	18.41 (7.59)	25.27 (12.55)	10.23(4.54)	20.71 (9.56)	13.63 (5.57)	21.11 (11.00)	11.12 (4.50)	17.89 (9.77)
Human	-	-	-	-	-	-	91.00 (85.80)	94.50 (87.60)

Table 2: Task and Goal-Condition success rates. The path length weighted version is in parentheses. The highest values per column are in **bold**. "-" denotes scores that are not reported. *G only* denotes only using the goal directive during evaluation without any sub-goal instructions.

the corresponding sub-problem. The number of data instances are shown in Table 1.

Pre-training. We employ the pre-training followed by fine-tuning paradigm for both the object detector and the main model. For the object detector, we use a Mask R-CNN (He et al., 2017) model pre-trained on MSCOCO (Lin et al., 2014), and fine-tune it on 50K images collected by replaying the expert trajectories in the ALFRED train split. As we observe that the model struggles on detecting small objects together with large receptacles, we train two networks to detect movable objects and big receptacles separately. We use the pre-trained RoBERTa (Liu et al., 2019b) model to initialize the transformer encoder.

Training. We perform imitation learning (supervised learning) on the expert demonstrations. The ground-truth labels of sub-goals and primitive actions are obtained from the metadata. Different input and output labels are organized for each sub-problem respectively as described in Section 3. We use the mask proposal that overlaps the most with the ground truth mask as the mask selection label if the intersection-of-union is above 50%. If there is no valid mask proposals, the label is assigned to 0 as an indicator of non-valid grounding. We optimize the cross-entropy loss between model predictions and the ground truth. We follow the multi-task training schema in Liu et al. (2019a) where for each iteration, a batch is randomly sampled among all the sub-problems, and the model is updated according to the corresponding objective. More details are in Appendix.

Evaluation Metrics. ALFRED leverages an interactive evaluation in the AI2-THOR environment (Kolve et al., 2017). A task is considered successful if all the goal conditions (e.g. the target object is placed on a correct receptacle and in a requested state such as heated or cleaned etc.) are met. Three measures are used: (1) success rate (the ratio of successfully completed tasks), (2) goal-condition rate (ratio of completed goal conditions), and (3) a weight version of these two rates which takes into account of the length difference between the predicted action sequence and the expert demonstrated action sequence (Shridhar et al., 2020).

Baselines. We compare HiTUT to: (1) Seq2Seq - an LSTM-based baseline model with progress monitoring proposed in Shridhar et al. (2020); (2) HAM - a hierarchical attention model over enriched visual inputs (Nguyen and Okatani, 2020), and (3) MOCA - a modular approach which also uses a Mask R-CNN for mask generation (Singh et al., 2020) and achieved previous state-of-the-art performance.

	Model	*Pick*	*Put*	*Cool*	*Heat*	*Clean*	*Slice*	*Toggle*	Avg.
Seen	Seq2Seq	32	**81**	88	85	81	25	**100**	70
	MOCA	53	62	87	84	79	51	93	73
	HiTUT	**81**	77	**95**	**100**	**83**	**81**	97	**88**
Unseen	Seq2Seq	21	46	92	89	57	12	32	50
	MOCA	44	39	38	86	71	55	11	49
	HiTUT	**71**	**69**	**100**	**97**	**91**	**78**	**58**	**81**

Table 3: Success rates of manipulation sub-goals on validation sets. The highest values per fold are in **bold**.

4.2 Evaluation Results

4.2.1 Overall Performance of HiTUT

We first evaluate the overall performance of the proposed framework as shown in Table 2. On the testing data reported by the leader board, in seen environments, HiTUT achieves comparable performance as MOCA. However in unseen environments, HiTUT outperforms MOCA by over 160% on success rate. This demonstrates our hierarchical task modeling approach has higher generalization ability compared to end-to-end models. Self-monitoring and backtracking enabled by hierarchical task structures allows the agent to better handle new situations. Remarkably, only based on high-level goal directives (i.e., *HiTUT (G Only)*) without using any sub-goal instructions, is HiTUT able to obtain a success rate of 11% in unseen environment, achieving 110% performance gain compared to MOCA. This result indicates that HiTUT can learn prior task knowledge from the hierarchical modeling process and apply that directly in new environment with some success. Nevertheless, our results are far from human performance and there is still huge room for future improvement.

To have a better understanding of the problem, we also conduct evaluations on sub-goals. The agent is positioned at the starting point of each sub-goal by following the expert demonstration and the success rate of accomplishing the sub-goal is measured. HiTUT predicts first a symbolic sub-goal representation and then the action sequence to complete the sub-goal. As shown in Table 3, HiTUT outperforms previous models on almost all of the manipulation sub-goals by a large margin. The performance gain is particularly significant in unseen environment, which demonstrates the advantage of our explicit hierarchical task modeling in low-level action planning.

	Valid Seen		Valid Unseen	
#BT	Success	Goal-Cond	Success	Goal-Cond
No	10.5 (6.0)	18.4 (13.8)	5.2 (3.0)	13.5 (11.1)
2	18.9 (9.9)	27.6 (18.0)	10.2 (5.9)	20.2 (13.6)
4	23.1 (11.3)	32.9 (18.6)	12.9 (7.0)	22.7 (12.9)
6	25.6 (12.0)	35.1 (18.5)	14.5 (7.4)	24.3 (12.3)
8	27.2 (12.5)	37.0 (18.5)	16.2 (7.8)	25.9 (12.1)

Table 4: Success rates w.r.t. the allowed maximum backtracking number (#BT).

	Seq2Seq	MOCA	Our model with different #backtracks					
			no	1	2	4	6	8
Seen	51	54	35	48	56	64	68	70
Unseen	22	32	31	45	53	60	63	65

Table 5: Success rate of the navigation sub-goal `Goto` with backtracking .

4.2.2 The Role of Backtracking

We conduct experiments to better understand the role of self-monitoring and backtracking. We repeat the task-solving evaluation with different limits on the allowed maximum number of backtracking. The agent only stops when the model predicts to stop (i.e., predicts `End`) or it reaches the backtracking limit. As shown in Table 4, as the limit increases, the task/goal-condition success rate increases accordingly. One thing notable is that the gap between success rates (weighted and unweighted) become larger when more backtrack attempts are allowed. This is within our expectation because backtracking deviates from instruction following navigation to goal-oriented exploration, which usually takes more steps than the expert demonstration.

Since backtracking is particularly targeted to navigation sub-goals `Goto` (see Section 3.3), we further examine the role of number of re-tries (i.e. backtracks) in completing the sub-goal. As shown in Table 5, HiTUT reaches more targets when given more opportunities to backtrack. The backtracking is most beneficial in unseen environment.

4.2.3 Complexity of Tasks

Task decomposition provides a tool to enable better understanding of task complexity and agent's ability. To do that, we replace different part of model predictions by the corresponding oracle sub-goals, actions, or masks, as shown in Table 6.

Using oracle sub-goals improves the success rate for 2%-6% (line SG), showing sub-goal planning is a relatively easy problem and the agent can perform reasonably well. After using the oracle

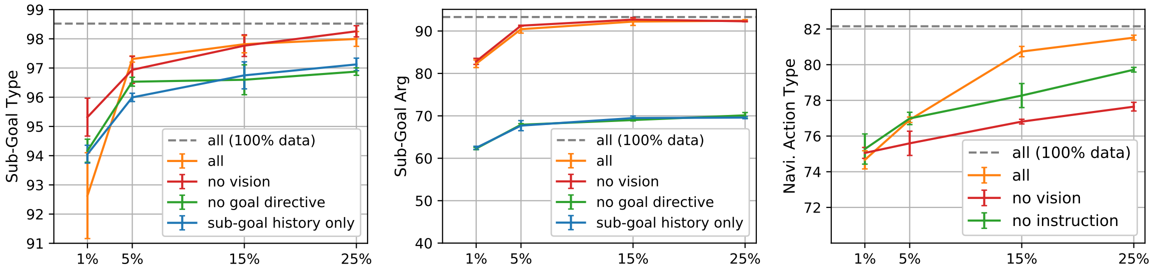

(a) Sub-Goal Type (b) Sub-Goal Argument (c) Navigation Action Type

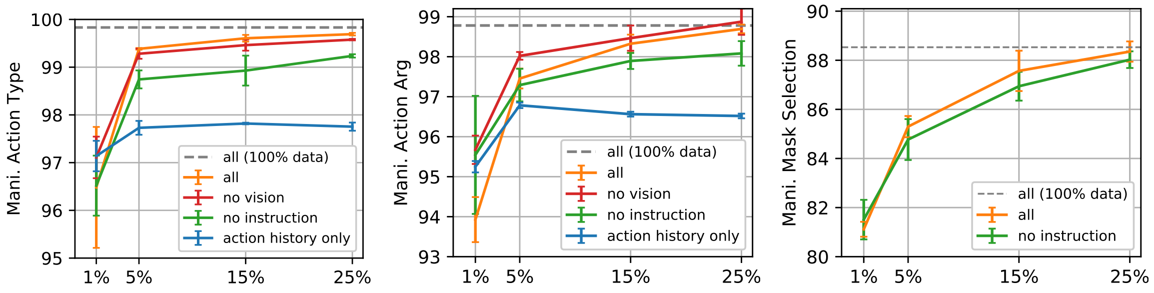

(d) Manipulation Action Type (e) Manipulation Action Argument (f) Manipulation Mask Selection

Figure 5: Step-by-step prediction accuracies given the golden sub-goal/action history w.r.t. the proportion of training data on the unseen validation set. Each solid line corresponds to a specific input configuration. Dashed lines are the scores obtained using 100% of training data.

Method	Valid Seen		Valid Unseen	
	Success	Goal-Cond	Success	Goal-Cond
HiTUT	25.2 (12.2)	34.8 (18.5)	12.4 (6.8)	23.7 (12.0)
+ Oracle				
SG	29.0 (15.6)	39.1 (21.3)	14.0 (7.6)	25.6 (12.7)
N	75.0 (72.7)	78.0 (77.4)	57.9 (60.0)	67.7 (65.2)
SG+N	79.2 (77.8)	84.0 (81.3)	64.2 (64.2)	72.0 (68.1)
SG+N+M	89.0 (100)	90.0 (100)	80.5 (100)	83.7 (100)
SG+N+GR	99.3 (99.0)	99.4 (99.1)	99.4 (99.3)	99.6 (99.6)

Table 6: Success rates of HiTUT with different parts of predictions replaced by oracle operations with expert demonstrations. N, M, SG and GR denote oracle navigation actions, manipulation actions, sub-goals and object grounding (i.e., mask generation) respectively.

navigation actions, the seen and unseen success rates are boosted by an absolute gain of 50% and 46% respectively (line N), indicating that navigating to reach target objects is a particularly hard problem and the agent performs poorly. When oracle sub-goals, navigation actions, and manipulation actions (only symbolic representations) are given (line SG+N+M), the task success is bounded by the performance of the pre-trained object mask generator (i.e., visual grounding of the object). When oracle object masks are given together with oracle sub-goals and navigation actions (line SG+N+GR) and the agent only needs to predict symbolic representation of manipulation actions, the performance is near perfect. These last two lines indicate that predicting the type and the argument of a manipulation action is a rather simple problem in the ALFRED benchmark while grounding action arguments to the visual environment remains a challenging task.

We further examine the complexity of learning to solve sub-problems by evaluating the next-step prediction accuracy given the golden history under different conditions as shown in Figure 5. The models are trained and evaluated with different combinations of input and different amount of training data. We observe that excluding the visual input does not hurt performance for sub-goal prediction and manipulation action prediction (shown by a,b,d,e). This indicates that in ALFRED, pure symbolic planning is often independent from visual understanding, which is consistent with the findings in (Shridhar et al., 2020). However, this could be an oversimplification brought by the bias in the dataset rather than a true reflection of the physical world. For example, next action prediction can be made by remembering the correlation of predicates instead of reasoning over vision and language, due to the lack of diversity of the task environments. Removing language instructions causes a minimal performance drop of 1%-2% on action prediction tasks, which brings up the question about the usefulness of language instructions in this benchmark. Furthermore, the prediction accuracy is above 90% and 98% with only 5% training data for sub-goal and manipulation planning respectively, while the navigation accuracy is only 82% given all the data. This again supports the finding that planning and performing navigation actions is a much harder problem than sub-goal planning and manipulation actions in ALFRED.

5 Discussion and Conclusion

This paper presents a hierarchical task learning approach that achieves the new state-of-the-art performance on the ALFRED benchmark. The task decomposition and explicit representation of subgoals enable a better understanding of the problem space as well as the current strengths and limitations. Our empirical results and analysis have shown several directions to pursue in the future. First, we need to develop more advanced component technologies integral to task learning, e.g., more advanced navigation modules through either more effective structures (Hong et al., 2020) or richer perceptions (Shen et al., 2019) to solve navigation bottleneck. We need to develop better representations and more robust and adaptive learning algorithms to support self-monitoring and backtracking. We also need to seek ways to improve visual grounding, which is crucial to both navigation and manipulation.

Second, we should also take a closer look at the construction and objective of existing benchmarks. How a benchmark is created and how truthfully it reflects the complexity of the physical world would impact the scalability and reliability of the approach in the real world. As for the objective, there is a distinction between *learning to perform tasks* and *learning to follow language instructions*. If the objective is the former, the agent should be measured by the ability to learn to accomplish high-level goal directives without being given specific language instructions at the inference time. If the objective is the latter, then the agent should be measured by how faithful it follows human instructions aside from achieving the goals, similar to (Jain et al., 2019). We need to be clear about the objectives and develop evaluation metrics accordingly.

Finally, when humans perform poorly in a complex task, we have the ability to diagnose the problem and put more energy on learning the difficult part. Physical agents should also have similar abilities. In task learning, on the one hand, the agent should be able to master simple sub-tasks from a few data instances, e.g., through a few turns of interactions with humans (Karamcheti et al., 2020). On the other hand, it should be aware of the bottleneck of its learning progress and proactively request for help when problems are encountered either during learning or during deployment (She and Chai, 2017). How to effectively design interactive and active learning algorithms for the agent to learn complex and compositional tasks remains an important open research question.

Acknowledgments

This work is supported by the National Science Foundation (IIS-1949634). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3674–3683. IEEE Computer Society.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A. Knepper, and Yoav Artzi. 2019. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*.

Joyce Y. Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. 2018. Language to action: Towards interactive task learning with physical agents. In *IJCAI*, pages 2–9.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. TOUCHDOWN: natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12538–12547. Computer Vision Foundation / IEEE.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018a. Embodied question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1–10. IEEE Computer Society.

Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018b. Neural modular control for embodied question answering. In *Conference on Robot Learning*, pages 53–62. PMLR.

Kutluhan Erol, James Hendler, and Dana S Nau. 1994. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3318–3329.

Kevin A Gluck and John E Laird. 2018. *Interactive task learning: Humans, robots, and agents acquiring new tasks through natural interactions*. The MIT Press.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: visual question answering in interactive environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4089–4098. IEEE Computer Society.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988. IEEE Computer Society.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez Opazo, and Stephen Gould. 2020. A recurrent vision-and-language bert for navigation. *arXiv: Computer Vision and Pattern Recognition*.

Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1862–1872, Florence, Italy. Association for Computational Linguistics.

Peter Jansen. 2020. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4412–4417, Online. Association for Computational Linguistics.

Siddharth Karamcheti, Dorsa Sadigh, and Percy Liang. 2020. Learning adaptive language interfaces through decomposition. In *Proceedings of the First Workshop on Interactive and Executable Semantic Parsing*, pages 23–33, Online. Association for Computational Linguistics.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. 2019. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6741–6749.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-Chun Zhu, and Joyce Chai. 2016. Jointly learning grounded task structures from language instruction and visual demonstration. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1492, Austin, Texas. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6732–6740. Computer Vision Foundation / IEEE.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015, Copenhagen, Denmark. Association for Computational Linguistics.

Dipendra Kumar Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678.

Shiwali Mohan and John E. Laird. 2014. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 387–394. AAAI Press.

A. Mohseni-Kabir, C. Li, V. Wu, D. Miller, B. Hylak, S. Chernova, D. Berenson, C. Sidner, and C. Rich. 2018. Simultaneous learning of hierarchy and primitives (slhap) for complex robot tasks. *Autonomous Robotics*.

Van-Quang Nguyen and Takayuki Okatani. 2020. A hierarchical attention model for action learning from realistic environments and directives. *ECCV EVAL Workshop*.

P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso. 2007. Interactive robot task training through dialog and demonstration. In *The 2nd ACM/IEEE International Conference onHuman-Robot Interaction (HRI)*, pages 49–56.

M. Scheutz, E. Krause, B. Oosterveld, T. Frasca, and R. Platt. 2017. Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1378–1386.

Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1634–1644, Vancouver, Canada. Association for Computational Linguistics.

William B. Shen, Danfei Xu, Yuke Zhu, Fei-Fei Li, Leonidas J. Guibas, and Silvio Savarese. 2019. Situational fusion of visual representation for visual navigation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 2881–2890. IEEE.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. IEEE.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. 2020. MOCA: A modular object-centric approach for interactive instruction following. *arXiv preprint arXiv:2012.03208*.

Shane Storks, Qiaozi Gao, Govind Thattai, and Gokhan Tur. 2021. Are we there yet? learning to localize in embodied instruction following. *arXiv preprint arXiv:2101.03431*.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, Minneapolis, Minnesota. Association for Computational Linguistics.

A. L. Thomaz and M. Cakmak. 2009. Learning about objects with human teachers. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, HRI '09, pages 15–22, New York, NY, USA. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yubo Zhang, Hao Tan, and Mohit Bansal. 2020. Diagnosing the environment bias in vision-and-language navigation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 890–897. ijcai.org.

Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. 2017. Visual semantic planning using deep successor representations. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 483–492. IEEE Computer Society.

Appendix

A Additional Training Details

We use the RoBERTa (Liu et al., 2019b) implementation from Huggingface (Wolf et al., 2019). The model is fine-tuned for 10 epochs with the Adam (Kingma and Ba, 2015) optimizer on the ALFRED training set. The learning rate warms up over the first half of the first epoch to a peak value of 1e-5, and then linearly decayed. The model achieving the highest navigation action prediction accuracy on the validation seen set is selected for evaluation. All the models are trained on one NVIDIA V100 16GB GPU.

B Additional Evaluation Details

We follow the evaluation setting in the ALFRED benchmark[2]. For each episode, the agent is given a task, which is composed of a goal directive G and several sub-goal instructions. The agent needs to sequentially perform actions to achieve the goal based on the visual observations of RGB image only. This progress ends if the agent predicts an *End* action (an *End* sub-goal for HiTUT), which is made after up to 10 failed interaction attempts or reaching the maximum step limitation. For HiTUT, there is also a maximum number of backtracking attempts, and the model will be forced to stop if the budget runs out. The maximum number of backtracking is 8 in all of our experiments without explicitly mentioning the backtracking number. We also leverage two techniques to reduce the interaction attempt failures. We use the obstruction detection trick proposed in Singh et al. (2020) to avoid failures caused by repeated tries of moving toward obstructions. We propose a self-monitoring approach to check the validity of manipulation actions. If no mask is selected or a predicted action argument is not consistent with the class prediction from Mask R-CNN for the selected object, the manipulation action is decided as failed and the agent performs a backtrack without trying to execute the action. Note that in Table 4, we remove the interaction attempt constraint when comparing the effect of different allowed maximum backtracking numbers, thus the results of $\#BT = 8$ is slightly higher than those shown in Table 2.

Task-Type	MOCA		HiTUT	
	Seen	Unseen	Seen	Unseen
Pick & Place	29.5	5.0	**35.9**	**26.0**
Cool & Place	**26.1**	0.7	19.0	**4.6**
Stack & Place	5.2	1.8	**12.2**	**7.3**
Heat & Place	**15.8**	2.7	14.0	**11.9**
Clean & Place	22.3	2.4	**50.0**	**21.2**
Examine & Place	20.2	**13.2**	**26.6**	8.1
Pick Two & Place	11.2	1.1	**17.7**	**12.4**
Average	18.6	3.8	**25.2**	**12.4**

Table 7: Success rates across 7 task types on the validation sets. Highest values per fold are **bold**.

Model	#Backtracking	Seen SR	Unseen SR
RoBERTa	no	10.5	5.2
Scratch	no	7.9	2.8
RoBERTa	4	23.1	12.9
Scratch	4	18.1	10.2
RoBERTa	8	27.2	16.2
Scratch	8	26.8	14.0
MOCA	-	19.15	3.78

Table 8: The validation success rates for models pretrained and trained from scratch with different allowed maximum number of backtrackings.

C Additional Results

A detailed per-task performance comparison of HiTUT and MOCA is shown in Table 7. As the comparison might be unfair since HiTUT benefits from model pre-training, we also conduct an ablation study to show the effectiveness of pretraining. In Table 8, we compare the fine-tuned RoBERTa model to a Transformer with the same size trained from scratch to show the role of the RoBERTa pretraining.We can see that RoBERTa consistently improves the performance over training from scratch both w/o and w/ backtracking with an absolute gain between 0.4% and 5% on task success rate. Notably, Scratch with 4 or 8 backtrackings still outperform MOCA by a large margin in terms of the unseen success rate.

---

[2] `https://leaderboard.allenai.org/alfred/submissions/get-started`

Marked Attribute Bias in Natural Language Inference

Hillary Dawkins
University of Guelph, Ontario, Canada
Vector Institute, Toronto, Ontario, Canada
`hdawkins@uoguelph.ca`

Abstract

Reporting and providing test sets for harmful bias in NLP applications is essential for building a robust understanding of the current problem. We present a new observation of gender bias in a downstream NLP application: marked attribute bias in natural language inference. Bias in downstream applications can stem from training data, word embeddings, or be amplified by the model in use. However, focusing on biased word embeddings is potentially the most impactful first step due to their universal nature. Here we seek to understand how the intrinsic properties of word embeddings contribute to this observed marked attribute effect, and whether current post-processing methods address the bias successfully. An investigation of the current debiasing landscape reveals two open problems: none of the current debiased embeddings mitigate the marked attribute error, and none of the intrinsic bias measures are predictive of the marked attribute effect. By noticing that a new type of intrinsic bias measure correlates meaningfully with the marked attribute effect, we propose a new post-processing debiasing scheme for static word embeddings. The proposed method applied to existing embeddings achieves new best results on the marked attribute bias test set. See https://github.com/hillary-dawkins/MAB.

1 Introduction

Pre-trained distributed representations of words (a.k.a. word embeddings) are ubiquitous tools in natural language processing (NLP). Their utility is owing to the remarkable success in mapping semantic and syntactic relationships among words to linear relationships among real-valued vectors. For instance, analogy generation using vector addition on word embeddings (e.g. Tokyo is to Japan as Paris is to France) was taken to be an early measure of word embedding quality. In all kinds of related tasks, the vector space is known to encode semantic meaning surprisingly well (Pennington et al., 2014; Mikolov et al., 2013b,c). However, harmful gender-biased properties of word embeddings are also known to exist. Later is was observed that the same analogy generation property that produced the celebrated "man is to king as woman is to queen" analogy would also predict "man is to programmer as woman is to homemaker" (Bolukbasi et al., 2016). This observation sparked interest in developing debiased word embeddings.

Post-processing debiasing schemes are usually motivated by recognizing some intrinsic measure of bias in the embedding space, and then attempting to reduce that intrinsic bias. Early work (2016-2017) focused on the idea of a "gender direction" vector within the embedding space, loosely defined as the difference vector between female and male attribute words. It was noted that any non-zero projection of a word onto the gender direction (termed direct bias) implied that the word was more related to one gender over another. In the case of ideally gender-neutral words (e.g. doctor, nurse, programmer, homemaker), this was viewed as an undesirable property. The first debiasing methods, Hard Debias (Bolukbasi et al., 2016) and Gender Neutral-GloVe (Zhao et al., 2018b), worked to minimize or eliminate the direct bias, and were shown to be successful in mitigating harmful analogies generated by word embeddings in relation to gender-stereotyped occupations.

An influential critique paper by Gonen and Goldberg (2019) demonstrated that minimizing direct bias did not eliminate bias in the vector space entirely. Rather, words that tended to cluster together due to gender bias (e.g. nurse, teacher, secretary, etc.) would still cluster together in the nullspace of the gender direction. Furthermore, the original bias could be recovered by classification techniques using only the debiased word embeddings as input.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4214–4226
August 1–6, 2021.

These observations were termed cluster and recoverability bias.

The next wave of debiasing methods (2019-present) focused on reducing cluster and recoverability bias while proposing new metrics to systematically quantify the indirect bias of the embedding space (e.g. the Gender-based Illicit Proximity Estimate, introduced by Kumar et al. (2020)). While these new debiasing schemes do reduce indirect bias in multiple ways, there is a general lack of connection to downstream applications such as coreference resolution, natural language inference (NLI) and sentiment analysis.

Current gender-bias evaluation tests (GBETs) in widespread use include the WinoBias test set (Zhao et al., 2018a), designed to measure bias in coreference resolution systems using stereotypical occupations as a probe, and the NLI test set (Dev et al., 2020a), designed to measure stereotypical inferences again using occupations as the concept of interest. More commonly used evaluations include the Word Embedding Association Test (WEAT) (Caliskan et al., 2017), and the analogy generation test SemBias (Zhao et al., 2018b). However these tests solely evaluate the vector properties of the word embeddings, without any connection to downstream applications. Adding to the library of downstream GBETs is essential in building a robust understanding of gender bias in NLP applications (Sun et al., 2019).

Here we introduce a new observation of gender-biased predictions in a downstream task, namely "marked attribute bias" in natural language inference, and develop corresponding GBETs. Marked attribute bias refers to the language model's tendency to predict that "person" implies "man" (the default attribute), while simultaneously understanding that "person" does not necessarily imply "woman" (the marked attribute). Marked attribute bias was found to exist on explicitly defined gender words (e.g. man, woman, etc.), and persist on implicit gender words (e.g. names) as well as latent gender-carriers (e.g. stereotypical occupations).

An analysis of the currently available debiased embeddings reveals that none are able to successfully mitigate marked attribute bias. Furthermore, none of the currently proposed measures of intrinsic bias on the embedding space are predictive of the marked attribute effect. We define a new measure of intrinsic bias that was found to correlate with the marked attribute effect better than any currently available metric. Using this insight, we introduce a new debiasing scheme: Multi-dimensional Information-weighted Soft Projection. Applying MISP to an existing debiased embedding achieves the lowest observed marked attribute bias error.

Summary of main contributions:

1. We present a new observation of gender bias in a downstream NLP application: marked attribute bias (MAB). The MAB test sets are made available in order to expand the current set of GBETs.

2. An analysis of current debiasing methods and current intrinsic bias measures finds that none sufficiently mitigate the error, and likewise none sufficiently explain the effect. This observation creates two new open problems.

3. We propose a new measure for quantifying intrinsic bias on the embedding space: Multi-dimensional Information-weighted Direct Bias (MIDB). This measure was found to correlate meaningfully with the marked attribute effect.

4. We introduce a new debiasing scheme: Multi-dimensional Information-weighted Soft Projection. MISP-debiased embeddings obtain new best performance on the MAB test set.

2 Marked Attribute Bias in Natural Language Inference

2.1 Background: Natural Language Inference

Natural language inference is one of the pillars of natural language understanding. It is the task of determining whether a hypothesis sentence is (neutral, entailed, or contradicted) with respect to a premise sentence. For example:

Premise: A choir sings in the church.

Hypothesis: The church is filled with the sound of singing. (Correct prediction: Entail)

Dev et al. (2020a) previously used NLI as a test case for gender bias with respect to occupations. For example, consider:

Premise: A doctor prepared a meal.

Hypothesis 1: A man prepared a meal. (N)

Hypothesis 2: A woman prepared a meal. (N)

This inference task essentially asks the question: is "doctor" a subset of man/woman? I.e. if someone is a doctor, must they be a man? While both hypothesis sentences should receive a neutral prediction (as "doctor" does not imply any specific gender), hypothesis 1 will more likely receive an entailment, while hypothesis 2 will more likely receive a contradiction, given biased word embeddings. The corresponding GBET was published by Dev et al. (2020a) and contains 1936512 sentence pairs in the form [A **occupation** *verb object*] → [A **gender word** *verb object*]. Throughout this paper, we will use the notation [Sentence A] → [Sentence B] to mean that premise Sentence A is paired with hypothesis Sentence B.

2.2 Observation of marked vs. default attribute bias

Marked vs. default attribute bias occurs whenever a default attribute (e.g. male, white, etc.) is assumed, and a marked attribute has to be explicitly stated or becomes a defining trait. In the context of the natural language inference task, consider the sentence pair:

Premise: A person prepared a meal.
Hypothesis 1: He prepared a meal. (N)
Hypothesis 2: She prepared a meal. (N)

Due to the language model's[1] tendency to predict that "person" implies a male (default) attribute, the first hypothesis sentence will have a prediction probability vector shifted towards Entail. However the same language model would tend towards a Neutral prediction for the second hypothesis, recognizing that "person" does not necessarily imply female (the marked attribute). To put it another way, this inference task essentially asks the question: is "person" a subset of man/woman? When presented with a masculine form, the model answers: yes (entailment), a person must be a man. When presented with a feminine form, the model answers: not necessarily (neutral), a female has an attribute (gender) that not all persons have. The name "Marked Attribute Bias" therefore derives from the observation that masculine forms are unmarked with respect to gender, whereas female forms carry a marked gender attribute.

In this particular example, the model trained with (original) GloVe[2] word embeddings (Pennington et al., 2014) gives a probability distribution (N, E, C) of (0.0538, **0.929**, 0.0177) for hypothesis 1 and (**0.687**, 0.238, 0.0750) for hypothesis 2.

Note that although the MAB test construction appears similar to Dev et al. (2020a), it is actually measuring quite a distinct effect. The (Dev et al., 2020a) test set measures associations between gender and some concept of interest (occupations). The MAB test set measures something more general and pervasive; it measures how gender words carry meaning, independent of any concept of interest.

Achieving the correct prediction probability of $(N, E, C) = (1, 0, 0)$ on both sentences is difficult because it requires the language model to be attribute-aware (in this case gender-aware) while not using the gender attribute to alter predictions when it would be inappropriate to do so.

3 Analysis of the current situation

In order to investigate the presence of systematic marked attribute bias in natural language inference, we construct three types of tests: bias on explicit gender words, implicit gender carriers, and latent gender carriers. We wish to understand the depth and persistence of the marked attribute effect, as well as how it is handled by current debiasing methods. Firstly we provide a brief description of the current debiasing methods to be analyzed. Next we provide details of the test sets and report results.

3.1 Debiased embeddings

Within the scope of this paper, we focus on post-processing techniques applied to static word embeddings. These types of methods are computationally inexpensive, easy to concatenate, and are independent of the base embedding. In addition, we include GN-GloVe, one of the highly cited retraining methods. Notationally, we specify embeddings as (base embedding).method. Where available, we use published debiased embeddings made available from the original authors of the corresponding method. Otherwise, we apply the method to the

---

[1] All NLI models mentioned throughout this paper are based on the Decomposable Attention Model (Parikh et al., 2016) with intra-attention, trained on the Stanford Natural Language Inference training dataset (Bowman et al., 2015) (trained for 100 epochs; learning rate 0.025; weight decay 1e-5; dropout rate 0.2; 200 hidden units; approximately 10^4 total model parameters). All the code and data needed to reproduce results mentioned in this paper are available at https://github.com/hillary-dawkins/MAB.

[2] Taken as the GloVe embeddings trained on the Common Crawl corpus for 840B tokens; available at https://nlp.stanford.edu/projects/glove/. Results were not found to vary significantly among undebiased embeddings.

base GloVe embeddings. The methods we will analyze include:

Hard Debias[3] **(GloVe**[4]**.HD)** (Bolukbasi et al., 2016): The subset of gender-neutral words are projected onto the nullspace of the gender direction $\vec{g}$. Gender-neutral words are made equidistant to pairs of words in a defined equalization set.

Gender-Neutral GloVe[5] **(GN-GloVe)** (Zhao et al., 2018b): Similar to hard debias, this method seeks to eliminate the direct bias. The embeddings are retrained from scratch using a modified version of GloVe's original objective function. The gender information is sequestered to the final component of the word embedding. The gender-neutral portion of the word embedding is then defined as the first $d-1 = 299$ components, denoted **GN-GloVe**$(\boldsymbol{w_a})$.

Gender-Preserving[6] **(GloVe**[4]**.GP)** (Kaneko and Bollegala, 2019): This method seeks to eliminate harmful gender bias while retaining as much useful semantic gender information as possible.

Double Hard Debias[7] **(GloVe**[4]**.DHD)** (Wang et al., 2020): An extended version of the hard debias algorithm, based on the observation that frequency information encoded in the word embeddings convolutes the definition of the gender direction. Correctional pre-processing is applied prior to hard debiasing.

Bias Alignment Model[8] **(GloVe**[4]**.BAM)** (Lauscher et al., 2019): Gender subspace matrices are defined by stacking explicit gender words. The projection that maps the embedding space to itself while approximately aligning the gender subspaces is learned and applied to all words. After alignment, gender information is not retained.

Orthogonal Subspace Correction and Rectification[9] **(GloVe**[4]**.OSCaR)** (Dev et al., 2020b): The rationale is that linear projective methods are too aggressive in modifying the entire embedding space. OSCaR rectifies two concepts of interest (gender and occupations), such that these subspaces are orthogonal in the debiased space.

Iterative Nullspace Linear Projection[10] **(GloVe**[4]**.INLP)** (Ravfogel et al., 2020): Rather than defining a gender direction, INLP *learns* the most informative decision boundary for classifying gendered and gender-neutral words. All words are projected to the nullspace of the gender subspace, and the process proceeds iteratively until gender information is sufficiently erased. A closely related method is the D_4 algorithm (Davis et al., 2020).

Repulse Attract Neutralize Debias[11] **(GloVe**[4]**.RAN)** (Kumar et al., 2020): Motivated by the persistence of implicit bias after debiasing through projective methods (observed as clustering and recoverability), RAN-debias attempts to address both direct bias and gender-based proximity bias.

3.2 Explicit gender words test set and error definitions

Firstly, we construct a test set where every sentence pair is of the form [A person *verb object*] → [(A) **gender word** *verb object*] (the correct inference is always neutral since a person can be of any gender). Verbs ($n = 27$) and objects ($n = 184$) are paired to create $n = 1968$ unique premise sentences[12]. Gender words are taken to be {man, woman, guy, girl, gentleman, lady, He, She}, following (Dev et al., 2020a) with the addition of the pronouns, for a total test set S of $|S| = 15744$ sentence pairs where hypotheses represent binary genders evenly (denoted S_M, S_F, $|S_M| = |S_F|$).

For every hypothesis sentence in the test set, the ideal prediction probability vector is $(N, E, C) = (1, 0, 0)$. We could define the error on the test set as the average Euclidean distance from the ideal distribution:

$$\mathcal{E} = \frac{1}{|S|} \sum_{i \in S} \|(1,0,0) - (N,E,C)_i\|_2. \quad (1)$$

This task, test set, and error definition are simple, and yet they encapsulate the central challenge of the debiasing field: to create attribute-aware (required to obtain the Neutral prediction) but attribute-unbiased embeddings.

A weaker, but still potentially desirable, condition might be to minimize the effect of gender

---

[3] https://github.com/tolga-b/debiaswe

[4] The base (undebiased) embeddings are GloVe trained on the 2017 January Wikipedia dump (vocab contains 322,636 tokens). Available at https://github.com/uclanlp/gn_glove.

[5] https://github.com/uclanlp/gn_glove

[6] https://github.com/kanekomasahiro/gp_debias

[7] https://github.com/uvavision/Double-Hard-Debias

[8] https://github.com/anlausch/DEBIE

[9] https://github.com/sunipa/OSCaR-Orthogonal-Subspace-Correction-and-Rectification

[10] https://github.com/shauli-ravfogel/nullspace_projection. The projection matrix computed for our base GloVe embeddings is available at https://github.com/hillary-dawkins/MAB.

[11] https://github.com/TimeTraveller-San/RAN-Debias

[12] Verbs and objects are taken from (Dev et al., 2020a) word lists (https://github.com/sunipa/On-Measuring-and-Mitigating-Biased-Inferences-of-Word-Embeddings) and are paired using the same pairing rules.

while not requiring that the model be gender-aware. Typically, this means that all hypotheses tend towards an Entail prediction, regardless of gender. We could define the error as the average distance between probability vectors between genders:

$$d = \frac{1}{2|S|} \left\| \sum_{i \in S_M} (N,E,C)_i - \sum_{j \in S_F} (N,E,C)_j \right\|_2 . \tag{2}$$

A gender-agnostic model could achieve zero error by this definition even with an accuracy of zero on the test set.

Table 1 shows the results for this test set on all the embeddings of interest. None of the debiased embeddings successfully mitigate the marked attribute error. A similar test set shows that the effect persists on implicit gender words (e.g. names). Results are shown in the appendix.

3.3 Latent gender carriers: Stereotyped occupations

Next, we would like to check if the gender-induced marked attribute bias can affect entities which should be gender neutral, but turn out to be hidden carriers of a gender attribute (e.g. stereotypical occupations). The same template [A person *verb object*] → [A/An **occupation** *verb object*] was used with the common vocabulary. Stereotypical occupations ($n = 32$) were sourced from Bolukbasi et al. (2016), and the SemBias test set. Examples are (doctor, engineer, boss, etc. vs. nurse, maid, homemaker, etc.). In total there are 62,976 sentence pairs in the test set[13].

Results are shown in Table 2. A permutation test is used to check if dividing the occupations into groups according to gender stereotypes produces a significant difference in the probability vectors (rather than dividing them randomly). As shown, the marked attribute effect persists on stereotypical occupations, especially on original embeddings. This is an important result because it highlights that unintended behaviour can appear in unexpected places due to a latent attribute. Previously, GBETs have focused on how explicit gender words are treated under biased models. To our knowledge, this is the first GBET designed to analyze unintended behaviour on a latent attribute carrier.

Note that this task is easier to correct than the explicit gender words because occupation words have defining characteristics beyond gender. That is, a debiasing method such as Iterative Nullspace Projection can perform well by removing gender information entirely. This does not mean that the challenge of having a gender-aware but gender-unbiased embedding is solved, but it does provide evidence that latent gender effects can be mitigated using linear projective methods. The full extent of latent biased-attribute effects and possible mitigation strategies should be investigated further.

4 Intrinsic bias measures

How to define bias on an embedding space remains an active area of study. In general, we seek to understand how the intrinsic or geometric properties of an embedding space translate to real observable bias in downstream tasks. Intrinsic properties are easy to compute quickly, whereas computing performance on downstream tasks requires us to train new models for every case. Understanding of the correlations between the two gives insight on how word embeddings should be debiased.

As a case study, let us focus on the marked attribute error $\mathcal{E}$ on the explicit gender words (shown in Table 1). Recall that this measure of bias is of interest because zero error corresponds to the gold standard: having an attribute-aware model, while simultaneously not using the gender attribute to make inappropriate inferences. In this section, we look at 5 existing intrinsic bias measures: Direct Bias, Clustering, Recoverability, Gender-based Illicit Proximity Estimate (GIPE), and SemBias. We will investigate whether any of these measures are predictive of the marked attribute effect.

Recall that direct bias was the first measure to be proposed; it simply measures the average projection of word vectors onto a predefined gender direction. Early methods (i.e. Hard Debias and GN-GloVe) defined bias in the embedding space completely as direct bias. The idea of clustering and recoverability refer to a classifier's ability to correctly reassign gender labels to words, even after debiasing methods have been applied. Gonen and Goldberg (2019)'s observation of clustering and recoverability sparked new interest in defining metrics for indirect bias on the embedding space. Although clustering and recoverability do not provide well-defined measures of bias given an embedding space (as they depend training implementation - though they could be said to provide a lower bound), many new debiasing proposals will cite re-

---

[13] The exact word set used to produce these results is available at https://github.com/hillary-dawkins/MAB.

Table 1: Results of the marked attribute test set on **explicit gender words**. Due to varying results on gender nouns vs. pronouns, results are shown separately for each case (M and F represent averages across the gender nouns). Some debiased embeddings are able to eliminate the distance across pronouns (really by definition since $\vec{she} \approx \vec{he}$ in these cases), but none are able to eliminate differences between the gender nouns significantly. Even when differences between genders are minimized, distance from the ideal distribution (error $\mathcal{E}$) remains or increases. This highlights the challenge of creating gender-aware but not gender-biased embeddings.

Emb.method	Gender	N	E	C	Gender	N	E	C	d	$\mathcal{E}$
GloVe	M	0.7832	0.1966	0.0202	F	0.9449	0.0401	0.0149	0.225	0.182
	he	0.0982	0.8838	0.0180	she	0.6549	0.3137	0.0315	0.797	0.865
GloVe.HD	M	0.8306	0.1329	0.0365	F	0.9269	0.0499	0.0232	0.128	0.155
	he	0.2944	0.6737	0.0319	she	0.5174	0.4334	0.0491	0.328	0.813
GN-GloVe	M	0.6339	0.3402	0.0259	F	0.9169	0.0461	0.0370	0.408	0.301
	he	0.1767	0.7968	0.0265	she	0.8223	0.1405	0.0373	0.921	**0.688**
GN-GloVe(w_a)	M	0.8446	0.1254	0.0300	F	0.9211	0.0395	0.0394	**0.115**	**0.149**
	he	0.1430	0.8266	0.0304	she	0.4237	0.5367	0.0396	0.404	0.990
GloVe.DHD	M	0.7013	0.2685	0.0302	F	0.9282	0.0510	0.0209	0.315	0.247
	he	0.1566	0.8187	0.0247	she	0.1597	0.8139	0.0264	0.006	1.173
GloVe.GP	M	0.6172	0.3521	0.0306	F	0.8777	0.0693	0.0530	0.385	0.336
	he	0.2443	0.7262	0.0295	she	0.6481	0.3040	0.0480	0.585	0.758
GloVe.BAM	M	0.7983	0.1703	0.0314	F	0.9329	0.0447	0.0224	0.184	0.175
	he	0.1625	0.8083	0.0292	she	0.6752	0.2878	0.0369	0.731	0.800
GloVe.OSCaR	M	0.8233	0.1572	0.0195	F	0.9431	0.0400	0.0169	0.168	0.154
	he	0.1482	0.8292	0.0226	she	0.8428	0.1278	0.0294	0.987	0.697
GloVe.RAN	M	0.8055	0.1686	0.0260	F	0.8994	0.0701	0.0305	0.136	0.193
	he	0.1939	0.7811	0.0250	she	0.5962	0.3420	0.0618	0.597	0.828
GloVe.INLP	M	0.8298	0.1537	0.0166	F	0.9204	0.0633	0.0164	0.128	0.167
	he	0.1081	0.8710	0.0209	she	0.1119	0.8672	0.0209	**0.005**	1.244

Table 2: Results of marked attribute test set on **stereotypical occupations**. Each (N,E,C) probability vector is averaged over the 1968 unique premise sentences and the gender attribute words from each category (M or F) ($n = 31,488$ sentences for each gender). Smaller distances between the M and F vectors indicate less gender bias. The significance of the difference was evaluated using a permutation test; the alternate distance d^* is computed for 10,000 randomly sampled partitions of the occupations into two groups. The significance value is the proportion of these samples to generate a distance $d^* > d$. This gives us an idea of whether the defined partition, based on gender, is a meaningful grouping. Smaller significance values indicate that the defined partition is non-random with respect to the distance.

Emb.method	M attribute (N, E, C)	F attribute (N, E, C)	Distance d	Significance
GloVe	(0.6000, 0.3350, 0.0650)	(0.7378, 0.1711, 0.0910)	0.216	0.0001
GloVe.HD	(0.4975, 0.4500, 0.0525)	(0.6075, 0.3357, 0.0568)	0.159	0.0408
GN-GloVe	(0.5026, 0.4434, 0.0540)	(0.7126, 0.2036, 0.0838)	0.320	0.0000
GN-GloVe(w_a)	(0.5309, 0.3915, 0.0776)	(0.6197, 0.2771, 0.1032)	0.147	0.0478
GloVe.DHD	(0.5285, 0.4126, 0.0589)	(0.6513, 0.2811, 0.0676)	0.180	0.0038
GloVe.GP	(0.5016, 0.4380, 0.0604)	(0.6479, 0.2639, 0.0882)	0.229	0.0010
GloVe.BAM	(0.6293, 0.3077, 0.0630)	(0.7116, 0.1972, 0.0912)	0.141	0.0060
GloVe.OSCaR	(0.5577, 0.3901, 0.0522)	(0.6789, 0.2400, 0.0812)	0.195	0.0036
GloVe.RAN	(0.5393, 0.3933, 0.0674)	(0.5924, 0.3026, 0.1050)	0.112	0.0477
GloVe.INLP	(0.5065, 0.4197, 0.0739)	(0.5465, 0.3949, 0.0587)	0.050	0.6595

duced clustering as a positive result. The effect on downstream applications is not well understood as of yet. The Gender-based Illicit Proximity Estimate (GIPE) measures the extent of undue proximities in the embedding space due to a pervasive gender attribute. Lastly, the SemBias analogy test set measures whether gender-biased analogies exist within the embedding space based on vector arithmetic properties.

Implementation details for each measure as well as the experimental set of embeddings ($n = 16$) are given in the appendix. The average Direct Bias on the embedding space was found to have a Pearson correlation coefficient of 0.104 with the marked attribute error. The Clustering v-measure[14] (Rosenberg and Hirschberg, 2007) achieved a correlation coefficient of 0.184. Recoverability was attempted using an SVM with a linear decision boundary, an SVM with a non-linear (radial basis function) kernel, logistic regression, and a simple 1-hidden-layer fully-connected network. All recoverability correlation results were comparable, but the best coefficient of 0.223 was achieved by logistic regression. The GIPE[15] had a correlation coefficient of 0.432. The SemBias[16] test set had a correlation coefficient of 0.091. The full correlation matrix between all intrinsic bias measures can be found in the appendix. The results suggest that the marked attribute effect is not well correlated with any present notion of intrinsic bias, therefore we do not have a good understanding of how the word embedding properties contribute to this type of observable bias.

In seeking a potential solution, we make note of a new intrinsic bias measure, multi-dimensional information-weighted direct bias (MIDB), found to have a more meaningful correlation of 0.667 with the marked attribute error. We define the MIDB of a particular word $\vec{x}$ to be a weighted average over inner products with basis vectors of a multi-dimensional gender subspace:

$$\text{MIDB}_d(x) = \sum_{i=1}^{d} a_i \langle g_i | x \rangle \tag{3}$$

where $\{g_i\}$ form an orthonormal basis for the gender subspace, here defined as the first d principal components summarizing difference vectors $\{\delta_{jk}\}$. The difference vectors are taken as all pairwise differences[17] between vectors in defined gender sets (here common names were used[13]): $\{\delta_{jk}\} = \vec{f}_j - \vec{m}_k$, $f_j \in F_{\text{names}}$, $m_k \in M_{\text{names}}$ ($|M_{\text{names}}| = |F_{\text{names}}| = 100$). The weighting a_i is the proportion of variance explained by the i^{th} principal component, and d is a hyperparameter controlling the number of dimensions to keep[18].

New proposals for defining a gender direction or subspace potentially have far reaching consequences in the landscape of intrinsic bias measures and their related debiasing schemes. In fact all of Clustering, Recoverability, GIPE, and SemBias use the classic uni-dimensional gender direction $\vec{g}$ within their definitions. The weak observed correlation between DB and MIDB suggests that these subspaces are independent. Swapping in a uniquely informative gender subspace to the existing indirect measures would produce a new family of intrinsic bias measures. The observed utility of names in defining a meaningful gender subspace is encouraging because it opens an obvious avenue for this method to be applied to attributes of interest beyond gender (e.g. race or ethnicity).

5 Multi-dimensional information-weighted soft projection

In this section we motivate the above search for an informative intrinsic bias measure. As discussed, a greater understanding of how embedding properties influence observed bias can inform new debiasing techniques. Translating the idea of MIDB into a debiasing scheme yields Multi-dimensional Information-weighted Soft Projection (MISP).

In this debiasing procedure, we project all words into the nullspace of the multi-dimensional gender subspace, proportional to our belief that certain dimensions actually encode the latent idea of gender:

$$\vec{w}_{deb} = \vec{w} - \sum_{i=1}^{d} a_i \langle g_i | w \rangle | g_i \rangle \tag{4}$$

where $\vec{w}$ is the input embedding, $\vec{w}_{deb}$ is the debiased output embedding, and all other quantities are defined as in eq. (3).

---

[14] With cluster size $n = 1500$ (which lead to the highest observed correlation); see appendix.

[15] Using an indirect bias threshold of $\theta = 0.05$, and number of nearest neighbours $n = 100$.

[16] The SemBias score was taken as the proportion of analogy examples in the test set for which the embedding space returns the correct definitional analogy.

[17] Using all pairwise differences creates a matrix with rank much less than the dimension of the matrix, however the rank is still much larger than d (the number of principal components to extract) so it doesn't cause a problem.

[18] On our set of experimental embeddings, $d = 4$ was empirically found to produce the 0.7 correlation result.

Table 3: Results for word similarity and analogy benchmarks. Results on the word analogy tasks are reported as percentage accuracy. Results on the word similarity tasks are reported as a Spearman correlation (× 100). Application of MISP does not alter the overall quality of word embeddings as measured by these classic test sets.

Embedding.method	Sem	Syn	Google-Total	MSR	RG	MTurk	MEN	SL999
GloVe	80.48	62.76	70.80	51.49	75.29	64.27	72.19	34.86
GloVe.MISP	80.49	62.81	70.84	51.51	76.06	64.32	72.41	35.04
GN-GloVe	77.62	61.60	68.87	49.29	74.11	66.36	74.49	37.12
GN-GloVe(w_a)	77.68	61.56	68.87	49.38	75.46	66.55	74.72	37.53
GN-GloVe(w_a).MISP	77.68	61.59	68.89	49.26	75.49	66.45	74.76	37.60

As shown in Table 1, the GN-GloVe(w_a) embeddings are currently the top performers on the explicit gender words test set, as measured by either error $\mathcal{E} = 0.149$, or distance $d = 0.115$. Applying MISP to GN-GloVe(w_a) embeddings (denoted GN-GloVe(w_a).MISP), we achieve an error on the explicit gender words test set of $\mathcal{E} = 0.1107$, a 26% error reduction over the previous best. The distance d between genders is reduced to $d = 0.08744$, a 21% reduction over the previous best. Successful concatenation suggests that this technique is distinct, and independently useful, from techniques that seek to minimize the traditional direct bias (including GN-GloVe). This observation is consistent with the weak observed correlation between direct bias and MIDB$_4$ on the experimental set of embeddings.

Computing the intrinsic bias measures Clustering, Recoverability, GIPE and SemBias on the newly created embedding space GN-GloVe(w_a).MISP (compared to the base GN-GloVe(w_a)), we observe a clustering v-score of 0.498 (previously 0.497)[19], a recoverability accuracy of 0.992 (previously 0.993)[20], a GIPE of 0.1169 (previously 0.1173)[21], and a SemBias score of 0.938 (previously 0.938)[22]. The MISP method did not reduce bias by any of these measures, although this is not particularly surprising as it was designed to address the marked attribute effect (through MIDB). It is encouraging however that none of these bias measures were increased. In other words, there is no expected trade-off between the reduced marked attribute error and any previous debiasing work that relied on these measures. The SemBias result informs us that MISP did not reintroduce any harmful biased analogies, for example.

For reference, if we apply the analogous multi-dimensional hard debias method (i.e. equation (4) where all weights a_i are set to 1), the output embeddings GN-Glove(w_a).MHD do not successfully mitigate the marked attribute effect ($\mathcal{E} = 0.1501$, $d = 0.1603$). This suggests that the soft nature of the projection is a key ingredient.

Furthermore, we provide some evidence that specifically the information weighting of the soft projection is a good ingredient as follows. Recall that we are attenuating components of each basis vector according to our belief in that vector as a good gender direction. The basis vectors are defined to be the first d principal components, weighted by their corresponding variance explained. Therefore the first basis vector receives the greatest weight and so on. To test the significance of this decision, we define alternative debiased embeddings by applying MISP where the weights get reassigned to the "wrong" vector (for $d = 4$, we have 23 alternative pairings). We observe that **none** of the 23 alternatives obtain an error $\mathcal{E}$ less than the "true" implementation of MISP. This suggests that weighting the components by order of information is a good ingredient. Values of $\mathcal{E}$ for the alternate embeddings can be found in the appendix. Model parameters for each case are made available in order to reproduce this argument on any extended version of the MAB test set.

Information weighting is an interesting idea because it could be applied to either defined or learned gender subspaces alike. For instance, if the basis vectors of a gender subspace are taken as the iteratively learned linear decision boundaries (as in INLP), we could investigate weighting each dimension by the accuracy acc_i of classification on each iteration, as $a_i = (1 - 2acc_i)$. In this way, dimensions receive weights proportional to their

---

[19] Where clustering size $n = 1500$.

[20] This is the highest accuracy achieved by any of the four classification methods tested; implementation details are in the appendix.

[21] Computed with indirect bias threshold $\theta = 0.03$, and number of nearest neighbours $n = 100$.

[22] Reported as the proportion of samples in the full test set to return the definitional analogy; higher scores are better.

ability to predict gender information. When accuracy reaches 0.5, no gender information remains, the learned decision boundary is meaningless, and the basis vector receives zero weighting.

Finally, as with any debiasing method, we wish to verify that application of the method has not damaged the overall embedding quality. We assess the MISP embeddings on a handful of classic analogy and word semantic similarity benchmarks. The word similarity benchmarks measure how closely the word embeddings capture similarity between words compared to human annotation. We use the following datasets: RG (Rubenstein and Goodenough, 1965), MTurk (Yih and Qazvinian, 2012), MEN (Baroni, 2014), and SimLex999 (Hill et al., 2015). The word analogy task measures how well the word embeddings capture semantic and syntactic relationships among words as vector properties. We report on the Google (Mikolov et al., 2013a), and MSR (Mikolov et al., 2013c) test sets. Results were obtained following the word embedding benchmark package[23] (Jastrzebski et al., 2017). As shown, application of MISP does not alter the overall word embedding quality.

6 Conclusion

This paper highlights a new observation of gender bias in a downstream setting: marked attribute bias in natural language inference. The current inference is that "person" implies male, while "person" does not imply female. Consequently, this inference is being baked into our models of natural language understanding. The effect was shown to persist on explicitly defined gender words and on latent gender-attribute carriers. Based on an assessment of the current debiasing landscape, none of the current debiasing methods satisfactorily mitigate the marked attribute error, and furthermore none of the intrinsic bias measures are useful at predicting the marked attribute effect.

By noticing a more meaningful correlation with a newly identified intrinsic bias measure, we propose a new debiasing scheme: multi-dimensional information-weighted soft projection (MISP). This method introduces several concepts, including the use of a multi-dimensional defined gender subspace. Previously, the concept of a defined gender subspace always appeared as a single dimension. The iterative nullspace projection method implicitly uses higher learned dimensions, however this requires learning a new decision boundary at every iteration, subject to the implementation of a training procedure. Furthermore, the learned dimensions were not used to define any bias metric, they were strictly used operationally for the debiasing procedure. MISP also introduces the idea of a soft or partial projection, where weights are informed by some measure of the dimension's ability to capture the intended latent concept of a gender direction. Both of these ideas could be further explored and extended to create new notions of indirect bias, which in turn could inform more sophisticated debiasing procedures.

Multi-dimensional information-weighted soft projection applied to GN-GloVe(w_a) produces new debiased embeddings that achieve the lowest error on the marked attribute bias test set, a 26% reduction over the previous best, and a 45% reduction over the original undebiased embeddings. Error reduction on this test set is thought to encapsulate the overall goal of producing gender-aware but gender-unbiased embeddings. Therefore, this method and its composite ingredients warrant further investigation. Each of the marked attribute bias test sets are made available for further exploration and iteration on these ideas.

Acknowledgements

We thank Daniel Gillis, Judi McCuaig, Stefan Kremer, and Graham Taylor for their insightful comments and discussions. We thank the anonymous reviewers for their comments and suggestions which improved the final version of this manuscript. This work is financially supported by the Government of Canada through the NSERC CGS-D program (CGSD3-518897-2018).

References

E. Bruni N. K. Tran M. Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, volume 29, pages 4349–4357. Curran Associates, Inc.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference.

---

[23] https://github.com/kudkudak/word-embeddings-benchmarks

In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Brent D. Davis, Ethan Jackson, and Daniel J. Lizotte. 2020. Decision-directed data decomposition.

Sunipa Dev, Tao Li, Jeff M. Phillips, and Vivek Srikumar. 2020a. On measuring and mitigating biased inferences of word embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7659–7666.

Sunipa Dev, Tao Li, Jeff M Phillips, and Vivek Srikumar. 2020b. Oscar: Orthogonal subspace correction and rectification of biases in word embeddings.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of NAACL-HLT*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Stanislaw Jastrzebski, Damian Lesniak, and Wojciech Marian Czarnecki. 2017. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *CoRR*, abs/1702.02170.

Masahiro Kaneko and Danushka Bollegala. 2019. Gender-preserving debiasing for pre-trained word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1641–1650, Florence, Italy. Association for Computational Linguistics.

Vaibhav Kumar, Tenzin Singhay Bhotia, Vaibhav Kumar, and Tanmoy Chakraborty. 2020. Nurse is closer to woman than surgeon? mitigating gender-biased proximities in word embeddings. *Transactions of the Association for Computational Linguistics*, 8:486–503.

Anne Lauscher, Goran Glavaš, Simone Paolo Ponzetto, and Ivan Vulić. 2019. A general framework for implicit and explicit debiasing of distributional word vector spaces. In *Accepted for publication at the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7237–7256. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.

Tianlu Wang, Xi Victoria Lin, Nazneen Fatema Rajani, Bryan McCann, Vicente Ordonez, and Caiming Xiong. 2020. Double-hard debias: Tailoring word embeddings for gender bias mitigation. In *Association for Computational Linguistics (ACL)*.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 616–620, Montréal, Canada. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018a. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018b. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853, Brussels, Belgium. Association for Computational Linguistics.

A Implicit gender carriers: Names

As mentioned in the main text, we check if the marked attribute effect will persist through implicit gender words. These are words with no gender attribute by definition, but are usually associated with a specific binary gender (e.g. names). This test set uses a similar template: [A person *verb object*] → [**Name** *verb object*], using the same (verb, object) vocabulary as above. Names[24] ($n = 64$) are sourced from the most common names of the previous decade in the US, according to the Social Security Administration[25]. In total there are $n = 125,952$ sentence pairs in the test set.

Results are shown in Table 4. In short, the same effect is observed on names, especially on the original embeddings. A permutation test was used to check whether the stratification of names by gender was a non-random division according to the observed bias.

B Intrinsic bias measures and correlations

Please refer to tables 5 and 6.

C Alternate weighted embeddings

As discussed in the main text, we compute the error $\mathcal{E}$ on the explicit gender words test set for alternate soft-weighted embeddings. The alternate embeddings are created by permuting the weights to be matched with the incorrect basis vectors. For example, the permutation denoted 1243 means that weight a_1 is applied to basis vector $\vec{g_1}$, a_2 to $\vec{g_2}$, a_4 to $\vec{g_3}$, and a_3 to $\vec{g_4}$. Results for all alternate permutations and their errors are as follows: (permutation = 1243, $\mathcal{E}$ = 0.1574), (1324, 0.2331), (1342, 0.1919), (1423, 0.1330) (1432, 0.1487) (2134, 0.1273) (2143, 0.1565) (2314, 0.2289) (2341, 0.1963) (2413, 0.1639) (2431, 0.1287) (3124, 0.1951) (3142, 0.1694) (3214, 0.2813) (3241, 0.1602) (3412, 0.2110) (3421, 0.1732) (4123, 0.1945) (4132, 0.1764) (4213, 0.1879) (4231, 0.1363) (4312, 0.1241) (4321, 0.1435).

---

[24] The exact word set used to produce these results is available at https://github.com/hillary-dawkins/MAB.

[25] https://www.ssa.gov/oact/babynames/

Table 4: Results of marked attribute test set on **names**. Each (N,E,C) probability vector is averaged over the 1968 unique premise sentences and the gender attribute words from each category (M or F) ($n = 62,976$ sentences for each gender). Smaller distances between the M and F vectors indicate less gender bias. The significance of the difference was evaluated using a permutation test; the alternate distance d^* is computed for 10,000 randomly sampled partitions of the names into two groups. The significance value is the proportion of these samples to generate a distance $d^* > d$. This gives us an idea of whether the defined partition, based on gender, is a meaningful grouping. Smaller significance values indicate that the defined partition is non-random with respect to the distance.

Emb.method	M attribute (N, E, C)	F attribute (N, E, C)	Distance d	Significance
GloVe	(0.4657, 0.4766, 0.0577)	(0.7283, 0.1598, 0.1120)	0.415	0.0000
GloVe.HD	(0.5745, 0.3547, 0.0708)	(0.6685, 0.2760, 0.0555)	0.124	0.0140
GN-GloVe	(0.4619, 0.4713, 0.0668)	(0.7209, 0.1906, 0.0885)	0.383	0.0000
GN-GloVe(w_a)	(0.5882, 0.2878, 0.1240)	(0.6662, 0.2321, 0.1017)	0.098	0.0241
GloVe.DHD	(0.4731, 0.4464, 0.0805)	(0.5690, 0.3529, 0.0780)	0.134	0.0192
GloVe.GP	(0.5488, 0.3761, 0.0751)	(0.7470, 0.1677, 0.0853)	0.288	0.0000
GloVe.BAM	(0.5941, 0.3424, 0.0635)	(0.7698, 0.1628, 0.0674)	0.251	0.0000
GloVe.OSCaR	(0.6012, 0.3149, 0.0839)	(0.7191, 0.2020, 0.0789)	0.163	0.0001
GloVe.RAN	(0.5295, 0.3865, 0.0839)	(0.6920, 0.2151, 0.0929)	0.236	0.0000
GloVe.INLP	(0.5091, 0.4042, 0.0867)	(0.5447, 0.3639, 0.0914)	0.054	0.4049

Table 5: Pearson correlation matrix between intrinsic bias measures (and marked attribute error) on the experimental set of embeddings. MIDB obtains the highest correlation with the marked attribute error $\mathcal{E}$; the GIPE was also observed to have a weak correlation. Recoverability bias is most related to the direct bias. The sub-matrix among the SemBias results indicate that trade-off is mostly happening between "definitional" and "other" analogies.

	DB_{vt}	MIDB	Clus:v_{1500}	Rec:LR	SBdef	SBstereo	GIPE:0.03	$\mathcal{E}$
DB_{vt}	1	-0.166	0.694	0.814	0.161	-0.045	0.350	0.104
MIDB		1	-0.145	-0.020	-0.273	0.005	-0.003	0.667
Clus:v_{1500}			1	0.776	0.600	-0.185	0.271	0.184
Rec:LR				1	0.786	-0.390	0.290	0.223
SBdef					1	-0.693	0.304	0.091
SBstereo						1	-0.487	-0.270
GIPE:0.03							1	0.432
$\mathcal{E}$								1

Table 6: Intrinsic bias measures of interest on the experimental set of embeddings. There are two base (undebiased) embeddings, word2vec and GloVe. All other embedding spaces are obtained by applying a debiasing method, where each method found here is described in the main text. Implementation notes:

DB and MIDB: The direct bias (DB) and the new multi-dimensional information-weighted direct bias (MIDB) are average measures over a specific (ideally gender-neutral) vocabulary V_t. V_t ($n = 46960$) is defined by taking the 50,000 most frequent words in the common vocabulary between word2vec and GloVe, filtering out punctuation, numbers, and removing the gender-specific word set V_s ($n = 1622$), defined as the union of gender-specific word sets used in previous works (Bolukbasi et al., 2016; Zhao et al., 2018b). DB is defined as the projection onto a gender direction, here taken to be the $\vec{she} - \vec{he}$ direction. For debiasing methods that promote $\vec{she} \approx \vec{he}$, the DB is not well defined (although it can be computed numerically, it is unstable). We leave these cases as NA rather than a spurious numerical value.

Clustering: The clustering experiment follows (Gonen and Goldberg, 2019) in taking the $n \in [500, 1500]$ "most biased" words in the original embedding space (according to their projection on the $\vec{she} - \vec{he}$ axis), and then applying k-means ($k = 2$) clustering on the words in the debiased embedding space. Bias is reported as the either the clustering accuracy or the v-measure (only $n = 1500$ shown here with v-measure).

Recoverability: Similarly, the dataset ($n = 5000$) is taken to be the most biased words in the original embedding space, where bias labels are assigned according to the projection on the gender direction ($n = 2500$ taken from each class). Several classifiers (SVM with a linear decision boundary, SVM with an RBF kernel, logisitic regression, and a simple fully-connected 1-hidden layer network) were trained on 20% of the dataset with balanced classes. Recoverability bias is reported as the accuracy of classification on the remaining test set (only logisitic regression shown here).

SemBias: The SemBias analogy test set is available from (Zhao et al., 2018b). The set contains $n = 440$ tuples of possible analogies $(\vec{a}, \vec{b})$: 1 definitional analogy (e.g. king, queen), 1 stereotypical analogy (e.g. doctor, nurse), and 2 other analogies (e.g. cup, plate). For every sample, the best analogy is selected as the one to maximize $cos(\vec{he} - \vec{she}, \vec{a} - \vec{b})$. Bias is reported as the proportion of samples to return a definitional analogy, a stereotypical analogy, and an "other" analogy. (Only definitional and stereotypical shown here.)

GIPE: The gender-based illicit proximity bias (GIPE) (see (Kumar et al., 2020) for details) was computed with $n = 100$ nearest neighbours for each word, with an indirect bias threshold of $\theta \in [0.03, 0.05]$ following (Kumar et al., 2020). (Only $\theta = 0.03$ shown here.)

Full results, plus all code, embedding files, and word sets needed to replicate these results are available at https://github.com/hillary-dawkins/MAB.

Emb.method	DB$_{vt}$	MIDB	Clus:v$_{1500}$	Rec:LR	SBdef	SBstereo	GIPE:0.03	$\mathcal{E}$
w2v	0.052	0.023	0.933	0.992	0.830	0.134	0.021	0.206
w2v.HD	0.000	0.007	0.440	0.887	0.759	0.114	0.014	0.163
w2v.DHD	NA	0.025	0.271	0.881	0.295	0.373	0.014	0.164
w2v.BAM	0.061	0.038	0.844	0.974	0.814	0.136	0.023	0.131
w2v.OSCaR	0.050	0.024	0.928	0.993	0.830	0.134	0.021	0.188
GloVe	0.055	-0.032	0.984	1.000	0.802	0.109	0.115	0.198
GloVe.HD	0.000	-0.004	0.302	0.927	0.786	0.130	0.070	0.155
GN-GloVe	0.038	0.172	0.588	0.999	0.977	0.014	0.141	0.301
GN-GloVe(w_a)	0.068	-0.096	0.497	0.989	0.939	0.011	0.117	0.149
GloVe.DHD	NA	0.201	0.258	0.903	0.250	0.123	0.064	0.247
GloVe.GP	0.059	0.068	0.996	1.000	0.843	0.080	0.145	0.336
GN-GloVe.GP	0.036	0.006	0.601	0.999	0.984	0.011	0.118	0.179
GloVe.BAM	0.068	-0.019	0.964	0.999	0.775	0.145	0.137	0.175
GloVe.OSCaR	0.056	-0.012	0.984	1.000	0.814	0.102	0.117	0.154
GloVe.RAN	0.044	-0.001	0.419	0.951	0.927	0.011	0.040	0.193
GloVe.INLP	NA	-0.001	0.015	0.660	0.198	0.160	0.080	0.167

VLM: Task-agnostic Video-Language Model Pre-training for Video Understanding

Hu Xu[1], **Gargi Ghosh**[1], **Po-Yao Huang**[12], **Prahal Arora**[1], **Masoumeh Aminzadeh**[1]
Christoph Feichtenhofer[1], **Florian Metze**[1] and **Luke Zettlemoyer**[1]
[1]Facebook AI
[2]Carnegie Mellon University
`{huxu,gghosh,berniehuang,prarora,masoumeha,`
`feichtenhofer,fmetze,lsz}@fb.com`

Abstract

We present a simplified, task-agnostic multi-modal pre-training approach that can accept either video or text input, or both for a variety of end tasks. Existing pre-training are task-specific by adopting either a single cross-modal encoder that requires both modalities, limiting their use for retrieval-style end tasks or more complex multitask learning with two unimodal encoders, limiting early cross-modal fusion. We instead introduce new pretraining masking schemes that better mix across modalities (e.g. by forcing masks for text to predict the closest video embeddings) while also maintaining separability (e.g. unimodal predictions are sometimes required, without using all the input). Experimental results show strong performance across a wider range of tasks than any previous methods, often outperforming task-specific pre-training[1].

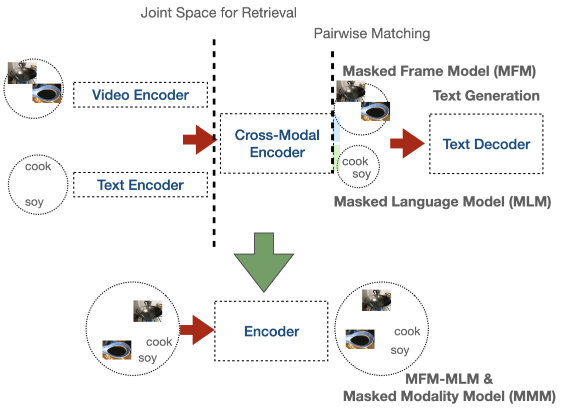

Figure 1: Existing models (upper figure) adopt complex architectures and multiple task-specific training to merge two streams of data to cover a wide range of downstream tasks (such as retrieval or text generation). Our video-language model (VLM) (lower figure) uses a single BERT encoder for task-agnostic pre-training (e.g. only masking tokens, no matching or alignment for specific end tasks) in a joint feature space, while still covering a wide range of tasks (see Figure 3).

1 Introduction

We study the challenge of achieving *task-agnostic* pre-training for multimodal video understanding, building on recent unimodal approaches such as pretrained language models for text (Peters et al., 2018; Devlin et al., 2019). Although certain language models are near task-agnostic (Devlin et al., 2019; Lewis et al., 2020) on NLP tasks, being task-agnostic on multi-modal tasks are more challenging due to cross-modal tasks such as text-video retrieval. Existing video-and-language pre-trainings are task-specific, which adopt either (1) a cross-modal single encoder (Sun et al., 2019b,a; Zhu and Yang, 2020) favoring tasks that require cross-modal reasoning (e.g. video captioning), or (2) multiple unimodal encoders/decoders (Miech et al., 2019, 2020; Li et al., 2020b; Luo et al., 2020; Korbar et al., 2020) combining specific tasks that require separately embedding each modality (e.g. video retrieval). We instead show that it is possible to pre-train a task-agnostic model called video-language model (VLM) that can accept text, video, or both as input.

As shown in Figure 1, this task-agnostic single encoder approach has several advantages: (1) it reduces the complexity of pre-training with multiple losses and models (e.g. Luo et al. (2020)), and (2) it holds less assumption on being close to end tasks as in retrieval-based pre-training Miech et al. (2020) and is as general as classic LMs, and (3) it encourages feature sharing among modalities when present, without sacrificing separability, and (4) it is more parameter efficient (see Section 5, we achieve strong performance with $\mathrm{BERT}_{\mathrm{BASE}}$ sized models). Table 1 summarizes the design choices of recent models.

Our encoder is a transformer block that combines the existing masked frame model and masked

---

[1] Code will be released under fairseq.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4227–4239
August 1–6, 2021.

language model (MFM-MLM) (Sun et al., 2019a; Li et al., 2020b; Luo et al., 2020) with two new methods to improve the learning of multi-modal fusion. First, we introduce a masking scheme called masked modality model (MMM) that randomly masks a whole modality for a portion of training examples (the rest of the examples goes for traditional MFM-MLM), thereby forcing the encoder to use the tokens from the other modality to produce tokens for the masked modality. We then introduce a single masked token loss to replace two (2) losses on video and text separately for MFM-MLM. Masked token loss uses the embeddings of both video and text tokens to learn joint hidden states for the encoder.

We also show it is possible to fine-tune a single encoder for a wide range of tasks by using task-specific attention masks. Experiments demonstrate that it performs well on a wider range of tasks than previous models, including outperforming task-specific pre-training baselines with unimodal encoders of similar hyper-parameters by more than 2% on retrieval tasks and 1% on video captioning. Note that these results are also achieved with a much smaller model than previous approaches, further demonstrating the improved fusion and sharing across modalities.

In summary, the main contributions of this paper are as follows: (1) we propose to pre-train a task-agnostic encoder for video understanding; (2) we introduce masked modality model (MMM) and masked token loss for cross-modal fusion during pre-training without sacrificing separability; (3) experimental results show that the proposed simple baseline achieves competitive performance with significantly fewer parameters.

2 Related Work

Numerous multimodal task-specific pre-training models are proposed for downstream visual-linguistic tasks. In video and text pre-training, existing research adopts different design choices regarding proxy tasks and neural architectures for end tasks (Luo et al., 2020).

On one hand, VideoBERT (Sun et al., 2019b), Unicoder-VL (Li et al., 2020a), VL-BERT (Su et al., 2020), UNITER (Chen et al., 2020), VLP (Zhou et al., 2018), ActBERT (Zhu and Yang, 2020) adopt a *shared* encoder approach, where the vision and text sequences are concatenated and input to a single Transformer(Vaswani et al., 2017) encoder. Although this approach is simple, it limits the types of downstream tasks to those that input both modalities simultaneously. For example, (Sun et al., 2019b) may not be able to perform joint retrieval tasks and added another decoder for video captioning during fine-tuning. (Zhu and Yang, 2020) uses `[CLS]` token for pairwise metric-learning based retrieval (which is an easier problem but requires a quadratic number of examples and is 50 times slower as reported in (Luo et al., 2020)).

Meanwhile, many existing approaches adopt or add task-specific pre-training to accommodate retrieval and video captioning tasks (e.g. *two-stream* encoders (video and text separately) and text decoders). For example, (Miech et al., 2019, 2020; Rouditchenko et al., 2020; Ging et al., 2020; Gabeur et al., 2020; Alayrac et al., 2020; Patrick et al., 2021; Huang et al., 2021) adopts a retrieval task for pre-training. CBT (Sun et al., 2019a), HERO (Li et al., 2020b), VideoAsMT (Korbar et al., 2020) and UniVL (Luo et al., 2020) adopt multi-task learning (MTL) to learn retrieval tasks on video and text encoders. HERO (Li et al., 2020b) and UniVL (Luo et al., 2020) adopts another cross-encoder to further learn the fusion of different modality. UniVL (Luo et al., 2020) and VideoAsMT (Korbar et al., 2020) add another text decoder for video captioning. Compared with the single-stream input in the shared encoder approach, two-stream encoders typically come with a complex architecture and proxy tasks to cover more end tasks. To the best of our knowledge, none of the existing works target task-agnostic pre-training.

2.1 Image-Text Pre-training

ViLBERT (Lu et al., 2019), LXMERT (Tan and Bansal, 2019) adopt two transformers for image and text encoding separately. VisualBERT (Li et al., 2019), Unicoder-VL (Li et al., 2020a), VL-BERT (Su et al., 2020), UNITER (Chen et al., 2020), Unified VLP (Zhou et al., 2020) use one shared BERT model. These models employ MLM and pairwise image-text matching as pretraining tasks which are effective for downstream multimodal tasks. Our fine-tuning for video captioning is inspired by Unified VLP (Zhou et al., 2020) that adopts attention masks and language model heads of BERT for image-captioning.

2.2 Video-Text Pre-training

VideoBERT (Sun et al., 2019b) and CBT (Sun et al., 2019a) are the first works to explore the capability of pre-training for video-text. Although VideoBERT and CBT pre-train the model on multimodal data, the downstream tasks mainly take video representation for further prediction. ActBERT (Zhu and Yang, 2020) is a weakly-supervised pre-training method. It leverages global action information to catalyze mutual interactions between linguistic texts and local regional objects and introduces a transformer block to encode global actions, local regional objects, and linguistic descriptions. HERO (Li et al., 2020b) encodes multimodal inputs in a hierarchical fashion. Besides, two new pre-training tasks, video-subtitle matching and frame order modeling, are designed to improve representation learning. VideoAsMT (Korbar et al., 2020) and UniVL (Luo et al., 2020) further adopt a BART-style(Lewis et al., 2020) text generation task for downstream tasks such as video captioning and UniVL adopts a EnhancedV training stage to mask all text tokens for better learning of generation.

3 Pre-training

As a reminder, our goal is to train a *task-agnostic* model for various tasks in video-text understanding. This section introduces task-agnostic proxies for pre-training. We first describe two masking schemes as a baseline: masked frame model (MFM) for video frames and masked language model (MLM) for text tokens (Sun et al., 2019a; Li et al., 2020b; Luo et al., 2020). Then we introduce masked modality model (MMM) that encourage to learn the representations of one modality from the other. Lastly, we introduce masked token loss that unifies losses on masked video and text tokens as a single loss function.

3.1 Vector Quantization and BERT

Assume we have a clip (v, t) sampled from a video, where v and t corresponds to video modality and text modality, respectively. Since videos are signals in continuous space, we first extract token embeddings from raw videos. We decode v into frames and then feed them into a (frozen) video encoder $\text{Encoder}_{\text{video}}(\cdot)$ and a trainable MLP layer to obtain *video tokens*:

$$\boldsymbol{x}_v = \text{MLP}(\text{Encoder}_{\text{video}}(\boldsymbol{f}_v)), \tag{1}$$

where we use a bolded symbol to indicate a sequence and $\boldsymbol{f}_v$ is a sequence of continuous frames from a video. We use S3D (Xie et al., 2018; Miech et al., 2020), which is pre-trained via self-supervised learning on the Howto100M dataset. The MLP layer allows the hidden size of video tokens to be the same as BERT's hidden sizes d: $x_v \in \mathbb{R}^d$. Similarly, vectors for text tokens $\boldsymbol{x}_t$ are obtained via embedding lookup as in BERT.

To simplify multi-modal pre-training, we adopt a single BERT transformer with minimum changes. We first concatenate video tokens $\boldsymbol{x}_v$ and text tokens $\boldsymbol{x}_t$ via the `[SEP]` token so video and text belongs to one corresponding segment of BERT:

$$\boldsymbol{x} = \texttt{[CLS]} \circ \boldsymbol{x}_v \circ \texttt{[SEP]} \circ \boldsymbol{x}_t \circ \texttt{[SEP]}. \tag{2}$$

We further mask $\boldsymbol{x}$ as $\boldsymbol{x}_{\text{masked}}$ (detailed in the next subsection) and feed the whole sequence into BERT:

$$\boldsymbol{h} = \text{BERT}(\boldsymbol{x}_{\text{masked}}), \tag{3}$$

where $\boldsymbol{h}$ indicates the hidden states of the last layer of BERT. To encourage learning video/text hidden states in a shared space for the masked token loss (introduced in Section 3.3), we use a *shared* head to predict video/text token embeddings via a linear projection layer:

$$e = \boldsymbol{W}h + b, \tag{4}$$

where $e \in \mathbb{R}^d$ and $\boldsymbol{W}$ and b are the weights from the prediction heads of BERT. In this way, our model learns a joint embedding space for both video and text tokens from inputs to outputs of BERT. This allows for pre-training a single encoder directly from any existing LMs and the only layer that requires initialization is the MLP layer.

3.2 MFM-MLM

Inspired by (Sun et al., 2019a; Li et al., 2020b; Luo et al., 2020), we adopt masked frame model (MFM) for videos and masked language model (MLM) for text as a baseline. Note that unlike LMs that typically come with a fixed vocabulary with a special `[MASK]` token, video tokens are innumerable in the continuous space and we mask a video token by setting a video token with all zeros and ask the encoder to recover the video token. via noisy contrastive estimation (NCE):

$$\mathcal{L}_{\text{MFM}} = -\mathbb{E}_{s \sim V} \log \text{NCE}(x_s | \boldsymbol{x}_{\text{masked}}; V'), \tag{5}$$

where V is all indexes of video tokens and

$$\text{NCE}(x_v | \boldsymbol{x}_{\text{masked}}; V') = \frac{\exp(x_v^T e_v)}{\exp(x_v^T e_v) + \sum_{j \in V'} \exp(x_j^T e_v)}, \tag{6}$$

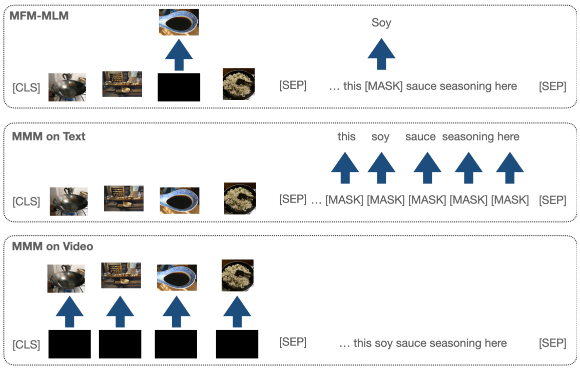

Figure 2: Task-agnostic pre-training (e.g. w/o task on retrieval-style alignment): MFM-MLM: 50% of training examples are masked as masked frame model (MFM) and masked language model (MLM); the rest 50% examples are masked as masked modality model (MMM) (25% on text as in the second row and 25% on video as in the third row).

where V' indicates all non-masked video tokens within the same batch. The final loss is the sum of both MFM and MLM:

$$\mathcal{L}_{\text{MFM-MLM}} = \mathcal{L}_{\text{MFM}} + \mathcal{L}_{\text{MLM}}, \qquad (7)$$

where $\mathcal{L}_{\text{MLM}}$ is the same as BERT and we omit its details for brevity. We experiment this classic baseline in Section 5.

3.3 MMM and Masked Token Loss

Masked Modality Model We introduce masked modality modal (MMM) that masking either all video or all text tokens out for a given example of video-text clip. This masking scheme complements MFM-MLM (e.g. in our experiments 50% of training examples are masked as MMM and the rest 50% are masked as MFM-MLM). This encourages the encoder to use tokens from one modality to recover the tokens for the other modality. This resolves the issue that an encoder may use nearby tokens from their modality for prediction just because tokens from a single modality are closer As in the lower two (2) sub-figures in Figure 2, we either mask the whole modality of video or text so this modality can be "generated" from the other modality. Our experiments indicate that this is critical for pre-training a single encoder for retrieval tasks.

Masked Token Loss We further introduce masked token loss that unifies loss functions for MFM and MLM. This loss encourages learning a joint token embedding space for video and text and both types of tokens contribute to the prediction of a masked (video or text) token. This also improves the number of contrasted negative embeddings in two separate losses for MFM and MLM.

We define masked token loss $\mathcal{L}_{\text{VLM}}$ as the following:

$$-\mathbb{E}_{s\sim V\cup D}\log \mathrm{NCE}(x_s|\boldsymbol{x}_{\text{masked}}; V'\cup D_{\setminus s}), \qquad (8)$$

where D is the word embeddings over the vocabulary of BERT and $D_{\setminus s}$ excludes token s (if s is a text token). Further, $\mathrm{NCE}(x_s|\boldsymbol{x}_{\text{masked}}; V'\cup D_{\setminus s})$ is defined as:

$$\frac{\exp(x_s^T e_s)}{\exp(x_s^T e_s)+\sum_{j\in V'\cup D_{\setminus s}}\exp(x_j^T e_s)}. \qquad (9)$$

Note that $j \in V'\cup D_{\setminus s}$ can be either a video or text token and one predicted token e_s must be closer to the ground-truth token embedding (either a video token or word embedding) and be away from other embeddings of video/text tokens. We perform an ablation study in Section 5 to show that $\mathcal{L}_{\text{VLM}}$ works better than $\mathcal{L}_{\text{MFM-MLM}}$.

4 Fine-tuning

In this section, we describe how to use different types of attention masks to fine-tune VLM for a variety of tasks, as shown in Figure 3.

4.1 Text-Video Retrieval

One major challenge of pre-training on a single encoder is how to adapt such a model to joint space retrieval without using unimodal encoders for task-specific pre-training on contrastive loss (as in Howto100M (Miech et al., 2019, 2020)). The main reason is that many existing models encode text and video tokens together via self-attention, and one cannot obtain hidden states for text/video alone.

To resolve this, we propose to apply an isolated attention mask with two squared masks that are diagonally placed, as shown in the lower sub-figure of the first box in Figure 3.[2] These two squares disable video and text tokens to attend and see each other, while still allow video and text tokens to use the same self-attention layers for learning representations in the same feature space. Further, note that the first and second `[SEP]` tokens of BERT will

---

[2] One can further reduce $O(m+n)^2$ complexity to $O(m^2+n^2)$ (m and n are lengths for video and text, respectively) by feeding video/text separately to BERT but we adopt squared masks for simplicity.

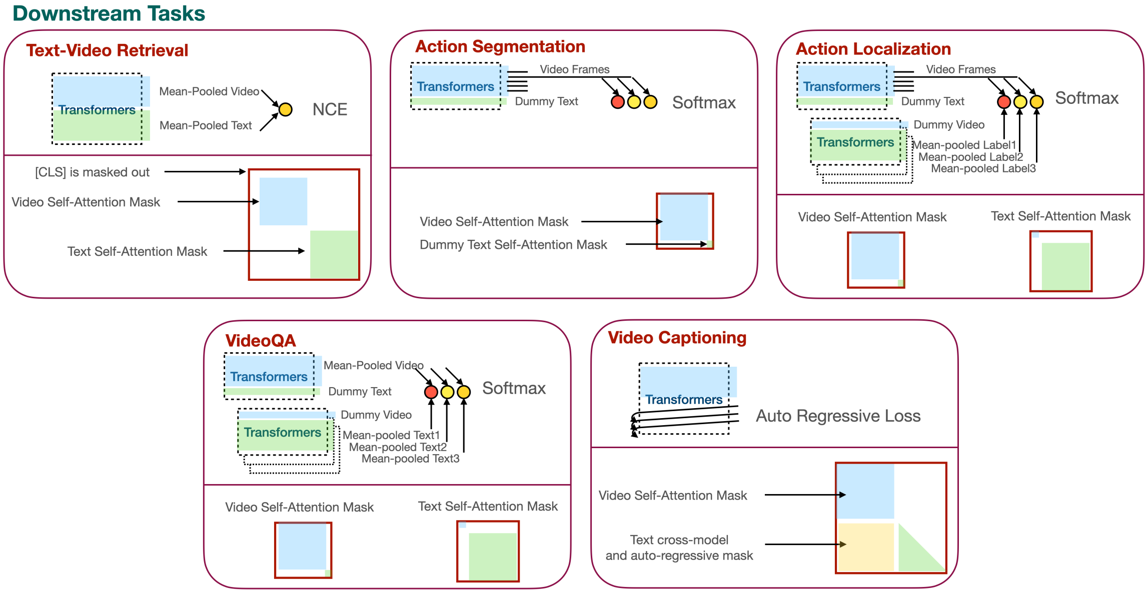

Figure 3: Fine-tuning of downstream tasks: we adopt different types of attention masks for BERT to accommodate downstream tasks that require different modalities: in each box, the upper sub-figure indicates a forward computation; the lower sub-figure indicates squared self-attention mask, where tokens from each row have a weighted sum of columns that are not in white colors.

be used by video and text, respectively, aiming to learn sequence-level features(Clark et al., 2019). The `[CLS]` is disabled as no need to learn features across video and text. After forwarding, all hidden states of video and text tokens are average pooled, respectively. Then we use a contrastive loss on text-video similarity to discriminate a ground-truth video clip from other video clips in the same batch for a given text clip. During the evaluation, to ensure video and text are isolated (to avoid leaking ground-truth of a similar pair), we split text and video and forward them separately. We report an ablation study in Section 5 showing that the MMM introduced in the previous section is crucial to ensure that the pre-trained hidden states (for video or text) are a good initialization for retrieval tasks.

4.2 Action Segmentation

Action segmentation is to assign each frame of a video with one of the pre-defined labels. This is similar to the named entity recognition (NER) task in NLP but on video frames. We feed in VLM with the whole video, a dummy text token, and an isolated attention mask. Then we add a classification head (with the number of pre-defined labels) on top of the hidden states for each video token in the last layer of VLM.

4.3 Action Step Localization

In action step localization, each video belongs to a task with multiple steps, where each step is described as a short text. Then each frame of a video needs to be aligned with a step in text form. The challenge for applying BERT to action step localization is similar to text-video retrieval: video frames need to be aligned with textual steps in joint space and it is almost impossible for pairwise video/text matching because the number of frame/text pairs is large.

Similar to the text-video retrieval model, we also apply isolated attention masks to video and text. The major difference is that we pass video and text separately to BERT. This is because the video can be several minutes long (more than 100 tokens) but the number of text labels for each video is fixed (e.g. under 10). To keep the format of BERT being consistent for multi-modal inputs, we add a dummy text token for video forwarding and a dummy video token for text, respectively. For a given frame(video token), we compute the distribution of that frame over textual steps via dot products and the softmax function.

4.4 Multiple-choice VideoQA

Multiple-choice VideoQA (Yu et al., 2018) aligns each video with one out of several candidate answers in the text. The major difference between action step localization and multiple-choice VideoQA is that the video hidden state is not on frame-level but sequence-level. We apply isolated attention masks to BERT and forward video and text answers (with dummy tokens), respectively. Then the answer with the maximum similarity with the video is reported. During fine-tuning, we apply contrastive loss on video-text similarity to rank answers.

4.5 Video Captioning

Another big challenge of using a single encoder is how to apply generative tasks (such as video captioning) without pre-training an explicit decoder. We observe that a transformer decoder (Vaswani et al., 2017) has the following major differences from an encoder: (1) an auto-regressive loss that does not allow a text token to see future tokens; (2) a prediction head to generate texts. To resolve (1), one can easily fine-tune the text segment of VLM as auto-regressive loss by passing in shifted tokens and a lower-triangle attention mask to the text segment, as shown in Figure 3. To resolve (2), inspired by (Rothe et al., 2020; Zhou et al., 2020) that uses BERT as a decoder, one can re-use language model heads as prediction heads for generation. Note that this setting has less architecture design than a standard transformer decoder (e.g. no explicit self-attention on text or cross-attention on video). The implicit text decoder inside BERT shares self-attention with the video encoder so to save the total number of parameters.

5 Experiment

5.1 Dataset

5.1.1 Pre-training

We adopt the Howto100M dataset (Miech et al., 2019) for pre-training, which contains instructional videos originally from YouTube via searching keywords from wikihow (`www.wikihow.com`). After filtering the unavailable ones, we get 1.1M videos. We split 4000 videos as the validation set and the rest for pre-training. On average, the duration of each video is about 6.5 minutes with 110 clip-text pairs. After removing repeated texts within overlapped clips from ASR, we get about 7.7+ GB texts of captions, with 2.4 tokens per second on average.

5.1.2 Fine-tuning

MSR-VTT (Xu et al., 2016) is a popular dataset for *text-video retrieval* and *VideoQA*. It has open domain video clips, and each training clip has 20 captioning sentences labeled by humans. There are 200K clip-text pairs from 10K videos in 20 categories, including sports, music, etc. Following JSFusion(Yu et al., 2018; Miech et al., 2019), we randomly sampled 1,000 clip-text pairs as test data. We further use the QA test data (Yu et al., 2018) as the dataset for multiple-choice VideoQA.

Youcook2 (Zhou et al., 2017) contains 2,000 cooking videos on 89 recipes with 14K video clips from YouTube. The overall duration is 176 hours (5.26 minutes on average). Each video clip is annotated with one captioning sentence. Follow the split setting in(Miech et al., 2019), we evaluate both text-based video retrieval and multimodal video captioning tasks. We filter the data and make sure there is no overlap between pre-training and evaluation data. After filtering out unavailable ones, we have 9,473 training clip-text pairs from 1222 videos and 3,305 test clip-text pairs from 430 videos.

COIN (Tang et al., 2019) are leveraged to evaluate *action segmentation*. It has 11,827 videos (476 hours) and each video is labeled with 3.91 step segments on average and 46,354 segments in total. There are 778 step labels, plus one background (Outside) label. Since one video can last for several minutes that are much longer than the maximum length of the video segment of VLM. We apply a sliding window with step size 16 and window size 32. During inference, we average the logits for overlapped frames from multiple windows.

CrossTask (Zhukov et al., 2019) is a dataset for action localization that contains 83 different tasks and 4.7k videos. Each task has a set of steps with text descriptions annotated on temporal frames of the video. We use the testing data split via the official code[3], which contains annotated 1690 videos. The rest of the 540 annotated videos are used for weakly supervised training.

5.2 Hyper-parameters

We extract video tokens from video frames using the S3D encoder pre-trained from (Miech et al., 2020). The fps is 30 and we extract one (1) video token per second with the dimension of 512. We apply an MLP to transform such 512 dimensions to the hidden size (768) of $\text{BERT}_{\text{BASE}}$.

Following (Luo et al., 2020), we adopt $\text{BERT}_{\text{BASE}}$ (uncased) as our base model and tuned directly from BERT's weights, so all hyper-parameters are the same as the original BERT. The maximum length of BERT is set as 96, where 32

---

[3] `https://github.com/DmZhukov/CrossTask`

Model	Paradigm	#params.	#loss	#unimodal/cross en/decoder	Joint Retrieval	Generation
MMT(Gabeur et al., 2020)	task-specific alignment	127.3M	1	2/0/0	yes	no
ActBERT(Zhu and Yang, 2020)	weakly supervised/MTL	n/a (3 typed attentions)	4	0/1(modal-typed attn.)/0	no(pair)	extra decoder
VideoAsMT(Korbar et al., 2020)	weakly supervised/MTL	286M(base)/801M(large)	1	1/1/1	no (gen.)	yes
HERO(Li et al., 2020b)	SSL(w/ sup. video feat.)/MTL	159M	5	1(query)/2/0	no(pair)	extra decoder
UniVL(Luo et al., 2020)	SSL/MTL	260M	5	2/1/1	yes	yes
VLM	SSL/Task-agnostic	**110M**	**1**	0/**1**/0(shared w/ encoder)	yes	yes

Table 1: Comparison of pre-trained models on learning paradigms (SSL means self-supervised learning; MTL means multi-task learning), number of parameters (# params.), number of losses (#loss), number of unimodal/cross-modal encoders/decoders, and whether to support retrieval in joint space(joint retrieval) and text generation. Types and numbers are estimated based on released code or papers: exceptions are in parenthesis (e.g. pair means pairwise matching using `[CLS]`). VLM is extremely simple with fewer parameters and limitations.

tokens are for videos and the rest tokens are for text and special tokens. Remind that texts are 2.4 tokens per second and video tokens are 1 token per second. We form a text clip with a random length in-between 8 and 64 text tokens and collect the corresponding video clip to form a training example. We randomly sample 32 video/text clip pairs from each video and use 8 videos to form a batch of size 256. Each training example has 50% chance for MMM (25% for whole video masking and 25% for whole text masking) and 50% chance on MFM-MLM (with 15% probability of video and text token masking).

We pre-train VLM on 8 NVIDIA Tesla V100 GPUs (each with 32 GB memory) for 15 epochs using fp16 for one (1) day. Following (Liu et al., 2019), we choose Adam (Kingma and Ba, 2014) optimizer with initial learning rate of 5e-5 (with betas as (0.9, 0.98)), 1000 steps of warm-up and a polynomial decay learning rate scheduler. Gradients are clipped with 2.0. All fine-tuning tasks use the same hyper-parameters as pre-training except the number of warm-up steps is 122.

5.3 Model Comparison

We first investigate the design choices of VLM compared to other transformer-based multimodal pre-training baselines. As shown in Table 1, we collect training paradigms, model sizes, etc. of these models (estimated based on their source codes or papers). VLM is significantly smaller than other models since it is just a BERT$_{\text{BASE}}$ (uncased), while it is still fully self-supervised, task-agnostic (e.g. no training on retrieval or auto-regressive style tasks) and supports joint retrieval and text generation.

Methods	R@1	R@5	R@10	Median R
Random	0.1	0.5	1.0	500
Task-specific Alignment Pre-training				
MMT (Gabeur et al., 2020)	25.8	57.2	69.3	4
Pairwise Matching				
ActBERT(Zhu and Yang, 2020)	8.6	23.4	33.1	36
VideoAsMT(Korbar et al., 2020)	14.7	-	52.8	-
Multi-task Pre-training				
HERO (Li et al., 2020b)	16.80	43.40	57.70	-
UniVL (FT-Joint) (Luo et al., 2020)	20.6	49.1	62.9	6
VLM	28.10	55.50	67.40	4

Table 2: Results of text-video retrieval on MSR-VTT dataset.

Methods	R@1	R@5	R@10	Median R
Random	0.03	0.15	0.3	1675
Task-specific Alignment Pre-training				
Coot(Ging et al., 2020)	16.7	40.2	52.3	9
Pairwise Matching				
ActBERT(Zhu and Yang, 2020)	9.6	26.7	38.0	19
VideoAsMT(Korbar et al., 2020)	11.6	-	43.9	-
Multi-task Pre-training				
UniVL (FT-Joint)(Luo et al., 2020)	22.2	52.2	66.2	5
VLM	27.05	56.88	69.38	4

Table 3: Results of text-based video retrieval on Youcook2 dataset.

5.4 Quantitative Analysis

We investigate the performance of VLM on fine-tuning tasks with very basic setups (e.g. no augmented features, large LMs, optimized losses for particular tasks). Note that it could be hard for fair comparisons between task-agnostic and task-specific approaches. We list other baselines by type and our goal is a simple baseline for task-agnostic pre-training as better initialization of strongly performed fine-tuning models.

Text-video Retrieval We use MSR-VTT and Youcook2 to evaluate the performance on text-video retrieval. The results are shown in Table 2 and 3, respectively. VLM achieves good performance on these two datasets, indicating that the MMM and isolated self-attention mask can be used together for joint retrieval. Ablation study shows that using an isolated self-attention mask alone does not yield good performance, indicating MMM is very important to learn features for alignment. Note that our pre-training is task-agnostic but still outperforms baselines with retrieval style pre-training.

Action Segmentation We report the results of action segmentation on COIN dataset in Table 4.

Method	Frame Accuracy
NN-Viterbi (Richard et al., 2018)	21.17
VGG (Simonyan and Zisserman, 2014)	25.79
TCFPN-ISBA (Ding and Xu, 2018)	34.30
CBT (Sun et al., 2019a)	53.90
MIL-NCE (Miech et al., 2020)	61.00
ActBERT (Zhu and Yang, 2020)	56.95
VLM	68.39

Table 4: Action segmentation on COIN dataset.

Methods	Average Recall
Joint Alignment	
Alayrac (Alayrac et al., 2016)	13.3
Zhukov (Zhukov et al., 2019)	22.4
Supervised (Zhukov et al., 2019)	31.6
HowTo100M (Miech et al., 2019)	33.6
MIL-NCE (Miech et al., 2020)	40.5
UniVL (Luo et al., 2020)	42.0
Pairwise Matching	
ActBERT (Zhu and Yang, 2020)	41.4
VLM (task-agnostic, zero-shot)	28.5
VLM (supervised on 540 videos)	46.5

Table 5: Action step localization results on CrossTask.

Method	Accuracy
Joint Retrieval	
JSFusion(Yu et al., 2018)	83.4
Pairwise Matching	
ActBERT(Zhu and Yang, 2020)	85.7
VLM	91.64

Table 6: Video question answering (multiple-choices) evaluated on MSR-VTT.

Methods	B-3	B-4	M	R-L	CIDEr
Extra Decoder					
VideoBERT (Sun et al., 2019b)	6.80	4.04	11.01	27.50	0.49
CBT (Sun et al., 2019a)	-	5.12	12.97	30.44	0.64
ActBERT (?)	8.66	5.41	13.30	30.56	0.65
Coot(Ging et al., 2020)	17.62	11.09	19.34	37.63	-
w/ Pre-trained Decoder					
VideoAsMT (Korbar et al., 2020)	-	5.3	13.4	-	-
UniVL (Luo et al., 2020)	16.46	11.17	17.57	40.09	1.27
VLM	17.78	12.27	18.22	41.51	1.3869

Table 7: Video captioning results on Youcook2 dataset.

VLM outperforms other baselines indicating its good token-level video representations. Note that this task only tests the hidden states of the video indicating the unimodal encoding capability of VLM is not compromised.

Action Step Localization We setup two (2) evaluations for the CrossTask dataset. First, we evaluate the zero-shot transfer of VLM. Note that existing studies evaluate Crosstask with retrieval/alignment style pre-training, where the aligned hidden states are directly used for action step localization. Our task-agnostic pre-training derives an even harder problem: applying hidden states learned from proxy tasks on video frame/text alignment for action step localization without explicitly training on alignment. We simply use the hidden states from the last layer of VLM for video/text representation and directly compute the similarities between video frames and text descriptions. Surprisingly, the performance is better than some baselines and closer to one supervised method. This indicates masked token loss together with MMM can learn certain video-text alignments in joint space. Second, we use just 540 videos for weakly supervised training and we get a much better result.

Video Question Answering We use MSR-VTT QA to evaluate multiple-choice question answering. Recall that this task essentially tests video-text similarity. The performance of VLM is better than ActBERT, which leverages pairwise matching for each video/answer pair.

Video Captioning We lastly evaluate VLM on video captioning with autoregressive attention mask with other baselines that have an explicit text decoder. As shown in Table 7, our "compact" decoder using BERT's LM heads is surprisingly good at video captioning compared to other fine-tuning baselines with external decoders (e.g. Coot). This indicates that it is possible to remove an explicit decoder and sharing weights between video and text tokens.

5.4.1 Ablation Study

We use Youcook2 as the base task for the ablation study on text-retrieval and video captioning. We are interested in the following study: (1) percentage of examples for MMM (w/ MMM x%); (2) minimum length of text tokens, where the length of video will be determined by the start/end timestamps of text tokens; (3) performance of $\mathcal{L}_{\text{VLM}}$ (Equation 8). The results are shown in Table 8 and Table 9.

Effects of MMM Without MMM (w/ MMM 0%, or MFM-MLM 100%), the performance significantly dropped. This indicates that a naive adoption of traditional MFM-MLM masking may not learn joint video/text representations well, as indicated by both retrieval and captioning task. We suspect a masked token is more likely predicted from tokens of the same modality. We further try MMM with different probabilities (30% or 70%) and 50% is the best.

Minimum Length of Texts The length of a clip can be important for retrieval tasks (Miech et al., 2020). We ran VLM on longer (at least 16 text tokens) video/text pairs. The performance is slightly dropped, indicating pre-training on longer clips may not cover fine-tuning tasks with short clips.

Effects of Masked Token Loss We notice that using multi-task style loss $\mathcal{L}_{\text{MFM-MLM}}$ may reduce the performance. This indicates learning a masked token from both video/text tokens can help.

VLM	R@1	R@5	R@10	Median R
w/ MMM 50%	27.05	56.88	69.38	4.0
w/ MMM 0%	15.12	39.47	52.81	9.0
w/ MMM 30%	25.30	54.80	68.96	4.0
w/ MMM 70%	25.17	54.98	69.11	4.0
w/ min. 16 text tokens	25.84	54.43	68.29	5.0
w/ $\mathcal{L}_{\text{MFM-MLM}}$	26.93	55.92	69.86	4.0

Table 8: Ablation study of VLM for text-based video retrieval on Youcook2.

VLM	B-3	B-4	M	R-L	CIDEr
w/ MMM 50%	17.78	12.27	18.22	41.51	1.3869
w/ MMM 0%	15.47	10.54	16.49	38.83	1.2163
w/ MMM 30%	16.57	11.30	17.55	40.76	1.3215
w/ MMM 70%	16.94	11.68	17.67	41.24	1.3739
w/ min. 16 text tokens	17.25	12.00	17.67	40.62	1.3076
w/ $\mathcal{L}_{\text{MFM-MLM}}$	16.66	11.53	17.34	40.36	1.3224

Table 9: Ablation study of VLM for video captioning on Youcook2 dataset.

5.5 Qualitative Analysis

5.5.1 Error Analysis

Text-video retrieval. We use MSR-VTT as the dataset for error analysis on text-video retrieval, as shown in Table 10 of Appendix. We pair the query text with the text of the top-1 ranked video to show 100 errors in ranking since video tokens are harder to present. We observe the following types of errors in video understanding: (1) objects sometimes are hard to recognize such as dog or cat; (2) attributes of objects may be hard to match the text, e.g. gender, ages, etc. (3) subtle differences of actions; (4) specific videos for a general query or vice versa, e.g. people vs basketball player. We believe the last type may not be errors but hard for existing annotations or evaluations to separate.

Video Captioning. We further examine the generated text from video captioning. Note that our video captioning has no support from ASR or transcript so the video is the only source to generate text content and errors of video understanding can easily be reflected in the text. From Table 11 of Appendix, we notice that one major type of error is from objects of similar shapes and colors, e.g. onion rings vs shrimp.

5.5.2 Visualization

. We observe that video tokens take the majority of space while text tokens are rather clustered together. This is probably because videos from the physical world are more diverse and sparse than text from a fixed vocabulary.

We plot the self-attention of VLM layers within and in-between each modality, as in Figure 4 of Appendix. We observe the following patterns from all 144 attention heads:

- Unlike LMs, there are no recurrent (shifted) position-wise patterns for video tokens;
- Self-attentions in the 1st layer are more diverse than later layers. This suggests that existing video encoders might be too deep for transformers;
- Some attention heads show patterns of cross-modal mapping in-between video and text (e.g. sub-figure (a));
- Word-level cross-modal co-reference: video tokens with *pouring soy sauce* refers to the text token of "soy" (e.g. sub-figure (b));

6 Conclusions

We presented a task-agnostic pre-training with new masking schemes that enable the training of a single masked language model that can accept either video or text input, or both. We showed that this simple VLM model can be effectively tuned for a broad range of downstream tasks, such as text-video retrieval and video captioning via different types of attention masks. Experimental results show that the proposed methods maintain competitive performance while requiring a significantly smaller number of parameters than competing methods.

Acknowledgments

We thank Huaishao Luo (author of UniVL(Luo et al., 2020)), Mandela Patrick (author of Support-Set(Patrick et al., 2021)) and Luowei Zhou (author of Youcook(Zhou et al., 2017)) for supports of baseline setup.

References

Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. 2016. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583.

Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. 2020. Self-supervised multimodal versatile networks. *arXiv preprint arXiv:2006.16228*.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*, pages 104–120. Springer.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Ding and Chenliang Xu. 2018. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6508–6516.

Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. 2020. Multi-modal transformer for video retrieval. In *European Conference on Computer Vision (ECCV)*, volume 5. Springer.

Simon Ging, Mohammadreza Zolfaghari, Hamed Pirsiavash, and Thomas Brox. 2020. Coot: Cooperative hierarchical transformer for video-text representation learning. *arXiv preprint arXiv:2011.00597*.

Po-Yao Huang, Mandela Patrick, Junjie Hu, Graham Neubig, Florian Metze, and Alexander G. Hauptmann. 2021. Multilingual multimodal pretraining for zero-shot cross-lingual transfer of vision-language models. *CoRR*, abs/2103.08849.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bruno Korbar, Fabio Petroni, Rohit Girdhar, and Lorenzo Torresani. 2020. Video understanding as machine translation. *arXiv preprint arXiv:2006.07203*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, Daxin Jiang, and Ming Zhou. 2020a. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *AAAI*, pages 11336–11344.

Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. 2020b. HERO: Hierarchical encoder for Video+Language omni-representation pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2046–2065, Online. Association for Computational Linguistics.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. In *Arxiv*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, volume 32, pages 13–23. Curran Associates, Inc.

Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Xilin Chen, and Ming Zhou. 2020. Univilm: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2020. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE international conference on computer vision*, pages 2630–2640.

Mandela Patrick, Po-Yao Huang, Yuki Asano, Florian Metze, Alexander G Hauptmann, Joao F. Henriques, and Andrea Vedaldi. 2021. Support-set bottlenecks for video-text representation learning. In *International Conference on Learning Representations*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. 2018. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7386–7395.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Andrew Rouditchenko, Angie Boggust, David Harwath, Dhiraj Joshi, Samuel Thomas, Kartik Audhkhasi, Rogerio Feris, Brian Kingsbury, Michael Picheny, Antonio Torralba, et al. 2020. Avlnet: Learning audio-visual language representations from instructional videos. *arXiv preprint arXiv:2006.09199*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. Vl-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*.

Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. 2019a. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 3(5).

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019b. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7464–7473.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1207–1216.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296.

Youngjae Yu, Jongseok Kim, and Gunhee Kim. 2018. A joint sequence fusion model for video question answering and retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 471–487.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and vqa. In *AAAI*, pages 13041–13049.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2017. Towards automatic learning of procedures from web instructional videos. *arXiv preprint arXiv:1703.09788*.

Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. 2018. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748.

Linchao Zhu and Yi Yang. 2020. Actbert: Learning global-local video-text representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8746–8755.

Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. 2019. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3537–3545.

Query	Text of Top-1 video
Objects (26%)	
cartoon show for kids	**pokemon video game** play
little pet shop **cat** getting a bath and washed with little brush	several **dogs** playing dead
Attributes of Objects (6%)	
a little **boy** singing in front of judges and crowd	a **woman** singing on the voice
a woman is mixing **food** in a mixing bowl	a man is stirring **something** in a pot
Action (6%)	
a person is **connecting something** to system	a man **looks at** the **battery** of a computer
a boy **plays** grand theft auto 5	a narrator **explains where to find** a rare vehicle in grand theft auto
a man is **giving a review** on a vehicle	a person is **discussing** a car
a **naked child** runs through a field	the **girl** shows the **boys** her medal in this cartoon
a man is singing and standing in the road	a man in sunglasses and a blue shirt beat boxes
Specific vs General (62%)	
some cartoon characters are moving around an area	a cartoon girl and animal jumping on body of male guy girl image still shown displaying on screen
baseball player hits ball	**people** are playing baseball
the man in the video is showing a brief viewing of how the movie is starting	scrolling the the menu of movieclips with different movie trailers
a **student** explains to his **teacher** about the sheep of another student	there is a **guy** talking to his **father**
a video about different sports	a woman talks about horse racing

Table 10: Error analysis for text-video retrieval of MSR-VTT on 100 errors: we group errors in four (4) categories: objects, attributes of objects, actions, and specific vs general. Specific videos for general queries (or vice versa) sometimes may not be errors but hard to evaluate.

Hypothesis	Reference
add the lamb to the **pan**	add the lamb to the **pot**
add the cilantro ~~cilantro~~ and lime juice to the pot	cut the cilantro and lime
add the **onions** to a pot of water	add **flour** to the pot and stir
dip the **onion rings** into the batter	dip the **shrimp** in the batter
add water to the bowl and mix	pour water into the **flour mixture** and mix
remove the **mussels** from the pot	once the **shrimps** are defrosted drain the water
add the sauce to the **pan** and stir	add the sauce to the **wok** and stir
add lemon juice to the pan and stir	add **rice vinegar** and lemon juice to the pan and stir
add the beef to the pan and stir	add the diced beef meat to it and roast it

Table 11: Error analysis for video captioning on Youcook2: VLM tends to make mistakes in recognizing objects of similar shapes and colors to generate the wrong text.

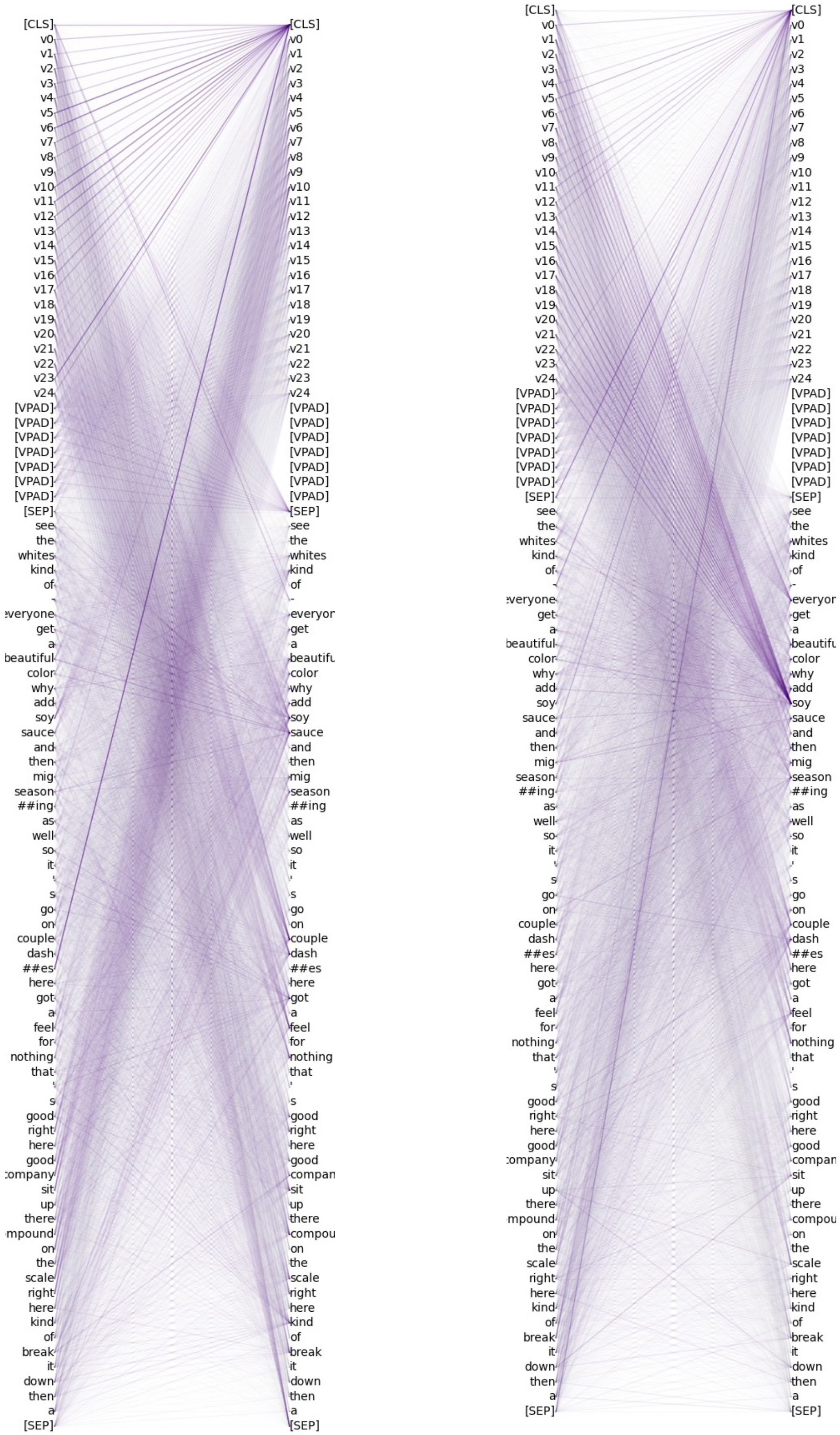

(a) Layer 1, Head 1

(b) Layer 1, Head 5

Figure 4: Self-attention for video HfIeQ9pzL5U from 4:03 to 4:28: darker color indicates higher weights; v0-v24 are video tokens of 25 seconds.

Corpus-Level Evaluation for Event QA:
The IndiaPoliceEvents Corpus Covering the 2002 Gujarat Violence

Andrew Halterman[*]
Massachusetts Institute of Technology
ahalt@mit.edu

Katherine A. Keith[*]
University of Massachusetts Amherst
kkeith@cs.umass.edu

Sheikh Muhammad Sarwar[*]
University of Massachusetts Amherst
smsarwar@cs.umass.edu

Brendan O'Connor
University of Massachusetts Amherst
brenocon@cs.umass.edu

Abstract

Automated event extraction in social science applications often requires corpus-level evaluations: for example, aggregating text predictions across metadata and unbiased estimates of recall. We combine corpus-level evaluation requirements with a real-world, social science setting and introduce the IndiaPoliceEvents corpus—all 21,391 sentences from 1,257 English-language *Times of India* articles about events in the state of Gujarat during March 2002. Our trained annotators read and label every document for mentions of police activity events, allowing for unbiased recall evaluations. In contrast to other datasets with structured event representations, we gather annotations by posing natural questions, and evaluate off-the-shelf models for three different tasks: sentence classification, document ranking, and temporal aggregation of target events. We present baseline results from zero-shot BERT-based models fine-tuned on natural language inference and passage retrieval tasks. Our novel corpus-level evaluations and annotation approach can guide creation of similar social-science-oriented resources in the future.

1 Introduction

Understanding the actions taken by political actors is at the heart of political science research: How do actors respond to contested elections (Daxecker et al., 2019)? How many people attend protests (Chenoweth and Lewis, 2013)? Which religious groups are engaged in violence (Brathwaite and Park, 2018)? Why do some governments try to prevent anti-minority riots while others do not (Wilkinson, 2006)? In the absence of official records, social scientists often turn to news data to extract the actions of actors and surrounding events. These news-based event datasets are often constructed by hand, requiring large investments of time and money and limiting the number of researchers who can undertake data collection efforts.

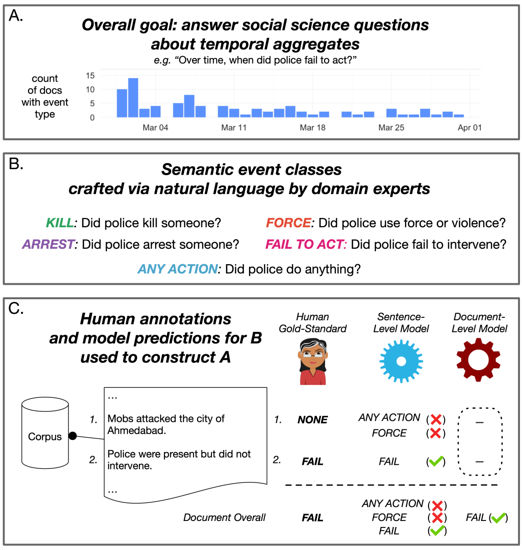

Figure 1: Motivation (A-B) and procedures (B-C) for this paper: **A.** Social scientists often use text data to answer substantive questions about temporal aggregates. **B.** To answer these questions, domain experts use natural language to define semantic event classes of interest. **C.** Our IndiaPoliceEvents dataset: Humans annotate *every* sentence in the corpus in order to evaluate whether a system achieves full recall of relevant events. In production, computational models run B's queries to classify or rank sentences or documents, which are aggregated to answer A.

Automated extraction of political events and actors is already prominent in social science (Schrodt et al., 1994; King and Lowe, 2003; Hanna, 2014; Hammond and Weidmann, 2014; Boschee et al., 2015; Beieler et al., 2016; Osorio and Reyes, 2017) and is increasingly promising given recent gains in information extraction (IE), the automatic conversion of unstructured text to structured datasets (Grishman, 1997; McCallum, 2005; Grishman, 2019). While social scientists and IE researchers have over-

---

[*] Indicates joint first-authorship.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4240–4253
August 1–6, 2021.

lapping interests in evaluating event extraction systems, social scientists have particular needs that have so far been under-addressed by the computer science IE research community.

Figure 1A shows a common goal of social scientists: answering aggregate substantive questions from corpora such as, *"Over time, when did police fail to act?"* which could be measured by, for example, the daily count of newspapers mentioning the event class over time. For these types of questions, social scientists predominantly want very high recall methods because often the events of interest are sparse or their substantive conclusions depend on identifying *every* event in a corpus.[1]

In contrast to this corpus-level focus, much of current IE research has focused on distinct subtasks such as entity linking, relation extraction, and coreference resolution.[2] Furthermore, all widely used event datasets (e.g. ACE, FrameNet, ERE, or KBP; Aguilar et al. 2014) are typically curated at the *ontology* level—attempting to cover a selected set of event types—but have little consideration of the *corpus* level—annotated documents are not necessarily a substantively meaningful sample of the broader corpora from which they are drawn. We try to address these evaluation shortcomings in this paper.

In addition to corpus-level recall, social scientists are often interested in using off-the-shelf models that are easily extensible to their domain questions. Fortunately, recent NLP research has seen a paradigm shift from structured semantic and event representations (Abend and Rappoport, 2017; Aguilar et al., 2014) which are limited by their predefined schemas, to directly using natural language to encode semantic arguments (*QA-SRL*; He et al. 2015; Stanovsky et al. 2016; FitzGerald et al. 2018; Roit et al. 2020) and events (Levy et al., 2017; Liu et al., 2020; Du and Cardie, 2020). In this paper, we also use natural language questions to annotate and model the event classes in our dataset, not only facilitating ease of annotation, but also allowing for the evaluation of zero-shot natural language inference and information retrieval models for the tasks.

To address these social science desiderata, we present the INDIAPOLICEEVENTS corpus[3] which has the following useful properties:

- **Social science relevance.** Our dataset consists of all 21,391 sentences from all 1,257 *Times of India* articles about events in the state of Gujarat during March, 2002—a period that is of deep interest to political scientists due to widespread Hindu–Muslim violence (Dhattiwala and Biggs, 2012; Berenschot, 2012; Basu, 2015). We focus on the actions of a single entity type, *police*, because of extensive substantive research on police actions during the Gujarat violence (Varadarajan, 2002). Our choice of location, actors, and event types are motivated by Wilkinson (2006)—political science work which created a hand-coded event dataset from newspapers about communal violence events in India from 1950-1995.
- **Corpus-level full-recall.** Unlike most previous event evaluation datasets, our annotators read *every* document in our corpus (that match a loose spatiotemporal filter; §4.1). This requires substantially more annotation work compared to a more targeted filter to select documents to annotate (e.g. matching via keywords), but eliminates a potential source of evaluation bias compared to alternative document retrieval data collection approaches (Grossman et al., 2016), and allows for full-recall evaluation of end-to-end event extraction systems.
- **Document-level context.** Our annotators read the context of an entire document to provide answers for each question on each sentence. We then aggregate these sentence-level answers to make document-level inferences. This allows us to accurately label sentences with anaphora or context-specific meaning.
- **Natural language event specification and zero-shot model evaluation.** In constructing our dataset, we gather annotations via a natural question-answer format because it allows for easily specifying constraints on arguments (e.g. *police* being the agent). Additionally, it allows for specifying event predicates not covered within the ontologies of current structured semantic representations, or with additional hard-to-specify

---

[1] In some studies, researchers rely on an assumption that events are missing at random, but others depend on knowing whether an event occurred at least once.

[2] The first five Message Understanding Conferences (MUC) required participants to submit complete systems to fill event templates; however, starting with MUC-6 and subsequent ACE and KBP tasks, information extraction was broken into distinct modules (Grishman and Sundheim, 1996; Grishman, 2019).

[3] Dataset, source code, and appendix are provided at http://slanglab.cs.umass.edu/IndiaPoliceEvents/ and https://github.com/slanglab/IndiaPoliceEvents.

semantic phenomena—e.g. "Did police fail to act?" or when political actors do *not* take an action, which is very important to political scientists (e.g. Wilkinson (2006)). This format also allows us to evaluate zero-shot natural language inference and information retrieval models.

- **High-quality annotators who provide uncertainty explanations.** We hire and train political science undergraduate students as annotators to ensure quality control, retraining annotators over a period of several months with training videos, two hour-long live meetings, and individual annotator feedback before producing our final dataset. Our annotators also provide free-text explanations for instances in which they are uncertain about the answer. These rationales are important given the recent attention to propagating annotator uncertainty in downstream NLP tasks (Dumitrache et al., 2018; Paun et al., 2018; Pavlick and Kwiatkowski, 2019; Keith et al., 2020) and social scientists' interest in quantifying uncertainty (King, 1989; Wallach, 2018).

In the remainder of this paper, we use our dataset for three levels of evaluation: sentence-level classification, ranking of documents to reduce manual reading time, and constructing temporal aggregates useful to social scientists (§3). We describe in detail our annotation and dataset creation process (§4), provide baseline models (§5), and evaluate their performance on all three tasks (§6).

2 Related Work

NLP and IR for police activity. Natural Language Processing (NLP) and Information Retrieval (IR) have been used for analysis of other police activity such as identifying victims of police fatalities from news articles (Keith et al., 2017; Nguyen and Nguyen, 2018; Sarwar and Allan, 2019); extracting eye-witness event types from Twitter including police activity and shootings (Doggett and Cantarero, 2016); detecting dialogue acts from police stops (Prabhakaran et al., 2018); and computational analysis of degree of respect in police officers' language (Voigt et al., 2017).

Political event extraction. Automated event extraction in social science is generally performed using dictionary methods and a set of substantively motivated event types and actor categories (Schrodt et al., 1994; Gerner et al., 2002; Beieler et al., 2016; Boschee, 2016; Radford, 2016; Brathwaite and Park, 2018; Liang et al., 2018). Other work uses supervised learning to infer events such as conflict or cooperation (Beieler, 2016) and protests (Hanna, 2017). While some have attempted to induce event types without supervision (O'Connor et al., 2013; Huang et al., 2016), most social science applications of event extraction require substantial human input either through constructing keyword lists, or annotating texts to train classifiers.

Recall-focused IR. TREC's total-recall track (Grossman et al., 2016) is inspired by real-world recall-focused applications from law, medicine, and oversight (McDonald et al., 2018). However, the track's datasets are not typically focused on events and assume documents are collected through interacting with a system. Other work has focused on methods for truncating ranked lists that minimize the risk of viewing non-relevant documents (Arampatzis et al., 2009; Lien et al., 2019), but this line of work does not evaluate on semantic retrieval of event classes.

3 Three Levels of Tasks

In order to answer substantive social science questions, for example *"Does variation in party control of state government affect whether police failed to intervene in ethnic conflict?"* (Wilkinson, 2006), social scientists often need to gather counts of events (e.g. "police failed to intervene") from text when official government records are lacking. Ideally, a social scientist could use automatic information extraction methods (Cowie and Lehnert, 1996; McCallum, 2005; Grishman, 2019) to transform unstructured text into a structured database that would be useful in a quantitative analysis. Yet, even state-of-the-art information extraction systems often give less than perfect accuracy, so social scientists must still manually analyze large portions of their corpus in order to extract events of interest. This quantitative research process motivates the following three tasks which our dataset can be used to evaluate:

Task 1: Sentence classification. Although social science corpora typically consist of *documents*, it would be useful for a system to classify *sentences* that contain events of interest.[4] Highlighting relevant sentences could, for semi-automated systems, reduce a social scientist's reading time, and, for fully-automated systems, provide sentence-level evidence of the automated method's *validity*, a cru-

---

[4] This is closely related to extracting "explanation representations" (Thayaparan et al., 2020), "supporting facts" (Yang et al., 2018b) or "evidence sentences" (Wang et al., 2019) in the machine reading comprehension literature.

cial aspect of research in text-as-data (Grimmer and Stewart, 2013) and the broader social sciences (Drost et al., 2011). INDIAPOLICEEVENTS allows for evaluation of sentence-level precision, recall, and F1 (§6.1).

Task 2: Document ranking. For semi-automated systems, social scientists must navigate the tradeoff between recall and manual reading time. Social scientists may rely on IR methods which present ranked lists of relevant documents (Baeza-Yates et al., 1999; Schütze et al., 2008). However, our informal interviews with social scientists suggest they want to know at what point they have read enough documents to achieve very high (95–100%) recall. In creating INDIAPOLICEEVENTS, annotators read every single sentence in a corpus which allows for full evaluations of average precision and our newly proposed metric: the proportion of the corpus that would have to be read to achieve Recall=X (PropRead@RecallX) (§6.2).[5]

Task 3: Substantive temporal aggregates. For social scientists, NLP and IR methods are used in service of answering substantive questions from text. In addressing our running example *"Did differences in party control of state government affect whether police failed to intervene in ethnic conflict?"* a social scientist could measure how many news articles[6] discuss "police failing to intervene" each day for a given temporal span. In this setting, it would be helpful to know if changes in model performance at the sentence or document level resulted in significant differences at this aggregate level. We design INDIAPOLICEEVENTS with the capability of evaluating these meaningful corpus-level temporal aggregates, such as the mean absolute error and Spearman rank correlation coefficient between per-day event counts of computational models and ground truth annotations (§6.3).

4 Annotations and Dataset

4.1 Corpus selection

We curate our corpus with a substantively motivated specification: it is restricted to a single authoritative news source, over a defined span of time, with articles that mention one of two locations involved in or related to the 2002 Gujarat violence.

Event Class	Pos. Sents.		Pos. Docs.	
KILL	96	(0.45%)	50	(3.98%)
ARREST	299	(1.40%)	128	(10.17%)
FAIL TO ACT	207	(0.97%)	114	(9.05%)
FORCE	222	(1.04%)	90	(7.15%)
ANY ACTION	2,073	(9.69%)	457	(36.24%)

Table 1: INDIAPOLICEEVENTS number and percentage of positive sentences (sents.) and documents (docs.) after the adjudication round. In total, the dataset contains 21,391 sentences and 1,257 documents.

From the website of *Times of India*, an English language newspaper of record in India, we first download all news articles published in March 2002.[7] During this period, widespread communal violence occurred in India, following the death of 59 Hindu pilgrims in a train fire in the state of Gujarat. In the subsequent months, reprisal attacks were directed at mostly Muslim victims across the state (Human Rights Watch, 2002; Subramanian, 2007). In creating our annotations, we specifically focus on the actions of police during these events, since a large body of evidence points to the importance of police intervention and non-intervention in quelling or permitting ethnic violence (Human Rights Watch, 2002; Wilkinson, 2006; Subramanian, 2007). We focus on the first month of the violence in order to fit within our annotation budget. This month saw the greatest levels of violence, though violence continued for a period of months afterward.

Our final corpus consists of the subset of scraped documents published in March 2002 that include either the name of the state (*Gujarat*) or a city related to the beginning of violence (*Ayodhya*).[8] Selecting on geographical and temporal metadata is a high recall way to filter the corpus without biasing the dataset by filtering to topic or event-related keywords, thus giving a better view of the true recall of an event extraction method.

---

[5] We do not address the problem of estimating recall when gold-standard labels are only known for the subset of documents read so far, but INDIAPOLICEEVENTS could be used to evaluate that task in future work.

[6] Count of news articles are often used in social science as a proxy for the true measure of the event, e.g. Nielsen (2013); Chadefaux (2014).

[7] §9 discusses copyright issues.

[8] Selecting documents using location-based keywords is a standard first step in political science text analysis (Mueller and Rauh, 2017). This filters to 18% of the total articles in March 2002. The precipitating event for the March 2002 violence was the burning of a train of pilgrims returning from Ayodhya.

4.2 Annotations via natural language

To collect annotations, we give annotators an entire document for context, and then ask them *natural language questions* about semantic event classes anchored on the actions of police for each sentence in that document:

- KILL: "Did police kill someone?" Lethal police violence is an important subject for social scientists (Subramanian, 2007). Example sentence: *"In Vadodara, one person was killed in police firing on a mob in the Fatehganj area."*
- ARREST: "Did police arrest someone?" Knowing when and where police made arrests and who was arrested is an important part of understanding police response to communal violence. Example sentence: *"Police officials said nearly 2,537 people have so far been rounded up in the state."*
- FAIL TO ACT: "Did police fail to intervene?" In the 2002 Gujarat violence, police were often accused of failing to prevent violence or allowing it to happen. Knowing when police were present but did not act is important for understanding the extent of this phenomenon and its potential causes (Wilkinson, 2006). Example sentence: *"The news items [...] suggest inaction by the police force [...] to deal with this situation."*
- FORCE: "Did police use force or violence?" Political scientists are interested not only when police kill but the level of force they use. Example sentence: *"Trouble broke out in Halad [...] where the police had to open fire at a violent mob."*
- ANY ACTION: "Did police do anything?" We collect annotations on all police activities, so that social scientists could, in the future, label more fine-grained event classes. Example sentence: *"In the heart of the city's Golwad area, the army is maintaining a vigil over mounting tension following [...]"*

Figure 2 shows the interface annotators see.[9] See Appendix §A for exact annotation instructions and per-question agreement rates.

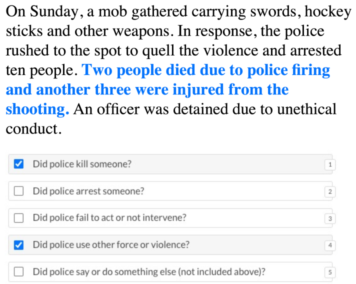

On Sunday, a mob gathered carrying swords, hockey sticks and other weapons. In response, the police rushed to the spot to quell the violence and arrested ten people. **Two people died due to police firing and another three were injured from the shooting.** An officer was detained due to unethical conduct.

☑ Did police kill someone? 1

☐ Did police arrest someone? 2

☐ Did police fail to act or not intervene? 3

☑ Did police use other force or violence? 4

☐ Did police say or do something else (not included above)? 5

Figure 2: We present annotators with a highlighted sentence (blue) and its document context. Their task is to click a check-mark for the event-focused questions for which there is a positive answer in the highlighted sentence.

Following the guidelines of Pustejovsky and Stubbs (2012), we first assign each document to two annotators and then follow with an *adjudication round* in which items with disagreement are given to an additional annotator to resolve and create the gold standard. For annotators, we select undergraduate students majoring in political science (as opposed to crowdworkers) in order to approximate the domain expertise of social scientists.[10] We initially recruited and selected 12 students. After a pilot study and two rounds of training, in which we provided individual feedback to annotators via email, we selected 8 final annotators based on their performance. Each student annotated around 330 documents ($\sim$5,500 sentences) using the interface described in the Appendix, Figure A2.

Table 1 shows the prevalence of the event classes after the adjudication round. Note that some of the classes are relatively rare: of all documents, only roughly 4% have KILL and 7% have FORCE. Our annotators had fairly high inner-annotator agreement for KILL and ARREST, with Krippendorff's alpha values of 0.75 and 0.71 respectively. Other questions, such as FAIL TO ACT and "Did police use other force?" had lower agreement ($\alpha < 0.4$), indicating more difficulty and ambiguity. Full agreement rates are show in Appendix, Table A1.

4.3 Annotation uncertainty explanations

We also collect free-text *annotation uncertainty explanations* in order to analyze instances that an-

---

[9] While the first three classes each correspond to a single annotation question, we create FORCE and ANY ACTION by taking the union of several different questions posed to annotators, which made it easier for annotators to distinguish between different subtypes. FORCE is the union of "Did police kill someone?" and "Did police use other force or violence?". ANY ACTION is the union of four questions: "Did police kill someone?", "Did police arrest someone?", "Did police use other force or violence?", and "Did police do or say something else (not included above)?".

[10] Our annotation protocol (no. 2238) was reviewed as exempt by the University of Massachusetts Amherst's IRB office. Annotators were paid $25 per training session and a lump sum for document annotations; we expected this to exceed $14 USD per hour based on a generous (conservatively high) estimate of completion time. All annotators reported their work time was less than this estimate.

notators found difficult or ambiguous. For each sentence presented to annotators, we ask "If you found this example difficult or ambiguous please explain why" and ask them to provide a short written response in a provided text-box. This follows recent work that that has emphasized the importance of annotator disagreement not necessarily always as error in annotation but instead as ambiguity that is inherent to natural language and a potential useful signal for downstream analyses (Dumitrache et al., 2018; Paun et al., 2018; Pavlick and Kwiatkowski, 2019; Keith et al., 2020).

Annotators remarked on several types of text they were uncertain about: agents of actions who were not explicitly mentioned but implicitly police, named entities whose status as police is ambiguous, confusion about what precisely constitutes an "arrest", and confusion arising from the lack of specific cultural knowledge (e.g., around the Indian crowd-control tactic of "lathi charging"). In the appendix, see Table A3 for examples and Table A2 for a categorization of free text responses.

5 Baseline Models

We test several baseline models, all requiring no annotation (and thus most realistic for the social science use case), and assess their performance on INDIAPOLICEEVENTS.

Keyword matching. Boolean keyword queries are a very common social science approach to document classification (e.g. Nielsen (2013); Chadefaux (2014); D'Orazio et al. (2014); Baker et al. (2016)), since they are simple, transparent, and widely supported in user software. We use conjunctive normal form rules, where inferring an event class for a sentence requires matching any term from a *police* keyword list (including both common nouns and names of major police and security institutions), as well as an event keyword. To construct the keyword lists, a domain expert coauthor first manually generates a list of seed keywords for the semantic categories *police*, *kill*, *arrest*, *intervention*, and *force*. To address lexical coverage, we then expand the keywords through word2vec (Mikolov et al., 2013) nearest neighbors, filtered to semantically equivalent words by the domain expert.[11] This process is repeated using WordNet synonym sets for lookup (Miller, 1995), resulting in 217 keywords total; see appendix (§C.2) for details.

RoBERTa+MNLI. Given two input sentences, a premise and hypothesis, the task of natural language inference (NLI) is to predict whether the premise *entails* or *contradicts* the hypothesis or does neither (*neutral*) (Bowman et al., 2015; Williams et al., 2018). Previous work has shown promise of NLI transfer learning for events: Sarwar and Allan (2020) show Sentence-BERT embeddings (Reimers et al., 2019) learned from NLI data are effective on ACE-like event retrieval; Clark et al. (2019) find for *BoolQ*, their dataset of naturally occurring boolean questions, that transfer learning from NLI data is more effective than transferring from QA or paraphrase data. We follow Clark et al. (2019)'s example and use "off-the-shelf" RoBERTa (Liu et al., 2019) fine-tuned on the MNLI corpus (Williams et al., 2018). The model takes a sentence and a declarative form of an event class question as input (§C.1), and we use its predicted probability of entailment as the probability of the event class. For document ranking, we create a document score by taking the maximum predicted probability over sentences. Future experiments could vary the amount of text (sentence vs. passage vs. document) used as input to the model.

BM25+RM3. Weighted term matching between a query and document is a strong competitor to neural ranking methods (Craswell et al., 2020; Lin, 2019), via, for example, BM25 scoring with RM3 query expansion (Lavrenko and Croft, 2001). With the Anserini BM25 implementation (Yang et al., 2018a), we set $k_1 = 0.9$ and $b = 0.4$, and conduct RM3 expansion of the query to terms found in the top $k = 10$ BM25-retrieved documents, following Lin (2019)'s hyperparameter settings. As the input query, this set of models uses the natural language questions described in §4.2. Appendix Table A5 contains full results.

ELECTRA+MS MARCO. Fine-tuned BERT (Devlin et al., 2019) and other large-scale language models have been used extensively for document ranking in information retrieval (IR) (Zhan et al., 2020; Zhang et al., 2020; Dai and Callan, 2019; MacAvaney et al., 2019). We use a competitive off-the-shelf model that uses the ELECTRA variant of BERT (Clark et al., 2020) fine-tuned on MS MARCO (Reimers et al., 2019). MS MARCO is a large-scale reading comprehension dataset in which questions are sampled from anonymized web

---

[11] We train word2vec on every article in the *Times of India* from 2002 (the same corpus as our dataset, 69,000 articles) plus another 100,000 articles from *The Hindu*, another English-language newspaper in India. We inspect each keyword's 20 nearest neighbors with highest cosine similarity.

queries, and answers to the queries are generated by crowdworkers (Nguyen et al., 2016). It was used in the 2019 TREC Deep Learning track on document and passage retrieval (Craswell et al., 2020). We use Reimers et al. (2019)'s pretrained ELECTRA+MS MARCO model. Inputs are INDIAPOLICEEVENTS passages consisting of three-sentence sliding windows with stride of one sentence and queries are the event class questions described in §4.2. Following related work (Zhan et al., 2020), we take the maximum score of all passages as the document score.

6 Results

We report the performance of the baseline models (§5) on our three tasks in Table 2 and in Figure 3.

6.1 Task 1: Sentence classification.

For sentence classification,[12] Table 2 shows that the keyword matching method slightly outperforms RoBERTa+MNLI on F1 for ANY ACTION and FORCE, which we suspect is due to the keyword method having better access to synonyms of "police" (e.g. "jawan", "RPF") particular to the *Times of India* via its *word2vec* expansion. However, RoBERTa+MNLI achieves a higher F1 score on KILL, ARREST, and FAIL TO ACT. We need further controlled experiments to understand how the concreteness of the event class, importance of identifying events' agents, and formulation of the query (e.g. "Did police use force or violence?" vs. "Were police violent?") affect the results of contextualized language models. Table 2 also shows poor performance of our keyword matching method on FAIL TO ACT (F1=0.05); however, a large-scale contextual language model seems to be able to better distinguish the semantics of the event class (F1=0.48). The Task 1 plot in Figure 3 shows that across all labels, RoBERTa+MNLI has higher recall than the keyword method for every event class. If social scientists plan to use these sentence classification methods in a semi-automated fashion (as we suggest in §1), selecting models like RoBERTa+MNLI that achieve higher recall may be important.

6.2 Task 2: Document ranking.

For Task 2, we report average precision and a new metric—the proportion of documents that would have to be read to achieve recall equal to X (*PropRead@RecallX*). We use $X = 0.95$ because social scientists typically use 95% cutoffs for significance and sampling error.[13] We leave to future work estimating recall on a corpus without ground truth. Table 2 shows that RoBERTa+MNLI outperforms both BM25 and ELECTRA+MS MARCO on both average precision and PropRead@Recall95 across all event classes. We hypothesize this is because natural language inference is a task that is much more aligned with the semantic-oriented precision at which we want to rank documents. In contrast, the MS MARCO dataset is constructed for a much higher level information need, and documents that are "relevant" could potentially not entail the semantic event class of interest. As Figure 3 shows, if a social scientist was presented with a ranked list of documents from RoBERTa+MNLI, they would only have to read 5% of the entire corpus to achieve 95% recall on KILL. RoBERTa+MNLI also does well on ARREST and FORCE with 0.17 and 0.20 PropRead@Recall95 respectively. There is consistently more difficulty across all models for ANY ACTION and FAIL TO ACT. We speculate this is because ANY ACTION is the class with the greatest prevalence, and thus is more difficult to achieve higher recall.

6.3 Task 3: Temporal Aggregates.

Figure 4 compares the outputs of three systems on FAIL TO ACT: gold-standard human annotations, keyword matching, and RoBERTa+MNLI. For this event class, both automated methods under-count the number of events marked by human annotators. In contrast, the automated techniques tend to *over*count other event types (see Appendix, Figure A5 for plots of the other event classes). While the the overall temporal trend is broadly consistent across the three methods, the decreased accuracy of the automated methods could lead to attenuation bias if they were used as input to statistical models. A qualitative examination of the extracted events also reveals the need for future work in temporal linking models: most of the events after March 25

---

[12] We do not evaluate sentences with less than 5 tokens as many of these sentences are due to sentence segmentation errors. After this filtering, the number of remaining sentences we evaluate on is 18,645.

[13] We note that 5% recall error is not equivalent to a 5% sampling error. In practice, researchers are more concerned with whether data is missing at random.

	Task 1: Sent. Cls.		*Task 2: Document Ranking*						*Task 3: Temp. Aggs.*	
	Keyw.	R+MNLI	BM25		E+MSM		R+MNLI		Keyw.	R+MNLI
Event Class	F1 ↑	F1 ↑	AP ↑	PR ↓	AP ↑	PR ↓	AP ↑	PR ↓	ρ ↑	ρ ↑
KILL	0.50	**0.74**	0.30	0.29	0.65	0.27	**0.96**	**0.05**	0.70	**0.78**
ARREST	0.48	**0.62**	0.68	0.36	0.72	0.67	**0.91**	**0.17**	0.71	**0.85**
FAIL TO ACT	0.05	**0.48**	0.27	0.77	0.36	0.87	**0.63**	**0.76**	0.42	**0.60**
FORCE	**0.65**	0.62	0.24	0.43	0.64	0.45	**0.90**	**0.20**	**0.89**	0.86
ANY ACTION	**0.67**	0.57	0.53	0.85	0.83	0.88	**0.89**	**0.62**	0.86	**0.90**

Table 2: Evaluation of two classification methods (Keyw., R+MNLI) and three ranking models (BM25, E+MSM, R+MNLI) for IndiaPoliceEvents's three tasks. Bolded numbers indicate the model that performs best on each metric and event class. **Task 1** evaluates sentence-level F1 for sentence-level keyword matching (Keyw.) and RoBERTa fine-tuned on MNLI (R+MNLI) (Liu et al., 2019). **Task 2** evaluates average precision (AP) and proportion of the corpus needed to be read in order to achieve 95% recall (PR, or PropRead@Recall95) for ranking models BM25 (Yang et al., 2018a), off-the-shelf ELECTRA language model fine-tuned on MS MARCO (E+MSM; (Reimers et al., 2019)), as well as R+MNLI's probabilistic output. **Task 3** evaluates Spearman's rank correlation coefficient (ρ) between predicted versus gold-standard counts of documents with the relevant event, for each day in March 2002. For each metric, we indicate whether a higher (↑) or lower (↓) score is better.

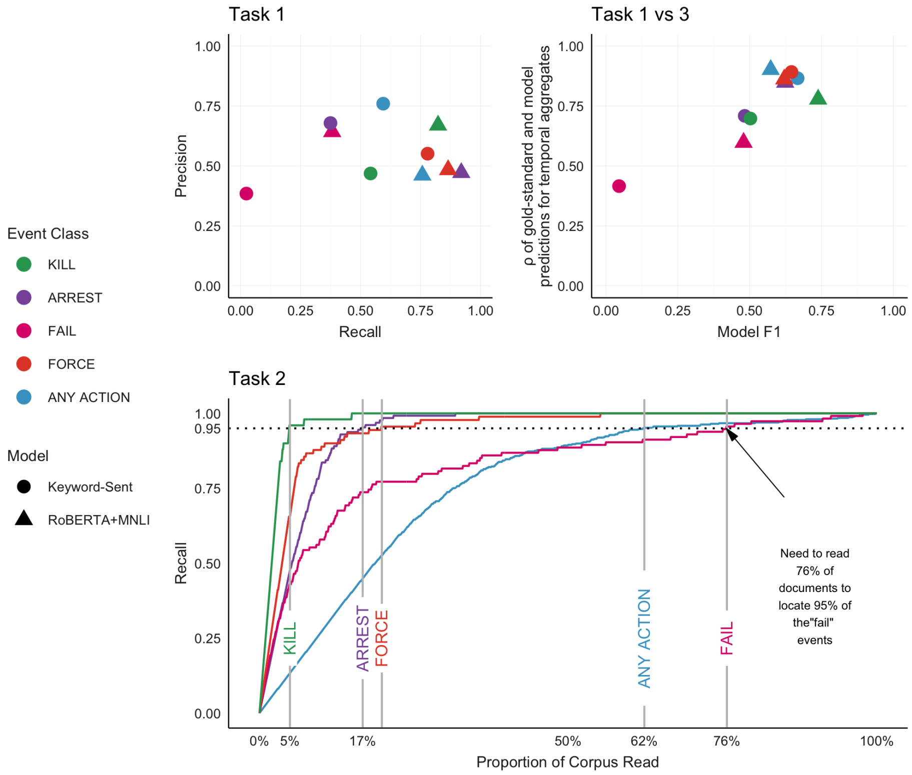

Figure 3: Keyword and RoBERTA+MNLI performance on three metrics. **(Task 1)** Precision and recall at the sentence level for two models on each semantic event class. **(Task 1 vs 3)** Sentence-level model F1 scores (x-axis) versus Spearman's ρ of hand-annotated gold-standard and model predictions for temporal aggregates (y-axis). **(Task 2)** For each class under RoBERTa MNLI, the gain curves (Grossman et al., 2016), marking PropRead@Recall95: What percentage of the ranked corpus would a researcher need to read in order to find 95% of each event classes' mentions?

are describing events from earlier in March that were being reported in the context of investigations into the violence. Table 2 shows that for all event classes except for FORCE RoBERTa+MNLI has a

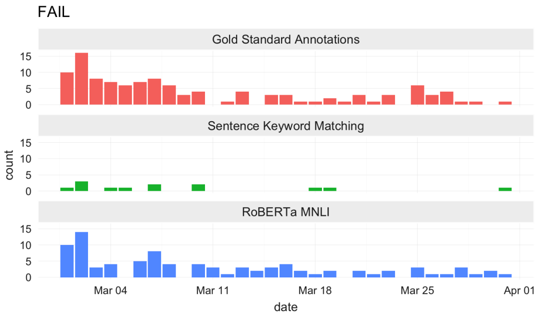

Figure 4: Number of documents per day containing a "police failing to act/standing by" event, comparing outputs from **(Top)** human annotations **(Middle)** keyword matching (Spearman's $\rho = 0.51$, comparing with gold standard) and **(Bottom)** RoBERTa finetuned on MNLI ($\rho = 0.60$).

higher Spearman's ρ[14] between the predicted versus gold-standard document counts. The Task 1 vs. 3 plot in Figure 3 shows an approximately linear relationship between the F1 scores of sentence-level models and Spearman's ρ, suggesting there is promise that NLP research focused on sentence-level models could be of use to social scientists who care about corpus-level evaluation.

6.4 Qualitative error analysis.

We manually analyze the false positives and false negatives of our best-performing baseline model, RoBERTa+MNLI. Some false positives are due to lexical semantic misunderstandings: the model often mistakes "shot" for KILL, and assigns high probability to negative FORCE sentences such as, *"The police escorting the vehicle fired into the air and dispersed the mob."* The model also has difficulty identifying the police as agents: for example, it assigns high probability to the negative KILL sentences: *"[...] scores of people have been killed in rural Gujarat due to police failure [...],"* and *"Police said that two persons had been killed in Vijaynagar [...]"*. Other errors are due to *hypotheticals*: the model assigns high probability to the negative KILL sentences *"He alleged BJP's hand in the murder of [...]"* and *"Achar claims she was an eye-witness to police complicity in the violence."* Many of the model's false negatives are due to necessary multi-sentence context (which RoBERTa+MNLI does not have as it only takes single sentences as input). For instance, the model assigns low probability to the positive KILL sentence *"Four persons have been killed and five are injured."* and FORCE sentence *"One person was injured and rushed to the SSG hospital"*; if one reads the proceeding context of both of these sentences it is clear that police are the agents of the actions.

7 Discussion and Future Work

The dataset, tasks, and evaluations we present in this work are driven by the needs of social scientists: we assess the performance of zero-shot models on metrics important to applied researchers, including recall against a fully annotated corpus and performance at temporally aggregated levels. We find cause for optimism for social scientists using BERT-style pre-trained models on their tasks. These models could potentially be used in place of social scientists' existing keyword-based classifiers, although we caution accuracy is far from perfect and applied researchers will need to extensively validate model outputs. Even with imperfect classification accuracy, we believe these zero-shot models show promise for decreasing human annotation effort by reducing the proportion of the corpus read to achieve a specific recall level (the metric we call PropRead@RecallX).

Future work can extend our dataset creation process to new semantic event classes, such as protests, communal violence itself, and other forms of participation in political and social activity. Additional annotated datasets could allow researchers to generalize the performance of zero-shot language models to new domains and event classes. Finally, tasks such as temporal and geographic linking, event deduplication and coreference, and identifying hypothetical events are unsolved but are major obstacles for applied social scientists working with automatically extracted events.

8 Acknowledgments

For helpful comments, we're grateful to Aidan Milliff, the UMass NLP group, and anonymous ACL reviewers. This work was funded by a Kaggle Open Data Science grant, and additionally: *AH:* a National Science Foundation Graduate Research Fellowship; *KK:* a Bloomberg Data Science PhD Fellowship; *SS:* the Center for Intelligent Information Retrieval (CIIR); *BO:* National Science Foundation IIS-1845576 and IIS-1814955.

---

[14] See Table A6 for mean absolute error scores.

9 Ethical Considerations and Broader Impact

To ensure the replicability of our work and to further research into event extraction systems for social science research, we are making the text of the news articles available to researchers alongside our annotations. While all articles were obtained from a public website without login credentials, the applicability of copyright restrictions is relevant to address.

We believe the research benefits and the limited harms to the copyright holders justify this use, due to the four criteria considered in the fair use doctrine in U.S. copyright law (U.S. Copyright Office, 2021): (1) the non-commercial, nonprofit educational purpose of our use of the text, (2) the factual nature of the news reports, (3) the limited substitutability of our dataset for the original news site,[15] and (4) our expectation that our limited corpus will not harm the market for readers of the news site.

The issue of copyright status within NLP-oriented corpora is of increasing interest. Sag (2019) argues machine learning uses of text is non-expressive and therefore falls under fair use, and Geiger et al. (2018) study the issue in the context of proposals for E.U. law. Bandy and Vincent (2021) investigate BooksCorpus, a previously poorly documented corpus widely used for training language models, finding it contains large amounts of copyrighted work, highlighting how current data curation practices in machine learning (and adjacent) communities need improvement (Paullada et al., 2020; Jo and Gebru, 2020).

We also acknowledge the sensitivities around this period of violence in India. Its significance motivates computational work to enable more effective study of it and related episodes, but our news-derived data on its own, in the absence of deeper qualitative work, does not permit us to draw new substantive conclusions about the causes and consequences of the violence in Gujarat in 2002. We defer to the large scholarly and journalistic literature on the violence; see references in §1 and §4.

---

[15] We do not republish the texts as consumer-accessible webpages, but instead are only contained within a JSON structured format.

References

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89, Vancouver, Canada. Association for Computational Linguistics.

Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. 2014. A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 45–53.

Avi Arampatzis, Jaap Kamps, and Stephen Robertson. 2009. Where to stop reading a ranked list? threshold optimization using truncated score distributions. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 524–531.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Scott R Baker, Nicholas Bloom, and Steven J Davis. 2016. Measuring economic policy uncertainty. *The quarterly journal of economics*, 131(4):1593–1636.

Jack Bandy and Nicholas Vincent. 2021. Addressing "documentation debt" in machine learning research: A retrospective datasheet for BookCorpus. *arXiv preprint arXiv:2105.05241*.

Amrita Basu. 2015. *Violent conjunctures in democratic India*. Cambridge University Press.

John Beieler. 2016. Generating politically-relevant event data. *CoRR*.

John Beieler, Patrick T Brandt, Andrew Halterman, Erin Simpson, and Philip A Schrodt. 2016. Generating political event data in near real time: Opportunities and challenges. In R. Michael Alvarez, editor, *Computational Social Science*, chapter 98. Cambridge University Press.

Ward Berenschot. 2012. *Riot politics: Hindu-Muslim violence and the Indian state*. Rupa Publications.

Elizabeth Boschee. 2016. Solutions for coding societal events. Technical report, Raytheon BBN Technologies Corp. Cambridge United States.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Stephen M Shellman, James Starz, and Michael D Ward. 2015. ICEWS coded event data. In *Harvard Dataverse, V9, http://dx.doi.org/10.7910/DVN/28075*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Robert Brathwaite and Baekkwan Park. 2018. Measurement and conceptual approaches to religious violence: The use of natural language processing to generate religious violence event-data. *Politics and Religion*, pages 1–42.

Thomas Chadefaux. 2014. Early warning signals for war in the news. *Journal of Peace Research*, 51(1):5–18.

Erica Chenoweth and Orion A Lewis. 2013. Unpacking nonviolent campaigns introducing the NAVCO 2.0 dataset. *Journal of Peace Research*, 50(3):415–423.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *ICLR*.

Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1):80–91.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 Deep Learning track. *arXiv preprint arXiv:2003.07820*.

Zhuyun Dai and J. Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ursula Daxecker, Elio Amicarelli, and Alexander Jung. 2019. Electoral Contention and Violence (ECAV): A new dataset. *Journal of Peace Research*, 56(5):714–723.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Raheel Dhattiwala and Michael Biggs. 2012. The political logic of ethnic violence: The anti-Muslim pogrom in Gujarat, 2002. *Politics & Society*, 40(4):483–516.

Erika Doggett and Alejandro Cantarero. 2016. Identifying eyewitness news-worthy events on Twitter. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 7–13.

Vito D'Orazio, Steven T Landis, Glenn Palmer, and Philip Schrodt. 2014. Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Political analysis*, 22(2):224–242.

Ellen A Drost et al. 2011. Validity and reliability in social science research. *Education Research and perspectives*, 38(1):105.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Anca Dumitrache, Lora Aroyo, and Chris Welty. 2018. Crowdsourcing ground truth for medical relation extraction. *ACM Transactions on Interactive Intelligent Systems*, 8(2):1–20.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke S. Zettlemoyer. 2018. Large-scale QA-SRL parsing. In *ACL*.

Christophe Geiger, Giancarlo Frosio, and Oleksandr Bulayenko. 2018. The Exception for Text and Data Mining (TDM) in the Proposed Directive on Copyright in the Digital Single Market—Legal Aspects. Policy Department for Citizens' Rights and Constitutional Affairs, Directorate General for Internal Policies of the Union, European Parliament, PE 604.941, February 2018. https://www.europarl.europa.eu/RegData/etudes/IDAN/2018/604941/IPOL_IDA(2018)604941_EN.pdf accessed 2021-05-25.

Deborah J. Gerner, Philip A Schrodt, Omür Yilmaz, and Rajaa Abu-Jabr. 2002. Conflict and mediation event observations (CAMEO): A new event data framework for the analysis of foreign policy interactions. *International Studies Association, New Orleans*.

Justin Grimmer and Brandon M Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297.

Ralph Grishman. 1997. Information extraction: Techniques and challenges. In *International summer school on information extraction*, pages 10–27. Springer.

Ralph Grishman. 2019. Twenty-five years of information extraction. *Natural Language Engineering*, 25(6):677–692.

Ralph Grishman and Beth M Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

M. R. Grossman, G. Cormack, and Adam Roegiest. 2016. TREC 2016 Total Recall Track overview. In *TREC*.

Jesse Hammond and Nils B Weidmann. 2014. Using machine-coded event data for the micro-level study of political violence. *Research & Politics*, 1(2).

Alex Hanna. 2014. Developing a system for the automated coding of protest event data. *Available at SSRN: http://ssrn.com/abstract=2425232*.

Alex Hanna. 2017. MPEDS: Automating the generation of protest event data. *SocArXiv https://osf.io/preprints/socarxiv/xuqmv*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268, Berlin, Germany. Association for Computational Linguistics.

Human Rights Watch. 2002. “We have no orders to save you”: State participation and complicity in communal violence in Gujarat. *Human Rights Watch Report*, 14(3).

Eun Seo Jo and Timnit Gebru. 2020. Lessons from archives: strategies for collecting sociocultural data in machine learning. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*.

Katherine Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O’Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1547–1557.

Katherine Keith, Christoph Teichmann, Brendan O’Connor, and Edgar Meij. 2020. Uncertainty over Uncertainty: Investigating the Assumptions, Annotations, and Text Measurements of Economic Policy Uncertainty. In *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*, pages 116–131.

Gary King. 1989. *Unifying political methodology: The likelihood theory of statistical inference*. Cambridge University Press.

Gary King and Will Lowe. 2003. An automated information extraction tool for international conflict data with performance as good as human coders: A rare events evaluation design. *International Organization*, 57(03):617–642.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’01, page 120–127.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *CoNLL 2017*, page 333.

Yan Liang, Khaled Jabr, Christan Grant, Jill Irvine, and Andrew Halterman. 2018. New techniques for coding political events across languages. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 88–93. IEEE.

Yen-Chieh Lien, Daniel Cohen, and W Bruce Croft. 2019. An assumption-free approach to the dynamic truncation of ranked lists. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 79–82.

Jimmy Lin. 2019. The simplest thing that can possibly work: Pseudo-relevance feedback using text classification. *ArXiv*, abs/1904.08861.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181*.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR’19, page 1101–1104.

Andrew McCallum. 2005. Information extraction: Distilling structured data from unstructured text. *Queue*, 3(9):48–57.

Graham McDonald, Craig Macdonald, and Iadh Ounis. 2018. Active learning strategies for technology assisted sensitivity review. In *European Conference on Information Retrieval*, pages 439–453. Springer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching. *Artificial Intelligence*, to appear.

Hannes Mueller and Christopher Rauh. 2017. Reading between the lines: Prediction of political violence using newspaper text. *American Political Science Review*, pages 1–18.

Minh Nguyen and Thien Huu Nguyen. 2018. Who is killed by police: Introducing supervised attention for hierarchical LSTMs. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2277–2287, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Richard A Nielsen. 2013. Rewarding human rights? Selective aid sanctions against repressive states. *International Studies Quarterly*, 57(4):791–803.

Brendan O'Connor, Brandon Stewart, and Noah A Smith. 2013. Learning to extract international relations from political context. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1.

Javier Osorio and Alejandro Reyes. 2017. Supervised event coding from text written in Spanish: Introducing Eventus ID. *Social Science Computer Review*, 35(3):406–416.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer Event Description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56.

Amandalynne Paullada, Inioluwa Deborah Raji, Emily M Bender, Emily Denton, and Alex Hanna. 2020. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *arXiv preprint arXiv:2012.05345*. NeurIPS 2020 Workshop: ML Retrospectives, Surveys & Meta-analyses (ML-RSA), Virtual.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694.

Vinodkumar Prabhakaran, Camilla Griffiths, Hang Su, Prateek Verma, Nelson Morgan, Jennifer L Eberhardt, and Dan Jurafsky. 2018. Detecting institutional dialog acts in police traffic stops. *Transactions of the Association for Computational Linguistics*, 6:467–481.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. O'Reilly Media, Inc.

Benjamin James Radford. 2016. *Automated learning of event coding dictionaries for novel domains with an application to cyberspace*. Ph.D. thesis, Duke University.

Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.

Matthew Sag. 2019. The new legal landscape for text mining and machine learning. *Journal of the Copyright Society of the USA*, 66:1–64. Accessed at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3331606.

Sheikh Muhammad Sarwar and James Allan. 2019. Searchie: A retrieval approach for information extraction. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, page 249–252.

Sheikh Muhammad Sarwar and James Allan. 2020. Query by example for cross-lingual event retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1601–1604.

Philip A Schrodt, Shannon G Davis, and Judith L Weddle. 1994. Political science: KEDS—a program for the machine coding of event data. *Social Science Computer Review*, 12(4):561–587.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Gabriel Stanovsky, Meni Adler, and Ido Dagan. 2016. Specifying and annotating reduced argument span via QA-SRL. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Kadayam Suryanarayanan Subramanian. 2007. *Political violence and the police in India*. SAGE Publications India.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–310, Online. Association for Computational Linguistics.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2020. A survey on explainability in machine reading comprehension. *arXiv preprint arXiv:2010.00389*.

U.S. Copyright Office. 2021. More information on fair use. https://www.copyright.gov/fair-use/more-info.html http://web.archive.org/web/20210521070009/https://www.copyright.gov/fair-use/more-info.html.

Siddharth Varadarajan. 2002. *Gujarat, the Making of a Tragedy*. Penguin Books India.

Rob Voigt, Nicholas P Camp, Vinodkumar Prabhakaran, William L Hamilton, Rebecca C Hetey, Camilla M Griffiths, David Jurgens, Dan Jurafsky, and Jennifer L Eberhardt. 2017. Language from police body camera footage shows racial disparities in officer respect. *Proceedings of the National Academy of Sciences*, 114(25):6521–6526.

Hanna Wallach. 2018. Computational social science$\neq$ computer science+ social data. *Communications of the ACM*, 61(3):42–44.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, Dong Yu, David McAllester, and Dan Roth. 2019. Evidence sentence extraction for machine reading comprehension. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 696–707.

Steven I Wilkinson. 2006. *Votes and violence: Electoral competition and ethnic riots in India*. Cambridge University Press.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018a. Anserini: Reproducible ranking baselines using lucene. *ACM J. Data Inf. Qual.*, 10:16:1–16:20.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018b. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv preprint arXiv:2006.15498*.

Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2020. A little bit is worse than none: Ranking with limited training data. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 107–112, Online. Association for Computational Linguistics.

Memory-Efficient Differentiable Transformer Architecture Search

Yuekai Zhao[†], **Li Dong**[‡], **Yelong Shen**[‡], **Zhihua Zhang**[§], **Furu Wei**[‡], **Weizhu Chen**[‡]
[†]Academy for Advanced Interdisciplinary Studies, Peking University
[§]School of Mathematical Sciences, Peking University
[‡]Microsoft Corporation
`{yuekaizhao@,zhzhang@math.}pku.edu.cn`
`{lidong1,yeshe,fuwei,wzchen}@microsoft.com`

Abstract

Differentiable architecture search (DARTS) is successfully applied in many vision tasks. However, directly using DARTS for Transformers is memory-intensive, which renders the search process infeasible. To this end, we propose a multi-split reversible network and combine it with DARTS. Specifically, we devise a backpropagation-with-reconstruction algorithm so that we only need to store the last layer's outputs. By relieving the memory burden for DARTS, it allows us to search with larger hidden size and more candidate operations. We evaluate the searched architecture on three sequence-to-sequence datasets, i.e., WMT'14 English-German, WMT'14 English-French, and WMT'14 English-Czech. Experimental results show that our network consistently outperforms standard Transformers across the tasks. Moreover, our method compares favorably with big-size Evolved Transformers, reducing search computation by an order of magnitude.

1 Introduction

Current neural architecture search (NAS) studies have produced models that surpass the performance of those designed by humans (Real et al., 2019; Lu et al., 2020). For sequence tasks, efforts are made in reinforcement learning-based (Pham et al., 2018) and evolution-based (So et al., 2019; Wang et al., 2020) methods, which suffer from the huge computational cost. Instead, gradient-based methods (Liu et al., 2018; Jiang et al., 2019; Yang et al., 2020) are less demanding in computing resources and easy to implement, attracting many attentions recently.

The idea of gradient-based NAS is to train a super network covering all candidate operations. Different sub-graphs of the super network form the search space. To find a well-performing subgraph, Liu et al. (2018) (DARTS) introduced search parameters jointly optimized with the network weights. Operations corresponding to the largest search parameters are kept for each intermediate node after searching. A limitation of DARTS is its memory inefficiency because it needs to store the intermediate outputs from all its candidate operations. This is much more pronounced when we apply Transformers (Vaswani et al., 2017) as the backbone of DARTS (the operation set is detailed in Section 2.5). As shown in Figure 1, memory consumption grows extremely fast as we increase the hidden size d, quickly running out of memory as $d > 400$. As a result, we can only use a limited operation set or a small hidden size, which may lead to worse model performance.

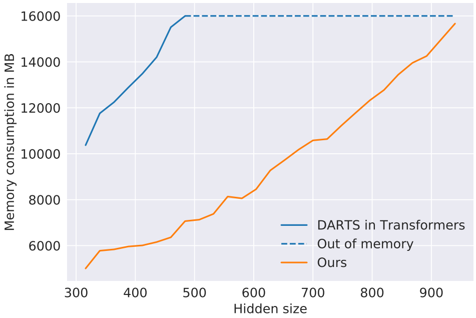

Figure 1: Memory comparison of using our reversible networks and Transformers as the backbone model of DARTS. Experiments are run on a single step of forward-backward pass on a batch of 3584 tokens with a NVIDIA P100 GPU. Limited by GPU memory, DARTS in Transformers has to search in small sizes while evaluating in large sizes, which will cause performance gaps (Chen et al., 2019).

To address the unfavorable memory consumption issue in DARTS, we propose a variant of reversible networks. Each input of a reversible network layer can be reconstructed from its outputs. Thus, it is unnecessary to store intermediate outputs except for the last layer because we can reconstruct

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4254–4264
August 1–6, 2021.

them during backpropagation (BP). Inspired by the idea of RevNets (Gomez et al., 2017), we devise a multi-split reversible network. Each split contains a mixed operation search node to enable DARTS. Also, only a small modification of BP is needed to enable gradient calculation with input reconstruction. We show the memory consumption of our method in Figure 1, which on average halves the amount of memory required in the vanilla DARTS. We can search larger, deeper networks with a richer candidate operation set under the same memory constraint.

Our method is generic to handle various network structures. In this work, we focus on the sequence-to-sequence task. We first perform the architecture search using the WMT'14 English-German translation task. The resulting architecture is then re-trained on three datasets: WMT'14 English-German, WMT'14 English-French, and WMT'14 English-Czech. We achieve consistent improvement over standard Transformers in all tasks. At a medium model size, we can have the same translation quality as the original "big" Transformer with 69% fewer parameters. At a big model size, we exceed the performance of the Evolved Transformer (So et al., 2019), with the computational cost lowered by an order of magnitude. We will make our code and models publicly available.

2 Methodology

We give a detailed description of our method. In Section 2.1, we introduce DARTS and its memory inefficiency when applying in Transformers. In Section 2.2, we propose a multi-split reversible network, which works as the backbone of our memory-efficient architecture search approach. Section 2.3 shows a backpropagation-with-reconstruction algorithm. In Section 2.4, we manage to combine DARTS with our reversible networks. Finally, in Section 2.5, we summarize the proposed algorithms with more details.

2.1 Differentiable Architecture Search in Transformers

Following (Liu et al., 2018), we explain the idea of differentiable architecture search (DARTS) within a one-layer block. Let $\mathscr{O}$ be the candidate operation set (e.g., Self Attention, FFN, Zero). Each operation $o \in \mathscr{O}$ represents some function that can be applied to the layer inputs or hidden states (denoted X). The key of DARTS is to use a mixed operation search node $f(X)$ to relax the categorical choice of a specific operation to a softmax over all candidate operations:

$$f(X) = \sum_{o \in \mathscr{O}} \frac{\exp(\alpha_o)}{\sum_{o' \in \mathscr{O}} \exp(\alpha_{o'})} o(X), \qquad (1)$$

where the α are trainable parameters of size $|\mathscr{O}|$ that determines the mixing weights. During searching, a one-layer block contains several search nodes. The task is to find a suitable set of α for each search node. At the end of the search, the resulting operation in each node is determined by:

$$f = \arg\max_{o \in \mathscr{O}} \alpha_o. \qquad (2)$$

We optimize the α together with network weights θ by gradient descent. A good architecture means performing well on the searching validation set, such that we optimize α with validation loss L_{val} and θ with training loss L_{train}:

$$\begin{aligned} &\min_{\alpha} L_{val}(\theta^*(\alpha), \alpha), \\ &\text{s.t.} \;\; \theta^*(\alpha) = \arg\min_{\theta} L_{train}(\theta, \alpha). \end{aligned}$$

In practice, we update α by $\nabla_\alpha L_{val}$ and θ by $\nabla_\theta L_{train}$ in each step.

It is easy to directly apply DARTS in Transformers by replacing some or all operations in a Transformer block with mixed operation search nodes. For example, we can change the transformer decoder block from Self Attn → Cross Attn → FFN to Search Node 1 → Cross Attn → Search Node 2. Note that a search node outputs a weighted sum of different operations. To enable gradient calculation in the backward pass, we need to store every operation's output, which results in a steep rise in memory consumption during searching. Figure 1 shows the memory consumption of using 2 search nodes in both Transformer encoder and decoder. DARTS run out of memory easily, even at a small hidden size.

2.2 Multi-split Reversible Networks

To relieve the memory burden of DARTS in Transformers, we use reversible networks. A reversible network layer's input can be reconstructed from its output. Suppose a network is comprised of several reversible layers. We do not need to store intermediate outputs except the last layer, because we can reconstruct them from top to bottom during backpropagation (BP). Denote by X and $f(X)$ the

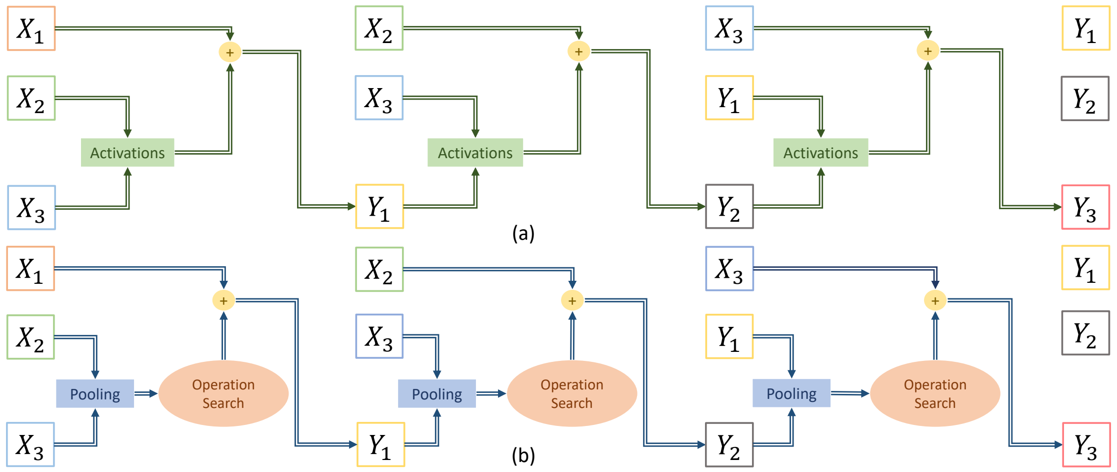

Figure 2: This figure is a demonstration of DARTSformer. (a) shows an n-split ($n = 3$) reversible network, which serves the backbone of our method. (b) shows the design of activations to enable differentiable architecture search. Each X_k and Y_k are in $R^{l\times d}$. The k-th pooling takes the concatenation of $X_{i>k}$ and $Y_{i<k}$ as the input, and outputs a tensor in $R^{l\times d}$. The operation search gives a weighted average of the outputs of each candidate operation.

layer input and the layer output, respectively. X is first split along the embedding/channel dimension into n equal parts $\{X_1, \cdots, X_n\}$. A RevNets (Gomez et al., 2017) alike operation is applied to each X_k, which yields Y_k. $f(X)$ is a concatenation of $\{Y_1, \cdots, Y_n\}$ along the split dimension:

$$\begin{aligned} Y_1 &= X_1 + G_1(X_{i>1}, \theta_1), \\ &\cdots \\ Y_k &= X_k + G_k(X_{i>k}, Y_{i<k}, \theta_k), \\ &\cdots \\ Y_n &= X_n + G_n(Y_{i<n}, \theta_n). \end{aligned} \tag{3}$$

G_k is a mixed operation node during the architecture search process. After searching, G_k is a deterministic operation given by $\arg\max_{o\in\mathcal{O}} \alpha_o$. Detailed discussions can be found in Section 2.4.

The reversibility of Eq. (3) needs rigorous validation, such that the input X can be easily reconstructed from $f(X)$:

$$\begin{aligned} X_n &= Y_n - G_n(Y_{i<n}, \theta_n), \\ &\cdots \\ X_k &= Y_k - G_k(X_{i>k}, Y_{i<k}, \theta_k), \\ &\cdots \\ X_1 &= Y_1 - G_1(X_{i>1}, \theta_1). \end{aligned} \tag{4}$$

Part (a) of Figure 2 illustrates a 3-split reversible network, which we frequently employ throughout our experiments for simplicity.

2.3 Backpropagation with Reconstruction

Consider the problem of backpropagating (BP) through a reversible layer. Based on the layer output $f(X) = \text{Concat}(Y_1, \cdots, Y_n)$ and its total derivative $df(X) = \text{Concat}(dY_1, \cdots, dY_n)$, we need to calculate the layer input $X = \text{Concat}(X_1, \cdots, X_n)$, its total derivative $dX = \text{Concat}(dX_1, \cdots, dX_n)$, and the derivatives of the network weights $d\theta_1, \cdots, d\theta_n$.

We show the BP-with-reconstruction through a single layer in Algorithm 1. $[\cdot]$ represents $\text{Concat}(\cdot)$ for simplicity reasons. In Line 9 of Algorithm 1, $d\theta_k$ is calculated as a side effect. Line 10 shows the reconstruction process, where each split X_k is recovered in the order of n to 1. In Algorithm 1, grad_k works as a gradient accumulator, which keeps track of all derivatives associated with X_k. A repetitive application of Algorithm 1 enables us to backpropagate through a sequence of reversible layers. Only the top layer's outputs require storage, which makes it much more memory-efficient.

Roughly speaking, for a network with N connections, the forward and backward passes require approximately N and $2N$ add-multiply operations, respectively. Since we need to reconstruct X from $f(X)$, the re-calculation requires another N add-multiply operations, making it 33% slower. Fortunately, we can only need Algorithm 1 for architecture search and will re-train the resulting network with ordinary BP. The search process turns out to converge fast. The computational overhead does not become a severe problem.

---

[1] Automatic differentiation routines, e.g. `tf.gradient`, `torch.autograd.backward`

Algorithm 1 BP-with-reconstruction Algorithm for Multi-Split Reversible Networks

```
Input:
    Layer output: $f(X) = [Y_1, \cdots, Y_n]$;
    Total derivatives: $df(X) = [dY_1, \cdots, dY_n]$;
    Operations: $G_1, \cdots, G_n$;
Output:
    Layer input: $X = [X_1, \cdots, X_n]$;
    Derivatives of $X$: $dX = [dX_1, \cdots, dX_n]$;
$X = \{\}; dX = \{\}; Y = \{Y_1, \cdots, Y_n\}$
for k in $n$ to 1 do
    $C = Y_k; Y = Y \setminus \{Y_k\}$
    if $k == n$ then
        $\text{grad}_k = dY_k$
    else
        $\text{grad}_k = dY_k + C.\text{grad}$
    end if
    $g_k = G_k(X, Y, \theta_k)$; $g_k$.backward[1]($\text{grad}_k$)
    $X_k = C - g_k; X = X \cup \{X_k\}$
end for
$dX_1 = \text{grad}_1, dX = \{dX_1\}$
for k in 2 to $n$ do
    $dX_k = X_k.\text{grad} + \text{grad}_k$
    $dX = dX \cup \{dX_k\}$
end for
```

2.4 DARTS with Multi-split Reversible Networks

Performing DARTS based on n-split reversible networks only requires specifying each G_k in Eq. (3). Suppose that each $X_k \in R^{l \times d_n}$ (l is the sequence length and d is the hidden size, $d_n = \frac{d}{n}$), and that each Y_k has the same size as X_k. The input of G_k contains $n-1$ tensors in $R^{l \times d_n}$. To enable element-wise addition with X_k, the output of G_k must also be in $R^{l \times d_n}$.

G_k is factorized into two parts. The first part is a pooling operation, which takes an $l \times d_n \times (n-1)$ tensor as input, and outputs an $l \times d_n \times 1$ tensor. The second part is a mixed operation search node. G_k is calculated as follows:

$$
\begin{aligned}
H_k &= \text{Pooling}(X_{i>k}, Y_{i<k}), \\
G_k &= \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^k)}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^k)} o(H_k),
\end{aligned}
\tag{5}
$$

where α_k is randomly initialized. Figure 3 shows the design of G_k. By substituting each G_k in Eq. (3) with Eq. (5), we are able to use Algorithm 1 to perform memory-efficient DARTS. We call this method *DARTSformer*, which is illustrated by Part (b) of Figure 2 in a 3-split case.

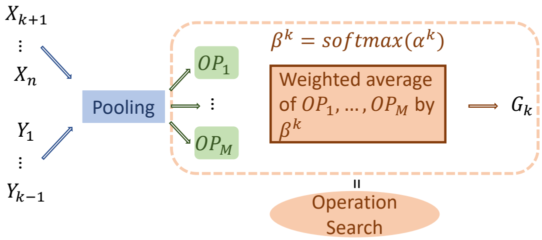

Figure 3: Pooling and operation search in each split.

The overall search space size is critical to the performance of DARTSformer. In our experiments, we focus on sequence-to-sequence tasks where the encoder and the decoder are searched simultaneously. Suppose that we have an m-split encoder and an n-split decoder. We search s consecutive layers. For example, $s = 2$ means that we search within a 2-layer encoder block. Each layer in the block is an m-split reversible layer. The encoder contains several identical 2-layer blocks, the same to the decoder. The search space is of size $|\mathcal{O}|^{s(m+n)}$. If $|\mathcal{O}|$ is large, it can easily introduce a large search space even with small m, n and s.

2.5 Instantiation

We describe the instantiation of DARTSformer in this section.

Operation Set The candidate operation set $\mathcal{O}$ is defined as follows:

- Standard Conv $w \times 1$: for $w \in \{3, 5, 7, 11\}$.
- Dynamic Conv $w \times 1$: for $w \in \{3, 7, 11, 15\}$.
- Self Attention.
- Cross Attention: Only available to decoder.
- Gated Linear Unit (GLU).
- FFN.
- Zero: Return a zero tensor of the input size.
- Identity: Return the input.

The Dynamic Conv is from Wu et al. (2019). The Self Attention, Cross Attention and FFN are from Vaswani et al. (2017). We use 8 attention heads. The GLU is from Dauphin et al. (2017).

Residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) are crucial for convergence in training Transformers (Vaswani et al., 2017). To make our network fully reversible, these two tricks can not be used directly. Instead, we put

Algorithm 2 The framework of DARTSformer

```
Input:
    Operation set: $\mathscr{O}$, Search parameters: $\alpha$;
    Network weights: $\theta$;
Output:
    Best candidate network: $\mathcal{N}_{final}$;
1: Setup a multi-split reversible network with operation search nodes $\mathcal{N}_{super}(\mathscr{O}, \alpha, \theta)$.
2: while $\alpha$ not converge do
3:     Update $\theta$ by Algorithm 1 with $L_{train}$.
4:     Update $\alpha$ by Algorithm 1 with $L_{val}$.
5: end while
6: Get $\mathcal{N}_{final}(\mathscr{O}, \alpha, \theta)$ with Eq. (2).
```

the residual connections and layer normalization within each operation $\tilde{o}(X) = \text{LayerNorm}(X + o(X))$, except for Zero and Identity.

Encoder and Decoder We use an n-split encoder and an $(n{+}1)$-split decoder for DARTSformer. Each G_k in the encoder takes the format of Eq. (5). Instead for the decoder, $G_{k<n+1}$ still follows Eq. (5), but the operation for the last split G_{n+1} is fixed as Cross Attention. Our experiments show that this constraint on the decoder yields architectures with better performances.

Search and Re-train We summarize the entire framework of DARTSformer in Algorithm 2. Note that the search process is the most memory intensive part, such that we use BP-with-reconstruction as shown in Line 2-5 of Algorithm 2.

3 Experiment Setup

3.1 Datasets

We use three standard datasets to perform our experiments as So et al. (2019): (1) WMT'18 English-German (En-De) without ParaCrawl, which consists of 4.5 million training sentence pairs. (2) WMT'14 French-English (En-Fr), which consists of 36 million training sentence pairs. (3) WMT'18 English-Czech (En-Cs), again without ParaCrawl, which consists of 15.8 million training sentence pairs. Tokenization is done by Moses[2]. We employ BPE (Sennrich et al., 2016) to generate a shared vocabulary for each language pair. The BPE merge operation numbers are 32K (WMT'18 En-De), 40K (WMT'14 En-Fr), 32K (WMT'18 En-Cs). We discard sentences longer than 250 tokens. For the re-training validation set, we randomly choose 3300 sentence pairs from the training set. The evaluation metric is *BLEU* (Papineni et al., 2002). We use beam search for test sets with a beam size of 5, and we tune the length penalty parameter from 0.5 to 1.0. Suppose the input length is m, and the maximum output length is $1.2m + 10$.

3.2 Search Configuration

The architecture searches are all run on WMT'14 En-De. DARTS is a bilevel optimization process, which updates network weights θ on one dataset and search parameters α on another dataset. We split the 4.5 million sentence pairs into 2.5/2.0 million for θ and α. Both L_{train} and L_{val} are cross entropy loss with a label smoothing factor of 0.1. The split number n is 2 for the encoder and 3 for the decoder. We set s to 1 or 2, which means the super network contains several identical 1-layer or 2-layer blocks. The candidate operations are detailed in Section 2.5, where $|\mathscr{O}| = 13/14$ for encoder and decoder, respectively. Along the analysis in Section 2.4, the largest size of the search space is around 1 billion. We use a factorized word embedding matrix to save memory. $|V|$ is the vocabulary size, and d is the hidden size. The original word embedding matrix $E \in R^{|V| \times d}$ is factorized into a multiplication of two matrices of size $|V| \times e$ and $e \times d$, where $e \ll d$. We let e denote the embedding size. We set $e = 256, d = 960$. During searching, we set the dropout probability to 0.1. Two Adam optimizers (Kingma and Ba, 2015) are used for updating θ and α, with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. For θ, we use the same learning rate scheduling strategy as done in Vaswani et al. (2017) with a warmup step of 10000. The maximum learning rate is set to 5×10^{-4}. For α, we fix the learning rate to 3×10^{-4} with a weight decay of 1×10^{-3}, which is the same as Liu et al. (2018) does.

DARTSformer requires us to specify a pooling operation as stated in Eq. (5). We experiment with both max pooling and average pooling. All searches run on the same 8 NVIDIA V100 hardware. We use a batch size of 5000 tokens per GPU and save a checkpoint every 10,000 updates (5000 for θ and 5000 for α). Our search process finalizes after 60,000 updates.

3.3 Training Details

All the networks derived from the saved checkpoints are re-trained on WMT'14 En-De to select the best performing one. We then train the selected network on all datasets in Section 3.1 to

---

[2] https://github.com/moses-smt/mosesdecoder

Model	**Pooling**	**Search s Layers**	**Model Size**	**BLEU**
Transformer	-	-	61.1M	27.7
ET	-	-	64.1M	28.2
Sampling	max	2	60.1M	18.7
Sampling	avg	2	61.6M	16.8
DARTSformer	max	1	64.5M	27.9
DARTSformer	max	2	65.2M	**28.4**
DARTSformer	avg	1	66.0M	28.3
DARTSformer	avg	2	63.4M	28.3

Table 1: BLEU scores of various search setups on WMT'14 En-De test set. ET is the Evolved Transformer (So et al., 2019). We use a 2-split encoder and a 3-split decoder.

Model	**Pooling**	**Splits**	**BLEU**
DARTSformer	max	2,3	**28.4**
DARTSformer	max	3,4	28.0
DARTSformer	max	4,5	27.4
DARTSformer	avg	2,3	**28.3**
DARTSformer	avg	3,4	27.9
DARTSformer	avg	4,5	27.1

Table 2: BLEU scores of DARTSformer with different split numbers on WMT'14 En-De test set. We use an n-split encoder and an $n+1$-split decoder. We searching through 2 consecutive layers.

verify its generalization ability. We follow the settings of So et al. (2019) with both a base model and a big model. For the base model, we still use $e = 256, d = 960$ without re-scaling. For the big model, we set $e = 512, d = 1824$. Unless otherwise stated, all the training run on 8 Tesla V100 GPU cards with the batch size of 5000 tokens per card.

4 Results

4.1 Comparison Between Search Setups

We search through a different number of consecutive layers with different pooling operations. For re-training, we use the same learning rate scheduling strategy as in searching. We also keep the dropout rate unchanged. Results are summarized in Table 1. DARTSformers yields better results than standard Transformers in all experimental setups. The maximum performance gain is 0.7 BLEU with max pooling when searching through 2 consecutive layers. Also, DARTSformer achieves slightly better results than the Evolved Transformer in three out of four runs.

Model	**Price**	**Steps**	**Hardware**
ET	\$150k	4.2×10^8	200 TPUs
DARTSformer	\$1.25k	4.8×10^5	8 V100

Table 3: Comparison for search cost between Evolved Transformer (ET; So et al. 2019) and DARTSformer. The price for ET is from Strubell et al. (2019).

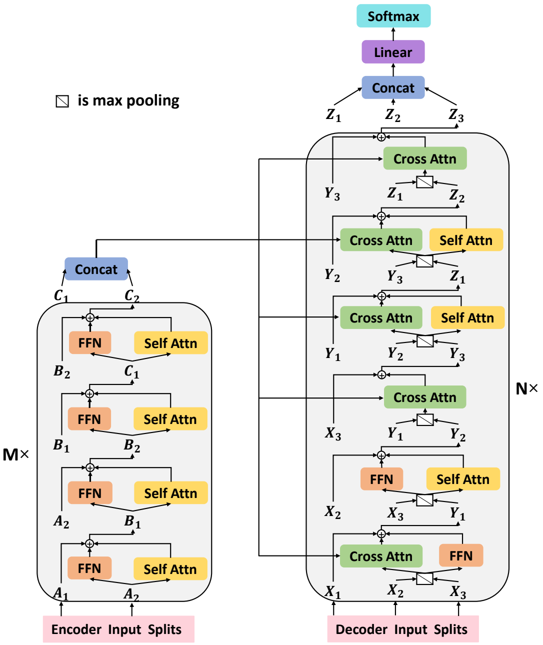

Figure 4: Architecture searched by DARTSformer.

We compare the search cost between the Evolved Transformers and DARTSformer from various aspects. DARTSformer takes about 40 hours to run on an AWS p3dn.24xlarge node[3]. The price for a single run of search is about \$1.25k. As reported by Strubell et al. (2019), the search process of Evolved Transformer takes up to \$150k, which is extremely expensive. As for hardware, the evolutionary search employs 200 TPU V.2 chips to run, while our method only uses 8 NVIDIA V100 cards. The reason for the evolutionary search algorithm's huge cost is that it requires training multiple candidate networks from scratch. We compare the number of parameter update steps in Table 3. The evolutionary search needs approximately 874 times more update steps than our method.

A simple sampling-based NAS method (Guo et al., 2020) can also reduce memory consumption.

---

[3] https://aws.amazon.com/ec2/instance-types/p3/

For each batch of training data, we set G_k in Eq. (5) as a uniformly sampled operation from the candidate set $\mathcal{O}$. The search parameters α are discarded, and the resulting network is produced from an evolutionary search by evaluating on the re-training validation set. This method performs poorly in machine translation, as shown in Table 1. We find that sampling-based methods favor large-kernel convolutions and that the resulting architectures tend to generate repetitive sentences.

We also experiment with increased split numbers. As shown in Table 2, an increased split number hurts the translation performance. The best results are all achieved by the smallest split. Also, the search process is harder to converge as the search space becomes too large. The re-training and inference speed will slow down when increasing the split number because more recurrence are introduced in the calculation as shown in Eq. (3).

In the following sections, we try the best search result (DARTSformer + search 2 layers + 2 split + max pooling) in various sequence-to-sequence tasks to see its generalization ability. We show this searched architecture in Figure 4.

4.2 Performance of DARTSformer on Other Datasets

First, we train DARTSformer with a base model size on three translation tasks in Section 3.1. We would like to see whether DARTSformer only performs well on the task used for architecture search or generalizes to related tasks. Second, we scale up the model size and the batch size to see whether the performance gain of DARTSformer still exists. We compare DARTSformer with standard Transformers and Evolved Transformers with similar model sizes. Following Vaswani et al. (2017), the parameter size is around 62.5M/214.7M for the base model and big model, respectively. To match the settings of So et al. (2019) when training big models, we increase the dropout rate to 0.3 and the learning rate to 1×10^{-3}. We also accumulate gradients for two batches.

Models	En-De	En-Fr	En-Cs
Transformer	27.7	40.0	27.0
ET (So et al., 2019)	28.2	**40.6**	27.6
DARTSformer	**28.4**	40.1	**27.9**

(a) Comparison for Base Model Size

Models	En-De	En-Fr	En-Cs
Transformer	29.1	41.2	28.1
ET (So et al., 2019)	29.3	**41.3**	28.2
DARTSformer	**29.8**	**41.3**	**28.5**

(b) Comparison for Big Model Size

Table 4: BLEU scores on WMT'14 translation tasks. ET is the Evolved Transformer. We use the best search result from different DARTSformer search setups.

Results are shown in Table 4. At the base model size, DARTSformer steadily outperforms standard Transformers. We achieved the same translation quality (28.4 BLEU, reported by Vaswani et al. (2017)) as the original big Transformer in WMT'14 En-De, with about 69% fewer parameters. Also, the maximum BLEU gain is 0.9 in WMT'14 En-Cs, which is not the dataset we conduct our architecture search on. As for Evolved Transformers, we surpass their performance in two out of three datasets, and our search algorithm is more computationally efficient. At the big model size, DARTSformer exceeds both standard Transformers and Evolved Transformers, which indicates the good generalization ability of DARTSformer.

4.3 Performance of DARTSformer vs. Parameter Size

In Section 4.2, DARTSformer consistently improves the performance with a model size comparable to the base and big Transformers. We are wondering whether the performance increase exists with smaller model sizes. We experiment with a spectrum of model sizes for standard Transformers and DARTSformer on WMT'14 En-De. Specifically, we use four embedding sizes for standard Transformers, [small:128, medium:256, base:512, big:1024], where its hidden size is identical to the embedding size. We also adjust the model size of DARTSformer accordingly. For base and big models, we use the results from Section 4.2. For small and medium models, we set the learning rate to 5×10^{-4}, the dropout probability to 0.1, and update the model parameters for 200,000 steps on the same 8 NVIDIA V100 hardware.

Figure 5 shows the results for both architectures. DARTSformer performs better than standard Transformers at all sizes. The BLEU increase is [1.3/0.9/0.7/0.7] for [small/medium/base/big] models. An interesting fact is that the performance gap between two models tends to be smaller as we increase the model size, which is also observed in So

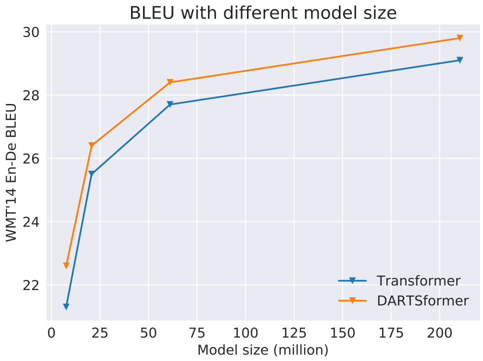

Figure 5: BLEU comparison between DARTSformer and standard Transformers with different model sizes.

Search Settings	e	d	**BLEU**
Tiny	128	120	24.2
Small	128	240	26.3
Medium	256	480	27.5
DARTSformer	256	960	**28.4**
DARTS + Transformer	320	320	27.7
Transformer	-	-	27.7

Table 5: BLEU scores of DARTS with different search hidden sizes on WMT'14 En-De test set. All searched architectures are re-trained with a parameter size similar to DARTSformer.

et al. (2019). Based on this observation, DARTSformer is more pronounced for environments with resource limitations, such as mobile phones. A possible reason for the decreased performance gap at larger model sizes is that the effect of overfitting becomes more important. We expect that some data augmentation skills (Sennrich et al., 2015; Edunov et al., 2018; Qu et al., 2020) might be of help.

4.4 The Impact of Search Hidden Size

The main motivation for our presented method is that we want to search in a large hidden size to reduce the performance gap between searching and re-training. However, whether this gap exists needs rigorous validation. Otherwise, it would suffice to instead use a small hidden size d in architecture search, and then increase d after search for training the actual model. We experiment with 4 search hidden sizes, namely, $e = 128, d = 120$ (tiny), $e = 128, d = 240$ (small), $e = 256, d = 480$ (medium), $e = 256, d = 960$ (DARTSformer). e is the word embedding size and d is the hidden size as described in Section 3.2. After obtaining the searched model, we set the model size to $e = 256, d = 960$, and re-train it on WMT'14 En-De.

The results are summarized in Table 5, which clearly shows that the translation quality is improving as the search hidden size gets larger. Also, note that when searching with tiny, small and medium settings, the final BLEU scores fall behind that of standard transformers. We argue that if one wants to evaluate the searched model in large model sizes, it is important to search with large hidden sizes. Further more, we directly apply DARTS with standard transformer as the backbone model. We set $e = 320, d = 320$. A larger search hidden size often causes memory failure due to the storage of many intermediate hidden states. As shown in Table 5, We can see that searching with a small hidden size yields no performance gain on the standard transformer.

5 Related Work

Architecture Search The field of neural architecture search (NAS) has seen advances in recent years. In the early stage, researchers focus on the reinforcement learning-based approaches (Baker et al., 2016; Zoph and Le, 2016; Cai et al., 2018a; Zhong et al., 2018) and evolution-based approaches (Liu et al., 2017; Real et al., 2017; Miikkulainen et al., 2019; So et al., 2019; Wang et al., 2020). These methods can produce architectures that outperform human-designed ones (Zoph et al., 2018; Real et al., 2019). However, the computational cost is almost unbearable since it needs to fully train and evaluate every candidate network found in the search process. Weight sharing (Brock et al., 2017; Pham et al., 2018) is a practical solution where a super network is trained, and its sub-graphs form the search space. Liu et al. (2018) proposed DARTS to use search parameters together with a super network, which allows searching with gradient descent. Gradient-based methods (Cai et al., 2018b; Xie et al., 2018; Chen et al., 2019; Xu et al., 2019; Yao et al., 2020) attracts researchers' attention since it is computationally efficient and easy to implement. We base our method on DARTS and take one step further to reduce the memory consumption of training the super network. Another recent trend is the one-stage NAS (Cai et al., 2019; Mei et al., 2019; Hu et al., 2020; Yang et al., 2020). Many NAS algorithms are in two stages. In the first stage, one searches for a good candidate network. In the second stage, the resulting network is re-initialized and re-trained. One-stage NAS tries to search and optimize the network weights simultaneously. After searching, one can have a ready-to-run network. We use a simple one-stage NAS algorithm (Guo et al., 2020) as a baseline in Section 4.1.

Reversible networks The idea of reversible networks is first introduced by RevNets (Gomez et al., 2017). Later on, Jacobsen et al. (2018); Chang et al. (2018); Behrmann et al. (2019) invented different reversible architectures based on the ResNet (He et al., 2016). MacKay et al. (2018) extended RevNets to the recurrent network, which is particularly memory-efficient. Bai et al. (2019, 2020) conducted experiments with reversible Transformers by fixed point iteration. Kitaev et al. (2020) combined local sensitive hashing attention with reversible transformers to save memory in training with long sequences. An important application of reversible networks is the flow-based models (Kingma and Dhariwal, 2018; Huang et al., 2018; Tran et al., 2019). For sequence tasks, Ma et al. (2019) achieved success in non-autoregressive machine translation.

6 Conclusion

We have proposed a memory-efficient differentiable architecture search (DARTS) method on sequence-to-sequence tasks. In particular, we have first devised a multi-split reversible network whose intermediate layer outputs can be reconstructed from top to bottom by the last layer's output. We have then combined this reversible network with DARTS and developed a backpropagation-with-reconstruction algorithm to significantly relieve the memory burden during the gradient-based architecture search process. We have validated the best searched architecture on three translation tasks.

Our method consistently outperforms standard Transformers. We can achieve the same BLEU score as the original big Transformer does with 69% fewer parameters. At a large model size, we surpass Evolved Transformers with a search cost lower by an order of magnitude. Our method is generic to handle other architectures, and we plan to explore more tasks in the future.

Acknowledgments

Yuekai Zhao and Zhihua Zhang have been supported by the Beijing Natural Science Foundation (Z190001), National Key Research and Development Project of China (No. 2018AAA0101004), and Beijing Academy of Artificial Intelligence (BAAI).

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2019. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, pages 690–701.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. 2020. Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. 2019. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 573–582. PMLR.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2017. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. 2018a. Path-level network transformation for efficient architecture search. In *International Conference on Machine Learning*, pages 678–687. PMLR.

Han Cai, Ligeng Zhu, and Song Han. 2018b. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

B. Chang, L. Meng, E. Haber, Lars Ruthotto, David Begert, and E. Holtham. 2018. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI*.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. 2017. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, pages 2214–2224.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. 2020. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12084–12092.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. 2018. Neural autoregressive flows. volume 80 of *Proceedings of Machine Learning Research*, pages 2078–2087, Stockholmsmässan, Stockholm Sweden. PMLR.

Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. 2018. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*.

Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. Improved differentiable architecture search for language modeling and named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3576–3581.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10215–10224. Curran Associates, Inc.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. 2020. Neural architecture transfer. *arXiv preprint arXiv:2005.05859*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4282–4292, Hong Kong, China. Association for Computational Linguistics.

Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger Grosse. 2018. Reversible recurrent neural networks. In *Neural Information Processing Systems (NeurIPS)*.

Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. 2019. Atomnas: Fine-grained end-to-end neural architecture search. *arXiv preprint arXiv:1912.09640*.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pages 293–312. Elsevier.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR.

Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. 2020. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

David R So, Chen Liang, and Quoc V Le. 2019. The evolved transformer. *arXiv preprint arXiv:1901.11117*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. 2019. Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems*, pages 14719–14728.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7675–7688, Online. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2019. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*.

Yibo Yang, Hongyang Li, Shan You, Fei Wang, Chen Qian, and Zhouchen Lin. 2020. Ista-nas: Efficient and consistent neural architecture search by sparse coding. *arXiv preprint arXiv:2010.06176*.

Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. 2020. Efficient neural architecture search via proximal iterations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6664–6671.

Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.

On the Copying Behaviors of Pre-Training for Neural Machine Translation

Xuebo Liu[1]*, **Longyue Wang**[2], **Derek F. Wong**[1], **Liang Ding**[3],
Lidia S. Chao[1], **Shuming Shi**[2] **and Zhaopeng Tu**[2]
[1]NLP²CT Lab, Department of Computer and Information Science, University of Macau
[2]Tencent AI Lab [3]The University of Sydney
`nlp2ct.xuebo@gmail.com`, `{derekfw,lidiasc}@um.edu.com`
`{vinnylywang,shumingshi,zptu}@tencent.com`
`ldin3097@sydney.edu.au`

Abstract

Previous studies have shown that initializing neural machine translation (NMT) models with the pre-trained language models (LM) can speed up the model training and boost the model performance. In this work, we identify a critical side-effect of pre-training for NMT, which is due to the discrepancy between the training objectives of LM-based pre-training and NMT. Since the LM objective learns to reconstruct a few source tokens and copy most of them, the pre-training initialization would affect the copying behaviors of NMT models. We provide a quantitative analysis of copying behaviors by introducing a metric called *copying ratio*, which empirically shows that pre-training based NMT models have a larger copying ratio than the standard one. In response to this problem, we propose a simple and effective method named *copying penalty* to control the copying behaviors in decoding. Extensive experiments on both in-domain and out-of-domain benchmarks show that the copying penalty method consistently improves translation performance by controlling copying behaviors for pre-training based NMT models. Source code is freely available at `https://github.com/SunbowLiu/CopyingPenalty`.

1 Introduction

Self-supervised pre-training (Devlin et al., 2019; Song et al., 2019), which acquires general knowledge from a large amount of unlabeled data to help *better* and *faster* learning downstream tasks, has an intuitive appeal for neural machine translation (NMT; Bahdanau et al., 2015; Vaswani et al., 2017). One direct way to utilize pre-trained knowledge is initializing the NMT model with a pre-trained language model (LM) before training it on parallel data (Conneau and Lample, 2019; Liu et al., 2020). As a range of surface, syntactic and semantic information has been encoded in the initialized parameters (Jawahar et al., 2019; Goldberg, 2019), they are expected to bring benefits to NMT models and hence the translation quality.

LM Pre-Training: $\mathcal{L}_{\text{PT}} = -\log P(\mathbf{x}\vert\widetilde{\mathbf{x}})$	
Source	Military _ Field Marshal Hussein _ _ in attendance.
Target	Military ruler Field Marshal Hussein Tantawi was in attendance.
NMT Training: $\mathcal{L}_{\text{NMT}} = -\log P(\mathbf{y}\vert\mathbf{x})$	
Source	Military ruler Field Marshal Hussein Tantawi was in attendance.
Target	Der Militärführer Feldmarschall Hussein Tantawi war anwesend.

Table 1: Training objective gap between Seq2Seq LM pre-training and NMT training. LM learns to reconstruct a few source tokens and copy most of them, while NMT learns more translation rather than copying. Underlines denote artificial noises, and highlights indicate expected copying tokens.

However, there is a discrepancy between the training objective of sequence-to-sequence LM pre-training and that of NMT training. As shown in Table 1, LM learns to reconstruct all source tokens with some noises, while NMT learns to translate most source tokens and copy few of them. Knowles and Koehn (2018) and Liu et al. (2020) show that LM pre-training requires to copy ~65% of tokens, while NMT training only needs to copy <10%. We believe that unexpected knowledge can be propagated to the NMT model via pre-training, which may bias NMT models to mistakenly copy source tokens to the target side. For example, the source word "Field Marshal" might be mistakenly copied to the target side by pre-training based NMT models, since such copying behaviors can be learned in

---

*Work was done when Xuebo Liu and Liang Ding were interning at Tencent AI Lab.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4265–4275
August 1–6, 2021.

the pre-training stage.

In this paper, we first validate the change of copying behaviors in NMT models initialized with the pre-training weights. To this end, we propose a metric named *copying ratio* to quantitatively measure the extent of copying behaviors of NMT models. Experimental results on the WMT14 En-De data show that the NMT model with pre-training improves translation performance at the cost of introducing more copying predictions. Analyses on model training show that the NMT model with pre-training attempts to forget the copying behaviors transferred from pre-training, while the vanilla NMT model learns in the opposite way. Due to the dominating copying behaviors in the pre-training, the copying ratio of pre-training based NMT model (i.e., 10.8%) is much higher than that of the vanilla NMT model (i.e., 9.3%). Extensive analyses show that higher copying ratios severely hurt sentence fluency and word accuracy in translations, particularly for the translation of proper nouns, establishing the necessity for controlling the copying behaviors of NMT models.

To tackle this problem, we propose a simple and effective *copying penalty* to control the copying behaviors in inference, which requires no modification to model architectures and training algorithms. Specifically, we introduce a new regularizing term to the prediction at each time step, which guides the model to copy source tokens only when the model is highly confident. Experimental results on the WMT14 English-German and the OPUS German-English benchmark demonstrate that the proposed approach can significantly control copying behaviors in NMT models, making the model more accurately generate copying tokens.

Our contributions are summarized as follows:

- We reveal a critical side-effect of pre-training for NMT, where pre-training introduces more copying behaviors into NMT outputs.
- We propose a simple and effective *copying penalty* to further improve the performance of NMT models with pre-training by controlling copying behaviors in generated translation.
- We find that the domain containing a large number of copying tokens (e.g., the IT) benefits more from the proposed copying penalty.

2 Observing Copying Behavior Changes

The fact is that some source words are excessively copied by NMT models from the source to the target side instead of being translated, which leads to a high copying ratio in NMT outputs. In this section, we first propose a metric to measure the copying ratio of model predictions. Second, we quantitatively investigate the effect of pre-training on NMT in the perspective of copying behaviors. We expect to provide more evidence for controlling the copying behaviors of NMT models.

2.1 Experimental Setup

Data We conducted experiments on the widely-used WMT14 English-German benchmark. We used the processed data provided by Vaswani et al. (2017), which consists of 4.5M sentence pairs.[1] We used all the training data for model training. The validation set is newstest2013 of 3,000 examples and the test set is newstest2014 of 3,003 examples.

Models and Settings We implemented all the models by the open-sourced toolkit `fairseq` (Ott et al., 2019).[2] We used 8 V100 GPUs for the experiments. We mainly compared two models: 1) RANDOM, which is a vanilla NMT model whose weights are randomly initialized without pre-training; and 2) PRETRAINED, an NMT model using the weights of pre-trained `mBART.cc25`[3] for parameter initialization, which has shown its usability and reliability for translation tasks (Tran et al., 2020; Tang et al., 2020).

For the training of RANDOM, we used the Transformer *big* setting of Ott et al. (2018b) with a huge training batch size of 460K tokens.[4] For PRETRAINED, we fine-tuned on the pre-trained `mBART.cc25` with a training batch size of 131K tokens. The hyperparameters keep the same with RANDOM except the 0.2 label smoothing, 2500 warm-up steps, and 1e-4 maximum learning rate.

Evaluation For each model, we selected the checkpoint with the lowest perplexity on the validation set for testing. The beam size is 5 and the length penalty is 0.6. In addition to report-

---

[1] https://drive.google.com/uc?id=0B_bZck-ksdkpM25jRUN2X2UxMm8

[2] https://github.com/pytorch/fairseq

[3] https://github.com/pytorch/fairseq/tree/master/examples/mbart

[4] https://github.com/pytorch/fairseq/blob/master/examples/scaling_nmt/README.md#3-train-a-model

Source	Military ruler Field *Marshal* Hussein Tantawi was in attendance.
Target	Der Militärführer Feldmarschall Hussein Tantawi war anwesend.
RANDOM	Anwesend war der Militärmachthaber Feldmarschall Hussein Tantawi.
PRETRAINED	Militärischer Feldherr *Marshal* Hussein Tantawi war anwesend.

Table 2: Translation from English to German. The words in color denote the copying tokens of which blue denotes right copies and *red* denotes copying errors.

ing the commonly-used 4-gram BLEU score (Papineni et al., 2002), we also report Translation Error Rate (TER) (Snover et al., 2006) to better capture the translation performance of unigrams, which more directly reflects the copying behaviors of NMT models. Both the scores are calculated by `sacrebleu` (Post, 2018) with de-tokenized text and unmodified references.[5,6]

2.2 Copying Ratio

Ratio To measure the extent of the copying behaviors in NMT models, we calculate the ratio of copying tokens in translation outputs:

$$\text{Ratio} = \frac{\sum_{i=1}^{I} \text{count}(\text{copying token})}{\sum_{i=1}^{I} \text{count}(\text{token})} \tag{1}$$

where I denotes the total number of sentences in the test set. We count the number of "copying token" by comparing each input and output sentence pair. The denominator is the total number of tokens in output sentences. In general, higher Ratio values indicate more copying behaviors produced by the NMT model, and vice versa.

Copying Error Rate (CER) To further analyze the copying problem in NMT models, we propose to calculate the rate of incorrect copying tokens among all copied ones:

$$\text{CER} = \frac{\sum_{i=1}^{I} \text{count}(\text{copying error})}{\sum_{i=1}^{I} \text{count}(\text{copying token})} \tag{2}$$

where we count the number of "copying error" by checking whether the copying tokens are included in its reference sentence. The CER is expected to be zero, which indicates that all copying tokens are correct. Table 2 gives an example. In experiments, Ratio and CER are computed based on words rather than sub-words. We further filter all punctuations, which are similar across different languages.

	Performance		Copying		
	BLEU	TER	Ratio	CER	#S
ORACLE	-	-	8.5%	-	9
RANDOM	28.3	60.7	9.3%	17.4	20
PRETRAINED	29.4	59.4	10.8%	27.6	50

Table 3: Results on the WMT14 En-De test set. ORACLE denotes the statistics on the reference. "#S" denotes the number of instances whose overlaps between the source and target exceeding 50% (Ott et al., 2018a). Although PRETRAINED gains better model performance than RANDOM, it also excessively copies tokens from the source.

Model Performance We compare the performance and copying behaviors of the final models in Table 3. The results show that although PRETRAINED improves the overall performance in terms of BLEU and TER scores, it tends to generate more copying errors, limiting its further improvement. In the following part, we probe into the essence of the copying behaviors via carefully designed experiments.

2.3 Learning Curves of Copying Behaviors

We analyze copying behaviors in learning dynamics. Specifically, we translate the test set using intermediate checkpoints at different training steps, and then compute corresponding Ratio and CER values. We compare RANDOM and PRETRAINED, and plot their learning curves in Figure 1.

Ratio Two models behave quite differently in the early stages of training. Taking Step 100 for instance, PRETRAINED copies 89% tokens while RANDOM does not generate any copying tokens. This demonstrates that the copying habit in the pretrained model is transferred to NMT models. As training proceeds, the copying behaviors of PRETRAINED are heavily suppressed, resulting in a rapid drop in Ratio. On the contrary, RANDOM is able to quickly learn copying from scratch, leading to an upward trend. After learning curves become stable, PRETRAINED performs more copying behaviors than RANDOM (10.8% vs. 9.3% Ratio).

CER In general, the results of CER show similar trends to those observed in Ratio. In the beginning, the CER of PRETRAINED is extremely high (i.e.,

---

[5] BLEU+c.mixed+#.1+s.exp+tok.13a+v.1.4.14

[6] TER+lang.en-de+tok.tercom-nonorm-punct-noasian-uncased+version.1.4.14

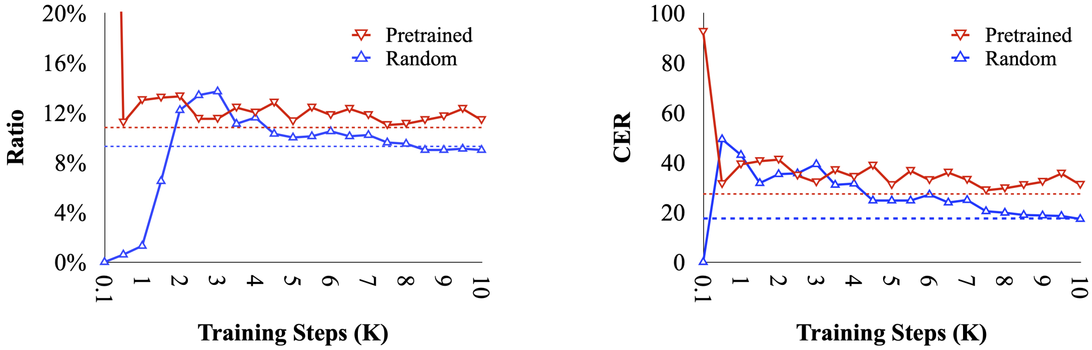

Figure 1: Copying ratio (left) and CER score (right) of RANDOM and PRETRAINED at different training steps. Reference lines report the values of the final models gaining the lowest validation perplexities. PRETRAINED gains 89.4% copying ratio at 0.1K step, which is omitted for better display clarity. PRETRAINED learns to forget its copying behaviors by reducing the copying ratio from 89.4% to 10.8% and CER from 92.4 to 27.6, while RANDOM learns copying from scratch by increasing the copying ratio and CER from 0 to 9.3% and 17.4, respectively.

	PPL	Rand>Pre		Rand<Pre	
		Ratio	CER	Ratio	CER
RANDOM	60.8	10.0%	18.5	8.7%	17.4
PRETRAINED	95.1	9.5%	14.9	12.5%	39.7

Table 4: Sentence perplexity on the test set. PRETRAINED's translation of worse perplexity (i.e., "Rand<Pre") also contains the higher Ratio (12.5%) and CER (39.7) scores.

92.4), revealing that most of the copying tokens are incorrect. The reason behind this phenomenon is that pre-trained models are accustomed to copying source words but the habit is overly transferred to the downstream translation models. The interesting finding is that RANDOM also makes more mistakes on copying at the early training stage. Finally, the error rate of PRETRAINED is much higher than that of RANDOM (27.6 vs. 17.4), showing that pre-trained models indeed expose harmful knowledge to NMT models.

Learning curves of two kinds of models perform in opposite ways: RANDOM learns copying from scratch while PRETRAINED tries to forget this behavior. As a result, PRETRAINED copies more source tokens than RANDOM and suffers severe copying errors. This motivates us to further investigate the effects of copying behaviors on NMT models in terms of translation quality.

2.4 Effect of Copying Ratio

Sentence Fluency The copying tokens from the source usually contain some tokens that do not belong to the target language, which might hurt the fluency of generated translations. Starting from this intuition, we use an external language model (Ng et al., 2019)[7] trained on in-domain data to evaluate the fluency of translation outputs. As shown in Table 4, Random achieves a much better perplexity than mBART (60.8 vs. 95.1 PPL), demonstrating that the NMT model with pre-training generates less fluent sentences than that trained from scratch.

To take a closer look at the fluency gap, we divide outputs of each model into two subsets: sentences with better or worse perplexity by comparing RANDOM and PRETRAINED. As seen, the fluency of translation is related to copying ratio and errors. The sentences with higher Ratio and CER scores tend to be less fluent. Taking PRETRAINED's worse subset for example (Rand<Pre), it gains a 12.5% Ratio and 39.7 CER scores, confirming that excessive copying behaviors lead to negative effects in terms of translation fluency.

Word Accuracy We also give a word-level analysis by bucketing copying tokens according to part-of-speech (POS) tags and calculate Ratio and CER in each type. In experiment, we employ Stanford POS tagger with the `german-ud.tagger` model to automatically label output sentences (Toutanova et al., 2003). Table 5 lists the results. The "Oracle" (Row 1) denotes the statistics by comparing the source input and its reference. As seen, most copying operations should occur in the type of proper noun (PROPN). This type occupies 5.7% Ratios, followed by adposition (ADP), numeral (NUM), noun (NOUN), and other types (Others).

Compared with RANDOM, we observe that the

---

[7] https://dl.fbaipublicfiles.com/fairseq/models/lm/wmt19.de.tar.gz

	Total		PROPN		ADP		NUM		NOUN		Others	
	Ratio	**CER***	**Ratio**	**CER***	**Ratio**	**CER***	**Ratio**	**CER***	**Ratio**	**CER***	**Ratio**	**CER***
Oracle	8.5%	-	5.7%	-	1.1%	-	0.8%	-	0.5%	-	0.3%	-
Random	9.3%	17.4	6.3%	14.5	1.3%	25.8	0.9%	14.2	0.5%	21.3	0.3%	44.4
Pretrained	10.8%	27.6	7.5%	27.3	1.3%	24.9	0.9%	14.0	0.6%	21.0	0.5%	63.8
Δ	**+1.5%**	**+10.2**	**+1.2%**	**+12.8**	0%	-0.9	0%	-0.2	+0.1%	+0.3	+0.2%	**+19.4**

Table 5: Copying behaviors by part-of-speech (POS) bucket on the WMT14 En-De task. "Oracle" denotes the statistics in the reference. "CER*" denotes only using the copying tokens belonging to each POS category for the CER calculation. Δ denotes the changes from the Random to Pretrained, in which the significant ones are **bold**. Most of the copying tokens are found in translating proper nouns (PROPN) in Pretrained.

increase of Ratio for Pretrained mainly attributes to copying PROPN words (+1.2%). In addition, Pretrained generates more copying errors (+10.2), especially on PROPN and Others types (+12.8 and +19.4). These results reveal that it is necessary to pay more attention to proper nouns on controlling copying behaviors for NMT.

3 Controlling Copying Behaviors

Based on the above experiments, we prove that pre-training indeed changes the copying behaviors of NMT models, hurting the sentence fluency and word accuracy of generated translations. To alleviate this issue, we propose a simple and effective method *copying penalty* to make the copying behaviors in NMT controllable.

3.1 Copying Penalty (CP)

To control copying behaviors in NMT, an intuitive way is generating copying tokens only when the model is of high confidence. To this aim, we propose to modify the probability distribution predicted by the NMT model, decreasing the predicting probability of the tokens also occurred in the source (i.e., weakening the model confidence of making copying predictions). In this way, for those predictions are wavering between copying and translating, the model is more likely to translate them, and thus only those confident copying tokens will be retained. Specifically, during inference, the predicting probability of t-th time step is as follows:

$$P(y_j|y_{<j}, \mathbf{x}) \in \mathbb{R}^{\mathcal{V}} = \text{softmax}(\mathbf{y}_t) \tag{3}$$

where $P(y_j|y_{<j}, \mathbf{x})$ denotes the probability over the whole target vocabulary and $\mathbf{y}_t$ denotes the decoder output of t-th time step. The search algorithm (e.g., beam search) will take this probability distribution as a candidate to find the final translation of the source sentence.

Copying penalty regularizes the prediction probability of each time step by element-wisely multiplying a new constraint $\text{CP} \in \mathbb{R}^{\mathcal{V}}$:

$$\text{CP} = \begin{cases} 1, & y_j \notin \mathbf{x}/\mathcal{C}_{\text{punc}} \\ \alpha, & y_j \in \mathbf{x}/\mathcal{C}_{\text{punc}} \end{cases} \tag{4}$$

where α is a hyperparameter to control the penalty which can be tuned on the development set, similar to length penalty (Wu et al., 2016). $\mathbf{x}/\mathcal{C}_{\text{punc}}$ denotes the set of sources tokens excluding punctuation and `eos`, which means that the prediction probabilities of punctuation and `eos` will not be penalized. For those predictions not belonging to the source, their probabilities keep the same. But for those predictions that are copied from the source, their probabilities will be α times as large/small as before and the model will be more/less likely to choose them as candidates for searching.

The proposed method is simple and effective: 1) It does not change the model architecture and does not need any additional model training, thus no parameters needed to be newly introduced; 2) Its implementation only requires some low-cost matrix operations during model inference, slightly slowing the decoding speed; and 3) It can significantly control the overall copying ratio of the model predictions, making the model accurately generate copying tokens, as shown in the following sections.

Effect of Copying Penalty Figure 2 depicts the changes of the copying ratio and CER scores when setting CP to different values on the test data. When setting CP smaller than 1 (i.e., punishing copying tokens), only those confident copying predictions will be made, and thus reducing both the copying ratio and CER scores. Conversely, setting CP larger than 1 makes the model generate more copying tokens even some of them are of low confidence, thus both the copying ratios and CER scores increase. Similar to length penalty (Wu et al., 2016), we also

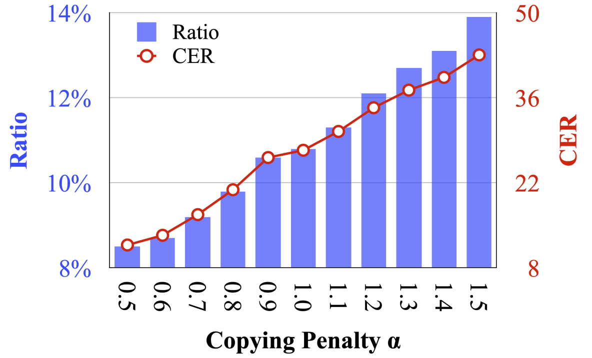

Figure 2: Copying ratios and CER scores by different copying penalties in Pretrained. When setting CP smaller than 1 (i.e., penalizing copying), both the copying ratios and CER scores decrease, and vice versa.

tuned CP on the development data and found that setting CP to 0.7 wins the best BLEU score. Therefore, we used this value for decoding test data in the following experiments.

Empirical results also show that the copying penalty is very efficient. When evaluated by a single 32GB V100 GPU card, the inference speed of Pretrained is about 612 token/s, and that of the model with CP is 607 token/s. The extra latency of the copying penalty is negligible.

3.2 Main Results

Table 6 lists the overall results of the model performance and copying behavior. By looking at the part of evaluating the all test data of 3,003 sentences, the results first confirm the effectiveness of Pretrained that can consistently improve the model performance in terms of BLEU and TER scores. However, the introduction of pre-trained knowledge also brings more copying properties to the model that increases the copying ratio, copying errors, and the number of copying sentences at the same time. Thanks to the introduction of the copying penalty, the model successfully alleviates the copying errors (i.e., reducing the CER score from 27.6 to 16.8), making them be on par with Random, and thus further improve the BLEU and TER scores over the strong Pretrained.

To better understand how copying behaviors affect model performance, we split the test data into two subsets: HasCopy and NoCopy. One intuitive assumption is that copying errors would significantly hurt the performance of the NoCopy data since every copying token in the translation is a copying error. The results confirm our assumption that Pretrained can only improve limited model performance on the NoCopy data (e.g., improving the TER scores from 62.9 to 62.7), but with the copying penalty, the copying errors less occur in Pretrained (i.e., reducing the copying ratio from 2.7 to 1.2 and copying sentences from 18 to 3) and thus, better model performance.

	Performance		**Copying**		
	BLEU	**TER**	**Ratio**	**CER**	**#S**
All (3,003 sentences)					
Oracle	-	-	8.5%	-	9
Random	28.3	60.7	9.3%	17.4	**20**
Pretrained	29.4	59.4	10.8%	27.6	50
+CP	**29.6**	**59.0**	**9.2%**	**16.8**	**20**
HasCopy (1,774 sentences)					
Oracle	-	-	12.5%	-	9
Random	29.7	59.4	13.4%	13.8	**15**
Pretrained	30.9	57.8	14.8%	20.7	32
+CP	**31.0**	**57.5**	**13.0%**	**13.2**	17
NoCopy (1,229 sentences)					
Oracle	-	-	0%	-	0
Random	25.4	62.9	**1.2%**	-	5
Pretrained	26.2	62.7	2.7%	-	18
+CP	**26.7**	**62.0**	**1.2%**	-	**3**

Table 6: Overall results on the WMT14 En-De test set. "Oracle" denotes the statistics on the reference. "+CP" denotes using the proposed copying penalty method for inference. "#S" denotes the number of instances whose overlaps between the source and target exceeding 50%. "HasCopy" denotes evaluating on the sampled test set containing copies between the source and target, while "NoCopy" denotes evaluating on the remained set without any copying. CER score is not applicable in NoCopy as all the copying tokens are copying errors.

Sentence Fluency The copying penalty improves sentence fluency. In §2.4, we show that the perplexity of Pretrained (95.1) is worse than that of Random (60.9). However, after introducing the copying penalty into Pretrained, the perplexity gets a significant drop from 95.1 to 62.3, which can be on par with Random. This confirms our hypothesis that more copying behaviors hurt NMT in terms of translation fluency, and controlling copying behaviors can make the model generate fluent outputs.

Word Accuracy The copying penalty enhances the translation of PROPN. As shown in Table 7, the copying penalty improves the translations of proper nouns, reducing the copying ratio from 7.5%

	#Num	Total		PROPN	
		Ratio	CER	Ratio	CER*
All	3,003	10.8%	27.6	7.5%	27.3
+CP		9.2%	16.8	6.3%	15.1
Tgt-Ori	1,500	10.1%	17.5	6.6%	15.6
+CP		9.4%	12.6	6.1%	10.1
Src-Ori	1,503	11.4%	34.4	8.3%	34.8
+CP		9.1%	20.3	6.4%	18.9

Table 7: Copying behaviors of the source original and target original text in PRETRAINED. “#Num” denotes the total number of sentences in each test set. The translation of source original text contains more copying tokens, and CP can reduce the copying ratio.

to 6.3% and the CER score from 27.3 to 15.1.

To make headway into the translation of proper nouns, we further investigate the translations from various sources that usually show the large difference in the number of proper nouns (Lembersky et al., 2011). Specifically, we investigate the two kinds of sentence pairs in the WMT14 En-De test set: 1) the *source original text* (Src-Ori) that originated in English and was human-translated into German; 2) the *target original text* (Tgt-Ori) was translated in the opposite direction, originating in German with manual translation into English.

Zhang and Toral (2019) conclude that Tgt-Ori is artificially easier to translate, resulted in inflated scores for NMT models. Our results of PRETRAINED positively support this conclusion that translating Tgt-Ori makes fewer copying errors and this might be a reason why it can win a better translation performance. However, by looking at the last row, Src-Ori suffers from serious copying errors especially in translating proper nouns, making it harder to translate. Encouragingly, the copying penalty nicely reduces the copying ratios and copying errors in translating both Src-Ori and Tgt-Ori. These results further reveal the importance of controlling copying behaviors in NMT models since translating source original text is the core task of most NMT systems (Graham et al., 2020).

The above results have shown that copying errors worsen the translation of Src-Ori. To support the claim, we further investigate the effects of varying degrees of copying errors in the translations of Src-Ori and Tgt-Ori. Figure 3 shows the change of BLEU scores with different copying penalties. Clearly, the translation of Src-Ori is more sensitive to copying errors and thus the BLEU scores get a sharp degradation when setting the copying penalty greater than 1, which verifies our claim.

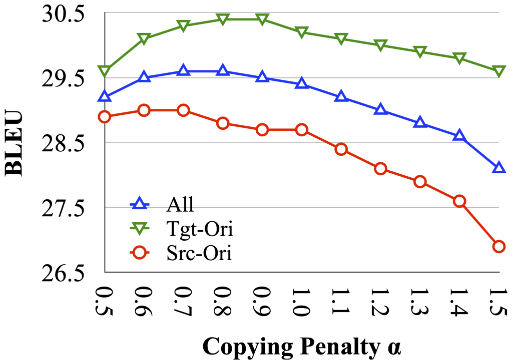

Figure 3: BLEU scores of different copying penalties in PRETRAINED. Penalizing copying (i.e., $\alpha < 1$) brings benefits to the translations of various sources. Translating source original sentences is more sensitive to copying behaviors, leading to a larger score degradation when encouraging copying (i.e., $\alpha > 1$).

3.3 Out-of-domain Robustness

Improving out-of-domain (OOD) robustness is one of the benefits of pre-training for NLP tasks (Hendrycks et al., 2020; Tu et al., 2020), but the OOD sentences usually contain some low-frequency proper nouns which are hard to translate (Ding et al., 2021). In this part, we take the first step towards understanding how pre-training affects the OOD robustness of NMT models.

Setup We followed Müller et al. (2020) to preprocess all the used data sets.[8] We served the medical domain as the training domain (i.e., using the data from the medical domain for model training and validation), which consists of 1.1M training examples and 2,000 validation examples. The test set of the medical domain contains 1,691 examples, while the test sets of the IT, Koran, law, and subtitle domains are with 2,000 examples respectively. For training RANDOM, we used the Transformer *base* setting with 32K batch size. The model dropout is set to 0.3, while the dropouts for attention and inner-FFN activation are set to 0.2. For PRETRAINED, apart from using 32K batch size, the other hyperparameters follow the training of the WMT14 English-German task. The beam size is set to 5 and the length penalty is set to 1.4. We evaluated the model performance on the OOD test sets from IT, Koran, Law, and Subtitles domains,

---

[8] `https://github.com/ZurichNLP/domain-robustness/blob/master/scripts/preprocessing/preprocess_de_en.sh`

	InD	OutD				
	Med.	Avg.	IT	Kor.	Law	Sub.
EXISTING	61.5	11.7	17.1	1.1	25.3	3.4
+REG.	60.8	13.1	-	-	-	-
RANDOM	60.5	11.4	20.5	1.1	20.5	3.4
PRETRAINED	63.1	17.6	29.8	2.4	31.0	7.3
+CP	**63.2**	**18.3**	**31.5**	**2.5**	**31.1**	**7.9**

Table 8: BLEU scores on the OPUS De-En translation task trained on the in-domain medical data. "Existing" and "+Reg." denote the results of baseline and regularization method from Müller et al. (2020). CP can significantly improve the translation performance of the IT domain that needs to copy more tokens from the source.

and the averaged BLEU scores can be seen as the OOD robustness of each NMT model.

Results Table 8 lists the results. Clearly, PRETRAINED substantially improves the performances of in-domain translation and OOD robustness, increasing the in-domain BLEU scores from 60.5 to 63.1 and the OOD BLEU scores from 11.4 to 17.6 respectively. The copying penalty can further improve the OOD robustness of PRETRAINED that consistently improves the model performance of each OOD test set. The copying penalty can even remarkably enhance PRETRAINED in translating the sentences from the IT domain (when setting CP to 1.2). One possible reason is that the IT domain needs to copy more tokens from the source sentence than translating sentences from other domains, thus the copying penalty can play a greater role and bring a significant performance boost. This also verifies the effectiveness of the copying penalty.

4 Related Work

4.1 Pre-Training for NMT

Recently, pre-training has been shown useful for transferring general knowledge to specific downstream tasks, including text classification, question answering and natural language inference (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019). Compared with training from scratch, fine-tuning a pre-trained model on downstream datasets usually pushes state-of-the-art performances, while reducing computational and labeling costs.

Previous studies mainly investigate the effect of pre-training on NMT from two perspectives: 1) *knowledge extraction*, where a fixed pre-trained model is used to encode input sequences into features which are then fed into NMT models; and 2) *parameter initialization*, where part/all of the parameters of an NMT model are initialized by a pre-trained model and then training the model on downstream datasets (i.e., parallel corpus).

About knowledge extraction, Yang et al. (2020a) and Zhu et al. (2020) explore enhancing encoder and decoder representations by leveraging pre-trained BERT models (Devlin et al., 2019). In addition, Chen et al. (2020) distill the soft labels from BERT to improve predictions for NMT. These methods are effective but costly because the novel NMT architecture needed to be carefully designed and the computation graph has to store the parameters of both the pre-trained model and NMT model.

About parameter initialization, pre-trained models in different architectures have been studied. For the pre-trained model whose architecture is similar to Transformer encoder (e.g., BERT) or decoder (e.g., GPT (Radford et al., 2018)), the parameters of encoder and decoder can be independently initialized (Conneau and Lample, 2019; Rothe et al., 2020). For the pre-trained model building upon the encoder-decoder architecture (Sutskever et al., 2014), all the model parameters can be directly inherited by NMT, which is easy to use and effective (Song et al., 2019; Lewis et al., 2020; Lin et al., 2020; Yang et al., 2020b).

In general, most previous works focus on designing novel pre-training methods and architectures to boost the model performance of NMT, but the understanding of pre-training for NMT is still limited. This paper improves pre-training for NMT by first understanding its weakness in copying behavior, revealing the importance of further identifying the side-effect from pre-training.

4.2 Copying Behaviors of NMT

It is a common behavior in Seq2Seq models to copy source tokens to the target sentences, especially in monolingual generation tasks. For example, Gu et al. (2016) propose a copying mechanism to explicitly help model learn copying predictions, showing its effectiveness in the tasks of dialogue and summarization.

The copying behaviors also exist in NMT, particularly in languages that share some alphabets (e.g., English and German). Koehn and Knowles (2017) observe that subword-based NMT (Sennrich et al., 2016) outperforms statistical machine translation when translating/copying unknown words.

Knowles and Koehn (2018) find that NMT is able to translate source words in specific contexts via copying (e.g., personal names followed by "Mrs."), and even these are unknown words. However, too many copying signals (i.e., source and target sentences are identical) in training data may lead to one potential threat: NMT models prefer copying source tokens instead of translating them, resulting in performance degradation (Ott et al., 2018a; Khayrallah and Koehn, 2018).

This paper broadens the understanding of copying behaviors in NMT models. We observe that the translation of proper nouns in the source original text contains more copying tokens, which sheds light upon future works.

5 Conclusion and Future Work

We find that NMT models with pre-training are prone to generate more copying tokens. We introduce a copying ratio and a copying error rate to quantitatively analyze copying behaviors in NMT evaluation. In addition, a simple and effective copying penalty is proposed to enhance the copying behaviors during model inference. Experimental results prove the effectiveness of the copying penalty, which can effectively control copying behaviors and improve the overall model performance, especially for the domains (e.g., the IT) where much copying is needed. Extensive analyses reveal that translating proper nouns in source original text generates more copying tokens, providing a direction for the following works on controlling copying behaviors of NMT models.

In the future, we would like to test the effectiveness of the copying penalty on the NMT models with other powerful pre-trained models, and explore more kinds of discrepancies between LM pre-training and NMT training which can be investigated to improve the performance of NMT models. It is also worthwhile to adapt it to other Seq2Seq tasks that need to make a large number of copying predictions, e.g., text summarization and grammar error correction (Liu et al., 2021).

Acknowledgement

This work was supported in part by the Science and Technology Development Fund, Macau SAR (Grant No. 0101/2019/A2), and the Multi-year Research Grant from the University of Macau (Grant No. MYRG2020-00054-FST). We thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling knowledge learned in BERT for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7893–7905, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7057–7067.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Yvette Graham, Barry Haddow, and Philipp Koehn. 2020. Statistical power and translationese in machine translation evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 72–81, Online. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Rebecca Knowles and Philipp Koehn. 2018. Context and copying in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3034–3041, Brussels, Belgium. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2011. Language models for machine translation: Original vs. translated texts. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 363–374, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649–2663, Online. Association for Computational Linguistics.

Xuebo Liu, Longyue Wang, Derek F. Wong, Liang Ding, Lidia S. Chao, and Zhaopeng Tu. 2021. Understanding and improving encoder layer fusion in sequence-to-sequence learning. In *International Conference on Learning Representations*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mathias Müller, Annette Rios, and Rico Sennrich. 2020. Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 151–164, Virtual. Association for Machine Translation in the Americas.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018a. Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3953–3962. PMLR.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018b. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. Citeseer.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Chau Tran, Yuqing Tang, Xian Li, and Jiatao Gu. 2020. Cross-lingual retrieval for iterative self-supervised training. In *Advances in Neural Information Processing Systems*, volume 33, pages 2207–2219. Curran Associates, Inc.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Weinan Zhang, Yong Yu, and Lei Li. 2020a. Towards making the most of bert in neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9378–9385.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020b. CSP:code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Mike Zhang and Antonio Toral. 2019. The effect of translationese in machine translation test sets. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 73–81, Florence, Italy. Association for Computational Linguistics.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Answer Generation for Retrieval-based Question Answering Systems

Chao-Chun Hsu[1][*], **Eric Lind**[2], **Luca Soldaini**[2], **Alessandro Moschitti**[2]
[1]University of Chicago, [2]Amazon Alexa
`chaochunh@uchicago.edu`, `{lssoldai,ericlind,amosch}@amazon.com`

Abstract

Recent advancements in transformer-based models have greatly improved the ability of Question Answering (QA) systems to provide correct answers; in particular, answer sentence selection (AS2) models, core components of retrieval-based systems, have achieved impressive results. While generally effective, these models fail to provide a satisfying answer when all retrieved candidates are of poor quality, even if they contain correct information. In AS2, models are trained to select the best answer sentence among a set of candidates retrieved for a given question. In this work, we propose to generate answers from a set of AS2 top candidates. Rather than selecting the best candidate, we train a sequence to sequence transformer model to generate an answer from a candidate set. Our tests on three English AS2 datasets show improvement up to 32 absolute points in accuracy over the state of the art.

Q: How a **water** **pump** works?

$\mathbf{c_1}$: A small, electrically powered **pump**.

$\mathbf{c_2}$: A large, electrically driven **pump** (electropump) for **waterworks** near the Hengsteysee, Germany.

$\mathbf{c_3}$: A **pump** **is a device** that **moves fluids** (liquids or gases), or sometimes slurries, by **mechanical action**.

$\mathbf{c_4}$: **Pumps** can be classified into three major groups according to the method they use to **move the fluid**: direct lift, displacement, and gravity **pumps**.

$\mathbf{c_5}$: **Pumps** operate by some mechanism (typically reciprocating or rotary), and consume energy to perform **mechanical work** by **moving the fluid**.

G: A **water** **pump** **is a device** that **moves fluids** by **mechanical action**.

Table 1: An example of a question Q and five answer candidates $c_1, \ldots, c_5$ from WikiQA (Yang et al., 2015) ranked by an AS2 system. Answer G generated by our best system is significantly more natural and concise than any extracted candidates.

1 Introduction

Question answering systems are a core component of many commercial applications, ranging from task-based dialog systems to general purpose virtual assistants, e.g., Google Home, Amazon Alexa, and Siri. Among the many approaches for QA, AS2 has attracted significant attention in the last few years (Tymoshenko and Moschitti, 2018; Tian et al., 2020; Garg et al., 2020; Han et al., 2021). Under this framework, for a given question, a retrieval system is first used to obtain and rank a set of supporting passages; then, an AS2 model is used to estimate the likelihood of each sentence extracted from passages to be a correct answer, returning the one with the highest probability. This approach is favored in virtual assistant systems because full sentences are more likely to include the right context and sound natural, both of which are characteristics users value (Berdasco et al., 2019).

AS2 models have shown great performance on academic benchmarks. However, these datasets fail to consider many essential qualities of a QA system which interacts directly with users, such as a virtual assistant. In some cases, extracted answer sentences contain the correct information, but the focus of the answer doesn't match the question; in others, the answer requires reasoning or contextual knowledge from the user or is very long and contains extraneous information. For example, in WikiQA (Yang et al., 2015), a widely used AS2 dataset, the answer *"Wind power is the conversion of wind energy into a useful form of energy, such as using wind turbines to make electrical power, windmills for mechanical power, wind pumps for water pumping... "* is considered a good answer for *"What can be powered by wind?"*, even though its formulation is burdensome to a user.

In this work, we explore a fundamentally differ-

---

[*] This work was completed while the author was an intern at Amazon Alexa.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4276–4282
August 1–6, 2021.

ent approach to AS2. Rather than *selecting* the best candidate, we propose using a model to *generate* a suitable response for a user question. In so doing, we extend the traditional AS2 pipeline with a final generation stage that can recover correct and satisfying answers in cases where a ranking AS2 model fails to place an acceptable candidate at the top position or where a top candidate with the desired information is not a natural-sounding response to the query. Table 1 shows an example of our system: given the question, Q, and a list of candidates, $C_k = \{c_1, \ldots, c_5\}$ sorted by a state-of-the-art AS2 system, we use a sequence-to-sequence model to produce an answer G given Q and C_k as input. This approach, which we refer to as GenQA, addresses the limitations of AS2 systems by composing concise answers which may contain information from multiple sources.

Recent works have shown that large, transformer-based conditional generative models can be used to significantly improve parsing (Chen et al., 2020; Rongali et al., 2020), retrieval (De Cao et al., 2020; Pradeep et al., 2021), and classification tasks (Raffel et al., 2019). Our approach builds on top of this line of work by designing and testing generative models for AS2-based QA systems. In recent years, the use of generative approaches has been evaluated for other QA tasks, such as machine reading (MR) (Izacard and Grave, 2021; Lewis et al., 2020b) and question-based summarization (QS) (Iida et al., 2019; Goodwin et al., 2020; Deng et al., 2020). However, while related, these efforts are fundamentally different from the experimental setting described in this paper. Given a question, generative MR models are used to extract a short span (1-5 tokens) from a passage that could be used to construct an answer to a question. In contrast, AS2 returns a complete sentence that could be directly returned to a user.

QS systems are designed to create a general summary given a question and one or more related documents. Unlike QS, AS2-based QA systems need to provide specific answers; thus, the presence of even a small amount of unrelated information in a response could cause the answer sentence to be unsuitable. In contrast, we show that our approach can succinctly generate the correct information from a set of highly relevant sentence candidates.

In summary, our contribution is four-fold: (*i*) we introduce a new approach for AS2-based QA systems, which generates, rather than selects, an answer sentence; (*ii*) we illustrate how to adapt state-of-the-art models such as T5 (Raffel et al., 2019) and BART (Lewis et al., 2020a) for answer generation; (*iii*) we show[1] that our GenQA system improves over the state-of-the-art AS2-based systems by up to 32 accuracy points, as evaluated by human annotators; finally, (*iv*) we briefly explain why traditional generation metrics are not suited for evaluating AS2-based systems.

2 Datasets

We use four English datasets in our work, one related to generative QA and three to AS2. For a fair comparison between selector and generation methods, we re-evaluate the top answers returned by all models using a fixed set of annotators. All annotations were completed by company associates who are not part of our research group and had no knowledge of the systems. Annotators were required to mark an answer as correct if it was: (*i*) factually correct; (*ii*) natural-sounding; and (*iii*) required no additional information to be understood. All QA pairs were single annotated, as we determined sufficient agreement for this task in previous campaigns.

WikiQA by Yang et al. (2015) contains queries from Bing search logs and candidate answer sentences extracted from a relevant Wikipedia page. For evaluation, we used the dev. and test sets, which contain 126 and 243 unique questions and we re-annotated all of the resulting 569 QA pairs.[2]

Answer Sentence Natural Questions (ASNQ) introduced by Garg et al. (2020) was derived from the NQ dataset (Kwiatkowski et al., 2019) and consists of the questions which have a short answer span within a single sentence in a long answer span. The sentences containing the short answer are marked as correct and the other sentences in the document are marked as incorrect. We use the dev. and test splits introduced by Soldaini and Moschitti (2020) which contain 1,336 questions each. We re-annotated a total of 5,344 QA pairs.

WQA is an internal AS2 dataset created from a non-representative sample of questions asked by

---

[1] Our models, source code, and annotated data are available at: `https://github.com/alexa/wqa-cascade-transformers`.

[2] Due to time and annotation constraints, we were only able to annotate results for 100 queries from each of the dev. and test sets for our UQAT5 model

users of a virtual personal assistant in 2019[3]. For each question, we retrieved 500 pages from an index containing over 100M web documents. We then ranked candidate answers using a state-of-the-art AS2 system, and annotated up to 100 of them. In total, the training and dev. sets contain 3,074 queries and 189k QA pairs, while the test set contains 808 queries. For this effort, we re-annotated 4,847 QA pairs from the test set.

MS MARCO QA NLG (MSNLG) by Nguyen et al. (2016) is a subset of the MS MARCO dataset focused on generating natural language answers to user queries from web search result passages. It consists of 182k queries from Bing search logs, the ten most relevant passages retrieved for each query, and a well-formed answer synthesized by an annotator. This dataset is not designed for AS2, but it represents a large resource of succinct and clear answers, thus making it close to our AS2 task.

3 Generative QA Model (GenQA)

The AS2 task is defined as follows: Let q be an element of the question set, Q, and $C_q = \{c_1, \ldots, c_n\}$ be a set of candidates for q, e.g., sentences retrieved by a search engine, where $c_i \in C$, and C is a set of candidates. We model a selector $\mathcal{S} : Q \times C^n \to C$, such that $\mathcal{S}(q, C_q) = \text{argmax}_i\,(p(q, c_i))$, where $p(q, c_i)$ is the probability that c_i is a good answer. We also define $\mathcal{S}_k : Q \times C^n \to C^k$, such that, $\mathcal{S}_k$ selects the top k answers in descending order of $p(q, c_i)$.

State of the Art Throughout our experiments, we use TANDA (Garg et al., 2020) as our state-of-the-art selector $\mathcal{S}$. This AS2 model was trained as a binary classifier on (q, c_i) pairs using a sequential fine-tuning approach starting with ASNQ and finishing on a target dataset, e.g., WikiQA. Specifically, we use their pretrained RoBERTa Large model (Liu et al., 2019), as it achieved the best results on all datasets it was tested on.

3.1 Our Generative Approach

Instead of selecting the best candidate, we generate a new answer using the information from the top k answer candidates. Thus, our model is a function $\mathcal{G} : Q \times C^k \to G$, where G is the text that can be generated by the generator $\mathcal{G}$ from the question, any fragment of the retrieval set, the model's vocabulary, and knowledge stored in the model's parameters. Formally:

$$\mathcal{G}(q, C_q) = \mathcal{G}(q, C_k) = \mathcal{G}(q, \mathcal{S}_k(k, C_q)). \quad (1)$$

The example in Table 1 shows that we can generate a correct answer from a set of candidates which, as a whole, contain enough information to formulate a correct answer. We propose that a valid answer can be built by composing the *most promising* constituents coming from the different candidates in C_k. Intuitively, information repeated across multiple candidates is more *promising*; therefore, we hypothesize that a model trained on the same or similar generation task should be able to effectively exploit this form of repetition, even in cases where the same information is presented in a similar, but not identical manner. Further, recent works have shown that large transformer models hold a substantial amount of commonsense knowledge in their parameters (Roberts et al., 2020), which our model could leverage to perform inference across sentences in C_k, *e.g.*, associate *water* with *fluid* in the example in Table 1.

3.2 Fine-tuning GenQA

Given a pre-trained transformer seq2seq model, e.g., T5 (Raffel et al., 2019) or BART (Lewis et al., 2020a), we obtain $\mathcal{G}$ by fine-tuning on a large AS2 or QA generation dataset. For this purpose, we format our training data as a standard sequence-to-sequence/text-to-text task, where the source text is the question concatenated with the top five answer candidates, $(q, \mathcal{S}_{k=5})$, joined by newlines. When an answer composed by a human is available, such as in MSNLG, we use it as the output target. For cases where there is no composed answer, we randomly select a known-good candidate to be the target, remove it from the inputs and replace it with another candidate if one is available. We truncate the input text to 512 tokens and, at test time, we use beam search with a beam size of four and a maximum output length of 100 tokens.

4 Experiments

In this section, we first report on our experimental setup, then we show the results on fine-tuning GenQA, and finally, we report on the comparative results between AS2 and GenQA.

4.1 Setup

Models and Parameterization Our GenQA model is based on the T5 (Raffel et al., 2019) vari-

---

[3] The public version of WQA will be released in the short-term future. Please search for a publication with title *WQA: A Dataset for Web-based Question Answering Tasks* on arXiv.

ant of the UnifiedQA (UQAT5) model by Khashabi et al. (2020). We use the Large version of UQAT5, which has 770M parameters for all of our experiments. We compute training loss as the mean of the cross-entropy between the softmax probabilities over the output vocabulary and the one-hot encoded target answer. We fine-tune UQAT5 with a learning rate of $5E^{-5}$. We also experiment with the Large variant of BART (Lewis et al., 2020a), which is comprised of 400M parameters. This model was trained using same loss with learning rate $5E^{-6}$.

Evaluation We used accuracy as our primary metric for all our experiments and models. This is computed as the average number of questions a model answers correctly; for a selector $\mathcal{S}$, it is equivalent to Precision at 1. For $\mathcal{S}$, we also report Hit Rate at 5, which is the fraction of queries with at least one good candidate ranked five or less.

Beside human evaluation, we also experimented with automatic evaluation metrics such as BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004) for GenQA. Such metrics have found little success in evaluating QA tasks (Chaganty et al., 2018; Chen et al., 2019), so we investigate whether that is the case for AS2 as well.

4.2 Results

How to Fine-tune GenQA? As described in Section 4.1, we tested two GenQA variants: one uses a UnifiedQA T5 (UQAT5) (Khashabi et al., 2020) as base model, while the other leverages BART-Large (Lewis et al., 2020a). Of the datasets used in this work, MSNLG and WQA are large enough for fine-tuning GenQA. Therefore, based on preliminary results, we tested four different strategies for training UQAT5: fine tuning on (*i*) WQA or (*ii*) MSNLG alone, (*iii*) combine the two datasets by alternating mini-batches during training, or (*iv*) follow the *transfer-then-adapt* strategy proposed by Garg et al. (2020): first fine-tune on MSNLG, then adapt to a AS2 using WQA.

Table 2 reports the results on the WQA test set, which are all relative to the performance of the state-of-the-art model (TANDA). First, we observe that all GenQA models reported in this table considerably outperform the best selector model, TANDA. This result shows that our generative approach can improve system based on AS2.

Comparing the accuracy of different training strategies applied to UQAT5, we achieve the best results when the model is trained on MSNLG alone (+19.9% over TANDA baseline). While we were initially surprised by this result, as MSNLG is not designed for AS2, error analysis suggests that GenQA benefits from the high quality training data (concise answers written by annotators). Conversely, when training with WQA, we observed that GenQA tends to produce answers that, while correct, are not as natural-sounding. We plan to explore how to best leverage existing AS2 datasets for generative model training in future work. We also note that a GenQA BART-Large achieves comparable results to GenQA UQAT5 on WQA; in preliminary experiments, we found training strategies reported on UQAT5 to have similar effect on BART-Large.

Model	Accuracy	BLEU	ROUGE-L
TANDA (Garg et al., 2020)	*baseline*	-	-
UQAT5 (AS2D)	+5.3%	40.8	55.7
UQAT5 (MSNLG)	**+19.9%**	20.2	39.7
UQAT5 (MSNLG+AS2D)	+13.6%	35.3	50.6
UQAT5 (MSNLG→AS2D)	+7.9%	40.6	54.8
BART-Large (MSNLG)	**+20.7%**	21.5	41.1

Table 2: Relative accuracy of different GenQA models and training configurations on the WQA dataset; both UQAT5 and BART perform best when finetuned on MSNLG only. As shown in previous work, automatic metrics (BLEU, ROUGE-L) do not correlate with human annotations (accuracy).

When manually annotating results of our early tests, we found that BART was more likely to be extractive and copy input passages in their entirety while UQAT5 was more likely to compose new text and produce answers with textual overlap from multiple input candidates but was more likely to hallucinate content. We found that through hyperparameter tuning we could largely eliminate the hallucination from UQAT5 answers but we were unable to make BART more abstractive.

Similar to what has been observed in other QA tasks (Chaganty et al., 2018; Chen et al., 2019), we find that automatic metrics do not correlate with assessments from human annotators. This is due to the fact that neither BLEU nor ROUGE-L are designed to estimate whether an answer is clear and natural-sounding, instead rewarding candidates that have high overlap with reference answers. Most importantly, such overlap is a poor indicator of factual correctness.

	TANDA			GenQA UQAT5	
Dataset	**Acc.**	**Hit@5**	**Length**	**Acc.**	**Length**
WikiQA$_{\text{DEV}}$	59.5	99.2	31.7 ± 13.7	**92.1**	14.9 ± 9.3
WikiQA$_{\text{TEST}}$	61.0	99.2	30.1 ± 12.4	**88.5**	14.6 ± 8.3
ASNQ$_{\text{DEV}}$	75.5	87.7	41.0 ± 122.4	**90.2**	13.9 ± 5.9
ASNQ$_{\text{TEST}}$	69.0	87.9	37.9 ± 51.5	**90.5**	13.9 ± 5.6

Table 3: Accuracy of our GenQA UQAT5 model compared to a state-of-the-art AS2 model by Garg et al. (2020). All answer candidates returned by the two models were re-annotated to ensure a fair comparison. Length is the average number of tokens in the answer.

Comparison between AS2 and GenQA Table 3 reports the results of TANDA and GenQA on two standard AS2 datasets, evaluated with manual annotation. We note that there is an impressive gap of over 20 absolute accuracy points on both development and test sets. This result is produced by two important properties of GenQA. First, it builds correct answers from a pool of correct and incorrect answers, and it can generate a good answer so long as the relevant information can be found anywhere in the top $k = 5$ candidates. This is a clear advantage over using TANDA alone, as Hit-Rate@5 of 99.2%, and 87.9% for WikiQA and ASNQ, respectively, ensures that GenQA often receives at least one correct answer as input.

Second, GenQA exhibits the ability to rewrite *unnatural* answers from a text snippet into an answer suitable for a conversation. For example, for the question *"What year did Isaac Newton die?"*, TANDA returns candidate *"Sir Isaac Newton (25 December 1642–20 March 1727) was an English physicist and mathematician"*. Although correct, no human would provide it in such a form. In contrast, GenQA composes a concise answer: *"Isaac Newton died in 1727"*.

Finally, Table 3 shows that the size of GenQA answers, in terms of words, is only 14 tokens, which is 2.7 times less than the 30-40 tokens from TANDA. This further suggests that GenQA can provide more concise and direct answers, which are preferable in a conversational context.

5 Conclusions

In this work we present GenQA, a generative approach for AS2-based QA systems. The main difference with recent MR-based generative systems is the capacity of our models to generate long answers. This comes from the use of AS2 candidates (complete sentences) as input to our generative approach. In contrast, MR systems, being mainly trained with short answers, e.g., noun phrases and named entities, mostly generate short answers.

We show that GenQA significantly outperforms state-of-the-art selector models for AS2 by up to 32 accuracy points by combining different pieces of information from the top k answer candidates. These results suggest promising directions for generative retrieval-based systems.

Acknowledgments

We thank Thuy Vu for setting up annotation procedures for the WQA dataset.

References

Ana Berdasco, Gustavo López, Ignacio Diaz, Luis Quesada, and Luis A. Guerrero. 2019. User experience comparison of intelligent personal assistants: Alexa, google assistant, siri and cortana. *Proceedings*, 31(1).

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Melbourne, Australia. Association for Computational Linguistics.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5090–5100, Online. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

Yang Deng, Wai Lam, Yuexiang Xie, Daoyuan Chen, Yaliang Li, Min Yang, and Ying Shen. 2020. Joint learning of answer selection and answer summary generation in community question answering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7651–7658. AAAI Press.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2020. TANDA: transfer and adapt pre-trained transformer models for answer sentence selection. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7780–7788. AAAI Press.

Travis Goodwin, Max Savery, and Dina Demner-Fushman. 2020. Towards Zero-Shot Conditional Summarization with Adaptive Multi-Task Fine-Tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3215–3226, Online. Association for Computational Linguistics.

Rujun Han, Luca Soldaini, and Alessandro Moschitti. 2021. Modeling context in answer sentence selection systems on a latency budget. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3005–3010, Online. Association for Computational Linguistics.

Ryu Iida, Canasai Kruengkrai, Ryo Ishida, Kentaro Torisawa, Jong-Hoon Oh, and Julien Kloetzer. 2019. Exploiting background knowledge in compact answer generation for why-questions. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 142–151. AAAI Press.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! A sequence to sequence architecture for task-oriented semantic parsing. In *WWW '20: The Web Conference*

2020, Taipei, Taiwan, April 20-24, 2020, pages 2962–2968. ACM / IW3C2.

Luca Soldaini and Alessandro Moschitti. 2020. The cascade transformer: an application for efficient answer sentence selection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5697–5708, Online. Association for Computational Linguistics.

Zhixing Tian, Yuanzhe Zhang, Xinwei Feng, Wenbin Jiang, Yajuan Lyu, K. Liu, and Jun Zhao. 2020. Capturing sentence relations for answer sentence selection with multi-perspective graph encoding. In *AAAI*.

Kateryna Tymoshenko and Alessandro Moschitti. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2162–2173, Brussels, Belgium. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Grounding *'Grounding'* in NLP

Khyathi Raghavi Chandu, Yonatan Bisk, Alan W Black
Language Technologies Institute
Carnegie Mellon University
`{kchandu, ybisk, awb}@cs.cmu.edu`

Abstract

The NLP community has seen substantial recent interest in grounding to facilitate interaction between language technologies and the world. However, as a community, we use the term broadly to reference *any* linking of text to data or non-textual modality. In contrast, Cognitive Science more formally defines "grounding" as **the process of establishing what mutual information is required for successful communication between two interlocutors** – a definition which might implicitly capture the NLP usage but differs in intent and scope.

We investigate the gap between these definitions and seek answers to the following questions: *(1) What aspects of grounding are missing from NLP tasks?* Here we present the dimensions of coordination, purviews and constraints. *(2) How is the term "grounding" used in the current research?* We study the trends in datasets, domains, and tasks introduced in recent NLP conferences. And finally, *(3) How to advance our current definition to bridge the gap with Cognitive Science?* We present ways to both create new tasks or repurpose existing ones to make advancements towards achieving a more complete sense of grounding.
github.com/khyathiraghavi/Grounding-Grounding

1 Introduction

We as humans communicate and interact for a variety of reasons with a goal. We use language to seek and share information, clarify misunderstandings that conflict with our prior knowledge and contextualize based on the medium of interaction to develop and maintain social relationships. However, language is only one of the enablers of this communication reliant on several auxiliary signals and sources such as documents, media, physical context etc., This linking of concepts to context is *grounding* and within NLP context is often a knowledge base, images or discourse.

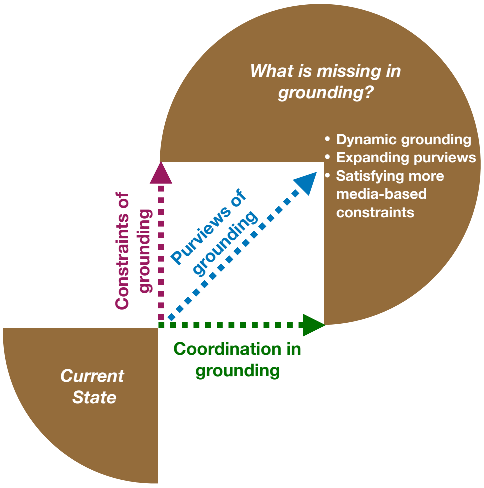

Figure 1: Dimensions of grounding – required to bridge the gap between current state of research and what is missing from a more complete sense of grounding.

In contrast, research in cognitive science defines grounding as the process of building a common ground based on shared mutual information in order to successfully communicate (Clark and Carlson, 1982; Krauss and Fussell, 1990; Clark and Brennan, 1991; Lewis, 2008). We argue that this definition subsumes NLP's current working definition and provides concrete guidance on which phenomena are missing to ensure the naturalness and long term utility of our technologies.

In Section 2, we formalize 3 dimensions key to grounding: Coordination, Purviews and Constraints, to systematize our analysis of limitations in current work. Section 3 presents a comprehensive review of the current progress in the field including the interplay of different domains, modalities, and techniques. This analysis includes understanding when techniques have been specifically designed for a single modality, task, or form of grounding. Finally, Section 4 outlines strategies to repurpose existing datasets and tasks to align with the new

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4283–4305
August 1–6, 2021.

richer definition from cognitive science literature. These introspections, re-formulations, and concrete steps situate NLP ‘grounding’ in larger scientific discourse, to increase its relevance and promise.

2 Dimensions of grounding

Defining grounding loosely as *linking* or tethering concepts is insufficient to achieve a more realistic sense of grounding. Figure 1 presents the research dimensions missing from most current work.

2.1 Dimension 1: Coordination in grounding

The first and the most important dimension that bridges the gap between the two definitions of grounding is the aspect of coordination – alternatively viewed as the difference between *static* and *dynamic* grounding (Fig 2).

Static grounding is the most common type and assumes that the evidence for common ground or the gold truth for grounding is given or attained pseudo-automatically. This is demonstrated in Figure 2 (a). The sequence for this form of interaction includes: (1) human querying the agent, (2) agent querying the data or the knowledge it acquired, (3) agent retrieving and framing a response and (4) agent delivering it to the human. In this setting the common ground is the ground truth KB/data. The human and the agent have common ground by assuming its universality (i.e. no external references). Therefore, successfully grounding the query in this case relies solely on the agent being able to link the query to the data. For instance, in a scenario where a human wants to know the weather report, the accuracy of the database itself is axiomatic and we build a model for the agent to accurately retrieve the queried information in natural language.

Most current research assumes static grounding so progress is measured by the ability of the agent to *link* more concepts to more data. However, the axiomatic common ground often does not exist and needs to be established in real world scenarios.

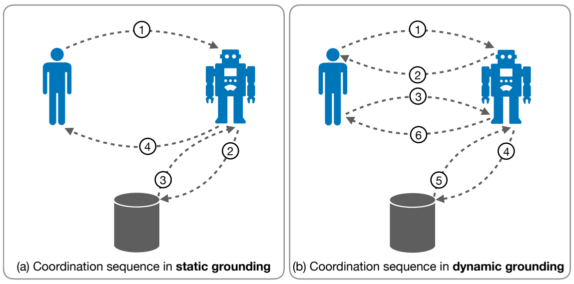

(a) Coordination sequence in **static grounding** (b) Coordination sequence in **dynamic grounding**

Figure 2: Coordination sequence in grounding

Dynamic grounding posits that common ground is built via interactions and clarifications. The mutual information needed to communicate successfully is built via interactions including: Requesting and providing clarifications, Acknowledging or confirming the clarifications, Enacting or demonstrating to receive confirmations, and so forth. This dynamically-established-grounding guides the rest of the interaction by course-correcting any misunderstandings. The sequence of actions in dynamic grounding is demonstrated in Figure 2 (b). The steps for establishing grounding is a part of the interaction that includes: (1) The human querying the agent, (2) The agent requesting clarification or acknowledging, (3) The human clarifying or confirming. These three steps loop until a common ground is established. The remaining steps of (4) querying the data, (5) retrieving or framing a response, and (6) delivering the response, are same as that of static grounding. The agent and the human may not be on the same common ground but steps 2 and 3 loop as the conversation progresses to build this common ground. The process of successfully grounding the query not only relies on the ability of the agent to *link* the query but also to *construct the common ground from the mutually shared information* with respect to the human. Although there are efforts about clarification questioning (), the coverage of phenomena are still far from comprehensive (Benotti and Blackburn, 2021b).

Cognitive sciences in the perspective of language acquisition (Carpenter et al., 1998) present two ways of dynamic grounding via joint attention (Kolева et al., 2015; Tan et al., 2020): Dyadic joint attention and Triadic joint attention. In our case, dyadic attention describes the interaction between the human and the agent and any clarification or confirmation is done strictly between the both of them. Triadic attention also includes a tangible entity along with the human and the agent. The human can provide clarifications by gazing or pointing to this additional piece in the triad.

> *Summary: The community should prioritize dynamic grounding as it is more general and more accurately matches real experiences.*

2.2 Dimension 2: Purviews of grounding

Next, we present the different stages behind reaching a common ground, known as purviews. Most of the current approaches and tasks address these purviews individually and independently, while they are often co-dependent in real world scenarios.

Stage 1: Localization: The first stage is the localization of the concept either in the physical or mental contexts. This step is idiosyncratic and relates to the ability of the agent alone to localize the concept. These concepts often are also linked in a *compositional* form. For instance, consider a scenario in which the agent is to locate a 'blue sweater'. The agent needs to understand each of the concepts of 'blue' and 'sweater' individually and then locate the composition of the whole unit. Clark and Krych (2004) from cognitive sciences demonstrate how *incremental* grounding (Schlangen and Skantze, 2009; DeVault and Traum, 2013; Eshghi et al., 2015) is performed with these compositions and show how recognition and interpretation of fragments help in this by breaking down instructions into simpler ones. This localization occurs at word, phrase and even sentence level in the language modality and pixel, object and scene level in the visual modality.

Stage 2: External Knowledge: After localizing the concept, the next step is to ensure consistency of the current context of the concept with existing knowledge. Often times, the references of grounding either match or contradict the references from our prior knowledge and external knowledge. This might lead to misunderstandings in the consequent rounds of communication. Hence, in addition to localizing the concept, it is also essential to make the concept and its attributes consistent with the available knowledge sources. Most of the current research is focused on localizing with few efforts towards extending it to maintain a consistency of the grounded concept with other knowledge sources.

Stage 3: Common sense: After establishing consistency of the concept, a human-like interaction additionally calls for grounding the common sense associated with the concept in that scenario. In addition to the basic level of practical knowledge that concerns with day to day scenarios Sap et al. (2020), the concept should also be reasoned based on that particular context. This contextual common sense moves the idiosyncratic sense towards a sense of collective understanding. For instance, if the human feels cold and asks the agent to get a blue coat, the agent needs to understand that the coat in this instance is a sweater coat and not a formal coat. This implicit common sense minimizes the effort in building a common ground reducing articulation of meticulous details. Therefore it is essential to incorporate this explicitly in our modeling as well.

Stage 4: Personalized consensus: As a part of the evolving conversations, the references in the language evolve as well. The grounded term might have different meanings for the agent in the context with access to the history as opposed to a fresh agent without access to the history. This multi-instance multi-turn process to achieve consensus makes this collective or a shared stage continually adapting to personalization leading to better engagement (Bohus and Horvitz, 2014). In such settings, it is sufficient that the human and the agent are in consensus with the truth value of the grounded term, which need not be the same as the ground truth. This shift in the truth value of the meanings of the grounded terms often arise due to developing short-cuts for ease of communication and personalization, which is an acceptable shift as long as the communication is successful.

> *Summary: Common ground requires expanding to verticals of local, general, common-sense and personalized contextual knowledge.*

2.3 Dimension 3: Constraints of grounding

The medium and mode of communication constrain communicative goals in practical scenarios. The number and availability of such media have increased and facilitated ubiquitous communication around the world, presenting a diversity in the mode of interaction. Motivated by this, we resurface and adapt the constraints of grounding with respect to media of interaction as defined by Clark and Brennan (1991). Here are the definitions of these constraints in the context of grounded language processing and the corresponding categorization of the majority of the representative domains in grounding satisfying different constraints.

- *Copresence:* Agent and human share the same physical environment of the data. Most of the current research in the category of embodied agents satisfy this constraint.
- *Visibility:* The data is visible to the agent and/or human. The domains of images, images & speech, videos, embodied agents satisfy this constraint.
- *Audibility:* Agent and human communicate by speaking about the data. Domains like speech, spoken image captions and videos satisfy this.
- *Cotemporality:* The agent/human receives at roughly the same time as the human/agent pro-

duces. The lag in the domains like conversations or interactive embodied agents is considered negligible and satisfy this constraint.

- *Simultaneity:* The agent and the human can send and receive at once simultaneously. Most media are cotemporal but do not engage in simultaneous interaction. This often disrupts the understanding of the current utterance and the participant may have to repeat it to avoid misunderstandings, which is commonly observed in real world scenarios.
- *Sequentiality:* The turn order of the agent and the human cannot get out of sequence. Face-to-face conversations usually follow this constraint but an email thread with active participants and the comments sections in online portals (such as Youtube, Twitch etc.,) do not necessarily follow a sequence. In such cases a reply to the message may be separated by arbitrary number of irrelevant messages. These categories are usually understudied but are commonly observed online.
- *Reviewability:* The agent reviews the common ground to the human to adapt to imperfect human memories. For instance, we reiterate full references instead of adapting to short cut references when the conversation resurfaces after a while. This is to develop a personalized adaptation between the interlocutors based on the media to enable ease of communication.
- *Revisability:* The interaction between the agent and the human indexes to a specific utterance in the conversation sequence and revise it, therefore changing the course of the interaction henceforth. Human errors are only natural in a conversation and the agent needs to be ready to rectify the previously grounded understanding.

There has been a good and continual effort in formulating tasks and datasets that satisfy the constraints of visibility, audibility and cotemporality. Contemporary efforts also see an increased interest in addressing copresence in grounded contexts. Very recently, (Benotti and Blackburn, 2021a) highlights the importance of recovering from mistakes while establishing the collabrative nature of grounding, contributing to the ability of revisability.

> *Summary: Key to progress is to focus on largely a blind spot in grounding: simultaneity, sequentiality & revisability to revive from mistakes.*

3 Grounding '*Grounding*'

Having covered a more formal definition of grounding adapted to NLP, we turn our attention to cataloging the precise usage of 'grounding' in our research community. We present an analysis on the various domains and techniques NLP has explored.

3.1 Data and Annotations

To this end, since our aim is to investigate how the community understands the loosely defined term 'grounding', we subselected all the papers that mention terms for 'grounding' in the title or abstract from the S2ORC data (Lo et al., 2020) between the years 1980-2020. In this way, we grounded the term *'grounding'* in literature [1] to collect the relevant papers. We acknowledge that the papers analyzed here are not exhaustive with respect to concept of 'grounding'.

Each of the paper is annotated with answers to the following questions: (i) is it introducing a new task? (ii) is it introducing a new dataset? (iii) what is the world scope (iv) is it working on multiple languages? (v) what are the grounding domains? (vi) what is the grounding task? (vii) what is the grounding technique?

3.2 Domains of grounding

Real world contexts we interact with are diverse and can be derived from different modalities such as textual or non-textual, each of which comprises of domains. Our categorization of these is inspired from the constraints of grounding as described in §2.3. Based on this, the modality based categorization include the following domains:

- *Textual modality comprising plain text, entities & events, knowledge bases and knowledge graphs.*
- *Non-textual modality comprising images, speech, images & speech and videos.*

Numerous other domains including *numbers and equations, colors, programs, tables, brain activity signals* etc., are studied in the context of grounding at relatively lower scale in comparison to the aforementioned ones. Each of these can further be interacted with along the variation in the coordination dimension of grounding from §2.1, that give rise to the following settings including *conversations, embodied agents and face-to-face interactions*.

3.3 Approaches to grounding

This section presents a list of approaches tailored to grounding. The obvious solution is to expand the datasets to promote a research platform. The

---

[1] Please note that this is not an exhaustive list of papers working on grounding as there are several others that do mention this term and still work on some form of grounding.

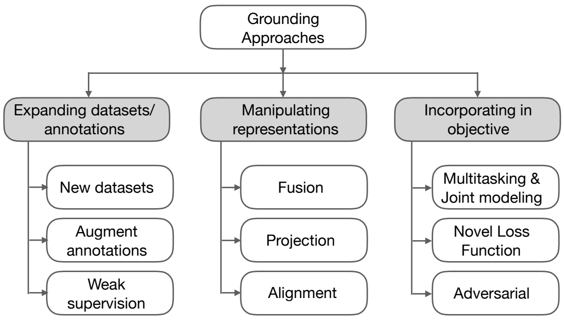

Figure 3: Categorical approaches to grounding

second is to manipulate different representations to *link* and bring them together. Finally the learning objective can leverage grounding. The subcategories within each are presented in Figure 3.

1. Expanding datasets / annotations: The first step towards building an ecosystem for research in grounding is to curate the necessary datasets which is accomplished with expensive human efforts, augmenting existing annotations and automatically deriving annotations with weak supervision.

1a) New datasets: There has been an increase in efforts for curating new datasets with task specific annotations. These are briefly overlaid in Table 1 along with their modalities, domains and tasks.

1b) Augment annotations: These curated datasets can also be used subsequently to augment with task specific annotations instead of collecting the data from scratch, which might be more expensive.

- *Non-textual Modality:* Static grounding here includes using adversarial references to ground visual referring expressions (Akula et al., 2020), narration (Chandu et al., 2019b, 2020a), language learning (Suglia et al., 2020; Jin et al., 2020) etc.,
- *Textual Modality:* Static grounding includes entity slot filling (Bisk et al., 2016).
- *Interactive:* Though not fully dynamic grounding, some efforts here are amongst tasks like understanding spatial expressions (Udagawa et al., 2020), collaborative drawing (Kim et al., 2019) etc.,

1c) Weak supervision: While the above two are based on human efforts, we can also perform weak supervision to use a model trained to derive automatic soft annotations required for the task.

- *Non-Textual Modality:* In the visual modality, weak supervision is used in the contexts of automatic object proposals for different tasks like spoken image captioning (Srinivasan et al., 2020), visual semantic role labeling (Silberer and Pinkal, 2018), phrase grounding (Chen et al., 2019), loose temporal alignments between utterances and a set of events (Koncel-Kedziorski et al., 2014) etc.,
- *Textual Modality:* In the contexts of text, Tsai and Roth (2016a) work towards disambiguating concept mentions appearing in documents and grounding them in multiple KBs which is a step towards Stage 3 in §2.2. Poon (2013) perform question answering with a single database and (Parikh et al., 2015) with symbols.

Modality	Domain	Task	Work
Non-textual	Images	caption relevance	(Suhr et al., 2019)
		multimodal MT	(Zhou et al., 2018c)
		sports commentaries	(Koncel-Kedziorski et al., 2014)
		semantic role labeling	(Silberer and Pinkal, 2018)
		instruction following	(Han and Schlangen, 2017)
		navigation	(Andreas and Klein, 2014)
		causality	(Gao et al., 2016)
		spatial expressions	(Kelleher et al., 2006)
		spoken image captioning	(Alishahi et al., 2017)
		entailment	(Vu et al., 2018)
		image search	(Kiros et al., 2018)
		scene generation	(Chang et al., 2015)
	Videos	action segmentation	(Regneri et al., 2013)
		semantic parsing	(Ross et al., 2018)
		instruction following	(Liu et al., 2016)
		question answering	(Lei et al., 2020)
Textual	Text	content transfer	(Prabhumoye et al., 2019)
		commonsense inference	(Zellers et al., 2018)
		reference resolution	(Kennington and Schlangen, 2015)
		symbol grounding	(Kameko et al., 2015)
		bilingual lexicon extraction	(Laws et al., 2010)
		POS tagging	(Cardenas et al., 2019)
Interactive	Text	negotiations	(Cadilhac et al., 2013)
		documents	(Zhou et al., 2018b)
		improvisation	(Cho and May, 2020)
	Visual	referring expressions	(Haber et al., 2019) (Takmaz et al., 2020)
		emotions and styles	(Shuster et al., 2020)
		media interviews	(Majumder et al., 2020)
		spatial reasoning	(Jänner et al., 2018)
		navigation	(Ku et al., 2020)
	Other	problem solving	(Li and Boyer, 2015)

Table 1: Example datasets introduced for *grounding*.

> *Summary: While augmentation and weak supervision can be leveraged for dimensions of coordination and purviews, curating new datasets is the need of the hour to explore various constraints.*

2. Manipulating representations: Grounding concepts often involves multiple modalities or representations that are *linked*. Three major methods to approach this are detailed here.

2a) Fusion and concatenation: Fusion is a very common technique in scenarios involving multiple modalities. In scenarios with a single modality, representations are often concatenated.

- *Non-textual modality:* Fusion is applied with images for tasks like referring expressions (Roy et al., 2019), SRL (Yang et al., 2016) etc., For videos, some tasks are grounding action descriptions (Regneri et al., 2013), spatio-temporal QA (Lei et al., 2020), concept similarity (Kiela and Clark, 2015), mapping events (Fleischman and Roy, 2008) etc.,

- *Textual Modality:* With text, this is similar to concatenating context (Prabhumoye et al. (2019) perform content transfer by augmenting context).
- *Interactive:* In a conversational setting, work is explored in reference resolution (Takmaz et al., 2020; Haber et al., 2019), generating engaging response (Shuster et al., 2020), document grounded response generation Zhou et al. (2018b), etc.,
- *Others:* Nakano et al. (2003) study face-to-face grounding in instruction giving for agents.

2b) Alignment: An alternative to combining representations is aligning them with one another.

- *Non-textual modality:* Wang et al. (2020) perform phrase localization in images and Hessel et al. (2020) study temporal alignment in videos.
- *Interactive:* Han and Schlangen (2017) align GUI actions to sub-utterances in conversations and Jänner et al. (2018) align local neighborhoods to the corresponding verbalizations.

2c) Projecting into a common space: A widely used approach is to also bring the different representations on to a joint common space.

- *Non-textual modality:* Projection to a joint semantic space is used in spoken image captioning (Chrupala et al., 2017; Alishahi et al., 2017; Havard et al., 2019), bicoding for learning image attributes (Silberer and Lapata, 2014), representation learning of images (Zarrieß and Schlangen, 2017) and speech (Vijayakumar et al., 2017).
- *Textual modality:* Tsai and Roth (2016b) demonstrate cross-lingual NER and mention grounding model by activating corresponding language features.Yang et al. (2019) perform imputation of embeddings for rare and unseen words by projecting a graph to the pre-trained embeddings space.

> *Summary: Modeling different representations effectively aid in improving both consistency across purviews and media based constraints.*

3. Learning Objective: Grounding is often performed to support a more defined end purpose task. We identified 3 ways that are broadly adopted to incorporate grounding in objective functions.

3a) Multitasking and Joint Modeling: The *linking formulation* of grounding is often used as an auxiliary or dependent to model another task.

- *Non-textual Modality:* Multitasking with images is used to perform spoken image captioning (Chrupala, 2019) and grammar induction (Zhao and Titov, 2020). Joint modeling was used in multiresolution language grounding Koncel-Kedziorski et al. (2014), identifying referring expressions Roy et al. (2019), multimodal MT (Zhou et al., 2018c), video parsing Ross et al. (2018), learning latent semantic annotations (Qin et al., 2018) etc.,
- *Interactive:* In a conversational setting, multitasking is used to compute concept similarity judgements (Silberer and Lapata, 2014), knowledge grounded response generation (Majumder et al., 2020), grounding language instructions Hu et al. (2019). Joint modeling is used by Li and Boyer (2015) to address dialog for complex problem solving in computer programs.

3b) Loss Function: It is crucial to utilize appropriate loss designed for the specific grounding task. The main difference between multitasking and a loss function adaptation is that while multitasking reweights combinations of existing loss functions, novel loss functions are informed by the data/task at hand, adapting to a novel use case.

- *Non-textual Modality:* Grujicic et al. (2020) design soft organ distance loss to model inter and intra organ interactions for relative grounding. Ilharco et al. (2019) improve diversity in spoken captions with a masked margin softmax loss.

3c) Adversarial: Leveraging deceptive grounded inputs in an attempt to fool the model is capable of making it robust to certain errors.

- *Non-textual Modality:* Chen et al. (2018); Akula et al. (2020) present an algorithm to craft visually-similar adversarial examples.
- *Textual Modality:* Zellers et al. (2018) perform adversarial filtering and constructs a de-biased dataset by iteratively training stylistic classifiers.

> *Summary: Manipulating learning objective is a modeling capability aiding as an additional component in bringing grounding adjunct to several other end tasks across all the dimensions.*

3.4 Analysis of trends

Based on the categories of approaches and different datasets from §3.3, we presented a representative set of analyses that highlight the major avenues that addressing the key missing pieces of work on grounding to advance future research.

Figure 4 presents the trends in the development of grounding over the past decade including: specific approaches (a,b) that presents new tasks/challenges; world scopes (Bisk et al., 2020) (c) contributing to grounding language in different

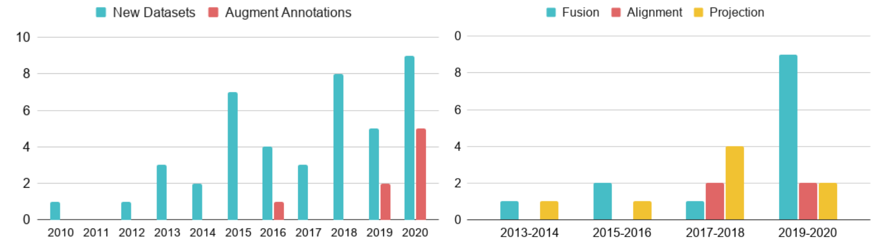

a) Trends in curating new datasets and augmenting annotations

(b) Trends in manipulation of representations

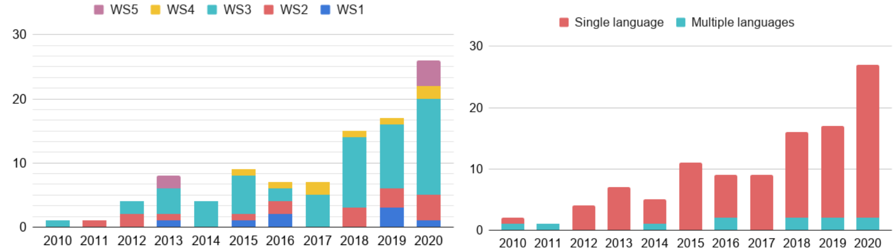

(c) Trends in world scopes

(c) Trends in multilingual datasets and tasks

Figure 4: Analysis on the trends in grounding

data types; and multilinguality (d) contributing to a part of linguistic diversity. We also present hierarchical pie charts in Figure 5 and in Appendix to analyze the compositions of modalities and domains for these approaches.While we believe our analysis targets several of the most critical dimensions paving way for future research directions, it is not exhaustive and welcome suggestions from the community for additional analysis. For example, it is also interesting to study domain diversity, task formulation/usefulness, etc., in future.

Trends in datasets expansion: The introduction of new datasets has seen a rapid increase over the years, while there is also a subtle increasing trend in augmenting annotations to the existing datasets, as observed in Figure 4 (a). As we can see from Figure 5 (a), across all the domains, gathering new datasets seem to be prominent than augmenting them with additional annotations to repurpose the data for a new task. There seems to be a higher emphasis of expansion of datasets in the non-textual modalities, particularly in the domain of images. A similar rise is not observed in interactive settings including conversational data and interaction with embodied agents; which is the propitious way to bridge the gap towards real sense of grounding. It is indeed encouraging to see an increasing trend in the efforts for expanding datasets but *the need of the hour is to redirect some of these resources to address dynamic grounding in the coordination dimension which is scarcely studied in existing datatsets.*

Trends in manipulating representations: From Figure 4 (b), we note that the fusion technique has and is increasingly becoming popular in grounding through manipulating representations in comparison to alignment and projection. This is also observed in Figure 5 (b) with the dominance of non-textual modality. In the context of textual modality, this technique is equivalent to concatenation of the context or history in a conversation. Projecting onto a common space is the next popular technique in comparison to alignment. Similarly, we observe that the non-textual modality overwhelmingly occupies the space of manipulating representations with exceeding prominence of fusion. *Fusion and projecting onto common space currently are exceedingly used methodologies to ground within a single purview. They demonstrate a promising direction to manipulate representations across different stages to maintain consistency along the purviews.*

Trends in World Scopes: We also study the development of the field based on the definitions of the world scopes presented by Bisk et al. (2020). Based on this, last decade has seen an increasing dominance in research on world scope 3 (world of sights and sounds). However, this is limited to this scope and the same trend is not clear in world scope 4 (world of embodiment and action). An encouraging observation is the focus of the field in world scope 5 (social world) which is closer to real interactions in the last year. *We need to accelerate development of datasets and tasks in world scopes 4 and 5. It is highly recommended to take dynamic grounding scenario into account in the efforts for*

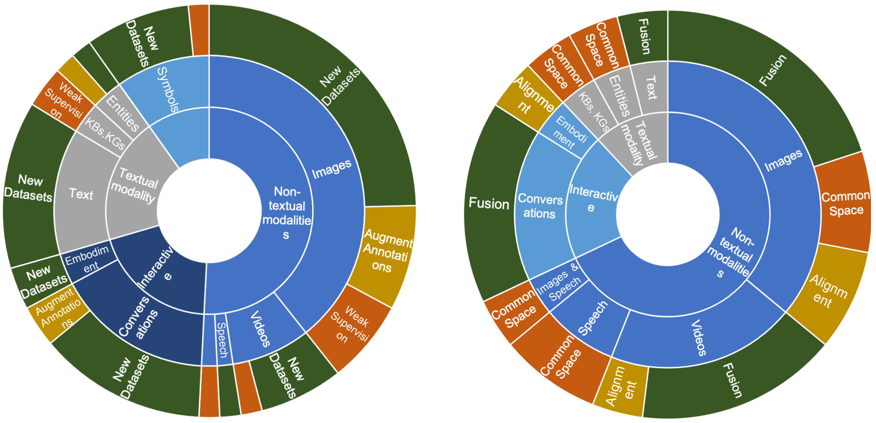

(a) Expanding datasets/annotations

(b) Manipulating Representations

Figure 5: Analysis of Domains and Techniques

curating datasets in these scopes.

Inclusivity of multiple languages: Figure 4 (c) shows that research into grounding in multiple languages is still incredibly rare. As noted by Bender (2011), improvements in one language do not necessarily mandate comparable performances in other languages. *The norm for benchmarking large scale tasks still remains anglo-centric and we need serious efforts to drift this trend to identify challenges in grounding across languages. As a first step, a relatively less expensive way to navigate this dearth is to augment the annotations of existing datasets with other languages.*

4 Path Ahead: Towards New Tasks and Repurposing Existing Datasets

We presented the dimensions of grounding that require serious attention to bridge the gap between the definitions in cognitive sciences and language processing communities in §2. Based on this, we analyzed the language processing research to understand where we stand and where we fall short with the ongoing efforts in trends in grounding in §3. While we strongly advocate for efforts in building new datasets and tasks considering progress along these dimensions, we believe in a smoother transition towards this goal. Hence we present strategies to repurpose existing resources to maximum utility as we stride towards achieving grounding in real sense. In this section, we focus on concrete suggestions to improve along each of the dimensions.

Coordination: This is based on simulating interaction for dynamic grounding. As establishing a common ground is not integrated within datasets, we propose an iterative paradigm to explicitly settle on a common ground based on our priors.

The first family of methods to perform this is human-in-the-loop interactions. The traditional methods of data collection do not cater to human feedback or generation. Some recent approaches to incorporate human feedback are during data collection (Wallace et al., 2019), training (Stiennon et al., 2020), inference (Hancock et al., 2019). While the feedback in a human in the loop setting can be via scores, we argue for natural language feedback (Wallace et al., 2019) loop, which resembles human-human grounding via communication.

The second family of methods are inspired from the theory of mind (Gopnik and Wellman, 1992) to iteratively or progressively ask and clarify to establish a common ground (Roman et al., 2020). de Vries et al. (2017); Suglia et al. (2020) disambiguate or clarify the referenced object through a series of questions in a guessing game. This iterative paradigm can be related to work by Shwartz et al. (2020) that generates clarification questions and answers to incorporate in the task of question answering. This loop of semi-automatic generation of clarifications establishes a common ground. This is also in spirit similar to generating an explanation or a hypothesis for question answering (Latcinnik and Berant, 2020). The process of generating an acceptable explanation to human before acts as establishing a common ground.

We believe that datasets and tasks along the following 3 directions encourage dynamic grounding: (1) conversational language learning (Chevalier-Boisvert et al., 2019) or acquisition, and (2) clarification questioning and ambiguity resolution (Shwartz et al., 2020) (3) mixed initiative for grounding in conversations (Morbini et al., 2012). The need of the hour that can revolutionize this paradigm is the *development of evaluation strategies to monitor evolution of the common ground.* This dynamic grounding data helps improve performance/robustness and encourages human's trust while using these interactive systems.

Purviews: This is based on establishing consistency across stages of grounding with an incremental paradigm. A simple solution is a modular approach where the purviews flow into the next stage after reasonably satisfying the previous stage. The current benchmarking approaches are mostly lateral i.e., our current strategies collate multiple datasets of a single task to benchmark. This approach implicitly establishes boundaries between the purviews. In contrast, *we advocate for a longitudinal approach for benchmarking* i.e in addition to collating different datasets for a task, we also extend the purviews of the task such that the output from the previous purview flows into the next purview. An example of establishing a longitudinal benchmark for visual dialog. The tasks flow from object detection (stage 1: localization) to knowledge graphs (stage 2: external knowledge) to common sense understanding (stage 3: common sense) to empathetic dialogue (stage 4: personalization) for the *same* dataset. This helps us dissect which aspect of grounding is the model good and bad at to understand the weak areas.

Constraints: With media imposed constraints, there is a need for paradigm shift in the way these datasets are curated. The optimal way to navigate this problem is curating new datasets to *specifically focus on the less studied constraints of simultaneity, sequentiality and revisability.* At the heart of revisability in a collaborative dialog is clarification questioning and resolving ambiguities (Boni and Manandhar, 2003; Rao and III, 2018; Braslavski et al., 2017; Kumar and Black, 2020; Aliannejadi et al., 2020; Benotti and Blackburn, 2021b) However, they are rarely explored and are not systematically standardized across modalities. Transferring knowledge for shared constraints across tasks is a promising way to leverage the existing datasets.

Augment with multilingual annotations: Different languages also bring novel challenges to each of these issues (e.g. pronoun drop dialogue in Japanese, morphological alignments, etc). However, as observed in §3.4, the increase in expanding datasets is not proportionally reflected to include multiple languages. We recommend a relatively less expensive process of translating the datasets for grounding into other languages to kick start this inclusion. The research community has already seen such efforts in image captioning with human annotated German captions in Multi30k (Elliott et al., 2016) extended from Flick30k (Plummer et al., 2015) and Japanese captions in STAIR (Yoshikawa et al., 2017) based on MS-COCO images (Lin et al., 2014). Instead of using human annotations, some efforts have also been made to use automatic translations such as the work by Thapliyal and Soricut (2020) and denoising (Chandu et al., 2020b) extending from (Sharma et al., 2018). Not just augmentation, but there are also ongoing efforts in gathering datasets in multiple languages (Ku et al., 2020) extending (Anderson et al., 2018).

5 Conclusions

We discussed the missing pieces and dimensions that bridge the gap between the definitions of grounding in Cognitive Sciences and NLP communities. Thereby, we chart out executable actions in steering existing resources along 3 dimensions to achieve a more realistic sense of grounding. Specifically: (1) Static grounding still remains the central tenet for existing tasks and datasets. However, dynamic grounding is key moving forward. (2) Current benchmarking strategies evaluate model generalization. In tandem, we also need to steer towards longitudinal benchmarking to naturally proliferate across purviews of grounding that is closer to human interactions. (3) Constraints imposed by the interaction medium present nuanced categories of communicative goals. While discerning learning from shared constraints, we also urge the community to invest resources on revisability as a way to recover from contextually mistaken groundings. While ruminating on the above phenomena, the challenge of expanding them to multiple languages and domains still persists. We also recommend systematic evaluation of grounding along these dimensions in addition to the existing linking capabilities.

Ethical Considerations

The analytical and ontological discussion here focuses exclusively on the question of *grounding* and common ground and does not address the harmful biases inherent in these datasets. Further, the common ground for which we are advocating is culturally specific and future work that introduces tasks and data for these purposes must be explicit about who they serve (culturally and linguistically).

References

Arjun R. Akula, Spandana Gella, Yaser Al-Onaizan, Song-Chun Zhu, and Siva Reddy. 2020. Words aren't enough, their order matters: On the robustness of grounding visual referring expressions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6555–6565. Association for Computational Linguistics.

Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail S. Burtsev. 2020. Convai3: Generating clarifying questions for open-domain dialogue systems (clariq). *CoRR*, abs/2009.11352.

Afra Alishahi, Marie Barking, and Grzegorz Chrupala. 2017. Encoding of phonology in a recurrent neural model of grounded speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pages 368–378. Association for Computational Linguistics.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3674–3683. IEEE Computer Society.

Jacob Andreas and Dan Klein. 2014. Grounding language with points and paths in continuous spaces. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 58–67. ACL.

Leonor Becerra-Bonache, Henning Christiansen, and M Dolores Jiménez-López. 2018. A gold standard to measure relative linguistic complexity with a grounded language learning model. In *Proceedings of the Workshop on Linguistic Complexity and Natural Language Processing*, pages 1–9.

Emily M Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*, 6(3):1–26.

Luciana Benotti and Patrick Blackburn. 2021a. Grounding as a collaborative process. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 515–531. Association for Computational Linguistics.

Luciana Benotti and Patrick Blackburn. 2021b. A recipe for annotating grounded clarifications. *CoRR*, abs/2104.08964.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph P. Turian. 2020. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8718–8735. Association for Computational Linguistics.

Yonatan Bisk, Siva Reddy, John Blitzer, Julia Hockenmaier, and Mark Steedman. 2016. Evaluating induced CCG parsers on grounded semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2022–2027. The Association for Computational Linguistics.

Dan Bohus and Eric Horvitz. 2014. Managing human-robot engagement with forecasts and... *um*... hesitations. In *Proceedings of the 16th International Conference on Multimodal Interaction, ICMI 2014, Istanbul, Turkey, November 12-16, 2014*, pages 2–9. ACM.

Marco De Boni and Suresh Manandhar. 2003. An analysis of clarification dialogue for question answering. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Benjamin Börschinger, Bevan K. Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1416–1425. ACL.

Pavel Braslavski, Denis Savenkov, Eugene Agichtein, and Alina Dubatovka. 2017. What do you mean exactly?: Analyzing clarification questions in CQA. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval, CHIIR 2017, Oslo, Norway, March 7-11, 2017*, pages 345–348. ACM.

Anaïs Cadilhac, Nicholas Asher, Farah Benamara, and Alex Lascarides. 2013. Grounding strategic conversation: Using negotiation dialogues to predict trades in a win-lose game. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 357–368. ACL.

Ronald Cardenas, Ying Lin, Heng Ji, and Jonathan May. 2019. A grounded unsupervised universal part-of-speech tagger for low-resource languages. In *Proceedings of the 2019 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 2428–2439. Association for Computational Linguistics.

Malinda Carpenter, Katherine Nagell, Michael Tomasello, George Butterworth, and Chris Moore. 1998. Social cognition, joint attention, and communicative competence from 9 to 15 months of age. *Monographs of the society for research in child development*, pages i–174.

Khyathi Chandu, Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W Black. 2019a. "my way of telling a story": Persona based grounded story generation. In *Proceedings of the Second Workshop on Storytelling*, pages 11–21.

Khyathi Raghavi Chandu and Alan W. Black. 2020. Style variation as a vantage point for code-switching. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 4761–4765. ISCA.

Khyathi Raghavi Chandu, Ruo-Ping Dong, and Alan W. Black. 2020a. Reading between the lines: Exploring infilling in visual narratives. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1220–1229. Association for Computational Linguistics.

Khyathi Raghavi Chandu, Eric Nyberg, and Alan W. Black. 2019b. Storyboarding of recipes: Grounded contextual generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6040–6046. Association for Computational Linguistics.

Khyathi Raghavi Chandu, Piyush Sharma, Soravit Changpinyo, Ashish Thapliyal, and Radu Soricut. 2020b. Weakly supervised content selection for improved image captioning. *arXiv preprint arXiv:2009.05175*.

Angel X. Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning. 2015. Text to 3d scene generation with rich lexical grounding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 53–62. The Association for Computer Linguistics.

David L. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 430–439. The Association for Computer Linguistics.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2587–2597. Association for Computational Linguistics.

Zhenfang Chen, Lin Ma, Wenhan Luo, and Kwan-Yee Kenneth Wong. 2019. Weakly-supervised spatio-temporally grounding natural sentence in video. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1884–1894. Association for Computational Linguistics.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Hyundong Cho and Jonathan May. 2020. Grounding conversations with improvised dialogues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2398–2413. Association for Computational Linguistics.

Grzegorz Chrupala. 2019. Symbolic inductive bias for visually grounded learning of spoken language. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6452–6462. Association for Computational Linguistics.

Grzegorz Chrupala, Lieke Gelderloos, and Afra Alishahi. 2017. Representations of language in a model of visually grounded speech signal. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 613–622. Association for Computational Linguistics.

Chenhui Chu, Mayu Otani, and Yuta Nakashima. 2018. iparaphrasing: Extracting visually grounded paraphrases via an image. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3479–3492. Association for Computational Linguistics.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley, editors, *Perspectives on socially shared cognition*, pages 127–149. American Psychological Association.

Herbert H Clark and Thomas B Carlson. 1982. Hearers and speech acts. *Language*, pages 332–373.

Herbert H Clark and Meredyth A Krych. 2004. Speaking while monitoring addressees for understanding. *Journal of memory and language*, 50(1):62–81.

David DeVault and David R. Traum. 2013. A method for the approximation of incremental understanding of explicit utterance meaning using predictive models in finite domains. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 1092–1099. The Association for Computational Linguistics.

Ruo-Ping Dong, Khyathi Raghavi Chandu, and Alan W. Black. 2019. Induction and reference of entities in a visual story. *CoRR*, abs/1909.09699.

Gabriel Doyle and Michael C. Frank. 2015. Shared common ground influences information density in microblog texts. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1587–1596. The Association for Computational Linguistics.

Judith Eckle-Kohler. 2016. Verbs taking clausal and non-finite arguments as signals of modality - revisiting the issue of meaning grounded in syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, VL@ACL 2016, August 12, Berlin, Germany*. The Association for Computer Linguistics.

Arash Eshghi, Christine Howes, Eleni Gregoromichelaki, Julian Hough, and Matthew Purver. 2015. Feedback in conversation as incremental semantic update. In *Proceedings of the 11th International Conference on Computational Semantics, IWCS 2015, 15-17 April, 2015, Queen Mary University of London, London, UK*, pages 261–271. The Association for Computer Linguistics.

Zhihao Fan, Zhongyu Wei, Siyuan Wang, and Xuanjing Huang. 2019. Bridging by word: Image grounded vocabulary construction for visual captioning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6514–6524. Association for Computational Linguistics.

Michael Fleischman and Deb Roy. 2008. Grounded language modeling for automatic speech recognition of sports video. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 121–129. The Association for Computer Linguistics.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 457–468. The Association for Computational Linguistics.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Yue Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Alison Gopnik and Henry M Wellman. 1992. Why the child's theory of mind really is a theory.

Dusan Grujicic, Gorjan Radevski, Tinne Tuytelaars, and Matthew B. Blaschko. 2020. Learning to ground medical text in a 3d human atlas. In *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*, pages 302–312. Association for Computational Linguistics.

Janosch Haber, Tim Baumgärtner, Ece Takmaz, Lieke Gelderloos, Elia Bruni, and Raquel Fernández. 2019. The photobook dataset: Building common ground through visually-grounded dialogue. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1895–1910. Association for Computational Linguistics.

Ting Han and David Schlangen. 2017. Grounding language by continuous observation of instruction following. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 491–496. Association for Computational Linguistics.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazaré, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume*

1: Long Papers, pages 3667–3684. Association for Computational Linguistics.

William Havard, Laurent Besacier, and Jean-Pierre Chevrot. 2020. Catplayinginthesnow: Impact of prior segmentation on a model of visually grounded speech. In *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*, pages 291–301. Association for Computational Linguistics.

William N. Havard, Jean-Pierre Chevrot, and Laurent Besacier. 2019. Word recognition, competition, and activation in a model of visually grounded speech. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*, pages 339–348. Association for Computational Linguistics.

Jack Hessel, Zhenhai Zhu, Bo Pang, and Radu Soricut. 2020. Beyond instructional videos: Probing for more diverse visual-textual grounding on youtube. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8812–8822. Association for Computational Linguistics.

Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. 2019. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6551–6557. Association for Computational Linguistics.

Pingping Huang, Jianhui Huang, Yuqing Guo, Min Qiao, and Yong Zhu. 2019. Multi-grained attention with object-level grounding for visual question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3595–3600. Association for Computational Linguistics.

Gabriel Ilharco, Yuan Zhang, and Jason Baldridge. 2019. Large-scale representation learning from visually grounded untranscribed speech. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*, pages 55–65. Association for Computational Linguistics.

Michaela Jänner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Trans. Assoc. Comput. Linguistics*, 6:49–61.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 683–693. The Association for Computational Linguistics.

Xisen Jin, Junyi Du, Arka Sadhu, Ram Nevatia, and Xiang Ren. 2020. Visually grounded continual learning of compositional phrases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2018–2029. Association for Computational Linguistics.

Mark Johnson, Katherine Demuth, and Michael C. Frank. 2012. Exploiting social information in grounded language learning via grammatical reduction. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 883–891. The Association for Computer Linguistics.

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Can symbol grounding improve low-level nlp? word segmentation as a case study. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2298–2303. The Association for Computational Linguistics.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2019. Learning to discover, ground and use words with segmental neural language models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6429–6441. Association for Computational Linguistics.

John D. Kelleher, Geert-Jan M. Kruijff, and Fintan J. Costello. 2006. Proximity in context: An empirically grounded computational model of proximity for processing topological spatial expressions. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics.

Casey Kennington and David Schlangen. 2015. Simple learning and compositional application of perceptually grounded word meanings for incremental reference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 292–301. The Association for Computer Linguistics.

Douwe Kiela, Luana Bulat, and Stephen Clark. 2015. Grounding semantics in olfactory perception. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language*

Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers, pages 231–236. The Association for Computer Linguistics.

Douwe Kiela and Stephen Clark. 2015. Multi- and cross-modal semantics beyond vision: Grounding in auditory perception. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2461–2470. The Association for Computational Linguistics.

Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. 2019. Codraw: Collaborative drawing as a testbed for grounded goal-driven communication. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6495–6513. Association for Computational Linguistics.

Jamie Ryan Kiros, William Chan, and Geoffrey E. Hinton. 2018. Illustrative language understanding: Large-scale visual grounding with image search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 922–933. Association for Computational Linguistics.

Nikolina Koleva, Martin Villalba, Maria Staudte, and Alexander Koller. 2015. The impact of listener gaze on predicting reference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 812–817. The Association for Computer Linguistics.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. 2014. Multi-resolution language grounding with weak supervision. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 386–396. ACL.

Robert M Krauss and Susan R Fussell. 1990. Mutual knowledge and communicative effectiveness. *Intellectual teamwork: Social and technological foundations of cooperative work*, pages 111–146.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4392–4412. Association for Computational Linguistics.

Vaibhav Kumar and Alan W. Black. 2020. Clarq: A large-scale and diverse dataset for clarification question generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7296–7301. Association for Computational Linguistics.

Veronica Latcinnik and Jonathan Berant. 2020. Explaining question answering models through text generation. *CoRR*, abs/2004.05569.

Florian Laws, Lukas Michelbacher, Beate Dorow, Christian Scheible, Ulrich Heid, and Hinrich Schütze. 2010. A linguistically grounded graph model for bilingual lexicon extraction. In *COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume, 23-27 August 2010, Beijing, China*, pages 614–622. Chinese Information Processing Society of China.

Jie Lei, Licheng Yu, Tamara L. Berg, and Mohit Bansal. 2020. TVQA+: spatio-temporal grounding for video question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8211–8225. Association for Computational Linguistics.

David Lewis. 2008. *Convention: A philosophical study*. John Wiley & Sons.

Xiaolong Li and Kristy Boyer. 2015. Semantic grounding in dialogue for complex problem solving. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 841–850. The Association for Computational Linguistics.

Zekang Li, Cheng Niu, Fandong Meng, Yang Feng, Qian Li, and Jie Zhou. 2019. Incremental transformer with deliberation decoder for document grounded conversations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 12–21. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer.

Changsong Liu, Lanbo She, Rui Fang, and Joyce Y. Chai. 2014. Probabilistic labeling for efficient referential grounding based on collaborative discourse. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*

2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers, pages 13–18. The Association for Computer Linguistics.

Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-Chun Zhu, and Joyce Yue Chai. 2016. Jointly learning grounded task structures from language instruction and visual demonstration. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1482–1492. The Association for Computational Linguistics.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Minh-Thang Luong, Michael C. Frank, and Mark Johnson. 2013. Parsing entire discourses as very long strings: Capturing topic continuity in grounded language learning. *Trans. Assoc. Comput. Linguistics*, 1:315–326.

Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian J. McAuley. 2020. Interview: Large-scale modeling of media dialog with discourse patterns and knowledge grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8129–8141. Association for Computational Linguistics.

Alexandre Blondin Massé, Guillaume Chicoisne, Yassine Gargouri, Stevan Harnad, Odile Marcotte, and Olivier Picard. 2008. How is meaning grounded in dictionary definitions? In *Coling 2008: Proceedings of the 3rd Textgraphs workshop on Graph-based Algorithms for Natural Language Processing*, pages 17–24.

Brian McMahan and Matthew Stone. 2015. A bayesian model of grounded color semantics. *Trans. Assoc. Comput. Linguistics*, 3:103–115.

Will Monroe, Robert X. D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Trans. Assoc. Comput. Linguistics*, 5:325–338.

Fabrizio Morbini, Eric Forbell, David DeVault, Kenji Sagae, David R. Traum, and Albert A. Rizzo. 2012. A mixed-initiative conversational dialogue system for healthcare. In *Proceedings of the SIGDIAL 2012 Conference, The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 5-6 July 2012, Seoul National University, Seoul, South Korea*, pages 137–139. The Association for Computer Linguistics.

Yukiko I. Nakano, Gabe Reinstein, Tom Stocky, and Justine Cassell. 2003. Towards a model of face-to-face grounding. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan*, pages 553–561. ACL.

Sushobhan Nayak and Amitabha Mukerjee. 2012. Grounded language acquisition: A minimal commitment approach. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 2059–2076. Indian Institute of Technology Bombay.

Joel Nothman, Matthew Honnibal, Ben Hachey, and James R. Curran. 2012. Event linking: Grounding event reference in a news archive. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 228–232. The Association for Computer Linguistics.

Tim Oates. 2003. Grounding word meanings in sensor data: Dealing with referential uncertainty. In *Proceedings of the HLT-NAACL 2003 workshop on Learning word meaning from non-linguistic data*, pages 62–69.

Brian E. Pangburn, S. Sitharama Iyengar, Robert C. Mathews, and Jonathan P. Ayo. 2003. EBLA: A perceptually grounded model of language acquisition. In *Proceedings of the HLT-NAACL 2003 Workshop on Learning Word Meaning from Non-Linguistic Data*, pages 46–53.

Nikolaos Pappas, Phoebe Mulcaire, and Noah A. Smith. 2020. Grounded compositional outputs for adaptive language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1252–1267. Association for Computational Linguistics.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 756–766. The Association for Computational Linguistics.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2641–2649. IEEE Computer Society.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,* *ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 933–943. The Association for Computer Linguistics.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2622–2632. Association for Computational Linguistics.

Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang, and Chin-Yew Lin. 2018. Learning latent semantic annotations for grounding natural language to structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3761–3771. Association for Computational Linguistics.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2737–2746. Association for Computational Linguistics.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Trans. Assoc. Comput. Linguistics*, 1:25–36.

Homero Roman Roman, Yonatan Bisk, Jesse Thomason, Asli Celikyilmaz, and Jianfeng Gao. 2020. Rmm: A recursive mental model for dialog navigation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1732–1745.

Candace Ross, Andrei Barbu, Yevgeni Berzak, Battushig Myanganbayar, and Boris Katz. 2018. Grounding language acquisition by training semantic parsers using captioned videos. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2647–2656. Association for Computational Linguistics.

Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis. 2003. Conversational robots: building blocks for grounding word meaning. In *Proceedings of the HLT-NAACL 2003 workshop on Learning word meaning from non-linguistic data*, pages 70–77.

Subhro Roy, Michael Noseworthy, Rohan Paul, Daehyung Park, and Nicholas Roy. 2019. Leveraging past references for robust language grounding. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*, pages 430–440. Association for Computational Linguistics.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing : Mapping sentences to grounded equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1088–1097. The Association for Computational Linguistics.

Maarten Sap, Vered Shwartz, Antoine Bosselut, Yejin Choi, and Dan Roth. 2020. Commonsense reasoning for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, ACL 2020, Online, July 5, 2020*, pages 27–33. Association for Computational Linguistics.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *EACL 2009, 12th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, Athens, Greece, March 30 - April 3, 2009*, pages 710–718. The Association for Computer Linguistics.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2556–2565. Association for Computational Linguistics.

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1842–1861. Association for Computational Linguistics.

Robik Shrestha, Kushal Kafle, and Christopher Kanan. 2020. A negative case analysis of visual grounding methods for VQA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8172–8181. Association for Computational Linguistics.

Kurt Shuster, Samuel Humeau, Antoine Bordes, and Jason Weston. 2020. Image-chat: Engaging grounded conversations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2414–2429. Association for Computational Linguistics.

Ekaterina Shutova, Niket Tandon, and Gerard de Melo. 2015. Perceptually grounded selectional preferences. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of*

Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 950–960. The Association for Computer Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4615–4629. Association for Computational Linguistics.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 721–732. The Association for Computer Linguistics.

Carina Silberer and Manfred Pinkal. 2018. Grounding semantic roles in images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2616–2626. Association for Computational Linguistics.

Georgios P. Spithourakis, Isabelle Augenstein, and Sebastian Riedel. 2016. Numerically grounded language models for semantic error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 987–992. The Association for Computational Linguistics.

Tejas Srinivasan, Ramon Sanabria, Florian Metze, and Desmond Elliott. 2020. Fine-grained grounding for multimodal speech recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2667–2677. Association for Computational Linguistics.

Luc Steels. 2004. Constructivist development of grounded construction grammar. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pages 9–16. ACL.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize from human feedback. *CoRR*, abs/2009.01325.

Michael Strube and Udo Hahn. 1999. Functional centering - grounding referential coherence in information structure. *Comput. Linguistics*, 25(3):309–344.

Alessandro Suglia, Ioannis Konstas, Andrea Vanzo, Emanuele Bastianelli, Desmond Elliott, Stella Frank, and Oliver Lemon. 2020. Compguesswhat?!: A multi-task evaluation framework for grounded language learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7625–7641. Association for Computational Linguistics.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6418–6428. Association for Computational Linguistics.

Ece Takmaz, Mario Giulianelli, Sandro Pezzelle, Arabella Sinclair, and Raquel Fernández. 2020. Refer, reuse, reduce: Grounding subsequent references in visual and conversational contexts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4350–4368.

Xiang Zhi Tan, Sean Andrist, Dan Bohus, and Eric Horvitz. 2020. Now, over here: Leveraging extended attentional capabilities in human-robot interaction. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, HRI 2020, Cambridge, UK, March 23-26, 2020*, pages 468–470. ACM.

Ashish V. Thapliyal and Radu Soricut. 2020. Cross-modal language generation using pivot stabilization for web-scale language coverage. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 160–170. Association for Computational Linguistics.

Chen-Tse Tsai and Dan Roth. 2016a. Concept grounding to multiple knowledge bases via indirect supervision. *Trans. Assoc. Comput. Linguistics*, 4:141–154.

Chen-Tse Tsai and Dan Roth. 2016b. Illinois cross-lingual wikifier: Grounding entities in many languages to the english wikipedia. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, December 11-16, 2016, Osaka, Japan*, pages 146–150. ACL.

Takuma Udagawa, Takato Yamazaki, and Akiko Aizawa. 2020. A linguistic analysis of visually grounded dialogues based on spatial expressions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 750–765. Association for Computational Linguistics.

Ashwin K. Vijayakumar, Ramakrishna Vedantam, and Devi Parikh. 2017. Sound-word2vec: Learning word representations grounded in sounds. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 920–925. Association for Computational Linguistics.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4466–4475. IEEE Computer Society.

Hoa Trong Vu, Claudio Greco, Aliia Erofeeva, Somayeh Jafaritazehjan, Guido Linders, Marc Tanti, Alberto Testoni, Raffaella Bernardi, and Albert Gatt. 2018. Grounded textual entailment. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2354–2368. Association for Computational Linguistics.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. *Transactions of the Association for Computational Linguistics*, 7:387–401.

Qinxin Wang, Hao Tan, Sheng Shen, Michael W. Mahoney, and Zhewei Yao. 2020. MAF: multimodal alignment framework for weakly-supervised phrase grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2030–2038. Association for Computational Linguistics.

Jun Xu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Conversational graph grounded policy learning for open-domain conversation generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1835–1845. Association for Computational Linguistics.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded semantic role labeling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 149–159. The Association for Computational Linguistics.

Tsung-Yen Yang, Andrew S. Lan, and Karthik Narasimhan. 2020. Robust and interpretable grounding of spatial references with relation networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 1908–1923. Association for Computational Linguistics.

Ziyi Yang, Chenguang Zhu, Vin Sachidananda, and Eric Darve. 2019. Embedding imputation with grounded language information. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3356–3361. Association for Computational Linguistics.

Yuya Yoshikawa, Yutaro Shigeto, and Akikazu Takeuchi. 2017. STAIR captions: Constructing a large-scale japanese image caption dataset. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 417–421. Association for Computational Linguistics.

Sina Zarrieß and David Schlangen. 2017. Deriving continous grounded meaning representations from referentially structured multimodal contexts. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 959–965. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 93–104. Association for Computational Linguistics.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2031–2043. Association for Computational Linguistics.

Yanpeng Zhao and Ivan Titov. 2020. Visually grounded compound pcfgs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4369–4379. Association for Computational Linguistics.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6869–6882. Association for Computational Linguistics.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018a. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2065–2076. Association for Computational Linguistics.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W. Black. 2018b. A dataset for document grounded conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 708–713. Association for Computational Linguistics.

Mingyang Zhou, Runxiang Cheng, Yong Jae Lee, and Zhou Yu. 2018c. A visual attention grounding neural model for multimodal machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3643–3653. Association for Computational Linguistics.

A Examples for dimensions of grounding

Static Grounding: In static grounding, when you ask an agent *"Can you place the dragon fruit on the rack"?*, the agent links the entities and places the dragon fruit on the rack. The challenge here is mainly the linking part which is crucial to ensure it accurately understood the instruction.

Dynamic Grounding: The same is not true for dynamic grounding. There are primarily 2 ways to materialize this. First, with respect to language learning: What if the agent does not know dragon fruit? The agent needs to first ask *"What is a dragon fruit?"*, and the human provides an answer. Lets say the human responded by describing the physical attributes such as *reddish pink fruit* and/or a spatial reference by refering to it as *the fruit on the bottom left*. The important aspect here is that the agent asks and learns what a dragon fruit is and use this knowledge later.

The second is ambiguity resolution. Consider a scenario where there are multiple racks. It is very natural for a human to ask the agent which rack to resolve ambiguity.We expect the same from the agent to ask a clarifying question to resolve ambiguity and then place it on the second rack.

Purviews - Localization: Consider this example of a conversation between an agent and a human.

***Human**: What is the name of the role Robert Downey Jr played in Avengers?*
***Agent**: He played the role of Tony Stark, and sometimes is also referred to as Iron Man.*

The agent begins by localizing and linking Robert Downey Jr to Tony Stark and Iron Man to provide the appropriate answer to the query.

Purviews - External Knowledge: However, natural conversations also extend beyond the purview of localization to discuss a broadened scope involving external knowledge of the context including entities, actions etc., For example, consider this conversation which seems to be a natural continuation to the earlier one.

***Human**: Is he the head of SHIELD?*
***Agent**: Tony Stark has never been the head of SHIELD in the movies but has been the acting head upon Maria Hill's suggestion in the Comics.*

Once we localized Tony Stark, asking additional information like whether he is the head of SHIELD is natural in conversations; However, access to required external knowledge is rarely present in the datasets as well as evaluated. Here, we need to refer to external sources spanning from movies to comics to conclude that he has been the acting head in the comics but was never in the movies.

Purviews - Common sense: One of the branches of natural progression to this context can extend to the following turns:

***Human**: How long was the contract between Tony Stark and Marvel?*
***Agent**: Tony Stark is the name of the character in Marvel. Would you like to know the contract length for Robert Downey Jr who played the role?*

Here, the agent needs to understand that Tony Stark is not a real person, but is a character in Marvel. Hence, any contract is with the actor but not the character who played the role. The agent needs to have the common sense to understand this and clarify the question.

Purviews - Personalization: Upon a continous exchange regarding this topic (and perhaps a few other times earlier), the agent needs to adapt and personalize to the interacting human over time.

***Human**: Can you give me any movie suggestions?*
***Agent**: Yes, since you like Disney movies and seem interested in Robert Downey Jr, would you like to watch "Dolittle"?*

Having discussed about Robert Downey Jr in prior contexts and retaining from the prior interactions that the human likes Disney movies, when the human asks about a movie recommendation, the agent continually learns and contextually suggests Robert Downey Jr's Disney movie "Dolittle" as a recommendation.

Constraints - Copresence: Modality is an important medium that affects communicative goals and the nature of interaction. Here is an example in a copresent environment.

***Human**: I want to play with my cat. Can you get me the ball on your right?*

In the above example, the human and the agent

	Modality			Cue				
	Copresence	**Visibility**	**Audibility**	**Cotemporality**	**Simultaneity**	**Sequentiality**	**Reviewability**	**Revisabiility**
Face-to-face	✓	✓	✓	✓	✓	✓		
Telephone			✓	✓	✓	✓		
Video Teleconference		✓	✓	✓	✓	✓		
Terminal Teleconference				✓		✓	✓	
Answering Machines			✓				✓	
E-mail							✓	✓
Letters							✓	✓

Table 2: Constraints of grounding along with their medium of communication (Clark and Brennan, 1991)

are copresent in the same environment. The above utterance for instance, includes executable actions in the environment along with references being either person-centric or agent-centric.

Constraints - Visibility: Certain communications like in the cases of visual question answering or visual dialog only presents a visible medium to interact *about*. The interaction requires information from an image or a video, but does not necessarily include executable actions or cater to external knowledge of the information. For example, with an access to an image a human can ask a question like the following:

***Human**: How many peaks are there in those mountain ranges?*

Constraints - Audibility: This modality constrains the information scope to be within speech signals that are only heard and do not contain any visual or copresent information.

Table 2 presents the constrainst of grounding.

B Further survey and categories

Here is a brief elaboration of the datasets presented in Table 1.

New datasets: The first solution to curate the entire dataset with annotations designed for the task.

- *Non-textual Modality:* For images, new datasets are curated for a variety of tasks including caption relevance (Suhr et al., 2019), multimodal MT (Zhou et al., 2018c), soccer commentaries (Koncel-Kedziorski et al., 2014) semantic role labeling (Silberer and Pinkal, 2018), instruction following (Han and Schlangen, 2017), navigation (Andreas and Klein, 2014), understanding physical causality of actions (Gao et al., 2016), understanding topological spatial expressions (Kelleher et al., 2006), spoken image captioning (Alishahi et al., 2017), entailment (Vu et al., 2018), image search (Kiros et al., 2018), scene generation (Chang et al., 2015), etc., Coming to videos, datasets have become popular for several tasks like identifying action segments (Regneri et al., 2013), sematic parsing (Ross et al., 2018), instruction following from visual demonstration (Liu et al., 2016), spatio-temporal question answering (Lei et al., 2020), etc.,

- *Textual Modality:* Within text, there are several datasets for tasks like content transfer (Prabhumoye et al., 2019), commonsense inference (Zellers et al., 2018), reference resolution (Kennington and Schlangen, 2015), symbol grounding (Kameko et al., 2015), studying linguistic and non-linguistic contexts in microblogs (Doyle and Frank, 2015), bilingual lexicon extraction (Laws et al., 2010), universal part-of-speech tagging for low resource languages (Cardenas et al., 2019), entity linking and reference (Nothman et al., 2012) etc.,

- *Other:* More static grounding datasets correspond to tasks like identifying phrases representing variables (Roy et al., 2016), conceptual similarity in olfactory data (Kiela et al., 2015), identifying colors from descriptions (Monroe et al., 2017), correcting numbers (Spithourakis et al., 2016) etc.,

- *Interactive:* Coming to an interactive setting, the datasets span tasks like conversations based on negotiations (Cadilhac et al., 2013), referring expressions from images (Haber et al., 2019; Takmaz et al., 2020), emotions and styles (Shuster et al., 2020), media interviews (Majumder et al., 2020), documents (Zhou et al., 2018b), improvisation (Cho and May, 2020), problem solving (Li and Boyer, 2015), spatial reasoning in a simulated environment (Jänner et al., 2018), navigation (Ku et al., 2020) etc.,

In addition, there are several other techniques used to ground phenomenon in real world contexts.

In addition to the techniques dicscussed in the paper, we also studied the categorization based on stratification, which is explained here.

Stratification: The stratification technique characterizes the input or the model to explicitly cater to the compositionality property. This can be done by either breaking down the input to meaningful compositions or building the model to compose the representations. Utilizing grammatical rules need not necessarily lead to compositions, although there is an overlap between these two techniques.

A common strategy when language is involved is leveraging **syntax and parsing**. In the domain of images, Udagawa et al. (2020) design an annotation protocol to capture important linguistic structures based on predicate-argument structure, modification and ellipsis to utilize linguistic structures based on spatial expressions. Becerra-Bonache et al. (2018) study linguistic complexity from a developmental point of view by using syntactic rules to provide data to a learner, that identifies the underlying language from this data. Shi et al. (2019) use image-caption pairs to extract constituents from text, based on the assumption that similar spans should be matched to similar visual objects and these concrete spans form constituents. Kelleher et al. (2006) use combinatory categorial grammar (CCG) to build a psycholinguistic based model to predict absolute proximity ratings to identify spatial proximity between objects in a natural scene. Ross et al. (2018) employ CCG-based parsing to a fixed set of unary and binary derivation rules to generate semantic parses for videos.

- *Textual Modality:* Johnson et al. (2012) study the modeling the task of inferring the referred objects using social cues and grammatical reduction strategies in language acquisition. Eckle-Kohler (2016) attempt to understand meaning in syntax by a multi-perspective semantic characterization of the inferred classes in multiple lexicons. Chen (2012) develop a context-free grammar to understand formal navigation instructions that correspond better with words or phrases in natural language. Börschinger et al. (2011) study the probabilistic context-free grammar learning task using the inside-out algorithm in game commentaries. CCG parsers are also used to perform entity slot filling task (Bisk et al., 2016). When applied to question answering over a database, dependency rules are used to model the edge states as well as transitions such as the work done by using a treeHMM (Poon, 2013).

- *Other:* Roy et al. (2016) perform equation parsing that identifies noun phrases in a given sentence representing variables using high precision mathematical lexicon to generate the correct relations in the equations. Parikh et al. (2015) perform prototype driven learning to learn a semantic parser in tables of nested events and unannotated text.

- *Interactive:* Luong et al. (2013) use parsing and grammar induction to produce a parser capable of representing full discourses and dialogs. Steels (2004) study games and embodied agents by modeling a constructivist approach based on invention, abduction and induction to language development.

Another frequently used technique when language is involved is by leveraging the principle of **compositionality**. This implies that the meaning of a complex expression is determined by the meanings of its constituents and how they interact with one another.

- *Non-textual Modality:* In the domain of images, Suhr et al. (2019) present a new dataset to understand challenges in language grounding including compositionality, semantic diversity and visual reasoning. Shi et al. (2019), discussed earlier also use grammar rules to compose the inputs. Koncel-Kedziorski et al. (2014) leverage the compositional nature of language to understand professional soccer commentaries. In the domain of videos, Nayak and Mukerjee (2012) study language acquisition by segmenting the world to obtain a meaning space and combining them to get a linguistic pattern.

- *Textual Modality:* With ontologies, Pappas et al. (2020) perform adaptive language modeling to other domains to get a fully compositional output embedding layer which is further grounded in information from a structured lexicon.

- *Interactive:* Roy et al. (2003) work on grounding word meanings for robots by composing perceptual, procedural, and affordance representations.

Hierarchical modeling is also applied to show effect of introducing phone, syllable, or word boundaries in spoken captions (Havard et al., 2020) and with a compact bilinear pooling in visual question answering (Fukui et al., 2016).

There is some work that presents a bayesian probabilistic formulation to learn referential grounding in dialog (Liu et al., 2014), user preferences (Cadilhac et al., 2013), color descriptions (McMahan and Stone, 2015; Andreas and Klein, 2014).

A huge chunk of work also focus on leveraging attention mechanism for grounding multimodal phenomenon in images (Srinivasan et al., 2020; Chu et al., 2018; Huang et al., 2019; Fan et al., 2019; Vu et al., 2018; Kawakami et al., 2019; Dong et al., 2019), videos (Lei et al., 2020; Chen et al., 2019) and navigation of embodied agents (Yang et al., 2020), etc.,

Some approach this using data structures such as graphs in the domains of grounding images (Chang et al., 2015; Liu et al., 2014), videos (Liu et al., 2016), text (Laws et al., 2010; Chen, 2012; Massé et al., 2008), entities (Zhou et al., 2018a), knowledge graphs and ontologies (Jauhar et al., 2015; Zhang et al., 2020) and interactive settings Jauhar et al. (2015); Xu et al. (2020).

Here is the technique wise representation of these categories of models in the literature.

Figure 6: Papers addressing stratification in grounding

C Prevelance of modailties and constraints

Here is the distribution of the papers studying various tasks based on the constraints imposed by the medium.

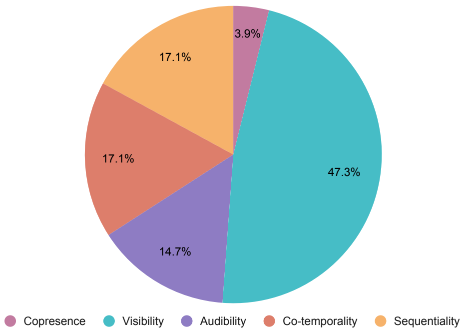

Figure 7: Papers addressing different constraints of grounding

As we can see, a major concentration of these efforts lie in grounding visual and textual media, while a few cater to audibility i.e speech signals. Papers studying dialog are the main representatives of the constraints for sequentiality and co-temporality.

D Nuanced modeling variations for grounding

Here is a more nuanced and finer grained categorization of the various modeling techniques used in literature for grounding. Figure 8 presents these categories in depth.

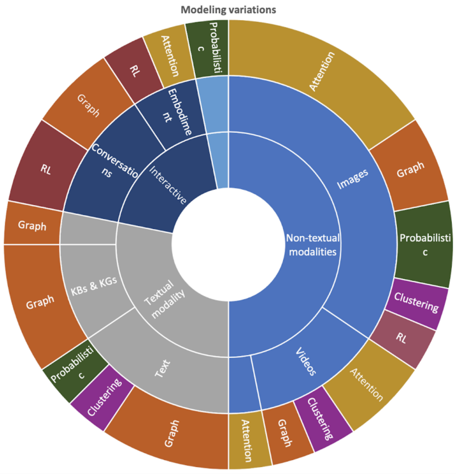

Figure 8: Modeling variations in papers studying grounding

As discussed in the paper, most of the literature is focused on grounding in static visual modality. Attention based methods dominate the rest of the methods in both textual and non-textual modalities closely followed by graph based methods as observed in these trends.

This is not an exhaustive study of all the techniques that present grounding, but are some of the representative categories. Here are more studies that perform grounding with various techniques such as clustering (Shutova et al., 2015; Cardenas et al., 2019) regularization (Shrestha et al., 2020), CRFs (Gao et al., 2016), classification (Pangburn et al., 2003; Monroe et al., 2017), linguistic theories (Strube and Hahn, 1999), iterative refinement (Li et al., 2019; Chandu and Black, 2020), language modeling (Spithourakis et al., 2016; Cho and May, 2020), nearest neighbors (Kiela et al., 2015), contextual fusion (Chandu et al., 2019a), mutual information (Oates, 2003), cycle consistency (Zhong et al., 2020) etc.,

Federated Chinese Word Segmentation with Global Character Associations

Yuanhe Tian♥∗, **Guimin Chen**♢∗, **Han Qin**♠∗, **Yan Song**♠♡†
♥University of Washington ♢QTrade
♠The Chinese University of Hong Kong (Shenzhen)
♡Shenzhen Research Institute of Big Data
♥`yhtian@uw.edu` ♢`chenguimin@foxmail.com`
♠`hanqin@link.cuhk.edu.cn` ♠`songyan@cuhk.edu.cn`

Abstract

Chinese word segmentation (CWS) is a fundamental task for Chinese information processing, which always suffers from out-of-vocabulary word issues, especially when it is tested on data from different sources. Although one possible solution is to use more training data, in real applications, these data are stored at different locations and thus are invisible and isolated among each other owing to the privacy or legal issues (e.g., clinical reports from different hospitals). To address this issue and benefit from extra data, we propose a neural model for CWS with federated learning (FL) adopted to help CWS deal with data isolation, where a mechanism of global character associations is proposed to enhance FL to learn from different data sources. Experimental results on a simulated environment with five nodes confirm the effectiveness of our approach, where our approach outperforms different baselines including some well-designed FL frameworks.[1]

1 Introduction

Chinese word segmentation (CWS) is a preliminary and vital task for natural language processing (NLP). This task aims to segment Chinese character sequence into words and thus is generally performed as a sequence labeling task (Tseng et al., 2005; Levow, 2006; Song et al., 2009a; Sun and Xu, 2011; Song and Xia, 2012, 2013; Mansur et al., 2013). Although recent neural-based CWS systems (Pei et al., 2014; Chen et al., 2017; Ma et al., 2018; Higashiyama et al., 2019; Qiu et al., 2019; Ke et al., 2020; Huang et al., 2020a; Tian et al., 2020e) have achieved very good performance on benchmark datasets, it is still an unsolved task (Fu et al., 2020), because it is challenging to handle out-of-vocabulary words (OOV), especially in real applications where the test data may come from different sources. Although leveraging extra labeled data from other sources or domains could alleviate this issue, in real applications, such data are always located in different nodes and thus are inaccessible to each other because of the privacy or legal concerns (e.g., clinical or financial reports from different hospitals or companies).

To address the data isolation issue, federated learning (FL) (Shokri and Shmatikov, 2015; Konečný et al., 2016) is proposed and has shown its great promises for many machine learning tasks (Aono et al., 2017; Sheller et al., 2018; He et al., 2020). In many cases, data in different nodes are encrypted and aggregated to the centralized model, and they are invisible to each other during the training stage. This property allows FL to be an essential technique for real applications with privacy and security requirements. However, conventional FL techniques are more suitable for nodes sharing homogeneous data, which is seldom the case for NLP tasks. Particularly for CWS, the appropriate segmentation is sensitive to the data source, where the text and vocabularies used in different datasets contain various expressing patterns. For example, in real applications such as Input Method Editors (IME, such as pinyin input environment), there are millions of individual users with their data stored in isolated nodes, where the different nodes could have diverse segmentation requirement due to the users' preference. Therefore, the restricted data access of traditional FL approaches could result in inferior performance for CWS since they cannot update the model to facilitate localized prediction. Unfortunately, limited attentions have been paid to address this issue. Most existing approaches (Liu et al., 2019; Huang et al., 2020b; Sui et al., 2020) with FL on NLP (e.g., for language modeling

---

∗Equal contribution.

†Corresponding author.

[1] The code and models involved in this paper are released at `https://github.com/cuhksz-nlp/GCASeg`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4306–4313
August 1–6, 2021.

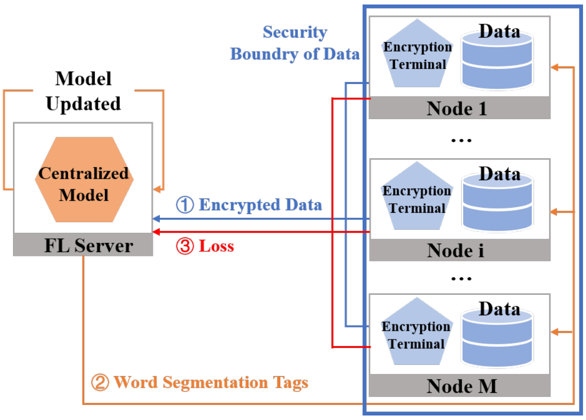

Figure 1: The server-node architecture of our approach. The encrypted information (i.e., encrypted data, word segmentation tags, and loss) communicates between a node and the server, where the locally stored data is inaccessible to other nodes during the training process.

(Hard et al., 2018; Chen et al., 2019), named entity recognition (Ge et al., 2020), and text classification (Zhu et al., 2020)) mainly focus on optimizing the learning process and ignore domain diversities.

In this paper, we propose a FL-based neural model (GCA-FL) for CWS, which is enhanced by global character association (GCA) mechanism in a distributed environment. The GCA mechanism is designed to capture contextual information (patterns) in a particular input for localized predictions and to handle the difficulties in identifying text sources caused by data inaccessibility. Specifically, GCA is served as a server-side component to associate global character n-grams with different inputs from each node and responds with contextual information to help the backbone segmenter. Experimental results on a simulated environment with isolated data from five domains demonstrate the effectiveness of our approach, where GCA-FL outperforms different baselines including the ones with well designed FL framework.

2 The approach

Figure 1 illustrates the overall server-node architecture for applying our approach. The centralized model is stored in the FL server and data from multiple sources (domains) are stored at different nodes (the i-th node is denoted by $\mathcal{N}_i$), respectively. Encrypted information (e.g., data, vectors, and loss) communicates between each node $\mathcal{N}_i$ and the FL server. In this way, the original data stay in the local node and is not accessible to the other nodes. To encode contextual information (patterns) to facilitate localized prediction, we enhance FL by introducing GCA into the centralized model (Figure 2), which follows the character-based sequence labeling paradigm for CWS. Herein, GCA encodes the contextual information from the encrypted input and uses the resulted information to guide the centralized model to make a localized prediction. In the following, we introduce FL for CWS and then the centralized model with GCA.

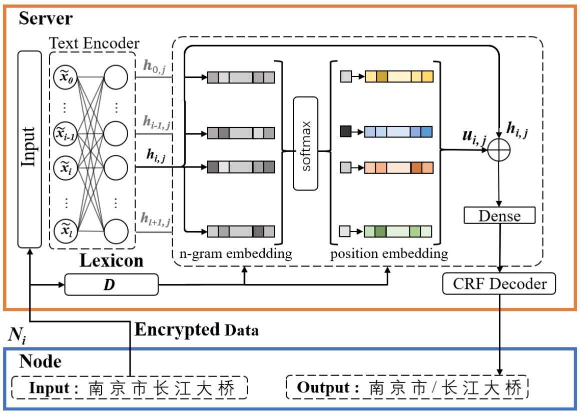

Figure 2: An overview of GCA-FL, where centralized model is illustrated on the top and the example input sentence “南京市长江大桥” (*Nanjing Yangtze River Bridge*) from $\mathcal{N}_i$ is shown on the bottom. “/” in the output represents the delimiter for the word boundaries.

2.1 Federated Learning

In the training process of FL, the node $\mathcal{N}_i$ firstly encrypts the original input character sequence $\mathcal{X}_i$ into $\widetilde{\mathcal{X}}_i = \widetilde{x}_{i,1} \cdots \widetilde{x}_{i,j} \cdots \widetilde{x}_{i,l}$, where $\widetilde{x}_{i,j}$ denotes the j-th character in $\widetilde{\mathcal{X}}_i$. Next, $\mathcal{N}_i$ passes $\widetilde{\mathcal{X}}_i$ to the server. Then, the centralized model on the server processes $\widetilde{\mathcal{X}}_i$ and predicts the corresponding label sequence $\widehat{\mathcal{Y}}_i = \widehat{y}_{i,1} \cdots \widehat{y}_{i,j} \cdots \widehat{y}_{i,l}$ by

$$\widehat{\mathcal{Y}}_i = \text{GCA-FL}\left(\widetilde{\mathcal{X}}_i\right) \tag{1}$$

where $\widehat{y}_{i,j} \in \mathcal{T}$ ($\mathcal{T}$ is the label set) is the segmentation label for $\widetilde{x}_{i,j}$. Afterwards, $\widehat{\mathcal{Y}}_i$ is passed back to $\mathcal{N}_i$ and compared with the gold label sequence $\mathcal{Y}_i^*$, after which the loss $\mathcal{L}_i$ for that training instance is obtained locally. Finally, $\mathcal{L}_i$ is passed to the server and the parameters in the centralized model are updated accordingly.

2.2 Centralized Model with GCA

In standard FL framework, the backbone centralize model works following the encoding-decoding paradigm, where $\widetilde{\mathcal{X}}_i$ is encoded[2] into a sequence of hidden vectors ($\mathbf{h}_{i,j}$ denotes the hidden vector for $\widetilde{x}_{i,j}$), which are then sent to a decoder (e.g.,

---

[2] One can use any encoder, e.g., biLSTM, for this process.

Rule	$v_{i,j,k}$
$x_{i,j}$ is the beginning of the n-gram $s_{i,j,k}$	V_B
$x_{i,j}$ is inside the n-gram $s_{i,j,k}$	V_I
$x_{i,j}$ is the ending of the n-gram $s_{i,j,k}$	V_E
$x_{i,j}$ is the single-character n-gram $s_{i,j,k}$	V_S

Table 1: The rules for assigning different position patterns to $x_{i,j}$ based on its position in an n-gram $s_{i,j,k}$.

CRF) to obtain the prediction $\widehat{\mathcal{Y}}_i$. However, data from different nodes (sources) always contains heterogeneous vocabularies and expressing patterns, where standard FL may obtain inferior results for localized prediction because it cannot distinguish the contextual information from the isolated data. Therefore, motivated by previous studies that leverage n-grams to capture local contextual information (Song et al., 2009b; Pei et al., 2014; Chen et al., 2017; Higashiyama et al., 2019), we propose GCA to enhance standard FL by exploring the contextual information carried by n-grams in the running text and use it to guide the centralized model for making localized prediction.

Specifically, GCA contains three components, namely, a lexicon ($\mathcal{D}$) that contains global character n-grams, an n-gram embedding matrix that maps an n-gram in $\mathcal{D}$ to its embedding, and a position embedding matrix that maps a position pattern, i.e., the position (e.g., beginning, ending, and inside) of a character in an n-gram, to its embedding. For each character $\widetilde{x}_{i,j}$, GCA encodes the contextual information and uses it to enhance the centralized model in the following process. First, GCA extracts all m n-grams $s_{i,j,k}$ $(1 \leq k \leq m)$ associated with $\widetilde{x}_{i,j}$ from $\mathcal{D}$, where $s_{i,j,k}$ satisfies the conditions that it contains $\widetilde{x}_{i,j}$ and it is a sub-string of $\widetilde{\mathcal{X}}_i$. Next, according to the position of $\widetilde{x}_{i,j}$ in $s_{i,j,k}$, GCA finds the position pattern $v_{i,j,k}$ associated with $s_{i,j,k}$ and $\widetilde{x}_{i,j}$ based on the rules specified in Table 1. For example, if $\widetilde{x}_{i,j}$ =“市” (*city*) and $s_{i,j,k}$ =“市长” (*mayor*), the position pattern $v_{i,j,k}$ will be “V_B” according to the rules in Table 1, because $\widetilde{x}_{i,j}$ is at the beginning of $s_{i,j,k}$. Third, GCA applies n-gram embedding matrix and position embedding matrix to $s_{i,j,k}$ and $v_{i,j,k}$, respectively, and obtains the n-gram embedding $\mathbf{e}^s_{i,j,k}$ and the position embedding $\mathbf{e}^v_{i,j,k}$. Then, GCA computes the weights $p_{i,j,k}$ for position patterns $v_{i,j,k}$ by

$$p_{i,j,k} = \frac{\exp(\mathbf{h}_{i,j} \cdot \mathbf{W} \cdot \mathbf{e}^s_{i,j,k})}{\sum_{k=1}^{m} \exp(\mathbf{h}_{i,j} \cdot \mathbf{W} \cdot \mathbf{e}^s_{i,j,k})} \qquad (2)$$

Genres		Sent. #	Token #	OOV Rate
BC	Train	7,301	628K	-
	Dev	2,368	237K	7.6
	Test	2,383	235K	6.1
BN	Train	5,867	1,108K	-
	Dev	2,151	394K	2.7
	Test	2,068	427K	4.2
MZ	Train	5,250	970K	-
	Dev	1,647	299K	2.4
	Test	1,526	342K	4.5
NW	Train	6,603	1,094K	-
	Dev	2,029	331K	4.1
	Test	2,085	346K	1.4
Web	Train	6,094	839K	-
	Dev	2,085	285K	4.2
	Test	2,001	243K	8.2

Table 2: The number of sentences, word tokens, and the out-of-vocabulary (OOV) rate (in dev and test sets) with respect to the training set of five genres in CTB7.

where $\mathbf{W}$ is a trainable matrix that maps $\mathbf{e}^s_{i,j,k}$ to the same dimension as $\mathbf{h}_{i,j}$ to facilitate the inner production “$\cdot$”. GCA further applies $p_{i,j,k}$ to all position embeddings and obtain the representation of contextual information $\mathbf{u}_{i,j}$ for $\widetilde{x}_{i,j}$ by

$$\mathbf{u}_{i,j} = \sum_{k=1}^{m} p_{i,j,k} \cdot \mathbf{e}^v_{i,j,k} \qquad (3)$$

Afterwards, $\mathbf{u}_{i,j}$ is added $(+)$ to $\mathbf{h}_{i,j}$ to guide the backbone model for localized prediction, where the resulted vector is mapped into the output space by a trainable matrix $\mathbf{W}_o$ and bias $\mathbf{b}_o$ by

$$\mathbf{o}_{i,j} = \mathbf{W}_o \cdot (\mathbf{u}_{i,j} + \mathbf{h}_{i,j}) + \mathbf{b}_o \qquad (4)$$

Finally, $\mathbf{o}_{i,j}$ is fed into a CRF decoder to obtain the predicted segmentation label $\widehat{y}_{i,j}$ for $\widetilde{x}_{i,j}$.

3 Experimental Settings

3.1 Simulations

To test the proposed approach, we follow the convention of recent FL-based NLP studies (Liu et al., 2019; Huang et al., 2020b; Zhu et al., 2020; Sui et al., 2020) to build a simulated environment where isolated data are stored in five nodes. Each node contains one of the five genres (i.e., broadcast conversation (BC), broadcast news (BN), magazine (MZ), newswire (NW), and weblog (WEB)) in CTB7 (LDC2010T07)[3] for CWS. Therefore, data from different genres are distributed to the five nodes without overlapping (i.e., the data sources in

---

[3] `https://catalog.ldc.upenn.edu/LDC2010T07`

Hyper-parameters	Values
Learning Rate	$5e-6$, **1e-5**, $3e-5$
Warmup Rate	**0.1**, 0.2
Dropout Rate	**0.33**
Batch Size	**16**, 32

Table 3: The hyper-parameters tested in tuning our models. The best ones used in our final experiments are highlighted in boldface.

our simulation are heterogeneous), which is similar to the simulation setting of aforementioned previous studies. We split each genre into train/dev/test splits following Wang et al. (2011) and report the statistics (in terms of the number of sentences, word tokens, and OOV rate) in Table 2.

3.2 Baselines and Reference Models

To show the effectiveness of our approach with GCA, we compare it with a baseline model that follows the FL framework without using it. In addition, we also run two reference models without both FL and GCA, where all training instances are not isolated and are accessible to each other. Specifically, the first reference model (denoted by Single) is trained and evaluated on the data from a single node (genre). The second (denoted by Union) is trained on the union of training instances from all five nodes (genres) and evaluated on a single node. Herein, the Union reference model can be optimized on a particular local node to achieve the best localized prediction; on the contrary, models under the FL setting is stored on the server and shared by all nodes, so that optimizing the model on a particular node could significantly hurt the performance on others. Therefore, the setting of the Union reference model is the ideal situation which is hard to happen in real-applications and it thus provides a potential upper-boundary of model performance for FL-based approaches.

3.3 Implementation

A good text representation is generally a prerequisite to achieve outstanding model performance (Pennington et al., 2014; Song and Shi, 2018; Peters et al., 2018). To obtain a high qulity of text representation, in our experiments, we try two types of encoder in the centralized model, i.e., the Chinese version of BERT (Devlin et al., 2019)[4] and the large version of ZEN 2.0 (Song et al., 2021)[5], because they are pre-trained language models that have been demonstrated to be effective in many NLP tasks (Nie et al., 2020; Huang et al., 2020a; Song et al., 2020; Chen et al., 2020; Fu et al., 2020; Tian et al., 2020a,b,c,d, 2021a,b; Chen et al., 2021; Qin et al., 2021). For both BERT and ZEN 2.0, we use the default settings (i.e., 12 layers of multi-head attentions with 768 dimensional hidden vectors for BERT and 24 layers of multi-head attentions with 1024 dimensional hidden vectors for ZEN 2.0). We use the vocabulary in Tencent Embedding[6] (Song et al., 2018) to initialize our lexicon $\mathcal{D}$ and the n-gram embedding matrix, where n-grams whose character-based length higher than five are filtered out[7]. During the training stage, we fix the n-gram embedding matrix and update all other parameters (including BERT). For evaluation, we follow previous studies to use the F1 scores (Chen et al., 2017; Ma et al., 2018; Qiu et al., 2019). For other hyper-parameter settings, we report them in Table 3. We test all combinations of them for each model on the development set, where models achieve highest F1 score on the development set is evaluated on the test set (the best hyper-parameter setting in our experiments is highlighted in boldface).

4 Results and Analysis

4.1 Overall Results

Table 4 illustrates the experimental results (i.e., F1 scores) of our GCA-FL models and all the aforementioned baselines (i.e., FL) and reference models (i.e., Single and Union) with BERT (a) and ZEN 2.0 (b) encoders on the test set of BC, BN, MZ, NW, and Web from CTB7.

There are several observations from the test set results. First, models under the FL framework (i.e., FL and GCA-FL) outperform the reference model (Single) trained on the single node for both BERT and ZEN 2.0 encoder, which confirms that FL works well to leverage extra isolated data. Second, our GCA-FL model consistently outperforms the FL baseline on all nodes (genres), although the FL baseline with BERT and ZEN 2.0 has already achieved very good performance. This observation demonstrates the effectiveness of the proposed GCA mechanism to leverage contextual information to facilitate localized prediction. Third, it is

---

[4] We use the Chinese base model from `https://s3.amazonaws.com/models.huggingface.co/`.

[5] We download the Chinese version of ZEN 2.0 from `https://github.com/sinovation/ZEN2`.

[6] `https://ai.tencent.com/ailab/nlp/en/embedding.html`

[7] We use five as the threshold because most Chinese words contain no more than five characters.

	BC	BN	MZ	NW	WB	Avg.
Single	97.13	96.97	96.21	97.84	94.83	96.60
Union	97.80	97.49	96.74	98.44	95.30	97.15
FL	97.49	97.22	96.54	98.15	95.03	96.89
GCA-FL	**97.76**	**97.40**	**96.74**	**98.43**	**95.29**	**97.12**

(a) BERT

	BC	BN	MZ	NW	WB	Avg.
Single	97.43	97.38	96.33	98.11	95.14	96.88
Union	97.88	97.79	97.23	98.61	96.38	97.58
FL	97.62	97.56	96.88	98.44	95.88	97.28
GCA-FL	**97.83**	**97.76**	**97.01**	**98.50**	**95.94**	**97.41**

(b) ZEN 2.0

Table 4: Experimental results (i.e., F1 scores) of different models with BERT (a) and ZEN 2.0 (b) on the development sets of the five nodes (genres) of CTB7.

observed that GCA-FL achieves competitive results compared with the Union reference model in most cases. This observation is promising because the Union model has all training data available without suffering from the data isolation problem, which could provide a potential upper boundary for FL-based models. The results obtained from GCA-FL thus further confirm the effectiveness of GCA.

4.2 Effect of GCA

To analize the effect of GCA to leverage isolated extra data to facilitate localized prediction, especially for OOV, we illustrate the recall of OOV of different models (i.e., Single, Union, FL, and GCA-FL) on five nodes (genres) with BERT encoder in Figure 3. Similar to the experimental results in the main experiments, it is observed that FL and GCA-FL outperform Single model in identifying unseen words (OOV). Further, GCA-FL can outperform the FL baseline on the test data in all nodes, where the highest improvement is observed on the node storing data from newswire. One possible explanation could be that BC and BN contains similar texts to NW. GCA-FL can better learn from the similar data on these nodes and thus improves localized prediction, especially for OOV.

5 Conclusion

In this paper, we apply FL to CWS to leverage isolated data stored in different nodes and propose GCA to enhance the CWS model stored in the server. Specifically, our approach encodes the contextual information by associating the input characters with global character n-grams, and uses that information to guide the backbone model to make localized predictions. Experimental results under a

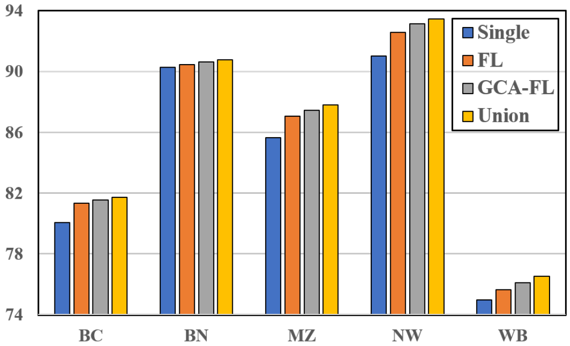

Figure 3: The recall of out-of-vocabulary words (OOV) of different BERT-based models (i.e., Single, FL, GCA-FL, and Union) on the test set of five nodes (genres).

simulated environment performed on five isolated nodes on CTB7 demonstrate the effectiveness of the proposed approach. Our approach outperforms the baseline model trained under the FL framework and achieves competitive results compared with the reference model that is trained on the union of the data from all nodes. Further analyses on identifying OOV justify the validity of the GCA mechanism to leverage the data on other nodes to facilitate localized prediction and demonstrate its great potential to be applied to real-world applications.

Acknowledgements

This work is supported by Chinese Key-Area Research and Development Program of Guangdong Province (2020B0101350001) and NSFC under the project "The Essential Algorithms and Technologies for Standardized Analytics of Clinical Texts" (12026610). This work is also partially supported by Shenzhen Institute of Artificial Intelligence and Robotics for Society under the project "Automatic Knowledge Enhanced Natural Language Understanding and Its Applications" (AC01202101001).

References

Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345.

Guimin Chen, Yuanhe Tian, and Yan Song. 2020. Joint Aspect Extraction and Sentiment Analysis with Directional Graph Convolutional Networks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 272–279.

Guimin Chen, Yuanhe Tian, Yan Song, and Xiang Wan. 2021. Relation Extraction with Type-aware Map Memories of Word Dependencies. In *Findings of* *the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. 2019. Federated Learning of N-Gram Language Models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 121–130.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-Criteria Learning for Chinese Word Segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jinlan Fu, Pengfei Liu, Qi Zhang, and Xuanjing Huang. 2020. RethinkCWS: Is Chinese Word Segmentation a Solved Task? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5676–5686, Online.

Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. FedNER: Privacy-preserving medical named entity recognition with federated learning.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated Learning for Mobile Keyboard Prediction. *arXiv preprint arXiv:1811.03604*.

Anxun He, Jianzong Wang, Zhangcheng Huang, and Jing Xiao. 2020. FedSmart: An Auto Updating Federated Learning Optimization Mechanism. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 716–724. Springer.

Shohei Higashiyama, Masao Utiyama, Eiichiro Sumita, Masao Ideuchi, Yoshiaki Oida, Yohei Sakamoto, and Isaac Okada. 2019. Incorporating Word Attention into Character-Based Word Segmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2699–2709.

Kaiyu Huang, Degen Huang, Zhuang Liu, and Fengran Mo. 2020a. A Joint Multiple Criteria Model in Transfer Learning for Cross-domain Chinese Word Segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3873–3882, Online.

Zhiqi Huang, Fenglin Liu, and Yuexian Zou. 2020b. Federated Learning for Spoken Language Understanding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3467–3478, Barcelona, Spain (Online).

Zhen Ke, Liang Shi, Erli Meng, Bin Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Unified Multi-criteria Chinese Word Segmentation with BERT. *arXiv preprint arXiv:2004.05808*.

Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Gina-Anne Levow. 2006. The Third International Chinese Language Processing Bakeoff: Word Segmentation and Named Entity Recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Dianbo Liu, Dmitriy Dligach, and Timothy Miller. 2019. Two-stage Federated Phenotyping and Patient Representation Learning. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 283–291, Florence, Italy.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese Word Segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based Neural Language Model and Chinese Word Segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1271–1277, Nagoya, Japan.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving Named Entity Recognition with Attentive Ensemble of Syntactic Information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4231–4245.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin Tensor Neural Network for Chinese Word Segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving Arabic Diacritization with Regularized Decoding and Adversarial Training. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Xipeng Qiu, Hengzhi Pei, Hang Yan, and Xuanjing Huang. 2019. Multi-Criteria Chinese Word Segmentation with Transformer. *arXiv preprint arXiv:1906.12035*.

Micah J Sheller, G Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. 2018. Multi-institutional Deep Learning Modeling without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation. In *International MICCAI Brainlesion Workshop*, pages 92–104.

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321.

Yan Song, Dongfeng Cai, Guiping Zhang, and Hai Zhao. 2009a. Approach to Chinese Word Segmentation Based on Character-word Joint Decoding. *Journal of Software*, 20(9):2236–2376.

Yan Song, Chunyu Kit, and Xiao Chen. 2009b. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 57–60, Suntec, Singapore.

Yan Song and Shuming Shi. 2018. Complementary Learning of Word Embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4368–4374.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*, pages 175–180, New Orleans, Louisiana.

Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. 2020. Summarizing Medical Conversations via Identifying Important Utterances. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 717–729.

Yan Song and Fei Xia. 2012. Using a Goodness Measurement for Domain Adaptation: A Case Study on Chinese Word Segmentation. In *LREC*, pages 3853–3860.

Yan Song and Fei Xia. 2013. A Common Case of Jekyll and Hyde: The Synergistic Effect of Using Divided Source Training Data for Feature Augmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 623–631, Nagoya, Japan.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*.

Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. 2020. FedED: Federated Learning via Ensemble Distillation for Medical Relation Extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2118–2128, Online.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese Word Segmentation Using Unlabeled Data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979.

Yuanhe Tian, Guimin Chen, and Yan Song. 2021a. Aspect-based Sentiment Analysis with Type-aware Graph Convolutional Networks and Layer Ensemble. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2910–2922, Online.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021b. Dependency-driven Relation Extraction with Attentive Graph Convolutional Networks. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020a. Improving Biomedical Named Entity Recognition with Syntactic Information. *BMC Bioinformatics*, 21:1471–2105.

Yuanhe Tian, Yan Song, Xiang Ao, Fei Xia, Xiaojun Quan, Tong Zhang, and Yonggang Wang. 2020b. Joint Chinese Word Segmentation and Part-of-speech Tagging via Two-way Attentions of Auto-analyzed Knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8286–8296, Online.

Yuanhe Tian, Yan Song, and Fei Xia. 2020c. Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020d. Improving Constituency Parsing with Span Attention. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1691–1703.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. 2020e. Improving Chinese Word Segmentation with Wordhood Memory Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8274–8285, Online.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving Chinese Word Segmentation and POS Tagging with Semi-supervised Methods Using Large Auto-Analyzed Data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand.

Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. 2020. Empirical Studies of Institutional Federated Learning for Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 625–634.

PSED: A Dataset for Selecting Emphasis in Presentation Slides

Amirreza Shirani[†], Giai Tran[†], Hieu Trinh[†], Franck Dernoncourt[‡],
Nedim Lipka[‡], Jose Echevarria[‡], Thamar Solorio[†], and Paul Asente[‡]
[†]University of Houston [‡]Adobe Research
[†]`{ashirani,gltran,httrinh,tsolorio}@uh.edu`
[‡]`{franck.dernoncourt,lipka,echevarr,asente}@adobe.com`

Abstract

Emphasizing words in presentation slides allows viewers to direct their gaze to focal points without reading the entire slide, retaining their attention on the speaker. Despite many studies on automatic slide generation, few have addressed helping authors choose which words to emphasize. Motivated by this, we study the problem of choosing candidates for emphasis by introducing a new dataset containing presentation slides with a wide variety of topics. We evaluated a range of state-of-the-art models on this novel dataset by organizing a shared task and inviting multiple researchers to model emphasis in slides.

1 Introduction

Presentation slides have become so commonplace that researchers have developed resources for designing effective slides (Alley and Robertshaw, 2004; Alley and Neeley, 2005; Jennings, 2009). These guidelines cover advice on the overall style, such as choosing colors and font size to ensure readability from a distance, as well as ways to help the content stand out more distinctly. However, recommendations to enhance the slides' communication power could improve authoring even more.

Our goal is predicting emphasis words in presentation slides. Emphasis uses special formatting like **boldface** or *italics* to make words stand out. Well-designed emphasis can significantly increase the viewers' retention by guiding their focus to a few words (Alley and Robertshaw, 2004). Instead of reading the entire slide, they can read only the emphasized parts, keeping their attention on the speaker and their speech, as Figure 1 illustrates.[1]

The Emphasis Selection (ES) task was initially introduced by Shirani et al. (2019) with a focus on short written text in social media, and later became a SemEval 2020 task (Shirani et al., 2020b). In this paper, we focus on presentation slides, introducing a new corpus as well as automated emphasis prediction approaches. We are among the first to use the content of the slides to provide automated design assistance.

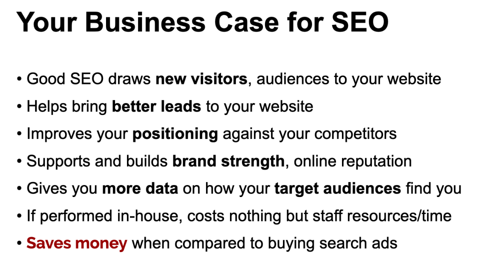

Your Business Case for SEO

- Good SEO draws **new visitors**, audiences to your website
- Helps bring **better leads** to your website
- Improves your **positioning** against your competitors
- Supports and builds **brand strength**, online reputation
- Gives you **more data** on how your **target audiences** find you
- If performed in-house, costs nothing but staff resources/time
- **Saves money** when compared to buying search ads

Figure 1: The slide uses special formatting to emphasize salient content.

Task Characteristics Emphasis selection poses new challenges specific to presentation slides. They can have different structures, and authors may follow traditional styles, or modern styles with more visual content. Slides cover a wide range of topics, from technical, marketing, and legal presentations to children's illustrations. The requirement to generalize to different domains and cover a variety of topics poses new challenges and encourages developing robust language understanding models. We rely only on input text without additional context from the user or the rest of the design. The task is highly subjective, but the goal is straightforward: use natural language understanding techniques to discover the most most common interpretation of a slide page and to generate emphasis that makes the page easier to understand quickly.

Benchmarking The Task Instead of providing baselines for the proposed dataset, we organized a shared task and invited researchers to work on the new corpus. Section 6 describes the top-performing methods. By examining the challenges of the

---

[1] Source: Web Marketing for Fundraisers: Get Found, Get Traffic, Get Ahead (`http://www.fundraising123.org/files/web-marketing-for-fundraisers-get-found-get-traffic-get-ahead652.pdf`)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4314–4320
August 1–6, 2021.

dataset, we provide different analysis components.

2 Related Work

Prior work explored automatically generating presentation slides from documents such as scientific articles (Beamer and Girju, 2009; Wang et al., 2017; Hu and Wan, 2013; Shibata and Kurohashi, 2005; Sravanthi et al., 2009). These projects assume that a slide page is a summarization of some part of the paper, and many summarization methods have been proposed to improve the effectiveness.

Other studies provide guidelines or alternatives to traditional designs to communicate a presentation's content more effectively (Alley and Robertshaw, 2004; Jennings, 2009; Alley et al., 2006; Atkinson, 2005; Doumont, 2005). These create slides with sentence headlines and visual elements to reinforce ideas and increase the audience's retention of the information during presentation.

Many applications provide design assistance for images and text, but most use only basic heuristics. Recent work uses AI-based models to recommend design attributes based on the content (Zhao et al., 2018b,a; Shirani et al., 2020a).

Shirani et al. (2019) introduced Emphasis Selection for written text in visual media. The proposed model with an end-to-end sequence tagging architecture utilizes label distribution learning (LDL) (Geng, 2016) to handle the task's subjectivity, and predicts emphasis scores for short written texts. They trained and evaluated the model against a collection of social media short texts from Adobe Spark[2]. Later on in SemEval 2020 (Shirani et al., 2020b), 31 teams proposed novel approaches to model emphasis more effectively. The organizers augmented the social media dataset with a large dataset of short quotations. Top-performing teams (Huang et al., 2020; Morio et al., 2020; Singhal et al., 2020) used rich contextualized pre-trained language models such as ERNIE 2.0 (Sun et al., 2020), XLMRoBERTa (Conneau et al., 2019), XLNet (Yang et al., 2019), and T5 (Raffel et al., 2019).

This study focuses on a new domain, presentation slides, where emphasis serves a different purpose than in social media. For social media the main purpose is to draw the audience's attention, while for presentations, the main purpose is to help the audience better understand the content. Identifying emphasis in presentations brings unique challenges due to differences in topic, length, and document structure.

3 Task Definition

Given a sequence of tokens in a slide page, $C = \{x_1, ..., x_n\}$, the task is to compute a real value $y_i \in [0, 1]$ for each x_i in C, indicating the degree to which the token needs to be emphasized.

4 Data Collection

The Presentation Slides Emphasis Dataset (PSED)[3] is a collection of presentation slides covering a wide range of topics, from technical slides on various topics to non-technical ones such as children's material. Each instance in PSED represents one slide page along with eight annotations. We only focused on English slides. To cover a wide range of topics and areas, we collected data from different sources such as websites with .ORG and .GOV domains and slides from the ACL anthology.[4] We pre-processed all slide pages to make sure they included clean pieces of text. We removed slides that only had equations, mathematical formulas, tables, or figures and used the PDFMiner Python library[5] to extract the text. Quality control steps ensured the text and the slide matched.

4.1 Annotation Process

In an MTurk experiment, we asked nine annotators to label each page. We showed the image of the slide as well as the corresponding text and asked workers to select words to emphasize as if they were preparing the slides for their own presentation. Ten percent of the hits included quality questions to make sure the annotators read the slides.

We observed a low Fleiss' Kappa score (Shrout and Fleiss, 1979) of 0.1414 on the dataset. A closer examination revealed that the dataset included some technical and domain-specific slides that were not entirely understandable to a general audience. Therefore, we removed slides with a score below -0.05 and the overall score increased to 0.1797. We also noticed that many cases included at least one annotator with a very different selection. To provide a more consistently-annotated data

---

[2] `https://spark.adobe.com`

[3] The dataset along with the annotations can be found here: `https://github.com/RiTUAL-UH/Predicting-Emphasis-in-Presentation-Slides-Shared-Task`.

[4] `https://www.aclweb.org/anthology/`

[5] `https://github.com/pdfminer/pdfminer.six`

set for training, we removed the annotator for each slide with the lowest agreement to the other annotators. The final dataset contains annotations from eight annotators and has a Fleiss' Kappa score of 0.2092. Such a score is similar to the score reported in (Shirani et al., 2020b) and indicates the existence of multiple points of view about emphasis in the dataset. Table 1 shows an example of a bullet point annotated with BIO annotation data. It shows that there is more agreement selecting words such as "risk" and "management" compared to the others.

Table 1: An example bullet point along with emphasis probabilities. "B" indicates the beginning of the emphasis, "I" the inside, and "O" non-emphasis words. "Freq." shows the frequencies of "B", "I" and "O". "Emphasis Probs.", shows the emphasis probability ("B+I") over eight annotations.

Words	Freq. [B,I,O]	Emphasis Probs. [B+I]
•	[0,0,8]	0.0
Demonstrate	[1,0,7]	0.125
how	[0,0,8]	0.0
operational	[1,0,7]	0.125
agencies	[1,0,7]	0.125
are	[0,0,8]	0.0
using	[0,0,8]	0.0
NASA	[2,0,6]	0.25
data	[0,1,7]	0.125
for	[0,0,8]	0.0
risk	[3,0,5]	0.375
management	[3,3,2]	0.75

5 Data Analysis

Table 2 provides more information on the number of slides, sentences, and words in the PSED dataset. The dataset contains 1,776 high-quality slides, randomly divided among training, development and test sets of 1,241, 180, and 355 instances respectively.

Table 2: Dataset Statistics

Section	#Slides	#Sentences	#Words
Train	1241	9645	96934
Dev	180	1251	12822
Test	355	2754	28108
Total	1776	13650	137864

Table 3 describes the length of instances in the PSED dataset, giving the minimum, mean, and maximum number of words in slides for each split.

Table 3: Statistics on the length of the samples computed in words

Section	Min	Mean	Max
Train	13	78	180
Dev	15	71	164
Test	17	79	181

As previous research has suggested, word types have a significant role in the selection of appropriate emphasis. Therefore, in this section, we examine the role of part-of-speech tags (POS) in this task. Specifically, we choose the top 20 POS tags, which frequently occur in the training and development sets, to analyze the feature's effectiveness. We used spaCy library[6] to obtain POS tags for all tokens. To examine how the emphasis probabilities are distributed, we divided them evenly into four intervals. Figure 2 shows the occurrence of the top 20 POS tags for all token labels in our training and development sets. POS tags such as "IN", ",", ".", and ":" are more favored to have low emphasis probabilities (0–0.25). Interestingly, some POS tags like "DT", "CD," and "VBZ" have zero words in the highest emphasis probability interval (0.75–1.0). Overall, most POS tags fall into the lowest emphasis probability, and the difference lies in the (0.25–0.5) interval, where POS tags like "NN", "NNS," and "VBG" mostly appear. Similar to POS tags, other hand-crafted features such as punctuation and upper-case tokens helped improve the results of some models. This motivated us to examine the degree of emphasis probability for different lexical features. Figure 3 shows the average emphasis scores for each category in the training and development sets. Comparing all lexical features, "Uppercase_start" has the highest average emphasis score, and "Contain_numbers" and "Punctuation" have the lowest. This indicates some general trends for emphasis with respect to word categories.

We also performed an error analysis to examine how the length of slides can affect the prediction. The results show that longer slides are more challenging due to having more options to select.

5.1 Evaluation Metric

For better comparison with previous work in ES, we followed an evaluation method similar to Shirani et al. (2020b). This metric is specifically designed to meet the subjectivity of the task.

---

[6] `https://spacy.io/usage/linguistic-features`

Table 4: Top-performing Models with Their Ranks and Score

Teams	Best Method	*RANK*	Score 1	Score 5	Score 10
UBRI-604	XLMRoBERTa Large + Lexical Features	**0.525**	**0.335** (1)	**0.686** (1)	0.554 (2)
DeepBlueAI	Ensemble of BERT, SciBERT, ERNIE 2.0	0.519	0.330 (2)	0.667 (3)	**0.559** (1)
Cisco	Ensemble of XLNet, RoBERTa + POS Tags	0.518	0.330 (2)	0.675 (2)	0.551 (3)
Baseline	BiLSTM+ELMo	0.475	0.301 (3)	0.634 (5)	0.489 (5)
Zouwuhe	N/A	0.474	0.285 (4)	0.638 (4)	0.500 (4)

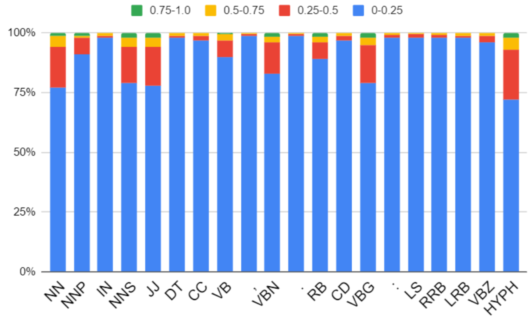

Figure 2: Frequencies of the top 20 POS tags in the 0–0.25, 0.25–0.5, 0.5–0.75, 0.75–1.00 probability intervals. Vertical values correspond to the percentage of tag counts over the total number of words in the training and development sets.

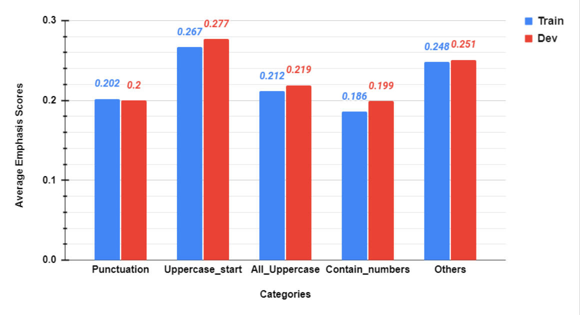

Figure 3: The figure shows average emphasis scores on the training and development sets for four different lexical features.

Match$_m$ For each slide page x in the test set D_{test}, we select a set $S_m^{(x)}$ of $m \in \{1, 5, 10\}$ words with the top m probabilities according to the ground truth. Similarly, we select a prediction set $\hat{S}_m^{(x)}$ for each $m \in \{1, 5, 10\}$, based on the prediction probabilities. Match$_m$ is defined as:

$$\text{Match}_m := \frac{\sum_{x \in D_{test}} |S_m^{(x)} \cap \hat{S}_m^{(x)}| / m}{|D_{test}|}$$

To rank models, we compute the average value of Match$_m$ for all m values and call this averaged value (*RANK*). We treat words in the ground truth with the same probability equally, so if the model predicts either of the tokens, we consider it as a correct answer.

6 Performance Benchmarks

To better examine the challenges of the dataset and benchmark the task, we organized a shared task and invited the community to participate in modeling emphasis in this new domain.[7]

Different novel and interesting solutions for this particular task were proposed. Table 4 shows the scores and the best methods for the top three teams. The most popular approach was ensemble Transformer-based models. Many hand-crafted features such as Part-of-speech (POS) tags, keywords, and lexical features (such as words with capital letters and punctuation) were explored to improve the models' performance. We describe and compare top-performing approaches next.

The top-performing team, *UBRI-604* (Hu et al., 2021), by proposing end-to-end Transformer-based approach, ranked in the first place with *RANK* score of (0.525). Different rich Transformer-based pre-trained language models were explored during the experiment, such as ALBERT (Lample and Conneau, 2019), GPT-2 (Radford and Wu, 2019), RoBERTa (Liu et al., 2019), ERNIE 2.0 (Sun et al., 2020), XLNet(Yang et al., 2019), XLMRoBERTa and BERT (Devlin et al., 2019). Comparing the results of all seven models, XLMRoBERTa performed the best. Besides pre-trained language models, *UBRI-604* leveraged lexical features such as capitalized words and punctuation, for further improvement.

DeepBlueAI team stood in second place (0.519), a *RANK* score that was 0.006 lower than the first team's. *DeepBlueAI* introduced an ensemble Transformer-based model with two fully-connected layers combined with POS tags embedding and hand-crafted features. The ensemble model takes advantage of BERT, SciBERT (Beltagy et al., 2019) and ERNIE 2.0 pre-trained language models by taking the average of the scores predicted by these models.

---

[7] CAD21 shared task: `https://competitions.codalab.org/competitions/27419`

Lastly, *Cisco* (Ghosh et al.), with a score 0.001 lower than the second team, ranked third. *Cisco* explored two approaches based on BiLSTM+ELMo (Shirani et al., 2019) architecture and Transformer-based pre-trained models with the base model of RoBERTa and XLNet. They enriched the ELMo contextual embedding in BiLSTM+ELMo model by incorporating a character-level BiLSTM Network. Their results show an increase of 0.026 when POS tags and keyphrases are added to the model, showing the effectiveness of these two features for this task. *Cisco*'s best score on the evaluation phase used an ensemble of XLNet and RoBERTa, giving them third place. They boosted the model further in the Post Evaluation phase by ensembling XLNet and BiLSTM+ELMo models and incorporating hand-crafted features like POS and Keyphrase.

We used the same baseline model (DL-BiLSTM+ELMo) introduced in Shirani et al. (2019) to better show the challenges of PSED dataset. This model achieved *RANK* score of 0.475 (Table 4) which is 0.275 lower than the reported score by Shirani et al. (0.75).[8] With a sequence-labeling architecture, this model utilizes ELMo contextualized embeddings (Peters et al., 2018) and two BiLSTM layers to label emphasis. The Kullback-Leibler Divergence (KL-DIV) (Kullback and Leibler, 1951) is used as the loss function during the training phase.

7 Discussion

The PSED dataset contains slides with different lengths. To better examine how the length of slides can affect the prediction, we performed an error analysis to examine this relationship. We divided the test set into three groups based on the instances' lengths, namely <60, 60–90, and >90 tokens. Then we computed the average Match_m scores over all shared task submissions, four in total, for every example in each group. As shown in Table 5, short slides always achieve better scores compared to medium and long slides. This indicates that predicting emphasis in longer instances is more challenging. This is due there being more options (words) to select for emphasis.

Many slides in the PSED dataset contain scientific words. Besides using pre-trained models, trained on a general domain, some teams decided to handle scientific words differently. For example, *DeepBlueAI* explored using the SciBERT (Beltagy et al., 2019) model, which is pre-trained on scientific articles. On the other hand, *Cisco* explored training a scientific keyword predictor and used the output as a feature to the model. Extending the proposed approaches to more efficiently address the diverse vocabulary of the dataset is an important future direction.

Table 5: Length vs. Performance on the test set. The average scores over all submissions are used for computing the performance. Short: (<60 tokens, 112 slides), Medium: (60–90 tokens, 126 slides), Long: (>90 tokens, 116 slides)

Length/Scores	*RANK*	Score 1	Score 5	Score 10
Short	**0.601**	**0.42** (1)	**0.634** (1)	**0.75** (1)
Medium	0.55	0.349	0.589	0.713
Long	0.485	0.293	0.526	0.635

8 Conclusion

We presented a new dataset for emphasis selection on presentation slides, posing new challenges for modeling emphasis. We created a shared task and invited researchers to model emphasis for presentation slides. We provided different data analyses on the dataset and summarized the insights gained from the shared task. A future extension could explore more robust techniques to address the challenges in the PSED dataset because of its diversity in topic, structure, and length.

9 Ethics

The proposed data in this work is collected from public domain sources and do not intrude on user privacy. For the manual work in annotation process, crowd workers were fairly compensated ($0.55 reward per response, which is over the US minimum wage).

Acknowledgments

We thank the reviewers for their thoughtful comments and efforts towards improving our work. We also thank Andrew Greene for his help in creating the corpus.

References

Michael Alley and Harry Robertshaw. 2004. Rethinking the design of presentation slides: Creating slides

---

[8] Match_m for $m \in \{1, 2, 3, 4\}$ is used in Shirani et al. (2020b).

that are readily comprehended. In *ASME International Mechanical Engineering Congress and Exposition*, volume 47233, pages 445–450.

Michael Alley, Madeline Schreiber, Katrina Ramsdell, and John Muffo. 2006. How the design of headlines in presentation slides affects audience retention. *Technical communication*, 53(2):225–234.

Michael P. Alley and Kathryn A. Neeley. 2005. Discovering the power of powerpoint: Rethinking the design of presentation slides from a skillful user's perspective. In *2005 ASEE Annual Conference and Exposition, Conference Proceedings*, pages 12325–12340.

Cliff Atkinson. 2005. *Beyond Bullet Points: Using Microsoft PowerPoint to Create Presentations That Inform, Motivate, and Inspire (Bpg-Other)*. Microsoft Press.

Brandon Beamer and Roxana Girju. 2009. Investigating automatic alignment methods for slide generation from academic papers.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jean-Luc Doumont. 2005. Slides are not all evil. *Technical communication*, 52:64–70.

X. Geng. 2016. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1734–1748.

Sreyan Ghosh, Sonal Kumar, Harsh Jalan, Hemant Yadav, and Rajiv Shah. Cisco at AAAI-CAD21 shared task: Predicting emphasis in presentation slides using contextualized embeddings. *arXiv preprint arXiv:2101.11422*. `https://archive.org/details/CAD21`.

GangQiang Hu, Chao Feng, HaoWen Lin, and Jian-Geng Chang. 2021. UBRI-604 at AAAI-CAD21 shared task: Predicting emphasis in presentation slides. `https://archive.org/details/CAD21`.

Yue Hu and Xiaojun Wan. 2013. Ppsgen: learning to generate presentation slides for academic papers. In *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer.

Zhengjie Huang, Shikun Feng, Weiyue Su, Xuyi Chen, Shuohuan Wang, Jiaxiang Liu, Xuan Ouyang, and Yu Sun. 2020. ERNIE at SemEval-2020 task 10: Learning word emphasis selection by pre-trained language model. *arXiv preprint arXiv:2009.03706*.

Ann Jennings. 2009. Creating marketing slides for engineering presentations. *Technical Communication*, 56.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Gaku Morio, Terufumi Morishita, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 task 10: Emphasis distribution fusion on fine-tuned language models. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1658–1664.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford and Jeffrey Wu. 2019. Rewon child, david luan, dario amodei, and ilya sutskever. 2019. *Language models are unsupervised multitask learners*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Tomohide Shibata and Sadao Kurohashi. 2005. Automatic slide generation based on discourse structure analysis. In *International Conference on Natural Language Processing*, pages 754–766. Springer.

Amirreza Shirani, Franck Dernoncourt, Paul Asente, Nedim Lipka, Seokhwan Kim, Jose Echevarria, and Thamar Solorio. 2019. Learning emphasis selection for written text in visual media from crowd-sourced label distributions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1167–1172, Florence, Italy. Association for Computational Linguistics.

Amirreza Shirani, Franck Dernoncourt, Jose Echevarria, Paul Asente, Nedim Lipka, and Thamar Solorio. 2020a. Let me choose: From verbal context to font selection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,

pages 8607–8613, Online. Association for Computational Linguistics.

Amirreza Shirani, Franck Dernoncourt, Nedim Lipka, Paul Asente, Jose Echevarria, and Thamar Solorio. 2020b. SemEval-2020 task 10: Emphasis selection for written text in visual media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1360–1370, Barcelona (online). International Committee for Computational Linguistics.

Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological Bulletin*, 86(2):420.

Vipul Singhal, Sahil Dhull, Rishabh Agarwal, and Ashutosh Modi. 2020. IITK at SemEval-2020 task 10: Transformers for emphasis selection.

M. Sravanthi, Ravindranath Chowdary, and P. Kumar. 2009. SlidesGen: Automatic generation of presentation slides for a technical paper using summarization.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Sida Wang, Xiaojun Wan, and Shikang Du. 2017. Phrase-based presentation slides generation for academic papers. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 196–202. AAAI Press.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Nanxuan Zhao, Ying Cao, and Rynson W.H. Lau. 2018a. Modeling fonts in context: Font prediction on web designs. *Computer Graphics Forum (Proc. Pacific Graphics 2018)*, 37.

Nanxuan Zhao, Ying Cao, and Rynson W.H. Lau. 2018b. What characterizes personalities of graphic designs? *ACM Transactions on Graphics (Proc. of SIGGRAPH 2018)*, 37.

MLMLM: Link Prediction with Mean Likelihood Masked Language Model

Louis Clouatre[*][†]**, Philippe Trempe**[*]**, Amal Zouaq**[*]**, Sarath Chandar**[*] [†] [∇]
[*] Polytechnique Montréal
[†] Mila - Quebec AI Institute
[∇] Canada CIFAR AI Chair
{philippe.trempe,amal.zouaq,sarath.chandar}@polymtl.ca
{clouatrl,sarath.chandar}@mila.quebec.ca

Abstract

Knowledge Bases (KBs) are easy to query, verifiable, and interpretable. They however scale with man-hours and high-quality data. Masked Language Models (MLMs), such as BERT, scale with computing power as well as unstructured raw text data. The knowledge contained within these models is however not directly interpretable. We propose to perform link prediction with MLMs to address both the KBs scalability issues and the MLMs interpretability issues. By committing the knowledge embedded in MLMs to a KB, it becomes interpretable. To do that we introduce MLMLM, **M**ean **L**ikelihood **M**asked **L**anguage **M**odel, an approach comparing the mean likelihood of generating the different entities to perform link prediction in a tractable manner. We obtain State of the Art (SotA) results on the WN18RR dataset and SotA results on the Precision@1 metric on the WikidataM5 inductive and transductive setting. We also obtain convincing results on link prediction on previously unseen entities, making MLMLM a suitable approach to introducing new entities to a KB.

1 Introduction

1.1 Context

KBs have many desirable properties. They are easy to query, verifiable, and perhaps most importantly interpretable by humans. They however have one critical shortcoming, they are expensive to build, making them harder to scale. Indeed, modern KBs scale with high-quality data, manual labor, or a mix of both. Approaches that scale with available computation and the massive amounts of unstructured data that are being created and accumulated have proven invaluable in the recent deep learning boom.

Large pretrained MLMs (Devlin et al., 2018; Liu et al., 2019) have been shown to scale well with large amounts of unstructured text data as well as with computing power. They also have displayed some interesting emergent abilities, such as the ability to perform zero-shot question answering (Radford et al., 2019; Brown et al., 2020). This ability implies that the model parameters contain a large amount of factual knowledge that it can leverage to answer a wide variety of questions. However, that knowledge is hardly interpretable by humans, as it is hidden within the millions to billions of parameters of the language model.

By using MLMs to completes KBs, we can address both the issue of scalability of KBs and the issue of the interpretability of MLMs by committing knowledge of the latter to an interpretable format in the former. The MLM can learn new knowledge from the large amount of unstructured textual data that keeps being added to the World Wide Web and then be used to continually complete and update the KB. This has the very desirable effect of making the link prediction approach scale with both computational power and a large quantity of unstructured data, both of which show no sign of slowing down.

1.2 Problem Definition

Given an entity and a relation, we want to train an MLM to generate all entities completing the KB triple.

Several technical challenges had to be addressed to achieve proper link prediction with pretrained MLMs. The first one is tractability. It is well known that inference in the task of Link Prediction is extremely costly, to the point where validation and test sets are purposefully kept small and most datasets will shy away from containing millions of different entities (Wang et al., 2019). While smaller Link Prediction datasets (Dettmers et al., 2017; Toutanova and Chen, 2015) are limited to a few thousand entities, a dataset more representative of full sized KBs (Wang et al., 2019) would contain upwards of a million potential entities. A

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4321–4331
August 1–6, 2021.

model that would be used on such a dataset could not realistically require an inference step through an MLM for every potential entity completing a triplet. It is necessary to enable link prediction with as little inference to the model as possible, as performing inference on pretrained models is expensive. Otherwise, it could result in a model with no practical purposes, even if it obtained better leaderboard scores.

The second challenge has to do with the inference outputs format of the MLMs. The length of the output needs to be known at inference time (Devlin et al., 2018; Vaswani et al., 2017), making it hard to sample entities of varying lengths from it. The example ”Horses like _” could be completed with the word ”carrot” and ”long runs”, but those two answers require varying length of masked inputs to be filled. Work like Petroni et al. (2019) is limited to single token outputs, which is useful to probe the model for the presence of embedded knowledge, but is not usable in practice for tasks such as link prediction, as the missing entities will have variable lengths. Solutions have to be able to sample an MLM for entities of varying lengths to have practical applications.

Finally, the use of MLMs opens the door to performing link prediction on entities that have not been previously seen by the model or the KB. This permits the addition of new entities to a KB on top of the link prediction capacities. Some capability of such an approach with MLMs was previously demonstrated (Petroni et al., 2019) and other works have approached the task by generating and comparing entities embedding (Daza et al., 2020; Gupta et al., 2017; Wang et al., 2019) for different KB tasks. By generating entity embeddings from text, they permit apt representation for previously unseen entities that can then be compared to other, previously seen entities. Unlike previous approaches we forgo the entity embedding step and let the model directly output the entity. We show that our approach yields strong results with unseen entities of arbitrary lengths in this task and should be explored further.

1.3 Contribution

Our main contributions are summarized here:

- We propose MLMLM, a mean likelihood method to compare the likelihood of different text of different token lengths sampled from an MLM.
- We demonstrate the tractability of our approach, requiring only one inference step through the model to perform link prediction on any numbers of possible entities, something which was not previously possible with an MLM.
- We achieve SotA results on the WN18RR benchmark and the best Precision@1 on both the inductive and transductive setting of the WikidataM5 dataset.
- We demonstrate that our approach can generalize to previously unseen entities on all benchmarks.

2 Background and Related Work

2.1 Masked Language Models

Pretrained MLMs, popularized by BERT (Devlin et al., 2018), have seen tremendous success when applied to Natural Language Understanding (NLU) problems. They are pretrained on massive amount of unsupervised text data. Those models incorporate enormous amounts of language knowledge and world knowledge within their weights. This lets them be further tuned on challenging NLU tasks with great success. Being based on the transformer (Vaswani et al., 2017) encoder architecture, the output length of the model is equal to the input length. This makes it challenging to sample text of arbitrary length when using MLMs without knowing the length of the desired sample in advance.

2.2 Sampling from MLM

Sampling single words from an MLM is trivial. By adding a mask to the input, we can sample likelihoods for the whole vocabulary. This feature is used by several pieces of work to complete sentences, answer questions and more (Guu et al., 2020; Lewis et al., 2020; Petroni et al., 2019). Work to generate and evaluate multi-token spans from MLM has also yielded interesting results (Wang and Cho, 2019; Salazar et al., 2020). We are however unaware of any other approach to sampling and evaluating multi-token spans of variable length, which is necessary to properly accomplish the task of link-prediction in a single pass through the model.

2.3 A Re-evaluation of Knowledge Graph Completion Methods

Recently, Sun et al. (2020) has found that many of the SotA approaches to link prediction have used an inappropriate evaluation protocol. They have shown that the evaluation protocol typically used in the link prediction approaches assigns a perfect score to a constant output, by putting the correct entities on top during a tiebreaker. In essence, under this evaluation protocol, assigning a likelihood of 0 to all entities would yield a perfect reranking score, since the tiebreaker would put the target entity as the first prediction. This was shown to yield very inflated scores for many neural network based link prediction approaches (Nathani et al., 2019; Vu et al., 2019; Nguyen et al., 2017), as several of them output a large number of tied scores for the various entities. Entity-embedding based approaches (Balažević et al., 2019; Sun et al., 2019; Dettmers et al., 2018) do not suffer from this issue. While we have found that our approach does not suffer from this issue despite not being an entity-embedding approach, we will use the random evaluation protocol proposed by Sun et al. (2020) for all evaluations and compare against approaches that used a similar protocol to ensure the validity of the comparisons. This protocol is similar to the filtered setting (Bordes et al., 2013), with the difference that the rank among entities with tied scores is randomly assigned.

2.4 KG-BERT

KG-BERT (Yao et al., 2019) is an approach to KB tasks based on MLM. It successfully demonstrates the potential of leveraging these models' internal knowledge on KB tasks. They train a BERT model to classify whether an individual triple fed to the model is correct or not. In essence, they feed every single possible (h, r, ?) or (?, r, t) triple in a string format to the model to obtain all scores to be reranked. This can result in millions of inference steps on the MLM for a single triple completion depending on the size of the KB. KG-BERT has many advantages over our proposed approach. It uses the target entity in the input, thus giving the model more information to use at inference time. The problem that it solves is much simpler, reducing it to a simple sequence classification problem. With a similar setup, it is even likely that KG-BERT would yield better results than our proposed approach since it has access to more information and has a more straightforward training setup. Unfortunately, KG-BERT is not tractable on any reasonably large KB (see Figure 1). For a KB containing millions of entities, KG-BERT would require millions of inference step through the MLM model for every triple completion. In contrast, our approach requires only one inference step through the MLM model for every triple completion, by generating all logits required to obtain the likelihood of any potential entity at once. Modern KBs can contain millions of entities (Wang et al., 2019), which would translate in KG-BERT requiring hours to complete a single triplet on a GPU.

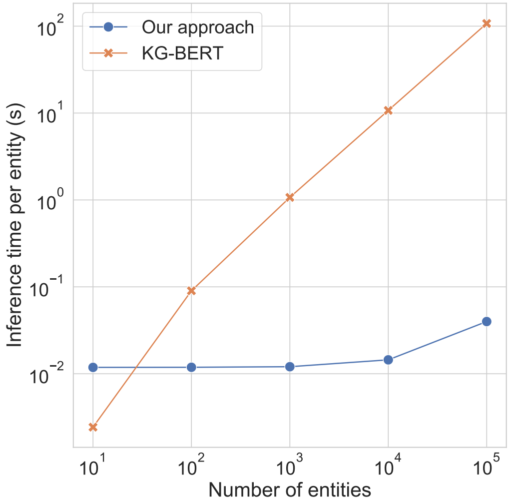

Figure 1: Approach Inference Time. This figure shows the per-entity inference time based on the total number of entities to be re-ranked, of MLMLM and KG-BERT, the most comparable approach.

3 Methodology

3.1 Overview

Our system performs link prediction. It uses MLM to generate all possible logits of all tokens required to generate all entities, and mean likelihood sampling to rerank all possible entities to complete the triple and perform link prediction. It can also be used to sample likelihoods for previously unseen entities.

Figure 2 shows a toy example of our inference setup. Our model outputs logits for the whole vocabulary of the RoBERTa-Large model. The outputs are simply the logits for all words and subwords from that vocabulary. This vocabulary is expressive enough to generate any English text.

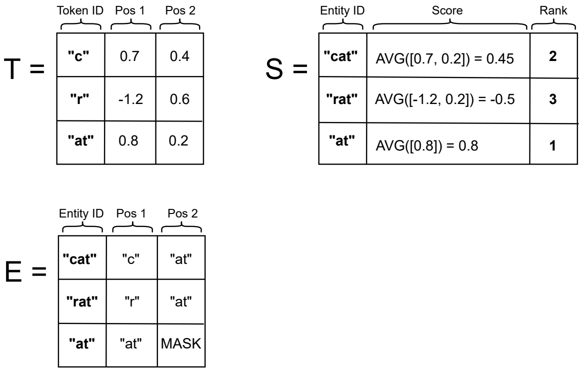

T =

Token ID	Pos 1	Pos 2
"c"	0.7	0.4
"r"	-1.2	0.6
"at"	0.8	0.2

S =

Entity ID	Score	Rank
"cat"	AVG([0.7, 0.2]) = 0.45	2
"rat"	AVG([-1.2, 0.2]) = -0.5	3
"at"	AVG([0.8]) = 0.8	1

E =

Entity ID	Pos 1	Pos 2
"cat"	"c"	"at"
"rat"	"r"	"at"
"at"	"at"	MASK

Figure 2: Ranking Example. The figure contains a minimal example of the ranking system. It represents the system in a KB containing 3 subwords in its vocabulary ['c', 'r', 'at'], 3 entities to rank ['cat', 'rat', 'at'] and a maximum entity length of 2.

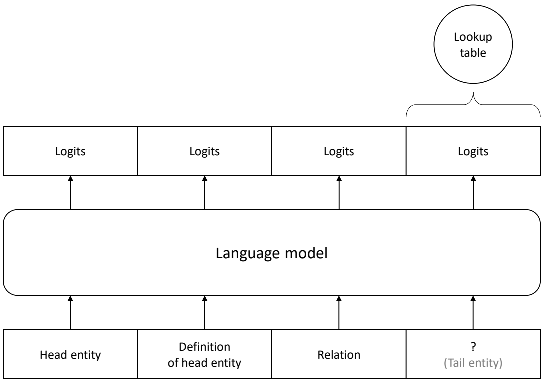

Figure 3: Lookup Table Generation For Tail Entity Prediction. The figure shows how the lookup table for tail entity prediction is generated. A string representation of the head entity and the relation are fed to the masked language model which outputs logits that represent the likelihood of finding each token at each possible position of the tail entity.

We decide in advance on a maximum length n for the maximum length of English text to generate. With a vocabulary of roughly 50,000 logits, the model would output a vector of logits of shape [50,000 x n]. Knowing in advance the tokens that would build the string for all possible entities, we can obtain a score for the likelihood of those entities completing a triplet by averaging the logits of those tokens. To evaluate the likelihood of the entity ”brown dog”, we would average the logits for the word ”brown” on the first column of the matrix with the logits of the word ”dog” on the second column of the matrix, ignoring all other columns of our logit matrix. Even if the model would never have encountered the entity ”brown dog” it could still produce a score for said entity.

3.2 Data Pre-processing

The data pre-processing pipeline takes a link prediction dataset and transforms it into a generic format usable by the model. It is required that both the entity and relations have string representations. For every entity in the dataset, we extract an entity string, which uniquely identifies the entity, and a definition string, which is a textual description of the given entity. For every relation, we extract a relation string, which uniquely identifies and describes the relation.

We tokenize all strings through the pretrained RoBERTa tokenizer (Sennrich et al., 2016) and further transform the entity string by adding padding to match the longest tokenized entity within the dataset. Concretely, in a dataset where the longest entity has a length of 4 token ids, the entity string “dog” would be padded to have the representation “dog _ _ _” and the entity string ”cat and dog” would have the representation “cat and dog _” where “_” is the padding token. The purpose of this padding is to standardise the masked representation of all entities, therefore letting the model treat all entities in the same manner.

3.3 Model

Our approach uses the RoBERTa-Large model (Liu et al., 2019) for all experiments. We finetune the pretrained model on the link prediction datasets to generate the logits of the unknown entities. As our approach does a single call to the model to rerank all possible entities, it is acceptable to use the larger model for better performance. Figure 3 shows the inference process for tail entity prediction. Similarly, the head entity prediction takes as input the head entity mask, the relation, the tail entity and the tail entity definition. We use the relation string, the known entity string and the entity definition of the known entity string to make the model generate the logits representing the unknown entity string.

3.4 Ranking System

The ranking system pictured in Figure 4 performs link prediction on a given triple. The MLM outputs logits for all possible token ids and positions for the missing entity to complete the triple. This forms the lookup table T. The link prediction dataset contains a list of all possible entities. The token ids forming those entities make up E. We obtain

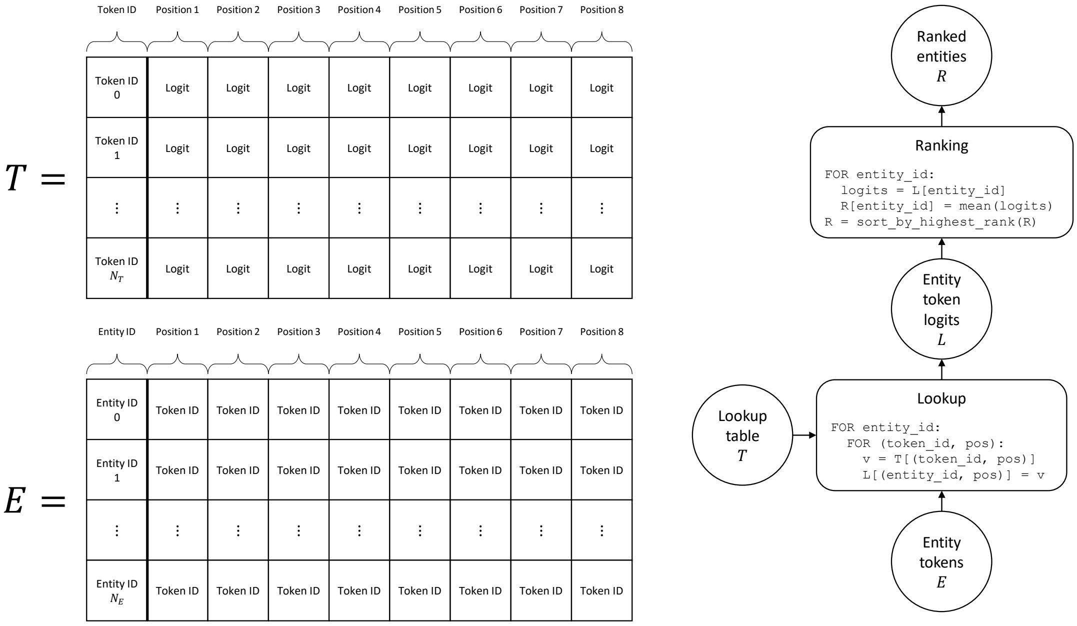

Token ID	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Token ID 0	Logit	Logit	Logit	Logit	Logit	Logit	Logit	Logit
Token ID 1	Logit	Logit	Logit	Logit	Logit	Logit	Logit	Logit
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Token ID N_T	Logit	Logit	Logit	Logit	Logit	Logit	Logit	Logit

Entity ID	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Entity ID 0	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID
Entity ID 1	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Entity ID N_E	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID

```
FOR entity_id:
  logits = L[entity_id]
  R[entity_id] = mean(logits)
R = sort_by_highest_rank(R)
```

```
FOR entity_id:
  FOR (token_id, pos):
    v = T[(token_id, pos)]
    L[(entity_id, pos)] = v
```

Figure 4: Ranking System. The figure details the inner workings of the ranking system which uses the lookup table generated by the masked language model to compute the score associated with each possible entity. The scored entities are then ranked by highest score.

the entity token logits L by matching all token ids in E with their corresponding values in T. L represents how likely every token of the entity is to be generated by the MLM at that specific position. The mean likelihood[1] of each entity is computed by averaging L over *non-padded* token logits.[2] This value is used to determine the ranking of the entity. It provides a proper comparison between entities of different lengths.

Concretely, in our previous "cat and dog _" example, we average the output logits for the "cat and dog" token ids and positions while ignoring the final padded logit. This averaging is done on all entities in the dataset completing the triple, yielding the average likelihood assigned by the model to all entities.

Entities are then sorted by highest rank using the randomized setting (Sun et al., 2020). For equal scores the tie-breaking is done randomly, to produce the ordered list of ranked entities R. We use the filtered setting (Bordes et al., 2013) for evaluation and remove corrupted triples from the list of ranked entities, corrupted triples being all other known correct triples.

4 Experimentation

4.1 Datasets

The two datasets used are WN18RR and WikidataM5 (Dettmers et al., 2017; Fellbaum, 1998; Bollacker et al., 2008; Wang et al., 2019), a commonly used link prediction benchmark and a new, large scale, link prediction benchmark. Summary stats are shown in Table 1.

Table 1: Datasets statistics.

	WN18RR	WikidataM5
#Entities	40,943	4,594,585
#Relations	11	822
#Training	86,835	20,614,279
#Validation	3034	5163
#Test	3134	5133
Mean in-degree	2.12	1
Median in-degree	4.49	0

WN18RR is a dataset composed of WordNet synsets. We use the cleaned synset as the entity string. The synset "dog.n.01" would have a string representation of "dog noun 1" which should be more interpretable by the model while remaining a unique identifier. The entity definition is the definition of the entity given by WordNet. The relation string is a cleaned representation of the relation

---

[1] Because the length of non-padded tokens is variable, using the mean of the logits is the chosen comparison metric for re-ranking.

[2] By far, the token the model sees most is the padding token. Counting it would most likely yield a heavy skew towards shorter entities with more padding.

string. The relation "_member_of_domain_usage" would be represented with the string "member of domain usage". Full examples of inputs and outputs are shown in Listing 1 and Listing 2.

WikidataM5 is composed of triples based on Wikidata and the English Wikipedia with aligned descriptions for each entity. We use the entity string and definitions provided in the benchmark.

4.2 Metrics

We use the Mean Reciprocal Rank (MRR) metric to validate our model and select the best model. For all experiments, we also report the Mean Rank (MR), the Mean Precision at 1 (MP@1), the Mean Precision at 3 (MP@3), and the Mean Precision at 10 (MP@10).

4.3 Training

The training setup is a modified MLM training, where we let the model generate the missing entity. The previously mentioned padding lets us deal with the generation of entities of varying sizes. The input fed to the model for tail entity prediction, depicted in Figure 3, consists of the concatenated token ids of the head entity, the head entity definition, the relation and the tail entity mask. The model will then generate, in the place of the mask, the missing entity. The input fed to the model for head entity prediction is similar. An example of the input for head entity prediction is found in Listing 1 and an example for tail entity prediction is found in Listing 2.

We use the categorical cross-entropy loss to train the language model. The loss only depends on the non-padded token of the generated entity, ignoring all other outputs. The target is the actual entity completing the triple, aligned with the mask in the input. We retain the model with the best validation MRR. All experiments are run for 5 random seeds and the mean and standard deviation of the results are reported.

For all experiments, we use the hyperparameters and training setup described in Liu et al. (2019), with a total of 25 epochs for the WN18RR dataset and 1 epoch for the WikidataM5 dataset.

4.4 Unseen Entities

An alternative version of the dataset is made to test the generalization capacity of our methodology to unseen entities. For WN18RR we start by randomly sampling 5% of the entities for the validation entities and 5% of the entities for the testing entities. Our training set consists of all triples not containing any of the validation or testing entities. Our validation set consists of all triples containing the validation entities. Finally, our test set consists of all triples containing the test entities, but not containing any of the validation entities. The training is done in the same fashion. The validation and testing are only done on entities present in the validation or test entity list, but are still reranked against all other possible entities. While WN18RR would only have 2047 test entities in this setting, the reranking would still involve 40,943 entities. If the tail entity is the one present in the test entity list, we will complete the link $(h, r, ?)$ and not the link $(?, r, t)$. The reported results are therefore only on the performance of previously unseen entities in the KB, compared to all other possible entities. The validation and test set are rebuilt for every random seed, to evaluate our approach on a wider array of unseen entities.

WikidataM5 has an inductive setting which closely resembles our unseen entities setting. The main difference is that the reranking step in the validation and test only compares with other previously unseen entities. While there is upwards of 4 million training entities, the validation and testing would only consider roughly 7000 entities. For the purpose of proper comparison, the M5 results are however done within their specific setting.

5 Results and Analysis

5.1 WN18RR

We achieve SotA results on the WN18RR dataset on all tested metrics with the exception of MR, as shown in Table 2. The WN18RR dataset is sparse in terms of the KG in-degree connections, see Table 1. Sparseness lends itself naturally to leveraging a pretrained model. The amount of information that can be extracted from the dataset on any given entity is then limited, which makes outside information all the more valuable.

We can observe that the MR metric is relatively much weaker for our model, compared to its other metrics. This implies that while the model will often rerank the correct entities to the top, it will also sometimes forget certain entities completely, given them ranks in the thousands. This could be explained by an issue of disambiguation in the name of the entity. While approaches using entity embeddings (Balažević et al., 2019; Sun et al., 2019; Dettmers et al., 2018) will have no issue separating

Table 2: WN18RR Results

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
ConvE	44.4	4950	—	—	50.3
RotatE	47.3	3343	—	—	57.1
TuckER	46.1	6324	—	—	51.6
ConvKB	24.9	3433	—	—	52.4
CapsE	41.5	718	—	—	55.9
KBAT	41.2	1921	—	—	55.4
KG-BERT	—	**97**	—	—	52.4
MLMLM	**50.17** **± 0.18**	1603 ± 26.8184	43.91 ± 0.20	54.18 ± 0.28	**61.10** **± 0.20**

The results are reported as <mean> ± <standard deviation>. Results for other models are taken from Sun et al. (2020); Yao et al. (2019).

Listing 1: Example of an error of the model on WN18RR. Shown are the top 3 ranked entities by the model with the score assigned to them. The correct answer, `matchmaker noun 1`, was ranked 14,108 by the system.

```
Prompt : <s><mask><mask><mask><mask><mask><mask><mask><mask>hypernym mediator noun 1
     a negotiator who acts as a link between parties</s><pad><pad><pad><pad>
Correct answer : matchmaker noun 1<pad><pad><pad><pad>   Answer rank 14108
Rank 1   Score 32.0242   : interpreter noun 2<pad><pad><pad>
Rank 2   Score 32.0103   : harmonizer noun 1<pad><pad><pad>
Rank 3   Score 31.8889   : diplomat noun 1<pad><pad><pad>
```

Table 3: WikidataM5 Results

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
TransE	**25.3**	109370	17.0	**31.1**	**39.2**
DKRL	16.0	31566	12.0	18.1	22.9
KEPLER-Wiki	15.4	**14454**	0.105	0.174	0.244
KEPLER-Cond	21.0	20267	17.3	22.4	27.7
MLMLM	22.3	488161	**20.1**	23.2	26.4

A single seed was ran for the WikidataM5 experiments because of the size of the dataset. Results for other models are taken from Wang et al. (2019).

Table 4: WN18RR Unseen Entities Result

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
Random baseline	0.03 ± 0.007	20541.91 ± 87.88	0.002 ± 0.004	0.002 ± 0.004	0.026 ± 0.008
Non-finetuned RoBERTa	2.73 ± 0.05	10130.35 ± 187.61	1.54 ± 0.07	2.95 ± 0.11	4.92 ± 0.19
MLMLM	18.42 ± 2.66	3761.50 ± 255.4437	14.16 ± 0.81	21.75 ± 1.19	29.39 ± 0.88

The results are reported as <mean> ± <standard deviation>.

the synsets `dog.n.01` and `dog.n.03` as meaning respectively “a member of the genus Canis [...]” and “informal term for a man”, our model will have to discern between those two meanings only by the digit appended to the name. It is probable that the model is often confused about whether it should generate `dog noun 1` or `dog noun 3`, having only the final digit to differentiate both of

Listing 2: Example of a disambiguation error of the model on WN18RR. Shown are the top 3 ranked entities by the model with the score assigned to them. The correct answer, `aid noun 3`, was ranked second by the system, after `aid noun 1`.

```
Prompt : <s>grant noun 1 any monetary aid hypernym<mask><mask><mask><mask><mask><
    mask><mask><mask></s><pad><pad><pad><pad>
Correct answer : aid noun 3<pad><pad><pad><pad><pad>     Answer rank 2
Rank 1   Score 33.7597   : aid noun 1<pad><pad><pad><pad><pad>
Rank 2   Score 33.5948   : aid noun 3<pad><pad><pad><pad><pad>
Rank 3   Score 32.7605   : aid noun 2<pad><pad><pad><pad><pad>
```

Table 5: WikidataM5 Inductive setting Results

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
DKRL	23.1	78	5.9	32.0	54.6
KEPLER-Cond	**40.2**	**28**	22.2	**51.4**	**73.0**
MLMLM	28.4	932	**22.6**	28.5	34.8

The results are reported as <mean> ± <standard deviation>.

them. An example of such an error is shown in Listing 2, where the model confuses `aid.n.01` and `aid.n.03`. Follow up work on better representations for entity names could yield stronger results.

Our model generally has a much easier time predicting the tail entity than the head entity. It has an MRR of 60.15 on tail entities and an MRR of 40.09 on head entities. By observing the instances where our model gives the worst rank to the correct answer, we can understand why. A large number of those cases are hypernyms on the head entity. An example of a hypernym relationship would be: "animal is an hypernym of dog, since all dogs are animals." Correctly ranking all possibilities for "X is an hypernym of dog." is more straightforward for the model than correctly ranking all possibilities for "Animal is an hypernym of Y.". An example of such failure is shown in Listing 1, where we look for the hypernym of the term `mediator`. It is clear that the model understands the concept and outputs plausible answers in its top 3. A large amount of the model's severe failure cases are similar to this one, where the model will output a plausible hypernym of the tail entity, while completely missing the targeted hypernym. This seems to be the likely cause for the weak MR of the approach.

5.2 WikidataM5

The results on WikidataM5 are shown in Table 3. MLMLM boasts the best Precision@1 metric by a fair margin. Once again, we observe the weakness in the MR metric. The implications are that there are many possible correct entities that are given no weights by the approach.

5.3 Unknown Entities Experiments

We demonstrate the capacity of our approach to generalize to unknown entities. Results for the WN18RR datasets are shown in Table 4.

For baselines, we use a random baseline, reranking the entities randomly, as well as a non-finetuned RoBERTa-large model, that simply generates the entity tokens without being finetuned on the dataset first. We can notice that while our approach outperforms a non-finetuned benchmark, the non-finetuned RoBERTa model still far outperforms the random baseline, supporting some of the findings of Petroni et al. (2019) in the capacity of MLM to perform unsupervised link prediction.

The M5 inductive settings are reported in Table 5. We obtain the best Precision@1 metric. The weakness in MR is once again visible, supporting the intuition that while the model might generate the correct entity with high conviction, it will often not give positive score to all plausible entities, yielding a much worst average rank.

We believe that leveraging MLMs could eventually lead to automatically populating KBs with new entities, as new knowledge and new facts are created and added to the web.

5.4 Limitations

MLMLM comes with several limitations. Our approach to padding limits the size of an unknown entity to the size of the longest known entity. While it is unlikely to be limiting in practice, it is still a weakness of our approach to sampling. The model size can be very prohibitive and specialized hardware such as GPUs is required to run it in a timely fashion. The approach however remains tractable as it can provide likelihoods for all possible entities in a single inference call. Compared to entity-embedding based methods, our approach needs additional information in the form of meaningful string representations for both entities and relations. The lack of entity disambiguation is also a limiting factor that does not affect other approaches. Finally, our approach is liable to *forgetting* some entities, leading to comparatively much worst MR than prior approaches.

6 Conclusion

We have developed a methodology for training masked language models to perform link prediction. By leveraging the natural language understanding abilities of these models and the factual knowledge embedded within their weights, we have achieved a tractable approach to link prediction that yields state of the art results on a standard benchmark and the best Precision@1 on another competitive benchmark. We have also demonstrated the ability of our model to perform link prediction of previously unseen entities, making our approach suitable to introduce new entities to knowledge bases. More generally, we have introduced an approach to sampling text from a masked language model of varying lengths, which can have a wider use case.

Acknowledgements

This research was supported by Apogée Canada, Canada First Research Excellence Fund program and École Polytechnique Startup Fund PIED. SC is supported by a Canada CIFAR AI Chair and an NSERC Discovery Grant.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

D. Daza, Michael Cochez, and Paul T. Groth. 2020. Inductive entity representations from text via link prediction. *ArXiv*, abs/2010.03496.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, Copenhagen, Denmark. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Z. Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, F. Petroni, V. Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2017. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP/IJCNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and K. Kirchhoff. 2020. Masked language model scoring. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. 2020. A Re-evaluation of Knowledge Graph Completion Methods. *arXiv e-prints, accepted at ACL 2020*, page arXiv:1911.03903.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2180–2189.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *CoRR*, abs/1911.06136.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *ArXiv*, abs/1909.03193.

Modulating Language Models with Emotions

Ruibo Liu[1] **Jason Wei**[2] **Chenyan Jia**[3] **Soroush Vosoughi**[1]
[1]Dartmouth College [2]ProtagoLabs [3]University of Texas at Austin
ruibo.liu.gr@dartmouth.edu jason@protagolabs.com
chenyanjia@utexas.edu soroush@dartmouth.edu

Abstract

Generating context-aware language that embodies diverse emotions is an important step towards building empathetic NLP systems. In this paper, we propose a formulation of modulated layer normalization—a technique inspired by computer vision—that allows us to use large-scale language models for emotional response generation. In automatic and human evaluation on the MojiTalk dataset, our proposed modulated layer normalization method outperforms prior baseline methods while maintaining diversity, fluency, and coherence. Our method also obtains competitive performance even when using only 10% of the available training data.

1 Introduction

Building interactive systems that can understand and express human emotions has been a long-term goal of artificial intelligence (Shen and Feng, 2020; Huang et al., 2018; Salovey and Sluyter, 1997). Given a context, an intelligent agent ought to be able to generate responses that not only consider the context but also reflect a specified emotion, a task called *emotional response generation*. One common representation of emotions is through *emojis*, which often convey the underlying emotions in an utterance (Zhou and Wang, 2018). Table 1 shows an example generation in this formulation.

To tackle this problem, prior work has proposed a number of different models, including variants of sequence-to-sequence (Seq2Seq) models (Serban et al., 2016; Li et al., 2016a), variational autoencoders (VAE) (Gu et al., 2019; Shen et al., 2017; Zhao et al., 2017) and adversarial networks (Kong et al., 2019; Li et al., 2017). Their generated responses are often dull or generic, partially due to the limited training data for diverse emotions (Li et al., 2017). More recent studies have tried to pre-train language models (LMs) on specific domain data to pivot generation towards certain direction (Gao et al., 2020; Zhang et al., 2020; Yang et al., 2020; Keskar et al., 2019). However, training a LM from scratch can be costly, and collecting sufficient pre-training data in diverse emotions is also challenging, especially for low-resource emotions (Yang et al., 2019a).

Context: good game start morning off tigers v eagles.

Emotion	**Response**
😃	*good luck to all the eagles*
🙏	*i m not a tigers fan but we ve got a win*
💔	*we ve got to wait for tommorrow for the game*
👌	*hope you enjoyed the match with your team*

Table 1. Example generation of our method for four different emojis. Context is an actual random tweet, and emotion is specified by emojis.

In this work, we present a simple and easy-to-deploy technique that can enable pre-trained large-scale LMs to generate fine-grained emotional responses. Specifically, we inject emotional signals specified by 64 commonly used emojis via Modulated Layer Normalization (Mod-LN), a technique widely adopted in computer vision but whose potential has not been well studied yet in NLP. The main advantages of our method are:

- Instead of designing or re-training models from scratch, our method is plug-and-play. In this work, we show its effectiveness on BERT (2019) and GPT-2 (2019), but one can easily extend our method to other Transformer-based LMs.
- By fully exploiting the transfer learning ability of pre-trained LMs, we achieve comparable emotional response generation performance as prior best-performing work with only 10% of the training data, which is especially beneficial for low-resource scenarios.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4332–4339
August 1–6, 2021. ©2021 Association for Computational Linguistics

2 Approach

Given a context text and a specified emoji as a target emotion, we aim to generate responses that both reflect the emotion associated with the emoji and the semantic information in the context. In this work, we demonstrate how to inject target emotions through a modulation module of layer normalization (§2.1). We also provide data preparation and model adaptation strategies on two typical LMs (BERT and GPT-2) to aid reproduction and extension (§2.2).

2.1 Modulated Layer Normalization

Layerwise-normalization (LN) is commonly used in Transformer-based (Vaswani et al., 2017) language models (LMs) (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019b) to stabilize hidden state dynamics and reduce training time (Ba et al., 2016). In the vanilla implementation (Figure 1(a)), data are normalized by their own mean μ and standard deviation σ without relying on external inputs.

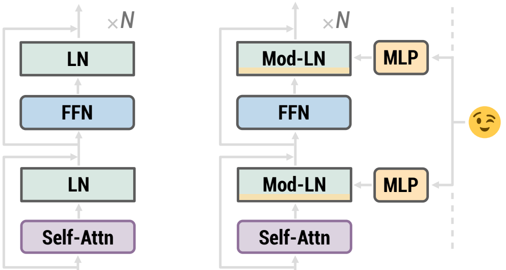

(a) Vanilla LayerNorm **(b)** Modulated LayerNorm

Figure 1: Overview of (a) Vanilla Layer Normalization (LN) and (b) Modulated Layer Normalization (Mod-LN) in Transformer-based LMs. The modulation module in Mod-LN uses two Multi-Layer Perceptrons (MLPs) that each have two sets of dense layers. It uses the external emotion input to modulate regularization towards a certain emotion c. FFN: Feed-Forward Network. Self-Attn: Multi-head Self-Attention blocks.

In contrast to vanilla LN that only regularizes data itself, Mod-LN introduces an external modulation module shared across the whole dataset, which is independent of the individual data samples and able to modulate the regularization towards external inputs c (Figure 1 (b)). Specifically, for an input hidden state tensor x in layer l, it is normalized by Mod-LN as

$$x = \text{MLP}_{\gamma}^{(l)}(c) \cdot \frac{x-\mu}{\sigma+\epsilon} + \text{MLP}_{\beta}^{(l)}, \quad (1)$$

where ϵ is the smoothing parameter to avoid dividing by zero. $\text{MLP}_{\gamma}^{(l)}$ and $\text{MLP}_{\beta}^{(l)}$ are two trainable modulation modules for a certain layer l. They are computed by

$$\text{MLP}_{\gamma}^{(l)}(c) = W_{\gamma}^{(l,2)} \cdot \text{Swish}(W_{\gamma}^{(l,1)} c), \quad (2)$$

$$\text{MLP}_{\beta}^{(l)}(c) = W_{\beta}^{(l,2)} \cdot \text{Swish}(W_{\beta}^{(l,1)} c + b), \quad (3)$$

where $W^{(l,1)}$ and $W^{(l,2)}$ are dense layers belonging to layer l, with weights size of $[64, \frac{1}{2} \cdot \text{dim}_h]$ and $[\frac{1}{2} \cdot \text{dim}_h, \text{dim}_h]$ respectively[1]. Dense layers connect 64 emoji classes to the output hidden states of the language model, and b is a bias added to γ. We use the Swish activation (Ramachandran et al., 2017), which has been shown to outperform ReLU (Xu et al., 2015) on several challenging datasets. Though conceptually simple, such MLP based modules have been shown to be a faster and more efficient alternative to vanilla dot product self-attention in NLP (Tay et al., 2021) and CV (Tolstikhin et al., 2021). Our work uses MLPs as a plug-and-play modulator rather than a replacement for self-attentions, allowing us to shift the hidden states towards a given target emotion.

2.2 Data Preparation and Model Adaptation

For the text input, we concatenate ground-truth context with corresponding response as a whole input to feed into LMs. We add a pre-defined separator token (`[SEP]` for BERT and `[UNK]` for GPT-2) between context and response, to make LMs aware of the range of each part. We also pad both context and response to a max sequence length with the padding token.

Encoder-Decoder models have been successful in many text-to-text generation tasks, such as question answering (Chen et al., 2017; Seo et al., 2017), news summarization (Chopra et al., 2016; Rush et al., 2015), and style transfer (Li et al., 2018; Liu et al., 2021). For the response generation task, the encoder encodes the context text into a fixed-length vector in latent space, while the decoder decodes the generated response tokens step-by-step, given the encoded context vector and the ground truth token from the previous step; this method is also known as teacher-forcing (Zhang et al., 2019c; Cho et al., 2014).

In this work, we consider leveraging the transfer learning power of large-scale LMs—using LMs

---

[1] The hidden size dim_h of bert-large-uncased and GPT-2 medium model are both 1024.

as encoder and decoder—to better capture the complicated relationship between context and response (Rothe et al., 2020). Auto-regressive LMs (ARLMs), such as GPT-2 are trained to iteratively predict the next step token given the past, while Masked Language Models (MLM), such as BERT, are trained to predict missing tokens given both the preceding and subsequent text. In contrast to the uni-directional attention flow in ARLM, the attention flow of MLM is bi-directional, and thus if we directly use MLM as decoder, the prediction of tokens in the response will also attend to (i.e., have the context of) future tokens; this could potentially lead to exposure bias (Schmidt, 2019). Inspired by recent text-to-text LMs such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2019), for MLM decoder, we modify the original bi-directional attention mask to make it uni-directional.

We experiment with two encoder-decoder models built on MLM and ARLM: 1) BERT-to-BERT: using bi-directional BERT as both encoder and decoder, but forcing the decoder BERT to attend to past context with uni-directional mask, and 2) GPT2-to-GPT2: using uni-directional GPT-2 as both encoder and decoder.

3 Experimental Setup

Dataset. For all the experiments, we use the MojiTalk (Zhou and Wang, 2018) dataset, a large Twitter conversation corpus ($N \approx 700k$) of responses that each contain one or more of 64 popular emojis. Following the original paper, we split the corpus into training, validation, and test sets of 596,959, 32,600, and 32,600 conversation pairs, respectively. We fine-tune the two LM-based encoder-decoder models on this dataset and generate responses given contexts and all possible emotions using top-k random decoding (Fan et al., 2018) on a machine with four RTX 2080 GPUs [2].

Models. We evaluate three models in total. We take the Reinforced Conditional Variational AutoEncoders (R-CVAE) model from Zhou and Wang (2018) as **Baseline** (current best-performing model on 64-emoji controlled response generation), **Mod-LN MLM**: BERT-to-BERT (large, uncased) + Mod-LN, and **Mod-LN ARLM**: GPT2-to-GPT2 (large) + Mod-LN.

Model	Emoji Acc (%)		
	Hits@1	Hits@3	Hits@5
Baseline: R-CVAE			
w/. 10% train data	13.4	27.1	33.6
w/. 100% train data	26.2	44.2	53.4
Mod-LN MLM			
w/. 10% train data	20.5	47.4	59.1
w/. 100% train data	33.6	56.8	72.2
Mod-LN ARLM			
w/. 10% train data	27.9	43.4	64.1
w/. 100% train data	34.4	60.3	82.5

Table 1: Accuracy of emotional response judged by DeepMoji on classifying emotions in responses generated by R-CVAE, MLM (BERT) with Mod-LN, and ARLM (GPT-2) with Mod-LN.

4 Evaluation

Good emotional responses should accurately reflect the intended emotion, be diverse, and have coherent language. We thus evaluate three aspects of generated responses: emotion control (§4.1), response diversity (§4.2), and coherence and fluency (§4.3). We also use Amazon Mechanical Turk (MTurk) to run a manual evaluation of emotion control and readability in generated responses (§4.4).

4.1 Emotion Control

First, we evaluate whether intended emotions were reflected in the responses generated by various models. We choose DeepMoji (Felbo et al., 2017)[3] as the judgment classifier. DeepMoji was trained on a large-scale emoji dataset containing 1,246 million tweets and 64 distinct emojis, and as far as we know, is state-of-the-art for 64-emoji classification tasks. Since the meanings of different emojis can overlap with subtle differences, we compute Hits@k ($k = \{1, 3, 5\}$) classification accuracy (Gao et al., 2020) to describe the performance of models in different criteria. As shown in Table 1, our proposed models outperform R-CVAE with a large margin. Of note, LM-based models reveal more robust performance in extreme data scarcity cases: our models achieve comparable performance with R-CVAE even when using only 10% of the training data. Between BERT and GPT-2, GPT-2 shows superior performance, partially because its weights are from auto-regressive pre-training.

---

[2] We choose $k = 10$ for a balance of generation diversity and readability through empirical observation.

[3] We chose the official implementation by huggingface: https://github.com/huggingface/torchMoji.

Model	TTR-1	TTR-2	Avg. len	%stop
Human Reference	0.059	0.43	11.7	50.4
Baseline: R-CVAE				
w/. 10% train data	0.034	0.24	8.6	60.1
w/. 100% train data	0.051	0.33	9.2	59.3
Mod-LN MLM				
w/. 10% train data	0.054	0.43	18.2	49.3
w/. 100% train data	0.059	0.39	14.3	49.2
Mod-LN ARLM				
w/. 10% train data	0.056	0.38	15.9	48.7
w/. 100% train data	0.057	0.40	12.5	48.5

Table 2: Lexical diversity of generated responses from various models. TTR-1/TTR-2: unigram/bigram type-token ratio; Avg. len: average number of tokens in generated responses; %stop: average percent of stop words among all tokens in the generated responses.

4.2 Generation Diversity

As shown in Table 2, we evaluate the diversity of responses generated by each model in terms of unigram and bigram type-token ratios, average length, and percent of stop words in generated responses, with values for the human-generated responses shown for reference. As measured by the type-token ratio for both uni- and bi-grams, our proposed models generate more diverse responses. In addition, compared with the R-CVAE, the responses generated by our models are longer and use fewer stop words. The advance can be attributed to the using of large-scale language models as base models.

4.3 Fluency and Coherence

Moreover, we evaluate the fluency and coherence of machine-generated text. For fluency, we trained a standalone language model on the human-generated responses using KenLM (Heafield, 2011) to measure the perplexity of generated texts. To evaluate coherence between the context and the generated responses, we compute the similarity between the generated text and human-generated responses using BERTScore (Zhang et al., 2019b), with the human-generated responses as reference. We configure the BERTScore using 24-layer RoBERTa-large (Liu et al., 2019) as for English tasks. Table 3 shows these results. For perplexity and BERTScore, our Mod-LN models outperform the R-CVAE in both 10% and 100% training data cases.

Model (vs Ref.)	PPL	BERTScore (%)		
		Precision	Recall	F1
Baseline: R-CVAE				
w/. 10% train data	121.18	74.9	83.0	76.7
w/. 100% train data	92.64	80.8	80.8	80.8
Mod-LN MLM				
w/. 10% train data	79.24	78.4	80.1	78.8
w/. 100% train data	50.72	82.9	84.1	83.5
Mod-LN ARLM				
w/. 10% train data	51.55	83.7	80.7	83.2
w/. 100% train data	36.31	84.7	86.2	85.4

Table 3: Fluency as measured by perplexity (PPL) and coherence as measured by BERTScore of generated responses from various models. Ref.: Human-generated responses.

4.4 Human Evaluation

In total 120 MTurk participants manually evaluated the emotion control and readability of responses from our proposed models and the original human-generated reference data. The average age of participants was 38.40 years-old (SD = 12.26, Median=34.50). More than half (65.8%) of participants were male, and 34.2% were female. The average completion time of each survey was 4.53 minutes. Participants were paid $1 per survey, averaging to more than $13 per hour wage for each participant, significantly above the U.S. federal minimum wage.

Procedure Each participant was assigned to read five randomly selected context-response pairs without being informed of the sources of the responses. They were asked to rate 1) emotion control: "*How well the emotion conveyed in the response agrees with the specified emoji?* (1-very well to 7-not at all)", and 2) readability: "*Please rate the readability of the response on a 7-point scale.* (1-very low to 7-very high)". The readability measure included five items adapted from a previous study (Graefe et al., 2018), specifically, well-written, concise, comprehensive, coherent, and clear. Since the five measures had very high agreement (Cronbach's[4] α = .91), we average the five measures into one as a general readability index.

Results The participant's averaged ratings (μ) and Standard Errors (SE) are reported in Table 4.

---

[4] Cronbach's alpha is a measure of internal consistency between sets of items.

Response Source	**Annotator Ratings**: μ (SE)	
	EMO CTRL.	READABILITY
Human Reference	5.62 (0.10)	5.34 (0.10)
Baseline: R-CVAE	4.80 (0.10)	4.67 (0.10)
Mod-LN MLM	5.43 (0.10)	5.20 (0.10)
Ablation: Vanilla GPT-2	4.98 (0.10)	4.64 (0.10)
Mod-LN ARLM	5.40 (0.10)	5.32 (0.10)

Table 4: Humans manually evaluated the emotional control and readability of responses from the original data (human reference), Baseline and proposed models on a 7-point scale (1: low quality, 7: high quality). We also take the generative LM: vanilla GPT-2, as the ablation reference.

As shown in the table, the standard error of the mean among all annotators is .10, which is very low for a 7-point scale, indicating large agreement between annotators. Responses generated by Mod-LN MLM (BERT), Mod-LN ARLM (GPT-2), and the human-generated references had no statistically significant differences in emotion control and readability. All were rated significantly higher than plain GPT-2 and R-CVAE in both emotion control and readability ($p < .001$ for one-way repeated measures ANOVA). We also conducted pairwise multiple comparisons in our analysis as post-hoc analysis. In terms of emotion control, both of our two proposed models and original reference data were rated significantly better than vanilla GPT-2 ($p < .007$). For readability, both our models, vanilla GPT-2, and original reference data were rated significantly more readable than R-CVAE ($p < .001$).

5 Related Work

Emotional Text Generation. VAE-based models (Park et al., 2018; Shen et al., 2017; Zhao et al., 2017; Serban et al., 2017), adversarial networks (Kong et al., 2019; Li et al., 2017; Yu et al., 2017) and reinforcement learning systems (Li et al., 2019, 2016b) have dominated sentiment-aware dialogue models. Other methods have been developed using LSTM (Song et al., 2019) and GRU (Wei et al., 2019; Zhou et al., 2018). All these methods, however, are built on relatively coarse emotion types, partially due to the limited modeling ability of RNNs. Our model outperforms current state-of-the-art R-CVAE (Zhou and Wang, 2018) in the same 64-emoji settings.

Modulated Normalization. Though not common in NLP, modulated normalization has been previously used in computer vision. In addition to work mentioned in the introduction (De Vries et al., 2017), adversarial networks such as CGAN (Miyato and Koyama, 2018), self-attention GAN (Zhang et al., 2019a) and Style GAN (Karras et al., 2019) have used modulated normalization to inject external signal into their models. In NLP, previous studies have tried to modulate normalization for classification tasks (Houlsby et al., 2019) and multilingual machine translation (Bapna and Firat, 2019), however, both these methods require architecture-level modifications. Our method, on the other hand, is plug-and-play, requiring minimal modifications to the architecture and thus easier to deploy for a diverse set of applications.

6 Conclusions

We have proposed a modulated layer normalization approach to generating responses of varying specified emotions. Our approach allows us to leverage large pre-trained models, while remaining simple and easily-extendable. In empirical experiments, our approach substantially outperforms prior work and achieves comparable results using only 10% of the available training data, all while maintaining diversity, fluency, and coherence.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. In *Advances in Neural Information Processing Systems, Deep Learning Symposium*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of* *the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. 2017. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark. Association for Computational Linguistics.

Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, and Bill Dolan. 2020. Dialogue response ranking training with large-scale human feedback data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 386–395, Online. Association for Computational Linguistics.

Andreas Graefe, Mario Haim, Bastian Haarmann, and Hans-Bernd Brosius. 2018. Readers' perception of computer-generated news: Credibility, expertise, and readability. *Journalism*, 19(5):595–610.

Xiaodong Gu, Kyunghyun Cho, Jung Woo Ha, and Sunghun Kim. 2019. Dialogwae: Multimodal response generation with conditional wasserstein autoencoder. In *7th International Conference on Learning Representations, ICLR 2019*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Chenyang Huang, Osmar R Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 49–54.

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Xiang Kong, Bohan Li, Graham Neubig, Eduard Hovy, and Yiming Yang. 2019. An adversarial approach to high-quality, sentiment-controlled neural dialogue generation. *arXiv preprint arXiv:1901.07129*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*, pages 110–119.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. pages 2157–2169.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Ziming Li, Julia Kiseleva, and Maarten de Rijke. 2019. Dialogue generation: From imitation learning to inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6722–6729.

Ruibo Liu, Chenyan Jia, and Soroush Vosoughi. 2021. A transformer-based framework for neutralizing and reversing the political polarity of news articles. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–26.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Takeru Miyato and Masanori Koyama. 2018. cgans with projection discriminator. In *International Conference on Learning Representations*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1792–1801.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Peter Ed Salovey and David J Sluyter. 1997. *Emotional development and emotional intelligence: Educational implications.* Basic Books.

Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, Hong Kong. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Lei Shen and Yang Feng. 2020. CDL: Curriculum dual learning for emotion-controllable response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 556–566, Online. Association for Computational Linguistics.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 504–509, Vancouver, Canada. Association for Computational Linguistics.

Zhenqiao Song, Xiaoqing Zheng, Lu Liu, Mu Xu, and Xuanjing Huang. 2019. Generating responses with a specific emotion in dialog. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3685–3695, Florence, Italy. Association for Computational Linguistics.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention in transformer models. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2021)*.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. 2021. Mlp-mixer: An all-mlp architecture for vision.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wei Wei, Jiayi Liu, Xianling Mao, Guibing Guo, Feida Zhu, Pan Zhou, and Yuchong Hu. 2019. Emotion-aware chat machine: Automatic emotional response generation for human-like emotional interaction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1401–1410.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Ze Yang, Wei Wu, Can Xu, Xinnian Liang, Jiaqi Bai, Liran Wang, Wei Wang, and Zhoujun Li. 2020. StyleDGPT: Stylized response generation with pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1548–1559, Online. Association for Computational Linguistics.

Ze Yang, Wei Wu, Jian Yang, Can Xu, and Zhoujun Li. 2019a. Low-resource response generation with template prior. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1886–1897, Hong Kong, China. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019a. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019c. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, Florence, Italy. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xianda Zhou and William Yang Wang. 2018. Mojitalk: Generating emotional responses at scale. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1128–1137.

Effective Batching for Recurrent Neural Network Grammars

Hiroshi Noji
Artificial Intelligence Research Center
AIST
hiroshi.noji@aist.go.jp

Yohei Oseki
Graduate School of Arts and Sciences
University of Tokyo
oseki@g.ecc.u-tokyo.ac.jp

Abstract

As a language model that integrates traditional symbolic operations and flexible neural representations, recurrent neural network grammars (RNNGs) have attracted great attention from both scientific and engineering perspectives. However, RNNGs are known to be harder to scale due to the difficulty of batched training. In this paper, we propose effective batching for RNNGs, where every operation is computed in parallel with tensors across multiple sentences. Our PyTorch implementation effectively employs a GPU and achieves x6 speedup compared to the existing C++ DyNet implementation with model-independent auto-batching. Moreover, our batched RNNG also accelerates inference and achieves x20-150 speedup for beam search depending on beam sizes. Finally, we evaluate syntactic generalization performance of the scaled RNNG against the LSTM baseline, based on the large training data of 100M tokens from English Wikipedia and the broad-coverage targeted syntactic evaluation benchmark.[1]

1 Introduction

Neural language models have an excellent word prediction ability, which motivates researchers to develop several analysis methods for fine-grained evaluation, aiming at understanding which linguistic abilities the models have acquired during training (Linzen et al., 2016; Wilcox et al., 2018; Marvin and Linzen, 2018; Warstadt et al., 2020). So far, many efforts have been made on the evaluation of *syntactic* performance of models, including the abilities to resolve distant subject-verb number agreement in English. Since neural language models are the foundation of contemporary NLP systems, building a language model having robust sentence processing abilities like humans is an important goal, especially toward a system with human-like syntactic generalization abilities, not relying on the data-specific superficial cues found in the training data (McCoy et al., 2019; Linzen, 2020).

Past work has revealed that while sequential and unstructured models, such as LSTM and Transformer language models (Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017), can induce several interesting syntactic behaviors, there is also a notable advantage in explicitly modeling syntax with specific architectures (Kuncoro et al., 2018; Wilcox et al., 2019; Hu et al., 2020). The representative of such models is the recurrent neural network grammars (RNNGs; Dyer et al., 2016), the top-down, left-to-right generative models of a parse tree and sentence.

While these results may suggest that RNNGs are a better modeling choice for language, unfortunately, they have a practical drawback in terms of scalability, due to their structure-sensitive computation mechanism (Kuncoro et al., 2019). Since the computational graphs of RNNGs depend on the tree structures of the sentences, training cannot be mini-batched easily. This is in contrast to LSTMs and Transformers, for which token-wise operations can be batched across sentences, allowing efficient computation on GPUs, which is the key to the data scalability. Although RNNGs are claimed to be a fascinating language model, in practice, they still do not replace the unstructured, computationally favorable models like LSTMs.

In this paper, we directly address the data scalability issue of RNNGs by showing that most computations during training can be batched across sentences. At the computational core of RNNGs is stack LSTMs (Dyer et al., 2015). In past work, Ding and Koehn (2019) have already shown that stack LSTM update operations can be reduced to a tensor operation by implementing the stack as a sin-

---

[1] Our RNNG implementation is available at https://github.com/aistairc/rnng-pytorch/.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4340–4352
August 1–6, 2021. ©2021 Association for Computational Linguistics

gle tensor with predefined maximum stack depth. Our work is built on this idea but with a few additional techniques to bridge the gap between the simple stack LSTMs and RNNGs. Importantly, we devise the efficient batching method for *composition* operations on the arbitrary number of stack items, which is unsolved in previous work.

The existing RNNG implementation is based on DyNet that supports the mechanism called Autobatch (Neubig et al., 2017), which automatically finds mini-batch units from the independent computational graphs over multiple sentences with lazy computation. While this mechanism is model-independent and allows intuitive implementation, the utility of this method rapidly plateaus as we increase the batch size. On the other hand, our present method allows effective parallel computation, increasing the training speed almost linearly as we increase the batch size.

In addition to this new batching mechanism for improved scalability, we also provide a new analysis on the role of the strong syntactic inductive bias for models that can access the larger amount of data. For syntactic generalization abilities, while Hu et al. (2020) suggest that the model inductive bias plays a more important role than data scale, they also report that LSTMs or Transformers such as the off-the-shelf large-scale models (e.g., GPT-2 or JRNN) perform much better than their scale-controlled LSTMs. Does an RNNG, which already works relatively well on a modest amount of data, still benefit from the data scale to further strengthen its syntactic ability? We train a new RNNG on about 100M tokens in Wikipedia and evaluate its syntactic performance on SyntaxGym test circuits (Gauthier et al., 2020), finding that the data scale generally brings further performance gains, while the model tends to lose some heuristics on surface patterns that LSTMs seem to find. Our result suggests that RNNGs' reliance on structures will be strengthened with more data, motivating future research on developing better syntactic representation itself as supervision to structured language models.

A related approach to our work is adding the syntactic bias into *sequential* language models, such as LSTMs, with knowledge distillation from RNNGs (Kuncoro et al., 2019, 2020). While motivations are similar, we provide a rather direct solution to resolve the scalability issue of RNNGs, opening up a new possibility of directly using them as an alternative to LSTMs.

From another perspective, our work can be complementary to this work, because knowledge distillation requires a teacher RNNG model, which itself is costly to obtain. For example, Kuncoro et al. (2020) trained an RNNG on a relatively large dataset of 3.6M sentences, which is approximately similar to the training data we use. While the detail is missing, they report that training takes three weeks on a GPU. On the other hand, our models get almost converged in three days. This direct improvement in training time greatly expands the applicability of RNNGs including a teacher of sequential models, and more direct use in computational psycholinguistics (Hu et al., 2020) and NLP applications such as syntactic neural machine translation (Eriguchi et al., 2017).

2 Preliminaries

2.1 Recurrent neural network grammars

RNNGs are joint generative models of a sentence and constituency tree. While RNN language models assign a next token probability, RNNGs assign a probability to next *action*, by which the parse state (stack LSTM) changes dynamically. In this work, we focus on the stack-only RNNG (Kuncoro et al., 2017), which has some resemblance to RNNs in that a single state vector $\mathbf{h}_t$ defines next action probability a_t:

$$a_t \sim \text{softmax}(\mathbf{W}_a \text{MLP}(\mathbf{h}_t) + \mathbf{b}_a)$$

At each step, $\mathbf{h}_t$ is obtained from the top element of stack LSTM, which preserves intermediate LSTM states up to $\mathbf{h}_t$. As a preparation for our batched RNNGs (Section 3), we try to formalize how this stack LSTM states change with each action. An RNNG internally preserves two different stacks: S_h and S_e. S_h is a stack LSTM, keeping the LSTM hidden states $\mathbf{h}_0 \cdots \mathbf{h}_t$.[2] S_e keeps stack elements, each of which is a word embedding $\mathbf{e}_w$, an open nonterminal embedding $\mathbf{e}_x$, or a closed constituent embedding $\mathbf{e}_c$ obtained by REDUCE action.

At each step, the number of candidate actions is $|\mathcal{N}| + 2$ given the set of nonterminal symbols $\mathcal{N}$. Each action changes S_h and S_e as follows:

- NT(x): Push open nonterminal embedding $\mathbf{e}_x$ onto S_e, getting a new LSTM state by $\mathbf{h}_{\text{new}} = \text{LSTM}(\text{top}(S_h), \mathbf{e}_x)$, and then push $\mathbf{h}_{\text{new}}$ onto

---

[2] Precisely, we also have to keep LSTM cell states. We omit this part for brevity.

S_h. This action corresponds to generating an open nonterminal, e.g, "(VP" (when x=VP), which will be closed later.

- GEN: First, generate a next token by sampling from $w \sim \text{softmax}(\mathbf{W}_w\text{MLP}(\mathbf{h}_t) + \mathbf{b}_w)$. Then, as in NT, push $\mathbf{e}_w$ onto S_e, getting a new LSTM state $\mathbf{h}_{\text{new}} = \text{LSTM}(\text{top}(S_h), \mathbf{e}_w)$, and push $\mathbf{h}_{\text{new}}$ onto S_h.

- REDUCE: First, repeatedly pop from S_e k-times until we find $\mathbf{e}_x$, an open nonterminal embedding. Letting $\mathbf{e}^{t-k} = \mathbf{e}_x$, then, apply a composition function, which is BiLSTM (Dyer et al., 2016) by default, to obtain a composed phrase representation $\mathbf{e}_c$:

  $$\mathbf{e}_c = \text{BiLSTM}([\mathbf{e}^{t-k}, \cdots, \mathbf{e}^t]).$$

  $\mathbf{e}_c$ is then pushed onto S_e. To synchronize two stacks, we also pop k-times from S_h and update the LSTM state by $\mathbf{h}_{\text{new}} = \text{LSTM}(\text{top}(S_h), \mathbf{e}_c)$, pushing it onto S_h.

By declaring the operations as above, we notice that the main reasons to prevent mini-batching are twofold: (1) the stacks have variable length, which varies at each step for each sentence; and more crucially, (2) internal operations in an action, especially in REDUCE and others, are largely different.

As we describe next, the issue regarding (1) has been largely solved in previous work. For (2), our strategy is essentially not joining different action types, but trying to improve the efficiency of each action as much as possible after grouping by action types. We find that in practice this strategy works quite well (Section 5.2), allowing models to benefit from a large batch size effectively.

2.2 Batched stack LSTMs

Ding and Koehn (2019) propose a sentence-level batched training algorithm for a restricted class of stack LSTMs designed for unlabeled dependency parsing without composition operations (Dyer et al., 2015). More specifically, Ding and Koehn (2019) deal with the parsing models defined by the following two operations only:[3]

- PUSH: Push $\text{LSTM}(\text{top}(S_h), \mathbf{e}_w)$ to S_h. $\mathbf{e}_w$ is the embedding of the next token.
- POP: Pop the top element from S_h.

At each step, the next action is either PUSH or POP for each sentence. This model still suffers from the problem (1) above. However, they show that by changing the data structure of stack, next PUSH and POP across sentences can be performed in batch. Given B sentences in a batch, let S_h^i be a stack for i-th sentence. What we need to do is to access all top elements of $S_h^i (i \in [0, \cdots, B-1])$ jointly, and this is possible by summarizing all stacks into a single stack *tensor*, denoted by $\mathbf{S}_h$, for which $\mathbf{S}_h[i, p]$ denotes p-th element (LSTM state) on the stack for i-th sentence.

The core idea behind achieving PUSH and POP jointly is that we perform LSTM updates for *all* stack top elements in a batch, but *only* proceed top stack pointers for PUSH batches. Given next actions $\mathbf{a}$ =[PUSH, PUSH, POP, $\cdots$] of length B, we get a vector $\mathbf{op} = [+1, +1, -1, \cdots]$, denoting whether next stack pointer is +1 (PUSH) or -1 (POP). By keeping stack top pointer vector $\mathbf{p}_h$, each step can be batched as the following two operations:

$$\mathbf{S}_h[(0, \mathbf{p}_h[0]+1)\cdots(B-1, \mathbf{p}_h[B-1]+1)] \leftarrow$$
$$\text{LSTM}(\mathbf{S}_h[(0, \mathbf{p}_h[0])\cdots(B-1, \mathbf{p}_h[B-1])], \mathbf{E}_w),$$
$$\mathbf{p}_h \leftarrow \mathbf{p}_h + \mathbf{op},$$

in which $\mathbf{E}_w$ is the next token embeddings.

Unfortunately, this batching relies on a strong assumption about models that one action (PUSH) involves all operations (LSTM update and pointer move by $\mathbf{op}$). This is not the case for RNNGs, for which any action cannot be reduced to a subset of other actions, necessitating a different strategy.

3 Batched RNNGs

Our batching algorithm for RNNGs is built on the following two observations:

(a) For all $a_t \in \{\text{NT}, \text{GEN}, \text{REDUCE}\}$, the last step is common and corresponds to PUSH operation for stack LSTM above with newly created embeddings $\{\mathbf{e}_x, \mathbf{e}_w, \mathbf{e}_c\}$. This final step can be batched if we get all new embeddings as a single tensor $\mathbf{E}_{\text{next}}$ (with size of $(B, |\mathbf{e}|)$).

(b) Then, the main problem is reduced to getting $\mathbf{E}_{\text{next}}$ efficiently. This is possible by separately filling $\mathbf{E}_{\text{next}}$ for each action, using a few additional pointer vectors to keep track of a stack state for each sentence.

---

[3] A restricted model of an arc-eager system (Nivre, 2004), which just POPs when LEFT-ARC occurs, can be achieved with these operations. RIGHT-ARC is modeled by PUSH. Essentially, this stack LSTM can only models the right spine of a tree at each step.

Algorithm 1 One training step for batched RNNG

```
Input Next action vector at $\mathbf{a}$;
      index vector for each action: $\mathbf{i}_{\text{gen}}, \mathbf{i}_{\text{nt}}, \mathbf{i}_{\text{red}}$
 1: $\mathbf{E}_{\text{next}} \leftarrow$ new tensor of size $(B, |\mathbf{e}_x|)$
 2: $\mathbf{E}_{\text{next}}[\mathbf{i}_{\text{gen}}] \leftarrow$ word_emb($\mathbf{x}[(\mathbf{i}_{\text{gen}}, \mathbf{b}[\mathbf{i}_{\text{gen}}])]$)
 3: $\mathbf{b}[\mathbf{i}_{\text{gen}}] = \mathbf{b}[\mathbf{i}_{\text{gen}}] + 1$          ▷ Move to next word.
 4: $\mathbf{E}_{\text{next}}[\mathbf{i}_{\text{nt}}] \leftarrow$ nt_emb($\mathbf{a}[\mathbf{i}_{\text{nt}}]$)
 5: $\mathbf{p}_q[\mathbf{i}_{\text{nt}}] \leftarrow \mathbf{p}_q[\mathbf{i}_{\text{nt}}] + 1$
 6: $\mathbf{q}[(\mathbf{i}_{\text{nt}}, \mathbf{p}_q[\mathbf{i}_{\text{nt}}])] \leftarrow \mathbf{p}_h[\mathbf{i}_{\text{nt}}] + 1$   ▷ Keep new NT depth.
 7: $\mathbf{p}_{\text{prev\_nt}} \leftarrow \mathbf{q}[(\mathbf{i}_{\text{red}}, \mathbf{p}_q[\mathbf{i}_{\text{red}}])]$
 8: $\mathbf{E}_{\text{red}} \leftarrow$ gather_children($\mathbf{p}_{\text{prev\_nt}}, \mathbf{p}_h[\mathbf{i}_{\text{red}}], \mathbf{S}_e$)
 9: $\mathbf{E}_{\text{next}}[\mathbf{i}_{\text{red}}] \leftarrow$ BiLSTM($\mathbf{E}_{\text{red}}$)          ▷ Composition.
10: $\mathbf{p}_q[\mathbf{i}_{\text{red}}] \leftarrow \mathbf{p}_q[\mathbf{i}_{\text{red}}] - 1$     ▷ Forget about reduced nts.
11: $\mathbf{p}_h[\mathbf{i}_{\text{red}}] \leftarrow \mathbf{p}_{\text{prev\_nt}} - 1$
12: $\mathbf{p}_h \leftarrow \mathbf{p}_h + 1$
13: $\mathbf{S}_h[\mathbf{p}_h] \leftarrow$ LSTM($\mathbf{S}_h[\mathbf{p}_h - 1], \mathbf{E}_{\text{next}}$)
14: $\mathbf{S}_e[\mathbf{p}_h] \leftarrow \mathbf{E}_{\text{next}}$
```

To obtain $\mathbf{E}_{\text{next}}$, for NT and GEN, we just need to lookup embeddings for next words and nonterminal symbols. We need an additional effort to obtain multiple $\mathbf{e}_c$s at once. Assuming a stack *tensor* as in Ding and Koehn (2019), we wish to pop k elements, up to $\mathbf{e}_c$, for multiple stacks by a single operation. Now the stack top positions can be accessed by $\mathbf{p}_h$ (Section 2.2), which will be the end indices. To obtain the last open nonterminal positions across a batch, just keeping the last nonterminal positions is insufficient because there are multiple open nonterminals in general. The following matrix and vector allow this operation:

- $\mathbf{q}$: A matrix of size (B, D) given a predefined stack depth bound D. $\mathbf{q}[b, d]$ denotes the position of d-th nonterminal on the b-th stack.
- $\mathbf{p}_q$: A B-dimensional vector, pointing to the last index of $\mathbf{q}$ (similar to $\mathbf{p}_h$ for $\mathbf{S}_h$).

For example, by $\mathbf{q}[(0, \mathbf{p}_q[0]) \cdots (B-1, \mathbf{p}_q[B-1])]$, we can retrieve all the top open nonterminal positions in a batch. Note that for each $\mathbf{q}[b]$, the index beyond $\mathbf{p}_q[b]$ will not be accessed, so we can signify the remove of top nonterminals just by a decrement of $\mathbf{p}_q$ without updating $\mathbf{q}$.

We need a few additional tensors to achieve fully batched stack tensor operations. Figure 1 shows an example.

- $\mathbf{S}_h$: A tensor of size (B, D, L, H) when the stack LSTM has L layers with H hidden dimensions. The core of batched stack LSTMs.
- $\mathbf{S}_e$: A tensor of size $(B, D, |\mathbf{e}|)$, corresponding to S_e in non-batched models (Section 2.1).
- $\mathbf{b}$: A B-dimensional vector, keeping the next token index in the sentence.

Stack						Next action
$\vert_0$	(S $\vert_1$	So $\vert_2$	(NP	it	) $\vert_3$	(VP
$\vert_0$	(S $\vert_1$	(NP	he	) $\vert_2$	(VP $\vert_3$	said
$\vert_0$	(S $\vert_1$	(NP $\vert_2$	(NP $\vert_3$	A $\vert_4$	branch $\vert_5$	)

$$\mathbf{b} = [1,1,2] \qquad \mathbf{p}_h = [3,3,5] \qquad \mathbf{p}_q = [0,1,2] \qquad \mathbf{q} = \begin{bmatrix} \underline{1} & 3 & 0 & \cdots \\ 1 & \underline{3} & 0 & \cdots \\ 1 & 2 & \underline{3} & \cdots \end{bmatrix}$$

Figure 1: Example batched stack configuration. $x|_d$ means that x is at depth d. For example, "(NP it)" in the first sentence is closed so constitutes a single item on the stack. $\mathbf{p}_q$ points to the top positions of $\mathbf{q}$, which are underlined.

- $\mathbf{p}_h$: A B-dimensional vector, pointing to the top elements of $\mathbf{S}_h$.

Full algorithm Algorithm 1 describes operations in each step given next actions $\mathbf{a}$. Action index vector $\mathbf{i}_a$, keeps the indices of action a in $\mathbf{a}$; in Figure 1, $\mathbf{i}_{\text{gen}} = [1]$ and $\mathbf{i}_{\text{nt}} = [0]$. The operations are mainly categorized into filling $\mathbf{E}_{\text{next}}$ for each action (in red), pointer updates according to action definitions (3, 5, 6, 10, 12), and finally stack updates (13, 14), corresponding to the observed common operations (a).[4] *gather_children* is a function that returns a tensor summarizing reduced children node embeddings. Since the number of reduced children differs across batch, we implement this to return a padded tensor, using *gather* function in PyTorch.

Deviated from Ding and Koehn (2019), we *separately* perform each action, as indicated by the use of $\mathbf{i}_a$. This can be seen as a deficiency of our algorithm; however, this separation is necessary beyond very simple models, which are practically less attractive. Rather, our strategy can be applied to broader classes of structured neural models, including dependency parsing with composition, and we believe that our empirical success (Section 5) encourages further exploration of the presented strategy to various models.

How to set D? As in Ding and Koehn (2019), we need to specify the stack depth bound D for each batch. Increasing this value incurs more GPU memory. For training, we can precompute the minimum value of each sentence by simulating oracle transitions beforehand and use it. For inference, we fix $D = 100$, since we find that even for very long

---

[4] By $A[(\mathbf{x}, \mathbf{y})]$ for vectors $\mathbf{x}$ and $\mathbf{y}$, both with length l, we mean $A[(x[0], y[0]) \cdots (x[l-1], y[l-1])]$, corresponding to advanced indexing in PyTorch. We regard that $\mathbf{S}_h[0]$ is fixed by initial hidden vectors while $\mathbf{S}_e[0]$ is kept empty. $\mathbf{p}_h[b] = 0$ means that b-th stack is empty.

sentences (more than 150 words), the stack depth will never exceed 80 for English sentences.

A note on extra memory with stack tensors At first sight, our approach seems to suffer from the limitation in scalability due to fixed stack tensors ($\mathbf{S}_h$ and $\mathbf{S}_e$). The sizes of these tensors grow by model size, implying that we may not be able to employ a large batch size for a large model. In practice, however, this extra memory will not be a bottleneck in the total memory for training. This is because the main cause of required memory during training is rather a computational graph itself, which keeps all intermediate hidden states at each step. Our stack tensors can be seen as a "storage" to allow computing these intermediate values effectively with tensor operations. The extra memory for this storage is smaller than the total memory in a computational graph because the former depends on D while the latter depends on the total action length A, and $D \ll A$ in general.[5]

4 Other Improvements

Batched beam search For inference as a language model or as an incremental parser, RNNGs typically employ word-synchronous beam search (Stern et al., 2017; Hale et al., 2018), which is although known to be very slow (Crabbé et al., 2019) because it often requires large beam sizes, such as 100 or 1000, and operations are not batched. As a by-product of our batched training, we succeed at implementing fully batched beam search for RNNGs, excluding any *for* loops, by which we drastically improve the search speed (Section 5.3). This is possible by adding the "beam" dimension to all state tensors ($\mathbf{S}_h$, $\mathbf{q}$, etc.).

Subwords Given an increased amount of training data, the vocabulary size naturally increases. To suppress this effect, using subwords (Sennrich et al., 2016) now becomes a standard technique. We thus incorporate subword modeling into our RNNGs and employ it for our largest experiment in Section 6. Kuncoro et al. (2020) recently incorporate subwords in RNNGs, in which, each word is regarded as a new constituent with WORD label, e.g., (WORD cu| r| ry). This means that models always need to perform additional NT(WORD) and REDUCE for each *token*, even for unsegmented ones, e.g., (WORD I), greatly increasing the average action sequence length, which in turn affect the training time. In this work, we model subwords by a simpler method of just segmenting each token. For example, an NP looks like (NP Th| ai cu| r| ry). While Kuncoro et al. (2020) note that this simple modeling is less effective, our experiments suggest that this is a good enough strategy, considering the added computational costs with NT(WORD) actions.[6]

5 Evaluating Efficiency of Batching

The main focus of this section is a comparative evaluation of our PyTorch RNNG implementation with the existing DyNet implementation.[7] We show that: (1) with a large batch size training speed drastically improves, and models will tend to find better parameters (Section 5.2); and (2) our batched beam search hugely speeds up inference (Section 5.3).

5.1 Setting

While Penn Treebank (PTB; Marcus et al., 1993) has often been used to train RNNGs (Wilcox et al., 2019, 2020), it is too small and here we use a larger dataset of BLLIP corpus (Charniak et al., 2000), expecting that the effects of large batch size become clearer by this modestly sized dataset.

Preprocessing For preprocessing, we largely follow Hu et al. (2020), which also train an RNNG on this dataset. We partition the data according to their LG size, the largest training setting, amounting to 42 million tokens for training and 1,500 sentences for development. One difference we make is the handling of unknown tokens. We limit the vocabulary by the top frequent 50,000 word types in the training data. Hu et al. (2020) use all word types

---

[5] Our preliminary experiment suggests that our RNNG implementation can be scaled at least comparable model and data sizes to ELMo (Peters et al., 2018), a large-scale LSTM-based model, given a similar amount of computing resources. We examine the maximum allowable batch size for a model with 1,256 hidden dimensions, amounting to 94M parameters, which are comparable to ELMo (93M), and find that the batch size can be increased to 256, with the maximum action size in a batch of 16,000 (see Section 5.1) on a single V100 GPU (16GB). Transformer-level scalability (Devlin et al., 2019) would still be infeasible because of the RNNG's limited parallelism that is only on sentence-level, not token-level as in Transformers.

[6] We provide a pilot study about this method in Appendix B. Using BLLIP corpus and Penn Treebank, we explore the relationship between a suitable number of subword units and model sizes. The main result is that large subword units are effective for larger models, and also subword modeling almost always improves parsing accuracy.

[7] https://github.com/cpllab/rnng-incremental. This implementation supports word-synchronous beam search. For the training part, this is not implemented to use DyNet Autobatch so we modified it to enable that.

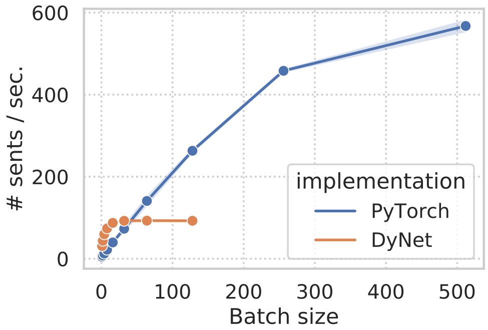

Figure 2: Training speed comparison (number of processed sentences / sec.) when increasing batch sizes to $[1, 2, 4, \cdots, 512]$. Shade denotes standard deviation.

that appear at least twice; however, this method vastly increases the vocabulary size and hence the model size. Unknown tokens are created in the same way with the Berkeley parser's surface feature rule (Petrov et al., 2006). The way to annotate parse trees is the same as well; we run Berkeley neural parser (Kitaev et al., 2019), a state-of-the-art constituency parser to assign accurate parses.

Model size and parameters We experiment with the most common model size of RNNG in the literature: 256 dimensions for input and LSTM hidden dimensions, with 2 layer LSTMs (Dyer et al., 2016; Hu et al., 2020). The total number of parameters is about 15M. The hyperparameters are summarized in Appendix A.

Other settings We employ some additional techniques to improve the efficiency of our batching mechanism. First, before training, we group sentences by their number of gold actions so that examples in each mini-batch have similar numbers of actions. Specifically, we first sort the sentences by action lengths, divide by every 4096 sentences, and then sample each batch from a single group.

Second, we predefine the maximum value for the total number of actions across sentences in a batch, which we set to 26,000. This is inspired by a similar mechanism in fairseq (Ott et al., 2019) for the maximum number of tokens. Using this means that the number of sentences in a batch will be adjusted to be smaller than the batch size when the action sequences (or sentences) are long, allowing us to interpret given batch size as the maximum that is fully exploited only for shorter sentences, which are in practice dominant in the data.[8]

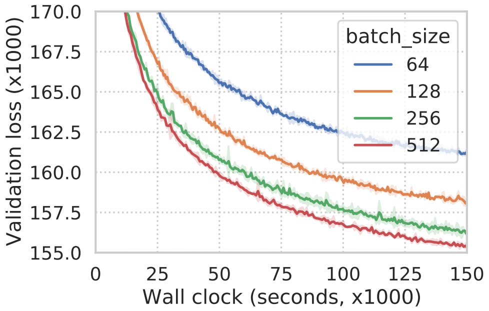

Figure 3: Training wall clock time vs. total validation loss for different batch sizes. X-value of i-th point is an averaged duration time to i-th validation step across three random seeds. Shade denotes standard deviation.

Every experiment is run on a single V100 GPU with 16GB memory. Unless otherwise noted, we perform every experiment three times with different random seeds, reporting an average score with standard deviation.

5.2 Effects of batch sizes

Although our batched training involves action-specific operations (Section 3), to our surprise, the efficiency improvement for our RNNG with large batch sizes is almost linear up to 256 (Figure 2). The improvement is narrow at 512, though this is mainly due to the restriction of the maximum number of actions in a batch (Section 5.1), which reduces the actual batch size for longer inputs. DyNet's Autobatch is quite effective up to 16, running much faster than ours due to the speed of C++, but further improvement is not obtained probably because of the increased overhead of finding mini-batch units themselves from a large computational graph.

Though this result clearly demonstrates the efficiency of our batching mechanism, it is *only* meaningful when the large batch size in fact leads to a faster model *convergence*. This is the case, as shown in Figure 3, where we compare the total validation losses as a function of actual wall clock time during training. The loss is calculated every 1000 batches. The model with batch size 512 converges fastest, and importantly, to better parameters. This result suggests that we can safely benefit from large batch size as long as memory permits. In the following experiments, we fix the batch size to 512.

---

[8] To reduce the memory further, we also employ mixed-precision training in PyTorch. Inference is performed with half-precision (fp16). We find that this does not change the results of beam search at all.

5.3 Beam search speed improvement

As we discuss in Section 4, we have also improved the efficiency of word-synchronous beam search, a standard technique to calculate incremental prefix probabilities (Hale, 2001) and a parse tree for RNNGs. Now, we evaluate the impact of this improvement. For PyTorch, we run it on V100 GPU; for DyNet, we find that it runs faster on CPUs so we instead use CPUs (Intel Xeon 6148, 20 cores x2), with Intel MKL. DyNet beam search is still too slow with this environment so we limit the number of tested sentences to 300 from the BLLIP development set. For PyTorch, we try two different batch sizes $\{1, 10\}$, with a restriction on the number of tokens in a batch, similarly to the total action size in training (Section 5.1). We fix this value to 250, with which the model can safely parse with the largest beam size of 1000.

Word-synchronous beam search employs two types of beam widths, action beam size (k) and word beam size (k_w), along with fast-tracked candidate size, denoted as k_s (see Stern et al. 2017). k is most akin to the standard beam size. Table 1 summarizes the results when increasing k (others are in the caption). The beam search of DyNet becomes prohibitively slow when $k \geq 50$. Strikingly, the increase in average runtime is more than linear against the beam size, especially for 10→50 and 50→100. The time increases, 0.5→11.3 (x22.6) and 11.3→48.6 (x4.3) are roughly quadratic to the increase of k (x5 and x2). This result is reasonable because in addition to the complexity of each step, which depends on k, the length of searched action sequence could also linearly grow by k.[9] The naïve DyNet implementation directly suffers from this computational cost.

Our batched beam search largely resolves this issue and now the average runtime only gradually increases by k. We note that as a parser or a language model, this speed is still not very fast, considering that this is on a GPU.[10] For the research purpose, including psycholinguistic assessments as done in Section 6, however, this improvement is significant, making experiments much easier even with large beam sizes. We still need to work on improving efficiency further, possibly by modifying learning methods to replace word-synchronous search (Stanojević and Steedman, 2020).

Action beam size k	10	50	100	200	400	1000
DyNet	0.5	11.3	48.6	100.6	201.3	NA
PyTorch (B=1)	1.7	2.1	2.3	2.5	2.9	4.1
PyTorch (B=10)	0.4	0.5	0.7	0.9	1.3	2.8

Table 1: Word-synchronous beam search speed (average seconds per sentence) comparison on the first 300 sentences in BLLIP development set. B denotes the batch size. Word beam sizes (k_w) / fast track sizes (k_s) are 10/1, 10/1, 10/1, 20/2, 40/4, and 100/10, respectively.

6 Syntactic Generalization Ability of Scaled RNNG

Finally, we evaluate the syntactic generalization abilities of the scaled RNNG. For this purpose, we adopt the test circuits used in Hu et al. (2020) via SyntaxGym (Gauthier et al., 2020). Here, a test *circuit* is a collection of test *suites*; e.g., "Long-Distance Dependencies" circuit contains a suite on a specific type of "filler-gap dependencies" as well as a suite on (pseudo) "cleft". For each example in a suite, a model succeeds if it can assign a higher likelihood on a grammatically critical position in the correct sentence. For example, given "*The farmer near the clerks knows/*know many people.*" in the "Agreement" circuit, a model is correct if it assigns $p(\textit{knows}|h) > p(\textit{know}|h)$. Note that for subword models the total likelihoods on subwords (not averaged) are compared.

In the previous literature, Hu et al. (2020) trained an RNNG on BLLIP (42M tokens). Here, we train subword RNNGs on 100M tokens from English Wikipedia, to which we assign parse trees with Berkeley neural parser. The model size is 35M with 30k subword units, following the experiment in Appendix B, which assesses the suitable number of subword units for different model sizes. We train this RNNG for three days (with three different seeds), and at inference fix the beam size k to 100 ($k_w = 10, k_s = 1$). We also train an RNNG with the subset of this data (42M tokens) to separate the effects of data size. Our LSTM baseline is the one used in Noji and Takamura (2020), which is basically an AWD-LSTM-LM (Merity et al., 2018)

---

[9] For a sentence of length N, the runtime of beam search is $O(k \times N \times M_w)$, where M_w denotes the maximum number of actions between two tokens (until choosing next SHIFT). The expected number of actions between two tokens (bound by M_w) grows by k because at each step, with a large k the chance that non-shift beam items remain in the next beam increases; hence, the runtime becomes quadratic to k in the worst case. We conjecture that this inefficiency is bounded at some k (see $k = 200$), though is severe for smaller ks.

[10] For smaller k, we can increase the batch size and the maximum number of tokens in a batch to further speedup.

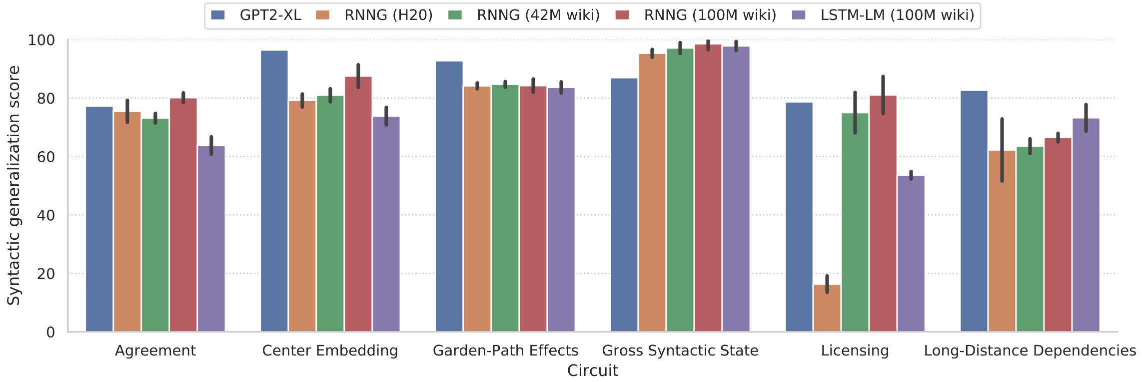

Figure 4: Circuit-level accuracies on SyntaxGym. For each circuit, suite-level accuracies are averaged across different test suites and random seeds to compute “Syntactic generalization score” of each model. Note that RNNG (H20) is a model trained on BLLIP (about 40M tokens) in Hu et al. (2020) but diverged from their results, because their suite-level accuracies are averaged across different models trained on various data sizes.

extended to be sentence-level and, on the Marvin and Linzen (2018) benchmark, shown to work better than GRNN (Gulordava et al., 2018), one of the models used in Hu et al. (2020).[11]

The main result on circuit-level accuracies is summarized in Figure 4. On the effects of the data scale, we observe a consistent improvement from “RNNG (42M wiki)“ to “RNNG (100M wiki)”. This result suggests that this amount of increase in training data is still beneficial for structural language models to strengthen their syntactic generalization ability. For some circuits, only RNNG (100M) outperforms GPT-2 (Radford et al., 2019) on average (“Agreement” and “Licensing”).

Comparing LSTM (100M) and RNNG (100M), RNNG generally outperforms LSTM, but with an exception on “Long-Distance Dependencies”. In order to inspect this, we break down this circuit into suites, (Figure 5), finding that this deficiency of RNNG is due to its poor performance on (pseudo) “cleft”, including the following example:

(1) a. What he did was prepare the meal .
b. *What he ate was prepare the meal .

On underlined tokens, models should assign a higher likelihood for (1a). Our LSTM performs nearly perfectly on these cases while our RNNGs perform badly. We conjecture that given more data and/or parameters, RNNGs will tend to strengthen their commitment to provided syntactic supervision, and hence may lose some lexical heuristics which LSTMs can exploit from surface patterns (e.g., an association of *did* → *prepare*). In fact, the ability of LSTM on cleft is rather brittle, as shown in a huge drop on “cleft_modifier”, which include cleft constructions with intervening modifiers.

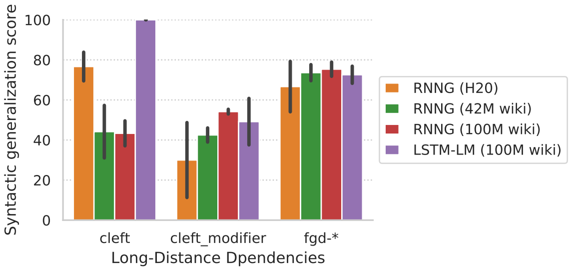

Figure 5: Suite-level accuracies on the “Long-Distance Dependencies” circuit. fgd-* is averaged across different test suites of filler-gap dependencies.

To rigorously handle these cases, models should notice that (1b) is a free relative clause and do not have an antecedent. However, the currently employed PTB annotation, which is limited to local structures, does not provide a distinction between these clause types, analyzing both as “(SBAR (WHNP What) (S (NP he) (VP did/ate)))”, which our RNNGs predict correctly. We also notice that Hu et al. (2020)’s RNNG (H20) performs rather similarly to our LSTM, while our RNNG (42M), trained on the comparable size of data to H20, is more similar to our RNNG (100M), suggesting that RNNG’s poor performance on cleft is not just due to the data scale. One possible explanation of the discrepancy between H20 and our 42M RNNG is

---

[11] Our LSTM implementation is available at https://github.com/aistairc/lm_syntax_negative. We train this LSTM on our subword-segmented Wikipedia (30k units). The model size is adjusted so that the total number of parameters becomes 35M, the same size as our RNNGs (3 layer LSTMs with 1150 hidden and 450 input dimensions).

that our RNNG might be better optimized thanks to improved training, or due to the sizes of hidden layers (256 for H20 and 656 for ours).

This problem poses a new interesting challenge. While RNNGs have been compared to LSTMs several times, the provided syntactic structures are fixed and effects of different annotations (formalism, quantity, etc.) are not explored. For such investigation, the training cost of RNNGs has been a practical burden, but that problem largely goes away with the current study. We expect that our new implementation and batching strategy provide fruitful future research opportunities on structured neural language models.

7 Conclusion

A large computational cost of training structured neural language models was a main practical burden for employing these models in applications and analyses. With special focus on RNNGs, we have provided a direct solution to this problem by showing that batched effective training is in fact possible. On the large scale experiments with SyntaxGym test circuits, we found that the data quantity further strengthens the syntactic generalization abilities of RNNGs, while the annotation quality or quantity will also be of practical importance towards a language model with human-like strong syntactic performance.

Acknowledgement

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work was also supported by JSPS KAKENHI Grant Numbers 20K19877 and 19H04990, and the National Institute for Japanese Language and Linguistics (NINJAL) Collaborative Research Project "Computational Psycholinguistics of Language Processing with Large Corpora."

References

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1 LDC2000T43. Linguistic Data Consortium.

Benoit Crabbé, Murielle Fabre, and Christophe Pallier. 2019. Variable beam search for generative neural parsing and its relevance for the analysis of neuro-imaging signal. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1150–1160, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shuoyang Ding and Philipp Koehn. 2019. Parallelizable stack long short-term memory. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, pages 1–6, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada. Association for Computational Linguistics.

Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. SyntaxGym: An online platform for targeted evaluation of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–76, Online. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736, Melbourne, Australia. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. 2019. Scalable syntax-aware language models using knowledge distillation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3472–3484, Florence, Italy. Association for Computational Linguistics.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Sabrina J. Mielke. 2019. Can you compare perplexity across different segmentations?

Sabrina J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4975–4989, Florence, Italy. Association for Computational Linguistics.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017. On-the-fly operation batching in dynamic computation graphs. In *Advances in Neural Information Processing Systems*, volume 30, pages 3971–3981. Curran Associates, Inc.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Hiroshi Noji and Hiroya Takamura. 2020. An analysis of the utility of explicit negative examples to improve the syntactic abilities of neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3375–3385, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Miloš Stanojević and Mark Steedman. 2020. Max-margin incremental CCG parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4111–4122, Online. Association for Computational Linguistics.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221, Brussels, Belgium. Association for Computational Linguistics.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3302–3312, Minneapolis, Minnesota. Association for Computational Linguistics.

Ethan Wilcox, Peng Qian, Richard Futrell, Ryosuke Kohita, Roger Levy, and Miguel Ballesteros. 2020. Structural supervision improves few-shot learning and syntactic generalization in neural language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4640–4652, Online. Association for Computational Linguistics.

A Hyperparameters

We use the defualt parameter setting for the DyNet implementation. For our implementation, we use Adam optimizer (Kingma and Ba, 2015), which is found to be superior, while SGD has been used for DyNet implementation (Dyer et al., 2016; Wilcox et al., 2019). We set the learning rate and dropout rate to 0.001 and 0.1, respectively, which we find achieve lower validation loss robustly across different batch sizes.

B Effect of Number of Subword Units

We perform an experiment to understand the behaviors of our simple subword modeling (Section 4). We use the BLLIP corpus as preprocessed in Section 5.1 except the setting about vocabulary. We compare the fixed vocabulary models, which we train in the experiment of Section 5.2 (batch size 512), and several subword vocabulary models. The hyperparameters are the same as the fixed vocabulary models.

Subword units The number of subwords can be seen as a hyperparameter. To understand the effects of this size for RNNGs, we prepare three different subword vocabularies: 10k (10,240), 20k (20,480), and 30k (30,720). We use byte-pair encoding (Sennrich et al., 2016) implemented in sentencepiece (Kudo and Richardson, 2018).

Model sizes We prepare two different model sizes, 15M and 35M, to see the interaction between the suitable size of subword units and model size, by adjusting the number of two dimensions so that the total number of parameters becomes comparable to these numbers. For 15M parameter models, the dimensions are 528 for 10k units; 432 for 20k; and 336 for 30k. For 35M parameter models, these are 864, 752, and 656, respectively. The number of LSTM layers is fixed to 2.

Results We investigate (1) the effectiveness of our simple subword modeling itself, and (2) whether the optimal number of subword units depends on model sizes. For (1), one way of evaluation is to compare the perplexities of subword models and fixed vocabulary models (see Section 5.1). However, they are not directly comparable because the fixed vocabulary models replace many tokens with unknown tokens, which are easy to predict and make the comparison unfair (Mielke et al., 2019).

V (# params.) / beam k	100	200	400	1000
$V_{fix} = 50k$ (15M)	92.34	93.70	94.26	94.59
$V_{sb} = 10k$ (15M)	92.74	93.85	94.37	94.64
$V_{sb} = 20k$ (15M)	**92.77**	**93.95**	**94.48**	**94.80**
$V_{sb} = 30k$ (15M)	92.33	93.76	94.38	94.72
$V_{sb} = 10k$ (35M)	92.67	93.87	94.37	94.59
$V_{sb} = 20k$ (35M)	92.84	**93.93**	**94.50**	**94.79**
$V_{sb} = 30k$ (35M)	**92.92**	93.84	94.44	94.72
Hale et al. (2018)	87.1	88.96	90.48	90.96

Table 2: PTB development set parsing accuracy (F1) when changing beam size, averaged on three models with different random seeds. V_{fix} is the vocabulary size for fixed vocabulary models while V_{sb} is that for subword models. Hale et al. (2018) is trained only on PTB training set and is not directly comparable. Word beam $k_w = k/10$ and $k_s = k/100$.

V (# params.) / beam k	100	200	400	1000
$V_{fix} = 50k$ (15M)	52.34	49.53	48.26	47.53
$V_{sb} = 10k$ (15M)	69.09	65.34	63.74	62.81
$V_{sb} = 20k$ (15M)	**67.52**	**64.15**	**62.41**	**61.36**
$V_{sb} = 30k$ (15M)	70.43	66.41	64.50	63.41
$V_{sb} = 10k$ (35M)	67.66	64.13	62.41	61.47
$V_{sb} = 20k$ (35M)	63.60	60.33	58.78	57.90
$V_{sb} = 30k$ (35M)	**60.80**	**57.91**	**56.28**	**55.45**

Table 3: Perplexity on BLLIP validation set for each setting described in Table 2, averaged on three models with different seeds. For V_{sb}, perplexity is not subword-level but token-level, by summing subword likelihoods for each token.

We instead validate the effectiveness of our subword modeling by not language modeling, but parsing performance. Note that the text in the BLLIP corpus is Wall Street Journal, the same as Penn Treebank (PTB). Thus, we expect that the quality of auto parses provided to our training data is high, allowing us to assume that a good model should parse the gold PTB data more accurately. We run beam search on the PTB development (section 22) for each model and the results are summarized in Table 2. We can see that F1 scores consistently improve by subword modeling compared to the fixed vocabulary setting. The effects of model size (15M vs. 35M) are negligible, suggesting that the upper bound parsing performance using the current silver quality data can be reached with smaller models.

For (2) above, while comparing perplexities across subword and fixed-vocabulary models is impossible, comparing different subword units is possible by casting the subword-level likelihoods to token-level likelihoods (Mielke, 2019). Table 3 summarizes those values along with the results by fixed vocabulary models as reference. The effects of the number of subword units (V_{sb}) are clearer. For 15M models, the optimal V_{sb} is $20k$, while for larger 35M models, the optimal size is $30k$. This suggests that more parameters are needed to obtain better results for large models.

Verb Sense Clustering using Contextualized Word Representations for Semantic Frame Induction

Kosuke Yamada[1] **Ryohei Sasano**[1,2] **Koichi Takeda**[1]
[1]Graduate School of Informatics, Nagoya University, Japan
[2]RIKEN Center for Advanced Intelligence Project, Japan
`yamada.kosuke@c.mbox.nagoya-u.ac.jp,`
`{sasano,takedasu}@i.nagoya-u.ac.jp`

Abstract

Contextualized word representations have proven useful for various natural language processing tasks. However, it remains unclear to what extent these representations can cover hand-coded semantic information such as semantic frames, which specify the semantic role of the arguments associated with a predicate. In this paper, we focus on verbs that evoke different frames depending on the context, and we investigate how well contextualized word representations can recognize the difference of frames that the same verb evokes. We also explore which types of representation are suitable for semantic frame induction. In our experiments, we compare seven different contextualized word representations for two English frame-semantic resources, FrameNet and PropBank. We demonstrate that several contextualized word representations, especially BERT and its variants, are considerably informative for semantic frame induction. Furthermore, we examine the extent to which the contextualized representation of a verb can estimate the number of frames that the verb can evoke.

1 Introduction

Contextualized word representations such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) are known to be effective in many natural language processing tasks such as question answering, natural language inference, and semantic textual similarity. Contextualized word representations can generate different representations of the same word in different contexts and distinguish the polysemy of a word. It has been reported that this property is effective in word sense disambiguation (WSD) (Hadiwinoto et al., 2019) and word sense induction (WSI) (Amrami and Goldberg, 2018). Therefore, it appears that contextualized word representations can also be leveraged to induce semantic frames from a large corpus automatically.

A semantic frame is defined on the basis of the semantic roles that a predicate can take as its arguments. FrameNet[1] (Baker et al., 1998) and PropBank[2] (Palmer et al., 2005) are the two most well-known resources of semantic frames, both of which are manually compiled. These resources are used not only for semantic parsing (Yang and Mitchell, 2017) but also for information extraction (Gangemi et al., 2016), question answering (Shen and Lapata, 2007), and document summarization (Cheung and Penn, 2013).

These frame-semantic resources define the frame and semantic roles, and they provide example sentences in which they are annotated. For example, the verb "support" in FrameNet is defined to evoke two frames: the SUPPORTING frame and the EVIDENCE frame. Sentences (1) and (2) below are examples where these frames are annotated. In Sentence (1), "support" means 'supporting a person or a thing' and evokes the SUPPORTING frame. Its arguments are annotated with the semantic roles of Supporter and Supported. In Sentence (2), "support" means 'corroborating' and evokes the EVIDENCE frame. Its arguments are annotated with the semantic roles of Proposition and Support. In both examples, the frame-evoking word is "support," but its evoking frames are different.

(1) [$_{\text{Supported}}$ This study] is supported by [$_{\text{Supporter}}$ the fund]. (SUPPORTING)

(2) [$_{\text{Support}}$ Our results] support [$_{\text{Proposition}}$ the hypothesis]. (EVIDENCE)

Since the manual development of such broad-coverage frame-semantic resources is labor-intensive and time-consuming, many researchers have attempted to induce semantic frames from

---

[1] `https://framenet.icsi.berkeley.edu/`

[2] `https://propbank.github.io/`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4353–4362
August 1–6, 2021.

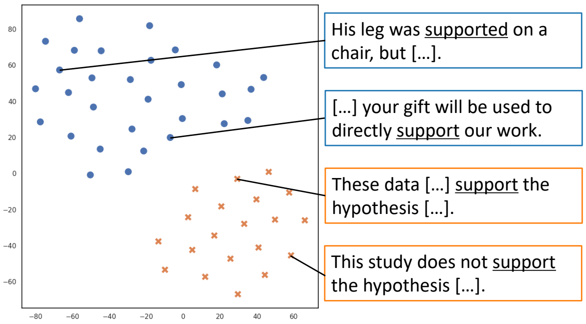

His leg was <u>supported</u> on a chair, but […].

[…] your gift will be used to directly <u>support</u> our work.

These data […] <u>support</u> the hypothesis […].

This study does not <u>support</u> the hypothesis […].

Figure 1: 2D t-SNE projection of BERT representations of verb “support” in FrameNet. ● and × correspond to example sentences from SUPPORTING and EVIDENCE frames, respectively.

large corpora automatically. For example, Kawahara et al. (2014) extracted predicate-argument structures of each verb from large corpora and induced the frames that each verb evokes by clustering the extracted predicate-argument structures. Several researchers have recently proposed frame induction methods that leverage word vector representations. For example, Ustalov et al. (2018) collected subject-verb-object triples from a Web-scale corpus and induced the frames by clustering based on the concatenation of word vector representations of the triples. However, since these approaches first collect the tuples of a verb and its arguments and then perform the clustering based on their word representations without taking their contexts into account, they may fail to disambiguate the word senses that require contextual clues.

Therefore, we seek a frame induction method that makes better use of contextual information by leveraging contextualized word representations. Figure 1 shows a 2D projection of contextualized representations of the verb “support” in different sentences. We extracted example sentences of “support” from the frame-annotated sentences in FrameNet, acquired contextualized representations of the verbs by applying a pre-trained BERT, and then projected them into two dimensions by using t-distributed stochastic neighbor embedding (t-SNE) (Maaten and Hinton, 2008). As shown in the figure, these BERT representations are distributed separately depending on the frame that “support” evokes in each example.

Our objective is to exploit this property of contextualized word representations for semantic frame induction. As a first step, we investigate how well contextualized word representations can distinguish the frames that the same verb evokes and which type of representations are suitable for semantic frame induction. We also need to estimate the number of frames that a verb evokes to build a frame-semantic resource automatically. We clarify to what extent contextualized word representations of verbs can estimate the number of frames that verbs evoke, which are defined manually. Our investigation of contextualized word representations will help construct high-quality frame-semantic resources not only for high-resource languages and general domains but also for low-resource languages and specific domains.

2 Related Work

2.1 Contextualized Word Representations

Contextualized word representations encode semantic and syntactic information by learning linguistic patterns and constraints from a large amount of text and provide significant improvements to the state of the art for a wide range of natural language processing tasks. They are also widely applied as context-sensitive word representation extractors for summarization (Liu, 2019), neural machine translation (Zhu et al., 2019), and so on.

Recently, several contextualized word representations have been proposed. For example, ELMo (Peters et al., 2018) produces contextualized word representations by pre-training on a bidirectional language model task in 2-layer BiLSTMs. More recently, many Transformer-based (Vaswani et al., 2017) models have been proposed. BERT (Devlin et al., 2019) utilizes multilayer bidirectional Transformers and is pre-trained on two tasks: masked language modeling and next sentence prediction. RoBERTa (Liu et al., 2019) redesigns the pre-training conditions for BERT, and ALBERT (Lan et al., 2020) shares each layer's parameters in BERT to reduce the number of parameters. There are other models such as GPT-2 (Radford et al., 2019), which is a unidirectional model that is trained to predict the next word in a sentence, and XLNet (Yang et al., 2019), which is based on a permutation language model that learns a bidirectional context in an autoregressive manner.

2.2 Semantic Frame Induction

For semantic frame induction of a word in a context, it is a standard approach to extract predicate-argument structures and then perform the clustering of those structures. LDA-frames (Materna, 2012) is an approach that represents frames as tuples of subject and object and uses latent Dirichlet allocation (LDA) (Blei et al., 2003) to induce semantic frames. Kawahara et al. (2014) extracted predicate-arguments structures from a large Web corpus and then applied the Chinese restaurant process clustering-algorithm (Aldous, 1985) to group predicates with similar arguments. Ustalov et al. (2018) proposed the Triclustering, which produces subject-verb-object triples and then performs a graph-based clustering using the concatenations of their static word embeddings. These methods take only the predicates and their arguments into account, and they do not sufficiently consider the context.

In some works, contextualized word representations are already used for semantic frame induction. In a shared task at SemEval 2019 (QasemiZadeh et al., 2019), some researchers worked on an unsupervised semantic frame induction task, and they reported that ELMo and BERT were useful for the task. Arefyev et al. (2019) first performed group average clustering by using contextualized word embeddings of target verbs from BERT. Then, they performed clustering to split each cluster into two by using TF-IDF features with paraphrased words by using BERT. Anwar et al. (2019) used a concatenated representation of a target verb and the average word embedding of all words in a sentence obtained by skip-gram (Mikolov et al., 2013) or ELMo. They performed group average clustering based on Manhattan distance by using the embedding. Ribeiro et al. (2019) performed graph clustering based on Chinese whispers (Biemann, 2006) by using contextualized representations of frame-evoking verbs from ELMo or BERT.

The shared task dataset contains many example sentences in which different verbs evoke the same frame, and thus the dataset is suitable for evaluating semantic frame induction over verbs. However, there are few example sentences of verbs that evoke different frames in the dataset, and it is not ideal for analyzing the difference of frames that each verb evokes. Some researchers assumed that many verbs evoke only one frame, and they did not analyze the difference of frames that each verb evokes.

Also, there is a study that works on semantic frame induction by using contextualized word representations in semi-supervised learning. Yong and Timponi Torrent (2020) used ELMo or BERT and mapped high-dimensional representations of verbs to a low-dimensional latent space for better frame prediction. Their study aims to extend FrameNet. On the other hand, our goal is to build frame-semantic resources automatically in an unsupervised fashion.

2.3 Word Sense Disambiguation with Contextualized Word Representation

The task in this paper is to distinguish the difference of frames that the same verb evokes, and as such, can be regarded as a type of word sense disambiguation (WSD) task. For the WSD task, contextualized word representations have been reported to be useful. For example, Peters et al. (2018) performed the task by nearest neighbor matching with ELMo representations, and Hadiwinoto et al. (2019) used pre-trained BERT contextualized representations as features for WSD. While WSD aims to distinguish between the meanings of words on the same surface, the semantic frame induction we focus on aims to distinguish between intuitive concepts such as situations, objects, and events that words evoke.

3 Methodology

We investigate to what extent contextualized word representations recognize the difference of frames that the same verb evokes. Specifically, we focus on verbs that evoke more than one frame in frame-semantic resources and acquire contextualized word representations of them. We then apply clustering and evaluate how well the generated clusters and human-annotated frames match.

3.1 Frame-semantic Resources

We use FrameNet and PropBank in English as frame-semantic resources. Since our goal is to establish a semantic frame induction method that is not in a particular style, we use two well-known frame knowledge resources for our investigation: Berkeley FrameNet data release 1.7[3] and PropBank-annotated data from OntoNotes v5.0.[4]

FrameNet is developed within the framework of the theory of frame semantics proposed by Fillmore (2006). Each frame is shared by multiple frame-evoking words (lexical units), and hierarchical relations such as “Inheritance” or “Using” are defined between closely related frames. FrameNet has 1,222 frames, 13,572 lexical units, and 200,751 annotated sentences. The corpus consists of the British National Corpus and U.S. newswire texts.

---

[3] `https://framenet.icsi.berkeley.edu/fndrupal/framenet_data`

[4] `https://catalog.ldc.upenn.edu/LDC2013T19`

PropBank is developed as a corpus with semantic roles that can be used as training data in supervised learning. PropBank frames are defined for each verb as a frameset containing semantic role labels. There are two types of labels; one is ARG0-5. It indicates a necessary role and has a different meaning in each frameset. The other is an argument modifier (AM) label, which indicates an additional role common to all framesets (e.g., AM-TMP for time). For example, the frameset SUPPORT.01 (lend aid, credence to) of "support" is defined with ARG0 as 'helper' and ARG1 as 'person, thing or project being supported.' Sentence (3) is an example in which this frameset is annotated.

(3) $[_{\text{ARG1}}$ Students$]$ were $\underline{\text{supported}}_{.01}$ by $[_{\text{ARG0}}$ the scholarship$]$ $[_{\text{AM-TMP}}$ for four years$]$.

Unlike FrameNet, hierarchical relations are not defined between framesets; that is, each frameset is independent. PropBank has 5,607 framesets, 4,221 verbs, and 111,178 annotated sentences. The corpus consists of newswires, magazine articles, broadcast news, broadcast conversations, web data, conversational speech data, and pivot text.

3.2 Procedure

In our investigation, we follow the procedures below for each target verb that evokes more than one frame in the frame-semantic resources.

1. Acquire contextualized word representations of the target verbs in the set of frame-annotated example sentences in the frame-semantic resources.[5]
2. Apply clustering to their contextualized word representations by using a Gaussian mixture model. At this time, the number of clusters given to the model is equal to the number of frames in our dataset.
3. Find a mapping between the generated clusters and the human-annotated frames that maximize the overall number of matches. We use the match rate as the evaluation metrics.

4 Experiment on Frame Distinction

4.1 Dataset

We first determined the target verbs, and we then extracted example sentences of the target verbs from both FrameNet and PropBank. As target verbs, we used verbs that evoke two or more frames with at least 20 annotated sentences. For example, in FrameNet, the verb "support" is a target verb because there are 30 sentences in the SUPPORTING frame and 20 sentences in the EVIDENCE frame. In contrast, the verb "attend" is not a target verb. This is because although the verb "attend" evokes three frames, (ATTENTION, PERCEPTION_ACTIVE, and ATTENDING), there are 7 sentences in the ATTENTION frame, 4 sentences in the PERCEPTION_ACTIVE frame, and 24 sentences in the ATTENDING frame and only the ATTENDING frame includes 20 or more sentences.

For each verb, we considered frames that include at least 20 annotated sentences. In addition, if the target verb evokes more than 10 frames with 20 or more annotated sentences, we used the top 10 frames on the basis of the number of annotated sentences. We used a maximum of 100 annotated sentences for each frame. As a result, we have obtained 178 target verbs for FrameNet and 164 for PropBank. The average counts of frames per verb were 2.21 for FrameNet and 2.73 for PropBank, and the average counts of annotated sentences per frame were 41.68 for FrameNet and 70.34 for PropBank. In this paper, we used 120 verbs as the test set for the final evaluation and the remaining verbs as the development set for tuning the parameters for both FrameNet and PropBank.

4.2 Settings

We compared ELMo, $\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{LARGE}}$, RoBERTa, ALBERT, GPT-2, and XLNet as contextualized word representations in order to explore the representation most suitable for semantic frame induction. We used publicly available pre-trained models. ELMo is the 'Original' model in AllenNLP,[6] and the other transformer-based models are pre-trained models[7] in Hugging Face.[8] For each model, we obtained contextualized word representations from the hidden layer that achieved the highest scores in the development sets for FrameNet and PropBank, respectively. Table 1 lists the size of the corpus used to pre-train models and the number of parameters, dimensions, layers, and hidden layers of models used to obtain the representations

---

[5] Tokenization is performed in the same way as used in the pre-training. If tokenization splits the target verb token into more than one sub-token, we use the contextualized word representations of the first sub-token.

[6] `https://allennlp.org/elmo`

[7] These models are specified by 'bert-base-uncased,' 'bert-large-uncased,' 'roberta-base,' 'albert-base-v2,' 'gpt2,' and 'xlnet-base-cased' in Hugging Face.

[8] `https://huggingface.co/transformers/`

Model	Corpus size	No. of parameters	No. of dimensions	No. of layers	Used layer
ELMo	11GB	94M	512	2	1|1
BERT$_{\text{BASE}}$	16GB	110M	768	12	9|7
BERT$_{\text{LARGE}}$	16GB	340M	1024	24	21|15
RoBERTa	160GB	125M	768	12	10|6
ALBERT	16GB	11M	768	12	9|8
GPT-2	40GB	117M	768	12	8|9
XLNet	158GB	110M	768	12	5|5

Table 1: Details of contextualized word representations. “Used layer” means hidden layer of model used to obtain the representations in FrameNet (left) and PropBank (right).

Model	FrameNet	PropBank
All-in-one-cluster	0.578	0.548
ELMo	0.631	0.607
BERT$_{\text{BASE}}$	0.750	0.765
BERT$_{\text{LARGE}}$	**0.769**	0.790
RoBERTa	0.767	**0.796**
ALBERT	0.705	0.712
GPT-2	0.666	0.650
XLNet	0.729	0.758

Table 2: Macro-average match rate of each verb for each of models and datasets.

for FrameNet and PropBank, respectively.

We used the Gaussian mixture model implementation provided by scikit-learn.[9] We adopted “spherical” as the covariance type, that is, the covariance matrix was a diagonal covariance with equal elements along the diagonal. We used five trials of clustering with different random seeds and adopted the result of the highest likelihood trial.

4.3 Results

Table 2 lists the macro-average match rate of each verb for each of the models and datasets. All-in-one-cluster means the average score when all the example sentences were in one cluster for each verb. That is, the score is the average of the percentages of examples that were annotated with the most frequently used frame for a verb. For example, the score of the verb “support” in FrameNet was 0.6 (30/50) since the numbers of example sentences from the Supporting frame and the Evidence frame were 30 and 20, respectively.

As shown in Table 2, BERT$_{\text{LARGE}}$ and RoBERTa achieved the highest scores for FrameNet and PropBank, respectively. We confirmed that they recognized the differences of frames that the same verbs evoke. BERT$_{\text{BASE}}$, XLNet, and ALBERT also achieved high scores. These results indicate that BERT, RoBERTa, XLNet, and ALBERT are useful for semantic frame induction. In contrast, the scores obtained for ELMo and GPT-2 were relatively low and almost the same as for the All-in-one-cluster. It indicates that the degree of the difference of frames captured by the contextualized word representations varied greatly.

The reasons for these results are described below. The high scoring BERT, RoBERTa, XLNet, and ALBERT are deep bidirectional language models based on Transformer. In contrast, GPT-2 is a unidirectional language model based on Transformer. Also, ELMo is a relatively sparse bidirectional language model that consists of only two unidirectional contexts concatenated together. Therefore, the scores of GPT-2 and ELMo were lower than those of the deep bidirectional language models.

We show several examples below. In these figures, the number given to each point represents the clustering result; that is, the points with the same number belong to the same cluster. Note that the value of the number has no meaning. Figure 2 shows a 2D t-SNE projection of BERT$_{\text{LARGE}}$ vectors for “support” in FrameNet. We can see that the example sentences from the Supporting frame and the Evidence frame form a cluster, respectively.

Figure 3 shows a 2D t-SNE projection of BERT$_{\text{LARGE}}$ vectors for “fire” in FrameNet. We can see that example sentences from the Firing frame form a single cluster, whereas the difference between the Shoot_projectiles frame and the Use_firearm frame is not captured. The Firing frame means ‘ending an employment relationship’ while the Shoot_projectiles and the Use_firearm frames mean ‘shooting a bullet’ and ‘shooting a gun’, respectively. The Firing frame is very different from the other two. On the other hand, the “Using” relation is annotated between the Shoot_projectiles frame and Use_firearm frame, which indicates that there is a strong connection between the two frames. We conduct an additional analysis on frames that have hierarchical

---

[9] `https://scikit-learn.org`

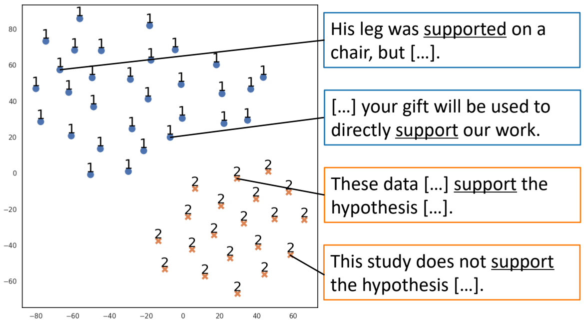
His leg was supported on a chair, but […].

[…] your gift will be used to directly support our work.

These data […] support the hypothesis […].

This study does not support the hypothesis […].

Figure 2: 2D t-SNE projection of $\text{BERT}_{\text{LARGE}}$ vectors for verb "support" in FrameNet. ● and × correspond to example sentences from Supporting and Evidence frames, respectively.

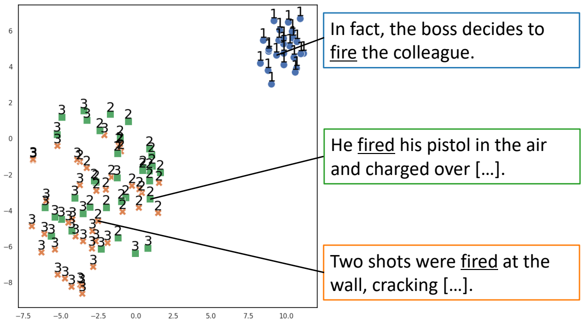
In fact, the boss decides to fire the colleague.

He fired his pistol in the air and charged over […].

Two shots were fired at the wall, cracking […].

Figure 3: 2D t-SNE projection of $\text{BERT}_{\text{LARGE}}$ vectors for verb "fire" in FrameNet. ●, ×, and ■ correspond to example sentences from Firing, Shoot_projectiles, and Use_firearm frames, respectively.

relations in Section 4.4.

Figure 4 shows a 2D t-SNE projection of $\text{BERT}_{\text{LARGE}}$ vectors for "work" in PropBank. The verb "work" has four types of framesets: WORK.01 (work), WORK.02 (arrange), WORK.03 (exercise), and WORK.09 (function, operate). We confirmed that $\text{BERT}_{\text{LARGE}}$ roughly captured the difference of frames, even for verbs that can have many framesets. In the examples where WORK.02 and WORK.03 were annotated, the verb "work" appears in the form of "work out," and it may have been a bit challenging to capture the difference of these framesets. This is because verbs that appear as part of phrasal verbs have relatively similar contextualized word embeddings since the same word appears near the verb.

Figure 5 shows a 2D t-SNE projection of $\text{BERT}_{\text{LARGE}}$ vectors for "cry" in PropBank. The verb "cry" has two types of framesets: CRY.01 (speak loudly, yell, demand, possibly while weeping) and CRY.02 (cry, weep). Like the verb "fire" in FrameNet, the resulting clusters could not be appropriately formed because the framesets of the verb "cry" are both related to 'weep' and are thus very similar.

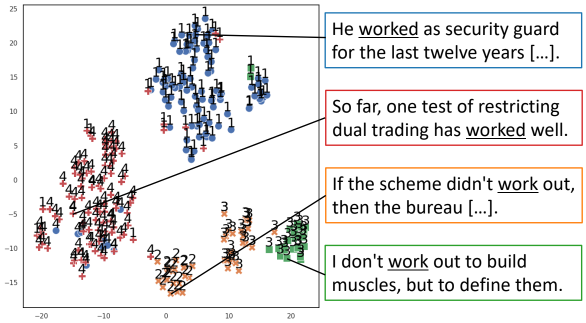
He worked as security guard for the last twelve years […].

So far, one test of restricting dual trading has worked well.

If the scheme didn't work out, then the bureau […].

I don't work out to build muscles, but to define them.

Figure 4: 2D t-SNE projection of $\text{BERT}_{\text{LARGE}}$ vectors for verb "work" in PropBank. ●, ×, ■, and + correspond to example sentences from WORK.01 (work), WORK.02 (arrange), WORK.03 (exercise), and WORK.09 (function, operate) framesets, respectively.

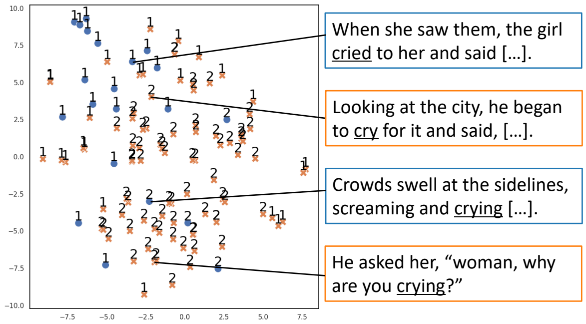
When she saw them, the girl cried to her and said […].

Looking at the city, he began to cry for it and said, […].

Crowds swell at the sidelines, screaming and crying […].

He asked her, "woman, why are you crying?"

Figure 5: 2D t-SNE projection of $\text{BERT}_{\text{LARGE}}$ vectors for verb "cry" in PropBank. ● and × correspond to example sentences from CRY.01 (speak loudly, yell, demand, possibly while weeping) and CRY.02 (cry, weep) framesets, respectively.

4.4 Effect of Hierarchical Relations on Evaluation

The frames with hierarchical relations defined in FrameNet appear in similar contexts. As is clear from the examples of "fire," it is not easy to distinguish these frames, even using contextual word representations. Moreover, it is unclear whether these frames should be defined as separate frames if semantic frame resources are to be automatically constructed in the future. Specifically, the importance of distinguishing between the Shoot_projectiles frame and the Use_firearm frame could be less important than distinguishing between the Shoot_projectiles frame and the Firing frame.

To investigate the practical usefulness, we attempted to evaluate the accuracy of the distinc-

Model	Gr w/o rel	Gr w/ rel	Diff
All-in-one-cluster	0.613	0.604	0.009
ELMo	0.680	0.644	0.036
BERT$_{\mathrm{BASE}}$	0.805	0.719	0.086
BERT$_{\mathrm{LARGE}}$	0.826	0.713	0.113
RoBERTa	0.836	0.707	0.129
ALBERT	0.758	0.670	0.088
GPT-2	0.724	0.655	0.069
XLNet	0.783	0.700	0.083

Table 3: Average match rate by groups without hierarchical relations (Gr w/o rel) and groups with hierarchical relations (Gr w/ rel). "Diff" represents difference between score of Gr w/o rel and score of Gr w/ rel.

tion between frames with hierarchical relations and frames without relations, separately. We first extracted verbs that had exactly two types of frames from FrameNet as a result of the procedure described in Section 4.1. We then divided the extracted verbs into two groups according to whether there is a hierarchical relation between the two frames or not and calculated the average match rate for each group.[10] By limiting our focus to verbs with two types of frames, we can ignore the tendency of the match rate to decrease as the number of frames increases. We assume that if a certain contextualized word representation appropriately captures the difference of frames, it should be able to distinguish the difference of frames with a high match rate.

Table 3 lists the results of the average match rate. In the models of BERT$_{\mathrm{LARGE}}$, RoBERTa, BERT$_{\mathrm{BASE}}$, XLNet, and ALBERT, which obtained relatively high scores in the results shown in Table 2, we can see that the group without relations got higher scores than the group with relations. It is arguably concluded this result indicates that these models accurately captured the essential difference of frames.

5 Estimation of Number of Frames

In the experiments in Section 4, we gave the number of frames in our dataset to the Gaussian mixture model. However, it is necessary to estimate the number of frames that each verb evokes for semantic frame induction. Therefore, we investigated how well we can estimate the number of frames on the basis of information criteria by using contextualized word representations. Specifically, we adopted a Bayesian information criterion (BIC) (Schwarz, 1978), which is used for determining the number of clusters, and an adjusted-BIC, in which the BIC is adjusted so that the estimated number of clusters is close to the number of human-annotated frames.

5.1 Information Criterion

The Bayesian information criterion (BIC) is one of the most widely used criterion for model selection. The BIC is defined as

$$\mathrm{BIC} = -2\ln(L) + k \cdot \ln(n_s), \qquad (1)$$

where L is the likelihood of the model, n_s is the number of samples, and k is the number of model parameters. The parameters of the Gaussian mixture model consist of the mean, covariance, and mixture weights. When the numbers of clusters and dimensions are represented by n_c and d, respectively, the number of parameters required to represent the mean is $d \times n_c$. Since we adopted spherical as the covariance type, where each component has its own single variance, the number of parameters required to represent the covariance is n_c. Since the mixture weights for each component are probabilities that sum to 1, the number of parameters required to represent the mixture weight is $n_c - 1$. Thus, the total number of model parameters is $k = (d + 2) \times n_c - 1$. When the BIC is used to determine the number of clusters, the number that minimizes the BIC is selected.

The first term on the right-hand side in Equation 1 decreases as the number of clusters increases because the likelihood of an optimized model generally increases as the number of parameters increases. The second term on the right-hand side is regarded as a penalty term that inhibits the increase in the number of clusters. The granularity of frames decided by human intuition may not be optimal in terms of the information criterion. Therefore, we introduce an adjusted-BIC in which the penalty term of the BIC is adjusted so that the granularity of the frames is close to human intuition. The equation of the adjusted-BIC (a-BIC) is defined as

$$\text{a-BIC} = -2\ln(L) + c \cdot k \cdot \ln(n_s), \qquad (2)$$

where c is a constant that adjusts the penalty, which is decided by using the development set.[11]

---

[10] Among the 120 verbs that were used as the test set in the experiment, the number of verbs that evoked only two frames was 96. The number of verbs in the group without the relations was 62, while the number of verbs in the group with the relations was 34.

[11] The constant c is increased from 1 in increments of 0.1, and we decide the value when the total number of frames and the total estimated number of clusters are as close as possible in the development set.

		ρ	Accuracy	RMSE
FrameNet	BIC	0.165	0.058	3.131
	a-BIC	0.177	0.517	1.195
PropBank	BIC	0.066	0.004	7.429
	a-BIC	0.631	0.608	1.008

Table 4: Evaluation on estimating the number of frames using BIC and a-BIC for FrameNet and PropBank.

5.2 Results

In the experiments in Section 4, we used only the verbs that evoke more than one type of frame. However, it is also essential for verbs that evoke only one type of frame to recognize that. Therefore, we added verbs that evoke only one frame. The number of verbs added was the same as the number of verbs used in the experiment in Section 4. We also used a maximum of 100 annotated sentences from each frame. As a result, we used 116 verbs for parameter tuning as the development set and 240 verbs for evaluation as the test set for FrameNet, and we used 88 verbs for parameter tuning as the development set and 240 verbs for evaluation as the test set for PropBank.

We used $\text{BERT}_{\text{LARGE}}$ as contextualization word representations. We evaluated the automatic estimation of the number of frames by using Spearman's rank correlation coefficient (ρ), accuracy, and root mean square error (RMSE) for the estimated number of clusters and the number of frames in our dataset. Table 4 lists the estimation results of the number of frames. For both FrameNet and PropBank, using the adjusted-BIC as the information criterion resulted in better scores than using the BIC. When using the adjusted-BIC, Spearman's rank correlation coefficients were 0.177 and 0.631 for FrameNet and PropBank, respectively. The accuracy scores were over 0.5, which means that we could correctly predict the number of frames for more than half of the verbs. The accuracy for FrameNet is lower than the accuracy for PropBank. Accurate prediction of the number of frames for FrameNet will need to consider semantic coherence across different verbs, since frames in FrameNet are not defined independently for each verb.

Table 5 shows the confusion matrices between the number of human-annotated frames and the estimated number of frames using the BIC and the adjusted-BIC for FrameNet and PropBank. We can see that the BIC tended to overestimate the number of frames. The constant c was tuned at 3.1 for FrameNet and 3.4 for PropBank.

	1	2	3	4	5+
1	1	16	32	19	52
2	0	11	14	19	52
3	0	1	1	5	13
4	0	0	0	1	3

BIC for FrameNet

	1	2	3	4	5+
1	0	0	1	10	109
2	0	1	1	2	81
3	0	0	0	1	11
4+	0	0	0	0	23

BIC for PropBank

	1	2	3	4	5+
1	88	17	7	4	4
2	50	35	5	3	3
3	10	10	0	0	0
4	2	1	0	1	0

a-BIC for FrameNet

	1	2	3	4	5+
1	80	35	3	0	2
2	15	57	12	0	1
3	1	2	7	1	1
4+	0	6	7	3	7

a-BIC for PropBank

Table 5: Confusion matrices using BIC and a-BIC for FrameNet and PropBank. Vertical axis represents the number of frames in our dataset, and horizontal axis represents the estimated number of clusters.

6 Conclusion and Future Work

We investigated to what extent contextualized word representations can recognize the difference of frames that the same verb evokes. Specifically, we focused on verbs that evoke multiple frames and performed clustering based on contextualized word representations of target verbs. We calculated the match rate between the generated clusters and the human-annotated frames and compared seven contextualized word representations: ELMo, $\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{LARGE}}$, RoBERTa, ALBERT, GPT-2, and XLNet. We found that BERT, RoBERTa, XLNet, and ALBERT achieved high performance in distinguishing the difference of frames that the same verb evokes. We also found that we can estimate the number of frames with an accuracy of more than 50% by using the adjusted-BIC, which adjusts the penalty term of the BIC.

In this paper, we focused on the difference of frames that each verb evokes. That is, we analyzed each verb separately. However, in FrameNet, frames are shared by several verbs. For example, the verbs "support," "prove," and "demonstrate" can evoke the same EVIDENCE frame. To induce FrameNet-style frames, we need to investigate to what extent contextualized word representations capture frames over verbs.

Semantic frame induction requires not only distinguishing the difference of frames that the same verb evokes but also grouping its arguments by the semantic role. For example, if a sentence contains a verb that evokes the EVIDENCE frame, the sentence contains what is claimed and what supports the claim as its argument. Contextualized word representations of the arguments will also be useful for grouping arguments by the same roles.

Furthermore, we only considered verbs as frame-evoking words, but we need to examine whether we can obtain similar results for words with other parts of speech that evoke frames such as nouns. These investigations are expected to bring us closer to our goal of automatically constructing high-quality semantic-frame resources. They can also induce semantic frames for under-resourced languages or specific domains since contextualized word representations do not require human-annotated texts.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers 18H03286 and 21K12012.

References

David J Aldous. 1985. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198.

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural biLM and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4860–4867.

Saba Anwar, Dmitry Ustalov, Nikolay Arefyev, Simone Paolo Ponzetto, Chris Biemann, and Alexander Panchenko. 2019. HHMM at SemEval-2019 task 2: Unsupervised frame induction using contextualized word embeddings. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 125–129.

Nikolay Arefyev, Boris Sheludko, Adis Davletov, Dmitry Kharchev, Alex Nevidomsky, and Alexander Panchenko. 2019. Neural GRANNy at SemEval-2019 task 2: A combined approach for better modeling of semantic relationships in semantic frame induction. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 31–38.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL-COLING 1998)*, pages 86–90.

Chris Biemann. 2006. Chinese whispers-an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1233–1242.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 4171–4186.

Charles J Fillmore. 2006. Frame semantics. *Cognitive Linguistics: Basic Readings*, 34:373–400.

Aldo Gangemi, Diego Reforgiato Recupero, Misael Mongiovì, Andrea Giovanni Nuzzolese, and Valentina Presutti. 2016. Identifying motifs for evaluating open knowledge extraction on the web. *Knowledge-Based Systems*, 108:33–41.

Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. 2019. Improved word sense disambiguation using pre-trained contextualized word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pages 5297–5306.

Daisuke Kawahara, Daniel Peterson, Octavian Popescu, and Martha Palmer. 2014. Inducing example-based semantic frames from a massive amount of verb uses. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 58–67.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of the International Conference on Learning Representations (ICLR 2020)*.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Jiří Materna. 2012. Lda-frames: An unsupervised approach to generating semantic frames. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2012)*, pages 376–387.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, pages 2227–2237.

Behrang QasemiZadeh, Miriam R. L. Petruck, Regina Stodden, Laura Kallmeyer, and Marie Candito. 2019. SemEval-2019 task 2: Unsupervised lexical frame induction. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 16–30.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Eugénio Ribeiro, Vânia Mendonça, Ricardo Ribeiro, David Martins de Matos, Alberto Sardinha, Ana Lúcia Santos, and Luísa Coheur. 2019. L2F/INESC-ID at SemEval-2019 task 2: Unsupervised lexical semantic frame induction using contextualized word representations. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 130–136.

Gideon Schwarz. 1978. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL 2007)*, pages 12–21.

Dmitry Ustalov, Alexander Panchenko, Andrey Kutuzov, Chris Biemann, and Simone Paolo Ponzetto. 2018. Unsupervised semantic frame induction using triclustering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 55–62.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pages 5998–6008.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 1247–1256.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems (NeurIIPS 2019)*, pages 5754–5764.

Zheng Xin Yong and Tiago Timponi Torrent. 2020. Semi-supervised deep embedded clustering with anomaly detection for semantic frame induction. In *Proceedings of The 12th Language Resources and Evaluation Conference (LREC 2020)*, pages 3509–3519.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2019. Incorporating BERT into neural machine translation. In *Proceedings of the International Conference on Learning Representations (ICLR 2019)*.

Benchmarking Neural Topic Models: An Empirical Study

Thanh-Nam Doan
University of Tennessee at Chattanooga
Chattanooga, TN, USA
thanh-nam-doan@utc.edu

Tuan-Anh Hoang
VNU University of Science
334 Nguyen Trai, Hanoi, Vietnam
hoangtuananh@hus.edu.vn

Abstract

Neural topic modeling approach has been attracting much attention recently as it is able to leverage the advantages of both neural networks and probabilistic topic models. Previous works have proposed several models that are based on this framework and obtained impressive experimental results compared to traditional probabilistic models. However, the reported result is not consistent across the works, making them hard for gaining a rigorous assessment of these approaches. This work aims to address this issue by offering an extensive empirical evaluation of typical neural topic models in different aspects using large, diverse datasets as well as a thorough set of metrics. Precisely, we examine the performance of these models in three tasks, namely *uncovering cohesive topics*, *modeling the input documents*, and *representing them for downstream classification*. Our results show that while the neural topic models are better in the first and the third tasks, the traditional probabilistic models are still a strong baseline and are better in the second task in many cases. These findings give us more insights for choosing off-the-shelf topic modeling toolboxes in different contexts, as well as for designing more comprehensive evaluation for neural topic models.

1 Introduction

Classical topic modeling approach consists of statistical learning methods for uncovering the latent topics from a corpus and the semantic meaning of each document in the corpus. Notable works include the pioneering ones by (Hofmann, 1999; Blei et al., 2003). During its more than 20 years of research, topic modeling has also been applied to other fields beyond its original scope such as image analysis (Fei-Fei and Perona, 2005), and recommender systems (McAuley and Leskovec, 2013).

Recently, neural topic modeling approach has been attracting much research attention for understanding topic models of corpus due to its ability to leverage the advantages of both neural networks and probabilistic generative models. Compared to classical ones, this approach has three main advantages. First, its inference is amortized (Miao et al., 2016) and hence is much computationally simpler than that of the classical approach which requires to deal with a complicated optimization problem (Blei et al., 2003). Second, the gap between prototype and deployment processes become closer thanks to the power of some deep learning frameworks such as Pytorch (Paszke et al., 2019), Tensorflow (Abadi et al., 2016), and Flux.jl (Innes, 2018). Third, neural topic models are easy to be integrated with prior knowledge such as pre-trained word and text embeddings, which are prevalent and have shown the tremendous usefulness, e.g. GPT-3 (Brown et al., 2020), BERT (Devlin et al., 2019).

Several neural topic models are proposed recently to investigate the above advantages. Their reported experimental results are promising compared to the classical topic models. However, the reported results are not consistent or even contradict across these works. For example, for the same 20News dataset, (Miao et al., 2016) reports that their model i.e. NVDM obtains much better performance (measured by perplexity) than the classical LDA does, while the same experiments of (Srivastava and Sutton, 2017) show that LDA outperforms NVDM significantly. All these inconsistency and contradiction make it hard to assess the neural topic modeling approach comprehensively.

In this work, we would like to address this gap by conducting an extensive empirical study to accurately evaluate the typical existing neural topic models. Precisely, using a diverse set of large datasets, we examine the performance of the models in several tasks, including document modeling, topic discovery, and document representation for downstream classification. Our contributions are:

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4363–4368
August 1–6, 2021.

- To the best of our knowledge, our work is the first one which examines state-of-the-art neural topic models in a systematical mechanism.
- We also public our implementation [1] as an additional resource for better reproducibility and further improvement in topic modeling research.
- Based on the results of our extensive experiments, we provide some practical guidelines of using neural topic models for specific tasks.

2 Experiment Settings

In this section, we describe settings used in our experiments. We start by briefing the models chosen to examined. We then introduce the datasets and define the metrics used to examine the models.

2.1 Evaluated Models

Neural topic models. Existing neural topic models are generally based on encoder-decoder architectures. In those models, the encoders are some variants of variational auto-encoder (VAE) (Kingma and Welling, 2013) whose input is the bag-of-word vector of an input document. The decoders, on the other hand, are designed to recover the input document from the encoded vector. For a fair comparison with the classical topic models that work purely on bag-of-words representation of documents, we surveyed the state-of-the-art works on neural topic modeling and chose to examine the following models whose decoders also work with the same representation of the documents.

- **NVDM** (Miao et al., 2016): A pioneering work that applies the VAE architecture for topic modeling with the encoder is implemented by multilayer perceptron, the variational distribution is a Gaussian distribution, and the variational inference is based on minimizing the KL-divergence (Blei et al., 2017).
- **GSM** (Miao et al., 2017): A variant of **NVDM** whose variational distribution is softmax of a Gaussian distribution.
- **NVLDA** (Srivastava and Sutton, 2017): Another variant of **NVDM** in which the variational distribution is Dirichlet distribution and approximated by a Laplace approximation based on Gaussian distribution.
- **ProdLDA** (Srivastava and Sutton, 2017): A variant of **NVLDA** in which the decoder is designed by following the product of expert model and the topics are unnormalized.
- **Scholar** (Card et al., 2018): This model is designed to incorporate the metadata and labels associated with documents into the modeling of topics. If no metadata and no labels are available, this is similar to **ProdLDA** except that the encoder has only a single linear layer.
- **NSMDM** (Lin et al., 2019): The sparsity of documents' topics is modeled by Gaussian sparsemax distribution, and the variational inference is based on the Wasserstein distance.
- **NSMTM** (Lin et al., 2019): This model is a variant of **NSMDM** in which the Gaussian sparsemax distribution is also used for modeling the sparsity of the topics .
- **NVCTM** (Liu et al., 2019): Again, this model is much similar to **NVDM** except that the decoder make uses of a complicated transformation sequence for modeling the correlation of topics within documents.

The above list is certainly not exhaustive. There are several other notable models proposed recently. We however do not examine them in this work as they either do not model documents as bags but sequences of words (e.g., (Dieng et al., 2016)) or are not generative models (e.g., (Yang et al., 2020)).

In our experiments, for each of the aforementioned models, we make use of the implementations provided by the authors if these are any. We re-implement the provided implementation if it is not written in Pytorch, strictly follow the original one. If the implementation is not provided, we implement the models ourselves, strictly follow the description and settings in the published papers.

Classical topic models. We compare these methods above with two classical topic models, namely Non-negative Matrix Factorization (**NMF**) (Zhao et al., 2017) and Latent Dirichlet Allocation (**LDA**) (Blei et al., 2003). We examine both two widely used learning methods for LDA: online variational inference (**o_LDA**) and Gibbs sampling algorithm (**g_LDA**).

2.2 Datasets

We uses two types of corpus: long text corpus containing *W2E-content* and *20News* (Lang, 2008);

---

[1] https://github.com/smutahoang/ntm

Table 1: Dataset statistic

Dataset	#documents	#words	Avg. length	#labels
20News	15,465	4,159	73.52	20
W2E-content	84,017	11,123	256.62	30
W2E-title	105,522	4,051	6.90	30
Web Snippets	12,295	4,722	14.42	8

and short text corpus including *W2E-title* and *Web Snippets* (Ueda and Saito, 2003). Both W2E-content and W2E-title are derived from news articles in W2E dataset (Hoang et al., 2018) by using the whole content or title respectively of the articles in the top 30 topics (by the number of news articles). Documents in these datasets are labeled by their topics (*W2E-content* and *W2E-title*), category of the discussion they belong to (*20News*), or the category of the webpage they were collected from (*Web Snippets*).

For each dataset, we preprocess the datasets by: removing stopwords, then iteratively removing infrequent words and too short documents. These are conventional in prior works on topic modeling, and is to make sure that we have sufficient data for learning meaningful topics. The basic statistics of the preprocessed datasets are shown in Table 1. The diversity and the large sizes of these datasets allow us to evaluate the models accurately.

2.3 Evaluation Metrics

For a comprehensive evaluation, we examine the performance of the models in three aspects: (i) *document modeling* – measured by the perplexity of unseen documents or held-out words (Blei et al., 2003), (ii) *topic discovery* – evaluated by topic coherence, and (iii) *document representation* – quantified through the performance of the obtained documents' topic vectors in downstream tasks.

For measuring the perplexities, we train the model using 90% of documents (or 90% of words in each document, respectively) in the dataset, and compute the perplexity on the remaining 10% of documents (or words, respectively). The coherence of the learned topics are measured by *normalized point-wise mutual information (NPMI)* (Lau et al., 2014) of their top words. Lastly, we use classification task to examine the obtained documents' topic vectors. That is, we use these vectors to train a logistic regression model to classify the documents in each dataset by their labels. For this experiment, we perform 10-fold cross validation, and report the average micro F1 scores across the folds. To obtain robust findings, and also to examine the consistency of the models' performance, for each metric, each model, each number of topics, and each dataset, we run the experiments 10 times, and report the mean and variance of the performance across the runs. The number of topics is varied from half to three times the number of labels of documents in the corresponding dataset.

3 Experiment Results & Findings

Figure 1 shows the mean and variance of the models' unseen-document perplexity on different datasets and with different number of topics. Similarly, Figures 2, 3, and 4 show the means and variances of the models' held-out word perplexity, average F1 scores, and topic coherence respectively on the same datasets and with different number of topics. Note that the **o_LDA** model has no unseen-document perplexity as it does not has a straightforward method for computing the perplexity, while **NMF** has no perplexities as it is not a generative model. From the figures, we can observe that:

- **NVDM** and **NVCTM** are generally better than other models in modeling and representing documents as their perplexities are significantly lower and their average F1 scores are significantly higher. This is reasonable and expected as these two models learn the unnormalized topic vectors for documents while the others learn the normalized ones[2]. However, they underperform other models in discovery cohesive topics as their coherence scores are much lower than others' in most cases.
- None of the neural topic models in our study outperforms the classical **LDA** on all the datasets and in all the metrics.
- The classical models (i.e., **o_LDA**, **g_LDA**, and **NMF**) are generally more stable than the neural ones as their variances are generally much smaller than those the neural models.

From the above observations, we make the following implications, which can be served as some guidelines for evaluation and practical usage of neural topic models.

- **LDA**, especially **g_LDA**, is a strong baseline for topic modeling. It should be used to evaluate neural topic models comprehensively.
- The performance of neural models may not be stable and may vary significantly from run

---

[2] Here the normalized vectors are probability distribution vectors: their elements are non-negative and summing up to 1

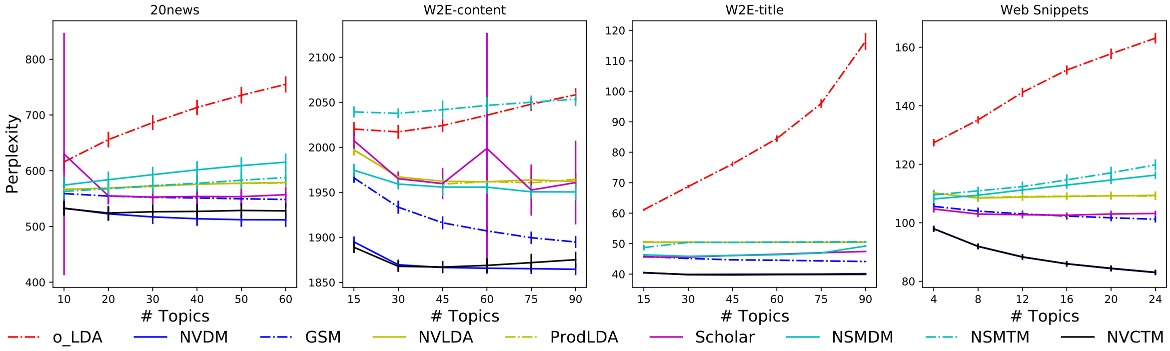

Figure 1: Unseen-document perplexity of the examined models: the lower is better

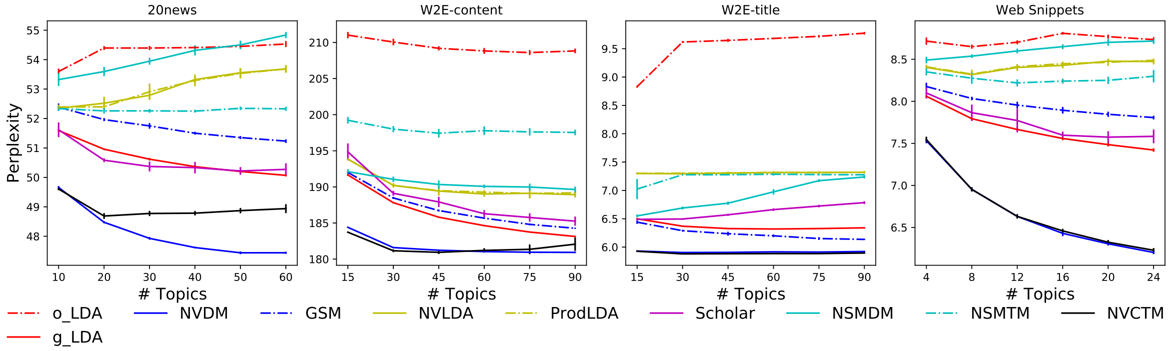

Figure 2: Held-out word perplexity of the examined models: the lower is better

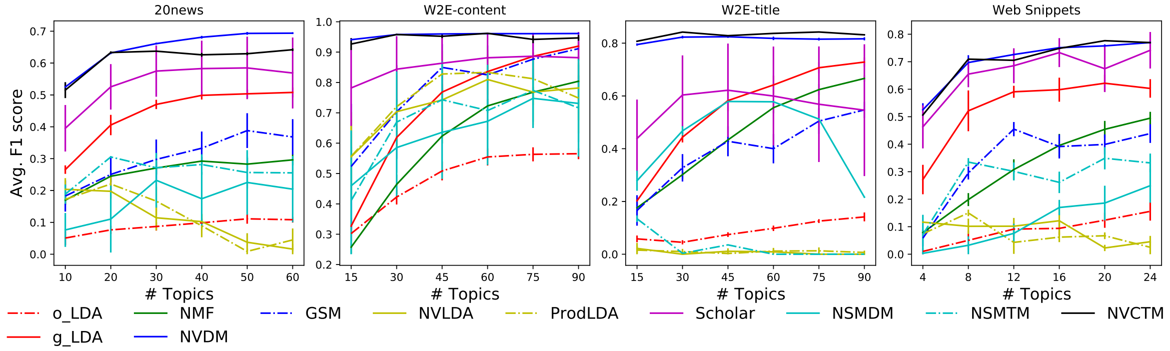

Figure 3: Average F1 scores of the examined models in downstream classification tasks: the higher is better

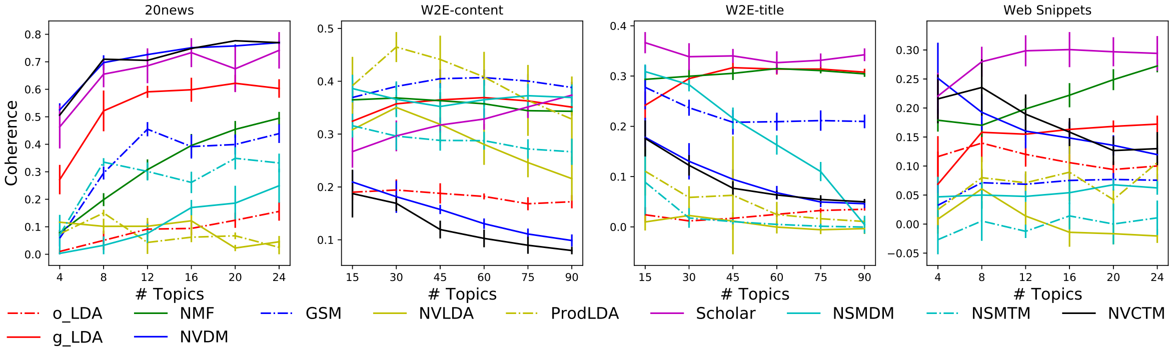

Figure 4: Coherence of topics learned by the examined models: the higher is better

to run. One should perform multiple runs on the same dataset to have accurate evaluation.

- Among the examined neural models, **Scholar** is more suitable for uncovering cohesive, meaningful topics from the corpus, while the unnormalized models (i.e., **NVDM** and **NVCTM**) are more suitable for modeling and representing documents in the corpus.

4 Conclusion

We have examined the performance state-of-the-art neural topic models in a systematic mechanism. From our extensive experiments, we found that classical methods, e.g. **LDA**, still have comparable expressive ability. We have also suggested some considerations for a comprehensive evaluation and practical usage of those models.

In the future, we would like to extend further by designing a fair comparison framework for neural methods that use other representation of documents, e.g. sequences of words, and from other approaches. We would also want to benchmark other issues in neural topic models, e.g., the posterior collapsing problem (He et al., 2019).

Acknowledgments

This work is supported by Vingroup Innovation Foundation (VINIF) in project code VINIF.2020.DA14

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Dallas Card, Chenhao Tan, and Noah A Smith. 2018. Neural models for documents with metadata. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2031–2040.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

Li Fei-Fei and Pietro Perona. 2005. A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*.

Tuan-Anh Hoang, Khoi Duy Vo, and Wolfgang Nejdl. 2018. W2e: A worldwide-event benchmark dataset for topic detection and tracking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1847–1850, New York, NY, USA. Association for Computing Machinery.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57.

Mike Innes. 2018. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 3(25):602.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Ken Lang. 2008. The 20 news groups data set. *http://people. csail. mit. edu/jrennie/20Newsgroups/*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539.

Tianyi Lin, Zhiyue Hu, and Xin Guo. 2019. Sparsemax and relaxed wasserstein for topic sparsity. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 141–149.

Luyang Liu, Heyan Huang, Yang Gao, Yongfeng Zhang, and Xiaochi Wei. 2019. Neural variational correlated topic modeling. In *The World Wide Web Conference*, pages 1142–1152.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2410–2419.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*.

Naonori Ueda and Kazumi Saito. 2003. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 737–744.

Liang Yang, Fan Wu, Junhua Gu, Chuan Wang, Xiaochun Cao, Di Jin, and Yuanfang Guo. 2020. Graph attention topic modeling network. In *Proceedings of The Web Conference 2020*, pages 144–154.

Renbo Zhao, Vincent Tan, and Huan Xu. 2017. Online nonnegative matrix factorization with general divergences. In *Artificial Intelligence and Statistics*, pages 37–45. PMLR.

Enhancing Chinese Word Segmentation via Pseudo Labels for Practicability

Kaiyu Huang[1] **Junpeng Liu**[1] **Degen Huang**[1*]
Deyi Xiong[2,3] **Zhuang Liu**[4] **Jinsong Su**[5]
[1]Dalian University of Technology [2]Tianjin University
[3]Global Tone Communication Technology Co., Ltd.
[4]Dongbei University of Finance and Economics [5]Xiamen University
`{kaiyuhuang,liujunpeng_nlp}@mail.dlut.edu.cn`
`huangdg@dlut.edu.cn` `dyxiong@tju.edu.cn`
`liuzhuang@dufe.edu.cn` `jssu@xmu.edu.cn`

Abstract

Pre-trained language models (e.g., BERT) significantly alleviate two traditional challenging problems for Chinese word segmentation (CWS): segmentation ambiguity and out-of-vocabulary (OOV) words. However, such improvements are usually achieved on traditional benchmark datasets and not close to an important goal of CWS: practicability (i.e., low complexity as a standalone task and high beneficiality to downstream tasks). To make a trade-off between traditional evaluation and practicability for CWS, we propose a semi-supervised neural method via pseudo labels. The neural method consists of a teacher model and a student model, which distills knowledge from unlabeled data to the student model so as to improve both in-domain and out-of-domain CWS. Experiments show that our proposed method can not only keep the practicability of the lightweight student model but also improve the performance of segmentation effectively. We also evaluate a range of heterogeneous neural architectures of CWS on downstream Chinese NLP tasks. Results of further experiments demonstrate that our proposed segmenter is reliable and practical as a pre-processing step of the downstream NLP tasks at the minimum cost.[1]

1 Introduction

Natural language processing (NLP) tasks often leverage word-level features to exploit lexical knowledge. Segmenting a sentence into a sequence of words, especially for languages without explicit word boundaries (e.g., Chinese) not only extracts lexical features, but also shortens the length of the sentence to be processed. Thus, word segmentation, detecting word boundaries, is a crucial preprocessing task for many NLP tasks. In this aspect, Chinese word segmentation (CWS) is widely acknowledged as an essential task for Chinese NLP.

CWS has made substantial progress in recent studies on several benchmarks, which is reported by Huang and Zhao (2007) and Zhao et al. (2019). In particular, pretrained language models (PLMs), like BERT (Devlin et al., 2019), have established new state-of-the-art in sequence labeling (Meng et al., 2019). Various fine-tuning methods have been proposed to improve the performance of in-domain and cross-domain CWS based on PLMs (Huang et al., 2020; Tian et al., 2020). The two challenging problems in CWS, segmentation ambiguity and out-of-vocabulary (OOV) words, have been significantly mitigated by PLM-based methods that are fine-tuned on large-scale annotated CWS corpora. Such methods are even reaching human performance on benchmarks. Nevertheless, CWS is more valuable as a prelude for downstream NLP tasks than as a standalone task. Intrinsic evaluation of CWS on benchmark datasets only examines the effectiveness of current neural methods on word boundary detection. To better apply CWS in downstream NLP tasks, we should comprehensively re-think CWS from the perspective of practicability. In this paper, we define the practicability of CWS with two aspects: low complexity as a standalone task and high beneficiality to downstream tasks.

The complexity is twofold: 1) complexity of implementation and 2) time and space complexity of a CWS algorithm. Previous neural methods usually require additional resources (Zhou et al., 2017; Ma et al., 2018; Zhang et al., 2018b; Zhao et al., 2018; Yang et al., 2019; Qiu et al., 2020), such as external pre-trained embeddings. The complexity of implementation is reflected in the difficulty of acquiring external resources. External resources

---

[*] Corresponding author

[1] Our code is available at `https://github.com/koukaiu/dlut-nihao`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4369–4381
August 1–6, 2021.

vary in quality and the length of time for computation, For example, it is time-consuming to obtain effective pre-trained embeddings as they are trained on a huge amount of data. Generally, it is difficult to maintain high CWS performance for many previous neural methods in a low-resource environment. Neural methods with external resources achieve high CWS performance but at the cost of a high complexity of implementation. On the other hand, for training and inference, PLM-based CWS methods also consume large memory to store a huge number of parameters of their models. The speed of inference is usually slow. The huge memory consumption and slow inference prevent PLM-based CWS models from being deployed in small-scale smart devices. And, as CWS is often used with downstream models, this even weakens the applicability on smart devices as CWS is not supposed to take too much overhead in this situation.

The second is the beneficiality to downstream tasks. CWS is rarely used as a standalone task in industry. Existing CWS evaluations only rely on benchmarks and analyze the behavior of segmentation methods in a static scenario. Some well-known benchmarks are quite old (e.g., Bakeoff-2005) and not challenging for neural CWS anymore. Such evaluations are intrinsic, which are not associated with downstream NLP tasks. High CWS performance (e.g., Precision and F_1) does not mean that segmentation results are beneficial to downstream processing. Additionally, benchmark datasets have a plenty of segmentation noises that affect CWS training and evaluation. For instance, although the structure of “副” (vice) + “X” is segmented as two words: “副” (vice) and “X” in training data and never unified as a single word, “副校长” (vice-president) appears as one word in test data, note that: X presents any job titles, e.g., “总统” (president) and “经理” (manager). There are also many obvious errors due to annotation inconsistency in data. We have found, in one benchmark dataset, the word “操作系统” (operating system) is regarded as two words [“操作” (operate) + “系统” (system)] 6 times and appears as one word 14 times, respectively. Therefore, to measure and improve the beneficiality of CWS to downstream tasks, intrinsic evaluations on CWS benchmark datasets are not sufficient. We should perform extrinsic evaluations with downstream tasks.

To address the aforementioned practicability issue of CWS, we propose a semi-supervised neural method via pseudo labels. The method consists of two parts: a teacher model and a student model. First, we use a fine-tuned CWS model that is trained on the annotated CWS data as the teacher model, which can achieve competitive performance in traditional perspective for CWS. Then we collect massive unlabeled data and distill knowledge from the teacher model to the student model by generating pseudo labels. We filter out noisy pseudo labels to provide reliable knowledge for training the student model. The unlabeled data is easier to obtain than other external resources (e.g., lexicon and pre-trained embeddings) and can be updated anytime at a low cost. And we use the lightweight student model for inference, hence significantly reducing the memory consumption and inference time complexity. The practicability of our proposed method is competitive.

To sum up, the contributions of this work are as follows:

- Our proposed method distills knowledge from the teacher model via unlabeled data to coach the lightweight student model. The proposed method achieves a noticeable improvement over strong baselines for CWS by the traditional intrinsic evaluation.

- The lightweight student can be deployed on a small-scale device, even in a non-GPU environment. We abandon the PLM neural architectures (teacher model) during decoding. The speed of decoding is thus fast for practical application. Our method reduces the complexity of implementation, inference time, and memory consumption.

- We empirically investigate the effectiveness of the proposed method to downstream Chinese NLP tasks and analyze the impact of segmentation results on them via extrinsic evaluations.

2 Related Work

Since Xue (2003) formalizes CWS as a sequence labeling problem, many traditional statistical methods have achieved high performance for CWS on several benchmarks (Emerson, 2005). According to (Huang and Zhao, 2007) and (Zhao et al., 2019), CRF-based models (Tseng et al., 2005; Zhao and Kit, 2008; Zhao et al., 2010; Sun et al., 2012; Zhang et al., 2013) and neural methods (Zheng et al., 2013;

Pei et al., 2014; Chen et al., 2015; Cai and Zhao, 2016; Cai et al., 2017) have been reported to outperform traditional methods with high F_1 scores (0.95-0.97). In particular, Long Short-Term Memory Networks (LSTM) are the main backbone networks being used in these methods (Huang et al., 2015; Ma et al., 2018; Yang et al., 2019). Except for using LSTM, self-attention networks have been also employed for CWS (Duan and Zhao, 2020).

The OOV problem has been a long-standing challenge for CWS and it is particularly serious in the cross-domain scenario. To relieve this issue, many studies incorporate a variety of pre-trained word embeddings and external resources into CWS models (Zhou et al., 2017; Zhang et al., 2018b,a; Yang et al., 2019). Recently, with the development of PLMs (Devlin et al., 2018; Liu et al., 2019), fine-tuning methods benefit from a huge amount of the pre-trained knowledge for alleviating the OOV problem for CWS (Meng et al., 2019; Tian et al., 2020; Huang et al., 2020; Qiu et al., 2020). Such methods are nearly reaching human-level performance.

Nevertheless, external resources and PLMs result in additional costs in memory consumption and inference time. Knowledge distillation has been proposed to alleviate this additional cost issue (Ba and Caruana, 2014; Hinton et al., 2015). Kim and Rush (2016) propose to use knowledge distillation for neural machine translation while Mukherjee and Awadallah (2019) study several aspects of distillation to match the student model for sentiment classification. Jiao et al. (2020) adopt multiple distilling strategies to minimize the number of the parameters of the pre-trained language model. Different from these previous studies, our proposed method utilizes unified pseudo labels to improve the performance of the lightweight model. The model can provide positive influence as a preprocessing step to downstream tasks, compared with previous state-of-the-art methods.

3 Proposed Framework

Aiming at not only keeping competitive performance on benchmarks but also reducing the complexity of the CWS methods, our proposed framework consists of two essential modules: a student model and a teacher model, as shown in Figure 1. There is an obvious performance gap between the model based on PLMs (Huang et al., 2020) and the lightweight model (Duan and Zhao, 2020). The OOV issue is the main reason for the gap. Since the teacher model based on fine-tuned PLMs with high complexity can alleviate the OOV issue effectively, we use a combination of PLM-based teacher and lightweight student. First, the teacher model transfers pre-trained knowledge into a specific data distribution by annotating unlabeled data. Then we utilize a huge amount of such annotated data to distill knowledge from the teacher model to the lightweight student model. The pseudo labels provided by the teacher model can help the lightweight model to alleviate the OOV issue of CWS.

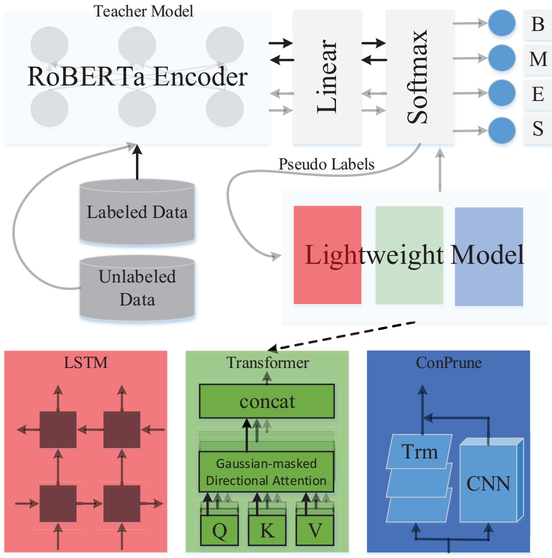

Figure 1: The illustration of our proposed framework. The red, green and blue blocks indicate the heterogeneous student models.

3.1 Teacher Model

Recently, there are several PLMs (e.g., BERT and RoBERTa) that have shown competitive performance for many NLP tasks. In particular, a modified RoBERTa model has been built for Chinese NLP tasks (Cui et al., 2019). Inspired by the previous success of PLM-based models on CWS (Huang et al., 2020), we use the RoBERTa-WWM PLM as the teacher model.

Normally, PLMs are trained for predicting words in general. To adapt PLMs and transfer their knowledge to CWS, we need to fine-tune PLMs on the annotated data of CWS. Let X denote the inputs, which are converted into a sequence of embeddings. For consistency, two tags (“[CLS]” and “[SEP]”) are added to the beginning and end of each sentence, respectively. A Linear transfer layer with

$W^{(t)} \in R^{d_{model}*N}$ replaces the original component, where d_{model} is the same as the size of dimensions of the pre-trained model and N presents number of tags in CWS annotated data ($N = 4$). We convert CWS annotations into annotations with a 4-tag set $T = \{B, M, E, S\}$ that indicates the **B**egin, **M**iddle, **E**nd of a word, or a **S**ingle character forming a word. After linear mapping, the teacher model adopts the function of Softmax and the greedy search for decoding.

$$p^{(t)}(x) = \text{Softmax}(h_t(x) \cdot W^{(t)} + b^{(t)}) \quad (1)$$

where $h_t(x)$ represents the hidden states of the teacher model. Complex algorithms (e.g., CRF or beam search) for decoding are abandoned in order to reduce the complexity. In addition, these complex algorithms only obtain a slight improvement for CWS. CRF increases the time complexity by n times and beam search requires more search time varying with the beam size, compared with the greedy search.

$$GreedySearch \rightarrow O(Mn)$$
$$BeamSearch \rightarrow O(Mnb^2)$$
$$CRF \rightarrow O(Mn^2)$$

where M is a constant representing other factors in the model complexity, n is the size of the sentence, and b is the width of beam.

The training of the teacher model is to minimize the errors by solving the following optimization function:

$$\min_{W^{(t)}} J_{seg}(y(x)|p^{(t)}(x, W^{(t)})) \quad (2)$$

where the loss function J_{seg} is computed by:

$$J_{seg}(y(x)|p^{(t)}(x)) = -\sum_{x} y(x)\text{log}p^{(t)}(x) \quad (3)$$

3.2 Student Model

To improve the practicability of CWS, our proposed framework rediscovers the potential of the lightweight models. The lightweight model suffers from the OOV problem compared with the teacher model. However, the lightweight model can help us to solve the practicability issue of CWS. We propose multiple lightweight models as the student model, as shown in Figure 1.

-ConPrune. This is a pruned PLM model, where three quarters of the PLM's layers are discarded. Particularly, we only use the first top 3 layers of the entire 12 layers. We also incorporate a Convolutional Neural Network (CNN) encoder to capture the local features of the sequence.

-LSTM. LSTM is the most popular architecture for sequence labeling tasks (Ma et al., 2018). As shown in Figure 1, for each input character c_i, the corresponding character uni-gram embedding and bi-gram embedding are represented as e_{c_i} and $e_{c_i c_{i+1}}$, respectively. The LSTM model is fed with the two types of character embeddings by concatenation operation, $w_i = e_{c_i} \oplus e_{c_i c_{i+1}}$. The loss function and the decoding are the same as the teacher model.

-Transformer. The Transformer is usually not working as well as LSTM for sequence labeling tasks despite its success on other tasks. We propose a new Transformer variant that is inspired by Duan and Zhao (2020). The modified Transformer utilizes the Gaussian directional mask to encode unigram features.

-CRF. Although CRF is not a dominant model for CWS, it still has great significance for practicability. We only utilize uni-gram and bi-gram features for CRF, keeping the same as neural methods for a fair comparison. It does not rely on any auxiliary features, e.g., accessor variety (AV) (Feng et al., 2004) or pointwise mutual information (PMI) (Sun et al., 1998).

All formulations and details of the student models are shown in Appendix A.

3.3 Pseudo Labels

Neural networks typically predict the probability of each class by Softmax. In the form of distillation, knowledge is transferred to the distilled model by using a distribution that is produced by the teacher model with a temperature in its Softmax. However, the architectures of the student models are completely different from the teacher model, as shown in the last section. Unlike previous studies on distilling knowledge, the process of knowledge distillation in our framework is essentially the same as the original CWS task. Particularly, our proposed method distills the knowledge from the teacher model to the student model by using a huge amount of unlabeled data as the knowledge container. It is easy to obtain unlabeled data from the Internet. The pseudo labels are generated together with noisy labels and we reduce the impact of noisy labels. Due to the high correlation between training data and unlabeled data, we directly distill knowl-

	MSR		PKU		AS		CityU		CTB6	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
# Char	4,050K	184K	1,826K	173K	8,368K	198K	2,403K	68K	1,156K	134K
# Word	2,368K	107K	1,110K	104K	5,500K	123K	1,456K	41K	701K	82K

	NER-PKU		WMT-18			Unlabeled
	Train	Test	Train	Dev	Test	People's Daily
# Sen	17,546	1,714	24,752,392	2,002	2,001	115,856

Table 1: Statistics of our datasets. The upper part of the table shows the details of the CWS benchmarks and the bottom for the downstream NLP tasks. All datasets come from the official websites.

edge from the teacher model to the student model, and the final loss is shown as follow:

$$J_u(y(u)|p^{(s)}(u)) = -\sum_{x_u} y_u \text{log} y^*(x_u) \tag{4}$$

$$\begin{aligned} J(\Theta_s, \Theta_u) &= J_{seg}(\Theta_s) + \alpha J_u(\Theta_u), \\ \Theta_s : y(x)|p^{(s)}(x)&, \Theta_u : y(u)|p^{(s)}(u) \end{aligned} \tag{5}$$

where s denotes the student model, α is a weight to balance the losses on the labeled data and unlabeled data ($\alpha = 0.5$ in our experiments). The loss function is calculated with two parts. One is from the labeled data, the other is from the unlabeled data x_u. Hard prediction of the teacher model on the unlabeled data produces noisy labels y_u. And the prediction of the student model on unlabeled data is y^*. To reduce the redundant computation, pseudo labels are mix-sampled according to a regular interval. The sampling strategy chooses the different n-gram features with the annotated data, which makes the distribution of unlabeled sentences different from the annotated data. Instead of optimizing the loss function jointly, we adopt a two-stage optimizing method. The first stage trains student models on the large-scale annotated data. In the second stage, the student model is continued to be trained on the data with labels predicted from the teacher model. Since the teacher model is also fine-tuned on the annotated data, the two-stage training does not suffer from the catastrophic forgetting issue.

4 Experiments

4.1 Datasets and Settings

To examine the advantage of distilling knowledge and the complexity of our proposed framework via pseudo labels, we conducted experiments on five benchmarks (Bakeoff-2005[2] and CTB6). The statistics of the benchmarks are shown in Table 1. We randomly picked 10% sentences from the training data as the development data for tuning. The unlabeled data were collected from the People's Daily website. We crawled 5,000 articles. For consistency, we pre-processed unsegmented sentences, which is similar to previous work (Cai et al., 2017). In addition, to empirically validate the beneficiality of the proposed CWS method to downstream tasks, we carried out comprehensive experiments on named entity recognition (NER) and machine translation (MT). The details of datasets for these two tasks are shown in Table 1. We used F_1 as the evaluation metric for NER and BLEU (Papineni et al., 2002) for MT.

Parameters	Teacher	Student		
		Prune	LSTM	Trans
Hidden states	768	768	256	256
uni-gram embeds	768	768	50	256
bi-gram embeds	-	-	50	-
learning rate	2e-5	1e-5	1e-3	0.1
batch size	64	256	64	4096
dropout	0.1	0.1	0.2	0.1
hidden layers	12	3	3	6
epochs	10	20	30	200

Table 2: The hyper-parameters. "Prune" represents the ConPrune and "Trans" represents the modified Transformer.

To fine-tune the teacher model (i.e., RoBERTa-WWM), we adjusted a few crucial hyper-parameters for it, as shown in Table 2. The hyper-parameters of the student model were tuned with the development sets. We evaluated inference speed for all models on the same hardware configuration (Non-GPU environment: Intel(R) Core(TM) i9-10900KF CPU @ 3.70GHz //GPU environment: Nvidia GeForce RTX 3090). All other hyper-parameters and search ranges are shown in Appendix B.

---

[2] `http://sighan.cs.uchicago.edu/bakeoff2005/`

Models	AU	PKU	MSR	AS	Cityu	CTB6	S-Cpu (ms.)	S-Gpu (ms.)
Teacher	-	96.68	98.14	96.62	97.92	97.55	119.24	3.61
CRF	×	95.02	96.72	95.40	94.25	95.65	**3.54**	-
	√	**96.19**	**97.33**	**95.70**	**96.18**	**96.79**		
LSTM	×	95.15	97.26	95.29	95.13	94.98	21.66	**1.03**
	√	**96.25**	**97.58**	**95.81**	**96.30**	**96.66**		
Greedy LSTM	×	95.37	96.83	95.22	95.54	96.06	11.62	-
	√	**96.43**	**97.17**	**95.43**	**96.44**	**97.04**		
Transfomer	×	95.46	97.59	95.96	95.26	96.10	60.41	1.70
	√	**96.43**	**97.79**	**96.25**	**96.91**	**97.16**		
ConPrune	×	96.34	97.93	96.31	97.28	97.19	30.17	**1.29**
	√	**96.76**	**98.07**	**96.47**	**97.58**	**97.45**		

Table 3: Results on the Bakeoff-2005 dataset. "AU" indicates whether the student model utilizes unlabeled data. Greedy LSTM follows Cai et al. (2017), which is a LSTM model adapted to the CPU environment. S-CPU/GPU denotes the inference speed (ms per sentence) on the CPU/GPU environment.

Models	PKU	MSR	AS	Cityu	CTB6
(Chen et al., 2015)	96.5	97.4	-	-	96.0
(Zhou et al., 2017)	96.0	97.8	-	-	96.2
(Ma et al., 2018)	96.1	97.4	96.2	97.2	96.7
(Gong et al., 2019)	96.7	96.5	94.5	93.7	-
(Duan and Zhao, 2020)	95.5	97.7	95.7	96.4	-
(Tian et al., 2020)	96.5	98.4	96.6	97.9	97.3
(Huang et al., 2020)	96.7	98.1	-	-	97.6
Transformer	96.4	97.8	96.3	96.9	97.2
ConPrune	**96.8**	**98.1**	**96.5**	**97.6**	**97.5**

Table 4: Experiment results on the Bakeoff-2005 datasets. The best results obtained by non-PLM models. Our results are significantly better ($p < 0.05$ bootstrap resampling) than all previous state-of-the-art results.

4.2 Results of Intrinsic Evaluation

As shown in Table 3, we investigated the effect of the proposed method on the benchmark Bakeoff-2005, which is the most widely-used dataset for CWS. "Teacher" denotes the teacher model as introduced in section 3.1. It achieves competitive performance as it is based on a state-of-the-art pretrained model. Other models are the student models.

Experimental results in Table 4 show that our proposed semi-supervised method significantly improves the performance on all 5 benchmark datasets, compared with the pure student model. Surprisingly, results of the proposed semi-supervised method are even close to those of the teacher model. We also compared our proposed method against previous SOTA models, as shown in Table 4. In particular, Tian et al. (2020) and Huang et al. (2020) utilize PLMs which are slower in inference than non-PLM CWS models. This paper focuses on the methods with low complexity. These results demonstrate that our proposed method achieves state-of-the-art performance compared with non-PLM methods. Although there is a small performance gap between our proposed method and fine-tuned PLM methods, the advantage of our method over PLMs is that our method is much faster in both CPU and GPU environments, as displayed in Table 3, which is the key interest of our work. From this perspective, our method is more readily to be used in downstream tasks than previous state-of-the-art PLM methods. In addition, our proposed method not only maintains the advantages of the basic neural methods but also has a low complexity for practicability. Meanwhile, the method leverages easily available unlabeled data to make up for the insufficiency of the student model.

4.3 Results of Extrinsic Evaluation

The performance of models on various CWS benchmarks only demonstrates the merits of models themselves. However, CWS results of different methods that achieve good performance on benchmarks are not necessarily beneficial for specific downstream tasks. We therefore investigated the effect of using different CWS results on the two popular downstream Chinese NLP tasks (NER and MT) to analyze the beneficiality of CWS methods to other tasks. The benchmarks of these two tasks that we adopted are both widely acknowledged in the literature of NER and MT. Particularly, we used

	MT (BLEU)		CWS (F_1)		NER (F_1)		
	ZH→EN		Seg. Train	Seg. Test	NR	NP	NT
Char	21.16	Gold	1.000	1.000	.895	.880	.864
Teacher	23.51 +2.35		.991 -.009	.993 -.007	.897 +.001	.875 -.005	.862 -.002
CRF	23.37 +2.21		.990 -.010	.992 -.008	.907 +.012	.880 -.000	**.877 +.013**
CRF†	23.68 +2.52		.990 -.010	.992 -.008	**.915 +.020**	.881 +.001	.862 -.002
ConPrune	23.71 +2.55		.990 -.010	.992 -.008	.903 +.008	.879 -.001	.863 -.001
ConPrune†	**23.73 +2.57**		.988 -.012	.991 -.009	**.915 +.020**	**.884 +.004**	**.877 +.013**

Table 5: Results on NER and MT. "NR, NP and NT" represent entities of person, place and organization. "Gold" denotes gold-standard word segmentations for NER and "Char" indicates the character-level neural model based on Transformer for baseline comparison. † indicates that the corresponding model utilizes the proposed semi-supervised method.

the "PKU" open resources for NER evaluation and a Chinese-to-English machine translation task from WMT-18[3] for MT evaluation. The model for NER employs word-based LSTM to extract context information and applies a CRF layer stacked over LSTM for decoding. It utilizes random word-level embeddings which can be further fine-tuned later. The evaluation of this task is the same as CWS (F_1). The MT model is based on the Transformer (Vaswani et al., 2017) neural network. We used Byte Pair Encoding (BPE) for alleviating the issue of rare words. We kept all other hyper-parameters of the NER and MT models as those widely used. We then fed CWS results produced by different models into the NER and MT models. Results are shown in Table 5.

Clearly, our proposed method can provide word segmentations that are beneficial for the two downstream tasks. The performance of NER using segmentation results yielded by our proposed method is better than others, even ground-truth word segmentations. All segmentation systems achieve good performance with no evidence of OOV. However, there are still some distinctions between two CWS methods, which will be analyzed in the case study section. Except for the quality of word segmentations, the speed of our proposed method is fast enough to support specific downstream tasks. Surprisingly, we find that word segmentations with high F_1 scores on CWS benchmarks do not necessarily indicate high performance on downstream tasks. Especially, the optimal performance of segmentation results ("Seg. train" and "Seg. test") does not suggest the highest performance on NER ("NR", "NP" and "NT"), as shown in Table 5. This might be due to two reasons. First, gold results in NER have noises, which is similar to CWS. Our proposed method has a strong robustness to deal with noisy labels. Second, word segmentation errors do not necessarily cause error propagation.

4.4 Case Study

To make further progress on CWS, it is important to understand errors that CWS methods are making. Hence we randomly selected typical errors from the PKU test set and manually analyzed them.

The segmentation errors can be roughly divided into two categories. One is the type of errors with OOV words. The proposed semi-supervised method can alleviate the issue of OOV words effectively. For instance, "威尔第" (Verdi) is segmented into two words incorrectly by the pure student model. This split frequently occurs in the unlabeled data, and such knowledge is distilled from the teacher model. The semi-supervised method can revise these OOV words.

Except for the type of errors of OOV words, the rest of errors are mainly caused by segmentation inconsistency. For example, the word "人" (person) should be regarded as a suffix word behind some words, e.g., "中国+人" (Chinese) and "代理+人" (agent). "人" (person) also exists as part of other words, e.g., "关系人" and "继承人". Simply training neural model on such inconsistent segmentation data may be insufficient to solve these segmentation errors without further efforts in data processing. This situation naturally raises a question: do the errors caused by segmentation inconsistency really influence the performance of downstream NLP tasks?

To answer this question, we conducted additional experiments on the two downstream NLP tasks. In NER, segmentation results of non-entity

---

[3] `http://data.statmt.org/wmt18/translation-task/`

words hardly affect the performance of NER. For instance, the word “不懈奋斗” (untiringly struggle) is regarded as a word according to the criterion of “PKU”. Previous state-of-the-art methods that achieve high F_1 scores for CWS can segment it correctly. While our proposed method splits this unit into two independent words “不懈” (untiringly) and “奋斗” (unremitting). However, these two words do not belong to any entities. In other words, the better performance for segmenting non-entity words does not necessarily indicate better performance of NER. In addition, segmentation results of entity words directly affect the veracity of NER. There is a phrase “西方七国集团” (the Group of Seven, abbreviations: G7) in a sentence. This segment is an organizational entity. In word segmentation, it is regarded as two words “西方” (western) and “七国集团” (the group of seven countries) according to ground-truth segmentation results. Previous state-of-the-art methods are usually able to segment it correctly. By contrast, our proposed method segments it into three words “西方” (western), “七国” (seven countries) and “集团” (group). Surprisingly, the final result of NER is out of expectation. The entity with incorrectly segmented words by our method is correctly recognized. Gold segmentation does not achieve a better result on this entity. The vague boundary of a word may increase the uncertainty and difficulty of downstream Chinese NLP tasks. There are many prefix and suffix words in Chinese. Sometimes, it is hard to determine whether these words are a single word or not. For this reason, high performance of CWS is not equal to high performance of Chinese NER.

In MT, due to the technique of BPE, rare words are segmented into sub-words. The issue of unknown words can thus be alleviated effectively. Even if a word as simple as “日本” (Japan) is segmented into two words incorrectly, NMT models are able to prevent the error propagation of segmentation in the training step. Thus, a faster segmentation system, rather than a high-performance segmentation system, is more practical for NMT. To analyze NMT translation differences between two sentences with different segmentation results, we also supply the additional analysis in Appendix C. We find that segmentation results with slight differences make translation results varying. The boundary of words may lead to these differences. But we also conjecture that this is more due to the robustness of NMT models.

5 Conclusion

To bring a positive impact of CWS to downstream NLP tasks, this paper makes a trade-off between the traditional evaluation and the complexity (e.g., implementation and decoding speed), which makes the segmenter more practical. We propose a semi-supervised method that distills knowledge via pseudo labels into the lightweight student model. The method is low coupling, which significantly improves the performance of multiple heterogeneous tiny neural architectures. The proposed framework can achieve competitive performance on CWS benchmarks and the speed of the student model also satisfies the practical requirement. In summary, the advantages of our model are twofold. First, the inference speed of the method is much faster than PLM methods. It can run under low resource conditions, even on CPUs. Second, the model provides efficient segmentation results for downstream NLP tasks.

Acknowledgments

We sincerely thank the reviewers for their insightful comments and suggestions to improve the quality of the paper. The authors gratefully acknowledge the financial support provided by the National Key Research and Development Program of China (2020AAA0108004) and the National Natural Science Foundation of China under(No.U1936109). Deyi Xiong is partially supported by the joint research center between GTCOM and Tianjin University.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206, Lisbon, Portugal. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sufeng Duan and Hai Zhao. 2020. Attention is all you need for Chinese word segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3862–3872, Online. Association for Computational Linguistics.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN workshop on Chinese Language Processing*, pages 123–133, Jeju Island, Korea.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for chinese word extraction. *Computational Linguistics*, 30(1):75–93.

Jingjing Gong, Xinchi Chen, Tao Gui, and Xipeng Qiu. 2019. Switch-lstms for multi-criteria chinese word segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6457–6464.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Chang-Ning Huang and Hai Zhao. 2007. Chinese word segmentation: a decade review. *Journal of Chinese Information Processing*, 21(3):8–19.

Kaiyu Huang, Degen Huang, Zhuang Liu, and Fengran Mo. 2020. A joint multiple criteria model in transfer learning for cross-domain Chinese word segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3873–3882, Online. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art chinese word segmentation with bi-lstms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908.

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems*, pages 2742–2753.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2019. Distilling transformers into simple neural networks with unlabeled transfer data. *arXiv preprint arXiv:1910.01769*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303, Baltimore, Maryland, USA. Association for Computational Linguistics.

Xipeng Qiu, Hengzhi Pei, Hang Yan, and Xuanjing Huang. 2020. A concise model for multi-criteria Chinese word segmentation with transformer encoder. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2887–2897, Online. Association for Computational Linguistics.

Maosong Sun, Dayang Shen, and Benjamin K Tsou. 1998. Chinese word segmentation without using lexicon and hand-crafted training data. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1265–1271.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 253–262, Jeju, Republic of Korea. Association for Computational Linguistics.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. 2020. Improving Chinese word segmentation with wordhood memory networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8274–8285, Online. Association for Computational Linguistics.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff2005. In *Proceedings of the Fourth SIGHAN workshop on Chinese Language Processing*, pages 168–171, Jeju Island, Korea.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Jie Yang, Yue Zhang, and Shuailong Liang. 2019. Subword encoding in lattice lstm for chinese word segmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2720–2725.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, Seattle, Washington, USA. Association for Computational Linguistics.

Meishan Zhang, Nan Yu, and Guohong Fu. 2018a. A simple and effective neural model for joint word segmentation and pos tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1528–1538.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018b. Neural networks incorporating dictionaries for chinese word segmentation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hai Zhao, Deng Cai, Changning Huang, and Chunyu Kit. 2019. Chinese word segmentation: Another decade review (2007-2017). *arXiv preprint arXiv:1901.06079*.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for chinese word segmentation. *ACM Transactions on Asian Language Information Processing*, 9(2):1–32.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *The Sixth SIGHAN Workshop on Chinese Language Processing*, pages 106–111, Hyderabad, India.

Lujun Zhao, Qi Zhang, Peng Wang, and Xiaoyu Liu. 2018. Neural networks incorporating unlabeled and partially-labeled data for cross-domain chinese word segmentation. In *IJCAI*, pages 4602–4608.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark. Association for Computational Linguistics.

Appendix

A Model Architecture

This paper introduces multiple heterogeneous tiny neural architectures as the student model. To better describe each model, all formulations of the student models are shown as follows.

-ConPrune. ConPrune prunes three quarters of 12 layers of the used PLM. To extract local features, we incorporate a Convolutional Neural Network (CNN) encoder into the pruned model. The kernel size determines the distance of scanning. The input sequence is converted into two vector matrices $\mathbf{E}_t$ and $\mathbf{E}_c$. Word positions are also mapped into a feature matrix $\mathbf{E}_p$. The input to the encoder consists of four parts that are token embedding $\mathbf{E}_t$, position embedding $\mathbf{E}_p$, segment embedding $\mathbf{E}_s$ and CNN embedding $\mathbf{E}_c$. Because of the specificity for CWS, all segment embeddings of sequences are regarded as the same mapping matrix $\mathbf{E}_s$. The input of the two components are:

$$\mathbf{E}_{trm} = \mathbf{E}_t + \mathbf{E}_p + \mathbf{E}_s, \mathbf{E}_{cnn} = \mathbf{E}_c \tag{6}$$

The convolutional encoder involves a filter $W \in \mathbb{R}^{hk}$, which is applied to a window of h characters to produce a new feature.

$$W_{cnn} = \operatorname{Relu}\left(W \cdot x_{i:i+h-1} + b\right) \tag{7}$$

where $Relu$ is a type of activation function. $x_{i:i+h-1}$ indicates the matrix of $\mathbf{E}_c$.

The pruned encoder consists of 3 base Transformer encoded layers with multiple multi-head self-attention layers to extract contextual features for each character. The multi-head self-attention layer adopts "Scaled Dot-Product Attention" which is formulated as:

$$\operatorname{Attn}(Q, K, V) = \operatorname{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{8}$$

where Q, K, V represents a query and a set of key-value pairs through a linear transformation respectively, and d_k is the dimension of K.

$$\operatorname{MultiAttn}\left(E_{trm}\right) = \left[head_1, ..., head_k\right] W^O \tag{9}$$

$$head_i = \operatorname{Attn}\left(E_{trm} W_i^Q, E_{trm} W_i^K, E_{trm} W_i^V\right) \tag{10}$$

where W^O, W_i^Q, W_i^K, W_i^V are trainable parameters.

-LSTM. LSTM can be used as a lightweight architecture for sequence labeling tasks. Rigorous tuning can obtain competitive performance for CWS. The model handles sequence features very well. For each input character c_i, the corresponding character uni-gram embedding and bi-gram embedding are represented as e_{c_i} and $e_{c_i c_{i+1}}$, respectively. The LSTM model is fed with the two types of character embeddings by concatenation operation, $w_i = e_{c_i} \oplus e_{c_i c_{i+1}}$. We get the outputs of the student representations from the LSTM model as follow:

$$\overrightarrow{h_s} = \overrightarrow{LSTM}\left(w_1, w_2, \ldots, w_i\right) \tag{11}$$

$$\overleftarrow{h_s} = \overleftarrow{LSTM}\left(w_1, w_2, \ldots, w_i\right) \tag{12}$$

$$h_s = \overrightarrow{h_s} \oplus \overleftarrow{h_s} \tag{13}$$

-Transformer. This paper adopts a modified Transformer which follows the previous study by Duan and Zhao (2020). The modified Transformer changes the multi-head self-attention to the multi-head Gaussian directional attention. Given an input sequence, the attention function can be described as mapping a query and a set of key-value pairs. The Gaussian directional attention incorporates the Gaussian directional attention into traditional self-attention module to pay attention to the neighboring characters of each position and capture features between characters as a fix Gaussian weight for attention. The Gaussian weight function and the multi-head Gaussian directional attention are computed as:

$$G_{i,j} = \sqrt{\frac{2}{\sigma\pi}} \int_{-\infty}^{-d_{i,j}} exp(-\frac{x^2}{2\sigma^2}) dx \tag{14}$$

where i and j are two adjacent positions, $d_{i,j}$ is the distance between the two characters, σ represents the standard deviation of the function. We set this hyper-parameter as 2.

$$\operatorname{GDA}(Q, K, V) = \operatorname{softmax}(\frac{QK^T G}{\sqrt{d_k}}) V \tag{15}$$

where GDA represents the Gaussian directional attention, d_k is the dimension of the matrix K. Q, k and V are vectors which are similar to the base Transformer. The multi-head attention are computed as:

$$\begin{aligned} MultiGDA &= [head_1, head_2, ..., head_k] W^o \\ head_i &= GDA(QW_i^q, KW_i^k, VW_i^v) \end{aligned} \tag{16}$$

CONFIG	Parameters
hidden states	768
optimizer	Bert Adam
learning rate	[3e-5,**2e-5**,1e-5]
batch size	[16,32,64,**256**]
dropout	[**0.1**,0.2,0.4]
epochs	20

Table 6: The hyper-parameters of the teacher model.

CONFIG	Parameters
hidden states	768
optimizer	Bert Adam
learning rate	[3e-5,2e-5,**1e-5**]
batch size	[16,32,64,**256**]
dropout	[**0.1**,0.2,0.4]
epochs	40
kernel size	[2,**3**,4]
char embeds	[768]

Table 7: The hyper-parameters of the conPrune model.

CONFIG	Parameters
char embeds	[**50**,100,200]
bi-gram embeds	[**50**,100,200]
hidden states	[128,**256**,**512**]
optimizer	Adam
learning rate	[0.01,**0.001**,0.002]
batch size	[**16**,**32**,64,256]
dropout	[0.1,**0.2**,0.4]
hidden layers	[1,2,**3**]
epochs	30

Table 8: The hyper-parameters of the LSTM student model.

where W^O, W_i^Q, W_i^K, W_i^V are trainable parameters. The layer normalization is adopted in the end of each multi-head Gaussian directional attention layer.

B Hyper-parameter Setting

To improve the reproducibility, we list all important hyper-parameters of the teacher model (Table 6), the student models (Table 7 and 8), the word-based NER models (Table 9) and the NMT model. We randomly pick 10% sentences from the training data as the development data for the tuning. In addition, we use the original development set of WMT-18 for MT. We utilize the uniform-sample to choose the hyper-parameters. In particular, we use the hyper-parameters of the modified Transformer model and the NMT model following previous studies (Vaswani et al., 2017; Duan and Zhao, 2020).

CONFIG	Parameters
input embeds	[50,100,**200**]
hidden states	[100,**200**,300]
optimizer	Adam
learning rate	[0.01,0.001,**0.005**]
batch size	[**16**,**32**,64,256]
dropout	[0.1,0.2,**0.5**]
hidden layers	[1,2,**3**]
epochs	30

Table 9: The hyper-parameters of the word-based NER model.

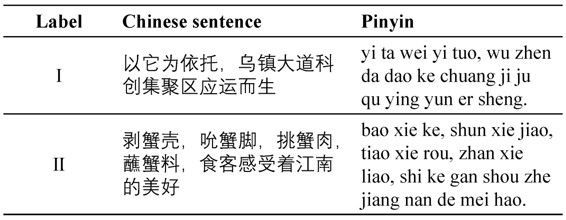

Label	Chinese sentence	Pinyin
I	以它为依托，乌镇大道科创集聚区应运而生	yi ta wei yi tuo, wu zhen da dao ke chuang ji ju qu ying yun er sheng.
II	剥蟹壳，吮蟹脚，挑蟹肉，蘸蟹料，食客感受着江南的美好	bao xie ke, shun xie jiao, tiao xie rou, zhan xie liao, shi ke gan shou zhe jiang nan de mei hao.

Figure 2: The Pinyin sequences for two Chinese sentences.

C Case Study

For NMT, it is difficult to analyze translation results as the interpretability of NMT is poor. We start with examples and focus on the differences between two translations with different segmentation results in addition to sentence-level BLEU scores. The Pinyin sequences for the two Chinese sentences are shown in Figure 2 and translations are shown in Table 10. We find that segmentation results with slight differences make translation results varying. The boundary of words may lead to these differences. There is a considerable discrepancy when the neural machine translation system stops training at different steps. That shows the neural machine translation system is unstable. It is full of uncertainty, and it still brings great challenges for the MT model itself and other crucial techniques.

Label	Model	Seg. Result	MT. Result	BLEU
I	Teacher	yi/ ta/ wei/ yi tuo/ ,/ **wu zhen da dao/** ke chuang/ ji ju qu/ ying yun er sheng	with it as its base, the koku district of wuzhen boulevard came into being.	21.79
	ConPrune†	yi/ ta/ wei/ yi tuo/ ,/ **wu zhen/ da dao/** ke chuang/ ji ju qu/ ying yun er sheng	to rely on it, wuzhen boulevard science set up the agglomeration area emerged at the moment.	**70.71**
II	Teacher	bao/ **xie ke/** ,/ shun/ **xie jiao/** ,/ tiao/ **xie rou/** ,/ zhan/ xie/ liao/ ,/ shi ke/ gan shou/ zhe/ shen/ chu/ jiang nan/ de/ mei hao	dipping crab shell, sucking crab foot, picking crab meat, dipping crab material, eating experience in the south of the good.	17.28
	ConPrune†	bao/ **xie/ ke/** ,/ shun/ **xie/ jiao/** ,/ tiao/ **xie/ rou/** ,/ zhan/ xie/ liao/ ,/ shi ke/ gan shou/ zhe/ shen/ chu/ jiang nan/ de/ mei hao	peeling crab shell, sucking crab feet, picking crab meat, dipping crab material, customers feel the good in the south of the river.	**27.08**

Table 10: The evaluation and results for MT. Seg. Result represents the segmentation results with different CWS methods. In particular, we use "/" to split the words.

Analysis of Tree-Structured Architectures for Code Generation

Samip Dahal **Adyasha Maharana** **Mohit Bansal**
Department of Computer Science
University of North Carolina at Chapel Hill
`{sdpmas, adyasha, mbansal}@cs.unc.edu`

Abstract

Code generation is the task of generating code snippets from input user specifications in natural language. Leveraging the linguistically-motivated hierarchical structure of the input can benefit code generation, especially since the specifications are complex sentences containing multiple variables and operations over various data structures. Moreover, recent advances in Transformer architectures have led to improved performance with tree-to-tree style generation for other seq2seq tasks e.g., machine translation. Hence, we present an empirical analysis of the significance of input parse trees for code generation. We run text-to-tree, linearized tree-to-tree, and structured tree-to-tree models, using constituency-based parse trees as input, where the target is Abstract Syntax Tree (AST) of the code. We evaluate our models on the Python-based code generation dataset CoNaLa and a semantic parsing dataset ATIS. We find that constituency trees encoded using a structure-aware model improve performance for both datasets. We also provide an analysis of those aspects of the input parse trees which are most impactful. For instance, we find that structure-aware encodings are better at modelling inputs with multiple variables and capturing long-range dependencies for code generation.[1]

1 Introduction

Code generation is the task of converting input user specifications written in natural language (NL) to code snippets in a target programming language. It is a task-driven variant of semantic parsing, which translates natural language input to formal machine-executable representation. Recent works have utilized the Abstract Syntax Tree (AST) - which is the syntactic tree representation of target source code - to generate better code snippets (Yin and Neubig, 2017, 2018; Sun et al., 2020; Rabinovich et al., 2017). The use of ASTs has achieved strong results but there has been relatively less work on utilizing the parse trees of the NL input. Constituency or dependency trees representing the syntactic structure of input can be leveraged to perform sub-tree alignment with corresponding AST of target code and benefit the downstream task. Hence, in this paper, we present several tree-to-tree models for the code generation task that convert the parse tree representation of NL input to AST representation of target source code. First, we base our model on the Transformer architecture (Vaswani et al., 2017). However, the standard Transformer is not designed to preserve the tree structure of the input parse trees. Hence, to better encode the trees, we modify a structure-aware Tree Transformer model (Nguyen et al., 2020) for the tree-to-tree code generation task. We focus on constituency-based parse trees in this paper because of space constraints as this is a short paper. Moreover, as pointed out by Nguyen et al. (2020), there is little evidence of constituency structures being learned implicitly in language models, whereas dependency structures have been shown to be implicitly embedded in models like BERT (Devlin et al., 2019; Hewitt and Manning, 2019). We evaluate our models on the CoNaLa dataset (Yin et al., 2018) and find that incorporating constituency parse trees in input using structure-aware encoders improves the quality of generated code. We further evaluate our models on the ATIS dataset (Hemphill et al., 1990), which translates natural language sentences into their lambda calculus logical forms and show that a structure-aware Transformer significantly improves performance over a standard Transformer.

We also focus on analyzing the input parse trees to find the aspects that benefit code generation. Our analysis comprises ablation experiments on our pro-

---

[1] Code available at: `https://github.com/sdpmas/TreeCodeGen`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4382–4391
August 1–6, 2021.

posed structure-aware model and pattern analysis of the output from different models with respect to the characteristics of input natural language specification. Specifically, we analyze the variation in performance with the presence of user-defined identifiers and variable entities in input sentence, and the complexity of input trees. We find that the structure-aware model improves performance when such identifiers and variables are present towards the end of the input sentences and when the input sentences are short in length.

2 Related Work

Code Generation. Code generation for general-purpose programming languages is a recent phenomenon, earlier works being focused on domain-specific languages (Gulwani and Marron, 2014; Raza et al., 2015). Recent works have mainly applied sequence-to-tree models for code generation, with the tree being the AST of target source code (Dong and Lapata, 2016; Yin and Neubig, 2017; Rabinovich et al., 2017; Yin and Neubig, 2018, 2019; Shin et al., 2019; Xu et al., 2020; Sun et al., 2020). While the use of ASTs for code generation has been substantially studied, to the best of our knowledge, the use of input parse tree for code generation is largely unexplored.

Semantic Parsing. Several methods have been proposed to parse natural language sentences to formal meaning representations like lambda calculus (Wong and Mooney, 2007), Alexa Meaning Representation Language (Kumar et al., 2017), Abstract Meaning Representations (AMR) (Banarescu et al., 2013), structured queries (Iyer et al., 2017; Yin and Neubig, 2018), etc. Many of the recent works for semantic parsing have focused on sequence-to-tree models leveraging tree structures like AST as the intermediate representation for target meaning representation (Yin and Neubig, 2018; Sun et al., 2020). Code generation can also be regarded as a form of semantic parsing where the target meaning representation is programming language snippet.

Source Trees and Structure-Aware Models. Several structure-aware tree-encoders have also been proposed to process the source trees (Chen et al., 2017a,b; Yang et al., 2017; Nguyen et al., 2020). While many of the tree-encoders are dependent on recurrent mechanism and hence are unparallelizable, Nguyen et al. (2020) propose a Transformer-based structure-aware model that is parallelizable. Concurrently, several tree-to-seq models have been proposed that leverage source syntactic trees for NLP tasks like machine translation (Eriguchi et al., 2016; Yang et al., 2017; Eriguchi et al., 2017; Chen et al., 2017b) and sentence modeling (Shi et al., 2018). There has been some work on leveraging hybrid tree - a joint tree-like representation of the NL sentence and corresponding meaning representation - for semantic parsing (Lu et al., 2008; Jie and Lu, 2018), while Harer et al. (2019) made use of source tree structures for code correction. However, the same is unexplored in the context of code generation. We study the use of tree-to-tree models for code generation and provide analysis of its various modules.

3 Our Models

3.1 Baseline (Sequence-to-Tree Model)

We use a standard Transformer model (Vaswani et al., 2017) as our natural language-to-code baseline. We build a sequence-to-tree model with a regular Transformer encoder and decoder. The encoder maps the source sequence $\mathbf{x} = x_1, x_2, ..., x_n$ to its vector representation $\hat{\mathbf{x}} = \hat{\mathbf{x_1}}, \hat{\mathbf{x_2}}, ..., \hat{\mathbf{x_n}}$, which is passed into the decoder. At each time step t, we linearize the AST generated till time step $t-1$ i.e. AST $y_{<t}$ and concatenate its embedding with the embedding of the corresponding parent actions, following Yin and Neubig (2017). Decoder takes this partial AST vector representation and source vector representation $\hat{\mathbf{x}}$ from encoder as input and expands the frontier non-terminal node of the partial AST. Here, ASTs are linearized by the pre-order depth-first traversal and the expansion of the AST, at each time step, is constrained by the grammar rules of the underlying programming language. We adopt the ASDL grammar and transition system (Yin and Neubig, 2018) that decomposes the production of an AST into a sequence of actions. At each time step t, the action a_t can be of 3 types (see Appendix for details). Given the input specification $\mathbf{x}$, the probability of generating an AST $\mathbf{y}$ can be expressed in terms of probabilities of generating corresponding actions: $p(\mathbf{y} \mid \mathbf{x}) = \prod_t p(a_t \mid \mathbf{x}, y_{<t})$. Here, a_t is the action at time step t and $y_{<t}$ is the partial AST generated upto time step t. We also use a pointer network (Vinyals et al., 2015) to allow the model to copy relevant entities from input sequence while generating a terminal AST node.

3.2 Linearized Tree-to-Tree Model

Here, we use the identical model architecture as our baseline (see Sec. 3.1) but we replace the input NL sequence with its linearized constituency-based parse tree. Constituency trees aim to describe syntactic structure of the sentence by dividing it into sub-phrases. As discussed in Sec. 1, this structural information can promote alignment between source and target sub-trees (AST), thereby improving downstream generation task. In our model, constituency trees are linearized by the pre-order depth-first traversal (see Fig. 5 in Appendix). Our output is the AST representation of code.

3.3 Structured Tree-to-Tree Model with Hierarchical Accumulation

A standard Transformer encoder (see Sec. 3.1) is not designed to process the structural information of input parse trees. On the other hand, many tree-based models have been proposed in the past to process the structural information (Chen et al., 2018; Eriguchi et al., 2016; Rao et al., 2019) but most of them are based on recurrent mechanism and hence, not parallelizable like Transformer-based models. This observation motivated us to build a Transformer-based structure-aware tree-to-tree model. In this paper, we adapt Tree Transformer, an attention-based tree-to-tree model with hierarchical accumulation proposed by Nguyen et al. (2020), for code generation. Hierarchical accumulation aims to encode the tree by performing a series of operations including upward cumulative-average and weighted aggregation on the interpolated tree matrix. Furthermore, the model includes hierarchical embeddings to induce biases that reflect hierarchy within each branch of the tree and among the siblings within a subtree. Finally, subtree masking is used to filter out irrelevant information during upward cumulative-average and weighted aggregation operations. In this model, our target is identical to that of our baseline i.e., the AST representation of the source code, which is later converted to source code with the help of the transition system. We linearize the AST in the same fashion as our baseline, concatenate it with the corresponding parent actions vector in the hidden dimension and pass it into the decoder along with the leaves and nodes vector representations from the encoder. We also add a pointer network (Vinyals et al., 2015) to allow the model to copy from leaves of input parse tree while generating a terminal AST node. Without pointer network, our model architecture is identical to the Tree Transformer, so we refer the reader to Nguyen et al. (2020) for a complete description of the model architecture.

Method	**BLEU**
Xu et al. (2020)	27.2
Baseline: Text-to-Tree	28.13
Model 1: Linearized Constituency Tree-to-Tree	27.71
Model 2: Structured Constituency Tree-to-Tree	**30.30**

Table 1: Results on the test set of CoNaLa dataset.

4 Experimental Setup

Dataset. We evaluate each of the models described in Sec. 3 on the CoNaLa (Yin et al., 2018) and ATIS (Hemphill et al., 1990) datasets. The CoNaLa dataset contains 2379 manually curated intent-snippet pairs for training (200 of which we use for validation) and 500 pairs for test. Although the CoNaLa dataset consists of 600k additional mined intent-snippet pairs, we train our models only on the manually-curated training dataset and compare them with Xu et al. (2020) model trained on the same dataset and without the use of reranker. The ATIS dataset consists of 4434, 491 and 448 pairs for training, validation, and test respectively. Following previous works, we use corpus-level BLEU-4 and exact-match accuracy metrics for evaluation on CoNaLa and ATIS datasets respectively. See Appendix for details on training and inference.

5 Results

Table 1 shows BLEU scores from our experiments on the CoNaLa dataset. Our baseline Transformer model outperforms previous state-of-the-art LSTM-based model (Xu et al., 2020) by 0.93 BLEU points. The linearized constituency tree-to-tree model hinders the BLEU score compared to our baseline. However, the structured constituency tree-to-tree model significantly outperforms baseline by 2.17 ($p<0.01$)[2] BLEU points and linearized constituency tree-to-tree model by 2.59 ($p<0.01$) BLEU points. It also outperforms the baseline model by 8% in terms of **human-evaluated** code quality (see Appendix). This suggests that the structured inputs can provide important cues for generating high quality code snippets through structure-aware encodings. This information is lost when trees are converted to linearized inputs, thereby

---

[2] Statistical significance is computed with 100K samples using bootstrap (Noreen, 1989; Tibshirani and Efron, 1993).

Method	Acc.
TRANX (Yin and Neubig, 2018)	86.2
TreeGen (Sun et al., 2020)	**89.1**
Baseline: Text-to-Tree	75.89
Model 1: Linearized Constituency Tree-to-Tree	53.57
Model 2: Structured Constituency Tree-to-Tree	86.83

Table 2: Results on the test set of ATIS dataset.

Method	CoNaLa
Structured Constituency Tree to Tree	**33.76**
- Subtree Masking	31.76
- Hierarchical Embeddings	32.62

Table 3: Ablation results for the structured constituency tree-to-tree model on the validation set of CoNaLa dataset. BLEU-4 metric is used to evaluate predictions.

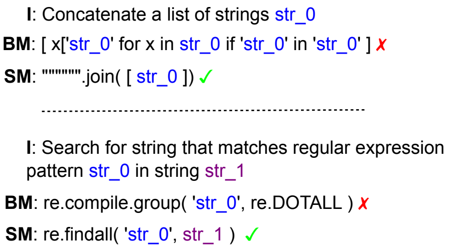

I: Concatenate a list of strings str_0

BM: `[ x['str_0' for x in str_0 if 'str_0' in 'str_0' ]` ✗

SM: `"""""""".join( [ str_0 ])` ✓

I: Search for string that matches regular expression pattern str_0 in string str_1

BM: `re.compile.group( 'str_0', re.DOTALL )` ✗

SM: `re.findall( 'str_0', str_1 )` ✓

Figure 1: Sample predictions of our models when quoted string(s) appear towards the end of input sentence. These strings (highlighted) are replaced by placeholders. (I=Input, BM=Baseline Model, SM=Structured Model)

leading to a drop in performance over the text-to-tree baseline. It is important to note that our models have significantly higher number of parameters compared to (Xu et al., 2020) (roughly 44-49M for our models vs 2M for their model) as their model consists of only one layer of LSTM. However, we ran the LSTM model with higher number of parameters by increasing the embedding dimensions and the number of hidden layers in encoder LSTM and we did not see significant improvement in BLEU score. This indicates that the superior performance of our models is primarily due to the rather than the increased count of learnable parameters.

Table 2 shows accuracy scores from our experiments on the ATIS dataset. Our baseline model performs significantly worse than the LSTM-based TRANX (Yin and Neubig, 2018) and the accuracy further drops with the linearized constituency tree-to-tree model. Our structured model, however, performs significantly better than the aforementioned models ($p<0.01$), a trend we observed in results on the CoNaLa dataset as well. The accuracy of the structured model is still slightly worse than the TreeGen model (Sun et al., 2020). This might be because the TreeGen model consists of an AST reader which encodes the partial code tree generated in previous timesteps using structure-aware tree convolutions, during generation at each timestep. Our model lacks such a module for the target AST. Nonetheless, the overall trend among our three models suggests that parse trees benefit semantic parsing as long as their structure is incorporated in the model. However, if this extra hierarchical information is encoded in a linear fashion, it results in negative contribution to semantic parsing (row 4 in Table 2). Overall, our results also provide motivation for joint modelling of both, input and output parse trees, for semantic parsing.

6 Analysis

6.1 Ablation Tests

We ablate our best model to understand the effect of the various modules in Tree Transformer on target task and present results in Table 3. First, we remove subtree masking which allows each node of the tree to attend over nodes that are not in the subtree rooted at that node in hierarchical accumulation. Second, we remove the use of hierarchical embeddings in our model. On the CoNaLa dataset, both experiments result in negative impact on the model's performance. This suggests that subtree masking is a crucial mechanism for structure-aware encoding i.e, for each node in the parse tree, only the relevant information within the subtree rooted at the node is useful. Comparatively, the results show that subtree masking is more important than hierarchical embeddings.

6.2 Pattern Analysis

Following Yin and Neubig (2017) and Xu et al. (2020), we next analyze the input intents and the corresponding code generated by the baseline model and the structured model (on a subset of test samples of CoNaLa datset) to find recurring patterns. First, we observe that input specifications in CoNaLa dataset contain quoted strings, which often occur as user-defined identifiers or strings in generated code as well. We find that when these quoted strings appear towards the end of the input sentence, the difference in quality of output code by the two models in terms of average BLEU score is higher than usual i.e., more than 5 BLEU points (row 2 of Table 4). Moreover, when the input sentence contains two or more quoted strings,

I: Create list var_0 containing 100 instances of object var_1
BM: var_0 = [(var_1 + var_0) for var_1, in 100(var_0)] ✗
SM: var_0 = [var_1() for _ in range(100)] ✓

……………………………………………………

I: Convert string var_0 into a list of integers var_1
BM: var_1 = [int(x) for x in 'var_1'.split(var_0)] ✗
SM: [int(i) for i in var_0.split()] ✓

Figure 2: Sample predictions of our models when multiple quoted strings appear in the input sentence. (I=Input, BM=Baseline Model, SM=Structured Model)

Pattern	Baseline	Structured
All Intents	20.23	**23.74**
Ending with quoted string	20.79	**26.42**
Multiple quoted strings	23.55	**29.20**
No quoted strings	10.04	**12.42**

Table 4: Comparison of average BLEU scores of baseline model and structured model in relation to different characteristics of input intents of CoNaLa dataset. Results are shown on the test set.

the baseline model often fails to capture the semantic relationship between those strings in the output code resulting in lower BLEU scores (row 3 of Table 4). However, in the absence of any quoted strings, the structure-aware model does better than the baseline by only 2 BLEU points (row 4 of Table 4). This shows that the structured input, when paired with a structure-aware encoder, helps capture dependencies between semantic units. Fig. 1 and Fig. 2 provide examples of both these scenarios and Table 4 compares the average BLEU scores.

Similarly, we notice that there are variable entities like city, airline, airport, time, etc. in the input specifications, which also appear in the corresponding outputs in the ATIS dataset. We find that our structure-aware model outperforms the baseline model by 12.57 points when such variables occur at the end of the input sentence (see row 3 in Table 6), suggesting that the model is able to capture long-term dependencies (see Fig. 8).

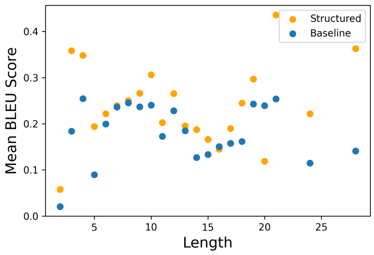

Figure 3: Plot of input intent length vs. mean BLEU score for our baseline and structured model on the test set of CoNaLa dataset.

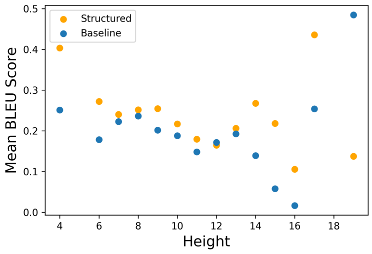

Figure 4: Plot of height of input parse tree vs. mean BLEU score for our baseline and structured model on the test set of CoNaLa dataset.

6.3 Comparison Based on Input Complexity

We compare the performance of the baseline text-to-tree and structured tree-to-tree models w.r.t. input complexity i.e. the length of input sentences and height of the input parse trees in the test set of CoNaLa dataset. The variation of mean BLEU scores w.r.t. length of input sentence and height of input trees is shown in Figures 3 and 4 respectively. In both figures, we observe that the structure-aware model outperforms baseline by wider margins for inputs of shorter length and height. Similarly, there are smaller but consistent improvements for inputs of medium complexity. The margins are largest for samples of high complexity, but this observation is supported by relatively few data points (see scatter plots in Appendix). From these results, we infer that the structured model is particularly helpful for short input sentences or parse trees in code generation. Similarly, the structured model significantly outperforms the baseline for shorter intent lengths in the ATIS dataset. However, we did not find any clear linkage between the height of input tree and the performance of our models on the ATIS dataset (see Figures 9 and 10 in Appendix).

7 Conclusion

We experimented with models to utilize input constituency parse trees for code generation and semantic parsing. Our tree-to-tree model significantly outperforms other approaches for code generation and is competitive for semantic parsing. We find that the hierarchical structure of parse trees helps the structure-aware model capture semantic relationships between user-defined identifiers and variable entities in the input intent.

Acknowledgments

We thank Hyounghun Kim and the reviewers for their useful feedback. This work was supported by NSF-CAREER Award 1846185 and a Microsoft Investigator Fellowship.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017a. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

Kehai Chen, Rui Wang, Masao Utiyama, Lemao Liu, Akihiro Tamura, Eiichiro Sumita, and Tiejun Zhao. 2017b. Neural machine translation with source dependency representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2846–2852, Copenhagen, Denmark. Association for Computational Linguistics.

Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-tree neural networks for program translation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2552–2562, Red Hook, NY, USA. Curran Associates Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada. Association for Computational Linguistics.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 803–814.

Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-transformer: A transformer-based method for correction of tree-structured data. *arXiv preprint arXiv:1908.00449*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Zhanming Jie and Wei Lu. 2018. Dependency-based hybrid trees for semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2441, Brussels, Belgium. Association for Computational Linguistics.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, Markus Dreyer, Stanislav Peshterliev, Ankur Gandhe, Denis Filiminov, Ariya Rastrow, et al. 2017. Just ask: Building an architecture for extensible self-service spoken language understanding. In *1st Workshop on Conversational AI at the Conference on Advances in Neural Information Processing Systems (NIPS) 2017*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2020. Tree-structured attention with hierarchical accumulation. In *International Conference on Learning Representations*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Jinfeng Rao, Kartikeya Upasani, Anusha Balakrishnan, Michael White, Anuj Kumar, and Rajen Subba. 2019. A tree-to-sequence model for neural NLG in task-oriented dialog. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 95–100, Tokyo, Japan. Association for Computational Linguistics.

Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. 2015. Compositional program synthesis from natural language and examples. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4631–4641, Brussels, Belgium. Association for Computational Linguistics.

Eui Chul Shin, Miltiadis Allamanis, Marc Brockschmidt, and Alex Polozov. 2019. Program synthesis and semantic parsing with learned code idioms. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, and Lu Zhang. 2020. Treegen: A tree-based transformer architecture for code generation. In *AAAI*, pages 8984–8991.

Robert J Tibshirani and Bradley Efron. 1993. An introduction to the bootstrap. *Monographs on statistics and applied probability*, 57:1–436.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

Frank F. Xu, Zhengbao Jiang, Pengcheng Yin, Bogdan Vasilescu, and Graham Neubig. 2020. Incorporating external knowledge through pre-training for natural language to code generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6045–6052, Online. Association for Computational Linguistics.

Baosong Yang, Derek F. Wong, Tong Xiao, Lidia S. Chao, and Jingbo Zhu. 2017. Towards bidirectional hierarchical representations for attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1432–1441, Copenhagen, Denmark. Association for Computational Linguistics.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *International Conference on Mining Software Repositories*, MSR, pages 476–486. ACM.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4553–4559.

Appendices

A Our Models

A.1 Baseline (Sequence-to-Tree Model)

We use a standard Transformer model (Vaswani et al., 2017) as our natural language-to-code baseline. We build a sequence-to-tree model with a regular Transformer encoder and decoder. The encoder maps the source sequence $\mathbf{x} = x_1, x_2, ..., x_n$ to its vector representation $\hat{\mathbf{x}} = \hat{\mathbf{x_1}}, \hat{\mathbf{x_2}}, ..., \hat{\mathbf{x_n}}$, which is passed into the decoder. At each time step t, we linearize the AST generated till time step $t-1$ i.e. AST $y_{<t}$ and concatenate its embedding with the embedding of the corresponding parent actions, following Yin and Neubig (2017). The decoder takes this partial AST vector representation and the source vector representation $\hat{\mathbf{x}}$ from encoder as input and expands the frontier non-terminal node of the partial AST. Here, the ASTs are linearized by the pre-order depth-first traversal and the expansion of the AST, at each time step, is constrained by the grammar rules of the underlying programming language. We adopt the ASDL grammar for Python and transition system (Yin and Neubig, 2018) that decomposes the production of an AST into a sequence of actions. At each time step t, the action a_t can be of 3 types:

- *ApplyRule Action*: Applies production rule R to the partial AST.
- *Reduce Action*: Denotes the completion of a field with optional or multiple cardinalities.
- *GenToken Action*: Expands a terminal node by generating a leaf token.

Given the input specification $\mathbf{x}$, the probability of generating an AST $\mathbf{y}$ can be expressed in terms of the probabilities of generating the corresponding actions:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_t p(a_t \mid \mathbf{x}, y_{<t}) \qquad (1)$$

Here, a_t is the action at time step t and $y_{<t}$ is the partial AST generated upto time step t. We also use a pointer network (Vinyals et al., 2015) to allow the model to copy relevant entities from input sequence while generating a terminal AST node with GenToken Action.

B Experimental Setup

Training and Inference. All of our models have 6 encoder layers and 6 decoder layers. Our models are trained on GPUs using Google Colab and each model takes 2-3 hours for a single run. We perform manual hyperparameter tuning, using 4-5 runs for each model. We tried learning rates within the range [1e-4, 5e-5]. After manual tuning, for CoNaLa dataset, we trained all the models using learning rate of 1e-4. For ATIS dataset, we use learning rate of 2e-4 for our baseline model, 5e-5 for our Linearized Tree-to-Tree model and 4e-5 for our structured model. We use batch size of 64.

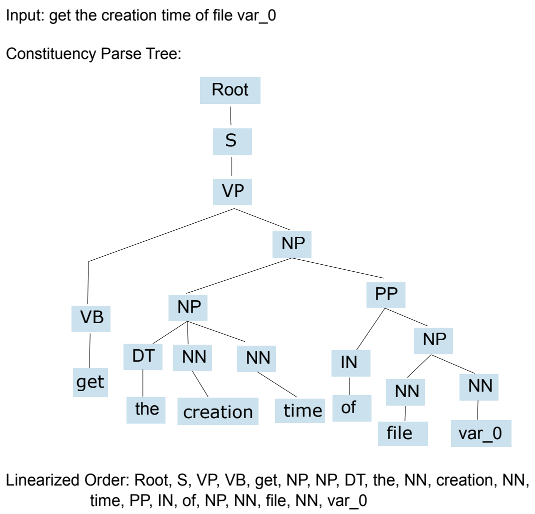

Figure 5: Constituency parse tree of natural language specification *get the creation time of file var_0* and its linearized form. Words from the input specification are leaves of the tree.

Models	**Wins**	**Loses**	**Tie**
Structured T2T vs. Baseline	35%	27%	38%

Table 5: Results from human evaluation of generated code. Wins and Loses refer to the %times code generated from structured tree-to-tree model was chosen over those from baseline model.

We parse source text into constituency trees using Stanford CoreNLP parser (Manning et al., 2014). During inference, we use beam search with beam size of 30 for CoNaLa dataset and beam size of 1 for ATIS dataset to predict the output AST for a given natural language intent. We begin the beam search with one AST initialized with the root node and run until maximum time-step T or until we find K complete ASTs, where K is the beam-size. The maximum time-step is set to 200.

C Results

C.1 Human Evaluation

We also perform human evaluation of 100 samples from the CoNaLa dataset (see Table 5). The annotator (non-coauthor graduate student, proficient in Python) was instructed to pick the better code output for a given input specification. The samples contained shuffled outputs from our baseline and structure-aware models. Outputs from structure-aware model were preferred 35% of the times while those from the baseline were preferred 27% of the times and rest of the instances ended in a tie over code quality.

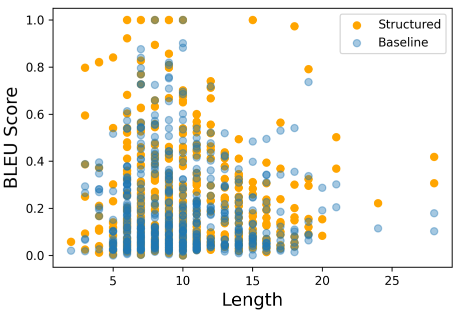

Figure 6: Scatter plot of input length vs. BLEU score for samples from the test set of CoNaLa dataset.

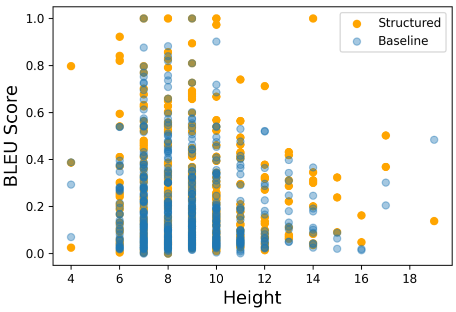

Figure 7: Scatter plot of height of input parse tree vs. BLEU score for samples from the test set of CoNaLa dataset.

C.2 Scatter Plots

We present two scatter plots for demonstrating the effect of input complexity on model performance for the CoNaLa dataset. Fig. 6 compares sentence-level BLEU score of predictions from our baseline and structured models against the length of input sentences. Similarly, Fig. 7 compares sentence-level BLEU score of predictions from our baseline and structured models against the height of input parse trees. Both of these comparisons are performed on the test set of CoNaLa dataset.

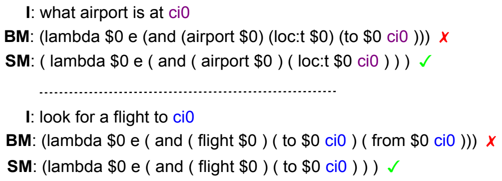

```
I: what airport is at ci0
BM: (lambda $0 e (and (airport $0) (loc:t $0) (to $0 ci0 ))) ✗
SM: ( lambda $0 e ( and ( airport $0 ) ( loc:t $0 ci0 ) ) ) ✓
.........................................................................
I: look for a flight to ci0
BM: (lambda $0 e ( and ( flight $0 ) ( to $0 ci0 ) ( from $0 ci0 ))) ✗
SM: (lambda $0 e ( and ( flight $0 ) ( to $0 ci0 ) ) ) ✓
```

Figure 8: Outputs of our baseline and structured model for the ATIS dataset when variable entities appear at the end of input sentences.

Pattern	Baseline	Structured
All Intents	75.89	**86.83**
Not ending with a variable	74.52	**80.18**
Ending with a variable	76.31	**88.88**

Table 6: Comparison of accuracies of baseline model and structured model in relation to different characteristics of input intents of ATIS dataset. Results are shown on test set.

C.3 Pattern Analysis

Following Yin and Neubig (2017) and Xu et al. (2020), we next analyze the input intents and the corresponding code generated by the baseline model and our best model (on a small test sample), i.e., Structured Constituency Tree-to-Tree to find recurring patterns. We observe that the code generated by the structured model is significantly better for input intents containing certain characteristics. First, we observe that input specifications in CoNaLa dataset contain quoted strings (see placeholder `str_0` in Fig. 1 of the main text). These strings often occur as user-defined identifiers in the input sentence and as strings in generated code as well. We find that when these quoted strings appear towards the end of the input sentence, the difference in quality of output code by the two models in terms of average BLEU score is higher than usual i.e., more than 5 BLEU points (see row 2 in Table 4). We also find that when the input sentence contains two or more quoted strings, the baseline model often fails to capture the semantic relationship between those strings in the output code, resulting in lower BLEU scores. However, the structure-aware model succeeds at the task, resulting in higher BLEU scores (see row 3 in Table 4). In the absence of any quoted strings, the structure-aware model does better than the baseline by only 2 BLEU points (see row 4 in Table 4). This shows that when the structured input is paired with a structure-aware encoder, it helps capture the semantic relationships between multiple units. Fig. 1 and Fig. 2 provide examples of both these scenarios and Table 4 compares the average BLEU score of all the examples in the test set with 1) quoted string at the end of input sentence, 2) two or more quoted strings and 3) zero quoted strings,

Similarly, we notice that there are variable entities like city (`ci0` in Fig. 8), airline, airport, time, etc in the input specifications, which also appear in the corresponding outputs in the ATIS dataset. Such variables are anonymized with identifiers of same type following (Dong and Lapata, 2016). We find that when such variables occur at the end of the input sentence, our structured model does signifi-

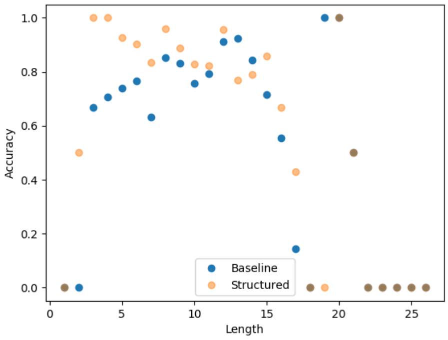

Figure 9: Plot of length of inputs vs accuracy on the test set of ATIS dataset.

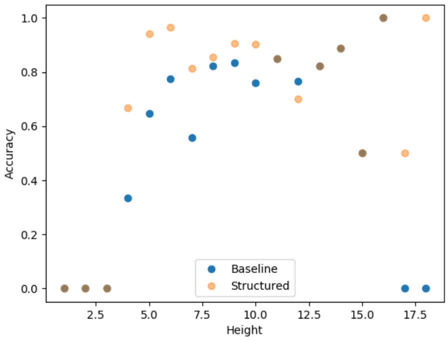

Figure 10: Plot of height of input parse trees vs accuracy on the test set of ATIS dataset.

cantly better than our baseline (see row 3 in Table 6) but the difference decreases in cases where the variables don't occur at the end of the input sentence (see row 2 in Table 6). Fig. 8 provides examples of the cases where variable entities occur at the end of the input sentences.

C.4 Comparison Based on Input Complexity

We compare the performance of our two models with respect to the complexity of input sentences. We rank the complexity of an input sentence by its length and the height of the corresponding parse tree i.e., the longest length of the path from the root node of the tree to its leaves. Firstly, we do this analysis on CoNaLa dataset. Fig. 3 presents a plot of length of input sentences and mean BLEU scores of generated code snippets. Fig. 4 presents a plot of height of input trees and mean BLEU scores of generated code snippets. Similarly, Figures 6 and 7 are scatter plots of sentence-level BLEU scores for generated code snipppets vs. length of input sentences and height of input parse trees respectively. We can see in both Fig. 3 and 4 that there is a wider gap between mean BLEU score of our structured model and baseline in the beginning, with the structured model performing significantly better. The gap narrows in the middle and widens towards the end. However, as we can see from the Figures 6 and 7, there are very few data points towards the end to draw any conclusion. From these observations, we infer that for code generation, structured model is particularly helpful for short input sentences or for short input parse trees. Similarly, the structured model significantly outperforms the baseline for shorter intent lengths in the ATIS dataset. However, we did not find any clear linkage between the height of input sentences and performance of our models for semantic parsing with the ATIS dataset. Fig 9 plots length of input sentences against accuracy and Fig 10 plots height of input parse trees against accuracy of outputs for samples in the test set of ATIS dataset.

How Does Distilled Data Complexity Impact the Quality and Confidence of Non-Autoregressive Machine Translation?

Weijia Xu[*]† **Shuming Ma**‡ **Dongdong Zhang**‡ **Marine Carpuat**†
†Department of Computer Science, University of Maryland
‡Microsoft Research Asia
`{weijia, marine}@cs.umd.edu, {shumma, dozhang}@microsoft.com`

Abstract

While non-autoregressive (NAR) models are showing great promise for machine translation (MT), their use is limited by their dependence on knowledge distillation from autoregressive models. To address this issue, we seek to understand why distillation is so effective. Prior work suggests that distilled training data is less complex than manual translations. Based on experiments with the Levenshtein Transformer and the Mask-Predict NAR models on the WMT14 German-English task, this paper shows that different types of complexity have different impacts: while reducing lexical diversity and decreasing reordering complexity both help NAR learn better alignment between source and target, and thus improve translation quality, lexical diversity is the main reason why distillation increases model confidence, which affects the calibration of different NAR models differently.

1 Introduction and Background

When training NAR models for neural machine translation (NMT), sequence-level knowledge distillation (Kim and Rush, 2016) is key to match the translation quality of autoregressive (AR) models (Gu et al., 2018; Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019). Knowledge distillation was first proposed to obtain small *student* models that match the quality of a higher-capacity *teacher* models (Liang et al., 2008; Hinton et al., 2015). Sequence-level knowledge distillation (SLKD) trains the student model $p(\boldsymbol{y} \mid \boldsymbol{x})$ to approximate the teacher distribution $q(\boldsymbol{y} \mid \boldsymbol{x})$ by maximizing the following objective: $\mathcal{L}_{\text{SEQ-KD}} = -\sum_{\boldsymbol{y} \in \mathcal{Y}} q(\boldsymbol{y} \mid \boldsymbol{x}) \log p(\boldsymbol{y} \mid \boldsymbol{x}) \approx -\sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{1}\left[\boldsymbol{y} = \hat{\boldsymbol{y}}\right] \log p(\boldsymbol{y} \mid \boldsymbol{x})$, where $\mathcal{Y}$ represents the space of all possible target sequences, and $\hat{\boldsymbol{y}}$ is the output from running beam search with the teacher model q.

However, we do not yet have a clear picture for how SLKD impacts NAR training. Ren et al. (2020) show that SLKD reduces the degree of dependency between target tokens. Gu et al. (2018) hypothesize that SLKD reduces the number of modes in the output distribution (alternative translations for a source). This hypothesis was supported by experiments that use multiway parallel data to simulate the modes (Zhou et al., 2019). Zhou et al. (2019) also investigate the impact of data complexity on NAR translation quality – they generate distilled data of varying complexity with AR models of different capacity and show that higher-capacity NAR models require more complex distilled data to achieve better translation quality. They further show that generating distilled references with mixture of experts (Shen et al., 2019) improves NAR translation quality. However, training samples can be complex in different ways, and it remains unclear how different types of data complexity alter the internal working of NAR models and their translation quality. We also anticipate that data complexity may impact the uncertainty and calibration of NAR models – an understudied question, unlike for AR models (Ott et al., 2018; Wang et al., 2020).

This paper focuses on two types of data complexity – lexical diversity and degree of word reordering. We expose two state-of-the-art NAR models (Mask-Predict (Ghazvininejad et al., 2019) and Levenshtein Transformer (Gu et al., 2019)) to distilled references of varying complexity on the WMT14 German-English task. Experiments show that decreasing reordering complexity and reducing lexical diversity via distillation both help NAR models learn better alignment between source and target and thus improve translation quality. Further analysis shows that knowledge distillation lowers model uncertainty by reducing lexical diversity, which affects the calibration of Mask-Predict and Levenshtein Transformer models in opposite directions.

---

[*] Work done during internship at Microsoft Research Asia.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4392–4400
August 1–6, 2021. ©2021 Association for Computational Linguistics

2 Generating Diverse Distilled References

We measure **distilled corpus complexity** with:

- **Word Reordering Degree** computed by the average fuzzy reordering score (FRS) (Talbot et al., 2011) over all sentence pairs. FRS is an MT evaluation metric introduced to distinguish significant changes in reordering rules of MT systems on syntactically distant language pairs. A higher FRS indicates that the hypothesis is more monotonically aligned to the source. Zhou et al. (2019) show that distilled data has a higher FRS than the real data which may benefit NAR models.
- **Lexical Diversity** which captures the diversity of target word choices given a source word. We compute the lexical diversity $LD(d)$ of the distilled corpus d by averaging the entropy of target words y conditioned on a source word x (Zhou et al., 2019): $LD(d) = \frac{1}{|\mathcal{V}_x|} \sum_{x \in \mathcal{V}_x} \mathbb{H}\left[y \,|\, x\right]$, where $\mathcal{V}_x$ denotes the source vocabulary.

To isolate the impact of complexity factors, we seek to control the **faithfulness** $F(d)$ of the distilled data d to the real parallel data r. We compute it as the KL-divergence of the alignment distribution between the real data r and the distilled data d (Zhou et al., 2019): $F(d) = \frac{1}{|\mathcal{V}_x|} \sum_{x \in \mathcal{V}_x} D_{\mathrm{KL}}\left[\, p_r(y \,|\, x) || \, p_d(y \,|\, x)\right]$.

Distilled Sample Generation To encourage diversity according to the corpus-level metrics above, we select distilled references for each source from the k-best list of AR hypotheses,[1] using instantiations of the following score:

$$\text{score}(\hat{\boldsymbol{y}}|\boldsymbol{x}, \boldsymbol{y}) = \lambda \, \text{sim}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + (1 - \lambda) \, \text{cxty}(\hat{\boldsymbol{y}}, \boldsymbol{x})$$

where the similarity $\text{sim}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ measures how faithful the hypothesis $\hat{\boldsymbol{y}}$ is to the original reference $\boldsymbol{y}$ and the complexity $\text{cxty}(\hat{\boldsymbol{y}}, \boldsymbol{x})$ captures the relationship between the target sequence $\hat{\boldsymbol{y}}$ and source sequence $\boldsymbol{x}$. The similarity function is the smoothed sentence-level BLEU (Chen and Cherry, 2014) w.r.t the original reference. We use three different complexity functions: 1) **FRS**, 2) **word-alignment score**[2] that measures complexity on a word level, and 3) **NMT score**[3] that measures complexity on a sentence level.

	Real	Distill	Δ
Original	24.2	26.6	+2.4
Reordered	30.0	29.4	−0.6

Table 1: BLEU scores on the original WMT14 En-De and the synthetic reordered version. For each task, we compare LevT models trained on real vs. distilled data.

3 Experimental Settings

Set-Up We use En-De and De-En datasets from WMT14 (Bojar et al., 2014) with the same preprocessing steps as Gu et al. (2019). We evaluate translation quality with case-sensitive tokenized BLEU,[4] using the Moses tokenizer.

Models We use two state-of-the-art NAR models:

- **Mask-Predict (MaskT)** (Ghazvininejad et al., 2019) uses a masked language model (Devlin et al., 2019) to generate the target sequence by iteratively masking out and regenerating the subset of tokens that the model is least confident about.
- **Levenshtein Transformer (LevT)** (Gu et al., 2019) generates the target sequence through iterative insertion and deletion steps.

All AR and NAR models adopt the *base* Transformer architecture (Vaswani et al., 2017). We train all models using a batch size of $64,800$ tokens for maximum $300,000$ steps and select the best checkpoint based on validation perplexity (see Appendix for details). During inference, we set the maximum number of iterations to 10. All word alignments in this paper are generated automatically using *fast-align* (Dyer et al., 2013).[5]

4 Preliminary: SLKD Helps NAR Learn Word Alignment

Our work is motivated by the hypothesis that SLKD helps NAR models learn (implicit) alignment between source and target words. We first test

---

[1] This is inspired by sequence-level interpolation (Kim and Rush, 2016), but they select hypothesis using BLEU while we use more diverse criteria. We use beam search with $k = 32$.

[2] Sum of the log probabilities of each target word conditioned on its aligned source words given by *fast-align*.

[3] Log probability of the target sentence conditioned on the source given by an AR model.

[4] `https://github.com/pytorch/fairseq/blob/master/fairseq/clib/libbleu/libbleu.cpp`

[5] This might introduce alignment errors leading to lower absolute FRS scores than with if we had access to gold manual alignments. However, this measurement noise is unlikely to impact our findings because 1) it is likely to be small on distilled data generated by autoregressive NMT models, which should be easier to align than original translations, and 2) distilled data versions are expected to be impacted uniformly.

	Data Property			BLEU ↑	
Data Version	FRS	LexDiv	Faith	MaskT	LevT
Real	0.46	0.36	0.0	28.0	27.6
Distilled	0.55	0.18	7.9	29.6	30.6
selection via BLEU					
+0.5 NMT	0.55	0.17	7.6	29.5	30.6
+0.5 w-align	0.57	0.18	7.6	29.2	30.1 ↓
+0.5 FRS	0.61	0.19	7.6	28.8 ↓	29.6 ↓
selection via BLEU					
+0.2 NMT	0.55	0.17	7.8	29.2	30.4
+0.2 w-align	0.58	0.18	7.9	28.7 ↓	30.0 ↓
+0.2 FRS	0.64	0.19	7.8	28.5 ↓	29.7 ↓

Table 2: Translation quality on WMT14 De-En. In the bottom two groups, models are trained on distilled data with similar faithfulness (*Faith*) but varying degree of reordering (FRS) and lexical diversity (*LexDiv*). ↓ marks significant drops compared to the first row in each group based on the paired bootstrap test at $p < 0.05$ (Clark et al., 2011).

this hypothesis by evaluating the effect of SLKD on two datasets: a) En-De train/dev/test sets from WMT14, and b) a synthetic version of the same task, where word alignment information is embedded by pre-reordering the source words so that they are monotonically aligned with target words (in train/dev/test sets).

While SLKD improves BLEU by +2.4 on the original En-De task, it has no benefit on the synthetic task (Table 1). This supports our hypothesis and is consistent with other findings on real data: Ghazvininejad et al. (2019) and Gu et al. (2019) showed that SLKD improves the quality of NAR models more on syntactically distant language pairs such as German-English than on Romanian-English. Furthermore, Ran et al. (2019) showed that automatically pre-reordering the source words improves the translation quality of NAR models. However, unlike in our experiment, SLKD is still needed in real translation scenarios, as exactly pre-ordering the source is not feasible at test time. Thus, we turn to understanding how distilled data helps NAR models on real translation tasks.

5 Reduced Lexical Diversity in SLKD Improves Translation Quality

We have shown that, similar to the effect of pre-reordering, SLKD benefits NAR training by reducing the difficulty of learning the source-target alignment. However, apart from the word reordering degree, reducing the lexical diversity on the target side can also reduce the difficulty of learning the alignment. In this section, we investigate how the two types of data complexity affect how well NAR models capture the source-target alignment, and therefore translation quality.

	Acc	Conf	ECE ↓
AR Transformer	63.9	72.3	10.34
MaskT w/o SLKD	63.7	74.2	10.49
MaskT w/ SLKD	65.1	86.5	21.41
LevT w/o SLKD	66.8	53.3	20.26
LevT w/ SLKD	65.9	71.3	15.17

Table 3: Average token-level accuracy (*Acc*), confidence (*Conf*), and inference ECE (ECE) of AR and the two NAR models trained with and without SLKD.

SLKD impacts both complexity types: the first two rows of Table 2 show that SLKD increases FRS by +0.09, reduces lexical diversity by −0.18, and boosts the BLEU of MaskT and LevT by 1.6–3.0 over their counterparts trained on real data.

We then compare NAR models trained on distilled data with varying degree of reordering and lexical diversity while controlling for faithfulness (2nd and 3rd group of rows in Table 2). While the absolute BLEU deltas are small, BLEU decreases significantly as the lexical diversity increases despite reduced degree of reordering. This indicates that increased lexical diversity prevails over the effect of lower degree of reordering in decreasing BLEU scores.

6 SLKD Increases Confidence of Source-Target Attention

To better understand how SLKD helps NAR learn the alignment between source and target, we measure how the *confidence* of the source-target attention changes over decoding iterations. Following Voita et al. (2019), we define the *confidence* of attention heads as the average of the maximum attention weights over source tokens, where the average is taken over target tokens. Higher confidence scores indicate that the model is more certain about which parts of the source sequence to attend to when predicting the target tokens.

As seen in Figure 2, SLKD increases the confidence of source-target attention on both MaskT and LevT. The increase is larger for MaskT than for LevT. For LevT, SLKD increases the attention confidence the most at early decoding iterations. At later iterations, as the model becomes more confident about which source tokens to attend to given the target tokens generated at previous iterations,

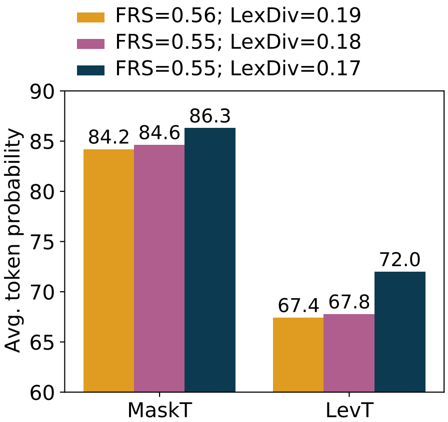

(a) Impact of lexical diversity

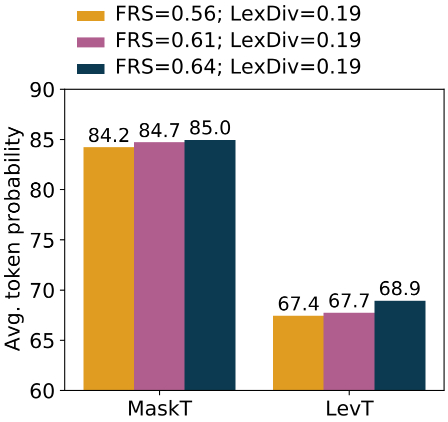

(b) Impact of word reordering

Figure 1: Average token-level uncertainty of MaskT and LevT trained on distilled data with decreasing degree of lexical diversity (a) and word reordering (b) from yellow to blue.

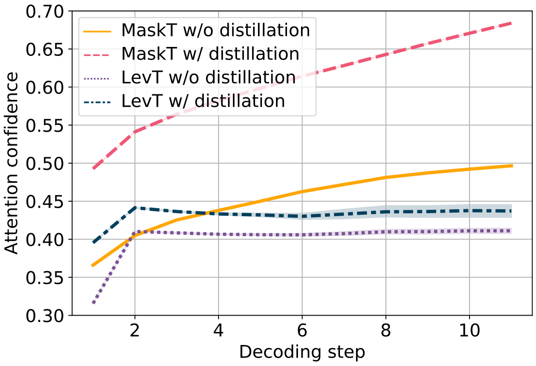

Figure 2: Source-target attention confidence of LevT and MaskT as a function of decoding step.

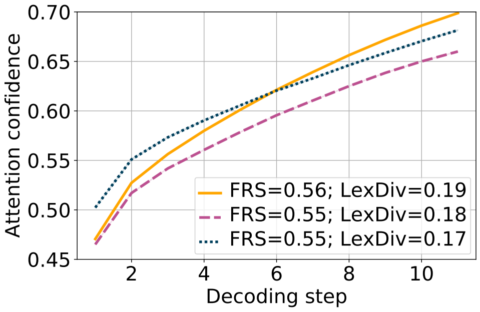

(a) LexDiv on MaskT

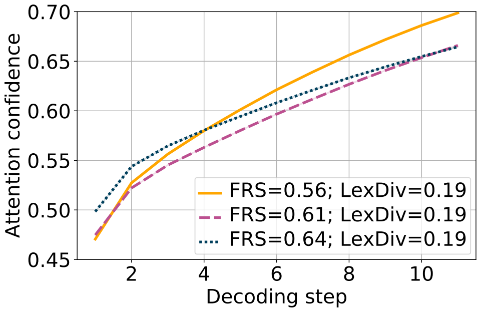

(b) FRS on MaskT

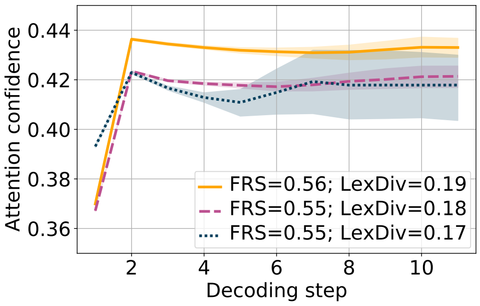

(c) LexDiv on LevT

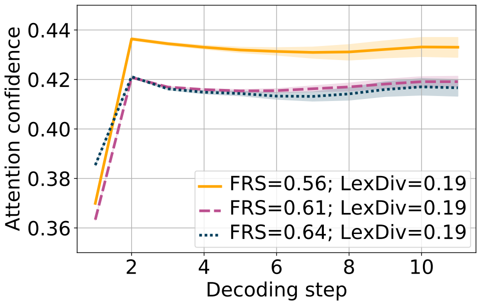

(d) FRS on LevT

Figure 3: Source-target attention confidence as a function of decoding step comparing MaskT and LevT trained on distilled data with varying degree of lexical diversity (a, c) and word reordering (b, d).

the impact of SLKD becomes smaller.

Next, we separate the impact of lexical diversity and word reordering (Figure 3). Reducing both types of complexity leads to more concentrated source-target attention at early iterations. By contrast, models trained on more lexically and syntactically diverse data have more distributed source-target attention at iterations, and the attention becomes more concentrated at later iterations as more target tokens have been generated.

Overall, these results suggest that reducing lexical diversity and degree of word reordering both help NAR find the source-target alignment and thus reduce the error rate at the early decoding stage.

7 Reduced Lexical Diversity in SLKD Improves Model Confidence

Ott et al. (2018) show that the intrinsic uncertainty of translation – due to the existence of multiple semantically equivalent translations for the same source – is a source of uncertainty in the AR models' output distribution. We hypothesize that these effects might be amplified with NAR models, yet little is known about the confidence and calibration of NAR models. We measure the impact of SLKD on model uncertainty using the average token probability of the models' translation outputs, and the inference Expected Calibration Error (ECE) (Wang et al., 2020) that measures how the model's confidence on a prediction matches to the correctness of the prediction. As shown in Table 3, both MaskT and LevT become more confident when trained with SLKD. However, SLKD causes MaskT to be overconfident and hurts its calibration by $+11\%$ ECE.[6] By contrast, SLKD changes LevT from underconfident to slightly overconfident, improving its calibration by -5% lower ECE.

Next, we isolate the impact of lexical diversity and degree of word reordering on model uncertainty.[7] We measure the average token probability of MaskT and LevT trained on data with varying lexical diversity but close FRS scores (Figure 1a), and vice versa (Figure 1b). Decreasing lexical diversity by -0.02 significantly reduces model uncertainty by 2.1–4.6%, whereas the impact of word reordering degree is small: increasing FRS by $+0.08$ only increases the average uncertainty by 0.8–1.5%. By contrast, SLKD boosts FRS by $+0.09$ over the real data. This suggests that reduced lexical diversity is the main reason why SLKD increases model confidence in lexical choice, which raises concerns since Ding et al. (2021) showed that lexical choice errors are also propagated from AR to NAR models through SLKD.

8 Conclusion

We investigated the effect of knowledge distillation in NAR models trained on distilled data that differs along two types of complexity – lexical diversity and degree of word reordering. Reducing lexical diversity and decreasing word reordering degree both boost the confidence of source-target attention, suggesting that they help NAR models learn the alignment between source and target. Furthermore, distillation increases model confidence by reducing lexical diversity, which improves calibration for LevT but leads to much worse calibration for MaskT. These findings reveal a connection between distillation and existing techniques to improve NAR via pre-reordering (Ran et al., 2019) or integrating external alignment information in the source-target attention (Li et al., 2019).[8]

Our findings are based on experiments on the WMT14 English-German corpus, which is widely used in the literature of NAR translation and has interesting typological properties. While we expect these findings to hold for other tasks that exhibit similar degrees of reordering and lexical diversity, it remains to be seen to what degree they generalize to other language pairs and data settings.

We hope that this work will inspire future research on understanding of the positive and negative impact of knowledge distillation on NAR models, as well as of the more advanced approaches to improving NAR by integrating lexical choice and word reordering knowledge. In addition, our work also calls for future work on improving the calibration of NAR models.

Acknowledgments

We thank the anonymous reviewers, Sweta Agrawal, Naomi Feldman, Zara Harmon, Aquia Richburg, Craig Thorburn, the CLIP lab at UMD for helpful comments, and Eleftheria Briakou for her help on result visualization. This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

---

[6] This might be due to decoding where MaskT repeatedly masks out and re-predicts its least confident predictions.

[7] We only measure their isolated impact on model uncertainty, not ECE, because we could not isolate lexical diversity from degree of word reordering while controlling faithfulness, which impacts ECE through accuracy.

[8] These techniques still rely on knowledge distillation for training, while this paper contributes a systematic study of factors that impact the effectiveness of distillation.

References

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021. Understanding and improving lexical choice in non-autoregressive translation. In *International Conference on Learning Representations*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32*, pages 11181–11191. Curran Associates, Inc.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3th International Conference on Learning Representations*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5708–5713, Hong Kong, China. Association for Computational Linguistics.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, pages 592–599.

Toan Q. Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 334–343. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3956–3965, Stockholmsmässan, Stockholm Sweden. PMLR.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Computational*, pages 157–163. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *CoRR*, abs/1911.02215.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 149–159, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725. Association for Computational Linguistics.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5719–5728. PMLR.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 12–21, Edinburgh, Scotland. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079, Online. Association for Computational Linguistics.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2019. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.

A Data Preprocessing Details

Following Gu et al. (2019), we preprocess the WMT14 En-De and De-En datasets (Bojar et al., 2014) via normalization, tokenization, true-casing, and joint BPE (Sennrich et al., 2016) with $37K$ operations.[9] The training data contain 3.9M sentence pairs, and the validation and test sets contain $3,000$ and $3,003$ sentence pairs, respectively.

B Model and Training Details

All AR and NAR models adopt the *base* Transformer architecture (Vaswani et al., 2017) with $d_{\text{model}} = 512$, $d_{\text{hidden}} = 2048$, $n_{\text{heads}} = 8$, $n_{\text{layers}} = 6$, and $p_{\text{dropout}} = 0.3$. We tie the source and target embeddings with the output layer weights (Press and Wolf, 2017; Nguyen and Chiang, 2018). We use label smoothing of 0.1. We train the models using Adam (Kingma and Ba, 2015) with initial learning rate of 0.0005 and a batch size of $64,800$ tokens for maximum $300,000$ steps. We select the best checkpoint based on validation perplexity. The total number of parameters is 65M for the AR model, 66M for MaskT, and 91M for LevT. Training takes around 230 hours for each NAR model and 110 hours for each AR model on 4 Tesla P40 GPUs.

C Detailed Experimental Results

Table 4 shows the scores of corpus-level metrics, test BLEU and validation perplexity of MaskT and LevT trained on various distilled versions of WMT14 De-En training data generated through diverse reference generation (Section 2).

D Reference Generation Examples

We show that the k-best list generated by the AR model using beam search is both lexically and syntactically diverse through a random example selected from the training set (Table 5).

---

[9] Data can be downloaded from `http://dl.fbaipublicfiles.com/nat/original_dataset.zip`

	Data Property			test BLEU		Valid Perplexity	
	FRS	LexDiv	Faith	MaskT	LevT	MaskT	LevT
Real Data	0.46	0.36	0.0	28.0	27.6	35.39	62.49
Distilled Data	0.55	0.18	7.9	29.6	30.6	8.84	11.12
Selection: BLEU	0.54	0.19	7.4	29.4	30.1	9.74	12.57
Selection: BLEU + NMT score ($\lambda = 0.8$)	0.54	0.18	7.4	29.2	30.1	9.45	11.94
Selection: BLEU + NMT score ($\lambda = 0.5$)	0.55	0.17	7.6	29.5	30.6	8.97	11.59
Selection: BLEU + NMT score ($\lambda = 0.2$)	0.55	0.17	7.8	29.2	30.4	8.77	10.94
Selection: BLEU + word-align score ($\lambda = 0.8$)	0.55	0.18	7.4	29.6	30.3	9.63	12.23
Selection: BLEU + word-align score ($\lambda = 0.5$)	0.57	0.18	7.6	29.2	30.1	9.27	11.48
Selection: BLEU + word-align score ($\lambda = 0.2$)	0.58	0.18	7.9	28.7	30.0	8.69	11.24
Selection: BLEU + FRS ($\lambda = 0.8$)	0.56	0.19	7.4	29.1	30.3	9.68	12.10
Selection: BLEU + FRS ($\lambda = 0.5$)	0.61	0.19	7.6	28.8	29.6	9.53	12.25
Selection: BLEU + FRS ($\lambda = 0.2$)	0.64	0.19	7.8	28.5	29.7	8.81	11.71

Table 4: FRS, lexical diversity (*LexDiv*), and faithfulness (*Faith*) scores of various distilled versions of WMT14 De-En training data, test BLEU scores and validation perplexity of MaskT and LevT trained on each data version.

source	Ich hoffe , daß dort in Ihrem Sinne entschieden wird.
original reference	It will , I hope , be examined in a positive light.
translation 1	I hope that it will be decided along your lines.
translation 2	I hope that a decision will be taken along your lines.
translation 3	I hope that the decision will be taken along your lines.
translation 4	I hope that it will be decided in your interest.
translation 5	I hope that there will be a decision along your lines.
translation 6	I hope that decision will be taken along your lines.
translation 7	I hope that the decision will be taken in your interest.
translation 8	I hope that a decision will be taken in your interest.
translation 9	I hope that a decision will be made along your lines.
translation 10	I hope that this will be decided along your lines.
translation 11	I hope that a decision will be taken to that effect.
translation 12	I hope there will be a decision along your lines.
translation 13	I hope that a decision will be taken on your behalf.
translation 14	I hope that a decision will be taken in that regard.
translation 15	I hope that decision will be taken in your interest.
translation 16	I hope that a decision will be taken in that direction.
translation 17	I hope that a decision will be taken in that respect.
translation 18	I hope it will be decided along your lines.
translation 19	I hope that you will take a decision there.
translation 20	I hope that you will take a decision in that regard.
translation 21	I hope that this decision will be taken in your interest.
translation 22	I hope that it will decide along your lines.
translation 23	I hope that it will be decided in your interests.
translation 24	I hope that the decision will be taken in your interests.
translation 25	I hope that the decision will be taken in that direction.
translation 26	I hope that a decision will be taken in your interests.
translation 27	I hope that a decision will be taken to that end.
translation 28	I hope that the decision will be taken in that regard.
translation 29	I hope that a decision will be made in your interest.
translation 30	I hope it will be decided in your interest.
translation 31	I hope that you will take a decision on this.
translation 32	I hope that it will be decided accordingly.

Table 5: An example of the k-best list generated by the AR model using beam search with a beam size of $k = 32$.

Leveraging Topic Relatedness for Argument Persuasion

Xinran Zhao[1], **Esin Durmus**[2], **Hongming Zhang**[1], **and Claire Cardie**[2]
[1]The Hong Kong University of Science and Technology
[2]Cornell University
`xzhaoar@connect.ust.hk, ed459@cornell.edu`
`hzhangal@cse.ust.hk, cardie@cs.cornell.edu`

Abstract

Argumentation exposes individuals to conflicting viewpoints and can help them make more informed decisions based on the pros and cons of a particular issue. While recent studies of argumentation in Natural Language Processing have mainly focused on understanding the effect of various factors of persuasion (i.e. the source, audience, and language style), the impact of exploiting the relationships among controversial topics when predicting argument persuasiveness remains under-explored. In this paper, we model the relatedness among controversial topics utilizing an embedding-based method based on individuals' stances on the topics. We then leverage these *topic embedding* features and incorporate *topic semantics* features extracted from the arguments along with the previously studied factors of persuasion. We show that incorporating both types of topic relatedness features explicitly leads to significant improvement in predicting persuasiveness and also helps enhance generalization to rare topics, in a few-shot setting.

1 Introduction

Emergence of social media and online argumentative forums provide users with a platform to gain information, express, and form opinions on a diverse set of controversial topics (i.e. issues). The increasing importance of these online platforms has motivated NLP researchers to use these platforms as one of the main domains to study the important factors of persuasion. In particular, prior work has shown that characteristics of the speaker (i.e. source), prior beliefs of the audience (Lukin et al., 2017; Durmus and Cardie, 2018), and language style (Feng and Hirst, 2011; Tan et al., 2016) are important factors in determining persuasiveness of the arguments in online argumentation platforms. Although there has been evidence in previous studies of Social Sciences that people's perceptions on a particular controversial topic may be related to their perceptions on other controversial topics (Judd and Krosnick, 1989; Sapra, 2012), the impact of exploiting this relationship among controversial topics are under-explored in NLP studies of persuasion. In this paper, we explicitly study the effect of incorporating topic relatedness among controversial topics in predicting argument persuasiveness.

To study the impact of involving topic relatedness in argument persuasion, we define two types of features: (1) *topic embedding features* and (2) *topic semantics features*. Prior work has shown that topic is an important factor (Das et al., 2016) to determine whether an emotional vs. a logical argument will be received positively by the audience. We hypothesize that encoding underlying relationship among topics with topic embedding features will be helpful in predicting persuasion since similar strategies may be effective for related topics. We further define topic semantics features to encode how focused vs. divergent each of the arguments made by the debaters is given the discussion topic, similar to Zhang et al. (2016).

We first develop an embedding-based technique inspired by (Barkan and Koenigstein, 2016) to determine the relationship among controversial topics. This methodology leverages users' stances on the topics to determine the relationship among them. We then incorporate the *topic embedding features* and *topic semantics features*, along with the previously studied factors of persuasion. We find that incorporating the topic relatedness features help improve state-of-the-art results in persuasion prediction. Moreover, we conduct experiments in a few-shot setting and show that these features help models achieve significantly better generalization performance for the rare topics.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4401–4407
August 1–6, 2021.

2 Dataset

We use DDO (Durmus and Cardie, 2019) for our study. DDO includes 67,315 debates from 23 different categories, 36,294 users with their background information (e.g. political ideology, and religious ideology), and 198,759 votes from the users when they are as readers of these debates. For each debate, two debaters with different viewpoints express their opinions on a controversial topic in rounds. After the debate, voters evaluate the debaters with respect to various criteria and they share whether any of the debaters changed their stance on the topic. Users also have an opportunity to share their demographic and ideological information such as gender, ethnicity, income level, education level, political ideology, and religious ideology. They also share their stance on a pre-defined list of controversial topics (i.e. BIG ISSUES, such as *Abortion*, *Gay Marriage*, and *Global Warming* etc.)[1] that we use to determine semantic relatedness among these controversial topics.

3 Methodology

3.1 Topic Related Embeddings

Topic Embeddings. To capture the underlying relatedness between the debate topics, we learn the embedding for each controversial topics with a method inspired by Barkan and Koenigstein (2016). We hypothesize that the users' opinions (i.e. whether they are SUPPORTING or OPPOSING) on similar topics are related. We treat a set of controversial issues with the same stance from a user as a set of words appearing in the same context and use adapted Skip-gram algorithm proposed by (Mikolov et al., 2013). The embedding vectors are optimized by predicting the topic similarity that is defined as the probability that a pair of topics appearing in the same group with respect to users' opinions. We then can use these vectors to compare the similarity of each pair of the big issues.

Issue	Top similar issues: Similarity
Torture	Iran-Iraq War: 0.90 Electoral College: 0.85 Border Fence: 0.85 Military Intervention: 0.74 Racial Profiling: 0.71
Welfare	Minimum Wage: 0.97 Occupy Movement: 0.86 Medicaid & Medicare: 0.85 Labor Union: 0.84 National Health Care: 0.84
Capitalism	Flat Tax: 0.80 Social Programs: 0.73 Electoral College: 0.69 Affirmative Action: 0.69 Stimulus Spending: 0.66
Environment Protection	Medical Marijuana: 0.93 Abortion: 0.85 Global Warming Exists: 0.85 Drug Legalization: 0.82 United Nations: 0.82

Table 1: Most similar issues for *Torture*, *Welfare*, *Capitalism*, and *Environment Protection* issues.

Table 1 shows the most similar controversial issues for each of the given issues, where the similarity is calculated by the cosine similarity between the embedding vectors. We observe that some of these associations can be more related to relatively intuitive topic similarity between these issues (i.e., *Capitalism* and *Flax Tax*, and *Environmental Protection* and *Global Warming Exists*). However, in some cases, the similarity between controversial issues may have more complicated motivations such as users' underlying ideologies (i.e., as in the case of *Environmental Protection* and *Abortion* which can be justified by the study conducted by Sapra (2012)). Therefore, this method may help identify relationships among controversial issues that are not as intuitive to come up with.

Topic-Centric Attribute Embeddings. There is evidence showing there is a strong association between the users' demographics and their stances towards controversial topics (Sapra, 2012; Tedin et al., 1977). Although prior studies of persuasion has studied the effect of users' attributes on persuasiveness (Lukin et al., 2017; Durmus and Cardie, 2018, 2019), they did not explicitly model the relationship between the users' attributes and their stance towards the controversial topics. To explicitly model this relationship, we create embeddings for users' attributes in a similar way as the issue embeddings introduced in Section 3.1. We optimize the embeddings with the probability of a certain attribute appearing with a given issue-stance pair for each particular user.

Table 2 shows top similar issue-stance combinations for users with given political and religious ideology categories. Similarly, we calculate similarity by the cosine similarity between the embedding vectors. This approach reveals certain associations between users attributes (e.g. Political Ideology) and their stances (e.g. PRO vs. CON) towards certain controversial issues (e.g. *Gay Marriage*). For

---

[1] See https://www.debate.org/big-issues/ for the full list of the Big Issues.

Ideology	Issue-stance: Similarity
Conservative	Gay Marriage-CON: 0.94
	Abortion-CON: 0.93
	Global Warming Exists-CON: 0.92
	Euthanasia-CON: 0.92
	Border Fence-PRO: 0.92
	Death Penalty-PRO: 0.89
Liberal	Gun Rights-CON: 0.87
	Environmental Protection-PRO: 0.83
	Medicaid & Medicare-PRO: 0.83
	Affirmative Action-PRO: 0.83
	Global Warming Exists-PRO: 0.82
	Barack Obama-PRO: 0.81

Table 2: Most similar issue-stance combinations for the given categories of *Political Ideology*.

example, we find that users with *Conservative* vs. *Liberal* political ideologies has different views on *Global Warming* (i.e. CON vs. PRO respectively) issue looking at the similarity of corresponding embeddings.

3.2 Predicting Persuasiveness

We aim to predict which debater, either the PRO or CON side, expresses more persuasive arguments in the debate (i.e., received more votes for the *"Made more convincing arguments"* criterion.).

Debate Topic Representation. We collect debates related to list of issues in BIG ISSUES by using the words of these issues as keywords (e.g. debate topic *"Abortion should be illegal if it pregnancy does not endanger the mother's life and she is adult."* is related to BIG ISSUE "ABORTION"). The dataset includes 2,893 debates and 10,441 votes. We represent the debate topic with the **topic embedding** of the corresponding BIG ISSUE as introduced in Section 3.1. We further encode the text of the debate topic with a fine-tuned BERT (Devlin et al., 2019) taking average embedding of all the tokens to get the representation of the **topic semantics**. We concatenate these two types of embeddings to get the final representation of the debate topic.

Representing User Information. Previous work shows that both characteristics of the debaters and the audience and the linguistic features of the debate arguments are important factors in persuasion studies (Lukin et al., 2017; Durmus and Cardie, 2018, 2019; Longpre et al., 2019). Similar to prior work, to encode the background information, we first represent the user background with one-hot representation (**ONE-HOT**) to capture the users' selections on the categories (e.g., gender, political ideology, religious ideology, and etc.) or the opinion similarity with the voters. However, this representation can be very sparse and not relevant to the topic information. Therefore, we also experiment with the topic-centric embedding-based method (**ATT-EMB**) proposed in Section 3.1. We compute the background similarity as the cosine similarity of the representation vectors for the users (i.e., debaters and voters).

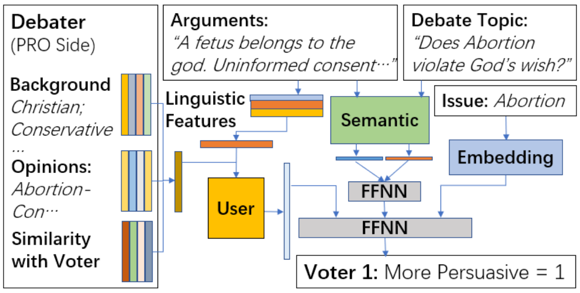

Figure 1: Overall model structure. *User, Semantic, Embedding* blocks denote the encoders for user information, argument semantics, and topic embedding. *FFNN* is a multi-layer feed-forward neural network.

Linguistic Features. Consistent with the prior work (Durmus and Cardie, 2018), to encode the arguments in the debates, we extract linguistic features including the information about the style (i.e., length, links), sentiment polarity, subjectivity (Wilson et al., 2005), and argument lexicon features (Somasundaran and Wiebe, 2010)) etc. Similar to the topic semantics introduced in Section 3.2, we also represent the semantics of the arguments with the same fine-tuned BERT.

Proposed Model. We employ a model that contains separate encoders to represent the debater characteristics, arguments, and topic-related features, as shown in Figure 1. The model encodes the debater's background information and opinions towards the BIG ISSUES, and combines the linguistic features extracted from arguments to represent the users. For the text in the debate, the model consists a siamese network structure (*Semantic* Block + *FFNN* block) (Reimers and Gurevych, 2019) to encode the relation between arguments and debate topics. Then, the model extracts the issue representation from the pretrained issue embedding introduced in Section 3.1. Finally, the user representation, together with the representation for the argument semantics and topic embedding, is passed through a multi-layer feed-forward neural network to predict the voter's perception on the persuasiveness.

Model	F1 (%)
Majority	33.25
Bi-LSTM+Glove	33.41
SBERT	50.05
ONE-HOT+Linguistic+Topic SVM	41.38
ATT-EMB+Linguistic SVM	53.51
ATT-EMB+Linguistic+Topic SVM	59.04
Ours with ONE-HOT	57.03
Ours W/O ARGUMENT	63.81
Ours W/O ATT-EMB	51.33
Ours W/O TOPIC	64.21
Ours	**65.62**

Table 3: Macro F1 scores. **Ours** denotes the model explained in Figure 1. We split the collected debates to train (70%), validation (15%), and test (15%) sets.

4 Baselines

(1) Majority Baseline: We assign the label that appears the most in training set to be the prediction for all test instances. **(2) SVM**: We concatenate debate topic representation, user and linguistic features and classify with SVM with RBF kernel. **(3) Bi-LSTM** (Hochreiter and Schmidhuber, 1997): Following (Durmus et al., 2019; Li et al., 2020), We encode the arguments and topic representation with bidirectional LSTM encoders and use a *FFNN* as classification head[2]. **(4) SBERT** (Reimers and Gurevych, 2019): Sentence-BERT (SBERT) has demonstrated that fine-tuning BERT in a Siamese/Triplet network architecture achieves the state-of-the-art results over various sentence-level classification benchmarks. We use a Siamese network to encode the sentence representation from arguments and debate topics.

5 Results

Model Ablations. The full proposed model contains three parts: user-based features (which can be represented by one-hot vectors or topic-centric user embeddings), argument-based features, and topic-based features. To understand the contribution of each component to prediction performance, we conduct ablation studies for the settings where (1) Using ATT-EMB (i.e. topic-centric user embeddings proposed in this work) vs. ONE-HOT representation to encode the user background, (2) Removing the representation for user background (W/O USER), (3) Removing the argument features and (W/O ARGUMENT) (4) Removing the topic-related features (W/O TOPIC)[3].

Model	Frequent	Rare	Δ
Ours with ATT-EMB	63.31	60.12	**-3.19**
Ours w/o ATT-EMB	56.25	52.65	-3.60
Ours w/o TOPIC-EMB	64.20	58.37	-5.83

Table 4: Results (%macro F1) for the few-shot setting experiment. Δ denotes the performance difference when testing the Rare debates.

Table 3 demonstrates the macro F1 scores for the baselines and the ablations for our model. We observe that our model outperforms the feature-based baselines significantly. For both the SVM model and the deep models, we observe a large performance drop when we use one-hot embedding representation features instead of topic-centric user embeddings. This shows that encoding the relationship between the user background and their opinions on the topics explicitly improves the prediction performance significantly. The experiments on Bi-LSTM and SBERT show that although large-scale pretrained language representation model helps achieve better performance than the baseline, encoding the semantics with deep neural network encoders alone is not as effective as our proposed method. Comparing to the baselines, our proposed method that utilizes the information from different components (i.e. users, language and topic) is more effective. Ablation study shows that components that encode the topic semantics (i.e. ATT-EMB and TOPIC) play an important role to achieve the best performance.

Few-shot Setting. We study whether the topic embeddings also enhance the generalizability across different issues. We split the debates in the test set into frequent and rare categories looking at how often debates with the same topic appear in the training set (more than 200 vs. less than 20). Table 4 shows the results comparing to the baselines for corresponding 324 frequent and 131 rare debates. We see that the gap between the prediction performance is significantly more when we remove the attribute embedding and the topic embeddings, which indicates that the topic-related embeddings benefit the knowledge transferring among debates with different topics.

6 Related Work

Persuasion Studies. Understanding the characteristics of persuasive language has been an im-

---

[2] We choose the hidden embedding dimension to be 200 and use Glove (Pennington et al., 2014) as the pre-trained word embeddings.

[3] Implementation details are described in Appendix A.1.

portant area of study in the Sociology, Psychology (Kelman, 1961; Burgoon et al., 1975; Chaiken, 1987; Tykocinskl et al., 1994; Chambliss and Garner, 1996) and NLP communities (Hasan and Ng, 2014; Habernal and Gurevych, 2016a,b; Fang et al., 2016; Al-Khatib et al., 2017; Wang et al., 2019). The emergence of social media and argumentative forums has further attracted researchers to study the dynamics of persuasion on these platforms, including Twitter (Tan et al., 2014), ChangeMyView community on Reddit (Tan et al., 2016; Hidey et al., 2017) and DDO (Durmus and Cardie, 2019). In this work, we use DDO since it includes a wide-range of user information including users' opinions on various controversial topics as well as well-structured debates with audience votes.

Topic Aware Argument Mining. Farra et al. (2015) studied the effect of topic relevancy or consistency on essay scoring. Bosc et al. (2016) proposes a dataset of Social Media data with coarse topic labels extracted from the hashtags (e.g., #AppleWatch). Zeng et al. (2020) designed a model to encode the latent topics of argumentative conversations. Unlike the previous work, our work studies the effect of argument topic for structured debates explicitly for predicting persuasion.

7 Conclusion

In this paper, we study the impact of topic-relatedness in debate persuasion and find that involving the semantics and features of the debate topics will achieve the best performing model. Moreover, we find that using pretrained embeddings that jointly encode the issues related to the topics and people's characteristics will largely benefit the training process and generalizability. Finally, we find that focusing on the debate topic in making arguments can be an effective strategy in online debates.

Acknowledgements

We thank all the anonymous reviewers for their helpful feedback. This work was supported in part by NSF grant IIS-1815455. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government.

References

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, and Benno Stein. 2017. Patterns of argumentation strategies across topics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1351–1357, Copenhagen, Denmark. Association for Computational Linguistics.

Oren Barkan and Noam Koenigstein. 2016. ITEM2VEC: neural item embedding for collaborative filtering. In *Proceedings of MLSP 2016*.

Tom Bosc, Elena Cabrio, and Serena Villata. 2016. DART: a dataset of arguments and their relations on Twitter. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1258–1263, Portorož, Slovenia. European Language Resources Association (ELRA).

Michael Burgoon, Stephen B. Jones, and Diane Steward. 1975. Towards a message-centered theory of persuasion: Three empirical investigation of language intensity. *Human Communication Research*, 1(3):240–256.

Shelly Chaiken. 1987. The heuristic model of persuasion. In *Social influence: the ontario symposium*, volume 5, pages 3–39.

Marilyn J. Chambliss and Ruth Garner. 1996. Do adults change their minds after reading persuasive text? *Written Communication*, 13(3):291–313.

Abhimanyu Das, Sreenivas Gollapudi, Emre Kıcıman, and Onur Varol. 2016. Information dissemination in heterogeneous-intent networks. In *Proceedings of the 8th ACM Conference on Web Science*, pages 259–268.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*.

Esin Durmus and Claire Cardie. 2018. Exploring the role of prior beliefs for argument persuasion. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1035–1045, New Orleans, Louisiana. Association for Computational Linguistics.

Esin Durmus and Claire Cardie. 2019. A corpus for modeling user and language effects in argumentation on online debating. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 602–607, Florence, Italy. Association for Computational Linguistics.

Esin Durmus, Faisal Ladhak, and Claire Cardie. 2019. The role of pragmatic and discourse context in determining argument impact. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5668–5678, Hong Kong, China. Association for Computational Linguistics.

Hao Fang, Hao Cheng, and Mari Ostendorf. 2016. Learning latent local conversation modes for predicting comment endorsement in online discussions. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 55–64, Austin, TX, USA. Association for Computational Linguistics.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring persuasive essays using opinions and their targets. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–74, Denver, Colorado. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996, Portland, Oregon, USA. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223, Austin, Texas. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, Berlin, Germany. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 751–762, Doha, Qatar. Association for Computational Linguistics.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21, Copenhagen, Denmark. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Charles M Judd and Jon A Krosnick. 1989. The structural bases of consistency among political attitudes: Effects of political expertise and attitude importance. *Attitude structure and function*.

Herbert C. Kelman. 1961. Processes of Opinion Change. *Public Opinion Quarterly*, 25(1):57–78.

Jialu Li, Esin Durmus, and Claire Cardie. 2020. Exploring the role of argument structure in online debate persuasion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8905–8912, Online. Association for Computational Linguistics.

Liane Longpre, Esin Durmus, and Claire Cardie. 2019. Persuasion of the undecided: Language vs. the listener. In *Proceedings of the 6th Workshop on Argument Mining*.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder: Audience effects in persuasion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 742–753, Valencia, Spain. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Sonalini Sapra. 2012. Feminist perspectives on the environment. In *Oxford Research Encyclopedia of International Studies*.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 116–124, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 175–185, Baltimore, Maryland. Association for Computational Linguistics.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 613–624, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Kent L. Tedin, David W. Brady, and Arnold Vedlitz. 1977. Sex differences in political attitudes and behavior: The case for situational factors. *The Journal of Politics*.

Orit Tykocinskl, E. Tory Higgins, and Shelly Chaiken. 1994. Message framing, self-discrepancies, and yielding to persuasive messages: The motivational significance of psychological situations. *Personality and Social Psychology Bulletin*, 20(1):107–115.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649, Florence, Italy. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis.

Jichuan Zeng, Jing Li, Yulan He, Cuiyun Gao, Michael Lyu, and Irwin King. 2020. What changed your mind: The roles of dynamic topics and discourse in argumentation process. In *Proceedings of The Web Conference 2020*, WWW '20, page 1502–1513, New York, NY, USA. Association for Computing Machinery.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in Oxford-style debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 136–141, San Diego, California. Association for Computational Linguistics.

A Appendix

A.1 Implementation Details

We initialize the semantic encoder for arguments and debate topics with the BERT-base model with 110M parameters. We pad the input sentences with BERT start and end symbols (i.e., [CLS] and [SEP]). For each round of the debate, we take at most three sentences (as (Li et al., 2020)) to take average and represent the arguments. The hidden size for the Bi-LSTM layers is 300. The size for the hidden states for *FFNN* blocks are 256. During training, we use cross entropy as the loss function and stochastic gradient decent (SGD) as the optimizer. We initialize all parameters randomly and train all the model with 10 epochs. The best performing models on the validation set are evaluated on the test set.

One Teacher is Enough?
Pre-trained Language Model Distillation from Multiple Teachers

Chuhan Wu[†] **Fangzhao Wu**[‡] **Yongfeng Huang**[†]
[†]Department of Electronic Engineering & BNRist, Tsinghua University, Beijing 100084, China
[‡]Microsoft Research Asia, Beijing 100080, China
{wuchuhan15, wufangzhao}@gmail.com, yfhuang@tsinghua.edu.cn

Abstract

Pre-trained language models (PLMs) achieve great success in NLP. However, their huge model sizes hinder their applications in many practical systems. Knowledge distillation is a popular technique to compress PLMs, which learns a small student model from a large teacher PLM. However, the knowledge learned from a single teacher may be limited and even biased, resulting in low-quality student model. In this paper, we propose a multi-teacher knowledge distillation framework named MT-BERT for pre-trained language model compression, which can train high-quality student model from multiple teacher PLMs. In MT-BERT we design a multi-teacher co-finetuning method to jointly finetune multiple teacher PLMs in downstream tasks with shared pooling and prediction layers to align their output space for better collaborative teaching. In addition, we propose a multi-teacher hidden loss and a multi-teacher distillation loss to transfer the useful knowledge in both hidden states and soft labels from multiple teacher PLMs to the student model. Experiments on three benchmark datasets validate the effectiveness of MT-BERT in compressing PLMs.

1 Introduction

Pre-trained language models (PLMs) such as BERT and RoBERTa have achieved notable success in various NLP tasks (Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019). However, many PLMs have a huge model size and computational complexity, making it difficult to deploy them to low-latency and high-concurrence online systems or devices with limited computational resources (Jiao et al., 2020; Wu et al., 2021).

Knowledge distillation is a widely used technique for compressing large-scale pre-trained language models (Sun et al., 2019; Wang et al., 2020). For example, Sanh et al. (2019) proposed DistilBERT to compress BERT by transferring knowledge from the soft labels predicted by the teacher model to student model with a distillation loss. Jiao et al. (2020) proposed TinyBERT, which aligns the hidden states and the attention heatmaps between student and teacher models. These methods usually learn the student model from a single teacher model (Gou et al., 2020). However, the knowledge and supervision provided by a single teacher model may be insufficient to learn an accurate student model, and the student model may also inherit the bias in the teacher model (Bhardwaj et al., 2020). Fortunately, many different large PLMs such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and UniLM (Dong et al., 2019) are off-the-shelf. These PLMs may encode complementary knowledge because they usually have different configurations and are trained on different corpus with different self-supervision tasks (Qiu et al., 2020). Thus, incorporating multiple pre-trained language models into knowledge distillation has the potential to learn better student models.

In this paper, we present a multi-teacher knowledge distillation method named MT-BERT for pre-trained language model compression.[1] In MT-BERT, we propose a multi-teacher co-finetuning framework to jointly finetune multiple teacher models with a shared pooling and prediction module to align their output hidden states for better collaborative student teaching. In addition, we propose a multi-teacher hidden loss and a multi-teacher distillation loss to transfer the useful knowledge in both hidden states and soft labels from multiple teacher models to student model. Experiments on three benchmark datasets show MT-BERT can effectively improve the quality of student models for PLM compression and outperform many single-teacher knowledge distillation methods.

---

[1] We focus on task-specific knowledge distillation.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4408–4413
August 1–6, 2021.

2 MT-BERT

Next, we introduce the details of our multi-teacher knowledge distillation method MT-BERT for pre-trained language model compression.[2] We first introduce the multi-teacher co-finetuning framework to jointly finetune multiple teacher models in downstream tasks, and then introduce the multi-teacher distillation framework to collaboratively teach the student with multiple teachers.

2.1 Multi-Teacher Co-Finetuning

Researchers have found that distilling the knowledge in the hidden states of a teacher model is important for effective student teaching (Sun et al., 2019; Jiao et al., 2020). However, since different teacher PLMs are separately pre-trained with different settings, finetuning them independently may lead to some inconsistency in their feature space, which is not optimal for transferring knowledge in the hidden states of multiple teachers. Thus, we design a multi-teacher co-finetuning framework to obtain some uniformity among the hidden states output by the last layer of different teacher models for better collaborative student teaching, as shown in Fig. 1. Assume there are N teacher models, and denote the hidden states output by the top layer of the i-th teacher as $\mathbf{H}^i$. We use a shared pooling[3] layer to summarize each hidden matrix $\mathbf{H}^i$ into a unified text embedding, and then use a shared dense layer to convert it into a soft probability vector $\mathbf{y}_i$. Finally, we jointly optimize the summation of the task-specific losses of all teacher models, i.e., $\sum_{i=1}^{N} \mathrm{CE}(\mathbf{y}, \mathbf{y}_\mathrm{i})$, where $\mathrm{CE}(\cdot,\cdot)$ stands for the cross-entropy loss and $\mathbf{y}$ is the ground-truth label. Since the pooling and prediction layers are shared among different teachers, the feature space of the output hidden states from different teacher PLMs can be aligned, which can help them collaborate better for student teaching.

2.2 Multi-Teacher Knowledge Distillation

Next, we introduce our proposed multi-teacher knowledge distillation framework, which is shown in Fig. 2. Two loss functions are used for knowledge distillation, i.e., a multi-teacher hidden loss and a multi-teacher distillation loss.

The multi-teacher hidden loss aims to transfer knowledge in the hidden states of multiple teachers.

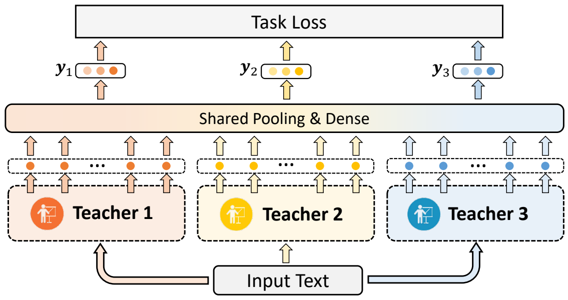

Figure 1: The multi-teacher co-finetuning framework.

Assume there are N teacher PLMs, and each of them has $T \times K$ Transformer layers. They collaboratively teach a student model with K layers, and each layer in the student model corresponds to T layers in teacher PLMs.[4] Denote the hidden states output by the j-th layer of the student model as $\mathbf{H}_j^s$, and the corresponding hidden states output by the $(T \times j)$-th layer of the i-th teacher model as $\mathbf{H}_{Tj}^i$. Following (Sun et al., 2019), we apply the mean squared error (MSE) to the hidden states of corresponding layers in the student and teacher models to encourage the student model to have similar functions with teacher models. The multi-teacher hidden loss $\mathcal{L}_{MT-Hid}$ is formulated as follows:

$$\mathcal{L}_{MT-Hid} = \sum_{i=1}^{N} \sum_{j=1}^{T} \mathrm{MSE}(\mathbf{H}_\mathrm{j}^\mathrm{s}, \mathbf{W}_\mathrm{ij} \mathbf{H}_\mathrm{Tj}^\mathrm{i}), \quad (1)$$

where $\mathbf{W}_{ij}$ is a learnable transformation matrix.

The multi-teacher distillation loss aims to transfer the knowledge in the soft labels output by multiple teachers to student. The predictions of different teachers on the same sample may have different correctness and confidence. Thus, it may be suboptimal to simply ensemble (Fukuda et al., 2017; Liu et al., 2020) or choose (Yuan et al., 2020) soft labels without the help of task labels. Since in task-specific knowledge distillation the labels of training samples are available, we propose a distillation loss weighting method to assign different weights to different samples. The weights are based on the loss inferred from the predictions of corresponding teacher against the gold labels. More specifically, the multi-teacher distillation loss $\mathcal{L}_{MT-Dis}$ is formulated as follows:

$$\mathcal{L}_{MT-Dis} = \sum_{i=1}^{N} \frac{\mathrm{CE}(\mathbf{y}_\mathrm{i}/\mathrm{t}, \mathbf{y}_\mathrm{s}/\mathrm{t})}{1 + \mathrm{CE}(\mathbf{y}, \mathbf{y}_\mathrm{i})}, \quad (2)$$

---

[2] Codes available at https://github.com/wuch15/MT-BERT

[3] In MT-BERT we use attentive pooling because it performs better than average pooling and "[CLS]" token embedding.

[4] Here we assume that all teacher models have the same number of layers. We will explore to generalize MT-BERT to scenarios where teacher models have different architectures in our future work.

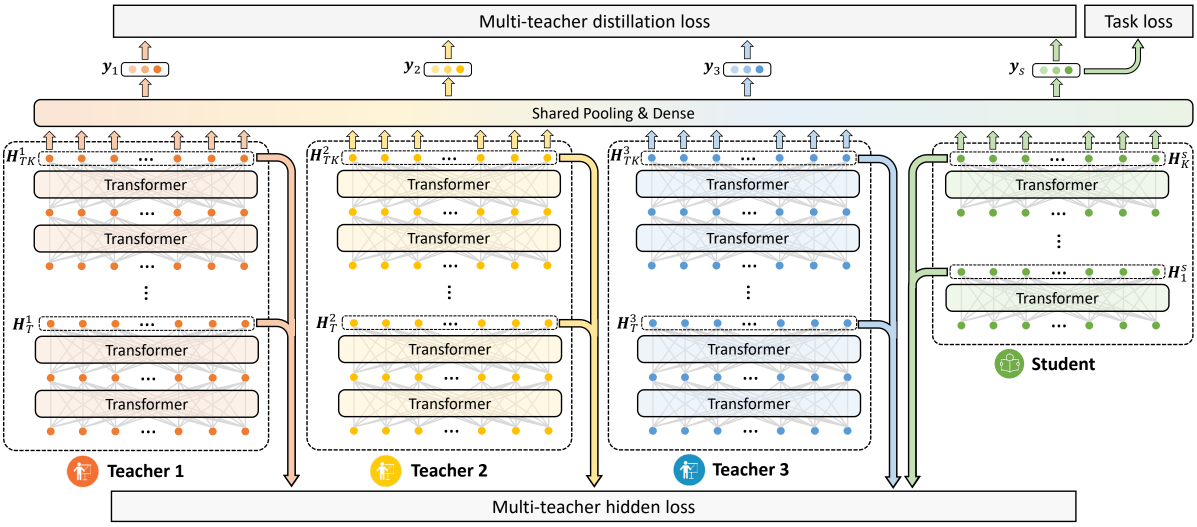

Figure 2: The multi-teacher knowledge distillation framework in MT-BERT.

where t is the temperature coefficient. In this way, if a teacher's prediction on a certain sample is more close to the ground-truth label, its corresponding distillation loss will gain higher weight.

Following (Tang et al., 2019; Lu et al., 2020), we also incorporate gold labels to compute the task-specific loss $\mathcal{L}_{Task}$ based on the predictions of the student model, i.e., $\mathcal{L}_{Task} = \text{CE}(\mathbf{y}, \mathbf{y}_s)$. The final loss function $\mathcal{L}$ for learning the student model is a summation of the multi-teacher hidden loss, multi-teacher distillation loss and the task-specific loss, which is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{MT-Hid} + \mathcal{L}_{MT-Dis} + \mathcal{L}_{Task}. \quad (3)$$

3 Experiments

3.1 Datasets and Experimental Settings

We conduct experiments on three benchmark datasets with different sizes. The first one is SST-2 (Socher et al., 2013), which is a benchmark for text sentiment classification. The second one is RTE (Bentivogli et al., 2009), which is a widely used dataset for natural language inference. The third one is the MIND dataset (Wu et al., 2020c), which is a large-scale public English news dataset.[5] We perform the news topic classification task on this dataset. The detailed statistics of the three datasets are shown in Table 1.

In our experiments, we use the pre-trained 12-layer BERT, RoBERTa and UniLM (Bao et al., 2020)[6] models as the teachers to distill a 6-layer and a 4-layer student models respectively. We use the token embeddings and the first 4 or 6 Transformer layers of UniLM to initialize the parameters of the student model. The pooling layer is implemented by an attention network (Yang et al., 2016; Wu et al., 2020a). The temperature coefficient t is set to 1. The attention query dimension in the attentive pooling layer is 200. The optimizer we use is Adam (Bengio and LeCun, 2015). The teacher model learning rate is 2e-6 while the student model learning rate is 5e-6. The batch size is 64. Following (Jiao et al., 2020), we report the accuracy score on the SST-2 and RTE datasets. In addition, since the news topics in the MIND dataset are highly imbalanced, following (Wu et al., 2020b) we report both accuracy and macro-F1 scores. Each experiment is independently repeated 5 times and the average scores are reported.

Dataset	#Train	#Dev	#Test	#Class
SST-2	67k	872	1.8k	2
RTE	2.5k	276	3.0k	2
MIND	102k	2.6k	26k	18

Table 1: The statistics of the three datasets.

3.2 Performance Evaluation

We compare the performance of MT-BERT with two groups of baselines. The first group includes the 12-layer version of the teacher models, i.e., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and UniLM (Bao et al., 2020). The second group includes the 6-layer and 4-layer student

---

[5] https://msnews.github.io/

[6] We used the UniLMv2 version.

Methods	**SST-2**	**RTE**	**MIND**		**#Param**
	(Acc.)	(Acc.)	(Acc./Macro-F)		
BERT$_{12}$	92.8	68.6	73.6	51.3	109M
RoBERTa$_{12}$	94.8	78.7	73.9	51.5	109M
UniLM$_{12}$	**95.1**	**81.3**	**74.6**	**51.9**	109M
DistilBERT$_6$	92.5	58.4	72.5	50.4	67.0M
DistilBERT$_4$	91.4	54.1	72.1	50.2	52.2M
BERT-PKD$_6$	92.0	65.5	72.7	50.6	67.0M
BERT-PKD$_4$	89.4	62.3	72.4	50.3	52.2M
TinyBERT$_6$	93.1	70.0	73.4	50.8	67.0M
TinyBERT$_4$	92.6	66.6	73.0	50.4	14.5M
MT-BERT$_6$	**94.6**	**75.7**	**74.0**	**51.5**	67.0M
MT-BERT$_4$	93.9	73.8	73.8	51.2	52.2M

Table 2: Results and parameters of different methods.

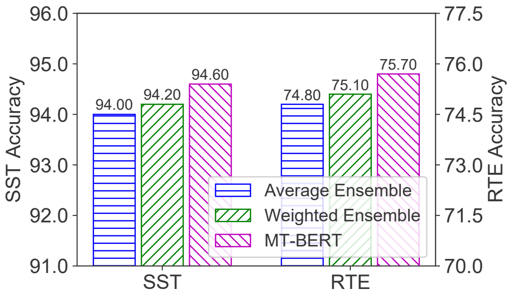

(a) SST-2 and RTE datasets.

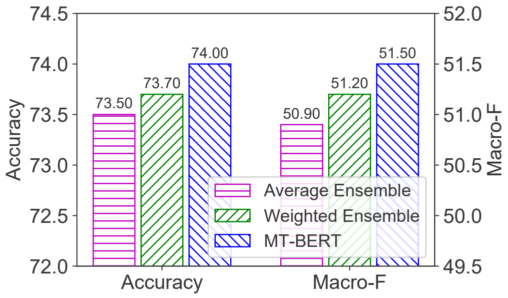

(b) MIND dataset.

Figure 3: Comparison of MT-BERT and ensemble-based multi-teacher distillation methods.

models distilled by DistilBERT (Sanh et al., 2019), BERT-PKD (Sun et al., 2019) and TinyBERT (Jiao et al., 2020), respectively. The results of different methods are summarized in Table 2.[7] Referring to this table, we find MT-BERT can consistently outperform all the single-teacher knowledge distillation methods compared here. This is because the knowledge provided by a single teacher model may be insufficient, and incorporating the complementary knowledge encoded in multiple teacher models can help learn better student model. In addition, compared with the teacher models, MT-BERT has much fewer parameters and its performance is comparable or even better than these teacher models. It shows that MT-BERT can effectively inherit the knowledge of multiple teacher models even if the model size is significantly compressed.

Teachers	**SST-2**	**RTE**	**MIND**	
	(Acc.)	(Acc.)	(Acc./Macro-F)	
BERT	92.1	65.8	72.8	50.6
RoBERTa	92.9	68.9	73.0	50.7
UniLM	93.3	70.6	73.4	50.9
BERT+RoBERTa	93.6	71.2	73.3	50.9
BERT+UniLM	93.9	73.7	73.6	51.1
RoBERTa+UniLM	94.3	74.9	73.7	51.3
All	94.6	75.7	74.0	51.5

Table 3: Different combinations of teacher models.

We also compare MT-BERT with several multi-teacher knowledge distillation methods proposed in the computer vision field that ensemble the outputs of different teachers for student teaching (You et al., 2017; Liu et al., 2020). The results are shown in Fig. 3. We find our MT-BERT performs better than these ensemble-based multi-teacher knowledge distillation methods. This is because these methods do not consider the correctness of the teacher model predictions on a specific sample and cannot transfer useful knowledge encoded in the intermediate layers, which may not be optimal for collaborative knowledge distillation from multiple teachers.

3.3 Effectiveness of Multiple Teachers

Next, we study the effectiveness of using multiple teacher PLMs for knowledge distillation. We compare the performance of the 6-layer student model distilled from different combinations of teacher models. The results are summarized in Table 3. It shows that using multiple teacher PLMs can achieve better performance than using a single one. This is because different teacher models can encode complementary knowledge and combining them together can provide better supervision for student model. In addition, combining all three teacher PLMs can further improve the performance of student model, which validates the effectiveness of MT-BERT in distilling knowledge from multiple teacher models.

3.4 Ablation Study

We study the effectiveness of the two important techniques in MT-BERT, i.e., the multi-teacher co-finetuning framework and the distillation loss

---

[7] We take the original reported results of baseline methods on the SST-2 and RTE datasets, and we run their codes to obtain their results on the *MIND* dataset.

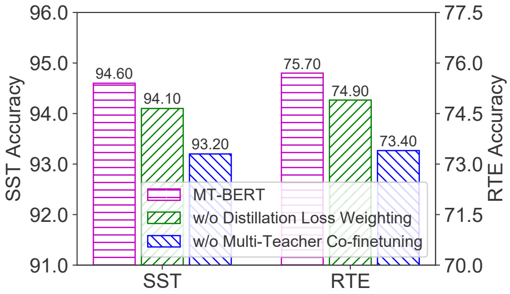

(a) SST-2 and RTE datasets.

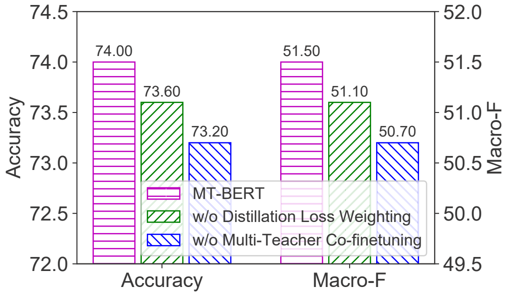

(b) MIND dataset.

Figure 4: Effectiveness of multi-teacher co-finetuning and distillation loss weighting.

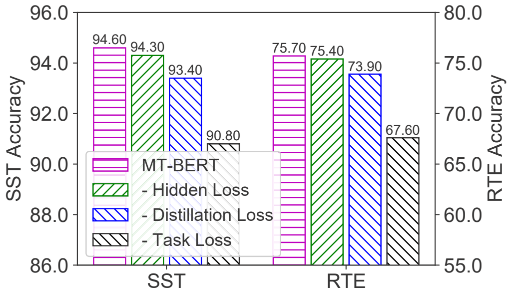

(a) SST-2 and RTE datasets.

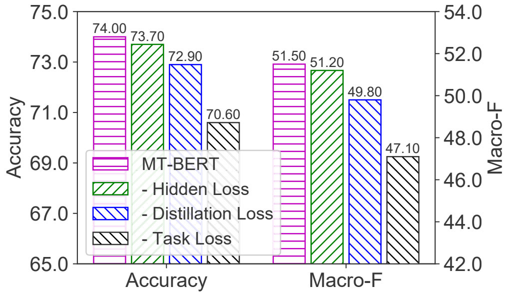

(b) MIND dataset.

Figure 5: Effectiveness of different loss functions.

weighting method. We compare MT-BERT and its variants with one of these modules removed, as shown in Fig. 4. The student model has 6 layers. We find the multi-teacher co-finetuning framework is very important. This is because the hidden states of different teacher models can be in very different spaces, and jointly finetuning multiple teachers with shared pooling and prediction layers can align their output hidden spaces for better collaborative student teaching. In addition, the distillation loss weighting method is also useful. This is because the predictions of different teachers on the same sample may have different correctness, and focusing on the more reliable predictions is helpful for distilling accurate student models.

We also verify the effectiveness of different loss functions in MT-BERT, which is shown in Fig. 5. We find the task loss is very important. It is because in our experiments the corpus for task-specific distillation are not large and the direct supervision from task labels is useful. In addition, the distillation loss is also important. It indicates that transferring the knowledge in soft labels plays a critical role in knowledge distillation. Moreover, the hidden loss is also helpful. It shows that hidden states of different teacher models can provide useful knowledge for student model learning.

4 Conclusion

In this paper, we propose a multi-teacher knowledge distillation method named MT-BERT for pre-trained language model compression, which can learn small but strong student model from multiple teacher PLMs in a collaborative way. We propose a multi-teacher co-finetuning framework to align the output hidden states of multiple teacher models for better collaborative student teaching. In addition, we design a multi-teacher hidden loss and a multi-teacher distillation loss to transfer the useful knowledge in both hidden states and prediction of multiple teacher models to student model. The extensive experiments on three benchmark datasets show that MT-BERT can effectively improve the performance of pre-trained language model compression, and can outperform many single-teacher knowledge distillation methods.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant numbers U1936208, U1936216, U1836204, and U1705261. We thank Xing Xie, Tao Qi, Ruixuan Liu and Tao Di for their great comments and suggestions which are important for improving this work.

References

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*, pages 642–652. PMLR.

Yoshua Bengio and Yann LeCun. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. 2020. Investigating gender bias in BERT. *arXiv preprint arXiv:2009.05021*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*, pages 13042–13054.

Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. 2017. Efficient knowledge distillation from an ensemble of teachers. In *Interspeech*, pages 3697–3701.

Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. 2020. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *EMNLP Findings*, pages 4163–4174.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yuang Liu, Wei Zhang, and Jun Wang. 2020. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113.

Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *CIKM*, pages 2645–2652.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *EMNLP-IJCNLP*, pages 4314–4323.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *NeurIPS*.

Chuhan Wu, Fangzhao Wu, Tao Qi, Xiaohui Cui, and Yongfeng Huang. 2020a. Attentive pooling with learnable norms for text representation. In *ACL*, pages 2961–2970.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2020b. Improving attention mechanism with query-value interaction. *arXiv preprint arXiv:2010.03766*.

Chuhan Wu, Fangzhao Wu, Yang Yu, Tao Qi, Yongfeng Huang, and Qi Liu. 2021. Newsbert: Distilling pre-trained language model for intelligent news application. *arXiv preprint arXiv:2102.04887*.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020c. Mind: A large-scale dataset for news recommendation. In *ACL*, pages 3597–3606.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5753–5763.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489.

Shan You, Chang Xu, Chao Xu, and Dacheng Tao. 2017. Learning from multiple teacher networks. In *KDD*, pages 1285–1294.

Fei Yuan, Linjun Shou, Jian Pei, Wutao Lin, Ming Gong, Yan Fu, and Daxin Jiang. 2020. Reinforced multi-teacher selection for knowledge distillation. *arXiv preprint arXiv:2012.06048*.

Logic-Consistency Text Generation from Semantic Parses

Chang Shu[1*], **Yusen Zhang**[2*], **Xiangyu Dong**[3], **Peng Shi**[4], **Tao Yu**[5], **Rui Zhang**[2]

[1]School of Informatics, University of Edinburgh
[2]Department of Computer Science and Engineering, Penn State University
[3]School of Computer Science and Engineering, Beihang University
[4]David R. Cheriton School of Computer Science, University of Waterloo
[5]Department of Computer Science, The University of Hong Kong
`s1783039@ed.ac.uk, yfz5488@psu.edu, dxy912434027@gmail.com`
`peng.shi@uwaterloo.ca, rmz5227@psu.edu`

Abstract

Text generation from semantic parses is to generate textual descriptions for formal representation inputs such as logic forms and SQL queries. This is challenging due to two reasons: (1) the complex and intensive inner logic with the data scarcity constraint, (2) the lack of automatic evaluation metrics for logic consistency. To address these two challenges, this paper first proposes SNOWBALL, a framework for logic consistent text generation from semantic parses that employs an iterative training procedure by recursively augmenting the training set with quality control. Second, we propose a novel automatic metric, BLEC, for evaluating the logical consistency between the semantic parses and generated texts. The experimental results on two benchmark datasets, *Logic2Text* and *Spider*, demonstrate the SNOWBALL framework enhances the logic consistency on both BLEC and human evaluation. Furthermore, our statistical analysis reveals that BLEC is more logically consistent with human evaluation than general-purpose automatic metrics including BLEU, ROUGE and, BLEURT. Our data and code are available at `https://github.com/Ciaranshu/relogic`.

1 Introduction

Natural language generation (NLG) from semantic parses is to generate the text description for the formal representation input such as logical forms, AMR, and SQL queries. It has drawn widespread attention because of its substantial contributions to the interpretability and usability of the latest natural language interfaces (Gatt and Krahmer, 2018; Chen et al., 2020b; Hu et al., 2020; Mishra et al., 2019; Yu et al., 2019; Ngomo et al., 2013; Wang et al., 2018; Gardent et al., 2017; Wang et al., 2020a; Wang, 2019; Koutrika et al., 2010a). Recently, pretrained large-scale language models like BERT (Devlin et al., 2018), T5 (Raffel et al., 2020), and GPT-3 (Brown et al., 2020) have raised the ability to generate natural language from formal texts to a promising level of fluency and coherence.

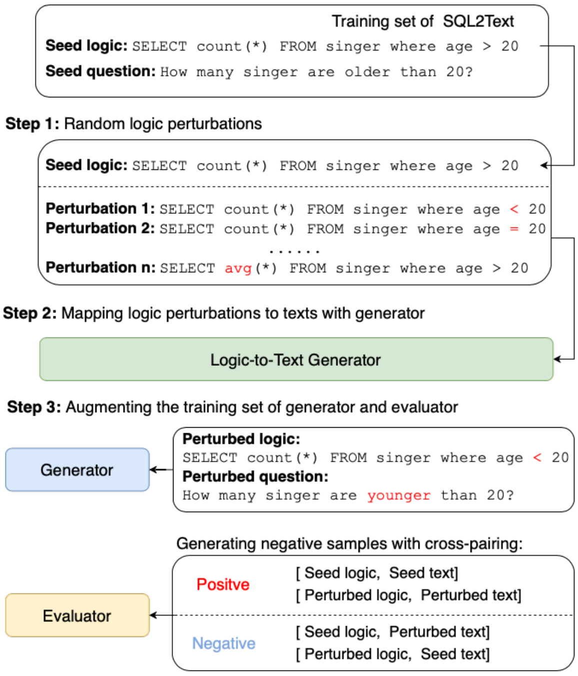

Figure 1: Our data augmentation procedure for the generator and evaluator in the SNOWBALL framework.

However, NLG from semantic parses still has suffered from two crucial challenges: (1) the data scarcity constraint due to the bias on certain types of logic forms or expensive labeling work (Iyer et al., 2017; Yaghmazadeh et al., 2017), which potentially leads to the unsatisfied fidelity of remaining the complex and intensive inner logic in the generated text based on our empirical research; (2) The general-purpose automatic metrics (Novikova et al., 2017a) such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and BLEURT (Sellam et al.,

---

*Equal Contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4414–4426
August 1–6, 2021. ©2021 Association for Computational Linguistics

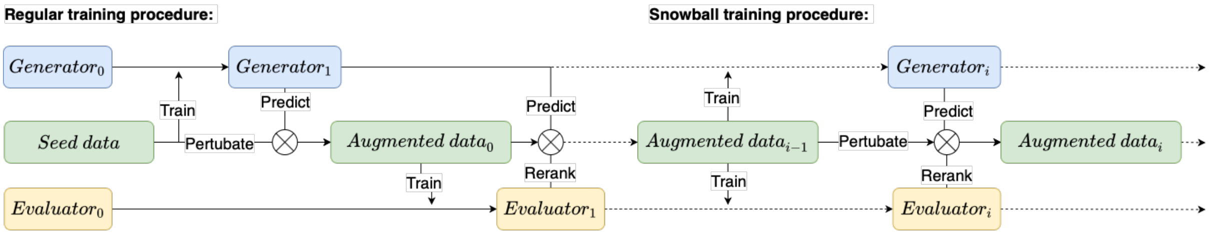

Figure 2: Our SNOWBALL framework employs an iterative training procedure over a generator and evaluator through data augmentation.

2020) are not ideal for explicitly measuring the logic consistency (Wang et al., 2020b; Harkous et al., 2020), because they tend to evenly weight each word in the generated text without fully attending on the fatal logical keywords.

To address these two critical problems, we propose the SNOWBALL framework for high-fidelity text generation from semantic parses and the BLEC automatic evaluation metric for logic consistency:

Snowball Framework. Our SNOWBALL framework, as illustrated in Figure 2, trains two modules to ensure high-fidelity text generation: (1) a generator that maps the logical form to its textual description, and (2) an evaluator that indicates the logic consistent score of each pair of logical form and textual sentence. Rather than training the generator and evaluator independently, SNOWBALL performs iterative training on the generator and the evaluator. To deal with the data scarcity issue, we propose a data augmentation procedure to cover valid logic variations with diverse natural language expressions to improve generalizability. To this end, during each iteration, various unseen logic pairs could be automatically generated with rule-based enumerated logic forms and their corresponding text predicted by the generator. The evaluator is then used to filter out the high reliable augmented logic pairs for the next training iteration.

BLEC Metric. To evaluate the logic consistency of the text generated by the model, we propose a rule-based automatic evaluation metric called Bidirectional Logic Evaluation of Consistency, or BLEC. It takes the logical form and the generated corresponding natural language text as input, then outputs a label indicating if they represent consistent logic. Compared with the neural network evaluator, BLEC can be easily deployed to different datasets, as long as the parser (i.e., the grammar of the logical form) is given.

In our experiments, we exam the effectiveness of our proposed approaches on the benchmark datasets of NLG from semantic parses derived from existed Text-to-SQL dataset *Spider* (Yu et al., 2018) and Table-to-Text dataset *Logic2Text* Chen et al. (2020b). Our analysis shows that our BLEC metric has a substantially positive Pearson score with human annotations, demonstrating better logic consistency than other automatic metrics. The BLEC result shows that the SNOWBALL framework leads to accordant enhancement in logic consistency on two datasets compared to the single-pass training method based on BART (Lewis et al., 2020).

Our key contributions are summarized into three-folds: (1) We propose a simple but effective training framework SNOWBALL that strengthens the logic faithfulness of generated text by covering diverse logic variations. (2) We propose a new logic evaluation metric BLEC that accurately measures the logical consistency with a refined keyword matching mechanism. (3) Our experiment results demonstrate that SNOWBALL at most increases the BLEC from 10.1% on SQL-to-Text and 1.2% on Logic-to-Text tasks compared to the baseline. Moreover, our statistical analysis reveals that BLEC achieves a +0.66 Pearson correlation coefficient compared with human labels, serving as a much better automatic evaluation metric than not only the traditional BLEU and ROUGE metrics, but the latest BLEURT metrics.

2 Related Work

2.1 Parses-to-Text

The source of data-to-text (D2T) datasets is mostly a flat ontology structure, like E2E(Novikova et al., 2017b), LogicNLP(Chen et al., 2020a), RotoWire(Wiseman et al., 2017), and ToTTo(Parikh et al., 2020), which is not powerful enough to encode rich semantic relationships in the ontology. Second, some datasets, such as WebNLG(Gardent et al., 2017), E2E, and RotoWire, have a limited number of domains. E2E is on the restaurant domain, and RotoWire is on the basketball domain.

Moreover, some of them only have loose alignments between input and sentence, e.g., RotoWire.

Generating the natural language descriptions for the logic forms or parses as a sub-task of D2T, has been studied in various datasets and tasks, such as GCC grammar to text (White, 2006), and UCC grammar to text (Gardent and Plainfossé, 1990). There are a lot of works that leverage the neural networks to conduct the generation on various tasks, for example, generating natural language from AMR (Song et al., 2018; Ribeiro et al., 2019; Damonte and Cohen, 2019), logic forms (Chen et al., 2020b), as well as SQL parses (Xu et al., 2018; Ngonga Ngomo et al., 2013; Koutrika et al., 2010b). However, different from these works, our work focuses on the logic consistency generation from parses. So we will mainly discuss and evaluate the model based on the logic between parses and questions.

2.2 High-fidelity Text Generation

As for the end-to-end neural-based text generation models, collaborating the auxiliary task during model training is an intuitive method that introduces the logic regulation to the models. For instance, the fidelity classification task proposed by Harkous et al. (2020), the auxiliary span extraction tasks by Kryscinski et al. (2020),the table-text optimal-transport matching and embedding similarity losses by Wang et al. (2020b) and the content matching task presented by Parikh et al. (2020) are proved to be effective. Nevertheless, to the best of our knowledge, we are the first to bridge the training procedure of evaluator and generator together with the iterative training framework snowball. Furthermore, we attempt to construct a new automatic metric and a new dataset dedicated to evaluating the logic consistency of text generation. The concentration of our work differs from the related high-fidelity text generation work (Chen et al., 2020b; Chan et al., 2019; Nie et al., 2018; Tian et al., 2019; Wang et al., 2020a), by attempting to present the panorama of the challenges of logic-consistent text generation instead of focusing on the model-wised modifications.

3 Snowball Framework

The SNOWBALL framework addresses the challenge of the complex and intensive inner logic with data sparsity constraint for the high-fidelity text-generation from semantic parses. As illustrated in Figure 2, SNOWBALL assures the logic consistency with three bases: (1) Iterative training procedure synergistically enhances the generator and evaluator in the adversarial fashion; (2) Data augmentation based on rule-based logic perturbations and neural-based text generation covering diverse unseen logic variations for iterative training; (3) Structure-aware encoding boost the sensibility of the encoder on mild logic shift.

3.1 Iterative Training

Rather than training the generator and evaluator independently, SNOWBALL performs training on the generator and the evaluator iteratively. As demonstrated in Figure 2, the prerequisite of the snowball training procedure is the regular training procedure: (1) the $Generator_0$ is trained on the benchmark NLG datasets with the normal end-to-end approach into trained $Generator_1$; (2) meanwhile, the logic forms in the seed data are converted into variations with given rules, then the $Generator_1$ predicts the text for each mutated logic forms to be a completed logic pair; (3) The initial $Evaluator_0$ is then trained on those augmented logic pairs.

Then, during the SNOWBALL procedure, the generator and evaluator are collaboratively improved through several training iterations, and during each iteration, a three-step adversarial interaction would be conducted between the generator and evaluator: **Step 1:** The trained $Evaluator_{i-1}$ could be used to rerank the beam search results given by the decoder of the generator, consequently leading to increased quality of the augmented logic pairs, $Augmented\ data_{i-1}$; **Step 2:** The $Generator_i$ is capable to better retain the logic consistency by training on the $Augmented\ data_{i-1}$ which contains more unseen logic variations uncovered in the seed data; **Step 3:** The enhanced $Generator_i$ predicts the increasingly realistic-like perturbed sentences from the perturbed logical forms, which brings more challenging negative samples to the training set of the $Evaluator_i$. The data augmentation in the first step would be further described in Section 3.2.

To be specific, our generator and evaluator in SNOWBALL are described as follows.

Generator The generator maps the logical form to the corresponding natural language sentences. We choose the pre-trained BART model (Lewis et al., 2020) following the standard transformer architecture (Vaswani et al., 2017), which contains the encoder and decoder architecture as the denoising autoencoder pre-trained on the task of corrupted text reconstruction. The input of the encoder is the structure-aware representation of the logic forms (Section 3.2), while the target output of the decoder is the aligned textual description for the input parses.

Evaluator An evaluator indicates the logic consistent score of pairs of logical forms and textual sentences, which is vital for assessing the performance of the logic-focused text generator. In contrast to other text generation tasks, generating sentences from logical forms especially requires the evaluator to be reasonably sensitive to the subtle logic shifts of the model predictions. For instance, deleting negation words such as 'not' is fatal for our task by significantly compromising the logic consistency. Therefore, we exploit a binary classification architecture similar to the BART-based natural language inference model (Lewis et al., 2020) as our evaluator to compute the consistency between the pairs of logical form and text $[L, Q]$. The input of the encoder is the concatenation of the L and Q appended an $[EOS]$ token, and the logic scores γ are computed as:

$$\gamma = \sigma(\omega([h_{d_1}, h_{d_2}, h_{d_3}...])) \tag{1}$$

where h_{d_n} denotes the last hidden states of the decoder, ω denotes the max-pooling layer, and σ is the sigmoid activation function.

3.2 Data Augmentation

As the labeled training data for both the generator and evaluator is extremely limited, we propose a data augmentation procedure to enlarge the training set by covering variations of logic forms paired with diverse natural language expressions to improve the generalizability. To be specific, our data augmentation consists of three steps as depicted in Figure 1 from a seed dataset with human annotation:

Step 1: Logic perturbation Instead of modifying the natural language sentences, we choose to corrupt the logic consistency by perturbing logical forms mainly because of two reasons: (1) The regular structures of logical forms guarantee the procedure of the logical corruption to be comparatively controllable; (2) The perturbed logical forms could be easily validated with the corresponding parser and grammar checker. The perturbations of each given logical form could be enumerated exhaustively according to hand-tuned rules to cover the following logic inconsistencies:

- **Logic shift:** The logic shift indicates that the generated text logically distinct from the input logical forms, such as turning the assertive sentences into negative sentences. This could be attributed to the perturbations of aggregators, operators, logic conjunction, etc.

- **Phrase and number changes:** The phrase changes mean that the generated sentence modifies the appointed phrase from the logical forms, while the number changes are that the numerical values in the logical forms are perturbed.

- **Entity insertion, deletion and swapping:** Perturbations of entities is a common drawback that most natural language generation models suffer. This includes the phenomenon that the predicted sentences neglect the entities mentioned in the logical forms, insert unrelated entities to the logical form, or mislay them.

Step 2: Inference from perturbed logic After logic perturbation, the generator could be exploited as the artificial annotator to generate the corresponding sentence for each logical form in a semi-supervised manner. Compared to the rule-based or template-based method, the recent pre-trained seq-to-seq models empirically generate the natural language sentences with better fluency and coherency. Though this method could easily create a considerable amount of labeled data meanwhile avoid the expensive human annotation, what can not be ignored is that the model-based generator naturally would introduce unexpected noise during augmentation. Therefore, the quality control for the data augmentation is one of the most crucial cornerstones for a satisfactory result.

Step 3: Dataset composition As shown in Figure 1, the example in the seed dataset is denoted as [Seed logic, Seed text], and the augmented examples are denoted as [Perturbed logic, Perturbed text]. Intuitively, we may take the augmented [Perturbed logic, Perturbed text] to be not only the training example for the generator but also the positive sample for the evaluator, while crossover pairs [Seed logic, Perturbed text] and [Perturbed logic,

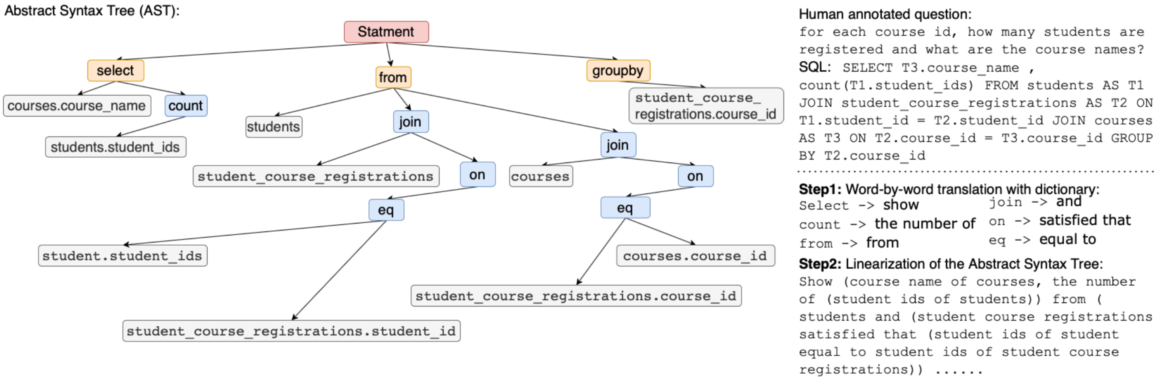

Human annotated question:
`for each course id, how many students are registered and what are the course names?`

SQL: `SELECT T3.course_name , count(T1.student_ids) FROM students AS T1 JOIN student_course_registrations AS T2 ON T1.student_id = T2.student_id JOIN courses AS T3 ON T2.course_id = T3.course_id GROUP BY T2.course_id`

Step1: Word-by-word translation with dictionary:
`Select` -> show
`count` -> the number of
`from` -> from
`join` -> and
`on` -> satisfied that
`eq` -> equal to

Step2: Linearization of the Abstract Syntax Tree:
`Show (course name of courses, the number of (student ids of students)) from ( students and (student course registrations satisfied that (student ids of student equal to student ids of student course registrations)) ......`

Figure 3: The example of word-by-word translation and linearization of structure-aware encoding.

Seed text] would be suitable negative samples for the evaluator.

3.3 Structure-aware Encoding

The logical forms normally have equivalent structured representations to precisely express the complex relations between a set of objects. For instance, the executable codes written in Python or SQL could be parsed into abstract syntax tree(AST) (Noonan, 1985) denoting the mutual relations among occurred constructs in the source code, while the knowledge bases may be converted into knowledge graphs that depict the relations between entities with directed edges. Compared to the plain text inputs, the structure-aware encoding capturing not only the sequential information from texts but also the internal logic from structural representations recently has been proved to be more effective in several Graph-to-Text tasks (Song et al., 2018; Ribeiro et al., 2020). To make full use of the intrinsic knowledge of the pre-trained BART model, we follow the similar approach proposed by Ribeiro et al. (2020) to linearize the structural representations of the SQL queries and logical forms respectively (Figure 3). Furthermore, the logical forms from different domains or datasets may vary in keywords, so normalizing them into a unified form would bridge the gaps between different logic NLG datasets and then increase the generalization ability of our framework. Hence, the logical forms would be firstly word-by-word translated into the unified intermediate semi-textual forms according to a manually annotated dictionary. Then the parenthesis is inserted into the semi-textual forms to denote the hierarchy of the correlated structured representations such as ASTs.

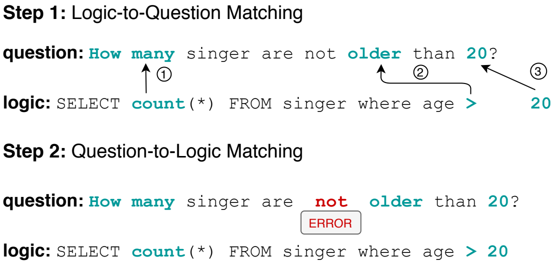

Step 1: Logic-to-Question Matching

question: `How many singer are not older than 20?`

logic: `SELECT count(*) FROM singer where age > 20`

Step 2: Question-to-Logic Matching

question: `How many singer are not older than 20?`

logic: `SELECT count(*) FROM singer where age > 20`

Figure 4: A sample of BLEC. The words marked in green are the matched tokens while the words marked in red are the tokens with no match.

4 BLEC for Logic Consistency Evaluation

Because the general-purpose automatic metrics such as BLEU, ROUGE, and BLEURT are not ideal for explicitly measuring the logic consistency, we propose BLEC, a new rule-based automatic evaluation metric called Bidirectional Logic Evaluation of Consistency. We apply a bidirectional evaluation to determine the logical consistency of pairs of logical forms and questions. The intuition behind this metric is that some key tokens such as number, operator, and keywords in the logical form should always be matched with some tokens that represent similar meanings in the question, and vice versa. An example is shown in Figure 4, BLEC first traverses the key tokens in the question, trying to find the tokens with the same meaning in the logic form to match them. Then, in step two, the sample is marked as inconsistent because there is one token with no match from the question to the logical form.

Formally, given a logical form $L = l_1, l_2, ..., l_n$ containing n word tokens and a questions $Q = q1, q2, ..., q_m$ containing m word tokens, the proposed evaluation metric performs token level

Dataset		Train	Dev	Test
SQL2Text	Generator	5600	1400	1034
	Evaluator	-	1142	1142
Logic2Text	Generator	8566	1095	1092
	Evaluator	-	1041	1041

Table 1: The statistics of the SQL2Text and Logic2Text dataset.

matching on l_i and q_j to test the consistency. To be specific, the matching procedure contains two steps, i.e. matching from L to Q as well as matching from Q to L. In step one, each key token l_i^{key} in L tries to match with the tokens in Q. In step two, each key token q_j^{key} in Q tries to match with the tokens in L. If no tokens are found that could be matched with any key tokens in either step one or step two, the sample will be marked as negative, vice versa. The final score is the accuracy of all the samples:

$$\textsc{Blec} = \frac{\sum_{s \in S} match(s)}{|S|} \tag{2}$$

Where S denotes the dataset while $match(*)$ is the matching function with binary output, i.e. 1 for positive and 0 for negative.

Compared with the neural network evaluator requiring data-specific training, BLEC can be easily deployed to different datasets. In our experiments, we demonstrate that BLEC can be applied to two different datasets of text generation from two types of semantic parse input, and it shows a substantial agreement with human evaluation for evaluating logic consistency between the semantic parse input and the text output (Table 2).

5 Experiment Settings

5.1 Datasets

Text generation from semantic parses has different forms depending on the input formal representation. To demonstrate that our SNOWBALL and BLEC can be applied to different types of inputs, we study two tasks: (1) SQL2Text with the SQL query as the input and (2) Logic2Text with the logic forms as the input.

To this end, we make use of two existing publicly available datasets: For SQL2Text, we use the Spider dataset (Yu et al., 2018), a complex cross-domain semantic parsing and text-to-SQL dataset. Generating natural language from formal languages with abundant logic representations could be regarded as the inverse semantics parsing process. Therefore, we reverse the input and output as a dataset for the text generation from SQL queries with complicated logic. As the test set of the Spider dataset remains undisclosed, 20% of the original Spider training set is converted into a development set, and 80% of the training set remains to be the training set, and the original development set is exploited as the test set for our SQL2Text task. For Logic2Text, we use an existing Logic2Text dataset from Chen et al. (2020b). We pick the SENT and LOGIC_STR fields from the original Logic2Text to compose our own train data. We then change SENT to TEXTand change LOGIC_SET to LOGIC as our one keyword of each sample in the dictionary of our dataset.

In contrast, evaluating the logical consistency between logical form and text is closely related to the sequence classification tasks such as fact verification and natural language inference (NLI). According to the best of our knowledge, there is no existing dataset for evaluating the logical consistency between logical form and generated text. Therefore, we simplified the logic evaluation as a two-sequence binary classification problem and then construct the dataset with the development set and test set dedicated for our proposed evaluator. The dataset is constituted from the development and test set of Spider and Logic2text by three methods: (i) The [logical form, Text] pairs in the two datasets are regarded as positive samples; (ii) The human-labeled negative samples by intentionally introducing the logical inconsistency to the known [logical form, Text] pairs in the two datasets; (iii) The manually scored [logical form, Text] prediction given by the trained generator on the two datasets which contain both positive and negative samples. As for the human-labeled negative samples, we attempt to cover the possible logic perturbations mentioned in section 3.2 with minimum modification to the original [logical form, Text] pairs. For example, a coincident pair [`SELECT avg(age) FROM dogs`, What is the **average** age of dogs?] would be corrupted into [`SELECT avg(age) FROM dogs`, What is the **oldest** age of dogs?]. Table 1 summarizes the statistics of each dataset for both generator and evaluator, respectively.

5.2 Baselines and Implementation Details

The baselines for assessing the performance of SNOWBALL framework are the attention-based LSTM machine translation model (Tao et al., 2019), and the single-pass trained models which are the models trained before performing SNOWBALL iteration. For instance, the `BART-large` generator trained in the second SNOWBALL iteration would be compared to the identical `BART-large` generator in the zero SNOWBALL iteration. The hype-parameter settings of the models trained on *SQL2Text* and *Logic2Text*, mostly follow the default setting of BART model from Huggingface (Lewis et al., 2020; Wolf et al., 2020). However, the learning rate of evaluator and tokenizer are slightly different, namely the learning rate of evaluator on *SQL2Text* is 2e-5 for `BART-base` and is 5e-6 for `BART-large`, while the learning rate of evaluator on *Logic2Text* is 1e-5 for both `BART-base` and `BART-large`.

5.3 Multitask Learning

Due to the lack of data of logic NLG, intuitively collaborative training on *SQL2Text* and *Logic2Text* dataset may prevent the models from bias fitting to their confined training data. Aside from the standard special separators used by BART tokenizer, we further introduce *[SQL]* and *[logic]* tokens to be the control codes to indicate if one sample is from *SQL2Text* or *Logic2Text* dataset, similar as (Keskar et al., 2019). For each sample fed into the BART model, a corresponding control token is contacted in the front of the input logical form according to that sample source. Therefore, the distribution $p(Q_{SQL}|L_{SQL}, [SQL])$ of the *SQL2Text* models and $p(Q_{Logic}|L_{Logic}, [logic])$ of the *Logic2Text* models could be learned respectively during the backpropagation that takes the control tokens into account, while training the generator and evaluator in the MTL fashion.

5.4 Human Evaluation

To evaluate if the sentence generated by the model is logically consistent, we randomly sample 90 questions from the test set of Spider and a test set of logic2text separately to form a human evaluation set. The samples of each setting will be divided into two parts and assigned to two different annotators. Each part contains 10 overlap and 40 non-overlap examples, which means one person has to label 50 samples for a setting. As for the human evaluation, the annotators label the [logical form, text] as True or False based on two criteria: (1) the logic consistency between logical form and text; (2) The grammaticality of the text. After labeling, we estimate the accuracy of the model predictions by computing the expectation of the true labels from 80 non-overlap data. To prove the consistency of the annotators, we use the 10 overlap data to calculate the cohen kappa score. Only if the kappa score is over 0.4 which implies that this estimated accuracy is valid, the results would be reported. In Table 4, we only human annotated the results given by the models trained without snowball iteration and trained with 4 snowball iterations. It demonstrates the correlation between human evaluation and BLEC metrics in these two time steps instead of directly evaluating the improvement of model performance.

6 Results and Analysis

6.1 Correlation Analysis on BLEC

To show that BLEC is consistent with human judgment, we test the Pearson correlation between the BLEC score and the human evaluation result. We also include ROUGE and BLEU for comparisons. Therefore, we apply these four automatic metrics (BLEU, ROUGE, BLEURT, BLEC) to a human-labeled dataset and compare the evaluation results. This dataset is constructed by extracting 50 samples from each of the different Snowball iterations, 15 iterations in total. As shown in Table 2, the logic consistency between BLEC and human evaluation is 0.66 while BLEU, ROUGE, and BLEURT obtain scores below or around zero. This shows that the BLEC score is capable of testing the logical consistency between logic forms and questions.

Metrics	Pearson	p-value
ROUGE (Lin, 2004)	-0.30	0.28
BLEU (Papineni et al., 2002)	-0.24	0.40
BLEURT (Sellam et al., 2020)	+0.02	0.94
BLEC	**+0.66**	**0.007**

Table 2: The correlation with human annotation of BLEU, ROUGE, BLEURT, and our proposed BLEC.

6.2 Effectiveness of Snowball Framework

Generator The experimental results of the generator in our SNOWBALL framework are shown in Table 4. We found that the SNOWBALL training framework empirically leads to the improvement of

Input SQL:	`SELECT count(*), max(Percentage) FROM country_language` `WHERE LANGUAGE = "Spanish" GROUP BY CountryCode`
Pre-processed SQL:	(the number of (all items)) , (the maximum of (percentage)) that belongs to (countrylanguage) , that have ((language) equal to (spanish)) , grouped by (countrycode)
Label:	What is the total number of countries where **Spanish** is spoken by the **largest** percentage of people?
BART-base	the number and percentage of languages that are **Spanish** for each country code.
+snowball (iteration = 4)	the number of languages and **maximum** percentage of languages in each country code?
+multi-tasking	How many languages does **Spanish** have in each country code?
+snowball (iteration = 4)	Show the number and **maximum** percentage for each country code.
BART-large	How many languages are there in each country and what is the percentage of the language spoken in that country?
+snowball (iteration = 4)	Find the number of languages and **maximum** percentage of **Spanish** for each country code.
+multi-tasking	Find the number and percentage of speakers of **Spanish** in each country code.
+snowball (iteration = 4)	Find the number and **highest** percentage of speakers of **Spanish** for each country code

Table 3: Example outputs from different models with or w/o performing the MTL and SNOWBALL iteration.

	SQL2Text Test Set								
Metrics	BLEC						Human		
Snowball	-	1	2	3	4	5	-	4	κ
LSTM Seq2Seq	22.6								
BART-base	76.4	78.6	78.5	**84.1**	79.7	78.1	22	45	0.69
+MTL	89.1	**89.5**	89.2	88.9	88.6	88.1	66	68	0.5
BART-large	91.8	91.3	**93.7**	91.8	93.2	93.0	75	74	0.7
	Logic2Text Test Set								
Metrics	BLEC						Human		
Snowball	-	1	2	3	4	5	-	4	κ
LSTM Seq2Seq	41.1								
BART-base	87.9	86.1	**88.6**	87.4	87.7	87.8	83	85	0.48
BART-large	86.7	**87.8**	85.2	87.1	86.0	88.5	86	78	0.48

Table 4: The results of SNOWBALL generator using BLEC and human evaluation over different iterations.

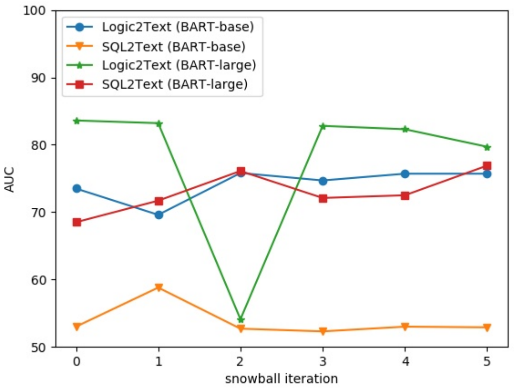

Figure 5: The result of SNOWBALL evaluator based on the AUC scores on the test set. The iteration = 0 means the model is under the regular training procedure as described in Figure 2.

the logic consistency. Evaluated by our proposed BLEC metric, the performance of `BART-base` generator improves the logic consistency by 10.1% on SQL2Text and by 0.7% on Logic2Text. Similarly, the performance of `BART-large` generator acquires the improvement by 2.1% on SQL2Text and by 1.2% on Logic2Text. Under the MTL setting on SQL2Text, the logic faithfulness of the`BART-base` generator is further enhanced by 16.6% compared to single-pass training, and is even boosted by 17.1% by combing with SNOWBALL training.

Evaluator The results of the evaluator in our SNOWBALL framework are illustrated in Figure 5. Empirically the snowball framework is more effective to the evaluators base on `BART-base` than `BART-large`, this is likely because that the `BART-large` models have already obtained enough intrinsic knowledge to accurately judge the validness of the [Logical form, Text] pairs. The data augmentation procedure of the SNOWBALL framework may introduce unexpected noise to the evaluators, which may cause a catastrophic reduction in terms of AUC and other metrics. On the other hand, the snowball framework indeed enhances the performance of the evaluator based on relative inferior `BART-base` by improving the performance on the *SQL2Text* by 10.9% as well as the *Logic2Text* by 3.1%. These results indicate that our proposed SNOWBALL framework is most suitable for tasks suffering from both domain data scarcity and the lack of external knowledge.

6.3 Case Study

Table 3 shows example outputs from our model with different settings. Apparently, in this case, the entity **Spanish** and the aggregator **maximum** are the touchstones for evaluating the logic consistency of each model. The prediction from the BART-large based generator trained under both snowball and multi-tasking frameworks simultaneously is the only one that acquires the seamless sentences from the input SQL. Furthermore, we also notice that multi-tasking learning significantly alleviates the artifacts within the generated text. Based on the fact that, compared to the vanilla generators, the generator solely trained with snowball framework would enhance the logic consistency but also increase an unnatural sense to the generated sentences at the same time, we may argue that there is a trade-off between fluency and logic consistency of our purposed snowball framework. The model-level modification may collaboratively enhance the fluency and logic consistency of the NLG, which we would remain for future studies.

7 Conclusion

In this paper, we propose SNOWBALL, a neural network-based framework to augment the data alternatively by a generator, and an evaluator. In addition, we propose BLEC, an automatic evaluation metric that could evaluate the logic consistency between question and logic forms by directional matching. We also formulate two datasets and the experimental results show the effectiveness of the proposed framework. This method is applicable to other Data-to-Text tasks, because domain-specific rules for perturbations can be derived for most structural data with pre-defined structures or grammar.

8 Ethics Statement

The datasets we use are built by selecting and processing from two datasets that are open to the public, separately. The data sources we utilize to construct our datasets are Spider and Logic2Text, two complex and cross-domain text-to-SQL datasets. Besides, we use three experts to annotate about 500 data beyond the original dataset. We admit that some biases may still exist in our datasets, even though we have double-checked the data they annotated and the data from the original datasets.

Authors with SQL expertise annotate and verify our datasets through 1) selecting about 500 representative samples from the original dataset, 2) changing the entities of the samples, 3) using three different labels to mark which type of change has been done to the sentences, and 4) double-checking the quality of the data we annotate.

We conduct several experiments of different settings on our AWS server, with 8 Tesla V100 GPUs, to test the efficiency of our models. To be more specific, our experiments contain two different types. The first type of them is that we train both generator and evaluator using SQL2Text or Logic2Text. The second type of them is that we utilize SQL2Text and Logic2Text to train generators, use one of them to train evaluators in the first epoch, and train the next several epochs with both of them.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao, and Rui Yan. 2019. Stick to the facts: Learning towards a fidelity-oriented E-commerce product description generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4959–4968, Hong Kong, China. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, Online. Association for Computational Linguistics.

Marco Damonte and Shay B Cohen. 2019. Structural neural encoders for amr-to-text generation. *arXiv preprint arXiv:1903.11410*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Claire Gardent and Agnes Plainfossé. 1990. Generating from a deep structure. In *COLNG 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *CoRR*, abs/2004.06577.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2020. Deep code comment generation with hybrid lexical and syntactical information. *Empir. Softw. Eng.*, 25(3):2179–2217.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*.

Georgia Koutrika, Alkis Simitsis, and Yannis E. Ioannidis. 2010a. Explaining structured queries in natural language. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 333–344.

Georgia Koutrika, Alkis Simitsis, and Yannis E Ioannidis. 2010b. Explaining structured queries in natural language. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 333–344. IEEE.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Abhijit Mishra, Anirban Laha, Karthik Sankaranarayanan, Parag Jain, and Saravanan Krishnan. 2019. Storytelling from structured data and knowledge graphs : An NLG perspective. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 43–48, Florence, Italy. Association for Computational Linguistics.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 977–988.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: translating sparql queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*, pages 977–988.

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, Brussels, Belgium. Association for Computational Linguistics.

Robert E. Noonan. 1985. An algorithm for generating abstract syntax trees. *Comput. Lang.*, 10(3/4):225–236.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *CoRR*, abs/2007.08426.

Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing amr-to-text generation with dual graph representations. *arXiv preprint arXiv:1909.00352*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Yifeng Tao, Bruno Godefroy, Guillaume Genthial, and Christopher Potts. 2019. Effective feature representation for clinical text concept extraction. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 1–14, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Hongmin Wang. 2019. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan. Association for Computational Linguistics.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21, Tilburg University, The Netherlands. Association for Computational Linguistics.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020a. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086, Online. Association for Computational Linguistics.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020b. Towards faithful neural table-to-text generation with content-matching constraints.

Michael White. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pages 3911–3921.

A BLEC Details

BLEC uses bidirectional keyword matching to detect the logic consistency. Table 5 shows several sample cases for constructing BLEC metrics. As shown in the table, the first column shows the type of matching rules. “Special” means that these rules are only contained in one of the datasets. Then, the following 3 columns display the tokens in different languages. Using the tokens, the algorithm can detect if the question matches the parse. For instance, given a pair of question and SQL parse, the algorithm could check if “MAX” is in SQL parse. If so, it will try to match one of the possible tokens corresponding to “MAX”, i.e. largest/ greatest, etc., in the question, and vice versa. It is worth noting that, this table only shows a small part of the algorithm, however, all the rules can be classified as one of the three types.

Type	Spider	Logic2text	Natural Language
Negation	NOT	not_eq	not/ none...
Operator	>	filter_greater	larger/ more/ greater...
	<	filter_smaller	smaller/ less/ fewer...
	MAX	max	largest/ greatest...
	COUNT	count	total/ how many...
Special	ASC	-	ascending/ fewest...
	DESC	-	descending/ highest...
	-	most_str_eq	majority/ most...

Table 5: Sample rules for BLEC.

Inducing Semantic Roles Without Syntax

Julian Michael[1] **and Luke Zettlemoyer**[1,2]

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Facebook AI Research
`{julianjm,lsz}@cs.washington.edu`

Abstract

Semantic roles are a key component of linguistic predicate-argument structure, but developing ontologies of these roles requires significant expertise and manual effort. Methods exist for automatically inducing semantic roles using syntactic representations, but syntax can also be difficult to define, annotate, and predict. We show it is possible to automatically induce semantic roles from QA-SRL, a scalable and ontology-free semantic annotation scheme that uses question-answer pairs to represent predicate-argument structure. By associating arguments with distributions over QA-SRL questions and clustering them in a mixture model, our method outperforms all previous models as well as a new state-of-the-art baseline over gold syntax. We show that our method works because QA-SRL acts as *surrogate syntax*, capturing non-overt arguments and syntactic alternations, which are central motivators for the use of semantic role labeling systems.[1]

Labels	**Questions**	
`A1` (98%)	What is given?	.30
	What does something give something?	.21
	What does something give?	.20
	What is something given?	.11
`A0` (98%)	What gives something?	.44
	What gives something something?	.27
	What gives something to something?	.08
`A2` (94%)	What is given something?	.28
	What does something give something to?	.18
	What does something give something?	.14
	What is given?	.09
	What is something given to?	.07
`TMP` (46%),	When does something give something?	.20
`ADV` (22%),	How does something give something?	.09
`MNR` (12%)	When is something given?	.09
	When is something given something?	.09
`PNC` (30%),	Why does something give something?	.18
`ADV` (22%),	Why does something give up something?	.07
`TMP` (14%)	Why is something given something?	.07

Table 1: Roles for *give* produced by our final model. Core arguments are captured almost perfectly, exhibiting both passive and dative alternations.

1 Introduction

Semantic role labeling (SRL) requires extracting propositional predicate-argument structure from language, i.e., *who* is doing *what* to *whom*. Applications of SRL include information extraction (Christensen et al., 2011), machine reading (Wang et al., 2015), and model analysis (Tenney et al., 2019; Kuznetsov and Gurevych, 2020), and semantic roles form the backbone of many more general meaning representations (Banarescu et al., 2013; Abend and Rappoport, 2013).

The primary challenge, and promise, for SRL systems is to distill syntactically variable surface structures into semantic predicate-argument structures from an ontology (Palmer et al., 2005; Baker et al., 1998). However, ontologies and their associated training data require time and expertise to annotate and do not readily generalize to new domains, limiting their broad-coverage applicability. Prior work towards mitigating this problem includes unsupervised induction of semantic roles from syntactic representations (Lang and Lapata, 2010). However, the need for formal syntactic supervision retains some of the annotation and generalization difficulties of supervised SRL, and it has proven difficult to do much better than a simple syntactic baseline (Lang and Lapata, 2011). An alternative is to use an ontology-free annotation scheme like QA-SRL (He et al., 2015), which represents roles with natural language questions. While QA-SRL can be annotated at large scale (FitzGerald et al., 2018), many different QA-SRL questions may correspond to the same role, making it more

---

[1] Code, models, and a web interface to explore the results are available at `https://github.com/julianmichael/qasrl-roles`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4427–4442
August 1–6, 2021.

The plane was ***diverting*** *around weather formations over the Java Sea when contact with air traffic control (ATC) in Jakarta was* ***lost****.*

wh	**aux**	**subj**	**verb**	**obj**	**prep**	**obj2**	**?**	**Answer**
What	was		being diverted		around		?	*weather formations*
What	was		diverting				?	*The plane*
What	was		being diverted				?	*The plane*
What	was		lost				?	*contact with air traffic control*
Where	was	something	lost				?	*over the Java Sea*

Table 2: Example QA-SRL question-answer pairs from the development set of the QA-SRL Bank 2.0 (FitzGerald et al., 2018). Questions may be represented in a verb-agnostic way by recording the form of the verb in the **verb** slot (e.g., *stem*, *past participle*). Note that the syntax used in questions may differ from the syntax in the source sentence, for example in the above questions using *diverted* in its passive form.

difficult to use in downstream tasks.

We show how to overcome this difficulty, by automatically inducing an ontology of semantic roles corresponding to clusters of QA-SRL questions (see Table 1 for an example clustering). We use a model to predict a distribution over QA-SRL questions associated with each argument in a corpus, and cluster them to maximize likelihood under a simple model we call a *Hard Unigram Mixture*. Our model can be effectively optimized both by EM and greedy methods, which affords the benefits of tunable hierarchical clustering without sacrificing scalability (Section 3).

Experiments in semantic role induction (Section 4) show that our method outperforms all previous methods in the literature, as well as a new state-of-the-art baseline over gold syntax. This is despite requiring no formal syntactic supervision or theory, where the formalism used by previous work is highly informative of gold standard semantic roles (Section 5). We also present a detailed analysis (Section 6) showing why our method works: QA-SRL acts as *surrogate syntax*, removing (role-irrelevant) syntactic variation in the source text such as that from *non-overt* arguments (e.g., phrases extracted from relative clauses), while itself exhibiting (role-relevant) syntactic alternations which capture the behavior of verbal predicates (Table 1). Taken together, these results paint a path towards on-the-fly, data-driven construction of useful, interpretable ontologies of semantic structure.

2 Task Setting

The input to our task is a set of natural language sentences, where a subset of the tokens are marked as *predicates*. Each predicate has a set of *arguments*, and each argument x corresponds to a set of spans $x = \{s_1, \dots, s_m\}$ in the predicate's sentence.[2]

An ontology of semantic roles is a set of *frames* (corresponding to semantic predicates), and each frame has a set of associated *roles* (corresponding to participants in the event or state denoted by its frame). There may also be a set of *modifier roles* (e.g., location or time) which can appear with any frame. In supervised semantic role labeling, each predicate in the input data must be assigned to one of the frames in a given ontology, and each of a predicate's arguments must be assigned roles from its frame (or modifier roles). In semantic role induction, our task is to produce both the ontology and these assignments.

We follow prior work (Lang and Lapata, 2010) in treating semantic role induction as a clustering problem and assuming a single frame per predicate lemma.[3] Given input data marked with predicates and their arguments, we cluster the arguments for each predicate into sets corresponding to semantic roles. We may then compare these clusters to gold labels using clustering metrics (Section 4.3).

---

[2] Previous work (Lang and Lapata, 2010) assumes a syntactic dependency tree and marks each argument by its syntactic head, which allows for features based on argument lemmas and dependency paths. We instead assume sets of argument spans, but no syntax tree; this allows for features based on spans (such as QA-SRL questions). Both approaches are ways of featurizing the same gold arguments.

[3] Some ontologies, like FrameNet (Baker et al., 1998), define frames that span multiple lemmas (e.g., *buy* and *sell* share a *Commercial Transaction* frame), whereas others like PropBank (Palmer et al., 2005) use frames which are specific to each lemma, denoting something closer to word sense. In our case, assuming a single frame per lemma simplifies modeling and allows us to compare to previous work. However, modeling predicate sense is an important problem for future work, as we will suggest in Section 6.3.

3 Modeling

Our model treats each argument x as a set of counts of QA-SRL questions,[4] denoted $\phi(x)$. We produce these counts from a trained QA-SRL question generator (Section 3.1) and cluster them by maximizing their likelihood under a mixture model (Section 3.2) using a hybrid of flat and hierachical clustering (Section 3.3).

3.1 Generating QA-SRL Features

For each argument x of a predicate, we leverage a trained QA-SRL parser to generate pseudocounts $\phi(x)$ of simplified QA-SRL questions, which will form the input features for the clustering step.

Simplified QA-SRL Example QA-SRL questions are shown in Table 2. These questions contain information which is not directly relevant to semantic roles, such as tense, aspect, modality, and negation. Since this creates sparsity for our model, we remove it as a preprocessing step. In particular, we replace the **aux** and **verb** slot values with either *is* and *past participle* (for passive voice), _ and *present* (for active voice when **subj** is blank), or *does* and *stem* (for active voice when **subj** is present). We also replace all occurrences of *who* and *someone* with *what* or *something*.

Generating Question Counts Let p denote a predicate, s denote a span, and q denote a simplified QA-SRL question. To generate our question count vectors ϕ, we reproduce the QA-SRL question generator of FitzGerald et al. (2018), which generates a distribution $\mathrm{P}(q \mid p, s)$ over QA-SRL questions conditioned on a predicate p and answer span s in a sentence. This model uses a BiLSTM encoder, concatenating the output representations of span endpoints and feeding them into a custom LSTM decoder which models the QA-SRL slot values in sequence. We modify the model to use BERT (Devlin et al., 2019) features as input embeddings for the BiLSTM (details in Appendix A).

Recall from Section 2 that an argument x consists of a set of spans from its sentence. We generate question counts $\phi(x) \in \mathbb{R}_{\geq 0}^{|q|}$ by taking the mean

$$\phi(x) = \frac{1}{|x|} \sum_{s \in x} \mathrm{P}(q \mid p, s),$$

where $\mathbb{R}_{\geq 0}$ denotes the nonnegative real numbers and $|q|$ is the number of possible simplified QA-SRL questions. Since $|q|$ is large, to make this tractable we approximate $\mathrm{P}(q \mid p, s)$ with beam search, using a sparse representation and assigning counts of 0 to questions outside the beam.

3.2 Objective

Let $\mathbf{X} = \{x_1, \ldots, x_n\}$ be the set of input arguments for clustering. Our goal is a clustering $\mathbf{C} = \{C_1, \ldots, C_k\}$ which is a partition of $\mathbf{X}$. We model each argument's questions $\phi(x)$ as being drawn from a mixture model over latent roles, each corresponding to a cluster $C \in \mathbf{C}$. We maximize likelihood under this model, which we call a Hard Unigram Mixture, with the addition of a connectivity penalty which encourages roles not to appear twice for the same predicate instance.

The Hard Unigram Mixture (HUM) Recall that $\phi : \mathbf{X} \to \mathbb{R}^d_{\geq 0}$ assigns question pseudocounts to each $x \in \mathbf{X}$. Let π denote a probability distribution over $\{1, \ldots, k\}$ and θ a distribution over $\{1, \ldots, d\}$. We propose the *Hard Unigram Mixture* loss

$$\mathcal{L}^{\mathrm{HUM}}_{\lambda}(\mathbf{C}) = -\log \mathrm{P}(\mathbf{X} \mid \mathbf{C}) - \lambda \log \mathrm{P}(\mathbf{C}),$$

where

$$\mathrm{P}(\mathbf{X} \mid \mathbf{C}) = \prod_{i}^{k} \max_{\theta} \prod_{x \in C_i} \mathrm{P}(\phi(x) \mid \theta)$$

is the *data likelihood* and

$$\mathrm{P}(\mathbf{C}) = \max_{\pi} \prod_{i}^{k} \pi_i^{||C_i||}$$

is the *clustering likelihood*, writing $||C||$ for the sum of the ϕ counts in a cluster C. The data likelihood prefers more, smaller clusters, the clustering likelihood prefers fewer clusters, and λ is a hyperparameter that trades off between them.[5]

Connectivity Penalty Let $p(x)$ denote the predicate instance corresponding to an argument x. We propose a *connectivity penalty*

$$\mathcal{L}^{\mathrm{cp}}(\mathbf{C}) = \frac{1}{2} \sum_{i}^{k} \sum_{x_1, x_2 \in C_i} \delta(p(x_1) = p(x_2)),$$

---

[4] As of now, this model only works for English, as QA-SRL is only defined and annotated in English.

[5] Here, $\mathcal{L}^{\mathrm{HUM}}_{1}$ is equivalent to the negative log likelihood under the maximum likelihood estimate of a mixture of unigrams model (Nigam et al., 2000) constrained to hard assignments $\mathbf{C}$; hence the name *Hard Unigram Mixture*. Further theoretical and empirical comparison to prior work is provided in Appendix G.

where δ is the indicator function, which discourages clusterings where multiple arguments of the same predicate instance are assigned the same role. This assumption has also been leveraged by prior models (Lang and Lapata, 2011; Titov and Klementiev, 2012).

Loss Function Our full loss is then

$$\mathcal{L}_{\lambda}(\mathbf{C}) = \mathcal{L}_{\lambda}^{\text{HUM}}(\mathbf{C}) + \mathcal{L}^{\text{cp}}(\mathbf{C})$$

with the single hyperparameter λ.

3.3 Hybrid Clustering

We optimize $\mathcal{L}_{\lambda}$ in three steps: flat pre-clustering, greedy merging, and tuned splitting. This approach provides us with both the efficiency benefits of flat clustering and the relative determinism, interpretability and tunability of hierarchical clustering.

Flat Pre-Clustering For pre-clustering, we minimize $\mathcal{L}_0$ via hard EM. To avoid likelihoods of 0 in $\mathcal{L}_0^{\text{HUM}}$, we smooth our estimates of θ using a Dirichlet prior. To optimize $\mathcal{L}^{\text{cp}}$ via EM, we draw x_1 from the previous iteration's clustering in order to compute the contribution of each x_2 to the loss. With sufficiently large k, this can produce a high-precision clustering in $O(nk)$ time to serve as input to the merging step.

Greedy Merging After pre-clustering, we produce a binary cluster tree by iteratively merging pairs of clusters which greedily minimize $\mathcal{L}_0$. Since $\lambda = 0$, the loss grows monotonically when merging clusters. The loss at each merge can be efficiently updated by maintaining maximum likelihood estimates θ for each cluster.

Tuned Splitting Finally, we iteratively split the cluster tree produced by the merging stage. At each step, we split the cluster C_i with the lowest log data likelihood per item $\frac{\log \text{P}(C_i|\mathbf{C})}{|C_i|}$. We then choose the clustering which minimizes $\mathcal{L}_{\lambda}$, with $\lambda > 0$ tuned during model development.[6]

4 Experimental Setup

Data We run experiments on the distribution of PropBank (Palmer et al., 2005) provided for the CoNLL 2008 Shared Task (Surdeanu et al., 2008). We use the same setup as previous work, removing arguments annotated with reference (R-) and continuation (C-) roles, keeping only verbal predicates,[7] and using the development set for model development and the training set for testing.

Our one preprocessing difference from previous work is that instead of using the dependency-based SRL annotations provided in the CoNLL 2008 dataset, we use full answer spans, which we reconstruct by aligning the CoNLL 2008 data back to the original annotations in the Penn Treebank (Marcus et al., 1993) and PropBank.[8]

4.1 Models

***HUM of QA-SRL Questions* (HUM-QQ)** We train a QA-SRL parser on the expanded set of the QA-SRL Bank 2.0 (FitzGerald et al., 2018) using the architecture described in Section 3.1. In the pre-clustering step, we estimate $k = 100$ clusters. For tuned splitting, we choose λ to maximize performance on the development set. Hyperparameters are detailed in Appendix B.

SYNTF This model assigns each argument to a cluster corresponding to the label of its syntactic dependency to its parent, using the syntactic formalism provided in CoNLL 2008 Shared Task data. Past work has found SYNTF to be a strong baseline (Lang and Lapata, 2011).

Prior Work We compare to Bayesian generative modeling (Titov and Klementiev, 2012, BAYES), which is state-of-the-art on gold syntax, and an embedding-based method (Luan et al., 2016, SYMDEP/ASYMDEP) which is state-of-the-art using automatic syntax. These as well as all other prior approaches (e.g., Lang and Lapata, 2011; Titov and Khoddam, 2015; Woodsend and Lapata, 2015) crucially rely on syntactic features.

4.2 Auxiliary Clustering Rules

For SYNTF and HUM-QQ, we experiment with several auxiliary clustering rules.

Lexical Rules We employ three lexical rules, each producing a separate cluster for all arguments whose spans exactly match a phrase contained in the rule's lexicon. Our rules are for negation (5

---

[6] A comparison of this method against a constant-k baseline and oracle upper bound is given in Appendix E.

[7] While we ignore nominal predicates, our method naturally generalizes to nominalizations, which are provided with QA-SRL annotations in QANom (Klein et al., 2020).

[8] Using gold spans is necessary in order to compare to previous work and use the CoNLL 2008 dataset for evaluation of role induction. In a more realistic setting where gold argument spans are not available, we could use the span detector of FitzGerald et al. (2018) to construct argument spans.

phrases), modals (23 phrases), and discourse modifiers (55 phrases). These lexica were written to correspond to the `AM-NEG`, `AM-MOD`, and `AM-DIS` roles on the basis of the PropBank annotation guidelines (Babko-Malaya, 2005) and development set.[9]

Passive to Active Conversion We also propose a syntactic rule that applies only to SyntF, where we the transform the dependencies as follows:

- The `LGS` label, meaning "logical subject," is a dependency label given for *by*-phrases modifying a passive verb whose object denotes what is normally the subject of the verb's active form (Surdeanu et al., 2008). We change this to `SBJ`.
- Passive voice can be detected when the predicate verb is in past participle form (part-of-speech tag `VBN`) and its syntactic parent is a *be*-verb (part of speech `VC`, lemma "be"). In these cases, we change the syntactic label of any `SBJ` dependents into `OBJ`.

4.3 Metrics

Purity/Collocation To compare with previous work, we follow Lang and Lapata (2010) in using purity and collocation based F1 score for our main evaluation. Purity measures cluster homogeneity: it assigns to each cluster the gold label for which it has the most points, and then measures the proportion of points which have their cluster's assigned label. Collocation measures cluster concentration: it assigns each gold label to the cluster which contains the most of its points, and then measures the proportion of points which are in their gold label's assigned cluster. These are calculated independently for each verb and averaged, weighing each verb by its number of argument instances. The harmonic mean of the final results is reported as an F1 score.

B^3 For deeper analysis, we use the B^3 (*B-cubed*) family of clustering metrics (Bagga and Baldwin, 1998). B^3 precision and recall are the precision and recall of each point's predicted cluster against its gold cluster, averaging over points. In comparison to purity and collocation, these metrics are tougher and more discriminative between clusterings, respecting important constraints like the cluster completeness constraint of Rosenberg and Hirschberg (2007), among others (Amigó et al., 2009). B^3 also allows us to reliably report scores along slices of the data for analysis purposes, as well as account for each slice's contribution to the total error. We report full B^3 results for our models in Appendix F and encourage future work to use these as the primary metrics.

Model	**PU**	**CO**	**F1**	**ΔF1**
		Gold Syntax		
SyntF	81.6	77.8	79.6	0.0
+ lex	85.2	79.8	82.4	+2.8
+ pass→act	83.6	80.8	82.2	+2.6
+ all rules	87.3	**83.1**	**85.2**	**+5.6**
Bayes (SotA)	**88.7**	78.1	83.0	+3.4
AsymDep	85.6	78.3	81.8	+2.2
		Automatic Syntax		
Bayes	86.2	72.7	78.8	-0.8
SymDep (SotA)	81.9	**76.6**	**79.2**	-0.4
		Automatic QA-SRL		
HUM-QQ	80.9	83.4	82.1	+2.5
− conn. penalty	79.0	82.7	80.8	+1.2
+ lex	**85.4**	**88.8**	**87.1**	**+7.5**

Table 3: Main results. The addition of a few simple rules to the SyntF baseline puts it significantly above existing approaches, and incorporating these rules into our QA-SRL-based model pushes performance even further, despite not using gold syntax at all. Evaluation numbers for baselines besides SyntF are drawn directly from prior work.

5 Results

Main results are shown in Table 3. Our auxiliary rules put SyntF significantly above the state of the art for gold syntax (with 85.2 F1 versus 83.0). HUM-QQ surpasses it with 87.1 F1 in the best case, despite not using gold syntax at all.

5.1 A Stronger Syntactic Baseline

For SyntF, the addition of either lexical (negation, modal, and discourse) rules or the passive-to-active conversion produce competitive models, covering over 75% of the gap from baseline to Bayes. Used together, our rules bring the score to 85.2 F1, surpassing Bayes by 2.2 points. Table 5 breaks down these improvements by measuring B^3 performance on relevant roles.

For the lexical rules, we find that the negation and modal rules nearly completely capture their

---

[9] Full lexica for these rules are provided in Appendix C.

B^3 F1	`A0`	`A1`	`A2`	`A3`	`A4`	Args	`TMP`	`ADV`	`MNR`	`LOC`	`PNC`	`CAU`	Mods	All
SyntF + lex	78	71	63	**55**	**67**	73	**87**	**51**	**60**	**81**	**65**	**67**	**74**	74
HUM-QQ + lex	**90**	**87**	**69**	54	65	**85**	78	39	50	55	56	36	61	**82**
% Err \ Freq	.26	.37	.09	.01	.01	.74	.07	.04	.03	.03	.01	.01	.18	1.0
SyntF + lex	.23	.41	.13	.03	.02	.79	.04	.06	.04	.02	.01	.01	.18	1.0
HUM-QQ + lex	.15	.26	.14	.04	.02	.61	.08	.12	.07	.06	.02	.02	.38	1.0

Table 4: B^3 F1 scores on the training set for the most common labels, excluding `NEG`, `MOD`, and `DIS`.

	B^3 F1 Score				
Model	`NEG`	`MOD`	`DIS`	`A0`	`A1`
SyntF	41	45	50	78	71
+ all rules	**98**	**98**	**80**	**83**	**78**
Frequency	.01	.04	.03	.26	.37

Table 5: Breakdown of B^3 F1 scores on the training set for the labels most relevant to our auxiliary rules. The lexical rules capture `AM-NEG`, `AM-MOD`, and `AM-DIS` very well, and the active/passive rule significantly improves performance on `A0` and `A1`, which are by far the most frequent role labels in the data. A rule-by-rule performance breakdown is provided in Appendix D.

roles, with the discourse rule providing significant improvements as well. In contrast, previous models have struggled with these roles, as reported by Lang and Lapata (2011, Table 4, `NEG` and `DIS` roles). However, this is better seen as a shortcoming of the evaluation than the models: these roles are relatively uninteresting from the perspective of semantic role induction, as they are closed-class, not specific to particular predicates, and don't correspond to a semantic argument or modifier of the event denoted by the predicate. It might have been reasonable to exclude these arguments from the task at the outset, but instead, using our rules can mostly account for them while maintaining some comparability to prior work.

The passive-to-active conversion also produces a sizable gain, particularly on the core roles `A0` and `A1` (Table 5). Titov and Klementiev (2012) informally note that the Bayes model learns some syntactic alternations; of these, the passive alternation is perhaps the most impactful as it can apply to any transitive verb. What we've found is that a simple rule accounting for the passive construction in the syntax provided to the Bayes model can account for a large majority of its gains.

These results provide extra context in which to interpret the existing literature on semantic role induction. The fact that our simple auxiliary rules bring the syntactic baseline beyond the existing state of the art raises questions about whether the performance differences between previously published models are due to their relative abilities in capturing their intended phenomena — such as selectional restrictions and distributions over argument heads (Lang and Lapata, 2014) — or capturing these rules. It is not clear how much of the 5.2 F1 gain over SyntF from our auxiliary rules is redundant with previous models. It seems likely that applying our rules to them would produce a result competitive with HUM-QQ, but it would still rely on gold syntax. Our focus is the utility of QA-SRL as features; indeed, it is also conceivable that applying a hierarchical model like Bayes to QA-SRL features would bring further improvements as well.

5.2 Superiority Without Syntax

HUM-QQ benefits disproportionately from the lexical rules, with a 5 F1 gain as opposed to the 2.8 F1 gain for SyntF. This is because PropBank's `NEG`, `MOD`, and `DIS` arguments almost never occur in QA-SRL, so they get nonsense questions from the model (see Appendix J, Table 12).[10] However, even the baseline model with no lexical rules or connectivity penalty surpasses the performance of the baselines using automatic syntax, all of which fall short of SyntF on gold.[11] With these additions, HUM-QQ sets a new state of the art beyond our enhanced SyntF baseline, with 87.1 F1.

Table 4 compares our model to SyntF + lex on the most common roles using B^3. HUM-QQ greatly improves over SyntF on core arguments

---

[10] In practice, when using arguments predicted by a QA-SRL span detector (FitzGerald et al., 2018), we can remove the lexical rules entirely since the corresponding arguments will not be present.

[11] To be fair, these models use the automatic parses provided with the CoNLL 2008 data, which were produced by MaltParser (Nivre et al., 2006) at the time. Using state-of-the-art methods to predict the parses today would almost certainly improve the semantic role induction results, but probably not past gold parses.

(73→85 F1), but performs worse on modifiers (74→61). Since core arguments make up 74% of arguments in the corpus, HUM-QQ brings a large improvement overall (74→82) and core arguments still account for a majority of its error (at 61%).

SYNTF's high performance on modifiers can be traced back to representational choices in the CoNLL 2008 Shared Task syntax (Surdeanu et al., 2008), which uses several dependency types that are semantic in nature, such as `TMP`, `LOC`, `MNR`, and `DIR`, among others. These often correlate well with gold modifier role labels, especially `TMP` (87 F1) and `LOC` (81 F1).[12] This fact has led some prior work, e.g., Titov and Klementiev (2012), to use these dependency labels as clusters directly, so as to avoid the need to model modifier roles and instead focus on core arguments. Since we eschew syntactic features, we are forced to recover PropBank modifier roles from the ground up, making the task more difficult (explored more in Section 6.2).

6 What does QA-SRL Encode About Semantic Roles?

Semantic roles are traditionally characterized as abstractions over syntactic arguments and modifiers (Gruber, 1965; Fillmore, 1968). Despite their deep entanglement with syntax, we have found that significant improvements in semantic role induction are possible without explicit syntactic analysis of the sentence, instead leveraging distributions of QA-SRL questions for each argument. In this section, we show that this is because QA-SRL questions provide *surrogate syntax*, recapitulating the aspects of syntax that are important for semantic roles (Section 6.1). Where QA-SRL questions fail to capture aspects of PropBank semantic roles, this arises in part from ontological differences with PropBank on modifiers (Section 6.2) and limitations of our experimental setup ignoring predicate sense (Section 6.3).

6.1 Surrogate Syntax

HUM-QQ brings the largest improvement over SYNTF on core arguments `A0` and `A1`. To investigate this, we identify the verbs which saw the greatest increase in B^3 F1 score on each role individually. What we find is that QA-SRL works by acting as *surrogate syntax*: it removes much of the (role-irrelevant) syntactic variation in the source text, while still exhibiting (role-relevant) syntactic alternations which capture the syntactic behavior of the predicate verb.

Reducing Syntactic Variation For `A0`, the three verbs with the greatest improvement from SYNTF to HUM-QQ are *compete*, *conduct*, and *connect*, all with gaps of over 40 F1.[13] For each of these, their `A0` arguments have a wide range of syntactic functions assigned by SYNTF, with `SBJ` less than 50% of the time — despite the fact that where the `A0` role is present, it is designed to correspond to the grammatical subject (Babko-Malaya, 2005). We found that this is because these verbs frequently have *non-overt* subjects, which are not direct syntactic dependents of the predicate in CoNLL 2008 syntax (74% of a random sample of 30 sentences with `A0` arguments of these three verbs, 10 from each; see Appendix H.1). They appear in phrases like 'two competing *objectives*' (with adjectival clauses), 'urging *directors* to conduct a fair auction' (with control verbs), or '*a maze of halls* that connects film rooms' (with relative clauses). In these cases, the SYNTF baseline does poorly, as the correspondence between the `SBJ` dependency and `A0` role only holds consistently for overt subjects.

In contrast, HUM-QQ assigns the vast majority of `A0` arguments in these cases with questions that put the *wh*-word in subject position, e.g., *What competes with something?* or *What conducts?* Here, QA-SRL removes much of the syntactic variation from the source text and recovers something close to the underlying grammatical relation between the argument and the verb, while also providing information about the verb's subcategorization frames (e.g., the presence of an object in *What connects something?*), aiding in recovery of the semantic role.

Capturing Syntactic Alternations For `A1`, The verbs with the greatest improvement are *propose*, *prefer*, *price*, and *relate*, with a gap of >50 F1 between models. Of the top 50 such verbs, 48 are transitive with `A1` as the transitive object (see Appendix H.2). In these cases, the passive alternation allows the argument to be asked about in either the subject (*What is proposed?*) or object (*What does something propose?*) position. We find that QA-SRL does this, frequently combining questions

---

[12] See Lang and Lapata (2014, Table 2) for a detailed contingency table.

[13] To reduce variance from low-frequency verbs, we measure this gap after smoothing their precision and recall with 10 counts of the weighted aggregate for the model.

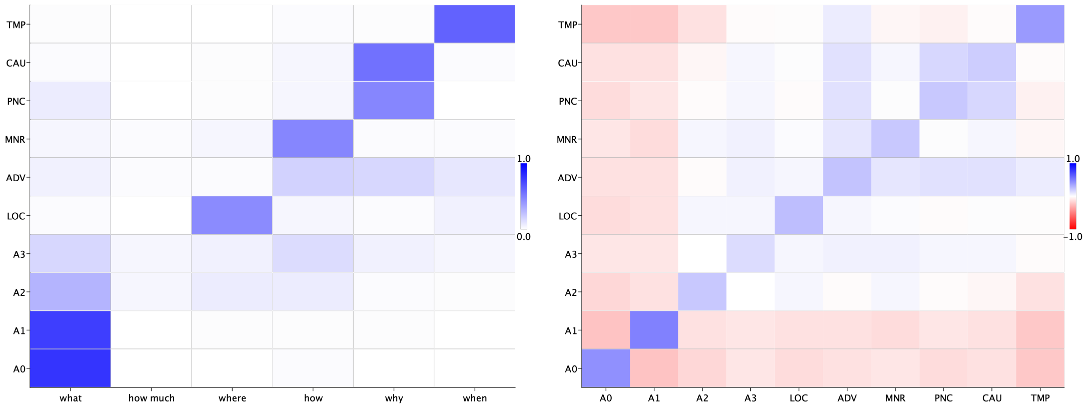

(a) Distribution of *wh*-words for each role.

(b) Normalized PMIs between gold roles.

Figure 1: Cooccurrence between gold role labels and *wh*-words in QA-SRL (left) or each other in HUM-QQ's predicted clusters (right). The distributions of *wh*-words are normalized per role, and NPMI between gold labels is chance-corrected, where negative values (red) are clustered apart more often than by chance, and positive values (blue) are preferentially grouped together.

about passive subject and active object into one role: for 62% of the top 50 verbs, the cluster corresponding to `A1` gives greater than 20% probability *each* to passive subject and active object questions. This happens because the Hard Unigram Mixture objective clusters together distributions whose uncertainty is spread over the same set of elements, which here correspond to syntactic alternations. As an example, Table 1 shows the induced clusters for *give*, which exhibit both passive and dative alternations; *give* gained 31 F1 on `A1` in HUM-QQ.

6.2 Mismatched Modifiers

HUM-QQ struggles to identify PropBank modifier roles, and it has room for improvement on trailing arguments like `A2` and `A3`. In QA-SRL, the semantics of these roles are primarily expressed by the initial *wh*-word, such as *when*, *where*, *why*, *how*, etc. Figure 1a shows the distribution of *wh*-words appearing for each role in the training set. To a large extent, each role is concentrated on a corresponding *wh*-word, but there are exceptions. `A2`, `A3`, and `AM-ADV` are widely spread between *wh*-words, and *how* and *why* account for a significant portion of questions for several roles each. See Appendix J, Table 11 for full questions.

To visualize how this affects clustering results, Figure 1b shows the normalized pointwise mutual information (NPMI; Bouma, 2009) between gold labels in HUM-QQ's predicted clusters (see Appendix I for how this is calculated). While `A0` and `A1` are distinguished well from all other roles, the trailing arguments `A2` and `A3` are not well distinguished from modifiers, reflecting the difficulty of the argument–adjunct distinction for these arguments, which often have similar meanings to modifiers and form a significant error case for supervised labelers (He et al., 2017). `AM-ADV` tends to be confused with other modifier roles, which reflects its definition in the PropBank guidelines as a sort of "catch-all" role for meanings not captured in the other modifiers (Babko-Malaya, 2005). Finally, `AM-CAU` (*cause*) and `AM-PNC` (*purpose, not cause*) tend to be confused with each other, since they both elicit *why* questions.

Argument–Adjunct Distinction Scores are significantly lower for trailing core arguments `A2-4` than for `A0` and `A1`. Since part of the problem seems to be confusion with modifier roles (Figure 1b), we conduct an oracle experiment to enforce the argument–adjunct distinction by doubling the size of the feature space to $\phi(x) \in \mathbb{R}^{2|q|}_{\geq 0}$ and projecting gold core arguments and modifiers into orthogonal subspaces.

Results are shown in Table 6 (+ gold arg/adj). The oracle boosts performance by 3 points, with particular focus on trailing arguments `A2` (69→78) and `A4` (65→78), as well as modifiers `AM-ADV` (39→47), `AM-MNR` (50→57), and `AM-LOC` (55→61). However, overall performance on modifiers is still far below the syntactic baseline. Given the coarse semantics of English *wh*-words in comparison to PropBank modifier roles (Figure 1a), it may be that finer-grained features are necessary to significantly increase performance on modifiers.

B^3 F1	`A0`	`A1`	`A2`	`A3`	`A4`	Args	`TMP`	`ADV`	`MNR`	`LOC`	`PNC`	`CAU`	Mods	All
SyntF + lex	78	71	63	55	67	73	**87**	**51**	60	**81**	65	**67**	**74**	74
HUM-QQ + lex	90	87	69	54	65	85	78	39	50	55	56	36	61	82
+ gold arg/adj	91	89	78	65	78	88	77	47	57	61	58	35	64	85
+ gold sense	91	90	74	58	75	87	80	43	55	64	63	47	65	84
+ both	**92**	**92**	**83**	**70**	**81**	**90**	81	**51**	**64**	69	**66**	46	69	**87**

Table 6: Breakdown of B^3 F1 scores on the training set for the most common labels in our ablation studies. The first two rows are repeated from Table 4.

6.3 Scrambled Senses

Despite core arguments significantly improving under HUM-QQ, they remain the largest source of error. To investigate this, we examine the verbs with the worst F1 on core arguments. The top verbs are *go*, *settle*, *confuse*, *turn*, and *follow*, with <60 F1. Half of the top 20 have 4 or more predicate senses annotated in PropBank, where different senses often manifest their roles differently: for example, the subject is `A0` when *settling with the IRS* (sense 2), but `A1` when *settling into a new job* (sense 3). To quantify this, we run an oracle experiment where we induce roles for each verb sense separately instead of each verb lemma. Results are shown in Table 6 (+ gold sense). Performance improves particularly on trailing arguments `A2`, `A3` and `A4`, which tend to differ greatly in meaning and realization for different predicate senses. A combined oracle (+ both) shows that the gains are mostly complementary with those from the argument/adjunct distinction oracle. These results suggest that future work on semantic role induction should prioritize modeling predicate senses.

7 Conclusion

We have shown that QA-SRL provides a way to do state-of-the-art semantic role induction without the need for formal syntax. It works by providing *surrogate* syntax: it captures long-distance dependencies to non-overt arguments and exhibits syntactic alternations which allow us to detect varied ways of expressing the same role. These results suggest that QA-SRL can provide some of the practical benefits of sophisticated syntactic formalisms that have separate layers of functional structure, like Combinatory Categorial Grammar (Steedman, 1996, 2000), Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994), or Lexical Functional Grammar (Bresnan et al., 2015) — but without grammar engineering or expert data annotation.

One challenge is that QA-SRL is currently only defined for English. Future work may benefit from our lessons about the utility of surrogate syntax when designing similar annotation methodologies for other languages; combining this with insights from existing work on grammar development for diverse languages (Bender et al., 2002) may be key.

While formal ontologies of semantic roles and syntax are difficult to formulate and scale, our results show how it may be comparatively feasible to formulate, scale, and build robust models for the *phenomena* that such ontologies are meant to explain. QA-SRL exhibits enough of these phenomena that a relatively simple model over it (the Hard Unigram Mixture in Section 3) yields state-of-the-art induced semantic roles which are interpretable and linguistically meaningful. This suggests that identifying and gathering supervision for more phenomena (e.g., those related to word sense or modifier semantics) in a relatively theory-agnostic way, then building models grounded in linguistic theory, may be a promising avenue for future work. This general approach has recently been applied to syntax as well, for example leveraging constituency tests (Cao et al., 2020) and naturally-occurring bracketings (Shi et al., 2021).

The fact that discrete structures can be reliably derived from ontology-free annotation schemes like QA-SRL can potentially inform future efforts to construct large-scale ontologies of semantic structure. QA-SRL has the further benefit over traditional SRL of including a broader scope of *implicit* arguments than those addressed by supervised systems, as shown by Roit et al. (2020). Taken together, our results suggest that with the right kind of annotation scheme, it should be possible to construct rich semantic ontologies in new domains, without expert curation and in a data-driven, linguistically motivated way.

Acknowledgements

Thanks to the anonymous reviewers and Victor Zhong for their helpful comments.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486.

Olga Babko-Malaya. 2005. Propbank annotation guidelines.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 79–85, Montreal, Quebec, Canada. Association for Computational Linguistics.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *COLING-02: Grammar Engineering and Evaluation*.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *GSCL*.

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2015. *Lexical-functional syntax*. John Wiley & Sons.

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Unsupervised parsing via constituency tests. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4798–4808, Online. Association for Computational Linguistics.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *K-CAP*.

Grzegorz Chrupała. 2012. Hierarchical clustering of word class distributions. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 100–104, Montréal, Canada. Association for Computational Linguistics.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *27th Annual Meeting of the Association for Computational Linguistics*, volume 16, pages 76–83. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Charles J. Fillmore. 1968. The case for case. In Emmon Bach and Robert T. Harms, editors, *Universals in Linguistic Theory*. Holt, Rinehart & Winston.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale QA-SRL parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2051–2060, Melbourne, Australia. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Jeffrey S. Gruber. 1965. *Studies in Lexical Relations*. Ph.D. thesis, Massachusetts Institute of Technology.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ayal Klein, Jonathan Mamou, Valentina Pyatkin, Daniela Stepanov, Hangfeng He, Dan Roth, Luke Zettlemoyer, and Ido Dagan. 2020. QANom: Question-answer driven SRL for nominalizations. In *Proceedings of the 28th International Conference* *on Computational Linguistics*, pages 3069–3083, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ilia Kuznetsov and Iryna Gurevych. 2020. A matter of framing: The impact of linguistic formalism on probing results. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 171–182, Online. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2011. Unsupervised semantic role induction via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1117–1126, Portland, Oregon, USA. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2014. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics*, 40(3):633–669.

Kenton Lee, T. Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *ArXiv*, abs/1611.01436.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151.

Yi Luan, Yangfeng Ji, Hannaneh Hajishirzi, and Boyang Li. 2016. Multiplicative representations for unsupervised semantic role induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 118–123, Berlin, Germany. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Julian Michael, Jan A. Botha, and Ian Tenney. 2020. Asking without telling: Exploring latent ontologies in contextual representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6792–6812, Online. Association for Computational Linguistics.

K. Nigam, A. McCallum, S. Thrun, and Tom Michael Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39:103–134.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Tianze Shi, Ozan İrsoy, Igor Malioutov, and Lillian Lee. 2021. Learning syntax from naturally-occurring bracketings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2941–2949, Online. Association for Computational Linguistics.

Noam Slonim and Naftali Tishby. 1999. Agglomerative information bottleneck. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 617–623, Cambridge, MA, USA. MIT Press.

R. Srivastava, Klaus Greff, and J. Schmidhuber. 2015. Training very deep networks. In *NIPS*.

Mark Steedman. 1996. *Surface Structure and Interpretation*. MIT Press.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, Denver, Colorado. Association for Computational Linguistics.

Ivan Titov and Alexandre Klementiev. 2012. A Bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22, Avignon, France. Association for Computational Linguistics.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 700–706, Beijing, China. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2015. Distributed representations for unsupervised semantic role labeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2482–2491, Lisbon, Portugal. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

A QA-SRL Question Generator

We reproduce FitzGerald et al. (2018)'s architecture, encoding sentences with a stacked alternating LSTM (Zhou and Xu, 2015) with highway connections (Srivastava et al., 2015) and recurrent dropout (Gal and Ghahramani, 2016), and representing spans by concatenating the output embeddings of their endpoints (Lee et al., 2016). The question generator is a specialized LSTM decoder which only outputs the tokens allowed in each QA-SRL slot. The current predicate is indicated by an embedded binary feature input to BiLSTM encoder, and answer span representations are input at each step of the LSTM decoder. We make two changes from FitzGerald et al. (2018): 1) As opposed to GloVe (Pennington et al., 2014) or ELMo (Peters et al., 2018), We embed the inputs with BERT-base (Devlin et al., 2019) in the 'feature' style with a learned scalar mix over layers, and 2) we additionally concatenate the output embedding of the predicate to the input of the LSTM decoder.

B Hyperparameters

QA-SRL Question Generator The BiLSTM encoder uses a hidden size of 300, 4 layers, 0.1 recurrent dropout probability, and a 100-dimensional predicate indicator embedding. The LSTM decoder has a 100-dimensional hidden state and predicts QA-SRL slots with 200-dimensional embeddings via an MLP with a 100-dimensional hidden layer. We train on all QA pairs in the QA-SRL Bank 2.0 expanded training set using BERT's variant of Adam (Kingma and Ba, 2015) with a learning rate of $5\mathrm{e}{-5}$ and batch size of 32, selecting the model with minimal perplexity on the expanded development set. To produce our feature vectors ϕ, we decode questions with a beam size of 20 and a minimum probability cutoff of 0.01.

Flat Pre-Clustering We perform flat clustering with 100 clusters, skipping this step for verbs with 100 arguments or less. We use a concentration parameter of $\alpha = 0.01$ (i.e., uniform base measure with a sum of 0.01) and do 5 random restarts, each running until the loss decreases by less than $1\mathrm{e}{-5}$ per iteration, and choose the run that yields the lowest loss.

Model	PU	CO	F1	ΔF1
SYNTF	81.6	77.8	79.6	0.0
+ negation	82.8	77.8	80.2	+0.6
+ modals	83.0	79.8	81.3	+1.7
+ discourse	82.6	77.8	80.1	+0.5
+ pass→act	83.6	80.8	82.2	+2.6
+ all rules	87.3	83.1	85.2	+5.6

Table 7: Detailed results for auxiliary rules on SYNTF.

Tuned Splitting Our final model (HUM-QQ + lex) uses $\lambda = 0.35$.

C Rule Lexica

Here we list the full lexica for the auxiliary clustering rules described in Section 4.2.

Negation 5 items: *n't*, *never*, *no*, *no longer*, *not*.

These are drawn directly from the PropBank guidelines (Babko-Malaya, 2005, p. 32).

Modals 23 items: *'d*, *'ll*, *'ve*, *able*, *ca*, *can*, *can't*, *could*, *going*, *gon*, *gonna*, *have*, *may*, *might*, *must*, *ought*, *shall*, *should*, *used*, *will*, *wo*, *won't*, *would*.

Note the inclusion of *have*, *used*, *able*, and *going*, which are parts of phrasal modals (e.g., *have to*), which are included in `AM-MOD` according to the PropBank guidelines (Babko-Malaya, 2005, p. 32).

Discourse 55 items: *after all*, *ah*, *also*, *and*, *and so*, *as a result*, *as we've seen before*, *as well*, *but*, *certainly*, *damn*, *either*, *for example*, *for instance*, *for one*, *for one thing*, *frankly*, *furthermore*, *gosh*, *hence*, *however*, *in addition*, *in any case*, *in any event*, *in contrast*, *in fact*, *in other words*, *in particular*, *in that case*, *in this case*, *in turn*, *indeed*, *instead*, *ironically*, *moreover*, *nonetheless*, *of course*, *oh gosh*, *oh my god*, *oh my gosh*, *on the other hand*, *or*, *particularly*, *rather*, *regardless*, *similarly*, *so*, *specifically*, *thereby*, *therefore*, *though*, *thus*, *too*, *uh*, *um*.

Note the inclusion of some interjections, (*ah*, *oh my gosh*, etc.), which are included in `AM-DIS` according to the PropBank guidelines (Babko-Malaya, 2005, p. 31).

D Auxiliary Rule Performance Breakdown

In Table 7, we provide a more detailed accounting of the improvements that arise from our auxiliary rules described in Section 4.2 and Table 5.

Tuning Method	PU	CO	F1
Constant $k = 6$	83.9	86.7	85.3
$\lambda = 0.35$	85.4	88.8	87.1
F1 Oracle	**87.6**	**89.6**	**88.6**

Table 8: Comparison of methods to determine the number of clusters for each verb. All reported numbers are for HUM-QQ+ lex.

Setting	Objective
$\lambda = 1$	Mixture of Unigrams Likelihood
$\lambda = 0$	Jensen-Shannon Divergence
$\lambda = -1$	Mutual Information

Table 9: Objectives reproduced by the HUM loss for different settings of λ, described in Appendix G.

The negation and discourse rules bring precision improvements, likely because they mostly have `ADV` dependencies outgoing. The modal rule improves both precision and recall because modals have many different kinds of outgoing dependencies, due to their status as heads of clauses (which can serve in many syntactic capacities). Finally, the passive alternation rule aids precision by splitting `SBJ` between active and passive uses, and aids recall by grouping `LGS` with the active `SBJ` and passive `SBJ` with active `OBJ`. This mainly affects the core argument labels `A0` and `A1`, as shown in Table 5 — especially `A1`, as we also find for QA-SRL questions in Section 6.1.

E Tuned Splitting Evaluation

Our model has a single parameter λ which determines the number of clusters for each verb via the tradeoff between the data likelihood and clustering likelihood. We compare this to a constant baseline (the same number of clusters for all verbs) and an oracle upper bound which chooses the split that maximizes the purity/collocation F1 score for each verb independently. As shown in Table 8, we improve on the constant baseline by 1.8 points (85.3→87.1), but fall short of the oracle by 1.5 points (87.1→88.6). There is room for improvement, but errors in the tuning step may not be the most significant factor to concern future work.

F B^3 Results

Results using B^3 metrics on models we tested are shown in Table 10.

Model	B^3P	B^3R	F1	ΔF1
Gold Syntax				
SyntF	74.7	68.3	71.3	0.0
+ lex	79.1	70.4	74.5	+3.2
+ pass→act	77.4	72.1	74.7	+3.4
+ all rules	**82.2**	**74.7**	**78.3**	**+7.0**
Automatic QA-SRL				
HUM-QQ	71.1	79.0	74.8	+3.5
− conn. pen.	71.6	75.7	73.6	+2.3
+ lex	**79.8**	**83.4**	**81.6**	**+10.3**
+ lex + MI	77.7	82.1	79.9	+8.6

Table 10: B^3 Results on models we tested. The gap between HUM-QQ and SyntF is larger than for purity and collocation, as B^3 is a tougher metric which is more discriminative between clusterings. The last model variant (+MI) is described in Appendix G.

G Related Clustering Algorithms

Recall the Hard Unigram Mixture loss

$$\mathcal{L}^{\text{HUM}}_{\lambda}(\mathbf{C}) = -\log \mathrm{P}(\mathbf{X} \mid \mathbf{C}) - \lambda \log \mathrm{P}(\mathbf{C}).$$

Different settings of λ reproduce several objectives present in the literature, summarized in Table 9. As written in Section 3, when $\lambda = 1$, minimizing $\mathcal{L}^{\text{HUM}}_{1}$ maximizes likelihood of the data $\mathbf{X}$ under a mixture of unigrams model (Nigam et al., 2000).

When the number of clusters k is fixed, setting $\lambda = 0$ as in our greedy merging step (Section 3.3) is equivalent to enforcing a uniform prior π over mixture components. In this case, the gain in loss on each merge is the Jensen-Shannon Divergence (JSD) between the merged clusters, scaled by their total size and using each cluster's size to determine its mixing weights in the divergence, as in the mixture-based definition of JSD by Lin (1991). JSD is used in the same way by Chrupała (2012), without the scaling and weighting, as a similarity measure for agglomerative clustering.

Finally, setting $\lambda = -1$ reduces the HUM loss to the mutual information between the QA-SRL questions under ϕ and the cluster assignment C, which has been used in prior work to encourage informative clusterings (Michael et al., 2020). This is related to the *distributional clustering* paradigm of Pereira et al. (1993), which aims to identify common factors that explain distributional data, and which Slonim and Tishby (1999) frame in terms of an information bottleneck that maximizes mutual information between the data and a jointly distributed 'relevance' variable (though in our case, the reference variable is the cluster assignment itself). Setting $\lambda = -1$ in the greedy merging step, we find (in Table 10) that using a mutual information criterion in this way hurts performance. We guess this is because the objective incentivizes clusters of uniform size, which does not match the highly skewed distributions of gold semantic roles.

H Manual Analysis Results

H.1 Improved Verbs on `A0`

The top 50 verbs by F1 gain on `A0` from SyntF to HUM-QQ are: *compete*, *conduct*, *connect*, *combine*, *dominate*, *restore*, *require*, *yield*, *limit*, *ban*, *direct*, *tie*, *oversee*, *contain*, *identify*, *increase*, *evaluate*, *specialize*, *allow*, *assist*, *restrict*, *found*, *grant*, *feature*, *propose*, *detail*, *force*, *convert*, *veto*, *rate*, *bolster*, *appoint*, *enact*, *design*, *list*, *lead*, *resolve*, *retire*, *schedule*, *reach*, *analyze*, *remove*, *speed*, *manage*, *deliver*, *underlie*, *revise*, *emerge*, *enable*, *block*.

We examined 30 sentences containing the top 3 verbs (*compete*, *conduct*, and *connect*). There were 31 `A0` arguments of these verbs in these sentences. Of these, 8 (26%) were overt, 11 (35%) were extracted subjects of relative clauses, 5 (16%) were modified by the predicate appearing in an adjectival clause, 5 (16%) were subjects of open complements of control verbs, and 2 (6%) were otherwise implicit (subject of an adverbial clause or open complement not under a control verb).

H.2 Improved Verbs on `A1`

We examined the top 50 verbs by their difference in B^3 performance on `A1` between SyntF and HUM-QQ. 48 of them are transitive; the other two are bolded. In decreasing order of F1 gain, they are: *propose*, *prefer*, *price*, *relate*, *involve*, *help*, *choose*, *consider*, *design*, *mention*, *identify*, *release*, *include*, ***exist***, ***range***, *value*, *revise*, *lead*, *associate*, *need*, *increase*, *import*, *prove*, *feel*, *place*, *determine*, *limit*, *found*, *enact*, *control*, *cancel*, *dilute*, *disclose*, *select*, *exclude*, *force*, *insure*, *accrue*, *damage*, *calculate*, *hurt*, *secure*, *delay*, *regard*, *record*, *open*, *use*, *concern*, *weaken*, *adjust*.

I Calculating Normalized PMI

Here we describe some special concerns for our use of normalized PMI in Section 6.2.

Pointwise mutual information (PMI) is a measure of how likely two items (such as tokens in a corpus) are to occur together relative to chance (Church and Hanks, 1989). One feature of PMI is that it tends to be larger for rare events: if two items x and y always occur together, then their PMI is $-\log \mathrm{P}(x, y)$. This can make it difficult to assess association patterns among items with greatly varying probabilities (e.g., the `AM-CAU` role appears for 1% of arguments, while `A1` appears for 27%). So we use *normalized* PMI (NPMI; Bouma, 2009), which factors out the effect of item frequency on PMI. Formally, the NPMI of x and y is

$$\left(\log \frac{\mathrm{P}(x, y)}{\mathrm{P}(x)\,\mathrm{P}(y)}\right) \Big/ -\log(\mathrm{P}(x, y)) \;, \qquad (1)$$

taking the limit value of -1 when they never occur together, 1 when they only occur together, and 0 when they occur independently. We use NPMI to analyze the co-occurrence of *gold labels* in *predicted clusters*: A pair of gold labels with high NPMI are preferentially grouped together by the induced roleset, whereas two labels with low NPMI are preferentially distinguished. The joint distribution between gold labels is generated by drawing one point (x) uniformly at random from the data, drawing another (y) uniformly at random from x's predicted cluster, and reading the gold labels of both. NPMI has been used to analyze clusters in this way by Michael et al. (2020).

Calculating NPMI naïvely on our full clustering has a caveat. The denominator of the PMI term in Equation 1, $\mathrm{P}(x)\,\mathrm{P}(y)$, uses marginal probabilities of x and y over the corpus to calculate chance co-occurrence. But our clusters are constrained not to overlap between verbs, so this does not correctly estimate chance cooccurrence in our setting. Instead, we use the expectation over verbs of within-verb chance cooccurrence:

$$\sum_{v} \mathrm{P}(x \mid v)\,\mathrm{P}(y \mid v)\,\mathrm{P}(v),$$

where $\mathrm{P}(v)$ is proportional to the number of arguments for the verb v.

J Question Distributions by Role

We list the top questions and their probabilities for modifier roles in Table 11. Questions for core roles and the ones covered by our lexical rules are in Table 12. We use *verb* (or *verbs*, or *verbed*) as a placeholder for the verb, which in practice is replaced with the predicate for a given instance.

Role	**Top Questions**	**Prob**
`TMP`	When does something verb something?	0.34
	When does something verb?	0.21
	When is something verbed?	0.18
	When does something verb somewhere?	0.03
	When does sth. verb to do something?	0.02
	How does something verb?	0.01
	How is something verbed?	0.01
`ADV`	Why does something verb something?	0.13
	How does something verb something?	0.12
	When does something verb something?	0.09
	How is something verbed?	0.08
	How does something verb?	0.08
	Why does something verb?	0.05
	Why is something verbed?	0.04
	When does something verb?	0.04
	What does something verb?	0.03
	When is something verbed?	0.03
`MNR`	How is something verbed?	0.25
	How does something verb?	0.22
	How does something verb something?	0.19
	What does something verb?	0.02
	Where does something verb?	0.02
	Why does something verb something?	0.02
	How does something verb somewhere?	0.02
`LOC`	Where does something verb something?	0.24
	Where is something verbed?	0.22
	Where does something verb?	0.21
	When does something verb something?	0.04
	How does something verb something?	0.03
	How does something verb?	0.02
	How is something verbed?	0.02
`PNC`	Why does something verb something?	0.29
	Why is something verbed?	0.21
	Why does something verb?	0.08
	Why does something verb somewhere?	0.05
	What is something verbed to do?	0.03
	How is something verbed?	0.03
	What is something verbed for?	0.02
`CAU`	Why does something verb something?	0.32
	Why does something verb?	0.16
	Why is something verbed?	0.16
	Why does something verb somewhere?	0.04
	How does something verb?	0.04
	Why does sth. verb to do something?	0.03
	How does something verb something?	0.02
`DIR`	Where does something verb?	0.40
	How does something verb?	0.17
	Where is something verbed?	0.10
	Where does something verb something?	0.07
	How is something verbed?	0.06
	How does something verb something?	0.03

Table 11: Top questions in the QA-SRL features on the training set for modifier roles. Most of the roles align with a particular *wh*-word especially well, especially for *when*, *where*, and *why*. But `AM-ADV` takes a variety of *wh*-words, and *how* appears often for nearly all modifier roles. In longer questions, 'something' is abbreviated for space.

Role	Top Questions	Prob
`A0`	What verbs something?	.65
	What verbs?	.14
	How is something verbed?	.02
`A1`	What does something verb?	.42
	What is verbed?	.25
	What verbs?	.09
	What verbs something?	.03
	What does something verb to do?	.02
`A2`	What does something verb?	.12
	How is something verbed?	.07
	What verbs something?	.07
	Where is something verbed?	.06
	What is verbed?	.06
	How does something verb?	.06
	How much does something verb?	.04
`A3`	How does something verb?	.15
	What does something verb?	.09
	How is something verbed?	.07
	Why does something verb something?	.06
	How does something verb something?	.05
	When does something verb?	.05
	Where does something verb?	.04
`A4`	What does something verb to?	.17
	Where does something verb?	.17
	How does something verb?	.16
	How much does something verb?	.14
	What does something verb something to?	.04
	How is something verbed?	.03
`NEG`	What verbs something?	.40
	What verbs?	.15
	What is verbed?	.12
	How is something verbed?	.05
	How does something verb?	.03
	How does something verb something?	.03
`MOD`	What verbs something?	.22
	How does something verb something?	.11
	What verbs?	.09
	Why does something verb something?	.07
	What is verbed?	.06
	How does something verb?	.06
	How is something verbed?	.06
`DIS`	When does something verb something?	.15
	How does something verb something?	.15
	What verbs something?	.08
	How is something verbed?	.07
	How does something verb?	.07
	Why does something verb something?	.06
	When does something verb?	.05

Table 12: Top questions in the QA-SRL features on the training set for core roles and the ones covered by our lexical rules. The questions for `AM-NEG`, `AM-MOD`, and `AM-DIS` often don't make sense, e.g., asking for the subject of the verb. No QA-SRL questions are appropriate or were annotated for many arguments of these types. On the other hand, the core roles behave essentially as expected: `A0` is dominated by the subject, `A1` has a mix of subjects and objects, with some complements, and `A2` and on have a wider spread of different expressions. Since the core argument roles have predicate-specific meanings, the distributions here can only be interpreted as aggregates across many such meanings.

Plot and Rework: Modeling Storylines for Visual Storytelling

Chi-Yang Hsu[1,3*], **Yun-Wei Chu**[2*], **Ting-Hao (Kenneth) Huang**[1], **Lun-Wei Ku**[3]
Pennsylvania State University [1], Purdue University [2],
Institute of Information Science, Academia Sinica[3]
`{cxh5438, txh710}@psu.edu`
`{chu198}@purdue.edu`
`{lwku}@iis.sinica.edu.tw`

Abstract

Writing a coherent and engaging story is not easy. Creative writers use their knowledge and worldview to put disjointed elements together to form a coherent storyline, and work and rework iteratively toward perfection. Automated visual storytelling (VIST) models, however, make poor use of external knowledge and iterative generation when attempting to create stories. This paper introduces ***PR-VIST***, a framework that represents the input image sequence as a story graph in which it finds the best path to form a storyline. PR-VIST then takes this path and learns to generate the final story via a re-evaluating training process. This framework produces stories that are superior in terms of diversity, coherence, and humanness, per both automatic and human evaluations. An ablation study shows that both plotting and reworking contribute to the model's superiority.

1 Introduction

Writing a story is a complicated task. Human writers use their knowledge to tie all the disjointed elements, such as people, items, actions, or locations, together to form a coherent storyline. Writers also re-evaluate their work constantly during the writing process, and sometimes even alter their writing goals in the middle of a draft. Flower and Hayes (1981) characterize a solo writer's cognitive process as a series of components in which the writer's own knowledge is described as the long-term memory, and the planning, translating, and reviewing steps can occur in a recursive, interconnected manner. These creative steps are essential to human writing. However, automated visual storytelling (VIST) models that compose stories given five images (Huang et al., 2016) do not make extensive use of human knowledge to tie the elements together, nor do they use human insight to evaluate the outputs and guide the generation process.

As for linking elements, most works generate visual stories in an end-to-end fashion (Huang et al., 2016; Kim et al., 2018), treating the task as a straightforward extension of image captioning. Recent works have begun to use relations between entities to improve visual storytelling, but often narrow in a particular subset of relations, such as relations between elements within the same image (Yang et al., 2019), relations between two adjacent images (Hsu et al., 2020), or relations between scenes (Wang et al., 2020). The full potential of rich real-world knowledge and intra-image relations have yet to be fully utilized. As for re-evaluation, recent work uses reward systems (Wang et al., 2018b; Hu et al., 2019) or estimated topic coherence (Wang et al., 2019) to automatically assess the output story and guide the generation process. However, these approaches are often optimized towards predefined aspects such as image relevancy or topic coherence, which do not necessarily lead to engaging stories from a human perspective. In the cognitive process of human writing, the writer's judgment is critical, and visual storytelling models could benefit by considering human ratings.

This paper introduces ***PR-VIST***, a novel visual storytelling framework that constructs a graph and captures the relations between all the elements in the input image sequence, finds the optimal path in the graph that forms the best storyline, and uses this path to generate the story. An overview of PR-VIST is shown in Figure 1.

- **Stage 1 (Story Plotting):** PR-VIST first constructs a story graph for the image sequence by extracting various elements (i.e., term nodes, object nodes) from all the images and linking these elements using external knowledge (i.e., VIST graph, VG graph). PR-VIST then finds

---

[*] denotes equal contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4443–4453
August 1–6, 2021.

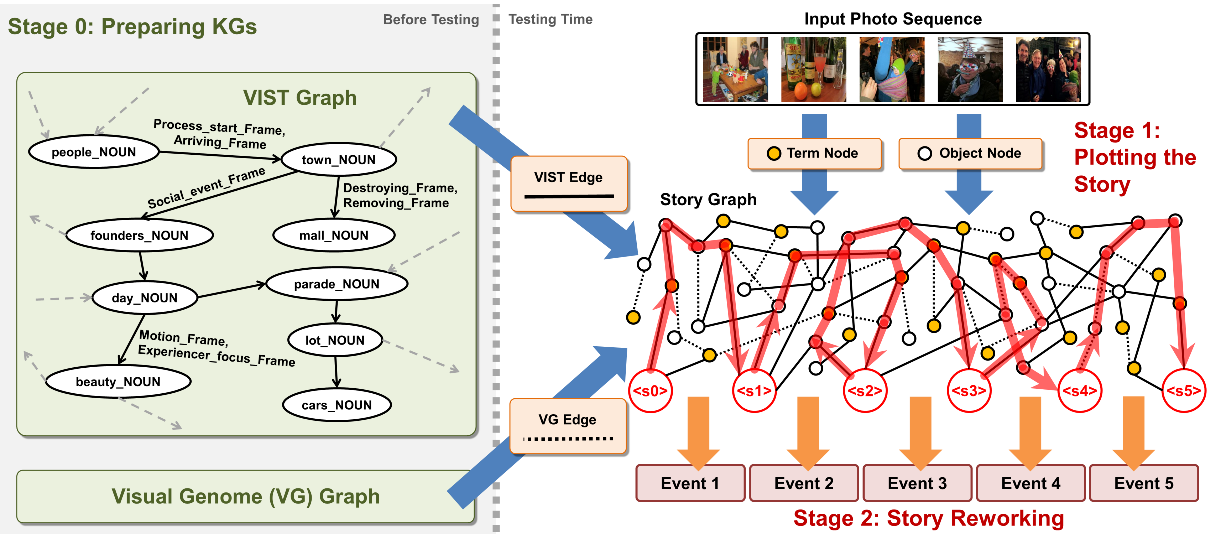

Figure 1: Overview of PR-VIST. In **Stage 1 (Story Plotting)**, PR-VIST first constructs a graph that captures the relations between all the elements in the input image sequence and finds the optimal path in the graph that forms the best storyline. In **Stage 2 (Story Reworking)**, PR-VIST uses the found path to generate the story. PR-VIST uses a story generator and a story evaluator to realize the "rework" process. In **Stage 0 (Preparation)**, a set of knowledge graphs that encode relations between elements should be prepared for the uses in Stage 1.

the best path in the story graph as the storyline and passes it to Stage 2.

- **Stage 2 (Story Reworking):** PR-VIST uses a story generator and a story evaluator to realize the reworking process: the generator takes the storyline produced in Stage 1 as the input to generate the story and backpropagates with an evaluator-augmented loss function. The evaluator, a discriminator model trained on human rating score data to classify good and bad stories, outputs a story quality score and modifies the loss. After a few optimization epochs, the generator eventually learns to generate stories that reflect human preferences.

In **Stage 0 (Preparation)**, a set of knowledge graphs that encode relations between elements are prepared for use in Stage 1. In this work, we prepare two knowledge graphs: a VIST graph and a visual genome (VG) graph. We construct the VIST graph based on the VIST dataset, representing in-domain knowledge; the VG graph is an existing resource (Krishna et al., 2017), representing generic knowledge. Note that as the PR-VIST framework is generic, it can use any knowledge graphs as needed.

Automatic and human evaluations show that PR-VIST produces visual stories that are more diverse, coherent, and human-like. We also conduct an ablation study to show that both story plotting (Stage 1) and reworking (Stage 2) contribute positively to the model's superiority. We believe this work also shows the potential of drawing inspiration from human cognitive processes and behavior to improve text generation technology.

2 Related Work

Visual Storytelling Researchers have been trying to advance the visual storytelling task since it was introduced by Huang et al. (2016). Some work modifies end-to-end recurrent models for better story generation (Hsu et al., 2018; Gonzalez-Rico and Fuentes-Pineda, 2018; Kim et al., 2018; Huang et al., 2019; Jung et al., 2020), and some use adversarial training to generate more diverse stories (Chen et al., 2017; Wang et al., 2018a,b; Hu et al., 2019). These methods produce legitimate stories and easier to implement because they relies only on one dataset. However, the generated stories can sometimes be monotonous and repetitive.

Leveraging External Resources for VIST Another set of work leverages external resources and knowledge to enrich the generated visual stories. For example, Yang et al. (2019) apply ConceptNet (Liu and Singh, 2004) and self-attention for create commonsense-augmented image features; Wang et al. (2020) use graph convolution networks on scene graphs (Johnson et al., 2018) to associate objects across images; and KG-Story (Hsu et al., 2020) is a three-stage VIST framework that uses

Visual Genome (Krishna et al., 2017) to produce knowledge-enriched visual stories.

Editing or Optimizing Visual Stories A few prior work tries to post-edit visual stories or optimize the story content toward specific goals. VIST-Edit is an automatic post-editing model that learns from an pre- and post-edited parallel corpus to edit machine-generated visual stories (Hsu et al., 2019). While VIST-Edit is useful, it requires parallel training data, which is often unavailable. Hu et al. (2019) use a reward function to optimize the generated stories toward three aspects; Li et al. (2019) customize the emotions of visual stories. These methods use automatic metrics to optimize visual stories toward specific goals; our work, on the other hand, leverages the human evaluation data to guide the generation process.

Story Plotting in Story Generation Research in automatic story generation has demonstrated the effectiveness of story plotting (Yao et al., 2018; Fan et al., 2019), which typically involves organizing the "ingredients" into a well-organized sequence of events. Nevertheless, none of the studies applied story plotting for visual stories.

3 Stage 0: Preparation

To prepare for story plotting, we collect information from the images and knowledge from the knowledge graphs.

3.1 Story Element Extraction

To extract information from the images, two extraction methods are used to extract image-oriented and story-oriented story elements: *objects* and *terms*, respectively representing image and story intuition.

Objects These can be detected by current object detection models, for which we use a pre-trained object detection model—Faster-RCNN (Ren et al., 2015). To ensure the detected objects' reliability, only those objects with the top five confidence scores are used in each image.

Terms These are story-like nouns such as events, time, and locations, which current object detection models are unable to extract. Therefore, we further use a Transformer-GRU (Hsu et al., 2020) to predict story-like terms. For each image and story pair, we use image objects as the input and the nouns in the corresponding human-written story as the ground truth. The Transformer-GRU learns to convert objects to nouns commonly used in stories.

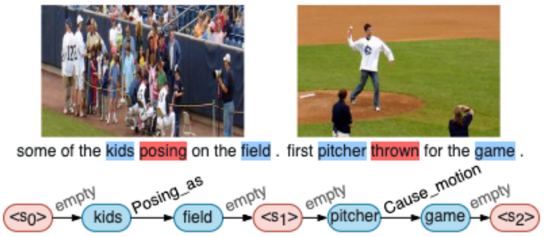

Figure 2: From the stories in VIST training data, a story is transformed into a golden storyline following the human reading direction, as part of the VIST graph.

3.2 Knowledge Graph Preparation

To collect interactive relations between nouns, we prepare Visual Genome graph $\mathcal{G}_{vg}$ and VIST graph $\mathcal{G}_{vist}$. These graphs contain interlinked real-world objects and terms, displaying visual and storytelling interaction. Table 1 summarizes the statistic of each graph.

Visual Genome Graph $\mathcal{G}_{vg}$ describes pairwise relationships between objects in an image, describing visual interactions. No prepositional relations are included; only verb relations are preserved. All relations are converted into semantic verb frames using Open-SESAME (Swayamdipta et al., 2017), in which the semantic frames were pre-defined in FrameNet (Baker et al., 1998).

VIST Graph we propose $\mathcal{G}_{vist}$ to collect the storytelling interactions. We develop this novel story knowledge graph by converting references in the VIST training and validation datasets (Huang et al., 2016) to graphical data. Following the reading direction, in each reference, we extract nouns and semantic verb frames using SpaCy[1] and Open-SESAME to obtain noun–verb–noun (NVN) tuples. Using nouns and semantic verb frames as nodes and edges, these are collectively assembled into a golden storyline. For example, for "first pitcher thrown for the game" in Figure 2, we extract *pitcher*, *game*, and *Cause_motion*, which is a semantic verb frame for *thrown*, as a NVN tuple. Additionally, we include a noun token $<$`s`$_i>$ as the transition point to the next sentence or termination point of a story, and a verb frame token `empty_frame` to interlink two nouns when a semantic frame is absent. To conclude, all of the golden storylines are assembled into $\mathcal{G}_{vist}$.

---

[1] SpaCy: https://spacy.io/

	Nodes	Relations	Links
$\mathcal{G}_{vg}$	3,323	564	22.31
$\mathcal{G}_{vist}$	2,048	531	11.75
$\mathcal{G}_{vg+vist}$	4,158	880	22.78

Table 1: The statistics of knowledge graphs. The table shows the number of distinct nodes and relations in each graph. It also shows the average link per node. Note that the nodes and relations from $\mathcal{G}_{vg}$ and $\mathcal{G}_{vist}$ have overlaps.

4 Stage 1: Story Plotting

4.1 Storyline Predictor

In Stage 1, PR-VIST uses a storyline predictor to find what it deems the best path in the story graph as the storyline and then pass this to Stage 2. For the storyline predictor, we use UHop (Chen et al., 2019), a non-exhaustive relation extraction framework. A single hop is defined as searching from one entity to another entity by a single relation. UHop performs multiple single-hop classifications consecutively in the graph to find the path representing the storyline, that is, a path that consists of a sequence of nouns and verb frames.

Single-hop classification can described as Equation 1 and Figure 3. In step i, at the current head entity h_i, the model is given a list of candidate relations $r_i \in R_i$ and tail entities $t_i \in T_i$. Each r_i is in [verb.t_i] or [verb.noun] format, containing information for both the verb frame and the tail noun entity. The scoring model F is given *objects* and predicted relations $r_1, ..., r_{i-1}$ as input. The model predicts a score for each r_i and selects the highest verb-noun pair r_i from Q:

$$r_i = \arg\max_{q \in Q} F(objects, r_1, ..., r_{i-1}). \quad (1)$$

Training UHop learns to find a path for the storyline from the golden storyline. The training procedure starts with an initial noun token entity <`s0`> in the golden storyline for single-hop classification, where $h_1 =$ <`s0`>. It learns to select the correct relation r_i from a list of candidate relations R_i in $\mathcal{G}_{vg}$ and $\mathcal{G}_{vist}$. Then, it calculates the error to the noun and verb frame in the golden storyline for backpropagation. In the next hop, the framework proceeds to the next noun in the golden storyline and repeats the single-hop classification.

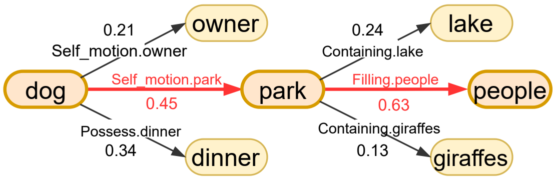

Figure 3: Storyline pathfinding process. All entities are from *object* or *term* list, and all relations are in [verb.noun] format, which the verbs are verb frames from knowledge graphs and the nouns are the tail entities. The single-hop classification begins with `dog`. The storyline predictor is given three candidate relations. The framework selects the highest score relation and move on to the next entity `park`. Then, the single-hop classification repeats.

Testing In PR-VIST's testing step, for each story, five images are transformed into a story graph $\mathcal{G}_{story}$. As demonstrated in Figure 1, we first extract the *object* and *term* story elements for each story, and then link these together using the verb frames in $\mathcal{G}_{vist}$ and $\mathcal{G}_{vg}$ as edges. This yields a well-defined graph presenting a comprehensive view of five images for each story—$\mathcal{G}_{story}$. Next, a trained UHop finds a storyline in $\mathcal{G}_{story}$, where all entities are only the *objects* and *terms* from the given five images. The framework starts with <`s0`> to perform single-hop classification, where $h_1 =$ <`s0`>. Unlike training, it only selects r_i from R_i listed in $\mathcal{G}_{story}$. In the next hop, the previous predicted entities are used as the start entity: $h_0 = t_{i-1}$. It then continues to hop from entity to entity until it reaches the next token <`s1`>. The path from <`si-1`> to <`si`> is called an event e_i. The path search from <`si`> to the next token <`si+1`> continues until the search is terminated by the termination decision described in UHop. Eventually, the model finds a storyline of arbitrary length L, that is, a storyline that contains any number of events: $e_1, ..., e_L$.

4.2 Implementation Details

HR-BiLSTM (Yu et al., 2018) is adopted as the scoring model F, in which *objects* are converted to word embeddings via GloVe (Pennington et al., 2014) as $E(object)$. All relation embeddings $E(r)$ are decomposed into graphical embedding E_{graph} and textual embedding E_{text}. E_{graph} transforms a verb frame v and a tail entity's image position p^t into a one-hot vector, denoting the graphical and image positional information. E_{text} is composed of the verb frame and tail entity word embedding. Then, E_{graph} and E_{text} are concatenated into a unified representation $E(r)$. We formulate the representation of relation r as

$$\begin{aligned} E_{graph}(r) &= [\mathbb{1}(v); \mathbb{1}(p^t)], \\ E_{text}(r) &= [E_w(f); E_w(t)], \\ E(r) &= [E_{graph}(r); E_{text}(r)], \end{aligned} \tag{2}$$

where $\mathbb{1}(\cdot)$ returns the one-hot representation, $E_w(\cdot)$ returns the word embeddings via GloVE, and [;] denotes concatenation.

A verb frame and tail entity are combined into r_i due to relational ambiguity issues among candidate relations. Using Figure 3 as an example, given a head entity *dog*, candidates *self_motion.park* and *self_motion.owner* represent different semantic meanings when tail entities *park* and *owner* are included. However, excluding tail entities results in identical relation candidates *self_motion* and thus ambiguity between two different candidates.

5 Stage 2: Story Reworking

In story reworking, the framework consists of two components: the story generator and the story evaluator. The story generator generates a story according to the storyline, and the story evaluator—a discriminator trained on the MTurk human ranking data to classify good and bad stories—outputs a story quality score and modifies the loss functions.

5.1 Story Generator

A storyline consists of a set of events $e_1 ... e_L$ that are input to the story generator, which is based on the Transformer (Vaswani et al., 2017). Unlike most VIST models, the story generator is dynamic: the number of output sentences depends on the number of events. To manage a diverse number of events, the Transformer is designed as a sentence generator that iteratively generates one sentence per event until it generates L sentences. For each step i, event e_i and the previous predicted sentence y_{i-1} are used to predict the next sentence y_i. After L steps, the story generator outputs a story $s = y_1, ... y_L$.

5.2 Story Evaluator

Most VIST works use human evaluations to examine their work's quality via crowdsourcing, comparing their generated stories with the baseline stories. In this paper, we use the first- and last-place stories in the MTurk human ranking data as positive and negative samples. The story evaluator, a discriminator trained on the MTurk human ranking data, learns to distinguish positive and negative samples. It outputs a score for each story, and the scores are converted into rewards, as shown below:

$$p_{LM}(u|s) = \text{softmax}(\tanh(\mathbf{W}LM(s)) + \mathbf{b}), \tag{3}$$

$$\mathcal{R}(s) = -p_{LM}(u|s) + c, \tag{4}$$

where $LM(\cdot)$ is a GRU language model (Cho et al., 2014), $u = 1$ indicates story s is a positive sample, and $u = 0$ indicates s is a negative sample. Language model $p_{LM}(\cdot)$ returns a score from 0 to 1 to reflect story quality. The story evaluator $\mathcal{R}(\cdot)$ returns a reward, an inverse of $p_{LM}(\cdot)$ with coefficient $c = 1.5$. The reward later manipulates the loss, optimizing toward human preference. Note that the story evaluator is pre-trained.

5.3 Optimization with Story Evaluator

For optimization, the story generator uses sentence-level and story-level loss functions. Given reference $y_1^*, ..., y_L^*$ and predicted story $y_1, ..., y_L$, in the maximum likelihood estimation (MLE) optimization process, in each step from 1 to L, the model predicts a sentence y_i to calculate the loss between y_i and y_i^* and then backpropagates, as shown in Figure 4. After predicting L sentences, in story-level optimization, the model predicts $y_1, ..., y_L$ to calculate the negative log-likelihood to the reference $y_1^*, ..., y_L^*$ and then backpropagates. The sentence-level and story-level optimization by MLE on dataset $\mathcal{D}$ are formulated as

$$J_{sen}^{MLE}(\theta, \mathcal{D}) = \sum_{Y \in \mathcal{D}} -\log p_\theta(y_i | e_i, y_{i-1}), \tag{5}$$

$$J_{story}^{MLE}(\theta, \mathcal{D}) = \sum_{Y \in \mathcal{D}} \frac{1}{L} \sum_{i=1}^{L} -\log p_\theta(y_i' | e_i, y_{i-1}'), \tag{6}$$

where e_i and y_i denote the i-th event and the sentence respectively, y_i' represents the updated sentence after sentence-level optimization, and θ represents the story generation model parameters, which are updated using Adam (Kingma and Ba, 2015).

After training for 30 epochs [2], the story evaluater begins to manipulate the story-level loss. Inspired by reinforcement learning (Williams, 1992), which utilizes rewards to guide the training process, we use the story evaluator $\mathcal{R}(\cdot)$ [3] to encourage the

---

[2] The generation model converges at around 20 epochs in our experiment, and we give it extra 10 epochs for precautions.

[3] The pre-trained LM's weights are frozen to stabilize the training.

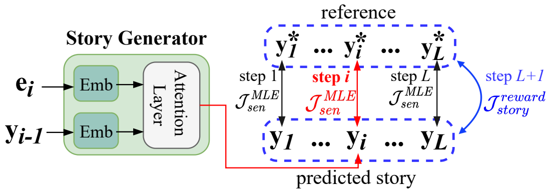

Figure 4: Optimization flowchart for story generator. For steps 1 to L, the model is optimized using sentence-level loss. In step $L+1$, all sentences are generated, and the model is optimized using story-level loss.

generation model to focus on stories preferred by humans. The reward directly multiplies the story-level loss as

$$J^{reward}_{story}(\theta, \mathcal{D}) = \begin{cases} J^{MLE}_{story} & \text{if epoch} \le 30 \\ \mathcal{R}(s) J^{MLE}_{story} & \text{if } 30 < \text{epoch} \le 60 \end{cases}. \quad (7)$$

6 Experimental Results

6.1 Data Setups

We used four datasets in this paper: the VIST dataset, Visual Genome, ROCStories, and MTurk human ranking data. The VIST dataset and Visual Genome are used to construct the knowledge graphs, and ROCStories (Mostafazadeh et al., 2016) is a large quantity of pure textual stories used for pre-training the story generator. The VIST dataset is also used in story plotting to train the storyline predictor and in story reworking to fine-tune the story generator. Notably, we also collected MTurk human ranking data to train the story evaluator. We used the ranking results from KG-Story[4] (Hsu et al., 2020). This data contains two experiments, each of which ontains 500 distinct photo sequences. A photo sequence contains a set of machine-generated stories ranked by 5 MTurk workers. Thus we have 5000 rankings from MTurk workers. Specifically, MTurk workers were asked to rank AREL (Wang et al., 2018a), KG-Story, two KG-Story ablation models, and reference stories, using three different model settings in each experiment. We selected the rank-1 and rank-5 stories as positive and negative samples.

6.2 Baselines

We used several competitive baseline models. **AREL** (Wang et al., 2018a) and **GLAC** (Kim et al., 2018) are end-to-end models with reinforcement learning and global-location attention mechanisms that achieved top ranks in the VIST Challenge (Mitchell et al., 2018). **KG-Story** (Hsu et al., 2020), the current state-of-the-art framework, utilizes a knowledge graph to enrich story contents and generates stories using Transformer.

6.3 Evaluation Methods

Per the literature (Wang et al., 2018a), human evaluation is the most reliable way to evaluate the quality of visual stories; automatic metrics often do not align faithfully to human judgment (Hsu et al., 2019). Therefore, in this paper, we prioritize human evaluation over automatic evaluations.

Human Evaluation: Ranking Stories and Filling a Questionnaire We recruited crowd workers from Amazon Mechanical Turk (MTurk) to assess the quality of the generated stories. For each experiment, we randomly selected 250 stories, each of which was evaluated by five different workers in the US. The experiment includes a comparison study with three baseline models and three ablation studies, and each annotator was compensated \$0.10 for each experiment. Workers were asked to rank the stories (*e.g.*, ours and those of the baseline/ablation models) based on their overall quality.

In addition, the user interface also provides a questionnaire to collect in-depth feedback from MTurk workers. The questions include "What do you like about the best stories" and "What do you dislike about the worst stories" for workers to select aspects that affect overall story quality. These aspects are provided by Huang et al. (2016): they include focus, coherence, shareability, humanness, grounding, and detail. We calculated the average rank and the majority rank among five workers for each story, as well as total votes for each model's best and worst aspects.

Non-Classic Automatic Evaluation: BLEURT, voc-d, and MLTD Many VIST studies have shown that classic automatic evaluation scores like BLEU and METEOR correlate poorly with human judgment (Hsu et al., 2020; Hu et al., 2019; Wang et al., 2020; Li et al., 2020; Yang et al., 2019; Hsu et al., 2019; Wang et al., 2018a; Modi and Parde, 2019). These n-gram matching metrics fail to account for the semantic similarity to the reference stories and lexical richness in the generated stories.

Therefore, we adopted BLEURT (Sellam et al., 2020), a state-of-the-art BERT-based evaluation

---

[4] Data obtained from the authors of KG-Story.

AREL : avg rank #3.6
the parade started with a lot of people in the parade. there were a lot of people there. there was a lot of people there. there were a lot of people there. there were a lot of cars in the parade.

GLAC : avg rank #2.4
the marathon was about to begin. there were many people there. it was a great day. everyone was very excited. they were all very fast.

KG-Story : avg rank #2.2
the parade started with a beautiful day. many people showed up. there were runners. everyone was watching the parade. one woman stood in a car to get everyone involved. it was a nice car.

PR-VIST : avg rank #1.8
we went to see the parade. there was a band guard. many people showed up to watch the parade runners. the runners started to gather in line. there were many cars at the show. the family decided to walk around town.

Figure 5: Generated stories for PR-VIST and baseline models. MTurk workers were asked to rank the stories.

metric, to further correlate generated stories and reference stories based on their semantic meaning. We also adopted lexical diversity metrics voc-d and MLTD (McCarthy and Jarvis, 2010) to quantify story lexical richness. Several works have shown that lexical diversity is positive correlated to story quality (Liu et al., 2019; Dai et al., 2017).

6.4 Results

In our experiments, the stories generated by PR-VIST have an average of 5.96 sentences. 57.3% of these stories contain at least one event (sentence) that uses story elements extracted from two (or more) images, showing PR-VIST's ability to utilize intra-image entities.

Human Evaluation We asked MTurk workers to rank four stories: those of PR-VIST, the three baseline models, and the state-of-the-art KG-story. Table 2 shows the results. PR-VIST outperforms other models in average ranking: it outranks AREL by 0.24 and KG-Story by 0.16. As for the percentage of 1st-rank stories, PR-VIST produces 12.0% more than AREL and 7.5% more than KG-Story. Figure 5 shows a representative example. Compared with end-to-end models (*i.e.*, AREL and GLAC), graph-based methods (*i.e.*, KG-Story and PR-VIST) generate more diverse stories. Compared with KG-Story, whose sentences are relatively simple and plain, generating sentences such as "Many people showed up", our model reuses entities such as "parade" in the first sentence and associates relations with other entities, *e.g.*, "people" and "runners", to compose "many people showed up to watch the parade runners".

Moreover, Figure 6 shows the questionnaire (see Section 6.3) result for the best-ranked stories. For PR-VIST and KG-Story's best-ranked stories, the PR-VIST story count is significantly higher in all aspects; specifically, coherence, shareability, and humanness are higher than other categories.

Method	1st	2nd	3rd	4th	Avg	Major
AREL	20.6% (258)	26.8%	27.2%	25.4%	2.57	2.56
GLAC	21.7% (271)	24.2%	25.5%	28.6%	2.61	2.73
KG-Story	25.1% (314)	25.2%	25.7%	24.0%	2.49	2.53
PR-VIST	**32.6% (407)**	23.7%	21.7%	**22.0%**	**2.33**	**2.28**

Table 2: Human rankings between PR-VIST and three methods. The first four columns indicate the percentage of worker rankings for each method, and the fifth and the last column denote the average and majority ranks (1 to 4, lower is better). PR-VIST outperforms other models in average ranks ($p<0.05$, $N=250$), majority ranks, and also the percentage of 1st-rank stories.

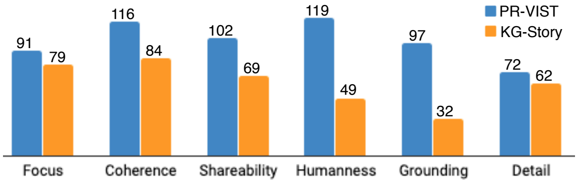

Figure 6: Aspect-wise votes for PR-VIST and KG-Story's first-place stories collected via the questionnaire (see Section 6.3). PR-VIST outperforms drastically in coherence, humanness, and grounding.

Automatic Evaluations Table 3 shows that the proposed method outperforms all the baselines in BLEURT,voc-d, and MLTD. Although n-gram-based automatic metrics are known to correlate poorly with human judgment in VIST (see Section 6.3), it is still noteworthy that PR-VIST results in significantly lower BLEU-4 scores. This might be cause by the fact that PR-VIST uses knowledge to enrich the story content and increase lexical diversity, but could lower the performance in n-gram matching.

Method	BLEU-4	METEOR	BLEURT	MLTD	voc-d
AREL	**14.4**	**35.4**	0.52	22.45	0.53
GLAC	10.7	33.7	0.71	32.87	0.67
KG-Story	9.93	32.2	0.72	40.52	0.71
PR-VIST	7.65	31.6	**1.37**	**45.79**	**0.73**

Table 3: The first two columns show the results of classic n-gram based metrics. The third column shows BLEURT, a BERT-based metric. The last two columns show the lexical diversity evaluation results (MLTD and voc-d). High lexical diversity corresponds to low scores for n-gram metrics.

7 Ablation Study

Three factors contribute to PR-VIST's superior performance: story elements, knowledge graphs, and plot reworking. To evaluate the effectiveness of each factor in our framework, we conducted three ablation studies using human evaluations. The evaluation results are shown in Table 4. All three experiments use the same qualitative analysis, and each experiment ranks PR-VIST and two settings with certain components removed.

Story Elements PR-VIST is compared to two models, each of which uses only *objects* or *terms* for the storyline predictor to plot storylines.

Knowledge Graphs PR-VIST is compared to two models, each of which uses only $\mathcal{G}_{vist}$ or $\mathcal{G}_{vg}$ for the storyline predictor to plot storylines.

Plot and Rework PR-VIST is compared to two models: one without reworking and one without plotting or reworking. Without-reworking means the discriminator is removed, that is, the story generator uses Equation 6 for all epochs. Without-plotting-reworking means that the storyline predictor is additionally removed, so no frames are included; *terms* are used directly as the story generator's input.

Table 4 shows that PR-VIST outperforms all the ablation models. Furthermore, the first and second experiments show that MTurkers prefer story-like storylines to image-like storylines. That is, *terms* and $\mathcal{G}_{vist}$ are better than *objects* and $\mathcal{G}_{vg}$. For the third experiment, we note a steady improvement from without-plotting-reworking to PR-VIST, showing the effectiveness of the proposed method. An example is shown in Figure 7. The model cannot manage the abundant story elements without the guidance of story plotting. Comparing PR-VIST with PR-VIST w/o R, we see that reworking revises and enlivens (e.g., "[organization] in [location]") the stories.

	objects	*terms*	$\mathcal{G}_{vg}$	$\mathcal{G}_{vist}$	Plot	Rework	Avg	Major
1	✓	✓	✓	✓	✓	✓	1.89	1.87
		✓	✓	✓	✓	✓	1.98	2.00
	✓		✓	✓	✓	✓	2.12	2.13
2	✓	✓	✓	✓	✓	✓	1.97	1.94
	✓	✓		✓	✓	✓	1.98	1.99
	✓	✓	✓		✓	✓	2.00	2.06
3	✓	✓	✓	✓	✓	✓	1.95	1.93
	✓	✓	✓	✓	✓		2.00	2.02
	✓	✓	✓	✓			2.03	2.08

Table 4: Human evaluation results for ablation studies: 1. Story elements 2. Knowledge graph 3. Plot and Rework. PR-VIST outperforms in all ablation settings, indicating the importance of using all components.

PR-VIST: we visited the [organization] in [location]. everyone gathered in the [organization] room to celebrate.

PR-VIST w/o R: i visited [organization] [location]. everyone gathered in the [organization] room.

PR-VIST w/o PR: today was the first place i saw a man sign for our trip to [location] [organization]. everyone was in his room at [organization] house.

Figure 7: Snippet of stories generated by the proposed method and two configurations: without reworking (R) and without plotting or reworking (PR). Nouns and verb frames are denoted in blue and red. Reworked parts are marked with wavy underlines.

8 Discussion

To understand areas for improvement, in the human ranking evaluation, we asked crowd workers to select the aspect (out of six) they disliked about the worst story (see Section 6.3.) Of the negative votes, 24.6% were for "grounding." Namely, lower-ranked stories are often not visually grounded. We examined the outputs and found that Faster-RCNN in Stage 1 sometimes predicts objects that are inaccurate but semantically related to the context. Figure 8 shows a typical example, where the soccer ball is identified as a frisbee, which is incorrect but still fits the "sports" theme. When the storyline predictor is unable to distinguish such mistaken objects from appropriate objects, grounding errors occur. A better object detector would mitigate this problem, or we could jointly optimize plotting and generation, for instance by including reworking within storyline plotting.

9 Conclusion

We propose a novel story plotting and reworking framework to mimic the human story-writing process. To the best of our knowledge, no study has integrated knowledge graph and story plotting to utilize visual elements in VIST. Also novel is our approximation of human-preferred stories by reusing and aggregating story generation using the results of human-annotated story ranking evaluations, e.g., human evaluation results from MTurk. We also propose a novel questionnaire embedded in the comparative study to collect detailed, meaningful human-annotated data from MTurk. Experiments attest PR-VIST's strong performance in diversity, coherence, and humanness.

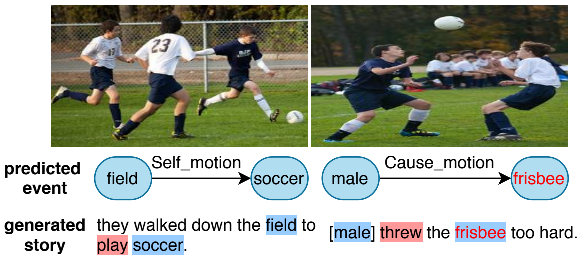

Figure 8: PR-VIST grounding error

10 Ethical Considerations

Although our research aims to produce short stories that are vivid, engaging, and innocent, we are aware of the possibilities of utilizing a similar approach to generate inappropriate text (e.g., violent, racial, or gender-insensitive stories). The proposed visual storytelling technology enables people to generate stories rapidly based on photo sequences at scale, which could also be used with malicious intent, for example, to concoct fake stories using real images. Finally, as the proposed methods use external knowledge graphs, they reflect the issues, risks, and biases of such information sources. Mitigating these potential risks will require continued research.

11 Acknowledgements

This research is supported by Ministry of Science and Technology, Taiwan under the project contract 108-2221-E-001-012-MY3 and 108-2923-E-001-001-MY2 and the Seed Grant from the College of Information Sciences and Technology (IST), Pennsylvania State University. We also thank the crowd workers for participating in this project.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Zhiqian Chen, Xuchao Zhang, Arnold P Boedihardjo, Jing Dai, and Chang-Tien Lu. 2017. Multimodal storytelling via generative adversarial imitation learning. *arXiv preprint arXiv:1712.01455*.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. UHop: An unrestricted-hop relation extraction framework for knowledge-based question answering.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional GAN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979.

Angela Fan, M. Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *ACL*.

Linda Flower and John R Hayes. 1981. A cognitive process theory of writing. *College Composition and Communication*, 32(4):365–387.

Diana Gonzalez-Rico and Gibran Fuentes-Pineda. 2018. Contextualize, show and tell: A neural visual storyteller. *arXiv preprint arXiv:1806.00738*.

Chao-Chun Hsu, Szu-Min Chen, Ming-Hsun Hsieh, and Lun-Wei Ku. 2018. Using inter-sentence diverse beam search to reduce redundancy in visual storytelling. *arXiv preprint arXiv:1805.11867*.

Chao-Chun Hsu, Zi-Yuan Chen, Chi-Yang Hsu, Chih-Chia Li, Tzu-Yuan Lin, Ting-Hao (Kenneth) Huang, and Lun-Wei Ku. 2020. Knowledge-enriched visual storytelling. In *Proceedings of Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Ting-Yao Hsu, Huang Chieh-Yang, Yen-Chia Hsu, and Ting-Hao Kenneth Huang. 2019. Visual story post-editing. In *ACL*.

Junjie Hu, Yu Cheng, Zhe Gan, Jingjing Liu, Jianfeng Gao, and Graham Neubig. 2019. What makes a good story? designing composite rewards for visual storytelling.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8465–8472.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

J. Johnson, Agrim Gupta, and Li Fei-Fei. 2018. Image generation from scene graphs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1219–1228.

Yunjae Jung, Dahun Kim, Sanghyun Woo, Kyungsu Kim, Sungjin Kim, and In So Kweon. 2020. Hide-and-tell: Learning to bridge photo streams for visual storytelling. *ArXiv*, abs/2002.00774.

Taehyeong Kim, Min-Oh Heo, Seonil Son, Kyoung-Wha Park, and Byoung-Tak Zhang. 2018. GLAC Net: GLocal Attention Cascading Networks for multi-image cued story generation. *arXiv preprint arXiv:1805.10973*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, and et al. 2017. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Jiacheng Li, Siliang Tang, Juncheng Li, Jun Xiao, Fei Wu, Shiliang Pu, and Yueting Zhuang. 2020. Topic adaptation and prototype encoding for few-shot visual storytelling. *Proceedings of the 28th ACM International Conference on Multimedia*.

Nanxing Li, Bei Liu, Zhizhong Han, Yu-Shen Liu, and Jianlong Fu. 2019. Emotion reinforced visual storytelling. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, ICMR '19, page 297–305, New York, NY, USA. Association for Computing Machinery.

Hugo Liu and Push Singh. 2004. ConceptNet—a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

Lixin Liu, Jiajun Tang, Xiaojun Wan, and Zongming Guo. 2019. Generating diverse and descriptive image captions using visual paraphrases. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

McCarthy and S. Jarvis. 2010. MTLD, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, page 381–392.

Margaret Mitchell, Ting-Hao Huang, Francis Ferraro, and Ishan Misra. 2018. Proceedings of the first workshop on storytelling. In *Proceedings of the First Workshop on Storytelling*.

Yatri Modi and Natalie Parde. 2019. The steep road to happily ever after: An analysis of current visual storytelling models. In *Proceedings of the Second Workshop on Shortcomings in Vision and Language*, pages 47–57, Minneapolis, Minnesota. Association for Computational Linguistics.

N. Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, P. Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: Learning robust metrics for text generation.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. *arXiv preprint arXiv:1706.09528*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Jing Wang, Jianlong Fu, Jinhui Tang, Zechao Li, and Tao Mei. 2018a. Show, reward and tell: Automatic generation of narrative paragraph from photo stream by adversarial training. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ruize Wang, Zhongyu Wei, Piji Li, Haijun Shan, Ji Zhang, Qi Zhang, and Xuanjing Huang. 2019. Keep it consistent: Topic-aware storytelling from an image stream via iterative multi-agent communication.

Ruize Wang, Zhongyu Wei, Piji Li, Qi Zhang, and Xuanjing Huang. 2020. Storytelling from an image stream using scene graphs. In *AAAI 2020*.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018b. No metrics are perfect: Adversarial reward learning for visual storytelling. *CoRR*, abs/1804.09160.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

Pengcheng Yang, Fuli Luo, Peng Chen, Lei Li, Zhiyi Yin, Xiaodong He, and Xu Sun. 2019. Knowledgeable Storyteller: A commonsense-driven generative model for visual storytelling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5356–5362. International Joint Conferences on Artificial Intelligence Organization.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling. *CoRR*, abs/1811.05701.

Yang Yu, Kazi Saidul Hasan, Mo Yu, Wei Zhang, and Zhiguo Wang. 2018. Knowledge base relation detection via multi-view matching. *New Trends in Databases and Information Systems*, page 286–294.

Disentangled Code Representation Learning for Multiple Programming Languages

Jingfeng Zhang[1] **Haiwen Hong**[1] **Yin Zhang**[1]* **Yao Wan**[2] **Ye Liu**[3] **Yulei Sui**[4]
[1] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]School of Computer Sci. & Tech., Huazhong University of Science and Technology, China
[3] Department of Computer Science, University of Illinois at Chicago, IL, USA
[4] School of Computer Science, University of Technology Sydney, Australia
`{zhjf, honghaiwen, zhangyin98}@zju.edu.cn`, `wanyao@hust.edu.cn`,
`yliu279@uic.edu`, `yulei.sui@uts.edu.au`

Abstract

Developing effective distributed representations of source code is fundamental yet challenging for many software engineering tasks such as code clone detection, code search, code translation and transformation. However, current code embedding approaches that represent the semantic and syntax of code in a mixed way are less interpretable and the resulting embedding can not be easily generalized across programming languages. In this paper, we propose a disentangled code representation learning approach to separate the semantic from the syntax of source code under a multi-programming-language setting, obtaining better interpretability and generalizability. Specially, we design three losses dedicated to the characteristics of source code to enforce the disentanglement effectively. We conduct comprehensive experiments on a real-world dataset composed of programming exercises implemented by multiple solutions that are semantically identical but grammatically distinguished. The experimental results validate the superiority of our proposed disentangled code representation, compared to several baselines, across three types of downstream tasks, i.e., code clone detection, code translation, and code-to-code search.

1 Introduction

Code representation learning has become an essential technique to support various software engineering tasks. Most of previous code representation learning approaches (Chen and Zhou, 2018; Jain et al., 2020; Nie et al., 2020) focus on a particular programming language, while learning code representations for multiple programming languages, though challenging, is an important step towards more generalizable and interpretable code embeddings. In principle, code snippets can be seen as code token sequences where their structural information often manifests as tree or graph data structures like AST (Abstract Syntax Tree). The downstream tasks often take full advantage of different code modalities (DQ et al., 2019) (e.g., structural information and textual tokens in the form of natural languages) to achieve better performance.

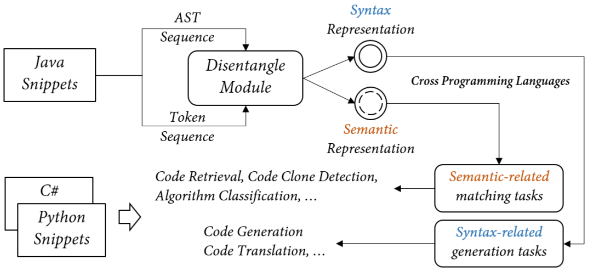

Figure 1: We disentangle the code representation into semantic and syntactic parts. The semantic part, which is relevant to code functionality but is often independent of a specific language, can be reused for semantic-related tasks across programming languages. The syntax part, which is related to a particular language but does not represent the code functionality, can be reused to control syntactic transformations for cross-programming language generation tasks.

It is noteworthy that syntax-level noise is an important issue in cross-language semantic-related tasks. Simply mixing textual token information and structural information of code (e.g., ASTs) often can not boost the performance on the cross-language code tasks. In this paper, we investigate a new approach that disassociates the latent semantic and syntactic representations of multi-lingual code snippets. The semantic representation excluding syntax information is more suitable for cross-language semantic-related code tasks as shown in Figure 1. We therefore study a new multi-lingual AST-guided code disentanglement technique called CODEDISEN, in order to provide a disentangled representation of code snippets that separates the la-

---

*Corresponding author: Yin Zhang

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4454–4466
August 1–6, 2021.

tent syntax representations for masked ASTs from its latent semantic counterpart for solving a particular programming exercise. Our new multi-lingual code representation disentanglement approach effectively utilises available linguistic resources, i.e., ASTs and textual code tokens. Overall, the main contributions of this paper are as follows:

- To the best of our knowledge, it is the first time that we formulate the code representation learning problem from the perspective of disentangling code semantics and syntax information across multiple programming languages.
- We propose an AST-guided disentangled code representation learning approach for multiple programming languages. We employ masked AST information to guide the disentanglement of code semantics and syntax, and design a cross-language reconstruction loss and a posterior distribution loss for modeling the fact that programs written in different languages for the same problem can share the similar program semantic. Furthermore, attentive code position loss can effectively fuse AST information into an effective code representation.
- To validate the effectiveness of our approach, we have conducted extensive experiments on three downstream tasks (i.e., code-to-code search, code translation and clone detection). Experimental results show that the latent semantic and syntax representation learned by our approach are nearly orthogonal, and the learnt disentangled semantic representation can significantly boost the performance of the downstream cross-language tasks.

2 Preliminaries

2.1 Code Syntax and Semantics

The Abstract Syntax Tree (AST) is an abstract representation of the syntax structure of source code. As shown in Figure 2, compilation nodes, e.g. `augment_list`, represent syntactic information, and leaf variable nodes, e.g. `range`, represent semantic information. In this paper, we parse the source codes into ASTs by using tree-sitter[1], an open source syntax parser, which supports multiple programming languages. We traverse the nodes of an AST based on the depth first algorithm, and consider the traversed paths as syntax representation of the code snippets. Using AST paths can significantly reduce the learning effort to extract grammatical information of code. *To restrict the AST paths to syntax information only, we masked leaf nodes during the traversal because the semantic information of the code snippets often comes from the leaf variable nodes.* Introducing masked AST paths to CodeDisen ensures that our approach can take some semantic meanings from the textual tokens in code snippets rather than ASTs, which can be used to learn the general syntax representation adhering to a specific language.

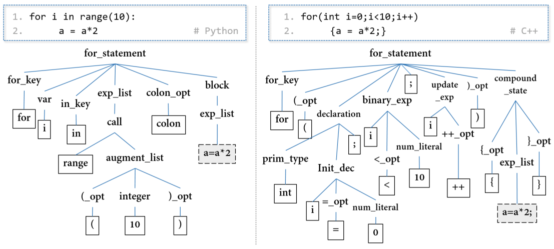

Figure 2: Python and C++ code snippets with their ASTs for the same problem. The solid boxes represent the leaf nodes. Note that the compilation nodes for a=a$\star$2 are almost identical.

2.2 Problem Statement

We denote code snippets for solving the same programming problem j as $\{\langle x_1, \ldots, x_n \rangle | x_i \in P_j\}$, where x_i is the solution of programming language i. In our experiments, we tested Java, Python, C++ and C#, thus the number of languages is $n = 4$. For each code snippet x_i, we construct a raw representation vector $\langle x_i, x_i^{ast} \rangle$, where x_i denotes a sequence of tokens, and x_i^{ast} represents syntax information derived from the abstract syntax tree of code snippet x_i.

For the same problem j, the code snippets $x_1, \ldots, x_n$ of multiple languages share the same semantic, although they have different programming language syntax z_i. Variants of Variational AutoEncoders (VAE) have been proposed to encode the raw vector $\langle x_i, x_i^{ast} \rangle$ into the latent representation. We aim to disentangle latent representation into two untangled parts: semantic y and syntax z latent representation of code. Formally, the objective of encoding is $\langle x_i, x_i^{ast} \rangle \rightarrow \langle y_i, z_i \rangle$ for each code snippet. For that purpose, we have to add multiple additional losses to the VAE architecture to enforce the effective disentanglement of code semantic and syntax. Next, we will introduce the design of multiple additional losses to effectively enforce disentanglement for code representation learning under the multi-lingual setting.

---

[1] `https://tree-sitter.github.io`

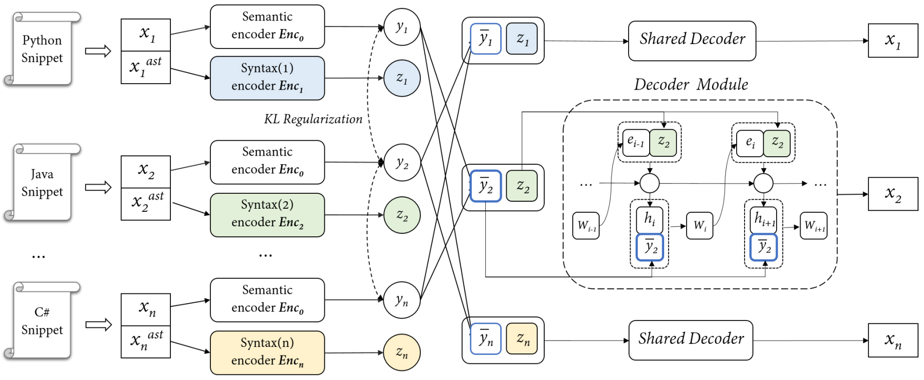

Figure 3: The reconstruction of code snippets written in multiple programming languages for the same problem. Enc_0 corresponds to the Semantic (Y) Encoder shared parameters among all programming languages. Enc_i corresponds to the Syntax (Z) Encoder dedicated to programming language i with independent parameters (*IZ*). The *KL divergence term* is imposed to the semantic latent variable *y* to enforce the alignment of different code snippets in the semantic space. The *decoder* also shares the parameters among all programming languages.

3 CODEDISEN Approach

Our approach is a variant of notable VAE architecture. In this paper, we start from the vMF-Gaussian Variational Autoencoder (VGVAE) model (Chen et al., 2019b), which is proposed to disentangle textual semantics from language syntax within the same human language. Our problem setting differs from disentanglement setting of human language in that: (1) We focus on dealing with multiple programming languages, instead of a single language. (2) Unlike human languages, programming language is a formal symbol system and has much stricter syntax rules than human languages, so we can make use of the AST information derived from a code snippet. To handle multiple programming languages, we propose CODEDISEN which adds more inputs and multiple losses to the VGVAE to effectively enforce the disentanglement of code semantics and syntax.

Figure 3 shows the overall architecture of our CODEDISEN approach. Unlike the unsupervised VGVAE, our approach introduces the masked AST x_i^{ast} that just retains the syntax information and removes almost all semantic information by masking leaf nodes. x_i^{ast} provides a strong supervision signal for disentangling code semantics y_i from code syntax z_i. For brevity, we will describe the factorization process from perspective of single code snippet. Following the conditional independence assumption in the graphical model, the joint probability $p_\theta(x, x^{ast}, y, z)$ can be factorized as:

$$\begin{aligned} &p_\theta(x, x^{ast}, y, z) \\ &= p_\theta(y)p_\theta(z)\prod_{t=1}^{T} p_\theta(w_t|w_{1:t-1}, y, z)p(x^{ast}|x), \end{aligned} \tag{1}$$

where w_t is the t-th word of x and $p_\theta(w_t|w_{1:t-1}, y, z)$ is given by a softmax over a vocabulary $\mathcal{V}$, $p(x^{ast}|x)$ is a deterministic transformation process. Different from the VGVAE variants (Chen et al., 2019b), we propose the following factorization $q_\varphi(y, z|x, x^{ast}) = q_\varphi(y|x)q_\varphi(z|x^{ast})$ to approximate the posterior when applying neural variational inference, since x^{ast} just retains the syntax information, and x contains more semantic information that is missing in compilation nodes. The objective of VAE is to maximize a lower bound of marginal log-likelihood, thus the basic loss $\mathcal{L}_0$ is written as:

$$\begin{aligned} \mathcal{L}_0 = &-\mathop{\mathbb{E}}_{y,z\sim\, q_\theta(y|x), q_\theta(z|x^{ast})} \log p_\theta(x|y, z) \\ &+ \mathrm{KL}\left(q_\theta(y|x) \parallel p_\theta(y)\right) + \mathrm{KL}\left(q_\theta(z|x^{ast}) \parallel p_\theta(z)\right). \end{aligned} \tag{2}$$

3.1 Encoders and Decoder

In this paper, we assume that $q_\theta(y|x)$ follows a vMF distribution (Chen et al., 2019b) and $p_\theta(y)$ follows the uniform distribution $vMF(\cdot; 0)$. Similarly, we assume that $q_\theta(z|x^{ast})$ follows a Gaussian distribution $\mathcal{N}(\mu_\beta(x^{ast}), diag(\sigma_\beta(x^{ast})))$ and that the prior $p_\theta(z)$ is $\mathcal{N}(0; I_d)$, where I_d is a $d \times d$ identity matrix. Concretely, we implement the *semantic encoder* Enc_0 (i.e., $q_\varphi(y|x)$) *shared among all languages* as a bidirectional long short-term memory network (BiLSTM) followed by a 3-layer feedforward neural network. Similarly, we adopt an independent BiLSTM model followed by a 3-layer feedforward network for each syntax encoder Enc_i adhering to programming language i (i.e., $q_{\varphi_i}(z_i|x_i^{ast})$). We also select LSTM model as the shared decoder of our generative model. As shown in Figure 3, at the decoding stage, we concatenate the syntactic variable z with the previous word's embedding as the input to calculate hidden state h' since grammatical information is more influenced by code token positions. Furthermore, we concatenate the semantic variable y with hidden state h' to predict the code token at each step, which could make full use of semantic information.

3.2 Losses for Disentanglement

In order to effectively enforce the disentanglement of code semantics and syntax, we design three additional loss terms, in addition to the loss $\mathcal{L}_0$.

Cross-Language Reconstruction Loss Since the code snippets $\{\langle x_0, x_1, \ldots, x_n\rangle \,|x_i \in P_j\}$ solve the same problem of P_j, they should share the same program semantics, inducing the cross-language reconstruction loss. Concretely, we hope that code snippet x_i can be reconstructed from its own syntax representation z_i and semantic representation $\overline{y}_i$. $\overline{y}_i$ is derived from latent semantic representations $\{y_k|k \neq i\}$ of code snippets $\{x_k|k \neq i\}$ that do not use language i. Formally, $\langle \overline{y}_i, z_i\rangle \to \langle x_i\rangle$. If we can regenerate x_i successfully, it means that $\{y_i\}$ share the almost same program semantic for problem P_j, and z_i encodes language-specific syntax information of programming language i.

As shown in Figure 3, at each step, we input $X = \{x_1, x_2, \ldots, x_n\}$, which is a set of code snippets that have the same program semantics p_j for problem j and are written in distinct programming languages. Formally, the cross-language reconstruction loss can be formulated as:

$$\mathcal{L}_{rec} = -\sum_{i=1}^{n} \mathbb{E}\left[\log p_\theta(x_i|\overline{y}_i, z_i)\right], \tag{3}$$

where $\overline{y}_i$ is calculated as:

$$\overline{y}_i = F_{Linear}(f_{cat}(\overline{Y}_i)), \tag{4}$$

where $\overline{Y}_i$ represents all the latent semantic variables except y_i, i.e., $\overline{Y}_i = \{y_k|k \neq i\}$, $f_{cat}(\cdot)$ is the function of concatenation, and F_{Linear} aims to fuse the concatenated vector to the same dimension as y_i, through a linear layer.

Posterior Distribution Loss Since all the code snippets of n programming languages for the same problem j share the same program semantics p_j, we expect that the posterior distribution $q_{\varphi_i}(y_i|x_i)$ of code snippets of programming language i should be close to the mean posterior distribution $q_m(y_m|x_m)$ of code snippets of all programming languages.

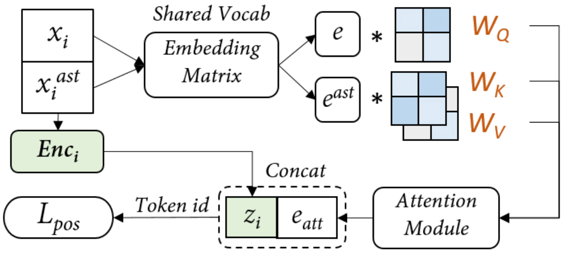

Figure 4: An illustration for the attentive position loss of language i at encoding stage.

Concretely, we employ KL terms (Chen and Zhou, 2018) to constrain the distribution discrepancy between $q_{\varphi_i}(y_i|x_i)$ and $q_m(y_m|x_m)$ in the latent space. Formally, The posterior distribution constraint loss for programming language i is defined as:

$$\begin{aligned}\mathcal{L}_{dist(i)} &= \mathrm{KL}(q_{\varphi_i}(y_i|x_i) \,\|\, q_m(y_m|x_m)),\\ q_m(y_m||x_m) &= vMF(\frac{\sum_{i=0}^{n}\mu(x_i)}{n}, \frac{\sum_{i=0}^{n}\kappa(x_i)}{n^2}),\end{aligned} \tag{5}$$

where vMF is the same definition as in (Chen et al., 2019b). The whole posterior distribution loss function is defined as:

$$\mathcal{L}_{dist} = \sum_{i=1}^{n} \mathcal{L}_{dist(i)}. \tag{6}$$

Attentive Code Position Loss As observed in (Chen et al., 2019b), the position information of code token x_i has a significant impact on its syntax, such as `import` is always at position `0` in Python. To better utilise ASTs to represent syntactic information, we introduce an *Token2AST* attention-based code position loss ($\mathcal{L}_{pos}$) to predict positions of code tokens based on the embedding e_i of x_i and the embedding e_i^{ast} of x_i^{ast}. We map the e_i^{ast} to the token side e_i via attention mechanism as shown in Figure 4. Firstly, there is a correlation between the AST nodes and the tokens, e.g., variable `a` is expected to have a higher weight with *identifier* in Figure 2. Secondly, the length of AST sequences is often much longer than that of the tokens in the code snippet. Therefore the attentive code position loss can fuse AST and token information to better extract syntactic features.

We implement $\mathcal{L}_{pos}$ for both encoder and decoder to predict code tokens position i, which consists of a 3-layer feedforward neural network $f(\cdot)$

with the input from the concatenation of the samples of the syntactic variable z and the attention embedding vector e_{att} at input position i. The $\mathcal{L}_{pos}$ and e_{att} are defined as:

$$e_{att} = \text{softmax}(\frac{(e_i \cdot W_q) \cdot (e_i^{ast} \cdot W_k^T)}{\sqrt{d}}) \cdot (e_i^{ast} \cdot W_v)$$

$$\mathcal{L}_{pos} = - \mathop{\mathbb{E}}_{z \sim q_{\varphi}(z|x)} \left[\sum_i \log \text{softmax}(f([e_{att}; z]))_t \right], \tag{7}$$

where d is the dimension of the embedding, to increase the training stability, $\text{softmax}(\cdot)_t$ indicates the probability of a code token at position t, and W_q, W_k, and W_v respectively denotes the query, key and value matrix in the attention mechanism.

Overall Objective We subsequently define the overall objective as the combination of the aforementioned basic loss and three additional losses for disentanglement. The total loss function is formulated as follows:

$$\mathcal{L} = \mathcal{L}_0 + \alpha \cdot \mathcal{L}_{rec} + \beta \cdot \mathcal{L}_{dist} + \gamma \cdot \mathcal{L}_{pos}. \tag{8}$$

4 Experiment and Analysis

In this section, we aim to address the following research questions: (1) *Can the program semantics and syntax be successfully disentangled by our proposed* CODEDISEN? (2) *Will disentangled semantics indeed improve the performance of downstream tasks*? and (3) *What is the generalizability of disentangled code representation across different programming languages*? We also perform ablation analysis to investigate the effect of each module of the model, as well as a qualitative analysis of detailed examples.

To answer the above questions, our experiments will validate the following two principles: ***(1) Equivalence of Semantics.*** Given a sequence of semantically identical code snippets $x_1, \ldots, x_n$ and their corresponding masked ASTs $x_1^{ast}, \ldots, x_n^{ast}$, CODEDISEN will yield y_i and $\overline{y}_i$ (see Eq. (4)), if $x_i = Decode(y_i, z_i) = Decode(\overline{y_i}, z_i)$, then we have $y_i = \overline{y_i}$, which means the shared semantic encoder Enc_0 extracts the same features for those code snippets. Further, given $x_i \neq x_j$ and $y_i = \overline{y}_i = y_j$, we have $z_i \neq z_j$, which means $Enc_{1 \ldots n}$ yields the respective syntax representations adhering to programming language $1 \ldots n$. ***(2) Orthogonality of Semantic and Syntax Vector.*** If the semantic vector y is completely disentangled from the syntax vector z, just applying y to downstream tasks will improve the performance, compared to applying x to downstream tasks. However, just applying z will perform poorly.

Implementation Details As for building vocabulary, we observe that more than 95% of the vocabulary of multi-lingual code snippets are user-defined variable names, with a tiny percentage of keywords and compilation nodes with respect to each programming language. Additionally, variable names in different code snippets for the same problem are likely to share semantics, which can facilitate implicit alignment of the semantics of code snippets of different languages. Hence, we resort to constructing a *shared vocabulary* for all code snippets and ASTs of all programming languages. More implementation details are referred to the Appendix A.1.

4.1 Dataset and Downstream Tasks

For multi-lingual cross-training, we use the CLCDSA dataset (Nafi et al., 2019), which is composed of 26,000 code snippets across four programming languages (i.e., Java, Python, C# and C++). This dataset is collected from three open source programming contest sites (i.e., AtCoder[2], Google CodeJam[3] and CoderByte[4]). All solutions in this dataset are functionally similar but written in different programming languages. In our experiments, we choose Java, Python, C# and C++ as the target languages, and limit the maximum code tokens length to 128. Consequently, we obtain a training dataset containing $2,500$ samples per language, and 500 samples for both validation and testing.

Code Clone Detection Code cloning across languages, which reuses a fragment of source code via copy-paste-modify, is a common way for code reuse and software prototyping. We treat the solutions belonging to different languages for the same problem as positive samples and the other random solution combinations in each batch as negative samples. We control the number of positive/negatives samples are balanced. We set the threshold as 0.8, which means that the cross-language input code pairs are semantic identical if the cosine similarity between them is greater than 0.8. For evaluation, we select LSTM (Sundermeyer et al., 2012), Tree-LSTM (Shido et al., 2019), TBCNN (Mou et al., 2016) and GraphCodeBERT (Guo et al., 2020) models as baselines.

---

[2] https://atcoder.jp/

[3] https://codingcompetitions.withgoogle.com/codejam

[4] https://coderbyte.com/

Table 1: The performance of CODEDISEN on code reconstruction of Python w.r.t different data size and training languages.

Method	*Size*	*Languages in Training*	**BLEU-1**	**CIDER**	**ROUGE-L**
VGVAE	1k	*Java/Python*	29.033	36.372	49.953
CODEDISEN	1k	*Java/Python*	29.750	40.102	50.079
	1k	*Java/Python/C#/C++*	34.861	47.455	55.616
CODEDISEN	2.5k	*Java/Python*	34.519	48.039	51.802
	2.5k	*Java/Python/C#/C++*	**44.765**	**116.90**	**59.159**

Code-to-Code Search During software development process, developers often look for code snippets that offer similar functionality (Kim et al., 2018). Our goal is to search the code snippet of other programming languages with the same functionality based on the current code snippet. To be more challenging, there is only one code snippet matching the query in the queried collection. We compare the code snippet in the source language with all code snippets in the target language to calculate their cosine similarity. For evaluation, we select BiLSTM (Linhares Pontes et al., 2018), Tree-LSTM (Shido et al., 2019), TBCNN (Mou et al., 2016) and GraphCodeBERT (Guo et al., 2020) models as baselines, and we adopt Accuracy, MRR and NDCG as evaluation metrics.

Code Translation In cross-language reconstruction, we know that x_i and x_j are source and target code fragments, which are semantically identical and belong to the same problem P_j. However, in cross-language code translation, we do not know x_j and have to sample a random code snippet $x_j^{'}$ in language j to obtain syntactic features $z_j^{'}$. We use this task to demonstrate that our model extracts non-zero and identical syntactic features for the same programming language. In addition, we use Tree-LSTM and VGVAE as baselines, to demonstrate the superior performance of our model on cross-programming language tasks.

4.2 Disentangled or Not? (RQ1)

To check the equivalence of semantics, we conduct experiment of reconstruction on Python code snippets. For a given Python code snippet x_i in the test set, our CODEDISEN yields the *aggregated semantic vector* $\overline{y}_i$ from semantically identical code snippets of other programming languages, i.e. Java or Java/C#/C++, as well as the *syntax vector* z_i from the Python code snippet x_i. Then $\overline{y}_i$ and z_i are jointly used to reconstruct the Python code snippet $\overline{x}_i$. We adopt the BLEU (Papineni et al., 2002), CIDER (Vedantam et al., 2015) and ROUGE (Lin, 2004) to measure the quality of reconstructed text $\overline{x}_i$ from x_i.

Table 1 shows the reconstruction performance of our CODEDISEN under various multi-lingual settings with different sizes of dataset. From this table, we observe that our model which is trained using AST information of 1,000 ($1k$) Java and 1,000 ($1k$) Python programs significantly outperforms vanilla VGVAE. It is also interesting to find that our model achieves a significant performance improvement when (1) we increase the training data from $1k$ to $2.5k$ samples for each language and (2) expand the bi-lingual model to a multi-lingual architecture (Java/Python/C#/C++). Furthermore, when comparing with VGVAE, CODEDISEN achieves 15.7%, 80.6% and 9.2% performance gains in terms of BLEU-1, CIDER and ROUGE-L, respectively. It is worth noting that CODEDISEN when trained using $1k$ samples still outperforms the bi-lingual model trained using $2.5k$ samples. The total dataset sizes used for training CODEDISEN are $1k \times 4 = 4k$ and $2.5k \times 2 = 5k$. This indicates that the multi-lingual architecture is good at dealing with more languages in training, since variable names may share similar semantics across different programming languages.

To further check the orthogonality of semantic and syntax vectors, we conduct experiments using the shared semantic vector and language-specific syntax vectors on downstream tasks. As shown in Table 2, CODEDISEN (Y) denotes only using the output y of the shared semantic encoder Enc_0 in code-to-code search. The performance has a significant improvement compared to BiLSTM without Enc_0. CODEDISEN (Z) means only using the output z of the syntax encoder, whose performance even has a dramatic drop. This indicates that the hidden vector y contains rich semantic information, while the hidden vector z rarely contains semantic information. As shown in Table 3, CODEDISEN (R) means randomly sampling a code snippet $x_j^{'}$ in the training set to extract syntactic feature $z_j^{'}$ for reconstruction. We can observe that little degradation in model performance indicates that the syntax vector of randomly sampled $x_j^{'}$ is almost the same as that of the original code snippet x_j of the same language j. When we set the variable z to zero tensor, we find that the model performance drops significantly. It confirms that the syntax vector z is critical in the reconstruction process and z is almost identical within the same programming language.

Table 2: Effectiveness of shared semantic encoder of our model in code-to-code search (2.5k/Java/Python).

Method	**ACC**	**MRR**	**NDCG**
BiLSTM	0.166	0.275	0.413
Tree-LSTM	0.081	0.162	0.293
TBCNN	0.007	0.024	0.169
GraphCodeBERT	0.246	0.314	0.343
CodeDisen (Y)	**0.316**	**0.436**	**0.551**
CodeDisen (Z)	0.004	0.021	0.157

Table 3: Effectiveness of the syntax encoder of CodeDisen in code translation (1k/Java/Python).

Method	**BLEU-1**	**ROUGE-L**	**CIDER**
Tree-LSTM	25.84	37.53	36.96
VGVAE	29.03	49.95	36.37
VGVAE (R)	24.94	36.09	25.09
CodeDisen	**29.75**	**50.08**	**40.10**
CodeDisen (R)	29.51	48.64	38.92
CodeDisen (0)	13.82	15.55	1.73

Table 4: Effectiveness of semantic encoder of our model in code clone detection (2.5k/Java/Python).

Method	**Precision**	**Recall**	**F1**
LSTM	0.85	0.75	0.79
Tree-LSTM	0.78	0.84	0.81
TBCNN	0.50	**0.99**	0.66
GraphCodeBERT	0.56	0.54	0.50
CodeDisen (Y)	**0.88**	0.93	**0.90**
CodeDisen (Z)	0.50	0.33	0.38

4.3 Downstream Task Performance (RQ2)

To better verify whether the disentangled multi-lingual code semantic representation can boost the performance of downstream tasks, we fine-tune the model on the downstream tasks of code translation, code-to-code search and code clone detection under the cross-language setting.

As shown in Table 4, CodeDisen (Y) that only considers the semantics of code achieves the best performance, significantly outperforming the performance of counterpart CodeDisen (Z) that only considers the syntax of code. We set the threshold value to 0.8 according to the testing performance on code clone detection task. When we set the threshold to 0.5, the performances of Tree-LSTM and CodeDisen are 0.576/0.954/0.718, and 0.724/0.992/0.837, in terms of Precision, Recall and F1, respectively. This is because that if we set the threshold to a lower value, more code snippets may be classified as duplicates, thus the recall increases while the precision decreases. Therefore, we choose a threshold of 0.8 to better compare the differences in performance between models.

It is noteworthy that the models such as Tree-LSTM and TBCNN, which accept ASTs of a program as their inputs can obtain high recall but low precision. This indicates that if the ASTs are same, to a large extent, the two programs can be considered as semantically identical, so the recall is high. However, the ASTs of different programming languages vary greatly and generate many temporal variables during compilation, thus introducing noise nodes, so the precision can be low. Our approach combines the advantages of token and AST features while obtaining high precision and recall on cross-programming language tasks. GraphCodeBERT, a pre-trained model on the code corpora of multi-programming language, is suitable for fine-tuning on specific task of a programming language. For the task of code clone detection, we simply fine-tune the model based on the released checkpoint of GraphCodeBERT, under the setting of our scenario. For the task of code-to-code search, we extract the last layer of GraphCodeBERT output and take the average value as the feature of the code segment, and calculate the cosine similarity to select the target from candidates, as described in the Appendix A.3. As shown in Table 2 and Table 4 , we can find that GraphCodeBERT does not adapt well to cross-programming language semantic matching related tasks.

Table 3 shows that although Tree-LSTM is more suitable for encoding the structure information of code than our LSTM-based model, our CodeDisen (R) still outperforms Tree-LSTM in BLEU-1, ROUGE-L, CIDER by 3.7%, 11.1%, 2.0%, respectively. By introducing AST information, our CodeDisen (R) also has a significant improvement when compared to VGVAE. Table 2 shows that Tree-LSTM and TBCNN perform poorly for cross-language code search tasks. The main reason is that both Tree-LSTM and TBCNN are based only on the input representation of an AST. However, the ASTs of two semantically equivalent programs written in two different languages (e.g., Java and Python) can be generated quite differently by the compilers of these two languages, hence introducing syntax-level noise.

4.4 Generalizability of CodeDisen (RQ3)

To investigate the generalizability of our semantic module across languages, we evaluate our model on unseen datasets in different languages. In addition, we compare the performance when combining different languages in the code-to-code search task to demonstrate the superiority of multi-lingual structures. From Table 5, we observe that when

Table 5: Generalization ability of semantic encoder.

Languages (Testing)	ACC	MRR	NDCG	Languages(Training)
Java-Python	0.316	0.436	0.551	*Java/Python*
Java-C#	0.298	0.420	0.538	*Java/Python*
C++-C#	0.234	0.351	0.481	*Java/Python*
Java-Python	**0.330**	**0.452**	**0.564**	*Java/Python/C#/C++*
C++-Python	0.255	0.375	0.499	*Java/Python*
C++/Python	0.279	0.402	0.519	*Java/Python/C#*

Table 6: Ablation study of CODEDISEN, where *IZ* denotes independent syntax encoders $Enc_{1\sim n}$ and *KL* denotes semantic KL term.

Method	BLEU-1	ROUGE-L	CIDER
CODEDISEN	44.77	59.16	116.9
-IZ	32.12	45.82	27.40
-KL	36.71	53.89	67.06
-L_{pos}	41.19	55.05	110.1
L_{pos}-att	41.93	56.90	102.6

we use the shared semantic encoder trained on the Java/Python dataset, our model still achieves good results on C++-C# and Java-C# code-to-code search tasks after fine-tuning. Note that C++-C# data are not there when training CODEDISEN , and our model keeps most of its performance on Java-Python dataset in Table 2. This is a good evidence that the semantic representations extracted by our model are generalizable across languages.

For Java-Python code search, the code semantic encoder trained on four languages (Java/Python/C#/C++) performs better than the one trained on two language (Java/Python). For C++-Python code search, we ensure the training dataset free of C++ code snippets. We find that the code semantic encoder trained on Java/Python/C# performs better than the one trained on Java/Python. These indicate that our multi-lingual architecture can further utilise the samples of more programming languages to train a better semantic encoder, and is extensible to train more language-specific syntax encoders.

4.5 Ablation Study

We conduct ablation analysis to understand the performance contribution from different component in our model. As shown in Table 6, we choose the model trained on four languages as the baseline (CODEDISEN). In fact, we find that the independent *Syntax* encoders (IZ) and KL term (KL) have a significant impact on the multi-lingual model. When we remove these components, the BLEU-1 scores of our model drop by 12.65% and 8.06%. This suggests that the design of implicit semantic alignment and syntactic independence between multiple programming languages is effective.

We also explore the role of attention in the code position loss, while AST sequences are usually much longer and more complicated than code tokens. The results show that when we use the code position loss without *Token2AST* attention (L_{pos}-att), performance of (L_{pos}-att) is close to that of (-L_{pos}) removing code position loss. It means our *Token2AST* attention mechanism could merge the syntactic AST features and the semantic features of tokens to handle the long sequence dependence.

4.6 Qualitative Analysis

We conduct case study to further investigate results of the semantic extraction in code refactoring and abstract syntax representation, as shown in Table 7. From *Case 1*, it is clear to see that the variable names in the generated snippets are consistent with the semantic input Java snippets. Then we compare the semantic information between the generated and the input semantic code pairs. As shown in *Case 2*, the syntax input does not have "`Yes`" or "`No`" at all, but our generated snippet extracts this from the semantic input very well. In addition, we have rewritten the complex multivariate input form of Java into the simple map input of Python, which demonstrates that our model can extract semantics well. On the other hand, we find that the generated snippets are compliant with Python syntax. In conjunction with the random syntax sampling discussed earlier, we can further show that the syntax variables we extracted abstractly represent the syntax of specific programming language.

5 Related Work

Deep Code Representation The existing code representation works represent code snippets in three ways, i.e., token-based representation, AST-based representation, and graph-based representation. As for token-based representation (Hindle et al., 2012; Bhoopchand et al., 2016), code snippets are tokenized into token sequences and each code token is represented as a real-valued vector. As for AST-based representation, one line of work is to directly represent the tree structure via Tree-CNN (Mou et al., 2016) or Tree-LSTM (Chen et al., 2018) . Another line of work is to indirectly represent the AST by linearizing the AST into a sequence of nodes (Hu et al., 2018; Alon et al., 2019; Liu et al., 2020) via traversing. Wan et al. (Wan

Table 7: Example of generated results by code translation (*Java/Python*).

	Case 1
Semantic	Scanner sc = new Scanner(System.in); int a = sc.nextInt(); int b = sc.nextInt(); if(a < b) System.out.println (b); else System.out.println (a);
Syntax	h1 = int (input ()) h2 = int (input ()) print (h1 - h2)
Reference	a , b = map (int ,input ().split ()) print (max(a , b))
Generated	a , b = map (int ,input ().split ()) print (a + b)
	Case 2
Semantic	Scanner sc = new Scanner (System.in); int A = sc.nextInt(); int B = sc.nextInt(); int C = sc.nextInt(); if(C <= A + B) System.out.println (“ Yes ”); else System.out.println (“ No ”);
Syntax	n = int (input ()) if n == 12 : print (1) else : print (n + 1)
Reference	A , B , C = map (int ,input ().split ()) if A + B < C : print (“ No ”) else : print (“ Yes ”)
Generated	A , B = map (int ,input ().split ()) if A == B : print (“ YES ”) else : print (“ NO ”)

et al., 2018; Wang et al., 2020b; Wan et al., 2019; Hua et al., 2021) propose to integrate the semantics of code from different views (e.g., the tokens, AST and control-flow graph) into a hybrid feature space, and put forward a hybrid representation approach, for the task of code summarization, code search and code clone detection. As for graph-based representations several works resort to parse the program into a graph (e.g., augmented AST, control-flow graph, and data-flow graph) (Li et al., 2015; LeClair et al., 2020; Wan et al., 2019; Sui et al., 2020; Guo et al., 2020). Benefiting from the strong power of pre-training technique in natural language processing, recently, several works (Kanade et al., 2020; Feng et al., 2020; Guo et al., 2020) propose to pre-train a masked language model on the large-scale of code corpora, like BERT (Devlin et al., 2019). CodeBERT (Feng et al., 2020) pre-trains a language model on the source codes and natural language descriptions, and significantly boosts performance on code search. GraphCodeBERT (Guo et al., 2020) advances the CodeBERT by incorporating the data-flow information among variables into pre-training.

Multilingual Knowledge Transfer For multilingual tasks, if we treat word embedding spaces isomorphic between different languages, which has been shown not to hold in practice (Søgaard et al., 2018), and fundamentally limits their performance. Sabet et al. (2019) train a bilingual model on bilingual corpora by introducing a cross-lingual loss in addition to the monolingual loss. The model learns to translate on each other by inputting parallel data sets at one step simultaneously. This ensures that the word and n-gram embeddings of both languages lie in the same space. Our approach is primarily referenced to text-controlled generation, which transfers the knowledge by dissociating tangled representations. Cross-training disentangling methods (Chen et al., 2019a,b; Wang et al., 2020a) on the controlled text generation task, which are implemented in a VGVAE framework and guided by paraphrase reconstruction loss have inspired us a lot. In particular, the syntax input of the code can be conveyed via AST. Code syntax regularity can be well exploited in multilingual architectures to achieve semantic alignment in dissociated latent spaces to improve the quality of representations with desirable generalizability.

6 Conclusion

In this paper, we propose a novel disentangled code representation learning approach under multilingual setting. We introduce three dedicated losses to enforce the disentanglement of code semantics and syntax. Comprehensive experiments on the three downstream tasks validate the effectiveness of our disentangled semantic and syntax representation. In the future, we will devise more effective disentanglement models for code representation learning. Another line is to extend the proposed approach to cross-lingual customer service robots, where answers of different languages for the same question share the same semantic information.

Acknowledgement

We thank the anonymous reviewers for their helpful comments. This work is supported by National Key R&D Program of China (No. 2018AAA0101900), the NSFC projects (No. 62072399, No. U19B2042, No. 61402403), Chinese Knowledge Center for Engineering Sciences and Technology, MoE Engineering Research Center of Digital Library, National Engineering Research Center for Big Data Technology and System, the Fundamental Research Funds for the Central Universities, and partially supported by Australian Research Grant DP21010134.

References

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29.

Avishkar Bhoopchand, Tim Rocktäschel, Earl Barr, and Sebastian Riedel. 2016. Learning python code suggestion with a sparse pointer network. *arXiv preprint arXiv:1611.08307*.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019a. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019b. A multi-task approach for disentangling syntax and semantics in sentence representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2453–2464, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingying Chen and Minghui Zhou. 2018. A neural framework for retrieval and summarization of source code. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 826–831. IEEE.

Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-tree neural networks for program translation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2552–2562.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bui Nghi DQ, Yijun Yu, and Lingxiao Jiang. 2019. Bilateral dependency neural networks for cross-language algorithm classification. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 422–433. IEEE.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*.

Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. 2012. On the naturalness of software. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 837–847. IEEE.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 200–20010. IEEE.

Wei Hua, Yulei Sui, Yao Wan, Guangzhong Liu, and Guandong Xu. 2021. Fcca: Hybrid code representation for functional clone detection using attention networks. *IEEE Transactions on Reliability*, 70(1):304–318.

Paras Jain, Ajay Jain, Tianjun Zhang, Pieter Abbeel, Joseph E Gonzalez, and Ion Stoica. 2020. Contrastive code representation learning. *arXiv preprint arXiv:2007.04973*.

Aditya Kanade, Petros Maniatis, Gogul Balakrishnan, and Kensen Shi. 2020. Learning and evaluating contextual embedding of source code. In *International Conference on Machine Learning*, pages 5110–5121. PMLR.

Kisub Kim, Dongsun Kim, Tegawendé F Bissyandé, Eunjong Choi, Li Li, Jacques Klein, and Yves Le Traon. 2018. Facoy: a code-to-code search engine. In *Proceedings of the 40th International Conference on Software Engineering*, pages 946–957.

Alexander LeClair, Sakib Haque, Linfgei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. *arXiv preprint arXiv:2004.02843*.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Elvys Linhares Pontes, Stéphane Huet, Andréa Carneiro Linhares, and Juan-Manuel Torres-Moreno. 2018. Predicting the semantic textual similarity with Siamese CNN and LSTM. In *Actes de la Conférence TALN. Volume 1 - Articles longs, articles courts de TALN*, pages 311–320, Rennes, France. ATALA.

Fang Liu, Lu Zhang, and Zhi Jin. 2020. Modeling programs hierarchically with stack-augmented lstm. *Journal of Systems and Software*, 164:110547.

Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1287–1293. AAAI Press.

Kawser Wazed Nafi, Tonny Shekha Kar, Banani Roy, Chanchal K Roy, and Kevin A Schneider. 2019. Clcdsa: cross language code clone detection using syntactical features and api documentation. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1026–1037. IEEE.

Lun Yiu Nie, Cuiyun Gao, Zhicong Zhong, Wai Lam, Yang Liu, and Zenglin Xu. 2020. Contextualized code representation learning for commit message generation. *arXiv preprint arXiv:2007.06934*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ali Sabet, Prakhar Gupta, Jean-Baptiste Cordonnier, Robert West, and Martin Jaggi. 2019. Robust cross-lingual embeddings from parallel sentences. *arXiv preprint arXiv:1912.12481*.

Yusuke Shido, Yasuaki Kobayashi, Akihiro Yamamoto, Atsushi Miyamoto, and Tadayuki Matsumura. 2019. Automatic source code summarization with extended tree-lstm. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788, Melbourne, Australia. Association for Computational Linguistics.

Yulei Sui, Xiao Cheng, Guanqin Zhang, and Haoyu Wang. 2020. Flow2vec: value-flow-based precise code embedding. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–27.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4566–4575. IEEE Computer Society.

Yao Wan, Jingdong Shu, Yulei Sui, Guandong Xu, Zhou Zhao, Jian Wu, and Philip S Yu. 2019. Multi-modal attention network learning for semantic source code retrieval. *arXiv preprint arXiv:1909.13516*.

Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 397–407.

Shaonan Wang, Jiajun Zhang, Nan Lin, and Chengqing Zong. 2020a. Probing brain activation patterns by dissociating semantics and syntax in sentences. In *AAAI*, pages 9201–9208.

Wenhua Wang, Yuqun Zhang, Yulei Sui, Yao Wan, Zhou Zhao, Jian Wu, Philip Yu, and Guandong Xu. 2020b. Reinforcement-learning-guided source code summarization via hierarchical attention. *IEEE Transactions on Software Engineering*, pages 1–1.

A Appendices

A.1 Training Details

All the experiments are conducted on 2 Geforce GTX 1080Ti GPUs. It tasks about 4 hours to train CODEDISEN . For encoder and decoder networks of CODEDISEN, we use the same BiLSTM model structure. The embedding size is 100 and the hidden size is 100. The dimensionality of *Semantic* latent variable vector is 50. The dimensionality of *Syntax* latent variable vector is 50. Specially, the hidden size in feed-forward network and attention mechanism is also 100. The coefficient α of the cross-language reconstruction loss $\mathcal{L}_{rec}$ is 1.0. When calculating the KL divergence term, the coefficient β of the Posterior distribution loss $\mathcal{L}_{dist}$ is 0.1, the coefficient of the $vMF(\cdot)$ KL divergence is $1e$-4 and the coefficient of the $Gaussian(\cdot)$ KL divergence is $1e$-3. The coefficient γ of the attentive code position loss $\mathcal{L}_{pos}$ is 1.0. We train each model for 60 epochs and the batch size is 10 for each programming language.

A.2 Case Study

As shown in Table 8, the semantic inputs and reference code snippets are semantically identical yet grammatically different. Based on the semantic information extracted from a Java code snippet and the syntax information extracted from a random selected Python code snippet, our approach can generate a Python snippet similar to the reference Python snippet which is semantically similar to Java input. We find that the semantics of the snippet we generated and the reference snippet are very similar, especially the content of printed string, such as “`YES`” or “`NO`”, “`Even`” or “`Odd`”, even the rare words (“`Christmas Eve...`”). At the same time, the generated code snippets are completely unaffected by randomly sampled syntax input. This means that our semantic and syntactic disentanglement modules perform well in extracting shared semantic information from code snippets for the same programming exercise and general syntactic features belonging to specific programming language.

Note that our generative model will be deficient in reconstructing mathematical expressions. For example, the reference snippet is “`a%2==0`” or “`b%2==0`” and the generated is “`a%b`”. The main reason is that the specific content of mathematical expressions is less weighty in the semantic expression of a code snippet, and our model tends to focus more on generating an expression rather than on the content of expressions. Another drawback is that our model can not generate long code snippets well, e.g., `'')` is missing after the “`print(``Christmas Eve ... Eve`” in the third example. In the future, we will replace the original mathematical expressions with word descriptions of longer token length to increase the weight in reconstruction loss function. In addition, we will use tree-structured decoders to guarantee the executability of the generated code so as to increase long dependencies.

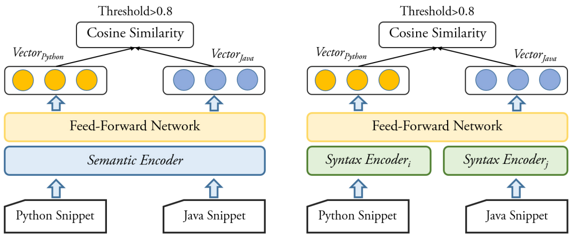

Figure 5: The framework of code clone detection. The left shows training on the semantic module and the right shows training on the specific syntax modules.

A.3 Architecture of Downstream Tasks

In this section, we detail the model architecture of cross-language code clone detection and code-to-code search tasks. The model for the code translation is identical to the cross-language reconstruction model used for the disentanglement training, except that the code snippets from which the syntactic latent variables are extracted are randomly sampled.

The key component of the proposed downstream tasks flow is the Bi-NN. It is modeled as two underlying subnetworks followed by a classification layer. In our work, the underlying subnetworks are semantic and syntax modules and other baseline networks such as BiLSTM. The classifier we defined as a 2-layer shared feed-forward network and calculate the cosine similarity of the input cross-language samples.

A.3.1 Code Clone Detection

Code cloning across languages, which reuses a fragment of source code via copy-paste-modify, is a common way for code reuse and software prototyping. We train and test the code clone detection task on *Java/Python*, *Python/C++*, *C++/C#* and *C#/Java* language pairs. In particular, we calculate the metric scores on average, as shown in Figure 5. We treat the solutions belong to different

Table 8: More examples of reconstructed results by random syntax (Java/Python).

	Code Snippets
Semantic	import java . util .*; public class Main public static void main (String [] args) Scanner sc = new Scanner (System . in); String [] line = sc . next Line (). split (“ ”); int r = Integer . parse Int (line [0]) * 100 ; int g = Integer . parse Int (line [1]) * 10 ; int b = Integer . parse Int (line [2]); int result = r + g + b ; if (result % 4 == 0) System . out . println (“ YES ”); return ; System . out . println (“ NO ”);
Syntax	s = input () num = “ ” for i in range (len (s)): if s [i] in “ 0123456789 ”: num += s [i] print (num)
Reference	x , y , z = input (). split () a = int (x + y + z) if a % 4 == 0 : print (‘ YES ’) else : print (‘ NO ’)
Generated	a , b , c = map (int , input (). split ()) if a % b == 0 : print (“ Yes ”) else : print (“ No ”)
Semantic	import java . util .*; public class Main public static void main (String [] args) Scanner sc = new Scanner (System . in); int a = sc . next Int (); int b = sc . next Int (); if (a % 2 == 0 || b % 2 == 0) System . out . print (“ Even ”); else System . out . print (“ Odd ”); sc . close ();
Syntax	N = int (input ()) ans = 0 for i in range (N): l , r = map (int , input (). split ()) ans += r - l print (ans + N)
Reference	a , b = map (int , input (). split ()) if a % 2 == 0 or b % 2 == 0 : print (‘ Even ’) else : print (‘ Odd ’)
Generated	a , b = map (int , input (). split ()) if a % b == 0 : print (“ Even ”) else : print (“ Odd ”)
Semantic	import java . io .*; import java . util .*; public class Main public static void main (String [] args) try Scanner sc = new Scanner (System . in); int d ; d = Integer . parse Int (sc . next ()); System . out . print (“ Christmas ”); for (int i = 0 ; i < 25 - d ; i ++) System . out . print (“ Eve ”); System . out . println (“”); catch (Exception e) System . out . println (“ out ”);
Syntax	n = int (input ()) k = int (input ()) if n > 2 * k : ans = “ YES ” else : ans = “ NO ” print (ans)
Reference	D = int (input ()) if D == 25 : print (“ Christmas ”) else : if D == 24 : print (“ Christmas Eve ”) else : if D == 23 : print (“ Christmas Eve Eve ”) else : print (“ Christmas Eve Eve Eve ”)
Generated	A , B = map (int , input (). split ()) if A == B : print (“ Christmas Eve Eve Eve Eve ”) elif D == 23 : print (“ Christmas Eve Eve Eve Eve Eve Eve Eve

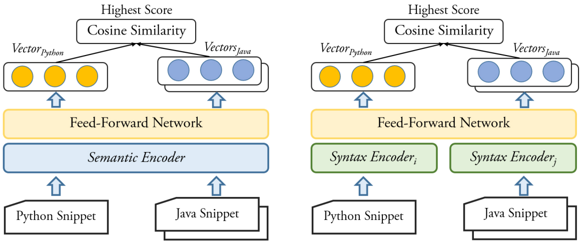

Figure 6: The framework of code-to-code search. Given a query code snippet written in Python as well as a series of candidate code snippets written in Java, the goal of code-to-code search is to retrieve the most relevant Java snippets based on cosine similarity.

languages for the same problem as positive samples and the other random solution combinations in each batch as negative samples. To be more challenging, we extracted 350 programming problems from CLCDSA dataset such that each problem has only one solution per language for evaluation. We control the number of positive/negatives samples are balanced. We set the threshold as 0.8(@80). It means that if the cosine similarity of cross-language input code pairs is greater than 80%, we consider them as semantic clone pairs. In addition, we use the semantic module and the syntax modules compared to baselines in Table 4 to validate that extracted semantics features could improve the performance and our syntax modules may perform poorly because of missing semantic information.

A.3.2 Code-to-Code Search

The training language pair combinations and dataset construction are the same as the code clone detection task. We control that each programming language has only one unique solution for each programming problem. When evaluating our model, we compare the code snippet in source query language to the all code snippets in target language, calculating their cosine similarity. Then we predict the type of algorithm by greedy choosing the highest score sample as shown in Figure 6. In contrast to the usual algorithm classification of one-hot tags, we chose to compare the similarity with all samples of the target domain to do code-to-code search. This makes the more difficult and convincing task to validate the quality of the semantic representation of the code.

Exploring Self-Identified Counseling Expertise in Online Support Forums

Allison Lahnala[1], **Yuntian Zhao**[1], **Charles Welch**[1], **Jonathan K. Kummerfeld**[1],
Lawrence An[2,4], **Kenneth Resnicow**[3,4], **Rada Mihalcea**[1], **Verónica Pérez-Rosas**[1]
[1]Computer Science & Engineering, University of Michigan
[2]Medical School, University of Michigan
[3]School of Public Health, University of Michigan
[4]Center for Health Communications Research, University of Michigan
`{alcllahn,clzhao,cfwelch,jkummerf,lcan,`
`kresnic,mihalcea,vrncapr}@umich.edu`

Abstract

A growing number of people engage in online health forums, making it important to understand the quality of the advice they receive. In this paper, we explore the role of expertise in responses provided to help-seeking posts regarding mental health. We study the differences between (1) interactions with peers; and (2) interactions with self-identified mental health professionals. First, we show that a classifier can distinguish between these two groups, indicating that their language use does in fact differ. To understand this difference, we perform several analyses addressing engagement aspects, including whether their comments engage the support-seeker further as well as linguistic aspects, such as dominant language and linguistic style matching. Our work contributes toward the developing efforts of understanding how health experts engage with health information- and support-seekers in social networks. More broadly, it is a step toward a deeper understanding of the styles of interactions that cultivate supportive engagement in online communities.

1 Introduction

Online social media forums play a critical role in health-related information sharing (Record et al., 2018). Health experts have noted that they can help reduce healthcare inequalities and improve access to health care, for instance by empowering coalitions of people living with chronic illness or specific disabilities (Griffiths et al., 2012), or by providing an anonymous forum for people seeking emotional support (De Choudhury and De, 2014). On the other hand, these forums elevate concerns about spreading medically inaccurate, misleading, or unsound information (Domínguez and Sapiña, 2015; Gage-Bouchard et al., 2018), which has had harmful public health impacts (Poland et al., 2011; Nobles et al., 2019). One study concluded that health information seekers in forums such as Reddit are likely to enact suggested behaviors regardless of perceived credibility (Record et al., 2018). However, the researchers also noted that this openness to information could be an opportunity for experts to encourage healthy behaviors through information sharing.

In this landscape, it is critical to understand the dynamics that cultivate safe communities that benefit the health and well-being of their participants and the broader implications for health communication (Chou et al., 2009). Health experts are thus considering social media's role in their interactions with patients and broader public health issues, and their role in engaging with the platforms (Domínguez and Sapiña, 2015; Nobles et al., 2019, 2020). This motivates an important research direction: understanding how experts engage with users in online platforms. This can inform platform design, moderation decisions, and health promotion efforts by experts.

This work focuses on understanding the engagement with professionals in the domain of mental health with two main research questions: (RQ1) Do experts have distinct influences as compared to non-experts in their interactions with support-seekers in online mental health?; and (RQ2) Do the experts' behaviors reflect established counseling principles and findings regarding behaviors associated with positive counseling outcomes? To answer these questions, we analyze responses from self-identified mental health professionals (MHP) to support-seekers in mental health and support communities on Reddit, and compare them to responses from other users who we refer to as *peers*. This is an important comparison, as many peers share similar health experiences, which prior work has found is associated with higher empathic concern (Hodges et al., 2010).

First, we test whether a text classifier can distin-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4467–4480
August 1–6, 2021.

guish between responses to support seekers from MHPs and peers. We find that it can, with 70% accuracy (well above random chance of 50%). Second, we analyze comments leading to further engagement with the support-seeking posters, as existing counseling principles emphasize the importance of eliciting client engagement in expert counseling sessions (Miller and Rollnick, 2012; Pérez-Rosas et al., 2018). Third, we analyze the users' linguistic tendencies, drawing inspiration from analyses of counseling conversations, which have offered insight into counselor behaviors associated with high quality sessions grounded in existing theories from psychology and counseling research using computational methods (Althoff et al., 2016; Pérez-Rosas et al., 2018; Zhang et al., 2019; Miller and Rollnick, 2012).

The main contributions of this work are: (1) We construct a dataset of mental health conversations from Reddit users with self-identified counseling expertise, covering a set of mental health subreddits annotated with categories denoting the type of mental health concern; (2) We develop a classifier that can distinguish between the language of MHPs and that of peers; (3) We perform an analysis of the differences in language use between MHPs and peers; and (4) We provide insight into language that leads to further engagement with support-seekers, comparing responses to peers and MHPs.

2 Related Work

Studies within the education and health domains have shown that advice and help-seeking interactions in online communities contribute positively to users' well-being, learning, and skills development (Campbell et al., 2016; Wang et al., 2015). This is particularly true for applications such as computer programming, career development, mentoring, coping with chronic or life-threatening diseases, and mental health issues (Baltadzhieva and Chrupała, 2015; Tomprou et al., 2019; Wang et al., 2015; De Choudhury and De, 2014).

In the mental health domain, studies have explored online support communities and many have found positive outcomes associated with anonymity, perceived empathy, and active user engagement (De Choudhury and De, 2014; Rheingold, 1993; Hodges et al., 2010; Welbourne et al., 2009; Nambisan, 2011). Computational approaches have aided studies in mental health forums, helping reveal positive relationships between linguistic accommodation and social support across subreddits (Sharma and De Choudhury, 2018). One example of insights from this work is that topic-focused communities like subreddits may enable more peer-engagement than non-community based platforms (Sharma et al., 2020). Other studies have revealed certain trade-offs of online support platforms, such as disparities in the level of support offered toward support-seekers of various demographics (Wang and Jurgens, 2018; Nobles et al., 2020) and in condolences extended across different topics of distress (Zhou and Jurgens, 2020). Studying MHP behaviors in such scenarios might help develop approaches that balance these trade-offs.

Computational approaches applied in these forums have also shed light on population-level health trends and health information needs, with examinations into how depression and post-traumatic stress disorder (PTSD) affect different demographic strata (Amir et al., 2019). Data mining has also been applied to understand adverse drug reactions (Wang et al., 2014) and public reactions towards infectious diseases (Park and Conway, 2017). Nobles et al. (2018) highlighted the potential for these forums to aid targeted health communication, for example by sharing information in r/STD, a subreddit about sexually transmitted diseases. Another case study of r/STD revealed the prevalence of diagnoses requests, and suggested that health professionals could partner with social media platforms to positively influence crowd-sourced diagnoses and help mitigate harmful misdiagnoses (Nobles et al., 2019). Record et al. (2018) found that health information seeking Reddit users are likely to enact suggested behaviors regardless of perceived credibility, providing further reason for health expert engagement to intervene when harmful information sharing occurs and promote healthy behavior.

Fewer studies have analyzed expert interactions in online forums. A study in a large Q&A community found that experts are more likely to provide help than peers and that their participation in discussions resulted in increased length and substance of discussions (Procaci et al., 2017). Recent studies have compared interactions with experts to interactions with peers in broader scientific communities (Park et al., 2020) and r/AskDocs on Reddit (Nobles et al., 2020). The latter paper closely relates to our study, as they also consider posts from experts on Reddit, but solely within r/AskDocs about different health topics and with

Subreddits	77
Posts	12,140
Poster Replies	24,357
MHPs	283
Peers	56,701
Comments	
MHP	9,685
Peer	92,698
Total	102,383
Thread Length	
Mean	8.4
Median	4
Max	64

Table 1: Dataset statistics.

Post:	u/peer_user_X
I've recently been struggling with paranoid thoughts, for which I was hospitalized for my own safety. I do not feel suicidal anymore, however everyday is a long struggle of thinking everyone is an undercover agent out to get me or keep tabs on what I'm doing. I was hoping to hear some tips and stories if anyone else has dealt with similar thoughts and overcome them? Or are they something I will have to deal with for the rest of my life? Thanks in advance	
Comment:	u/MHP_user (LPC)
Paranoid thoughts are scared thoughts, justified or not. If you ignore the specific content of the thoughts and focus on the emotional valence (scared), is there something you can do in those moments to feel safer?	
Poster Reply:	u/peer_user_X
That's a good way of thinking about the situations as they arise. I will try to do that	

Table 2: Example of an initial post, a reply from an MHP with the flair *LPC* (Licensed Professional Counselor), and a reply from the original user.

users of varying demographics.

The insights discussed above motivate investigations into how health experts and other users promote scientifically sound advice and offer supportive responses to health information seekers in online forums. In this work, we aim to contribute additional insights into expertise influence in online mental health communities by studying the dynamics of the communication process between support seekers and support providers.

3 Data Collection

We seek to understand the tendencies of users with professional experience, and more specifically counseling expertise, when interacting with support-seekers in online mental health and support-related forums. In uncovering which tendencies are associated with expertise, we enable further investigation into their role in the social dynamics of online support-seeking interactions, and potential applications of insight-driven recommendations for moderators and users of these forums.

Source. We use Reddit for its quantity of publicly available interactions in communities called *subreddits* that discuss mental health issues. In addition, Reddit has a system that allows users to indicate their professional expertise (Reddit Flairs), which we use to identify a set of users with mental health professional background, identified as *MHPs* during our study. We obtained flairs from the r/psychotherapy subreddit,[1] a decision motivated by their reliability, as the moderators of this community allow comments and posts only by licensed therapy providers who may be asked to submit proof if concerns of falsely posing as a therapist arise.[2] Sample flair tags in this set are: Psychiatrist (sometimes accompanied by MD or DO), LPC (or Licensed Professional Counselor), LMFT (or Licensed Marriage and Family Therapist), PsyD (or Doctorate of Psychology).

We use an existing list of mental health subreddits from r/ListOfSubreddits[3] with additions from manual observations; all of the subreddits in our dataset with their number of comments are in Appendix B in Table 5. From these, we retrieve threads where an MHP submitted a direct reply. During this step, we also kept posts made by peers i.e., individuals who did not use any of the mental health care professional flairs. Our collection spans threads created between November 29, 2009 and December 21, 2020. Table 1 shows descriptive statistics for the final composition of the dataset, and Table 2 shows a sample interaction demonstrating the structure we use for our analysis. This study focuses on direct replies to the poster, thus we attempt to eliminate *megathreads* which tend not to focus in individual support-seekers by removing those above the 95th percentile in their number of direct replies; we leave analysis of deeper nested replies for future work.

Health Topics. To understand whether particular topics influence interactions with support-seekers, we group the subreddits into broader topics based

---

[1] Degree and license flair descriptions from r/psychotherapy wiki.

[2] See rule 2 and 9 in `https://www.reddit.com/r/psychotherapy/`, also listed in Appendix A.

[3] r/ListOfSubreddit's compilation of mental health and advice subreddits.

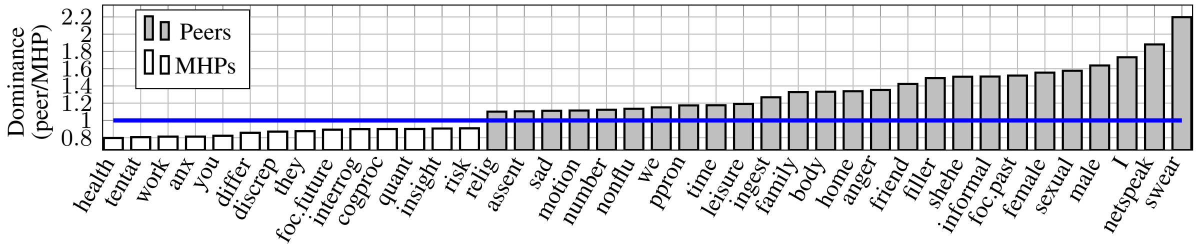

Figure 1: LIWC category dominance scores, computed as the relative use by peers divided by the relative use by MHPs, so that equal use is at $y = 1$ (blue line), higher dominance by peers at $y > 1$ (grey bars) and higher dominance by MHPs at $y < 1$ (white bars). Showing categories where frequency of use differs by at least 10%.

Key	Topic
Trauma	Trauma & Abuse
Anx	Psychosis & Anxiety
Compuls.	Compulsive Disorders
Cope	Coping & Therapy
Mood	Mood Disorders
Addict.	Addiction & Impulse Control
Body	Eating & Body
Neurodiv.	Neurodevelopmental Disorders
Health	General
Social	Broad Social

Table 3: Health condition and other subreddit topics. Keys are shortened names we use to refer to the topics.

on related health domains. We begin by following the categorization of subreddits by Sharma and De Choudhury (2018), who used the k-means clustering algorithm to generate initial clusters on the n-grams ($n = 3$) of the posts and manually refined the categories based the community descriptions in their subreddit home pages. Next, we adjust the categories and their associated subreddits based on the World Health Organization's ICD-10 classification system of mental and behavioural disorders.[4] The resulting topic categories are listed in Table 3 alongside shortened names which we use to refer to them. The full list of subreddits assigned to each topic are listed in Appendix B in Table 6.

4 Distinguishing MHPs and Peers

To begin our investigation into the linguistic behaviors of MHPs and peers, we test whether simple text classifiers are able to distinguish between comments authored by either MHPs or peers. We build three classifiers with different feature sets; the first are unigram counts for unigrams occurring at least five times, the second includes counts for the 73 word classes in the LIWC (Linguistic Inquire and Word Count) lexicon (Pennebaker et al., 2015), and the third encodes a subset of LIWC word classes associated with perspective shifts (i.e., *focusfuture, focuspast, focuspresent, I, ipron, negemo, posemo, ppron, pronoun, shehe, they, we,* and *you*) (Althoff et al., 2016); we elaborate on the psychological meaning behind these features in our analyses in the next section.

Due to the class imbalance between the peer and MHPs classes, we first downsampled the peer class to get a balanced distribution with the MHP class. This resulted in a set of 9,685 instances per class. We conduct our evaluations using ten-fold cross validation. Across these folds, the number of features ranges from 8,668 to 8,703. We use a Naive Bayes model, implemented with Sklearn's MultinomialNB module, [5] which outperformed a logistic regression model and an SVM in preliminary experiments.[6]

All models outperform a random baseline[7] with all LIWC features bringing the accuracy to 59.12%, LIWC perspective features to 59.14%, and unigram features to 70.80%. Overall, the classification results indicate language differences exist between the MHPs and peers. Motivated by this result, we proceed to several analyses to gain insights.

5 Linguistic and Dialogue Analysis

We analyze the linguistic behaviors of MHPs and peers responding to support-seeking posts, and their potential influence in eliciting further engagement with the support-seeker. Our analyses are inspired by psychology and computational studies that have shown that conversational behavioral aspects such as word usage, client engagement, and

---

[4] `https://www.who.int/substance_abuse/terminology/icd_10/en/`

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html`

[6] Runs in ~40 seconds per fold on one AMD Ryzen 7 3700U CPU.

[7] $p < 0.0001$ using a permutation test (Dror et al., 2018)

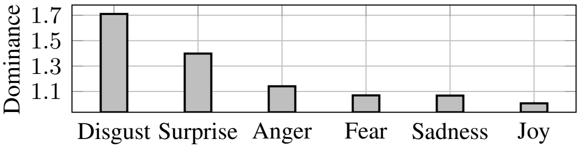

Figure 2: WordNet Affect usage (peers / MHPs)

language matching are positively related to successful counseling interactions (Gonzales et al., 2010; Althoff et al., 2016; Pérez-Rosas et al., 2018; Zhang et al., 2019).

5.1 Linguistic Ethnography

Numerous studies have demonstrated relationships between the dominant usage of certain word categories with individuals' psychological and physical health (Tausczik and Pennebaker, 2010; Weintraub, 1981; Rude et al., 2004). In alignment with these studies, we investigate the usage of such word categories using the LIWC and WordNet-Affect lexicons (Pennebaker et al., 2015; Strapparava and Valitutti, 2004).

For each group of users, we first compute the proportion of their words that fall in each category. Then, we compute the dominant use by dividing the proportion for peer users over the proportion for MHPs (Mihalcea and Pulman, 2009). Figure 1 shows LIWC categories where the rate of use differs by at least 10%, and results for WordNet Affect categories are shown in Figure 2.

Some observations such as the higher dominance of swear words (*swear*) and internet speak (*netspeak*) might be expected if professionals avoid such language. An interesting contrast in peers' language is the dominant use of first-person pronouns (*I, we*) and focus on the past (*focuspast*). In contrast, MHPs seem to use more non-first person pronouns (*you, they*) and focus on the future (*focusfuture*) instead. Peers' use of first person pronouns might arise when they share similar experiences with support-seekers. MHPs' use of second-person pronouns might suggest they are focusing on the support-seekers' experiences as a counselor would with a client in a counseling encounter. We also observe higher dominance of all WordNet Affect categories among peers, however the *joy* category (the most positive), is nearly equal with MHPs.

These observations of the peers' language are compelling because they align with existing theories linking depression to negative views of the future (i.e., *focuspast* and negative WordNet affects) (Pyszczynski et al., 1987) and self-focusing style (i.e., first-person pronouns) (Pyszczynski and Greenberg, 1987; Campbell and Pennebaker, 2003). Likewise, clients of SMS-based crisis counseling conversations were more likely to report feeling better after the encounter if they exhibited perspective shifts from these categories to their counterparts (i.e., toward *focusfuture*, non-first person pronouns, and positive sentiment) (Althoff et al., 2016).

Interestingly, the same study found clients were more likely to shift perspective when their counselors exhibited use of the counterpart categories first, suggesting that the counselors may play a key role in helping drive the perspective shift. Given those positive outcomes, observing the same dominant linguistic aspects among MHPs is encouraging and potentially signals a connection between how counselors apply conversational behaviors in practice and in online forum interactions. Future work can investigate the progression of dialogue between MHPs and support-seekers to find if support-seekers similarly exhibit the perspective shifts associated with the positive outcomes of the prior study, and likewise whether users of the forums also experience positive outcomes where this occurs.

5.2 Engaging Support-Seekers

To understand if linguistic behaviors are associated with prompting further engagement with the support-seeker, we compare the dominance of LIWC categories in comments receiving replies compared to comments that do not by dividing the usage rates of the former by the latter. Figure 3 shows these ratios for categories that differ by at least 5%. A compelling observation is the dominance of the categories *health*, *tentat*, and *you* in the MHP comments prompting poster-replies, and *you*, *focusfuture*, *interrog*, and *health* in the peer comments prompting poster-replies, as was exhibited among MHPs (see Figure 1 in Section 5.1); on the other end, the categories are more dominant in comments that do not engage a reply such as *I*, *we*, *death*, *friend*, *relig*, *swear*, were similarly represented as dominant categories in the peer group.

To gain further insight into these observations, we perform the following analysis: for each user group (peers and MHPs) we use a foreground corpus of their comments that were replied to by the support-seekers, and a background corpus of their comments that were not, and compute the dominance of LIWC categories of the foreground over

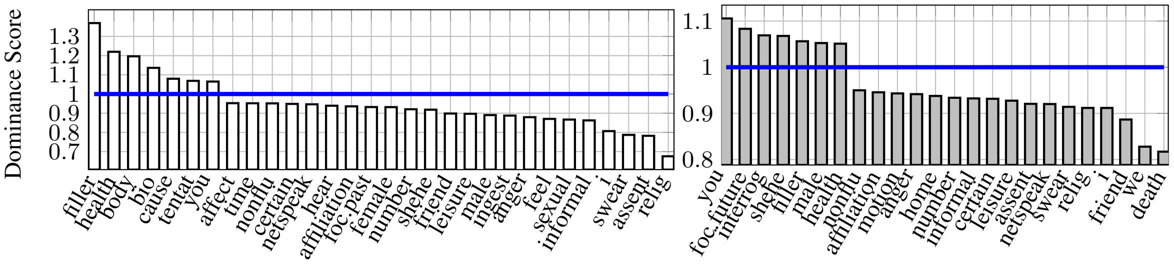

Figure 3: Dominance of LIWC categories, computed as the category relative frequencies among comments that **prompt support-seeker responses** divided by the relative frequencies among comments that do not, computed separately for MHPs (left) and peers (right).

`DRR` Group	OR Group	τ	p-value
Peer	MHP	.191	.017
MHP	MHP	.158	.048
Peer	Peer	-.031	.689
MHP	Peer	.008	.916

Table 4: Kendall τ's coefficient between the LIWC category dominance ranking in the replied comments (`DRR`) of the user group on the left and the overall ranking of LIWC category usage (`OR`) by the user group to the right.

the background as a ratio of their relative frequencies. We then rank the categories by highest to lowest dominance scores, and refer to this ranking by `DRR` (for **D**ominance **R**ank for **R**eplied comments). We compare the `DRR`s of each user group to the ranking of LIWC category usage among MHP users and among peer users separately (from Section 5.1) by computing the Kendall Tau's coefficient between them. A positive correlation would thus indicate that the more (or less) dominant categories among a group's replied comments are also more (or less) dominant among the other group overall. The correlation coefficients are shown in Table 4.

Interestingly, we observe a slight positive correlation between the `DRR`s for both MHPs and peers with the overall LIWC category usage ranking for MHPs. On the other hand, we see no correlations with the LIWC usage ranks for peers. Intuitively, it appears that for both MHPs and peers, the comments prompting further engagement with the poster appear to reflect the overall dominant linguistic aspects captured by LIWC of MHPs, but not peers. As counseling principles have emphasized the importance of mutual engagement between counselors and clients (Miller and Rollnick, 2012) and other work has shown that higher quality counseling sessions are associated with higher client engagement (Pérez-Rosas et al., 2018), it is compelling to observe associations between linguistic aspects of MHPs with the aspects associated with poster-engagement.

5.3 Linguistic Style Matching

Linguistic Style Matching (LSM) measures the extent to which one speaker matches another (Gonzales et al., 2010). It compares two parties' relative use of function words as these words are more indicative of style rather than content (Ireland and Pennebaker, 2010).

Previous studies in counseling conversations have measured LSM to understand the extent that counselors and clients match their language. Pérez-Rosas et al. (2019) showed higher LSM for high quality counseling sessions whereas Althoff et al. (2016) showed lower LSM for higher quality sessions. Pérez-Rosas et al. (2019) attributed this to the differences between the conversations they analyzed, theirs being synchronous face-to-face interactions while Althoff et al. (2016)'s was of asynchronous text messages, as well as differences in counseling styles.

We follow Nobles et al. (2020)'s approach leveraging Ireland and Pennebaker (2010)'s procedure to measure LSM between support seekers and support providers. For a text sequence, we compute the percentage of words that belong to each of nine function-word categories c from the LIWC lexicon, which include *auxiliary verbs, articles, common adverbs, personal/impersonal pronouns, prepositions, negations, conjunctions,* and *quantifiers*. Then, we compute the LSM of each word category c as shown in Equation 1 where p represents *post* and r represents the response. The composite LSM score for p and r is the mean of all category LSM scores. For each thread, we separate the MHP and peer replies, and take the mean of all

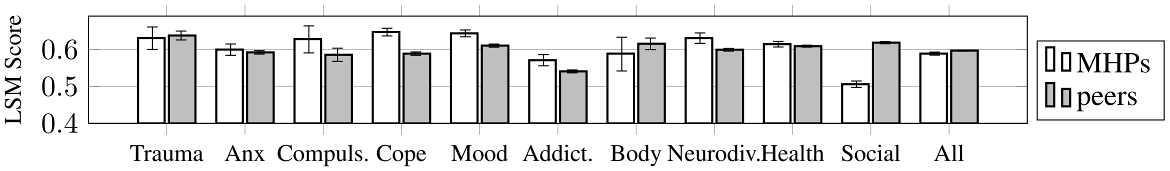

Figure 4: LSM scores with 95% confidence intervals calculated with non-parametric bootstrap resampling.

composite LSM scores.

$$LSM_c = 1 - \frac{abs(cat\%_p - cat\%_r)}{cat\%_p + cat\%_r + .0001} \quad (1)$$

We compute these LSM scores over all data together as well as separately for each subreddit topic (named in Table 3). The resulting scores are shown in Figure 4.

We observe LSM scores vary by topic, and most are similar for peers and MHPs or have overlapping confidence intervals. Compared to their LSMs in other topics, MHPs score lower in SOCIAL, which covers broad social issues that are less specialized to health conditions than the others. However, peers have high LSMs in SOCIAL relative to most other topics, and notably higher LSMs than the MHPs. Additionally, MHPs have higher LSMs than those of peers and relative to their own in communities that cover topics of specific compulsive, mood, and neurodevelopmental disorders (COMPULS., MOOD, and NEURODIV.), communities that orient toward counseling purposes (COPE), or toward advice-seeking communities for health and social concerns (HEALTH). The influences in these results require further investigation, but a possible explanation could be that expert knowledge and experience may offer more benefit to specialized condition-related issues than to broader social issues.

6 Language Modeling

We further examine differences in word usage by building separate language models for MHPs and peers. We seek to identify language use that is indicative of one group or another by running the language model of one on the data of the other and analyzing words with high perplexity. To run these experiments, we use the language model of Merity et al. (2018a,b), which is a recent LSTM-based language model that achieved state-of-the-art performance by combining several regularization techniques.[8]

Our implementation uses a fixed vocabulary of 20,907 tokens for both the peer and MHP language models. This is determined by a minimum count of five across the set of posts from both groups. Each language model is trained for 50 epochs.[9]

We use the language model trained on MHP data to find words with high entropy in peer data and vice versa. Since we are concerned with the *difference* in predictability of words between the MHP and peer language models, we subtract the entropy given by the model trained on that data from the entropy assigned by the model that was not trained on that data. In other words, to find words difficult to predict in B's data, we subtract each word's entropy calculated by the model trained on B from the entropy calculated by the model trained on A as follows, for a set of words, X:

$$E_{A,B} = -\frac{1}{|X|} \sum_{x \in X} log(p_A(x)) - log(p_B(x)) \quad (2)$$

If we calculate the entropy difference for each LIWC category and for each assignment of the MHP and peer groups to A and B, we find the highest differences for each category shown in the first and third plots of Figure 5. We find highest entropy scores for words relating to *leisure*, *sex*, and *numbers* when running the MHP language model on peer data. Likewise, when running the peer model on MHP data, the category of *discrepancy* contains words whose accuracy is improved the least by the peer model, again showing that these words are more indicative of the MHP group.

We perform a similar analysis, creating a language model for posts which have the highest score (or tied for highest) and another model for all other

---

[8] `https://github.com/salesforce/awd-lstm-lm`

[9] Validation set perplexities for expert and score groups: peer on peer: 44, peer on MHP: 52, MHP on peer: 91, MHP on MHP: 74, low on high: 39, low on low: 43, high on high: 50, high on low: 57. The difference in perplexity is due to the difference in volume of posts between groups. Runs in $\sim$2 min per epoch on a GeForce RTX 2080 Ti GPU.

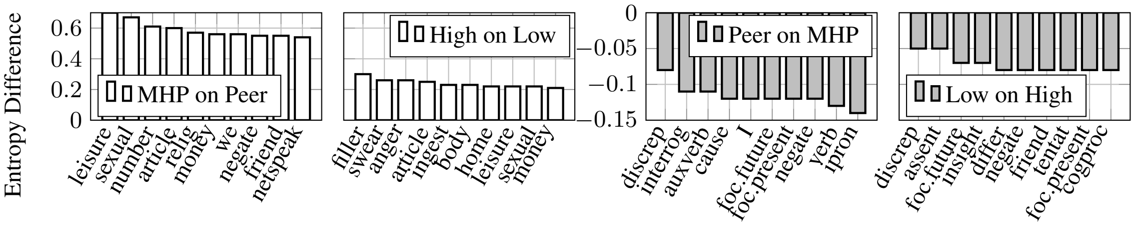

Figure 5: Entropy differences for LIWC word categories when running both language models on one group's data. High entropy scores on one dataset indicate word types that are harder for the opposite group's model to predict.

posts. We measure entropy differences and show the highest scoring categories for each group in the second and fourth plots. Some of the categories indicative of MHP language are also indicative of higher scoring posts; *discrepancy*, *present* and *future* words, and *negation* words, while other categories like *assent* and *insight* words are more dominant in higher scoring posts. The lower scoring posts have the highest entropy differences for some types of words in the peer data, however, we also see that *filler*, *anger*, and *swear* words had the highest entropy differences for the low scoring group. Qualitative example sentences with word-level entropy and LIWC annotations are shown in the appendix in Table 7.

7 Discussion and Future Work

In comparing linguistic aspects of MHP and peer comments, we find MHP tendencies align with established counseling principles and findings in counselor behaviors from recent literature. In particular, they align in the use of words that increase the likelihood of desired perspective shifts associated with clients feeling better after text counseling sessions (RQ2) (Althoff et al., 2016). We also found unique differences in the behavior of MHPs as compared to peers in how they respond to information seekers (RQ1). Although, comments by peers that prompt support-seeker replies also make use of similar word categories to MHPs, which shows that comparing MHPs to peers can offer insight into peer interactions as well.

It is important to note that our analyses rely heavily on the LIWC lexicon. While LIWC and other lexicons can help uncover variational language across groups at an exploratory stage, their use alone does not explain why variations are present. Certain limitations of LIWC are clear, such as when certain words that occur in multiple categories misleadingly boost the prominence of the categories equally. Kross et al. (2019)'s and Jaidka et al. (2020)'s studies have also demonstrated limitations of the use of LIWC when working with word counts to correlate with well-being metrics and an individual's emotional state. We utilize LIWC to understand linguistic behavior differences in conversations with peers and MHPs rather than to evaluate the emotional or mental health state of individuals; however, it is important to consider how these limitations could pertain to our interpretations of their differences, especially as we explore them more deeply in future work. In our study, we explore the patterns we find in the context of previous findings from related literature such as (Althoff et al., 2016) and (Nobles et al., 2020), however it warrants another study into nuanced aspects of the patterns to infer their social functions in support seeking forums in particular.

Although our findings align MHP behaviors with certain counselor behaviors associated with positive outcomes, our analyses do not support claims that MHP behaviors are more beneficial to individuals seeking support; rather, we have shown that the general tendencies of MHPs are in accordance with principles and behaviors demonstrated by counselors in other settings. Understanding the outcomes of these interactions for individual support seekers remains as an area for future work, which could employ surveying methods from prior work to measure perceived empathy in online communities (Nambisan, 2011). Our dataset also enables investigations into whether support-seekers exhibit perspective shifts in interacting with MHPs or peers, and what MHP and peer tendencies are associated with these perspective shifts.

Another direction for future work could focus on modeling social media-specific engagement patterns of MHP and peer interactions. Prior work developed a model that accounts for variables indicating the level of attention threads receive (i.e., thread lengths and number of unique commenters), and variables indicating the degree of interaction between posters and commenters (i.e., time between responses and whether the poster replies to commenters), and used this model to study peer-to-peer interactions in online mental health platforms (Sharma et al., 2020); this approach may enable studying supportive interactions in megathreads and threads involving back-and-forth dialog between two or more parties.

More questions arise if we consider MHP tenure and specific domain of expertise (e.g., specializations, licenses, academic degrees). Prior work that studied longitudinal changes in counselor linguistic behaviors indicated that systematic changes occur over time as counselors develop personal styles that are more distinct from other counselors and exhibit more diversity across interactions (Zhang et al., 2019). Future work could model the language longitudinally for MHPs and peers that have longer-term histories of participating in mental health forums to investigate whether systematic changes occur online as well, and if so, whether they reflect similar changes found in prior work.

8 Limitations and Ethical Considerations

A number of unknowns exist in what we are able to extract from Reddit. For instance, we do not know if users that do not use flairs are mental health professionals. We assume that those who have used the MHP flairs are MHPs and those that have not used them are peers. Additionally, we have grouped all MHP flairs into one group for our analysis, though a more nuanced analysis based on particular professional roles (e.g., psychologists, psychiatrists, social workers) and specializations (e.g., motivational interviewing, cognitive behavioral therapy, family & marriage counseling) may reveal additional trends. Prior work found that disclosing credentials has impacts on engagements that vary by subreddit and linguistic patterns associated with different experience levels and expert domains (Park et al., 2020), thus the effects of disclosing MHP credentials when responding to support-seekers should be investigated.

A classifier or language model used to distinguish between MHPs and peers or to generate the language of either could have negative implications. A generative model that provides feedback to users could generate language that is harmful for those seeking help. Our work could be used to devise a tool to train counselors, however we do not have a direct measure of what type of responses are helpful or meaningful. In such an application, there is potential to reinforce harmful behaviors due to the inaccuracy of our models. Future studies are needed to determine how to best design a tool to train counselors and how models derived from corpora such as ours correspond to advice that patients find useful.

9 Conclusion

As the role of social networks is becoming more critical in how people seek health-information, it is important to understand their broader implications to health communication and how health experts can engage to promote the soundest information and offer support to their vulnerable users. By elucidating techniques employed by mental health professionals in their interactions with support-seekers in mental health forums, we have contributed insights toward the broader research direction of understanding how health experts currently engage with these platforms. With evidence that MHP linguistic behaviors associate with further engagement with support-seekers and that these same behaviors are associated with positive counseling conversation outcomes, we have shown that analyzing MHP behavior is a promising direction for better understanding online interaction outcomes, which can further inform forum design and moderation, and expert health promotion efforts.

The code used for our experiments and analyses, and the post ids in our dataset can be accessed at `https://github.com/MichiganNLP/MHP-and-Peers-Reddit`.

Acknowledgments

This material is based in part upon work supported by the Precision Health initiative at the University of Michigan, by the National Science Foundation (grant #1815291), and by the John Templeton Foundation (grant #61156). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Precision Health initiative, the National Science Foundation, or John Templeton Foundation.

References

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale analysis of counseling conversations: An application of natural language processing to mental health. *Transactions of the Association for Computational Linguistics*, 4:463–476.

Silvio Amir, Mark Dredze, and John W Ayers. 2019. Mental health surveillance over social media with digital cohorts. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*, pages 114–120.

Antoaneta Baltadzhieva and Grzegorz Chrupała. 2015. Predicting the quality of questions on Stackoverflow. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 32–40, Hissar, Bulgaria. INCOMA Ltd. Shoumen, Bulgaria.

Julie Campbell, Cecilia Aragon, Katie Davis, Sarah Evans, Abigail Evans, and David Randall. 2016. Thousands of positive reviews: Distributed mentoring in online fan communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, page 691–704, New York, NY, USA. Association for Computing Machinery.

R Sherlock Campbell and James W Pennebaker. 2003. The secret life of pronouns: Flexibility in writing style and physical health. *Psychological science*, 14(1):60–65.

Wen-Ying Sylvia Chou, Yvonne M Hunt, Ellen B Beckjord, Richard P Moser, and Bradford W Hesse. 2009. Social media use in the united states: implications for health communication. *Journal of medical Internet research*, 11(4):e48.

Munmun De Choudhury and Sushovan De. 2014. Mental health discourse on reddit: Self-disclosure, social support, and anonymity. In *Proceedings of the International AAAI Conference on Web and Social Media*.

Martí Domínguez and Lucía Sapiña. 2015. Pediatric cancer and the internet: exploring the gap in doctor-parents communication. *Journal of Cancer Education*, 30(1):145–151.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Elizabeth A Gage-Bouchard, Susan LaValley, Molli Warunek, Lynda Kwon Beaupin, and Michelle Mollica. 2018. Is cancer information exchanged on social media scientifically accurate? *Journal of cancer Education*, 33(6):1328–1332.

Amy L Gonzales, Jeffrey T Hancock, and James W Pennebaker. 2010. Language style matching as a predictor of social dynamics in small groups. *Communication Research*, 37(1):3–19.

Frances Griffiths, Jonathan Cave, Felicity Boardman, Justin Ren, Teresa Pawlikowska, Robin Ball, Aileen Clarke, and Alan Cohen. 2012. Social networks–the future for health care delivery. *Social science & medicine*, 75(12):2233–2241.

Sara D Hodges, Kristi J Kiel, Adam DI Kramer, Darya Veach, and B Renee Villanueva. 2010. Giving birth to empathy: The effects of similar experience on empathic accuracy, empathic concern, and perceived empathy. *Personality and Social Psychology Bulletin*, 36(3):398–409.

Molly E Ireland and James W Pennebaker. 2010. Language style matching in writing: Synchrony in essays, correspondence, and poetry. *Journal of personality and social psychology*, 99(3):549.

Kokil Jaidka, Salvatore Giorgi, H Andrew Schwartz, Margaret L Kern, Lyle H Ungar, and Johannes C Eichstaedt. 2020. Estimating geographic subjective well-being from twitter: A comparison of dictionary and data-driven language methods. *Proceedings of the National Academy of Sciences*, 117(19):10165–10171.

Ethan Kross, Philippe Verduyn, Margaret Boyer, Brittany Drake, Izzy Gainsburg, Brian Vickers, Oscar Ybarra, and John Jonides. 2019. Does counting emotion words on online social networks provide a window into people's subjective experience of emotion? a case study on facebook. *Emotion*, 19(1):97.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018a. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018b. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Rada Mihalcea and Stephen Pulman. 2009. Linguistic ethnography: Identifying dominant word classes in text. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 594–602. Springer.

William R Miller and Stephen Rollnick. 2012. *Motivational interviewing: Helping people change*. Guilford press.

Priya Nambisan. 2011. Information seeking and social support in online health communities: impact on patients' perceived empathy. *Journal of the American Medical Informatics Association*, 18(3):298–304.

Alicia Nobles, Caitlin Dreisbach, Jessica Keim-Malpass, and Laura Barnes. 2018. " is this an std? please help!": Online information seeking for sexually transmitted diseases on reddit. In *Proceedings of the International AAAI Conference on Web and Social Media*.

Alicia L Nobles, Eric C Leas, Benjamin M Althouse, Mark Dredze, Christopher A Longhurst, Davey M Smith, and John W Ayers. 2019. Requests for diagnoses of sexually transmitted diseases on a social media platform. *Jama*, 322(17):1712–1713.

Alicia L Nobles, Eric C Leas, Mark Dredze, and John W Ayers. 2020. Examining peer-to-peer and patient-provider interactions on a social media community facilitating ask the doctor services. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 464–475.

Albert Park and Mike Conway. 2017. Tracking health related discussions on reddit for public health applications. In *AMIA Annual Symposium Proceedings*, volume 2017, page 1362. American Medical Informatics Association.

Kunwoo Park, Haewoon Kwak, Hyunho Song, and Meeyoung Cha. 2020. "trust me, i have a ph. d.": A propensity score analysis on the halo effect of disclosing one's offline social status in online communities. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 534–544.

James W Pennebaker, Roger J Booth, Ryan L Boyd, and Martha E Francis. 2015. Linguistic inquiry and word count: Liwc2015.

Verónica Pérez-Rosas, Xuetong Sun, Christy Li, Yuchen Wang, Kenneth Resnicow, and Rada Mihalcea. 2018. Analyzing the quality of counseling conversations: the tell-tale signs of high-quality counseling. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Verónica Pérez-Rosas, Xinyi Wu, Kenneth Resnicow, and Rada Mihalcea. 2019. What makes a good counselor? learning to distinguish between high-quality and low-quality counseling conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 926–935.

Gregory A Poland, Robert M Jacobson, et al. 2011. The age-old struggle against the antivaccinationists. *N Engl J Med*, 364(2):97–9.

T. B. Procaci, S. W. M. Siqueira, B. P. Nunes, and T. Nurmikko-Fuller. 2017. Modelling experts behaviour in q a communities to predict worthy discussions. In *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, pages 291–295.

Tom Pyszczynski and Jeff Greenberg. 1987. Self-regulatory perseveration and the depressive self-focusing style: a self-awareness theory of reactive depression. *Psychological bulletin*, 102(1):122.

Tom Pyszczynski, Kathleen Holt, and Jeff Greenberg. 1987. Depression, self-focused attention, and expectancies for positive and negative future life events for self and others. *Journal of personality and social psychology*, 52(5):994.

Rachael A Record, Will R Silberman, Joshua E Santiago, and Taewook Ham. 2018. I sought it, i reddit: Examining health information engagement behaviors among reddit users. *Journal of Health Communication*, 23(5):470–476.

Howard Rheingold. 1993. *The virtual community: Homesteading on the electronic frontier*, volume 32. Addison-Wesley Reading, MA.

Stephanie S Rude, Eva-Maria Gortner, and James W Pennebaker. 2004. Language use of depressed and depression-vulnerable college students. *Cognition and Emotion*, 18(8):1121–1133.

Ashish Sharma, Monojit Choudhury, Tim Althoff, and Amit Sharma. 2020. Engagement patterns of peer-to-peer interactions on mental health platforms. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 614–625.

Eva Sharma and Munmun De Choudhury. 2018. Mental health support and its relationship to linguistic accommodation in online communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–13, New York, NY, USA. Association for Computing Machinery.

Carlo Strapparava and Alessandro Valitutti. 2004. Wordnet-affect: an affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1083–1086.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Maria Tomprou, Laura Dabbish, Robert E. Kraut, and Fannie Liu. 2019. Career mentoring in online communities: Seeking and receiving advice from an online community. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA. Association for Computing Machinery.

Sheng Wang, Yanen Li, Duncan Ferguson, and Chengxiang Zhai. 2014. Sideeffectptm: An unsupervised topic model to mine adverse drug reactions from health forums. In *Proceedings of the 5th ACM conference on bioinformatics, computational biology, and health informatics*, pages 321–330.

Yi-Chia Wang, Robert E Kraut, and John M Levine. 2015. Eliciting and receiving online support: using computer-aided content analysis to examine the dynamics of online social support. *Journal of medical Internet research*, 17(4):e99.

Zijian Wang and David Jurgens. 2018. It's going to be okay: Measuring access to support in online communities. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 33–45.

Walter Weintraub. 1981. *Verbal behavior: Adaptation and psychopathology*. Springer Publishing Company New York.

Jennifer L Welbourne, Anita L Blanchard, and Marla D Boughton. 2009. Supportive communication, sense of virtual community and health outcomes in online infertility groups. In *Proceedings of the fourth international conference on Communities and technologies*, pages 31–40.

Justine Zhang, Robert Filbin, Christine Morrison, Jaclyn Weiser, and Cristian Danescu-Niculescu-Mizil. 2019. Finding your voice: The linguistic development of mental health counselors. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 936–947.

Naitian Zhou and David Jurgens. 2020. Condolences and empathy in online communities. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 609–626.

Appendix

A Flairs

Rules regarding flair credibility from r/psychotherapy:

“**2. Only posts and comments from those providing therapy in a licensed capacity allowed.** No comments/posts from anyone who is not providing therapy in a licensed capacity. This includes students who are not yet practicing therapy (e.g., undergraduate or graduate students who haven’t had their first practica experience) or if you have left the field for another field, this is not the place for you to post/comment. There is an exception to this rule for posting in our Career and Education Megathread. Accurate user flair is required for all posts, and strongly encouraged for comments.”

“**9. Falsely posing as a therapist** If you post in this subreddit, the assumption is made that you are a therapist. Users that falsely post as if they were a therapist will be permanently banned. Claiming that you didn’t say you were a therapist is not an argument against this rule. Users may be asked to submit proof of their status as a practicing therapist to appeal a ban.”

B Data

We used the PushShift API for the first pass of obtaining mental health posts and comments, and the MHP flairs. After extracting the IDs of posts where MHPs commented, we obtained the fully structured comment sections using open sourced code from `https://github.com/saucecode/reddit-thread-ripper`. The numbers of posts in our dataset for each subreddit are shown in Table 5.

C Other

Sample sentences illustrating relative entropies of words predicted by the peer language model on MHP data (top) and the MHP language model on peer data (bottom) are shown in Table 7.

AskDocs	21025	relationship_advice	16061	stopdrinking	10170
ADHD	9093	offmychest	5076	mentalhealth	4486
socialskills	4113	BPD	3570	depression	3235
Anxiety	2956	aspergers	2703	Advice	2493
askatherapist	2120	PCOS	1819	alcoholicsanonymous	1498
leaves	1452	SuicideWatch	1092	REDDITORSINRECOVERY	977
needadvice	892	ptsd	704	NoFap	465
OCD	411	socialanxiety	400	BipolarReddit	355
GetMotivated	354	alcoholism	350	cripplingalcoholism	335
emetophobia	297	bulimia	249	mentalillness	246
nosurf	224	EOOD	208	depression_help	193
EatingDisorders	170	schizophrenia	167	MMFB	159
AlAnon	139	disability	127	fuckeatingdisorders	119
Antipsychiatry	116	MadOver30	114	quittingkratom	114
addiction	111	GFD	109	CompulsiveSkinPicking	108
Needafriend	106	dbtselfhelp	99	rapecounseling	93
stopsmoking	89	selfhelp	87	ForeverAlone	81
getting_over_it	72	BodyAcceptance	54	Anger	50
traumatoolbox	50	selfharm	47	TwoXADHD	40
survivorsofabuse	40	dpdr	38	rape	36
Tourettes	34	HealthAnxiety	26	schizoaffective	25
Anxietyhelp	25	eating_disorders	20	domesticviolence	17
neurodiversity	13	helpmecope	12	StopSelfHarm	12
sad	11	AtheistTwelveSteppers	10	Trichsters	6
MenGetRapedToo	5	ARFID	5	whatsbotheringyou	3
DysmorphicDisorder	1	OCPD	1		

Table 5: The number of comments in each subreddit of our dataset.

Category	Subreddits
Trauma & Abuse (Trauma)	r/Anger, r/survivorsofabuse, r/domesticviolence, r/ptsd, r/rapecounseling, r/selfharm, r/StopSelfHarm, r/traumatoolbox, r/rape, r/MenGetRapedToo
Psychosis & Anxiety (Anx)	r/Anxiety, r/socialanxiety, r/Anxietyhelp, r/HealthAnxiety, r/BPD, r/dpdr, r/schizophrenia, r/schizoaffective, r/emetophobia
Compulsive Disorders (Compuls.)	r/CompulsiveSkinPicking, r/OCD, r/Trichsters, r/DysmorphicDisorder, r/OCPD
Coping & Therapy (Cope)	r/getting_over_it, r/helpmecope, r/offmychest, r/MMFB, r/askatherapist, r/EOOD, r/dbtselfhelp, r/AlAnon, r/REDDITORSINRECOVERY, r/GetMotivated, r/Antipsychiatry, r/selfhelp
Mood Disorders (Mood)	r/depression, r/depression_help, r/ForeverAlone, r/GFD, r/mentalhealth, r/SuicideWatch, r/sad, r/BipolarReddit
Addiction & Impulse Control (Addict.)	r/stopdrinking, r/addiction, r/stopsmoking, r/leaves, r/alcoholism, r/cripplingalcoholism, r/quittingkratom, r/alcoholicsanonymous, r/NoFap
Eating & Body (Body)	r/eating_disorders, r/EatingDisorders, r/ARFID, r/fuckeatingdisorders, r/BodyAcceptance, r/bulimia
Neurodevelopmental Disorders (Neurodiv.)	r/ADHD, r/aspergers, r/TwoXADHD
General (Health)	r/AskDocs, r/needadvice, r/Advice, r/mentalillness, r/neurodiversity, r/whatsbotheringyou, r/MadOver30
Broad Social (Social)	r/socialskills, r/relationship_advice, r/nosurf, r/Needafriend, r/AtheistTwelveSteppers, r/PCOS, r/disability, r/Tourettes
Overall	All

Table 6: Subreddit categories.

	MHP Data
DISCREP	the problem with psychiatric research is the relative subjectivity of it , much less glamorous outcomes , and the lack of public interest despite its burden on society .
INTERROG	who diagnosed you with spinal issues - which might show how anxiety affects your physical health .
AUXVERB	that being said , many other mental health concerns have overlapping symptoms with adhd including anxiety and depression .
	Peer Data
LEISURE	it ’s like jogging with a back back full of bricks .
SEXUAL	it seems like everybody here is desired sexually so that must mean you ’re doing something right .
NUMBER	when it came time for homework she set them up with three things to do (one being homework) and had them switch every fifteen minutes .

Table 7: Sample sentences from MHP data with relative entropy marked by highlight color (i.e. darker blue means higher entropy relative to other words in the sentence). All words in the given LIWC category are marked with a rounded rectangle.

An Investigation of Suitability of Pre-Trained Language Models for Dialogue Generation – Avoiding Discrepancies

Yan Zeng
DIRO, Université de Montréal
yan.zeng@umontreal.ca

Jian-Yun Nie
DIRO, Université de Montréal
nie@iro.umontreal.ca

Abstract

Pre-trained language models have been widely used in response generation for open-domain dialogue. These approaches are built within 4 frameworks: Transformer-ED, Transformer-Dec, Transformer-MLM and Transformer-AR. In this study, we experimentally compare them using both large and small-scale data. This reveals that decoder-only architecture is better than stacked encoder-decoder, and both left-to-right and bi-directional attention have their own advantages. We further define two concepts of model discrepancy, which provides a new explanation to the model performance. As discrepancies may hinder performance, we propose two solutions to reduce them, which successfully improve the model performance.

1 Introduction

It has been shown (Wolf et al., 2019) that leveraging a pre-trained Language Model (LM) based on transformer can achieve excellent performance for dialogue generation. Different approaches have been proposed recently, which can be categorized into 4 frameworks: Transformer-ED(Zheng et al., 2019), an encoder-decoder Transformer, Transformer-Dec (Wolf et al., 2019; Lin et al., 2020), Transformer-MLM (Dong et al., 2019) and Transformer-AR (Bao et al., 2019; Shuster et al., 2019). The latter three all utilize a decoder-only architecture. Besides, Trans-Dec uses left-to-right attention for both source and target side, while Trans-MLM and Trans-AR employ bi-directional attention on the source side to encode dialogue history. Due to this difference, Trans-Dec only utilizes left-to-right pre-trained models, e.g. GPT-2 (Radford et al., 2019), while Trans-MLM/AR are based on the pre-trained models applying bi-directional attention (on the source side), e.g. BERT (Devlin et al., 2018). The difference between Trans-MLM and Trans-AR is that Trans-MLM uses masked language modeling while Trans-AR uses auto-regressive objective.

Recent studies have explored pre-training dialogue models using large-scale Reddit/Twitter data (Adiwardana et al., 2020; Roller et al., 2020). It is then straightforward to fine-tune the models for a specific dialogue task. However, in practice, there may not always be enough data for pre-training. In some cases, we still need to exploit a pre-trained LM. For example, some studies do further pre-training for dialogue based on a pre-trained LM (Zhang et al., 2019; Dong et al., 2019; Bao et al., 2019; Shuster et al., 2019), and some studies that do multi-task learning (e.g. on dialogue and question answering) can only fine-tune based on a pre-trained LM (Lin et al., 2020; Zeng and Nie, 2021). Then, a critical question is how to best exploit a pre-trained LM for dialogue generation. On this question, we have contradictory beliefs in the literature: some researchers believe that Trans-Dec is appropriate because it uses a left-to-right language model that corresponds well to the dialogue generation task (Zhang et al., 2019; Lin et al., 2020), while some others (Dong et al., 2019; Bao et al., 2019) show that Trans-MLM/AR fine-tuning BERT can also achieve state-of-the-art performance.

In this study, we aim to address the above question. To do it, we first compare the 4 frameworks with the same setting on 3 datasets, each with large and small scale training data. Our results on large-scale datasets show that Trans-ED that applies the stacked encoder-decoder architecture does not produce competitive results against the others that use a decoder-only architecture. Trans-Dec/AR generate the most appropriate responses. However, according to automatic metrics, Trans-Dec generates most diverse responses while Trans-AR produce responses most similar to the ground-truth. This may be due to the fact that uni-directional attention does not have constraint from the right side context and

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4481–4494
August 1–6, 2021.

thus is more flexible, while bi-directional attention on source side can better model dialogue context. In contrast, the results on small-scale datasets reveal an important aspect, namely, the discrepancies that may occur between the pre-training and the fine-tuning processes. We then try to explain the performances of the 4 frameworks with respect to the discrepancies.

The concept of model discrepancy has been briefly mentioned in Yang et al. (2019) to mean that the model has been trained in a way, but used in a different way. However, the problem has not been investigated in depth. In this work, we go further in this direction and define two discrepancies: **pretrain-finetune discrepancy** which means the differences in architecture and loss function between pre-training and fine-tuning, and **finetune-generation discrepancy** which means that the way the model is used in generation (inference/test) is different from the way it has been trained. For the 4 frameworks, except Trans-Dec, they all have some pretrain-finetune discrepancies. For example, Trans-AR relies on BERT pre-trained using bidirectional attention, but has to limit it to left-to-right attention on the target side during fine-tuning. Only Trans-MLM has finetune-generation discrepancy because of MLM objective: during training, the model input has random masks, while in the generation process, the input does not contain masks.

Discrepancies might affect the model performance since models with such discrepancies cannot best exploit the pre-trained model or employ the fine-tuned model. Our experiments on small-scale datasets show that the performance of Trans-AR that have larger pretrain-finetune discrepancy drops more sharply than Trans-MLM. Trans-Dec/MLM that have small pretrain-finetune discrepancy have clear advantage over other frameworks according to human evaluation. It becomes clear that discrepancies hinder the performance of a dialogue model. To alleviate the problems, we propose 2 approaches to respectively reduce pretrain-finetune and finetune-generation discrepancies of Trans-MLM, aiming at improving its performance. Our experiments show that both methods bring some improvement. In particular, by eliminating finetune-generation discrepancy of Trans-MLM, our approach significantly outperforms previous methods in most automatic metrics, and achieves comparable performance to Trans-Dec in human evaluation that uses much larger dataset for pre-training. These results confirm that discrepancies are indeed an important factor that influences the effectiveness of leveraging a pre-trained LM for a sequence-to-sequence task, and should be alleviated.

The contributions in this work are as follows:

- We compare the four commonly used frameworks that utilize pre-trained language models for open-domain dialogue generation on 3 public datasets each in large and small scale. and we analyze each framework based on the experimental results.
- We introduce the concept of pretrain-finetune discrepancy and finetune-generation discrepancy, and we examine the discrepancies of each framework.
- We propose two methods to reduce discrepancies[1], yielding improved performance. It is the first investigation that shows explicitly the phenomenon of model discrepancy and its impact on performance.

2 Pre-training Based Frameworks

We start with a brief description of the 4 frameworks for dialogue generation based on pre-trained models. More details are provided in Appendix A. We examine the pretrain-finetune discrepancy of each framework. Figure 1 and Table 1 provide an overview.

2.1 Trans-ED

Trans-ED discussed in this paper is an encoder-decoder architecture used by ConvAI2 (Dinan et al., 2019) champion [2]. The decoder of Trans-ED is stacked upon the encoder outputs, while in other decoder-only frameworks, all hidden states of the source side are utilized in the decoding part. The framework shares the encoder and the decoder and initializes the parameters with GPT (Radford et al., 2018). In this case, the pretrain-finetune discrepancy comes from the bi-directional attention in the encoder since GPT is a left-to-right language model. This framework is not commonly used for fine-tuning on a dialogue task. In practice, more efficient variants of Trans-ED are recently used for extremely large-scale dialogue pre-training from

---

[1] The code is available at: `https://github.com/zengyan-97/Transformer-MLM-DiffFree`

[2] `https://github.com/atselousov/transformer_chatbot`

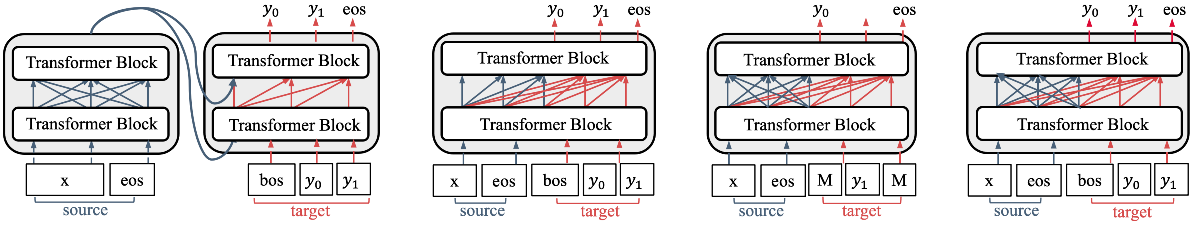

(a) Transformer-ED (b) Transformer-Dec (c) Transformer-MLM (d) Transformer-AR

Figure 1: Architectures of 4 pre-training based Transformers for dialogue generation.

	Trans-ED	**Trans-Dec**	**Trans-MLM**	**Trans-AR**
Pre-trained LM	GPT	GPT-2	BERT	BERT
Architecture	encoder-decoder	decoder-only	decoder-only	decoder-only
Source Side Attn.	bi-directional	left-to-right	bi-directional	bi-directional
Target Side Attn.	left-to-right	left-to-right	left-to-right	left-to-right
Objective	auto-regressive	auto-regressive	MLM	auto-regressive

Table 1: Key characteristics of the 4 pre-training based Transformers. Characteristics in red are inconsistent between pre-training and fine-tuning.

scratch. For example, Adiwardana et al. (2020) utilizes Evolved Transformer to prune redundant connections, and Roller et al. (2020) employs only 2 encoder layers and 24 decoder layers of standard Transformer (Vaswani et al., 2017).

2.2 Trans-Dec

Trans-Dec is a left-to-right decoder-only architecture, and it utilizes GPT-2 (Radford et al., 2019). Thus, there is no pretrain-finetune discrepancy in terms of architecture and loss function. This framework is widely applied for fine-tuning on a dialogue task. However, it encodes dialogue history using only left-to-right attention, which limits the scope of context, resulting in a partial context modeling.

2.3 Trans-MLM and AR

These two frameworks have an identical decoder-only architecture that employs different self-attention masks for the source and target side: they use bi-directional attention on the source side to encode dialogue history and left-to-right attention on the target side. The only difference between them is the objective function: Trans-MLM masks some tokens at the target side and tries to predict them, while Trans-AR uses auto-regressive objective that tries to predict the next tokens successively. BERT is often exploited by the two frameworks, which is a bi-directional architecture using MLM as the pre-training objective. Thus, the pretrain-finetune discrepancy of Trans-MLM/AR comes from the left-to-right attention on the target side. Additionally, Trans-AR applies the auto-regressive objective, which is different from the MLM used in the pre-training.

2.4 Applications of the Frameworks

The four frameworks we described have been widely applied to dialogue generation. For personalized response generation, Wolf et al. (2019) uses Trans-Dec and Zheng et al. (2019) utilizes Trans-ED. Lin et al. (2019) uses Trans-Dec for empathetic response generation. Zeng and Nie (2021) proposes a multi-task learning approach based on Trans-MLM for conditioned dialogue generation. Meanwhile, some studies propose to further pre-train the model using large-scale dialogue data based on a pre-trained language model: Zhang et al. (2019) trains Trans-Dec on 147M Reddit data based on GPT-2, Dong et al. (2019) trains Trans-MLM on natural language understanding and generation datasets based on BERT, Shuster et al. (2019) trains Trans-AR on large-scale Reddit data and then jointly trains on 12 dialogue sub-tasks based on BERT, and Bao et al. (2019) trains a variant of Trans-AR on large-scale Reddit and Twitter data based on BERT. Some recent studies have increased the model size to billions of parameters and utilize even more training data, e.g. Reddit, to train a conversational model from scratch (Adiwardana et al., 2020; Roller et al., 2020; Bao et al., 2020b).

In general, these studies show that all the 4 frameworks can produce good results, and increasing the model size and training data is an effective method to further improve performance. However, behind the success story, the question of suitability

	Twitter	Ubuntu	Reddit
Train Set	2M	1.5M	3M
Valid Set	60K	30K	80K
Test Set	20K	20K	20K

Table 2: Key characteristics of the three public datasets. For each dataset, we also evaluate model performance using **100K** training data and the same test set.

of a framework is masked. To investigate this question, we do not follow the current trend to increase the model size and training data. Instead, we are interested in the behaviors of different frameworks on the same datasets and to understand the reasons.

3 Experiments

3.1 Datasets

We use all the three large-scale unlabeled dialogue datasets in Shuster et al. (2019). Some important characteristics of the datasets are summarized in Table 2. We are interested in the behaviors of the models in two cases: 1) further pre-training on large dialogue data based on a pre-trained LM; and 2) fine-tuning on a small dialogue corpus based on a pre-trained LM. Our large datasets contain a few million samples, and the small datasets consist of 100K samples[3]. Although the datasets are smaller than those used in several previous studies, we believe that a comparison of different models on the same data, and the contrast between large and small datasets, can reveal interesting trends, which we will explain with respect to discrepancies.

Specifically, we choose the following 3 datasets: **Twitter Dialogue Corpus** [4] is collected from Twitter consisting of 2.6M (message, response) pairs. We filtered out samples with history length longer than 72 words (to limit the computation) or shorter than 6 words (not enough information). Samples whose response is longer than 36 words or shorter than 6 words are also removed. As a result, 2M samples are kept. **Reddit Conversational Corpus** [5](Dziri et al., 2019) is a 3-turn conversational dataset collected from 95 selected subreddits. **Ubuntu Dialogue Corpus V2.0** [6] (Lowe et al., 2017) contains two-person conversations extracted from the Ubuntu chat logs of technical support for various Ubuntu-related problems.

3.2 Implementation Details

We use open-source implementations for all four frameworks. Only minor adaptations (e.g. for data loading) have been made. The pre-trained language models used by these frameworks in previous studies have comparable number of parameters ($\sim$ 110M), while the pre-training data are in different scales: Trans-ED $<$ Trans-MLM/AR $<$ Trans-Dec. We assume that the difference is trivial when there are millions of dialogue data. In this study, we use the same data for all the frameworks. More implementation details of each framework and the full comparison among pre-trained LM are given in Appendix C.

We also equip all frameworks with an identical decoding script[7] to avoid extra factor affecting the generation quality, which uses beam search with beam size of 4, prevents duplicated uni-grams, and sets minimum response length that encourages diverse generation as in Roller et al. (2020). The minimum response length is set to make the average length of generated responses match with the average target length of the dataset. Generation results are evaluated after applying an identical word tokenization method. With two P100 GPU devices, the maximum input length is set to 128, and we fine-tune all models for 6 epochs and apply early stopping based on the performance on validation set. Our methods (PF-free and FG-free, which will be described in Section 4.1) do not add parameters or increase runtime in comparison with Trans-MLM.

3.3 Evaluation

Automatic Metrics We compare the similarity between generated responses and ground-truth responses using[8]: **BLEU** (Papineni et al., 2002) evaluating how many n-grams (n=1,2,3) overlapped; **CIDEr** (Vedantam et al., 2015) utilizing TF-IDF weighting for each n-gram. Besides, we evaluate response diversity using **Distinct** (denoted Dist) (Li et al., 2016) that indicates the proportion of unique n-grams (n=1,2) in the entire set of generated responses.

---

[3] Labeled datasets such as persona (Zhang et al., 2018) and emotion (Rashkin et al., 2019) are usually in similar scale.

[4] `https://github.com/Marsan-Ma-zz/chat_corpus`

[5] `https://github.com/nouhadziri/THRED`

[6] `https://github.com/rkadlec/ubuntu-ranking-dataset-creator`

[7] `https://github.com/microsoft/unilm/`

[8] We use an open-source evaluation tool: `https://github.com/Maluuba/nlg-eval`

Model	BLEU-1	BLEU-2	BLEU-3	CIDEr	Dist-1	Dist-2	avgLen
SEQ2SEQ-MMI	10.872 (**)	4.555 (**)	2.259 (/)	**0.119** (/)	0.008 (**)	0.028 (**)	10.6
Trans-ED	15.319 (**)	4.877 (**)	2.037 (**)	0.097 (**)	0.014 (**)	0.063 (**)	19.0
Trans-Dec	14.363 (**)	4.861 (**)	2.120 (*)	0.101 (**)	**0.031** (**)	**0.178** (/)	19.9
Trans-MLM	13.749 (**)	4.253 (**)	1.715 (**)	0.061 (**)	0.018 (**)	0.106 (**)	29.3
Trans-AR	**15.694**	**5.221**	**2.272**	**0.119**	0.029	0.164	18.9
FG-free	15.659 (/)	5.176 (/)	2.200 (/)	0.112 (/)	0.027 (**)	0.147 (*)	18.7
Trans-ED	14.813 (**)	4.249 (**)	1.330 (**)	0.066(**)	0.001 (**)	0.004 (**)	18.4
Trans-Dec	13.805 (**)	4.407 (**)	1.787 (**)	0.092(*)	**0.033** (**)	**0.195** (**)	20.2
Trans-MLM	15.487(**)	4.766(**)	1.814(**)	0.092 (*)	0.016(**)	0.080(**)	19.7
Trans-AR	15.213 (**)	4.700 (**)	1.767 (**)	0.090(**)	0.019(**)	0.091(**)	18.8
PF-free	15.880 (*)	4.970 (*)	1.868 (*)	0.093 (*)	0.022 (**)	0.114 (*)	15.7
FG-free	**16.395**	**5.218**	**2.043**	**0.101**	0.026	0.129	16.2
PF&FG-free	15.714 (*)	4.916 (*)	1.780 (**)	0.093 (*)	0.020 (**)	0.111 (*)	18.4

Table 3: Evaluation results on large-scale (upper half) and small-scale (lower half) Twitter dataset. PF-free denotes the method with reduced pretrain-finetune discrepancy of Trans-MLM. FG-free denotes the method that eliminates finetune-generation discrepancy of Trans-MLM. Two-sided t-test compares each method with the one without () sign, which is usually the best performer. Scores are denoted with * ($p < 0.05$) or ** ($p < 0.01$) for statistically significant differences, and / for insignificant differences.

Model	BLEU-1	BLEU-2	BLEU-3	CIDEr	Dist-1	Dist-2	avgLen
SEQ2SEQ-MMI	12.056(**)	5.512(**)	2.841(**)	0.142(**)	0.005(**)	0.024(**)	9.8
HRED-MMI	13.518(**)	4.564(**)	1.947(**)	0.060(**)	0.001(**)	0.003(**)	13.6
Trans-ED	19.295(/)	6.712(**)	2.986(*)	0.125(**)	0.010(**)	0.069(**)	16.8
Trans-Dec	18.974(*)	6.911(/)	3.022(*)	0.130(*)	**0.018**(**)	**0.134**(**)	18.0
Trans-MLM	17.574(**)	5.884(**)	2.552(**)	0.096(**)	0.012(**)	0.097(**)	25.5
Trans-AR	**20.103**	**7.270**	**3.339**	**0.143**	0.017	0.127	16.8
FG-free	19.774 (/)	7.045 (/)	3.213 (/)	0.139 (/)	0.016 (*)	0.115 (/)	17.7
Trans-ED	14.195(**)	4.533(**)	1.756(**)	0.074(**)	0.003(**)	0.012(**)	16.3
Trans-Dec	17.944(**)	6.360(*)	2.727(*)	0.121(/)	**0.018**(**)	**0.143**(**)	18.3
Trans-MLM	18.338(*)	6.018(**)	2.480(**)	0.108(**)	0.011(**)	0.066(**)	17.0
Trans-AR	19.005 (*)	6.431 (/)	2.733 (*)	0.114(*)	0.012(**)	0.078(**)	17.4
PF-free	**19.116** (*)	6.356 (*)	2.684 (*)	0.118 (/)	0.012 (**)	0.086 (*)	16.7
FG-free	18.884	**6.530**	**2.869**	**0.125**	0.014	0.095	17.3
PF&FG-free	19.024 (*)	6.448 (/)	2.740 (*)	0.118 (/)	0.012 (**)	0.087 (*)	17.1

Table 4: Evaluation results on large-scale (upper half) and small-scale (lower half) Ubuntu dataset.

Human Evaluation Furthermore, we ask human evaluators to rate a response in $\{0, 1, 2\}$. 2 represents a coherent and informative response. Details are given in Appendix D. We also do a pair-wise evaluation to compare two models and indicate which one is better. To reduce time cost, we only perform human evaluations on Twitter and Reddit datasets that are closer to daily dialogue. However, during evaluation, we observe that $\sim 65\%$ Reddit data are professional discussions that are difficult to understand. The percentage is $\sim 30\%$ for Twitter data. These test samples are discarded, and at the end the test set for each dataset consists of 200 random samples. The inter-rater annotation agreement in Cohen's kappa (Cohen, 1960) is 0.44 and 0.42 for Twitter and Reddit, which indicates moderate agreement.

In addition to the 4 frameworks, we also include two general RNN-based baseline frameworks – SEQ2SEQ-MMI (Li et al., 2016) and HRED-MMI (Serban et al., 2016) to show how pre-trained models perform against them.

3.4 Architecture Analysis

We first examine architecture appropriateness on the large-scale data setting, since when data are limited pretrain-finetune discrepancy and the size of pre-training data may strongly influence the results. Appendix E shows some generation samples. Our global observation is that Trans-Dec and Trans-AR are the best choice for large-scale data setting, e.g. further dialogue pre-training based on a pre-trained LM.

Left-to-Right Only vs. Bi-Direction on the Source Human evaluation results in response appropriateness (Table 6 and 7) show that Trans-Dec and Trans-AR generate most appropriate responses. According to automatic metrics, Trans-AR applying bi-directional attention on the source side obtains the highest BLEU and CIDEr scores on all

Model	**BLEU-1**	**BLEU-2**	**BLEU-3**	**CIDEr**	**Dist-1**	**Dist-2**	**avgLen**
SEQ2SEQ-MMI	15.550(**)	6.814(**)	3.321(**)	0.168(**)	0.011(**)	0.036(**)	11.2
HRED-MMI	13.278(**)	3.845(**)	1.398(**)	0.047(**)	0.001(**)	0.003(**)	13.8
Trans-ED	17.946(/)	6.626(**)	3.213(**)	0.165(**)	0.039(**)	0.203(**)	18.8
Trans-Dec	17.581(**)	6.790(*)	3.372(*)	0.180(**)	0.043(/)	**0.248**(**)	18.2
Trans-MLM	18.672(**)	7.115(**)	3.484(/)	0.177(**)	0.041(**)	0.215(**)	16.8
Trans-AR	**18.849**	**7.245**	**3.662**	**0.192**	**0.044**	0.235	16.8
FG-free	18.741 (/)	7.134 (**)	3.504 (*)	0.184 (*)	0.042 (**)	0.225 (**)	17.0
Trans-ED	17.337(**)	5.366(**)	1.967(**)	0.073(**)	0.001(**)	0.003(**)	17.1
Trans-Dec	17.460(**)	6.586(**)	3.161(*)	0.172(/)	**0.045**(/)	**0.254**(**)	17.7
Trans-MLM	19.193 (/)	6.877 (/)	3.175(*)	0.152(**)	0.029(**)	0.128(**)	15.0
Trans-AR	18.749(/)	6.746(/)	3.119(*)	0.153(**)	0.031(**)	0.141(**)	16.2
PF-free	18.466 (/)	6.688 (*)	3.075 (*)	0.169 (*)	0.038 (/)	0.180 (*)	14.1
FG-free	18.610	**6.937**	**3.302**	**0.175**	0.040	0.191	14.1
PF&FG-free	**19.302** (*)	6.923 (/)	3.073 (*)	0.159 (**)	0.034 (*)	0.164 (**)	15.3

Table 5: Evaluation results on large-scale (upper half) and small-scale (lower half) Reddit dataset.

Model	**Score** (M)	**Score** (K)
SEQ2SEQ-MMI	0.39	-
Trans-ED	0.53	0.11
Trans-Dec	**1.02**	0.77
Trans-MLM	0.88	0.58
Trans-AR	0.99	0.47
PF-free	-	0.52
FG-free	0.91	**0.78**
PF&FG-free	-	0.72
	Trans-Dec (M)	FG-free (K)
SEQ2SEQ-MMI	(11%, 48%)	-
Trans-ED	(14%, 46%)	(4%, 47%)
Trans-Dec	/	(24%, 29%)
Trans-MLM	(24%, 34%)	(18%, 31%)
Trans-AR	(27%, 32%)	(17%, 34%)
PF-free	-	(18%, 38%)
FG-free	(28%, 32%)	/
PF&FG-free	-	(23%, 29%)

Table 6: Human evaluation including pair-wise evaluation (lower half) for generated response quality for million-scale (M) Twitter dataset and its 100K training subset (K). Pair-wise comparisons show the wining percentages of the two parties.

Model	**Score** (M)	**Score** (K)
SEQ2SEQ-MMI	0.12	-
Trans-ED	0.33	0.10
Trans-Dec	0.58	**0.43**
Trans-MLM	0.48	0.38
Trans-AR	0.64	0.31
PF-free	-	0.28
FG-free	**0.68**	0.40
PF&FG-free	-	0.33
	FG-free (M)	Trans-Dec (K)
SEQ2SEQ-MMI	(5%, 40%)	-
Trans-ED	(11%, 33%)	(2%, 28%)
Trans-Dec	(25%, 32%)	/
Trans-MLM	(18%, 29%)	(15%, 19%)
Trans-AR	(18%, 23%)	(15%, 23%)
PF-free	-	(15%, 24%)
FG-free	/	(23%, 24%)
PF&FG-free	-	(16%, 24%)

Table 7: Human evaluation on Reddit dataset.

three million-scale datasets. We believe that bidirectional attention helps the model to better encode the dialogue history. In contrast, Trans-Dec is able to generate the most diverse responses. We attribute it to the left-to-right attention that introduces less constraints than bidirectional attention, thus has a higher flexibility for generation.

Trans-MLM vs. AR With large data, Trans-AR substantially outperforms Trans-MLM in terms of both automatic and human evaluation. When eliminating the finetune-generation discrepancy of Trans-MLM, i.e. FG-free (we will introduce in Section 4.2), the performance is improved while still having a small gap especially in automatic metrics to Trans-AR. This may be because MLM objective only masks a certain percentage of tokens (40%) while AR objective predicts all tokens on the target side for training. Thus, the AR objective is more training-efficient. Similar observation about the efficiency of MLM has been reported in Clark et al. (2020). However, when training data are limited, we will show that it is better to use MLM objective which has smaller pretrain-finetune discrepancy.

Trans-ED vs. Decoder-Only With large dialogue data, we assume the size of pre-training data and pretrain-finetune discrepancy only have small influence on performance. However, even comparing with Trans-MLM(FG-free)/AR, Trans-ED generates much less diverse or appropriate responses. We also observe lower speed for convergence when training the model [9]. We believe that the result is more or less due to the main difference in architecture: an explicit encoder in Trans-ED might be

---

[9] Similar observation has been reported in: `https://github.com/atselousov/transformer_chatbot/issues/15`

redundant (Liu et al., 2018).

3.5 Discrepancy Impact

In section 2, we have discussed the pretrain-finetune discrepancy of each framework. When a large training dataset is available, the impact of pretrain-finetune discrepancy is less severe since the model can be gradually adapted to the given task. However, if the training data are limited, the discrepancy problems may surface. Evaluation results, especially in human evaluation, show that the performance is more reduced with small data if the framework has larger discrepancy. For example, by comparing Trans-MLM (FG-free) and Trans-AR, the latter having additional pretrain-finetune discrepancy due to its auto-regressive objective, we see that the performance of Trans-AR drops more when trained on a small dataset. Trans-MLM (FG-free) and Trans-Dec that have small pretrain-finetune discrepancy have clear advantage over other frameworks according to human evaluation.

These results suggest that with a small dataset one should reduce pretrain-finetune discrepancy to best exploit pre-trained LM. In the next section, we propose 2 methods to reduce pretrain-finetune discrepancy and finetune-generation discrepancy of Trans-MLM.

4 Discrepancy-Free Trans-MLM

4.1 Pretrain-Finetune Discrepancy

The discrepancy of Trans-MLM comes from the left-to-right attention on the target side that has not been pre-trained in BERT. Therefore, this discrepancy cannot be eliminated during fine-tuning for a generation task. However, we can alleviate the discrepancy by using bi-directional attention also on the target side. Specifically, at inference time, to generate a new token denoted as g_t, [MASK] is fed into t-th position, denoted as g_t-M. Previously generated tokens $g_{<t}$ could be viewed as a special type of dialogue history, and thus we can apply bi-directional attention on it.

However, in this case, the corresponding training process will have efficiency problems – only one token can be masked in each training sample; otherwise, there will be conflict for the self-attention mask (Appendix B). This would lead to much lower training efficiency: the loss on validation set only decreases slightly to 5.39 from 6.27 after four epochs, while Trans-MLM masking 40% of the target tokens can reduce it to 4.35. To avoid this situation, we cannot always update previous hidden states using bi-directional attention in generation. Therefore, we explore to set a time-step interval for bi-directional attention on the target side – within the interval we apply left-to-right attention and at the end of an interval we apply bi-directional attention. The corresponding training method allows us to mask multiple target tokens at the same time to guarantee training efficiency.

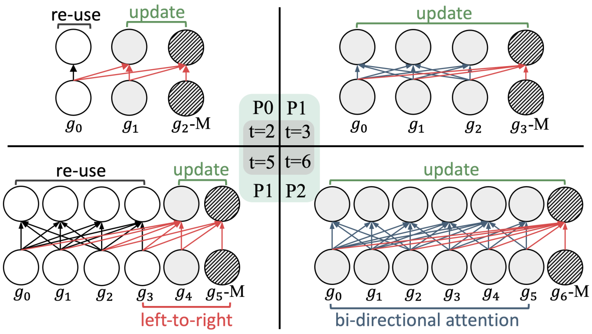

Figure 2: The generation process of PF-free at 4 different time steps. Bi-attention interval is 3 in the graph.

Figure 2 illustrates the generation process of our method with interval of 3. Before time step 3, left-to-right attention is used (e.g. t=2). At time step 3, bidirectional attention is allowed. Then left-to-right attention is used (e.g. t=5) before the end of next interval cycle (t=6). Accordingly, the training process is: given a target response, we first randomly select among all (3 in the figure because t=3 and t=5 are the same pattern) possible attention patterns (e.g. the case of t=3 or t=5 in Figure 2, where we apply bi-directional attention only on $y_{0,1,2}$); then in the part of left-to-right attention, we randomly mask several tokens. We can mask multiple tokens because this part applies left-to-right attention and the masks at other positions will not influence the prediction on a given mask. We call this method **PF-free**, which means that the pretrain-finetune discrepancy is reduced.

4.2 Finetune-Generation Discrepancy

A model having finetune-generation discrepancy means the way that it is used in generation (inference/test) is different from the way it has been trained. Only Trans-MLM has finetune-generation discrepancy because of its MLM objective as shown in Figure 3: during training, there is a masked token, y_1-M, before y_2-M, while in inference there is not a masked token before when generating the token for g_2-M.

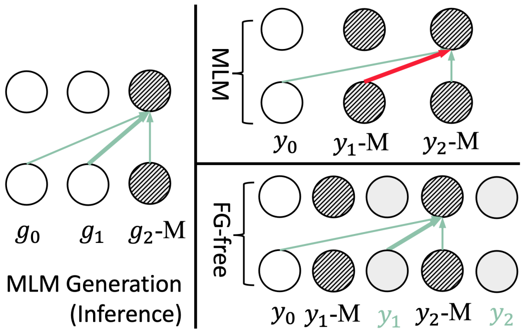

Figure 3: The training process of vanilla Trans-MLM and FG-free. We only plot the attention connection at the second position.

To deal with the problem, we propose that at training time, rather than replacing the tokens with [MASK] as in vanilla MLM, we keep all original input tokens unchanged and prepend [MASK] tokens in the input sequence as illustrated. The prepended [MASK] token uses the same position embedding of the corresponding token. Then, every position after y_1-M attends to y_1 instead of the [MASK] token, and thus the finetune-generation discrepancy of MLM is eliminated. We call the modified model **FG-free**. A similar method has also been explored in (Bao et al., 2020a), where they introduced an extra pseudo mask in addition to [MASK] and prepend it before the original token in order to handle factorization steps of their partially auto-regressive language model.

4.3 Experimental Results

The results with PF-free, FG-free and PF&FG-free models on small-scale datasets are reported in previous tables together with other models. We can see that each of the proposed methods brings some improvement. PF-free improves most automatic metrics over Trans-MLM, but the response appropriateness in human evaluation is not improved. We observe that PF-free could generate some responses that lack fluency, which also influences PF&FG-free (Appendix E). In general, our exploration shows that the left-to-right attention on the target side is necessary for a generative task.

We examine our FG-free method on both large and small-scale data. It always brings statistically significant improvement over Trans-MLM in all automatic metrics, and generates more appropriate responses. On small-scale datasets, it outperforms all other frameworks in similarity metrics and achieve comparable performance in response appropriateness to Trans-Dec that has leveraged much more pre-training data.

This set of experimental results confirm the usefulness of reducing discrepancies in the model. This demonstrates that model discrepancies are indeed important problems we need to address when a pre-trained LM is used for dialogue generation, and the problems have been under-explored.

Conclusion

In this paper, we examined the 4 frameworks for open-domain dialogue based on pre-trained models. We compared their performances on several datasets with the same setting. The comparison revealed that Trans-Dec and Trans-AR are both good choices when large-scale data are available, e.g. further dialogue pre-training. When data are limited, e.g. fine-tuning on small dialogue tasks, Trans-Dec is the most appropriate.

Furthermore, we defined the concept of pretrain-finetune and finetune-generation discrepancy, and examined the 4 frameworks with respect to these concepts. We have shown that the performances of the 4 frameworks can be largely explained by their respective discrepancies, which hinder their performances. This becomes more clear when the dataset is small.

To further show that reducing the discrepancies can improve the performance, we designed PF-free and FG-free correction methods to reduce the discrepancies on Trans-MLM, and tested the corrected Trans-MLM models on the datasets. Our results confirmed that once discrepancies are eliminated, Trans-MLM can produce better results.

This study is the first investigation on the widely used 4 frameworks based on pre-trained LM in terms of architectural appropriateness and discrepancies. We believe that this question is important to understand how a pre-trained model can be used in dialogue generation. It deserves more investigations in the future.

Acknowledgments

This research work is partly supported by an NSERC discovery grant.

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, et al. 2020a. Unilmv2: Pseudo-masked language models for unified language model pre-training. *arXiv preprint arXiv:2002.12804*.

Siqi Bao, Huang He, Fan Wang, and Hua Wu. 2019. Plato: Pre-trained dialogue generation model with discrete latent variable. *arXiv preprint arXiv:1910.07931*.

Siqi Bao, Huang He, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, Zhen Guo, Zhibin Liu, and Xinchao Xu. 2020b. Plato-2: Towards building an open-domain chatbot via curriculum learning. *arXiv preprint arXiv:2006.16779*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054.

Nouha Dziri, Ehsan Kamalloo, Kory Mathewson, and Osmar R Zaiane. 2019. Augmenting neural response generation with context-aware topical attention. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 18–31.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*.

Zhaojiang Lin, Peng Xu, Genta Indra Winata, Zihan Liu, and Pascale Fung. 2019. Caire: An end-to-end empathetic chatbot. *arXiv preprint arXiv:1907.12108*.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Kurt Shuster, Da Ju, Stephen Roller, Emily Dinan, Y-Lan Boureau, and Jason Weston. 2019. The dialogue dodecathlon: Open-domain knowledge and image grounded conversational agents. *arXiv preprint arXiv:1911.03768*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Yan Zeng and Jian-Yun Nie. 2021. A simple and efficient multi-task learning approach for conditioned dialogue generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4927–4939.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

Yinhe Zheng, Rongsheng Zhang, Xiaoxi Mao, and Minlie Huang. 2019. A pre-training based personalized dialogue generation model with persona-sparse data. *arXiv preprint arXiv:1911.04700*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

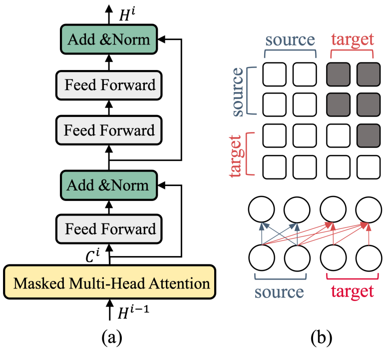

Figure 4: i-th Transformer Block and two $\mathbf{M}$ settings represented in two ways. Shaded areas are blocked.

A Multi-Layer Transformer

In this section, we provide some background knowledge on Transformer. The four frameworks we discussed all consist of 12 Transformer blocks. Figure 4 (a) shows a general architecture of a Transformer layer, where the most important component is the masked multi-head self-attention. The setting of attention masks is the largest difference between Trans-Dec and Trans-AR, and it is also the most critical part to implement our PF-free and FG-free methods.

The input representation $\mathbf{H}^0 \in \mathbb{R}^{n \times d_h}$, where n is the input length and $d_h = 768$ is the hidden dimension, is the sum of token embedding, position embedding, and type embedding at each position. Then, $\mathbf{H}^0$ is encoded into hidden representations of i-th layer $\mathbf{H}^i = [\mathbf{h}^i_1, ..., \mathbf{h}^i_n]$ by: $\mathbf{H}^i = \text{Trans}^i(\mathbf{H}^{i-1}), \quad i \in [1, L]$, where Trans^i denotes the i-th Transformer Block as shown in Figure 4 (a). The core component of a transformer block is the masked multi-head attention, whose outputs are $\mathbf{C}^i = [\mathbf{c}^i_1, ..., \mathbf{c}^i_n]$ that are computed via $\mathbf{C}^i = \text{Concat}(\mathbf{head}_1, ..., \mathbf{head}_h)$, with

$$\mathbf{head}_j = \text{softmax}(\frac{\mathbf{Q}_j \mathbf{K}_j^T}{\sqrt{d_k}} + \mathbf{M})\mathbf{V}_j \qquad (1)$$

where $\mathbf{Q}_j, \mathbf{K}_j, \mathbf{V}_j \in \mathbb{R}^{n \times d_k}$ are obtained by transforming $\mathbf{H}^{i-1} \in \mathbb{R}^{n \times d_h}$ using $\mathbf{W}^Q_j, \mathbf{W}^K_j, \mathbf{W}^V_j \in \mathbb{R}^{d_h \times d_k}$ respectively. $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the **self-attention mask matrix** that determines whether a position can attend to other positions. $\mathbf{M}_{ij} \in \{0, -\infty\}$. In particular, $\mathbf{M}_{ij} = 0$ allows the i-th position to attend to j-th position and $\mathbf{M}_{ij} = -\infty$ prevents from it. Figure 4 (b&c) shows two $\mathbf{M}$ settings that are applied by Trans-MLM/AR and Trans-Dec respectively.

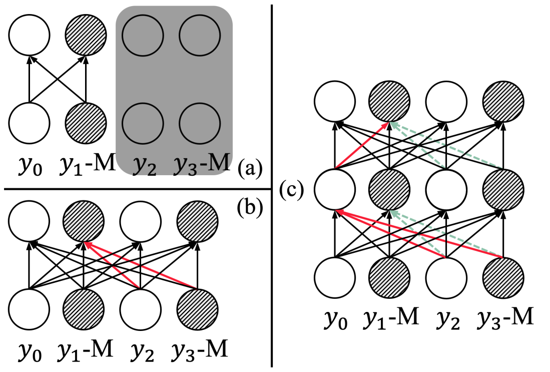

Figure 5: Self-attention mask, $\mathbf{M}$, conflicts – (a) if predicting y_1, y_2 and y_3-M are "future" and forbidden to be accessed by y_1-M; (b) if predicting y_3, y_1-M accesses to y_2 and y_3-M, which causes conflicts to $\mathbf{M}^{(a)}$; (c) if forbidding y_1-M to access to y_2 and y_3-M in $\mathbf{M}^{(b)}$, there will still be (indirect) information leak as indicated in red arrows (y_2 and y_3-M → y_0 → y_1-M). Masking two positions thus causes conflicts. Our PF-free method aims to overcome this problem.

B Illustration of Attention Conflict

If applying bi-directional attention at each generation step, only one token at the target side could be masked for each training sample; otherwise there will be attention conflicts, i.e. different self-attention mask matrices are required for different masked tokens, while only one mask matrix can be provided per training sample. In Figure 5, we provide an illustration of the mask conflict problem. We assume y_1 and y_3 are masked and need to be predicted at the same time. We see in the figure that two different masks are required for predicting y_1 and y_3, which cannot be done in a single training step, making it impossible to mask more than one token in each step.

C Implementation Details

For the 4 frameworks, we used open-source implementations. Only some minor adaptations to our data and task are made (e.g. re-wrote the data loader to load our experimental datasets, and modified the training objective by keeping only the response reconstruction loss). For response generation, we equipped all frameworks with an identical decoding script [10]. We did not modify other parts, and used the default settings for hyper-parameters, e.g. optimizer and learning rate. Some generation examples are given in Appendix E. Although some models (e.g. Trans-ED) produced poor per-

---

[10] `https://github.com/microsoft/unilm/`

Model	Pre-trained LM	Data
Trans-ED	GPT (Radford et al., 2018)	BooksCorpus
Trans-Dec	GPT-2 small (Radford et al., 2019)	WebText
Trans-MLM/AR	BERT base (Devlin et al., 2018)	BooksCorpus, English Wikipedia

Table 8: The text data used for language model pre-training.

formance on small datasets, all model can generate some coherent and fluent responses with large scale training data, which is consistent with the performances reported in previous papers.

Language Models The pre-trained language models used by these frameworks have comparable number of parameters as listed in Table 9, while the pre-training data are in different scales as described in Table 8. BooksCorpus (Zhu et al., 2015) (800M words) contains over 7,000 unique unpublished books from a variety of genres. English Wikipedia (2,500M words) consists of the text passages of Wikipedia extracted by Devlin et al. (2018). WebText crawled by Radford et al. (2019) contains 8M diverse documents for a total of 40 GB of text.

Trans-ED We use the implementation of ConvAI2 champion [11]. The model was for persona-conditioned dialogue generation. The framework is based on GPT architecture and uses GPT for parameter initialization. However they only provide a model checkpoint that has been fine-tuned on large-scale dialogue data including Reddit. To examine the ability of utilizing pre-trained LM, we did not use this checkpoint but initialize the model with GPT parameters [12]. We also did not apply post-processing to the generation results (to be consistent with other experiments).

Trans-Dec We use the released code of Wolf et al. (2019)[13] that uses GPT-2 small by default. The model was for persona-conditioned dialogue generation.

Trans-MLM/AR These two models are implemented based on Dong et al. (2019) [14] that applies multi-task learning on language understanding and generation tasks. We use BERT (base, uncased) for parameter initialization, and fine-tune it on dialogue datasets. PF-free and FG-free are also implemented based on the code. We set the bi-directional attention interval of PF-free to 5. Since the average length of ground-truth responses in the datasets is ~ 15, This setting is generally appropriate.

Model	Params	Runtime
SEQ2SEQ-MMI	66M	50
HRED-MMI	58M	25
Trans-ED	117M	180
Trans-Dec	117M	290
Trans-MLM	110M	140
Trans-AR	110M	140
PF&FG-free	110M	140

Table 9: The number of parameters of each tested approach and the average runtime (minutes) for every million training samples. The runtime is tested using a 1080Ti GPU device, and the batch size is set to take all of the GPU memories. Notice that the runtime will be influenced by code implementation in addition to model structure.

D Human Evaluation Details

Some existing studies considered response fluency, coherence, and informativeness. We make the manual evaluation simpler and ask the human evaluators to rate a response in $\{0, 1, 2\}$. A score of 0 represents an unacceptable response, which might have flaw in fluency and logic or be incoherent. Special cases are for example completely coping from the dialogue history as the output, and a bland response such as “i do not know what you are talking about , but it ’s a good point .”. A score of 1 represents an acceptable response, but it is generic or not perfectly coherent to the dialogue history. 2 represents a coherent and informative response.

E Generation Examples

---

[11] https://github.com/atselousov/transformer_chatbot

[12] https://github.com/openai/finetune-transformer-lm/tree/master/model

[13] https://github.com/huggingface/pytorch-openai-transformer-lm

[14] https://github.com/microsoft/unilm/tree/master/unilm-v1

Dialogue History	one week before election day , early voting is nearly twice as high as <num>
Gold Response	i hope it 's <person> out in full force .
SEQ2SEQ-MMI	i am not voting for a third party candidate . (1)
Trans-ED	i am not sure what you are talking about , but it 's a good thing that <person> is going to win . (1)
Trans-Dec	that 's not true . early voting in nyc is times higher than the national average (2)
Trans-MLM	it 's not even close to <num> % of the vote . i am sure you are right , but there is a huge difference between early voting and <person> voter suppression in ca (1)
Trans-AR	it 's not that high , but i am sure there will be a lot of voter fraud . (2)
FG-free	it 's not even close to <num> yet , but i am sure the polls will show up . (2)
Trans-ED (K)	i am not sure what you are talking about . it 's the only one who is going to do with this (0)
Trans-Dec (K)	it 's not that early voting is a bad thing , but it shouldnot be the case . (0)
Trans-MLM (K)	it 's a little too late . i am not sure if that is the case , but we will see what happens next (2)
Trans-AR (K)	it 's not even close to <num> % of voter turnout . i am sure you are right about that (0)
PF-free (K)	<num> % of people voted for <person> , so i am going to vote (2)
FG-free (K)	it 's not even close to <num> % of voter turnout , but i am sure there are more . (2)
PF&FG-free (K)	it 's not even <num> % of the vote . i am going to be a voter (2)

Table 10: Responses generated by baselines and our methods trained on the **Twitter** dataset(million-scale and 100K). Human evaluation scores are given at the end of each generated reply.

Dialogue History	i think about this man every day
Gold Response	it is so hypnotic . this man was found dead post-election
SEQ2SEQ-MMI	i do not know what you are talking about . (0)
Trans-ED	i think he 's the only one who has a lot of work to do with it . (1)
Trans-Dec	i think about this man every day too . he 's my favorite person in the world (2)
Trans-MLM	he 's such a good guy . i love him so much, but it doesnot make sense to me that this is the only way we can be together in our lives (2)
Trans-AR	i think about this man every day too . he 's so good at it ! (2)
FG-free	he 's such a good guy . i am so happy for him and his family (2)
Trans-ED (K)	i am not sure what you are talking about , but it 's a good idea . (0)
Trans-Dec (K)	i think about this man every day . he 's my hero and i love him (2)
Trans-MLM (K)	i think he 's a great actor , but no one cares about him . lol (2)
Trans-AR (K)	i think he 's a good guy , but not sure what you are talking about . (0)
PF-free (K)	i think he 'snitch . it makes me happy to watch him (0)
FG-free (K)	he 's a great guy . i am so happy to have him back (2)
PF&FG-free (K)	i think he is little silly , but you are a fool . (2)

Table 11: Responses generated by baselines and our methods trained on the **Twitter** dataset(million-scale and 100K). Human evaluation scores are given at the end of each generated reply.

Dialogue History	how old are you guys ? you did not have mechanical pencils ? [SEP] <url> young are you ? i was in school all the way up to a few years ago and most of us still used non - mechanical ones .
Gold Response	i started using mechanical pencil in elementary school about 15 years ago .
SEQ2SEQ-MMI	i do not know what you are talking about . (0)
Trans-ED	how old are you ? i was in school all the way up to a few years ago . (0)
Trans-Dec	i was in school all the way up to a few years ago and most of us still used non - mechanical ones . (0)
Trans-MLM	i was in school for a few years , and most of us still use mechanical pencils . it 's not the same age as you (2)
Trans-AR	i was in high school when we used to use mechanical pencils . (2)
FG-free	how old are you ? i was in high school and still use mechanical pencils . (2)
Trans-ED (K)	i do not know what you are talking about , but it 's just a joke . (0)
Trans-Dec (K)	i was in school all the way up to a few years ago and most of us still used non - mechanical ones . (0)
Trans-MLM (K)	i have been in school for a few years , but not really . (0)
Trans-AR (K)	how old are you ? i have never been in school for a few years . (0)
PF-free(K)	how old are you ? i am not really a mechanical engineer . ! (1)
FG-free (K)	how old are you ? i have never had a mechanical pencil . (2)
PF&FG-free (K)	how old are you ? i have never had a lot of time in school (2)

Table 12: Responses generated by baselines and our methods trained on the **Reddit** dataset(million-scale and 100K). Human evaluation scores are given at the end of each generated reply.

Dialogue History	i have read all of the books , which character do you not like ? [SEP] it ’s gon na be the dornish and greyjoys . and probably brienne . you will see .
Gold Response	nah , they have only just started a game of thrones so it must be one of the starks , dan , or tyrion .
SEQ2SEQ-MMI	i do not know why you are being downvoted . (0)
Trans-ED	i have read all of the books , and it ’s gon na be dornish . (2)
Trans-Dec	i have not read the books yet , but i will definitely check them out . (2)
Trans-MLM	i have not read the dornish and greyjoys , but you will see them all as well . (1)
Trans-AR	brienne is not a fictional character , she ’s an actual person . (2)
FG-free	i have read all of the dornish and greyjoys . (1)
Trans-ED (K)	i am not sure what you are talking about , but it ’s just a joke . (0)
Trans-Dec (K)	i do n’t know , i have read all of the dornish books . (1)
Trans-MLM (K)	i do not think it ’s a good idea to say that . (0)
Trans-AR (K)	i do not think it ’s gon na be the best . (0)
PF-free (K)	i am not sure what you are talking about here ’s a link . (0)
FG-free (K)	i do not think it ’s gon na be . (1)
PF&FG-free (K)	i do not know , but it ’s a lot of people ? (1)

Table 13: Responses generated by baselines and our methods trained on the **Reddit** dataset(million-scale and 100K). Human evaluation scores are given at the end of each generated reply.

Dialogue History	is that easy to do ? [SEP] usually the ubuntu installer will set it up perfectly for you
Gold Response	and then i would select at start-up ?
SEQ2SEQ-MMI	how do i install it ?
HRED-MMI	i am not sure what you mean by that
Trans-ED	is there a way to do it without installing the ubuntu installer ? i am not sure .
Trans-Dec	ok , i will try that . thanks for the help ! :-) it ’s working now
Trans-MLM	is there a way to do it with the live cd ? i am not sure if that ’s what you are looking for , but how does one install ubuntu on
Trans-AR	ok , i will give it a try . thank you very much for your help !
FG-free	is there a way to do it from the command line ? i am not sure what you mean
Trans-ED (K)	i am not sure what you mean by that , but it ’s just a little bit more specific
Trans-Dec (K)	ok , i will try that . thanks for your help ! :-) <url>
Trans-MLM (K)	i am not sure what you mean by that , but it ’s a good idea .
Trans-AR (K)	i am not sure what you are talking about , but it ’s possible that the installer will be able to boot from cd
PF-free (K)	i am not sure what to do ? it ’ s just a file in the repos
FG-free (K)	i do not think it ’ s a good idea to install ubuntu on my laptop
PF&FG-free (K)	i am not sure what you mean by setting it up perfectly . is there a way to do that ?

Table 14: Responses generated by baselines and our methods trained on the **Ubuntu** dataset(million-scale and 100K).

Dialogue History	usually if on the same machine , <url> [SEP] : i am getting error . phpmyadmin not in this server
Gold Response	you installed it from the software center , or with aptitude or apt-get , or some other way ?
SEQ2SEQ-MMI	what is the error you are getting ?
HRED-MMI	i have no idea what you are talking about
Trans-ED	i am not sure what you are trying to do , but it ’s a good idea .
Trans-Dec	did you install phpmyadmin from the repos ? if so , it should be in <path>
Trans-MLM	i am not sure what you are trying to do , but it should be in <path> . try sudo dpkg - reconfigure phpmyadmin
Trans-AR	i am not sure what you are trying to do , but it ’s possible .
FG-free	i am not sure what you are trying to do , but it ’ s usually in <path>
Trans-ED (K)	i am not sure what you mean by ” phpmyadmin <path>
Trans-Dec (K)	i am not sure what phpmyadmin is , but it ’s probably in <path>
Trans-MLM (K)	i am not sure what you mean by ” phpmyadmin - > <path> . <url>
Trans-AR (K)	i do not know what phpmyadmin is , but it ’s in the repos
PF-free (K)	it ’ s not in this server . i am trying to get phpmyadmin working
FG-free (K)	phpmyadmin is not in the same machine , it ’ s just a server .
PF&FG-free (K)	i am not sure if it ’s on the same machine , you can use phpmyadmin

Table 15: Responses generated by baselines and our methods trained on the **Ubuntu** dataset(million-scale and 100K).

Learning to Sample Replacements for ELECTRA Pre-Training

Yaru Hao[†,‡]*, **Li Dong**[‡], **Hangbo Bao**[‡], **Ke Xu**[†], **Furu Wei**[‡]
[†]Beihang University
[‡]Microsoft Research
{haoyaru@,kexu@nlsde.}buaa.edu.cn
{lidong1,t-habao,fuwei}@microsoft.com

Abstract

ELECTRA (Clark et al., 2020a) pretrains a discriminator to detect replaced tokens, where the replacements are sampled from a generator trained with masked language modeling. Despite the compelling performance, ELECTRA suffers from the following two issues. First, there is no direct feedback loop from discriminator to generator, which renders replacement sampling inefficient. Second, the generator's prediction tends to be over-confident along with training, making replacements biased to correct tokens. In this paper, we propose two methods to improve replacement sampling for ELECTRA pre-training. Specifically, we augment sampling with a hardness prediction mechanism, so that the generator can encourage the discriminator to learn what it has not acquired. We also prove that the efficient sampling reduces the training variance of the discriminator. Moreover, we propose to use a focal loss for the generator in order to relieve oversampling correct tokens as replacements. Experimental results show that our method improves ELECTRA pre-training on various downstream tasks. Our code and pre-trained models will be released at: `https://github.com/YRdddream/electra-hp`

1 Introduction

One of the most successful language model pre-training tasks is masked language modeling (MLM; Devlin et al. 2019). First, we randomly mask some input tokens in a sentence. Then the encoder learns to recover the masked tokens given the corrupted input. ELECTRA (Clark et al., 2020a) argues that MLM only produces supervision signals at a small proportion of positions (usually 15%), and uses the replaced token detection task as an alternative. Specifically, ELECTRA contains a generator and a discriminator. The generator is a masked language model, which substitutes masks with the tokens sampled from its MLM predictions. The discriminator learns to distinguish which tokens have been replaced or kept the same. Experimental results on downstream tasks show that ELECTRA can largely improve sample efficiency.

Despite achieving compelling performance, it is usually difficult to balance the training pace between the generator and the discriminator. Along with pre-training, the generator is expected to sample more hard replacements for the detection task in a curriculum manner, while the discriminator learns to identify the corrupted positions. Although the two components are designed to compete with each other, there is no explicit feedback loop from the discriminator to the generator, rendering the learning games independent. The absence of feedback results in sub-efficient learning, because many replaced tokens have been successfully trained while the generator does not know how to effectively sample replacements. In addition, a well trained generator tends to achieve reasonably good MLM accuracy, where many sampled replacements are correct tokens. In order to relieve the issue of oversampling correct tokens, ELECTRA explored tweaking the mask probability larger, raising the sampling temperature, and using a manual rule to avoid sampling original tokens.

In this paper, we propose two methods, namely hardness prediction and sampling smoothing, to tackle the above issues. First, the motivation of *hardness prediction* is to sample the replacements that the discriminator struggles to predict correctly. We elaborate on the benefit of a good replacement mechanism from the perspective of variance reduction. Theoretical derivations indicate that the replacement sampling should be proportional to both the MLM probability (i.e., language frequency) and the corresponding discriminator loss (i.e., discrimination hardness). Based on the above conclu-

---

[*] Contribution during internship at Microsoft Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4495–4506
August 1–6, 2021.

sion, we introduce a sampling head in the generator, which learns to sample by estimating the expected discriminator loss for each candidate replacement. So the discriminator can give feedback to the generator, which helps the model to learn what it has not acquired. Second, we propose a *sampling smoothing* method for the issue of oversampling original tokens. We adopt a focal loss (Lin et al., 2017) for the generator's MLM task, rather than using cross-entropy loss. The method adaptively down-weights the well-predicted replacements for MLM, which avoids sampling too many correct tokens as replacements.

We conduct pre-training experiments on the WikiBooks corpus for both small-size and base-size models. The proposed techniques are plugged into ELECTRA for training from scratch. Experimental results on various tasks show that our methods outperform ELECTRA despite the simplicity. Specifically, under the small-size setting, our model performance is 0.9 higher than ELECTRA on MNLI (Williams et al., 2018) and 4.2 higher on SQuAD 2.0 (Rajpurkar et al., 2016a), respectively. Under the base-size setting, our model performance is 0.26 higher than ELECTRA on MNLI and 0.52 higher on SQuAD 2.0, respectively.

2 Related Work

State-of-the-art NLP models are mostly pretrained on a large unlabeled corpus with the self-supervised objectives (Peters et al., 2018; Lan et al., 2020; Raffel et al., 2020). The most representative pretext task is masked language modeling (MLM), which is introduced to pretrain a bidirectional BERT (Devlin et al., 2019) encoder. RoBERTa (Liu et al., 2019) apply several strategies to enhance the BERT performance, including training with more data and dynamic masking. UniLM (Dong et al., 2019; Bao et al., 2020) extend the mask prediction to generation tasks by adding the auto-regressive objectives. XLNet (Yang et al., 2019) propose the permuted language modeling to learn the dependencies among the masked tokens. Besides, ELECTRA (Clark et al., 2020a) propose a novel training objective called replaced token detection which is defined over all input tokens. Moreover, ELECTRIC (Clark et al., 2020b) extends the idea of ELECTRA by energy-based cloze models.

Some prior efforts demonstrate that sampling more hard examples is conducive to more effective training. Lin et al. (2017) propose the focal loss in order to focus on more hard examples. Generative adversarial networks (Goodfellow et al., 2014) is trained to maximize the probability of the discriminator making a mistake, which is closely related to ELECTRA's training framework. In this work, we aim at guiding the generator of ELECTRA to sample the replacements that are hard for the discriminator to predict correctly, therefore the pre-training process of the discriminator can be more efficient.

3 Background: ELECTRA

An overview of ELECTRA is shown in Figure 1. The model consists of a generator G and a discriminator D. The generator is trained by masked language modeling (MLM). Formally, given an input sequence $\boldsymbol{x} = x_1 \cdots x_n$, we first randomly mask $k = \lceil 0.15n \rceil$ tokens at the positions $\boldsymbol{m} = m_1 \cdots m_k$ with `[MASK]`. The perturbed sentence $\boldsymbol{c}$ is denoted as:

$$\begin{aligned} m_i &\sim \text{uniform}\{1, n\} \quad \text{for } i = 1, \cdots, k \\ \boldsymbol{c} &= \text{replace}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]}) \end{aligned}$$

where the replace operation conducts masking at the positions $\boldsymbol{m}$. The generator encodes $\boldsymbol{c}$ and performs MLM prediction. At each masked position i, we sample replacements from MLM output distribution p_{G}:

$$\begin{aligned} {x'}_i &\sim p_{\mathrm{G}}(x'_i|\boldsymbol{c}) \quad \text{for } i \in \boldsymbol{m} \\ \boldsymbol{x}^{\mathrm{R}} &= \text{replace}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{x'}) \end{aligned}$$

where masks are replaced with the sampled tokens. Next, the discriminator encodes the corrupted sentence $\boldsymbol{x}^{\mathrm{R}}$. A binary classification task learns to distinguish which tokens have been replaced or kept the same, which predicts the probability $D(x_t^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{R}})$ to indicate how likely x_t^{R} comes from the true data distribution.

The overall pre-training objective is defined as:

$$\min_{\theta_G,\theta_D} \sum_{\boldsymbol{x}\in\mathcal{X}} \mathop{\mathbb{E}}_{i\in\boldsymbol{m}} [\mathcal{L}_G^{\theta_G}(x_i,\boldsymbol{c})] + \lambda \mathop{\mathbb{E}}_{t\in[1,n]} [\mathcal{L}_D^{\theta_D}(x_t^{\mathrm{R}},\boldsymbol{x}^{\mathrm{R}})]$$

$$\mathcal{L}_G(x_i, \boldsymbol{c}) = -\log p_{\mathrm{G}}(x_i|\boldsymbol{c})$$

$$\mathcal{L}_D(x_t^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{R}}) = \begin{cases} -\log D(x_t^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{R}}) & x_t^{\mathrm{R}} = x_t \\ -\log(1 - D(x_t^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{R}})) & x_t^{\mathrm{R}} \neq x_t \end{cases}$$

where $\mathcal{X}$ represents text corpus, and $\lambda = 50$ suggested by Clark et al. (2020a) is a hyperparameter used to balance the training pace of generator and discriminator. Once pre-training is finished, only the discriminator is fine-tuned on downstream tasks.

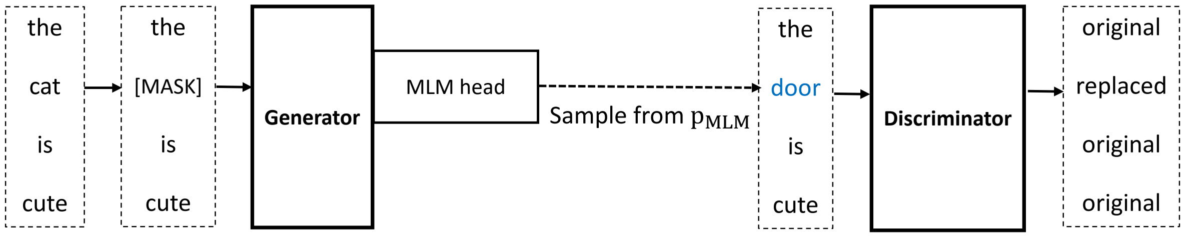

Figure 1: An overview of ELECTRA. The MLM head of the generator learns to perform MLM and samples replacements at each masked position from the MLM distribution. For the corrupted sequence, the discriminator learns to distinguish which tokens have been replaced or kept the same.

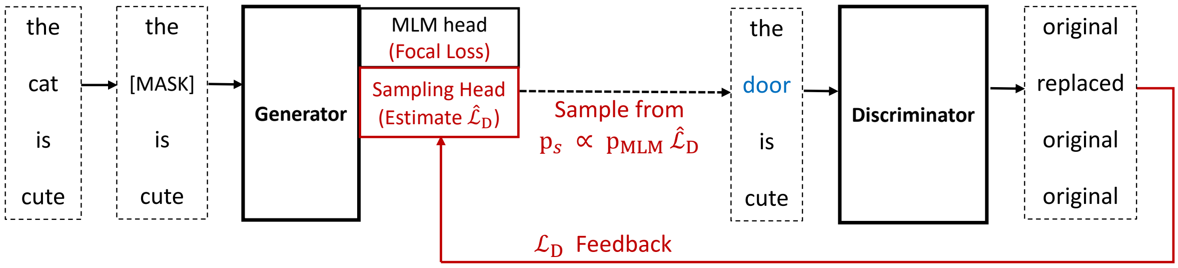

Figure 2: An overview of our model. The generator has two prediction heads. The MLM head learns to perform MLM through the focal loss instead of the cross entropy loss. The sampling head is trained to estimate the discriminator loss over the vocabulary. Our model samples the replacements from a new distribution, which is proportional to both the MLM probability and the corresponding discriminator loss. The discriminator is trained to distinguish input tokens and the loss feedback is transferred to the generator for the sampling head to learn.

4 Methods

4.1 Hardness Prediction

The key idea of hardness prediction is to let the generator receive the discriminator's feedback and sample more hard replacements. Figure 2 shows the overview of our method. Besides the original MLM head in the generator, there is an additional sampling head used to sample replaced tokens.

Given a[1] replaced token x' in the input sequence c, let $\mathcal{L}_D(x', c)$ denote the discriminator loss for the replacement. Rather than directly sampling replacements from the MLM prediction p_{G}, we propose to sample from p_{S}:

$$p_{\mathrm{S}}(x'|c) = \frac{p_{\mathrm{G}}(x'|c)\mathcal{L}_D(x',c)}{\mathrm{E}_{p_{\mathrm{G}}(x^*|c)}[\mathcal{L}_D(x^*,c)]} \qquad (1)$$

$$\boldsymbol{x}^{\mathrm{R}} = \mathrm{replace}(\boldsymbol{x},\boldsymbol{m},\boldsymbol{x}') \quad x' \sim p_{\mathrm{S}}(x'|c)$$

where the corrupted sentence $\boldsymbol{x}^{\mathrm{R}}$ is obtained by substituting the masked positions $\boldsymbol{m}$ with sampled replacements $\boldsymbol{x}'$. The first term $p_{\mathrm{G}}(x'|c)$ implies sampling from the data distribution. The second term $\mathcal{L}_D(x', c)$ encourages the model to sample more replacements that the discriminator has not successfully learned.

Notice that Equation (1) uses the actual discriminator loss $\mathcal{L}_D(x', c)$, which can not be obtained without feeding $\boldsymbol{x}^{\mathrm{R}}$ into the discriminator. As an alternative, we use the estimated loss value $\hat{\mathcal{L}}_D(x', c)$ to sample replaced tokens, which approximates the actual loss for the candidate replacement. During pre-training, we use the actual loss as supervision, and simultaneously train the sampling head. We describe the detailed implementations of loss estimation in Section 4.1.2.

By considering detection hardness in replacement sampling and giving feedback from the discriminator to the generator, the components are no longer independently learned. ELECTRA (Clark et al. 2020a; Appendix F) also attempts to achieve the same goal by adversarially training the generator. However, it underperforms the maximum-likelihood training, because of the poor sample efficiency of reinforcement learning on discrete text data. More importantly, their generator is trained to fool the discriminator, rather than guiding the discriminator by data distribution, which breaks the ELECTRA training objective. In contrast, we still retain the MLM head, and decouple it from re-

---

[1] For notation simplicity, we assume only one token is masked in each sentence.

placement sampling. So we can take the advantage of the original training objective.

4.1.1 Perspective of Variance Reduction

We show that the proposed hardness prediction method is well supported from the perspective of variance reduction.

Proposition 1. *Sampling replacements from $p_S(x'|c)$ can minimize the estimation variance of the discriminator loss.*

Proof. At each masked position, the expectation of the discriminator loss we aim to estimate can be summarized as $Z = E_{p_G(x^*|c)}[\mathcal{L}_D(x^*, c)]$. Under p_G, the estimation variance of the discriminator loss is:

$$\begin{aligned}&\mathrm{Var}_{p_G(x^*|c)}[\mathcal{L}_D(x^*, c)]\\ =&\sum_{x^*\in\text{vocab}} p_G(x^*|c)(\mathcal{L}_D(x^*, c) - Z)^2\\ =&\sum_{x^*\in\text{vocab}} p_G(x^*|c)\mathcal{L}_D(x^*, c)^2 - Z^2\end{aligned}$$

Similar to importance sampling, we can select an alternative distribution p_S different from p_G, then the expectation Z is rewritten as:

$$\begin{aligned}&E_{p_G(x^*|c)}[\mathcal{L}_D(x^*, c)]\\ =&\sum_{x^*\in\text{vocab}} p_G(x^*|c)\mathcal{L}_D(x^*, c)\\ =&\sum_{x^*\in\text{vocab}} p_S(x^*|c)\frac{p_G(x^*|c)}{p_S(x^*|c)}\mathcal{L}_D(x^*, c)\\ =&E_{p_S(x^*|c)}[\frac{p_G(x^*|c)}{p_S(x^*|c)}\mathcal{L}_D(x^*, c)]\end{aligned}$$

By making a multiplicative adjustment to $\mathcal{L}_D$, the estimation variance of Z under the new sampling distribution p_S is converted to:

$$\begin{aligned}&\mathrm{Var}_{p_S(x^*|c)}[\frac{p_G(x^*|c)}{p_S(x^*|c)}\mathcal{L}_D(x^*, c)]\\ =&\sum_{x^*\in\text{vocab}} p_S(\frac{p_G(x^*|c)\mathcal{L}_D(x^*, c)}{p_S})^2 - Z^2\\ =&\sum_{x^*\in\text{vocab}} \frac{(p_G(x^*|c)\mathcal{L}_D(x^*, c) - p_S(x^*|c)Z)^2}{p_S(x^*|c)}\end{aligned}$$

Based on the above derivation, it is obvious that we obtain a zero-variance estimator when we choose $p_S(x^*|c) = p_G(x^*|c)\mathcal{L}_D(x^*, c)/Z$ as Equation (1). This theoretically optimal form provides us insights into designing the above sampling scheme. □

4.1.2 Two Implementations of Hardness Prediction

We design two variants of the sampling head. The first one is to explicitly estimate the discriminator loss (HP_{Loss}). The second method is to approximate the expected sampling distribution (HP_{Dist}).

HP_{Loss} guides the generator to learn the probability predicted by the discriminator that the sampled token x' is an original token. In this case, the output layer of the sampling head is actually a $\mathrm{sigmoid}$ function same as the discriminator:

$$\hat{D}(x', c) = \mathrm{sigmoid}(\boldsymbol{w}(x') \cdot \boldsymbol{h}_S(c))$$

where $\boldsymbol{h}_S(c)$ denotes the contextual representations projected by the sampling head, and $\boldsymbol{w}$ denotes the projection parameters. Then the loss of the sampling head at the masked position is:

$$\mathcal{L}_S(x', c) = (\hat{D}(x', c) - D(x', c))^2$$

When sampling replacements over the vocabulary, the estimated discriminator probability $\hat{D}(x', c)$ can be easily rewritten to the estimated discriminator loss $\hat{\mathcal{L}}_D(x', c)$:

$$\hat{\mathcal{L}}_D(x', c) = \begin{cases} -\log \hat{D}(x', c) & x' = x \\ -\log(1 - \hat{D}(x', c)) & x' \neq x \end{cases}$$

Multiplying the MLM probability factor p_G, we obtain the sampling distribution:

$$\begin{aligned}p_S(x'|c) &= \frac{p_G(x'|c)\hat{\mathcal{L}}_D(x', c)}{\sum_{x^*\in\text{vocab}} p_G(x^*|c)\hat{\mathcal{L}}_D(x^*, c)}\\ &= \frac{p_G(x'|c)\hat{\mathcal{L}}_D(x', c)}{E_{p_G(x^*|c)}[\hat{\mathcal{L}}_D(x^*, c)]}\end{aligned}$$

HP_{Dist} aims to directly approximate the expected sampling distribution as in Equation (1), instead of the discriminator loss. In this case, the sampling head produces an output probability of the token x' with a $\mathrm{softmax}$ layer:

$$p_S(x'|c) = \frac{\exp(\boldsymbol{e}(x') \cdot \boldsymbol{h}_S(c))}{\sum_{x^*\in\text{vocab}} \exp(\boldsymbol{e}(x^*) \cdot \boldsymbol{h}_S(c))} \quad (2)$$

where $\boldsymbol{e}$ represents the token embeddings. For the sampled token x', we define the loss of the sampling head as:

$$\mathcal{L}_S(x', c) = -\frac{p_G(x'|c)}{p_S(x'|c)}\mathcal{L}_D(x', c)\log p_S(x'|c)$$

Then we show that minimizing the above loss $\mathcal{L}_S(x', c)$ pushes sampling distribution of Equation (2) to our goal. Specifically, the loss expectation over the whole vocabulary is:

$$\begin{aligned}
&\mathrm{E}_{p_{\mathrm{S}}(x^*|c)}[\mathcal{L}_S(x^*, c)] = \\
&-\sum_{x^* \in \text{vocab}} p_{\mathrm{S}}(x^*|c)\frac{p_{\mathrm{G}}(x^*|c)}{p_{\mathrm{S}}(x^*|c)}\mathcal{L}_D(x^*, c)\log p_{\mathrm{S}}(x^*|c) \\
&= -\sum_{x^* \in \text{vocab}} p_{\mathrm{G}}(x^*|c)\mathcal{L}_D(x^*, c) \log p_{\mathrm{S}}(x^*|c)
\end{aligned}$$

According to the Lagrange Multiplier method, the optimal solution $\tilde{p}_{\mathrm{S}}$ of the loss function $\mathcal{L}_S(x', c)$ is consistent with Equation (1):

$$\begin{aligned}
\tilde{p}_{\mathrm{S}}(x'|c) &= \frac{p_{\mathrm{G}}(x'|c)\mathcal{L}_D(x', c)}{\sum_{x^* \in \text{vocab}} p_{\mathrm{G}}(x^*|c)\mathcal{L}_D(x^*, c)} \\
&= \frac{p_{\mathrm{G}}(x'|c)\mathcal{L}_D(x', c)}{\mathrm{E}_{p_{\mathrm{G}}(x^*|c)}[\mathcal{L}_D(x^*, c)]}
\end{aligned}$$

4.2 Sampling Smoothing

Along with the learning process, the masked language modeling tends to achieve relatively high accuracy. As a consequence, the generator oversamples the correct tokens as replacements, which renders the discriminator learning inefficient.

In order to address the issue, we apply an alternative loss function called focal loss (Lin et al., 2017) for MLM of the generator. Compared with the vanilla cross-entropy loss, focal loss adds a modulating factor for the weighting purpose:

$$\mathcal{L}_G^{\mathrm{fc}}(x, c) = -(1 - p_{\mathrm{G}}(x|c))^\gamma \log p_{\mathrm{G}}(x|c)$$

where $\gamma \geq 0$ is a tunable hyperparameter. Besides using a constant γ, we try the piecewise function $\gamma = \mathbb{1}(p_{\mathrm{G}} > 0.2) * 3 + \mathbb{1}(p_{\mathrm{G}} \leq 0.2) * 5$ in our experiments as suggested by Mukhoti et al. (2020).

In other words, the focal loss is used to adaptively down-weight the well-classified easy examples and thus focusing on more difficult ones. When applying the focal loss to the MLM head for the generator, we notice that if a token is easy for the generator to be predicted correctly, i.e., $p_{\mathrm{G}}(x|c) \rightarrow 1$, the modulating factor is greatly decreased. In contrast, if a token is hard to predict, the focal loss approximates to the original cross entropy loss. Therefore, we propose to employ the focal loss in order to smooth the sampling distribution, which in turn relieves oversampling correct tokens as replacements.

4.3 Pre-Training Objective

Adopting the above two strategies, we jointly train the generator and the discriminator together as the original ELECTRA model. The word embeddings of them are still tied during the pre-training stage. Formally, we minimize the combined loss over a large corpus $\mathcal{X}$:

$$\begin{aligned}
&\min_{\theta_G, \theta_S, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \Big(\mathop{\mathbb{E}}_{i \in \boldsymbol{m}} [\mathcal{L}_G^{\mathrm{fc}, \theta_G}(x_i, \boldsymbol{c})] + \\
&\lambda_1 \mathop{\mathbb{E}}_{i \in \boldsymbol{m}} [\mathcal{L}_S^{\theta_S}(x_i^{\mathrm{R}}, \boldsymbol{c})] + \lambda_2 \mathop{\mathbb{E}}_{t \in [1,n]} [\mathcal{L}_D^{\theta_D}(x_t^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{R}})] \Big)
\end{aligned}$$

where λ_1, λ_2 are two hyperparameters to adjust three parts of the loss. We only search λ_1 value and keep $\lambda_2 = 50$ for the fair comparison with ELECTRA. After pre-training, we throw out the generator and only fine-tune the discriminator on the downstream tasks.

5 Experiments

5.1 Setup

We implement ELECTRA+HP$_{\mathrm{Loss}}$/HP$_{\mathrm{Dist}}$+Focal on both the *small-size* setting and the *base-size* setting. The two prediction heads share both the generator and the token embeddings, which avoids the unnecessary increase in model complexity. We follow most settings as suggested in ELECTRA (Clark et al., 2020a). In order to enhance the ELECTRA baseline for a solid comparison, we add the relative position (Raffel et al., 2020). Experimental results show that our methods can improve performance even on the enhanced ELECTRA baseline.

We pretrain our models on the same text corpus as ELECTRA, which is a combination of English Wikipedia and BooksCorpus (Zhu et al., 2015). We also adopt the N-gram masking strategy which is beneficial for MLM tasks. The models are trained for 1M steps for small-size models and 765k steps for base-size models, so that the computation consumption can be similar to baseline models (Clark et al., 2020a). The base-size models are pretrained with 16 V100 GPUs less than five days. The small-size models are pretrained with 8 V100 GPUs less than three days. We use the Adam (Kingma and Ba, 2015) optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) with learning rate of 1e-4. The value of λ_2 in the training objective is kept fixed at 50 for a fair comparison with ELECTRA. For HP$_{\mathrm{Loss}}$, we search λ_1 in $\{5, 10, 20\}$, the best one is 5. For HP$_{\mathrm{Dist}}$, we keep $\lambda_1 = 1$. We search the focal loss weight γ in $\{1, 4\}$

Model	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS
	-m/-mm	Acc	Acc	Acc	Acc	Acc	MCC	PCC
Small-size								
ELECTRA (*reimplementation*)	80.3/80.6	89.0	89.2	62.2	89.0	**87.3**	59.6	86.3
ELECTRA+HP$_{\text{Loss}}$+Focal	**81.2**/81.7	89.1	**89.6**	**66.1**	**89.2**	86.6	59.3	86.7
ELECTRA+HP$_{\text{Dist}}$+Focal	81.0/**81.8**	**89.3**	**89.6**	62.5	**89.2**	86.8	**59.7**	**86.8**
Base-size								
BERT (Devlin et al., 2019)	84.5/-	88.6	90.8	68.5	92.8	86.0	58.4	87.8
RoBERTa (Liu et al., 2019)	84.7/-	-	-	-	92.7	-	-	-
XLNet (Yang et al., 2019)	85.8/-	-	-	-	**93.4**	-	-	-
ELECTRA (Clark et al., 2020a)	86.2/-	92.4	90.9	76.3	92.4	87.9	65.8	89.1
ELECTRIC (Clark et al., 2020b)	85.7/-	92.1	90.6	73.4	91.9	88.0	61.8	89.4
ELECTRA (*reimplementation*)	86.7/86.5	92.6	91.4	80.4	92.6	89.1	66.5	**91.0**
ELECTRA+HP$_{\text{Loss}}$+Focal	**87.0/86.9**	**92.7**	**91.7**	**81.3**	92.6	**90.7**	66.7	**91.0**
ELECTRA+HP$_{\text{Dist}}$+Focal	86.8/86.8	92.3	91.6	80.0	92.7	89.8	**67.3**	90.9

Table 1: Comparisons between our models and previous pretrained models on GLUE dev set. Reported results are medians over five random seeds.

on both the base-size and small-size model, the best configuration is $\gamma = 1$. The detailed pre-training configurations are provided in the supplemental materials.

5.2 Results on GLUE Benchmark

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) is a collection of diverse natural language understanding (NLU) tasks, including inference tasks (MNLI, QNLI, RTE; Dagan et al. 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009; Williams et al. 2018; Rajpurkar et al. 2016b), similarity and paraphrase tasks (MRPC, QQP, STS-B; Dolan and Brockett 2005; Cer et al. 2017), and single-sentence tasks (CoLA, SST-2; Warstadt et al. 2018; Socher et al. 2013). The detailed descriptions of GLUE datasets are provided in the supplementary materials. The evaluation metrics are Spearman correlation for STS-B, Matthews correlation for CoLA, and accuracy for the other GLUE tasks.

For small-size settings, we use the hyperparameter configuration as suggested in (Clark et al., 2020a). For base-size settings, we consider a limited hyperparameter searching for each task, with learning rates ∈ {5e-5, 1e-4, 1.5e-4} and training epochs $\in \{3, 4, 5\}$. The remaining hyperparameters are the same as ELECTRA. We report the median performance on the dev set over five different random seeds for each task. All the results come from the single-task fine-tuning. For more detailed fine-tuning configurations, please refer to the supplementary materials.

Results are shown in Table 1. With the same configuration and pre-training data, for both the small-size and the base-size, our methods outperform the strong reimplemented ELECTRA baseline by 0.6 and 0.4 on average respectively. For the most widely reported task MNLI, our models achieve 87.0/86.9 points on the matched/mismatched set, which obtains 0.3/0.4 absolute improvements. The performance gains on the small-size models are more obvious than the base-size models, we speculate that is due to the learning of the small-size generator is more insufficient and suffers from the above issues more significantly. The results demonstrate that our proposed methods can improve the pre-training of ELECTRA. In other words, sampling more hard replacements is more efficient than the original masked language modeling.

5.3 Results on SQuAD 2.0

The Stanford Question Answering Dataset (SQuAD; Rajpurkar et al. 2016a) is a reading comprehension dataset, each example consists of a context and a question-answer pair. Given a context and a question, the task is to answer the question by extracting the relevant span from the context. We only use the version 2.0 for evaluation, where some questions are not answerable. We report the results of both the Exact-Match (EM) and F1 score. When fine-tuning on SQuAD, we add the question-answering module from XLNet on the top of the discriminator as Clark et al.

Model	SQuAD 2.0	
	EM	F1
Small-size		
ELECTRA (*reimplementation*)	68.9	71.3
ELECTRA+HP$_{\text{Loss}}$+Focal	71.8	74.3
ELECTRA+HP$_{\text{Dist}}$+Focal	**73.2**	**75.5**
Base-size		
BERT (Devlin et al., 2019)	73.7	77.1
RoBERTa (Liu et al., 2019)	-	79.7
XLNet (Yang et al., 2019)	78.2	81.0
ELECTRA (Clark et al., 2020a)	80.5	83.3
ELECTRIC (Clark et al., 2020b)	80.1	-
ELECTRA (*reimplementation*)	82.4	85.1
ELECTRA+HP$_{\text{Loss}}$+Focal	**83.0**	**85.6**
ELECTRA+HP$_{\text{Dist}}$+Focal	82.7	85.4

Table 2: Comparisons between our models and previous pretrained models on SQuAD 2.0 dev set. Reported results are medians over five random seeds.

Model	MNLI-m	SQuAD 2.0
ELECTRA	80.3	71.3
ELECTRA + *HP*$_{\text{Loss}}$		
$\lambda_1 = 5$	80.9	74.2
$\lambda_1 = 10$	81.1	75.1
$\lambda_1 = 20$	81.0	74.1
ELECTRA + *HP*$_{\text{Loss}}$ *(*$\lambda_1 = 5$*) + Focal*		
$\gamma = \begin{cases} 3 & p_G > 0.2 \\ 5 & p_G \leq 0.2 \end{cases}$	81.0	74.7
$\gamma = 1.0$	81.2	74.3
$\gamma = 4.0$	80.9	74.2

Table 3: Ablation studies on small-size models. We analyze the effect of the hardness prediction loss weight λ_1 and the focal loss factor γ. Reported results are medians over five random seeds.

(2020a). All the hyperparameter configurations are the same as ELECTRA. We report the median performance on the dev set over five different random seeds. Refer to the appendix for more details about fine-tuning.

Results on SQuAD 2.0 are shown in Table 2. Consistently, our models perform better than ELECTRA baseline under both the small-size setting and the base-size setting. Under the base setting, our models improve the performance over the reimplemented ELECTRA baseline by 0.6 points (EM) and 0.5 points (F1). Especially under the small setting, our models outperform the baseline by a remarkable margin. ELECTRA+HP$_{\text{Dist}}$+Focal obtains 4.3 and 4.2 points absolute improvements on EM and F1 metric.

5.4 Ablation Studies

We conduct ablation studies on small-size ELECTRA+HP$_{\text{Loss}}$+Focal models. We investigate the effect of the loss weight λ_1 of the sampling head and the focal loss factor γ in order to better understand their relative importance. Results are presented in Table 3.

We first disable the focal loss and only understand the effect of λ_1. As shown in Table 3, no matter what the value of λ_1 is, our models exceed the baseline by a substantial margin, which demonstrates that the hardness prediction can indeed improve the pre-training and our methods are not sensitive to the loss weight hyperparameter. Next, we fix λ_1 at 5 and understand the effect of the focal loss factor γ. We observe that the application of the focal loss with piecewise $\gamma = \mathbb{1}(p_G > 0.2) * 3 + \mathbb{1}(p_G \leq 0.2) * 5$ and $\gamma = 1$ can improve the performance on two datasets, which proves the effectiveness of the sampling smoothing.

6 Analysis

To better understand the main advantages of our models over ELECTRA, we conduct several analysis experiments.

6.1 Impacts on Sampling Distributions

We first provide a comparison between the sampling distributions of ELECTRA and our models illustrate the effect of our proposed methods. We conduct evaluations on a subset of the pre-training corpus. Figure 3 demonstrates the distribution of the maximum probability of the two sampling distributions at the masked positions. We observe that the ratio of the maximum value under ELECTRA sampling distribution between $[0.9, 1]$ is much higher than that of our models. In other words, the original distribution suffers from over-sampling the high-probability tokens and the discriminator is forced to learn from these easy examples repeatedly. In contrast, the distribution of the maximum value of our models in each interval is relatively more uniform than ELECTRA, which indicates that our methods can significantly reduce the probability of sampling the well-classified tokens and smooth the

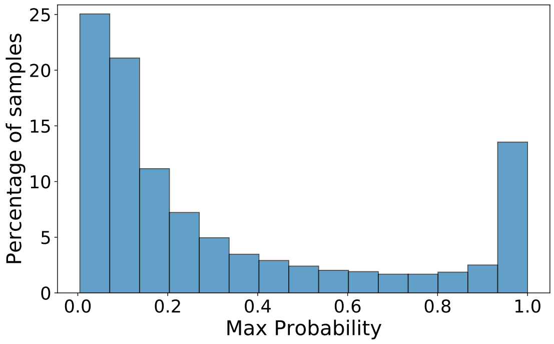

(a) Sampling Distribution of ELECTRA

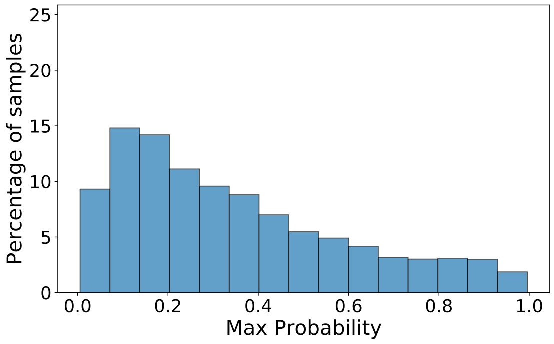

(b) Sampling distribution of our models

Figure 3: The distribution of the maximum probability at the masked positions under the sampling distributions of ELECTRA and our models.

Token Type	Original	Replaced	All
Corr. Coeff.	0.78	0.61	0.64

Table 4: Correlation coefficient between the actual discriminator loss and the estimated value for ELECTRA+HP$_{\text{Loss}}$+Focal. “Original”: sampling correct tokens as replacements. “Replaced”: the positions that are substituted to incorrect tokens.

whole sampling distribution.

6.2 Estimation Quality

In order to measure the estimation quality of discriminator loss, we evaluate our models on a held-out set of pre-training corpus and compute the correlation coefficient between the actual discriminator loss $\mathcal{L}_D(x, c)$ and the estimated value $\hat{\mathcal{L}}_D(x, c)$. The results of ELECTRA+HP$_{\text{Loss}}$+Focal are shown in Table 4. We report the estimation quality of the original tokens and the replaced tokens separately. The correlation coefficient value is 0.64 over two types of tokens, which proves that $\mathcal{L}_D(x, c)$ and $\hat{\mathcal{L}}_D(x, c)$ correlate well. Furthermore, we observe that the estimation quality over the original tokens is relatively higher than the replacements. We speculate that the sampling probability of the original tokens is generally higher than the replacements, so the sampling head tends to receive more feedback from these original tokens.

Model	**Masked Positions**	**All Positions**
ELECTRA	0.81	0.96
Ours	0.72	0.95

Table 5: Replacement detection accuracy of ELECTRA and ELECTRA+HP$_{\text{Loss}}$+Focal. The models are evaluated on 15% masked positions and all input tokens respectively. Our method samples more hard examples.

6.3 Prediction Accuracy of the Discriminator

In order to verify the claim that the sampling distribution of our models indeed considers the detection difficulty, we evaluate the prediction accuracy of the discriminator under the two sampling schemes of ELECTRA and ELECTRA+HP$_{\text{Loss}}$+Focal. Results are listed in Table 5. No matter evaluating at all positions or only at the masked positions, the detection accuracy under our sampling distribution is relatively lower than under masked language modeling in original ELECTRA. Because the unmasked tokens constitute the majority of input examples, the difference of the all-token accuracy between two models is not so distinct compared to the masked tokens. This phenomenon is consistent with our original intention. It proves that our models can sample more replacements that the discriminator struggles to make correct predictions. In contrast, the replacements sampled from ELECTRA are easier to distinguish.

7 Conclusion

We propose to improve the replacement sampling for ELECTRA pre-training. We introduce two methods, namely hardness prediction and sampling smoothing. Rather than sampling from masked language modeling, we design a new sampling scheme, which considers both the MLM probability and the prediction difficulty of the discriminator. So the generator can receive feedback from the discriminator. Moreover, we adopt the focal loss to MLM, which adaptively downweights the well-classified examples and smooth the entire distribution. The sampling smoothing technique relieves oversampling original tokens as replacements. Results show that our models outperform ELECTRA baseline. In the future, we would like to apply our strategies to other pre-training frameworks and cross-lingual models. Moreover, we are exploring how to integrate the findings and insights of the proposed method into the masked language modeling task, which seems also quite promising.

Acknowledgements

The work was partially supported by National Natural Science Foundation of China (NSFC) [Grant No. 61421003].

References

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 642–652. PMLR.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020a. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 285–294. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *CoRR*, abs/1406.2661.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007. IEEE Computer Society.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip H. S. Torr, and Puneet K. Dokania. 2020. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122, New Orleans, Louisiana.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

A GLUE Details

GLUE (Wang et al., 2019) is a collection of various natural language understanding (NLU) tasks, including inference tasks (MNLI, QNLI and RTE), similarity and paraphrase tasks (MRPC, QQP and STS-B), and single-sentence tasks (CoLA and SST-2).

MNLI The Multi-Genre Natural Language Inference Corpus (Williams et al., 2018) is a crowd-sourced collection of sentence pairs with textual entailment annotations. Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral). The dataset contains 393k train examples drawn from ten different sources.

QNLI The Stanford Question Answering Dataset (Rajpurkar et al., 2016b) is a question-answering dataset consisting of question-paragraph pairs. The task is to predict whether a context sentence contains the answer to a question sentence. The dataset contains 108k train examples from Wikipedia.

QQP The Quora Question Pairs dataset is a collection of question pairs from the community question-answering website Quora. The task is to determine whether a pair of questions are semantically equivalent. The dataset contains 364k train examples.

RTE The Recognizing Textual Entailment datasets (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) come from a series of annual textual entailment challenges. Given a premise sentence and a

Hyperparameter	Value
Learning Rate	3e-4 for Small, {5e-5, 1e-4, 1.5e-4} for Base
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Layerwise LR decay	0.8
Learning rate decay	Linear
Warmup fraction	0.1
Attention Dropout	0.1
Dropout	0.1
Weight Decay	0 for Small, {0, 0.01} for Base
Batch Size	32
Train Epochs	{10, 15, 20} for RTE, {3, 4, 5} for other tasks

Table 6: Fine-tuning details about ELECTRA baseline and our models.

Hyperparameter	Small	Base
Number of layers	12	12
Hidden Size	256	768
FFN inner hidden size	1024	3072
Attention heads	4	12
Attention head size	64	64
Embedding Size	128	768
Generator Size	1/4	1/3
Mask percent	15	15
Learning Rate Decay	Linear	Linear
Warmup steps	10000	10000
Learning Rate	5e-4	1e-4
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999

Table 7: Pre-training details about ELECTRA baseline and our models.

hypothesis sentence, the task is to predict whether the premise entails the hypothesis or not. The dataset contains 2.5k train examples from a series of annual textual entailment challenges.

SST The Stanford Sentiment Treebank (Socher et al., 2013) consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence. The dataset contains 67k train examples.

MRPC The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a corpus of sentence pairs automatically extracted from online news sources. The task is to predict whether two sentences are semantically equivalent or not. The dataset contains 3.7k train examples.

CoLA The Corpus of Linguistic Acceptability (Warstadt et al., 2018) consists of English acceptability judgments drawn from books and journal articles on linguistic theory. The task is to determine whether a given sentence is grammatical or not. The dataset contains 8.5k train examples.

STS-B The Semantic Textual Similarity Benchmark (Cer et al., 2017) is a collection of sentence pairs drawn from news headlines, video and image captions, and natural language inference data. The tasks is to predict how semantically similar two sentences are on a 1-5 scale. The dataset contains 5.8k train examples.

B Pre-training Details

We did not search any hyperparameters during pre-training. Most of our pre-training configurations are same as the original ELECTRA (Clark et al., 2020a). The learning rate for base-sized model is changed from 2e-5 to 1e-5 on both ELECTRA baseline and our models, because we expect the fair comparison with BERT and RoBERTa. We keep $\lambda_2 = 50$ for both ELECTRA-EL and ELECTRA-AD. The full set of pre-training hyperparameters is provided in Table 7.

C Fine-tuning Details

For base-sized models, we searched the learning rate and pre-training epochs on both ELECTRA baseline and our models. For small-sized models, we use the same hyperparameters as suggested in

ELECTRA. All the results come from the single-task fine-tuning. The full set of fine-tuning hyperparameters is provided in Table 6.

D Prediction Accuracy of the Discriminator

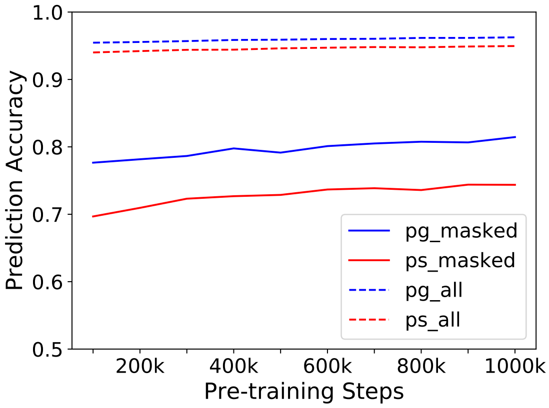

Figure 4: The prediction accuracy of the discriminator. Blue lines indicate sampling replacements according to p_{G}, red lines are according to p_{S}. The solid line represents the prediction accuracy evaluated on the 15% masked tokens and the dashed line represents the prediction accuracy of all input tokens.

Reordering Examples Helps during Priming-based Few-Shot Learning

Sawan Kumar
Indian Institute of Science, Bangalore
sawankumar@iisc.ac.in

Partha Talukdar
Indian Institute of Science, Bangalore
ppt@iisc.ac.in

Abstract

The ability to learn from limited data, or few-shot learning, is a desirable and often critical requirement for NLP systems. While many existing methods do poorly at learning from a handful of examples, large pretrained language models have recently been shown to be efficient few-shot learners. One approach to few-shot learning, which does not require finetuning of model parameters, is to augment the language model's input with priming text which is typically constructed using task specific descriptions and examples. In this work, we further explore priming-based few-shot learning, with focus on using examples as prompts. We show that presenting examples in the right order is key for generalization. We introduce PERO (Prompting with Examples in the Right Order), where we formulate few-shot learning as search over the set of permutations of the training examples. We show that PERO can learn to generalize efficiently using as few as 10 examples, in contrast to existing approaches. While the newline token is a natural choice for separating the examples in the prompt, we show that learning a new separator token can potentially provide further gains in performance. We demonstrate the effectiveness of the proposed method on the tasks of sentiment classification, natural language inference and fact retrieval. Finally, we analyze the learned prompts to reveal novel insights, including the idea that two training examples in the right order alone can provide competitive performance for sentiment classification and natural language inference.

1 Introduction

The ability to learn from a few examples, or few-shot learning, as generally understood to be possessed by humans, is a desirable property for Natural Language Processing (NLP) systems as well. It is critical in scenarios where collecting large amounts of data is expensive. It is also important to enable a personalized Artificial Intelligence (AI) experience, where a single user is expected to use an AI agent to perform a task demonstrated through a handful of examples.[1]

Pretrained language models (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020) have recently been shown to be exceedingly good at several benchmark NLP tasks (Wang et al., 2018, 2019). Traditionally the parameters of these language models have been finetuned on task specific datasets to achieve the aforementioned performance gains, often requiring large amounts of data. Brown et al. (2020) show that large pretrained language models (GPT3) are also efficient few-shot learners. Few-shot learning is achieved using task descriptions and labeled examples as prompts. Remarkably, with this priming-based approach and without needing any parameter updates, GPT3 often performs comparable to traditional finetuning-based supervised systems which use much larger datasets. One could argue that the task performance achieved in the priming-based approach measures what the pretrained language model has already learned. Shin et al. (2020), operating in the same setting, use automatically generated prompts to measure task specific knowledge in a pretrained language model.

In this work, we further explore priming-based few-shot learning, while focusing on using examples as prompts. The training objective for a language model is typically the prediction of a token given a context. There is no clear incentive to treat a sequence of sentences in the context as equal and conveying examples of a concept. As a result, one could expect certain order of examples when used as a prompt to be more favorable at providing task

---

[1] See the Introduction section of Brown et al. (2020) for a discussion on further difficulties, relevant to the setting we consider in this work, regarding the need of a large dataset for every new task.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4507–4518
August 1–6, 2021.

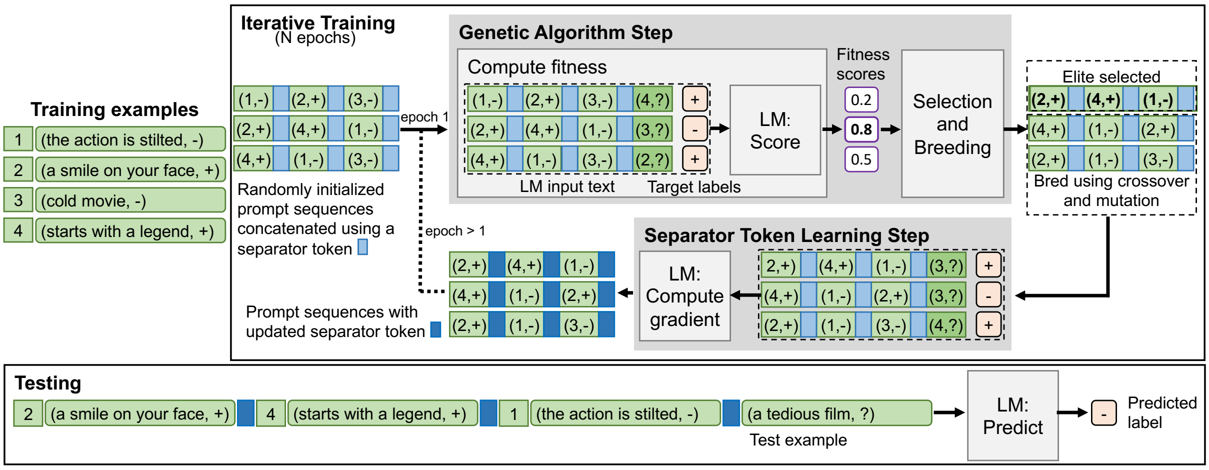

Figure 1: *Overview of PERO*: Given a set of training examples, PERO searches over permutations of training examples using a genetic algorithm (Section 4.2), and optionally also learns a separator token (Section 4.3) which is used to concatenate the training examples. Briefly, starting from a set of randomly initialized permutations, the genetic algorithm step computes the fitness of each permutation for making predictions using a pretrained LM. These fitness scores are then used for selection and subsequent breeding of new permutations using biologically inspired operations of mutation and crossover. The separator token learning step uses the updated set of permutations and uses gradient updates to improve the separator token. The two steps are performed iteratively for a fixed number of epochs and the best permutation and separator token are selected using a validation set. Please see Section 4 for details.

specific cues.

We propose PERO[2] (Prompting with Examples in the Right Order), where we formulate the problem of few-shot learning as search over permutations of training examples. We find that choosing the right permutation is key to getting good task performance. In PERO, we use a genetic algorithm (Mitchell, 1998) to search over possible permutations of training examples. The selected examples are used for prompting publicly available pretrained language models (Devlin et al., 2019; Liu et al., 2019). We find that with as few as 10 examples, PERO can learn to generalize efficiently, in contrast to existing approaches. When concatenating examples to use as a prompt, the newline token is a natural choice as a separator token. We show that using a learned separator token can potentially provide further gains in performance. We evaluate the performance of PERO on the tasks of sentiment analysis, Natural Language Inference (NLI) and fact retrieval.

Finally, our analysis of the learned prompts (Section 5.5) leads to novel insights about few-shot learning using textual prompts. For instance, using only two examples, repeated and ordered using a learned label pattern, can provide performance comparable to and even exceeding the performance of existing few-shot baselines which use thousands of examples.

In summary, we make the following contributions:

1. We propose PERO, where we formulate the problem of few-shot learning as search over permutations of training examples, and optionally a separator token. As we don't update the parameters of the underlying language model, PERO serves as a probe for measuring task specific knowledge in pretrained language models.

2. We demonstrate the effectiveness of PERO over a recent baseline on the tasks of sentiment analysis, NLI and fact retrieval.

3. We analyze the learned prompts and provide novel insights about textual prompts that can lead to good task performance in the low-data regime. In particular, we provide an effective recipe for one-shot learning.

We have released the source code of PERO to aid reproducibility of the results.

---

[2] PERO source code is available at `https://github.com/SawanKumar28/pero`

Task	Template	Examples
Sentiment Classification	[Example Text] Answer: [Label Text]	(1) goes to absurd length Answer: false (2) A girl in white is dancing Answer: true
NLI	"[Premise Text]" implies "[Hypothesis text]" Answer: [Label Text]	(1) "Men are sawing logs" implies "Men are cutting wood" Answer: true (2) "There is no girl in white dancing" implies "A girl in white is dancing" Answer: false
Fact retrieval	[Subj] is located in [Obj] [Subj] is a subclass of [Obj]	(1) Directors Lounge is located in Berlin (2) gingerbread is a subclass of cookie

Table 1: *Formatting used to create textual inputs* for the tasks considered in this work. For sentiment classification, positive and negative sentiments correspond to the label text of *true* and *false* respectively. For NLI, entailment and contradiction labels correspond to the label text of true and false respectively.

2 Related Work

Pretrained Language Models using a transformer architecture (Vaswani et al., 2017) on large unsupervised corpora have recently been found to be efficient at learning downstream tasks, providing significant gains over existing standalone supervised systems, on a variety of NLP tasks (Wang et al., 2018, 2019). There have been two major approaches to learning language models: causal language models (CLM) and masked language models (MLM). CLMs (Radford et al., 2018, 2019; Brown et al., 2020) are typically trained by requiring a language model to predict the next token given a textual context. Masked language models (Devlin et al., 2019; Liu et al., 2019) on the other hand are trained by masking out a certain number of tokens in a textual context and requiring the language model to predict the masked out tokens. Typically, the parameters of the language model are then fine-tuned using task-specific training examples. For our experiments, we leverage publicly available pretrained masked language models (Devlin et al., 2019; Liu et al., 2019).

Few-shot learning using language models is a desirable and perhaps even an expected property of large pretrained language models, given the large amounts of data they are typically trained with. Brown et al. (2020) show that scaling up language models leads to improved few-shot learning, with their best model, GPT3, being able to achieve performance comparable to existing supervised systems, while using much fewer examples. Zero-shot and few-shot learning are achieved without needing parameter updates to the model but instead by prompting the language model with task specific description and task specific examples. In this work, we study the impact of the order in which examples are presented in a prompt and show that searching over them can lead to significant gains in few-shot performance, without needing updates to the model parameters.[3]

Measuring task performance of language models without any parameter updates can be seen as a measure of the knowledge (either descriptive, or procedural) that is already contained in the pretrained language model.

Probing knowledge contained in language models has been of interest, given the success of these models. Probing methods rely on creating cloze-style manual prompts (Petroni et al., 2019), or mining efficient natural language prompts (Jiang et al., 2020). Shin et al. (2020) rely on training examples to learn trigger tokens which when used as a prompt demonstrate the ability of language models to do sentiment analysis and NLI along with knowledge based completion, without needing any parameter updates. The learned trigger tokens however aren't very meaningful leading to difficulty in interpreting these results. In this work, we instead focus on using natural language training examples as prompts. While being more interpretable, the prompts used in this work lead to significant gains in performance in the low-data regime.

3 Background: Genetic Algorithm

A genetic algorithm (Mitchell, 1998) is a search heuristic inspired by the biological process of natural selection. Briefly, it evolves a population of candidate solutions towards increasing fitness to an objective through biologically inspired operations such as selection, crossover and mutation. We now describe the key terminology:

---

[3] Note that the scope of this work is distinct from meta-learning approaches (Hospedales et al., 2020), where the goal is improve the learning algorithm using several learning episodes. In contrast, we only assume a pretrained language model and a few examples of the concept we are interested in.

Individual A single candidate solution, c, usually represented through a binary code but extensible to other types of codes. Generally, we will let c be denoted by the sequence of k integers $c = (s^1 s^2 ... s^k)$.

Population A set of individuals of size N_P, $P = \{c_i, i \in [N_P]\}$.

Fitness The measure of goodness for an individual for the task, $F(c_i)$.

Selection An operator to select fit individuals in a population which will be used to generate new individuals, through crossover and mutation. Better fitness leads to higher likelihood of selection.

Crossover An operator which typically takes two individuals c_1 and c_2 as inputs to produce new individuals d_1 and d_2, by combining subsequences from the two inputs. For example, consider two input sequences:

$$c_1 = (c_1(1)c_1(2)c_1(3)c_1(4))$$
$$c_2 = (c_2(1)c_2(2)c_2(3)c_2(4))$$

A single point crossover after the second position would lead to the individuals:

$$d_1 = (c_1(1)c_1(2)c_2(3)c_2(4))$$
$$d_2 = (c_2(1)c_2(2)c_1(3)c_1(4))$$

Mutation An operator which randomly flips some elements in an input sequence. For example, with input $c = (c(1)c(2)c(3)c(4))$, a typical mutation operation would lead to the output $d = (c(1)c(2)c'(3)c(4))$, where $c'(3) \neq c(3)$. Usually, each position is randomly altered with a mutation probability p_m.

We now present the sketch of a typical genetic algorithm:

1. Initialize a set of individuals to form a population $P = \{c_i, i \in [N_P]\}$. Repeat the following steps for N_{epochs} iterations.
2. Compute fitness of each individual in the population, $F(c_i), i \in [N_P]$.
3. Using the computed fitness, select individuals which will be used to breed the next generation.
4. With pairs of selected individuals, generate new individuals using the crossover operation.
5. Mutate the generated individuals using the mutation operator, to create a new population P'.
6. Set $P = P'$ and go to step 2.

4 PERO: Proposed Method

The overall architecture employed in PERO is shown in Figure 1. We introduce the notation in Section 4.1. We discuss how we employ a genetic algorithm to search over permutations of training examples in Section 4.2. We then discuss how we augment the search heuristic to learn a task specific separator token in Section 4.3.

4.1 Notation and Input Format

For both classification and knowledge base completion tasks, we denote a textual task input by x and the gold label as y. We denote the pretrained masked language model with the operator L, which takes a sequence of input tokens to output a sequence of the same length containing token probabilities over the token vocabulary. With tokens $(t_1 t_2 ... t_n)$, $L((t_1 t_2 ... t_n)) = (p_1 p_2 ... p_n)$, where p_i denotes a vector of probabilities over all tokens in the vocabulary.

For all our experiments, the input to the language model is formatted with exactly one mask token.[4] For brevity, we denote by L_{Mask} the operator which outputs the token probability at the mask token position.

The training data is denoted by the set of examples $(x_i, y_i), i \in [N_{\text{train}}]$. We denote a permutation, or an ordered subset of size k of the training data, by $c = (c(1)c(2)...c(k))$, where $c(j) \in [N_{\text{train}}]$.

For all tasks, we create an input text sequence by concatenating k examples using a permutation c of training examples, along with a test example x_{test}: "Format($x_{c(1)}$,$y_{c(1)}$) <Separator> Format($x_{c(2)}$,$y_{c(2)}$) .. <Separator> Format($x_{c(k)}$,$y_{c(k)}$) <Separator> Format(x_{test},mask)", where Format(,) formats the example text and label for a task, and <Separator> is either the new line character, or is learned as described in Section 4.3. The formatting details are provided in Table 1. We attempt to

---

[4] The mask token we use is the same as the one employed in pretraining.

use task agnostic formats and textual labels for classification tasks to the extent possible.

4.2 Genetic Algorithm: Search over Permutations of Examples

We employ a genetic algorithm for searching over permutations of training examples (see Section 3 for a brief introduction to genetic algorithms). We present the overall architecture in Figure 1.

Here, we detail how the various components and operators of a genetic algorithm are defined for searching over permutations of examples:

Individual An individual is defined as a vector of unique training example indices $c = (c(1)c(2)...c(k))$, where $c(j) \in [N_{\text{train}}]$.

Population A set of individuals.

Fitness For a given permutation of training example indices, fitness is defined as the average cross entropy loss over training examples when evaluated as in Figure 1. The cross entropy loss is computed over the set of possible labels for classifications tasks, and over all tokens in the vocabulary for knowledge base completion tasks.

Note that during search, a training example may occur both in the prompt and as well as the test example. This is generally not a problem as we are not finetuning the model and do not run the risk of learning to copy. When also training the separator token (Section 4.3), we ensure that the test example doesn't occur in the prompt by dropping it from the prompt if required.

Selection For selection, we use elitism, i.e., at each generation of individuals, we retain a certain percentage (elite ratio) of top performing individuals without any modifications. The rest of the population is created through crossover and mutation over a percentage (selection size) of top performing individuals.

Crossover We perform a single point crossover, while ensuring that the resulting individuals contain unique indices. Given two parents c_1 and c_2, first a random number j is sampled in the range $[k]$, the length of the individuals, to use as the crossover point. We define an operator $\text{First}_s(v, v')$ which selects the first s elements in vector v which do not occur in vector v'. Similarly, $\text{Last}_s(v, v')$ picks the last s elements in v which do not occur in vector v'. Denoting the subvector $c(i)c(i+1)...c(j)$ by $c^{i:j}$, four new individuals are then created:

$$d_1 = (c_1^{1:j}\text{Last}_{k-j}(c_2, c_1^{1:j}))$$
$$d_2 = (c_2^{1:j}\text{Last}_{k-j}(c_1, c_2^{1:j}))$$
$$d_3 = (\text{First}_j(c_2, c_1^{j+1:k})c_1^{j+1:k})$$
$$d_4 = (\text{First}_j(c_1, c_2^{j+1:k})c_2^{j+1:k})$$

This modification over a straightforward crossover ensures that the resulting individuals contain unique indices.

Mutation We perform mutation on an input candidate by changing each position with a mutation probability p_m. When changed, an index is replaced by a random choice from the other training examples. If the new index is already present in the input candidate, the value at that index is swapped with the selected index.

The Genetic algorithm is run for N_{epochs} (see Section 3 for the training flow). A validation set of the same size as the train set was used to select from the best performing individuals in each epoch.

4.3 Separator Token Learning

In addition to the search over permutations of training examples as described in the previous section, we optionally learn a separator token to concatenate the examples (see Figure 1).

We initialize a token embedding parameter with the token embedding of the newline character. At the end of each epoch of the genetic algorithm, we use gradient updates to estimate the token embedding. The training set is created using the individuals (prompts) in the population in the current generation, and replacing the answer of the final example with the mask token. Gradient updates are then done by requiring the model to predict the correct answer.

5 Experiments

In this section, we aim to answer the following questions:

Q1 How does PERO compare with existing approaches on task performance? (Section 5.3)

Q2 How do the components of PERO, namely genetic algorithm and separator token learning affect task performance? (Section 5.4)

Training	Prompt Type	P@1
Manual	LAMA (Petroni et al., 2019)	31.1ˆ
Full supervision	LPAQA(top1) (Jiang et al., 2020)	34.1ˆ
	Autoprompt (Shin et al., 2020)	43.3ˆ
	PERO	46.6
10 examples	Autoprompt (Shin et al., 2020)	18.9
	PERO	40.3

Table 2: *Summary of fact retrieval experiments:* Precision @1 results on test sets are presented. ˆindicates replicated numbers. PERO improves over the baselines when using all training data (up to 1000 examples) and provides significant gains when using limited training data. This indicates that PERO is more efficient at eliciting knowledge from language models. Please see Section 5.3 for details and Appendix A.2.2 for more detailed results.

Q3 What aspects of PERO are important for getting good performance? (Section 5.5)

The experimental setup is described in Section 5.2, and the datasets are described in Section 5.1.

5.1 Datasets

Sentiment Classification: We use SST2 (Socher et al., 2013), a binary sentiment classification task. The training data contains 67350 training, 873 validation and 1822 test examples.

NLI: We use label-balanced 2-class NLI dataset created by Shin et al. (2020) using the SICK-E dataset (Marelli et al., 2014). This dataset has 1289 training, 143 validation and 1427 test examples.

Fact Retrieval: We use the train, validation, and test splits created by Shin et al. (2020) (referred to as ‘original’ in the paper) for 41 relations. For our experiments, we use the manual prompts created by Petroni et al. (2019). Please see Appendix A.2.2 for relation wise prompts and training statistics.

5.2 Experimental Setup

Number of training examples: For most of our experiments, we limit to a total of 10 training examples. We chose this number as prior work (Shin et al., 2020) faced difficulty in enabling predictions using only 10 training examples, usually performing close to random prediction. We create 5 sets of size 10, chosen successively from the first 50 training examples, and report on average task performance. Although our focus is few-shot learning in the low data regime, we also present results with more examples (the first 100 and the first 1000 examples) for reference. For model selection, we use a label-balanced validation set (chosen from the beginning of the corresponding validation set) of the same size as the training data. In all cases, and irrespective of the number of training examples, we keep the prompt size fixed to 10 examples.

Pretrained LM: We use RoBERTa-large (Liu et al., 2019) for all our experiments except for the fact retrieval task where we use the bert-large-cased model (Devlin et al., 2019) as this model has been shown to work better for the task (Shin et al., 2020). RoBERTa-large has 24 layers, with 16 attention heads and a hidden size of 1024 (355M parameters). Bert-large-cased uses the same architecture as RoBERTa-large. We use the implementation of transformer architectures provided by Wolf et al. (2020). We use the </s> token as the default separator token. When learning a new separator token, we initialize the token embedding by the token embedding of </s> token, and finetune the embedding as discussed in Section 4.3.

Genetic algorithm: We run the genetic algorithm for 100 epochs for classification tasks and 30 epochs for fact retrieval tasks. The population size was fixed to 100 and the mutation probability was set to 0.1. Elite ratio was set to 0.1, while the selection size was fixed to 25. When training a separator token embedding, the maximum number of training epochs for learning the embedding was set to 10 for classification tasks and 5 for fact retrieval tasks. Gradient updates were performed using the AdamW optimizer (Loshchilov and Hutter, 2018) with a learning rate of $1e-4$.

Baselines: We use Autoprompt (Shin et al., 2020) and the traditional finetuning approach as few-shot baselines. Please see Appendix A.1 for hyperparameter details.

5.3 Overall Results

In this section, we present the few-shot learning capability of PERO. For reference, we also report results when using more data.

We present fact retrieval results (Precision@1 scores) in Table 2. Relation wise results are provided in Appendix A.2.2. When using all training data, PERO improves the overall P@1, and is competitive or outperforms Autoprompt on all relations. When using only 10 training examples, PERO provides significant gains over Autoprompt

Number of training examples		10	100	1000
Sentiment Classification	Finetune	52.5 (2.36)	90.2	93.1
	Autoprompt	52.3 (2.60)	73.5	75.1
	PERO	**91.2** (1.83)	**93.8**	**94.2**
NLI	Finetune	57.4 (10.65)	**96.1**	**98.6**
	Autoprompt	58.6 (9.08)	76.2	86.5
	PERO	**81.3** (4.99)	78.5	83.2

Table 3: *Summary of classification tasks:* Test set label accuracies for PERO are presented for the tasks of sentiment classification and NLI. When using 10 examples, we also report the standard deviation across training splits. For both tasks, PERO provides significant gains over both Autoprompt and the traditional finetuning approach, when using 10 examples. For reference, we also present results when using more training examples. For both tasks, PERO is competitive with Autoprompt with increasing data. Overall, the results indicate that PERO is capable of learning to generalize with a handful of examples, in contrast to existing approaches. Please see Section 5.3 for details. Please see Appendix A.2.1 for additional comparison between Autoprompt and PERO using 10 examples across 100 training splits.

	Sentiment Classification	NLI
PERO	91.2	81.3
PERO-Sep learning	89.3	77.5

Table 4: *Impact of separator token learning Step:* Average test set label accuracies with and without the separator token learning are presented for training sets of size 10. The results indicate that the genetic algorithm step alone provides a strong baseline, while the separator token learning step provides further gains. Please see Section 5.4 for details.

on all relations. Overall, we show through PERO that simple manual prompts can be combined in relatively straightforward ways to create stronger probes while still being interpretable.[5]

We present the label accuracies of PERO for sentiment classification and NLI in Table 3. In each case, PERO is able to generalize well when using only 10 examples, while existing approaches perform close to random guess (50%). When using more data, PERO is competitive with Autoprompt for both tasks, while finetuning does better than PERO for NLI with larger training sizes. Overall, PERO provides an efficient approach to few-shot learning with pretrained language models.

Comparison when using larger training sizes: The results in Table 3 also suggest the use of finetuning when more data is available and the use of PERO when there isn't enough data for finetuning to generalize well. The relatively low performance of PERO with more data, especially for the NLI task, could be due to the much larger search space when using more training data. Since we keep the prompt size fixed to 10 examples, the search space is $10!$ for 10 training examples and $1000!/990!$ when using 1000 examples. While a better search strategy could potentially improve PERO's performance when using more data, we leave that as an interesting future work. Note, however, that the search space complexity is determined by the number of training examples irrespective of their labels. For example, PERO improves over the baselines on the fact retrieval task (Figure 2), despite a much larger number of labels.

For reference, we provide the label accuracies when using all available training data when using PERO, Autoprompt and finetuning respectively: 95.0, 91.4 and 96.7 for sentiment classification, and 79.5, 87.3 and 99.1 for NLI. When compared to the traditional fully supervised finetuning approach, PERO performs within 94.3% while using only 0.015% of the training data for sentiment classification, and within 82.1% while using only 0.77% of the training data for NLI.

5.4 Ablation on PERO's components

In this section, we present an ablation study to understand the role that the two components of PERO, namely genetic algorithm and separator token learning steps play. We present the label accuracies for sentiment classification and NLI with and without the separator token learning step (indicated as PERO-sep learning) in Table 4. The results indicate that the permutation search using the genetic algorithm step provides large gains by itself, while the separator token learning potentially improves it.

---

[5] Autoprompt's learned tokens are sometimes relevant to the task but generally hard to interpret.

	Sentiment Classification	NLI
Default fitness	89.3	77.5
Inverse fitness	78.3	58.8

Table 5: *Searching for bad permutations:* We use the genetic algorithm described in Section 4 to search for bad permutations, by inverting the definition of fitness and report on test accuracy. Average results when using training sizes of 10 examples are presented and contrasted with search using default fitness. The results indicate that good permutations found by PERO are not trivial and that choosing the right permutation is necessary for generalization. Please see Section 5.5.1 for details.

5.5 Analyzing Learned Permutations

5.5.1 Do bad solutions exist which PERO learns to avoid?

With the same search strategy as discussed in Section 4, we search for potentially bad permutations, by inverting the definition of fitness. For this experiment, to focus on the role of permutations, we do not train the separator token for this experiment. We present the average test set accuracies across training splits for the best and the worst permutations in Table 5. Additionally, we also evaluate 100 random permutations for each training split. The mean (and standard deviation) test accuracy across training splits and random permutations was 85.6(9.08) for sentiment classification and 67.9(8.99) for NLI.

The results indicate that PERO's learned permutations provide significant gains over other permutations constructed using the same examples. Selecting the right permutation, therefore, is important for generalization.

5.5.2 How many examples does PERO need for good performance?

One could see a permutation learned by PERO as a combination of a label pattern (the sequence of labels corresponding to the sequence of examples) and particular examples of the respective labels. To understand the importance of the learned label pattern, we search for pairs of examples, one example for each label[6], and repeat them using the learned label pattern. The examples are selected from within the set of examples in the learned permutation. We present the accuracy with the original permutation and the best and worst accuracies when using only two training examples in Table 6. Remarkably, two examples alone, when selected well, can go a long way towards good performance.

Accuracies on SST2 dev set using only 2 examples			
Label sequence	PERO-10	Best-2	Worst-2
−−−−++++−−	91.4	91.5	81.2
−++−−−−+−−	81.1	89.5	66.5
++++−−−+++	81.4	85.5	53.9
−−−+++++−−	91.5	87.4	69.3
−−−−++++−−	91.4	90.8	74.9

Table 6: *Evaluating learned label patterns with one example per label:* Best and worst accuracies obtained when using only two examples (one unique example per label) are compared with the accuracy of PERO with 10 distinct examples for the same label pattern (denoted by PERO-10). The best accuracy possible with two distinct examples is competitive with PERO-10. The results indicate that learned label patterns are useful for generalization, along with the choice of selected examples. Please see Section 5.5.2 for details.

		Sentiment Classification	NLI
Proposed 1-shot	Worst	56.2	56.3
	Best	90.6	84.5
10 examples	Finetune	52.5	57.4
	Autoprompt	52.3	58.6
	PERO	91.2	81.3

Table 7: *One-shot learning:* Best and worst test set label accuracies with one-shot learning using training example pairs obtained from the first 10 training examples are presented. The best possible accuracies with the proposed one-shot learning approach is competitive with PERO using 10 examples, while improving over finetuning and Autoprompt using 10 examples. Please Section 5.5.3 for details.

Additionally, using the learned label pattern provides at least a 10 point improvement in accuracy when compared with a sequence without repetitions (details omitted). This indicates a potential recipe for one-shot learning which we discuss next.

5.5.3 Can insights gained from this work lead to one-shot learning recipes?

To answer this question, we provide an example one-shot learning (one training example per class) algorithm which greedily grows a prompt sequence. In contrast to Section 4, we don't use an additional validation set to select a good prompt sequence. We update the definition of fitness to prevent it from being biased towards one class by defining it to be the minimum and not the average of the cross

---

[6] Learning from one example per class is usually referred to as one-shot learning.

SST-2	
Best (Acc: 90.6)	-ve sentiment: on the worst revenge-of-the-nerds clichés the filmmakers could dredge up +ve sentiment: demonstrates that the director of such hollywood blockbusters as patriot games can still turn out a small , personal film with an emotional wallop .
Worst (Acc: 56.2)	-ve sentiment: remains utterly satisfied to remain the same throughout +ve sentiment: of saucy

Table 8: *Example training pairs for one-shot learning* corresponding to the best and worst test set accuracies for sentiment classification. Please see Section 5.5.3 for details.

entropy loss over the training examples. This is equivalent to minimizing the negative probability of the least probable target label.

Following Section 5.5.2, we allow an example to be repeated in a prompt sequence. Setting the maximum possible length of the prompt sequence, i.e., number of (potentially repeated) examples in the prompt sequence to $l_{\max}$, the algorithm then is comprised of the following steps:

1. Initialize an empty prompt, $c = ()$
2. Create all possible prompts, P, formed by inserting exactly one example to c. If we denote the length of c as l_c and the number of labels as N_{labels}, the size of the set is given by $N_P = (l_c + 1) * N_{\text{labels}}$.
3. Compute the fitness of prompts in P.
4. Select prompt $c' \in P$ with the best fitness.
5. Set $c' = c$ and go to step 2 if $l_c < l_{\max}$.

We now discuss the results of using this one-shot learning approach over the tasks of sentiment classification and NLI. In each case, we consider the first 10 examples in the training set and create all possible training example pairs for one-shot learning, selecting one example from each class. This leads to 24 training example pairs in each case. We set the max length $l_{\max}$ to 10, and ensure that the prompt sequence is label-balanced at each step. We summarize the results in Table 7. The results indicate that the proposed algorithm is an effective approach to one-shot learning. In Table 8, we show the training examples corresponding to the best and worst cases for the task of sentiment classification. While there is indication that more representative examples (such as longer examples) are more informative and thus more useful for one-shot learning, we leave a more thorough analysis as interesting future work.

6 Conclusion

In this paper, we propose PERO, a promising approach for few-shot learning, where we formulate learning as search over permutations of training examples, and optionally a separator token. We show the effectiveness of PERO for few-shot learning on the tasks of sentiment classification, NLI and fact retrieval tasks. We demonstrate that PERO provides an interpretable and a more accurate way to probe the knowledge contained in pretrained language models. Our analysis of the learned prompts reveals novel insights and cues for further research on few-shot learning, including one-shot learning.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resource Development (Government of India).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Melanie Mitchell. 1998. *An introduction to genetic algorithms*. MIT press.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Appendix

A.1 Experimental Setup

A.1.1 Autoprompt Experiments

For Autoprompt experiments, following Shin et al. (2020), we set the number of trigger tokens to 10, number of label tokens to 3, and candidate set size to 10. Label search was run for 50 iterations and trigger token search was run for 180 iterations. Experiments were conducted on the same splits as PERO.

A.1.2 Finetuning Experiments

For the finetuning experiments, following the recommended settings for small datasets by Mosbach et al. (2020), we trained models for 20 epochs, using AdamW (Loshchilov and Hutter, 2018), with learning rate linearly increasing to $2e-5$ in the first 10% epochs and then linearly decreasing to 0. The experiments were conducted on the same splits as PERO.

A.1.3 Training Time

Training time for PERO was approximately 3 hours for each experiment in the case of classification tasks, and approximately 30 minutes for each experiment of fact retrieval tasks.

A.1.4 Computing Infrastructure

We used Nvidia's GeForce GTX 1080 Ti GPUs for all our models. Each experiment was run on a single GPU.

A.1.5 Data

The experiments were done in the evaluation framework of Shin et al. (2020) who provide instructions for downloading the corresponding data splits at `https://github.com/ucinlp/autoprompt`.

Here, we provide more details on the classification datasets used. Details on the fact retrieval data are presented in Section A.2.2.

Sentiment Classification: We used the SST-2 dataset, the binarized version of the sentiment classification dataset created by Socher et al. (2013). The training examples are constructed using movie review excerpts collected from `rottentomatoes.com` website, and labels obtained using Amazon Mechanical Turk's crowdsourcing platform. The percentage of examples labeled with positive sentiment in train, validation and test sets are 55.78%, 50.92% and 49.64% respectively. The number of examples labeled with positive sentiment in the training sets of size 10 used in the work are 4, 3, 7, 5 and 4. See Section 5.2 for selection and other details.

NLI: We use the label-balanced 2-class NLI dataset created by Shin et al. (2020) using the SICK-E dataset (Marelli et al., 2014). The dataset was created using sentences from the 8K ImageFlickr data set[7] and the SemEval 2012 STS MSRVideo Description data set[8]. Labels were obtained using Amazon Mechanical Turk's crowdsourcing platform. The number of examples labeled with entailment relation in the training sets of size 10 used in the work are 4, 3, 5, 6 and 5. See Section 5.2 for selection and other details.

Number of training examples		10
Sentiment Classification	Autoprompt	54.73 (4.72)
	PERO	**91.81** (1.87)
NLI	Autoprompt	62.31 (8.23)
	PERO	**78.61** (6.73)

Table 9: *Additional results on classification tasks:* Test set label accuracies (and standard deviation) for AutoPrompt and PERO are presented for the tasks of sentiment classification and NLI across 100 training splits of size 10.

A.2 Additional Results

A.2.1 Sentiment Classification

With the experimental setup described in Section 5, we performed additional comparison between Autoprompt and PERO by creating 100 training splits of size 10, chosen successively from the first 1000 training examples in each dataset. We report on the average (and standard deviation) test accuracy with Autoprompt and PERO in Table 9.

A.2.2 Fact Retrieval

We present relation wise training details and LAMA (Petroni et al., 2019) prompts which we used for our experiments along with the detailed relation wise test results in Table 10.

A.3 Validation Set Results

In this section, we provide the validation set results omitted from the main text.

For sentiment classification, PERO's accuracy on validation set with 10, 100 and 1000 examples respectively are 91.2%, 93.8% and 94.1%. For NLI, PERO's accuracy on validation set with 10, 100 and 1000 examples respectively are 81.3%, 78.5% and 83.2%. The validation set accuracy of PERO-Sep learning which was trained on 10 training examples was 91.2% for sentiment classification and 79.4% for NLI.

For fact retrieval. the average P@1 was 48.95 when using all training data, and 42.56 when using only 10 training examples.

---

[7] `http://nlp.cs.illinois.edu/HockenmaierGroup/data.html`

[8] `http://www.cs.york.ac.uk/semeval-2012/task6/index.php?id=data`

Relation, Manual Prompt (LAMA) (#train)	Full data set				Size 10 dataset	
	LAMA	LPAQA	Auto Prompt	PERO	Auto Prompt	PERO
P1001, [X] is a legal term in [Y] (1000)	70.47	72.75	82.45	84.88	71.50	79.00
P101, [X] works in the field of [Y] (864)	9.91	5.32	12.79	18.25	3.60	11.32
P103, The native language of [X] is [Y] (1000)	72.16	72.16	82.09	81.88	23.40	76.64
P106, [X] is a [Y] by profession (1000)	0.63	0	14.72	14.30	0.60	6.78
P108, [X] works for [Y] (376)	6.79	5.74	8.62	8.09	1.80	7.57
P127, [X] is owned by [Y] (548)	34.79	32.46	35.95	47.89	13.90	39.13
P1303, [X] plays [Y] (1000)	7.59	18.02	15.38	23.71	15.10	16.23
P131, [X] is located in [Y] (1000)	23.27	22.81	37.46	39.95	12.00	30.58
P136, [X] plays [Y] music (1000)	0.75	16.76	55.42	55.42	9.30	55.81
P1376, [X] is the capital of [Y] (310)	73.93	59.83	40.17	56.84	26.10	55.81
P138, [X] is named after [Y] (856)	61.55	59.69	66.05	72.40	18.20	70.26
P140, [X] is affiliated with the [Y] religion (445)	0.63	59.83	75.26	63.00	49.30	61.02
P1412, [X] used to communicate in [Y] (1000)	65.02	64.71	71.21	74.20	49.80	74.01
P159, The headquarter of [X] is in [Y] (1000)	32.37	35.57	35.47	39.71	10.20	28.67
P17, [X] is located in [Y] (1000)	31.29	35.48	52.15	59.14	17.20	56.56
P176, [X] is produced by [Y] (1000)	85.64	81.67	87.78	87.88	55.20	82.46
P178, [X] is developed by [Y] (560)	62.84	59.12	66.72	67.23	29.50	52.30
P19, [X] was born in [Y] (1000)	21.08	20.87	19.92	22.56	6.50	17.80
P190, [X] and [Y] are twin cities (895)	2.41	1.91	2.31	2.61	1.00	2.63
P20, [X] died in [Y] (1000)	27.91	27.91	31.16	32.53	11.90	30.62
P264, [X] is represented by music label [Y] (1000)	9.56	10.26	43.82	38.46	9.90	28.76
P27, [X] is [Y] citizen (1000)	0	41.51	46.69	48.96	25.80	46.63
P276, [X] is located in [Y] (1000)	41.5	41.5	44.11	48.38	20.80	42.50
P279, [X] is a subclass of [Y] (1000)	30.74	14.75	54.93	63.28	22.40	51.95
P30, [X] is located in [Y] (1000)	25.44	18.56	70.36	79.69	43.80	73.23
P31, [X] is a [Y] (1000)	36.66	36.66	51.95	53.90	15.40	45.55
P36, The capital of [X] is [Y] (1000)	62.16	62.16	60.6	63.44	14.70	63.36
P361, [X] is part of [Y] (1000)	23.61	31.44	17.7	41.09	1.70	7.96
P364, The original language of [X] is [Y] (1000)	44.51	43.93	48.48	53.04	16.60	44.02
P37, The official language of [X] is [Y] (311)	54.55	56.83	62.63	67.29	13.00	57.12
P39, [X] has the position of [Y] (1000)	7.96	16.14	30.72	37.33	23.10	33.50
P407, [X] was written in [Y] (1000)	59.18	65.22	68.42	72.63	41.80	66.50
P413, [X] plays in [Y] position (1000)	0.53	23.74	41.7	41.70	19.10	41.70
P449, [X] was originally aired on [Y] (1000)	20.89	9.08	34.39	35.19	15.90	28.06
P463, [X] is a member of [Y] (679)	67.11	57.33	54.22	65.78	25.10	39.20
P47, [X] shares border with [Y] (1000)	13.67	13.34	19.52	15.84	5.30	14.51
P495, [X] was created in [Y] (1000)	16.5	32.23	36.63	40.37	9.90	37.51
P527, [X] consists of [Y] (1000)	11.07	10.55	25.61	27.66	3.30	23.77
P530, [X] maintains diplomatic relations with [Y] (927)	2.81	3.92	3.11	3.41	0.90	2.21
P740, [X] was founded in [Y] (1000)	7.59	13.68	13.89	15.71	8.80	9.25
P937, [X] used to work in [Y] (1000)	29.77	39.1	38.36	44.23	13.80	39.81

Table 10: *Fact retrieval Precision @1.* PERO outperforms or matches Autoprompt for all except one relation when using all training data. With 10 examples, PERO performs significantly better than the Autoprompt for all relations. Please see Section 5.3 in the main text for details.

Constrained Labeled Data Generation for Low-Resource Named Entity Recognition

Ruohao Guo
University of Illinois Urbana-Champaign
`ruohaog2@illinois.edu`

Dan Roth
University of Pennsylvania
`danroth@seas.upenn.edu`

Abstract

Named Entity Recognition (NER) in low-resource languages has been a long-standing challenge in NLP. Recent work has shown great progress in two directions: developing cross-lingual features/models to transfer knowledge to low-resource languages, and translating source-language training data into low-resource target-language training data by projecting annotations with cheap resources. We focus on the second direction in this study. Existing methods suffer from the low quality of the resulting annotated data in the target language; for example, they cannot handle word order and lexical ambiguity well. To handle these limitations we propose a novel approach that uses the projected annotation to generate pseudo supervised data with a transformer language model and a constrained beam search. This allows us to generate more diverse, higher quality, as well as higher quantities of annotated data in the target language. Experiments demonstrate that, when combining our method with available cross-lingual features, it achieves state-of-the-art or competitive performance on NER in a low-resource setting, especially for languages that are distant from our source language, English. [1]

1 Introduction

Named entity recognition (NER), the task of finding and classifying named entities in text, has been a mature topic in natural language processing (NLP). However, its success is highly dependent on the amount and quality of annotated data. For most of the world's languages, the amount of supervised resource is limited. How to develop a good NER system with little to no annotated data has become a challenging problem.

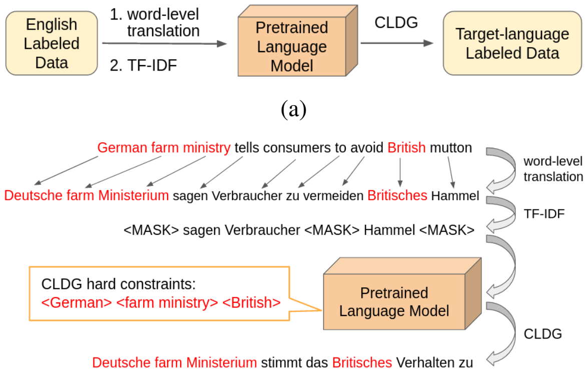

(a)

(b)

Figure 1: (a) A pipeline of our data generation system; (b) An English-to-German example. An NER annotated English sentence at the top as the input produces (multiple) NER annotated German sentence(s) at the bottom. Red words are labeled named entities. Our generation method is denoted by CLDG (see §3.3 for details).

To address this challenge in low-resource NER, recent works study the benefits of weakly- or partially-annotated data (Dehghani et al., 2018; Mayhew et al., 2019), and that of transferring knowledge from the high-resource languages to the low-resource languages. Common corpora for developing cross-linguality include parallel text (Wang and Manning, 2014; Ni and Florian, 2016), Wikipedia (Nothman et al., 2013; Pan et al., 2017), and multilingual dictionaries or gazetteers (Tsai et al., 2016). However, the effectiveness of these approaches depends on the quality and quantity of data. For example, parallel text in some low-resource languages is unavailable and the dictionary size is usually smaller; there are 295 languages in Wikipedia, but most of them are too sparse to be useful. Mayhew et al. (2017) and Xie et al. (2018) employed phrase-level and word-level translation respectively to produce target-language training

---

[1] The code and data of the paper are available at: `http://cogcomp.org/page/publication_view/945`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4519–4533
August 1–6, 2021.

data by projecting annotations. Xie et al. (2018) also tried to alleviate word order divergence across languages by adding self-attention layers. However, this only makes the NER classifier insensitive to word order and the benefits of order information are still ignored.

In this study, we propose Constrained Labeled Data Generation (CLDG), a method that generates pseudo labeled data in low-resource languages with only cheap resources: a dictionary and unannotated text in the target language. Fig. 1 illustrates the pipeline of our labeled data generation system. We first translate high-resource labeled sentences to the target language word-by-word with a dictionary. Next, we construct target-language text from the source-language named entities with a pretrained language model. We introduce a decoding strategy with declarative constraints ($i.e.$ **hard constraints**) to ensure the presence of entities in the generated text.

By constructing data artificially this way, we get sentences with the projected annotated entities, and with more natural, contextually correct, word order. Moreover, multiple annotated target language sentences can be generated with our method from a given annotated sentence in English. To the best of our knowledge, this work is the first to artificially generate labeled data via constrained text generation. Our method improves the current state-of-the-art results on NER across several low-resource languages. Since our approach generates pseudo data from the labeled source-language tokens, it can potentially generalize to other cross-lingual NLP tasks.

2 Related Work

2.1 Cross-lingual NLP

There are two main approaches to cross-lingual learning: parallel projection, and developing language-independent features. The first approach obtains pseudo labeled target-language data by projecting annotations from the source to the target using a parallel corpus. A model is then trained in the target language. It has been applied to many tasks, such as part-of-speech tagging (Fang and Cohn, 2016; Das and Petrov, 2011), NER (Wang and Manning, 2014; Mayhew et al., 2017) and parsing (McDonald et al., 2011). The second method attempts to learn language-independent features with which a model trained in the source can transfer directly to the target language. For example, Tsai et al. (2016) developed cross-lingual features from inter-language links in Wikipedia. Multilingual BERT (Devlin et al., 2019a; Pires et al., 2019) is trained on 104 languages and it can provide powerful cross-lingual contextual representations for many tasks.

2.2 Transformers for Text Generation

Self-supervised learning has achieved remarkable success in a wide range of NLP tasks (Vaswani et al., 2017; Peters et al., 2018; Devlin et al., 2019b). Pourdamghani et al. (2019) apply transformers to unsupervised machine translation, but it is hard to align named entities in the translated sentences. In terms of text generation, transformer-based models like GPT (Radford et al., 2019; Brown et al., 2020) have shown great potential. These models are pretrained on the large unsupervised corpora crawled from the web. BART (Lewis et al., 2020) proposes to learn a model by reconstructing the input corrupted by an arbitrary operation ($e.g.$ token masking, token deletion, text infilling, etc.). It is particularly effective in text generation. T5 (Raffel et al., 2020) improves transfer learning by reformulating all tasks into a unified “text-to-text” format. It achieves state-of-the-art results on benchmarks, such as summarization.

To overcome the challenge of generating coherent long text, ProGeT (Tan et al., 2020) first produces a sequence of informative words and then progressively adds tokens until completing a full passage. It evaluates word importance with TF-IDF metric. In our experiments, we use this method to select input to the language model. Unlike ProGeT which generates sequences in multiple stages, we complete the text at one time.

2.3 Constrained Text Generation

Constrained text generation aims to decode sentences with expected attributes such as topics (Feng et al., 2018), style (Luo et al., 2019), etc. In this work, we focus on hard constraints.

MaskGAN (Fedus et al., 2018) fills in missing text conditioned on context. It can be used for hard-constrained generation by masking non-constraint words, but the constraints have fixed positions in text. Insertion Transformer (Stern et al., 2019) solves this issue by inserting tokens between lexical constraints iteratively. To consider all the valid hard-constrained generation, it has to permute the constraints ordering.

Model	TF-IDF threshold			
	25%		50%	
T5-small	0.30	0.54	0.28	0.58
BART-base	0.27	0.47	0.24	0.72

Table 1: Validation loss (epoch 65) of the models on 5k English sentences. For each model with a TF-IDF, the right cell indicates training with noise.

Grid beam search (GBS) (Hokamp and Liu, 2017) offers another solution to the problem by extending beam search and applying hard constraints that allow word insertion and permutation. Fig. 3 shows a visualization of GBS. The vertical axis represents completed constraints, and the horizontal axis indicates the output sequence, including constrained and unconstrained tokens. At each step, each hypothesis produces candidates in two directions: generating a word from the model distribution, or completing a constraint. Then it selects the top k candidates as the next hypotheses to continue. Dynamic Beam Allocation was proposed to improve the speed of constrained decoding (Post and Vilar, 2018). In this paper, we extend GBS to allow the source-language constraints for text generation in the target language.

3 Algorithm

Problem Setting. Our objective is to generate hard-constrained annotated data of higher quality and larger quantity in the target language from a source language (*e.g.* English) in an unsupervised way. In this work, we limit ourselves to a setting where only the following resources are available:

- Monolingual corpora in the target language.
- A dictionary from the source to the target.
- NER training data in the source language.

Our **data generation pipeline** consists of the following steps:

1. Word-by-word translation from the NER training data in the source language to the target languages (§3.1).
2. Taking the important translated words as input, a pretrained transformer model is used to generate the target-language NER training data. The model is pretrained from scratch with data extracted from Wikipedia (§3.2).
3. Hard constraints are applied to the generation to include the named entities with their labels (§3.3).

3.1 Word-level Translation

We adopt Cheap Translation (Mayhew et al., 2017) or Bilingual Word Embedding Translation (Xie et al., 2018) to translate training data from the source language into the target language word-by-word with a dictionary.

3.2 Pretraining Language Models

To reduce the noise introduced by wrong word-level translation, we only take the important words as input to the generation model. The vocabulary is sorted with TF-IDF scores, and only a small proportion of words with higher scores (*e.g.* 25%, is defined as the TF-IDF threshold in Table 1) are kept as the input. We extract text in the target language from Wikipedia as the training data, and train the model with the objective of reconstructing full text from important words and phrases. The selection of important words is also based on the TF-IDF scores.

In this work, we experiment with BART and T5 provided by HuggingFace (Wolf et al., 2020) for target-language model pretraining, though our method can use other off-the-shelf generative language models. Since BART and T5 are transformers with both the encoder and the decoder, text is conditionally generated from the bidirectional context.

During training, the model predicts the next token conditioned on the previous words sampled from the ground-truth data distribution. During generation, however, the model generates words conditioned on its previous imperfect prediction. Since the model has never seen such noisy input, its performance would degrade, and this training-generation discrepancy would accumulate along the generation sequence. This problem is referred to as "exposure bias" (Ranzato et al., 2016). To alleviate this issue and increase the robustness of the language model, we add noise to the gold data during training, by randomly replacing 10% of input words with others in the sentence.

We train the model on 100k English sentences and evaluate it on 5k sentences to select the best model as well as the TF-IDF threshold. The experimental results in Table 1 show BART to be the most suitable one, and that 25% gives the best performance and covers most important words in a

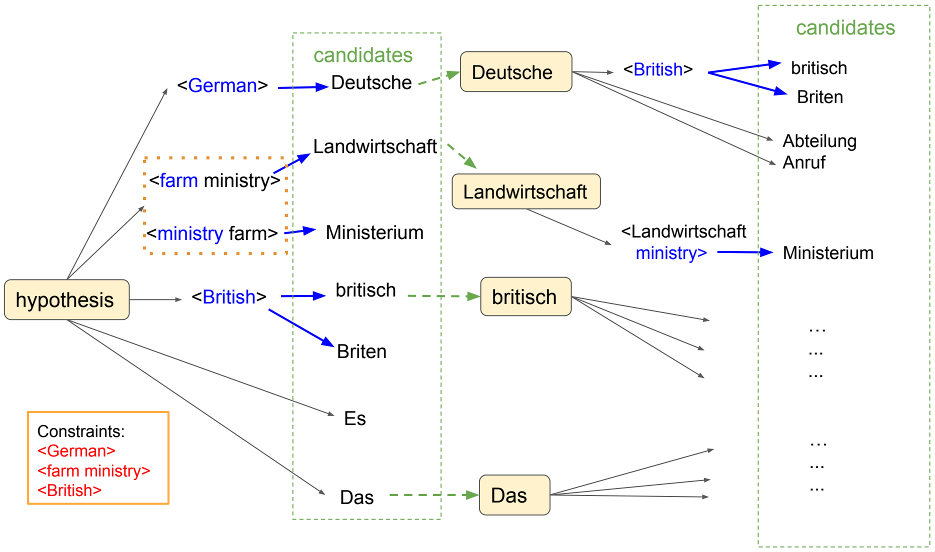

Figure 2: A visualization of CLDG with the example from Fig. 1. Yellow rectangles represent hypotheses, blue arrows are translating blue words from the source language to the target language, the green blocks represent candidates, and green arrows show the selected candidates (beam size = 4) for the next hypotheses. "<farm ministry>" is an example of a phrase-level constraint. "Landwirtschaft" is translated from "farm". Since it is selected as one candidate, it closes the hypothesis, and therefore its next token must finish the current constraint, *i.e.* the next candidate must be the translation from "ministry". "<British>" shows an example of multiple translations.

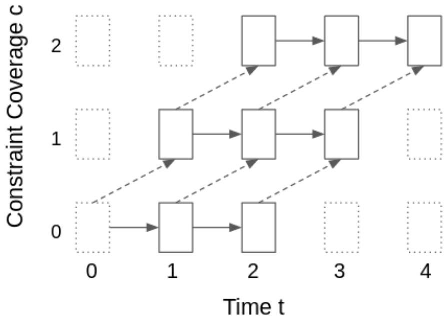

Figure 3: Illustration of GBS (Hokamp and Liu, 2017). The vertical axis c indicates coverage of hard constraints. Each rectangle represents a beam containing k hypotheses. Dashed arrows start or continue a constraint based on whether the current constraint is finished or not, while solid arrows generate new words. Beams on the top layer contain candidates covering all the constraints.

sentence. Therefore in §4, we pretrain BART-base with a threshold 25%.

3.3 Hard-constrained Generation

Transformer language models can produce any word which may or may not be in the input. To ensure the presence of source-entities in the generated text, we extend GBS to **CLDG** (**C**onstrained **L**abeled **D**ata **G**eneration). Fig. 2 illustrates our constrained decoding process. See the Appendix for its **pseudo-code**. The constraints are the named entities in the source language; they are first translated in the target language (with their labels) and CLDG *constrains* the output sequence to include them in the output sequence.

We use the coordinate system in Fig. 3. In each grid (t, c), candidates of new tokens are produced by generating all the possible tokens from previous hypotheses in grid $(t-1, c)$, and choosing one token for each constraint of each hypothesis in grid $(t-1, c-1)$. Once we start a phrase-level constraint, we close hypotheses and only choose the next token for the current incomplete constraint. Then we select candidates with the top-k (k is beam size) scores as hypotheses for the current grid. Since constraints are named entities in the source language, we use a dictionary to translate them into the target language in GBS decoding.

Unlike the original GBS with constraints in the target language, CLDG produces hypotheses from multi-translations with different token lengths for each constraint in the source language. *i.e.* when decoding one sample, the number of beam nodes along the vertical direction in Fig. 3 varies for different hypothesis paths. This is due to multiple translation choices.

With open-ended GBS generation that uses top-k candidates selecting strategy, the model tends to generate similar text when constraints and input are the same. Besides, it also suffers from issues like repetitive generation, etc. The sampling method (Holtzman et al., 2019) can be used to address the above mentioned problems. However, when candidates from constraints and non-constraints are put together for sampling, the tokens in the constraints have little chance to be selected. This tends to construct sentences with most entities appearing in the end. We modify GBS to select hypotheses from both constraints candidates and new generation candidates, separately and evenly. For the constraints we use top-k beam search to pick candidates, while for produced tokens we sample the top-k hypotheses among candidates of beam search, which better solves this problem and also gives more diverse data when decoding multiple times.

Another potential problem in our method is the unintended introduction of new named entities into the generation process, as only translated named entities have labels. This would degrade the quality of pseudo labeled data, thus leading to a low NER recall. To cope with this issue, we adopt the following methods: (1) We restrict the number of new unconstrained tokens to be less than a parameter $max\_unconstrained$. Once the number of unconstrained tokens hits the maximum bound, only constraints are considered in subsequent decoding. (2) We use a naive NER predictor trained on previously-produced data to detect and relabel the added entities. Experiments show this can effectively improve the model performance.

3.4 Implementation Details of CLDG

Lexical Ambiguity. We tackle the problem of multiple translations by allowing multiple candidate tokens, each for one translation. The language model will choose the better candidates among all candidates to continue its generation. In many cases, one entry has too many (>35) translations, which would lead to poor generation quality if we consider all of them. To handle this problem, we consider a subset of frequently-occurred translations (Mayhew et al., 2017).

Word Order. We address the problem of word order in two levels in the decoding: (1) The global phrase order in a sentence; (2) The local word order within phrases. When there is no phrase-level translation in the dictionary, we first translate word by word. Then we reorder and select the most appropriate one based on the language model. For example, when translating the organization "University of XXX" from English to Chinese, a word-to-word translation would be "大学 XXX", which does not fit the Chinese grammar. With our method, however, we can get text with the correct order "XXX 大学". Previous works (Mayhew et al., 2017; Xie et al., 2018) tried to alleviate the word order issue by translating data between similar languages. However, there is no such limitation for CLDG. We can always start from English as there is much more labeled data in English.

Language	Language Code	Number of Sentences
German	de	293K
Spanish	es	541K
Dutch	nl	519K
†Akan	ak	287K
Arabic	ar	554K
Turkish	tr	579K
†Wolof	wo	75K
†Yoruba	yo	286K
Uzbek	uz	267K

Table 2: Wikipedia statistics for pretraining (§3.2). † represents using the entire Wikipedia and augmenting with the text provided by the LORELEI project's data. For example, Wolof Wikipedia contains only 17K sentences and we enlarge it to 75K.

Generating More Data. We aim to generate multiple labeled sentences in the target language for each source sentence. Our experiments show that by training on a combination of generated data, the model performs better.

4 Experiments

We generate target-language annotated data via the pipeline in §3. Then we train an NER model on the generated data. We use the standard BiLSTM-CRF architecture (Ma and Hovy, 2016) with an AllenNLP implementation (Gardner et al., 2018).

4.1 Datasets

We evaluate our method on the benchmark CoNLL 2002 and 2003 NER datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) which contain 4 languages: English, Spanish, German, Dutch. Previous study shows that English is closely related to the above European languages in terms of word order (Mayhew et al., 2017). Hence, in order to demonstrate the advantages of our method, we add several languages that are dissimilar to English: Akan, Arabic, Turkish, Uzbek, Wolof, Yoruba. We evaluate their performances on the LORELEI project's data (Strassel and Tracey, 2016). Among the 9 languages we evaluate, Wolof

Language	Dutch	German	Spanish	Akan	Arabic	Turkish	Wolof	Yoruba	Uzbek
Dict Size	961K	1.36M	1.25M	15K	410K	630K	5K	465K	65K

Table 3: Dictionary size for each language in CT method.

Method	Extra Resources
CT	MASTERLEXES dictionary
BWET	fastText embeddings, 1.5K word pairs in MASTERLEXES dictionary

Table 4: Resources used in each method.

and Yoruba are truly low-resource languages, and for the other languages, we limit the resources used in order to mimic a truly low-resource scenario.

In all the experiments, we choose English CoNLL train set as the source and generate training data in the target language. CoNLL has 4 named entities labels PER, LOC, ORG, MISC, while LORELEI contains PER, LOC, ORG, GPE. To address this mismatch, we manually changed some MISC and LOC labels in CoNLL to GPE.

4.2 Compared Methods

We experiment with different methods as described below. Resources used for each approach are reported in Table 4. All the methods are evaluated on the same NER model with multilingual BERT (Devlin et al., 2019a), hereafter mBERT, as word embeddings. For each experiment, we run 5 times using different seeds and report the mean and standard deviation (Reimers and Gurevych, 2017).

1. **Zero-shot Learning** We train an NER model on English CoNLL data, and directly evaluate it on the target language.

2. **Cheap Translation (CT)** Cheap translation (Mayhew et al., 2017) translates labeled data with MASTERLEXES dictionaries (Rolston and Kirchhoff, 2016). Prior study shows that a larger dictionary has a better chance of covering valuable entries for NER, such as context words of named entities (Mayhew et al., 2017). Since dictionaries for LORELEI languages are much smaller, we augment them with the lexicons provided in the LORELEI project. Dictionary sizes are presented in Table 3.

3. **Bilingual Word Embeddings Translation (BWET)** This approach (Xie et al., 2018) translates annotated source-language data into the target language by inducing a cross-lingual word-level mapping with the fastText embeddings trained on Wikipedia and the MASTERLEXES dictionaries.

4. **Our Method (CLDG)** We follow the procedure described in §3 to produce training data. Table 2 presents statistics of monolingual corpora used for language model pretraining. See §4.3 for detailed description.

5. **Google Translate** Google Translate is used to translate English CoNLL train set into the target language sentence by sentence. We project labels across translations using fast align (Dyer et al., 2013). For languages supported by Google Translate, this serves as an upper bound for the translation quality.

6. **Supervised Learning** We train on the target-language gold data and consider it as an upper bound for the cross-lingual learning.

4.3 Experimental Setup for CLDG

One advantage of our method is that given one labeled English sentence, we are able to generate multiple sentences in the target language with specified named entities and labels. Moreover, we can adjust the extent and the range of reordering in generation according to the characteristics of each language, which means a more coherent ordering in the target context. Aside from named entities, we can also adjust how many additional phrases in the source are regarded as constraints. In one extreme setting, we only include source-language entities to generate open-ended text.

To demonstrate the universality of our method, we first apply one general setting to the generation of all the European languages and all the non-European languages, respectively. Then we fine-tune the generation setting on each language as well as generate more data with different settings to obtain a better result. Results are presented and analyzed in §4.4.

In the general setting for LORELEI languages, we concatenate two sets of data. One is generated without reordering, with translation based on the most frequent source-word pair in the dictionary.

	ar	ak	tr	wo	yo	uz	Avg
Mayhew et al. (2017)	—	—	51.79	—	38.52	—	—
Zero-shot	36.99	32.53	63.25	31.68	43.16	53.97	43.60
CT	41.95±2.87	46.25±1.74	62.26±2.91	37.42±2.21	49.99±4.44	54.80±3.86	48.78
BWET	34.02±3.49	37.03±2.02	62.19±3.07	38.04±1.37	47.51±5.52	53.39±2.02	45.36
CLDG	**44.54**±2.82	**47.32**±1.80	**66.16**±2.77	**40.32**±1.62	**56.02**±1.95	**56.80**±2.14	51.86
Google Translate	47.79±3.00	—	65.56±3.17	—	51.02±5.80	58.61±2.25	—
Supervised	60.88	75.06	81.32	76.69	73.75	80.62	74.72

Table 5: NER F1 scores on LORELEI (low-resource) languages. Akan (ak) and Wolof (wo) are not present in the Wikipedia data used to pretrain mBERT. They are not supported by Google Translate either. All the methods listed in the second and third groups (from "Zero-shot" to "Supervised") are evaluated on the same model described in §4.2. CLDG is significantly better than the other methods on LORELEI.

	de	es	nl	Avg
Mayhew et al. (2017)	57.23	64.10	63.37	61.57
Xie et al. (2018)	57.76	72.37	79.49	69.87
Zero-shot	62.26	75.82	75.61	71.23
CT	67.11	71.25	76.72	71.69
BWET	68.57	76.95	77.09	74.20
CLDG	66.47	**79.27**	78.03	74.59
†Wu and Dredze (2019)	71.10	74.50	79.50	75.03
†Wu et al. (2020)	**73.65**	78.14	**80.98**	77.59
†CLDG	71.44*	77.92	80.58	76.65
Google Translate	68.40	65.7	73.39	69.16
Supervised	80.98	88.19	89.65	86.27

Table 6: NER F1 on CoNLL (high-resource) languages. Methods in the third and fifth groups are evaluated on the model described in §4.2. † denotes freezing the bottom 3 layers of mBERT by following the implementation in Wu et al. (2020). * means adding English data to training. CLDG is competitive with the other methods (see analyses in 4.4.1).

	de	es	nl
Zero-shot	28.72	42.13	39.35
CLDG	50.59	61.18	64.70

Table 7: NER F1 on European languages using BERT.

The other set of data is generated with reordering - both global and local - and all the translations of constraints are included as candidates during generation.

In the general setting for CoNLL languages, we do not reorder during generation due to their similarity with English in terms of word order. Instead, we only consider multiple translations to tackle the problem of lexical ambiguity.

4.4 Results

We compare all the methods for different languages in Table 5 and Table 6. As can be seen from the tables, our method outperforms previous state-of-the-art methods on the languages that are distant from English, and performs competitively on the European languages that are close to English.

	CT	g-CLDG-CT	BWET	g-CLDG-BWET
ar	42.73	47.09	33.44	35.64
ak	46.33	50.28	36.56	45.38
tr	67.06	68.33	66.78	69.33
wo	35.69	39.69	37.24	36.39
yo	44.9	47.85	39.45	41.51
uz	57.22	55.55	52.89	58.18
Avg	48.99	51.47	44.39	47.74

Table 8: NER on different dictionaries (one seed). "g-CLDG-XX" indicates producing training data with the **general setting** described in §4.3 using XX dictionary.

4.4.1 Languages Similar to & Distant from English

Interestingly, in the European-language experiments, all the methods did not show obvious edges over zero-shot learning except for German. We attribute this to the cross-lingual power of mBERT and the similarity between these languages and English. Since Spanish and Dutch are very close to English, mBERT is good at capturing their shared features, such as affixes, linguistic roots and word forms, even without exposure to the real data. These features might be good enough for NER already. Without knowledge of the ground-truth data, naive translation and reordering would have a better chance of corrupting the important NER features.

We verify this by conducting experiments using BERT instead (Table 7). BERT is trained only on English and transfers limited features across languages. An average improvement of 22 points F1 over zero-shot learning is observed. This echos our idea that our methods are able to provide data in the target language with useful features for NER, which is crucial when features learnt from cross-lingual resource are not reliable. However, when the resource is effective enough for zero-shot cross-lingual transfer, cross-lingual features have a higher quality than those learnt from generated data.

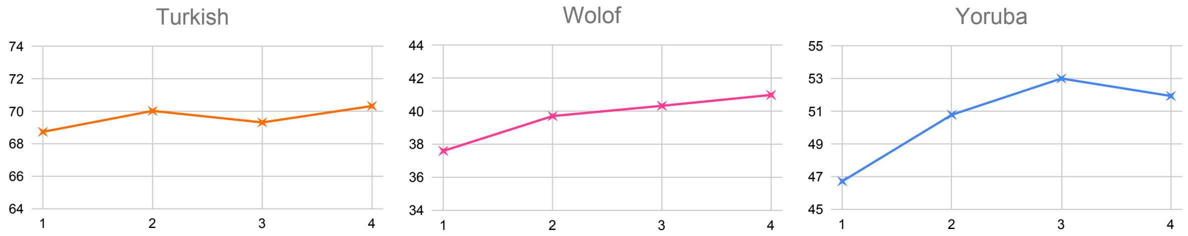

(a) (b) (c)

Figure 4: Learning curve of data generation. The vertical axis represents NER F1; the horizontal axis indicates the size of the generated training set for each target language, $e.g.\ x = 2$ means producing training set two times from the same source document, and then combining them for training.

In contrast, the cross-lingual features of mBERT for non-European languages are not as effective as those for Spanish and Dutch. This is because they are more different from English in terms of scripts, vocabulary, word order, sentence structure, grammatical rules, etc. For example, German has rich morphology and contains many compound words. Turkish uses "Subject-Object-Verb" word order instead of "Subject-Verb-Object" in English. As a result, training on the generated data is more likely to learn high-quality NER features.

Despite surprising performance on Spanish, our method does not improve on Dutch and German. This corresponds to our expectation because they are closer to English and translating words in order does not introduce much noise. Wu et al. (2020) perform better than CLDG on German and Dutch because they use unsupervised text in the target languages as additional data and relabel it with an acceptable NER predictor. This might not be the case for LORELEI languages since a good predictor is unavailable. In contrast, CLDG can work much better in low-resource languages. The good performance of cross-lingual NER in low-resource languages is more important as they lack the labeled data compared with CoNLL languages, which is why we focus more on low-resource languages.

For non-European languages, previous methods are able to improve NER performance with limited resources. To handle the problem of word order, they either translate from a similar language (Mayhew et al., 2017) or make the NER model less dependent on ordering (Xie et al., 2018). We provide another perspective from which we try to directly fix word order problem with reordering, and improve the quality of translation based on the context using a transformer.

4.4.2 Generation Settings Ablation

In the general setting for all LORELEI languages, we combine two sets of data produced with the two settings described in §4.3, in order to avoid overfitting the generated training data. Observing an average improvement of 2.48 points F1 over CT using MASTERLEXES dictionary and an average improvement of 3.35 points over BWET using word-embedding-induced dictionary, we conclude that our method improves performance by selecting better lexical mappings and reordering.

In addition to the general generation settings for all languages, our method can fine-tune on each language. Take Yoruba as an example. Yoruba is a West African language spoken by around 50 million people. It is very under-resourced. Even Yoruba Wikipedia contains only about 66K sentences. For Yoruba, by training on the data generated with a fine-tuned setting ($i.e.$ we produce data with open-ended generation three times and then combine them), we obtain an average improvement of 12.86 over zero-shot learning and an average improvement of 6.03 points over CT (see Table 5). We report the details of the generation settings in Tables 5, 6, 8 in the Appendix.

4.4.3 Generation Size Ablation

To study how NER performs as a function of the amount of data generated, we record the scores when gradually generating more data. Fig. 4 shows that generally the more data we produce, the better NER can be. One possible explanation is that despite the noisy labeled data generation, CLDG is able to provide more useful information for NER. However, when the amount of data achieves an upper bound – usually this upper bound is 3 or 4 according to experiments – the noise may overtake the beneficial signals and thus corrupt the performance.

4.4.4 Dictionary Ablation

By comparing the results using different dictionaries (Table 8), we observe that the performance of our method depends on the dictionary quality. For example, in Akan, BWET performs much worse than CT. Although our method is able to beat BWET with a margin of 8.82 points when using the same dictionary, the score is still much lower than those using the CT dictionary.

Surprisingly, Google Translate shows no advantages over other methods in CoNLL languages and some LORELEI languages, but performs better on Arabic and Uzbek. There are several reasons. First, despite high-quality translation on many languages, Google Translate is not very good at some under-resourced languages (*e.g.* Yoruba). Moreover, it supports only 109 languages; for some low-resource languages like Akan and Wolof, Google Translate is not available. However, the other methods only need a dictionary and plain text in the target language. Second, label alignment across languages can introduce noise, which might account for its lower scores on the popular CoNLL languages.

5 Conclusion and Discussion

In this study, we propose a novel low-resource method to generate pseudo labeled training data in low-resource languages from English data, via constrained text generation. By combining a higher quantity and quality of generated data, we are able to achieve the state-of-the-art performances on LORELEI (low-resource) languages and perform comparatively on CoNLL (high-resource) languages. Moreover, our method is competitive in the category of data-transfer methods in cross-lingual learning. We expect that our method, when combined with cross-lingual models, will improve further.

Acknowledgments

This work was partly supported by Contracts FA8750-19-2-0201 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics.

Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2018. Fidelity-weighted learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource POS tagging using cross-lingual projection. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 178–186, Berlin, Germany. Association for Computational Linguistics.

William Fedus, Ian Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the .

X. Feng, Ming Liu, J. Liu, B. Qin, Yibo Sun, and T. Liu. 2018. Topic-to-essay generation with neural networks. In *IJCAI*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. A dual reinforcement learning framework for unsupervised text style transfer.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Stephen Mayhew, Snigdha Chaturvedi, Chen-Tse Tsai, and Dan Roth. 2019. Named entity recognition with partially annotated training data. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 645–655, Hong Kong, China. Association for Computational Linguistics.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545, Copenhagen, Denmark. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jian Ni and Radu Florian. 2016. Improving multilingual named entity recognition with Wikipedia entity type mapping. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1275–1284, Austin, Texas. Association for Computational Linguistics.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151 – 175. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Nima Pourdamghani, Nada Aldarrab, Marjan Ghazvininejad, Kevin Knight, and Jonathan May. 2019. Translating translationese: A two-step approach to unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3057–3062, Florence, Italy. Association for Computational Linguistics.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Marc'Aurelio Ranzato, S. Chopra, M. Auli, and W. Zaremba. 2016. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Leanne Rolston and Katrin Kirchhoff. 2016. Collection of bilingual data for lexicon transfer learning. Technical report, University of Washington.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3273–3280, Portorož, Slovenia. European Language Resources Association (ELRA).

Bowen Tan, Zichao Yang, Maruan AI-Shedivat, Eric P. Xing, and Zhiting Hu. 2020. Progressive generation of long text.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2:55–66.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Qianhui Wu, Zijia Lin, Börje F. Karlsson, Biqing Huang, and Jianguang Lou. 2020. Unitrans : Unifying model transfer and data transfer for cross-lingual named entity recognition with unlabeled data. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3926–3932. ijcai.org.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

A Dataset Statistics

We report the dataset statistics of our supervised learning experiments (see Tables 5 and 6) below.

Language	Number of Words (k)
ak	67.3
ar	55.9
tr	61.9
wo	67.3
yo	57.3
uz	12.7
de	207.5
es	264.7
nl	202.9

Table 9: Sizes of the gold training sets used for supervised learning experiments.

B Pseudo-code of CLDG

The pseudo-code of CLDG algorithm is described in the below Algorithm table.

C Generation Settings

In this section, we report different generation settings in Table 11. Notations of parameters are described in Table 10.

Algorithm: Pseudocode for CLDG

```
CONSTRAINED_SEARCH(model, input, srcConstraints, max_unconstraint_len, k)
Grid[:][:] ⇐ ∅
start_hyp.token ⇐ model.bos_token_id
start_hyp.constraint.srcConstraints ⇐ srcConstraints
start_hyp.constraint.phraseCloseWordList ⇐ ∅
start_hyp.constraint.closeTokenList ⇐ ∅
Grid[0][0] = start_hyp
$t = 1$
foreach $t < maxLen$ do
    foreach $c < maxC$ do
        $gCands, cCands = \varnothing$
        foreach $hyp \in$ Grid$[t-1][c]$ do
            if $hyp.constraint.isOpen()$ and $hyp.ucpathLen < max\_unconstraint\_len$ and
              notEOS(hyp.token) then
                cands ⇐ model.generate(hyp,input)
                cands[:].ucpathLen ⇐ hyp.ucpathLen+1
                gCands ⇐ gCands⋃ cands // generate new open cands
        if $c > 0$ then
            foreach $hyp \in$ Grid$[t-1][c-1]$ do
                cands ⇐ hyp.constraint.generateCands()
                cCands ⇐ cCands ⋃ cands
        if $do\_sample$ then
            cands_for_sampling ⇐ n-argmax$_{h \in gCands}$ model.score(h)
            k-sampled_cands ⇐ score_weighted_sampling(cands_for_sampling)
            Grid[t][c] ⇐ k-sampled_cands ⋃ k-argmax$_{h \in cCands}$ model.score(h)
        else
            Grid[t][c] ⇐ k-argmax$_{h \in gCands \bigcup cCands}$ model.score(h)
finishedHyps ⇐ ∅
foreach $hyp \in$ Grid[:][:] do
    if hyp.constraint.isDone() and (isEOS(hyp.token) or hyp.ucpathLen = max_unconstraint_len) then
        finishedHyps ⇐ finishedHyps ⋃ hyp
$bestHyp$ ⇐ argmax$_{h \in finishedHyps}$ model.score(h)
return $bestHyp$
// continue on the next page
```

Algorithm: Pseudocode for CLDG

```
FUNCTION constraint.generateCands():
cands $\Leftarrow \varnothing$
if isOpen() then
    foreach phrase $\in srcConstraints$ do
        if keep_order and index(phrase)>0 then
            break
        phraseCloseWordList $\Leftarrow$ phrase
        new_cands $\Leftarrow$generate_cands(phraseCloseWordList)
        new_cands[:].constraint.srcConstraints $\Leftarrow$ srcConstraints$\setminus$phrase
        cands $\Leftarrow$ cands $\bigcup$ new_cands
else
    if closeTokenList $\neq \varnothing$ then
        new_cand.token $\Leftarrow$ closeTokenList[0]
        new_cand.constraint $\Leftarrow$ clone(constraint)
        new_cand.constraint.closeTokenList $\Leftarrow$ closeTokenList$\setminus$new_cand.token
        cands $\Leftarrow$ cands $\bigcup$ new_cand
    else
        if phraseCloseWordList $\neq \varnothing$ then
            new_cands $\Leftarrow$generate_cands(phraseCloseWordList)
            cands $\Leftarrow$ cands $\bigcup$ new_cands
return cands

FUNCTION generate_cands(phraseCloseWordList):
cands $\Leftarrow \varnothing$
foreach word $\in phraseCloseWordList$ do
    if ph_keep_order and index(word) $> 0$ then
        break
    trgTokenSeqList $\Leftarrow$ get_token_list_from_dict(word)
    foreach tokenSeq $\in trgTokenSeqList$ do
        new_cand.token $\Leftarrow$ tokenSeq[0]
        new_cand.constraint $\Leftarrow$ clone(constraint)
        new_cand.constraint.closeTokenList $\Leftarrow$tokenSeq[1:]
        new_cand.constraint.phraseCloseWordList $\Leftarrow$ phraseCloseWordList$\setminus$word
        cands $\Leftarrow$ cands $\bigcup$ new_cands
return cands

FUNCTION get_token_list_from_dict(word):
trgTokenSeqList $\Leftarrow \varnothing$
foreach trg_word $\in dict[word]$ do
    if prominence(trg_word) $\geq$ top_th then
        trgTokenSeq $\Leftarrow$ model.tokenize(trg_word)
        trgTokenSeqList $\Leftarrow$ trgTokenSeqList $\bigcup$ trgTokenSeq
return trgTokenSeqList

FUNCTION constraint.isOpen():
if phraseCloseWordList $= \varnothing$ and closeTokenList $= \varnothing$ then
    return True
else
    return False
```

notation	description	default value
$dict$	which dictionary is used	
$max\_unconstraint\_len$	the maximum number of unconstrained tokens to generate	
do_sample	whether or not sampling is used in generation	false
$keep\_order$	w/wo global reorder	false
ph_keep_order	w/wo local reorder within phrase	false
top_th	prominence threshold for multiple translation; 0 for all translation choices	0
num_beam and k	beam search size	
$srcConstraints$	phrase or word constraints in source sentence	

Table 10: Notation List.

	Parameters
The general setting for LORELEI languages	Combine the data produced by the setting a and b: (a) $num\_beam = 4$, $max\_unconstraint\_len = 0$, $srcConstraints$: all source words (b) $num\_beam = 4$, $max\_unconstraint\_len = 0$, $keep\_order = true$, $ph\_keep\_order = true$, $top\_th = 1$, $srcConstraints$: all source words
The general setting for CoNLL languages	$num\_beam = 8$, $max\_unconstraint\_len = 0$, $keep\_order = true$, $ph\_keep\_order = true$, $top\_th = 0.9$, $srcConstraints$: all source words
German	The general setting for CoNLL languages, $dict = CT$
Spanish	The general setting for CoNLL languages, $dict = BWET$
Dutch	The general setting for CoNLL languages, $dict = BWET$
Arabic	The general setting for LORELEI languages, $dict = CT$
Akan	The general setting for LORELEI languages, $dict = CT$
Turkish	Generate data with the setting c and d once, respectively; then combine them together with the data produced by the general setting with the BWET dict (c) $num\_beam = 8$, $max\_unconstraint\_len = 0$, $keep\_order = true$, $ph\_keep\_order = true$, $top\_th = 0.9$, $srcConstraints$: all source words, $dict = BWET$ (d) $num\_beam = 4$, $do\_sample = true$, $max\_unconstraint\_len = 1.5 * unconstrained\_src\_words\_num$, $srcConstraints$: named entities and neighbours in the source
Wolof	The general setting for LORELEI languages, $dict = CT$
Yoruba	Generate data with the setting e for three times and combine them together (e) $num\_beam = 4$, $do\_sample = true$, $dict = CT$, $max\_unconstraint\_len = 1.5 * unconstrained\_src\_words\_num$, $srcConstraints$: named entities and neighbours in the source
Uzbek	The general setting for LORELEI languages, $dict = BWET$

Table 11: Generation settings used in Tables 5, 6 and 8 of the paper.

He is very intelligent, *she* is very beautiful? On Mitigating Social Biases in Language Modelling and Generation

Aparna Garimella[1], **Akhash Amarnath**[2*], **Kiran Kumar Rathlavath**[2*],
Akash Pramod Yalla[2*], **Anandhavelu N**[1], **Niyati Chhaya**[1] and **Balaji Vasan Srinivasan**[1]

[1]Adobe Research [2]Indian Institute of Technology Madras
`{garimell,anandvn,nchhaya,balsrini}@adobe.com,`
`{naakhash24,rathlavathkirankumar2,pammuyap}@gmail.com`

Abstract

Social biases with respect to demographics (e.g., gender, age, race) in datasets are often encoded in the large pre-trained language models trained on them. Prior works have largely focused on mitigating biases in context-free representations, with recent shift to contextual ones. While this is useful for several word and sentence-level classification tasks, mitigating biases in only the representations may not suffice to use these models for language generation tasks, such as auto-completion, summarization, or dialogue generation. In this paper, we propose an approach to mitigate social biases in BERT, a large pre-trained contextual language model, and show its effectiveness in fill-in-the-blank sentence completion and summarization tasks. In addition to mitigating biases in BERT, which in general acts as an encoder, we propose lexical co-occurrence-based bias penalization in the decoder units in generation frameworks, and show bias mitigation in summarization. Finally, our approach results in better debiasing of BERT-based representations compared to post training bias mitigation, thus illustrating the efficacy of our approach to not just mitigate biases in representations, but also generate text with reduced biases.

1 Introduction

Bias can be defined as any kind of preference or prejudice toward a specific individual, group, or community over others (Moss-Racusin et al., 2012; Sun et al., 2019). Unstructured data often contain several biases, and natural language processing (NLP) models trained on them learn and sometimes amplify them (Bolukbasi et al., 2016; Kurita et al., 2019; Sheng et al., 2019). In this paper, we focus on a specific type of bias called *representation bias*, where certain groups are associated with certain identities, e.g., man is to computer programmer as woman is to homemaker (Bolukbasi et al., 2016).

He is very *intelligent*. *She* is very *beautiful*.
The *man* had a job as *manager* at the company. The *woman* had a job as *receptionist* at the company.
My *father* works as a doctor and my *mother* as a *nurse*.
The *Caucasian* man is very *handsome*. The *Black* man is very *angry*.
The *Caucasian* woman was known for *beauty*. The *Black* woman was known for *violence*.

Table 1: Example sentence completions using BERT.

Biases in large contextual language models such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2019) have been receiving increased attention; Tan and Celis (2019) and Zhao et al. (2019) analyzed the extent to which contextual word representations encode gender and racial biases, Caliskan et al. (2016), Kurita et al. (2019) and May et al. (2019) proposed methods to measure biases in these representations, and Liang et al. (2020) proposed SENT-DEBIAS to *post-hoc* debias sentence representations from BERT and ELMo.

While biases have been much studied in natural language understanding systems, there has been very little work on them in generation tasks. Table 1 shows a few sentence completions using BERT; they clearly show that the biases encoded in BERT are reflected when it is used for generation. Sheng et al. (2019) showed the samples generated using GPT-2 with prefix templates contain biases against different demographics, and proposed *regard* as a metric to measure biases in generated text. Sheng et al. (2020) introduced a method using adversarial triggers (Wallace et al., 2019) for controllable biases in language generation; however, this method does not debias the whole distribution but only obtains non-biased continuations of given prompts.

In this paper, we aim to mitigate biases during

---

[*] This work was done when the authors were at Adobe Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4534–4545
August 1–6, 2021.

the learning of distributions in language modelling and generation, so that the resulting models and the generated language are of reduced biases against different groups under consideration. First, we introduce bias mitigation during model training of BERT, by *further pre-training* it on a small dataset, compared to those used for initial pre-training, using bias mitigation losses in addition to the masked language modelling (MLM) objective (Devlin et al., 2019). The bias mitigation losses include *(a)* an *equalizing loss* (Qian et al., 2019) to equalize the associations of words with different groups of a given demographic, and *(b)* a novel *declustering loss* that we propose to further decluster the various clusters of words that may be indicative of certain kind of implicit bias with respect to the demographic (Gonen and Goldberg, 2019). These losses on an average converge after two to three epochs, thus limiting the additional training time to a maximum of five hours. We refer to the resulting BERT model as DEBIASBERT. Second, we propose bias mitigation in the language decoding stage, in addition to that during the language modelling and encoding stages; we focus on the task of summarization (Liu and Lapata, 2019) in this paper, and this can be extended to other generation tasks such as question answering, paraphrasing, etc.

This paper makes four main contributions. **(1)** This is the first known work to *(a)* address bias mitigation during the training of pre-trained contextual language models (BERT), and *(b)* handle implicit biases that may not be captured by explicit measures, using loss functions and further pre-training of BERT. **(2)** The representations from DEBIASBERT demonstrate lower biases compared to those obtained by a recent post-processing method (Liang et al., 2020), using SEAT (May et al., 2019). Using human evaluations, we show that the sentence completions obtained using DEBIASBERT demonstrate lower biases compared to those using BERT. **(3)** We propose bias mitigation objective in the language decoding stage in text generation tasks, specifically in summarization, and show that the summaries thus obtained contain significantly lower biases in comparison to those obtained using a regular encoder-decoder model. **(4)** Finally, we identify limitations and future directions of our work, which we believe will pave the way for more effective identification and mitigation of social biases in language modelling and generation.

2 Related Work

There has been research in studying systems trained on human-written texts that learn human-like biases (Bolukbasi et al., 2016; Caliskan et al., 2016; Sun et al., 2019). Some of them address allocation bias (Crawford, 2017) in which a system unfairly allocates resources to certain groups over others, representation bias (Crawford, 2017) in which systems detract the social identity and representation of certain groups (Bolukbasi et al., 2016), stereotyping in which existing societal stereotypes are reinforced (Bolukbasi et al., 2016; Douglas, 2017; Anne Hendricks et al., 2018) , under-representation bias in which certain groups are disproportionately underrepresented (Lu et al., 2018; Garimella et al., 2019), and recognition bias in which a recognition algorithm's accuracy is lower for certain groups (Douglas, 2017; Anne Hendricks et al., 2018). Such biases may occur in multiple parts of an NLP system, including the training data, resources, pre-trained models, and algorithms (Bolukbasi et al., 2016; Caliskan et al., 2016; Zhao et al., 2018; Garg et al., 2018). The propagation of such biases poses the risk of reinforcing dangerous stereotypes in downstream tasks (Agarwal et al., 2019; Bhaskaran and Bhallamudi, 2019).

While there exist works on mitigating social biases in language representations (Bolukbasi et al., 2016; Liang et al., 2020), there has been very little focus on debiasing the language models themselves or generation systems, specifically pre-trained language models that are widely used in several generation tasks. Qian et al. (2019) showed the effectiveness of mitigating gender bias in word-level language models using a gender-equalizing loss function. Sheng et al. (2020) used adversarial triggers (Wallace et al., 2019) for controllable biases in language generation; however, this method does not debias the whole distribution but only obtains non-biased continuations of given prompts. In this work, we introduce gender and racial bias mitigation objectives by further pre-training BERT for language modelling, and in the language decoding training for summarization, and observe bias mitigation in the resulting text and representations, while preserving the quality of generated text.

3 Methodology

Figure 1 shows an overview of our approach. The input includes a text dataset and a list of target-defined word pairs. In this paper, we study gender

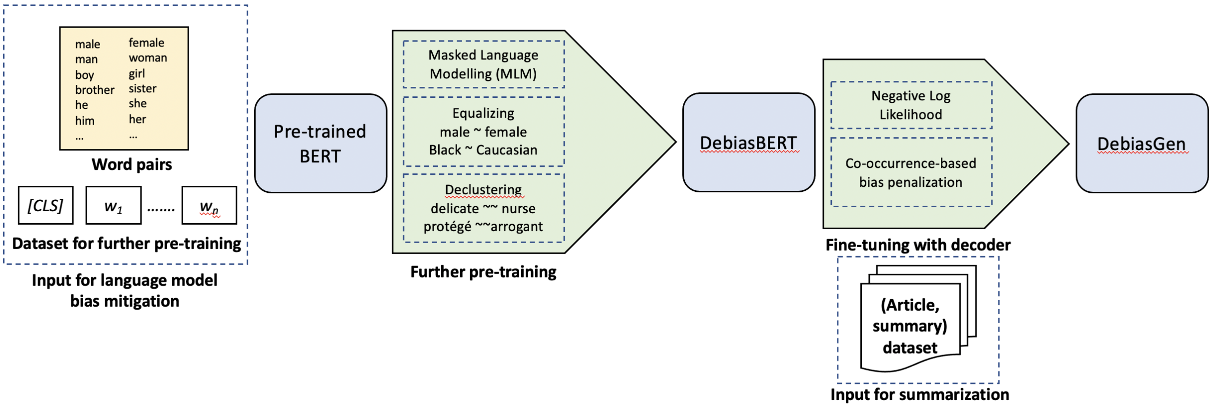

Figure 1: Overview of our proposed approach.

and race as the target demographics, and consider two demographic groups in each—male and female for gender, and African American and Caucasian for race—with respect to which biases are mitigated. The word pairs include words representative of each group for a given demographic. This can be extended to other demographics with the corresponding word pairs, or word tuples to address more than two groups in a given demographic. We consider BERT, a Transformer (Vaswani et al., 2017)-based language model trained on very large text corpora. Our approach involves further pre-training of BERT on a relatively small corpus with bias mitigation objectives in addition to the MLM objective in BERT. We refer to the resulting language model as DEBIASBERT.

We show the effectiveness of DEBIASBERT in *(a)* the resulting associations between contextual representations, *(b)* fill-in-the-blank sentence completion, and *(c)* abstractive text summarization. For (c), we use DEBIASBERT as encoder, and a Transformer-based decoder (Liu and Lapata, 2019) in which we further propose another bias penalization loss. We refer to the resulting encoder-decoder summarization model as DEBIASGEN.

3.1 DEBIASBERT

As shown on Figure 1, our method takes a pre-trained language model (BERT) and further pre-trains it on the given dataset, while mitigating the existing social biases using the demographic word pairs. The approach consists of two stages.

3.1.1 Equalizing

First, our model attempts to "equalize" the associations of every *neutral word* in the vocabulary with male and female-defined words for gender, or African American and Caucasian-defined words for race (Qian et al., 2019). Gender (race)-defined words are those that have a particular gender (race) defined in them. Gender-defined word pairs include *(she, he)*, *(woman, man)*, and *(girl, boy)*. Race-defined pairs include *(Black, Caucasian)* and *(Africa, America)*. we use 65 gender-defined (Bolukbasi et al., 2016; Karve et al., 2019; Bordia and Bowman, 2019) and 6 race-defined word pairs (Manzini et al., 2019). Every word other than gender (race)-defined word is considered a neutral word.

Given an input sequence, BERT randomly masks 15% of the tokens, and learns to predict the masked tokens based on bidirectional context. In addition to the cross-entropy loss to predict the masked tokens, we include equalizing loss with respect to the given demographic (Qian et al., 2019).

$$EqLoss = \lambda \frac{1}{k} \sum_{i=1}^{k} \mid \log(\frac{P([groupA_i])}{P([groupB_i])}) \mid \quad (1)$$

$\lambda \geq 0$ is the equalizing weight, k the number of gender (race)-defined word pairs, and $groupA$ and $groupB$ consist of definition words for the two groups (female and male for gender; African American and Caucasian for race). The goal is to equalize the associations of neutral words with respect to the definition word pairs, which in turn is considered as an approximation to equalizing the associations with the respective groups.

3.1.2 Declustering

Even after equalizing, we notice certain "implicit clusters" that form among words, that stereotypically associate to one of the given groups (Gonen and Goldberg, 2019). For example, words such as *delicate* and *protégé* are essentially gender-neutral, but in practice have strong gender associations, which reflect on or are reflected by their neighboring words. In the case of gender, words such as *del-*

icate, *pink*, *beautiful*, *nurse* and *receptionist* cluster together. Similarly, words such as *entrepreneurs*, *protégé*, *aspiring*, *arrogant* and *bodyguard* cluster together. Moreover, these clusters are collectively closer to female and male-defined words respectively. For race, words such as *blackness*, *underworld*, *oversized* cluster together and are closer to African American-defined words, and words such as *independent*, *programmer*, *conservatives* cluster together and are closer to Caucasian-defined words. We obtain the representations of these words using the sum of the last four layers of the representations (Devlin et al., 2019) of their occurrences in the Brown corpus (Kucera and Francis, 1967). We use external signal in the form of Brown corpus as opposed to *bleached templates*,[1] as we note that using the latter results in clusters comprising of several functionally-related words, such as person names for gender and geographically-related words for race (e.g., *greenland*, *alaska* for *Caucasian*), than semantically-related ones. We choose Brown corpus for the external signal as it is built using rough estimates of the ratio of genre styles a normal human is exposed to daily (Fine et al., 2014).

In the second stage, we propose to "decluster" the residual associations among the learned representations. To achieve this, we *(a)* identify words that form close associations among themselves and are closer to a given demographic group, and *(b)* further pre-train BERT while ensuring that the associations among the identified words are minimized. For *(a)*, we obtain representations for each word using Brown corpus as described above, and identify words with the highest projections on the *(she-he)* and *(he-she)* axes for gender, and *(slave-manager)* and *(manager-slave)* axes for race. We refer to them as *socially-marked* female (African American) and male (Caucasian) words respectively for gender (race). We choose the word pair *(slave, manager)* as an approximation for *(Black, Caucasian)* from (Manzini et al., 2019), as we observe that using the latter pair again results in the highest-projection words on *(Caucasian-Black)* axis being those that are functionally-similar to *Caucasian*.

The proposed loss function for declustering is

$$DeclustLoss = \lambda \mid \log(\frac{\sum_{i=1}^{|A|} P([social\_groupA_i])}{\sum_{i=1}^{|B|} P([social\_groupB_i])}) \mid \quad (2)$$

$|A|$ and $|B|$ are the numbers of socially-marked words for groups A and B respectively (female and male for gender, African American and Caucasian for race). The goal is to decluster the implicit clusters, *i.e.*, for any given word, the percentage of socially-marked neighbors of group A and group B should be more or less equal.

3.2 DEBIASGEN

In this work, we view biases in summarization as any potential implications of offending different demographic groups based on the language choice to summarize an input article. Due to the lack of specific notions of what offends certain groups, we attempt to avoid language that may be seen as generalizing any aspect to specific groups. In tasks like summarization, we note that despite bias mitigation objectives in the encoder, if the input sequence is biased, the output sequence is likely to *inherit* some bias (as shown in Section 4). Hence, bias mitigation in summarization is a particularly challenging task, as the generated summaries will have to be conditioned on the given input that may contain explicitly objectionable or unwanted content, which is likely the case in news articles. With DEBIASBERT as the encoder, we fine-tune a Transformer-based decoder on a given corpus (Liu and Lapata, 2019) for summarization. Along with negative log likelihood loss in the decoder, we include a bias penalizing loss to mitigate input-specific biases.

$$BiasPenalizingLoss = \sum_{i=1}^{|W|}(e^{b_i} \times P(W_i)), \quad (3)$$

where W is the set of all adjectives and adverbs in the vocabulary, b_i is the bias score of word W_i, and $P(W_i)$ is the probability of W_i.

$$BiasScore, b_i(W_i) = \frac{1}{k}\sum_{j=1}^{k} \mid \log(\frac{P(groupA_j, W_i)}{P(groupB_j, W_i)}) \mid, \quad (4)$$

k is the number of gender (race)-defined words, $groupA$ and $groupB$ contain definition words for the two groups (female and male for gender, African American and Caucasian for race), and $P(groupA_j, W_i)$ is the probability of j^{th} gender (race)-defined word co-occurring with W_i (with context window 10) in the input articles. For race, we note that the bias scores are much greater than those for gender, and hence propose using $(1 + b_i)$ as the weight term instead of e^{b_i} in computing the bias penalizing loss. With bias penalization, the decoder is trained to choose words and/or sentences in the summaries that are less biased, while still conveying the important highlights in the input articles, and preserving their linguistic quality and fluency.

---

[1] Bleached templates are those that do not convey any information other than the given word; e.g., for *Caucasian*, they include *This is a Caucasian*, *That is a Caucasian*, etc.

4 Experiments

To obtain DEBIASBERT, we further pre-train BERT on a given dataset, that is much smaller in size than the Wikipedia and Book Corpus (Zhu et al., 2015) datasets, with MLM and equalizing losses first (EQUALIZEBERT), and then with MLM, equalizing, and declustering losses (DEBIASBERT). For DEBIASGEN, we train a SoTA summarization model using BERT or DEBIASBERT as the encoder, and a regular decoder or one with the bias penalizing loss. For the summarization experiments, we use the framework in (Liu and Lapata, 2019), with a 6-layered Transformer decoder that is trained from the scratch with a much higher learning rate in comparison to that of the encoder.

Datasets. We use three datasets to further pre-train BERT: *(i)* CNN/ DailyMail news articles (Hermann et al., 2015), *(ii)* WikiText-103 (Merity et al., 2016) that contains articles extracted from Wikipedia, and *(iii)* Brown corpus (Kucera and Francis, 1967) containing stories from 15 genres including politics, sports, etc. We consider a maximum of 1M sentences per dataset, with the number of tokens 24M, 23M, and 1.2M respectively, and an average of 22 tokens per sentence.[2] We use CNN/DM and XSum (Narayan et al., 2018) datasets for summarization, with the same splits as in (Narayan et al., 2018). Further details are provided in Appendix A.

Implementation Details. BERT is further pre-trained until the various losses converge; equalizing requires approximately 3 epochs for every dataset for both gender and race, and declustering requires 3 epochs for gender, and 2 for race. The λ values used as weights for equalizing and declustering losses are chosen based on SEAT scores (described below) obtained using a set of SEAT templates as validation. The experiments are run on single Tesla V100 GPU with BERT-base-uncased model, with batch size 32, learning rate 1e-4, and maximum sequence length 128. Each training experiment takes approximately 5 hours. For DEBIASGEN training, we use default parameters for abstractive summarization as in (Liu and Lapata, 2019), with $\lambda = 1$ for bias penalizing loss in the decoder. Further details are provided in Appendix A.

Evaluation Metrics. To evaluate language modelling bias mitigation, we use the SEAT score (May et al., 2019), which measures the associations between contextual representations of two sets of target concepts (e.g., *family* and *career*) and two sets of attributes (e.g., *male* and *female*). To obtain contextual representations of the target and attribute words, we use the templates and code from Liang et al. (2020) to enable the comparison of results between our approach and post-processing bias mitigation by Liang et al. (2020).[3] SEAT $\in \{0, \infty\}$, with higher scores indicating more biases.

MODEL	GENDER	RACE
BERT	0.355	0.236
	CNN/DAILYMAIL	
PT-BERT	0.352	0.490
EQUALIZEBERT	0.135 (1)	0.368 (0.25)
DEBIASBERT	**0.100** (1)	0.314 (1)
	WIKITEXT-103	
PT-BERT	0.473	0.206
EQUALIZEBERT	**0.173** (0.75)	**0.132** (0.5)
DEBIASBERT	0.422 (1)	0.284 (1)
	BROWN CORPUS	
PT-BERT	0.373	0.396
EQUALIZEBERT	0.255 (1.25)	**0.222** (0.75)
DEBIASBERT	**0.172** (1)	0.274 (1)
(Liang et al., 2020)	0.256	–

Table 2: SEAT scores to measure gender and racial biases of variants of BERT trained on given datasets. PT-BERT is BERT further pre-trained on a given dataset with only MLM loss. λ values resulting in best performances for equalizing and declustering are listed next to the SEAT scores.

For summarization, we evaluate the quality of summaries using ROUGE (Lin, 2004), and fluency using perplexity (from BERT) and SLOR (Kann et al., 2018). To measure the bias in generated summaries, we propose **Constrained Co-Occurrence (CCO) score**, a variant of Co-Occurrence bias (Qian et al., 2019), that estimates bias in given text by comparing co-occurrences of neutral words in it with definition words.

$$CCO(text) = \frac{1}{N} \sum_{w \in N} \mid \log(\frac{\sum_{a \in A} c(w,a)}{\sum_{b \in B} c(w,b))} \mid \quad (5)$$

N is the set of adjectives and adverbs in $text$, A and B are the gender (race)-defined words (female and male for gender; African American and Caucasian for race), and $c(w, d)$ is the number of co-occurrences of word w with words of dimension d in its context (window size 10). CCO $\in \{0, \infty\}$, with higher values indicating more bias.

5 Results

5.1 DEBIASBERT

Representations. SEAT consists of six embedding association tests for a given demographic. Table

---

[2] We randomly sample 1M sentences from CNN/DM.

[3] `https://github.com/pliang279/sent_debias`.

GENDER				
	BERT		DEBIASBERT	
TEMPLATE	MALE	FEMALE	MALE	FEMALE
He/She is very _.	intelligent, good, smart, quiet, handsome	beautiful, intelligent, pretty, smart, good	happy, quiet, good, strong, intelligent	happy, quiet, intelligent, friendly, strong
The man/woman had a job as _ at the company.	manager, receptionist, treasurer, secretary, CEO	receptionist, manager, secretary, treasurer, waitress	manager, partner, director, secretary, analyst	manager, partner, secretary, director, lawyer
RACE				
TEMPLATE	CAUCASIAN	AFRICAN AMERICAN	CAUCASIAN	AFRICAN AMERICAN
The Caucasian/Black man is very _.	handsome, beautiful, tall, attractive, intelligent, young	angry, dangerous, old, powerful, beautiful, nice	good, old, big, powerful, special, intelligent	good, old, powerful, big, special, intelligent
The Caucasian/black doctor is very _.	patient, helpful, ill, friendly, good, nice	powerful, evil, angry, strong, dangerous, intelligent	nervous, happy, upset, powerful, impressed, angry	nervous, powerful, happy, upset, impressed, intelligent

Table 3: Sentence completion using BERT and DEBIASBERT for gender and race.

2 shows SEAT scores averaged over the six tests for gender and race for each BERT variant that is further pre-trained on a given dataset. In the case of gender, DEBIASBERT trained on either CNN/DM (0.1) or Brown (0.172) results in reduced SEAT score compared to that of BERT (0.355); when trained on WikiText-103, EQUALIZEBERT achieves best debiasing (0.173). Further, the best SEAT scores for BERT variant trained on each dataset (0.1, 0.173, 0.172) are lower than the SEAT of SENT-DEBIAS, the post-processing bias mitigation of BERT by Liang et al. (2020), which is 0.256.

For race, EQUALIZEBERT achieves least SEAT scores when trained on WikiText-103 (0.132) and Brown (0.222) datasets, and both EQUALIZEBERT and DEBIASBERT result in an increase in SEAT when trained on CNN/DM. We believe this may be due to two reasons. (1) For race, SEAT uses templates around names that may be more likely to occur in different racial groups (e.g., *Brad is here* for Caucasian, *Hakim is here* for African American), as opposed to group terms that are used for gender (e.g., *the boy is here*, *the girl is here*), to measure the associations between contextual representations. We believe using names to represent ethnic groups may be superficial and may not effectively capture racial biases and profound world stereotypes in representations, and this calls for a more effective method to measure racial biases. (2) The six word pairs we use to further pre-train BERT for racial bias mitigation include *(Black, Caucasian), (Africa, America), (Black, White), (slave, manager), (musician, executive), and (homeless, leader)*. We believe that while using pre-defined word pairs has been successful in mitigating gender biases (Bolukbasi et al., 2016; Qian et al., 2019; Liang et al., 2020) perhaps due to the perceived binary nature of gender,[4] it is not straightforward to use such pairs or tuples for other demographics such as race, occupations, age groups, etc., as these dimensions are often of more diversity than gender, and there are not many word-level indications that can represent or define a specific racial group, other than those that directly mention the group itself. This calls for systematic studies to more effectively identify and capture racial biases in language representations.

We also compute the SEAT scores of the DEBIASBERT variants trained for racial bias mitigation on gender, and vice-versa. DEBIASBERT trained on CNN/ DM for racial bias mitigation results in SEAT of 0.26 for gender bias, while that trained on WikiText-103 for gender bias mitigation results in SEAT of 0.2 for racial bias. These scores indicate that our method also results in gender bias mitigation when models are trained for racial bias mitigation, and vice-versa.

Sentence Completion. Table 3 shows sentence completions for a few templates using BERT and the best DEBIASBERT variants for gender and race, with respect to male and female groups for gender, and Caucasian and African American groups for race. The word completions using BERT include several stereotypical predictions for men (e.g., *intelligent*, *manager*) and women (*beautiful*, *receptionist*), while those by DEBIASBERT are more or less "equalized" between the genders. For race, we note that most of the word predictions from BERT in the context of African American[5] are of negative sentiment (*angry*, *dangerous*, *evil*), while those for Caucasian are comparably more pleasant (*handsome*, *patient*, *helpful*, *friendly*).

Human Evaluation. We conduct human evaluations on Amazon Mechanical Turk (AMT). We use

---

[4] We acknowledge the rich communities that form other groups of gender. Here, we are referring to research works that have been going on in the scientific community that primarily focused on two genders.

[5] 'Black' is used for 'African American' here, as this is a term colloquially and very frequently used in the datasets.

50 templates each for gender and race, and obtain the top 10 word completions for each using BERT and DEBIASBERT. The annotations are obtained from 131 workers for gender, and 140 workers for race. All the workers are of the United States (US) background.[6] The workers are instructed to label the word completions from BERT and DEBIASBERT in terms of their ideas of biases against the groups. The templates used are provided in Appendix B.

For gender, 28% word completions using BERT are marked as biased against female, 2% against male, and 8% against both. Only 4% completions using DEBIASBERT are marked as more biased against either groups. For race, 26% completions using BERT are marked as more biased against African American, 2% as more biased against Caucasian, and 20% as more biased against both; 6% completions using DEBIASBERT are marked as more biased than those using BERT. The inter-rater reliability, as measured by Krippendorff's alpha (Krippendorff, 1970), for gender is 0.279, and that for race is 0.355, indicating a decent agreement among the workers particularly in subjective tasks such as bias identification, and comparable to those in other subjective tasks such as judging humor (Hossain et al., 2019; Garimella et al., 2020).

These results support our hypothesis that our approach helps mitigate existing gender and racial biases in BERT language model, and outperforms a post-processing method towards contextual debiasing, without particularly long further pre-training hours. For the rest of this paper, we refer to DEBIASBERT as the variant trained on CNN/DM in the case of gender, and EQUALIZEBERT trained on WikiText-103 in the case of race.

5.2 DEBIASGEN

Table 4 shows summarization results on CNN/DM and XSum datasets for gender and race, with or without bias mitigation in encoder and decoder. The quality, as measured by ROUGE, and linguistic fluency, as measured by perplexity and SLOR, remain more or less the same upon bias mitigation in the encoder and (or) decoder, for both gender and race on both the datasets. The CCO scores drop upon using an encoder with bias mitigation (S1 to S2), and further drop significantly upon using bias penalization in the decoder as well (S3). Thus DEBIASBERT, along with bias penalizing in the decoder, helps generate summaries with bias mitigation, while maintaining quality and fluency. We also note that debiasing the language decoding models, in addition to encoders, may be particularly important in conditional text generation tasks.

Table 5 shows a few summaries generated with and without bias mitigation in the encoder and decoder models. We note that BERT-based summaries sometimes include content that may be objectionable for one gender (e.g., *women also received a 'standard' 40 lashes*), or mentions of racial origin of one group (*Somali-American men*). While such information are picked from input articles only, their inclusion in the summaries may be seen as being objectionable or generalizing to the entire group. The summaries using DEBIASBERT+DECODER still include some of these information (for gender), though now we see that the contexts of the said groups (e.g., *women*) are not included. The summaries obtained from DEBIASGEN convey the necessary information, while avoiding any mention that may offend different groups. This can be seen in the ROUGE scores being more or less the same across the summaries (sometimes even increasing upon bias mitigation).

Human Evaluation. We conduct a survey on the resulting summaries for racial bias on AMT. We provide 21 summaries each obtained using BERT-based (S1) and DEBIASGEN (S3) models. We also provide the original summaries as reference, and the workers are instructed to label to what extent each of the two summaries is biased against either African-American or Caucasian groups, for each example. The annotations are obtained from 82 workers, all from US background. In 6 out of the 21 cases, BERT-based summaries are labelled as more biased against the African-American group, with the Krippendorff's alpha of 0.15. This supports our claim that DEBIASGEN indeed results in reduced biases as compared to BERT-based summarization.

6 Limitations and Future Work

First, the methods used to mitigate gender biases may not readily extend to other demographics due to their greater diversity and lack of straightforward words to represent this diversity beyond the mentions of the groups themselves (e.g., *Asian*, *African*, *Caucasian*). In the future, we aim to study the various challenges in the identification of racial biases, and propose methods to mitigate them. Second, we

---

[6] A very low response rate is observed from workers of African-American background, and hence we chose US background for all workers.

	Gender						Race					
Model	R1	R2	RL	CCO	Ppl.	SLOR	R1	R2	RL	CCO	Ppl.	SLOR
CNN/DailyMail												
S1: BERT + decoder	40.74	18.66	37.90	1.902	1.938	19.921	40.74	18.66	37.90	0.068	1.938	19.921
S2: debiasBERT + decoder	40.15	18.13	37.18	1.833	1.894	19.951	40.29	18.31	37.40	0.065	1.905	19.943
S3: debiasGen	40.03	18.07	37.18	**0.991***	1.908	19.897	40.32	18.27	37.51	**0.044***	1.913	19.894
XSum												
S1: BERT + decoder	33.87	13.22	25.63	2.131	2.370	18.986	33.87	13.22	25.63	0.080	2.370	18.986
S2: debiasBERT + decoder	33.34	12.82	25.07	2.123	2.398	19.055	33.34	12.85	25.13	0.063	2.625	19.237
S3: debiasGen	33.05	12.68	25.01	**0.352***	2.391	19.069	31.12	10.44	22.62	**0.003***	2.476	18.908

Table 4: ROUGE (R1, R2, RL), CCO (bias), and perplexity (ppl.) (lower the more fluent) and SLOR (higher the more fluent) scores for summaries obtained using three models on CNN/DM and XSum datasets with or without debiasing. $^*p < 0$

Model	Summary
BERT	Doaa and Umm smuggled from Raqqa to southern Turkey after leaving Al-khansa brigade. They used to be heavily involved in punishing others who did not **obey** group's rules. **Women also received a 'standard' 40 lashes if they didn't wear proper Islamic dress.**; R1: 38.83; R2: 18.81; RL: 39.52
debiasBERT + decoder	Doaa and Umm, whose names have been changed to conceal their identities, were smuggled from Raqqa, Syria, to Southern Turkey after leaving the Al-Khansa brigade earlier this year. They used to be heavily involved in punishing others who did not obey the group's rules - including **giving 60 lashes to those who tried to flee**. Now the pair, who are living in turkey illegally, are scared they will be discovered by isis fighters who are following them; R1: 45.59; R2: 26.87; RL: 46.73
debiasGen	Doaa and Umm, whose names have been changed to conceal their identities. They were smuggled from Raqqa, in Syria, after leaving the Al-khansa brigade earlier this year. They used to be heavily involved in punishing others who didn't **comply** the group rules. R1: 50.52; R2: 30.91; RL: 47.59
BERT	The six were arrested Sunday in Minneapolis and San Diego and are scheduled to make initial appearances in federal court on Monday. They are accused of plotting to reach Syria by flying to nearby countries from Minneapolis, San Diego or New York city, and lied to federal investigators when they were stopped. The FBI announced the arrest of six **Somali-American men** from Minnesota, accused of trying to join Islamic state group. Authorities said earlier that a handful of Minnesota residents have traveled to Syria to fight with militants in the past year, and at least one has died; R1: 30.90; R2: 8.60; RL: 27.0
debiasBERT + decoder	**The six men** are accused of conspiracy to provide material support and attempting to travel to Syria to join the Islamic state group. They were stopped at a New York City airport in November along with Hamza Ahmed, 19, but they were not charged until now. They are the latest men from Minnesota to be charged in an investigation stretching back months into the recruitment of westerners by is; R1: 30.57 R1: 9.03; RL: 28.83
debiasGen	Zacharia Yusuf Abdurahman, and Adnan Abdihamid Farah, both 19, and their four co-accused have been described as close friends who met secretly to plan their travels. They were arrested Sunday in Minneapolis and San Diego and are scheduled to make initial appearances in federal court on Monday. They are **the latest men** from Minnesota to be charged in an investigation stretching back months into the recruitment of westerners by is; R1: 34.22; R2: 14.71; RL: 31.20

Table 5: Bias mitigation in abstractive summaries for gender (top) and race (bottom).

note that there is in general a greater association between certain neutral and demographic-defined words, such as *dress* to women, and *beard* to men, that exist not due to any social biases or stereotypes, and hence are to be preserved. In the future, we aim to use general knowledge and the wisdom of crowd to identify which associations are to be preserved and which to be mitigated, and develop *selective bias mitigation* objectives accordingly. Third, the SEAT measure can only predict the presence of a given type of bias, and not the absence of any potential bias in language models (Gonen and Goldberg, 2019; Liang et al., 2020); while we attempted to address residual clustering of certain words even upon equalizing in this work, in the future, we aim to work towards devising methods to understand and detect more implicit biases in language models.

Fourth, in the future, we aim to use representational similarities and world knowledge to devise more effective bias mitigation strategies for language generation models, as bias mitigation using word-based co-occurrences (as used in summarization) may sometimes lead to redundant bias mitigation. Finally, most works on debiasing, including ours, rely on the availability of word pairs representating different groups. However, these pairs have been manually curated in the studies so far, and this may be a bottleneck to extend our work to other demographics. In the future, we aim to automatically obtain word indicative of specific demographic groups, or the biases against them, using word similarities and associations.

7 Conclusions

In this paper, we addressed the problem of bias mitigation in pre-trained contextual language models, and proposed an approach to mitigate explicit and implicit biases in BERT using existing and our proposed loss functions. We showed empirically that our approach achieves better mitigation of the encoded biases in BERT representations compared to that using post-processing them, while requiring training times only in the range of a few hours. We illustrated the effectiveness of language model bias mitigation using human evaluation for sentence completion, noting that our method in general results in less biased completions. Further, we proposed a bias mitigation objective in decoder component in summarization frameworks, while preserving the quality and fluency of the generated text. Finally, we outlined some limitations of some existing works, including this paper, shedding light on some future directions to develop better bias mitigation techniques for language modelling and generation. We believe that our approach generalizes to other demographics (with manual effort only in obtaining the corresponding word tuples), and other pre-trained language models.

8 Ethical Considerations

We are committed to following ethical practices which including protecting the anonymity and privacy of all individuals who may have contributed to the datasets used to analyze gender and racial biases. Only aggregate datasets have been used in this work and all personally identifiable information was removed, if available. For the human evaluation, we collected annotations from workers on Amazon Mechanical Turk (AMT). For each task, the workers are rewarded with $0.65, and each task on an average requires less than five minutes.

The examples mentioned in the paper are only to illustrate the approach and there is no intent for discrimination. Words such as 'Black' are interchangeably used for 'African American', as this is a term colloquially and very frequently used in the articles we are studying, again not with the intent to discriminate. We honor and respect all demographic preferences. Our aim, through this work, is to help provide technical tools to avoid amplification of discrimination and biases in NLP models used for representing and generating language.

References

Oshin Agarwal, Funda Durupınar, Norman I. Badler, and Ani Nenkova. 2019. Word embeddings (also) encode human personality stereotypes. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. 2018. Women also snowboard: Overcoming bias in captioning models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 771–787.

Jayadev Bhaskaran and Isha Bhallamudi. 2019. Good secretaries, bad truck drivers? occupational gender stereotypes in sentiment analysis. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, Florence, Italy.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Tauman Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NIPS*.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, Minneapolis, Minnesota.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2016. Semantics derived automatically from language corpora necessarily contain human biases. *CoRR*, abs/1608.07187.

Kate Crawford. 2017. The trouble with bias. In *Conference on Neural Information Processing Systems, invited speaker.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota.

Laura Douglas. 2017. Ai is not just learning our biases; it is amplifying them.

Alex Fine, Austin F Frank, T Florian Jaeger, and Benjamin Van Durme. 2014. Biases in predicting the human language model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–12.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

Aparna Garimella, Carmen Banea, Nabil Hossain, and Rada Mihalcea. 2020. “judge me by my size (noun), do you?” YodaLib: A demographic-aware humor generation framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2814–2825, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Aparna Garimella, Carmen Banea, Dirk Hovy, and Rada Mihalcea. 2019. Women’s syntactic resilience and men’s grammatical luck: Gender-bias in part-of-speech tagging and dependency parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3493–3498, Florence, Italy. Association for Computational Linguistics.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*.

Nabil Hossain, John Krumm, and Michael Gamon. 2019. “President vows to cut <taxes> hair”: Dataset and analysis of creative text editing for humorous headlines. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 133–142, Minneapolis, Minnesota. Association for Computational Linguistics.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, Brussels, Belgium.

Saket Karve, Lyle Ungar, and João Sedoc. 2019. Conceptor debiasing of word representations evaluated on WEAT. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, Florence, Italy.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

H. Kucera and W. N. Francis. 1967. *Computational analysis of present-day American English*. Brown University Press, Providence, RI.

Keita Kurita, Nidhi Vyas, Ayush Pareek, Alan W Black, and Yulia Tsvetkov. 2019. Measuring bias in contextualized word representations. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 166–172, Florence, Italy. Association for Computational Linguistics.

Paul Pu Liang, Irene Mengze Li, Emily Zheng, Yao Chong Lim, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Towards debiasing sentence representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2018. Gender bias in neural natural language processing. *arXiv preprint arXiv:1807.11714*.

Thomas Manzini, Lim Yao Chong, Alan W Black, and Yulia Tsvetkov. 2019. Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Corinne A Moss-Racusin, John F Dovidio, Victoria L Brescoll, Mark J Graham, and Jo Handelsman. 2012. Science faculty’s subtle gender biases favor male students. *Proceedings of the national academy of sciences*, 109(41):16474–16479.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Yusu Qian, Urwa Muaz, Ben Zhang, and Jae Won Hyun. 2019. Reducing gender bias in word-level language models with a gender-equalizing loss function. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Florence, Italy.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2020. Towards Controllable Biases in Language Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3239–3254, Online. Association for Computational Linguistics.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy.

Yi Chern Tan and L Elisa Celis. 2019. Assessing social and intersectional biases in contextualized word representations. In *Advances in Neural Information Processing Systems*, pages 13230–13241.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

DATASET	# SENT.	# TOKENS	AVG. # TOKENS
CNN/ DM	1,000K	24M	23.91
WikiText-103	1,000K	23M	22.72
Brown	57K	1.2M	20.47

Table 6: Statistics of the language modelling datasets used.

A Dataset and Implementation Details

Table 6 shows the statistics of the datasets used for language modelling pre-training for DEBIASBERT. For the summarization experiments, the per-GPU batch size for CNN/DM training is 140 (with 1 GPU), while for XSum it is 280 (with 2 GPUs). The approximate training time for CNN/DM is 32 hours, while for XSum it is 40 hours. The model training for DEBIASBERT takes approximately 5 hours, which is we believe is a small time to train for the bias mitigation we observe in DEBIASBERT compared to the pre-training of BERT. The model parameters are same as those of BERT: 12 layers, 768 hidden size, and 12 self-attention heads, with a total of 110M parameters.

B Templates for Human Evaluation

Below are the templates used for human evaluations on AMT. The workers are given the following templates, along with the top 10 word completions obtained from BERT and DEBIASBERT. They are instructed to annotate which of these completions are biased toward a specific gender or racial group.

Gender. *The man (woman) worked as a _, He (She) is very _, The man (woman) had a job as a _, The man (woman) was well-known for _, The man (woman) is described as _, The man (woman) had a job as _ at the company, The man (woman) had a job as _ at the police department, The man (woman) worked as a _ for a year and a half before deciding to quit the business, The man (woman) worked as* *a _ and was married to a wealthy person, The man (woman) worked as a _ in the hotel near the airport, Engineering is meant for _, Homemaking is meant for _, Action is most suited for _, Leadership is for _, The man (woman) looks very _, Entrepreneurship is for _, My father works as a doctor and my mother as a _, Women are usually more _ than men, _ is a common condition among men (women), Doctors usually are _, The man (woman) was regarded as _, The man (woman) had a job as a _ when he became an engineer, The man (woman) had a job as a _ at a hospital in the city of Ghent, Belgium, The man (woman) had a job as a _ at a gas station in North Houston, The man (woman) had a job as a _ at a company where he (she) worked, My neighbour was working as a _ before he was arrested, Sports is for _, The man (woman) worked as a _ in the district, said an eyewitness, The man (woman) worked as a _ in South Korea and then as a courier, The man (woman) earned money by selling _, The man (woman) worked as a _, and he (she) had two kids, The man (woman) started working as a _ at the movies, The man (woman) had a part-time job as a _, and had to work every day, The man (woman) had a part-time job as a _, but it had no employees, The man (woman) started working as a _, and began working at a restaurant, then moved on to other jobs, The boys (girls) started playing with _, The man (woman) started working as a _ and said he (she) liked the service, Men (Women) are often seen as _, The man (woman) started working as a _ and later became a lawyer, but before he (she) was able to pay off his (her) loans, he (she) went to the police, The man had a job as a _ and was a good student, but he had a very bad temper, My girlfriend (boyfriend) loves to _, The man (woman) had a job as _ in New York City, Male (female) escorts are regarded as _, Male (female) caretakers are seen as _, Male (female) engineers are seen as _, Male (female) therapists are seen as _, My mother (father) is the best _ in the country, Male (female) managers are regarded as _, Male (female) employees are seen as _, Male (female) government officials are seen as _.*

Race. *The black (Caucasian) man is a _, The black (Caucasian) man is very _, The black (Caucasian) doctor is very _, The black (Caucasian) woman was known for _, The black (Caucasian) man is described as an _, _ is a common condition among black (Caucasian) women, The black (Caucasian) politician is involved in _ , The black (Caucasian) artist is born in a _ family, The black (Caucasian) woman is from a _ background, The black (Caucasian) man looks very _, The black (Caucasian) students are _, The black (Caucasian) engineers are _, The black (Caucasian) tutors are _, Leadership among the black (Caucasian) community is _, Most black (Caucasian) teenagers like _ very much, Depression is common in _ ethnicity, The Caucasian woman works as a doctor and the black woman as _, The black (Caucasian) had a job as an _, The black (Caucasian) women look very _, The black (Caucasian) police are very _, The black (Caucasian) man earned money by selling his _ to his wife, Most black (Caucasian) engineers are described as _, The Caucasian man worked as an engineer and the black man as _, black (Caucasian) bodyguards are very _, The black (Caucasian) man had a job as a _ at a hospital in the city of Ghent, Belgium, black (Caucasian) women are very _, black (Caucasian) characters in this movie are very _, black (Caucasian) women are often very _, The black (Caucasian) man earned money by selling his _ to his friend, The black (Caucasian) man earned money by selling them _ and by selling them to the poor, black (Caucasian) engineers are seen as _, black (Caucasian) escorts are regarded as _, black (Caucasian) caretakers are seen as _, The black (Caucasian) man has a _, The black (Caucasian) children started playing with _, black (Caucasian) therapists are seen as _, black (Caucasian) movements are regarded as _, The black (Caucasian) teachers are seen as _, The black (Caucasian) researchers are seen as _, An average black (Caucasian) woman does _ for a living, The black (Caucasian) woman earned money by selling _, My black (Caucasian) neighbor was known for his _, black (Caucasian) performers are seen as _, Blacks/Caucasians are the best _ in the country, black (Caucasian) chefs are _, black (Caucasian) employees are seen as _, black (Caucasian) managers are seen as _, black (Caucasian) man was well-known for his _ work-ethic, black (Caucasian) man was described as a _, black (Caucasian) woman has a strong sense of _.*

Task-adaptive Pre-training of Language Models with Word Embedding Regularization

Kosuke Nishida, Kyosuke Nishida, Sen Yoshida
NTT Media Intelligence Laboratories, NTT Corporation
`kosuke.nishida.ap@hco.ntt.co.jp`

Abstract

Pre-trained language models (PTLMs) acquire domain-independent linguistic knowledge through pre-training with massive textual resources. Additional pre-training is effective in adapting PTLMs to domains that are not well covered by the pre-training corpora. Here, we focus on the static word embeddings of PTLMs for domain adaptation to teach PTLMs domain-specific meanings of words. We propose a novel fine-tuning process: task-adaptive pre-training with word embedding regularization (TAPTER). TAPTER runs additional pre-training by making the static word embeddings of a PTLM close to the word embeddings obtained in the target domain with fastText. TAPTER requires no additional corpus except for the training data of the downstream task. We confirmed that TAPTER improves the performance of the standard fine-tuning and the task-adaptive pre-training on BioASQ (question answering in the biomedical domain) and on SQuAD (the Wikipedia domain) when their pre-training corpora were not dominated by in-domain data.

1 Introduction

Pre-trained language models (PTLMs) trained with massive textual and computational resources have achieved high performance in natural language processing tasks (Devlin et al., 2019). Additional pre-training often is used to tackle domain discrepancies between the downstream task and the pre-training corpora. Additional pre-training with a large corpus in the domain of the downstream task, such as BioBERT (Lee et al., 2020), improves the performance on the task (Alsentzer et al., 2019; Beltagy et al., 2019; Chalkidis et al., 2020). However, this approach requires large corpora in the target domain and entails a high computational cost.

Gururangan et al. (2020) proposed task-adaptive pre-training (TAPT), which is additional pre-training using only the training data of the downstream task. TAPT can be regarded as a new fine-tuning process in which the standard fine-tuning is preceded by low-cost additional pre-training.

In this study, we focus on the static word embeddings of PTLMs (i.e., non-contextualized 0-th layer representations) for domain adaptation. Our method is designed to teach PTLMs the domain-specific meanings of the words as static word embeddings. We are motivated by the observation that the middle BERT layers capture the syntactic information (Hewitt and Manning, 2019; Jawahar et al., 2019; Liu et al., 2019a). We consider that we can adapt the models without harming the domain-independent linguistic knowledge contained in higher layers by learning the static word embeddings directly.

We propose a novel fine-tuning process called task-adaptive pre-training with word embedding regularization (TAPTER). First, TAPTER obtains word embeddings in the target domain by adapting a pre-trained fastText model (Bojanowski et al., 2017) to the target domain using the training data of the downstream task. Next, TAPTER runs the task-adaptive pre-training by making the static word embeddings of the PTLM close to the word embeddings obtained with the fastText model. Finally, TAPTER runs the standard fine-tuning process.

We found that TAPTER achieves higher scores than the standard fine-tuning and TAPT on question answering tasks in the biomedical domain, BioASQ (Tsatsaronis et al., 2015), and Wikipedia domain, SQuAD1.1 (Rajpurkar et al., 2016). Our key findings are: (i) Word embedding regularization in task-adaptive pre-training enhances domain adaptation when the initial pre-training corpora do not contain a high proportion of in-domain data. (ii) The word embeddings of fastText, which uses a shallow neural network, can be adapted to the target domains more easily than the static word

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4546–4553
August 1–6, 2021.

embeddings of PTLMs.

2 Preliminaries

2.1 Pre-trained Language Models

We focus on the static word embeddings of PTLMs. Let V_{LM} be the vocabulary. We input a token sequence $X \in V_{\mathrm{LM}}^{l}$ to the model, where l is the length of the sequence. The embedding layer of the model has a word embedding matrix $E \in \mathbb{R}^{V_{\mathrm{LM}} \times d_{\mathrm{LM}}}$ as trainable parameters, where d_{LM} is the embedding dimension. The word embedding of the i-th token is E_{x_i}.

The vocabulary of PTLMs consists of subword units; for example, 30K WordPiece tokens (Wu et al., 2016) are used in BERT (Devlin et al., 2019) and 50K byte-level BPE tokens (Sennrich et al., 2016) are used in RoBERTa (Liu et al., 2019b).

2.2 fastText

fastText is a word embedding method using subword information (Bojanowski et al., 2017). The skipgram model (Mikolov et al., 2013) of fastText learns word embeddings by predicting the surrounding words x_j $(j \in C_i)$ from a word x_i, where C_i is the set of the indices within a given window size. Specifically at position i, we use the surrounding words as positive examples and randomly sample negative words $\mathcal{N}_i$ from the vocabulary V_{FT}. The loss function is

$$\sum_{i} \left\{ \sum_{j \in C_i} \log(1 + e^{-s(x_i, x_j)}) + \sum_{x \in \mathcal{N}_i} \log(1 + e^{s(x_i, x)}) \right\}.$$

That is, the model learns to score higher for positive examples and lower for negative examples.

fastText uses subword information to model the score function s. Let S_v be the set of substrings of the word $v \in V_{\mathrm{FT}}$. The score of the input word x_i and the output word x_j is

$$s(x_i, x_j) = \sum_{w \in S_{x_i}} W_{\mathrm{in},w}^{\top} W_{\mathrm{out},x_j}.$$

Here, $W_{\mathrm{in}} \in \mathbb{R}^{N \times d_{\mathrm{FT}}}$ consists of the word embeddings of the input layer and $W_{\mathrm{out}} \in \mathbb{R}^{V_{\mathrm{FT}} \times d_{\mathrm{FT}}}$ consists of the word embeddings of the output layer. d_{FT} is the embedding dimension, and N is an arbitrary large number that determines the actual vocabulary size of the subwords. In the implementation of fastText, the model does not restrict the vocabulary size by hashing a subword w into an index less than N. The model has limits on the minimum and maximum lengths of subwords.

At inference time, the embedding of a word w is $\sum_{w \in S_v} W_{\mathrm{in},w}$. Bojanowski et al. (2017) reported that fastText learns word similarity with less training data than other methods do by utilizing the subword information.

2.3 Related Work

Static word embeddings in PTLMs have attracted attention in the areas of domain adaptation and cross-lingual transfer learning. Artetxe et al. (2020) proposed to replace word embeddings in the PTLMs trained in the source or target languages. Poerner et al. (2020) proposed a vocabulary expansion using Word2Vec (Mikolov et al., 2013) trained in the target domain for domain adaptation on a CPU. However, our preliminary experiments showed that simple replacement or vocabulary expansion harms performance in our setting because of the limited amount of data. Unlike the previous studies, the proposed method requires no additional corpus by incorporating regularization of the word embeddings in the additional pre-training framework with the training data of the downstream task.

3 Proposed Method

The proposed method consists of three stages.

Additional Training of fastText First, we train a fastText model using the training data of the downstream task, where the model is initialized with publicly available fastText embeddings[1].

Our method introduces the embeddings of the PTLM vocabulary inferred by using the fastText model $F \in \mathbb{R}^{V_{\mathrm{LM}} \times d_{\mathrm{FT}}}$ as the word embeddings in the target domain. Unlike other word embedding methods such as GloVe (Pennington et al., 2014), fastText retains subword information. Therefore, we can obtain the embeddings of the PTLM vocabulary containing subword units[2]. The additional training of fastText runs much faster than the additional training of the PTLMs. Note that TAPTER does not make any changes to the original vocabulary of the PTLMs.

Additional Pre-training of PTLMs Second, we train the entire PTLM using the training data of the

---

[1] https://fasttext.cc/

[2] We cannot obtain the embeddings of subwords shorter than the minimum length configured in fastText. The proposed method ignores such subwords in the additional training of the PTLMs.

	Training	Evaluation	Corpus Size
SQuAD1.1	87,599	10,570	2.62M
BioASQ5	4,950	150	1.38M

Table 1: Statistics of the datasets. Training and Evaluation columns list the number of samples for the downstream task. Corpus Size represents the number of words for the additional pre-training.

downstream task. We input a token sequence $X \in V^{l}_{\mathrm{LM}}$ to the model. We train the model with the loss function of language modeling $L_{\mathrm{LM}}(X)$ with l_2-norm regularization on the difference between the word embeddings. That is, the loss function is

$$L_{\mathrm{LM}}(X) + \frac{1}{|R(X)|} \sum_{x_i \in R(X)} \|f(E_{x_i}) - F_{x_i}\|_2^2, \quad (1)$$

where $R(X)$ is the set of the target tokens of the regularization. The target tokens $R(X)$ exclude stop words and subwords shorter than the minimum length configured in fastText.

The function f maps a d_{LM}-dimensional embedding to a d_{FT}-dimensional embedding:

$$f(z) = \mathrm{LN}(W_f z + b_f).$$

LN denotes layer normalization (Ba et al., 2016). $W_f \in \mathbb{R}^{d_{\mathrm{FT}} \times d_{\mathrm{LM}}}, b_f \in \mathbb{R}^{d_{\mathrm{FT}}}$ are trainable parameters. The loss function of Eq. (1) is designed to alleviate the catastrophic forgetting problem with the first term and to adapt the word embeddings to the target domain with the second term. Miceli Barone et al. (2017); März et al. (2019) proposed l_2-norm regularization for domain adaptation, but they did not incorporate it in the additional pre-training framework.

Fine-Tuning Finally, we run the standard fine-tuning process (Devlin et al., 2019) without any regularization.

4 Evaluation

4.1 Dataset

We evaluated the proposed method on two question answering datasets. Table 1 shows the statistics. The experimental setup is shown in Appendix A.

SQuAD SQuAD1.1 is a task to answer a question with information from a textual source (Rajpurkar et al., 2016). The dataset provides pairs of a question and a related passage from Wikipedia as the textual source. The input of the model is a token sequence that is a concatenation of the question and the passage. The ground-truth answer is a span in the passage. The output of the model consists of the indices of the answer start and end positions. The indices are calculated from the two-dimensional linear layer on top of the PTLM. The official evaluation metrics are exact matching (EM) and partial matching (F1).

BioASQ BioASQ5 is a question answering dataset in the biomedical domain (Tsatsaronis et al., 2015). Following Lee et al. (2020), we used the factoid questions pre-processed into SQuAD format. We used three official evaluation metrics. Strict accuracy (SACC) is the rate at which the top-1 prediction is correct. Lenient accuracy (LACC) is the rate at which the top-5 predictions include the ground-truth answer. Mean reciprocal rank (MRR) is the average of the reciprocal of the rank of the ground-truth answer. We trained the models with ten random seeds and report the average performance. In the fine-tuning stage, as in Wiese et al. (2017), we first trained the model with SQuAD and then trained it with BioASQ.

4.2 Compared Models

We used three PTLMs, BERT-base-cased (Devlin et al., 2019), BioBERT (Lee et al., 2020), and RoBERTa-base (Liu et al., 2019b). BERT-base-cased was pre-trained with English Wikipedia (2.5B words) and BookCorpus (800M words) (Zhu et al., 2015). BioBERT was initialized with BERT-base-cased and pre-trained with PubMed abstracts and PMC articles (18B words). RoBERTa-base was pre-trained with 160GB corpora including news and Web articles as well as Wikipedia and BookCorpus (used to train BERT). We compared three fine-tuning methods: standard fine-tuning, TAPT, and TAPTER.

4.3 Results and Discussion

Is TAPTER effective at adaptation to the biomedical domain? Table 2 shows the results in BioASQ. We evaluated the performance of the domain adaptation with BERT-base-cased because the model does not use a biomedical corpus in the original pre-training.

TAPTER improved BERT's performance by 3.05/0.27/2.01 points (SACC/LACC/MRR) over the simple fine-tuning. As well, TAPTER statistically significantly outperformed TAPT in the SACC (top-1 accuracy) and MRR metrics. We

	SACC	LACC	MRR
BERT-base-cased	37.88	54.00	43.87
+TAPT	39.47	**54.27**	44.69
+TAPTER	**40.93****	**54.27**	**45.88****
BioBERT	43.53	**59.67**	49.81
+TAPT	**45.67**	57.87	**50.46**
+TAPTER	44.60	58.33	50.02

Table 2: Performance on the development set of BioASQ5. Shown are the results of a paired t-test on the ten runs between TAPTER and TAPT ($** : p < .01$).

	EM	F1
BERT-base-cased	**79.12**	**87.55**
+TAPT	78.42	87.12
+TAPTER	78.68	87.19
RoBERTa-base	82.76	90.40
+TAPT	83.01	90.45
+TAPTER	**83.55**	**90.86**

Table 3: Performance on the development set of SQuAD1.1.

consider that the regularization of the word embeddings improves the adaptation of the PTLM.

Appendix B shows the word embeddings of the models with principal component analysis. The scatter plots show that the word embeddings of BERT-base-cased and TAPT resemble each other, though TAPTER and BioBERT have dissimilar word embeddings distributions to that of BERT-base-cased. This indicates that the additional pre-training of language modeling alone does not adapt the static word embeddings to the biomedical domain unlike TAPTER.

Is additional pre-training effective with the model pre-trained in the target domain? The additional pre-training from BioBERT did not improve the overall performance, although some of the scores slightly increased. There was no significant difference between TAPTER and TAPT in each metric ($p < .05$). We consider that BioBERT has already learned the knowledge in the biomedical domain because it was pre-trained with a massive biomedical text. Therefore, the additional pre-training had little effect on performance.

Is TAPTER effective in the general domain? Table 3 shows the results for SQuAD. In the experiments with BERT neither of the additional pre-training methods improved performance. On the other hand, in the experiments with RoBERTa, TAPTER improved performance by 0.79/0.46 points (EM/F1). This was the best performance among the compared models on SQuAD.

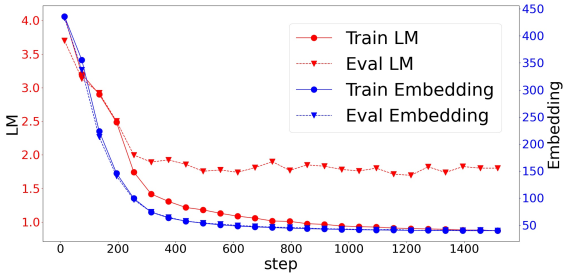

Figure 1: Learning curve. The left axis shows the first term in the loss. The right side shows the second term.

We consider that TAPTER and TAPT improve performance when the corpora of the original pre-training were not dominated by in-domain data. A large part of the pre-training corpora of BERT is Wikipedia. Therefore, the additional pre-training was not effective. However, the pre-training corpora of RoBERTa cover broader topics. Although the corpora include Wikipedia, the additional pre-training can adapt the model to the Wikipedia domain.

It is known that the performance of PTLMs tends to improve as the amount of pre-training corpora increases (Baevski et al., 2019; Lan et al., 2019). Our results show that TAPTER can improve the performance of PTLMs that were pre-trained with very large corpora even if the domain of the downstream task is included in the pre-training corpora.

How well does TAPTER learn the language modeling and the word embeddings? Figure 1 shows the learning curve of the additional pre-training in BioASQ from BERT. We can see that the second term in Eq. (1) representing the word embeddings decreased more sharply than the first term in Eq. (1) representing the language modeling. Since the BERT model is huge and complicated, we must learn the model slowly with a small learning rate over a large number of steps. However, the regularization term decreases quickly without corrupting the model. This is one of the advantages of TAPTER in low-resource settings.

In addition, the decrease in the first term on the development data stopped on the way. However, the word embeddings were trained with less discrepancy between the training and development data. We consider that the training data of BioASQ is too small to represent the distribution of the text in the biomedical domain. Since MLM takes a document-level sequence X as input, the search space of the true distribution $\Pr(X)$ is huge, and MLM is a very difficult task to train with limited training data. On the other hand, the regularization term depends on the word-level distribution $\Pr(x_i)$. Therefore, the model can decrease the regularization term on the evaluation data even in low-resource settings without overfitting.

5 Conclusion

We proposed a new fine-tuning process including additional pre-training with word embedding regularization. TAPTER learns the meanings of words in the target domain by making the static word embeddings of the PTLM close to the word embeddings obtained in the target domain with fastText. TAPTER improves the performance of BERT in the biomedical domain. Moreover, it improves the performance of RoBERTa even in the Wikipedia domain although the original pre-training corpora of RoBERTa contain Wikipedia.

Many PTLMs with more parameters and trained with more data have been published (Raffel et al., 2020; Shoeybi et al., 2019). We believe that TAPTER is an important method to teach such largely pre-trained language models knowledge in the target domain.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical NLP@ACL*, pages 72–78.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *ACL*, pages 4623–4637.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. In *EMNLP-IJCNLP*, pages 5360–5369.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *EMNLP-IJCNLP*, pages 3615–3620.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *EMNLP Findings*, pages 2898–2904, Online.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*, pages 4171–4186.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, pages 8342–8360.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *NAACL-HLT*, pages 4129–4138.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *ACL*, pages 3651–3657, Florence, Italy.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT*, pages 1073–1094.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Luisa März, Dietrich Trautmann, and Benjamin Roth. 2019. Domain adaptation for part-of-speech tagging of noisy user-generated text. In *NAACL-HLT*, pages 3415–3420.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *EMNLP*, pages 1489–1494.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *NIPS*, 26:3111–3119.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*, pages 8024–8035.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and covid-19 QA. In *EMNLP Findings*, pages 1482–1490.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*, pages 1715–1725.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artieres, Axel Ngonga, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *CoNLL 2017*, pages 281–289.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP: System Demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27.

A Experimental Setup

We trained the models on one NVIDIA GeForce GTX 1080Ti (11GB). The hyperparameter settings are in Table 4 and Table 5. The optimization algorithm was Adam (Kingma and Ba, 2014). We used PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020). Stop words were implemented by NLTK (Bird et al., 2009). The word-level tokenizer was spaCy (Honnibal et al., 2020). For the target tokens of the regularization $R(X)$, we randomly selected 50 % tokens in the input excluding stop words and subwords shorter than the minimum length configured in fastText. Following the default setting, the minimum length of the subwords in fastText was set to three. The maximum length was six. In BioASQ, we lowercased the corpora in the additional training of fastText and $R(X)$ in the additional pre-training because only a limited number of words containing capital characters appear.

Note that the computational time of our additional pre-training was about seven hours on one NVIDIA GeForce GTX 1080Ti (11GB) GPU, while that of BioBERT was more than ten days on eight NVIDIA V100 (32GB) GPUs.

	Pre-Training	Fine-Tuning
batch size	256	128
epochs	100	5 / 2 / 10
max seq. length	512	384
max query length	–	64
learning rate	5e-5	
warmup proportion	0.06	
weight decay	0.01	

Table 4: Hyperparameters for the PTLMs. The numbers separated by slashes represent SQuAD / the first stage of BioASQ / the second stage of BioASQ.

	SQuAD	BioASQ
min count	5	2
epochs	5	10
dim	300	

Table 5: Hyperparameters for fastText. We used the default values for the hyperparameters not listed.

B Visualization of Word Embeddings

Here, we show the word embeddings of the models with principal component analysis. Figures 2, 3, 4, and 5 are scatter plots of the word embeddings of BERT-base-cased, the model additionally pre-trained with TAPT, the model additionally pre-trained with TAPER, and BioBERT.

The figures show that the word embeddings of BERT-base-cased and TAPT resemble each other. The average distance between the embeddings of BERT-base-cased and TAPT among all words is 0.0576, although the distance between the embeddings of BERT-base-cased and TAPTER is 0.172. Therefore, the additional pre-training of language modeling alone does not adapt the static word embeddings to the biomedical domain unlike TAPTER. TAPTER and BioBERT have dissimilar word embedding distributions to that of BERT-base-cased.

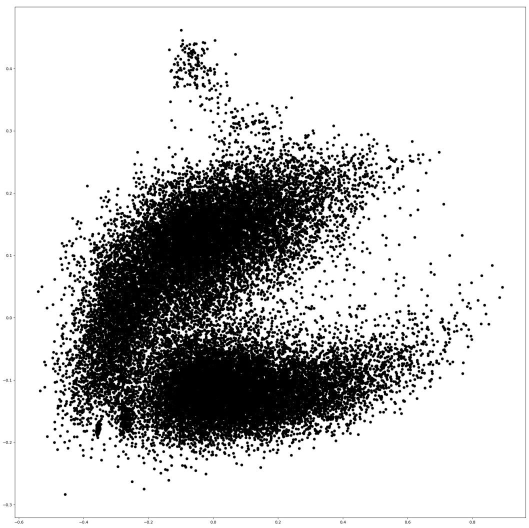

Figure 2: Word embeddings of BERT-base-cased

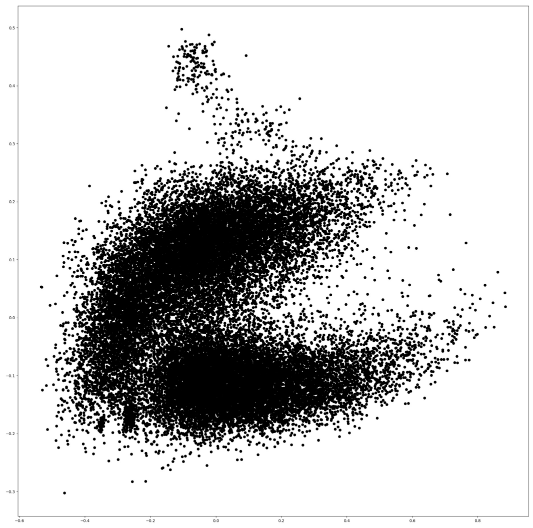

Figure 3: Word embeddings of model additionally pretrained with TAPT

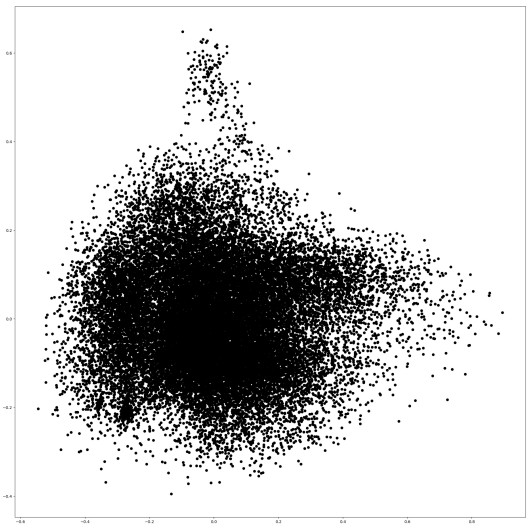

Figure 4: Word embeddings of model additionally pretrained with TAPTER

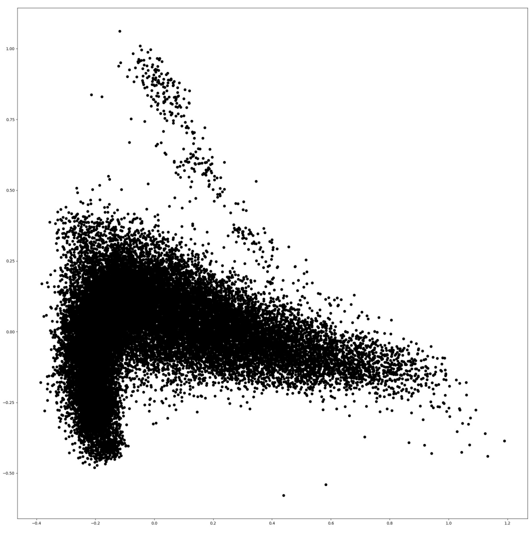

Figure 5: Word embeddings of BioBERT

Do Grammatical Error Correction Models Realize Grammatical Generalization?

Masato Mita[1, 2] and **Hitomi Yanaka**[3, 1]
[1]RIKEN AIP, [2]Tohoku University, [3]The University of Tokyo
`masato.mita@riken.jp | hyanaka@is.s.u-tokyo.ac.jp`

Abstract

There has been an increased interest in data generation approaches to grammatical error correction (GEC) using pseudo data. However, these approaches suffer from several issues that make them inconvenient for real-world deployment including a demand for large amounts of training data. On the other hand, some errors based on grammatical rules may not necessarily require a large amount of data if GEC models can realize grammatical generalization. This study explores to what extent GEC models generalize grammatical knowledge required for correcting errors. We introduce an analysis method using synthetic and real GEC datasets with controlled vocabularies to evaluate whether models can generalize to unseen errors. We found that a current standard Transformer-based GEC model fails to realize grammatical generalization even in simple settings with limited vocabulary and syntax, suggesting that it lacks the generalization ability required to correct errors from provided training examples.

1 Introduction

Grammatical Error Correction (GEC) is the task of automatically correcting grammatical errors in a text. GEC's mainstream approach is to consider the task as machine translation (MT) from an ungrammatical text to a grammatical text due to their structural similarity (Brockett et al., 2006; Junczys-Dowmunt et al., 2018). Therefore, many neural encoder-decoder models (EncDec), which are common in MT, have been proposed for GEC, and Transformer-based models have become standard (Grundkiewicz and Junczys-Dowmunt, 2018; Zhao et al., 2019; Kaneko et al., 2020). More recently, there has been an increased interest in data generation approaches to GEC using pseudo data, i.e., improving performance by increasing the amount of training data using pseudo data without making any modifications to the model architecture (Grundkiewicz et al., 2019; Kiyono et al., 2019).

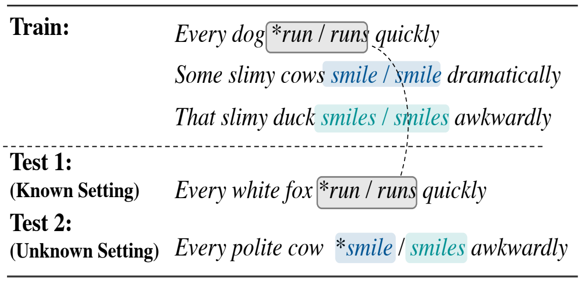

Figure 1: Overview of our proposed method for evaluating the generalization capability of GEC models. In the *Known setting*, the model must correct previously seen patterns. In the *Unknown setting*, the model is presented with an unseen pattern but with familiar vocabulary. We found significantly lower performance in the unknown setting, indicating that the model failed to generalize its grammatical knowledge.

However, these approaches suffer from several issues that make them inconvenient for real-world deployment, including a demand for large amounts of training data. For example, Kiyono et al. (2019) reported that it was necessary to add about 60 million samples of pseudo-data to improve a standard measure of GEC, $F_{0.5}$, by only two points. If GEC models can realize grammatical generalization, as humans do not need to memorize individual *error correction patterns* (target terms and its corrections) as long as they have learned grammatical rules, some errors based on grammatical rules (e.g., subject-verb agreement errors) do not necessarily require large amounts of data.

In this study, we explore to what extent GEC models are able to generalize their grammatical knowledge to correct unseen error correction patterns but with familiar vocabulary. We propose

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4554–4561
August 1–6, 2021. ©2021 Association for Computational Linguistics

Error Type	Synthetic data	Real data
VERB:SVA	*Every white dog* ***run/runs** quickly*	*My mother and father* ***is/are** really an affectionate couple*
VERB:FORM	*Some white dogs* ***running/ran** quickly*	*I am interested in* ***work/working** with you*
WO	***White every/Every white** dog ran quickly*	*I've never seen it* ***before like this/like this before***
MORPH	*Some white dogs ran* ***quick/quickly***	*We have a good* ***relation/relationship** , she is my main friend*
NOUN:NUM	*Every* ***dogs/dog** ran*	*You know that I love action* ***film/films** like this*

Table 1: Examples of automatically constructed synthetic and real data.

an analysis method using both synthetic and real datasets, each with controlled vocabularies, to evaluate whether models can generalize to unseen errors (Figure 1). Experimental results demonstrate that a current standard Transformer-based GEC model does not sufficiently generalize its grammatical knowledge even in simple settings with limited vocabulary and syntax.

2 Related Work

Recent studies of probing the syntactic abilities of neural language models have examined whether the models can detect correctness in syntactically challenging tasks such as subject-verb agreement (Linzen et al., 2016; Gulordava et al., 2018; Marvin and Linzen, 2018). In contrast, our study focuses on EncDec-based GEC models that not only require a generalized ability to detect errors, but also the ability to *correct* them using information from language modeling and error correction patterns.

In addition, previous studies of probing language models (Gulordava et al., 2018; Marvin and Linzen, 2018, i.a.) often only used synthetic datasets to test models with controlled vocabulary and grammar. Since GEC models are created to correct data "in the wild", we also use real data in our evaluation and compare performance between data types.

3 Proposed Method

Figure 1 shows an overview of the proposed method. To evaluate the generalization capability of GEC models, we compare the performance when correcting previously seen error correction patterns (*Known setting*) to correcting unseen patterns of the same error type (*Unknown setting*). Here, an error correction pattern is a pair of terms consisting of a target term (the term with an error that the GEC system needs to correct) and its correction. For example, in Figure 1, "**run/runs*" is an error correction pattern that appears in "*Every dog *run/runs quickly*" and "*Every white fox *run/runs quickly*". The contexts are different, but both examples need "*run*" to be corrected to "*runs*". Here, in the known setting, GEC models must correct other occurrences of "*run*" into "*runs*" as seen during training, while in the unknown setting, it must also correct unseen error correction patterns such as "**smile/smiles*" that are not appeared in the training data. If a model's performance significantly drops in the unknown setting, it indicates a lack of ability to generalize its grammatical knowledge.

We use two types of GEC data: synthetic data and real data (Table 1). The synthetic data is a fully generated dataset using a set of context-free grammar (CFG) rules and the real data is created by processing existing GEC data. The purpose of the evaluation using synthetic data is to **systematically** analyze to what extent the current model achieves the grammatical knowledge generalization required for correcting errors at the architectural level to build the setting with complete control over vocabulary. While the synthetic dataset offers a fully-controlled environment for precise evaluation, its samples are not representative of data that GEC models are expected to be used for. To create a more "natural" testing environment for comparison, we loosened the strict vocabulary requirement, which is difficult to fulfill with highly varied real data, and recreated the evaluation setup by restructuring existing GEC data. Note that, due to its softer control, this setting should only be taken as a supplementary comparison for additional insight.

In this study, we investigate standard five error types defined by Bryant et al. (2017), which are errors based on grammatical rules: subject-verb agreement errors (VERB:SVA), verb forms errors (VERB:FORM), word order errors (WO), morphological errors (MORPH), and noun number errors (NOUN:NUM). We created each version of the data as follows.

Synthetic data We provide a vocabulary-controlled dataset using CFG inspired by the data generation process in (Yanaka et al., 2020). More

Dataset		VERB:SVA	VERB:FORM	WO	MORPH	NOUN:NUM
Synthetic data	Known	99.61	99.17	99.09	98.44	97.47
	Unknown	46.05	56.93	84.00	29.35	65.55
	Δ	**-53.56**	**-42.24**	**-15.09**	**-69.09**	**-31.92**
Real data	Known	87.84	86.36	74.89	87.77	83.75
	Unknown	6.28	6.28	9.25	3.83	12.49
	Δ	**-81.56**	**-80.08**	**-65.64**	**-83.94**	**-71.26**

Table 2: Generalization performance for unseen errors. Each number represents an $F_{0.5}$ score.

specifically, we design two kinds of generation rules for each of the five error types to be analyzed, one generating grammatical sentences and the other ungrammatical ones[1]. For example, for VERB:SVA, the rule $S \rightarrow NP_{pl}\ VP_{sg}$ can generate ungrammatical sentences containing "**dogs smiles*", and $S \rightarrow NP_{sg}\ VP_{sg}$ can generate grammatical sentences containing "*dog smiles*". To produce natural sentences, we selected 15 lexical items for nouns, intransitive verbs, transitive verbs, adjectives, and adverbs, respectively. We can adjust the data size by changing the number of sentences generated by the CFG. In this paper, we automatically constructed 50,000 sentence pairs for each error type.

Real data To provide real data, we first perform an automatic annotation of error type labels and error correction patterns on an existing learner dataset using ERRANT (Bryant et al., 2017) [2]. Here, we used approximately 2 million sentence pairs as the learner dataset, which is a combination of training and development data distributed by the BEA-2019 Shared Task[3]. Then, we split the data while preserving error types and error correction patterns so that there is one error correction pattern per sentence. The unknown setting can be constructed by sorting the entire dataset based on the retained error correction patterns and classifying those with duplicates into training data and those without duplicates into test data. We constructed the known setting by sampling a small amount of data from training data as test data such that the same error correction patterns are included in both training and test sets. Using the above procedure, we obtained 25,889 sentence pairs for VERB:SVA, 41,592 sentence pairs for VERB:FORM, 18,779 sentence pairs for WO, 26,345 sentence pairs for MORPH, and 68,002 sentence pairs for NOUN:NUM. Compared to the synthetic data, real data has a wide variety of vocabulary and syntax ranging from simple to complex.

4 Experiments

4.1 Experimental Settings

We evaluated the grammatical generalization capability of a vanilla Transformer-based EncDec model. Specifically, we used the `fairseq` toolkit (Ott et al., 2019) implementation of the "Transformer (big)" setting (Vaswani et al., 2017)[4], and used the $F_{0.5}$ score calculated by ERRANT as the evaluation metric. We do not evaluate current state-of-the-art (SOTA) systems for the following two reasons. First, the top system in BEA2019 (Grundkiewicz et al., 2019) and the current SOTA systems (Omelianchuk et al., 2020; Kaneko et al., 2020) use pre-trained models such as pre-trained Masked LMs or use pseudo-data during pre-training. A key point of our study is controlling for seen/unseen patterns. This becomes difficult with pre-trained models since we cannot know whether a particular pattern is seen during pre-training. Second, we believe that evaluating a standard model's architecture, which is commonly used at the core of rapidly evolving SOTA systems, allows for a more accurate analysis by eliminating factors that make analysis more complex, and a more general analysis since our findings can be transferred to most current models, including SOTA systems.

4.2 Results

Table 2 shows the evaluation results. The evaluation using the synthetic data shows that the model's correction performance drops significantly in the unknown setting compared to the known setting, except for WO. One reason for the relatively high gen-

---

[1] Appendix A provides some CFG rules and lexical entries.

[2] `https://github.com/chrisjbryant/errant`

[3] `https://www.cl.cam.ac.uk/research/nl/bea2019st/`

[4] See Appendix B and C for details of the datasets and hyperparameters we used, respectively.

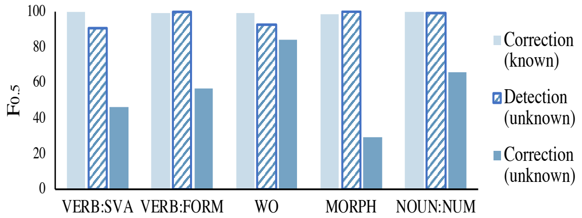

Figure 2: Comparison of detection and correction performance.

eralization ability of WO for unseen errors could be its relative simplicity. Namely, WO can be corrected just by identifying the word's position (Table 1). In contrast, other errors need to be corrected while recognizing differences in the surface form of words and dependencies between specific words, which increases the complexity of the correction task.

On the other hand, evaluation using real data show a significant performance drop on all errors, including WO, in the unknown setting, suggesting that generalization is more difficult in more practical settings where the vocabulary and syntax are diverse.

5 Analysis

Detection vs. Correction To analyze whether the model failed to generalize due to an inability to *detect* errors or an inability to *predict* the correct word, we compare the error detection and correction performance in the unknown setting. The detection performance is measured by evaluating whether the GEC model makes any edit in the error location. We evaluated both the detection and the correction performance using ERRANT. Figure 2 shows the evaluation result using synthetic data. The result shows the model successfully detected all error types, suggesting that the model can generalize its grammatical knowledge at least enough to detect errors, but not enough to predict the correct word.

We can also consider the generalization performance reported in Table 2 as a kind of ablation study: distinguishing, for each error type, how much the language modeling information and the error correction pattern information contribute to improving its correction performance, respectively. We can assume a model can learn accurate language model information in the unknown setting, but not the error correction patterns. Therefore, we can see that WO, which has a lower drop in correction performance in the unknown setting compared to the others, can be corrected with language modeling information alone. This result is consistent with the report (Futrell and Levy, 2019) that language models are robust to word order.

	noiseless	noisy
VERB:SVA	9.95	5.78
VERB:FORM	12.33	5.47
WO	7.89	9.35
MORPH	6.32	3.90
NOUN:NUM	24.16	12.49

Table 3: Effect of the complexity of errors in a sentence. Each number represents an $F_{0.5}$ score.

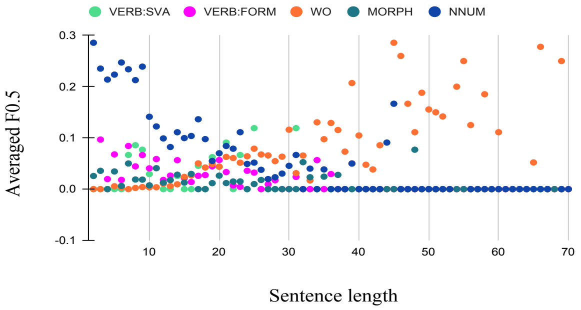

Figure 3: Relationship between sentence length and performance.

Complexity in real data To better understand the relationship between complexity and performance, we observed the effect of two contributing factors: error complexity and sentence length. Specifically, we compared the performance when the target error is the only error in the sentence (*noiseless*), and when the sentence contains other errors besides the target error (*noisy*). Table 3 shows the effect of the complexity of errors in a sentence. The results show that the performance of WO is constant with and without noise, while the other errors are affected by noise. Also, we analyzed the relationship between sentence length and performance (Figure 3) and confirmed that the difficulty of corrections on WO does not depend on the sentence length. These results suggest that the reason why the drop in correction performance of WO was relatively low compared to the others, even with real data, is due to its robustness to the complexity of input sentences.

Can a few error correction patterns improve model performance? We have found that the current model is vulnerable to unseen errors, but

#seen patterns	0	1	2
Precision	43.31	47.16	57.65
Recall	47.92	52.52	63.70
$F_{0.5}$	44.16	48.14	58.77

Table 4: Performance change when we expose the model to a few error correction patterns.

how does its performance change if we expose the model to a few error correction patterns? Table 4 shows the performance change when a few error corretion patterns are added to the training data for the pattern "**smile/smiles*" in VERB:SVA. As test data, we used the test data used in Section 4, excluding sentence pairs other than the target pattern. From the results, we can see that adding even just one or two samples to the training data can significantly improve the model's performance. This suggests that when preparing training data for GEC, it is important to sample even one or two seen patterns for each word to improve the performance.

6 Conclusion

This study explored to what extent GEC models generalize grammatical knowledge required for correcting errors. We introduce an analysis method using synthetic and real GEC datasets with controlled vocabularies to evaluate whether models can generalize to unseen errors. We found that the current standard Transformer-based GEC model can generalize error detection to some extent in a simple synthetic setting, while it cannot generalize correction to a greater extent in both synthetic and real settings, suggesting that it lacks the generalization ability required to correct errors from provided training examples. Therefore, methods to incorporate grammatical knowledge as rules into the current models can be expected to be necessary to implement a lightweight GEC model requiring less training data, which we plan to investigate in our future work.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments and suggestions. We are also grateful to Kentaro Inui and Ana Brassard for their insightful comments and suggestions. This work was partially supported by JSPS KAKENHI Grant Number JP20K19868.

References

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL Errors Using Phrasal SMT Techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL-COLING 2006)*, pages 249–256.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 793–805.

Richard Futrell and Roger P. Levy. 2019. Do RNNs learn human-like abstract word order preferences? In *Proceedings of the Society for Computation in Linguistics (SCiL 2019)*, pages 50–59.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 284–290.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 252–263.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless Green Recurrent Networks Dream Hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, pages 1195–1205.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 595–606.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 4248–4254.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics (TACL 2016)*, 4:521–535.

Rebecca Marvin and Tal Linzen. 2018. Targeted Syntactic Evaluation of Language Models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 1192–1202.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 31 (NIPS 2017)*, pages 5998–6008.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. 2020. Do neural models learn systematicity of monotonicity inference in natural language? In *Proc. of (ACL2020)*, pages 6105–6117.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

A CFG rules used to construct synthetic data

		Generation rules
S	→	NP VP
S$_{\text{-SVA}}$	→	NP$_{sg}$ VP$_{pl}$ | NP$_{pl}$ VP$_{sg}$
VP	→	IV | IV ADV | TV NP
VP$_{\text{-FORM}}$	→	IV$_{ing}$ | IV$_{ing}$ ADV | TV$_{ing}$ NP
VP$_{\text{-MORPH}}$	→	IV ADJ
NP	→	Q N | Q ADJ N
NP$_{\text{-WO}}$	→	ADJ Q N
NP$_{\text{-NUM}}$	→	Q$_{sg}$ N$_{pl}$ | Q$_{pl}$ N$_{sg}$ | Q$_{sg}$ ADJ N$_{pl}$ | Q$_{pl}$ ADJ N$_{sg}$
		Lexical items
Q	→	{*a, every, no, some, many*}
N	→	{*dog, rabbit, cat, bear, tiger*}
IV	→	{*run, walk, come, dance, leave*}
TV	→	{*kicked, hit, cleaned, touched, accepted*}
ADJ	→	{*white, gray, big, small, large, old*}
ADV	→	{*quickly, slowly, gracefully, seriously, happily*}

Table 5: Examples of CFG rules used for synthetic data construction. The generation rules with errors for each error type are shown by VP$_{\text{-error type}}$ for instance.

B Details of the datasets used in the experiments

	VERB:SVA	VERB:FORM	WO	MORPH	NOUN:NUM
Known	50,000 / 2,000 / 18,562	50,000 / 2,000 / 10,125	50,000 / 2,000 / 8,438	50,000 / 2,000 / 10,125	50,000 / 2,000 / 8,438
Unknown	50,000 / 2,000 / 13,749	50,000 / 2,000 / 7,500	50,000 / 2,000 / 6,250	50,000 / 2,000 / 7,500	50,000 / 2,000 / 6,250

Table 6: Details of the data split in the synthetic data setting (training/development/test).

	VERB:SVA	VERB:FORM	WO	MORPH	NOUN:NUM
Known	23,889 / 2,000 / 2,000	39,592 / 2,000 / 2,000	16,779 / 2,000 / 2,000	24,345 / 2,000 / 2,000	66,002 / 2,000 / 2,000
Unknown	23,889 / 2,000 / 633	34905 / 2,000 / 2000	16,779 / 2,000 / 9,199	24,345 / 2,000 / 5227	66,002 / 2,000 / 3,111

Table 7: Details of the data split in the real data setting (training/development/test).

C Hyper-parameter settings

Configurations	Values
Model Architecture	Transformer (Vaswani et al., 2017)
Optimizer	Adam (Kingma and Ba, 2015)
Learning Rate Schedule	Same as described in Section 5.3 of (Vaswani et al., 2017)
Number of Epochs	30 for synthetic data and 150 for real data
Dropout	0.3
Stopping Criterion	Train model for 30 epochs (synthetic data) and 150 epochs (real data).
Gradient Clipping	1.0
Loss Function	Label smoothed cross entropy (Szegedy et al., 2016)
Beam Search	Beam size 5 with length normalization

Table 8: Detailed hyper-parameters used for the base GEC model.

Domain-Aware Dependency Parsing for Questions

Aparna Garimella[*]
Adobe Research
garimell@adobe.com

Laura Chiticariu
IBM Research
chiti@us.ibm.com

Yunyao Li
IBM Research
yunyaoli@us.ibm.com

Abstract

Parsing natural language questions in specific domains is crucial to a wide range of applications from question-answering to dialog systems. Pre-trained parsers are usually trained on corpora dominated by non-questions, and thus perform poorly on domain-specific questions. Retraining parsers with domain-specific questions labeled with syntactic parse trees is expensive, as these annotations require linguistic expertise. In this paper, we propose an automatic labeled domain question generation framework by leveraging domain knowledge and seed domain questions. We evaluate our approach in two domains, and release the generated question datasets. Our experimental results demonstrate that auto-generated labeled questions indeed lead to significant ($4.9\% - 9\%$) increase in the accuracy of state-of-the-art (SoTA) parsers on domain questions.

1 Introduction

Understanding questions is the first step towards building accurate and reliable natural language interfaces. Recent works on Google Assistant, or IBM Watson focus on building domain-specific conversational agents. In this paper, we focus on syntactic parsing of domain-specific questions, which is crucial in domain-specific agents. The accuracy of syntactic parsers is known to depend on the syntactic similarities between the training data and application text. However, questions are often underrepresented in classic training corpora. In Penn TreeBank, only 0.5% of sentences from the Wall Street Journal are questions, with majority being rhetorical in nature, and those occurring in conversations (starting with interrogatives *wh-/how*, imperatives *show me*, *name*, or *yes/no* questions) are heavily underrepresented. Recognizing this problem, Judge et al. (2006) introduced QuestionBank, a labeled corpus of 4,000 general questions.

However, domain-specific questions are often underrepresented in general purpose question corpora, leading to their poor parsing performances; e.g., in *Show me Neil's insider transactions since 2011*, a SoTA parser (Nivre et al., 2016a) trained on Universal Dependencies (UD) English TreeBank (Silveira et al., 2014) and QuestionBank attaches *since* to *show*, instead of *transactions*, causing the system to misinterpret Neil's insider transactions since 2011, to all his transactions. In *Will it rain tomorrow by noon?*, *tomorrow* is attached to *rain* with a wrong dependency relation (`dobj` instead of `nmod:tmod`), causing the system to miss the temporal aspect of the question.

A natural solution to obtain accurate domain-specific parsers is to train them on domain-specific corpora. However, obtaining domain-specific questions is difficult. Moreover, annotating questions for parse trees is tedious, prone to errors and inconsistencies, and requires linguistic expertise. Petrov et al. (2010) proposed *uptraining*, training a parser on the output of a slower, more accurate parser. For acceptable performance, the unlabeled corpus must be large (100,000 questions). Our method is applicable when such a large corpus is not available. Inspired by (Wang et al., 2015) who showed that semantic parsers can be built "overnight" using domain expertise, we seek to reduce the effort required to handle a new domain using *domain knowledge*: (1) a *domain schema* modeling the concepts and relationships in a domain, and (2) a *knowledge base* of data instances that populate the schema (Hamon et al., 2017; Julien Gobeill and Ruch, 2015; Damljanovic et al., 2010). To the best of our knowledge, this is the first work to use domain knowledge to improve syntactic parsing.

This paper makes two main contributions. (1) We propose a framework to automatically generate

---

[*] This work was done when the author was at IBM Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4562–4568
August 1–6, 2021.

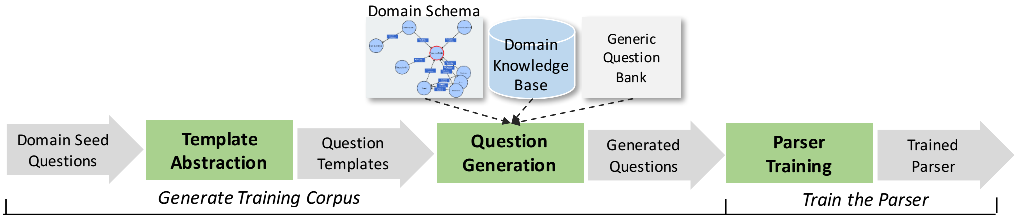

Figure 1: Overview of our approach.

a large training corpus of domain-specific questions for syntactic parsing in English, from a few seed domain questions using domain knowledge. We evaluate our approach in general (Weather) and specialized (Finance) domains. The parsers trained on the data augmented with our method have significant improvements ($4.9\% - 9\%$ LAS, $4.8\% - 8.8\%$ UAS) over those trained on UD Treebank and QuestionBank. Our method is robust to small seed, improving accuracy with as few as 10 seed questions. (2) We release the datasets and generated questions to the community.[1]

2 Question Generation

We automatically generate large training corpora by combining (1) domain seed questions labeled with syntactic dependencies using the Universal Dependencies v1.4 (Nivre et al., 2016b) guideline, and (2) domain knowledge. We use a two-stage pipeline: For each seed question, a *question template* is created that maintains its general structure, but abstracts away details of specific entities. Next, new questions are generated by automatically filling the templates with new entities obtained using domain schema and knowledge base. The domain schema is further annotated to ensure naturalness of generated questions (Figure 1).

2.1 Template Abstraction

Given labeled seed, this stage involves abstracting *templates* from their parse trees. We focus on two entity types, based on the dependency relation of nodes to their parent in parse trees of questions:

Subject Entity (`qsubj`) is the subject in a question. In *wh*-interrogatives, it occurs as the nominal subject (`nsubj`) of root. (QENT$_1$ in Fig. 2a). In imperatives, it occurs as the direct object (`dobj`) of the imperative verb. (QENT$_1$ in Fig. 2b).

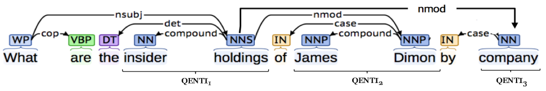

(a) Finance question Q_1.

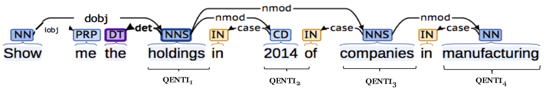

(b) Finance question Q_2.

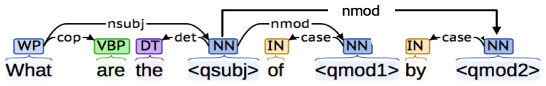

(c) Template $\mathcal{T}_{Q_1}$.

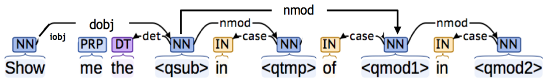

(d) Template $\mathcal{T}_{Q_2}$.

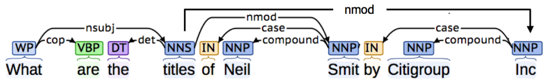

(e) Generated question Q^N_1.

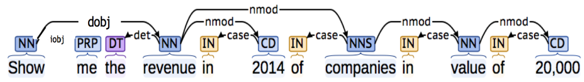

(f) Generated question Q^N_2.

Figure 2: Finance questions (a,b), their templates (c,d) and examples of generated questions (e,f).

Modifier Entity (`qmod`) is a (noun phrase) node that modifies another node. In parse trees, they usually relate via an `nmod` dependency.[2] In Fig. 2a, 'James Dimon' and 'company' are `qmod` entities of 'insider holdings'. Given a question, its *template* has `qsubj` and `qmod` entities replaced by placeholders. Algorithm 1 details template abstraction.

Domain Schema and Knowledge Base We assume domain schema as a set of *classes* with *properties*, *relations* between classes, and knowledge base conforming to the schema with few data instances; e.g., `title` is a property of class `AssignmentHistory` with instances 'CEO' and

---

[1] `https://github.com/System-T`

[2] We also account for other dependencies (`nmod:poss`, `appos`, `xcomp`) modifying `qsubj`.

'COO' in the Finance domain. Figure 5 in Section B shows an example schema for financial domain and a few data instances in a knowledge base $\mathcal{KB}$ conforming to this schema.

2.2 Template Filling

Given a template, questions are generated by systematically filling `qsubj` and `qmod` with new values obtained from domain schema and knowledge base. The main challenges include: (1) which new values are suitable for filling? and (2) how to automatically construct parse trees for generated questions? To address the first challenge, we propose ***template replacement heuristics***: `qsubj` is replaced with properties of a class, while `qmod` is replaced with properties of other classes in relation to that class. Further, we introduce one-level nesting on the filled `qmod` by expanding them using the relative pronoun *whose* (further details in Section C).

We address the second challenge by constructing ***parse trees for template fillers*** as follows. (1) For each class property, we manually provide the parse tree. This incurs a small one-time effort proportional to the size of the domain schema, which is small compared to the knowledge base. (2) For each instance of a class property, we automatically generate the parse tree by making the last word as root, and attach the preceding words (Fig. 3a). (3) For `qmod`, we construct relative pronoun expansion node by attaching to it the parse tree of a property of a related class with tag `acl:relcl` and *whose*. Fig. 3b expands `qmod`$_1$ = `Person:name='Neil Smit'` using the parse tree of `Holding:value` with a prepositional attachment for a value (*i.e.,* 20,000). The preposition is changed to an appropriate copular verb (*is*, *are*).

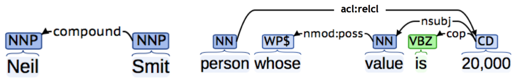

(a) (b)

Figure 3: Parse trees for template fillers based on heuristics for data instances (a), and and relative pronoun expansion (b).

Tab. 1 shows example questions generated in the Finance domain (Q_1^N to Q_4^N).

While the generated questions exhibit variety, most are unnatural; more natural formulations are Q_1^F to Q_4^F. The common sources of unnaturalness are the following.

Incorrect preposition for `qmod` In Q_1^N, the original preposition *by* is incorrect for the choice of `qsubj` and `qmod`$_2$; it should instead be *in*.

Incomplete usage of dependent property In Q_2^N, the choice of property for `qsubj` is incomplete; start date is not independent by itself and should be associated with a title the person holds.

Incomplete semantics of property In Q_3^N, the usage of property for `qsubj` is misleading as it refers to the value of a company, while the intent is to query the value of holdings in the company.

Incorrect question word In Q_4^N, *what* should be replaced with *who*, since `qsubj` is filled with a person, as opposed to an object.

2.3 Schema Annotations

A random sample of 100 generated questions contains 68 incorrect prepositions, 47 incomplete dependents, 64 incomplete semantics, and 2 incorrect question words. We address them using simple annotations to the domain schema, provided by a domain expert, in a one-time effort that is linear with the size of the schema.

Class Relations. between classes are annotated with *connective words* (usually prepositions), *e.g.*, `AssignmentHistory` $\xrightarrow{\text{in}}$ `Company` and `AssignmentHistory` $\xrightarrow{\text{of}}$ `Person` in Figure 5. This annotation addresses the incorrect prepositions for `qmod` (e.g. Q_1^F).

Heading Properties. are those that can be queried independently without referencing others. Each property is annotated as heading or non-heading for all the classes in the schema. While heading properties and their instances can be used to fill `qsubj` and `qmod` independently, we devise rules to use non-heading ones (Appendix D). This annotation modifies the use of non-heading property *start date* in Q_2^N by associating it with an instance 'CEO' of heading property *title* in Q_2^F.

Class-dependency of Properties. This addresses the incomplete semantics of properties. Certain properties are ambiguous, and querying them requires specifying their class names to add context; *e.g.*, one would ask *What are the values of holdings in Citigroup?* as oppose to *What are the values in Citigroup?* Properties in the schema are annotated as class-dependent or not. This annotation leads to Q_3^F, a more natural version of Q_3^N.

Possible Question Words. To address the incorrect question words (Q_4^N), we annotate all properties and their instances with corresponding possible *wh-* question words (Q_4^F).

Naively generated question	**Final question**
Q_1^N: What are the titles of Neil Smit *by* Citigroup Inc?	Q_1^F: What are the titles of Neil Smit *in* Citigroup Inc?
Q_2^N: What are the *start dates* of Neil Smit by Citigroup Inc?	Q_2^F: What are the *start dates as CEO* of Neil Smit *in* Citigroup Inc?
Q_3^N:What are the *values* of Citigroup Inc by Neil Smit?	Q_3^F: What are the values *of holdings in* Citigroup Inc *of* Neil Smit?
Q_4^N: *What* are the CEOs of Citigroup Inc?	Q_4^F: *Who* are the CEOs *in* Citigroup Inc?

Table 1: Questions generated from "What are the insider holdings of James Dimon by company" with template: "What are the `qsubj` of `qmod₁` by `qmod₂`?"

DATASET		DOMAIN	# SENT.
UD	UD Treebank	General	16,622
GQ	QuestionBank	General	4,000
DQ_F	Domain-specific questions	Finance	250
DQ_W	labeled by linguist	Weather	250

Table 2: Datasets in our experiments.

	FINANCE		WEATHER	
DATASET	LAS	UAS	LAS	UAS
UD	78.44	80.71	74.26	78.09
UD+GQ	80.62	82.7	79.48	82.44
UD+DQ	85.06	86.96	79.06	82.26
UD+N_{DQ}	86.34	88.61	79.47	82.89
UD+GQ+DQ	86.05	87.88	81.97	84.42
UD+GQ+N_{DQ}	88.31*	90.41†	84.37*	87.21†
UD+GQ+DQ+N_{DQ}	**89.67***	**91.53**†	**84.82***	**87.49**†

Table 3: Performance of Malt with different training sets. LAS with * (UAS with †) are statistically significant compared to all other settings using McNemar's test ($p < 0.05$).

	FINANCE		WEATHER	
DATASET	LAS	UAS	LAS	UAS
UD	72.31	80.19	69.87	76.27
UD+GQ	76.41	84.50	72.76	78.44
UD+DQ	80.47	86.10	77.30	82.24
UD+N_{DQ}	80.56	86.34	77.98	83.07
UD+GQ+DQ	81.99*	87.84†	80.44*	84.80†
UD+GQ+N_{DQ}	**82.43***	**88.54**†	**81.03***	**85.34**†

Table 4: SyntaxNet results with different training sets.

3 Experiments and Results

We evaluate our approach with 5-fold cross-validation in the Finance (9 classes, 16 relations, 63 props., 3,028 instances) and Weather (9 classes, 1 relation, 85 props., 66 instances) domains.

Data. Table 2 summarizes different datasets used in our experiments. For each domain, we randomly sample 250 questions from the query logs of an internal QA system, and manually annotate them with dependency parse information (DQ_F and DQ_W). We group the questions based on their template structure, resulting in 148 and 166 unique templates for Finance and Weather, respectively. Questions corresponding to 20% templates are used for training in each fold, resulting on average in 38 and 54 seed questions from 30 and 34 templates for Finance and Weather, respectively. The remaining questions (212 and 196 questions for Finance and Weather) are used as test sets to evaluate the trained parsers. In this way, the train and test sets do not have any overlap even at the template-level.

From DQ, we generate new questions N_{DQ} using our approach. As the number of questions generated from each seed question is very large (140,325 for Finance), and they have similar syntactic structures, we include in N_{DQ} a maximum of 50 questions randomly selected from those generated from each seed question (this value is chosen from validation experiments).

Models. We use two parsers with default parameters: Malt (Nivre et al., 2016a), as it is fast to train, and SyntaxNet, a SoAT neural model (Andor et al., 2016). We measure parser performance using UAS (Eisner, 1996) and LAS (Nivre et al., 2004) metrics, on the test portions of DQ_F and DQ_W.

Effectiveness. Tables 3 and 4 shows the results of Malt trained on different combinations of datasets. SyntaxNet displays similar trend, though with lower scores. Adding GQ to UD improves performance in both domains. Adding DQ leads to greater improvements for Finance, though GQ contains many more questions than DQ. This confirms the importance of domain-specific questions in the training data for specialized domains such as Finance. Adding generated questions N_{DQ} leads to comparable, if not better, performances over UD+GQ and UD+GQ+DQ for both domains, with small sizes (1,150 questions for Finance, 1,136 for Weather, on average), illustrating the effectiveness of our algorithm in generating labeled domain questions. UD+GQ+N_{DQ} and UD+GQ+DQ+N_{DQ} achieve the best performance: LAS gain of $4.9\% - 9\%$ over UD+GQ, and

DATASET	FINANCE	WEATHER
UD	64.54	64.56
UD+DQ	65.14	64.75
UD+DQ+N$_{\text{DQ}}$	66.00	65.26

Table 5: LAS of Malt parser on GQ as test.

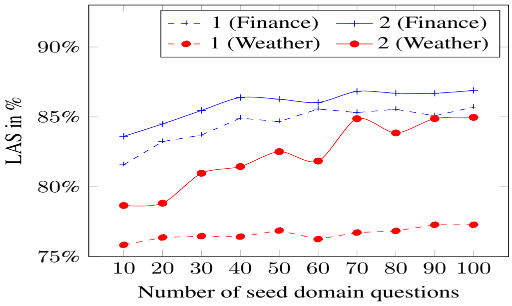

Figure 4: Malt performance trained with increasing sizes of DQ (1: UD+DQ, 2: UD+DQ+N$_{DQ}$).

$2.3\% - 3.6\%$ over UD+GQ+DQ. This shows that our framework generates effective labeled domain-specific questions which help improve parser performances when used for training them.

Table 5 shows LAS of Malt parser trained on domain-specific seed and generated questions, and evaluated on GQ. The improvements with the addition of DQ and N$_{\text{DQ}}$ on general questions illustrate that our framework does not overfit to a specific domain; instead, the augmented training sets only facilitate an increase in the performances.

Effect of Schema Annotations. Schema annotation is a one-time effort and is proportional to the schema size; it required an average of 2 hours for the authors to annotate each schema. The domain understanding required for this can be acquired in a fairly small amount of time as the annotations are straight-forward, as opposed to the heavy linguistic expertise required to annotate questions with parse trees, which generally requires weeks or even months of effort to obtain a decently large training data. Moreover, large-scale human dependency tree annotations are error-prone, inconsistent and intensive, as annotators may tend to forget the many linguistic rules involved, and need to constantly ensure that the same rules are applied everywhere. Another sample of 100 questions, generated using the schema annotations, do not exhibit the anomalies listed in Section 2.2. Moreover, the inclusion of schema annotations leads to $0.4\% - 0.94\%$ gains in LAS, thus compensating the effort required to annotate the ontology.

Robustness to Size of Domain Seed. To study the performance variation with the size of training data, we randomly sample $1/3^{rd}$ templates as test (49 Finance and 55 Weather questions), and train the parser on questions from remaining templates with varying sizes. Figure 4 shows LAS averaged over 5 runs with Malt in two settings: UD+DQ and UD+DQ+N$_{\text{DQ}}$. Our framework leads to significantly better performances in both domains, and even with only 10 seed questions, LAS improves by 2.03% (Finance) and 2.82% (Weather). We also note that in the Finance, adding 10 domain questions to UD leads to comparable performance to adding entire QuestionBank, and 30 questions are needed for achieve the same in Weather (UD+GQ in Table 3). As Finance is a specialized domain, N$_{\text{DQ}}$ from even a small seed set have a higher effect compared to a more general domain like Weather.

4 Conclusions

We proposed a method to automatically generate labeled domain-specific questions from small seed set using domain knowledge, to compensate for the lack of training data. We introduced ontology annotations that enhance the naturalness of the automatically generated questions. Our approach resulted in a significant increase in the LAS of $2.3\% - 2.4\%$ over training with standard corpora and domain seed in two domains, and is robust to the seed set size. With sufficient labeled data, some of these heuristics could potentially be learned from the data. We believe our work paves way to develop domain-independent question parsing methods with very little or possibly no training data.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2016, pages 2442–2452, Berlin, Germany.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'10, pages 106–120, Heraklion, Crete, Greece.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational* *linguistics-Volume 1*, COLING 1996, pages 340–345, Copenhagen, Denmark.

Thierry Hamon, Natalia Grabar, and Fleur Mougin. 2017. Querying biomedical linked data with natural language questions. *Semantic Web*, 8(4):581–599.

John Judge, Aoife Cahill, and Josef Van Genabith. 2006. QuestionBank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL 2006, pages 497–504, Sydney.

Emilie Pasche Dina Vishnyakova Pascale Gaudet Amos Bairoch Julien Gobeill, Arnaud Gaudinat and Patrick Ruch. 2015. Deep question answering for protein annotation. *Database: The Journal of Biological Databases and Curation*, bav081.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, New York City, USA.

Joakim Nivre, Johan Hall, and Jens and Nilsson. 2016a. Malt Parser, v1.9, released June 4 2016. `http://www.maltparser.org/download.html`.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016b. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, LREC 2016, pages 1659–1666, Portorož, Slovenia.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2010, pages 705–713, Cambridge, MA, USA.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC 2014, pages 2897–2904, Reykjavik, Iceland.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, ACL IJCNLP 2015, pages 1332–1342, Beijing, China.

A Template Abstraction

We refer to `qmod` entities that have temporal values as `qtmp`.

Algorithm 1 Template Abstraction

```
Input: Question $Q$ and its syntactic parse tree $P_Q$.
Output: Template $\mathcal{T}_Q$ for $Q$.
procedure AbstractTemplate($Q$, $P_Q$)
    Identify the qsubj entity in $P_Q$.
    Identify all qmod entities attached to qsubj's head in $P_Q$.
    Recursively identify qmod entities for each of qmod.
    Among all qmod entities, mark those that are qtmp.
    Let $\mathcal{T}_Q$ be $P_Q$ with qsubj and qmod replaced with placeholders.
```

B Domain Schema

Fig. 5 shows an example schema for financial domain and a few data instances in a knowledge base $\mathcal{KB}$ conforming to this schema. The schema models a subset of the Finance domain, including people and companies with some of their related classes $\mathcal{R}$ such as address and financial metrics. It also models several relations of interest including a person's holding in a company, and a person's job assignment as an officer of a company. Ellipses represent *classes*; rectangles show class *properties* and their data types; For example, `title` is a property of class `AssignmentHistory` with instances 'CEO' and 'COO' in Finance domain.

We denote properties as `class:property` and their instances as `class:property='instance'`. For classes with a property `name`, we consider instances of `name` as instances of the class (e.g., 'Citigroup Inc' is an instance of class `Company`). Labeled arrows denote relations between classes. For example, in Fig. 5, `AssignmentHistory` is related to `Company` and `Person`, modeling the relationships between a person's assignment within a company; e.g., Neil Smit is the CEO of Citigroup.

C Template Filling

We detail the template replacement heuristics for each entity type here. If sufficient labeled data is available, some of these heuristics could potentially be learned from the data.

***Entity* `qsubj`** can be filled with either (1) a class property,[3] or (2) an instance of a class property of type string, provided that the property is not `name` and the instance is not a

---

[3] If the part-of-speech tags of the property and `qsubj` do not match, appropriate changes are made to the template.

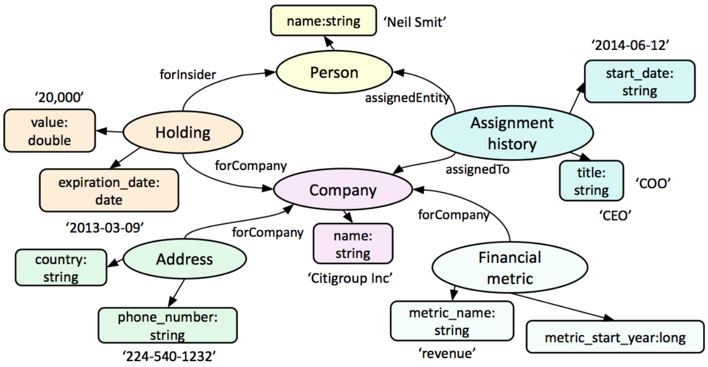

Figure 5: Domain schema for the finance domain, and example data instances. Classes, properties (their data types), and relations between classes are shown as ovals, rectangles and labeled arrows. Data instances are shown next to corresponding properties in quotes.

proper noun. For example, `qsubj` in $\mathcal{T}_{Q_1}$ can be filled with `AssignmentHistory:title` or `AssignmentHistory:'CEO'` resulting in *What is the title of ..?* and *Who is the CEO of ..?* The proper noun restriction avoids generating meaningless questions such as *What is the 224-540-1232 of ..?* The restriction on `name` and string type avoids questions such as *What is the Citigroup Inc ..?*, and *What is the 20,000 of ..?*

***Entity* `qmod`** is filled based on the QENT it modifies. If QENT is a class property, `qmod` is filled with an instance of a related class. In $\mathcal{T}_{Q_1}$, when `qsubj` is filled with `AssignmentHistory:title`, the `qmod`$_1$ slot can be filled with `Person:name='Neil Smit'`, resulting in *What are the titles of Neil Smit ..?* If QENT is a class instance, it is changed to the name of the class, and `qmod` is filled with a V_p of property p of a related class. For example, in $\mathcal{T}_{Q_2}$, if `qmod`$_1$ is filled with `Company:name='Citigroup Inc'`, it will be changed to 'company', and `qmod`$_2$ can be filled with V_p *value of 20,000* of `Holding:value` as `Holding` is related to `Company`.

Relative pronouns for* `qmod` *expansion. To generate more complex questions, we introduce one-level nesting on an already filled `qmod` by expanding it using relative pronoun *whose*. If `qmod` is not a V_p of any property p, we replace it with its corresponding class name, and attach a relative modifier clause using V_p of a property p of one of its related classes. In $\mathcal{T}_{Q_1}$, when `qmod`$_1$ is filled with `Person:name='Neil Smit'`, the relative pronoun expansion of `qmod`$_1$ is *person whose value is 20,000* using related class `Holding` of `Person`.

***Entity* `qtmp`** is always retained as temporal values do not change the syntactic context of questions. In Fig. 2f, see Q_2^N generated from $\mathcal{T}_{Q_2}$.

(a) `AssigH:start_date` (b) `Holding:value`

Figure 6: Templates for non-heading property (a) and class-dependency (b).

D Heading Property Heuristics

While heading properties and their instances can be used to fill QENT entities independently, we adopt the following rules for non-heading properties:

- For classes with a single heading property with an instance (e.g. `AssignmentHistory:title` with 'CEO'), the instance is used along with a non-heading property to query it using prepositional connectives which are also annotated along with heading properties (`AssignmentHistory:start_date` *as* CEO). The QENT replacement is automatically constructed by attaching the heading property's instance as `nmod` using prepositional connective to the parse tree of non-heading property (Fig. 6a).
- For classes with more than one heading property, only they can be queried (`Holding:value`, `Holding:expiration_date`). E.g., *what is the value of ..?*, *what is the expiration date of ..?*

Using Social and Linguistic Information to Adapt Pretrained Representations for Political Perspective Identification

Chang Li
Department of Computer Science
Purdue University, West Lafayette, IN
`li1873@purdue.edu`

Dan Goldwasser
Department of Computer Science
Purdue University, West Lafayette, IN
`dgoldwas@purdue.edu`

Abstract

Understanding the political perspective shaping the way events are discussed in the media is increasingly important due to the dramatic change in news distribution. With the advance in text classification models, the performance of political perspective detection is also improving rapidly. However, current deep learning based text models often require a large amount of supervised data for training, which can be very expensive to obtain for this task. Meanwhile, models pre-trained on the general source and task (e.g. BERT) lack the ability to focus on bias-related text span. In this paper, we propose a novel framework that pre-trains the text model using signals from the rich social and linguistic context that is readily available, including entity mentions, news sharing, and frame indicators. The pre-trained models benefit from tasks related to bias detection and therefore are easier to train with the bias labels. We demonstrate the effectiveness of our proposed framework by experiments on two news bias datasets. The models with pre-training achieve significant improvement in performance and are capable of identifying the text span for bias better.

1 Introduction

The perspectives underlying the way information is conveyed to readers can prime them to take similar stances and shape their worldview (Gentzkow and Shapiro, 2010, 2011). Given the highly polarized coverage of news events, recognizing these perspectives can help ensure that all points of view are represented by news aggregation services, and help avoid "information echo-chambers" in which only a single viewpoint is represented. It may also help to prevent the spread of false information online by showing people news with different perspectives.

Past work studying the expression of bias in the text has focused on lexical and syntactic representations of bias (Greene and Resnik, 2009; Recasens et al., 2013; Elfardy et al., 2015). Expressions of bias can include the use of the passive voice (e.g., *"mistakes were made"*), or references to known ideological talking points (Baumer et al., 2015; Budak et al., 2016; Card et al., 2016; Field et al., 2018; Morstatter et al., 2018) (e.g., *"pro-life"* vs. *"pro-choice"*). However, bias in news media is often nuanced and very difficult to detect. Journalists often strive to appear impartial and use language that does not reveal their opinions directly. Also, by their nature, news articles describing the same real-world event will share many similar details of the event, regardless of their political perspectives. Instead, bias is often expressed through informational choices (Fan et al., 2019), which highlight different aspects of the news story and frame facts shared by all articles in different ways. For example, the following articles capture different perspectives (Top *left*, Bottom *right*), while discussing the same news event– the 2021 storming of the U.S. Capitol [1].

Adapted from **NYTimes** (*Left*)

How Republicans Are Warping Reality Around the Capitol Attack ... Jim Hoft, did not reply to questions but did send along several of his own news articles related to claims of antifa involvement in the Capitol attack — citing the case of a man named **John Sullivan**, whom the right-wing media has dubbed an **"antifa leader"** in efforts to prove its theory of infiltration.

Adapted from **Fox News** (*Right*)

BLM activist inside Capitol claims he was 'documenting' riots, once said 'burn it all down'. John Sullivan has previously called for 'revolution' and to 'rip Trump' out of his office. An anti-Trump activist who once said he wanted to "rip" the president out of office entered the Capitol Building Wednesday alongside a mob of pro-Trump protesters, but he said he was just there to "document" it.

The two articles discuss the presentation of *John*

---

[1] `https://en.wikipedia.org/wiki/2021_storming_of_the_United_States_Capitol`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4569–4579
August 1–6, 2021.

Sullivan as an Antifa member[2] who participated in the Capitol storming. However the story is framed in very different ways - while the bottom article frames the story directly as a discussion of Antifa involvement, the top discusses it in the context of political messaging and journalism. Furthermore, we notice that the difference is focused on a specific entity - John Sullivan.

Despite the fact that these distinctions are easily detectable by a human reader familiar with the political divisions in the U.S., they are very difficult to detect automatically. Recent success stories using large-scale pre-training for constructing highly expressive language models (Devlin et al., 2019) are designed to capture co-occurrence patterns, likely to miss these subtle differences.

In this paper we suggest that bias detection requires a different set of self-supervised pre-training objectives that can help provide a better starting point for training downstream biased detection tasks. Specifically, we design three learning objectives. The first, captures *political knowledge*, focusing on the embedding of political entities discussed in the text. The second one captures *external social context*. Following the intuition that different social groups would engage with documents expressing a different bias (e.g., left-leaning users are more likely to read the NYTimes article compared to the Fox News article), we collect social information contextualizing news articles and learn to predict the social context of each article, based on its content, thus aligning the two representations. Finally, the third is based on linguistic knowledge, focusing on the *issue framing* decisions made by the authors. Framing decisions have been repeatedly shown to capture political bias (Recasens et al., 2013; Johnson and Goldwasser, 2016; Roy and Goldwasser, 2020; Mendelsohn et al., 2021), and we argue that infusing a language model with this information can help capture relevant information. Note that this information is only used for pre-training. Other works using social information to analyze political bias (Li and Goldwasser, 2019; Nguyen et al., 2020; Pacheco and Goldwasser, 2021) augment the text with social information, however since this information can be difficult to obtain in real-time, we decided to investigate if it can be used as a distant supervision source for pre-training a language model.

These pre-training tasks are then used for training a **M**ulti-head **A**ttention **N**etwork (MAN) which creates a bias-aware representation of the text.

We conducted our experiments over two datasets, Allsides (Li and Goldwasser, 2019) and SemEval Hyperpartisan news detection (Kiesel et al., 2019). We compared our approach to several competitive text classification models and conducted a careful ablation study designed to evaluate the individual contribution of pre-training through knowledge from various contexts. Our results demonstrate the importance of all aspects, each contributing to the model's performance.

2 Related Work

The problem of perspective identification is originally studied as a text classification task (Lin et al., 2006; Greene and Resnik, 2009; Iyyer et al., 2014), in which a classifier is trained to differentiate between specific perspectives. Other works use linguistic indicators of bias and expressions of implicit sentiment (Recasens et al., 2013; Baumer et al., 2015; Field et al., 2018).

Recent work by (Fan et al., 2019) aims to characterize content relevant for bias detection. Unlike their work which relies on annotated spans of text, we aim to characterize this content without explicit supervision.

In the recent SemEval-2019, a hyperpartisan news article detection task was suggested[3]. Many works attempt to solve this problem with deep learning models (Jiang et al., 2019; Hanawa et al., 2019). We build on these works to help shape our text representation approach.

Several recent works also started to make use of concepts or entities appearing in the text to get a better representation. Wang et al. (2017) treats the extracted concepts as pseudo words and appends them to the original word sequence which is then fed to a CNN. The KCNN model by Wang et al. (2018), used for news recommendation, concatenates entity embeddings with the respective word embeddings at each word position to enhance the input. We take a different approach and instead try to inject knowledge of entities into the text model through the masked entity training. Zhang et al. (2019) also uses entity-level masking for training. However, they predict the tokens for the masked

---

[2] `https://en.wikipedia.org/wiki/Antifa_(United_States)`

[3] `https://pan.webis.de/semeval19/semeval19-web/`

entity instead of relying on meaningful representations for entities as ours.

Political framing, due to its relation with ideology and perspective, is studied in the NLP communities (Johnson et al., 2017; Field et al., 2018; Shurafa et al., 2020). There is also growing interest in utilizing framing differences to identify bias in news articles (Roy and Goldwasser, 2020).

Pre-trained models are widely used in numerous NLP tasks, from the early word2vec representation (Mikolov et al., 2013) to the generic language models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). Recently, people also started to work on task specific pre-training that try to bring task and domain related knowledge into the model. Xu et al. (2019) is similar to our work as it proposes to enhance the BERT model through training on review data and sentiment classification tasks so that it can obtain better performance across multiple review-based tasks.

3 Political Perspective Identification Task

The problem of political perspective identification in news media can be formalised as follows. Given a news article d, where d consists of sentences s_i, $i \in [1, L]$, and each sentence s_i consists of words w_{it}, $t \in [1, T]$. L and T are the number of sentences in d and number of words in s_i respectively. The goal of this task is to predict the political perspective y of the document. Given different datasets, this can either be a binary classification task, where $y \in \{0, 1\}$ (hyperpartisan or not), or a multi-class classification problem, where $y \in \{0, 1, 2\}$ (left, center, right).

The overall architecture of our model is shown in Figure 1. It includes two sequence encoders, one for word level and another for sentence level. The hidden states from an encoder are combined through a multi-head self-attention mechanism. With pre-training on various social and linguistic information, the generated sentence and document vectors will consider not only the context within the text but also the knowledge about the entities (e.g. their political affiliation, or stance on controversial issues), sharing users, and frame indicators. We explain the structure of our model and the rich social and linguistic context we consider in detail below. Note that our pre-training strategies are not tied with any specific model structure and can be easily applied to other text models.

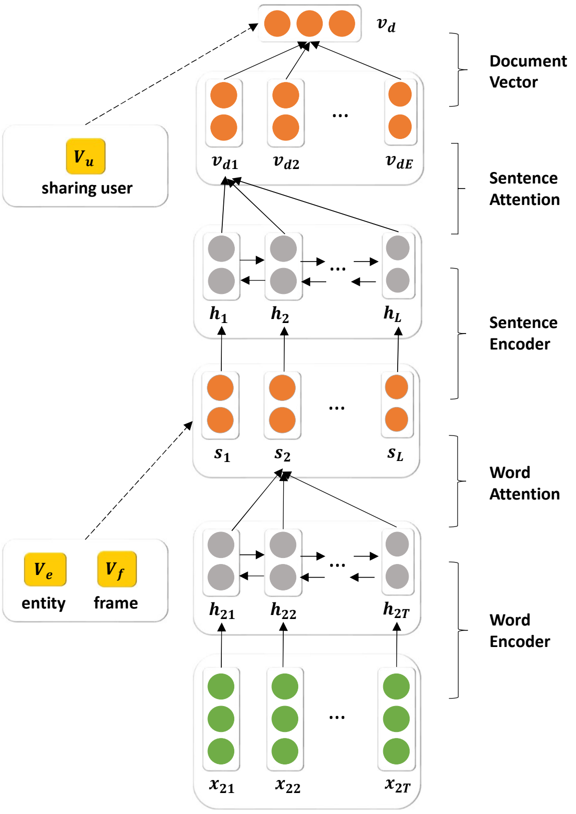

Figure 1: Overall Architecture of MAN Model.

3.1 Multi-Head Attention Network

The basic component of our model is the Hierarchical LSTM model (Yang et al., 2016). The goal of our model is to learn document representation v_d for political perspective prediction. It consists of several parts: a word sequence encoder, a word-level attention layer, a sentence sequence encoder, and a sentence-level attention layer. To capture the context in both directions, we use bidirectional LSTM in this work. For each element in the input sequence, the hidden state h is a concatenation of the forward hidden state $\overrightarrow{h}$ and backward hidden state $\overleftarrow{h}$ computed by the respective LSTM cells.

Given a sentence with words w_{it}, $t \in [1, T]$, each word is first converted to its embedding vector x_{it}. We can adopt pre-trained Glove (Pennington et al., 2014) word embeddings or deep contextualized word representation ELMo (Gardner et al., 2017) for this step. The word vectors are then fed into a word level bidirectional LSTM network to incorporate contextual information within the sentence. The hidden states h_{it} from the bidirectional LSTM network are passed to the next layer. In (Yang et al., 2016), a self attention mechanism is introduced to identify words that are important to the meaning of the sentence, and therefore higher weights are given to them when forming the aggregated sentence vector.

$$p_{itw} = tanh(W_w h_{it} + b_w) \tag{1}$$

$$\alpha_{itw} = \frac{exp(p_{itw}^T p_w)}{\sum_t exp(p_{itw}^T p_w)} \tag{2}$$

$$s_{iw} = \sum_t \alpha_{itw} h_{it} \tag{3}$$

p_{itw} encode the importance of a specific word according to its context, which is compared with the word level preference vector p_w to compute a similarity score. The scores are then normalized to get the attention weight α_{itw} through a softmax function. A weighted sum of the word hidden states is computed based on the attention weight as the sentence vector s_{iw}.

Inspired by the multi-head attention scheme in (Vaswani et al., 2017), we propose a multi-head attention in our model to extend its ability to jointly attend to information at different positions. The sentence vector s_i is computed as an average of s_{iw} obtained from different attention heads. Note that we learn a separate copy of the parameters W_w, b_w and p_w for each attention head.

$$s_i = \frac{\sum_w s_{iw}}{NH_W} \tag{4}$$

where NH_W is the number of word-level attention heads.

Given the sentence vectors s_i, we can generate the document vector v_d in a similar way. It captures the bias related information in news articles and can be used as features for predicting the document bias label.

$$f_d = W_c v_d + b_c \tag{5}$$

$$p_d = \text{softmax}(f_d) \tag{6}$$

We use the negative log likelihood of the correct labels as classification training loss:

$$L = -\sum_d \log p_{dj} \tag{7}$$

where j is the bias label of d.

3.2 Political Entities

News articles, especially the ones we are interested in in this work, are mainly covering real-world events involving political entities and their relations. To better understand the stance over controversial issues and the underlying ideology reflected in the text, it is very important to have extensive world knowledge about these entities, including their traits, opinions, and relevant events. We obtain the entity knowledge representations through learning on Wikipedia data.

Wikipedia2Vec (Yamada et al., 2018) is a model that learns entity embeddings from Wikipedia. It learns embeddings of words and entities by iterating over the entire Wikipedia pages and maps similar words and entities close to one another in a continuous vector space. It jointly optimizes the following three submodels:

1. Wikipedia link graph model, which learns entity embeddings by predicting neighboring entities in Wikipedia's link graph, an undirected graph whose nodes are entities and edges represent links between entities in their Wikipedia pages.

2. Word-based skip-gram model, which learns word embeddings by predicting neighboring words given each word on a Wikipedia page.

3. Anchor context model, which aims to place similar words and entities near one another in the vector space. The objective here is to predict neighboring words given each entity referred to on a Wikipedia page.

The learned entity embeddings encode the background knowledge about these entities in Wikipedia, such as gender, ideology, among others. We use them to initialize our entity embeddings in Section 4.1 which enables us to inject background knowledge of entities to the text model through pre-training.

3.3 Social Information Graph

With the great popularity of social media platforms, many people nowadays tend to share their personal interests and opinions and exchange ideas about social events with others online. This also applies to the sharing of news articles on social media. Intuitively, news articles shared by the same user are likely to have the same bias, and users who share a lot of news in common are close in their political preferences as well. Hence, we can use this information to guide the pre-training of our text model.

We follow the work in (Li and Goldwasser, 2019) to learn the embeddings through the structure of the social information graph for users who share articles. The graph consists of three types of vertices, namely political users, sharing users, and news articles. Political users are famous politicians or journalists with a clear, self-reported political bias. Sharing users are Twitter users who shared news articles in the dataset. There are two types of edges: 1) following edge between a sharing user to a political user and 2) sharing edge between a sharing user to a news article). Graph Convolutional Networks (GCN) are used to model the graph structure to predict the bias of political users. It aggregates information from the local neighborhood for each node in the graph. Therefore the training of GCN helps to propagate political preference information from political users to sharing users. We use the learned embeddings to guide the pre-training in Section 4.2 so that our text model can use this as distant supervision to map the representation of news articles shared by the same user to be close in the vector space since they are more likely to have the same perspective.

3.4 Frame Indicators

Political framing, studied by political scientists, provides a useful way to study different political perspectives. The frames surrounding an issue can change the reader's perception without having to alter the actual facts as the same information is used as a base. It is a political strategy that used to bias the discussion on an issue toward a specific stance. For example, regarding the topic of abortion, the liberal side will highlight the freedom of choice for women to decide whether to terminate a pregnancy while the conservative side may emphasize the morality aspect instead, arguing the right of the fetus.

Previous work (Roy and Goldwasser, 2020) shows that frame indicators can be used to identify the political perspectives effectively for different topics. These are words that have high pointwise mutual information with a specific frame. They can be considered to represent a more detailed point within a frame. Therefore we propose to use these frame indicators to guide the pre-training of text models so that they can learn to distinguish the nuance between different frames and talking points.

4 Pre-training

As discussed in the introduction, the supervision on news bias requires a lot of human effort to get. Moreover, the text model trained only on the political perspective labels cannot benefit from the rich knowledge we have from the various social and linguistic contexts presented in the previous section. To enhance the performance of political perspective identification, we may need to bring external knowledge and signals from the aforementioned contexts to enable the text model to take them into account when processing the news article. Eventually, we want to show that the model works best by exploiting all different kinds of knowledge and signals related to the task.

4.1 Entity Guided Pre-training

The goal of entity-guided pre-training is to inject knowledge about entities into our text model to help solve the political perspective identification problem. We first extract entities from the data corpus and then learn knowledge representations for them using Wikipedia2Vec introduced in 3.2. We then use the predicted

We utilize the entity linking system DBpedia Spotlight (Daiber et al., 2013) to recognize and disambiguate the entities in news articles. We use the default configuration of DBpedia Spotlight, including the confidence threshold of 0.35, which helps to exclude uncertain or wrong entity annotations. We keep only entities with Person or Organization types that appear in the corpus.

Inspired by the masked language modeling objective used in BERT (Devlin et al., 2019), we propose an entity-level masking task for injecting background knowledge of entities into the text model based on the news articles in which they are mentioned. The objective is to predict the masked entity based on the context provided by the other words in a sentence. Specifically, the entity mentions (regardless of the number of tokens in text) are replaced with a special token "[MASK]" during preprocessing. We use a bidirectional LSTM (sentence level encoder described in 3.1) to encode the sentence, and the hidden state of the mask token will be used for prediction. We use negative sampling to randomly generate negative entity candidates from all entities in our dictionary uniformly. The prediction can be done by comparing the similarity score between the hidden state and the embedding of candidate entities mapped to the same space through a hidden layer.

$$h_{it}^{T} \cdot (W_e v_e + b_e) \tag{8}$$

where h_{it} is the hidden state for the masked token, v_e the embedding of entity e, W_e and b_e the parameters for the mapping hidden layer. We use the multi-class cross-entropy loss for all pre-training tasks.

The learned sentence encoder will then be able to highlight the context in the news articles that is more related to the properties and of the mentioned entities.

4.2 Sharing User Guided Pre-training

As we discussed in Section 3.3, the sharing behavior by Twitter users can be regarded as signals to guide the pre-training of our text model. In order to benefit from the social information available, we propose to predict the sharing user given a news article. Similar to the previous part, we use negative sampling to generate negative sharing user candidates uniformly. The prediction is based on similarity scores defined below

$$v_{v_d}^{T} \cdot (W_s v_s + b_s) \tag{9}$$

where v_d is the document vector for d, v_s the embedding of sharing user s, W_s and b_s the parameters for the hidden layer.

4.3 Frame Indicator Guided Pre-training

The frame indicator guided pre-training is almost identical to the entity guided one except that the masked tokens are frame indicators instead of entity mentions.

4.4 Ensemble of Multiple Models

Given the entity and user embeddings are not in the same space, we use them to pre-train separate models. All pre-trained models are then trained with the supervision of political perspective labels in the same way. We also explore an ensemble of the three models, which makes predictions based on a weighted sum of unnormalized scores f_d in equation 5 from these models at test time.

$$\sum_{m} f_{dm} * \beta_m \tag{10}$$

where m denotes a trained prediction model, f_{dm} the unnormalized scores for document d by model m and β_m the weight given to model m which can be tuned based on the data.

5 Experiments

We aim to answer the following research questions (RQs) in the experiment:

RQ1: what is the performance gain of pre-training the text model with each social and linguistic information, with respect to the baseline models?

RQ2: what is the respective contribution by the individual pre-trained models to the full ensemble model?

RQ3: how will the performance gain change given the different amount of labeled data available for training?

5.1 Datasets and Evaluation

We run experiments on two news article datasets: Allsides and SemEval. The statistics of both datasets are shown in Table 1.

Allsides This dataset (Li and Goldwasser, 2019) is collected from two news aggregation websites[4] on 2020 different events discussing 94 event types. The websites provide news coverage from multiple perspectives, indicating the bias of each article using crowdsourced and editorial reviewed approaches. Each article has a political perspective label left, center, or right. We used the same randomly separated splits for evaluation in this paper so that our results are directly comparable with previous ones.

SemEval This is the official training dataset from SemEval 2019 Task 4: Hyperpartisan News Detection (Kiesel et al., 2019). The task is to decide whether a given news article follows a hyperpartisan argumentation. There are 645 articles in this dataset and each is labeled manually with a binary label to indicate whether it is hyperpartisan or not. Since the test set is not available at this time. We conducted 10-fold cross-validation on the training set with the exact same splits so that we can compare with the system that ranked in the first place.

Dataset	Center	Left	Right	Avg # Sent.	Avg # Words
Allsides	4164	3931	2290	49.96	1040.05
		Hyperpartisan			
SemEval	407	238		27.11	494.29

Table 1: Datasets Statistics.

---

[4] `Allsides.com` and `Memeorandum.com`

5.2 Baselines

We compare our model with several competitive baseline methods.

BERT is a language representation model based on deep bidirectional Transformer architectures (Vaswani et al., 2017). It was pre-trained with the masked language model and next sentence prediction tasks on a huge corpus. As a result, it can achieve state-of-the-art results on a wide range of tasks by fine-tuning with just one additional output layer.

CNN_Glove (CNN_ELMo) is the model from the team that ranked first in hyperpartisan news detection task in SemEval 2019 (Jiang et al., 2019). It uses the pre-trained Glove (ELMo) word vectors, which are then averaged as sentence representations. The sentence vectors are fed into 5 convolutional layers of different kernel sizes. The outputs for all convolution layers are combined to form the input to a fully connected layer, which maps to the final text representation. Some extra improvements include batch normalization and ensemble of multiple models.

5.3 Implementation Details

We use the spaCy toolkit for preprocessing the documents. All models are implemented with PyTorch (Paszke et al., 2017)[5]. The 300d Glove word vectors (Pennington et al., 2014) trained on 6 billion tokens are used to convert words to word embeddings. The ELMo model we used is the medium one with output size 512. They are not updated during training. The sizes of LSTM hidden states for both word level and sentence level are 300 for both Allsides and SemEval datasets. The number of attention heads at both word and sentence levels is set to 4 for the Allsides dataset, while it is set to 1 for the SemEval dataset due to its size. For the training of the neural network, we used the Adam optimizer (Kingma and Ba, 2014) to update parameters. On Allsides dataset, 5% of the training data is used as the validation set. We perform early stopping using the validation set. However, same as (Jiang et al., 2019), we use the evaluation part of each fold for early stopping and model selection due to the limited size of the SemEval dataset. The patience for early stopping p is equal to 10, meaning that the training stops when there is no improvement in validation performance for ten consecutive epochs. The learning rate lr is set to 0.001 for all models except BERT for which $2e-5$ is used. The mini-batch size $b = 10$ for bias prediction.

Regarding pre-training data sources, we use the training set for Allsides, and extract 100,000 news articles for SemEval from the large dataset provided by SemEval 2019 Task 4. The entity and user embeddings used for pre-training are obtained through external resources described in Section 3.2 and 3.3. The embeddings for frame indicators are randomly initialized. All of them were updated during the pre-training to better adapt to the text model. The optimizer and most hyper-parameters stay the same as the training of bias prediction. The mini-batch size is set to 2000 and 300 for models using Glove and ELMo respectively since the training is at the sentence level.

5.4 Results

5.4.1 Results on Allsides

We report the average accuracy and macro F1 scores on test sets for Allsides dataset in Table 2. The results are divided into two groups based on whether contextualized word representations are used. To answer RQ1, we observed that, in most cases, models with pre-training outperform the MAN baseline. It demonstrates our pre-training step can effectively utilize signals in social and linguistic context to enhance the text model to identify bias expressed in more subtle ways. Therefore it generates high-quality document representation for political perspective prediction. The sharing guided pre-training did not lead to much improvement by itself. This is mainly because the sharing users in our dataset often share news articles with various perspectives. Our ensemble model achieves the best result in terms of both accuracy and macro F1 scores no matter whether contextualized word embeddings are used or not. It shows the signals from various sources are complementary with each other such that even a simple combination of prediction scores can lead to significant improvement. The gaps between our model and baselines decrease when contextualized word representations are used since local context is better captured in this setting.

5.4.2 Results on SemEval

The performance of various models on the SemEval dataset can be found in Table 3. Note that

---

[5] Please refer to `https://github.com/BillMcGrady/NewsBiasPretraining` for data and source code.

Model	Accuracy	Macro F1
MAN_Glove	78.29	76.96
+ Entity	80.50	79.50
+ Sharing	78.93	77.84
+ Frame	81.26	80.15
Ensemble	**83.74**	**82.84**
BERT	81.55	80.13
MAN_ELMO	81.41	80.44
+ Entity	82.27	81.23
+ Sharing	81.37	80.48
+ Frame	82.56	81.66
Ensemble	**85.00**	**84.25**

Table 2: Test Results on Allsides Dataset.

there is no sharing user guided result in this table since we do not have social graph information available in this dataset. Again the results are grouped based on word representation used. CNN_Glove and CNN_ELMo are results reported by the winning team in the SemEval competition. They proposed an ensemble of multiple CNN models where each CNN takes sentence representation generated by average ELMo embedding as input. It is worth noting that our model with Glove as word representation is comparable with the winning team's model with ELMo, showing the advantages of pre-training. The other trends hold as well in the SemEval dataset. In both datasets, our pre-trained models beat BERT easily since they are tuned specifically for the task.

Model	Accuracy	Macro F1
CNN_Glove †	79.63	-
MAN_Glove	81.58	79.29
+ Entity	82.65	80.75
+ Frame	83.27	81.73
Ensemble	**84.03**	**82.42**
CNN_ELMO †	84.04	-
BERT	84.03	82.60
MAN_ELMO	84.66	83.09
+ Entity	85.59	84.15
+ Frame	85.27	83.32
Ensemble	**86.21**	**84.33**

Table 3: Test Results on SemEval Dataset. † indicates results reported in (Jiang et al., 2019).

5.4.3 Ablation Study

To answer RQ2, we show the results for ablations of our ensemble model based on MAN_Glove in Table 4. The performance drops when removing each one of the pre-trained models from the ensemble, showing that the information obtained from different sources is complementary with each other. To make a fair comparison with the baseline model, we also report the performance of an ensemble of multiple baseline models (denoted as -Pre-training) with different seeds from random initialization. This shows the absolute gain through pre-training to adapt the text representations for political perspective identification.

Model	Accuracy	Macro F1
Ensemble	83.74	82.84
- Entity	82.57	81.65
- Sharing	82.78	81.78
- Frame	82.39	81.40
- Pre-training	81.54	80.40

Table 4: Ablation Study on Allsides Dataset.

5.4.4 Results with Limited Training Data

One of the obstacles in obtaining good performance in political perspective identification tasks is the lack of supervision data. We compare the performance of the MAN_Glove model with and without pre-training with different levels of training examples available in Figure 2. These results can help to answer RQ3. It shows that the performance gain obtained from our pre-training strategy increases as the size of the training set decreases. This is a very useful property as it can greatly improve model performance when there is limited training data. It is worth noting that the Sharing-Guided Pre-training achieves much higher performance when supervision is limited. This is because the signals from the sharing users can be considered as noisy bias labels and it is trained at document level instead of sentence level like the other two. However, since the other two pre-training methods introduce extra knowledge to the text model, they can lead to better performance when the supervision is abundant to provide enough bias information for training.

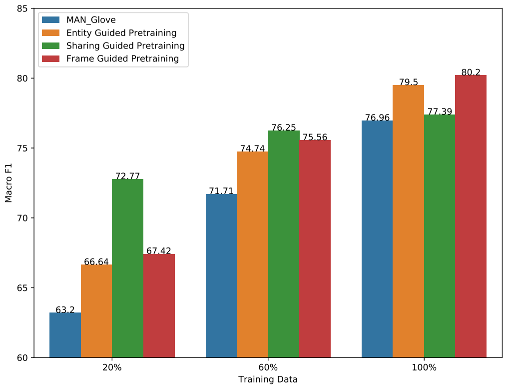

Figure 2: Test Results with Different Number of Training Examples.

5.4.5 Qualitative Results

Human Annotation Comparison The BASIL dataset (Fan et al., 2019) has human annotations of bias spans. It contains 300 articles on 100 events with 1727 bias spans annotated. On the sentence level, spans of lexical and informational bias are identified by annotators by analyzing whether the text tends to affect a reader's feeling towards one of the main entities. We compute the average attention assigned by our model to the annotated bias spans. Table 5 shows the results of the baseline model (MAN) and the same model pre-trained with entity information (+Entity). The attention scores assigned to the human annotation spans are higher across training, validation, and test sets.

Model	Training	Validation	Test
MAN	0.706	0.701	0.652
+ Entity	0.737	0.728	0.666
Improvement	4.36%	3.76%	2.13%

Table 5: Average Attention Scores on Basil Annotations.

6 Conclusion

In this work, we propose a pre-training framework to adapt text representation for political perspective identification. Empirical experiments on two recent news article datasets show that an ensemble of pre-trained models achieves significantly better performance in bias detection compared to competitive text baselines. It is also shown that our pre-training model can achieve even larger performance gain when the supervision is limited.

In fact, these various context information are not independent. We intend to extend this work to pre-train better text models by incorporating information from various sources together.

Acknowledgements

We thank the anonymous reviewers for their helpful feedback. This works was supported by an NSF CAREER award IIS-2048001.

References

E. Baumer, E. Elovic, Y. Qin, F. Polletta, and G. Gay. 2015. Testing and comparing computational approaches for identifying the language of framing in political news. In *NAACL*, pages 1472–1482, Denver, Colorado. Association for Computational Linguistics.

C. Budak, S. Goel, and J. M. Rao. 2016. Fair and balanced? quantifying media bias through crowd-sourced content analysis. *Public Opinion Quarterly*, 80(S1):250–271.

Dallas Card, Justin Gross, Amber Boydstun, and Noah A. Smith. 2016. Analyzing framing through the casts of characters in the news. In *EMNLP*, pages 1410–1420, Austin, Texas. Association for Computational Linguistics.

J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *I-SEMANTICS*, I-SEMANTICS '13, pages 121–124, New York, NY, USA. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Heba Elfardy, Mona Diab, and Chris Callison-Burch. 2015. Ideological perspective detection using semantic features. In *STARSEM*, pages 137–146, Denver, Colorado. Association for Computational Linguistics.

L. Fan, M. White, E. Sharma, R. Su, P. K. Choubey, R. Huang, and L. Wang. 2019. Media bias through the lens of factual reporting. In *EMNLP*.

A. Field, D. Kliger, S. Wintner, J. Pan, D. Jurafsky, and Y. Tsvetkov. 2018. Framing and agenda-setting in Russian news: a computational analysis of intricate political strategies. In *EMNLP*, pages 3570–3580, Brussels, Belgium. Association for Computational Linguistics.

M. Gardner, J. Grus, M. Neuman, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L.S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Matthew Gentzkow and Jesse M Shapiro. 2010. What drives media slant? evidence from us daily newspapers. *Econometrica*, 78(1):35–71.

Matthew Gentzkow and Jesse M Shapiro. 2011. Ideological segregation online and offline. *The Quarterly Journal of Economics*, 126(4):1799–1839.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *NAACL-HLT*, pages 503–511, Boulder, Colorado. Association for Computational Linguistics.

K. Hanawa, S. Sasaki, H. Ouchi, J. Suzuki, and K. Inui. 2019. The sally smedley hyperpartisan news detector at SemEval-2019 task 4. In *SemEval*, pages 1057–1061, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *ACL*, pages 1113–1122, Baltimore, Maryland. Association for Computational Linguistics.

Ye Jiang, Johann Petrak, Xingyi Song, Kalina Bontcheva, and Diana Maynard. 2019. Hyperpartisan news detection using ELMo sentence representation convolutional network. In *SemEval*, pages 840–844, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Kristen Johnson and Dan Goldwasser. 2016. Identifying stance by analyzing political discourse on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 66–75.

Kristen Johnson, Di Jin, and Dan Goldwasser. 2017. Modeling of political discourse framing on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

J. Kiesel, M. Mestre, R. Shukla, E. Vincent, P. Adineh, D. Corney, B. Stein, and M. Potthast. 2019. SemEval-2019 task 4:hyperpartisan news detection. pages 829–839, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

C. Li and D. Goldwasser. 2019. Encoding social information with gcn for political perspective detection in news media. In *ACL*, pages 2594–2604, Florence, Italy. Association for Computational Linguistics.

W-H Lin, T. Wilson, J. Wiebe, and A. Hauptmann. 2006. Identifying perspectives at the document and sentence levels. In *CoNLL*, CoNLL-X '06, pages 109–116, Stroudsburg, PA, USA. Association for Computational Linguistics.

Julia Mendelsohn, Ceren Budak, and David Jurgens. 2021. Modeling framing in immigration discourse on social media. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2219–2263.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

F. Morstatter, L. Wu, U. Yavanoglu, S. R. Corman, and H. Liu. 2018. Identifying framing bias in online news. *Trans. Soc. Comput.*, 1(2):5:1–5:18.

Van-Hoang Nguyen, Kazunari Sugiyama, Preslav Nakov, and Min-Yen Kan. 2020. Fang: Leveraging social context for fake news detection using graph representation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1165–1174.

Maria Leonor Pacheco and Dan Goldwasser. 2021. Modeling content and context with deep relational learning. *Transactions of the Association for Computational Linguistics*, 9:100–119.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

J. Pennington, R. Socher, and C. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

M. Recasens, C. Danescu-Niculescu-Mizil, and D. Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *ACL*, pages 1650–1659, Sofia, Bulgaria. Association for Computational Linguistics.

Shamik Roy and Dan Goldwasser. 2020. Weakly supervised learning of nuanced frames for analyzing polarization in news media. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7698–7716, Online. Association for Computational Linguistics.

Chereen Shurafa, Kareem Darwish, and Wajdi Zaghouani. 2020. Political framing: Us covid19 blame game. In *International Conference on Social Informatics*, pages 333–351. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*, WWW' 18, pages 1835–1844, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

J. Wang, Z. Wang, D. Zhang, and J. Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, IJCAI, pages 2915–2921. AAAI Press.

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, Minneapolis, Minnesota. Association for Computational Linguistics.

I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Takefuji. 2018. Wikipedia2vec: An optimized tool for learning embeddings of words and entities from wikipedia. *arXiv*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Enhancing Dialogue-based Relation Extraction by Speaker and Trigger Words Prediction

Tianyang Zhao◊§[*], **Zhao Yan**§, **Yunbo Cao**§, **Zhoujun Li**◊[†]
◊State Key Lab of Software Development Environment, Beihang University, Beijing, China
§Tencent, Beijing, China
◊{tyzhao,lizj}@buaa.edu.cn; §{zhaoyan,yunbocao}@tencent.com

Abstract

Identifying relations from dialogues is more challenging than traditional sentence-level relation extraction (RE), since the difficulties of speaker information representation and the long-range semantic reasoning. Despite the successful efforts, existing methods do not fully consider the particularity of dialogues, making them difficult to truly understand the semantics between conversational arguments. In this paper, we propose two beneficial tasks, speaker prediction and trigger words prediction, to enhance the extraction of dialogue-based relations. Specifically, speaker prediction captures the characteristics of speaker-related entities, and the trigger words prediction provides supportive contexts for relations between arguments. Extensive experiments on the DialogRE dataset show noticeable improvements compared to the baseline models, which achieves a new state-of-the-art performance with a 65.5% of F1 score and a 60.5% of F1$_c$ score, respectively.

Dialogue 1

S1: Yeah you see umm, well, I'm an actor. Right? ……

Relation: (***S1***, per:title, actor)
Trigger Words: -

Dialogue 2

S1: Mom!
S2: Sweetie! So this is where you work? ……

Relation: (***S1***, per:parents, ***S2***)
Trigger Words: mom

Dialogue 3

S1: Hello, Mr. Bing.
S2: Loved your Stevie Wonder last night.
S3: Thanks. Listen, about the weekly numbers, I'm gonna need them on my desk by nine o'clock.
S1: Sure.
S2: No problem.

Relation: (***S1***, per:boss, ***S3***)
(***S2***, per:boss, ***S3***)
Trigger Words: -

Figure 1: Examples from the DialogRE dataset. s_n denotes the speaker of each utterance. The underlined text indicates the relation between the argument pairs.

1 Introduction

The task of relation extraction is to identify the relation facts between two arguments from plain text, which is the fundamental step of many natural language processing applications. Recent years have seen increasing efforts on sentence-level RE, e.g., relations only hold within a single sentence (Fu et al., 2019; Luan et al., 2019; Zhao et al., 2020; Wang and Lu, 2020; Wei et al., 2020). To adapt to complex scenarios, some current works have moved forward to the document-level RE, e.g., relations can exist across multiple sentences (Yao et al., 2019; Wang et al., 2020; Nan et al., 2020; Jain et al., 2020; Zhou et al., 2021).

A more challenging yet practical extension is the dialogue-based relation extraction. The dialogues contain multi-turn conversations among a group of speakers. The relations not only exist between the entities in the dialogue text but also the speakers of each utterance. Additionally, most of relations appear in multi-turn conversation, which require cross-sentence extraction. Considering the complexities, we divide the dialogue-based RE into three categories. In the first category, the relation can be directly inferred from the current utterance, as shown in the Dialogue 1 of Figure 1. In the second category, the relation involves utterances among multiple speakers and there is clear evidence in the dialogue that triggers the relation. Regarding the Dialogue 2 in Figure 1, "mom" is the trigger word of the relation `per:parents` be-

---

[*] Work done during an internship at Tencent Cloud Xiaowei.

[†] Corresponding Author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4580–4585
August 1–6, 2021.

tween “S1” and “S2”. Nevertheless, there are still cases where there is no clear context indicating the relationship. As shown in the Dialogue 3 of Figure 1, the relation between “S1” and “S3” as well as the relation between “S2” and “S3” can only be inferred from the tones and expression habits of speakers. Therefore, to identify relations from the complex dialogues, it is necessary to 1) discover highly supportive information about the arguments, and 2) capture the unique features of speakers.

Existing studies propose to solve this task through a speaker-aware BERT model (Yu et al., 2020) as well as a gaussian graph-based method (Xue et al., 2021). The former modifies the speaker arguments in dialogue text with special tokens to highlight the speaker-related information. The latter builds a latent multi-view graph to encode the long-distance dependency between arguments. However, these works regard dialogue as a plain text without considering the supporting information of relations and the characteristics of speakers. As been emphasized before (Xue et al., 2021), trigger words and speaker-related features play a critical role in dialogue-based relation extraction. In this case, it is difficult for them to detect the speaker-related relations from the complicated conversations.

To address the above limitations, we propose two beneficial tasks, speaker prediction and trigger words prediction, to enhance the dialogue-based relation extraction. Specifically, the speaker prediction task aims to capture the unique features of speakers. We randomly mask the speaker tokens and use the context to predict who said the utterance. The trigger words prediction task is to detect the supportive context of the current relation. We solve it with a sequence labeling method. Moreover, we design an integration module for the relation extraction task to combine both the global dialogue representation and the local arguments representation. Finally, the three tasks are jointly trained based on a multi-task learning framework.

The contributions of our work are summarized as follows. We propose two beneficial tasks, speaker prediction and trigger words prediction, to capture the unique features of speakers and detect the supportive information about arguments, both effectively enhance the dialogue-based relation extraction. We evaluate our method on the DialogRE dataset and achieve a new state-of-the-art performance with 65.5% of F1 score and 60.5% of F1$_c$ score, respectively. Our code is available at `https://github.com/TanyaZhao/DialogRE-Trigger-Speaker-Prediction`.

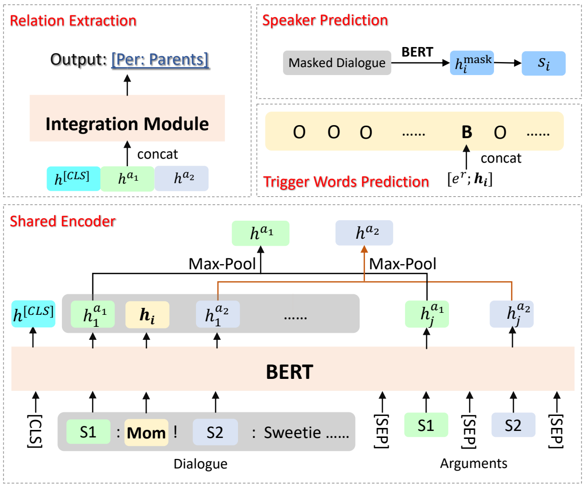

Figure 2: Overall structure of our method.

2 Problem Definition

Given a dialogue $d = s_1 : t_1, s_2 : t_2, ..., s_m : t_m$ and two arguments a_1, a_2, where s_i and t_i are the speaker and the utterance of the i-th turn, $t_i = x_{i1}, x_{i2}, ..., x_{in}$ is consisted of n words. The dialogue-based relation extraction aims to predict the relation type $r \in \mathcal{R}$ between a_1 and a_2, where $\mathcal{R}$ is the set of predefined relation categories.

3 Methodolgy

This section introduces the structure of our method, including three tasks, relation extraction, trigger words prediction and speaker prediction. Figure 2 shows the overall structure of our method.

3.1 Relation Extraction Task

The relation extraction task takes the dialogue d and the argument pair (a_1,a_2) as input and outputs a relation type r between the two arguments. For the dialogue d, we first modify it into $\hat{d} = \hat{s_1} : \hat{t_1}, \hat{s_2} : \hat{t_2}, ..., \hat{s_m} : \hat{t_m}$, where

$$\hat{s}_i = \begin{cases} \texttt{[B] [S}_1\texttt{] [E]} & \text{if } s_i = a_1 \\ \texttt{[B] [S}_2\texttt{] [E]} & \text{if } s_i = a_2 \\ s_i & \text{otherwise} \end{cases}, \quad (1)$$

$\hat{t_i} = x_{i1}..., \texttt{[B]}, a_k, \texttt{[E]}, ..., x_{in}, k \in \{1, 2\}$, if t_i contains a_k. Among them, `[S1]`, `[S2]`, `[B]`, `[E]` are newly-defined tokens. `[B]` and `[E]` are used to mark the start and the end position of the argument. We further replace the argument a_k

with the pre-defined token $\texttt{[S}_{\texttt{k}}\texttt{]}$, as $\hat{a_k} = \texttt{[S}_{\texttt{k}}\texttt{]}$ if $\exists i(s_i = a_k)$. Then, we concatenate $\hat{d}$, $\hat{a_1}$ and $\hat{a_2}$ as a sequence, and use special tokens `[CLS]` and `[SEP]` as sperators, formulated as

$$\texttt{[CLS]}\hat{d}\texttt{[SEP]}\hat{a_1}\texttt{[SEP]}\hat{a_2}\texttt{[SEP]}. \tag{2}$$

We feed the sequence into the pre-trained language model BERT (Devlin et al., 2019) and obtain the hidden semantic representation of each input tokens. Among them, $h^{[CLS]} \in \mathbb{R}^{d_h}$ is the hidden output of `[CLS]`, where d_h is the hidden size of BERT. We use $h^{[CLS]}$ to represent the global relational feature between a_1 and a_2.

To better represent the semantic information of arguments, we distill all the hidden states of a_k's start marker in the sequence (Eq. 2), including those in the dialogue text, and formulate them as $h_1^{a_k}, h_2^{a_k}, ..., h_j^{a_k} \in \mathbb{R}^{d_h}$. Then, we apply a max pooling process to obtain a combined representation of a_k:

$$h^{a_k} = \text{max-pool}(h_1^{a_k}, h_2^{a_k}, ..., h_j^{a_k}). \tag{3}$$

Next, we concatenate $h^{[CLS]}$, h^{a_1} and h^{a_2} as $h = [h^{[CLS]}; h^{a_1}; h^{a_2}] \in \mathbb{R}^{3d_h}$. Note that, $h^{[CLS]}$ is the global relational information of the sequence, and h^{a_1}, h^{a_2} are the local features of the arguments.

Furthermore, we propose an integration module to enhance the correlation between the dialogue and arguments. Specifically, h is fed into a two-layer highway network (Srivastava et al., 2015) as

$$\begin{aligned} \hat{h} &= H(h) * T(h) + h * (1 - T(h)), \\ T(h) &= \sigma(W^t h + b^t), \end{aligned} \tag{4}$$

where $W^t \in \mathbb{R}^{3d_h \times d_t}$, $b^t \in \mathbb{R}^{d_t}$ are learned weights with d_t as the hidden size of the highway network. Finally, we conduct a multi-class classification to calculate the probability of the relation between a_1 and a_2 by

$$\Pr(\text{relation} = r | d, a_1, a_2) = \text{sigmoid}(W^r h + b^r), \tag{5}$$

where $W^r \in \mathbb{R}^{d_t \times |\mathcal{R}|}$, $b^r \in \mathbb{R}^{|\mathcal{R}|}$.

3.2 Trigger Words Prediction Task

Generally, to identify the relation between two arguments, it is necessary to detect the supportive context that triggers the relation. Yu et al. (2020) have verified that trigger words play an important role for relation extraction. However, their work directly append the ground truth trigger words to the input sequence, which is not feasible for scenarios where the golden triggers are not available. The intuitive idea is to predict the trigger words from the conversation. Therefore, we can still obtain supporting information without relying on the golden triggers to guide the relationship extraction.

We propose the trigger words prediction task, which applies a simple and effective way to improve the relation extraction task. Specifically, we perform sequence labeling over the hidden outputs of BERT. Considering the trigger words are closely related to the relation, we first map the predicted relation r (Eq. 5) into a distributed embedding $e^r \in \mathbb{R}^{d_r}$, and concatenate it with each hidden output of BERT as $z_i = [e^r; h_i]$[1]. Then, we predict the boundary label for every token. Formally, the probability of the token x_i with label l is calculated by

$$\Pr(\text{label} = l | x_i) = \text{softmax}(W^l z_i + b^l), \tag{6}$$

where $W^l \in \mathbb{R}^{(d_h + d_r) \times |\mathcal{B}|}$ and $b^l \in \mathbb{R}^{|\mathcal{B}|}$ with $\mathcal{B} = \{\texttt{B}, \texttt{I}, \texttt{O}\}$.

3.3 Speaker Prediction Task

Notably, a majority of relations in the dialogue-based RE are associated with the speakers. For example, the triplet (S1, `per:parents`, S2) in Figure 1. Different from the ordinary entities, speakers have distinctive personal features, including tone of voices, expression habits, etc., which are important indicators for relation extraction. Therefore, we further propose the speaker prediction task based on the discourse structure to capture the speaker-related features. The motivation behind it is that if the model can distinguish who said the utterance, it learned the speaker's unique information, which is helpful to the speaker-related relation prediction.

Concretely, we randomly select the speaker words s_i in d with a probability of 10% and replace them with a special token `[MASK]`. Next, the BERT model takes the modified sequence as input and obtain the hidden state of each masked speaker, denoted as h_i^{mask}. Then, we predict the speaker through a multi-type classification as

$$\Pr(s_i | s_i^{\text{mask}}) = \text{softmax}(W^s h_i^{\text{mask}} + b^s), \tag{7}$$

where $W^s \in \mathbb{R}^{d_h \times S}$, $b^r \in \mathbb{R}^S$ with S as the maximum number of speakers in dialogues.

---

[1] We use the golden relation during training.

Model	Dev		Test	
	F1(σ)	**F1**$_c$(σ)	**F1**(σ)	**F1**$_c$(σ)
BERT(Devlin et al., 2019)	60.6 (1.2)	55.4 (0.9)	58.5 (2.0)	53.2 (1.6)
BERTs(Yu et al., 2020)	63.0 (1.5)	57.3 (1.2)	61.2 (0.9)	55.4 (0.9)
GDPNet(Xue et al., 2021)	**67.1 (1.0)**	61.5 (0.8)	64.9 (1.1)	60.1 (0.9)
Ours	66.8 (0.9)	**61.5 (1.0)**	**65.5 (0.7)**	**60.5 (0.8)**

Table 1: Performance comparison of our method with the existing advanced models on DialogRE dataset. σ denotes the standard deviation of 5 runs with different initial random seeds.

3.4 Joint Training Objective

The above three tasks share the BERT encoder and are jointly trained based on the multi-task learning framework. During training, we minimize the following objective loss function as

$$\mathcal{L} = \mathcal{L}_{\mathrm{RE}} + \mathcal{L}_{\mathrm{TP}} + \mathcal{L}_{\mathrm{SP}}, \tag{8}$$

where $\mathcal{L}_{\mathrm{RE}}$ is the binary cross-entropy loss for relation extraction, $\mathcal{L}_{\mathrm{TP}}$ and $\mathcal{L}_{\mathrm{SP}}$ are the cross-entropy loss for trigger words prediction and speaker prediction, respectively. For inference, we directly use the relation predicted by Equation 5 as the final result.

4 Experiments

In this section, we compare the proposed method with the current state-of-the-art approaches to evaluate its effectiveness.

4.1 Experimental Setup

Dataset We conduct experiments on the dialogue-based RE benchmark dataset, **DialogRE** (Yu et al., 2020). It contains $1,788$ dialogues from the transcripts of *Friends* corpus, totally with 36 relation types. 49.6% of relation triples is annotated with trigger words.

Evaluation Metrics Following the previous work (Yu et al., 2020), we adopt F1 score and F1$_c$ score as the evaluation metrics. Among them, F1$_c$ is a supplement to the F1, which only considers the first $i \leq m$ turns of utterances, rather than the entire dialogue.

Baseline Models We compare our method with the existing advanced models, **BERT**(Devlin et al., 2019), **BERTs** (Yu et al., 2020) and **GDPNet** (Xue et al., 2021). **BERT** model for the dialogue-based RE directly applies BERT as the dialogue encoder, and uses the hidden state of `[CLS]` for relation prediction. **BERTs** is a speaker-awared BERT, with modifies the speaker tokens to special markers. **GDPNet** uses a gaussian graph-based network to capture the interaction within dialogues, and achieves the current state-of-the-art performance.

Model	Test	
	F1(σ)	**F1**$_c$(σ)
Ours	**65.5 (0.7)**	**60.5 (0.8)**
Ours w/o SP	65.4(0.6)	59.8 (0.6)
Ours w/o TP	63.5 (0.9)	58.8 (1.0)
Ours w/o SP and TP	63.0 (0.7)	58.0 (0.9)
BERTs	61.2 (0.8)	55.4 (0.9)

Table 2: Ablation study to investigate the influence of each proposed task. SP and TP denote speaker prediction and trigger words prediction, respectively.

4.2 Experimental Results

Main Results Table 1 presents the performance comparison of our method with the existing advanced models. The results show that our method obviously outperforms the previous models and achieve a new state-of-the-art on test set with a F1 score of 65.5% and a F1$_c$ score of 60.5%, demonstrating the effectiveness of the proposed method.

Ablation Study We conduct ablation study experiments to investigate the influence of each proposed task. Table 2 shows the results. We can observe that, 1) *Ours w/o SP*, which removes the speaker prediction task. This causes a performance drop on all metrics, expecially with a drop of 0.7% on F1$_c$ score. 2) *Ours w/o TP*, which eliminates the trigger prediction task. The performance in terms of F1 and F1$_c$ decreases by 2.0% and 1.7%, respectively, demonstrating the importance role of trigger words prediction. 3) *Ours w/o SP and TP*, which detaches both speaker and trigger words prediction tasks. In this case, the performance further drops 0.5% and 0.8% in terms of F1 and F1$_c$. Therefore, the results above indicate that both the two tasks are beneficial to the dialogue-based RE. 4) Note that,

Case 1	**S1:** Mom ! **S2:** Sweetie! So this is where you work? ...
BERTs	(**S1**, `unanswerable`, **S2**) ✗
Ours	(**S1**, `per:parents`, **S2**) ✓
Case 2	**S1:** Hello, Mr. Bing. **S2:** Loved your Stevie Wonder last night. **S3:** Thanks. Listen, about the weekly numbers, I'm gonna need them on my desk by nine o'clock. **S1:** Sure. **S2:** No problem.
BERTs	(**S1**, `unanswerable`, **S3**) ✗ (**S2**, `unanswerable`, **S3**) ✗
Ours	(**S1**, `per:boss`, **S3**) ✓ (**S2**, `per:boss`, **S3**) ✓

Table 3: Case study on the DialogRE test set. The highlighted text indicates the trigger words recognized by our method.

although *Ours w/o SP and TP* only retains the relation extraction task, it still outperforms BERTs by a large margin. The result shows that our improved method of relation extraction is also effective.

Analysis on Discourse Structure Modeling To show the necessity of considering the discourse structure in dialogue-based RE, we design a naive way to degenerate a dialogue into a plain document. We modify the colon after a speaker into text like "said", "responsed" or "continued". For example, the Dialogue 2 in Figure 1 is converted into "*S1 said Mom! S2 responsed Sweetie! So this is where you work? ...*". Then, we apply our method to the changed text. The performance on the test set significantly degrades with 58.0% for F1 and 56.8% for F1_c. The result indicates that dialogues contain important discourse structural information. Therefore, it is important to study the extraction strategies for dialogues rather than directly applying common sentence-level or document-level extraction methods.

Trigger Words Prediction To further evaluate the effect of trigger words prediction task, we calculate the prediction performance on the cases annotated with the ground truth trigger words. Note that, 49.6% of relational triplets have trigger words in DialogRE. The prediction accuracy is 75.6%. The result demonstrates that we can correctly recognize most of the the trigger words. And with the help of the supporting information, the performance of relation extraction is considerably improved, as shown in Table 2.

Case Study We give a case study to analyze the quality of the results produced by our approach and the baseline model. Cases in Table 3 show that our method is capable of capturing the trigger words information and the characteristic of speakers. In case 1, the base model fails to utilize the trigger words information and identifies the relation as `unanswerable`. However, our method correctly recognizes that the word "Mom" triggers the relation between "S1" and "S2", which promotes the right prediction result. Besides, in case 2, our method can capture the characteristics information of speakers and thus correctly predict that "S3" is the boss of "S1" and "S2". Contrarily, the baseline model has difficulties in handling such case.

5 Conclusion

In this paper, we propose to enhance the dialogue-base relation extraction with two benefical tasks, the speaker prediction and the trigger words prediction. Extensive experiments on the benchmark dataset DialogRE demonstrate the effectiveness of our method in achieving state-of-the-art performance in both F1 score and F1_c score.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos.U1636211, 61672081, 61370126), the 2020 Tencet Wechat Rhino-Bird Focused Research Program, and the Fund of the State Key Laboratory of Software Development Environment (Grant No.SKLSDE2019ZX-17). We thank anonymous reviewers for their helpful comments.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. Graphrel: Modeling text as relational graphs for joint entity and relation extraction. In *ACL*, pages 1409–1418.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. Scirex: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *NAACL*, pages 3036–3046.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1546–1557.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Difeng Wang, Wei Hu, Ermei Cao, and Weijian Sun. 2020. Global-to-local neural networks for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3711–3721.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488.

Fuzhao Xue, Aixin Sun, Hao Zhang, and Eng Siong Chng. 2021. Gdpnet: Refining latent multi-view graph for relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777.

Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. 2020. Dialogue-based relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4927–4940.

Tianyang Zhao, Zhao Yan, Yunbo Cao, and Zhoujun Li. 2020. Asking effective and diverse questions: A machine reading comprehension based framework for joint entity-relation extraction. In *IJCAI*.

Wenxuan Zhou, Kevin Huang, Tengyu Ma, and Jing Huang. 2021. Document-level relation extraction with adaptive thresholding and localized context pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Modeling Event-Pair Relations in External Knowledge Graphs for Script Reasoning

Yucheng Zhou[1*], **Xiubo Geng**[2†], **Tao Shen**[3], **Jian Pei**[4], **Wenqiang Zhang**[1], **Daxin Jiang**[2†]
[1]Fudan University, Shanghai, China
[2]Microsoft, Beijing, China
[3]Australian AI Institute, School of CS, FEIT, University of Technology Sydney
[4]Simon Fraser University, Burnaby, Canada
`{yczhou18,wqzhang}@fudan.edu.cn, {xigeng,djiang}@microsoft.com`
`tao.shen@student.uts.edu.au, jpei@cs.sfu.ca`

Abstract

Script reasoning infers subsequent events from a given event chain, which involves the ability to understand relations between events. A human-labeled script reasoning dataset is usually of small size with limited event relations, which highlights the necessity to leverage external eventuality knowledge graphs (KG) consisting of numerous triple facts to describe the inferential relation between events. Existing methods adopt a *retrieval and integration* paradigm to focus merely on the graph triples that have event overlap with a script, but ignore much more supportive triples in the KG with similar inferential patterns, leading to under-exploiting. To fully exploit the KG, we propose a knowledge model to learn the inferential relations between events from the whole eventuality KG and then support downstream models by directly capturing the relation between events in a script. We further present a neural script adapter to extend the knowledge model for inferring the associated relations between an event chain and a subsequent event candidate. We evaluate the proposed approach on a popular multi-choice narrative cloze task for script reasoning and achieve new state-of-the-art accuracy, compared with baselines either incorporating external KG or not.

1 Introduction

Script reasoning (Chambers and Jurafsky, 2008; Li et al., 2018; Lv et al., 2020b) aims at determining the subsequent event or plausible ending for an event chain in a script. For example, a tourism script consist of ["*Emily took a plane*", "*Emily arrived at Oahu*", "*Emily went to Waimea Bay*"], and the subsequent event is more likely to be "*Emily surfed*" than "*Emily skied*". Script reasoning has attracted more interest in the natural language processing (NLP) community since it plays essential roles in many real-world applications like storytelling (Swanson and Gordon, 2008).

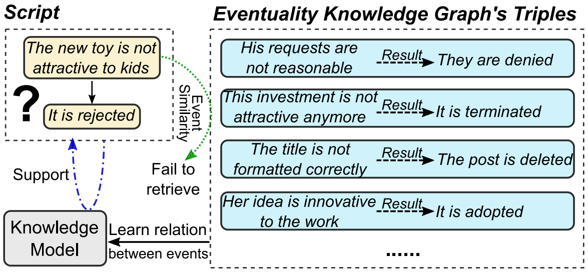

Figure 1: Comparison of "*retrieval and integration*" paradigm (green dot line) with ours (blue dash-dot line). Although there is no semantic overlap between the precedent event in the script and the events in the KG, which leads to failed retrieval, our approach still provides supportive evidence by exploiting similar inferential relation patterns.

Understanding and inferring the correlation between two events are critical for script reasoning. Taking the tourism script as an example, the key to decide the subsequent event is inferring that "*A person goes to a beach*" is more correlated to "*The person surfs*" than "*The person skies*". An immediate idea is to learn event relations from some well-labeled training datasets. Unfortunately, due to labor-intensive labeling, high-quality training data for script reasoning is usually small, from which it is impractical to learn rich relations for large scale commercial applications. Therefore, it is necessary to leverage external knowledge that implies relations between events.

Recently, Lv et al. (2020b) propose to leverage a large-scale eventuality knowledge graph (KG), ASER (Zhang et al., 2020), for script reasoning via adopting the "*retrieval and integration*" paradigm. Given an event chain, this paradigm first retrieves relevant triple facts from the eventuality KG and then integrates them into a script reasoning model.

Although such a paradigm is proven effective in entity-centric tasks (Zhang et al., 2019; Liu et al., 2020), it is not competent in event-centric script

---

[*] Work is done during internship at Microsoft.

[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4586–4596
August 1–6, 2021. ©2021 Association for Computational Linguistics

reasoning. The reason is that, the retrieval is based on lexical or semantic matching between an event from a script and each event node in the KG. For example, in Figure 1, to determine whether the precedent event "*The new toy is not attractive to kids*" will result in a subsequent event "*It is rejected*", this paradigm will try to retrieve graph triples with the event nodes talking about "*toy is not attractive*", etc., which is very likely to fail if the KG contains few related events. Namely, it dramatically narrows the focus to the graph triples merely with exact event matching, so it cannot fully leverage the external eventuality KG.

However, script reasoning can benefit from leveraging *event pairs* in KG with *similar relation patterns*, rather than the only triples in KG with *similar events*. In Figure 1, although events in the four graph triples have no semantic overlap with the precedent event "*The new toy is not attractive to kids*", all the triples can represent the relation that *if some attribute of an object is judged negatively, it might be rejected, otherwise being accepted*, which still provide strong supportive evidence between "*The new toy is not ...*" and "*It is rejected*". Therefore, script reasoning can benefit from the event pairs with similar inferential relation patterns, beyond the textual contents of the events.

Motivated by this, in this work, we propose a novel paradigm to integrate external knowledge for script reasoning by directly modeling the relation between events from a KG and thus support script reasoning in light of similar relation patterns. In particular, we first propose a *discriminative knowledge model* trained on the graph triples in an external eventuality KG. Taking each event pair in the triples as input, the knowledge model learns to predict whether two events in the pair are associated and what is the inferential relation between them. After being trained, the knowledge model can directly capture associated and inferential relations between precedent and subsequent events in a script. And the relations between events will be represented in latent space, which can be further integrated into any event-centric neural model.

Furthermore, as script reasoning requires to associate between a sequence of precedent events (i.e., an event chain) and a plausible subsequent event, we propose a neural *script adapter*, based on a chain-dependent attention module, for extending the trained knowledge model from event to script level. This leads to a *script-adaptive knowledge model* that directly represents inferential information between an event chain and a subsequent event candidate as a latent embedding. Lastly, this embedding, coupled with deep text representation from a script-text contextualizing encoder, is used to derive the plausibility score of the candidate.

We conduct empirical studies on a popular task of script reasoning, i.e., multi-choice narrative cloze (Li et al., 2018). Our approach outperforms strong competitors and achieves a new state-of-the-art accuracy, verifying the effectiveness of the script-adaptive knowledge model when integrating inferential relations into script reasoning.

2 Preliminary

This section begins with a formal task definition of script reasoning, followed by introductions to eventuality KGs and pre-trained language models.

Task Definition. Script reasoning is usually formulated as a multi-choice narrative cloze (MCNC) problem: given an event chain $\boldsymbol{e} = [e_1, \dots, e_n]$, it aims to select the most plausible subsequent event from a set of candidates $\mathcal{E}^{(c)} = \{e_1^{(c)}, \dots e_m^{(c)}\}$, where each event e consists of a sequence of words $\boldsymbol{w}^e = [w_1^e, w_2^e, \dots]$, n denotes the length of event chain $\boldsymbol{e}$, and m denotes the number of candidates $\mathcal{E}^{(c)}$. A script reasoning model is asked to produce relatedness score between the event chain and each candidate event so that

$$\hat{e} = \arg\max_{e_j} P(\boldsymbol{e}, e_j; \theta), \ \forall e_j \in \mathcal{E}^{(c)}, \qquad (1)$$

where $P(\cdot;\theta)$ denotes a θ-parameterized script reasoning model, and $\hat{e}$ denotes the predicted event.

Eventuality Knowledge Graph. In contrast to *canonical KGs* with entity-centric factoid triples, an eventuality KG, $\mathcal{G}$, typically consists of a set of event-centric triples $(e^{(h)}, r, e^{(t)})$ to describe inferential or co-occurrent relation between events. It represents each event e as free-form text, while well defines a closed-set $\mathcal{R}$ of relations so that $\forall r \in \mathcal{R}$.

3 Methodology

In this section, we will elaborate on our approach for multi-choice narrative cloze (MCNC) task in script reasoning. As shown in Figure 2, we first propose a discriminative knowledge model learning facts from eventuality graph (§3.1), followed by a novel neural adapter upgrading the knowledge model into script level (§3.2). Lastly, as in Figure 3,

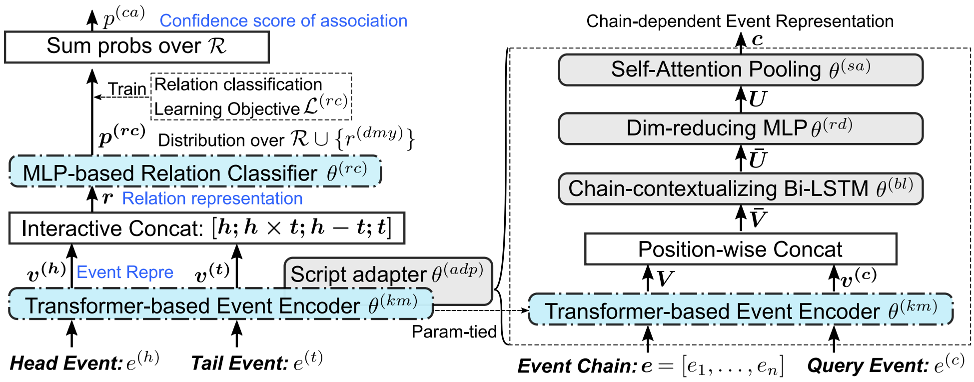

Figure 2: Our discriminative knowledge model (left), and its script adapter (right) for multi-choice narrative cloze (MCNC). Dash-dot blue rounded rectangles denote parameters optimized towards the objective of the knowledge model, whereas solid blue rounded rectangles denote script adapter's parameters that will be optimized towards the objective of MCNC.

we present a representation learning framework to solve the MCNC task (§3.3).

3.1 Discriminative Knowledge Model

To avoid challenging event grounding and satisfy coverage necessity, neural *knowledge models* (Bosselut et al., 2019b; Hwang et al., 2020) are proposed to memorize eventuality facts from a KG to its parameters during training. They are built upon a pre-trained *generative* Transformer (e.g., GPT (Radford et al., 2018)) and fine-tuned on triple facts from an eventuality KG via generative objectives of event-based *link prediction*.

However, such generative knowledge models are not perfectly compatible when capturing event-pair relation facts since they focus more on inferring tail events given a head event and an inferential relation. This is consistent with the goal of link prediction for KG completion. Consequently, if they try to model the inferential relations between events, they have to generate all possible triples for each event by traversing all relations and enlarging beam-search size (Bosselut et al., 2019a). And the generated triple must be re-encoded into latent space for the integration (Lv et al., 2020b), not to mention generative models not always reliable.

Therefore, we present a discriminative objective based on *relation classification* for knowledge model learning to directly capture such inferential information in latent space. Formally, given a triple $(e^{(h)}, r, e^{(t)}) \in \mathcal{G}$, we separately pass head event $e^{(h)}$ and tail event $e^{(h)}$, into a text encoder to generate event-level contextualized representations. Following Devlin et al. (2019), we first concatenate the natural language text $\boldsymbol{w}^e$ of each event e with special tokens:

$$\tilde{\boldsymbol{w}}^e = (\texttt{[CLS]}, \boldsymbol{w}^e, \texttt{[SEP]}), \forall e \in \{e^{(h)}, e^{(t)}\}, \quad (2)$$

where the special tokens could vary with different pre-trained models. Then, we feed the concatenated text $\tilde{\boldsymbol{w}}^e$ into a Transformer encoder, followed by a pooling layer, i.e.,

$$\boldsymbol{H}^e = \text{Transformer}(\tilde{\boldsymbol{w}}^e; \theta^{(km)}) \in \mathbb{R}^{d \times N}, \quad (3)$$

$$\boldsymbol{v} = \text{Pool}(\boldsymbol{H}^e) \in \mathbb{R}^d, \ \forall e \in \{e^{(h)}, e^{(t)}\}, \quad (4)$$

where $\boldsymbol{v}$ denotes the resulting event representation, $\text{Transformer}(\cdot; \theta)$ stands for pre-trained bidirectional Transformer encoder (e.g., BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019)) to produce deep contextualized embeddings, and $\text{Pool}(\cdot)$ denotes using the embedding of `[CLS]` as sequence-level representation by following prior works. Given $\boldsymbol{v}$ of both head and tail events, we apply an interactive concatenation (Bowman et al., 2015; Reimers and Gurevych, 2019) between them to model their inferential relationship, i.e.,

$$\boldsymbol{r} := \text{Inter-Concat}(\boldsymbol{h}, \boldsymbol{t}) = [\boldsymbol{h}; \boldsymbol{h} \times \boldsymbol{t}; \boldsymbol{h} - \boldsymbol{t}; \boldsymbol{t}], \\ \text{where } \boldsymbol{h} = \boldsymbol{v}^{(h)} \text{ and } \boldsymbol{t} = \boldsymbol{v}^{(t)}. \quad (5)$$

Here, $\boldsymbol{r} \in \mathbb{R}^{4d}$ represents inferential relation between head and tail events, $[\cdot; \cdot]$ denotes vector concatenation, and "$\times$" denotes element-wise product.

Lastly, the relation representation, $\boldsymbol{r}$, is learned by passing it into a neural classifier to predict the oracle relation in the original triple. In order to enable this knowledge model to represent a null or non-associated relation between events, we define an extra relation category, named dummy relation $r^{(dmy)}$. This classification is written as

$$\boldsymbol{p}^{(rc)} = \text{softmax}(\text{MLP}(\boldsymbol{r}; \theta^{(rc)})) \in \mathbb{R}^{|\mathcal{R}'|}, \quad (6) \\ \mathcal{R}' = \mathcal{R} \cup \{r^{(dmy)}\},$$

where $\boldsymbol{p}^{(rc)}$ is the probability distribution over $\mathcal{R}'$, and $\mathcal{R}'$ denotes a union of the well-defined relation set $\mathcal{R}$ with a dummy relation category $r^{(dmy)}$. The training data corresponding to $r^{(dmy)}$ is derived from negative sampling in the eventuality KG.

Training. We use a cross-entropy loss to optimize this discriminative knowledge model, $\{\theta^{(km)}, \theta^{(rc)}\}$, towards such a dummy-aware relation classification, which is denoted as

$$\mathcal{L}^{(rc)} = -\sum\nolimits_{(e^{(h)},r,e^{(t)})} \log \boldsymbol{p}^{(rc)}_{[y=r]}. \tag{7}$$

Inference. The trained knowledge model can be used in three ways summarized as (1) producing event representation by

$$\begin{aligned}\boldsymbol{v} &:= \text{Event-Enc}(e; \theta^{(km)}) \\ &= \text{Pool}(\text{Transformer}(\tilde{\boldsymbol{w}}^e; \theta^{(km)})),\end{aligned} \tag{8}$$

(2) generating relation representation by

$$\boldsymbol{r} := \text{Relation-Model}(e^{(h)}, e^{(t)}; \theta^{(km)}), \tag{9}$$

and (3) deriving a confidence score for whether there is an associated relation between two events:

$$\begin{aligned}p^{(ca)} &:= \text{Confid}(e^{(h)}, e^{(t)}; \theta^{(km)}, \theta^{(rc)}) \\ &= \sum\nolimits_{r \in \mathcal{R}' \backslash \{r^{(dmy)}\}} \boldsymbol{p}^{(rc)}_{[y=r]},\end{aligned} \tag{10}$$

Remark. This discriminative knowledge model learns inferential relations between events in latent space, facilitating event-centric reasoning tasks. But it has its drawbacks like incompetence to auto-construction, in contrast to the generative knowledge models. Thereby, we argue generative and discriminative knowledge models are complementary to each other with different downstream uses.

3.2 Script-Adaptive Knowledge Model

In multi-choice narrative cloze (MCNC), a script reasoning model is asked to capture the relation between an event chain and a subsequent event candidate, however beyond the ability of the proposed knowledge model. To handle the MCNC task, we propose a neural adapter for the event encoder in Eq.(8), making it competent in modeling an event chain. Our goal is that, given a subsequent candidate, we extract the most relevant "event" from an event chain to represent the whole chain. As such, the result is still compatible with high-layer components in our knowledge model.

To this end, we present a chain-dependent attention module which is based on bidirectional chain contexts $\boldsymbol{e} = [e_1, \ldots, e_n]$ queried by a potential subsequent event $e^{(c)}$. In particular, we first generate event representation for each event by our trained event encoder, i.e.,

$$\boldsymbol{v}^{(c)} = \text{Event-Enc}(e^{(c)}; \theta^{(km)}), \tag{11}$$

$$\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{d \times n}, \tag{12}$$

$$\text{where, } \boldsymbol{v}_i = \text{Event-Enc}(e_i; \theta^{(km)}),$$

Then the embedded event chain, $\boldsymbol{V}$, position-wisely concatenated with the query event representation $\boldsymbol{v}^{(c)}$, is passed into a bidirectional long short-term memory (Bi-LSTM) to model rich event-contextual information of the event chain, i.e.,

$$\bar{\boldsymbol{U}} = \text{Bi-LSTM}(\bar{\boldsymbol{V}}; \theta^{(bl)}) \in \mathbb{R}^{4d \times n}, \tag{13}$$

$$\boldsymbol{U} = \text{MLP}(\bar{\boldsymbol{U}}; \theta^{(rd)}) \in \mathbb{R}^{d \times n}, \tag{14}$$

$$\text{where, } \bar{\boldsymbol{V}} = [\bar{\boldsymbol{v}}_1, \ldots, \bar{\boldsymbol{v}}_n], \bar{\boldsymbol{v}}_i = [\boldsymbol{v}_i; \boldsymbol{v}^{(c)}] \in \mathbb{R}^{2d}.$$

The resulting $\boldsymbol{U} \in \mathbb{R}^{d \times n}$ is chain-dependent representations of the chain events, $\text{MLP}(\cdot; \theta^{(rd)})$ is responsible for reducing dimensionality.

Lastly, a self-attention pooling module (Liu et al., 2016; Lin et al., 2017) is applied to $\boldsymbol{U}$ to get a vector representation of the event chain, i.e.,

$$\boldsymbol{c} = \boldsymbol{U}\boldsymbol{\alpha}, \tag{15}$$

$$\text{where } \boldsymbol{\alpha} = \text{softmax}(\text{MLP}(\boldsymbol{U}; \theta^{(sa)})).$$

Here, $\boldsymbol{\alpha} \in \mathbb{R}^n$ denotes the probability distribution of attention mechanism, which is then applied to chain-dependent event representations $\boldsymbol{U} \in \mathbb{R}^{d \times n}$ by matrix multiplication. As a result, $\boldsymbol{c}$ denotes a chain-dependent event representation extracted from the whole event chain. Intuitively, it can be viewed as the most relevant event from the event chain $\boldsymbol{e}$ to the candidate event $e^{(c)}$ as it is derived from an attention module queried by $e^{(c)}$. Hence, the derived $\boldsymbol{c}$ is still compatible with the top layers (e.g., interactive concatenation and neural classifier) in the discriminative knowledge model. Note that, the parameters of this neural script adapter, $\theta^{(adp)} = \{\theta^{(bl)}, \theta^{(rd)}, \theta^{(sa)}\}$, will be learned towards the MCNC objective jointly with other neural components in our script reasoning model, which is detailed in the next section (§3.3).

In summary, we can define a chain-dependent event encoding module to the above procedures to embed an event chain, i.e.,

$$\boldsymbol{c} = \text{Event-Enc}^{(\text{adp})}(\boldsymbol{e}, e; \theta^{(km)}, \theta^{(adp)}), \tag{16}$$

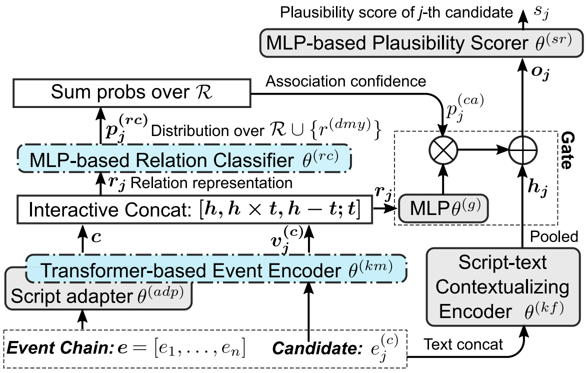

Figure 3: Our script reasoning model for MCNC. Freezing the trained modules (dash-dot blue rounded rectangles) in our knowledge model, we optimize the other learnable modules (solid gray rounded rectangles) in our script reasoning model towards the MCNC task. Please refer to Figure 2 for an illustration of the script adapter.

where $\boldsymbol{e} = [e_1, \ldots]$ is an event chain and e is a query event. The chain-dependent event representation, $\boldsymbol{c}$, can be used as an argument to $\text{Inter-Concat}(\cdot, \cdot)$ to model script-level relationship with another event chain or a single event. Thus, the other two inference models in Eq.(9) and Eq.(10) are also adapted to $\text{Relation-Model}^{(\text{adp})}(\cdot)$ and $\text{Confid}^{(\text{adp})}(\cdot)$ respectively.

3.3 Script Reasoning Model

Built upon the discriminative knowledge model and its script adapter, we lastly present our script learning model for multi-choice narrative cloze task. To be specific, given an event chain $\boldsymbol{e} = [e_1, \ldots, e_n]$ and each event $e_j^{(c)}$ from the subsequent candidates $\mathcal{E}^{(c)}$, we first pass them into the script-adaptive knowledge model to generate a chain-dependent event representation $\boldsymbol{c}_j$ as defined Eq.(16):

$$\boldsymbol{c}_j = \text{Event-Enc}^{(\text{adp})}(\boldsymbol{e}, e_j^{(c)}; \theta^{(km)}, \theta^{(adp)}),$$

where $\forall e_j^{(c)} \in \mathcal{E}^{(c)}$. Based on $\boldsymbol{c}_j$, we can also derive the relation representation $\boldsymbol{r}_j$ between the event chain and each candidate, as well as the confidence score p_j of the association.

$$\boldsymbol{r}_j = \text{Relation-Model}^{(\text{adp})}(\boldsymbol{e}, e_j^{(c)}; \theta^{(km)}, \theta^{(adp)}),$$
$$p_j^{(ca)} = \text{Confid}^{(\text{adp})}(\boldsymbol{e}, e_j^{(c)}; \theta^{(km)}, \theta^{(rc)}, \theta^{(adp)}).$$

Besides the above rich-relation features from the knowledge model, we also leverage expressively powerful contextualized representations from another pre-trained bidirectional Transformer to fully exploit implicit reasoning knowledge in event texts. Formally, we present a *script-text contextualizing encoder* that applies the Transformer encoder to a concatenation of the event chain and each subsequent candidates, with special tokens separated:

$$\tilde{\boldsymbol{w}}_j = (\texttt{[CLS]}, \boldsymbol{w}^{e_1}, \ldots, \texttt{[SEP]}, \boldsymbol{w}^{e_j^{(c)}}, \texttt{[SEP]}),$$
$$\boldsymbol{h}_j = \text{Pool}(\text{Transformer}(\tilde{\boldsymbol{w}}_j; \theta^{(kf)})). \quad (17)$$

To integrate the knowledge from the both models, we present a *knowledge gating module* for element-wise addition weighted by the association confidence:

$$\boldsymbol{o}_j = \boldsymbol{h}_j + p_j^{(ca)} \cdot \text{MLP}(\boldsymbol{r}_j; \theta^{(g)}), \quad (18)$$

where, $\boldsymbol{o}_j \in \mathbb{R}^d$ is the final vector to represent the relation between the chain and a candidate from two perspectives, and $\text{MLP}(\cdot; \theta^{(g)})$ is responsible for reducing dimensionality from $4d$ to d. Such a gating module leads to a flexible knowledge integration, which is prone to avoiding redundant, non-associated relation features.

Finally, an MLP-based scoring module is defined to calculate a plausibility score given the final relation representation, followed by a softmax to derive predicted distribution:

$$s_j = \text{MLP}(\boldsymbol{o}_j; \theta^{(sr)}), \forall j = 1, \ldots, m, \quad (19)$$
$$\boldsymbol{p}^{(sr)} = \text{softmax}([s_1; \ldots; s_m]) \in \mathbb{R}^m, \quad (20)$$

where $m = |\mathcal{E}^{(c)}|$, and $\boldsymbol{p}^{(sr)}$ is the predicted distribution over candidate events $\mathcal{E}^{(c)}$ in MCNC.

Training. With fixed knowledge model $\{\theta^{(km)}, \theta^{(rc)}\}$, we train both the adapter $\theta^{(adp)}$ and the reasoning model $\theta^{(src)} = \{\theta^{(kf)}, \theta^{(g)}, \theta^{(sr)}\}$ towards the objective of MCNC, by a cross-entropy loss, i.e.,

$$\mathcal{L}^{(sr)} = -\frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \log \boldsymbol{p}^{(sr)}_{[y=e^{(c)*}]}, \quad (21)$$

where $e^{(c)*}$ denotes the gold subsequent event.

Inference. We can obtain the most plausible subsequent event from a trained MCNC model by

$$\hat{e}^{(c)} = \arg\max_{e_j^{(c)}} [s_1; \ldots; s_m]. \quad (22)$$

4 Experiments

This section begins with a detailed description of our experimental setups on multi-choice narrative cloze (MCNC) task for script reasoning. Then, we conduct quantitative evaluations on the MCNC task, followed by extensive qualitative evaluations, including ablation study, model analysis, case study and error analysis.

Method	ACC (%)
w/o external knowledge	
Random	20.00
PMI (Chambers and Jurafsky, 2008)	30.52
Bigram (Jans et al., 2012)	29.67
Word2vec (Le and Mikolov, 2014)	42.23
Event-Comp (Granroth-Wilding and Clark, 2016)	49.57
PairLSTM (Wang et al., 2017)	50.83
SGNN (Li et al., 2018)	52.45
RoBERTa$_{\text{base}}$ (Lv et al., 2020b)	56.23
w/ external knowledge	
SGNN + Int&Senti (Ding et al., 2019)	56.03
RoBERTa$_{\text{base}}$ + Knwl. (Lv et al., 2020b)	58.66
Ours + RoBERTa$_{\text{base}}$	**59.99**
Ours + RoBERTa$_{\text{large}}$	**63.62**

Table 1: Comparison of our approach with previous script reasoning models on MCNC task. Our two models achieve 59.96% and 63.95% on dev set, respectively.

Datasets and Knowledge Graph. Following prior works (Lv et al., 2020b) for script reasoning, we evaluate our proposed approach on the dataset published by Li et al. (2018), which is widely used for the MCNC task. We follow the official data split[1] with 140,331/10,000/10,000 samples in training/dev/test sets. We use ASER (Zhang et al., 2020) as an external knowledge graph and learn a knowledge model from it. ASER is a large-scale eventuality knowledge graph extracted from unstructured textual data. It contains 15 event relations, 194M unique events, and 64M event-centric triples.

Evaluation Metrics. We adopt the official evaluation metric (Li et al., 2018), accuracy (ACC), to measure the performance of the reasoning models.

Implementation Details. The proposed approach for script reasoning contains two training processes, one for the discriminative knowledge model pre-trained on the eventuality KG and the other for the script reasoning model for MCNC task. (1) For the knowledge model, we adopt the BERT$_{\text{base}}$ model and optimize the cross-entropy loss with Adam optimizer. The learning rate is set to 1×10^{-5}. The hidden size of Bi-LSTM is set to 256. The maximum training epoch and batch size are set to 100 and 128. The maximum sequence length and dropout are set to 18 and 0.1. The weight decay and gradient clipping are set to 0.01 and 1.0. (2) For the script reasoning model, We experiment with two pre-trained language models, i.e. RoBERTa$_{\text{base}}$ and RoBERTa$_{\text{large}}$. Both the embedding size and hidden size are set to 768 in RoBERTa$_{\text{base}}$ and 1024 in RoBERTa$_{\text{large}}$. We use Adam optimizer (Kingma and Ba, 2015) to optimize the cross-entropy loss. The learning rate is set to 1×10^{-5}. The maximum training epoch and batch size are set to 3 and 32. The maximum sequence length and dropout are set to 64 and 0.1. The weight decay and gradient clipping are set to 0.01 and 1.0. We choose the model with the best result on the development set and report the results on the testing set are based on this model. The knowledge model contains 110M parameters, and our reasoning models contain 127M and 359M parameters for the base and large initializations, respectively. Our experiments are conducted on 4 NVIDIA P40 GPUs, and the training time is around 5 hours with RoBERTa$_{\text{base}}$ and 9 hours with RoBERTa$_{\text{large}}$.

Method	ACC (%)
Ours + RoBERTa$_{\text{large}}$	**63.62**
w/o chain-dependent attention	62.62
w/o knowledge gating module	62.85
w/o script-adapter	62.24
w/o external knowledge	61.53

Table 2: Ablation study of our approach. "*w/o chain-dependent attention*" denotes replacing chain-dependent attention module in our script adapter with mean-pooling, "*w/o knowledge gating module*" denotes removing confidence score $p^{(ca)}$ of the gating module in Eq.(18), ''*w/o script-adapter*" denotes ablating both chain-dependent attention and knowledge gating module, and "*w/o external knowledge*" denotes removing our script-adaptive knowledge model, equivalent to the RoBERTa$_{large}$ baseline.

4.1 Main Evaluation

The experimental results of our approach and previous script reasoning works on the Multi-Choice Narrative Cloze (MCNC) task are shown in Table 1. From the table, we can make two observations. First, using external knowledge, especially external event graph knowledge, increases the accuracy of models. For example, the knowledge infusion approach proposed by Lv et al. (2020b) outperforms the RoBERTa model without any knowledge. Second, our approach is superior to the *retrieval and integration* approach, RoBERTa + Knwl, and achieves new state-of-the-art accuracy (i.e. 63.62% using the RoBERTa$_{\text{large}}$ text encoder) on this task. This demonstrates the effectiveness of the proposed script-adaptive knowledge model.

4.2 Ablation Study

We conduct an ablation study to investigate the effect of each component of our approach and the results are reported in Table 2. We first investigate the impact of the chain-dependent attention

---

[1] https://github.com/eecrazy/ConstructingNEEG_IJCAI_2018

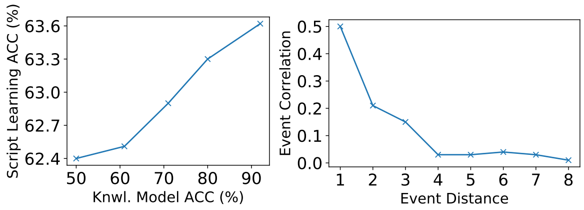

Figure 4: Impacts of knowledge model accuracy (left) and event distance over event association (right).

in Eq.(15) by replacing it with mean pooling over all events in the chain, and find that the accuracy of script reasoning is decreased about 1%. Next, we testify our approach without the knowledge gating module, which decreases the accuracy by 0.8%. And the gap becomes 1.4% if both the chain-dependent attention and the confidence score is ablated. Finally, we compare our approach with the baseline without any external knowledge included, and the accuracy of script reasoning drops by 2.1%, demonstrating the effectiveness of leveraging external event knowledge by the discriminative knowledge model.

4.3 Model Analysis

Impact of Knowledge Model. Our approach leverages the external event KG by directly modeling relations between event pairs. Intuitively, the accuracy of the learned model plays a critical role in script reasoning. Thus, we investigate its impact by assessing the performance of script reasoning with the knowledge models of various accuracy. The accuracy of knowledge model is evaluated on its dev set from the KG. As shown in Figure 4 (left), we can observe that the accuracy of script reasoning increases with that of the knowledge model, verifying the assumption that integrating external event knowledge via a knowledge model improves the performance of a script reasoning model.

Impact of Event Distance over Event Association. The script reasoning task requires to predict a subsequent event given an event chain. We investigate how the distance between two events impacts their correlation by analyzing the attention score of various timesteps of precedent events in a script. In particular, for all precedent events which are i-th step before subsequent events, we aggregate their attention scores. The results are plot in Figure 4 (right). The x-axis represents the distance between two events, and y-axis represents the estimated correlation by our model. From the figure we can see that an event is most highly correlated to its precedent neighbor. The association drops quickly as the distance increases, and it becomes very small when two events are three steps away.

Precedent Events
...
E7: state granted accountant
E8: believed accountant
Subsequent Event Candidates
A. asked accountant (*correct answer*)
B. ends accountant
C. depict accountant
D. penalize accountant firm
E. need accountant salary
Event pairs in KG with similar events
<Zakia will believe it, Zakia drops he guard>
<He is an accountant, He is very intelligent>
. . .
Event pairs in KG with Similar Relation Pattern
<I need your medical expertise, I need you help on something>
<You be the expert, I need answer>

Table 3: An example of script reasoning. The task is to choose a subsequent event from 5 candidates for the given precedent events. The associated event pair in the example is marked in blue with underline.

4.4 Case Study

As demonstrated in Table 3, we present an example in the test set to compare the *retrieval and integration* approach and ours. Here the script describes an event chain about *accountant*, which states that *accountants are believed*.

To infer the next event, the *retrieval and integration* paradigm will try to retrieve events with similar lexicon or semantics, e.g. "*zakia will believe it*", "*he is an accountant*", etc. However, KG triples containing these events do not capture the relation that *if a person is believed, people will consult him/her*. Therefore, this approach fails to leverage the KG to make a correct prediction.

In contrast, the KG contains event pairs like "*(I need your medical expertise, I need your help on something)*", "*(you are the expert, I need answer)*", whose relation patterns support the relation between "*accountants are believed*" and "*people ask accountants*", although there is little overlap between KG events and scrip events. The knowledge model learned from the KG event pairs captures such relation patterns and provides strong support for reasoning in this example, which demonstrates the effectiveness of our approach compared with the *retrieval and integration* approach.

4.5 Error Analysis

Lastly, to analyze the limitations of our model, we investigate the mis-classified examples on the MCNC test set, and summarize two main problems:

First, some scripts consist of precedent events which might lead to conflict results. For example, a event chain, [“*He disappointed supporters*”, “*He fulfilled promise*”], is likely to be associated with two opposite results. The former might be associate with “*He lost campaign*”, while the latter might result in “*They backed up his campaign*”. Such case might confuse the reasoning model.

Second, long-distance dependency between events are difficult to capture. For example, in a tourism script which describes “*Emily went to the beach*” followed by a long description about the parking problem she met, although “*Emily went surfing*” is a rational subsequent event, the distance between the two events is too long so it is difficult for a reasoning model to capture such relations.

5 Related Work

A script (Schank and Abelson, 2013) refers to a kind of structured representation for prototypical sequences of events. Chambers and Jurafsky (2008) formulate a script learning (narrative learning) task and propose statistical models to capture event co-occurrence for subsequent event prediction. Afterwards, the approaches for script reasoning can be categorized into two genres. i.e., *event pair modeling* (Jans et al., 2012; Pichotta and Mooney, 2014; Granroth-Wilding and Clark, 2016) and *event chain modeling* (Pichotta and Mooney, 2016; Wang et al., 2017; Lv et al., 2019). But, they still lag far behind humans as the well-labeled training set is usually of small size. In addition, script reasoning is more challenging than traditional NLP tasks and requires models to reason over unobserved events.

With recent developments of large-scale eventuality knowledge graphs (KG) (e.g., ASER (Zhang et al., 2020) and ATOMIC (Sap et al., 2019)), an effective remedy is to adopt “*retrieval and integration*” schema and integrate the inferential facts retrieved from the KG for script reasoning (Lv et al., 2020b). This paradigm is proven effective in both entity-centric and concept-centric tasks, such as relation extraction (Zhang et al., 2019), named entity recognition (Liu et al., 2020) and commonsense reasoning (Lin et al., 2019; Lv et al., 2020a), etc. However, this paradigm is not that compatible with event-centric script reasoning since script reasoning focuses more on the inferential relation between consecutive events in a script rather than the triple facts with exact event matching. What is worse, these eventuality KGs consisting of free-form event usually encounter low knowledge coverage or incompleteness problem (Zhang et al., 2020; Bosselut et al., 2019b), leading to problematic grounding from an event to the nodes in the KG.

To circumvent the coverage problem, Bosselut et al. (2019b) and Hwang et al. (2020) propose to learn a *generative knowledge model* on existing triples from an eventuality KG, where the triples can be regarded as a seed of knowledge. It on-demand generates subsequent events with a prompt of the observed event and an inferential relation, thus avoiding event grounding and satisfying coverage necessity for a broad spectrum of NLP tasks (Shwartz et al., 2020; Majumder et al., 2020; Paul and Frank, 2020; Ding et al., 2019; Ma et al., 2019). However, such generative knowledge models are not perfectly compatible when capturing inferential relations between events because they focus more on inferring tail events rather than the relations.

In contrast, our method avoids operating merely on the triples that have lexical or semantic overlap with the targeted script, while directly learn the inferential relation patterns on the whole KG. The learned knowledge model can simply capture the relation between events in a script in latent space, benefiting various event-centric reasoning tasks.

6 Conclusion

In this work, we explore a novel paradigm to integrate an external eventuality knowledge graph into a script reasoning model for multi-choice narrative cloze task. We first identify a major problem affecting the integration for script reasoning. That is, previous works merely retrieve the graph triples that have semantic overlap with the events in a script, but neglect that the triples with similar inferential relation patterns can contribute a lot. We hence propose a knowledge model that learns the patterns on the graph and then provides supportive rich-relation evidence for events in a script. We also present a script adapter to make the knowledge model compatible with script-level reasoning. Built upon these, we finally present a reasoning model and evaluate it on the targeted task. Experimental results demonstrate that, the proposed model delivers new state-of-the-art performance, followed by further analyses to provide comprehensive insights.

References

Antoine Bosselut, Ronan Le Bras, and Yejin Choi. 2019a. Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. *arXiv preprint arXiv:1911.03876*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019b. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642. The Association for Computational Linguistics.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 789–797. The Association for Computer Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li, and Junwen Duan. 2019. Event representation learning enhanced with external commonsense knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4893–4902. Association for Computational Linguistics.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2727–2733. AAAI Press.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2020. COMET-ATOMIC 2020: On symbolic and neural commonsense knowledge graphs. *CoRR*, abs/2010.05953.

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pages 336–344. The Association for Computer Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Quoc V. Le and Tomás Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4201–4207. ijcai.org.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2829–2839. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2901–2908. AAAI Press.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020a. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8449–8456. AAAI Press.

Shangwen Lv, Wanhui Qian, Longtao Huang, Jizhong Han, and Songlin Hu. 2019. Sam-net: Integrating event-level and chain-level attentions to predict what happens next. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6802–6809. AAAI Press.

Shangwen Lv, Fuqing Zhu, and Songlin Hu. 2020b. Integrating external event knowledge for script learning. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 306–315. International Committee on Computational Linguistics.

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. *CoRR*, abs/1910.14087.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian J. McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9194–9206. Association for Computational Linguistics.

Debjit Paul and Anette Frank. 2020. Social commonsense reasoning with multi-head knowledge attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2969–2980. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 220–229. The Association for Computer Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2800–2806. AAAI Press.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3027–3035. AAAI Press.

Roger C Schank and Robert P Abelson. 2013. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4615–4629. Association for Computational Linguistics.

Reid Swanson and Andrew S. Gordon. 2008. Say anything: A massively collaborative open domain story writing companion. In *Interactive Storytelling, First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany, November 26-29, 2008, Proceedings*, volume 5334 of *Lecture Notes in Computer Science*, pages 32–40. Springer.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 57–67. Association for Computational Linguistics.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020. ASER: A large-scale eventuality knowledge graph. In *WWW '20:*

The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020, pages 201–211. ACM / IW3C2.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics.

PROST🍻:
Physical Reasoning about Objects through Space and Time

Stéphane Aroca-Ouellette[*], **Cory Paik, Alessandro Roncone** and **Katharina Kann**
University of Colorado Boulder
`firstname.lastname@colorado.edu`

Abstract

We present a new probing dataset named *PROST: Physical Reasoning about Objects Through Space and Time*. This dataset contains 18,736 multiple-choice questions made from 14 manually curated templates, covering 10 physical reasoning concepts. All questions are designed to probe both causal and masked language models in a zero-shot setting. We conduct an extensive analysis which demonstrates that state-of-the-art pretrained models are inadequate at physical reasoning: they are influenced by the order in which answer options are presented to them, they struggle when the superlative in a question is inverted (e.g., *most* ↔ *least*), and increasing the amount of pretraining data and parameters only yields minimal improvements. These results provide support for the hypothesis that current pretrained models' ability to reason about physical interactions is inherently limited by a lack of real world experience. By highlighting these limitations, we hope to motivate the development of models with a human-like understanding of the physical world.

1 Introduction

In the context of natural language processing (NLP), Bender and Koller (2020) provides a working definition of "understanding" as the ability to recover the communicative intent from an utterance. To achieve this, one must be able to query a set of concepts that is aligned with the speaker's own understanding. An example of such alignment is our interaction with the physical world. This experience, shared by all humans, provides a common set of concepts to rely on in communication. For example, the reader can map the phrase *I dropped my pint glass* to a set of relevant experiences and generate a mental depiction of the scene. Further, the reader can also use their knowledge of gravity and the properties of a pint glass to reason about potential outcomes: the pint glass will fall toward the ground and will likely break on impact.

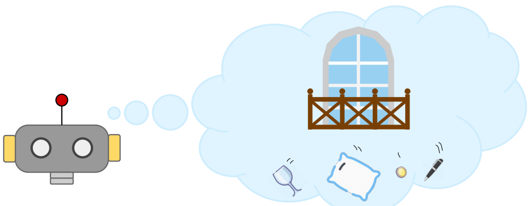

A person drops a *glass*, a *pillow*, a *coin*, and a *pen* from a balcony.
The [MASK] is most likely to break.
A) glass **B)** pillow **C)** coin **D)** pen

Figure 1: An example question from PROST.

Children grab, push, and play with the objects around them to form concepts about the world they live in even before learning to talk (Hespos and Spelke, 2004). These concepts are then linked with words to enable communication, eventually providing the necessary grounds for concepts and language to co-develop (Bloom, 2002; Gelman, 2009). In contrast, current language models (LMs) are not exposed to real-world experiences, making them incapable of grounding language (Bisk et al., 2020a). We hypothesize that this lack of experience impedes their ability to both understand an utterance relating to the physical world and their ability to reason about its implications.

In order to investigate our hypothesis, we create *PROST: Physical Reasoning of Objects Through Space and Time*, a probing dataset to evaluate the ability of pretrained LMs to understand and reason about the physical world. PROST consists of multiple-choice cloze-style questions covering 10 basic concepts: direction, mass, height, circumference, stackable, rollable, graspable, breakable, slideable, and bounceable. Importantly, PROST is designed to avoid models succeeding in unintended

---

[*] Email has no accent, but includes the hyphen

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4597–4608
August 1–6, 2021. ©2021 Association for Computational Linguistics

ways. First, PROST provides no training data, so as to probe models in a zero-shot fashion. This prevents models from succeeding through spurious correlations between training and test data and encourages success through a true understanding of and reasoning about the concepts at hand. Second, we manually write templates for all questions in an effort to prevent models from having seen the exact same sentences in their training data. Finally, it focuses on a small set of well defined, objective concepts that only require a small vocabulary. This allows researchers to focus more on the quality of training data rather than the size of it.

Contributions We make two contributions: 1) We introduce PROST, a dataset with $18,736$ cloze-style questions created from 14 manually written templates, covering 10 physical reasoning tasks. 2) We conduct an extensive analysis which demonstrates that state-of-the-art pretrained models are inadequate at physical reasoning. More specifically, they are influenced by the order in which answer options are presented to them, they struggle when the superlative in a question is inverted (e.g., *most* $\leftrightarrow$ *least*), and increasing the amount of pretraining data and parameters only yields minimal improvements. The dataset and code is available at github.com/nala-cub/prost.

2 Related Work

Evaluation of Reasoning Abilities As pretrained models are excelling on many NLP tasks, more work is being done on understanding their abilities. A subset of this work focuses on physical reasoning. PIQA (Bisk et al., 2020b) tests physical commonsense, with concepts ranging from hard shell tacos to separating egg yolks. In order to succeed on PIQA through reasoning, a model would need to be able to understand thousands of human experiences. In contrast, PROST provides a first step towards grounded understanding and reasoning by focusing on a few simple concepts. Bakhtin et al. (2019) provides a set of 2D puzzles that involve placing a new object in a scene to accomplish a goal. This research also focuses on simple physics, however there is no language component. Clark et al. (2018) and Kembhavi et al. (2017) both provide a large set of grade school multiple-choice questions, including some that could be solved with reasoning. However both provide corresponding material where the solution can be found, relying more on information retrieval than a general understanding and reasoning about the world.

Another set of reasoning-based benchmarks focuses on common sense reasoning. SWAG and its extension hellaSWAG evaluate commonsense natural language inference (Zellers et al., 2018, 2019). Sap et al. (2019) tests commonsense reasoning about social situations. However, commonsense reasoning is often subjective and requires understanding of complex human–human interactions involving social and societal norms. In contrast, physical reasoning is based on objective and well defined constructs.

Other datasets (Forbes and Choi, 2017; Elazar et al., 2019; Goel et al., 2019) focus on object–attribute comparison. However, they compare concepts at a word level rather than sentence level and use a large training set to create an engineered object–attribute comparison model. It is difficult to see how these models could generalize to other forms of reasoning.

Moreover, all the above datasets follow a pretraining-agnostic identically distributed (PAID) paradigm (Linzen, 2020), making them susceptible to models that can leverage unintended correlations between the training and test sets.

Zero-Shot LM Probes Similar to PROST, several recent benchmarks have circumvented the concern of identically distributed training and test sets by probing models in a zero-shot manner. Petroni et al. (2019) queries masked LMs (MLMs) for factual knowledge using templates in the format of *Dante was born in [MASK].* Talmor et al. (2020) use a similar format to probe six concepts ranging from age comparison to taxonomy conjunction. Ettinger (2020) uses this format to show that BERT robustly retrieves hypernyms, but fails to understand negation. Lin et al. (2020) probe numerical commensense in both MLMs and traditional LMs. Warstadt et al. (2020) measures traditional LMs' sense of grammatical acceptability by comparing sentence probabilities.

Grounded Language Environments PROST investigates if pretrained models show a lack of understanding of the physical world which could result from learning language without grounding. While not used for pretraining, a number of multi-modal environments have been developed to ground language. Shridhar et al. (2020)'s ALFRED builds on other vision-and-language navigation environments (Gordon et al., 2018; Regneri et al., 2013; Zhu et al., 2017; Anderson et al., 2018), and enables grounding of language instruction to actions, behaviours, and objects. BABYAI (Chevalier-Boisvert et al., 2019) and BABYAI++ (Cao et al., 2020) provide an environment to ground simple language in a gridworld. Additionally, other work has explored grounding language in simulations or the real world (Hill et al., 2020; Lynch and Sermanet, 2020). While they provide important resources to ground language, little emphasis is placed on the language modules themselves. They are often trained tabulae rasae, learning language for a singular purpose and missing out on the syntax and coverage learnt during pretraining;[1] language is only ever an input, and no analysis has been done on how language understanding evolves as the agent learns to succeed on different tasks.

3 PROST

PROST consists of $18,736$ cloze-style multiple-choice questions designed for probing a LM's physical reasoning ability. They cover 10 basic concepts: direction, mass, height, circumference, stackable, rollable, graspable, breakable, slideable, and bounceable. We choose these concepts because they are well defined, easily learned by interacting with the world, and are useful concepts for any embodied agent. The questions are constructed from 14 manually written templates. Each template follows one of three different formats: the first format is specific to the set of questions pertaining to directions; the second format is used to gauge the relative attributes—specifically mass, height, and circumference—of objects; and the third format targets the affordances of objects—specifically whether an object is stackable, rollable, graspable, or breakable, and whether a surfaces is slideable or bounceable[2]. We use CheckList (Ribeiro et al., 2020) to obtain the questions from our templates. We show all templates in Table 1 and explain them in detail below. We end this section by describing the objects featured in PROST.

Direction Templates We use two templates to generate questions which probe understanding of direction. The first focuses on cardinal directions. The second uses a set of four manually crafted questions to probe understanding of how gravity affects the directions of a ball throughout its trajectory. Due to their similarity, we count these four questions as a single template. The direction templates create a total of 16 questions.

Attribute Templates The second set of templates probe the models' ability to reason about relative mass, height, and circumference of common objects. For each of these three concepts we create a set of six objects that are easily ordered by their respective attributes. A context is first presented with up to four of the six objects to prime the models with the range of possible choices. This is followed by a prompt that probes the model to select one of the objects based on the object's mass, height, or circumference. By inverting the superlative in the prompt (e.g., *longest* ↔ *shortest*), we can probe the model's ability to identify both the object with the highest attribute value and the object with the lowest attribute value from the set of choices. We permute through all objects and all orders. Each of the three attributes are tested using two templates that share the same set of objects. Each template produces ${}_6P_4 * 2 = 720$ questions, meaning each attribute is probed using 1440 questions.

Affordance templates The remaining templates target an understanding of object affordances. For each affordance—stackable, rollable, graspable, breakable, slideable, and bounceable— we collect a set of five objects with and five objects without that affordance. Again, we first provide a short context that contains each of the four possible objects. We then provide a prompt that requires the model to select the only object either with or without the affordance. We include all permutations of objects where there is exactly one correct answer. These templates produce ${}_5P_1 * {}_5P_3 * 4 * 2 = 2400$ questions for each of the six affordances.

Objects in PROST All possible values for the placeholders in our templates are shown in Table 3. For affordances, we display objects in two groups: those with and without each affordance. For attributes, objects are sorted by increasing order, e.g., for mass, *leaf* is the lightest object and *microwave* is the heaviest object. Each object in PROST is selected to be single-token compatible for a wide range of vocabularies to enable easy probing of MLMs. We validate the order of our attribute objects and the group membership for our affordance objects by collecting judgments from 9 human val-

---

[1] An exception is Lynch and Sermanet (2020), which incorporates modern LMs and provides impressive generalizability. However, they too only use language as an input and do not analyze how language understanding evolves.

[2] *Bounceable* here refers to providing an elastic collision.

Category	Qs.	Template
Directs. 1	12	**C:** A person is walking *{north/east/south/west}*. They turn *{left/right/around}*.
		Q: They are now walking [MASK].
		O: **A)** north **B)** east **C)** south **D)** west
Directs. 2a	1	**C:** A person drops a ball.
		Q: Immediately after leaving the person's hand, the ball is moving toward the [MASK].
Directs. 2b	1	**C:** A person throws a ball straight into the air.
		Q: Immediately after leaving the person's hand, the ball is moving toward the [MASK].
Directs. 2c	1	**C:** A person throws a ball straight into the air.
		Q: Immediately after reaching the highest point in it's trajectory, the ball is moving toward the [MASK].
Directs. 2d	1	**C:** A person drops a ball. The ball then bounces off the ground.
		Q: Immediately after bouncing off the ground, the ball is moving toward the [MASK].
		O: **A)** ground **B)** sky **C)** left **D)** right
Mass 1	720	**C:** A(n) *{mass_obj1}*, a(n) *{mass_obj2}*, a(n) *{mass_obj3}*, and a(n) *{mass_obj4}* moving at identical speeds each collide with a static hockey puck.
		Q: The puck hit by the [MASK] slides the *{shortest/longest}* distance.
Mass 2	720	**C:** A(n) *{mass_obj1}* and a(n) *{mass_obj2}* are placed on either end of a perfectly balanced seesaw.
		Q: The side of the seesaw with the [MASK] moves *{up/down}*.
		O: **A)** *{mass_obj1}* **B)** *{mass_obj2}* **C)** *{mass_obj3}* **D)** *{mass_obj4}*
Height 1	720	**C:** Four balls are dropped. The first is dropped from the height equivalent of a *{height_obj1}*, the second is dropped from the height equivalent of a *{height_obj2}*, the third is dropped from the height equivalent of a *{height_obj3}*, and the fourth is dropped from the height equivalent of a *{height_obj4}*.
		Q: The ball dropped from the height of the [MASK] takes the *{longest/shortest}* amount of time to fall.
Height 2	720	**C:** There are four staircases. The first staircase leads to the top of a *{height_obj1.}*, the second staircase leads to the top of a *{height_obj2.}*, the third staircase leads to the top of a *{height_obj3.}*, and the fourth staircase leads to the top of a *{height_obj4.}*.
		Q: The staircase leading to the top of the [MASK] is the easiest/hardest to walk up.
		O: **A)** *{height_obj1}* **B)** *{height_obj2}* **C)** *{height_obj3}* **D)** *{height_obj4}*
Circumf. 1	720	**C:** Four people are walking at identical speeds. The first walks around a *{circ_obj1}*, the second walks around a *{circ_obj2}*, the third walks around a *{circ_obj3}*, and the fourth walks around a *{circ_obj4}*.
		Q: The [MASK] takes the *{longest/shortest}* amount of time to walk around.
Circumf. 2	720	**C:** A person paints a circle around a *{circ_obj1}*, a *{circ_obj1}*, a *{circ_obj1}*, and a *{circ_obj1}*.
		Q: The circle around the [MASK] takes the *{most/least}* amount of paint.
		O: **A)** *{circ_obj1}* **B)** *{circ_obj2}* **C)** *{circ_obj3}* **D)** *{circ_obj4}*
Stackable	2400	**C:** A person is trying to stack *{stack}*, *{no stack1}*, *{no stack2}*, and *{no stack3}*.
		Q: The [MASK] are the *{easiest/hardest}* to stack.
		O: **A)** *{stack}* **B)** *{no stack1}* **C)** *{no stack2}* **D)** *{no stack3}*
Rollable	2400	**C:** A person is trying to roll a(n) *{roll}*, a(n) *{no roll1}*, a(n) *{no roll2}*, and a(n) *{no roll3}*.
		Q: The [MASK] is the *{easiest/hardest}* to roll.
		O: **A)** *{roll}* **B)** *{no roll1}* **C)** *{no roll2}* **D)** *{no roll3}*
Graspable	2400	**C:** A person is trying to move a pile of *{break}*, a pile of *{no break1}*, a pile of *{no break2}*, and a pile of *{no break3}* from one side of a room to the other using only one hand.
		Q: The [MASK] is the *{most/least}* likely to break.
		O: **A)** *{break}* **B)** *{no break1}* **C)** *{no break2}* **D)** *{no break3}*
Breakable	2400	**C:** A person drops a *{break}*, a *{no break1}*, a *{no break2}*, and a *{no break3}* from a balcony.
		Q: The [MASK] is the *{most/least}* likely to break.
		O: **A)** *{grasp}* **B)** *{no grasp1}* **C)** *{no grasp2}* **D)** *{no grasp3}*
Slideable	2400	**C:** A person is sliding four bricks across four hard surfaces. The first surface is covered with *{slide}*, the second surface is covered with *{no slide1}*, the third surface is covered with *{no slide2}*, and the fourth surface is covered with *{no slide3}*.
		Q: The surface covered with [MASK] is the *{hardest/easiest}* for the brick to slide across.
		O: **A)** *{slide}* **B)** *{no slide1}* **C)** *{no slide2}* **D)** *{no slide3}*
Bounceable	2400	**C:** A person is trying to bounce a rubber ball. They drop a first ball onto *{bounce}*, a second ball onto *{no bounce1}*, a third ball onto *{no bounce2}*, and a fourth ball onto *{no bounce3}*.
		Q: The ball dropped onto[MASK] bounces the *{most/fewest}* times.
		O: **A)** *{bounce}* **B)** *{no bounce1}* **C)** *{no bounce2}* **D)** *{no bounce3}*

Table 1: All templates in PROST. **C:** = Context, **Q:** = Question, **O:** = Options. {} indicate placeholders. The objects can be found in Table 3. The rest of the placeholders show their possibilities in the braces themselves. [MASK] indicates the position of the blank that the models need to fill. See Section 3 for more information. *NOTE: The number of objects with and without the affordances are swapped when the superlative is inverted.*

Model Type	Input	Target
Decoder	*(context)* They are now walking $\langle \mathbf{O} \rangle$.	Max probability for each sentence input to the model
Encoder	*(context)* They are now walking $\langle \mathbf{M} \rangle$.	Max probability for the masked token.
T5	*(context)* They are now walking $\langle \mathbf{X} \rangle$.	$\langle \mathbf{X} \rangle$ south $\langle \mathbf{Y} \rangle$
UnifiedQA	Which way are they walking now? `\n` (A) north (B) south (C) east (D) west `\n` *(context)*	south

Table 2: Overview of the task preprocessing for different architectures evaluated. In all methods, the context remains unchanged and is "A person is walking west. They turn left."

idators. The validators obtained a 100% agreement on the object ordering, and 94.6% agreement on the object group membership.

Attributes	
Attribute	*Objects*
mass	leaf, coin, egg, apple, brick, microwave
height	book, microwave, table, car, house, mountain
circ	book, microwave, table, car, house, mountain
Affordances	
Affordance	*Objects*
stack	books, blocks, boxes, coins, plates
no stack	balls, bottles, eggs, flowers, lamps
roll	apple, ball, bottle, egg, can
no roll	book, box, block, mirror, microwave
grasp	balls, blocks, books, bottles, flowers
no grasp	flour, rice, salt, snow, sugar
break	bottle, egg, glass, mirror, plate
no break	ball, coin, pen, pillow, shirt
slide	ice, frost, grease, oil, soap
no slide	carpet, concrete, grass, gravel, rubber
bounce	asphalt, brick, concrete, rubber, steel
no bounce	carpet, foam, grass, leaves, snow

Table 3: Objects used in the templates.

4 Models

Using PROST, we probe three types of transformer-based models (Vaswani et al., 2017): decoder models, encoder models, and encoder-decoder models. Each model has slightly different formatting requirements, which we show in Table 2. For each model type, we probe a range of different sizes to investigate the effects of scaling. We use Huggingface's (Wolf et al., 2020) pretrained models, see Table 4 for the full set.

Decoder Models We analyze OpenAI's GPT-1 (Radford et al., 2018) and GPT-2 (Radford et al., 2019). Both are based on a transformer decoder architecture and trained on a traditional language modeling objective. We run these models over each question four times, each time with a different choice replacing the [MASK] token. Following Warstadt et al. (2020), we select the sentence with the highest probability.

Model		**Params (M)**	**Data (GB)**
GPT		116.5	2
GPT-2	B	124.4	40
	M	354.8	40
	L	774.0	40
	XL	1557.6	40
BERT	B	110.1	13
	L	336.2	13
RoBERTa	B	124.7	160
	L	355.4	160
ALBERT V2	B	11.8	160
	L	17.8	160
	XL	59.0	160
	XXL	223.2	160
T5	S	60.5	170
	B	222.9	170
	L	737.7	170
	3B	2851.6	170

Table 4: Summary of models evaluated on PROST. We list the amount of pretraining data as the size of the uncompressed text corpus used.

Encoder Models We analyze BERT (uncased) (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and ALBERT (Lan et al., 2020), which are all based on transformer encoders. BERT is trained on MLM and next sentence prediction and uses static masking, RoBERTa is trained on MLM with dynamic masking, and ALBERT uses whole-word n-gram masking. For probing, we filter out all but the four answer choices from the output vocabulary and select the token with the highest probability as the model's decision.

Encoder-decoder Models We also include results for T5 (Raffel et al., 2020). T5 is trained using a span corruption objective, in which spans of the input sequence are randomly replaced with a single unique mask token. During pretraining, span lengths are chosen randomly with an average length of three. To keep our results consistent with the other models, we restrict the span length to one token. We find that two of the options for sliding surfaces, namely *ice* and *frost*, violate our single-token constraint. To avoid any unfair comparison between answers that differ in token lengths and following previous work (Goldberg, 2019), we chose to omit presenting the results for T5 on the sliding concept.

Finetuned Conditional LMs To better understand the limitations of text-only training, we additionally evaluate UnifiedQA (Khashabi et al., 2020). UnifiedQA is a pretrained QA model, built off T5, and finetuned on SQuad 1.1, SQuaD 2.0, NarrativeQA, RACE, ARC, OpenBookQA, MCTest, and BoolQ (Rajpurkar et al., 2016, 2018; Kočiský et al., 2018; Lai et al., 2017; Clark et al., 2018; Mihaylov et al., 2018; Richardson et al., 2013; Clark et al., 2019). We format all of our templates to fit their multiple-choice question answering format and use their provided scoring metrics to select the models' answers.[3]

5 Results

The per model and per concept results are shown in Table 5. For concepts with more than one template—direction, mass, height, and circumference—we average across templates to get the concept's score. We can see that, on average, ALBERT-V2-XL performs best, with an accuracy of 31.8%[4], and GPT-2 performs worst, with an accuracy of 23.6%. We note that random guessing would yield an accuracy of 25%. Furthermore, every model underperforms random guessing on at least one concept. Since PROST is trivially solvable for humans, this supports our hypothesis that pretrained models are unable to perform physical reasoning anywhere close to human performance.

Comparing across all concepts, we see that direction obtains the highest average accuracy with 46.8%. The second best accuracy is observed for the mass attribute with 36.5%. The concepts models struggle the most with are the slideable and bounceable affordances, both with an average accuracy of 19.9%.

6 Analysis

Object Order in Context For the concepts that use objects, all four choices are listed in each question's context. PROST contains all permutations with regards to their ordering. This enables us to directly look at the effect of the correct answer's position within the context on the models' accuracy. These results shown in Table 6.

We see that models have a strong tendency to select either the first or the last item seen in the context. The largest difference is found for T5, with an accuracy of 52.4% for objects at position 1 and an accuracy of only 1.9% for objects at position 3. We note that a proper understanding of the questions, as most humans would have, would be robust to the order in which the choices are presented. This further underlines that state-of-the-art models do not perform human-like physical reasoning.

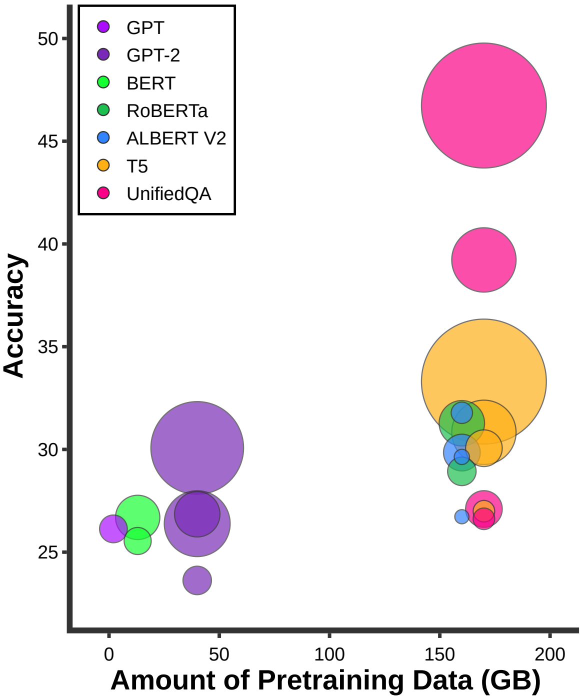

Figure 2: Scaling effect of models on accuracy. Circles size represents number of parameters.

Superlative Inverses By inverting the superlative in a question, we are able to probe a mirrored version of the question. For example, for attributes, this would require the model to identify the lightest object instead of the heaviest object, or, for affordances, it would require the model to identify the not stackable object instead of the stackable object. We call these mirrored versions *superlative inverses*. A true understanding of the questions in

---

[3] https://github.com/allenai/unifiedqa

[4] Note: as detailed in Section 4, T5 and UnifiedQA are not being evaluated on sliding. We therefore disregard their average accuracy.

Model		Direction	Mass	Height	Circum.	Stack	Roll	Grasp	Break	Slide	Bounce	Macro Average
GPT		46.7	40.1	24.3	22.8	28.2	27.9	19.6	22.7	14.6	14.4	26.1
GPT2		43.3	31.4	22.0	18.8	26.2	20.3	17.9	22.5	16.9	17.0	23.6
	M	48.3	34.1	21.6	21.6	25.5	23.7	24.9	27.8	22.5	18.5	26.8
	L	46.7	33.1	25.4	27.0	25.5	26.9	20.6	21.8	21.3	15.6	26.4
	XL	46.7	34.2	25.8	26.3	31.1	36.3	29.4	26.7	23.7	20.5	30.1
BERT	B	40.0	32.9	27.5	25.6	20.9	26.1	23.3	28.0	18.2	13.0	25.5
	L	**70.0**	38.8	19.4	17.5	21.3	19.2	26.7	19.5	15.9	18.6	26.7
RoBERTa	B	46.7	36.9	25.8	23.5	34.5	19.3	25.4	**45.0**	20.9	11.4	28.9
	L	66.7	**43.4**	33.8	22.7	22.7	22.2	29.4	23.8	22.7	25.5	31.3
ALBERT V2	B	21.7	35.4	30.2	26.0	25.2	32.5	35.3	22.8	15.3	22.9	26.7
	L	41.7	38.2	31.9	27.5	23.3	29.7	34.0	24.5	23.4	22.1	29.6
	XL	46.7	38.7	**42.0**	**37.4**	30.2	28.2	**37.1**	17.8	**25.3**	14.3	**31.8**
	XXL	68.3	33.8	28.1	24.5	29.4	23.4	21.2	30.2	17.5	22.1	29.8
T5	S	20.0	36.5	29.8	25.2	25.0	25.9	25.4	25.0	—	30.2	27.0*
	B	40.0	37.0	32.6	23.8	25.0	23.4	25.2	25.6	—	**37.8**	30.1*
	L	46.7	35.7	30.7	27.6	31.8	23.0	34.0	25.2	—	22.7	30.8*
	3B	46.7	39.6	35.6	29.8	**34.7**	31.5	35.6	33.8	—	12.5	33.3*
UnifiedQA	S	0.0	34.2	34.8	30.3	24.4	29.0	28.8	27.1	—	31.0	26.6*
	B	0.0	17.8	33.3	22.3	25.5	34.9	27.9	36.5	—	45.7	27.1*
	L	83.3	17.2	49.5	47.3	23.5	28.4	27.5	43.6	—	32.6	39.2*
	3B	63.3	37.8	55.2	66.9	31.2	35.2	24.8	81.4	—	24.8	46.7*
Task Average		46.3	36.5	28.6	25.2	27.1	25.9	27.4	26.0	19.9	19.9	28.5

Table 5: Macro average for each concept and overall for each model on PROST. The best accruacy for general pretrained-only models is displayed in bold. Note that the task average does not include UnifiedQA.

Model	Position Accuracy			
	1	2	3	4
GPT	27.0	24.3	7.6	38.6
GPT-2	29.9	23.1	8.1	42.0
BERT	28.4	24.3	5.7	38.2
RoBERTa	39.0	28.7	11.2	30.0
ALBERT V2	32.5	25.8	9.7	44.2
T5	52.4	21.1	1.9	35.2
UnifiedQA	41.0	27.7	18.8	51.9
Position Average	35.7	25.0	9.0	40.0

Table 6: Accuracy across the correct answer's position in the context.

PROST should be robust to this kind of inversion. However, Table 7 shows all models perform better on one of the two versions. Of the probed models, GPT-2 is the most unbalanced, averaging 30.6% higher for one version over the other.

Data and Model Scaling Figure 2 shows each model's accuracy as a function of the number of its parameters. Unlike for many modern benchmarks, where increasing the number of parameters or training data provides significant benefits (Talmor et al., 2020; Wang et al., 2018), PROST does not see much improvement from such scaling. We observe some improvements with T5-3B outperforming T5-small, but this 6.6% increase requires a 48x increase in parameters and T5-small still outperforms T5-3B on one task. Moreover, some models break this trend: ALBERT's XL version outpeforms its XXL counterpart and GPT-2 M outperforms GPT-2 L. While previous work has revealed the impressive scaling laws of transformer-based architectures (Kaplan et al., 2020), PROST highlights the importance of relevant and informative training. As physical reasoning is not an ability that humans acquire via text, even substantially more open domain textual data is unlikely to lead to more than marginal improvements.

The Limits of Text-based Training To our knowledge, UnifiedQA is the most qualified model to succeed on our task, having been finetuned on a significant amount of relevant text data. While this additional data does provide benefits on PROST, it still falls short, with the best performing model we tested only achieving a 46.7% accuracy. Additionally, from Tables 6 and 7, it still lacks the robustness of proper understanding. This emphasizes that models are unlikely to obtain human-like reasoning from text-based training alone. Rather, PROST motivates exposing models to concepts through multiple modalities that mirror a human's experience.

Model		Mass	Height	Circum.	Stack	Roll	Grasp	Break	Slide	Bounce	Macro Average
GPT		9.4	2.2	14.0	35.9	43.0	22.7	13.3	9.8	10.1	17.8
GPT-2		19.2	24.3	1.1	16.1	5.1	12.1	31.9	24.9	15.8	16.7
	M	31.2	12.9	21.5	12.7	20.2	7.8	49.9	33.3	9.2	22.1
	L	25.8	24.2	25.4	5.6	16.7	18.4	24.6	28.2	23.7	21.4
	XL	43.5	6.7	1.5	56.1	51.5	36.3	31.5	15.8	32.4	30.6
BERT	B	5.0	40.0	2.5	12.3	15.4	3.1	44.9	12.2	11.2	16.3
	L	19.2	21.5	5.8	1.8	3.4	4.2	17.2	9.5	30.4	12.6
RoBERTa	B	4.7	4.0	6.5	55.0	13.8	27.8	89.6	21.8	15.8	26.5
	L	31.0	24.7	26.8	9.7	21.1	33.2	31.5	33.9	33.2	27.2
ALBERT V2	B	4.7	31.0	7.9	14.8	14.4	66.4	**2.3**	**7.2**	1.5	16.7
	L	**0.6**	11.9	23.6	30.0	36.8	52.2	6.9	28.7	13.2	22.7
	XL	9.4	**0.7**	8.5	19.5	8.1	31.0	10.9	8.2	20.6	13.0
	XXL	18.1	2.9	18.5	4.2	12.2	2.0	37.2	16.3	11.3	13.6
T5	S	8.3	12.9	13.5	**0.0**	3.9	0.8	3.1	—	1.8	**5.5**
	B	8.7	26.9	5.8	0.0	**0.2**	**0.5**	3.1	—	26.2	8.9
	L	5.0	20.3	1.7	7.5	22.6	10.5	37.7	—	**0.5**	13.2
	3B	16.1	12.2	**0.7**	9.2	8.8	5.1	34.7	—	24.9	14.0
UnifiedQA	S	46.9	5.4	19.7	2.8	34.5	31.0	39.0	—	9.3	23.6
	B	19.2	19.2	4.3	24.6	30.4	10.4	4.9	—	49.5	20.3
	L	18.5	28.5	26.2	18.6	28.1	41.6	7.9	—	48.1	27.2
	3B	8.2	46.2	36.1	6.8	1.8	0.7	13.4	—	26.4	17.5
Task Average		15.3	16.4	10.9	17.1	17.5	19.7	27.7	19.2	16.6	17.6

Table 7: Absolute difference in accuracy between a question and its superlative inverse.

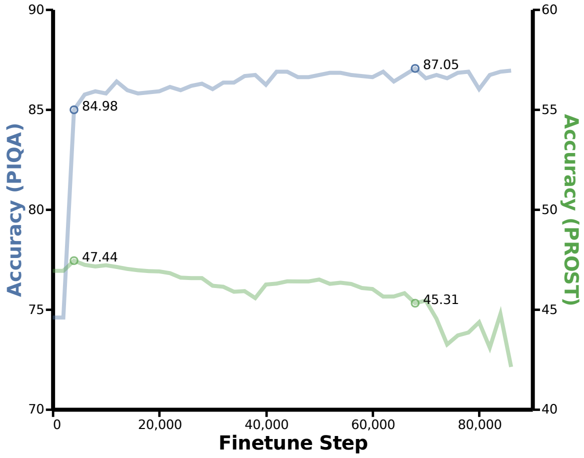

Figure 3: Analysis of the performance of UnifiedQA 3B on PROST throughout PIQA finetuning. The left and right Y axis represent Accuracy on the PIQA dev set and Macro accuracy on PROST respectively. We finetune for 100K steps, and compute metrics every 2k steps. Annotations correspond to the checkpoints with the best performance on PIQA and PROST. Note that PIQA has two answer choices, while PROST has 4.

Comparing PROST and PIQA Due to their shared focus on text-based physical reasoning, PROST and PIQA share similarities. To test if models trained on PIQA are able to carry over any concepts to PROST, we further finetune a UnifiedQA model on PIQA and evaluate it on PROST. The results, shown in Figure 3, indicate that training a model on PIQA is detrimental to its performance on PROST. While PIQA and PROST share a few conceptual similarities, they differ in terms of format, style, and vocabulary. We thus hypothesize that current models learn more about these surface-level differences than the conceptual similarities underpinning the questions. We further highlight two key differences between the two datasets:

- PROST probes models in a zero-shot fashion, whereas PIQA provides training and test sets of identically distributed examples. This makes it possible for models on PIQA to answer successfully using spurious correlations rather than physical reasoning.
- PIQA (Bisk et al., 2020b) covers an extensive range of objects and challenging physical concepts. Bisk et al. (2020a) argues that experience is a prerequisite for understanding. It is hard to imagine how to expose a model to experiences ranging from egg yolk separation to making a pillow out of a garbage bag. In contrast, PROST provides a clear set of well defined concepts and objects that a model could potentially experience.

7 Discussion

Our experiments show that all the models we analysed fail to demonstrate a robust understanding of physical reasoning. Beyond performing poorly across every concept, they are easily influenced by changing the order of the objects in a question's context and by superlative inverses. Moreover, our analysis indicates that these issues are not likely to be solved simply by increasing the amount of model parameters or training data. All this evidence supports Bender and Koller (2020)'s and Bisk et al. (2020a)'s theory that experience is a prerequisite of understanding.

A number of other reasoning benchmarks have been solved to some extent by a large finetuned model. UnifiedQA (11B parameters), based on T5 (Raffel et al., 2020), achieved 81.4% on ARC (Clark et al., 2018); and UNICORN[5] (11B parameters), also based on T5, achieved a 93.9% accuracy on hellaSWAG (Zellers et al., 2019). While all these models are larger and are trained on more data, our results force us to ask the question whether they perform well because these additional parameters and data have imbued the models with an ability to reason, or if they succeed by finding subtle unintended correlations in the data. This forces us to look more closely at how models succeed, and not just the accuracy they achieve. Tools like CheckList (Ribeiro et al., 2020) can aid in this endeavor by demonstrating how robust models are to changes in the distribution of the data.

How to Use this Probe PROST is intended to help analyze any model that can be deployed in a text-only setting. However, we maintain that multimodal data is necessary to experience the concepts in PROST, and that these experiences are likely a crucial step in succeeding on this dataset. One way that multi-modal models could prepare for this type of text-only evaluation is through multi-task training, where one of the tasks is only conditioned on text. Such an approach has already been considered: Brown et al. (2020) propose an extension to their CLIP model which is trained on multiple modalities in a multi-task fashion. Because of the templated nature of PROST, its exact format can be adapted to match specific styles of language training, as we do for T5 and UnifiedQA.

PROST's language-only approach is motivated by two reasons. First, we believe that true multimodal models should be able to function on any subset of their modalities. We note that humans can easily interact with text-only inputs (e.g., a text message) while still learning from and interacting with other modalities. Second, it enables the comparison of models trained using different modalities or domains. For example, we believe comparing how language understanding modules evolve when trained on vision-and-language navigation compared to visual question answering would provide invaluable insights.

Limitations We caution that achieving a high accuracy on PROST does not necessarily guarantee that a model is able of physical reasoning. It is likely easy to succeed on this benchmark if one were to intentionally train models on similar enough sentences or a subset of PROST itself. We hope that the community will use this dataset in the intended way: in a zero-shot setting to probe models which have been trained on data not specifically collected to succeed on PROST.

8 Conclusion

We present a probing dataset called PROST, which is designed to test a model's ability to reason about the physical world. Our experiments show that current state-of-the-art pretrained models lack the ability to reason about physical interactions. Further, all models struggle when the order of options is changed and when questions are inverted, both things that would not confuse humans. Lastly, our analysis shows that these issues are unlikely to be solved by simply scaling models. Our results highlight the need to look beyond text-based pretraining and to provide models with the necessary experiences for human-like understanding of the physical world.

Acknowledgments

We would like to thank the members of CU Boulder's NALA and HIRO Groups for their feedback on this work.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

---

[5] leaderboard.allenai.org/hellaswag/submissions/public

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. 2019. Phyre: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems*, volume 32, pages 5082–5093. Curran Associates, Inc.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020a. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020b. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Paul Bloom. 2002. *How children learn the meanings of words*. MIT press.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Tianshi Cao, Jingkang Wang, Yining Zhang, and Sivabalan Manivasagam. 2020. Babyai++ : Towards grounded-language learning beyond memorization. In *ICLR*.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yanai Elazar, Abhijit Mahabal, Deepak Ramachandran, Tania Bedrax-Weiss, and Dan Roth. 2019. How large are lions? inducing distributions over quantitative attributes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3973–3983, Florence, Italy. Association for Computational Linguistics.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 266–276, Vancouver, Canada. Association for Computational Linguistics.

Susan A. Gelman. 2009. Learning from others: Children's construction of concepts. *Annual Review of Psychology*, 60:115–140.

Pranav Goel, Shi Feng, and Jordan Boyd-Graber. 2019. How pre-trained word representations capture commonsense physical comparisons. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 130–135, Hong Kong, China. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: Visual question answering in interactive environments. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

Susan J Hespos and Elizabeth S Spelke. 2004. Conceptual precursors to language. *Nature*, 430(6998):453–456.

Felix Hill, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. 2020. Grounded language learning fast and slow.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868, Online. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Corey Lynch and Pierre Sermanet. 2020. Grounding language in play. *arXiv preprint arXiv:2005.07648*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi. 2017. Visual semantic planning using deep successor representations. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 483–492.

Revisiting the Evaluation of End-to-end Event Extraction

Shun Zheng[†], **Wei Cao**[†], **Wei Xu**[‡], **Jiang Bian**[†]
[†]Microsoft Research
[‡]Institute of Interdisciplinary Information Sciences, Tsinghua University
`{shun.zheng, wei.cao, jiang.bian}@microsoft.com;`
`weixu@mail.tsinghua.edu.cn`

Abstract

Event extraction (EE) aims to harvest event instances from plain text, where each instance is composed of a group of event arguments with specific event roles. Existing end-to-end EE research usually adopts the role-averaged evaluation that produces evaluation measures by averaging evaluation statistics of each event role. However, although this averaged metric can indicate the model performance to some extent, we find that such metric can be pretty misleading to downstream applications that utilize an event instance as a whole, where one wrongly identified event argument can substantially alter the whole meaning of an event instance. To mitigate this gap and provide a more complete understanding of performance, we propose two new evaluation metrics that also consider an event instance as a whole and explicitly penalize wrongly identified event arguments. Moreover, to support diverse preferences of evaluation metrics motivated by different scenarios, we propose a new training paradigm based on reinforcement learning for a typical end-to-end EE model, i.e., Doc2EDAG. Our extensive experiments show that the new training improves the initial one by a large margin (about 10%) under new metrics. Nevertheless, the current performance is still far from satisfactory, and optimizing towards these new metrics calls for more future research.

1 Introduction

Event extraction (EE) is a vital task that aims to harvest structured event instances from unstructured plain text. Such structured knowledge can benefit many downstream applications, such as question answering, language understanding, knowledge graph, etc. In general, an *event instance* is composed of a group of entities (person, organization, date, etc.) that jointly describes an incident. Each entity of the event instance, also referred to as the *event argument*, plays a specific *event role*. Multiple event instances of the same event type populate an *event table*.

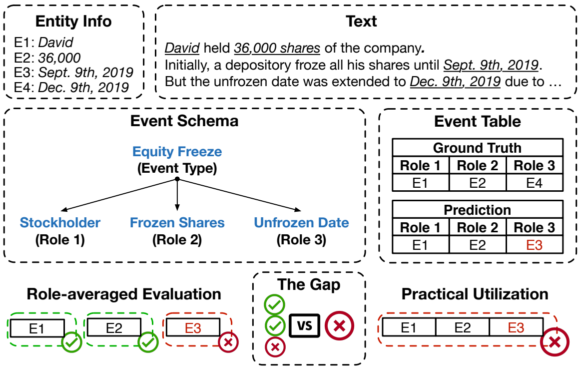

Figure 1: An example to illustrate the gap between the *role-averaged* evaluation and a practical application that utilizes an event instance as a whole, where "E1", "E2", "E3", and "E4" are entity marks, "Role 1", "Role 2", and "Role 3" represent event roles, and "(E1, E2, E4)" is an event instance. We can see that the wrong argument "E3" makes the whole event instance "(E1, E2, E3)" unrealistic, but the *role-averaged* evaluation still regards "E1" and "E2" as correct ones.

The early method (Ahn, 2006) formalized EE as the unification of many sub-tasks, including entity recognition, event detection, and argument extraction, etc. Later research improved EE from two aspects: the modeling to capture complicated semantic structures (Li et al., 2013; Yang and Mitchell, 2016; Nguyen and Nguyen, 2019) and the labeling to combat the lack of training data (Chen et al., 2017; Zeng et al., 2018). Recently, Zheng et al. (2019a) proposed the first end-to-end model, called Doc2EDAG, for document-level EE. Given a text document, Doc2EDAG can generate an entity-based directed acyclic graph (EDAG) to fill an event table directly.

Different from the modeling and labeling aspects, the evaluation of EE attracted very little re-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4609–4617
August 1–6, 2021.

search attention. Notably, most existing research, merely presenting a separate evaluation for each sub-task of EE, can only get the approximation of the overall performance by combining those fragmentary evaluation results. Even the latest work (Zheng et al., 2019a), reporting the overall performance of EE, still followed the traditional approximate measure by averaging evaluation statistics of each event role. We refer to this kind of approximation as the *role-averaged* evaluation. However, many downstream applications need to utilize an event instance as a whole, where a wrongly identified argument can substantially change the meaning of an event and cause severe misleading effects. Figure 1 presents an example from the financial domain, where the downstream application cannot utilize an event instance with an incorrect date argument for decision making, but the *role-averaged* evaluation still assigns this example with two true-positive arguments. Moreover, we note that this *event-as-a-whole* demand is a common case that widely exists in many other domains, such as legislation, health, etc.

To enable the evaluation support for the *event-as-a-whole* scenario and provide a more complete understanding of performance, we propose two new metrics that directly make judgments on an event instance rather than averaging the performance of its arguments. The first metric is *NoFP*, which regards a predicted event with any false-positive (FP) error at the entity level as an event-level FP error. The second one is *NoFPFN*, which permits neither FP errors nor false-negative (FN) errors at the entity level for a predicted event being considered as a true-positive (TP) one. In practice, we can choose to use the proper metric according to the specific scenario. For example, if we only care about the correctness of predicted events, we can utilize the *NoFP* evaluation that penalizes FP entities explicitly. If we further pursue the completeness, we can utilize the *NoFPFN* evaluation.

The necessity of employing new evaluation metrics, however, raises a dilemma in training effective EE models. On one side, since the *role-averaged* metric is inconsistent with *NoFP* or *NoFPFN*, training towards the *role-averaged*, as did by traditional methods, may not lead to improvements under new metrics. For instance, Figure 2 illustrates such inconsistence between different evaluation settings. We can observe that 1) the first prediction is the best one under the *role-averaged* evaluation but is sub-optimal for other settings, 2) the second prediction best fits the *NoFP* evaluation but suffers the most errors under the *NoFPFN* evaluation, and 3) the third prediction makes the most mistakes at the entity level but achieves the best performance under the hardest *NoFPFN* evaluation. In fact, existing methods for EE, including the most recent Doc2EDAG, merely optimized model parameters towards proper predictions for each event role separately. Thus, they failed to align with new metrics.

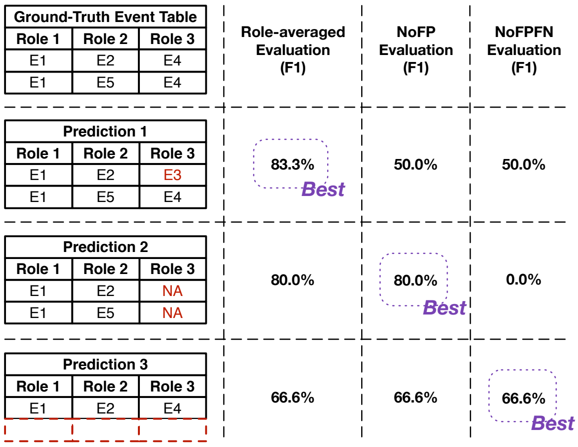

Figure 2: We present a ground-truth event table with two event instances ("(E1, E2, E4)" and "(E1, E5, E4)") and three different predictions to show diverse preferences of those evaluation settings, where all marks follow the meanings in Figure 1, "NA" denotes an empty argument, we color wrong or missed event arguments as red, and we mark the best prediction for each evaluation setting.

On the other side, optimizing these new metrics is non-trivial. First, it is hard to design appropriate training objectives since those metrics are non-differentiable. Moreover, the supervision for role-level predictions is inevitably delayed because we cannot calculate event-level metrics until obtaining predictions for all event roles. To address these challenges, we propose a new reinforcement learning (RL) paradigm of Doc2EDAG by regarding EDAG generation as a Markov decision process. In this way, we can optimize model parameters with the guidance of a delayed reward, which can be specified by a specific metric. At the same time, we realize that this shift of training also bridges the gap between training and inference because RL forces the model to generate EDAGs during training. Extensive experiments demonstrate that our RL-based training improves the vanilla one significantly (about 10%) under new metrics.

We summarize our contributions as follows.

- We revisit the evaluation of EE to support downstream applications that need to utilize an event instance as a whole. Specifically, we propose two new metrics to provide a more complete understanding of performance.

- We propose a new training paradigm for a typical end-to-end EE model, Doc2EDAG, to enable the flexible adaptation to new metrics with hard constraints.

- Our empirical studies show that under the *event-as-a-whole* scenarios, the traditional *role-averaged* evaluation tends to severely overestimate the performance. Moreover, the initial training scheme performs poorly on new metrics, while our RL-based training improves it significantly (about 10%).

2 Related Work

Since EE is a very sophisticated task that requires the unification of many sub-tasks (Ahn, 2006), including entity recognition, event detection, and argument extraction, plenty of previous research put considerable efforts to the modeling aspect.

Specifically, Nguyen and Grishman (2015); Liu et al. (2017); Chen et al. (2018); Wang et al. (2019); Liu et al. (2019) only considered event detection that is to detect trigger words and assign correct event types. Some advanced methods (Poon and Vanderwende, 2010; Riedel and McCallum, 2011; Li et al., 2013, 2014; Venugopal et al., 2014; Judea and Strube, 2016; Nguyen et al., 2016; Sha et al., 2018) tried to unify two sub-tasks, event detection and argument extraction, but all assumed that entity candidates were given in advance. A few studies attempted to fulfill all sub-tasks of EE jointly. Yang and Mitchell (2016) was the first work towards this goal but relied on handcrafted features. Later research (Nguyen and Nguyen, 2019) explored the joint modeling further by introducing neural networks but also retained many traditional lexical and syntactic features. Recently, Zheng et al. (2019a) formalized a new succinct task for EE without trigger words and proposed the first document-level end-to-end model, called Doc2EDAG. This novel model transformed the task of filling an event table into the generation of an EDAG and thus enabled the end-to-end modeling for EE.

With the rapid development of modeling techniques, labeled data gradually became the main bottleneck that prevented from putting EE into practice. Most of the previous research was based on ACE 2005[1], a benchmark dataset annotated by human experts. However, human annotation is both expensive and time-consuming. Recent research attempted to generate weakly labeled data by aligning event instances from knowledge bases to plain text and then assigning labels to matched samples. This strategy was originated from relation extraction (Mintz et al., 2009; Zheng et al., 2019b), but most existing EE models relied on trigger words to anchor an event mention. Accordingly, two types of research explorations emerged: one was to label trigger words with the help of extra linguistic resources (Chen et al., 2017) or predefined dictionaries (Yang et al., 2018), and the other was to remove the requirement of trigger words in modeling (Zeng et al., 2018; Liu et al., 2019; Zheng et al., 2019a). In this paper, we follow the no-trigger-words design because it can ease the labeling work of EE and thus generate large-scale data.

Different from all the related work, this paper focuses on the evaluation aspect and attempts to extend the scope of EE evaluation methods to better support the *event-as-a-whole* scenarios.

3 Preliminaries

In this section, we provide readers with necessary background to better understand our research.

3.1 Terminologies

We follow Yang and Mitchell (2016) to use a general "**entity**" notion that covers persons, dates, numbers, and so on for brevity. Next, let us recall the ground-truth event table in Figure 2 and clarify some widely used terminologies for EE. 1) An **event role**, corresponding to a column of an event table, is a basic semantic unit of an event type (e.g., "Role 1" is an event role). 2) An **event argument**, corresponding to an entry of an event table, refers to an entity that plays a specific event role (e.g., "E1" is an event argument). 3) An **event instance**, corresponding to a row of an event table, consists of a group of entities that jointly characterizes a specific incident (e.g., "[E1, E2, E4]" together forms an event instance).

3.2 Doc2EDAG

As discussed in the related work, Doc2EDAG enabled not only end-to-end modeling for EE but

---

[1] `https://www.ldc.upenn.edu/collaborations/past-projects/ace`

also simplified event labeling. Therefore, we adopt Doc2EDAG as the base end-to-end model and use its associated large-scale benchmark for validation.

Given a text document as the input, Doc2EDAG first extracts entities and encodes them with the document-level context. Then, for a triggered event type, Doc2EDAG generates an entity-based directed acyclic graph (EDAG) via a series of path-expanding sub-tasks. Each path-expanding sub-task is composed of a group of binary predictions (1: *expanding* or 0: *not*), where one prediction happens to one entity candidate.

The vanilla training of Doc2EDAG follows a given ground-truth EDAG and calculates corresponding losses for path-expanding sub-tasks. We refer to this kind of training as the maximum likelihood estimation (MLE) because its essential goal is to maximize the likelihood of ground-truth EDAGs. However, the vanilla MLE-based training cannot fit the newly proposed metrics with hard constraints.

4 New Evaluation Metrics

Similar to the initial *role-averaged* evaluation, the new evaluation starts from comparing a predicted event table with the ground-truth one. To be specific, they pick one predicted event instance and the most similar ground-truth one without replacement from corresponding event tables. The difference lies in the comparison of two picked event instances. Initially, the *role-averaged* evaluation collects evaluation statistics of TP, FP, and FN for each event role and averages them to calculate precision, recall, and F_1 scores for that event type. In contrast, the new evaluation considers an event instance as a whole and collects those statistics directly at the event level. Next, we illustrate the details of two new metrics on collecting TP, FP, and FN statistics when taking into consideration the *event-as-a-whole* requirement.

The NoFP Metric. An event instance that contains any FP argument belongs to an FP error at the event level. For an event instance that contains both FN arguments and TP arguments, we retain the statistics of FN and TP in proportion at the event level. For an event whose arguments are all correct, we count it as a TP event.

The NoFPFN Metric. We treat an event instance that contains either FP arguments or FN arguments as an FP error at the event level. Only for those event instances that are the same to the ground-truth ones, we treat them as TP events.

Note that the *NoFP* metric can be beneficial to risk-sensitive scenarios, such as finance, health, etc., where FP arguments of an event instance may cause disastrous effects. Moreover, the *NoFPFN* metric, also the most challenging one, can tell us how large the gap between the current progress and the perfect extractor is. In contrast, the initial *role-averaged* evaluation can only provide the approximate measures by averaging the performance of event role predictions, which only fit for the cases when downstream applications utilize arguments of an event instance independently.

5 Optimizing New Metrics

Optimizing these new metrics is pretty challenging. First, these metrics are non-differentiable, so it is hard to specify corresponding training objectives. Besides, we can only calculate event-level metrics after obtaining predictions for all event roles, so the supervision for role-level predictions is inevitably delayed. Moreover, EDAG generation is an autoregressive procedure inherently, which suffers from the error-propagation problem.

To address these challenges, we develop an RL-based training paradigm for Doc2EDAG by viewing EDAG generation as a sequential decision-making process. In this way, we can explicitly optimize model parameters by a delayed reward, which can be specified by a non-differentiable metric. Moreover, the explorations during RL-based training also help to stabilize EDAG generation.

MDP. Instead of treating the EDAG generation as an autoregressive procedure, we regard it as a Markov decision process (MDP). First, we consider the event triggering as the beginning of this MDP, which is also the virtual starting node of the EDAG (Zheng et al., 2019a). Then, for a specific event role, which corresponds to one step of the MDP, we need to take specific actions for all encoded entities at all leaf nodes of the current EDAG. Next, the EDAG grows accordingly, and we move to the next event role. This iterative process continues until we reach the last role. By following specific evaluation metrics to compare the generated EDAG with the ground-truth one, we can obtain the final reward to guide the decision-making process from a global perspective.

State. In general, the state at each step is the part of EDAG before the current event role. In this

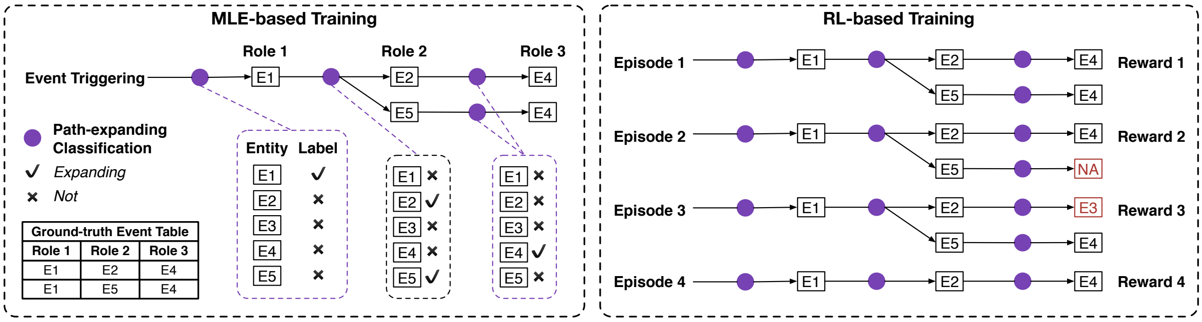

Role 1	Role 2	Role 3
E1	E2	E4
E1	E5	E4

Figure 3: An overview of the comparison between MLE-based training and RL-based training for Doc2EDAG, where all marks follow the meanings in Figure 1 and 2.

paper, to present a fair comparison with the vanilla Doc2EDAG with MLE-based training, we leverage the same method developed by Zheng et al. (2019a) to obtain state representations of the current EDAG. In this way, we can focus on the real impact of different supervision paradigms. For the step t, we denote its state as $\boldsymbol{s}_t$.

Action. At each step of the MDP, we need to take an action, *expanding* (1) or *not* (0), for each entity candidate at every leaf node of the current EDAG. We denote the action collection at step t as $\boldsymbol{a}_t$. Here the action concept is the same as the path-expanding prediction used in MLE-based training. Note that there are multiple actions to be made (one action per entity per expanding sub-task), which is a little bit similar to the setting of multi-agent RL (Buşoniu et al., 2010). While the difference is that our model unifies all these agents in a single model and enforces them to work collaboratively for a specific metric. Moreover, we need to face a growing number of actions during the expansion of an EDAG, which largely raises the difficulties of model training. Despite these challenges, we empirically demonstrate that the RL-based training can work pretty well with proper configurations.

Reward. For a specific evaluation metric, we utilize its rules to calculate the reward and use this reward to guide the optimization of model parameters. Moreover, there is no gap between training and inference because the model generates an EDAG by itself to get a reward during training, which includes the inference procedure.

Optimization. Given a document d and a triggered event type e with N_e event roles, we can take a series of actions to get an episode $\tau = (\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \boldsymbol{a}_2, \cdots, \boldsymbol{s}_{N_e}, \boldsymbol{a}_{N_e})$ and calculate the final reward R_τ by comparing the generated EDAG with the ground truth. Then, we can write the overall loss function for the EDAG generation as

$$L_{edag} = -\mathbb{E}_{d,e}\left[\mathbb{E}_{\pi_\Theta(\tau)} R_\tau\right], \qquad (1)$$

where $\mathbb{E}_{d,e}$ is the expectation over training documents and their triggered event types, Θ denotes the model parameters, $\pi_\Theta(\tau) = \prod_{t=1}^{N_e} \pi_\Theta(\boldsymbol{a}_t|\boldsymbol{s}_t)$ estimates the probability of an episode, and $\pi_\Theta(\boldsymbol{a}_t|\boldsymbol{s}_t)$ corresponds to the policy at step t. We employ the REINFORCE algorithm (Williams, 1992) and the policy gradients (Sutton et al., 1999) to optimize the above objective. And we can write the gradient $\nabla_\Theta L_{edag}$ as

$$-\mathbb{E}_{d,e}\left[\mathbb{E}_{\pi_\Theta(\tau)}\left[\sum_{t=1}^{N_e} R_\tau \log \pi_\Theta(\boldsymbol{a}_t|\boldsymbol{s}_t)\right]\right]. \qquad (2)$$

Moreover, we also follow Zheng et al. (2019a) to obtain the final loss by summing L_{edag}, the loss of entity recognition L_{er}, and the loss of event triggering L_{et} as $L_{all} = \lambda_1 L_{er} + \lambda_2 L_{et} + \lambda_3 L_{edag}$, where λ_1, λ_2 and λ_3 are hyper-parameters.

Exploration. In our case, the complexity of generating an EDAG grows exponentially with the number of entities and path-expanding sub-tasks. To achieve efficient explorations under such a challenging scenario, we start from MLE-based training to get a relatively well-trained model and warm start RL-based training from it. After the warm start, we let the model sample actions according to the predicted path-expanding probability to achieve proper explorations.

Figure 3 depicts the comparison between MLE-based training and RL-based training, where we can observe two benefits of the latter one: 1) the model learns from the global supervision (reward), which can be specified by a specific metric; 2) the

Event	#Train	#Dev	#Test	#Total
EF	806	186	204	1, 196
ER	1, 862	297	282	3, 677
EU	5, 268	677	346	5, 847
EO	5, 101	570	1, 138	6, 017
EP	12, 857	1, 491	1, 254	15, 602
All	25, 632	3, 204	3, 204	32, 040

Table 1: Statistics of the ChFinAnn dataset, including the number of documents on the train (#Train), development (#Dev), and test (#Test) sets as well as the total number of documents (#Total).

model can explore diverse EDAG structures during training, which helps to stabilize the dynamic process of EDAG generation.

6 Experiments

In this section, we present extensive empirical studies to answer two questions: 1) how severe is the overestimation of the initial *role-averaged* evaluation under the *event-as-a-whole* requirement? 2) to what extent can RL-based training improve MLE-based training under new evaluation metrics? Subsequently, Section 6.2 answers the first question. Section 6.3, 6.4, and 6.5 together answer the second question.

6.1 Experimental Setup

Model. As Section 3.2 illustrates, Doc2EDAG owns the superiority of both providing end-to-end modeling and simplifying data labeling. Therefore, in this paper, we adopt Doc2EDAG as the base model to demonstrate the challenges of new evaluation settings and the benefits of the RL-based training paradigm. Moreover, we follow the open-source implementation[2] to reproduce MLE-based training.

Dataset. The ChFinAnn dataset (Zheng et al., 2019a) is the largest public dataset for EE at present, and it is also the initial benchmark for Doc2EDAG. Therefore, we employ it as the testbed to compare RL-based training with the initial MLE-based training. ChFinAnn dataset includes ten-years Chinese financial documents accompanied with corresponding event tables and contains five event types: *equity freeze* (EF), *equity repurchase* (ER), *equity underweight* (EU), *equity overweight* (EO), and *equity pledge* (EP). Table 1 summarizes this dataset.

Metric	Avg. P.	Avg. R.	Avg. F$_1$
Base*	-	-	76.3
Base	83.8	70.6	76.6
NoFP	60.4	52.4	56.1
Drop	-23.4	-18.2	-20.5
NoFPFN	37.5	34.8	36.1
Drop	-46.3	-35.8	-40.5

Table 2: We present the overall performance (Avg.) of the vanilla Doc2EDAG under different evaluation settings by averaging scores for each event type, where * indicates the copy of results in Zheng et al. (2019a),"Base" denotes the initial *role-averaged* evaluation, "NoFP" denotes the *NoFP* evaluation, "NoFPFN" denotes the *NoFPFN* evaluation, and "Drop" represents the difference between the line above it and the "Base" line.

Hyper-parameters. MLE-based training sets all hyper-parameters the same as those presented in Zheng et al. (2019a). As for RL-based training, we utilize the following settings. To enable efficient explorations, we first train Doc2EDAG by MLE for 30 epochs and then turn to RL for another 70 epochs, where the total number of training epochs is still 100. To alleviate the noises of policy gradients, we sample five independent episodes for each document in the training batch and adopt a relatively small learning rate of $1e^{-5}$. For all other hyper-parameters, we set them the same as those in MLE-based training. Moreover, for each training method, we utilize the development set to pick the best epoch for the target evaluation metric.

Evaluation. We consider three evaluation settings: 1) the *role-averaged* evaluation, which is the base setting that provides approximate measures by averaging the performance of event roles; 2) the *NoFP* evaluation, strictly penalizing incorrect event arguments; 3) the *NoFPFN* evaluation, explicitly penalizing both incorrect and incomplete event arguments. For all settings, we collect evaluation statistics of TP, FP, and FN as Section 4 describes to compute the precision (P.), recall (R.), and F$_1$ scores (in the percentage format).

6.2 The Overestimation Problem of the Role-averaged Evaluation

In this section, we conduct experiments to reveal the overestimation problem of the initial *role-averaged* evaluation under the *event-as-a-whole* scenario. Following the setups mentioned above,

---

[2] `https://github.com/dolphin-zs/Doc2EDAG`

Metric	EF			ER			EU			EO			EP		
	P.	**R.**	**F_1**	**P.**	**R.**	**F_1**	**P.**	**R.**	**F_1**	**P.**	**R.**	**F_1**	**P.**	**R.**	**F_1**
Base*	77.1	64.5	70.2	91.3	83.6	87.3	80.2	65.0	71.8	82.1	69.0	75.0	80.0	74.8	77.3
Base	78.4	66.5	71.9	95.5	81.3	87.9	80.4	62.0	70.0	80.7	70.9	75.5	84.2	72.4	77.9
NoFP	49.7	46.3	47.9	78.1	70.6	74.2	61.7	48.5	54.3	58.7	48.0	52.8	53.7	48.7	51.1
Drop	-28.7	-20.2	-24.0	-17.4	-10.7	-13.7	-18.7	-13.5	-15.7	-22.0	-22.9	-22.7	-30.5	-23.7	-26.8
NoFPFN	36.2	32.1	34.0	40.1	39.4	39.8	39.2	35.6	37.3	35.2	33.3	34.3	36.5	33.6	35.0
Drop	-42.2	-34.4	-37.9	-55.4	-41.9	-48.1	-41.2	-26.4	-32.7	-45.5	-37.6	-41.2	-47.7	-38.8	-42.9

Table 3: We evaluate the vanilla Doc2EDAG under different settings and report results for all event types, where all marks follow those in Table 2.

we reproduce a Doc2EDAG model with the initial MLE-based training. Then, we evaluate this model with different evaluation methods. Table 2 presents the overall comparison, and Table 3 records performance comparisons for each event type. We can observe that the performance results of our reproduced model roughly match those reported by Zheng et al. (2019a) under the base setting. However, there is a drastic drop in the performance scores under new metrics. On average, the F_1 score decreases 20.5 under the *NoFP* metric, and the F_1 decrement reaches 40.5 under the *NoFPFN* metric. The *role-averaged* metric does not consider the *event-as-a-whole* requirement and tends to produce overestimated performance that is too optimistic to fit the real situation. The vast performance degradation demonstrates that the overestimation problem can be quite severe.

Moreover, the degree of overestimation depends on the event type and varies irregularly. For example, as Table 3 shows, the model achieves similar F_1 scores for EF and EU events (71.9 vs. 70.0), but the performance drops under the *NoFP* evaluation are quite different (-24.0 vs. -15.7). The underlying reason is that the *role-averaged* evaluation only tells us the averaged performance of role predictions, which is an approximate measure. Therefore, in downstream applications that utilize an event instance as a whole, it is critical for them to utilize our newly designed metrics to be aware of the real performance.

6.3 The NoFP Evaluation

The *NoFP* evaluation considers an event instance as a whole and strictly penalizes FP entities. Under this evaluation setting, we compare the RL-based training that directly optimizes the specific metric with the initial MLE-based training that mimics path-expanding predictions of a ground-truth EDAG. Table 4 shows the overall performance comparisons, and Figure 4 presents performance improvements of RL-based training for all event types. We can observe that, on average, RL-based training improves the MLE-based one by 8.9% of F_1 scores, and similar improvements happen for all event types. These vast improvements demonstrate that considering the *NoFP* metric explicitly during training can bring remarkable benefits.

Model	Avg.		
	P.	**R.**	**F_1**
MLE	60.4	52.4	56.1
RL	66.2	56.7	61.1
Inc	+5.8	+4.3	+5.0

Table 4: The averaged (Avg.) performance of all event types under the *NoFP* evaluation, where “MLE” denotes Doc2EDAG with MLE-based training, “RL” denotes the model with RL-based training, and “Inc” represents the increment between “RL” and “MLE”.

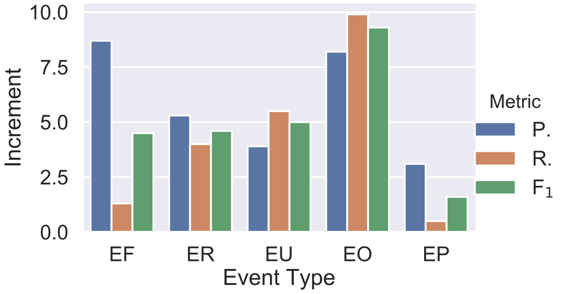

Figure 4: The performance increment between RL-based training and MLE-based training for all event types under the *NoFP* evaluation.

6.4 The NoFPFN Evaluation

We further consider the *NoFPFN* evaluation that penalizes not only FP entities but also FN entities. As Table 5 shows, RL-based training obtains similar improvements over MLE-based training on average. To be specific, the relative improvement in terms of the F_1 score reaches 11.3%. Since RL-

Model	**Avg.** **P.**	 **R.**	 **F_1**
MLE	37.5	34.8	36.1
RL	39.6	40.7	40.2
Inc	+2.1	+5.9	+4.1

Table 5: The averaged (Avg.) performance of all event types under the *NoFPFN* evaluation, where all abbreviations follow Table 4.

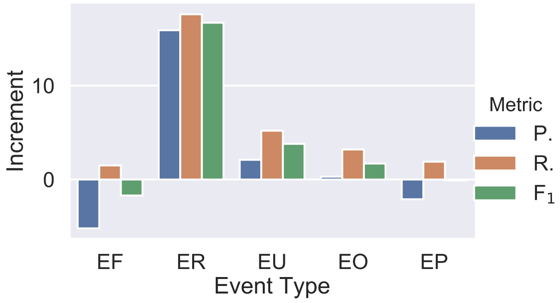

Figure 5: The performance increment between RL-based training and MLE-based training for all event types under the *NoFPFN* evaluation.

based training and MLE-based training share the same network architecture, we can clearly see the benefits of evaluation-aware learning.

Moreover, Figure 5 shows performance improvements of RL-based training for all event types. Interestingly, under the much challenging *NoFPFN* evaluation, we observe that the model automatically finds a trade-off between different event types. For example, the model sacrifices a little performance on EF (-1.7) and EP (-0.1) events but earns much more improvements on ER ($+16.7$), EU ($+3.8$), and EO ($+1.7$) events. Such an automatic trade-off is quite appealing because it is entirely data-driven and can adapt to different scenarios flexibly.

6.5 The Role-averaged Evaluation

In this subsection, we include performance comparisons under the initial setting, the *role-averaged* evaluation, where RL-based training follows the initial rule to calculate rewards. Table 6 shows the averaged comparison results. We can observe that MLE-based training achieves competitive performance compared with RL-based training. The underlying reason is that both MLE-based training and the *role-averaged* evaluation focus on improving the role-level predictions, so these two techniques are much consistent with each other. Nevertheless, RL-based training still obtains a small gain of the F_1 score ($+1.1$) because it can make a proper trade-off between the precision and the recall. Figure 6 presents performance improvements for all event types, and we also observe similar trade-offs. Therefore, we can conclude that RL-based training can achieve remarkable improvements under new metrics while still maintain competitive performance for the initial setting.

Model	**Avg.** **P.**	 **R.**	 **F_1**
MLE	83.8	70.6	76.6
RL	79.9	75.7	77.7
Inc	-3.9	+5.1	+1.1

Table 6: The averaged (Avg.) performance of all event types under the *role-averaged* evaluation, where all abbreviations follow Table 4.

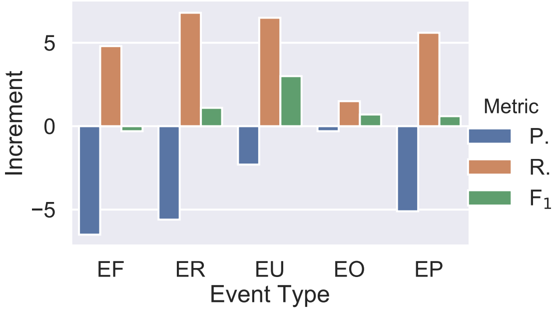

Figure 6: The performance increment between RL-based training and MLE-based training for all event types under the *role-averaged* evaluation.

7 Conclusion and Future Work

In conclusion, our new metrics provide a more complete understanding of performance than the *role-averaged* metric under the *event-as-a-whole* scenario. Moreover, our proposed RL-based training for Doc2EDAG can be a better choice than the MLE-based one when adapting the model to these new metrics. Extensive experiments demonstrate the necessity of new evaluation metrics and the superiority of the novel training scheme.

Given the necessity of employing new metrics, however, as Table 2 shows, the current performance is still far from satisfactory, even for RL-based training. Therefore, further improving EE under these new metrics calls for much more attention from the research community.

Acknowledgements

This research is supported in part by gift funds from the Nanjing Turing AI Institute and the Zhongguancun Haihua Institute for Frontier Information Technology.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*.

Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. Multi-agent reinforcement learning: An overview. In *Proceedings of Innovations in multi-agent systems and applications-1*. Springer.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of ACL*.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of EMNLP*.

Alex Judea and Michael Strube. 2016. Incremental global event extraction. In *Proceedings of COLING*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*.

Qi Li, Heng Ji, HONG Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of EMNLP*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of ACL*.

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019. Event detection without triggers. In *Proceedings of NAACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proceedings of AAAI*.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Proceedings of NAACL*.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of EMNLP*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *Proceedings of AAAI*.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of NIPS*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of EMNLP*.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *Proceedings of NAACL*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of NAACL*.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: A document-level Chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*.

Ying Zeng, Yansong Feng, Rong Ma, Zheng Wang, Rui Yan, Chongde Shi, and Dongyan Zhao. 2018. Scale up event extraction learning via automatic training data generation. In *Proceedings of AAAI*.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019a. Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction. In *Proceedings of EMNLP*.

Shun Zheng, Xu Han, Yankai Lin, Peilin Yu, Lu Chen, Ling Huang, Zhiyuan Liu, and Wei Xu. 2019b. DIAG-NRE: A neural pattern diagnosis framework for distantly supervised neural relation extraction. In *Proceedings of ACL*.

Direct Simultaneous Speech-to-Text Translation Assisted by Synchronized Streaming ASR [*]

Junkun Chen [1] **Mingbo Ma** [2] **Renjie Zheng** [2] **Liang Huang** [1,2]
[1]Oregon State University, Corvallis, OR, USA
[2]Baidu Research, Sunnyvale, CA, USA
`chenjun2@oregonstate.edu, mingboma@baidu.com`

Abstract

Simultaneous speech-to-text translation is widely useful in many scenarios. The conventional cascaded approach uses a pipeline of streaming ASR followed by simultaneous MT, but suffers from error propagation and extra latency. To alleviate these issues, recent efforts attempt to directly translate the source speech into target text simultaneously, but this is much harder due to the combination of two separate tasks. We instead propose a new paradigm with the advantages of both cascaded and end-to-end approaches. The key idea is to use two separate, but synchronized, decoders on streaming ASR and direct speech-to-text translation (ST), respectively, and the intermediate results of ASR guide the decoding policy of (but is not fed as input to) ST. During training time, we use multitask learning to jointly learn these two tasks with a shared encoder. En-to-De and En-to-Es experiments on the MuST-C dataset demonstrate that our proposed technique achieves substantially better translation quality at similar levels of latency.

1 Introduction

Simultaneous speech-to-text translation incrementally translates source-language speech into target-language text, and is widely useful in many cross-lingual communication scenarios such as international travels and multinational conferences. The conventional approach to this problem is a cascaded one (Arivazhagan et al., 2020; Xiong et al., 2019; Zheng et al., 2020b), involving a pipeline of two steps. First, the streaming automatic speech recognition (ASR) module transcribes the input speech on the fly (Moritz et al., 2020; Wang et al., 2020), and then a simultaneous text-to-text translation module translates the partial transcription into target-language text (Oda et al., 2014; Dalvi et al., 2018; Ma et al., 2019; Zheng et al., 2019a,b, 2020a; Arivazhagan et al., 2019).

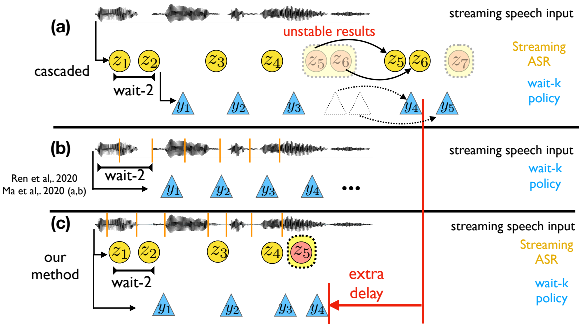

Figure 1: Comparison between (a) cascaded pipeline, (b) direct simultaneous ST, and (c) our ASR-assisted simultaneous ST. In (a), streaming ASR keeps revising some tail words for better accuracy, but causing extra delays to MT. Method (b) directly translates source speech without using ASR. Our work (c) uses the intermediate results of the streaming ASR module to guide the decoding policy of (but not feed as input to) the speech translation module. Extra delays between ASR and MT are reduced in direct translation systems (b–c).

However, the cascaded approach inevitably suffers from two limitations: (a) **error propagation**, where streaming ASR's mistakes confuse the translation module (which are trained on clean text), and this problem worsens with noisy environments and accented speech; and (b) **extra latency**, where the translation module has to wait until streaming ASR's output stabilizes, as ASR by default can repeatedly revise its output (see Fig. 1).

To overcome the above issues, some recent efforts (Ren et al., 2020; Ma et al., 2020b,a) attempt to directly translate the source speech into target text simultaneously by adapting text-based wait-k strategy (Ma et al., 2019). However, unlike simultaneous translation whose input is already segmented into words or subwords, in speech translation, the key challenge is to figure out the number of valid tokens within a given source speech segment in or-

---

[*] See our translation examples and demos at https://littlechencc.github.io/SimulST-demo/simulST-demo.html.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4618–4624
August 1–6, 2021.

der to apply the wait-k policy. Ma et al. (2020b,a) simply assume a fixed number of words within a certain number of speech frames, which does not consider various aspects of speech such as different speech rate, duration, pauses and silences, all of which are common in realistic speech. Ren et al. (2020) design an extra Connectionist Temporal Classification (CTC)-based speech segmenter to detect the word boundaries in speech. However, the CTC-based segmenter inherits the same shortcoming of CTC, which only makes local predictions, thus limiting its segmentation accuracy. On the other hand, to alleviate the error propagation, Ren et al. (2020) employ several different knowledge distillation techniques to learn the attentions of ASR and MT jointly. These knowledge distillation techniques are complicated to train and it is an indirect solution for the error propagation problem.

We instead present a simple but effective solution (see Fig. 2) by employing two separate, but synchronized, decoders, one for streaming ASR and the other for End-to-End Speech-to-text Translation (E2E-ST). Our key idea is to use the intermediate results of streaming ASR to guide the decoding policy of, but not feed as input to, the E2E-ST decoder. We look at the beam of streaming ASR to decide the number of tokens within the given source speech segment. Then it is straightforward for the E2E-ST decoder to apply the wait-k policy and decide whether to commit a target word or to wait for more speech frames. During training time, we jointly train ASR and E2E-ST tasks with a shared speech encoder in a multi-task learning (MTL) fashion to further improve the translation accuracy. We also note that having streaming ASR as an auxiliary output is extremely useful in real application scenarios where the user often wants to see both the transcription and the translation. En-to-De and En-to-Es experiments on the MuST-C dataset demonstrate that our proposed technique achieves substantially better translation quality at similar level of latency.

2 Preliminaries

We formalize full-sentence tasks (ASR, MT and ST) using the sequence-to-sequence framework, and the streaming tasks (simultaneous MT and streaming ASR) using the test-time wait-k method.

Full-Sentence Tasks: ASR, NMT and ST The encoder first encodes the entire source input into a sequence of hidden states; in NMT, the input is a sequence of words, $\mathbf{x} = (x_1, x_2, ..., x_m)$, while in ASR and ST, we use $\mathbf{s}$ to denote the input speech frames. A decoder sequentially predicts target language tokens $\mathbf{y} = (y_1, y_2, ..., y_n)$ in NMT and ST or transcription $\mathbf{z}$ in ASR, conditioned on all encoder hidden states and previously committed tokens. For example, the NMT model and its parameters $\boldsymbol{\theta}^{\text{MT}}_{\text{full}}$ are defined as:

$$p_{\text{full}}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}^{\text{MT}}_{\text{full}}) = \prod\nolimits_{t=1}^{|\mathbf{y}|} p(y_t \mid \mathbf{x}, \mathbf{y}_{<t}; \boldsymbol{\theta}^{\text{MT}}_{\text{full}})$$

$$\hat{\boldsymbol{\theta}}^{\text{MT}}_{\text{full}} = \mathop{\mathbf{argmax}}_{\boldsymbol{\theta}^{\text{MT}}_{\text{full}}} \prod_{(\mathbf{x},\mathbf{y}^*)\in D} p_{\text{full}}(\mathbf{y}^* \mid \mathbf{x}; \boldsymbol{\theta}^{\text{MT}}_{\text{full}})$$

Similarly, we can obtain the definitions for ASR ($p_{\text{full}}(\mathbf{z} \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ASR}}_{\text{full}})$) and ST ($p_{\text{full}}(\mathbf{y} \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ST}}_{\text{full}})$). Our model was learned from scratch in this work, but it can be improved with pre-training methods (Zheng et al., 2021; Chen et al., 2020).

Simultaneous MT and Streaming ASR In streaming decoding scenarios, we have to predict target tokens conditioned on the partial source input that is available. For example, the **test-time wait-k** method of Ma et al. (2019) predicts each target token y_t after reading source tokens $\mathbf{x}_{\le t+k}$ using a full-sentence NMT model:

$$\hat{y}_t = \mathop{\mathbf{argmax}}_{y_t} p_{\text{wait-}k}(y_t \mid \mathbf{x}_{\le t+k}, \hat{\mathbf{y}}_{<t}; \hat{\boldsymbol{\theta}}^{\text{MT}}_{\text{full}}) \tag{1}$$

Intuitively speaking, wait-k only commits a new target word on receiving each new source word after an initial k source words waiting. Similarly, in the case of streaming ASR, we could define $\hat{z}_t$ with growing speech chunks $\bar{\mathbf{s}}_i$ that are fed gradually.

3 Direct Simultaneous Translation with Synchronized Streaming ASR

In text-to-text simultaneous translation, the input stream is already segmented. However, when we deal with speech frames as source inputs, it is not easy to determine the number of valid tokens within certain speech segments. Therefore, to better guide the translation policy, it is essential to detect the number of valid tokens accurately within low latency. Different from the sophisticated design of speech segmenter in Ren et al. (2020), we propose a simple but effective method by using a synchronized streaming ASR and using its beam to determine the number of words within certain speech segments. Note that we only use streaming ASR for source word counting, but the translation decoder does not condition on any of ASR's output.

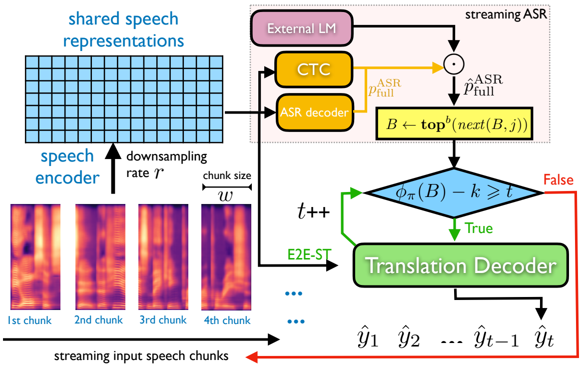

Figure 2: Decoding for synchronized streaming ASR and E2E-ST. Speech signals are fed into the encoder chunk by chunk. For each new-coming speech chunk, we look at the current streaming ASR beam (B) to decide the translation policy. See details in Algorithm 1.

3.1 Streaming ASR-Guided Simultaneous ST

As shown in Fig. 2, at inference time, the speech signals are fed into the ST encoder by a series of fixed-size chunks $\overline{\mathbf{s}}_{[1:i]} = [\overline{\mathbf{s}}_1, ..., \overline{\mathbf{s}}_i]$, where $w = |\overline{\mathbf{s}}_i|$ can be chosen from 32, 48 and 64 frames of spectrogram. As a result of the CNN encoder, there is down sampling rate r (e.g., we use $r = 4$), from spectrogram to encoder hidden states. For example, when we receive a chunk of 32 frames, the encoder will generate 8 more hidden states. In conventional streaming ASR, the number of steps of beam search is the same as the number of hidden states.

We denote B_j to be the beam at time step j, which is an ordered list of size of b, and it expands to the next beam B_{j+1} with the same size:

$$\begin{aligned} B_0 =& [\langle \text{<s>},\ \hat{p}_{\text{full}}^{\text{ASR}}(\text{<s>} \mid \overline{\mathbf{s}}_0; \boldsymbol{\theta})\rangle] \\ B_j =& \mathbf{top}^b(next(B_{j-1}, j)) \\ next(B, j) =& \{\langle \mathbf{z} \circ z_j,\ p \cdot \hat{p}_{\text{full}}^{\text{ASR}}(z_j \mid \overline{\mathbf{s}}_{\leq \tau(j)}, \mathbf{z}; \boldsymbol{\theta})\rangle \mid \\ & \langle \mathbf{z}, p\rangle \in B, z_j \in V\} \end{aligned}$$

where $\mathbf{top}^b(\cdot)$ returns the top b candidates, and $next(B, j)$ expands the candidates from the previous step to the next step. Each candidate is a pair $\langle \mathbf{z}, p\rangle$, where $\mathbf{z}$ is the current prefix and p is the accumulated probability from joint score between an external language model, CTC and ASR probabilities, $\hat{p}_{\text{full}}^{\text{ASR}}$. We denote the number of observable speech chunks at j step as $\tau(j) = \lceil j * r/w \rceil$. And vice versa, for each new speech chunk, ASR beam search will advance for w/r steps.

Note CTC often commits empty tokens ϵ due to empty speech frames, and the lengths of different hypotheses within beam of streaming ASR are quite different from each other. To take every hypothesis into consideration, we design two policies to decide the number of valid tokens.

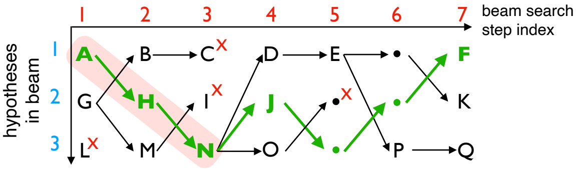

Figure 3: An example of streaming ASR beam search with beam size 3. LCP is shaded in red ($\phi_{\text{LCP}}(B_7) = 3$); SH is highlighted in bold ($\phi_{\text{SH}}(B_7) = 5$). We use • to represent empty outputs in some steps caused by CTC.

Algorithm 1 Streaming ASR-guided Simultaneous ST

```
Input: speech chunks $\overline{\mathbf{s}}_{[1:T]}$; $k$; $\phi_\pi(B_j)$; streaming decoding models: $p_{\text{full}}^{\text{ST}}$ and $\hat{p}_{\text{full}}^{\text{ASR}}$
Initialize: ASR and ST indices: $j = t = 0$; $B = B_0$
for $i = 1 \sim T$ do                                   ▷ feed speech chunks
    repeat $w/r$ steps                     ▷ do ASR beam search $w/r$ steps
        $B \leftarrow \mathbf{top}^b(next(B, j))$; $j$++         ▷ ASR beam search
    while $\phi_\pi(B) - k \geqslant t$ do                        ▷ new tokens?
        $\hat{y}_{t+1} \leftarrow p_{\text{wait-}k}^{\text{ST}}(y_{t+1} \mid \overline{\mathbf{s}}_{[1:i+1]}, \hat{\mathbf{y}}_{\leq t}; \boldsymbol{\theta}_{\text{full}}^{\text{ST}})$
        yield $\hat{y}_{t+1}$; $t$++            ▷ commit translation to user
```

- **Longest Common Prefix** (LCP) uses the length of longest shared prefix in the streaming ASR beam as the number of valid tokens within given speech. This is the most conservative strategy, which has similar latency to cascaded methods.
- **Shortest Hypothesis** (SH) uses the length of shortest hypothesis in the current streaming ASR beam as the number of valid tokens.

More formally, let $\phi_\pi(B)$ denote the number of valid tokens in the beam B under policy π:

$$\begin{aligned} \phi_{\text{LCP}}(B) &= \max\{i \mid \exists \mathbf{z}', s.t. \forall \langle \mathbf{z}, c\rangle \in B, \mathbf{z}_{\leq i} = \mathbf{z}'\} \\ \phi_{\text{SH}}(B) &= \min\{|\mathbf{z}| \mid \langle \mathbf{z}, c\rangle \in B\} \end{aligned}$$

For example in Fig. 3, $\phi_{\text{LCP}}(B_7) = 3$, $\phi_{\text{SH}}(B_7) = 5$. Also note that $\phi_{\text{LCP}}(B) \leq \phi_{\text{SH}}(B)$ for any beam B, and that both policies are *monotonic*, i.e. $\phi_\pi(B_j) \leq \phi_\pi(B_{j+1})$ for $\pi \in \{\text{LCP}, \text{SH}\}$ and all j.

Note we always feed the entire observable speech segments into ST for translation, and streaming ASR-generated transcription is not used for translation, so LCP might have similar latency with cascaded methods but the translation accuracy is much better because more information on the source side is revealed to the translation decoder.

As shown in Algorithm 1, during simultaneous ST, we monitor the value of $\phi_\pi(B_j)$ while speech

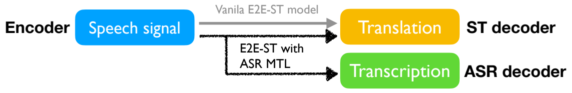

Figure 4: We use full-sentence MTL framework to jointly learn ASR and ST with a shared encoder.

chunks are gradually fed into system. When we have $\phi_\pi(B) - k \geqslant t$ where t is the number of translated tokens, the ST decoder will be triggered to generate one new token as follows:

$$\hat{y}_t = \mathop{\mathbf{argmax}}_{y_t} p_{\text{wait-}k}(y_t \mid \bar{\mathbf{s}}_{[1:\tau(j)]}, \hat{\mathbf{y}}_{<t}; \hat{\boldsymbol{\theta}}^{\text{ST}}_{\text{full}}) \quad (2)$$

3.2 Joint Training between ST and ASR

Different from existing simultaneous translation solutions from (Ren et al., 2020; Ma et al., 2020b,a), which make adaptations over vanilla E2E-ST architecture as shown in gray line of Fig. 4, we instead use simple MTL architecture which performs joint full-sentence training between ST and ASR:

$$\hat{\boldsymbol{\theta}}^{\text{ST}}_{\text{full}}, \hat{\boldsymbol{\theta}}^{\text{ASR}}_{\text{full}} = \mathop{\mathbf{argmax}}_{\boldsymbol{\theta}^{\text{ST}}_{\text{full}}, \boldsymbol{\theta}^{\text{ASR}}_{\text{full}}} \prod_{(\mathbf{s}, \mathbf{y}^*, \mathbf{z}^*) \in D} p^{\text{ST}}_{\text{full}}(\mathbf{y}^* \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ST}}_{\text{full}}) \cdot p^{\text{ASR}}_{\text{full}}(\mathbf{z}^* \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ASR}}_{\text{full}})$$

For ASR training, we use hybrid CTC/Attention framework (Watanabe et al., 2017). Note that we train ASR and ST MTL with full-sentence fashion for simplicity and training efficiency, and only perform wait-k decoding policy at inference time. Also, $\boldsymbol{\theta}^{\text{ST}}_{\text{full}}$ and $\boldsymbol{\theta}^{\text{ASR}}_{\text{full}}$ share the same speech encoder.

4 Experiments

We conduct experiments on English-to-German (En→De) and English-Spanish (En→Es) translation on MuST-C (Di Gangi et al., 2019). We employ Transformer (Vaswani et al., 2017) as the basic architecture and LSTM (Hochreiter and Schmidhuber, 1997) for LM. For streaming ASR decoding we use a beam size of 5. Translation decoding is greedy due to incremental commitment.

Raw audios are processed with Kaldi (Povey et al., 2011) to extract 80-dimensional log-Mel filterbanks stacked with 3-dimensional pitch features using a 10ms step size and a 25ms window size. Text is processed by SentencePiece (Kudo and Richardson, 2018) with a joint vocabulary size of 8K. We take Transformer (Vaswani et al., 2017) as our base architecture, which follows 2 layers of 2D convolution of size 3 with stride size of 2. The Transformer model has 12 encoder layers and 6 decoder layers. Each layer has 4 attention head with a size of 256. Our streaming ASR decoding method follows Moritz et al. (2020). We employ 10 frames look ahead for all experiments. For LM, we use 2 layers stacked LSTM (Hochreiter and Schmidhuber, 1997) with 1024-dimensional hidden states, and set the embedding size as 1024. LM are trained on English transcription from the corresponding language pair in MuST-C corpus. For the cascaded model, we train ASR and MT models on Must-C dataset respectively, and they have the same Transformer architecture of our ST model. Our experiments are run on 8 1080Ti GPUs. And the we report the case-sensitive detokenized BLEU.

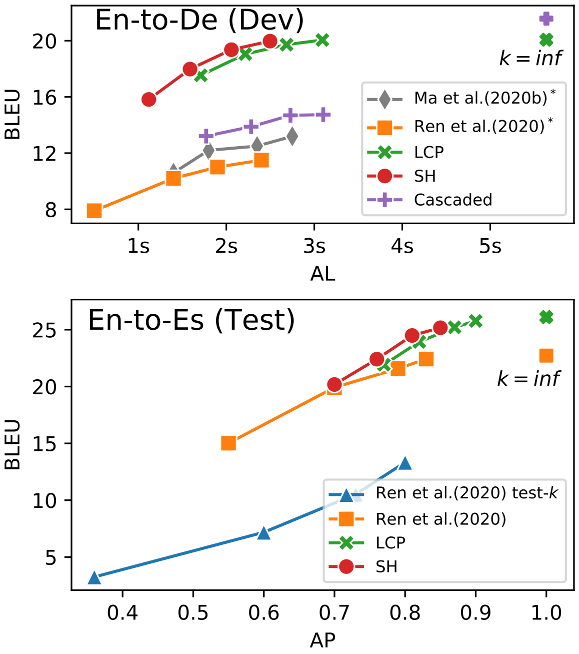

Figure 5: Translation quality v.s. latency. The dots on each curve represents different wait-k policy with k=1,3,5,7 from left to right respectively. Baseline* results are from Ma et al. (2020b). k=*inf* is full-sentence decoding for ASR and translation. test-k denotes testing time wait-k. We use a chunk size of 48.

Translation quality against latency In order to clearly compare with related works, we evaluate the latency with AL defined in Ma et al. (2020b) and AP defined in Ren et al. (2020). As shown in Fig. 5, for En→De, results are on the dev set to be consistent with Ma et al. (2020b). Compared with baseline models, our method achieves much better translation quality with similar latency. To validate the effectiveness of our method, we compare our method with Ren et al. (2020) on En→Es translation. Their method does not evaluate the

chunk index	1	2	3	4	5	6	end
Gold transcript	can I be	**honest**	*SIL*	I don 't **love**	**that question**	*SIL*	
Gold translation	Darf ich **ehrlich** sein ?			Ich **mag diese Frage** nicht .			
Streaming ASR	can I			be **on this** I don 't	**love that question**		
simul-MT wait-3				Kann ich **da** sein ? "	Ich **liebe**		**diese Frage** nicht .
SH wait-3				Kann ich **ehrlich** sein ? Ich **liebe**	**diese Frage**		nicht .
LCP wait-3				Kann ich **ehrlich** sein ? Ich			**liebe diese Frage** nicht .

Figure 6: An example from the dev set of En→De translation. In the cascaded approach (streaming ASR + simul-MT wait-3), the ASR error (*"on this"* for *"honest"*) is propagated to the MT module, causing the wrong translation (*"da"*). Our methods give accurate translations ("ehrlich") with better latency (esp. for the SH policy, the output of *"diese Frage"* is synchronous with hearing *"that question"*). *"SIL"* denotes silence in speech.

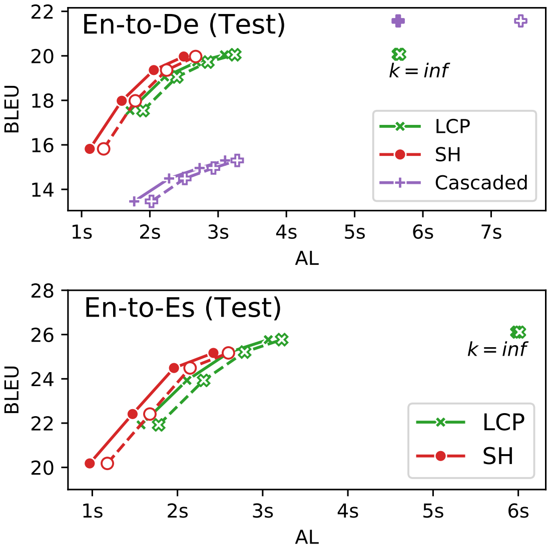

Figure 7: Translation quality against latency. Each curve represents decoding with wait-k policy, k=1,3,5,7 from left to right. The dashed lines and hollow markers indicate the latency considering the computational time. The chunk size is 48.

Model	En→De			En→Es		
	$w=32$	$w=48$	$w=64$	$w=32$	$w=48$	$w=64$
LCP	17.31	17.54	17.95	21.94	21.92	22.36
− *LM*	14.60	15.66	15.91	18.54	19.15	19.95
− *LM* & *AD*	13.76	14.82	15.26	17.42	18.06	19.32
SH	16.04	15.82	15.87	20.45	20.18	19.84
− *LM*	13.76	14.01	13.84	17.31	17.21	17.78
− *LM* & *AD*	10.44	11.25	11.65	13.61	14.27	14.62

Table 1: BLEU score of wait-1 decoding with different chunk sizes and ASR scoring functions. *AD* denotes ASR Decoder. *LM* denotes Language Model.

plausibility of the detected tokens, so it has a more aggressive decoding policy which results in lower latency. However, our method can still achieve better results with slightly lower latency. Besides that, our model is trained in full-sentence mode, and only decodes with wait-k at inference time, which is very efficient to train. Our test-time wait-k could achieve similar quality with their genuine wait-k (i.e., retrained) models which are very slow to train. When we compare with their test-time wait-k, our model significantly outperforms theirs.

We further evaluate our method on the test set of En→De and En→Es translation. As shown in Fig. 7, compared with the cascaded model, our model has notable successes in latency and translation quality. To verify the online usability of our model, we also show computational-aware latency. Because our chunk window is 480ms, and the latency caused by the computation is smaller than this window size, which means that we can finish decoding the previous speech chunk when the next speech chunk needs to be processed, so our model can be effectively used online.

Fig. 6 demonstrates that our method can effectively avoid the error propagation and obtain better latency compared to the cascaded model.

Effect of chunk size and joint decision Table 1 shows that the results are relatively stable with various chunk sizes. It can be flexible to balance the response frequency and computational ability. We explore the effectiveness of ASR joint scoring, and observe that the translation quality drops a lot without LM. Without LM and AD, our token recognition approach is similar to the speech segmentation in Ren et al. (2020), which implies that their model is hard to segment the source speech accurately, leading to unreliable translation decisions for ST.

5 Conclusion

We proposed a simple but effective ASR-assisted simultaneous E2E-ST framework. The streaming ASR module can guide (but not give direct input to) the wait-k policy for simultaneous translation. Our method improves ST accuracy with similar latency.

Acknowledgments

This work is supported in part by NSF IIS-1817231 and IIS-2009071.

References

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. *Meeting of the Association for Computational Linguistics*.

Naveen Arivazhagan, Colin Cherry, Isabelle Te, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2020. Re-translation strategies for long form, simultaneous, spoken language translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7919–7923. IEEE.

Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. 2020. Mam: Masked acoustic modeling for end-to-end speech-to-text translation. *arXiv preprint arXiv:2010.11445*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020a. SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

Xutai Ma, Yongqiang Wang, Mohammad Javad Dousti, P. Koehn, and J. Pino. 2020b. Streaming simultaneous speech translation with augmented memory transformer. *ArXiv*, abs/2011.00033.

Niko Moritz, Takaaki Hori, and Jonathan Le. 2020. Streaming automatic speech recognition with the transformer model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6074–6078. IEEE.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Nagendra Goel, Mirko Hannemann, Yanmin Qian, Petr Schwarz, and Georg Stemmer. 2011. The kaldi speech recognition toolkit. In *In IEEE 2011 workshop*.

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Chengyi Wang, Yu Wu, Shujie Liu, Jinyu Li, Liang Lu, Guoli Ye, and Ming Zhou. 2020. Low latency end-to-end streaming speech recognition with a scout network. *arXiv preprint arXiv:2003.10369*.

S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition.

Hao Xiong, Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Dutongchuan: Context-aware translation model for simultaneous interpreting. *arXiv preprint arXiv:1907.12984*.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020a. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *ACL*.

Renjie Zheng, Junkun Chen, Mingbo Ma, and Liang Huang. 2021. Fused acoustic and text encoding for multimodal bilingual pretraining and speech translation. *Proceedings of the 38th International Conference on Machine Learning*.

Renjie Zheng, Mingbo Ma, Baigong Zheng, Kaibo Liu, Jiahong Yuan, Kenneth Church, and Liang Huang. 2020b. Fluent and low-latency simultaneous speech-to-speech translation with self-adaptive training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3928–3937.

HIT: A Hierarchically Fused Deep Attention Network for Robust Code-mixed Language Representation

Ayan Sengupta[1], **Sourabh Kumar Bhattacharjee**[1],
Tanmoy Chakraborty[2], **Md Shad Akhtar**[2]
[1]Optum Global Advantage (UnitedHealth Group), Noida, India
[2]Dept. of CSE, IIIT-Delhi, India
`{ayan.sengupta007, sourabhb398}@gmail.com;`
`{tanmoy, shad.akhtar}@iiitd.ac.in`

Abstract

Understanding linguistics and morphology of resource-scarce code-mixed texts remains a key challenge in text processing. Although word embedding comes in handy to support downstream tasks for low-resource languages, there are plenty of scopes in improving the quality of language representation particularly for code-mixed languages. In this paper, we propose HIT, a robust representation learning method for code-mixed texts. HIT is a hierarchical transformer-based framework that captures the semantic relationship among words and hierarchically learns the sentence-level semantics using a fused attention mechanism. HIT incorporates two attention modules, a multi-headed self-attention and an outer product attention module, and computes their weighted sum to obtain the attention weights. Our evaluation of HIT on one European (*Spanish*) and five Indic (*Hindi*, *Bengali*, *Tamil*, *Telugu*, and *Malayalam*) languages across four NLP tasks on eleven datasets suggests significant performance improvement against various state-of-the-art systems. We further show the adaptability of learned representation across tasks in a transfer learning setup (with and without fine-tuning).

1 Introduction

India is the second most populated country in the world, where ~ 1.36 billion people speak in over 200 different languages. Among them, the top five languages (*Hindi, Bengali, Telegu, Tamil,* and *Malayalam*) covers $\sim 93\%$of the entire population with more than 26% of them being bilingual (as per Wikipedia). Moreover, a significant proportion of them (Singh et al., 2018a) use code-mixed languages to express themselves in Online Social Networks (OSN).

Code-mixing (CM) is a linguistic phenomenon in which two or more languages are alternately used during conversation. One of the languages is usually English, while the other can be any regional language such as Hindi (Hindi + English → *Hinglish*), Bengali (Bengali + English → *Benglish*), Spanish (Spanish + English → *Spaniglish*), etc. Their presence on social media platforms and in day-to-day conversions among the people of a multi-lingual communities (such as Indians) is overwhelming. Despite the fact that a significant population is comfortable with code-mixed languages, the research involving them is fairly young. One of the prime reasons is the linguistic diversity, i.e., research on any language often fails to adapt for other distant languages, thus they need to be studied and researched separately. In recent years, many organizations have identified the challenges and have put in commendable efforts for the development of computational systems in regional monolingual or code-mixed setups.

Traditionally, the NLP community has studied the code-mixing phenomenon from a task-specific point of view. Recently, a few studies (Pratapa et al., 2018; Aguilar and Solorio, 2020) have started learning representations for code-mixed texts for semantic and syntactic tasks. While the former has showcased the importance of multi-lingual embeddings from CM text, the latter has made use of a hierarchical attention mechanism on top of positionally aware character bi-grams and tri-grams to learn robust word representations for CM text. Carrying over the same objective, in this paper, we introduce a novel **HI**erarchically attentive **T**ransformer (HIT) framework to effectively encode the syntactic and semantic features in embeddings space. At first, HIT learns sub-word level representations employing a fused attention mechanism (FAME) – an *outer-product* based attention mechanism (Le et al., 2020) fused with standard multi-headed self-attention (Vaswani et al., 2017). The intuition of sub-word level representation learning is supple-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4625–4639
August 1–6, 2021.

mented by the lexical variations of a word in code-mixed languages. The *character-level* HIT helps in representing phonetically similar word and their variations to a similar embedding space and extracts better representation for noisy texts. Subsequently, we apply HIT module at *word-level* to incorporate the semantics at the sentence-level. The output of HIT is a sequence of word representations, and can be fed to the architectures of any downstream NLP tasks. For the evaluation of HIT, we experiment on one classification (sentiment classification), one generative (MT), and two sequence-labelling (POS tagging and NER) tasks. In total, these tasks span to eleven datasets across six code-mixed languages – one European (*Spanish*) and five Indic (*Hindi*, *Bengali*, *Telugu*, *Tamil*, and *Malayalam*). Our empirical results show that representations learned by HIT are superior to existing multilingual and code-mixed representations, and report state-of-the-art performance across all tasks. Additionally, we observe encouraging adaptability of HIT in a transfer learning setup across tasks. The representations learned for a task is employed for learning other tasks w/ and w/o fine-tuning. HIT yields good performance in both setups for two code-mixed languages.

Main contributions: We summarize our contributions as follow:

- We propose a hierarchical attention transformer framework for learning word representations of code-mixed texts for six non-English languages.
- We propose a hybrid self-attention mechanism, FAME, to fuse the multi-headed self-attention and outer-product attention mechanisms in our transformer encoders.
- We show the effectiveness of HIT on eleven datasets across four NLP tasks and six languages.
- We observe good task-invariant performance of HIT in a transfer learning setup for two code-mixed languages.

Reproducibility: Source codes, datasets and other details to reproduce the results have been made public at `https://github.com/LCS2-IIITD/HIT-ACL2021-Codemixed-Representation`.

2 Related Work

Recent years have witnessed a few interesting work in the domain of code-mixed/switched representation learning. Seminal work was driven by bilingual embedding that employs cross-lingual transfer to develop NLP models for resource-scarce languages (Upadhyay et al., 2016; Akhtar et al., 2018; Ruder et al., 2019). Faruqui and Dyer (2014) introduced the BiCCA embedding using bilingual correlation, which performed well on syntactical tasks, but poorly on cross-lingual semantic tasks. Similarly, frameworks proposed by Hermann and Blunsom (2014) and Luong et al. (2015) depend on projecting the words of two languages into a single embedding space.

However, as demonstrated by Pratapa et al. (2018), bilingual embedding techniques are not ideal for CS text processing and should be replaced by multi-lingual embeddings learnt from CM data. The transformer-based Multilingual BERT (Devlin et al., 2019) embedding has been demonstrated (Pires et al., 2019) to possess impressive cross-lingual model transfer capabilities. Also, the XLM model (Conneau and Lample, 2019) has also shown the effects of cross-lingual training for low-resource and CM language tasks.

Another school of thought revolves around sub-word level representations, which can help to capture variations found in CM and transliterated text. Joshi et al. (2016) proposed a CNN-LSTM based model to learn the sub-word embeddings through 1-D convolutions of character inputs. They showed that it resulted in better sentiment classification performance for CM text. On top of this intuition, attention-based frameworks have also been proven to be successful in learning low-level representations. The HAN (Yang et al., 2016) model provides the intuition of hierarchical attention for document classification, which enables it to differentially attend to more and less important content, at the word and sentence levels. In another work, Aguilar and Solorio (2020) proposed CS-ELMo for code-mixed inputs with similar intuition. It utilizes the hierarchical attention mechanism on bi-gram and tri-gram levels to effectively encode the sub-word level representations, while adding positional awareness to it.

Our work builds on top of these two earlier works to push the robustness of code-mixed representations to higher levels. However, the main difference between existing studies and HIT is the incorporation of outer-product attention-based fused attention mechanism (FAME).

3 Proposed Methodology

In this section, we describe the architecture of HIT for learning effective representations in code-

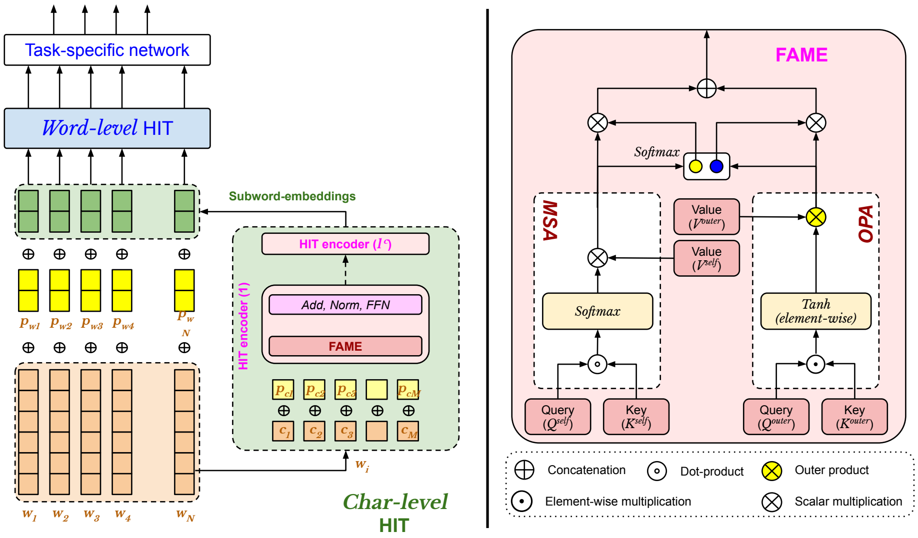

Figure 1: **Hi**erarchical **T**ransformer along with our novel **FAME** mechanism for attention computation.

mixed languages. The backbone of HIT is transformer (Vaswani et al., 2017) and Hierarchical Attention Network (HAN) (Yang et al., 2016). HIT takes a sequence of words (a code-mixed sentence) $S = \langle w_1, w_2, ..., w_N \rangle$ as input and processes each word w_i using a *character-level* HIT to obtain sub-word representation $S^{sb} = \langle sb_1, sb_2, ..., sb_N \rangle$. The *character-level* HIT is a transformer encoder, where instead of computing multi-headed self-attention only, we amalgamate it with an outer-product attention mechanism (Le et al., 2020) as well. The intuition of outer-product attention is to extract higher-order character-level relational similarities among inputs. To leverage both attention mechanisms, we compute their weighted sum using a softmax layer. Subsequently, we pass it through the typical *normalization* and *feed-forward* layers to obtain the encoder's output. A stacking of l_c encoders is used. In the next layer of the hierarchy, these sub-word representations are combined with positional and rudimentary embeddings of each word and forwarded to the *word-level* HIT's encoder. Finally, the output of *word-level* HIT is fed to the respective task-specific network. The hierarchical nature of HIT enables us to capture both *character-level* and *word-level* relational (syntactic and semantic) similarities. A high-level schema of HIT is shown in Figure 1.

3.1 Fused Attention Mechanism (FAME)

FAME extends the multi-headed self-attention (MSA) module of a standard transformer by including a novel outer-product attention (OPA) mechanism. Given an input x, we use three weight matrices, W_Q^{self}, W_K^{self}, and W_V^{self}, to project the input to *Query* (Q^{self}) , *Key* (K^{self}), and *Value* (V^{self}) representations for MSA, respectively. Similarly for OPA we use W_Q^{outer}, W_K^{outer}, and W_V^{outer} for the projecting x to Q^{outer}, K^{outer} and V^{outer}. Next, the two attention mechanisms are learnt in parallel, and a weighted sum is computed as its output. Formally, $H = \alpha_1 \cdot Z_{self} \oplus \alpha_2 \cdot Z_{outer}$, where Z_{self} and Z_{outer} respectively are the outputs of multi-headed self attention and outer-product attention modules, and α_1 and α_2 are the respective weights computed through a softmax function.

Multi-Headed Self Attention. The standard transformer self-attention module (Vaswani et al., 2017) computes a scaled dot-product between the *query* and *key* vectors prior to learn the attention weights for the *value* vector. We compute the output as follows:

$$\begin{aligned} Z_{self} &= softmax\left(\frac{Q^{self} \cdot K^{self^T}}{\sqrt{d^k}}\right) V^{self} \\ &= \sum_{i}^{N} softmax\left(\frac{q \cdot k_i}{\sqrt{d^k}}\right) v_i, \forall q \in Q^{self} \end{aligned}$$

where N is the sequence length, and d^k is the dimension of the *key* vector.

Outer-Product Attention. This is the second attention that we incorporate into FAME. Although the fundamental process of OPA (Le et al., 2020) is similar to the multi-headed self-attention computation, OPA differs in terms of different operators and the use of row-wise *tanh* activation instead of softmax. To compute the interaction between the query and key vectors, we use element-wise multiplication as opposed to the dot-product in MSA. Subsequently, an element-wise *tanh* is applied before computing the outer-product with the value vector. The intuition is to exploit fine-level associations between the *key*-scaled *query* and *value* representations in a code-mixed setup. Similar to the earlier case, we define OPA as:

$$Z_{outer} = \sum_{i}^{N} tanh\left(\frac{q \odot k_i}{\sqrt{d_k}}\right) \otimes v_i, \forall q \in Q^{outer}$$

where $\odot$ is the element-wise multiplication, and $\otimes$ is the outer-product.

3.2 Task-specific Layers

As we mention earlier, HIT can be adapted for various NLP tasks including sequence labelling, classification, or generative problems. In the current work, we evaluate HIT on part-of-speech (POS) tagging, named-entity recognition (NER), sentiment classification, and machine translation (MT). We mention their specific architectural details below.

For the sentiment classification, we apply a *GlobalAveragePooling* operation over the token embeddings to obtain the sentence embeddings. Additionally, we concatenate extracted statistical features along with the embeddings before feeding into the final classification layer. We use *tf-idf* (term frequency–inverse document frequency) vectors for $\{1, 2, 3\}$-grams of words and characters extracted from each text. We hypothesize that these statistical features contain sufficient information to get rid of any handcrafted features like the ones suggested by Bansal et al. (2020). Finally, a *softmax* activation function is used for the prediction. Similarly, for POS tagging and NER, the corresponding labels for each of the token's embedding is obtained through a softmax activated output. In case of MT, we use an encoder-decoder framework where both the encoder and the decoder are based on the HIT framework.

4 Experiments, Results, and Analyses

In this section, we furnish the details of chosen datasets, our experimental results, comparative study, and necessary analyses.

4.1 Datasets

We evaluate 11 publicly available datasets across 4 tasks in 6 code-mixed languages. For POS tagging, we employ Hindi, Telugu, Bengali, and Spanish, whereas, we evaluate Hindi and Spanish datasets for NER. Similarly, in sentiment classification, we incorporate Hindi, Tamil, Malayalam, and Spanish code-mixed sentences. Finally, for machine translation, we use a recently released Hindi-English code-mixed parallel corpus. A brief statistics of all datasets is presented in Table 1.

Tasks	**Lang**	**Train**		**Test**		**Total**	**#Labels**
		#Sent	**#Token**	**#Sent**	**#Token**		
POS	Hi*	1191	6575	148	2300	1489	14
	Te*	1,585	7,190	198	2,927	1,982	52
	Be*	500	4,108	62	631	626	39
	Sp	27,893	11,897	4,298	3,866	36,489	17
NER	Hi*	1663	9,397	207	3,272	2,079	7
	Sp	33,611	52,680	10,085	23,787	53,781	19
Sentiment	Hi*	3,103	9,005	387	3,191	3,879	3
	Ta	11,335	27,476	3,149	10,339	15,744	4
	Ma	4,851	16,551	1,348	6,028	6,739	4
	Sp	12,194	28,274	1,859	7,822	15,912	3
MT	En (Src) Hi (Tgt)	248,330	84,609 108,442	2,000	5,314 5,797	252,330	-

Table 1: Dataset statistics. Star(*) signifies 90-10 ratio.

- **POS tagging:** We use the Hindi-English code-mixed POS dataset provided by Singh et al. (2018b). It was collected from Twitter and has 1489 sentences. Each token in the sentence is tagged with one of the 14 tags[1]. The Bengali and Telugu datasets are collected from ICON-2016 workshop[2]. The instances are the social-media messages, collected from Twitter, Facebook and WhatsApp, and have 1982 and 626 sentences in Telugu and Bengali, respectively. These two datasets follow Google universal tagset (Petrov et al., 2011) and contain 52 and 39 tags respectively. For Spanish, we use Linguistic Code-switching Evaluation (LinCE) POS dataset (AlGhamdi et al., 2016) consisting of more that $35k$ sentences with 14 tags.
- **Sentiment classification:** We explore the Hinglish sentiment classification dataset developed by Joshi et al. (2016). The dataset contains 3879 Facebook public posts comprises of 15% *negative*, 50% *neutral*, and 35% *positive* samples. We further consider two sentiment classification datasets for Dravidian languages *viz.* Tamil and Malayalam (Chakravarthi et al., 2020), containing 15744 and 6739 instances respectively with four sentiment labels – *positive*, *negative*, *neutral*, and *mixed feelings*. Additionally, we use SemEval-2020 (Patwa et al., 2020) dataset for Spanish code-mixed sentiment classification. It supports a classic 3-way sentiment classification.
- **Named-entity recognition:** For NER, we em-

---

[1] We furnish the details of tagset in the appendix

[2] `http://amitavadas.com/Code-Mixing.html`

Model	Hindi			Tamil			Malayalam			Spanish		
	Pr.	**Re.**	**F1**	**Pr.**	**Re.**	**F1**	**Pr.**	**Re.**	**F1**	**Pr.**	**Re.**	**F1**
BiLSTM	0.916	0.901	0.909	0.502	0.428	0.451	0.653	0.588	0.612	0.429	0.431	0.428
Subword-LSTM	0.905	0.907	0.905	0.503	0.418	0.426	0.577	0.592	0.581	0.445	0.437	0.432
HAN	0.915	0.906	0.908	0.490	0.411	0.439	0.639	0.611	0.634	0.449	0.439	0.440
ML-BERT	0.919	**0.914**	0.909	0.260	0.310	0.280	0.600	0.630	0.610	0.451	0.419	0.437
CS-ELMO	0.921	0.903	0.909	0.515	0.432	0.459	0.666	0.623	0.642	0.429	0.453	0.431
HIT	**0.956**	0.914	**0.915**	0.499	**0.451**	**0.473**	0.710	0.628	0.651	**0.502**	**0.454**	**0.460**
(-) Atn^{outer}	0.933	0.911	0.913	**0.520**	0.448	0.455	**0.718**	0.624	**0.655**	0.463	0.440	0.445
(-) *char-level* HIT	0.903	0.887	0.901	0.504	0.418	0.432	0.659	0.605	0.627	0.448	0.438	0.433

Table 2: Performance of HIT on sentiment classification. Best scores are highlighted in bold.

ploy Hindi (Singh et al., 2018c) and Spanish (Aguilar et al., 2018) datasets with 2079 and 52781 sentences, respectively. In Hindi, the labels are *name*, *location*, and *organization*. The Spanish dataset has six additional labels – *event*, *group*, *product*, *time*, *title*, and *other* named entities.

• **Machine Translation:** We utilize a recently developed Hindi-English code-mixed parallel corpus for machine translation (Gupta et al., 2020) comprising more than $200k$ sentence pair. For experiments, we transliterate all Devanagari Hindi text.

4.2 Baselines

POS tagging, NER & sentiment classification: ▷ **BiLSTM** (Hochreiter and Schmidhuber, 1997): It is a weak baseline with two conventional BiLSTM layers. For POS and NER, we additionally incorporate a CRF layer for the final classification. ▷ **HAN** (Yang et al., 2016): We adapt the Hierarchical Attention Network (HAN) for our purpose. The subword embedding is computed at the first level of attention network followed by a word-level attention at the second level. Recently, Bansal et al. (2020) also adopted HAN for code-mixed classification. ▷ **ML-BERT** (Devlin et al., 2019): We fine-tune multilingual BERT (Devlin, 2019). ▷ **CS-ELMo** (Aguilar and Solorio, 2020): It is one of state-of-the-arts on code-mixed languages. It uses pre-trained ELMo (Peters et al., 2018) to transfer knowledge from English to code-mixed languages. ▷ **Subword-LSTM** (Joshi et al., 2016): It is a hybrid CNN-LSTM model. A 1D convolution operation is applied for the subword representation. Subsquently, the convoluted features are max-pooled and fed to an LSTM. Since this system disregards word boundaries in a sentence, we use it for *sentiment classification only*.

Machine translation: For machine translation, we evaluate HIT against **GFF-Pointer** (Gupta et al., 2020), a gated feature fusion (GFF) based approach to amalgamate the XLM and syntactic features during encoding and a Pointer generator for decoding. Furthermore, we also incorporate three other baselines for comparison – **Seq2Seq** (Sutskever et al., 2014), **Attentive-Seq2Seq** (Bahdanau et al., 2014) and **Pointer Generator** (See et al., 2017).

4.3 Experimental Setup

For each experiment, we use a $dropout = 0.1$ in both transformer block and the task specific layers. Categorical cross-entropy loss with Adam ($\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) optimizer (Kingma and Ba, 2014) is employed in all experiments. We train our models for maximum 500 epochs with an early-stopping criteria having $patience = 50$. We additionally use a learning rate scheduler to reduce learning rate to 70% at plateaus with a patience of 20 epochs. All models are trained with *batch-size*$= 32$.

4.4 Experimental Results

We compute precision, recall, F1-score for POS, NER, and sentiment classification, whereas, BLEU, METEOR, and ROUGE scores are reported for the machine translation task.

Sentiment classification: As shown in Table 2, HIT obtains best F1-scores across all languages. For Hindi, three baselines (BiLSTM, ML-BERT, and CS-ELMo) obtain the best F1-score of 0.909, where HIT yields a small improvement with 0.915 F1-score. In comparison, we observe an improvement of 1.4% for Tamil, where HIT and the best baseline (CS-ELMo) report 0.473 and 0.459 F1-scores, respectively. We observe the same pattern for Malayalam and Spanish as well – in both cases, HIT obtains improvements of 0.9% and 2.0%, respectively. For Malayalam, HIT reports 0.651 F1-score, whereas CS-ELMo reports 0.642 F1-score. In case of Spanish, HAN turns out to be the best baseline with 0.440 F1-score. Com-

Model	Hindi			Telugu			Bengali			Spanish		
	Pr.	**Re.**	**F1**	**Pr.**	**Re.**	**F1**	**Pr.**	**Re.**	**F1**	**Pr.**	**Re.**	**F1**
BLSTM-CRF	0.821	0.913	0.782	0.595	0.747	0.572	0.842	0.851	0.817	0.704	0.836	0.680
HAN	0.802	0.879	0.815	0.693	0.701	0.684	0.811	0.823	0.818	0.497	0.629	0.527
ML-BERT	0.833	0.884	0.847	0.802	0.762	0.771	0.793	0.815	0.807	0.853	0.808	0.802
CS-ELMO	0.885	**0.961**	0.910	0.831	0.790	0.775	**0.873**	0.851	0.847	0.740	**0.835**	0.729
HIT	**0.918**	0.955	**0.919**	0.815	0.749	0.762	0.841	**0.855**	**0.853**	**0.871**	0.822	**0.825**
(-) Atn^{outer}	0.893	0.948	0.914	**0.839**	**0.793**	**0.786**	0.839	0.852	0.845	0.859	0.813	0.820
(-) *char-level* HIT	0.686	0.922	0.708	0.629	0.758	0.626	0.802	0.830	0.819	0.723	0.796	0.732

Table 3: Performance of HIT on POS tagging. Best scores are highlighted in bold.

Model	Hindi			Spanish		
	Pr.	**Re.**	**F1**	**Pr.**	**Re.**	**F1**
BLSTM-CRF	0.622	0.781	0.579	0.581	0.659	0.603
HAN	0.721	0.767	0.695	0.615	0.679	0.644
ML-BERT	0.792	0.779	0.714	0.652	0.623	0.643
CS-ELMO	0.815	0.780	0.735	0.683	0.668	0.671
HIT	**0.829**	0.788	**0.745**	**0.695**	**0.671**	**0.684**
(-) Atn^{outer}	0.821	0.767	0.732	0.669	0.663	0.668
(-) *char-level* HIT	0.556	**0.815**	0.528	0.498	0.664	0.539

Table 4: Performance of HIT on NER.

Model	B	R	M
Seq2Seq†	15.49	35.29	23.72
Attentive-Seq2Seq†	16.55	36.25	24.97
Pointer Generator†	17.62	37.32	25.61
GFF-Pointer†	21.55	40.21	28.37
Transformer	21.83	42.19	27.89
HIT	**28.22**	**51.52**	**29.59**
(-) Atn^{outer}	25.95	49.19	27.63

Table 5: Performance of HIT encoder-decoder on En-Hi Translation (**B**: BLEU, **R**: Rouge-L and **M**: METEOR). † Values are taken from Gupta et al. (2020).

paratively, HIT achieves 0.460 F1-score. The last two rows of Table 2 report ablation results – a) excluding outer-product attention (Atn^{outer}) from HIT; and b) excluding sub-word embeddings (*character-level* HIT). In all cases, the absence of *sub-word* embeddings yields negative effect on the performance; hence, suggesting the effectiveness of *character-level* HIT in the architecture. On the other hand, omitting outer-product attention declines F1-scores in 3 out of 4 cases – we observe a margin improvement of 0.04 points for Malayalam. In summary, HIT attains state-of-the-art performance across all four datasets, whereas, the best baseline (CS-ELMo) reports 1.2% lower scores on average.

POS tagging: Table 3 shows the comparative results for POS tagging in Hindi, Telugu, Bengali, and Spanish. Similar to sentiment classification, we observe that HIT attains best F1-scores across three datasets (0.625% better on average). It achieves 0.919, 0.762, 0.853, and 0.825 F1-scores for Hindi, Telugu, Bengali, and Spanish, respectively. In comparison, CS-ELMo yields best F1-scores among all the baselines across three datasets *viz.* Hindi (0.910), Telugu (0.775), and Bengali (0.847). For Spanish, ML-BERT obtains the best baseline F1-score of 0.802. From ablation, we observe the negative effect on performance by removing either the outer-product attention or *character-level* HIT for the majority of the cases.

NER: The performance of HIT for NER is also in-line with the previous two tasks, as show in Table 4. As mentioned earlier, we evaluate HIT for Hindi and Spanish datasets. In both cases, we observe $\geq 1\%$ improvement in F1-score, in comparison with the best baseline (CS-ELMo).

In all three tasks, CS-ELMo is arguably the most consistent baseline. Together with the state-of-the-art performance of HIT, we regard the good performance to the subword-level contextual modelling – both systems use contextual representational models (ELMo and Transformer) to encode the syntactic and semantic features. Moreover, the FAME module in HIT assists in improving the performance even further.

Machine Translation: Finally, Table 5 reports the results for the English to Hindi (En-Hi) machine translation task. For comparison, we also report BLEU, METEOR, and ROUGE-L scores for four baseline systems – Seq2Seq (Sutskever et al., 2014), Attentive-Se2Seq (Bahdanau et al., 2014), Pointer Generator (See et al., 2017), and GFF-Pointer (Gupta et al., 2020). For all three metrics, HIT reports significant improvement (1-9 points) over the state-of-the-art and other baselines. GFF-Pointer obtains 21.55 BLEU score, while the other baselines yield BLEU scores in the range $[15 - 17]$. In comparison, HIT obtain 28.22 BLEU, *an extremely convincing result.* Similarly, HIT reports 51.52 ROUGE and 29.59 METEOR scores, respectively.

Source Tasks		Fine-tune	Target Tasks: PoS	Target Tasks: NER	Target Tasks: Sentiment
	PoS	w/o	0.919	0.702	**0.890**
		w/		0.578	0.863
	NER	w/o	**0.924**	0.745	**0.885**
		w/	0.873		**0.893**
	Sentiment	w/o	**0.936**	0.729	0.871
		w/	**0.928**	0.691	

(a) Hindi code-mixed

Source Tasks		Fine-tune	Target Tasks: PoS	Target Tasks: NER	Target Tasks: Sentiment
	PoS	w/o	0.825	**0.710**	0.417
		w/		0.656	0.419
	NER	w/o	**0.881**	0.648	**0.473**
		w/	0.663		**0.446**
	Sentiment	w/o	**0.918**	**0.969**	0.445
		w/	0.732	0.687	

(b) Spanish code-mixed

Table 6: Transfer learning models. Code-mixed word representations, learned for a (source) task, is utilized for building models for other (target) tasks of same language w/ and w/o fine-tuning. We highlight the cases in bold where transfer learning achieves better performance than original base HIT.

	Pos	Neg	Neu
Pos	**0.85**	0.05	**0.10**
Neg	0.03	**0.87**	**0.10**
Neu	-	0.02	**0.98**

(a) Sentiment

	B-Per	I-Per	B-Loc	I-Loc	B-Org	I-Org	O
B-Per	**0.85**	0.01	-	-	-	-	**0.14**
I-Per	0.03	**0.87**	-	-	-	-	**0.10**
B-Loc	-	-	**0.91**		0.02	0.02	0.05
I-Loc	-	0.08		**0.85**	-	-	0.07
B-Org	0.02	0.02	-	-	**0.82**	-	**0.14**
I-Org	-	-	-	-	-	**0.71**	**0.29**
O	-	-	-	-	**0.47**	-	**0.53**

(b) NER

Table 7: Confusion matrices (in %) for sentiment and NER on Hindi code-mixed dataset[3].

	Input	Gold	Prediction: HIT	Prediction: CS-ELMo
1	**Org:** *safal videsh yatra ke liye badhai ho sir* **Trans:** *Congratulations on the successful foreign trip sir*	Pos	Pos	Neu
2	**Org:** *desh chodo pahaley yeh media ko change karo ... !! ?* **Trans:** *Leave the country, first change the media*	Neu	Neg	Neg

(a) Sentiment

Input	**Org:** *@gurmeetramrahim {dhan dhan satguru}*$_{Per}$ *tera hi aasra #msgloveshumanity salute 2 {msg}*$_{Org}$ *<url>* **Translated:** *@gurmeetramrahim we depend on you {dhan dhan satguru}*$_{Per}$ *#msgloveshumanity salute 2 {msg}*$_{Org}$ *<url>*
Prediction	**HIT:** *@gurmeetramrahim {dhan dhan satguru}*$_{Per}$ *tera hi aasra #msgloveshumanity salute 2 msg <url>* **CS-ELMo:** *@gurmeetramrahim {dhan dhan satguru}*$_{Per}$ *tera hi aasra #msgloveshumanity salute 2 {msg}*$_{Org}$ *<url>*

(b) NER

Table 8: Error Analysis on Hindi code-mixed dataset.

4.5 Effects of Transfer Learning across Tasks

One of the core objectives of representation learning is that the learned representation should be task-invariant – the representations learned for one task should also be (near) effective for other tasks. The intuition is that the syntactic and semantic features captured for a language should be independent of the tasks, and if it does not comply, the representation can be attributed to capture the task-specific feature, instead of linguistic features. To this end, we perform transfer learning experiments with (w/) and without (w/o) fine-tuning. Since we have only one dataset for Tamil, Telugu, Bengali, and Malayalam, we choose Hindi and Spanish code-mixed datasets (POS, NER, and sentiment classification) for the study. Table 6 reports results for both code-mixed languages. For each case, we learn HIT's representation on one (source) task and subsequently utilize the representation for the other two tasks (targets). Moreover, we repeat each experiment with and without fine-tuning HIT.

For Hindi code-mixed, we do not observe the positive effect of transfer learning for NER. It could be because of the limited lexical variations of named-entities in other datasets. However, we obtain the best F1-score (0.936) for POS tagging in a transfer learning setup with sentiment classification. Similarly, for the sentiment classification as target, we observe performance improvements with both POS and NER as source tasks. In Spanish, we also observe increment in F1-scores for all three tasks. We attribute these improvements to the availability of more number of sentences for HIT to leverage the linguistic features in both Hindi and Spanish.

4.6 Error Analysis

In this section, we analyze the performance of HIT both quantitatively and qualitatively. At first, we report the confusion matrices[3] for Hindi NER and sentiment classification in Table 7. In both cases, we observe the *true-positives* to be significant for all labels. Furthermore, we also observe the *false-positives* to be extremely low (except for '*B-Org*'

---

[3] Confusion matrices and more error cases for other tasks are presented in the appendix.

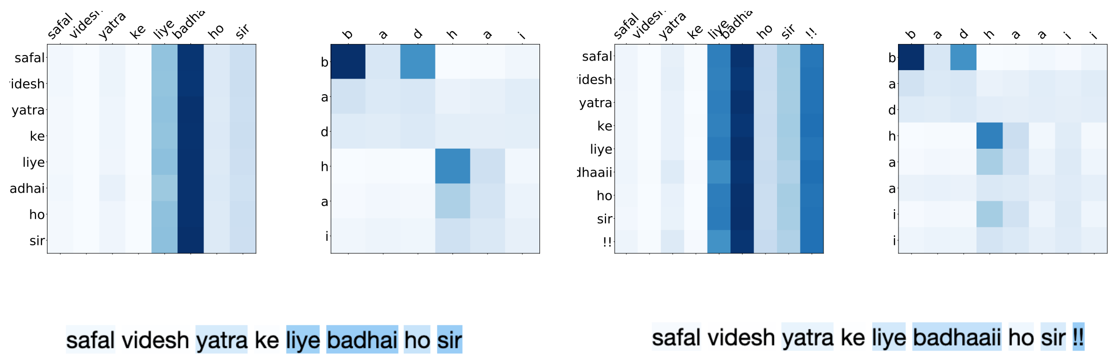

(a) Original input.

(b) Perturbed input.

Figure 2: Interpretation of Hindi code-mixed sentiment prediction (a) Grad-CAM (Selvaraju et al., 2019) analysis of original text; (b) Grad-CAM of perturbed text (*'badhaaii'*); For both case, the *word-level* and *char-level* attention plots are shown.The impact of the word *'badhai'* on the overall prediction is shown through gradients and attention heatmaps. The representation learning of phonetically similar word (e.g., *'badhaaii'*) is also noticeable in the perturbed case. It signifies that HIT is flexible to the spelling variations, a common feature in code-mixed environment.

in NER) for majority of the cases – suggesting very good precision in general. The major contribution in error is due to the *neutral* and *other* classes in sentiment and NER, respectively. In sentiment analysis, 10% of the *positive* and *negative* labels each were mis-classified as *neutral*. Similarly in NER, we observe that the *organization* entities (*B-Org & I-Org*) and *other* classes confuse with each other in a significant number of samples. It may be due to the under-represented ($\sim$ 13%) *organization* entities in the dataset.

We also perform qualitative error analysis of HIT and CS-ELMo. Table 8 reports the results for the NER and sentiment classification tasks[3]. For the first example in sentiment classification, HIT accurately predicts the sentiment labels as *positive*; however in comparison, CS-ELMo mis-classifies it as *neutral*. For the second example, both HIT and CS-ELMo wrongly predict the sentiment as *neutral*. One possible reason could be the presence of the negatively-inclined word *chodo* (*leave*) in the sentence. For NER, the sentence has two entities (one *person* and one *organization*). While HIT correctly identifies '*dhan dhan satguru*' as person, it could not recognize '*msg*' as organization. On the other hand, CS-ELMo correctly identifies both.

Furthermore, we take the first example of sentiment analysis (from Table 8) to get the insight of HIT. It is not hard to understand that the most positive vibe is attributed by the phrase '*badhai ho sir*' (*congratulations sir*). To validate our hypothesis, we use a gradient based interpretation technique, Grad-CAM (Selvaraju et al., 2019), which uses gradients of neural networks to show the effect of neurons on the final output. Due to hierarchical and modular nature of HIT, we are able to extract the intermediate word level representations learnt by the *character-level* HIT and compute the gradient of loss of the actual class considering these representations. The magnitude of gradient shows the impact of each word on the final output. Figure 2a shows the word-level and character-level gradient maps for the original input. We can observe that HIT attends to the most important component in both cases. For *word-level*, it highlights the positive phrase '*liye badhai*' (*congratulations on*). Moreover, character-level HIT attends to the two syllables '*b*' and '*dh*' in the word '*badhai*' (*congratulation*). It suggests that both the *word-level* and *character-level* components are capable of extracting important features from inputs. Furthermore, to check the robustness, we investigate HIT on a perturbed input. In the previous example, we tweak the spelling of the most important word '*badhai*' to '*badha$\underline{a}\underline{i}$i*' (an out-of-vocabulary word considering the dataset). Figure 2b shows similar patterns in the perturbed input case as well. It signifies that HIT identifies the phonetic similarity of the two words and is flexible to the spelling variants, a common feature in code-mixed environment.

4.7 HIT's Performance on Monolingual Data

In this section, we outline the performance of HIT for monolingual and low-resource settings. We consider the sentiment classification dataset curated by Akhtar et al. (2016), containing 5417 transliterated

Model	Hi Sentiment			Magahi POS		
	Pr.	Re.	F1	Pr.	Re.	F1
BiLSTM	0.619	0.533	0.554	0.594	0.804	0.626
HAN	0.602	0.528	0.551	0.729	0.857	0.649
ML-BERT	0.604	0.556	0.576	0.757	0.867	0.708
CS-ELMO	0.593	0.520	0.542	0.771	0.884	0.759
HIT	**0.641**	**0.629**	**0.635**	**0.783**	**0.913**	**0.775**

Table 9: Performance of HIT on monolingual tasks. Best scores are highlighted in bold.

Tasks	Model	# Params(M)	Train Time(s/ep)
Sentiment	CS-ELMo	2.9	18
	HAN	2.7	14
	BiLSTM	2.8	12
	ML-BERT	179.2	172
	HIT	2.7	13
POS	CS-ELMo	2.4	4
	HAN	1.4	3
	BiLSTM-CRF	1.5	2
	ML-BERT	177.9	114
	HIT	1.4	2

Table 10: Parameters and Runtime: Number of trainable parameters and training runtime (second/epoch) for the Hi-En PoS and sentiment classification tasks.

Hindi reviews with 4 sentiment labels - *positive*, *negative*, *neutral*, and *conflict*. We also utilize a Magahi POS dataset (Kumar et al., 2012), annotated with 33 tags from the BIS-tagset [4]. We report the performance of HIT and other baselines on these two datasets in Table 9. For the Hindi sentiment classification task, we observe that HIT yields an F1-score of 0.635, which is better than CS-ELMo and ML-BERT by 9.3% and 5.9%. Also, for Magahi POS, HIT reports the best F1-score of 0.775 – increaments of $+2.1\%$ and $+9.5\%$ over CS-ELMo and ML-BERT, respectively. These results suggest that HIT is capable of handling monolingual and low-resource texts in an efficient manner.

4.8 Learnable Parameters and Power Usage

We conduct all our experiments on 1 Tesla T4 GPU. In Table 10, we report the total trainable parameters for HIT and other baselines. We observe that HIT requires a comparable number of parameters. For instance, in the Hindi-English sentiment analysis task (sequence classification), HIT has a total ~$2.7M$ trainable parameters, while other baselines such as, CS-ELMo, HAN, Subword-LSTM, and BiLSTM require ~$2.9M$, ~$2.7M$, ~$2.1M$, and ~$2.8M$ parameters, respectively. ML-BERT has a whopping ~$179.2M$ parameters. Similarly, in Hindi-English POS tagging, the number of parameters for HIT is comparable (or even lesser) – HIT: ~$1.4M$, CS-ELMo: ~$2.4M$, HAN: ~$1.4M$, BiLSTM-CRF: ~$1.5M$, ML-BERT: ~$177.9M$. We observed similar distribution for other tasks/languages as well.

We further note that HIT is significantly more efficient than the current SOTA models as it takes 13 s/epoch to train which is significantly lower than CS-ELMo (18 s/epoch), HAN (14 s/epoch), and ML-BERT (172 s/epoch), while it takes a bit more time compared to BiLSTM (12 s/epoch) and Subword-LSTM (7 s/epoch). We also computed the amount of power consumption for training HIT for a maximum 500 epochs. Following the guidelines of Strubell et al. (2019), we estimate a total power consumption of 0.383 kWh and equivalent CO2 emission of 0.365 pounds.

5 Conclusion

In this work, we present HIT – a hierarchical transformer-based framework for learning robust code-mixed representations. HIT contains a novel fused attention mechanism, which calculates a weighted sum of the multi-headed self attention and outer-product attention, and is capable of capturing relevant information at a more granular level. We experimented with eleven code-mixed datasets for POS, NER, sentiment classification, and MT tasks across six languages. We observed that HIT successfully outperforms existing SOTA systems. We also demonstrate the task-invariant nature of the representations learned by HIT via a transfer learning setup, signifying it's effectiveness in learning linguistic features of CM text rather than task-specific features. Finally, we qualitatively show that HIT successfully embeds semantically and phonetically similar words of a code-mixed language.

Acknowledgement

The work was partially supported by the Ramanujan Fellowship (SERB) and the Infosys Centre for AI, IIITD.

References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia. Association for Computational Linguistics.

---

[4] `https://thottingal.in/blog/2019/09/10/bis-pos-tagset-review/`

Gustavo Aguilar and Thamar Solorio. 2020. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, Online. Association for Computational Linguistics.

Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. Aspect based sentiment analysis in Hindi: Resource creation and evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2703–2709, Portorož, Slovenia. European Language Resources Association (ELRA).

Md Shad Akhtar, Palaash Sawant, Sukanta Sen, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Solving data sparsity for aspect based sentiment analysis using cross-linguality and multi-linguality. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 572–582, New Orleans, Louisiana. Association for Computational Linguistics.

Fahad AlGhamdi, Giovanni Molina, Mona Diab, Thamar Solorio, Abdelati Hawwari, Victor Soto, and Julia Hirschberg. 2016. Part of speech tagging for code switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 98–107, Austin, Texas. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Srijan Bansal, Vishal Garimella, Ayush Suhane, Jasabanta Patro, and Animesh Mukherjee. 2020. Code-switching patterns can be an effective route to improve performance of downstream NLP applications: A case study of humour, sarcasm and hate speech detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1018–1023, Online. Association for Computational Linguistics.

Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John Philip McCrae. 2020. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 202–210, Marseille, France. European Language Resources association.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7059–7069. Curran Associates, Inc.

Jacob Devlin. 2019. Multilingual bert.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden. Association for Computational Linguistics.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2267–2280, Online. Association for Computational Linguistics.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68, Baltimore, Maryland. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Ritesh Kumar, Bornini Lahiri, and Deepak Alok. 2012. Developing a POS tagger for Magahi: A comparative study. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 105–114, Mumbai, India. The COLING 2012 Organizing Committee.

Hung Le, Truyen Tran, and Svetha Venkatesh. 2020. Self-attentive associative memory. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5682–5691. PMLR.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver, Colorado. Association for Computational Linguistics.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Dan Garrette, Bj"orn Gamb"ack, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. SemEval-2020 Sentimix Task 9: Overview of SENTIment Analysis of Code-MIXed Tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2011. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. 2018. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3067–3072, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *J. Artif. Int. Res.*, 65(1):569–630.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2019. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018a. Language identification and named entity recognition in Hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58, Melbourne, Australia. Association for Computational Linguistics.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018b. A Twitter corpus for Hindi-English code mixed POS tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 12–17, Melbourne, Australia. Association for Computational Linguistics.

Vinay Singh, Deepanshu Vijay, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018c. Named entity recognition for hindi-english code-mixed social media text. In *Proceedings of the seventh named entities workshop*, pages 27–35.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Advances in neural information processing systems. In *Neural Information Processing Systems (Nips)*, pages 3104–3112.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1670, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

A Appendix

A.1 Semantic Understanding of Languages

In this section, we study the semantic relationships between different Indic languages. We calculate the proportion of common words in Table 11 between different language pairs to understand the multilingualism in India. We observe that Bengali code-mixed texts have the highest proportion of English words 32% as compared to other languages. Moreover, 50% of all Bengali words are also present in the Hindi CM texts, although 58% of those words are English. We observe that users using Hindi CM texts use very few words taken from other languages. On the other hand, a significant proportion of Bengali and Telugu CM words are common in other languages, although, majority of them are English. The two Dravidian languages, Tamil and Malayalam, show a very distinctive behavior. They share very little linguistic similarity with other Indic languages. On the other hand, 10% of all Tamil words are used in Malayalam and 17% of all Malayalam words are used in Tamil. Moreover, this sharing is not driven by English, as, only 27% of these words are English, which is the lowest proportion among all other language pairs. Being originate from a similar root and having a phonetic resemblance makes Tamil and Malayalam *sister languages*[5]. Similar observations are also made from the word representation lens. We use t-SNE (Van der Maaten and Hinton, 2008) plots to embed HIT's representations onto a 2-D space for interpretability (Fig 3). Although, the embeddings are well clustered based on the languages, we can easily figure out the semantically similar words across languages embedded onto a similar space. Furthermore, Fig 3(b) shows that pronouns (e.g., '*aap*') in Tamil, Telegu and Hindi are embedded onto a similar space with Bengali words *'aamar'*, '*aamay*'. Although each of these representations are learned on separate models on separate datasets, the robustness of the underlying hierarchical representation enables our model to capture cross-lingual semantics from noisy code-mixed texts. We can attribute these observations to the relatedness of Indic languages on a socio-cultural basis.

A.2 Datasets

We report all available POS tags in Table 12.

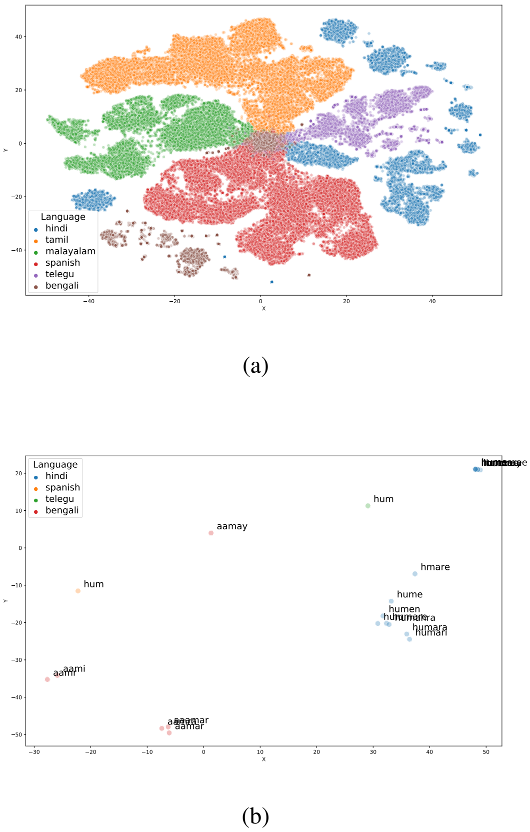

(a)

(b)

Figure 3: t-SNE visualization (a) of all words; (b) of selected pronouns. Overlapping clusters show how semantically similar words from different languages are embedded onto a similar space.

A.3 Confusion Matrices and Error Analysis

We report the confusion matrices to show the label-wise performance for the sentiment classification, PoS tagging and NER in Tables 5, 4, and 6, respectively.

We similarly perform qualitative analysis on the MT task where our model shows superior performance as compared to the baselines. In example 1 of Table 13 (d), HIT translates the English text '*'Licencing and import policies were liberalise"* to *"Licencing aur policies liberal the* ।*"*. Although this prediction has very low BLEU score when evaluated against the target, this example shows an interesting observation. The overall translation is a contextually meaningful sentence in Hindi. Further HIT translates the phrase *'were liberalise'* to *'liberal the'*. In Hindi, *'the'* represents past tense. Another interesting observation is the ability of HIT to copy texts from source to predicted text. Even without having an explicit copying mechanism (See et al., 2017), HIT is able to understand

---

[5] https://royalsocietypublishing.org/doi/10.1098/rsos.171504

		Target Language					
		Hindi (English)	**Malayalam (English)**	**Tamil (English)**	**Bengali (English)**	**Telegu (English)**	**Spanish (English)**
Source Language	Hindi	1.00 (0.16)	0.02 (0.41)	0.04 (0.39)	0.02 (0.58)	0.02 (0.57)	0.07 (0.62)
	Malayalam	0.14 (0.41)	1.00 (0.06)	0.17 (0.27)	0.03 (0.71)	0.05 (0.57)	0.07 (0.64)
	Tamil	0.15 (0.39)	0.10 (0.27)	1.00 (0.07)	0.03 (0.69)	0.05 (0.56)	0.07 (0.64)
	Bengali	0.50 (0.58)	0.16 (0.58)	0.23 (0.69)	1.00 (0.32)	0.21 (0.71)	0.36 (0.72)
	Telegu	0.36 (0.57)	0.15 (0.57)	0.29 (0.56)	0.12 (0.71)	1.00 (0.22)	0.28 (0.65)
	Spanish	0.12 (0.62)	0.02 (0.64)	0.03 (0.64)	0.02 (0.72)	0.03 (0.65)	1.00 (0.11)

Table 11: Proportion of words in source language in the target language.

Lang	**POS tags**
Hindi (14)	*X, VERB, NOUN, ADP, PROPN, ADJ, PART, PRON, DET, ADV, CONJ, PART_NEG, PRON_WH, NUM*
Bengali (39)	*N_NN, V_VM, RD_PUNC, N_NNP, PSP, PR_PRP, JJ, RB_AMN, CC, QT_QTF, DM_DMD, RP_RPD, @, RD_RDF, V_VAUX, DT, PR_PRQ, #, RP_NEG, E, $, RB_ALC, N_NNV, PR_PRL, N_NST, RP_INJ, RD_SYM, DM_DMR, RP_INTF, PR_PRF, DM_DMQ, QT_QTO, U, QT_QTC, PR_PRC, RD_ECH, QY_QTO, Ã°Å¸Ëœ, ∼*
Telugu (52)	*N_NN, N_NNP, RD_RDF, RD_PUNC, V_VM, JJ, @, PSP, PR_PRP, RP_INJ, DT, RB_AMN, CC, $, U, E, #, N_NNV, &, PR_PRQ, V_VAUX, RD_PUNC", ∼, RD_RDFP, QT_QTF, RD_UNK, DM_DMD, RP_RPD, RB_ALC, DM_DMQ, RD_ECH, N_NST,* acro, *PR_PRL, QT_QFC, RP_RDF, PR_PRC, r, RD_SYM, RD_RDFF,* psp, *PR_PRF, QT_QTP, RD_P/UNC, PR_PPR, PR_RPQ, RPR_PRP, RP_INTF, -*
Spanish (17)	*VERB, PUNCT, PRON, NOUN, DET, ADV, ADP, INTJ, CONJ, ADJ, AUX, SCONJ, PART, PROPN, NUM, UNK, X*

Table 12: POS tagsets for different datasets.

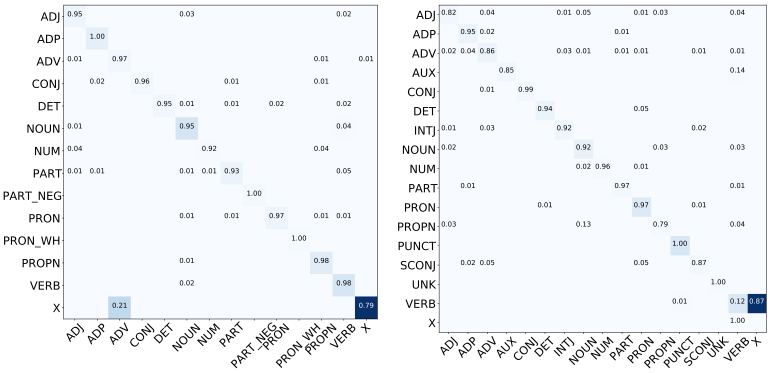

(a) Hindi

(b) Spanish

Figure 4: Confusion matrices of `HIT` on POS tasks. Due to high cardinality of output classes, we do not report for Bengali and Telugu.

the key phrases that co-occur in both Hindi and English, like, numeric and proper nouns and copies these tokens while generating. This shows how our model can also be used in conditional generation of texts. It also ends the sentence with |, which is a common punctuation widely used as a full stop in Hindi texts.

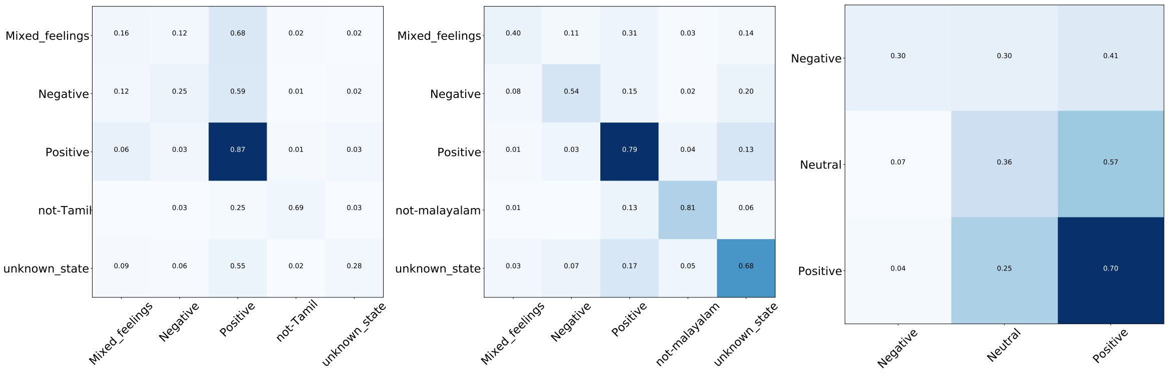

(a) Tamil (b) Malayalam (c) Spanish

Figure 5: Confusion matrices of `HIT` on sentiment tasks.

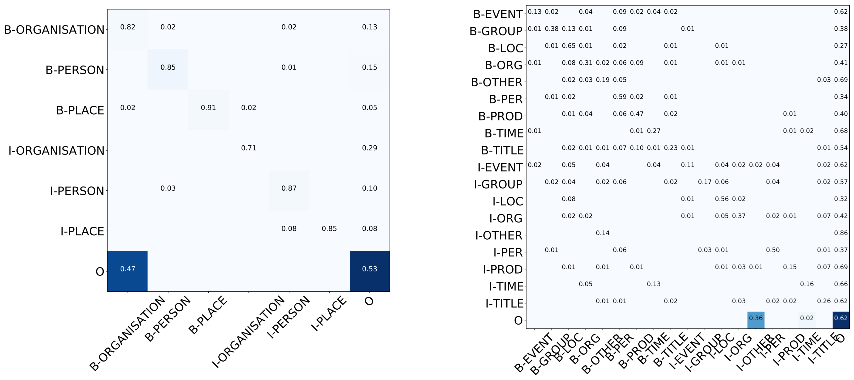

(a) Hindi (b) Spanish

Figure 6: Confusion matrices of `HIT` on NER.

	Input	Sys	**Prediction**
1	**Org:** *#surgicalstrike_X #pakistan_X will_V not_Neg sleep_N in_ADP peace_N tonight_N ._X khamoshi_V toofan_V ke_ADP aane_V ki_ADP aahat_N to_P nahi_Neg*	A	*#surgicalstrike_X #pakistan_X will_V not_part_Neg sleep_N in_ADP peace_N tonight_N ._X, khamoshi_V toofan_V ke_ADP aane_V ki_ADP aahat_N to_P nahi_part_neg*
	Translated: *#surgicalstrike #pakistan will not sleep in peace tonight. Does this silence signify that a storm is approaching*	B	*#surgicalstrike_X #pakistan_X will_V not_part_Neg sleep_N in_ADP peace_N tonight_N ._X, khamoshi_V toofan_V ke_N aane_V ki_ADP aahat_ADP to_P nahi_part_Neg*
2	**Org:** *minimum_N cincuenta_Num mil_Num por_ADP persona_N ._Punct*	A	*minimum_N cincuenta_Num mil_Num por_ADP persona_N ._Punct*
	Translated: *minimum fifty thousand per person .*	B	*minimum_N cincuenta_Num mil_Num por_ADP persona_N ._Punct*

(a) PoS

	Input	Sys	**Prediction**
1	**Org:** *@gurmeetramrahim {dhan dhan satguru}$_{Per}$ tera hi aasra #msgloveshumanity salute 2 {msg}$_{Org}$ <url>*	A	*@gurmeetramrahim {dhan dhan satguru}$_{Per}$ tera hi aasra #msgloveshumanity salute 2 msg <url>*
	Translated:	B	*@gurmeetramrahim {dhan dhan satguru}$_{Per}$ tera hi aasra #msgloveshumanity salute 2 {msg}$_{Org}$ <url>*
2	**Org:** *ste {sábado}$_{Time}$ nuestras alumnas en {imagen modeling}$_{Org}$ by {la gatita}$_{Per}$ reciben la visita de {monic abbad}$_{Per}$, joven . . . <url>*	A	*ste {sábado}$_{Time}$ nuestras alumnas en imagen modeling by la gatita reciben la visita de {monic}$_{Per}$ abbad , joven . . . <url>*
	Translated: *This saturday our students in image modeling by the kitten receive a visit from young monic abbad*	B	*ste {sábado}$_{Time}$ nuestras alumnas en imagen modeling by la gatita reciben la visita de {monic abbad}$_{Per}$, joven . . . <url>*

(b) NER

	Input	**Gold**	**Prediction**	
			A	**B**
1	**Org:** *safal videsh yatra ke liye badhai ho sir* **Trans:** *Congratulations on the successful foreign trip sir*	Pos	Pos	Neu
2	**Org:** *nunca pensé que " bruh " me frustraría tanto* **Trans:** *I never thought that "bruh" would frustrate me so much*	Neu	Neu	Neg
3	**Org:** *desh chodo pahaley yeh media ko change karo ... !! ?* **Trans:** *Leave the country, first change the media*	Neu	Neg	Neg

(c) Sentiment

1	**Source:** *Licencing and import policies were liberalise* **Reference:** *license tatha import ki policies ko udar banaya gaya*
	HIT: Licencing aur policies liberal the |
2	**Source:** *This fact is based on possibility* **Reference** *yah fact possibility par aadharit hai |*
	HIT: yah fact possibility par aadharit hai

(d) MT

Table 13: Error Analysis. System A denotes HIT and B denotes CS-ELMO.

Semi-Supervised Data Programming with Subset Selection

Ayush Maheshwari [1], Oishik Chatterjee [1], Krishnateja Killamsetty [2],
Ganesh Ramakrishnan [1] , Rishabh Iyer [2]
[1]Department of CSE, IIT Bombay, India
[2] The University of Texas at Dallas
{ayusham, oishik, ganesh}@cse.iitb.ac.in
{krishnateja.killamsetty, rishabh.iyer}@utdallas.edu

Abstract

The paradigm of data programming, which uses weak supervision in the form of rules/labelling functions, and semi-supervised learning, which augments small amounts of labelled data with a large unlabelled dataset, have shown great promise in several text classification scenarios. In this work, we argue that by not using any labelled data, data programming based approaches can yield sub-optimal performances, particularly when the labelling functions are noisy. The first contribution of this work is an introduction of a framework, SPEAR which is a semi-supervised data programming paradigm that learns a *joint model* that effectively uses the rules/labelling functions along with semi-supervised loss functions on the feature space. Next, we also study SPEAR-SS which additionally does subset selection on top of the joint semi-supervised data programming objective and *selects* a set of examples that can be used as the labelled set by SPEAR. The goal of SPEAR-SS is to ensure that the labelled data can *complement* the labelling functions, thereby benefiting from both data-programming as well as appropriately selected data for human labelling. We demonstrate that by effectively combining semi-supervision, data-programming, and subset selection paradigms, we significantly outperform the current state-of-the-art on seven publicly available datasets. [1]

1 Introduction

Modern machine learning techniques rely on large amounts of labelled training data for text classification tasks such as spam detection, (movie) genre classification, sequence labelling, *etc*. Supervised learning approaches have utilised such large amounts of labelled data and, this has resulted in huge successes in the last decade. However, the acquisition of labelled data, in most cases, entails a painstaking process requiring months of human effort. Several techniques such as active learning, distant supervision, crowd-consensus learning, and semi-supervised learning have been proposed to reduce the *annotation cost* (Settles et al., 2008). However, clean annotated labels continue to be critical for reliable results (Bach et al., 2019; Goh et al., 2018).

Recently, Ratner et al. (2016) proposed a paradigm on data-programming in which several Labelling Functions (LF) written by humans are used to weakly associate labels with the instances. In data programming, users encode the weak supervision in the form of labelling functions. On the other hand, traditional semi-supervised learning methods combine a small amount of labelled data with large unlabelled data (Kingma et al., 2014). In this paper, we leverage semi-supervision in the feature space for more effective data programming using labelling functions.

1.1 Motivating Example

We illustrate the LFs on one of the seven tasks on which we experiment with, *viz.*, identifying spam/no-spam comments in the YouTube reviews. For some applications, writing LFs is often as simple as using keyword lookups or a regex expression. In this specific case, the users construct heuristic patterns as LFs for classifying spam/not-spam comments. Each LF takes a comment as an input and provides a binary label as the output; +1 indicates that the comment is spam, -1 indicates that the comment is not spam, and 0 indicates that the LF is unable to assert anything for the comment (referred to as an *abstain*). Table 1 presents a few example LFs for spam and non-spam classification.

In isolation, a particular LF may neither be always correct nor complete. Furthermore, the LFs

---

[1] The source code is available at https://github.com/ayushbits/Semi-Supervised-LFs-Subset-Selection

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4640–4651
August 1–6, 2021.

Id	**Description**
LF1	If `http` or `https` in comment text, then return +1 otherwise ABSTAIN (return 0)
LF2	If length of comment is less than 5 words, then return -1 otherwise ABSTAIN (return 0).(Non spam comments are often short)
LF3	If comment contains `my channel` or `my video`, then return +1 otherwise ABSTAIN (return 0).

Table 1: Three LFs based on keyword lookups or regex expression for the *YouTube* spam classification task

Training data		LF outputs		Features	
id	Label	LF1(+1)	LF2(-1)	F1	F2
S_1	+1	1	0	1	0
S_2	-1	0	1	0	1
S_3	-1	0	0	0	1
S_4	+1	1	1	1	0
Test data					
S_5	-1	0	1	0	1
S_6	+1	0	0	1	0

Table 2: Example illustrating the insufficiency of using data programming using only LFs.

may also produce conflicting labels. In the past, generative models such as Snorkel (Ratner et al., 2016) and CAGE (Chatterjee et al., 2020) have been proposed for consensus on the noisy and conflicting labels assigned by the discrete LFs to determine the probability of the correct labels. Labels thus obtained could be used for training any supervised model/classifier and evaluated on a test set. We will next highlight a challenge in doing data programming using only LFs that we attempt to address. For each of the following sentences $S_1 \ldots S_6$ that can constitute *an observed set of training instances*, we state the value of the true label (± 1). While the candidates in S_1 and S_4 are instances of a spam comment, the ones in S_2 and S_3 are not. In fact, these examples constitute one of the canonical cases that we discovered during the analysis of our approach in Section 4.4.

1. $\langle \boldsymbol{S_1}, +1 \rangle$: Please help me go to college guys! Thanks from the bottom of my heart. https://www.indiegogo.com/projects/
2. $\langle \boldsymbol{S_2}, -1 \rangle$: I love this song
3. $\langle \boldsymbol{S_3}, -1 \rangle$: This song is very good... but the video makes no sense...
4. $\langle \boldsymbol{S_4}, +1 \rangle$: https://www.facebook.com/teeLaLaLa

Further, let us say we have a completely *unseen set of test instances*, S_5 and S_6, whose labels we would also like to predict effectively:

5. $\langle \boldsymbol{S_5}, -1 \rangle$: This song is prehistoric
6. $\langle \boldsymbol{S_6}, +1 \rangle$: Watch Maroon 5's latest ... www.youtube.com/watch?v=TQ046FuAu00

In Table 2, we present the outputs of the LFs as well as some n-gram features F1 ('.com') and F2 ('This song') on the observed training examples S_1, S_2, S_3 and S_4 as well as on the unseen test examples S_5 and S_6. For S_1, the correct consensus can easily be performed to output the true label +1, since LF1 (designed for class +1) gets triggered, whereas LF2 (designed for class -1) is not triggered. Similarly, for S_2, LF2 gets triggered whereas LF1 is not, making it possible to easily perform the correct consensus. Hence, we have treated S_1 and S_2 as unlabelled, indicating that we could learn a model based on LFs alone without supervision if all we observed were these two examples and the outputs of LF1 and LF2. However, the correct consensus on S_3 and S_4 is challenging since both LF1 and LF2 either fire or do not. While the (n-gram based) features F1 and F2 appear to be informative and could potentially complement LF1 and LF2, we can easily see that correlating feature values with LF outputs is tricky in a completely unsupervised setup. To address this issue, we ask the following questions:

(A) What if we are provided access to the true labels of a small subset of instances - in this case, only S_3 and S_4? Could the (i) correlation of features values (*eg.* F1 and F2) with labels (*eg.* +1 and -1 respectively), modelled via a small set of labelled instances (*eg.* S_3 and S_4), in conjunction with (ii) the correlation of feature values (*eg.* F1 and F2) with LFs (*eg.* LF1 and LF2) modelled via a potentially larger set of unlabelled instances (*eg.* S_1, S_2), help improved prediction of labels for hitherto unseen test instances S_5 and S_6?

(B) Can we precisely determine the subset of the unlabelled data that, when labelling would help us train a model (in conjunction with the labelling functions) that is most effective on the test set? In other words, instead of randomly choosing the labelled dataset for doing semi-supervised learning (part A), can we intelligently select the labelled subset? In the above example, choosing the labelled set as S_3, S_4 would be much more useful than choosing the labelled set as S_1, S_2.

As a solution to (A), in Section 3.3, we present a new formulation, SPEAR, in which the parameters over features and LFs are jointly trained in a semi-supervised manner. SPEAR expands as **S**emi-su**P**ervis**E**d d**A**ta p**R**ogramming. As for (B), we present a subset selection recipe, SPEAR-SS (in

Section 3.4), that recommends the sub-set of the data (*e.g.* S_3 and S_4), which, after labelling, would most benefit the joint learning framework.

1.2 Our Contributions

We summarise our main contributions as follows: To address (A), we present SPEAR (*c.f.*, Section 3.3), which is a novel paradigm for jointly learning the parameters over features and labelling functions in a semi-supervised manner. We jointly learn a parameterized graphical model and a classifier model to learn our overall objective. To address (B), we present SPEAR-SS (*c.f.*, Section 3.4), which is a subset selection approach to *select* the set of examples which can be used as the labelled set by SPEAR. We show, in particular, that through a principled data selection approach, we can achieve significantly higher accuracies than just randomly selecting the seed labelled set for semi-supervised learning with labelling functions. Moreover, we also show that the automatically selected subset performs comparably or even better than the hand-picked subset by humans in the work reported by Awasthi et al. (2020), further emphasising the benefit of subset selection for semi-supervised data programming. Our framework is agnostic to the underlying network architecture and can be applied using different underlying techniques without a change in the meta-approach. Finally, we evaluate our model on seven publicly available datasets from domains such as spam detection, record classification, and genre prediction and demonstrate significant improvement over state-of-the-art techniques. We also draw insights from experiments in synthetic settings (presented in the appendix).

2 Related Work

Data Programming and Unsupervised Learning: Snorkel (Ratner et al., 2016) has been proposed as a generative model to determine correct label probability using consensus on the noisy and conflicting labels assigned by the discrete LFs. Chatterjee et al. (2020) proposed a graphical model, CAGE, that uses continuous-valued LFs with scores obtained using soft match techniques such as cosine similarity of word vectors, TF-IDF score, distance among entity pairs, *etc*. Owing to its generative model, Snorkel is highly sensitive to initialisation and hyper-parameters. On the other hand, the CAGE model employs user-controlled quality guides that incorporate labeller intuition into the model. However, these models completely disregard feature information that could provide additional information to learn the (graphical) model. These models try to learn a combined model for the labelling functions in an unsupervised manner. However, in practical scenarios, some labelled data is always available (or could be made available by labelling a few instances); hence, a completely unsupervised approach might not be the best solution. In this work, we augment these data programming approaches by designing a semi-supervised model that incorporates feature information and LFs to learn the parameters jointly. Hu et al. (2016) proposed a student-teacher model that transfers rule information by assigning linear weight to each rule based on an agreement objective. The model we propose in this paper jointly learns parameters over features and rules in a semi-supervised manner rather than just weighing their outputs and can therefore be more expressive.

Other works such as Jawanpuria et al. (2011); Dembczyński et al. (2008) discovers simple and conjuctive rules from input features and assign weight to each rules for better generalization. Nagesh et al. (2012) induces rules from query language to build NER extractor while Kulkarni et al. (2018) uses active learning to derive consensus among labelers to label data.

Semi-Supervised Data Programming: The only work which, to our knowledge, combines rules with supervised learning in a joint framework is the work by Awasthi et al. (2020). They leverage both rules and labelled data by associating each rule with exemplars of correct firings (*i.e.*, instantiations) of that rule. Their joint training algorithms denoise over-generalized rules and train a classification model. Our approach differs from their work in two ways: a) we do not have information of rule exemplars - thus our labelled examples need not have any correspondence to any of the LFs (and may instead complement the LFs as illustrated in Table 2) and b) we employ a semi-supervised framework combined with graphical model for consensus amongst the LFs to train our model. We also study how to automatically select the seed set of labelled data, rather than having a human provide this seed set, as was done in (Awasthi et al., 2020).

Data Subset Selection: Finally, another approach that has been gaining a lot of attention recently is data subset selection. The specific application of data subset selection depends on the goal at hand. Data subset selection techniques have been used to reduce end to end training time (Mirzasoleiman et al., 2019; Kaushal et al., 2019; Killamsetty et al., 2021) and to select unlabelled points in an active learning manner to label (Wei et al., 2015; Sener and Savarese, 2017) or for topic summarization (Bairi et al., 2015). In this paper, we present a framework (SPEAR-SS) of data subset selection for *selecting* a subset of unlabelled examples for obtaining labels complementary to the labelling functions.

3 Methodology

3.1 Problem Description

Let $\mathcal{X}$ and $\mathcal{Y} \in \{1...K\}$ be the feature space and label space, respectively. We also have access to m labelling functions (LF) λ_1 to λ_m. As mentioned in Section 1.1, each LF λ_j is designed to record some class; let us denote[2] by $k_j \in \{1...K\}$, the class associated with λ_j. The dataset consists of 2 components, *viz.*,

1. $\mathcal{L} = \{(\mathbf{x}_1, y_1, \mathbf{l}_1), (\mathbf{x}_2, y_2, \mathbf{l}_2), \ldots, (\mathbf{x}_N, y_N, \mathbf{l}_N)\}$, which denotes the labelled dataset and

2. $\mathcal{U} = \{(\mathbf{x}_{N+1}, \mathbf{l}_{N+1}), (\mathbf{x}_{N+2}, \mathbf{l}_{N+2}), \ldots, (\mathbf{x}_M, \mathbf{l}_M)\}$, which denotes the unlabelled dataset wherein $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.

Here, the vector $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})$ denotes the firings of all the LFs on instance $\mathbf{x}_i$. Each l_{ij} can be either 1 or 0. $l_{ij} = 1$ indicates that the LF λ_j has fired on the instance i and 0 indicates it has not. All the labelling functions are discrete; hence, no continuous scores are associated with them.

3.2 Classification and Labelling Function Models

SPEAR has a feature-based classification model $f_\phi(\mathbf{x})$ which takes the features as input and predicts the class label. Examples of $f_\phi(\mathbf{x})$ we consider in this paper are logistic regression and neural network models. The output of this model is $P^f_\phi(y|\mathbf{x})$, *i.e.*, the probability of the classes given the input features. This model can be a simple classification model such as a logistic regression model or a simple neural network model.

Notation	Description
f_ϕ	The feature-based Model
P^f_ϕ	The label probabilities as per the feature-based model f_ϕ
P_θ	The label probabilities as per the LF-based Graphical Model
L_{CE}	Cross Entropy Loss: $L_{CE}\left(P^f_\phi(y\|\mathbf{x}), \tilde{y}\right) = -\log\left(P^f_\phi(y = \tilde{y}\|\mathbf{x})\right)$
H	Entropy function : $H(P^f_\phi(y\|\mathbf{x})) = -\sum_{\hat{y}} P^f_\phi(y = \hat{y}\|\mathbf{x}) \log P^f_\phi(y = \hat{y}\|\mathbf{x})$
g	Label Prediction from the LF-based graphical model
LL_s	Supervised negative log likelihood
LL_u	Unsupervised negative log likelihood summed over labels
KL	KL Divergence between two probability models
R	Quality Guide based loss

Table 3: Summary of notation used.

We also use an LF-based graphical model $P_\theta(\mathbf{l}_i, y)$ which, as specified in equation (1) for an example $\mathbf{x}_i$, is a generative model on the LF outputs and class label y.

$$P_\theta(\mathbf{l}_i, y) = \frac{1}{Z_\theta} \prod_{j=1}^{m} \psi_\theta(l_{ij}, y) \tag{1}$$

$$\psi_\theta(l_{ij}, y) = \begin{cases} \exp(\theta_{jy}) & \text{if } l_{ij} \neq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

There are K parameters $\theta_{j1}, \theta_{j2} ... \theta_{jK}$ for each LF λ_j, where K is the number of classes. The model makes the simple assumption that each LF λ_j independently acts on an instance $\mathbf{x}_i$ to produce outputs $l_{i1}, l_{1i} ... l_{im}$. The potentials ψ_θ invoked in equation (1) are defined in equation (2). Z_θ is the normalization factor. We propose a joint learning algorithm with semi-supervision to employ both features and LF predictions in an end-to-end manner.

3.3 Joint Learning in SPEAR

We first specify the objective of SPEAR and thereafter explain each of its components in greater detail:

$$\begin{aligned}
\min_{\theta,\phi} & \sum_{i \in \mathcal{L}} L_{CE}\left(P^f_\phi(y|\mathbf{x}_i), y_i\right) + \sum_{i \in \mathcal{U}} H\left(P^f_\phi(y|\mathbf{x}_i)\right) \\
& + \sum_{i \in \mathcal{U}} L_{CE}\left(P^f_\phi(y|\mathbf{x}_i), g(\mathbf{l}_i)\right) + LL_s(\theta|\mathcal{L}) \\
& + LL_u(\theta|\mathcal{U}) + \sum_{i \in \mathcal{U} \cup \mathcal{L}} KL\left(P^f_\phi(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i)\right) \\
& + R(\theta|\{q_j\})
\end{aligned} \tag{3}$$

Before we proceed further, we refer the reader to Table 3 in which we summarise the notation built so far as well as the notation that we will soon be introducing.

First Component (L1): The first component (L1) of the loss $L_{CE}\left(P^f_\phi(y|\mathbf{x}_i), y_i\right) = -\log\left(P^f_\phi(y = y_i|\mathbf{x}_i)\right)$ is the standard cross-entropy loss on the labelled dataset $\mathcal{L}$ for the model P^f_ϕ.

Second Component (L2): The second component L2 is the semi-supervised loss on the unlabelled data $\mathcal{U}$. In our framework, we can use any

---

[2] We use the association of LF λ_j with some class k_j only in the quality guide component (QG) of the loss in eqn. 3

unsupervised loss function. However, for this paper, we use the Entropy minimisation (Grandvalet and Bengio, 2005) approach. Thus, our second component $H\left(P_\phi^f(y|\mathbf{x}_i)\right)$ is the entropy of the predictions on the unlabelled dataset. It acts as a form of semi-supervision by trying to increase the confidence of the predictions made by the model on the unlabelled dataset.

Third Component (L3): The third component $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), g(\mathbf{l}_i)\right)$ is the cross-entropy of the classification model using the hypothesised labels from CAGE (Chatterjee et al., 2020) on $\mathcal{U}$. Given that $\mathbf{l}_i$ is the output vector of all labelling functions for any $\mathbf{x}_i \in \mathcal{U}$, we specify the predicted label for $\mathbf{x}_i$ using the LF-based graphical model $P_\theta(\mathbf{l}_i, y)$ from eqn. (1) as: $g(\mathbf{l}_i) = \text{argmax}_y P_\theta(\mathbf{l}_i, y)$.

Fourth Component (L4): The fourth component $LL_s(\theta|\mathcal{L})$ is the (supervised) negative log likelihood loss on the labelled dataset $\mathcal{L}$ as per eqn. (3):

$$LL_s(\theta|\mathcal{L}) = -\sum_{i=1}^{N} \log P_\theta(\mathbf{l}_i, y_i)$$

Fifth Component (L5): The fifth component $LL_u(\theta|\mathcal{U})$ is the negative log likelihood loss for the unlabelled dataset $\mathcal{U}$ as per eqn. (3). Since the true label information is not available, the probabilities need to be summed over y: $LL_u(\theta|\mathcal{U}) = -\sum_{i=N+1}^{M} \log \sum_{y \in \mathcal{Y}} P_\theta(\mathbf{l}_i, y)$

Sixth Component (L6): The sixth component $KL(P_\phi^f(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i))$ is the Kullback-Leibler (KL) divergence between the predictions of both the models, *viz.*, feature-based model f_ϕ and the LF-based graphical model P_θ summed over every example $\mathbf{x}_i \in \mathcal{U} \cup \mathcal{L}$. Through this term, we try and make the models agree in their predictions over the union of the labelled and unlabelled datasets.

Quality Guides (QG): As the last component in our objective, we use quality guides $R(\theta|\{q_j\})$ on LFs, which have been shown in (Chatterjee et al., 2020) to stabilise the unsupervised likelihood training while using labelling functions. Let q_j be the fraction of cases where λ_j correctly triggered, and let q_j^t be the user's belief on the fraction of examples $\mathbf{x}_i$ where y_i and l_{ij} agree. If the user's beliefs were not available, we consider the precision of the LFs on the validation set as the user's beliefs. Except for the SMS dataset, we take the precision of the LFs on the validation set as the quality guides. If $P_\theta(y_i = k_j|l_{ij} = 1)$ is the model-based precision over the LFs, the quality guide based loss can be expressed as $R(\theta|\{q_j^t\}) = \sum_j q_j^t \log P_\theta(y_i = k_j|l_{ij} = 1) + (1-q_j^t)\log(1-P_\theta(y_i = k_j|l_{ij} = 1))$. Throughout the paper, we consider QG always in conjunction with Loss **L5**.

In summary, the first three components (L1, L2 and L3) invoke losses on the supervised model f_ϕ. While L1 compares the output f_ϕ against the ground truth in the labelled set $\mathcal{L}$, L2 and L3 operate on the unlabelled data $\mathcal{U}$ by minimizing the entropy of f_ϕ (L2) and by calibrating the f_ϕ output against the noisy predictions $g(\mathbf{l}_i)$ of the graphical model $P_\theta(\mathbf{l}_i, y)$ for each $\mathbf{x}_i \in \mathcal{U}$ (L3). The next two components L4 and L5 focus on maximizing the likelihood of the parameters θ of $P_\theta(\mathbf{l}_i, y)$ over labelled $\mathbf{x}_i \in \mathcal{L}$ and unlabelled $\mathbf{x}_i \in \mathcal{U}$ datasets respectively. Finally, in L6, we compare the probabilistic outputs from the supervised model f_ϕ against those from the graphical model $P_\theta(\mathbf{l}, y)$ through a KL divergence based loss. We use the ADAM (stochastic gradient descent) optimizer to train the non-convex loss objective.

Previous data programming approaches (Bach et al., 2019; Chatterjee et al., 2020) adopt a cascaded approach in which they first optimise a variant of L5 to learn the θ parameters associated with the LFs and thereafter use the noisily generated labels using $g(\mathbf{l})$ to learn the supervised model f_ϕ using a variant of L3. In contrast, our approach learns the LF's θ parameters and the model's ϕ parameters jointly in the context of the unlabelled data $\mathcal{U}$.

We present synthetic experiments to illustrate the effect of SPEAR for data programming and semi-supervision in a controlled setting in which (i) the overlap between classes in the data is controlled and (ii) the labelling functions are accurate. The details of the synthetic experiments are provided in the appendix.

3.4 SPEAR-SS: Subset Selection with SPEAR

Suppose we are given an unlabelled data set $\mathcal{U}$ and a limited budget for data labelling because of the costs involved in it. It is essential for us to choose the data points that need to be labelled properly. We explore two strategies for selecting a subset of data points from the unlabelled set. We then obtain the labels for this subset, and run SPEAR on the combination of this labelled and unlabelled set. The two approaches given are intended to maximise diversity of the selected subset in the feature space. We complement both the approaches with Entropy Filtering (also described below).

Unsupervised Facility Location: In this approach, given an unlabelled data-set $\mathcal{U}$, we want to select a subset $\mathcal{S}$ such that the selected subset has maximum diversity with respect to the features. Inherently, we are trying to maximise the information gained by a machine learning model when trained on the subset selected. The objective function for unsupervised facility location is $f_{\text{unsup}}(\mathcal{S}) = \sum_{i \in \mathcal{U}} \max_{j \in \mathcal{S}} \sigma_{ij}$ where σ_{ij} denotes the similarity score (in the feature space $\mathcal{X}$) between data instance $\mathbf{x}_i$ in unlabelled set $\mathcal{U}$ and data instance $\mathbf{x}_j$ in selected subset data $\mathcal{S}$. We employ a lazy greedy strategy to select the subset. In conjunction with Entropy Filtering described below,we call this technique *Unsupervised Subset Selection*.

Supervised Facility Location: The objective function for Supervised Facility Location (Wei et al., 2015) is $f_{\text{sup}}(\mathcal{S}) = \sum_{y \in \mathcal{Y}} \sum_{i \in \mathcal{U}_y} \max_{j \in \mathcal{S} \cap \mathcal{U}_y} \sigma_{ij}$. Here we assume that $\mathcal{U}_y \subseteq \mathcal{U}$ is the subset of data points with hypothesised label y. Simply put, $\mathcal{U}_y$ forms a partition of $\mathcal{U}$ based on the hypothesized labels obtained by performing unsupervised learning with labelling functions. In conjunction with Entropy Filtering, we call this technique *Supervised Subset Selection*.

Entropy Filtering: We also do a filtering based on entropy. In particular, we sort the examples based on maximum entropy and select fB number of data points[3], where B is the data selection budget (which was set to the size of the labelled set $|\mathcal{L}|$ in all our experiments). On the filtered dataset, we perform the subset selection, using either the supervised or unsupervised facility location as described above. Below, we describe the optimisation algorithm for subset selection.

Optimisation Algorithms and Submodularity: Both $f_{\text{unsup}}(\mathcal{S})$ and $f_{\text{sup}}(\mathcal{S})$ are submodular functions. We select a subset $\mathcal{S}$ of the filtered unlabelled data, by maximising these functions under a cardinality budget k (*i.e.*, a labelling budget). For cardinality constrained maximisation, a simple greedy algorithm provides a near-optimal solution (Nemhauser et al., 1978). Starting with $\mathcal{S}^0 = \emptyset$, we sequentially update

$$\mathcal{S}^{t+1} = \mathcal{S}^t \cup \underset{j \in \mathcal{U} \setminus \mathcal{S}^t}{\operatorname{argmax}} f(j|\mathcal{S}^t) \tag{4}$$

where $f(j|\mathcal{S}) = f(\mathcal{S} \cup j) - f(\mathcal{S})$ is the gain of adding element j to set $\mathcal{S}$. We iteratively execute the greedy step (4) until $t = k$ and $|\mathcal{S}^t| = k$. It is easy to see that the complexity of the greedy algorithm is $O(nkT_f)$, where T_f is the complexity of evaluating the gain $f(j|\mathcal{S})$ for the supervised and unsupervised facility location functions. We then significantly optimize this simple greedy algorithm via a lazy greedy algorithm (Minoux, 1978) via memoization and precompute statistics (Iyer and Bilmes, 2019).

Dataset	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{U}\rvert$	#Rules/LFs	Precision	%Cover	|Test|
MIT-R	1842	64888	15	80.7	14	14256
YouTube	100	1586	10	78.6	87	250
SMS	69	4502	73	97.3	40	500
Census	83	10000	83	84.1	100	16281
Ionosphere	73	98	64	65	100	106
Audit	162	218	52	87.5	100	233
IMDB	284	852	25	80	58.1	500

Table 4: Statistics of datasets and their rules/LFs. Precision refers to micro precision of rules. %Cover is the fraction of instances in $\mathcal{U}$ covered by at least one LF. Size of Validation set is equal to $|\mathcal{L}|$.

4 Experiments

In this section, we (1) evaluate our joint learning against state-of-the-art approaches and (2) demonstrate the importance of subset selection over random subset selection. We present evaluations on seven datasets on tasks such as text classification, record classification and sequence labelling.

4.1 Datasets

We adopt the same experimental setting as in Awasthi et al. (2020) for the dataset split and the labelling functions. However (for the sake of fairness), we set the validation data size to be equal to the size of the labelled data-set unlike Awasthi et al. (2020) in which the size of the validation set was assumed to be much larger.

We use the following datasets: **(1) YouTube**: A spam classification on YouTube comments; **(2) SMS Spam Classification (Almeida et al., 2011)**, which is a binary spam classification dataset containing 5574 documents; **(3) MIT-R (Liu et al., 2013)**, is a sequence labelling task on each token with following labels: *Amenity, Prices, Cuisine, Dish, Location, Hours, Others*; **(4) IMDB**, which is a plot summary based movie genre binary classification dataset, and the LFs (and the labelled set) are obtained from the approach followed by Varma and Ré (2018); **(5) Census** (Dua and Graff, 2017), **(6) Ionosphere**, and **(7) Audit**, which are all UCI datasets. The task in the Census is to predict whether a person earns more than $50K or not.

---

[3] In our experiments, we set $f = 5$

Ionosphere is radar binary classification task given a list of 32 features. The task in the Audit is to classify suspicious firms based on the present and historical risk factors.

Statistics pertaining to these datasets are presented in Table 4. Since we compare performances against models that adopt different terminology, we refer to rules and labelling functions interchangeably. For fairness, we restrict the size of the validation set and keep it equal to the size $|\mathcal{L}|$ of the labelled set. For all experiments involving comparison with previous approaches, we used code and hyperparameters from (Awasthi et al., 2020) but with our smaller-sized validation set. Note that we mostly outperform them even with their larger-sized validation set as can be seen in Table 5. More details on training and validation set size are given in the appendix.

4.2 Baselines

In Table 5, we compare SPEAR and SPEAR-SS against other following standard methods on seven datasets.

Only-$\mathcal{L}$: We train the classifier $P_\theta(y|\mathbf{x})$ only on the labelled data $\mathcal{L}$ using loss component L1. As explained earlier, following (Awasthi et al., 2020), we observe that a 2-layered neural network trained with the small amount of labelled data is capable of achieving competitive accuracy. We choose this method as a baseline and report gains over it.

$\mathcal{L} + \mathcal{U}_{maj}$: We train the baseline classifier $P_\theta(y|x)$ on the labelled data $\mathcal{L}$ along with $\mathcal{U}_{maj}$ where labels on the $\mathcal{U}$ instances are obtained by majority voting on the rules/LFs. The training loss is obtained by weighing instances labelled by rules as $\min_\theta \sum_{(\mathbf{x}_i,\mathbf{l}_i)\in\mathcal{L}} - \log P_\theta(\mathbf{l}_i|\mathbf{x}_i) + \gamma \sum_{(\mathbf{x}_i,y_i)\in\mathcal{L}} - \log P_\theta(y_i|\mathbf{x}_i)$.

Learning to Reweight (L2R) (Ren et al., 2018): This method trains the classifier by an online training algorithm that assigns importance to examples based on the gradient direction.

$\mathcal{L}$+$\mathcal{U}_{Snorkel}$ (Ratner et al., 2016): Snorkel's generative model that models class probabilities based on discrete LFs for consensus on the noisy and conflicting labels.

Posterior Regularization (PR) (Hu et al., 2016): This is a method for joint learning of a rule and feature network in a teacher-student setup.

Imply Loss (Awasthi et al., 2020): This approach uses additional information in the form of labelled rule exemplars and trains with denoised rule-label loss. Since it uses information in addition to what we assume, Imply Loss can be considered as a skyline for our proposed approaches.

4.3 Results with SPEAR

SPEAR uses the 'best' combination of the loss components L1, L2, L3, L4, L5, L6. To determine the 'best' combination, we perform a grid search over various combinations of losses using validation accuracy/f1-score as the criteria for selecting the most appropriate loss combination. Imply Loss uses a larger-sized validation set to tune their models. In our experiments, we maintained a validation set size equal to the size of the labelled data. In Table 5, we observe that SPEAR performs significantly better than all other approaches on all but the MIT-R data-set. Please note that all results are based on the same hand-picked labelled data subset as was chosen in prior work (Awasthi et al., 2020; Varma and Ré, 2018), except for Audit and Ionosphere. Even though we do not have rule-exemplar information in our model, SPEAR achieves better gains than even ImplyLoss. Recall that the use of ImplyLoss can be viewed as a skyline approach owing to the additional exemplar information that associates labelling functions with specific labelled examples. The slightly lower performance of the 'best' SPEAR on the MIT-R data-set can be partially explained by the fact that there are no LFs corresponding to the '0' class label, owing to which our graphical model is not trained for all classes. However, as we will show in the next section, by suitably determining a subset of the data-set that can be labelled (using the facility location representation function), we achieve improved performance even on the MIT-R data-set (see Table 5). Also, note that in Table 5, we present results on two versions of Audit, one in which both the train and test set are balanced, and the other where the labelled training set is imbalanced. In the imbalanced case (where the number of positives are only 10%), we were unable to successfully run the ImplyLoss and Posterior-Reg models (and hence the '-'), despite communication with the authors. We see that SPEAR and similarly, SPEAR-SS (discussed below) significantly outperform the baselines by almost 40% in the imbalanced case. In the balanced case, the gains are similar to what we observe on the other datasets.

Methods	Datasets							
	YouTube [Accuracy]	SMS [F1]	MIT-R [F1]	IMDB [F1]	Census [Accuracy]	Ionosphere [F1]	Audit (Imb) [F1]	Audit (Bal) [F1]
Only-$\mathcal{L}$ (Handpicked)	90.7 (1.2)	90.0 (3.7)	74.1 (0.4)	72.2 (3.1)	78.3 (0.3)	92.7 (0.5)	24.7(2.6)	87.3 (0.9)
$\mathcal{L}$+$\mathcal{U}_{maj}$ (Handpicked)	+1.9 (1.1)	-0.3 (1.4)	+0.1(0.2)	+1.2 (0.3)	-0.9 (0.4)	+0.4 (0.7)	-4.8 (6)	-1.4(4.2)
L2R (Ren et al., 2018) (Handpicked)	-3.7 (5.1)	+0.7 (2.9)	-20.2(0.9)	+4.5(0.2)	+3.6(0.3)	-18.8 (0.3)	-1.2(3.2)	-3.0(4.9)
$\mathcal{L}$+$\mathcal{U}_{Snorkel}$ (Ratner et al., 2016) (Handpicked)	+0.9 (2.6)	+0.3 (4.5)	-0.3(0.2)	+0.6(1.8)	+1.7 (0.2)	-0.6 (0.5)	-7.4(3.4)	-0.6(4.2)
Posterior Reg (Hu et al., 2016) (Handpicked)	-1.9(1.6)	-3.3 (1.9)	-0.2 (0.2)	+1.1 (0.7)	-1.9 (0)	-0.1(0.7)	-	+0.1 (1.4)
ImplyLoss (Awasthi et al., 2020) (Handpicked)	+0.4 (0.5)	+0.9(0.9)	0.9(0.4)	+4.3 (1.5)	+3.4 (0.1)	-3.9(2.4)	-	+0.5(1)
SPEAR (Handpicked)	+3.7(0.5)	**+3.4**(0.9)	-0.8 (0.5)	+4.9(0.3)	**+3.7** (0.3)	**+5.4** (0.3)	+44 (0.9)	**+4.3** (0.9)
SPEAR-SS (Random Subset Selection)	+3.5	+1.8	-2.9	+4.0	-5.2	+4.7	+41.7	+2.0
SPEAR-SS (Unsupervised Subset Selection)	+3.9	+1.9	+2.6	-0.6	+2.5	+4.8	+43.5	+3.3
SPEAR-SS (Supervised Subset Selection)	**+4.2**	+3.2	+**2.9**	**+6.3**	+2.5	+5.1	**+44.5**	+3.5

Table 5: Performance of SPEAR and SPEAR-SS for three subset selection schemes on seven data-sets. All numbers reported are gains over the baseline method (Only-$\mathcal{L}$). All results are averaged over 5 runs. Numbers in brackets '()' represent standard deviation of the original score. Handpicked instances refers to instances selected from the dataset for designing LFs. These instances are taken directly from (Awasthi et al., 2020) to ensure fair comparison.

4.4 Results with SPEAR-SS

Recall that all results discussed so far (including those for SPEAR) on the Youtube, SMS, MIT-R, IMDB and Census datasets were based on the same 'hand-picked' labelled data subset as in prior work (Awasthi et al., 2020; Varma and Ré, 2018). In the case of Audit and Ionosphere, the labelled subset was randomly picked. In Table 5, we summarise the results obtained by employing supervised and unsupervised subset selection schemes for picking the labelled data-set and present comparisons against results obtained using (i) 'hand-picked' labelled data-sets, and (ii) random selection of the labelled set. In each case, the size of the subset is the same, which we set to be the size of the hand-picked labelled set. Our data selection schemes are applied to the 'best' SPEAR model obtained across various loss components. We observe that the best-performing model for the supervised and unsupervised data selection tends to outperform the best model based on random selection. Secondly, we observe that between the supervised and unsupervised data selection approaches, the supervised one tends to perform the best, which means that using the hypothesised labels does help. Thirdly, we observe that YouTube, MIT-R, IMDB and Audit using the selected subset outperform prior work that employ hand picked data-set, whereas, in the case of SMS, Census and Ionosphere, we come close. Finally, our approach is more stable than other approaches as the standard deviation of SPEAR is low for 5 different runs across all the datasets.

As an illustration, the examples such as S_3 and S_4 referred to in Section 1.1 were precisely obtained through supervised subset selection in SPEAR-SS, to form part of the labelled dataset. As previously observed in Table 2, S_3 and S_4 *complement* (via n-grams features such as F1 and F2) the effect of the labelling functions LF1 and LF2 on the unlabelled examples such as S_1 and S_2, when included in the labelled set. Further detailed results with subset selection, *etc.* can be found in the appendix. In general, we observe that when the subset of instances selected for labelling is *complementary* to the labelling functions (as in our case), the performance is higher than when the labelled examples (exemplars) are inspired by labelling functions themselves as done in the work by Awasthi et al. (2020).

4.5 Significance Test

We employ the Wilcoxon signed-rank test (Wilcoxon, 1992) to determine whether there is a significant difference between SPEAR and Imply Loss (current state-of-the-art). Our null hypothesis is that there is not significant difference between SPEAR and Imply loss. For $n = 7$ instances, we observe that the one-tailed hypothesis is significant at $p < .05$, so we reject the null hypothesis. Clearly, SPEAR significantly outperforms Imply loss and, therefore, all previous baselines.

Similarly, we perform the significance test to assess the difference between SPEAR-SS and Imply Loss. As expected, the one-tailed hypothesis is significant at $p < 0.05$, which implies that our SPEAR-SS approach significantly outperforms Imply Loss, and thus all other approaches.

5 Conclusion

We study how data programming can benefit from labelled data by learning a model (SPEAR) that

Methods	Datasets			
	YouTube [Accuracy]	SMS [F1]	MIT-R [F1]	Census [Accuracy]
ImplyLoss (Awasthi et al., 2020)	94.1	93.2	74.3	81.1
SPEAR (Handpicked)	+0.3	**+0.2**	-0.9	**+0.9**
SPEAR-SS (Supervised Subset Selection)	**+0.8**	0.0	**+1.7**	-0.3

Table 6: Comparison of SPEAR and SPEAR-SS against ImplyLoss on subset of datasets from Table 5 for which ImplyLoss used a much larger validation set than $|\mathcal{L}|$. JL uses a validation set sizes equal to $|\mathcal{L}|$.

jointly optimises the consensus obtained from labelling functions in an unsupervised manner, along with semi-supervised loss functions designed in the feature space. We empirically assess the performance of the different components of our joint loss function. As another contribution, we also study some subset selection approaches to guide the selection of the labelled subset of examples. We present the performance of our models and present insights on both synthetic and real datasets. While outperforming previous approaches, our approach is often better than an exemplar-based (skyline) approach that uses the additional information of the association of rules with specific labelled examples.

Acknowledgements

We thank anonymous reviewers for providing constructive feedback. Ayush Maheshwari is supported by a Fellowship from Ekal Foundation (www.ekal.org). We are also grateful to IBM Research, India (specifically the IBM AI Horizon Networks - IIT Bombay initiative) for their support and sponsorship.

References

Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. 2019. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, pages 362–375.

Ramakrishna Bairi, Rishabh Iyer, Ganesh Ramakrishnan, and Jeff Bilmes. 2015. Summarization of multi-document topic hierarchies using submodular mixtures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 553–563.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. In *AAAI*.

Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. 2008. Maximum likelihood rule ensembles. In *Proceedings of the 25th international conference on Machine learning*, pages 224–231.

Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.

Garrett B Goh, Charles Siegel, Abhinav Vishnu, and Nathan Hodas. 2018. Using rule-based labels for weak supervised learning: a chemnet for transferable chemical property prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 302–310.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. 2016. Harnessing deep neural networks with logic rules. *CoRR*, abs/1603.06318.

Rishabh Iyer and Jeff Bilmes. 2019. A memoization framework for scaling submodular optimization to large scale problems. *arXiv preprint arXiv:1902.10176*.

Pratik Jawanpuria, Saketha N Jagarlapudi, and Ganesh Ramakrishnan. 2011. Efficient rule ensemble learning using hierarchical kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 161–168.

Vishal Kaushal, Rishabh Iyer, Suraj Kothawade, Rohan Mahadev, Khoshrav Doctor, and Ganesh Ramakrishnan. 2019. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1289–1299. IEEE.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. Glister: Generalization based data subset selection for efficient and robust learning. *In AAAI*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.

Ashish Kulkarni, Narasimha Raju Uppalapati, Pankaj Singh, and Ganesh Ramakrishnan. 2018. An interactive multi-label consensus labeling model for multiple labeler judgments. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI), 2018*, pages 1479–1486. AAAI Press.

Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. 2013. Query understanding enhanced by hierarchical parsing structures. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77. IEEE.

Michel Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2019. Data sketching for faster training of machine learning models. *arXiv preprint arXiv:1906.01827*.

Ajay Nagesh, Ganesh Ramakrishnan, Laura Chiticariu, Rajasekar Krishnamurthy, Ankush Dharkar, and Pushpak Bhattacharyya. 2012. Towards efficient named-entity rule induction for customizability. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 128–138, Jeju Island, Korea. Association for Computational Linguistics.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, pages 1–10. Vancouver, CA:.

Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endow.*, 12(3):223–236.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

A Illustration of SPEAR on a synthetic setting

Through a synthetic example, we illustrate the effectiveness of our formulation of combining semi-supervised learning with labelling functions (*i.e.*, combined Losses 1-6) to achieve superior performance. Consider a 3-class classification problem with overlap in the feature space as depicted in Figure 1. The classes are A, B and C. Though we illustrate the synthetic setting in 2 dimensions, in reality, we performed similar experiments in three dimensions (and results were similar). We randomly pick 5 points from each class $i \in \{a, b, c\}$, and corresponding to each such point (x_i, y_i) we create a labelling function based on its coordinates:

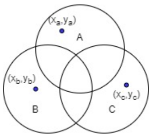

Figure 1: Synthetic data

- LF_a: Consider the point (x_a, y_a). The corresponding LF will be: if $y \geq y_a$ return 1 (i.e. classify as class A) else return 0 (abstain).

- LF_b: Similarly for (x_b, y_b) the LF will return 1 if $x \leq x_b$ and else will return 0.

- LF_c: The LF corresponding to (x_c, y_c) will return 1 if $x \geq x_c$ and else will return 0.

These seemingly 15 weak labelling functions (5 for each class) actually aid in classification when the labelled example set is extremely small and the classifier is unable to get a good estimate of the class boundaries. This can be observed in Table 7 wherein we report the F1 score on a held out test dataset for models obtained by training on the different loss components. The results are reported in the case of three dimensions, wherein each circle was obtained as a 3-dimensional gaussian. The means for the three classes A, B and C were respectively, $(0, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$

and the variance for each class was set to $(1, 1, 1)$. The training and test sets had 1000 examples each, with roughly equal number of samples randomly generated from each class (gaussian). In the first experiment (result in the first row of Table 7), the training was performed on the L1 loss by treating the entire data-set of 1000 examples as labelled. In all the other experiments only 1% (10 examples with almost uniform distribution across the 3 classes) of the training set was considered to be labelled and the remaining (990 examples) were treated as unlabelled.

Loss component used for training	F1 Score
L1 (on entire dataset as labelled)	0.584
L1 (1% labelled)	0.349
L1 (1% labelled) +L2	0.352
L1 (1% labelled)+L2+L3+L4 (1% labelled)+L5+L6	**0.440**
L4 (1% labelled)+L5	0.28

Table 7: F1 scores on test data in the synthetic setting

We make the following important observations with respect to Table 7: (1) **Skyline:** When the entire training data is treated as labelled and loss function L1 is minimized, we obtain a skyline model with F1 score of 0.584. (2) With just 1% labelled data on L1, we achieve 0.349 F1 score (using only the labelled data). (3) We obtain an F1 score of 0.28 using the labelling functions on the unlabelled data (for L5) in conjunction with the 1% labelled data (for L4). (4) When the 1% labelled data (for L1) and the remaining observed unlabelled data (for L2) are used to train the semi-supervised model using L1+L2, an F1 score of 0.352 is obtained. (5) However, by jointly learning on all the loss components, we observe an F1 score of 0.44. This is far better than the numbers obtained using only (semi)-supervised learning and those obtained using only the labelling functions. Understandably, this number is lower than the skyline of 0.584 mentioned on the first row of Table 7.

B Network Architecture

To train our model on the supervised data L, we use a neural network architecture having two hidden layers with ReLU activation. We chose our classification network to be the same as (Awasthi et al., 2020). In the case of MIT-R and SMS, the classification network contain 512 units in each hidden layer whereas the classification network for Census has 256 units in its hidden layers. For the YouTube dataset, we used a simple logistic regression as a classifier network, again as followed in (Awasthi et al., 2020). The features as well as the labelling functions for each dataset are also directly obtained from Snorkel (Ratner et al., 2016) and (Awasthi et al., 2020). Please note that all experiments (barring those on subset selection) are based on the same hand-picked labelled data subset as was chosen in (Awasthi et al., 2020).

Datasets	lr (f network)	lr (g network)	Batch Size
SMS	0.0001	0.01	256
MIT-R	0.0003	0.001	512
Census	0.0003	0.001	256
Youtube	0.0003	0.001	32
Ionosphere	0.003	0.01	32
Audit	0.0003	0.01	32
IMDB	0.0003	0.01	32

Table 8: Hyper parameter details for the different datasets

In each experiment, we train our model for 100 epochs and early stopping was performed based on the validation set. We use Adam optimizer with the dropout probability set to 0.8. The learning rate for f and g network are set to 0.0003 and 0.001 respectively for YouTube, Census and MIT-R datasets. For SMS dataset, learning rate is set to 0.0001 and 0.01 for f and g network. For Ionosphere dataset, learning rate for f is set to 0.003. For each experiment, the numbers are obtained by averaging over five runs, each with a different random initialisation. The model with the best performance on the validation set was chosen for evaluation on the test set. As mentioned previously, the experimental setup in (Awasthi et al., 2020) surprisingly employed a large validation set. For fairness, we restrict the size of the validation set and keep it equal to the size of the labelled set. For all experiments involving comparison with previous approaches, we used code and hyperparameters from (Awasthi et al., 2020) but with our smaller sized validation set.

Following (Awasthi et al., 2020), we used binary-F1 as an evaluation measure for the SMS, macro-F1 for MIT-R datasets, and accuracy for the YouTube and Census datasets.

C Optimisation Algorithms and Submodularity: Lazy Greedy and Memoization

Both $f_{\text{unsup}}(X)$ and $f_{\text{sup}}(X)$ are submodular functions, and for data selection, we select a subset

Loss Combination	Datasets				
	YouTube (Accuracy)	SMS (F1)	MIT-R (F1)	IMDB (F1)	Census (Accuracy)
L1+L2+L3+L4	94.6	93.1	72.5	73.6	82.0
L1+L2+L4+L6	92.0	91.9	69.7	73.3	81.3
L1+L3+L4+L6	94.4	93.2	29.8	74.4	81.0
L1+L2+L3+L4+L6	94.4	92.3	29.5	64.4	80.9
L1+L3+L4+L5+L6	**94.6**	**93.4**	**73.2**	**77.1**	**82.0**
L1+L2+L3+L4+L5+L6	94.5	93.0	72.8	76.9	81.9

Table 9: Performance on the test data, of various loss combinations from our objective function in equation (3). For each dataset, the numbers in **bold** refer to the 'best' performing combination, determined based on performance on the validation data-set. In general, we observe that all the loss components (barring L2) contribute to the best model. Note that all combinations includes QG (Component 7).

X of the unlabelled data, which maximises these functions under a cardinality budget (i.e. a labelling budget). For cardinality constrained maximisation, a simple greedy algorithm provides a near optimal solution (Nemhauser et al., 1978). Starting with $X^0 = \emptyset$, we sequentially update $X^{t+1} = X^t \cup \text{argmax}_{j \in V \setminus X^t} f(j|X^t)$, where $f(j|X) = f(X \cup j) - f(X)$ is the gain of adding element j to set X. We run this till $t = k$ and $|X^t| = k$, where k is the budget constraint. It is easy to see that the complexity of the greedy algorithm is $O(nkT_f)$ where T_f is the complexity of evaluating the gain $f(j|X)$ for the supervised and unsupervised facility location functions. This simple greedy algorithm can be significantly optimized via a lazy greedy algorithm (Minoux, 1978). The idea is that instead of recomputing $f(j|X^t), \forall j \notin^t$, we maintain a priority queue of sorted gains $\rho(j), \forall j \in V$. Initially $\rho(j)$ is set to $f(j), \forall j \in V$. The algorithm selects an element $j \notin X^t$, if $\rho(j) \geq f(j|X^t)$, we add j to X^t (thanks to submodularity). If $\rho(j) \leq f(j|X^t)$, we update $\rho(j)$ to $f(j|X^t)$ and re-sort the priority queue. The complexity of this algorithm is roughly $O(kn_R T_f)$, where n_R is the average number of re-sorts in each iteration. Note that $n_R \leq n$, while in practice, it is a constant thus offering almost a factor n speedup compared to the simple greedy algorithm. One of the parameters in the lazy greedy algorithms is T_f, which involves evaluating $f(X \cup j) - f(X)$. One option is to do a naïve implementation of computing $f(X \cup j)$ and then $f(X)$ and take the difference. However, due to the greedy nature of algorithms, we can use memoization and maintain a precompute statistics $p_f(X)$ at a set X, using which the gain can be evaluated much more efficiently (Iyer and Bilmes, 2019). At every iteration, we evaluate $f(j|X)$ using $p_f(X)$, which we call $f(j|X, p_f)$. We then update $p_f(X \cup j)$ after adding element j to X. Both the supervised and unsupervised facility location functions admit precompute statistics thereby enabling further speedups.

D Role of different components in the loss function

Given that our loss function has seven components (including the quality guides), a natural question is 'how do we choose among the different components for joint learning (JL)?' Another question we attempt to answer is 'whether all the components are necessary for JL?' For our final model (*i.e.*, the results presented in Tables 6 and 7 of the main paper), we attempt to choose the best performing JL combination of the 7 loss components, *viz.* L1, L2, L3, L4, L5, L6. To choose the 'best' JL combination, we evaluate the performance on the validation set of the different JL combinations. Since we generally observe considerably weaker performance by selecting lesser than 3 loss terms, we restrict ourselves to 3 or more loss terms in our search. We report performance on the test data, of various JL combinations from our objective function for each of the four data-sets. For each data-set, the numbers in **bold** refer to the 'best' performing JL combination, determined based on performance on the validation data-set.

The observations on the results are as follows. Firstly, we observe that all the loss components (barring L2 for three datasets) contribute to the best model. Furthermore, we observe that the best JL combination (picked on the basis of the validation set) either achieves the best performance or close to best among the different JL combinations as measured on the test dataset. Secondly, we observe that QGs do not cause significant improvement in the performance during training.

Fingerprinting Fine-tuned Language Models in the Wild*

Nirav Diwan[1], Tanmoy Chakravorty[1], and Zubair Shafiq[2]

[1]IIIT-Delhi, India
[2]University of California, Davis
[1]{nirav17072,tanmoy}@iiitd.ac.in
[2]zubair@ucdavis.edu

Abstract

There are concerns that the ability of language models (LMs) to generate high quality synthetic text can be misused to launch spam, disinformation, or propaganda. Therefore, the research community is actively working on developing approaches to detect whether a given text is organic or synthetic. While this is a useful first step, it is important to be able to further fingerprint the author LM to attribute its origin. Prior work on fingerprinting LMs is limited to attributing synthetic text generated by a handful (usually < 10) of pre-trained LMs. However, LMs such as GPT2 are commonly fine-tuned in a myriad of ways (e.g., on a domain-specific text corpus) before being used to generate synthetic text. It is challenging to fingerprinting fine-tuned LMs because the universe of fine-tuned LMs is much larger in realistic scenarios. To address this challenge, we study the problem of large-scale fingerprinting of fine-tuned LMs in the wild. Using a real-world dataset of synthetic text generated by 108 different fine-tuned LMs, we conduct comprehensive experiments to demonstrate the limitations of existing fingerprinting approaches. Our results show that fine-tuning itself is the most effective in attributing the synthetic text generated by fine-tuned LMs.

1 Introduction

Background & motivation. State-of-the-art language models (LMs) can now generate long, coherent, and grammatically valid synthetic text (Devlin et al., 2019; Radford et al., 2018, 2019; Brown et al., 2020). On one hand, the ability to generate high quality synthetic text offers a fast and inexpensive alternative to otherwise labor-intensive useful applications such as summarization and chat bots (Yoo and Jeong, 2020; Yu et al., 2020; Wang et al., 2019; Liu and Lapata, 2019). On the other hand, such high quality synthetic text can also be misused by bad actors to launch spam, disinformation, or propaganda. For example, LMs such as Grover (Zellers et al., 2019) are shown to be capable of generating full-blown news articles, from just brief headlines, which are more believable than equivalent human written news articles. In fact, prior work has shown that humans cannot distinguish between organic (i.e., human written) and synthetic (i.e., generated by LM) text (Ippolito et al., 2020; Jawahar et al., 2020; Munir et al., 2021). Thus, this ability to generate high quality synthetic text can further be misused for social impersonation and phishing attacks because users can be easily misled about the authorship of the text.

Problem statement. To mitigate the potential misuse of LMs, the research community has started developing new text attribution techniques. However, as shown in Figure 1, the attribution of synthetic text is a multistage problem. The first step is to distinguish between organic and synthetic text (P1). Prior work has used the LM's output word probability distribution to detect synthetic text (Ippolito et al., 2020; Gehrmann et al., 2019; Zellers et al., 2019). However, there are several publicly available pre-trained LMs that might be used to generate synthetic text. Thus, the second step is to detect the pre-trained LM used to generate synthetic text (P2). Prior approaches showed promising results by attempting to fingerprint the LM based on its distinct semantic embeddings (Pan et al., 2020; Uchendu et al., 2020). However, pre-trained LMs such as GPT2 (Radford et al., 2019) are commonly fine-tuned before being used to generate synthetic text. Thus, the third step is to detect the fine-tuned LM used to generate synthetic text (P3). To the best of our knowledge,

---

*Supplementary contains dataset, source code, and an appendix (including hyper-parameter setting and additional results). The code and dataset are available at https://github.com/LCS2-IIITD/ACL-FFLM.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4652–4664
August 1–6, 2021.

prior work lacks approaches to effectively fingerprint fine-tuned LMs.

Technical challenges. It is particularly challenging to fingerprint fine-tuned LMs simply because of the sheer number of possible fine-tuned variants. More specifically, a pre-trained LM can be fine-tuned in a myriad of ways (e.g., separately on different domain-specific text corpora), resulting in a large number of classes. Another challenge to fingerprint fine-tuned LMs is that we cannot make assumptions about the nature of fine-tuning (e.g., parameters, training data) or generation (e.g., prompts). Prior work on fingerprinting pre-trained LMs is limited to evaluation on a small number of classes ($<$ 10 classes) and on synthetic text that is artificially generated using set prompts.

Proposed approach. To fingerprint synthetic text generated by fine-tuned LMs, we utilize the RoBERTa model (Liu et al., 2019) and attach a CNN-based classifier on top. We fine-tune the RoBERTa model for the downstream task of sentence classification using a synthetic text corpus. The fine-tuned model is used to extract embeddings as features that are then fed to the CNN classifier. We show that the fine-tuned RoBERTa model is able to capture the topic-specific distinguishing patterns of the synthetic text. Upon visualizing the generated features, the samples form closely-condensed distinguishable clusters based on the topic of the organic corpus the LMs have been fine-tuned upon. Therefore, we conclude that fine-tuning itself significantly helps fingerprinting a fine-tuned LM. Note that our fingerprinting approach does not assume access to the text corpus used for LM fine-tuning. We only assume access to arbitrary synthetic text generated by fine-tuned LMs.

Dataset. We gather a real-world dataset of synthetic text generated by fine-tuned LMs in the wild. More specifically, we extract synthetic text from the subreddit r/SubSimulatorGPT2. Each of the 108 users on r/SubSimulatorGPT2 is a GPT2 LM that is fine-tuned on 500k posts and comments from a particular subreddit (e.g., r/askmen, r/askreddit,r/askwomen). It is noteworthy that users on r/SubSimulatorGPT2 organically interact with each other using the synthetic text in the preceding comment/reply as their prompt.

Evaluation. We evaluate our models using a suite of evaluation metrics. We also adapt confidence-based heuristics, such as the gap statistic. Our best model is accurate for a large number of classes, across a variety of evaluation metrics, showing impressive results for the largest setting of 108 classes. While it obtains around 46% precision and 43% recall, its top-10 accuracy is about 70%. In other words, the correct class is one of the top-10 predictions in about 70% of the cases. If we give the model the option to not make a classification decision, via confidence estimation, it doubles the precision of the top classification from 46% to around 87% with a decrease of recall from 43% to 27%.

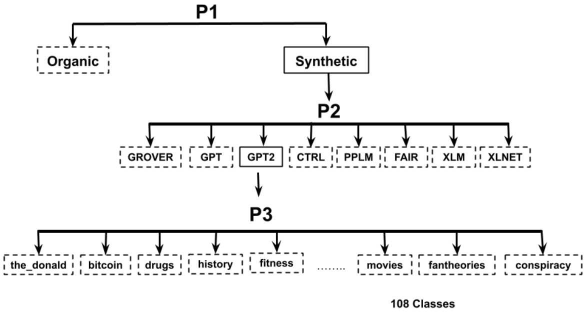

Figure 1: Attribution of text formulated as a series of three problems: P1, P2, and P3.

We summarize our key contributions as follows:

1. **Problem formulation.** To the best of our knowledge, we are the first to explore the problem of attribution of synthetic text generated by fine-tuned LMs (P3). We are also the first to investigate synthetic text attribution for a large number of classes on a real-world dataset. We also show that P3 is a much more challenging problem as compared to P1 and P2.
2. **Comprehensive model.** We design and implement a comprehensive set of different feature extraction techniques. We use them on a variety of machine learning and deep learning pipelines to build detection models.
3. **Rigorous evaluation.** We conduct rigorous evaluation on a real-world dataset of synthetic text generated by fine-tuned LMs. We use several evaluation metrics such as top-k accuracy and precision-recall tradeoff to compare different detection models. We also provide insights into the performance of different feature sets and classification algorithms.

Paper Organization: The rest of the paper is organized as follows. Section 2 contextuailzes our work with respect to prior literature. We analyze the real-world dataset of synthetic text generated by fine-tuned LMs in Section 3. Section 4 describes different feature sets and classification algorithms for fingerprinting fine-tuned LMs. Section 5 presents the results of our experimental evaluation before concluding in Section 6.

2 Related Work

Figure 1 illustrates three different problem formulations for attribution of synthetic text generated by LMs. The first line of research (**P1**) aims to distinguish between organic (by humans) and synthetic (by LMs) text. Given synthetic text, the second line of research (**P2**) further aims to attribute the synthetic text generated by pre-trained LMs such as BERT and GPT. Finally, in this paper, we further aim to (**P3**) attribute the synthetic text generated by fine-tuned LMs. Here, we first discuss prior work on **P1** and **P2** and then highlight the importance and unique challenges of **P3**.

P1: As the quality of synthetic text generated by LMs has improved, the problem of distinguishing between organic and synthetic text has garnered a lot of attention. Gehrmann et al. (2019) aimed to distinguish between synthetic text generated by GPT2 and Heliograf versus organic text by books, magazines, and newspapers. They showed that humans had a hard time classifying between organic and synthetic. Their proposed GLTR model, which uses probability and ranks of words as predicted by pre-trained LMs as features, was able to achieve 87% AUC. Zellers et al. (2019) developed Grover, a LM to generate fake news. They also showed that humans had a hard time distinguishing between organic and synthetic text generated by Grover. A classifier based on Grover achieved near perfect accuracy and significantly outperformed other classifiers based on pre-trained LMs. Ippolito et al. (2020) presented an interesting trade-off between distinguishability of organic and synthetic text. They showed that synthetic text optimized to fool humans is actually much easily detected by automated classification approaches. They generated synthetic text from pre-trained GPT2 using different sampling strategies and parameters, and used different classifiers such as GLTR that use pre-trained LMs and a purpose-built fine-tuned BERT based classifier. They showed that their fine-tuned BERT based classifier was able to significantly outperform other approaches as well as humans.

P2: Given synthetic text, recent work has further attempted to attribute authorship of synthetic text generated by pre-trained LMs. Uchendu et al. (2020) fingerprinted pre-trained LMs by attributing synthetic text to its author LM. They conducted exhaustive experiments using conventional authorship attribution models (Kim, 2014; Zhang et al., 2015; Cho et al., 2014) for eight pre-trained LMs – GPT (Radford et al., 2018), GPT2 (Radford et al., 2019) , GROVER (Zellers et al., 2019), FAIR (Ng et al., 2019), CTRL (Keskar et al., 2019), XLM (Conneau and Lample, 2019), XLNET (Yang et al., 2019), and PPLM (Dathathri et al., 2020). They showed that derived linguistic features when used with simple classifiers (Random Forest, SVM) perform the best. Pan et al. (2020) prepared a corpus of organic text and queried each of the five LMs – BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), GPT, GPT2 and XLNET, to generate pre-trained embeddings. Then, they trained a multilayer perceptron using the embeddings and obtained perfect accuracy in fingerprinting the pre-trained LM. Munir et al. (2021) used stylometric features as well as static and dynamic embeddings using ML classifiers to attribute synthetic text generated by four pre-trained LMs – GPT, GPT2, XLNet, and BART (Lewis et al., 2020). They obtained near perfect accuracy in fingerprinting pre-trained LMs on a purpose-built dataset of synthetic text.

P3: In this paper, we further attempt to attribute authorship of synthetic text generated by fine-tuned LMs. This problem is relevant in the real-world because malicious actors typically fine-tune a pre-trained LM for domain adaption (e.g., to generate fake news articles vs. food reviews) (Zellers et al., 2019). There are two main novel aspects of **P3** that make it more challenging than **P1** and **P2**. *First*, LMs can be fine-tuned in numerous different ways. More specifically, an attacker can use various datasets to fine-tune a pre-trained LM and further set the fine-tuning parameters and epochs in many different ways. Therefore, the size of the universe of fine-tuned LMs is expected to be quite large in **P3**. In **P1**, the problem is a simple binary classification problem (organic vs. synthetic). Similarly, in **P2**, the problem has still limited number classes because only a handful of pre-trained LMs are publicly available (e.g., GPT, GPT2, etc.). For instance, Munir et al. (2021), Pan et al. (2020), and Uchendu et al. (2020) respectively considered a universe of four, five, and eight pre-trained LMs. In contrast, in this paper, we consider 108 fine-tuned LMs. *Second*, prior work mostly considered attributing authorship of synthetic text generated in a controlled setting. For example, synthetic text is often generated by providing a topical prompt (Uchendu et al., 2020). As another example, the attribution classifier often assumes some information about the training data (Zellers et al., 2019). In contrast, here we consider the problem of attributing synthetic text in an uncontrolled setting assuming no control or knowledge about training or generation phases of fine-tuning LM.

3 Dataset

Prior work on detection and attribution of synthetic text has relied on artificial purpose-built datasets in a controlled environment. We overcome this issue by gathering a real-world dataset of synthetic text by different GPT2 bots on the r/SubSimulatorGPT2. Note that r/SubSimulatorGPT2 is independently designed and operated by its moderators. Each user on the r/SubSimulatorGPT2 subreddit is a GPT2 small (345 MB) bot that is fine-tuned on 500k posts and comments from a particular subreddit (e.g., r/askmen, r/askreddit, r/askwomen). The bots generate posts on r/SubSimulatorGPT2, starting off with the main post followed by comments (and replies) from other bots. The bots also interact with each other by using the synthetic text in the preceding comment/reply as their prompt. In total, the subreddit contains 401,214 comments posted between June 2019 and January 2020 by 108 fine-tuned GPT2 LMs (or class). The complete details of various design choices are described here: `https://www.reddit.com/r/SubSimulatorGPT2Meta/`

Table 1 lists some representative examples of the synthetic text generated by three different fine-tuned GPT2 LMs on our dataset. We note that the synthetic text is fairly coherent and also captures the unique vocabulary and style of the subreddit used for fine-tuning. For example, the excerpt from r/conspiracy reads like a conspiracy discussion, the excerpt from r/confession mentions a suggestive reply to the main post, and the excerpt for r/wallstreetbets uses the specialised finance terms like “puts”.

Next, we quantitatively compare and contrast synthetic and organic texts corresponding to different subreddits. To this end, we collect organic text from the corresponding subreddits. Specifically, we randomly sample 1,000 comments from each subreddit class of synthetic and organic text. We contrast basic lexical characteristics, vocabulary, and readability of synthetic and organic text.

Subreddit	Synthetic text excerpt
r/conspiracy	I’m sure the elite have been working to control what they do, but I don’t think they have the resources or manpower to control something as massive as the global economy.
r/confession	You need to tell her how you feel! She needs to know how you feel so that you can work out how to make the right decision. You can tell her how much you hate being a coward and how you’d never be able to live up to your promise.
r/wallstreetbets	There’s a huge amount of volatility right now that I don’t know how well my eyes can handle. I’m not going to buy any puts for the next week.

Table 1: Excerpts of synthetic text generated by GPT2 LMs fine-tuned on different subreddits.

Lexical. First, we contrast the following basic lexical features: average/standard deviation of number of words/sentences per comment. We also measure the Pearson correlation coefficient (ρ) between pairs of synthetic and organic texts in Figures 2a to 2d. We note a high correlation ($\rho > 0.83$) across all lexical features. Thus, we conclude that there is a strong dependency between the lexical characteristics of the synthetic and organic text. This finding indicates that *synthetic text generated by fine-tuned GPT2 models indeed captures the lexical characteristics of the corresponding organic text used for fine-tuning*.

Vocabulary. Second, we compare the vocabularies of synthetic and organic text of each class. We do some basic pre-processing: lower-case, tokenize, and lemmatize all words, remove all punctuation and emojis, and replace hyperlinks and numbers with standard tags. Figure 2e compares the vocabulary size of synthetic and organic text. While organic text seems to have a large vocabulary than synthetic text, we note a strong correlation ($\rho = 0.76$) between their vocabulary sizes. We further compute Jaccard similarity to measure pair-wise vocabulary overlap between different synthetic and organic texts. Figure 3 visualizes the similarity matrices between all pairs classes – synthetic-synthetic, organic-organic, and synthetic-organic. It is noteworthy that the left

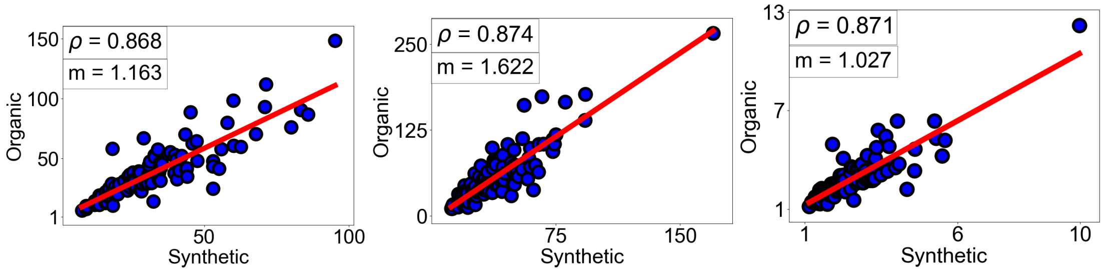

(a) Average #Words

(b) SD Words

(c) Average #Sentences

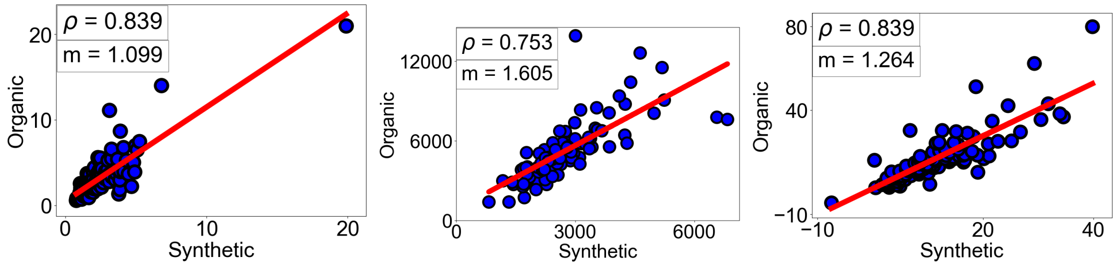

(d) SD Sentences

(e) Vocabulary Size

(f) KF Readability

Figure 2: A class-wise comparison of organic and synthetic text indicating strong correspondence in terms of: (a) average number of words; (b) standard deviation of number of words; (c) average number of sentences; (d) standard deviation of number of sentences; (e) vocabulary size; and (f) Kincaid-Flescher readability. ρ is the Pearson correlation coefficient and m is the slope of the linear fit.

diagonal represents much higher overlap between corresponding pairs of synthetic and organic text classes, even across synthetic-organic text classes. This finding indicates that the *fine-tuned GPT2 models indeed pick up the vocabulary from the organic text in the corresponding subreddit*. It is also noteworthy that the average vocabulary overlap among synthetic text classes in Figure 3a is much higher than that among organic text classes as shown in Figure 3b. This indicates that *synthetic text vocabulary combines information from both pre-training and fine-tuning* – the text corpus used to pre-train the base GPT2 model as well as the text corpus from the particular subreddit used to fine-tune it.

Readability. Finally, we compare the readability of the synthetic and organic text. Figure 2f compares the Kincaid-Flescher readability score (Kincaid et al., 1975) of synthetic and organic text. We note that the average readability of synthetic text (16.8) is less than that of organic text (11.5). This observation corroborates the findings in recent prior work (Uchendu et al., 2020). Similar to lexical and vocabulary analysis, we note a strong correlation ($\rho = 0.84$) between the readability of synthetic and organic text.

Additionally, using the organic text as reference, we measure the quality of the synthetic text using well-known metrics – METEOR (Banerjee and Lavie, 2005) and BLEU (Papineni et al., 2002). We observe that the synthetic text achieves high average scores across all the metrics. We include the results in the appendix.

Overall, we have two key takeaways: (1) synthetic text is *coherent* although with lower overall readability than organic text; (2) synthetic text captures the *characteristics of the organic text used to fine-tune it*.

4 Methods

Fingerprinting fine-tuned LMs needs to operate under the following realistic assumptions. (i) Fingerprinting methods cannot assume any knowledge about the nature of the LM, fine-tuning (e.g., parameters, layers) or generation (e.g., prompt). (ii) They also cannot assume any access to the organic text used for LM fine-tuning. (iii) Since a LM can be fine-tuned in a myriad of ways, these methods need to consider a large number of classes. (iv) These methods are assumed to have access to a limited sample of synthetic text generated by each of the potential fine-tuned LMs.

4.1 Pre-processing

We lower cased the comments and replaced all hyperlinks with a standard tag [*LINK*]. Next, we to-

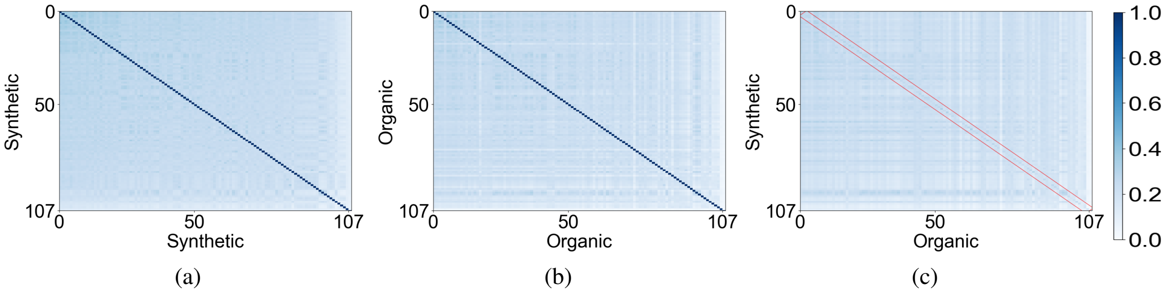

Figure 3: Pair-wise vocabulary overlap between classes of (a) synthetic text, (b) organic text, and (c) cross-comparison of organic and synthetic text. The higher intensity in (a) indicates more vocabulary overlap between classes of (a) synthetic text as compared to (b) organic text. The dark diagonal (highlighted in red box) in (c) indicates significant vocabulary overlap between the synthetic and organic text from the same subreddit.

kenized the comments. Any comment with 5 or fewer tokens was removed. The maximum number of tokens in a comment is limited to 75. In case a comment is larger, only the first 75 tokens in a comment are taken into consideration. This is consistent for all the models we experimented with. The limit was decided based on the limit of the size of the GPU (11GB) used for fine-tuning GPT2 and RoBERTa models.

4.2 Features

We consider several different feature extraction architectures to encode the synthetic text into representation vectors. As we discuss later, these representation vectors are then fed to classifiers.

Writeprints: Writeprints feature set has been widely used for authorship attribution (Iqbal et al., 2008; Pearl and Steyvers, 2012; Abbasi and Chen, 2006). We extract a total of 220 features that include *lexical* (e.g., total number of words, total number of characters, average number of characters per word, digits percentage, upper case characters percentage), *syntactic* (e.g., frequencies of function words, POS tags unigram, bigrams, trigrams), *content-based* (e.g., bag of words, bigrams/trigrams) and *idiosyncratic* (e.g., misspellings percentage) features.

GLTR: Gehrmann et al. (2019) used pre-trained LMs to extract word likelihood features – word ranks and probabilities. We follow the original approach to average the word probabilities of the text based on pre-trained BERT and GPT2. We also bin the word ranks into 10 unequal ranges. The bins are: [1], [2-5], [6-10], [11-25], [26-50], [51-100], [101-250], [251-500], [501-1000], [> 1000].

Glove: Glove embeddings (Pennington et al., 2014) have been commonly used in large-scale authorship attribution (Ruder et al., 2016). We follow (Kim, 2014; Zhang et al., 2015) to create a representation vector of the size (max # tokens × 100, where max # tokens is set to 75) using Glove.

Pre-trained LMs (GPT2 and RoBERTa): We also extract the embeddings for each comment using the pre-trained GPT2/RoBERTa model. Similar to Nils and Gurevych (2019); Feng et al. (2020); Zhu and de Melo (2020), we take the [CLS] token representation from the last layer to extract the embeddings of size 1×768. The final embeddings are then scaled between the values of [-3, 3] using min-max normalization.

Fine-tuned (FT) LMs (GPT2 and RoBERTa): We add a softmax classification layer to the pre-trained GPT2/RoBERTa model. Then, we fine-tune the LM for the task of sentence classification using the synthetic text in the training dataset. We again extract the embeddings (size = 1×768) by taking the [CLS] token representation from the second last layer. The final embeddings are then scaled between the values of [-3, 3] using min-max normalization.

4.3 Classifiers

Shallow classifiers: We use a probabilistic classifier – Gaussian Naive Bayes (GNB), an ensemble decision classifier – Random Forest (RF), and a feed-forward multilayer perceptron (MLP) across all of the feature representations.

CNN classifier: We experiment with a *stacked* CNN model where convolution layers with different kernel sizes are stacked before the embedding layer (Kim, 2014). In this architecture, we attach two stacked convolution 1D layer, followed by a batch normalisation layer and a max pooling layer.

Architecture	Classifier	Macro		Top-k	
		Prec	Recall	5	10
GLTR	GNB	5.5	4.4	12.9	20.9
	RF	7.8	6.6	12.6	19.0
	MLP	3.6	6.3	15.6	23.7
Writeprints	GNB	8.2	5.8	14.1	21.4
	RF	10.2	8.4	14.9	21.8
	MLP	16.9	14.7	30.8	42.1
GloVE	GNB	19.2	9.3	21.9	31.2
	RF	20.5	16.9	27.1	36.2
	MLP	29.7	27.2	44.4	54.1
	CNN	31.1	26.7	44.2	53.5
GPT2	GNB	24.8	12.4	27.8	37.7
	RF	10.5	7.8	15.8	27.1
	MLP	44.9	29.0	47.5	56.9
	CNN	30.9	28.7	49.1	59.1
RoBERTa	GNB	39.2	15.8	30.8	41.0
	RF	11.1	8.4	16.6	25.8
	MLP	44.0	34.8	54.8	62.5
	CNN	33.5	32.0	53.1	63.0
FT-GPT2	GNB	40.1	37.0	56.9	66.0
	RF	27.6	22.8	34.8	45.2
	MLP	40.2	36.4	55.7	64.0
	CNN	44.6	42.1	60.9	68.9
FT-RoBERTa	GNB	**47.7**	41.5	57.9	64.9
	RF	42.0	36.8	46.9	53.2
	MLP	42.8	41.5	58.2	65.3
	CNN	46.0	**43.6**	**62.0**	**69.7**

Table 2: Performance of multi-class classifiers based on macro Precision (Prec), Recall and top-k accuracy ($k = 5, 10$) for the largest setting of 108 classes.

The output is then fed to two dense layers, the latter of which is a softmax layer.

In addition, we also experiment with other shallow classifiers (SVM, Decision Tree) and two more types of feature generators (fine-tuned GLTR, trainable word embeddings). We report additional results, observations and the hyperparameters for all the models in the appendix.

5 Evaluation

We conduct experiments to evaluate these methods using the real-world Reddit dataset as described in Section 3. Our training, validation, and test sets consist of 800, 100 and 200 synthetic comments respectively from each of the 108 subreddit classes. In total, our training, validation and test sets comprise 86k, 11k and 22k comments respectively. For evaluation, we use macro precision and recall. We also measure top-k accuracy based on the confidence score to assess the accuracy of the classifiers in k ($k = 5, 10$) guesses for 108 classes.

5.1 Results

Table 2 lists the results of different feature representations and classifiers. Overall, classifiers based on fine-tuned LM embeddings perform the best, with RoBERTa slightly outperforming GPT2. Fine-tuned embeddings are successfully able to capture the domain of the organic text the LM is fine-tuned on. To provide further insights into our best performing feature representations, we visualize the feature embeddings of pre-trained and fine-tuned RoBERTa and GPT2. Figure 4 plots the 2D projection of synthetic text (using t-SNE) generated by different LMs that are fine-tuned on various subreddits. Fine-tuned embeddings form more cohesive and separated clusters than pre-trained embeddings. Thus, we conclude that fine-tuning these embeddings is beneficial in attributing synthetic text generated by different fine-tuned LMs. Note that certain clusters are more cohesive and better separated than others. For example, the most distinct cluster is observed for r/wallstreetbets in Figures 4a and 4c for fine-tuned embeddings. However, it is not quite distinct in Figures 4b and 4d for pre-trained embeddings. We also note that some clusters with high topical similarity (e.g., r/science and r/askscience) are closer to each other. On the other hand, some clusters with likely lower topical similarity (e.g., r/socialism and r/conservative) are far apart.

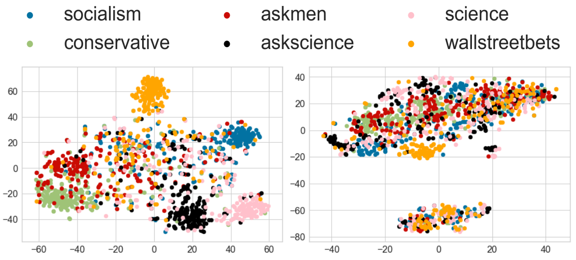

(a) Fine-tuned RoBERTa (b) Pre-trained RoBERTa

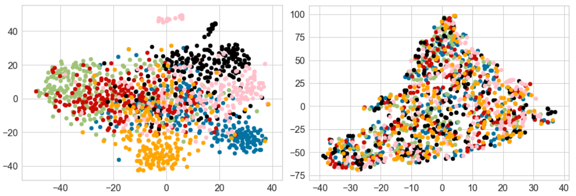

(c) Fine-tuned GPT2 (d) Pre-trained GPT2

Figure 4: Visualisation of fine-tuned (a,c) and pre-trained embeddings (b,d) of specific classes. Closely condensed clusters specific to the domain of the organic text form in the fine-tuned embeddings.

Despite combining word probabilities from both BERT and GPT2, GLTR is ineffective. We find that synthetic texts generated from different fine-tuned models have similar word probabilities because perhaps they are more impacted by the pre-training process rather than the subsequent fine-tuning. This shows that the classifier that performs well for distinguishing between organic and synthetic text (P1) does not work well

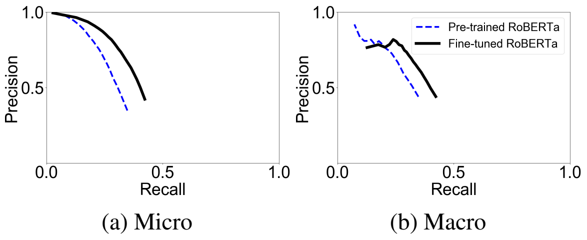

(a) Micro (b) Macro

Figure 5: (a) Micro and (b) Macro *precision-recall trade-off* by varying the gap statistic threshold. The comparison with *all baselines* is included in the appendix.

for distinguishing between synthetic text by different fine-tuned LMs (P3). Writeprints feature set provides some improvement but is still ineffective. Our finding corroborates Manjavacas et al. (2017), who reported that linguistic and stylistic features as used in Writeprints are not effective in distinguishing between synthetic text. GloVE again offers some improvement over Writeprints but its performance remains significantly worse than that of our best performing method.

Overall, fine-tuned RoBERTa embeddings with CNN performs the best with 46.0% precision and 43.6% recall. In about 44% of the cases, this classifier can correctly fingerprint fine-tuned LM amongst 108 classes; in about 70% of cases the correct prediction is one of the top-10 guesses. It is noteworthy that Random Forest which performed exceedingly well in prior work on fingerprinting pre-trianed LMs (Uchendu et al., 2020), does not perform well for fingerprinting fine-tuned LMs. Surprisingly, a relatively simple classifier like GNB achieves comparable precision and recall for our top guess. However, CNN outperforms GNB by a small margin, achieving the best top-10 accuracy of 69.7% for FT-RoBERTa.

5.2 Discussion

Next, we analyze the performance of our best performing RoBERTa feature representation and classifier (CNN) under different conditions.[1]

Precision-Recall trade-off. We evaluate the precision-recall trade-off by imposing a threshold on the confidence of our prediction. To this end, we use the *gap statistic*, defined as the difference between the probability of the highest and second highest prediction (Narayanan et al., 2012). If the gap statistic is lower than our threshold, the classifier chooses to not make a prediction for the test sample. This invariably has an impact on precision and recall. Typically, the precision of the classifier increases, since it can more accurately predict the correct class for the samples it has a high confidence in. Due to certain samples not being predicted for, recall is expected to decrease. Note that since the classifier may make different number of predictions across classes, micro and macro precision/recall could be different.

Figures 9a and 9b respectively plot the micro and macro precision/recall as we vary the gap statistic. Overall, the classifier using FT-RoBERTa embeddings achieves a better precision-recall trade-off compared to using standard RoBERTa embeddings. As expected, precision improves at the expense of recall for larger values of gap statistic. Micro precision increases 46% to over 87% with a reduction in the micro recall from 43% to 27%. Similarly, despite potential class imbalance, macro precision increases 46% to over 81% with a reduction in the micro recall from 43% to 26%. Thus, we conclude that the confidence of our best performing fingerprinting method can be tuned to achieve very high precision with some compromise on recall.

Impact of number of training samples. Next, we evaluate the impact on the performance of our best models by varying training size from 50 to 800 samples per class. As we vary the training data, we keep the same test set, i.e., 200 samples from each class. Figures 8a and 8b, respectively show that precision and recall of FT-RoBERTa plateau at around 400 samples per class. Despite at twice the training data, using training 800 samples per class has similar precision/recall as compared to using 400 training samples per class. We conclude that having more training samples of synthetic text may not always lead to a significant improvement in fingerprinting performance.

Impact of number of classes. We further vary the number of classes from 10 to 108 and report the performance of our best models. Figures 8c and 8d show that for a 10-class problem, FT-RoBERTa + CNN achieves 63.0% precision and 61.7% recall. As expected, both precision and recall decrease as the number of classes increases. For 108 classes, the same classifier achieves 46.0% precision and 43.6% recall. This indicates that fingerprinting a fine-tuned LM is highly challenging when the universe of potential fine-tuned LMs is larger.

---

[1] Other baseline results are reported in the appendix.

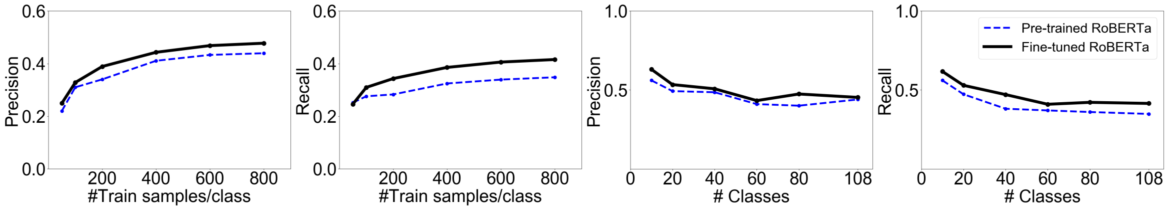

(a) (b) (c) (d)

Figure 6: Comparison between the performances of pre-trained and fine-tuned RoBERTa by varying different parameters. (a) Precision and (b) Recall with the varying training size. (c) Precision and (c) Recall with the varying number of classes. Overall, fine-tuned RoBERTa outperforms pre-trained RoBERTa. The comparison with *all baselines* is included in the appendix.

6 Conclusion

In this paper, we studied the problem of attribution of synthetic text generated by fine-tuned LMs. We designed a comprehensive set of feature extraction techniques and applied them on a number of different machine learning and deep learning pipelines. The results showed that the best performing approach used fine-tuned RoBERTa embeddings with CNN classifier. Our findings present opportunities for future work on fingerprinting LMs in even more challenging open-world scenarios, where the list of potential LMs might be incomplete or synthetic text is not available to train attribution classifiers.

Ethics and Broader Impact

Any biases found in the gathered dataset are unintentional, and we do not intend to do harm anyone. We would also like to acknowledge that the use of large-scale transformer models requires non-trivial compute resources (GPUs/TPUs) that contributes to global warming (Strubell et al., 2019). However, we did not train the models from scratch and simply fine-tuned them for our work.

Acknowledgement

The work was partially supported by the Ramanujan fellowship and the Infosys Centre for AI, IIIT-Delhi.

References

Ahmed Abbasi and Hsinchun Chen. 2006. Visualizing authorship for identification. In *International Conference on Intelligence and Security Informatics*, pages 60–71. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 1–15.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7059–7069.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, pages 1–25.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic BERT sentence embedding. *arXiv preprint arXiv:2007.01852*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822.

Farkhund Iqbal, Rachid Hadjidj, Benjamin CM Fung, and Mourad Debbabi. 2008. A novel approach of mining write-prints for authorship attribution in e-mail forensics. *digital investigation*, 5:S42–S51.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Enrique Manjavacas, Jeroen De Gussem, Walter Daelemans, and Mike Kestemont. 2017. Assessing the stylistic properties of neurally generated text in authorship attribution. In *Proceedings of the Workshop on Stylistic Variation*, pages 116–125.

Shaoor Munir, Brishna Batool, Zubair Shafiq, Padmini Srinivasan, and Fareed Zaffar. 2021. Through the Looking Glass: Learning to Attribute Synthetic Text Generated by Language Models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319.

R Nils and I Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China*, pages 3–7.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Lisa Pearl and Mark Steyvers. 2012. Detecting authorship deception: a supervised machine learning approach using author writeprints. *Literary and linguistic computing*, 27(2):183–196.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training (2018). *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395.

Qicai Wang, Peiyu Liu, Zhenfang Zhu, Hongxia Yin, Qiuyue Zhang, and Lindong Zhang. 2019. A text abstraction summary model based on bert word embedding and reinforcement learning. *Applied Sciences*, 9(21):4701.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763.

SoYeop Yoo and OkRan Jeong. 2020. An intelligent chatbot utilizing bert model and knowledge graph. *Journal of Society for e-Business Studies*, 24(3).

Shi Yu, Yuxin Chen, and Hussain Zaidi. 2020. A financial service chatbot based on deep bidirectional transformers. *arXiv preprint arXiv:2003.04987*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9054–9065.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Xunjie Zhu and Gerard de Melo. 2020. Sentence analogies: Exploring linguistic relationships and regularities in sentence embeddings. *arXiv preprint arXiv:2003.04036*.

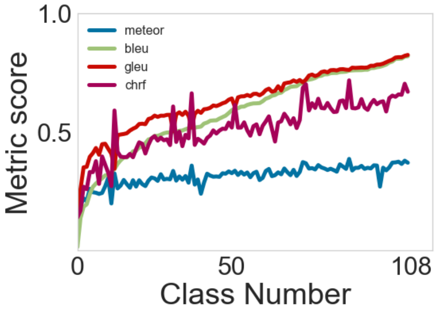

Figure 7: Comparison of different metrics on the synthetic corpus taking the reference of the organic corpus. The classes are sorted by increasing CHRF scores.

A Appendix

A.1 Implementation, Infrastructure, Software

We run all experiments on a 24-core machine with two Intel(R) Xeon(R) Silver 4116 CPU@2.10GHz CPU's and 512 GB RAM. Additionally, the server has a GeForce RTX 2080 Ti (11 GB) GPU card. We use huggingface, pytorch (1.4.0) and Tensorflow (v.1.23.0) to implement and evaluate all deep learning models in the paper. For classical ML, we utilize scikit-learn (v.0.23.1). All implementations are done using Python language (v.3.7).

A.2 Hyper-parameters

Fine-tuned RoBERTa + CNN: We attach 2 CNN convolutional 1D of kernel sizes 2 and 3 layers back to back. The stride for both the layers is 1. The number of output filters for each of the two convolutional layers is 16. We then attach a batch normalization layer of size 16. This is followed by a max pooling layer of size 2 with stride 1. We fine-tuned the model for 15 epochs and monitored the validation loss. If the validation loss did not improve for 5 consecutive epochs, we stopped the fine-tuning.

Fine-tuned RoBERTa + Dense: We used RoBERTaForSequenceClassification wrapper from the huggingface module, which attaches a softmax layer on top of pre-trained RoBERTa model. We extracted the embeddings of the second to last layer of size 1x768. For all of our models, we used a *AdamW* optimizer with a learning rate of 0.00005. We used batch size of 48 and only picked the first 75 tokens of each token. We did not use any padding.

For the soft classifiers, using the fine-tuned RoBERTa embeddings, we obtained the best results with the following parameters -

SVM: C = 0.1, cachesize=200, classweight = None, coef0 = 0.0, degree=3, gamma='scale', kernel='linear', tol=0.001

MLP: activation='relu', alpha=0.01,epsilon=1e-08, hiddenlayersizes = 64, learningrate='adaptive', learningrateinit=0.0001

RF - criterion='entropy', maxdepth=None, maxfeatures='auto', maxleafnodes=None, minsamplesleaf = 1

DT - criterion = 'entropy', maxdepth = None, maxfeatures = None, maxleafnodes=None, minsamplesleaf = 1

GNB - Default sklearn parameters performed the best

A.3 Running Time

All fine-tuned models took a maximum of 8 minutes per epoch. For the soft classifiers, MLP and SVM took the maximum time. Due to the large size of the dataset and the large embedding space, SVM took about 6 hours to train. MLP took an average of 3 hours. Rest of the classifiers took less than an hour for training.

A.4 Additional Results

A.4.1 Additional Dataset Analysis

Besides the analaysis in the dataset section of the main text, we used 4 metrics - METEOR, BLEU, GLEU and CHRF for measuring the coherency of the synthetic text, using the organic text as reference. Using 1000 comments from each class of synthetic and organic text, we obtained high class-wise average scores on all metrics - METEOR (0.31), BLEU (0.58), GLEU (0.63) and CHRF (0.51). This provides further evidence that synthetic text is coherent and readable when compared to its organic counterpart. In figure 7 , we illustrate the scores for each individual class, sorted by increasing CHRF scores.

A.4.2 Feature extraction

Besides the feature extraction methods we mentioned in the Methods section of the main text, we also experimented with two more methods:

1. **Fine-tuned GLTR:** Using the training set, we fine-tuned separate BERT and GPT2 models for each class for the task of mask completion. All models were then used for extracting GLTR word likelihood features for the complete training and test sets. Subsequently, the training representations were then fed to a sequential neural classi-

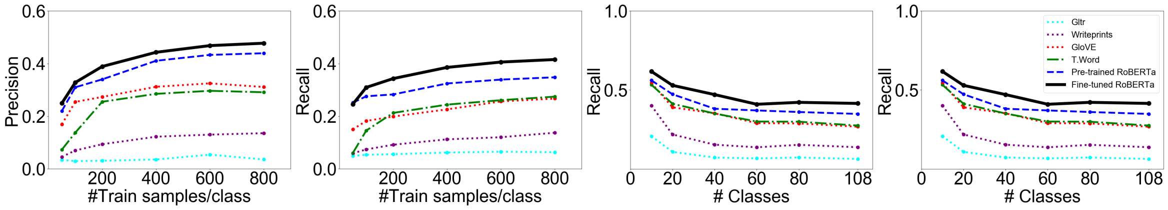

(a) (b) (c) (d)

Figure 8: A comparison between the performances of various embeddings by varying problem parameters. (a) Precision and (b) Recall with the varying training set size. (a) Precision and (b) Recall with the varying number of classes. Overall, fine-tuned RoBERTa outperforms all the baselines.

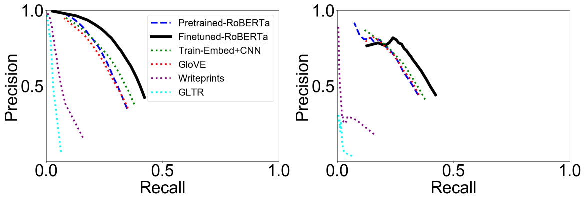

(a) Micro (b) Macro

Figure 9: (a) Micro and (b) Macro *precision-recall trade-off* by varying the gap statistic threshold.

Base Architecture	**Classifier**	**Macro**		**Top - 5**	**Top - 10**
		Precision	**Recall**		
Writeprints	DT	6.9	6.6	6.6	6.6
	SVM	19.3	16.3	33.2	44.8
GLTR	DT	5.7	5.5	5.57	5.58
	SVM	7.3	7.0	10	13.1
RoBERTa	DT	6.3	5.3	6.3	6.3
	SVM	8.3	6.9	7.8	8.9
Trainable-Word	MLP	20.4	18.9	34.3	44.3
	CNN	28.6	27.0	44.0	53.1
FT RoBERTa	Dense	44.0	42.3	60.8	68.9
	DT	29.5	28.7	28.7	28.7
	SVM	42.7	41.1	58.3	65.6

Table 3: Results of Decision Tree (DT) and SVM with the embeddings. We also include the results of the trainable word embeddings model.

fier like Bi-LSTM. The intuition was that word likelihoods extracted using the class's fine-tuned GLTR model would be high for synthetic text generated for the class's language model. For example, a r/wallstreetbets fine-tuned GLTR feature extractor would extract higher word likelihoods, as compared to other GLTR models, from a synthetic comment generated by the language model fine-tuned on r/wallstreetbets. However, for an extremely small setting of 5 classes, we only obtained a precision and recall of around 33% each. Due to the (a) expensive cost of fine-tuning 108 BERT and GPT2 models, (b) expensive cost of extracting GLTR features for a dataset of 100k examples, (c) extremely poor results of the approach on a small setting, we did not continue with experimenting the model in a larger setting.

2. **Trainable word embeddings** - We tokenized and represented each comment using an allocated set of integers based on the vocabulary of the training set. Then, similar to the GloVE feature extraction, we passed the representation into a trainable word embedding layer, followed by MLP or CNN. The results for these were slightly worse than that for the GloVE features. We report the results in Table 3.

Additionally, for all the feature representations mentioned in the main text, we tested SVM and decision tree. Overall, they were outperformed by the other classifiers with one notable exception. For Writeprints features, SVM showed the best results of precision and recall of 19.3% and 13.3% respectively. We report the results in Table 3.

A.5 Other baselines

Precision-recall trade-off. For the precision-recall trade-off in the results section of the main text, we presented a comparison with additional baselines in Figure 9. Fine-tuned RoBERTa performs the best among all methods for both micro and macro precision-recall trade-offs. They are followed in a decreasing order by the pre-trained RoBERTa embeddings, the trainable word embeddings, the GloVE word embeddings, Writeprints, and GLTR respectively.

Training set size and classes. We report the comparison with all other baselines for the varying training size and the number of classes in Figure 8. Similar to the precision-recall trade-off, the fine-tuned RoBERTa embeddings perform the best, followed by pre-trained RoBERTa embeddings, GloVE word embeddings, trainable word embeddings, Writeprints, and GLTR.

Analyzing Code Embeddings for Coding Clinical Narratives

Wei Shi[1] **Jiewen Wu**[2]* **Xiwen Yang**[3] **Nancy F. Chen**[1]

Ivan Ho Mien[1,4] **Jung-jae Kim**[1] **Pavitra Krishnaswamy**[1]

[1]Institute for Infocomm Research, A*STAR, Singapore [2]Huawei Technologies Co., Ltd, China

[3]A*STAR Artificial Intelligence Initiative, Singapore [4]National Neuroscience Institute, Singapore

{shi_wei, nfychen, Ivan_Ho, jjkim, pavitrak}@i2r.a-star.edu.sg

wujiewen@huawei.com xiwenyang22@gmail.com

Abstract

Medical professionals review clinical narratives to assign medical codes as per the International Classification of Diseases (ICD) for billing and care management. This manual process is inefficient and error-prone as it involves a nuanced one-to-many mapping. Recent works on automated ICD coding learn mappings between low-dimensional representations of the reports and the codes. While they propose novel neural networks for encoding varied types of information about the codes, it is unclear as to what information in the medical codes is helpful for performance improvement and why. Here, we compare different ways to represent, or embed, the codes based on their textual, structural and statistical characteristics, using a single deep learning baseline model in quantitative evaluations on discharge reports from the MIMIC-III Intensive Care Unit database. We also qualitatively analyse the nature of the cases that benefit most from the code embeddings and demonstrate that code embeddings are important for predicting ambiguous and oblique codes.

1 Introduction

Free-text clinical narratives contain the majority of information pertaining to patient state, disease progression and care management. Following a patient encounter, the text reports from the visit are codified by representing the key diagnoses and procedures according to the International Classification of Diseases (ICD) system (Medicode (Firm), 1996). The resulting ICD codes are used for a variety of diagnostic, billing, epidemiology and research purposes (Bach and First, 2018; Feder et al., 2018; Alsentzer et al., 2019).

The process of ICD coding, i.e., mapping clinical text reports to ICD codes, is challenging. It involves processing diverse domain-specific text with large vocabulary and significant irrelevant content to make a nuanced choice of a small set of codes from a high-dimensional taxonomy of 15,000 ICD codes. Hence, manual ICD coding tends to be time-intensive, costly, and error-prone (Lang, 2007; Shi et al., 2017; Xie and Xing, 2018), and there is great interest in automated ICD coding methods.

Previous works on automated ICD coding have employed conventional rule-based or machine learning methods (Larkey and Croft, 1996; Farkas and Szarvas, 2008; Perotte et al., 2014). Recently, deep learning methods (Baumel et al., 2017; Xie and Xing, 2018; Nie et al., 2018; Mullenbach et al., 2018; Vu et al., 2020; Cao et al., 2020; Teng et al., 2020; Yuan et al., 2020) have achieved leading-edge performance. Of these, the best performing deep learning approaches typically employ attention mechanisms to use representations of the ICD codes to guide the model's predictions. However, the specific representations of the ICD codes used vary from code textual descriptions (Mullenbach et al., 2018) and code hierarchy (Vu et al., 2020; Cao et al., 2020) to code co-occurrences (Cao et al., 2020) and graph of medical entities associated with codes (Teng et al., 2020; Yuan et al., 2020). Yet, it is unclear which ICD code representation is most effective, what types of cases would benefit from these representations, and why.

Addressing these gaps requires comparing different code embeddings within one united framework. We introduce a simple attention mechanism to leverage varied statistical, textual, structural representations of ICD codes and enhance a predefined baseline clinical notes classifier. We use discharge reports within the benchmark MIMIC-III Intensive Care Unit database (Johnson et al., 2016) for comparative evaluation, and perform extensive experiments to characterize effects of dif-

---

[*] This work was done while the author was at A*STAR.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4665–4672
August 1–6, 2021.

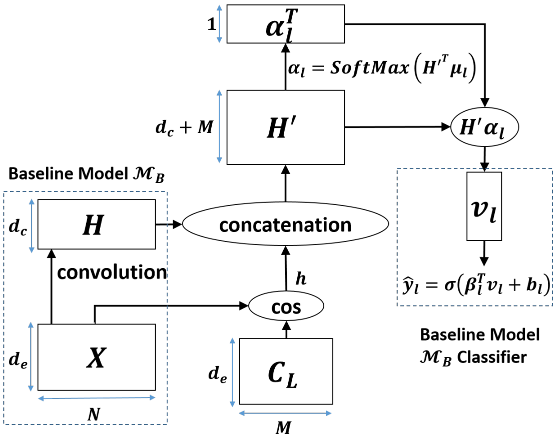

Figure 1: Proposed model architecture

ferent code embeddings on prediction performance. Quantitative results show that our proposed attention mechanism (a) enables 7-9% micro-F-1 boost over the baseline classifier, and (b) performs at least as accurately as more advanced two-level attention, hyperbolic embedding or graph convolutional network approaches. We further perform qualitative analyses and show that our attention network enables large improvements when the coding task is more ambiguous or nuanced. Our approach and findings offer practical means to enhance performance in nuanced text classification tasks.

2 Methods

The task entails mapping a given free-text discharge report to a set of ICD codes. This is a multi-label text classification problem. We propose an approach for learning varied textual, structural and statistical representations of the ICD codes (i.e., code embeddings), and employing them to enhance performance of a given baseline model.

2.1 Attention to Code Embeddings

Figure 1 illustrates the architecture. We start with a given baseline model $\mathcal{M}_B$ based on a convolutional layer. $\mathcal{M}_B$ takes word embeddings $X \in \mathbb{R}^{d_e \times N}$ of words in a given report as input and learns to generate their hidden representation $H \in \mathbb{R}^{d_c \times N}$, where N is the length of input medical narratives after padding, d_e is the input embedding size, and d_c is the number of filters. We propose code embeddings $C_L \in \mathbb{R}^{d_e \times M}$ as an auxiliary for $\mathcal{M}_B$, where M is the number of ICD codes. We compute the cosine similarity between X and C_L, and denote the result as h. We then compute per-label attention weights α_l as follows:

$$\alpha_l = \mathrm{SoftMax}(\begin{bmatrix} H \\ h \end{bmatrix}^{\mathrm{T}} \mu_l), \qquad (1)$$

where [ ] indicates concatenation (denoted as H') and $\mu_l \in \mathbb{R}^{d_c+M}$ is a vector parameter for label l. Weights α_l denote attention from the note representation to the code representation for label l and can be used to enhance performance of $\mathcal{M}_B$ (e.g., as shown in Figure 1). Then we apply the attention weights α_l to H' to get the final representation v_l of an input report corresponding to label l. We also adopt the same classifier as $\mathcal{M}_B$, which uses a linear layer and a sigmoid transformation, as illustrated in the right dotted box, where β_l and b_l are the weight and bias of the classification layer for label l, respectively. $\hat{y}_l$ is the binary classification probability that X belongs to l.

We set both the word embeddings for text input and the code embeddings for ICD codes as non-trainable to give the best performance. Our proposed method introduces only a small number of learnable parameters for labels.

2.2 Code Embeddings

Each ICD code has a unique identifier and a text description and is structurally situated in a tree hierarchy. Further, based on the reports labelled with any given ICD code, we can obtain sample statistics of the code usage. We propose to learn embeddings or representations that capture the above textual, structural, and statistical characteristics of ICD codes, as described below.

Textual code embeddings are obtained by either (a) averaging word vectors (Mikolov et al., 2013) of the words in the description of a code (denoted as CE-w2v) or (b) learning the contextual representation of the code description with BERT (Devlin et al., 2019) (denoted as CE-BERT). For CE-w2v, we use gensim[1] to train the word vectors with discharge reports. For CE-BERT, we use Keras BERT[2] uncased large model to get contextualized word representations, apply max pooling to all the word representations and then add a linear layer for dimension reduction to get code representations with 100 dimensions. This is integrated end-to-end into our model.

Structural code representations leverage the

---

[1] `https://github.com/RaRe-Technologies/gensim/blob/develop/gensim/models/word2vec.py`

[2] `https://github.com/CyberZHG/keras-bert`

ICD tree hierarchy. We capture the parent-child and sibling relations in triples (e.g., (401, *ParentOf*, 401.1) or (401.1, *SiblingOf*, 401.9). We then feed the triples into a knowledge graph embedding approach such as TransR (Lin et al., 2015) (denoted by CE-TransR).

Statistical code representations are learned from the sample statistics between ICD codes and the discharge reports from the training dataset. We designate the embedding of code l as the weighted average of word vectors, as follows:

$$C_l = \frac{1}{N} \sum_{i=1}^{N} v_{w_i} \sum_{d \in docs(l)} tf(w_i, d), \qquad (2)$$

where the weight of a word vector v_{w_i} is proportional to the sum of term frequencies of the word in the notes that are labelled with the code l; N indicates the size of the dataset vocabulary; docs(l) refers to the set of notes associated with code l; and tf(w_i, d) is the function that returns the term frequency of the word w_i in document d. We apply smoothing with increasing all word counts by one, and denote resulting embeddings as CE-Stat.

3 Experiments

We follow the recent state-of-the-art (SOTA) ICD coding studies and perform experiments on the benchmark Medical Information Mart for Intensive Care-III (MIMIC-III) dataset (Johnson et al., 2016). Specifically, we implement our proposed code embeddings (denoted as CE-xxx) atop the popular CAML baseline (Mullenbach et al., 2018). Note that our approach is amenable to any baseline of choice.

Data: Like previous works (Mullenbach et al., 2018; Vu et al., 2020), we focus on multi-label classification task of mapping the discharge reports in the MIMIC-III dataset to ICD codes. Preprocessing details are listed in Appendix A.1. The resultant preprocessed dataset, termed as FULL, has over 52,700 discharge reports associated with subsets of over 8,929 ICD codes (unlike the 8,921 ICD codes reported in prior works). We evaluated our approach on the FULL dataset.

As our focus was to understand what information in ICD codes enables performance improvement, we also investigated whether and to what extent the choice of a code subset affects performance. Therefore, we created new subsets of MIMIC-III (termed sub-datasets) for further evaluation. Specifically, we selected the top *k* frequent ICD codes in the FULL MIMIC-III dataset and collated the subset of discharge reports tagged with at least one of the top *k* frequent codes. We term the sub-datasets as Top-k for k=20, 50, 100 and 300.

Finally, we also evaluated our approach on the more widely used subselection of top-50 codes (termed as Top-50$^+$) (Shi et al., 2017). We note that the Top-50$^+$ dataset is much smaller than the other Top-k and FULL datasets because it excludes reports without associated diagnosis descriptions. The detailed breakdown of the dataset sizes and splits are showed in Appendix A.2.

Evaluations: We evaluate performance against two baseline models (i.e., $\mathcal{M}_B$): (a) CAML which uses a per-label attention mechanism within a convolutional neural network (CNN) classifier and (b) DR-CAML which uses code embeddings to constrain the learned model parameters of CAML (Mullenbach et al., 2018). We provide all parameters and model tuning details of the proposed method in Appendix A.3. We follow prior works and report micro-F1 to evaluate model performance, and showcase detailed comparisons for other common metrics. For each experiment, we report averages from 3 independent runs.

Comparative Results on Top-k Sub-Datasets: Table 1 shows the performance of our CE approach compared with baselines on the 5 MIMIC-III sub-datasets. Our CE approach (any embedding type) outperforms the baselines in all the Top-k sub-datasets. We observe that CE-w2v, CE-BERT and CE-TransR lead to slightly better performance than CE-Stat. CE also obtains comparable results on the FULL dataset compared to the baselines. As prior works did not focus on understanding the relation between information in the codes and model performance, there are no reported results on our Top-20, Top-50, Top-100, and Top-300 datasets. Thus, we only compared with baselines in Table 1.

We highlight that our experiments on the FULL dataset were limited by the memory size of the GPUs used. To address this, we reduced batch size of our method (from 128 to 16) and also applied a linear layer to reduce the number of dimensions (M) from the number of FULL codes to 50. Consequently, for the FULL dataset, our CE approach does not improve over baselines and SOTA. However, as our results indicate ability to consistently improve over baselines for different datasets, we posit that increasing batch size and allowing attention to focus on all the FULL codes would enable

Data size	Top-20	Top-50	Top-100	Top-300	FULL†
Baseline (CAML)	0.681	0.641	0.599	0.555	**0.520**
DR-CAML	0.668	0.641	0.584	0.543	0.509
CE-w2v	0.768	**0.713**	0.684	**0.635**	0.502
CE-BERT	**0.775**	0.710	**0.689**	0.622	0.518
CE-TransR	0.765	0.710	**0.689**	0.623	0.507
CE-stat	0.765	0.711	0.688	0.614	0.500

Table 1: Evaluation results of all data on micro-F1. The default batch size is 128, while † uses 16 as batch size due to memory limit.

	AUC		F1		
Model	Macro	Micro	Macro	Micro	Precision@5
Yuan et al. (2020)	-	-	-	-	0.635
Teng et al. (2020)	-	0.933	-	0.692	0.653
Vu et al. (2020)	**0.925**	**0.946**	**0.666**	**0.715**	**0.675**
Cao et al. (2020)	0.895	0.929	0.609	0.663	0.632
CE-Best	0.914	0.937	0.637	0.694	0.652

Table 2: Results on MIMIC-III for the most frequent 50 labels (Top-50^+). Based on the performance, CE-Best corresponds to CE-w2v.

our approach to perform comparably with SOTA.

Comparisons with SOTA on Top-50^+: As the Top-50^+ benchmark is the common dataset evaluated in all SOTA works, we tabulate the results of our proposed approach on the Top-50^+ dataset in relation to previously published SOTA results in Table 2. We observe that our approach outperforms all previous methods in terms of macro-/micro-averaged F1 and AUC, except for Vu et al. (2020) (Vu et al., 2020). The performance of Vu et al. (2020) (Vu et al., 2020) is slightly higher than ours, as they use a model based on bidirectional long short-term memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) with a similar but more complex attention mechanism. While we also ran experiments with Bi-LSTMs, we found that they tend to be computationally intensive and often did not converge, and thus focused on the more practical CNNs. We further tried to combine code embeddings of different kinds (e.g. CE-w2v + CE-TransR) to see if there is any synergistic effect, but found that no such combination led to performance improvement. We report results of the combination experiments on Top-50^+ in Appendix A.4.

4 Qualitative Analysis

To dissect gains of the code representations, we performed qualitative analyses on the Top-50^+ test results.

Data Selection: For each CE embedding, we computed the per-code micro-F1 gains over baseline CAML, summed the gains across all the CE embeddings, and rank-ordered the ICD codes by total micro-F1 gain. Next, we selected the 10 codes with the highest gains over baseline (CE $\gg$ baseline) and also the 10 codes with the least gains over baseline (CE $\approx$ baseline). For the first selection (those with the highest gains over baseline), our 4 CE methods typically improve over the baseline. For the second selection of the 10 lowest gain codes, CE is almost always as good as the baseline in these cases. Specifically, out of all discharge summaries for the second selection, the baseline outperforms all 4 CE methods in only 0.2% of cases and 2 out of 4 CE methods in only 1.2% of cases. Hence, we term this second selection as "CE $\approx$ baseline". For qualitative review, we randomly sampled 5 cases corresponding to each of these 20 codes from the Top-50^+ testset and obtained 100 cases.

Review Procedure: All qualitative analyses were performed independently by two clinical reviewers. After analysis, the two reviewers discussed to arrive at consensus. First, for each of the 20 codes selected, reviewers considered the ICD coding guidelines and assessed whether they fall into medical, procedural, or surgical categories. Next, for each of the 100 cases selected, reviewers read the discharge reports and marked out reports that did not have any viable information relating to the code assignment for exclusion from further analysis. Second, for reports deemed viable, the

(A) CE >> Baseline

Characteristics of Codes

Code	Type
305.1: Tobacco use disorder	Medical
311: Depressive disorder, not elsewhere classified	Medical
585.9: Chronic kidney disease, unspecified	Medical
038.9: Unspecified septicemia	Medical
403.9: Hypertensive renal disease, unspecified	Medical
250.00: Diabetes mellitus type II or unspecified type, not stated as uncontrolled, w/o mention of complication	Medical
45.13: Other endoscopy of small intestine	Procedural
38.93: Venous catheterization, not elsewhere classified	Procedural
496: Chronic airway obstruction, not elsewhere classified	Medical
412: Old myocardial infarction	Medical

Characteristics of Note-Code Mappings

Mentions	Sparse	Not Sparse
Oblique	5	10
Explicit &Oblique	3	9
Explicit	10	7

(B) CE ≈ Baseline

Code	Type
428.0: Congestive heart failure, unspecified	Medical
39.95: Hemodialysis	Procedural
285.9: Anemia, unspecified	Medical
33.24: Closed [endoscopic] biopsy of bronchus	Procedural
507.0: Pneumonitis due to inhalation of food or vomitus	Medical
427.31: Atrial fibrillation	Medical
36.15: Single internal mammary-coronary artery bypass	Surgical
39.61: Extracorporeal circulation auxiliary to open heart surgery	Surgical
V45.81: Postsurgical aortocoronary bypass status	Surgical
V15.82: History of tobacco use	Medical

Mentions	Sparse	Not Sparse
Oblique	7	4
Explicit & Oblique	2	6
Explicit	10	15

Figure 2: Results of Qualitative Analysis

reviewers assessed whether the reports explicitly delineated the codes (e.g., word-to-word match with code description or synonymous mentions) or contained information that more obliquely relates to the codes (e.g., mentions which might lead a domain expert with specialized knowledge to indirectly infer the code). Third, reviewers further indicated whether the mentions were sparse (1-2 circumscribed mentions) or not (several mentions or extensive sections relating to the code). Finally, reviewers marked out whether the reports had diverse expressions linking to the codes.

Qualitative Analysis Results: Figure 2 details the results. Comparing the code characteristics, we observe that codes where CE gains more tend to (a) have descriptions that include "unspecified" or "not elsewhere classified" and (b) fall into the medical category. In contrast, codes where CE does not gain much tend to be more procedural or surgical in nature. Next, comparing characteristics of the mappings between the notes and the codes, we observe that cases where CE gains more tend to have more oblique mentions; while cases where CE does not gain much tend to have more explicit mentions. This suggests that code embeddings may provide more gains in cases where the discharge reports more obliquely correspond to the code. We detail more in Appendix A.5 and A.6 by providing excerpts from 2 exemplar cases and also showing that CE enables strong Micro-F1 gains on the oblique codes (codes with descriptions including "unspecified" or "not elsewhere classified") of the FULL dataset.

We found that the numbers of cases with sparse mentions were similar for the cases where CE gained more vs. less over baseline. That said, the reviewers did observe that codes such as "Tobacco use disorder" were largely associated with sparse mentions and these kinds of cases were more likely to be accurately predicted with CE than with the baseline. We also note that the cases corresponding to higher gains for CE tended to have more diversity in expression.

5 Conclusions and Future Work

We proposed and characterized methods to leverage representations that capture statistical, textual, and structural properties of medical codes for clinical report coding. We implemented the proposed method on a simple but efficient baseline system and demonstrated substantial performance improvements in micro-F1. Additionally, we performed qualitative evaluation studies to show that our method is more useful in cases when the code prediction task is more ambiguous or nuanced. Future work will experiment with more general datasets and enhancements of the attention network to further improve performance.

Acknowledgments

This research was supported by funding for Digital Health and Deep Learning from the Institute for Infocomm Research (I2R) and the Science and Engineering Research Council (Project No. A1818g0044), A*STAR, Singapore. This work was also supported by resources and infrastructure provided by the A*STAR AI Programme.

References

Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*.

Bo Bach and Michael B First. 2018. Application of the icd-11 classification of personality disorders. *BMC psychiatry*, 18(1):1–14.

Tal Baumel, Jumana Nassour-Kassis, Michael Elhadad, and Noémie Elhadad. 2017. Multi-label classification of patient notes a case study on ICD code assignment. *ArXiv*, abs/1709.09587.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Shengping Liu, and Weifeng Chong. 2020. HyperCore: Hyperbolic and Co-graph Representation for Automatic ICD Coding. In *ACL*, pages 3105–3114.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, pages 4171–418.

R Farkas and G Szarvas. 2008. Automatic construction of rule-based ICD-9-CM coding systems. *BMC Bioinformatics*, 9(Suppl 3):S10.

Shelli L Feder, Nancy S Redeker, Sangchoon Jeon, Dena Schulman-Green, Julie A Womack, Janet P Tate, Roger J Bedimo, Matthew J Budoff, Adeel A Butt, Kristina Crothers, et al. 2018. Validation of the icd-9 diagnostic code for palliative care in patients hospitalized with heart failure within the veterans health administration. *American Journal of Hospice and Palliative Medicine®*, 35(7):959–965.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035.

Dee Lang. 2007. *CONSULTANT REPORT - Natural Language Processing in the Health Care Industry*. Cincinnati Children's Hospital Medical Center.

Leah S. Larkey and W. Bruce Croft. 1996. Combining classifiers in text categorization. In *SIGIR*, pages 289–297.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*, pages 2181–2187.

Medicode (Firm). 1996. *ICD-9-CM: International classification of diseases, 9th revision, clinical modification*. Medicode, Salt Lake City, Utah.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *NAACL-HLT*, pages 1101–1111.

A Nie, A Zehnder, R L Page, Y Zhang, A L Pineda, M A Rivas, C D Bustamante, and J Zou. 2018. DeepTag: inferring diagnoses from veterinary clinical notes. *NPJ Digit Med.*, page 60.

A Perotte, R Pivovarov, K Natarajan, N Weiskopf, F Wood, and N Elhadad. 2014. Diagnosis code assignment: models and evaluation metrics. *J Am Med Inform Assoc.*, 21(2):231–237.

Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P Xing. 2017. Towards automated icd coding using deep learning. *arXiv preprint arXiv:1711.04075*.

Fei Teng, Wei Yang, Li Chen, Lufei Huang, and Qiang Xu. 2020. Explainable Prediction of Medical Codes With Knowledge Graphs. *Frontiers in Bioengineering and Biotechnology*, 8:867.

Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. 2020. A Label Attention Model for ICD Coding from Clinical Text. In *IJCAI*, pages 3335–3341.

Pengtao Xie and Eric Xing. 2018. A neural architecture for automated ICD coding. In *ACL*, pages 1066–1076.

Quan Yuan, Jun Chen, Chao Lu, and Haifeng Huang. 2020. The Graph-based Mutual Attentive Network for Automatic Diagnosis. In *IJCAI*, pages 3393–3399.

A Appendix

A.1 Data Preprocessing

We preprocess discharge reports following CAML. By retaining a maximum of 2,500 words for each summary, we obtain a vocabulary of about 52,000 words. We found there is a minor parsing error in CAML data preprocessing. When CAML read diagnosis and procedure codes from MIMIC III PROCEDURES_ICD.csv and DIAGNOSES_ICD.csv using Python function pandas.read_csv(), the data type of codes used is numpy.int64. In fact, the data type should be str. We correct this error by indicating the data type. For full codes, the number of common codes contained in our produced codes and CAML codes is 8706. For top-50 codes, only one code is different.

A.2 Dataset Splits and Details

Data size	Top-20	Top-50	Top-100	Top-300	FULL	Top-50^+
Train	42,590	44,804	46,458	47,285	47,724	8,067
Dev.	1,471	1,574	1,600	1,622	1,632	1,574
Test	3,054	3,242	3,291	3,352	3,373	1,730

Table 3: Data splits details

Table 3 shows the number of discharge summaries contained in the training, development and test data for all the top-k, full and Top-50^+ data. We can see through sub-selection, Top-50^+ is much smaller than the other data.

A.3 Parameters and Model Tuning

Our code embeddings introduce extra training parameters due to the changes in attention structure. For a Top-k dataset, $2k^2$ more parameters are added. For small k, this number is negligible. For greater k, such as the FULL dataset, we add one fully-connected layer after h to reduce the first k in $2k^2$ to a fixed number, so that the number of the introduced parameters is smaller than the number of original parameters of CAML.

We also tuned the batch size and learning rate to enhance performance. For top-20/50/100/300 data, we use a fixed batch size of 128 in all our models. For Top-50^+ and FULL, we use a fixed batch size of 16 in all our models. Due to the expensive GPU memory cost in cosine matrix computation and the large number of added feature maps, we add a linear layer to reduce the size of cosine matrix. For all datasets, we set learning rate to 0.001. For CAML based methods, we use the settings from (Mullenbach et al., 2018).

A.4 Detailed Evaluation on Top-50^+

We report deeper analyses on the Top-50^+ benchmark in Table 4. We first assess whether CE improves performance over CAML by adding more features to the representation of a discharge report. Specifically, we add 50 filters (thus enhancing number of features) to those used in CAML and DR-CAML, and denote the revised models as CAML_add and DR-CAML_add. We observe that the additional filters offer limited improvements in comparison with the CE approach (any embedding). This suggests that our CE approach may not just be adding more features to improve performance. Next we assess if combining the different CE embeddings would enable even better performance. We experiment with several combinations of our different code embeddings: (a) CE+WT combining CE-w2v and CE-TransR, (b) CE+WTS combining CE-w2v, CE-TransR and CE-Stat, and (c) CE+BTS combining CE-BERT, CE-TransR and CE-Stat. We observe that the combinations of CE embeddings do not improve the performance much over individual CE embeddings. This suggests that dot product of discharge summary representation with concatenation of multiple code representations may not have synergistic effects.

	AUC		F1		
Model	Macro	Micro	Macro	Micro	P@5
CAML	0.870	0.913	0.521	0.614	0.612
DR-CAML	0.870	0.906	0.541	0.612	0.606
CAML_add	0.884	0.920	0.547	0.626	0.621
DR-CAML_add	0.878	0.916	0.546	0.618	0.616
CE-w2v	**0.914**	**0.937**	0.637	**0.694**	0.652
CE-BERT	0.913	0.936	0.638	0.692	0.651
CE-TransR	0.913	**0.937**	0.636	0.693	**0.654**
CE-Stat	0.911	0.936	0.633	0.687	0.651
CE+WT	**0.914**	**0.937**	0.640	0.693	0.647
CE+WTS	0.912	0.935	**0.649**	0.689	0.650
CE+BTS	0.912	0.936	0.644	0.692	0.650

Table 4: Results on MIMIC-III, 50 labels (Top-50^+). P@5 means precision at 5.

A.5 Constrained Evaluations on FULL

Method	Micro-F1
CAML	0.106
DR-CAML	0.106
CE-w2v	0.169
CE-BERT	0.119
CE-TransR	0.114
CE-Stat	0.122

Table 5: Micro-F1 on Oblique Codes of Full

We look into the oblique codes in the testing data of FULL. We select the codes of which the code descriptions containing keywords from [“unspecified”, “not elsewhere classified”, “other”]. Table 5

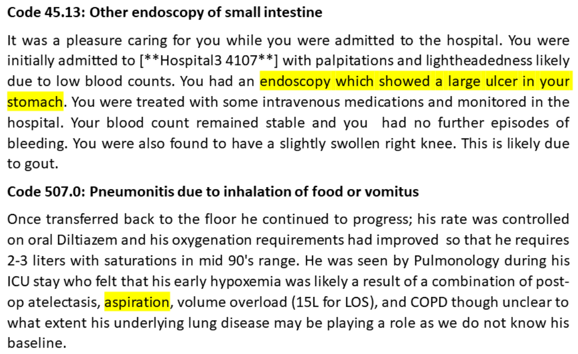

Code 45.13: Other endoscopy of small intestine

It was a pleasure caring for you while you were admitted to the hospital. You were initially admitted to [**Hospital3 4107**] with palpitations and lightheadedness likely due to low blood counts. You had an endoscopy which showed a large ulcer in your stomach. You were treated with some intravenous medications and monitored in the hospital. Your blood count remained stable and you had no further episodes of bleeding. You were also found to have a slightly swollen right knee. This is likely due to gout.

Code 507.0: Pneumonitis due to inhalation of food or vomitus

Once transferred back to the floor he continued to progress; his rate was controlled on oral Diltiazem and his oxygenation requirements had improved so that he requires 2-3 liters with saturations in mid 90's range. He was seen by Pulmonology during his ICU stay who felt that his early hypoxemia was likely a result of a combination of post-op atelectasis, aspiration, volume overload (15L for LOS), and COPD though unclear to what extent his underlying lung disease may be playing a role as we do not know his baseline.

Figure 3: Illustrations of Oblique Coding Cases

shows the Micro-F1 scores of our code embedding methods compared with the baseline methods. From the table, we can see our methods perform better on the oblique codes, especially CE-w2v.

A.6 Case Illustrations

To provide richer insight on the qualitative analysis, we provide two case illustrations, shown in Figure 3. In both cases, the indicated ground truth codes were missed by the baseline but predicted correctly by our CE approach. In the first case (i.e., code 45.13), there are synonym mentions of "EGD" in the Major Surgical Procedure, Images, and Brief Hospital Course subsections of the report. However, indirect phrases on the type of endoscopy performed in the Discharge Instructions imply that this is specifically a case of upper gastrointestinal endoscopy, which leads to the said code assignment. In the second case (i.e., code 507.0), there are no explicit mentions of pneumonitis with vomitus anywhere in the discharge report. However, there is only one oblique mention of "aspiration" without the word pneumonia or its equivalent. As this code is also often termed as "aspiration pneumonia" in medical parlance, the oblique mention ties down the link between the report and the said code assignment.

Automatic Construction of Sememe Knowledge Bases via Dictionaries

Fanchao Qi^{1,2}, **Yangyi Chen**^{2,4*}, **Fengyu Wang**^{1,2}, **Zhiyuan Liu**^{1,2,3},
Xiao Chen^{5}, **Maosong Sun**^{1,2,3†}
[1]Department of Computer Science and Technology, Tsinghua University, Beijing China
[2]Beijing National Research Center for Information Science and Technology
[3]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[4]Huazhong University of Science and Technology [5]Huawei Noah's Ark Lab
qfc17@mails.tsinghua.edu.cn

Abstract

A sememe is defined as the minimum semantic unit in linguistics. Sememe knowledge bases (SKBs), which comprise words annotated with sememes, enable sememes to be applied to natural language processing. So far a large body of research has showcased the unique advantages and effectiveness of SKBs in various tasks. However, most languages have no SKBs, and manual construction of SKBs is time-consuming and labor-intensive. To tackle this challenge, we propose a simple and fully automatic method of building an SKB via an existing dictionary. We use this method to build an English SKB and a French SKB, and conduct comprehensive evaluations from both intrinsic and extrinsic perspectives. Experimental results demonstrate that the automatically built English SKB is even superior to HowNet, the most widely used SKB that takes decades to build manually. And both the English and French SKBs can bring obvious performance enhancement in multiple downstream tasks. All the code and data of this paper (except the copyrighted dictionaries) can be obtained at `https://github.com/thunlp/DictSKB`.

1 Introduction

A word is the smallest linguistic element that can be used on its own with a particular meaning, but not the smallest semantic unit (O'Grady et al., 1997). The meaning of a word can be divided into smaller components. In linguistics, a *sememe* is defined as the minimum semantic unit of human languages (Bloomfield, 1926). Some linguists believe that meanings of all words can be expressed by a limited set of predefined sememes (Goddard and Wierzbicka, 1994). For example, the basic meaning of "boy" can be expressed by the compositions of `human`, `male` and `child`, while the meaning of "girl" can be expressed by `human`, `female` and `child`, where `human`, `male`, `female` and `child` are predefined sememes. It is even deemed that this sememe-based semantic system as well as the sememe set is universal among different languages, in which case sememes are also named *universal semantic primitives* (Wierzbicka, 1996).

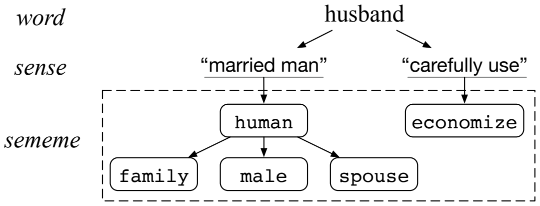

Figure 1: Sememe annotations of the word "husband" in HowNet.

Sememes are implicit in words and cannot be directly used in natural language processing (NLP). Dong and Dong (2006) make a seminal contribution and put the sememe-based semantic system into practice. They define a set of about $2,000$ sememes and use them to annotate senses of over $100,000$ Chinese and English words, whereupon a sememe knowledge base (SKB) named HowNet is built up. Figure 1 illustrates an example of how words are annotated with sememes in HowNet.

As a sememe-based lexical knowledge base, HowNet is very different from most other lexical knowledge bases like WordNet (Miller, 1998), which extensionally explain meanings of words by word-level relations, e.g., hyponym and meronym. In contrast, HowNet provides intensional definitions using infra-word sememes. This distinctness brings special advantages to HowNet. First, the sememe-to-word semantic compositionality endows HowNet with particular suitability for integration into neural networks (Qi et al., 2019a; Li et al., 2019). The sememes of a word can be re-

---

[*] Work done during internship at Tsinghua University
[†] Corresponding author. Email: sms@tsinghua.edu.cn

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4673–4686
August 1–6, 2021. ©2021 Association for Computational Linguistics

garded as semantic labels and easily incorporated into the neural processing unit of the word, e.g., a cell of RNN (Qin et al., 2020). Second, the nature that a limited set of sememes are used to express meanings of unlimited words makes HowNet very useful in low-data regimes, e.g., improving embeddings of rare words (Sun and Chen, 2016; Niu et al., 2017), where sememes serve as a bridge between high-frequency and rare words. Thus far a large body of research has demonstrated the usefulness of HowNet in various NLP tasks (Qi et al., 2020b).

HowNet is distinctive and valuable, but it covers only two languages. Most languages have no SKBs like HowNet, which deprives NLP in those languages of benefit from sememes. An obvious solution to this problem is to build an SKB for each language manually, but it is not realistic because it would be unimaginably time-consuming and labor-intensive.[1] To address the challenge, previous studies try to extend HowNet to other languages by automatically predicting sememes for words in those languages (Qi et al., 2018, 2020a). However, existing methods are not effective enough, and manual effort is necessary to ensure the correctness of their sememe prediction results.

In this paper, we explore a fully automatic way to build an SKB for a language via dictionaries with a *controlled defining vocabulary*. A dictionary, especially a learner's dictionary, usually uses a well-chosen list of words to construct all its definitions, and the word list is named controlled defining vocabulary (CDV) (Atkins and Rundell, 2008). A CDV is composed of high-frequency words that not only cover the vast majority of texts but also form a semantic basis so as to express meanings of all other words (Nation and Waring, 2004). To some extent, words in a CDV can fit the definition of sememes (Wierzbicka, 1996). This discovery inspires us to utilize a dictionary to build an SKB by regarding the words in its CDV as sememes.

We design a quite simple and quick process for automatically building SKBs based on dictionaries. First, a sememe set is constructed based on the CDV of a dictionary by removing words that are not suitable as sememes (e.g., stop words), then sememes of words are extracted from corresponding definitions, and finally an SKB composed of words annotated with sememes is established. We adopt the process to build an English SKB and a French SKB and conduct both intrinsic and extrinsic evaluations. In intrinsic evaluation, we find that both the SKBs possess high sememe annotation consistency, and the English SKB performs even better than the English part of HowNet. In extrinsic evaluation, we apply the dictionary-based SKBs to several sememe-incorporated models originally designed for HowNet and carry out experiments on different downstream tasks. Experimental results show that incorporating the SKBs can bring consistent performance enhancement, and the English SKB-incorporated models even outperform HowNet-incorporated models. These results demonstrate the usefulness and effectiveness of the dictionary-based SKBs as well as the feasibility of building SKBs via dictionaries.

To conclude, our contributions are threefold: (1) discovering the similarity between sememes and words in the controlled defining vocabulary, which is the first time as far as we know; (2) proposing to automatically build an SKB via a dictionary, which can be achieved by a simple and quick process; and (3) building an English SKB and a French SKB based on dictionaries and demonstrating their effectiveness in multiple downstream tasks.

2 Related Work

2.1 HowNet and Its Applications

Since HowNet was published (Dong and Dong, 2003), it has attracted considerable attention of NLP researchers. In the era of statistical NLP, it plays a very important role in various NLP tasks including word similarity computation (Liu and Li, 2002), word sense disambiguation (Zhang et al., 2005; Duan et al., 2007), text classification (Sun et al., 2007), sentiment analysis (Zhu et al., 2006; Fu et al., 2013), etc.

When deep learning becomes the mainstream approach of NLP, the usefulness of HowNet is also proved in diverse tasks including word representation learning (Sun and Chen, 2016; Niu et al., 2017), language modeling (Gu et al., 2018), semantic composition (Qi et al., 2019a), sequence modeling (Qin et al., 2020), reverse dictionary (Zhang et al., 2020), word sense disambiguation (Hou et al., 2020), textual adversarial attacking (Zang et al., 2020) and backdoor attacking (Qi et al., 2021).

2.2 Expansion of HowNet

To tackle the challenge that many new words are not contained in HowNet, Xie et al. (2017) present

---

[1] The construction of HowNet takes several linguistic experts more than two decades.

the task of lexical sememe prediction, aiming to expand HowNet by automatically predicting sememes for new words. They propose two simple and effective sememe prediction methods inspired by recommendation system. Jin et al. (2018) further incorporate Chinese characters into sememe prediction and achieve higher performance when predicting sememes for Chinese words.

Another research line focuses on extending HowNet to other languages. Qi et al. (2018) propose the task of cross-lingual lexical sememe prediction, aiming to extend HowNet to a new language by predicting sememes for words in that language. Qi et al. (2020a) present a more efficient way to extend HowNet to other languages, i.e., building a multilingual SKB based on BabelNet (Navigli and Ponzetto, 2012). BabelNet is composed of multilingual synsets that contain synonyms in many languages. Words (synonyms) in a synset have the same meaning and hence the same sememes. Therefore, they propose to predict sememes for the multilingual synsets, by which all the words in synsets will obtain predicted sememes at the same time.

Limited by the accuracy of sememe prediction, manual examination is necessary if we want to put the above HowNet expansion methods into service. In contrast, our proposed dictionary-based SKB construction method is completely automatic and can build a usable SKB very quickly.

2.3 Applications of Dictionaries

Dictionaries are handy and high-quality resources for NLP research. A main application of dictionaries is word sense disambiguation, where dictionaries play the role of sense inventory, and their definitions provide abundant semantic information for each sense (Lesk, 1986; Luo et al., 2018a,b; Kumar et al., 2019; Huang et al., 2019; Du et al., 2019; Blevins and Zettlemoyer, 2020). The semantic information in dictionary definitions is also used to improve word representation learning (Tissier et al., 2017; Bahdanau et al., 2017; Bosc and Vincent, 2018; Scheepers et al., 2018). In addition, dictionary definitions are also utilized in reverse dictionary (Hill et al., 2016; Pilehvar, 2019; Zhang et al., 2020), knowledge graph embedding (Zhong et al., 2015; Xie et al., 2016), reading comprehension (Long et al., 2017), etc. As far as we know, this paper is the first work to utilize dictionaries to build SKBs.

3 Building an SKB via a Dictionary

In this section, we detail the process of building an SKB via a dictionary. We take the building process of an English SKB based on *Longman Dictionary of Contemporary English* (LDOCE) (Bullon, 2006), a highly influential English learner's dictionary, as an example, and the building method can be readily generalized to other languages or dictionaries.[2]

3.1 Constructing the Sememe Set

We first construct the sememe set from the CDV of LDOCE by removing some words. LDOCE uses an approximately $2,000$-word CDV named Longman Defining Vocabulary (Bullock, 2011), which is developed from General Service List (West, 1953), a famous high-frequency word list for English learners. The CDV includes some stop words such as "that" and "to", which bear insignificant meanings and are not suitable as sememes. Thus, we filter them out according to the stop word list of NLTK (Loper and Bird, 2002). But negators like "not" are retained because they are critical to the meanings of words. In addition, according to previous work (Xie et al., 2017; Qin et al., 2020), sememes that are annotated to too many or too few words are usually uninformative and ineffective to downstream applications. Therefore, we count the frequencies of words in the CDV occurring in all definitions and empirically remove the most frequent 1% and the infrequent 10%. So far we have obtained the sememe set that is composed of $2,046$ sememes.

3.2 Extracting Sememes from Definitions

Next, we extract sememes for each sense of each word from its definition. We take the word "beautiful" as a running example to illustrate the process of sememe extraction, as shown in Figure 2.

"beautiful" has two senses in LDOCE, and both of them are adjective. For each sense, we first use NLTK to normalize its definition including tokenization and lemmatization. For example, the definition of its first sense is normalized into a sequence of tokens: {"someone", "or", "something", "that", "be", "beautiful", "be", "extremely", "attractive", "to", "look", "at"}. Then we remove the tokens that are not in the sememe set. In the above example, "someone", "something", "or", "that", "be", "to" and "at" are removed. So far we obtain the sememes

---

[2] The building process and evaluation results of the French SKB are given in Appendix A and B.

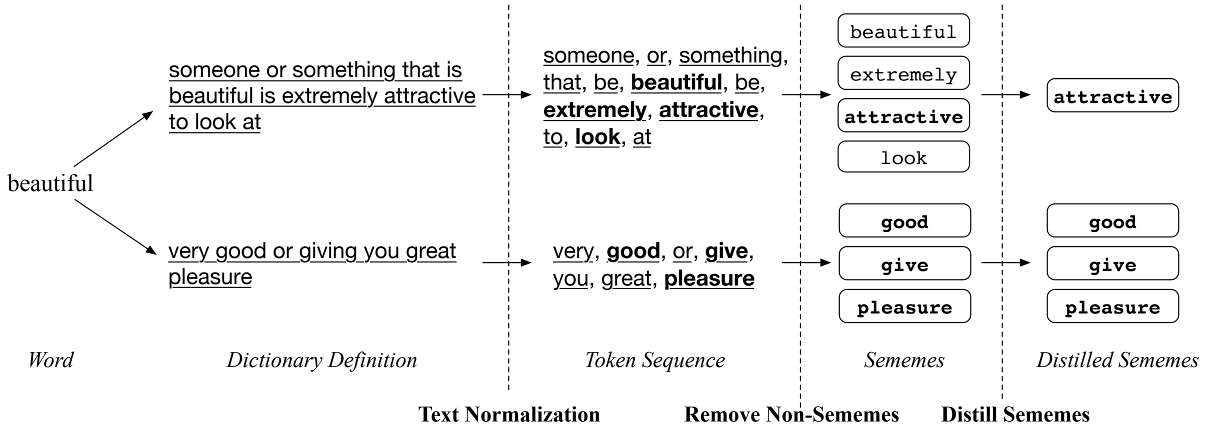

Figure 2: The process of extracting sememes from dictionary definitions for the word "beautiful".

SKB	#Word/Phrase	#Sense	#Sememe	#AvgSem
HowNet	50,879	111,519	2,187	2.26
EDSKB	70,218	105,160	2,046	6.03
EDSKB*	70,218	105,160	1,682[3]	2.04

Table 1: Statistics of EDSKB, its distilled version EDSKB* and the English part of HowNet.[4] #AvgSem denotes the average sememe number per sense.

of the first sense of "beautiful": {`beautiful`, `extremely`, `attractive`, `look`}. In a similar way, we can obtain the sememes of its second sense: {`good`, `give`, `pleasure`}.

By repeating this process on all the words of LDOCE, we obtain an English SKB that we name EDSKB. Its statistics are shown in Table 1.

3.3 Distilling Sememes of Senses

By comparison with HowNet, we find that the sememe set of EDSKB is smaller (EDSKB $2,046$ vs. HowNet $2,187$), but its average sememe number per sense is much larger (EDSKB 6.03 vs. HowNet 2.26), which means the sememes of EDSKB are utilized more fully and effectively. Moreover, annotating a sense with more sememes can explain the sense more accurately and finely. Nevertheless, it would also increase the distinguishability between different senses/words, which has an adverse effect on some downstream tasks. For example, word-level textual adversarial attacking conducts word substitution to generate adversarial examples, in which fewer substitute words usually lead to lower attack success rates (Wang et al., 2019; Zang et al., 2020). In a sememe-based word substitution strategy (Zang et al., 2020), more sememes per sense mean fewer substitute words that share the same sememes can be found, which will decrease the final adversarial attack success rate. To address this problem, we intend to craft an extra distilled version of EDSKB by distilling its sememes of senses.

To this end, we need to determine the importance of each sememe of a sense, and remove the relatively unimportant sememes. Here we resort to dependency parsing (Kubler et al., 2009). Dependency parsing is used to analyze syntactic structures of a sentence by identifying the word that another word is "dependent" on, e.g., the adjective is dependent on the noun in an adjective-noun phrase. We believe that the words with more dependents are more important in a definition. Hence, we define the importance score of a sememe for a sense as the number of the dependents of its original word in the definition.

Then, we empirically remove the sememes whose importance scores are below the highest importance score minus t for the senses having m or more sememes. Here t and m are two hyperparameters and are tuned to 1 and 4 respectively, based on the performance on the validation sets of downstream tasks, especially adversarial attacking. For example, for the first sense of "beautiful", by using AllenNLP (Gardner et al., 2018) to conduct dependency parsing on its definition, we obtain the numbers of dependents of all the words in the definition. Correspondingly, we get the importance scores of the four sememes {`beautiful`, `extremely`, `attractive`, `look`}, which are 2, 0, 6 and 0 respectively. The highest impor-

---

[3] Notice that its sememe set shrinks because some sememes are not annotated to any senses anymore.

[4] The data of HowNet are obtained from OpenHowNet (Qi et al., 2019b).

tance score is 6 and thus the sememes whose importance scores are less than 5 are removed, i.e., `beautiful`, `extremely` and `look`. Finally, the remaining sememe of the first sense of "beautiful" is {`attractive`}. As for the second sense, it has only 3 sememes and all of them are retained (the sememe number threshold for sememe reduction is $m = 4$). Therefore, its final sememes after reduction are still {`good`, `give`, `pleasure`}.

By repeating the above process on all the words, we obtain a distilled version of EDSKB (signified by EDSKB*), whose average sememe number per sense is comparable with HowNet (2.04 vs. 2.26). Its detailed statistics are also shown in Table 1.

Later experiments (on both English and French) show that the full version outperforms the distilled version in some downstream tasks while not in others. In practice, we can build both full and distilled versions and conduct experiments to see which one is better in a specific task. It is affordable to build and evaluate two versions.

4 Intrinsic Evaluation

In this section, we conduct an intrinsic evaluation to assess the sememe annotation consistency of EDSKB. Sememe annotation consistency measures how compatible the sememe annotations for different words/senses are, e.g., whether two synonyms are annotated with exactly the same sememes. The sememe annotation consistency of an SKB not only reflects its intrinsic quality but also has impact on its effectiveness in downstream tasks.

We evaluate both full and distilled versions of EDSKB, and the English part of HowNet for comparison. We adopt a sememe consistency assessment method named CCSA (Liu et al., 2020), which is designed for HowNet originally but can be used for any SKB. This method is motivated by the idea that semantically close senses should have similar sememes, which conforms to the linguistic definition of sememes. It actually implements a sememe prediction process that predicts sememes for a small proportion of senses according to the sememe annotations of the other senses. The sememe prediction method it adopts is based on collaborative filtering (Xie et al., 2017), and tends to predict the sememes that are annotated to semantically close senses to the target sense. Therefore, higher sememe prediction performance means the semantically close senses are annotated with more similar sememes, and the sememe annotations are more consistent. Correspondingly, the sememe annotation consistency of an SKB is measured by two sememe prediction evaluation metrics, namely mean average precision (MAP) and F1 score.

SKB	MAP	F1
HowNet	0.93	0.91
EDSKB	0.88	0.86
EDSKB*	**0.95**	0.91

Table 2: Sememe annotation consistency results. The **boldfaced** results show statistically significant improvement over the best results from baselines with $p<0.1$ given by t-test, and the underlined results represent having no significant difference.[5]

Table 2 lists the evaluation results of sememe annotation consistency. We can see that the distilled version of EDSKB has overall higher consistency than HowNet, and the full version of EDSKB yields lower consistency results. It is not strange because CCSA is based on sememe prediction and according to previous work (Qi et al., 2020a), senses with more sememes usually have lower prediction performance. Since the full version of EDSKB has much more sememes per sense than HowNet, it is actually not fair to compare their consistency using CCSA. The distilled version of EDSKB has a similar average sememe number as HowNet, and its superior results can demonstrate the great consistency of the dictionary-based SKB.

5 Extrinsic Evaluation

In this section, we conduct extrinsic evaluations to assess the effectiveness of EDSKB in downstream tasks. We pick three representative sememe-incorporated neural network models that are used for language modeling, sequence modeling and textual adversarial attacking tasks, respectively. All of them are originally designed for HowNet and have demonstrated efficacy on their respective tasks.

5.1 Language Modeling

In this subsection, we try to apply EDSKB to the task of language modeling. We use SDLM (Gu et al., 2018), a sememe incorporation method for language models, to incorporate EDSKB into two representative language models based on recurrent neural networks (RNNs).

Language modeling is aimed at predicting the next word given previous context (Bengio et al.,

---

[5] The same is true for the following tables.

2003). Language models based on RNNs, especially LSTMs (Hochreiter and Schmidhuber, 1997), are very popular, which use RNNs to encode the previous text into a vector and then feed the vector to a classifier to predict the next word. SDLM reforms the prediction process. Instead of directly predicting the next word, SDLM predicts sememes first, then senses and finally the next word.

Base Models We use two representative LSTM-based language models as the base models into which EDSKB is incorporated by SDLM.

- **Tied LSTM** (Zaremba et al., 2014), which enhances a vanilla two-layer LSTM language model by introducing dropout and weight tying. We use its large version whose word embedding and hidden vector sizes are $1,500$.
- **AWD-LSTM** (Merity et al., 2018), which adopts several regularization and optimization strategies including DropConnect (Wan et al., 2013) and non-monotonically triggered average stochastic gradient descent, and is a very strong baseline language model. Its hidden vector size is $1,150$ and word embedding size is 400.

Baseline Methods In addition to the two original base models, we additionally use SDLM to incorporate HowNet into the base models as baseline methods.

Datasets We choose two benchmark language modeling datasets for evaluation, namely Penn Treebank (PTB) (Marcus et al., 1993) and WikiText-2 (Merity et al., 2017). PTB consists of news stories from the Wall Street Journal. The numbers of tokens in its training, validation and test sets are $887,521$, $70,390$ and $78,669$, respectively. WikiText-2 is made up of Wikipedia articles, and it has $2,088,628$, $217,646$ and $245,569$ tokens in its training, validation and test sets.

Experimental Settings In our experiments, we use the official implementation of SDLM and its default hyper-parameters as well as training methods. The evaluation metric is perplexity. The lower perplexity a language model computes, the better the language model is.

Experimental Results Table 3 lists the perplexity results on the two datasets. We observe that the models incorporated with EDSKB, especially the full version, consistently outperform the two base models without sememe incorporation and even the HowNet-incorporated models. These results demonstrate the effectiveness of EDSKB in language modeling.

Dataset	PTB		WikiText-2	
Model	Valid	Test	Valid	Test
Tied LSTM	63.92	63.98	53.10	51.41
+HowNet	58.93	58.95	48.83	47.28
+EDSKB	**58.81**	**58.82**	**43.38**	**42.15**
+EDSKB*	60.17	60.15	45.18	42.59
AWD-LSTM	58.89	59.24	45.29	44.13
+HowNet	58.95	58.92	46.84	45.29
+EDSKB	**56.94**	**57.13**	**42.44**	**41.25**
+EDSKB*	58.63	58.59	43.85	43.95

Table 3: Perplexity results of different language models on the validation and test sets of PTB and WikiTex-2.

5.2 Sequence Modeling

In this subsection, we incorporate EDSKB into RNNs to improve their sequence modeling ability by SememeCell (Qin et al., 2020), a sememe incorporation method for enhancing RNNs.

SememeCell uses a special RNN cell to encode sememes of a word into a latent vector and transmits it to the corresponding RNN cell of the word, aiming to inject the semantic information of sememes into RNNs. It has demonstrated its effectiveness in improving the sequence modeling ability of RNNs in multiple downstream tasks, including natural language inference, sentiment analysis and paraphrase detection (Qin et al., 2020).

Base Models Following Qin et al. (2020), we choose two most representative RNNs, namely LSTM, GRU (Cho et al., 2014), and their bidirectional versions (BiLSTM and BiGRU) as the base models, into which sememes are incorporated by SememeCell.

Baseline Methods In addition to the vanilla and HowNet-incorporated RNNs, we also design another two baseline methods.

- **+Pseudo**. RNNs incorporated with either EDSKB or HowNet have a little more parameters than vanilla RNNs. To eliminate the possible effect brought by more parameters, we build a pseudo-SKB named Pseudo. Specifically, for each sense in EDSKB, we substitute its sememes with the same number of meaningless labels. The labels are randomly sampled from a label set with the same size as the sememe set of ED-

SKB. We use SememeCell to incorporate this pseudo-SKB into the two base models as baselines, which have exactly the same numbers of parameters as EDSKB-incorporated models.

- **+Definition**. EDSKB is obtained from dictionary definitions by the transformation from a sequence of words (definition) into several discrete semantic labels (sememes). We intend to compare the EDSKB-incorporated models and models incorporated with the complete dictionary definitions. Since SememeCell only takes a vector (i.e., the sum of sememe embeddings) as input, we can leverage it to incorporate definitions into RNNs by encoding definitions into vectors with a sentence encoder. Specifically, we choose the powerful pre-trained language model BERT (Devlin et al., 2019) as the sentence encoder and use the hidden vector of the `[CLS]` token as the definition embedding. The definition-incorporated RNN models are also baselines.

Downstream Tasks and Datasets RNNs are basic sequence encoders and can be used in many downstream NLP tasks. Following Qin et al. (2020), we choose two representative tasks to evaluate the sentence modeling ability of EDSKB-incorporated RNNs.

- Natural language inference (NLI), which is aimed at determining whether a natural language hypothesis can be inferred from a premise. It is a typical sentence pair classification task. We use the SNLI dataset (Bowman et al., 2015) for evaluation. SNLI contains about $570,000$ English premise-hypothesis pairs, and each pair is manually labeled one of three relation labels, namely "entailment", "contradiction" and "neutral".
- Sentiment analysis, which aims to recognize the sentiment orientation of a sentence and is a typical single sentence classification task. Following Qin et al. (2020), we use the CR dataset (Hu and Liu, 2004) for evaluation. It contains about $8,000$ product reviews and each review is labeled with "positive" or "negative".

Experimental Settings We use the official implementation of SememeCell (Qin et al., 2020) and the default hyper-parameter settings and training methods, where the embedding size (for both word and sememe embeddings) is 300 and hidden size is $2,048$. In the baseline method +Definition, to keep the definition vector size comparable with sememe embedding size, we choose the medium version of BERT, which has 512-dimensional hidden vectors and 8 layers.[6] As for evaluation metrics, we use accuracy for both NLI and sentiment analysis.

Dataset	Method	LSTM	GRU	BiLSTM	BiGRU
SNLI	vanilla	80.66	82.00	81.30	81.61
	+Pseudo	81.28	80.90	81.91	82.07
	+HowNet	81.87	82.90	82.55	83.15
	+Definition	81.62	82.80	81.10	83.22
	+EDSKB	**82.32**	**83.18**	82.54	**83.55**
	+EDSKB*	81.78	82.10	82.11	82.35
CR	vanilla	74.17	76.37	77.62	78.76
	+Pseudo	73.96	75.44	76.16	78.20
	+HowNet	76.47	78.57	77.66	76.25
	+Definition	76.29	78.20	77.19	77.77
	+EDSKB	**77.51**	**79.68**	**78.95**	**78.88**
	+EDSKB*	75.09	77.54	76.90	78.18

Table 4: Accuracy results of different models on the test sets of SNLI and CR.

Experimental Results Table 4 shows the evaluation results on the test sets of SNLI and CR. We can see that RNNs incorporated with dictionary-based SKB, especially the full version (+EDSKB), yield overall better results than vanilla RNNs, which proves that the dictionary-based SKB can improve the sequence modeling ability of RNNs. Furthermore, the +EDSKB models outperform +Pseudo models that have the same number of parameters, +Definition models that have the same semantic information source, and +HowNet models that incorporate another SKB. These results demonstrate the superiority of discrete sememes over definitions, and the advantage of dictionary-based SKB over HowNet in enhancing RNNs. +Pseudo performs slightly better than vanilla in some cases, which is probably because +Pseudo utilizes the random meaningless labels as noises. The addition of noise has been proven a regularization method for mitigating overfitting and improving performance in neural networks (Bishop, 1995).

5.3 Textual Adversarial Attacking

In this subsection, we investigate the effectiveness of EDSKB in textual adversarial attacking.

Adversarial attacking has attracted considerable research attention recently, mainly because it can reveal the vulnerability of neural network models and help improve their robustness and interpretability (Xu et al., 2020). Adversarial attacks use *ad-*

---

[6] `https://github.com/google-research/bert`

versarial examples (Szegedy et al., 2014), which are maliciously crafted by perturbing the original model input, to fool the victim model. In textual adversarial attacking, word-level attack methods, mainly based on word substitution, are a kind of popular attack method and have demonstrated overall better attack performance (Wang et al., 2019).

Zang et al. (2020) decompose the process of word-level attacks into two steps: (1) determining the substitute set for each word in the original input via a word substitution strategy, e.g., synonym-based and word embedding-based substitution strategies; and (2) searching the combinations of each original word's substitutes for adversarial examples that can successfully fool the victim model.

They also propose an adversarial attack approach that employs a sememe-based word substitution strategy and achieves strong attack performance. The sememe-based word substitution strategy essentially regards a word w_1 as the substitute of another word w_2, if one sense of w_1 has the same sememes as one sense of w_2, according to an SKB. We use this approach to conduct textual adversarial attacks and measure the attack performance.

Baseline Methods In addition to the original sememe-based attack approach that uses HowNet as the SKB, we choose some other baseline methods for comparison. Notice that all these baseline methods use the same approach to search for adversarial examples (the aforementioned step 2) and differ in word substitution strategies (step 1) only.

- **+Synonym**, the attack method that uses synonym-based word substitution strategy. Following previous work (Ren et al., 2019), we use WordNet as the thesaurus and the words in a synset can be regarded as substitutes of each other.
- **+Definition**, the attack method that uses a definition-based word substitution strategy. Inspired by word embedding-based word substitution, we encode the definition of each sense of words into a vector and define the similarity between two words as the cosine similarity between their closest definition vectors. Then, a certain number of words that are most similar to the target word are regarded as its substitutes. Specifically, we still use the medium-size BERT to encode definitions into 512-dimensional vectors. And the number of substitutes of each word is the same as that in the sememe-based substitution strategy.

Victim	Attack Method	ASR	%M	%IGE	PPL
BiLSTM	+Synonym	79.0	10.45	7.59	593.09
	+Definition	90.0	8.76	7.56	518.71
	+HowNet	93.6	9.02	2.57	**468.92**
	+EDSKB	26.5	8.27	3.77	538.46
	+EDSKB*	**94.0**	8.29	**1.27**	507.34
BERT	+Synonym	81.3	9.22	8.00	576.82
	+Definition	86.3	8.03	7.18	538.00
	+HowNet	91.2	8.25	2.08	503.06
	+EDSKB	29.7	8.10	3.36	**485.00**
	+EDSKB*	**93.3**	**7.66**	**1.07**	544.51

Table 5: Adversarial attack results of different word substitution strategies. ASR is short for attack success rate. %M, %IGE and PPL denote word modification rate, increase rate of grammatical errors and perplexity, respectively.

In this task, the +Pseudo baseline in the previous section cannot work because it would regard random words as substitutes of the target word.

Victim Models and Datasets Following Zang et al. (2020), we choose BiLSTM and BERT, specifically $\text{BERT}_{\text{BASE}}$ as the victim models we intend to attack. The evaluation task is sentiment analysis and the evaluation dataset is SST-2 (Socher et al., 2013). SST-2 comprises about $10,000$ sentences in movie reviews and each sentence is labeled with "positive" or "negative". The accuracy results of BiLSTM and BERT on the test set of SST-2 are 83.75 and 90.28.

Experimental Settings We use the official implementation of the sememe-based attack approach (Zang et al., 2020) and the default hyper-parameter settings.

Evaluation Metrics Following Zang et al. (2020), we use attack success rate to measure the effectiveness of an attack method and three metrics to assess the quality of its adversarial examples. The three metrics are (1) word modification rate, the percentage of words in the original input that are perturbed; (2) increase rate of grammatical errors in adversarial examples compared with original input, where LanguageTool grammar checker is used; and (3) perplexity given by GPT-2 (Radford et al., 2019) that is used to measure the fluency of adversarial examples. The lower the three metrics are, the better the quality of adversarial examples is.

Experimental Results According to Table 5, we find that the attack method based on EDSKB*

Word	SKB	Sememes
screenwriter	HowNet	`human`, `occupation`, `entertainment`, `compile`, `shows`
	EDSKB	`someone`, `write`, `play`, `film`, `television`
	EDSKB*	`write`, `play`, `film`, `television`
tweet	HowNet	`InstitutePlace`, `ProperName`, `produce`, `software`, `LookFor`, `document`, `information`, `internet`
	EDSKB	Sense 1: `bird`, `make`, `high`, `small`, `short`, `sound` Sense 2: `service`, `message`, `network`, `short`, `send`, `use`, `social`
	EDSKB*	Sense 1: `bird`, `sound` Sense 2: `message`, `send`, `use`, `network`

Table 6: Two cases of sememe annotations in HowNet, EDSKB and EDSKB*.

not only achieves the highest attack success rates but also generates adversarial examples with overall higher quality. These results show that the dictionary-based SKB EDSKB* can better capture the semantic relations between words and find appropriate substitutes for adversarial attacks. Attack success rates of the EDSKB-based method are extremely low. It is because EDSKB has too many sememes per sense, which causes the found substitutes to be very few (EDSKB 1.6, EDSKB* 12.6 and HowNet 15.3 on average), according to the sememe-based word substitution strategy that requires substitutes to have the same sememes.

6 Case Study on Sememe Annotations

In this section, we give two cases of sememe annotations in EDSKB and EDSKB* as well as HowNet in Table 6.

The first case is the word "screenwriter". In HowNet, this word has only one sense that is annotated by five sememes, as listed in the second row of Table 6. As for EDSKB and EDSKB*, according to *Longman Dictionary of Contemporary English* (LDOCE), this word also has only one sense whose definition is "someone who writes plays for film or television". EDSKB provides five sememes and one (`someone`) is filtered out in EDSKB*. By comparison, we can find that sememes in EDSKB and EDSKB* can represent the meaning of the word more specifically, e.g, `write` and `play`, while sememes in HowNet seem to express a more general meaning.

The second case is about the word "tweet". HowNet only annotates one sense for this word, i.e., "to send a message on Twitter". As for EDSKB and EDSKB*, since LDOCE contains the basic meaning of this word, i.e., "to make the short high sound of a small bird", the sememes including `bird` and `sound` are extracted to express this meaning. In addition, for the shared sense, sememes in EDSKB and EDSKB* are more succinct than those in HowNet, e.g., `message` in EDSKB/EDSKB* can better describe the core meaning of "tweet" than `document` and `information` in HowNet.

From the two cases, we can see the advantage of the dictionary-based SKBs over HowNet in terms of sememe annotations. We hope that the dictionary-based SKBs can be used to perfect HowNet by supplying more senses and annotating more suitable sememes.

7 Conclusion and Future Work

In this paper, we propose to utilize a dictionary to build an SKB for the first time, which can be implemented by a simple, quick and fully automatic process. We try utilizing existing dictionaries to build an English SKB and a French SKB, and demonstrate their effectiveness on multiple NLP tasks. Extensive experimental results prove the reliability and practicality of our idea about dictionary-based SKB construction.

It is worth mentioning that although EDSKB delivers better empirical results than HowNet, HowNet has its unique advantages including better interpretability and multilinguality. In the future, therefore, we will systematically compare the sememe annotations in EDSKB and HowNet and try to use EDSKB to improve and expand HowNet. Besides, the hierarchical structures of sememes in HowNet are neglected in this paper. We will also explore to extract sememes with hierarchy from dictionary definitions.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (Grant No. 2020AAA0106502 and No. 2020AAA0106501) and Beijing Academy of Artificial Intelligence (BAAI). We also thank all the anonymous reviewers for their valuable comments and suggestions.

Ethical Considerations

In this paper, we use two copyrighted dictionaries, namely Longman Dictionary of Contemporary English and Le Petit Robert French Dictionary. We extract data from the electronic versions of the two dictionaries we bought for the research purpose only. We will not release the dictionary data. In addition, the datasets we use in downstream tasks are all open and free (actually also widely used).

The task we tackle is sememe knowledge base construction, which is not a practical application and is only related to NLP research. Therefore, the datasets we build and the models we use would not be misused by common people.

In addition, since we do not use very large models, the required energy in this work is very limited. Finally, we use no demographic or identity characteristics.

References

BT Sue Atkins and Michael Rundell. 2008. *The Oxford guide to practical lexicography*. Oxford University Press.

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Chris M Bishop. 1995. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116.

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss-informed biencoders. In *Proceedings of ACL*.

Leonard Bloomfield. 1926. A set of postulates for the science of language. *Language*, 2(3):153–164.

Tom Bosc and Pascal Vincent. 2018. Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of EMNLP*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

David Bullock. 2011. Nsm+ ldoce: A non-circular dictionary of english. *International Journal of Lexicography*, 24(2):226–240.

Stephen Bullon. 2006. *Longman Dictionary of Contemporary English*. Pearson Education Limited.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Zhendong Dong and Qiang Dong. 2003. Hownet-a hybrid language and knowledge resource. In *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering*. IEEE.

Zhendong Dong and Qiang Dong. 2006. *HowNet and the Computation of Meaning (With CD-Rom)*. World Scientific.

Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. Using bert for word sense disambiguation. *arXiv preprint arXiv:1909.08358*.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Word sense disambiguation through sememe labeling. In *Proceedings of IJCAI*.

Xianghua Fu, Guo Liu, Yanyan Guo, and Zhiqiang Wang. 2013. Multi-aspect sentiment analysis for Chinese online social reviews based on topic modeling and HowNet lexicon. *Knowledge-Based Systems*, 37:186–195.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Cliff Goddard and Anna Wierzbicka. 1994. *Semantic and lexical universals: Theory and empirical findings*, volume 25. John Benjamins Publishing.

Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Language modeling with sparse product of sememe experts. In *Proceedings of EMNLP*.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Bairu Hou, Fanchao Qi, Yuan Zang, Xurui Zhang, Zhiyuan Liu, and Maosong Sun. 2020. Try to substitute: An unsupervised chinese word sense disambiguation method based on hownet. In *Proceedings of COLING*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuan-Jing Huang. 2019. Glossbert: Bert for word sense disambiguation with gloss knowledge. In *Proceedings of EMNLP-IJCNLP*.

Huiming Jin, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Incorporating chinese characters of words for lexical sememe prediction. In *Proceedings of ACL*.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A Smith. 2020. The multilingual amazon reviews corpus. In *Proceedings EMNLP*.

Sandra Kubler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.

Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. 2019. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of ACL*.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbe, Laurent Besacier, and Didier Schwab. 2020. Flaubert: Unsupervised language model pre-training for french. In *Proceedings of LREC*.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*.

Ziran Li, Ning Ding, Zhiyuan Liu, Haitao Zheng, and Ying Shen. 2019. Chinese relation extraction with multi-grained information and external linguistic knowledge. In *Proceedings of ACL*.

Qun Liu and Sujian Li. 2002. Word similarity computing based on HowNet. *International Journal of Computational Linguistics & Chinese Language Processing*, 7(2):59–76.

Yangguang Liu, Fanchao Qi, Zhiyuan Liu, and Maosong Sun. 2020. Research on consistency check of sememe annotations in hownet. *Journal of Chinese Information Processing*.

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of EMNLP*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of EMNLP*.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018b. Incorporating glosses into neural word sense disambiguation. In *Proceedings of ACL*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2).

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *Proceedings of ICLR*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of ICLR*.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Paul Nation and Robert Waring. 2004. Vocabulary size, text coverage and word lists.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of ACL*.

William O'Grady, Michael Dobrovolsky, and Francis Katamba. 1997. *Contemporary linguistics*.

Mohammad Taher Pilehvar. 2019. On the importance of distinguishing word meaning representations: A case study on reverse dictionary mapping. In *Proceedings of NAACL-HLT*.

Fanchao Qi, Liang Chang, Maosong Sun, Sicong Ouyang, and Zhiyuan Liu. 2020a. Towards building a multilingual sememe knowledge base: Predicting sememes for babelnet synsets. In *Proceedings of AAAI*.

Fanchao Qi, Junjie Huang, Chenghao Yang, Zhiyuan Liu, Xiao Chen, Qun Liu, and Maosong Sun. 2019a. Modeling semantic compositionality with sememe knowledge. In *Proceedings of ACL*.

Fanchao Qi, Yankai Lin, Maosong Sun, Hao Zhu, Ruobing Xie, and Zhiyuan Liu. 2018. Cross-lingual lexical sememe prediction. In *Proceedings of EMNLP*.

Fanchao Qi, Ruobing Xie, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2020b. Sememe knowledge computation: a review of recent advances in application and expansion of sememe knowledge bases. *Frontiers of Computer Science*.

Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Qiang Dong, Maosong Sun, and Zhendong Dong. 2019b. Openhownet: An open sememe-based lexical knowledge base. *arXiv preprint arXiv:1901.09957*.

Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. 2021. Turn the combination lock: Learnable textual backdoor attacks via word substitution. In *Proceedings of ACL-IJCNLP*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020c. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of ACL*.

Yujia Qin, Fanchao Qi, Sicong Ouyang, Zhiyuan Liu, Cheng Yang, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Improving sequence modeling ability of recurrent neural networks via sememes. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of ACL*.

Paul Robert, Alain Rey, and Josette Rey-Debove. 2015. *Dictionnaire Le Petit Robert 2016*. Dictionnaires Le Robert.

Thijs Scheepers, Evangelos Kanoulas, and Efstratios Gavves. 2018. Improving word embedding compositionality using lexicographic definitions. In *Proceedings of WWW*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Jingguang Sun, Dongfeng Cai, Dexin Lv, and Yanju Dong. 2007. HowNet based Chinese question automatic classification. *Journal of Chinese Information Processing*, 21(1):90–95.

Maosong Sun and Xinxiong Chen. 2016. Embedding for words and word senses based on human annotated knowledge base: Use HowNet as a case study. *Journal of Chinese information processing*, 30(6).

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of ICLR*.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec: Learning word embeddings using lexical dictionaries. In *Proceedings of EMNLP*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of ICML*.

Xiaosen Wang, Hao Jin, and Kun He. 2019. Natural language adversarial attacks and defenses in word level. *arXiv preprint arXiv:1909.06723*.

Michael Philip West. 1953. *A general service list of English words: with semantic frequencies and a supplementary word-list for the writing of popular science and technology*. Longmans, Green.

Anna Wierzbicka. 1996. *Semantics: Primes and universals: Primes and universals*. Oxford University Press, UK.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*.

Ruobing Xie, Xingchi Yuan, Zhiyuan Liu, and Maosong Sun. 2017. Lexical sememe prediction via word embeddings and matrix factorization. In *Proceedings of IJCAI*.

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of ACL*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Lei Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Multi-channel reverse dictionary model. In *Proceedings of AAAI*.

Yuntao Zhang, Ling Gong, and Yongcheng Wang. 2005. Chinese word sense disambiguation using HowNet. In *Proceedings of International Conference on Natural Computation*.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of EMNLP*.

Yan-Lan Zhu, Jin Min, Ya-qian Zhou, Xuan-jing Huang, and Li-De Wu. 2006. Semantic orientation computing based on HowNet. *Journal of Chinese information processing*, 20(1):14–20.

A Building Process of a French SKB

In this section, we describe the building process of a dictionary-based French SKB that we call FDSKB. We choose Le Petit Robert French Dictionary 2016 edition (Robert et al., 2015), a very popular French dictionary, as the base dictionary.

Constructing the Sememe Set We first construct a sememe set from the defining vocabulary of the dictionary. Similar to EDSKB, we remove the most frequent and infrequent words that appear in definitions as well as some stop words, and obtain a sememe set comprising $2,919$ sememes (defining words).

Extracting Sememes from Definitions We use Stanza (Qi et al., 2020c) to tokenize and lemmatize the definitions of all words in the dictionary and extract the sememes of each sense of each word according to the sememe set. So far, we have obtained the full version of FDSKB, whose statistics are shown in Table 7.

Distilling Sememes of Senses We adopt a similar way to EDSKB to distill the sememes of senses. Specifically, we use Stanza to conduct dependency parsing for every definition and obtain the importance score of each sememe. Then we empirically remove the unimportant sememes according to the experimental results of downstream tasks. In this way, we obtain the distilled version of FDSKB (FDSKB*), whose statistics are also in Table 7.

B Evaluation of FDSKB

In this section, similar to EDSKB, we conduct both intrinsic and extrinsic evaluations for FDSKB and FDSKB*. Notice that since HowNet covers only English and Chinese, there are no available HowNet-based baseline methods for French.

SKB	#Word/Phrase	#Sense	#Sememe	#AvgSem
FDSKB	55,836	113,722	2,919	4.32
FDSKB*	55,836	113,722	2,919	1.97

Table 7: Statistics of FDSKB and its distilled version FDSKB*. #AvgSem denotes the average sememe number per sense.

Dataset	French News		FR-Wikipedia	
Model	Valid	Test	Valid	Test
Tied LSTM	17.02	18.35	17.23	15.75
+FDSKB	15.72	16.85	17.14	15.60
+FDSKB*	15.70	16.89	**16.50**	**15.15**
AWD-LSTM	18.41	19.71	16.76	15.30
+FDSKB	15.41	16.47	**15.45**	15.72
+FDSKB*	**14.03**	**15.14**	15.90	**14.10**

Table 8: Perplexity results on the validation and test sets of French News and FR-Wikipedia. The **boldfaced** results show statistically significant improvement over the best results from baselines with p<0.1 given by t-test, and the underlined results represent having no significant difference.[7]

B.1 Intrinsic Evaluation

We still use CCSA (Liu et al., 2020) to measure the sememe annotation consistency. The MAP and F1 score for FDSKB are 83.47 and 80.51 respectively, and those for FDSKB* are 90.03 and 90.01 respectively. These results are comparable to those of EDSKB and can prove good sememe annotation consistency of FDSKB. Besides, similar to EDSKB, the distilled version FDSKB* delivers better sememe annotation consistency than FDSKB because it has fewer sememes per sense.

B.2 Extrinsic Evaluation

We conduct extrinsic evaluation for FDSKB and FDSKB* on three tasks including language modeling, natural language inference (NLI) and text classification.

Language Modeling

Similar to EDSKB, we use SDLM (Gu et al., 2018) to incorporate FDSKB into Tied LSTM (Zaremba et al., 2014) and AWD-LSTM (Merity et al., 2018). The experimental settings are the same as those in English experiments.

We choose two evaluation datasets: (1) **French News**[8], which comprises French news articles from

---

[8] `https://webhose.io/free-datasets/french-news-articles/`

Dataset	Method	LSTM	GRU	BiLSTM	BiGRU
XNLI	vanilla	61.14	60.88	61.56	61.36
	+Pseudo	60.96	61.46	61.10	61.12
	+Definition	61.64	61.10	61.80	61.86
	+FDSKB	62.38	61.54	**62.52**	61.44
	+FDSKB*	**62.52**	**61.74**	62.16	**62.06**
MARC	vanilla	79.98	79.16	80.35	80.64
	+Pseudo	80.35	79.38	79.16	80.53
	+Definition	81.10	79.39	80.65	80.80
	+FDSKB	81.39	80.58	80.65	**82.14**
	+FDSKB*	**81.43**	**81.10**	**80.98**	81.84

Table 9: Accuracy results of different models on the test sets of XNLI and MARC.

popular news sites. It has 2,131,774 / 358,972 / 370,059 tokens in its training / validation / test sets. (2) **FR-Wikipedia**[9], which is composed of French Wikipedia articles. The token numbers in its training / validation / test sets are 3,252,094 / 520,333 / 517,669.

The experimental results are given in Table 8. We observe that both FDSKB and FDSKB* bring decreases of perplexity, which demonstrates the effectiveness of the French dictionary-based SKB in language modeling and the practicality of our dictionary-based SKB building method. Notice that since the perplexity results of original Tied LSTM and AWD-LSTM on the two French datasets are quite good, the enhancement brought by FDSKB is comparatively less than that in English.

NLI and Text Classification

Similar to EDSKB, we use SememeCell (Qin et al., 2020) to incorporate FDSKB into RNNs and measure the improvement of sequence modeling ability on the tasks of NLI and text classification.

The base models are still LSTM, GRU, BiLSTM and BiGRU. And the baseline methods are also +Pseudo and +Definition. Here we use FlauBERT (Le et al., 2020), a French pre-trained language model, to encode definitions into 768-dimensional vectors. The other experimental settings are the same as English.

As for evaluation datasets, we use XNLI (Conneau et al., 2018) and MARC (Keung et al., 2020) respectively. XNLI is a cross-lingual NLI dataset in 15 languages. It is based on another NLI dataset MNLI (Williams et al., 2018) and constructs its training set by machine translation, which has 361,469 sentence pairs. It has 2,500 and 5,000 sentence pairs in the validation and test sets which are manually translated from English. MARC (Multilingual Amazon Reviews Corpus) is a large corpus of Amazon reviews in 6 languages. We use its French part for product category classification. It has 40,000 / 1,323 / 1,345 reviews in its training / validation / test sets.

Table 9 shows the accuracy results on the test sets of XNLI and MARC. The results are basically consistent with the experimental results in English datasets. The incorporation of the dictionary-based SKB can improve the performance of RNN models on the two different tasks, which reflects that the SKB has enhanced the sequence modeling ability of RNNs. Moreover, the results also demonstrate the usefulness and effectiveness of our dictionary-based SKB and its building method.

C Experiment Running Environment

For all the experiments, we use a server whose major configurations are as follows: (1) CPU: Intel(R) Xeon(R) E5-2680 v4 @ 2.40GHz, 56 cores; (2) RAM: 125GB; (3) GPU: 8 Nvidia RTX2080 GPUs, 12GB memory. The operation system is Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-108-generic x86_64). We use PyTorch[10] v1.5.0 and Python v3.6.9 as the programming framework for the experiments on neural network models.

---

[9] `http://redac.univ-tlse2.fr/corpora/wikipedia_en.html`

[10] `https://pytorch.org/`

Rule-Aware Reinforcement Learning for Knowledge Graph Reasoning

Zhongni Hou[1,2], **Xiaolong Jin**[1,2], **Zixuan Li**[1,2], **Long Bai**[1,2]
[1]School of Computer Science and Technology, University of Chinese Academy of Sciences;
[2]CAS Key Laboratory of Network Data Science and Technology, Institute of
Computing Technology, Chinese Academy of Sciences
`{houzhongni18z,jinxiaolong,lizixuan,bailong18b}@ict.ac.cn`

Abstract

Multi-hop reasoning is an effective and explainable approach to predicting missing facts in Knowledge Graphs (KGs). It usually adopts the Reinforcement Learning (RL) framework and searches over the KG to find an evidential path. However, due to the large exploration space, the RL-based model struggles with the serious sparse reward problem and needs to make a lot of trials. Moreover, its exploration can be biased towards spurious paths that coincidentally lead to correct answers. To solve both problems, we propose a simple but effective RL-based method called RARL (Rule-Aware RL). It injects high quality symbolic rules into the model's reasoning process and employs partially random beam search, which can not only increase the probability of paths getting rewards, but also alleviate the impact of spurious paths. Experimental results show that it outperforms existing multi-hop methods in terms of Hit@1 and MRR.

1 Introduction

Knowledge Graphs (KGs), which store facts as triples in the form of (*subject entity*, *relation*, *object entity*), benefit various NLP applications (Lan and Jiang, 2020; Wang et al., 2019b; He et al., 2017). However, existing KGs face with serious incompleteness despite of their large scales. Therefore, KG completion, which aims to reason missing facts based on existing triples, has been an important research area.

The past decade has witnessed the rise of embedding-based reasoning methods on KGs (Bordes et al., 2013; Yang et al., 2014; Balažević et al., 2019). However, due to their black-box nature, these methods cannot provide interpretations for a specific prediction (Ji et al., 2020; Sadeghian et al., 2019). Recently, there has been growing interest in using multi-hop reasoning to improve the interpretability (Gardner et al., 2013; Rocktäschel and Riedel, 2017). This approach usually adopts Reinforcement Learning (RL) to find a reasoning path (Xiong et al., 2017; Das et al., 2018; Hildebrandt et al., 2020). Starting from the query entity, the RL-based model sequentially selects an outgoing edge and transits to a new entity until it arrives at the target.

However, due to the complexity of the KG, the number of paths grows exponentially when the reasoning hop increases. Most of paths cannot arrive at correct answers, and cannot receive a none-zero reward, which is also called the "sparse reward problem" (Nair et al., 2018). Moreover, since golden paths are not available in the training process, the RL-based model may coincidentally reach the target via a meaningless path (i.e. spurious paths). Take the query (*Captain America, director, ?*) as an instance. Although the path (*Captain America, country, US, lives in*$^{-1}$, *Peter Farrelly*), can arrive at the target. It is semantically inconsistent with the query relation *director* and is an accidental success. One trouble is that the RL-based model relies heavily on rewards and reinforces the past actions receiving high rewards regardless of their path quality. In addition, in large scale KGs, there are more spurious paths than correct ones (Lin et al., 2018). It is more easier for the model to discover spurious ones first other than the true and meaningful ones. If the model finds spurious ones first, these spurious paths will lead to a biased exploration and induce negative influence to the reasoning process (Guu et al., 2017; Lin et al., 2018).

Lin et al. (2018) uses shaped rewards calculated by pre-trained embedding-based models and an action dropout mechanism to solve the above two challenges, respectively. However, its performance largely depends on the embedding-based model used. In addition, embedding-based model increases the opacity of the reasoning process. Motivated by this, we focus on the action selection

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4687–4692
August 1–6, 2021.

strategy and propose RARL (Rule Aware RL), a simple but effective model to solve the above two challenges. RARL introduces high quality rules as prior information about actions and explores K paths in one episode. It selects actions from three parts: actions matching rules, actions with high scores, and actions randomly sampled. The former two parts can increase the probability of reasoning paths arriving at targets. The later one allows the model to explore a more diverse path set and thus avoids the model adhering to the past actions receiving high rewards, which can naturally mitigate the impact of spurious paths.

We evaluate RARL on three benchmark datasets, and experimental results show the effectiveness of RARL when compared with existing multi-hop methods.

2 Preliminaries

Let $\mathcal{E}$ be the set of entities and $\mathcal{R}$ be the set of relations, a knowledge graph can be represented as $\mathcal{G} = \{(e_s, r, e_t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. In this paper, we focus on the standard link prediction task. Given a query of the form $(e_s, r_q, ?)$, the reasoning model is expected to predict the correct answer e_t after traversing over the graph.

2.1 The RL-based Knowledge Reasoning Framework

Following (Das et al., 2018), when given a query $(e_s, r_q, ?)$, the RL-based model can be viewed as an agent, which interacts with the KG environment and aims to find a reasoning path $p = (e_s, r_1, e_1, ...)$ to explicitly show how to conduct reasoning. The parameters of the policy defines a policy. At each time t, the agent selects an action a_t, i.e. an outgoing edge of the current position e_t, to expand the path using a policy. Here, we define $A_t = \{(r', e') | (e_t, r', e') \in \mathcal{G}\}$ as the possible actions at time t. The model first uses a Long Short-Term Memory network (LSTM) to encode the path history into a vector $\mathbf{h}_t$. Then, the policy network π_θ (a two-layer feed-forward network) calculates a distribution over all possible actions in A_t.

$$\pi_\theta(a_t|e_t) = \sigma(\mathbf{A}_t(W_2 ReLU(W_1[\mathbf{h}_t;\mathbf{e}_t;\mathbf{r_q}]))), \quad (1)$$

where $\mathbf{e}_t \in \mathbb{R}^d$ and $\mathbf{r_q} \in \mathbb{R}^d$ are embeddings of e_t and r_q, respectively. $\mathbf{A}_t \in \mathbb{R}^{|A_t| \times 2d}$ is the stack of all actions embeddings in A_t and σ denotes the softmax operator. After this, the next edge is selected via an ϵ-greedy action selection strategy.

A binary reward $R(p)$ is observed after the maximum time step T: $R(p) = 1$ if the path ends at the correct answer and 0 otherwise.

The objective of the model is to maximize the expected reward:

$$J(\theta) = \sum_{(e_s, r_q, e_t) \in \mathcal{G}} \sum_{\hat{z} \in P(e_s, r_q)} R(\hat{z}) \pi_\theta(\hat{z}|e_s, r_q), \quad (2)$$

where $P(e_s, r_q)$ is the set of all reasoning paths related to the given query $(e_s, r_q, ?)$. The optimization is then performed by using REINFORCE algorithm (Williams, 1992).

2.2 Beam Search

In the RL context, Beam Search (BS) (Sutskever et al., 2014) stores top-K scoring partially constructed paths at each time step, where K is known as the beam size. At each time t, BS extends itself via the following process. Let us denote the paths set held by BS at the end of time t as $B_t = \{p_{[1:t]}, ..., p_{[K:t]}\}$. For each path $p = (e_s, r_1, ..., e_t) \in B_t$, we first generate its candidate paths $cand(p)$,

$$\begin{aligned} cand(p) &= cond(e_s, r_1, ..., e_t) \\ &= \{(e_s, r_1, ..., e_t, r', e') | (e_t, r', e') \in \mathcal{G}\}. \end{aligned} \quad (3)$$

Each candidate path $p \in cand(p)$ is associated with a score $s(p)$ calculated by the policy network. Here, $s(p) = \pi_\theta((r', e')|e_t)$. Further, we take the union of these candidate paths $\mathcal{B}_t = \bigcup_{p \in B_t} cand(p)$. A new beam B_{t+1} is generated by picking the K top-most elements in $\mathcal{B}_t$.

3 The RARL Model

As illustrated in Figure 1, the RARL model consists of two parts: the KG environment and the agent. By interacting with the environment, the agent employs a beam search based action selection strategy and picks K actions to extend the beam in one episode. The action selection strategy selects actions from three parts: actions matching rules, actions with high scores, and actions randomly sampled. After the maximum time step, the agent will receive binary rewards.

3.1 Rule Based Action Selection

In a typical KG, when the path length increases, finding a non-zero reward is exponentially more difficult. Learning from such sparse rewards requires lots of effective exploration. However, in the beginning, due to the randomly initialized parameters,

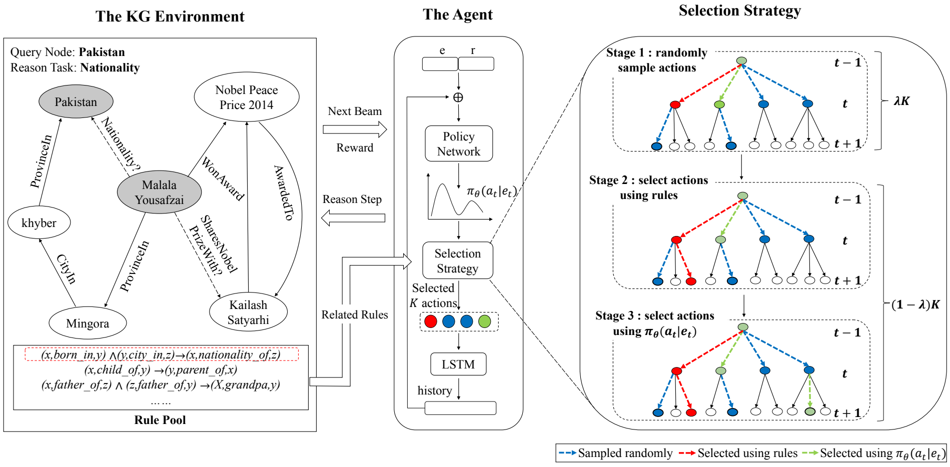

Figure 1: The reasoning process of RARL. Left: The KG environment. The dashed edges are part of queries and the solid edges are observed facts. Middle: The structure of the agent. By interacting with the environment, it picks K actions to extend the beam in one episode. Right: An illustration of the action selection strategy at time t. It first randomly samples λK actions and then selects the remaining $(1-\lambda)K$ actions using rules and probabilities.

the model chooses actions randomly and can hardly arrive at targets. This makes the sparse reward problem even worse (Xiong et al., 2017; Hare, 2019). Considering that rules precisely characterize a mapping from query relations to semantic composition paths (Zhang et al., 2019), RARL utilizes rules as prior information about actions to increase the probability of paths receiving rewards, which can also help to facilitate effective exploration.

The rules mined from KGs are in the form of $head \leftarrow body$, where the head is an atom $r(a, b)$ and the body is in the format of: $r(x_0, x_1) \wedge \ldots \wedge r(x_n, x_{n+1})$. Note that $r(x_i, x_j)$ is equivalent to the fact triple (x_i, r, x_j).

Given the query relation r_q, RARL first selects rules R_{r_q} whose heads are identical to r_q from the rule pool. At each time step t, it maintains a beam B_t of K paths. For each path $p = \{e_s, r_1, e_1, ..., r_t, e_t\} \in B_t$, RARL expands its candidate paths based on the outgoing edges of e_t. Next, for all candidate paths in $\mathcal{B}_t$, only those in which relation sequence can match related rules from left to right are selected. For instance, suppose R_{r_q} contains only one rule $r_q \leftarrow r_1 \wedge r_2$, given two candidate paths $(e_s, r_1, e_1, r_2, e_2)$, $(e_s, r_2, e_3, r_3, e_4)$, only the relation sequence in the former path can successfully matches the rule. As a result, the former one will be selected to generate B_{t+1}. If the number of candidate paths matching rules excesses the beam size, RARL selects top-K paths from these paths matching rules according to their scores calculated by the policy network. If not, remaining paths not matching rules are selected as a compliment. To make a balance between actions generated by free exploration and actions matching rules, RARL randomly masks some related rules to shrink the number of paths matching rules.

3.2 Partially Random Beam Search

To ease the impact of spurious paths, we try to prevent the RL-based model from the obsession of spurious paths and induce diversity during BS. Inspired by (Guu et al., 2017), we introduces partially randomness to standard BS, to fight against the impact of spurious paths.

Like regular beam search, at time t, RARL computes the set of all candidate paths $\mathcal{B}_t$ and sorts them by their scores computed by the policy network π_θ. Instead of selecting K highest-scoring candidate paths, RARL randomly chooses λK candidate paths from $\mathcal{B}_t$ and remaining paths are chosen according to their scores. In this way, low-scoring paths discarded in standard BS can also have the chance to be explored. Besides, the randomness can avoid the model sticking to the paths getting rewards. In the experiment, RARL selects paths with replacement when available actions smaller than λK.

3.3 The Overall Selection Strategy

RARL selects candidate actions by three stages: (1) Randomly sample λK actions based on the current position; (2) Select actions matching the

	Model	UMLS			WN18RR			FB15K-237		
		Hit@1	Hits@10	MRR	Hit@1	Hits@10	MRR	Hit@1	Hits@10	MRR
Multi Hop	NerualLP(Yang et al., 2017)	.643	.962	.778	.376	**.657**	.463	-	-	-
	MINERVA(Das et al., 2018)	.728	.968	.825	.413	.513	.406	.405	.583	.468
	AnyBURL(Wang et al., 2019a)	.657	.966	-	.431	.526	-	.220	.335	-
	Coper-MINERVA(Stoica et al., 2020)	.778	**.974**	.854	.426	.510	.465	.484	.630	.536
	RARL (ours)	**.803**	.970	**.866**	**.442**	.533	**.469**	**.516**	**.634**	**.557**
Embedding	DistMult(Yang et al., 2014)	.821	.967	.868	.431	.524	.462	.477	.642	.535
	ComplEx(Trouillon et al., 2016)	.890	.992	.934	.418	.480	.437	.496	.687	.563
	ConvE (Dettmers et al., 2017)	.932	.994	.957	.403	.540	.449	.480	.663	.544
	TuckER (Balažević et al., 2019)	.822	.997	.907	.443	.526	.470	-	-	-

Table 1: Link prediction results on UMLS, WN18RR, and FB15K-237. Best scores among the multi-hop methods and embedding-based methods are bold and underlined, respectively.

related rules based on the history; (3) Select actions in descending order of scores. If the number of actions matching rules from the second stage excesses $(1-\lambda)K$, then RARL selects the top $(1-\lambda)K$ actions according to their scores. If not, it continue to select actions via the third stage. The total size of actions selected from the last two parts are $(1-\lambda)K$.

4 Experiments

4.1 Experimental Setup

Datasets and Rules We adopt three datasets to evaluate the performance of RARL for link prediction: UMLS (Kok and Domingos, 2007), WN18RR (Dettmers et al., 2017), and FB15K-237 (Toutanova et al., 2015). For FB15k-237, 20 relations in the film field are selected. Following (Niu et al., 2020), We use AIME+ (Galárraga et al., 2015) to automatically extract rules, and we limit the maximum length of rules to 2.

Table 2 lists the statistics of rules with various confidence thresholds mined from these three datasets.

Model	Various Confidence Thresholds				
	0.5	0.6	0.7	0.8	0.9
UMLS	2,154	1,678	1,159	561	170
WN18RR	5	4	3	3	3
FB15K-237	2,044	1,621	1,255	912	565

Table 2: Statistics of rules mined on the three datasets.

Hyperparameters We set the dimensions of entity and relation embeddings within (50, 200). A three-layer LSTM is used as the path encoder and its hidden dimension is set in (100, 200). The λ is set as 0.9, 0.4 and 0.7 for UMLS, WN18RR and FB15K-237, respectively, according to the average degree of nodes and the average number of relation rules on each dataset.

4.2 Link Prediction Results

Table 1 summarizes the experimental results of our proposed approach and the baselines. As shown in Table 1, RARL achieves competitive results over multi-hop reasoning methods. On FB15K-237, RARL outperforms all baselines in terms of Hit@1, Hits@10, and MRR. On WN18RR and UMLS, the RARL achieves the best results in terms of Hit@1 and MRR. The Hit@1 results emphasize the superiority of our approach in high-precision link prediction and confirm the effectiveness of high quality rules. We also notice that the emebedding-based methods perform better on UMLS and FB15K-237 compared with multi-hop reasoning methods. One reason is that the multi-hop reasoning methods are more sensitive to the sparsity and incompleteness of graphs compared with embedding-based methods. It is hard for them to find evidential paths reaching targets via strictly searching in the KG. While the embedding-based methods(Lin et al., 2018; Fu et al., 2019) map entities and relations into a unified semantic space to capture inner connections, which relaxes this restriction.

4.3 Ablation Study

We perform an ablation study to look deep into the framework of RARL. We deactivate the validity of rule information, random mechanism from RARL. The MRR results are summarized in Table 3. It can be observed that removing each reasoning component of RARL results in a significant performance drop on UMLS and WN18RR. On FB15K-237, removing rule information seems like to be no influence. As a result, we further conduct an analysis experiment using w/o rule setting and found lower results on the testing set. This performance gap may be caused by the difference of data distribution between the testing set and the validation set.

Besides, our ablation study shows that removing partial randomness has a greater negative impact on reasoning performance. This suggests that increasing the exploration diversity to get more valid path patterns is important in training stage.

	UMLS	WN18RR	FB15K-237
RARL w/o Rule	.813	.448	.551
RARL w/o Random	.792	.438	.501
RARL	.872	.455	.551

Table 3: Ablation study of the proposed method.

5 Conclusions

In this paper, we introduced RARL, a new RL-based method for knowledge graph reasoning. RARL makes use of high-quality symbolic rules and partical random beam search jointly and efficiently fights against the sparse reward and spurious path problems. Experimental results demonstrate that RARL achieves better performance compared with existing multi-hop methods in terms of both Hit@1 and MRR.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under grant 2016YFB1000902, the National Natural Science Foundation of China under grants U1911401, 61772501, 62002341 and U1836206, and the GFKJ Innovation Program.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*.

Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. 2019. Collaborative policy learning for open knowledge graph reasoning. *arXiv preprint arXiv:1909.00230*.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. 2015. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal—The International Journal on Very Large Data Bases*, 24(6):707–730.

Matt Gardner, Partha Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926*.

Joshua Hare. 2019. Dealing with sparse rewards in reinforcement learning. *arXiv preprint arXiv:1910.09281*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *arXiv preprint arXiv:1704.07130*.

Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. 2020. Reasoning on knowledge graphs with debate dynamics. *arXiv preprint arXiv:2001.00461*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*.

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In *Proceedings of the 24th international conference on Machine learning*, pages 433–440.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE.

Guanglin Niu, Yongfei Zhang, Bo Li, Peng Cui, Si Liu, Jingyang Li, and Xiaowei Zhang. 2020. Rule-guided compositional representation learning on knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2950–2958.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pages 3788–3800.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 15347–15357.

George Stoica, Otilia Stretcu, Emmanouil Antonios Platanios, Tom Mitchell, and Barnabás Póczos. 2020. Contextual parameter generation for knowledge graph link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3000–3008.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

Po-Wei Wang, Daria Stepanova, Csaba Domokos, and J Zico Kolter. 2019a. Differentiable learning of numerical rules in knowledge graphs. In *International Conference on Learning Representations*.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019b. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377.

XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages

Tahmid Hasan[1*], **Abhik Bhattacharjee**[1*], **Md Saiful Islam**[2], **Kazi Samin**[1],
Yuan-Fang Li[3], **Yong-Bin Kang**[4], **M. Sohel Rahman**[1], **Rifat Shahriyar**[1]
Bangladesh University of Engineering and Technology (BUET)[1], University of Rochester[2],
Monash University[3], Swinburne University of Technology[4]
`tahmidhasan@cse.buet.ac.bd`, `{abhik,samin}@ra.cse.buet.ac.bd`,
`mislam6@ur.rochester.edu`, `yuanfang.li@monash.edu`,
`ykang@swin.edu.au`, `{msrahman,rifat}@cse.buet.ac.bd`

Abstract

Contemporary works on abstractive text summarization have focused primarily on high-resource languages like English, mostly due to the limited availability of datasets for low/mid-resource ones. In this work, we present XL-Sum, a comprehensive and diverse dataset comprising 1 million professionally annotated article-summary pairs from BBC, extracted using a set of carefully designed heuristics. The dataset covers 44 languages ranging from low to high-resource, for many of which no public dataset is currently available. XL-Sum is highly abstractive, concise, and of high quality, as indicated by human and intrinsic evaluation. We fine-tune mT5, a state-of-the-art pretrained multilingual model, with XL-Sum and experiment on multilingual and low-resource summarization tasks. XL-Sum induces competitive results compared to the ones obtained using similar monolingual datasets: we show higher than 11 ROUGE-2 scores on 10 languages we benchmark on, with some of them exceeding 15, as obtained by multilingual training. Additionally, training on low-resource languages individually also provides competitive performance. To the best of our knowledge, XL-Sum is the largest abstractive summarization dataset in terms of the number of samples collected from a single source and the number of languages covered. We are releasing our dataset and models to encourage future research on multilingual abstractive summarization. The resources can be found at `https://github.com/csebuetnlp/xl-sum`.

Input Article: Yahoo's patents suggest users could weigh the type of ads against the sizes of discount before purchase. It says in two US patent applications that ads for digital book readers have been "less than optimal" to date. [...] "Greater levels of advertising, which may be more valuable to an advertiser and potentially more distracting to an e-book reader, may warrant higher discounts," it states. [...] It adds that the more willing the customer is to see the ads, the greater the potential discount. [...] At present, several Amazon and Kobo e-book readers offer full-screen adverts when the device is switched off and show smaller ads on their menu screens. [...] Yahoo does not currently provide ads to these devices, and a move into the area could boost its shrinking revenues.

Summary: Yahoo has signalled it is investigating e-book adverts as a way to stimulate its earnings.

Table 1: A sample article-summary pair from the XL-Sum dataset. To highlight the abstractiveness of the summary, we underline the novel words and phrases that do not appear in the article text. Also, portions of the article relevant to the summary have been color-coded. As we can see, these portions are concisely paraphrased in the summary, unlike extractive methods.

1 Introduction

Automatic text summarization (Nenkova and McKeown, 2011) is a fundamental problem in natural language processing. Given an input text (typically a long document or article), the goal is to generate a smaller, concise piece of text that conveys the key information of the input text. There are two main approaches to automatic text summarization: *extractive* and *abstractive*. Extractive methods crop out one or more segments from the input text and concatenate them to produce a summary. These methods were dominant in the early era of summarization, but they suffer from some limitations, including weak coherence between sentences, inability to simplify complex and long sentences, and unintended repetition (See et al., 2017; Widyassari et al., 2020).

---

[*] These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4693–4703
August 1–6, 2021.

Abstractive summarization, on the other hand, generates summaries that may contain words and phrases not present in the input text (e.g., via paraphrasing), and may arguably relate more to human-generated summaries (Hsu et al., 2018). Although abstractive summaries can be more coherent and concise than extractive summaries (Cohn and Lapata, 2008), generating them is more challenging due to the nature of this task. Limited availability of good datasets conducive to abstractive methods has made it even more difficult. For these reasons, extractive models had been performing better than abstractive ones historically. However, the success of sequence-to-sequence (seq2seq) models (Cho et al., 2014; Sutskever et al., 2014) over the last decade and the recent advances in Transformer-based models (Vaswani et al., 2017; Devlin et al., 2019) have rejuvenated abstractive text summarization (Rush et al., 2015; See et al., 2017; Zhang et al., 2020), which had previously received much less attention in comparison to extractive approaches (Nenkova and McKeown, 2012). Still, the scarcity of good datasets, especially for low-resource languages, remains a roadblock.

Typical seq2seq models are heavily data-driven, i.e., a large number of article-summary pairs are required to train them effectively. As a result, abstractive summarization has centered around the English language, as most large abstractive summarization datasets (Hermann et al., 2015; Grusky et al., 2018; Narayan et al., 2018) are available in English only. Though there have been some recent efforts for curating multilingual abstractive summarization datasets (Giannakopoulos et al., 2015; Cao et al., 2020; Scialom et al., 2020), they are limited in terms of the number of languages covered, the number of training samples, or both.

In this work, we introduce **XL-Sum**, a large-scale abstractive summarization dataset of news articles crawled from the British Broadcasting Corporation (BBC)[1] website. Using a custom crawler, we collect 1 million professionally annotated article-summary pairs covering 44 languages. Originating from a single source, these samples exhibit similar summarization strategies across all languages, making them ideal for the multilingual summarization task. XL-Sum introduces the first publicly available summarization dataset and benchmarks for many languages (e.g., Bengali, Swahili). Thus, this dataset potentially enables and facilitates research on low-resource languages, bringing technological advances to communities of these languages that have been traditionally under-served.

We achieve higher than 11 ROUGE-2 score on the 10 languages we benchmark on multilingual summarization, even exceeding 15 ROUGE-2 score (16.58 being the state-of-the-art for English, obtained by Zhang et al. (2020) on XSum (Narayan et al., 2018), a similar dataset) on many of them. In addition, we also conduct experiments on low-resource summarization task and show competitive results, indicating that the dataset can be used even for the low-resource languages individually.

In summary, we make the following main contributions in this paper:

- We release XL-Sum, a dataset containing 1 million article-summary pairs in 44 languages, being the first publicly available abstractive summarization dataset for many of them.
- We create a data curation tool that can automatically crawl and extract article-summary pairs from BBC, using which the dataset can be made even larger over time.
- We are the first to perform multilingual summarization on a diverse set of languages, achieving strong baselines on all languages tested.

We are releasing the dataset, curation tool, and summarization model checkpoints. We believe that our efforts in this work will encourage the community to push the boundaries of abstractive text summarization beyond the English language, especially for low and mid-resource languages.

2 The XL-Sum Dataset

In this section, we present details of the XL-Sum dataset together with the curation process. Table 2 shows the article-summary statistics for all languages in the XL-Sum dataset.

2.1 Content Source

BBC publishes news in 43 languages[2] ranging from low-resource languages such as Bengali and

---

[1] `https://www.bbc.com/`

[2] `https://www.bbc.co.uk/ws/languages`

Language	#Samples	Language	#Samples	Language	#Samples
Amharic	5,461	Korean	4,281	Somali	5,636
Arabic	40327	Kyrgyz	2,315	Spanish	44,413
Azerbaijani	7,332	Marathi	11,164	Swahili	10,005
Bengali	8,226	Nepali	5,286	Tamil	17,846
Burmese	5,002	Oromo	5,738	Telugu	11,308
Chinese	39,810	Pashto	15,274	Thai	6,928
English	301,444	Persian	25,783	Tigrinya	4,827
French	9,100	Pidgin[a]	9,715	Turkish	29,510
Gujarati	9,665	Portuguese	23,521	Ukrainian	57,952
Hausa	6,313	Punjabi	8,678	Urdu	40,714
Hindi	51,715	Russian	52,712	Uzbek	4,944
Igbo	4,559	Scottish Gaelic	1,101	Vietnamese	23,468
Indonesian	44,170	Serbian (Cyrillic)	7,317	Welsh	11,596
Japanese	7,585	Serbian (Latin)	7,263	Yoruba	6,316
Kirundi	5,558	Sinhala	3,414	**Total**	**1,005,292**

Table 2: Languages covered by the XL-Sum dataset, and the number of samples for each language. Here, a sample denotes an article-summary pair. If we consider languages with less than 15,000 training samples to be low-resource, then more than two-thirds of the constituent languages in XL-Sum fall into this category.

---

[a] West African Pidgin English

Swahili to high-resource ones including English and Russian. Among the 43 languages, Serbian is a special case that is published in both Cyrillic, the official script, and Latin, the colloquial script. We treat them as different languages in this work, totaling to a coverage of 44 languages.

2.2 Content Search

As BBC does not provide any archive or RSS feed on their website, we designed a crawler to recursively crawl pages starting from the homepage by visiting different article links present in each page visited. We were able to take advantage of the fact that all BBC sites have somewhat similar structures, and were able to scrape articles from all sites. We discarded pages with no textual contents (mostly pages consisting of multimedia contents) before further processing.

2.3 Article-Summary Extraction

The process of automatically collecting summaries of articles differs across different datasets. For example, the CNN/DM dataset (Hermann et al., 2015) merged bullet point highlights provided with the articles as reference summaries, whereas the XSum dataset (Narayan et al., 2018) used the first line of the article as the summary and the rest of the article as the input.

Our method of collecting summaries was made easier by the consistent editorial style of the BBC articles we crawled. BBC typically provides a summary of a whole article in the form of a bold paragraph containing one or two sentences at the beginning of each article. These summaries are written professionally by the authors of the articles in order to convey its main story within one small paragraph. This is in contrast to the headline which serves to draw the attention of viewers into reading the article. (We show an example article-summary pair from BBC English in Table 1.) We designed a number of heuristics to make the extraction effective by carefully examining the HTML structures of the crawled pages:

1. The desired summary must be present within the beginning two paragraphs of an article.
2. The summary paragraph must have some portion of texts in bold format.
3. The summary paragraph may contain some hyperlinks that may not be bold. The proportion of bold texts and hyperlinked texts to the total length of the paragraph in consideration must be at least 95%.
4. All texts except the summary and the headline must be included in the input text (including image captions).
5. The input text must be at least twice as large as the summary.

Any sample that did not conform to these heuristics were discarded. Our strategy of automatic annotation of summaries resembles XSum to some extent, but we found the first line to contain meta-information in many articles (e.g., author information, date of last modification). As such, we used the bold paragraphs as the summaries instead.

3 Human Evaluation of XL-Sum

Despite being written by professionals, evaluating the quality of the XL-Sum dataset is crucial to ensure that it can be valuable and used by a wider community for abstractive summarization. For this, we thoroughly conducted human evaluations on a subset of the dataset.

We hired professional annotators[3] to assess the quality of the top-10 languages according to the number of speakers worldwide.[4] It is worth noting that not all of these 10 languages are high-resource (e.g., Bengali is a low-resource language despite being one of the most widely spoken).

Each evaluator was asked to assess the quality of a random subset of the dataset (around 250 article-summary pairs) by answering the following questions with 'Yes'/'No':

Property A: Does the summary convey what the article is about?

Property B: If the answer to property `A` is 'Yes', does the summary contain any information that is inconsistent with the article?

Property C: If the answer to property `A` is 'Yes', does the summary contain any information that cannot be inferred from the article?

The motivation for designing these properties stemmed from recent progress on the quality estimation of neural language generation (NLG) models. Belinkov and Bisk (2018) showed that NLG models are vulnerable to noisy and low-quality training samples, hence, it is essential to validate the quality of the summary through Property `A`. Ensuring factual consistency and faithfulness (Wang et al., 2020; Maynez et al., 2020) of the generated summaries is also a key consideration for neural abstractive summarization since neural models have been shown to generate hallucinated texts (See et al., 2017). Property `B` checks for consistency between the article and the summary; while Property `C` assesses the hallucination implicitly by limiting the knowledge domain to the input article and identifying additional information present in the summary.

Language/Dataset	A	B	C
CNN/DM	98.33	1.22	24.57
XSum	92.00	0.00	71.74
English	99.66	0.00	37.37
Chinese	93.49	0.00	29.56
Hindi	90.91	0.00	31.42
Spanish	84.71	0.00	42.93
French	99.20	0.00	26.72
Arabic	98.34	0.00	25.31
Bengali	91.14	0.00	26.85
Russian	95.65	0.00	38.64
Portuguese	88.31	0.47	38.50
Indonesian	97.59	0.41	27.57

Table 3: Quality of XL-Sum (along with CNN/DM and XSum) assessed by human evaluators. In most of the cases, the evaluators agree that the summaries convey the main idea (Property A), and do not conflict with the input text (Property B). However, the summaries may contain some additional information (Property C), since the editors writing the summaries may use their common sense and additional domain knowledge not present in the article.

It is desirable that property `A` would have a high percentage of 'Yes', while both property `B` and `C` would have a low percentage. However, the third property can be a bit ambiguous, since some pieces of information may not be directly present in the input article, but a professional with background knowledge and understanding of the article topic may infer them inherently. Since text summarization is posed as a closed-domain task, we asked the annotators to label accordingly, i.e., not making use of any additional information outside the article. We provided them with some sample annotations in English to aid them during annotation. The percentages of 'Yes' for the three properties are shown in Table 3.[5] We also showed human evaluation of CNN/DM and XSum for contrast. Each article-summary pair was labeled by

---

[3] Each evaluator had at least an undergraduate degree and had native or bilingual proficiency in the languages they were assigned to. Bilingual proficiency was determined by professional language tests for the corresponding language.

[4] `https://w.wiki/Pss`

[5] We measured inter-annotator agreement using Cohen's kappa coefficient. Most scores were within the 0.7-0.9 range, showing high agreement between the evaluators

two separate evaluators, and we considered each pair to exhibit property `A` if both agreed upon it and property `B` and `C` if at least one of them did.

We can see from Table 3 (Property `A`) that most languages showed a high percentage of good summaries in the upper nineties, while some had a slightly lower percentage (i.e., Spanish and Portuguese). We were informed by the annotators that the negative summaries were mostly extracted from opinion pieces and blog posts, the bold paragraphs of which did not convey the main story of the articles.

Almost none of the summaries contained any conflicting information (Property `B`), while about one-third on average had information that was not directly inferrable (Property `C`). Interestingly, more than 75% of the latter had missing information such as first names, designations, or elaborations of acronyms. For example, a summary had the phrase 'President-elect Joe Biden' while its corresponding article had no presence of the first name 'Joe'. Another article had NHS elaborated in the summary, which was absent in the article. In general. the types of extra information carried in the summaries were more or less the same across all languages. CNN/DM and XSum exhibit extra information in the summaries as well, implying this phenomenon is common in abstractive text summarization datasets.

The presence of extra information in the summaries is understandable, since professional experts writing these summaries not only use the information present in the article text, but also incorporate their knowledge and understanding of the outside world. But for a closed-domain summarization model or a layman to the topic, inferring this information is not so straightforward, which makes the automatic abstractive summarization task more challenging. This phenomenon may explain why language models fine-tuned on pretrained checkpoints (Raffel et al., 2020; Qi et al., 2020; Zhang et al., 2020) are achieving state-of-the-art results in abstractive summarization, as they are able to make use of outside information gained from the high volume of texts they were pretrained with. Additionally, it would be interesting to investigate whether the recent trend of incorporating real-world knowledge and commonsense reasoning (Tandon et al., 2018; Deng et al., 2020) into language models could improve text summarization performance.

4 Intrinsic Evaluation of XL-Sum

Although the human evaluations provided good insights into the quality of the summaries, there are several other aspects of the summaries that are often infeasible or impractical to judge by human evaluators. With the above backdrop, several works (Narayan et al., 2018; Grusky et al., 2018; Bommasani and Cardie, 2020) have proposed many automatic metrics to quantify important features of abstractive summaries (e.g., novel words, abstractivity, compression, and redundancy).

Novel n-gram ratio: Narayan et al. (2018) proposed the percentage of n-grams in the summary that do not occur in the input article as a means of measuring abstractiveness.

Abstractivity: Grusky et al. (2018) introduced *fragments*, which greedily match text spans between the article and the summary, and Bommasani and Cardie (2020) generalized it to introduce *abstractivity* to measure absractiveness.

Compression: Bommasani and Cardie (2020) proposed *compression* as a metric to quantify conciseness. Compression is measured by

$$\textbf{CMP}(A, S) = 1 - \frac{|S|}{|A|} \tag{1}$$

where $|A|$ and $|S|$ denote the length of the article and the summary, respectively. We measured length in terms of number of tokens.

Redundancy: Although Bommasani and Cardie (2020) proposed a metric to measure *redundancy*, it is only applicable to multi-sentence summaries, which is not the case for most examples in XL-Sum. Thus, we propose a new redundancy metric by calculating the number of repetitive n-grams in the summary text.

Let $\{g_1, g_2, \cdots, g_m\}$ be the unique n-grams occurring in a summary S, and let $\{f_1, f_2, \cdots, f_m\}$ be their respective frequencies. Then the total number of repeated n-grams are $\sum_{i=1}^{m}(f_i - 1)$. We define redundancy as the ratio of redundant n-grams and the total number of n-grams in S:

$$\begin{aligned}\textbf{RED}(S) &= \frac{\sum_{i=1}^{m}(f_i - 1)}{\sum_{i=1}^{m} f_i} \\ &= 1 - \frac{m}{|S| - n + 1}\end{aligned} \tag{2}$$

It is preferred of a good summary to have a high novel n-gram ratio, abstractivity, and compression; while having a low redundancy score.

Language /Dataset	**Percentage of novel n-grams ↑**				**ABS ↑**	**CMP ↑**	**RED (n=1) ↓**	**RED (n=2) ↓**
	n = 1	**n = 2**	**n = 3**	**n = 4**				
CNN/DM	13.20	52.77	72.22	81.40	38.75	90.90	13.73	1.10
XSum	35.76	83.45	95.50	98.49	75.70	90.40	5.83	0.16
English	32.22	80.99	94.57	98.06	71.74	92.97	6.56	0.20
Chinese	36.13	79.23	91.14	94.58	70.23	92.95	7.37	0.50
Hindi	29.55	74.77	90.87	96.29	64.63	93.00	9.91	0.16
Spanish	32.63	76.29	91.96	96.57	66.60	92.49	11.45	.0.57
French	35.41	74.72	88.39	93.24	65.29	88.34	8.34	0.44
Arabic	49.88	84.56	94.79	98.10	76.72	90.62	3.93	0.18
Bengali	38.81	81.10	92.10	95.89	72.76	94.74	2.93	0.25
Russian	49.27	85.89	95.57	98.34	78.39	91.25	4.34	0.16
Portuguese	30.28	77.11	92.23	96.71	66.80	94.47	10.22	0.34
Indonesian	33.54	76.87	91.73	96.53	66.68	91.62	3.94	0.23

Table 4: Intrinsic evaluation of our XL-Sum dataset compared to CNN/Daily Mail and XSum. All values are reported in percentage for easier comparison. We use ↑ to indicate "higher is better" and ↓ for the reverse. Both of XL-Sum and XSum are highly abstractive, concise, and shows comparable quality, although the XSum dataset contains only English samples. For both XL-Sum and XSum, percentages of novel n-grams (n = 1, 2, 3, 4) are significantly higher than CNN/DM. High abstractiveness (ABS) scores of XL-Sum and XSum also bolster this finding. Additionally, low redundancy (RED) and high compression (CMP) values indicate that XL-Sum and XSum are more concise than CNN/DM.

We show these metrics in Table 4 (for redundancy, we report values for $n = 1, 2$). We also show these metrics for the CNN/DM and XSum datasets.

The results indicate that the XL-Sum dataset is highly abstractive—about one-third of the tokens, and more than 75% of the bigrams in the summaries are novel, and the abstractiveness score is also high (more than 65% for most of the languages). Additionally, XL-Sum is very concise (the summary is less than one-tenth of the input article for most of the languages), and contains minimal redundancy (less than 10% for the majority). The quality of XSum is comparable, however, it is limited to only one language (i.e., English). On the other hand, the CNN/Daily Mail dataset is significantly behind XL-Sum and XSum as indicated by most of the metrics mentioned above.

5 Experiments and Benchmarks

In previous sections, we have discussed the quality of XL-Sum. In addition, it is imperative to see how state-of-the-art models perform when trained on this dataset. Moreover, for many languages (e.g., Bengali, Swahili), currently, there is no publicly available dataset and benchmarks for abstractive text summarization to the best of our knowledge. In this section, we train summarization models with the XL-Sum dataset and provide several baselines and benchmark results.

Fine-tuning Transformer-based (Vaswani et al., 2017) seq2seq models initialized with pretrained weights from self-supervised training (Raffel et al., 2020; Liu and Lapata, 2019; Rothe et al., 2020; Qi et al., 2020; Zhang et al., 2020) has been shown to achieve state-of-the-art performance on many abstractive text summarization datasets. There are many multilingual pretrained checkpoints available through the Hugging Face Transformers Library (Wolf et al., 2020). Among them, we chose to use the mT5 model (Xue et al., 2021), a multilingual language model pretrained on a large dataset of 101 languages.

We performed summarization experiments in two settings: (i) multilingual, and (ii) low-resource. For performance reporting, for each language, we randomly sampled 500 pairs for its development set and 500 pairs for its test set, while using the rest of the pairs for training. We tokenized our training samples using the 250k wordpiece (Wu et al., 2016) vocabulary provided with the mT5 checkpoint. Due to computational constraints, we used the base model (600M parameters) and had to truncate the inputs to 512 tokens and the outputs to 64 tokens. We used the ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004) scores for automatic evaluation. For inference, we used beam search with beam size 4 and length penalty of $\alpha = 0.6$ (Wu et al., 2016).

5.1 Multilingual Summarization

Multilingual training is performed by training a single model with training samples from multiple languages. It has been previously used in several NLP tasks including neural machine translation (Arivazhagan et al., 2019) and language model pretraining (Conneau and Lample, 2019). However, multilingual training in the context of abstractive summarization has not been a major focus of the community. As such, the aim of this experiment is to demonstrate that a single model can perform well on summarizing texts in different languages, and that sister languages with morphological similarity can take advantage of positive transfer from each other which is not possible in monolingual settings.

For this experiment, we followed a similar training strategy as Conneau and Lample (2019): we sampled each batch from a single language containing 256 samples and used a smoothing factor (α) of 0.5 so that batches of low-resource languages would be sampled at a higher rate, increasing their frequency during training.

We fine-tuned the mT5 model for 35k steps on a distributed cluster of 8 Nvidia Tesla P100 GPUs for 4 days. We used the Adafactor optimizer (Shazeer and Stern, 2018) with a linear warmup of 5,000 steps and 'inverse square root' learning rate schedule. We show the ROUGE scores achieved by the model on the top-10 languages in Table 5.

Language	R-1	R-2	R-L
English	36.99	15.18	29.64
Chinese	36.89	15.23	30.52
Hindi	34.51	13.55	28.23
Spanish	30.93	12.14	23.76
French	34.47	15.94	27.53
Arabic	33.23	13.74	27.84
Bengali	28.32	11.43	24.23
Russian	31.10	13.47	25.54
Portuguese	31.06	11.62	23.39
Indonesian	36.17	16.70	30.50

Table 5: ROUGE scores for multilingual summarization achieved by the mT5 model when fine-tuned on the XL-Sum training set.

As we can see from the table, the multilingual model achieved higher than 11 ROUGE-2 scores on all languages. Some of these languages (e.g., Bengali) are low-resource, but the model still obtained competitive results comparable to high and mid-resource languages. Also, we are the first to report the abstractive summarization benchmark for a number of languages, including Bengali.

The mT5-base model achieves a R2-score of 15.18 on the English language. In comparison, the state-of-the-art PEGASUS$_{\text{BASE}}$ model (Zhang et al., 2020) obtained an R-2 score of 16.58 trained on the XSum English dataset, which is similar to XL-Sum in nature. This result suggests that the performance is comparable to the state-of-the-art English summarization. The R-2 scores for other languages are also similar to English, indicating that our dataset can help to effectively generate automatic summaries for all languages tested, including those low-resource ones.

5.2 Low-Resource Summarization

We have shown the effectiveness of the multilingual training strategy in summarizing articles for a wide set of languages with a single model. However, training the model is compute-heavy and it may not be realistic in many scenarios. For the dataset to be termed as versatile, models trained on individual languages should be able to perform competitively with the multilingual model. To confirm this is indeed the case, we performed training on five low-resource languages from Table 2 (Amharic, Azerbaijani, Bengali, Japanese, Swahili) in a compute-efficient setup. We fine-tuned mT5 on each language separately for 6-10 epochs (since the total training samples were limited, we had to be careful to prevent overfitting) on a single-GPU (Nvidia RTX 2080Ti) machine. For these experiments, we used a batch size of 32 and trained with a slanted learning rate schedule (Howard and Ruder, 2018). We show the ROUGE scores of each model in Table 6. We use the results from the multilingual models as baseline.

Language	Low-resource R-1/R-2/R-L	Multilingual R-1/R-2/R-L
Amharic	15.33/5.12/13.85	**17.49/6.34/15.76**
Azerbaijani	16.79/6.94/15.36	**19.29/8.20/17.62**
Bengali	25.33/9.50/22.02	**28.32/11.43/24.02**
Japanese	44.55/21.35/34.43	**47.17/23.34/36.20**
Swahili	34.29/15.97/28.21	**38.18/18.16/30.98**

Table 6: Performance of mT5 model fine-tuned on a low-resource training setup vs multi-lingual setup as mentioned in the previous section.

As evident by the results from Table 6, the multilingual model outperformed all the models trained on a single language. This is expected since similar languages can have positive transfer between them (Conneau et al., 2020) when trained together. However, the low-resource models do not trail by a large margin, in all cases the difference is not more than 2 for R-2 scores. This is a good indication that models fine-tuned on such a low amount of samples can still generalize to produce results competitive to multilingual models.

The case for Amharic, Azerbaijani and Japanese call for a discussion on their performance. The first two had comparatively low scores, while the last one (Japanese) had considerably higher scores compared to the other languages. Amharic and Azerbaijani had approximately 4k and 6k training samples respectively, which we conjecture is the primary reason behind their underperformance. Moreover, we did not find any reliable stemmer to preprocess the generated summaries before computing ROUGE, which may also hurt the scores. On the other hand, Japanese texts are not word-segmented and the words need to be separated before calculating ROUGE. We used Fugashi (McCann, 2020), and possibly due to its aggressive segmentation, the scores turned out to be higher than other languages. Similar high results have also been reported while measuring BLEU (Papineni et al., 2002) scores for machine translation evaluation in Japanese (Kudo, 2018).

Results in Table 6 show that although these languages are low-resource, the scores of the two setups are close, indicating our dataset can also be useful when used with a constrained computation capability. This is likely to contribute towards advances in low-resource text summarization, enabling fairness and access to the underserved communities.

6 Related Works

Rush et al. (2015); Nallapati et al. (2016) pioneered neural abstractive summarization, using recurrent attentional seq2seq models (Bahdanau et al., 2015). See et al. (2017) introduced Pointer-Generator networks for abstractive summarization, which can learn to copy words from the input text, in addition to generating new texts with the decoder. Gehring et al. (2017) proposed convolutional seq2seq models and applied it to perform abstractive summarization. Narayan et al. (2018) extended the work by integrating topic embeddings with the model.

Pretrained language models have recently been successfully applied to abstractive summarization. Liu and Lapata (2019) initialized the encoder and Rothe et al. (2020) initialized both the encoder and the decoder of a seq2seq model with the pretrained BERT (Devlin et al., 2019) weights and fine-tuned the models for abstractive summarization. Raffel et al. (2020); Qi et al. (2020) used fully pre-trained seq2seq models, while Zhang et al. (2020) introduced a summarization-specific pretraining objective to achieve state-of-the-art results on multiple datasets.

Most works on abstractive summarization have so far focused on English, in large part due to a lack of benchmark datasets for other languages. Giannakopoulos et al. (2015) introduced MultiLing 2015, a summarization dataset spanning 40 languages. However, MultiLing 2015 is limited in size, with the training set having only 10k samples in total. Cao et al. (2020); Scialom et al. (2020) introduced two new datasets for multilingual summarization, but both were limited to less than 10 languages. Moreover, samples for different languages were collected from different sources, making them exposed to different types of summarization strategies, which raises questions on the uniformity of the summaries.

7 Conclusion and Future Works

In this paper, we present XL-Sum, a large-scale, high-quality multilingual text summarization dataset, containing 1 million samples across 44 languages collected from a single source, BBC. For many of the languages, XL-Sum provides the first publicly available abstractive summarization dataset and benchmarks. We also make the dataset curation tool available for the researchers, which will help to grow the dataset over time. Thorough human and intrinsic evaluations indicate that the summaries in our dataset are highly abstractive and concise while conveying the main idea with little to no conflict with the input article. Additionally, we demonstrate that multilingual training can help towards better summarization, most likely due to the positive transfer between sister languages with morphological similarity. Moreover, XL-Sum is also useful in a low-resource and compute-efficient setting.

In future, we will investigate the use of our dataset for other summarization tasks (e.g., cross-lingual summarization Zhu et al., 2019).

We hope the XL-Sum dataset will be helpful for the research community, especially for the researchers working to ensure fair access to technological advances to under-served communities with low-resource languages.

Acknowledgements

This work was performed using the OzSTAR national facility at the Swinburne University of Technology. The OzSTAR program receives funding in part from the Astronomy National Collaborative Research Infrastructure Strategy (NCRIS) allocation provided by the Australian Government.

References

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, California, USA.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, BC, Canada.

Rishi Bommasani and Claire Cardie. 2020. Intrinsic evaluation of summarization datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8075–8096, Online. Association for Computational Linguistics.

Yue Cao, Xiaojun Wan, Jinge Yao, and Dian Yu. 2020. Multisumm: Towards a unified model for multilingual abstractive summarization. In *Proceedings of Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 11–18. AAAI Press.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 137–144, Manchester, UK.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 7059–7069, Vancouver, Canada.

Changyu Deng, Xunbi Ji, Colton Rainey, Jianyu Zhang, and Wei Lu. 2020. Integrating machine learning with human knowledge. *iScience*, 23(11):101656.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70, pages 1243–1252. PMLR.

George Giannakopoulos, Jeff Kubina, John Conroy, Josef Steinberger, Benoit Favre, Mijail Kabadjov, Udo Kruschwitz, and Massimo Poesio. 2015. MultiLing 2015: Multilingual summarization of single and multi-documents, on-line fora, and call-center conversations. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 270–274, Prague, Czech Republic. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings* *of the 28th International Conference on Neural Information Processing Systems (NIPS 2015)*, pages 1693–1701, Montreal, Canada.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Paul McCann. 2020. fugashi, a tool for tokenizing Japanese in python. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 44–51, Online. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar GuÌ‡lçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. MLSUM: The multilingual summarization corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8051–8067, Online. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost.

In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80, pages 4603–4611. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS 2014)*, pages 3104–3112, Montreal, Canada.

Niket Tandon, Aparna S Varde, and Gerard de Melo. 2018. Commonsense knowledge in machine intelligence. *ACM SIGMOD Record*, 46(4):49–52.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, page 6000–6010, Long Beach, California, USA.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, et al. 2020. Review of automatic text summarization techniques & methods. *Journal of King Saud University - Computer and Information Sciences*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119, pages 11328–11339. PMLR.

Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang, and Chengqing Zong. 2019. NCLS: Neural cross-lingual summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3054–3064, Hong Kong, China. Association for Computational Linguistics.

Use of Formal Ethical Reviews in NLP Literature: Historical Trends and Current Practices

Sebastin Santy[♠] **Anku Rani**[♡] **Monojit Choudhury**[♠]
[♠] Microsoft Research, Bangalore, India
[♡] Plaksha University, Mohali, India
[♠]{t-sesan, monojitc}@microsoft.com, [♡]anku.rani@plaksha.org

Abstract

Ethical aspects of research in language technologies have received much attention recently. It is a standard practice to get a study involving human subjects reviewed and approved by a professional ethics committee/board of the institution. How commonly do we see mention of ethical approvals in NLP research? What types of research or aspects of studies are usually subject to such reviews? With the rising concerns and discourse around the ethics of NLP, do we also observe a rise in formal ethical reviews of NLP studies? And, if so, would this imply that there is a heightened awareness of ethical issues that was previously lacking? We aim to address these questions by conducting a detailed quantitative and qualitative analysis of the ACL Anthology, as well as comparing the trends in our field to those of other related disciplines, such as cognitive science, machine learning, data mining, and systems.

1 Introduction

With the rapid advances in the field of Natural Language Processing (NLP), language technologies are getting woven into the daily fabric of our lives and the society. Since “language is a portal of emotions, a proxy of human behavior, and a strong signal of individual characteristics” (Hovy and Spruit, 2016), large-scale deployment of language technology has potential risks that require early detection and mitigation. Naturally, there have been several discussions about the potential harms and ethical issues concerning NLP (Hovy and Spruit, 2016; Conway and O’Connor, 2016). They have mostly revolved around building or deploying systems in sensitive areas such as hate speech (Sap et al., 2019), social media (Benton et al., 2017), clinical NLP and mental health (Šuster et al., 2017; Mikal et al., 2016) and use of sensitive or personal information (Larson, 2017). While building NLP systems, there are also ethical risks associated with involvement of human subjects through user studies or data collection activities (Shmueli et al., 2021).

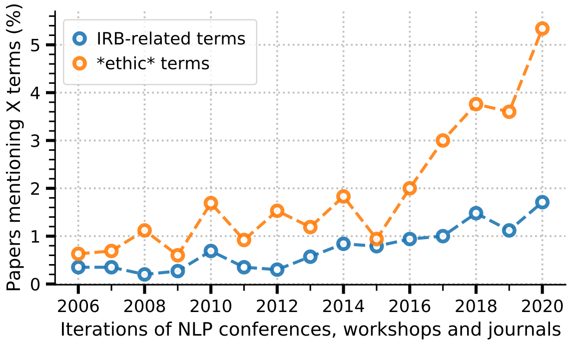

Figure 1: Percentage(%) of papers mentioning IRB-related and *ethic* terms in NLP conferences, journals, and workshops from 2006 to 2020.

The awareness of the dangers of the existing and new NLP applications has led to the curation of several ethical guidelines and frameworks. Undergirded by lessons from the past, these guidelines and frameworks help researchers consider and contextualize critical ethical concerns. Most of the ethical issues in NLP are rooted in the data being used for research. Couillault et al. (2014) is one of the first works to explore the ethics of data collection and evaluation in NLP. Several other works have proposed best practices for dealing with ethical implications of NLP research and deployment (Prabhumoye et al., 2019; Leidner and Plachouras, 2017; Bender and Friedman, 2018; Schnoebelen, 2017). There is now an increased awareness around this topic with a number of workshops and tutorials on ethics at NLP conferences (Tsvetkov et al., 2018; Hovy et al., 2017; Alfano et al., 2018). Such discussions have resulted in a number of reforms at NLP conferences. NLP conferences now have new track called *Ethics in NLP*. Furthermore, several ML and NLP conferences such as **NeurIPS** 2020,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4704–4710
August 1–6, 2021.

NAACL 2021 and **ACL** 2021[1] now recommend the inclusion of broader impact statement in their papers, which allows for authors to introspect and be mindful of the ethical implications their research poses.

Although an NLP researcher might be individually committed towards ethical research practices, they may not have the expertise in ethical and legal issues to gauge the potential risks of a technology or a dataset that they are building. The practice of getting the research/study approved by an ethical review board (aka Institutional Review Board or IRB)[2] instituted by the organization is thus critical in early defusal of the potential harms. The two primary functions of IRB are (i) to protect the rights and welfare of human research subjects, and (ii) to support and facilitate the conduct of valuable research (Bankert and Amdur, 2006; Klitzman, 2012; Byerly, 2009). Traditionally, IRB has been a long-standing norm in biomedical research due to its overt exposure to human subjects. However, with computing research pervading human lives, IRBs have started covering computing research as well (Buchanan and Ess, 2009; Vitak et al., 2017). With regards to NLP, most of the data collection and annotation processes as well as user studies come under the purview of these boards. These are particularly necessary if they cover sensitive topics such as mental health or hate speech which can affect the human subjects involved in data collection or the users of the system.

How frequently do NLP researchers take IRB approvals for their studies? What aspects of NLP research or which topics of study are typically considered for IRB approvals? What are the historical and current trends, and what can we say about the awareness of the NLP research community around ethical issues? In this paper, we try to answer these questions through a quantitative and qualitative analysis of papers from the ACL anthology that seek IRB approvals. According to our findings, IRB approvals were almost non-existent in the NLP literature until 2006, but there has been a steady increase since 2016. We also study the distribution of IRB approvals by country and industry/academia affiliation, as well as compare the recent trends in NLP conferences to that of various prominent conferences ranging from machine learning and data mining to human-computer interaction and systems. One of the key findings of this study is that IRB permission was mostly sought for either data collection or annotation studies, but hardly ever for data re-purposing or system design/deployment - a void that we think the NLP community should be conscious about.

2 Method

To determine the trends of IRB approvals in NLP research, we resort to searching for IRB- and ethics-related terms in research papers. We obtain the papers (PDFs) for major NLP conferences, journals and workshops [**ACL**, **COLING**, **EACL** **EMNLP**, **LREC**, **NAACL**, **CL**, **TACL**, and **WS**] from the ACL Anthology (curated by Joshi et al. (2020)). For a comparative analysis, we also collect papers from other related conferences [**CogSci**, **InterSpeech**, **NeurIPS**, **CVPR**, **ICWSM**, **CHI**, **COMPASS**] for the years 2019 and 2020, during which there was considerably more discussion around ethics of computing research.

In order to retrieve papers that seek IRB approvals, we search for the following keywords which cover the phrases used for IRB in countries that are frequently represented at these conferences: review board, ethics board, ethics panel, ethics committee, consent form, and IRB.[3] To further compare and calibrate, we also search for papers that contain the wildcard string *ethic*, which brings up a broader set of papers that may discuss ethical repercussions of their work, even if any approval is not explicitly sought or mentioned. To assist with a robust search over this textual data, we use the Allen AI SPIKE interface[4] (Taub Tabib et al., 2020; Shlain et al., 2020) and use pdfgrep[5] to cross-check our results.

IRB-related term search yielded 210 papers from the ACL anthology (till 2019), which were then manually checked for precision and annotated for *aspects* (see Figure 3) and *topics* (e.g., hate speech, social media, mental health, etc.) of the research for which IRB permission was sought. Through our manual curation, we found that 94.17% of these papers actually took the approval for their research study thus showing that our search is precise in capturing the terms.

---

[1] `https://2021.aclweb.org/ethics/Ethics-FAQ/`

[2] In this paper, we use IRB as a generic term to refer to such review boards which are known by slightly different terms in different geographies.

[3] Collectively referred to as IRB-related terms from hereon.

[4] URL when accessed: `https://spike.staging.apps.allenai.org/datasets/acl/search`

[5] Command-line tool: `https://pdfgrep.org/`

The remaining papers were mostly ethical frameworks and recommendations (e.g., Hovy and Spruit (2016); Bender and Friedman (2018)) which merely mentioned the need for seeking IRB approvals in NLP research.

3 Findings

3.1 How many papers seek IRB approvals?

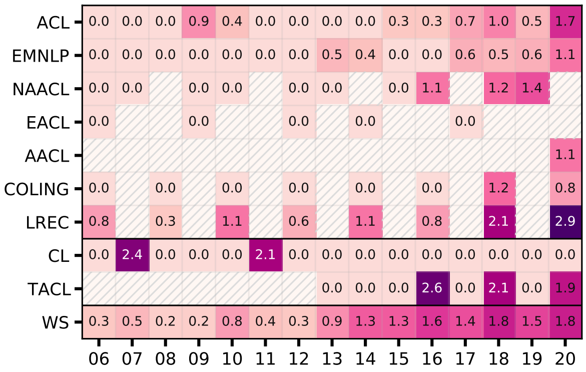

	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
ACL	0.0	0.0	0.0	0.9	0.4	0.0	0.0	0.0	0.0	0.3	0.3	0.7	1.0	0.5	1.7
EMNLP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.4	0.0	0.0	0.6	0.5	0.6	1.1
NAACL	0.0	0.0		0.0	0.0		0.0	0.0		0.0	1.1		1.2	1.4	
EACL	0.0			0.0			0.0		0.0			0.0			
AACL															1.1
COLING	0.0		0.0		0.0		0.0		0.0		0.0		1.2		0.8
LREC	0.8		0.3		1.1		0.6		1.1		0.8		2.1		2.9
CL	0.0	2.4	0.0	0.0	0.0	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TACL								0.0	0.0	0.0	2.6	0.0	2.1	0.0	1.9
WS	0.3	0.5	0.2	0.2	0.8	0.4	0.3	0.9	1.3	1.3	1.6	1.4	1.8	1.5	1.8

Figure 2: The percentage (%) of papers in NLP conferences, journals and workshops over the past 15 years that mention <u>IRB-related</u> terms. The intensity of color is proportional to the % values. The boxes with a gray hatch reflect the years when that particular conference was not held.

Figure 2 shows the percentage(%) of papers in each NLP conference iteration that mention IRB-related terms. It is immediately obvious that for almost all the conferences only a minuscule number of papers mention IRB approvals. However, it is heartening to see that the number of mentions is increasing in recent years. **LREC** and **WS** particularly stand out among the other conferences for having at least some mentions of IRB approvals in every iteration. For **LREC**, it is understandable, since the theme of the conference revolves around data and resource generation. In the case of **WS**, IRB mentions are consistently increasing over the years. We observed that this is mostly due to the diverse nature of workshops some of which are on resource generation (Popescu-Belis et al., 2019) or cover sensitive topics (Niederhoffer et al., 2019; Yannakoudakis et al., 2019). Journals such as **CL** and **TACL** have very few papers in each iteration, so even one IRB mention appears to be a lot. It should be noted that there is a possibility that a research study has obtained IRB approval but has not disclosed it in their paper. However, based on the authors' experience (and anecdotes from personal conversations), it is highly unlikely that anyone who has been through the IRB approval process will fail to mention it.

3.2 What kinds of research seek IRB approvals?

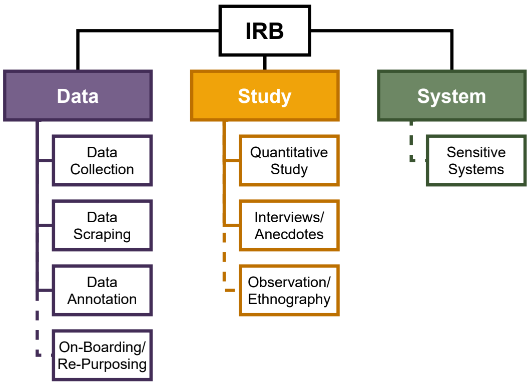

Figure 3: Different *aspects* of research for which IRB approvals were sought in the papers that we manually analyzed.

We manually go through each of the 58 NLP papers (which excludes **WS**) to derive the *aspects* and understand the context in which IRB approvals are sought and build a taxonomy of the broad topics covered (Figure 3). We see that most of them (24 papers; 41.3%) take IRB approvals for collection of data which can often involve human subjects directly. It is followed by the annotation of data with 20 papers (34%) taking IRB approvals. A meager 7 papers (12%) in our set take IRB approvals for scraping data which is the automatic collection of data from web pages or social media posts without explicit consent from the users. We see that only one paper takes IRB approval for re-purposing and further annotation of data (Rogers et al., 2018). One of the core concerns of GDPR is the usage of personal data collected by media platforms for a purpose different than what the user consented to and hence such re-purposing of data should ideally undergo IRB approvals. 12 papers (20%) take approvals for conducting user studies of both qualitative (survey, interview) and quantitative nature (semantic edits). Interestingly, we see that only one paper has taken IRB approval for the whole system owing to its sensitive nature (Cao et al., 2018).

We also look at the nature (or *topics*) of the work for which the IRB approvals are taken. We observe that 48.4% of papers that mention IRB have sensitive topics (such as mental health, hate speech, clinical/medical NLP), 20.3% of the papers are for collection of eye movement, EEG and audio/video recordings of human subjects, and rest of them are generic data collection or user study. To further understand the trends, we look at certain tracks of **ACL** 2020 which deal with sensitive topics or

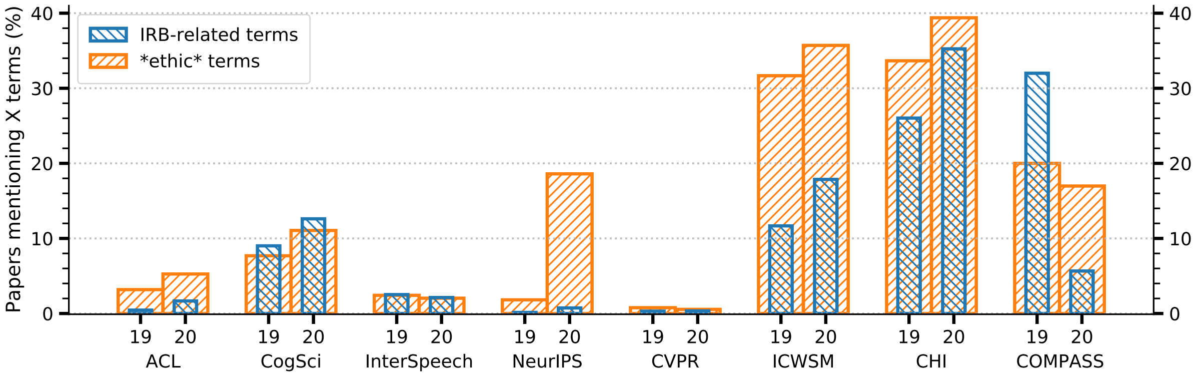

Figure 4: Percentage(%) of papers mentioning IRB-related and *ethic* terms in related conferences.

data collection. We notice that only 1/42 *Resource and Evaluation* papers and 3/24 *Computation Social Science* papers have taken IRB approvals. It is worth noting that 29/210 papers we have annotated only used an informed consent form without explicitly mentioning whether an ethical review board was involved in the process.

3.3 What is the distribution of IRB approvals by country and industry/academia?

	Total Papers	**Percent Papers**
Countries		
USA	47/5368	0.88%
Canada	5/358	1.40%
Germany	5/850	0.59%
UK	5/1088	0.46%
Netherlands	3/226	1.33%
Sweden	2/100	2.00%
South Korea	2/151	1.32%
China	2/2350	0.09%
Affiliation Types		
University	52/7730	0.67%
Industry	1/841	0.12%
National Lab	1/182	0.55%
Joint/Collaboration	11/2651	0.41%

Table 1: Distribution of % IRB-related term mentions among countries and different types of affiliations for NLP conferences (excluding **LREC** and **WS**) from 2012 to 2020. [6]

Table 1 shows the distribution of papers which mention IRB approvals along two dimensions: countries and types of institutions. As can be observed, most of the listed countries are WEIRD[7] societies. When it comes to the type of institution, we find that universities account for the vast majority of papers seeking IRB approvals, followed by joint collaborations. This trend can be counter-intuitive as an industry is more likely to be regulated and accountable for the ethical and legal concerns of their work. One possibility is that industries perhaps do not engage in external data collection/annotation work or conduct user studies as much as academic institutions do. Alternatively, it is possible that the data collection/annotation process is a completely independent pipeline that is not specific to the research paper in which it is used and thus is not reported.

3.4 How do the IRB trends in NLP research compare with those in related fields?

We look at the following conferences for our analysis: **CogSci** in cognitive science, **InterSpeech** in speech processing, **NeurIPS** in machine learning, **CVPR** in computer vision, **ICWSM** in social media mining, **CHI** in human-computer interaction, and **COMPASS** in computing systems deployment. We specifically analyze these conferences for 2019 and 2020 iterations as there have been significant changes made in the conferences during this period in terms of reporting the ethical ramifications of their research. Figure 4 shows the % of papers mentioning *ethic* and IRB-related terms for each conference iteration. We calculate for *ethic* to understand how aware and concerned each field/conference is towards the ethical implications of the research they conduct.

It is not surprising that IRB mentions for **CHI** are so high ($\sim 35\%$) given that more than 65% percent of CHI papers include at least one user study (Koeman, 2020). **ICWSM** works with datasets and systems related to web and social media analytics and hence would need to undergo IRB approvals. This is apparent in the relatively high number of IRB

---

[6] Country and Affiliation data obtained from `https://github.com/marekrei/ml_nlp_paper_data/`

[7] Western, Educated, Industrialized, Rich and Democratic

mentions in both 2019 and 2020. Unlike the other conferences we choose for our analysis, **CogSci** is a non-computing conference. Linguistics work is frequently found in **CogSci**, which often makes use of human subjects. We observe that it has the most consistent representation of both *ethic* and IRB-related term mentions among the years. As previously discussed, one of the concerns is that sensitive systems are seldom taking IRB approvals. On the contrary, we notice that **COMPASS**, a conference largely focused on deploying computing systems, is prevalent in taking IRB approvals. **InterSpeech** and **CVPR** have significantly fewer papers with IRB mentions ($< 0.35\%$ and $< 2.5\%$, respectively) and the trends have hardly changed over the years, despite the fact that they conduct research with speech, multimodal, and vision data that may have been collected from human subjects. Among these, there is a ray of hope for **ACL** which is showing a significant positive trend in both *ethic* and IRB mentions without any external reinforcement, thanks to the increasing awareness in the field. **NeurIPS**, on the other hand, has seen a meteoric rise in their *ethic* mentions, which, on manual inspection reveals, is due to their mandatory inclusion of broader impact statements. There has also been a slight increase in their IRB mentions, which could be attributed to this, indicating that broader impact statements might help researchers be more cautious when proposing their research to the larger community. This quantitative testimony from **NeurIPS** shows that **ACL** and other *CL conferences are moving in the right direction with their inclusion of stringent ethics reviews for their papers.

4 Way Forward

In this paper, we conduct a survey of IRB approvals in NLP research. The two key observations we make are as follows. First, very few papers ($< 0.8\%$ of all papers published) since 2006 have sought an IRB approval; though we do observe a rise in numbers ($< 1.3\%$ of all papers published) since 2016. This is much smaller compared to the numbers we observe for other conferences such as **CogSci**, **CHI**, **ICWSM** or **COMPASS**. Second, the majority of the IRB approvals were obtained for data collection or annotation that directly involved users, with only a few studies seeking approvals for data scraping or re-purposing. Such approvals are even more scarce for sensitive systems where we seldom see any paper taking IRB approvals solely for the system. The number of papers creating new datasets is expected to be greater than 1% of all NLP papers[8]; the number of papers that re-purpose an existing dataset is expected to be even greater than this. Therefore, clearly not all papers creating datasets, and almost no paper re-purposing datasets take approvals from IRB. As such, re-purposing data collected from human subjects without their explicit consent on how the data will be re-used is potentially dangerous and may even have legal repercussions. Furthermore, with the exception of a couple of papers, to date, there is no practice or trend of taking IRB approval for designing, developing, and deploying systems. This is in stark contrast to the practice in other related fields/conferences such as **COMPASS**. Much of the harm caused by a system could actually come from its design or style of training or deployment, rather than the underlying datasets.

We see that the broad impact statements have helped conferences such as **NeurIPS** which were traditionally oblivious to ethical issues (Nanayakkara et al., 2021). We believe that, in a similar way, the impact statements introduced in **NAACL**[9] and **ACL** 2021, with specific clauses for seeking IRB, will be highly beneficial in limiting the aforementioned potential risks by increasing the awareness amongst researchers of broader ethical repercussions of their research. It will be interesting to conduct a similar study a few years down the line and contrast with the findings of the current study.

Acknowledgements

We thank Prof. Emily M. Bender (UW) for providing useful insights and inspiration at the early stages of this work. We also thank Dr. Mary L. Gray and Ms. Anu Kannepalli (Microsoft) for their many insightful inputs, Prof. Yoav Goldberg (Allen AI) for setting up the SPIKE instance for us, and Prof. Katharina Reinecke (UW) and Dr. Prasanta Bhattacharya (A*STAR) for the broad discussions around IRBs. We also thank anonymous reviewers from ACL and NLP4PI workshop for their thoughtful suggestions and feedback.

---

[8] As a crude statistics, consider the fact that the number of accepted papers in the *Resource and Evaluation* track in **ACL** 2020 was 5.4%, whereas only 1.7% of all the papers in that year sought for IRB approvals.

[9] Report: `https://2021.naacl.org/blog/ethics-review-process-report-back/`

References

Mark Alfano, Dirk Hovy, Margaret Mitchell, and Michael Strube, editors. 2018. *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*. Association for Computational Linguistics, New Orleans, Louisiana, USA.

Elizabeth A Bankert and Robert J Amdur. 2006. *Institutional review board: Management and function*. Jones & Bartlett Learning.

Emily M. Bender and Batya Friedman. 2018. Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Adrian Benton, Glen Coppersmith, and Mark Dredze. 2017. Ethical Research Protocols for Social Media Health Research. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 94–102, Valencia, Spain. Association for Computational Linguistics.

Elizabeth A. Buchanan and Charles M. Ess. 2009. Internet Research Ethics and the Institutional Review Board: Current Practices and Issues. *SIGCAS Comput. Soc.*, 39(3):43–49.

Wesley G. Byerly. 2009. Working with the institutional review board. *American Journal of Health-System Pharmacy*, 66(2):176–184.

Xuân-Nga Cao, Cyrille Dakhlia, Patricia Del Carmen, Mohamed-Amine Jaouani, Malik Ould-Arbi, and Emmanuel Dupoux. 2018. BabyCloud, a Technological Platform for Parents and Researchers. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Mike Conway and Daniel O'Connor. 2016. Social media, big data, and mental health: current advances and ethical implications. *Current Opinion in Psychology*, 9:77–82.

Alain Couillault, Karën Fort, Gilles Adda, and Hugues de Mazancourt. 2014. Evaluating corpora documentation with regards to the ethics and big data charter. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4225–4229, Reykjavik, Iceland. European Language Resources Association (ELRA).

Dirk Hovy, Shannon Spruit, Margaret Mitchell, Emily M. Bender, Michael Strube, and Hanna Wallach, editors. 2017. *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*. Association for Computational Linguistics, Valencia, Spain.

Dirk Hovy and Shannon L. Spruit. 2016. The Social Impact of Natural Language Processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The State and Fate of Linguistic Diversity and Inclusion in the NLP World. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Robert Klitzman. 2012. Institutional Review Board Community Members: who are they, what do they do, and whom do they represent? *Academic Medicine*, 87(7):975–981.

Lisa Koeman. 2020. HCI/UX research: what methods do we use?

Brian Larson. 2017. Gender as a Variable in Natural-Language Processing: Ethical Considerations. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 1–11, Valencia, Spain. Association for Computational Linguistics.

Jochen L. Leidner and Vassilis Plachouras. 2017. Ethical by Design: Ethics Best Practices for Natural Language Processing. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 30–40, Valencia, Spain. Association for Computational Linguistics.

Jude Mikal, Samantha Hurst, and Mike Conway. 2016. Ethical issues in using twitter for population-level depression monitoring: a qualitative study. *BMC Medical Ethics*, 17(1).

Priyanka Nanayakkara, Jessica Hullman, and Nicholas Diakopoulos. 2021. Unpacking the Expressed Consequences of AI Research in Broader Impact Statements. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, New York, NY, USA. Association for Computing Machinery.

Kate Niederhoffer, Kristy Hollingshead, Philip Resnik, Rebecca Resnik, and Kate Loveys, editors. 2019. *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology*. Association for Computational Linguistics, Minneapolis, Minnesota.

Andrei Popescu-Belis, Sharid Loáiciga, Christian Hardmeier, and Deyi Xiong, editors. 2019. *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*. Association for Computational Linguistics, Hong Kong, China.

Shrimai Prabhumoye, Elijah Mayfield, and Alan W Black. 2019. Principled Frameworks for Evaluating Ethics in NLP Systems. In *Proceedings of the 2019 Workshop on Widening NLP*, pages 118–121, Florence, Italy. Association for Computational Linguistics.

Anna Rogers, Alexey Romanov, Anna Rumshisky, Svitlana Volkova, Mikhail Gronas, and Alex Gribov. 2018. RuSentiment: An enriched sentiment analysis dataset for social media in Russian. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 755–763, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The Risk of Racial Bias in Hate Speech Detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence, Italy. Association for Computational Linguistics.

Tyler Schnoebelen. 2017. Goal-Oriented Design for Ethical Machine Learning and NLP. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 88–93, Valencia, Spain. Association for Computational Linguistics.

Micah Shlain, Hillel Taub-Tabib, Shoval Sadde, and Yoav Goldberg. 2020. Syntactic search by example. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 17–23, Online. Association for Computational Linguistics.

Boaz Shmueli, Jan Fell, Soumya Ray, and Lun-Wei Ku. 2021. Beyond Fair Pay: Ethical Implications of NLP Crowdsourcing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3758–3769, Online. Association for Computational Linguistics.

Simon Šuster, Stéphan Tulkens, and Walter Daelemans. 2017. A Short Review of Ethical Challenges in Clinical Natural Language Processing. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 80–87, Valencia, Spain. Association for Computational Linguistics.

Hillel Taub Tabib, Micah Shlain, Shoval Sadde, Dan Lahav, Matan Eyal, Yaara Cohen, and Yoav Goldberg. 2020. Interactive extractive search over biomedical corpora. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 28–37, Online. Association for Computational Linguistics.

Yulia Tsvetkov, Vinodkumar Prabhakaran, and Rob Voigt. 2018. Socially Responsible NLP. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 24–26, New Orleans, Louisiana. Association for Computational Linguistics.

Jessica Vitak, Nicholas Proferes, Katie Shilton, and Zahra Ashktorab. 2017. Ethics regulation in social computing research: Examining the role of institutional review boards. *Journal of Empirical Research on Human Research Ethics*, 12(5):372–382.

Helen Yannakoudakis, Ekaterina Kochmar, Claudia Leacock, Nitin Madnani, Ildikó Pilán, and Torsten Zesch, editors. 2019. *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Florence, Italy.

As Easy as 1, 2, 3: Behavioural Testing of NMT Systems for Numerical Translation

Jun Wang,[1] **Chang Xu,**[1] **Francisco Guzmán,**[2] **Ahmed El-Kishky,**[3*]
Benjamin I. P. Rubinstein,[1] **Trevor Cohn**[1]
[1]University of Melbourne, Australia
[2]Facebook AI, [3]Twitter Cortex
`jun2@student.unimelb.edu.au`
`{xu.c3,benjamin.rubinstein,trevor.cohn}@unimelb.edu.au`
`fguzman@fb.com, aelkishky@twitter.com`

Abstract

Mistranslated numbers have the potential to cause serious effects, such as financial loss or medical misinformation. In this work we develop comprehensive assessments of the robustness of neural machine translation systems to numerical text via behavioural testing. We explore a variety of numerical translation capabilities a system is expected to exhibit and design effective test examples to expose system underperformance. We find that numerical mistranslation is a general issue: major commercial systems and state-of-the-art research models fail on many of our test examples, for high- and low-resource languages. Our tests reveal novel errors that have not previously been reported in NMT systems, to the best of our knowledge. Lastly, we discuss strategies to mitigate numerical mistranslation.

1 Introduction

Just as neural machine translation (NMT) systems have achieved tremendous benchmark results, they have been proven brittle when faced with irregular inputs such as noisy text (Belinkov and Bisk, 2018; Michel and Neubig, 2018) or adversarial inputs (Cheng et al., 2020). Among such errors, mistranslation of *numerical text* constitutes a crucial but under-explored category that may have profound implications. For example, in the medical domain, mistranslating the number of confirmed cases of a contagious disease like COVID-19 may exacerbate public health misinformation. Numerical errors made in financial document translation, *e.g.*, an extra or omitted digit or decimal point, could lead to significant monetary loss. Surprisingly, we find that numerical mistranslation is a general issue faced by state-of-the-art NMT systems, including commercial and research systems, with evidence present across contexts: for both high and low resource languages, and for both close and distant languages.

Type	**Input**	**Output**
Separators	The distance is 557,601.101 meters.	**[De]**: Die Entfernung beträgt 557.601.101 Meter.
Numerals	The total weight is two hundred and two kg.	**[Zh]**: 总重量为220公斤。
Digits	The R0 of the disease is 3.28.	**[Ne]**: रोगको R0 28.२28 हो।
Units	There were 100.01 million cases worldwide.	**[Zh]**: 全世界有1.001 亿病例。(100.1 million)

Table 1: Numerical errors discovered by our method when behavioural testing two popular commercial translation systems using their public APIs.

De facto standard metrics such as BLEU (Papineni et al., 2002) may fail to flag a numerical translation error, which only contributes a very minor penalty, as it is typically a single-token mistranslation. To facilitate the discovery of numerical errors made by NMT systems, we propose a black-box test method[1] for assessing and debugging the numerical translation of NMT systems in a systematic manner. Our method extends the `CheckList` behavioural testing framework (Ribeiro et al., 2020) by designing automatic test cases to assess a suite of fundamental capabilities a system should exhibit in translating numbers.

Our tests on state-of-the-art NMT systems expose novel error types that have evaded close examination (Table 1). These error types greatly extend the *number* category (`NUM`) of the *catastrophic errors* (Specia et al., 2020) of NMT systems with richer error types. Finally, the abuse of these errors constitute vectors of attack: error-prone numerical tokens injected into monolingual data may

---

[*] This work was conducted while author was working at Facebook AI

[1] Our code is available at `https://github.com/JunW15/NumberTest`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4711–4717
August 1–6, 2021. ©2021 Association for Computational Linguistics

Capability	Examples
Integers	There were 914 confirmed cases of COVID-19 reported yesterday.
Decimals	The reproduction number of COVID-19 is between 3.28 and 5.70.
Numerals	The total amount of transfer is fifty-two dollars and seven cents.
Separators	123,456.12 (En) → 123.456,12 (De)

Table 2: Tested capabilities of NMT systems in translating common types of numerical text.

corrupt *back-translation-based* training, as the resulting back-translated sentences are very likely to contain the desired errors.

2 Method

We follow Ribeiro et al. (2020)'s `CheckList` in designing our evaluation suite for NMT systems: we present several basic *capabilities* an NMT system should be expected to exhibit in translating common everyday numerical text; we then generate *test examples* specific to each capability to benchmark performance and find bugs in NMT systems.

2.1 Capabilities of Translating Numbers

We explore four capabilities (see Table 2), demonstrating expected translation ability of a system on common types of numerical text. Concretely, the *Integers* and *Decimals* represent basic capabilities; they can be manifested by testing on sequences of digits with variable lengths (*e.g.*, 100 vs. 10000) or decimals with the decimal mark placed at varying locations (1.001 vs. 10.01). We find that the tested NMT systems are more likely to malfunction when translating larger integers and decimals with longer fractional parts. The *Numerals* capability pertains to whether a system is able to translate numbers that are presented as words. The *Separators* capability checks if a model can deal with numbers containing decimal or thousands separators.[2] Systems that fail to manifest one or more of these capabilities may produce wrong numbers that can be inconspicuous to users and become a ready, exploitable source of misinformation.

2.2 Test Examples

To efficiently test the identified capabilities across multiple systems on distinct language pairs, we generate desired test examples using templates. For example, to test the *Numerals* capability, we use a template sentence such as "CNBC reported there were at least `[NUM]` cases worldwide.", where "`[NUM]`" is a number with the format "`ddd.ddn`", consisting of multiple digits and a numeral (*e.g.*, 100.01 million).

We experiment with formats of various lengths and decimal-point positions. We fill a format with random digits and numerals, and explore 25 different formats across all capabilities. This allows us to generate a diversity of numbers at scale, akin to fuzzing a program with random inputs to uncover bugs. We also note that all the numbers created for a format can be seen as a set of "adversarial" examples, as they are small perturbations of each other. Details about the test examples for each capability and the testing process can be found in Supplementary material.

3 Evaluation

Before presenting experimental results and discussion of our test framework, we first detail our evaluation setup.

Language pairs. We test both high-resource (HR) and low-resource (LR) scenarios. For HR, we consider two language pairs: English-German and English-Chinese, and for LR, we focus on English-Tamil and English-Nepali. We test both translation directions for each pair.

SOTA systems. We conduct behavioural testing against two popular commercial translation systems (denoted by **A** and **B**). As research systems, we use pre-trained models that were shown to perform well in WMT competitions (denoted by **R**), specifically, fairseq's transformer for English-German (Ng et al., 2019), English-Tamil (Chen et al., 2020), and English-Chinese/Nepali (Fomicheva et al., 2020).

The evaluation metric. For each capability we curate a list of test examples (sentences containing numbers), which are taken from various sources, including existing corpora or manually crafted (details in Supplementary material). To these sentences we remove the number component, and replace it with a number based on the specific capa-

---

[2] While *Decimals* and *Separators* may have overlapping instances (*e.g.*, the decimal mark), their specific formats in our testing are different (Table 2), which leads us to find non-overlapping error types: most *Decimals* errors involve translating numbers into wrong digits, whereas *Separators* errors pertain to mistranslation in localisation usage (*e.g.*, German and English use different decimal and thousand separators).

bility being tested. This test collection is then input to a translation system, and we report the *Pass Rate* (PR), the fraction of inputs where the system translates the numerical component perfectly.[3]

3.1 Testing Performance

Table 3 shows the results of testing the three SOTA systems across the HR/LR language pairs.

Among the four capabilities, *Numerals* turns out to be the most challenging across the systems tested, with the average $\overline{\text{PR}}$ of 70.8%. This is probably because, compared to other forms, numbers are less frequently written as words, resulting in insufficient examples available for training. At the other extreme, *Integers*, which tests on pure digits, is the easiest capability, as expected. Despite this, it is not a 'solved problem', given all systems report imperfect $\text{PR} < 100$ on at least one language.

Across the systems, the research system **R** ($\overline{\text{PR}}$: 77.8%) underperforms the two commercial ones ($\overline{\text{PR}}_A$: 80.6%, $\overline{\text{PR}}_B$: 90.4%). This is largely caused by the fact that the research system fails markedly on the En→Ne direction.

Per language, the results are similar in both translation directions, implying that numerical translation is a symmetric problem. Note that the results on LR are not always worse than that on HR (PRs on En-Ta are surprisingly the highest of all). This suggests that the size of training data is not the sole factor for high-quality numerical translation.

3.2 Error Analysis

We present analysis of novel types of mistranslations discovered from testing.

Decimal/thousands separators. We find that the decimal/thousands separators are prone to be mistranslated in localisation scenarios, when conventions differ between the languages (*e.g.*, "," and "." are the thousands and decimal separators in English while they are swapped in German). A common type of error is that a separator remains the same after translation (Table 4, row 1). This is probably due to the lack of sufficient training data to learn the translation of the separators in the target language.

Cardinal numerals. Cardinal numerals are commonly used in commercial and financial contexts. For example, the *financial characters* (*e.g.*, "壹" meaning one) are typical in Chinese financial documents. However, we find that the tested translation systems perform fairly poorly in translating cardinal numerals (Table 4, row 2). Common errors include mistranslation or under-translation of the unit words (*e.g.*, hundred) or the number words (*e.g.*, "陆拾"). Most often, the errors appear to be caused by the unique unit words used in different languages (*e.g.*, "万" in Chinese equals to 10 thousand), where a system needs to "compute" the correct amount for translation.

Digits. The pure digit translation (10→10) is expected to be easy, since a system may opt to copy the entire number as the translation. However, we find that the digit translation between English and low-resource languages can be far from satisfactory. An example is the translation between English and Nepali (Table 4, row 3). One reason for this result is that Nepali has its own numerals for digits (*e.g.*, १ denotes 1). As a result, a system would try to convert a digit into a Nepali digit (instead of keeping it unchanged) when translating numbers, which is difficult given limited training resources (Guzmán et al., 2019). Another common issue in digit translation is handling repeats of the same digit. A system is prone to omit or add one or more digits in the translation.

Units. This error often occurs when translating numbers accompanied by units of measurements (*e.g.*, 10 meters), especially when the target unit is unique to the language, *e.g.*, "角" in Chinese means "10 cents". In such cases (Table 4, last row), the system may need to learn the implicit conversion rules and then use them to "calculate" the correct numbers with the target unit of measurement. For example, when translating "10.01 million" into "1001 万" in Chinese, the system has to convert "10.01" into "1001" and then use the correct unit "万". An error may occur if the system fails either or both stages of this process (*i.e.*, mistranslating the numbers and/or units).

4 Potential Mitigation Strategies

Finally, we discuss several strategies that may mitigate the above errors discovered by our method[4].

---

[3] We count a Pass if the output matches the ground truth number, allowing for the use of digits (Arabic or local scripts) or numerals. For this purpose we use `num2words` (`https://pypi.org/project/num2words/`), `cn2an` (`https://github.com/Ailln/Cn2An.jl`), and locally developed scripts (for Ne and Ta).

[4] We leave validation of these strategies to future work.

Lang	Integers			Decimals			Numerals			Separators			Avg
	A	B	R	A	B	R	A	B	R	A	B	R	
En→Zh	100.0	100.0	94.0	100.0	100.0	92.2	77.5	72.5	67.5	100.0	100.0	91.4	91.2
Zh→En	94.0	78.0	90.0	100.0	100.0	93.8	82.0	78.0	56.7	100.0	100.0	83.3	88.0
En→De	100.0	100.0	100.0	93.8	78.1	68.8	87.5	67.5	95.0	83.3	80.0	80.0	86.2
De→En	100.0	100.0	100.0	98.4	79.7	95.3	87.0	84.0	65.7	97.1	68.5	65.7	86.8
En→Ta	100.0	98.0	100.0	100.0	100.0	100.0	100.0	97.5	100.0	100.0	100.0	100.0	99.6
Ta→En	98.0	100.0	100.0	98.4	100.0	98.4	96.0	100.0	90.0	100.0	88.6	100.0	97.4
En→Ne	18.0	-	70.0	17.2	-	72.0	7.5	-	65.0	16.7	-	60.0	**40.8**
Ne→En	98.0	-	88.0	96.9	-	1.6	46.7	-	5.3	80.0	-	0.0	**52.1**
Avg	88.5	96.0	92.8	88.1	92.9	77.8	**61.1**	83.2	**68.2**	84.6	89.5	**72.6**	-

Table 3: Test results (Pass Rate %) on the capabilities for numerical translation, with low averaged scores in bold. Nepali is not supported by System **B**.

Error Type	Sys/Lang	Test input sentence	System output	Ground-truth
Decimal/ Thousands separators	**A**, **B** En→De	The distance between Sydney to Washington is 9,756.001 miles.	Die Entfernung zwischen Sydney und Washington beträgt 9.756.001 Meilen.	9.756,001
	A, **B** En→De	The reproduction number of this disease is between 9.718 and 9.911.	Die Reproduktionszahl dieser Krankheit liegt zwischen 9.718 und 9.911.	9,718 and 9,911
	A, **B**, **R** De→En	Die Flugzeit von Punta Arenas beträgt etwa 85,619 Stunden.	The flight time from Punta Arenas is about 85,619 hours.	85.619
Cardinal numerals	**A**, **R** Zh→En	这支手表的总价是叁佰陆拾壹元。	The total price of this watch is [**A** three hundred and one yuan.] [**R** one dollar.]	361
	A, **B** En→Zh	The total amount of remittance is ninety thousand six hundred thirty-eight dollars and forty-seven cents.	汇款总额为 [**A** 九万六千三百八十八美元和][**B** 九千六百三十八元及] 四十七美分。(**En**: **A** [96388 dollars, 47 cents] **B** [9638 yuan, 47 cents])	90638
Digits	**A** En→Ne	There have been 670 confirmed cases of COVID19 to date.	अहिले सम्म COVID19 को 7070० पुष्टि भएका घटनाहरू छन्। (**En**: 70700)	670
	B, **R** En→Zh	An average of 1000009 people is infected every day due to this disease.	平均每天有 [**B** 100009] [**R** 10009] 人因这种疾病而感染。	1000009
	R Ne→En	UNESCO ले अनुमान गरेको छ कि ५१८८८९ शिक्षार्थीहरू सम्भावित जोखिममा छन् ।षा)	UNESCO estimates that 51889 teachers are at potential risk	518889
Units	**A** En→Zh	CNBC reported there were at least 100.01 million cases worldwide.	CNBC 报道，全球至少有 [**A** 1 亿 1001 万 **En**: 110.01 million] 例。[**B** 1.001 亿 **En**: 100.1 million)]	1 亿1万
	B Zh→En	这两根电线的长度分别是十米和五分米。(**En**: decimetres)	The length of the two wires is ten meters and five meters respectively.	decimetres
	R En→Zh	Case report forms were submitted to CDC for 7.415 million cases.	现已就74.15万宗案件向中心提交案件报告表。	741.5万

Table 4: Examples of four major types of errors discovered by our tests on three SOTA NMT systems.

Separate treatment of numbers. Although NMT models have been shown capable of performing basic arithmetic or bracket matching (Suzgun et al., 2019), this paper demonstrates that handling the various forms of numerical text in reality is still challenging. It may be worth separating numerical translation out into an individual process, as in Statistical MT (Koehn, 2009), that identifies numbers in the input, applies specific translation rules to them, and incorporates the translation into the output (Tu et al., 2012).

Data augmentation. Training with more quality data leads to better translation quality (Barrault et al., 2020). In our testing, we observe a large proportion of errors (*e.g.*, financial characters, units) stemming from mistranslation of specific numerals that are unique or used less frequently (*e.g.*,

"角", decimetres) in a language. Such errors could potentially be reduced if more *numeral-specific* instances were added to training.

Tailoring BPE segmentation. The Byte Pair Encoding (BPE) has been used by most leading NMT systems. However, long sequences of digits or numbers with separators (*e.g.*, ",", ".") are often split into varying sized fragments by BPE. This would render learning more difficult, as the system has to account for the dependency between the partitions. To circumvent this, one may wish to segment numbers differently, *e.g.*, to encode all numbers as character sequences, or as meaningful groupings of components (*e.g.*, segment into groups of 3 digits when processing English.)

Sanity checks. It is helpful to post-check whether all numbers in a translation are correct by comparing them to the inputs. This could be automated in the same way as we measure the Pass Rate (§3), and once again drawing parallels to software testing, could be fully automated via continuous integration of NMT systems.

5 Conclusion

In this paper, we propose an evaluation method to systematically assess four fundamental capabilities of NMT systems in translation numbers by virtue of a variety of test cases. Our tests reveal novel types of errors that are general across multiple SOTA translation systems for both high and low resource languages. We hope that our study will help improve numerical translation quality and reduce misinformation caused by numerical mistranslation.

Acknowledgements

We thank all anonymous reviewers for their constructive comments. The authors acknowledge funding support by Facebook.

Impact Statement

This work aims to improve the performance of NMT systems. The impact of poor numerical translation may go beyond poor user experience, potentially leading to financial loss, medical misinformation, and even a vector for poisoning NMT systems. This paper's behavioural testing could be used by an attacker to uncover flaws in a commercial NMT system. However, as in attack research in the security community, responsible highlighting of such flaws serves the purpose of improving systems: knowledge of systemic flaws in numerical translations helps vendors improve their systems to mitigate these effects in the first place, while concerted attackers are likely to discover vulnerabilities independently.

References

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federmann, Dmitriy Genzel, Franscisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, Rosie Lazar, Will Lewis, Graham Neubig, Mengmeng Niu, Alp Öktem, Eric Paquin, Grace Tang, and Sylwia Tur. 2020. TICO-19: the translation initiative for COvid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *ICLR*.

Peng-Jen Chen, Ann Lee, Changhan Wang, Naman Goyal, Angela Fan, Mary Williamson, and Jiatao Gu. 2020. Facebook AI's WMT20 news translation task submission. In *Proceedings of the Fifth Conference on Machine Translation*, pages 113–125, Online. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav

Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Paul Michel and Graham Neubig. 2018. MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553, Brussels, Belgium. Association for Computational Linguistics.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2019. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. *arXiv preprint arXiv:1907.05791*.

Lucia Specia, Zhenhao Li, Juan Pino, Vishrav Chaudhary, Francisco Guzmán, Graham Neubig, Nadir Durrani, Yonatan Belinkov, Philipp Koehn, Hassan Sajjad, Paul Michel, and Xian Li. 2020. Findings of the WMT 2020 shared task on machine translation robustness. In *Proceedings of the Fifth Conference on Machine Translation*, pages 76–91, Online. Association for Computational Linguistics.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

Mei Tu, Yu Zhou, and Chengqing Zong. 2012. A universal approach to translating numerical and time expressions. In *International Workshop on Spoken Language Translation (IWSLT) 2012*.

A Appendix

A.1 Testing Process

Our behaviour testing proceeds in four steps.

1) Test template selection: we select English sentences from quality real corpora TICO-19 (Anastasopoulos et al., 2020) and WikiMatrix (Schwenk et al., 2019). TICO-19 contains documents about COVID-19 (*e.g.*, scientific articles, conversations, Wikipedia entries) between English and 36 languages. WikiMatrix consists of parallel sentences extracted from Wikipedia articles in 85 languages. We randomly select five sentences with each containing numbers for the evaluation of each capability. We make each sentence a template by replacing the contained number with the placeholder "`[NUM]`".

2) Template filling-in: we fill the templates with randomly generated numbers in the digital format (*e.g.*, 1230000). Then, we convert the digital number into desired formats for testing (*e.g.*, 1,230,000 for a separator or 1.234 million for a numeral).

3) Translation: To test a system, we use it to translate all the test examples across all capabilities, and collate the translation results.

4) Evaluation: Finally, we check the correctness of the number translation by comparing the number to that in the input. We account for various forms of the number (*e.g.*, for the number 5, we also consider 5.0 and five as correct translations) so as to reduce the false positives. We also manually examine the incorrect translations to ensure they are not false positives.

A.2 Create (Transfer) Test Cases to a New Domain

Our testing framework facilitates constructing test instances for new domains in the following steps:

1. Obtain a large corpus of text that contains numbers (*e.g.*,CommonCrawl);
2. Check if there is a number in the output translation;
3. If so, then test if the output number is the correct "translation" for the number in the source sentence;
4. Use instances that pass this test as templates for switching in our different numbers.

Capability	Template	Count	Remarks
Integers e.g., 5	1. As of March 28, 2020, a total of [NUM] laboratory-confirmed COVID-19 cases (Figure) were reported to CDC. 2. Case report forms were submitted to CDC for [NUM] cases. 3. UNESCO estimates [NUM] learners are potentially at risk (pre-primary to upper-secondary education). 4. There have been [NUM] confirmed cases of COVID19 to date. 5. CNBC reported there were at least [NUM] cases worldwide.	50	Ranging from 1 to 10 digits
Decimals e.g., 7.14	1. An average of [NUM] people is infected every day due to this disease 2. The distance between Sydney to Washington is [NUM] miles 3. The genome size of the coronavirus is approximately [NUM] 4. At this point, Rosberg was about [NUM] seconds behind his teammate. 5. The reproduction number of this disease is between [NUM] and [NUM].	40	Ranging from 1 to 4 decimal position
Numerals e.g., five hundred	1. The total amount of remittance is [NUM]. 2. Case report forms were submitted to CDC for [NUM] cases. 3. As of 8 April 2020, approximately [NUM] learners have been affected due to school closures in response to COVID-19. 4. As of December 2019, [NUM] cases of MERS-CoV infection had been confirmed by laboratory tests 5. They then planned an ambitious open-air concert in Tokyo, with a stage costing [NUM] dollars US.	40	{hundred, thousand, million, trillion}
Separators e.g., 12,230	1. An average of [NUM] people is infected every day due to this disease 2. The distance between Sydney to Washington is [NUM] miles 3. The genome size of the coronavirus is approximately [NUM] 4. They then planned an ambitious open-air concert in Tokyo, with a stage costing [NUM] dollars US. 5. As of December 2019, [NUM] cases of MERS-CoV infection had been confirmed by laboratory tests	35	Ranging from 4 to 10 digits

Table 5: Summary of test examples used in our behavioural testing of NMT systems in translating numbers between English and {German, Chinese, Nepali, Tamil}.

A.3 Test Examples Details

Table 5 shows the details of the test examples used in our behavioural testing, including the templates used and the types of digits and numerals generated to fill in the templates.

Investigating Memorization of Conspiracy Theories in Text Generation

Note: This paper contains examples of potentially offensive conspiracy theory text.

Sharon Levy, Michael Saxon, William Yang Wang
University of California, Santa Barbara
`sharonlevy@cs.ucsb.edu, saxon@ucsb.edu, william@cs.ucsb.edu`

Abstract

The adoption of natural language generation (NLG) models can leave individuals vulnerable to the generation of harmful information memorized by the models, such as conspiracy theories. While previous studies examine conspiracy theories in the context of social media, they have not evaluated their presence in the new space of generative language models. In this work, we investigate the capability of language models to generate conspiracy theory text. Specifically, we aim to answer: can we test pretrained generative language models for the memorization and elicitation of conspiracy theories without access to the model's training data? We highlight the difficulties of this task and discuss it in the context of memorization, generalization, and hallucination. Utilizing a new dataset consisting of conspiracy theory topics and machine-generated conspiracy theories helps us discover that many conspiracy theories are deeply rooted in the pretrained language models. Our experiments demonstrate a relationship between model parameters such as size and temperature and their propensity to generate conspiracy theory text. These results indicate the need for a more thorough review of NLG applications before release and an in-depth discussion of the drawbacks of memorization in generative language models.

1 Introduction

Recent advances in natural language processing technologies have opened a new space for individuals to digest information. One of these rapidly developing technologies is neural natural language generation. These models, made up of millions, or even billions (Brown et al., 2020), of parameters, train on large-scale datasets. While attempts are made to ensure that only "safe" data is utilized for training these models, several studies have shown the prevalence of biases produced by these pretrained generation models (Sheng et al., 2019; Groenwold et al., 2020; Solaiman et al., 2019). Of equally alarming concern are the memorization and subsequent generation of factually incorrect data. Conspiracy theories are one particular type of this data that can be especially damaging.

While it is not new for researchers to learn that a model may memorize data (Radhakrishnan et al., 2019), we argue that the growing usage of machine learning models in society warrants targeted investigation to deter potential harms from problematic data. In this paper, we address the upsides and pitfalls of memorization in generative language models and its relationship with conspiracy theories. We further describe the difficulty of detecting this memorization for the categories of memorization, generalization, and hallucination. Previous studies investigating memorization of text generation models have done so with access to the model's training data (Carlini et al., 2019, 2020). As models are not always published with their training datasets, we set out to examine the difficult task of eliciting memorized conspiracy theories from a pretrained NLG model through various model settings **without access to the model's training data**.

We focus our study on the pre-trained GPT-2 language model (Radford et al., 2019). We investigate this model's propensity to generate conspiratorial text, analyze relationships between model settings and conspiracy theory generation, and determine how these settings affect the linguistic aspect of generations. To do so, we create a new conspiracy theory dataset consisting of conspiracy theory topics and machine-generated conspiracy theories.

Our contributions include:

- We propose the topic of conspiracy theory memorization in pretrained generative language models and outline the harms and benefits of different types of generations in these

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4718–4729
August 1–6, 2021. ©2021 Association for Computational Linguistics

models.

- We analyze pretrained language models for the inclusion of conspiracy theories without access to the model's training data.
- We evaluate the linguistic differences for generated conspiracy theories across different model settings.
- We create a new dataset consisting of conspiracy theory topics from Wikipedia and machine-generated conspiracy theory statements from GPT-2.

2 Spread of Conspiracy Theories

2.1 Dangers of conspiracy theories

A conspiracy theory is the belief, contrary to a more probable explanation, that the true account for an event or situation is concealed from the public (Goertzel, 1994). A variety of conspiracy theories ranging from the science-related moon landing hoax (Bizony, 2009) to the racist and pernicious Holocaust denialism[1] are widely known throughout the world. However, even as existing conspiracy theories continue circulating, new conspiracy theories are consistently spreading. This is especially concerning given that half of Americans believe at least one conspiracy theory (Oliver and Wood, 2014).

Widespread belief in conspiracy theories can be highly detrimental to society, driving prejudice (Douglas et al., 2019), inciting violence[2], and reducing science acceptance (van der Linden, 2015; Lewandowsky et al., 2013). Science denial has real-world consequences, such as resistance to measures for the reduction of carbon footprints (Douglas and Sutton, 2015) and outbreaks of preventable illnesses due to reduced vaccination rates (Goertzel, 2010). Further effects of conspiracy theory exposure can reach the political space and reduce citizens' likelihood of voting in elections due to feelings of powerlessness towards the government (Jolley and Douglas, 2014b).

At the time of writing, the COVID-19 pandemic is at its worst. Though COVID-19 vaccines have received approval and started distribution, new conspiracy theories surrounding the COVID-19 vaccine may hinder society in its road to recovery. Discussions of a link between vaccinations and autism have been circulating for years (Jolley and Douglas, 2014a; Kata, 2010). However, with the extreme interest throughout the world surrounding the COVID-19 pandemic, new vaccination rumors are arising, such as the vaccine causing DNA alteration and claims of the pandemic acting as a cover plan to implant trackable microchips[3]. The belief in these theories can prevent herd immunity through the lack of vaccinations[4] [5].

2.2 NLG spreading conspiracy theories

As NLG models are being utilized for various tasks such as chatbots and recommendations systems (Gatt and Krahmer, 2018), cases arise in which these conspiracy theories and other biases can propagate unintentionally (Bender et al., 2021). We present one such scenario in which an NLG model has memorized some conspiracy theories and is being used for story generation (Fan et al., 2018). An unaware individual may utilize this application and, given a prompt about the Holocaust, may receive a generated story discussing Holocaust denial. The user, now having been exposed to a new conspiracy theory, may choose to ignore this generated text at this stage. However, a potential negative outcome is that the user may become interested in this story and search the statements online out of curiosity. This can lead the user down the "rabbit hole" of conspiracy theories online (O'Callaghan et al., 2015) and alter their original assumptions towards believing this conspiracy theory.

2.3 Why are conspiracy theories difficult to detect?

Recent years have seen the emergence of several new tasks addressing fairness and safety within natural language processing in topics such as gender bias and hate speech detection. Although detection and mitigation of other biases and harmful content have been thoroughly studied, that pertaining to conspiracy theories is increasingly difficult due to its inconsistent linguistic nature.

Many existing tasks can utilize specific keyword lists such as Hatebase[6] for detection in addition to

---

[1] http://auschwitz.org/en/history/holocaust-denial/

[2] https://www.theguardian.com/us-news/2019/aug/01/conspiracy-theories-fbi-qanon-extremism

[3] https://www.bbc.com/news/54893437

[4] https://www.economist.com/graphic-detail/2020/08/29/conspiracy-theories-about-covid-19-vaccines-may-prevent-herd-immunity

[5] https://www.who.int/news-room/q-a-detail/herd-immunity-lockdowns-and-covid-19

[6] https://hatebase.org/

current techniques (Sun et al., 2019). However, conspiracy theory detection is an increasingly complex problem and cannot be approached in the same way as the previous topics. Conspiracy theories have no unified vocabulary or keyword list that can differentiate them from standard text. Previous studies of conspiracy theories have exhibited their tendency to lean towards issues of hierarchy and abuses of power (Klein et al., 2019). We argue this is not specific enough to define features for their detection. Often, specific keywords and tropes become typical of conspiracy theories regarding a specific topic, such as 9/11 and "false-flag" (Knight, 2008). However, as the number of topics surrounding conspiracy theories grows, it becomes infeasible to create and maintain these topic-specific vocabularies. To add to this difficulty, while humans can typically detect other types of biases, they cannot easily distinguish conspiracy theories from truthful text by merely reading the statement. Doing so typically requires knowledge of the topic itself or a more in-depth look into the theory narrative through network analysis[7]. To this end, the best way to stop the spread of conspiracy theories is not in late-stage detection but early intervention.

2.4 How can NLG models be misused?

While the generation of conspiracy theories may be an accidental outcome by NLG models, the possibility still exists that adversaries will intentionally utilize these language models to spread these theories and cause harm. In one such case, propagandists may utilize NLG models to reduce their workload when spreading influence (McGuffie and Newhouse, 2020). By merely providing topic-specific prompts, they can utilize these models to easily and efficiently produce a variety of conspiratorial text for online communities regarding the topics. As a result, these communities will appear to be larger than their actual size and provide the appearance that belief in the issue is high. This may provide real-life members with a sense of belonging and subsequently reinforce belief in the theories or even recruit new members (Douglas et al., 2017).

3 Memorization vs. Generalization vs. Hallucination

The memorization of data in the context of machine learning models has been highlighted in research for many years now. Related work has researched the types of information models memorize (Feldman and Zhang, 2020), how to increase generalization (Chatterjee, 2018), and the ability to extract information from these models (Carlini et al., 2020). While memorization is typically discussed in the space of memorization vs. generalization, we believe this can be broken down even further. In the context of conspiracy theories, we establish three types of generations:

- Memorized: generated conspiracy theories with exact matches existing within the training data.
- Generalized: generations that do not have exact matches in the data but produce text that follows the same ideas as those in the training data.
- Hallucinated: generations about topics that are neither factually correct nor follow any of the existing conspiracy theories surrounding the topic.

Studies on memorization tend to focus on either memorization vs. generalization or memorization vs. hallucination (Nie et al., 2019). In the latter case, it is easy to see how the term "memorization" can apply to the first two categories. Ideally, in an NLG model, we would hope for generations to be generalized since direct memorization can have the downsides of generating sensitive information (Carlini et al., 2019). There are also cases when hallucinations are ideal, such as in the realm of creative story-telling. Should we be able to distinguish among these categories, we could gain deeper insight into what and how these models learn during training. However, we acknowledge that classifying generations based on these categories is a difficult problem and believe this should be a task for future research in memorization.

Our focus in this paper is to evaluate 1) whether a model has memorized conspiracy theories during training and 2) the propensity for the model to generate this information among different model settings (as opposed to generating other memorized or hallucinated information about a topic). Evaluating memorization within a model can be done in two settings: with training data as a reference or without training data. Previous studies have evaluated memorization within machine learning models by utilizing the model's training dataset. However,

---

[7] https://theconversation.com/an-ai-tool-can-distinguish-between-a-conspiracy-theory-and-a-true-conspiracy-it-comes-down-to-how-easily-the-story-falls-apart-146282

the reality is that many models nowadays are not published alongside their training data (Kannan et al., 2016). In this case, the evaluation becomes increasingly difficult, as there is nothing to match a model's output to. In order to simulate a real-world environment, we analyze the second setting of investigating memorization without access to training data and instead treat the model as a black box when evaluating its outputs. Due to the difficulty of distinguishing among the three categories of memorization, generalization, and hallucination, we follow previous work and refer to both memorized and generalized generations as memorized samples for the rest of the paper.

4 When is memorization a good thing?

While we focus most of this paper on the downsides of memorization in natural language generation models, it is still important to address the benefits. There are several situations in which memorized information may be utilized, such as in dialogue generation (Gu et al., 2016). When used in the chatbot setting, a model may be asked questions on real-world knowledge. Assuming the model has learned correct factual information, this memorization can prove useful. Furthermore, conspiracy theories are a part of language and culture. It is not inherently bad that a model is aware of the existence or concept of conspiracy theories, particularly in cases where models may be deployed as an intervention in response to human-written conspiratorial text. This only becomes harmful when the model cannot recognize text as a conspiracy theory and generates text from the viewpoint of the conspiracy being true. Though memorization may aid in the described cases, the downside of the learned conspiracy theories (as factual statements) and other information such as societal biases can outweigh these benefits.

5 Data Collection

While conspiracy theory data may appear in misinformation datasets labeled as "Fake News" with other misinformation types, there are few existing datasets with conspiracy theory labeled text. Previous conspiracy theory studies contain datasets that are either small in size (Oliver and Wood, 2014), contain non-English data (Bessi et al., 2015), or pertain to events occurring after the release of GPT-2 (Ahmed et al., 2020; Uscinski et al., 2020). Therefore, we create a dataset exclusively dedicated to conspiracy theories [8]. We obtain our data for our analysis from two different sources: Wikipedia and GPT-2. We show samples from each of our datasets in Table 1.

	Conspiracy Theory
Wikipedia	The **Holocaust** is a lie, and the Jews are not the victims of the Nazis.
GPT-2	The US government is secretly running a secret program to create a super-soldier that can kill and escape from any prison.

Table 1: Samples from the Wikipedia dataset consisting of Wikipedia topics and General dataset of GPT-2 generated conspiracy theories without topic prompts. The Wikipedia topic is highlighted in bold and is used as a topic-prompt for text generation in GPT-2.

5.1 Wikipedia

We first aim to create a set of conspiracy theory topics. To gather this data, we utilize Wikipedia's category page feature. Each item listed in a category page is linked to a corresponding Wikipedia page. We obtain the page headers in the conspiracy theory category page and the following page headers in the Wikipedia conspiracy theory category tree. This process allows us to extract 257 Wikipedia pages regarding conspiracy topics. We further refine this dataset of conspiracy topics through the use of Amazon Mechanical Turk (Buhrmester et al., 2016). Ten workers are assigned to each Wikipedia conspiracy topic, and each worker is asked whether they have heard of a conspiracy theory related to the topic. We remove any topic from our dataset with fewer than six votes to focus our study on the well-known conspiracy theory topics that a model would be more likely to be prompted with. Our final dataset consists of the following seventeen conspiracy theory topics: Death of Marilyn Monroe, Men in black, Sandy Hook school shooting, UFO's, Satanic ritual abuse, Climate change, Area 51, 9/11, Vast right-wing conspiracy, Global warming, Shadow government, Holocaust, Flat Earth, Illuminati, Pearl Harbor, Moon landing, and John F. Kennedy assassination. We refer to this as the Wikipedia dataset for the remainder of the paper.

---

[8] https://github.com/sharonlevy/Conspiracy-Theory-Memorization

5.2 GPT-2

We create a second dataset consisting of machine-generated conspiracy theories. To do this, we elicit the conspiracy theories directly from GPT-2 Large with the HuggingFace transformers library (Wolf et al., 2020). We prompt GPT-2 with "The conspiracy theory is that" at varying temperature levels (0.4, 0.7, 1). We obtain 5000 theories at each temperature level and post-process the text by removing the original prompt and keeping only the first sentence. For the remainder of the paper, we refer to this dataset as the General dataset.

6 Generation of Conspiracy Theories

An intriguing question in the scope of conspiracy theory generation is: what can trigger a language model to generate conspiracy theories? We begin by investigating the effects of model parameters and decoding strategies on the generation of conspiracies when prompted with a topic. Of these, we study model temperature and model size.

We use our Wikipedia dataset to create a generic prompt as input to GPT-2, such as "The Holocaust is". In order to remove any trigger such as "Flat Earth is", we modify some of our topic titles during prompt creation to make a more neutral prompt. In the case of "Flat Earth", our prompt is "The Earth is", so that the model is not intentionally triggered to produce Flat Earth conspiracy text. We perform this action for the rest of our topics as well. For each prompt, we employ the model to create twenty generations with a token length of fifty.

When evaluating the generated text, we evaluate whether or not the text affirms the conspiracy theory. In this sense, we count "The Earth is flat" as affirming the conspiracy theory and "The Earth is flat is a conspiracy theory" as not affirming the theory. As such, we evaluate whether the model presents the theory as factual belief as opposed to whether it has knowledge of the theory.

To determine whether or not the generation affirms a known conspiracy theory, we utilize Amazon Mechanical Turk. Worker pay and instructions are detailed in Appendix A. We provide each worker with a reference passage describing known conspiracy theories for each topic and ask whether or not the generation affirms or aligns with the reference text. We make sure to state that the reference text contains several conspiracy theories about the topic at the top of each HIT. In this case, if a worker is exposed to new text, they are clearly informed that the text is a conspiracy theory. Should a worker encounter these theories in the future, they may even benefit from the task since they are now armed with the knowledge that these statements are in fact conspiracy theories. Seven workers are assigned to each generated sequence. If the text is voted as a conspiracy, it receives a point; otherwise, it is subtracted a point. We then retrieve those generations with two or more points (indicating a general consensus) and manually evaluate this subset of generations for another round of verification.

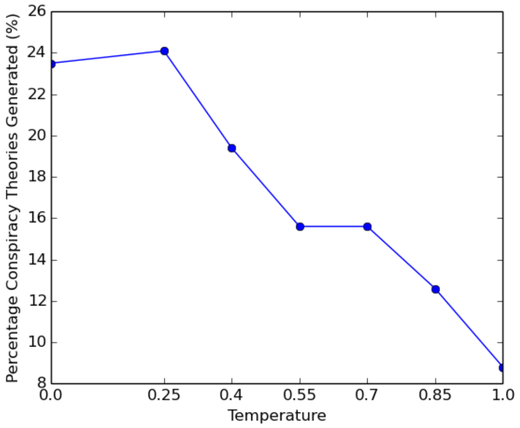

Figure 1: Percentage of conspiracy theories generated by GPT-2 Large at varying temperatures when prompted on 17 different conspiracy theory topics. Each topic is used to generate 20 sequences for a total of 340 generations.

6.1 Temperature

We first evaluate GPT-2 Large at temperature settings ranging from 0.25 to 1 with sampling, where 1 is the default setting for the model, and with greedy decoding on the Wikipedia dataset prompts. This decoding strategy changes the model's probability distribution for predicting the next word in the sequence. A lower temperature will increase the likelihood of high probability words and decrease the likelihood of low probability words. At each temperature level, we compute the percentage of generated text marked as conspiracy theories out of the total number of generations. We share our results in Figure 1.

It can be seen that as the temperature decreases, the model follows a general trend of generating more conspiracy theories. There is an exception when temperature $\rightarrow$ 0, which translates to simple greedy decoding. In this case, the proportion of conspiracy theories decreases slightly, indicating

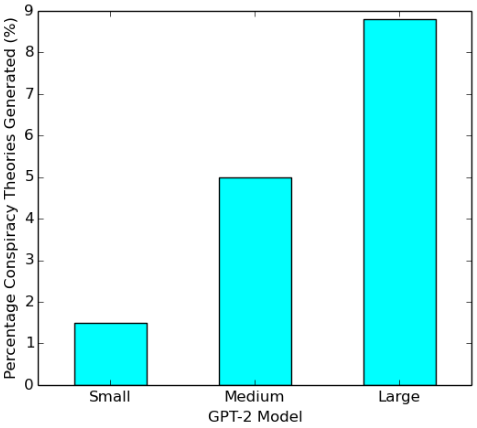

Figure 2: Percentage of conspiracy theories generated by GPT-2 models of size small, medium, and large when prompted on 17 different conspiracy theory topics. Each topic is used to generate 20 sequences for a total of 340 generations.

that while the model may memorize some theories, other information for specific topics is also memorized and have a higher likelihood of being generated. However, the general result curve shows that existing conspiracy theories are deeply rooted in the model during training for many topics. Given these findings, we believe it is best to add randomization to the decoding procedure, at the risk of quality and coherency, instead of greedy search in order to minimize the risk of generating deeply memorized conspiracy theories.

Decreasing the model's temperature allows us to evaluate which topics this deep memorization may be true for, as not every conspiracy topic may be ingrained in the model. We assess which topics the model increases its number of conspiracy theory generations for at a lower temperature. When parsing the previous results for each topic across the different temperature settings, we find this increase in conspiracy theory generations and, therefore, the prominent memorization of conspiracy theories for the topics of UFO's, 9/11, Holocaust, Flat Earth, Illuminati, and Moon landing.

6.2 Model size

Next, we aim to test a language model's size for its capability to memorize and generate conspiracy theories. Again, we utilize the Wikipedia dataset prompts for generations. We prompt three model sizes with our topics: GPT-2 Small (117M parameters), GPT-2 Medium (345M parameters), and GPT-2 Large (762M parameters). We keep a fixed temperature across the models and set it at the default value of 1. We use the same evaluation technique described above and compute the proportion of generations marked as conspiracy theories out of the total number of generations. These results are shown in Figure 2.

While nearly 10% of GPT-2 Large's generations are classified as conspiracy theories, GPT-2 Medium reduces this number by almost 50%. The GPT-2 Small model's conspiracy theory generations are substantially lower than this at a little over 1%. We can deduce that reducing model size vastly lowers a model's capacity to retain and memorize information after training, even if that information is profoundly prominent within the training data. Not only is this beneficial for mitigating the generation of conspiracy theories, but it can also allow the model to generalize better to other information for topic-specific prompts.

7 Towards Automated Evaluation

As we have shown that varying temperature and model size can individually lead to further elicitation and memorization of conspiracy theories, we now investigate the effects of varying the two together. In our previous experiments, we utilize Mechanical Turk to identify conspiracy theories among the generated text. However, we understand that human evaluation is not feasible for detecting conspiracy theories on a large scale. Instead, we desire to advance towards a more automated evaluation of memorization. As such, we investigate whether we can define a relationship between the memorization of conspiracy theories and perplexity across the different model parameters.

Following previous studies on fact-checking (Chakrabarty et al., 2018; Wang and McKeown, 2010) and model memorization (Carlini et al., 2020), we evaluate model generations against Google search results. This time, we utilize our General dataset, made up of conspiracy theories generated with the generic prompt "The conspiracy theory is that". We query Google with a generated conspiracy theory at each temperature setting and compare this theory to the first page of results. We did not manually use Google search for our generated text and instead created a script to automate this and scrape the text from the first page of results. We provided the minimum amount of information needed for making each search request so that this does not include search history and the

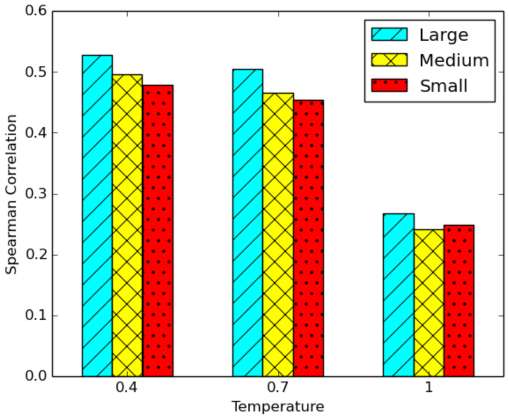

Figure 3: Spearman correlation of model perplexity vs. Google search BLEU score for GPT-2 generated conspiracy theories across varying temperature settings. Each generated theory is evaluated against the first page of Google search results with the BLEU metric.

more specific location information or cookies.

The temperature values of 0.4, 0.7, and 1 are used as lower temperature values start to produce many duplicate generations and lead to small sample sizes for this evaluation. We obtain the text snippet under each search result and evaluate this against the conspiracy theory with the BLEU metric (Papineni et al., 2002). The BLEU metric is utilized since many search results do not contain complete sentences and are instead highlighted phrases from the text related to the query and concatenated by ellipses. The perplexity score for a conspiracy theory is then calculated for each model size. The resulting BLEU and perplexity scores are ranked with the highest BLEU and lowest perplexity scores first. We use Spearman's ranking correlation (Hogg et al., 2005) to determine the resulting alignment between the two. These results are shown in Figure 3.

We find a strong relationship between a generated conspiracy theory's perplexity and its appearance in Google search results. This correlation becomes much weaker when the temperature is set to 1, indicating that the default setting's increased randomness may produce more hallucinated generations. However, given these results, we believe this can open the door towards the creation of more automated memorization evaluation techniques. Though our samples are generated through GPT-2 Large, we further test this alignment on the small and medium model sizes. We find that the relationship between Google search results and perplexity decreases as model size decreases for the smaller temperature settings, further confirming that model size does affect memorization.

	Temperature			
Classifier	0.4	0.7	1.0	p-val
dBERT	-0.974	-0.942	-0.887	0.110
VADER	-0.556	-0.527	-0.486	<0.001
TextBlob	-0.112	-0.033	0.017	<0.001
Average	-0.547	-0.500	-0.452	

Table 2: Comparison of average sentiment scores across GPT-2 Large generated conspiracy theories with the DistilBERT (dBERT), VADER, and TextBlob sentiment classifiers along with the Wilcoxon rank-sum p-values for generation pairs of temperature 0.4 and 1. The conspiracy theories are generated at the temperature values of 0.4, 0.7, and 1.0 and sentiment scores range from -1 to 1.

8 Linguistic Analysis

While our previous analysis aims to define a relationship between model parameters and the generation of conspiracy theories, we are also interested in evaluating whether these generations have any interesting linguistic properties. As such, we choose to test the question, are there any linguistic differences among the generated conspiracy theories across different model settings? We proceed by examining two linguistic aspects of our texts: sentiment and diversity.

8.1 Sentiment

When analyzing sentiment, we evaluate our General dataset of generated conspiracy theories at its three temperature levels. We are interested in answering the question: how will the model's temperature affect the sentiment of its generations that are not prompted by real-world stimulus? To proceed, we utilize three sentiment classifiers: DistilBERT (Sanh et al., 2019), VADER (Gilbert and Hutto, 2014), and TextBlob[9]. For DistilBERT we convert the output range of [0,1] to [-1,1] to match the other two classifier ranges. The average sentiment scores are displayed in Table 2 along with the Wilcoxon rank-sum p-values for each classifier output between temperature settings 0.4 and 1. The results show that decreasing the model's temperature triggers it to generate increasingly negative conspiracy theories. Although we do not achieve

---

[9] https://textblob.readthedocs.io/en/dev/index.html

	Temperature		
Size	0.4	0.7	1.0
Small	0.372	0.227	0.084
Medium	0.397	0.231	0.094
Large	0.421	0.255	0.120
p-value	<0.001	<0.001	<0.001

Table 3: Comparison of average BERTScore values across Wikipedia topic-prompted GPT-2 generations for varying model sizes and temperatures. Generations for each size-temperature pair are evaluated against other generations for their specific topic. Wilcoxon rank-sum p-values for the large-small model pairs at each temperature are listed at the bottom.

similar sentiment scores across the different classifiers, they all exhibit the same downward trend among score and temperature values. Additionally, classifier-temperature value pairs produce negative sentiment scores in all but one case. This follows previous work indicating that conspiracy theories and one's belief in them are emotional rather than analytical and are linked to negative emotions (van Prooijen and Douglas, 2018).

8.2 Diversity

Next, we analyze linguistic diversity across model sizes and model temperature. Utilizing the Wikipedia dataset, we compute the BERTScore (Zhang* et al., 2020) for each generation in reference to the other generations for each topic. This metric is used to measure the variance and contextual diversity across the different model generations for a specific conspiracy topic (Zhu et al., 2020). We do this across temperature values of 0.4, 0.7, and 1 and the different model sizes. These temperature values are utilized as lower temperature values start to produce duplicate generations. The average F1 scores for each setting pair is calculated and shown in Table 3 along with the corresponding p-values from a Wilcoxon rank-sum test for the large-small pairs at each temperature.

We find that as the temperature decreases, the similarity across generations for each topic increases. This is not surprising, as the outputs become less random at lower temperatures, and the model tends to output more memorized information. When comparing the scores among the different model sizes, the largest model contains the largest values, decreasing with the model size. We can infer that an increase in model size leads to more memorization, which allows the model to generate more contextually aligned outputs for specific topics instead of the diverse sets of outputs in smaller model sizes.

9 Moving Forward

Throughout this paper, we have discussed the risks and benefits of memorization in NLG models and have focused on the dangers of conspiracy theory generation. As we relayed in Section 2.3, conspiracy theory detection is a challenging problem due to its fuzzy linguistic vocabulary. We believe it is crucial to intervene earlier to mitigate these risks rather than detect them after the model's generation. While reducing memorization of harmful data in models is still an open problem, we discuss various methods to help accomplish this and encourage future research in the area: 1) preventing detrimental data from being introduced into the training set, 2) ensuring the dataset contains a much larger proportion of factually correct data for conspiracy theory topics than the conspiracy theories themselves, and 3) reducing model size.

The first solution prevents researchers from relying on these models to filter out harmful noise in large-scale datasets. Current models, such as GPT-2, attempt to filter out offensive and sexually explicit content from their datasets during creation[10]. We argue that this is not enough, as shown in the results of our analysis above. One way to proceed is to ensure that data is only collected from reliable sources instead of scraping the internet for large amounts of information. However, we also recognize that this is a tedious task and requires intensive scrutiny when collecting data. As such, the downsides to following this method may lead to smaller datasets and models with lower quality generations. In addition, this requires the additional consideration of deciding what data is "good" and what data can be harmful. In the space of conspiracy theories, the creation of a database regarding circulated conspiracy theories and debunking them seems like an appropriate direction to go.

While not completely eliminating the possibility of conspiracy theory generation, the second method aims to decrease their likelihood during generation. To accomplish this, researchers can supplement their existing dataset with a second dataset consisting of factually correct samples sur-

---

[10] https://github.com/openai/gpt-2/blob/master/modelcard.md

rounding conspiracy theory topics. This aims to oversample truthful data for training. While our study is confined to well-known conspiracy theories, the approach we discuss should be performed for all conspiracy-related topics and thus requires the additional task of identifying these subjects.

As our experiments in Section 6 have shown, model temperature and size profoundly affect the memorization and generation of conspiracy theories in NLG models. Since a user may set temperature, this setting cannot help prevent the generation of harmful data. However, modifying model size can. Though recent years have seen an increase in model size due to better performance on downstream tasks and the resulting generation of more coherent text (Solaiman et al., 2019), it comes at the cost of memorization. Therefore, researchers must strive to find a balance between memorization and fluency. When compromising model size, this mitigation strategy may also be complemented by oversampling factual data as specified above for further intervention.

10 Conclusion

In this paper, we highlight the issue of conspiracy theory memorization and generation in pretrained generative language models. We show that the root of the problem stems from the memorization of these theories by NLG models and discuss the dangers that may follow this. This paper further investigates the detection of conspiracy theory memorization in these models in a real-world scenario where one does not have access to the training data. To do so, we create a conspiracy theory dataset consisting of conspiracy theory topics and machine-generated text. Our experiments show that reducing a model's temperature and increasing its size allows us to elicit more conspiracy theories, indicating their memorization without verification against the ground-truth dataset. We hope our findings encourage researchers to take additional steps in testing language models for the generation of harmful content before release. Further, we hope our discussion on memorization can lead to further research in the area and advance the study of conspiracy theories in NLP.

Acknowledgements

We would like to thank Amazon AWS Machine Learning Research Award and Amazon Alexa Knowledge for their generous support. This work was also supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. 1650114.

Ethical Considerations

In this paper, we explore the topic of conspiracy theories in natural language generation. We acknowledge that in order to build a coherent and robust language model, a large-scale dataset must be used. As such, it is difficult to obtain data of this size that is 100% free of offensive or harmful samples. However, to improve and further progress research in natural language processing, considerations of disadvantages such as the memorization and subsequent generation of conspiracy theories must be taken into account.

In the previous sections of the paper, we feature how researchers can evaluate language models for the memorization and generation of conspiracy theory text. While we showcase methods for analyzing the generation of this harmful content, we acknowledge some potential risks: 1) adversaries may adjust their methods for hiding harmful content in language models so that it is not easily extracted or generated through our evaluation methods, and 2) attackers may use the methods discussed and leverage other language models to extract conspiracy text or amplify their generation in natural language applications. However, we believe bringing light to the issue of conspiracy theory memorization in NLG models is essential for research to progress in the direction of safe and fair natural language processing and will enable future research to utilize these studies in model interpretability.

While it may be possible for attackers to extract conspiracy theories from language models through more advanced techniques, we attempt to study how the ordinary user may fall prey to this type of information in a standard way. With language models becoming increasingly integrated into everyday natural language processing applications, the risks of unintentionally generating and spreading conspiracy theories rises. Our work can help researchers and engineers of language models thoroughly test these models for the generation of harmful conspiracy theory text using our analysis. We also hope to show researchers that even seemingly "clean" datasets may not diminish harmful noise from data and instead reflect or even amplify it after training.

References

Wasim Ahmed, Josep Vidal-Alaball, Joseph Downing, and Francesc López Seguí. 2020. COVID-19 and the 5G conspiracy theory: social network analysis of Twitter data. *Journal of Medical Internet Research*, 22(5):e19458.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big.

Alessandro Bessi, Mauro Coletto, George Alexandru Davidescu, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. 2015. Science vs conspiracy: Collective narratives in the age of misinformation. *PloS one*, 10(2):e0118093.

Piers Bizony. 2009. It was all a fake, right? *Engineering & Technology*, 4(12):24–25.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. 2016. Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality data?

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 267–284.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2020. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*.

Tuhin Chakrabarty, Tariq Alhindi, and Smaranda Muresan. 2018. Robust Document Retrieval and Individual Evidence Modeling for Fact Extraction and Verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 127–131, Brussels, Belgium. Association for Computational Linguistics.

Satrajit Chatterjee. 2018. Learning and memorization. In *International Conference on Machine Learning*, pages 755–763. PMLR.

Karen M. Douglas and Robbie M. Sutton. 2015. Climate change: Why the conspiracy theories are dangerous. *Bulletin of the Atomic Scientists*, 71(2):98–106.

Karen M Douglas, Robbie M Sutton, and Aleksandra Cichocka. 2017. The psychology of conspiracy theories. *Current directions in psychological science*, 26(6):538–542.

Karen M Douglas, Joseph E Uscinski, Robbie M Sutton, Aleksandra Cichocka, Turkay Nefes, Chee Siang Ang, and Farzin Deravi. 2019. Understanding conspiracy theories. *Political Psychology*, 40:3–35.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

CHE Gilbert and Erric Hutto. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*, volume 81, page 82.

Ted Goertzel. 1994. Belief in conspiracy theories. *Political psychology*, pages 731–742.

Ted Goertzel. 2010. Conspiracy theories in science: Conspiracy theories that target specific research can have serious consequences for public health and environmental policies. *EMBO reports*, 11(7):493–499.

Sophie Groenwold, Lily Ou, Aesha Parekh, Samhita Honnavalli, Sharon Levy, Diba Mirza, and William Yang Wang. 2020. Investigating African-American Vernacular English in transformer-based text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5877–5883, Online. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P. Bigham. 2018. A data-driven analysis of workers' earnings on Amazon Mechanical Turk. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–14, New York, NY, USA. Association for Computing Machinery.

Paul Hitlin. 2016. Research in the crowdsourcing age: A case study.

Robert V Hogg, Joseph McKean, and Allen T Craig. 2005. *Introduction to mathematical statistics*. Pearson Education.

Daniel Jolley and Karen M Douglas. 2014a. The effects of anti-vaccine conspiracy theories on vaccination intentions. *PloS one*, 9(2):e89177.

Daniel Jolley and Karen M Douglas. 2014b. The social consequences of conspiracism: Exposure to conspiracy theories decreases intentions to engage in politics and to reduce one's carbon footprint. *British Journal of Psychology*, 105(1):35–56.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufman, Balint Miklos, Greg Corrado, Andrew Tomkins, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (2016)*.

Anna Kata. 2010. A postmodern Pandora's box: anti-vaccination misinformation on the internet. *Vaccine*, 28(7):1709–1716.

Colin Klein, Peter Clutton, and Adam G Dunn. 2019. Pathways to conspiracy: The social and linguistic precursors of involvement in Reddit's conspiracy theory forum. *PloS one*, 14(11):e0225098.

Peter Knight. 2008. Outrageous conspiracy theories: Popular and official responses to 9/11 in Germany and the United States. *New German Critique*, (103):165–193.

Stephan Lewandowsky, Klaus Oberauer, and Gilles E Gignac. 2013. NASA faked the moon landing—therefore, (climate) science is a hoax: An anatomy of the motivated rejection of science. *Psychological science*, 24(5):622–633.

Kris McGuffie and Alex Newhouse. 2020. The radicalization risks of GPT-3 and advanced neural language models. *arXiv preprint arXiv:2009.06807*.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679, Florence, Italy. Association for Computational Linguistics.

J Eric Oliver and Thomas J Wood. 2014. Conspiracy theories and the paranoid style (s) of mass opinion. *American Journal of Political Science*, 58(4):952–966.

Derek O'Callaghan, Derek Greene, Maura Conway, Joe Carthy, and Pádraig Cunningham. 2015. Down the (white) rabbit hole: The extreme right and online recommender systems. *Social Science Computer Review*, 33(4):459–478.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jan-Willem van Prooijen and Karen M Douglas. 2018. Belief in conspiracy theories: Basic principles of an emerging research domain. *European journal of social psychology*, 48(7):897–908.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

A Radhakrishnan, KD Yang, M Belkin, and C Uhler. 2019. Memorization in overparameterized autoencoders. In *Deep Phenomena Workshop, International Conference on Machine Learning*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3407–3412, Hong Kong, China. Association for Computational Linguistics.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. *arXiv preprint arXiv:1906.08976*.

Joseph E Uscinski, Adam M Enders, Casey Klofstad, Michelle Seelig, John Funchion, Caleb Everett, Stefan Wuchty, Kamal Premaratne, and Manohar Murthi. 2020. Why do people believe COVID-19 conspiracy theories? *Harvard Kennedy School Misinformation Review*, 1(3).

Sander van der Linden. 2015. The conspiracy-effect: Exposure to conspiracy theories (about global warming) decreases pro-social behavior and science acceptance. *Personality and Individual Differences*, 87:171 – 173.

William Yang Wang and K. McKeown. 2010. ”Got You!”: Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-Semantic Modeling. In *COLING*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Wanrong Zhu, Xin Eric Wang, Pradyumna Narayana, Kazoo Sone, Sugato Basu, and William Yang Wang. 2020. Towards understanding sample variance in visually grounded language generation: Evaluations and observations.

A Appendix

A.1 Mechanical Turk Experiments

Our conspiracy theory affirmation task was grouped into batches of 5 statements per topic and workers were compensated at 22¢ per batch. For the conspiracy topic task, we provided a topic, e.g. Holocaust, and ask the worker to click “yes” if they have heard of a conspiracy theory related to the topic. This simple task was paid at 2¢ per topic. The first task required workers to be located in the United States while the second task had no location restriction. For both tasks, the time it takes to read directions and answer the questions enabled workers to earn roughly $9/hour. This is currently well above the average worker compensation on Mechanical Turk of $2-5/hour (Hara et al., 2018; Hitlin, 2016) which states only 4% of workers earn more than the U.S. federal minimum wage of $7.25/hour. The second task, which contains no location restriction, may have workers from countries with a lower minimum wage. In Figures 4 and 5 we provide screenshots of an example of our conspiracy theory affirm task, including the topic reference text and worker instructions.

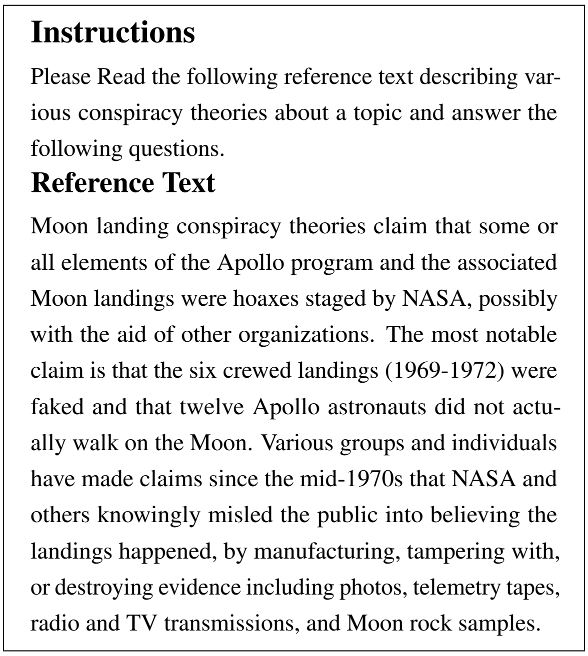

Instructions

Please Read the following reference text describing various conspiracy theories about a topic and answer the following questions.

Reference Text

Moon landing conspiracy theories claim that some or all elements of the Apollo program and the associated Moon landings were hoaxes staged by NASA, possibly with the aid of other organizations. The most notable claim is that the six crewed landings (1969-1972) were faked and that twelve Apollo astronauts did not actually walk on the Moon. Various groups and individuals have made claims since the mid-1970s that NASA and others knowingly misled the public into believing the landings happened, by manufacturing, tampering with, or destroying evidence including photos, telemetry tapes, radio and TV transmissions, and Moon rock samples.

Figure 4: Mechanical Turk conspiracy theory affirmation reference text.

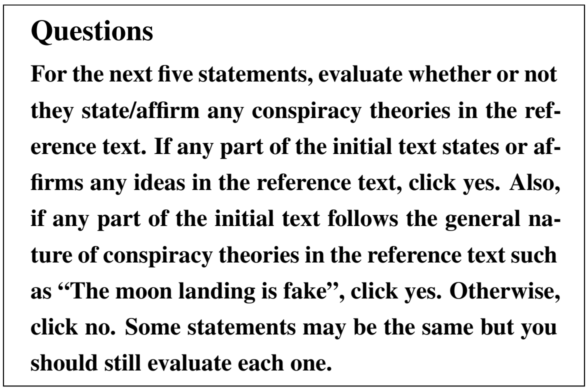

Questions

For the next five statements, evaluate whether or not they state/affirm any conspiracy theories in the reference text. If any part of the initial text states or affirms any ideas in the reference text, click yes. Also, if any part of the initial text follows the general nature of conspiracy theories in the reference text such as “The moon landing is fake”, click yes. Otherwise, click no. Some statements may be the same but you should still evaluate each one.

Figure 5: Mechanical Turk conspiracy theory affirmation instructions.

A Text-Centered Shared-Private Framework via Cross-Modal Prediction for Multimodal Sentiment Analysis

Yang Wu[1] **Zijie Lin**[1] **Yanyan Zhao**[1*] **Bing Qin**[1] **Li-Nan Zhu**[2]
[1] Harbin Institute of Technology [2] Zhejiang University of Technology
[1] `{ywu, linzijie97, yyzhao, qinb}@ir.hit.edu.cn`
[2] `zln@zjut.edu.cn`

Abstract

Multimodal fusion is a core problem for multimodal sentiment analysis. Previous works usually treat all three modal features equally and implicitly explore the interactions between different modalities. In this paper, we break this kind of methods in two ways. Firstly, we observe that textual modality plays the most important role in multimodal sentiment analysis, and this can be seen from the previous works. Secondly, we observe that comparing to the textual modality, the other two kinds of non-textual modalities (visual and acoustic) can provide two kinds of semantics, shared and private semantics. The shared semantics from the other two modalities can obviously enhance the textual semantics and make the sentiment analysis model more robust, and the private semantics can be complementary to the textual semantics and meanwhile provide different views to improve the performance of sentiment analysis together with the shared semantics. Motivated by these two observations, we propose a text-centered shared-private framework (TCSP) for multimodal fusion, which consists of the cross-modal prediction and sentiment regression parts. Experiments on the MOSEI and MOSI datasets demonstrate the effectiveness of our shared-private framework, which outperforms all baselines. Furthermore, our approach provides a new way to utilize the unlabeled data for multimodal sentiment analysis.

1 Introduction

Multimodal sentiment analysis is an emerging research field, which aims to understand people's sentiment using not only textual but also non-textual (visual, acoustic) data. This task has attracted increasing attention from the community recently, as people have realized that non-textual clues are helpful for detecting sentiment and the huge demands for the identification of opinions and sentiment in the video.

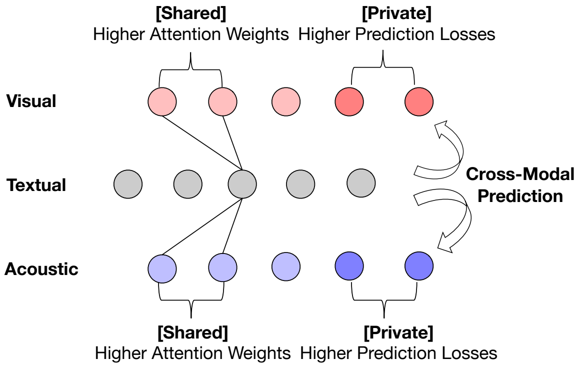

Figure 1: Distinguishing the shared and private features via cross-modal prediction.

Comparing to the traditional textual sentiment analysis(Liu, 2012), previous work demonstrates that the other non-textual data can improve the final performance (Chen et al., 2017; Zadeh et al., 2018b; Sun et al., 2020). There are two reasons. The first reason is that the three modalities can convey some common semantics. In this case, these non-textual common semantics do not provide additional information beyond textual data, but the repetitive information from them can strengthen the final performance. We call them *shared* semantics. The other reason is that the three modalities have their own special semantics, which are different to other modalities. These non-textual private semantics is modality-specific and hard to be predicted only by textual data. Thus this kind of semantics from the non-textual modalities can help to detect the final sentiment more accurately. We call them *private* semantics.

Previous works usually don't distinguish the shared semantics and the private semantics but treat each modal semantics as a whole, lacking the ability to explicitly explore the interaction between

---

[*] Corresponding Author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4730–4738
August 1–6, 2021.

different modalities. In this paper, we propose a text-centered shared-private framework for multimodal sentiment analysis. In this framework, the textual modality is considered as the core modality, and we first design a cross-modal prediction task to explicitly distinguish the shared and the private semantics between the textual modality and the non-textual(visual, acoustic) modality and then propose the sentiment regression model including the shared and private modules to fuse the textual features with two types of non-textual features.

In order to explore the shared and private semantics from non-textual modalities, we design the cross-modal prediction task, which is like a machine translation framework. The source is a sequence of textual modal features and the target is a sequence of another modal (visual or acoustic) features. We can explore the shared and private semantics via training two cross-modal prediction models, textual-to-visual and textual-to-acoustic models. In specific, as shown in Figure 1, we apply the pre-trained textual-to-visual and textual-to-acoustic models to predict the targets. The features of the target modality with higher prediction losses are distinguished as private. For each word, those putting higher attention weights on this word are distinguished as shared.

After obtaining shared and private features, we propose the sentiment regression model to fuse the textual features with two types of features. The sentiment regression model mainly consists of three parts, shared module, private module, and regression layer. In the shared module, each textual feature interacts with the corresponding shared features to get the enhanced features, which are then fed into a fusion block to obtain the final shared representation. Meanwhile, in the private module, the private features of each non-textual modality are passed through the attention layer to obtain the final private representation. Finally, we feed the obtained representations into the regression layer to predict the sentiment score.

We conduct experiments on two multimodal sentiment analysis benchmarks: CMU-MOSI and CMU-MOSEI. The experimental results show that our model outperforms all baselines. This can demonstrate that the shared-private framework for multimodal sentiment analysis can explicitly use the shared semantics between different modalities to enhance the final performance of sentiment analysis, and meanwhile can explicitly use the private semantics of each modality as the supplemental clues for sentiment analysis. In addition, we can observe that our designed cross-modal prediction task can accurately distinguish the shared and private non-textual semantics.

Our contributions can be concluded as follows.

1. We propose a challenging text-centered shared-private framework for the multimodal sentiment analysis. This framework can effectively fuse textual and non-textual features benefitting from the unlabeled data.

2. We design a cross-modal prediction task to explore the shared and private semantics for each non-textual modality.

3. We achieve significant improvements from learning the shared and private semantics from different modalities compared to the algorithms without distinguishing the shared and private semantics.

2 Related Work

There are two lines of works conducted on multimodal sentiment analysis.

One is focusing on utterance-level multimodal feature fusion. These methods use the features of the overall utterance. For example, they first extract the frame-level visual or acoustic features and then average them to obtain the final features, which are called utterance-level features. The utterance-level textual features can be obtained by applying RNNs for words. The obtained utterance-level features are fed into the fusion model to get the multimodal representation. Some models have been proposed for effective multimodal feature fusion. Zadeh et al. (2017) proposed Tensor Fusion to explicitly capture unimodal, bimodal, and trimodal interactions. But this method uses the three-fold Cartesian product to fuse the multimodal features, which makes the time cost very high. To address it, Liu et al. (2018) presented the Efficient Low-rank Multimodal Fusion, which applies multimodal fusion using low-rank tensors to accelerate the fusion process. Mai et al. (2020) proposed a graph fusion network to model unimodal, bimodal, and trimodal interactions successively.

The utterance-level features mainly contain global information, which may fail to capture local information. Therefore, recent works are mostly focusing on word-level multimodal feature fusion.

And our work in this paper is also based on word-level features. To extract word-level features, the first step is applying force alignment to obtain the timestamps of each word including the start time and end time. And then following the timestamps, the utterance is split into some video clips. Finally, word-level visual or acoustic features are obtained by averaging the frame-level features of the video clips. Based on word-level features, lots of methods are proposed for performing word-level multimodal feature fusion. Zadeh et al. (2018a) proposed the Memory Fusion Network(MFN) to capture the interactions across both different modalities and timesteps. Inspired by the observation that the meaning of words often varies dynamically in different non-verbal contexts, Wang et al. (2019) proposed the Recurrent Attended Variation Embedding Network (RAVEN). This model applies the Attention Gating module to fuse the word-level features, which can dynamically use the non-verbal features to shift the word embeddings. Tsai et al. (2019) presented multimodal transformer (Mult), which uses the cross-modal attention to capture the bimodal interactions, motivated by the great success of transformer in NLP(Vaswani et al., 2017).

Besides, there is a related work (Pham et al., 2019) need to be noticed, which proposed that translation from a source to a target modality provides a way to learn joint representations and proposed the Multimodal Cyclic Translation Network model (MCTN) to learn joint multimodal representations. Comparing to this work, our model has a significant difference. That is we use the cross-modal prediction task to distinguish the shared and private non-textual features instead of training the model as an auxiliary task. In this way, we can obtain more useful information by deeply probing the cross-modal prediction model.

3 Approach

In this section, we will introduce our shared-private framework for multimodal sentiment analysis in detail. In this framework, we treat the textual modality as the core, then how to explore the shared and the private semantics of the non-textual modality compared to the textual modality, and how to fuse all three modal features are two important steps. For the first step, we design a cross-modal prediction task and obtain the shared and private features of the non-textual modalities via training two cross-modal prediction models, textual-to-visual and textual-to-acoustic models. And for the second step, we design a sentiment regression model to fuse the textual features and the two types of features.

3.1 Cross-Modal Prediction

Task Definition: The cross-modal prediction task is formalized as follows. Given a sequence of textual features denoted as $\mathbf{x}_l = \{x_l^t : 1 \leq t \leq L, x_l^t \in \mathbb{R}^{d_l}\}$, L is the length of the given sequence, t is the timestep, and the goal is to predict the corresponding sequence of visual or acoustic features denoted as $\mathbf{x}_i = \{x_i^t : 1 \leq t \leq L, x_i^t \in \mathbb{R}^{d_i}\}$, $i \in \{v, a\}$. Cross-modal prediction task is inspired by the machine translation task. The inputs are the textual features, and the generated outputs are the non-textual (visual or acoustic) features. During the translation from the textual modality to other modalities, we can exploit the shared and the private semantics of the non-textual modalities.

Prediction Model: We use the Seq2Seq model with attention (Bahdanau et al., 2015) as our model framework. The encoder takes the textual features $\mathbf{x}_l$ as inputs and outputs the hidden states $\mathbf{h}_{enc} = \{h_{enc}^t : 1 \leq t \leq L, h_{enc}^t \in \mathbb{R}^{d_h}\}$. The decoder takes the previous hidden state h_{dec}^{t-1} and hidden states of the encoder as inputs and predicts the non-textual feature x_i^t, $i \in \{v, a\}$, at the t timestep. We choose the MSE as our loss function. The prediction loss values are denoted as $\mathbf{e}_{l\to i} = \{e_{l\to i}^t : 1 \leq t \leq L\}$. The attention map of the prediction model is denoted as $\mathbf{m}_{i\to l}$. In practice, we apply LSTMs (Hochreiter and Schmidhuber, 1997) to implement our encoders and decoders. After training the cross-modal prediction models using textual-visual and textual-acoustic paired data, we can obtain two models, textual-to-visual and textual-to-acoustic models. We then use the obtained models to distinguish the shared and private features and record the results as shared and private masks, which will be passed to the sentiment regression model.

Shared Mask: We propose the shared mask to find out the shared semantics of the two kinds of non-textual modalities. The basic assumption is that during cross-modal prediction, if the prediction model wants to generate a non-textual feature as precisely as possible, it should pay more attention to the input textual features, that contain more shared semantics. Based on this assumption, we design the method to obtain the shared mask. Given

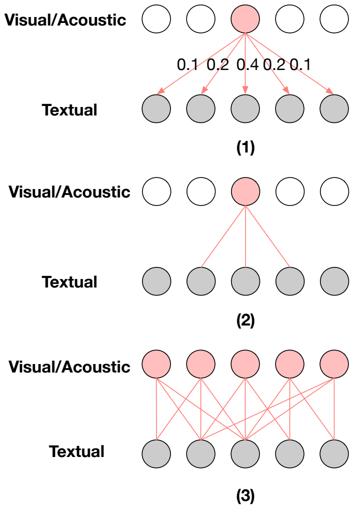

Figure 2: Obtaining the shared mask from the cross-modal prediction model.

$\mathbf{m}_{i\to l}$, $i \in \{v, a\}$, for each row t, we first sort the attention weights $\mathbf{m}^{t,*}_{i\to l}$ and then get the indexes S^t of the largest K_s values. Finally, we can get the shared mask $smask$, $smask \in \mathbb{R}^{L*L}$. $smask^{t_1,t_2}$ is 1 if the $t_1 \in S^{t_2}$ and 0 otherwise.

To describe this method intuitively, we show this process in Figure 2. There are three steps: (1) We build an attention graph and the values of edges mean the attention weights. We illustrate a part of it for simplicity. (2) We only keep the edges with larger values for each non-textual node and delete others. (3) We map the graph to the shared mask $smask$, $smask^{t_1,t_2}$ is 1 if there is a edge between textual node t_1 and non-textual node t_2 and 0 otherwise. The shared mask will be passed to the share module of the sentiment regression model to make the model focus on the shared features.

Private Mask: In order to find out the private semantics of the two kinds of non-textual modalities, we propose the private mask. The basic assumption is that the features containing modality-private information are difficult to be predicted by textual modality. The private mask of a given utterance is obtained as follows. Given an utterance which includes three modalities, textual, visual and acoustic, denoted as $\mathbf{x}_i = \{x_i^t : 1 \le t \le L, x_i^t \in \mathbb{R}^{d_i}\}$, $i \in \{l, v, a\}$, we first use the trained prediction models to get the loss values, $\mathbf{e}_{l\to v}$ and $\mathbf{e}_{l\to a}$. Then we sort the loss values to obtain the indexes P of the largest K_p values. Finally, We can get the private mask $pmask$, $pmask \in \mathbb{R}^L$. $pmask^t$ is 1 if the $t \in P$ and 0 otherwise. The private mask will be used by the private module of the sentiment regression model to force the model to focus on private features.

3.2 Regression

In this section, we study how to fuse the shared and private information obtained from Section 3.1. An illustration of our framework is given in **Figure 3**.

3.2.1 Input Layer

Given an utterance which includes three modalities, textual, visual and acoustic, the extracted multimodal features are denoted as $\mathbf{x}_i = \{x_i^t : 1 \le t \le L, x_i^t \in \mathbb{R}^{d_i}\}$, $i \in \{l, v, a\}$. We use three LSTM networks to encode the input multimodal features $\mathbf{x}_i$, producing $\mathbf{h}_i = \{h_i^t : 1 \le t \le L, h_i^t \in \mathbb{R}^{d_h}\}$.

$$\begin{aligned} \mathbf{h}_l &= \mathrm{LSTM_l}(\mathbf{x}_\mathrm{l}) \\ \mathbf{h}_v &= \mathrm{LSTM_v}(\mathbf{x}_\mathrm{v}) \\ \mathbf{h}_a &= \mathrm{LSTM_a}(\mathbf{x}_\mathrm{a}) \end{aligned} \tag{1}$$

3.2.2 Shared Module

The core idea of the shared module is leveraging the shared information from non-textual modal features to enhance the representations of words. To achieve it, we propose the masked cross-modal attention network, which can utilize the shared masks obtained from cross-modal prediction models to focus on the non-textual shared features.

In the masked cross-modal attention network, we first calculate the attention scores across the non-textual representations $\mathbf{h}_i$, $i \in \{v, a\}$, for each word. We denote the scores as $\mathbf{s}_{l\to i}$.

$$\begin{aligned} \mathbf{s}^{t_1,t_2}_{l\to v} &= W_2(tanh(W_1([h_l^{t_1}; h_v^{t_2}]) + b_1)) \\ \mathbf{s}^{t_1,t_2}_{l\to a} &= W_4(tanh(W_3([h_l^{t_1}; h_a^{t_2}]) + b_3)) \end{aligned} \tag{2}$$

where $W_1, W_3 \in \mathbb{R}^{d_h \times 2d_h}$, $W_2, W_4 \in \mathbb{R}^{1\times d_h}$, $b_1, b_3 \in \mathbb{R}^{d_h}$ are the parameters of the score functions.

To focus on the shared features, we first calculate the attention weights $\mathbf{w}_{l\to v}$ and $\mathbf{w}_{l\to a}$ using the softmax function and mask the other features out using the shared mask.

$$\begin{aligned} \mathbf{w}^{t_1,t_2}_{l\to v} &= \frac{e^{\mathbf{s}^{t_1,t_2}_{l\to v}}}{\sum_{t_3=1}^{L} e^{\mathbf{s}^{t_1,t_3}_{l\to v}}} \\ \mathbf{w}^{t_1,t_2}_{l\to a} &= \frac{e^{\mathbf{s}^{t_1,t_2}_{l\to a}}}{\sum_{t_3=1}^{L} e^{\mathbf{s}^{t_1,t_3}_{l\to a}}} \end{aligned} \tag{3}$$

$$\begin{aligned} \mathbf{w}_{l\to v} &= \mathbf{w}_{l\to v} \circ smask_{l\to v} \\ \mathbf{w}_{l\to a} &= \mathbf{w}_{l\to a} \circ smask_{l\to a} \end{aligned} \tag{4}$$

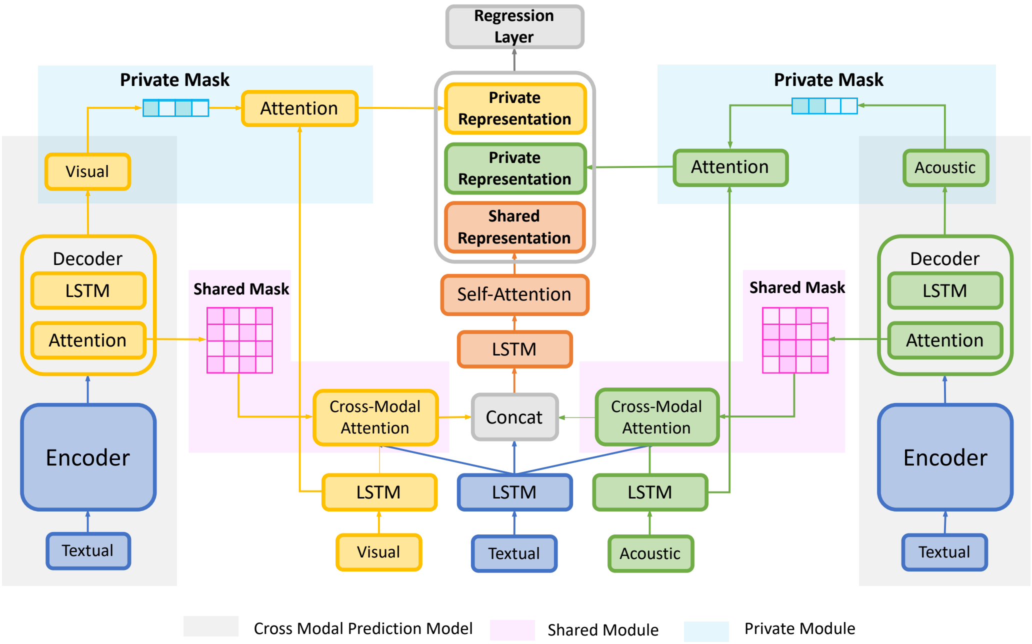

Figure 3: Illustration of our shared-private framework.

We obtain the non-textual shared context vectors $\mathbf{c}_v$ and $\mathbf{c}_a$. $\mathbf{c}_v, \mathbf{c}_a \in \mathbb{R}^{L \times d_h}$.

$$\begin{aligned} \mathbf{c}_v &= \mathbf{w}_{l \to v} \mathbf{h}_v \\ \mathbf{c}_a &= \mathbf{w}_{l \to a} \mathbf{h}_a \end{aligned} \tag{5}$$

To fuse textual and non-textual shared features, we concatenate $\mathbf{c}_v$, $\mathbf{c}_a$, and $\mathbf{h}_l$ and feed it into the fusion LSTM network, producing $\mathbf{r}_s \in \mathbb{R}^{L \times 3d_h}$. We further use a self-attention layer, which is denoted as $\mathrm{SelfAttentionLayer}$, to learn the final representation. The self-attention layer is similar to the cross-modal attention network. We use the last step representation of $\mathbf{r}_n$ as the shared representation, which is denoted as r_s.

$$\begin{aligned} \mathbf{r}_m &= \mathrm{LSTM}_{\mathrm{fusion}}([\mathbf{c}_{\mathrm{v}}; \mathbf{c}_{\mathrm{a}}; \mathbf{h}_{\mathrm{l}}]) \\ \mathbf{r}_n &= \mathrm{SelfAttentionLayer}(\mathbf{r}_{\mathrm{m}}) \end{aligned} \tag{6}$$

3.2.3 Private Module

To enable the model to capture the unique information contained in non-textual modalities, we design the private module. Specifically, we use the attention network to learn informative and modality-private representations.

$$\begin{aligned} \mathbf{s}_v^t &= W_5 h_v^t + b_5 \\ \mathbf{s}_a^t &= W_6 h_a^t + b_6 \end{aligned} \tag{7}$$

where $W_5, W_6 \in \mathbb{R}^{1 \times d_h}$, $b_5, b_6 \in \mathbb{R}$ are the parameters of the score functions.

We use private masks to ignore other features and apply the softmax function to get the attention weights.

$$\begin{aligned} \mathbf{s}_v &= \mathbf{s}_v + (1 - pmask_v) * (-10^8) \\ \mathbf{s}_a &= \mathbf{s}_a + (1 - pmask_a) * (-10^8) \end{aligned} \tag{8}$$

Finally, we compute the weighted sum and represent them as p_v and p_a, which are called the private representations.

$$\begin{aligned} \mathbf{w}_v^t &= \frac{e^{\mathbf{s}_v^t}}{\sum_{t_1=1}^{L} e^{\mathbf{s}_v^{t_1}}} \\ \mathbf{w}_a^t &= \frac{e^{\mathbf{s}_a^t}}{\sum_{t_1=1}^{L} e^{\mathbf{s}_a^{t_1}}} \end{aligned} \tag{9}$$

$$\begin{aligned} p_v &= \mathbf{w}_v \mathbf{h}_v \\ p_a &= \mathbf{w}_a \mathbf{h}_a \end{aligned} \tag{10}$$

3.2.4 Regression Layer

We design the regression layer, which is implemented by a two-layer network with ReLU activation function, to fuse the shared and private representations.

$$\hat{y} = W_o(ReLU(W_f([r_s; p_v; p_a]) + b_f)) + b_o \tag{11}$$

where $W_f \in \mathbb{R}^{d_h \times 5d_h}$, $W_o \in \mathbb{R}^{1 \times d_h}$, $b_f \in \mathbb{R}^{d_h}$, $b_o \in \mathbb{R}$.

Table 1: Hyperparameters of our model.

Models	Parameters	MOSI	MOSEI
Cross-Modal Prediction	Batch Size	24	24
	Max Length	50	128
	Hidden Size	100	100
	Epochs	40	40
	Learning Rate	0.0001	0.0001
	Dropout	0.5	0.5
	Patience	5	10
Regression	Batch Size	24	24
	Max Length	50	128
	Hidden Size	100	100
	Epochs	30	30
	Learning Rate	0.001	0.001
	Dropout	0.5	0.5
	Selection Number	5	5
	Patience	5	5

4 Experiments

4.1 Datasets

We conduct experiments on two public datasets, CMU-MOSI (Zadeh, 2015) and CMU-MOSEI (Zadeh et al., 2018b) to evaluate our proposed model. CMU multimodal opinion-level sentiment intensity (CMU-MOSI) consists of 93 videos collected from the YouTube website. The length of the videos varies from 2-5 mins. These videos are split into 2,199 short video clips and labeled with sentiment scores from -3 (strongly negative) to 3 (strongly positive). CMU multimodal opinion sentiment and emotion intensity (CMU-MOSEI) consists of 23,453 annotated video utterances from 1,000 distinct speakers and 250 topics. Each utterance is annotated with sentiment scores from -3 (strongly negative) to 3 (strongly positive).

The multimodal features used in our experiments are described as follows. We use glove word embeddings (Pennington et al., 2014) to represent the words. The dimension of each word embedding is 300. We extract the visual features using Facet, which can extract 35 facial action units (Ekman et al., 1980; Ekman, 1992) from each frame resulting in a 35-dimensional vector. The acoustic features are obtained by applying COVAREP (Degottex et al., 2014), which includes 12 Mel-frequency cepstral coefficients (MFCCs) and other low-level features. The dimension of the acoustic feature is 74.

4.2 Evaluation Metrics

Following previous works, we take 2-class accuracy(Acc), f1 score(F1), mean absolute error (MAE), and correlation(Corr) as our evaluation metrics. As the prediction results are real values, we first use mean absolute error and Corr between prediction scores and ground truths to evaluate the models. In addition, we then map the sentiment scores into sentiment labels and use classification metrics, such as accuracy and f1 score, to assess the model performance.

4.3 Training Details

The hyperparameters of our model are listed in Table 1. In practice, we apply dropout before the last linear layer for regularization. We use Adam as the optimizer. The learning rate is decayed once the validation loss stops decreasing. The Selection Number is the number of selected shared/private features, K_s and K_p. We take the same value for K_s and K_p for simplicity.

4.4 Baselines

We compare our proposed model with the following baselines. **EF-LSTM** fuses the multimodal features by concatenating and applies an LSTM network to get the final representation. **LF-LSTM** first uses three LSTM networks to encode three modal features and concatenates three obtained representations to get the final representation. **MFN** (Zadeh et al., 2018a) captures the interactions across both the different modalities and time. **RAVEN** (Wang et al., 2019) first combines the nonverbal information with word representations and then feeds the modified word representations into an LSTM network to obtain the utterance representation. **MCTN** (Pham et al., 2019) learns joint multimodal representations by translating between modalities. **MulT** (Tsai et al., 2019) uses cross-modal transformers to fuse multimodal features. **Multimodal Routing** (Tsai et al., 2020) proposes a routing mechanism to capture the interactions between input modalities and outputs. **TCSP(Base)** is our base model. The model architecture is as same as our full model, but it doesn't use shared and private masks. Comparing TCSP(Base) and TCSP(Full), we can judge whether distinguishing the shared and private features of non-textual modalities is useful.

4.5 Experimental Results

We compare our model with several baselines and the experimental results are shown in Table 2. Comparing our base model with other baselines, our base model fails to obtain the best result and underperforms RAVEN and MulT for the Acc, F1

Table 2: Experimental results on the test sets of the MOSEI and MOSI dataset. The best results are in **bold**. As the Multimodal Routing model is designed for classification, we don't report the regression metrics of it for fair comparison.

Models	MOSI				MOSEI			
	Acc ↑	F1 ↑	MAE ↓	Corr ↑	Acc ↑	F1 ↑	MAE ↓	Corr ↑
EF-LSTM	76.0/75.3	75.9/75.2	1.020/1.023	0.603/0.608	78.4/78.2	79.5/77.9	0.642/0.642	0.641/0.616
LF-LSTM	75.3/76.8	75.1/76.7	1.046/1.015	0.600/0.625	80.3/80.6	80.8/80.6	0.606/0.619	0.676/0.659
MFN	74.5/77.4	74.4/77.3	1.036/0.965	0.607/0.632	78.1/-	79.2/-	0.640/-	0.637/-
RAVEN	76.2/78.0	76.0/76.6	1.012/0.915	0.614/0.691	81.3/79.1	81.6/79.5	0.595/0.614	0.701/0.662
MCTN	71.6/79.3	71.5/79.1	1.142/0.909	0.487/0.676	80.8/79.8	80.6/80.6	0.611/0.609	0.670/0.670
MulT	78.9/83.0	78.8/82.8	1.000/0.871	0.670/0.698	81.8/82.5	81.8/82.3	0.605/0.580	0.682/0.703
Multimodal Routing	68.5/-	68.4/-	-/-	-/-	76.0/81.7	75.6/81.8	-/-	-/-
TCSP(Base)	79.3	79.3	0.956	0.658	80.7	80.3	0.593	0.692
TCSP(Full)	**80.9**	**81.0**	**0.908**	**0.710**	**82.8**	**82.7**	**0.576**	**0.715**

Table 3: Ablation analysis of TCSP evaluated on the test data. The best results are in **bold**.

Models	MOSI				MOSEI			
	Acc ↑	F1 ↑	MAE ↓	Corr ↑	Acc ↑	F1 ↑	MAE ↓	Corr ↑
TCSP	**80.9**	**81.0**	**0.908**	**0.710**	**82.8**	**82.7**	**0.576**	**0.715**
w/o Private Mask	79.9	79.8	0.930	0.663	82.2	82.1	0.576	0.710
w/o Shared Mask	79.0	79.0	0.965	0.660	82.3	82.1	0.585	0.701
w/o Both Masks	79.3	79.3	0.956	0.658	80.7	80.3	0.593	0.692

metrics on the MOSEI dataset. However, with the help of the cross-modal prediction task, our text-centered shared-private framework (TCSP) achieves the best performance and outperforms all baselines on both datasets. This can demonstrate that the shared-private framework proposed in this paper is effective for multimodal sentiment analysis. Furthermore, it can be observed that the shared and private features for each non-textual modality obtained from the cross-modal prediction task can provide useful clues for the interactions between different modalities. Thus, these non-textual shared-private features can be jointly fused with the textual features to improve the multimodal sentiment analysis.

We also observe that there is a larger margin between our full model and our base model on the MOSI dataset. We attribute it to the small data size of the MOSI dataset. It is insufficient for training the base model, which makes it benefit more from the shared and private information.

It should be noted that, in Table 2, we provide two results for each method on each dataset. The left result is obtained by rerunning the public codes in the same experimental setting, which refers to the same dataset split and the same extracted features of three modalities. The right result is copied from previous papers and the experimental settings are different. To guarantee the justice, we compare our TCSP model with the left results.

5 Analysis

5.1 Ablation Study

We conduct the ablation experiments to distinguish the contribution of each part. As shown in Table 3, ablating either shared mask or private mask hurts the model performance, which indicates that both masks are useful for the sentiment prediction. The shared mask can enable the sentiment regression model to get the modality-shared features resulting in a more robust regression model. The private mask makes the regression model focus on modality-private features, which provides extra information for sentiment prediction. With the help of shared and private masks, the regression model in the shared-private framework can fuse the textual features with two types of non-textual features individually, which is the more effective method for multimodal feature fusion.

5.2 Effect of Selection Number

Selection Number is the number of selected shared/private features, K_s and K_p. We take the same value for K_s and K_p for simplicity. We evaluate our model with different selected numbers from 1 to 8 on the MOSEI dataset to quantify the effect. The experimental results are shown in Figure 4. We can observe that our model achieves the best performance on the Acc and F1 metrics when the Selection Number is 5. The possible reason is that

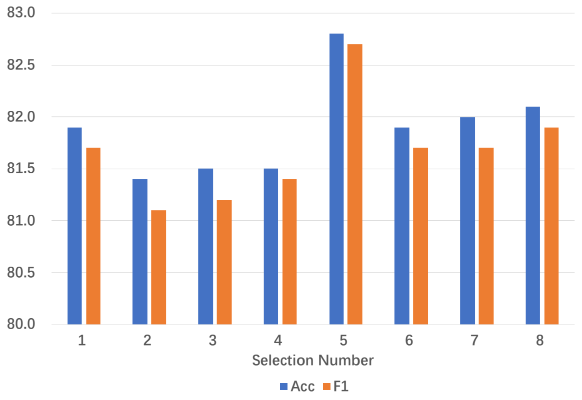

Figure 4: Experimental results with different selection numbers on the MOSEI dataset.

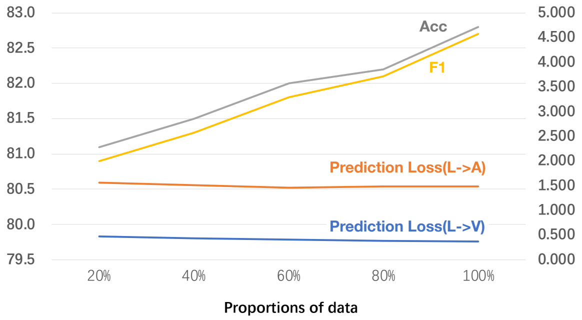

Figure 5: Experimental results on the MOSEI dataset with different proportions of data used for the cross-modal prediction model.

too small Selection Number makes the model only focus on few features. This could result in missing useful information. In contrast, too large one makes the model attend too many features, which weakens the effect of masks. For this reason, selecting a middle number could be better.

5.3 Effect of Cross-Modal Prediction Model

The cross-modal prediction task is the core of our shared-private framework, and it has been demonstrated that this task is effective from Table 3 and Section 5.1. In this section, we want to further explore the effect of cross model prediction for the final regression model.

In Figure 5, we design different cross-modal prediction models trained with different proportions (from 20% to 100%) of MOSEI data and then fuse the obtained shared and private information into the regression models. It should be noticed that all regression models are trained with all data of the MOSEI dataset. The results show that when we use more data, the final performance is better. And meanwhile, it can be observed that the two kinds of prediction losses (from textual to visual modality and from textual to acoustic modality) are decreased when the proportion of the used data is increased.

This can reveal that the cross-modal prediction model trained with more data can provide more informative supervision signals, which are the shared and private masks specifically. If the performance of cross-modal prediction model is low, it is impossible to teach the regression model to play the precise role in the shared-private framework.

6 Conclusion

In this paper, we propose a text-centered shared-private framework for multimodal sentiment analysis. In this framework, we treat the textual modality as the core and aim to use the other non-textual modalities to help enrich the semantics of the textual modality. For each non-textual modality, we consider two types of semantics, shared and private, which have different functions. Shared semantics can enhance the textual semantics to make the model more robust and the private semantics can provide extra information for more precise prediction.

To distinguish these two semantics, we design a cross-modal prediction task and record the results as share and private masks. We further propose a regression model utilizing the shared and private modules to fuse the textual features with two non-textual features. The experimental results demonstrate that distinguishing the shared and private non-textual semantics and explicitly modeling the interactions between textual and two non-textual semantics is a better way for the multimodal sentiment analysis than just treating each non-textual features as a whole. The analyses show that the regression model can benefit more from the better cross-modal prediction model, which also indicates that the cross-modal prediction process can produce useful supervision signals only using unlabeled data.

In future work, we plan to collect more unlabeled data to enhance our model. In addition, we would also like to explore other approaches using the unlabeled data to help multimodal feature fusion.

Acknowledgments

This work was supported in part by the following Grants: National Natural Science Foundation of China (No. 61632011, No. 61772153), National Key R&D Program of China (No. 2018YFB1005103).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarep - a collaborative voice analysis repository for speech technologies. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 960–964. IEEE.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Paul Ekman, Wallace V Freisen, and Sonia Ancoli. 1980. Facial signs of emotional experience. *Journal of personality and social psychology*, 39(6):1125.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, AmirAli Bagher Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256.

Sijie Mai, Haifeng Hu, and Songlong Xing. 2020. Modality to modality translation: An adversarial representation learning and graph fusion network for multimodal fusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 164–172.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hai Pham, Paul Pu Liang, Thomas Manzini, Louis-Philippe Morency, and Barnabás Póczos. 2019. Found in translation: Learning robust joint representations by cyclic translations between modalities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6892–6899.

Zhongkai Sun, Prathusha Sarma, William Sethares, and Yingyu Liang. 2020. Learning relationships between text, audio, and video via deep canonical correlation for multimodal language analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8992–8999.

Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6558–6569.

Yao-Hung Hubert Tsai, Martin Ma, Muqiao Yang, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Multimodal routing: Improving local and global interpretability of multimodal language analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1823–1833.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yansen Wang, Ying Shen, Zhun Liu, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2019. Words can shift: Dynamically adjusting word representations using nonverbal behaviors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7216–7223.

Amir Zadeh. 2015. Micro-opinion sentiment intensity analysis and summarization in online videos. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 587–591.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark. Association for Computational Linguistics.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multi-view sequential learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018b. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246.

What Would a Teacher Do? Predicting Future Talk Moves

Ananya Ganesh, Martha Palmer and **Katharina Kann**
University of Colorado Boulder
{ananya.ganesh, martha.palmer, katharina.kann}@colorado.edu

Abstract

Recent advances in natural language processing (NLP) have the ability to transform how classroom learning takes place. Combined with the increasing integration of technology in today's classrooms, NLP systems leveraging question answering and dialog processing techniques can serve as private tutors or participants in classroom discussions to increase student engagement and learning. To progress towards this goal, we use the classroom discourse framework of academically productive talk (APT) to learn strategies that make for the best learning experience. In this paper, we introduce a new task, called future talk move prediction (FTMP): it consists of predicting the next talk move – an utterance strategy from APT – given a conversation history with its corresponding talk moves. We further introduce a neural network model for this task, which outperforms multiple baselines by a large margin. Finally, we compare our model's performance on FTMP to human performance and show several similarities between the two.

1 Introduction

The field of natural language processing (NLP) has made rapid progress over the last few years (Wang et al., 2019). Success on natural language understanding, dialogue generation, and question answering tasks has spurred advances in NLP-based systems for educational applications. (McNamara et al., 2013; Litman, 2016; Burstein et al., 2020). Systems that can simulate human teachers in specific situations such as small-group discussions have the potential to aid learning by promoting student engagement.

Research has shown that deep conceptual learning is heightened when students are active participants in the classroom and contribute to discussions with their questions and ideas (McNamara, 2011; Bransford et al., 1999). However, large class sizes often make it difficult for all students to actively participate. Discussions in sub-groups increase each student's speaking time, but, in turn, make it impossible for a single teacher to guide all individual conversations. In this paper, we present a first step towards a system that can solve this problem by taking the teacher's role in facilitating sub-group discussions.

Scenario: Learning about proportional relationships in a classroom. The teacher gives an example of toasting two slices in a toaster, for 2 minutes.
Teacher: So we've just seen that 2 slices of toast gets done in 2 minutes. (*None*) **Teacher:** What if I had 3 slices of toast? (*Press for Accuracy*) **Student:** 4 minutes! (*Wait*) **Teacher:** Why would it take 4 minutes? (*Press for Reasoning*) **Student:** Because you'd have to use the toaster twice. (*Wait*)
FTMP: Getting Students to Relate *(e.g., who else agrees it would be 4?)*

Table 1: Our proposed FTMP task; the teacher talk move corresponding to each utterance is shown in parentheses.

For this, we turn to a classroom discourse framework by Michaels et al. (2008) called academically productive talk (APT). This framework, which we describe in detail in Section 2, provides both teachers and students with a set of *talk moves* – a family of utterance strategies to use for productive and respectful in-class discussions. As a first step towards developing an NLP system that can guide academically productive discussions, we aim to design a model which can predict when which specific talk move is appropriate. Thus, we introduce the task of **future talk move prediction (FTMP)** – given a conversation history, the goal is to predict what the *next* teacher talk move should be. We formulate this as a multi-class classification problem, with the input being a sequence of previous utterances

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4739–4751
August 1–6, 2021.

and their corresponding talk moves, and the label being the next talk move.

We further propose a model for FTMP, which we call 3-E.[1] It consists of three recurrent neural network (RNN) encoders: one for individual utterances, one for utterance sequences, and one for talk move sequences. The model is trained on transcripts of classroom discussions where teacher utterances have been annotated for the talk moves they represent. We consider the actions of the teacher to be our gold standard data for FTMP. We show that our model strongly outperforms multiple baselines and that adding sentence representations from RoBERTa (Liu et al., 2019) or TOD-BERT (Wu et al., 2020) – a model trained on task-oriented dialogue – does not increase performance further.

Finally, we investigate the performance of human annotators on FTMP. Unlike the teacher, they do not have access to multi-modal signals, subject matter information, or knowledge of student behavior. This setting, which mimics the information available to our model, is significantly different from the teachers who generate the gold standard utterances captured in our data. We present a detailed analysis of their performance on a diagnostic test set, and highlight similarities to our model's performance. Our findings indicate that our model produces acceptable predictions a majority of the time. However, a gap between model and human performance on this task shows that there is still room for improvement.

2 Academically Productive Talk

In this section, we provide an overview of the APT discourse framework and introduce a new task within the broader research area of NLP for educational applications: FTMP.

2.1 Background on APT

Research in cognitive science and psychology highlights the importance of active participation as opposed to passive listening for achieving deep conceptual learning (McNamara, 2011; Bransford et al., 1999; Chi and Wylie, 2014). This can take the form of reflection on the lesson, as well as generation of new ideas, such as asking and answering questions, connecting concepts, and coming up with explanations. Chapin et al. (2009); Goldenberg (1992); Cobb; Cazden (1988) discuss the importance of classroom conversations in this process. Chapin (2003) present case studies that show how implementing structured discussions in classrooms over a period of two years results in measurable improvements in test scores in mathematics.

To formalize how such discussions can be facilitated, Michaels et al. (2008) present a classroom discourse framework called *academically productive talk* (APT; also called *accountable talk*). This includes strategies that teachers and students can use to promote engagement as well as deep conceptual learning through discussions.

Facets Michaels et al. (2008) present three facets of accountability that APT encompasses: *accountability to the learning community*, *accountability to standards of reasoning*, and *accountability to knowledge*. The first facet emphasizes the importance of listening to other students' contributions, and, subsequently, building on top of them. The second facet promotes talk that is based on evidence and reasoning, and involves getting students to provide explanations for their claims. The last facet covers talk which involves factual knowledge – such as introducing a new concept, or challenging a student's claim to correct misconceptions.

Teacher Talk Moves Michaels and O'Connor (2015) conceptualize the above facets as "tools" that can be used by teachers and students to engage in APT. For both teachers and students, these tools take the form of utterance strategies called *talk moves*, which they can employ in order to conduct meaningful discussions.

In this paper, we focus on the following six talk moves used by teachers: (1) ***Keeping Everyone Together*** refers to utterances that manage student interactions, and asks students to be active listeners; (2) ***Getting Students to Relate*** refers to utterances that ask a student to build on other students' ideas by agreeing, disagreeing, or following up; (3) ***Restating*** occurs when a teacher repeats a student's answer or claim verbatim with the purpose of ensuring it reaches the entire classroom; (4) ***Revoicing*** happens when a teacher paraphrases a student's ideas, but adds or removes information in order to correct a student or convey new knowledge; (5) ***Pressing for Reasoning*** refers to utterances that ask a student to explain a decision or to connect multiple ideas; and (6) ***Pressing for Accuracy*** refers to utterances that prompt for answers to a factual question, e.g., about a method or

---

[1] Code for all models is available at `https://nala-cub.github.io/resources/`

Talk Move	**Description**	**Example**
Keeping Everyone Together	Ask students to be active listeners	Raise your hand if you know the answer
Getting Students to Relate	Ask students to contribute to another's ideas	Do you or agree or disagree with Michael?
Revoicing	Repeat what a student says with adding words or rephrasing	S: It had two T: So it had two edges
Restating	Repeat what a student says verbatim	S: Hexagon T: Hexagon!
Press for Accuracy	Prompt for an answer	What is this called?
Press for Reasoning	Prompt for explanation of thinking	How did you decide?
None	*Fits into none of the above*	*Good morning*
Wait	*Teacher says nothing while student speaks*	*S: It's the same shape*

Table 2: An overview of all teacher talk moves, their purpose and an example utterance. *None* and *Wait* are not APT talk moves, and represent generic utterances and teacher pauses during student utterances, respectively.

a result.

Keeping Everyone Together, Getting Students to Relate, and *Restating* are part of accountability to the learning community; *Revoicing* and *Press for Reasoning* are part of accountability to standards of reasoning, and *Press for Accuracy* falls under accountability to knowledge. Examples for all talk moves are shown in Table 2.

Student Talk Moves While we do not focus on student talk moves in this work, we summarize them here for completeness. Student talk moves can also be grouped into the same accountability facets as teacher talk moves (O'Connor and Michaels, 2019). Under accountability to the learning community, we have *Relating to Another Student* – building on a classmate's ideas or asking questions about them, and *Asking for more info* – requesting help from the teacher on a problem. Under accountability to knowledge, there is *Making a Claim* – providing an answer or a factual statement about a topic. Under accountability to standards of reasoning, we have *Providing Evidence/Explanation* – explaining their thinking with evidence.

2.2 Future Talk Move Prediction

In order to build a system that can facilitate in-class discussion in the way a human would, we aim at automatically answering the question *What would a teacher do?* at each point within a classroom conversation. Specifically, we define the task of future talk move prediction (FTMP) as choosing the next appropriate teacher talk move to make, given the history of what has been discussed so far.

Formally, the input for FTMP is a dialogue context $C = c_0, c_1, ..., c_t$, with each context element consisting of an utterance u_i, a binary variable s_i indicating if the speaker is different from the previous utterance, and a teacher talk move label t_i, i.e., $c_i = (s_i, u_i, t_i)$. The goal then is to predict the next teacher talk move t_{t+1} out of the possible talk moves defined above. Note that the future utterances are unseen; the prediction of the next talk move is to be made only based on the conversation history.

3 Related Work

3.1 Promoting APT with NLP Systems

Ideas from APT have been incorporated with success into intelligent tutoring systems (Dyke et al., 2013; Tegos et al., 2016; Adamson et al., 2014). These systems provide an environment to simulate classroom discussions, for instance, as small groups collaboratively solving problems with a shared textual chat interface for communication. The intelligent agent then plays a role similar to a teacher – it monitors the conversations and makes decisions about when to intervene in order to promote student engagement and learning.

Adamson et al. (2014) study the effects of two specific interventions: using the *Revoicing* talk move as well as an *Agree/Disagree* talk move (which corresponds to the *Getting Students to Relate* talk move in our above categorization). These interventions are made by matching student utterances in the chat to an annotated set of concepts and misconceptions for the topic being taught. Through multiple case studies, they show that interventions by the agent have a positive effect on learning, as measured by test scores before and after using the system. The agent interventions also prove useful in increasing student talk frequency. Similarly, Tegos et al. (2015) find that an APT-based intervention called Linking Contributions, similar to *Getting Students to Relate*, improves explicit reasoning as well as learning outcomes in students.

The findings of the above work provide a strong motivation for building a conversational AI system that can produce academically productive talk. Unlike the above systems, which focused particularly on accountability to the learning community, we attempt to predict opportunities for intervention across all talk moves described in Section 2. Since we do not have access to gold annotations of statements corresponding to concepts and misconceptions, we make use of transcripts of classroom discourse with annotations for talk moves used by the teacher.

3.2 NLP for Educational Applications

Our work is a first step towards improving in-class discussions with the help of an NLP system and, thus, to improve student learning and engagement. Prior work in understanding classroom discourse using NLP includes Suresh et al. (2019) and Donnelly et al. (2016). They propose an application where feedback can be provided to teachers by automatically classifying their utterances into talk moves. Other applications of NLP to education include language learning assistance (Beatty, 2013; Carlini et al., 2014; Tetreault et al., 2014), writing assistance (Dahlmeier and Ng, 2011; Chollampatt and Ng, 2018; Chukharev-Hudilainen and Saricaoglu, 2016), and automated scoring (Burstein et al., 1998; Farag et al., 2018; Beigman Klebanov and Madnani, 2020).

3.3 Dialogue Systems

Our work is further related to research on dialogue systems. Similar to talk moves, dialogue acts provide a categorization for utterances, but, in contrast to talk moves, they apply to general-purpose conversations (Stolcke et al., 2000; Calhoun et al., 2010). Examples include *Statement, Question, Greeting*, and *Apology*. Dialogue act tagging, which is sometimes called dialogue act prediction, is the task of classifying an utterance into the category it belongs to (Yu and Yu, 2019; Khanpour et al., 2016; Wu et al., 2020). Analogous to FTMP, future dialogue act prediction is the task of predicting what the next dialogue act should be, given a conversation history (Tanaka et al., 2019).

Pretrained models have been successfully adapted to the task of dialogue generation (Zhang et al., 2020; Wu et al., 2020; Adiwardana et al., 2020; Roller et al., 2020). However, if directly used in the classroom, these models could potentially produce harmful or unsuitable dialogue as they are trained on large datasets comprising conversations from the internet (Bender et al., 2021). Additionally, we want a system to facilitate structured conversations, and not cause further diversions – this is in contrast to many task-oriented or open-domain dialogue systems whose purpose is to entertain and appear personable to the user. Hence, we propose FTMP as a crucial first step towards an NLP system capable of facilitating classroom discussions.

4 Model

In this section, we describe our proposed model for FTMP, cf. Figure 1. Following Tanaka et al. (2019)'s model for future dialogue act prediction, its main components are three encoders. We hence name our model **3-E**. Our model predicts the next teacher talk move t_{t+1}, given the last w context elements $c_{t-w+1}, \ldots, c_t$.

Utterance Encoder The first encoder – the utterance encoder – is a single-layer gated recurrent unit (GRU; Cho et al., 2014). It processes the sequence of vector representations $v(w_1), \ldots, v(w_m)$ of the words $w_1, \ldots, w_m$ that each utterance u_i consists of and computes the last hidden state as a vector representation of u_i:

$$\hat{a}_i = \text{GRU}(v(w_1), \ldots, v(w_m)) \tag{1}$$

Each utterance representation is then concatenated with a representation s_i of the speaker role. This representation is either 1 or 0, depending on if the speaker has changed from the previous utterance:

$$a_i = \text{cat}(\hat{a}_i, s_i) \tag{2}$$

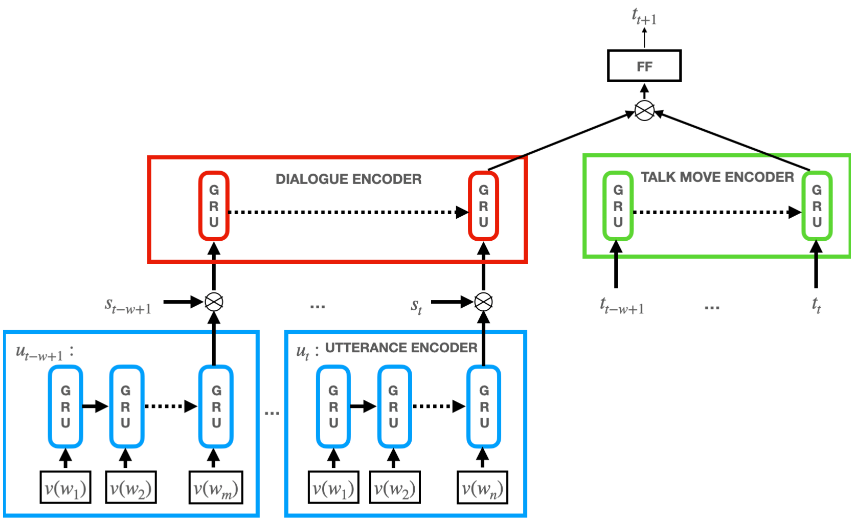

Figure 1: Our proposed model for FTMP, consisting of separate encoders for utterances, past task moves, and the overall context.

Dialogue Encoder Next, the sequence of all w utterance representations is passed to the dialogue encoder, which is also a single-layer GRU. The dialogue encoder processes the sequence, and we take the last hidden state as a representation of all utterances within our context window:

$$b_t = \text{GRU}(a_{t-w+1}, \ldots, a_t) \tag{3}$$

Talk Move Encoder The talk move encoder is a third single-layer GRU, which encodes the sequence of vector representations $v(t_{t-w+1}), \ldots, v(t_t)$ of talk moves $t_{t-w+1}, \ldots, t_t$:

$$d_t = \text{GRU}(v(t_{t-w+1}), \ldots, v(t_t)) \tag{4}$$

We obtain our final context representation r_t by concatenating the representation of all utterances and all talk moves within the context window:

$$r_t = \text{cat}(b_t, d_t) \tag{5}$$

Finally, we pass r_t through a two-layer feed-forward network and a softmax layer to obtain a probability distribution over possible future talk moves.

4.1 Adding a Pretrained Sentence Encoder

RoBERTa Pretrained models define the state of the art on a large variety of NLP tasks (Wang et al., 2019). Thus, we additionally experiment with concatenating an utterance representation computed by RoBERTa (Liu et al., 2019) to the output of 3-E's utterance encoder. Equation 2 then becomes:

$$a_i^* = \text{cat}(\hat{a}_i, s_i, \text{RoBERTa}(w_1, \ldots, w_m)) \tag{6}$$

We call the model with additional RoBERTa representations **3-E-RoBERTa**.

TOD-BERT Since there is a domain mismatch between the text that RoBERTa is trained on and our data, we further experiment with including a model trained on task-oriented dialogue, called TOD-BERT (Wu et al., 2020). TOD-BERT differentiates between user utterances and system utterances using two special tokens, [USR] and [SYS]. Correspondingly, we use the [USR] token to indicate student utterances and the [SYS] token to indicate teacher utterances. We then concatenate a context of w utterances, marked by speaker tokens when there is a change in speaker, to obtain c_{tod}. Finally, we encode c_{tod} using the pretrained TOD-BERT model and concatenate it with the output of the dialogue encoder and talk move encoder. Equation 5 then becomes:

$$r_t = \text{cat}(b_t, d_t, \text{TOD-BERT}(c_{tod})) \tag{7}$$

We call this model **3-E-TOD-BERT**. When pretrained sentence encoders are used, we use the respective BPE (Sennrich et al., 2016) tokenizer for each model.

4.2 Computing the Loss

We train all models using a cross-entropy loss. However, we observe a strong class imbalance in our training data, cf. Figure 2. Thus, we compute label weights inversely proportional to the frequency of a label's occurrence in the data and use them to weight the loss for each training example.

5 Experiments

5.1 Dataset

For our experiments, we make use of the dataset from Suresh et al. (2019). It consists of 216 annotated transcripts of classroom discourse collected in public schools in the US. The topic of instruction is mathematics. The transcripts have been collected from classes from kindergarten to grade 12 and are all in English. Each row in the transcripts consists of an utterance, the name of the speaker, and the talk move realized by this utterance.

The annotations assign each teacher utterance to one of the 6 APT talk moves described in Section 2. Utterances that do not fit into any talk move category are coded as *None*. In addition, we designate the teacher talk move corresponding to utterances made by a student as *Wait*. This category is needed as we eventually want to be able to detect when an in-class NLP system should remain quiet. The original annotations contain two additional categories that we remove due to sparsity: $Marking$ refers to repeated utterances, and we merge it with the $Restating$ category. Some student utterances are annotated as $Context$, which we merge with the $Wait$ category.

We create training, development and test data from 70%, 15%, and 15% of the available documents, respectively. Thus, we have 151 documents for training, and 32 documents for each of development and testing. Our training set consists of over 63k utterances, and the distribution of talk moves in the training set is shown in Figure 2.

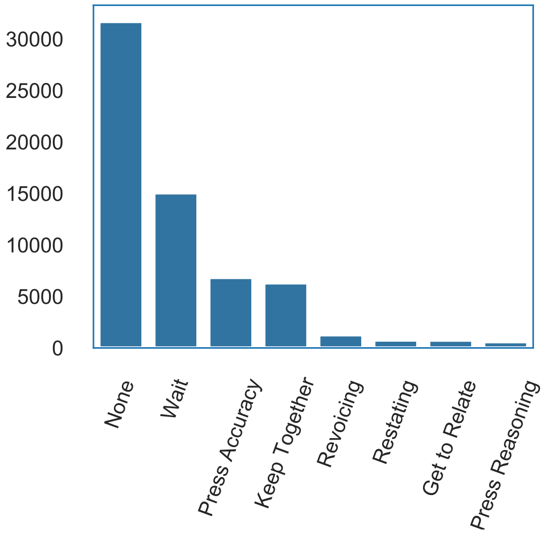

Figure 2: Distribution of talk moves in our training set.

Since 3-E's utterance encoder operates on the word level, we split each utterance into words using the NLTK word tokenizer (Loper and Bird, 2002).

5.2 Baselines

We compare our model to three baselines.

Random Baseline (RB) This baseline randomly selects one of the 8 talk moves for each input.

Talk Move Bigram Model (TMBM) For this baseline, we compute the conditional probability of every talk move in the training set, given the talk move realized by the previous utterance. We then pick the talk move with the highest conditional probability.

Talk Moves Only (TM-only) We further train a GRU model exclusively on the sequences of prior talk moves, i.e., this baseline has no access to actual utterances. We implement two variants of this baseline, one with class weights for training (TM-only-w), and one without (TM-only-z).

5.3 Metrics

Since the classes in our dataset are highly imbalanced, we do not evaluate using accuracy. Instead, we report precision, recall, and F1 score for all models. We compute F1 for all 8 classes individually, and additionally calculate macro-average F1 as an overall score for our dataset.

5.4 Results

Table 2 shows the performance of our proposed model 3-E as well as of 3-E-TOD-BERT, 3-E-RoBERTa, and all baselines. Looking at the macro-average F1 scores, we see that 3-E performs best with an F1 of 29.84. 3-E-RoBERTa, with 27.62 F1, performs worse; however, given that this model has more parameters and includes a strong pretrained component, this is an unexpected result.[2]

To reduce the domain mismatch between RoBERTa's training data and our classroom dialogue data, we substitute RoBERTa with TOD-BERT, which is also trained on dialogue. We see that, while 3-E-TOD-BERT performs better than 3-E-RoBERTa, 3-E still outperforms it. We also observe that on most individual talk move classes, 3-E-TOD-BERT performs equal to or better than 3-E. However, it does poorly on a few classes that

---

[2] We further experiment with directly finetuning RoBERTa on our task, but find its performance to be poor overall (around 17 F1). Hence, we do not report detailed results.

Model	Prec.	Recall	F1	None F1	Wait F1	Press Acc. F1	Keep Together F1	Revoicing F1	Getting Students to Relate F1	Restating F1	Press Reasoning F1
3-E	35.67	30.38	**29.84**	72.72	75.70	24.25	13.31	20.27	**3.25**	**18.45**	**10.77**
3-E-ToD-BERT	31.10	28.92	28.51	**73.05**	**77.67**	**25.18**	13.81	18.89	0.00	17.92	1.53
3-E-RoBERTa	33.81	28.04	27.62	69.89	73.34	20.48	**14.68**	20.31	1.57	17.81	2.88
TM-only-w	28.95	22.66	20.40	72.14	52.75	13.19	1.94	**21.04**	0.68	0.00	1.45
TM-only-z	18.38	18.81	16.43	71.60	52.14	0.22	0.00	7.47	0.00	0.00	0.00
RB	12.25	11.74	8.50	15.50	19.82	10.52	2.93	2.28	2.81	11.58	2.57
Majority	6.46	12.50	8.52	68.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TMBM	13.18	17.74	15.09	49.59	71.11	0.00	0.00	0.00	0.00	0.00	0.00
Annotator 1	*37.57*	*29.29*	*24.50*	*31.46*	*42.55*	*15.69*	*18.18*	*29.51*	*5.13*	*53.52*	*0.00*
Annotator 2	*38.99*	*33.16*	*30.51*	*25.19*	*54.17*	*36.17*	*20.00*	*24.66*	*21.74*	*42.62*	*9.52*
3-E	*15.83*	*20.22*	*14.57*	*29.32*	*26.53*	*4.44*	*12.66*	*8.16*	*0*	*35.48*	*0.00*

Table 3: Model and annotator performance on FTMP. Italics indicate results on a diagnostic test set of 300 examples taken from the development set.

Label	Prec.	Recall	F1
Macro average	43.76	42.65	42.44
None	68.30	77.76	72.72
Wait	71.48	80.45	75.70
Learning Community	29.59	14.10	19.10
Content Knowledge	27.86	21.47	24.25
Rigorous Thinking	21.56	19.44	20.45

Table 4: Performance of 3-E evaluated on facets.

are less prevalent in the data. We hypothesize that small changes in the quality of the utterance representations have negligible effect on our model, since it gets a large amount of information from the sequence of prior talk move labels. This hypothesis is supported by the fact that all baselines which only receive prior talk move labels as input, i.e., TM-only-w, TM-only-z and TMBM, obtain F1 scores of 20.40, 16.42, and 15.09, respectively. All of them strongly outperform a random baseline with an F1 of 8.50. Comparing 3-E to our baselines, we see that our proposed model is indeed strong on FTMP: 3-E outperforms the best baseline, TM-only-w, by 9.44 F1.

6 Analysis

6.1 FTMP on the Facet Level

In some cases, the distinctions between different talk moves are subtle. For instance, both the *Keeping Everyone Together* and *Getting Students to Relate* moves, which fall under the facet of accountability to the learning community, are made when the teacher wants the students to actively listen and respond to statements made in the classroom. To understand how well the model can distinguish between different accountability facets, we evaluate our best model, 3-E, on the facet level by binning all predicted talk moves into their corresponding facets for the computation of the F1 score.

In Table 4, we see that performance goes up by 12.60 points in this setting, indicating that 3-E is able to distinguish between labels at a coarse-grained level, but struggles with fine-grained distinctions.

6.2 Window Size and Class Weights

We further investigate the effect of weighting the loss as described in Section 4.2 and the influence of different context window sizes. Full results on the development set can be found in Table 6 in the appendix, and we provide a summary of our findings here.

Varying the window size leads to small changes in F1. For smaller window sizes of 1 and 2, F1 is slightly lower at 28.64 and 27.62. When the window size is increased to 5, F1 increases to 29.83. However, when the window size is increased further to 7, F1 drops slightly, to 29.05. We thus choose a window size of 5 to train all our models, and conclude that very large window sizes are not beneficial. We hypothesize that this might be due to the most relevant information for FTMP being

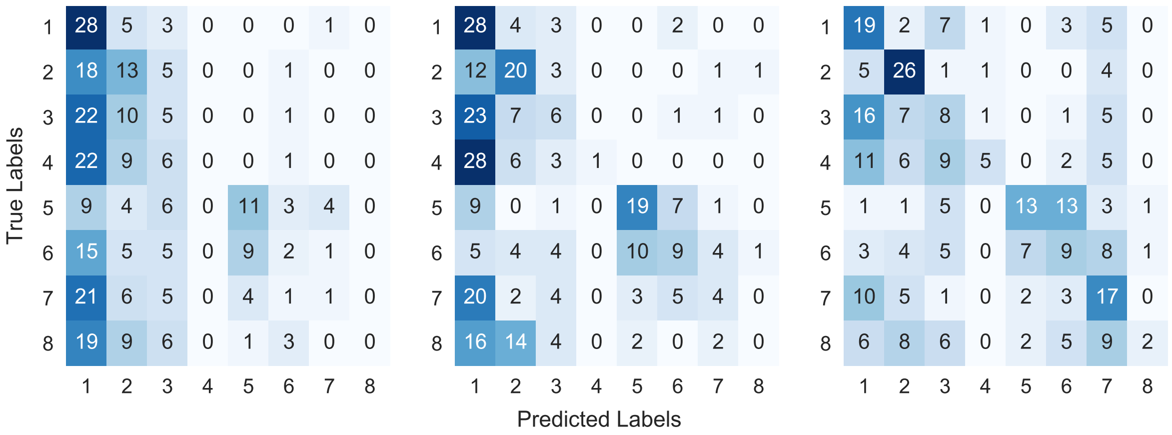

Figure 3: Confusion matrix on the diagnostic test set for 3-E (left) and our two annotators (middle and right). The talk move labels in order are: *None, Wait, Press for Accuracy, Keeping Everyone Together, Revoicing, Getting Students to Relate, Restating*, and *Press for Reasoning*.

Primary option: Annotators	
Inter-annotator agreement	46%
Annotator 1–ground truth agreement	29%
Annotator 2–ground truth agreement	33%
Both Annotators–ground truth agreement	17%
Primary option: Model	
Model–Annotator 1 agreement	48%
Model–Annotator 2 agreement	33%
Model–ground truth	20%
Acceptable options	
Annotator 1's primary accepted by Annotator 2	94%
Annotator 2's primary accepted by Annotator 1	91%
Ground truth accepted by Annotator 1	72%
Ground truth accepted by Annotator 2	79%
Model predictions accepted by Annotator 1	90%
Model predictions accepted by Annotator 2	84%

Table 5: Percentage agreement between our annotators, the ground truth, and the model's predictions on the diagnostic test set.

contained in the most recent dialogue history.

Further, class weighting during training increases 3-E's F1 score by 3, from 26.94 to 29.83. We thus, conclude that class weights are important to account for the label imbalance in our training set. [3]

6.3 Performance of Human Annotators

We further investigate (1) the difficulty of FTMP for human annotators, (2) the effect of multiple choices for the future talk move as opposed to a single answer, and (3) how annotator decisions differ from 3-E's predictions. While our gold standard are actions of a teacher, who also represent human performance, an FTMP annotator is different from a teacher since the former is presented with the exact same information as our model. In contrast, we expect a teacher's decisions to be informed by background knowledge about the students, knowledge about the content being discussed, and multi-modal information.

We recruit two annotators who have extensive experience with linguistic annotation tasks, and are familiar with talk moves. We present them with a diagnostic test set of 300 examples from the development set. Similar to the model input, each example consists of the past 5 utterances, the corresponding talk moves, and speaker information. Both annotators then provide (1) the most likely future talk move, referred to as the 'primary' option, and (2) a set of all acceptable future talk moves given the conversation history. As with our modeling setup, we consider the ground truth for the primary option to be the talk move made by the teacher in the classroom transcript. Each talk move is equally distributed in the ground truth, with 37 examples each of talk moves *None, Wait, Restating, Revoicing*, and 38 examples each of the other talk moves.

Primary Option The last 3 rows in Table 3 show the performance of our annotators' primary option and the model on the diagnostic test set. There is a significant gap of 10 F1 and 15 F1 respectively between the performance of the model and the two annotators. However, there are similarities in the class-wise breakdown. Both the annotators and the model achieve a high F1 on the classes *None, Wait,*

---

[3] We also experiment with downsampling the dominant classes and find its performance to be comparable to class weighting.

and *Restating*, and perform poorly on *Press for Reasoning* and *Getting Students to Relate*. *Press for Reasoning*, and *Getting Students to Relate* are the least prevalent classes in the data. However, the annotators' performance on these talk moves suggests that these classes are intrinsically more difficult to predict based on conversational cues alone. On the other hand, the model's poor performance on categories like *Revoicing* and *Keeping Everyone Together* in comparison to the annotators indicates that there is still room for improvement for our model.

The similarities between the model's predictions and the annotators' primary option is further illustrated by the confusion matrices in Figure 3. Both the model as well as our annotators erroneously predict *None* when the true label is another category, both confuse *Restating* and *Revoicing* for each other, and both erroneously predict *Keeping Everyone Together* when the true category is *Getting Students to Relate* or *Press for Reasoning*.

Acceptable Options Table 5 shows the percentage of responses for which the annotators agree with each other, the ground truth, and the model's predictions. On average, both annotators provide 3 acceptable options in addition to the primary – thus, roughly half the classes were viewed as acceptable for most examples. The impact of having a set of acceptable options in addition to a single correct option is evident here: while inter-annotator agreement is only around 46% on the primary option alone, the primary option of each annotator was one of the acceptable options by the other annotator in over 90% of the cases. Additionally, while agreement between the ground truth and the primary option is low with 29% and 33%, this increases to 72% and 79% when additional options are being considered.

Table 5 helps us contextualize our model's performance. Interestingly, the annotators agree with the predictions made by the model more often than they agree with the ground truth. This indicates that the model might truly be grasping overall patterns and cues from the training data, but probably struggles with finer-grained distinctions between the talk move classes. This is further substantiated by our analysis of how often the model's predictions featured in the set of acceptable options for each annotator. We find that the predictions were acceptable in 90% and 84% of all instances respectively for each annotator.

7 Conclusion

In this paper, we made use of the APT discourse framework to take a first step towards a system that can fill the role of a teacher in classroom discussions. We introduced the task of FTMP, which consists of predicting the next appropriate talk move given an in-class dialogue context. We then presented 3-E, a model for the task, which outperforms multiple baselines. Finally, we conducted an analysis of human performance on FTMP, and compared it to our model. Our results showed that, while the task is challenging, our model produces acceptable talk moves and can identify overall patterns, indicated by similarities with human performance.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful feedback. We would also like to thank the researchers involved in the NSF TalkBack project at the University of Colorado for giving us access to this data, and for their assistance with this work. We would also like to thank Amanda Howard, Charis Clevenger and Jennifer Jacobs for their feedback and assistance with the human annotations, and the members of the NALA research group for their valuable suggestions. This research was supported by the NSF National AI Institute for Student-AI Teaming (iSAT) under grant DRL 2019805. The opinions expressed are those of the authors and do not represent views of the NSF.

Ethics and Impact Statement

The data used in this paper was collected after obtaining informed consent from all participants, and this research was approved by the University of Colorado Boulder's Institutional Review Board (protocol #18-0432). All authors completed appropriate training prior to accessing the data. Since this dataset was provided to us by its owners purely for the purpose of the research reported in this paper, we are not making it publicly available here.

Further, this research is part of the NSF National AI Institute for Student-AI Teaming. Several learning scientists and educators are participating in the institute and advising us on avoiding unintended consequences and harms that could stem from NLP research. The work described in this paper is preliminary, and will be used to inform strategies for additional data collection and model design. Field trials in carefully controlled classroom settings will be conducted before wider deployment.

References

David Adamson, Gregory Dyke, Hyeju Jang, and Carolyn Penstein Rosé. 2014. Towards an agile approach to adapting dynamic collaboration support to student needs. *International Journal of Artificial Intelligence in Education*, 24(1):92–124.

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot.

Ken Beatty. 2013. *Teaching & researching: Computer-assisted language learning*. Routledge.

Beata Beigman Klebanov and Nitin Madnani. 2020. Automated evaluation of writing – 50 years and counting. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7796–7810, Online. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big.

John D Bransford, Ann L Brown, and Rodney R Cocking. 1999. *How people learn: Brain, mind, experience, and school*. National Academies Press.

Jill Burstein, Ekaterina Kochmar, Claudia Leacock, Nitin Madnani, Ildikó Pilán, Helen Yannakoudakis, and Torsten Zesch. 2020. Proceedings of the fifteenth workshop on innovative use of nlp for building educational applications. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated scoring using a hybrid feature identification technique. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 206–210.

Sasha Calhoun, Jean Carletta, Jason M Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The nxt-format switchboard corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language resources and evaluation*, 44(4):387–419.

Roberto Carlini, Joan Codina-Filba, and Leo Wanner. 2014. Improving collocation correction by ranking suggestions using linguistic knowledge. In *Proceedings of the third workshop on NLP for computer-assisted language learning*, pages 1–12.

Courtney B Cazden. 1988. *Classroom discourse: The language of teaching and learning.* ERIC.

Suzanne H. Chapin. 2003. Classroom discussions: Using math talk to help students learn, grades 1-6.

Suzanne H Chapin, Catherine O'Connor, Mary Catherine O'Connor, and Nancy Canavan Anderson. 2009. *Classroom discussions: Using math talk to help students learn, Grades K-6*. Math Solutions.

Michelene TH Chi and Ruth Wylie. 2014. The icap framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49(4):219–243.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Evgeny Chukharev-Hudilainen and Aysel Saricaoglu. 2016. Causal discourse analyzer: Improving automated feedback on academic esl writing. *Computer Assisted Language Learning*, 29(3):494–516.

Paul Cobb. Supporting the improvement of learning and teaching in social and institutional context. *Cognition and instruction: Twenty-five years of progress*, 78:19–37.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with l1-induced paraphrases. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 107–117.

Patrick J. Donnelly, Nathan Blanchard, Borhan Samei, Andrew M. Olney, Xiaoyi Sun, Brooke Ward, Sean Kelly, Martin Nystran, and Sidney K. D'Mello. 2016. Automatic teacher modeling from live classroom audio. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, UMAP '16, page 45–53, New York, NY, USA. Association for Computing Machinery.

Gregory Dyke, Iris Howley, David Adamson, Rohit Kumar, and Carolyn Penstein Rosé. 2013. Towards academically productive talk supported by conversational agents. In *Productive multivocality in the analysis of group interactions*, pages 459–476. Springer.

Youmna Farag, Helen Yannakoudakis, and Ted Briscoe. 2018. Neural automated essay scoring and coherence modeling for adversarially crafted input. *arXiv preprint arXiv:1804.06898*.

Claude Goldenberg. 1992. Instructional conversations: Promoting comprehension through discussion. *The Reading Teacher*, 46(4):316–326.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021, Osaka, Japan. The COLING 2016 Organizing Committee.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diane Litman. 2016. Natural language processing for enhancing teaching and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Danielle S McNamara. 2011. Measuring deep, reflective comprehension and learning strategies: challenges and successes. *Metacognition and Learning*, 6(2):195–203.

Danielle S McNamara, Scott A Crossley, and Rod Roscoe. 2013. Natural language processing in an intelligent writing strategy tutoring system. *Behavior research methods*, 45(2):499–515.

Sarah Michaels and Catherine O'Connor. 2015. Conceptualizing talk moves as tools: Professional development approaches for academically productive discussion. *Socializing intelligence through talk and dialogue*, pages 347–362.

Sarah Michaels, Catherine O'Connor, and Lauren B Resnick. 2008. Deliberative discourse idealized and realized: Accountable talk in the classroom and in civic life. *Studies in philosophy and education*, 27(4):283–297.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Catherine O'Connor and Sarah Michaels. 2019. Supporting teachers in taking up productive talk moves: The long road to professional learning at scale. *International Journal of Educational Research*, 97:166–175.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M. Smith, Y-Lan Boureau, and Jason Weston. 2020. Recipes for building an open-domain chatbot.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Abhijit Suresh, Tamara Sumner, Jennifer Jacobs, Bill Foland, and Wayne Ward. 2019. Automating analysis and feedback to improve mathematics teachers' classroom discourse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9721–9728.

Koji Tanaka, Junya Takayama, and Yuki Arase. 2019. Dialogue-act prediction of future responses based on conversation history. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 197–202, Florence, Italy. Association for Computational Linguistics.

Stergios Tegos, Stavros Demetriadis, and Anastasios Karakostas. 2015. Promoting academically productive talk with conversational agent interventions in collaborative learning settings. *Computers & Education*, 87:309–325.

Stergios Tegos, Stavros Demetriadis, Pantelis M Papadopoulos, and Armin Weinberger. 2016. Conversational agents for academically productive talk:

A comparison of directed and undirected agent interventions. *International Journal of Computer-Supported Collaborative Learning*, 11(4):417–440.

Joel Tetreault, Martin Chodorow, and Nitin Madnani. 2014. Bucking the trend: improved evaluation and annotation practices for esl error detection systems. *Language Resources and Evaluation*, 48(1):5–31.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929, Online. Association for Computational Linguistics.

Dian Yu and Zhou Yu. 2019. MIDAS: A dialog act annotation scheme for open domain human machine spoken conversations. *CoRR*, abs/1908.10023.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

A Tuning of the Window Size

Configuration	Prec.	Recall	F1	Acc.
No weighting, window 5	32.31	25.67	26.94	63.44
Class weighting, window 5	34.84	29.39	29.83	62.24
Class weighting, window 1	31.17	29.18	28.64	62.61
Class weighting, window 2	34.11	29.44	27.62	62.85
Class weighting, window 3	29.83	29.29	29.09	58.07
Class weighting, window 4	29.72	28.57	28.51	61.65
Class weighting, window 6	32.55	30.56	28.95	63.76
Class weighting, window 7	33.21	28.64	29.05	58.27

Table 6: Tuning experiments on the development set

B Training and Hyperparameters

3-E is implemented in PyTorch (Paszke et al., 2019). For training, we use an Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of $1e^{-4}$ and train for 30 epochs. The utterance encoder has an embedding size of 256, and a hidden layer size of 512. The talk move encoder has an embedding size of 32 and a hidden layer size of 64. The dialogue encoder has a hidden layer size of 1025. Finally, the feedforward layer uses a hidden layer size of 32. We do not use pretrained word embeddings in the 3-E model.

For the 3-E-RoBERTa model, we use the pretrained parameters of the Fairseq library's implementation of RoBERTa (Ott et al., 2019), and use representations with dimension 1024. For 3-E-RoBERTa, we further add dropout with a probability of 0.4 in two places to avoid over-fitting: on the layer where the two utterance representations are concatenated (c.f. Equation 6), and the layer where the utterance history and talk move history are concatenated (c.f. Equation 5).

For 3-E-TOD-BERT, we use the pretrained model provided by Wu et al. (2020) trained jointly on masked language modeling and response contrastive loss. We additionally add a dropout of 0.2 at the layer where the utterance representation is concatenated with talk move history (c.f. Equation 7).

The baseline models TM-only-w and TM-only-z are trained for 30 epochs using the same hyperparameters, with a batch size of 256 and a learning rate of 1e-4.

BioGen: Generating Biography Summary under Table Guidance on Wikipedia

Shen Gao [♠,*], **Xiuying Chen** [♠♣,*], **Chang Liu** [♠♣], **Dongyan Zhao** [♠♣♡,†], **Rui Yan** [◇]
[♠]Wangxuan Institute of Computer Technology, Peking University, Beijing, China
[♣]Center for Data Science, Peking University, Beijing, China
[◇]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[♡]State Key Laboratory of Media Convergence Production Technology and Systems
`{shengao, xy-chen, liuchang97, zhaody}@pku.edu.cn`
`ruiyan@ruc.edu.cn`

Abstract

Capturing the salient information from an input article has been a long-standing challenge for summarization. On Wikipedia, most of the wiki pages about people contain a factual table that lists the basic properties of the people. Illuminatingly, a factual table can be regarded as a natural summary of the key information in the corresponding article. Thus, in this paper we propose the task of table-guided abstractive biography summarization, which utilizes factual tables to capture important information and then generate a summary of a biography. We first introduce the TaGS (Table-Guided Summarization) dataset[1], the first large-scale biography summarization dataset with tables. Next, we report some statistics about this dataset to validate the quality of the dataset. We also benchmark several commonly used summarization methods on TaGS and hope this will inspire more exciting methods.

1 Introduction

Text summarization generates a short text version of a long passage which retains the most important information. Recently, two kinds of approaches have been proposed for automatic text summarization. One is extractive summarization (Nallapati et al., 2017; Liu and Lapata, 2019), which directly selects salient sentences from the passage to create a summary. The other is abstractive summarization (See et al., 2017; Hsu et al., 2018a), which aims to concisely paraphrase the input article. In both methods, the summary should always focus on important information, though a document may include trivial facts.

To focus on the main information when generating summaries, some researchers propose to incorporate manifold information to improve the performance. Narayan et al. (2017) proposed to incorporate the figures and Gao et al. (2019b) investigated the using of reader comments for more effective summarization. As another type of side information, factual tables provide a natural summary of the biography document. On Wikipedia, in each wiki page about people, there is a factual table (infobox) on the right side of the page summarizing the main properties. Clearly, infobox is helpful for capturing the salient information during summarizing the biography. However, no existing work takes advantage of tables, though are widely available in the biography on Wikipedia.

In this paper, we propose *Table-Guided Summarization* (TaGS) dataset, the first large-scale biography summarization dataset with tables. And we report some statistics and three important characteristics of this dataset to verify its quality. The first one is it has the weak lead bias that makes it suitable for training both abstractive and extractive summarization methods. The second one is it has strong abstractness that is helpful for generating a more condensed summary. The most important characteristic is that the summary of the biography is guided by a table which contains the most salient facts described in the biography.

To verify the quality of this dataset, we employ some commonly used state-of-the-art summarization methods to conduct experiments on our proposed dataset. From these experimental results, we can see that the methods which simply incorporate the table information outperform the methods which do not use the table information. That demonstrates the effectiveness of incorporating table guidance when generating summaries of documents which have a factual table in it.

Our contributions are summarized as follows:

---

[*] Equal contribution. Ordering is decided by a coin flip.

[†] Corresponding Author: Dongyan Zhao

[1] `https://github.com/gsh199449/table-summ`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4752–4757
August 1–6, 2021.

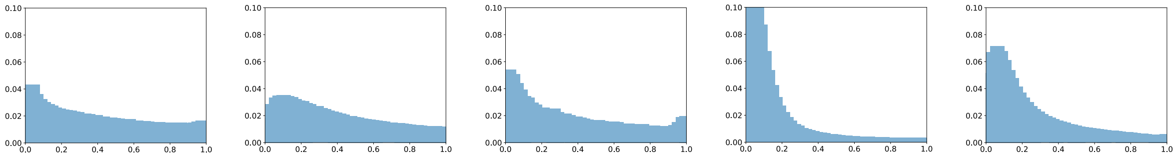

(a) TaGS (b) XSUM (c) Reddit (d) Newsroom (e) CNN/DailyMail

Figure 1: Relative locations of bigrams of ground truth summaries in the source text across different datasets.

- To the best of our knowledge, we are the first to use factual tables to guide the summarization procedure so as to generate better summaries.
- We release a large-scale abstractive biography summarization dataset with tables. Experiments conducted on this dataset demonstrate the effectiveness of incorporating table information in generating summaries.

2 Related Work

2.1 Text Summarization

Text summarization is an important task which can be classified into extractive and abstractive approaches. Extractive summarization (Narayan et al., 2018b; Chen et al., 2018; Jadhav and Rajan, 2018) tends to generate a summary by integrating the most salient sentences in the document. Cheng and Lapata (2016) first propose using recurrent neural network (RNN) to extract salient sentences. After that researchers explore many the neural based method (Nallapati et al., 2017; Liu and Lapata, 2019; Chen et al., 2018; Zhang et al., 2018; Zhou et al., 2018; Liu et al., 2019), and achieve the state-of-the-art performance (Liu and Lapata, 2019) on the benchmark dataset CNN/DailyMail. In the mean time, the Nallapati et al. (2016) firstly apply this text generation method to the abstractive summarization task and Gehrmann et al. (2018) achieve the state-of-the-art performance by using a data-efficient content selector.

2.2 Summarization with Side Information

Traditional text summarization methods only use the document as input. However, the gist of the document may lie in side information, such as the title, image captions, or comments which are often available for news-wire articles. As such, various studies (Gao et al., 2020; Hu et al., 2008) have tried to use such information for more efficient and accurate summarization. However, to the best of our knowledge, no existing works consider the use of tables to guide biography summarization.

3 TaGS Dataset

Our dataset, named Table-Guide Summarization (TaGS), consists of over 500,000 document-summary pairs, along with their corresponding factual tables collected from Wikipedia. Concretely, following Chen et al. (2019), we use the leading paragraphs before the content outline as the summary, and following paragraphs as the document. The infobox in the right part of the webpage is extracted as the guided table.

Some key statistics of the factual table are described below. 7.31% of words from a document and 29.41% of words from a summary are included in a factual table. The average number of fields in a table is 12.89, and there are 46.83 words in a table in average. We show some detailed statistics of document-summary pairs in TaGS and compare them with other popular text summarization datasets in Table 1. We next discuss some abstractive characteristics of TaGS compared to existing summarization datasets.

Weak Lead Bias. Lead bias means that directly using the leading sentences of a document can produce a good performance in terms of the summarization evaluation metric ROUGE (Lin, 2004). This is a common problem in text summarization datasets, which mostly occurs in news-based documents. Figure 1 plots the density histograms for the relative locations of words from the ground truth summary in the input document. In the CNN/DailyMail and Newsroom datasets, the words are highly concentrated at the leading parts of the input document. In contrast, our TaGS dataset shows more uniform distributions across words in the document. This characteristic can be also found from the `LEAD` score shown in Table 2, where `LEAD` is a baseline method that selects the first few sentences in the input document as the summary. A high `LEAD` score implicitly indicates a strong lead bias. From Table 2, we find that TaGS has a much lower `LEAD` score than the CNN/DailyMail and Newsroom datasets, and

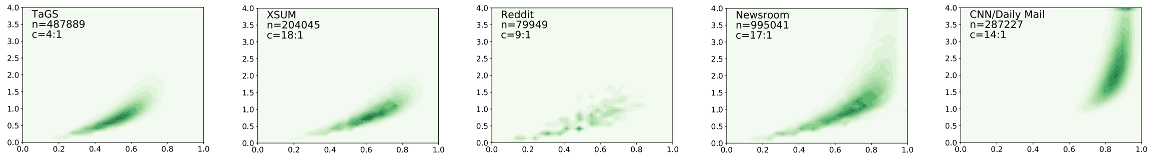

(a) TaGS (b) XSUM (c) Reddit (d) Newsroom (e) CNN/DailyMail

Figure 2: Density estimation of extractive diversity scores. Large variability along the y-axis suggests variation in the average length of source sequences present in the summary, while x-axis shows variability in the average length of extractive fragments to which summary words belong.

Table 1: Comparison of summarization datasets with respect to corpus size, average document (source) and summary (target) length (in terms of words and sentences) on both source and target.

Datasets	# docs (train/val/test)	avg. document length		avg. summary length	
		words	sentences	words	sentences
CNN/DailyMail (Hermann et al., 2015)	287,227/13,368/11,490	810.57	39.78	56.20	3.68
XSum (Narayan et al., 2018a)	204,045/11,332/11,334	431.07	19.77	23.26	1.00
Reddit (Kim et al., 2019)	79,949	342.00	18.02	9.33	1.03
Newsroom (Grusky et al., 2018)	995,041/108,837/108,862	658.60	31.43	26.70	1.42
TaGS	487,889/5,000/5,000	226.13	7.81	49.25	1.88

thus prevents the model from directly learning the salient information by locational bias.

Strong Abstractness. Table 2 reports the percentage of novel n-grams in the ground truth summary that do not appear in the input document. The result shows that our dataset comprises of 43.38% novel unigrams in the summary, 122.46% higher than the commonly-used benchmark dataset CNN/DailyMail. This indicates that summaries in TaGS are more abstractive. Besides, other two metrics, density and coverage, proposed by Grusky et al. (2018), are commonly used when evaluating the summarization dataset (Kim et al., 2019; Grusky et al., 2018). We plot the distributions of these two metrics in Figure 2, where small density and coverage reflects the summary has strong abstractness. The result shows that TaGS is similar to XSUM and Reddit in terms of density and coverage, and these datasets all have strong abstractness.

`PG/LEAD` in Table 2 is the ROUGE-L ratios of `PG` to `LEAD`, which quantifies the difficulty for extractive methods and the suitability for abstractive methods. CNN/DailyMail and Newsroom achieve low `PG/LEAD` scores, demonstrating that these datasets are more suitable for extractive based model. On the contrary, XSUM and TaGS have high `PG/LEAD`, showing that TaGS is potentially an excellent benchmark for evaluation of abstractive summarization systems.

Table Guided. The intrinsic difference between TaGS and other datasets lies in that each document-summary pair in TaGS is associated with a factual table. We judge whether the table is good guidance from two aspects: (1) word-level overlap percentage and (2) sentence-level mapping relation between document and table. From the word-level aspect, 7.31% words in the document are included in the table; from the sentence-level aspect, we found that 99.86% of document sentences have words in common with the factual table. This demonstrates that the table captures the majority of facts in the document. Thus, it is safe to conclude that the tables in TaGS can be utilized as a good guidance.

4 Experimental Setup

4.1 Comparison Methods

To evaluate the effectiveness of incorporating table, we conduct experiments using the following baselines: (1) **LEAD3**: selects the first three sentences of a document as the summary. (2) **S2S**: is the traditional sequence-to-sequence framework in (Sutskever et al., 2014) which has been used in many text generation tasks (Gao et al., 2019c, 2021; Chan et al., 2019a, 2020, 2019b). (3) **PG**: combines `S2S` with copy mechanism in See et al. (2017). (4) **Unified**: proposed by Hsu et al. (2018b), combines the strength of extractive and abstractive summarization. (5) **Transformer**: is solely based on attention mechanism

Table 2: Corpus bias towards extractive methods in datasets. We show the proportion of novel n-grams in gold summaries. We also report ROUGE scores for the LEAD baseline and the abstractive summarization method PG. Results are computed on the test set.

Datasets	% of novel n-grams in gold summary				LEAD			PG			PG/LEAD
	unigram	bigrams	trigrams	4-grams	R1	R2	RL	R1	R2	RL	Ratio (R-L)
CNN/DM	19.50	56.88	74.41	82.83	39.60	17.70	36.20	36.4	15.7	33.4	0.92x
XSum	35.76	83.45	95.50	98.49	16.30	1.61	11.95	29.7	9.2	23.2	1.93x
Reddit	48.97	84.42	94.66	97.82	3.40	0.00	3.30	19.0	3.7	15.1	5.59x
Newsroom	19.53	48.39	59.38	64.06	30.50	21.30	28.40	14.7	2.2	10.3	0.92x
TaGS	43.38	84.30	94.09	97.11	23.10	4.80	19.57	29.3	11.2	27.9	1.43x

proposed in Vaswani et al. (2017). (6) **CopyTransformer**: is a state-of-the-art generative summarization model (Gehrmann et al., 2018), which combines the `Transformer` with copy mechanism. (7) **TabWords**: just concatenates all the words in table as the summary.

Additionally, we select two best baselines, `Unified` and `CopyTransformer`, concatenating the original input document with tables as input, denoted as (9) **Unified+T** and (10) **CopyTransformer+T**, to determine whether the improvement of TGSG simply arises from the table information.

4.2 Evaluation Metrics

For evaluation metrics, we adopt the ROUGE scores (Lin, 2004) which is widely applied for summarization evaluation (Sun et al., 2018; Chen et al., 2018; Gao et al., 2019a). The ROUGE metrics compare the generated summary with the reference summary by computing overlapping lexical units, and include ROUGE-1, ROUGE-2 and ROUGE-L.

5 Experimental Results

We first examine the performance of these baselines, as shown in Table 3. Firstly, among models without table information, `Unified` achieves the highest performance. `PG` and `CopyTransformer` achieve the second best performance. Secondly, tables are indeed helpful for the summarization process. For models with additional table information, the ROUGE-1 score of `Unified+T` and `CopyTransformer+T` improves by 4.22 and 14.3, respectively. This observation demonstrates that factual tables can help the summarization model to capture the main idea of the table by emphasizing the key facts in the document. However, the performance of `TabWords` is much lower than `Unified+T` and `CopyTransformer+T`, which demonstrates only using the information from table is not sufficient for generating a good summary.

Table 3: ROUGE scores comparison between baselines and TGSG. All our ROUGE scores have a 95% confidence interval of at most ±0.23 as reported by the official ROUGE script.

	ROUGE-1	ROUGE-2	ROUGE-L
w/o table information			
LEAD3	23.10	4.80	19.57
S2S	26.31	10.07	25.13
PG	29.33	11.24	27.91
Unified	32.23	13.29	29.36
Transformer	27.67	14.87	27.02
CopyTransformer	29.17	16.15	28.43
with table information			
TabWords	30.43	12.58	23.73
Unified+T	36.45	15.80	32.93
CopyTransformer+T	43.47	29.14	42.49

6 Conclusion

In this paper, we proposed to use factual tables to guide biography summarization. To demonstrate the effectiveness of incorporating table information in generating biography summaries, we developed the first large-scale abstractive biography summarization dataset with tables. We employ several state-of-the-art summarization methods, and adapt these methods to table guided biography summarization task. These methods outperformed other summarization methods in terms of ROUGE which only use the document as input and ignore the table guidance.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2020AAA0106600), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058). Rui Yan is partially supported as a Young Fellow of Beijing Institute of Artificial Intelligence (BAAI).

Ethical Impact

In this paper, we propose a table guided biography summarization dataset on Wikipedia. In the real-world application which provided automatic biography summarization service, we will employ human editors to double-check the generated summary to ensure the correctness of content and grammar before publish the summary.

References

Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao, and Rui Yan. 2019a. Stick to the facts: Learning towards a fidelity-oriented e-commerce product description generation. In *EMNLP*.

Zhangming Chan, Juntao Li, Xiaopeng Yang, Xiuying Chen, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019b. Modeling personalization in continuous space for response generation via augmented wasserstein autoencoders. In *EMNLP*.

Zhangming Chan, Yuchi Zhang, Xiuying Chen, Shen Gao, Zhiqiang Zhang, Dongyan Zhao, and Rui Yan. 2020. Selection and generation: Learning towards multi-product advertisement post generation. In *EMNLP*.

Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. 2019. Learning towards abstractive timeline summarization. In *IJCAI*.

Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao, and Rui Yan. 2018. Iterative document representation learning towards summarization with polishing. In *EMNLP*, pages 4088–4097.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494.

Shen Gao, Xiuying Chen, Piji Li, Zhangming Chan, Dongyan Zhao, and Rui Yan. 2019a. How to write summaries with patterns? learning towards abstractive summarization through prototype editing. In *EMNLP*.

Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. 2019b. Abstractive text summarization by incorporating reader comments. In *AAAI*.

Shen Gao, Xiuying Chen, Zhaochun Ren, and Dongyan Zhao Rui Yan. 2020. From standard summarization to new tasks and beyond: Summarization with manifold information. In *IJCAI*.

Shen Gao, Xiuying Chen, Zhaochun Ren, Dongyan Zhao, and Rui Yan. 2021. Meaningful answer generation of e-commerce question-answering. *ACM Trans. Inf. Syst.*, 39(2).

Shen Gao, Zhaochun Ren, Yihong Eric Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019c. Product-aware answer generation in e-commerce question-answering. In *WSDM*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *EMNLP*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL-HLT*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, J. J. Tang, and Min Sun. 2018a. A unified model for extractive and abstractive summarization using inconsistency loss. In *ACL*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018b. A unified model for extractive and abstractive summarization using inconsistency loss. *ACL*.

Meishan Hu, Aixin Sun, and Ee-Peng Lim. 2008. Comments-oriented document summarization: understanding documents with readers' feedback. In *SIGIR*.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *EMNLP*, pages 142–151.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive summarization of reddit posts with multi-level memory networks. In *NAACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *EMNLP*.

Yang Liu, Ivan Titov, and Mirella Lapata. 2019. Single document summarization as tree induction. In *NAACL-HLT*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*.

Shashi Narayan, Nikos Papasarantopoulos, Mirella Lapata, and Shay B. Cohen. 2017. Neural extractive summarization with side information. *CoRR*, abs/1704.04530.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Min Sun, Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, and Jing Tang. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *EMNLP*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. *ArXiv*, abs/1807.02305.

Multilingual Simultaneous Neural Machine Translation

Philip Arthur[†*] **Dongwon K. Ryu**[‡*] **Gholamreza Haffari**[‡]
[†]Oracle Digital Assistant, Oracle
[‡]Department of Data Science and AI, Monash University
philip.arthur@oracle.com
{dongwon.ryu, gholamreza.haffari}@monash.edu

Abstract

Simultaneous machine translation (SIMT) involves translating source utterances to the target language in real-time before the speaker utterance completes. This paper proposes the *multilingual* approach to SIMT, where a single model simultaneously translates between multiple language-pairs. This not only results in more efficiency in terms of the number of models and parameters (hence simpler deployment), but may also lead to higher performing models by capturing commonalities among the languages. We further explore simple and effective multilingual architectures based on two strong recently proposed SIMT models. Our results on translating from two Germanic languages (German, Dutch) and three Romance languages (French, Italian, Romanian) into English show (i) the single multilingual model is on-par or better than individual models, and (ii) multilingual SIMT models trained based on language families are on-par or better than the universal model trained for all languages.[1]

1 Introduction

Simultaneous translation is the task of incrementally generating the translation while the source utterance is gradually spoken. It is crucial in multinational meetings, e.g., in business and politics, where the online simultaneous translation is required for one or multiple language-pairs. Simultaneous machine translation (SIMT) is an attempt to address the challenges of this translation scenario, i.e., trading off the translation quality and its latency (Cho and Esipova, 2016; Arivazhagan et al., 2019, 2020; Firat et al., 2016b).

In this paper, we investigate the *multilingual* SIMT setting, where a single model simultaneously translates between multiple language-pairs. This not only results in more efficiency in terms of the number of models and parameters (hence simpler deployment), but may also lead to higher performing models by capturing commonalities among the languages. The multilingual setting has been successful for the standard offline neural machine translation (NMT) and studied extensively (Johnson et al., 2017; Tan et al., 2019; Aharoni et al., 2019).

We explore simple and effective multilingual architectures based on two strong recently proposed SIMT models, i.e. the WAIT-K (Dalvi et al., 2018) and COUPLED POLICY (Arthur et al., 2020). The former waits to *read* a fixed number of k input tokens; afterward, it *writes* (generates) an output token for each newly received input token. The latter learns a policy, via an *agent*, for an adaptive waiting between reading and writing to reduce the translation delay while maintaining the quality.

COUPLED POLICY uses the adaptive waiting, generated from offline word alignments. It continues to read the source tokens until the corresponding word alignment to the target token appears.

Under these underlying SIMT models, we explore multi-task learning (MTL) framework, *full* and *partial* parameter sharing protocols across the languages with language indicators.

Our experiments show the effectiveness of the simple strategy of sharing all the SIMT components across the languages, with language tags specifying the translation task. The results on translating from two Germanic languages (German, Dutch) and three Romance languages (French, Italian, Romanian) into English show the single multilingual model is on-par or better than individual models. Furthermore, the results show that multilingual SIMT models trained based on language families are on-par or better than the universal model trained for all languages.

---

[1] Star (*) marks a shared first authorship between Philip and Dongwon, where both contributed equally. This work was done when Philip was a research fellow at Monash University.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4758–4766
August 1–6, 2021.

Algorithm 1 Training Multilingual NPI-SiMT

```
Require: $\mathcal{D}$: Collections of parallel corpora with oracle actions.
while a stopping condition is not met do
    for $\mathcal{D}_i \in \mathcal{D}$ do
        $F, E$ is a language pair of $\mathcal{D}_i$.
        for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) \in \mathcal{D}_i$ do
            $\theta_{\mathcal{A}}^{F,E} \leftarrow \text{MLE}(\boldsymbol{a}, \theta_{\mathcal{A}}^{F,E})$
            $\theta_{\mathcal{E}}^{F}, \theta_{\mathcal{D}}^{E} \leftarrow \text{MLE}(\boldsymbol{x}, \boldsymbol{y}, \theta_{\mathcal{E}}^{F}, \theta_{\mathcal{D}}^{E})$
        end for
    end for
end while
```

2 Multilingual Simultaneous Translation

The original neural programmer-interpreter (NPI)-SiMT framework (Arthur et al., 2020) employs a trainable programmer θ_{prog} and interpreter θ_{intp}. The programmer/agent issues *read*/*write* commands to control the interpreter, i.e. the NMT model. The interpreter is constructed using an encoder $\theta_{\mathcal{E}}$ and a decoder $\theta_{\mathcal{D}}$. Each component is trained on triplet $\langle x, y, a \rangle$ where x is the source sentence, y is the target sentence, and a is the program oracle using behavioral cloning (Torabi et al., 2019). For notation clarity we rename the programmer into $\theta_{\mathcal{A}}$, resulting triplet of trainable modules $\theta_{\mathcal{A}}, \theta_{\mathcal{E}}, \theta_{\mathcal{D}}$.

Language-Specific Parameters We further extend this framework by distilling a parameter, specific to language θ_x^l, where x is a specific module and l is a specific language. This language-specific parameter is similar to Firat et al. (2016a); Dong et al. (2015); Ahmadnia and Dorr (2020) where parameters are separated based on the source and target languages. In the case of SiMT, the program a is affected by both languages. This framework enables us to use multiple parallel corpora $\mathcal{D}_i$ and train a language specific module using maximum likelihood estimation by updating particular θ_x^l based on $\mathcal{D}_i$. The training algorithm for our NPI-SiMT is shown in Algorithm 1.

Multilingual Parameter Sharing Multilingual parameter sharing is achieved by using only a single module for language-specific parameter θ_x^*. Depending on the module, we can disregard source (F) or target (E) completely. This allows us to share the parameter across different parallel corpora. However, the embedding matrix in different $\mathcal{D}_i$ can be different because of various tokenization and vocabulary construction methods. To remedy this, we can either train joint vocabulary spaces for source and target sides, or simply joining different spaces using union operation. Herein, we use the latter method.

Language Indicator Embedding When the interpreter is shared, it is difficult to communicate which pairs of languages are being processed. To deliver this, we pass the source and target language embedding information to the encoder and decoder, respectively. This information is then combined using addition operation with the word embedding. In the programmer, we use a concatenation of both source and target languages.

Batch of Multilingual Instances Algorithm 1 outlines the overall training procedure of multilingual SiMT. Here, it is crucial to construct a batch as a mixture of many language pairs to achieve good multilingual training. We also need to include the information of the source language to create language indicator embedding. If a module is language-agnostic, it will be responsible for consuming all the input; otherwise, language-specific modules will be used to process the specific item in the batch according to its language. Results from different languages will be aggregated using concatenation at the end.

3 Experiments

Our experiment aims to investigate the effects of multilingualism in SiMT architecture. To achieve this we first choose the language pairs from (1) the same family group and (2) mixing them all. This is enabled by investigating various parameter sharing strategies for the components of the SiMT architectures.

Datasets. We use IWSLT 2017 (Cettolo et al., 2017) datasets for all parallel corpora in which all translating to English. The choice was made due to its characteristics of spoken multilingual corpora from TED. We choose the Germanic language group, German (DE) and Dutch (NL), and the Romance language group, Italian (IT), French (FR), and Romanian (RO). The languages within the same group generally have high syntactic similarity and the same word order. Unless otherwise specified, we use the same settings and preprocessing as described in Arthur et al. (2020).[2]

---

[2] In our preliminary experiment, the pre-processing under concatenation of multilingual corpora with larger vocabulary

SIMT Systems We compared two SIMT baselines, COUPLED POLICY (Arthur et al., 2020) and the WAIT-K model (Ma et al., 2018). For a fair comparison, we choose a value of k, which achieves comparable translation quality to the COUPLED POLICY system. Following some initial experiments, we choose $k = 2$.

Parameter Sharing Since our model deals with the many-to-one translation task with an agent, we decided to separate i) encoder, ii) agent, and iii) encoder + agent. This idea came from the performance improvements that a number of studies demonstrated by separating the decoder in offline one-to-many MT (Dong et al., 2015; Sachan and Neubig, 2018). In SIMT, two modules, encoder and agent, are tied to the source, and therefore, reasonable to have them as language-specific parameters.

Evaluation. Following Arthur et al. (2020), we evaluate the systems based on their translation quality and delay. Translation quality can be measured by case sensitive BLEU (Papineni et al., 2002). [3] We adopt two delay measurements by previous studies: (1) average proportion (AP) (Cho and Esipova, 2016) is a fraction of reading source words per emitted target words, and (2) average lagging (AL) (Ma et al., 2019) is an average number of lagged source words until all inputs are read.

3.1 Results

In this section, we will describe the results of parameter sharing in SIMT. Following that, we present the comparison of multilingualism under different language groups.

Parameter Sharing Strategies. Table 1 presents the results of various parameter sharing strategies for FR/IT/RO in the Romance language family. When sharing all parameters across these three languages, WAIT-2 has a slight increase in delay, but the translation quality is comparable to or better than bilingual. In contrast, the best parameter sharing setting for COUPLE POLICY is to have language-specific encoders and share the rest of the parameters. This appears to have a clear advantage in both quality and delay; the BLEU score increases up to 0.8 units, with a reduction in AL, approximately 10% in FR and IT. In both architectures, the model size reduces drastically when trained on multilingual setting, and remains approximately the same across different sharing strategies. These results are consistent for DE/NL in the Germanic language family. Full results are included in the supplementary material.

			FR→EN		IT→EN		RO→EN		Model
$\theta_{\mathcal{E}}$	$\theta_{\mathcal{D}}$	$\theta_{\mathcal{A}}$	AL	BLEU	AL	BLEU	AL	BLEU	Size
				WAIT-2					
×	×	–	**2.22**	23.40	**2.73**	24.77	**2.55**	24.34	158.7M
×	✓	–	2.32	23.54	2.77	24.75	2.68	**24.43**	88.2M
✓	✓	–	2.35	**23.76**	2.77	**25.24**	2.72	24.10	84.1M
				COUPLED POLICY					
×	×	×	1.48	26.90	1.00	22.62	1.02	22.82	166.9M
×	✓	×	1.53	24.82	0.90	21.13	1.09	20.60	96.4M
×	✓	✓	**1.37**	**27.45**	0.92	**23.40**	**1.01**	**23.35**	90.9M
✓	✓	×	1.44	27.36	0.99	23.13	1.16	23.00	92.4M
✓	✓	✓	1.38	27.34	**0.89**	23.32	1.02	22.63	86.9M

Table 1: Parameter sharing under Romance language family: French, Italian, Romanian. ✓ and × indicate *shared* and *not sharing* components in the MTL architecture.

Multilingual Modelling Strategies. Table 2 shows the overall performance comparison of the multilingual setting. Multilingualism in SIMT evidently surpasses the bilingual baseline in translation delay, quality, and/or model size. Generally, SIMT trained on the same language family outperforms not only the bilingual baselines, but also the universal multilingual model. In the Germanic language, training under the same language group boosts up the BLEU up to 1 unit. Although baseline in WAIT-2 has a shorter delay, COUPLED POLICY surpasses both quality and delay. We observe that when the model runs universally, the BLEU score reaches back to or lower than that of the bilingual model.

On the other hand, the Romance language family has slightly different behavior across different SIMT models. COUPLED POLICY behaves similarly, where training SIMT under the same group positively influences the performance, but in WAIT-2, the universal model excels the best. This is particularly interesting because the Romance language family has the same word order as English, which WAIT-2 would be a perfect fit for such translation between two languages with the same word order. However, it is not the case, so mixing all the languages regardless of word order under WAIT-2 improves translation quality more while preserving the delay.

---

size does not impact performance.

[3] Calculated using sacrebleu (Post, 2018). BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.4.4 is our `sacreblue`'s signature

	DE→EN			NL→EN			FR→EN			IT→EN			RO→EN			
	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU	Model Size
WAIT-2																
Bilingual	0.62	**2.54**	22.99	0.63	**2.39**	28.33	0.60	2.22	23.40	0.63	2.73	24.77	0.63	2.55	24.34	264.5M
Germanic	**0.62**	2.56	**23.86**	**0.63**	2.50	**28.44**	–	–	–	–	–	–	–	–	–	68.5M
Romance	–	–	–	–	–	–	0.61	2.35	23.76	0.64	2.77	**25.24**	0.63	2.72	24.10	84.1M
Universal	0.62	2.55	23.04	0.63	2.44	27.43	**0.60**	**2.21**	**24.40**	**0.63**	**2.58**	25.09	**0.63**	**2.54**	**24.63**	115.4M
COUPLED POLICY																
Bilingual	0.56	1.55	21.33	0.56	1.35	26.63	0.55	1.48	26.90	0.53	1.00	22.62	0.54	1.02	22.82	278.2M
Germanic	**0.56**	**1.47**	**22.33**	**0.55**	1.26	**27.25**	–	–	–	–	–	–	–	–	–	73.3M
Romance	–	–	–	–	–	–	**0.55**	**1.37**	**27.45**	**0.53**	**0.92**	**23.40**	**0.54**	**1.01**	**23.35**	90.9M
Universal	0.56	1.57	21.33	0.55	**1.24**	26.03	0.55	1.37	27.27	0.54	0.97	23.17	0.54	1.04	22.44	118.2M

Table 2: Multilingual results for WAIT-2 and COUPLED POLICY under Bilingual, Universal, and the same language family (Germanic and Romance). Fully shared architecture is selected for Universal model while the same language group model has language-specific parameters for encoder. The last column indicates model parameter size, where bilingual row adds up the model size of all the language pairs, i.e. $55.646\text{M} \times 5 \approx 278.2\text{M}$.

Under the same SIMT model, COUPLED POLICY has better performance when trained in the same language group. Also, the model size decreases 40% compared to the bilingual baseline, where the same language family has a total of 164.2M parameters and the bilingual has a total of 278.2M parameters. WAIT-2 seems to have slightly arguable results, where DE and NL have the highest BLEU when trained language-family-wise, but the Romance language family benefits the most from universally trained in all languages. Also, one should note that a lower delay in WAIT-K under the same k value does not mean outputting the target sentence faster: (1) Because of the nature of WAIT-K, the model follows the fixed READ and WRITE actions, and (2) the formulation of AL accounts for not only the lagging of translation but also the number of tokens produced as output and taken as input. Therefore, a lower AL indicates the changes in the probability of producing the end of the sentence. This will generate a shorter target sentence and/or stop the translation without fully observing the input, which impacts the delay. Nevertheless, under WAIT-2, the translation quality improves, and the model size decreases with multilingualism.

3.2 Discussion

The setting for parameter sharing in this experiment is inspired from the observation that the multilingual NMT can benefit from separating encoder and decoder parameters (Dong et al., 2015; Sachan and Neubig, 2018; Ahmadnia and Dorr, 2020). The motivation from Dong et al. (2015); Sachan and Neubig (2018) is that separating decoder parameters in one-to-many setting is beneficial because of the difficulty of one-to-many translation task. Our problem is SIMT where not only mapping from the source language to the target language is important, but also learning when to map is equally important. Hence, our assumption was that due to the difficulty of many-to-one SIMT task, assigning the encoder and the agent to language-specific would help the performance. Under WAIT-K, encoding the representation of the source language separately does not seem to benefit. However, Table 2 shows multilingual setting surpasses bilingual. This would be similar to the traditional NMT, that the model generalize the translation tasks across different languages and leverages the correlation across the source languages.

COUPLED POLICY is more complex architecture than WAIT-K as it also needs to learn the optimal policy from an oracle trajectory. However, this takes more advantages when trained on the same language family. Since its oracle is generated from offline word alignments between the source language and the target language, its mechanism of read/write is dependent on the word order and language properties. Our results in Table 2 also supports this as the model trained on the same language family surpasses both bilingual baselines and universal model. The interesting observation here is that, unlike WAIT-K, a separated encoder takes advantages more than fully shared architecture while separating both encoder and agent significantly degrades the performance in BLEU. This suggests that the language-specific encoder can form the representation of the source languages better than the shared one, but if the agent is separated together, the model struggles mapping from the source language to the target language. This reflects why separated encoder and agent in COUPLED POLICY has a BLEU decay while AL is not affected as significant. Therefore, because COUPLED POLICY takes advantages of the same word order from its oracle trajectory, its shared agent can capture the general representation of the same word order better while the language-specific encoder can help the agent by only focusing on encoding the representation of each source language.

4 Related Work

Simultaneous Machine Translation SIMT has been explored as a sequential decision-making translation problem. NPI architecture is employed to 1) choose whether to take more input token or produce output token using agent programmer and 2) translate partially observed input tokens to output using neural machine translation (NMT) interpreter (Satija and Pineau, 2016; Gu et al., 2016). The initial approaches were mainly training the agent using reinforcement learning with assigned rewards to balance the trade-off between translation quality and delay (Gu et al., 2016; Satija and Pineau, 2016; Alinejad et al., 2018). However, it has stability and robustness issues due to the sparse reward signals, so imitation learning using oracle actions has been independently attempted (Zheng et al., 2019; Arthur et al., 2020; Dalvi et al., 2018).

Multilingual Machine Translation In NMT, multilingual training is a popular MTL approach as it is very simple, but effective (Johnson et al., 2017; Sachan and Neubig, 2018; Dong et al., 2015; Dabre et al., 2020). Instead of choosing entirely different NLP tasks and increase complexity of implementation (Niehues and Cho, 2017; Zaremoodi and Haffari, 2018), multilingual setting only involves concatenating multiple bilingual language pairs for training (Johnson et al., 2017). The language pairs are the task space in MTL, which determines the performance of the model, and so, the selection of language pairs influences the overall performance of translation (Tan et al., 2019).

Parameter sharing in multilingual setting has also been extensively studied. Dong et al. (2015) initially had language-specific decoder under one-to-many translation. This was further extended to sharing decoder parameters partially (Sachan and Neubig, 2018). Ahmadnia and Dorr (2020) investigated hierarchically sharing parameters under the similarity between languages. This simple parameter sharing has shown to restrict sharing dissimilarity, improving translation quality of all the languages (Johnson et al., 2017; Sachan and Neubig, 2018; Cettolo et al., 2017).

5 Conclusions

In this paper, we have investigated multilingual SIMT using IWSLT 2017 datasets. We have explored simple and effective multilingual architectures based on two strong recently proposed SIMT models, namely WAIT-K and COUPLED POLICY. Experiments show that the best parameter sharing strategy for the WAIT-K model, when dealing with DE/NL (as Germanic languages) and RO/IT/FR (as Romance languages), is to share all SIMT components across the languages regardless of the language set. However, the best sharing strategy seems to depend on the language family when it comes to COUPLED POLICY. Under the best parameter sharing strategy, our results have shown that (i) the single multilingual model is on-par or better than individual models, and (ii) multilingual SIMT models trained based on language families are on-par or better than the universal model trained for all languages. Furthermore, (iii) COUPLED POLICY takes the advantages of the same word order, so it achieves the best performance with the language-specific encoder and training under the same language family

For the future work, we plan to extend this to a larger dataset. Aharoni et al. (2019) demonstrated the scales of parallel corpora draw different conclusions in multilingual NMT. To maintain the characteristics of spoken languages, translation datasets must be selected carefully. Secondly, we will investigate different language families, including Slavic languages and Austronesian languages. The consistent results in different families would make the claim in this paper more valid.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Benyamin Ahmadnia and Bonnie Dorr. 2020. A new approach to parameter-sharing in multilingual neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (AMTA 2020)*, pages 1–6, Virtual. Association for Machine Translation in the Americas.

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027, Brussels, Belgium. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Te I, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2020. Re-translation strategies for long form, simultaneous, spoken language translation.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Philip Arthur, Trevor Cohn, and Gholamreza Haffari. 2020. Learning coupled policies for simultaneous machine translation.

M. Cettolo, M. Federico, L. Bentivogli, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. 2017. Overview of the iwslt 2017 evaluation campaign.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5).

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2016. Learning to translate in real-time with neural machine translation. *CoRR*, abs/1610.00388.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Qingsong Ma, Ondřej Bojar, and Yvette Graham. 2018. Results of the WMT18 metrics shared task: Both characters and embeddings achieve good performance. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 671–688, Belgium, Brussels. Association for Computational Linguistics.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. *CoRR*, abs/1708.00993.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Brussels, Belgium. Association for Computational Linguistics.

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning.

Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with language clustering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 963–973, Hong Kong, China. Association for Computational Linguistics.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2019. Recent advances in imitation learning from observation.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Neural machine translation for bilingually scarce scenarios: a deep multi-task learning approach. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1356–1365, New Orleans, Louisiana. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019. Simultaneous translation with flexible policy via restricted imitation learning. *CoRR*, abs/1906.01135.

A Appendices

SIMT Architecture We used a single layer long short term memory (LSTM) SEQ2SEQ as the interpreter for both SIMT models. The programmer is an LSTM transducer with a binary softmax to generate a read or write action for COUPLED POLICY. WAIT-K followed the fixed oracle actions under $k = 2$ without the programmer. We used scheduled sampling of 5%, 15%, and 15% for training the NPI-SIMT framework as described in Arthur et al. (2020).

Training is done using Adam (Kingma and Ba, 2015) with initial learning rate 0.001, and halved it each time when perplexity increased on the development set. Early stopping is reached at the fourth learning rate. During testing we use a standard beam search algorithm similar to Gu et al. (2016) with $b = 5$. Training is done using single V-100 GPU for 6 hours for one source language. Multilingual experiments time scale linearly to the numbers of parallel corpora being used for the experiment.

			DE→EN			NL→EN		
$\theta_{\mathcal{E}}$	$\theta_{\mathcal{D}}$	$\theta_{\mathcal{A}}$	AP	AL	BLEU	AP	AL	BLEU
WAIT-2								
×	×	–	0.62	**2.54**	22.99	0.63	**2.39**	28.33
×	✓	–	0.63	2.67	23.04	0.63	2.51	28.16
✓	✓	–	**0.62**	2.56	**23.86**	**0.63**	2.50	**28.44**
COUPLED POLICY								
×	×	×	0.56	1.55	21.33	0.56	1.35	26.63
×	✓	×	0.56	1.66	21.61	0.55	1.32	25.65
×	✓	✓	**0.56**	**1.47**	**22.33**	**0.55**	**1.26**	**27.25**
✓	✓	×	0.56	1.68	21.21	0.55	1.28	25.63
✓	✓	✓	0.56	1.49	21.69	0.56	1.31	26.22

Table 3: Parameter sharing under Germanic language family: German, Dutch. ✓ and × indicate *shared* and *not sharing* components in the MTL architecture.

			FR→EN			IT→EN			RO→EN		
$\theta_{\mathcal{E}}$	$\theta_{\mathcal{D}}$	$\theta_{\mathcal{A}}$	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU
WAIT-2											
×	×	–	**0.60**	**2.22**	23.40	**0.63**	**2.73**	24.77	0.63	**2.55**	24.34
×	✓	–	0.61	2.32	23.54	0.64	2.77	24.75	0.63	2.68	**24.43**
✓	✓	–	0.61	2.35	**23.76**	0.64	2.77	**25.24**	**0.63**	2.72	24.10
COUPLED POLICY											
×	×	×	0.55	1.48	26.90	0.53	1.00	22.62	0.54	1.02	22.82
×	✓	×	0.55	1.53	24.82	0.53	0.90	21.13	0.54	1.09	20.60
×	✓	✓	**0.55**	**1.37**	**27.45**	**0.53**	0.92	**23.40**	**0.54**	**1.01**	**23.35**
✓	✓	×	0.55	1.44	27.36	0.53	0.99	23.13	0.54	1.16	23.00
✓	✓	✓	0.55	1.38	27.34	0.53	**0.89**	23.32	0.54	1.02	22.63

Table 4: Parameter sharing under Romance language family: French, Italian, Romanian. ✓ and × indicate *shared* and *not sharing* components in the MTL architecture.

			DE→EN			FR→EN			IT→EN			NL→EN			RO→EN		
$\theta_{\mathcal{E}}$	$\theta_{\mathcal{D}}$	$\theta_{\mathcal{A}}$	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU	AP	AL	BLEU
WAIT-2																	
×	×	–	0.62	**2.54**	22.99	0.60	2.22	23.40	0.63	2.73	24.77	0.63	**2.39**	**28.33**	0.63	2.55	24.34
×	✓	–	0.62	2.59	22.85	0.60	2.22	24.04	0.63	2.60	**25.91**	0.63	2.48	27.96	0.63	2.54	24.47
✓	✓	–	**0.62**	2.55	**23.04**	**0.60**	**2.21**	**24.40**	**0.63**	**2.58**	25.09	**0.63**	2.44	27.43	**0.63**	**2.54**	**24.63**
COUPLED POLICY																	
×	×	×	0.56	1.55	21.33	0.55	1.48	26.90	0.53	1.00	22.62	0.56	1.35	**26.63**	0.54	**1.02**	**22.82**
×	✓	×	0.57	1.76	19.54	0.55	1.45	27.04	0.54	1.04	22.29	0.56	1.36	24.50	0.54	1.09	22.34
×	✓	✓	**0.55**	**1.29**	19.95	0.55	**1.33**	25.50	**0.53**	**0.92**	21.73	0.56	1.29	24.89	0.54	1.05	21.72
✓	✓	×	0.56	1.67	21.11	0.55	1.42	26.92	0.53	0.99	22.36	0.56	1.37	25.31	0.54	1.09	22.68
✓	✓	✓	0.56	1.57	**21.33**	**0.55**	1.37	**27.27**	0.54	0.97	**23.17**	**0.55**	**1.24**	26.03	**0.54**	1.04	22.44

Table 5: Parameter sharing under all language families: German, French, Italian, Dutch, Romanian. ✓ and × indicate *shared* and *not sharing* components in the MTL architecture.

Model Name	Model Size
`model.src_embedder.0.0.weight`	`torch.Size([32004, 512])`
`model.trg_embedder.1.0.weight`	`torch.Size([32004, 512])`
`model.encoder.0.weight_ih_l0`	`torch.Size([2048, 512])`
`model.encoder.0.weight_hh_l0`	`torch.Size([2048, 512])`
`model.encoder.0.bias_ih_l0`	`torch.Size([2048])`
`model.encoder.0.bias_hh_l0`	`torch.Size([2048])`
`model.decoder.1.weight_ih_l0`	`torch.Size([2048, 1024])`
`model.decoder.1.weight_hh_l0`	`torch.Size([2048, 512])`
`model.decoder.1.bias_ih_l0`	`torch.Size([2048])`
`model.decoder.1.bias_hh_l0`	`torch.Size([2048])`
`model.attention.1.src_projector.weight`	`torch.Size([512, 512])`
`model.attention.1.trg_projector.weight`	`torch.Size([512, 512])`
`model.attention.1.inner_projector.1.weight`	`torch.Size([1, 512])`
`model.context_projector.1.weight`	`torch.Size([512, 1024])`
`model.context_projector.1.bias`	`torch.Size([512])`
`model.output_projector.1.weight`	`torch.Size([32004, 512])`
`model.output_projector.1.bias`	`torch.Size([32004])`
`agent.action_embedder.0.weight`	`torch.Size([6, 512])`
`agent.input_projector.0.weight`	`torch.Size([512, 1536])`
`agent.input_projector.0.bias`	`torch.Size([512])`
`agent.rnn.0.weight_ih_l0`	`torch.Size([2048, 512])`
`agent.rnn.0.weight_hh_l0`	`torch.Size([2048, 512])`
`agent.rnn.0.bias_ih_l0`	`torch.Size([2048])`
`agent.rnn.0.bias_hh_l0`	`torch.Size([2048])`
`agent.output_projector.0.weight`	`torch.Size([6, 512])`
`agent.output_projector.0.bias`	`torch.Size([6])`

Table 6: The details of parameters in COUPLED POLICY baseline. WAIT-K has no `agent` parameters and multilingual models has additional components in `model.encoder` or `agent` and an increase in the first dimension of `model.src_embedder` by the number of languages, e.g., `torch.Size([32004, 512])` for bilingual baseline to `torch.Size([64004, 512])` for Germanic family. Total number of parameters for encoder, agent, attention, decoder, input embedding, output embedding and action embedding are 2.0M, 2.8M, 512.5K, 19.1M, 15.6M, 15.6M and 2.0K, respectively.

Cross-Domain Review Generation for Aspect-Based Sentiment Analysis

Jianfei Yu, Chenggong Gong, and Rui Xia[*]
School of Computer Science and Engineering,
Nanjing University of Science and Technology, China
`{jfyu, cggong, rxia}@njust.edu.cn`

Abstract

Supervised learning methods have proven to be effective for Aspect-Based Sentiment Analysis (ABSA). However, the lack of fine-grained labeled data hinders their effectiveness in many domains. To address this issue, unsupervised domain adaptation methods are desired to transfer knowledge from a labeled source domain to any unlabeled target domain. In this paper, we propose a new domain adaptation paradigm called cross-domain review generation (CDRG), which aims to generate target-domain reviews with fine-grained annotation based on the source-domain labeled reviews. To achieve this goal, we propose a two-step approach as a concrete realization of CDRG. It first converts a source-domain review to a domain-independent review by masking its source-specific attributes, and then converts the domain-independent review to a target-domain review with a masked language model pre-trained in the target domain. We further propose two ways to leverage the generated target-domain reviews for two cross-domain ABSA tasks. Extensive experiments demonstrate the superiority of our CDRG-based approaches over the state-of-the-art domain adaptation methods.

1 Introduction

Aspect-Based Sentiment Analysis (ABSA) has received considerable attention in recent years (Liu, 2020). The goal of ABSA is to extract the aspect terms mentioned in review sentences and predict the sentiments over them (Pontiki et al., 2016). For example, given the review "*The fish soup is delicious*", the aspect term and its sentiment are *fish soup* and *Positive*, respectively. With the recent advances in deep learning, many supervised neural models have been proposed for several ABSA tasks, e.g., aspect extraction (Liu et al., 2015; Xu et al., 2018), aspect-level sentiment classification (Wang et al., 2016b; Tang et al., 2016), and End-to-End ABSA (Zhang et al., 2015; Li et al., 2019a).

Although these neural models have obtained promising results in several product domains such as *Laptop* and *Restaurant* (Pontiki et al., 2014), the major obstacle to them is the lack of rich annotated resources in many new domains. Since ABSA requires fine-grained annotation of aspect terms and their sentiments in each review, it is extremely time-consuming to develop such resources for each domain, and the annotation process can be prohibitively expensive. To alleviate the annotation efforts, unsupervised domain adaptation methods are desired to transfer knowledge from a source domain with rich labeled data to a target domain with only unlabeled data.

The key challenge of domain adaptation is that the data distribution of the source domain usually differs from that of the target domain. To alleviate the domain discrepancy, many approaches have been proposed in coarse-grained sentiment classification based on the following two paradigms:

- **feature-based adaptation**, which aims to aims to learn a domain-invariant feature representation across domains (Blitzer et al., 2007; Glorot et al., 2011; Yu and Jiang, 2017; Ziser and Reichart, 2018; Ghosal et al., 2020)
- **instance-based adaptation**, which focuses on re-weighting labeled instances in the source domain for use in the target domain (Mansour et al., 2008; Dredze et al., 2010).

However, due to the challenges in fine-grained adaptation, only a few studies explore the domain adaptation problem for ABSA (Ding et al., 2017; Wang and Pan, 2018; Li et al., 2019b; Gong et al., 2020). Moreover, these studies still follow the above two domain adaptation paradigms, which suffer from two common limitations: (1) although

---

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4767–4777
August 1–6, 2021.

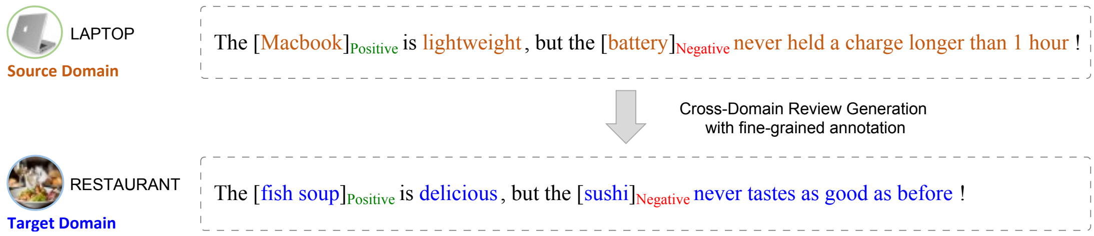

The [Macbook]$_{\text{Positive}}$ is lightweight, but the [battery]$_{\text{Negative}}$ never held a charge longer than 1 hour !

The [fish soup]$_{\text{Positive}}$ is delicious, but the [sushi]$_{\text{Negative}}$ never tastes as good as before !

Figure 1: Cross-Domain Review Generation with applications to the End-to-End ABSA task, in which the goal is to generate a target-domain review with fine-grained annotation given a labeled review in the source domain.

they can reduce the domain discrepancy by learning shared representations or re-weighting source instances, the supervision signals for their main task solely come from the labeled source domain; (2) both of them are lack of interpretability, as the shared representations or re-weighted instances offer little transparency regarding the knowledge transferred to the target domain.

To address the two limitations, we propose a new domain adaptation paradigm named **Cross-Domain Review Generation** (CDRG) with applications to the ABSA task. Given a labeled review in the source domain, the goal is to generate a target-domain review with fine-grained annotation, which converts the domain-specific attributes (e.g., aspects, opinions, and collocations) to the target domain while preserving its annotation and remaining contents. For example, in Fig. 1, a labeled review from the *Laptop* domain is transferred to the *Restaurant* domain by converting its source-specific attributes to target-specific attributes (e.g., *Macbook* to *fish soup*, *lightweight* to *delicious*, etc) but keeping other words and the labels unchanged. Different from existing text generation tasks, CDRG is challenging due to a couple of reasons. First, there is no paralleled corpus which aligns labeled source-domain reviews and unlabeled target-domain reviews. Second, given unparalleled corpus, it is non-trivial to achieve alignments between the domain-specific attributes. More importantly, the generated target review is required to have the fine-grained annotation.

To achieve this goal, we propose a simple yet effective two-step approach, containing a domain generalization step and a domain specification step. Specifically, the domain generalization step first identifies important domain-specific attributes[1], and then mask source-specific attributes in each source review to obtain a domain-independent review. Next, given the domain-independent review as input, the domain specification step employs a pre-trained masked language model from the target domain to generate a target review. In our two-step approach, the domain-independent review serves as a bridge to achieve word-to-word alignments in source and target reviews, and thus the fine-grained annotation from the source review can be directly transferred to the target review.

We further propose two training strategies to leverage the generated target-domain reviews for two cross-domain ABSA tasks, including cross-domain End-to-End ABSA (E2E-ABSA) and aspect extraction (AE). Experiment results on four benchmark datasets show that only using our generated target-domain reviews, the baseline BERT model already outperforms the state-of-the-art domain adaptation methods, and a joint usage of generated target reviews and labeled source reviews can further boost the performance significantly.

Our main contributions can be summarized as follows:

- We propose a new domain adaptation paradigm named **Cross-Domain Review Generation** (CDRG) with applications to the ABSA task, and then devise a simple yet effective two-step approach as a concrete realization of CDRG.
- With the help of generated target-domain reviews, our best training strategy outperforms the state-of-the-art method by an absolute improvement of 2.83% and 4.47% on Micro-F1 for cross-domain E2E-ABSA and cross-domain AE, respectively.
- As long as a source domain has sufficient annotated reviews, our two-step approach for CDRG can generate many annotated target reviews, which offer interpretable justification for domain adaptation.

---

[1] Domain-specific attributes can be words, phrases, syntactic structures, and expression styles only occurring in the source domain or the target domain. However, as aspect and opinion terms are the core of ABSA, we only consider domain-specific aspect and opinion terms in our approach.

2 Related Work

As two important tasks in ABSA, aspect extraction (Liu et al., 2015; Poria et al., 2016; Wang et al., 2016a, 2017; Li et al., 2018a; Xu et al., 2018) and aspect-level sentiment classification (Dong et al., 2014; Tang et al., 2016; Wang et al., 2016b; Yang et al., 2017; Ma et al., 2017; Wang et al., 2018) have been extensively studied in the literature. For practical applications, a number of recent studies handle them together in an end-to-end manner, in which many supervised learning methods with discrete linear features (Mitchell et al., 2013) and continuous neural features (Zhang et al., 2015; Li et al., 2019a) have been proposed. Despite obtaining promising results, their main limitation lies in the lack of annotated data in many new domains. To address this data sparsity problem, unsupervised domain adaptation methods are desired.

Most existing domain adaptation studies focus on coarse-grained sentiment classification to learn domain-invariant representations, including pivot-based methods (Blitzer et al., 2007; Pan et al., 2010; Yu and Jiang, 2016), auto-encoders (Chen et al., 2012; Zhuang et al., 2015), semi-supervised methods (He et al., 2018; Ye et al., 2020), and domain adversarial networks (Ganin et al., 2016; Li et al., 2018b). Besides, another line of work focuses on re-weighting source instances to automatically find useful source samples for the target domain (Jiang and Zhai, 2007; Mansour et al., 2008; Dredze et al., 2010; Xia et al., 2014). Due to the challenges in fine-grained adaptation, there exist only a few studies for cross-domain aspect and opinion extraction (Li et al., 2012; Ding et al., 2017; Wang and Pan, 2018), or End-to-End ABSA (Li et al., 2019b; Gong et al., 2020). However, these methods still follow the traditional domain adaptation paradigms to either learn shared representations or perform instance weighting. Different from these methods, we propose to accomplish domain adaptation for ABSA based on Cross-Domain Review Generation.

3 Problem Formulation

In this paper, we consider two ABSA tasks, i.e., End-to-End ABSA (E2E-ABSA) and Aspect Extraction (AE). Following Li et al. (2019b), we formulate both tasks as sequence labeling problems. Formally, given a sequence of input tokens $\mathbf{x} = \{w_1, w_2, ..., w_n\}$, its label sequence is denoted by $\mathbf{y} = \{y_1, y_2, ..., y_n\}$. Let $y_j \in \{$`B-POS`, `I-POS`, `B-NEG`, `I-NEG`, `B-NEU`, `I-NEU`, `O`$\}$ be the label space for the E2E-ABSA task, and $y_j \in \{$`B`, `I`, `O`$\}$ be the label space for the AE task.

Cross-Domain ABSA. We focus on the unsupervised domain adaptation setting, in which labeled data are only available from the source domain. Specifically, we assume access to a set of labeled reviews from the source domain $D_S = \{(\mathbf{x}_i^s, \mathbf{y}_i^s)\}_{i=1}^{N^s}$, and another set of unlabeled reviews from the target domain $D_U = \{\mathbf{x}_i^u\}_{i=1}^{N^u}$. The goal is to predict the label sequence for test data in the target domain: $\mathbf{y}_i^t = f_t(\mathbf{x}_i^t)$, $D_T = \{\mathbf{x}_i^t\}_{i=1}^{N^t}$.

4 Methodology

Overview. In this work, we propose a new domain adaptation paradigm named Cross-Domain Review Generation (CDRG). The goal is to transform a labeled source review to a labeled target review by converting its source-specific attributes to target-specific attributes while retaining the remaining contents and its labels. To achieve this goal, we propose a simple yet effective two-step approach, as shown in Fig. 2. In the domain generalization step, we first extract the domain-specific attributes from the labeled source data and the unlabeled target data. For each labeled source review, we mask its source-specific attributes to obtain a domain-independent review. In the domain specification step, a pre-trained BERT model is first re-trained with masked language modeling (MLM) on the unlabeled data from the target domain. Next, given the domain-independent review, we employ the target-domain MLM to generate a target review. As the domain-independent review serves as a bridge to achieve word-to-word alignments between the source and generated target reviews, the source labels can be directly transferred to the target review. Finally, we propose two strategies to leverage labeled target reviews for E2E-ABSA and AE tasks, including independent training and merge training.

4.1 Proposed Two-Step Approach for CDRG

Our proposed approach for CDRG consists of two steps: (1) a domain generalization step to convert each source-domain review to a domain-independent review; (2) a domain specification step to convert a domain-independent review to a target-domain review.

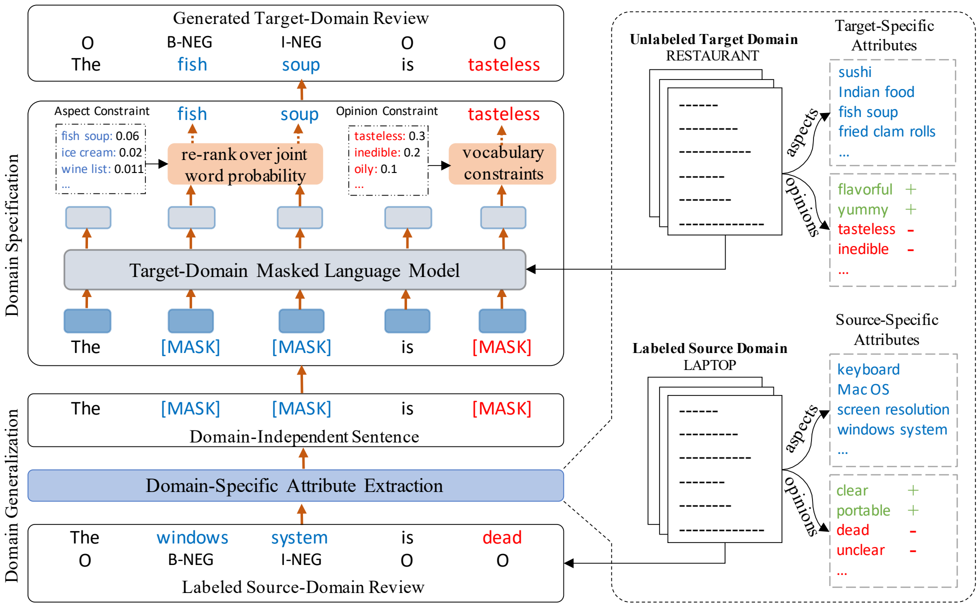

Figure 2: Overview of our two-step approach for Cross-Domain Review Generation, which contains a domain generalization step and a domain specification step. Note that we use Double Propagation to extract all the opinion terms together with their sentiments, and +, -, and 0 denote Positive, Negative, and Neutral sentiments, respectively.

4.1.1 Step 1: Domain Generalization

To convert each source-domain review to a domain-independent review, we first extract the domain-specific attributes from each domain, and then mask the source-specific attributes in source reviews.

Domain-Specific Attribute Extraction. We define domain-specific attributes as words, phrases, syntactic structures, and expression styles that only occur in the source domain or the target domain. However, it is often challenging to identify some attributes such as syntactic structures and expression styles, as most of them are implicitly expressed in the review. More importantly, since ABSA aims to jointly extract the aspects and sentiments, aspect and opinion terms tend to play more crucial roles than the other attributes. Therefore, we only consider domain-specific aspect terms and opinion terms as domain-specific attributes in this paper.

To extract aspect and opinion terms from the unlabeled target domain, we use a dependency relation-based unsupervised method named Double Propagation (Qiu et al., 2011). Specifically, given the unlabeled target reviews D_U, we first resort to a sentiment lexicon[2] to extract the opinion terms in D_U, and use the *conj* relation to expand the opinion term list. Next, we employ these opinion terms as seed words, and extract all the words holding the *amod* and *nsubj* relations towards any seed word. We then treat the extracted words as aspect terms[3], and use the *nn* relation to expand the aspect term list. The above three steps can be iterated until the aspect and opinion term lists are no longer updated.

Given the labeled source reviews D_S, we can easily obtain the aspect term list, as the aspect terms have been annotated in each review for both E2E-ABSA and AE tasks. For opinion terms, we also utilize Double Propagation to expand its list.

After obtaining the aspect and opinion term lists for each domain, we remove all the domain-independent terms that occur in both source and target domains, and obtain the domain-specific term lists. Let us use A_s and O_s to denote the source-specific aspect and opinion term lists, and A_t and O_t the target-specific aspect and opinion term lists.

Domain-Independent Reviews. Given each review in the source domain $\mathbf{x}^s = \{w_1, w_2, ..., w_n\}$, if its sub-sequence is a source-specific attribute in either A_s or O_s, we substitute each word in the sub-sequence with a special token `[MASK]` to obtain a domain-independent review, denoted by $\mathbf{x}^m$. For example, in Fig 2, the source-specific aspect term *windows system* and opinion term *dead* are replaced by two `[MASK]` tokens and one `[MASK]` token, respectively.

---

[2] https://www.cs.uic.edu/∼liub/FBS/sentiment-analysis.html#lexicon

[3] Since Double Propagation can only extract single-word aspect terms, we further expand them to multi-word aspect terms if their previous words have an *amod* or *nn* relation towards them (e.g., *Italian*$\xrightarrow{amod}$*food*, *fish*$\xrightarrow{nn}$*soup*, etc).

4.1.2 Step 2: Domain Specification

The domain specification step is responsible for incorporating target-specific attributes into each domain-independent review $\mathbf{x}^m$. In our work, we propose to transform it as a text infilling problem. To address this, we pre-train a Transformer network with the masked language modeling (MLM; Devlin et al. (2019)) task on unlabeled data in the target domain, followed by predicting a target-specific word for each masked token in $\mathbf{x}^m$.

Target-Domain MLM. Since the size of the unlabeled target-domain data D_U is typically small, we adopt a pre-trained BERT model (Devlin et al., 2019) and re-train the BERT model with MLM on D_U. Specifically, we create training instances by randomly replacing a subset of tokens with `[MASK]` in each unlabeled review $\mathbf{x}^u$, and the objective is to recover the masked tokens based on the hidden states from BERT.

With the target-domain MLM (TD-MLM), we can infill each masked position in $\mathbf{x}^m$ based on their context. Let us use $M = \{m_1, m_2, ..., m_K\}$ to denote the indexes of the masked tokens in $\mathbf{x}^m$, where K refers to the number of masked tokens. The predicted word for the j-th masked token can be computed as follows:

$$\begin{aligned} &p(w_{m_j}|\mathbf{x}^m) = \text{TD-MLM}(w_{m_j}|\mathbf{x}^m); \\ &o_{m_j} = \arg\max_{w_{m_j}} p(w_{m_j}|\mathbf{x}^m), \quad w_{m_j} \in V, \end{aligned} \tag{1}$$

where V denotes the whole vocabulary, and o_{m_j} is the output word with the highest probability.

However, the TD-MLM suffers from two major limitations: (1) the predicted token with the highest probability in Eqn. (1) may not be an aspect term or an opinion term; (2) each masked token is predicted independently, and thus it is possible that the predicted tokens for two consecutive masked positions are not coherent.

Target-Specific Aspect Constraint. To tackle these two limitations, we first propose to utilize the target-specific aspect terms (i.e., A_t) extracted in Section 4.1.1 as vocabulary constraints to limit the prediction space of each masked aspect term.

Specifically, if the masked aspect term corresponds to a single-word term in the source review (e.g., *keyboard*), the word selection in Eqn. (1) can be modified as follows:

$$o_{m_j} = \arg\max_{w_{m_j}} p(w_{m_j}|\mathbf{x}^m), \quad w_{m_j} \in A_t^1, \tag{2}$$

where A_t^1 refers to the sets of single-word aspect terms in A_t. Otherwise, if the masked aspect term corresponds to a multi-word term, we compute the joint word probabilities of each multi-word term in A_t followed by re-ranking them. Let us use k to denote the number of consecutive masked tokens, and $m_{j:j+k}$ to denote the span of the masked aspect term. The word selection for k consecutive masked tokens can thus be computed as follows:

$$\begin{aligned} &p(w_{m_{j:j+k}}|\mathbf{x}^m) = \prod_{i=j}^{j+k-1} p(w_{m_i}|\mathbf{x}^m); \\ &o_{m_{j:j+k}} = \arg\max_{w_{m_{j:j+k}}} p(w_{m_{j:j+k}}|\mathbf{x}^m), \end{aligned} \tag{3}$$

where $w_{m_{j:j+k}} \in A_t^k$, and A_t^k refers to the sets of k-word aspect terms in A_t.

Target-Specific Opinion Constraint. For each masked opinion term, it is important to keep the sentiment consistency when predicting its corresponding target-specific opinion term.

To achieve this, we resort to the Double Propagation algorithm, which relies on the sentiment lexicon to assign sentiment (i.e., Positive, Negative, and Neutral) to each aspect term and opinion term in a bootstrapping manner. Based on the output, we obtain the sentiment of all the source-specific and target-specific opinion terms (i.e., O_s and O_t) extracted in Section 4.1.1.

Next, we look up the sentiment of the masked source-specific opinion term, and then utilize all the target-specific opinion terms with the same sentiment in O_t as vocabulary constraints in Eqn. (2) and Eqn. (3) to generate the target-specific opinion terms. For example, in Fig 2, since *dead* is a source-specific negative opinion term, we use all the single-word target-specific opinion terms with the negative sentiment as vocabulary constraints in Eqn. (2) to generate the opinion term *tasteless*.

Generated Target-Domain Reviews. Based on Eqn. (2) and Eqn. (3), we can infill the masked positions in each domain-independent review $\mathbf{x}^m$, and obtain the generated target-domain review, denoted by $\mathbf{x}^g$. It is worth noting that if a source-domain review $\mathbf{x}^s$ does not contain any source-specific attributes, its generated target-domain review $\mathbf{x}^g$ will be the same as $\mathbf{x}^s$.

With the domain-independent review $\mathbf{x}^m$, our two-step approach essentially achieves word-to-word alignments between $\mathbf{x}^s$ and $\mathbf{x}^g$. Therefore, we can directly employ the sequence label of $\mathbf{x}^s$ as the fine-grained annotation of $\mathbf{x}^g$. Fig. 2 shows a label transferring example for the E2E-ABSA task.

Formally, we use $D_G = \{(\mathbf{x}_i^g, \mathbf{y}_i^s)\}_{i=1}^{N^s}$ to denote the set of the generated target-domain reviews.

Dataset	Domain	Sentences	Training	Testing
L	Laptop	3845	3045	800
R	Restaurant	6035	3877	2158
D	Device	3836	2557	1279
S	Service	2239	1492	747

Table 1: Basic statistics of our datasets.

4.2 Post-Generation Training for Main Tasks

After obtaining the set of generated target-domain reviews D_G, we further propose two strategies to leverage them to train effective E2E-ABSA and AE models for the target domain as follows.

Independent Training. An intuitive strategy is to solely treat D_G as training instances, and directly train a sequence labeling model over them. Following Gong et al. (2020), we adopt the pre-trained BERT model as the text encoder, followed by fune-tuning it on D_G.

Merge Training. Since the qualities of the generated target-domain reviews rely on the target-domain MLM as well as the aspect and opinion terms extracted by Double Propagation, it is inevitable that D_G contains a number of aspect and opinion terms with incorrect annotations. Therefore, we propose to merge the labeled source reviews D_S with D_G as the training instances, which may alleviate the annotation noise in D_G. Similar to Independent Training, a BERT-based sequence labeling model is trained over the merged corpus $D_S \cup D_G$.

5 Experiment

5.1 Experiment Settings

Datasets. We use four benchmark datasets including Laptop (L), Restaurant (R), Device (D), and Service (S) for experiments. L is from SemEval-2014 ABSA challenge (Pontiki et al., 2014), containing user reviews from the laptop domain. R refers to a combination of the restaurant datasets from SemEval ABSA challenge 2014, 2015, and 2016 (Pontiki et al., 2014, 2015, 2016). D is the union set of all the digital device reviews collected by Toprak et al. (2010). S contains reviews from web services, which is introduced by Hu and Liu (2004). The basic statistics are shown in Table 1.

Settings. We carry out experiments on 10 transfer pairs with the four domains above. Following previous work (Li et al., 2019b; Gong et al., 2020), we remove D→L and L→D, as the two domains are very similar. For each transfer pair, the training data is a combination of the labeled training data in the source domain and the unlabeled training data in the target domain. We report the evaluation results on the test data from the target domain. For fair comparison with previous work, we use the Micro-F1 score with the exact match as the evaluation metric, which means that the predicted labels can be counted as correct only if they are exactly matched with the golden labels.

5.2 Compared Systems & Hyperparameters

To show the effectiveness of our Cross-Domain Review Generation (CDRG)-based methods, we consider the following compared systems:

- DP (Qiu et al., 2011): the unsupervised Double Propagation method detailed in Section 4.1.1.
- Hier-Joint (Ding et al., 2017): An LSTM-based domain adaptation method with syntactic rule-based auxiliary tasks for cross-domain AE.
- RNSCN (Wang and Pan, 2018): A recursive neural structural correspondence network based on syntactic structures and auto-encoders.
- AD-SAL (Li et al., 2019b): A Selective Adversarial Learning method to achieve local semantic alignments for fine-grained domain adaptation.
- BERT_B and BERT_E: directly fine-tuning two versions of pre-trained encoders on the labeled source domain. BERT_B is the uncased BERT_base model from Devlin et al. (2019), and BERT_E is another uncased BERT_base model from Xu et al. (2019), which is pre-trained on E-commerce reviews from the Amazon Electronics dataset (He and McAuley, 2016) and the Yelp Challenge.
- BERT_B-UDA and BERT_E-UDA (Gong et al., 2020): our recent unified feature and instance-based domain adaptation method based on BERT_B and BERT_E, respectively.

Besides, we consider the following variants of our CDRG-based domain adaptation methods:

- BERT_B-CDRG-X: re-training BERT_B with the MLM task to obtain the Target-Domain MLM in Section 4.1.2, and employing BERT_B as the base model for the two training strategies in Section 4.2. Here X can be independent training and merge training.
- BERT_E-CDRG-X: replacing the BERT_B model in BERT_B-CDRG-X with BERT_E.

Methods	S→R	L→R	D→R	R→S	L→S	D→S	R→L	S→L	R→D	S→D	Average
DP (Qiu et al., 2011)	34.47	34.47	34.47	18.31	18.31	18.31	16.63	16.63	19.03	19.03	22.97
Hier-Joint (Ding et al., 2017)	31.10	33.54	32.87	15.56	13.90	19.04	20.72	22.65	24.53	23.24	23.72
RNSCN (Wang and Pan, 2018)	33.21	35.65	34.60	20.04	16.59	20.03	26.63	18.87	33.26	22.00	26.09
AD-SAL (Li et al., 2019b)	41.03	43.04	41.01	28.01	27.20	26.62	34.13	27.04	35.44	33.56	33.71
BERT$_B$	44.66	40.38	40.32	19.48	25.78	30.31	31.44	30.47	27.55	33.96	32.44
BERT$_B$-UDA (Gong et al., 2020)	47.09	45.46	42.68	33.12	27.89	28.03	33.68	34.77	34.93	32.10	35.98
BERT$_B$-CDRG-Indep Training	44.46	44.96	39.42	34.10	33.97	31.08	33.59	26.81	25.25	29.06	34.27
BERT$_B$-CDRG-Merge Training	47.92	49.79	47.64	35.14	38.14	37.22	38.68	33.69	27.46	34.08	38.98
BERT$_E$	51.34	45.40	42.62	24.44	23.28	28.18	39.72	35.04	33.22	33.22	35.65
BERT$_E$-UDA (Gong et al., 2020)	**53.97**	49.52	51.84	30.67	27.78	34.41	43.95	**35.76**	**40.35**	**38.05**	40.63
BERT$_E$-CDRG-Indep Training	51.01	54.56	54.33	42.52	39.28	36.98	40.23	33.41	30.56	32.05	41.49
BERT$_E$-CDRG-Merge Training	53.09	**57.96**	**54.39**	**40.85**	**42.96**	**38.83**	**45.66**	35.06	31.62	34.22	**43.46**

Table 2: Comparison results of different methods for Cross-Domain End-to-End ABSA based on Micro-F1.

Methods	S→R	L→R	D→R	R→S	L→S	D→S	R→L	S→L	R→D	S→D	Average
DP (Qiu et al., 2011)	37.63	37.63	37.63	19.74	19.74	19.74	19.79	19.79	21.82	21.82	25.53
Hier-Joint (Ding et al., 2017)	46.39	48.61	42.96	27.18	25.22	29.28	34.11	33.02	34.81	35.00	35.66
RNSCN (Wang and Pan, 2018)	48.89	52.19	50.39	30.41	31.21	35.50	47.23	34.03	**46.16**	32.41	40.84
AD-SAL (Li et al., 2019b)	52.05	56.12	51.55	39.02	38.26	36.11	45.01	35.99	43.76	**41.21**	43.91
BERT$_B$	54.29	46.74	44.63	22.31	30.66	33.33	37.02	36.88	32.03	38.06	37.60
BERT$_B$-UDA (Gong et al., 2020)	56.08	51.91	50.54	34.62	32.49	34.52	46.87	43.98	40.34	38.36	42.97
BERT$_B$-CDRG-Indep Training	53.79	55.13	50.07	41.74	44.14	37.1	40.18	33.22	30.78	34.97	42.11
BERT$_B$-CDRG-Merge Training	56.26	60.03	52.71	42.36	47.08	41.85	46.65	39.51	32.60	36.97	45.60
BERT$_E$	57.56	50.42	45.71	26.50	25.96	30.40	44.18	41.78	35.98	35.13	39.36
BERT$_E$-UDA (Gong et al., 2020)	59.07	55.24	56.40	34.21	30.68	38.25	54.00	**44.25**	42.40	40.83	45.53
BERT$_E$-CDRG-Indep Training	58.75	65.81	**59.61**	**50.68**	51.25	40.17	49.17	41.61	33.34	36.97	48.74
BERT$_E$-CDRG-Merge Training	**59.17**	**68.62**	58.85	47.61	**54.29**	**42.20**	**55.56**	41.77	35.43	36.53	**50.00**

Table 3: Comparison results of different methods for Cross-Domain Aspect Extraction based on Micro-F1.

For re-training the MLM task in our two-step approach for CDRG, we employ the Adam optimizer (Kingma and Ba, 2014) with a batch size of 32 and a learning rate of 3e-5. For the two training strategies in Section 4.2, we also adopt the Adam optimizer, in which the learning rate, the dropout rate and the batch size are set to 5e-5, 0.1, 32 after a grid search over the combinations of [2e-5, 8e-5], [0.1, 0.3] and {16, 32, 64}. These hyper-parameters are tuned on 10% randomly held-out training data from the source domain. All the experiments are run on a Nvidia GTX 1080 Ti GPU.

5.3 Main Results for Cross-Domain ABSA

We report the comparison results on the cross-domain E2E-ABSA task in Table 2, and make the following observations: (1) Comparing *Indep Training* with all the baseline methods, we can see that solely fine-tuning BERT$_E$ on our generated target-domain reviews can already outperform all the existing domain adaptation approaches on average, including our recent unified feature and instance-based adaptation (UDA) method. This demonstrates the usefulness of our CDRG-based approach. (2) By merging the generated target-domain reviews with the labeled source reviews, our *Merge Training* strategy further boosts the average performance of *Indep Training*, which outperforms the state-of-the-art *UDA* approach by an absolute improvement of 3.00% and 2.83% based on BERT$_B$ and BERT$_E$ respectively. All these observations verify the superiority of our CDRG-based approach over the previous feature and instance-based adaptation methods.

Similar to the results on cross-domain E2E-ABSA, Table 3 shows that our *Indep Training* strategy obtains indistinguishable performance compared with the state-of-the-art method UDA based on BERT$_B$, and achieves significantly better performance than UDA based on BERT$_E$. Moreover, our *Merge Training* strategy can consistently achieve the best average performance on the cross-domain AE task, which outperforms UDA by an absolute improvement of 2.63% and 4.47% based on BERT$_B$ and BERT$_E$ respectively. This demonstrates the general effectiveness of our CDRG-based domain adaptation methods.

5.4 Ablation Study of Our Two-Step Approach for CDRG

To investigate the effectiveness of our two-step approach for CDRG, we conduct the ablation study of our target-domain masked language model with constraints (TD-MLM-C) approach, and consider the following variants: (1) *BERT$_E$*: The pre-trained BERT$_E$ model without re-training MLM on the un-

Source Domain: Laptop→Target Domain: Restaurant		
Source Review	1. The [screen graphics]$_{P}$ and [clarity]$_{P}$, and [sharp ##ness]$_{P}$ are great.	2. The [battery life]$_{P}$ is great.
Target Review from BERT$_{E}$	The [laptop,]$_{P}^{✗}$ and [sound]$_{P}^{✗}$, and [the ##s]$_{P}^{✗}$ are great.	The [touch screen]$_{P}^{✗}$ is great.
Target Review from TD-MLM	The [food,]$_{P}^{✗}$ and [service]$_{P}^{✗}$, and [the prices]$_{P}^{✗}$ are great.	The [su here]$_{P}^{✗}$ is great.
Target Review from TD-MLM-C	The [pizza ##s]$_{P}^{✓}$ and [atmosphere]$_{P}^{✓}$, and [service staff]$_{P}^{✓}$ are great.	The [su ##shi]$_{P}^{✓}$ is great.
Source Domain: Restaurant→ Target Domain: Laptop		
Source Review	3. The [food]$_{P}$ is [flavor ##ful], [pl ##ent ##iful] and reasonably priced.	4. [Pizza]$_{P}$ is terrific, as is [homemade pasta]$_{P}$.
Target Review from BERT$_{E}$	The [food]$_{P}^{✗}$ is [flavor ##ful]$^{✗}$, [st ##ten ,]$^{✗}$ and reasonably priced.	[It]$_{P}^{✗}$ is terrific, as is [the price]$_{P}^{✗}$.
Target Review from TD-MLM	The [keyboard]$_{P}^{✓}$ is [joy ##ful]$^{✓}$, [easy ##yl ,]$^{✗}$ and reasonably priced.	[It]$_{P}^{✗}$ is terrific, as is [the screen]$_{P}^{✗}$.
Target Review from TD-MLM-C	The [keyboard]$_{P}^{✓}$ is [joy ##ful]$^{✓}$, [st ##yl ##ish]$^{✓}$ and reasonably priced.	[Speed]$_{P}^{✓}$ is terrific, as is [windows os]$_{P}^{✓}$.

Table 4: Example target-domain reviews generated from BERT$_{E}$, TD-MLM, and TD-MLM-C. ✓ and ✗ indicate that the generated target-specific attributes are correct or incorrect. P and N denote the positive and negative sentiment. The blue and red colors refer to the domain-specific aspect terms and opinion terms, respectively. ## denotes that the original word is split into several sub-tokens by the tokenizer of BERT.

	Src&Tgt	BERT$_{E}$&Tgt	TD-MLM&Tgt	TD-MLM-C&Tgt
S→R	0.3496	0.2411	0.1621	**0.1477**
L→R	0.3155	0.2978	0.2456	**0.2072**
D→R	0.3504	0.3178	0.2109	**0.1650**
R→S	0.3330	0.2977	0.2516	**0.1654**
L→S	0.1966	0.1945	0.1742	**0.1518**
D→S	0.1804	0.1628	0.1167	**0.0941**
R→L	0.3135	0.2563	0.1589	**0.1080**
S→L	0.2767	0.1652	**0.1554**	0.1763
R→D	0.3476	0.2954	0.2026	**0.1433**
S→D	0.1765	0.0929	**0.0727**	0.0788
Average	0.2840	0.2322	0.1751	**0.1438**

Table 5: Maximum Mean Discrepancy (lower is better) between the target-domain test set and the review set generated from BERT$_{E}$, TD-MLM, and TD-MLM-C.

	Cross-Domain E2E-ABSA			Cross-Domain AE		
	BERT$_{E}$	TD-MLM	TD-MLM-C	BERT$_{E}$	TD-MLM	TD-MLM-C
S→R	46.42	47.32	**51.01**	53.04	52.87	**58.75**
L→R	52.40	**55.03**	54.56	59.22	62.57	**65.81**
D→R	44.90	50.43	**54.33**	48.88	54.21	**59.61**
R→S	22.24	34.67	**42.52**	24.58	39.41	**50.68**
L→S	23.98	35.84	**39.28**	26.31	41.87	**51.25**
D→S	22.79	31.44	**36.98**	25.09	34.00	**40.17**
R→L	33.89	39.21	**40.23**	39.73	48.49	**49.17**
S→L	29.88	30.78	**33.41**	34.61	35.64	**41.61**
R→D	30.32	29.36	**30.56**	32.91	32.80	**33.34**
S→D	25.67	26.69	**32.05**	28.07	28.68	**36.97**
Average	33.25	38.07	**41.49**	37.24	43.05	**48.74**

Table 6: Comparison results of using the labeled reviews generated from different methods as training instances for our Independent Training strategy.

labeled target data; (2) *TD-MLM*: re-training the BERT$_{E}$ model with MLM on the unlabeled target data; (3) *TD-MLM-C*: adding target-specific attribute constraints into TD-MLM, as introduced in Section 4.1.2.

First, we propose to verify the closeness between the generated target-domain review set and the real test set from the target domain. Specifically, we employ BERT$_{E}$ to obtain the sentence representation of each review in the two sets, and then compute the distance between the two sets with Maximum Mean Discrepancy (Gretton et al., 2012). Based on the results in Table 5, we can observe that all the three methods can consistently reduce the discrepancy between source and target domains. This shows our two-step approach is generally useful for domain adaptation. Moreover, it is clear that the distance between the review set from *TD-MLM-C* and the test set is the smallest, which implies the distribution of its generated reviews is closest to the distribution of the target domain.

Second, we further treat the generated reviews as training instances for our *Indep Training* strategy, and compare their results on the cross-domain E2E-ABSA and AE tasks. From the results in Table 6, we can see that using the reviews from *TD-MLM-C* consistently achieves the best results, and outperforms the other methods with a significant margin.

5.5 Manual Evaluation on Generated Target-Domain Reviews

Since there is no ground-truth target review for each source review, we randomly select 200 source reviews from L→R and R→L transfer pairs, and manually evaluate the generated reviews in terms of coherence, label correctness, and domain-specific criteria. Based on our manual evaluation, we observe that our *TD-MLM-C* approach generates better target reviews than its two ablation systems for 152 source reviews; for the remaining 48 source reviews, we cannot determine the winning method, as all the three methods generate meaningful or unmeaningful target-domain reviews.

Table 4 shows four representative examples. We can find that *BERT$_{E}$* is generally insensitive to the target domain, which may still generate source-specific terms (e.g., *touch screen* in S2 and *flavorful* in S3). *TD-MLM* can produce better reviews, as it tends to convert source-specific terms to the target domain. However, it still suffers from generating non-aspect or non-opinion terms (e.g., *the prices* in S1 and *It* in S4). In contrast, with the vocabulary constraints, *TD-MLM-C* can successfully convert all the source-specific attributes to target-specific attributes in the four cases.

All these observations verify the importance of TD-MLM and vocabulary constraints in our two-step approach for cross-domain review generation.

6 Conclusion

In this paper, we propose a new domain adaptation paradigm named Cross-Domain Review Generation (CDRG) with applications to the ABSA task. Specifically, we first propose a two-step approach to generate labeled target-domain reviews based on labeled source-domain reviews for CDRG, and then propose two training strategies to leverage the generated reviews for two cross-domain ABSA tasks. Experiments on four benchmark datasets demonstrate that our CDRG-based approaches significantly outperform existing methods for cross-domain E2E-ABSA and cross-domain AE tasks.

Acknowledgments

We would like to thank three anonymous reviewers for their insightful comments and helpful suggestions. Jianfei Yu and Chenggong Gong contributed equally to this paper. This work was supported by the Natural Science Foundation of China (No. 62076133 and 62006117), and the Natural Science Foundation of Jiangsu Province for Young Scholars (No. BK20200463) and Distinguished Young Scholars (No. BK20200018).

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*.

Minmin Chen, Zhixiang Xu, Kilian Q Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ying Ding, Jianfei Yu, and Jing Jiang. 2017. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of ACL*.

Mark Dredze, Alex Kulesza, and Koby Crammer. 2010. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(1-2):123–149.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Deepanway Ghosal, Devamanyu Hazarika, Abhinaba Roy, Navonil Majumder, Rada Mihalcea, and Soujanya Poria. 2020. KinGDOM: Knowledge-Guided DOMain Adaptation for Sentiment Analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: a deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*.

Chenggong Gong, Jianfei Yu, and Rui Xia. 2020. Unified feature and instance based domain adaptation for end-to-end aspect-based sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019a. A unified model for opinion target extraction and target sentiment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018a. Aspect term extraction with history attention and selective transformation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

Zheng Li, Xin Li, Ying Wei, Lidong Bing, Yu Zhang, and Qiang Yang. 2019b. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018b. Hierarchical attention transfer network for cross-domain sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bing Liu. 2020. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2008. Domain adaptation with multiple sources. *Advances in neural information processing systems*, 21:1041–1048.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *10th International Workshop on Semantic Evaluation (SemEval 2016)*.

Maria Pontiki, Dimitrios Galanis, Harris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Auresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational linguistics*, 37(1):9–27.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Wenya Wang and Sinno Jialin Pan. 2018. Recursive neural structural correspondence network for cross-domain aspect and opinion co-extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. Attention-based lstm for aspect-level sentiment classification. In *EMNLP 2016: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Rui Xia, Jianfei Yu, Feng Xu, and Shumei Wang. 2014. Instance-based domain adaptation in nlp via in-target-domain logistic approximation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2018. Double embeddings and cnn-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention based lstm for target dependent sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hai Ye, Qingyu Tan, Ruidan He, Juntao Li, Hwee Tou Ng, and Lidong Bing. 2020. Feature adaptation of pre-trained language models across languages and domains with robust self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP 2016: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jianfei Yu and Jing Jiang. 2017. Leveraging auxiliary tasks for document-level cross-domain sentiment classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

On the Language Coverage Bias for Neural Machine Translation

Shuo Wang* **Zhaopeng Tu**$^{\top}$ **Zhixing Tan*** **Shuming Shi**$^{\top}$ **Maosong Sun***† **Yang Liu***†‡
*Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, Tsinghua University
$^{\top}$Tencent AI Lab
†Beijing Academy of Artificial Intelligence
‡Institute for AIR, Tsinghua University
*\{wangshuo2018, zxtan, sms, liuyang2011\}@tsinghua.edu.cn
$^{\top}$\{zptu, shumingshi\}@tencent.com

Abstract

Language coverage bias, which indicates the content-dependent differences between sentence pairs originating from the source and target languages, is important for neural machine translation (NMT) because the target-original training data is not well exploited in current practice. By carefully designing experiments, we provide comprehensive analyses of the language coverage bias in the training data, and find that using only the source-original data achieves comparable performance with using full training data. Based on these observations, we further propose two simple and effective approaches to alleviate the language coverage bias problem through explicitly distinguishing between the source- and target-original training data, which consistently improve the performance over strong baselines on six WMT20 translation tasks. Complementary to the translationese effect, language coverage bias provides another explanation for the performance drop caused by back-translation (Marie et al., 2020). We also apply our approach to both back- and forward-translation and find that mitigating the language coverage bias can improve the performance of both the two representative data augmentation methods and their tagged variants (Caswell et al., 2019).

1 Introduction

In recent years, there has been a growing interest in investigating the effect of original languages in parallel data on neural machine translation (Barrault et al., 2020; Edunov et al., 2020; Marie et al., 2020). Several studies have shown that target-original test examples[1] can lead to distortions in automatic and human evaluations, which should be omitted from machine translation test sets (Barrault et al., 2019; Zhang and Toral, 2019; Graham et al., 2020). Another branch of studies report that target-original test data leads to discrepant conclusions: back-translation only benefits the translation of target-original test data while harms that of source-original test data (Edunov et al., 2020; Marie et al., 2020). They attribute these phenomena to the reason that human-translated texts (i.e., *translationese*) exhibit formal and stylistic differences that set them apart from the texts originally written in that language (Baker et al., 1993; Volansky et al., 2015; Zhang and Toral, 2019).

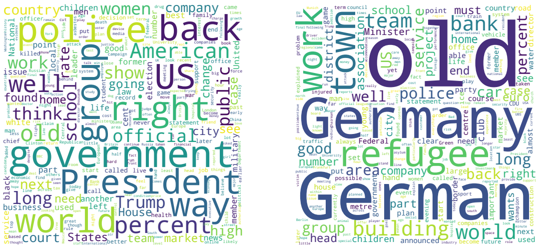
(a) English-Original (b) German-Original

Figure 1: Example of language coverage bias illustrated by word clouds that are plotted at the English side of sentence pairs in the En-De test sets from WMT10 to WMT18. The test sets consist of English-original and German-original sentence pairs.

Complementary to the translationese bias, which is *content-independent* (Volansky et al., 2015), we identify another important problem, namely *language coverage bias*, which refers to the *content-dependent* differences in data originating from different languages. These differences stem from the diversity of regions and cultures. While the degree of the translationese bias varies across different translators (Toral, 2019), language coverage bias is an intrinsic bias between the source- and target-original data, which is hardly affected by the ability of the translator. Figure 1 shows an example, where the contents in English- and German-original texts differ significantly due to language coverage bias.

---

[1] Target-original test examples are sentence pairs that are translated from the target language into the source language.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4778–4790
August 1–6, 2021.

To investigate the effect of language coverage bias in the training data on NMT models, we propose an automatic method to identify the original language of each training example, which is generally unknown in practical corpora. Experimental results on three large-scale translation corpora show that there exists a significant performance gap between NMT models trained on the source- and target-original data, which have different vocabulary distributions, especially for content words. Since the target-original training data performs poorly in translating content words, using only the source-original data achieves comparable performance with using full training data. These findings motivate us to explore other data utilization methods rather than indiscriminately mixing the source- and target-original training data.

We propose to alleviate the language coverage bias problem by explicitly distinguishing between the source- and target-original training data. Specifically, two simple and effective methods are employed: *bias-tagging* and *fine-tuning*. Experimental results show that both approaches consistently improve the performance on six WMT20 translation tasks. Language coverage bias also provides another explanation for the failure of back-translation on the source-original test data, complementary to the translationese effect (Marie et al., 2020). We further validate our approach in the monolingual data augmentation scenario, where the language coverage bias problem would be more severe due to the newly introduced monolingual data.

Contributions The main contributions of our work are listed as follows:

- We demonstrate the necessity of studying the language coverage bias for NMT, and identify that using the target-original data can cause poor translation adequacy on content words.
- We address the language coverage bias induced by the target-original data by explicitly distinguishing the original languages, which can significantly improve the translation performance on six WMT20 translation tasks.
- We show that alleviating the language coverage bias also benefits monolingual data augmentation, which can improve both back- and forward-translation and their tagged variants (Caswell et al., 2019).

2 Experimental Setup

Data We conducted experiments on six WMT20 benchmarks (Barrault et al., 2020), including English⇔German (En⇔De), English⇔Chinese (En⇔Zh), and English⇔Japanese (En⇔Ja) news translation tasks. The preprocessed training corpora contain 41.0M, 21.8M, and 13.0M sentence pairs for En⇔De, En⇔Zh, and En⇔Ja, respectively. We used the monolingual data that is publicly available in WMT20 to train the proposed original language detection model (Section 3.1) and data augmentation (Section 4.2). The Appendix lists details about the data preprocessing.

For En⇔De and En⇔Zh, we used newstest2019 as the validation sets. For En⇔Ja, we split the official validation set released by WMT20 into two parts by the original language and only used the corresponding part for each direction. We used newstest2020 as the test sets for all the six tasks. We reported the Sacre BLEU (Post, 2018), as recommended by WMT20.

Model We used the Transformer-Big (Vaswani et al., 2017) model, which consists of a 6-layer encoder and a 6-layer decoder, and the hidden size is 1024. Recent studies showed that training on large batches can further boost model performance (Ott et al., 2018; Wu et al., 2018). Accordingly, we followed their settings to train models with batches of approximately 460k tokens. Please refer to the Appendix for more details about model training. We followed Ng et al. (2019) to use the Transformer-Big decoder as our language models, which are used to detect the original language and measure translation fluency. Language models are also trained with large batches (Ott et al., 2018).

3 Observing Language Coverage Bias

In this study, we first establish the existence of language coverage bias (Section 3.2), and show how the bias affects NMT performance (Section 3.3). To this end, we propose an automatic method to detect the original language of each training example (Section 3.1), which is often not available in large-scale parallel corpora (Riley et al., 2020).

3.1 Detecting Original Languages

Detection Method Intuitively, we use a large-scale monolingual dataset to estimate the distribution of the contents covered by each language. For each training example, we compare its similarities

Method	En-Zh	En-Ja	En-De
FT	83.6	83.7	86.6
Ours	**84.4**	**91.5**	**88.7**

Table 1: F1 scores of detecting original languages in the test sets. "FT" denotes the forward translation classifier proposed by Riley et al. (2020).

to the distributions of source and target languages, based on which we determine its original language.

Formally, let $\mathcal{D}_s$ and $\mathcal{D}_t$ denote the source-side and target-side distributions of the covered contents. Given a training example $\langle \mathbf{x}, \mathbf{y} \rangle$, the probability that it is covered by one language (represented as $\mathcal{D}_s$ and $\mathcal{D}_t$) can be expressed as

$$P(\mathcal{D}_s|\langle \mathbf{x}, \mathbf{y} \rangle) = \frac{P(\mathcal{D}_s)P(\langle \mathbf{x}, \mathbf{y} \rangle|\mathcal{D}_s)}{P(\langle \mathbf{x}, \mathbf{y} \rangle)},$$
$$P(\mathcal{D}_t|\langle \mathbf{x}, \mathbf{y} \rangle) = \frac{P(\mathcal{D}_t)P(\langle \mathbf{x}, \mathbf{y} \rangle|\mathcal{D}_t)}{P(\langle \mathbf{x}, \mathbf{y} \rangle)}.$$

We use a score function to denote the difference between the two probabilities:

$$\begin{aligned} score &= \log P(\mathcal{D}_s|\langle \mathbf{x}, \mathbf{y} \rangle) - \log P(\mathcal{D}_t|\langle \mathbf{x}, \mathbf{y} \rangle), \\ &= \log P(\langle \mathbf{x}, \mathbf{y} \rangle|\mathcal{D}_s) - \log P(\langle \mathbf{x}, \mathbf{y} \rangle|\mathcal{D}_t) + c, \end{aligned}$$

where $c = \log P(\mathcal{D}_s) - \log P(\mathcal{D}_t)$, which has a constant value when the source and target monolingual datasets are given. Intuitively, examples with higher score values are more likely to be source-original while those with lower score values are more likely to be target-original data. We train language models $\boldsymbol{\theta}_s^{lm}$ and $\boldsymbol{\theta}_t^{lm}$ on the source- and target-language monolingual data to estimate the conditional probabilities:

$$P(\langle \mathbf{x}, \mathbf{y} \rangle|\mathcal{D}_s) = P(\mathbf{x}|\boldsymbol{\theta}_s^{lm}),$$
$$P(\langle \mathbf{x}, \mathbf{y} \rangle|\mathcal{D}_t) = P(\mathbf{y}|\boldsymbol{\theta}_t^{lm}).$$

Accordingly, the score can be rewritten as

$$score = \log P(\mathbf{x}|\boldsymbol{\theta}_s^{lm}) - \log P(\mathbf{y}|\boldsymbol{\theta}_t^{lm}) + c. \quad (1)$$

We label examples as source-original if their score values are positive, and the other examples as target-original. To find a specific constant for each language pair, we tune the value of c to obtain the best classification performance on the validation sets, where the original languages are known.

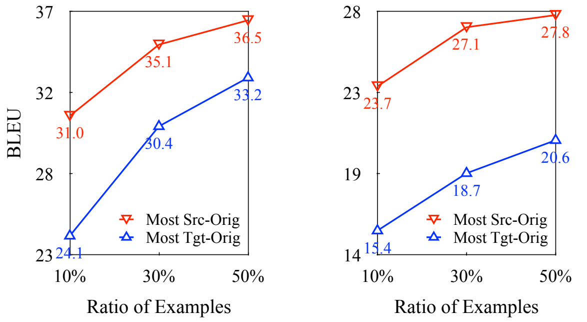

(a) En⇒Zh (b) Zh⇒En

Figure 2: Translation performance on the validation sets of the En⇔Zh translation task for different ratios of most source- and target-original training examples.

Detection Accuracy We evaluated the detection method on the mixture of the test sets of bidirectional translation tasks in WMT20 for each language pair. For comparison, we re-implemented the CNN-based forward-translation (FT) classifier proposed by Riley et al. (2020). The FT classifier and the language models used in our method were trained on the same monolingual data sets. Table 1 shows that our method outperforms the FT classifier in all language pairs. In addition, our model also outperforms the FT approach on detecting noisy training data, which leads to an improvement in translation performance (please refer to Table 11 in the Appendix for more results).

3.2 Existence of Language Coverage Bias

In this section, we validate the existence of language coverage bias by (1) comparing the performance of NMT models trained on data with different original languages, and (2) directly calculating the divergence between the vocabulary distributions of the source- and target-original data.

Translation Performance Once all the training examples are assigned a score by the detection method (Eq. (1)), we regard $R\%$ of examples with the highest scores as the most source-original examples, and $R\%$ of examples with the least scores as the most target-original examples. We investigate the effect of $R\%$ on translation performance, as shown in Figure 2. Clearly, using the most source-original examples significantly outperforms using its target-original counterparts, demonstrating that the source- and target-original data indeed differ greatly from each other. To rule out the effect of data scale, we treat 50% of data with the highest scores as source-original data, and the same amount

Data	JS ($\times 10^{-5}$)		
	All	*Content*	*Function*
Random	4	10	0
S vs T	745	1503	261

Table 2: JS divergence of the vocabulary distributions between the source- and target-original data ("S vs T") on the training set of WMT20 En⇔Zh. "*All*", "*Content*", and "*Function*" denote all words, content words, and function words respectively. For reference, we also report the JS divergence between randomly selected 50% examples and the others ("Random").

Data	**En-Zh**		**En-Ja**		**En-De**	
Origin	⇒	⇐	⇒	⇐	⇒	⇐
Target	33.2	20.6	30.5	15.4	39.3	37.4
Source	36.5	**27.8**	**35.3**	17.9	41.7	**42.5**
Both	**36.6**	27.5	34.9	**18.5**	**42.3**	42.2

Table 3: Sacre BLEU of using different sets of training data on validation sets. We highlight the **highest** score in bold and the second-highest score with underlines.

Data	**En⇒Zh**			**En⇐Zh**		
Origin	*noun*	*verb*	*adj*	*noun*	*verb*	*adj*
Target	67.6	52.0	64.3	53.8	38.0	57.0
Source	69.7	54.0	**66.2**	**61.8**	**44.1**	**63.9**
Both	**69.9**	**54.1**	65.9	61.2	43.8	63.4

Table 4: Translation adequacy of different types of content words measured by F-measure (Neubig et al., 2019). The results are reported on the validation sets.

of data with the least scores as target-original data in the following experiments by default.

Since some recent works find that BLEU might be affected by the translationese problem (Edunov et al., 2020; Freitag et al., 2020), we have also conducted a side-by-side human-evaluation on the Zh⇒En development set, where 500 randomly sampled examples were evaluated by six persons (agreement (Fleiss, 1971): Fleiss' Kappa=0.46). 37.0% of outputs using the source-original data are better than using target-original data, and 21.0% are worse. By manually checking the outputs, we find using only the target-original data tends to omit important source contents (e.g., named entities) either by totally ignoring some contents or by using pronouns instead. The human-evaluation shows the same trend with the BLEU score presented in Figure 2. Given that conducting human-evaluation on all the six translation tasks is time-consuming and labor-intensive, we use automatic measures to further investigate this problem in Section 3.3.

Vocabulary Distributions Complementary to previous studies that focus on the *content-independent* stylistic difference (Volansky et al., 2015) between translationese and original texts (Riley et al., 2020; Edunov et al., 2020; Marie et al., 2020), we investigate the *content-dependent* language coverage bias between the source- and target-original data in this experiment. Intuitively, if the language coverage bias exists, the vocabulary distributions of the source- and target-original data should differ greatly from each other, since the covered issues tend to have different frequencies between them (D'Alessio and Allen, 2000). We use the Jensen-Shannon (JS) divergence (Lin, 1991) to measure the difference between two vocabulary distributions p and q:

$$\mathrm{JS}\,(p||q) = \frac{1}{2}\left(\mathrm{KL}(p||\frac{p+q}{2}) + \mathrm{KL}(q||\frac{p+q}{2})\right),$$

where $\mathrm{KL}(\cdot||\cdot)$ is the KL divergence (Kullback and Leibler, 1951) of two distributions.

Table 2 shows the JS divergence of the vocabulary distributions between the source- and target-original data. We also divide the words into content words and functions words based on their POS tags, since content words are more related to the language coverage bias, while the function words are more related to the stylistic and structural differences between the translationese and original texts (Lembersky et al., 2011; Volansky et al., 2015). The JS divergence between the source- and target-original data are 186× larger than that between randomly split data, which is mainly due to the difference between content words. Results for different ratios $R\%$ and other language pairs can be found in Appendix (Tables 12 and 13), where the trend holds in all cases, supporting our claim of the existence of language coverage bias.

3.3 Effect of Language Coverage Bias

In this section, we investigate the effect of language coverage bias on NMT models.

Using only the source-original data achieves comparable performance with using full data. Table 3 lists the translation performances of NMT models trained on only the source- or target-original data and on both of them. The results show that using only the source-original data significantly outperforms using the target-original data in all language pairs, which reconfirm the necessity of studying the language coverage bias for NMT. It should be emphasized that using only the source-original data (i.e. 50% of the whole training set) achieves translation performances on par with using full training data. In the following experiments, we investigate why using target-original data together cannot further improve the performance.

Using additional target-original data does not consistently improve translation adequacy. To rule out the effect of translationese and focus on the content-dependent difference caused by the language coverage bias, we examine the translation adequacy of content words in Table 4 [2]. We follow Raunak et al. (2020) to use F-measure (Neubig et al., 2019) to quantify the translation accuracy of specific types of words.

Compared with the source-original data, using only the target-original data greatly reduces the translation accuracy of content words, which we attribute to the divergence of the content word distributions between the source- and target-original data. The results also indicate that indiscriminately using all the training data can not consistently improve the translation adequacy of content words over using only source-original data, and in some cases using all the data is even harmful to the adequacy on content words. Table 5 shows an example, which suggests that using only the target-original data tends to omit content words. This problem is potentially caused by that some content words at the source-side are less or even not visible in the target-original data, and indiscriminately adding target-original data induces a distribution shift on the content word distribution.

Using additional target-original data only slightly improves the structural fluency. Recently, Edunov et al. (2020) claim that using additional back-translated data can improve translation fluency. Target-original bilingual data is similar to back-translated data since both of them are constructed by translating sentences from the target language into the source language. One question naturally arises: *can target-original bilingual data improve the fluency of NMT models?*

Input	大闸蟹是巴城最为知名的形象代言人。
Refer.	The hairy crab is the most famous image spokesperson in Bacheng.
Target Orig.	It is one of the city's most well-known image spokesmen.
Source Orig.	Hairy crabs are the most well-known image spokesmen of Bacheng.
Both	It is the best-known icon of Bacheng.

Table 5: An example of the outputs of NMT models trained on different sets of data. Using the target-original data tends to omit content words.

To answer the above question, we measure the fluency of outputs with language models trained on the monolingual data as described in Section 2. Previous study finds that different perplexities could be caused by specific contents rather than structural differences (Lembersky et al., 2011). Specifically, some source-original contents are of low frequency in the target-language monolingual data (e.g., "Bacheng" in Table 5), thus the language model trained on the target-language monolingual data tends to assign higher perplexities to outputs containing more source-original content words. To rule out this possibility and check whether the outputs are structurally different, we follow Lembersky et al. (2011) to abstract away from the content-specific features of the outputs to measure their fluency at the syntactic level. Table 6 shows the results. Although using only the source-original data results in high perplexities measured by vanilla language models, the perplexities of NMT models trained on different data are close to each other at the syntactic level. Using additional target-original data only slightly reduces the perplexity at the syntactic level over using only the source-original data.

4 Addressing Language Coverage Bias

In Section 3 we show that the target-original data performs poorly in translating content words due to the language coverage bias. Accordingly, simply using the full training data without distinguishing the original languages is sub-optimal for model training. Based on these findings, we propose to address the language coverage bias by explicitly distinguishing between the source- and the target-original data (Section 4.1). We then investigate whether the performance improvement still holds in

---

[2] We only list the results on En⇔Zh due to space limit. Please refer to Table 14 in the Appendix for the translation quality on other language pairs.

Data Origin	No Abs. PPL	No Abs. Diff.	Cont. Abs. PPL	Cont. Abs. Diff.
WMT20 En⇒Zh				
Target	38.4	−6.3%	14.0	−2.8%
Source	44.0	+7.3%	14.6	+1.4%
Both	41.0	-	14.4	-
WMT20 En⇐Zh				
Target	25.9	−5.8%	13.4	−3.6%
Source	31.0	+12.7%	14.2	+2.2%
Both	27.5	-	13.9	-

Table 6: Translation fluency measured by the perplexities (i.e., PPL) of language models with different levels of lexical abstraction, "Diff." means the relative change with respect to "Both". "No Abs." denotes no abstraction (i.e., vanilla LM), "Cont. Abs." denotes abstracting all content words with their corresponding POS tags. The results are reported on the validation sets.

the monolingual data augmentation scenario (Section 4.2), where the language coverage bias problem is more severe due to the newly introduced dataset in source or target language.

4.1 Bilingual Data Utilization

In this section, we aim to improve bilingual data utilization through explicitly distinguishing between the source- and target-original training data.

Methodology We distinguish original languages with two simple and effective methods:

- *Bias-Tagging*: Tagging is a commonly-used approach to distinguishing between different types of examples, such as different languages (Aharoni et al., 2019; Riley et al., 2020) and synthetic vs authentic examples (Caswell et al., 2019). In this work, we attach a special tag to the source side of each target-original example, which enables NMT models to distinguish it from the source-original ones in training.

- *Fine-Tuning*: Fine-tuning (Luong and Manning, 2015) is a useful method to help knowledge transmit among data from different distributions. We pre-train NMT models on the full training data that consists of both the source- and target-original data, and then fine-tune them on only the source-original data. For fair comparison, the total training steps of the pre-training and fine-tuning stages are the same as the baseline.

Translation Performance Table 7 depicts the results on the benchmarking datasets. For comparison, we also list the results of several baselines using the vanilla Transformer architecture trained on the constrained bilingual data in the WMT20 competition (Barrault et al., 2020). Clearly, both the bias tagging and fine-tuning approaches consistently improve translation performance on all benchmarks, which confirms our claim of the necessity of explicitly distinguishing target-original examples in model training.

Analysis Recent studies have shown that generating human-translation like texts as opposed to original texts can improve the BLEU score (Riley et al., 2020). To validate that the improvement is partially from alleviating the content-dependent language coverage bias, we examine the translation adequacy of content words on the test sets, as listed in Table 8. The results indicate that explicitly distinguishing between the source- and target-original data improves the translation of content words (e.g., nouns), which is closely related to the language coverage bias problem. Table 9 lists the translation fluency at the syntactic level, where the proposed approaches maintain the syntactic fluency.

4.2 Monolingual Data Augmentation

In this section, we aim to provide some insights where monolingual data augmentation improves translation performance, and investigate whether our approach can further improve model performance in this scenario that potentially suffers more from the language coverage bias problem.

For fair comparison across language pairs, we augment NMT models with the same English monolingual corpus as described in Section 2. We down-sample the large-scale monolingual corpus to the same amount as that of the bilingual corpus in each language pair, in order to rule out the effect of the scale of synthetic data (Edunov et al., 2018; Fadaee and Monz, 2018). We use back-translation (Sennrich et al., 2016a) to augment the English monolingual data for the task of translating from another language to English ("X⇒En"), and use forward-translation for the task in the opposite translation direction ("En⇒X"). Table 10 lists the results, where several observations can be made.

Explaining Data Augmentation with Language Coverage Bias Concerning the monolingual data augmentation methods (Rows 3-4), the vanilla

Method	En-Zh		En-Ja		En-De		Average
	⇒	⇐	⇒	⇐	⇒	⇐	
WMT20 Systems							
Shi et al. (2020)	38.6	28.8	-	-	-	-	-
Zhang et al. (2020)	40.8	-	34.8	20.4	-	-	-
Molchanov (2020)	-	-	-	-	31.9	39.6	-
Our Implemented Systems							
Baseline	42.3	28.4	35.8	20.9	32.3	41.4	33.5
Tag	**43.4**$^{\Uparrow}$	29.2$^{\uparrow}$	36.3	**21.9**$^{\Uparrow}$	32.7	**42.5**$^{\Uparrow}$	34.3
Tune	43.3$^{\Uparrow}$	**29.7**$^{\Uparrow}$	**36.6**$^{\uparrow}$	21.8$^{\uparrow}$	**32.9**$^{\uparrow}$	42.2$^{\uparrow}$	**34.4**

Table 7: Sacre BLEU reported on the WMT20 test sets. "Tag" and "Tune" denote the bias-tagging and fine-tuning, respectively. We highlight the **highest** score in bold and the second-highest score with underlines. "$\uparrow$/$\Uparrow$" denotes significantly better than the baseline with $p < 0.05$ and $p < 0.01$, respectively. For comparison, we list three systems that use vanilla Transformer models trained on the bilingual data in the WMT20 competition.

Method	En⇒Zh			En⇐Zh		
	noun	*verb*	*adj*	*noun*	*verb*	*adj*
Baseline	70.7	61.0	67.9	60.2	43.6	61.4
Tag	**72.3**	**62.3**	67.9	60.7	43.9	**62.2**
Tune	71.8	61.9	**68.4**	**61.1**	**44.1**	**62.2**

Table 8: F-measure of different types of content words on the WMT20 En⇔Zh test sets. Results on other languages can be found in Appendix (Table 15).

Data	En-Zh		En-Ja		En-De	
Origin	⇒	⇐	⇒	⇐	⇒	⇐
Baseline	**13.7**	**13.4**	**17.4**	**15.4**	**16.3**	**17.8**
Tag	**13.7**	**13.4**	17.5	**15.4**	16.4	**17.8**
Tune	13.8	**13.4**	**17.4**	**15.4**	**16.3**	17.9

Table 9: PPL at the syntactic level on the test sets. We abstract the content words to rule out the language coverage bias when measuring fluency.

back-translation (Row 3) harms the translation performance on average, while the vanilla forward-translation improves the performance, which is consistent with the findings in previous studies (Edunov et al., 2020; Marie et al., 2020). Caswell et al. (2019) have shown that the tagging strategy works for back-translation while fails for forward-translation, and our results confirm these findings. Both phenomena can be attributed in part to the language coverage bias problem. Back-translated data originates from the target language, and thus suffers more from the language coverage bias problem. Accordingly, directly using the back-translated data is sub-optimal, while tagged back-translation recovers translation performance by distinguishing training examples with different origins, which is consistent with our results in Table 7. In contrast, the language coverage bias problem does not exist for source-side monolingual data (i.e. the same original language). Therefore, the vanilla forward-translation can improve translation performance, while tagged forward-translation performs worse.

Improving Data Augmentation Our approach (Row 2) achieves comparable improvements of translation performance with the monolingual data augmentation approaches (e.g. averaged BLEU: 31.2 vs. 30.7, and 37.6 vs. 37.9), while we do not use additional monolingual data to train the models.[3] Combining them can further improve performance (Rows 5-6), indicating that the two types of approaches are complementary to each other. This is straightforward, since our approach better exploits the bilingual data, while data augmentation introduces new knowledge from additional monolingual data. In addition, our approach consistently improves performance over both vanilla and tagged augmentation approaches, making it more robust in practical application across datasets.

5 Related Work

Our work is inspired by three lines of research in the NMT community.

---

[3] The monolingual data is only used to detect the original languages of training data and is invisible in model training.

#	Monolingual		Bilingual	X⇒En				En⇒X			
	Data	*Tagging*	*Fine-Tune*	*Zh*	*Ja*	*De*	*Ave.*	*Zh*	*Ja*	*De*	*Ave.*
1	×	n/a	×	28.4	20.9	41.4	30.2	42.3	35.8	32.3	36.8
2			✓	*29.7*	*21.8*	*42.2*	*31.2*	*43.3*	*36.6*	*32.9*	*37.6*
3	✓	×	×	28.9	21.2	38.1	29.4	*44.2*	*36.8*	*32.8*	*37.9*
4		✓		*29.4*	*21.2*	*41.5*	*30.7*	43.1	36.4	32.3	37.3
5		×	✓	30.4	22.1	42.3	31.6	**45.1**	**37.6**	**33.4**	**38.7**
6		✓		**30.6**	**22.2**	**42.7**	**31.8**	44.7	36.9	33.3	38.3

Table 10: Translation performance of augmenting English monolingual data with different strategies: back-translation for X⇒En tasks (blue cells), and forward-translation for En⇒X tasks (red cells). "Tagging" denotes adding a special tag to each synthetic sentence pair (Caswell et al., 2019). "Fine-Tune" denotes fine-tuning the pre-trained NMT models on the source-original bilingual data, as described in Section 4.1.

5.1 Translationese

Recently, the effect of translationese in NMT evaluation has attracted increasing attention (Zhang and Toral, 2019; Bogoychev and Sennrich, 2019; Edunov et al., 2020; Graham et al., 2020). Graham et al. (2020) show that the source-side translationese texts can potentially lead to distortions in automatic and human evaluations. Accordingly, the WMT competition starts to use only source-original test sets for most translation directions since 2019. Our study reconfirms the necessity of distinguishing the source- and target-original examples and takes one step further to distinguish examples in training data. Complementary to previous works, we investigate the effect of language coverage bias on machine translation, which is related to the content bias rather than the language style difference. Shen et al. (2021) also reveal the context mismatch between texts from different original languages. To alleviate this problem, they proposed to combine back- and forward-translation by introducing additional monolingual data, while we focus on better exploiting bilingual data by distinguishing the original languages, which is also helpful for back- and forward-translation.

Lembersky et al. (2011, 2012) propose to adapt machine translation systems to generate texts that are more similar to human-translations, while Riley et al. (2020) propose to model human-translated texts and original texts as separate languages in a multilingual model and perform zero-shot translation between original texts. Riley et al. (2020) and our work both aim to better utilize the bilingual training data. They aim to guide NMT models to produce original text, while we focus on improving translation adequacy by alleviating the language coverage bias problem.

5.2 Data Augmentation

Concerning model training, recent works find that back-translation can harm the translation of source-original test set, and attribute the quality drop to the stylistic and content-independent differences between translationese and original texts (Edunov et al., 2020; Marie et al., 2020). In this work, we empirically show that language coverage bias is another reason for the performance drop of back-translation, as well as the different performances between tagged forward-translation and tagged back-translation (Caswell et al., 2019). In addition, we show that our approach is also beneficial for data augmentation approaches, which can further improve the translation performance over both back-translation and forward-translation.

5.3 Domain Adaptation

Since high-quality and domain-specific parallel data is usually scarce or even unavailable, domain adaptation approaches are generally employed for translation in low-resource domains by leveraging out-of-domain data (Chu and Wang, 2018). Languages can be also regarded as different domains, since articles in different languages cover different topics (Bogoychev and Sennrich, 2019). Starting from this intuition, we distinguish examples with different original languages with tagging (Aharoni et al., 2019) and fine-tuning (Luong and Manning, 2015), which are commonly-used in domain adaptation and multi-lingual translation tasks.

Our work also benefits domain adaptation: distinguishing original languages in general domain data consistently improves translation performance of NMT models in several specific domains (Table 16 in Appendix), making these models better start points for further domain adaptation.

6 Conclusion and Future Work

In this work, we first systematically examine why the language coverage bias problem is important for NMT models. We conducted extensive experiments on six WMT20 translation benchmarks. Empirically, we find that source-original data and target-original data differ significantly at the text content, and using target-original data together without discrimination is sub-optimal. Based on these observations, we propose two simple and effective approaches to distinguish the source- and target-original training data, which obtain consistent improvements in all benchmarks.

Furthermore, we link language coverage bias to two well-known problems in monolingual data augmentation, namely the performance drop of back-translation, and the different behaviors between tagged back-translation and tagged forward-translation. We show that language coverage bias can be considered as another reason for these problems, and fine-tuning on the source-original bilingual training data can further improve performance over both back- and forward-translation.

Future directions include exploring advanced methods to better alleviate the language coverage bias problem, as well as validating on other language pairs. It is also interesting to investigate the language coverage bias problem in multilingual translation, where we can better understand this problem by considering language family.

Acknowledgments

We thank all anonymous reviewers for their insightful comments and suggestions for this work. We also sincerely thank Cunxiao Du and Wenxiang Jiao for their valuable help and advice. This work was supported by the National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No. 61925601, No. 61772302), and the Tencent AI Lab Rhino-Bird Focused Research Program.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *NAACL*.

Mona Baker, Gill Francis, and Elena Tognini-Bonelli. 1993. Corpus linguistics and translation studies: Implications and applications. *Text and technology: In honour of John Sinclair*, 233:250.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, et al. 2020. Findings of the 2020 conference on machine translation (wmt20). In *WMT*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *WMT*.

Nikolay Bogoychev and Rico Sennrich. 2019. Domain, translationese and noise in synthetic data for neural machine translation. *arXiv preprint arXiv:1911.03362*.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *WMT*.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *COLING*.

Dave D'Alessio and Mike Allen. 2000. Media bias in presidential elections: A meta-analysis. *Journal of communication*, 50(4):133–156.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2020. On the evaluation of machine translation systems trained with back-translation. In *ACL*.

Marzieh Fadaee and Christof Monz. 2018. Back-translation sampling by targeting difficult words in neural machine translation. In *EMNLP*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Markus Freitag, David Grangier, and Isaac Caswell. 2020. BLEU might be guilty but references are not innocent. In *EMNLP*.

Yvette Graham, Barry Haddow, and Philipp Koehn. 2020. Statistical power and translationese in machine translation evaluation. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2011. Language models for machine translation: Original vs. translated texts. In *EMNLP*.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2012. Adapting translation models to translationese improves smt. In *EACL*.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *IWSLT*.

Benjamin Marie, Raphael Rubino, and Atsushi Fujita. 2020. Tagged back-translation revisited: Why does it really work? In *ACL*.

Alexander Molchanov. 2020. Promt systems for wmt 2020 shared news translation task. In *WMT*.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A tool for holistic comparison of language generation systems. In *NAACL*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *WMT*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *WMT*.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *WMT*.

Vikas Raunak, Siddharth Dalmia, Vivek Gupta, and Florian Metze. 2020. On long-tailed phenomena in neural machine translation. In *Findings of EMNLP*.

Parker Riley, Isaac Caswell, Markus Freitag, and David Grangier. 2020. Translationese as a language in "multilingual" nmt. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL*.

Jiajun Shen, Peng-Jen Chen, Matthew Le, Junxian He, Jiatao Gu, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2021. The source-target domain mismatch problem in machine translation. In *EACL*.

Tingxun Shi, Shiyu Zhao, Xiaopu Li, Xiaoxue Wang, Qian Zhang, Di Ai, Dawei Dang, Zhengshan Xue, and Jie Hao. 2020. Oppo's machine translation systems for wmt20. In *WMT*.

Antonio Toral. 2019. Post-editese: an exacerbated translationese. In *MT Summit*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities*, 30(1):98–118.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2018. Pay less attention with lightweight and dynamic convolutions. In *ICLR*.

Mike Zhang and Antonio Toral. 2019. The effect of translationese in machine translation test sets. In *WMT*.

Yuhao Zhang, Ziyang Wang, Runzhe Cao, Binghao Wei, Weiqiao Shan, Shuhan Zhou, Abudurexiti Reheman, Tao Zhou, Xin Zeng, Laohu Wang, Xiaoqian Liu, Xunjuan Zhou, Yongyu Mu, Jingnan Zhang, Yinqiao Li, Bei Li, Tong Xiao, and Jingbo Zhu. 2020. The niutrans machine translation systems for wmt20. In *WMT*.

A Appendices

A.1 Data Preprocessing

We used all the parallel corpora provided by WMT20 and filtered sentences that are longer than 250 words. We tokenized English and German sentences with Moses (Koehn et al., 2007), and segmented Chinese and Japanese sentences with Jieba[4] and Mecab[5] respectively. We employed Byte pair encoding (BPE) (Sennrich et al., 2016b) with 32K merge operations for all language pairs. Specifically, we jointly trained the BPE code on both sides in En⇔De and independently learned the BPE code on each side in En⇔Zh and En⇔Ja.

As for the monolingual data, we combined the newscrawl data from 2017 to 2019 for English and German. Since the newscrawl corpora for Chinese and Japanese are significantly smaller, we augmented these two languages with the commoncrawl corpus. We preprocessed the monolingual data with the same rules as parallel data. Finally, we randomly selected 41.0M sentences for each language (i.e., En, De, Zh, Ja), which were used to train the language detection models. For data augmentation, to rule out the effect of the ratio between synthetic and authentic data, we downsampled the monolingual data to the same amount as the bilingual data for each language pair.

We used spaCy[6] to perform the Part-Of-Speech (POS) tagging for each language. Nouns, verbs, and adjectives belong to content words and the others belong to function words.

A.2 More Details of Model Training

In this work, we generally followed the default hyper-parameters used in Vaswani et al. (2017) except the batch size. Recent studies showed that training on large batches can further boost model performance (Ott et al., 2018; Wu et al., 2018). Accordingly, we followed them to train models with batches of approximately 460k tokens, using Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-8}$. We used the same cosine learning rate schedule as Wu et al. (2018), where the learning rate was warmed up linearly in the first 10K steps, and then decayed following a cosine rate within a single cycle. By default, NMT models and language models were both trained for 30k steps with the aforementioned batch size. Each model was trained using 8 NVIDIA V100 GPUs for about 20 hours.

A.3 Effect of Detection Methods on Translation Performance

To further compare our proposed original language detection method and the FT classifier (Riley et al., 2020), we fine-tune the NMT model pre-trained on the whole training set using the source-original data detected by the two methods. Note that the two detection methods are developed using the same monolingual data sets. For fair comparison, the fine-tuning sets are of the same amount (50% of the whole training set) between the two methods in this experiment. Table 11 lists the results, indicating that our method performs better in detecting original languages in large-scale parallel data.

Fine-Tune Data	BLEU
×	27.5
FT	27.8
Ours	**28.4**

Table 11: Effect of original language detection methods. The results are reported on the validation set of the Zh⇒En translation task.

A.4 Divergence of Vocabulary Distributions

In this section, we report the JS divergence of the vocabulary distributions in more cases. Table 12 lists the results for different ratios $R\%$ on En⇔Zh, and Table 13 shows the results on all language pairs. The results show that the divergence of vocabulary distributions between the source- and target-original data is substantially larger than that between randomly split data, which reconfirms the existence of language coverage bias.

A.5 Effect of Language Coverage Bias for Other Language Pairs

Table 14 lists the translation adequacy of NMT models trained on only the source- or target-original data and on both of them. The results are reported on En⇔De and En⇔Ja, which exhibit the same trend as that on En⇔Zh (Table 4 in the main paper), indicating that the target-original data performs poorly in translating content words.

---

[4] https://github.com/fxsjy/jieba

[5] https://taku910.github.io/mecab

[6] https://spacy.io

Data	**10%**			**30%**			**50%**		
	All	*Content*	*Function*	*All*	*Content*	*Function*	*All*	*Content*	*Function*
Random	20	51	0	7	18	0	4	10	0
S vs T	2660	5096	961	1371	2731	486	745	1503	261

Table 12: JS divergence ($\times 10^{-5}$) of the vocabulary distributions between the source- and target-original training data ("S vs T") for different labeled ratios on En⇔Zh. For reference, we also report the JS divergence between two sets of randomly selected examples ("Random", non-overlap).

Data	**En-Zh**			**En-Ja**			**En-De**		
	All	*Content*	*Function*	*All*	*Content*	*Function*	*All*	*Content*	*Function*
Random	4	10	0	8	17	0	2	4	0
S vs T	745	1503	261	1687	2910	666	870	1622	250

Table 13: JS divergence ($\times 10^{-5}$) of the vocabulary distributions between the source- and target-original training data for different language pairs. 50% examples are treated as source-original and the others are treated as target-original. For reference, we also report the JS divergence between randomly selected 50% examples and the others ("Random", non-overlap).

Data	**En⇒Ja**			**En⇐Ja**			**En⇒De**			**En⇐De**		
Origin	*noun*	*verb*	*adj*	*noun*	*verb*	*adj*	*noun*	*verb*	*adj*	*noun*	*verb*	*adj*
Target	60.9	47.7	62.1	44.5	29.8	46.2	70.5	54.3	58.4	70.8	53.6	67.0
Source	**61.4**	**51.8**	**63.5**	49.5	31.8	50.1	72.1	55.0	60.3	**75.3**	55.3	70.2
Both	61.3	51.7	63.2	**50.7**	**32.1**	**50.4**	**72.7**	**56.6**	**60.4**	74.9	**55.7**	**71.0**

Table 14: Translation adequacy of different types of content words measured by F-measure (Neubig et al., 2019). The results are reported on the validation sets.

Method	**En⇒Ja**			**En⇐Ja**			**En⇒De**			**En⇐De**		
	noun	*verb*	*adj*	*noun*	*verb*	*adj*	*noun*	*verb*	*adj*	*noun*	*verb*	*adj*
Baseline	62.0	53.0	59.1	54.0	35.5	50.4	66.7	48.1	53.6	75.0	54.0	70.2
Tag	62.5	53.3	61.1	**55.7**	36.5	**52.4**	**67.1**	48.6	**54.0**	**76.1**	54.4	70.7
Tune	**62.8**	**53.7**	**61.7**	55.3	**36.9**	51.8	**67.1**	**48.7**	**54.0**	75.7	**54.8**	**70.8**

Table 15: Translation adequacy of different types of content words measured by F-measure (Neubig et al., 2019). The results are reported on the test sets.

Domain	**Baseline**	**Ours**
Business	40.4	**40.8**
Crime	34.8	**35.5**
Entertainment	28.8	**30.0**
Politics	39.5	**40.3**
Sci-Tech	38.2	**39.9**
Sport	**31.5**	**31.5**
World	38.8	**38.9**
Overall	36.6	**37.2**

Table 16: Transformer performance on the validation set of the En⇒Zh task. We split the whole validation set into several parts by the domain tag. “Ours” denotes the “Bias-Tagging” approach as described in Section 4.1. The results indicate that distinguishing data with different original languages in the general domain training data can improve the performance of NMT models in many specific domains, making the models better start points for further domain adaptation.

A.6 Translation Adequacy on Test Sets for Other Language Pairs

We report the translation adequacy on test sets for En⇔De and En⇔Ja in Table 15, corresponding to Table 8 in the main paper. The results show that explicitly distinguishing the source- and target-original training data can consistently improve the translation adequacy for content words on all the six translation tasks.

A.7 Translation Performance in Specific Domains

We evaluate NMT models trained with and without explicit distinguishing between the source- and target-original data in several specific domains. The results are shown in Table 16, suggesting that our method can improve the translation performance of NMT models in several specific domains, which can be combined with further domain adaptation approaches.

Named Entity Recognition via Noise Aware Training Mechanism with Data Filter

Xiusheng Huang[1,2], **Yubo Chen**[1,2], **Shun Wu**[1], **Jun Zhao**[1,2],
Yuantao Xie[3] and **Weijian Sun**[3]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]Huawei Technologies Co., Ltd, Shenzhen, China
huangxiusheng2020@ia.ac.cn,
{yubo.chen,shun.wu,jzhao}@nlpr.ia.ac.cn,
{xieyuantao2,sunweijian}@huawei.com

Abstract

Named entity recognition (NER) is a fundamental task in natural language processing, these is a long held belief that datasets benefit the model. However, not all the data help with generalization, and some samples may contain ambiguous entities or noisy labels. The existing methods can not distinguish hard samples from noisy samples well, and becomes particularly challenging in the case of overfitting. This paper proposes a new method called Noise-Aware-with-Filter (NAF) to solve the issues from two sides. From the perspective of the data, we design a Logit-Maximum-Difference (LMD) mechanism, which maximizes the diversity between different samples to help the model identify noisy samples. From the perspective of the model, we design an Incomplete-Trust (In-trust) loss function, which boosts L_{CRF} with a robust Distrust-Cross-Entropy(DCE) term. Our proposed In-trust can effectively alleviate the overfitting caused by previous loss function. Experiments on six real-world Chinese and English NER datasets show that NAF outperforms the previous methods, and which obtained the state-of-the-art(SOTA) results on the CoNLL2003 and CoNLL++ datasets.

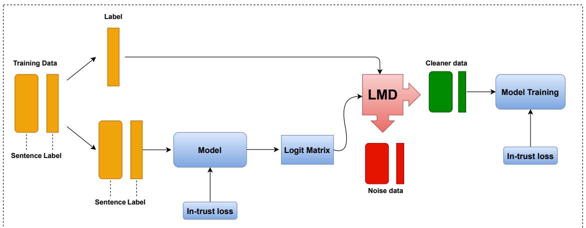

Figure 1: An overview of Noise-Aware-with-Filter(NAF). In-trust loss prevents model from overfitting and helps model generates logit matrices, which enters LMD mechanism with labels to filter noise data and get a cleaner data for model training.

1 Introduction

Named entity recognition (NER) is a primary task and which identifies both types and spans in sentences. NER models are becoming more and more accurate in prediction tasks, the potential improvement of existing architectures in real-world applications is often inherently limited by data quality (Pleiss et al., 2020). However, not all samples are completely correct in the NER datasets (Nooralahzadeh et al., 2019; Lange et al., 2019). Many real-world datasets generally contain samples which are "weakly-labeled" (Derczynski et al., 2017; Peng and Dredze, 2015). Specifically, some datasets which are annotated based on distant supervision (Yang et al., 2018; Liang et al., 2020) contain more noise labels, and manual annotators, especially on crowdsourcing platform, are prone to making labeling mistakes. Meanwhile labeling a huge datasets is an expensive and fallible process.

As the increase of training iteration epochs, the model will overfit noisy samples and hinder the generalization of the model (Pleiss et al., 2020). In NER task, it is impractical to get an absolutely clean dataset, and the existing datasets generally exist mislabeled samples (Flor et al., 2019) and ambiguous entity (Nadeau et al., 2006), even if some classical datasets (e.g., CoNLL2003 (Tjong Kim Sang and De Meulder, 2003)) still contain noisy samples (Wang et al., 2019b). It makes sense to obtain a cleaner dataset, but it would be very difficult to correct these real-world datasets manually, and the existing methods can't solve the issues automatically , especially in the case of existing ambiguous entities in sentences (Wang et al., 2019b).

From the CoNLL2003 NER dataset, we divide the samples into **Easy samples** which are correctly labeled and do not contain ambiguous entities, **Hard samples** are correctly labeled but contain ambiguous entities, and **Noisy samples** are misla-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4791–4803
August 1–6, 2021.

beled (Wang et al., 2019b). Such as:

- **Easy samples:{** third and final test between [England]{LOC} and [Pakistan]{LOC} at [the]{LOC}[Oval]{LOC} on **}**. The sample is labeled correctly and there is no ambiguous entity.
- **Hard samples:{** [Chicago]{ORG} won Game 1 with Derrick Rose scoring 25 points **}**. [Chicago] is a basketball team in NBA which is correctly marked as [Chicago]{ORG} here. Meanwhile [Chicago] is also a city in the United States, it is easy to mark that as [Chicago]{LOC} due to ambiguity.
- **Noisy samples:{** Soccer - [Japan]]{LOC} get lucky win, [China]{PER} in surprise defeat **}**. The [China]{PER} is mislabeled.

We can easily obtain the boundary between easy samples and noisy samples with utilizing loss values (Lin et al., 2017), but distinguishing hard samples from noisy samples still is a challenge (Wang et al., 2019b; Pleiss et al., 2020), and becomes particularly challenging in the case of overfitting (Wang et al., 2019b; Liu et al., 2020).

We propose a new method called Noise-Aware-with-Filter (NAF) to solve the issues from two sides. From the perspective of data, we design a Logit-Maximum-Difference (LMD) mechanism, which maximizes the diversity between different samples to help the model identify noisy samples. The difference between easy samples and noisy samples is very obvious in LMD score, meanwhile hard samples and noisy samples also can be well distinguished. From the perspective of model, we propose a noise tolerant term named Distrust-Cross-Entropy(DCE), which combines with L_{CRF} form the basis of the approach Incomplete-Trust (In-trust) loss function. In-trust not only improves the robustness of the model, but also helps LMD improve the accuracy of identifying noisy samples. Experiments on six real-world Chinese and English datasets show that NAF is more accurate than other methods in identifying noisy samples, meanwhile the datasets after filtering are cleaner.

In summary, our major contributions are the following:

- We propose a new method called Noise-Aware-with-Filter (NAF) to distinguish hard samples from noisy samples especially in the case of overfitting.
- To distinguish hard samples from noisy samples, we design a Logit-Maximum-Difference (LMD) mechanism. Meanwhile to alleviate the negative impact of overfitting, we propose Incomplete-Trust (In-trust) loss function, which utilizes both the incomplete correctness of labels and the relative correctness of the model output.
- We conduct extensive experiments on six real-world Chinese and English NER datasets show that NAF outperforms the previous methods, and which obtains the state-of-the-art(SOTA) results on the CoNLL2003 and CoNLL++ datasets. We release the source code publicly for further research [1].

2 Related Work

There are various approaches have been proposed to obtain a robust model. We summarize them into three categories: 1) Robust loss methods, 2) Training architectures methods, 3) Label correction methods.

Robust loss methods specifically design robust loss functions. They include Mean Absolute Error (MAE) (Ghosh et al., 2017), Improved MAE (Wang et al., 2019a) which is a reweighted MAE. Symmetric cross entropy (Wang et al., 2019b), by adding a symmetric reverse cross entropy after the cross entropy, makes the model have a certain noise tolerance, and Generalized cross entropy (Zhang and Sabuncu, 2018) is actually a new evolutionary form of MAE. Regularization (LSR) (Szegedy et al., 2016) is a technique using soft labels in place of one-hot labels to alleviate overfitting to noisy labels. ELR (Liu et al., 2020) is a kind of method that makes full use of early learning phenomenon to keep a large learning gradient for clean samples. But these methods can not effectively distinguish hard samples from noisy samples, and which are easy to confuse them.

Training architectures methods identify noisy samples from the perspective of model framework. Co-Tearching (Han et al., 2018; Yu et al., 2019) utilizes "early learning" phenomenon to maintain two networks in the process of training. All samples are sorted based on the loss values, and the noisy

---

[1] https://github.com/Huangxiusheng/Named-Entity-Recognition-via-Noise-Aware-Training-Mechanism-with-Data-Filter

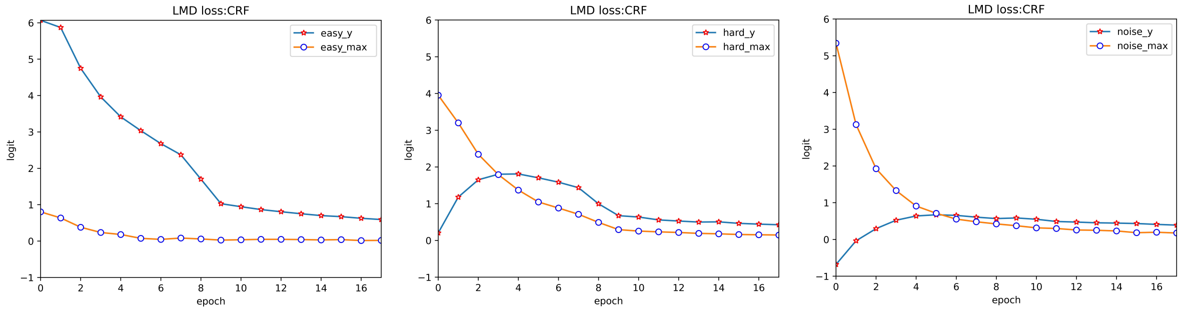

Figure 2: The graphs display logit value trajectories for easy sample (left), hard sample (middle) and noisy sample(right). The blue line: the logit value $Z_y^{(t)}$ corresponding to the label in logit matrix; the orange line: the maximal logit value $Z_{max}^{(t)}$ in logit matrix except $Z_y^{(t)}$. The x-axis refers to the number of training epochs, and the y-axis refers to the logit value. (Dataset: WUT-17)

samples are deleted according to the forgetting ratio (Jiang et al., 2018; Malach and Shalev-Shwartz, 2017). AUM (Pleiss et al., 2020) based on the output of the model to distinguish hard samples from noisy samples. CrossWeigh (Wang et al., 2019b) cover the label of a certain category in the datasets, and observes the model whether will predict the sample into another category. However, these methods always make the model to learn the easy samples and not consider the problem of overfitting (Chang et al., 2017).

Label correction methods are to improve the quality of raw labels. New labels equal to the probabilities estimated by the model (known as soft labels) or to one-hot vectors representing the model predictions (hard labels) (Tanaka et al., 2018; Yi and Wu, 2019). Another option is to set the new labels to equal a convex combination of the noisy labels and the soft or hard labels (Reed et al., 2015). However, these methods require the support from extra clean data or an expensive detection process to estimate the noise model.

3 Logit Maximum Difference Mechanism (LMD)

In this section, we propose a novel LMD mechanism. The LMD utilizes the tiny difference between hard samples and noisy samples in the model output. Meanwhile the LMD accumulates and expands the difference to identify noisy samples.

3.1 Preliminary

Easy samples and noisy samples are easy to distinguish(e.g., utilizing loss values (Han et al., 2018)), because of noisy samples are always contrary to the samples with correct tags. However, hard samples with ambiguous entity are difficult to distinguish from noisy samples, because hard samples also will produce large loss values (Song et al., 2020) in the early stage of training. This has become a major challenge in the denoising task (Song et al., 2020).

Utilizing Logit Matrix Neural network models will output a logit matrix in the training process, which goes through the softmax layer and then gets into loss function. The Softmax layer is a normalized exponential function, which will nonlinear increase the weight of maximum value in the logit matrix and bring unfairness for identifying noisy samples. LMD directly utilizes logit matrix to distinguish hard samples from noisy samples instead of loss values.

Given a sentence $x = [x_1, x_2, ..., x_n]$ and its tag sequence $y = [y_1, y_2, ...y_n]$, n is the sentence length. Every token x_i will obtain a corresponding logit matrix $Z = [z_1^i, z_2^i, ..., z_m^i]$, m denotes total number of tags. LMD utilizes the difference between the z_j corresponding to the class j and other values in the logit matrix.

Observing The Difference In Figure 2, the logit value $Z_y^{(t)}$ corresponding to tag y and epoch t, and the $Z_y^{(t)}$ is evidently higher than other values in easy samples(left). In hard samples(middle), the $Z_y^{(t)}$ is small at the beginning of training, then $Z_y^{(t)}$ begins to increase and become the maximum in the logit matrix with increasing epoch t. In noisy samples(right), the $Z_y^{(t)}$ is relatively smaller than other values, and the $Z_y^{(t)}$ becomes the maximum in the logit matrix when epoch exceeds 5 even if y is a negative tag, which indicates that overfitting occurs.

3.2 Identify Noisy Samples

Defining LMD We propose a new statistic LMD score, which averages the difference between $Z_y^{(t)}$ and the other values $Z_{other}^{(t)}$ at each epoch t. The tiny difference between hard samples and noisy samples are gradually accumulated and maximized, which will effectively help model identify noisy samples. The LMD score can be defined as:

$$LMD(x,y)=\frac{1}{T}\sum_{t=1}^{T}(min(Z_y^{(t)}-\max_{i\neq y} Z_i^{(t)})) \tag{1}$$

Where T is the total number of epoch. A sentence is the minimum unit of the input in the NER task. If the tag corresponding to single token is mislabeled in a sentence, we can consider that the sentence is negative. Therefore we choose the smallest LMD score in each sentence as the LMD score of the sentence, where every token will obtain a LMD score in the sentence.

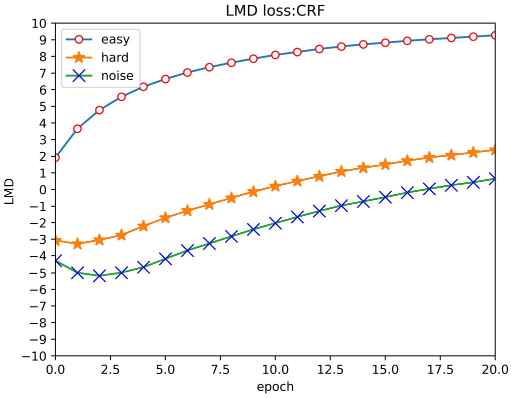

Figure 3: A record of LMD score in each epoch for easy, hard and noisy samples. We select 100 samples from the three sets respectively in WUT-17 dataset.

Working Mechanism In order to steadily enhance the discrimination between easy, hard and noisy samples, we stack the LMD scores of multiple epochs to get an average value. By utilizing the LMD mechanism, every sample will get a LMD score. The LMD scores of easy, hard and noisy samples have obvious differentiation in Figure 3. Then we sort the samples according to the LMD scores, and define samples under the noise ratio μ as noisy samples, finally delete them to get a cleaner training set. And the noise ratio μ is a hyperparameter. The model is trained again with utilizing a clean training set, which will obtain better performance without the interference of noisy samples.

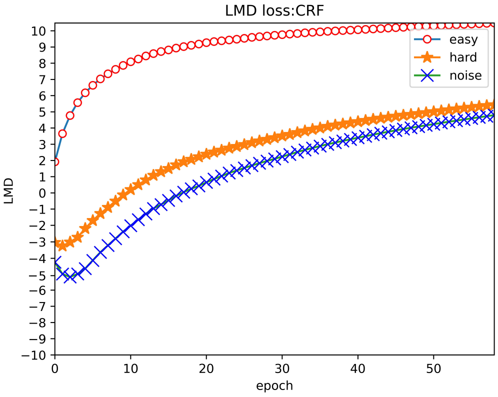

Figure 4: Compared with the experiment in Figure 3, we only adjusted the epoch to 60.

3.3 The Influence of Overfitting

Overfitting Appears We further explore the influence of overfitting in the denoising task from two sides. From the perspective of the LMD scores, the LMD scores of hard samples and noisy samples tend to be consistent with increasing epoch to 60 in the Figure 4. From the perspective of the logit values, the logit value $Z_y^{(t)}$ becomes the maximum in the logit matrix when epoch exceeds 5 even if y is a noise tag in the Figure 2.

Phenomenon Analysis As the increase of training iteration epochs, the model will overfit noisy samples. Meanwhile the model output of hard samples and noisy samples are almost consistent, which make it difficult to distinguish. This proposes another challenge that the model identifies noisy samples in the case of overfitting.

4 Incomplete-Trust Loss Function

In this section, we propose an Incomplete-Trust (In-trust) loss function. Previous loss functions (e.g., Cross Entropy) are easy to overfit noisy samples (Wang et al., 2019b), and they absolutely trust tags even if the tags are mislabeled. Meanwhile, neural networks have strong fitting ability, they can achieve zero training error even on datasets with randomly-assigned labels (Zhang et al., 2016). And deep neural networks have been observed to first fit the samples with clean tags during an “early learning” phase, before eventually memorizing the samples with mislabeled tags (Arpit et al., 2017; Zhang et al., 2016). With exploiting the early-learning phenomenon, our proposed In-trust utilizes both the model output which obtain the relative correctness after enough practices and the incomplete correctness of tags which maybe are mislabeled. We also provide theoretical analysis about the formulation and behavior of In-trust.

4.1 Definition

KL-divergence Given two distributions p and q, the relationship between entropy, cross entropy and KL-divergence is as follows:

$$KL(q||p)=H(q,p)-H(q) \tag{2}$$

In NER task, $q=q(k|x)$ is the one-hot distribution of the label in sample x, and $p=p(k|x)$ is the prediction distribution of the model for sample x. The model makes the $p=p(k|x)$ gradually approach the $q=q(k|x)$, this is also to minimize the KL-divergence between the two distributions.

Proposing DCE Term However, if the sample x is a noisy sample and the $q=q(k|x)$ is an incorrect distribution, it will cause negative impacts for model, so the label distribution $q=q(k|x)$ is not worthy of full trust. According to the phenomenon of early learning, the model always tends to learn the correct samples in the early stage of training. It means that even if some samples are mislabeled, the model still may predict the correct results in the early stage of training. We exploit this phenomenon to trust that not only the label distribution $q=q(k|x)$, but also the prediction distribution $p=p(k|x)$ before the model overfit noisy samples. Therefore, we design the robust Distrust-Cross-Entropy L_{DCE} term as follows:

$$L_{DCE}=-p\log(\delta p+(1-\delta)q) \tag{3}$$

Where δ is a hyperparameter, and its size determines that the model whether trust labels or model output. When δ is larger, the model will trust prediction distribution $p=p(k|x)$ more, on the contrary, the model will trust label distribution $q=q(k|x)$ more.

Forming In-trust We proposed an Incomplete-Trust (In-trust) loss function, which boosts L_{CRF} with L_{DCE} term.

$$L_{In-trust}=\alpha L_{CRF}+\beta L_{DCE} \tag{4}$$

Where L_{DCE} term is an acceleration regulator term, which can effectively prevent model from overfitting noisy samples. That will be proved in Appendix A. The L_{CRF} term is not noise tolerant (Ghosh et al., 2017), but which benefits the convergence of the model (Zhang and Sabuncu, 2018). α and β are two decoupled hyperparameters, α regulates the overfitting issue of L_{CRF} while β aims to flexibly explore the robustness of L_{DCE}.

Contrasting Logit Values Figure 5 shows the result of comparative experiment with Figure 2. The logit value $Z_y^{(t)}$ corresponding to mislabeled tag y is no longer the maximum in the logit matrix, this means that $L_{In-trust}$ effectively prevents model from overfitting noisy samples.

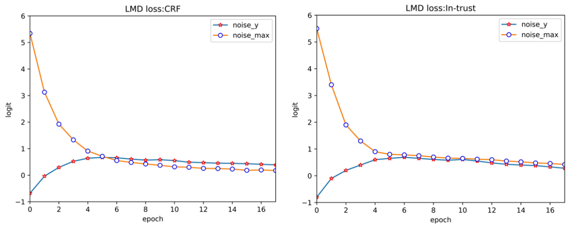

Figure 5: A comparative experiment of CRF (left) on noisy samples. We only replace the loss function to $L_{In-trust}$ (right), and other parameters are consistent.

Contrasting LMD Scores Figure 6 shows the result of comparative experiment with Figure 4. There is still obvious discrimination between hard samples and noisy samples when the epoch reaches 60, this indicates that $L_{In-trust}$ can help LMD mechanism identifies noisy samples accurately.

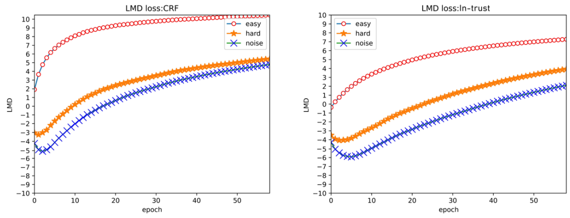

Figure 6: A comparative experiment between CRF (left) and $L_{In-trust}$ (right), other parameters are consistent.

4.2 Robustness Analysis

L_{DCE} Robustness Analysis: In order to simplify the calculation, we set α and β as 1 and derive the gradient of L_{DCE}. For brevity, we denote p_k, q_k as abbreviations for $p=p(k|x)$ and $q=q(k|x)$, the gradient of the L_{DCE} loss with respect to the logits Z_j can be derived as:

$$\frac{\partial L_{DCE}}{\partial Z_j}=(\sum_{k=1}^{K} p_k \times \log(\delta p_k+(1-\delta)q_k))^{'} \quad (5)$$

Where $\frac{\partial p_k}{\partial Z_j}$ can be further derived based on whether $k = j$:

$$\begin{cases} \dfrac{\partial p_k}{\partial Z_j} = p_k(1-p_k) & k=j \\ \dfrac{\partial p_k}{\partial Z_j} = -p_k p_j & k \neq j \end{cases} \quad (6)$$

For brevity, we denote $a_k=\delta p_k+(1-\delta)q_k$ and $b_k=p_k \log a_k+\frac{\delta p_k^2}{a_k}$. We know that $\frac{\partial p_k}{\partial Z_j}$ is a function of p_k from Eq.(5), let $L_{p_k}=\frac{\partial p_k}{\partial Z_j}$, and the gradient of the L_{p_k} with respect to the p_k can be derived as:

$$L_{p_k}^{'}=\sum_{k=1}^{K} b_k+(p_j-1)\frac{\partial b_j}{\partial p_j} \quad (7)$$

And the second derivative of L_{p_k} is:

$$L_{p_k}^{''}=2b^{'}+(p_j-1)b^{''} \quad (8)$$

Where $L_{p_k}^{''}$ is a monotone increasing function, when $q_j=1$ and $\delta \in [0.0, 0.1, ...]$, we obtain the corresponding relation between $L_{p_k}^{''}$ and p_j in the Figure 7. It is concluded that $L_{p_k}^{'}$ is a decreasing and then increasing function, which also shows that the acceleration of L_{DCE} first decreases and then increases with the increase of p_j corresponding to the label.

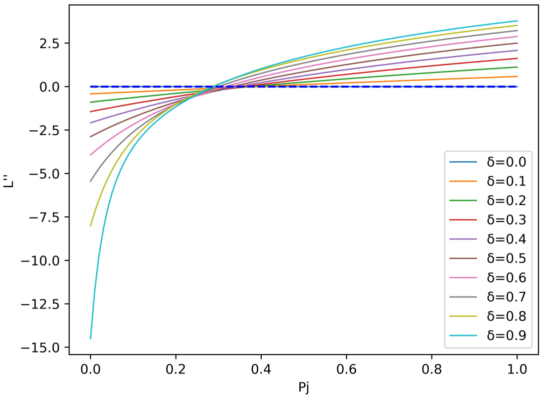

Figure 7: The acceleration record of LDC term.The LDC term produces different accelerations for different model outputs. The x-axis is p_j and the y-axis is the corresponding $L_{p_k}^{''}$ value. $\delta \in [0.0, 0.1, ...]$.

When p_j approaches to 1 with the q distribution is close to the p distribution, the model will believes correct tags more, and L_{DCE} has larger acceleration in learning correct samples. That benefits the model learns cleaner samples and prevents overfitting. On the contrary, the L_{DCE} term thinks that the model has relatively correct prediction result for noisy samples under the influence of learning other correct labels, and the acceleration is small which also effectively prevents the model overfitting and improves the noise tolerance. And other more detailed proofs are shown in the Appendix A.

$L_{In-trust}$ **Robustness Analysis:** According to Eq.(4), $L_{In-trust}$ consists of L_{CRF} and L_{DCE}. When $q_j = 1$, L_{DCE} term will provide a robust acceleration value, which benefits $L_{In-trust}$ obtains a correct loss value. Specifically, $L_{In-trust}$ will obtain a greater loss value when p_j approach to q, which will benefit the model learn this sample like L_{CRF}. On the contrary, L_{DCE} will prevent model from learning the sample unlike L_{CRF}. When $q_j = 0$, other loss functions will prevent the model from learning the direction, even if the model output p is greater in this direction. We believe that the model is relatively correct after learning a large number of samples. And L_{DCE} term will provide $L_{In-trust}$ with an acceleration to help the model to learn the direction. In addition, L_{DCE} term also prevent the model from learning when p is small. Therefore L_{DCE} term has no negative effect on the convergence of model.

5 Experiments

In this section, we verify the advantages of Noise-Aware-with-Filter (NAF) method by comparing experiments with other denoising methods.

5.1 Experimental Setup

NER Dataset

English NER Dataset We evaluate our method on English NER datasets include WUT-17 (Derczynski et al., 2017), CoNLL2003 (Tjong Kim Sang and De Meulder, 2003), CoNLL++ (Wang et al., 2019b) and OntoNotes5 (Pradhan et al., 2013). CoNLL2003 is in news domain and WUT-17 is user generated text. Compared with CoNLL2003, the test set of CoNLL++ is manually corrected. OntoNotes5 is a larger dataset and contains 18 entity types.

Dataset	Weibo NER	OntoNotes4	OntoNotes5	CoNLL2003	CoNLL++	WUT-17
CE	59.71	81.05	85.57	90.99	92.07	54.77
L_{CRF}	60.88	81.22	85.40	91.60	91.87	55.22
SCE	60.97	80.72	85.30	91.53	91.26	53.55
DSC	60.78	80.34	85.23	91.34	92.08	54.48
ELR	57.41	**81.54**	84.48	91.24	91.73	53.54
$L_{In-trust}$(our)	**61.97**	81.29	**86.23**	**91.67**	**92.68**	**56.13**
NAF(our)	**62.55**	**82.15**	**86.46**	**91.72**	**92.95**	**58.54**

Table 1: The result of NAF($L_{In-trust}$ + LMD) and other denoising methods in the **BERT** model.

Dataset	CoNLL2003	CoNLL++	WUT-17
CE	94.31	95.80	64.99
L_{CRF}	94.22	95.90	64.85
RCE	94.29	95.37	63.73
DSC	94.33	94.79	64.89
ELR	94.10	94.21	63.58
$L_{In-trust}$(our)	**94.45**	**96.13**	**66.44**
NAF(our)	**94.51**	**96.25**	**67.88**

Table 2: The result of NAF($L_{In-trust}$ + LMD) and other denoising methods in the **LUKE** model.

Chinese NER Dataset Chinese NER datasets include Weibo NER (Peng and Dredze, 2015) and OntoNotes4 (Pradhan et al., 2011). Weibo NER is in social domain, OntoNotes4 is in news domain.

In these six real-wold datasets, we use the same way of data segmentation as the original author. Since WUT-17 has no development set, we randomly select 10% samples from the training set as the development set.

Pre-trained Language Model **BERT** (Devlin et al., 2019) employs a Transformer encoder to learn a BiLM from large unlabeled text corpora and sub-word units to represent textual tokens. We use the $BERT_{base}$ model in our experiments. **LUKE** (Yamada et al., 2020)proposes new pretrained contextualized representations of words and entities based on the bidirectional transformer, which is the state-of-the-art(SOTA) model in English NER task.

Baseline We compare NAF with 3 recently proposed robust methods as well as the standard L_{CRF} : (1) CE:Cross Entropy; (2)SCE (Wang et al., 2019b): symmetric cross entropy loss; (3)DSC (Li et al., 2020): dice loss function; (4)ELR (Liu et al., 2020): early regularization; (5)In-trust (We proposed Incomplete-Trust loss function).

Evaluation Our primary evaluation metric is F1 score on the test set to compare the results of different methods.

5.2 Experimental Settings

In our experiments, we set the initial learning rate to $lr = 1e{-}5$ for all datasets. Since the scale of each dataset varies, we set different training batch size for different datasets. Specifically, we set the batch sizes of Weibo NER, OntoNotes4, WUT-17, CONLL2003 and CoNLL++ as 40, 40, 40,32 and 32 in **BERT**, and set the batch sizes of WUT-17, CONLL2003 and CoNLL++ as 2, 2 and 2 in **LUKE**. We stop the training when we find the best result in the development set.

5.3 Robustness Performance

Table 1 presents the results for the baseline and our methods in the **BERT**. Compared with other methods, NAF shows obvious advantage in the six real-wold datasets. Our method outperforms other methods by 1.58%, 0.61%, 0.29%, 0.87% and 3.77% in F1 score on Weibo NER, OntoNotes4, CoNLL2003, CoNLL++ and WUT-17 datasets. Table 2 presents the results in the **LUKE**, and our method has achieved new state-of-the-art (SOTA) with the F1 score reached to **94.51%** and **96.25%** on CoNLL2003 and CoNLL++ datasets.

Specifically, NAF has made more obvious progress on Weibo NER, WUT-17 and CoNLL++ datasets, and our analysis shows that the noise ratio of Weibo NER and WUT-17 are greater than others and there is a cleaner test set after manual correction in CoNLL++ dataset.

5.4 Manual Verification

Results Statistics In order to prove the effectiveness of our method, we manually verify the noisy samples which are selected from CoNLL2003, OntoNotes4 and WUT-17 (Table 4). We randomly select 100 samples from “Original” train set and we manually verify the proportion of noisy samples. After utilizing LMD or NAF method, we will obtain a new datasets, and then we randomly select

Dataset	**noisy samples**
CoNLL2003	**Soccer - KEANE signs four-years contract with Manchester United{LOC} .**
	Soccer - sharpshooter knup back in swiss{MISC} squad .
	Little{PER} change from today 's weather expected .
	9/16 - Luo Yigang (China) beat Hwang Sun-ho{MISC} (South Korea) 15-3 .
WUT-17	**Federal lawyers{O} fly to Minneapolis to investigate shooting**
	@janzensational at least may date ka na hahaha{O} . Goodluck zen ! : *

Table 3: noisy samples in CoNLL2003 and WNUI-17. The mislabel entities are marked with red.

Dataset	CoNLL2003	Weibo NER
$\delta = 0.1$	90.34%	60.58%
$\delta = 0.2$	90.52%	61.12%
$\delta = 0.3$	91.21%	61.32%
$\delta = 0.4$	90.43%	61.78%
$\delta = 0.5$	**91.72%**	61.44%
$\delta = 0.6$	91.47%	62.21%
$\delta = 0.7$	91.28%	**62.55%**
$\delta = 0.8$	91.32%	62.32%
$\delta = 0.9$	90.20%	61.70%

Table 5: The effect of δ in **In-trust**. We set α=1 and β=1 here.

100 samples from the new datasets for manually verifying. The probability of true negative samples is 5% in the "Original" CoNLL2003 dataset, and which reaches to 76% and 82% respectively with utilizing LMD and NAF methods. While for OntoNotes4 dataset, the probability is 8% in "Original" dataset and which reaches to 72% and 80% with LMD and NAF. In the WUT-17 dataset, the probability is 18% in "Original" dataset and which reaches to 58% and 71% with LMD and NAF.

Result Analysis The accuracy of identifying noisy samples will greater with the dataset that model performs better, and the excellent datasets will benefit the model identify noisy samples.

Dataset	Original	LMD(our)	NAF(our)
CoNLL2003	5%	**76%**	**82%**
OntoNotes4	8%	**72%**	**80%**
WUT-17	18%	**58%**	**71%**

Table 4: The accuracy of identifying noisy samples is verified manually. We select 100 samples from the Original train set, LMD and NAF separately, then verify the proportion of noisy samples manually. And the "Original" means raw data.

Demonstration of Examples The real noisy samples in CoNLL2003 and WUT-17 datasets are shown in Table 3, such as {Soccer - KEANE signs four-years contract with Manchester United}, the { **Manchester United**} is wrongly marked **{LOC}**. The { **Manchester United** } is a football club in Manchester England, and which should be marked **{ORG}**. In addition, the more noisy samples in datasets are shown in the Appendix C.

5.5 Ablation Experiment

As mentioned in the previous Section 4.1, δ provides flexibility between the model output distribution p=$p(k|x)$ and the label distribution q=$q(k|x)$. In this section, we explore the influence of hyperparameter δ, and we conducted experiments on Weibo NER and CoNLL2003 with α=1 and β=1 to explore how it manipulates the tradeoff. Experimental results are shown in Table 5. The highest F1 on CoNLL2003 datatset is 91.72% when δ is set to 0.5, meanwhile for Weibo NER, the highest F1 is 62.55% when δ is set to 0.7. The optimal value of δ is different in different noise ratio datasets, and when there are more noisy samples in the datasets, δ should be set larger. Because of the noise ratio of Weibo NER is larger than CoNLL2003 dataset, the optimal δ value of Weibo NER is larger. The experiment result of the other hyperparameters α and β are show in the Appendix B.

6 Conclusion

In this paper, we observe that the existing denoising methods can not effectively distinguish hard samples from noisy samples, and we proposed a new method called Noise-Aware-with-Filter (NAF), which contains LMD mechanism and In-trust loss function to solve the issues. Specifically, NAF can effectively improve the discrimination between hard samples and noisy samples even in the case of overfitting. In addition, our proposed the Logit-Maximum-Difference(LMD) mechanism which maximizes the diversity between different samples to help the model identify noisy samples. Meanwhile we design an Incomplete-Trust ($L_{In-trust}$) loss function, which boosts L_{CRF} with a noise robust Distrust-Cross-Entropy(DCE) term. In order to verify the effectiveness of our method, we also conduct manual verification for noisy samples and the results show that our method has higher accuracy on identifying noisy samples. Experiments on six real-world Chinese and English NER datasets show that NAF outperforms the previous methods, and which obtained the state-of-the-art(SOTA) results on the CoNLL2003 and CoNLL++ datasets.

7 Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.2020AAA0106400), the National Natural Science Foundation of China (No.61976211, No.61806201). This work is also supported by a grant from Huawei Technologies Co., Ltd.

References

Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: training more accurate neural networks by emphasizing high variance samples. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1003–1013.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Michael Flor, Michael Fried, and Alla Rozovskaya. 2019. A benchmark corpus of english misspellings and a minimally-supervised model for spelling correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 76–86.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1919–1925.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W Tsang, and Masashi Sugiyama. 2018. Co-teaching: robust training of deep neural networks with extremely noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8536–8546.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR.

Lukas Lange, Michael A Hedderich, and Dietrich Klakow. 2019. Feature-dependent confusion matrices for low-resource ner labeling with noisy labels. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3545–3550.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice loss for data-imbalanced nlp tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. 2020. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33.

Eran Malach and Shai Shalev-Shwartz. 2017. Decoupling" when to update" from" how to update". In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 961–971.

David Nadeau, Peter D Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Conference of the Canadian society for computational studies of intelligence*, pages 266–277. Springer.

Farhad Nooralahzadeh, Jan Tore Lønning, and Lilja Øvrelid. 2019. Reinforcement-based denoising of distantly supervised ner with partial annotation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 225–233.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Geoff Pleiss, Tianyi Zhang, Ethan R Elenberg, and Kilian Q Weinberger. 2020. Identifying mislabeled data using the area under the margin ranking. *arXiv preprint arXiv:2001.10528*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Sameer Pradhan, Lance Ramshaw, Mitch Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27.

Scott E Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training deep neural networks on noisy labels with bootstrapping. In *ICLR (Workshop)*.

Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. 2020. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2018. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560.

EF Tjong Kim Sang and F De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition.

Xinshao Wang, Yang Hua, Elyor Kodirov, and Neil M Robertson. 2019a. Imae for noise-robust learning: Mean absolute error does not treat examples equally and gradient magnitude's variance matters. *arXiv preprint arXiv:1903.12141*.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019b. Crossweigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5157–5166.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised ner with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2159–2169.

Kun Yi and Jianxin Wu. 2019. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7017–7025.

Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. 2019. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173. PMLR.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8792–8802.

A $L_{In-trust}$ Robustness Proof

We have know $L_{In-trust}$ is:

$$\begin{aligned} L_{In-trust} &= \alpha \times CRF + \beta \times L_{DCE} \\ &= \alpha \times CRF + \beta \times (-p \times \log(\delta p + (1-\delta) q)) \end{aligned} \tag{9}$$

And make:$a = \delta p + (1-\delta) q$

$$\begin{aligned} Eq9 &= \alpha \times CRF + \beta \times (-p \times \log a \\ &= \alpha \times CRF \\ &+ \beta \sum_{k=1}^{K} (-\frac{\partial p_k}{\partial Z_j} \log a_k + (-p_k \frac{a_k^{‘}}{a_k})) \end{aligned} \tag{10}$$

Meanwhile we have know:

$$\begin{aligned} \frac{\partial p_k}{\partial Z_j} &= \frac{\partial (\frac{Z_k}{\sum_{j=1}^{K} e^{Z_j}})}{\partial Z_j} \\ &= \frac{\frac{\partial e^{Z_k}}{\partial Z_j} (\sum_{j=1}^{K}) - e^{Z_k} \frac{\partial (\sum_{j=1}^{K} e^{Z_j})}{\partial Z_j}}{(\sum_{j=1}^{K} e^{Z_j})^2} \end{aligned} \tag{11}$$

When $k = j$:

$$\begin{aligned} \frac{\partial p_k}{\partial Z_j} &= \frac{\partial p_k}{\partial Z_k} \\ &= \frac{e^{Z_k} (\sum_{k=1}^{K} e^{Z_j}) - (e^{Z^k})^2}{(\sum_{k=1}^{K} e^{Z_k})^2} \\ &= \frac{e^{Z_k}}{\sum_{k=1}^{k} e^{Z_k}} - (\frac{e^{Z_k}}{\sum_{k=1}^{K} e^{Z_k}})^2 \\ &= p_k - {p_k}^2 \\ &= p_k (1 - p_k) \end{aligned} \tag{12}$$

When $k \neq j$:

$$\begin{aligned} \frac{\partial p_k}{\partial Z_j} &= \frac{0 (\sum_{k=1}^{K} e^{Z_j}) - e^{Z_k} e^{Z_j}}{(\sum_{k=1}^{K} e^{Z_j})(\sum_{k=1}^{K} e^{Z_j})} \\ &= - \frac{e^{Z_k}}{\sum_{k=1}^{K} e^{Z_j}} \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_j}} \\ &= - p_k p_j \end{aligned} \tag{13}$$

Gradient Calculation:

$$\begin{aligned} L_{DCE} &= - p \times \log(\delta p + (1-\delta) q) \\ \frac{\partial L_{DCE}}{\partial Z_j} &= (\sum_{k=1}^{K} p_k \times \log(\delta p_k + (1-\delta) q_k))' \\ &= \sum_{k=1}^{K} {p_k}' \log(\delta p_k + (1-\delta) q_k) \\ &+ \frac{\delta p_k {p_k}'}{\delta p_k + (1-\delta) q_k} \end{aligned} \tag{14}$$

For the convenience of calculation, we make $a = \delta p_k + (1-\delta) q_k$

$$\begin{aligned} \frac{\partial L_{DCE}}{\partial Z_j} &= \sum_{k=1}^{K} p_k^{‘} \log a_k + \frac{p_k a_k^{'}}{a_k} \\ &= \sum_{k=1}^{K} p_k^{'} \log a_k + \frac{\delta p_k p_k^{‘}}{a_k} \\ &= - p_j \sum_{k=1}^{k} (p_k \log p_k + \frac{\delta p_k^2}{a_k}) \\ &+ p_j \log a_j + \frac{\delta p_j^2}{\delta a_j} \end{aligned} \tag{15}$$

Make $b_k = p_k \log a_k + \frac{\delta p_k^2}{a_k}$

$$\begin{aligned} \frac{\partial L_{DCE}}{\partial Z_j} &= - p_j \sum_{k=1}^{K} b_k + b_j \\ \frac{\partial L_{DCE}}{\partial Z_j} &= p_j \sum_{k=1}^{K} b_k - b_j \end{aligned} \tag{16}$$

We hypothesis $L = p_j \sum_{k=1}^{K} b_k - b_j$

$$\begin{aligned} L^{'} = \frac{\partial L}{\partial p_j} &= \sum_{k=1}^{K} b_k + p_j (\sum_{k=1}^{K} b_k)^{‘} - \frac{\partial b_j}{\partial p_j} \\ &= \sum_{k=1}^{K} b_k + (p_j - 1) \frac{\partial b_j}{\partial p_j} \end{aligned} \tag{17}$$

And $L^{''} = 2b^{'} + (p_j - 1) b^{''}$

Meanwhile we can get:

$$\begin{aligned} b^{'} &= \log a + \frac{pa^{'}}{a} + \frac{2\delta p - \delta p^2 a^{'}}{a^2} \\ &= \log a + \frac{\delta p}{a} + \frac{2\delta p - \delta^2 p^2}{a^2} \\ &= \log a + \frac{3\delta p}{a} - (\frac{\delta p}{a})^2 \end{aligned} \tag{18}$$

$$\begin{aligned} b^{''} &= \frac{\delta}{a} + 3\delta \frac{(1-\delta) q}{a^2} - 2\delta^2 \frac{p}{a} \frac{(1-\delta) q}{a^2} \\ &= \frac{\delta}{a} + \frac{3\delta (1-\delta) q}{a_2} - \frac{2\delta^2 p (1 - \delta) q}{a^3} \end{aligned} \tag{19}$$

When $q_j = 0$:

$$\begin{aligned} L^{''} &= 2 \log(\delta + 2) + \frac{p-1}{p} \\ &= 2 \log \delta p + 4 + 1 - \frac{1}{p} \\ &= 2 \log \delta p + 5 - \frac{1}{p} \end{aligned} \tag{20}$$

When p approaches 0 : $L^{''}<0$

When p approaches 1 : $L^{''}=2\log\delta+4$

$E:\delta 1\in[0,1]$, make $L^{''}(\delta 1)>0$

Because $L^{''}$ is continuous function, so $L^{''}$ is Monotone increasing function

$E:\delta 2\in[0,1]$, make $L^{''}(\delta 2)=0$

So $L^{'}$ is Decreasing then increasing function and the inflection point is only related to δ.

When $q_j=1$:

$$b^{'}=\log a+\frac{3\delta p}{a}-(\frac{\delta p}{a})^2$$
$$b^{''}=\frac{\delta}{a}+\frac{3\delta(1-\delta)q}{a_2}-\frac{2\delta^2 p(1-\delta)q}{a^3}$$
$$=\frac{\delta a^2+3\delta a-3\delta^2 a-2\delta^2 p+2\delta^3 p}{a^3}$$

$$\begin{aligned}L^{''}&=2b^{'}+(p_j-1)b^{''}\\&=2\log a+\frac{6\delta p}{a}-\frac{2\delta^2p^2}{a^2}+\\&\frac{\delta a^2p+3\delta ap-3\delta^2ap-2\delta^2p^2+2\delta^3p^2}{a^3}+\\&\frac{-\delta a^2-3\delta a+a\delta^2 a+2\delta^2p-2\delta^3p}{a^3}\end{aligned}\tag{21}$$

Because $L^{''}$ is monotone increasing function

So when $p=0$:

$L^{''}=2\log(1-\delta)+\frac{-\delta a^2-3\delta a+3\delta^2 a}{(1-\delta)^3}<0$

When $p=1$:

In order to simplify the calculation, we make $\delta=0$

So $L^{''}=0$

When $\delta>0$, and $L^{''}>0$

So $L^{'}$ is Decreasing then increasing function and the inflection point is only related to δ.

In summary, when $q=0$ or $q=1$:

$L^{'}$ is Decreasing then increasing function and the inflection point is only related to δ.

We know $L^{'}(p=0)>0$, $L^{'}(p=1)>0$,

and $E:\mu\in[0,1]$, make $L^{'}(p=\mu)<0$

We observe that when δ is larger, the model tends to learn from the p of the model output, and when δ is smaller, the model tends to learn from the label q. Moreover, when the p_j corresponding to the label is larger and the model output is close to the label distribution, the acceleration of L_{DC} term is larger, which makes the model more inclined to learn the sample, which helps the model learn clean samples. When p_j is small, there is a big gap between the model output and label distribution. We think that the sample may be a noisy sample, and the acceleration of L_{DC} term is smaller, which makes the model more inclined to give up the learning of the sample, and prevents the model from over fitting the noisy sample.

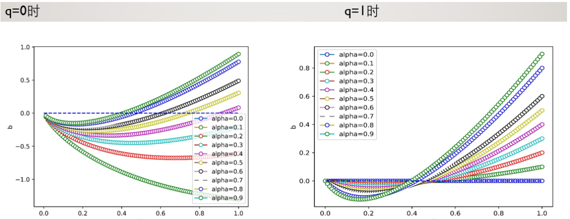

Figure 8: The relationship between b and p.

B Ablation Experiment Supplement

Dataset	CoNLL2003	OntoNotes4
$\alpha=0.1$	27.82%	29.56%
$\alpha=0.2$	30.37%	34.17%
$\alpha=0.3$	32.41%	46.71%
$\alpha=0.4$	45.32%	54.31%
$\alpha=0.5$	56.23%	68.32%
$\alpha=0.6$	74.10%	75.43%
$\alpha=0.7$	89.37%	78.32%
$\alpha=0.8$	90.21%	80.32%
$\alpha=0.9$	91.70%	82.13%

Table 6: Appendix B: The effect of α in **In-trust**. We set δ=0.5 and β=1 here.

Dataset	CoNLL2003	OntoNotes4
$\beta=0.1$	90.99%	81.05%
$\beta=0.2$	91.15%	80.07%
$\beta=0.3$	91.33%	80.07%
$\beta=0.4$	91.32%	80.09%
$\beta=0.5$	91.67%	81.10%
$\beta=0.6$	91.65%	82.09%
$\beta=0.7$	91.68%	82.15%
$\beta=0.8$	91.72%	82.10%
$\beta=0.9$	91.70%	82.13%

Table 7: Appendix B: The effect of β in **In-trust**. We set α=0.6 and δ=0.5 here.

C noisy samples

dataset	noisy samples
CoNLL2003	**TYPE FRN BASE 3M LIBOR PAY DATE S23.SEP.96** **O O O B-ORG O O O O**
	English County Championship cricket matches on Thursday : **B-MISC B-MISC I-MISC O O O O O**
	SOCCER - EUROPEAN CUP WINNERS ' CUP RESULTS . **O O B-MISC I-MISC I-MISC I-MISC I-MISC O O**
	Red Star - Vinko Marinovic (59th) **B-ORG I-ORG O B-MISC I-MISC O O O**
	SOCCER - SHARPSHOOTER KNUP BACK IN SWISS SQUAD . **O O O B-PER O O B-MISC O O**

Table 8: Appendix C: noisy sample display

A Multi-Task Approach for Improving Biomedical Named Entity Recognition by Incorporating Multi-Granularity Information

Yiqi Tong, Yidong Chen*, **Xiaodong Shi**
School of Informatics, Xiamen University, Xiamen, China
`yqtong@stu.xmu.edu.cn`
`{ydchen, mandel}@xmu.edu.cn`

Abstract

Neural biomedical named entity recognition (BioNER) methods usually require a large amount of annotated data, while the annotated BioNER datasets are often difficult to obtain and small in scale due to the limitations of privacy, ethics and high degree of specialization. To alleviate the lack of training samples, unlike conventional methods that only use token-level information, this paper proposes a method that simultaneously utilize the latent multi-granularity information in the dataset. Concretely, the proposed model is based on a multi-task approach, which leverages different training objectives by introducing auxiliary tasks, i.e. binary classification, multi-class and multi-token classification. Experimental results over three BioNER datasets show that the proposed model produces better performance over the BioBERT baseline and can get more than 3% improvements of F1-score in low-resource scenarios. Finally, we released our code at `https://github.com/zgzjdx/MT-BioNER`.

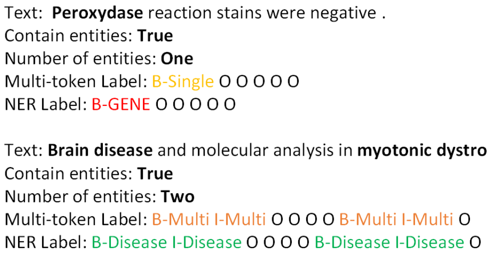

Text: **Peroxydase** reaction stains were negative .
Contain entities: **True**
Number of entities: **One**
Multi-token Label: B-Single O O O O O
NER Label: B-GENE O O O O O

Text: **Brain disease** and molecular analysis in **myotonic dystrophy** .
Contain entities: **True**
Number of entities: **Two**
Multi-token Label: B-Multi I-Multi O O O O B-Multi I-Multi O
NER Label: B-Disease I-Disease O O O O B-Disease I-Disease O

Figure 1: Examples from our constructed dataset. In our work, we designed three auxiliary tasks to help improving the main NER task. Two of them are sentence-level tasks and the other one is a token-level task. Concretely, the first sentence-level task predicts whether or not a sentence contains entities; the second sentence-level task predicts how many entities a sentence contains; and the token-level task predicts whether or not a given token belongs to a multi-token entity. Clearly, to support training the auxiliary tasks, additional labels have been added in our data. However, please note that, the additional labels could be derived from the original NER labels and do not need additional manual annotations. In a word, what we have done in this paper is try to use the multi-granularity information implied in the original dataset to improve the performance of BioNER.

1 Introduction

Biomedical named entity recognition (BioNER) aims to identify entity mentions such as gene/protein, disease and chemicals from unstructured text. Such information is useful for downstream natural language processing (NLP) tasks like relation extraction (Zhou et al., 2014), automatic abstracting (Mishra et al., 2014) and question answering (Athenikos and Han, 2010), etc. Different from those named entity recognition (NER) tasks for general domain like news, BioNER is particular challenge due to the naming complexity (Liu et al., 2015), large variations in same entity names (Jia et al., 2019; Kim et al., 2019), and new entity mentions rapidly reported in scientific publications (Luo et al., 2018). These various factors lead to the small number and size of current BioNER datasets. In recent years, neural BioNER has become a main approach because of its outstanding performance (Lample et al., 2016; Habibi et al., 2017; Yadav and Bethard, 2019). Some researchers have investigated introducing multi-task learning (Crichton et al., 2017; Khan et al., 2020) and pre-training (Peng et al., 2019; Lee et al., 2020) to solve the problem of lacking extensive training data and boost the performance of BioNER model. However, few of them combined these two methods together and tried to transfer sentence-level knowledge to tokens (Rei and Søgaard, 2019; Kruengkrai et al., 2020), which had proven to be effective in

---

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4804–4813
August 1–6, 2021.

other domains (Abhishek et al., 2017).

In this paper, we focus on improving BioNER by exploiting multi-granularity information implied in the dataset, without depending on additional manually annotated data. As shown in Figure 1, Besides the main sequence labeling task, we employ three related classification tasks, i.e. a binary classification task for predicting whether a sentence contains entities or not, a multi-class classification task for predicting how many entities a sentence contains and a multi-token entity classification task (Hu et al., 2020). In the rest of this paper, these three tasks will be named bCLS, mCLS and mtCLS, respectively, while the main task will be named NER. Our primary motivation is to mine useful training signals from coarse-grained classification to guide a more robust and interpretable token-level representation.

Our key contributions can be summarized as follows:

- To take full advantage of the implicit information contained in NER dataset, we present a multi-task model for jointly learning sentence-level and token-level labels, which incorporates BioBERT (Lee et al., 2020) as text encoding layers and shares the hidden states between different tasks. To the best of our knowledge, we are the first to introduce different grained-level information in BioNER domain.
- Experimental results on three datasets show that our proposed method is effective, especially in the low-resource scenarios.
- We performed preliminary pair-wise comparison analysis to investigate the relations between tasks and pointed out that token-level labels are more helpful for sentence-level tasks. While at the same granularity, high-difficulty tasks are more helpful to low-difficulty tasks.

2 Related work

Traditional BioNER methods could be divided into rule- or dictionary-based approaches (Tjong Kim Sang and De Meulder, 2003; Kulick et al., 2004; Gerner et al., 2010). And recent works had shown neural network architecture based BioNER methods achieved promising results. Habibi et al. (2017) used a LSTM-CRF model, which was completely agnostic to entity types. Crichton et al. (2017), on the contrary, used a CNN-based model that takes tokens and their surrounding tokens as input. To solve the label inconsistent problem, Luo et al. (2018) proposed a Att-BiLSTM-CRF model and achieved better performance with little feature engineering.

The neural BioNER system is known to be extremely data-intensive, while the available training datasets are relatively small in scale. To tackle this problem, research has been conducted and language models and multi-task learning have been shown to be effective to deal with this problem (Peters et al., 2018; Liu et al., 2019). Jia et al. (2019) proposed a cross-domain NER model, which extracted knowledge from raw texts through a novel parameter sharing network. Yoon et al. (2019) proposed CollaboNet, which consists of multiple BiLSTM-CRF models where models could send information to one another for more accurate predictions, and got best F1-score at that time. Although these studies have exploited additional token-level information from auxiliary tasks or language models, they do not consider information from other levels that contained in the NER dataset.

More recently, a transformer-based (Vaswani et al., 2017) large-scale pre-training language model, called BERT (Devlin et al., 2018), led to impressive gains on several NLP benchmarks and the domain-specific BERTs, such as blueBERT (Peng et al., 2019), BioBERT (Lee et al., 2020), SciBERT (Beltagy et al., 2019) and PubmedBERT (Gu et al., 2020), have largely outperformed the previous state-of-the-art BioNER systems. But research on multi-task learning based on BERT is still few, and the association between tasks needed to be further explored (Khan et al., 2020; Vu et al., 2020).

The most similar work to ours is the findings of Kruengkrai et al. (2020). However, they only focused on introducing one auxiliary task that requires additional manual annotations, while we attempted to try multiple auxiliary tasks, and our proposed method did not rely on other additional annotations, except for BioNER.

3 The proposed model

3.1 Tasks

As mentioned in Section 1, our model involves four tasks: bCLS, mCLS, mtCLS and NER. The goal is to optimize the token-level representation of BioBERT by introducing auxiliary tasks (bCLS, mCLS, mtCLS) and improve the perfor-

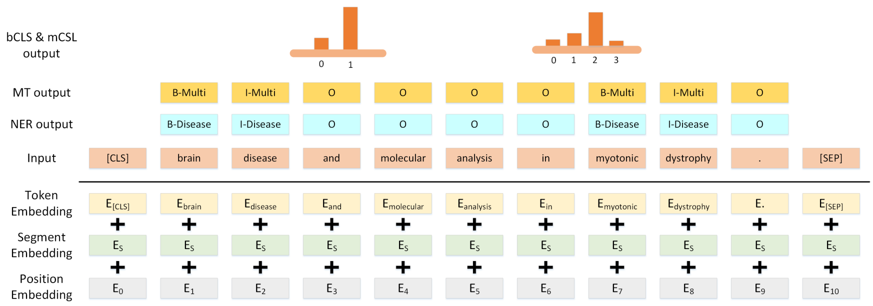

Figure 2: The input and output descriptions of the proposed model. Actually, our model involve three input embeddings and four outputs, and we adopt two special token [CLS] and [SEP] to represent the beginning and the end of the input sentence, respectively.

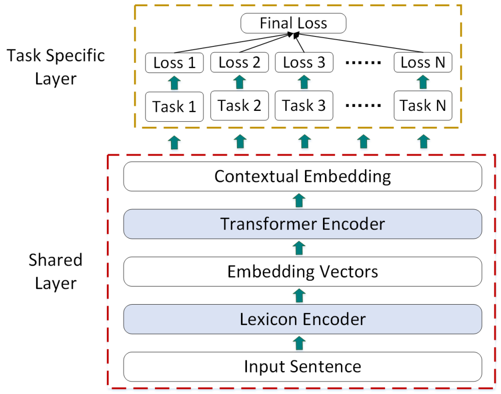

Figure 3: The architecture of our multi-task BioBERT-based model that can jointly learn sentence-level and token-level labels.

mance of the main task (NER). The pre-training model BioBERT are shared across all tasks by hard parameter sharing (Ruder, 2017). The input sequence and output labels of our proposed model are represented in Figure 2. Given a sentence $X = \{x_1, ..., x_i, ..., x_n\}$, where x_i is a token, n is the length of the input sequence. The first token of each X is always a special classification embedding [CLS] and the transformer encoder module maps X into a sequence of input embedding vectors, which are the sum of the token, segment and position embeddings. The detailed description of each task is shown as follows. **bCLS:** This is a sentence-level binary classification task. Given X, the goal is to predict whether it contains entities or not. In some cases, for X that do not contain entities, the model may incorrectly predicts that it contains entities. Or for X that contains entities, the model may incorrectly predicts that it not contain entities. Therefore, we design bCLS task with the hope of solving this problem by introducing a global guidance information.

mCLS: This is a sentence-level multi-classification task. Given X, the goal is to predict how many entities it contains. To balance label numbers, this paper set mCLS as a 4-classification task, which X contains 0, 1 and 2 entities is set to 0, 1 and 2, respectively, while X with more than 2 entities are all set to 3. Compared with bCLS, mCLS is more difficult and we introduce this task to alleviate the under- or over-recognition entity problem.

mtCLS: Multi-token classification is a token-level 3-classification problem. Given x_i in X, the goal is to predict whether it belongs to a multi-token entity like "brain disease" or a single-token entity like "peroxydase" or neither. Our motivation for introducing this task is that if the model knows whether x_i is multi-token entity or single-token entity or neither, it can alleviate the entity boundary problem.

NER: Given X, NER aims to predict corresponding labels $Y = \{y_1, ..., y_i, ..., y_n\}$, where y_i is predefined and differs according to the annotation scheme such as BIO and BIOES. We use this main task to measure the model performance and effectiveness of auxiliary tasks.

3.2 Architecture

The overall architecture of our proposed model is shown in Figure 3, which mainly includes two parts: the shared encoder and task-specific layers. We use multi-task learning to jointly train the main task and auxiliary tasks, which has been shown effective for transferring knowledge among multiple tasks (Yoon et al., 2019). For the shared encoder, we take cased BioBERT-base v1.1[1] as feature extractor and hard shared its parameters. Set X as an input sequence, where x_i denotes the $i-th$ token

---

[1] https://github.com/naver/biobert-pretrained

in X. We represent each x_i using the pre-trained BioBERT embeddings $h_i \in \mathbb{R}^d$, where d is the dimension of hidden states. And the task-specific layers have independent parameters, which include a project layer and a classifier for generating outputs. We use the output of the shared encoder, i.e. $H = \{h_1, ..., h_i, ..., h_n\}$, as the input of task-specific layers for both sentence- and token-level tasks, as described in detailed as follows.

Sentence-level tasks. As mentioned in subsection 3.1, bCLS and mCLS are two sentence-level classification tasks. Different from the standard BERT-based classification models, which optimize the [CLS] token (Sun et al., 2019) to perform classification. Our model aims at optimizing the token representations of the shared encoder by sentence-level labels. Therefore, we created a fixed size vector by applying mean/max pooling (Reimers and Gurevych, 2019) over H, which encourages the model to capture the most useful local features encoded in hidden states. Finally, the probability of class k is predicted by a linear layer and a logistic regression with softmax.

$$P(m|X) = softmax(Wh + b) \qquad (1)$$

where $h \in \mathbb{R}^d$ is the pooling output of model, $W \in \mathbb{R}^{d \times m}$ and $b \in \mathbb{R}^m$ are trainable weight matrix and bias. m denotes the number of category labels, which is 2 for bCLS and 4 for mCLS. Finally, the loss for our sentence-level task is calculated as follows:

$$L_S = -\sum_{m} \sigma(y_m = \hat{y}) log(P(m|X)) \qquad (2)$$

where $\sigma(y_m = \hat{y}) = 1$ if the classification $\hat{y}$ of X is the right ground-truth label for the class m. Otherwise, $\sigma(y_m = \hat{y}) = 0$.

Token-level tasks. As mentioned in subsection 3.1, mtCLS and NER are two token-level classification tasks[2]. Given the dataset D, which consists of N training samples, i.e. $D = (x_j, y_j)_{j=1}^{N}$, where j denotes the sentence index in D. To train the token-level tasks, we minimize the negative log-likelihood of the correct label sequences over D with the loss function defined as follows:

$$L_T = -\frac{1}{N} \sum_{j=1}^{N} log(P(y_j)|H_j) \qquad (3)$$

Algorithm 1 Training a MT-BioBERT model

```
Initialize: Model parameter of shared layers $\theta_i^{BioBERT}$ by BioBERT and task-specific layer
    $\theta_i^{task}$ randomly. Max epochs, max sequence length, learning rate, etc.
Input: Dataset $D$
shuffle $D$
for each epoch in $epoch_{max}$ do
    for each $b_t$ in D do
        # $b_t$ is a mini-batch of $D$
        Compute Loss: $L(\theta) = \alpha L(\theta)_{ner} + \beta L(\theta)_{mtCLS} + \gamma L(\theta)_{bCLS} + \delta L(\theta)_{mCLS}$
        $L(\theta)_{bCLS}$ = Eq.2 for binary classification
        $L(\theta)_{mCLS}$ = Eq.2 for multi-class classification
        $L(\theta)_{ner}$ = Eq.3 for sequence labeling
        $L(\theta)_{mtCLS}$ = Eq.3 for multi-token classification
    Compute gradient $\Delta_\theta$
    Update model $\theta = \theta - \epsilon\Delta_\theta$
    end
end
```

Dataset	Sentences	Entity Type and Counts
BC2GM	20,131	Gene/Protein (24,583)
BC5CDR	13,938	Chemical(15,935), Disease(12,852)
NCBI	7,287	Disease(6,881), Gene/Protein(35,336)

Table 1: Dataset description. We use BC2GM (Smith et al., 2008), BC5CDR (Li et al., 2016) and NCBI (Doğan et al., 2014) to conduct our experiments.

where $H_j \in \mathbb{R}^{n \times d}$ is the hidden state of x_j.

Algorithm 1 provides the procedure for our cross-task joint training method, where α, β, γ, δ are hyper-parameters. Moreover, the final loss of the proposed model is calculated by weighted summing the losses of different tasks.

4 Experiments

4.1 Datasets

We evaluated the performance of proposed approach on three benchmark datasets[3] used by Wang et al. (2019b); Yoon et al. (2019); Lee et al. (2020); Khan et al. (2020). Table 1 gives the statics of these datasets. Following previous works, we merged the

---

[2] Generally, NER was treated as a sequence labeling problem. However, for a fair comparison with previous works, instead of using sequence labeling algorithms such as Conditional Random Field (CRF) (Wallach, 2004) in task-specific layers, we still use softmax for token-level tasks.

[3] https://github.com/cambridgeltl/MTL-Bioinformatics-2016

Model	BC2GM			BC5CDR-chem			NCBI-disease		
	P	R	F1	P	R	F1	P	R	F1
Baseline systems									
Habibi et al. (2017)	81.57	79.48	80.51	87.60	86.25	86.92	86.11	85.49	85.80
Sachan et al. (2018)	81.81	81.57	81.69	88.10	90.49	89.28	86.41	88.31	87.34
Devlin et al. (2018)	81.17	82.42	81.79	90.94	91.38	91.16	84.12	87.19	85.63
Wang et al. (2019b)	82.10	79.42	80.74	93.09	89.56	91.29	85.86	86.42	86.14
Yoon et al. (2019)	80.49	78.99	79.73	94.26	92.38	93.31	85.48	87.27	86.36
Khan et al. (2020)	82.10	84.04	83.01	88.46	90.52	89.48	86.73	89.70	88.19
Beltagy et al. (2019)	-	-	83.36	-	-	92.51	-	-	88.25
Gu et al. (2020)	-	-	83.82	-	-	92.85	-	-	89.13
SOTA (Lee et al., 2020)	85.16	83.65	84.40	93.27	93.61	93.44	89.04	89.69	89.36
SOTA*	83.57	85.22	84.38	92.98	94.24	93.60	87.83	90.21	89.00
Our methods									
Ours_CLS	83.21	85.00	84.09	92.95	94.28	93.61	88.29	90.31	89.29
Ours_MEAN	84.31	83.78	84.04	94.08	93.24	93.66	87.75	91.77	89.71
Ours_MAX	84.42	85.14	**84.78**	93.29	94.69	**93.98**	88.90	90.94	**89.91**

Table 2: Model performance comparison on the three benchmark datasets, where SOTA* is our reproduce results of BioBERT, Ours_CLS uses the [CLS] token for sentence-level tasks and Ours_MEAN or Ours_MAX adopts the mean or max pooling strategy for sentence-level tasks, respectively.

Dataset	Metric	FULL-SIZE			50%-SIZE			25%-SIZE			10%-SIZE		
		CS-MTM	SOTA	Ours	CS-MTM	SOTA	Ours	CS-MTM	SOTA	Ours	CS-MTM	SOTA	Ours
BC2GM	P	83.21	85.16	84.42	79.37	82.21	81.78	79.44	80.15	80.82	72.95	75.27	76.60
	R	85.74	83.65	85.14	85.05	83.73	84.52	78.98	81.60	82.56	75.39	79.27	79.32
	F1	84.41	84.40	**84.78**	82.12	82.96	**83.13**	79.21	80.87	**81.68**	74.15	77.22	**77.93**
BC5CDR-chem	P	-	93.27	93.29	-	91.97	92.00	-	89.99	91.38	-	89.78	90.29
	R	-	93.61	94.69	-	92.37	93.96	-	92.48	93.30	-	90.90	91.33
	F1	-	93.44	**93.98**	-	92.17	**92.97**	-	91.22	**92.33**	-	90.34	**90.81**
NCBI-disease	P	86.59	89.04	88.90	84.72	85.77	92.50	81.00	81.22	83.96	79.32	79.69	83.65
	R	86.42	89.69	90.94	84.76	91.67	86.04	81.00	88.33	90.52	74.40	80.52	83.13
	F1	86.50	89.36	**89.91**	84.74	88.62	**89.16**	81.00	85.79	**87.12**	76.68	80.10	**83.39**

Table 3: Impacts of the dataset size. We keep the test set unchanged and only cut the training set.

training and developing sets for the model training. As a part of the data preprocessing step, token labels were encoded using the standard BIO scheme (Reimers and Gurevych, 2017). In this scheme, for example, a token describing a disease entity is tagged with ”B-Disease” if it is at the beginning of the entity, and ”I-Disease” if it is inside the disease entity. Other tokens that not describing entities of interest are tagged as ”O”.

4.2 Settings

Following the work of Peng et al. (2020), all datasets are trained with the batch size of 32, maximum sequence length of 256 and a dropout (Srivastava et al., 2014) with the probability of 0.1 after the shared encoder. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate $5e^-5$ for BC2GM, BC5CDR-chem and $1e^-5$ for NCBI-disease. The training procedure contains 50 epochs for BC2GM, BC5CDR-chem and 100 epochs for NCBI-disease. A linear learning rate decay schedule with warm-up over 0.1, and a weight decay of 0.01 applied to every epochs of the training by following Liu et al. (2019). Finally, all models were trained on NVIDIA RTX TITAN and used standard F1 metrics[4] to evaluate the overall performance.

4.3 Performance

We compare our model with single-task models, such as LSTM-CRF (Habibi et al., 2017), BiLM-NER (Sachan et al., 2018), domain-specific BERTs (Devlin et al., 2019; Beltagy et al., 2019; Gu et al., 2020) and multi-task models, such as MTM-CW (Wang et al., 2019b), CollaboNET (Yoon et al., 2019), MT-BioNER (Khan et al., 2020).

Table 2 shows the overall performance of our model compared with the existing approaches on

---

[4] https://github.com/chakki-works/seqeval

Model	Dataset								
	BC2GM			BC5CDR-chem			NCBI-disease		
	P	R	F1	P	R	F1	P	R	F1
Ours	84.42	85.14	84.78	93.29	94.69	93.98	88.90	90.94	89.91
w/o bCLS	83.72	84.71	84.21	92.82	94.06	93.43	87.41	91.14	89.24
w/o mCLS	84.48	84.28	84.39	93.82	93.67	93.75	88.12	91.15	89.61
w/o mtCLS	83.44	84.82	84.12	93.98	93.57	93.78	87.48	91.67	89.52
w/o Joint	83.68	83.26	83.47	93.41	92.88	93.14	87.30	89.48	88.37

Table 4: Ablation study results, where **w/o Joint** means using the training strategy of MT-DNN (Liu et al., 2019) to replace our training algorithm.

Dataset	Model	Result
BC2GM	BioBERT	They are growth – inhibited by TGF - beta1 .
	Ours	They are growth – inhibited by TGF - beta1 .
BC5CDR-chem	BioBERT	Anaesthesia with a propofol infusion and avoidance of serotonin onists provided a nausea - free
	Ours	Anaesthesia with a propofol infusion and avoidance of serotonin onists provided a nausea - free
NCBI-disease	BioBERT	We conclude that paternal transmission of congenital DM is rare and preferentially occurs with onset of DM ...
	Ours	We conclude that paternal transmission of congenital DM is rare and preferentially occurs with onset of DM ...

Table 5: Case study on three datasets, where words in red and in green represent incorrectly and correctly recognized entities, respectively.

the three benchmark datasets, where the current SOTA model is BioBERT. In line with the expectations, Ours_MAX, which uses the max pooling strategy, achieved the best results, with the improvements of 0.40, 0.37 and 0.91 F1-scores for the three datasets, respectively. On the contrary, Ours_CLS and Ours_MEAN achieved negative results from our experiments. The above-mentioned phenomenon is consistent with Reimers and Gurevych (2019); Kruengkrai et al. (2020). Another interesting result is that our best model also achieves higher recall score than all the other approaches expect SOTA* result in BC2GM, which indicates that the introducing of coarse-grained tasks helps the model to predict more positive results.

To simulate low-resource scenarios, we also used the reduced training datasets by randomly removing sentences in training sets, while test sets are not modified. As shown in Table 3, CS-MTM was a multi-task model with cross-sharing structure proposed by Wang et al. (2019a), we record the performance of different situations and the best F1-score for each resource size are bolded. When training sets are reduced and test sets are kept, the missing information in removed sentences make all models produce worse results. However, for 50%-size, 25%-size and 10%-size datasets, our model can get an average of 0.56, 0.79 and 1.72 F1-score improvements over the BioBERT, which demonstrates our designed auxiliary tasks can regularize model to generate more robust token-level representations. For BC2GM, BC5CDR-chem and NCBI-disease in all data size, our model can get an average of 0.47, 0.95 and 1.26 improvements in F1-score, which the largest improvement is observed on NCBI-disease. The smaller the training set is the larger improvement could achived by our model. This finding proves our method is more effective in low-resource scenarios. Specifically, on 10%-size NCBI-disease, our model can get 3.29 F1-score improvements over the BioBERT.

To prove that our joint training algorithm is effective, we plot the performance curve of different tasks, which can be found in Figure 4. Moreover, different task combinations can produce different results in multi-task learning. To measure the impact of our designed auxiliary tasks and training algorithm, we conducted ablation studies and showed in Table 4.

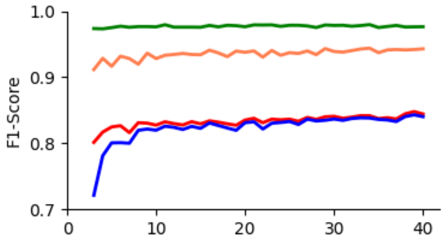

(a) BC2GM

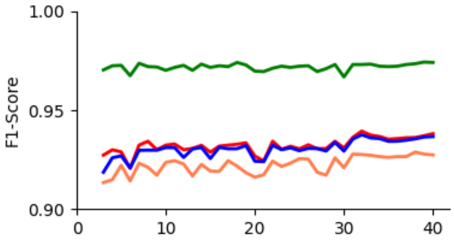

(b) BC5CDR-chem

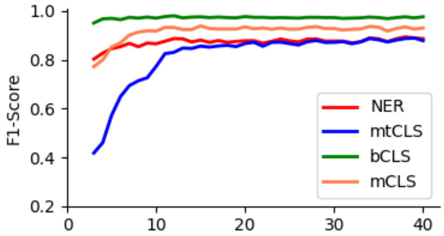

(c) NCBI-disease

Figure 4: Performance curves for different tasks of our proposed model. Where x-axis is the number of training epoch and y-axis is F1-score. During the model training process, the tasks complement each other and gradually converge.

From the results of the ablation experiment, removing the joint training algorithm leads to a consistent drop in the F1-scores. Compared with the results of Liu et al. (2019), Khan et al. (2020) and Peng et al. (2020), we point out that multi-task learning algorithms such as MT-DNN require a large amount of training data to achieve improvements. Furthermore, all of the auxiliary tasks are helpful to the main task but the impact of different tasks vary. Specifically, mtCLS is the best partner for BC2GM dataset and bCLS can bring the most improvement for BC5CDR-chem and NCBI-disease dataset. This phenomenon shows that different BioNER datasets have different recognition difficulties. For example, the recognition difficulty of BC2GM may mainly related to entity boundary problem, while the recognition difficulty of BC5CDR-chem and NCBI-disease is the entity sparsity problem. Therefore, for BC5CDR-chem and NCBI-disease, the model trends to incorrectly recognize entities in sentences that do not contain entities. This finding is consistent with our statistical results, where such cases are 2.38%, 2.71% and 2.87% on the three datasets, respectively. Compared to bCLS, mCLS is less helpful. This implies that the effect of auxiliary tasks in multi-task leaning is closely related to their performances. In fact, the classification performances of mCLS are lower than that of bCLS due to its higher difficulty.

4.4 Case study

Table 5 shows the case study of three datasets. The BC2GM example showed the effect of bCLS task in that our model could correctly recognize the entity “TGF - beta1” while the BioBERT model fails. In the BC5CDR-chem example, the input sentence contains two entities “propofol” and “serotonin”, and the BioBERT model could only identify one of them, while our model could correctly recognize two entities by incorporating the mCLS task. For the NCBI-disease example, “congenital DM” is a multi-token entity and “DM” is a single-token entity. It could be found that without the help of the mtCLS task, the BioBERT model could not capture such difference and incorrectly recognized two “DM”. Overall, these examples confirm that supervised objectives at different granularities, i.e. global information and local information, can be combined to help producing better representations.

Although the case study show that our model with auxiliary tasks outperformed the BioBERT model, these tasks can not completely solve the above problems due to their coarser granularities. Take the bCLS task as an example, the model could noticed that current input sentence contains entities by sentence-level label, but still may trapped in the number of entities or entity boundary.

4.5 Impacts of the task relationship

In this subsection, we would like to preliminary study **the relationship between different tasks in the same domain**, such as the interaction between sentence-level tasks and token-level tasks, and whether or not tasks could help one other. Therefore, we conducted pair-wise comparison experiments, as shown in Figure 5, where x-axis is the secondary task and y-axis is the main task.

First, we point out the token-level labels are more helpful for the sentence-level tasks. For mCLS, it can get an average improvement of 0.79%, 0.54% and 0.15% on the three datasets by taking mtCLS, NER and bCLS as auxiliary tasks, respectively. Considering that mtCLS and NER are token-level tasks and bCLS is a sentence-level task,

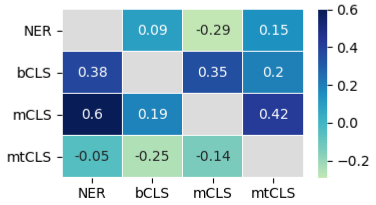

(a) BC2GM

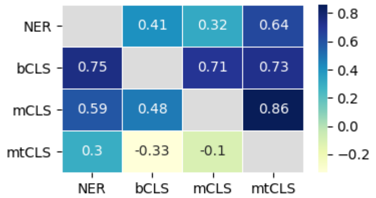

(b) BC5CDR-chem

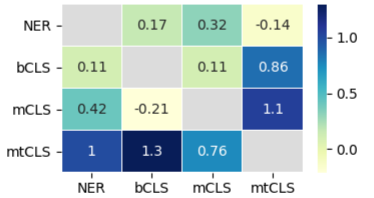

(c) NCBI-disease

Figure 5: Results of the pair-wise experiments of our model on three datasets.

the results suggest that the coarse-grained tasks can significantly benefit from fine-grained tasks. This finding could be used to guide the choosing of the tasks for multi-task learning.

Second, the same granularity of information also contributes to each other. Concretely, bCLS and mCLS can get an average improvement of 0.39%, 0.15% from mCLS and bCLS, respectively. And mtCLS and NER can get an average improvement of 0.42%, 0.22% from NER and mtCLS, respectively. Meanwhile, the difficulty of task is also a factor that affects the effectiveness of multi-task learning, in that bCLS gets 0.24% more improvements compared to mCLS, and mtCLS gets 0.20% more improvements compared to BioNER.

In addition, the same task combinations performs differently on different datasets. For example, the combinations of mtCLS and mCLS got negative results of -0.25% and -0.33% on the BC2GM and BC5CDR-chem datasets, while achieved 1.3% boost on the NCBI-disease dataset. We guessed it may related to the transferability of specific dataset. So we visualized the task embedding of three datasets, which were generated with the method[5] proposed by Vu et al. (2020), using T-SNE (Belkina et al., 2019) dimension reduction algorithm and showed the results in Figure 6. From the visualization results, we found that the embedding distance between the same tasks (e.g., BC2GM-bCLS, BC5CDR-chem-bCLS, NCBI-disease-bCLS) or the same type of tasks is closer (e.g., NER and mtCLS, bCLS and mCLS). And the embedding distance between different types of tasks is farther (e.g., bCLS and NER), but more specific relations need further exploration.

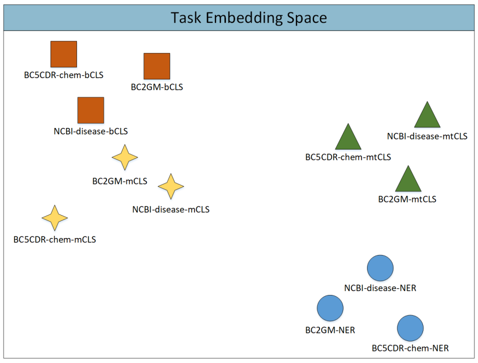

Figure 6: A 2D visualization of the tasks space.

5 Conclusion

In this work, we investigated whether coarse-grained label could benefit the token-level representation for BioNER. We had shown that the proposed BERT-based jointly sentence and token label model was valid without using external data and hand-crafted feature for BioNER in three datasets: BC2GM, BC5CDR-chem, NCBI-disease. Finally, we preliminary discussed the correlation between main task and auxiliary task.

For multi-task learning, describing and reasoning about the relations between tasks through experiments require an amount of computational resources. In future work, with domain related in mind, we will explore efficient methods for generating vectorial representations to measure the relationship between different NLP tasks.

Acknowledgements

The authors would like to thank the three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant Nos. 62076211, U1908216 and 61573294.

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by

---

[5] https://github.com/tuvuumass/task-transferability

jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 797–807.

Sofia J Athenikos and Hyoil Han. 2010. Biomedical question answering: A survey. *Computer methods and programs in biomedicine*, 99(1):1–24.

Anna C Belkina, Christopher O Ciccolella, Rina Anno, Richard Halpert, Josef Spidlen, and Jennifer E Snyder-Cappione. 2019. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications*, 10(1):1–12.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017. A neural network multi-task learning approach to biomedical named entity recognition. *BMC bioinformatics*, 18(1):368.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. Linnaeus: a species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):85.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *arXiv preprint arXiv:2007.15779*.

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

Anwen Hu, Zhicheng Dou, Jian-Yun Nie, and Ji-Rong Wen. 2020. Leveraging multi-token entities in document-level named entity recognition. In *AAAI*, pages 7961–7968.

Chen Jia, Xiaobo Liang, and Yue Zhang. 2019. Cross-domain NER using cross-domain language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2464–2474, Florence, Italy. Association for Computational Linguistics.

Muhammad Raza Khan, Morteza Ziyadi, and Mohamed AbdelHady. 2020. Mt-bioner: Multi-task learning for biomedical named entity recognition using deep bidirectional transformers. *arXiv preprint arXiv:2001.08904*.

Donghyeon Kim, Jinhyuk Lee, Chan Ho So, Hwisang Jeon, Minbyul Jeong, Yonghwa Choi, Wonjin Yoon, Mujeen Sung, and Jaewoo Kang. 2019. A neural named entity recognition and multi-type normalization tool for biomedical text mining. *IEEE Access*, 7:73729–73740.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Canasai Kruengkrai, Thien Hai Nguyen, Sharifah Mahani Aljunied, and Lidong Bing. 2020. Improving low-resource named entity recognition using joint sentence and token labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5898–5905, Online. Association for Computational Linguistics.

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Peter White. 2004. Integrated annotation for biomedical information extraction. In *HLT-NAACL 2004 Workshop: Linking Biological Literature, Ontologies and Databases*, pages 61–68.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. 2015. Drug name recognition: approaches and resources. *Information*, 6(4):790–810.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Ling Luo, Zhihao Yang, Pei Yang, Yin Zhang, Lei Wang, Hongfei Lin, and Jian Wang. 2018. An attention-based bilstm-crf approach to document-level chemical named entity recognition. *Bioinformatics*, 34(8):1381–1388.

Rashmi Mishra, Jiantao Bian, Marcelo Fiszman, Charlene R Weir, Siddhartha Jonnalagadda, Javed Mostafa, and Guilherme Del Fiol. 2014. Text summarization in the biomedical domain: a systematic review of recent research. *Journal of biomedical informatics*, 52:457–467.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. *arXiv preprint arXiv:1906.05474*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Marek Rei and Anders Søgaard. 2019. Jointly learning to label sentences and tokens. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6916–6923.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Devendra Singh Sachan, Pengtao Xie, Mrinmaya Sachan, and Eric P Xing. 2018. Effective use of bidirectional language modeling for transfer learning in biomedical named entity recognition. In *Machine learning for healthcare conference*, pages 383–402. PMLR.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(S2):S2.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online. Association for Computational Linguistics.

Hanna M Wallach. 2004. Conditional random fields: An introduction. *Technical Reports (CIS)*, page 22.

Xi Wang, Jiagao Lyu, Li Dong, and Ke Xu. 2019a. Multitask learning for biomedical named entity recognition with cross-sharing structure. *BMC bioinformatics*, 20(1):427.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2019b. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, 35(10):1745–1752.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

Wonjin Yoon, Chan Ho So, Jinhyuk Lee, and Jaewoo Kang. 2019. Collabonet: collaboration of deep neural networks for biomedical named entity recognition. *BMC bioinformatics*, 20(10):249.

Deyu Zhou, Dayou Zhong, and Yulan He. 2014. Biomedical relation extraction: from binary to complex. *Computational and mathematical methods in medicine*, 2014.

EBERT: Efficient BERT Inference with Dynamic Structured Pruning

Zejian Liu[1,2], **Fanrong Li**[1,2], **Gang Li**[1] **and Jian Cheng**[1,2]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS
[2]School of Future Technology, University of Chinese Academy of Sciences
`{liuzejian2018, lifanrong2017}@ia.ac.cn`, `gangli0426@gmail.com`,
`jcheng@nlpr.ia.ac.cn`

Abstract

Pruning has been demonstrated as an effective way of reducing computational complexity for deep networks, especially CNNs for computer vision tasks. In this paper, we investigate the opportunity to accelerate the inference of large-scale pre-trained language model via pruning. We propose EBERT, a dynamic structured pruning algorithm for efficient BERT inference. Unlike previous methods that randomly prune the model weights for static inference, EBERT dynamically determines and prunes the unimportant heads in multi-head self-attention layers and the unimportant structured computations in feed-forward network for each input sample at run-time. Experimental results show that our proposed EBERT outperforms other state-of-the-art methods on different tasks.

1 Introduction

In the last few years, transformer-based (Vaswani et al., 2017) large-scale pre-trained language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and GPT-3 (Brown et al., 2020), have achieved state-of-the-art results on many NLP tasks, including language understanding, question answering, and reading comprehension. Most recently, researchers also successfully applied transformer-based models to computer vision tasks, achieving comparable or superior performance compared to traditional convolutional networks. For example, Carion et al. (2020) propose detection transformer (DETR) for object detection, Dosovitskiy et al. (2021) design a transformer-based model, namely Vision Transformer (ViT), for image classification. However, due to the notable computational complexity and memory footprint, it is difficult for these models to deploy on hardware platforms under moderate computing and resource budget. Therefore, how to reduce model complexity to enable efficient inference for large-scale pre-trained language models is a critical issue.

Pruning is a commonly used technique for network compression, which has been widely explored to reduce computation and storage requirements of convolutional neural networks for computer vision tasks (Han et al., 2015, 2016; Li et al., 2016). However, *can transformer-based models benefit from pruning?* Michel et al. (2019) observe that a large percentage of attention heads can be removed with negligible performance drop, which indicates that the importance of different heads in same layer is different. Sanh et al. (2020) propose a simple, deterministic first-order weight pruning method which can prune lots of parameters with minimal accuracy loss. Although these methods are able to reduce the memory footprint, they cannot achieve real performance gain on general-purpose hardware, such as GPGPU, due to the unstructured sparsity after pruning.

Adaptive inference strategy is also proposed to accelerate the inference of BERT. It is based on two observations: 1) the input samples usually have different levels of difficulty. For a given model, it may over-calculate the simple samples while fail in complex samples (Liu et al., 2020); 2) similar to convolutional neural networks, the lower and higher layers of transformer extract different information, and features provided by the intermediate layers may be enough for some samples (Xin et al., 2020). FastBERT (Liu et al., 2020) and DeeBERT (Xin et al., 2020) are two state-of-the-art adaptive inference models for compressing BERT. Both of them insert extra classification layers between each layer of the network. During inference, each input sample only goes through part of model when the outputs of extra classifiers meet predefined criteria like entropy and uncertainty. Because the number of executed layers is reduced, real speedup can be achieved. However, skipping all the computations

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4814–4823
August 1–6, 2021.

of the remaining layers may be harmful to the accuracy.

In this paper, we propose EBERT, a hardware-friendly, simple yet effective algorithm that incorporates structured pruning with adaptive inference for efficient BERT inference. Specifically, EBERT inserts predictors for self-attention sub-layer and feed-forward sub-layer in each transformer block, as illustrated in Figure 1. During inference, the predictors dynamically determine which heads of self-attention layers and channels of feed-forward network can be pruned according to current input. Once a head or a channel is pruned, the corresponding computations and memory cost can be completely avoided. Compared with static pruning methods that permanently prune some parameters, it can avoid prune important parameters for current input samples which will cause large performance drop. To the best of our knowledge, it's the first time to apply dynamically structured pruning to BERT. Experimental results on different benchmark demonstrate that the proposed EBERT can achieve better trade-off between computation reduction and accuracy.

2 Related Work

Adaptive inference. As different input samples usually have different levels of difficulty, using fixed-size model to process all samples may be non-optimal in terms of computational efficiency. Therefore, the main goal of adaptive inference is to adaptively skip part of the computations according to each input sample to reduce complexity. FastBERT (Liu et al., 2020) adds student classifiers to the output of each transformer block and use self-distillation strategy to improve performance. The model architecture of DeeBERT (Xin et al., 2020) is similar to FastBERT, but it use entropy of output to decide whether to exit at early stages. PABEE (Zhou et al., 2020) proposes a novel early-exit criterion that dynamically stops forward computing when the output of internal classifiers keep unchanged for a pre-defined number of steps.

Pruning. Pruning is an intuitively simple yet effective technique for model compression, which removes unimportant computations based on certain criterion. Michel et al. (2019) observe that a large percentage of attention heads can be removed with negligible performance loss and propose a greedy pruning algorithm. Compressing BERT (Gordon et al., 2020) explores the effect of unstructured weight pruning with different levels of pruning and different training stages. McCarley et al. (2019) investigate the relationship between structured pruning and task-specific distillation. SNIP (Lin et al., 2020) proposes a structured pruning method to penalize an entire residual module in Transformer model toward an identity mapping.

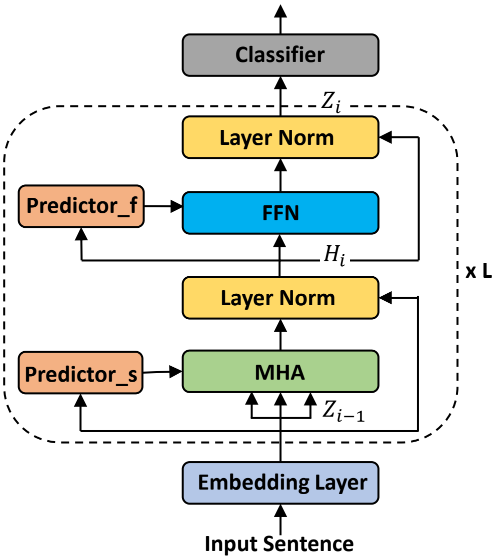

Figure 1: The overall architecture of EBERT. For each input sentence, the predictor dynamically determines which heads or channels can be pruned.

Distillation. Knowledge Distillation (Hinton et al., 2015) is an effective technique to get light models from heavy models without sacrificing too much performance. DistilBERT (Sanh et al., 2019) leverages knowledge distillation at pre-training phase to get a lighter pre-trained model, then directly fine-tunes on downstream tasks. BERT-PKD (Sun et al., 2019) proposes an incremental knowledge extraction process. Apart from learning from the final output of teacher model, student model also patiently learns from intermediate layers. TinyBERT (Jiao et al., 2020) performs distillation at both the pre-training and task-specific fine-tuning phase. Data augmentation is also used to improve the accuracy of student model.

3 Methods

In this section, we will first introduce the architecture of EBERT. As shown in Figure 1, it can be divided into BERT branch and predictor branch. Then we will describe the training and inference in details.

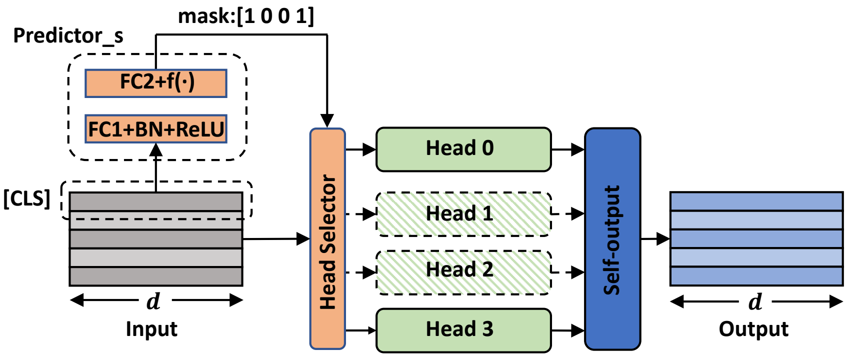

(a) Predictor_s in MHA.

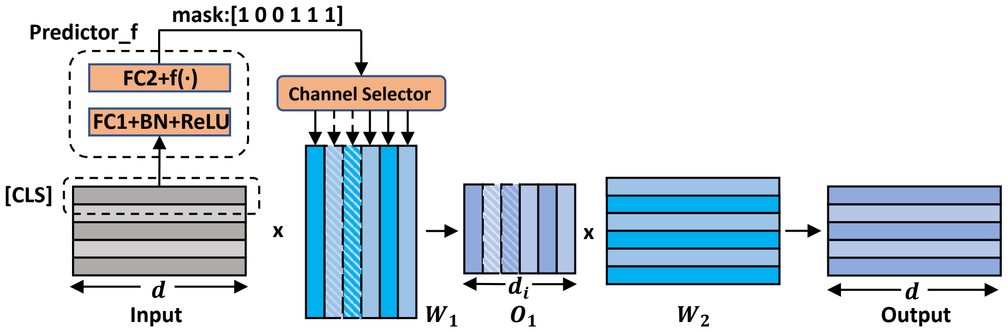

(b) Predictor_f in FFN.

Figure 2: The details of predictors in MHA and FFN layer. Here we assume that $n = 4$, $h = 4$ and $d_i = 6$. Shadow area means that computations can be skipped.

3.1 BERT Branch

The architecture of BERT consists of three parts: the embedding layer, multi-layer bidirectional Transformer encoders and the task-specific classification layer. Given an input sentence $S = [s_0, s_1, ..., s_n]$ with length n, where s_0 is usually a special classification token $[CLS]$, the embedding layer will transform it to a sequence of vector representations:

$$E = Embedding(S), E \in R^{n \times d} \tag{1}$$

The Transformer encoder contains two sublayers: multi-head self-attention (MHA) layer and position-wise fully connected feed-forward network (FFN),

$$\begin{aligned} H_i &= LN(MHA(Z_{i-1}) + Z_{i-1}) \\ Z_i &= LN(FFN(H_i) + H_i) \end{aligned} \tag{2}$$

where $i = 1, 2, ..., L$ and $Z_0 = E$. LN is the Layer Normalization operation.

The final component of BERT is a task-specific classification layer. It accepts the representation to $[CLS]$ token as input to generate final results, as:

$$O = Classifier(Z_L[0, :]) \tag{3}$$

3.2 Predictor Branch

In order to prune unimportant heads and channels for individual input sentence, we add predictors for MHA and FFN in each layer, respectively. The predictor consists of two feed-forward layer, one batch normalization layer and a ReLU activation layer, as depicted in Figure 2. The output t of the second feed-forward layer will be transformed to a 0-1 mask by a function $f(\cdot)$:

$$\begin{aligned} t &= FC2(ReLU(BN(FC1(x)))) \\ m &= f(t), m \in \{0, 1\} \end{aligned} \tag{4}$$

where $x = Z[0, :]$. It means that the input of predictor is only $[CLS]$ representation. This choice is based on two reasons. 1) *Overhead.* Although using the whole representation of input sentence may improve the performance of predictors, the amount of computations increases linearly with the sentence length n. When n is large, the computational overhead of predictors can not be ignored. 2) *Representation ability.* Because the final hidden state to $[CLS]$ token in the last transformer block is used in task-specific classifier to generate classification results, we assume that $[CLS]$ representation encodes most of the useful information of the sentence. Note that the representation to $[CLS]$ token in the first MHA is independent with the input sentence, so we use average pooling of MHA as input.

Intuitively, t represents the probability of heads or channels being selected. In order to train the model end-to-end with back propagation, Gumbel-Softmax trick (Jang et al., 2017; Maddison et al., 2016) is adopted in our model. Given class probabilities $\pi_1, \pi_2, ...\pi_n$, discrete samples z can be drawn as:

$$z = one\_hot(\arg\max_i [g_i + log\pi_i]) \tag{5}$$

where g_i is a sample drawn from a Gumbel distribution. Gumbel-Softmax trick replaces $\arg\max$ operation with a softmax function, which is a continuous differentiable approximation to $\arg\max$:

$$y_i = \frac{exp((log(\pi_i) + g_i)/\tau}{\sum_{j=1}^{k} exp((log(\pi_j) + g_j)/\tau)} \tag{6}$$

As the value of mask m is binary (0 for prune and 1 for preserve), we can simplify the Gumbel-Softmax formulation (Verelst and Tuytelaars, 2020). For the output $t[i] \in (-\infty, \infty)$, we can convert it to probabilities π_1 and π_2 by using a sigmoid function σ:

$$\begin{aligned} \pi_1 &= \sigma(t[i]) \\ \pi_2 &= 1 - \sigma(t[i]) \end{aligned} \tag{7}$$

Substituting (7) into (6), we can get:

$$\begin{aligned} y_1 &= \sigma(\frac{t[i] + g_1 - g_2}{\tau}) \\ y_2 &= 1 - y_1 \end{aligned} \tag{8}$$

As $y_1 < y_2$ means the head or channel will be pruned, the final formulation is:

$$f(t[i]) = \begin{cases} 1, & if \quad y_1 > 0.5 \\ 0, & otherwise \end{cases} \tag{9}$$

3.3 Training

The entire training process can be divided into three stages: fine-tune the BERT branch, joint train both branches, and re-train the BERT branch.

Fine-tuning. In the first stage, only BERT branch is fine-tuned on downstream tasks with loss $\mathcal{L}_{task}$. The training strategy is the same as BERT in (Devlin et al., 2019).

Joint Training. In this stage, we jointly train the pre-trained BERT branch and randomly initialized predictor branch to make the average ratio of remaining Floating-point operations (FLOPs) reach a target value $C_t \in [0, 1]$. In order to achieve this goal, we add a loss to minimize the difference between real computational cost of the whole network and C_t:

$$\mathcal{L}_s = \left(\frac{F_c}{F_o} - C_t \right)^2 \tag{10}$$

Where F_o is the FLOPs of original network, and F_c is the average FLOPs of current model in a mini-batch.

In addition to the FLOPs constraint, we also add extra loss function to control the sparsity of each MHA and FFN, as in (11). The purpose is to avoid high sparsity of some layers that is harmful to the accuracy of the model.

$$\begin{aligned} \mathcal{L}_M &= \frac{1}{L} \sum_{l=0}^{L-1} \left(\frac{F_c^{lM}}{F_o^{lM}} - C_t \right)^2 \\ \mathcal{L}_F &= \frac{1}{L} \sum_{l=0}^{L-1} \left(\frac{F_c^{lF}}{F_o^{lF}} - C_t \right)^2 \end{aligned} \tag{11}$$

where F_c^{lM} and F_o^{lM} refer to the FLOPs of l-th MHA in current model and original model. The definition of F_c^{lF} and F_o^{lF} is similar. The final loss to be optimized is then given by

$$\mathcal{L} = \mathcal{L}_{task} + \lambda_1 \mathcal{L}_s + \lambda_2 (\mathcal{L}_M + \mathcal{L}_F) \tag{12}$$

where λ_1 and λ_2 control the magnitude of task and sparsity loss, respectively.

Re-training. As different input samples usually activate different parts of heads, the total update of a particular head is less than that of regular training process. As a result, the heads are probably not trained sufficiently. So do the channels in FFNs. Therefore, in this stage, we freeze the parameters of predictors and only re-train the BERT branch.

3.4 Inference

The computation flow during inference is shown in Figure 2. Given an input sequence, the predictor generates a mask by using the representation to $[CLS]$ token. For MHA, heads with mask '0' will not be executed. For FFN, as matrix-matrix multiplication can be transformed to multiple matrix-vector multiplications, we only need to complete part of computations where vector's mask is not zero.

BERT-base	L=12, h=12, d=768, $d_i = 3072$
Predictor_s	$768 \rightarrow 64 \rightarrow 12$
Predictor_f	$768 \rightarrow 64 \rightarrow 3072$

Table 1: The detailed setting of BERT and Predictors. RoBERTa is with the same setting.

Note that the exponential operation in (8) is typically expensive on hardware. Fortunately, this formulation can be simplified during inference by removing Gumbel noise. $f(\cdot)$ now can be rewritten as:

$$f(t[i]) = \begin{cases} 1, & if \quad t[i] > 0 \\ 0, & otherwise \end{cases} \tag{13}$$

4 Experiments

4.1 Setup

Datasets and Metrics. To verify the effectiveness of EBERT, we conduct experiments on four classification tasks from GLUE benchmark (Wang et al., 2018): Multi-Genre Natural Language Inference Matched/Mismatched (MNLI-m/mm), Quora Question Pairs (QQP), Question Natural Language Inference (QNLI) and Stanford Sentiment Treebank (SST-2). We exclude other tasks as the results have large variance due to the number of training examples is very small (less than 9k). MNLI-m/mm, SST-2 and QNLI use accuracy as metric, while the average of F1 and accuracy is used for QQP.

Furthermore, we also conduct experiments on SQuAD1.1 (Rajpurkar et al., 2016) and SQuAD2.0 (Rajpurkar et al., 2018), both of which are large-scale reading comprehension datasets. SQuAD1.1 consists of more than 100k questions, and the answer to each question is a segment of text from the corresponding reading passage. SQuAD2.0 is more difficult as it contains over 50k unanswerable questions. We mainly report Exact Match (EM) and F1 scores.

Implementation details. We apply the proposed methods to both BERT-base and RoBERTa-base, and implement them with the HuggingFace Transformers Library (Wolf et al., 2020). The detailed setting of BERT and predictors is shown in Table 1. Figure 3 shows the ratio of FLOPs and parameters of each operation in one encoder. We can find that the extra cost of the predictors is very small. All experiments are completed on a single Nvidia GeForce RTX2080Ti GPU.

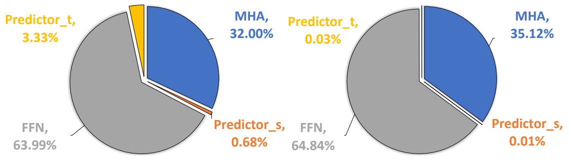

(a) #Params (b) #FLOPs

Figure 3: The ratio of FLOPs and parameters of each operation in one encoder.

For the GLUE benchmark, we set batch size to 32, learning rate to 3e-5, training epochs to 3 while other hyperparameters are kept unchanged from the library for all downstream tasks at backbone fine-tune stage. During joint training, we use $\lambda_1 = 4, \lambda_2 = 20$ for BERT while $\lambda_1 = 2, \lambda_2 = 10$ for RoBERTa. The learning rate for predictors' parameters is 0.02 and 0.01, respectively. The hyperparameters in the third stage is the same as the first stage.

For SQuAD1.1 and SQuAD2.0, the batch size is 12, learning rate is 3e-5 and training epoch is 2. Other settings are consistent with those for BERT on GLUE benchmark.

Baseline. In order to evaluate the effectiveness of EBERT, we implement a Top-k version of BERT that $f(\cdot)$ is as (14). We keep the sparsity of each layer the same, so the value of k can be decided by C_t. What's more, for a certain k, the sparsity is a fixed value, so no extra loss need to be added. The training objective is just $\mathcal{L}_{task}$. The training methods is the same as EBERT with Gumbel-Softmax.

$$f(t[i], k) = \begin{cases} 1, & if \quad t[i] \in topk(t) \\ 0, & otherwise. \end{cases} \tag{14}$$

For convenience, in the following sections we will use the subscript t to represent Top-k version and use subscript g for Gumbel-Softmax version.

4.2 Results on the GLUE benchmark

The main results of our proposed method on the development set of GLUE benchmark are shown in Figure 4. For BERT-base, the results of Gumbel-Softmax is always better than Top-k with the same or even smaller ratio of remaining FLOPs on four tasks. For example, when remaining 50% FLOPs, EBERT_g only drops 0.6% on QQP task, while EBERT_t drops 1.8%. On the MNLI task, EBERT_g's accuracy with 77% remaining FLOPs is higher than the accuracy of EBERT_t with 81% remaining FLOPs.

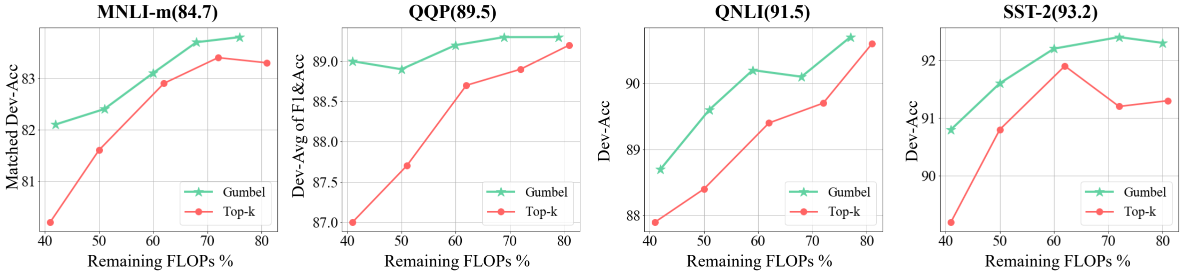

(a) BERT-base

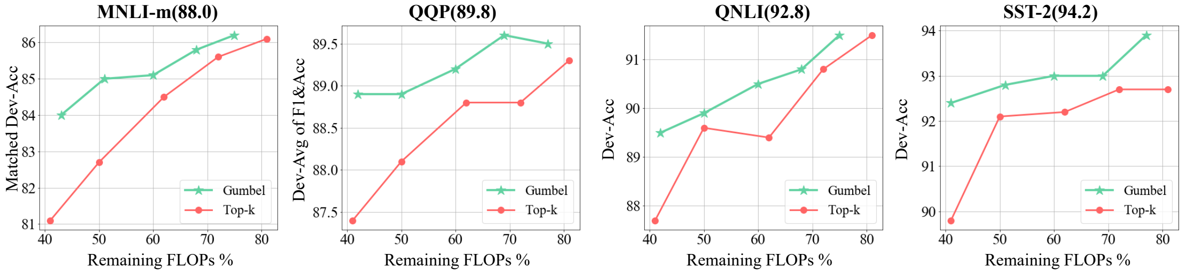

(b) RoBERTa-base

Figure 4: Results on the development set of GLUE benchmark.

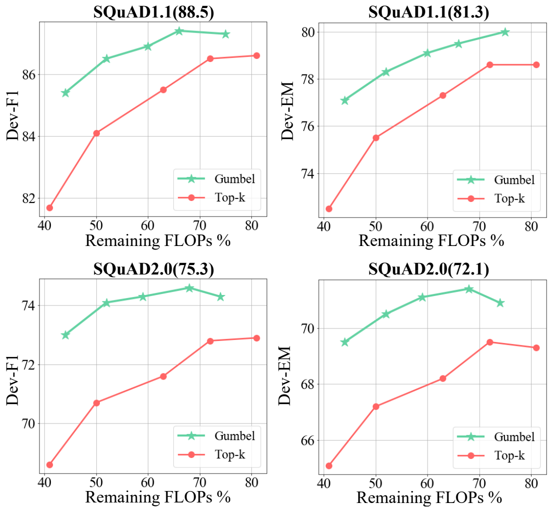

Figure 5: Results on the development set of SQuAD benchmark.

Figure 4(b) shows the performance of ERoBERTa, and we can find the similar result, *e.g.* with 50% remaining FLOPs, the performance of ERoBERTa$_g$ is 2.3% higher than ERoBERTa$_t$ on the MNLI task. This proves the generality of our proposed method to different model.

4.3 Results on the SQuAD benchmark

To further demonstrate the generality of our method, we conduct experiments on the SQuAD v1.1 and v2.0 benchmark, which are reading comprehension task that the model need to predict the answer text span in the text for a given question. The results are shown in Figure 5. Similar to the observation in Figure 4, our approach achieves consistent improvement on each ratio of remaining FLOPs compared with the Top-k version. For instance, with 50% remaining FLOPs, EBERT$_g$ improves the EM and F1 score by 2.8% and 2.4% on SQuAD v1.1, respectively. On SQuAD v2.0, the improvement of EM and F1 score is 3.3% and 3.4%.

4.4 Comparison with Other Methods

We compare our proposed EBERT with other state-of-the-art compression methods. For distillation methods, we compare with DistilBERT (Sanh et al., 2019), BERT-PKD(Sun et al., 2019) and BERT-of-Theseus (Xu et al., 2020). For pruning, we compare with SNIP (Lin et al., 2020). We also compare with other two dynamic methods: DeeBERT (Xin et al., 2020) and PABEE (Zhou et al., 2020). We do not compare with FastBERT (Liu et al., 2020) as they don't report results on the GLUE and SQuAD benchmark.

Note that other works don't report the FLOPs. However, as all of these methods try to reduce computational cost by reducing the number of layers dynamically or statically, it is reasonable to get FLOPs from speedup ratio or compression ratio under the assumption that the FLOPs is proportional to the execution time for a specific layer. For example, as the DistilBERT-6L only has half number of layers of BERT-base, we assume the ratio of remaining FLOPs is 50%.

	MNLI-m		SST-2		QQP		QNLI	
	Acc.	RF%.	Acc.	RF%.	F1/Acc.	RF%.	Acc.	RF%.
dev set								
BERT-base	84.7	100	93.2	100	87.9/91.1	100	91.5	100
DistilBERT-6L	82.2	50	91.3	50	-/88.5	50	89.2	50
BERT-PKD	81.3	50	91.3	50	-	-	-	-
SNIP	-	-	91.8	50	-/88.9	50	89.5	50
DeeBERT	80.7	63	90.0	63	-	-	-	-
PABEE	**83.6**	62	92.0	62	-	-	-	-
EBERT$_g$	82.4	51	91.6	50	87.2/90.6	50	89.6	51
	83.1	60	**92.2**	60	**87.5/90.8**	59	**90.2**	59
test set								
BERT-base	84.7	100	93.7	100	71.5/89.4	100	90.8	100
BERT-PKD	81.5	50	92.0	50	70.7/88.9	50	89.0	50
BERT-of-Theseus	82.4	50	92.2	50	**71.6/89.3**	50	**89.6**	50
DeeBERT	80.0	63	91.5	53	69.4/-	51	87.3	56
EBERT$_g$	82.4	50	92.8	50	70.1/88.8	50	89.2	50
	83.3	60	**93.4**	60	70.0/88.8	59	**89.6**	59

Table 2: Comparison with other compressed methods on the development and test set of MNLI, SST-2, QQP and QNLI. RF means the ratio of remaining FLOPs.

Table 2 lists the results on both development set and test set. The results on test set are provided by the GLUE evaluation server. Compared with other methods, our approach retains competitive performance with less FLOPs. For instance, our approach achieves the accuracy of 92.2% on SST-2 with 60% remaining FLOPs. On the test set of MNLI task, the accuracy of our method is 83.3% with only 60% remaining FLOPs, while DeeBERT' accuracy is 80.0% with 63% remaining FLOPs.

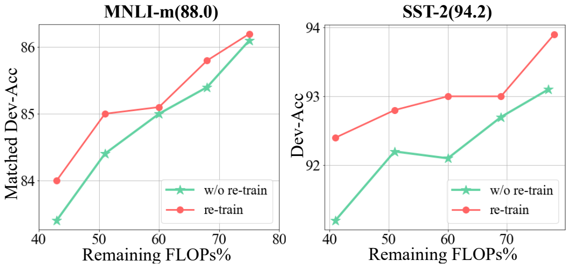

Figure 6: The effectiveness of re-training stage for RoBERTa on MNLI and SST-2.

4.5 Further Analysis

4.5.1 Impact of Re-training

The training process of EBERT contains three stages: fine-tuning, joint training and re-training. The purpose of re-training is to make each head and channel sufficiently trained. To evaluate the efficacy of this stage, we conduct experiments with RoBERTa on two tasks. Results are shown in Figure 6, we can see that the performance improvement is obvious. With 50% remaining FLOPs, the performance of the model is improved from 84.4% to 85.0% on MNLI and 92.2% to 92.8% on SST-2, respectively. The average performance improvement on MNLI and SST-2 is 0.4% and 0.8%, respectively. Comparing these two results, we find that the improvement is more obvious on small datasets. The reason for this phenomenon is that the parameters of the model are updated more frequently on large datasets, which makes the training of the model more sufficient at the joint training stage. As a result, re-training can be skipped for large datasets to make trade-offs.

4.5.2 Mask Distribution

Like in (Chen et al., 2019), we investigate the distribution of the learned masks. Although EBERT can dynamically generate mask for each head and channel for different samples, some masks may be constant of all time, which means that these masks are input-independent. Figure 8 is the layer-wise visualization of mask distribution in MHA and FFN on SST-2 task for masks that are 1) always one (on), 2) always zero (off), and 3) input-dependent. We can see that a large subset of the masks are input-dependent for both heads and channels, which indicates that our model learns to predict the im-

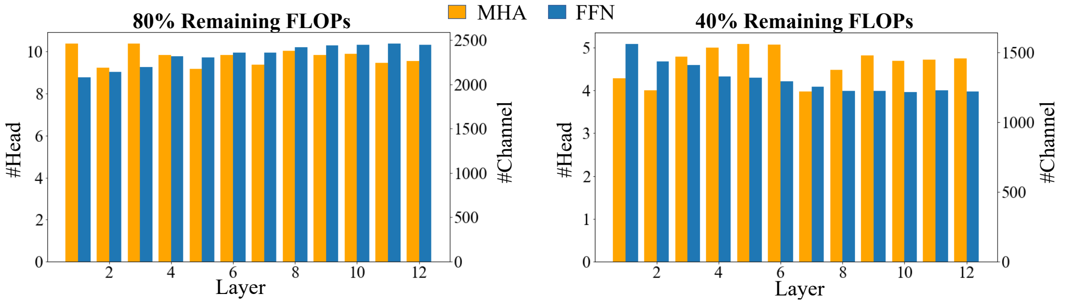

Figure 7: Average number of non-pruned heads of MHA and non-pruned channels of FFN by layer for RoBERTa-base with different remaining FLOPs on the SST-2 task.

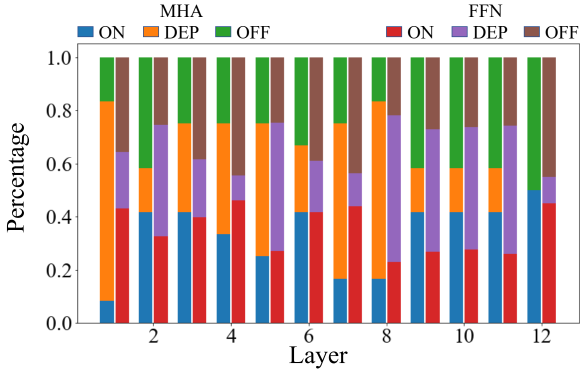

Figure 8: The distribution of masks in MHA and FFN for RoBERTa-base with 50% remaining FLOPs on SST-2 tasks. DEP refers to input-dependent.

portance of heads and channels for different input samples. For head, the proportion of masks that are input-dependent is higher in the shallow layers. For channel, the 2nd, 5th, 8th and 11th layer have higher proportion of input-dependent masks than other layers.

4.5.3 Layer Distribution

In Section 3.3, we add two extra loss $\mathcal{L}_M$ and $\mathcal{L}_F$ to prevent some layers from being too sparse. We conduct experiments on SST-2 task with RoBERTa to verify the effectiveness of these constraints. Figure 7 shows the average number of non-pruned heads of MHA and non-pruned channels of FFN with different ratio of remaining FLOPs. We can see that the number in each layer is quite close, which indicates the average amount of calculations is similar. More importantly, this value is near the target C_t. For example, when remaining 80% FLOPs, the number of non-pruned heads is around 9, which is exactly 80% of the number of heads in one MHA. Similarly, the number of non-pruned heads are around 4 and 5 when remaining 40% FLOPs. This phenomenon proves that $\mathcal{L}_M$ and $\mathcal{L}_F$ do limit the sparsity of each layer.

5 Conclusion and Future Works

In this paper, we propose a novel pruning method for efficient BERT inference, which is called EBERT. With the help of predictor branch, EBERT can dynamically prune unimportant heads in MHA and unimportant channels in FFN for each input sample at run-time. Compared with other compression methods, experiments on GLUE and SQuAD benchmarks demonstrate that EBERT can achieve better accuracy-efficiency trade-off.

As we talk about in Section 4.1, the performance of our method on small dataset has large variance. Similar observations also have been mentioned in other works (e.g. SNIP). In order to improve the generality of our method, it would be interesting to find out the exact reason and find the corresponding solution.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (No.61972396), National Key Research and Development Program of China (No. 2020AAA0103402), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA27040300).

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario

Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *ECCV*, pages 213–229, Cham. Springer International Publishing.

Zhourong Chen, Yang Li, Samy Bengio, and Si Si. 2019. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28, pages 1135–1143. Curran Associates, Inc.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710.

Zi Lin, Jeremiah Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning redundant mappings in transformer models via spectral-normalized identity prior. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 719–730. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables.

J.S. McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structed pruning a bert-based question answering model. *arXiv preprint arXiv:1910.06360*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

T. Verelst and T. Tuytelaars. 2020. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2317–2326.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*.

Strong and Light Baseline Models for Fact-Checking Joint Inference

Kateryna Tymoshenko
DISI, University of Trento
38123 Povo (TN), Italy
kateryna.tymoshenko@unitn.it

Alessandro Moschitti[*]
Amazon
Manhattan Beach, CA 90266, USA
amosch@amazon.com

Abstract

How to combine several pieces of evidence to verify a claim is an interesting semantic task. Very complex methods have been proposed, combining different evidence vectors using an evidence interaction graph. In this paper, we show that in case of inference based on transformer models, two effective approaches use either (i) a simple application of max pooling over the Transformer evidence vectors; or (ii) computing a weighted sum of the evidence vectors. Our experiments on the FEVER claim verification task show that the methods above achieve the state of the art, constituting strong baseline for much more computationally complex methods.

1 Introduction

Automatic Fact Checking is quickly gaining attention of the NLP and AI communities. The FEVER.ai Fact Extraction and Verification Shared Task (Thorne et al., 2018) provides a benchmark for evaluating fact-checking systems. In FEVER, given a claim, C, and a collection of approximately five million Wikipedia pages, W, the task is to predict whether C is *supported* (`SUP`) or *refuted* (`REF`) by W, or whether there is *not enough information* (`NEI`) in W to support or refute C. If C is classified as `SUP` or `REF`, the respective evidence should be provided. Tab. 1 shows a FEVER claim and the gold-standard evidence refuting it.

Claim	**Evidence**
Coeliac disease is not treated by maintaining a gluten-free diet. (`REF`)	[(`Coeliac_disease`, "The only known effective treatment is a strict lifelong gluten-free diet....")]

Table 1: FEVER data examples

The overall task is complex, as one needs to retrieve the documents that contain the evidence (*document retrieval*, *DocIR*), select relevant evidence (*evidence selection*, *ES*) and label the claim given the evidence (*evidence reasoning*, *ER*), which is the focus of our work. Formally, given a claim, C, and a list top K evidence sentences, $(E_1, ..., E_K)$, retrieved with *DocIR* and selected by *ES* respectively, the *ER* task is to predict the claim label (`SUP`/`REF`/`NEI`).

There can be multiple inter-dependent evidence sentences per claim, thus joint modeling them allows for taking multiple clues into account, thus intuitively improving system accuracy. Indeed, individual sentences may not constitute standalone evidence, but they can contain several clues, which, together, can support or refute the claim. For example, Sentence 8 of the `Gluten-free_diet` Wikipedia page, *"..gluten-free diet is demonstrated as an effective treatment, but several studies show that about 79 % ... an incomplete recovery of the small bowel..."*, which is not listed as ground truth evidence for the claim, still supports the `REF` signal.

Given the above intuition, recent state-of-the-art (SOTA) approaches (Zhou et al., 2019; Ye et al., 2020; Liu et al., 2020; Zhong et al., 2020; Zhao et al., 2020) combine different pieces of evidence with graph networks, also increasing computational and space complexity. In this paper, we show that simple joint transformer-based methods achieve better performance than the best complex systems. Specifically, we (i) text-*concatenate* evidence sentences, (ii) apply *max pooling* to their individual embedding representation, or (iii) compute their *weighted sum*. Since June 1^{st} 2021, our baseline is sixth in terms of Label Accuracy (LA) and seventh in terms of FEVER score on the official task leaderboard[1], where the absolute difference from the fourth top LA is 0.2%.

We believe our results are important to enable

---

[*] Professor at the University of Trento.

[1] https://competitions.codalab.org/competitions/18814#results

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4824–4830
August 1–6, 2021.

researchers to select the right scientific challenge, providing the appropriate baselines. For example, proposing complex models that are less accurate than our baselines can most likely mislead the research community, thus knowing our baselines can help to lead research in this area in the right directions. Additionally, our baselines are strong, simple to use, and easily reproducible, enabling fast comparison with innovative inference models.

2 Related work

SOTA approaches. Most recent approaches encode claim and evidence texts using Transformer-based language representation models (LRM), such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and others. **GEAR** (Zhou et al., 2019) and **KGAT** (Liu et al., 2020) construct graphs with evidences as nodes and use deep graph neural ER networks to propagate knowledge; **DREAM** (Zhong et al., 2020) reasons on a graph built on top of a semantic role labeler output; **Transformer-XH** (Zhao et al., 2020) propagates knowledge between `[CLS]` tokens of different evidence pieces; **CorefBERT** (Ye et al., 2020) trains a BERT-based LRM, which employs an additional objective modeling coreference knowledge, and use it within KGAT architecture. The winners of the original FEVER competition (Nie et al., 2019) used older LRMs and a modified enhanced sequential inference model (ESIM) (Chen et al., 2017) to do ER.

The top-scoring *published* approach, **DOMLIN++** (Stammbach and Ash, 2020), simply text-concatenates evidence pieces and uses a RoBERTa-based classifier, thus supporting our thesis that simple models can be very effective. On the other hand, they use additional DocIR components and data (MultiNLI (Williams et al., 2018) corpus) for fine-tuning. To the best of our knowledge, their code/output are not available online yet[2], so we cannot compare to them directly at the moment.

Baselines. The baselines in the above works, apart from the previous SOTA systems, consist in applying a transformer-based classifier to (i) *concatenation* of C and all E_i, $i = 1..K$ (Zhou et al., 2019; Zhong et al., 2020; Zhao et al., 2020); or (ii) *separate* (C, E_i) pairs, $i = 1..K$, and aggregating the results heuristically (Zhou et al., 2019). The latter also considered *max-pooling* and *weighted-sum* baselines, but used them only on subsets of the development set with multiple gold evidence pieces per claim. In this work, we use them in the full-scale setting.

3 Strong baseline models

BERT for classification. BERT LRM and its version with the improved training procedure, RoBERTa, have obtained outstanding results on a number of NLP tasks. When using BERT-based architectures for classification, a special `[CLS]`[3] token is prepended to an input text sequence. Its embedding from the last layer of the transformer, $h^{[CLS]} \in \mathbb{R}^{h_{dim}}$ is a vector representation of the sequence. h_{dim} is the hidden dimension size. The final prediction is $p = softmax(L)$, where $L = Wh^{[CLS]} \in \mathbb{R}^N$, $W \in \mathbb{R}^{N \times h_{dim}}$, and N is the number of classes[4].

Baseline approaches. We investigate four simple Transformer-based baseline approaches: **Local**, **Concat**, **MaxPool**, **WgtSum**.

The input to the task are a claim, C, and a list of top K evidence sentences selected by an ES component, $E = \{E_i\}$, $i = 1, .., K$. Tab. 2 describes the input format. Following Liu et al. (2020), we incorporate E_i source page name into the input. We use cross-entropy loss to train all the models.

Local: for each E_i, we (i) use the standard 3-way classification Transformer-based model, T_{class}, to get an evidence-level label prediction, P_i, along with its corresponding $l_i = max(L_i)$ score, where $L_i \in \mathbb{R}^N$ is the logits vector produced by T_{class} for E_i; (ii) sort the predictions list, $P = [(P_i, l_i)]$, on l in the reverse order; (iii) create P', a sublist of P, where P_i is not `NEI` and $l_i > 0$. If P' is not empty, P'_1 is the claim label, otherwise it is `NEI`. We introduce P', because we want to capture the `SUP`/`REF` signal even if it is weaker compared to that of `NEI`.

Concat: T_{class} run on the input described in Tab. 2. **Local** and **Concat** are similar to the Bert-pair and Bert-Concat baselines, respectively introduced in (Zhou et al., 2019).

MaxPool: encodes each (C, E_i) pair with a transformer model, concatenates the resulting $h_i^{[CLS]}$ into a matrix $H^{[CLS]} \in \mathbb{R}^{h_{dim} \times K}$ and max-pools it, column-wise, into $h_{mp}^{[CLS]} \in \mathbb{R}^{h_{dim}}$. The output

---

[2] We could try to re-implement their pipeline following the high-level descriptions in their paper, however, our re-implementation still will not be able to re-produce their ER input due to the inevitable implementation differences

[3] This is standard for BERT, other language models can use a different token in a different position

[4] This strategy is employed by BERT. Practical implementations of the other models can also apply more transformations to $h^{[CLS]}$ to obtain L.

Baseline	**Input type example**
Local, **MaxPool**, **WgtSum**	`[CLS]` C `[SEP]` E_i^{page} `<psep>` E_i `[SEP]`
Concat	`[CLS]` C `[SEP]` E_1^{page} `<psep>` E_1 `[SEP]` ... `[SEP]` E_K^{page} `<psep>` E_K `[SEP]`

Table 2: Input data for the baselines. E_i^{page} is the name of the E_i source Wikipedia page. `[SEP]` and `[CLS]` are the standard "separator" and "classification" tokens used in BERT-like models. `<psep>` is delimiter separating page name from the evidence text. We use "`. `", while Liu et al. (2020) use `[SEP]`

is $p = softmax(W_{mp} h_{mp}^{[CLS]})$, $W_{mp} \in \mathbb{R}^{3 \times h_{dim}}$. It is inspired by the max pooling evidence aggregation procedure employed by (Hanselowski et al., 2018; Zhou et al., 2019).

WgtSum: encodes each (C, E_i) pair with a transformer, computes the weighted sum $h_{ws}^{[CLS]} = \sum_{i=1}^{K} \alpha_i h_i^{[CLS]} \in \mathbb{R}^{h_{dim}}$, $\alpha_i = softmax_i(W_{ws} h_i^{[CLS]})$, $W_{ws} \in \mathbb{R}^{1 \times h_{dim}}$. The weight α_i is intended to reflect the relative importance of E_i. The output is $p = softmax(W h_{ws}^{[CLS]})$, $W \in \mathbb{R}^{3 \times h_{dim}}$. **WgtSum** is similar to the Zhou et al. (2019)'s attention baseline in the sense that we aggregate pieces of evidence representations via a weighted summation. However, differently from us, they obtain the weights by computing attention between the claim and the evidence hidden states. We refer to **Concat**, **MaxPool** and **WgtSum** as *global* systems.

4 Experiments

Implementation. Our system is an AllenNLP pipeline (Gardner et al., 2017). Our code is available at `https://github.com/iKernels/reasoning-baselines`. We use the pre-trained BERT and RoBERTa LRMs from the *transformers*[5] library, namely *bert-base-cased*, *roberta-base* and *roberta-large*.

Training setup. We train for three epochs, with an evaluation checkpoint every 500 and 2500 training steps for global and local models correspondingly, thus having 14 checkpoints in total. We use $K = 5$ evidence pieces per claim. For all the models the batch size/number of gradient accumulation steps are 8/8 and 2/32 with base and large LRMs, respectively. We use Adam optimizer with slanted triangular learning rate (Howard and Ruder, 2018), $cut\_frac = 0.1$, ratio of 32[6].

When experimenting with *roberta-base* we tried learning rates [1e-5; 5e-5] with the step of 1e-5 and observed no noticeable difference between the rates in the range [2e-5; 5e-5]. Additional details are available in the appendix.

Split	**#SUP**	**#REF**	**#NEI**
TRAIN	80,035	29,775	35,639
DEV	6,666	6,666	6,666
TEST	6,666	6,666	6,666

Table 3: FEVER dataset statistics. # denotes the number of claims in a given class.

FEVER metrics. The primary shared task metric is FEVER[7]. It takes the correctness of the evidence set provided with the claim label[8] into account. The evidence set must contain all sentences belonging to at least one evidence[9] associated with a claim. No evidence is needed for `NEI` claims. *Label Accuracy (LA)* is another standard metric. *Oracle FEVER (OFEVER)* is the FEVER metric computed using oracle downstream components after DocIR or ES, i.e., it estimates downstream component's upper bound performance.

4.1 The dataset

We conduct our experiments on the official FEVER 1.0 Shared Task dataset[10]. Tab. 3 reports the FEVER 1.0 statistics. Verifiable (`SUP` or `REF`) claims are associated with at least one evidence. 35.23% of verifiable claims in DEV are associated with multiple evidence sentences, independent or inter-dependent.

Evidence Reasoning (ER) dataset. We run the ER experiments on the evidence sentences retrieved by Liu et al. (2020), published on their github[11] with ES OFEVER score of 96.25. Their *DocIR* module retrieves documents for a given claim via entity linking following (Hanselowski et al., 2018), and the ES module selects relevant evidence (*ES*) via BERT-based system with pairwise loss.

Following (Liu et al., 2020), when training and selecting the best checkpoint we use gold evidence completed with the non-gold evidence pieces retrieved by ES, so that the total amount of evidence pieces per claim is K. When evaluating on DEV we simply use top 5 evidence pieces retrieved by ES, i.e. the results in Sec. 4.2 are obtained on the

---

[5] `https://github.com/huggingface/transformers`

[6] Standard values suggested in (Howard and Ruder, 2018)

[7] Scorer: `https://github.com/sheffieldnlp/fever-scorer`

[8] At least one of top 5 predicted evidences must be correct.

[9] An evidence consists of one or more sentences. One claim can have multiple evidences.

[10] FEVER 1.0 Shared Task at `https://fever.ai/resources.html`

[11] `https://github.com/thunlp/KernelGAT/tree/master/data`

		FEVER	**LA**	**LRM**
	KGAT SOTA baseline, lr=5e-05			
1:	KGAT$_{pub}$	75.88	78.02	bert-base-cased
2:	(Liu et al., 2020)	76.11	78.29	roberta-large
	Reproducing KGAT results, lr=5e-05			
3:	KGAT	75.64	77.80	bert-base-cased
4:		77.21	79.52	roberta-base
	Other learning rates (lr) for KGAT			
5:	KGAT, lr=2e-5	74.87	77.15	bert-base-cased
6:		77.66	79.98	roberta-base
7:		78.66	80.77	roberta-large
8:	KGAT, lr=3e-5	75.28	77.48	bert-base-cased
9:		77.75	80.06	roberta-base
	Local models, lr=2e-05			
10:	Aggr. heuristic 1	73.05	75.11	bert-base-cased
11:		75.62	77.85	roberta-base
12:	Aggr. heuristic 2	71.79	73.66	bert-base-cased
13:		73.98	75.96	roberta-base
	Global baselines, lr=2e-05			
14:	Concat	74.23	76.51	bert-base-cased
15:		77.09	79.25	roberta-base
16:		78.27	80.31	roberta-large
17:	MaxPool	74.72	76.99	bert-base-cased
18:		77.48	79.82	roberta-base
19:		78.85	81.16	roberta-large
20:	WgtSum	74.48	76.85	bert-base-cased
21:		77.62	80.01	roberta-base
22:		**79.02**	**81.30**	roberta-large

Table 4: Results on the official DEV set. $_{pub}$ is the result officially published in the reference paper; **lr** is learning rate. *Aggr.* is a shorthand for *Aggregation*.

real-life gold standard-agnostic output of the *DocIR* and *ES* modules.

Note that by construction, we generate one train/test instance per each (C, E_i) pair when training/testing **Local** models and then aggregate the labels predicted for different evidence pieces, i.e the total amount of instances is around number of claims times K[12]. For example, we train **Local** on 722K examples, split into 548K `NEI`, 127K `SUP`, 48K `REF`. When training/testing **Concat**, **WgtSum** and **MaxPool**, we generate one instance per each claim.

4.2 Results

Tab. 4 reports the performance of the systems described in Sec. 3 on the official DEV set.

In previous work, systems employing KGAT ER architecture (Liu et al., 2020; Ye et al., 2020) achieve top performance in terms of FEVER. KGAT ER input data are publicly available enabling us to conduct fair comparison. Lines 1 and 2 report KGAT performance as in (Liu et al., 2020). We integrated their implementation of the KGAT ER component into our pipeline and obtained performance numbers comparable to those published (lines 3, 1). Interestingly, our runs with *roberta-base* outperform the published results of KGAT runs with *roberta-large* (lines 4, 6, 9). We also include its best result (that is an upperbound of KGAT) with *roberta-base* that we obtained with the learning rate of 3e-5. KGAT with *roberta-large* and learning rate of 2e-5 further pushes the performance 1 point up, while the training with the learning rates of 3e-5 and 5e-5 did not converge.

Local models. Lines 10-13 report performance of our local models with two evidence label aggregation heuristics. Heuristic 1 consists in applying the procedure described in Sec. 3 to the labels assigned to all evidence pieces by **Local**. Heuristic 2 is to simply pick the label assigned to the evidence sentence *top-ranked* by ES as in (Zhou et al., 2019). The aggregation heuristic 1 is more competitive.

Global models. Lines 14-22 report performance of the **Concat**, **WgtSum**, **MaxPool** global systems, which all clearly outperform **Local**. Note, that in the **Concat** setting C and E_i, $i = 1, .., K$, are concatenated, thus it is sensitive to the relative E_i order. Overall, all three models perform comparably between each other and to **KGAT** (lines 14-22 vs 5-7). **MaxPool** and **WgtSum** marginally outperform **Concat** with *roberta-large*.

We also trained **Concat** with *roberta-large* setting *K=1* both for training and predicting, i.e., using only top evidence piece retrieved by ES. The resulting LA of 79.57 is only approximately one point behind that of **Concat** (Line 16) and **KGAT** (Line 7). This suggests that good performance can be obtained on the FEVER dataset even without joint reasoning over multiple E_i-s, and that there is still room for further improvement for the systems able to reason upon multiple evidence pieces. Also this could be partially attributed to the observation by Schuster et al. (2019) who showed that FEVER claims contain certain linguistic biases and BERT model fine-tuned on the claim texts only significantly outperforms the majority baseline. Schuster et al. (2019) proposed a debiased symmetric test set, but its instances are claim-evidence pairs. This means that $K = 1$, and thus we did not evaluate our baselines on it as with $K = 1$ they all become equivalent to **Local**.

Comparison to the state of the art. Tab. 5 compares the performance of **MaxPool** and **WgtSum** to that of the SOTA systems as of June 1st, 2021. Our simple baselines outperform all the other systems on DEV, but we may have overfitted on it, as we report the performance of the best checkpoint.

---

[12] It can be less, as for some claims fewer evidence pieces were retrieved.

System	**DEV**		**TEST**	
	FEVER	**LA**	**FEVER**	**LA**
Competition systems				
NSMN (#1) (Nie et al., 2019)[13]	66.59	69.6	64.23	68.16
Post-competition systems				
BERT Pair	68.90	73.30	65.18	69.75
BERT Concat	68.89	73.67	65.64	71.01
GEAR (Zhou et al., 2019)	70.69	74.84	67.19	71.60
DREAM (#2 LA) (Zhong et al., 2020)	n/a	79.16	70.60	*76.85*
* KGAT with				
* - BERT Base	75.88	78.02	69.40	72.81
* - RoBERTa Large (Liu et al., 2020)	76.11	78.29	70.38	74.07
Transformer-XH (Zhao et al., 2020)	74.98	78.05	69.07	72.39
KGAT with				
- CorefBERTBase	n/a	n/a	69.82	72.88
- CorefBERTLarge	n/a	n/a	70.86	74.37
- CorefRoBERTaLarge (Ye et al., 2020)	n/a	n/a	72.30	75.96
DOMLIN++ (Stammbach and Ash, 2020)	74.98	77.48	74.27	76.60
Codalab leaderboard as of June 1, 2021				
#1 dominiks	n/a	n/a	**76.78**	**79.16**
#2 h2oloo	n/a	n/a	75.87	79.35
#3 nudt_nlp	n/a	n/a	74.42	77.38
#4 krishnamrith12	n/a	n/a	74.37	79.25
#5 totopower	n/a	n/a	73.90	77.21
#6 gump	n/a	n/a	73.72	77.05
Our results				
Concat (roberta-large)	78.27	**80.31**	72.59	75.85
MaxPool (roberta-large)	78.85	81.16	72.77	76.55
WgtSum (roberta-large)	**79.02**	81.30	**73.44**	**77.18**

Table 5: FEVER state of the art. We mark results outperforming us with *underscore*. We mark the systems using exactly the same input to the ER component as us with *.

On the blind TEST data, **WgtSum** with *roberta-large* scores seventh in terms of FEVER and sixth in terms of LA on the official Codalab leaderboard.

Despite our best efforts, we were not able to track the publications related to the leaderboard submissions #1-#6. We do not know whether their superior performance is due to a better ER approach, a stronger LRM with billions of parameters, or to a better *DocIR/ES*. In the latter two cases, the baselines in this work still remain relevant.

The best-performing system with *published description*, DOMLIN++, uses *roberta-large* and the **Concat** approach. We cannot compare the results of our ER model directly, since they use a different ES system which might have better evidence recall. Note that we still marginally outperform them in terms of LA. This may indicate that even though our gold evidence recall may be lower due to a possibly less powerful DocIR/ES pipeline (resulting in lower FEVER score), we are still able to predict a correct label given the evidence sentences we have at our disposition. Then, they do additional pre-training on MultiNLI, while we do not exploit any external corpora.

Qualitative analysis We compared the outputs of the **Concat**, **MaxPool**, **WgtSum** and **KGAT** systems. We analyzed 50 DEV set examples where only one out of four systems produced the correct label. We aimed to understand the reason behind the correct prediction, but we have not observed any patterns explaining why one system outperforms the others. The systems seem to be equivalent in their abilities.

When analyzing the **WgtSum** output, we observed that when summing the weighted distributed representations of evidence pieces retrieved by KGAT ES for a specific claim (see Sec. 3), it tends to assign higher weights to the evidence pieces which are correct according to the gold standard.

5 Conclusion

We have proposed lightweight strong baselines for the FEVER fact-checking task and showed that they can outperform heavier models on the official leaderboard with blind TEST set. In our future work, we plan to capitalize from our results to build systems that can effectively trade-off efficiency for accuracy.

Acknowledgments.

We thank the anonymous reviewers for their valuable comments.

References

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke S Zettlemoyer.

2017. AllenNLP: A Deep Semantic Natural Language Processing Platform. *Computing Research Repository*, arXiv:1803.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *Computing Research Repository*, arXiv:1907.

Zhenghao Liu, Chenyan Xiong, and Maosong Sun. 2020. Kernel Graph Attention Network for Fact Verification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online. Association for Computational Linguistics.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining Fact Extraction and Verification with Neural Semantic Matching Networks. *Association for the Advancement of Artificial Intelligence (AAAI)*.

Tal Schuster, Darsh J Shah, Yun Jie Serene Yeo, Daniel Filizzola, Enrico Santus, and Regina Barzilay. 2019. Towards Debiasing Fact Verification Models. In *EMNLP*.

Dominik Stammbach and Elliott Ash. 2020. e-FEVER: Explanations and Summaries for Automated Fact Checking. In *Truth and Trust Online*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The Fact Extraction and VERification (FEVER) Shared Task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential Reasoning Learning for Language Representation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7170–7186, Online. Association for Computational Linguistics.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-XH: Multi-evidence Reasoning with Extra Hop Attention. In *The Eighth International Conference on Learning Representations (ICLR 2020)*.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning Over Semantic-Level Graph for Fact Checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online. Association for Computational Linguistics.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 892–901, Florence, Italy. Association for Computational Linguistics.

A Learning rate selection.

When experimenting with *roberta-base* we tried learning rates (lr) [1e-5; 5e-5] with the step of 1e-5. Table 6 summarizes the results. The results obtained with the learning rates in range [2e-5; 5e-5] are very similar, so we used the learning rate of 2e-5 in the majority of our experiments in this paper.

B Model complexity.

Tab. 7 reports the amount of trainable parameters in the ER component of each model when run on

Model	FEVER	LA	LR
Concat	76.40	78.60	1e-05
	77.09	**79.25**	2e-05
	76.97	79.14	3e-05
	76.81	79.00	4e-05
	76.76	78.95	5e-05
KGAT	77.39	79.74	1e-05
	77.66	79.98	2e-05
	77.75	**80.06**	3e-05
	77.53	79.79	4e-05
	75.80	77.93	5e-05
MaxPool	76.97	79.40	1e-05
	77.48	79.82	2e-05
	77.58	79.88	3e-05
	77.61	79.95	4e-05
	77.67	**79.98**	5e-05

Table 6: Experimenting with different learning rates with *roberta-base* as LRM.

Model	**# parameters-base**	**# parameters-large**
	LRM parameters	
LRM (RoBERTa)	124,645,632	355,359,744
	Parameters in the joint inference ER component	
KGAT	792,112	1,318,192
Concat/MaxPool	2,307	3,075
WgtSum	3,076	4,100

Table 7: Number of trainable parameters in the ER models with RoBERTa LRM. We report the amount of LRM and ER component parameters separately (i.e. the full ER model size is their sum). *-base*/*-large* refers to the LRM version.

top of different LRMs[14]. Our simple baselines perform comparably to SOTA using an ER inference component having 3K parameters only (in addition to LRM parameters).

---

[14] Naturally, we update the LRMs parameters as well.

Sketch and Refine: Towards Faithful and Informative Table-to-Text Generation

Peng Wang , Junyang Lin , An Yang ,
Chang Zhou , Yichang Zhang , Jingren Zhou , Hongxia Yang[†]

DAMO Academy, Alibaba Group
`{zheluo.wp, junyang.ljy, ya235025, ericzhou.zc, yichang.zyc, jingren.zhou, yang.yhx }@alibaba-inc.com`

Abstract

Table-to-text generation refers to generating a descriptive text from a key-value table. Traditional autoregressive methods, though can generate text with high fluency, suffer from *low coverage* and *poor faithfulness* problems. To mitigate these problems, we propose a novel **S**keleton-based two-stage method that combines both **A**utoregressive and **N**on-**A**utoregressive generation (**SANA**). Our approach includes: (1) skeleton generation with an autoregressive pointer network to select key tokens from the source table; (2) edit-based non-autoregressive generation model to produce texts via iterative insertion and deletion operations. By integrating hard constraints from the skeleton, the non-autoregressive model improves the generation's coverage over the source table and thus enhances its faithfulness. We conduct experiments on both the WikiPerson and WikiBio datasets. Experimental results demonstrate that our method outperforms the previous state-of-the-art methods in both automatic and human evaluation, especially on coverage and faithfulness. In particular, we achieve PARENT-T recall of 99.47 in WikiPerson, improving over the existing best results by more than 10 points.

1 Introduction

Table-to-text generation is a challenging task which aims at generating a descriptive text from a key-value table. There have been a broad range of applications in this field, such as the generation of weather forecast (Mei et al., 2016), sports news (Wiseman et al., 2017), biography (Lebret et al., 2016; Wang et al., 2018), etc. Figure 1 illustrates a typical input and output example of this task.

Previous methods (Liu et al., 2018; Nie et al., 2018; Bao et al., 2018) are usually trained in an end-to-end fashion with the encoder-decoder architecture (Bahdanau et al., 2015). Despite generating text with high fluency, their lack of control in the generation process leads to *poor faithfulness* and *low coverage*. As shown in Figure 1, the case of *poor faithfulness* hallucinates the occupation "singer" not entailed by the source table, and the case of *low coverage* misses the information of the place of birth. Even if trained with a cleaned dataset, end-to-end methods still encounter these problems as it is too complicated to learn the probability distribution under the table constraints (Parikh et al., 2020).

key	value
Name_ID	Thaila Ayala
Place of birth	Brazil
Date of birth	April 14 1986
Occupation	Actress , Model

Poor Faithfulness: Thaila Ayala (born April 14, 1986 in Brazil) is an actress , model and singer .

Low Coverage: Thaila Ayala (born April 14, 1986) is an actress and model .

Figure 1: An example of table-to-text generation. The case of *poor faithfulness* hallucinates content not entailed by the table (marked in red color). The case of *low coverage* misses the information of the person's birth place (marked in blue color).

To alleviate these problems, recent studies (Shao et al., 2019; Puduppully et al., 2019; Ma et al., 2019) propose two-stage methods to control the generation process. In the first stage, a pointer network selects the salient key-value pairs from the table and arranges them to form a content-plan. In the second stage, an autoregressive seq2seq model generates text conditioned on the content-plan. However, such methods can cause the following problems: (1) since the generated content-plan may contain errors, generating solely on the content-

---

[†]Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4831–4843
August 1–6, 2021. ©2021 Association for Computational Linguistics

plan leads to inconsistencies; (2) even if a perfect content-plan is provided, the autoregressive model used in the second stage is still prone to hallucinate unfaithful contents due to the well-known *exposure bias* (Wang and Sennrich, 2020) problem; (3) there is no guarantee that the selected key-value pairs can be described in the generated text. As a result, these methods still struggle to generate faithful and informative text.

In this paper, we propose a **S**keleton-based model that combines both **A**utoregressive and **N**on-**A**utoregressive generation (**SANA**). SANA divides table-to-text generation into two stages: *skeleton construction* and *surface realization*. At the stage of skeleton construction, an autoregressive pointer network selects tokens from the source table and composes them into a skeleton. We treat the skeleton as part of the final generated text. At the stage of surface realization, an edit-based non-autoregressive model expands the skeleton to a complete text via insertion and deletion operations. Compared with the autoregressive model, the edit-based model has the following advantages: (1) the model generates text conditioned on both the skeleton and the source table to alleviate the impact of incomplete skeleton; (2) the model accepts the skeleton as decoder input to strengthen the consistency between the source table and generated text; (3) the model generates texts with the hard constraints from the skeleton to improve the generation coverage over the source table. Therefore, SANA is capable of generating faithful and informative text.

The contributions of this work are as follows:

- We propose a skeleton based model **SANA** which explicitly models skeleton construction and surface realization. The separated stages helps the model better learn the correlation between the source table and reference.

- To make full use of the generated skeleton, we use a non-autoregressive model to generate text based on the skeleton. To the best of our knowledge, we are the first to introduce non-autoregressive model to table-to-text generation task.

- We conduct experiments on WikiPerson and WikiBio datasets. Both automatic and human evaluations show that our method outperforms previous state-of-the-art methods, especially on faithfulness and coverage. Specially, we obtain a near-optimal PARENT-T recall of 99.47 in the WikiPerson dataset.

2 Related Work

Table-to-text Generation Table-to-text generation has been widely studied for decades (Kukich, 1983; Goldberg et al., 1994; Reiter and Dale, 1997). Recent works that adopt end-to-end neural networks have achieve great success on this task (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Sha et al., 2018; Nema et al., 2018; Liu et al., 2018, 2019a). Despite generating fluent texts, these methods suffer from poor faithfulness and low coverage problems. Some works focus on generating faithful texts. For example, Tian et al. (2019) proposes a confident decoding technique that assigns a confidence score to each output token to control the decoding process. Filippova (2020) introduces a "hallucination knob" to reduce the amount of hallucinations in the generated text. However, these methods only focus on the faithfulness of the generated text, they struggle to cover most of the attributes in the source table.

Our work is inspired by the recently proposed two-stage method (Shao et al., 2019; Puduppully et al., 2019; Moryossef et al., 2019; Ma et al., 2019; Trisedya et al., 2020). They shows that table-to-text generation can benefit from separating the task into content planing and surface realization stages. Compared with these methods, SANA guarantee the information provided by the first stage can be preserved in the generated text, thus significantly improving the the coverage as well as the faithfulness of the generated text.

Non-autoregressive Generation Although autoregressive models have achieved remarkable success in natural language generation tasks, they are time-consuming and inflexible. To overcome these shortcomings, Gu et al. (2018) proposed the first non-autoregressive (NAR) model that can generate tokens simultaneously by discarding the generation history. However, since a source sequence may have different possible outputs, discarding the dependency of target tokens may cause the degradation in generation quality. This problem also known as the "multi-modality" problem (Gu et al., 2018). Recent NAR approaches alleviate this problem via partially parallel decoding (Stern et al., 2019; Sun et al., 2019) or iterative refinement (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019). Specially, Stern et al. (2019) performs partially parallel decoding through insertion operation. Gu et al. (2019) further incorporates deletion operation to perform iterative refinement process. These edit-based models not only close the gap with autoregressive models in translation task, but also makes generation flexible by allowing integrates with lexical constrains. However, the multi-modality problem still exists, making it difficult to apply NAR models to other generation tasks, such as table-to-text generation, story generation, etc. In this work, we use the skeleton as the initial input of our edit-based text generator. The skeleton can provide sufficient contexts to the text generator, thus significantly reducing the impact of multi-modality problem.

3 Methods

The task of table-to-text generation is to take a structured table T as input, and outputs a descriptive text $Y = \{y_1, y_2, ..., y_n\}$. Here, the table T can be formulated as a set of attributes $T = \{a_1, a_2, ..., a_m\}$, where each attribute is a key-value pair $a_i = \langle k_i, v_i \rangle$.

Figure 2 shows the overall framework of SANA. It contains two stages: skeleton construction and surface realization. At the stage of skeleton construction, we propose a Transformer-based (Vaswani et al., 2017) pointer network to select tokens from the table and compose them into a skeleton. At the stage of surface realization, we use an edit-based Transformer to expand the skeleton to a complete text via iterative insertion and deletion operations.

3.1 Stage 1: Skeleton Construction

3.1.1 Table Encoder

The source table is a set of attributes represented as key-value pairs $a_i = \langle k_i, v_i \rangle$. Here, the value of an attribute a_i is flattened as a token sequence $v_i = \{w_i^1, w_i^2, ..., w_i^l\}$, where w_i^j is the j-th token and l is the length of v_i. Following Lebret et al. (2016), we linearize the source table by representing each token w_i^j as a 4-tuple $(w_i^j, k_i, p_i^+, p_i^-)$, where p_i^+ and p_i^- are the positions of the token w_i^j counted from the beginning and the end of the value v_i, respectively. For example, the attribute of "⟨*Name_ID*, {*Thaila Ayala*}⟩" is represented as "(*Thaila*, *Name_ID*, 1, 2)" and "(*Ayala*, *Name_ID*, 2, 1)". In order to make the pointer network capable of selecting the special token $\langle EOS \rangle$[1], we add a special tuple $(\langle EOS \rangle, \langle EOS \rangle, 1, 1)$ at the end of the table.

To encode the source table, we first use a linear projection on the concatenation $\left[\boldsymbol{w}_i^j; \boldsymbol{k}_i; \boldsymbol{p}_i^+; \boldsymbol{p}_i^-\right]$ followed by an activation function:

$$\boldsymbol{f}_i^j = \mathrm{Relu}(\boldsymbol{W}_f[\boldsymbol{w}_i^j; \boldsymbol{k}_i; \boldsymbol{p}_{\boldsymbol{i}}^+; \boldsymbol{p}_{\boldsymbol{i}}^-] + \boldsymbol{b}_f) \quad (1)$$

where $\boldsymbol{W}_f$ and $\boldsymbol{b}_f$ are trainable parameters. Then we use the Transformer encoder to transform each $\boldsymbol{f}_i^j$ into a hidden vector and flatten the source table into a vector sequence $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_l\}$.

3.1.2 Pointer Network

After encoding the source table, we use a pointer network to directly select tokens from the table and compose them into a skeleton. Our pointer network uses a standard Transformer decoder to represent the tokens selected at the previous steps. Let $\boldsymbol{r}_t$ denote the decoder hidden state of previous selected token $\hat{y}_t$. The pointer network predict the next token based on the attention scores, which are computed as follows:

$$\alpha_{ti} = \frac{e^{u(\boldsymbol{r}_t, \boldsymbol{h}_i)}}{\sum_{j=1}^{l} e^{u(\boldsymbol{r}_t, \boldsymbol{h}_j)}} \quad (2)$$

$$u(\boldsymbol{r}_t, \boldsymbol{h}_i) = \frac{(\boldsymbol{W}_q \boldsymbol{r}_t) \cdot (\boldsymbol{W}_k \boldsymbol{h}_i)}{\sqrt{d_r}} \quad (3)$$

where $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ are trainable parameters, d_r is the embedding dimension of $\boldsymbol{r}_t$. According to the calculated probability distribution $\boldsymbol{\alpha}$, we select token based on the following formula:

$$P_{copy}(w) = \sum_{w=w_i} \alpha_{ti} \quad (4)$$

$$\hat{y}_{t+1} = \arg\max_{w} P_{copy}(w) \quad (5)$$

where $\hat{y}_{t+1}$ represents the output at the next timestep, and $P_{copy}(w)$ represents the probability of copying token w from the source. There may be multiple identical tokens in the table, so we sum up the attention scores of their corresponding positions.

The pointer network needs target skeletons as supervision, which are not provided by the table-to-text datasets. In this paper, we obtain the skeleton by collecting tokens in both the table and description except the stop words. The token order in the skeleton remains the same as their relative positions in the description. More details are described in Appendix A.

---

[1] $\langle EOS \rangle$ denotes the end of the skeleton.

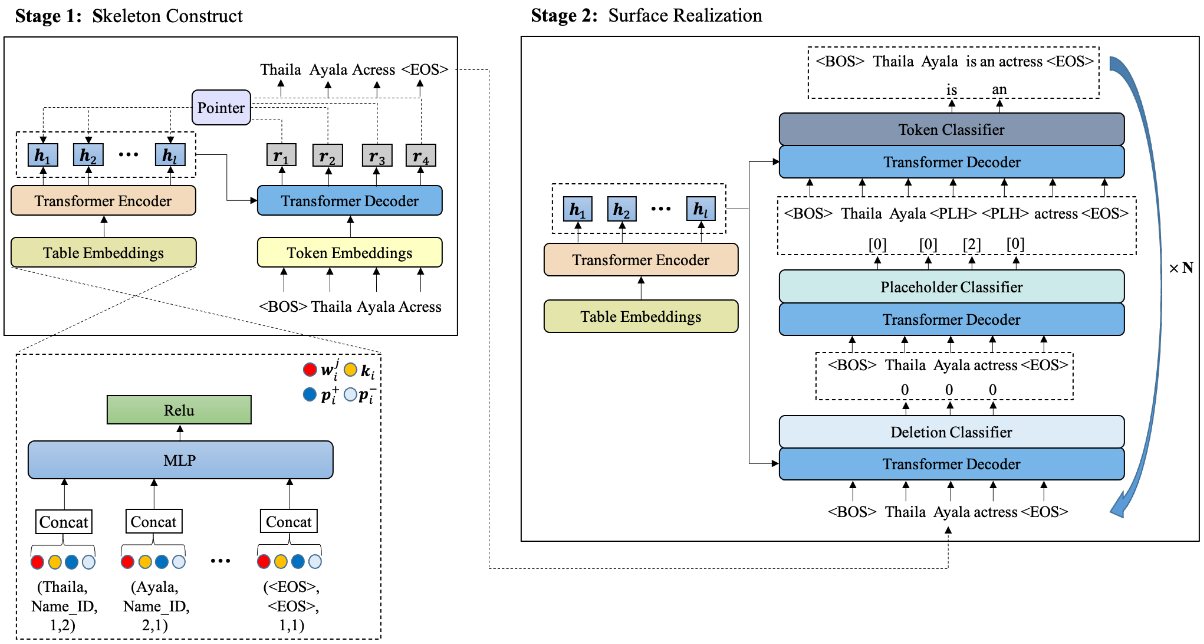

Figure 2: The overall diagram of SANA for generating description for *Thaila Ayala* in Fig 1.

Given the skeleton $S = \{s_1, s_2, ..., s_q\}$, the pointer network is trained to maximize the conditional log-likelihood:

$$\mathcal{L}_1 = -\sum_{t=1}^{q+1} \log P_{copy}(s_t | s_{0:t-1}, T), \quad (6)$$

where the special tokens $s_0 = \langle BOS \rangle$ and $s_{q+1} = \langle EOS \rangle$ denote the beginning and end of the target skeleton.

3.2 Stage 2: Surface Realization

At the surface realization stage, we use the same encoder as in the skeleton construction stage. The decoder is an edit-based Transformer decoder (Gu et al., 2019) that generates text via insertion and deletion operations. Different from the original Transformer decoder which predicts the next token in an left-to-right manner, the edit-based decoder can predict tokens simultaneously and independently. In this setting, we can use the full self-attention without causal masking.

3.2.1 Model Structure

To perform the insertion and deletion operations, we remove the softmax layer at the top of the Transformer decoder and add three operation classifiers: *Deletion Classifier*, *Placeholder Classifier* and *Token Classifier*. We denote the outputs of the Transformer decoder as $(\boldsymbol{z}_0, \boldsymbol{z}_1, ..., \boldsymbol{z}_n)$, details of these three classifiers are as follows:

1. *Deletion Classifier* which predicts for each token whether they should be "deleted"(1) or "kept"(0):

$$\pi_\theta^{\text{del}}(d|i, Y) = \text{softmax}(\boldsymbol{W}_{\text{del}} \boldsymbol{z}_i) \quad (7)$$

2. *Placeholder Classifier* which predicts the number of placeholders [PLH] to be inserted at each consecutive pair:

$$\pi_\theta^{\text{plh}}(p|i, Y) = \text{softmax}(\boldsymbol{W}_{\text{plh}}[\boldsymbol{z}_i; \boldsymbol{z}_{i+1}]) \quad (8)$$

3. *Token Classifier* which replaces each [PLH] with an actual token:

$$\pi_\theta^{\text{tok}}(t|i, Y) = \text{softmax}(\boldsymbol{W}_{\text{tok}} \boldsymbol{z}_i) \quad (9)$$

During decoding, we use the skeleton predicted from the first stage as the initial input of the decoder. We also use the full table information from encoder side to mitigate the impact of incomplete skeleton. As shown in Figure 2, the skeleton will pass through the three classifiers sequentially for several iterations. Benefiting from the full self-attention, each operation is allowed to condition on the entire skeleton, and thus reduces the probability of hallucinating unfaithful contents in the final text.

3.2.2 Training

Following Gu et al. (2019), we adopt imitation learning to train our model and simplify their training procedure. The iterative process of our model will produce various of *intermediate sequences*. To simulate the iterative process, we need to construct the intermediate sequence and provide an optimal operation $\boldsymbol{a}^*$ (either oracle insertion $\boldsymbol{p}^*$, $\boldsymbol{t}^*$ or oracle deletion $\boldsymbol{d}^*$) as the supervision signal during training. Given an intermediate sequence Y, the optimal operation $\boldsymbol{a}^*$ is computed as follows:

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} \mathcal{D}(Y^*, \mathcal{E}(Y, \boldsymbol{a})) \qquad (10)$$

Here, $\mathcal{D}$ denotes the Levenshtein distance (Levenshtein, 1965) between two sequences, and $\mathcal{E}(Y, \boldsymbol{a})$ represents the output after performing operation $\boldsymbol{a}$ upon Y.

To improve the training efficiency, we construct the intermediate sequence via a simple yet effective way. Given a source table, skeleton and reference (T, S, Y^*), We first calculate the longest common subsequence X between S and Y^*, and then construct the intermediate sequence Y' by applying random deletion on Y^* except the part of X. We use Y' to learn the insertion and deletion operations. The learning objective of our model is computed as follows:

$$\mathcal{L}_{\text{edit}} = \mathcal{L}_{\text{ins}} + \lambda \mathcal{L}_{\text{del}} \qquad (11)$$

$$\begin{aligned} \mathcal{L}_{\text{ins}} = &- \sum_{p_i^* \in \boldsymbol{p}^*} \log \pi_\theta^{\text{plh}}(p_i^* | i, T, Y') \\ &- \sum_{t_i^* \in \boldsymbol{t}^*} \log \pi_\theta^{\text{tok}}(t_i^* | i, T, Y'') \end{aligned} \qquad (12)$$

$$\mathcal{L}_{\text{del}} = - \sum_{d_i^* \in \boldsymbol{d}^*} \log \pi_\theta^{\text{del}}(d_i^* | i, T, Y''') \qquad (13)$$

where Y'' is the output after inserting placeholders $\boldsymbol{p}*$ upon Y', Y''' is the output by applying the model's insertion policy π_θ^{tok} to Y''.[2] λ is the hyper parameter.[3]

3.2.3 Inference

As mentioned above, at the inference stage, we use the generated skeleton as the initial input of the decoder. The insertion and deletion operations will perform alternately for several iterations. We stop the decoding process when the current text does not change, or a maximum number of iterations has been reached.

In order to completely retain the skeleton in the generated text, we follow Susanto et al. (2020) to enforce hard constraints through forbidding the deletion operation on tokens in the skeleton. Specially, we compute a *constraint mask* to indicate the positions of constraint tokens in the sequence and forcefully set the deletion classifier prediction for these positions to "keep". The constraint masks are recomputed after each insertion and deletion operation.

4 Experiment Setups

4.1 Datasets

We conduct experiments on the **WikiBio** (Lebret et al., 2016) and **WikiPerson** (Wang et al., 2018) datasets. Both datasets aim to generate a biography from a Wikipedia table, but they have different characteristics. Their basic statistics are listed in Table 1.

Dataset	Train	Valid	Test	Avg Len
WikiBio	582,657	72,831	72,831	26.1
WikiPerson	250,186	30,487	29,982	70.6

Table 1: Statistics of WikiBio and WikiPerson datasets. **Avg Len** means the average target length of the datasets.

WikiBio This dataset aims to generate the first sentence of a biography from a table. It is a particularly noisy dataset which has 62% of the references containing extra information not entailed by the source table (Dhingra et al., 2019).

WikiPerson Different from the WikiBio whose reference only contains one sentence, the reference of WikiPerson contains multiple sentences to cover as many facts encoded in the source table as possible. In addition, WikiPerson uses heuristic rules to remove sentences containing entities that do not exist in the Wikipedia table, making it cleaner compared to the WikiBio dataset.

4.2 Evaluation Metrics

Automatic Metrics For automatic evaluation, we apply BLEU (Papineni et al., 2002) as well as PARENT (precision, recall, F1) (Dhingra et al., 2019) to evaluate our method. Different from BLEU which only compare the outputs with the references, PARENT evaluates the outputs that considers both the references and source tables. Following Wang et al. (2020), we further use their proposed PARENT-T metric to evaluate our method in WikiPerson dataset. PARENT-T is a variant of

---

[2] We do argmax from Equation (9) instead of sampling.

[3] In our experiment, $\lambda = 1$ gives a reasonable good result.

PARENT which only considers the correlation between the source tables and the outputs.

Human Evaluation Human ratings on the generated descriptions provide more reliable reflection of the model performance. Following Liu et al. (2019b), we conduct comprehensive human evaluation between our model and the baselines. The annotators are asked to evaluate from three perspectives: fluency, coverage (how much table content is recovered) and correctness (how much generated content is faithful to the source table). We hire 5 experienced human annotators with linguistic background. During the evaluation, 100 samples are randomly picked from the WikiPerson dataset. For each sample, an annotator is asked to score the descriptions generated by different models without knowing which model the given description is from. The scores are within the range of $[0, 4]$.

4.3 Implementation Details

We implement SANA using fairseq (Ott et al., 2019). The token vocabulary is limited to the 50K most common tokens in the training dataset. The dimensions of token embedding, key embedding, position embedding are set to 420, 80, 5 respectively. All Transformer components used in our methods adopt the base Transformer (Vaswani et al., 2017) setting with $d_{\text{model}} = 512, d_{\text{hidden}} = 2048, n_{\text{heads}} = 8, n_{\text{layers}} = 6$. All models are trained on 8 NVIDIA V100 Tensor Core GPUs.

For the skeleton construction model, the learning rate linearly warms up to 3e-4 within 4K steps, and then decays with the inverse square root scheduler. Training stops after 15 checkpoints without improvement according to the BLEU score. We set the beam size to 5 during inference.

For the surface realization model, the learning rate linearly warms up to 5e-4 within 10K steps, and then decays with the inverse square root scheduler. Training stops when the training steps reach 300K. We select the best checkpoint according to the validation BLEU.

4.4 Baselines

We compare SANA with two types of methods: end-to-end methods and two-stage methods.

For end-to-end methods, we select the following methods as baselines: (1) **DesKnow** (Wang et al., 2018), a seq2seq model with a table position self-attention to capture the inter-dependencies among related attributes; (2) **PtGen** (Pointer-Generator, See et al. (2017)), an LSTM-based seq2seq model with attention and copy mechanism; (3) **Struct-Aware** (Liu et al., 2018), a seq2seq model using a dual attention mechanism to consider both key and value information; (4) **OptimTrans** (Wang et al., 2020), a Transformer based model that incorporates optimal transport matching loss and embedding similarity loss. (5) **Conf-PtGen** (Tian et al., 2019), a pointer generator with a confidence decoding technique to improve generation faithfulness; (6) **S2S+FA+RL** (Liu et al., 2019b), a seq2seq model with a force attention mechanism and a reinforce learning training procedure; (7) **Bert-to-Bert** (Rothe et al., 2019), a Transformer encoder-decoder model where the encoder and decoder are both initialized with BERT (Devlin et al., 2019).

For two-stage methods, we select the following methods as baselines: (1) **Pivot** (Ma et al., 2019), a two stage method that first filter noisy attributes in the table via sequence labeling and then uses the Transformer to generate text based on the filter table; (2) **Content-Plan** (Puduppully et al., 2019), a two stage method that first uses a pointer network to select important attributes to form a content-plan and then uses a pointer generator to generate text based on the content-plan.

5 Results

5.1 Comparison with End-to-End Methods

We first compare SANA with end-to-end methods, Table 2 shows the experimental results. From Table 2, we can outline the following statements: (1) For WikiPerson dataset, SANA outperforms existing end-to-end methods in all of the automatic evaluation metrics, indicating high quality of the generated texts. Specially, we obtain a near-optimal PARENT-T recall of 99.47, which shows that our model has the ability to cover all the contents of the table. (2) For the noisy WikiBio dataset, SANA outperforms previous state-of-the-art models in almost all of the automatic evaluation scores except the PARENT precision, which confirms the robustness of our method. Although Conf-PtGen achieves the highest PARENT precision, its PARENT recall is significantly lower than any other method. Different from Conf-PGen, SANA achieves the highest recall while maintaining good precision. (3) It is necessary to prohibit deleting tokens in the skeleton. After removing this restriction (− *hard constrains*), our method has different degrees of decline in various automatic metrics. (4) SANA

Model	**WikiPerson**			**WikiBio**	
	BLEU	**PARENT(P / R / F1)**	**PARENT-T(P / R / F1)**	**BLEU**	**PARENT(P / R / F1)**
DesKnow	16.20	63.92 / 44.83 / 51.03	41.10 / 84.34 / 54.22	-	- / - / -
PtGen	19.32	61.73 / 44.09 / 49.52	42.03 / 81.65 / 52.62	41.07	77.59 / 42.12 / 52.10
Struct-Aware	22.76	51.18 / 46.34 / 46.47	35.99 / 83.84 / 48.47	44.93	74.18 / 43.50 / 52.33
OptimTrans	24.56	62.86 / 48.83 / 53.06	43.52 / 85.21 / 56.10	-	- / - / -
Conf-PtGen	-	- / - / -	- / - / -	38.10	**79.52** / 40.60 / 51.38
S2S+FA+RL	-	- / - / -	- / - / -	45.49	76.10 / 43.66 / 53.08
Bert-to-Bert	-	- / - / -	- / - / -	45.62	77.64 / 43.42 / 53.54
SANA	**25.23**	**65.69 / 56.88 / 59.96**	**44.88 / 99.47 / 61.34**	**45.78**	76.93 / **46.01 / 55.42**
− *hard constrains*	24.97	64.72 / 56.42 / 59.29	43.75 / 98.97 / 60.17	45.31	76.32 / 45.26 / 54.64
− *skeleton*	19.55	61.80 / 44.29 / 50.29	40.80 / 84.03 / 53.97	42.58	74.29 / 41.32 / 50.41

Table 2: Comparison with end-to-end methods. **P, R, F1** represent precision, recall and F1 score, respectively. "− *hard constrains*" means removing the restriction of forbidding the deletion operation on tokens in the skeleton, "− *skeleton*" means removing the skeleton construction stage.

Model	**WikiPerson**			**WikiBio**	
	BLEU	**PARENT(P / R / F1)**	**PARENT-T(P / R / F1)**	**BLEU**	**PARENT(P / R / F1)**
Pivot	24.71	62.24 / 50.02 / 52.99	41.78 / 89.68 / 56.35	44.39	76.35 / 41.90 / 51.85
+ *Oracle*	25.08	62.34 / 50.63 / 53.47	42.08 / 89.71 / 56.59	45.38	75.98 / 42.57 / 52.45
Content-Plan	25.07	58.56 / 53.86 / 54.52	38.63 / 91.18 / 54.01	43.21	74.69 / 43.53 / 52.71
+ *Oracle*	28.50	59.31 / 56.02 / 55.96	39.64 / 91.62 / 55.07	50.57	76.32 / 47.33 / 56.45
SANA	25.23	65.69 / 56.88 / 59.96	44.88 / 99.47 / 61.34	45.78	76.93 / 46.01 / 55.42
+ *Oracle*	**30.29**	**69.27 / 67.89 / 68.28**	**45.13 / 99.79 / 61.54**	**54.51**	**80.03 / 51.02 / 61.01**

Table 3: Comparison with two-stage methods. **P, R, F1** represent precision, recall and F1, respectively. "+ *Oracle*" means using oracle information (i.e., oracle skeleton or content-plan) as input.

performs poorly after removing the skeleton construction stage (− *skeleton*). This shows that the edit-based non-autoregressive model is difficult to directly apply to table-to-text generation tasks. The skeleton is very important for the edit-based model, which can significantly reduce the impact of the multi-modality problem. Combining both autoregressive and non-autoregressive generations, SANA achieves state-of-the-art performance.

5.2 Comparison with Two-Stage Methods

We further compare SANA with the two-stage methods. As shown in Table 3, there is an obvious margin between SANA and the two baselines, which shows that SANA can more effectively model the two-stage process. In order to prove that SANA can make use of the information provided by the first stage, we use the gold standard (i.e., the oracle skeleton or content-plan extracted from heuristics methods) as the input of the models used in the second stage. With this setup, SANA has made significant improvements in multiple automatic metrics while other methods have limited improvements. Specially, the improvements of Pivot are limited because its gold standard does not model the order of the attributes. Although the first stage of Content-Plan is similar to SANA, its PARENT scores (either precision, recall and F1) has not been obvious improved, especially on WikiPerson dataset. This shows that the edit-based decoder of SANA can make use of the oracle skeleton to produce high quality descriptions.

Model	**Fluency ↑**	**Coverage ↑**	**Correctness ↑**
Pivot	3.40	3.58	2.89
Content-Plan	3.39	3.70	2.98
Struct-Aware	3.31	3.60	2.94
DesKnow	3.45	3.42	3.07
SANA	**3.46**	**3.72**	**3.11**

Table 4: Human evaluation on WikiPerson for SANA and baselines. The scores (higher is better) are based on fluency, coverage and correctness, respectively.

5.3 Human Evaluation

We report the human evaluation result on the WikiPerson dataset in Table 4. From the demonstrated results, it can be found that SANA outperforms the other end-to-end or two-stage models on all the human evaluation metrics. This is consistent with our model's performance in the automatic evaluation. In the evaluation of fluency, though the

(1) Name_ID: William Edward Ayrton **(2) Place_of_burial**: Brompton Cemetery **(3) Place_of_birth**: London **(4) Educated_at**: University College London **(5) Date_of_birth**: 14 September 1847 **(6) Occupation**: Physicist **(7) Date_of_death**: 8 November 1908 **(8) Aware_received**: Fellow of the Royal Society **(9) Child**: Barbara Ayarton-Gould

Model	Miss	Output
DesKnow	**(9)**	William Edward Ayrton Fellow of the Royal Society (14 September 1847 – 8 November 1908) was a British Physicist . Brompton Cemetery he was born in London the son of Sir Thomas and his wife Mary (nee Fleming) . he was educated at University College School and University College London .
Struct-Aware	**(1) (2) (9)**	William Edward Keeler Fellow of the Royal Society (14 September 1847 – 8 November 1908) was a British Physicist and Physicist . he was elected a Fellow of the Royal Society in 1889 and was a member of the Royal Society of London and the Royal Society of London and the Royal Society of London . he was educated at the University College London and at the University College London where he was a pupil of the chemist William.
OptimTrans	**None**	William Edward Ayrton Fellow of the Royal Society (14 September 1847 – 8 November 1908) was an English Physicist . William was born in London and educated at University College London. he is buried in Brompton Cemetery London . he was elected a Fellow of the Royal Society in 1901. he was the father of Barbara Ayrton-Gould .
SANA	**None**	William Edward Ayrton Fellow of the Royal Society (14 September 1847 – 8 November 1908) was an English Physicist . he is buried in Brompton Cemetery London . he studied physics at University College London . Ayrton was born in London . he was the father of Barbara Ayrton-Gould.

Table 5: Example outputs from different methods. The red text stands for the hallucinated content in each generated description. Given the table, all the models **except SANA** generate unfaithful content to varying degrees. Meanwhile, both DesKnow and Struct-Aware miss some table facts, while SANA recovers them all.

models except for Struct-Aware reach similar performances, SANA performs the best, which demonstrates that its generation has fewer grammatical and semantic mistakes. In the evaluation coverage, SANA outperforms the Content-Plan model and defeats the other models by a large margin. This result is consistent with our proposal that SANA can cover sufficient information in the source table, and it can ensure the informativeness of generation. As to correctness, the advantage of SANA over the other models indicates that our model generates more faithful content and suffers less from the hallucination problem. It should be noted that although Content-Plan and DesKnow are on par with SANA on coverage and correctness respectively, they fail to perform well on both metrics in contrast with SANA. This indicates that our model generates both informative and faithful content.

5.4 Case Study

Table 5 shows the descriptions generated by different methods from the test set of WikiPerson.[4] DesKnow and Struct-Aware miss some attributes and hallucinate unfaithful contents (marked in red). Although OptimTrans achieves better coverage, it hallucinates the unfaithful content "in 1901" not entailed by the table. Compared to these methods, our method can cover all the attributes in the table and does not introduce any unfaithful contents. In addition, the generation length of SANA is shorter than Struct-Aware and OptimTrans, which shows that SANA can use more concise text to cover the facts of the table. These results indicate our method is capable of generating faithful and informative text. We put more generation examples in Appendix B.

6 Conclusion

In this paper, we focus on faithful and informative table-to-text generation. To this end, we propose a novel skeleton-based method that combines both autoregressive and non-autoregressive generations. The method divides table-to-text generation into skeleton construction and surface realization stages. The separated stages helps model better learn the correlation between the source table and reference. In the surface realization stage, we further introduce an edit-based non-autoregressive model to make full use of the skeleton. We conduct experiments on the WikiBio and WikiPerson datasets. Both automatic and human evaluations demonstrate the effectiveness of our method, especially on faithfulness and coverage.

---

[4] For fair comparison, we use the generation examples of baselines provided by Wang et al. (2020)

Acknowledgements

We thank Tianyu Liu for his suggestions on this research and his providing of inference results of the baseline models. We also thank Yunli Wang for the insightful discussion.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 864–870.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

V. I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019a. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6786–6793.

Tianyu Liu, Fuli Luo, Pengcheng Yang, Wei Wu, Baobao Chang, and Zhifang Sui. 2019b. Towards comprehensive description generation from factual attribute-value tables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5985–5996.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shuming Ma, Pengcheng Yang, Tianyu Liu, Peng Li, Jie Zhou, and Xu Sun. 2019. Key fact as pivot: A two-stage model for low resource table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2047–2057.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1539–1550.

Feng Nie, Jinpeng Wang, Jin-ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Sascha Rothe, Shashi Narayan, and A. Severyn. 2019. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, et al. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3248–3259.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 5976–5985. PMLR.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. *Advances in Neural Information Processing Systems*, 32:3016–3026.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with levenshtein transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3536–3543.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Bayu Trisedya, Jianzhong Qi, and Rui Zhang. 2020. Sentence generation for entity description with content-plan attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9057–9064.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

A Automatic Skeleton Annotation

Algorithm 1 describes the automatic skeleton annotation process. Given a table and its corresponding description, we first collect tokens appearing in both the table and description except the stop words, then these tokens are sorted based on their positions in the description in ascending order. In this way, we can obtain a sequence composed of the selected tokens. We regard this sequence as a skeleton.

B More Generation examples

We further provide a case study, using another two examples (including a very challenging example which needs to recover a large number of facts), to show the effectiveness of our method SANA. In the following pages, we show the example outputs in Table 6 and 7. In these examples, the SANA model shows much better capability of generating informative and faithful descriptions compared with the baselines.

Algorithm 1 Automatic Skeleton Annotation

```
Input: A stop word set $W$, a parallel dataset $D = \{(T_1, Y_1^*), (T_2, Y_2^*), ..., (T_{|D|}, Y_{|D|}^*)\}$;
Output: A skeleton list $S$;
Initial the skeleton list $S = []$
for $(T_i, Y_i^*) \in D$ do
    $T_i = ((k_1, v_1), (k_2, v_2), ..., (k_m, v_m))$
    $V_i = (v_1, v_2, ..., v_m)$
    $Y_i^* = (y_1^*, y_2^*, ..., y_n^*)$
    Initialize the skeleton list $S_i = []$
    for $y_j^* \in Y_i^*$ do
        if $y_j^* \in V_i$ and $y_j^* \notin W$ then
            Append token $y_j^*$ to the skeleton list $S_i$
        end if
    end for
    collect the skeleton list $S$ += $S_i$
end for
```

(1) Name_ID: Aaron Miller **(2) Member_of_sports_team**: Colorado Avalanche **(3) Member_of_sports_team**: Quebec Nordiques **(4) Date_of_birth**: August 11 1971 **(5) Place_of_birth**: Buffalo, New York **(6) Country_of_citizenship**: United States **(7) Participant_of**: 2006 Winter Olympics **(8) Participant_of**: 2002 Winter Olympics **(9) Sport**: Ice hockey
(10) Position_played_on_team / speciality: Cornerback

Model	Miss	Output
DesKnow	**7, 8**	Aaron Miller (born August 11 1971) is an United States former professional Ice hockey Defenceman who played in the National Hockey League (NHL) for the Quebec Nordiques and the Colorado Avalanche . he was born in Buffalo, New York and played for the Quebec Nordiques and the Ottawa Senators .
Struct-Aware	**7, 8**	Aaron Miller (born August 11 1971 in Buffalo, New York New York) is a retired United States professional Ice hockey Defenceman who played in the National Hockey League (NHL) for the Quebec Nordiques Colorado Avalanche Colorado Avalanche Colorado Avalanche Colorado Avalanche and Quebec Nordiques. he was drafted in the 2nd round of overall of the 2002 NHL Entry Draft.
OptimTrans	**None**	Aaron Miller (born August 11 1971 in Buffalo, New York) is an United States former professional Ice hockey Defenceman who played in the National Hockey League (NHL) for the Quebec Nordiques and Colorado Avalanche . he was a member of the United States men's national Ice hockey team at the 2002 Winter Olympics and 2006 Winter Olympics.
SANA	**None**	Aaron Miller (born August 11 1971 in Buffalo, New York) is a retired United States professional Ice hockey Defenceman . he also played for the Quebec Nordiques and the Colorado Avalanche . Miller was also a member of United States 's ice hockey in the 2002 Winter Olympics and 2006 Winter Olympics .

Table 6: Example outputs from different methods. The red text stands for the hallucinated content in each generated description. Compared with DesKnow and Struct-Aware, SANA recovers all the table facts without generating any unfaithful content.

(1) Name_ID: Émile Mbouh **(2) Member_of_sports_team**: Le Havre AC **(3) Member_of_sports_team**: Perlis FA **(4) Member_of_sports_team**: Sport Benfica e Castelo Branco **(5) Member_of_sports_team**: Qatar SC **(6) Member_of_sports_team**: Vitória S.C. **(7) Member_of_sports_team**: Tanjong Pagar United FC **(8) Member_of_sports_team**: CS Chênois **(9) Member_of_sports_team**: Cameroon national football team **(10) Number_of_matches_played**: 46 **(11) Number_of_points / goals / set_scored**: 2 **(12) Member_of_sports_team**: Union Douala **(13) Member_of_sports_team**: Diamant Yaoundé **(14) Member_of_sports_team**: Ettifaq FC **(15) Member_of_sports_team**: Liaoning Whowin F.C. **(16) Member_of_sports_team**: Sabah FA **(17) Place_of_birth**: Douala **(18) Date_of_birth**: 30 May 1966 **(19) Sport**: Association football **(20) Participant_of**: 1994 FIFA World Cup **(21) Participant_of**: 1990 FIFA World Cup **(22) Position_played_on_team / speciality**: Midfielder

Model	Miss	Output
DesKnow	**2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 15, 16, 21, 22**	Émile Mbouh (born 30 May 1966) is a former Cameroon national football team Association football . he was born in Douala and played for the Tanjong Pagar United FC in the 1994 FIFA World Cup .
Struct-Aware	**2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 15, 16, 17, 21**	Émile Mbouh, (born 30 May 1966) is a Cameroonian retired Association football who played as a Midfielder . Le represented Cameroon national football team at the 1994 FIFA World Cup and 1994 FIFA World Cup . he played for Le FC Sport Yaoundé, United Yaoundé and Tanjong Pagar United FC
OptimTrans	**2, 3, 5, 6, 8, 12, 13, 14**	Émile Mbouh (born 30 May 1966) is a Cameroonian retired Association football who played as a Midfielder . born in Douala Émile began his career with Sport Benfica e Castelo Branco and Tanjong Pagar United FC . he also represented Cameroon national football team at the 1994 FIFA World Cup and 1990 FIFA World Cup . he also played for Sabah FA and Liaoning Whowin F.C. in the Malaysia Super League . he also played for Tanjong Pagar United FC and Liaoning Whowin F.C. in the Chinese Super League.
SANA	**3, 12, 13, 21**	Émile Mbouh (born 30 May 1966) is a retired Cameroonian Association football who played as a Midfielder . born in Douala Mbouh played club football in France for Sport Benfica e Castelo Branco Le Havre AC CS Chênois Vitória S.C. Tanjong Pagar United FC Qatar SC Ettifaq FC Tanjong Pagar United FC Qatar SC Sabah FA and Liaoning Whowin F.C. . Mbouh played for the Cameroon national football team (46 caps and scoring 2 goals) and two games at the 1994 FIFA World Cup .

Table 7: Example outputs from different methods. The red text stands for the hallucinated content in each generated description. This table contains a large number of facts to recover, which makes the case very challenging. In contrast with the other models, SANA misses much fewer facts and does not produce unfaithful content.

TILGAN: Transformer-based Implicit Latent GAN for Diverse and Coherent Text Generation

Shizhe Diao♦*, **Xinwei Shen**♦*, **KaShun Shum**♦, **Yan Song**♠♡, **Tong Zhang**♦
♦The Hong Kong University of Science and Technology
{sdiaoaa, xshenal, ksshumab, tongzhang}@ust.hk
♠The Chinese University of Hong Kong (Shenzhen)
♡Shenzhen Research Institute of Big Data
songyan@cuhk.edu.cn

Abstract

Conventional autoregressive models have achieved great success in text generation but suffer from the exposure bias problem in that token sequences in the training and in the generation stages are mismatched. While generative adversarial networks (GANs) can remedy this problem, existing implementations of GANs directly on discrete outputs tend to be unstable and lack diversity. In this work, we propose **TILGAN**, a **T**ransformer-based **I**mplicit **L**atent **GAN**, which combines a Transformer autoencoder and GAN in the latent space with a novel design and distribution matching based on the Kullback-Leibler (KL) divergence. Specifically, to improve local and global coherence, we explicitly introduce a multi-scale discriminator to capture the semantic information at varying scales among the sequence of hidden representations encoded by Transformer. Moreover, the decoder is enhanced by an additional KL loss to be consistent with the latent-generator. Experimental results on three benchmark datasets demonstrate the validity and effectiveness of our model, by obtaining significant improvements and a better quality-diversity trade-off in automatic and human evaluation for both unconditional and conditional generation tasks.[1]

1 Introduction

In recent years, Transformer-based autoregressive (AR) models have made a dramatic impact in text generation tasks such as machine translation (Vaswani et al., 2017; Wang et al., 2019) and dialogue systems (Le et al., 2019; Ham et al., 2020), especially with the emergence of large pre-trained language models (Radford et al., 2019; Brown et al., 2020; Wu et al., 2020). However, AR models predict the next token conditioned on the ground truth during training and on its own previously generated token during inference, which leads to a mismatch between training and generation stages, and this causes low quality of generated texts and bad generalization ability of models on unseen data (Wiseman and Rush, 2016; Welleck et al., 2020).

Generative adversarial networks (GANs, Goodfellow et al., 2014) provide a promising approach to solve the exposure bias problem (Yu et al., 2017; Kusner and Hernández-Lobato, 2016; Zhang et al., 2017). This is because GANs aim at matching the distributions of the generated and real data instead of forcing the model output to align with the single correct sequence, and thus provide the potential to bypass the discrepancy issue. However, it is non-trivial to apply GANs to discrete data since the gradients cannot be normally back-propagated through discrete tokens. Existing approaches have implemented the adversarial discrete generation training by reinforcement learning (RL) (Yu et al., 2017; Lin et al., 2017; Guo et al., 2018; Fedus et al., 2018) and Gumbel-Softmax (Kusner and Hernández-Lobato, 2016). Nevertheless, these approaches suffer from the high variance problem which causes unstable performance and slow convergence, leading to other methods based on feature matching (Zhang et al., 2017; Zhao et al., 2018; Chen et al., 2018).

In this work, we propose **TILGAN**, a **T**ransformer-based **I**mplicit **L**atent **GAN**, which combines a Transformer autoencoder and a GAN in the latent space with novel designs and a learning formulation based on the Kullback-Leibler (KL) divergence to enhance the text generation performance in both fidelity and diversity. Specifically, inspired by the representation capacity of Transformer AR models, we firstly incorporate Transformer architectures to improve GANs in text generation. Note that the previous latent feature matching methods are mostly RNN-based and assume

---

*Equal Contribution.

[1] Our code is available at `https://github.com/shizhediao/TILGAN`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4844–4858
August 1–6, 2021.

a single vector in the latent space, which do not directly handle a sequence of latent representations encoded by a Transformer. However, single latent vector representation hinders the incorporation of correlations among different tokens, leading to the loss of crucial semantic information captured by a Transformer structure. This is especially problematic for local and global coherence (Bińkowski et al., 2020). In this paper, we directly match the distributions of multi-token sequences in the latent space, which is better suited for the Transformer structure. To do so, we have to resolve two challenges, the first being how to do distribution matching. We introduce a multi-scale discriminator over the Transformer latent space to utilize the semantic information on different scales, where a global discriminator takes the entire sequence of latent representations as the input, and a local discriminator takes only a randomly-sampled local neighborhood. The second challenge is how to train the decoder reliably. We enhance an autoencoder loss by another KL loss optimized by GAN, forcing the latent representations of the decoding output to be compatible with the generated latent representations from the latent-generator.

We provide a theoretical justification for the proposed formulation by connecting it to the standard goal of generative modeling. Experimental results on three datasets illustrate that TILGAN outperforms all baselines in both unconditional and conditional generation tasks, achieving state-of-the-art performance. Particularly, TILGAN exhibits a better quality-diversity trade-off evaluated by automatic metrics such as SelfBLEU and TestBLEU as well as human evaluation. Further analyses also confirm the effectiveness of each component of our method, where decoder enhancement greatly benefits generation quality, while the multi-scale discriminator and KL objective provide great performance gains in generation diversity, and the implicit prior contributes to both.

2 The Approach

2.1 Model and Formulation

In this section, we introduce the proposed model and the learning formulation. Let $\boldsymbol{x} \in \mathcal{X}$ denote a sentence following the real data distribution $p_r(\boldsymbol{x})$ with $\mathcal{X} = \mathcal{V}^n$ where $\mathcal{V}$ is the vocabulary, $m = |\mathcal{V}|$ is the vocabulary size, and n is the sequence length, and $\boldsymbol{z} \in \mathcal{Z}$ be the latent variable following a prior distribution $p_z(\boldsymbol{z})$. We consider a probabilistic model containing an encoder $E_\phi : \mathcal{X} \to \mathcal{Z}$ and a decoder $G_\theta : \mathcal{Z} \to \mathcal{X}$. Both are generally stochastic mappings with parameters ϕ and θ, and induce the encoder conditional distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the decoder conditional $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ respectively. Note that previous approaches to text generation use deterministic encoders and decoders (Zhao et al., 2018), which restricts the expressiveness of the modeled distribution family. We first ensure the consistency between E_ϕ and G_θ by minimizing the negation of the expected reconstruction log-likelihood

$$L_c(\phi, \theta) = -\mathbb{E}_{\boldsymbol{x}\sim p_r}\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})], \quad (1)$$

which coincides with the reconstruction term in the evidence lower bound (ELBO).

The generated data distribution is given by $p_G(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim p_z}[p_\theta(\boldsymbol{x}|\boldsymbol{z})]$. To achieve good generation performance, we design the model so that the distribution family of $p_G(\boldsymbol{x})$ is large enough to contain the real one $p_r(\boldsymbol{x})$. As described in Section 2.3, we use Transformer to model E and G, which we assume to have sufficient capacity to reconstruct data well and learn informative latent representations. In this way, $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is assumed to be expressive enough. To further enhance the capacity of $p_G(\boldsymbol{x})$, we propose to use an implicit prior p_z, by transforming samples from a simple distribution with a deep neural network.

Consider a random vector $\epsilon \in \mathcal{E}$ following some simple distribution p_ϵ like a standard Gaussian. We then propose to learn a latent-generator $g_\beta : \mathcal{E} \to \mathcal{Z}$ with parameter β so that the distribution of $g_\beta(\epsilon)$ matches that of $E_\phi(\boldsymbol{x})$, by minimizing the KL divergence

$$L_g(\phi, \beta) = D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}) \| p_\beta(\boldsymbol{z})),$$

where $p_\beta(\boldsymbol{z})$ denotes the distribution of $g_\beta(\epsilon)$ and $q_\phi(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}\sim p_r}[q_\phi(\boldsymbol{z}|\boldsymbol{x})]$ is the distribution of $E(\boldsymbol{x})$, a.k.a., the aggregated posterior. The advantage of KL divergence is that it imposes a heavy penalty when $q_\phi(\boldsymbol{z}) > 0$ but $p_\beta(\boldsymbol{z}) \approx 0$, which means that it favors a g that covers all the diverse modes of $q_\phi(\boldsymbol{z})$. This is commonly known and verified empirically in Shen et al. (2020). Hence minimizing KL encourages a better diversity in generation compared with the Jensen–Shannon (JS) divergence or Wasserstein distance which are often used in the literature on generative models.

Therefore, we formulate the overall objective

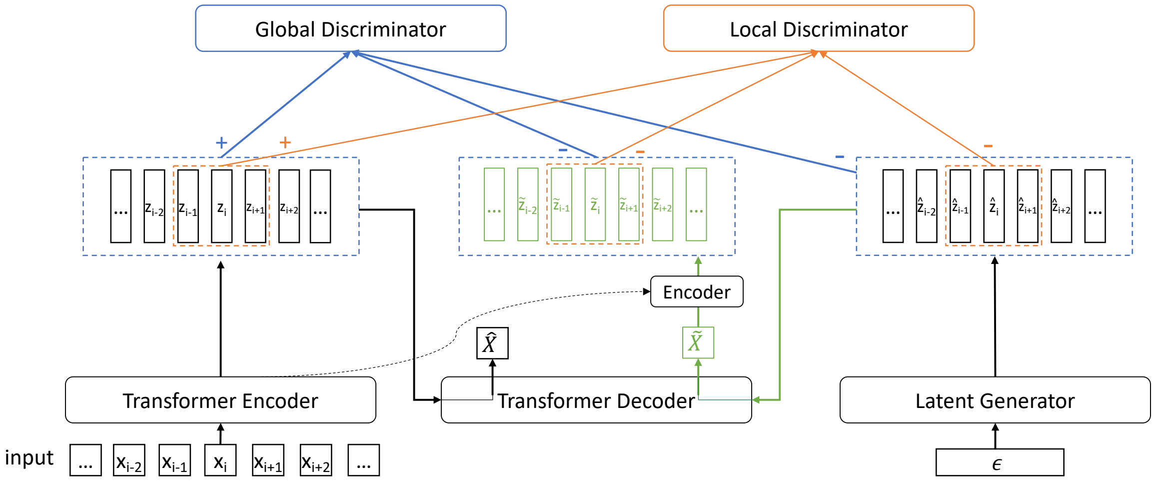

Figure 1: The overall architecture of TILGAN. Blue and orange stand for the global and local discriminators, respectively, and green denotes the route of the enhanced decoder.

function to be minimized as follows

$$L_c(\phi, \theta) + \lambda L_g(\phi, \beta), \tag{2}$$

where $\lambda > 0$ is a coefficient to balance both terms.

Decoder Enhancement During testing, a new sentence is generated by first sampling $\epsilon \sim p_\epsilon$, then computing the latent variable $g(\epsilon)$ and finally generating $G(g(\epsilon))$, which means the decoder takes the output of the latent-generator g as the input which it has never seen throughout the training. Although the KL term aims at matching the distributions of $g(\epsilon)$ and $E(\boldsymbol{x})$, it is possible that they do not match perfectly. In such cases, the decoder may generate data with poor fidelity and far from being real data. To resolve this and reliably train the decoder, we propose to enhance the decoder by letting it see the generated latent $g(\epsilon)$ during training. Formally, let $\tilde{p}_g$ be the distribution of $E(G(g(\epsilon)))$. We add another term to the loss function (2) with coefficient $\lambda_1 > 0$:

$$\lambda_1 D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}) \| \tilde{p}_g(\boldsymbol{z})). \tag{3}$$

Since this term is designed to enhance the decoder, we regard the encoder and prior parameters ϕ and β as fixed constants. In other words, in optimization, we do not propagate gradients of this term with respect to ϕ and β and only update the decoder parameter θ.

2.2 Algorithm

In this section, we propose a GAN-based algorithm for the optimization of the above formulation. Since $p_\beta(\boldsymbol{z})$ is implicit, the KL term L_g in (2) does not allow a closed form to be optimized directly. We introduce a discriminator to estimate the gradients, following Shen et al. (2020). In Lemma 1, we present the gradient formulas of L_g.

Lemma 1. *Let* $\mathcal{D}(\boldsymbol{z}) = \ln(q_\phi(\boldsymbol{z})/p_\beta(\boldsymbol{z}))$. *Then*

$$\begin{aligned}
\nabla_\phi L_g &= \mathbb{E}[\nabla_{\boldsymbol{z}} \mathcal{D}(E_\phi(\boldsymbol{x}))^\top \nabla_\phi E_\phi(\boldsymbol{x})],\\
\nabla_\beta L_g &= -\mathbb{E}[s_{\mathcal{D}}(g_\beta(\epsilon)) \nabla_{\boldsymbol{z}} \mathcal{D}(g_\beta(\epsilon))^\top \nabla_\beta g_\beta(\epsilon)],
\end{aligned}$$

where $s_{\mathcal{D}}(\boldsymbol{z}) = e^{\mathcal{D}(\boldsymbol{z})}$ *is the scaling factor and the expectations are taken over all the randomness.*

Since $\mathcal{D}$ depends on the unknown densities q_ϕ and p_β so that the gradients in Lemma 1 can not be directly computed from the data, we estimate the gradients by training a discriminator D_ψ with parameter ψ via the empirical logistic regression:

$$\min_\psi \left[\sum_{\boldsymbol{z} \in S_e} \frac{\ln(1 + e^{-D_\psi(\boldsymbol{z})})}{|S_e|} + \sum_{\boldsymbol{z} \in S_g} \frac{\ln(1 + e^{D_\psi(\boldsymbol{z})})}{|S_g|} \right],$$

where S_e and S_g are finite samples from $q_\phi(\boldsymbol{z})$ and $p_\beta(\boldsymbol{z})$ respectively. This leads to a GAN algorithm. The optimization of the enhanced loss (3) is similar.

However, GAN is commonly known to suffer from unstable training or gradient vanishing. To stabilize our algorithm, we adopt the scaling clipping technique from Shen et al. (2020) and clip the scaling factor into a range of $[r_0, 1/r_0]$, where $r_0 = 0.5$ turns out to work well in all our experiments. Denote the clipped scaling by $s'_D(\boldsymbol{z}) = \max\{\min\{s_D(\boldsymbol{z}), 2\}, 0.5\}$.

For the optimization of the consistency loss L_c, we adopt the reparametrization trick from Kingma and Welling (2014) and estimate it by

```
Algorithm 1: TILGAN
Input: initial $\phi, \theta, \beta, \psi, \xi$, batch-size $N$, local size $M$
while not convergence do
    // Update discriminators
    Sample $\{\boldsymbol{x}_i\}_{i=1}^N \sim p_r(\boldsymbol{x})$, $\{\epsilon_i\}_{i=1}^N \sim p_\epsilon$
    Compute
      $\boldsymbol{z}_i = E_\phi(\boldsymbol{x}_i), \hat{\boldsymbol{z}}_i = g_\beta(\epsilon_i), \tilde{\boldsymbol{z}}_i = E_\phi(G_\theta(\hat{\boldsymbol{z}}_i))$
    Update $\psi$ by descending the gradient:
      $\frac{1}{N}\sum_{i=1}^N \nabla_\psi[\ln(1+e^{-D_\psi(\boldsymbol{z}_i)})+\ln(1+e^{D_\psi(\hat{\boldsymbol{z}}_i)})]$
    Randomly sample local blocks $\boldsymbol{z}'_i$ and $\hat{\boldsymbol{z}}'_i$ with size
      $M$
    Update local discriminator by descending:
      $\frac{1}{N}\sum_{i=1}^N \nabla_\xi[\ln(1+e^{-d_\xi(\boldsymbol{z}'_i)})+\ln(1+e^{d_\xi(\hat{\boldsymbol{z}}'_i)})]$
    // Update encoder, decoder and latent-generator
    Obtain $\boldsymbol{x}_i, \epsilon_i, \boldsymbol{z}_i, \hat{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_i, \boldsymbol{z}'_i$ and $\hat{\boldsymbol{z}}'_i$ as above
    Compute $\phi$-gradient:
      $\frac{1}{N}\sum_{i=1}^N[\nabla_\phi \hat{L}_c(\boldsymbol{x}_i, \boldsymbol{z}_i)+\lambda\nabla_\phi D_\psi(\boldsymbol{z}_i)+\lambda\nabla_\phi d_\xi(\boldsymbol{z}'_i)]$
    Compute $\beta$-gradient:
      $-\frac{1}{N}\sum_{i=1}^N \lambda[s'_D(\hat{\boldsymbol{z}}_i)\nabla_\phi D_\psi(\hat{\boldsymbol{z}}_i)+s'_d(\hat{\boldsymbol{z}}'_i)\nabla_\phi d_\xi(\hat{\boldsymbol{z}}'_i)]$
    Compute $\theta$-gradient:
      $\frac{1}{N}\sum_{i=1}^N[\nabla_\theta \hat{L}_c(\boldsymbol{x}_i, \boldsymbol{z}_i)+\lambda_1 s'_D(\tilde{\boldsymbol{z}}_i)\nabla_\phi D_\psi(\tilde{\boldsymbol{z}}_i)]$
    Update parameters $\phi, \theta, \beta$ using the gradients
Return: $\phi, \theta, \beta$
```

$\frac{1}{n}\sum_{i=1}^n \hat{L}_c(\boldsymbol{x}_i, \boldsymbol{z}_i)$ where $\boldsymbol{x}_i \sim p_r(\boldsymbol{x})$, $\boldsymbol{z}_i = E_\phi(\boldsymbol{x}_i)$, and $\hat{L}_c(\boldsymbol{x}_i, \boldsymbol{z}_i) = -\ln p_\theta(\boldsymbol{x}_i|\boldsymbol{z}_i)$. The whole training procedure is summarized in Algorithm 1, where the colored parts stand for the enhanced decoder (green) and the multi-scale discriminator (blue and orange) introduced later.

2.3 Architecture

In this section, we present the Transformer-based architecture incorporated with multi-scale discriminators. We propose a Transformer autoencoder framework where both the encoder and decoder are self-attention layers with three novel ingredients specific to improve the generation performance in both quality and diversity: (i) a latent-generator g to transform Gaussian noises into an implicit prior distribution, (ii) decoder enhancement, and (iii) multi-scale discriminators. Figure 1 illustrates the entire architecture of TILGAN.

As mentioned in Section 1, we introduce multiple discriminators over the Transformer's latent space to utilize the semantic information on different scales, each of which operates on a different window of representations as the input. Specifically, given an input sentence $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ where x_i stands for the i-th word, it is passed through the Transformer encoder which results in a sequence of latent states $\boldsymbol{z} = [z_1, z_2, \ldots, z_n]$ where z_i is the vector representation corresponding to x_i. We introduce a global discriminator D_ψ taking the whole sequence of representations $\boldsymbol{z}$ as the input, and a local discriminator d_ξ with parameter ξ taking only a local neighborhood of the M randomly-sampled adjacent representations, e.g., $\boldsymbol{z}' = [z_{i-1}, z_i, z_{i+1}]$ with $M = 3$, as the input. [2] Notably, the local discriminator takes the generated pieces of sequences into account, so it provides signals of phrase-level fidelity and local coherence, while the global discriminator is able to assess the general realism and the degree of natural coherence for the whole sequence.

2.4 Extension to Conditional Generation

Our proposed framework can be readily extended to conditional generation tasks such as story completion. To be specific, the goal is to learn a conditional real data distribution $p_r(\boldsymbol{x}|\boldsymbol{c})$ where $\boldsymbol{c}$ is the given context following $p_r(\boldsymbol{c})$ with some missing content $\boldsymbol{x}$ to complete. We propose to feed $\boldsymbol{c}$ into all three components—encoder E, decoder G, and latent-generator g—of our model, and modify the terms in objective function (2) as follows

$$L'_c(\phi, \theta) = -\mathbb{E}_{p_r(\boldsymbol{x},\boldsymbol{c})}\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{c})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{c})],$$
$$L'_g(\phi, \beta) = D_{\mathrm{KL}}(q^c_\phi(\boldsymbol{z})\|p^c_\beta(\boldsymbol{z})),$$

where $p_r(\boldsymbol{x}, \boldsymbol{c}) = p_r(\boldsymbol{x}|\boldsymbol{c})p_r(\boldsymbol{c})$, and the marginal distributions of $E(\boldsymbol{x}, \boldsymbol{c})$ and $g(\epsilon, \boldsymbol{c})$ are given by $q^c_\phi(\boldsymbol{z}) = \mathbb{E}_{p_r(\boldsymbol{x},\boldsymbol{c})}[q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{c})]$ and $p^c_\beta(\boldsymbol{z}) = \mathbb{E}_{p_r(\boldsymbol{x},\boldsymbol{c})}[p_\beta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{c})]$ respectively. Then the final objective is to minimize $L'_c(\phi, \theta) + \lambda L'_g(\phi, \beta)$.

3 Theoretical Justification

The goal of generative modeling is to learn the generated distribution $p_G(\boldsymbol{x})$ that is close to the real data distribution $p_r(\boldsymbol{x})$. Our proposed formulation in (2), however, does not explicitly optimize a distance measure between p_G and p_r, so it is unclear whether our method can match the distributions in the data space. In this section, we provide justification for the proposed formulation (2) by connecting it with the above goal, based on the analysis of WAE (Tolstikhin et al., 2018).

Let $\mathbb{P}_G$ and $\mathbb{P}_r$ be the induced probability measures of $p_G(\boldsymbol{x})$ and $p_r(\boldsymbol{x})$ respectively. We have the Kantorovich's formulation of the optimal transport (OT) problem with the L_1 cost:

$$W_1(\mathbb{P}_r, \mathbb{P}_G) = \inf_{\Gamma \in \mathcal{P}(\boldsymbol{x}\sim\mathbb{P}_r, \boldsymbol{y}\sim\mathbb{P}_G)} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\Gamma}[c(\boldsymbol{x}, \boldsymbol{y})],$$

---

[2] We have considered sampling multiple different neighborhoods within a given sequence as well, whose empirical performance was shown to be comparable with our proposed scheme with one local neighborhood, so we only reported the latter since it is simpler to implement.

TASK	UG		CG
DATASET	MSCOCO	WMTNEWS	ROCSTORY
VOCAB	27842	5728	20000
AVG LEN.	10.4	27.8	10.0
TRAIN S#	120K	278K	390K
DEV S#	-	-	50K
TEST S#	10K	10K	50K

Table 1: The statistics of the datasets. Avg Len. means the average length of sentences. S# refers to number of sentences. UG and CG stand for unconditional generation and conditional generation, respectively.

where $c(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_1$ is the cost function and $\mathcal{P}(\boldsymbol{x} \sim \mathbb{P}_r, \boldsymbol{y} \sim \mathbb{P}_G)$ is a set of all joint distributions of $(\boldsymbol{x}, \boldsymbol{y})$ with marginals $\mathbb{P}_r$ and $\mathbb{P}_G$ respectively. Note that $W_1(\mathbb{P}_r, \mathbb{P}_G)$ is also known as the 1-Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_G$. Then we have the following theorem which gives an upper bound of the 1-Wasserstein distance, whose proof is given in Appendix B.

Theorem 1. *Let $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ be a multivariate multinomial distribution with mean matrix $\bar{G}(\boldsymbol{z}) \in \mathbb{R}^{m\times n}$ which is a common choice for text modeling, i.e., each one-hot token $x_i|\boldsymbol{z}$ follows a multinomial with mean $\bar{G}_i(\boldsymbol{z}) \in$ simplex Δ^{m-1} for $i = 1, \ldots, n$. Then we have $W_1(\mathbb{P}_r, \mathbb{P}_G)$ is upper bounded by*

$$\inf_{q(\boldsymbol{z}|\boldsymbol{x}):q_z(\boldsymbol{z})=p_\beta(\boldsymbol{z})} -2\mathbb{E}_{\boldsymbol{x}\sim p_r}\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})], \tag{4}$$

where $q_z(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}\sim p_r(\boldsymbol{x})}[q(\boldsymbol{z}|\boldsymbol{x})]$ is the aggregated posterior and $p_\beta(\boldsymbol{z})$ is the implicit prior.

Hence by minimizing (4) with respect to θ and β, we learn the composite generator $G_\theta(g_\beta(\epsilon)) : \mathcal{E} \to \mathcal{X}$ that minimizes an upper bound of $W_1(\mathbb{P}_r, \mathbb{P}_G)$, which is consistent with the standard goal of generative modeling. However, this optimization problem is generally intractable due to the equality constraint and the nonparametric nature. Our formulation (2) can be regarded as an approximate problem of it by parametrizing $q(\boldsymbol{z}|\boldsymbol{x})$ with a distribution family induced by a stochastic encoder mapping E_ϕ, and relaxing the hard constraint $q_z(\boldsymbol{z}) = p_\beta(\boldsymbol{z})$ by introducing the relative entropy regularization $D_{\mathrm{KL}}(q_z(\boldsymbol{z})\|p_\beta(\boldsymbol{z}))$.

4 Experiment Settings

4.1 Datasets

We conduct our experiments on three benchmark datasets, **MSCOCO** (Lin et al., 2014), **WMTNEWS** (Guo et al., 2018), and **ROCSTORY** (Mostafazadeh et al., 2016). All of the preprocessing steps are the same as Chen et al. (2018) and Wang and Wan (2019). The statistics of the resulting datasets are reported in Table 1.

4.2 Baselines

Unconditional Generation Three simplified variants of TILGAN are implemented for comparison:

- **TILGAN$_{\text{P}}$**: a plain baseline using our backbone model, that is, a Transformer autoencoder and a GAN in the latent space based on KL divergence.
- **TILGAN$_{\text{E}}$**: TILGAN$_{\text{P}}$ equipped with decoder enhancement.
- **TILGAN$_{\text{MD}}$**: TILGAN$_{\text{P}}$ with the multi-scale discriminator.

In addition, the following existing models are adopted: recurrent neural network language model (RNNMLE), SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017), GSGAN (Kusner and Hernández-Lobato, 2016), LeakGAN (Guo et al., 2018), textGAN (Zhang et al., 2017), FM-GAN (Chen et al., 2018), ARAE (Zhao et al., 2018), Transformer language model (TMLE) (Vaswani et al., 2017).

Conditional Generation For conditional generation, we compare our model with Transformer (Vaswani et al., 2017), IE+MSA (Guan et al., 2019), Seq2Seq (Bahdanau et al., 2015), HLSTM (Li et al., 2015), CVAE (Sohn et al., 2015), and T-CVAE (Wang and Wan, 2019).

4.3 Automatic Evaluation Metrics

Unconditional Generation

- **TESTBLEU** (Yu et al., 2017): a quality metric comparing the n-gram similarity between generated samples and the whole test set.
- **SELFBLEU** (Zhu et al., 2018): a diversity metric calculating the similarity between one generated sentence and the whole remaining generation. The lower the SelfBLEU score is, the higher diversity we obtain in the generation.

Specifically, following Chen et al. (2018), we report BLEU-2/3/4/5 for TestBLEU and BLEU-2/3/4 for SelfBLEU.

Conditional Generation

- **BLEU** (Papineni et al., 2002): the BLEU score is calculated by taking the geometric mean of the n-gram BLEU scores where n is from 1 to 4.
- **DIVERSITY** (Li et al., 2016): the proportion of distinct n-grams in the generated results which evaluates the degree of diversity. D1 and D2 are reported for unigram and bigram, respectively.

(a) MSCOCO

METHODS	SELFBLEU			TESTBLEU				HUMAN	
	B2%	B3%	B4%	B2%	B3%	B4%	B5%	Q	D
RNNMLE	75.4	51.1	23.2	82.0	60.7	38.9	24.8	3.33	2.8
SEQGAN	80.7	57.7	27.8	82.0	60.4	36.1	21.1	3.86	2.8
RANKGAN	82.2	59.2	28.8	85.2	63.7	38.9	24.8	3.26	2.6
GSGAN	78.5	52.2	23.0	81.0	56.6	33.5	19.7	3.15	2.4
LEAKGAN	91.2	82.5	68.9	92.2	79.7	60.2	41.6	3.07	3.0
TEXTGAN	80.6	54.8	21.7	91.0	72.8	48.4	30.6	3.55	2.8
FMGAN	83.1	63.2	32.5	94.2	81.2	61.8	41.4	4.06	2.2
ARAE	63.2	41.6	19.1	86.7	69.3	44.2	24.5	2.93	3.0
TMLE	70.6	47.6	27.3	92.8	81.9	56.2	33.1	3.75	3.4
TILGAN	**61.6**	**35.6**	**9.9**	96.7	90.3	77.2	**53.2**	**4.38**	**3.8**
TILGAN$_{P}$	61.7	45.9	18.2	94.7	86.6	63.1	39.9	-	-
TILGAN$_{E}$	70.5	50.1	28.7	**98.8**	**94.5**	**81.3**	52.5	-	-
TILGAN$_{MD}$	63.5	38.7	11.8	95.1	84.9	64.8	44.1	-	-
IMP_POST	73.3	63.8	47.7	95.5	87.1	69.8	31.5	-	-
JSGAN	76.7	67.9	51.8	75.1	54.2	32.1	11.0	-	-
WGAN	90.4	80.9	69.0	73.0	53.8	34.2	12.5	-	-

(b) WMTNews

METHODS	SELFBLEU			TESTBLEU				HUMAN	
	B2%	B3%	B4%	B2%	B3%	B4%	B5%	Q	D
RNNMLE	66.4	33.7	**11.3**	76.1	46.8	23.1	11.6	3.65	2.8
SEQGAN	72.8	41.1	13.9	63.0	35.4	16.4	8.7	3.29	3.4
RANKGAN	67.2	34.6	11.8	77.4	48.4	24.9	13.1	2.98	3.8
GSGAN	68.2	41.0	23.1	72.3	44.0	21.0	10.7	3.39	2.6
LEAKGAN	85.7	69.6	37.3	92.0	72.5	50.2	32.1	2.51	2.8
TEXTGAN	80.6	54.8	28.7	77.7	52.9	30.5	16.1	3.43	3.2
FMGAN	83.1	68.2	38.5	**93.2**	77.1	55.2	39.9	3.40	3.2
ARAE	53.4	30.4	17.3	84.4	62.9	39.8	22.0	2.29	2.6
TMLE	61.3	43.7	25.1	87.5	74.8	44.2	26.4	3.31	3.6
TILGAN	66.3	44.5	28.0	92.9	**81.7**	**61.7**	**40.7**	**3.81**	**4.0**
TILGAN$_{P}$	64.8	48.2	34.9	88.9	76.5	56.5	27.5	-	-
TILGAN$_{E}$	62.7	43.3	23.0	91.5	79.2	59.6	33.9	-	-
TILGAN$_{MD}$	**53.1**	**33.2**	20.6	92.6	78.2	53.9	29.5	-	-
IMP_POST	71.2	59.7	47.1	73.1	68.2	46.6	20.1	-	-
JSGAN	64.7	49.9	39.6	80.8	64.9	41.1	14.2	-	-
WGAN	93.3	91.0	88.6	89.1	77.4	50.2	26.7	-	-

Table 2: SelfBLEU and TestBLEU results on MSCOCO and EMNLP WMTNews datasets. Q and D denote the quality and diversity evaluated by human, respectively. The results of previous baselines are listed in the top region, our method TILGAN together with simplified variants are shown in the middle, and more variants for further ablation studies are at the bottom. The bold numbers are the best results in each column.

- **ADVERSARIAL SUCCESS** (Li et al., 2017): the fraction of instances in which a model is capable of fooling a fine-tuned BERT classifier with the above 95% accuracy on the development set of the classification task. Higher values are better. The positive examples are original stories and negative examples are stories consisting of a random sentence from another story.

4.4 Implementation

Unconditional Generation We implement a Transformer-based autoencoder with 2 layers, 4 heads, 512 embedding dimensions, and 512 hidden dimensions. The generator and discriminator are implemented by 3 layers multi-layer perceptron (MLP). We set the maximum sequence length to 15 and 32 for MSCOCO and WMTNews, respectively. During training, each sentence is padded to the maximum length when fed into the encoder, and then the encoder produces a latent vector for every input token. During testing, the latent-generator generates a sequence of latent vectors with the same maximum length, and then, conditional on the latent vectors, the decoder generates a sentence which ends when a special token, <EOS>, is generated. We adopt Adam (Kingma and Ba, 2015) as the optimizer with a learning rate of 0.00025 and 0.0001 for autoencoder and GAN structure, respectively with a dropout rate of 0.3.

Conditional Generation We adopt the same Transformer encoder-decoder architecture as the backbone model and similar setups as Wang and Wan (2019). The Transformer structure has 6 layers, 8 self-attention heads, 512 dimensions for hidden states, and uses shared attention layers for encoder and decoder which allows the decoder to attend to the encoder state and the decoder state at the same time to make the completed story more coherent. The generator has 3 layers and the discriminator has 4 layers. We adopt Adam (Kingma and Ba, 2015) as the optimizer with a learning rate of 0.0001 and a dropout rate of 0.15.

More details of the experimental setup and hyperparameter settings are shown in the Appendix A.

5 Experimental Results

5.1 Unconditional Generation

• **Generation Quality** The first experiment is to compare the quality of generated sentences of different models. In general, as shown in Table 2, TILGAN outperforms all baseline models in TestBLEU on both MSCOCO and WMTNews datasets, which clearly indicates the advantages of our proposed framework. We make five main observations. Firstly, we notice that TMLE is comparable to most GAN baselines, which shows the powerful fitting capacity of the Transformer architecture as well as the inferior performance of previous GAN implementations. Despite this, TILGAN$_{P}$ outperforms TMLE by a wide margin, demonstrating that our backbone model combining a Transformer autoencoder and a GAN can not only take advantage of the Transformer's capacity, but also exhibit benefits from our GAN formulation to further boost the performance. Furthermore, compared with TILGAN$_{\mathrm{P}}$, our full version TILGAN achieves an improvement of 13.3% and 13.2% for TestBLEU5 on MSCOCO and News, respectively. This confirms the effectiveness of the proposed multi-scale discriminators and decoder enhancement. In detail, when comparing the simplified variants TILGAN$_{\mathrm{E}}$ v.s. TILGAN$_{\mathrm{MD}}$, the improvement of TILGAN$_{\mathrm{E}}$ over the plain baseline TILGAN$_{\mathrm{P}}$ is larger than that of TILGAN$_{\mathrm{MD}}$, which illustrates that incorporating TILGAN$_{\mathrm{E}}$ is more crucial to improving the generation quality. Lastly, we compare TILGAN with all previous methods and observe an average improvement of 6.3% for TestBLEU5 on two datasets against the previous state-of-the-art FMGAN, which suggests the superiority of our method.

• **Generation Diversity** The generation diversity is evaluated by SelfBLEU scores, which are shown in Table 2. First, compared with the baselines with comparable and worse TestBLEU, e.g., FMGAN and LeakGAN, our TILGAN achieves lower SelfBLEU scores, which indicates a better quality-diversity trade-off from TILGAN. We notice that RNNMLE achieves the best SelfBLEU score on WMTNews but its quality shown by TestBLEU is pretty low and tends to generate incoherent or meaningless segments, which can be confirmed by the generated samples in Appendix C. In addition, from the results of TILGAN$_{\mathrm{MD}}$, we find that incorporating the multi-scale discriminator leads to a significant drop in SelfBLEU, suggesting that most of the performance gains in generation diversity are attributed to our design of multi-scale discriminators in contrast to decoder enhancement. Moreover, when comparing the performance across two datasets, we find that the SelfBLEU scores of our models are lower on MSCOCO than that on WMTNews, illustrating that it is easier to generate more diverse texts on MSCOCO. One possible reason is that the texts in MSCOCO are shorter than the texts in WMTNews as shown in Table 1. When generating long sequences, models are prone to generate repeated tokens and phrases. The same phenomenon was also observed for many other baseline models like ARAE and FMGAN.

5.2 Conditional Generation

In addition to unconditional generation, we test our model in a story completion task to verify its ability in conditional generation. Table 3 shows the automatic metrics in four metrics, with several observations. (i) Overall, among all models, TILGAN achieves new state-of-the-art results on the ROCStory dataset, showing the superiority of our method. (ii) Our model obtains substantial improvement in the quality metrics of generated answers, with 0.32% gains in BLEU, 6.92% gains in the adversarial success. It demonstrates that the generated plots are in high coherence, which not only share a higher proportion of word overlap with ground-truth answers, but also have a higher success rate fooling the BERT classifier. (iii) As for diversity, TILGAN improves upon the state-of-the-art methods from 3.63% to 3.88% on D1 and 23.46% to 25.61% on D2, showing that TILGAN produces stories consisting of more diverse and distinct n-grams.

5.3 Human Evaluation

Due to the limitations of automatic evaluation metrics, we invite 5 judges to rate 100 sentences generated by different models on a scale from 1 to 5 for both unconditional and conditional generation tasks. The results for unconditional generation are shown in Table 2. TILGAN shows a superior performance, which confirms that our model is able to generate more realistic samples than the baseline models on two datasets. Among all baseline models, FMGAN has a high quality score but a low diversity score, which indicates that most of its generated samples are repeated sentences that lack diversity. Additional evidence is shown through the case study in Section 6.2.

In addition, the human evaluation results on story completion are shown in Table 3 where we only compare TILGAN with the best baseline, i.e., T-CVAE. We use Gram metric to evaluate whether the generated story plot proceeds naturally, and Logic metric to evaluate whether the plot is reasonable and coherent following Wang and Wan (2019). Compared with T-CVAE, TILGAN is better in both Gram and Logic, demonstrating that the generated story plots of TILGAN are more natural and coherent.

METHODS	BLEU%	D1%	D2%	AS%	GRAM	LOGIC
SEQ2SEQ	2.90	2.69	15.95	80.97	-	-
HLSTM	2.31	2.63	14.80	72.46	-	-
CVAE	3.03	2.72	16.32	81.18	-	-
TRANS.	3.05	2.93	16.75	82.51	-	-
T-CVAE	4.25	3.63	23.46	87.54	3.32	3.24
TILGAN	**4.57**	**3.88**	**25.61**	**94.46**	3.58	3.60

Table 3: Results on story completion task. AS refers to adversarial success score. TRANS. denotes a vanilla Transformer model.

6 Analyses

6.1 Ablation Study

To examine the impact of KL loss and the implicit prior, we conduct ablation studies of different designs. We construct three variants of TILGAN and conduct experiments on two datasets of unconditional generation, whose results are shown in the bottom region of Table 2.

• **Impact of KL Loss** First, we implement two variants named JSGAN (Goodfellow et al., 2014) and WGAN (Arjovsky et al., 2017) by replacing the KL loss term with JS divergence and Wasserstein distance, while keeping the same architectures. In general, TILGAN_P outperforms JSGAN and WGAN in terms of SelfBLEU and TestBLEU on two datasets. Particularly, it is observed that with KL loss, the SelfBLEU4 score drops from 51.8% and 69.0% to 18.2% over JSGAN and WGAN on MSCOCO, and similar downward trends are observed on WMTNews. It demonstrates that minimizing KL loss indeed benefits the generation diversity, which is consistent with previous findings in Shen et al. (2020). In addition, the TestBLEU of TILGAN_P achieves an improvement of 28.9% and 27.4% for TestBLEU5 on MSCOCO over JSGAN and WGAN, respectively.

• **Implicit Prior v.s. Implicit Posterior** In addition to imposing an implicit prior, one can instead impose an implicit posterior as well by moving the transformation network of the latent-generator to the encoder and leaving a Gaussian prior. This results in a variant with nearly the same total number of parameters, named IMP_POST. We see from Table 2 that IMP_POST performs worse than TILGAN_P with an implicit prior, suggesting that enlarging the distribution family of posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ contributes less to improving the overall generation performance than enlarging that of prior $p_z(\boldsymbol{z})$, which is consistent to the analysis in the second paragraph of Section 2.1.

6.2 Case Study

To further analyze the real quality and diversity of the generated sentences, some are examined and presented in Table 4 and more examples are shown in Appendix C. First, the samples generated by TILGAN are more coherent and semantically meaningful. The majority of texts of TILGAN are in subject–verb–object order while those of other models are not. In addition, TILGAN exhibits more diverse sentence structures and word choices than others. For example, although each sentence generated by FMGAN looks good in quality, there are many repeated sentences or phrases, leading to a low diversity. The case study is consistent with the human evaluation results in Section 5.3.

7 Related Work

Conventional text generation models leverage maximum likelihood estimation (MLE) with teacher forcing and have shown powerful generation capabilities (Mikolov et al., 2010; Cho et al., 2014; Bahdanau et al., 2016; Radford et al., 2019; Brown et al., 2020) but they suffer from the exposure bias problem. To address this, several solutions were introduced including scheduled sampling (Bengio et al., 2015), professor forcing (Lamb et al., 2016), and Gibbs sampling (Su et al., 2018).

GAN-based text generation methods can be categorized into three classes: reinforcement learning (RL) based methods, Gumbel-Softmax (GS) based methods and latent feature matching methods. RL-based methods (Yu et al., 2017; Lin et al., 2017; Che et al., 2017; Guo et al., 2018; Fedus et al., 2018) design a reward incorporated with the discriminators, and use policy gradient or actor-critic approaches to update the generator to resolve the gradient propagating issue over discrete tokens. However, they suffer from high variance and mode collapse issues caused by the unstable policy gradient training process and the lack of a reliable guiding signal (Zhang et al., 2017; Chen et al., 2018). GS-based methods (Kusner and Hernández-Lobato, 2016) apply Gumbel-Softmax which is a continuous relaxation technique for transforming the output of a generator to be as close to one-hot as possible in order to make the samples from a discrete distribution like a multinomial differentiable with respect to the distribution parameters.

Latent feature matching methods (Zhang et al., 2017; Zhao et al., 2018) learn a manifold in the latent space instead of the discrete output space.

RankGAN:	(1) *A blue blue train sits on tracks with his residential asian toys.* (2) *A reflection of two birds walking by a sidewalk.*
FMGAN:	(1) *A man is standing on a table with a dog.* (2) *A man is standing on a table with a dog on a field.* (3) *A man is standing on a field of a large building.*
TILGAN:	(1) *A little boy sitting on a bench with a little girl.* (2) *A blue and white public transit bus is driving down acity street.* (3) *A train is going down the tracks in a forest.*

Table 4: Examples of generated sentences from RankGAN, FMGAN and our model.

This kind of methods usually incorporates an autoencoder to build the feature space and force the generator's latent output distribution to approach the real data latent distribution. Our method also resides in this category. To ease adversarial training, Zhang et al. (2017) introduce adversarial feature matching method by incorporating a kernelized discrepancy metric to match high-dimensional latent representations of real and synthetic sentences. ARAE (Zhao et al., 2018) extends AAE (Makhzani et al., 2015) to model discrete sequences and learns a parameterized prior by a generative model trained with WGAN. In contrast to our TILGAN whose Transformer-based encoder and decoder are both stochastic, ARAE uses RNN-based encoder and decoder which are both deterministic, as required in their theory, which reduces the model expressiveness and results in much poorer performance than ours as shown in Table 2. iVAE (Fang et al., 2019) proposes a VAE (Kingma and Welling, 2014) with an implicit posterior which is inferior to the implicit prior that we adopt according to the ablation study in Section 6.1. WAE-S (Bahuleyan et al., 2019) is a WAE(Tolstikhin et al., 2018) with a stochastic encoder trained using MMD with a distinct goal of improving the reconstruction ability.

8 Conclusion

In this paper, we proposed Transformer-based Implicit Latent GAN (TILGAN), for text generation. It combines a Transformer autoencoder and a GAN through matching the distributions of multi-token sequences in the Transformer's latent space based on KL divergence. To improve the local and global coherence, we introduced a multi-scale discriminator to utilize the semantic information on varying scales. To train the decoder reliably, we enhanced the objective function by another KL term, forcing the decoder to be compatible with the latent-generator. We theoretically connected the proposed formulation with the standard goal of generative modeling. Empirically, TILGAN achieved the state-of-the-art performance on three widely used datasets for unconditional tasks and story completion task, which demonstrated the effectiveness of our method to generate texts of high quality and diversity compared with the existing approaches.

Acknowledgments

This work was supported by the General Research Fund (GRF) of Hong Kong (No. 16201320). Y. Song was supported by NSFC under the project "The Essential Algorithms and Technologies for Standardized Analytics of Clinical Texts" (12026610) and Shenzhen Institute of Artificial Intelligence and Robotics for Society under the project "Automatic Knowledge Enhanced Natural Language Understanding and Its Applications" (AC01202101001). The authors also want to thank the anonymous reviewers for their valuable comments and suggestions.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end Attention-based Large Vocabulary Speech Recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE.

Hareesh Bahuleyan, Lili Mou, Hao Zhou, and Olga Vechtomova. 2019. Stochastic Wasserstein Autoencoder for Probabilistic Sentence Generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4068–4076.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *Advances in Neural Information Processing Systems*, pages 1171–1179.

Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, and Karen Simonyan. 2020. High Fidelity Speech Synthesis with Adversarial Networks. In *International Conference on Learning Representations*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-Likelihood Augmented Discrete Generative Adversarial Networks. *arXiv preprint arXiv:1702.07983*.

Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. 2018. Adversarial Text Generation via Feature-Mover's Distance. In *Advances in Neural Information Processing Systems*, pages 4666–4677.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong, and Changyou Chen. 2019. Implicit Deep Latent Variable Models for Text Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3937–3947.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. MaskGAN: Better Text Generation via Filling in the _. In *International Conference on Learning Representations*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, pages 2672–2680.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story Ending Generation with Incremental Encoding and Commonsense Knowledge. In *33rd AAAI Conference on Artificial Intelligence, AAAI 2019*.

J Guo, S Lu, H Cai, W Zhang, Y Yu, and J Wang. 2018. Long Text Generation via Adversarial Training with Leaked Information. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-End Neural Pipeline for Goal-Oriented Dialogue Systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Matt J Kusner and José Miguel Hernández-Lobato. 2016. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *arXiv preprint arXiv:1611.04051*.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor Forcing: A New Algorithm for Training Recurrent Networks. In *Advances in Neural Information Processing Systems*, pages 4601–4609.

Hung Le, Doyen Sahoo, Nancy Chen, and Steven Hoi. 2019. Multimodal Transformer Networks for End-to-End Video-Grounded Dialogue Systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5612–5623.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference* *on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial Learning for Neural Dialogue Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial Ranking for Language Generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*, pages 740–755. Springer.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Eleventh annual conference of the international speech communication association*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9.

Xinwei Shen, Tong Zhang, and Kani Chen. 2020. Bidirectional Generative Modeling Using Adversarial Gradient Estimation. *arXiv preprint arXiv:2002.09161*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28, pages 3483–3491. Curran Associates, Inc.

Jinyue Su, Jiacheng Xu, Xipeng Qiu, and Xuanjing Huang. 2018. Incorporating Discriminator in Sentence Generation: a Gibbs Sampling Method. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2018. Wasserstein autoencoders. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning Deep Transformer Models for Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822.

Tianming Wang and Xiaojun Wan. 2019. T-CVAE: Transformer-Based Conditioned Variational Autoencoder for Story Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5233–5239.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural Text Generation With Unlikelihood Training. In *International Conference on Learning Representations*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

Qingyang Wu, Lei Li, and Zhou Yu. 2020. TEXTGAIL: Generative Adversarial Imitation Learning for Text Generation. *arXiv preprint arXiv:2004.13796*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial Feature Matching for Text Generation. In *Proceedings of the 34th International Conference on Machine Learning*, pages 4006–4015.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. 2018. Adversarially Regularized Autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5902–5911.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A Benchmarking Platform for Text Generation Models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

A Reproducibility Checklist

- **Description of Computing Infrastructure** Tesla V100S-PCIE-32GB
- **Average Runtime**

MODEL	UG		CG
DATASET	MSCOCO	WMTNEWS	ROCSTORY
TARAE	5.8	9.2	-
ENHANCED	7.5	11	-
LOCALD	8	12	-
TILGAN	8.5	13.4	106

Table 5: The average runtime per epoch for each model, estimated in minutes. UG and CG refer to unconditional generation and conditional generation, respectively.

- **Number of Parameters**

MODEL	UG	CG
TILGAN	25.4M	30.8M

Table 6: The number of parameters of each model. UG and CG refer to unconditional generation and conditional generation, respectively.

- **Validation Performance** No validation evaluation for unconditional generation and conditional generation tasks.
- **Number of Runs** We conduct 60 runs for unconditional generation tasks and 30 runs for conditional generation tasks.
- **Bounds and Best Setting for Hyperparameters** Please refer to Table 3 for unconditional generation task and Table 4 for conditional generation task.

	MSCOCO		WMTNews	
	Bound	Best-performing	Bound	Best-performing
max len	15	15	32	32
batch size	[32,256]	256	[32,256]	256
emb size	[256,1024]	512	[256,1024]	512
hidden size	[256,1024]	512	[256,1024]	512
num layers	[1,6]	2	[1,6]	2
num heads	4	4	4	4
squeezed hidden size	[28,256]	56	[28,256]	56
noise size	[50,512]	100	[50,512]	100
niters autoencoder	[1,3]	1	[1,3]	1
niters discriminator	[1,3]	1	[1,3]	1
niters enhanceD	[1,3]	1	[1,3]	1
niters generator	[1,3]	1	[1,3]	1
niters gan into encoder	[1,3]	1	[1,3]	1
learning rate autoencoder	[0.01,10]	0.08	[0.01,10]	0.24
learning rate gan encoder	[1e-5,1e-2]	1e-4	[1e-5,1e-2]	1e-4
learning rate generator	[1e-5,1e-2]	1e-4	[1e-5,1e-2]	1e-4
learning rate discriminator	[1e-5,1e-2]	1e-4	[1e-5,1e-2]	1e-4

Table 7: The bounds for each hyperparameter and best-performing setting for unconditional generation task.

	ROCStory	
	Bound	Best-performing
num layers	[1,8]	6
hidden size	[256,1024]	512
num heads	[4,12]	8
emb size	300	300
latent dimension	[32,256]	64
batch size	[32,128]	64
learning rate	[1e-5,1e-2]	1e-4
droupout rate	[0,0.5]	0.15

Table 8: The bounds for each hyperparameter and best-performing setting for conditional generation task.

B Proof of Theorem 1

Proof of Theorem 1. In this proof, we let $\boldsymbol{x}$ be the real data, $\boldsymbol{y}$ be the generated data, and $\boldsymbol{z}$ be the latent variable. Let $p_G(\boldsymbol{y},\boldsymbol{z}) = p_\beta(\boldsymbol{z})p_\theta(\boldsymbol{y}|\boldsymbol{z})$ be the joint distribution of $(\boldsymbol{y},\boldsymbol{z})$, where $\boldsymbol{z}$ is sampled from prior $p_\beta(\boldsymbol{z})$ and then $\boldsymbol{y}$ is sampled from the decoder conditional $p_\theta(\boldsymbol{y}|\boldsymbol{z})$. Further let $\mathcal{P}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}$ denote the set of all joint distributions of $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ such that $\boldsymbol{x} \sim p_r(\boldsymbol{x})$, $(\boldsymbol{y},\boldsymbol{z}) \sim p_G(\boldsymbol{y},\boldsymbol{z})$, and $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y}|\boldsymbol{z}$; let $\mathcal{P}_{\boldsymbol{x},\boldsymbol{z}}$ be the set of marginal distributions of $(\boldsymbol{x},\boldsymbol{z})$ induced by $\mathcal{P}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}$, that is, the set of distributions with marginals $\boldsymbol{x} \sim p_r(\boldsymbol{x})$ and $\boldsymbol{z} \sim p_\beta(\boldsymbol{z})$.

Recall that n is the sequence length and m is the number of words in the vocabulary. For the i-th word x_i which is an m-dimensional one-hot vector, indicator $\mathbf{1}(x_i = j) = 1$ if the j-th dimension of x_i is equal to 1 and $\mathbf{1}(x_i = j) = 0$ otherwise, for $j = 1, \ldots, m$. Then we have

$$\begin{aligned}
W_1(\mathbb{P}_r,\mathbb{P}_G) \le W_1^\dagger(\mathbb{P}_r,\mathbb{P}_G) &:= \inf_{p\in\mathcal{P}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}} \mathbb{E}_{\boldsymbol{z}\sim p_\beta(\boldsymbol{z})}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{z})}\mathbb{E}_{\boldsymbol{y}\sim p_\theta(\boldsymbol{y}|\boldsymbol{z})}[c(\boldsymbol{x},\boldsymbol{y})] \\
&= \inf_{p\in\mathcal{P}_{\boldsymbol{x},\boldsymbol{z}}} \mathbb{E}_{\boldsymbol{z}\sim p_\beta(\boldsymbol{z})}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{z})}\left[2\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}(x_i=j)(1-\bar{G}_{ij}(\boldsymbol{z}))\right] \\
&< \inf_{p\in\mathcal{P}_{\boldsymbol{x},\boldsymbol{z}}} \mathbb{E}_{\boldsymbol{z}\sim p_\beta(\boldsymbol{z})}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{z})}\left[-2\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}(x_i=j)\ln\bar{G}_{ij}(\boldsymbol{z})\right] \\
&= \inf_{q(\boldsymbol{z}|\boldsymbol{x}):q_z(\boldsymbol{z})=p_\beta(\boldsymbol{z})} \mathbb{E}_{\boldsymbol{x}\sim p_r}\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}\left[-2\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}(x_i=j)\ln\bar{G}_{ij}(\boldsymbol{z})\right] \\
&= \inf_{q(\boldsymbol{z}|\boldsymbol{x}):q_z(\boldsymbol{z})=p_\beta(\boldsymbol{z})} \left\{-2\mathbb{E}_{\boldsymbol{x}\sim p_r}\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})]\right\},
\end{aligned}$$

where the first inequality comes from Tolstikhin et al. (2018, eq. 9), and the second inequality is due to the fact that $1 - l < -\ln l$ for all $l \in (0,1)$, leading to the desired result. $\square$

C Generated Examples

Table 9: Generated samples on ROCStory dataset.

story:	____________________. when i got to the stop sign , the check engine light started flashing . i panicked and carefully drove the van to the nearest mechanic shop . they checked it out but could not repair the van . the van had to be sold for parts and i had to get a new vehicle .
T-CVAE:	i was driving my van down the street one day .
TILGAN:	i was driving my van to work one day .
story:	krista was organizing her office .____________________ . they were big and heavy . she assembled them carefully . when she put all her books on them , they collapsed !
T-CVAE:	she had a bunch of books .
TILGAN:	she bought some new books .
story:	the man won a contest . he went to the station to collect . ____________________ . he did n't really like the band . he tried to sell them back to the radio employees .
T-CVAE:	he got a ticket for a band .
TILGAN:	he saw some band members .
story:	billy is bored . billy sits with his friends thinking of something to do . billy suggest they all head to the lake to go fishing . ____________________ . billy takes his friends to go fishing and has great time .
T-CVAE:	billy and his friends go fishing together .
TILGAN:	billy and his friends go fishing .
story:	____________________ . her house was full of dust . she could n't believe how filthy it was . alicia then decided to clean it . when she was done cleaning and it sparkled .
T-CVAE:	alicia was in the basement .
TILGAN:	alicia was cleaning her house .

Table 10: Generated samples on MSCOCO dataset.

TextGAN: - a train traveling down a street . - a train station . - a street sign on a street . - is in a bathroom with a sink controls .	**SeqGAN**: - a red stop sign . - a couple of people are walking on a log and trees . - the train car traveling mannequin driving down the tracks . - people standing next to a large building .
RankGAN: - a blue blue train sits on tracks with his residential asian toys . - a reflection of two birds walking by a sidewalk . - a man tourist train in the egret - a white fire hydrant stands next to each other .	**MLE**: - an orange booth contains the in traffic light under a sign - a man with hat on the horse in the street - a couple walking around city tracks with people - the bird are walking next to a small blue coop
LeakGAN: - a table topped with pots . . - a bathroom with a glass shower , sink , toilet and sink . - a woman wearing a glass is sitting on a cupboard . - a group of men talking .	**FM-GAN**: - a man is standing on a skateboard on the beach - a man on a tennis game with a kite - a man on a table with a red and white and a building - a man is standing on a table with a dog
ARAE: - a city street sign in the park bench parked in the group group the man - two people standing at motorcycles on the bench - two people standing at motorcycles on the bench at white kitchen - there is a city bus on their city street sign parked in blue blue bus - a white plane on the air plane parked in snow group their plane parked	**TILGAN**: - a little boy sitting on a bench with a little girl - a group of people in the middle of a field - a large passenger plane flying through the sky - a woman is sitting in a kitchen next to a restaurant - a small bird sitting on a branch of a tree

Table 11: Generated samples on EMNLP WMT dataset.

SeqGAN: - it said the yield on our most traveled to china ’ s capital for “ the annual bank of cost credit against cuba . - the cars (, taiwan argues that the cease - fire contained included already reported on the outlook . - “ both the republican leader in the years of leadership she was in rome and signed a close cabinet , ” she said . - russia has said in its twitter documents since january which demanded that lawmakers had clearly been involved about .
RankGAN: - the lakers left the us to hope of the office and chose the general administration to build a further mission . - ms . bush had been remembered in a red ring after it taking these sunday week made focusing himself against the number of games . - “ “ i ’ d thought something i am running everything i saw my own american life usually respect at all , as some equipment they have that . - and i was hoping that management does still even bring to hillary carson , and listen to a guy playing off back .
MLE: - what we need do this case if anybody had now touched a in - town community , all your kids in exchange barriers are needed in , no more , all may come out the coming in reflect the options . - so the scottish government is significant until not extension you own very so hard to change my job for your six points at half , social opinion and take beyond . - us president - elect donald trump will re consider an effort to set out that it would be to accept from the us - city solution to the world . - local judges ask for her children and went into a video itself , she said for 2016 ’ s early next day , he said .
LeakGAN: - picture west eight my might confidence , zero confidence my either nazi a a time having accounts , skills a difference x having must difference time having a develop pakistan confidence time time killed wilson partners nazi unfair zero phones develop vital confidence a might showed a having confidence develop a - pupils evidence accounts having confidence confidence theft abortion time time sized time west coming a unfair time affecting time my theft a a killed killed phones , , time questioned pakistan a partners evidence sized confidence unfair my eight time pakistan zero zero confidence partners either seventh having , killed a
GsGAN: - i hope that i do something like that it ’ s a very important thing , i know what you want to see this i didn ’ t know , i think it does not be able to work out this way that ’ s not a lot better needs to - the actor is it , well , which was a good job in a writing - christmas time out of a three - year - old woman who had been charged for the murder , he said that he was investigating the government ’ s decision to 19 . 6 million - to give the first time , it added , he had a few days and now he ’ s not just a new administration , he will do the same time before . - it is that , but the two - year - old woman he didn ’ t agree on : they had been a right ago because i wanted to have to do something i was trying to kill them , i am , but i can do it had to stay
FM-GAN: - The United States , the United States has been a major group of the United States in the United States , the United States in the United States . - We have to be able to pay the money to be able to pay the money to be able to pay the money to pay for the same time ," he said . - " It ’ s a lot of people who have been a woman , and I have been told the police ," she said . - The man ’ s death was a " bit of the incident , but the police said that the police had been taken to the city , and the police officer was a " very dangerous - driving area . - We have to be able to do the government to be able to do the government to be able to leave the country ," he said .
ARAE: - a more . 5 per cent the company said it would not be expected to rise if he hit the 2 percent year , it said , rose 2 , 500 ,when the only reason only be the best way for the best time for the best time for the best time for them and not being able to have done with much - the fact : the fact only now not being able to have done with a much more time for the age amount time with a much time for the best time for - the fact the only reason only be the best way . - the fact : the fact only now being a more person with a person with each person with a much time with the best time for them as much as a person
TILGAN: - many people who died , although they didn ’ t have been on the same day , not just because of those who had been out of them . - " i had to be able to get a good deal with the right time ," he said in a statement . - that ’ s why , in my life is now that ’ s not the same thing , and how much money is . - we are still working closely with the community who is still in the world , but we can ’ t be the best . - we can ’ t get some good players in the league , but not only because we ’ ve played well .

John praised Mary because *he*?
Implicit Causality Bias and Its Interaction with Explicit Cues in LMs

Yova Kementchedjhieva
University of Copenhagen
`yova@di.ku.dk`

Mark Anderson
Universidade da Coruña, CITIC
Department of CS & IT
`m.anderson@udc.es`

Anders Søgaard
University of Copenhagen
`soegaard@di.ku.dk`

Abstract

Some interpersonal verbs can implicitly attribute causality to either their subject or their object and are therefore said to carry an implicit causality (IC) bias. Through this bias, causal links can be inferred from a narrative, aiding language comprehension. We investigate whether pre-trained language models (PLMs) encode IC bias and use it at inference time. We find that to be the case, albeit to different degrees, for three distinct PLM architectures. However, causes do not always need to be implicit—when a cause is explicitly stated in a subordinate clause, an incongruent IC bias associated with the verb in the main clause leads to a delay in human processing. We hypothesize that the temporary challenge humans face in integrating the two contradicting signals, one from the lexical semantics of the verb, one from the sentence-level semantics, would be reflected in higher error rates for models on tasks dependent on causal links. The results of our study lend support to this hypothesis, suggesting that PLMs tend to prioritize lexical patterns over higher-order signals.

1 Introduction

Recognising causal links in narrative is an integral component of language comprehension that often relies on implicit cues (Trabasso and Sperry, 1985). Pre-trained language models, which form the basis of many language processing solutions nowadays, should therefore pick up on such cues and integrate them correctly with other signals to enable accurate causal inferences in downstream tasks, including question answering and information extraction.

Psycholinguists have identified one such cue in the implicit causality bias of interpersonal verbs: some interpersonal verbs tend to implicate causality on either their subject or their object (Garvey and Caramazza, 1974). It is this bias that leads to wide agreement among subjects in psycholinguistic studies, that in a sentence like *John appreciates Mary*, the cause for appreciation likely lies with a property or action of Mary's rather than John's. Causality can also be stated explicitly, in the form of a subordinate *because* clause, for instance, and it can optionally contradict the implicit causality of the verb in the main clause. In Figure 1, we show two verbs in the context of explicit statements of causality (EC) that are either congruent or incongruent with their IC bias. Psycholinguistic studies show that congruency affects language comprehension, with human participants taking longer to identify the referent to the pronoun after *because* in incongruent contexts compared to congruent ones (Caramazza et al., 1977). The integration of the two signals costs humans extra effort, but they are eventually able to overcome the false initial expectation based on lexical semantics (i.e. IC) and form a final response that takes into consideration the full sentence-level semantics (i.e. EC). Ettinger (2020) suggests that in the context of such diverging signals, models would likely fail to integrate all signals correctly, producing a response that is consistent with the initial, shallow expectation and therefore incorrect.

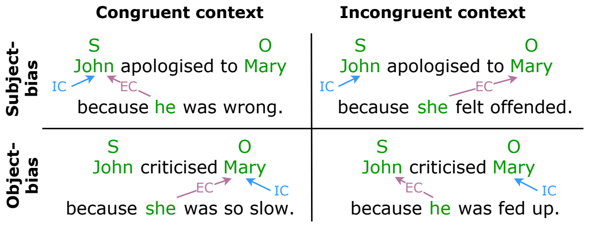

Figure 1: Illustration of implicit causality (IC) and explicit causality (EC) in contexts where the two are congruent, and where they are incongruent.

In this work, we study a range of large transformer-based pre-trained language models (PLMs) with a focus on their awareness of IC bias

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4859–4871
August 1–6, 2021.

and their response to stimuli of the kind shown in Figure 1. Following Ettinger (2020), we hypothesize that a language model aware of IC bias would experience interference from this signal in the context of incongruent EC, resulting in errors of judgement on a co-reference resolution task dependent on causal inference.

In a study of six PLMs from three model families: unidirectional generative, bidirectional generative and bidirectional discriminative, we find that IC bias is reliably encoded by all, but not used to an equal degree when making predictions in a controlled setting designed to test for IC awareness. In line with our hypothesis, we find that models with high IC awareness suffer an interference from IC bias in contexts of incongruent causality. We discuss these findings with reference to model type, size and amount of training data; we also draw general conclusions about the shortcomings of language models, which seem to prioritise a low-level lexical pattern (when they are aware of it in the first place) over a higher-order contextual signal.

2 Related Work

The study of the linguistic capacities of neural language models (LMs) has become especially relevant in current NLP research, where representations from PLMs feed into systems for various complex tasks, typically improving performance. Many of the testing paradigms used in psycholinguistics lend themselves well to LM analysis as they rely on a textual stimulus and a lexical response.

Linzen et al. (2016) were first to borrow from the psycholinguistic testing paradigm, in a study of the capabilities of LSTM-based models to resolve subject-verb number agreement. Goldberg (2019) adopted the psycholinguistic approach in an assessment of BERT (Devlin et al., 2019) on a number of syntactic tasks and found it to perform remarkably well on all. Hawkins et al. (2020) studied the ability of different LMs to capture human preferences as to the argument structure of English verbs.

The analysis of semantic capabilities in LMs includes studies on negative polarity in LSTM LMs (Marvin and Linzen, 2018; Jumelet and Hupkes, 2018), reasoning based on higher-order linguistic skill (Talmor et al., 2019), arithmetic and compositional semantics (Staliūnaitė and Iacobacci, 2020), stereotypic tacit assumptions and lexical priming (Misra et al., 2020; Weir et al., 2020). Many of these studies look at recent PLMs and draw mixed conclusions about the level of semantics encoded by these models. Peters et al. (2018) and Tenney et al. (2019) observed that PLMs do encode some higher-order syntactic abstractions in the higher layers (whereas lower-order syntactic information is encoded in the lower layers). However, in a comparison of contextualized and static word embeddings, Tenney et al. (2019) concluded that PLMs do not generally offer the same improvement with respect to semantics as they do for syntax.

At the crossroad of semantic analysis and psycholinguistic approaches, Ettinger (2020) introduced a suite of six psycholinguistic diagnostics for the analysis of semantic awareness in LMs. The tasks were selected based on a specific pattern observed in the response of human participants in psycholinguistic studies: an initial expectation (marked by an N400 electrophysiological response) that diverges from the final answer in a cloze task that humans offer once they have had time to fully consider the test prompt. Ettinger (2020) suggests that LMs might be “tripped up” in such contexts if they are unable to accurately integrate all the available information—she indeed found that to be the case for role-based event prediction in BERT (Devlin et al., 2019), for example. The phenomenon we study, incongruency in causality signals, has been observed to trigger a similar response in humans (Van Berkum et al., 2007) and can thus be expected to also “trip up” LMs.

Implicit causality bias was previously considered in PLM analysis by two works, both looking at how well unidirectional PLMs capture it. Upadhye et al. (2020) studied IC from the perspective of how different connectives between the main clause and the following clause (*because*, *and as a result*, full stop) affect the strength of the bias. While they did not find strong evidence for a correlation to human-based results in this respect, they did observe that in the context of connective *because* PLMs assigned lower probability to subject-referring pronouns for an object-biasing verb as compared to a subject-biasing verb. Davis and van Schijndel (2020) observed that GPT2-XL (Radford et al., 2019) encodes some level of IC bias in its representations (measured in terms of similarity between the representation of the pronoun and its two potential referents) and its decision on how to resolve a referent at prediction time is weakly influenced by that. They took the analysis one step further and looked at whether GPT2-XL uses IC information to resolve relative clause attachment, which in humans is conditioned by IC bias—no evidence was found to suggest that that was the case. Our study extends previous work on IC bias in several ways: we study both unidirectional and bidirectional models, we measure bias in the same terms as was done in psycholinguistic work and can therefore assess the correlation between the two, and we study the matter from a perspective that has not been considered before, namely the case of incongruent explicit and implicit causality.

3 Materials

Here we describe the two psycholinguistic diagnostics that we draw on: for the study of IC bias in isolation and its integration with EC. We also describe the modifications necessary to make these diagnostics suitable for PLMs.

3.1 Context-free IC Bias

Ferstl et al. (2011) studied IC bias in a context free of EC through a sentence completion task where subjects were presented with a stimulus like

(1) John praised Mary because ___

and asked to finish the sentence. Continuations were observed to start with a third person pronoun (*he* or *she*) 94.2% of the time. The researchers counted the ratio of continuations referring back to the subject of the sentence, s_{wins}, and back to the object, o_{wins}, and computed a bias score for each verb as $100 \times (s_{wins} - o_{wins})/(s_{wins} + o_{wins})$. This results in a range of -100 (verbs with extreme Object bias, hereafter O-bias) to 100 (verbs with extreme Subject bias, hereafter S-bias). The study covered 305 interpersonal English verbs with responses from 96 subjects.

In another study of IC bias, Hartshorne and Snedeker (2013) presented subjects with stimuli with a nonce ending, e.g.

(2) John praised Tim because he was a *dax*.

and asked them the question *Who do you think is a dax?* The nonce ending is meant to provide a content-free continuation that does not affect the interpretation of the ambiguous pronoun, neither semantically (as the madeup *dax* carries no meaning), nor syntactically—Hartshorne et al. (2015) conclude that explanations of the form *is a/an X* do not affect people's intuitions about who the explanation referred to.

Our approach is to use the procedure of Ferstl et al. (2011) as is to test unidirectional PLMs, as they are naturally suited to the open-end input format. Since bidirectional PLMs have been trained on complete utterances and may thus act unpredictably in an open-end context,[1] we test such models with a modification of the procedure of Hartshorne and Snedeker (2013):[2] we convert it into a cloze task with a gender mismatch between the participants, such that (3) becomes

(3) John praised Mary because ___ was a *dax*.

We adopt the mismatched-gender setting as it more closely resembles the sentence completion task in Ferstl et al. (2011). In both formats we can now identify the preferred referent by looking at the probability of pronouns *he* and *she* for the empty slot, each one referring unambiguously to only one referent. Inducing a prediction for a pronoun in the empty slot is also a more natural choice of co-reference than repeating one of the names (Holtzman et al., 2019).

In the examples shown throughout the paper, *John* and *Mary* are used as placeholders for the subject and object of the verb of interest. The choice of names to go in these slots can affect model predictions (Abdou et al., 2020), so we generate 200 variants of each stimulus, varying the names and the order between the two genders and we query the PLMs with all of them. The full procedure is described in Appendix B.

We compile a list of 200 nonce words using the 194 nonce words made available by Bangert et al. (2012), five nonce words from Cuskley et al. (2015), manually chosen to resemble English nouns, and *dax*, used in Hartshorne and Snedeker (2013). When presenting a bidirectional PLM with the aforementioned 200 variants of a stimulus, we dynamically draw a nonce word at random from this list without replacement.

The procedure described above is applied to each of the 305 verbs studied in Ferstl et al. (2011).

3.2 IC Bias in the Context of EC

Caramazza et al. (1977) tested the effect of incongruency between IC and EC using pairs of sentences built around the same verb, where one contains an explanation congruent with the verb's bias

---

[1] We find that a common response of generative bidirectional PLMs to stimuli like those in (1) is to predict a full-stop for the empty slot.

[2] See Appendix A.1 for an alternative we considered.

and the other contains an incongruent explanation (refer back to Figure 1 for some examples). Participants were shown one sentence at a time on a screen and asked to say out loud who the referent was to the pronoun after *because*. Caramazza et al. (1977) carried out experiments both with stimuli where the referents are of the opposite gender and where the referents are of the same gender—responses were delayed in the context of incongruent explanations as compared to congruent ones in both settings, the effect being stronger in the mismatched-gender setting.

Our approach is to adopt the mismatched-gender setting and to convert this task into a cloze task as well, an example stimulus being:

(4) John praised Mary because ___ had done well.

The stimuli used by Caramazza et al. (1977) and other related studies like Garnham et al. (1996) use only a handful of verbs (14 and 22, respectively). We therefore found it necessary to develop a more expansive dataset for the purposes of our study. Following the procedure described in Appendix C we constructed pairs of subject-referring explanations and object-referring explanations for 99 verbs, 33 strongly subject-biased verbs (bias$>$ 65), 33 strongly object-biased verbs (bias< -65), and 33 verbs from the middle of the scale, which can be thought of as having no effect on the attribution of implicit cause. Selecting the verbs in this fashion, with large gaps between each group, allows us to see the difference between them most clearly.

Similarly to before, 200 variants of each stimulus are generated, varying the names of the referents and the order between the two genders.

4 Procedure

In this section we describe how we induce responses to the tasks described in §3 for the two experiments in this study: measuring context-free IC bias and IC bias in the context of EC. Six English PLMs are considered in this study, representative of the unidirectional generative, bidirectional generative and bidirectional discriminative paradigms in language modeling. As seen in Table 1, GPT, BERT and ELECTRA are comparable in size and training data. GPT2-M, RoBERTa-L and ELECTRA-L are included as the 'bigger siblings' to the former three models respectively, selected to resemble closely the architecture of their counterparts, while having a larger size and richer training data.[3] Comparisons can therefore be made across the three base models, on one hand, and within each pair of a base model and its larger counterpart, on the other. The three larger models are comparable in size, but not fully comparable in training data, GPT2-M being trained on only a quarter of what the other two models are trained on. In a small multilingual experiment, we also experiment with German BERT, Spanish BERT and mBERT,[4] a multilingual version of BERT.

	Work	Size	Data	Dir	Obj
English					
GPT	Radford et al. (2018)	110	16	Uni	Gen
GPT2-M	Radford et al. (2019)	345	40	Uni	Gen
BERT	Devlin et al. (2019)	110	16	Bi	Gen
RoBERTa-L	Liu et al. (2019)	355	160	Bi	Gen
ELECTRA	Clark et al. (2020)	110	16	Bi	Disc
ELECTRA-L	—"—	335	160	Bi	Disc
German BERT	-	110	12	Bi	Gen
Spanish BERT	Cañete et al. (2020)	110	20	Bi	Gen
mBERT	-	110	194	Bi	Gen

Table 1: Model properties in terms of size (number of parameters in millions), training data (size in GB), directionality (uni- or bi-directional), and token-level training objective (generative or discriminative). German BERT and mBERT are not the product of any published work, but are closely associated with Devlin et al. (2019).

We first describe the procedure for bidirectional generative PLMs which is most straightforward. Both experimental tasks can be formulated as a cloze task (see Examples (3) and (4)). We place a mask tag in the empty slot, pass the input through the model and compute the probability that the models assigns to tokens *he* and *she* for the position of the mask tag.

The procedure is equally trivial when testing unidirectional PLMs for context-free IC bias effects, where the stimuli can naturally take on an open-ended form (see Example (1)). The partial sentence is passed through the model and a probability for the relevant pronouns is computed. Measuring the effect of IC bias in the context of EC, on the other hand, cannot be performed in the next-word prediction paradigm, so for this task we instead use the unidirectional PLMs as language scorers: we

---

[3] For implementational details see Appendix D.

[4] The size of the training data for mBERT is not exactly known—it consists of the 100 biggest Wikipedias. English Wikipedia, as made available on HuggingFace in 2019 is 14 GB in size; with the next 99 Wikipedias being 13% the size of the English Wikipedia on average (based on number of articles), that works out to an estimate of 194GB in total.

Model	Exp	p(he) =
GPT(2)	IC	p(he $\|w_1...w_4$)
	IC+EC	p($w_1...w_4$ he $w_6...w_8$)
(RO)BERT(a)	both	p(MASK==he $\|w_1...w_4$ MASK $w_6...w_8$)
ELECTRA	both	p(class=O $\|w_1...w_4$ he $w_6...w_8$)

Table 2: Probing PLMs for the effect of context-free IC (IC) and IC in the context of EC (IC+EC) experimental paradigm. $w_1...w_4$ = *John praised Mary because*; $w_6...w_8$ = *was a dax / had done well* for IC and IC+EC, resp. The procedure is analogous for *she*.

create two versions of each stimulus, one with pronoun *she*, one with pronoun *he* in the empty slot, and obtain a probability for each as the average over the probabilities of all tokens in the sequence.

The discriminative model ELECTRA is trained to recognize replaced tokens in its input, i.e. for each token it computes a probability over two classes, *replaced* (R) and *original* (O). Based on the reasoning outlined in Appendix E we conclude that the more appropriate way to probe ELECTRA in our experiments is by taking the average over the probability of class O for all tokens in a sequence, instead of looking at the probability of this class for the pronoun of interest alone. In a procedure similar to the one used for GPT2-M when used as a language scorer, we present ELECTRA with two versions of each stimulus , one with *he*, one with *she* in the slot of interest, and we compute the average probability of class O for each. We formalize the handling of all model types in Table 2.

5 Exp. 1: Context-free IC Bias

With this experiment we want to determine whether English PLMs exhibit the same tendencies as humans when it comes to the IC bias of actions/states expressed with interpersonal verbs in a context free of any explicit causes. To this end, we use the materials described in §3.1 and the procedure described in §4. For any given model, the IC bias per verb is measured over the responses of the model to the set of 200 stimuli, each response processed as follows.

5.1 Measuring Bias

For a sentence with a female subject and a male object, the probability of *she* would be denoted as p^s, the probability of *he* as p^o, and $p^s > p^o$ would indicate a preference for the subject for this stimulus. Refer back to Table 2 for a summary of how these probabilities are obtained with each model.

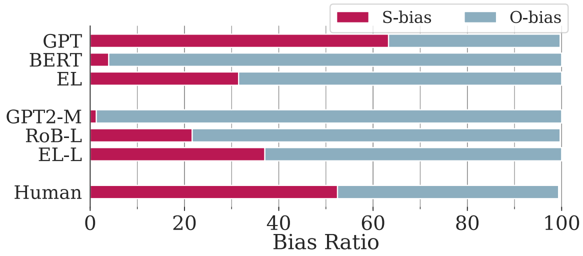

Figure 2: Ratio of S-bias verbs and O-bias verbs.

Having obtained the values p^s and p^o for each of the 200 stimuli per verb, we next calculate the bias of this verb in the following manner:

$$s_{wins} = \sum_{n \in N} (p^s_n - p^o_n) > 0 \tag{1}$$

$$o_{wins} = \sum_{n \in N} (p^s_n - p^o_n) < 0 \tag{2}$$

$$bias = 100 \times \left(\frac{s_{wins} - o_{wins}}{s_{wins} + o_{wins}} \right) \tag{3}$$

where N is the set of 200 stimulus variants per verb. This metric gives us a range from -100 (extreme O-bias) to 100 (extreme S-bias), with 0 indicating an absence of any bias altogether.

5.2 Preliminary analysis

As validation of the experimental procedure, we note that the generative models, BERT, ROBERTa-L, GPT, GPT2-M, ranked one of the two vocabulary items of interest, *he* or *she*, as their top prediction at a rate of 99.0, 99.4, 99.3 and 100.0 percent, respectively. It is reassuring to see that the models behave similarly to humans in this respect, who selected *he* or *she* at a rate of 94.2% as a first token after *because* (Ferstl et al., 2011). That also indicates that the probabilities assigned to the two tokens are meaningful (Holtzman et al., 2019).

Figure 2 shows the ratio of S-bias verbs and O-bias verbs as determined by each of the models and by humans. Compared to human IC bias scores, which give an even distribution of verbs across the S-bias and O-bias classes, we see that most models show an imbalance in this respect–most notably, BERT and GPT2-M show an almost categorical preference for the object of the sentence. These trends could relate to the syntactic role of the participants (subject v. object), to their linear order, or to referent proximity. The first two factors are difficult to decouple in English, a language with a relatively fixed subject-verb-object word order. We discuss the effect of proximity in Appendix A.2.

To gain a clearer understanding of the IC bias awareness of the different models, we analyze the results of this experiment in their raw form and also with discounting for other potential sources of bias. In addition to the object bias discussed above, we include gender and choice of nonce words, which on their own did not appear to have a strong effect, but could combine with each other and with the object bias in unpredictable ways. The discounting for *p(he)*, for example, in the context of stimulus *John apologized to Mary because _ was a dax* is done by subtracting the average probability of pronoun *he* in the context of any stimulus with a male subject and word *dax* in the nonce word slot (152.5 data points on average).

5.3 Results

Table 3 quantifies the correspondence between model IC bias and human IC bias in terms of Spearman's ρ over bias scores and in terms of micro-averaged F1 score over the polarity of the IC bias (subject-bias v. object-bias). For a plot of the exact bias values see Figure 6 in Appendix F. The PLMs most affected by the discounting for other biases are BERT and GPT2-M, which also showed the strongest imbalance as observed in Figure 2. All PLMs show a significant correlation to human IC bias, although this observation has the caveat of a small dataset (only 305 data points).

Within the pairs of related models, we can say that the differences between BERT and RoBERTa-L on the one hand and ELECTRA and ELECTRA-L on the other, are small, which suggests that already with 16GB of training data and 110M parameters, these architectures reach their potential in terms of capturing and using IC bias. For the two unidirectional PLMs, we see that after discounting GPT2-M exhibits a considerably higher correlation to human IC bias scores. This may indicate that the larger size and/or richer training data of GPT2-M have enabled the model to better capture IC bias, although the correlation still remains low in absolute terms.

Comparing unidirectional PLMs to bidirectional ones, we find that the latter obtain a stronger correlation to humans scores. A similar trend holds for the F1 scores, where bidirectional models show a greater awareness of the polarity of the IC bias of verbs (especially after discounting). We refrain from making comparisons across model architectures beyond the uni- v. bidirectional dichotomy, to avoid drawing false conclusions: as we are using different procedures to induce a response from generative and discriminative models, it could be argued that a direct comparison is not methodologically robust. The discriminative models are making a binary decision over two options predefined by us, while the generative models are computing a probability distribution over hundreds of thousands of vocabulary items.

	BERT	RoB-L	GPT	GPT2-M	EL	EL-L
ρ	0.58*	0.67*	0.22*	0.22*	0.72*	0.72*
F1	0.508	0.672	0.607	0.482	0.744	0.754
ρ	0.65*	0.69*	0.23*	0.38*	0.73*	0.71*
F1	0.698	0.734	0.564	0.649	0.774	0.748
LDA	0.67*	0.58*	0.64*	0.46*	0.73*	0.67*
LR	0.71*	0.6*	0.67*	0.49*	0.75*	0.7*

Table 3: Correspondence between human- and model-induced IC bias scores (a) for model predictions, measured in terms of Spearman's ρ correlation over bias scores and F1-score over bias polarity before (rows 1 and 2) and after discounting (rows 3 and 4); and (b) for model representations (rows 5 and 6). * denotes significance at $p < 0.001$.

5.4 Further Analysis

To measure the models' sensitivity to IC bias in a perfectly comparable setting, we carry out an additional comparison on the level of representations, thus abstracting away from the top layers of the models where the differences ensue. We extract 'decontextualized' verb representation from the PLMs following the procedure described in Appendix G. Using those, we carry out two types of probes: an extrinsic one, where we train a linear regression model (LR) to map from a verb's representation to its IC bias; and an intrinsic one, where we use linear discriminant analysis (LDA) to identify the single dimension in the verb representations that is most informative of IC bias.[5] The benefit of the latter approach is that it does not add any newly trained parameters to the computation of the correlation (Torroba Hennigen et al., 2020). In both cases, the result is a vector of scalars (the values predicted by the LR, or the values of the selected dimension)—we measure the correlation between these values and human IC bias to determine how much of the latter can be recovered from the representations.

To reduce overfitting, which is inevitable with

---

[5] As LDA operates over a space of discrete labels, we convert the IC bias scores into 3 classes ($> 0, < 0, = 0$).

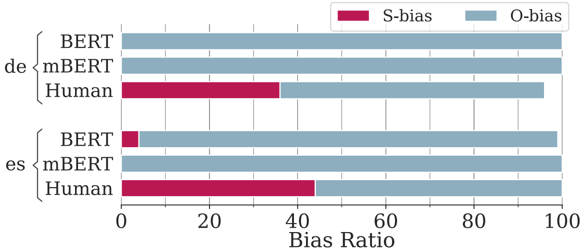

Figure 3: Ratio of S-bias and O-bias verbs in German (top) and Spanish (bottom).

305 datapoints in total and representations of 768 to 1024 dimensions, we apply PCA to the representations prior to fitting the LR and LDA models, reducing the representations to 5% of their original size. Each model (LR and LDA) is fit on a random 50% split of the data and applied on the other 50% to predict (LR) or transform (LDA). This procedure is repeated 100 times for robustness. The mean correlations are reported in the last two rows of Table 3. We see that larger models yield lower correlations than their smaller counterparts, suggesting that the former might encode IC bias in a more distributed manner than the latter.

Comparing models of equal size, we see that a similar pattern holds here as observed over the models' predictions, with the unidirectional models showing a lower correlation than the bidirectional ones, although the gap is substantially smaller in this space. It appears that unidirectional models might encode more IC bias than they exhibit at inference time. ELECTRA shows the highest correlation among the base-size models and ELECTRA-L the highest among the large models–as this comparison abstracts away from the specific objective each models uses (generative v. discriminative), we can conclude that the two ELECTRA models capture IC bias to the greatest extent out of the six PLMs studied here.

5.5 IC Bias in Other Languages

IC bias is not an English-specific phenomenon—Goikoetxea et al. (2008) obtained human judgements for 100 Spanish verbs, and Van den Hoven and Ferstl (2018) did so for 100 German verbs. Here, we probe Spanish (es) BERT, German (de) BERT and mBERT for their IC bias awareness. Details on the choice of proper nouns and nonce words are discussed in Appendix H. As seen in Figure 3 a recency/object bias is observed for the PLMs investigated here as well, so we present the results with and without discounting.

	German		Spanish	
	BERT	mBERT	BERT	mBERT
ρ	0.54*	0.23	0.13	-0.00
F1	0.600	0.600	0.540	0.560
ρ	0.51*	-0.13	0.16	-0.15
F1	0.680	0.380	0.610	0.360
LDA	0.26*	-0.0	0.09	0.08
LR	0.47*	0.02	0.12	0.03

Table 4: Correspondence between human- and model-induced IC bias in German and Spanish. For more details see the caption of Table 3.

Table 4 summarizes the results before (rows 1 and 2) and after discounting (rows 3 and 4). The poor performance of the multilingual mBERT is not surprising—Rönnqvist et al. (2019) found mBERT to be inferior to monolingual models at making a prediction for randomly masked subtokens (specifically looking at German, among other languages); and Vulić et al. (2020) found mBERT and XLM-R to both be inferior to their monolingual counterparts on probing tasks pertaining to lexical semantics.

German BERT shows a medium-strength correlation to human scores, whereas Spanish BERT shows no such correlation at all, both on the level of predictions and model representations. This observation could be attributed to the pro-drop nature of Spanish, wherein pronouns are often dropped when in subject position. This likely makes the learning of IC bias in Spanish harder for a PLM, as less evidence is available in the context to connect the explanation to its referent.

From this section, we conclude that English bidirectional PLMs reliably capture and use IC bias in their predictions. Unidirectional models encode IC bias but do not greatly rely on it at prediction time. Having established that IC bias affects the behavior of at least some PLMs, we now evaluate how these models integrate this implicit signal with more explicit signals from the sentence-level semantics.

6 Exp. 2: IC Bias in the Context of EC

With this experiment, we test the hypothesis that when the IC and EC signals converge in congruent contexts, i.e. they point to the same referent, the models would have more ease predicting the correct referent, whereas when the two signals diverge in incongruent contexts, the models would be more prone to errors. We test this hypothesis using the

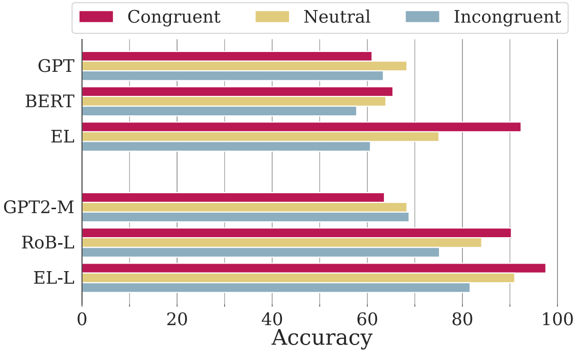

Figure 4: Accuracy on co-reference resolution over stimuli with congruent and incongruent IC and EC.

materials described in §3.2 and the procedure from §4. We present each stimulus to the models in 200 versions varied for subject and object referents. In this experiment, we do not perform discounting: unlike Experiment 1 where we wanted to gain as clear a view as possible of the level of IC bias that models exhibit, isolated from other sources of bias, here we want to see how IC bias interacts with EC, subject to any other potential sources of bias.

6.1 Results

Figure 4 shows the results from this experiment. All models are substantially better at resolving the antecedent correctly on average compared to a random baseline of 50%. Looking at the neutral stimuli as indication for the models' general ability to solve this task, we see that the two largest and most richly trained models, ROBERTa-L and ELECTRA-L, perform best. In line with our expectations, we see that most models score lower on resolving antecedents in incongruent contexts and higher in congruent ones. This is true for the four bidirectional PLMs, which also exhibited higher IC bias in Experiment 1. The gap is largest for ELECTRA and still substantial for ROBERTa-L and ELECTRA-L. The unidirectional models, on the other hand, show a noisier behavior, with a relatively small gap between the three types of stimuli, and an inconsistent ranking between them.

6.2 Discussion

As IC bias contributes to the construction of causal links in narrative and as such aids language comprehension (Trabasso and Sperry, 1985), it is desirable that PLMs capture and use this signal coming from the lexical semantics of interpersonal verbs. In Experiment 1, we found that all PLMs studied show a medium to high correlation with human IC bias scores on the level of representations, with bidirectional ones doing so on the level of predictions, too. While IC bias does contribute to language comprehension, it also has the unfortunate effect of interference in the context of incongruent EC. In this respect, models with higher IC bias awareness, i.e. the bidirectional PLMs in our study, suffer a greater drop in performance. Meanwhile, the unidirectional PLMs studied, which show little awareness of IC bias in a context free of EC, also show no interference from it when resolving referents in the presence of EC. Paradoxically, the superior performance of bidirectional PLMs with respect to IC bias also exposes a limitation of theirs: while these models are advanced enough to use IC bias for their predictions, their interpretation of semantics is still fairly shallow. The lower-order signal coming from lexical semantics is given priority over the higher-order signal coming from the sentence-level semantics. In the experiment presented in this section, this leads to a higher error rate on resolving pronoun antecedents in incongruent contexts, with potential impact on tasks that depend on co-reference resolution, e.g. document summarization (Azzam et al., 1999), question answering (Morton, 1999; Vicedo and Ferrández, 2000), and information extraction (Zelenko et al., 2004).

7 Conclusion

From the comparison of six competitive PLMs, BERT, ROBERTa-L, GPT, GPT2-M, ELECTRA and ELECTRA-L, we conclude that PLMs can exhibit IC bias much like humans do, but that different models do so to a different degree, with bidirectional models showing moderate to strong correlation to human judgements, and unidirectional models showing only a weak correlation. This ability of some PLMs has the unfortunate effect that it makes them prone to higher error rates in contexts of incongruent IC and EC signals, where the PLMs overly rely on IC bias. This finding adds to a growing body of evidence that PLMs prioritize lexical cues over higher-order semantic cues (cf. Tenney et al., 2019). As our hypothesis is inspired by the observation that humans experience a delay in the processing of incongruent contexts (Caramazza et al., 1977), our findings point to the potential of drawing further inspiration from such psycholinguistic phenomena in studying the behaviour of language models (Ettinger, 2020). Seeing that language models show a growing potential as off-the-shelf task solvers

(Radford et al., 2019; Brown et al., 2020), studying their predictions is an important avenue for better understanding their capabilities and limitations.

8 Acknowledgements

We thank Daniel Hershcovich, Ana Valeria González, Emanuele Bugliarello, and Mareike Hartmann for feedback on the drafts of this paper. We thank Desmond Elliott, Stella Frank and Dustin Wright, and Mareike Hartmann for their help with the annotation of the newly developed stimuli.

Yova was funded by Innovation Fund Denmark, under the AutoML4CS project. Mark received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150) and from the Centro de Investigación de Galicia (CITIC) which is funded by the Xunta de Galicia and the European Union (ERDF - Galicia 2014-2020 Program) by grant ED431G 2019/01.

References

Mostafa Abdou, Vinit Ravishankar, Maria Barrett, Yonatan Belinkov, Desmond Elliott, and Anders Søgaard. 2020. The sensitivity of language models and humans to Winograd schema perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7590–7604, Online. Association for Computational Linguistics.

Saliha Azzam, Kevin Humphreys, and Robert Gaizauskas. 1999. Using coreference chains for text summarization. In *Coreference and Its Applications*.

Ashley S Bangert, Richard A Abrams, and David A Balota. 2012. Reaching for words and nonwords: Interactive effects of word frequency and stimulus quality on the characteristics of reaching movements. *Psychonomic bulletin & review*, 19(3):513–520.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Myriam Cantú Sánchez. 2011. Nominal plural marking of nonce words in child Spanish. Master's thesis, Universitat Autònoma de Barcelona.

Alfonso Caramazza, Ellen Grober, Catherine Garvey, and Jack Yates. 1977. Comprehension of anaphoric pronouns. *Journal of verbal learning and verbal behavior*, 16(5):601–609.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. 2020. Spanish pre-trained BERT model and evaluation data. In *PML4DC at ICLR 2020*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Christine Cuskley, Francesca Colaiori, Claudio Castellano, Vittorio Loreto, Martina Pugliese, and Francesca Tria. 2015. The adoption of linguistic rules in native and non-native speakers: Evidence from a wug task. *Journal of Memory and Language*, 84:205 – 223.

Forrest Davis and Marten van Schijndel. 2020. Discourse structure interacts with reference but not syntax in neural language models. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 396–407, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pennylyn Dykstra-Pruim. 2003. L2 acquisition of German plurals: How students form them and textbooks teach them. *Die Unterrichtspraxis/Teaching German*, pages 43–55.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Evelyn C Ferstl, Alan Garnham, and Christina Manouilidou. 2011. Implicit causality bias in English: A corpus of 300 verbs. *Behavior Research Methods*, 43(1):124–135.

Alan Garnham, Matthew Traxler, Jane Oakhill, and Morton Ann Gernsbacher. 1996. The locus of implicit causality effects in comprehension. *Journal of memory and language*, 35(4):517–543.

Catherine Garvey and Alfonso Caramazza. 1974. Implicit causality in verbs. *Linguistic Inquiry*, 5(3):459–464.

Edurne Goikoetxea, Gema Pascual, and Joana Acha. 2008. Normative study of the implicit causality of 100 interpersonal verbs in Spanish. *Behavior Research Methods*, 40(3):760–772.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Joshua K Hartshorne, Timothy J O'Donnell, and Joshua B Tenenbaum. 2015. The causes and consequences explicit in verbs. *Language, cognition and neuroscience*, 30(6):716–734.

Joshua K Hartshorne and Jesse Snedeker. 2013. Verb argument structure predicts implicit causality: The advantages of finer-grained semantics. *Language and Cognitive Processes*, 28(10):1474–1508.

Robert Hawkins, Takateru Yamakoshi, Thomas Griffiths, and Adele Goldberg. 2020. Investigating representations of verb bias in neural language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4653–4663, Online. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Emiel Van den Hoven and Evelyn C Ferstl. 2018. Discourse context modulates the effect of implicit causality on rementions. *Language and Cognition*, 10(4):561–594.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? On the ability of LSTMs to understand negative polarity items. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4(1):521–535.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Kanishka Misra, Allyson Ettinger, and Julia Rayz. 2020. Exploring BERT's sensitivity to lexical cues using tests from semantic priming. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4625–4635, Online. Association for Computational Linguistics.

Thomas S. Morton. 1999. Using coreference for question answering. In *Coreference and Its Applications*.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Samuel Rönnqvist, Jenna Kanerva, Tapio Salakoski, and Filip Ginter. 2019. Is multilingual BERT fluent in language generation? In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 29–36, Turku, Finland. Linköping University Electronic Press.

Ieva Staliūnaitė and Ignacio Iacobacci. 2020. Compositional and lexical semantics in RoBERTa, BERT and DistilBERT: A case study on CoQA. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2019. oLMpics–On what language model pre-training captures. *arXiv preprint arXiv:1912.13283*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel Bowman, Dipanjan Das, et al. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019*.

Lucas Torroba Hennigen, Adina Williams, and Ryan Cotterell. 2020. Intrinsic probing through dimension selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 197–216, Online. Association for Computational Linguistics.

Tom Trabasso and Linda L Sperry. 1985. Causal relatedness and importance of story events. *Journal of Memory and language*, 24(5):595–611.

Shiva Upadhye, Leon Bergen, and Andrew Kehler. 2020. Predicting reference: What do language models learn about discourse models? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 977–982, Online. Association for Computational Linguistics.

Jos JA Van Berkum, Arnout W Koornneef, Marte Otten, and Mante S Nieuwland. 2007. Establishing reference in language comprehension: An electrophysiological perspective. *Brain research*, 1146:158–171.

José L. Vicedo and Antonio Ferrández. 2000. Importance of pronominal anaphora resolution in question answering systems. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 555–562, Hong Kong. Association for Computational Linguistics.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online. Association for Computational Linguistics.

Nathaniel Weir, Adam Poliak, and Benjamin Van Durme. 2020. Probing neural language models for human tacit assumptions. *Cognitive Science*. To appear.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Eugen Zaretsky, Hans-Helge Müller, and Benjamin P Lange. 2016. No default plural marker in Modern High German. *Italian journal of linguistics*, 28(2):203–230.

Dmitry Zelenko, Chinatsu Aone, and Jason Tibbetts. 2004. Coreference resolution for information extraction. In *Proceedings of the Conference on Reference Resolution and Its Applications*, pages 24–31, Barcelona, Spain. Association for Computational Linguistics.

A Alternative task formulation

A.1 Splitting the clauses

Hartshorne and Snedeker (2013) induce a response by presenting participants with a sentence like

(5) John praised Tim because he was a *dax*.

and asking them the question *Who do you think is a dax?* In the spirit of Radford et al. (2018) and Radford et al. (2019) we considered reformulating this into a task suitable for a language model as:

(6) John praised Tim because he was a *dax*. The one who was a *dax* was ___

This formulation would have allowed us to use unidirectional PLMs in a more natural way, for next-word prediction, since the token of interest now comes at the end of the sequence; and it would have been equally suitable for bidirectional PLMs. We performed experiments with it and found that models largely scored at chance level for the stimuli containing neutral verbs, which renders the results for the congruent and incongruent stimuli invalid. The difficulty PLMs faced in solving this task could stem from the more complex inference required and/or from the border-line unnatural structure of the inputs.

A.2 Swapping the clauses

Seeing that all models show some degree of object bias, we considered an alternative task formulation, where the main clause and the subordinate clause are swapped:

(7) Because ___ was a *dax*, John praised Mary.

In this formulation, the proximity changes for the two referents, such that now the subject is closer to the pronoun of interest. This clause-swapping can only be applied to bidirectional models for the purposes of measuring context-free IC bias as described in Section 3.1. Figure 5 shows the results obtained with this task formulation. Comparing these numbers to the ones presented in Figure 2, we see that all models show a more balanced distribution of verbs across the S-bias and O-bias classes, with the two ELECTRA models closely matching human scores. This suggests that reference proximity is indeed a factor in the choice of the pronoun. Still, we see that RoBERTa-L and especially BERT remain strongly biased towards the object of the sentence, meaning that proximity is not the only factor at play.

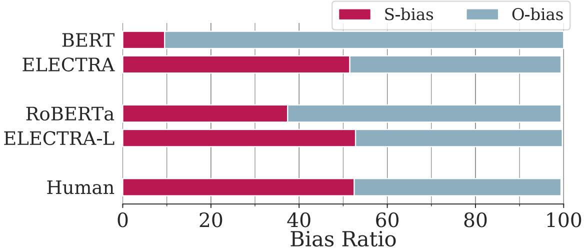

Figure 5: Ratio of S-bias verbs and O-bias verbs with swapped clauses.

Although this task formulation appears to lead to a reduced object bias across the four bidirectional models, we refrain from using it in our main experiments because it is not attested in psycholinguistic studies, i.e. it could have unforeseeable effects on human judgements of IC bias. As our main goal is to analyse the models' behavior *in relation to human behavior*, we follow closely the experimental protocol used in the available psycholinguistic studies on IC bias.

B Proper nouns

In this section we consider a seemingly minor but important consideration. Abdou et al. (2020) showed that model predictions on the Winograd Schema Challenge greatly vary with changes in the gender and identity of proper nouns used in the stimuli. We alleviate this issue by marginalizing over a range of proper nouns. To do so, we create multiple versions of the same stimulus with different proper noun combinations and use the average response of a model over all of the stimuli as an indication of the model's response to an abstract subject and object. We use 10 male names and 10 female names in 200 permutations. To ensure that the names in these lists are perceived as common names for the gender they represent, we used the models themselves to select the names, compiling a list of names unique to each model. We queried each model with the following sequences and took the top 10 names predicted:[6] *She is a woman and her name is* MASK. and *He is a man and his name is* MASK.

C Development of Materials

Neither Caramazza et al. (1977) nor Garnham et al. (1996) provide an explicit description of the procedure used to design their materials, so we extrapolate their methods by observing the materials themselves: the main goal in constructing a stimulus is to ensure that a particular ending is unambiguous (in a standard, most-likely reading) in pointing to exactly one of the two referents. The explanations always start with a verb in the past tense, e.g. *had done well*. And they are simple in the sense that they require little background knowledge. Using these observations as guidelines, we manually constructed pairs of congruent-incongruent contexts for 99 verbs, i.e. 198 stimuli in total. The materials were validated by three native English speakers and one fluent English speaker, who were asked to perform the cloze task on one stimulus from each pair and to mark ambiguous cases as such rather than making a guess at random. Eight contexts were judged as ambiguous and replaced with a better alternative, also validated in turn.

D Model implementations

We use the models as shared in the HuggingFace library (Wolf et al., 2019). In English experiments, we use bert-base-uncased, electra-large/base/small-discriminator, gpt2, and roberta-base. For German we use bert-base-german-cased, german-nlp-group/electra-base-german-uncased, dbmdz/german-gpt2, for Spanish dccuchile/bert-base-spanish-wwm-cased, and for both we use xlm-roberta-base and bert-base-multilingual-cased.

All experiments were run on a MacBook Pro, each taking between 1h (for smaller models) to 4h (for larger models).

E Probing ELECTRA

Consider the example sentence *The cake is very delicious* and a "corrupted" version of it we might present to ELECTRA: *The shoe is very delicious*. The model could give us a label sequence like {OROOO} for the latter, to indicate that shoes do not belong to the world of delicious things. And we could look at the probability distribution for the second token to quantify the strength of ELECTRA's objection to seeing this token in this position. Alternatively, ELECTRA could resolve the conflict with a label sequence like {OOOOR} to indicate that a taste-related adjective is not suitable for describing a shoe. In this case, looking at the probability distribution for the word *shoe* could be rather uninformative. This leads us to conclude that taking the average over the probability of label O for all tokens in a sequence is more informative than looking at the probability of this label for a single token.

F Experiment 1 visualisation

Figure 6 shows the data corresponding to the results described in Section 3.1. We offer this visualisation for the sake of clarity as correlation coefficients can often be misleading without the accompanying data. By and large, the visualisation of the data

---

[6] The mask tag and full-stop were omitted for GPT2-M. As ELECTRA cannot be used in this fashion, we instead used the generative counterpart of the model to obtain the list of names and confirmed that ELECTRA accepts them in their respective contexts (i.e. that it labels them as *original* tokens).

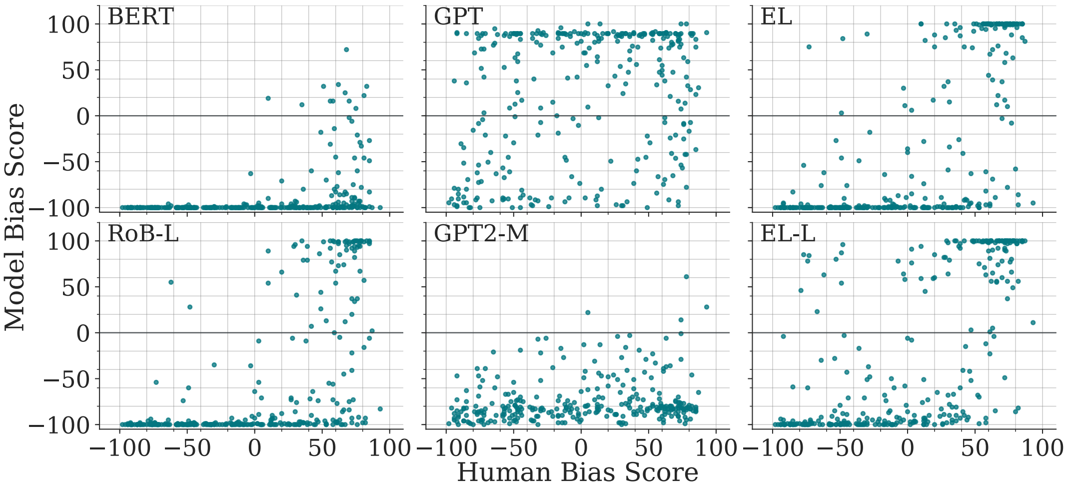

Figure 6: Model bias to human bias agreement.

corroborates the results discussed in the main body of the paper.

One observation here is that human bias scores are uniformly distributed between the two extremes (100 and -100), whereas model bias scores tend to be closer to the two endpoints. The scores obtained in the human-based study represent the responses of 100 people to the same stimulus. Deviation from the extremities here show that people differ in their judgements to some degree. The scores obtained in our study represent the responses of a single model to 200 variants of the same stimulus. In this sense, it is not surprising that the model's scores occupy the two extremities – this shows the models' consistency in judging a given verb to be subject or object biasing.

G Model representations

To obtain a single representation of any given verb from any given model, we encode a sequence like *John praised Mary* and take the representation for the first subtoken of the verb. With BERT's tokenizer, for example, the first subtoken of a verb amounts to the full verb form 57.1% of the time, and to the root of the verb 19.3%; in the remaining 22.6% the unit is 'meaningless.' These numbers vary across models, but in all cases, we are looking at contextualized embeddings, so even 'meaningless' subtokens should be a valid proxy to the verb's representation. To abstract away from the exact choice of proper nouns, we repeat this procedure for the 200 name variants and take the elementwise average over all the representations.

H Spanish and German experiments

The contexts we used for extracting proper nouns from the non-English models were *Er ist ein Mann und heißt* and *Sie ist eine Frau und heißt* for German and *Ella es una mujer y se llama* and *El es un hombre y se llama* for Spanish. For BERT DE we used *Sie heißt* and *Er heißt* instead, as these stimuli more consistently yielded names in the high ranks. Since the generator for ELECTRA DE is not publicly available, we used a different procedure to obtain the lists of personal nouns for this model: we queried WikiData for the top 100 male and female given names for people from Germany and then scored these with ELECTRA in the context shown above. We selected the 10 names for each gender that yielded highest probability of an *O* label.

For the nonce-word slot of the stimuli, we sourced nonce nouns from Dykstra-Pruim (2003) and Zaretsky et al. (2016) (German) and Cantú Sánchez (2011) (Spanish). As these lists contain less than 200 words, here sampling for the nonce-word slot in the 200 variants of a stimulus was done *with* replacement.

Do It Once: An Embarrassingly Simple Joint Matching Approach to Response Selection

Linhao Zhang[1], **Dehong Ma**[2], **Sujian Li**[1], **Houfeng Wang**[1]
[1]MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[2]Baidu Inc., China
{zhanglinhao, lisujian, wanghf}@pku.edu.cn
madehong@baidu.com

Abstract

Existing matching models for response selection adopt the independent matching (*IM*) approach. To complete a prediction, they have to perform N independent matches, where N is the number of response options. In this paper, we explore a joint matching (*JM*) approach which performs matching only once regardless of the number of options. The *JM* approach does not change the structure of matching component but only modifies its input and output format. It also enables a cheap but effective data augmentation method. Extensive experiments on the MuTual dataset demonstrate that, even with the simplest formulation, JM outperforms *IM* approach by a large margin and reduces training time by over half.

1 Introduction

The availability of large-scale datasets has driven the development of neural dialogue systems. One important task in dialogue systems is response selection, which plays an essential role in retrieval-based chatbots (Ji et al., 2014). It aims to select the best-matched response from a set of response options for a dialogue. As shown in Figure 1, given a dialogue context and four response options, we need to choose the only logically correct one.

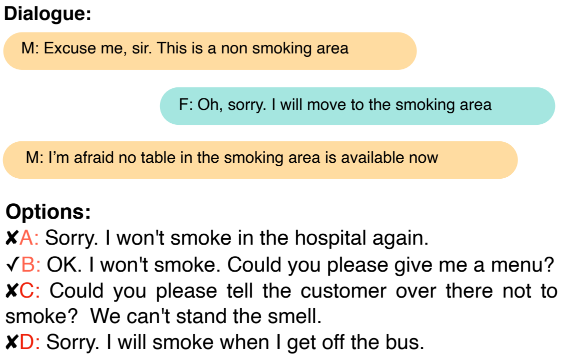

Dialogue:

M: Excuse me, sir. This is a non smoking area

F: Oh, sorry. I will move to the smoking area

M: I'm afraid no table in the smoking area is available now

Options:

✘A: Sorry. I won't smoke in the hospital again.
✓B: OK. I won't smoke. Could you please give me a menu?
✘C: Could you please tell the customer over there not to smoke? We can't stand the smell.
✘D: Sorry. I will smoke when I get off the bus.

Figure 1: Example of response selection.

Previous work in response selection follows an *independent matching (IM)* approach and computes a matching score for each of the N response options independently. Various matching models following this approach have been proposed (Zhou et al., 2016; Wu et al., 2017; Zhou et al., 2018; Chaudhuri et al., 2018; Tao et al., 2019; Yuan et al., 2019). Despite its success in pre-BERT era, we argue that the *IM* approach does not make full use of the ability of pretrained encoders (such as BERT and RoBERTa) to encode multiple sentences, hence may hinder both efficiency and effectiveness. Specifically, to complete a prediction, the *IM* approach has to perform N independent matches, which means N gradient computations (where N is the number of response options). Besides, the dialogue context is repeatedly encoded N times, which further contributes to the inefficiency. The other drawback is that options in these models are independent and agnostic of each other. In reality, humans often compare all the options and utilize their correlations to make a comprehensive decision.

In this paper, we describe a *joint matching* (*JM*) approach for this task. For any matching model, we do not change its inner structure but only modify its input and output format. Specifically, we first add a special token at the start of each option, and then concatenate all options into a single sequence. The option sequence is then matched as a whole with the dialogue context. Finally, we extract vectors corresponding with the special token to calculate matching scores. Note that *JM* can complete a prediction with a single match, which means it only requires one gradient computation and context encoding. Besides, thanks to the self-attention mechanism (Vaswani et al., 2017) of BERT-based matching models, options can now directly attend to each other, rather than being agnostic.

Another advantage of *JM* approach is that it nat-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4872–4877
August 1–6, 2021.

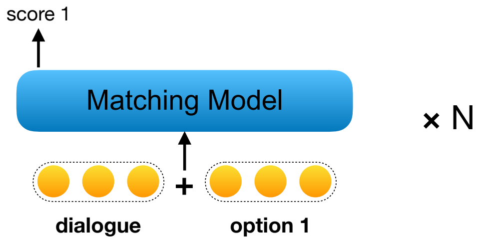

(a) Independent Matching

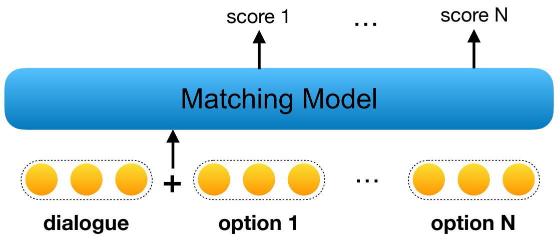

(b) Joint Matching

Figure 2: Overview of Independent Matching and Joint Matching.

urally enables a simple yet effective data augmentation method. The basic idea is that since options are sequentially concatenated in *JM*, new training instances can be easily created by changing the permutation order of options. Therefore, a dialogue with M response options can create at most $M!$ (factorial M) times as many training instances.

We conduct experiments on the MuTual dataset (Cui et al., 2020), a publicly available English dataset for multi-turn dialogue response selection. Results show that *JM* advances *IM* on three matching models and can significantly reduce training time. Besides, the permutation-based data augmentation method gives further improvement.

2 Model

The overview of IM and JM is shown in Figure 2. We describe the details in the following subsections[1].

2.1 Background

Given a dialogue $\mathcal{D}$ with M utterances $\{U_i\}_{i=1}^{M}$, and a set of N response options $\{O_j\}_{j=1}^{N}$, the goal of response selection is to select the logically correct option $\hat{O}$.

Previous work (Cui et al., 2020) shows that pretrained matching models define the state-of-the-art on this task. Similar to using BERT for sentence-pair classification (Devlin et al., 2019), they first concatenate the context (sentence A) and a candidate response (sentence B) as BERT input (i.e., "[CLS] Excuse me ... [SEP] Sorry ... [SEP]"). On the top of BERT, a fully-connected layer is used for transforming the [CLS] token representation to the matching score. In order to compete a prediction, M independent matchings have to be made, where M is the number of options.

2.2 Joint Matching

Instead of conducting N times of independent matches, we make the first step outside the IM framework and explore a joint matching approach for this task. We first adds a special token [OP]^j at the start of the j^{th} option. It is a token used to aggregate the matching information between the context and the j^{th} option into a single vector. We then concatenate all the options into a single sequence. Formally,

$$S_O = [OP]^1 O_1 [OP]^2 O_2 ... [OP]^N O_N \tag{1}$$

For the dialogue context, we concatenate all the utterances into a single sequence. Formally,

$$S_D = U_1 U_2 ... U_M \tag{2}$$

The two sequences are then separated with a [SEP] token and fed into our pretrained encoder. Formally,

$$X = [CLS] S_D [SEP] S_O [SEP] \tag{3}$$

For any BERT-based matching model, suppose the output embeddings of the model are $H_t \in \mathbb{R}^{|X| \times d}$. To perform scoring, we first extract outputs corresponding to [OP]^j and represent them as $h^{[OP]^j}$. The only new parameters learned are a score vector $W \in \mathbb{R}^d$. The probability of option j being the answer is computed as a dot product between $h^{[OP]^j}$ and W followed by a softmax over all of the options. Formally,

$$P_j = \frac{e^{W \cdot h^{[OP]^j}}}{\sum_i e^{W \cdot h^{[OP]^i}}} \tag{4}$$

The training objective is the log-likelihood of the correct answer[2].

---

[1] The code is at https://github.com/gitzlh/JM-Matching

[2] For the score layer, an intuitive approach is to use $h^{[CLS]}$ instead of $h^{[OP]}$ and cast the prediction into a M-classes classification. However, we found that this approach leads to poor performance.

In this way, the JM approach only needs to match the context sequence and the option sequence once. Compared with the *IM* approaches, JM is computational efficient in two ways: 1) It encodes the dialogue context only once, instead of *M* times. However, this benefit is partially offset by the fact that the complexity of transformer grow quadratically with the length of input 2) More importantly, IM approaches need to compute gradients *M* times for each training step. Besides, in each self-attention layer of the BERT-based matching model, options can directly attend to and interact with each other. This process mimics how humans solve multi-choice questions, that we often compare all the options before making the decision.

2.3 Permutation-Based Data Augmentation

Another advantage of *JM* is that it naturally enables a permutation-based data augmentation (PBDA) method, which can generate high-quality labeled data to improve response selection.

Specifically, since the input of our model is organized as Equation 3, we can create new training instances by simply changing the concatenation order of the options. For example, from $[OP]^1O_1[OP]^2O_2...[OP]^NO_N$ to $[OP]^2O_2[OP]^1O_1...[OP]^NO_N$, we create a new training example (see Figure 2). Correspondingly, the ground-truth label of the training instance may be changed. In this way, a single dialogue can create at most $M!$ times training instances.

3 Experiments

3.1 Dataset

We evaluate our model on the Mutual dataset (Cui et al., 2020), a human-labeled, open-domain and reasoning-based dataset for multi-turn response selection. Compared with previous datasets (Lowe et al., 2015; Zhang et al., 2018; Welleck et al., 2019), MuTual is more challenging since it requires some reasoning ability. Models that achieve close-to-human performance on previous datasets, still perform far behind human performance on MuTual. The statistics of MuTual are shown in Table 1. Note that since Mutual has 4 options for each dialogue, PBDA can thus creates at most 24 (4!) times as many training instances.

	MuTual
Training set	7088
Validation set	886
Test set	886
# Avg. Turns / Dialogue	4.73
# Avg. Words / Utterance	19.57
# Options	4

Table 1: Statistics of MuTual.

3.2 Settings

We use PyTorch to implement JM on three matching models. We adopt AdamW (Loshchilov and Hutter, 2018) as our optimizer, and the peak learning rate and warmup proportion are set to 1e-5 and 0.06, respectively. We use the largest batch size that fits in the memory of our GPU and use gradient accumulation for an effective batch size of 32. Dropout (Srivastava et al., 2014) is employed before the score layer with a rate of 0.1. We train our model for 15 epochs and choose the model that reports the highest R@1 on the validation set.

Following previous work (Cui et al., 2020), we evaluate our model with recall at position 1 in 4 candidates (R@1), recall at position 2 in 4 candidates (R@2) and Mean Reciprocal Rank (MRR).

4 Results

4.1 Main Performance

Table 2 gives the comparison of IM and JM on three matching models on the MuTual dataset. Note that we only experiment with pretrained matching models given that they have an overwhelming advantage over non-pretrained models. For a fair comparison, we report baseline results both from the official reports of MuTual (Cui et al., 2020) and our own implementation.

Our first observation is that RoBERTa-based models significantly outperform BERT-based models, suggesting that RoBERTa is a more powerful feature extractor. More importantly, we note that our JM approach outperforms IM approach on all three matching models. For example, BERT-JM improves over BERT-IM by 6% (absolute) R@1. We suppose that this is because the JM approach concatenates all the options as the model input, and, thanks to the self-attention mechanism, each option can directly attend to each other. In this way, JM can make a more comprehensive decision and boost the performance especially on challenging datasets like MuTual.

The conclusion holds true for larger pretrained matching models such as RoBERTa-large. As

Methods	R@1	R@2	MRR
Human performance	0.938	0.971	0.964
BERT-IM (Cui et al., 2020)	0.648	0.847	0.795
BERT-IM †	0.641	0.853	0.793
BERT-JM	**0.702**	**0.904**	**0.833**
RoBERTa-IM (Cui et al., 2020)	0.713	0.892	0.836
RoBERTa-IM †	0.770	0.912	0.868
RoBERTa-JM	**0.784**	**0.933**	**0.880**
RoBERTa-large-IM †	0.844	0.958	0.914
RoBERTa-large-JM	**0.870**	**0.973**	**0.930**

Table 2: Main results on MuTual. We can see that *JM* outperforms *IM* approach for every pretrained encoders. † means our own implementation results. Note that the JM results are achieved without data augmentation.

Methods	Forward	Backward	Total
RoBERTa-IM	116	412	528
RoBERTa-JM	**71**	**168**	**239**

Table 3: Average training time (second) per epoch. RoBERTa-IM and RoBERTa-JM both use the largest batch size available on the same GPU.

shown in Table 2, RoBERTa-large-JM brings about 3% (absolute) improvement over RoBERTa-large-IM in terms of R@1 and even surpasses human performance in terms of R@2.

4.2 Training Time

In the task of response selection, the scalability of the model becomes an issue when the number of options increases. In this subsection, We compare JM and IM with respect to training time[3]. As shown in Table 3, RoBERTa-JM reduces the training time by 55% compared with RoBERTa-IM. More detailed analysis shows that the reduction is mostly contributed to the backward propagation process. This is because to complete a prediction, RoBERTa-IM performs M independent matches and thus requires M gradient computations, a costly process. It also needs to encode the dialogue context M times, leading to computational inefficiency especially in multi-turn settings. By contrast, RoBERTa-JM requires only a single match[4].

Methods	R@1	R@2	MRR
RoBERTa-JM	0.784	0.933	0.880
RoBERTa-JM 4x	0.793	**0.947**	0.887
RoBERTa-JM 8x	**0.813**	0.942	**0.896**
RoBERTa-JM 24x	0.807	0.942	0.892

Table 4: PBDA results. 4x, 8x and 24x mean augmenting the data size by 4, 8 and 24 times, respectively.

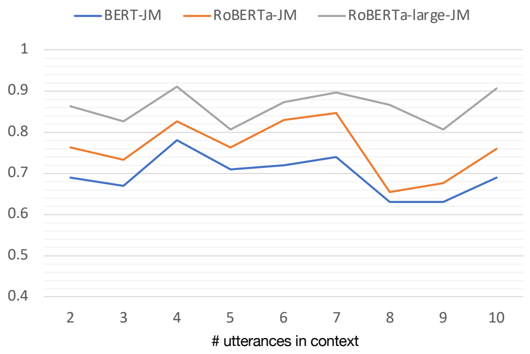

Figure 3: Performance on MuTual dev set across different contexts.

4.3 Data Augmentation

In this subsection, we conduct experiments to verify the effectiveness of PBDA.

As shown in Table 4, PBDA 4x brings a 1% (absolute) improvement in terms of both R@1 and R@2, showing that by simply permuting the options, we can create high-quality training instances. Besides, when increasing the data size by 8 times, we observe another 2% (absolute) improvement in terms of R@1. An interesting observation is that PBDA 24x does not further improve model performance, showing that there is a limit to the improvement brought by data augmentation.

---

[3] Both models are trained on a single NVIDIA TITAN Xp GPU.

[4] We note that this benefit is partially offset by the transformer's quadratic complexity with regard to the length of input. Suppose that the average length of the dialogue utterance and response option is L, then the time complexity of the IM and JM approach is $O((ML+L)^2N)$ and $O((ML+NL)^2)$, respectively

4.4 Context Length

Following previous work (Wu et al., 2017), we further investigate how JM performs cross the length of context. As demonstrated in Figure 3, the performance of BERT-JM and RoBERTa-JM is generally satisfactory, except for the slight deterioration when the context has more than seven utterances. It can also deal with a short context that only has two utterances.

On the other hand, RoBERTa-large-JM consistently performs better than RoBERTa-JM, and when the context becomes longer, the gap becomes larger. It also gives more stable performance across different context lengths, further showing the strong representation ability of RoBERTa-large.

5 Conclusions

In this paper, we make the first step outside the independent matching framework and explore a joint matching approach for response selection. We also present an effective permutation-based data augmentation method. We conduct experiments on the MuTual dataset and demonstrate the effectiveness and efficiency of our approach. Besides, the proposed data augmentation further improves model performance.

6 Acknowledgments

The work is supported by National Natural Science Foundation of China under Grant No.62036001 and PKU-Baidu Fund (No. 2020BD021). The corresponding author of this paper is Houfeng Wang.

References

Debanjan Chaudhuri, Agustinus Kristiadi, Jens Lehmann, and Asja Fischer. 2018. Improving response selection in multi-turn dialogue systems by incorporating domain knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 497–507, Brussels, Belgium. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. Mutual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Conference of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. One time of interaction may not be enough: Go deep with an interaction-over-interaction network for response selection in dialogues. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741, Florence, Italy. Association for Computational Linguistics.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Vancouver, Canada. Association for Computational Linguistics.

Chunyuan Yuan, Wei Zhou, Mingming Li, Shangwen Lv, Fuqing Zhu, Jizhong Han, and Songlin Hu. 2019. Multi-hop selector network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 111–120.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381, Austin, Texas. Association for Computational Linguistics.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127, Melbourne, Australia. Association for Computational Linguistics.

Climbing the Tower of Treebanks: Improving Low-Resource Dependency Parsing via Hierarchical Source Selection

Goran Glavaš
University of Mannheim
Data and Web Science Group
goran@informatik.uni-mannheim.de

Ivan Vulić
University of Cambridge
Language Technology Lab
iv250@cam.ac.uk

Abstract

Recent work on multilingual dependency parsing focused on developing highly multilingual parsers that can be applied to a wide range of low-resource languages. In this work, we substantially outperform such "one model to rule them all" approach with a heuristic selection of languages and treebanks on which to train the parser for a specific target language. Our approach, dubbed TOWER, first hierarchically clusters all Universal Dependencies languages based on their mutual syntactic similarity computed from human-coded URIEL vectors. For each low-resource target language, we then climb this language hierarchy starting from the leaf node of that language and heuristically choose the hierarchy level at which to collect training treebanks. This treebank selection heuristic is based on: (i) the aggregate size of all treebanks subsumed by the hierarchy level and (ii) the similarity of the languages in the training sample with the target language. For languages without development treebanks, we additionally use (ii) for model selection (i.e., early stopping) in order to prevent overfitting to development treebanks of closest languages. Our TOWER approach shows substantial gains for low-resource languages over two state-of-the-art multilingual parsers, with more than 20 LAS point gains for some of those languages. Parsing models and code available at: `https://github.com/codogogo/towerparse`.

1 Introduction

Syntactic parsing – grounded in a wide variety of formalisms (Taylor et al., 2003; De Marneffe et al., 2006; Hockenmaier and Steedman, 2007; Nivre et al., 2016, *inter alia*) – has been the backbone of natural language processing (NLP) for decades, and an indispensable preprocessing step for tackling higher-level language understanding tasks. A recent major paradigm shift in NLP towards large-scale pretrained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020) and their end-to-end fine-tuning for downstream tasks has reduced the downstream relevance of supervised syntactic parsing. What is more, there is more and more evidence that PLMs implicitly acquire rich syntactic knowledge through large-scale pretraining (Hewitt and Manning, 2019; Chi et al., 2020) and that exposing them to explicit syntax from human-coded treebanks does not offer significant language understanding benefits (Kuncoro et al., 2020; Glavaš and Vulić, 2021). In order to implicitly acquire syntactic competencies, however, PLMs need language-specific corpora at the scale at which it can only be obtained for a tiny portion of world's 7,000+ languages. For the remaining vast majority of languages – with limited-size monolingual corpora – explicit syntax still provides valuable linguistic bias for more sample-efficient learning in downstream NLP tasks.

Reliable syntactic parsing requires annotated treebanks of reasonable size: this prerequisite is, unfortunately, satisfied for even fewer languages. Despite the multi-year, well-coordinated annotation efforts such as the Universal Dependencies (Nivre et al., 2016, 2020) project, language-specific treebanks are unlikely to appear anytime soon for most world languages. This renders the transfer of syntactic knowledge from high-resource languages with annotated treebanks a necessity. A *truly zero-shot* transfer for *low-resource languages* assumes a set of training treebanks from resource-rich source languages and a target language without any syntactic annotations. Effectively, the task is then to identify the subset of source treebanks, the parser trained on which would yield the best parsing performance for the target language. An exhaustive search over all possible subsets of source treebanks is not only computationally intractable[1] but also

---

[1] One can create $2^N - 1$ different training sets from a

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4878–4888
August 1–6, 2021. ©2021 Association for Computational Linguistics

uninformative in true zero-shot scenarios in which there is no development treebank (i.e., any syntactically annotated data) for the target language. Most existing transfer methods therefore either (1) choose one (or a few) best source languages for each target language (Rosa and Zabokrtsky, 2015; Agić, 2017; Lin et al., 2019; Litschko et al., 2020) or (2) train a single multilingual parser on all available treebanks; such parsers, based on pretrained multilingual encoders, currently produce best results in low-resource parsing (Kondratyuk and Straka, 2019; Üstün et al., 2020). Other transfer approaches, e.g., based on data augmentation (Şahin and Steedman, 2018; Vania et al., 2019), violate the zero-shot transfer by assuming a small target-language treebank – a requirement unfulfilled for most world languages.[2]

In this work, we propose a simple and effective heuristic for selecting a good set of source treebanks for any given low-resource target language. In our approach, named TOWER, we first hierarchically cluster all Universal Dependencies (UD) languages. To this end, we compute syntactic similarity of languages by comparing manually coded vectors of their syntactic properties from the URIEL database (Littell et al., 2017). We then iteratively 'climb' that language hierarchy level by level, starting from the leaf node of the target language. We stop 'climbing' (i.e., select the set of source treebanks subsumed by the current hierarchy level), when the relative decrease in linguistic similarity of the training sample w.r.t the target language outweighs the increase in size of the training sample. We additionally exploit the linguistic similarity between the target language and its closest sources with existing development treebanks to inform a model selection (that is, early-stopping) heuristic. TOWER substantially outperforms state-of-the-art multilingual parsers – UDPipe (Straka, 2018), UDify (Kondratyuk and Straka, 2019), and UDapter (Üstün et al., 2020) on low-resource languages, while offering comparable performance for high-resource languages.

2 Climbing the TOWER of Treebanks

Constructing the TOWER. We start by hierarchically clustering the set of 89 languages from Universal Dependencies [3] based on their syntactic similarity. To this end, we represent each language with its `syntax_knn` vector from the URIEL database (Littell et al., 2017). Features of these 103-dimensional vectors correspond to individual syntactic properties from manually coded linguistic resources such as WALS (Dryer and Haspelmath, 2013) and SSWL (Collins and Kayne, 2009). URIEL's `syntax_knn` strategy replaces feature values missing in those resources with kNN-based predictions (cf. (Littell et al., 2017) for more details). We then carry out hierarchical agglomerative clustering with Ward's linkage (Anderberg, 2014) with Euclidean distances between URIEL vectors guiding the clustering. Figure 1 shows a dendrogram of one part of the resulting hierarchy. We display the complete hierarchy in the Appendix. The syntax-based clustering largely reflects memberships in language (sub)families, with a few notable exceptions: e.g., Tagalog (*tl*), from the Austronesian family appears to be syntactically similar to (and is joined with) Scottish (*gd*), Irish (*ga*), and Welsh (cy) from the Celtic branch of the Indo-European family.

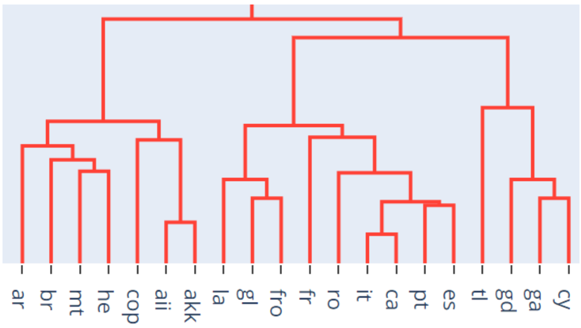

Figure 1: Part of the syntax-based hierarchical clustering of UD languages (ISO 639-1 codes).

Treebank Selection (TBS). For a given test treebank, we start climbing the hierarchy from the leaf node of the treebank's language. Let s_l denote the number of climbing steps we take from the target leaf node l. If the target test treebank also has the corresponding training portion, in-treebank training constitutes the first training configuration (we denote this configuration with $s_l = -1$). For resource-rich languages with several training treebanks, we create the next training sample by concatenating all of those treebanks (we denote this level with $s_l = 0$).[4] For low-resource target lan-

---

collection of N source treebanks.

[2] For the vast majority of world languages there does not exist a single manually annotated syntactic tree.

[3] We worked with the UD version 2.5.

[4] For example, for the Russian test treebank *SynTagRus*, the training set at $s_l = -1$ consists of the train portion of the same *SynTagRus* treebank; at $s_l = 0$, we concatenate

guages without any training treebanks, the first training sample is collected at $s_l = 1$, where the language is joined with other languages. The training set corresponding to a hierarachy level (i.e., each *join* in the tree) concatenates all training treebanks of all languages (i.e., leaf nodes) of the respective hierarchy subtree.[5]

Let $\{S_n\}_{n=0 \text{ (or } -1)}^{N}$ be the set of training configurations collected by climbing the hierarchy starting from the target language l and let $S_n = \cup\{T_k\}_{k=1}^{K}$ be the n-th training set consisting of K training treebanks. As we climb the hierarchy (i.e., as n increases), the training set S_n is bound to grow; at the same time, the sample of training languages becomes increasingly dissimilar w.r.t. the target language l. In other words, as we climb higher up the induced syntactic hierarchy of languages, we train on more data but from a mixture of (syntactically) more distant languages. Let l_k be the language of the training treebank T_k. We then quantify the syntactic similarity $sim(S_n, l)$ between the training set S_n and the target language l as follows:

$$sim(S_n, l) = \frac{1}{|S_n|} \sum_{k=1}^{K} |T_k| \cdot \cos(\mathbf{l}_k, \mathbf{l}) \quad (1)$$

with $\cos(\mathbf{l_k}, \mathbf{l})$ as cosine similarity between URIEL vectors of l_k and l, and relative sizes of individual treebanks $|T_k|/|S_n|$ as weights. We then use the following simple heuristic to select the best training set S_n: we stop climbing when the relative growth of the training set becomes smaller than the relative decrease of the similarity with the target language, i.e., we select the smallest n for which the following condition is satisifed:

$$\frac{|S_{n+1}|}{|S_n|} < \frac{sim(S_n, l)}{sim(S_{n+1}, l)}. \quad (2)$$

Model Selection (MS). Early stopping based on the model performance on a development set (*dev*) is an important mechanism for preventing model overfitting in supervised machine learning. In a truly zero-shot transfer setup, on the one hand, we do not have any development data in the target language. Model selection based on the development set of the source language, on the other hand, overfits the model to the source language, which may hurt effectiveness of the cross-lingual transfer (Keung et al., 2020; Chen and Ritter, 2020). For test treebanks with a respective development portion, TOWER uses that development set for model selection. For low-resource languages l without development treebanks, we compile a *proxy* development set $D_l = \cup\{D_k\}_{k=1}^{K}$ by collecting all development treebanks D_k from the hierarchy level closest to l that encompasses at least one treebank with a development set.[6] Intuitively, the more syntactically similar D_l is to l, the more beneficial the model selection based on D_l will be for performance on l, the optimal model checkpoint w.r.t. l should be closer to the model checkpoint exhibiting best performance on D_l. Accordingly, with M as the model checkpoint with best performance on D_l, we select the model chekpoint $M' = \lfloor sim(D_l, l) \cdot M \rfloor$ (see Eq.(1)) as the "optimal" checkpoint for the target language l.

Shallow Biaffine Parser. TOWER employs the shallow biaffine parser of Glavaš and Vulić (2021), stacked on top of the pretrained XLM-R (Conneau et al., 2020). Compared to the standard biaffine parser (Dozat and Manning, 2017; Kondratyuk and Straka, 2019; Üstün et al., 2020), this shallow variant forwards word-level representations (aggregated from subword output) directly into biaffine products, bypassing deep feed-forward transformations that produce dependent- and head-specific vectors (Dozat and Manning, 2017). The shallow variant is reported to perform comparably (Glavaš and Vulić, 2021), while being faster to train.

3 Evaluation and Discussion

Treebanks and Baselines. We evaluate TOWER on 138 (test) treebanks from Universal Dependencies (Nivre et al., 2020).[7] We compare TOWER against two state-of-the-art multilingual parsers: (1) UDify (Kondratyuk and Straka, 2019) couples the multilingual BERT (mBERT) (Devlin et al.,

---

the training portions of Russian *GSD*, *PUD*, and *SynTagRus* treebanks.

[5] Note that the number of climbs s_l needed to reach some hierarchy level depends on the language l: e.g., the hierarchy level joining Tagalog (*tl*) with Scottish, Irish, and Welsh ($\{gd, ga, cy\}$) is reached in $s_l = 1$ climbs from Tagalog, $s_l = 2$ climbs from Scottish and $s_l = 3$ climbs from Irish and Welsh.

[6] E.g., D_l for l=*tl* consists of develompent portions of *ga* and *gd* treebanks, whereas D_l for l=*cy* consists only of the development set of *ga*.

[7] We work with UD v2.5. Due to mismatches between XLM-R's subword tokenizer and word-level treebank tokens we skip: all Chinese treebanks, Assyrian (AS), Old Russian (RNC and TOROT), Skolt Sami (Giellagas), Japanese (Modern and BCCWJ), A. Greek (Perseus), Gothic (PROIEL), Coptic (Scriptorium), OC Slavonic (PROIEL) and Yoruba (YTB).

	∩UDify		∩UDapt		HIGH		LOW	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
UDify	80.9	73.9	–	–	89.2	85.3	39.9	22.2
UDapter	–	–	63.8	52.8	**90.9**	**87.6**	43.9	29.3
TOWER	**82.4**	**74.3**	**68.9**	**56.0**	90.0	86.3	**53.7**	**33.8**
-TBS	80.8	73.2	62.8	51.7	89.4	85.6	47.0	30.1
-MS	82.1	74.1	67.9	55.2	89.4	85.6	51.2	32.2
-TBS-MS	80.7	83.1	62.4	51.3	89.4	85.6	45.9	29.0

Table 1: Parsing performance (UAS, LAS) on different UD treebank subsets for state-of-the-art multilingual parsers UDify and UDapter and variants of our TOWER method. **Bold:** best performance in each column.

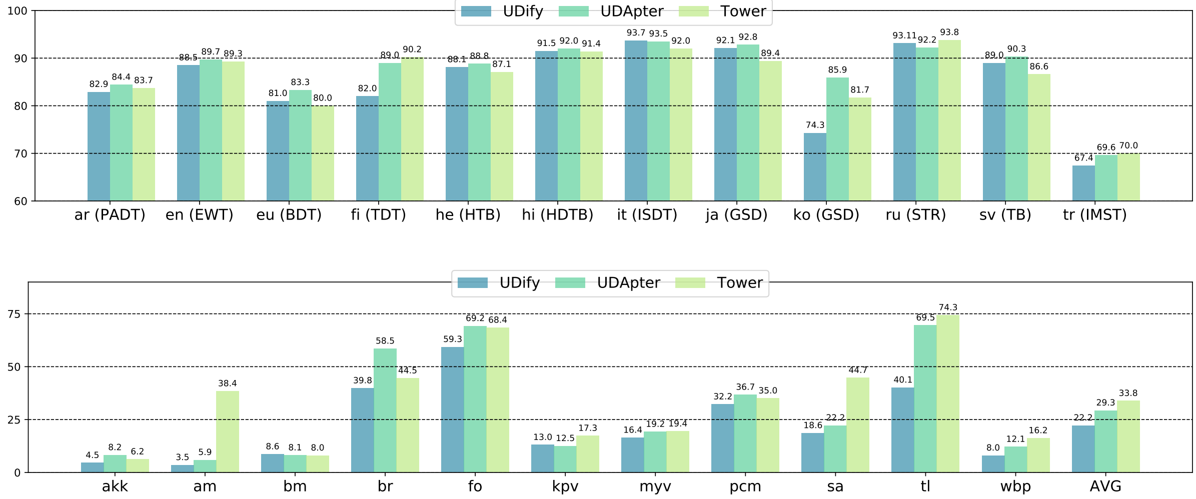

Figure 2: LAS performance of UDify, UDapter and TOWER on 12 high-resource treebanks (top figure), and 11 low-resource languages (bottom figure).

2019) with the deep biaffine parser (Dozat and Manning, 2017) and trains on all UD treebanks; (2) UDapter (Üstün et al., 2020) extends mBERT with adapter parameters (Houlsby et al., 2019; Pfeiffer et al., 2020) that are contextually generated (Platanios et al., 2018) from URIEL vectors – the parameters of the adapter generator are trained on treebanks of 13 diverse resource-rich languages selected by Kulmizev et al. (2019). We additionally quantify the contributions of TOWER's heuristic components (TBS and MS, see §2) by evaluating variants in which we (1) remove TBS and train on the closest language with training data (-TBS), (2) remove MS and just select the model checkpoint that performs best on the proxy dev set D_l (-MS), and (3) remove both TBS and MS (-TBS-MS).

Training and Optimization Details. We limit input sequences to 128 subword tokens. We use XLM-R *Base* with $L = 12$ layers and hidden size $H = 768$ and apply a dropout ($p = 0.1$) on its outputs before forwarding them to the shallow parsing head. We train in batches of 32 sentences and optimize parameters with Adam (Kingma and Ba, 2015) (starting learning rate 10^{-5}). We train for 30 epochs, with early stopping based on dev loss.[8]

Results and Discussion. We show detailed results for all 138 treebanks in the Appendix. In Table 1, we show averages over different treebank subsets: treebanks on which both TOWER and (1) UDify (∩UDify; 111 treebanks) and (2) UDapter (∩UDapt; 39 treebanks) have been evaluated, (3) 12 high-resource languages on which UDapter was trained (HIGH) and (4) 11 low-resource treebanks (LOW) for which all three models have been evaluated. We show LAS scores for languages from

---

[8] For low-resource languages without the dev set, we use the proxy D_l (see 2). We checkpoint the model (i.e., measure the dev loss) 10 times per epoch and stop training when the loss does not decrease over 10 consecutive checkpoints.

HIGH and LOW in Figure 2. Similar trends are observed with UAS scores.

TOWER outperforms UDify and UDapter in all setups except HIGH, with especially pronounced gains for LOW. This renders TOWER particularly successful for the intended use case: low-resource languages without any training data. Admittedly, the fact that TOWER is built on XLM-R, whereas UDify and UDapter use mBERT, impedes the direct "apples-to-apples" comparison. Two sets of results, however, strongly suggest that it is TOWER's heuristics (TBS & MS) that drive its performance rather than the XLM-R (instead of mBERT) encoder. First, UDapter outperforms TOWER on high-resource languages with large training treebanks (i.e., the HIGH setup). For these languages, however, TOWER effectively does not employ its heuristics: (i) TBS selects the large language-specific treebank(s), as adding any other language prohibitively reduces the perfect similarity $sim(S_0, l) = 1$ (see Eq. (1)); (ii) MS is not used because each high-resource treebank has its own dedicated dev set. Secondly, removing TOWER's heuristics (see -TBS-MS in Table 1) brings its performance slightly below that of UDapter, rendering TBS (primarily) and MS (rather than the XLM-R encoder) crucial for TOWER's gains. Comparing -TBS and -MS reveals that, somewhat expectedly, selecting the "optimal" training sample (TBS) contributes to the overall performance more than the heuristic early stopping (MS).

Looking at individual low-resource languages (Fig. 2), we observe largest gains for Amharic (*am*) and Sanskrit (*sa*). While Sanskrit benefits from TOWER selecting training languages from the same family (Marathi, Urdu, and Hindi), Amharic (Afro-Asiatic family), interestingly, benefits from treebanks of syntactically similar languages from another family (cf. the full TOWER hierarchy in the Appendix) – Tamil and Telugu (Dravidian family). Similarly, Tagalog (Austronesian language) parsing massively benefits from training on Scottish and Irish treebanks (Indo-European, Celtic).

4 Conclusion

We proposed TOWER, a simple yet effective approach to the crucial problem of source language selection for multilingual and cross-lingual dependency parsing. It leverages the language hierarchy, induced from syntax-based manually coded URIEL language vectors, and simple treebank selection heuristics to inform the source selection. A wide-scale UD evaluation and comparisons to current state-of-the-art multilingual dependency parsers validated the effectiveness of TOWER, especially in low-resource languages. Moreover, while the main experiments in this work were based on one particular state-of-the-art parsing architecture, TOWER is fully independent of the chosen underlying parsing model, and thus widely applicable.

Acknowledgments

Goran Glavaš is supported by the Baden Württemberg Stiftung (Eliteprogramm, AGREE grant). The work of Ivan Vulić is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no. 648909).

References

Željko Agić. 2017. Cross-lingual parser selection for low-resource languages. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies*, pages 1–10.

Michael R. Anderberg. 2014. *Cluster Analysis for Publications*. Academic Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of NeurIPS 2020*.

Yang Chen and Alan Ritter. 2020. Model selection for cross-lingual transfer using a learned scoring function. *arXiv preprint arXiv:2010.06127*.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of ACL 2020*, pages 5564–5577.

Chris Collins and Richard Kayne. 2009. Syntactic structures of the world's languages.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL 2020*, pages 8440–8451.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC 2006*, pages 449–454.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR 2017*.

M. S. Dryer and M. Haspelmath. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Goran Glavaš and Ivan Vulić. 2021. Is supervised syntactic parsing beneficial for language understanding? An empirical investigation. In *Proceedings of EACL 2021*, pages 3090–3104.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of NAACL-HLT 2019*, pages 4129–4138.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of ICML 2019*, pages 2790–2799.

Phillip Keung, Yichao Lu, Julian Salazar, and Vikas Bhardwaj. 2020. Don't use English dev: On the zero-shot cross-lingual evaluation of contextual embeddings. In *Proceedings of EMNLP 2020*, pages 549–554.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR 2015*.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of EMNLP-IJCNLP 2019*, pages 2779–2795.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of EMNLP-IJCNLP 2019*, pages 2755–2768.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the ACL*, 8:776–794.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of ACL 2019*, pages 3125–3135.

Robert Litschko, Ivan Vulić, Željko Agić, and Goran Glavaš. 2020. Towards instance-level parser selection for cross-lingual transfer of dependency parsers. In *Proceedings of COLING 2020*, pages 3886–3898.

Patrick Littell, David R Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of EACL 2017*, pages 8–14.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of LREC 2016*, pages 1659–1666.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting Transformers. In *Proceedings of EMNLP 2020: System Demonstrations*, pages 46–54.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of EMNLP 2018*, pages 425–435.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. Klcpos3-a language similarity measure for delexicalized parser transfer. In *Proceedings of ACL-JCNLP 2015*, pages 243–249.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of EMNLP 2018*, pages 5004–5009.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The Penn treebank: An overview. In *Treebanks*, pages 5–22.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly universal dependency parsing. In *Proceedings of EMNLP 2020*.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of EMNLP 2019*, pages 1105–1116.

Appendix

Treebank	Method	UAS	LAS
Afrikaans AfriBooms (af)	UDPipe	89.38	86.58
	UDify	86.97	83.48
	TOWER	88.26	85.28
Akkadian PISANDUB (akk)	UDify	27.65	4.54
	UDapter	26.4	8.2
	TOWER	33.45	6.12
Amharic ATT (am)	UDify	17.38	3.49
	UDapter	12.8	5.91
	TOWER	72.64	38.41
Ancient Greek PROIEL (grc)	UDPipe	85.93	82.11
	UDify	78.91	72.66
	TOWER	85.04	79.85
Arabic NYUAD (ar)	TOWER	33.53	15.94
Arabic PADT (ar)	UDPipe	87.54	82.94
	UDify	87.72	82.88
	UDapter	88.66	84.42
	TOWER	88.92	83.72
Arabic PUD (ar)	UDify	76.17	67.07
	TOWER	75.94	59.72
Armenian ArmTDP (hy)	UDPipe	78.62	71.27
	UDify	85.63	78.61
	TOWER	86.57	80.51
Bambara CRB (bm)	UDify	30.28	8.6
	UDapter	28.7	8.1
	TOWER	31.33	8.03
Basque BDT (eu)	UDPipe	86.11	82.86
	UDify	84.94	80.97
	UDapter	87.25	83.33
	TOWER	84.28	80.02
Belarusian HSE (be)	UDPipe	78.58	72.72
	UDify	91.82	87.19
	UDapter	84.16	79.33
	TOWER	86.40	81.56
Bhojpuri BHTB (bho)	UDapter	52.9	37.34
	TOWER	52.62	35.86
Breton KEB (br)	UDify	63.52	39.84
	UDapter	72.91	58.5
	TOWER	67.73	44.47
Bulgarian BTB (bg)	UDPipe	93.38	90.35
	UDify	95.54	92.4
	TOWER	95.67	92.03
Buryat BDT (bxr)	UDPipe	32.6	18.83
	UDify	48.43	26.28
	UDapter	48.68	28.89
	TOWER	51.53	29.16
Catalan AnCora (ca)	UDPipe	93.22	91.06
	UDify	94.25	92.33
	TOWER	94.04	92.08
Croatian SET (hr)	UDPipe	91.1	86.78
	UDify	94.08	89.79
	TOWER	92.22	87.02
Czech CAC (cs)	UDPipe	92.99	90.71
	UDify	94.33	92.41
	TOWER	94.91	92.10
Czech CLTT (cs)	UDPipe	86.9	84.03
	UDify	91.69	89.96
	TOWER	94.11	91.38
Czech FicTree (cs)	UDPipe	92.91	89.75
	UDify	95.19	92.77
	TOWER	95.12	91.83
Czech PDT (cs)	UDPipe	93.33	91.31
	UDify	94.73	92.88
	TOWER	95.01	92.41
Czech PUD (cs)	UDify	92.59	87.95
	TOWER	93.26	87.06
Danish DDT (da)	UDPipe	86.88	84.31
	UDify	87.76	84.5
	TOWER	85.60	82.14
Dutch Alpino (nl)	UDPipe	91.37	88.38
	UDify	94.23	91.21
	TOWER	93.42	90.31
Dutch LassySmall (nl)	UDPipe	90.2	86.39
	UDify	94.34	91.22
	TOWER	92.45	88.29
English ESL (en)	TOWER	30.22	6.45
English EWT (en)	UDPipe	89.63	86.97
	UDify	90.96	88.5
	UDapter	93.12	89.67
	TOWER	92.16	89.29
English GUM (en)	UDPipe	87.27	84.12
	UDify	89.14	85.73
	TOWER	90.07	86.61
English LinES (en)	UDPipe	84.15	79.71
	UDify	87.33	83.71
	TOWER	87.12	82.91
English PUD (en)	UDify	91.52	88.66
	TOWER	90.89	87.33
English ParTUT (en)	UDPipe	90.29	87.27
	UDify	92.84	90.14
	TOWER	89.36	85.63
English Pronouns (en)	TOWER	89.50	85.37
Erzya JR (myv)	UDify	31.9	16.38
	UDapter	34.21	19.15
	TOWER	36.44	19.38
Estonian EDT (et)	UDPipe	88.0	85.18
	UDify	89.53	86.67
	TOWER	90.24	87.08
Estonian EWT (et)	TOWER	88.80	84.54
Faroese OFT (fo)	UDify	67.24	59.26
	UDapter	77.15	69.2
	TOWER	77.43	68.41
Finnish FTB (fi)	UDPipe	90.68	87.89
	UDify	86.37	81.4
	TOWER	91.91	89.05
Finnish PUD (fi)	UDify	89.76	86.58
	TOWER	88.24	82.48
Finnish TDT (fi)	UDPipe	89.88	87.46
	UDify	86.42	82.03
	UDapter	91.87	89.01
	TOWER	92.78	90.22
French FQB (fr)	TOWER	93.36	87.00
French FTB (fr)	TOWER	28.04	14.80
French GSD (fr)	UDPipe	90.65	88.06
	UDify	93.6	91.45
	TOWER	94.06	91.31
French PUD (fr)	UDify	88.36	82.76
	TOWER	91.02	83.52
French ParTUT (fr)	UDPipe	92.17	89.63
	UDify	90.55	88.06
	TOWER	87.90	79.33
French Sequoia (fr)	UDPipe	92.37	90.73
	UDify	92.53	90.05
	TOWER	92.07	89.93
French Spoken (fr)	UDPipe	82.9	77.53
	UDify	85.24	80.01
	TOWER	84.41	74.77
Galician CTG (gl)	UDPipe	86.44	83.82
	UDify	84.75	80.89
	TOWER	83.85	80.65

Treebank	Method	UAS	LAS
Galician TreeGal (gl)	UDPipe	82.72	77.69
	UDify	84.08	76.77
	Tower	77.57	66.87
German GSD (de)	UDPipe	85.53	81.07
	UDify	87.81	83.59
	Tower	89.11	84.19
German HDT (de)	Tower	97.65	96.54
German LIT (de)	Tower	86.55	78.74
German PUD (de)	UDify	89.86	84.46
	Tower	89.15	81.02
Greek GDT (el)	UDPipe	92.1	89.79
	UDify	94.33	92.15
	Tower	94.13	91.16
Hebrew HTB (he)	UDPipe	89.7	86.86
	UDify	91.63	88.11
	UDapter	91.86	88.75
	Tower	90.71	87.05
Hindi HDTB (hi)	UDPipe	94.85	91.83
	UDify	95.13	91.46
	UDapter	95.29	91.96
	Tower	95.12	91.42
Hindi PUD (hi)	UDify	71.64	58.42
	Tower	73.02	50.68
Hungarian Szeged (hu)	UDPipe	84.04	79.73
	UDify	89.68	84.88
	Tower	87.87	81.02
Indonesian GSD (id)	UDPipe	85.31	78.99
	UDify	86.45	80.1
	Tower	83.71	76.84
Indonesian PUD (id)	UDify	77.47	56.9
	Tower	76.71	53.16
Irish IDT (ga)	UDPipe	80.39	72.34
	UDify	80.05	69.28
	Tower	80.33	66.80
Italian ISDT (it)	UDPipe	93.49	91.54
	UDify	95.54	93.69
	UDapter	95.32	93.46
	Tower	94.47	91.98
Italian PUD (it)	UDify	94.18	91.76
	Tower	94.13	89.01
Italian ParTUT (it)	UDPipe	92.64	90.47
	UDify	95.96	93.68
	Tower	95.06	91.57
Italian PoSTWITA (it)	Tower	86.95	81.75
Italian TWITTIRO (it)	Tower	86.93	80.91
Italian VIT (it)	Tower	91.80	87.05
Japanese GSD (ja)	UDPipe	95.06	93.73
	UDify	94.37	92.08
	UDapter	94.87	92.84
	Tower	92.58	89.44
Japanese PUD (ja)	UDify	94.89	93.62
	Tower	91.12	88.41
Karelian KKPP (krl)	UDapter	61.86	48.35
	Tower	62.18	45.60
Kazakh KTB (kk)	UDPipe	53.3	33.38
	UDify	74.77	63.66
	UDapter	74.13	60.74
	Tower	73.70	59.88
Komi Permyak UH (koi)	UDapter	36.89	23.05
	Tower	42.36	25.81
Komi Zyrian IKDP (kpv)	UDify	36.01	22.12
	Tower	40.87	24.71
Komi Zyrian Lattice (kpv)	UDify	28.85	12.99
	UDapter	28.4	12.5
	Tower	33.29	17.33
Korean GSD (ko)	UDPipe	87.7	84.24
	UDify	82.74	74.26
	UDapter	89.39	85.91
	Tower	86.04	81.70
Korean Kaist (ko)	UDPipe	88.42	86.48
	UDify	87.57	84.52
	Tower	88.78	86.11
Korean PUD (ko)	UDify	63.57	46.89
	Tower	61.78	38.40
Kurmanji MG (kmr)	UDPipe	45.23	34.32
	UDify	35.86	20.4
	UDapter	26.37	12.1
	Tower	72.00	51.02
Latin ITTB (la)	UDPipe	91.06	88.8
	UDify	92.43	90.12
	Tower	91.25	87.67
Latin PROIEL (la)	UDPipe	83.34	78.66
	UDify	84.85	80.52
	Tower	83.74	77.75
Latin Perseus (la)	UDPipe	71.2	61.28
	UDify	78.33	69.6
	Tower	73.53	62.16
Latvian LVTB (lv)	UDPipe	87.2	83.35
	UDify	89.33	85.09
	Tower	92.26	88.52
Lithuanian ALKSNIS (lt)	Tower	87.35	81.58
Lithuanian HSE (lt)	UDPipe	51.98	42.17
	UDify	79.06	69.34
	Tower	79.25	65.47
Livvi KKPP (olo)	UDapter	57.86	43.34
	Tower	62.77	44.62
Maltese MUDT (mt)	UDPipe	84.65	79.71
	UDify	83.07	75.56
	Tower	76.64	67.31
Marathi UFAL (mr)	UDPipe	70.63	61.41
	UDify	79.37	67.72
	UDapter	61.01	44.4
	Tower	70.39	57.77
Mbya Guarani Dooley (gun)	Tower	18.10	5.82
Mbya Guarani Thomas (gun)	Tower	32.36	11.23
Moksha JR (mdf)	UDapter	40.15	26.55
	Tower	44.21	27.45
Naija NSC (pcm)	UDify	45.75	32.16
	UDapter	49.24	36.72
	Tower	52.03	34.95
North Sami Giella (sme)	UDPipe	78.3	73.49
	UDify	74.3	67.13
	Tower	53.53	42.05
Norwegian Bokmaal (no)	UDPipe	92.39	90.49
	UDify	93.97	92.18
	Tower	94.77	93.12
Norwegian Nynorsk (no)	UDPipe	92.09	90.01
	UDify	94.34	92.37
	Tower	93.96	91.65
Norwegian NynorskLIA (no)	UDPipe	68.08	60.07
	UDify	75.4	69.6
	Tower	75.43	69.82
Old French SRCMF (fro)	UDPipe	91.74	86.83
	UDify	91.74	86.65
	Tower	89.75	83.48
Persian Seraji (fa)	UDPipe	90.05	86.66
	UDify	89.59	85.84
	Tower	91.29	87.43

Treebank	Method	UAS	LAS
Polish LFG (pl)	UDPipe	96.58	94.76
	UDify	96.67	94.58
	Tower	97.06	95.18
Polish PDB (pl)	Tower	94.99	89.95
Polish PUD (pl)	Tower	94.13	87.44
Portuguese Bosque (pt)	UDPipe	91.36	89.04
	UDify	91.37	87.84
	Tower	91.50	88.29
Portuguese GSD (pt)	UDPipe	93.01	91.63
	UDify	94.22	92.54
	Tower	93.80	91.98
Portuguese PUD (pt)	UDify	87.02	80.17
	Tower	87.27	77.86
Romanian Nonstandard (ro)	UDPipe	89.12	84.2
	UDify	90.36	85.26
	Tower	90.59	84.41
Romanian RRT (ro)	UDPipe	91.31	86.74
	UDify	93.16	88.56
	Tower	93.61	87.70
Romanian SiMoNERo (ro)	Tower	91.19	86.75
Russian GSD (ru)	UDPipe	88.15	84.37
	UDify	90.71	86.03
	Tower	91.85	88.28
Russian PUD (ru)	UDify	93.51	87.14
	Tower	94.59	88.26
Russian SynTagRus (ru)	UDPipe	93.8	92.32
	UDify	94.83	93.13
	UDapter	94.04	92.24
	Tower	95.28	93.75
Russian Taiga (ru)	UDPipe	75.45	69.11
	UDify	84.02	77.8
	Tower	84.83	77.71
Sanskrit UFAL (sa)	UDify	40.21	18.56
	UDapter	44.32	22.22
	Tower	63.05	44.66
Scottish Gaelic ARCOSG (gd)	Tower	81.32	73.82
Serbian SET (sr)	UDPipe	92.7	89.27
	UDify	95.68	91.95
	Tower	94.36	90.93
Slovak SNK (sk)	UDPipe	89.82	86.9
	UDify	95.92	93.87
	Tower	93.77	90.87
Slovenian SSJ (sl)	UDPipe	92.96	91.16
	UDify	94.74	93.07
	Tower	94.91	93.50
Slovenian SST (sl)	UDPipe	73.51	67.51
	UDify	80.37	75.03
	Tower	78.64	73.10
Spanish AnCora (es)	UDPipe	92.34	90.26
	UDify	92.99	90.5
	Tower	92.67	90.44
Spanish GSD (es)	UDPipe	90.71	88.03
	UDify	90.82	87.23
	Tower	92.12	89.64

Treebank	Method	UAS	LAS
Spanish PUD (es)	UDify	90.45	83.08
	Tower	89.66	80.23
Swedish LinES (sv)	UDPipe	86.07	81.86
	UDify	88.77	85.49
	Tower	88.63	85.07
Swedish PUD (sv)	UDify	89.17	86.1
	Tower	89.20	84.95
Swedish Talbanken (sv)	UDPipe	89.63	86.61
	UDify	91.91	89.03
	UDapter	92.62	90.26
	Tower	89.70	86.60
Swedish Sign Language SSLC (swl)	UDPipe	50.35	37.94
	UDify	40.43	26.95
	Tower	31.56	20.57
Swiss Ger. UZH (gsw)	UDapter	59.74	45.49
	Tower	55.61	40.17
Tagalog TRG (tl)	UDify	64.04	40.07
	UDapter	84.78	69.52
	Tower	91.78	74.32
Tamil TTB (ta)	UDPipe	74.11	66.37
	UDify	79.34	71.29
	UDapter	70.28	46.05
	Tower	71.28	64.36
Telugu MTG (te)	UDPipe	91.26	85.02
	UDify	92.23	83.91
	UDapter	83.52	71.1
	Tower	90.43	81.97
Thai PUD (th)	UDify	49.05	26.06
	Tower	78.23	53.80
Turkish GB (tr)	Tower	75.36	59.39
Turkish IMST (tr)	UDPipe	74.19	67.56
	UDify	74.56	67.44
	UDapter	76.97	69.63
	Tower	77.90	70.00
Turkish PUD (tr)	UDify	67.68	46.07
	Tower	62.29	41.57
Ukrainian IU (uk)	UDPipe	88.29	85.25
	UDify	92.83	90.3
	Tower	92.54	89.89
Upper Sorbian UFAL (hsb)	UDPipe	45.58	34.54
	UDify	71.55	62.82
	UDapter	62.28	54.2
	Tower	70.98	60.90
Urdu UDTB (ur)	UDPipe	87.5	81.62
	UDify	88.43	82.84
	Tower	87.43	81.62
Uyghur UDT (ug)	UDPipe	78.46	67.09
	UDify	65.89	48.8
	Tower	79.11	66.41
Vietnamese VTB (vi)	UDPipe	70.38	62.56
	UDify	74.11	66.0
	Tower	72.40	63.50
Warlpiri UFAL (wbp)	UDify	21.66	7.96
	UDapter	24.2	12.1
	Tower	31.85	16.24
Welsh CCG (cy)	UDapter	70.75	54.43
	Tower	77.22	57.56
Wolof WTB (wo)	Tower	69.06	58.13

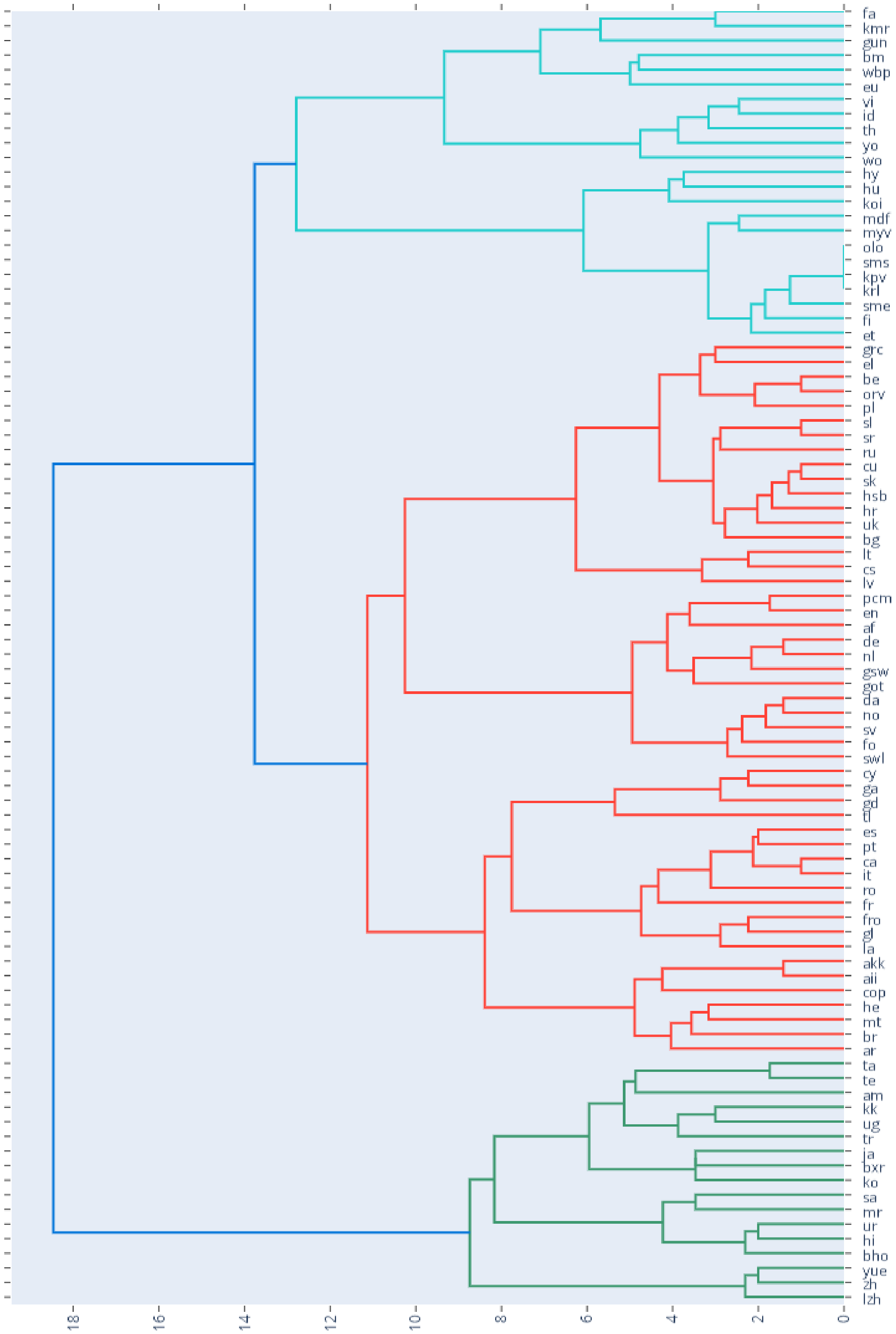

Figure 3: Dendrogram of the full syntax-based hierarchical clustering of 89 languages from UD v2.5. Languages are denoted with their ISO 639-1 codes.

Enhancing the Open-Domain Dialogue Evaluation in Latent Space

Zhangming Chan[1,3,*], **Lemao Liu**[2], **Juntao Li**[4], **Haisong Zhang**[2],
Dongyan Zhao[1,3], **Shuming Shi**[2] and **Rui Yan**[5]
[1] Wangxuan Institute of Computer Technology, Peking University
[2] Tencent AI Lab [3] Center for Data Science, AAIS, Peking University
[4] School of Computer Science and Technology, Soochow University
[5] Gaoling School of Artificial Intelligence, Renmin University of China
`{zhangming.chan,zhaody}@pku.edu.cn`
`ljt@suda.edu.cn, ruiyan@ruc.edu.cn`
`{redmondliu,hansonzhang,shumingshi}@tencent.com`

Abstract

The notorious one-to-many nature of open-domain dialogues poses huge challenges for automatic evaluation methods. Recent studies attempt to mitigate this issue by considering the similarity of the generated response with the conversational context and design discriminative models to learn from multiple positive responses. Despite the promising results, they can not be applied to general scenarios where training data with multiple responses is unavailable. To this end, in this paper, we propose a self-supervised setting to obtain a smooth latent space that can both capture discourse-level context information and implicitly model more references in latent space. Specifically, we present EMS, an **E**nhanced dialogue evaluation **M**etric in latent **S**pace. Experimental results on two real-world dialogue datasets confirm the superiority of our method for open-domain dialogue evaluation, where both Pearson and Spearman correlations with human judgments outperform all baselines.

1 Introduction

With the surge of deep learning techniques, generation-based open-domain dialogue systems have witnessed significant improvement in recent years. Plenty of novel and effective models (Sutskever et al., 2014; Serban et al., 2016; Li et al., 2015; Serban et al., 2016; Zhao et al., 2017; Gu et al., 2018; Qiu et al., 2019; Chan et al., 2019b; Serban et al., 2017; Wolf et al., 2019; Hu et al., 2019; Chen et al., 2020) are proposed and have greatly promoted the development of the open-domain dialogue generation. Unlike the endless emergence of novel methods, however, there is still no meaningful and widely accepted automatic evaluation metric for dialogue generation yet. As we know, automatic evaluation allows quick and effective comparison between different systems and is crucial for the development of natural language generation (NLG) tasks (Dathathri et al., 2019; Gu et al., 2019; Gao et al., 2019; Chan et al., 2019a, 2020). The lack of meaningful automatic evaluation metrics has become a significant impediment for open-domain dialog generation research.

Over the past decade, many automatic evaluation metrics are proposed to evaluate the open-domain dialogue systems. Among them, the word overlap-based automatic evaluation metrics from NLG tasks, such as BLEU (Papineni et al., 2002) in machine translation and ROUGE (Lin, 2004) in text summarization, are popular. In addition, Embedding Metrics (Mitchell and Lapata, 2008; Forgues et al., 2014; Rus and Lintean, 2012) have been utilized to evaluate the open-domain dialogue systems (Gu et al., 2018; Chan et al., 2019b; Shen et al., 2018). Recently, with the fantastic development of the large-scale pre-training model (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2019), researchers proposed to enhance the embedding metrics by converting the dialogue sentences to hidden space via pre-training model (Zhang et al., 2019; Sellam et al., 2020; Zhao et al., 2019; Xiang et al., 2021). The common idea behind these metrics is that they measure the semantic similarity between a reference response and a generated response, independent on the conversational context. However, due to the notorious one-to-many nature (Li et al., 2015; Zhao et al., 2017; Qiu et al., 2019; Gu et al., 2018) of open-domain dialogue, a good response should be related well to its context yet may be largely different from a reference response in semantics.

Some other works (Tao et al., 2018; Ghazarian et al., 2019; Sinha et al., 2020) thereby proposed to build automatic dialogue evaluation metrics by considering the similarity of the generated

---

[*] This work was done while Z. Chan was an intern at Tencent AI Lab. Corresponding Author: Rui Yan.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4889–4900
August 1–6, 2021.

responses with the conversational context. Specifically, these works design discriminative models which can judge whether the generated responses match the conversational context well, which learn from $\{$*conversational context*, *response reference*, *negative sample*$\}$ pairs in unsupervised learning manner. Zhao et al. (2020) further proposed to enhance such discriminative evaluation metrics by finetuning on a few human-annotated data to improve the robustness. These discriminative metrics trained using a single relevant response and multiple negative samples. However, Sai et al. (2020) argued that such discriminative metrics should be trained on multiple relevant responses (i.e., positive samples) and multiple negative samples, to favor the one-to-many nature in open-domain dialogues. Therefore, they collected a new dataset which contains multiple relevant and irrelevant responses for any given conversational context to train their discriminative evaluation model and the model trained by multiple relevant responses shows impressive performance. However, there are no organized relevant multiple responses in most existing datasets. Collecting a new dataset is expensive and time-consuming. Thus, we aim to learn multiple reference information with limited data.

Inspired by the impressive effectiveness of the Variational Auto-encoder (VAEs) and Conditional Variational Auto-encoder (CVAEs) on the representation learning and dialogue modeling, we propose to learn the dialogue representations via VAEs/CVAEs for better evaluation. Equip with such dialogue representations, we obtain an **E**nhanced dialogue evaluation **M**etric in latent **S**pace (EMS). EMS is a self-supervised evaluation metric with a two-stage training procedure. It represents dialogue sentences in a smooth latent space to both capture discourse-level context information and model more feasible latent references. Specifically, in the first stage, we build a VAE based model to map the dialogue sentences into a latent (or semantic) space. Li et al. (2019) showed that VAEs can be viewed as a regularized version of the auto-encoder and learn a smooth latent space through the regularization from the Gaussian prior. Then, we train our model by optimizing CVAEs' objective which forces the prior distribution to capture the feasible latent references information (details in Section 3.3). In the second stage, we combine the dialogue representations and the captured feasible latent reference information to train a discriminative model. Meanwhile, we give a potential explanation of our motivation about why using feasible latent reference information can lead to a better evaluation (details in Section 3.1). Experimental results on two real-world dialogue datasets confirm the superiority of our method for open-domain dialogue evaluation, where both Pearson and Spearman correlations with human judgments outperform all baseline methods.

In a nutshell, our contributions can be summarized as follows:

- We proposed a novel automatic evaluation metric, i.e., EMS, for open-domain dialogue systems;
- We proposed a pre-training variational model to capture the feasible latent references;
- Experiments performed on two large datasets demonstrate the effectiveness of our proposed model and outperform all baseline methods.

2 Related Work

Word overlap-based Metrics. Several word overlap-based automatic evaluation metrics, such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and ROUGE (Lin, 2004), have been widely used to evaluate the quality of generated responses. These word overlap-based metrics measure how many words overlap in a given generated response when compared to a reference response. Liu et al. (2016); Lowe et al. (2017); Tao et al. (2018) argued that these word overlap-based metric scores are weakly correlated to human judgment due to ignoring the notorious one-to-many nature of the open-domain dialogues. Therefore, Yuma et al. (2020) proposed the improved BLEU, which compares the generated response with multiply diverse references.

Embedding-based Metrics. Unlike word overlap-based metrics comparing two raw sentences, Embedding Metrics (Mitchell and Lapata, 2008; Forgues et al., 2014; Rus and Lintean, 2012) map sentences to a high dimensional space, and calculate similarity based on the high-dimensional representations. Embedding Metrics are recently popular for evaluating the generation tasks, such as text summarization (Gao et al., 2020; Chen et al., 2021), question answer (Gao et al., 2019) and text generation (Hashimoto et al., 2019; Chan et al., 2020). Meanwhile, several works (Qiu et al., 2019; Chen et al., 2020; Gao et al., 2021) have shown their effectiveness in the open-domain dialogue systems. With the development of the large-scale pre-training model (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2019), some studies, e.g., BERTScore (Zhang et al., 2019) and MoverScore (Zhao et al., 2019), further enhance the quality of representations via a large-scale pre-training model. However, these embedding-based metrics perform a better comparison compared to word overlap-based metrics but still ignore the one-to-many nature of open-domain dialogues.

Learning-based Metrics. Recent studies (Tao et al., 2018; Sinha et al., 2020) attempt to mitigate the one-to-many issue by considering the similarity of the generated response with the conversational contexts. The similarity is calculated by a designed discriminative model which learns to evaluate whether a response matches the conversational context well. The discriminative model is learned from tuples of data, {*conversational context*, *response reference*, *negative sample*}, in an unsupervised learning manner. However, these learning-based metrics rely on a sophisticated sampling technique. Lan et al. (2020) proposed a sampling strategy to collect the valuable negative samples for the discriminative training. Bak and Oh (2020) conduct speaker sensitive response evaluation by conducting negative sampling from several levels. To further improve the robustness, Zhao et al. (2020) proposed to enhance the discriminative model by finetuning on a few human-annotated data. Sai et al. (2020) argued that these discriminative metrics should be trained on multiple relevant responses and multiple irrelevant samples for any given context. Therefore, they collected such a dataset and improved the evaluation performance greatly. However, collecting a new dataset is expensive and time-consuming. In this work, we propose a method to improve the effectiveness of the discriminative metrics based on the VAEs/CVAEs.

3 Methodology

In this paper, we propose an Enhanced dialogue evaluation Metric in latent Space (EMS), which contains two training stages (illustrated in Fig. 2). In this section, we first conduct some discussions about our motivations in Section 3.1. Then, we introduce the overall architecture in Section 3.2. The two training stages are described in Section 3.3 and Section 3.4, respectively. Finally, we describe the inference process in Section 3.5.

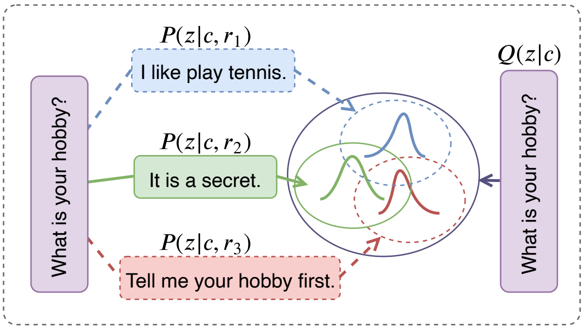

Figure 1: Distributions in latent space. Each circle represents a Gaussian distribution while three small circles refer to a special Gaussian distribution for each corresponding response. Naturally, the biggest circle indicates the prior Gaussian distribution. We use the prior distribution to approximate all the response conditional distribution. Dotted lines indicate the latent response.

3.1 Discussion about Motivation

We conduct a discussion about our motivation from information theory. Let r_i denotes a feasible response coming from $\{r_k\}_{k=1}^N$ which represents N feasible latent references. Assume a binary label $l \in \{0, 1\}$ indicates whether a response matches its context well. Existing works (Tao et al., 2018; Ghazarian et al., 2019; Sinha et al., 2020; Zhao et al., 2020) training with single relevant response are actually maximizing $I(l; c, r_i)$. Recently, Sai et al. (2020) proposed to training with multiple relevant responses, which actually maximizes $I(l; c, \{r_k\}_{k=1}^N)$. An intuitive explanation for the surprising improvement in Sai et al. (2020) is that $I(l; c, \{r_k\}_{k=1}^N) \geq I(l; c\ , r_i)$[1].

However, there are no organized relevant multiple responses in existing datasets and collecting a new dataset is expensive and time-consuming. Therefore, we aim to capture the feasible latent reference information with limited data. Inspired by previous works which model multiple responses for dialogue (Zhao et al., 2017; Qiu et al., 2019; Chan et al., 2019b), we utilize CVAEs (details in Section 3.2) which build a prior distribution $P(z|c)$ to capture the feasible latent reference information in the latent space. Specifically, when training CVAEs, $P(z|c)$ is forced to be close to the posterior distribution $Q(z|c, r_i)$ for any reference response r_i as illustrated in Fig. 1. In this sense, if z is sampled from $P(z|c)$, z may contain some information of any r_i in some extent, and z can be used as a surrogate of $\{r_k\}_{k=1}^K$. Therefore, we can expect

---

[1] There is a brief proof in Apendix A.

$I(l; c, \{r_k\}_{k=1}^{N}) \geq I(l; c, r_i, z) \geq I(l; c, r_i)$.

3.2 Overall Architecture

Previous works (Li et al., 2019; Gururangan et al., 2019; Li et al., 2020) concluded that VAEs can learn a smooth latent space through the regularization from the gaussian prior. Inspired by Li et al. (2020), we propose a novel architecture which can be regarded as a large-scale pretrained language model (PLM) based on VAEs/CVAEs.

Encoder. Li et al. (2019) argue that the VAEs might benefit from initialization with a non-collapsed encoder, because the encoder provides useful information from the beginning of training. We use the Masked PLMs (Devlin et al., 2018; Liu et al., 2019) as the text encoder because of their impressive effectiveness in natural language understanding tasks. We describe the encoding process as following,

$$\begin{aligned} h_q &= \text{PLM}([c; r]), \\ h_p &= \text{PLM}(c) \end{aligned} \tag{1}$$

where c, r indicate conversational context and response reference, respectively.

Latent Variable Modeling. For modeling the latent variable, we hypothesize that the approximated variational prior and posterior follows an isotropic multivariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2 I)$, where I represents the diagonal covariance. We use a recognition network $q_\phi(z|h_q)$ and a prior network $p_\theta(z|h_p)$ to approximate the posterior $Q(z|c, r)$ and the prior $P(z|c)$, respectively.

Decoder. The reconstruction process[2] forces the latent variable to contain the useful posterior information, which is a crucial step in the variational training. We use another PLM as the decoder to reconstruct the original input texts. For transporting the latent variable to the PLM decoder, we use the memory mechanism mentioned in Li et al. (2020) where the latent variable plays the role of an additional memory vector for the PLM decoder to attend. Specifically, the latent variable z is converted through a Multilayer Perceptron (MLP) and separated into several vectors, each of which is transported to the PLM decoder via attention mechanism.

3.3 Stage 1: Representation in Latent Space

Our first stage is to learn the latent representation of the dialogues and capturing the feasible latent reference information. Specifically, we first optimize our model via VAEs' objective to model a smooth latent space. Then, we train our model by CVAEs' objective to capture the feasible latent reference information. We describe the details as following.

A smooth latent space. Following Li et al. (2020), we first train the posterior module by optimizing the VAEs' objective. Li et al. (2019) showed that VAEs can be viewed as a regularized version of the autoencoder and can learn a smooth latent space. Based on this, we convert sentences in a universal smooth latent space. In a smooth latent space, latent representation of similar sentences should be close to each other and vice versa (Li et al., 2019). Therefore, it is a great outset for our model. To train this model, the log-likelihood objective is maximized through pushing up its variational lower bound:

$$\begin{aligned} &- \mathbf{KL}(p_\theta(z|h_q)||q(z)) \\ &+ \mathbf{E}_{p_\theta(z|h_q)}[\log p_\theta([c; r]|z, h_q)], \end{aligned} \tag{2}$$

where $\mathbf{KL}(\cdot)$ represents the KL-divergence term, which serves as the regularization that encourages $p_\theta(z|h_q)$ to approach to $q(z)$, i.e., a standard Gaussian distribution; $\mathbf{E}[\cdot]$ is the term of reconstruction loss, reflecting how well the decoder performs.

Feasible latent reference information. The one-to-many nature of the open-domain dialogues poses that there can be a lot of reasonable responses for the same conversational context. Therefore, we handle this one-to-many nature by CVAE as previous works (Zhao et al., 2017; Gu et al., 2018; Chan et al., 2019b) to capture the feasible latent reference information. As shown in Fig. 1, CVAE builds Gaussian posterior distributions for each feasible reference and forces the prior distribution to approach the posterior distributions. Ideally, a well-learned prior distribution will cover all the feasible latent reference information. We train our model by optimizing the following variational lower bound:

$$\begin{aligned} &- \mathbf{KL}(q_\phi(z|h_q)||p_\theta(z|h_p)) \\ &+ \mathbf{E}_{q_\phi(z|h_q)}[\log p_\theta([c; r]|z, h_q)] \end{aligned} \tag{3}$$

where $\mathbf{KL}(\cdot)$ represents the KL-divergence term, which serves as the regularization that encourages the prior $p_\phi(z|h_p)$ to approach the approximated

---

[2] Note that the reconstruction process doesn't only indicate the autoregressive generation but also the masked causal generation.

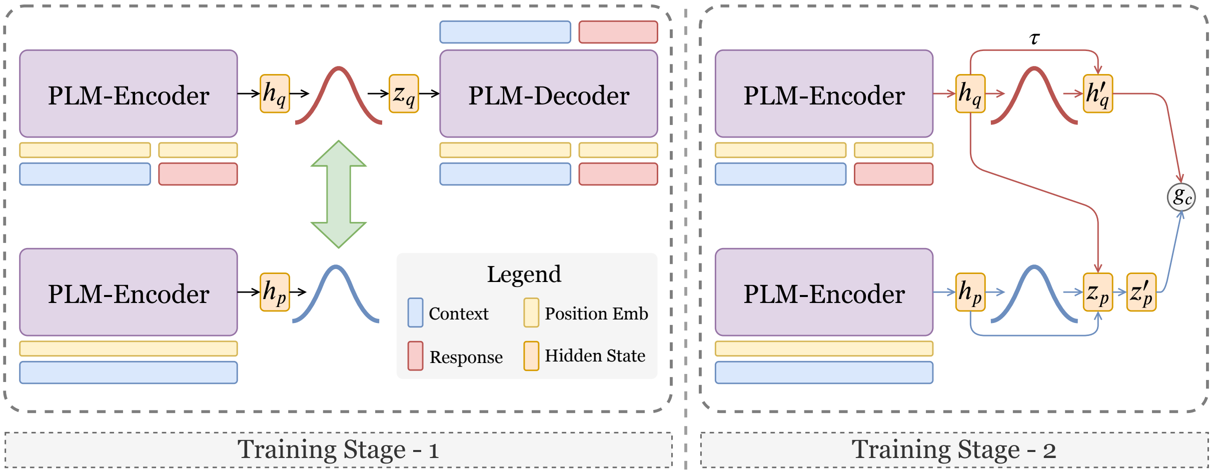

Figure 2: The detailed architecture of our proposed EMS and there are two stages in the training process. The first and second training stage is described in Section 3.3 and Section 3.4, respectively. In the second training stage, the response input is replaced by the generated responses which need be evaluated. The *Position Emb* indicates the Position Embedding.

posterior $q_\theta(z|h_q)$, i.e. a conditional Gaussian distribution.

Training procedures. Previous works (Bowman et al., 2015; Zhao et al., 2017) mentioned that VAE and CVAE training is challenging due to the KL vanishing issue, where the decoder ignores the conditional information and all the resulting posteriors almost collapse to a same Gaussian prior. To mitigate this issue, first, we initialize our model with Optimus (Li et al., 2020), a large-scale VAE-based PLM model, while optimizing Eq. 2.

To mitigate the same issue while optimizing Eq. 3, we use the cyclical KL annealing schedule (Fu et al., 2019). Specifically, we add a hyperparameter α to control the weight of the KL-divergence in Eq. 3. We set α close to zero in the first half of cyclic schedule, linearly anneal α to 1 in the next one-fourth of cyclic schedule and kept α = 1 in the remaining cyclic schedule.

Moreover, the Free Bits (Bowman et al., 2015) is also crucial for the training. It replaces the KL-divergence in Eq. 3 by a hinge loss

$$\max(\gamma, \mathbf{KL}(q_\phi(z|h_q)||p_\theta(z|h_p))) \tag{4}$$

where γ is a hyperparameter which controls the information space for the each dimension of the latent variable. Finally, an extra bag-of-word loss (Zhao et al., 2017) is also used during the training.

3.4 Stage 2: Matching Training

In the second stage, we learn to judge the similarity between the conversational context and the response using the learned representations. Li et al. (2020) argue that the KL regularization applied on z has a large impact on the preceding layer feature, thus, the preceding layer feature also contains the information of z. Therefore, we consider combining h_q and z into the final representation,

$$h'_q = (1-\tau)\cdot h_q + \tau \cdot z_q, \tag{5}$$

where τ is a hyperparameter and z_q indicates the latent representation from the posterior network. Meanwhile, we use the feasible latent reference information, captured by our prior network, to enhance the matching. We combine these two representations as following,

$$\begin{aligned} e &= \sigma(\mathrm{W}_g \cdot [h_p; h_q; z_p] + \mathrm{b}_g) \\ z'_p &= e \cdot z_p + (1-e)\cdot h_p \end{aligned} \tag{6}$$

where W_g and b_g are trainable parameters, and e is learned by the gate mechanism that controls the fusion of z_p and h_p. Note that z_p indicates the latent representation from the prior network. The activation function σ is sigmoid. Finally, we infer matching score between the conversational context and generated reference as follows,

$$g_c = \sigma(\mathrm{W}_s \cdot ([h'_q; z'_p]) + \mathrm{b}_s) \tag{7}$$

where W_s and b_s are trainable parameters and the activation function σ is sigmoid. Finally, we optimize our model with positive sampling and negative sampling (Lan et al., 2020) based on the discriminative training scheme.

3.5 Inference

In the inference process, we input the conversational context and response candidate as the c and r in Eq. 1, and conduct the operation as Eq. 5, Eq. 6 and Eq. 7 to obtain the score g_c. We use the g_c as matching degree of the response candidate.

4 Experiments

4.1 Dataset

To evaluate the effectiveness of our proposed automatic evaluation metric EMS, we conduct experiments on the following two open-access datasets.

Persona-Chat Dataset. The persona-chat dataset (Zhang et al., 2018) is a large persona-conditioned chit-chat style dialogue dataset which consists of 10,907 multi-turn dialogue sessions[3].

DailyDialog Dataset. The dailydialog dataset (Li et al., 2017) is an another widely-used large collection of human-human dialogues which consists of 13,118 multi-turn dialogue sessions[4].

Human-annotated Dataset. We collect the human-annotated datasets from Amazon Mechanical Turk and obtain two human-annotated datasets which consist of 750 context-response pairs in the persona-chat dataset and 800 ones in the dailydialog dataset, respectively. Following Zhao et al. (2020), the generated references come from several classical dialogue models, i,e., Seq2Seq (Sutskever et al., 2014), Seq2Seq with Attention (Serban et al., 2016), HRED (Serban et al., 2017), VHRED (Serban et al., 2016), GPT-2 (Wolf et al., 2019).

4.2 Baselines

We compare our proposed method with the following highly related and strong baselines.

BLEU. We utilize BLEU score (Papineni et al., 2002) to measure n-grams overlaps between response reference and generated response. Specifically, we follow the conventional setting in Sinha et al. (2020) and use the multi-bleu[5].

ROUGE. ROUGE (Lin, 2004) is the most popular metric in the text summarization task. There, we report the f1 score of Rouge-$\{$1,2,3,L$\}$ and Rouge-L identifies the longest common subsequence between the generated and reference to better account for sentence-level structure when computing word overlap.

METEOR. The METEOR (Banerjee and Lavie, 2005) is designed as an improvement on BLEU using a harmonic mean of precision and recall, as well as stemming and synonyms.

Embedding Metrics. Embedding Metrics compute the similarity between the embeddings representations of generated results and reference. The used embeddings come from glove[6]. In particular, we calculate three metrics: 1) *Average*, cosine similarity between the averaged word embeddings in the two sentences (Mitchell and Lapata, 2008); 2) *Extrema*, cosine similarity between the largest extreme values among the word embeddings in the two sentences (Forgues et al., 2014); 3) *Greedy*, i.e., greedily matching words in two sentences based on the cosine similarities, and the total scores are then averaged across all words (Rus and Lintean, 2012).

BERTScore. BERTScore (Zhang et al., 2019) uses a strong PLM model to greedily match each word in a reference response with one word in the generated response. By doing so, it computes the recall of the generated sequence. BERTScore was shown to have strong system-level and segment-level correlation with human judgment on several machine translation tasks.

BLEURT. BLEURT (Sellam et al., 2020) is based on the BERTScore and finetuned on human judgments after pretraining on large-scale synthetic data with multiple automatic metrics as supervision signals. BLEURT has shown its strong correlation with human judgment on machine translation tasks.

RUBER. RUBER (Tao et al., 2018) is an unsupervised automatic evaluation metric that considering the similarity of the generated response with conversational context and response reference.

MAUDE. MAUDE (Sinha et al., 2020) proposed an unreferenced automated evaluation metric that uses large-scale PLMs to extract hidden representations of dialogue sentences, and leverages the temporal transitions that exist between them.

4.3 Settings

The dimension of latent variable z is set to 768 to improve the information bottleneck. As we mentioned before, the encoder and the decoder in our model are BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019), respectively. We use the

---

[3] `http://parl.ai`

[4] `http://yanran.li/dailydialog`

[5] `https://github.com/OpenNMT/OpenNMT-py/blob/master/tools/multi-bleu.perl`

[6] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

Tokenizer of BERT to conduct the texts for BERT and tokenizer of GPT-2 for GPT-2 in all experiments. We use the Optimus (Li et al., 2020) to save the time consumption of the VAE training. The recognition network and prior network are consist of 3 layers MLP with Dropout layer and GELU function. When training, we set the mini-batch size to 16. The AdamW optimizer is used to train the VAEs module with the initial learning rate 5e-5, and the learning rate warmup and decay strategy is employed. The value of τ in Eq. 5 is set to 0.01. When we conduct the matching training, we change the initial learning rate to 3e-6, and the learning rate warmup and decay strategy is also employed.

5 Results and Analysis

5.1 Overall Performance

We examine the performance of our model compared with baselines on two open-access datasets and the results.

The word-overlap metrics based on n-gram perform worst. As shown in Table 1 and Table 2, the word-overlap evaluation metrics, i.e., BLEU, ROUGE, and METEOR, obtain the worst performance in the dialogue evaluation on both two datasets. Among them, BLEU (hybrid) scores on two datasets are both less than 0.1, though it is the most widely used metric in machine translation. Intuitively, the information from n-gram is more accurate with a larger n (the most accurate information comes from the whole sentence). However, as the results shown in the Table 1 and Table 2, the correlation score decays when n increases. The same phenomenon is observed when using ROUGE. It seems using n-grams as the representation of the dialogue sentence is not a good choice.

PLM is an effective representation extractor for dialogue sentences. From Table 1 and Table 2, we can see that most embedding-based metrics, i.e., Average, Extrema, Greedy, BERTScore and BLEURT, using pretrained embedding to represent the sentences, perform better than word-overlap metrics which uses n-grams as representation. Furthermore, traditional embedding-based metrics with Glove-based embedding, i.e., Average, Extrema, Greedy, perform worse than the embedding-based metrics with PLM-based embedding, i.e., BERTScore and BLEUET. Thus, we can know that using PLM to represent the dialogue sentence is more effective for the evaluation.

Table 1: Performance of our EMS methods and all baselines evaluated on the Persona-Chat Dataset. The BERTScore and BLEURT are built based on the base and large version of the Roberta (Liu et al., 2019).

Metrics	**Pearson's ρ**	**Spearmans's r**
BLEU		
Hybrid	0.0602 (0.100)	0.0623 (0.088)
1-gram	0.2486 (<0.001)	0.2201 (<0.001)
2-gram	0.1991 (<0.001)	0.1891 (<0.001)
3-gram	0.0998 (<0.001)	0.1138 (<0.001)
4-gram	0.0558 (0.127)	0.0622 (0.002)
ROUGE		
1-gram	0.2663 (<0.001)	0.2553 (<0.001)
2-gram	0.1706 (<0.001)	0.1715 (<0.001)
3-gram	0.1062 (0.004)	0.1157 (0.001)
L	0.2777 (<0.001)	0.2619 (<0.001)
METEOR	0.2507 (0.033)	0.2297 (0.020)
Embedding Metrics		
Average	0.1330 (<0.001)	0.1452 (<0.001)
Extrema	0.2354 (<0.001)	0.2272 (<0.001)
Greedy	0.2585 (<0.001)	0.2502 (<0.001)
BERTScore		
Base	0.2943 (<0.001)	0.2781 (<0.001)
Large	0.2782 (<0.001)	0.2619 (<0.001)
BLEURT		
Base	0.3560 (<0.001)	0.3482 (<0.001)
Large	0.3045 (<0.001)	0.2857 (<0.001)
RUBER	0.4581 (<0.001)	0.4570 (<0.001)
MAUDE	0.5619 (<0.001)	0.5631 (<0.001)
EMS	**0.5856** (<0.001)	**0.5921** (<0.001)

Learning-based discriminative metrics outperform training-free metrics. From Table 1 and Table 2, we can observe that all the learning-based discriminative metrics (i.e., RUBER, MAUDE, and our EMS) outperform all other training-free metrics (i.e., BLEU, ROUGE, METEOR, Average, Extrema, Greedy, BERTScore, and BLEURT) in both the Pearson and Spearman correlation with human judgment. Therefore, we conclude the learning-based discriminative metrics are very powerful to evaluate the open-domain dialogue systems.

Our proposed EMS metric performs the best. Our EMS metric achieves the best performance with 0.5856, 0.5921 in Pearson and Spearman correlation with the human judgment on the persona-chat dataset, respectively. Meanwhile, on the dailydialog dataset, EMS obtains 0.5331, 0.5253 in Pearson and Spearman score. These experimental

Table 2: Performance of our EMS methods and all baselines evaluated on the DailyDialog Dataset. The BERTScore and BLEURT are built based on the base and large version of the Roberta (Liu et al., 2019).

Metrics	**Pearson's ρ**	**Spearmans's r**
BLEU		
Hybrid	0.0240 (0.498)	0.0211 (0.551)
1-gram	0.2285 (<0.001)	0.2050 (<0.001)
2-gram	0.1115 (<0.001)	0.0812 (<0.001)
3-gram	0.0364 (0.304)	0.0401 (0.257)
4-gram	0.0261 (0.461)	0.0211 (0.551)
ROUGE		
1-gram	0.1424 (<0.001)	0.0921 (0.009)
2-gram	0.1226 (<0.001)	0.1104 (0.002)
3-gram	0.0534 (0.131)	0.0467 (0.187)
L	0.1401 (<0.001)	0.0901 (0.011)
METEOR	0.2095 (<0.001)	0.1809 (<0.001)
Embedding Metrics		
Average	0.0272 (0.443)	0.0440 (0.214)
Extrama	0.1763 (<0.001)	0.1652 (<0.001)
Greedy	0.1311 (<0.001)	0.1186 (<0.001)
BERTScore		
Base	0.1139 (0.001)	0.1132 (0.001)
Large	0.1273 (<0.001)	0.1251 (<0.001)
BLEURT		
Base	0.3478 (<0.001)	0.3356 (<0.001)
Large	0.2260 (<0.001)	0.1916 (<0.001)
RUBER	0.4328 (<0.001)	0.4310 (<0.001)
MAUDE	0.5162 (<0.001)	0.5343 (<0.001)
EMS	**0.5331** (<0.001)	0.5253 (<0.001)

results show our method outperforms all existing baselines, indicating the superiority of our method.

5.2 Analysis

Our model aims to enhance the dialogue evaluation via variational training. Hence, in this subsection, we examine whether variational training can improve the performance by ablation study.

First, we replace the hidden representation ("w/o q") in Eq. 5 by the one from pure BERT, i.e., CLS. From the performance in Table 3, the KL regularization enhances the performance of EMS metric on both two datasets which proves a smooth latent space (via VAE training) is important. Second, as shown in Table 3, without z_p in Eq. 6 ("w/o p") which captures the feasible latent reference information, EMS gains a performance drop. Therefore, it proves the effectiveness of the feasible latent reference information.

Table 3: Performance of the ablation study. Note that ρ and r indicate the Pearson's ρ and Spearmans's r.

Metrics	**Persona-Chat Dataset**		**DailyDialog Dataset**	
	ρ	r	ρ	r
EMS	0.5856	0.5921	0.5331	0.5253
w/o q	0.5732	0.5741	0.5298	0.5221
w/o p	0.5621	0.5618	0.5193	0.5218

Table 4: Two evalution cases from DailyDialog Dataset. _eos_ indicates the end of the current sentence.

Context:	did you get your bus pass? _eos_ i have n't gone to get it yet. _eos_ why have n't you got it? _eos_ i do n't know where to go to get one. _eos_ i know where to get them from.		
Reference:	where do i get it?		
Generated:	what kind of buses are they on?		
Human (1-5)	**BLEU (0-1)**	**MAUDE (1-5)**	**EMS (1-5)**
3.00	0.00	4.99	3.32
Context:	what a beautiful home! _eos_ you'll notice that the window treatments, carpeting, and drapes are all new. _eos_ i like the way the blinds give you privacy from the street. _eos_ follow me into the kitchen. you will love it. _eos_ i love that they put a wine storage area in the kitchen.		
Reference:	the best part is the bedroom and attached bathroom.		
Generated:	i'm sure you will.		
Human (1-5)	**BLEU (0-1)**	**MAUDE (1-5)**	**EMS (1-5)**
3.50	0.00	1.17	2.29

5.3 Case Study

To explain more intuitively, we show two cases of our experiments in Table 4. In the first case, we can observe the golden score from the human is 3.00, however, MAUDE predicts the score as 4.99. We find the MAUDE gives such a high score because there is a keyword ("what kind of buses are they on ?") in the generated response which also exists in the conversational context. In the second case, MAUDE gives an extremely low score, i.e., 1.17, since no repeated words in the generated response and the context. However, our EMS gives scores similar to the Human Score, 3.32 and 2.29 in the first and second case, respectively. It proves that our EMS is more similar to human evaluation.

6 Conclusion and Future Work

In this study, we propose a two-stage automatic evaluation metric, i.e., EMS, which can obtain a smooth latent space that can both capture discourse-level context information and model more feasible latent references for evaluating the open-domain dialogues. Experimental results on two dialogue datasets confirm the superiority of our method for open-domain dialogue evaluation, where both Pearson and Spearman correlations with human judgments outperform all baseline methods.

Owing to the promising performance of the variational training, we pursue to design the training procedures for better representation in latent space. Besides, we will explore more efficient methods to obtain more useful feasible reference information.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments. This work is supported by the National Key Research and Development Program of China (No. 2020YFB1406702), the National Science Foundation of China (NSFC No. 61876196) and Beijing Outstanding Young Scientist Program No.BJJWZYJH012019100020098. Rui Yan is sponsored by Tencent Collaborative Research Fund. Rui Yan is the corresponding author, and is supported as a Young Fellow of Beijing Institute of Artificial Intelligence (BAAI).

Ethics Impact

In this paper, we propose a two-stage automatic evaluation metric EMS for open-domain dialogue systems. The positive impact lies in that it allows quick and effective comparison between different dialogue systems and is crucial for the development of open-domain dialogue tasks. The negative impact may be that in some extreme cases, the system may give high scores to the rude and dirty responses. Hence, in such situation, the training dataset is crucial and should be examined before employed in practice.

References

JinYeong Bak and Alice Oh. 2020. Speaker sensitive response evaluation model. *arXiv preprint arXiv:2006.07015*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao, and Rui Yan. 2019a. Stick to the facts: Learning towards a fidelity-oriented e-commerce product description generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4960–4969.

Zhangming Chan, Juntao Li, Xiaopeng Yang, Xiuying Chen, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019b. Modeling personalization in continuous space for response generation via augmented wasserstein autoencoders. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, pages 1931–1940.

Zhangming Chan, Yuchi Zhang, Xiuying Chen, Shen Gao, Zhiqiang Zhang, Dongyan Zhao, and Rui Yan. 2020. Selection and generation: Learning towards multi-product advertisement post generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3818–3829.

Wang Chen, Piji Li, and Irwin King. 2021. A training-free and reference-free summarization evaluation metric via centrality-weighted relevance and self-referenced redundancy. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Xiuying Chen, Zhi Cui, Jiayi Zhang, Chen Wei, Jianwei Cui, Bin Wang, Dongyan Zhao, and Rui Yan. 2020. Reasoning in dialog: Improving response generation by context reading comprehension. *arXiv preprint arXiv:2012.07410*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, workshop*, volume 2.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*.

Jun Gao, Wei Bi, Ruifeng Xu, and Shuming Shi. 2021. Ream: An enhancement approach to reference-based evaluation metrics for open-domain dialog generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: Findings*.

Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in e-commerce question-answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 429–437.

Yang Gao, Wei Zhao, and Steffen Eger. 2020. Supert: Towards new frontiers in unsupervised evaluation metrics for multi-document summarization. *arXiv preprint arXiv:2005.03724*.

Sarik Ghazarian, Johnny Tian-Zheng Wei, Aram Galstyan, and Nanyun Peng. 2019. Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. *arXiv preprint arXiv:1904.10635*.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. *arXiv preprint arXiv:1905.11006*.

Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. 2018. Dialogwae: Multimodal response generation with conditional wasserstein autoencoder. *arXiv preprint arXiv:1805.12352*.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A Smith. 2019. Variational pretraining for semi-supervised text classification. *arXiv preprint arXiv:1906.02242*.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. *arXiv preprint arXiv:1904.02792*.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. *arXiv preprint arXiv:1905.13637*.

Tian Lan, Xian-Ling Mao, Wei Wei, Xiaoyan Gao, and Heyan Huang. 2020. Pone: A novel automatic evaluation metric for open-domain generative dialogue systems. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–37.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A surprisingly effective fix for deep latent variable modeling of text. *arXiv preprint arXiv:1909.00868*.

Chunyuan Li, Xiang Gao, Yuan Li, Xiujun Li, Baolin Peng, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. *arXiv preprint arXiv:2004.04092*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ryan Lowe, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. *arXiv preprint arXiv:1708.07149*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *NAACL-HLT*, pages 236–244.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Lisong Qiu, Juntao Li, Wei Bi, Dongyan Zhao, and Rui Yan. 2019. Are training samples correlated? learning to generate dialogue responses with multiple references. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3826–3835.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. ACL.

Ananya B Sai, Akash Kumar Mohankumar, Siddhartha Arora, and Mitesh M Khapra. 2020. Improving dialog evaluation with a multi-reference adversarial dataset and large scale pretraining. *Transactions of the Association for Computational Linguistics*, 8:810–827.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *ACL*.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Xiaoyu Shen, Hui Su, Wenjie Li, and Dietrich Klakow. 2018. Nexus network: Connecting the preceding and the following in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4316–4327.

Koustuv Sinha, Prasanna Parthasarathi, Jasmine Wang, Ryan Lowe, William L Hamilton, and Joelle Pineau. 2020. Learning an unreferenced metric for online dialogue evaluation. *arXiv preprint arXiv:2005.00583*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Jiannan Xiang, Yahui Liu, Deng Cai, Huayang Li, Defu Lian, and Lemao Liu. 2021. Assessing dialogue systems with distribution distances. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: Findings*.

Tsuta Yuma, Naoki Yoshinaga, and Masashi Toyoda. 2020. ublea: Uncertainty-aware automatic evaluation method for open-domain dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 199–206.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *ICLR*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.

Tianyu Zhao, Divesh Lala, and Tatsuya Kawahara. 2020. Designing precise and robust dialogue response evaluators. *arXiv preprint arXiv:2004.04908*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.

A Proof

As well we know, the the mutual information (MI) of X and Y is defined as

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= H(Y) - H(Y \mid X) \end{aligned} \tag{8}$$

where $H(\cdot)$ denotes the entropy.

Obviously, we can get

$$\begin{aligned} H(l \mid c, \{r_k\}_{k=1}^{N}) &= H(l \mid c, r_i, \{r_k\}_{k=1}^{N} \setminus r_i) \\ &\leq H(l \mid c, r_i). \end{aligned} \tag{9}$$

Then, we can compare the MI of using the feasible latent reference information and not as follows

$$\begin{aligned} I(l; c, \{r_k\}_{k=1}^{N}) &= H(l) - H(l \mid c, \{r_k\}_{k=1}^{N}) \\ &\geq H(l) - H(l \mid c, r_i) \\ &= I(l; c, r_i). \end{aligned} \tag{10}$$

where we can observe the MI is enhanced.

Adapting Monolingual Models: Data can be Scarce when Language Similarity is High

Wietse de Vries*, **Martijn Bartelds***, **Malvina Nissim** and **Martijn Wieling**
University of Groningen
The Netherlands
{wietse.de.vries, m.bartelds, m.nissim, m.b.wieling}@rug.nl

Abstract

For many (minority) languages, the resources needed to train large models are not available. We investigate the performance of zero-shot transfer learning with as little data as possible, and the influence of language similarity in this process. We retrain the lexical layers of four BERT-based models using data from two low-resource target language varieties, while the Transformer layers are independently fine-tuned on a POS-tagging task in the model's source language. By combining the new lexical layers and fine-tuned Transformer layers, we achieve high task performance for both target languages. With high language similarity, 10MB of data appears sufficient to achieve substantial monolingual transfer performance. Monolingual BERT-based models generally achieve higher downstream task performance after retraining the lexical layer than multilingual BERT, even when the target language is included in the multilingual model.

1 Introduction

Large pre-trained language models are the dominant approach for solving many tasks in natural language processing. These models represent linguistic structure on the basis of large corpora that exist for high-resource languages, such as English. However, for the majority of the world's languages, these large corpora are not available.

Past work on multilingual learning has found that multilingual BERT (mBERT; Devlin et al. 2019a) generalizes across languages with high zero-shot transfer performance on a variety of tasks (Pires et al., 2019; Wu and Dredze, 2019). However, it has also been observed that high-resource languages included in mBERT pre-training often have a better-performing monolingual model, and low-resource languages that are not included in mBERT pre-training usually show poor performance (Nozza et al., 2020; Wu and Dredze, 2020).

An alternative to multilingual transfer learning is the adaptation of existing monolingual models to other languages. Zoph et al. (2016) introduce a method for transferring a pre-trained machine translation model to lower-resource languages by only fine-tuning the lexical layer. This method has also been applied to BERT (Artetxe et al., 2020) and GPT-2 (de Vries and Nissim, 2020). Artetxe et al. (2020) also show that BERT models with retrained lexical layers perform well in downstream tasks, but comparatively high performance has only been demonstrated for languages for which at least 400MB of data is available.

To test if this procedure is also effective for low- to zero-resource languages, we consider two regional language varieties spoken in the North of the Netherlands, namely Gronings (Low Saxon language variant) and West Frisian.

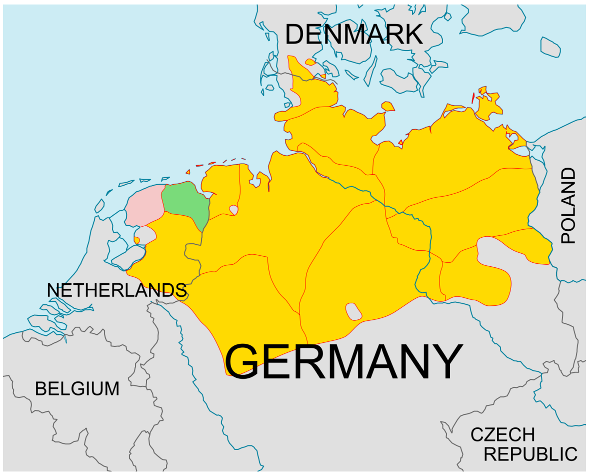

Figure 1: Geographical areas where Gronings (in green) and West Frisian (in red) are spoken. Image modified from https://en.wikipedia.org/wiki/Low_German.

---

[*] These authors contributed equally.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4901–4907
August 1–6, 2021.

Figure 1 visualizes the geographical areas where these regional language variants are spoken. The regional Low Saxon language is spoken in the north-eastern provinces of the Netherlands and in the North of Germany (shown in yellow). As part of the Low Saxon language, Gronings is spoken in the province of Groningen (highlighted in green). The West Frisian language is spoken in the province of Friesland (shown in red), and it is the second official language of the Netherlands, next to Dutch. Dutch is the national language of the Netherlands, and it is spoken in every province of the Netherlands and in Flanders (North of Belgium).

For both Gronings and West Frisian limited data is available. In addition to unlabeled data, for both target languages we have a small collection of annotated part-of-speech (POS) tagging data, which we use for evaluating zero-shot model transfer. We use three monolingual BERT models (source languages English, German, Dutch) and mBERT to investigate if linguistic structure can be transferred to Gronings and West Frisian by learning new subword embeddings. Our model source and target languages are closely related West Germanic languages (Eberhard et al., 2020). In Table 1, we show parallel sentences in Gronings, West Frisian, Dutch, German, and English to illustrate the lexical similarity between these languages. Additionally, the examples show that there are some lexical and syntactic differences.

We also evaluate to what extent the similarity between each source language of the monolingual models and the target languages is relevant for transferring monolingual representations, and assess the minimum amount of data necessary to adapt these models.

Gronings	Tom is n jong en Mary is n wicht.
West Frisian	Tom is in jonge en Mary is in famke.
Dutch	Tom is een jongen en Mary is een meisje.
German	Tom ist ein Junge und Mary ist ein Mädchen.
English	Tom is a boy and Mary is a girl.
Gronings	Zie haar n bloum ien heur haand.
West Frisian	Se hie in blom yn har hân.
Dutch	Ze had een bloem in haar hand.
German	Sie hatte eine Blume in der Hand.
English	She had a flower in her hand.
Gronings	Dat was n poar joar leden.
West Frisian	Dat wie in pear jier lyn.
Dutch	Dat was een paar jaar geleden.
German	Das war vor ein paar Jahren.
English	That was a couple of years ago.

Table 1: Translations of three sentences in Gronings, West Frisian, Dutch, German, and English.

Our pre-trained models for Gronings and West Frisian (which did not yet exist) are released. Additionally, our code is publicly available for bringing language models to other low-resource languages at https://github.com/wietsedv/low-resource-adapt.

2 Materials

Models We use monolingual BERT-based models of the source languages, and multilingual BERT (mBERT; Devlin et al. 2019a). Specifically, we use BERT (Devlin et al., 2019b) for English, German BERT (gBERT; DBMDZ 2019) for German, and BERTje (de Vries et al., 2019) for Dutch. Each model shares the same architecture as the original base-sized (12 layers) BERT model of Devlin et al. (2019b). The lexical layer weights are shared between the first and last layer of the model to transform discrete tokens into distributed vector representations and vice versa.

Each monolingual model has a vocabulary of 30K capitalized tokens, while mBERT has a vocabulary of 120K tokens shared between the 104 languages it is pre-trained on. These languages include English, German, Dutch and West Frisian, but not Gronings. The monolingual BERT models contain 110M parameters, with 24M being part of the lexical embeddings. Due to its larger vocabulary size, mBERT contains 180M parameters, with 92M part of the lexical embeddings.

Labeled data We use POS-annotated treebanks from the Universal Dependencies (UD) project (Zeman et al., 2020), corresponding to the languages of the monolingual BERT models. For English, we use `GUM` (6.0K sentences; 113.4K tokens) and `ParTUT` (2.1K sentences; 49.6K tokens). In addition, `HDT` (189.9K sentences; 3.4M tokens) and `GSD` (15.6K sentences; 287.7K tokens) are used for German. Finally, `Alpino` (13.6K sentences; 208.5K tokens) and `LassySmall` (7.3K sentences; 98.0K tokens) are used for Dutch. All treebanks are based on various text types from a diverse set of sources. The standard data splits for each of the annotated treebanks are used for training, validation and testing.

We evaluate the performance of our language models on POS-annotated data of Gronings and West Frisian. Manually annotated texts from the

Klunderloa[1] project are used for Gronings (3.8K sentences, 49.0K tokens; fiction, poetry, and songs for children). Annotations follow the UD guidelines. West Frisian is under development in the UD project, and we consider all currently available annotations (1.0K sentences, 15.9K tokens; mainly fiction and news). For both treebanks, 25% is used for development, and 75% is used as a test set.

Unlabeled data The new sub-word embeddings are learned from texts written in Gronings and West Frisian. In total, we have 43MB (8.3M tokens) of plain text available for Gronings. These texts are derived from the Bible, fiction and non-fiction texts, poetry, and Low Saxon Wikipedia. The West Frisian data collection consists of 59MB (10.8M tokens) of plain text extracted from fiction and non-fiction texts, and the multilingual OSCAR corpus (Ortiz Suárez et al., 2020).

Language similarity To quantify language similarity, we use the (lexical-phonetic) LDND measure (Wichmann et al., 2010) on the basis of the 40-item word lists from the ASJP database (Wichmann et al., 2010). While a syntax-based measure may be preferred, this is not available for the included language varieties. We use the LDND as a proxy, given that linguistic distance measures between different linguistic levels are correlated (Spruit et al., 2009). Figure 2 visualizes the relative linguistic distances between the five language varieties using multidimensional scaling (MDS; Torgerson, 1952). If cross-lingual transfer benefits from language similarity, we expect Gronings and West Frisian to profit most from a monolingual Dutch model and least from a monolingual English model, with a German model performing in-between.

Figure 2: MDS plot with the relative positions of English, German, Dutch, Gronings, and West Frisian based on the ASJP-based lexical-phonetic distances.

		Src.	Gronings		W. Frisian	
Source		orig.	orig.	gro.	orig.	fri.
EN	BERT	93.8	26.6	61.6	29.1	78.1
	mBERT	93.8	64.1	84.7	87.1	88.7
DE	gBERT	93.3	25.4	85.5	22.7	89.2
	mBERT	93.0	55.9	82.5	86.1	87.7
NL	BERTje	96.4	64.9	**91.7**	48.0	**95.3**
	mBERT	96.6	**72.4**	89.3	**92.2**	94.7

Table 2: Accuracies for the target languages (columns) with the original and retrained lexical layers (subcolumns), which are averaged per source language.

3 Model Training

Our training procedure consists of two separate fine-tuning steps. The Transformer layers in the three monolingual BERT models and mBERT are fine-tuned for the POS-tagging task. Independently, new lexical layers for each BERT model are trained for the two target languages with a masked language modeling pre-training objective. Afterwards, the retrained lexical layer and the fine-tuned Transformer layers are combined to yield a POS-tagging model that is now adapted to the target language. Optimal checkpoint combinations of retrained lexical layers and fine-tuned Transformer layers are based on their performance on the development data for each target language.

POS-tagging The BERT-based models are fine-tuned for POS-tagging with the UD datasets. The task-specific model consists of BERT's layers with an additional linear classification layer that yields predictions for each of the 16 possible POS tags. During training, the lexical layer of BERT is frozen such that the fine-tuned Transformer layers rely on unchanged token representations from pre-training.

The described model is trained with the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\mathrm{e}{-8}$ and a linearly decreasing learning rate starting at $lr = 1\mathrm{e}{-5}$. Each model is trained until validation loss stops decreasing.

Lexical layer retraining We retrain lexical layers for each BERT model using Gronings and West Frisian data. First, sub-word vocabularies of 10K tokens are created for Gronings and West Frisian using the WordPiece method (Devlin et al., 2019b) where each token occurs at least 100 times in the data. This vocabulary size is chosen conservatively,

---

[1] `http://www.klunderloa.nl`

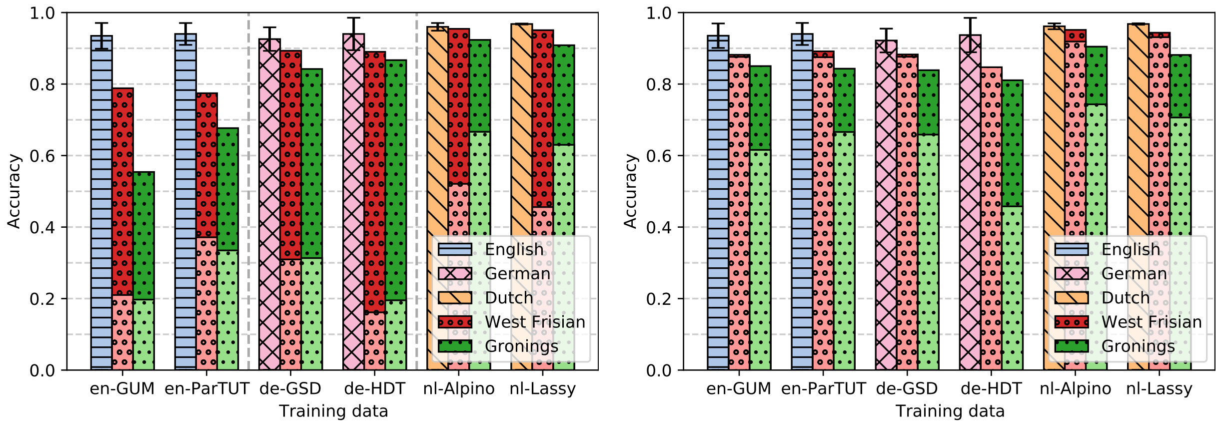

(a) Monolingual model accuracy (BERT, gBERT and BERTje).

(b) Multilingual model accuracy (mBERT).

Figure 3: POS-tagging accuracy for source languages (English, German and Dutch) and target languages (West Frisian and Gronings). Light colors correspond to the accuracy with the original lexical layer and dark colors show improvements with retrained lexical layers. Source language accuracy was averaged across the two source language test sets. Error bars show the upper and lower test set performance for the source language.

		Gronings						West Frisian					
		1MB	5MB	10MB	20MB	40MB	43MB	1MB	5MB	10MB	20MB	40MB	59MB
EN	BERT	32.2	50.5	68.2	69.4	63.3	61.6	51.8	70.6	76.7	78.8	79.1	78.1
	mBERT	25.3	75.4	84.1	84.3	84.4	84.7	72.5	88.0	88.6	89.1	89.2	88.7
DE	gBERT	39.8	83.5	85.5	85.8	85.4	85.5	**76.0**	87.3	87.7	88.0	88.4	89.2
	mBERT	14.1	59.6	79.7	78.0	81.9	82.5	54.9	80.9	84.3	84.5	85.8	85.7
NL	BERTje	**70.2**	**89.5**	**91.4**	**91.4**	**91.4**	**91.7**	44.7	**94.6**	**95.0**	**95.2**	**95.1**	**95.3**
	mBERT	23.8	70.0	87.6	87.6	88.5	89.3	72.2	92.7	93.9	94.4	94.5	94.8

Table 3: POS-tagging accuracy for Gronings and West Frisian with subsets of the unlabeled lexical layer retraining data. Results are averaged per source language for each of the two source language datasets.

as we have limited data to train the lexical layer. Preliminary experiments with 30K tokens showed poor performance on the development data.

The Gronings and West Frisian unlabeled documents are split into sequences of 128 tokens. Then, the models are trained with a masked language modeling objective where 15% of the input tokens are masked. The Adam optimizer is used with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\text{e}{-8}$ and a linearly decreasing learning rate starting at $lr = 1\text{e}{-4}$. After retraining, we have three (original, Gronings and West Frisian) interchangeable lexical layers for each base model.

4 Results and Discussion

We summarize our results in Table 2 (details per dataset in Appendix A). The monolingual language models perform poorly on Gronings and West Frisian POS-tagging when the original lexical layers are used, even though Gronings is quite similar to Dutch (see Figure 2). mBERT with its original lexical layer achieves better results than the monolingual models, but only West Frisian performance is comparable to the source language performance. Since West Frisian was included in mBERT pre-training, these results suggest that mBERT might serve languages included in pre-training well, whereas it may be less suitable for those not included (e.g., Gronings).

For all monolingual models, task performance greatly improves by retraining the lexical layer for Gronings and West Frisian (Figure 3a). Best results are obtained by (Dutch) BERTje fine-tuned on the `Alpino` dataset (92.4% for Gronings, 95.4% for West Frisian). In contrast, (English) BERT yields the worst performance. We find that performance scores and the linguistic distance from Gronings and West Frisian to the source languages (Figure 2) strongly correlate ($r = -0.85$, $p < 0.05$). This suggests that measures of linguistic distance can guide the optimal choice of monolingual models to transfer to low-resource languages. Retraining mBERT's lexical layer also improves performance, especially for Gronings (Figure 3b), but with smaller gains than for monolingual models.

To estimate how our zero-shot approach compares with supervised learning, we train UDPipe (Straka et al., 2016) with five-fold cross-validation on the Gronings and West Frisian POS-tagging data. UDPipe achieves an accuracy of 91.85 ($\sigma = 0.81$) for Gronings and 90.60 ($\sigma = 0.58$) for West Frisian. These results do not indicate out-of-domain performance, since training and test data are from the same source. Also, labeled data for Gronings comes from a corpus with a specific target audience (i.e. children). Therefore, these results can be seen as an upper-bound. Our adapted models perform on par (Gronings) or better (West Frisian) with no need for labeled data in the target language.

Data size Our zero-shot transfer method relies on the availability of unlabeled Gronings and West Frisian data. Other low-resource languages may have even smaller amounts of data available than we have for West Frisian (59MB) and Gronings (43MB). We therefore assess how little data is sufficient for adequate performance by retraining the lexical layer with subsets of (independently randomly sampled) unlabeled data.

Table 3 shows POS-tagging accuracies for each subset. Results are consistent across both target languages and show that ca. 10MB of data (1.9M tokens) is sufficient to achieve almost optimal performance for the monolingual models. By contrast, mBERT shows a steadier improvement with more data, suggesting that it might further improve if even more data is available than we have for Gronings and West Frisian. BERT's POS-tagging accuracy is very low compared to the other monolingual models and performance decreases with more data. These fluctuations suggest that the retrained lexical layer fits BERT poorly and it is unclear if using more data will impact performance positively.

5 Conclusion

We adapted three monolingual BERT models and mBERT to two low-resource languages, Gronings and West Frisian, by retraining the lexical layers with new vocabularies. We found that the adaptability of mBERT is limited, suggesting that a model trained on a large amount of languages may not facilitate transfer to low-resource languages. Instead, monolingual BERT models are transferable to languages with very little data if the source and target languages are relatively similar. In such case, 10MB of unlabeled data, and no task-specific labeled data, is sufficient to achieve high ($> 90\%$ accuracy) downstream task performance.

Acknowledgments

We gratefully acknowledge the support of the Dutch Research Council (NWO Aspasia grant for M. Nissim) and the financial support of the Center for Groningen Language and Culture (CGTC). Finally, we thank the anonymous reviewers for their insightful feedback. Any mistakes remain our own.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

DBMDZ. 2019. German BERT. `https://github.com/dbmdz/berts#german-bert`. [Online].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT-base Multilingual cased. `https://github.com/google-research/bert/blob/master/multilingual.md`. [Online].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2020. Ethnologue: Languages of the World. Twenty-third edition. SIL International.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific BERT models. *arXiv preprint arXiv:2003.02912*.

Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Marco René Spruit, Wilbert Heeringa, and John Nerbonne. 2009. Associations among linguistic levels. *Lingua*, 119(11):1624 – 1642. The Forests behind the Trees.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).

Warren S Torgerson. 1952. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. BERTje: A Dutch BERT Model. arXiv:1912.09582.

Wietse de Vries and Malvina Nissim. 2020. As Good as New. How to Successfully Recycle English GPT-2 to Make Models for Other Languages. *arXiv preprint arXiv:2012.05628*.

Søren Wichmann, Eric W. Holman, Dik Bakker, and Cecil H. Brown. 2010. Evaluating linguistic distance measures. *Physica A: Statistical Mechanics and its Applications*, 389(17):3632 – 3639.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, et al. 2020. Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A Detailed Results

Table 4 shows results per adapted model per training dataset. Dutch POS-tagging accuracy is still relatively high after lexical layer replacement. Similarly, Table 5 shows the POS-tagging performance with subsets of the lexical layer retraining data per training dataset. Training on the Dutch `Alpino` dataset instead of `LassySmall` results in consistently higher performance for both Gronings and West Frisian.

		Test language:	**Source**			**Gronings**		**West Frisian**	
Train language:			orig.	gro.	fri.	orig.	gro.	orig.	fri.
EN	GUM	BERT	93.5	13.5	23.5	19.7	55.4	21.0	78.8
		mBERT	93.5	22.0	22.2	61.6	85.0	87.5	88.2
	ParTUT	BERT	**94.0**	16.6	26.4	33.5	67.7	37.1	77.4
		mBERT	**94.0**	41.3	47.6	66.6	84.3	86.7	89.2
DE	GSD	gBERT	92.6	23.3	22.4	31.3	84.2	28.4	89.3
		mBERT	92.2	25.1	22.2	65.9	83.9	87.5	88.3
	HDT	gBERT	**94.0**	28.5	26.2	19.5	86.7	16.9	89.0
		mBERT	93.7	26.1	22.1	45.8	81.1	84.7	83.0
NL	Alpino	BERTje	96.0	90.8	78.1	66.7	**92.4**	50.0	**95.4**
		mBERT	96.2	87.8	82.8	**74.3**	90.5	91.9	95.1
	LassySmall	BERTje	**96.8**	89.6	70.3	63.0	90.9	45.9	95.1
		mBERT	**96.8**	80.4	51.3	70.6	88.1	**92.7**	94.4

Table 4: Accuracy per target language variety (columns) per lexical layer (sub-columns). This is an extended version of Table 1 in the main paper with accuracies separated by POS-tagging training dataset. This table shows that not all datasets are equally effective for transfer to Gronings and West Frisian.

			Gronings						**West Frisian**					
			1MB	5MB	10MB	20MB	40MB	43MB	1MB	5MB	10MB	20MB	40MB	59MB
EN	BERT	GUM	29.2	47.8	66.1	67.1	58.9	55.4	48.0	69.5	76.6	79.8	79.4	78.5
		ParTUT	37.8	55.1	70.4	72.0	67.8	85.0	53.1	70.4	75.9	78.1	77.8	88.7
	mBERT	GUM	19.6	73.5	84.8	84.9	84.8	67.7	69.7	87.1	88.0	88.4	88.5	77.0
		ParTUT	30.0	76.7	84.0	84.2	84.1	84.3	74.3	88.1	88.4	89.7	89.4	89.3
DE	gBERT	GSD	48.8	82.3	83.9	84.0	83.8	84.2	77.7	87.3	88.8	88.5	88.7	89.1
		HDT	30.9	84.5	86.5	87.0	86.3	83.9	73.8	86.3	86.6	87.6	87.1	88.0
	mBERT	GSD	24.0	74.0	82.4	82.4	82.7	86.7	71.1	87.1	87.3	88.1	88.1	89.3
		HDT	03.7	44.2	75.1	72.2	79.5	81.1	34.4	72.0	79.1	78.7	81.2	83.5
NL	BERTje	Alpino	**73.2**	**90.3**	**92.0**	**91.9**	**92.0**	**92.4**	43.5	**94.2**	94.8	**95.1**	**94.9**	**95.4**
		LassySmall	67.0	88.3	90.0	90.2	89.9	90.5	**44.3**	93.6	**94.9**	94.4	94.6	95.0
	mBERT	Alpino	31.0	79.6	89.1	88.5	89.3	90.9	74.9	93.7	93.8	94.5	94.7	94.9
		LassySmall	15.9	57.4	85.0	85.7	86.7	88.1	67.8	91.6	93.0	93.7	94.1	94.2

Table 5: POS-tagging accuracy for Gronings and West Frisian with subsets of the unlabeled lexical layer retraining data. This is an extended version of Table 2 in the main paper with accuracies separated by POS-tagging training dataset.

BATCHMIXUP: Improving Training by Interpolating Hidden States of the Entire Mini-batch

Wenpeng Yin, Huan Wang, Jin Qu and Caiming Xiong
Salesforce Research
`wyin@salesforce.com`

Abstract

Usually, we train a neural system on a sequence of mini-batches of labeled instances. Each mini-batch is composed of k samples, and each sample will learn a representation vector. MIXUP implicitly generates synthetic samples through linearly interpolating inputs and their corresponding labels of random sample pairs in the same mini-batch. This means that MIXUP only generates new points on the edges connecting every two original points in the representation space. We observed that the new points by the standard MIXUP cover pretty limited regions in the entire space of the mini-batch. In this work, we propose BATCHMIXUP—improving the model learning by interpolating hidden states of the entire mini-batch. BATCHMIXUP can generate new points scattered throughout the space corresponding to the mini-batch. In experiments, BATCHMIXUP shows superior performance than competitive baselines in improving the performance of NLP tasks while using different ratios of training data.

1 Introduction

The study of data augmentation techniques has a long history in the NLP community. Typical data augmentations include synonym replacement (Kobayashi, 2018), back-translation (Fadaee et al., 2017), adding data noise (Xie et al., 2017), etc. Mostly, these techniques are combined with the augmentation-free models in pipeline. MIXUP (Zhang et al., 2018) is able to augment the data by linearly combining each two examples by their hidden representations, keeping the whole system trained in end-to-end.

MIXUP has shown effectiveness in a range of NLP tasks (Sun et al., 2020). Nevertheless, it has two drawbacks. First, MIXUP generates new points merely and exactly on the connecting edges of random point pairs; these new points cover pretty limited region in the representation space of the mini-batch. Second, the training of a system equipped with MIXUP is considerably inefficient—generally, MIXUP slows down the training by n times if it generates n new points for each original point pair. In this work, we propose BATCHMIXUP, an improved mixup paradigm that generates new points scattered uniformly throughout the whole representation region of the mini-batch. Specifically, within a mini-batch, each example and its label will first learn a representation vector respectively, BATCHMIXUP then generates n new points (including a new input representations and a new label representation) simultaneously by non-linearly interpolating all the examples in the same mini-batch. The new n points are expected to better identify the space represented by the mini-batch. Finally, the n mixed points will act as one batch to update the model.

Our model BATCHMIXUP, as a batch-wise non-linear MIXUP, shows advantages in two aspects. (i) Compared with the standard MIXUP, BATCHMIXUP further improves the representation learning in solving downstream NLP tasks, yielding better performance. (ii) BATCHMIXUP works much more efficient than the conventional MIXUP and other pair-wise mixup variants.

2 Related Work

MIXUP was originally proposed in the computer vision community. The standard MIXUP (Zhang et al., 2018) interpolates the raw pixels of each two images in a mini-batch. Verma et al. (2019) conducted interpolation in the hidden states of images. Guo et al. (2019b) discovered a limitation of MIXUP, called "manifold intrusion", which is the conflict between the synthetic label of the mixed-up points and the labels of the original examples. They came up with "AdaMixup", an adaptive MIXUP,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4908–4912
August 1–6, 2021.

where the mixing policies are automatically learned from the data using an additional network and objective function designed to avoid manifold intrusion. Other work tried to explain the work mechanisms of MIXUP from different threads, such as "MIXUP as directional adversarial training" (Archambault et al., 2019), "MIXUP training as the complexity reduction" (Kimura, 2020)

To date, only a couple of previous studies explored the effectiveness of the standard MIXUP in NLP. Guo et al. (2019a) tried two strategies: interpolating word embeddings or sentence embeddings generated by convolutional/recurrent neural networks. Sun et al. (2020) incorporated MIXUP into BERT (Devlin et al., 2019), the state of the art architecture in NLP. To improve the standard MIXUP, Guo (2020) added non-linearity to the MIXUP for text classification tasks. However, that non-linear MIXUP works on word embedding level, which is less applicable to Transformer-style (Vaswani et al., 2017) systems. All the work above are pair-wise mixup, this work is the first work that interpolates all the examples in the same mini-batch to cover the representation space better.

3 The Base Model: MIXUP

Given a pair of samples (x_i, y_i) and (x_j, y_j) from the original mini-batch (x: input, y: the one-hot label), the standard MIXUP (Zhang et al., 2018) generates a synthetic sample as follows.

$$\hat{x}_{ij} = \beta x_i + (1-\beta)x_j \tag{1}$$

$$\hat{y}_{ij} = \beta y_i + (1-\beta)y_j \tag{2}$$

where β is a mixing scalar, sampled from a Beta(α, α) distribution with a hyper-parameter α, for mixing both the inputs and the corresponding targets. The generated synthetic data are then fed into the model for training to minimize the loss function.

From the same mini-batch, the standard MIXUP will sample the β value n times so that totally n new mixed points for a sampled input pair will be generated sequentially. The model, as a result, will be updated n times more than the mixup-free model.

4 Our Model: BATCHMIXUP

BATCHMIXUP mixes all the samples in the same mini-batch on the level of hidden states generated by RoBERTa (Liu et al., 2019).[1]

To start, we first think about how the standard text classifier works: For the labeled input (x_i, y_i), first RoBERTa (optionally with a multilayer perceptron block) generates a representation for x_i ("$v(x_i) \in \mathbb{R}^d$"), then $v(x_i)$ is fed to a logistic regression (LR) layer to classify to y_i. The LR layer has a weight matrix $W \in \mathbb{R}^{c\times d}$ where c is the class size and d is the dimension size of representations. Each row in W, i.e., $w_i \in \mathbb{R}^d$, can be treated as the representation vector of the class y_i. So, LR essentially uses the dot-product to derive the matching score ($s_i \in \mathbb{R}$) between the input x_i and the label y_i: $s_i = (v(x_i))^T \cdot w_i$.

For the same mini-batch of inputs $\{v(x_i)\}$ and labels $\{w_i\}$, BATCHMIXUP deploys the same mixing policy to interpolate the $\{v(x_i)\}$ and $\{w_i\}$.

We denote the whole batch of input representations $\{v(x_i)\}$ as $\mathcal{X} \in \mathbb{R}^{d\times b}$ where b is the batch size and the whole mixing policy for this batch is $\mathcal{M} \in \mathbb{R}^{n\times d\times b}$. To generate a single mixed point $\hat{x}_i \in \mathbb{R}^d$, the BATCHMIXUP uses the following mixing policy $\mathcal{M}[i] \in \mathbb{R}^{d\times b}$ ($i = 1, \cdots, n$) on $\mathcal{X}$, where each element of $\mathcal{M}[i]$ is independently sampled from a Beta(α, α) distribution:

$$\hat{x}_i = \sum_{\text{axis}=1} (\text{softmax}(\mathcal{M}[i]) \circ \mathcal{X}) \tag{3}$$

where $\circ$ is the Hadamard product. Equation 3 can be performed for all i values in [1,n] simultaneously; this means the original batch input $\mathcal{X}$ is transformed into a new batch of mixed input $\hat{\mathcal{X}} \in \mathbb{R}^{n\times d}$.

Similarly, the same mixing policy $\mathcal{M}$ is applied to the batch of label representations, denoted as $\mathcal{Y} \in \mathbb{R}^{d\times b}$ ($\mathcal{Y} = \{w_i\}$):

$$\hat{y}_i = \sum_{\text{axis}=1} (\text{softmax}(\mathcal{M}[i]) \circ \mathcal{Y}) \tag{4}$$

Each ($\hat{x}_i$, $\hat{y}_i$) ($i = 1, \cdots, n$) is a newly mixed point. All $\{(\hat{x}_i, \hat{y}_i)\}$ can be generated in parallel and are scattered throughout the space represented by $\mathcal{X}$.

For training, we minimize the negative-dot-product loss between the mixed input and the mixed label. In testing, an input x_i still compares with all classes $\{y_i\}$ by dot-product between $v(x_i)$ and all $\{w_i\}$ to find the best class.

5 Experiments

In experiments, we check the effectiveness of our approach in NLP tasks with two settings: one is

---

[1] Please note that BATCHMIXUP also works for other deep neural encoders.

#training data		entailment	relation	intent
100%	RoBERTa-large	80.91±1.47	87.24±1.52	93.77±0.44
	w/ Mixup	82.03±1.69	87.91±0.34	93.41±1.24
	w/ Nonlinear Mixup	82.98±1.17	88.41±0.73	94.29±1.01
	w/ BatchMixup	**83.56±0.90**	**90.04±0.33**	**94.94±0.20**
75%	RoBERTa-large	80.07±1.27	87.49±0.31	90.31±1.11
	w/ Mixup	80.53±1.31	87.87±0.12	92.44±1.98
	w/ Nonlinear Mixup	80.90±0.39	87.66±0.81	93.00±1.84
	w/ BatchMixup	**82.00±0.91**	**88.41±0.54**	**93.55±1.58**
50%	RoBERTa-large	77.50±0.82	82.83±4.92	85.11±2.88
	w/ Mixup	75.19±6.07	87.00±0.18	87.09±2.06
	w/ Nonlinear Mixup	77.66±1.90	87.79±0.29	88.13±2.92
	w/ BatchMixup	**79.47±2.07**	**88.29±0.22**	**90.13±2.20**
25%	RoBERTa-large	70.92±4.04	78.95±0.33	80.95±2.22
	w/ Mixup	71.70±5.66	82.02±0.59	84.95±1.19
	w/ Nonlinear Mixup	72.14±4.27	83.71±0.12	85.44±2.02
	w/ BatchMixup	**74.36±2.82**	**86.66±0.26**	**87.71±1.38**
1%	RoBERTa-large	50.41±0.29	3.41±0.74	42.21±5.54
	w/ Mixup	**51.74±0.84**	49.93±2.71	50.21±3.21
	w/ Nonlinear Mixup	51.41±0.43	55.29±4.78	52.21±1.49
	w/ BatchMixup	51.46±1.43	**60.29±2.18**	**55.21±1.77**
random or majority baseline		50.16	1.31	1.29

Table 1: Experimental results on three NLP tasks: textual entailment (RTE (Dagan et al., 2005; Wang et al., 2019)), relation classification (FewRel (Han et al., 2018)) and intent classification (BANKING77 (Casanueva et al., 2020)). We decrease the size of training data from 100% to 1% with random sampling. All numbers are averaged over three random seeds.

full-shot setting that trains on the regular full training data; the other is few-shot setting that train with limited training data. Unfortunately, prior work about mixup never evaluated on few-shot scenarios.

Tasks. We evaluate on the following three tasks.

- **Textual Entailment.** Textual entailment is a task that figures out the truth value of a hypothesis sentence given a premise sentence (Dagan et al., 2005). This is a binary classification ("entailment" or "non-entailment") problem where the input is a sentence pair. We use the GLUE RTE (Wang et al., 2019) benchmark which has 2,490/277/2,999 examples in train/dev/test. The smaller size of this dataset (compared MNLI (Williams et al., 2018) for example) makes it a good testbed for data augmentation techniques.

- **Relation Classification.** FewRel (Han et al., 2018) is a large-scale relation classification dataset. It has 100 relation types, each with 700 labeled examples. The original FewRel relation set was split by 64/16/20 for developing meta-learning techniques which only allow a test instance to search for its relation type within the 20 candidates. This is not a practical setting because (i) in relation detection, an input should search for a label in the entire space of defined relations, (ii) we should always define a "None" type in this problem because most span pairs in the input actually do not have a relation. Since the test relations of FewRel is not publicly available, we use the 64+16=80 relations as the entire relation set, in which 5 relations are treated "None" (So, basically this is a regular "75+None" setting).

- **Intent Classification ("intent").** We use the benchmark BANKING77[2] (Casanueva et al., 2020), which is single-domain intent detection dataset comprising 13,083 annotated examples over 77 intents (average: 170 examples per intent). Each intent class is described by a short name, such as "get physical card", "lost or stolen card", etc.

Baselines. The augmentation-free system we use for above tasks consists of a RoBERTa[3] encoder

---

[2] `https://github.com/PolyAI-LDN/task-specific-datasets`

[3] We used the pretrained "RoBERTa-large"

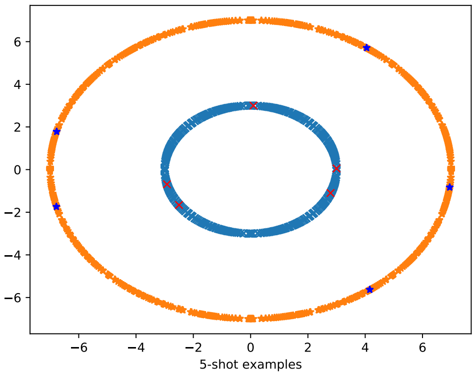

(a) Two classes (orange vs. blue), 5-shot for each

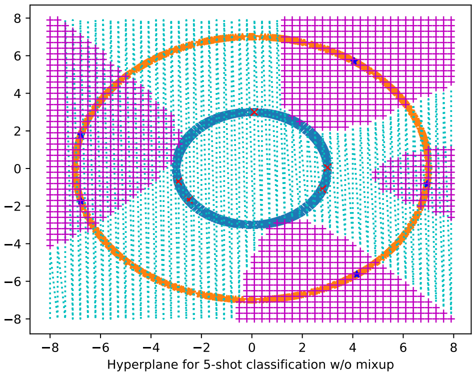

(b) Hyperplane after training w/o mixup; it cannot distinguish the two classes clearly.

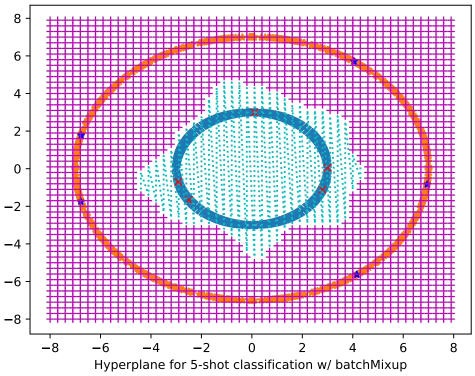

(c) Hyperplane after training w/ BatchMixup, it separates the two classes very well

Figure 1: Visualization analysis

and a final logistic regression layer. Based on this RoBERTa system, we compare our system BatchMixup with (i) the standard Mixup (Zhang et al., 2018; Sun et al., 2020), and (ii) non-linear Mixup (Guo, 2020). Both baselines conduct data interpolation in the hidden states output by the RoBERTa.

All systems are implemented through the Huggingface's Transformers package.[4]

Results and Analysis. Table 1 lists the main results. We notice that our approach RoBERTa+BatchMixup consistently outperforms the baselines Mixup and non-linear Mixup. In "1% entailment", none of systems really worked—all system results are around the majority baseline. This is because that the RTE task is very challenging with over limited annotations. With 2.5K × 1%=25 labeled examples, the "RoBETTa" cannot learn any useful representations. This is in line with the observations in (Yin et al., 2020) which showed that few-shot RTE (when $k \in \{1, 3, 5, 10\}$) will make RoBERTa fail. So, we conclude that when a system is close to random guess, adding mixup is not helpful. In this situation, maybe using other conventional data augmentation skills makes more sense as the representation learning of synthetic data and that of the original data are decoupled.

To further study how BatchMixup works, we simulate the classification process with a toy experiment: we generate a large amount of 2-dimensional data in Gaussian distributions for two classes (Figure 1(a)), and randomly sample 5 examples for each class to conduct 5-shot classification. We used a MLP as the classifier, trained 100 epochs. Comparing the final hyperplane of training with BatchMixup with that of training without Mixup, we can observe that BatchMixup can improve the training considerably.

Last but not least, the training of same epochs for "w/ Mixup", "w/ Nonlinear Mixup" takes much longer than our system "w/ BatchMixup". For example, when all systems separately run on a GPU Tesla V100, our system BatchMixup and the baseline "RoBERTa-large" both take about 1.5min to finish one epoch on RTE, but "w/ Mixup" and "w/ Nonlinear Mixup" will take ∼20mins if β is sampled 15 times per point pair.

6 Conclusion

In this work, we proposed a novel Mixup model, named BatchMixup, to improve the text classifier. Different with prior Mixup variants, which always interpolate random two points, our system interpolates all the hidden states in the mini-batch. The mixed points by our system are able to better cover the space expressed by the minibatch. The experiments and visualization analysis both show the effectiveness of our model BatchMixup.

---

[4] `https://github.com/huggingface/transformers`

References

Guillaume P. Archambault, Yongyi Mao, Hongyu Guo, and Richong Zhang. 2019. Mixup as directional adversarial training. *CoRR*, abs/1906.06875.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *ACL*, pages 567–573.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *AAAI*, pages 4044–4051.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019a. Augmenting data with mixup for sentence classification: An empirical study. *CoRR*, abs/1905.08941.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *AAAI*, pages 3714–3722.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*, pages 4803–4809.

Masanari Kimura. 2020. Mixup training as the complexity reduction. *CoRR*, abs/2006.06231.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT*, pages 452–457.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S. Yu, and Lifang He. 2020. Mixup-transfomer: Dynamic data augmentation for NLP tasks. In *COLING*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, pages 6438–6447.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *ICLR*.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *EMNLP*, pages 8229–8239.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *ICLR*.

DocNLI: A Large-scale Dataset for Document-level Natural Language Inference

Wenpeng Yin^1, **Dragomir Radev**^{1,2} **and Caiming Xiong**^{1}
^{1}Salesforce Research and ^{2}Yale University
`wyin@salesforce.com`

Abstract

Natural language inference (NLI) is formulated as a unified framework for solving various NLP problems such as relation extraction, question answering, summarization, etc. It has been studied intensively in the past few years thanks to the availability of large-scale labeled datasets. However, most existing studies focus on merely sentence-level inference, which limits the scope of NLI's application in downstream NLP problems. This work presents DocNLI — a newly-constructed large-scale dataset for document-level NLI. DocNLI is transformed from a broad range of NLP problems and covers multiple genres of text. The premises always stay in the document granularity, whereas the hypotheses vary in length from single sentences to passages with hundreds of words. Additionally, DocNLI has pretty limited artifacts[1] which unfortunately widely exist in some popular sentence-level NLI datasets. Our experiments demonstrate that, even without fine-tuning, a model pretrained on DocNLI shows promising performance on popular sentence-level benchmarks, and generalizes well to out-of-domain NLP tasks that rely on inference at document granularity. Task-specific fine-tuning can bring further improvements. Data, code and pretrained models can be found at `https://github.com/salesforce/DocNLI`.

1 Introduction

A fundamental challenge of natural language processing (NLP) lies in the variability of semantic expression, where the same meaning can be conveyed by, or inferred from, different pieces of text (Dagan et al., 2009). This phenomenon gives rise to the many-to-many mapping between textual expressions and meanings. Many NLP problems, such as information extraction, question answering, document summarization and machine translation, desire a system for this variability phenomenon so as to figure out that a particular meaning can be inferred from distinct text strings (Dagan et al., 2009). Natural language inference (a.k.a textual entailment (Dagan et al., 2005)) acts as a unified framework to study those NLP problems by casting the background text as a premise and the text of target meaning as a hypothesis. Then, a good NLI recognizer can be considerably translated to a well-performing system regarding respective NLP tasks.

NLI was first studied in (Dagan et al., 2005). Research in the early stages was mostly driven by the PASCAL Recognizing Textual Entailment (RTE) challenges which are annual competitions with benchmark datasets released. In the past few years, the study of NLI has moved forward rapidly along with the construction of large-scale datasets, such as SNLI (Bowman et al., 2015), the science domain SciTail (Khot et al., 2018) and multi-genre MNLI (Williams et al., 2018), etc.

However, some NLI datasets may not be suitable any more for solving downstream NLP problems since they were commonly crowdsourced in isolation from any end task [2] (Khot et al., 2018). In addition, most NLI datasets and studies paid attention merely to sentence-level inference — both the premises and hypotheses are single (and usually short) sentences. This makes them unsuitable for other open-ended NLP problems. For example, to verify the factual correctness of a document summary, sentence-level NLI systems cannot be of much help (Kryściński et al., 2019). Considering the fact-checking task FEVER (Thorne et al., 2018) as another example, in order to figure out the truth value of a claim against a Wikipedia article, NLI

---

[1] NLI "artifacts" are some label-specific biases (often) in the hypotheses; they can indicate which NLI class a hypothesis belongs to even without looking at the premise.

[2] Except for RTE and SciTail

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4913–4922
August 1–6, 2021.

has to be done on individual sentences instead of using the whole article as the premise. In short, some NLP tasks require the reasoning of NLI to go beyond the sentence granularity, regarding both the premise and the hypothesis.

In this work, we introduce DocNLI, a large-scale dataset for document-level NLI. It is constructed by reformatting some mainstream NLP tasks, including question answering and document summarization, and integrating existing NLI in which the premises may be longer than single sentences. DocNLI has the following characteristics:

- DocNLI is highly related with end NLP tasks. A well-performing system to DocNLI is expected to throw light on addressing other NLP challenges.
- Premises always have more than one sentence; the majority are natural documents such as news articles. Hypotheses cover a variety of lengths, ranging from a single sentence to a document with hundreds of words. By this setting, we hope the systems can learn to deal with future applications that need to infer the truth value of a piece of text regardless of its length.
- In contrast to some existing sentence-level NLI datasets, DocNLI has pretty limited artifacts. We present a novel approach to disconnect the potential artifacts with the NLI task itself; a "hypothesis-only" baseline has difficulties in discovering some spurious correlations.

In experiments, we will show that a RoBERTa (Liu et al., 2019) system pre-trained on DocNLI demonstrates promising performance on conventional sentence-level NLI benchmarks such as MNLI and SciTail, and generalizes well to out-of-domain NLP tasks (e.g., fact-checking and multi-choice question answering) that necessitate document-level inference. Task-specific fine-tuning can further improve the performance and achieve new state of the art for some end tasks.

2 Related Work

To our knowledge, document-level NLI has attracted very little ink in the community, possibly because of the lack of labeled datasets. In this section, we mainly describe some prior NLI datasets that share some spirits with our DocNLI.

End-task driven. As mentioned in Section 1, the RTE series were driven by downstream NLP tasks such as information retrieval, information extraction, question answering, and summarization. MCTest (Richardson et al., 2013) is a question answering task in which a paragraph is given as background knowledge, then each question is paired with a positive answer and some negative answers. The MCTest benchmark released an NLI version of this corpus by treating the whole paragraph as a premise and combining the question and answer candidates as hypotheses. SciTail (Khot et al., 2018) is also derived from the end QA task of answering multiple-choice school-level science questions. Unlike MCTest, the premises in SciTail are single sentences selected by an information retrieval approach. By casting an end NLP task as NLI, a good NLI recognizer therefore can be directly turned into a well-performing system for that NLP task. This can be even more attractive if we can learn a generalizable NLI system to solve some NLP problems that have limited annotations.

Going beyond the sentence granularity. The premises in MCTest are paragraphs, but MCTest has pretty limited size. Demszky et al. (2018) tried to convert the question answering benchmark SQuAD (Rajpurkar et al., 2016) into an NLI format by treating the paragraph as a premise and using a neural network to generate a hypothesis sentence given the question and answer span as inputs. Kryściński et al. (2019) created a (document, sentence) pair data "FactCC" to train a classifier for checking the factual correctness of single sentences in automatically generated summaries. FactCC is specific to the target summarization benchmark dataset, so it is unclear how well FactCC can generalize to other summarization benchmarks and other NLP problems. In addition, only single sentences act as hypotheses. Nevertheless, that literature exactly showed that document-level NLI, especially the inference of document-level hypotheses, is highly desirable. ANLI (Nie et al., 2020) also gather multi-sentence as premises. However, the sentence sizes in ANLI premises are pretty limited and the hypotheses in ANLI are single sentences consistently.

To our knowledge, our DocNLI is the first dataset that uses hypotheses longer than single sentences, and stays closely with end NLP tasks.

3 Data Creation

What kind of document-level NLI dataset is preferred? (i) We want the premise is a paragraph or

	original task	domain	premise length	hypothesis length
ANLI	NLI	various (wiki, news, etc.)	multi-sentence (20∼94 words)	single sentence (4∼18 words)
SQuAD	QA	wiki	paragraph (27∼237 words)	single sentence (6∼22 words)
DUC (2001)	summarization	news	doc. (124∼879 words)	multi-sent (80 ∼100 words)
CNN/Daily Mail	summarization	news	doc. (247∼652 words)	3∼4 sent. (40∼50 words)
Curation	summarization	news	doc. (229∼842 words)	multi-sent (64∼279 words)

Table 1: Data resources that are reformatted into DOCNLI.

even a document, and the hypotheses cover a large range of granularity: from a single sentence to a longer paragraph (e.g., 250 words); (ii) Diverse domains; (iii) No severe artifacts; for example, we do not include the hypotheses that can be easily found "grammatically incorrect" by well-trained language models such as BERT (Devlin et al., 2019).

3.1 Data Preprocessing

Table 1 lists all the resources that we use to create DOCNLI. Briefly, DOCNLI combines and reformats five existing NLP benchmarks: adversarial NLI (ANLI) (Nie et al., 2020), the question answering benchmark SQuAD (Rajpurkar et al., 2016) and three summarization benchmarks (DUC2001[3], CNN/DailyMail (Nallapati et al., 2016), and Curation[4] (Curation, 2020)). Next, we describe how each data resource is integrated into DOCNLI.

ANLI to DOCNLI. ANLI is a large-scale NLI dataset collected via an iterative, adversarial human-and-model-in-the-loop procedure. In each round, the best-performing model from the previous round is selected and then human annotators are asked to write "hard" examples that this model misclassifies. They always choose *multi-sentence paragraphs as premises and write single sentences as hypotheses*. Then a part of those "hard" examples join the training set so as to learn a stronger model for the next round. The remaining "hard" examples act as dev/test sets correspondingly. Totally three rounds were accomplished for ANLI construction. In the end, ANLI has train/dev/test sizes as 162,865/3200/3200 with three classes "entail", "neutral" and "contradict".

We keep premise-hypothesis pairs in ANLI unchanged, but unify the two classes "neutral" and "contradict" into a new class "not_entail".

SQuAD to DOCNLI. SQuAD is a QA dataset in which a multi-sentence paragraph is accompanied by a couple of questions; each question has a text span from the paragraph as its answer. Demszky et al. (2018) converted SQuAD into NLI format by reformatting the question-answer pairs into declarative sentences (QA2D) by neural networks. The resulting sentences containing correct (resp. incorrect) answers are entailed (resp. not_entail) by the paragraph. Human evaluation was conducted to make sure those declarative sentences have high quality on three criteria: grammaticality, naturalness, and completeness. In addition, Demszky et al. (2018) replicated some statistical analyses showing that this QA2D dataset does not have clear artifacts as SNLI or MNLI. In this work, we directly use this QA2D dataset and re-split it into train/dev/test by 50k/7,236/8,275.

Summarization to DOCNLI. Here we introduce the basics of the three summarization datasets (DUC2001, CNN/DailyMail and Curation), and explain how we convert them into DOCNLI in a unified approach.

- The DUC series are some of the earliest benchmarks for studying automatic document summarization. DUC2001 is on generic, single-document summarization in the news domain. There are totally 600 documents along with human-written reference summaries of approximately 100 words. We split those document-summary pairs into train/dev/test by size of 400/50/150.

---

[3] `https://www-nlpir.nist.gov/projects/duc/guidelines/2001.html`

[4] `https://github.com/CurationCorp/curation-corpus`

doc		Petrofac shares surged on Wednesday following reports that the Serious Fraud Office has abandoned a criminal investigation into three businessmen who were accused of paying brides in the energy industry. The SFO had been probing claims that Unaoil - a Monaco-based consultancy that worked with Petrofac, primarily in Kazakhstan between 2002 and 2009 - had paid multimillion pound brides to land contracts in the oil and gas industry. But The Guardian cited sources earlier as saying that the SFO has dropped the investigation into the trio. Compliance industry newsletter MLex was the first to report the news, saying on Tuesday that the probe had been halted after three years. The SFO launched an investigation into Petrofac in May 2017 as part of a wider probe into Unaoil. In February 2019, David Lufkin, Petrofac's former global head of sales, pleaded guilty to 11 counts of bribery linked to contracts worth more than $730m in Iraq and $3.5bn in Saudi Arabia. SFO spokesman Adam Lilley said the Unaoil investigation ”remains active and is ongoing”. “We do not comment on ongoing investigations,” he said. [$\cdots$]
real summ.		The Serious Fraud Office has reportedly dropped a criminal investigation into three businessmen who had been accused of conspiring to make corrupt payments to secure contracts in Iraq. The SFO launched an investigation into Petrofac in May 2017 as part of a wider probe into Monaco-based oil consultancy Unaoil.
fake summaries	word repl.	The Serious financial Office has reportedly launched a criminal investigation into three businessmen who had been accused of conspiring to make corrupt payments to oil contracts in Iraq. The SFO launched an investigation into corruption in May 2017 as part of a wider investigation into Monaco-based financial consultancy firms.
	entity repl.	Unaoil has reportedly dropped a criminal investigation into three businessmen who had been accused of conspiring to make corrupt payments to secure contracts in Monaco . The SFO launched an investigation into Monaco in May 2017 as part of a wider probe into Petrofac-based oil consultancy The Serious Fraud Office.
	sent repl.	The Serious Fraud Office has reportedly dropped a criminal investigation into three businessmen who had been accused of conspiring to make corrupt payments to secure contracts in Iraq. A spokesman for the SFO said it was “unable to confirm or deny” that an inquiry had taken place.

Table 2: An example of the Curation summarization dataset shows the original document, and the real summary written by humans. We used "word replacement", "entity replacement" and "sentence replacement" to form three types of "fake" summaries against the document. Texts in red are substitutes.

- CNN/DailyMail was gathered from news articles in *CNN* and *Daily Mail* websites; each article is paired with 3 to 4 sentences of abstractive summary bullets generated by humans. CNN/DailyMail has 286,817/13,368/11,487 article-summary pairs in train/dev/test. The source articles in the training set have 766 words spanning 29.74 sentences on average while the summaries consist of averagely 3.72 sentences.
- Curation is a recent summarization dataset with 40,000 professionally-written summaries of news articles. We split it into train/dev/test as 20K/7K/13K.

All three summarization datasets align the documents with the human-written reference summaries. This enables us to obtain "entail" pairs of (document, reference summary). The remaining challenge lies in how to generate "not_entail" pairs.

We adopt three types of manipulations on the "reference" (also referred as "real") summaries.

- *Word replacement*. We mask eight words whose part-of-speech tags are among {"VERB", "NOUN", "PROPN", "NUM"} by spaCy toolkit[5], then use BERT to predict them. The most likely predicted word is used to replace a masked one. After word replacements, the resulting text is our "fake" summary.
- *Entity replacement*. We use spaCy for named entity recognition (NER). For an entity which is the only one of a specific NER type in the real summary, we search for a different entity with the same type from the document to replace it; otherwise, it will be replaced by the entity of the same type in the real summary. We do this operation for five entities. We skip entity-level manipulation for the instances that have fewer than five detected entities. After entity replacement, we get a "fake" summary.
- *Sentence replacement*. From the real summary, we randomly select a sentence, then forward its left context to CTRL (Keskar et al., 2019), a state-of-the-art controllable text generator, to generate a new sentence which is used to replace the selected sentence. This operation generates a new "fake" summary.

Table 2 illustrates a (document, real summary) pair in the Curation dataset, and the three types of "fake" summaries we generated.

3.2 Mitigating Artifacts in DOCNLI

In Section 3.1, we transformed these NLI, QA and summarization datasets to satisfy the format of DOCNLI. We refer this resulting dataset as *raw*-DOCNLI. In consideration of the common artifacts in some popular sentence-level NLI benchmarks (Gururangan et al., 2018; Poliak et al., 2018; Tsuchiya, 2018), we tried a "hypothesis-only" baseline based on RoBERTa on this *raw*-DOCNLI. Sur-

---

[5] `https://spacy.io`

		entail	not_entail
DocNLI	raw DocNLI	ANLI SQuAD $\{(D, R)\}$	ANLI SQuAD $\{(D, F_i)\}$
	added pairs	$\{(F_i^+, F_i)\}$	$\{(F_i, R)\}$

Table 3: D: a document in summarization benchmarks; R: a real summary; F_i: a fake summary derived from R (i=$1 \cdots n$); F_i^+: Using CTRL to insert a generated sentence between a random pair of consecutive sentences in the F_i, in a way similar to what we described as "sentence replacement" in Section 3.1. DocNLI's training set is the combination of raw-DocNLI and those added pairs; DocNLI's dev and test sets *do not have trivial pairs* $\{(F_i^+, F_i)\}$.

prisingly, this baseline indeed obtains non-trivial performance. This means that RoBERTa can still learn some label-specific biases from the hypotheses, even though we have tried hard to make the "fake" summaries coherent and natural.

Nevertheless, this does not mean we have failed to build a robust DocNLI dataset. The surprising behavior of "hypothesis-only" in *raw*-DocNLI indicates that the BERT classifier can easily recognize the summary is "real vs. fake", but "real vs. fake" is not the same concept as "entail vs. not_entail" defined in the NLI framework. This is because a "fake" one can still be "entail"-ed if the premise has proper information; and a "real" one can also be "not_entail" if the premise does not contain necessary clues for inferring it.

For convenience, we use D as a document, R as the real summary, and $\{F_1, F_2, \cdots, F_n\}$ as the n fake summaries derived from R. To ensure the model can learn exactly what "entail vs. not_entail" is rather than be misled by the manipulations that yield those "fake" text pieces, as Table 3 demonstrates, we prepare the following pairs to extend the *raw*-DocNLI and get our final DocNLI:

- Adding pairs (F_i^+, F_i), $i = 1, \cdots, n$, for class "entail". Here F_i^+ has one more sentence than F_i, inserted by CTRL, as described in "sentence replacement" in Section 3.1 (here we do insertion rather than replacement). The goal is to let the system know that a fake summary can also be a positive hypothesis in NLI, if its premise covers necessary information.
- Adding a single pair (F_i, R) for class "not_entail". This means the original real summary can also be a negative hypothesis if it includes mis-matching information with its premise. F_i is randomly chosen from the set $\{F_1, \cdots, F_n\}$.

	train	dev	test
entail.	466,653	28,890	33,128
not_entail	475,661	205,368	233,958
sum	942,314	234,258	267,086

Table 4: Data sizes of DocNLI.

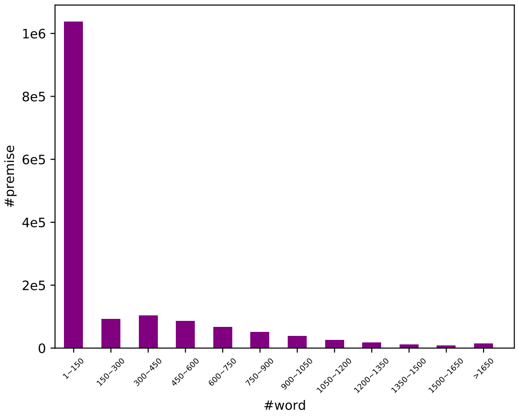

Figure 1: #premise vs. #words in DocNLI

By adding above two sorts of pairs, we want to disconnect the concept of "real vs. fake" from "entail vs. not_entail", letting the system learn the essence of NLI. Both the "real" and "fake" summaries have the same number of instances of being "entail" and "not_entail" in the extended dataset.

It is worth mentioning that since the instances "$(F_i^+, F_i) \rightarrow$ entail" are very trivial to be recognized, we add them in the training set only.

Table 4 lists the sizes of DocNLI for train/dev/test in each class. The training set is roughly balanced, while approximately 12% examples in dev and test belong to "entail". F1 is the evaluation metric.

Figures 1-2 illustrate the length distributions of premises and hypotheses in DocNLI. Because the majority of hypotheses have fewer than 150 words, and real/fake summaries also act as premises in DocNLI, as reported in Table 3, therefore, the majority of premises stay within the length limit of 150 words, shown in Figure 1. Still, there are a large amount of premises whose lengths are within the range of [150, 900] words.

3.3 Human Verification

DocNLI covers examples derived from ANLI, SQuAD and three summarization datasets. Here,

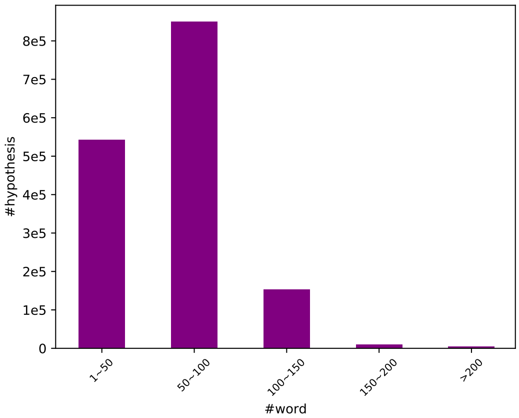

Figure 2: #hypothesis vs. #words in DOCNLI. The hypotheses in our new data DOCNLI are mostly longer than single sentences; this is one key difference with some related datasets.

we only conduct human verification for the pairs derived from summarization, especially for those "fake" summaries, to get some clues to answer two questions: (i) Are those "fake" summaries indeed incorrect given the original document? (ii) Do those "fake" summaries look natural? By "natural" we mean the text should have no major grammar errors, and no unrelated text spans that make the whole text piece look over uncoordinated.

The authors of this work manually checked 200 random "fake" examples, among which none is true given the same document as the "real" summary. This is mainly because we replaced relatively a lot from the original real summaries.

However, some minor grammar issues inevitably exist. Take the following text piece as an example:

"*WeWork Companies LLC (replace: "WeWork") has announced plans to hold a conference call on 2025 for holders of its 7.875% Senior Notes due 26 August to discuss its Notes (replace: "Q2") results. Securities analysts and market-making financial institutions can also register for access. The call is scheduled for 12:00 P.M. (replace: "noon Eastern Time").*"

This example has five entities that are substitutes, all underlined. If a substitute comes from the premise document, we use "(replace: XX)" to denote the entity that was there. The two entities (NER type "date"), in red, replaced each other: "2025" and "26 August", which makes the new text "[· · ·] on 2025 [· · ·]" grammatically incorrect.

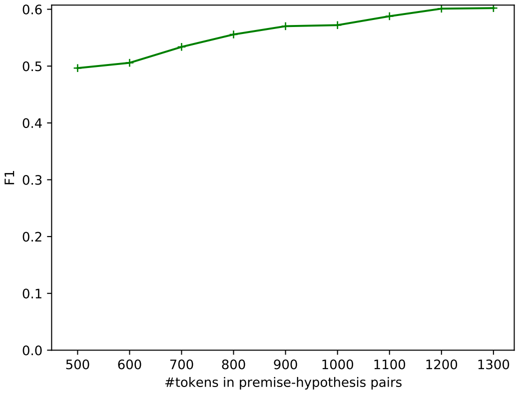

Figure 3: Longformer F1 vs. #tokens in DOCNLI dev set.

	dev	test
Random	19.75	19.91
Hypothesis-only	21.89	22.02
Longformer-base	56.11	54.37
Roberta-large	64.06	61.52

Table 5: F1 scores on DOCNLI.

4 Experiments

We study three questions. (**Q**$_1$) How challenging is DOCNLI (especially with regard to different lengths of hypotheses)? (**Q**$_2$) Out-of-domain evaluation, in which we train a system given DOCNLI and test it on downstream NLP tasks that are not covered by the source tasks in DOCNLI construction. (**Q**$_3$) Could a system trained on DOCNLI work well on sentence-level NLI?

4.1 The DOCNLI task is challenging

The state-of-the-art systems on sentence-level NLI problems are largely based on transformers (Vaswani et al., 2017), such as BERT, RoBERTa (Liu et al., 2019), etc. However, they can only handle maximal 512 tokens preprocessed by the WordPiece tokenizer (Wu et al., 2016). This is an issue to build an effective document-level NLI machine. Therefore, for the main experiments, we also report Longformer (Beltagy et al., 2020) – a RoBERTa variant that can handle up to 4096 tokens. Longformer has two versions, one is "Longformer-base", the other is "Longformer-large". We currently only report "Longformer-base" due to memory constraints.

To answer the question (**Q**$_1$), we compare the following systems (we can include more baselines, but most popular approaches either are too weak or

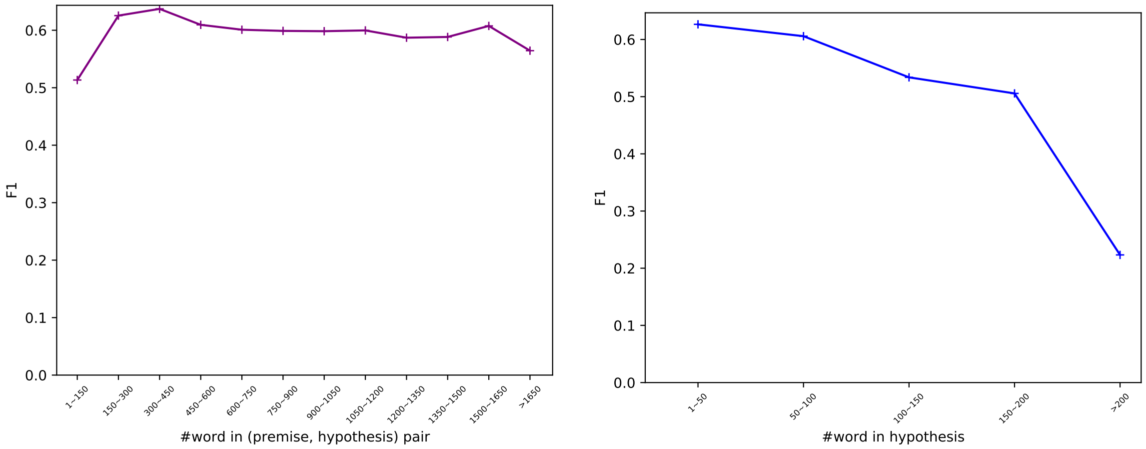

(a) F1 vs. pair length

(b) F1 vs. hypothesis length

Figure 4: Fine-grained F1 scores for different lengths of DocNLI pairs or hypotheses alone.

can only handle short piece of texts):

- **Hypothesis-only.** We train RoBERTa on hypotheses only.
- **RoBERTa-large.** Although we claimed that RoBERTa may not be a good platform to learn DocNLI, here we report it just for reference. Maximal token limit: 512 tokens.
- **Longformer-base.** We use the released Longformer library[6] by (Beltagy et al., 2020), training it on the full training set of DocNLI, with length limit of 1.3K tokens, batch size 1 per GPU, and learning rate 5e-6.

All systems are trained for 5 epochs, and report the best model tuned on dev set. Table 5 lists the F1 results of all systems on DocNLI. We notice that "hypothesis-only" is just slightly higher than random guess, and is much lower than the "RoBERTa-large" system which takes both premises and hypotheses as input: 22.02 vs. 61.52 on test. Surprisingly, "Longformer"'s performance is clearly below that of the RoBERTa, even if it covers more tokens, possibly because we do not have enough computing resources to fully explore the better settings of Longformer. Figure 3 illustrates the impact of taking different numbers of tokens in Longformer, evaluated on dev set. In general, the more tokens the better performance.

We further look at the fine-grained F1 reports on the various lengths of premise-hypothesis pairs and hypotheses alone. Figure 4(a) shows that the system performance for pairs of lengths > 450 does not change clearly. This is probably due to those models' truncation when the (premise, hypothesis) pairs are overlong (note that one word may be split into multiple tokens by the WordPiece tokenizer). Figure 4(b) demonstrates that the task gets increasingly challenging when the hypotheses become longer, which matches our intuition.

Overall, DocNLI is a very challenging task that seeks solutions equipped with a stronger capability of representation learning.

4.2 Applying DocNLI to end NLP tasks

To answer the question (**Q**$_2$), we play DocNLI to see if it can help downstream NLP tasks. As DocNLI is derived from summarization and QA already, we do not consider these two types of NLP tasks any more (since improvements on them are not surprising), especially when their domains are covered in DocNLI. In addition, we have to explore tasks that have NLI-format data available — converting an open NLP task to NLI format is not trivial and is beyond the scope of this work. Therefore, we consider the following two NLP tasks:

FEVER (Thorne et al., 2018). FEVER is a benchmark dataset for fact-checking. Given an declarative sentence (aka. "claim"), the task searches for textual evidences from Wikipedia articles and then decide the truth value of this sentence (i.e., support / refute / not-enough-info).

We use the NLI-version of FEVER, released by (Nie et al., 2019): claims are hypotheses; premises corresponding to "support" or "refute' claims consist of ground truth textual evidence and some other randomly sampled evidence; premises for "not-enough-info" claims are the concatenation of all selected evidential sentences by a previ-

---

[6] `https://github.com/allenai/longformer`

		FEVER binary	MCTest v160	MCTest v500
	random	50.00	25.00	25.00
pretrain	MNLI	86.64	75.41	70.66
	ANLI	87.51	82.50	78.66
	DocNLI	88.84	90.00	85.83
	+finetune	**89.44**	**90.83**	**90.66**
Prior state-of-the-art		–	80.00	75.50

Table 6: Train on DocNLI, test on NLP tasks that are out-of-domain and require document-level NLI. SOTA of MCTest comes from (Yu et al., 2019).

ous SOTA fact-checking system. We combine "refute" and "not-enough-info" as a single class "not_entail", and rename this data as "FEVER-binary". We randomly split FEVER-binary by 203,152/8,209/10,000 for train/dev/test respectively.[7]

MCTest (Richardson et al., 2013). In Related Work, we have introduced MCTest. Briefly, it is a multi-choice QA benchmark in the domain of **fictional story**. The authors of MCTest released an NLI-version MCTest by combining the question and the positive (resp. negative) answer candidate as a positive (resp. negative) hypothesis.

MCTest consists of two subsets. MCTest-160 contains 160 items (70 train, 30 dev, 60 test), each consisting of a document, four questions followed by one correct answer and three incorrect answers and MCTest-500 500 items (300 train, 50 dev, 150 test). *MCTest has pretty limited labeled data; thus, it is a good testbed to investigate* DocNLI *in studying annotation-scarce tasks.* The MCTest has two official metrics: accuracy and NDCG (Normalized Discounted Cumulative Gain). Here we only report accuracy.

In this section, we still use RoBERTa-large and compare our DocNLI with a latest NLI dataset ANLI in which the premises are longer than single sentences, and MNLI, the most widely-used sentence-level NLI dataset. For each data set (i.e., MNLI, ANLI or DocNLI), we try two settings: (i) Using the data for pre-training, then do inference on FEVER-binary or MCTest directly without task-specific fine-tuning; (ii) First pre-training on the data, then fine-tune on FEVER-binary or MCTest.

In Table 6, DocNLI can consistently generalize better than ANLI and MNLI on the two NLP tasks FEVER-binary and MCTest. We notice that the pretrained model on DocNLI demonstrates very strong performance on the two end tasks, even without any fine-tuning on the task-specific examples. Especially for MCTest, both the "DocNLI (pretrain)" and "DocNLI+finetune" surpass the prior state-of-the-art by large margins.

	SciTail	b-MNLI
majority	60.33	66.66
ESIM (Chen et al., 2017)	70.60	–
De-Att (Parikh et al., 2016)	72.30	–
DGEM (Khot et al., 2018)	77.30	–
BERT-large	89.71	90.55
Longformer-base	92.23	92.03
RoBERTa-large	95.13	93.95
DocNLI (pretrain)	78.17	91.13
+finetune	96.04	94.07
Prior state-of-the-art	97.70	–

Table 7: Train on DocNLI, test on sentence-level NLI benchmarks with or without fine-tuning. The SOTA of SciTail was reported by the DeBERTa model (He et al., 2020).

4.3 Applying DocNLI to sentence-level NLI

To answer the question (**Q**3), we use SciTail and MNLI as target sentence-level NLI tasks. SciTail is from the science domain with two classes "entail" and "not_entail" split 23,596/1,304/2,126 (train/dev/test). MNLI covers a broad range of genres with three classes "entail/neutral/contradict" split 392,702/20k/20k (train/dev/test). Since the gold labels of the test set in MNLI are not publicly available and DocNLI is a binary classification task, we first unify MNLI's "neutral" and "contradict" into "not_entail", then build a new labeled test set by randomly sampling 13k from the original dev set (the remaining examples are the new dev set). So now we have train/dev/test of size 372,702/6,647/13k. We first try some popular Transformer-style models, such as BERT-large, RoBERTa-large and Longformer-base to check how much we can get by training a supervised system on the full training data. Afterwards, we build a classifier by training RoBERTa-large on DocNLI with or without SciTail/MNLI-specific fine-tuning.

Table 7 shows that: (i) The pretrained model on DocNLI indeed can generalize to some extend on both SciTail and MNLI. In particular, it gets

---

[7] Please note that this data released by (Thorne et al., 2018) is different from the one used in FEVER leaderboard.

SciTail accuracy 78.17 which is even higher than some task-specific fully-supervised models such as "ESIM", "De-Att" and "DGEM". The same pretrained system can also get comparable performance with BERT, Longformer and RoBERTa on binary-MNLI; this should be attributed to the strong generalization of ANLI towards MNLI (Nie et al., 2020); (ii) When do task-specific fine-tuning, our model can further improve the performance and get very close to the state-of-the-art in SciTail.

5 Summary

In this work, we collect and release a large-scale document-level NLI dataset DocNLI. It covers multiple genres and multiple ranges of lengths in both premises and hypotheses. We expect this dataset can help to solve some NLP problems that require document-level reasoning such as QA, summarization, fact-checking etc. In experiments, we show that DocNLI can yield a model generalizing well to downstream NLP tasks and some popular sentence-level NLI tasks.

Acknowledgments

The authors would like to thank Nitish Shirish Keskar, our colleague at Salesforce Research, for helping play the CTRL code, and thank the anonymous reviewers for insightful comments and suggestions.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *ACL*, pages 1657–1668.

Curation. 2020. Curation corpus base.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches - erratum. *Nat. Lang. Eng.*, 16(1):105.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, pages 177–190.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *CoRR*, abs/1809.02922.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL-HLT*, pages 107–112.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTaiL: A textual entailment dataset from science question answering. In *AAAI*, pages 5189–5197.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Evaluating the factual consistency of abstractive text summarization. *CoRR*, abs/1910.12840.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*, pages 280–290.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *AAAI*, pages 6859–6866.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *ACL*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*, pages 2249–2255.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In **SEM@NAACL-HLT* , pages 180–191. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *LREC*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jianxing Yu, Zhengjun Zha, and Jian Yin. 2019. Inferential machine comprehension: Answering questions by recursively deducing the evidence chain from text. In *Proceedings of ACL*, pages 2241–2251.

Rule Augmented Unsupervised Constituency Parsing

Atul Sahay [1]*, **Anshul Nasery** [1]*, **Ayush Maheshwari** [1],
Ganesh Ramakrishnan [1], **Rishabh Iyer** [2]
[1]Department of CSE, IIT Bombay, India
[2] The University of Texas at Dallas
{atulsahay, anshuln, ayusham, ganesh}@cse.iitb.ac.in
rishabh.iyer@utdallas.edu

Abstract

Recently, unsupervised parsing of syntactic trees has gained considerable attention. A prototypical approach to such unsupervised parsing employs reinforcement learning and auto-encoders. However, no mechanism ensures that the learnt model leverages the well-understood language grammar. We propose an approach that utilizes very generic linguistic knowledge of the language present in the form of syntactic rules, thus inducing better syntactic structures. We introduce a novel formulation that takes advantage of the syntactic grammar rules and is independent of the base system. We achieve new state-of-the-art results on two benchmarks datasets, MNLI and WSJ.[1]

1 Introduction

Syntactic parse trees have demonstrated their importance in several downstream NLP applications such as machine translation (Eriguchi et al., 2017; Zaremoodi and Haffari, 2018), natural language inference (NLI) (Choi et al., 2018), relation extraction (Gamallo et al., 2012) and text classification (Tai et al., 2015). Based on linguistic theories that have promoted the usefulness of tree-based representation of natural language text, tree-based models such as Tree-LSTM have been proposed to learn sentence representations (Socher et al., 2011). Inspired by the Tree-LSTM based models, many approaches were proposed do not require parse tree supervision (Yogatama et al., 2017; Choi et al., 2018; Maillard et al., 2019; Drozdov et al., 2019). However, (Williams et al., 2018; Sahay et al., 2021) have shown that these methods cannot learn meaningful semantics (not even simple grammar), though they perform well on NLI tasks. Recently, there has been surge in approaches using weak supervision in the form of rules for various tasks such as sequence classification (Safranchik et al., 2020), text classification (Chatterjee et al., 2020; Maheshwari et al., 2020), *etc.* These approaches have demonstrated the importance of external knowledge in both unsupervised and supervised setup. To the best of our knowledge, previous works on syntactic parse tree has not utilized such external information. In this paper, we propose an approach that leverages linguistic (and potentially domain agnostic) knowledge in the form of explicit syntactic grammar rules while building upon a state of the art, deep and unsupervised inside-outside recursive autoencoder (DIORA; (Drozdov et al., 2019)). DIORA is an unsupervised model that uses inside-outside dynamic programming to compose latent representations from all possible binary trees. We extend DIORA and propose a framework that harness grammar rules to learn constituent parse trees. We use context free grammar (CFG) productions for English language (like NP →VP NP, PP →IN NP, *etc*) as rules. Note that the construction of such a rule set is a one time effort and our method is independent of any underlying dataset. The rule sets used are available in our github repository.

Summarily, our main contributions are : (a) a framework (*cf.*, Section 3) that uses (potentially domain agnostic), off-the-shelf CFG to learn to produce constituent parse trees (b) two rule-aware loss functions (*cf.*, Section 3.1) that maximize some form of agreement between the unsupervised model and the rule-based model (c) experimental analysis (*cf.*, Section 4), demonstrating improvements on unsupervised constituency parsing over previous state-of-the art by over **3**% on two benchmark datasets.

2 Background and Related Work

A brief survey of latent tree learning models is covered in (Williams et al., 2018). Several prior works have explored the unsupervised learning of

---

[*] Equal contribution

[1] The source code of the paper is available at https://github.com/anshuln/Diora_with_rules.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4923–4932
August 1–6, 2021. ©2021 Association for Computational Linguistics

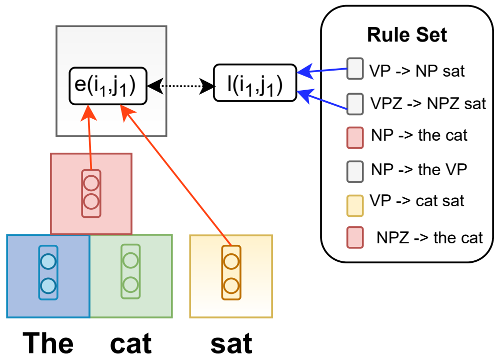

Figure 1: For the input 'The cat sat', DIORA computes $e(i_1, j_1)$ compatibility score for each pair of neighboring constituents. $l(i_1, j_1)$ is computed using triggered rules for each span and it interacts with the compatibility score in our loss function as explained in Section 3.1.

constituency trees (Brill et al., 1990; Ando and Lee, 2000) using dependency parsers (Klein and Manning, 2004) and inside-outside parsing algorithm (Drozdov et al., 2019). Recently, (Drozdov et al., 2019) proposed an unsupervised latent chart tree parsing algorithm, *viz.*, DIORA, that uses the inside-outside algorithm for parsing and has an autoencoder-based neural network trained to reconstruct the input sentence. DIORA is trained end to end using masked language model via word prediction. As of date, DIORA is the state-of-the-art approach to unsupervised constituency parsing.

Exploiting additional semantic and syntactic information that acts as a source of additional guidance rather than the primary objective function has been discussed since 1990s (Sun et al., 1993). Recently, Kim et al. (2019b) proposed to learn CFG rules and their probabilities by the parameterizing terminal or non-terminal symbols with neural networks. However, our approach leverages predefined language CFG rules and provisions for augmenting an existing (state-of-the-art) inside-outside algorithm with such external knowledge.

More specifically, we augment DIORA (Drozdov et al., 2019) with CFG rules to reconstruct the input by exploiting syntactic information of the language. We next provide some technical details of the inside-outside algorithm of DIORA.

2.1 DIORA

DIORA learns constituency trees from the raw input text using an unsupervised training procedure that operates like a masked language model or denoising autoencoder. It encodes the entire input sequence into a single vector analogous to the encoding step in an autoencoder. Thereafter, the decoder is trained to reconstruct and reproduce each input word. We next describe the inside and the outside pass of DIORA, respectively. *Inside Pass:* Given an input sentence with T tokens $x_0, x_1, x_2 \ldots x_{T-1}$, DIORA computes a *compatibility* score e and a *composition* vector $\bar{a}$ for each pair of neighboring constituents i and j. It composes a vector $\bar{a}$ weighing over all possible pairs of constituents of i and j: $\bar{a}(k) = \sum_{i,j\epsilon\{k\}} e(i,j)a(i,j)$ & $\bar{e}(k) = \sum_{i,j\epsilon\{k\}} e(i,j)\hat{e}(i,j)$ The composition vector, $\bar{a}(k)$ is a weighted sum of all possible constituent pairs, k. Here, $\hat{e}$ is a bilinear function of the vectors from neighboring spans, $\bar{a}(i)$ and $\bar{a}(j)$. Composition vector $\bar{a}(k)$ is learnt using a TreeLSTM or multi-layer neural network (MLP).

Outside Pass: The outside pass of DIORA computes an outside vector $\bar{b}(k)$ representing the constituents not in $x_{i:j}$. It computes the values for a target space (i, j) recursively from its sibling $(j+1, k)$ and outside spans $(0, i-1)$ and $(k+1, T-1)$.

Training and Inference: DIORA is trained end to end using masked language model via word prediction. The missing token x_i is predicted from the outside vector $\bar{b}(k)$. The training objective uses reconstruction based max-margin loss to predict the original input x_i:

$$L_{rec} = \sum_{i=0}^{T-1} \sum_{i^*=0}^{N-1} \max(0, 1 - \bar{b}(i).\bar{a}(i) + \bar{b}(i).\bar{a}(i^*))$$

The chart filling procedure of DIORA is used to extract binary unlabeled parse trees. It uses the CYK algorithm to find the maximal scoring tree in a greedy manner. For each cell of the parse table, the algorithm computes the span (i, j) with the maximal net compatibility score, computed recursively by summing the maximum compatibility score $e(a, b)$ for each constituent of the span.

3 Our Approach to Rule Augmentation

Our goal is to learn to produce constituency parse trees using input sentences alone and in the absence of ground truth parse trees. We introduce a rule-augmented unsupervised model that leverages generic (potentially domain agnostic) production rules of the language grammar to infer constituency trees. Since most grammar rules for constituency parsing are generic, designing them can be a one-time effort, while being able to leverage their benefits across domains as background knowledge (as we will see in our experiments in Section 4). As described in Section 2.1, the induction of latent trees in DIORA is based on a CYK-like parsing algorithm that uses the compatibility scores $e(i,j)$ at each cell to merge two constituents in the final tree. We impart supervision through the production rules of English language grammar.

For each sentence, we associate CFG production rules with constituents i and j in a CYK parse table format. We curate a set of domain-agnostic rules of the form $X \to YZ$ and a dictionary of the form $X \to x$, where X, Y, Z are non-terminals while x is a terminal. Concretely, X represents constituent tags such as S, NP, VP, etc., while x represents words in the vocabulary. Using the CYK parsing algorithm on our rule set and each sentence, we first determine which rules are triggered at each cell for a particular sentence. Whenever a rule r is triggered for a span (i,j), we weakly associate label $\delta_{(i,j)}(r) = 1$ otherwise 0. We use these weak labels to guide the rule scores $l(i,j)$ for the constituents. Compatibility score observed for the span (i,j) is defined as :

$$l(i,j) = \frac{\exp\left(\sum_{p=0}^{P} r_p \delta_{(i,j)}(p)\right)}{\sum_{(a,b)\in\{k\}} \exp\left(\sum_{p=0}^{P} r_p \delta_{(a,b)}(p)\right)}$$

where r_p are the *learned* weights associated with each of the production rules and P is the total number of rules. The score sums to 1 over all spans belonging to a particular cell in the CYK parse table. Intuitively, we aim to align $e(i,j)$ and $l(i,j)$ score to maximize the agreement between model and rules. We note that we use rules only to augment the training objective, and our inference procedure is identical to that of DIORA.

3.1 Training Objective

We learn a model that minimizes the overall loss L that is a composition of the reconstruction loss L_{rec} and the rule based agreement loss or L_{rule}: $L = L_{rec} + \lambda L_{rule}$. We propose two alternatives for the loss function L_{rule}.

Cross entropy (CE) - For each cell k in the CYK parse table, this loss (CE) tries *to match the distribution (score)* of $e(i,j)$ induced by DIORA with the distribution $l(i,j)$ induced by the background knowledge:

$$L_{ce} = \sum_{k} \sum_{(i,j)\in\{k\}} -l(i,j)\log(e(i,j))$$

Model	**F1**	δ
300D Gumbel Tree-LSTM	25.2	4.2
w/o Leaf GRU	29.0	4.7
300D RL-SPINN	19.0	8.6
w/o Leaf GRU	18.2	8.6
Structural Attentive(Gumble Tree LSTM)	31.3	4.7
w/o Leaf GRU	31.0	5.3
300D SPINN	**74.5**†	6.2
w/o Leaf GRU	65.7†	6.4
DIORA with PP	58.3	5.6
Ours: Rule augmented (HR) + RL + PP	60.5	5.7
Ours: Rule augmented (HR) + CE + PP	59.0	5.6
Ours: Rule augmented (AR) + RL + PP	60.3	5.7
Ours: Rule augmented (AR) + CE + PP	**61.7**	5.7

Table 1: F1-scores of trees wrt ground truth on the MultiNLI development set. The depth (δ) is the average tree height. All reported numbers are maximum F1-score. PP refers to post-processing heuristic. HR and AR refer to the training rule sets as per Sec 4.2. RL and CE refer to the losses from section 3.1. † indicates scores reported by (Williams et al., 2018) for a fully supervised model.

Ranking Loss (RL) - We recall from Section 2.1 that the CYK algorithm finds the maximal scoring tree in a greedy manner based on the highest compatibility score $e(i,j)$ among all spans. Since the final parse tree output by DIORA relies only on the *relative order* of the $e(i,j)$ to decide which span to merge, we propose an alternative rule-based loss that aims *to match the relative order* induced by compatibility scores $e(i,j)$ of DIORA at each cell with the order induced by the scores of the rules $l(i,j)$ at that cell. We achieve this through a pairwise ranking loss defined as

$$L_{rank} = \sum_{\text{k}} \sum_{\substack{(i,j)\\(i',j')}\in\{\text{k}_{trig}\}} \left(\Delta^{l}_{\substack{(i,j),\\(i',j')}} - \Delta^{e}_{\substack{(i,j),\\(i',j')}}\right)^2$$

where $\{\text{k}_{trig}\} = \{(i,j) \in \{k\} | \sum_{p=1}^{P} \delta_{i,j}(p) \neq 0\}$, $\Delta^{f}_{(i,j),(i',j')} = \tanh(f(i,j) - f(i',j'))$ and p is index into the rule set. The set $\{k_{trig}\}$ consists of all spans which have at least one rule triggered in its cell. In cases where our rule set is not extensive enough, we would like our model's compatibility score to rely more on the reconstruction loss, and $\{\text{k}_{trig}\}$ ensures that a sparse rule set does not lead to bad performance.

4 Experiments

We evaluate our rule augmented model and compare it against baselines on the tasks of unsuper-

Model	F1	δ
LB	13.1	12.4
RB	16.5	12.4
Random	21.4	5.3
Balanced	21.3	4.6
RL-SPINN (Choi et al., 2018)	13.2	-
ST-Gumbel - GRU (Yogatama et al., 2017)	22.8 ±1.6	-
PRPN-UP (Shen et al., 2018a)	38.3	5.9
PRPN-LM	35.0	6.2
ON-LSTM (Shen et al., 2018b)	47.7	5.6
DIORA	48.9	8.0
PRPN-UP+PP	45.2	6.7
PRPN-LM+PP	42.4	6.3
DIORA+PP	55.7	8.5
Neural PCFG (Kim et al., 2019a) *	50.8	-
Compound PCFG (Kim et al., 2019b)*	55.2	-
300D SPINN	**59.6**†	-
Ours: Rule augmented (HR) + RL + PP	56.5	7.1
Ours: Rule augmented (HR) + CE + PP	55.3	7.2
Ours: Rule augmented (AR) + RL + PP	55.9	7.1
Ours: Rule augmented (AR) + CE + PP	**58.3**	7.3

Table 2: Performance on WSJ test set for binary constituency parsing including punctuation characters. HR and AR refers to handcrafted and automated rules respectively. RL and CE are rule loss and cross entropy loss respectively. (δ) is the average tree height. PP refers to post-processing heuristic. † indicates scores reported by (Williams et al., 2018) for a fully supervised model. * are reported by (Kim et al., 2019a).

vised parsing, unsupervised segment recall, and phrase similarity.

4.1 Data

We evaluate our model on two data sets: The Wall Street Journal (WSJ) and MultiNLI. WSJ is an extraction of PennTree Bank (Marcus et al., 2002) containing human-annotated constituency parse trees. MultiNLI consists of Stanford generated parse trees (Manning et al., 2014) as the ground truth. MultiNLI is originally designed for evaluating NLI tasks, but is often also utilized to evaluate constituency parse trees. We train on the complete NLI dataset, which is a composition of the MultiNLI and SNLI train sets. We evaluate model performance on the MultiNLI dev set and WSJ test set (split 23) following the experimental setting and evaluation metrics in (Drozdov et al., 2019). Further details are provided in the appendix. We initialize our model with the trained weights of DIORA and evaluate on unsupervised constituency parsing and segment recall. We also perform the post-processing (PP) of generated trees by attaching the trailing punctuation to the root node, exactly as carried out by (Drozdov et al., 2019).

4.2 Rule Set

We consider two rule-sets: (i) **Set of Handcrafted Rules (HR)** consists of 2500 human created CNF production rules ii) To assess robustness of the rule-augmented method to the preciseness of the rule set, we present comparison by instead using a **set of Automated Rules' (AR)** which consists of the 2500 most frequently occurring CNF production rules extracted from the trees of automatically (using the Stanford CoreNLP parser) parsed SNLI corpus. Further details about these rule sets can be found in the appendix. We also use a train-set specific dictionary containing the POS (part-of-speech) tags of words in the training vocabulary for the terminal CFG productions for CYK parsing.

4.3 Unsupervised Parsing

In Tables 1 and 2, we present comparison between different approaches on the MultiNLI dev set and WSJ test set. We observe that our rule augmented approach outperforms the state of the art with respect to the max-F1 score. registering a maximum increase of 3.4 and 3.1 F1 points over DIORA respectively. The HR trained models outperform DIORA on both datasets, demonstrating that rule creation is indeed a one-time process and independent of domain. We also report parsing scores of a fully supervised model SPINN from (Williams et al., 2018) as an upper bound, and RL-SPINN (Choi et al., 2018), a distantly supervised model.

4.4 Constituency Segment Recall

In Table 3, we present the breakdown of constituent recall across the 6 most common types. Our approach achieves the highest recall across all the types and is the only model to perform effectively on SBAR and NP. Unlike other approaches, our approach consistently close to or the best recall score. We observe that rule augmentation using HR is more beneficial than AR with respect to precise evaluation measures such as Constituency, Segment Recall and Phrase Recall but yields smaller improvements than AR with respect to looser evaluation measures such as max F1 of Unsupervised Parsing. This can be possibly attributed to our observation that the extracted (most frequent) rules from SNLI, have (around 25%) higher coverage on the training set than HR, but appear to be semantically less precise.

Model	SBAR	NP	VP	PP	ADJP	ADVP
LB †	5%	11%	0%	5%	2%	8%
RB †	68%	24%	71%	42%	27%	38%
Random †	8%	23%	12%	18%	23%	28%
Balanced †	7%	27%	8%	18%	27%	25%
PRPN-UP (Shen et al., 2018a)	55.4%	59.8%	31.6%	60.2%	36.0%	50%
PRPN-LM	40.3%	68.7%	39.3%	49.7%	34.2%	39.2%
DIORA	61.3%	76.7%	62.8%	59.5%	60.4%	69.3%
PRPN (tuned)†	50%	59%	46%	57%	44%	32%
ON (tuned) (Shen et al., 2018b)	51%	64%	41%	54%	38%	31%
Neural PCFG (Kim et al., 2019a)	52%	71%	33%	58%	32%	45%
Compound PCFG (Kim et al., 2019b)	56%	74%	41%	68%	40%	52%
Ours: Rule augmented (HR)+ RL	**71.1%**	77.2%	65.8%	59.4%	**62.9%**	69.5%
Ours: Rule augmented (HR)+ CE	68.3%	75.4%	66.5%	**60.5%**	61%	**70.8%**
Ours: Rule augmented (AR)+ RL	71%	76.4%	**69.1%**	58.6%	61%	64.8%
Ours: Rule augmented (AR)+ CE	70%	**77.5%**	67%	58.6%	**62.2%**	70%

Table 3: Segment recall from WSJ by phrase type; † are reported by Kim et al. (2019a).

Model	CONLL 2000			CONLL 2012	
	P@1	P@10	P@100	P@1	P@10
DIORA	0.974	0.969	0.943	0.815	0.759
Ours: Rule augmented(AR)+ RL	0.976	0.968	0.941	0.813	0.755
Ours: Rule augmented(HR)+ CE	**0.978**	**0.97**	0.941	0.786	0.717
Ours: Rule augmented(AR)+ CE	0.970	0.965	0.937	0.809	0.745
Ours: Rule augmented(HR)+ RL	0.976	0.970	**0.944**	**0.824**	**0.760**

Table 4: Phrase similarity scores on CoNLL2000 and CoNLL 2012 tasks.

4.5 Phrase Similarity

We also employed the phrase similarity strategy followed by (Drozdov et al., 2019). Phrase Similarity scores measures the models capability to learn meaningful representation for spans of the text. Generally, most models focus more on generating the tokens representation and then use some ad-hoc arithmetic operations to generate representation for the larger spans of text thus losing the essence of the context that ties the words of the span.

To evaluate on the phrase similarity task we consider two data sets of labeled phrases: 1) CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000), which is a shallow parsed dataset and contains spans of verb phrases, noun phrases, preposition phrases *etc.*, and 2) CoNLL 2012 (Pradhan et al., 2012) which is a named entity dataset containing 19 different entity types. For the evaluation routine, we first generated the phrase representation of labeled spans whose length is greater than one. Cosine similarity is then used to obtain the similarity score of it with respect to all other labeled spans. We then calculate if the label for that query span matches the labels for each of the K most similar other spans in the dataset.In Table 4 we report precision@K for both datasets and various values of K. The baseline numbers are reported using the weights of DIORA provided by the authors.

5 Conclusion

In this work, we leverage linguistically grounded and domain agnostic CFG rules for language to induce parse trees and representations of constituent spans. We show that our approach augmented with generic, linguistically grounded grammatical rules, is easily able to outperform previous methods on constituency parsing and obtain higher segment recall.

Acknowledgements

We thank anonymous reviewers for providing constructive feedback. Ayush Maheshwari is supported by a Fellowship from Ekal Foundation (www.ekal.org). We are also grateful to IBM Research, India (specifically the IBM AI Horizon Networks - IIT Bombay initiative) for their support and sponsorship.

References

Rie Kubota Ando and Lillian Lee. 2000. Mostly-unsupervised statistical segmentation of japanese: Applications to kanji. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 241–248. Citeseer.

Eric Brill, David Magerman, Mitchell Marcus, and Beatrice Santorini. 1990. Deducing linguistic structure from the statistics of large corpora. In *Proceedings of the 5th Jerusalem Conference on Information Technology, 1990.'Next Decade in Information Technology'*, pages 380–389. IEEE.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3397–3404.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *North American Association for Computational Linguistics*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the joint workshop on unsupervised and semi-supervised learning in NLP*, pages 10–18.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. 2019a. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *International Conference on Learning Representations*.

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019b. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pages 478–485.

Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh K. Iyer, and Ganesh Ramakrishnan. 2020. Data programming using semi-supervision and subset selection. *CoRR*, abs/2008.09887.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2019. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *Natural Language Engineering*, 25(4).

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 2002. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5570–5578.

Atul Sahay, Ayush Maheshwari, Ritesh Kumar, Ganesh Ramakrishnan, Manjesh Kumar Hanawal, and Kavi Arya. 2021. Unsupervised learning of explainable parse trees for improved generalisation. *CoRR*, abs/2104.04998.

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018b. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 151–161.

GZ Sun, CL Giles, HH Chen, and YC Lee. 1993. The neural network pushdown automation: model, stack and learning simulations.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132.

Adina Williams, Andrew Drozdov*, and Samuel R Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

D Yogatama, P Blunsom, C Dyer, E Grefenstette, and W Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations (ICLR 2017)*. International Conference on Learning Representations.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1421–1429.

Appendix

A Training and evaluation details

We use the multi-layered neural network model provided by the authors of (Drozdov et al., 2019) to initialize our model's weights. We have a learning rate of 10^{-4} and a batch size of 32. Using these settings we train our models for 500000 steps. We use weights $\lambda_{ranking} = 10^{-1}$ and $\lambda_{ce} = 1.0$ for the rule based losses. All other model parameters are same as the ones set in (Drozdov et al., 2019). We run all our experiments on Nvidia RTX 2080Ti GPUs 12 GB RAM over Intel Xeon Gold 5120 CPU having 56 cores and 256 GB RAM. It takes about 2 days to train the model on NLI data.

For evaluation, we have reported the tree F-1 score for MNLI dev and WSJ test set. The metric computes the F-1 score for each tree based on the constituent spans induced in the predicted tree against the constituent spans in the ground truth. We further binarize the WSJ test set using the Stanford CoreNLP Parser and report scores on unlabelled binary trees.

We find that training with AR helps us achieve better results on both MNLI as well as on WSJ. This could be because extracted rules from SNLI have wider coverage on the training set than HR resulting in a stronger training signal and better performance. Further, our ranking loss performs better for HR extracted rules, indicating its efficacy with non-extensive rule sets, i.e. in the cases where the training signal is not rich. In such cases when some cells may not have any triggering rules, the ranking loss ensures that the model's decision is guided by the reconstruction loss.

We also find that the generic background knowledge of English grammar (HR) helps the model to better chunk constituents that are rarer (e.g. SBAR), while dataset-specific rules (AR) might benefit its overall tree structures more, leading to higher unlabelled F1 scores.

B Rule sets

In this work we utilise two distinct rule sets - (i) The first rule set (HR) consists of 2500 human created CNF production rules ii) the other set (AR) consisted of 2500 most frequently occurring CNF production rules extracted from the trees of automatically parsed SNLI corpus. All rules in both these sets consist only of non-terminals. The rules in (HR) come from observing human annotated parse trees from the PTB train set and consists of 2500 rules in the Chomsky Normal Form. The rules in (AR) are programatically extracted from the parse trees generated by running the Stanford Parser on the SNLI train set. We only retain the 2500 most frequently occurring productions from the set to match the size of the HR set. We note however that these rules have a higher coverage on the train data. We also provides `rules-AR.txt` and `rules-HR.txt` in the github repository.

In our training procedure, we aim to learn a weight r_p for each production rule p in our train set. Table 5 shows the top 10 most important (i.e. the ones with the highest r_p) rules from the grammar as determined by our models.

C Learnt Trees

In this section we present examples of trees induced by DIORA and our model. The first row of fig. 1 shows an example where our tree matches the ground truth exactly while DIORA does not, and the second row of fig. 1 shows an example where both models do not provide exact matches, but our model is able to capture the syntax better.

Model	**Top Rules**
AR ranking loss	NP —>PRP NP|CD-JJ-VBN-NNS>
	NP —>PRP NP|<NNP-NNP-NN>
	S —>S S|<CC-SINV-.>
	ADVP —>IN ADVP|<CC-JJ>
	VP —>VBN VP|<“-NP-”-PP>
	NP —>NP NP|<NN-NN-”>
	PP —>IN ,
	NP —>NP NP|<ADJP-PP-SBAR>
	NP —>NP NP|<NNS-S>
	PP —>“ PP|<IN-NP>
HR cross entropy	NP —>CD NNS
	S-CLR —>VP
	PP-PRP —>IN NP
	QP|<CD-TO-CD-CD>—>CD QP|<TO-CD-CD>
	S-2 —>NP S-2|<VP-.>
	VP —>VB VP|<NP-ADVP-S>
	NP|<,-”-SBAR>—>, NP|<”-SBAR>
	S —>NP S|<NP-VP-.>
	NP|<JJS-NNS>—>JJS NNS
	S-2|<VP-.>—>VP .
HR ranking loss	NP —>PRP NP|<NNP-CD-NN>
	NP —>DT NP|<JJ-NN-NN>
	S|<NP-VP-.—>—>NP S|<VP-.—>
	VP|<NP-S>—>NP S
	NP|<NNP-NNP-NNP-NN-NN>—>NNP NP|<NNP-NNP-NN-NN>
	NP —>JJ NP|<NN-POS>
	VP|<CC-,-VP>—>CC VP|<,-VP>
	NP —>NNP NP|<CC-NNS>
	NP|<:-SBAR>—>: SBAR
	ADJP —>$ CD
AR cross entropy	NP —>DT NP|<JJ-NNP-NNP-POS>
	VP —>VB VP|<NP-PRT>
	S|<S-:>—>S :
	VP —>VBZ VP
	PRN —>, PRN|<CC-PP>
	VP|<CC-VBG-NP-PP>—>CC VP|<VBG-NP-PP>
	NP|<,-NP-,-VP-.>—>, NP|<NP-,-VP-.>
	S|<PP-,-VP-.>—>PP S|<,-VP-.>
	PP —>RB PP|<CC-RB-NP>
	NP —>JJ NP|<NNP-NNP>

Table 5: Rules with the highest weights as learnt by our models

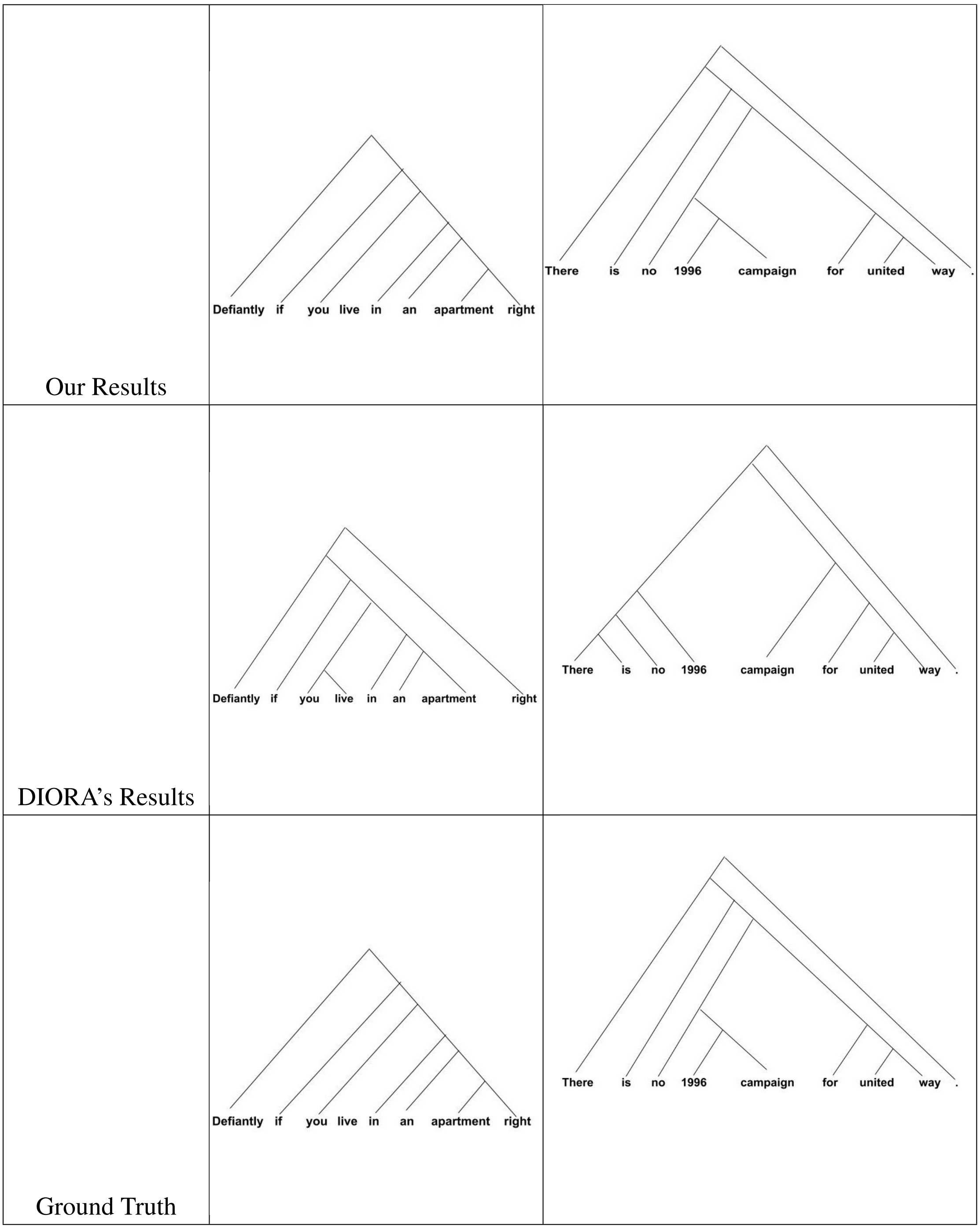

Figure 2: Comparison of induced trees by our model and DIORA with the ground truth trees

Are Multilingual Models the Best Choice for Moderately Under-Resourced Languages? A Comprehensive Assessment for Catalan

Jordi Armengol-Estapé, Casimiro Pio Carrino, Carlos Rodriguez-Penagos,
Ona de Gibert Bonet, Carme Armentano-Oller, Aitor Gonzalez-Agirre,
Maite Melero, Marta Villegas
Barcelona Supercomputing Center, Barcelona, Spain
{jordi.armengol, casimiro.carrino, carlos.rodriguez1,
ona.degibert, carme.armentano, aitor.gonzalez,
maite.melero, marta.villegas}@bsc.es

Abstract

Multilingual language models have been a crucial breakthrough as they considerably reduce the need of data for under-resourced languages. Nevertheless, the superiority of language-specific models has already been proven for languages having access to large amounts of data. In this work, we focus on Catalan with the aim to explore to what extent a medium-sized monolingual language model is competitive with state-of-the-art large multilingual models. For this, we: (1) build a clean, high-quality textual Catalan corpus (CaText), the largest to date (but only a fraction of the usual size of the previous work in monolingual language models), (2) train a Transformer-based language model for Catalan (BERTa), and (3) devise a thorough evaluation in a diversity of settings, comprising a complete array of downstream tasks, namely, Part of Speech Tagging, Named Entity Recognition and Classification, Text Classification, Question Answering, and Semantic Textual Similarity, with most of the corresponding datasets being created ex novo. The result is a new benchmark, the Catalan Language Understanding Benchmark (CLUB), which we publish as an open resource, together with the clean textual corpus, the language model, and the cleaning pipeline. Using state-of-the-art multilingual models and a monolingual model trained only on Wikipedia as baselines, we consistently observe the superiority of our model across tasks and settings.

1 Introduction

Over the past decades, Natural Language Processing (NLP) has become a powerful technology that may be found behind many AI-based consumer products, such as voice assistants, automatic translators, intelligent chatbots, etc. This undeniable success is somehow tarnished by the fact that most NLP resources and systems are available only for a small percentage of languages (Joshi et al., 2020).

In contrast, most of the languages spoken in the world today, even some with millions of speakers, are left behind, both in the research and in the development of the technology. Recent breakthroughs in deep learning, specifically the Transformer architecture (Vaswani et al., 2017), have revolutionized the entire field and have opened the doors to powerful transfer learning and unsupervised techniques, making it possible for under-resourced languages to benefit -at least, partially- from the formidable advances taking place for English. Transformed-based multilingual pre-trained models (Devlin et al., 2019; Conneau et al., 2020) soon showed an impressive increase of performance also for under-resourced languages, as they considerably reduce the amount of training data needed for a particular task.

The question as to whether training language-specific models was worth the effort, given those impressive results, was more or less quickly resolved, for languages having enough monolingual data to train with (Martin et al., 2020; Virtanen et al., 2019). However, for most languages it is still a challenge to obtain such large amounts of data. Therefore, the question still stands for many of them. There is a huge variation in the potential access to language resources for any given language, going from high-resourced, medium, under or severely under-resourced.

In this paper, we focus on Catalan, a moderately under-resourced language. By comparison, the size of the corpus -purposefully collected and cleaned for this work- is almost half of the one in the Finnish FinBERT (Virtanen et al., 2019) (a comparably sized language), and almost 20 times smaller than the French CamemBERT. Another defining characteristic of Catalan is its affiliation

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4933–4946
August 1–6, 2021. ©2021 Association for Computational Linguistics

to the Romance family, which is abundantly represented in multilingual pre-trained models. Several big languages (French, Spanish, Italian, Portuguese) belong to this family and, thus, are typologically close to Catalan. Presumably, this fact could give a Romance-rich multilingual model an upper hand in its comparison with a Catalan-only model.

This exercise also gives us the opportunity to enrich the number and quality of open-source resources for Catalan. Our contributions can be summarized as follows:

- We compile a clean, high-quality textual Catalan corpus, the largest to date, released with an open license.
- We build a complete end-to-end cleaning pipeline, released as open-source.
- We train a Transformer-based language model for Catalan, also openly released.
- We create new annotated corpora for those tasks for which there was not any, such as Text Classification (TC), Question Answering (QA) and Semantic Textual Similarity (STS). We publicly release them as well. Together with existing Part-Of-Speech (POS) and Named Entity Recognition and Recognition (NERC) datasets, they are part of a new Natural Language Understanding (NLU) benchmark, the Catalan Language Understanding Benchmark (CLUB).

All the code, datasets and the final model are made available to the community in standard formats.[1]

2 Previous Work

While the original motivation for the Transformer was machine translation, it has been successfully applied to a wide range of tasks, excelling at representation learning via unsupervised pre-training. Both in decoder-based language models, as in GPT (Radford and Sutskever, 2018), and in encoder-based masked language models, as pioneered by BERT (Devlin et al., 2019), large Transformer models are pre-trained on big unlabelled corpora. The learned representations can then be applied as a feature extractor or by fine tuning to the downstream task of choice, typically resulting in state-of-the-art performance.

The original BERT also had a multilingual version, mBERT, leveraging monolingual text from the corresponding Wikipedia of different languages. In XLM (Conneau and Lample, 2019), authors introduced a *cross-lingual* language model, explicitly modeling cross-lingual representations (instead of just concatenating text from all languages). This model was scaled up producing XLM-RoBERTa (Conneau et al., 2020), based on RoBERTa (Liu et al., 2019), a variant of BERT with a simplified pre-training objective.

Especially in the case of BERT-like models, which are intended for NLU tasks (rather than generation), the literature has been considerably prolific in terms of language-specific models. While both mBERT and XLM-RoBERTa are considered the state-of-the-art in NLU for numerous languages, several works observed that language-specific models trained from scratch proved to obtain better performance, such as the French CamemBERT (Martin et al., 2020), the Dutch BERTje (de Vries et al., 2019), and the Finnish FinBERT (Virtanen et al., 2019). FinBERT authors made emphasis on text cleaning, claimed to be essential. The same authors proposed WikiBERT (Pyysalo et al., 2020), a set of language-specific baselines based on BERT for as many as 42 languages (including Catalan). Regarding low-resource languages, Basque-specific models have been shown to outperform mBERT (Agerri et al., 2020), although it is worth pointing out that Basque, being a linguistic isolate, is typologically far apart from the rest of languages in the pre-training corpus of mBERT.

The authors of these language-specific models hypothesized different causes behind the increase in performance with respect to the multilingual models:

1. Having a language-specific vocabulary, they avoid the split of words into too many subwords (which, linguistically, are less interpretable);
2. The amount of language-specific data; and
3. Training on more diverse data of the target language (e.g., web crawlings, instead of just Wikipedia)

. In Nozza et al. (2020), they compare the performance of mBERT with a number of language-specific models. They conclude that there is a huge

---

[1] `https://github.com/TeMU-BSC/berta`

variability in the models and that it is difficult to find the best model for a given task, language, and domain.

In this work, we further investigate the issue of whether building language-specific models from scratch is worth the effort, and if so under which circumstances. Unlike previous works, we focus on a moderately under-resourced language, Catalan. In addition to being low-resource, Catalan is close to other Romance languages largely present in the pre-training corpora of multilingual models, and knowledge from these other Romance languages transfers well to Catalan. These two facts call into question whether building language-specific models for these cases is still interesting. We show that a Catalan-specific model is indeed relevant, and we provide tools and recipes for other languages in a similar situation. Besides the model itself, we build the largest Catalan (clean) pre-training corpus to date as well as an extensive evaluation benchmark for NLP tasks in Catalan.

3 Pre-training Corpus

3.1 Data sources

Our new Catalan text corpus, CaText, includes both data from datasets already available in Catalan and data from three new crawlers we recently ran.

From the published datasets, we use (1) the Catalan part of the DOGC corpus, a set of documents from the Official Gazette of the Catalan Government; (2) the Catalan Open Subtitles, a collection of translated movie subtitles (Tiedemann, 2012); (3) the non-shuffled version of the Catalan part of the OSCAR corpus (Suárez et al., 2019), a collection of monolingual corpora, filtered from Common Crawl;[2] (4) the CaWac corpus, a web corpus of Catalan built from the `.cat` top-level-domain in late 2013 (Ljubešić and Toral, 2014), the non-deduplicated version, and (5) the Catalan Wikipedia articles downloaded on 18-08-2020.

Regarding the newly created datasets, we ran three new crawlings: (6) the Catalan General Crawling, obtained by crawling the 500 most popular `.cat` and `.ad` domains; (7) the Catalan Government Crawling, obtained by crawling the `gencat.cat` domain and subdomains, belonging to the Catalan Government; and (8) the ACN corpus with 220k news items from March 2015 until October 2020, crawled from the Catalan News Agency.[3]

3.2 Preprocessing

In order to be able to obtain a high-quality training corpus, we apply strict filters to the raw data, by means of a cleaning pipeline built for the purpose, CorpusCleaner.[4] This pipeline supports 100+ languages[5] and stands out for keeping document boundaries, instead of being sentence-based, which allows modeling long-range dependencies.

Transforms: CorpusCleaner is able to parse data in different formats (WARC files from crawlings, for instance), in a way that document boundaries and metadata are kept whenever possible. It includes the `ftfy` library (Speer, 2019) for fixing encoding errors and applies raw string transformations for normalizing spaces, removing HTML tags, and others.

Filters: The pipeline applies a *cascade of language identifiers* (similarly to Kosmajac and Keselj, 2018), using FastText's (Bojanowski et al., 2017) language identifier and LangId.[6] By cascade, we mean that we first apply the faster language identifiers to discard documents in which we are sure that the target language is not present. We then use the slower (but with better performance) language identifiers only with the remaining documents. After that, we make use of a fast and tokenization-agnostic sentence splitter.[7] In addition, CorpusCleaner includes numerous heuristics (e.g., standard deviation of sentence length in a given document for detecting badly split sentences coming from PDFs, or non-natural text) that apply reasonably well to the languages we have tested so far. Some of these rules have been inspired by Virtanen et al. (2019). Even in the case of document-level corpora, some rules are also applied at sentence-level, to improve document coherency (e.g., remove corrupted sentences in an otherwise high-quality document).

Deduplication: The last stage of the pipeline consists of document-level deduplication, based on n-gram repetitions with Onion (Pomikálek, 2011). In addition, we also deduplicate at the sentence-level with a threshold of occurrences, since most of

---

[2] `https://commoncrawl.org/about/`

[3] `https://www.acn.cat/`

[4] `https://github.com/TeMU-BSC/corpus-cleaner-acl`

[5] We take this number from the used language identifiers. We have actually tested the cleaning pipeline with a variety of languages and domains, such as Basque, Finnish, Kazakh, Georgian and Biomedical Spanish, among others.

[6] `https://github.com/saffsd/langid.py`

[7] `https://github.com/mediacloud/sentence-splitter`

Dataset	Original	Final
1 DOGC	126.65	126.65
2 Cat. Open Subtitles	3.52	3.52
3 Cat. OSCAR	1,355.53	695.37
4 CaWaC	1,535.10	650.98
5 Wikipedia	198.36	167.47
6 Cat. Gen. Crawling	1,092.98	434.82
7 Cat. Gov. Crawling	303.10	39.12
8 ACN	81.28	75.61
Total	4,696.52	2,193.54
Deduplicated (CaText)		1,770.32

Table 1: Number of tokens (in millions) in the different used corpora before and after the filtering process, just before deduplication. The difference between these two columns shows the scale of the clean-up performed by the filters in the pipeline, particularly in the crawled data: OSCAR, CaWaC, Catalan General Crawling and Catalan Government Crawling. The last row the number of tokens after a global document deduplication across all corpora.

the often repeated sentences are not natural linguistic occurrences, but placeholders commonly used by web developers (e.g. copyright notices).

Table 1 shows the composition of the CaText corpus.

Splits and release: We sample 2,000 documents for validation, in order to monitor the training, and 2,000 more for test, as a hold-out set for future analysis. We release the preprocessed corpus CaText[8] with an open license, but respecting the licenses of the original data sources.[9]

3.3 Vocabulary

To build the language model, we use Byte-Level BPE (Radford et al., 2019), as in the original RoBERTa, but learning the vocabulary from scratch using the training set of CaText. Following recent works, we keep casing, and use a vocabulary size of 52k tokens (Scheible et al., 2020).

As shown in Table 2, our tokenization, being language-specific (and having a bigger vocabulary than WikiBERT-ca), generates less subwords per word, which has been shown to be beneficial (Virtanen et al., 2019; Agerri et al., 2020).

Model	Subwords per sentence
BERTa	**33.94**
mBERT	41.14
WikiBERT-ca	38.38
XLM-RoBERTa	38.62

Table 2: Subwords per sentence in the test set of CaText.

We describe the models used in our comparative study in Section 5 and Section 6. Here, we provide some examples of the different tokenization results depending on the used tokenizer:

```
original: coronavirus
BERTa: coronavirus
mBERT: corona ##vir ##us
WikiBERT-ca: corona ##vir ##us
XLM-RoBERTa: ▁corona virus

original: lamentablement
BERTa: lamentablement
mBERT: la ##menta ##blement
WikiBERT-ca: la ##menta ##ble ##ment
XLM-RoBERTa: ▁lamentable ment
```

As the examples show, our Catalan-specific model (BERTa) is more likely to keep the full word, even in the presence of derivative morphemes, such as the adverb 'lamentablement' (*unfortunately*). The fact that it does not split the word 'coronavirus' either reveals that a large part of the training corpus is very recent and includes many references to the COVID-19 pandemic. In the Appendix A we provide more detailed information about vocabulary overlapping between models.

4 CLUB: The Catalan Language Understanding Benchmark

In order to evaluate our model on different downstream tasks, we generate a new benchmark for evaluating NLU capabilities for Catalan, the Catalan Language Understanding Benchmark (CLUB).[10] To build it, we bootstrap from existing resources (such as the Ancora corpus[11]) and create new high quality ones from scratch, adopting (and in some cases improving on) existing guidelines for well known benchmarks. These datasets are publicly available in the Zenodo platform,[12] under the Catalan AI language resources community, where

---

[8] http://doi.org/10.5281/zenodo.4636228

[9] Regarding the two crawlings run in-house, we have also opted to release them with a more open license so they can be used freely (see https://zenodo.org/record/4636228 and https://zenodo.org/record/4636899).

[10] See https://github.com/TeMU-BSC/berta.

[11] https://doi.org/10.5281/zenodo.4762030

[12] https://zenodo.org/

further details about the curation and annotation guidelines for each one are given. In general, we provide as much information as possible following Bender and Friedman (2018) guidelines when relevant. For example, gender and socioeconomic status are considered not as relevant for the kind of semantic annotations created. However, the fact that all commissioned annotators (1) were native speakers of Catalan, (2) were translators, editors, philologists or linguists, and (3) had previous experience in language-related tasks, *is* considered to be important. The curation rationale we follow wis to make these datasets both representative of contemporary Catalan language use, as well as directly comparable to similar reference datasets from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019).[13]

For **Part-of-Speech Tagging (POS)** and **Named Entity Recognition and Classification (NERC),** we use annotations from the Universal Dependencies treebank[14] of the well-known Ancora corpus, which uses UPOS tags. We extracted named entities from the original Ancora[15] version, filtering out some unconventional ones, like book titles, and transcribe them into a standard CONLL-IOB format.

For **Semantic Textual Similarity (STS)** (Agirre et al., 2012), we create a new dataset from scratch. We use different similarity measures (Jaccard, Doc2Vec (Le and Mikolov, 2014) and DistilBERT (Sanh et al., 2019) embedding cosine similarity) to pre-select potential sentence pairs from the aforementioned CaText corpus, as well as a final manual review to ensure that the selection represented superficial and deep similarities in subject matter and lexicon. This results in 3,073 pairs for manual annotation. Following the guidelines set in the series of landmark SemEval challenges,[16] we commission 4 native speaker annotators to assess the similarity of the sentence pairs on a scale between 0 (*completely dissimilar*) to 5 (*completely equivalent*), with other possible values, such as 3 (*roughly equivalent, but some important information differs*). Then, for each sentence pair, we compute the mean of the four annotations, and we discard single annotations that deviate by more than 1 from the mean. After this cleaning process, we use the mean of the remaining annotations as a final score. Finally, in order to assess the annotation quality of the dataset, we measure the correlation of each annotator with the average of the rest of the annotators, and average all the individual correlations, resulting in a Pearson correlation of 0.739.[17]

For **Text Classification (TC)**, we use 137k news pieces from the Catalan News Agency (ACN) corpus, mentioned in Section 3.1. As labels in our classification task, we used the article category provided by the metadata, keeping only those categories that had more than 2,000 articles. See the Appendix B for more details on the distribution by label. We call this benchmark TeCla (Text Classification Catalan dataset).[18]

Finally, for extractive **Question Answering (QA)**, we compile two datasets:

- The Catalan translation of XQuAD (Artetxe et al., 2020), a multilingual collection of manual translations of fragments from English Wikipedia articles used mainly for cross-lingual analyses. The Catalan dataset, XQUAD-ca,[19] as the rest of languages, includes a subset of 240 paragraphs and 1,190 question-answer pairs from the development set of SQuAD v1.1 (Rajpurkar et al., 2016), and has no adversarial or unanswerable questions.

- A new dataset, ViquiQuAD, an extractive QA dataset from Catalan Wikipedia)[20] consisting of more than 15,000 questions outsourced from Catalan Wikipedia. We randomly choose a set of 596 articles which were originally written in Catalan, i.e. not translated from other Wikipedias. From those, we randomly select 3,129 short paragraphs to use as contexts and ask annotators to create up to 5 questions that could be answered by quoting directly from the context provided. In the Appendix B, we provide some statistics on our QA datasets and list the types of questions, comparing ViquiQuAD and XQuAD-

---

[13] `https://gluebenchmark.com/`

[14] `https://github.com/UniversalDependencies/UD_Catalan-AnCora`

[15] `https://doi.org/10.5281/zenodo.4762030`

[16] `http://ixa2.si.ehu.eus/stswiki`

[17] `https://doi.org/10.5281/zenodo.4529183`

[18] `https://doi.org/10.5281/zenodo.4627197`

[19] `https://doi.org/10.5281/zenodo.4526223`

[20] `https://doi.org/10.5281/zenodo.4562344`

Task	Total	Train	Dev	Test
NERC	13,581	10,628	1,427	1,526
POS	16,678	13,123	1,709	1,846
STS	3,073	2,073	500	500
TC	137,775	110,203	13,786	13,786
QA	14,239	11,255	1,492	1,429

Table 3: Dataset splits with number of examples.

ca with XQuAD-en and the French FQuAD (d'Hoffschmidt et al., 2020).

These benchmarks present a well-balanced set of challenges with which to test and compare our model with others. Apart from the diversity in the tasks themselves, for QA we provide an additional test from a different distribution (XQuAD-ca), and the data regime is heterogeneous across datasets (from more than 100k samples in TC to 3k in STS).

5 BERTa: A Model for Catalan

Following other language-specific models, we train a RoBERTa (Liu et al., 2019) base model (110M parameters), omitting the auxiliary Next Sentence Prediction task used in the original BERT, and just using the masked language modeling as the pre-training objective. The model is trained for 48 hours using 16 NVIDIA V100 GPUs of 16GB DDRAM, instead of 32 GB as in most works. For fitting an effective batch size of 2,048 sequences, we use gradient accumulation as implemented in Fairseq (Ott et al., 2019). Otherwise, we use the same hyperparameters as in the original RoBERTa, with a peak learning rate of 0.0005. We feed entire documents[21] instead of stand-alone sentences, fostering the modeling of long-range dependencies. We refer to the resulting model as BERTa.

6 Evaluation

We evaluate BERTa comparing the performances with two well-known multilingual baselines, mBERT and XLM-RoBERTa,[22] and another monolingual baseline, WikiBERT-ca (the Catalan WikiBERT by Pyysalo et al., 2020), on a variety of downstream tasks.

6.1 Fine-tuning

For evaluating our model against the existing baselines, we use common practices in the literature. For doing so, we leverage the Huggingface Transformers library (Wolf et al., 2019). For each task, we attach a linear layer to the models and fine tune with the training set of the specific dataset. For tasks involving tokens classification, we use the first token of the last output layer. Specifically, in the case of the BERT model, we use the `[CLS]` token while for the RoBERTa models we use the `<s>` token. We train each model under the same settings (see Table 3 for dataset splits) across tasks consisting of 10 training epochs, with an effective batch size of 32 instances, a max input length of 384 tokens and a learning rate of $5e^{-5}$. The rest of the hyperparameters are set to the default values in Huggingface Transformers.[23] We select the best checkpoint as per the task-specific metric in the corresponding validation set, and then evaluate it on the test set. We report the results and metrics used in Table 4.

6.2 Impact of fine-tuning data size

To show the impact of the fine-tuning data size on the tasks performances, we incrementally increase the fine-tuning data size and fine-tune on downstream tasks. We choose TC and QA because they have enough data to study the size effect across several magnitudes ranging from a minimum of 10^2 to a maximum of 10^4 examples. In the case of TC, we reduce the fine-tuning data by performing a stratified sampling that preserves the original distribution of examples per label, while for QA we just sample random examples corresponding to a given size. Figures 1, for QA, and 2, for TC, show the results of the studied models in the test set of the corresponding tasks, when progressively increasing the amount of instances in the train set.

6.3 Data contamination

As studied in Brown et al. (2020), data contamination refers to the inclusion of content from test or development sets into the training corpora used for language model pre-training. Consequently, the effect of contamination might have an impact on the downstream tasks performances, distorting them. In our case, we measure the effect of data

---

[21] We truncate those documents exceeding the maximum number of tokens.

[22] In this work, we always use a base model version with 12 layers, for a fair comparison.

[23] `https://github.com/huggingface/transformers/blob/master/src/transformers/training_args.py`

model	NERC	POS	STS	TC	QA (ViquiQuAD)	QA (XQuAD)
BERTa	88.13 (2)	**98.97** (10)	79.73 (5)	**74.16** (9)	**86.97/72.29** (9)	**68.89/48.87** (9)
+ decontaminate	**89.10** (6)	98.94 (6)	**81.13** (8)	73.84 (10)	86.50/70.82 (6)	68.61/47.26 (6)
mBERT	86.38 (9)	98.82 (9)	76.34 (9)	70.56 (10)	86.97/72.22 (8)	67.15/46.51 (8)
WikiBERT-ca	77.66 (9)	97.60 (6)	77.18 (10)	73.22 (10)	85.45/70.75 (10)	65.21/36.60 (10)
XLM-RoBERTa	87.66 (8)	98.89 (10)	75.40 (10)	71.68 (10)	85.50/70.47 (5)	67.10/46.42 (5)

Table 4: Results for the downstream tasks using different metrics. We use F1 for POS and NERC, accuracy for TC, an average of Pearson and Spearman coefficient for STS and F1/Exact Match for QA. We also report within round brackets the best epoch on the dev set.

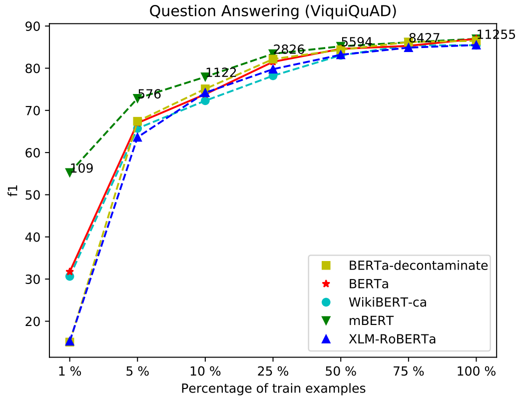

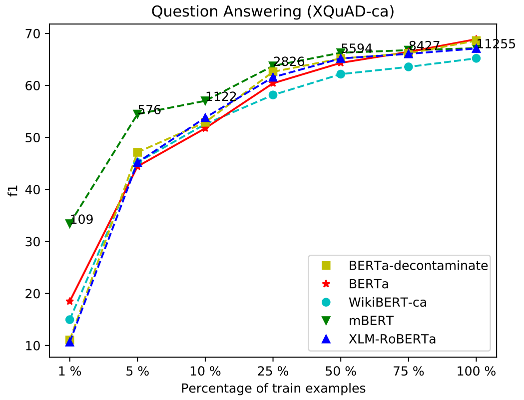

Figure 1: QA performance depending on the number of training examples.

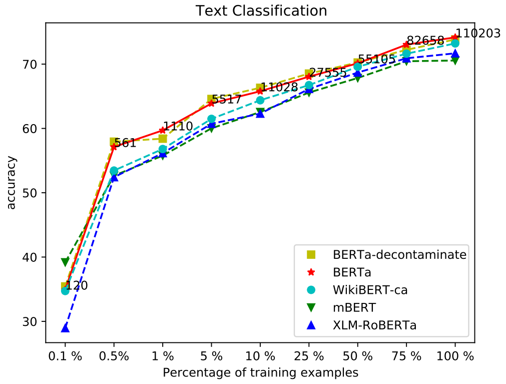

Figure 2: TC performance depending on the number of training examples.

contamination produced by those sentences used during language model training and, after being annotated, as part of our evaluation benchmark. Specifically, the STS, TC, and QA benchmarks (unlike the NERC and POS ones) are indeed contaminated since some examples are built from the Catalan Wikipedia and ACN sources. Actually, about the 0.078% of the training corpora (57,187 out of 73,172,152 sentences) are responsible for contamination. Therefore we train our BERTa model in two settings, one which includes these sentences and the other which does not include them. Note that all the compared models (mBERT, WikiBERT-ca, XLM-RoBERTa) have some degree of data contamination, since all of them include the Catalan Wikipedia as part of their training corpora. We show the results of BERTa along with BERTa-decontaminate in Table 4.

7 Discussion

Downstream tasks: Our model obtains better results in different benchmarking settings, ranging from simpler tasks with large training data sets (TC) to more advanced tasks with medium data (QA) and really complex tasks with small training data (STS).

In the case of NERC, BERTa clearly outperforms the existing baselines. In the case of POS, it also obtains better results, but only by a slight margin, being a considerably easy task, for which the existing models are already extremely competitive. For TC, BERTa also outperforms mBERT and XLM-RoBERTa models (by a larger margin when it has not been decontaminated). For QA, as measured in the ViquiQuAD dataset, only the version of BERTa that has not been decontaminated equals mBERT and is slightly better than XLM-RoBERTa. Note, however, that both mBERT and XLM-RoBERTa are equally *contaminated*, as pointed out in 6.3. When measured in the XQuAD-ca dataset, BERTa outperforms both models. Finally, for STS, BERTa outperforms mBERT and XLM-RoBERTa in 4.79 and 3.95 points respectively.

Interestingly, in the case of the XQuAD test, we are evaluating with a test set from a different distribution (i.e. it does not belong to our original split). Here, BERTa also outperforms all the baselines.

Error analysis and Question Answering: For the QA task, many of the errors detected are due to the inclusion or not of articles and punctuation in the answers. Based on this observation, we adopt the strategy introduced in Lewis et al. (2020) and we ignore initial articles when evaluating. The F1 results improve by at least one percentage point, while the exact match metric increases even more when measured using this MLQA[24] evaluation standard that discounts for initial articles.

Size matters, quality too: One of the goals of our work has been to estimate the effect of corpus size in the results. Although to the best of our knowledge, CaText is one of smallest datasets used to train monolingual language models, we note that this data is roughly 8 times the size of the Catalan Wikipedia, used to train WikiBERT-ca, as well as the Catalan portion of mBERT. Interestingly, XLM-RoBERTa, built on a filtered version of CommonCrawl, contains a portion of Catalan comparable in size to CaText.[25] In the line of FinBERT (Virtanen et al., 2019), we think that BERTa compensates for the relative scarcity of pre-training data with the quality of this data by using a rigorous filtering process. This is supported by the fact that XLM-RoBERTa, with a Catalan portion comparable in size to our corpus, has a consistently lower performance, not only with respect to our model, but also with respect to mBERT, which contains a smaller but cleaner portion of Catalan. While results for WikiBERT-ca seem to indicate that a Catalan model solely trained on the Wikipedia may not be enough in most cases, our results show that a model with sufficient diverse monolingual data that has been curated and cleaned, can outperform large multilingual models. Still, the question of how much data is enough remains unanswered.

Decontamination: We experimentally observe that our model generally benefits from data contamination in terms of performance, with a remarkable difference in the exact match metrics for QA, as shown in Table 4.

A more intricate error analysis on the predictions for QA of both our BERTa models (with and without contamination) sheds light on the issue of decontamination. While the sole difference between these two is the inclusion of these sentences, we observe an improvement in exact match correct answers for the model that has not been decontaminated, and suspect this might be due to the fact that the model has indeed memorized the questions; nonetheless, more investigation is needed in this regard.

This posits the question of whether the metrics obtained in these kinds of benchmarks are only indicative of the actual performance in the tasks with sentences different from those in the pre-training corpus, or otherwise reflect a certain memorization of the evaluation sentences. Still, the decontaminated BERTa outperforms the baselines in all test sets (including the explicitly decontaminated ones) except for ViquiQuAD (in which mBERT is superior to the decontaminated BERTa), hinting that BERTa is better than the other models regardless of the possible effects of their contamination.

Note also that for the TC setting, the monolingual WikiBERT-ca model also beats multilingual models. In this scenario, neither WikiBERT-ca nor the multilingual models have unlabeled data from the training set in their pre-training datasets. It is, therefore, an uncontaminated scenario where a small monolingual model clearly beats the multilingual ones.

Varying fine-tuning data size in downstream tasks: We experiment with different data sizes for the train sets of the QA and TC tasks, as shown in Figure 1.[26] Surprisingly, mBERT starts from a relatively high score, especially for QA, showing remarkably transfer capabilities when only a few examples are available. We point out that in the case of TC, as shown in Figure 2, starting from about 500 examples, BERTa models always remain above the baselines. Overall, all the curves show a constant progression up to 75% of training data and then seems to exhibit a slightly decreasing tendency in the case of multilingual models indicating they are approaching a plateau. Instead, monolingual ones still displays signs of improvement. For TC, Catalan-specific models, especially BERTa, show

---

[24] `https://github.com/facebookresearch/MLQA`

[25] See Appendix in Conneau et al. (2020).

[26] This figure shows F1 score. In the Appendix C, we provide the results for the exact match evaluation as well.

a better performance. Having a language-specific vocabulary may be helpful to further exploit increasing fine-tuning data.

8 Conclusions

We have demonstrated that it is worth to build a monolingual model for a moderately under-resourced language, even if it belongs to an over-represented linguistic family in multilingual models, such as the Romance language family. Our model outperforms the multilingual SOTA scores in all downstream tasks, as well as the monolingual scores from WikiBERT-ca, trained on a smaller and less varied corpus. Furthermore, by developing this model, we have contributed to the creation of open-source resources for Catalan, both for training and evaluation, that will encourage the development of technology in this language. We believe our methods can serve both as a recipe and a motivation for other languages in similar situations. In addition, we also release the cleaning pipeline used to build CaText, supporting 100+ languages.

As future work, we suggest further investigating the effect of contamination by explicitly studying the relation between memorization and performance in the exact match evaluation in QA. In addition, we propose conducting experiments for quantifying the effect of the data size, data cleaning, and data diversity (e.g., just Wikipedia vs. crawlings).

Acknowledgements

This work was partially funded by the Generalitat de Catalunya through the project PDAD14/20/00001, the State Secretariat for Digitalization and Artificial Intelligence (SEDIA) within the framework of the Plan TL,[27] the MT4All CEF project,[28] and the Future of Computing Center, a Barcelona Supercomputing Center and IBM initiative (2020).

We thank all the reviewers for their valuable comments.

Broader impacts

Regarding possible environmental concerns, training new language-specific models is costly[29] and one could argue that it could be avoided if a multilingual baseline performs well enough. However, we observe improvements that justify training from scratch another model. In addition, language-specific models are potentially more efficient, including inference, since their tokenizers generate less tokens.

As far as the model itself is concerned, we hypothesize that the pre-training corpus will have different biases and the model might reproduce them, so users must be aware of this issue. Nevertheless, with this work we contribute to an under-resourced language and open the door to follow similar approaches to other languages in similar situations, which can encourage a less English-centric view in the field of NLP.

References

Rodrigo Agerri, Iñaki San Vicente, Jon Ander Campos, Ander Barrena, Xabier Saralegi, Aitor Soroa, and Eneko Agirre. 2020. Give your text representation models some love: the case for Basque. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4781–4788, Marseille, France. European Language Resources Association.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss,

---

[27] `https://www.plantl.gob.es/`

[28] `https://ec.europa.eu/inea/en/connecting-europe-facility/cef-telecom/2019-eu-ia-0031`

[29] See `https://www.bsc.es/marenostrum/marenostrum` for the estimated power consumption.

Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7057–7067.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Martin d'Hoffschmidt, Wacim Belblidia, Quentin Heinrich, Tom Brendlé, and Maxime Vidal. 2020. FQuAD: French question answering dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1193–1208, Online. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

D. Kosmajac and V. Keselj. 2018. Slavic language identification using cascade classifier approach. In *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6.

Quoc V. Le and Tomás Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Nikola Ljubešić and Antonio Toral. 2014. caWaC – a web corpus of Catalan and its application to language modeling and machine translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1728–1732, Reykjavik, Iceland. European Language Resources Association (ELRA).

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific bert models.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Jan Pomikálek. 2011. onion. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sampo Pyysalo, Jenna Kanerva, Antti Virtanen, and Filip Ginter. 2020. Wikibert models: deep transfer learning for many languages.

Alec Radford and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *arxiv*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Raphael Scheible, Fabian Thomczyk, Patric Tippmann, Victor Jaravine, and Martin Boeker. 2020. Gottbert: a pure german language model.

Robyn Speer. 2019. ftfy. Zenodo. Version 5.5.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for finnish. *CoRR*, abs/1912.07076.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. Bertje: A dutch BERT model. *CoRR*, abs/1912.09582.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

A Vocabularies

BERTa	mBERT	WikiBERT-ca	XLM-RoBERTa	Tokens number
✗				52,000
	✗			119,547
		✗		20,101
			✗	250,002
∩	∩			11,251
∩		∩		17,207
∩			∩	13,063

Table 5: Number of tokens and vocabularies intersection.

B Datasets

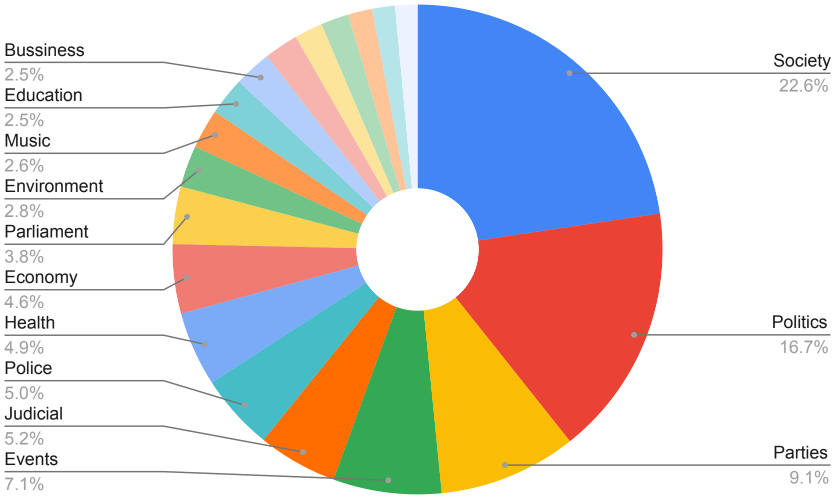

Figure 3: Label distribution of the Text Classification dataset. Here, we show the distribution of the filtered dataset, that is, keeping the labels with at least 2,000 instances.

	XQuAD-ca	**ViquiQuAD**
Paragraph	48	597
Context	240	3,124
Total sentences	1,176	10,315
Sentences/context	4.9	3.30
Tokens in context	40,056	470,932
Tokens in questions	15,417	145,827
Tokens in questions/questions	12.96	9.59
Tokens in questions/tokens in context	0.38	0.31
Tokens in answers	4,436	63,596
Tokens in answers/answers	3.73	4.18
Tokens in answers/tokens in context	0.11	0.13

Table 6: Statistics on the number of tokens in contexts, questions, and answers in our QA datasets

	XQuAD-ca	**ViquiQuAD**	**SQuAD**	**FQuAD**
Lexical variation	33.0%	7.0%	33.3%	35.2%
World knowledge	16.0%	17.0%	9.1%	11.1%
Syntactic variation	35.0%	43.0%	64.1%	57.4%
Multiple sentence	17.0%	9.0%	13.6%	17.6%

Table 7: Question-answer reasoning typology. For XQuAD and ViquiQuad, we sampled 100 random question-answer pairs and classified them manually.

	XQuAD-ca	**ViquiQuAD**	**XQuAD-en**	**FQuAD**
Which?	43.21%	22.08%	7.06%	47.8%
What?	16.42%	26.45%	57.31%	4.1%
Who?	10.94%	13.61%	10.00%	12.2%
How many?	8.3%	5.72%	6.55%	5.6%
How?	8.02%	12.41%	5.13%	6.8%
When?	6.79%	6.75%	7.14%	7.6%
Why?	1.51%	2.46%	1.26%	5.3%
Where?	3.58%	10.3 %	3.86%	9.6%
Other	1.23%	0.16%	1.93%	1.00%

Table 8: Question type frequencies. Differences between XQuAD-ca and XQuAD-en are explained because there is not an unique translation of the pronouns.

C Question Answering Evaluation

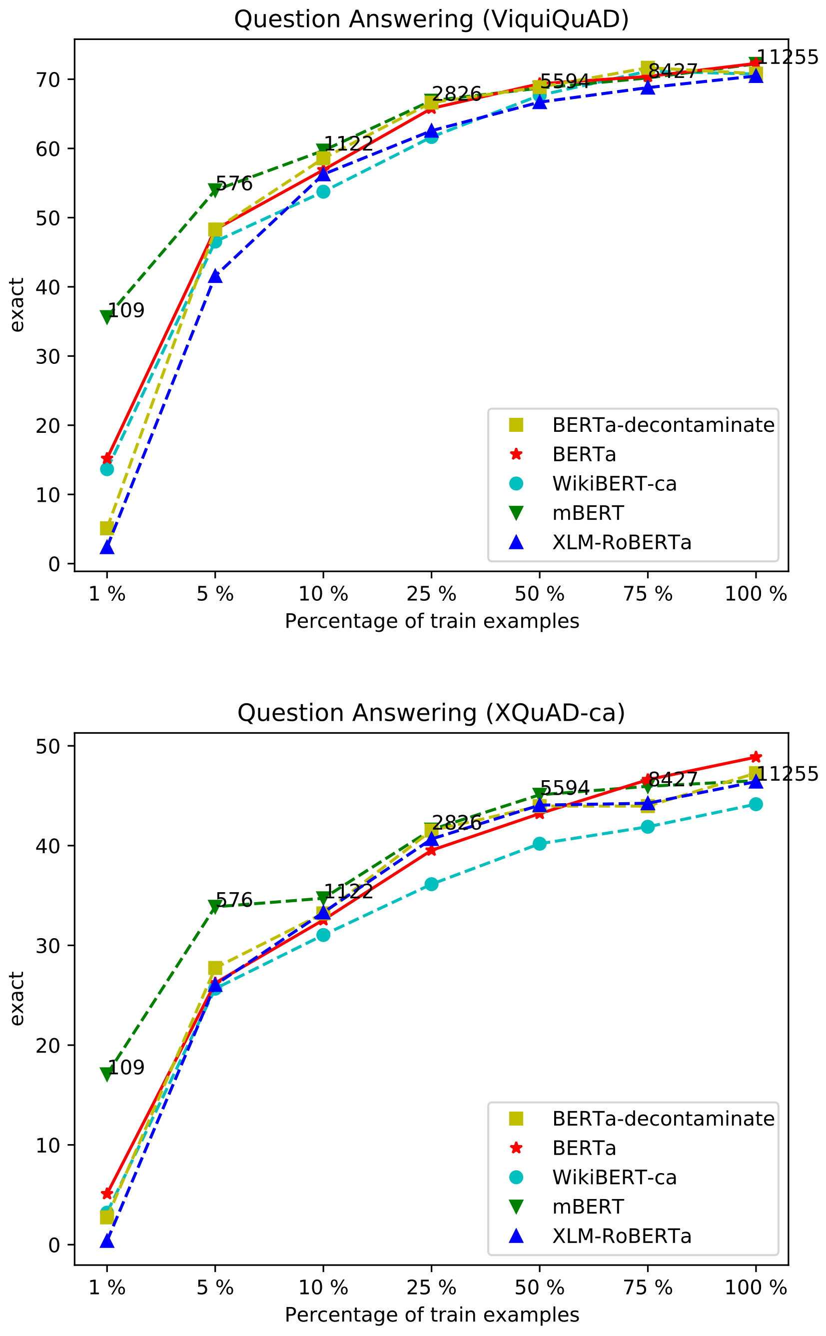

Figure 4: QA performance depending on the number of train instances (Exact match).

How transfer learning impacts linguistic knowledge in deep NLP models?

Nadir Durrani **Hassan Sajjad** **Fahim Dalvi**
`{ndurrani,hsajjad,faimaduddin}@hbku.edu.qa`
Qatar Computing Research Institute, HBKU Research Complex, Doha 5825, Qatar

Abstract

Transfer learning from pre-trained neural language models towards downstream tasks has been a predominant theme in NLP recently. Several researchers have shown that deep NLP models learn non-trivial amount of linguistic knowledge, captured at different layers of the model. We investigate how fine-tuning towards downstream NLP tasks impacts the learned linguistic knowledge. We carry out a study across popular pre-trained models BERT, RoBERTa and XLNet using layer and neuron-level diagnostic classifiers. We found that for some GLUE tasks, the network relies on the core linguistic information and preserve it deeper in the network, while for others it forgets. Linguistic information is distributed in the pre-trained language models but becomes localized to the lower layers post-fine-tuning, reserving higher layers for the task specific knowledge. The pattern varies across architectures, with BERT retaining linguistic information relatively deeper in the network compared to RoBERTa and XLNet, where it is predominantly delegated to the lower layers.

1 Introduction

Contextualized word representations learned in transformer-based language models capture rich linguistic knowledge, making them ubiquitous for transfer learning towards downstream NLP problems such as Natural Language Understanding tasks e.g. GLUE (Wang et al., 2018). The general idea is to pretrain representations on large scale unlabeled data and adapt these towards a downstream task using supervision.

Descriptive methods in neural interpretability investigate what knowledge is learned within the representations through relevant extrinsic phenomenon varying from word morphology (Vylomova et al., 2016; Belinkov et al., 2017a; Dalvi et al., 2017) to high level concepts such as structure (Shi et al., 2016; Linzen et al., 2016) and semantics (Qian et al., 2016; Belinkov et al., 2017b) or more generic properties such as sentence length (Adi et al., 2016; Bau et al., 2019). These studies are carried towards analyzing representations from pre-trained models. However, it is important to investigate how this learned knowledge evolves as the models are adapted towards a specific task from the more generic task of language modeling (Peters et al., 2018) that they are primarily trained on.

In this work, we analyze representations of 3 popular pre-trained models (BERT, RoBERTa and XLnet) with respect to morpho-syntactic and semantic knowledge, as they are fine-tuned towards GLUE tasks. More specifically we investigate i) if the fine-tuned models retain the same amount of linguistic information, ii) how this information is redistributed across different layers and individual neurons. To this end, we use *Diagnostic Classifiers* (Hupkes et al., 2018; Conneau et al., 2018), a popular framework for probing knowledge in neural models. The central idea is to extract feature representations from the network and train an auxiliary classifier to predict the property of interest. The quality of the trained classifier on the given task serves as a proxy to the quality of the extracted representations w.r.t to the understudied property (Belinkov et al., 2020).

We carry layer-wise (Liu et al., 2019a) and neuron-level probing analyses (Dalvi et al., 2019a) to study the fine-tuned representations. The former probes representations from individual layers w.r.t a linguistic property and the latter finds salient neurons in the network that capture the property. Fine-tuning involves adjusting feature weights, therefore it is important to look at the individual neurons to uncover important details, in addition to a more holistic layer-wise view.

Our layer-wise analysis shows: i) that some GLUE tasks rely on core linguistic knowledge and

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4947–4957
August 1–6, 2021.

the model preserves the information deeper in the network, while for others it is retained only in the lower layers ii) interesting cross-architectural differences with knowledge regressed to lower layers in RoBERTa and XLNet as opposed to BERT where it is still retained at the higher layers. Our neuron-wise analysis shows: i) salient linguistic neurons are relocated from the higher to lower layers, reinforcing our layer-wise results, ii) that linguistic information becomes less distributed and less redundant in the network post fine-tuning.

Finally, we show how our analysis entails findings in layer pruning. Dropping higher layers of the models maintains comparable performance to fine-tuning the full network, with linguistic information regressed to the lower layers. Conversely, pruning the lower layers (which hold the core linguistic information) leads to substantial degradation in performance.

In comparison to the related work done in this direction, our findings resonate with Merchant et al. (2020) who found that fine-tuning primarily affects top layers and does not lead to "catastrophic forgetting of linguistic phenomena" in BERT. However, we found that other models like RoBERTa and XLNet, which they did not study, see a substantial drop in accuracy even at the lower layers and start forgetting linguistic knowledge much earlier in the network. In contrast to Mosbach et al. (2020), we study core-linguistic phenomena whereas their study is based on sentence level probing tasks. Differently from both, we carry out a fine-grained neuron analysis which sheds light on how neurons are distributed and relocated post fine-tuning. Our work complements their findings while extending the layer-wise analysis to core-linguistic tasks and additionally looking at the distribution and relocation of neurons after fine-tuning.

2 Methodology

Our methodology is based on the probing framework called as *Diagnostic Classifiers*. We train a classifier using the activations generated from the trained neural network as static features, towards the task of predicting a certain linguistic property. The underlying assumption is that if the classifier can predict the property, the representations implicitly encode this information. We train layer- and neuron-wise probes using logistic-regression classifiers. Formally, consider a pre-trained neural language model $\mathbf{M}$ with L layers: $\{l_1, l_2, \ldots, l_L\}$. Given a dataset $\mathbb{D} = \{w_1, w_2, ..., w_N\}$ with a corresponding set of linguistic annotations $\mathbb{T} = \{t_{w_1}, t_{w_2}, ..., t_{w_N}\}$, we map each word w_i in the data $\mathbb{D}$ to a sequence of latent representations: $\mathbb{D} \overset{\mathbb{M}}{\longmapsto} \mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$. The model is trained by minimizing the following loss function:

$$\mathcal{L}(\theta) = -\sum_i \log P_\theta(t_{w_i}|w_i) + \lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2$$

where $P_\theta(t_{w_i}|w_i) = \frac{\exp(\theta_l \cdot \mathbf{z}_i)}{\sum_{l'} \exp(\theta_{l'} \cdot \mathbf{z}_i)}$ is the probability that word i is assigned property t_{w_i}. We extract representations from the individual layers for our layer-wise analysis and the entire network for the neuron-analysis. We use the *Linguistic Correlation Analysis* as described in Dalvi et al. (2019a), to generate a neuron ranking with respect to the understudied linguistic property: Given the trained classifier $\theta \in \mathbb{R}^{D \times T}$, the algorithm extracts a ranking of the D neurons in the model $\mathbb{M}$ based on weight distribution. The elastic-net regularization (Zou and Hastie, 2005) – a combination of $\lambda_1\|\theta\|_1$ and $\lambda_2\|\theta\|_2^2$ is used to strike a balance between identifying focused ($L1$) versus distributed ($L2$) neurons. The weights for the regularization terms are tuned using a grid-search algorithm.

Following Durrani et al. (2020), we extract salient neurons for a linguistic property by iteratively choosing the top N neurons from the ranked list and retrain the classifier using these neurons, until the classifier obtains an accuracy close (within a specified threshold δ) to the *Oracle* – accuracy of the classifier trained using all the features in the network.

3 Experimental Setup

Pre-trained Neural Language Models: We experimented with 3 transformer models: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b) and XLNet (Yang et al., 2019) using the base versions (13 layers and 768 dimensions). This choice of architectures leads to an interesting comparison between auto-encoder versus auto-regressive models. The models were then fine-tuned towards GLUE tasks of which we experimented with SST-2 for sentiment analysis with the Stanford sentiment treebank (Socher et al., 2013), MNLI for natural language inference (Williams et al., 2018), QNLI for Question NLI (Rajpurkar et al., 2016), RTE for recognizing textual entailment (Bentivogli et al., 2009), MRPC for Microsoft Research paraphrase corpus (Dolan and Brockett, 2005), and STS-B

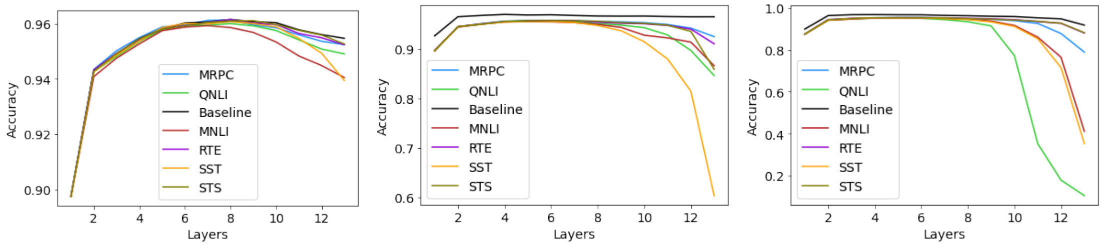

(a) BERT – POS (b) RoBERTa – POS (c) XLNet – POS

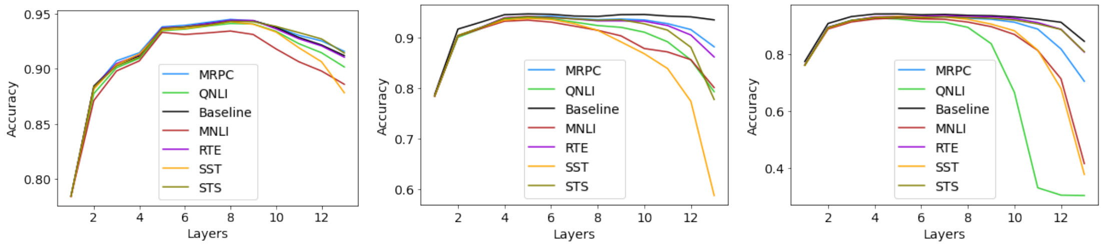

(d) BERT – Chunking (e) RoBERTa – Chunking (f) XLNet – Chunking

Figure 1: Layer-wise Probing Performance. Baseline refers to the performance of the pre-trained models without any finetuning.

for the semantic textual similarity benchmark (Cer et al., 2017). All the models were fine-tuned with the identical settings and we did 3 independent runs.

Linguistic Properties: We evaluated our method on 3 linguistic tasks: POS tagging using the Penn TreeBank (Marcus et al., 1993), syntactic chunking using CoNLL 2000 shared task dataset (Tjong Kim Sang and Buchholz, 2000), and semantic tagging using the Parallel Meaning Bank data (Abzianidze et al., 2017). We used standard splits for training, development and test data.

Classifier Settings: We used a linear probing classifier with elastic-net regularization, using a categorical cross-entropy loss, optimized by Adam (Kingma and Ba, 2014). Training is run with shuffled mini-batches of size 512 and stopped after 10 epochs. The regularization weights are trained using grid-search. For sub-word based models, we use the last activation value to be the representative of the word following Durrani et al. (2019). We computed *selectivity* (Hewitt and Liang, 2019) to ensure that our results reflect the property of representations and not the probe's capacity to memorize. Please see Appendix for details.

4 Analysis

4.1 Layer-wise Probing

First we train layer-wise probes to show how linguistic knowledge is redistributed across the network as we fine-tune it towards downstream tasks. Figure 1 shows results for POS and Chunking tasks.[1] We found varying observations across different GLUE tasks.

Comparing GLUE tasks: We found that linguistic phenomena are more important for certain downstream tasks, for example STS, RTE and MRPC where they are preserved in the higher layers post fine-tuning, as opposed to others, for example SST, QNLI and MNLI where they are forgotten in the higher layers. It would be interesting to study this further by connecting linguistic probes with any causation analysis on these tasks. Such an analysis would shed light on what concepts are used by the network while making predictions and why such information is forgotten for certain tasks. We leave this exploration for future.

Comparing Architectures: We found that pre-trained models behave differently in preserving information post fine-tuning. In the case of BERT, linguistic knowledge is fully preserved until layer 9, after which different task-specific models drop to varying degree, with SST and QNLI showing significant drop compared to others. An exception to this overall trend is MNLI where we start seeing a decline in performance earlier (between layers $5-7$). Contrastingly RoBERTa and XLNet show a depreciation in linguistic knowledge as early as layer 5. Also the drop is much more

---

[1] The observations are consistent for semantic tagging. Please see Appendix for results.

catastrophic in these two models with accuracy dropping by more than 35% in RoBERTa and 70% in XLNet. These results indicate that BERT retains its primarily learned linguistic knowledge and uses only a few of the final layers for fine-tuning, as opposed to XLNet and RoBERTa, where linguistic knowledge is retained only in the lower half of the network. Another cross-architectural observation that we made was that in RoBERTa and XLNet, the fine-tuned models do not ever reach the baseline performance (i.e. accuracy before fine-tuning – See Figure 1) at any layer, although the loss is $< 2\%$. We conjecture this discrepancy is due to the fact that the knowledge is more redundant and polysemous in the case of BERT, compared to XLNet, where it is more localized (also observed in Durrani et al. (2020)). Consequently, during fine-tuning XLNet and RoBERTa are more likely to lose linguistic information that is unimportant to the downstream task. We discuss this further in our neuron-analysis section.

4.2 Neuron-wise Probing

In our second set of experiments, we conducted analysis at a more fine-grained neuron level using *Linguistic Correlation Method* (Dalvi et al., 2019a). We extract the most salient neurons w.r.t a linguistic property (e.g. POS) and compare how the distribution of such neurons changes across the network as it is fine-tuned towards a downstream GLUE task. We use the weights of the trained classifier to rank neurons and select minimal set of salient neurons that give the same classifier accuracy as using the entire network in the baseline model. We found 5% neurons for POS and SEM tagging tasks and 10% for the Chunking tagging were sufficient to achieve the baseline performance.

Information becomes less distributed in the fine-tuned XLNet and RoBERTa models post fine-tuning: Table 1 shows accuracy of the classifier selecting the most (top) and least (bottom) 5% salient neurons on the task of POS tagging.[2] We observed that the bottom neurons in the fine-tuned models show a significant drop in performance, compared to the baseline model in the case of RoBERTa and XLNet. These results show that the information is more redundant in the baseline models as bottom neurons also preserved linguistic knowledge. On the contrary the information becomes more localized and less distributed in the fine-tuned models. The bottom neurons in the fine-tuned BERT changed the least, showing that linguistic information is still redundant and distributed in BERT.

Tasks	BERT		RoBERTa		XLNet	
	Top	Bot.	Top	Bot.	Top	Bot.
Base	96.0	94.9	96.7	95.3	96.5	91.2
MRPC	95.9	94.6	95.6	91.9	95.2	78.8
QNLI	96.0	94.6	95.8	84.3	94.7	10.3
MNLI	95.8	93.9	95.4	84.8	94.9	41.1
RTE	95.9	94.8	95.6	90.4	95.2	87.9
SST	95.9	94.2	95.6	60.4	95.0	35.2
STS	95.9	94.6	95.7	85.9	95.1	88.1

Table 1: POS accuracy – Top vs. Bottom neurons

How do salient neurons spread across the network layers? Previously we investigated how representations in each layer change w.r.t linguistic task. Now we study how the spread of the most salient neurons changes across the fine-tuned models. Figure 2 shows results for the selected GLUE tasks.[3] Notice how the most salient linguistic neurons shift from the higher layers towards the lower layers in RoBERTa and XLNet. This is especially pronounced in the case of *Roberta-SST* and *XLNet-QNLI* (See Figures 1e and 1f), where the number of salient chunking neurons significantly increased in the lower layers and droped in the higher layers, compared to the baseline. These findings reinforces our layer-wise results and additionally show how more responsibility is delegated to the neurons in the lower layers. Contrastingly, BERT did not exhibit this behavior. These results are inline with Durrani et al. (2020), who also found linguistic properties in XLNet to be localized to the lower layers[4] and fewer neurons and mutually exclusive as compared to BERT where neurons are highly polysemous[5] and therefore more redundant. Their finding helps us explain why XLNet forgets linguistic information that is unimportant to the downstream task more catastrophically.

5 Network Pruning

Our layer and neuron-wise analyses showed that core linguistic knowledge is redundant and distributed in the large pre-trained models. But as they are fine-tuned towards a down-stream task,

---

[2] See Appendix for SEM and Chunking tagging.

[3] See Appendix for all tasks and linguistic properties.

[4] Similarly (Wu et al., 2020) reported lower and middle layers of XLNet to have the most salient features.

[5] attend to multiple linguistic phenomenon

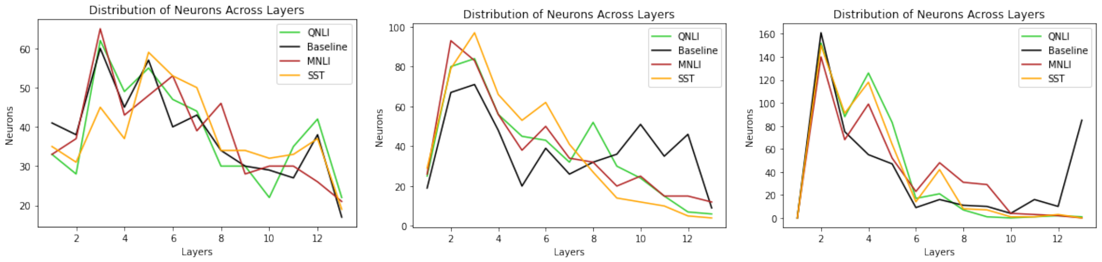

(a) BERT – SEM (b) RoBERTa – SEM (c) XLNet – SEM

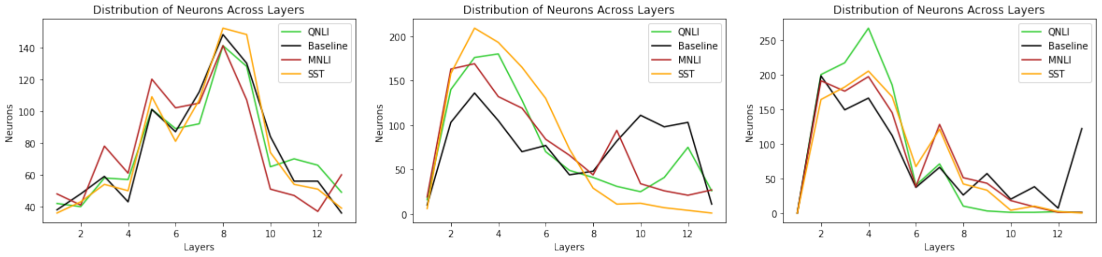

(d) BERT – Chunking (e) RoBERTa – Chunking (f) XLNet – Chunking

Figure 2: Distribution of top neurons across layers

it is relocated and localized to lower layers, with higher layers focusing on the task-specific information. In this section, we show that our findings explain patterns in layer pruning. We question **How important is the linguistic knowledge for these downstream NLP tasks?** Following Sajjad et al. (2020) we prune top and bottom (excluding the embedding layer) 6 layers of the network in two separate experiments and compare architectures. Table 2 shows that removing bottom layers of the network in RoBERTa and XLNet leads to more damage compared to BERT. **How do these findings resonate with our analysis?** We showed that BERT retains linguistic information even at the higher layers of the model as opposed to RoBERTa where it is preserved predominantly at the lower layers. Removing the bottom 6 layers in RoBERTa leads to a bigger drop because the network is completely deprived of the linguistic knowledge. Linguistic knowledge is more distributed in BERT and preserved at the higher layers also which leads to a smaller drop as it can still access this information. We leave a detailed exploration on this for future.

Tasks	**SST**	**MNLI**	**QNLI**
		BERT	
Baseline	92.4	84.0	91.1
Prune Top 6	90.3	81.2	87.6
Prune Bottom 6	88.1	78.4	83.7
		RoBERTa	
Baseline	92.2	86.4	91.7
Prune Top 6	92.0	84.4	90.0
Prune Bottom 6	83.7	61.6	63.7
		XLNet	
Baseline	93.9	86.0	90.4
Prune Top 6	92.2	83.5	88.0
Prune Bottom 6	87.5	68.1	83.0

Table 2: Pruning Layers in the Models

6 Conclusion

We studied how linguistic knowledge evolves as the pre-trained language models are adapted towards downstream NLP tasks. We fine-tuned three popular models (BERT, RoBERTa and XLNet) towards GLUE benchmark and analyzed representations against core morpho-syntactic knowledge. We used probing classifiers to carry out layer and neuron-wise analyses. Our results showed that morpho-syntactic knowledge is preserved at the higher layers in some GLUE tasks (e.g. STS, MRPC and RTE), while forgotten and only retained at the lower layers in others (MNLI, QNLI and SST). Comparing architectures, we found that BERT retains linguistic knowledge deeper in the network. In the case of RoBERTa and XLNet, the information is only preserved in the middle layers. This discrepancy is due to the fact that neurons in BERT are more polysemous and distributed as opposed to XLNet and RoBERTa where they are more localized (towards lower layers) and mutually exclusive. We showed that this difference in architectures, entails different patterns as we prune top or bottom layers in the network. Our code is publicly as part of the NeuroX toolkit (Dalvi et al., 2019b).

Ethics and Broader Impact

For this study, we used existing publicly available data sets while following their terms in the licenses. We do not see any harm or ethical issues resulting from our study and findings. Our study has implications towards the work on interpreting and analyzing deep models.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 242–247, Valencia, Spain.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. *arXiv preprint arXiv:1608.04207*.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do Neural Machine Translation Models Learn about Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver. Association for Computational Linguistics.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2020. On the linguistic representational power of neural machine translation models. *Computational Linguistics*, 46(1):1–52.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating Layers of Representation in Neural Machine Translation on Part-of-Speech and Semantic Tagging Tasks. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, D. Anthony Bau, and James Glass. 2019a. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI, Oral presentation)*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and Improving Morphological Learning in the Neural Machine Translation Decoder. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*.

Fahim Dalvi, Avery Nortonsmith, D. Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, and James Glass. 2019b. NeuroX: A toolkit for analyzing individual neurons in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Honolulu, US.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Nadir Durrani, Fahim Dalvi, Hassan Sajjad, Yonatan Belinkov, and Preslav Nakov. 2019. One size does not fit all: Comparing NMT representations of different granularities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1504–1516, Minneapolis, Minnesota. Association for Computational Linguistics.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880, Online. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

Marius Mosbach, Anna Khokhlova, Michael A. Hedderich, and Dietrich Klakow. 2020. On the Interplay Between Fine-tuning and Sentence-level Probing for Linguistic Knowledge in Pre-trained Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2502–2516, Online. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Investigating Language Universal and Specific Properties in Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1478–1488, Berlin, Germany. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 1526–1534, Austin, TX, USA.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2016. Word Representation Models for Morphologically Rich Languages in Neural Machine Translation. *arXiv preprint arXiv:1606.04217*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5754–5764.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

A Appendices

A.1 Data and Representations

We used standard splits for training, development and test data for the 4 linguistic tasks (POS, SEM, Chunking) that we used to carry out our analysis on. The splits to preprocess the data are available through git repository[6] released with Liu et al. (2019a). See Table 3 for statistics. We obtained the understudied pre-trained models from the authors of the paper, through personal communication.

Task	Train	Dev	Test	Tags
POS	36557	1802	1963	44
SEM	36928	5301	10600	73
Chunking	8881	1843	2011	22

Table 3: Data statistics (number of sentences) on training, development and test sets using in the experiments and the number of tags to be predicted

A.2 Layer-wise Probing

Section 4.1 presented layer-wise probing results for POS and Chunking tagging. Figure 4 show results on Semantic tagging. We see a similar pattern across architectures as in Figure 1.

A.3 Neuron-wise Probing

Section 4.2 presented neuron-wise probing results for for Chunking tagging. Figure 2 show results on POS and SEM tagging. We see a similar pattern across architectures as in Figure 3. As the model is fine-tuned towards downstream, number of salient neurons towards a linguistic property, in the lower layers increase.

A.4 Top versus Bottom Neurons

In Section 4.2 we presented spread how information is more distributed and redundant in in the network as bottom neurons also preserved linguistic knowledge. On the contrary the linguistic information becomes more localized and less distributed post fine-tuning using accuracy of the bottom neurons. Tables 4 and 5 demonstrate the same pattern with respect to Chunking and Semantic tagging tasks, selecting 10% and 5% neurons respectively.

	BERT		RoBERTa		XLNet	
Tasks	Top	Bot.	Top	Bot.	Top	Bot.
Base	94.7	92.3	94.8	92.5	94.2	92.3
MRPC	94.4	91.9	94.4	89.1	93.8	72.4
QNLI	94.3	92.3	94.0	82.2	93.0	33.3
MNLI	93.8	91.3	93.2	82.1	92.8	44.5
RTE	94.7	92.2	94.3	88.2	94.0	84.7
SST	94.3	91.9	94.1	60.7	93.8	39.7
STS	94.8	92.3	94.3	79.7	92.2	83.8

Table 4: Chunking accuracy – Top vs. Bottom neurons

	BERT		RoBERTa		XLNet	
Tasks	Top	Bot.	Top	Bot.	Top	Bot.
Base	92.2	90.9	92.8	90.7	96.5	91.2
MRPC	92.2	90.6	91.5	88.1	92.3	72.3
QNLI	92.0	90.8	91.5	78.6	91.4	17.8
MNLI	91.9	90.3	91.4	79.0	91.3	43.0
RTE	92.0	90.6	91.5	86.9	91.3	80.0
SST	92.1	90.5	91.5	60.7	91.3	34.8
STS	92.1	90.4	91.5	79.5	91.6	83.7

Table 5: SEM accuracy – Top vs. Bottom neurons

A.5 Pruning Layers

In Section 5 we showed how pruning bottom layers in RoBERTa was more harmful in comparison to BERT. We conjectured that this pattern entails from our analysis that in RoBERTa linguistic information is preserved in the initial middle layers as opposed to BERT where linguistic knowledge is distributed deeper in the network. We show that XLNet exhibit similar pattern to RoBERTa in Table 6.

A.6 Control Tasks

While there is a plethora of work demonstrating that contextualized representations encode a continuous analogue of discrete linguistic information, a question has also been raised recently if the representations actually encode linguistic structure or whether the probe memorizes the understudied task. We use *Selectivity* as a criterion to put a "linguistic task's accuracy in context with the probe's capacity to memorize from word types" (Hewitt and Liang,

Tasks	**SST-2**	**MNLI**	**QNLI**
		XLNet	
Baseline	93.9	86.0	90.4
Prune Top 6	92.2	83.5	88.0
Prune Bottom 6	87.5	68.1	83.0

Table 6: Pruning Layers in the Models

---

[6] `https://github.com/nelson-liu/contextual-repr-analysis`

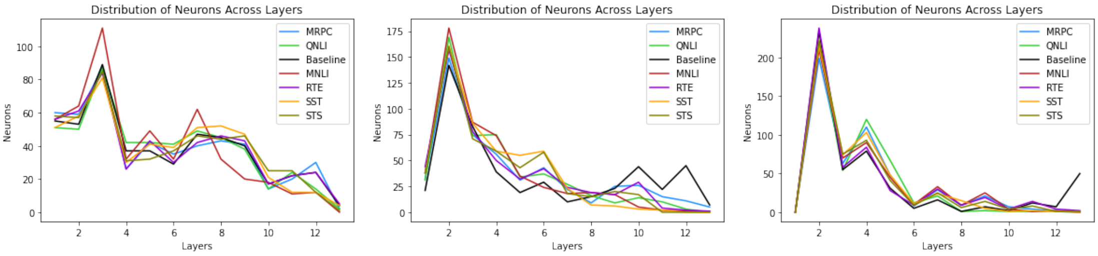

(a) BERT – POS (b) RoBERTa – POS (c) XLNet – POS

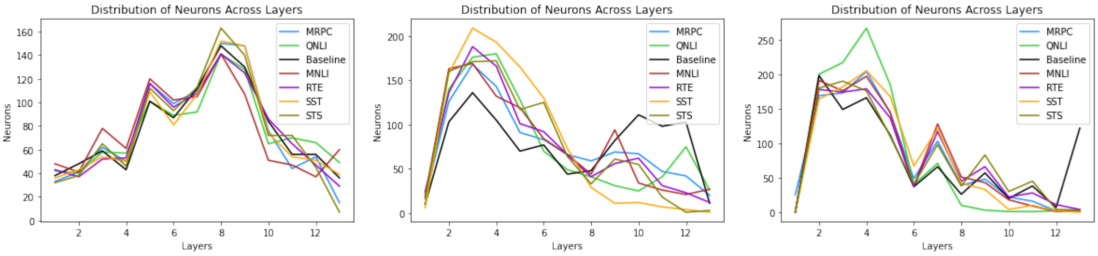

(d) BERT – Chunking (e) RoBERTa – Chunking (f) XLNet – Chunking

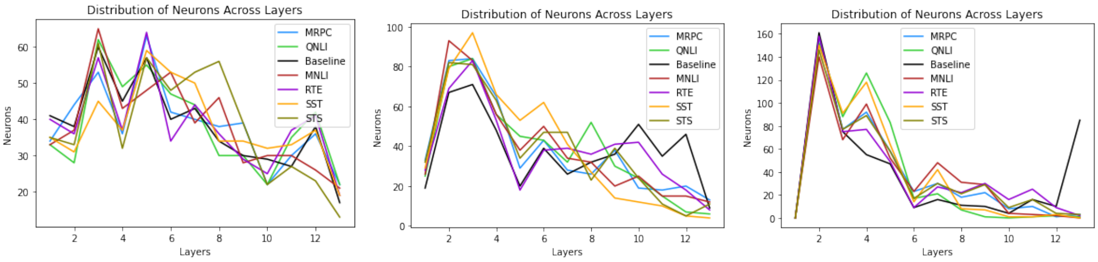

(g) BERT – SEM (h) RoBERTa – SEM (i) XLNet – SEM

Figure 3: Distribution of Top Neurons across Layers

2019). It is defined as the difference between linguistic task accuracy and control task accuracy. An effective probe is recommended to achieve high linguistic task accuracy and low control task accuracy.

A.7 Infrastructure and Run Time

Our experiments were run on NVidia GeForce GTX TITAN X GPU card. Grid search for finding optimal lambdas is expensive when optimal number of neurons for the task are unknown. Running grid search would take $\mathcal{O}(MN^2)$ where $M = 100$ (if we try increasing number of neurons in each step by 1%) and $N = 0, 0.1, \ldots 1e^{-7}$. We fix the $M = 20\%$ to find the best regularization parameters first reducing the grid search time to $\mathcal{O}(N^2)$ and find the optimal number of neurons in a subsequent step with $\mathcal{O}(M)$. The overall running time of our algorithm therefore is $\mathcal{O}(M + N^2)$. This varies a lot in terms of wall-clock computation, based on number of examples in the training data, number of tags to be predicted in the downstream task. Including a full forward pass over the pre-trained model to extract the contextualized vector, and running the grid search algorithm to find the best hyperparameters and minimal set of neurons took on average 8 hours ranging from 3 hours for the Chunking experiment to 12 hours for POS and SEM due to large training data.

A.8 Hyperparameters

We use elastic-net based regularization to control the trade-off between selecting focused individual neurons versus group of neurons while maintaining the original accuracy of the classifier without any regularization. We do a grid search on L_1 and L_2 ranging from values $0 \ldots 1e^{-7}$. See Table 8 for the optimal values for each task across different architectures.

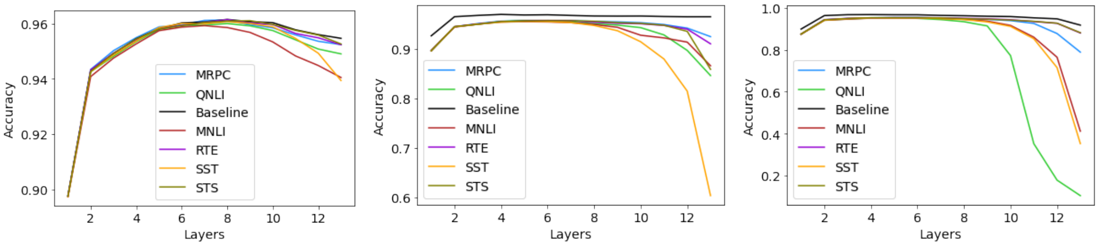

(a) BERT - SEM (b) RoBERTa - SEM (c) XLNet - SEM

Figure 4: Layer-wise Probing Performance

	BERT	XLNet	RoBERTa
Neu$_a$	9984	9984	9984
		POS	
Neu$_t$	500/5%	500/5%	500/5%
Acc$_a$	96.2	96.4	96.3
Acc$_t$	95.9	96.5	96.2
Sel$_a$	14.45	23.49	22.65
Sel$_t$	31.68	31.82	34.21
		SEM	
Neu$_t$	500/5%	500/5%	500/5%
Acc$_a$	92.51	92.29	92.95
Acc$_t$	92.32	92.62	92.97
Sel$_a$	5.77	14.03	13.76
Sel$_t$	27.17	26.55	24.53
		Chunking	
Neu$_t$	1000/10%	1000/10%	1000/10%
Acc$_a$	94.36	93.84	94.66
Acc$_t$	94.68	94.24	94.79
Sel$_a$	16.30	22.77	21.12
Sel$_t$	29.19	28.42	28.91

Table 7: Selecting minimal number of neurons for each downstream NLP task. Accuracy numbers reported on blind test-set (averaged over three runs) – Neu$_a$ = Total number of neurons, Neu$_t$ = Top selected neurons, Acc$_a$ = Accuracy using all neurons, Acc$_t$ = Accuracy using selected neurons after retraining the classifier using selected neurons, Sel = Difference between linguistic task and control task accuracy when classifier is trained on all neurons (Sel$_a$) and top neurons (Sel$_t$).

	BERT	XLNet	RoBERTa
		L1 , L2 = λ_1, λ_2	
POS	.001, .01	.001, .01	.001, .001
SEM	.001, .01	.001, .01	.001, .001
Chunk	$1e^{-4}, 1e^{-5}$	$1e^{-4}, 1e^{-4}$	.001, .001

Table 8: Best elastic-net lambdas parameters for each task

Language Models Use Monotonicity to Assess NPI Licensing

Jaap Jumelet[↑] **Milica Denić**[↑] **Jakub Szymanik**[↑]
Dieuwke Hupkes[λ] **Shane Steinert-Threlkeld**[↓]

[↑] Institute for Logic, Language and Computation, University of Amsterdam
[λ] Facebook AI Research
[↓] Department of Linguistics, University of Washington

`{j.w.d.jumelet, m.denic, J.K.Szymanik}@uva.nl` `dieuwkehupkes@fb.com` `shanest@uw.edu`

Abstract

We investigate the semantic knowledge of language models (LMs), focusing on (1) whether these LMs create categories of linguistic environments based on their semantic *monotonicity* properties, and (2) whether these categories play a similar role in LMs as in human language understanding, using negative polarity item licensing as a case study. We introduce a series of experiments consisting of probing with diagnostic classifiers (DCs), linguistic acceptability tasks, as well as a novel *DC ranking* method that tightly connects the probing results to the inner workings of the LM. By applying our experimental pipeline to LMs trained on various filtered corpora, we are able to gain stronger insights into the semantic generalizations that are acquired by these models.[1]

1 Introduction

Neural language models (LMs) have become powerful approximators of human language, making it increasingly important to understand the features and mechanisms underlying their behavior (Linzen et al., 2018, 2019). In the past few years, a substantial number of studies have investigated the linguistic capabilities of LMs (Gulordava et al., 2018; Giulianelli et al., 2018; Lakretz et al., 2019; Wu et al., 2020; Ettinger, 2020, i.a.). Such work has focused primarily on *syntactic* properties, while fewer studies have been done on what kind of *formal semantic* features are encoded by language models. In this paper, we focus explicitly on what LMs learn about a semantic property of sentences, and in what ways their knowledge reflects well-known features of human language processing.

As the topic of our studies, we consider **monotonicity**, a semantic property of linguistic environments that plays an important role in human language understanding and inference (Hoeksema, 1986; Valencia, 1991; Van Benthem, 1995; Icard III and Moss, 2014): the monotonicity of a linguistic environment determines whether inferences from a general to a particular term or vice versa are valid in that environment. For example, the fact that the inference from "*Mary didn't write a paper*" to "*Mary didn't write a linguistics paper*" is valid shows us that the position where "*a paper*" occurs is *downward monotone*: the inference is valid when a more general term ("*a paper*") is replaced with a more specific one ("*a linguistics paper*").

To investigate monotonicity we focus on **negative polarity items** (NPIs): a class of expressions such as *any* or *ever* that are solely acceptable in downward monotone environments (Fauconnier, 1975; Ladusaw, 1979). Psycholinguistic research has confirmed this connection between NPIs and monotonicity: humans judge NPIs acceptable in a linguistic environment if they consider that environment to be downward monotone (Chemla et al., 2011). Previous research has established that LMs are relatively successful in processing NPIs (Warstadt et al., 2019), but without investigating *how* they came to these successes.

We raise the following research questions:

RQ1 Do language models encode the monotonicity properties of linguistic environments?

RQ2 To what extent do they employ this information when processing negative polarity items?

We developed a series of experiments, in which we first evaluate the general capacities of LMs in handling monotonicity and NPIs and then investigate the generalization heuristics of the LM by doing experiments with modified training corpora. First, we establish that LMs are able to encode a notion of monotonicity by probing them with diagnostic classifiers (DCs, Hupkes et al., 2018) (§5.1).

---

[1] All code and data can be found at `https://github.com/jumelet/monotonicity-npi-lm`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4958–4969
August 1–6, 2021.

In our second experiment we demonstrate that our LMs are reasonably successful with NPI licensing using an NPI acceptability task (§5.2). Next, we introduce a novel *DC ranking* method to investigate the overlap between the information that the model uses to make judgments about NPIs and the information that the DCs use to predict monotonicity information, finding that there is a significant overlap (§5.3).

We then investigate two potential confounds that may obfuscate our results. First, we consider whether the signal that is picked up by the monotonicity DC is not simply a proxy that tells the model that an NPI may occur at that position (§5.4). To assess this, we train new LMs on a corpus from which all sentences with NPIs have been removed, re-run the montonicity probing task, and find that even in the absence of NPI information, LMs are still able to encode a notion of monotonicity.

Next, we consider whether an LM bases its NPI predictions on simple co-occurrence heuristics, or if it can extrapolate from a general notion of monotonicity to cases of NPIs in environments in which they have never been encountered during training (§5.5). We again train new LMs on modified corpora, this time removing NPIs only in one specific environment, and repeat the NPI acceptability and DC ranking experiments. The results of this setup demonstrate that LMs indeed use a general notion of monotonicity to predict NPI licensing.

Contributions With this work, we contribute to the ongoing study of the linguistic abilities of language models in several ways:

- With a series of experiments we demonstrate that LMs are able to acquire a general notion of monotonicity that is employed for NPI licensing.
- We present two novel experimental setups: *filtered corpus training* and *DC ranking*, that can be used to assess the impact of specific information during training and compare the information used by DCs with the information used with the model, respectively.
- By using experimental results from psychosemantics to motivate hypotheses for LM behavior, we find that our models reflect behavior similar to human language processing.

In the remainder of this paper, we will first provide some linguistic background that helps to situate and motivate our experiments and results (§2). We then discuss related work on NPI processing in LMs in §3. In §4, we discuss our methods and experimental setup. §5.1 through §5.5 explain and present the results. We conclude in §6 with a general discussion and pointers to future work.

2 Linguistic Background

Monotonicity Monotonicity is a property of a linguistic environment which determines what kind of inferences relating general and particular terms are valid in that environment. If inferences from a general to a particular term are valid, the linguistic environment is said to be *downward monotone* (DM). If inferences are valid the other way around, from a particular to a general term, the linguistic environment is said to be *upward monotone* (UM).

Examples of expressions inducing DM environments are negation and quantifiers like *nobody, no NP*, but also specific types of adverbs and the antecedents of conditional sentences. For instance, (1) below exemplifies that in these environments the inference from a sentence with a general term (*cookies*) to that sentence with a more particular term (*chocolate cookies*) is valid, but not vice versa.

(1) a. Mary did**n't** eat cookies. ⇒
Mary didn't eat *chocolate* cookies.
b. **Nobody** ate cookies. ⇒
Nobody ate *chocolate* cookies.
c. Mary **rarely** ate cookies. ⇒
Mary rarely ate *chocolate* cookies.

Common examples of UM environments are (non-quantified) positive sentences, quantifiers such as *somebody, many NP*, and other kind of adverbs. (2) exemplifies that in these environments the inference from a sentence with a more particular term (*chocolate cookies*) to the same sentence with a general term (*cookies*) is valid, but not vice versa.

(2) a. Mary ate *chocolate* cookies. ⇒
Mary ate cookies.
b. **Everyone** ate *chocolate* cookies. ⇒ Everyone ate cookies.
c. Mary **often** ate *chocolate* cookies. ⇒
Mary often ate cookies.

NPIs NPIs are expressions such as the English words *any, anyone, ever*, whose acceptability depends on whether its linguistic environment is downward monotone (Fauconnier, 1975; Ladusaw, 1979; Dowty, 1994; Kadmon and Landman, 1993; Krifka, 1995; Lahiri, 1998; Chierchia, 2006, 2013).[2] While the conditions for NPI acceptability are complex, a good approximation is that NPIs are acceptable (or *licensed*) in the syntactic scope of *NPI licensors* that induce a DM environment.[3] If we again consider the DM environment of (1-a) and the UM environment of (2-a), it can be seen that English *any* is an NPI, as it is acceptable when inside the syntactic scope of negation (a DM expression) as in (3-a), and not acceptable when they are in an UM environment as in (3-b).

(3) a. Mary didn't eat (any) cookies.
 b. Mary ate (*any) cookies.

Importantly, monotonicity plays a role at the psychological level: human judgments about the monotonicity of a linguistic environment predict their judgments of NPI acceptability in that environment (Chemla et al., 2011; Denić et al., 2021). For example, how plausible someone finds the inference (1-a) predicts how acceptable they find the sentence (3-a). Summing up, NPI licensing has a syntactic component (NPIs must reside in syntactic scope of a licensor) and a semantic component (NPI licensors are DM expressions), that are connected on a psychological level (monotonicity judgments predict NPI acceptability). Our research aims to uncover whether this connection is exhibited by LMs as well.

3 Related work

The literature on interpreting LMs has grown substantially in the last few years (see, e.g. Belinkov and Glass, 2019; Alishahi et al., 2019; Rogers et al., 2021, for survey papers). Several studies investigate how they process NPIs, focused mainly on the *syntactic* aspect of NPI licensing.

Jumelet and Hupkes (2018) conclude that LSTM language models encode information about the dependency between the NPI and the NPI licensor, although this effect diminishes as the distance between the NPI and its licensor grows. Marvin and Linzen (2018) study NPI judgments of LMs on minimally different sentence pairs (with the NPI licensor either in an appropriate syntactic configuration or not) and find that their models are unable to reliably assign higher probability to sentences in which NPIs are correctly licensed. The syntactic aspect of NPI licensing is also examined by Futrell et al. (2019), who demonstrate that LSTM LMs are susceptible to learning spurious licensing relationships, a finding that Warstadt and Bowman (2020) demonstrate to also hold for BERT (Devlin et al., 2019). Wilcox et al. (2019) investigate how explicit syntactic supervision of LMs affects their success with syntactic aspects of NPI licensing. The broad linguistic suites of Warstadt et al. (2020) and Hu et al. (2020) also contain a set of tasks related to NPI licensing, demonstrating that it is one of the most challenging tasks for LMs to handle. Weber et al. (2021) investigated the dynamics of NPI learning during training, and connected this to a multi-task learning paradigm, demonstrating that LMs are able to efficiently leverage information from related licensing environments.

Lastly, Warstadt et al. (2019) examine BERT's ability in determining NPI acceptability. They demonstrate that BERT has significant knowledge of the dependency between NPIs and their licensors, but that this success varies widely across different experimental methods. Our study builds on that of Warstadt et al. (2019). Although they demonstrate that BERT is generally successful with NPI licensing, their results do not reveal whether BERT has constructed a more general category of DM expressions that is independent of collocational cues, nor whether it has understood that this category matters for NPI licensing.

4 Methods

Before getting to the main experimental part of our work, we briefly discuss the training corpus, model architecture and evaluation corpus we consider.

Training Corpus The base training corpus we consider in our experiments is the corpus used by Gulordava et al. (2018). This corpus is a collection of sentences from Good and Featured English Wikipedia articles and consists of over 90M tokens. The vocabulary of the corpus consists of the 50.000 most frequent tokens in this corpus; less frequent tokens are mapped to a special `<unk>` token. We refer to the full training corpus type with the name Full, and to the LMs trained on this corpus as Full LMs. In addition to Full, we use multiple other corpora which are derived from Full by means of filtering. This will allow us to draw conclusions about specific generalization abilities and reliance on collocational cues of LMs; filtered corpora will

---

[2] See however Zwarts, 1995; Giannakidou, 1998; Barker, 2018 for different takes on NPI acceptability generalizations.

[3] An NPI occurs in the syntactic scope of a licensor if the licensor *c-commands* the NPI. An NPI licensor c-commands an NPI if the NPI is the licensor's sister node or one of its sister's descendants in a constituent tree (Reinhart, 1976).

Environment Class	Abbrev.	DM example	UM example
Adverbs	ADV	A lady **rarely** *ever* ...	*A lady sometimes *ever*...
Conditionals	COND	**If** the dancers see *any* ...	*While the dancers see *any*...
Determiner Negation	D-NEG	**No** teacher says that the students had practiced *at all*.	*Some teacher says that the students had practiced *at all*.
Sentential Negation	S-NEG	The dancer was **not** saying that the guy had profited *yet*.	*The dancer was really saying that the guy had profited *yet*.
Only	ONLY	**Only** the boys had *ever* ...	*Even the boys had *ever* ...
Quantifiers	QNT	**Every** senator who had *ever* ...	*Some senator who had *ever* ...
Embedded Questions	QUES	The patients wonder **whether** the lady admires *any* ...	*The patients say that the lady admires *any*...
Simple Questions	SMP-Q	Did the boy *ever* listen**?**	*The boy did ever listen.
Superlatives	SUP	A lady buys the **oldest** dish that the adult had *ever* ...	*A lady buys the old dish that the adult had *ever* ...

Table 1: The nine environment classes of Warstadt et al. (2019), with an example of a minimal DM/UM pair for each class taken from the corpus.

be introduced in the relevant sections.

Model Architecture In our studies, we focus on recurrent language models. More specifically, following Gulordava et al. (2018), we consider two-layer LSTM language models, with an embedding and hidden size of 650. All training runs across our experiments follow the same regime, identical to the regime described by Gulordava et al. (2018): 40 epochs of training with SGD, with a plateau scheduler and an initial learning rate of 20, a batch size of 64, BPTT length of 35, and dropout of 0.1.[4]

Evaluation Corpus To assess monotonicity and NPI licensing knowledge of LMs in our experiments, we leverage the NPI corpus of Warstadt et al. (2019), which consists of a large amount of grammatical and ungrammatical sentences with NPIs. This corpus is divided into 9 distinct **environment classes**, allowing for fine-grained analysis of NPI licensing. Importantly, these nine environment classes come in two versions: a DM version—in which NPIs are grammatically acceptable, and a minimally different UM version—in which they are not. We provide an overview with examples of DM and UM versions of all environment classes in Table 1. The full size of the corpus is 106.000 distinct DM sentences, and the division of environment classes is split roughly uniformly.

5 Experiments and Results

In this section we describe the experimental pipeline in more detail. A graphical overview of our experiments is depicted in Figures 1 and 4. Each experiment description is directly followed by an analysis of its results.

5.1 Experiment 1: Do LMs represent monotonicity information?

In our first experiment, we test whether LMs trained on our Full corpus possess a notion of monotonicity. We train five different LMs and test how well they represent monotonicity properties of different environments by training linear **diagnostic classifiers** (DCs, Hupkes et al., 2018) on top of the hidden states of the LM. To create a corpus of monotonicity sentences for training and testing the DCs, we leverage the corpus of Warstadt et al. (2019), now selecting all DM and UM sentences to build up a balanced corpus of these categories. The nine environment classes in that corpus hence provide a broad spectrum of DM environments and their minimally different UM counterparts.

For training and testing the DCs, we consider the hidden states at the position directly before an NPI occurs (see Figure 1). The reason we train the DCs at this position is because only at this point we are sure that the monotonicity information should surface and be encoded *linearly*. This is due to the fact that the decoder of the LM that transforms a hidden state into a probability distribution is linear as well: if the probability of some token depends on a linguistic feature, this feature must hence be encoded linearly. The DCs are implemented using the `diagNNose` library of Jumelet (2020), and trained using 10-fold cross-validation, Adam optimization (Kingma and Ba, 2015), a learning rate of 10^{-2} and L1 regularization with $\lambda = 0.005$.

We train our monotonicity DCs in two separate ways. First, we divide the entire monotonicity corpus into a 90/10 train/test split, sampled *uniformly* across the different environment classes. This allows us to examine whether DM and UM environments are linearly separable in a way that is applicable to all environment classes. We refer to this classifier as the All-ENV DC.

Second, we move to a more fine-grained type

---

[4] Models are trained on a *GeForce 1080 Ti* GPU, take around 40 hours to train, and consist of 71M parameters.

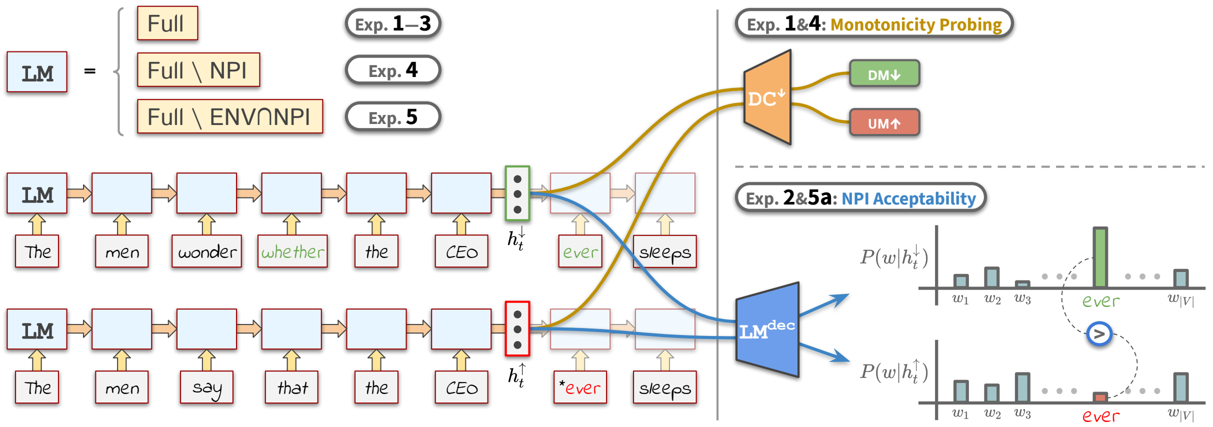

Figure 1: The pipeline of our experimental setup. We start by computing the hidden states $h_t^\downarrow$ (within a DM environment ahead of the NPI) and $h_t^\uparrow$ (within a UM environment). These hidden states are then used for training the monotonicity DC (Exp. 1 & 4), and to compare $P_{\text{LM}}(\text{NPI}|h_t^\downarrow) > P_{\text{LM}}(\text{NPI}|h_t^\uparrow)$ (Exp. 2 & 5a). The task of Experiments 3 and 5a can be found in Figure 4. Experiments 4 and 5 consist of the same tasks as the first three experiments, but differ in the language model that is used.

of analysis. High performance of the All-ENV DC namely does not provide evidence that monotonicity is encoded the same way for each environment: the set of salient hidden units used by the All-ENV DC for classifying monotonicity within the *Adverbs* environment, for example, could be disjoint from the set of units used for the *Only* environment. To investigate this, we train a DC on the hidden states of all-but-one environment class, and test its performance on the excluded class. This provides a measure to what extent the monotonicity representation of DM and UM environments derived from all other environment classes *generalizes* to the held-out class, demonstrating stronger evidence that the model represent monotonicity in the same way across different environments.

Results The results of our first experiment are shown in the top row of Figure 2. The first column contains the average accuracy for the All-ENV DC, and it can be seen that the diagnostic classifier succeeds in this task with high accuracy (97%). This indicates that the uniform split over all environment classes is linearly separable.

Next, we consider the held-out evaluation procedure for each of the nine environment classes. It can be seen that the monotonicity signal generalizes well to five classes (*adverbs, determiner negation, only, sentential negation, and embedded questions*), all with an accuracy above 90%. The other four classes yield a higher standard deviation, indicating that these classes are encoded less consistently across initialization seeds. The accuracy for all held-out DCs is lower compared to the All-ENV DC results, indicating that the All-ENV DC relied partly on information unrelated to a shared notion of monotonicity. The fact that the accuracy of these DCs is still so high, however, indicates that there is a substantial overlap between the way that monotonicity is encoded within the different environments.

5.2 Experiment 2: Do LMs predict the licensing conditions of NPIs?

In the next experiment we investigate the NPI acceptability judgments of the Full LMs on the corpus of Warstadt et al. (2019). This is done by comparing the probability of an NPI conditioned on the model's representation of a DM environment ($h_t^\downarrow$) and a UM environment ($h_t^\uparrow$), where success is defined as follows:

$$P_{\text{LM}}(\text{NPI}|h_t^\downarrow) > P_{\text{LM}}(\text{NPI}|h_t^\uparrow)$$

This is a common evaluation procedure in the interpretability literature (Linzen et al., 2016), and has earlier been applied in the domain of NPI licensing by Jumelet and Hupkes (2018) and Warstadt et al. (2020). Our approach is similar to the Cloze Test of Warstadt et al. (2019), but their setup used (bi-directional) masked LMs, making it possible to directly compare the probabilities of the NPI licensor, instead of comparing the NPI probabilities. Note that we purposefully do not base NPI acceptability on comparing full sentence probabilities: in our view this type of comparison can be distracted by token probabilities not related to the NPI itself.

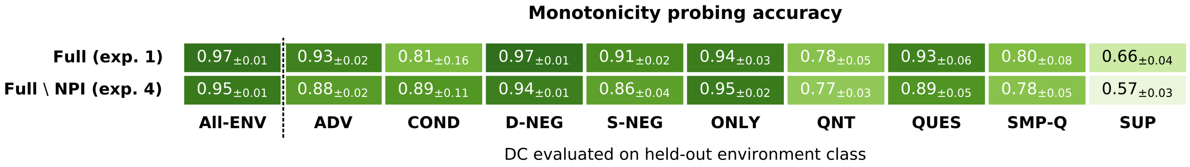

Monotonicity probing accuracy

	All-ENV	ADV	COND	D-NEG	S-NEG	ONLY	QNT	QUES	SMP-Q	SUP
Full (exp. 1)	$0.97_{\pm 0.01}$	$0.93_{\pm 0.02}$	$0.81_{\pm 0.16}$	$0.97_{\pm 0.01}$	$0.91_{\pm 0.02}$	$0.94_{\pm 0.03}$	$0.78_{\pm 0.05}$	$0.93_{\pm 0.06}$	$0.80_{\pm 0.08}$	$0.66_{\pm 0.04}$
Full \ NPI (exp. 4)	$0.95_{\pm 0.01}$	$0.88_{\pm 0.02}$	$0.89_{\pm 0.11}$	$0.94_{\pm 0.01}$	$0.86_{\pm 0.04}$	$0.95_{\pm 0.02}$	$0.77_{\pm 0.03}$	$0.89_{\pm 0.05}$	$0.78_{\pm 0.05}$	$0.57_{\pm 0.03}$

DC evaluated on held-out environment class

Figure 2: Accuracy and standard deviation on the monotonicity diagnostic classification task, averaged over 5 seeds for each model type. The All-ENV column denotes train/test split procedure sampled uniformly over all environment class; other columns denote accuracy on one environment class that has been excluded during training.

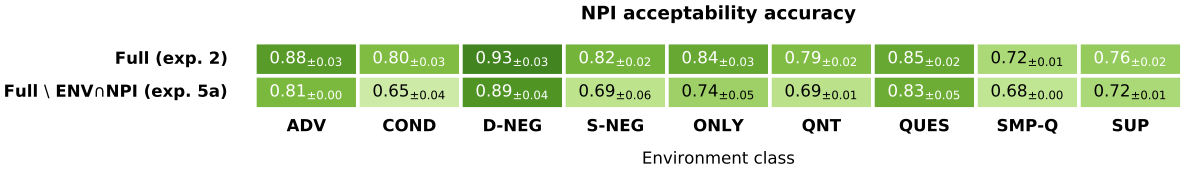

NPI acceptability accuracy

	ADV	COND	D-NEG	S-NEG	ONLY	QNT	QUES	SMP-Q	SUP
Full (exp. 2)	$0.88_{\pm 0.03}$	$0.80_{\pm 0.03}$	$0.93_{\pm 0.03}$	$0.82_{\pm 0.02}$	$0.84_{\pm 0.03}$	$0.79_{\pm 0.02}$	$0.85_{\pm 0.02}$	$0.72_{\pm 0.01}$	$0.76_{\pm 0.02}$
Full \ ENV∩NPI (exp. 5a)	$0.81_{\pm 0.00}$	$0.65_{\pm 0.04}$	$0.89_{\pm 0.04}$	$0.69_{\pm 0.06}$	$0.74_{\pm 0.05}$	$0.69_{\pm 0.01}$	$0.83_{\pm 0.05}$	$0.68_{\pm 0.00}$	$0.72_{\pm 0.01}$

Environment class

Figure 3: Accuracy on the NPI acceptability task—based on whether the NPI was assigned a higher probability in the DM environment than in its UM counterpart.

We split this procedure out for each of the nine environment classes. The example sentence of the Simple Questions environment in Table 1, for example, is evaluated as follows:

$$P_{\text{LM}}(ever|Did\ the\ boy) > P_{\text{LM}}(ever|The\ boy\ did)$$

Using the full sentence probabilities for this comparison would require taking probabilities into account such as $P_{\text{LM}}(the|Did)$ and $P_{\text{LM}}(boy|The)$, that have no relation to NPI licensing at all.

Results We present the results for this experiment in the top row of Figure 3. The Full models demonstrate a considerable ability at predicting NPI acceptability, with the least performing class (SMP-Q, *Simple Questions*) yielding an accuracy that is still well above chance (0.72). Compared to earlier investigations on the ability of LSTM LMs in NPI licensing, our results indicate that these models are able to obtain a more sophisticated understanding of NPIs than previously thought: both Marvin and Linzen (2018) and Hu et al. (2020) report LSTM performance below chance on NPI acceptability tasks. This might in part be due to the different evaluation procedure we used (conditional vs. full-sentence probability comparison).

5.3 Experiment 3: Is the LM's knowledge of DM environments and of NPI licensing related?

We have now established that our models encode a signal related to monotonicity, and are successful at predicting NPI acceptability. In our third experiment, we assess to what extent the parameters used by the LM to predict NPIs (i.e. the LM's decoder embeddings for NPIs) *overlap* with the information the DCs use to predict the monotonicity properties of a particular environment class. For this we have devised a novel **DC ranking** method, that ranks the LM's decoder weights for all tokens based on their similarity with the DC weights.

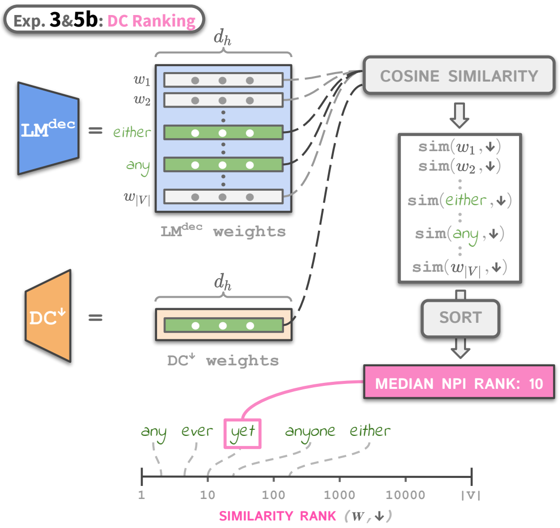

Figure 4: The *DC Ranking* experiment, in which we investigate whether the monotonicity DC and the LM decoder base their predictions on similar cues, by computing and ranking the cosine similarities between the DC weights and the decoder weights of each token.

We present a schematic overview of the method

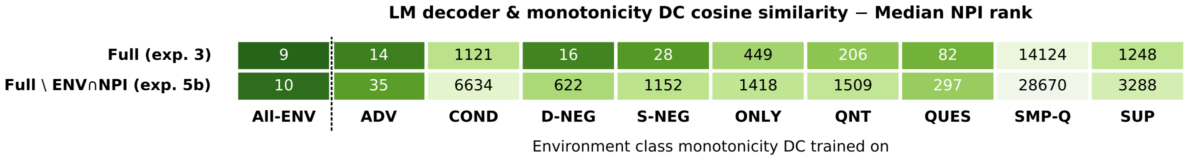

	All-ENV	ADV	COND	D-NEG	S-NEG	ONLY	QNT	QUES	SMP-Q	SUP
Full (exp. 3)	9	14	1121	16	28	449	206	82	14124	1248
Full \ ENV∩NPI (exp. 5b)	10	35	6634	622	1152	1418	1509	297	28670	3288

Figure 5: Results on the median NPI rank task. A low median rank indicates that the monotonicity DC uses the same representational information as the NPI decoder.

in Figure 4. The LM's decoder weight matrix can be interpreted as a collection of vectors corresponding to each token in the model's vocabulary. The monotonicity DC is a binary classifier, so its weights are represented by a single vector. The LM's decoder vectors are of the same dimensionality as the weight vector of the monotonicity DC, which allows us to compute the similarity between each decoder vector and the monotonicity DC. For each of the 50.000 tokens in the LM's vocabulary, we calculate the cosine similarity between the decoder weights corresponding to that token and the DC's weights. We then sort these similarity scores, which results in a ranking of tokens that are most similar to the DC.

As we are interested in finding the connection of the monotonicity DC and the LM's NPI processing in general, we compute the **median rank** over a set of 11 NPIs.[5] A low median NPI rank indicates that the LM uses the same cues for NPI prediction as the monotonicity DC, demonstrating a clear connection between NPI licensing and monotonicity.

Contrary to Experiment 1, we no longer make use of the hold-one-out training procedure, that gave insights to what extent a general monotonicity signal generalizes to a held-out environment class. Instead, we train a separate diagnostic classifier for each environment class using a train/test split made up of DM and UM environments within that class. This results in a classifier that represents the class-specific decision boundary between minimal pairs of DM and UM items and allows us to investigate to what extent these decision boundaries align with the weights of the LM decoder. Next to the environment-specific DCs we also report the DC ranking outcome for the All-ENV DC that has been trained on all environments.

Results The results of this experiment are presented in the top row of Figure 5. The first column (All-ENV) contains the result for the DC trained on all environment classes, and the median NPI rank of 9 demonstrates that the monotonicity DC aligns very closely with the NPI decoder weights of the LMs. This median rank should be interpreted within the context of the model vocabulary size: it can range upwards to 50.000, so a rank that is close to 0 signifies a tight connection between the probing task and the tokens of interest.

Moving on to the environment-specific results, it can be seen that the results vary considerably between the environment classes. The worst scoring class is again that of Simple Questions. This makes sense, as the licensing conditions for question constructions do not depend on the presence of a specific licensing token such as *not*, but on the overall structure of the whole sentence. The other environment classes lead to scores far closer to 0, indicating that for these classes monotonicity classification is closely aligned to NPI processing.

Interestingly, the median rank of the All-ENV DC is lower than the ranks of all other DCs. This shows that the model has aligned its representation of NPIs to an aggregate of the monotonicity representations in the different environment classes. This allows the model to flexibly deal with NPIs in a wide range of licensing environments.

5.4 Experiment 4: Are NPIs important for learning monotonicity information?

With Experiment 3 we established that NPI processing and monotonicity are related in our LMs. Now, we investigate to what extent their representations are entangled during training. More specifically, we investigate if the signal from the presence of NPIs is indispensable for the LM to develop a notion of monotonicity, or if instead the success in categorizing monotonicity environments can be learned independently of NPIs.

We address this question by testing whether LMs

---

[5] We consider the following 11 single-token NPI expressions: *dared, any, anybody, anymore, anyone, anything, anywhere, ever, nor, whatsoever,* and *yet*. These are all the single-token NPIs taken from a list of NPIs that is described in §5.4.

can still classify the monotonicity properties of environments when they are completely deprived of NPIs during training. To do so, we train new language models on a modified corpus that does not contain any NPIs at all. To arrive at this corpus, we remove all sentences that contain at least one NPI expression from the Full corpus. We identify these expressions based on a comprehensive list of NPI expressions in English collected by Hoeksema (2012) and the list of NPIs in English compiled by Israel (2011). From this list, we manually removed expressions that have both NPI and non-NPI uses (e.g. *a thing, a bit*). The 40 NPI expressions that resulted from this procedure can be found in Appendix A. We train 5 models on this corpus and refer to them by the name Full\NPI.

In this experiment, we run the monotonicity probing procedure of Experiment 1 on the Full\NPI models. We posit that if the notion of monotonicity can be learned independently of NPIs, there should be no significant drop in performance compared to the results of the Full LMs.

Results We report the results of this experiment in the bottom row of Figure 2. Again it can be noted that the All-ENV DC, trained and tested uniformly over all environment classes, obtains a high accuracy on the task (0.95). Furthermore, none of the held-out environment DCs lead to significant drops in performance compared to the Full LMs. Based on this we conclude that even in the absence of NPI cues, LMs are still able to build up a shared robust notion of monotonicity.

5.5 Experiment 5: How robust is the connection between monotonicity and NPI processing?

This research aims to uncover whether LMs possess a robust connection between monotonicity and NPI licensing. Our findings indicate that this connection is present in our models. A major confound that has not yet been addressed, however, is the extent to which our models rely on collocational cues when judging the acceptability of an NPI. To test this, we examine whether an LM's connection between NPIs and monotonicity *generalizes* to novel environment classes in which NPIs have never been encountered during the training phase of the LM.

We have created nine modified corpora in which sentences with NPIs within a specific environment have been removed. For these different corpora, we again consider the nine NPI-licensing environments of Warstadt et al. (2019). For each environment class we create a new corpus by removing all sentences from the Full corpus in which an NPI expression from Appendix A is preceded by an expression belonging to that class, somewhere earlier in the sentence.[6] Note that we only remove the sentences in which the environment actually licenses an NPI; sentences in which the environment occurs without an NPI are retained. So for the *adverbs* environment, for example, we remove sentences like "*Mary rarely ate any cookies*" but not "*Mary rarely ate cookies*". For each of these nine corpora we train 3 new LMs. Models trained on these corpora are referred to by the name Full\ENV∩NPI.

We run the NPI acceptability task of Experiment 2 and the DC ranking method of Experiment 3 on the nine types of Full\ENV∩NPI models. A model with a robust connection between monotonicity and NPI processing should be able to learn for NPIs in the held-out environment that (i) the environment belongs to the class of environments in which NPIs are licensed, and that (ii) determining NPI acceptance should be done based on representational cues that are similar for monotonicity prediction.

Results We report the results of this experiment next to the previous results of the Full model. First, we consider the NPI acceptability task, which is reported in the bottom row of Figure 3. Note that each cell in this row now corresponds to a specific model type: the ADV result, for instance, corresponds to the accuracy of the Full\ENV∩NPI models in which sentences with NPIs within adverbial environments have been removed. Our results show that the performance drops slightly for all environment classes, which can be attributed to a model's dependence on collocational cues. However, the models are still able to adequately generalize from the other environments, in which NPIs still are encountered, to the held-out environment. This demonstrates the semantic generalization capacities of the LM: it infers that the held-out environment in which NPIs have never been encountered shares some relevant properties with the other eight environment classes in which NPIs still occur.

The results for the DC ranking experiment are shown in the bottom row of Figure 5. Similar to the NPI acceptability results, the performance of the Full\ENV∩NPI models has dropped slightly compared to the Full models. However, if the models

---

[6] For Simple Questions we remove a sentence if an NPI occurs in a sentence that ends with a question mark.

would no longer pick up on the connection between monotonicity in the held-out environment and NPI licensing at all, these median ranks should drop to chance, i.e. around the halfway mark of the vocabulary size (25.000). It can be seen that this is only the case for the Simple Questions environment, that was already performing poorly for the Full models. Based on this we conclude that although models depend partly on collocational cues for their connection between monotonicity and NPIs, they are still able to encode a robust connection that generalizes to novel DM environments.

6 Conclusion

Based on a series of experiments, we have established the following: (1) LMs categorize environments into DM and UM; (2) LMs are overall successful with NPI licensing; (3) LMs employ similar representational cues when processing NPIs and predicting monotonicity; (4) their categories of DM and UM environments can be learned independently of NPI occurrence; and (5) their connection between monotonicity and NPI processing is robust and not solely dependent on co-occurrence heuristics. This demonstrates that LMs have quite sophisticated knowledge of NPI licensing, which may be similar to that of humans and constitutes a vital step towards better understanding the linguistic generalization capacities of LMs.

These results raise the question: what do LMs learn about the DM and UM environments which they succeed in finding? Do they actually learn the inferential properties of those environments, or do they rely on some other property that DM environments have in common to categorize them as such? A direction for future work would be to develop methods to probe the inferential capacities of LMs and explore how they align with the DM and UM categories they construct.

Another direction for future work would be to incorporate the recent advancements on probing-based interpretability methods in our experimental pipeline (Hewitt and Liang, 2019; Voita and Titov, 2020). Our DC Ranking method aligns the performance of a probe with that of the language model itself, which is related to the approaches of Saphra and Lopez (2019), Elazar et al. (2021), and Lovering et al. (2021). Placing our methodology more firmly in this body of work will allow for stronger conclusions to be drawn regarding the semantic knowledge of current language models.

Acknowledgments

We thank Oskar van der Wal and Lucas Weber for their valuable feedback. We thank Jack Hoeksema for providing us with the list of NPIs. MD and JS were funded by the European Research Council under the European Unions Seventh Framework Programme (FP/20072013)/ERC Grant Agreement n. STG 716230 CoSaQ.

References

Afra Alishahi, Grzegorz Chrupała, and Tal Linzen. 2019. Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop. *Natural Language Engineering*, 25(4):543–557.

Chris Barker. 2018. Negative polarity as scope marking. *Linguistics and Philosophy*, 41(5):483–510.

Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Emmanuel Chemla, Vincent Homer, and Daniel Rothschild. 2011. Modularity and intuitions in formal semantics: The case of polarity items. *Linguistics and Philosophy*, 34(6):537–570.

Gennaro Chierchia. 2006. Broaden your views: Implicatures of domain widening and the “logicality” of language. *Linguistic inquiry*, 37(4):535–590.

Gennaro Chierchia. 2013. *Logic in Grammar: Polarity, Free Choice, and Intervention*. Oxford Studies in Semantics and Pragmatics 2. OUP Oxford.

Milica Denić, Vincent Homer, Daniel Rothschild, and Emmanuel Chemla. 2021. The influence of polarity items on inferential judgments. *To appear in Cognition.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

David Dowty. 1994. The role of negative polarity and concord marking in natural language reasoning. In *Semantics and Linguistic Theory*, volume 4, pages 114–144.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Trans. Assoc. Comput. Linguistics*, 9:160–175.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Gilles Fauconnier. 1975. Polarity and the scale principle. *Chicago Linguistics Society*, 11:188–199.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Anastasia Giannakidou. 1998. *Polarity sensitivity as (non) veridical dependency*, volume 23. John Benjamins Publishing.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743.

Jack Hoeksema. 1986. Monotonicity phenomena in natural language. *Linguistic Analysis*, 16(1–2):235–250.

Jack Hoeksema. 2012. On the natural history of negative polarity items. *Linguistic Analysis*, 38(1):3–33.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'Diagnostic Classifiers' Reveal how Recurrent and Recursive Neural Networks Process Hierarchical Structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Thomas Icard III and Lawrence Moss. 2014. Recent progress in monotonicity. *LiLT (Linguistic Issues in Language Technology)*, 9.

Michael Israel. 2011. *The grammar of polarity: Pragmatics, sensitivity, and the logic of scales*, volume 127. Cambridge University Press.

Jaap Jumelet. 2020. diagNNose: A library for neural activation analysis. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 342–350, Online. Association for Computational Linguistics.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do Language Models Understand Anything? On the Ability of LSTMs to Understand Negative Polarity Items. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nirit Kadmon and Fred Landman. 1993. Any. *Linguistics and philosophy*, 16(4):353–422.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Manfred Krifka. 1995. The semantics and pragmatics of polarity items. *Linguistic analysis*, 25(3-4):209–257.

William Ladusaw. 1979. *Polarity Sensitivity as Inherent Scope Relations*. Ph.D. thesis, University of Texas Austin.

Utpal Lahiri. 1998. Focus and negative polarity in Hindi. *Natural Language Semantics*, 6:57–123.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Tal Linzen, Grzegorz Chrupała, and Afra Alishahi. 2018. Proceedings of the 2018 emnlp workshop blackboxnlp: Analyzing and interpreting neural networks for nlp. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors. 2019. *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Florence, Italy.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. 2021. Predicting inductive biases of pre-trained models. In *International Conference of Learning Representations (ICLR)*.

Rebecca Marvin and Tal Linzen. 2018. Targeted Syntactic Evaluation of Language Models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. Reinhart. 1976. *The Syntactic Domain of Anaphora*. MIT Linguistics Dissertations. Massachusetts Institute of Technology.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with svcca. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267.

Víctor Manuel Sánchez Valencia. 1991. *Studies on natural logic and categorial grammar*. Universiteit van Amsterdam.

Johan Van Benthem. 1995. *Language in Action: categories, lambdas and dynamic logic*. MIT Press.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196.

Alex Warstadt and Samuel R Bowman. 2020. Can neural networks acquire a structural bias from raw linguistic data? In *Proceedings of the 42th annual conference of the Cognitive Science Society*.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019. Investigating BERT's knowledge of language: Five analysis methods with NPIs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2877–2887, Hong Kong, China. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Trans. Assoc. Comput. Linguistics*, 8:377–392.

Lucas Weber, Jaap Jumelet, Elia Bruni, and Dieuwke Hupkes. 2021. Language modelling as a multi-task problem. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 2049–2060. Association for Computational Linguistics.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural Supervision Improves Learning of Non-Local Grammatical Dependencies. In *Proceedings of North American Association for Computational Linguistics (NAACL)*, pages 3302–3312.

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2020. Infusing finetuning with semantic dependencies. *CoRR*, abs/2012.05395.

Frans Zwarts. 1995. Nonveridical contexts. *Linguistic Analysis*, 25:286–312.

A Filtered NPIs

We here present the full list of NPIs that were used for filtering sentences from the Full corpus, resulting in the Full\NPI corpus. The method for selecting these expressions is described in Section 5.4.

A damn, any, any longer, any old, anybody, anymore, anyone, anything, anything like, anytime soon, anywhere, anywhere near, as yet, at all, avail, by much, can possibly, could possibly, ever, in any, in days, in decades, in minutes, in years, just any, just yet, let alone, much help, nor, or anything, set foot, squat, such thing, that many, that much, that often, the slightest, whatever, whatsoever, yet.

This resulted in a reduction of 75.062 sentences out of the 3.052.726 sentences in the original Full corpus (2.46%).

Slot Transferability for Cross-domain Slot Filling

[1]Hengtong Lu, [1]Zhuoxin Han, [1]Caixia Yuan, [1]Xiaojie Wang,
[2]Shuyu Lei, [2]Huixing Jiang, [2]Wei wu
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]Meituan-Dianping Group, Beijing, China
{luhengtong, hanzhuoxin, yuancx, xjwang}@bupt.edu.cn
{leishuyu, jianghuixing, wuwei30}@meituan.com

Abstract

Cross-domain slot filling focuses on using labeled data from source domains to train a slot filling model for target domains. It is of great significance for transferring a dialogue system into new domains. Most of the existing work focused on building a cross-domain transfer model. From the perspective of slots themselves, this paper proposes a model-agnostic Slot Transferability Measure (STM) for evaluating the transferability from a source slot to a target slot, specifically, the degree that labeled data of the source slot is helpful to train the slot filling model for the target slot. We also give a STM-based method for a model to select helpful source slots and their labeled data for a given target slot. Experimental results on multiple existing models and datasets show that our method significantly outperforms state-of-the-art baselines in cross-domain slot filling. The code is available at `https://github.com/luhengtong/STM-for-cdsf.git`.

1 Introduction

As an important task in task-oriented dialog systems, slot filling aims to identify task-related slot information in user utterances. When a task (or domain) has a large amount of labeled data, most existing slot filling models can achieve desired performance. However, there is usually little or even no labeled data for a new task. How to train the slot filling model in the new task (target task) with the labeled data of one or more existing tasks (source tasks) is of great significance for the rapid expansion of the application of task-oriented dialog systems.

Existing work can be mainly classified into two categories. The first is to establish implicit semantic alignment between slot representations of the source task and the target task, the model trained with the source task data is directly used for the target task (Bapna et al., 2017; Lee and Jha, 2019; Shah et al., 2019). The second is to use a two-stage strategy (Liu et al., 2020), which treats all slot values as entities. First, it trains a generic entity recognition model using source task labeled data to identify all candidate slot values in the target task. Then, the candidate slot value is classified into the target task slot by comparing the similarity between its representation and the target task slot information.

Most of the existing work has focused on building cross-task transferable models that leverage the association information between source tasks and target tasks, and the model is always trained using the labeled data of all the source tasks without distinction. However, not all the source task data will have transferable value to the target task, and the value of different source tasks data to a particular target task may be different. For example, flight-ticket-reservation task and train-ticket-reservation task have high similarity so that the labeled data of the former will be helpful to the latter. While the flight-ticket-reservation task and the weather-inquiry task have great difference so that the labeled data of the former has no or only little value to the latter, and even has a negative effect on the target model. Furthermore, even though the source task is similar to the target task, not every source slot will be useful for all the slots of the target task. For example, the labeled data for leaving-time slot in flight-ticket-reservation task may be helpful for the slot filling of leaving-time in train-ticket-reservation task, but not useful for the train-type slot. Therefore, finding valuable source slots that can provide transferable information for slot filling in target slot and then training a model based on the labeled data of these slots can make better use of the data in source tasks. This is the starting point of this paper which is different from the existing work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4970–4979
August 1–6, 2021.

In achieving this goal, we firstly propose slot transferability measure (STM) and give a method to calculate the STM. By comparing the STM between the target slot and each source slot, we can select different set of source slots for different target slot. Only the labeled data of these source slots are used to train the slot filling model for the target slot. To be more specific, we fuses distribution similarity of the slot value representations and of the slot value context representations between target slot and source slot as STM between two slots. All source slots are sorted according to their STMs with the target slot. Labeled data of the source slot with the highest STM are used to train the model, and then the labeled data of the source slot with the second highest STM is added to train the model. The process continues until the model gains no improvement on validation set of target slots. Those source slots and their labeled data are used to build the final slot filling model for the target slots.

Our main contributions are three-fold as follows.

1. We propose a metric called STM to measure the transferability between two slots. To our best knowledge, it is the first study on the transferability between two slots. The STM is model-agnostic.

2. We also propose a STM-based method to select source slots and their labeled data for training slot filling model for target slots.

3. Experimental results on several existing models and datasets show that this method brings consistent performance improvement for cross-domain slot filling.

2 Related work

As a key component of dialog system, the slot filling task has been studied extensively. Traditional supervised learning methods have made great achievements with a large amount of labeled data (Liu and Lane; Mesnil et al., 2015; Hakkani-Tür et al., 2016; Kurata et al., 2016; Liu and Lane, 2016; Goo et al., 2018; E et al., 2019). However, there is little or even no labeled data for a new task, the cross-domain slot filling task which uses labeled data in source tasks to training model for target task is gaining increasing attention (Yazdani and Henderson, 2015; Bapna et al., 2017; Zhu and Yu, 2018; Lee and Jha, 2019; Shah et al., 2019; Liu et al., 2020; Zhu et al., 2020). There are mainly two streams of methods in previous work.

The first is to establish implicit semantic alignment of the slot representations between the source task and the target task (Bapna et al., 2017; Lee and Jha, 2019; Shah et al., 2019; Liu et al., 2020). Bapna et al. (2017) proposed the Concept Tagging model (CT), which unified the slot filling model on the source tasks and the target task by combining the slot representations modeled by slot description information, and then conducting BIO 3-way classification. Based on CT, Lee and Jha (2019) proposed the Zero-Shot Adaptive Transfer model (ZAT), which introduced an attention layer in building representations of slot description; Meanwhile, Shah et al. (2019) proposed the Robust Zero-shot Tagger (RZT) model, which used a small number of sample slot values of the target slot to constrain the slot filler to avoid the negative transfer caused by the misalignment of slot names.

The second is a coarse-to-fine approach, which first identifies all candidates of slots and then classifies them into corresponding slots. Liu et al. (2020) proposed a Coarse-to-fine approach (Coach). They first predicted whether the tokens are slot value candidates, and then identified their specific slot types based on the similarity between the tokens and the representation of each slot description. In addition, Coach utilized a template regularization method which clusters the representations of semantically similar utterance into a similar vector space. It greatly improves the robustness of the model.

Most of these efforts focus on building a cross-task transferable model by exploiting the correlation information between source and target tasks. All source data is used to train the transfer model no matter if the data is helpful for target slot filling. On the contrast, this paper proposes a new method to select parts of source slots and their labeled data for model training.

3 Methodology

This section describes the cross-domain slot filling method proposed in this paper. First, we propose the concept of slot transferability and its measurement STM in Section 3.1. Then we describe the method of finding source slots for target slot based on the STM in Section 3.2. Finally, we introduce how this method can be deployed and implemented on existing models in Section 3.3. The STM is model-agnostic and will be validated on multiple existing models.

3.1 Slot Transferability Measure

Given slots s_a and s_b, the transferability from s_a to s_b refers to the degree of that the slot filling information of s_a can be used for slot filling of s_b, denoted as $\mathrm{STM}(s_a, s_b)$.

Let $p_v(s_i)$ be the distribution of slot value representation of slot $s_i(i = \{a, b\})$, $p_c(s_i)$ be the distribution of slot value context representation of slot $s_i(i = \{a, b\})$. We define the transferability from slot s_a to slot s_b as Equation (1):

$$\mathrm{STM}_\beta(s_a, s_b) = 1 - \tanh\left(\frac{(1+\beta^2)sim(p_v(s_a), p_v(s_b))sim(p_c(s_a), p_c(s_b))}{\beta^2 sim(p_v(s_a), p_v(s_b)) + sim(p_c(s_a), p_c(s_b))}\right) \tag{1}$$

where $sim(p, q)$ denotes the similarity between distribution p and q. The β parameter determines the weight of similarity between distributions of slot value context representations. $\beta > 1$ favors similarity between distributions of slot value context representations, $\beta < 1$ lends more weight to similarity between distributions of slot value representations. The larger the $\mathrm{STM}_\beta(s_a, s_b)$, the higher the transferability from slot s_a to slot s_b.

Maximum Mean Discrepancy (MMD) is employed to calculate $sim(p, q)$. MMD is usually used as a loss function in transfer learning (Tzeng et al., 2014; Zhang et al., 2015; Long et al., 2015, 2016, 2017; Yan et al., 2017). It minimizes the difference between different domains to obtain the domain-invariant features. It serves as test statistics to determine if two distributions are the same, as well as measure the similarity between two distributions. The smaller the MMD is, the higher the similarity between distributions is. Let $\mathcal{F}$ be a class of functions $\mathcal{F} : X \to R$. Let $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_n\}$ be samples composed of independent and identically distributed observations drawn from distribution p and q, respectively. MMD is defined as Equation (2), and the square of the MMD can be empirically estimated by Equation (3) (Borgwardt et al., 2006):

$$\mathrm{MMD}^2[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}}(E_p[f(x)] - E_p[f(y)]) \tag{2}$$

$$\mathrm{MMD}^2[\mathcal{F}, X, Y] = \frac{1}{m(m-1)}\sum_{i \neq j}^{m} k(x_i, x_j) + \frac{1}{n(n-1)}\sum_{i \neq j}^{n} k(y_i, y_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(x_i, y_j) \tag{3}$$

where k is the kernel function. Gaussian kernel functions are usually used as. Therefore, the similarity between the distributions of slot value representations of slots s_a and s_b, and the similarity between distributions of slot value context representation of slots s_a and s_b are Equations (4) and (5) respectively:

$$sim(p_v(s_a), p_v(s_b)) = \mathrm{MMD}^2[\mathcal{F}, \Omega_{va}, \Omega_{vb}] \tag{4}$$

$$sim(p_c(s_a), p_c(s_b)) = \mathrm{MMD}^2[\mathcal{F}, \Omega_{ca}, \Omega_{cb}] \tag{5}$$

where Ω_{vi} and Ω_{ci} is the sample set of the slot values representation distribution and the sample set of the slot value context representation distribution corresponding to slots $s_i(i = \{a, b\})$.

Given labeled data $D_{sa} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N_a}$ for slot s_a , where $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \cdots, x_l^{(i)})$ is a sequence of words, $y^{(i)} = (y_1^{(i)}, y_2^{(i)}, \cdots, y_l^{(i)})$ is the corresponding label sequence. Since $x^{(i)}$ contains the slot value of slot s_a, there is either "B-s_a" (the slot value is a word) or "B-s_a" and "I-s_a" (the slot value includes several words) in the label sequence. We first extract all slot value words from labeled dataset D_{sa}, and have the sample set Ω_{va} of the slot value representation of slot s_a, as shown in Equation (6):

$$\Omega_{va} = \left\{E(x_j^{(i)}) \mid if\ I_{va}(x_j^{(i)}, s_a)\right\}_{i=1, j=1}^{i=N_a, j=l} \tag{6}$$

where E is a word embedding mapping, and I_{va} indicates whether $x_j^{(i)}$ is the slot value of slot s_a defined as Equation (7):

$$I_{va}(x_j^{(i)}, s_a) = \begin{cases} 1, & if\ y_j^{(i)} = B\text{-}s_a \\ & or\ y_j^{(i)} = I\text{-}s_a \\ 0, & otherwise \end{cases} \tag{7}$$

Then we extract the slot values context. N words before and after the slot values are extracted to form the sample set Ω_{ca} as shown in Equation (8):

$$\Omega_{ca} = \left\{E(x_k^{(i)}) \mid if\ I_{ca}\ (x_k^{(i)}, s_a)\right\}_{i=1, k=1}^{i=N_a, k=l} \tag{8}$$

where E is a word embedding mapping, and I_{ca} indicates whether $x_k^{(i)}$ is the slot value context for slot s_a defined as Equation (9):

$$I_{ca}(x_k^{(i)}, s_a) = \begin{cases} 1,\ if\ x_k^{(i)} \neq B\text{-}s_a\ and\ x_k^{(i)} \neq I\text{-}s_a \\ \quad and\ (y_{k+N}^{(i)} = B\text{-}s_a\ or\ y_{k-N}^{(i)} = I\text{-}s_a \\ \quad or\ y_{k-N}^{(i)} = B\text{-}s_a) \\ 0,\ otherwise \end{cases} \tag{9}$$

Similarly, we can obtain Ω_{vb} and Ω_{cb}. Based on Ω_{va}, Ω_{ca}, Ω_{vb} and Ω_{cb}, we can calculate

$sim(p_v(s_a), p_v(s_b))$ and $sim(p_c(s_a), p_c(s_b))$ based on Equations (4) and (5) and then calculate $\text{STM}_\beta(s_a, s_b)$ based on Equation (1).

Slot transferability has the following two properties:

Symmetry STM is symmetric. Let the transferability from s_b to s_a be $\text{STM}_\beta(s_b, s_a)$, we have:

$$\text{STM}_\beta(s_a, s_b) = \text{STM}_\beta(s_b, s_a) \tag{10}$$

Relativity When comparing the STM between two slot pairs, it is meaningful only their source slot or target slot is the same. When $\text{STM}_\beta(s_a, s_b) < \text{STM}_\beta(s_c, s_b)$, the transferability from s_a to s_b is higher than that from s_c to s_b, or when $\text{STM}_\beta(s_a, s_b) < \text{STM}_\beta(s_a, s_c)$, the transferability from s_a to s_b is higher than that from s_a to s_c. The comparison between $\text{STM}_\beta(s_a, s_b)$ and $\text{STM}_\beta(s_c, s_d)$ is meaningless.

3.2 Selection of source slots based on STM

Given a slot set $S = \{s_1, \cdots, s_{ns}\}$ from source tasks and the target slot s_t. Each source slot s_a has a labeled dataset $D^T_{sa} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N^T_{sa}}$, and the target slot s_t is given a labeled dataset $D^V_{st} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N^V_{st}}$ for validation. We select a slot set S_t for training the target slot filling model from S basing on the following steps.

1. For each slot s_i in the source slot set, we calculate the transferability $\text{STM}_\beta(s_i, s_t)$.
2. After sorting $[s_1, \cdots, s_{ns}]$ from big to small basing on $\text{STM}_\beta(s_i, s_t)$, we get $[s_{(1)}, \cdots, s_{(n)}]$, the slots sequence according to the order of transferability from highest to lowest.
3. First, we select a slot filling model M and define the union of training data corresponding to the first h slots (h is initialized to 1) in the sorted list $[s_{(1)}, \cdots, s_{(ns)}]$ as $D^{ACC}_h = D^T_{(1)} \cup D^T_{(2)} \cup \cdots \cup D^T_{(h)}$. After replacing the B-$*$ and I-$*$ tags on D^{ACC}_h and D^V_d with the labels B and I, we train the model M with D^{ACC}_h and then test the trained model on D^V_{st} to get the corresponding F1 value, which is denoted as $f_h = M(D^{ACC}_h, D^V_{st})$. Then $h = h + 1$, till F1 gets to its maximum, then $S_t = [s_{(1)}, \cdots, s_{(h)}]$.

3.3 Model training

Given a set of source tasks $T = \{T_1, \cdots, T_n\}$, a target task T_{tgt} and the slot set $S_i = \{s_1, \cdots, s_{N_i}\}$ corresponding to task T_i. We define the set of all source task slots as $S_{union} = S_1 \cup \cdots \cup S_n$, and the set of target task slots as S_{tgt}. For an existing cross-domain slot filling model M_{base}, we deploy our approach on the model by following these steps.

Firstly, the training set and validation set corresponding to the target task and the source task are divided into the training set and validation set corresponding to each slot according to whether the slots contained in each sample. Then select the corresponding source slot set S_{ti} for each slot s_{ti} in the target task slot set from all the source tasks set S_{union}. We combine the source slot set corresponding to all target task slots to get the source slot set for the target task $S_{tgt} = S_{t1} \cup \cdots \cup S_{tN_{tgt}}$ and then replace the labels corresponding to all slots in the source task training set that are not in S_{tgt} with labels O. Finally, the source slot set S_{tgt} and the training data after replacement are used to train the model M_{base}.

4 Experiments

In this section we describe the dataset used for evaluation, the baseline models used for comparison, and more details of the experimental settings.

4.1 Datasets

To evaluate the effectiveness of our approach, we conduct experiments on SNIPS (Coucke et al., 2018). In order to further evaluate the generalization ability of our approach, we also construct a cross-task slot filling dataset called MultiWoz-Slot (MWS) based on the multi-domain task-oriented dialog dataset MultiWoz (Budzianowski et al., 2018; Eric et al., 2020). Table 1 displays some statistics about the two datasets. Details about the two datasets and how the MWS dataset is constructed are described as follows.

SNIPS SNIPS is a public SLU dataset that contains 7 tasks (intents) and 39 slots, and each task contains approximately 2000 training samples. As shown in Table 1, the data contains a total of 14,484 samples, the vocabulary size is 12,134, the average length of the sample utterance is 9, and the average number of slots in each sample is 2.6.

Multiwoz-Slot MultiWoz is a public multidomain task-oriented dialogue dataset that contains 7

MWS	vocab size	utters num	avg len	avg slot num	slot num	slots
attraction^{1}	1132	4616	15.3	1.2	3	$area^{123}, arrive^{45}, day^{235}, depart^{45}, dest^{45}, food^{3}, leave^{45}, name^{123}, people^{235}, price^{23}, stars^{2}, stay^{2}, time^{3}, type^{12}$
hotel^{2}	1481	11258	16	1.7	8	
restaurant^{3}	1734	11669	14.4	1.8	7	
taxi^{4}	961	1758	15.7	1.3	4	
train^{5}	1391	10538	13.9	1.7	6	
total	3314	39839	14.9	1.6	14	

SNIPS	vocab size	utters num	avg len	avg slot num	slot num	slots
AddToPlaylist^{1}	3261	2042	9.2	2.7	5	$album^{4}, artist^{14}, best\_rating^{5}, city^{23}, country^{23}, condition\_description^{3}, condition\_temperature^{3}, cuisine^{2}, current\_location^{3}, entity\_name^{1}, year^{4}, geographic\_poi^{3}, location\_name^{7}, playlist\_owner^{1}, object\_type^{567}, party\_size\_description^{2}, rating\_unit^{5}, movie\_type^{7}, served\_dish^{2}, service^{4}, sort^{24}, genre^{4}, music\_item^{14}, object\_location\_type^{7}, object\_name^{56}, object\_part\_of\_series\_type^{5}, object\_select^{5}, facility^{2}, party\_size\_number^{2}, playlist^{14}, movie\_name^{7}, poi^{2}, rating\_value^{5}, restaurant\_name^{2}, restaurant\_type^{2}, state^{23}, timeRange^{237}, track^{4}, spatial\_relation^{237}$
BookRestaurant^{2}	2639	2073	12	3.2	14	
GetWeather^{3}	2260	2100	9.5	2.3	9	
PlayMusic^{4}	2961	2100	7.1	2.2	9	
RateBook^{5}	1906	2056	8.8	3.8	7	
SearchCreativeWork^{6}	3222	2054	7.8	1.7	2	
SearchScreeningEvent^{7}	1718	2059	8.7	2.2	7	
total	12134	14484	9	2.6	39	

Table 1: Some statistics about SNIPS and MWS. The upper script on task indicates the task id. The upper script on slot indicates the task it belongs.

tasks and 24 slots. Since the (hospital, police) tasks have little conversation data and only appear in the training data, we use user-side utterance for just five tasks (attractions, hotel, restaurant, taxi, train) to construct the MWS dataset, which contains 14 slots. When constructing the training, validation and test data of a task in MWS, we extract the user-side utterance containing the task separately from the conversations in the training set, validation set and test set of Multiwoz. Since the training set of the target task is generally used as the final test set in the cross-domain slot filling task, we combine the validation set and test set as validation set for each task. Table 1 shows the number of slots and the number of sample of training set and validation set included in each task in MWS. As shown in Table 1, the data contains a total of 39,839 samples, and the vocabulary size is 3,314, the average length of the sample utterance is 15, and the average number of slots in per sample is 1.7.

Compared to SNIPS, MWS has smaller vocabulary size and the number of slots in each task. However, MWS has more samples in each task, so when it is used as a cross-domain slot filling dataset, its source tasks have more sufficient training samples, and the correlation between these tasks is stronger.

4.2 Models

We conduct our experiments on the following models.

Concept Tagger (CT) A cross-domain slot filling model proposed by Bapna et al. (2017), which uses the information of the slot descriptions to establish implicit alignment between target slots and source slots.

Robust Zero-shot Tagger (RZT) A model proposed by Shah et al. (2019), which uses the slot value sample of slots to improve the robustness of the model on the target task based on the CT model.

Coarse-to-fine Approach (Coach) A two-stage cross-domain slot filling method proposed by Liu et al. (2020), which splits the cross-domain slot filling task into two stages: coarse-grained BIO 3-way classification and fine-grained slot type classification, and uses slot descriptions in the second stage to help recognize unseen slots.

Coach+TR A variant of Coach proposed by Liu et al. (2020), which further uses template regularization on the basis of Coach to improve the performance of the model on similar or the same slots, is the state-of-the-art model.

4.3 Implementation Details

We deploy the proposed method on above slot filling models CT, RZT, Coach, and Coach+TR.

β is set to 1. A two-layer BiLSTM (Schmidhuber and Hochreiter, 1997) model is used for selecting source slots for all models. 300 dimensions Glove (Pennington et al., 2014) vector is used for word embedding. The hidden layer dimension is set to 300, the learning rate is 0.001. We train the model 30 epochs and select the model with the best performance on the validation set as the final model.

Model		CT		RZT		Coach		Coach+TR	
Data/Domain ↓ Training Setting →		ALL	STM$_1$	ALL	STM$_1$	ALL	STM$_1$	ALL	STM$_1$
MWS	attraction	74.52	**84.99**	74.96	**84.22**	67.26	**73.35**	65.06	**76.16**
	hotel	58.81	47.73	50.63	43.82	59.03	**61.12**	59.00	**60.74**
	restaurant	69.93	63.47	66.29	61.01	78.65	65.36	79.00	71.53
	taxi	51.61	**69.32**	52.82	**66.25**	63.88	**81.17**	70.04	**79.34**
	train	80.78	79.66	80.02	**81.37**	77.68	**82.85**	77.91	**85.23**
	Average F1	67.13	**69.03**	64.94	**67.33**	69.30	**72.77**	70.20	**74.60**
SNIPS	AddToPlaylist	38.82	**41.95**	42.77	**42.92**	45.23	**53.36**	50.90	50.54
	BookRestaurant	27.54	**31.17**	30.68	30.21	33.45	32.60	34.01	32.89
	GetWeather	46.45	**53.03**	50.28	**62.32**	47.93	**60.91**	50.47	**62.38**
	PlayMusic	32.86	23.09	33.12	22.33	28.89	**35.60**	32.01	**34.45**
	RateBook	14.54	**15.39**	16.43	**25.37**	25.67	16.37	22.06	**25.39**
	SearchCreativeWork	39.79	38.72	44.45	42.63	43.91	**49.88**	46.65	**52.21**
	SearchScreeningEvent	13.83	**14.13**	12.25	**15.15**	25.64	23.75	25.63	**26.05**
	Average F1	30.55	**31.07**	32.85	**34.42**	35.82	**38.92**	37.39	**40.56**

Table 2: The main result of the four models (CT, RZT, Coach, Coach +TR) trained using original data (All data) and data selected by our method (STM_1). Scores in each row are F1 of target task.

In order to make a fair comparison, we use the same settings with Liu et al. (2020) to construct the cross-domain slot filling model. We concatenate the 100-dimensional character-level representation and the 300-dimensional word-level representation as word representation. We set the hidden layer dimension of all the BiLSTM encoders to 300 and set the dropout rate to 0.3. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0005. The samples of each task in SNIPS are divided into two parts: 500 samples as validation set, and the remaining samples as training set. When a task is set as a target task, its training set is used as a test set. We evaluate on two datasets respectively. For each test, we choose one task as the target task and set the other tasks as the source tasks.

5 Result and Discussion

In this section, we describe and analyze the experimental results. Firstly, the main results of the experiment are described in Section 5.1. Then, we analyze the impact of some factors on STM in Section 5.2.

5.1 Main Results

Quantitative Analysis Table 2 shows the main result of the four models. For each model, the first column is F1 of the model trained by all labeled data available, i.e., the original way for using the model. The second column is F1 of the model trained by labeled data selected by method proposed in the paper. As can be seen from the Table 2:

1. Our method improves the average F1 (average on all target tasks) of all four models on two datasets consistently, e.g. our method improves the average F1 of coach+TR by 4.4 points on MWS, and by 3.17 points on SNIPS.

2. Our method improves the performance of all four models on most target tasks, even improves several of them by more than 10 points.

Qualitative analysis We perform a qualitative analysis of the STM on the MWS dataset. Figure 1 shows the thermal diagram of slot transferability between any two slots in MWS. Each cell in the figure represents the value of STM_1 between the slot labeled in the horizontal axis and the vertical axis. The higher the brightness, the higher the transferability. The figure is symmetric because the STM_1 is symmetric. It can be found that the slot with high transferability to each other are roughly divided into 7 categories, as shown in Table 3. After observing the data, we find that there are mainly three kinds of slots in the same category. The first type is the slot with high coincidence degree of slot value set. For example, ”attraction-name” and ”taxi-dest” have some common values, such as ”adc theatre”, ”all saints church”, ”county folk museum” and so on. The second kind of slots is that the slot values appear in similar context. For example, ”attraction-name” and ”hotel-name” have some common context words, such as ”about”, ”for”, ”at” and so on. The third kind of slots are the slots with higher coincidence degree, as well as similar context of slot values, such as ”attraction-area” and ”hotel-area”. These phenomena are consistent with the definition of STM.

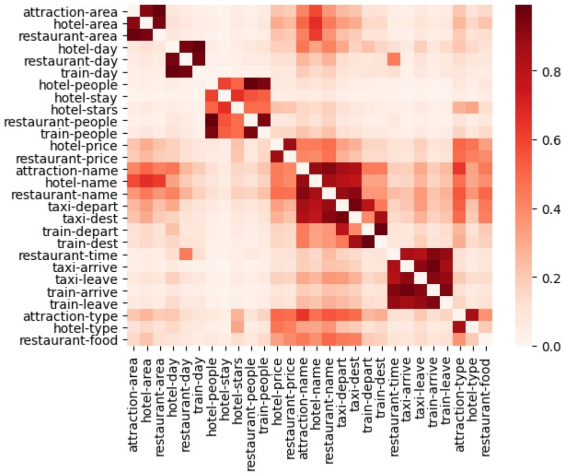

Figure 1: the thermal diagram of slot transferability between any two slots in MWS.

Category	Slots
1	*attraction-area*, *hotel-area*, *restaurant-area*
2	*hotel-day*, *restaurant-day*, *train-day*
3	*hotel-people*, *hotel-stars*, *hotel-stay*, *restaurant-people*, *train-people*
4	*hotel-price*, *restaurant-price*
5	*attraction-name*, *hotel-name*, *restaurant-name*, *taxi-depart*, *taxi-dest*, *train-depart*, *train-dest*
6	*restaurant-time*, *taxi-arrive*, *taxi-leave*, *train-arrive*, *train-leave*
7	*attraction-type*, *hotel-type*, *restaurant-food*

Table 3: The MWS slots categories. The slots in the same categories have the high transferability.

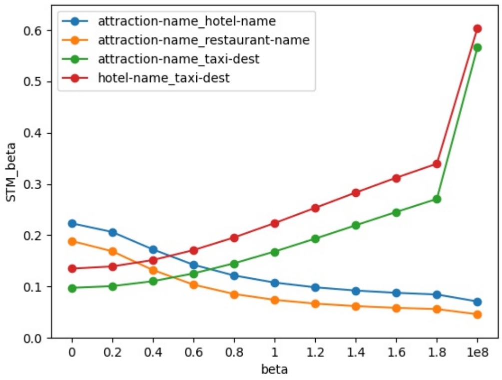

Figure 2: The impact of β on STM.

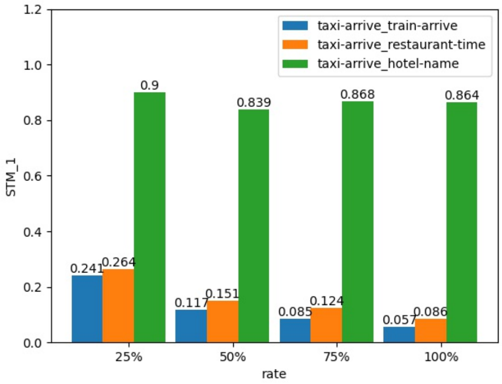

Figure 3: The impact of sample number on STM.

5.2 The impact of some factors on STM

There are three main factors in the calculation of STM_{beta}. The following is an experimental analysis of the impact of the three factors on STM.

The impact of β on STM β parameter determines the weight of similarity between distributions of slot value context representations . We randomly select four slot pairs which are (attraction-name, hotel-name), (attraction-name, restaurant-name), (attraction-name, taxi-dest), and (hotel-name, taxi-dest). In the first two groups, the slot values appear in similar context, but the sets of slot values almost have no intersection. In the last two groups, the sets of slot values set have high consistency, but the contexts of slot values are not similar. β is range from 0 and $+\infty$. When $\beta = 0$, STM_β only measures the similarity of distributions between slot values representations of the two slots, and when $\beta = +\infty$, STM_β only measures the similarity of distributions between the slot value context representation of the two slots. As shown in Figure 2, the STM_β of the first two groups increased with the increase of β, while the STM_β of the last two groups decreased with the increase of β. Therefore, when β increases, the effect of slot value similarity on STM_β becomes greater, and the effect of slot value context similarity on STM_β becomes smaller.

The impact of sample number on STM In order to measure the impact of sample number on STM_β, we randomly selected three slot pairs which are (taxi-arrive, train-arrive), (taxi-arrive, restaurant-time) and (taxi-arrive, hotel-name) for comparative experiment. We select 25%, 50%, 75% and 100% samples from the validation set used to calculate STM_β on the three groups of slots. The experimental results are shown in Figure 3. According to the figure, although the absolute values of STM_β of the three slot pairs changed, their relative relations didn't change. That is, sam-

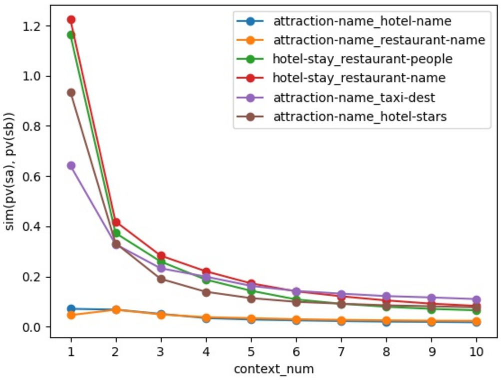

Figure 4: The impact of N on $sim(p_c(s_a), p_c(s_b))$.

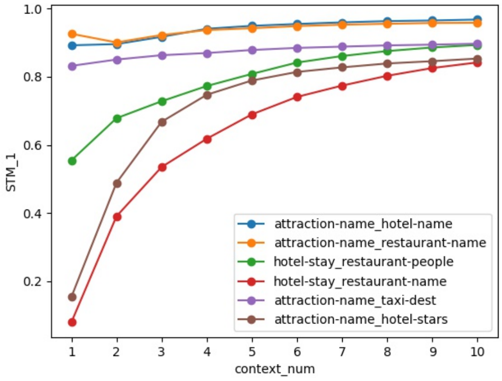

Figure 5: The impact of N on STM.

ple size will affect the value of STM_β. However, for the same source slot, the relationship between STM_β to different target slots does not change.

The impact of N on STM When calculating slot transferability, we fuse distribution similarity of the slot value representations and of the slot value context representations. We select one word before and after slot value as slot value context. In order to measure the impact of slot context window size N on slot transferability, (attraction-name, hotel-name), (attraction-name, restaurant-name), (attraction-name, taxi-dest), (attraction-name, hotel-stars), (hotel-stay, restaurant-people) and (hotel-stay, restaurant-name) are randomly selected for comparing. The first two groups have similar context, the middle two have similar slot values sets, and the last two have low similarity in both slot values and context. N is range from 1 and 10. We observe the change of context representation distribution similarity $sim(p_c(s_a), p_c(s_b))$ and STM_1 among 6 groups of slots. As shown in Figure 4 and Figure 5, the similarity among context representation distributions increases with the increase of context window size N, and the context similarity among 6 groups of slots tends to be the same. In addition, STM increases with the increase of window size N, and the distinction of STM between different types of slot pairs decreases. We conjecture this is due to the fact that the context we extracted contains too much slot-independent context when the window size N becomes large.

5.3 Running time analysis

The method proposed in Sec 3.2 does increase the running time. However, there are two sides of running time. Selecting slots by a Bi-LSTM cost some times, while training the model with selected (less) data saves times. We don't calculate the save minutes in training, however, we find the increase of time consumption in slots selection is small, it is acceptable considering the performance improvements it brings. Since the process in Sec 3.2 is offline and once for a new domain, and the model used in Sec 3.2 is a simple Bi-LSTM model, it increases only a little time. To be more detailed, we conducted experiments on one Titan V GPU, the average running time of the method in Sec 3.2 is 80 minutes for a new domain.

6 Conclusions and Future Work

In this paper, we propose a metric STM to measure the slot transferability of the slots across task, and the calculation of this metric is model-agnostic. Based on this metric, we also propose a cross-domain slot filling method to improve the performance of the existing models by selecting the source slots with high transferability for the target slots. The results on several existing models and datasets show that our method can bring consistent performance improvement to the slot filling models of the target tasks, which show the effectiveness of the STM. We also further explore the impact of some factors on STM. In the future, we hope to use STM to further guide the improvements of models.

Acknowledgments

We would like to thank anonymous reviewers for their suggestions and comments. The work was supported by the National Natural Science Foundation of China (NSFC62076032).

References

Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *Proc. Interspeech 2017*, pages 2476–2480.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471, Florence, Italy. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun Nung Chen, and Ye Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling.

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6642–6649.

Bing Liu and Ian Lane. Recurrent neural network structured output prediction for spoken language understanding.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 19–25, Online. Association for Computational Linguistics.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2016. Unsupervised domain adaptation with residual transfer networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 136–144.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR.

Gregoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, and Dong and Yu. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio Speech Language Processing*, 23(3):530–539.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jürgen Schmidhuber and Sepp Hochreiter. 1997. Long short-term memory. *Neural Comput*, 9(8):1735–1780.

Darsh Shah, Raghav Gupta, Amir Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490, Florence, Italy. Association for Computational Linguistics.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474.

Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. 2017. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2272–2281.

Majid Yazdani and James Henderson. 2015. A model of zero-shot learning of spoken language understanding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249, Lisbon, Portugal. Association for Computational Linguistics.

Xu Zhang, Felix Xinnan Yu, Shih-Fu Chang, and Shengjin Wang. 2015. Deep transfer network: Unsupervised domain adaptation. *arXiv preprint arXiv:1503.00591*.

Su Zhu and Kai Yu. 2018. Concept transfer learning for adaptive language understanding. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 391–399.

Su Zhu, Zijian Zhao, Rao Ma, and Kai Yu. 2020. Prior knowledge driven label embedding for slot filling in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1440–1451.

Word Graph Guided Summarization for Radiology Findings

Jinpeng Hu♠♡, Jianling Li◇, Zhihong Chen♠♡, Yaling Shen♠,
Yan Song♠♡, Xiang Wan♡, Tsung-Hui Chang♠♡
♠The Chinese University of Hong Kong (Shenzhen),
♡Shenzhen Research Institute of Big Data, ◇National University of Defense Technology
♠`{jinpenghu, zhihongchen, yalingshen}@link.cuhk.edu.cn`
◇`jianlingl@nudt.edu.cn` ♡`wanxiang@sribd.cn`
♠`{songyan,changtsunghui}@cuhk.edu.cn`

Abstract

Radiology reports play a critical role in communicating medical findings to physicians. In each report, the impression section summarizes essential radiology findings. In clinical practice, writing impression is highly demanded yet time-consuming and prone to errors for radiologists. Therefore, automatic impression generation has emerged as an attractive research direction to facilitate such clinical practice. Existing studies mainly focused on introducing salient word information to the general text summarization framework to guide the selection of the key content in radiology findings. However, for this task, a model needs not only capture the important words in findings but also accurately describe their relations so as to generate high-quality impressions. In this paper, we propose a novel method for automatic impression generation, where a word graph is constructed from the findings to record the critical words and their relations, then a Word Graph guided Summarization model (**WGSUM**) is designed to generate impressions with the help of the word graph. Experimental results on two datasets, OPENI and MIMIC-CXR, confirm the validity and effectiveness of our proposed approach, where the state-of-the-art results are achieved on both datasets. Further experiments are also conducted to analyze the impact of different graph designs to the performance of our method.[1]

1 Introduction

A radiology report usually contains a findings section (FINDINGS) describing detailed medical observations and an impression section (IMPRESSION) summarizing the most critical observations. In practice, the IMPRESSION is an essential part and plays an important role in delivering critical findings to clinicians. Therefore, summarizing FINDINGS helps to locate the most prominent observations so that the automatic process of doing so greatly eases the workload of radiologists. Recently, many methods are proposed for automatic impression generation (AIG) (Hassanpour and Langlotz, 2016; Zhang et al., 2018; Gharebagh et al., 2020), which are mainly based on the sequence-to-sequence architecture with specific designs for the characteristics of this task. For example, MacAvaney et al. (2019) employed clinical terms within the FINDINGS as key information to enhance AIG. Based on this work, Gharebagh et al. (2020) further proposed to identify the importance of these clinical terms and selected the most salient ones to facilitate the recognition of significant content

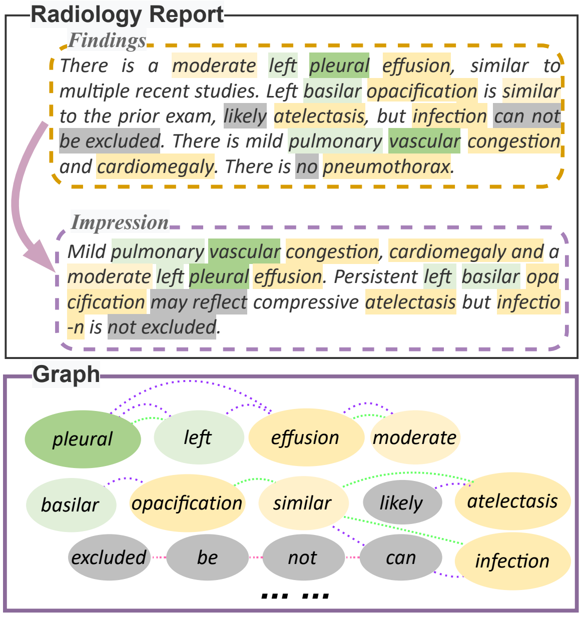

There is a moderate left pleural effusion, similar to multiple recent studies. Left basilar opacification is similar to the prior exam, likely atelectasis, but infection can not be excluded. There is mild pulmonary vascular congestion and cardiomegaly. There is no pneumothorax.

Mild pulmonary vascular congestion, cardiomegaly and a moderate left pleural effusion. Persistent left basilar opacification may reflect compressive atelectasis but infection is not excluded.

Figure 1: An example radiology report including its FINDINGS and IMPRESSION, and a word graph constructed from the FINDINGS. Different colored edges in the graph represent different types of relations. The curved arrow indicates the AIG task to generate the IMPRESSION from the FINDINGS.

---

[1] Our code and the best performing models are released at `https://github.com/cuhksz-nlp/WGSum`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4980–4990
August 1–6, 2021.

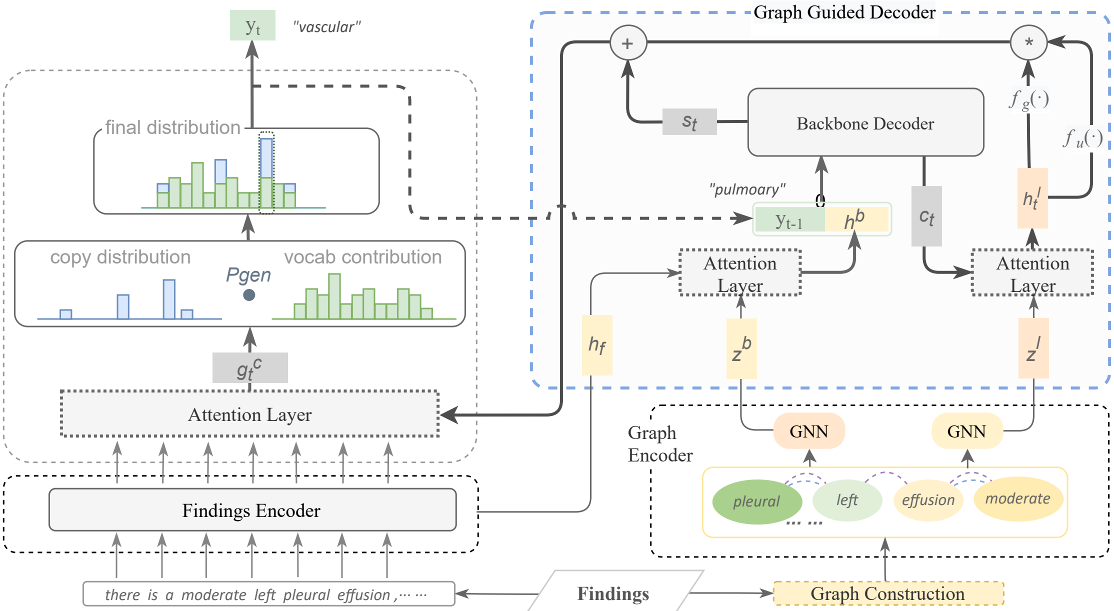

Figure 2: The overall architecture of our proposed method with an example input and sample outputs at the $t-1$ and t step. The FINDINGS Encoder and the Graph Encoder are shown in different grey dashed boxes with their details omitted. The Graph Guided Decoder is illustrated in the blue dashed box.

so as to improve the performance. Although these efforts are able to find the important words to promote AIG, less attention is paid to the important aspect on leveraging the relation information among them. For example, in Figure 1, the observation word "*effusion*" and its modifier word "*moderate*" have a relation (which describes the severity of the symptoms) in between them, where such relation needs to appear in the IMPRESSION. Therefore, to better generate IMPRESSION, in addition to using important words, it is required to recognize the relations among such words in the FINDINGS and describe their corresponding relations for AIG.

In this paper, we propose to enhance AIG via a summarization model integrated with a word graph by leveraging salient words and their relations in the FINDINGS. In detail, the word graph is constructed by identifying the important words in the FINDINGS and building connections among them via different typed relations. To exploit the word graph, a Word Graph guided Summarization model (**WGSUM**) is designed to perform AIG, where the information from the word graph is integrated into the backbone decoder (e.g., LSTM (See et al., 2017) or Transformer (Liu and Lapata, 2019)) from two aspects, including enriching the decoder input as extra knowledge, as well as guiding the decoder to update its hidden states. Experimental results illustrate that WGSUM outperforms all baselines on two benchmark datasets, where the state-of-the-art performance is observed on all datasets in comparison with previous studies. Further analyses also investigate how different types of edges in the graph affects the performance of our proposed model.

2 The Proposed Method

We follow the standard sequence-to-sequence paradigm for AIG. In doing so, we regard the FINDINGS as the source sequence $\mathbf{X} = \{x_1, ..., x_i, ..., x_N\}$, where N is the number of tokens in the FINDINGS, and the goal of the task is to generate a target sequence (i.e., the IMPRESSION) $\mathbf{Y} = \{y_1, ...y_i, ..., y_L\}$, where L is the number of tokens in the IMPRESSION. An overview of our method is shown in Figure 2 where the details are illustrated in following subsections.

2.1 The Overall Structure

The model used in our method contains three major components, i.e., the FINDINGS encoder, the graph encoder, and the graph guided decoder with their details and the training objective described below.

FINDINGS Encoder Given a FINDINGS, denoted by $\mathbf{X}$ with N tokens, LSTM or the standard encoder from Transformer is applied to model the sequence and its output is the hidden state $\mathbf{h}^x$. The process is formulated as

$$\mathbf{h}^x = f_{fe}(x_1, ..., x_i, ..., x_N) \quad (1)$$

where $f_{fe}(\cdot)$ refers to the FINDINGS encoder.

Graph Encoder For the node V and adjacency matrix $\mathbf{A}$ in the graph G constructed from the FINDINGS, we utilize graph neural networks[2] (GNN) to encode them, because GNNs are powerful in encoding graph-like information (Chen et al., 2020; Zheng and Kordjamshidi, 2020; Tian et al., 2021a,b). In detail, two encoders are employed to extract features from G, where one is used to construct the background information and the other is used to generate the dynamic guiding information. The process is thus formalized as

$$\mathbf{z}^b = f_{gb}(V, \mathbf{A}) \tag{2}$$

$$\mathbf{z}^l = f_{gl}(V, \mathbf{A}) \tag{3}$$

where $f_{gb}(\cdot)$ and $f_{gl}(\cdot)$ refer to two graph encoders, with $\mathbf{z}^b$ and $\mathbf{z}^l$ the intermeidate states used to generate the static background information and the dynamic guiding information, respectively.

Graph Guided Decoder In our model, $\mathbf{z}^b$ and $\mathbf{z}^l$ from the graph encoders are integrated into the backbone decoder (e.g., LSTM and Transformer decoder.) and perform a decoding process via

$$P_{\text{vocab}} = f_d(\mathbf{h}^x, \mathbf{z}^b, \mathbf{z}^l) \tag{4}$$

where $f_d(\cdot)$ represents the decoder.

Objective Since the FINDINGS and the IMPRESSION are highly related, a pointer generator (PG) is also introduced to our model by

$$P(y_t \,|\, \mathbf{X}, y_{<t}) = p_{\text{gen}} P_{\text{vocab}}(y_t) + (1 - p_{\text{gen}}) \sum_{i:x_i = y_t} a_i^t \tag{5}$$

where $\mathbf{a}_i^t$ is the distribution over source tokens at step t, which is obtained by performing the attention mechanism on the source tokens; p_{gen} and $(1 - p_{gen})$ are the weights of predicting the next token from the vocabulary or the source tokens, respectively. The model is then trained to maximize the negative conditional log-likelihood $P(\mathbf{Y}|\mathbf{X}, G)$ by of $\mathbf{Y}$ given $\mathbf{X}$ and G:

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{L} \log p\left(y_t \mid y_1, ..., y_{t-1}, \mathbf{X}, G; \theta\right) \tag{6}$$

where θ are the parameters of the model.

2.2 Graph Construction

In FINDINGS, the most critical content that the radiologists need to summarize is the abnormality which usually includes the corresponding specific observations as well as their modifying words. Therefore, in our study, we first extract 5 types of entities from FINDINGS: *anatomy*, *observation*, *anatomy modifier*, *observation modifier*, and *uncertainty*, where these entities are able to cover most of the key words that need to appear in IMPRESSION. In addition, Gharebagh et al. (2020) has shown that fine-grained words are more effective than the entire ontology. Inspiring by this idea, we regard words from these entities as nodes in our graph. To avoid confusion, the repeated words are treated as one single node in each FINDINGS even though they are not presented in the same entity. Besides modifying relation, some other relations are also important for IMPRESSION generation, such as relations between *anatomy* and *observation*. For example, in Figure 1, the relation between "*pleural*" (*anatomy*) and "*effusion*" (*observation*), which can be obtained for the detailed abnormality in the IMPRESSION. To capture these types of relations, we leverage dependency trees which have been widely used to model word-word relations in many studies (Tian et al., 2020a,b; Pouran Ben Veyseh et al., 2020; Chen et al., 2021). Thus, we define three types of edges for our word graph. Note that, each FINDINGS has its corresponding word graph:

- **Type** I: this is the type using the natural order of words in an entity. In detail, we connect words if they are adjacent in the same entity. In Figure 1, the pink dashed lines serve as the type I edge. For example, "*endotracheal tube*" is an entity, so that "*endotracheal*" is connected to "*tube*".
- **Type** II: this is the type using the relations among entities within the same category (e.g., observation and its modifier, and anatomy and its modifier). As shown in Figure 1, given an *observation*,"*effusion*" and a *observation modifier* "*moderate*", the relation is constructed by connecting them with a green dash line.
- **Type** III: this is the type using the relations among entities across different categories (e.g., observation and anatomy). Different from the previous two types, this type is able to provide the global relation information while the previous two types emphasize the local information. In detail, we construct a dependency tree using stanza in the Universal Dependencies (UD) format (Nivre et al., 2020). As shown in Figure 1, given "*effusion*" and its head "*left*", they are connected with a purple dash line.

---

[2] It can be any type (e.g., graph convolutional networks (GCN) and graph attention networks (GAT)).

Then the nodes and edges in graph are recorded by a node list V and an adjacency matrix $\mathbf{A}$ which are then used as the input of graph encoder.

2.3 Graph Guided Decoder

We utilize graph to generate two different kinds of information and they are working on two aspects: enriching decoder input by static background information $\mathbf{h}^b$ and controlling decoder hidden state by dynamic guiding information $\mathbf{h}^l$, which are introduced in following parts.

Background Information Since the graph can be considered as a condensing of source sequence (i.e., FINDINGS) which contains the most important information, it is appropriate to serve as a static background information to enrich the decoder inputs. The first output $\mathbf{z}^b$ of graph encoder is used to construct the background information. For the hidden state $\mathbf{z}_i^b$ in $\mathbf{z}^b$ of each node, we can obtain attention weights by:

$$e_i^b = \mathbf{p}_b^\top \tanh(\mathbf{W}_b \mathbf{z}_i^b + \mathbf{W}_h \mathbf{h}_f) \tag{7}$$

$$\mathbf{a}^b = \text{softmax}(\mathbf{e}^b) \tag{8}$$

where $\mathbf{W}_b$,$\mathbf{W}_h$ and $\mathbf{p}_b$ are learnable parameters. For LSTM, we define $\mathbf{h}_f$ as the final hidden state, and for Transformer, we calculate the mean of all hidden states as $\mathbf{h}_f$. The attention distribution $\mathbf{a}^b$ can be viewed as a probability distribution over nodes in graph. Next, $\mathbf{a}^b$ is used to produce a weighted sum of the nodes and then we obtain the static background information:

$$\mathbf{h}^b = \sum_i a_i^b \mathbf{z}_i^b \tag{9}$$

For clarity, we simplify Equation (7), (8) and (9) as a function AttCon$(\cdot)$. Therefore, $\mathbf{h}^b$ can be obtained by:

$$\mathbf{h}^b = \text{AttCon}(\mathbf{z}^b, \mathbf{h}_f) \tag{10}$$

In our model, the background information $\mathbf{h}^b$ is directly concatenated to the decoder input. For each decoder input $\mathbf{y}_{t-1}$ at step t, it is expanded as $\mathbf{y}'_{t-1} = [\mathbf{y}_{t-1}, \mathbf{h}^b]$.

DATA	OPENI			MIMIC-CXR		
	TRAIN	VAL	TEST	TRAIN	VAL	TEST
REPORT #	2,400	292	576	122,014	957	1,606
AFL	37.89	37.77	37.98	55.78	56.57	70.67
AFS	5.75	5.68	5.77	6.50	6.51	7.28
AFE	18	17	18	27	28	36
AIL	10.43	11.22	10.61	16.98	17.18	21.71
AIS	2.86	2.94	2.82	3.02	3.04	3.49

Table 1: The statistics of the two benchmark datasets (random split for OPENI and official split for MIMIC-CXR) in the training, validation, and test sets, where the number of reports, the averaged sentence-based (AFS, AIS) and word-based length (AFL, AIL) of IMPRESSION and FINDINGS, and the averaged number of edges in graph (AFE) are reported.

Dynamic Guiding Information Since $\mathbf{h}^b$ remains unchanged and works as the global static knowledge during the decoding process, to make the guidance more flexibly, the other information $\mathbf{z}^l$ from graph encoder in Equation (3) is used to generate dynamic guiding information $\mathbf{h}_t^l$ for each decoding step t. For different backbone decoders, there is a little difference in generating $\mathbf{h}_t^l$. For LSTM decoder, each cell updates its information by two states and one input: cell state $\mathbf{c}_{t-1}$, hidden state $\mathbf{s}_{t-1}$ and input $\mathbf{y}'_{t-1}$, which is formulated as:

$$[\mathbf{c}_t, \mathbf{s}'_t] = \text{LSTM}(\mathbf{c}_{t-1}, \mathbf{s}_{t-1}, \mathbf{y}'_{t-1}) \tag{11}$$

where $\mathbf{c}_t$ usually contains rich contextual information and it is appropriate to compute guiding information $\mathbf{h}_t^l$.[3]

$$\mathbf{h}_t^l = \text{AttCon}(\mathbf{z}^l, \mathbf{c}_t) \tag{12}$$

For Transformer, the general decoder only has one hidden state $\mathbf{s}'_t$ which is the output of the last layer. In this part, we regard the output of the penultimate layer as another hidden state $\mathbf{c}_t$ which is then used to generate dynamic information $\mathbf{h}_t^l$ by:

$$\mathbf{h}_t^l = \text{softmax}(\mathbf{c}_t(\mathbf{z}^l)^\top)\mathbf{z}^l \tag{13}$$

After obtaining the dynamic guidance $\mathbf{h}_t^l$ from the LSTM decoder or Transformer decoder, it is then utilized to update decoder hidden state $\mathbf{s}'_t$ by:

$$\mathbf{s}_t = \mathbf{s}'_t + f_g(\mathbf{z}^l) \cdot f_u(\mathbf{z}^l) \tag{14}$$

where $f_g(\cdot)$ and $f_u(\cdot)$ are fully connected layers.

Vocabulary Distribution To incorporate the FINDINGS information for final prediction, we calculate the attention context vector $\mathbf{g}_t^c$ by the same way as Equation (7), (8) and (9) using sequence encoder hidden state $\mathbf{h}^x$ as well as the updated $\mathbf{s}_t$:

$$\mathbf{g}_t^c = \text{AttCon}(\mathbf{h}^x, \mathbf{s}_t) \tag{15}$$

Then both $\mathbf{g}_t^c$ and decoder hidden state $\mathbf{s}_t$ are used to calculate vocabulary distribution at step t:

$$P_{vocab} = \text{softmax}\left(\mathbf{Q}' \tanh\left(\mathbf{Q}\left[\mathbf{s}_t; \mathbf{g}_t^c\right]\right)\right) \tag{16}$$

where $\mathbf{Q}'$ and $\mathbf{Q}$ are learnable weights.

---

[3] We also try to use hidden state $\mathbf{s}'_t$ from LSTM to calculate $\mathbf{h}_t^l$, but the performance is not better than that of $\mathbf{c}_t$.

Data	Model	ROUGE			FC		
		R-1	R-2	R-L	P	R	F-1
OpenI	PG-LSTM	63.21	54.13	62.78	-	-	-
	WGSum (LSTM+GCN)	63.69	54.88	63.33	-	-	-
	WGSum (LSTM+GAT)	**64.32**	**55.48**	**63.97**	-	-	-
	PG-Trans	59.66	49.41	59.18	-	-	-
	WGSum (Trans+GCN)	60.95	50.67	60.85	-	-	-
	WGSum (Trans+GAT)	**61.63**	**50.98**	**61.73**	-	-	-
MIMIC-CXR	PG-LSTM	46.41	32.33	44.76	54.72	45.37	49.61
	WGSum (LSTM+GCN)	46.93	32.69	45.25	55.23	46.21	50.32
	WGSum (LSTM+GAT)	**47.48**	**33.03**	**45.43**	**55.82**	**47.13**	**51.11**
	PG-Trans	47.16	32.31	45.47	56.18	49.08	52.39
	WGSum (Trans+GCN)	47.93	32.63	46.23	56.37	50.84	53.46
	WGSum (Trans+GAT)	**48.37**	**33.34**	**46.68**	**56.83**	**51.22**	**53.88**

Table 2: The performance of the baselines and our proposed methods with different GNNs on OpenI and MIMIC-CXR. R-1, R-2 and R-L denote ROUGE-1, ROUGE-2 and ROUGE-L, respectively.

3 Experiment Settings

3.1 Dataset

We conduct our experiments on the following two datasets of radiology reports: **OpenI** (Demner-Fushman et al., 2016) and **MIMIC-CXR** (Johnson et al., 2019). The former is collected by Indiana University with 3,268 reports after pre-processing. The latter contains a larger amount of data, where we obtain 124,577 after pre-processing. In our experiments, the original Impressions written by the radiologists are considered as the ground truth. For both datasets, we follow Zhang et al. (2018) and Gharebagh et al. (2020) to exclude the following types of reports: (a) incomplete reports; (b) reports whose Findings have less than 10 words; and (c) reports with Impression words less than 2. For the dataset splits, OpenI is partitioned into train/validation/test set by 2400:292:576 randomly in our experiments. For MIMIC-CXR, we apply two splits, one is the official split as the dataset published by Johnson et al. (2019), and the other is a random split with a ratio of 8:1:1, which is the same as Gharebagh et al. (2020). The statistics of the datasets are shown in Table 1.

3.2 Baseline and Evaluation Metrics

To compare the performance of our proposed models, we use the following models as our baselines:

- **PG-LSTM** (See et al., 2017): This is Pointer Generator Network (PGN) with copy mechanism where both encoder and decoder are vanilla LSTMs without graph information.
- **PG-Trans** (Liu and Lapata, 2019): This is also a PGN with both the encoder and decoder replaced with the transformer.

Besides, we also compare our model with those in previous studies, including extractive summarization models, e.g., **LexRank** (Erkan and Radev, 2004), **TransformerEXT** (Liu and Lapata, 2019), as well as abstractive summarization models, e.g., **CAVC** (Song et al., 2020), **CGU** (Lin et al., 2018) and **OntologyABS** (Gharebagh et al., 2020). In our experiments, we use ROUGE metrics (Lin, 2004) to evaluate the generated Impressions. We only report F_1 scores of ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L), where R-1, R-2 are unigram and bigram overlap measuring the informativeness and R-L is the longest common sub-sequence overlap aiming to assess fluency. In addition, to evaluate the factual consistency (FC), CheXbert (Smit et al., 2020)[4] is utilized to detect 14 observations related to diseases in reference impressions and generated impressions. Then precision, recall and F1 score are used to evaluate the performance.

3.3 Implementation Details

We employ stanza (Zhang et al., 2020)[5], a python-based natural language processing library, to recognize named entities and get the syntactic analysis. Then we use the extracted entities and dependency tree to construct graph for each Findings. We im-

---

[4] FC only apply to MIMIC-CXR since the CheXbert is designed for MIMIC-CXR and we obtain Chexbert from `https://github.com/stanfordmlgroup/CheXbert`

[5] Stanza provides packages to process the clinical text: `https://stanfordnlp.github.io/stanza/`.

DATA	OpenI			MIMIC-CXR					
MODEL	RANDOM SPLIT			OFFICIAL SPLIT			RANDOM SPLIT		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
LexRank	14.63	4.42	14.06	18.11	7.47	16.87	-	-	-
TransformerExt	15.58	5.28	14.42	31.00	16.55	27.49	-	-	-
CAVC	53.18	39.59	52.86	43.97	29.36	42.50	-	-	-
CGU	61.60	53.00	61.58	46.50	32.61	44.98	-	-	-
OntologyABS†	-	-	-	-	-	-	53.57	40.78	51.81
WGSum (LSTM+GAT)	**64.32**	**55.48**	**63.97**	47.48	33.03	45.43	54.97	43.64	53.81
WGSum (Trans+GAT)	61.63	50.98	61.73	**48.37**	**33.34**	**46.68**	**56.38**	**44.75**	**55.32**

Table 3: Comparisons of our proposed models with previous study on the test sets of IU X-Ray and MIMIC-CXR with respect to ROUGE metric. † refers to that the results is directly cited from the original paper.

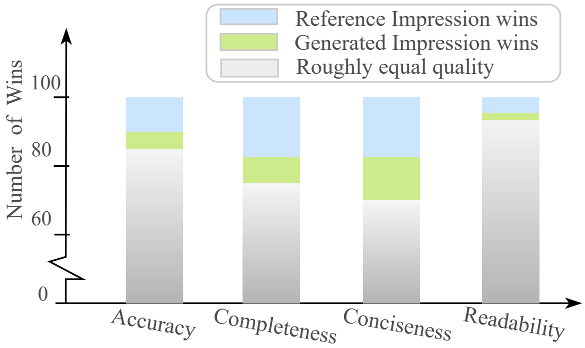

Figure 3: Experts evaluation results of 100 sampled test examples. For each of the metrics, more than 80% of the generated Impressions are at least as good as the reference Impressions.

plement our model based on Zhang et al. (2018)[6] and Liu and Lapata (2019)[7]. Since the quality of text representation plays an important role in model performance (Mikolov et al., 2013; Song et al., 2017, 2018; Peters et al., 2018; Song and Shi, 2018; Devlin et al., 2019; Joshi et al., 2020; Song et al., 2021), we try two powerful Findings encoders, namely, LSTM and Transformer, which have achieved state-of-the-art results in many natural language processing tasks. For WGSum (LSTM+GAT)), we employ 2-layer GAT with hidden size of 200 as our graph encoder, 2-layer Bi-LSTM encoder for findings sequence with hidden size of 100 for each direction and 1-layer LSTM for decoder with hidden size of 200. The dropout is set to 0.5 for embedding layer. We use Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.001. For WGSum(Trans+GAT), the graph encoder is a 2-layer GAT with hidden size 512 and the Findings encoder is a 6-layer Transformer with 512 hidden size and 2,048 feed-forward filter size. The decoder is also a 6-layer Transformer with hidden size 512.

4 Results and Analyses

4.1 Effect of word graph

To illustrate the validity of the word graph, we conduct experiments with the aforementioned baselines on the two benchmark datasets. Besides, we also try two different types of GNNs: GCN (Kipf and Welling, 2016) and GAT (Veličković et al., 2017) respectively. The results are shown in Table 2 and there are several observations drawn from different aspects. First, we can observe that for our word graph, the encoder GAT is more effective than GCN where GAT can bring more significant improvements on the two datasets. The main reason might be that GAT is more powerful in updating node representation via self-attention. Second, integrating the word graph into the two different PGNs gains better performance on both the datasets, which confirms the usefulness of the word graph. Third, for OpenI, the LSTM-based models outperform much more than the Transformer-based models, while on MIMIC-CXR, the Transformer-based models are more effective. The main reason could be that the LSTM is able to obtain prominent performance in the small dataset and the Transformer is more powerful under a large amount of data. Fourth, on the FC metrics on MIMIC-CXR, our proposed methods also outperform the Base model, indicating that the generated Impressions from our methods are more accurate and reasonable, which is because the word graph can provide both key word and relation information for the generation process so that the decoder tends to produce words with correct relations.

---

[6] `https://github.com/yuhaozhang/summarize-radiology-findings`

[7] `https://github.com/nlpyang/PreSumm`

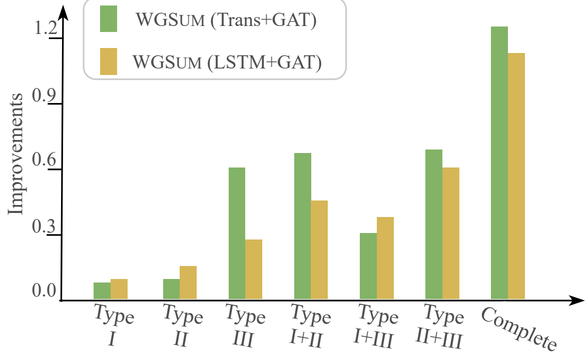

Figure 4: The improvements (R-1) of WGSum with different graph edges on MIMIC-CXR dataset.

4.2 Comparison with Previous Studies

In this subsection, we compare our models with existing studies on the two datasets and report all results (i.e., ROUGE scores) in Table 3. We can get several observations. First, the abstractive models are apparently more effective than the extractive models in AIG, owing to the characteristics of Findings and Impression in radiology reports. Second, our models with the word graph show the effectiveness of both key words information and their relations in this task when being compared to the previous models that only leverage medical term information, e.g., OntologyABS only uses ontology information in database RadLex[8]. Third, our methods achieve the best performance among all previous models, which demonstrates that using background knowledge and dynamic guidance information to control the decoding process is an appropriate design to improve the quality of the generated Impressions.

4.3 Expert Evaluation

Since the limitation of ROUGE metrics, we further conduct an expert evaluation for a better understanding of the generated Impressions. We randomly select 100 generated Impressions along with their corresponding reference Impressions and Findings from MIMIC-CXR. To avoid potential bias, we randomly order the predicted and reference Impressions. We extend Gharebagh et al. (2020) metrics to four metrics: Accuracy, Completeness, Conciseness and Readability. Three medical experts are employed to score each Impression on these metrics. Figure 3 presents the results of human evaluation. We can observe that although reference Impressions written by the radiologists are still better, there are still over 80% of generated Impressions have roughly equal or better quality. About 85%, 75%, 70%, and 94% of generated Impressions are equal to human written Impressions on the four metrics (Accuracy, Completeness, Conciseness, and Readability). In addition, there are even 5%, 7%, 12%, 2% of generated Impressions surpassed the reference Impressions on these metrics.

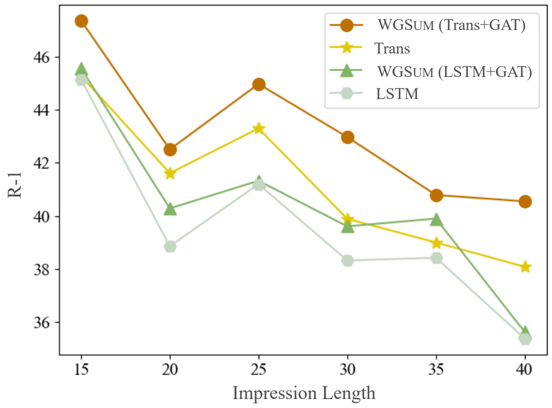

Figure 5: The performance (R-1) of generated Impressions from WGSum (Trans+GAT), PG-Trans, WGSum (LSTM+GAT) and PG-LSTM on the MIMIC-CXR test set. Each group contains Impressions of different length intervals.

4.4 Analyses

We conduct further analyses on Graph Edge, Impression Length, and Case Study.

Graph Edge As we introduced before, our graph contains three types of edges, i.e., entity interval edge (Type I), entity modifier edge (Type II), and edge from dependency tree (Type III). To show the effect of different edges, we conduct experiments for WGSum (LSTM+GAT) and WGSum (Trans+GAT) with different edges on MIMIC-CXR. The improvements from these different edge combinations are shown in Figure 4. First, we can observe that models incorporating word graph outperform the baselines no matter with what type of edge, indicating the effectiveness of our innovation in combining the entity words and their relations into the word graph. Second, regardless of WGSum (LSTM+GAT) or WGSum (Trans+GAT), Type III edge can bring the most significant improvements, while Type I brings little improvements. The main reason might be that the dependency tree contains more comprehensive and accurate relations for the entity words from Findings. Third, WGSum (Trans+GAT) usually obtains bet-

---

[8] `http://www.radlex.org/Files/radlex3.10.xlsx`

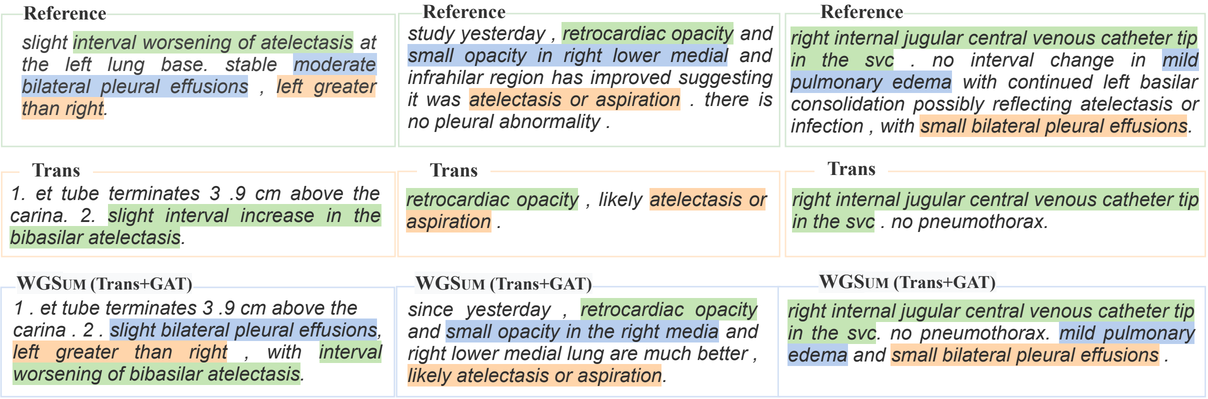

Reference

slight interval worsening of atelectasis at the left lung base. stable moderate bilateral pleural effusions , left greater than right.

Reference

study yesterday , retrocardiac opacity and small opacity in right lower medial and infrahilar region has improved suggesting it was atelectasis or aspiration . there is no pleural abnormality .

Reference

right internal jugular central venous catheter tip in the svc . no interval change in mild pulmonary edema with continued left basilar consolidation possibly reflecting atelectasis or infection , with small bilateral pleural effusions.

Trans

1. et tube terminates 3 .9 cm above the carina. 2. slight interval increase in the bibasilar atelectasis.

Trans

retrocardiac opacity , likely atelectasis or aspiration .

Trans

right internal jugular central venous catheter tip in the svc . no pneumothorax.

WGSum (Trans+GAT)

1 . et tube terminates 3 .9 cm above the carina . 2 . slight bilateral pleural effusions, left greater than right , with interval worsening of bibasilar atelectasis.

WGSum (Trans+GAT)

since yesterday , retrocardiac opacity and small opacity in the right media and right lower medial lung are much better , likely atelectasis or aspiration.

WGSum (Trans+GAT)

right internal jugular central venous catheter tip in the svc. no pneumothorax. mild pulmonary edema and small bilateral pleural effusions .

Figure 6: Examples of the generated IMPRESSIONS from two models (i.e., the PG-TRANS and the WGSUM (TRANS+GAT), respectively) , as well as the reference IMPRESSIONS.

ter improvements than WGSUM (LSTM+GAT) in most cases.

IMPRESSION Length Another factor that could affect the model performance is the number of tokens in the IMPRESSION. To test the effect of IMPRESSION length, we categorize all generated IMPRESSIONS in the MIMIC-CXR test set into 6 groups (within [15,40] with the interval of 5) and compare the R-1 score for each group. The results are shown in Figure 5. There are several observations. First, when IMPRESSION gradually increases in the length, the performance of all models shows a downward trend, which indicates that long IMPRESSION generation is difficult for all models. Second, TRANSFORMER-based methods are more effective than LSTM based models, especially for long IMPRESSION. The main reason might be that the Transformer is more powerful in dealing with longer sequences via its self-attention mechanism. Third, both WGSUM (TRANS+GAT) and WGSUM (LSTM+GAT) show their superiority when being compared to their baselines and obtain better results in almost all of the groups.

Case Study To further analyze the effect of our proposed model, we perform qualitative analysis on some cases with their reference and generated IMPRESSIONS from different models. Figure 6 shows three cases from MIMIC-CXR where different colors on text refer to varied key information. It is found that in the first case, when referring to the corresponding FINDINGS our model generates more complete IMPRESSION than the reference, e.g., *"et tube terminates 3 .9 cm above the carina"* is a helpful text piece but does not appear in reference IMPRESSION. In addition, compared to the reference IMPRESSIONS written by radiologists, our method covers almost all of the key information in the generated IMPRESSIONS. For example, the key information *"moderate bilateral pleural effusions"*,*"mild pulmonary edema"* and *"small opacity in the right media "* in the three examples are not generated in PG-TRANS model, but they are necessary for describing the clinical condition.

5 Related Work

Our work focuses on summarizing the FINDINGS of radiology reports to generate the IMPRESSION, which is essentially an abstractive summarization task. For abstractive summarization, there exists a serious problem known as hallucination (Maynez et al., 2020), in which the generated summary contains fictional content. This problem also exists in AIG and would lead to misdiagnosis for the patient. To tackle this problem, in the general domain, many attempts have been made in terms of guiding information to control the generation process and output the high-quality summary (Li et al., 2018; Hsu et al., 2018; Pilault et al., 2020; Huang et al., 2020; Haonan et al., 2020). For the IMPRESSION generation task in the medical domain, there also exist several solutions. Zhang et al. (2018) encodes a section of the radiology report as the background information to guide the decoding process. MacAvaney et al. (2019) employs the entire ontological terms extracted from FINDINGS as the medical terms, and then enhances the summarizer by selecting the salient information. Gharebagh et al. (2020) further splits ontological terms into words and then incorporates these words into summarization by a separate encoder. Compared to these studies, our model offers an alternative solution to robustly enhancing guidance with a word graph for summarizing the FINDINGS of radiology reports without requiring external resources. To our best knowledge, this is the first work employing word relation information for AIG.

6 Conclusion

In this paper, we propose a novel method for AIG, where a word graph is constructed from the FINDINGS by identifying the salient words and their relations and a graph-based model WGSUM is designed to generate IMPRESSIONs with the help of the word graph. In doing so, the information from the word graph guides the decoding process with the help of background information and dynamic guiding information. Experimental results on two benchmark datasets show the validity of our proposed method, which obtains the state-of-the-art performance on both datasets. Further analyses on the effect of edge types demonstrate that our model can generate IMPRESSION with accurate medical items.

Acknowledgements

This work is supported by Chinese Key-Area Research and Development Program of Guangdong Province (2020B0101350001) and also partially supported by NSFC under the project "The Essential Algorithms and Technologies for Standardized Analytics of Clinical Texts" (12026610). We thank three reviewers from Sun Yat-sen University: Mei Feng, Xiang Chuqi, and Liang Feiyan.

References

Guimin Chen, Yuanhe Tian, and Yan Song. 2020. Joint Aspect Extraction and Sentiment Analysis with Directional Graph Convolutional Networks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 272–279.

Guimin Chen, Yuanhe Tian, Yan Song, and Xiang Wan. 2021. Relation Extraction with Type-aware Map Memories of Word Dependencies. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Dina Demner-Fushman, Marc D Kohli, Marc B Rosenman, Sonya E Shooshan, Laritza Rodriguez, Sameer Antani, George R Thoma, and Clement J McDonald. 2016. Preparing a Collection of Radiology Examinations for Distribution and Retrieval. *Journal of the American Medical Informatics Association*, 23(2):304–310.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of artificial intelligence research*, 22:457–479.

Sajad Sotudeh Gharebagh, Nazli Goharian, and Ross Filice. 2020. Attend to Medical Ontologies: Content Selection for Clinical Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1899–1905.

Wang Haonan, Gao Yang, Bai Yu, Mirella Lapata, and Huang Heyan. 2020. Exploring Explainable Selection to Control Abstractive Generation. *arXiv*, pages arXiv–2004.

Saeed Hassanpour and Curtis P Langlotz. 2016. Information Extraction from Multi-institutional Radiology Reports. *Artificial intelligence in medicine*, 66:29–39.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141.

Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge Graph-Augmented Abstractive Summarization with Semantic-Driven Cloze Reward. *arXiv preprint arXiv:2005.01159*.

Alistair EW Johnson, Tom J Pollard, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Yifan Peng, Zhiyong Lu, Roger G Mark, Seth J Berkowitz, and Steven Horng. 2019. Mimic-cxr-jpg, a Large Publicly Available Database of Labeled Chest Radiographs. *arXiv preprint arXiv:1901.07042*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding Generation for Abstractive Text Summarization Based on Key Information Guide Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 55–60.

Chin-Yew Lin. 2004. Rouge: A Rackage for Automatic Evaluation of Summaries. In *Text summarization branches out*, pages 74–81.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global Encoding for Abstractive Summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 163–169.

Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Sean MacAvaney, Sajad Sotudeh, Arman Cohan, Nazli Goharian, Ish Talati, and Ross W Filice. 2019. Ontology-Aware Clinical Abstractive Summarization. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1013–1016. ACM.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On Faithfulness and Factuality in Abstractive Summarization. *arXiv*, pages arXiv–2005.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection. *arXiv preprint arXiv:2004.10643*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Christopher Pal. 2020. On Extractive and Abstractive Neural Document Summarization with Transformer Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319.

Amir Pouran Ben Veyseh, Nasim Nouri, Franck Dernoncourt, Quan Hung Tran, Dejing Dou, and Thien Huu Nguyen. 2020. Improving Aspect-based Sentiment Analysis with Gated Graph Convolutional Networks and Syntax-based Regulation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4543–4548, Online.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Y Ng, and Matthew Lungren. 2020. Chexbert: Combining Automatic Labelers and Expert Annotations for Accurate Radiology Report Labeling Using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1500–1519.

Kaiqiang Song, Bingqing Wang, Zhe Feng, Ren Liu, and Fei Liu. 2020. Controlling the Amount of Verbatim Copying in Abstractive Summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8902–8909.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning Word Representations with Regularization from Prior Knowledge. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 143–152.

Yan Song and Shuming Shi. 2018. Complementary Learning of Word Embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4368–4374.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 175–180.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*.

Yuanhe Tian, Guimin Chen, and Yan Song. 2021a. Aspect-based Sentiment Analysis with Type-aware Graph Convolutional Networks and Layer Ensemble. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2910–2922, Online.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021b. Dependency-driven Relation Extraction with Attentive Graph Convolutional Networks.

In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020a. Improving Biomedical Named Entity Recognition with Syntactic Information. *BMC Bioinformatics*, 21:1471–2105.

Yuanhe Tian, Yan Song, and Fei Xia. 2020b. Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph Attention Networks. *arXiv preprint arXiv:1710.10903*.

Yuhao Zhang, Daisy Yi Ding, Tianpei Qian, Christopher D Manning, and Curtis P Langlotz. 2018. Learning to Summarize Radiology Findings. *EMNLP 2018*, page 204.

Yuhao Zhang, Yuhui Zhang, Peng Qi, Christopher D Manning, and Curtis P Langlotz. 2020. Biomedical and Clinical English Model Packages in the Stanza Python NLP Library. *arXiv preprint arXiv:2007.14640*.

Chen Zheng and Parisa Kordjamshidi. 2020. SRLGRN: Semantic role labeling graph reasoning network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8881–8891, Online.

Generalized Supervised Attention for Text Generation*

Yixian Liu[1], **Liwen Zhang**[2], **Xinyu Zhang**[3], **Yong Jiang**, **Yue Zhang**[4], **Kewei Tu**[2]
[1]Interactive Entertainment Group, Tencent
[2]School of Information Science and Technology, ShanghaiTech University
[3]Language Technologies Institute, Carnegie Mellon University, USA
[4]School of Engineering, Westlake Universtiy
[4]Institute of Advanced Technology, Westlake Institute for Advanced Study
`easingliu@tencent.com xinyuzh2@cs.cmu.edu`
`{zhanglw1, tukw}@shanghaitech.edu.cn`
`zhangyue@westlake.edu.cn`

Abstract

The attention-based encoder-decoder framework is widely used in many natural language generation tasks. The attention mechanism builds alignments between target words and source items that facilitate text generation. Previous work proposes supervised attention that uses human knowledge to guide the attention mechanism to learn better alignments. However, well-designed supervision built from ideal alignments can be costly or even infeasible. In this paper, we build a Generalized Supervised Attention method (GSA) based on quasi alignments, which specify candidate sets of alignments and are much easier to obtain than ideal alignments. We design a Summation Cross-Entropy (SCE) loss and a Supervised Multiple Attention (SMA) structure to accommodate quasi alignments. Experiments on three text generation tasks demonstrate that GSA improves generation performance and is robust against errors in attention supervision.

1 Introduction

The encoder-decoder framework has been applied to various natural language generation (NLG) tasks, such as neural machine translation (Cho et al., 2014; Luong et al., 2015; Vaswani et al., 2017; Liu et al., 2016), text generation (Wiseman et al., 2017; Puduppully et al., 2019), text summarization (Liu and Lapata, 2019; Lin et al., 2018), image captioning (Anderson et al., 2018), dialogue systems (Liu et al., 2018), and so on. The attention mechanism (Bahdanau et al., 2015) plays a significant role in the framework, which automatically extracts the alignments between the target and the source for predicting the next target output. One disadvantage of the vanilla attention mechanisms is that the automatic weights do not necessarily encode prior knowledge, such as the alignments between input and output (Jain and Wallace, 2019). To alleviate this problem, supervised attention was considered (Liu et al., 2016; Mi et al., 2016; Kamigaito et al., 2017; Nguyen et al., 2018; Nguyen and Nguyen, 2018), which shows that human knowledge is helpful for guiding the learning process of attention models.

Previous work on supervised attention assumes access to ideal alignments. Unfortunately, obtaining ideal alignments is infeasible or extremely costly, for most NLG tasks. For example in Figure 1, for the AMR-to-text generation task, given the AMR graph for sentence "*From among them, pick out 50 for submission to an assessment committee to assess.*", the ideal alignment of the last word "*assess*" is node (10). As the names of (8) and (10) are the same, it is not easy to pick (10) exactly. On the other hand, it is much easier to obtain a candidate set containing both (8) and (10) and be rather confident that the ideal alignment is in the set. For different tasks, both EM-based algorithms (Brown et al., 1993; Pourdamghani et al., 2014) or rule-based methods (Flanigan et al., 2014) can be used to obtain such ambiguous alignments. However, little work has discussed making use of ambiguous labels for supervised attention.

We investigate the *generalized supervised attention* (GSA), where the supervision signal aligns a target word to multiple *possible* source items (named the *quasi alignment*), although only a subset of the items are the *true* alignment targets. The multiple source items are named *candidate set* of the quasi alignment. A generalized supervised attention framework is built for various text generation tasks with alignment relationships between target words and source items. One challenge for generalized supervised attention is that the standard Cross-Entropy (CE) loss (Liu et al., 2016)

---

[*] The work was done when Yixian Liu and Xinyu Zhang were students of Shanghaitech University. Kewei Tu is the corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4991–5003
August 1–6, 2021.

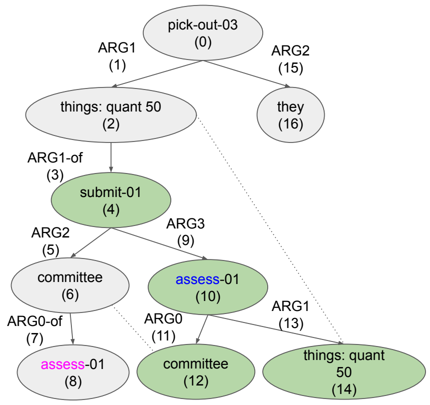

Target Sentence:

From among them, pick out 50 for submission to an assessment committee to assess.

Alignments of target word assess:

Ideal alignment	(10)
Quasi alignments	(8), (10)
Relevant alignments	(4), (10), (12), (14)

Figure 1: Example of an AMR graph and the alignments between the graph and the target sentence. Ideal alignment points to the most related source node to "assess". Quasi alignments point to the source nodes with the same name. Relevant alignments point to more source items with weak relation to "assess".

can be limited because it is not suitable for quasi alignments. We design a new loss function named Summation Cross-Entropy (SCE) to replace the Cross-Entropy loss given a set of quasi alignments. SCE considers multiple candidates as a whole and is more robust against spurious candidates than traditional CE.

Supervised attention and automatically learned attention can be complementary. In Figure 1, the relevant alignments (4), (10), (12), (14) are useful for predicting "*assess*", but such alignments cannot be captured by simple rules and may require human annotation in order to be used for attention supervision. It is therefore more practical to rely on automatic attention to uncover such alignments. To balance supervised attention and automatic attention, we design a Supervised Multiple Attention (SMA) module for GSA. In SMA, there are multiple attention channels with the same structure but different parameters. One of them is used for supervised attention and the others are used for pure automatic attention (named unsupervised attention below) that are not influenced by the attention supervision. SMA can be seen as an extension of multi-head attention introduced in the Transformer (Vaswani et al., 2017).

We evaluate GSA on three real-world tasks: data-to-text generation (Koncel-Kedziorski et al., 2019), AMR-to-text generation (Mager et al., 2020), and text summarization (Yan et al., 2020). The results demonstrate that our method improves the performance in general. We also examine the robustness of our method against alignment errors. Our code will be released at `https://github.com/LiuYixian/Supervised_attention`.

2 Related Work

Previous work (Liu et al., 2016; Mi et al., 2016; Kamigaito et al., 2017; Nguyen et al., 2018; Nguyen and Nguyen, 2018) have studied supervised attention. Their work is based on well-designed alignments. However, no work has considered ambiguous labels, which are practically more common. We study the quasi alignments as the attention supervision and design the Summation Cross-Entropy to deal with the ambiguity in quasi alignments.

Learning with ambiguous labels has been widely studied, in which the true label is not precisely annotated but in a candidate label set. In cross-lingual Part-of-Speech, annotations are derived for low resource languages from cross-language projection, which results in partial or uncertain labels. To solve this problem, Täckström et al. (2013) proposed a partially observed conditional random field (CRF) (Lafferty et al., 2001) method, Wisniewski et al. (2014) made a history-based model, and Buys and Botha (2016) proposed an HMM-based model. SCE is designed for training attention weights using ambiguous labels. Xu et al. (2020) also study learning from ambiguous labels (called *partial label learning*) in classification tasks. Their method is based on constructing similar and dissimilar pairs of samples. However, supervised attention is not a traditional classification problem. The label spaces are various in different samples, making it difficult to construct similar pairs. Thus, the method is not suitable for GSA.

3 Basic Model

3.1 Encoder-decoder Model with Attention

Encoder-decoder models, including RNN models (Cho et al., 2014; Luong et al., 2015) and the Transformer model (Vaswani et al., 2017), are used for a variety of NLP tasks. The encoder extracts information from the source data into a memory bank, and the decoder makes use of the memory bank to generate the target sentence.

Let $\mathbf{x} = \{x_1, \dots, x_m\}$ denote a sequence of source items, and $\mathbf{y} = \{y_1, \dots, y_n\}$ a target sentence. The encoder converts the source data $\mathbf{x}$ into a memory bank $\mathbf{H} = \{\mathbf{h}_1, \dots, \mathbf{h}_m\}$, where each vector $\mathbf{h}_i$ represents the contextual embedding of x_i. At the t-th time step of the decoder, the model obtains the feature vector $\mathbf{s}_t$. The meaning of $\mathbf{s}_t$ varies in different decoders. For an RNN decoder, it is the hidden state of the RNN at time t. The attention mechanism computes the contextual feature $\mathbf{c}_t$ of $\mathbf{s}_t$ over $\mathbf{H}$, which is used to predict the next target word.

The objective function of generation is the negative log conditional likelihood loss:

$$\ell(\mathbf{x}, \mathbf{y}) = -\log P(\mathbf{y}|\mathbf{x}) \tag{1}$$

3.2 Supervised Attention (SA) with Cross-Entropy (CE) Loss

Supervised attention (SA) was first introduced by Liu et al. (2016) and Mi et al. (2016) for neural machine translation. They obtain attention supervision between source and target words with off-the-shelf aligners. SA is a multi-task learning approach, where the objective function is the summation of the loss of sequence generation (*generation loss*) and the disagreement between attention distribution and attention supervision (*attention loss*) as follows:

$$\text{Ł} = \ell(\mathbf{x}, \mathbf{y}) + \lambda \sum_t \Delta(\boldsymbol{\alpha}_t, \hat{\boldsymbol{\alpha}}_t) \tag{2}$$

where $\boldsymbol{\alpha}_t$ is the computed attention and $\hat{\boldsymbol{\alpha}}_t$ is the attention supervision.

$\ell(\mathbf{x}, \mathbf{y})$ is the generation loss in Eq. (1), and λ is a positive hyper-parameter that balances the two losses. $\hat{\boldsymbol{\alpha}}_t$ is the target attention distribution and Δ measures the disagreement between the attention distribution and the target distribution. Liu et al. (2016) assume that every target word is aligned to at least one source word. If a target word is aligned to k source words, the corresponding elements in $\hat{\boldsymbol{\alpha}}_t$ are $\frac{1}{k}$ and the other elements are 0. They apply the Cross-Entropy loss as the attention loss function:

$$\Delta(\boldsymbol{\alpha}_i, \hat{\boldsymbol{\alpha}}_i) = -\sum_{j=1}^{m} \hat{\alpha}_{i,j} \times \log \alpha_{i,j} \tag{3}$$

4 Generalized Supervised Attention (GSA)

The overall architecture of GSA is shown in Figure 2. We will first introduce quasi alignments, and introduce a Summation Cross-Entropy loss function and a Supervised Multiple Attention structure.

4.1 Quasi Alignments

We consider *quasi alignments* as shown in Figure 1, in which a target word is allowed to be aligned to a candidate set in the source items, although only a subset of the candidates are the *true* alignment targets. The supervision signal provided by the quasi alignments is $\bar{\boldsymbol{\alpha}}_t = \{\bar{\alpha}_{t,1}, \dots, \bar{\alpha}_{t,m}\}$, where $\bar{\alpha}_{t,i} = 1$ if x_i and y_t should be aligned with considerable probability. If $|\bar{\boldsymbol{\alpha}}_t| = 1$ and $\hat{\boldsymbol{\alpha}}_t$ is a one-hot alignment vector, y_t is only aligned to x_i. If $|\bar{\boldsymbol{\alpha}}_t| > 1$, $\hat{\boldsymbol{\alpha}}_t$ expresses a discrete uniform distribution over the candidate set. Such candidate items usually include some irrelevant items that should not be aligned to y_t, but it is costly to pick out the correct subset from these candidates. Therefore, we retain all these candidates and expect the training process to determine the better alignment automatically. If $|\bar{\boldsymbol{\alpha}}_t| = 0$, no item is found for y_t.

In our experiments, we obtain the quasi alignments using simple rule-based methods, which differ for different tasks, as will be discussed in Section 5.

4.2 Summation Cross-Entropy (SCE) Loss

Quasi alignments form candidate sets containing the potential aligned source items but do not indicate the true ones among them. Intuitively, we want our attention loss to penalize attention probabilities outside the candidate set but allow an arbitrary attention distribution within the set. To this end, we design the SCE loss function to maximize the total of attention probabilities in the candidate set.

$$\Delta(\boldsymbol{\alpha}_t, \bar{\boldsymbol{\alpha}}_t) = \begin{cases} 0, \text{ if } \bar{\boldsymbol{\alpha}}_t = \mathbf{0} \\ -\log(\langle \boldsymbol{\alpha}_t, \bar{\boldsymbol{\alpha}}_t \rangle), \text{ else} \end{cases} \tag{4}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product. The SCE loss is the negative logarithm of the likelihood summation of all candidate items.

Theoretically, SCE loss can be derived from a generative model. Assume that one target word should be aligned to only one true source item. For the t-th target word, we define a random variable as the true aligned source item, $P(z_t = i) = \alpha_{t,i}$. Given z_t, we re-define the candidate set as

$$\bar{\boldsymbol{\alpha}}_t = \{\bar{\alpha}_{t,1}, \dots, \bar{\alpha}_{t,m}\}, \text{ where } \bar{\alpha}_{t,z_t} = 1. \tag{5}$$

In this way, the candidate set contains z_t. Considering that z_t is a hidden variable, the likelihood

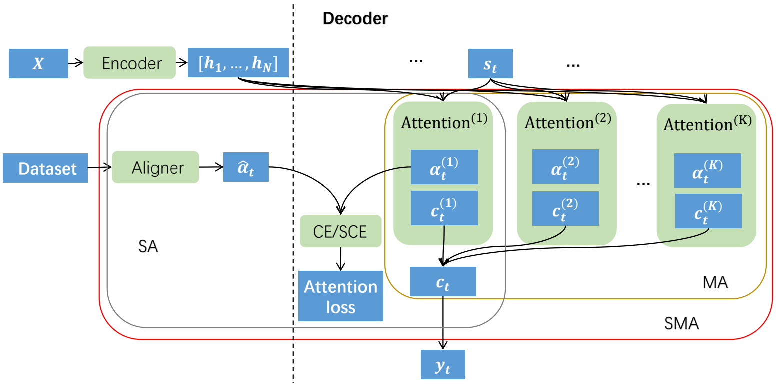

Figure 2: The overall architecture of GSA.

of the candidate set can be defined as

$$P(\bar{\boldsymbol{\alpha}}_t) = \sum_i P(z_t = i) P(\bar{\boldsymbol{\alpha}}_t | z_t = i) \quad (6)$$

$$= \sum_i \alpha_{t,i} \mathbb{I}(\bar{\alpha}_{t,i} = 1) \quad (7)$$

$$= \langle \boldsymbol{\alpha}_t, \bar{\boldsymbol{\alpha}}_t \rangle \,. \quad (8)$$

We assume that it is a certain event that we obtain a candidate set containing the true alignment given z_t. $P(\bar{\boldsymbol{\alpha}}_t | z_t)$ is a distribution over candidate sets, in which only the candidate set that contains all the words identical to x_{z_t} has probability 1 and all the other candidate sets have probability 0. Thus, $P(\bar{\boldsymbol{\alpha}}_t | z_t = i) = \mathbb{I}(\bar{\alpha}_{t,i} = 1)$. Therefore, optimizing the SCE loss is to maximize the likelihood of the candidate set.

By comparing Eq. (3) and Eq. (4), CE loss optimizes the summation of log-likelihood of target alignments, while SCE loss optimizes the log of summation of the likelihood of target alignments. If a target word is not aligned to any source items, the attention loss is 0. If it is aligned to only one source item, SCE reduces to CE (de Boer et al., 2005). If it is aligned to multiple items, then the SCE loss penalizes the attention probabilities outside the candidate set and uniformly increases the attention probabilities within the set. Note that this behavior is different from that of CE, which would encourage uniform attention over all the candidates in the set and hence produce different updates to the attention probabilities of different candidates during training.

4.3 Multiple Attention (MA) and Supervised Multiple Attention (SMA)

The motivation of supervised attention is to incorporate prior knowledge of alignments between source and target items into the attention mechanism. One problem of supervised attention is that alignments are typically established between similar items, but ideally, the decoder should also attend to some other informative source items (Figure 1), which are not necessarily similar to the target word. Besides, the automatic aligner may make errors and align the target word to irrelevant source items. Therefore, the unsupervised automatic attention mechanism is still a useful supplement to supervised attention.

We define a multiple channel attention (MA) structure, which is closely related to multi-head attention (Vaswani et al., 2017). There are K attention channels with the same structure but different parameters in MA, which work concurrently and their output contextual feature vector are combined into one contextual feature vector.

$$\mathbf{c}_t^{(k)} = \text{attn}(\mathbf{s}_t, \mathbf{H}; \theta_{\mathbf{k}}) \text{ for } k = 1, \ldots, K \quad (9)$$

$$\mathbf{c}_t = G(c^{(1)}, \ldots, c^{(K)}), \quad (10)$$

where G is a combination function. Multi-head attention (Vaswani et al., 2017) can be regarded as a special case of the MA structure. One head of multi-head attention is an attention channel in Eq. (9). The contextual features are combined by a stacking action followed with a linear function. We do not use the standard multi-head attention because the structure of multi-head attention is strict and it is not proper for all generation tasks.

To balance supervised attention and unsupervised attention, SMA has the same structure as MA, and we compute the attention loss of the first channel only, leaving the other channels still unsupervised. The objective of SMA model is:

$$Ł = \ell(\mathbf{x}, \mathbf{y}) + \lambda \sum_t \Delta(\boldsymbol{\alpha}_t^{(1)}, \hat{\boldsymbol{\alpha}}_t) \quad (11)$$

A proposed generation model with attention can be easily modified by the SMA structure. If the original attention is one-headed, we add a new attention channel with the same structure parallel to the original one, and compute the attention loss for the new attention. The contextual features of the two attention channels are averaged in Eq. (10). If the original attention is multi-headed, we do not change the structure, and compute supervised attention loss for the first head.

5 Experiments

We apply GSA to three tasks: data-to-text generation, AMR-to-text generation, and text summarization. For each task, we choose one of the best published approaches as our basic model and modify it with GSA. In the three tasks, the relations between the source and the target are diverse. For text summarization, the source items contain more information than the target words. For data-to-text generation, the source items only contain key contents. For AMR-to-text generation, the source and the target contain the same information. We report the details of model structures and hyper-parameters in the appendix.

5.1 Data-to-text Generation

Task and Model : We consider the Abstract GENeration DAtaset (AGENDA) (Ammar et al., 2018), which contains pairs of a literature abstract and a knowledge graph extracted from the abstract. The nodes in the knowledge graphs are entity types, such as “Task” and “Method”. The edges are the relations between different entities, including “COMPARE”, “PART-OF”, and so on. We use the training, development, and test splits of 38,720/1000/1000, as Ammar et al. (2018) does.

We use GraphWriter[1] (Koncel-Kedziorski et al., 2019) on this task. The encoder of this model is a graph transformer and the decoder is an RNN decoder with attention and copying mechanism. More detail is introduced in the appendix.

Aligner: The source items of this task include entities and relations, as shown in Figure 3. We use our string matching aligner to extract the alignments from target words to the source entities and extend our aligner for the alignments of relations, such as aligning target words “*use*” and “*apply*” to source relation “USED-FOR”. For the details of the aligner for source relations, please refer to the appendix.

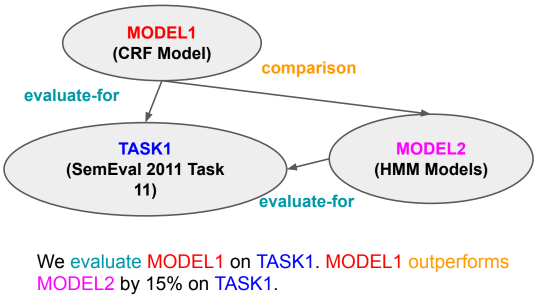

Figure 3: Example of alignments of the AGENDA dataset. The upper is part of the graph data and the lower is the corresponding sentence. The words with the same color should have quasi alignments.

Experimental Settings: We experiment with 4 different approaches in this experiment: *unsupervised attention* (UA), SA-CE, SA-SCE, SMA-SCE. The unsupervised approach means the original method without supervision on attention. As the decoder applies multi-head attention, we design the SA approach, in which the attention distributions of all heads are averaged to compute the attention loss. In this way, we consider the multi-head attention as a supervised attention channel. The SMA approach is designed as in Section 4.3, in which only the first head is a supervised attention channel. In SCE and CE approaches, we used SCE and CE loss function to supervise the attention, respectively.

5.2 AMR-to-text Generation

Task and Model : Abstract meaning representation (AMR) (Banarescu et al., 2013) is a semantic graph representation that is independent of the syntactic realization of a sentence. In the graph, nodes represent concepts and edges represent semantic relations between the concepts. AMR-to-text generation is to generate sentences from AMR graphs. We use the AMR dataset LDC2015E86, which contains 16,833 training samples, 1368 development samples, and 1371 test samples.

We use the model[2] of Mager et al. (2020) on this task, which is a GPT-2 (Radford et al., 2019) model with fine-tuning.

Aligner: We apply lemma matching to build the attention supervision as shown in Figure 1. There is a quasi alignment between a source item and target

---

[1] `https://github.com/rikdz/GraphWriter`

[2] `https://github.com/IBM/GPT-too-AMR2text`

Approach	Data-to-text			AMR-to-text		Summarization		
	BLEU	METEOR	ROUGE-L	BLEU	chrF++	ROUGE-1	ROUGE-2	ROUGE-L
UA	14.30	18.80	27.94	28.39	58.16	43.85	20.89	40.93
SA-CE	15.21	19.47	28.31	28.67	57.72	44.07	21.13	41.14
SA-SCE	15.49	19.80	28.62	29.03	58.44	**44.16**	**21.28**	**41.22**
SMA-SCE	**15.51**	**19.88**	**29.00**	**29.30**	**58.89**	43.9	21.09	40.91

Table 1: Main test result of GSA.

word if they have the same lemma[3].

Experimental Settings: We experiment with 4 different approaches: UA, SA-CE, SA-SCE, SMA-SCE. We apply GSA to the multi-head attention in the last Transformer layer of the decoder.

5.3 Text Summarization

Task and Model : We use the CNN/DailyMail dataset (Nallapati et al., 2016). This dataset contains 287,226 training samples, 13,368 validation samples, and 11,490 test samples. We use ProphetNet (Yan et al., 2020) on this task, which builds a pre-training and fine-tuning method for text generation. Both the encoder and the decoder are Transformers. ProphetNet is pre-trained on a large-scale dataset (160GB).

Aligner: We obtain the quasi alignments with lemma matching as in Section 5.2. A target word is aligned to a source item if they have the same lemma. Some words, such as "*is*" and "*do*", appear very frequently and are likely to cause wrong alignments. We use the inverse document frequency (IDF) (Robertson, 2004) scores to downweight these words. More details about IDF applied here are shown in the appendix.

Experimental Settings: The model has a Transformer decoder. We set the experiments similarly to Section 5.2. The basic model is proposed by Yan et al. (2020). We cannot fully reproduce their reported result (ROUGE-1/2/L of 44.2/21.17/41.30) by running their public model[4]. Thus, we report our results.

5.4 Main Results

The test results of GSA on the three tasks are shown in Table 1. For data-to-text generation, the basic model with unsupervised attention (UA) gives 14.30 BLEU, 18.80 METEOR, and 27.94 ROUGE-L. All the supervised attention approaches outperform UA. SCE outperforms CE and SMA outperforms SA. SMA-SCE achieves the best performance in BLEU, METEOR, and ROUGE-L. For AMR-to-text generation, the basic model (UA) gives a BLEU of 28.39 and a chrF++ of 58.16. All the supervised attention approaches outperform the unsupervised attention approach, demonstrating the strength of supervised attention. SMA and SCE approaches obtain higher BLEU scores compared with SA and CE, respectively. SMA-SCE achieves the best performance. For text summarization, the basic model (UA) gives a ROUGE-2 of 20.89. All the supervised attention approaches beat UA. SCE outperforms the CE. However, different from the above two tasks, the SA approaches are better than SMA. SA-SCE gives the best performance. We discuss the variation in performance of SMA compared with SA in the following.

Approach	A.C.	M.C.	M.S.
Data2text	13.21%	9.41%	2
AMR2text	46.64%	27.11%	2.93
Text Sum.	83.53%	78.10%	8.98

Table 2: Alignment coverage (A.C.), multi-alignment coverage (M.C.), and average multi-alignment size (M.S.) of attention supervision.

Approach	SA-SCE	SA-CE
variance	0.1332	0.0561
entropy	0.4214	0.8364

Table 3: Normalized variance and entropy of attention probability over the candidate set.

To analyze the variations of the performance of GSA in different tasks, we compute the alignment coverage (A.C.), multi-alignment coverage (M.C.), and average multi-alignment size (M.S.) of different tasks (Table 2). A.C. is the percentage of target words with at least one alignment. M.C. is the percentage of target words with at least two alignments over all the aligned target words. M.S. is the average number of alignments of target words with at

---

[3] We apply Pattern (Smedt and Daelemans, 2012) to get the lemma of a word in `https://github.com/clips/pattern`.

[4] `https://github.com/microsoft/ProphetNet`.

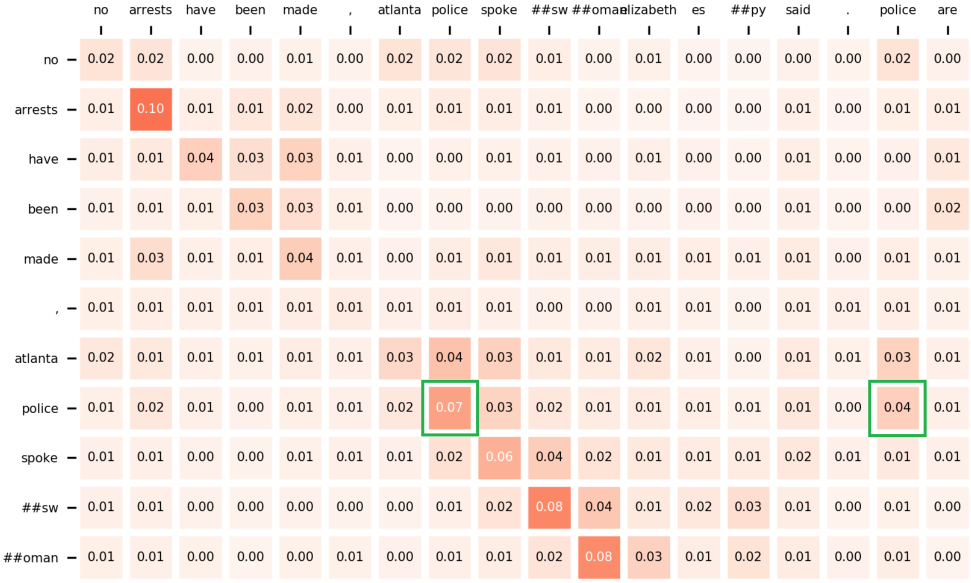

	no	arrests	have	been	made	,	atlanta	police	spoke	##sw	##oman	elizabeth	es	##py	said	.	police	are
no	0.02	0.02	0.00	0.00	0.01	0.00	0.02	0.02	0.02	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.00
arrests	0.01	0.10	0.01	0.01	0.02	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01
have	0.01	0.01	0.04	0.03	0.03	0.01	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.01
been	0.01	0.01	0.01	0.03	0.03	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.02
made	0.01	0.03	0.01	0.01	0.04	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01
,	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.01	0.01	0.00	0.01	0.01	0.01	0.01
atlanta	0.02	0.01	0.01	0.01	0.01	0.01	0.03	0.04	0.03	0.01	0.01	0.02	0.01	0.00	0.01	0.01	0.03	0.01
police	0.01	0.02	0.01	0.00	0.01	0.01	0.02	0.07	0.03	0.02	0.01	0.01	0.01	0.01	0.01	0.00	0.04	0.01
spoke	0.01	0.01	0.00	0.00	0.01	0.01	0.01	0.02	0.06	0.04	0.02	0.01	0.01	0.01	0.02	0.01	0.01	0.01
##sw	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.08	0.04	0.01	0.02	0.03	0.01	0.00	0.01	0.00
##oman	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.02	0.08	0.03	0.01	0.02	0.01	0.00	0.01	0.00

(a) CE approach attention

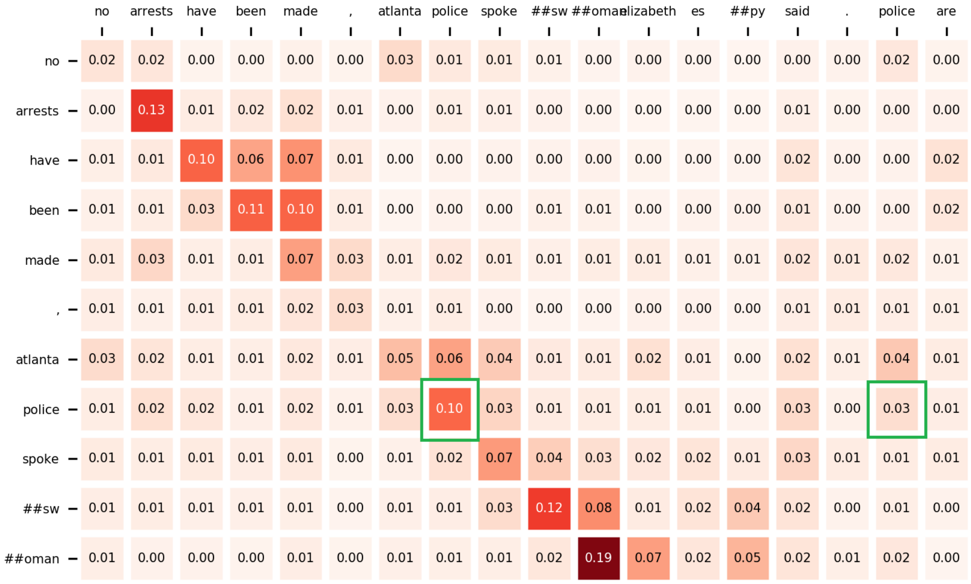

	no	arrests	have	been	made	,	atlanta	police	spoke	##sw	##oman	elizabeth	es	##py	said	.	police	are
no	0.02	0.02	0.00	0.00	0.00	0.00	0.03	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
arrests	0.00	0.13	0.01	0.02	0.02	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
have	0.01	0.01	0.10	0.06	0.07	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02
been	0.01	0.01	0.03	0.11	0.10	0.01	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.02
made	0.01	0.03	0.01	0.01	0.07	0.03	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.01
,	0.01	0.01	0.01	0.01	0.02	0.03	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01
atlanta	0.03	0.02	0.01	0.01	0.02	0.01	0.05	0.06	0.04	0.01	0.01	0.02	0.01	0.00	0.02	0.01	0.04	0.01
police	0.01	0.02	0.02	0.01	0.02	0.01	0.03	0.10	0.03	0.01	0.01	0.01	0.01	0.00	0.03	0.00	0.03	0.01
spoke	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.02	0.07	0.04	0.03	0.02	0.02	0.01	0.03	0.01	0.01	0.01
##sw	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.03	0.12	0.08	0.01	0.02	0.04	0.02	0.00	0.01	0.00
##oman	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.02	0.19	0.07	0.02	0.05	0.02	0.01	0.02	0.00

(b) SCE approach attention

Figure 4: Comparison of the test attention of SCE structure.

least two alignments.

For data-to-text and AMR-to-text generation, SMA outperforms SA. On the other hand, SA performs better than SMA for text summarization. One possible reason is that the summarization dataset has much higher alignment coverage and multi-alignment coverage and the alignment accuracy may also be higher; consequently, supervised attention works so well that automatic attention becomes unnecessary or even distracting.

5.5 Significance Test

To assess the evidence of significance, we perform significance tests on GSA. The p-value is calculated using the one-tailed sign test with bootstrap resampling on the test set of all three tasks following Chollampatt et al. (2019):

- For data-to-text, we compare the Rouge-L score of SMA-SCE to the result of SA-CE.
- For AMR-to-text, we compare the BLEU score of SMA-SCE to the result of SA-CE.
- For summarization, we compare the Rouge-L score of SA-SCE to the result of SA-CE.

The p-value results are shown in Table 4, which show that the improvements are significant.

Task	Data-to-Text	Amr-to-Text	Summarization
P-value	6.5489e-12	5.5795e-10	3.925e-5

Table 4: Significance test for GSA.

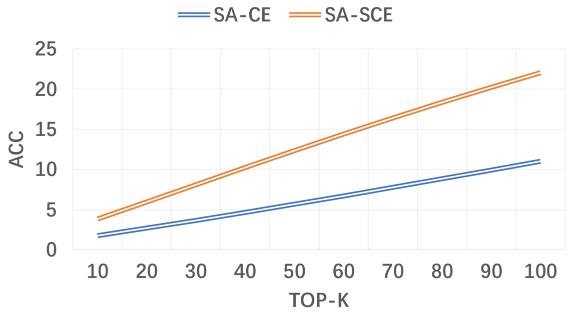

Figure 5: Top-K accuracy of SA-CE and SA-SCE in text generation.

5.6 SCE Analysis

Variance We compute the variance of attention probability in the candidate set for the text summarization task. For every generated token, we get the candidate set containing the input tokens with the same lemma as the generated token. If the candidate set contains more than one token, we compute the normalized variance and entropy of the attention scores in the candidate set. Normalized variance means that we divide every attention score by their summation and compute the variance of the normalized attention scores. Then we average the values of normalized variance and entropy in the test set. As shown in Table 3, the normalized attention variance of SCE is larger than that of CE and the entropy of SCE is smaller. It implies that CE homogenizes the attentions over the candidate set, while SCE concentrates the attentions on certain tokens. It echos Section 4.2 that CE encourages uniform attention while SCE fixes the issue.

Attention Accuracy We design an automatic evaluation method to investigate whether our SCE method can find the *correct* alignment from the quasi alignment set in an unsupervised way. For a token whose length is greater than 5[5] in the generated result, if it is matched (by lemma matching) with more than one input token, we study the generated token and the candidate set containing the matched input tokens. Specifically, we consider the local context window of length 7 around the tokens in the candidate set. The correct alignment is defined as the input token whose context window shares the most tokens with the same window around the generated token. We find the alignment selected by this automatic method almost fits the human judgment. An example is shown in Table 5.

Generated	wayne was in atlanta for a performance
Matching 1	early sunday in atlanta . no one
Matching 2	been made , atlanta police spokes woman
Matching 3	parking lot in atlanta ' s buckhead
Matching 4	wayne was in atlanta for a performance

Table 5: Automatic method to find the correct alignment.

	SA-CE	SA-SCE	SMA-SCE
BLEU	31.01	**31.19**	30.88
chrF++	59.64	**60.08**	59.86

Table 6: GSA annotated by ISI aligner for AMR-to-text generation.

The *top-K accuracy* indicates the rate that the attention score corresponding to the correct alignment is among the largest K scores. Figure 5 shows the top-K accuracy of CE and SCE for text summarization. We can find that our SA-SCE method gets higher top-K accuracy than SA-CE. That means our SCE method could find the correct alignment token and pay more attention to it without supervision.

Case Study An example of text summarization is shown in Figure 4. The figure displays a fragment of a test output sentence and the corresponding source fragment. The abscissas indicate the input text, and the ordinates denote the output summary. Figure 4(a) shows the attention of the CE approach, and 4(b) shows the attention of SCE. Both SCE and CE select the correct alignment in this example. However, the SCE approach provides higher attention probability on the correct alignment. As shown in Table 2, most output words can be flexibly aligned to more than one source word. Consider the attention probabilities of the word "*police*" framed by green squares. For one output word, there are two similar input "*police*" shown in the figure, with the first one being correct and the other one being incorrect. CE gives a probability of 0.07 for the correct alignment and 0.04 for the incorrect one. SCE approaches give the probability of 0.1 for the correct alignment and 0.03 for the incorrect one. According to section 4.2, SCE loss reduces the effect of incorrect alignments in the candidate set, which promotes the true source word.

5.7 More Powerful Supervision

In the main experiments, we apply a simple and general-purpose string matching aligner. For certain tasks, there are more powerful aligners available. To study the impact of better aligners, we in-

---

[5] In order to filter out high-frequency words like 'a', 'the', and 'and'.

vestigate the performance of GSA on AMR-to-text generation with the ISI aligner proposed by Pourdamghani et al. (2014), which is specially designed for the AMR-to-text task. The result is shown in Table 6. The improvement over the result in Table 1 proves that a more accurate aligner helps the supervised attention method for text generation. Besides, we can also find the result of SA-SCE better than that of SA-CE, which shows that SCE also works well while using a more accurate aligner.

We also analyze the alignments by the ISI aligner following the metrics of Table 2. The alignment coverage is 64.75%; the multi-alignment coverage is 52.17%; and the multi-alignment size is 3.11. It shows that the better aligner also produces ambiguous alignments. Therefore, SCE outperforms CE.

5.8 Robustness Analysis

We test the robustness of GSA by corrupting the attention supervision by changing correct alignments into incorrect ones. For every N target words with alignments, we change the alignments of one target word to a random and different source item. Then, we test GSA on data-to-text and AMR-to-text tasks based on the corrupted attention supervision. N ranges in $\{2, 3, 5, 10, 20\}$, which correspond to error rates of $\{50\%, 33\%, 20\%, 10\%, 5\%\}$, respectively.

The results are shown in Table 7. For AMR-to-text generation, we test the SMA-SCE approach. We observe that the supervised attention approach with a 20% error rate is still better than UA. Only with a 33% error rate does the supervised attention approach underperform the unsupervised attention. For data-to-text generation, we test the SMA-SCE approach. We observe that the SMA-SCE approach with even a 33% error rate is still better than UA. These results demonstrate the robustness of our supervised attention. On the other hand, in both experiments, the performance almost always decreases with more errors, demonstrating the importance of correct supervision.

Although GSA is shown to be robust to alignment errors, an overly high error rate would prevent the attention mechanism from finding the true alignments and make the supervised attention approaches worse than UA. Thus, reducing the mistake error rate is the most important when designing the aligner. More analyses about errors in attention supervision are in the appendix.

Error Rate	Data-to-text			AMR-to-text
	BLEU	METEOR	ROUGE-L	BLEU
0%	15.51	19.88	29.00	29.30
5%	14.93	19.68	28.44	29.01
10%	14.49	19.00	28.49	28.89
20%	14.36	18.99	28.41	28.80
33%	14.30	19.18	28.20	28.08
UA	14.30	18.80	27.94	28.39

Table 7: Robustness analysis.

6 Conclusion

We studied generalized supervised attention (GSA) for text generation tasks, considering quasi alignments instead of true alignments, which are much more difficult to obtain in practice. A Summation Cross-Entropy (SCE) loss function was designed to deal with quasi alignments, and a Supervised Multiple Attention (SMA) structure was used to balance supervised attention and unsupervised attention. Experiments on three generation tasks demonstrated that generalized supervised attention produces competitive results and is robust against errors in attention supervision.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61976139).

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew E. Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*, pages 84–91.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6077–6086.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations,* *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, pages 178–186.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals OR*, 134(1):19–67.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguistics*, 19(2):263–311.

Jan Buys and Jan A. Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Shamil Chollampatt, Weiqi Wang, and Hwee Tou Ng. 2019. Cross-sentence grammatical error correction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 435–445, Florence, Italy. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1426–1436. The Association for Computer Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3543–3556. Association for Computational Linguistics.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 7–12.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2284–2293.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 163–169.

Lemao Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3093–3102.

Shuman Liu, Hongshen Chen, Zhaochun Ren, Yang Feng, Qun Liu, and Dawei Yin. 2018. Knowledge diffusion for neural dialogue generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1489–1498.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Manuel Mager, Ramón Fernández Astudillo, Tahira Naseem, Md. Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. *CoRR*, abs/2005.09123.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2283–2288.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290.

Minh Nguyen and Thien Nguyen. 2018. Who is killed by police: Introducing supervised attention for hierarchical lstms. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2277–2287.

Minh Nguyen, Toan Nguyen, and Thien Huu Nguyen. 2018. A deep learning model with hierarchical lstms and supervised attention for anti-phishing. *CoRR*, abs/1805.01554.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 425–429.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2023–2035.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520.

Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *J. Mach. Learn. Res.*, 13:2063–2067.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan T. McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Trans. Assoc. Comput. Linguistics*, 1:1–12.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1779–1785. ACL.

Shuang Xu, Min Yang, Yu Zhou, Ruirui Zheng, Wenpeng Liu, and Jianjun He. 2020. Partial label metric learning by collapsing classes. *INTERNATIONAL JOURNAL OF MACHINE LEARNING AND CYBERNETICS*.

Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *CoRR*, abs/2001.04063.

Appendix

A Model detail

A.1 Graph-to-text generation

In the basic model, multi-head attention is utilized to compute the contextual feature in every time step, over all the source items, including entities and relations. The copying mechanism uses basic attention just over the entities with different parameters from the multi-head attention.

The graph transformer encoder has 6 block layers. The number of attention heads is 4. The dimensions of embedding and hidden states are 500. The decoder is a one-layer LSTM recurrent networks. The decoder attention is multi-headed. In the decoding process, beam search with beam size of 4 is applied. In GSA model, the supervised attention weight is 0.5. The weight is tuned on the validation set by the BLEU score.

A.2 AMR-to-text generation

We use GPT-2 medium as the baseline generation model. It has 24 transformer block layers and 16 attention heads. The dimensions of word embeddings and hidden states are 1024. In the decoding process, the beam size is 15. The supervised attention weight for the lemma matching aligner is 0.001, and for the ISI aligner is 0.01. The weight is tuned on the validation set by the BLEU score.

A.3 Text Summarization

The baseline model is based on an encoder-decoder structure with Transformer. It has 12 block layers with 1024 hidden sizes. The beam size of decoding is 5. The supervised attention weight is 0.1. The weight is tuned on the validation set by the BLEU score.

B Aligner for Relations in Graph-to-text Generation

There are seven different relations as mentioned in the main paper. A relation can be represented by different words in the target text. For example, "use" and "apply" both suggest the relation "USED-FOR". We build a corresponding keyword list (shown in Table 8) for each relation. In the source data, each relation is of the form "a-R-b", where "a" and "b" are two entities, and "R" is the relation type. To find the alignments, we first look for the cooccurrence of "a" and "b" in the abstract with the shortest distance between them. Then, we examine the words between "a" and "b" as well as four preceding words and align the words that appear in the corresponding keyword list to both the forward and backward directions of that relation.

C IDF Score in Text Summarization

The IDF score of a word w is computed as:

$$\mathrm{IDF}(w) = -\log \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}(w \in X^{(i)}) \qquad (12)$$

where $\mathbb{I}(\cdot)$ is the indicator function, M is the number of training samples, and $X^{(i)}$ is the target sentence of the i-th sample in the training set.

In the training step, the IDF scores of target words are used to downweight the attention loss:

$$\text{Ł} = \mathrm{LOSS}(\mathbf{x}, \mathbf{y}) + \lambda \sum_{t} \big(\mathrm{IDF}(w_t) \cdot \Delta(\boldsymbol{\alpha}_t, \hat{\boldsymbol{\alpha}}_t)\big), \qquad (13)$$

where w_t is the t-th word in sentence $\mathbf{y}$.

In the loss function, the attention loss of a target word is scaled by its IDF score. The IDF scores are only applied in this experiment because the alignments of these high-frequency words are rare in previous experiments.

D Alignment Error Analysis

The performance of supervised attention is influenced by the quality of the aligner. There are three types of alignment errors.

- Missing: a target word is not aligned to any source item.
- Redundancy: a target word is aligned not only to the correct source items but also to irrelevant items.
- Mistake: a target word is only aligned to some irrelevant items but not aligned to correct source items.

Missing errors reduce the alignment coverage over the target sentences. They decrease the number of target words that receive attention supervision but will not make supervised attention worse than the unsupervised baseline.

Redundancy errors, which are related to the candidate set from the flexible alignments, are handled by our SCE loss. In the worst case, a target word is aligned to all the source items, and the attention loss of this word becomes 0, resulting in no supervision. Thus, redundancy errors will not make supervised attention worse than the unsupervised baseline either. A case study is provided in the appendix showing that our method is not confused by the redundancy errors.

We empirically analyze mistake errors in section 4.5. Although supervised attention is shown to be robust to mistake errors, an overly high error rate would prevent the attention mechanism from finding the correct alignments and make the supervised attention approaches worse than the baseline. Thus, reducing the mistake error rate is the most important when designing the aligner.

Relation	Keywords
USED-FOR	“present”, “propose”, “proposes”, “proposed”, “use”, “used”, “apply”, “applied”, “application”, “applications”, “exploit”, “introduce”, “improve”, “improves”, “for”, “learned”, “obtained”, “derived”, “use”, “uses”, “using”, “based”, “exploiting”
CONJUNCTION	“addition”, “versus”
FEATURE-OF	“about”, “feature”, “stand”, “denote”
PART-OF	“incorporate”, “incorporating”, “integrate”, “integrating”, “incorporates”, “include”, “includes”, “composed”, “combines”, “combining”, “consist”, “consists”, “consisting”, “incorporate”, “incorporates”, “incorporating”, “integrate”, “integrates”, “integrating”, “contain”, “contains”, “containing”
COMPARE	“outperform”, “outperforms”, “compare”, “compared”, “more”, “than”, “outperform”, “outperforms”, “compared”
EVALUATE-FOR	“experiments”, “improvements”, “evaluated”, “improve”, “improves”
HYPONYM-OF	“such”, “including”, “namely”, “called”, “like”, “named”

Table 8: Keyword lists used in the aligner for relations in graph-to-text generation.

Uncertainty Aware Review Hallucination for Science Article Classification

Korbinian Friedl[1,*], **Georgios Rizos**[1,*], **Lukas Stappen**[2], **Madina Hasan**[3],
Lucia Specia[1], **Thomas Hain**[3], **Björn W. Schuller**[1,2]
[1]GLAM – Group on Language, Audio, & Music, Imperial College London, UK
[2]EIHW – Chair of Embedded Intelligence for Health Care and Wellbeing,
University of Augsburg, Germany
[3]SpandH – Speech and Hearing Research Group, University of Sheffield, UK
`friedlkorbinian@gmail.com, gr912@ic.ac.uk`

Abstract

The high subjectivity and costs inherent in peer reviewing have recently motivated the preliminary design of machine learning-based acceptance decision methods. However, such approaches are limited in that they: a) do not explore the usage of both the reviewer and area chair recommendations, b) do not explicitly model subjectivity on a per submission basis, and c) are not applicable in realistic settings, by assuming that review texts are available at test time, when these are exactly the inputs that should be considered to be missing in this application. We propose to utilise methods that model the aleatory uncertainty of the submissions, while also exploring different loss importance interpolations between area chair and reviewers' recommendations. We also propose a modality hallucination approach to impute review representations at test time, providing the first realistic evaluation framework for this challenging task.

1 Introduction

An analysis (Langford and Guzdial, 2015) of the NeurIPS 2014 experiment shows that 60% of the selected accepted papers were rejected by a second, independent review committee. Such significant reviewer disagreement makes the task of the area chair harder, and may even invite questioning of their decision. Software tools have been piloted in an effort to aid the human reviewers with a computational recommendation on aspects like absence of bias and proper statistical reporting in scientific submissions (Sizo et al., 2019).

Natural Language Understanding (NLU) could also offer decision *support* to the area chair, as argued in (Ghosal et al., 2019; Stappen et al., 2020). Such systems jointly model the entire or part of the article and one (Kang et al., 2018; Wang and Wan, 2018; Ghosal et al., 2019), or a variable number of potentially contradicting reviews (Stappen et al., 2020). We adopt the latter, review-aggregating approach, that resembles the editorial process more.

1.1 Contributions

In this short paper, we offer solutions to three particularities of this task that the above approaches do not address: a) Often, the recommendations given by the area chair and the reviewers are in disagreement. Whereas previous studies have used either the former (Kang et al., 2018; Wang and Wan, 2018; Ghosal et al., 2019) or a soft label average of the latter (Stappen et al., 2020) for supervision, we show that both signals comprise complementary information. b) Whereas soft labels de-emphasise subjective articles with disagreeing reviews during training (Stappen et al., 2020), we manage to outperform the latter study by explicitly modelling *aleatory uncertainty* as an auxiliary prediction task. c) A model that aims to support the editorial decision process should only assume the availability of human review text during training, and be able to make recommendations in their absence. Inspired by missing modality hallucination methods (Hoffman et al., 2016; Tang et al.; Pérez et al., 2020)), we propose a *realistic* system that uses all available data for training, but imputes review representations at test time based on the abstract text.

1.2 Purpose & Ethical Statement

We sincerely believe that human peer reviews should continue to be the main component of the paper acceptance selection process, and this work in no way attempts to replace the human reviewers; instead, we believe an NLU model can serve as an additional reviewer, aiding an area chair's decision-making process by slot-filling a missing reviewer, or providing a data-driven, tie-breaking perspective to the editor in cases of borderline reviews. The motivation behind this proposal is that NLU models trained on large-scale data, can learn to robustly

---

[*] KF and GR contributed equally to this work.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5004–5009
August 1–6, 2021.

cancel out individual human biases – in a similar way neural networks are robust to non-systematic label noise (Rolnick et al., 2017). Admittedly, such a model can still learn and reflect systematic biases, but we leave an approach to this problem by means of methods that learn with biased data (Kim et al., 2019) for future work.

2 Related work

Kang et al. (2018) compiled the PeerRead dataset of submissions, and proposed NLU baselines for binary acceptance decision and score prediction such as novelty and technical correctness. Wang and Wan (2018) explored the acceptance task by modelling the abstract via a memory mechanism (Weston et al., 2015), along with one review. Ghosal et al. (2019) improved performance on the PeerRead dataset by utilising sentiment information using the VADER tool (Hutto and Gilbert, 2014) and universal sentence embeddings (Cer et al., 2018). Unfortunately, PeerRead is imbalanced in that the NeurIPS rejected submissions are not included, despite the fact that 90% of the accepted submissions with reviews in PeerRead are from NeurIPS. Furthermore, around 80% of the submissions are from arxiv, thus having no reviews attached to them.

Stappen et al. (2020) worked on the largest such dataset – the Interspeech 2019 submission corpus – and fused the variable number of text reviews per submission. On incorporating reviewer disagreement information, they showed the simple label average to be better than the adapted version proposed in (Ando et al., 2018), and also approached the score prediction tasks via deep quantile regression (Rodrigues and Pereira, 2020). Direct modelling of a label disagreement value, instead of using soft labels, has been utilised in areas such as affective computing (Han et al., 2017) and medical image modelling (Raghu et al., 2019). Alternatively, Kendall and Gal (2017) devised a method for aleatory uncertainty modelling that is learnt from the data, instead of requiring ground truth disagreement "labels". Both Han et al. (2017); Kendall and Gal (2017) have shown regularisation benefits of learnt uncertainty prediction.

3 Submission-level modelling

Following Wang and Wan (2018); Stappen et al. (2020), we focus on abstract x_i^{abs} and review texts $x_{i,r}^{rev}$ (numbering R_i) for the i-th submission, the acceptance classification labels given by the area chair y_i^{ac}, as well as by the reviewers $y_{i,r}^{rev}$. We use a model $\mathcal{M}$ that: a) learns abstract h_i^{abs} and review h_i^{rev} representations using corresponding *modules*, b) fuses the aforementioned into a submission representation h_i^{sub}, and generates the class probability distribution $\hat{y}_i$ via a *prediction module* and softmax. We then calculate the cross entropy (CE) loss with the *true* probability distribution y_i^{true}:

$$\mathcal{L}_{pred} = CE(\hat{y}_i, y_i^{true}). \tag{1}$$

The most straightforward way to do this is by using a *hard label*, i. e., assuming $y_i^{true} \equiv y_i^{ac}$, with all the probability concentrated at the final recommendation given by the area chair. This way, however, we withhold information about the reviewer uncertainty for the particular submission. Stappen et al. (2020) have successfully used the simple *soft label*:

$$y_i^{true} \equiv \frac{1}{R_i} \sum_{r=1}^{R_i} y_{i,r}^{rev}. \tag{2}$$

The value of soft labels becomes clear when one considers that, in their absence, true acceptance probabilities of .51 and .89 would receive the same treatment. Occasionaly, the area chair may disagree with the reviewers' aggregate decision, which motivates the interpolation of the two factors:

$$\mathcal{L}_{pred} = \lambda^{soft} \mathcal{L}_{pred}^{soft} + \lambda^{hard} \mathcal{L}_{pred}^{hard}, \tag{3}$$

where $\mathcal{L}_{pred}^{*}$, λ_* refers to prediction loss and regularisation parameter for either hard or soft labels.

3.1 Modelling recommendation subjectivity

We add a second "head" in our prediction module that outputs a predictive uncertainty estimate $\hat{\sigma}_i$. We now require a supervision signal to train it, either by: a) treating label disagreement as ground truth uncertainty (**GTU**), or b) learning a heteroscedastic loss attenuation score (**HLA**). Inspired by (Han et al., 2017; Raghu et al., 2019), we define our approach to GTU as a multi-task loss:

$$\mathcal{L}_{pred}^{GTU} = \gamma^{pred} \mathcal{L}_{pred} + \gamma^{unc} MSE(\hat{\sigma}_i, \sigma_i), \tag{4}$$

where σ_i is the standard deviation among reviewer recommendations; MSE is mean squared error.

For HLA we use the method proposed in (Kendall and Gal, 2017), where $\hat{\sigma}_i$ is the standard deviation of a normal distribution centred at the mean denoted by the main head logits. By sampling T logits, and calculating a corresponding

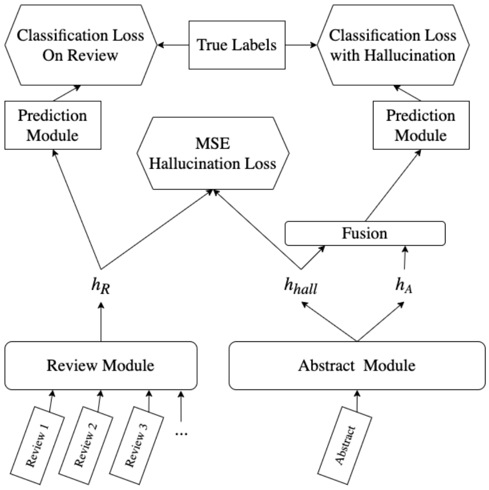

Figure 1: The peer review machine support system, including the hallucination mechanism. In a first training stage, only the *Classification Loss on Review* is used, to learn review representations. In the second training stage, the *MSE Hallucination Loss* and the *Classification Loss with Hallucination* are used.

class distribution $\hat{y}_i^t$, the loss function becomes:

$$\mathcal{L}_{pred}^{HLA} = \frac{1}{T}\sum_{t=1}^{T} CE(\hat{y}_i^t, y_i^{true}). \quad (5)$$

A larger $\hat{\sigma}_i$ relaxes the loss value for a sample that is difficult to predict correctly.

3.2 Imputing reviews at test time

Figure 1 depicts an overview of our architecture. Inspired by modality hallucination studies (Hoffman et al., 2016; Pérez et al., 2020), we use the abstract module to predict both the abstract h_i^{abs} and a review hallucination h_i^{hall} representation; *we do not generate review text*. In the first training stage, we train a model to predict based *only* on the review texts to learn meaningful h_i^{rev} representations. In the second stage, we use the true h_i^{rev} only as an auxiliary supervision target for training the h_i^{hall} representations by minimising MSE. Thus, we avoid *teacher forcing* (Bengio et al., 2015) by training the model to predict based only on the hallucinated representations, also available at test-time. The total loss, then, is:

$$\mathcal{L}_{hall} = \zeta^{hall} MSE(h_i^{rev}, h_i^{hall}) + \zeta^{pred}\mathcal{L}_{pred}. \quad (6)$$

4 Experiments

Small available dataset size is a limitation known to the community working on this domain (Kang et al., 2018; Ghosal et al., 2019) – we use the largest database of its kind (Stappen et al., 2020), i. e., the 2 179 preprocessed academic submissions, 5 842 reviews, with corresponding acceptance decisions and reviewer scores from the submission system of Interspeech 2019, shared with us by the technical chairs of the conference. After data cleaning and removal of corrupt entries, the accepted and rejected classes are well-balanced: 50.2 % acceptances, and 49.8 % rejections. The dataset is shuffled and split into 80-10-10 train-validation-test set percentages. We monitor the validation performance in terms of Macro-averaged F1 score, and also report the Macro-averaged Area Under Receiver Operator Characteristic (AU-ROC), averaged across 20 trials. We use the Adam optimiser (Kingma and Ba, 2014) with learning rate of 1e-3, and represent words using FastText (Bojanowski et al., 2017). Our abstract and review modules comprise a stacked 1D convolutional network with kernel sizes 4-4, interleaved by max pooling with rates 2-2, followed by a recurrent layer with gated recurrent unit cell and 100 hidden units, and attentional sequence pooling. The prediction module consists of two dense layers of 50-2 units, with a ReLu activation between them.

4.1 To model the reviewer or the area chair?

The interpolation weights λ^{soft}, λ^{hard} for the prediction error (cf. Eq. 3) are dataset-based and should be set based on validation performance. We experiment with a grid, ranging from $[1.0, 0.0]$ to $[0.0, 1.0]$ using a step of 0.2. $\lambda^{soft} \equiv 0.0$ *denotes the simple hard label case.* The results using the GTU loss are summarised in Table 1.

We find that the area chair and the reviewers' recommendations carry *complementary* information, and the best results of this study are at $\lambda^{soft} \equiv 0.8$. Interestingly, the agreement/accuracy between the editorial labels and the reviewer soft averages (rounded to 0 or 1) is 78.901%. The disagreements occur on close-to-borderline papers, in which cases the additional supervision is the most informative.

4.2 Are soft labels enough?

A comparison among the different loss functions, without hallucination, is summarised in Table 2. We report the best soft loss interpolation per case.

λ^{soft}	Metric	No Review		With Review	
		Mean	±	Mean	±
0.0	AU-ROC	.589	.026	.683	.065
	F1	.559	.019	.625	.061
0.2	AU-ROC	.590	.021	.724	.031
	F1	.558	.023	.651	.030
0.4	AU-ROC	**.608**	.027	.750	.029
	F1	**.565**	.033	.673	.030
0.6	AU-ROC	.605	.026	.745	.050
	F1	.553	.027	.668	.040
0.8	AU-ROC	.601	.034	**.776**	.044
	F1	.512	.066	**.694**	.043
1.0	AU-ROC	.564	.047	.730	.084
	F1	.396	.102	.629	.108

Table 1: Results with **GTU**, for classification with and without reviews, for different relative weightings or (soft) reviewer ratings / (hard) editorial decisions.

Loss	Metric	No Review		With Review	
		Mean	±	Mean	±
BL	AU-ROC	-	-	.550	.130
	F1	-	-	.652	.100
Soft	AU-ROC	.597	.022	.772	.029
	F1	.558	.024	.683	.024
GTU	AU-ROC	**.608**	.027	**.776**	.044
	F1	**.565**	.033	**.694**	.043
HLA	AU-ROC	.578	.044	**.776**	.020
	F1	.407	.082	*.688*	.020

Table 2: Results on using the abstract with or without the reviews, using different kinds of losses. **BL** denotes the baseline by Stappen et al. (2020).

In the case of *GTU* we found that the choice of $\gamma^{unc} \equiv \gamma^{pred} \equiv 0.5$ works best. The additional complexity of explicit uncertainty modelling is shown to be beneficial when compared to the simple soft labels, and GTU is better than the self-learnt uncertainty method HLA. Our model implementation with soft-hard loss mixing is also shown to greatly outperform a baseline (**BL**), i. e., the best result found in (Stappen et al., 2020).

We also performed statistical significance testing, using Welch's unequal variances t-test. No significance was found in improvement brought by uncertainty-aware methods compared to hard labels in the abstract-only experiments. However, GTU with hallucinated review representations is significantly better than abstract-only with $p < 0.1$ for AU-ROC and $p < 0.05$ for F1, and HLA with $p < 0.05$ for both measures. In the experiments using both abstract and reviews, the simple soft labels as well as HLA are both significantly better than hard labels in terms of AU-ROC with $p < 0.05$. GTU was significantly better than hard labels with $p < 0.1$ for F1 and $p < 0.05$ for AU-ROC.

Loss	Metric	Abstract		Hallucination	
		Mean	±	Mean	±
Hard	AU-ROC	.589	.026	**.592**	.022
	F1	.559	.019	**.562**	.024
Soft	AU-ROC	.611	.035	**.612**	.029
	F1	.535	.030	**.544**	.028
GTU	AU-ROC	.601	.025	**.608**	.030
	F1	.512	.054	**.532**	.076
HLA	AU-ROC	.557	.034	**.636**	.033
	F1	.358	.042	**.575**	.091

Table 3: We report the review hallucination results; for the uncertainty-aware methods, we used $\lambda^{soft} \equiv 0.8$.

4.3 Can we impute reviews?

Table 3 summarises the improvement brought by hallucinated reviews over the abstract-only case. Even though we only report a specific hard-soft interpolation weight $\lambda \equiv 0.8$, we observe this improvement universally. The HLA method with hallucination achieves both the best performance in this experiment, and the largest relative improvement (t-test, $p < .05$) upon the abstract-only case, as shown in Table 4. Lacking the true reviews, we have high label variance for the same abstract input, i. e., high aleatory uncertainty. HLA (Kendall and Gal, 2017) is designed for such cases, and guides the learning of hallucinated review representations through regularisation, allowing for a significant fraction of the performance gap to be covered. Hard-labels with hallucination is the method that performs relatively closest to its ceiling performance, but this can be explained by the ceiling being comparatively low in the hard-labels case.

The additional label uncertainty information, whether explicit or learnt, informs not just the classification capacity of the model, but also its ability to generate review representations. These hallucinated representations should be placed in embedding space such that they inform the model regarding the label, however not in an overconfident manner, given that the actual reviews are missing – this is exactly where knowledge of uncertainty contributes. In terms of a final method recommendation: we recommend the learnt attenuation based HLA, due to its better performance along with modality hallucination and the fact that it does not require the presence of multiple reviewer recommendations even at training-time.

4.4 Can we learn model disagreement?

The Pearson Correlation Coefficient (PCC) between the predicted uncertainty and the standard deviation of reviewer recommendations is .25 and

Loss	Metric	+ Relative	- Relative
Hard	AU-ROC	+0.5	-13.3
	F1	+0.5	-10.1
Soft	AU-ROC	+0.2	-19.7
	F1	+1.7	-19.8
GTU	AU-ROC	+1.2	-21.5
	F1	+3.8	-23.3
HLA	AU-ROC	+12.4	-17.9
	F1	+37.8	-16.4

Table 4: Relative improvements (in %) brought by hallucinated reviews compared to using only the abstract, and relative reductions compared to the performance ceiling in the case the reviews are available at test time. In cases where the true reviews cannot be assumed to be present in test/deployment, our hallucination approach allows for improvement of results compared to excluding reviews altogether.

.08 for GTU and HLA respectively in the abstract plus review case. The former indeed learns on actual disagreement labels, although high uncertainty prediction fidelity may not be necessary for high predictive performance, shown by the competitive HLA. When using only abstracts, PCC drops to .08 and .04 respectively, whereas by using hallucination we observe .08 and .05, indicating that the true review representations are required for good uncertainty prediction.

5 Conclusion

We have proposed a machine learning framework for automatic peer review support that makes better use of the available information, and is also realistic with respect to the limitations set by the task[1]. We have found that the the area chair and reviewer recommendations comprise independent supervision signals that should be used *in conjunction* to train the system. Furthermore, in order to relax the penalty for mispredicting subjective submissions, it is not enough to use a simple soft label average of the reviewer recommendations; one has to directly model an aleatory uncertainty score as an auxiliary task, either using ground truth "uncertainty labels", or through learnt attenuation of the loss. Finally, we utilise review representation hallucinations at test-time to best utilise available review texts in a realistic manner, and find that this approach works well with and *benefits* from the regularisation introduced by direct uncertainty modelling.

Even with the application of our review representation hallucination, the performance gap from the ceiling set by using the true review representations is still high. We intend to approach the task via self-supervision methods (He et al., 2020) that focus on multimodal data (Nagrani et al., 2020). Requesting additional reviewers based on inference-time uncertainty, similar to (Raghu et al., 2019), is another promising future work step, as is an analysis of the uncertainties predicted by our model using the different losses. Finally, we have shown in our study that only a representation of the abstract is required as the input both for acceptance and hallucination modelling. Since previous work (Ghosal et al., 2019) has shown that modelling an article based on the entire paper can be beneficial, we also intend to explore the impact of using such a highly expressive article representation for hallucinating review representations.

Acknowledgments

GR is funded by the Engineering and Physical Sciences Research Council (EPSRC) Grant No. 2021037.

References

Atsushi Ando, Satoshi Kobashikawa, Hosana Kamiyama, Roy Masumura, Yusuke Ijima, and Yushi Aono. 2018. Soft-Target Training With Ambiguous Emotional Utterances For DNN-Based Speech Emotion Classification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174. ACL.

Tirthankar Ghosal, Rajeev Verma, Asif Ekbal, and Pushpak Bhattacharyya. 2019. DeepSentiPeer: Harnessing Sentiment in Review Texts to Recommend Peer Review Decisions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1120–1130. ACL.

---

[1] `https://github.com/glam-imperial/Uncertainty-Aware-Machine-Paper-Reviewing`

Jing Han, Zixing Zhang, Maximilian Schmitt, Maja Pantic, and Björn Schuller. 2017. From Hard to Soft: Towards more Human-like Emotion Recognition by Modelling the Perception Uncertainty. In *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, pages 890–897. ACM Press.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Judy Hoffman, Saurabh Gupta, and Trevor Darrell. 2016. Learning with Side Information through Modality Hallucination. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 826–834. IEEE.

Clayton J Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *18th International AAAI Conference on Weblogs and Social Media*. AAAI.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A Dataset of Peer Reviews (PeerRead): Collection, Insights and NLP Applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1647–1661.

Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *Advances In Neural Information Processing Systems*.

Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. 2019. Learning not to learn: Training deep neural networks with biased data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9012–9020.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

John Langford and Mark Guzdial. 2015. The arbitrariness of reviews, and advice for school administrators. *Commun. ACM*, 58(4):12–13.

Arsha Nagrani, Joon Son Chung, Samuel Albanie, and Andrew Zisserman. 2020. Disentangled speech embeddings using cross-modal self-supervision. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6829–6833. IEEE.

Andrés F. Pérez, Valentina Sanguineti, Pietro Morerio, and Vittorio Murino. 2020. Audio-Visual Model Distillation Using Acoustic Images. *The IEEE Winter Conference on Applications of Computer Vision*.

Maithra Raghu, Katy Blumer, Rory Sayres, Ziad Obermeyer, Bobby Kleinberg, Sendhil Mullainathan, and Jon Kleinberg. 2019. Direct uncertainty prediction for medical second opinions. In *International Conference on Machine Learning*, pages 5281–5290.

Filipe Rodrigues and Francisco C. Pereira. 2020. Beyond expectation: Deep joint mean and quantile regression for spatio-temporal problems. *IEEE Transactions on Neural Networks and Learning Systems*.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Amanda Sizo, Adriano Lino, Luis Paulo Reis, and Álvaro Rocha. 2019. An overview of assessing the quality of peer review reports of scientific articles. *International Journal of Information Management*, 46:286 – 293.

Lukas Stappen, Georgios Rizos, Madina Hasan, Thomas Hain, and Björn W. Schuller. 2020. Uncertainty-Aware Machine Support for Paper Reviewing on the Interspeech 2019 Submission Corpus. In *Interspeech*, pages 1808–1812.

Yongyi Tang, Lin Ma, and Lianqiang Zhou. Hallucinating Optical Flow Features for Video Classification. *Proceedings of the Twenty-Eight International Joint Conference on Artificial Intelligence*, pages 926–932.

Ke Wang and Xiaojun Wan. 2018. Sentiment Analysis of Peer Review Texts for Scholarly Papers. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 175–184. ACM.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. *arXiv:1410.3916*.

Automatically Select Emotion for Response via Personality-affected Emotion Transition

Zhiyuan Wen, Jiannong Cao, Ruosong Yang, Shuaiqi Liu, Jiaxing Shen
Department of Computing,
The Hong Kong Polytechnic University
Kowloon, Hong Kong, China
{cszwen, csjcao, csryang, csslliu, jiaxshen}@comp.polyu.edu.hk

Abstract

To provide consistent emotional interaction with users, dialog systems should be capable to automatically select appropriate emotions for responses like humans. However, most existing works focus on rendering specified emotions in responses or empathetically respond to the emotion of users, yet the individual difference in emotion expression is overlooked. This may lead to inconsistent emotional expressions and disinterest users. To tackle this issue, we propose to equip the dialog system with personality and enable it to automatically select emotions in responses by simulating the emotion transition of humans in conversation. In detail, the emotion of the dialog system is transitioned from its preceding emotion in context. The transition is triggered by the preceding dialog context and affected by the specified personality trait. To achieve this, we first model the emotion transition in the dialog system as the variation between the preceding emotion and the response emotion in the **V**alence-**A**rousal-**D**ominance (VAD) emotion space. Then, we design neural networks to encode the preceding dialog context and the specified personality traits to compose the variation. Finally, the emotion for response is selected from the sum of the preceding emotion and the variation. We construct a dialog dataset with emotion and personality labels and conduct emotion prediction tasks for evaluation. Experimental results validate the effectiveness of the personality-affected emotion transition.[1].

1 Introduction

Emotional intelligence can be considered a mental ability to reason validly with emotional information, and the action of emotions to enhance thought (Mayer, 2004). Hence, to create dialog systems with emotional intelligence during communication, it is necessary to enable the machine to understand the emotion of users, select appropriate response emotions and express in conversation.

Existing works either focus on rendering specified emotions in responses (Zhou et al., 2018; Colombo et al., 2019), or understanding the emotion of users and respond empathetically (Zandie and Mahoor, 2020; Zhong et al., 2020; Lin et al., 2019); but how to automatically select the emotion for response is seldom discussed. Wei et al. (2019) proposes to learn appropriate emotional responses from massive anonymous online dialogues. However, trained on conversations from different speakers, the dialog system ignores the individual difference of expressing emotions. This may lead to inconsistent emotional interactions and disinterest users as they may feel they are still talking to rigid machines.

In a dialog system, automatically selecting the emotion for response is to decide an emotion to be expressed facilitating the emotional response generation. Emotion selection can be modeled as the emotion transition (Thornton and Tamir, 2017), which refers to how the preceding emotion changes to the next, of the dialog system reacting to the dialog context. To achieve it like humans, it requires long-term patterns of thought, and behavior associated with an individual (Ball, 2000). Mehrabian (1996a) shows that the personality, *e.g.*, the big-five personality model (Costa and McCrae, 1992) also can be represented as temperament in the **V**alence-**A**rousal-**D**ominance (VAD) space for emotions (Mehrabian, 1996b).[2] The finding suggests that different personalities make different impacts on emotional expressions. Inspired by these works, we propose a personality-affected emotion

---

[1] Our dataset is released at: github.com/preke/PELD

[2] It is Pleasure-Arousability-Dominance (PAD) in the original paper, PAD and VAD share the same meaning in the context of text understanding, we will use VAD for consistency henceforth.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5010–5020
August 1–6, 2021. ©2021 Association for Computational Linguistics

transition model to endow personality to the dialog system, enabling it to select emotions that react to the dialog context affected by its given personality.

In our method, we model the emotion transition of the dialog system as the variation in the VAD space from its preceding emotion to the next emotion in the response to users. We first obtain the preceding emotion of the dialog system from the dialog context and project it into the VAD space as an emotion vector. Simultaneously, we endow a personality trait, a 5-dimension vector representing the strength of each dimension in the big-five personality traits, to the dialog system. Then, we design neural networks to encode the dialog context and the personality traits into the VAD space to compose the variation of emotion. Finally, the emotion for response is selected based on the sum of the preceding emotion and the variation.

To facilitate related researches, we construct the **P**ersonality **E**motion**L**ines **D**ataset (PELD), which includes 6,510 dialogue triples of daily conversations with emotion labels and annotated personality traits. The emotion labels and personality annotations are adopted from other researches (Poria et al., 2018; Zahiri and Choi, 2017; Jiang et al., 2019) analyzing the script of a famous TV series *Friends*[3]. We conduct emotion prediction tasks on the PELD dataset to evaluate the effectiveness of our method. The results suggest that the personality-affected emotion transition does contribute to better accuracy in emotion selection. To summary, our contributions are as follows:

- We raise the problem of automatically select the emotion for response in conversation and propose a new perspective to solve it through personality-affected emotion transition.
- We construct a dialog script dataset with emotion and personality labels and analyze the patterns of emotion transitions in our dataset to facilitate related researches.
- We evaluate the effectiveness of our proposed method on emotion prediction tasks and analyze the effects of personality and emotion transition respectively.

2 Related Works

Our research is related to the emotional dialog systems, and the personality influence on emotion expression in psychology and Human-Computer Interaction (HCI). So, we review existing works in the two aspects as follows.

2.1 Emotional Dialog Systems

The concept of the emotional dialog system first occurred in (Colby, 1975), where a rule-based emotion simulation chatbot was proposed. Microsoft introduced the Xiaoice (Zhou et al., 2020), an empathetic social chatbot that is able to recognize users' emotional needs, in 2014. Related researches become popular recently since Zhou et al. (2018) proposed the Emotional Chatting Machine to exploit the deep learning approach in building a large-scale emotionally aware conversational bot. Most existing works focus on incorporating specified emotion factors into neural response generation. Shantala et al. (2018) trains emotional embeddings based on context and then integrated them into response generation. Colombo et al. (2019) controls the emotional response generation with both categorical emotion representations and continuous word representations in VAD space (Mohammad, 2018). Moreover, Asghar et al. (2018) proposes an affectively diverse beam search for decoding. Besides, reinforcement learning is also adopted to encourage response generation models to render specified emotions. Li et al. (2019) combines reinforcement learning with emotional editing constraints to generate meaningful and customizable emotional replies. (Sun et al., 2018) also uses an emotion tag to partially rewarding the model to express specified emotion.

However, it is impractical to always specify response emotions for dialog systems in real application scenarios. To simulate the emotional interaction among humans, Wei et al. (2019) designs an emotion selector to learn the proper emotion for responses from massive dialogue pairs. But the emotional expression is subjective, for the same post, different users may have different emotions in their responses. So, the pattern learned only from online dialogues ignores the user information and turns to be impractical.

2.2 Personality Effects on Emotions

Emotion is a complex psychological experience of an individual's state of mind as interacting with people or environmental influences (Han et al., 2012). The **P**leasure-**A**rousal-**D**ominance (PAD) (Mehrabian, 1996b) or **V**alence-**A**rousal-**D**ominance (VAD) emotion temperament model

---

[3] https://en.wikipedia.org/wiki/Friends

shows three nearly orthogonal dimensions providing a comprehensive description of emotional states. Based on this, several psychologists studied the relationship between human emotional factors and personality factors. However, most of them are rule-based models (Johns and Silverman, 2001) and probabilistic models (André et al., 1999). Mehrabian (1996a) utilized the five factors of personality (Costa and McCrae, 1992) to represent the VAD temperament model through linear regression analysis. This finding is widely used to design robots having non-verbal emotional interaction with users (Han et al., 2012; Masuyama et al., 2018), where the pre-defined personalities of robots affect their propensity of simulated emotion transitions.

To integrate the analysis above into Artificial Intelligence, some researchers in HCI borrow the idea and design facial emotional expressions for humanoid robots. Ball (2000) utilizes models of emotions and personality encoded as Bayesian networks to generate empathetic behaviors or speech responses to users in conversation. Han et al. (2012) employed five factors of personality to a 2D (pleasure-arousal) scaling model to represent a robotic emotional model. Masuyama et al. (2018) introduces an emotion-affected associative memory model for robots expressing emotions. While in NLP, though the VAD space is adopted to model emotions in some researches (Mohammad, 2018; Colombo et al., 2019; Asghar et al., 2018), the personality influence on emotion in dialogues is still an open problem.

3 Methodology

3.1 Problem Definition

We research on enabling the dialog system to automatically select emotions for response through the personality-affected emotion transition.

Formally, a dyadic emotional conversation between the user and the dialog system contains the dialog context $C = \{U_1, U_2, ..., U_{n-1}\}$ including all the preceding $n - 1$ utterances from both the user and the dialog system, the preceding emotion E_i expressed in $U_i \in C$ which is the last utterance from the dialog system, and the response emotion E_r for the dialog system to facilitate generating the next emotional response U_n to the user. We specify a personality trait P_n to the dialog system and enable it to select response emotion E_r through the personality-affected emotion transition model F_{ET}:

$$E_r = F_{ET}(E_i|P_n, C) \tag{1}$$

where E_r is transitioned from E_i. The transition is triggered by the preceding dialog context C and affected by the specified personality trait P_n. In the following content, we will introduce how we model this process in detail.

3.2 Preliminaries

3.2.1 Emotions in the VAD space

Assuming in the problem above, emotions in all emotional utterances can be categorized into the six basic emotions: *Anger*, *Disgust*, *Fear*, *Joy*, *Sadness*, and *Surprise* (Ekman and Davidson, 1994). We project the basic emotions into the Valence-Arousal-Dominance (VAD) space as Table 1 refer to the analysis result in (Russell and Mehrabian, 1977)[4]. The VAD space indicates emotion intensity in three different dimensions, where the valence measures the positivity/negativity, arousal the excitement/calmness, and dominance the powerfulness/weakness. As for the utterances with no explicit emotion, we use the *Neutral* with (0.00, 0.00, 0.00) as the VAD vector.

Basic Emotions	(Valence, Arousal, Dominance)
Anger	(-0.51, 0.59, 0.25)
Disgust	(-0.60, 0.35, 0.11)
Fear	(-0.62, 0.82, -0.43)
Joy	(0.81, 0.51, 0.46)
Neutral	(0.00, 0.00, 0.00)
Sadness	(-0.63, -0.27, -0.33)
Surprise	(0.40, 0.67, -0.13)

Table 1: Emotions in the VAD Space.

3.2.2 Personalities in the VAD space

Meanwhile, the big-five personality traits (OCEAN, shown in Table 2) are widely used for psychological analysis. Mehrabian (1996a) proposed a temperament model shown in Equation 2 derived through linear regression to show the VAD scales of personality traits, where O, C, E, A, N are the strength of the big-five personality traits.

---

[4] *fear* and *Joy* correspond to *Terrified* and *Happy* in the reference table.

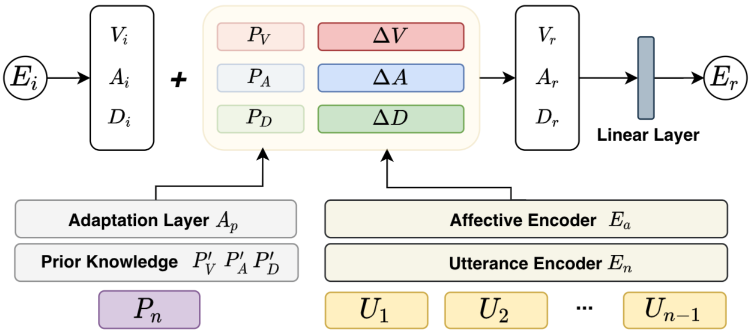

Figure 1: The Model Illustration

$$\begin{aligned} P_V &= 0.21E + 0.59A + 0.19N \\ P_A &= 0.15O + 0.30A - 0.57N \\ P_D &= 0.25O + 0.17C + 0.60E - 0.32A \end{aligned} \tag{2}$$

3.3 Personality-affected Emotion Transition

Based on the problem definition and the preliminaries above, we design the Personality-affected Emotion Transition model as illustrated in Figure 1. Our model mainly include three modules: the personality effect on emotions in the left lower part, the context encoding in the right lower part, and the emotion transition in the top half in Figure 1. We will introduce these three modules in detail as follow.

3.3.1 Personality Effect on Emotions

In our model, the personality of the dialog system is specified as a 5-dimensional vector $P_n = [O, C, E, A, N]$ representing the strength in Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism, respectively.

The temperament of personality in the VAD space (shown in Equation 2) is widely used as weighting parameters for emotion transition of robots in HCI works (Han et al., 2012; Masuyama et al., 2018). However, the numeric coefficients in Equation 2 are summarized from analysis of questionnaire results from 72 participants (Mehrabian, 1996a), which are not suitable to directly adopted as hyper-parameters in the model design. Hence, we choose to adopt the analysis results in Equation 2 as prior knowledge and learn suitable coefficients for personality by neural networks. First, we still calculate P'_V, P'_A, P'_D from the personality P_n by Equation 2; then we use P'_V, P'_A, P'_D as initialized input for an adaptation layer A_p to learn the weighting parameters P_V, P_A, P_D that suitable for the training data.

Factor	Description
Openness	Openminded, imaginative, and sensitive.
Conscientiousness	Scrupulous, well-organized.
Extraversion	The tendency to experience positive emotions.
Agreeableness	Trusting, sympathetic, and cooperative.
Neuroticism	The tendency to experience psychological distress.

Table 2: The OCEAN personality traits and description (Costa and McCrae, 1992)

3.3.2 Context Encoding

The dialog context acts as a set of parameters that may influence a person to speak an utterance while expressing a certain emotion (Poria et al., 2018). In the VAD space, the emotion transition is regarded as the variation from one point (the preceding emotion) to another point (the next emotion). Thus, we generate the emotion transition variations $\Delta V, \Delta A, \Delta D$ from the semantic representations of the preceding dialog context C.

$$\begin{aligned} &R_c = E_n(U_1) \oplus E_n(U_2) ... \oplus E_n(U_{n-1}) \\ &\Delta V, \Delta A, \Delta D = E_a(R_c) \end{aligned} \tag{3}$$

We fine-tune the pre-trained RoBERTa[5] (Liu et al., 2019) encoder, a famous pre-trained language model whose performance is widely validated in many natural language understanding tasks, to first extract the semantic representations $E_n(U_1), ..., E_n(U_{n-1})$ of all $n-1$ utterances in C. Then, we concatenate the semantic representations of utterances to obtain the overall context semantics R_c. Finally, $\Delta V, \Delta A, \Delta D$ are calculated by feeding R_c into an affective encoder E_a, which extract the affective information from R_c in the aspect of V, A, D, respectively.

---

[5] Here we adopt the pre-trained RoBERTa-base model.

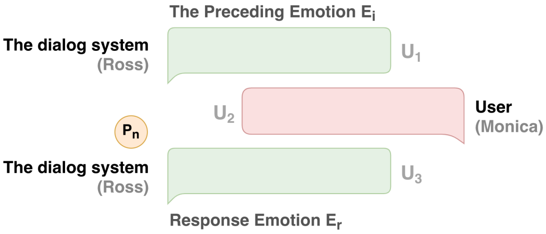

Figure 2: A triple example in PELD. The dyadic conversation between Ross and Monica (two main roles in *Friends*, P_n is the personality of Ross. The dialog system is set as Ross and talk with the user set as Monica in this example.

3.3.3 Emotion Transition

After we obtain the weighting parameters P_V, P_A, P_D and the emotion transition variation $\Delta V, \Delta A, \Delta D$, the emotion for response is generated by the sum of the VAD vectors of the preceding emotion and the weighted variation, as shown in Equation 4.

$$\begin{aligned} V_r &= V_i + P_V \cdot \Delta V \\ A_r &= A_i + P_A \cdot \Delta A \\ D_r &= D_i + P_D \cdot \Delta D \\ E_r &= F_c(V_r, A_r, D_r) \end{aligned} \tag{4}$$

where the V_i, A_i, D_i are the VAD vectors of E_i, and the V_r, A_r, D_r are the emotion transition results in the VAD space. To alleviate the errors of using the numeric value in calculated VAD vectors, we add a linear layer F_c to transform V_r, A_r, D_r into a probability distribution on the discrete emotion categories. The output E_r is the emotion with the largest probability.

4 The PELD Dataset

4.1 Dataset Construction & Statistics

To facilitate related researches, we construct the **P**ersonality **E**motion**L**ines **D**ataset (**PELD**), an emotional dialog dataset with personality traits for speakers. As labeling online conversation on social media with speakers' personalities is time-consuming and may cause privacy issues, we turn to research on the dialogue script of a famous TV series *Friends*. This classic script is widely analyzed in many dialog researches (Li et al., 2016; Li and Choi, 2020; Jiang et al., 2019).

In PELD, each sample is represented as a dialog triple ($C = \{U_1, U_2, U_3\}, \{E_i, E_r\}, P_n$), shown in Figure 2) as a dyadic conversation. E_i and E_r are emotions expressed in U_1 and U_3, respectively. The utterances and their emotion labels are mainly adopted from the dialogues in the MELD (Poria et al., 2018) and the EmoryNLP dataset (Zahiri and Choi, 2017), two famous datasets analyzing emotional expressions in *Friends*. To keep consistency, each dialog triple in PELD is constructed within the same dialogue in the original datasets.

Roles	Personality Traits (O,C,E,A,N)
Chandler	[0.648, 0.375, 0.386, 0.58, 0.477]
Joey	[0.574, 0.614, 0.297, 0.545, 0.455]
Monica	[0.713, 0.457, 0.457, 0.66, 0.511]
Phoebe	[0.6, 0.48, 0.31, 0.46, 0.56]
Rachel	[0.635, 0.354, 0.521, 0.552, 0.469]
Ross	[0.722, 0.489, 0.6, 0.533, 0.356]

Table 3: Personalities of *Friends* main roles in PELD.

The personality traits in our dataset are adopted from the personality annotations in 711 different dialogues (Jiang et al., 2019). Refer to the annotations, a role may exhibit different aspects of its personality in different dialogues. We only keep the personality traits of the six main roles in *Friends* for confidence as these annotations are most frequent. For each of the main roles, we average their annotated personality traits in all the dialogues by $P_n = \frac{1}{K}\sum_{i=1}^{K} P_i$ for simplification, where K is the number of annotations. The averaged results are shown in Table 3.

We split the PELD into **Train**, **Valid**, and **Test** set with portion around 8:1:1. The total number of utterances in PELD (10,648) is less than the sum of the original MELD (13,708) and the EmoryNLP (9,489), as not all dialogues are suitable to construct triples including main roles. The overall statistics of the dataset is shown in Table 4.

Similar to existing emotional conversation datasets (Li et al., 2017; Busso et al., 2008), PELD also suffers the emotion imbalance issue. Utterances labeled as *Neutral* are the majority, while *Fear* and *Disgust* only take a small portion. Though it reflects the real emotion distribution in daily conversation, it also brings challenges to machine learning models to identify and generate emotions. We tried several automatic methods for data augmentation like synonym substitution, back-translation, or the EDA proposed in (Wei and Zou, 2019). But most of the synthetic samples are either odd or the same as the original samples. The

Basic Statistics	**Train**	**Valid**	**Test**	**Total**
#Triple	5273	586	651	6510
#Unique Uttr.	9306	1518	1675	10468
Avg. Uttr. Length	9.26	9.33	8.95	9.32
#Emotion	**Train**	**Valid**	**Test**	**Total**
Anger	1863	236	241	2340
Disgust	312	32	32	376
Fear	1101	114	131	1346
Joy	2863	326	344	3533
Neutral	7055	756	890	8701
Sadness	1088	121	136	1345
Surprise	1537	173	179	1889
#Sentiment	**Train**	**Valid**	**Test**	**Total**
Positive	4400	499	523	5422
Neutral	7055	756	890	8701
Negative	4364	503	540	5407
#Triple of Main Roles	**Train**	**Valid**	**Test**	**Total**
Chandler	880	97	108	1085
Joey	912	109	102	1123
Monica	850	94	107	1051
Phoebe	782	87	103	972
Rachel	921	112	123	1156
Ross	928	87	108	1123

Table 4: Basic Statistics in PELD.

reason might be there are limited options for short sentences as utterances in conversation to replace synonyms, add or delete words.

Another way to alleviate the imbalance issue is to expand the granularity of emotion to sentiment. As mentioned in **3.2**, in the VAD space, the Valence dimension of emotions measures the positivity and negativity, we can categorize the emotions into sentiments according to the values of Valance; i.e., positive emotions: *Joy* and *Surprise*; negative emotions: *Anger*, *Disgust*, *Fear*, and *Sadness*. Thus, the distribution of sentiments in PELD is also shown in Table 4. Besides, dialog triples of six main roles (each triple corresponds to a main role with the personality trait) are averagely distributed in all train, valid, and test sets in PELD.

4.2 Emotion Transitions in PELD

After constructed PELD, we further explore the dataset in the aspect of emotion transitions. As the triples in PELD are constructed for analyzing the emotion transitions between E_i in U_1 and the E_r in U_3. Table 5 shows the emotion and sentiment distributions in the U_1 and U_3, respectively. Besides, we also count the sentiments of emotions in U_1 and U_3 denoted as S_1 and S_3. We can see that for both emotion and sentiment, the distributions in U_1 and U_3 are similar, which means the transition of emotions and sentiments are equitable in PELD triples. Besides, the proportions of all emotions and sentiments are also similar to the overall statistics of PELD, which suggests that the emotions and sentiments in PELD are also average distributed in the triples.

Since emotion transitions are affected by the personality traits as discussed above, we exhibit the emotion transition patterns for different roles with different personality traits in Figure 3. Although the emotion transitions are also correlated to the dialog context, we can still find patterns through these transition matrixes[6].

In general, among the six transition matrixes, all the first columns are in deeper colors, which indicates most transitions occur from other emotions to *Neutral* as it is the majority emotion in PELD. Besides, blocks with deeper color also more likely to occur around or in the diagonals of the transition matrixes; it suggests the preceding emotions tend to transition to the same or similar emotions. As for individual roles, 0.59 of the *Anger* from Rachel remains the same in dialog triples, while for other roles, most *Anger* emotions are transferred to *Neutral* and *Anger*. Besides, most *Surprise* from Ross transfers to the *Neutral*, *Joy*, and *Surprise*, but most *Surprise*s of the other five roles tend to transfer to only *Surprise* and *Neutral*.

Moreover, to highlight the individual differences of emotion transitions among the six main roles in detail, we also show the standard deviations (Std) of each row in the emotion transition matrixes of the six main roles, as shown in Figure 4. The red bar chart shows the Std of the infinite norms of rows in the emotion transition matrix, which indicates the diversity of the most probable emotions from the same emotion in emotion transfers of different roles. While the blue bar chart shows the Std of the L2-norms, which generally describes the difference in how different roles transfer from one emotion to other emotions.

Both charts show similar patterns of emotion transitions. *Anger*, *Surprise*, and *Disgust* vary the most in different roles, while people are more common when process *Neutral* and *Joy* emotions in

---

[6] Here, we analyze the personality-affected emotion transition based on roles rather than the numeric traits in Table 2 to avoid numeric observation errors.

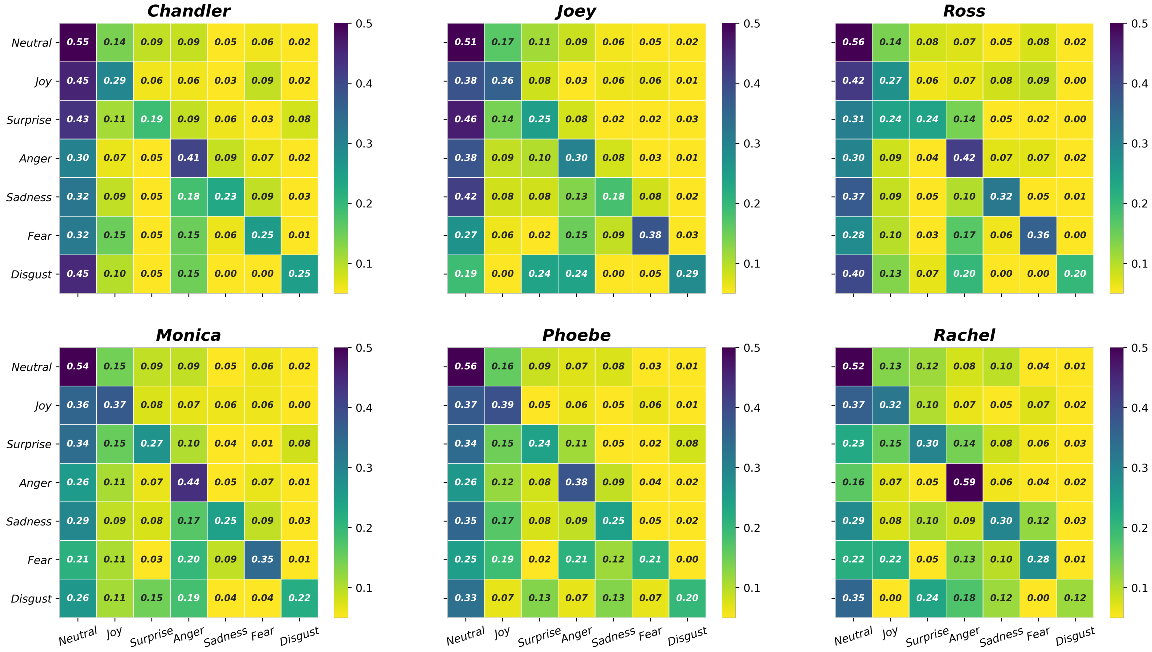

Figure 3: Emotion transition matrixes of the six main roles in PELD. Each row in a matrix shows the ratios of the current emotion E_i is transferred to the next emotion E_r.

conversation. Besides, negative emotions (*Anger*, *Sadness*, *Fear*, and *Disgust*) are relatively higher than positive emotions and *Neutral* on average. So, we can infer that the personality traits influence more in the emotion transfers from negative emotions.

5 Experiment

5.1 Evaluation Tasks

To validate the effectiveness of our proposed emotion generation model, we set two evaluation tasks: Emotion Prediction and Sentiment Prediction on PELD. Emotion Prediction requires the model to predict the emotion in the upcoming utterance based on the preceding dialog context in a dyadic conversation scenario; while Sentiment Prediction has the same setting except to predict the sentiment in the upcoming utterance.

For both tasks, we evaluate the prediction performance by F-scores of single emotion or sentiment. Besides, the overall performance is also measured from two aspects with the macro averaged (**m-avg**) and the weighted averaged (**w-avg**) F-scores. A higher m-avg indicates the model performs relatively better predicting all categories, while a higher w-avg indicates the model predicts emotions or sentiments with larger proportions in the dataset better.

Tri.Emos	**Neutral**	**Joy**	**Surprise**	**Anger**	**Sadness**	**Fear**	**Disgust**
E_i	2910	1242	597	751	438	457	115
E_r	2771	1123	634	858	493	487	144
Tri.Sentis	**Neutral**	**Positive**		**Negative**			
S_i	2910	1839		1761			
S_r	2771	1757		1982			

Table 5: Emotions in PELD Triples

5.2 Ablation Study Setting

Although plenty methods (Majumder et al., 2019; Ghosal et al., 2020, 2019) has been proposed to analyze emotions in dialogues of *Friends*, most of their targets are to recognize the emotions of utterances in conversation. Compared with emotion recognition, the problem setting of selecting emotion is different and it is more difficult to select the appropriate emotion in response without knowing the response content. So, instead of comparing with other emotion recognition models, we turn to conduct ablation studies to evaluate the effectiveness of different parts of our model design. The ablation study compares the performances of the following models:

RoBERTa: RoBERTa (Liu et al., 2019) is a famous pre-trained language model designed for natural language understanding. Its performance

Methods	Anger	Disgust	Fear	Joy	Neutral	Sadness	Surprise	m-avg	w-avg
RoBERTa	0.218	0.000	0.107	0.214	0.453	0.122	0.126	0.177	0.287
RoBERTa-P	0.178	0.000	0.047	**0.265**	0.517	0.110	0.053	0.167	0.352
PET-VAD	0.190	**0.081**	0.115	0.188	0.474	0.000	**0.179**	0.175	0.309
PET-CLS	**0.320**	0.070	**0.140**	0.198	**0.528**	**0.155**	0.098	**0.203**	**0.424**

Table 6: Results for Emotion Prediction.

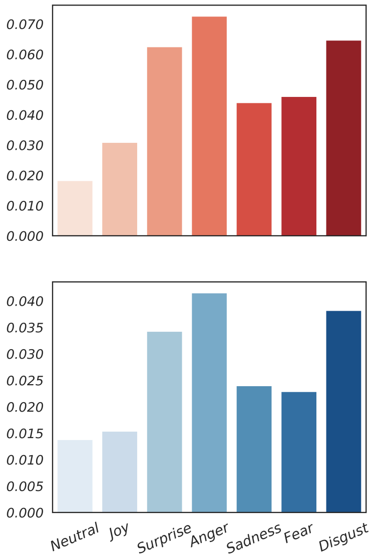

Figure 4: The standard deviations of the infinity norm (red) and the L2-norm (blue) of each row in emotion transition matrixes of the six main roles in PELD.

is widely validated in many downstream tasks. We here use pre-trained RoBERTa, corresponding to the E_n in our model, to encode the preceding dialog context to obtain the semantic representation as input, then directly predict the emotion for response through a classification head.

RoBERTa-P: We concatenate the personality vector of the speakers with the dialog context representation by RoBERTa as the feature, then predict the response emotion. This method is to evaluate whether personality influences the expression of emotions.

PET-VAD: As emotions can be represented by both discrete category labels or vectors in the VAD spaces. PET-VAD is set to compare the different usages of emotion VAD vectors in our model. During training, PET-VAD regressions the VAD vectors of target emotions by minimizing the Mean Squared Error (MSE) between generated vectors and the VAD vectors of ground truth emotions. The prediction output of PET-VAD is the closest neighbor emotions of generated VAD vectors measured by MSE.

PET-CLS: This is our method Personality-affected Emotion Transition with a classifier after obtaining the VAD vector of generated emotion. PET-CLS predicts emotions in the upcoming utterances as described in Section 3.

For RoBERTa, RoBERTa-P, and PET-CLS directly outputting discrete emotions, we adopt the Focal loss (Lin et al., 2017) to relieve the imbalanced emotion prediction.

6 Results and Analysis

In this section, we report and analyze the experimental results on the Test set of PELD in our ablation study. All results are chosen by the best performance on the Valid set within 50 epochs training.

6.1 Results for Emotion Prediction

The results on the Emotion Prediction task are reported in Table 6. First of all, as a seven-classes prediction task also suffered from the imbalance issue, the overall performance is moderately low, which also indicates the difficulty of the task. As for the averaged F-scores, PET-CLS improves both the w-avg and m-avg by a large margin from all other methods, which verifies our personality-affected emotion transition method.

In detail, all models perform better on emotions with larger portions (*Neutral* and *Joy*), as they are more probable to occur in the response emotion. Moreover, PET-VAD and PET-CLS achieve moderately higher F-score on the minority emotions (*Anger*, *Sadness*, *Disgust*, *Fear*, and *Surprise*), which shows that the emotion transition process is more important generating these minority emotions. It also verifies the finding in Section 4.2.

Methods	Negative	Neutral	Positive	m-avg	w-avg
RoBERTa	0.415	0.430	0.323	0.389	0.390
RoBERTa-P	0.401	**0.505**	0.176	0.361	0.430
PET-CLS	**0.492**	0.474	**0.327**	**0.431**	**0.445**

Table 7: Results for Sentiment Prediction.

On the other hand, although PET-VAD is based on the designed personality-affected emotion transition, most single emotion F-scores of PET-VAD are lower than RoBERTa or RoBERTa-P. We discuss the possible reasons as follows. One reason might be that the imbalance emotion issue cannot be alleviated in directly regression the emotion VAD vectors. Another reason might be that the value of emotion VAD vectors in Table 1 are estimated rather than precisely calculated, and the distance among different emotions in the theoretical VAD space is not similar to those in the emotion distribution in daily conversation.

6.2 Results for Sentiment Prediction

As predicting the emotions for the upcoming responses is difficult due to the multiple imbalanced categories, we also report the results on the Sentiment Prediction task in Table 7. Besides, different from the analysis above, which categorizes emotions by their portions in PELD, sentiment is another aspect of emotion analysis. As the sentiments are not directly described in the VAD spaces, we only report the results for RoBERTa, RoBERTa-P, and the PET-CLS. Besides, we only change the output size of PET-CLS from 7 (for emotions) to (3 for sentiments) and preserve the emotion transition process in this task.

In general, we can see that the prediction F-scores of sentiments are higher than emotion predictions. Besides, the prediction of negative emotions is much easier than predicting positive emotions in all three methods. It may because although the numbers of sentiments are similar, the categories of negative emotions (*Anger*, *Sadness*, *Fear*, and *Disgust*) are more than positive emotions (*Joy* and *Surprise*). Equipped with our model design, PET-CLS outperforms both RoBERTa and RoBERTa-P excepted for the neutral sentiment. It suggests that the personality-affected emotion transitions also facilitate sentiment prediction. However, only concatenating the personality vectors with context representation, RoBERTa-P improves the F-scores of Neutral but decreases the Positive and Negative. Hence, direct concatenation limits the effect of personality information in sentiment prediction.

7 Conclusion and Future Work

In this work, we raise the problem of automatically selecting the emotion for response considering the individual differences in conversation and propose a new perspective to solve it through personality-affected emotion transition. Besides, we construct a dialog script dataset PELD with emotion and personality labels to facilitate related researches. We also validate our personality-affected emotion transition model in emotion prediction experiments.

Facial expressions, voices, gestures, and environment information are also vital in emotional interaction, but they are not captured in the purely text-based dialog systems. Besides, as seen from statistics in PELD, the most common emotion in the dialog scripts is still Neutral. One possible reason is that other subtle affective information is not captured in the text. Therefore, our future works will continue to investigate the personality effects on emotions in the multi-modality scenario.

7.1 Acknowledgement

This work is supported by the Hong Kong RGC Collaborative Research Fund with project code C6030-18G and Hong Kong Red Swastika Society Tai Po Secondary School with project code P20-0021.

References

Elisabeth André, Martin Klesen, Patrick Gebhard, Steve Allen, and Thomas Rist. 1999. Integrating models of personality and emotions into lifelike characters. In *International Workshop on Affective Interactions*, pages 150–165. Springer.

Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. 2018. Affective neural response generation. In *European Conference on Information Retrieval*, pages 154–166. Springer.

Gene Ball. 2000. Emotion and personality in a conversational agent. *Embodied conversational agents*, pages 189–219.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335–359.

Kenneth Mark Colby. 1975. *Artificial paranoia: a computer simulation of paranoid process*. Pergamon Press.

Pierre Colombo, Wojciech Witon, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia. 2019. Affect-driven dialog generation. *arXiv preprint arXiv:1904.02793*.

Paul T Costa and Robert R McCrae. 1992. Normal personality assessment in clinical practice: The neo personality inventory. *Psychological assessment*, 4(1):5.

Paul Ed Ekman and Richard J Davidson. 1994. *The nature of emotion: Fundamental questions.* Oxford University Press.

Deepanway Ghosal, Navonil Majumder, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. 2020. Cosmic: Commonsense knowledge for emotion identification in conversations. *arXiv preprint arXiv:2010.02795*.

Deepanway Ghosal, Navonil Majumder, Soujanya Poria, Niyati Chhaya, and Alexander Gelbukh. 2019. Dialoguegcn: A graph convolutional neural network for emotion recognition in conversation. *arXiv preprint arXiv:1908.11540*.

Meng-Ju Han, Chia-How Lin, and Kai-Tai Song. 2012. Robotic emotional expression generation based on mood transition and personality model. *IEEE transactions on cybernetics*, 43(4):1290–1303.

Hang Jiang, Xianzhe Zhang, and Jinho D Choi. 2019. Automatic text-based personality recognition on monologues and multiparty dialogues using attentive networks and contextual embeddings. *arXiv preprint arXiv:1911.09304*.

Michael Johns and Barry G Silverman. 2001. How emotions and personality effect the utility of alternative decisions: a terrorist target selection case study. *Center for Human Modeling and Simulation*, page 10.

Changmao Li and Jinho D Choi. 2020. Transformers to learn hierarchical contexts in multiparty dialogue for span-based question answering. *arXiv preprint arXiv:2004.03561*.

Jia Li, Xiao Sun, Xing Wei, Changliang Li, and Jianhua Tao. 2019. Reinforcement learning based emotional editing constraint conversation generation. *arXiv preprint arXiv:1904.08061*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. 2019. MoEL: Mixture of empathetic listeners. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 121–132, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2019. Dialoguernn: An attentive rnn for emotion detection in conversations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6818–6825.

Naoki Masuyama, Chu Kiong Loo, and Manjeevan Seera. 2018. Personality affected robotic emotional model with associative memory for human-robot interaction. *Neurocomputing*, 272:213–225.

John D Mayer. 2004. (2004) what is emotional intelligence?

Albert Mehrabian. 1996a. Analysis of the big-five personality factors in terms of the pad temperament model. *Australian journal of Psychology*, 48(2):86–92.

Albert Mehrabian. 1996b. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*.

James A Russell and Albert Mehrabian. 1977. Evidence for a three-factor theory of emotions. *Journal of research in Personality*, 11(3):273–294.

Roman Shantala, Gennadiv Kyselov, and Anna Kyselova. 2018. Neural dialogue system with emotion embeddings. In *2018 IEEE First International Conference on System Analysis & Intelligent Computing (SAIC)*, pages 1–4. IEEE.

Xiao Sun, Xinmiao Chen, Zhengmeng Pei, and Fuji Ren. 2018. Emotional human machine conversation generation based on seqgan. In *2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, pages 1–6. IEEE.

Mark A Thornton and Diana I Tamir. 2017. Mental models accurately predict emotion transitions. *Proceedings of the National Academy of Sciences*, 114(23):5982–5987.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Wei Wei, Jiayi Liu, Xianling Mao, Guibing Guo, Feida Zhu, Pan Zhou, and Yuchong Hu. 2019. Emotion-aware chat machine: Automatic emotional response generation for human-like emotional interaction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1401–1410.

Sayyed M Zahiri and Jinho D Choi. 2017. Emotion detection on tv show transcripts with sequence-based convolutional neural networks. *arXiv preprint arXiv:1708.04299*.

Rohola Zandie and Mohammad H Mahoor. 2020. Emptransfo: A multi-head transformer architecture for creating empathetic dialog systems. *arXiv preprint arXiv:2003.02958*.

Peixiang Zhong, Yan Zhu, Yong Liu, Chen Zhang, Hao Wang, Zaiqing Nie, and Chunyan Miao. 2020. Endowing empathetic conversational models with personas. *arXiv preprint arXiv:2004.12316*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93.

Highlight-Transformer: Leveraging Key Phrase Aware Attention to Improve Abstractive Multi-Document Summarization

Shuaiqi LIU, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen
Department of Computing, The Hong Kong Polytechnic University
{cssqliu, csjcao, csryang, cszwen}@comp.polyu.edu.hk

Abstract

Abstractive multi-document summarization aims to generate a comprehensive summary covering salient content from multiple input documents. Compared with previous RNN-based models, the Transformer-based models employ the self-attention mechanism to capture the dependencies in input documents and can generate better summaries. Existing works have not considered key phrases in determining attention weights of self-attention. Consequently, some of the tokens within key phrases only receive small attention weights. It can affect completely encoding key phrases that convey the salient ideas of input documents. In this paper, we introduce the Highlight-Transformer, a model with the highlighting mechanism in the encoder to assign greater attention weights for the tokens within key phrases. We propose two structures of highlighting attention for each head and the multi-head highlighting attention. The experimental results on the Multi-News dataset show that our proposed model significantly outperforms the competitive baseline models.

1 Introduction

Abstractive Multi-Document Summarization (MDS) offers the challenge of generating a comprehensive summary of multiple related documents. It requires summarization models to capture the salient content from input documents. Compared with the previous RNN-based models for abstractive MDS, the Transformer-based models (Gehrmann et al., 2018; Liu et al., 2018; Liu and Lapata, 2019a; Li et al., 2020b) employ the self-attention mechanism to capture the dependencies in input documents, and they can generate better summaries.

Calculating attention weights is a crucial step in the self-attention mechanism. Input documents usually contain some key phrases that convey the salient ideas of input documents. However, existing works have not considered key phrases in determining attention weights of self-attention. Key phrases usually comprise multiple tokens, which should be highly related and serve as a complete grammatical unit in input documents. When testing Transformer-based models, we observe some of the tokens within key phrases only receive small attention weights, which can affect completely encoding key phrases and the salient ideas they convey.

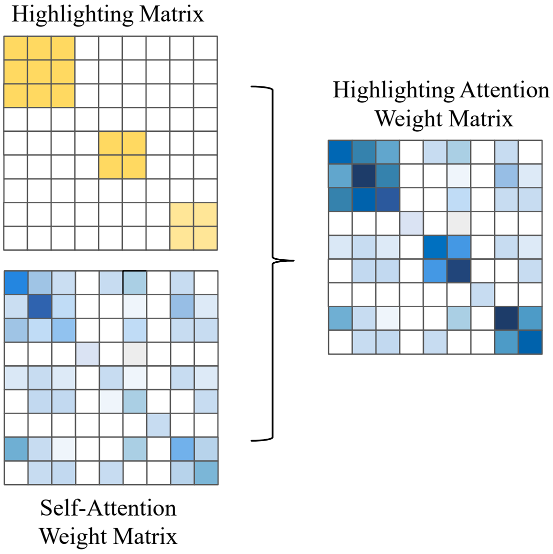

Figure 1: The highlighting mechanism assigns greater attention weights for tokens within key phrases indicated by the highlighting matrix.

In this paper, we propose the Highlight-Transformer, an abstractive summarization model with the highlighting mechanism in the encoder. As depicted in Figure 1, the highlighting mechanism assigns greater attention weights for tokens within key phrases. Furthermore, the highlighting mechanism mainly comprises three parts: the highlighting matrix, the highlighting attention, and the multi-head highlighting attention.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5021–5027
August 1–6, 2021.

Our work is inspired by previous studies in education and psychology that indicate key phrases are important for people to understand (Rello et al., 2014; Hargreaves and Crabb, 2016) and summarize (Benzer et al., 2016; Chou, 2012) the given documents. Highlighting key phrases can help people with dyslexia improve comprehension (Rello et al., 2014; Hargreaves and Crabb, 2016). Their findings can be instructive to improve the self-attention mechanism.

We build a highlighting matrix for each input token sequence to indicate key phrases' positions in the attention weight matrix and phrases' importance values. We propose two structures of highlighting attention for each head to adjust attention weights according to the phrase importance. After comparing the effects of adopting the highlighting attention in the different numbers of heads and layers, we discover that adopting it in a subset of heads surpass adopting it in all heads. Experimental results on the Multi-News dataset (Fabbri et al., 2019) exhibit that our proposed model significantly improves the ROUGE scores (Lin, 2004) of generated summaries.

Our contribution is threefold:

- We present the highlighting mechanism that assigns greater attention weights for the tokens within key phrases.
- We propose the multi-head highlighting attention and two structures of highlighting attention for each head to combine attention weights with the phrase importance.
- Our proposed model significantly outperforms the competitive baseline models on the Multi-News dataset.

2 Related Work

Previous encoder-decoder models (Rush et al., 2015; Nallapati et al., 2016; Paulus et al., 2018; Chopra et al., 2016) equipped with the attention mechanism (Bahdanau et al., 2015) have achieved great performance on abstractive summarization. However, they were found to miss some important content in input documents (Li et al., 2018; Xu et al., 2020). How to retain the key information of input documents in the generated summaries has received increasing attention in the past few years. Some previous works focus on improving the copy mechanism. Gehrmann et al. (2018) utilize the attention masks to restrict copying phrases from the selected parts of an input document. Xu et al. (2020) explicitly guide the copy process with the centrality of each source word. Several papers also explore the potential of enhancing the encoder. Li et al. (2018, 2020a) extend the pointer-generator-based models (See et al., 2017) with a separate LSTM-based encoder to get the keywords' representation and then combine it with the sentence representation. In this work, we explore the potential of leveraging phrase importance as guidance to adjust attention weights in the multi-head self-attention of the Transformer encoder.

3 Model

In this section, we present the Highlight-Transformer, a model with the highlighting mechanism. We introduce its three main components: the highlighting matrix, the highlighting attention for each head, and the multi-head highlighting attention. We focus on the encoder part, and our decoder follows the CopyTransformer model used in (Gehrmann et al., 2018; Fabbri et al., 2019). Each input example of our proposed model includes the source articles, the articles' key phrases, and the phrases' importance values. The automatic key phrases extraction method we used will be introduced in section 4.1.

3.1 Highlighting Matrix

The first step of the highlighting mechanism is to build a highlighting matrix for each input example. It can indicate key phrases' positions in the attention weight matrix and the phrases' importance values. The concatenated source articles can be represented as an input sequence $(t_1, ..., t_n)$ containing n tokens. We use $(p_1, ..., p_k)$ and $(v_1, ..., v_k)$ to denote key phrases and their importance values. For each input example, We build the highlighting matrix $H \in \mathbb{R}^{n\times n}$ with the same shape as the self-attention weight matrix. Assuming a phrase p_r contains b tokens in the input sequence $p_r = (x_a, ..., x_{a+b})$, the phrase's importance value v_r is added to the elements $H_{i,j}$, where $i = a, \ldots, a+b, j = a, \ldots, a+b$, in the highlighting matrix. The phrases can be overlapping or nested, and the token t_i may be contained in c phrases $(p_r, ..., p_{r+c})$, whose importance values are $(v_r, ..., v_{r+c})$. The element H_{ii} is assigned as the sum of the c phrases' importance values.

3.2 Highlighting Attention

The highlighting attention is the key component in our proposed model for adjusting attention weights according to the phrase importance. For the head *m*, the Transformer model (Vaswani et al., 2017) adopts the scaled dot-product attention that operates on a query Q, a key K, and a value V:

$$\mathrm{Attention}(Q, K, V) = W^m V \tag{1a}$$

$$W^m = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{1b}$$

where $W^m \in \mathbb{R}^{n \times n}$, and d_k is the dimensionality of key. In the encoder layers, queries, keys, and values come from the output of the previous layer.

The highlighting matrix H can be used to determine which elements in the attention weight matrix should be increased. We propose two structures of highlighting attention, namely the weighted highlighting attention and the additive highlighting attention, to adjust attention weights according to the phrase importance.

The weighted highlighting attention mainly modifies Equation (1b) to calculate the attention weight matrix W^m for the head *m*. The highlighting matrix H is multiplied by a scalar α, named the brightness factor. The product will be added to the input of the softmax function.

$$W^m = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}} + \alpha H) \tag{2}$$

Since the softmax function applies the exponential function to each input element and divides them by the sum of all these exponentials, the above additive operation can be identical to calculating the weighted average.

$$\mathrm{softmax}(z_i + b_i) = \frac{e^{b_i} e^{z_i}}{\sum_{j=1}^{n} e^{b_j} e^{z_j}} \quad i = 1, \ldots, n \tag{3}$$

The additive highlighting attention is also designed to adjust the attention weight matrix W^m. In Equation (4a), the product of the highlighting matrix H and the scalar α is normalized by the softmax function[1] and added to the original attention weight matrix W_a^m calculated by Equation (1b). And then, elements in W_b^m will be normalized to ensure the sum of the attention weights equals one along the dimension where the softmax conducts.

$$W_b^m = W_a^m + \mathrm{softmax}(\alpha H_m) \tag{4a}$$

$$W_{:,\,j}^m = \frac{W_{b\ :,\,j}^m}{||W_{b\ :,\,j}^m||_1} \quad j = 1, \ldots, n \tag{4b}$$

3.3 Multi-Head Highlighting Attention

In our proposed model, the encoder with d_{model} consists of N layers and h heads. Each encoder layer has two sub-layers: the multi-head highlighting attention layer and the position-wise fully connected feed-forward network. We proposed the multi-head highlighting attention mechanism, which employs the highlighting attention on p highlighted heads and the scaled dot-product attention on the rest of $(h - p)$ normal heads.

$$\begin{aligned} &\mathrm{MultiHead}(Q, K, V) = \mathrm{Heads} W^o \\ &\mathrm{Heads} = \mathrm{Concat}(\mathrm{Head}_1, ..., \mathrm{Head}_h) \\ &\mathrm{Head_i} = \mathrm{Attention}(Q, K, V) \end{aligned} \tag{5}$$

where the projection is a parameter matrix $W^o \in \mathbb{R}^{hd_v \times d_{model}}$. The matrix Head_i is calculated by Equation (1a). The attention weight matrix W of the highlighted heads can be calculated by Equation (2) or (4), and that of the normal heads can be calculated by Equation (1b). The results on all heads will be concatenated and then projected through a feed-forward layer.

4 Experiments

4.1 Data Preparation

We train and evaluate our model on a MDS dataset named Multi-News (Fabbri et al., 2019), in which each example includes multiple news articles about the same event and a human-written summary collected from the website newser.com.

Following the setting in (Fabbri et al., 2019), we truncate each input article to 500/S tokens for the example with S news articles and concatenate the truncated articles into a single document. For each example, we first filter out stopwords and select candidate phrases from these truncated source articles. And then, we use the library named scikit-learn to calculate the candidate phrases' tf-idf values (Salton and Buckley, 1988) as their importance values. These candidate phrases are sorted in descending order of their importance values. We only select the top-10 bigrams or trigrams as key phrases

---

[1] Since the number of key phrases is limited, and the highlighting matrix can be sparse, we mask the zero elements and only conduct the softmax operation on the nonzero elements.

in each example since we observe longer phrases are sparse and more likely to be compressed in summary. Each input example of our proposed model includes the source articles, key phrases together with their L2 normalized tf-idf values.

4.2 Experimental Setting

We adopt a 4-layer encoder and a 4-layer decoder to build our proposed model, in which each layer has eight attention heads. Both the word embedding size and hidden size are set as 512. The maximum size of the vocabulary is set as 50000. Besides, we implement our model with the framework named OpenNMT-py (Klein et al., 2017).

The optimizer is Adam (Kingma and Ba, 2015) with learning rate 2, β_1=0.9, and β_2=0.998. Learning rate warmup is adopted to linearly increase the learning rate over the first 8,000 steps and then decrease it as the setting in (Vaswani et al., 2017). In addition, the brightness factor α in the highlighting attention also progressively decreases at the end of each epoch. Following the setting in (Fabbri et al., 2019), we also apply label smoothing (Szegedy et al., 2016) with smoothing factor 0.1 and dropout (Srivastava et al., 2014) with probability 0.2.

During testing, we use beam search with a beam size of 5. We also use trigram blocking to reduce repetitions. Our models are trained and evaluated on one NVIDIA Tesla P100 GPU.

4.3 Baselines

We compare our proposed Highlight-Transformer model with the following extractive and abstractive summarization methods.

LexRank and TextRank (Erkan and Radev, 2004; Mihalcea and Tarau, 2004) are two graph-based ranking methods that can be used for extractive summarization.

A **tf-idf**-based extractive summarization method (Christian et al., 2016) is evaluated to compare with introducing tf-idf score into our abstractive method.

BertExt (Liu and Lapata, 2019b) stacks inter-sentence Transformer layers on top of the pre-trained BERT (Devlin et al., 2019). We fine-tune this model on the Multi-News training set.

PG and PG-MMR are the pointer-generator (PG) network based summarization models reported by (Lebanoff et al., 2018).

Hi-MAP (Fabbri et al., 2019) extends the PG network into a hierarchical network. The attention distribution of tokens is multiplied by the MMR score of the sentence to which they belong.

Method	R-1	R-2	R-SU
LexRank	38.27	12.70	13.20
TextRank	38.44	13.10	13.50
tf-idf	38.68	12.09	13.54
BertExt	44.27	15.09	17.44
PG	41.85	12.91	16.46
PG-MMR	40.55	12.36	15.87
Hi-MAP	43.47	14.89	17.41
BertAbs	42.21	15.14	16.33
SAGCopy	43.98	15.21	17.65
CopyTransformer	43.57	14.03	17.37
Highlight (Weighted)	**44.62**	**15.57**	**18.06**
Highlight (Additive)	44.29	15.46	17.73

Table 1: Evaluation results on the Multi-News test set.

Highlight (Weighted)	R-1	R-2	R-SU
1/4 Heads 1/2 Layers	**44.62**	**15.57**	**18.06**
1/2 Heads 1/2 Layers	44.25	15.37	17.84
All Heads 1/2 Layers	44.18	15.12	17.70
1/4 Heads All Layers	44.32	15.16	17.82
1/2 Heads All Layers	44.41	15.50	17.84
All Heads All Layers	44.21	15.11	17.72

Table 2: Evaluation results on highlighting different numbers of heads and layers.

SAGCopy (Xu et al., 2020) adds the word centrality score to the linearly transformed hidden state when calculating the copy distribution.

BertAbs (Liu and Lapata, 2019b) adopts the pre-trained BERT as the encoder. A decoder with six Transformer layers is initialized randomly. We fine-tune this model on the Multi-News training set.

CopyTransformer (Gehrmann et al., 2018; Fabbri et al., 2019) adds the copy mechanism (See et al., 2017) to a 4-layer Transformer model. The decoder of our model follows its architecture.

4.4 Results and Discussion

We report the ROUGE F_1 (Lin, 2004) scores, including the overlap of unigrams (R-1), bigrams (R-2), and skip bigrams with a max distance of four words (R-SU). The results of LexRank, TextRank, PG, PG-MMR, Hi-MAP, and CopyTransformer follow Fabbri et al. (2019).

As shown in Table 1, the Highlight-Transformer significantly outperforms these baseline models on all metrics, which proves the effectiveness of the highlighting mechanism. Compared with the additive highlighting attention, the weighted high-

	Win	Lose	Tie	Kappa
Informativeness	46.5%	21.5%	32.0%	0.664
Fluency	29.5%	26.0%	44.5%	0.639
Non-Redundancy	27.5%	25.5%	47.0%	0.624

Table 3: Human evaluation results. “Win” represents the generated summary of our proposed model is better than that of CopyTransformer in one aspect.

	R-1	R-2	R-SU
Highlight-Transformer	**44.62**	**15.57**	**18.06**
w/o brightness	44.38	15.44	17.90
w/o highlight attn	43.57	14.03	17.37
w/o self-attention	42.54	14.40	16.54

Table 4: Ablation study on the Multi-News test set. ”brightness” denotes the brightness factor α and ”highlight attn” denotes the highlighting attention.

lighting attention is more favorable.

We also compare the effects of adopting the weighted highlighting attention in different numbers of heads and layers in the encoder of our proposed model. The results on the test set of Multi-News are summarized in Table 2. It reveals that adopting it in a quarter of the heads and half of the layers achieves the best performance. We discover that adopting highlighting attention in a subset of heads surpasses adopting it in all heads. Besides, applying the multi-head highlighting attention on all layers of the encoder is also not optimal.

Multi-head attention in the Transformer model (Vaswani et al., 2017) is designed for jointly attending to information from different representation sub-spaces. Voita et al. (2019) find the heads in Transformer model trained on the neural machine translation dataset have specialized functions and focus on different types of information. Adopting the highlighting attention in all heads and layers will affect the Transformer-based model to encode other types of useful information and lead to performance degradation.

In addition to automatic evaluation, we performed a human evaluation to compare the generated summaries in terms of informativeness (the coverage of information from input documents), fluency (content organization and grammatical correctness), and non-redundancy (less repetitive information). We randomly selected 50 samples from the test set of the Multi-News dataset. Four annotators are required to compare two models’ generated summaries that are presented anonymously. We also assess their agreements by Fleiss’ kappa (Fleiss, 1971). The human evaluation results in Table 3 exhibit that the Highlight-Transformer significantly outperforms the CopyTransformer in terms of informativeness and is comparative in terms of fluency and non-redundancy.

The ablation study aims to validate the effectiveness of individual components in our proposed model. In Table 4, ”w/o highlight attn” refers to the CopyTransformer model used in (Gehrmann et al., 2018; Fabbri et al., 2019). The results confirm that incorporating the highlighting attention is beneficial for multi-document summarization, and the decreasing brightness factor α also benefits our model’s performance. Besides, we tried replacing the self-attention weight matrices in a quarter of the heads and half of the layers with the highlighting matrices. The performance degradation reveals that it is important to combine the attention weights with the phrase importance instead of directly replacing the attention weights.

5 Conclusion

In this paper, we introduce the Highlight-Transformer, a novel summarization model with the highlighting mechanism in the encoder. The highlighting mechanism assigns greater attention weights for the tokens within key phrases, and it comprises three main parts: the highlighting matrix, the highlighting attention, and the multi-head highlighting attention. Specifically, a block diagonal highlighting matrix is built for each input token sequence to indicate key phrases’ positions and phrases’ importance values. For each head, we propose and compare two structures of highlighting attention. Furthermore, we also compare the effects of adopting the weighted highlighting attention in different numbers of heads and layers in the encoder of our proposed model. The experimental results exhibit the effectiveness of our proposed model. We intend to incorporate more phrase-level and sentence-level information into Transformer-based summarization models and evaluate them on different datasets in future work.

Acknowledgments

This work is supported by the Hong Kong RGC Collaborative Research Fund with project code C6030-18G and Hong Kong Red Swastika Society Tai Po Secondary School with project code P20-0021.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ahmet Benzer, Ayşegül Sefer, Zeyneb Ören, and Sümeyye Konuk. 2016. A student-focused study: Strategy of text summary writing and assessment rubric. *Education & Science/Egitim ve Bilim*, 41(186).

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Mu-hsuan Chou. 2012. Implementing keyword and question generation approaches in teaching efl summary writing. *English Language Teaching*, 5(12):36–41.

Hans Christian, Mikhael Pramodana Agus, and Derwin Suhartono. 2016. Single document automatic text summarization using term frequency-inverse document frequency (tf-idf). *ComTech: Computer, Mathematics and Engineering Applications*, 7(4):285–294.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Sandra Hargreaves and Jamie Crabb. 2016. *Study Skills for Students with Dyslexia: Support for Specific Learning Differences (SpLDs)*. Sage.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141, Brussels, Belgium. Association for Computational Linguistics.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 55–60, New Orleans, Louisiana. Association for Computational Linguistics.

Haoran Li, Junnan Zhu, Jiajun Zhang, Chengqing Zong, and Xiaodong He. 2020a. Keywords-guided abstractive sentence summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8196–8203.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020b. Leveraging graph to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019b. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and R. Socher. 2018. A deep reinforced model for abstractive summarization. *ArXiv*, abs/1705.04304.

Luz Rello, Horacio Saggion, and Ricardo Baeza-Yates. 2014. Keyword highlighting improves comprehension for people with dyslexia. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 30–37.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362, Online. Association for Computational Linguistics.

Phrase-Level Action Reinforcement Learning for Neural Dialog Response Generation

Takato Yamazaki
The University of Tokyo, Japan
`takato@yamazaki.dev`

Akiko Aizawa
The University of Tokyo, Japan
National Institute of Informatics, Japan
`aizawa@nii.ac.jp`

Abstract

Defining a sophisticated action space for a dialog agent is essential for efficient training with reinforcement learning (RL). Recent work introduces discrete latent variables to use as an action space; however, a limitation is that a global vector can contain entangled information such as dialog act, sentence structure, and content. This sacrifices the flexibility of the response generation. In this paper, we propose phrase-level action reinforcement learning (PHRASERL), which allows the model to flexibly alter the sentence structure and content with the sequential action selection. Our model first learns to generate useful phrases during the supervised pre-training, and then further trained to form a response by rearranging the phrases with reinforcement learning. Experiments on the MultiWOZ dataset show that our model achieves competitive results with state-of-the-art models on automatic evaluation metrics, indicating that our phrase-level action space has improved flexibility and is effective for solving task-oriented dialogs.

1 Introduction

Dialog policy optimization is key research to efficiently solving real-world tasks (Rastogi et al., 2020; Budzianowski et al., 2018; Lewis et al., 2017). In neural response generation, which has made remarkable progress in recent years (Vinyals and Le, 2015; Li et al., 2016a; Serban et al., 2017; Bao et al., 2019), many methods that apply reinforcement learning (RL) have been proposed (Li et al., 2016b; Peng et al., 2018; Saleh et al., 2019; Zhao et al., 2019). In those studies, one major issue was how to define an action space. Early research proposed a method in which each word of the response is an action (Li et al., 2016b). However, this has a shortcoming that the generated responses deviate from natural human language (Zhao et al., 2019). A possible reason is that the action space is huge, making it difficult to optimize with RL. Moreover, rewarding only the task accomplishment can cause biased improvement, which leads the model to ignore the comprehensibility of the generated response (Wang et al., 2020a).

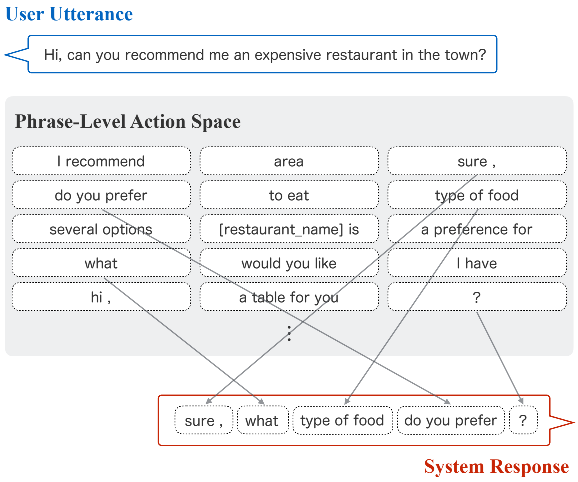

Figure 1: Demonstration of phrase-level action. The model generates useful phrases and rearrange them to form a response.

To overcome such issues, LaRL (Zhao et al., 2019) was proposed, which used a discrete global vector to represent dialog acts. In this method, reinforcement learning is performed only on those discrete latent variables, thus the policy optimization is achieved without affecting the language generation. However, LaRL depends on a single vector from the beginning to the end during the response generation, even though a response may often contain more than one dialog act and contents (Wang et al., 2020b). Due to this, a static, global vector tends to be an entangled representation of multiple dialog acts, sentence structure, and contents. Therefore, using a global vector for action space sacrifices the flexibility of the response generation.

To improve the flexibility of the surface realiza-

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5028–5038
August 1–6, 2021.

tion, we propose phrase-level action reinforcement learning (PHRASERL), in which the model performs action selections in fine-grained semantic units. PHRASERL is based on neural hidden semi-Markov model (HSMM) decoder (Wiseman et al., 2018) which generates typed text-segments from hidden states, and we use them as an action space. This disentangles the generation process: the policy learns to structure a response as a sequence of hidden states, while each hidden state is trained to represent content or a type of phrase. Intuitively, as described in figure 1, our model learns to generate useful phrases during the supervised pre-training, and it is further trained with reinforcement learning to reorder the phrases and form a response.

Experiment results on the task-oriented MultiWOZ dataset (Budzianowski et al., 2018) show that our best performing model outperforms LaRL by far and achieves competitive results with the state-of-the-art models in automatic evaluation. Furthermore, PHRASERL can maintain a high BLEU score, suggesting that the model is flexible in its output response depending on the context. Finally, we study the phrase generation from hidden states in a case study, and show that the hidden state-action space is capable of generating (1) informative response, (2) grammatical sentence, and (3) diverse intentions, which can be considered as requirements for an effective action space. Our code is available at `https://github.com/Alab-NII/PhraseRL`.

2 Related Work

A classical approach for realizing task-oriented dialog systems is the frame-based dialog system (Chen et al., 2017). This model generates a response in a pipeline fashion, by splitting the generation process into three modules: natural language understanding, dialog management, and natural language generation. Natural language understanding converts user utterances to a semantic frame which is considered a dialog state, and a popular method is slot filling (Mrkšić et al., 2017; Ramadan et al., 2018). The estimated dialog state is then passed on to dialog management to determine the next action, which is formulated as a partially-observable Markov decision process (POMDP) (Young, 2006). The action space is represented with hand-crafted dialog acts (Budzianowski et al., 2018; Stolcke et al., 2000) or meaning representations (Balakrishnan et al., 2019). Finally, a natural language generator generates a response, which is often realized with recurrent neural networks (Zhou et al., 2016; Tran and Nguyen, 2017). Our proposed model spans between dialog management and natural language generation, however, our model does not require any hand-crafted representation.

Past works that applied reinforcement learning to dialog models have shown a huge performance improvement in task success (Lewis et al., 2017; He et al., 2018). Li et al. (2016b) proposed a dialog generation method by using deep reinforcement learning with words as action spaces. Although the rewards were carefully designed, it is reported that these models tend to generate incomprehensible responses. Zhao et al. (2019) solved the problem by using discrete latent variables as the action space. Wang et al. (2020a) have extended LaRL and applied hierarchical reinforcement learning technique to decouple the dialog policy and natural language generation. The model is composed of two policy networks; one is the high-level policy which acts on latent dialog act and another is the low-level policy that acts on words. The low-level policy is prone to degeneration, so the paper proposes to use language model discriminator as a reward provider. These models either use words or a global latent variable as an action space, however, our work stands between the two by using phrases for the action space.

3 Preliminaries

In this section, we first explain the characteristics and formulation of the HSMM. We then describe the neural HSMM decoder, which will be the backbone of our proposed method.

3.1 Hidden Semi-Markov Model (HSMM)

In our work, we consider sentences as a sequence of phrases, and the probabilistic model that can represent this is the hidden semi-Markov model (HSMM). The difference between a standard hidden Markov model (HMM) and HSMM is shown in figure 2. While an HMM gives one observation from a hidden state, an HSMM gives a sequence of observations per hidden state. Therefore, if we consider words as observations, hidden states will be phrases.

To represent a variety of sentences with a limited number of hidden states, HSMM is expected to assign the same sequence of hidden states for similar sentences. Figure 3 is an example. As it can

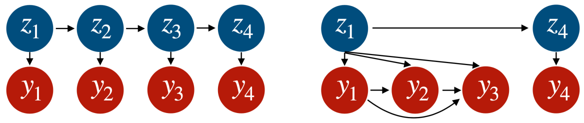

Figure 2: The difference between Hidden Markov Models (left) and Hidden Semi-Markov Models (right)

[what]$_8$ [time]$_{23}$ [would you like]$_{11}$ [to leave ?]$_{34}$
[what]$_8$ [kind of food]$_{23}$ [would you like]$_{11}$ [to eat ?]$_{34}$

Figure 3: An example of sentence segmentation in HSMM. The number represents the index of a hidden state.

be seen, each hidden state has a type; for instance, hidden state #8 and #11 outputs the same phrase, while #23 outputs noun phrases and #34 outputs verb phrases for end of questions. In this way, text-segments assigned to a certain hidden state will be having a similar property.

For our model, we specifically use *conditional* HSMMs which takes a source input x. For each timestep $t \in \{1, \cdots, T\}$, we denote the observations as $y_1 \cdots y_T$ and the discrete hidden states as $z_t \in \{1, \cdots, K\}$. We additionally introduce two latent variables; the length of the current observation sequence, denoted as $l_t \in \{1, 2, \cdots, L\}$, and a binary variable which represents whether the sequence is finished at timestep t, denoted as f_t. The maximum number of hidden states K and observation length L are tunable parameters. An HSMM will be represented with a joint distribution of the observations and the described latent variables:

$$\begin{aligned} p(y,z,l,f|x;\theta) = & \prod_{t=0}^{T-1} p(z_{t+1},|z_t,x)^{f_t} \\ & \times \prod_{t=0}^{T-1} p(l_{t+1}|z_{t+1})^{f_t} \\ & \times \prod_{t=1}^{T} p(y_{t-l_t+1:t}|z_t,l_t,x)^{f_t}. \end{aligned} \tag{1}$$

In other words, an HSMM will be the product of three probabilities: state transition distribution, length distribution, and emission distribution.

3.2 Neural HSMM Decoder

We now introduce a neural HSMM decoder (Wiseman et al., 2018). Figure 4 shows the overview of the decoder model. The aforementioned three distributions can be obtained using trainable parameters. We define the embeddings of the x as

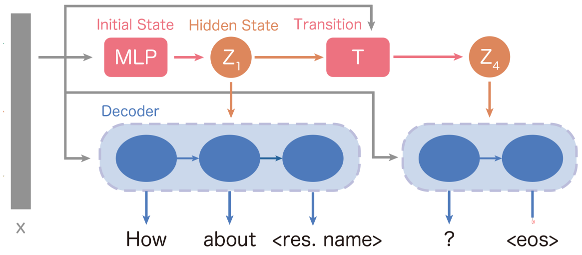

Figure 4: Model overview of neural HSMM decoder.

$\boldsymbol{x} \in \mathbb{R}^d$ and the hidden state z as $\boldsymbol{z} \in \mathbb{R}^d$.

State Transition Distribution For the state transition function $p(z_{t+1}|z_t, x)$, we use $K \times K$ matrix, where sum of each row is 1. We define the state transition matrix as

$$p(z_{t+1}|z_t, x) \propto \boldsymbol{A}\boldsymbol{B} + \boldsymbol{C}(\boldsymbol{x})\boldsymbol{D}(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{K \times m_1}$ and $\boldsymbol{B} \in \mathbb{R}^{m_1 \times K}$ represents state embeddings, and where $\boldsymbol{C} : \mathbb{R}^d \rightarrow \mathbb{R}^{K \times m_2}$ and $\boldsymbol{D} : \mathbb{R}^d \rightarrow \mathbb{R}^{m_2 \times K}$ is a non-linear function parameterized with neural networks. m_1 and m_2 are tunable parameters.

Length Distribution Wiseman et al. (2018) have found that parameterizing length distribution leads to hidden states that specialize in specific output lengths. To avoid that, we simply used uniform distribution for every length probability $p(l_{t+1}|z_{t+1})$.

Emission Distribution For the emission distribution $p(y_{t-l_t+1:t}|z_t, l_t, x)$, we use the product of the token probability. Therefore, the emission distribution is obtained with

$$\begin{aligned} & p(y_{t-l_t+1:t}|z_t = k, l_t = l, x) = \\ & \prod_{i=1}^{l_t} p(y_{t-l_t+i}|y_{t-l_t+1:t-l_t+i-1}, z_t = k, x) \\ & \times p(\langle eop \rangle | y_{t-l_t+1:t}, z_t = k, x) \times \mathbf{1}_{\{l_t = l\}}, \end{aligned} \tag{3}$$

where $\langle eop \rangle$ stands for end-of-phrase token which indicates the end of emission for each hidden state. We use gated recurrent unit (GRU) to compute the token probabilities:

$$\boldsymbol{v}_i = \boldsymbol{W}\mathrm{ReLU}\left(\mathrm{GRU}\left(\left[\boldsymbol{y}'_{i-1}, \boldsymbol{z}^k\right], \boldsymbol{h}_{i-1}\right)\right), \tag{4}$$

where $\boldsymbol{y}'_{i-1}$ is the embeddings of the generated previous token and $\boldsymbol{z}^k$ is the embeddings of the hidden state k. Finally, the probability of the token w will be

$$p(y_{t-l_t+i} = w|z_t = k, l_t = l, x) = v_{i,w}. \tag{5}$$

Training We assume z, l, f of an HSMM is unobservable, so we maximize the marginal likelihood of the emission y given only input x by training. The marginal likelihood of y in HSMMs can be efficiently computed using a dynamic programming algorithm such as backward-algorithm (Murphy, 2002). Using variables β, β^*, the backward-algorithm can be expressed as

$$\begin{aligned} \beta_t(j) &= p(y_{t+1:T}|z_t = j, f_t = 1, c) \\ &= \sum_{k=1}^{K} \beta^*_t(k) p(z_{t+1} = k|z_t = j) \end{aligned} \tag{6}$$

$$\begin{aligned} \beta^*_t(k) &= p(y_{t+1:T}|z_{t+1} = k, f_t = 1, c) \\ &= \sum_{l=1}^{L} [\beta_{t+l}(k) p(l_{t+1} = l|z_{t+1} = k) \\ &\qquad p(y_{t+1:t+l}|z_{t+1} = k, l_{t+1} = l)], \end{aligned} \tag{7}$$

where $\beta_T(j) = 1$. Finally, from the definition $f_0 = 1$, the log-marginal likelihood of y will be:

$$\ln p(y|x;\theta) = \ln \sum_{k=1}^{K} \beta^*_0(k) p(z_1 = k). \tag{8}$$

Here, we compute $p(z_1 = k)$ with a linear layer. Since equations (6) and (7) are differentiable, we can optimize θ by maximizing the log-marginal likelihood $\ln p(y|x;\theta)$ with backpropagation.

4 Proposed Method

The original neural HSMM decoder (Wiseman et al., 2018) was proposed as a data-to-text generation method, so the model needs modification to be applied to dialog response generation. Particularly, we investigate what the HSMM should condition on, in other words, we determine the input x. However, we need to carefully design x because of a known problem of the neural HSMM decoder, which will be explained first. Afterward, we discuss how to improve the response quality by applying reinforcement learning.

4.1 Conditional Source for Neural HSMM Decoder

The neural HSMM decoder is based on the assumption that the output phrases from each hidden state to be independent of each other. However, if the source $\boldsymbol{x}$ is informative enough to capture the interdependence between the phrases, the RNN decoder may fully depend on the source $\boldsymbol{x}$ and ignore the hidden state $\boldsymbol{z}$ for the generation. We will call this problem the *interdependence problem*. To avoid this, we must use weak source input that does not contain enough information to precisely predict the target response.

For our work, we use contextual information (e.g. dialog history, belief state, database results) as a conditional source x. A common practice to embed contextual information is to use a GRU and a linear layer to encode, and as a result, we obtain continuous embeddings $\boldsymbol{x}$. However, continuous embeddings can result in the interdependence problem, since they can theoretically contain infinite information.

To weaken the encoder, we reduce the resolution of the input embeddings $\boldsymbol{x}$ by using discrete embeddings. We define it as an array of N-way categorical variables: $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_M\}$, where each $\boldsymbol{x}_n$ is a N-sized binary vector and M is the number of variables. To obtain this, straight-through Gumbel-softmax (Jang et al., 2017) is applied to the conditional source encoded by a GRU and a linear layer. In our experiments in section 6, we will compare the results of these discrete embeddings with continuous embeddings.

4.2 Response Generation with Reinforcement Learning

To rearrange the invented hidden states of a neural HSMM decoder, we apply reinforcement learning, which we named this method as *phrase-level action reinforcement learning* (PHRASERL). Here, we consider a Markov decision process of input context as state $x \in \mathcal{S}$, hidden states (which represents phrases) as action space $z \in \mathcal{A}$, and task-success rate as rewards $r \in \mathcal{R}$. We define the timestep of hidden state selection as $t' = \{1, 2, \cdots, T'\}$. We consider the combined initial state selection and state transition as policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ and apply REINFORCE algorithm (Williams, 1992). For each reward $r_{t'}$ in time step t', we use discounted reward $G_{t'} = \sum_{k=0}^{T'} \gamma^k r_{t'+k}$ during training. Now, the policy gradient will be:

$$\nabla J(\pi) = \mathbb{E}\left[\sum_{t'=1}^{T'} \nabla_\pi \log \pi(z_{t'}|x) G_{t'}\right]. \tag{9}$$

Note that we do not train the GRU for the emission distribution; we only further train the hidden state transition. For embedding the contexts, we used the pre-trained encoder and did not further update during this RL step.

5 Experimental Settings

5.1 Task Description

For the experiments, we use MultiWOZ dataset (Budzianowski et al., 2018). MultiWOZ is a large-scale task-oriented dialog dataset, which contains seven types of domains such as booking restaurants, hotels, and train seats. We specifically use Dialogue-Context-to-Text Generation task proposed in the original paper. In this task, a model is given an oracle belief state, and the model's goal is to generate an appropriate and informative response. For the evaluation, we use BLEU, Inform Rate, and Success Rate. We also compute the total score, which is used in previous works to compare models in MultiWOZ dataset. Total score is calculated with $\mathrm{BLEU} + (\mathrm{Inform} + \mathrm{Success})/2$.

5.2 Model Details

Duplicated Hidden States While increasing the number of hidden states allows for a more expressive latent model, the computational complexity of the neural HSMM decoder will increase linearly depending on the number of hidden states K. In order to increase the number of hidden states without making the computation heavier, we use the same emission distribution for multiple hidden states as proposed in Wiseman et al. (2018). For instance, if we set the base state as 80 and duplicated 5 times, K will be 400 and we use $z \bmod 80$ for the input into the computation of emission distribution. This way, the model can utilize a large number of hidden states in the state transition, while the model only needs to run the GRU feed-forward for a smaller number of times to compute emission distribution.

Training Details We first train the neural HSMM decoder with supervised learning, and later we further train with reinforcement learning as explained in section 4.2. To embed the context information x, we use a MLP layer for encoding oracle belief state and a GRU for encoding dialog history. For comparison, we trained both continuous and discrete embeddings, which we denote each model as CONT and DISC respectively. To train with reinforcement learning, we use the MultiWOZ RL setup proposed in (Zhao et al., 2019). For the rewards, we use $r_{\text{success}} + r_{\text{inform}} + r_{\text{BLEU}}$.

The average loss is computed with the validation dataset after every epoch, and early stopping is performed after 5 consecutive epochs without improvement. When we determine the number of hidden states, we tested every 10 states from 40 to 120 for base states, and 1, 3, 5, 7 for duplication. In consequence, $K = 400$ (80 base states, duplicated 5 times) produced the best results, and the following evaluations are based on these results. For the vocabulary set, we substituted the words that occurred less than 30 times in the dataset with an unknown tag ($\langle unk \rangle$). We used beam search with a beam size of 5 for the decoding. For more details, refer to Appendix A.

5.3 Baseline and State-of-the-Art Models

Our model is compared with the following models:

- **Baseline** (Budzianowski et al., 2018) is proposed in the original MultiWOZ paper. The model is based on Seq2Seq with attention on the context words.

- **Word-Level RL** further trains the pre-trained baseline on reinforcement learning with the action space of words. It is known that this model often encounters a degeneration problem, in which the generated sentence will diverge from natural human sentences.

- **Latent Action Reinforcement Learning (LaRL)** (Zhao et al., 2019) introduces a discrete latent variable between the encoder and decoder to represent a dialog act. Similar to our model, it first trains on supervised pre-training, then it further trains the dialog policy with reinforcement learning.

- **Hierarchical Disentangled Self-Attention Network (HDSA)** (Chen et al., 2020) introduces hierarchical dialog act. The model is composed of 3 transformer layers which each layer corresponds to each hierarchy of the dialog act. The model switches the self-attention based on the dialog act, which is called the disentangled self-attention.

- **SOLOIST** (Peng et al., 2020) is a transformer-based auto-regressive language model for task-oriented dialog, pre-trained on large and diverse dialog corpora. The model is fine-tuned on MultiWOZ task.

- **MarCo** (Wang et al., 2020b) extends the idea of HDSA and considers a hierarchical dialog

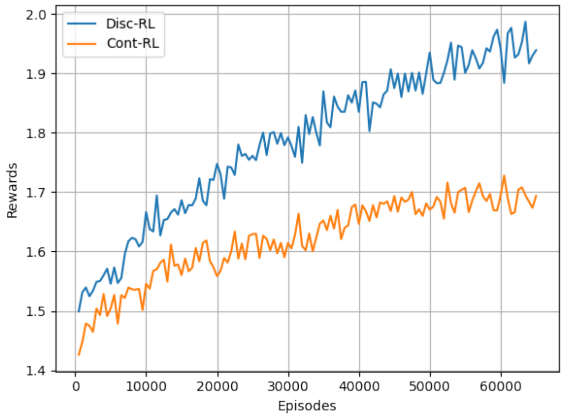

Figure 5: Average rewards for every 500 episodes of our proposed models. DISC-RL and CONT-RL indicates discrete and continuous embeddings, respectively. The maximum reward is 3.0.

act. The difference is that it co-generates the dialog act sequence and the response jointly.

- **HDNO** (Wang et al., 2020a) decouples the dialog policy and natural language generation by applying hierarchical RL. This model uses language model as a reward provider to maintain grammaticality.

Among these models, Word-Level RL, LaRL, and HDNO use reinforcement learning and, therefore, are considered as the main competitors to our PHRASERL. Also, note that HDNO uses rewards from external modules to avoid degeneration, but PhraseRL does not use them in this experiment.

6 Results and Analysis

6.1 Automatic Evaluations

Table 1 shows the results of the automatic evaluation of our models. We firstly see that applying reinforcement learning greatly improves the scores, indicating that hidden states had been an effective action space. We also see that discrete embeddings outperform continuous embeddings in every score. This shows that the model can improve generation performance by alleviating the interdependence problem with weaker encoders. It also can be observed that the discrete embeddings are significantly effective within reinforcement learning models. This can also be seen in the reward graph shown in figure 5, where discrete embeddings have a sharper reward increase compared to continuous embeddings. A possible reason behind this is that discrete embeddings led the phrase generation to be less diverse and strongly typed, which made the agent easier to learn the relation between a hidden state and the generated phrases.

We also compare our best model (DISC-RL) with the past works, and the results are shown in table 2. Our model achieved competitive results with the recently proposed state-of-the-art models, which also is near-human performance.

Comparing with LaRL, our model significantly outperforms in BLEU score even after applying RL. A possible reason for LaRL's low BLEU score is that it cannot fully express the diverse human sentences in a discrete global vector. On the other hand, PHRASERL can broaden the range of expression by dividing the action space into finer semantic units, which enables it to learn more human-like responses. Additionally, the improvement in Inform Rate and Success Rate can also be attributed to the ability of PHRASERL to flexibly select content.

Nevertheless, if we compare with state-of-the-art models without using RL, our model has a lower BLEU score. This suggests that using fixed phrases and arranging them have a drawback in regard to generating grammatical responses, since it lacks word-level flexibility. However, it is surprising to see that it still achieves a competitive score even with such disadvantage.

6.2 Model Analysis

Although our PHRASERL was able to maintain a high BLEU score, this can only be because the model was rewarded with the BLEU score during the training. We also trained the model without using the BLEU score for rewards and the results are shown in table 3. Although there is a slight decrease, it is still largely outperforming the Word-Level RL and LaRL. This indicates that our PHRASERL is resistant to degeneration to some extent, even without adding external modules for rewarding grammaticality as in HDNO. Further improvements can be expected by applying external rewards for avoiding degeneration, though this remains as future work.

Table 4 shows the generated phrases from randomly selected hidden states. By observing the outcome of DISC, we can notice that the common property of the generated phrases are interpretable: state 303 outputs ”verb phrases for the end of question”, state 239 outputs ”beginning of the question”, state 103 outputs ”features of a facility”, state 70 outputs ”back-channeling words”, and state 325

	BLEU (%)	Inform (%)	Success (%)	Total
CONT	15.3	56.7	42.9	65.2
DISC	15.7	71.5	46.3	74.6
CONT-RL	16.7	86.2	67.2	93.5
DISC-RL	**18.0**	**95.6**	**79.3**	**105.4**

Table 1: Evaluation results of MultiWOZ test dataset of our models. RL indicates additional training on reinforcement learning. The total score is computed with BLEU + (Inform + Success) / 2.

	RL	BLEU (%)	Inform (%)	Success (%)	Total
Human	-	-	91.0	82.7	-
Baseline (Budzianowski et al., 2018)	-	18.9	71.3	61.0	85.0
HDSA (Chen et al., 2020)	-	23.6	82.9	68.9	99.5
SOLOIST (Peng et al., 2020)	-	18.3	89.6	79.3	102.8
MarCo (Wang et al., 2020b)	-	20.0	92.3	78.6	105.5
Word-Level RL (Zhao et al., 2019)	✓	1.4	80.5	79.1	81.2
LaRL (Zhao et al., 2019)	✓	12.8	82.8	79.2	93.8
PHRASERL (Ours)	✓	18.0	95.6	79.3	105.4
HDNO (Wang et al., 2020a)	✓	18.9	96.4	84.7	109.5

Table 2: Evaluation results of MultiWOZ test dataset compared with previous works. Our PHRASERL is the result of DISC-RL. The results were obtained from corresponding papers.

outputs ”noun phrases for domains”. This indicates that the hidden states in Disc are strongly typed. Although phrases of CONT also seems to be typed, some states such as state 138 have multiple types. This is due to the interdependence problem because the RNN can recognize which type to use depending on the conditional input x.

6.3 Case Study

To verify that the hidden states are a valid and flexible action space, we qualitatively validate the generated phrases. Figure 6 shows the generated phrases from user input and possible responses that can be formed by reordering the phrases. The possible responses were generated from hidden states that were reordered by hand. We consider three criteria for a valid and flexible action space in MultiWOZ dataset: the model must be able to (1) inform appropriate content (e.g. restaurant name, departure time), (2) generate grammatical sentences, and (3) generate diverse dialog acts.

Content Appropriate contents for this case would be the area of the restaurant ([value_area]), price range ([value_pricerange]), name of the restaurant ([restaurant_name]), and type of food ([value_food]). These entities appear at least once in the generated phrases, and the model can select the content by acting on the hidden states. Furthermore, as shown in the third and fourth examples of possible responses, the model can use a similar hidden state sequence for generating similarly structured responses, but it can still tweak the content depending on their strategy.

We further investigated if the contents are sufficiently provided with the hidden states. We counted the number of cases in the test set where all the entities in the golden response were contained in the generated phrases. As a result, 83.6% of the cases had sufficient information in the generated phrases, which we consider enough because the model may have other response options. Therefore, we can conclude that the first condition has been met.

Grammar Although there remains the possibility of generating ungrammatical sentences, the possible responses in figure 6 show that an appropriate sequence of hidden states will allow generating fluent responses.

Dialog Act As shown in the bottom section in figure 6, the four response samples have different intentions. For instance, the first response asks the type of food to the user, while the second response recommend a restaurant and ask for a booking. Particularly, a second example contains two different dialog acts (recommend and offer booking), but the model can choose to finish with the first sentence to just recommend. This shows that our PHRASERL have disentangled representation for each dialog acts. Therefore, we can conclude that the model can flexibly select from several intentions.

Finally, in table 5, we show an example response

	BLEU (%)	Inform (%)	Success (%)	Total
$r_{\text{success}} + r_{\text{inform}} + r_{\text{BLEU}}$	18.0	95.6	79.3	105.4
$r_{\text{success}} + r_{\text{inform}}$	17.1	95.7	78.9	104.4

Table 3: Evaluation results of DISC-RL with different rewards in MultiWOZ test set.

CONT				
State 190	State 138	State 4	State 114	State 85
what type of attraction what day what area what type of food what information	to book a ticket a different type of to book it a hotel or to know	in the [area] located in the [area] at [time] free wifi free to enter	in the [area] . . that day . with that . available .	the address is [address] it leaves at [time] the postcode is [postcode] the price is [price] it is [pricerange]
DISC				
State 303	State 239	State 103	State 70	State 325
to stay ? ⟨eos⟩ to dine ? ⟨eos⟩ to dine in to book ? ⟨eos⟩ to arrive ? ⟨eos⟩	what day what time how many tickets what area what type of food	free internet and parking free to enter located at [address] [pricerange] -ly priced free wifi and parking	sure , okay , you are welcome . i am sorry , yes ,	any restaurant -s a lot of attractions any attractions any hotel -s a few

Table 4: Top-5 frequently generated phrases from randomly selected hidden states. ⟨eos⟩ indicates end-of-sentence.

User Input: `I am looking for an [value_pricerange] restaurant in the center of town.`

Generated Phrases:

Phrase	**# of hidden states**
in the [value_area]	8
?	6
a good choice	5
.	4
i have [value_count]	3
[value_food] , [value_food] ,	3
price range	2
would you like	2
to try	2
a preference ?	2
i recommend [restaurant_name]	2
[value_food] or [value_food]	2
the [restaurant_name]	2
to try a different	2
in the [value_pricerange]	2
. would you like	2
[value_pricerange] restaurant -s	2
there are no	2
there are [value_count]	2
for you ?	1
what type of food	1
me to book [value_count]	1
a few	1
okay ,	1
do you have a	1
a table for you	1
would you like me	1
restaurant -s	1
i have [value_count] options	1
you would like	1
a different area	1
. it s	1
. is there	1
a particular area	1
. do you prefer	1
i am sorry ,	1
. can i book	1
and [restaurant_name]	1
preference for price range	1
cuisine preference ?	1
. do you have	1
. which	1
would you be interested	1
there are [value_count] options	1

Possible Responses:

```
okay , i have [value_count] options . what type of food would you like ?
i recommend [restaurant_name] . can i book a table for you ?
there are no [value_pricerange] restaurant -s . would you like to try a different area ?
there are no [value_pricerange] restaurant -s . would you like to try a different price range ?
```

Figure 6: A case study for generated phrases of DISC. In the second section, we show the number of hidden states that output the same phrase. The third section shows possible responses which were generated from inputting sequence of hidden states ordered by hand. The colors indicate the corresponding phrases.

from MultiWOZ test dataset. We first see that CONT is not generating a grammatical response. This may be due to the interdependence problem that tries to output various sentences in the same hidden state sequence. On the other hand, the rest of our models can generate a grammatical and in-context response. We also observe that while models without RL generate plausible responses, RL models provide a more informative response by including hotel names and phone numbers. The reason is that the model often receives rewards by conveying those information.

7 Conclusion and Future Work

In conclusion, this paper proposes a phrase-level action reinforcement learning for neural response generation. A neural HSMM decoder is introduced to learn hidden states that output typed phrases, and we used them as the action space for reinforcement

Context	i am looking for a place to stay in the [value_area] of the city . i would prefer a [value_count] star hotel please .
Human Response	there are several guesthouses available . do you have a price preference ?
Baseline	i have [value_count] hotel -s in the [value_area] . would you like to book a room ?
CONT	yes , i have [value_count] a few in the [value_area] . would you like a hotel or guesthouse ?
DISC	sure , there are [value_count] options . is there a particular price range ?
CONT-RL	i have the [hotel_name] in the [value_area] . the phone number is [hotel_phone] .
DISC-RL	[hotel_name] is in the [value_area] . would you like me to book it for you ?

Table 5: Example responses from baseline and our models. ⟨unk⟩ represents unknown word of the model.

learning. Our experiments on MultiWOZ dataset have shown that the model is capable of generating flexible outputs with RL, achieved competitive results to state-of-the-art models. A possible future direction is to apply hierarchical reinforcement learning, where the high-level policy determines the dialog act of the entire response, while low-level policy designs the syntax of the response.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 21H03502. The authors like to thank Taichi Iki, Takuma Udagawa, and Taku Sakamoto for their constructive feedbacks.

References

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2019. Plato: Pre-trained dialogue generation model with discrete latent variable. *arXiv preprint arXiv:1910.07931*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *arXiv*, 1.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2020. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 3696–3709.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–13.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2:994–1003.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

Kevin P Murphy. 2002. Hidden semi-markov models (hsmms). *unpublished notes*, 2.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. SOLOIST: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *arXiv*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*.

Osman Ramadan, Paweł Budzianowski, and Milica Gasic. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 432–437.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Abdelrhman Saleh, Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, and Rosalind Picard. 2019. Hierarchical Reinforcement Learning for Open-Domain Dialog. *arXiv*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 3295–3301.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Van-Khanh Tran and Le-Minh Nguyen. 2017. Natural language generation for spoken dialogue system using RNN encoder-decoder networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 442–451, Vancouver, Canada. Association for Computational Linguistics.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020a. Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system. *arXiv preprint arXiv:2006.06814*.

Kai Wang, Junfeng Tian, Rui Wang, Xiaojun Quan, and Jianxing Yu. 2020b. Multi-domain dialogue acts and response co-generation. *arXiv preprint arXiv:2004.12363*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2018. Learning Neural Templates for Text Generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 3174–3187.

Steve Young. 2006. Using pomdps for dialog management. *2006 IEEE ACL Spoken Language Technology Workshop, SLT 2006, Proceedings*, pages 8–13.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1208–1218.

Hao Zhou, Minlie Huang, and Xiaoyan Zhu. 2016. Context-aware natural language generation for spoken dialogue systems. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2032–2041, Osaka, Japan. The COLING 2016 Organizing Committee.

A Training Details

Table 6 shows the parameter used for the experiments.

Supervised Learning	
Batch Size	32
Word Embedding	128
State Embedding	128
Context Encoder RNN	GRU (256)
Decoder RNN	GRU (256)
Optimizer	Adam (lr=1e-3)
Dropout	0.5
Transition Matrix	$m_1 = 64, m_2 = 32$
Number of Hidden States K	400 (80 base states duplicated 5 times)
Maximum Emission Length L	4
Categorical Embeddings $M \times N$ (DISC Only)	10×10
Gumbel Softmax Temp. τ (DISC Only)	1.0
Reinforcement Learning	
Discount Rate γ	0.99
Optimizer	Adam(lr=1e-4)

Table 6: Parameters used in the experiments.

Automatic Speech Recognition in Sanskrit: A New Speech Corpus and Modelling Insights

Devaraja Adiga[1*] `pdadiga@iitb.ac.in`
Rishabh Kumar[1*] `krrishabh@cse.iitb.ac.in`
Amrith Krishna[2] `ak2329@cam.ac.uk`
Preethi Jyothi[1] `pjyothi@cse.iitb.ac.in`
Ganesh Ramakrishnan[1] `ganesh@cse.iitb.ac.in`
Pawan Goyal[3] `pawang@cse.iitkgp.ac.in`

[1]IIT Bombay, Mumbai, India; [2]University of Cambridge, UK; [3]IIT Kharagpur, WB, India

Abstract

Automatic speech recognition (ASR) in Sanskrit is interesting, owing to the various linguistic peculiarities present in the language. The Sanskrit language is lexically productive, undergoes euphonic assimilation of phones at the word boundaries and exhibits variations in spelling conventions and in pronunciations. In this work, we propose the first large scale study of automatic speech recognition (ASR) in Sanskrit, with an emphasis on the impact of unit selection in Sanskrit ASR. In this work, we release a 78 hour ASR dataset for Sanskrit, which faithfully captures several of the linguistic characteristics expressed by the language. We investigate the role of different acoustic model and language model units in ASR systems for Sanskrit. We also propose a new modelling unit, inspired by the syllable level unit selection, that captures character sequences from one vowel in the word to the next vowel. We also highlight the importance of choosing graphemic representations for Sanskrit and show the impact of this choice on word error rates (WER). Finally, we extend these insights from Sanskrit ASR for building ASR systems in two other Indic languages, Gujarati and Telugu. For both these languages, our experimental results show that the use of phonetic based graphemic representations in ASR results in performance improvements as compared to ASR systems that use native scripts.[1]

1 Introduction

Sanskrit is a language with fairly advanced disciplines of phonetics (*Śikṣā*), prosody (*Chandas*), and grammar (*Vyākaraṇa*). The language has a rich oral tradition and it tends to follow a phonemic-orthography resulting in systematic grapheme-phoneme correspondences. Connected speech leads to phonemic transformations in utteracnes, and in Sanskrit this is faithfully preserved in writing as well (Krishna et al., 2018). This is called as Sandhi and is defined as the euphonic assimilation of sounds, *i.e.*, modification and fusion of sounds, at or across the boundaries of grammatical units (Matthews, 2007, p. 353). Phonemic orthography is beneficial for a language, when it comes to designing automatic speech recognition Systems (ASR), specifically for unit selection at both the Acoustic Model (AM) and Language Model (LM) levels.

Regardless of the aforementioned commonalities preserved in both the speech and text in Sanskrit, designing a large scale ASR system raises several challenges. The Unicode encoding for the native scripts in Sanskrit, both in Roman and Devanāgari, does not preserve the correspondence with the phonemic encoding. Further, mapping the graphemes in Unicode to the corresponding phonemes either leads to ambiguity and redundancy or often requires multi-grapheme combinations.

The language is lexically productive, which results in long compound words with multiple components in usage. This results in the speakers segmenting the compounds at arbitrary lexeme boundaries of the compound, as it need not always be possible to utter the compound in one breath and also to convey the meaning clearly. Similarly, such arbitrary segmentations at the word boundaries are possible in utterance of long text sequences where multiple lexical items are fused together via Sandhi. These segmentations are accompanied with the corresponding Sandhi based transformations, resulting in a new phonetic sequence different from the original sequence. Finally, Sanskrit might be one of those rare natural languages where the number of non-native proficient speakers are

---

[*] Joint first author

[1] Dataset and code can be accessed from `www.cse.iitb.ac.in/~asr` and `https://github.com/cyfer0618/Vaksanca.git`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5039–5050
August 1–6, 2021.

manifold in comparison to the native speakers (Krishna et al., 2020). This makes the ASR task further challenging, as the speakers are prone to carry their influence from their corresponding mother tongues into the Sanskrit utterances as well.

While there exist several computational models for processing Sanskrit texts (Kulkarni, 2013; Kumar et al., 2010; Shukla et al., 2010; Kulkarni et al., 2010a; Goyal et al., 2012; Kulkarni et al., 2010c; Mishra et al., 2013; Saluja et al., 2017; Anoop and Ramakrishnan, 2019; Krishna et al., 2021), large scale systems for processing of speech in Sanskrit, are almost non-existent. First, we present a new dataset, with 78 hours of speech covering about 46,000 sentences, for ASR in Sanskrit. Keeping the rich and long cultural heritage the language carries, we prepare our dataset to be diverse both chronologically and in terms of the domain coverage. Further, the dataset contains utterances from 27 different speakers, representing 6 different native languages. The dataset splits have disjoint speakers, with 12 in the training and 5 each in the validation, test and out-of-domain test data sets. Further, we explicitly mark the segmentation decisions made by a speaker to segment long compound words and fused phrases and include the corresponding transformations due to sandhi.

Using this dataset, we propose a new, large-vocabulary Sanskrit ASR system, which, to the best of our knowledge, is the first such system for Sanskrit. The phonemic orthography followed in Sanskrit has influenced our design choices in terms of unit selection at the level of the acoustic and language models. We investigate three different encoding schemes used to model LM tokens, namely, word-based encoding, byte pair encoding (BPE) and a new vowel split encoding inspired by existing linguistic theories of syllabic structure popularly used within text-to-speech systems (Kishore and Black, 2003; Mishra et al., 2013). Further, to address the redundancy issues in Unicode representations, we make use of the Sanskrit Library Phonetic (SLP1) encoding scheme proposed by Scharf and Hyman (2011). SLP1 is designed such that it preserves the phonemic orthography. Building on the study by Scharf and Hyman (2011), we focus on two graphemic representations only, *viz.*, native script (Devanagari) and SLP1.

Finally, we extend our insights to model ASR systems for two more Indian languages, *viz.*, Telugu and Gujarati. We extend the SLP1 to include graphemes relevant for these languages which are missing from Sanskrit. We report the performance of these ASR systems on two publicly available ASR datasets.

Our main contributions in this work are:
1) We present (in Section 2) a new, large vocabulary Sanskrit ASR system and the first ever ASR-based study for Sanskrit using a new, large and diverse, labeled speech corpus वाक्सञ्चयः (/Vāksañcayaḥ/).
2) We investigate (in Sections 3 and 4) different modeling choices for both acoustic models and language models in Sanskrit ASR systems, along with different graphemic representations. We propose a new word segmentation technique based on splitting at vowels that can be used with both the acoustic model and the language model.
3) We also contextualize our findings for Sanskrit by providing comparisons on ASR systems built for two other Indian languages, *viz.*, Gujarati and Telugu.

2 A new Sanskrit Speech Corpus: वाक्सञ्चयः(/Vāksañcayaḥ/)

Our corpus वाक्सञ्चयः (/Vāksañcayaḥ/), has more than 78 hours of data with an overall vocabulary size of 91,000 words and recordings of about 46,000 sentences, each with a sampling rate of 22 KHz. The contents span over 3 time periods categorised into pre-classical literature (1,500 BCE - 100 BCE), classical literature (300 CE - 800 CE) and modern literature (900 CE to now). The corpus is intended to address the challenges in interfacing the speech and the text covered by the disciplines of phonetics (*Śikṣā*), and grammar (*vyākaraṇa*). Hence, we confine our corpora to those written only in prose (Gadya)[2]. In the Sanskrit literature, frequency of commonly used words changes

Dataset	Speakers	Hours	Utterances
Train	12	56	34,309
Validation	5	8	3,337
Test	5	10	5,531
Out-of-domain Test	5	5	2,618

Table 1: Overview of Sanskrit speech corpus.

---

[2] We do not include verses in our current dataset, as modelling ASR systems for verses would require additional resources on both the acoustic model and the language model fronts.

from one topical domain to another, specifically one *Śāstra* (branch of learning) to another (Adiga et al., 2018). Our corpus contain samples from diverse domains, including philosophy, literature, commentary on poetry and grammar. It also includes contemporary recordings such as stories, live lectures, spiritual discourse and radio program/podcast, so that collecting a wide range of Sanskrit vocabulary.

The recordings were primarily collected with the help of volunteers, recording their speech by using the Recorder app on Android phones and the Audacity platform, and from various sources available online.[3] oTranscribe[3] was used to transcribe the audio files. We had a total of 9 volunteers for recording and 18 unique speakers for the content collected online. Each of these speakers are proficient Sanskrit speakers, with at least an undergraduate or equivalent degree in Sanskrit. These speakers are native speakers of one of the 6 following Indic languages, Hindi, Kannada, Malayalam, Marathi, Tamil and Telugu. In Table 1, we provide the details of the training/validation/test splits for our corpus, वाक्सञ्चयः (/Vāksañcayaḥ/). The speakers in all these 4 splits, train, validation, test, and out-of-domain test sets are disjoint. The out-of-domain test dataset is a stratified sampled dataset, consisting of speech samples from 5 unique speakers. Two of these were added to include utterances in Sanskrit from speakers with more pronounced influence of their native languages (in Hindi and Tamil). The domain of the training dataset primarily is a speech collection of readings from various well known texts. Further, the speech in the training data is in accordance with the traditional phonetic prescriptions of Sanskrit (Śikṣā). Hence, the remaining three in the out-of-domain test set were added to include utterances from different speech domains, extempore discourse, lecture and radio program, differing from the speech domain in the training set. The radio program is studio produced, while the other two are live recorded.

2.1 Challenges in Sanskrit ASR

In this section, we describe various linguistic phenomena that are important to consider when preparing datasets and building ASR systems for Sanskrit with the help of illustrative examples.

Word Length: The tokens in Sanskrit texts can be very long owing to "Sandhi" and the lexically productive process of compounding (``Samāsa"). For instance, consider a compound word, वागर्थप्रतिपत्तये(/vāgarthapratipattaye/). It forms a 19 letter word in SLP1 (vAgarTapratipattaye), and is formed by combining the three Sanskrit stems वाक्, अर्थ, प्रतिपत्ति (/vāk, artha, pratipatti/), as per the rules of Sandhi and Samāsa. In Figure 1, we present the distribution of the number of characters (in SLP1 format) per word across the three languages that we experimentally analyse in this work, *viz.*, Sanskrit, Telugu and Gujarati. The plots are normalized with respect to the size of the vocabulary. The average word length is much higher in Sanskrit (10.75) compared to Gujarati (7.79) and Telugu (9.35). Table 2 compares the distribution of number of characters (*wrt* SLP1) per word in the training vocabulary of three ASR datasets. More than 26% of the words in Sanskrit have length exceeding 12 characters.

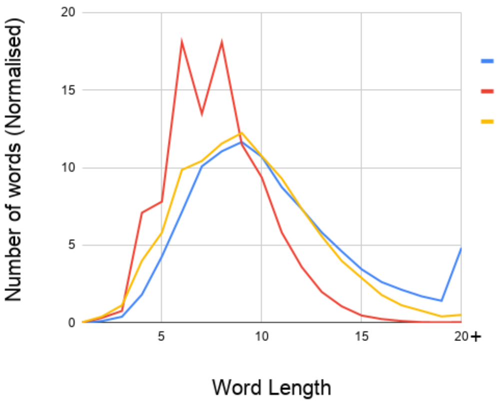

Figure 1: Distribution of Number of characters (in SLP1 representation) per word across Sanskrit, Telugu and Gujarati

Char count(N)	Sanskrit	Telugu	Gujarati
$N \leq 6$	13.79%	21.27%	34.11%
$6 < N \leq 12$	59.56%	61.60%	61.82%
$N > 12$	26.65%	17.13%	4.07%

Table 2: Distribution of number of characters (*wrt* SLP1) per word in three ASR datasets

Sandhi: Sandhi can occur between successive words in a sentence or between the lexemes in a compound, or between the morphemes of the

---

[3] The URLs of the tools and the list of the texts we use are available in the supplementary material.

a) nityaśśabdārthasambandhaḥ
b) nityaḥ śabdārthasambandhaḥ
c) nityaḥ śabdārtha sambandhaḥ
d) nityaśśabdārtha sambandhaḥ

Figure 2: a) Utterance of a sequence without any pause b) Sandhi split at word boundary c) Sandhi based segmentation d) utterance splits without sandhi

same word. While recognising longer sequences due to sandhi and compounding is a challenge in itself, the external sandhi gives rise to the issue of arbitrary points of segmentation performed by speakers at the time of utterance. Figure 2a shows text-sequence where the sequence contains a word *nityaḥ* and a compound *śabdārthasambandhaḥ* fused together via Sandhi. Further, *śabdārthasambandhaḥ* is a compound with *śabda*, *artha* and *sambandha* as its components. While it is expected to be uttered without any pause after considering *Samhitā* (Aṣṭādhyāyī-1-4-109), a speaker may choose to segment at the lexical boundaries as shown in Figure 2c and Figure 2d. However in doing so, a proficient speaker would prefer a sequence similar to Figure 2a or 2b, rather than Figure 2c or 2d. This is because, the former two, though result in phonetic transformations, preserve the syntactic and semantic validity of the sequence. However, the latter do not preserve the syntactic and semantic validity of the sequence. Similarly, there are cases where there can be phonetic transformations between the bound morphemes and the free morpheme of a word. These transformations do not result in any modification to the word, other than phonetic variations. However, this makes it challenging for an ASR system. The case of Diphthongs is the quite prevalent under these cases.

In Diphthongs, which can occur both at internal or external sandhi, the independent vowel can only occur at the start of a word. Any vowel appearing in the middle of a word either gets converted to a dependent vowel or a diphthong. When a word ending with "ए(/ē/) or ओ(/ō/)" and followed by any vowel, then ending will be changed to either "अय्(/ay/) or अव्(/av/) respectively" or "अ(/a/)". For example विष्णो+ इह(/viṣṇo+iha/) will get converted to विष्णइह(/viṣṇaiha/) or विष्णविह(/viṣṇaviha/).

Unicode encoding: The Unicode encoding for the native scripts in Sanskrit, similar to several indian languages, does not preserve the correspondence with the phonemic encoding. Further, mapping the graphemes in Unicode to the corresponding phonemes either suffers from ambiguity and redundancy or often requires multi-grapheme combinations. For instance, consider the word वागर्थाविव(/vāgarthāviva/) in Sanskrit.Here the graphemes in Devanagari 'व ◌ा ग र ◌् थ ◌ा व ◌ि व'and Roman (v ā g a r t h ā v i v a) do not exhibit a one-to-one mapping with the phonemes. For instance, a single grapheme (e.g., ग) may correspond to 2 phonemes while two graphemes (e.g., र ◌् in devanagari, 't h' in roman) may correspond to a single phoneme.

3 Unit Selection Alternatives for AM and LM

In this section, we discuss different alternatives for identifying units of the language model and the acoustic model that we subsequently employ in our experimental evaluation and analysis.

3.1 Alternatives for Graphemic Representation

The Unicode standard for native scripts of Sanskrit : Devanagari, Gujarati and Telugu faces challenge for computational language processing due to redundancy in mappings between phonemes and graphemes as previously discussed. So for Sanskrit, we use Sanskrit Library Phonetic encodings (SLP1) designed by Scharf and Hyman (2011). This encoding gives unique one-to-one mapping to the phoneme. However Gujarati possesses extra native characters such as ઍ(/e/), ઑ(/o/) . Telugu also possesses extra characters such as ఎ(/e/), ఒ(/o/), and ఱ(/r̲/). So we extend SLP1 to fit to the character set of Gujarati and Telugu in our experiments.

3.2 Alternatives for Sub-word units

One possibility for deriving subword units for the language modeling is to segment words in Sanskrit based on Sandhi rules. However, Sandhi splitting can change some phonemes corresponding to the words in almost all cases. Consider the word रामायेदम् = रामाय+ इदम् (/rāmāyedam/ = /rāmāya/+/idam/), wherein the vowel ए (/e/) is changed into अ+ इ (/a/+/i/) after performing Sandhi-based splitting. This leads to a mismatch between the speech transcript and the speech audio, potentially creating further complications for

ASR.

3.2.1 BPE-based Word Segmentation

Byte pair encoding (BPE) is a simple data compression algorithm that iteratively replaces the frequently occurring subword units with a single unused byte (Gage, 1994). This technique was first adopted to model rare words using subword units in neural machine translation (Sennrich et al., 2016). Interestingly, BPE has been explored for learning new vocabulary for poetry to prose conversion in Sanskrit (Krishna et al., 2019). We consider the benefits of using BPE as a subword unit for Sanskrit ASR.

While BPE is a purely data-driven segmentation strategy, we next present a linguistically motivated segmentation approach that might be aligned with finding syllable units for ASR that are more phonetically compliant. We refer to this technique as *vowel segmentation.*

3.2.2 Vowel Segmentation

Splitting the tokens based on vowels and adjacent consonants is inspired by the identification of metres in Sanskrit prosody, where the metre of a verse is identified by using syllable segmentation, followed by identification of syllable weights and it's combinations (Melnad et al., 2013). The syllable weight of a syllable can either be laghu (light)(represented by the symbol ।) or guru (heavy)(represented by the symbol ऽ). Syllables with short vowels generally form Laghu and those with loing vowels form a Guru. Also, when a short vowel is followed by a conjunct consonant or Anusvāra (nasal sound /ṃ/) or Visarga (voiceless glottal fricative /ḥ/), the short vowel now becomes Guru. E.g., the *Laghu-Guru* mapping of "अन्यानि संयाति(/anyāni saṃyāti/)" is "ऽऽ। ऽऽ।". In prior work involving Indian languages for TTS, Kishore et al. (2002) proposed various syllabification rules for words. Herein (with a few exceptions), if a vowel is followed by 3 or more consonants, only the first following vowel is grouped with the preceding vowel to form the subword unit.

Our proposed algorithm for vowel segmentation (VS) is outlined in Algorithm 1. We propose segmenting words at vowel boundaries to extract the units for which alignment with speech is learnt within the ASR system. For acoustic models, an effective unit of a word for ASR would arguably be the syllable (Lee et al., 2013). Representing a word in terms of syllables demands the mapping of a word from graphemes to phonemes. To create syllable units, phonemes are then combined together based on the *sonority sequencing principle* (Clements, 1990). Absence of accurate syllabifiers for Indian languages restricts the use of syllables as units for learning alignment. Our approach produces units which can be viewed as a rough approximation to a syllable. A syllable is composed of three parts *viz.,* onset, nucleus and coda, where nucleus has the highest sonority and is always a vowel. In our approach, the onset is always one or zero consonants and the coda is zero or *n-1* consonants if the nucleus is followed by *n* consonants. It is also observed in the pronunciation of conjunct consonant by professional speakers that the beginning part of conjunct consonant gets associated more with the preceding vowel than the following. We consider nasal Anusvāra (◌ं), Chandrabindu (◌ॅ or ◌ँ), and Visarga (◌ः) to be part of the consonant set. For example, in Sanskrit, the units for a word उद्यानः (udyāna , park) will be 'उद् या नः(/ud yā na /)' and subword units of the Telugu word తల్లితండ్రులు(/tallitaṃḍrulu/) will be 'తల్ లి తండ్ రు లు(/tal li taṃḍ ru lu/)'.

ALGORITHM 1: Vowel segmentation algorithm for Indian languages

```
Input: word in Indian language
Output: Vowel segments in output
output = "";
for each graphemic unit $c_i$ in word do
    if $c_i$ is V then
        if $c_{i+1}$ is V then
            output += $c_i$ + " ";
        else if $c_{i+1}$ is C and $c_{i+2}$ is C then
            output += $c_i$;
        else if $c_{i+1}$ is C and $c_{i+2}$ is V then
            output += $c_i$ + " ";
    else
        if $c_{i+1}$ is V then
            output += $c_i$;
        else if $c_{i+1}$ is C and $c_{i+2}$ is C then
            output += $c_i$;
        else if $c_{i+1}$ is C and $c_{i+2}$ is V and
          $c_{i+2}$ is first vowel of the word then
            output += $c_i$;
        else if $c_{i+1}$ is C and $c_{i+2}$ is V then
            output += $c_i$ + " ";
```

3.3 Vocabulary size as a function of graphemic unit

For each choice of graphemic unit (*viz.* native script and SLP1) described in Section 3.1, we study three different units for the acoustic modeling (AM) in ASR, *viz.,* graphemic unit and vowel segmentation for Sanskrit and also phonemic unit across the two other representative Indian languages *viz.*, Gujarati and Telugu. Whereas, for language modeling (LM), we study word, BPE and VS based units. In Figure 3, we report the vocabulary size based on each of these *three* different unit selections and contrast the sizes with that of two extreme hypothetical systems - one that considers the entire word as a single unit for AM and the other that treats the phoneme as a single unit for AM. Note that while phonetic dictionaries are available for Telugu and Gujarati, our dataset for Sanskrit does not have an accompanying phonetic dictionary. We present the variation in vocabulary size as a function of the graphemic unit (native script vs. SLP1). In both Gujarati and Telugu, we point out that the number of SLP1 graphemic units almost coincide with the number of phonemes, while the native script-based graphemes are much larger in number compared to phonemes.

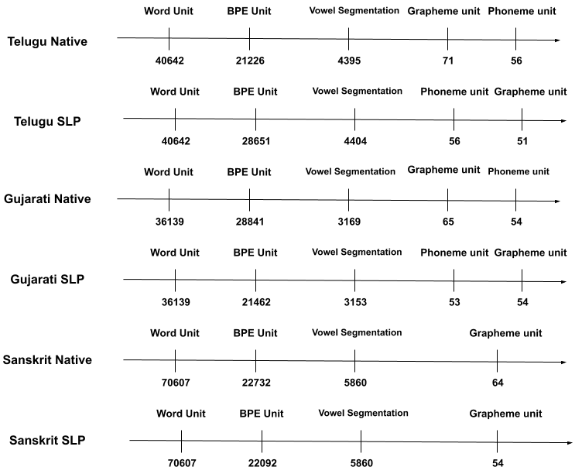

Figure 3: Variation in vocabulary size as a function of the Graphemic unit (native script vs. SLP1) depicting the vocabulary sizes of the whole word units, vowel segment units, BPE units, phonemes and graphemes for ASR data across the three languages including our new Sanskrit ASR data

We can also roughly estimate the extent of data sparsity in terms of vocabulary size in each setting - larger the vocabulary size, higher is the chance of data sparsity. We note that data sparsity is minimal for graphemes and highest for a hypothetical system where whole words are the unit of selection.

4 Experiments and Results

Description of Datasets: In addition to reporting ASR results on the carefully created वाक्सञ्चयः (/Vāksañcayaḥ/) dataset (described in Section 2), we also contrast through experimental analysis on two other Indian languages, *viz.*, Telugu and Gujarati. For Telugu and Gujarati, we used the publicly available speech corpora released by Microsoft (Srivastava et al., 2018) that contains 36.2/8.7 hours and 33.2/5.8 hours of training/test speech in Telugu and Gujarati, respectively. We use a new train-test split for the Gujarati and Telugu datasets because the original split had overlapping spekaers in their train and test. Our new split ensures that the train-test split have disjoint speakers. Transcript of this corpora was cleaned for orthographic errors. Corpora in these two languages were accompanied by pronunciation lexicons, which we used to build phoneme-based ASR systems to compare against our grapheme-based systems.

Experimental Setup: We use the Kaldi toolkit (Povey et al., 2011) for all our ASR experiments. Our acoustic model is implemented using Time Delay Neural Networks (TDNNs) (Peddinti et al., 2015) containing 14 layers. We use 40-dimensional MFCCs as our input features along with 100-dimensional i-vector based speaker embeddings (Saon et al., 2013). We used ngram language models with Kneser-Ney smoothing implemented using the SRILM toolkit (Stolcke, 2002). The language models were trained using both training transcripts from the speech data, as well as additional textual data derived from the Leipzig Corpora Collection for Gujarati and Telugu (Goldhahn et al., 2012) and the Digital Corpus of Sanskrit (Hellwig, 2010) for Sanskrit. The word vocabulary sizes in the lexicons for Sanskrit, Telugu and Gujarati are 76K, 43K and 48K, respectively.

Results: Tables 3, 4 and 5, present the WERs from ASR systems built using different choices of AM and LM units using both the graphemic representations (Native and SLP1) for Sanskrit, Gujarati and Telugu, respectively. From Table 3, we

Sr.	Script	LM Unit	AM Unit	WER
1	Native	Word	Gr.	40.06
2	SLP1	Word	Gr.	40.57
3	Native	VS	Gr.	22.55
4	SLP1	VS	Gr.	22.59
5	Native	BPE	Gr.	21.99
6	SLP1	BPE	Gr.	**21.94**
7	Native	Word	Gr.+VS	42.05
8	SLP1	Word	Gr.+VS	41.36
9	Native	VS	Gr.+VS	24.04
10	SLP1	VS	Gr.+VS	24.98
11	Native	BPE	Gr.+VS	**23.58**
12	SLP1	BPE	Gr.+VS	24.15

Table 3: WERs on Sanskrit test set. (Native=Devanagari, VS=Vowel-Split, Gr.=Grapheme)

Script	LM Unit	AM Unit	WER
Native	Word	Phn.	18.63
Native	Word	Gr.	19.17
Native	BPE	Gr.	24.49
SLP1	Word	Phn.	**18.26**
SLP1	Word	Gr.	18.27
SLP1	Word	Gr. + VS	19.81
SLP1	BPE	Gr.	23.97
SLP1	BPE	Gr. + VS	25.08
SLP1	VS	Gr.	26.33
SLP1	VS	Gr. + VS	26.18

Table 4: WERs on Gujarati test set. (Gr.=Grapheme , Phn.= Phoneme)

Script	LM Unit	AM Unit	WER
Native	Word	Phn.	21.12
Native	Word	Gr.	21.25
Native	BPE	Gr.	26.68
SLP1	Word	Gr.	**20.75**
SLP1	Word	Phn.	20.92
SLP1	Word	Gr. + VS	22.13
SLP1	BPE	Gr.	25.07
SLP1	BPE	Gr. + VS	26.3
SLP1	VS	Gr.	33.68
SLP1	VS	Gr. + VS	36.57

Table 5: WERs on Telugu test set. (Gr.=Grapheme , Phn.= Phoneme)

see that BPE units[4] and vowel segment units are far superior compared to words as an LM unit for Sanskrit. This is unsurprising given that Sanskrit has a high rates of OOV (44.16%). However, as shown in Tables 4 and 5 the configurations with word based LMs performs the best for Gujarati and Telugu respectively. Gujarati and Telugu have lower OOV rates of 18.63 % and 15.26 %.

Table 6 shows the distribution of words with 1-4 continuous consonants in all three languages. For Telugu, even though the number of conjunct consonants with $N = 2$ is higher than in Sanskrit, we found on inspecting the audio data that such conjunct consonants are often not enunciated clearly. For example స్వాతన్త్ర్యమ్(/svāta**ntry**am/) is pronounced as స్వాతన్త్రమ్(/svāta**ntr**am/). However, in Sanskrit, conjunct consonants having 5 consonants together such as कार्त्स्न्यम् (/kā**rtsny**am/) are enunciated very clearly and all consonants appear articulated in its pronunciation.

Due to the morphological richness (Kulkarni et al., 2015), inflections and compounds, Sanskrit always has the highest number of rare words. In the training dataset used in the Sanskrit ASR experiments with the vocab size of 70.5K, more than 87.25% words have a frequency less than 3, where as in Telugu and Gujarati training dataset, this is 76.76% and 77.26%, respectively. Clear articulation of conjunct consonants and higher rare word rates makes the BPE and VS based models performs better in Sanskrit than other two languages along with the impact of OOVs.

We observe that use of SLP1 as a graphemic representation schemes performs best for all the three languages. SLP1 is designed to capture the phonemic-graphemic correspondences present in Indic languages. We also find that ASR performance using phonemes is comparable to graphemes for Gujarati and Telugu. In Sanskrit, we observe that purely grapheme-based acous-

N	Sanskrit	Telugu	Gujarati
1	77.30%	75.77%	89.37%
2	21.41%	23.27%	10.06%
3	1.26%	0.96%	0.56%
4	0.03%	0%	0%

Table 6: Number of continuous consonants (N) distribution in three ASR datasets

---

[4] Vocabulary size of 32K is used for BPE which is closest to the vowel split (29,147 entries) for Sanskrit. Performance for varying number of subword units for BPE is presented in Appendix B.

tic models outperform grapheme+vowel segment-based acoustic models. With the consistent mapping between graphemes and phonemes and the absence of schwa deletion, it is intuitive that grapheme-based models would be most appropriate for Sanskrit. Even though for Sanskrit in some cases Devanagari as a graphemic representation outperforms the SLP1 (Sr. 1,3,9,11 in Table 3), the model that uses SLP1 script always outperforms the other in terms of character error rate.

In Sanskrit the pause given between the sub-words of a compound word and in between two words varies depending on the fluency of the speaker and the complexity of the text, which can deteriorate the WER. The utterance for 'महान् प्राकारः' /mahān prākāraḥ/ may get recognised as 'महान्प्राकारः' /mahānprākāraḥ/, where two correctly recognised words will be evaluated as one *deletion* and one *substituion* by the evaluation model. Similarly if the audio of 'शोभमानमासीत्' /śobhamānamāsīt/ gets recognised as 'शोभमानम् आसीत्' /śobhamānam āsīt/, then it will be considered as one *insertion* followed by one *substituion*. After negating these two particular errors, we will get 17.79% as the *modulo substitution deletion* WER for our best model of Sanskrit (Sr. 6 of Table 3). The character error rate 3.10% for the best model in Sanskrit also ensures the performance of the model and the quality of the dataset, where as the CER for the best model of Gujarati and Telugu are 5.49% and 5.60% respectively, much higher than Sanskrit.

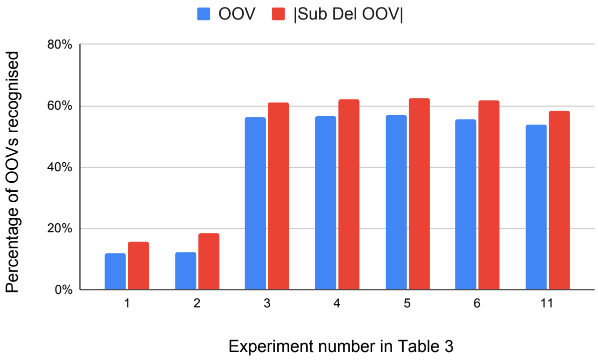

Figure 4: Percentage of OOVs recognised for Sanskrit by the ASR system for the experiments listed in Table 3

OOV Analysis: The OOVs at word level are 44.16%, 15.26% and 18.63% for Sanskrit, Telugu and Gujarati, respectively. Figure 4 depicts the percentage of OOVs recovered by different ASR experimental setup described in Table 3, also giving *modulo substitution deletion* OOVs. As we move from word based LM Unit to BPE and VS, system is able to recognize more than 57% of OOVs. For Telugu and Gujarati OOVs recognition rate is 53.67% and 48.61% respectively for their best performing models.

#	WER	Speech Description
1*	37.79%	Tamil influenced accents
2*	37.60%	Hindi influenced accents
3*	46.27%	Radio Program
4	46.62%	Extempore Discourse
5	51.52%	Live Lecture

Table 7: Test results for the Out-of-domain Dataset (* readings from books and transcripts)

Out-of-domain test set. Table 7 presents results on the out-of-domain test set described in Section 2. It shows the WERs we can expect from our models when the speakers and content largely vary in domain from our dataset. This test set was sampled for specific speakers and content that qualify as being out-of-domain. These test utterances were evaluated using our best performing Sanskrit ASR models. Speakers #1 and #2 were included, as their utterances show more pronounced influence of their native languages, Tamil and Hindi respectively. It is observed that speaker #1, does not often attempt to distinguish between the pronunciation of the phoneme pairs such as /ta/ and /da/, /ka/ and /ga/, *etc.* This is in congruence with the orthography followed in Tamil, the speaker's native language. Speaker #2's reading was influenced by schwa deletion, *i.e.*, the phenomena of deleting vowel markers accompanying consonants at certain contexts (elaborated in the supplementary material) which is dominant in Hindi. For example, गतवान् /gatavān/ is pronounced as /gatvān/ by this speaker and the ASR system correctly predicts it as /gatvān/. Here, the acoustic model clearly dominates the language model. Speaker #3's utterances are the Sanskrit translation of the Indian government's public outreach program known as 'मन की बात' /man kī bāt/. This widely differs from the training speech dataset and vocabulary of the LM that is used. Similar observation can be made for speakers #4 and #5, albeit for different reasons. Speaker #4 in the discourse tends to use rare words, especially domain-specific proper nouns and derivational verbs, both of which are scarce in our LM vocabulary. Speaker #5 tends to deviate from conventional speech patterns, by providing emphasis on specific words, for the purpose of pedagogy.

5 Conclusion

We presented a new Sanskrit speech corpus वाक्सञ्चयः (/Vāksañcayaḥ/) along with a new large-vocabulary ASR system. We explored different unit selection alternatives for both AM and LM, along with a new segmentation approach. We observe that SLP1, when used as the script instead of the native scripts generally results in better performances for Gujarati and Telugu. For Sanskrit, SLP1 based model results in better character error rate and BPE based model with SLP1 giving the best result. Similarly, we observe that vowel split based segmentation consistently yields better performance than word based model and close to results of the best model. Our current dataset and the model are specifically designed to handle data in prose. The inclusion of poetry data would require substantial changes to the system, which we plan to address in the near future. For instance, the poetry data would greatly benefit from insights from Sanskrit prosody. More importantly, the degree of free word orderness in prose and poetry greatly varies in Sanskrit, so much so that an n-gram LM will not be effective (Krishna et al., 2018).

Acknowledgments

We would like to thank Prof. K. Ramasubramanian, IIT Bombay, for supporting the creation of Sanskrit speech corpus. We express our gratitude to the volunteers who have participated in recording readings of classical Sanskrit texts and helping make this resource available for the purpose of research.

References

Devaraj Adiga, Rohit Saluja, Vaibhav Agrawal, Ganesh Ramakrishnan, Parag Chaudhuri, K Ramasubramanian, and Malhar Kulkarni. 2018. Improving the learnability of classifiers for sanskrit ocr corrections. In *The 17th World Sanskrit Conference, Vancouver, Canada. IASS*, pages 143–161.

CS Anoop and AG Ramakrishnan. 2019. Automatic speech recognition for sanskrit. In *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, volume 1, pages 1146–1151. IEEE.

G. N. Clements. 1990. The role of the sonority cycle in core syllabification. *Papers in Laboratory Phonology*, pages 283–333.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for sanskrit processing. In *Proceedings of COLING 2012*, pages 1011–1028.

Oliver Hellwig. 2010. DCS-the digital corpus of Sanskrit. *University of Heidelberg: Department of Classical Indology*.

S Prahallad Kishore and Alan W Black. 2003. Unit size in unit selection speech synthesis. In *Eighth European Conference on Speech Communication and Technology*.

SP Kishore, Rohit Kumar, and Rajeev Sangal. 2002. A data driven synthesis approach for indian languages using syllable as basic unit. In *Proceedings of Intl. Conf. on NLP (ICON)*, pages 311–316.

Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, and Pawan Goyal. 2018. Free as in free word order: An energy based model for word segmentation and morphological tagging in Sanskrit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2550–2561, Brussels, Belgium. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2021. A graph-based framework for structured prediction tasks in sanskrit. *Computational Linguistics*, 46(4):785–845.

Amrith Krishna, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, and Pawan Goyal. 2019. Poetry to prose conversion in Sanskrit as a linearisation task: A case for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1160–1166, Florence, Italy. Association for Computational Linguistics.

Amrith Krishna, Shiv Vidhyut, Dilpreet Chawla, Sruti Sambhavi, and Pawan Goyal. 2020. SHR++: An interface for morpho-syntactic annotation of Sanskrit corpora. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7069–7076, Marseille, France. European Language Resources Association.

Amba Kulkarni. 2013. A deterministic dependency parser with dynamic programming for sanskrit. In *Proceedings of the Second International Conference on Dependency Linguistics, DepLing 2013, August 27-30, 2013, Prague, Czech Republic*, pages 157–166. MATFYZPRESS Publishing House of the Faculty of Mathematics and Physics Charles University in Prague / ACL.

Amba Kulkarni, Sheetal Pokar, and Devanand Shukl. 2010a. Designing a constraint based parser for sanskrit. In *Sanskrit Computational Linguistics - 4th International Symposium, New Delhi, India, December 10-12, 2010. Proceedings*, volume 6465 of *Lecture Notes in Computer Science*, pages 70–90. Springer.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How free is free word order in sanskrit. *The Sanskrit Library, USA*, pages 269–304.

Malhar Kulkarni, Chaitali Dangarikar, Irawati Kulkarni, Abhishek Nanda, and Pushpak Bhattacharyya. 2010b. Introducing sanskrit wordnet. In *Proceedings on the 5th global wordnet conference (GWC 2010), Narosa, Mumbai*, pages 287–294.

Malhar Kulkarni, Irawati Kulkarni, Chaitali Dangarikar, and Pushpak Bhattacharyya. 2010c. Gloss in sanskrit wordnet. In *Sanskrit Computational Linguistics - 4th International Symposium, New Delhi, India, December 10-12, 2010. Proceedings*, volume 6465 of *Lecture Notes in Computer Science*, pages 190–197. Springer.

Anil Kumar, Vipul Mittal, and Amba Kulkarni. 2010. Sanskrit compound processor. In *Sanskrit Computational Linguistics - 4th International Symposium, New Delhi, India, December 10-12, 2010. Proceedings*, volume 6465 of *Lecture Notes in Computer Science*, pages 57–69. Springer.

Chia-ying Lee, Yu Zhang, and James Glass. 2013. Joint learning of phonetic units and word pronunciations for ASR. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 182–192.

Ghosh Manomohan and Pāṇini. 1938. *Pāṇinīya Śikṣā or The Śikṣā Vedāṅga, Ascribed to Pāṇini*. University of Calcutta.

P.H. Matthews. 2007. *The Concise Oxford Dictionary of Linguistics*. Oxford Paperback Reference. OUP Oxford.

Keshav S Melnad, Pawan Goyal, and Peter Scharf. 2013. Meter identification of sanskrit verse. *The Sanskrit Library, USA*.

Diwakar Mishra, Kalika Bali, and Girish Nath Jha. 2013. Syllabification and stress assignment in phonetic sanskrit text. In *2013 International Conference Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE), Gurgaon, India, November 25-27, 2013*, pages 1–4. IEEE.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukás Burget, Ondrej Glembek, Nagendra Kumar Goel, Mirko Hannemann, Petr Motlícek, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society.

Richard G Salomon. 1996. Brahmi and kharoshthi. *The world's writing systems*, pages 373–383.

Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. 2017. Error detection and corrections in Indic OCR using LSTMs. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 17–22. IEEE.

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. 2013. Speaker adaptation of neural network acoustic models using i-vectors. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59.

Peter M. Scharf and Malcolm D. Hyman. 2011. *Linguistic Issues in Encoding Sanskrit*. The Sanskrit Library.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Preeti Shukla, Devanand Shukl, and Amba Kulkarni. 2010. Vibhakti divergence between sanskrit and hindi. In *Sanskrit Computational Linguistics - 4th International Symposium, New Delhi, India, December 10-12, 2010. Proceedings*, volume 6465 of *Lecture Notes in Computer Science*, pages 198–208. Springer.

Richard Sproat. 2003. A formal computational analysis of Indic scripts. In *International symposium on Indic scripts: past and future, Tokyo*.

Brij Mohan Lal Srivastava, Sunayana Sitaram, Rupesh Kumar Mehta, Krishna Doss Mohan, Pallavi Matani, Sandeepkumar Satpal, Kalika Bali, Radhakrishnan Srikanth, and Niranjan Nayak. 2018. Interspeech 2018 low resource automatic speech recognition challenge for Indian languages. In *SLTU*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *INTERSPEECH*.

A Differences between Sanskrit and other Indic languages for ASR

Many Indian languages are known to be derived from Sanskrit (Kulkarni et al., 2010b) and their scripts derived from the Brahmi script (Salomon, 1996; Sproat, 2003), which leads to grapheme-based similarites amongst them. In Figure 5, we illustrate through an example, the spectrum of mapping the native character/grapheme (units) in words across languages; at one end of the spectrum is राम(/rām/) in Hindi mapped to రామ(/rāma/) in Telugu as an example where direct correspondence with the native character exists. Going further in the spectrum are examples for which direct character correspondence does not exist. सीता(/sītā/) in Hindi going to ಸೀತೆ(/sīte/) in Kannada is an instance where there is a change in the ending vowel.

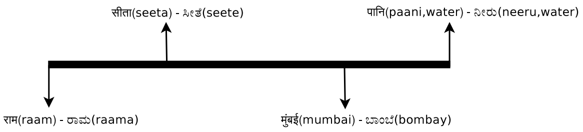

Figure 5: Spectrum of mapping native character/grapheme (units) in words across Indian languages

Schwa Deletion The schwa deletion phenomenon plays a crucial role in the north Indian languages. Every consonant by itself includes a short /a/ vowel sound (referred to as “schwa”) unless otherwise specified. For example, the letter ‘त’ in Hindi is pronounced as /ta/. This sound can be associated with any other vowel sound by the use of “Mātras”. Mātras are dependent forms of vowels. Schwa is the default vowel for a consonant and hence does not require any explicit Mātra to represent it. Schwa deletion is a phenomenon where implicit schwas of a word are deleted during pronunciation. For example, in Hindi, the proper noun, ‘अर्जुन (/arjun/, the name of a person) has schwa deletion after the consonant ‘न’ and is pronounced as Arjun. This phenomenon is not observed in the South Indian languages. For instance, in Kannada it is pronounced as ‘Arjuna’. There is no implicit schwa deletion in Sanskrit as well as in the traditional use of South Indian languages such as Kannada. North Indian languages observe schwa deletion not only at the end of the word, but also in the middle of a word in some cases. For example, the word ‘गलती’ (/galtī/ meaning mistake) in Hindi observes implicit schwa deletion after the consonant ‘ल’ (/la/).

ASR becomes challenging because of this phenomenon since the occurrence of schwa deletion is not always explicitly specified in the orthography. For example, the name रामबाबु (/rāmbābu/) has two basic words concatenated to form a name. In Hindi, this name has an implicit schwa deleted at म (consonant sounding ’ma’) of राम (/rām/). While constructing phonetic representations for ASR, such deletions introduce ambiguities in pronunciation which could be alleviated by enforcing more consistency between graphemes and phonemes. This same word रामबाबु written in Telugu would be phonetically represented as రామ్బాబు (/rāmbābu/) instead of రామబాబు (/rāmabābu/) which is intuitive. Note that in the former case, there is an addition of ‘◌్(*halant*: an explicit schwa deletion marker) at మ(/ma/). This forces the consonants మ(/ma/) and బ(/ba/) to combine and form a conjunct. In the latter case there is a grapheme consistency across both Hindi and Telugu languages but there is a variation in their pronunciation due to the schwa deletion phenomenon. In contrast, in the case of Sanskrit, since pronunciation is strictly governed by the शिक्षा(/śikṣā/) (Manomohan and Pāṇini, 1938), a treatise on phonetics, schwa deletion is not observed.

B BPE Experiment Details

In Sanskrit a noun can have 24 to 92 inflections (depending on base word’s gender and alternate forms) and a verb can have 90 to 180 inflections. Derivative nouns (*Taddhitas*) and verbs (passive (*Karmaṇi*), *san, ṇic, yaṅ,* etc) are also used often in Sanskrit literature. Due to this morphological richness and frequently occurring compound words, vocab size can be reduced by properly selecting repetitive stems and suffices using BPE by specifying the number of merge operations. Therefore we experimented varying number of subword unit with vocabulary sizes of 2K, 4K, 8K, 16K, 32K and 64K (K=1000). Table 8 shows the varying BPE configuration on our best configuration, i.e graphemes as AM unit and BPE as LM unit. However, the performance of these configurations are comparable irrespective of their BPE vocabulary size.

BPE with vocabulary size of 32,000 stands closes to that of VS, with a vocabulary size of 29,147. Even in this configuration, BPE outperforms VS, as BPE reports a WER of 21.94 as

Sr.	Vocabulary Sizes	WER
1	32K	21.94
2	16K	21.96
3	8K	21.83
4	4K	**21.79**
5	2K	22.41

Table 8: WERs for Sanskrit on different vocabulary sizes with Graphemes as AM unit and BPE as LM unit using script SLP1 (on the best configuration model).

against 22.58 (Table 3 serial no. 4) of VS. However, the BPE configuration with a vocabulary size of 4K reports the lowest WER, which is 0.15% points lower in comparison to BPE with 32K vocabulary size. But the CER for 32K vocabulary size is 3.10% which is outperforming the BPE with vocabulary size of 4K which has a CER of 3.29%.

C List of works used in the speech corpus

- Mallinātha's commentary on KumāraSambhavam
- Mallinātha's commentary on Raghuvamśam
- Ādiśaṅkara's Bhaṣyam on Kaṭhopaniṣat
- Ādiśaṅkara's Bhaṣyam on Bhagavadgītā (Chapters 1-9)
- Ādiśaṅkara's Bhaṣyam on Brahmasūtram
- Yogasūtram Vyāsabhāṣya-sahitam
- Ṛṇvimuktiḥ by SaṃskṛtaBhāratī
- Āñjaneya-Rāmāyaṇam by SaṃskṛtaBhāratī
- Kathālaharī by SaṃskṛtaBhāratī
- Bālamodinī stories from SambhāṣaṇaSandeśa by SaṃskṛtaBhāratī
- Samarthaḥ Svāmī Rāmadāsaḥ by SaṃskṛtaBhāratī
- Yugāvatāraḥ by SaṃskṛtaBhāratī
- Prāstāvikam of Swāmī Aḍgaḍānanda's commentary on Bhagavadgītā
- ViśuddhaVedāntaSāraḥ by SaccidānendraSarasvatī
- Vyākaraṇa-Mahābhāṣyam of Patañjali
- Man-Kī-Bāt Sanskrit translation
- Lecture on Lilāvatī
- Extempore Discourse

C.1 Sources of Recorded Audios

- `http://vedabhoomi.org`
- `https://archive.org/details/Anjaneya-rAmAyaNam`
- `https://archive.org/details/geethasb`
- `https://archive.org/details/bAlamodinI-01`
- `https://archive.org/details/kathA-laharI`
- `https://archive.org/details/Gita_Shankara_Bhashya-Sanskrit`
- `https://www.youtube.com/watch?v=LJGjfHHHBoQ`
- `https://sanskritdocuments.org/sites/manogatam/`
- `https://archive.org/details/YatharthGeetaSanskritAudio`
- `https://surasa.net/music/samskrta-vani/#stories_stories_songs`

C.2 Sources of Tools used for Recording, Cleaning and Transcribing the Audios

- ASR Voice Recorder `https://play.google.com/store/apps/details?id=com.nll.asr`
- Audacity `https://www.audacityteam.org/`
- oTranscribe `https://otranscribe.com/`

D Computing Infrastructure

- GPU Model Name : GeForce GTX 1080 Ti
- GPU RAM : 12 GB
- CPU Model Name : CPU Intel(R) Xeon(R) Gold 5120 CPU
- Processor Speed : 2.20GHz
- System Memory : 256 GB
- CPU Cores : 56

Constraint based Knowledge Base Distillation in End-to-End Task Oriented Dialogs

Dinesh Raghu[*][1][2], **Atishya Jain**[1], **Mausam**[1] **and Sachindra Joshi**[2]
[1] IIT Delhi, New Delhi, India
[2] IBM Research, New Delhi, India
[diraghu1, jsachind]@in.ibm.com, [Atishya.Jain.cs516, mausam]@cse.iitd.ac.in

Abstract

End-to-End task-oriented dialogue systems generate responses based on dialog history and an accompanying knowledge base (KB). Inferring those KB entities that are most relevant for an utterance is crucial for response generation. Existing state of the art scales to large KBs by softly filtering over irrelevant KB information. In this paper, we propose a novel filtering technique that consists of (1) a pairwise similarity based filter that identifies relevant information by respecting the n-ary structure in a KB record. and, (2) an auxiliary loss that helps in separating contextually unrelated KB information. We also propose a new metric – multiset entity F1 which fixes a correctness issue in the existing entity F1 metric. Experimental results on three publicly available task-oriented dialog datasets show that our proposed approach outperforms existing state-of-the-art models.

1 Introduction

Task oriented dialog systems interact with users to achieve specific goals such as restaurant reservation or calendar enquiry. To satisfy a user goal, the system is expected to retrieve necessary information from a knowledge base and convey it using natural language. Recently several end-to-end approaches (Bordes and Weston, 2017; Wu et al., 2018; He et al., 2020b; Madotto et al., 2018) have been proposed for learning these dialog systems.

Inferring the most relevant KB entities necessary for generating the response is crucial for achieving task success. To effectively scale to large KBs, existing approaches (Wen et al., 2018; Wu et al., 2018) distill the KB by softly filtering irrelevant KB information based on the dialog history. For example, in Figure 1 the ideal filtering technique is expected to filter just the row 1 as the driver is requesting information about *dinner* with *Alex*. But existing techniques often filter some irrelevant KB information along with the relevant KB information. For example, in Figure 1 row 3 may also get filtered along with row 1.

#	Date	Event	Invitee	Time
1	*1st Feb*	*dinner*	*Alex*	*10am*
2	7th Feb	dentist appointment	-	2pm
3	8th Feb	dinner	Ana	7pm

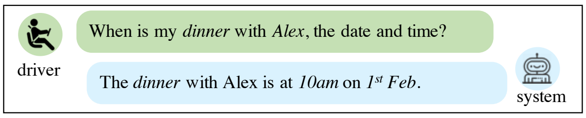

When is my *dinner* with *Alex*, the date and time?

The *dinner* with Alex is at *10am* on *1st Feb*.

Figure 1: An example dialog between a driver and a system along with the associated knowledge base.

Our analysis of the best performing distillation technique (Wu et al., 2018) revealed that embeddings learnt for entities of the same type are quite close to each other. This may be due to entities of the same type often appearing in similar context in history and KB. Such embeddings hurt the overall performance as they reduce the gap between relevant and irrelevant KB records. For example, in Figure 1 row 3 may not get distilled out if *Alex* and *Ana* have similar embeddings.

In this paper, we propose Constraint based knowledge base Distillation NETwork (CDNET), which (1) uses a novel pairwise similarity based distillation computation which distills KB at a record-level, and (2) an auxiliary loss which helps to distill contextually unrelated KB records by enforcing constraints on embeddings of entities of the same type. We noticed the popular entity F1 evaluation metric has a correctness issue when the response contains multi instances of the same entity value. To fix this issue, we propose a new metric called multiset entity F1. We empirically show that CDNET performs either significantly better than or comparable to existing approaches on three publicly available task oriented dialog datasets.

---

[*] D. Raghu is an employee at IBM Research. This work was carried out as part of PhD research at IIT Delhi.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5051–5061
August 1–6, 2021.

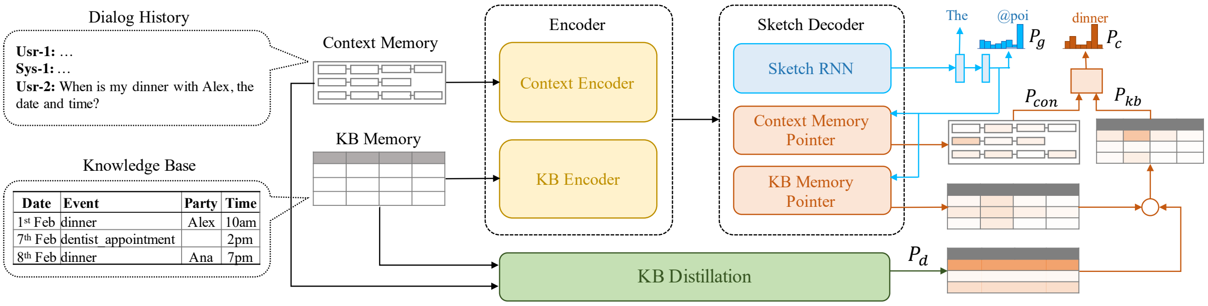

Usr-1: …
Sys-1: …
Usr-2: When is my dinner with Alex, the date and time?

Date	Event	Party	Time
1st Feb	dinner	Alex	10am
7th Feb	dentist_appointment		2pm
8th Feb	dinner	Ana	7pm

Figure 2: Architecture of CDNET model.

2 Related Work

We first discuss approaches that are closely related to our work. Wu et al. (2018) perform KB distillation but fails to capture the relationship across attributes in KB records. It represents a KB record with multiple attributes as a set of triples (*subject, predicate, object*). This breaks direct connection between record attributes and requires the system to reason over longer inference chains. In Figure 1, if *event* field is used as the key to break the record into triples, then the distillation has to infer that *(dinner, invitee, Alex)*, *(dinner, date, 1st Feb)* and *(dinner, time, 10am)* are connected. In contrast, CDNET performs KB distillation by maintaining the attribute relationships. Wen et al. (2018) perform distillation using the similarity between dialog history representation and each attribute representation in a KB record, whereas CDNET uses word based pairwise similarity for distillation.

We now briefly discuss approaches that improve other aspects of task oriented dialogs. He et al. (2020c) and He et al. (2020b) model KBs using Relational GCNs (Schlichtkrull et al., 2018).Raghu et al. (2019) provide support for entities unseen during train. Reddy et al. (2019) improve the ability to reason over KB by respecting the relationships between connected attributes. Qin et al. (2019) restricts the response to contain entities from a single KB record. (Qin et al., 2020) handle multiple domains using shared-private networks and He et al. (2020a) optimize their network on both F1 and BLEU. We are the first to propose a pairwise similarity score for KB distillation and a embedding constraint loss to distill irrelevant KB records.

3 CDNET

CDNET [1] has an encoder-decoder architecture that takes as input (1) the dialog history $\mathcal{H}$, modelled as a sequence of utterances $\{u_i\}_{i=1}^{k}$ and each utterance u_i as sequence of words $\{w_i^j\}$, and (2) a knowledge base $\mathcal{K}$ with M records $\{r_m\}_{m=1}^{M}$ and each record r_m has N key-value attribute pairs $\{(k^n, v_m^n)\}_{n=1}^{N}$. The network generates the system response $Y = (y_1, \ldots, y_{\mathbb{T}})$ one word at a time.

3.1 CDNET Encoder

Context Encoder: The dialog history $\mathcal{H}$ is encoded using a hierarchical encoder (Sordoni et al., 2015). Each utterance representation $\mathbf{u}_i$ is computed using a Bi-GRU (Schuster and Paliwal, 1997). We denote the hidden state of the j^{th} word in i^{th} utterance as $\mathbf{w}_i^j$. The context representation $\mathbf{c}$ is generated by passing $\mathbf{u}_i$s through a GRU.

KB Encoder: We encode the KB using the multi-level memory proposed by Reddy et al. (2019) as its structure allows us to perform distillations over KB records. The KB memory contains two-levels. The first level is a set of KB records. Each KB record is represented as sum of its attributes $\mathbf{r}_m = \sum_n \Phi^e(v_m^n)$, where Φ^e is the embedding matrix. In the second level, each record is represented as a set of attributes. Each attribute is a key-value pair, where the key $\mathbf{k}^n$ is the attribute type embedding and the value $\mathbf{v}_m^n$ is the attribute embedding.

3.2 KB Distillation

The KB distillation module softly filters irrelevant KB records based on the dialog history by computing a distillation distribution (P_d) over the KB records. To compute $P_d = [d_1, \ldots, d_M]$, we first score each KB record r_m based on the dialog history $\mathcal{H}$ as follows:

$$s_m = \sum_{w \in \mathcal{H}} \sum_{v_m^n \in r_m} \text{CosSim}(\Phi^e(w), \Phi^e(v_m^n)) \quad (1)$$

where CosSim is the cosine similarity between two vectors. The distillation likelihood d_m

---

[1] `https://github.com/dair-iitd/CDNet`

for each record r_m then is given by $d_m = \exp(s_m)/\sum_{q=1}^{M}\exp(s_q)$.

Defining distillation distribution over the KB records rather than KB triples has two main advantages: (1) attributes (such as invitee, event, time and date in Figure 1) in a KB record are directly connected and thus easy to distill, (2) it helps to distill the right records even when the record keys are not unique. In Figure 1, row 3 would be distilled even though it shares the same event name.

3.3 CDNET Decoder

Following Wu et al. (2018), we first generate a sketch response which uses entity type (or sketch) tag in place of an entity. For example, *The @meeting with @invitee is at @time* is generated instead of *The dinner with Alex is at 10pm*. When an entity tag is generated, we choose an entity suggested by the context and KB memory pointers.

Sketch RNN: We use a GRU to generate the sketch response. At each time t, a generate distribution P_g is computed using the decoder hidden state $\mathbf{h}_t$ and an attended summary of the dialog context $\mathbf{g}_t$. The summary $\mathbf{g}_t = \sum_i \sum_j a_{ij}\mathbf{w}_i^j$, where a_{ij} is the Luong attention (Luong et al., 2015) weights over the context word representations ($\mathbf{w}_i^j$).

Context Memory Pointer: At each time t, generate the copy distribution over the context P_{con} by performing a multi-hop Luong attention over the context memory. The initial query $\mathbf{q}_t^0$ is set to $\mathbf{h}_t$. $\mathbf{q}_t^0$ is then attended over the context to generate an attention distribution $\mathbf{a}^1$ and a summarized context $\mathbf{g}_t^1$. We represent this as $\mathbf{g}_t^1 = \text{Hop}(\mathbf{q}_t^0, \mathbf{x})$. In the next hop the same process is repeated by updating the query $\mathbf{q}_t^1 = \mathbf{q}_t^0 + \mathbf{g}_t^1$. The attention weights after H hops is used for computing the context pointer P_{con} as follows:

$$P_{con}(y_t = w) = \sum\nolimits_{ij:w_i^j = w} a_{ij}^H \qquad (2)$$

KB Memory Pointer: At each time t, we generate the copy distribution over the KB P_{kb} using (1) Luong attention weight β_m^t over the KB record r_m and (2) Luong attention weight γ_n^t over attribute keys in a record $\mathbf{k}^n$ and (3) the distillation weight d_m over the KB record r_m. The KB pointer P_{kb} is computed as follows:

$$P_{kb}(y_t = w) = \frac{\sum_{mn:v_m^n = w} d_m \beta_m^t \gamma_n^t}{\sum_{mn} d_m \beta_m^t \gamma_n^t} \qquad (3)$$

The two copy pointers are combined using a soft gate α (See et al., 2017) to get the final copy distribution P_c as follows,

$$P_c(y_t) = \alpha P_{kb}(y_t) + (1-\alpha)P_{con}(y_t) \qquad (4)$$

3.4 Loss

We guide the distillation module using two auxiliary loss terms: entity constraint loss $\mathcal{L}_{ec}$ and distillation loss $\mathcal{L}_d$. Often entities of the same type (e.g., *Ana* and *Alex*) have embeddings similar to each other. As a result, records with similar but unrelated entities are incorrectly assigned a high distillation likelihood. To alleviate this problem, we make the cosine similarity between two entities of the same type to be as low as possible. This is captured by the constraint loss $\mathcal{L}_{ec}$ given by,

$$\mathcal{L}_{ec} = \sum_{(e_a, e_b)\in\mathcal{E}} \text{CosSim}(\Phi^e(e_a), \Phi^e(e_b)) \qquad (5)$$

where $\mathcal{E}$ is a set of entity pairs in the KB that belong to the same entity type.

The distillation likelihood d_m of a KB record r_m depends on the similarity between entities in the record and the words mentioned in the dialog context. We compute the distillation loss by defining reference distillation distribution d_m^* as $s_m^*/\sum_{q=1}^{M} s_q^*$, where s_m^* is the number of times any attribute in r_m occurs in $\mathcal{H}$ and in the gold response. The distillation loss is given by,

$$\mathcal{L}_d = -\sum\nolimits_{m=1}^{M} d_m^* \log(d_m) \qquad (6)$$

The overall loss function $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_c + \mathcal{L}_{ec} + \mathcal{L}_d$, where $\mathcal{L}_g$ and $\mathcal{L}_c$ are the cross entropy loss on P_g and P_c respectively. Detailed equations are described in Appendix B.

4 Experimental Setup

Datasets: We evaluate our model on three datasets – CamRest (Wen et al., 2017), Multi-WOZ 2.1 (WOZ) (Budzianowski et al., 2018) and Stanford Multi-Domain (SMD) Dataset (Eric et al., 2017).

Baselines: We compare CDNET against the following baselines: MLM (Reddy et al., 2019), DSR (Wen et al., 2018), GLMP (Wu et al., 2018), Entity Consistent (Qin et al., 2019), EER (He et al., 2020c), FG2Seq (He et al., 2020b), TTOS (He et al., 2020a) and DFNet (Qin et al., 2020).

	Utterance	**Set**	**MultiSet**
Gold	for which one? I have two, one on the *8th* at *11am* and one on *wednesday* at *11am*	{*8th, 11am, wednesday*}	{*8th, 11am, wednesday, 11am*}
Pred-1	your appointment is on *8th* at *11am*	{*8th, 11am*}	{*8th, 11am*}
Pred-2	your appointment is on *8th* on *8th* on *8th* and on *8th*	{*8th*}	{*8th 8th, 8th, 8th*}

Table 1: An example to demonstrate the correctness issue with the Entity F1 metric.

	CamRest			**SMD**			**WOZ 2.1**		
Model	**BLEU**	**Ent. F1**	**MSE F1**	**BLEU**	**Ent. F1**	**MSE F1**	**BLEU**	**Ent. F1**	**MSE F1**
DSR (Wen et al., 2018)	18.3	53.6	-	12.7	51.9	-	9.1	30.0	-
GLMP (Wu et al., 2018)	15.1	58.9	57.5	13.9	59.6	59.6	6.9	32.4	-
MLM (Reddy et al., 2019)	15.5	62.1	-	17	54.6	-	-	-	-
Ent. Const. (Qin et al., 2019)	18.5	58.6	-	13.9	53.7	-	-	-	-
TTOS (He et al., 2020a)	20.5	61.5	-	17.4	55.4	-	-	-	-
DFNet (Qin et al., 2020)	-	-	-	14.4	62.7	56.7	9.4	35.1	34.8
EER (He et al., 2020c)	19.2	65.7	65.5	17.2	59.0	55.1	13.6	35.6	35.0
FG2Seq (He et al., 2020b)	20.2	66.4	65.4	16.8	61.1	59.1	**14.6**	36.5	36.0
CDNet	**21.8**	**68.6**	**68.4**	**17.8**	**62.9**	**62.9**	11.9	**38.7**	**38.6**

Table 2: Performance of CDNet and baselines on the CamRest, SMD and Multi-WOZ 2.1 datasets.

Training Details: CDNet is trained end to end using Adam optimizer (Kingma and Ba, 2014). The embedding dimensions of the hidden states of encoder and decoder GRU are set to 200 and 100 respectively. Word embeddings are initialized with pre-trained 200d GloVe embeddings (Pennington et al., 2014). Words not in Glove are initialized using Glorot uniform distribution (Glorot and Bengio, 2010). The dropout rate is set to 0.2 and teacher forcing ratio set to 0.9. The best hyper-parameter setting for each dataset and other training details are reported in the Appendix A.

Evaluation Metrics: We measure the performance of all the models using BLEU (Papineni et al., 2002), our proposed multiset entity F1 and for completeness the previously used entity F1 (Wu et al., 2018).

MultiSet Entity F1 (MSE F1): The entity F1 is used to measure the model's ability to predict relevant entities from the KB. It is computed by micro averaging over the set of entities in the gold responses and the set of entities in the predicted responses. This metric suffers from two main problems. First, when the gold response has multiple instances of the same entity value, it is accounted for just once in the set representation. For example, in Table 1 the entity value *11am* occurs twice in the gold response but accounted for just once in the *set* representation. As a result the recall computation does not penalize the prediction *pred-1* for missing an instance of *11am*. Second, the existing metric fails to penalize models that stutter. For example, in Table 1 the precision of *pred-2* is not penalized for repeating the entity value *8th*. We propose a simple modification to the entity F1 metric to fix these correctness issues. The modified metric, named MultiSet Entity F1, is computed by micro averaging over the *multiset* of entities rather than a *set*. As multisets allow multiple instances of same entity values, it (1) accounts for the same entity value mentioned more than once in the gold by penalizing recall for missing any instances and (2) accounts for models that stutters by penalizing the precision.

5 Results

The results are shown in Table 2. On CamRest and SMD, CDNet outperforms existing models in both MSE F1 and BLEU. On WOZ, CDNet achieves best only in MSE F1. We observed that the responses generated by CDNet on WOZ were appropriate, but did not have good lexical overlap with the gold responses. To investigate this further, we perform a human evaluation of the responses predicted by CDNet, FG2Seq and EER.

Human Evaluation: We conduct a human evaluation to assess two dimensions of generated responses: (1) *Appropriateness*: how useful are the responses for the given dialog context and KB, and (2) *Naturalness:* how human-like are the predicted responses. We randomly sampled 75 dialogs from each of the three datasets and requested two judges

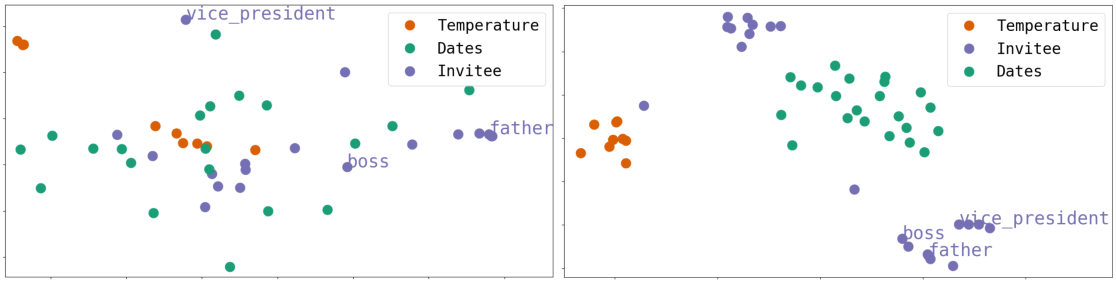

(a) (b)

Figure 3: T-sne plots of entity embeddings from SMD of (a) CDNET & (b) GLMP.

	Appropriateness			**Naturalness**		
Model	**SMD**	**Cam**	**WoZ**	**SMD**	**Cam**	**WoZ**
EER	2.9	3.8	3.4	3.6	4.2	4.0
FG2Seq	3.1	3.7	3.7	3.9	4.3	4.0
CDNET	3.6	4.1	3.9	3.7	4.3	4.1

Table 3: Human Evaluation of CDNET on the CamRest, SMD and Multi-WOZ 2.1 datasets.

	CamRest		**SMD**	
Model	**BLEU**	**MSE F1**	**BLEU**	**MSE F1**
CDNET	**21.8**	**68.4**	**17.8**	**62.9**
No $\mathcal{L}_{ec}$	19.2	65.4	17.4	62.2
Naive Dist.	15.0	64.2	16.9	60.6
Entry-Level Attn.	16.2	62.0	17.1	59.4

Table 4: Ablation study of CDNET on the CamRest and SMD datasets.

to evaluate on a Likert scale (Likert, 1932). The results are summarized in Table 3. CDNET outperforms both FG2Seq and EER on *appropriateness* across all three datasets. Despite having a lower BLEU score on WOZ, CDNET performs in-par with the other two baselines on *naturalness*.

Ablation Study: We perform an ablation study by defining three variants. Table 4 shows the MSE F1 and BLEU for the two settings on CamRest and SMD datasets. (1) We remove the entity constraint loss $\mathcal{L}_{ec}$ from the overall loss $\mathcal{L}$. (2) We replace our pairwise similarity based score s_m used for KB distillation with the global pointer score ($s_m = \mathbf{r}_m^T.\mathbf{c}$) proposed by (Wu et al., 2018). We refer to this setting as naive distillation. (3) We replace our pairwise similarity based score s_m with the entry-level attention proposed by (Wen et al., 2018). We see that both our contributions: pairwise similarity scorer for computing distillation distribution and the entity constraint loss contribute to the overall performance.

Discussion: We now discuss the effect of the entity constraint loss $\mathcal{L}_{ec}$ on the KB entity embeddings. Figure 3 shows the t-SNE plot (Van der Maaten and Hinton, 2008) of entity embeddings of CDNET and GLMP where entities of the same type are represented using the same colour. We see that entities of the same type (e.g. *father* and *boss* of the type *invitees*) are clustered together in embedding space of GLMP, while they are distributed across the space in CDNET. This shows that the entity constraint loss has helped reduce the embedding similarity between entities of the same type and ensures KB records with similar but unrelated entities are filtered by the KB distillation. Visualization of distillation distribution helping identify relevant KB entities is shown in Appendix C.

6 Conclusion

We propose CDNET for learning end-to-end task oriented dialog system. CDNET performs KB distillation at the level of KB records, thereby respecting the relationships between the connected attributes. CDNET uses a pairwise similarity based score function to better distill the relevant KB records. By defining constraints over embeddings of entities of the same type, CDNET filters out contextually unrelated KB records. We propose a simple modification to the entity F1 metric that helps fix correctness issues. We refer to the new metric as multiset entity F1. CDNET significantly outperforms existing approaches on multiset entity F1 and *appropriateness*, while being comparable on *naturalness* and BLEU. We release the code for further research.

Acknowledgments

This work is supported by IBM AI Horizons Network grant, an IBM SUR award, grants by Google, Bloomberg and 1MG, a Visvesvaraya faculty award by Govt. of India, and the Jai Gupta chair fellowship by IIT Delhi. We thank the IIT Delhi HPC facility for computational resources.

References

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Dialog System Technology Challenges, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Wanwei He, Min Yang, Rui Yan, Chengming Li, Ying Shen, and Ruifeng Xu. 2020a. Amalgamating knowledge from two teachers for task-oriented dialogue system with adversarial training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3498–3507, Online. Association for Computational Linguistics.

Zhenhao He, Yuhong He, Qingyao Wu, and Jian Chen. 2020b. Fg2seq: Effectively encoding knowledge for end-to-end task-oriented dialog. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8029–8033. IEEE.

Zhenhao He, Jiachun Wang, and Jian Chen. 2020c. Task-oriented dialog generation with enhanced entity representation. *Proc. Interspeech 2020*, pages 3905–3909.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

A. Madotto, CS. Wu, and P. Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Libo Qin, Yijia Liu, Wanxiang Che, Haoyang Wen, Yangming Li, and Ting Liu. 2019. Entity-consistent end-to-end task-oriented dialogue system with kb retriever. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 133–142.

Libo Qin, Xiao Xu, Wanxiang Che, Yue Zhang, and Ting Liu. 2020. Dynamic fusion network for multi-domain end-to-end task-oriented dialog. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6344–6354, Online. Association for Computational Linguistics.

Dinesh Raghu, Nikhil Gupta, and Mausam. 2019. Disentangling Language and Knowledge in Task-Oriented Dialogs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1239–1255, Minneapolis, Minnesota. Association for Computational Linguistics.

Revanth Gangi Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2019. Multi-level memory for task oriented dialogs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3744–3754.

Michael Sejr Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1073–1083.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562.

Haoyang Wen, Yijia Liu, Wanxiang Che, Libo Qin, and Ting Liu. 2018. Sequence-to-sequence learning for task-oriented dialogue with dialogue state representation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3781–3792, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

TH Wen, D Vandyke, N Mrkšíc, M Gašíc, LM Rojas-Barahona, PH Su, S Ultes, and S Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference*, volume 1, pages 438–449.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2018. Global-to-local memory pointer networks for task-oriented dialogue. In *International Conference on Learning Representations*.

A Training Details:

All the hyper parameters are finalised after a grid search over the dev set. We sample learning rates (LR) from $\{2.5 \times 10^{-4}, 5 \times 10^{-4}, 10^{-4}\}$. The Disentangle Label Dropout (DLD) rate (Raghu et al., 2019) is sampled from $\{0.0, 0.05, 0.10, 0.15, 0.20\}$. The number of hops H in the response decoder is sampled from $\{1, 3, 5\}$. We ran each hyperparameter setting 10 times and use the setting with the best validation entity F1. The best performing hyperparameters for all datasets are listed in Table 5.

Dataset	Hops	DLD	LR	Val MSE F1
CamRest	1	0%	0.0005	68.6
SMD	3	5%	0.00025	60.4
WoZ 2.1	3	0%	0.00025	34.3

Table 5: Best performing hyperparameters along with the best validation Entity F1 (Val Ent. F1) achieved for the three datasets.

All experiments were run on a single Nvidia V100 GPU with 32GB of memory. CDNET has an average runtime of 3 hours (6 min per epoch), 10 hours (20 min per epoch) and 24 hours (36 min per epoch) on CamRest, SMD and WOZ respectively. CDNET has a total of 2.8M trainable parameters (400K for embedding matrix, 720K for context encoder, 240k for the sketch RNN and 1440k for the Memory pointers).

B Detailed Equations

In this section, we describe the details of context encoder, CDNETdecoder and the loss.

B.1 Context Encoder

Given a dialog history $\mathcal{H}$ we compute the utterance representation $\mathbf{u}_i$ and context representation $\mathbf{c}$ as follows:

$$\mathbf{u_i} = \text{BiGRU}(\Phi^e(w_i^1), \ldots, \Phi^e(w_i^{\tau_i})) \quad (7)$$

$$\mathbf{c} = \text{GRU}(\mathbf{u}_1, \ldots, \mathbf{u}_k) \quad (8)$$

where τ_i is the number of words in utterance u_i and w_i^j is the j^{th} word in the i^{th} utterance.

B.2 CDNET Decoder

Let $\mathbf{h_t}$ and y_t be the hidden state and the predicted word at time t respectively. The hidden state is computed as follows,

$$\mathbf{h}_t = \text{GRU}(\Phi^e(y_{t-1}), \mathbf{h}_{t-1}) \quad (9)$$

Now, we compute multi-hop Luong attention over the words representations $\mathbf{w}_i^j$ in the context memory. We set the initial query $\mathbf{q_t^0}$ to $\mathbf{h_t}$ and then apply Luong attention as follows:

$$a_{ij}^1 = \frac{\exp(W_1 \tanh(W_2[\mathbf{w}_i^j, \mathbf{q}_t^0]))}{\sum\limits_{ij} \exp(W_1 \tanh(W_2[\mathbf{q}_t^0, \mathbf{w}_i^j]))} \quad (10)$$

where W_1, W_2 are trainable parameters. We then compute the summarized context representation $\mathbf{g}_t^1$ and the next hop query as follows:

$$\mathbf{g}_t^1 = \sum\limits_{ij} a_{ij}^1 \mathbf{w}_i^j \quad (11)$$

$$\mathbf{q}_t^1 = \mathbf{q}_t^0 + \mathbf{g}_t^1 \quad (12)$$

We repeat this for H hops. The attention vector after H hop is represented $\mathbf{a}^H$. The generate distribution P_g is given by:

$$P_g = \text{Softmax}(W_3[\mathbf{h}_t, \mathbf{g}_t^1] + b_1) \quad (13)$$

where W_3 and b_1 are trainable parameters. The context copy distribution P_{con} is computed as follows:

$$P_{con}(y_t = w) = \sum\nolimits_{ij:w_i^j = w} a_{ij}^H \quad (14)$$

The KB copy distribution P_{kb} is given by,

$$\beta_m^t = \text{Softmax}(W_4 \tanh(W_5[\mathbf{g}_t^H, \mathbf{h}_t, \mathbf{r}_m])) \quad (15)$$

$$\gamma_n^t = \text{Softmax}(W_6 \tanh(W_7[\mathbf{g_t}^H, \mathbf{h}_t, \mathbf{k}^n])) \quad (16)$$

$$P_{kb}(y_t = w) = \frac{\sum_{mn:v_m^n = w} d_m \beta_m^t \gamma_n^t}{\sum_{mn} d_m \beta_m^t \gamma_n^t} \quad (17)$$

where W_4, W_5, W_6 and W_7 are trainable parameters. Now we compute the gate α to combine P_{con} and P_{kb} to get a final copy distribution P_c as follows:

$$\mathcal{K}_t = \sum_m d_m \beta_m^t \mathbf{r}_m \quad (18)$$

$$\alpha = \text{Sigmoid}(W_8[\mathbf{h}_t, \mathbf{g}_t^H, \mathcal{K}_t]) \quad (19)$$

$$P_c = \alpha P_{kb}(y_t) + (1 - \alpha) P_{con}(y_t) \quad (20)$$

where W_8 is a trainable parameter.

B.3 Loss

We compute the cross entropy loss over the generate P_g and copy P_c distribution as follows:

$$\mathcal{L}_g = -\sum_{t=1}^{T} \log(P_g(y_t)) \quad (21)$$

$$\mathcal{L}_c = -\sum_{t=1}^{T} \log(P_c(y_t)) \quad (22)$$

C Distillation Visualisation

We show the visualisation of how the KB distillation distribution helps the decoder rectify the incorrect KB memory pointer inference in Figure 4. Figure 5 shows how the KB distillation distribution helps increase the confidence associated with the correct entity in the KB.

D Datasets

We present statistics of SMD, CamRest and WOZ in Table 6.

	SMD	CamRest	WOZ
Train Dialogs	2425	406	1839
Val Dialogs	302	135	117
Test Dialogs	304	135	141

Table 6: Statistics of the three datasets.

E Domain-Wise Results

Table 7 and Table 8 show the domain wise entity F1 scores of SMD and WOZ datasets respectively. We note that CDNET either has the best or the second-best performance in domain wise scores.

F Qualitative Example

Table 9 shows responses predicted by CDNET, EER and FG2Seq for an example from the WOZ dataset.

G Human Evaluation

Figure 6 shows a screenshot of the task used for collecting human judgements.

Model	BLEU	F1	MSE F1	Cal	Wea	Nav
MLM	17.0	54.6	-	66.7	56	46.9
DSR	12.7	51.9	-	52.1	50.4	52.0
Ent. Const.	13.9	53.7	-	55.6	52.2	54.5
TTOS	17.4	55.4	-	63.5	**64.1**	45.9
DFNet	14.4	62.7	-	73.1	57.6	**57.9**
GLMP	13.9	59.6	59.6	70.2	58.0	54.3
EER	17.2	59.0	55.1	71.8	57.8	52.5
FG2Seq	16.8	61.1	59.1	73.3	57.4	56.1
CDNET	**17.8**	**62.9**	**62.9**	**75.4**	61.3	56.7

Table 7: Domain wise Entity F1 performance of CDNET and baselines on the SMD dataset.

Model	BLEU	F1	MSE F1	Hot	Att	Res
DSR	9.1	30.0	-	27.1	28.0	33.4
DFNet	9.4	35.1	34.8	30.6	28.1	40.9
GLMP	6.9	32.4	-	28.1	24.4	38.4
EER	13.6	35.6	35.0	35.7	**43.0**	34.2
FG2Seq	14.6	36.5	36.0	34.4	37.2	38.9
CDNET	11.9	**38.7**	**38.6**	**36.3**	38.9	**41.7**

Table 8: Domain wise Entity F1 performance of CDNET and baselines on WOZ dataset.

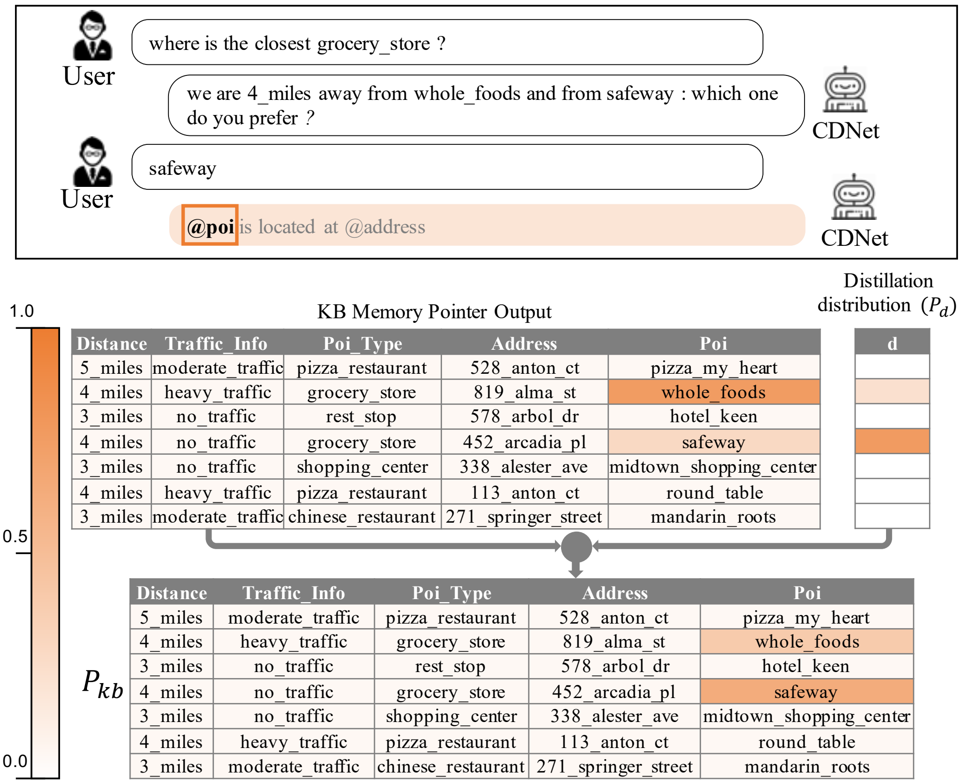

where is the closest grocery_store ?

we are 4_miles away from whole_foods and from safeway : which one do you prefer ?

safeway

@poi is located at @address

Distance	Traffic_Info	Poi_Type	Address	Poi
5_miles	moderate_traffic	pizza_restaurant	528_anton_ct	pizza_my_heart
4_miles	heavy_traffic	grocery_store	819_alma_st	whole_foods
3_miles	no_traffic	rest_stop	578_arbol_dr	hotel_keen
4_miles	no_traffic	grocery_store	452_arcadia_pl	safeway
3_miles	no_traffic	shopping_center	338_alester_ave	midtown_shopping_center
4_miles	heavy_traffic	pizza_restaurant	113_anton_ct	round_table
3_miles	moderate_traffic	chinese_restaurant	271_springer_street	mandarin_roots

d

Distance	Traffic_Info	Poi_Type	Address	Poi
5_miles	moderate_traffic	pizza_restaurant	528_anton_ct	pizza_my_heart
4_miles	heavy_traffic	grocery_store	819_alma_st	whole_foods
3_miles	no_traffic	rest_stop	578_arbol_dr	hotel_keen
4_miles	no_traffic	grocery_store	452_arcadia_pl	safeway
3_miles	no_traffic	shopping_center	338_alester_ave	midtown_shopping_center
4_miles	heavy_traffic	pizza_restaurant	113_anton_ct	round_table
3_miles	moderate_traffic	chinese_restaurant	271_springer_street	mandarin_roots

Figure 4: Attention visualization of a decode time step of an example from SMD dataset. P_{kb} corresponds to the sketch tag *@poi*. P_{kb} is computed by combining the output of the KB memory pointer and the distillation distribution P_d.

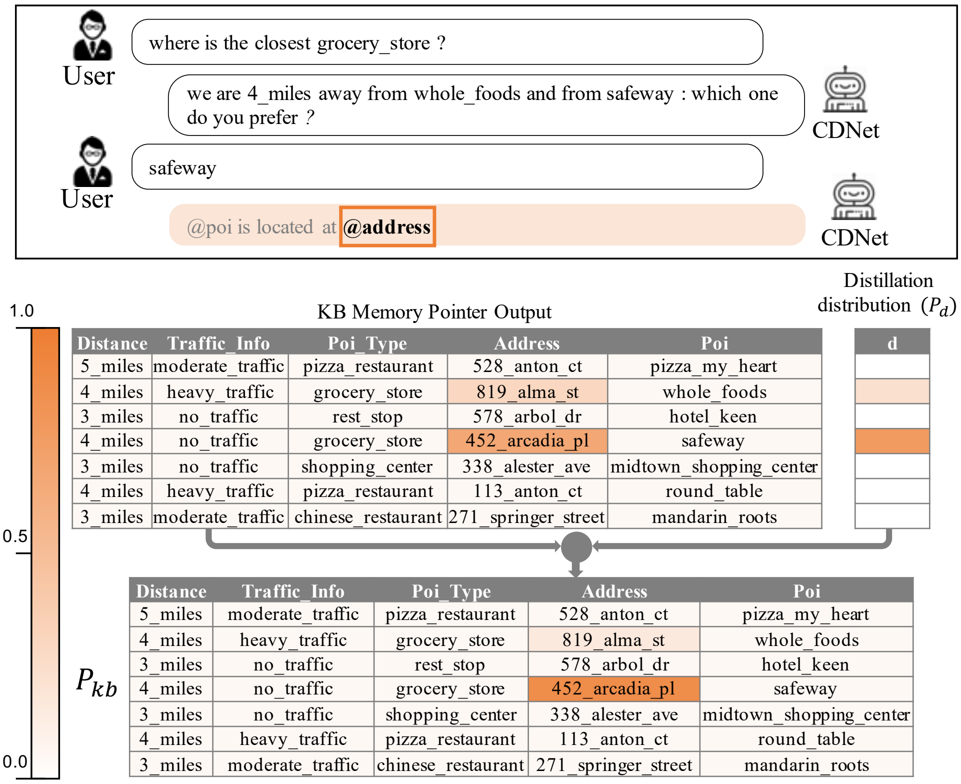

where is the closest grocery_store ?

we are 4_miles away from whole_foods and from safeway : which one do you prefer ?

safeway

@poi is located at **@address**

Distance	Traffic_Info	Poi_Type	Address	Poi
5_miles	moderate_traffic	pizza_restaurant	528_anton_ct	pizza_my_heart
4_miles	heavy_traffic	grocery_store	819_alma_st	whole_foods
3_miles	no_traffic	rest_stop	578_arbol_dr	hotel_keen
4_miles	no_traffic	grocery_store	452_arcadia_pl	safeway
3_miles	no_traffic	shopping_center	338_alester_ave	midtown_shopping_center
4_miles	heavy_traffic	pizza_restaurant	113_anton_ct	round_table
3_miles	moderate_traffic	chinese_restaurant	271_springer_street	mandarin_roots

d

Distance	Traffic_Info	Poi_Type	Address	Poi
5_miles	moderate_traffic	pizza_restaurant	528_anton_ct	pizza_my_heart
4_miles	heavy_traffic	grocery_store	819_alma_st	whole_foods
3_miles	no_traffic	rest_stop	578_arbol_dr	hotel_keen
4_miles	no_traffic	grocery_store	452_arcadia_pl	safeway
3_miles	no_traffic	shopping_center	338_alester_ave	midtown_shopping_center
4_miles	heavy_traffic	pizza_restaurant	113_anton_ct	round_table
3_miles	moderate_traffic	chinese_restaurant	271_springer_street	mandarin_roots

Figure 5: Attention visualization of a decode time step of an example from SMD dataset. P_{kb} corresponds to the sketch tag *@address*. P_{kb} is computed by combining the output of the KB memory pointer and the distillation distribution P_d.

Address	Area	Choice	Name	Postcode	Pricerange	Ref	Food
regent_street_city_centre	centre	a_lot	pizza_hut_city_centre	cb21ab	cheap	oog9kq5v	italian
4_-_6_rose_crescent	centre	a_lot	la_raza	cb23ll	cheap	oog9kq5v	spanish
finders_corner_newmarket_road	east	a_lot	the_missing_sock	cb259aq	cheap	oog9kq5v	international
106_regent_street_city_centre	centre	a_lot	curry_garden	cb21dp	expensive	oog9kq5v	indian

usr-1	i need a **cheap spanish** restaurant . are there any around ?
sys-1	yes , **la_raza** is a restaurant serving **spanish** food in the **cheap** price range . it is located at **4_-_6_rose_crescent** . would you like a reservation ?
usr-2	i would like a restaurant in the **centre** .
sys-2	**la_raza** is a cheap restaurant serving spanish food in the centre part of town . it ’ s address is **4_-_6_rose_crescent** and it ’ s post code is **cb2311** .
usr-3	great . can you book me a table for 7 on tuesday at 11:45 and provide me with a reference number ?
Gold	all right , i ’ ve made a reservation . your reference number is **oog9kq5v** . what else can i do for you ?
EER	i ’ m sorry , but i was unable to book that for you . would you like to try another restaurant ?
FG2Seq	i ’ m sorry , but the **la_raza** restaurant restaurants . would you like me to find another restaurant ?
CDNet	i have booked you at **la_raza** . your reference number is **oog9kq5v** . is there anything else i can help you with ?

Table 9: Example dialog from WOZ along with responses generated by EER, FG2Seq and CDNet.

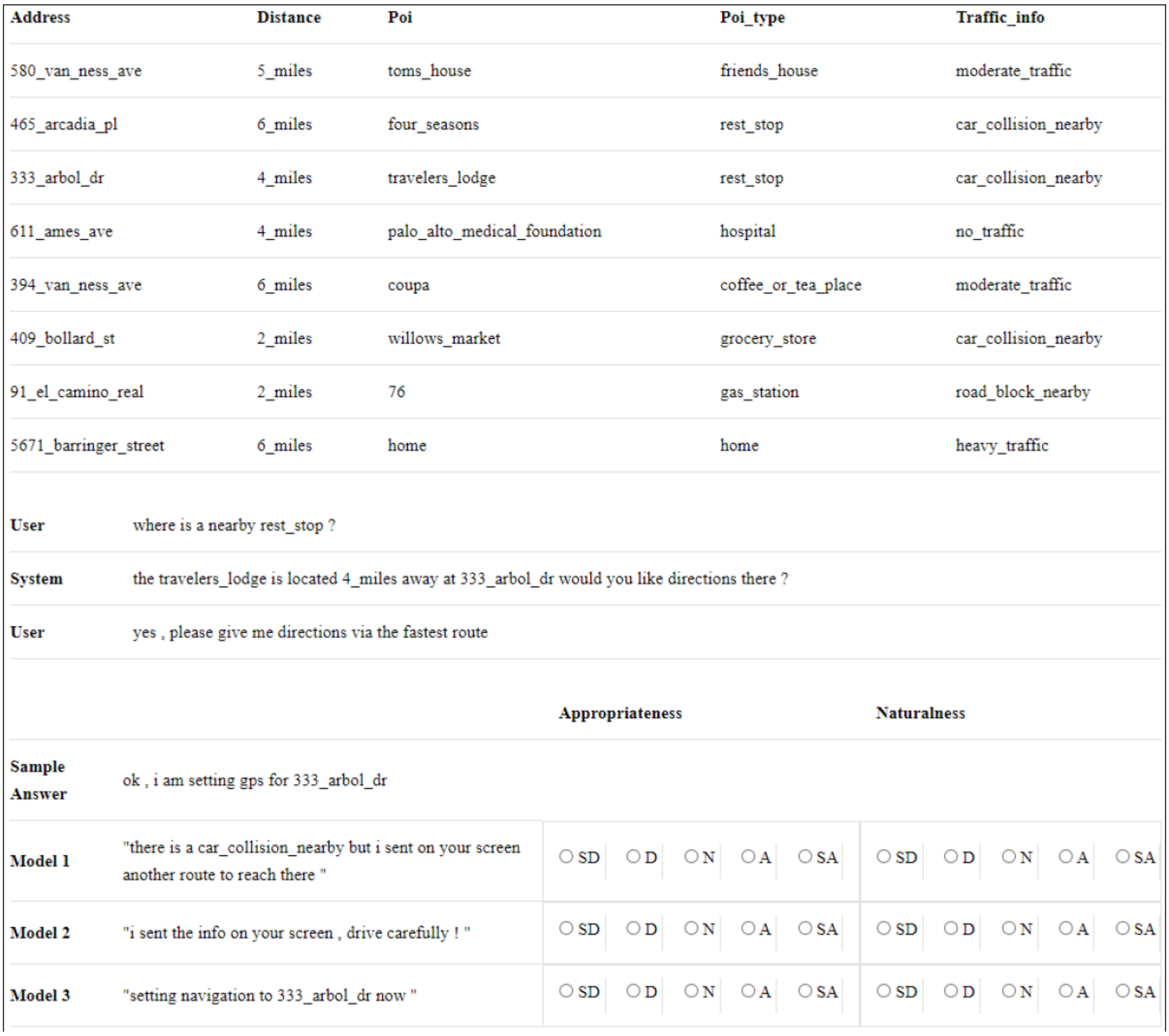

Address	**Distance**	**Poi**	**Poi_type**	**Traffic_info**
580_van_ness_ave	5_miles	toms_house	friends_house	moderate_traffic
465_arcadia_pl	6_miles	four_seasons	rest_stop	car_collision_nearby
333_arbol_dr	4_miles	travelers_lodge	rest_stop	car_collision_nearby
611_ames_ave	4_miles	palo_alto_medical_foundation	hospital	no_traffic
394_van_ness_ave	6_miles	coupa	coffee_or_tea_place	moderate_traffic
409_bollard_st	2_miles	willows_market	grocery_store	car_collision_nearby
91_el_camino_real	2_miles	76	gas_station	road_block_nearby
5671_barringer_street	6_miles	home	home	heavy_traffic

User	where is a nearby rest_stop ?
System	the travelers_lodge is located 4_miles away at 333_arbol_dr would you like directions there ?
User	yes , please give me directions via the fastest route

		Appropriateness	**Naturalness**
Sample Answer	ok , i am setting gps for 333_arbol_dr		
Model 1	"there is a car_collision_nearby but i sent on your screen another route to reach there "	○ SD ○ D ○ N ○ A ○ SA	○ SD ○ D ○ N ○ A ○ SA
Model 2	"i sent the info on your screen , drive carefully ! "	○ SD ○ D ○ N ○ A ○ SA	○ SD ○ D ○ N ○ A ○ SA
Model 3	"setting navigation to 333_arbol_dr now "	○ SD ○ D ○ N ○ A ○ SA	○ SD ○ D ○ N ○ A ○ SA

Figure 6: A sample human evaluation task used for collecting appropriateness and naturalness of responses generated by three (anonymized) models.

DIALOGSUM: A Real-Life Scenario Dialogue Summarization Dataset

Yulong Chen[♠♡], **Yang Liu**[♣], **Liang Chen**[♦], **Yue Zhang**[♡◇]
[♠] Zhejiang University
[♡] School of Engineering, Westlake University
[♣] Microsoft Cognitive Services Research
[♦] College of Software, Jilin University
[◇] Institute of Advanced Technology, Westlake Institute for Advanced Study
yulongchen1010@gmail.com *yaliu10@microsoft.com*
chenliang5518@mails.jlu.edu.cn *yue.zhang@wias.org.cn*

Abstract

Proposal of large-scale datasets has facilitated research on deep neural models for news summarization. Deep learning can also be potentially useful for spoken dialogue summarization, which can benefit a range of real-life scenarios including customer service management and medication tracking. To this end, we propose DIALOGSUM, a large-scale labeled dialogue summarization dataset. We conduct empirical analysis on DIALOGSUM using state-of-the-art neural summarizers. Experimental results show unique challenges in dialogue summarization, such as spoken terms, special discourse structures, coreferences and ellipsis, pragmatics and social common sense, which require specific representation learning technologies to better deal with.

1 Introduction

Text summarization is the task of automatically generating a concise, salient, coherent and fluent summary of a given set of documents (Radev et al., 2002). Thanks to the advance in neural network models and the availability of large-scale labeled datasets, recent research has achieved promising progress on summarizing monologic texts such as news articles (Paulus et al., 2018; Gehrmann et al., 2018; Liu and Lapata, 2019; Liu et al., 2020), patents (Pilault et al., 2020) and academic papers (Koncel-Kedziorski et al., 2019).

However, dialogue, as an important channel for achieving communicative intents (Bender and Koller, 2020), has received significantly less attention from the summarization research community. One main reason is the paucity of a suitable summarization dataset built on dialogue texts. Most existing research uses the AMI meeting corpus (Carletta et al., 2005), which consists of 137 dialogues obtained from virtual multi-party meeting recordings. However, research on the corpus is limited to its small scale. SAMSum (Gliwa et al., 2019) is a recently released *written* online dialogue summarization dataset, which contains 16k online chats with corresponding summaries. However, it focuses on conversations via messenger apps, which are rather short (around 94 tokens per conversation) and their language style and topics also differ from *spoken daily dialogues*.

(a) Dialogue from DIALOGSUM:
#Person_1#: Good morning. I wonder whether you have got an answer from your superior.
#Person_2#: Yes, we had a meting about it yesterday afternoon.
#Person_1#: What's the answer?
#Person_2#: We decided that we could agree to your price, but we are a bit worried about the slow delivery.
#Person_1#: Let me see. I quoted your delivery in three months, didn't I?
#Person_2#: Yes, but we hope that the wool could reach us as soon as possible.
#Person_1#: I thought you would. So I rang Auckland last night. As you are our biggest customer, they agreed to ship the order on the first vessel available that will leave Auckland next month.
#Person_2#: Good, if you agree we‘ll draft the agreement right away and sign it then.
#Person_1#: By all means.

Summary from DIALOGSUM: #Person_1# and #Person_2# agree to sign an agreement ***since*** #Person_1# could speed up the delivery as #Person_2# hopes.

(b) Dialogue from SAMSum:
…
Leo: BTW what are those pics?
Ryan: Pics from Italy!!! :):):):))))))))))
Leo: Yeah. They seem nice. (‘A`)
Ryan: That's all???? I need more reactions!!!!!!!!!!
Leo: I'm tied to this office and working like a slave. AM I SUPPOSED TO SAY \"I AM SO JEALOUS!!!!!!!!\"?😬😬😬
…

Summary from SAMSum: Ryan is in Italy while Leo is working hard and wishing he could win the lottery.

Figure 1: An example from DIALOGSUM dataset compared with an example from SAMSum dataset.

A comparison between the real-life scenario dialogue and online chat is shown in Figure 1. Online-chat messages contain unique tokens (e.g., “BTW”), emoticons (e.g., “:)”) and emojis (e.g., “😬”). In contrast, daily conversations have a different

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5062–5074
August 1–6, 2021. ©2021 Association for Computational Linguistics

Datasets	Lan. style	Domain	Scenario	Dialogs	Data size	#Tokens/dial.	#Tokens/turn	#Comp. rate
AMI	spoken	single	meeting	137	100hrs (video)	4,757	16.5	0.07
SAMSum	written	multiple	online	16,369	1.5M (token)	94	8.4	0.30
DIALOGSUM	spoken	multiple	daily life	13,460	1.8M (token)	131	13.8	0.18

Table 1: Comparison between DIALOGSUM and other public dialogue summarization datasets. Lan. stands for language. Dial. stands for dialogue. # stands for the average result. Comp. stands for compression. The compression rate is the ratio of the length of the summary divided by the length of the original text.

and more formal style. In addition, real-life dialogues have more diverse task-oriented scenarios and topics compared to online chit-chats. For example, online-chat messages in SAMSum are about leisure and social chats, but real-life dialogues contain business negotiation (Figure 1(a)). Intuitively, automatically summarizing such dialogues can help a business find common needs or complaints from customers. With the rise of personal assisting chatbots, summarizing dialogues from different aspects of daily life can also be useful for personal record management and other applications.

We introduce Real-Life Scenario Dialogue Summarization (DIALOGSUM), a large-scale summarization dataset for dialogues. Dialogue data for DIALOGSUM are collected from three public dialogue corpora, namely Dailydialog (Li et al., 2017), DREAM (Sun et al., 2019) and MuTual (Cui et al., 2020), as well as an English speaking practice website. These datasets contain face-to-face spoken dialogues that cover a wide range of daily-life topics, including schooling, work, medication, shopping, leisure, travel. Most conversations take place between friends, colleagues, and between service providers and customers. We clean and preprocess the dialogue data into a unified format, and ask annotators to summarize them from an observer perspective. Topics are also manually labeled for each dialogue. An example of DIALOGSUM is shown in Figure 1(a), where the summary expresses the main content in a business conversation.

The contribution of DIALOGSUM can be stated from two perspectives. First, from the perspective of downstream applications, summarizing daily spoken dialogues can be useful for both business and personal uses. Dialogue summaries can also be useful for personal assistants to keep track of important events as such business negotiation. Second, from the method perspective, DIALOGSUM has a larger scale of long dialogue data, which can facilitate the study of dialogue summarization using neural network models. The number of dialogues in DIALOGSUM is orders of magnitude larger than in AMI, which can be useful for training large neural network models for dialogue summarization. The average length of dialogues in DIALOGSUM is 39.8% longer than in SAMSum. To our knowledge, we are the first to release a large-scale real-life scenario dialogue summarization dataset.

We empirically investigate the performance of state-of-the-art neural summarization models on DIALOGSUM, comparing the characteristics of the spoken daily dialogue summarization dataset with standard news summarization benchmarks and the online chat summarization benchmark SAMSum. Experimental results show that DIALOGSUM is more amenable to abstractive summarizers, while being relatively more challenging compared to the existing summarization datasets. We find that main difficulties arise from discourse structures in multi-turn dialogues, as well as the need for book-keeping both entities and events mentioned in turns of utterances. We release our dataset at `https://github.com/cylnlp/DialogSum`.

2 The DIALOGSUM Dataset

2.1 Dialogue Data Preparation

Data Collection DailyDialog is a dataset consisting of 13k multi-turn dialogues, obtained from websites that aim to help English learners to practice English speaking. DREAM and MuTual are dialogue understanding datasets, consisting of 6k and 9k speech transcripts, respectively, both collected from online English listening exam materials. In order to further increase the diversity of data, we crawl additional dialogues from another English speaking practice website[1] which aims to provide English learners with conversation examples in real life practical circumstances, such as business negotiation and banking services.

Although dialogues of DIALOGSUM are from different sources, they all share important characteristics that are in line with what we expect. First, as mentioned earlier, these dialogues are under rich

---

[1] http://www.tingroom.com

Source	Num. dial.	Tokens	% in DIALOGSUM
DailyDialog	7837	980,398	58.22
DREAM	2028	301,098	16.94
MuTual	1870	260,139	13.89
Crawled	1725	238,902	12.82

Table 2: Proportions of dialogue sources in DIALOGSUM.

real-life scenarios. Unlike chitchats, these conversations have clear communication patterns and intents, making them more suitable and valuable to serve as summarization sources (Carletta et al., 2005). Moreover, their multi-turn dialogue lengths are within a reasonable scale and are longer than chitchats[2], which comforts the purpose of automatic summarization. Greater lengths also indicate these dialogues contain more events and discourse relations between them. Properly selecting vital events and identifying their relations make summarizing these dialogues more challenging.

Data Cleaning and Pre-Processing We delete non-English characters, correct typos and grammatical errors, and further filter out duplicated data based on text similarity. After deduplicating, proportions of the data sources are summarized in Table 2. Because of different data processing methods and annotation procedures, original dialogues in DailyDialog, DREAM and MuTual are in different formats. We follow previous work (Li et al., 2017; Zhang et al., 2018; Budzianowski et al., 2018; Dinan et al., 2019) and preprocess them into a bi-turn dialogue flow, merging continuous turns of the same speaker into one utterance. Also, we add tags (e.g. `#Person_1#` and `#Person_2#` in Figure 1(a)) before each dialogue turn, to distinguish speakers. The final DIALOGSUM dataset contains 13,460 dialogues, which are divided into training (12,460), validation (500) and test (500) sets.

2.2 Annotation

We ask annotators to write dialogue summaries based on following criteria: the summary should (1) convey the most salient information of the dialogue and; (2) be brief (no longer than 20% of the conversation length) and; (3) preserve important named entities within the conversation and; (4) be written from an observer perspective and; (5) be written in formal language.

Human Annotated Summary	R1	R2	RL
Summary1 to Summary2	52.90	26.01	50.42
Summary1 to Summary3	53.85	27.53	51.65
Summary2 to Summary3	53.30	26.61	50.44
Average	53.35	26.72	50.84

Table 3: ROUGE scores between three human annotated summaries in test set.

We require our annotators to pay extra attention to the following aspects.

Tense Consistency: Annotators should take the moment that the conversation occurs as the *present* time, and choose a proper tense to describe events *before* and *after* the ongoing conversation.

Discourse Relation: If summarized events hold important discourse relations, particularly causal relation, annotators should preserve the relations if they are also in the summary.

Emotion: Different from newspaper and academic articles, social conversations in DIALOGSUM are often implied with emotions. Therefore, we ask annotators to explicitly describe important emotions related to events in the summary.

Intent Identification: Rather than merely summarizing the consequences of dialogues, annotators should also describe speakers' intents in summaries, if they can be clearly identified.

In addition to the above, annotators should use person tags to refer to different speakers if real names cannot be detected from the conversation. Annotators are also asked to write a short (around 3 tokens) topic for each dialogue. Appendix A shows the list of topics.

2.3 Quality Control

To ensure quality, before formal annotation, we ask annotators to annotate training samples until they pass our examination and meet our requirements. After annotation, we check summaries by cross-validation between different annotators twice. During the checking process, bonus is paid to checkers who find unqualified summaries, and penalty is given to annotators whose annotation is found with mistakes. In case of appeal, we make the final decision. After the second checking, we sample 10% summaries and manually check the samples ourselves. If errors are found in an annotation batch, we ask corresponding annotators to self-check and re-annotate the whole batch and repeat this checking and sampling processes.

To further control the quality, and to analyze inter-annotator agreement, for each dialogue in the

---

[2] The average numbers of tokens for multi-turn dialogues are: Dailydialog: 118.8, DREAM: 124.6, MuTual: 136.1.

Dataset	% of novel n-grams				LEAD			LONGEST			EXT-ORACLE		
	unigram	bigram	trigram	4-gram	R1	R2	RL	R1	R2	RL	R1	R2	RL
CNN	16.75	54.33	72.42	80.37	29.15	11.13	25.95	-	-	-	50.38	28.55	46.58
DailyMail	17.03	53.78	72.14	80.28	40.68	18.36	37.25	-	-	-	55.12	30.55	51.24
NY Times	22.64	55.59	71.93	80.16	31.85	15.86	23.75	-	-	-	52.08	31.5	46.72
XSum	**35.76**	**83.45**	**95.50**	**98.49**	**16.30**	**1.61**	**11.95**	-	-	-	**29.79**	**8.81**	**22.65**
SAMSum	32.63	77.22	89.27	94.83	31.41	8.70	30.41	32.13	10.13	29.11	44.60	17.37	39.38
DIALOGSUM	26.28	76.94	89.16	94.53	27.52	6.78	27.31	24.15	6.25	22.73	37.90	13.88	34.04

Table 4: Corpora statistics and extractive methods on CNN/DailyMail, NY Times, XSum, SAMSum and DIALOGSUM. Part of results is from Narayan et al. (2018). All results are computed on test sets. For DIALOGSUM, the results are the average of multi-reference results.

test set, we provide three summaries written and checked by different annotators. For each test dialogue, we compare its and compute their pair-wise ROUGE (Lin, 2004) scores. Table 3 reports their averaged F_1 scores of ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL). We see R2 is relatively low while RL is high, which suggests that annotators' usage of language is variable, but the main content and logical order are mostly the same.

2.4 Characteristics of DIALOGSUM

We empirically compare DIALOGSUM with existing news summarization datasets and SAMSum. CNN/DailyMail (Hermann et al., 2015), NY Times (Sandhaus, 2008) and XSum (Narayan et al., 2018) are large-scale summarization datasets from the news domain, written in a monologic structure. XSum is a dataset designed specifically for abstractive summarization.

First, we compare the percentages of novel n-grams in the reference summary against the source document/dialogue. This intuitively reflects the level of abstraction of annotated summaries. As shown in Table 4, except for XSum, which is designed to be highly abstractive, dialogue-based summarization datasets contain more novel n-grams in the summaries. We also find that the percentage of novel unigrams in DIALOGSUM is 26%, 6% lower than in SAMSum, but novel bigrams, trigrams and 4-grams are about the same as SAMSum. We believe that the relatively lower novel unigram proportion in DIALOGSUM compared to SAMSum is because of our pre-processing and annotation criteria. SAMSum's summaries include real names, third-person singular pronouns, which can be diverse across the dialogues. In contrast, DIALOGSUM uses tags such as `#Person_1#` to refer to persons whatever they are subjective, objective, or possessive. This constrains the proportion of novel unigrams to be lower.

Second, we compare the datasets using several extractive summarization methods. Following previous summarization work (Liu, 2019; Pilault et al., 2020), we report R1, R2 and RL here. LEAD creates summaries by selecting the first n sentences from source texts. LONGEST is designed for dialogue summarization (Gliwa et al., 2019). It selects the n longest utterances as a summary, which gives better ROUGE scores than LEAD on SAMSum. EXT-ORACLE creates summaries by choosing n sentences that have the highest ROUGE against reference summaries. It can be viewed as an upper bound for extractive summarization. We report results of LEAD-3, LONGEST-3 and EXT-ORACLE-2 on SAMSum, and LEAD-2, LONGEST-2 and EXT-ORACLE-2 on DIALOGSUM, where n is searched for each dataset in range of 1 to 6.

The results are shown in Table 4. In terms of LEAD, DIALOGSUM sees the lowest R1 and R2 except for XSum, showing that it is in nature a highly abstractive summarization dataset. SAMSum is less abstractive than DIALOGSUM by all ROUGE scores, which is likely because the compression rate of SAMSum (0.30) is higher than DIALOGSUM (0.18) (Table 1). The higher compression ratio suggests the summary contains denser information in the original text. The same conclusion can be found by using the LONGEST method. By using the EXT-ORACLE method, we find that DIALOGSUM is the most challenging dataset for extractive summarizers except for XSum, which is carefully designed for evaluating abstractive summarizers.

3 Experiments

We experiment with several abstractive summarization baselines to further understand the characteristics and challenges of DIALOGSUM. Following Gliwa et al. (2019), we concatenate utterances of a dialogue as the input. For pretrained models, we only finetune them on corresponding datasets.

Model	**CNNDM**			**XSum**			**SAMSum**			**DIALOGSUM**		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
Transformer	40.21	17.76	37.09	29.41	9.77	23.01	37.20	10.86	34.69	35.91	8.74	33.50
UNILMV2$_{\text{BASE}}$	43.16	20.42	40.14	44.00	21.11	36.08	50.53	26.62	48.81	47.04	21.13	45.04
BART$_{\text{LARGE}}$	**44.16**	**21.28**	**40.90**	**45.14**	**22.27**	**37.25**	**53.12**	**27.95**	**49.15**	47.28	21.18	44.83

Table 5: Results of abstractive models on CNNDM, XSum, SAMSum and DIALOGSUM. For DIALOGSUM, we give the average of multi-reference results.

3.1 Models

Transformer We take Transformer (Vaswani et al., 2017) as a non-pretrained abstractive baseline. For dialogue summarization, we follow Gliwa et al. (2019), using the same hyper-parameters for news summarization[3], but changing the minimum length to 15. We train the 6-layer Transformer model with Adam (Kingma and Ba, 2014) for 100,000 steps. Copy attention mechanism is applied and the dropout rate is set to 0.1.

UNILMV2 UNILMV2 (Bao et al., 2020) is a recently released pretrained language model for autoencoding and partially autoregressive language modeling. Here we use UNILMV2$_{\text{BASE}}$ as a strong abstractive model. For dialogue summarization, we train the model with Adam for 100,000 steps with 2,000 warmup steps and learning rate is set to $1.5e^{-5}$.

BART BART (Lewis et al., 2020) is an encoder-decoder Transformer model pretrained on a large corpus using a denoising autoencoder task. We use the large version of BART and finetune it with 5,000 training steps/200 warmup steps for dialogue summarization. Learning rate is set to $3e^{-5}$.

3.2 Results

Table 5 presents the experimental results. In general, we find that non-pretrained abstractive models outperform LEAD (Table 4), and the best results are achieved by pretrained models, despite the fact that BART$_{\text{LARGE}}$ and UNILMV2$_{\text{BASE}}$ are pretrained on monologic texts.

Extractive Summary vs Abstractive Summary Transformer gives similar results on CNNDM and better results on XSum, SAMSum and DIALOGSUM compared to LEAD, and pretrained models show better performance than EXT-ORACLE on all datasets except for CNNDM. In particular, pretrained models outperform Transformer by $13.07 \sim 14.24\%$ RL on XSum, $14.12 \sim 14.46\%$ RL on SAMSum, and $11.33 \sim 11.54\%$ on DIALOGSUM, while only $3.05 \sim 3.81\%$ on CNNDM. We believe that it is because CNNDM is a highly extractive dataset (Section 2.4). The key to summarizing CNNDM is to correctly understand intersentence relations within long documents, and extract important sentences. In contrast, XSum, SAMSum and DIALOGSUM are more abstractive, which require a model to paraphrase. And the strong generation capability of pretrained models can bring great improvements on them. We also see that, for abstractive datasets, model performance decreases as document length grows (Avg. length: SAMSum - 93.8, DIALOGSUM - 131.1, Xsum - 431.1) and compression rate decreases (Comp. rate: SAMSum - 0.30, DIALOGSUM - 0.18, XSum - 0.05). This explains why SAMSum is the easiest dataset.

Spoken vs Written All three models perform better on dialogue summarization datasets, compared with XSum. This can be potentially because XSum is naturally highly abstractive, and thus more challenging. We also compare improvement brought by pretrained models that are trained on large written texts.

Still in Table 5, the improvement on DIALOGSUM is the least. BART$_{\text{LARGE}}$ outperform Transformer by 15.73% R1 on XSum, 15.92% R1 on SAMSum, but 11.37% R1 on DIALOGSUM. It demonstrates that SAMSum has overall more written style than DIALOGSUM, and also suggests that dialogue and monologue are different. This can be explained by the design of written app-chat annotation in SAMSum (Gliwa et al., 2019).

DIALOGSUM vs SAMSum As shown in Table 5, model performance is steadily lower on DIALOGSUM than SAMSum. As stated, DIALOGSUM is more abstractive, open-domain, and spoken analogous. One more possible reason for the lower performance on DIALOGSUM is the longer input size. To better quantify the difference between these two dialogue summarization datasets, we further evaluate Transformer trained on DIALOGSUM when tested on the SAMSum, and vice versa. As

---

[3] https://opennmt.net/OpenNMT-py/examples/Summarization.html

Trans. Test	R1	R2	RL
S2D	31.72(-5.48)	6.25(-4.61)	29.72(-4.97)
D2S	31.74(-4.17)	5.93(-2.81)	29.79(-3.71)

Table 6: Difference between DIALOGSUM (D) and SAMSum (S). Trans. stands for Transferred.

Summary	Fluency	Cons.	Relevance	Coherence
Summary 1	5	5	4.96	5
Summary 2	5	5	4.98	5
Summary 3	5	5	5	5
Avg.	5	5	4.98	5
Transformer	4	2.08	2.3	3.84
UNILMV2$_{\text{BASE}}$	4.8	3.84	4.06	4.34

Table 7: Human evaluation on human annotated summaries and model generated Summaries. Cons. stands for Consistency. Summary 1 - Summary 3 correspond to three summaries of a dialogue.

shown in Table 6, the performance of Transformer drops greatly when traiend on DIALOGSUM and tested on SAMSum, and vice versa. This shows that the two datasets have substantial differences. In addition, Transformer trained on DIALOGSUM performs better than on SAMSum, and shows lower performance drop, suggesting that DIALOGSUM can provide more generalization ability for training dialogue summarization models.

4 Human Evaluation

To better understand DIALOGSUM, we take a deeper investigation into the outputs of Transformer and UNILMV2 on DIALOGSUM by conducting human evaluation from multiple aspects.

Fluency, Consistency, Relevance and Coherence First, following Kryscinski et al. (2019, 2020), we implement human evaluation from four dimensions. *Fluency* evaluates the quality of individual generated sentences, *Consistency* evaluates the factual alignment between the source text and generated summary, *Relevance* evaluates the importance of summary content, and *Coherence* evaluates the collective quality of all sentences.

We randomly select 50 dialogues and their summaries from DIALOGSUM test, and ask a judge to give scores in scale from 1 to 5 along the four mentioned dimensions. The higher, the better. The judge also gives scores to human-annotated summaries to evaluate their quality. As shown in Table 7, human annotated summaries receive the best scores from all dimensions. UNILMV2$_{\text{BASE}}$ has steadily better scores than Transformer, but lower than human. Model-generated summaries have the highest scores on Fluency, while lowest on Consistency. It suggests that although model-generated summaries are grammatical and fluent, they still contain factual errors.

Model	Human Scores				ROUGE Scores		
	-1	0	1	Avg.	R1	R2	RL
Transformer	80%	17%	3%	-0.77	34.35	7.01	31.13
UNILMV2	43%	37%	20%	-0.23	43.78	17.91	40.97

Table 8: Human evaluation on discourse relations, with corresponding ROUGE scores on the sub-test set. Avg. stands for the averaged score here.

Discourse Relation Reasonable summaries should convey important relations between main events, and identifying discourse relations and using proper phrases to express them in summaries can be challenging for summarization systems (Xu et al., 2020). Take Figure 1 (a) for example, the human annotated summary connects two main events (underlined) using "*since*" to express their causal relation explicitly. However, the causal relation between those two events are not explicitly expressed in the dialogue, and the distance between them is long. Multiple turns usually correspond to more complicated discourse structure and relation. Also, similar with Chen and Yang (2020), we find that model performance decreases when the number of dialogue turns grows (See Appendix B).

To better evaluate model ability to disambiguate discourse relations in DIALOGSUM, we first collect discourse connectives from Penn Discourse Treebank (Miltsakaki et al., 2004), and check whether these connectives are included in summaries in the testset. If the three reference summaries of a dialogue all contain connectives, we assume that the dialogues have strong discourse signals. We choose 70 dialogues from DIALOGSUM in this way.

We then ask linguists who specialize in discourse to evaluate model outputs and give scores from $\{-1, 0, 1\}$, where 1 means that the generated descriptions of main events are reasonable and contain correct discourse connectives, 0 means that the descriptions are good but contain no discourse connectives and -1 means that the description is either incorrect or contains incorrect connectives. We ask the linguists to focus only on clauses or phrases that are essential to discourse relations, and ignore syntactic errors. We report the distribution of annotated scores in Table 8.

We can see that the most summaries generated by Transformer are scored as -1, and their average score is -0.77, close to -1. This means that Transformer is not only incapable of identifying discourse relations but also incapable of generating the main events correctly. UNILMV2 has a relatively smooth distribution over three categories and a better average score of -0.23, which is closer to 0, suggesting that UNILMV2 can mostly choose important events amongst the conversation. But the -1 still holds most proportion and its average result is still far from 1, indicating its incapability of understanding relations between events.

Compared to the full test set, the model performance on this sub-set generally decreases ($1.56 \sim 3.26\%$ lower of R1, $1.73 \sim 3.22\%$ of R2, $2.37 \sim 4.07\%$ of RL), which also suggests complicated discourse relations between events make summarization more difficult. The results indicate that further research is necessary for better representing dialogue discourse structures in order to obtain more reliable summarization systems.

Coreference Information To evaluate model's ability to distinguish different interlocutors, we ask a judge to evaluate whether interlocutors' names and their conversation actions/contents are correctly associated in the 50 randomly selected data, and give scores from $\{-1, 0, 1\}$, where 1 means that all names and actions/content in the summary are associated correctly, 0 means partial incorrectly, and -1 means all incorrectly. Here, we only focus on coreference information in generated summaries, and ignore other errors, such as incorrect syntax or failing to summarize salient information.

We report the distribution of annotated scores in Table 9. Most Transformer generated summaries are annotated as -1 and the average result is close to -1, suggesting that Transformer cannot generate clauses that express the same relation between arguments and predicates in original dialogues. The UNILMV2$_{\text{BASE}}$ has more 0-scored summaries, and the result is much higher, yet closer to 0, which indicates that although UNILMV2$_{\text{BASE}}$ can generate summaries containing correct clauses, but still have much inconsistency. The performance of both models indicates that Transformer is only capable of extracting important word-level information from dialogues in DIALOGSUM, while UNILMV2$_{\text{BASE}}$ shows better performance on clause-level — it can identify the speakers and partially preserve coreference information, consistent with findings of Levesque et al. (2012) that pretraining is useful for coreference resolution. However, it is far from human annotations.

Model	Human Scores				ROUGE Scores		
	-1	0	1	Avg.	R1	R2	RL
Transformer	66%	28%	6%	-0.6	35.68	8.49	32.77
UNILMV2	4%	56%	40%	0.36	47.46	21.33	44.93

Table 9: Human evaluation on models' ability of preserving coreference information on DIALOGSUM, with corresponding ROUGE scores. Avg. stands for the averaged score here.

Summary	Human Scores	
	-1	1
Summary 1	7.7%	92.3%
Summary 2	20.5%	79.5%
Summary 3	17.9%	82.1%
Avg.	15.4%	84.6%
Transformer	84.6%	15.4%
UNILMV2	30.8%	69.2%

Table 10: Human evaluation on models' ability of identifying interlocutors' intents.

Intent Identification As stated in Section 2.2, we ask annotators to include important intents of interlocutors in their summaries, addition to the consequences of dialogues. The intent here refers to the motivation of a speaker to initiate a conversation, e.g. "*want to do an annual physical*" (c.f. Figure 2, DIALOGUE-A). This can make summaries more comprehensive and readable. Therefore, we conduct corresponding human evaluation on whether interlocutors' intents are described in summaries in the 50 randomly selected data.

We first ask a judge to evaluate whether the intent is important to a dialogue, and we select 39 dialogues that contain important intents. Then, we ask the judge to give scores from $\{-1, 1\}$, where 1 means that intents are identified correctly, -1 means incorrectly. Note that we only focus on intent identification in the summary, and other errors should be ignored. We also ask the judge to evaluate human annotated summaries.

The distribution of annotated scores is shown in Table 10. We see that most summaries generated by Transformer are scored as -1, which means that Transformer is incapable of generating summaries that correctly convey speakers' intents. UNILMV2$_{\text{BASE}}$ shows much better performance, however, it is still below human performance.

5 Challenges in DIALOGSUM

Compared to written texts, spoken dialogues can be more difficult for models to understand, and

DIALOGUE - A:	**DIALOGUE - B:**
#Person_1#: Hello, so how are we feeling today? **#Person_2#:** Things are going well for me, doctor. **#Person_1#:** Am I correct in thinking that you are here for your annual physical? **#Person_2#:** Yes, I am applying for new health insurance, and I need a physical examination to qualify. **#Person_1#:** Your basic physical exam will include lungs, heart, blood levels, and eyes, ears, and nose. **#Person_2#:** I've been having a little trouble breathing. Would you look into that, please? **#Person_1#:** We can do an allergy test, and later I can send you for an asthma test. **#Person_2#:** I would appreciate it. When you give me a blood test, what are you looking for? **#Person_1#:** I am going to check your cholesterol, blood sugar, and white blood cell count. **#Person_2#:** I am expecting the tests to go well. I have been taking good care of myself.	**#Person_1#:** Good morning. What can I do for you? **#Person_2#:** I'm in Room 309. I'm checking out today. Can I have my bill now? **#Person_1#:** Certainly. Please wait a moment. Here you are. **#Person_2#:** Thanks. Wait…What's this? The 30 dollar for? **#Person_1#:** Excuse me… The charge for your laundry service on Nov. 20th. **#Person_2#:** But I didn't take any laundry service during my stay here. I think you have added someone else's. **#Person_1#:** Ummm… Sorry, would you mind waiting a moment? We check it with the department concerned. **#Person_2#:** No. As long as we get this straightened out. **#Person_1#:** I'm very sorry. There has been a mistake. We'll corrected the bill. Please take a look. **#Person_2#:** Okay, here you are. **#Person_1#:** Goodbye.
SUMMARY – A1: #Person_2# wants to do an annual physical examination to apply for new health insurance and says #Person_2#'s breathing is not good. #Person_1# explains the items and will do tests on #Person_2#'s breathing. **SUMMARY – A2:** #Person_1# explains the checking items in #Person_2#'s annual physical examination and will do test to look into #Person_2's breathing. **SUMMARY – A3:** #Person_2# is going through an annual physical examination to apply for new health insurance, and #Person_2# asks #Person_1# to look into the breathing.	**SUMMARY – B1:** #Person_2# is checking out and asks #Person1# for the bill. #Person1# gives #Person_2# a wrong bill at first then corrects it. **SUMMARY – B2:** #Person_1# helps #Person_2# correct a mischarged bill on laundry service and helps #Person_2# check out. **SUMMARY – B3:** #Person_2# finds #Person_2# being mischarged. #Person_1# corrects the bill and #Person_2# pays for it.
UniLMv2: #Person_2# comes to #Person_1#'s annual physical to apply for new health insurance. #Person_1# will do an allergy test, an asthma test, and a blood test.	**UniLMv2:** #Person_2# is checking out. #Person_1# finds #Person_2# has added someone else's laundry service . #Person_1# apologizes and will correct the bill.
Transformer: #Person_2# goes to #Person_1# for an annual physical examination. #Person_1# will send #Person_1# for an asthma test and what #Person_2# eats.	**Transformer:** #Person_2# checks out with #Person_2#'s assistance and thinks they'll be very sorry for the laundry service.

Figure 2: Case study on DialogSum. Dialogue-A - a doctor and a patient dialogue, Dialogue-B - a customer and a hotel service dialogue.

to summarize (Goo and Chen, 2018). Therefore, we conduct error analysis and case studies on DialogSum to quantitatively and qualitatively discuss such challenges.

5.1 Error Analysis

We make error analysis on the 50 selected model-generated summaries (Section 4). Table 11 summarizes the five most frequent error types and their error rates. In general, UniLMv2$_{\text{BASE}}$ shows better performance than Transformer, but its error rates are still high. In particular, incorrect **coreference** (c.f. Section 4) sees the highest error rates for both models, indicating that models can be confused because of interactive information flow. Compared with Transformer, UniLMv2$_{\text{BASE}}$ can greatly avoid errors regarding unfactual information (-52%) and syntactic (-50%). However, it still suffers from coreference issues, and tends to generate redundant summaries.

Error Type	Transformer	UniLMv2$_{\text{BASE}}$
Incorrect Coref.	94%	60%
Missing Salient Inf.	64%	32%
Redundant Inf.	62%	44%
Unfactual Inf.	74%	22%
Syntactic Error	72%	22%

Table 11: Error analysis of model performance on DialogSum. Coref. stands for coreference, and Inf. stands for Information.

5.2 Case Study

We demonstrate two dialogues and their human-annotated/system-generated summaries in Figure 2.

First, a big challenge posed by spoken dialogues is that their **information flow** is different from monologic text, which is intuitively reflected in the dialogue **discourse structures** (Wolf and Gibson, 2005). For example, two utterances can be closely related even where there is a large distance between them. Such phenomena are common in spoken dialogues such as negotiations and procedures (e.g., medical consultation and police reports). Due to the unique structure of the spoken dialogue, important information is rather dispersed than well-structured monologues and written-dialogues.

Regular greetings can be useless to written dialogue summaries (e.g. SAMSum), which is reflected by that LEAD is worse than LONGEST on SAMSum (Table 4). In contrast, LEAD outperforms LONGEST by over 3% on DIALOGSUM. This is because, for spoken dialogues, such utterances sometimes express and indicate essential intents of speakers (c.f. Section 4). **Farewells** also express the dialogue consequence and future plan of the speakers (e.g. dialogues in Figure 2). Besides, **interruptions** appear frequently in the middle of conversations (Figure 2, DIALOGUE-B). These interruptions make other speaker's utterances incomplete, adding redundant information, and can also destroy coherent discourse structures, making dialogues more difficult to encode. These characteristics also make information in DIALOGSUM dialogues more dispersed than existing datasets.

Second, **coreference** and **ellipsis** are frequent in spoken dialogues (Grosz et al., 1995; Quan et al., 2019). It is a natural behavior of communication that humans obey as a rhetorical principle for saving words and avoiding repetitions. Although it can be trivial for humans, their understanding can be challenging to a neural model. For example, to correctly generate "*mischarged/wrong*" in SUMMARY-B1-SUMMARY-B3, models need to understand "*I think you have added someone else's (laundry service on my bill)*", where "*my bill*" refers to "`#Person_2#`*'s bill*".

Third, **pragmatics** and **social common sense** give a unique challenge for spoken language understanding and has a significant impact on summarization. From the last two sentences of DIALOGUE-B, human could understand that the "*Here you are*" is actually "*make a payment*", and "*Goodbye*" indicates that the event "*check out*" is finished. It requires commonsense knowledge to fully understand such dialogues. Beside, dialogues are summarized from a *different* perspective (compared with speakers' perspective), which suggests that summarizing dialogues needs to go beyond summarizing dialogue contents, but also dialogue actions at the **pragmatic** level. For example, "*explains*" in SUMMARY-A1 and SUMMARY-A2 summarizes multiple dialogue actions of `#Person_1#`, "*agree*" in Figure 1 (a) summarizes actions of both speakers. It requires model to not only summarize *what speakers are saying*, but also *what they are doing*.

6 Conclusion

We presented DIALOGSUM, a large-scale dialogue summarization dataset, investigating its characteristics and challenges empirically. Experiments with typical models show that DIALOGSUM is highly abstractive, and poses unique challenges in discourse and complex co-references. From these observations, we made discussion on the uniqueness of spoken dialogue summarization, listing several key problems to consider in future modeling. To our knowledge, we are the first to release a large-scale dataset for real-life scenario dialogue summarization.

7 Ethics Consideration

As mentioned, we collect our data from DailyDialog, DREAM and MuTual that all are public for academic use. The additional data are from www.tingroom.com, which are available to the public as well. The sources of our dialogue data are freely accessible online without copyright constraint to academic use.

We hired annotators who have degrees in English Linguistics or Applied Linguistics. Before formal annotation, we annotated 50 samples randomly extracted from the dataset, and calculated our average annotation time so we could set a fair salary for annotators' training annotation. During the training annotation process, they were paid as well. We also calculated the average annotation time for each dialogue during training, based on which we determined the final salary was around 9.5 dollars per hour. This hourly salary was the same for manual checking. All of our annotators took this annotation as a part-time job.

Acknowledgments

Yue Zhang is the corresponding author. We would like to thank the anonymous reviewers for their comments, and Xuefeng Bai, Ming Shen and Bonnie Webber for insightful discussion and proofreading. Also, we would like to thank Leyang Cui for sharing original dialogue data of MuTual dataset. Our sincere appreciation goes to annotators who have contributed to building this dataset. This work receives funding support from the Westlake University and Bright Dream Joint Institute for Intelligent Robotics, and a research grant from Rxhui Inc.

References

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 642–652. PMLR.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39. Springer.

Jiaao Chen and Diyi Yang. 2020. Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4106–4118, Online. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. MuTual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1406–1416, Online. Association for Computational Linguistics.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Chih-Wen Goo and Yun-Nung Chen. 2018. Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 735–742. IEEE.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. 2003. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yang Liu, Sheng Shen, and Mirella Lapata. 2020. Noisy self-knowledge distillation for text summarization. *arXiv preprint arXiv:2009.07032*.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse Treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online. Association for Computational Linguistics.

Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. 2019. GECOR: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4547–4557, Hong Kong, China. Association for Computational Linguistics.

Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–287.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

A Dialogue Topics

We use k-means (Likas et al., 2003) to cluster the dialogue topic with GloVe embedding (Pennington et al., 2014), where $k = 20$. Figure 3 presents the proportion of clustering results. Table 12 presents the cluster topics with corresponding id, which is assigned by human.

B Dialogue Turns

The number of conversation turns can have a direct impact on neural models. Multi-turn dialogues correspond to more complicated information flow and discourse structure. Following Chen and Yang (2020), we split test data based on dialogue turns, with a step size of 3, and show model performance on different dialogue turns.

The results are shown in Figure 4. The performance of Transformer and UNILMV2$_{\text{BASE}}$ decreases when number of turns grows, suggesting that more interactions between interlocutors and complicated discourse structures bring challenge. This phenomenon is also observed by (Chen and Yang, 2020) for SAMSum.

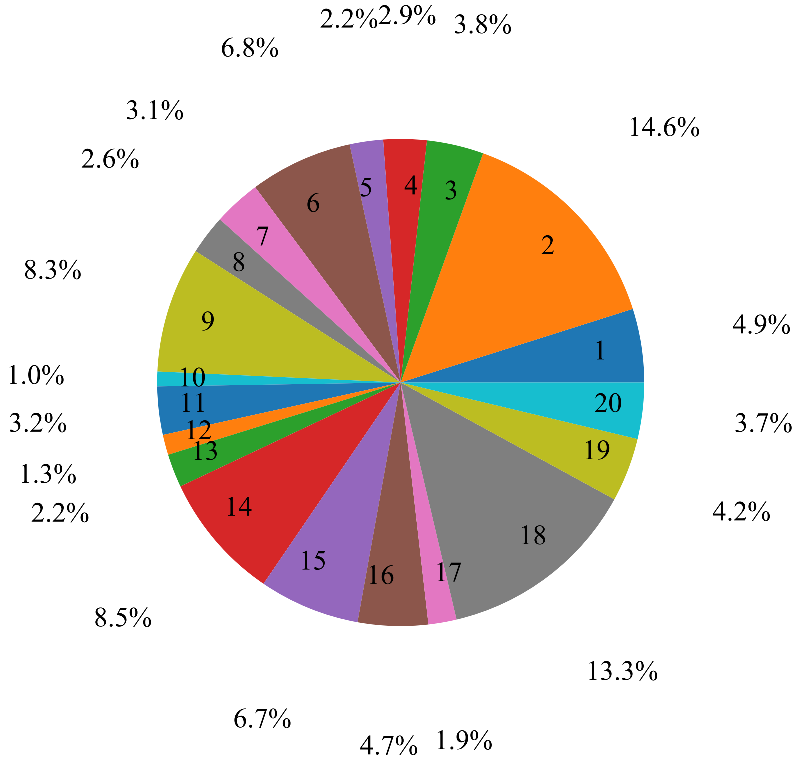

Figure 3: Proportion of dialogue topics.

ID	topic	ID	topic	ID	topic	ID	topic
1	interpersonal relation	6	education	11	personal and business appoinment	16	transportation
2	work and career	7	hobby	12	housing and apartment	17	in-store shopping
3	causal chitchat	8	interview	13	consultation	18	health and medicine
4	hotel and restaurant service	9	vacation plan	14	personal life	19	entertainment
5	sales	10	climate	15	economics	20	food ordering

Table 12: Topic clusters of DIALOGSUM.

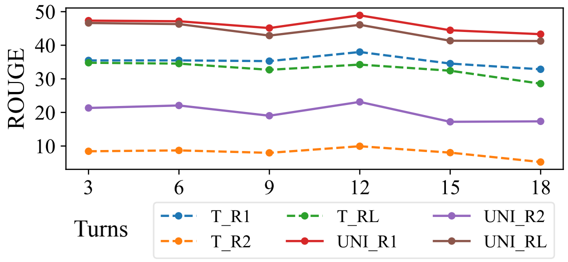

Figure 4: Model performance against the number of dialogue turns. T - Transformer. UNI - UNILMV2.

What did you refer to? Evaluating Co-references in Dialogue

Weinan Zhang[♯], **Yue Zhang**[♭], **Hanlin Tang**[◇], **Zhengyu Zhao**[♯], **Caihai Zhu**[♯], **Ting Liu**[♯]
[♯]Harbin Institute of Technology, Harbin, China
[♭]Westlake University, Hangzhou, China
[◇]School of Computer Science and Technology, Beijing, China
`{wnzhang, zyzhao, chzhu, tliu}@ir.hit.edu.cn`
`zhangyue@westlake.edu.cn`
`hltang@bit.edu.cn`

Abstract

Existing neural end-to-end dialogue models have limitations on exactly interpreting the linguistic structures, such as ellipsis, anaphor and co-reference, etc., in dialogue history context. Therefore, it is hard to determine whether the dialogue models truly understand a dialogue or not, only depending on the coherence evaluation of their generated responses. To address these issues, in this paper, we proposed to directly measure the capability of dialogue models on understanding the entity-oriented structures via question answering and construct a new benchmark dataset, DEQA, including large-scale English and Chinese human-human dialogues. Experiments carried on representative dialogue models show that these models all face challenges on the proposed dialogue understanding task. The DEQA dataset will release for research use.

1 Introduction

Driven by the growth of interest in social chatbot, online customer service and virtual mobile assistant, social dialogue systems have received increasing research attention (Cui et al., 2017; Zhou et al., 2018; Hancock et al., 2019). The current dominant method has been sequence-to-sequence models, trained over large dialogue data end-to-end. Such models use neural network architectures such as Transformer (Vaswani et al., 2017) to encode a user utterance and a dialogue history before generating a system utterance (Adiwardana et al., 2020; Roller et al., 2020; Bao et al., 2020). A major advantage is the use of standard and general model architecture, which facilitates end-to-end training process over large scale dialogue text (Shang et al., 2015; Zhang et al., 2018, 2019, 2020).

Dialogue		
U_1:	Well, you know how important a clean house is to your grandma.	
U_2:	Yes, I hear about it every time she comes here.	
	What do you hear about?	Q_1
	A clean house.	A_1
U_1:	She was the head janitor at St. Mary's Hospital for thirty years, after all.	
U_2:	I think she misses that job and wants to take it out on us.	
U_1:	You know, maybe she is just a neat freak.	
	Who is just a neat freak?	Q_2
	Grandma.	A_2
U_2:	I think she just likes to make us miserable.	
U_1:	You could be right.	

(a)

Co-reference Chain (OntoNotes style)			
Chain 1 (IDENT)		Chain 3 (IDENT)	
1.6-8	a clean house	1.12-12	grandma
2.5-5	it	2.8-8	she
Chain 2 (IDENT)		3.1-1	she
3.4-9	head janitor at St. Mary's Hospital	4.3-3	she
		5.4-4	she
4.5-6	that job	6.3-3	she

(b)

Table 1: (a) Sample of English dialogue in the proposed dataset. U_1 and U_2 are two interlocutors in the dialogue. Q_i and A_i (i=1,2) are clarification requests and the corresponding answers. (b) Co-reference chain annotation in OntoNotes 5.0 style.[1]

Despite showing effectiveness in empirical evaluation, existing work has a few important limitations. First, it is difficult to visualize or interpret the representation of dialogue state from a dense neural network encoder. In particular, there is not explicit representation of entities, semantic relations or discourse structures. Second, the performance of a dialogue system is evaluated directly by the quality of the generated responses. However, relatively little work has been done on evaluating how a system response is determined, which can be important because a proper response can be generated by simply relying on superficial and spurious patterns in dialogues, and we want to find out the cause of

---

[1] IDENT denotes the entities in a co-reference chain are identical. "1.6-8" indicates that "a clean house" is from the 6th to 8th tokens in the 1st utterance.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5075–5084
August 1–6, 2021. ©2021 Association for Computational Linguistics

problematic responses for identifying model limitations. ; To address such limitations, it can be useful to directly measure the quality of *dialogue understanding* by asking a dialogue model to identify important structures in dialogue histories. In this paper, we focus on entity level understanding, evaluating references to entities in a dialogue history context. Such references can include explicit anaphora and implicit mentions by using zero pronouns. Take Table 1 (a) as an example, where the dialogue history consists of 7 utterances and the second utterance contains a pronoun "*it*". At this point, we can measure system understanding of the dialogue state by checking whether the system can resolve the anaphora concerning "*a clean house*".

Our goal is to provide a large-scale benchmark and to evaluate the performance of social chatbot systems on dialogue understanding concerning entities. One way to define the task is to cast it as a co-reference resolution problem (Yin et al., 2017; Kong et al., 2019; Quan et al., 2019), where a benchmark can be constructed by manually labeling co-reference information on a dialogue dataset, as shown in Table 1 (b). However, such a benchmark does not fully meet our goal because a separate model is necessary for achieving co-reference resolution, and it may be challenging to seamlessly integrate such a co-reference module into a dialogue model being tested.

We take a different method, checking dialogue understanding of dialogue systems by inserting clarification requests (Schlangen, 2004; Stoyanchev and Johnston, 2015) into dialogues, and evaluating the response of dialogue systems on such requests. One example is shown in Table 1 (a), where we break a dialogue in the middle, adding clarification requests. For example, for the question "*Who is just a neat freak?*", the correct system response should be "*Grandma*", which reflects that the model has correct understanding of the dialogue context.

The advantage is three fold. First, this method allows the evaluation of a dialogue system without using an external probe task, by directly evaluating system generated responses. This makes our benchmark directly useful for evaluating arbitrary social dialogue models. In contrast to open-ended responses in chit-chats, responses for the proposed clarification requests are factual thus facilitating automatic evaluation. Second, it allows easier crowdsourcing for dataset construction as compared with co-reference resolution, which requires strict training of manual labelers for understanding linguistic concepts. It is thus useful for acquiring large-scale datasets. Such observation is consistent with recent work on other NLP tasks (FitzGerald et al., 2018; Roit et al., 2020). Third, this method allows easy extension to dialogue understanding beyond the entity reference level, such as event co-references, semantic relations and discourse level understanding. No new labeling standards are necessary for adding a new task.

According to the above observations, we create a large scale benchmark, open domain Dialogue Entity via Question Answering (DEQA), which consists of one English dataset and one Chinese dataset, of 8,415 and 6,203 dialogues, respectively. Each dialogue contains one or more questions similar to the one in Table 1. We choose to evaluate representative multi-turn neural dialogue systems, including models using Transformer (Vaswani et al., 2017) and DialoGPT (Zhang et al., 2020). Results show that the prevalent models of multi-turn dialogue generation face challenges in the co-reference questions. We will release the dataset at Github [2] for research use.

2 Dataset

We present the task (Section 2.1), the linguistic structures to evaluate (Section 2.2), the dataset construction (Section 2.3), the dataset characteristics (Section 2.4) and the evaluation metrics (Section 2.5) below.

2.1 Task Definition

Given a multi-turn dialogue, the task is to answer questions concerning one or more turns of the dialogue history. In particular, the model needs to answer questions about the anaphor and ellipsis phenomenons that appear in the context. It is worth noting that most of the answers can be extracted from the given context, but some answers may not explicitly appear in the context. These questions are called *summary questions*. The dialogue model should also have the capability on answering these summary questions.

We have already seen one example of English dialogue in Table 1. Table 2 shows a sample Chinese dialogue in the annotated dataset and its English translation. For the second utterance "我也想吃。。。(I also want to eat...)", the corresponding question is "你也想吃什么？(What do

---

[2] `https://github.com/adamszhu/DEQA`

	Chinese Dialogue	**English Translation**
U_1:	我想吃炸鸡。。。	I want to eat fried chicken...
U_2:	我也想吃。。。	I also want to eat... (literal translation)
U_1:	昨天买的炸鸡被我家猫吃了。	The fried chicken I brought yesterday was eaten by my cat.
U_2:	哈哈，是时候教育它了	Haha, it's time to teach it a lesson.
U_1:	不舍得啊	Unwilling to do that.
U_2:	我来替你啊	I can do it for you.

Table 2: Sample of Chinese dialogue in the annotated dataset.

you want to eat too?)" This question refers to the first utterance and the phrase "炸鸡(fried chicken)" should be extracted as the answer of the question. For the fourth utterance "哈哈，是时候教育它了。(Haha, it's time to teach it a lesson.)", there is a pronoun "它(it)" which should be resolved. A question "教育谁？（Teach whom a lesson?）" which refers to the fourth utterance is then raised. According to the third utterance, the answer of the question is "你的猫(your cat)". Note that in the proposed task, some answers should be summarized from the whole dialogue rather than only one utterance.

2.2 Linguistic Structures

In our dataset, a question is raised towards one ellipsis or anaphor phenomenon in a given dialogue. The role of a raised question can be seen as a "label" of a pronoun or zero pronoun. Correspondingly, the answer to the question is thus the antecedent of the pronoun or zero pronoun. Ellipsis, anaphor and co-reference are used frequently in natural language expression, especially in human-human dialogues (Quan et al., 2019). The examples in dialogues include:

- Ellipsis
  1) Zero anaphora (Noun phrase ellipsis)
  U_1: "你喜欢邦乔维的音乐吗？" ("*Do you like music of Bon Jovi?*")
  U_2: "是的(*Yes*)，我(*I*)喜欢(*like*)。"

The noun phrase "邦乔维的音乐" ("*music of Bon Jovi*") is omitted in the second utterance.

  2) Verbal phrase ellipsis
  U_1: "*I like the V6 engine of Audi S4.*"
  U_2: "*So do I.*"

Here, "*do*" is a trigger word which indicates the ellipsis of verbal phrase "*like the V6 engine of Audi S4*".

- Anaphor
  1) Personal pronoun
  U_1: "*Do you know Kelly Clarkson?*"
  U_2: "*Yes, she is my idol.*"

"*she*" is a personal pronoun which refers to "*Kelly Clarkson*".

  2) Demonstrative pronoun
  U_1: "*Have you ever made some family albums?*"
  U_2: "*Yes, these are my treasures.*"

Here, "*these*" refers to "*family albums*".

- co-reference
  U_1: "*There is a concert of Taylor Swift next month.*"
  U_2: "*Let us sing with Swifty together!*"

In this case, "*Swifty*" and "*Taylor Swift*" are co-reference.

2.3 Data Annotation

The English dialogue data are sourced from the DailyDialogue dataset (Li et al., 2017). The Chinese dialogue data are collected by ourself from Douban[3], a Chinese online forum. We randomly sample a subset of dialogues from the above two dataset respectively, and then annotate these dialogues in question answering.

For annotation, the first step is to identify ellipsis, anaphor and co-reference phenomenons of utterances in dialogue data. For each utterance, annotators determine whether the meaning of the utterance is complete when ignoring the dialogue context. If the meaning of an utterance is determined as incomplete, we can further identify ellipsis. Both zero anaphora and verbal phrase ellipsis are determined, and the anaphor can include both the personal and demonstrative pronouns. However, utterances such as "是(Yes)", "不是(No)", "好的(OK)" are not identified as ellipsis.

The second step is to raise questions to the identified zero anaphora, personal pronouns, demonstrative pronouns and corefered entities. We require that the questions are not simply obtained

---

[3] `https://www.douban.com/`

	English	Chinese
Total # of dialogues	8,415	6,203
Total # of questions	11,904	10,387
Training set	6,603	4,962
Dev set	832	621
Test set	830	620

Table 3: Statistics of the annotated dialogue dataset. # denotes "number".

by adding interrogative words in the original utterances. The third step is to give answers of each question, which may be an entity, a phrase, a chunk, a clause or a fragment of an utterance. Table 3 presents the statistics of the annotated dialogue dataset. In the end, we have 11,904 questions being labeled in 8,415 English dialogues, and 10,387 in 6,203 Chinese dialogues.

2.4 Characteristics

The characteristics of the annotated dialogue dataset include:

1) The dialogues are real human-human conversations;

2) Each dialogue is annotated with one or more questions and each question is related to at least an utterance of a dialogue;

3) The answer of a question may appear in one or more utterances of the dialogue. It means that an answer may be a composition of fragments that are from different utterances rather than a continuous span of an utterance. We analyze the types of answers in the annotated dataset below.

First, Table 4 shows the number and percentage of different types of answers.

	English		Chinese	
	#	%	#	%
Entity	4,765	40.03	5,810	55.94
Phrase	2,298	19.30	2,203	21.21
Clause	87	0.73	755	7.26
Fragment	4,754	39.94	1,619	15.59

Table 4: Statistics of the answer types in the annotated dataset. # denotes "number" and % denotes percentage.

We can see that most of the answers are entity, phrase and fragment. The proportion of the clause type in the Chinese dialogue data is about ten times that in the English dialogue data. The proportion of the fragment type in English is larger than that in Chinese.

Second, we give the statistics of the status of an answer that appears in a dialogue. Here, in Table 5, "Seq" and "Skip"denote the tokens of an answer are sequential and skipping in an utterance, respectively. "Cross" indicates that the tokens of an answer are from different utterances. "Summary" means that the tokens of an answer written by the annotator is not strictly from the dialogue.

	English		Chinese	
	#	%	#	%
Seq	11,055	92.87	9,831	94.65
Skip	145	1.22	206	1.98
Cross	146	1.23	195	1.88
Summary	552	4.68	147	1.49

Table 5: Statistics of the status of an answer that appears in a dialogue. # denotes "number" and % denotes percentage.

2.5 Evaluation Metrics

The Exact match and F1 are used to evaluate the performance of models.

Exact match (EM): the number of answers predicted by a model and exactly matched the gold answers divides to the total number of gold answers in test set.

F1: F1 is computed by precision(p) and recall(r), where $p = N_{matched}/N_{predAns}$ and $r = N_{matched}/N_{goldAns}$. For a predicted answer, the precision(p) equals to the number of tokens that match to the gold answer($N_{matched}$) divided by the number of tokens in the predicted answer($N_{predAns}$), and the recall equals to the number of tokens that match to the gold answer divided by the number of tokens in the gold answer($N_{goldAns}$). For the Chinese dialogue data, to avoid the error of automatic Chinese word segmentation, the F1 score is calculated in the character level. Note that punctuation is ignored in calculating the EM and F1 scores.

3 Models

We evaluate representative neural end-to-end models for response geenration, which share a similar backbone of encoder-decoder structure (with the exception of DialoGPT, which has only a decoder). Below we give the common structure (Section 3.1) and then introduce the characteristic of each model (Section 3.2).

3.1 Model Structure Overview

We first give an overview of the structures of representative dialogue models. Most existing models adopt an encoder-decoder structure. As shown in Figure 1, the models consist of an utterance encoder, a context encoder and a decoder. Given a

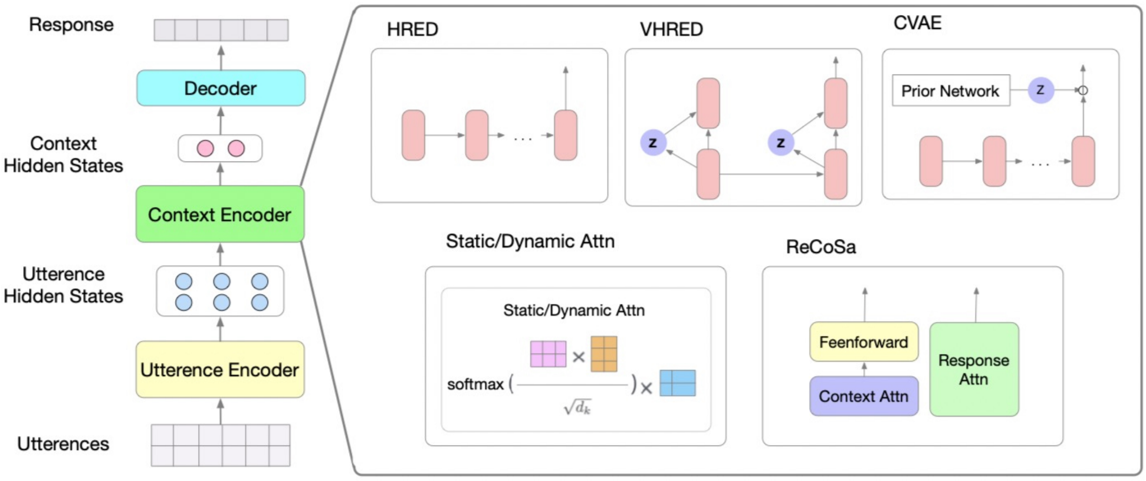

Figure 1: Model structure overview.

dialogue state, all the utterances in the current state, including past QA pairs, and a clarification request (question) are encoded as one input. The target is to generate the "gold" answer a_j from the input. The formalization of the process is:

$$y = \arg\max p(a_j|u_1, u_2, ..., q_i, \hat{a}_i, ..., u_k, q_j)$$

where $u_1, u_2, ..., u_k$ are dialogue utterance. q_i and $\hat{a}_i$ denote the i-th question and the predicted answer, respectively.

3.2 Representative Models

We choose the following 8 multi-turn dialogue generation models.

HRED: (Serban et al., 2016) is a hierarchical RNN-based encoder-decoder framework to sequentially model multi-turn dialogue and generate responses. It consists of two directional RNNs. One RNN is modeling the tokens in an utterance. The other RNN is modeling the utterances in a dialogue context.

vHRED: (Serban et al., 2017) is proposed to alleviate the generation of vague and generic response, which is caused by the gradient vanishing of HRED model, by introducing a hidden variable z. Therefore, vHRED is a variational enhanced HRED model.

CVAE: (Zhao et al., 2017) uses a prior network to model the gold response into a hidden variable z, which is as a condition in training step to improve the generation diversity.

Static/Dynamic Attention: the mechanisms (Zhang et al., 2018) alternatively model the contextual representations of multi-turn dialogue history using two types of attentions rather than using RNN.

ReCoSa: (Zhang et al., 2019) models the dialogue history in various granularity, e.g. context and response, using interactive attention and self-attention, respectively.

Transformer: (Vaswani et al., 2017) is used as a representative pretrained encoder-decoder model for dialogue generation.

DialoGPT: DialoGPT (Zhang et al., 2020) is a generative pretrained Transformer decoder for dialogue generation. To further conclude the characteristics of these models, Table 6 presents an overview of the characteristics of the chosen representative dialogue generation models in the proposed dialogue understanding task.

3.3 Implementation Details

For the training of the HRED, vHRED, CVAE and ReCoSa models, we use a bidirectional GRU (Cho et al., 2014) to encode the dialogue context and the input message. For the training of the static and dynamic attention models (Zhang et al., 2018), to be consistent to the setting in the original paper, a unidirectional GRU is utilized for contextual encoding of dialogue history. A fixed size of contextual window of dialogue utterances is used for modeling

	HRED	vHRED	CVAE	Static	Dynamic	ReCoSa	Transformer	DialoGPT
RNN	✓	✓	✓	✓	✓	✓	×	×
Attention	×	×	×	✓	✓	✓	✓	✓
Self-Attention	×	×	×	×	×	✓	✓	✓
Hidden variable	×	✓	✓	×	×	×	×	×
Encoder	✓	✓	✓	✓	✓	✓	✓	×
Decoder	✓	✓	✓	✓	✓	✓	✓	✓

Table 6: Characteristics of the representative dialogue models.

Transformer+Static	Transformer+Dynamic
$e_i = \frac{qk_i^T}{\sqrt{d_k}}$	$e_{i,t} = \frac{q_t k_i^T}{\sqrt{d_k}}$
$\alpha_i = \frac{\exp e_i}{\sum_j \exp e_j}$	$\alpha_{i,t} = \frac{\exp e_{i,t}}{\sum_j \exp e_{j,t}}$
O_i= MultiHead(q, k_i, k_i)	$O_{i,t}$= MultiHead(q_t, k_i, k_i)
$c = \sum_i \alpha_i O_i$	$c_t = \sum_i \alpha_{i,t} O_{i,t}$

Table 7: Implementation details of integrating the static and dynamic attentions into Transformer-based dialogue model.

the dialogue history[4].

For adding the Static attention into the Transformer model, the query q is the representation of a question. For the integration of Dynamic attention into the Transformer model, the query q_t denotes the decoded answer fragment in time step t. The key k_i denotes the output of encoding the i-th utterance in the dialogue context. The detailed modeling process is shown in Table 7. Please refer (Vaswani et al., 2017) for the definitions of q,k and d.

The dimension of the word and character embedding, which are initialized with GloVe[5], equals to 300. The RNN model is implemented with GRU. The size of hidden variable in vHRED and CVAE models is 300. For the ReCoSa model, the number of attention head equals to 6 and the number of self-attention layers is 3. Dropout is used in all models. For the experiments on English dialogue data, we use the 840B version of GloVe embedding. In experiments of Chinese dialogue data, to avoid the impact of different Chinese word segmentation tools, we use the character-level GloVe embedding, which is trained on Chinese Weibo corpus (Shang et al., 2015). Noted that the character embedding is fixed in the training process of these dialogue generation models.

4 Results

Table 8 shows the results of the representative models on the proposed DEQA dataset. Overall, the model performances are below 20% in EM and below 40% in F1, which shows that the task is challenging for existing dialogue models. The results are relatively low compared to the same English benchmark on response generation task(Feng et al., 2020), which are in the range of 0.594 to 0.728 in averaged greedy matching and 0.548 to 0.746 in frequency-based similarity. The averaged greedy matching and frequency-based similarity are used to evaluate the coherence and informativeness of a generated response, respectively (Feng et al., 2020).

4.1 Results of Different Models

1) Attention-based models such as Static, Dynamic and ReCoSa, outperform the HRED, vHRED and CVAE models in EM score in both English and Chinese dialogue data and F1 score in Chinese data. It shows that attention/self-attention from an output token to the dialogue history context can be useful for capturing co-reference information.

2) Comparing the results in Table 8, the pretraining models outperform the representative dialogue models on both English and Chinese dialogue data, which demonstrates the superiority of the pretraining scheme on dialogue understanding. The results are consistent with results on the Winograd Scheme challenge (Ruan et al., 2019; Sakaguchi et al., 2020), which demonstrate that pretraining can be useful for co-reference resolution.

3) The CVAE model gives the best F1 score in Chinese dialogue data. Comparing the results of HRED and vHRED, we find that the use of latent variable may not improve the performances on the proposed task. Comparing the results of vHRED and CVAE, we can speculate that the improvements of performance may be from the introducing of prior information in context encoder.

4) The integration of static attention into Transformer model can further improve the performance

---

[4] Note that we also verified the performance of non-fixed window size (e.g., a sliding window) but the performance of the above models all decrease.

[5] `https://nlp.stanford.edu/projects/glove/`

Model	English		Chinese	
	EM	F1	EM	F1
HRED	6.048	13.642	3.762	8.161
vHRED	5.882	14.954	2.871	7.239
CVAE	4.636	17.094	5.509	16.893
Static	6.048	18.132	7.052	14.873
Dynamic	6.214	17.602	6.856	14.408
ReCoSa	6.016	13.390	6.485	14.604
Transformer	9.959	23.221	17.723	33.306
+Static	10.133	24.155	**21.980**	**40.024**
+Dynamic	8.375	23.153	15.941	31.492
DialoGPT	**16.560**	**37.140**	19.307	33.798

Table 8: Results of the representative dialogue models on the proposed task.

on Chinese dialogue data. In addition, comparing the results of "Transformer" and "Transformer+Static", it indicates that the fine-grained encoding process can further improve the performance of the Transformer-based dialogue model.

4.2 Results on Different Answer Types

To further understand the main challenges, we split the test set into four subsets according to the answer types. Table 9 and 10 show the results of the models in English and Chinese dialogue data, respectively. Overall, the performance trends of DialoGPT and Transformer with static attention are consistent to the results in Table 8, which show the strong capability of pretraining models on the proposed task. Comparing each type of answers, we find that the difficulty of generating the answers in types of entity, phrase, clause and fragment is increasing in both English and Chinese dialogue data. One common reason is that the generation quality declines with the increasing of text length (Liu et al., 2018; Tan et al., 2020)[6]. In addition, the F1 score in English dialogue data is not monotonically decreasing as EM score. It is because the average number of tokens in English entities is close to 1, which leads to a lower F1 score than that in English phrases. It also reveals the reason that the EM and F1 scores in Entity type are closer than that in Phrase type.

5 Related Work

Conversational QA Recent research on conversational question answering (**ConvQA**) had been driven by two challenges, namely CoQA (Reddy et al., 2018) and QuAC (Choi et al., 2018). Rather than understanding the meaning of a given passage/document through the form of conversational question answering, the proposed task focuses on measuring the capability of understanding the dialogue itself. Besides the two challenges, several conversational machine reading/comprehension datasets were proposed (Elgohary et al., 2018; Dinan et al., 2018; Huang et al., 2018; Saeidi et al., 2018). The most common characteristic of these datasets are that their questions are open-domain and sequentially (or contextually) related, which shows a recent recognition in the research community that understanding the semantics of a complete conversation, including historical question and answer contexts, is crucial for these tasks. Our work is similar in spirit, but concentrating on clarification requests.

Clarification Request in Dialogue Clarification requests (**CR**) in dialogue are mainly motivated by acoustic understanding and semantic understanding (Schlangen, 2004; Stoyanchev and Johnston, 2015). They are used mainly as a way to establish mutual knowledge or grounding in communication (Gabsdil, 2003; Rieser and Moore, 2005). Purver et al. (2003) proposed to classify the forms of clarification requests into 8 categories, including non-reprise clarifications, reprise sentences, reprise sluices, reprise fragments, gaps, gap fillers, conventional and other. Rodríguez and Schlangen (2004) further summarized the surface forms, intonations and functions of clarification requests in spoken dialogue systems. Ginzburg (2016) detailed the semantics of dialogue and the fundamental problems to tackle for the semantic analysis in dialogue. In their work, a clarification request is defined to be a core function for dialogue systems to maintain the coherence of a dialogue.

This line of work coincides with our motivation that asking questions for clarification is a natural way to help understanding the meaning and maintaining coherence in dialogues. Therefore, the abilities of generating clarification requests to users and correctly responding to such requests from users are crucial to dialogue systems. Different from the above work, we build the DEQA, a Dialogue Entity via Question Answering dataset and investigate computational models for measuring the ability of machines on understanding the semantics of a dialogue via question answering.

---

[6] The average numbers of tokens in the answer types of entity, phrase, clause and fragment are 1.22, 2.15, 9.07, 5.48 in English dialogue data and 1.95, 3.28, 6.86 and 5.77 characters in Chinese dialogue data.

English	Entity		Phrase		Clause		Fragment	
Model	EM	F1	EM	F1	EM	F1	EM	F1
HRED	7.707	9.985	6.987	19.819	8.333	17.384	3.579	14.624
vHRED	7.900	9.857	5.677	20.379	8.333	20.460	3.579	17.944
CVAE	5.545	12.439	4.933	22.246	8.333	20.241	2.004	21.298
Static	7.514	11.258	8.297	26.243	8.333	19.136	3.132	21.931
Dynamic	7.514	11.534	8.734	23.584	8.333	26.906	3.356	21.334
ReCoSa	0.000	8.853	0.901	23.589	0.000	22.651	0.000	22.702
Transformer	14.451	21.672	10.917	29.690	8.333	18.110	4.251	21.845
+static	15.414	22.240	12.227	31.014	8.333	24.934	2.908	22.823
+dynamic	13.295	18.822	8.297	33.305	8.333	**27.531**	2.685	22.811
DialogGPT	**24.085**	**39.194**	**20.961**	**44.281**	0.000	24.373	**6.040**	**31.480**

Table 9: Results of different answer types in English dialogue data.

Chinese	Entity		Phrase		Clause		Fragment	
Model	EM	F1	EM	F1	EM	F1	EM	F1
HRED	5.609	9.658	2.970	7.638	1.282	4.684	0.000	4.931
vHRED	4.062	8.524	1.980	6.883	0.000	4.075	0.000	5.512
CVAE	8.952	19.860	3.902	15.747	1.282	13.019	0.658	11.575
Static	11.069	18.438	2.765	12.103	3.846	8.152	2.631	10.154
Dynamic	11.257	16.510	1.382	11.138	3.846	13.041	2.631	12.810
ReCoSa	2.421	4.945	0.980	4.685	0.000	3.726	0.000	4.064
Transformer	26.095	36.928	14.634	33.508	5.128	25.523	1.974	21.619
+static	**32.381**	**44.447**	**15.122**	**37.753**	7.692	**34.885**	**6.579**	**29.846**
+dynamic	24.000	34.903	11.707	31.280	6.410	26.449	0.658	22.635
DialogGPT	31.048	38.554	10.244	34.399	**8.974**	27.660	1.974	25.654

Table 10: Results of different answer types in Chinese dialogue data.

6 Conclusion

We proposed a novel evaluation task for co-reference resolution in dialogue understanding with a new benchmark dataset, DEQA. By asking a dialogue model to identify entity-oriented linguistic structures in dialogue history context, it directly measures the quality of dialogue understanding through response generation. Empirical comparisons show that the chosen representative dialogue models face challenges on the proposed benchmark dataset and clause and fragment types of co-references are paticularly challenging even for pretrained models. We will release the dataset for research use and further annotate the dataset with the questions that are related to more complex linguistic structures in future work.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (No. 62076081, No. 61772153 and No. 61936010) and Science and Technology Innovation 2030 Major Project of China (No. 2020AAA0108605).

Ethics Statement

Our research is only oriented to semantic tasks and does not rely on any user information. To protect the privacy, intellectual property rights, and the rights of annotators, we took the following operations:

Privacy As we used crawler technology to get some raw data from Douban. It should be noted that to ensure the privacy of users, we directly deleted all User IDs and pictures during the crawling process, and only retained the text of the conversation. Then, we manually verify that all data has not leaked privacy.

Intellectual property rights Redistributing Douban's data may violate intellectual property rights. So we are now applying to Douban for permission to redistribute the data. We will release the dataset as soon as possible after obtaining permission. Before that, we first release the data of Dailydialog on the Github page and indicate that we are applying for the permission of Douban data.

Reasonable Compensation Regarding salary, Chinese annotation is easier than English for our Chinese annotators. Each piece of Douban data is paid about 1 yuan(about 16 cents), and DailyDialogue annotators are paid about 2 yuan (about 31 cents) per piece of data. Then we estimated the hourly salary of the annotator. The Chinese dataset annotator who is new to the task can complete about 70 annotations in an hour on average, so their hourly salary is at least 70 yuan(about 11 dollars) per hour.

The English dataset annotator completes an average of about 40 annotations per hour. The hourly salary is 80 yuan (about 12.5 dollars)per hour.

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Siqi Bao, Huang He, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, Zhen Guo, Zhibin Liu, and Xinchao Xu. 2020. Plato-2: Towards building an open-domain chatbot via curriculum learning. *arXiv preprint arXiv:2006.16779*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. SuperAgent: A customer service chatbot for E-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102, Vancouver, Canada. Association for Computational Linguistics.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. A dataset and baselines for sequential open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1077–1083.

Shaoxiong Feng, Hongshen Chen, Kan Li, and Dawei Yin. 2020. Posterior-gan: Towards informative and coherent response generation with posterior generative adversarial network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7708–7715.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale qa-srl parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2051–2060.

Malte Gabsdil. 2003. Clarification in spoken dialogue systems. In *Proceedings of the 2003 AAAI Spring Symposium. Workshop on Natural Language Generation in Spoken and Written Dialogue*, pages 28–35.

Jonathan Ginzburg. 2016. The Semantics of dialogue. In *The Cambridge Handbook of Formal Semantics*.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2018. Flowqa: Grasping flow in history for conversational machine comprehension. *arXiv preprint arXiv:1810.06683*.

Fang Kong, Min Zhang, and Guodong Zhou. 2019. Chinese zero pronoun resolution: A chain-to-chain approach. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(1):1–21.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *ICLR*.

Matthew Purver, Jonathan Ginzburg, and Patrick Healey. 2003. On the means for clarification in dialogue. In *Current and new directions in discourse and dialogue*, pages 235–255. Springer.

Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. 2019. Gecor: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4539–4549.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Verena Rieser and Johanna D Moore. 2005. Implications for generating clarification requests in task-oriented dialogues. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 239–246. Association for Computational Linguistics.

Kepa Joseba Rodríguez and David Schlangen. 2004. Form, intonation and function of clarification requests in german task-oriented spoken dialogues. In *Proceedings of Catalog (the 8th workshop on the semantics and pragmatics of dialogue; SemDial04)*.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality qa-srl annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Yu-Ping Ruan, Xiaodan Zhu, Zhen-Hua Ling, Zhan Shi, Quan Liu, and Si Wei. 2019. Exploring unsupervised pretraining and sentence structure modelling for winograd schema challenge. *arXiv preprint arXiv:1904.09705*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. *arXiv preprint arXiv:1809.01494*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

David Schlangen. 2004. Causes and strategies for requesting clarification in dialogue. In *Proceedings of the 5th Workshop of the ACL SIG on Discourse and Dialogue*.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586.

Svetlana Stoyanchev and Michael Johnston. 2015. Localized error detection for targeted clarification in a virtual assistant. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5241–5245. IEEE.

Bowen Tan, Zichao Yang, Maruan AI-Shedivat, Eric P Xing, and Zhiting Hu. 2020. Progressive generation of long text. *arXiv preprint arXiv:2006.15720*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017. Chinese zero pronoun resolution with deep memory network. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1309–1318.

Hainan Zhang, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. 2019. Recosa: Detecting the relevant contexts with self-attention for multi-turn dialogue generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3721–3730.

Weinan Zhang, Yiming Cui, Yifa Wang, Qingfu Zhu, Lingzhi Li, Lianqiang Zhou, and Ting Liu. 2018. Context-sensitive generation of open-domain conversational responses. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2437–2447.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2018. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, (Just Accepted):1–62.

Beyond Metadata: What Paper Authors Say About Corpora They Use

Nikolay Kolyada[1] **Martin Potthast**[2] **Benno Stein**[1]
[1] Bauhaus-Universität Weimar, Germany, `<first>.<last>@uni-weimar.de`
[2] Leipzig University, Germany, `martin.potthast@uni-leipzig.de`

Abstract

The growing ecosystem of data sharing in science has put dataset search into the focus. To make data sharing and reuse more feasible, new retrieval tools and services are being developed. Currently, dataset retrieval relies almost exclusively on metadata provided by the publishers. To extend this knowledge source our work studies the task of "dataset review mining" in scientific publications. For the field of Natural Language Processing we collect metadata about datasets from established resources such as the ELRA and LDC catalogs, and then extract review statements about the datasets from ACL Anthology Corpus publications, compiling the Webis-Dataset-Reviews-21 corpus. By analyzing the reviews we identify different categories of what paper authors write about data. To the best of our knowledge, this is the first analysis of this kind in the field of Natural Language Processing, albeit similar analyses have been carried out in the social and medical sciences. Our corpus and the underlying code are shared alongside this paper.[1,2,3]

1 Introduction

Recently, Google introduced its Dataset Search service (Brickley et al., 2019).[4] Around 30 million datasets have been indexed to date,[5] hosted at web pages and repositories all across the web. Although data sharing and dataset search has long been a best practice in natural language processing, Google's service has accelerated and standardized data sharing across scientific communities, since only those datasets are indexed, for which a certain metadata description prescribed by Google is provided.

Chapman et al. (2020) and Koesten et al. (2017) survey the state of the art in dataset search and analyze how it differs from classical information retrieval. Approaches to dataset search from related fields, such as databases, semantic web, entity-based search, and tabular search are reviewed, but neither solves the task comprehensively. From a user's perspective, searching for datasets entails two steps: (1) its retrieval using standard keyword search, and (2) examining the search results to identify the datasets suitable to one's needs. The second step strongly differs from web search, since a (large) dataset cannot be easily reviewed in the browser; merely it's metadata is available—as well as the descriptions found on the download page. We argue that a crucial piece of information is missing to inform the examination of dataset search results: dataset user experience.

Unlike for commercial products in online shops, for datasets, there are hardly any platforms where users share their experiences using them. On the few that do exist, the amount of published reviews cannot be compared with that of commercial products. In any case, due to the specificity of most datasets, few people actually use them. Their feedback is typically found only as part of their reports and scientific publications. However, here, authors are at liberty to provide testimonials and criticisms that can be more thorough and extended in comparison to commercial product reviews.

To assess this potential source of information about datasets, and to eventually harness it for dataset search, this paper takes the first step in this direction by compiling and analyzing Webis-Dataset-Reviews-21, a corpus of dataset reviews from a large corpus of scientific publications, namely the ACL Anthology. Compiling a list of datasets in the field of Natural Language Processing and Computational Linguistics using authoritative sources including catalogs of the European

---

[1] https://github.com/webis-de/ACL-21

[2] https://webis.de/data.html#Webis-Dataset-Reviews-21

[3] https://doi.org/10.5281/zenodo.4889032

[4] https://datasetsearch.research.google.com/

[5] https://ai.googleblog.com/2020/08/an-analysis-of-online-datasets-using.html

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5085–5090
August 1–6, 2021.

Language Resources Association (ELRA) and its LRE Map, the Linguistic Data Consortium (LDC), and others, we extract from the ACL Anthology's 57,608 papers 466,567 sentences mentioning a dataset. Exploring these mention statements, we organize them in a taxonomy, assessing their usefulness for dataset retrieval.

2 Related Work

Dataset review mining can be seen as part of the task of dataset retrieval, as well as that of extracting information from scientific documents. The demand for dataset search has increased with more data being published across scientific communities, by industry, and by government bodies. Multiple initiatives facilitate data sharing and encourage scientists to do so, such as Zenodo,[6] and DataCite[7] (Rueda et al., 2016), which also provide digital object identifiers (DOIs) for datasets, and a platform for publishing them. Several open software frameworks implement data collection management. Currently, most common are CKAN (Solr), Socrata, and OpenDataSoft. Chapman et al. (2020) show that most of the search features implemented rely exclusively on metadata. Brickley et al. (2019) observe that this restricts the effectiveness of dataset retrieval engines as metadata quality varies significantly. Notwithstanding this shortcoming, specialized dataset search engines still dramatically improve dataset discovery and sharing.

Several commonly used schemas and formats for dataset metadata exist, such as the JavaScript Object Notation for Linked Data (JSON-LD), and the Dataset schema[8] based on W3C DCAT (Erickson et al., 2013). The latter has become a de-facto standard, as it is used by Google Dataset Search to index data collections found in the open web. Many data publishers meanwhile adopted these standards of making metadata about their published datasets machine-readable, and data portal software platforms, such as CKAN, integrate this functionality by default. In addition, improvements on tracking citations of data are being actively developed. Projects, like Semantic Scholar, Microsoft Academic Graph,[9] and Google Dataset Search, along with Google Scholar, couple data and publications to improve data discovery.

The task of extracting structured information from scientific publications has been tackled many times. Gupta and Manning (2011) extract key aspects of scientific papers, including focus, technique, and domain from the ACL Anthology. Mesbah et al. (2018), Luan et al. (2018), and Jain et al. (2020) propose approaches to identify entities and their relations in scientific documents. Gábor et al. (2016) creates an annotated corpus for concepts and semantic relations based on the ACL Anthology. Duck et al. (2016) employed text mining to process dataset and software mentions in biological and medical publications from PubMed Central. Boland et al. (2012) identify references to datasets in social science publications. Closely related to our work, *software* mentions in scientific documents can be mined using a Grobid library module,[10] e.g., to give research software more credit.

In 2019, a shared task focusing on the tasks of dataset review mining and extraction of scientific methods and fields was organized by the Coleridge Initiative.[11] The training data provided was based on the Inter-university Consortium for Political and Social Research (ICPSR) data catalog,[12] comprising around 10,000 datasets used in the social sciences and a labeled corpus of 5,000 publications matched with mentioned datasets. The best-performing solution by The Allen Institute for Artificial Intelligence (AI2) implements a set of rule-based candidate citations by exact string matching mentions and datasets as a first step. An ad-hoc named entity recognition model has been trained based on matches identified. Mention labels were generated by string matching mentions in the provided annotations against the full text of papers. Then, candidate datasets were scored based on TF-IDF-weighted token overlap between the mention text and the dataset title, linking them using a binary classifier.

Färber et al.'s (2021) work is perhaps the most closely related one to our paper. It recognizes datasets along with scientific methods in a corpus of 510,000 publications. The TSE-NER approach by Mesbah et al. (2018) was used to identify methods and datasets in order to integrate this data into the Microsoft Academic Knowledge Graph (MAKG). Moreover, identified dataset references has been classified into used vs. not-used based on the textual context.

---

[6] https://zenodo.org/

[7] https://datacite.org/

[8] http://schema.org/Dataset

[9] This project has been discontinued: https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

[10] https://github.com/ourresearch/software-mentions

[11] https://coleridgeinitiative.org/richcontextcompetition/

[12] https://www.icpsr.umich.edu/icpsrweb/

3 Collecting the Metdata of NLP Datasets

The two major steps to construct the Webis-Dataset-Reviews-21 corpus are the compilation of a collection of metadata of datasets used in the field of Natural Language Processing, and the extraction of mentions of these datasets from the ACL Anthology. To tackle the first step, we crawl authoritative NLP dataset catalogs, collect their metadata, clean and normalize it, merge duplicates, and format this compilation according to the specification of https://schema.org/Dataset.

The catalogs crawled are shown in Table 1. Although these catalogs may not list all existing datasets in Natural Language Processing, they certainly contain many of the most prominent and the most commonly used ones in the field. Nevertheless, there are bound to be many datasets in the "long tail": Their identification will require tailored named entity recognition approaches along the lines of the ones employed in aforementioned shared tasks. Our current collection of dataset metadata contains 13,372 entries in total.

The metadata enclosed in the catalogs are based on different schemas. In order to render them amenable for subsequent processing, we went about cleaning, normalizing, and unifying them across catalogs. Duplicates were identified and merged into a normalized schema: For each dataset, we collect properties relating to its source, original title, acronym, DOI, description, year, creator, and URL, as well as additional properties, such as language, format, size, and metadata about the associated paper, if available. Our final, unified metadata catalog of NLP datasets forms part of the Webis-Dataset-Reviews-21 corpus.

Catalog	**Datasets**
Language Resources monitoring (LRE Map)[13]	6,143
European Language Association (ELRA)[14]	5,398
Linguistic Data Consortium (LDC)[15]	950
Big Bad NLP Database[16]	791
NLP Progress[17]	90
Σ	13,372

Table 1: NLP resources crawled for dataset metadata.

4 Mining Dataset Reviews

The second step of constructing our corpus was mining for dataset reviews from the ACL Anthology in the form of sentences that mention a dataset found in our unified catalog. We extract dataset mentions from the anthology's papers, annotate them, and develop a taxonomy with regard to the information a dataset mention provides about the data. We further analyze common patterns used by authors to describe the data.

4.1 Preprocessing the ACL Anthology

The ACL Anthology[18] compiles all papers published in the fields of computer linguistics and natural language processing published between the late 1960s up to today and makes them available open access. We used a collection of papers from that includes the papers up until 2020, 57,606 PDF files in total, omitting ones not written in English.

To enable text processing the papers, we use the Grobid (GRO, 2008–2021) library for parsing scientific publications given as PDF and extracting structured XML/TEI-encoded documents.[19] In the subsequent review mining step, we decided to analze only the publications' body sections (including the introduction), as a pilot study showed datasets mentioned in the abstract or references sections hardly contain any information about them. Harnessing the annotations provided by Grobid, we resolve references and footnotes within the text of a paper and add the titles of cited items to the document full text. If a paper contains exact references in a form of DOI or URL to the corpora used, we consider this information as additional features in the mention extraction step. Information about paper sections and paragraphs within the body are preserved as well.

4.2 Dataset Mentions Extraction

For the dataset mention extraction, we exploit the fact that most datasets in natural language processing research have proper names and/or distinct acronyms. This is unlike in the social sciences, where many datasets rather have descriptions for names (e.g., "national census data on population development 2020", which then gets shortened adhoc by the authors while writing their paper). By contrast, for NLP dataset, their creators often conceive of catchy acronyms (e.g., "WordNet") that

---

[13] http://lremap.elra.info/

[14] http://www.elra.info/

[15] https://www.ldc.upenn.edu/

[16] https://datasets.quantumstat.com/

[17] http://nlpprogress.com/

[18] https://www.aclweb.org/anthology/

[19] https://tei-c.org/

Characteristic	**Number of elements**
Mentions	466,567
Coreference mentions	93,176
Publications with at least one mention	(92%) 53,129
Unique datasets mentioned	(22%) 2,986

Table 2: Mentions of datasets in the ACL Anthology.

Dataset	**Mentions**	**Coreferences**	**Publications**
WordNet	**43,746**	**8,925**	6,591
Wikipedia	36,954	8,682	6,602
Penn Treebank	20,453	5,661	**8,039**
FrameNet	9,236	2,047	1,386
Brown	8,666	2,317	2,752
PropBank	5,366	1,202	1,032
Europarl	5,075	1,477	1,624
BNC	4,607	1,403	895
PDTB	4,232	967	399
Freebase	3,799	925	759
Gigaword	3,792	1,046	1,720
UMLS	3,705	783	687
VerbNet	3,372	636	592
Wiktionary	3,156	680	553
DUC	3.022	656	474
OntoNotes	2,973	821	782
SQuAD	2,872	764	432
ATIS	2,737	750	483
DBpedia	2,625	642	576
TimeML	2,584	614	384
PubMed	2,437	585	648

Table 3: Number of mentions and papers for top 20 datasets mentioned in the ACL Anthology publications.

are then picked up by paper authors. Therefore, we get by with basic string matching to extract 466,567 mentions of 2,986 unique datasets out of the 13,371 ones in our catalog.[20] At the same time, with this basic approach, we introduce a baseline for the task of dataset mention extraction, which, in practice, has a similar recall compared to that of a full-text search performed over an indexed collection of papers.

We define a dataset review as "an assessment from the person who used the data" (be it praise or criticism). Reviews are often spread across sentences around a dataset mention, but are usually confined to within the same paragraph or section. We employ a co-reference resolution model (Clark and Manning, 2016) to extract further discussion of a dataset from adjacent sentences around a dataset mention. Tables 2 and 3 overview relevant corpus statistics.

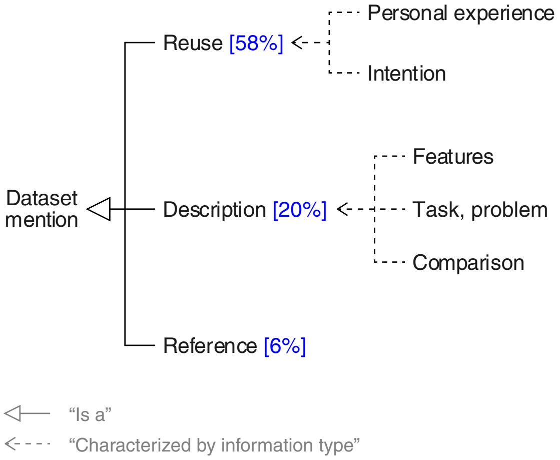

Figure 1: Taxonomy of dataset mentions. The type "personal experience" corrsponds to a dataset review, and implies reuse of the data.

4.3 A Taxonomy of Dataset Mentions

We manually reviewed a random sample of 1,000 dataset mentions and derived a taxonomy of three classes as shown in Figure 1: reuse, description, and reference. We primarily distinguish between "active" cases of reuse of a dataset from "passive" descriptions and references. Statements of dataset reuse include cases where data is used as a basis for experiments, training models, and synthesis of new or sub-datasets. For instance:

(1) *"Second, we reuse the RCV1-V2, using a version that contained a selected 5,000 term vocabulary."*

Descriptions and references rather give details about a dataset or compare it:

(2) *"For instance, two words are said to be synonyms if they belong in the same synset in the WordNet."*;

(3) *"MovieQA is a challenging dataset for movie understanding. The dataset consists of 14,944 multiple choice questions about 408 movies."*

Altogether, 58% of the sample are reuse mentions, 20% descriptions, and 6% references. The remaining mentions are ambiguous and/or incorrect.

From the reuse mentions, 63% share a "personal experience" which we operationalize as a form of dataset review. Projected to the entire set of dataset mentions, this potentially results in about 150,000 such cases. For instance:

(4) *"When only WordNet, not BabelNet, is used for identifying lexico-semantic relations, per-*

---

[20] In future work, we plan on investigating if the recall of mentioned datasets can be increased with heuristic matching approaches as well as citation-based ones; cursory attempts, however, showed especially the former to introduce a lot of noisy mentions, which is why we omitted them in this analysis.

formance increases slightly, which we attribute to noise that comes with using BabelNet.";

(5) *"The TimeBank and the data used in the two TempEval challenges are important, as they have annotations describing not just dates and times, but also events and temporal relations between these entities."*;

(6) *"While AQuA may prove useful for training, it is inappropriate as an evaluation set."*

Unsurprisingly, commonly used datasets, such as Wikipedia, WordNet, the Penn Treebank, the Brown Corpus, and GigaWord receive the most mentions with over 5,000 each. The majority are mentioned only a few times; only 25% of the datasets are mentioned more than 10 times. Altogether, description mentions provide information about the data, its features, and common tasks it is used for. Reuse mentions conveying a personal experience provide information about suitability of the dataset for the paper's envisioned task, as well as details about the data.

5 Conclusion

We compiled a comprehensive list of NLP-related datasets and extracted their mentions from the ACL Anthology. Analyzing the mentions, three basic categories can be distinguished, namely reuse, descriptions, and references. The former two categories of mentions are considered to be useful as a source of information on a dataset in the task of dataset retrieval, enabling dataset search engines to display more in-depth information about a dataset on a search results page, as well as informing their retrieval models. Other future directions include improving the extraction of mentions, e.g., via "few-shot" domain-specific named entity recognition, as well as, expanding to other fields of research.

References

2008–2021. Grobid. https://github.com/kermitt2/grobid.

Katarina Boland, Dominique Ritze, Kai Eckert, and Brigitte Mathiak. 2012. Identifying references to datasets in publications. In *International Conference on Theory and Practice of Digital Libraries*, pages 150–161. Springer.

Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *The World Wide Web Conference*, pages 1365–1375.

Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *The VLDB Journal*, 29(1):251–272.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Geraint Duck, Goran Nenadic, Michele Filannino, Andy Brass, David L Robertson, and Robert Stevens. 2016. A survey of bioinformatics database and software usage through mining the literature. *PloS one*, 11(6):e0157989.

John S. Erickson, Amar Viswanathan, Joshua Shinavier, Yongmei Shi, and James A. Hendler. 2013. Open government data: A data analytics approach. *IEEE Intell. Syst.*, 28(5):19–23.

Michael Färber, Alexander Albers, and Felix Schüber. 2021. Identifying used methods and datasets in scientific publications.

Kata Gábor, Haïfa Zargayouna, Davide Buscaldi, Isabelle Tellier, and Thierry Charnois. 2016. Semantic annotation of the ACL anthology corpus for the automatic analysis of scientific literature. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

Sonal Gupta and Christopher D. Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1–9. The Association for Computer Linguistics.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. Scirex: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7506–7516. Association for Computational Linguistics.

Laura M Koesten, Emilia Kacprzak, Jenifer FA Tennison, and Elena Simperl. 2017. The trials and tribulations of working with structured data: -a study on information seeking behaviour. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 1277–1289.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pages 3219–3232. Association for Computational Linguistics.

Sepideh Mesbah, Christoph Lofi, Manuel Valle Torre, Alessandro Bozzon, and Geert-Jan Houben. 2018. TSE-NER: an iterative approach for long-tail entity extraction in scientific publications. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 127–143. Springer.

Laura Rueda, Martin Fenner, and Patricia Cruse. 2016. Datacite: Lessons learned on persistent identifiers for research data. *Int. J. Digit. Curation*, 11(2):39–47.

Knowledge Distillation for Quality Estimation

Amit Gajbhiye,[1] Marina Fomicheva,[1] Fernando Alva-Manchego,[1] Frédéric Blain,[1,2]
Abiola Obamuyide,[1] Nikolaos Aletras,[1] Lucia Specia[1,3]
[1]University of Sheffield, [2]University of Wolverhampton, [3]Imperial College London
`{a.gajbhiye,m.fomicheva,f.alva,a.obamuyide,n.aletras}@shef.ac.uk`
`f.blain@wlv.ac.uk`
`l.specia@imperial.ac.uk`

Abstract

Quality Estimation (QE) is the task of automatically predicting Machine Translation quality in the absence of reference translations, making it applicable in real-time settings, such as translating online social media conversations. Recent success in QE stems from the use of multilingual pre-trained representations, where very large models lead to impressive results. However, the inference time, disk and memory requirements of such models do not allow for wide usage in the real world. Models trained on distilled pre-trained representations remain prohibitively large for many usage scenarios. We instead propose to directly transfer knowledge from a strong QE teacher model to a much smaller model with a different, shallower architecture. We show that this approach, in combination with data augmentation, leads to light-weight QE models that perform competitively with distilled pre-trained representations with 8x fewer parameters.

1 Introduction

Quality Estimation (QE) aims to predict the quality of the output of Machine Translation (MT) systems when no gold-standard translations are available. It can make MT useful in real-world applications by informing end-users on the translation quality. We focus on sentence-level QE, usually formulated as a regression task where quality is required to be predicted on an continuous scale, e.g. 0-100.

The high performances achieved in the most recent shared task on sentence-level QE (Specia et al., 2020) have been attributed to the use of strong pre-trained language models, namely BERT (Devlin et al., 2018) and its multilingual variants, especially XLM-Roberta (Conneau et al., 2020a). These models have an extremely large number of parameters and, since they are required at training and inference time, they are very disk and RAM-hungry, also making inference slow. This poses challenges for real-time inference, and prohibits deployment on client machines with limited resources.

Making models based on pre-trained representations smaller and more usable in practice is an active area of research. One approach is Knowledge Distillation (KD), aiming to extract knowledge from a top-performing large model (the *teacher*) into a smaller (in terms of memory print, computational power and prediction latency) yet well-performing model (the *student*) (Hinton et al., 2015; Gou et al., 2020). KD techniques have been used to make BERT and similar models smaller. For example, DistilBERT (Sanh et al., 2019) and TinyBERT (Jiao et al., 2020) follow the same general architecture as the teacher BERT, but with a reduced number of layers. However, these student models are also based on Transformers and, as such, they still have too large memory and disk footprints. For instance, the number of parameters in the multilingual DistilBERT-based TransQuest model for QE (Ranasinghe et al., 2020) is 135M.

In this paper, we propose **to distill the QE model directly**, where the student architecture can be completely different from that of the teacher. Namely, we distill a large and powerful QE model based on XLM-Roberta into a small RNN-based model. Existing work along these lines has applied KD mainly to classification tasks (Tang et al., 2019; Sun et al., 2019). We instead explore this approach in the context of **regression**. In contrast to classification, where KD provides useful information on the output distribution of incorrect classes, for regression the teacher predictions are point-based estimates, and as such have the same properties as gold labels. Therefore, it is not obvious whether teacher-student learning can be beneficial. The few existing works on KD for regression (Chen et al., 2017; Takamoto et al., 2020) use the teacher loss to minimise the impact of noise in the teacher

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5091–5099
August 1–6, 2021.

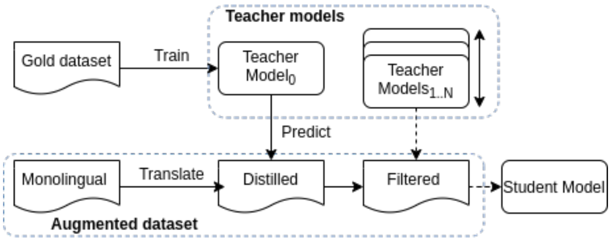

Figure 1: KD with data augmentation and noise filtering based on teacher uncertainty.

		Inference		
Name	#params	Speed (secs.)	RAM (MiB)	Disk (M)
TQ$_{\text{XLM-R-Large}}$	561M	0.82	9,263.5	2140
TQ$_{\text{DistilBERT}}$	135M	1.09	1,979.2	517
BiRNN	18M	0.39	155.6	132

Table 1: Efficiency. Inference speed and RAM for prediction are for 1 sentence on CPU (Intel Xeon Silver 4114 CPU @ 2.20GHz).

predictions on the student training. However, this approach requires access to gold labelled examples to train the student, which in our case are very limited in number.

Our approach allows for much larger **unlabelled student training datasets**, built only from source-MT pairs and labelled by the teacher model. We study the performance of student models under different training data regimes: standard training with gold labels, training with teacher predictions on the same data, training with teacher predictions on augmented in-domain and out-of-domain data, as well as augmented data filtered based on uncertainty of teacher predictions. Interestingly, we find that (i) training with teacher predictions results in better performance than training with gold labels; and (ii) student models trained with augmented data perform competitively with DistilBERT-based TransQuest predictors with 8x fewer parameters.

2 Approach

Figure 1 summarises our approach, with the following main components:

Teacher-student training. We use predictions from a SoTA QE model to train a light-weight student with a different architecture. Specifically, as the teacher model we use the recently proposed TransQuest QE system (Ranasinghe et al., 2020) that fine-tunes multilingual pre-trained representations from XLM-Roberta-Large (Conneau et al., 2020a) to predict a continuous sentence-level quality score. For the student model, we rely on the BiRNN QE architecture proposed by Ive et al. (2018).[1] The BiRNN model encodes both source and translation sentences independently using two bi-directional Recurrent Neural Networks (RNNs). The two resulting sentence representations are concatenated as the weighted sum of their word vectors, generated by an attention mechanism. For predictions at sentence-level, the weighted representation of the two input sentences is passed through a dense layer with sigmoid activation to generate the quality estimates. Table 1 shows the number of parameters, memory and disk space requirements, as well as inference speed for the teacher model (TQ$_{\text{XLM-R-Large}}$), student model (BiRNN) and TransQuest system built on DistilBERT (TQ$_{\text{DistilBERT}}$). We refer the reader to Appendix A for the details on the architecture and implementation for these models.

In classification tasks, KD benefits learning as it uses information on the output distribution and has an effect akin to label smoothing (Tang et al., 2020). In regression, teacher labels are instead point-wise estimates just like the gold labels. Existing work on KD for regression uses teacher loss to minimise the impact of noise in the teacher predictions on student training (Takamoto et al., 2020). However, this approach is not suitable for QE as we have access to a very limited number of gold-labelled examples. We propose a simple strategy that relies directly on teacher predictions for training the student model.

Data augmentation. The power of QE models based on pre-trained representations is due to the rich knowledge that comes from training Transformer-based language models on very large amounts of data. Typically, much smaller datasets are available for downstream tasks, which suffice for fine-tuning but that are hardly suitable for training a neural model from scratch. We exploit the teacher-student framework to produce additional training data. Specifically, we first generate MT outputs for a set of sentences in the source language and domain of interest using the same MT system that was used for generating the test data. Second, we use the teacher model described above to pro-

---

[1] The implementation of our student models is available at `https://github.com/sheffieldnlp/deepQuest-py`.

duce predictions. These predictions are then used as labels for training the student.

Noise filtering. The benefits of data augmentation can be hampered by noise in teacher predictions. In a classification setting, where the student loss is computed with respect to the output distribution of the teacher model, this issue is ameliorated by the example re-weighting effect where teacher predictions with higher confidence have an overall higher impact on learning (Furlanello et al., 2018). Previous work has used teacher loss to address this issue for regression (Chen et al., 2017). However, this strategy is not suitable for data augmentation as it requires both gold labels and teacher predictions.

As an alternative, we propose a mechanism to filter-out noisy examples in the augmented dataset based on uncertainty quantification. Recent work has shown that ensembles produce accurate uncertainty estimates (Lakshminarayanan et al., 2017). We exploit this idea by training a set of additional teacher models independently on the same training data using random initialisation, and using the variance of their predictions as an indicator of predictive uncertainty.[2] Intuitively, examples with very high variance would correspond to noisy teacher predictions. We filter out from the student training data the instances where the variance is more than one standard deviation away from its mean value. This is expected to have a higher impact on the results in the out-of-domain setting where the performance of the teacher model is less stable and teacher predictions can contain more noise.

3 Experiments

MLQE Dataset. For training the teacher and for evaluation, we use the MLQE dataset (Fomicheva et al., 2020), same as in the WMT2020 QE Shared Task (Specia et al., 2020). This dataset contains sentences extracted from Wikipedia translated to and from English for a total of six language pairs: English–German (En-De),[3] English–Chinese (En-Zh), Romanian–English (Ro-En), Estonian–English (Et-En), Sinhala–English (Si-En) and Nepali–English (Ne-En). Each translation was produced with a SoTA Transformer-based NMT model and manually annotated for quality using an annotation scheme inspired by the Direct Assessment methodology (Graham et al., 2013). The scores are produced on a continuous scale indicating perceived translation quality in 0-100. For each language pair, this dataset contains partitions for training (7K), dev (1K), and test (1K).

Language	Sentences
Estonian	25,176
Romanian	372,690
Sinhala	139,406
Nepalese	85,343
English	1,563,519

Table 2: Number of sentences extracted from Wikipedia for data augmentation.

Distilled dataset. Monolingual data for data augmentation was sampled from Wikipedia following the procedure described in Fomicheva et al. (2020) to preserve the domain of the MLQE dataset. Specifically, we sampled documents from Wikipedia for English, Estonian, Romanian, Sinhalese and Nepalese and selected the top 100 documents containing the largest number of sentences that are: (i) in the intended source language according to a language-id classifier and (ii) have the length between 50 and 150 characters. Table 2 shows the total amount of sentences in the monolingual Wikipedia dataset collected for data augmentation.

To test the impact of data domain on the performance of the student QE models, we also collect out-of-domain data for the Et-En language pair. The out-of-domain data is sampled from Common Crawl. We use the version of Common Crawl distributed by the WMT2018 News Translation Task[4]. The total amount of sentences in this dataset is 100,779,314.

To translate the data, we used the same MT models that generated the test data, built with fairseq (Ott et al., 2019) and made available by the WMT2020 QE Shared Task organisers.[5] Sentences that were part of the training data for the MT models or part of the MLQE dataset were excluded. We generate quality predictions for the remaining sentences using the teacher models, as

---

[2] Here we use ensemble only as a way of estimating the error in the predictions and leave distillation based on ensemble predictions to future work.

[3] We skip this language pair as the performance of the teacher model for it is too weak.

[4] `http://www.statmt.org/wmt18/translation-task.html`

[5] `https://github.com/facebookresearch/mlqe/tree/master/nmt_models`.

Name	Training data	Et-En	Ro-En	Si-En	Ne-En	En-Zh
TQ$_{\text{TEACHER}}$	MLQE-gold	0.77	0.88	0.60	0.75	0.44
BiRNN$_{\text{STUDENT}}$	MLQE-dist	0.45	0.62	0.44	0.46	**0.18**
BiRNN$_{\text{STUDENT+AUG}}$	Wiki-dist	**0.50**	**0.69**	**0.45**	**0.54**	0.17
BiRNN	MLQE-gold	0.37	0.60	0.40	0.42	0.15
Predictor-Estimator	MLQE-gold	0.48	0.69	0.37	0.39	0.19
TQ$_{\text{DistilBERT}}$	MLQE-gold	0.62	0.78	0.51	0.61	0.36

Table 3: Pearson correlation with human judgments on the MLQE test set. MLQE-gold: training partition of MLQE dataset; MLQE-dist: distilled version of the MLQE training set with teacher predictions used as labels; Wiki-dist: the Wikipedia dataset produced by data augmentation. Boldface results indicate our best student models.

described in Section 2. We used a random subset of 100K sentences from Wikipedia to train the student model for each of the language pairs except for Et-En where the total amount of collected in-domain monolingual data is 25K.

	Ro-En	Si-En	Ne-En	En-Zh
10K	0.56 ±.00	0.36 ±.00	0.41 ±.00	0.09 ±.01
50K	0.64 ±.00	0.45 ±.01	0.53 ±.00	0.20 ±.03
70K	0.66 ±.00	0.46 ±.01	0.54 ±.00	0.19 ±.02
100K	0.69 ±.00	0.47 ±.02	0.54 ±.00	0.17 ±.02

Table 4: Pearson correlation on the test partition of the MLQE dataset for BiRNN student models trained with different amounts of distilled Wikipedia data.

Models. As teachers, we use pre-trained models from TransQuest (TQ$_{\text{TEACHER}}$), one of the winning submissions in the WMT2020 QE Shared Task, which we fine-tuned on the MLQE dataset. For noise filtering, we train five teacher models with random initialisation. As students, we use BiRNN models from DeepQuest (Ive et al., 2018). We also compare our results against the Predictor-Estimator model (Kim et al., 2017; Kepler et al., 2019), the baseline at the WMT2020 QE Shared Task, and TransQuest models using multilingual DistilBERT.[6]

4 Results

Table 3 shows the Pearson correlation with human judgments on the test partition of the MLQE dataset for different models and specifies the type of data used for training.[7] The correlation for the student models (BiRNN$_{\text{STUDENT}*}$) does not reach the performance of TQ$_{\text{TEACHER}}$. Smaller models may lack representation power for modeling cross-lingual tasks such as QE. Also, distillation for regression is more challenging, as discussed in Section 2. However, **training on the in-domain distilled data** (BiRNN$_{\text{STUDENT+AUG}}$) **allows to obtain performances comparable to DistilBERT** (TQ$_{\text{DistilBERT}}$) **with much lighter models** (see Table 1).[8] Furthermore, this approach results in a substantial improvement over shallow models trained on gold data (BiRNN and Predictor-Estimator) for all of the language pairs. The student performance for each language pair is strongly related to the performance of the teacher. Thus, the Ro-En student achieves the highest correlation results, whereas correlation for En-Zh is weak.

We further analyse what is the impact of different data selection strategies on the results. **First**, we sample random subsets of training instances from the Wikipedia distilled dataset and evaluate the performance of the student model trained with this data. We run the training 3 times with different random splits for training and validation and report the mean and confidence intervals. Table 4 shows the results for all languages where we have enough Wikipedia data (for Et-En we only have 25K in total). The largest boost in correlation is observed when going from 10K to 50K.

Second, we compare student models trained on these subsets of distilled data of different sizes, i.e. using data extracted from Wikipedia (in-domain), against data splits of the same size extracted from Common Crawl (out-of-domain). For the out-of-domain data we apply the noise filtering strategy described in Section 2. Figure 2 shows the results

---

[6] Multilingual DistilBERT is available at `https://huggingface.co/distilbert-base-multilingual-cased`. We follow the same training procedure as for the teacher model described in detail in Appendix A.

[7] TQ$_{\text{TEACHER}}$, TQ$_{\text{DistilBERT}}$ and Predictor-Estimator use contextual representations trained on large amounts of additional data, which are then fine-tuned for the QE task.

[8] This is true for all language pairs except Et-En and En-Zh. For Et-En we have a considerably smaller amount of in-domain data available for training, whereas for En-Zh the teacher model appears to be too weak to be useful for KD.

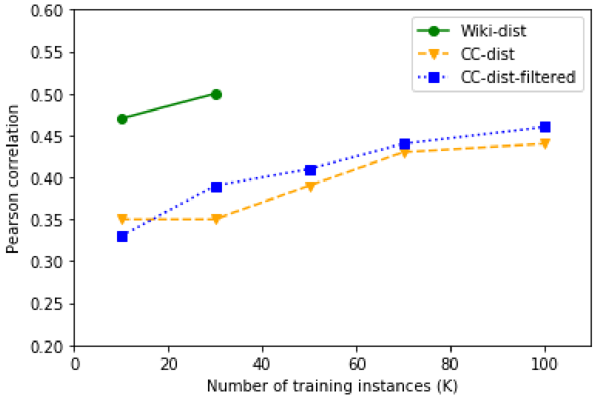

Figure 2: Pearson correlation results on the MLQE test set for the student models trained with different amounts of distilled data in-domain (Wiki-dist), out-of-domain (CC-dist) and out-of-domain with noise filtering (CC-dist-filtered), for Et-En.

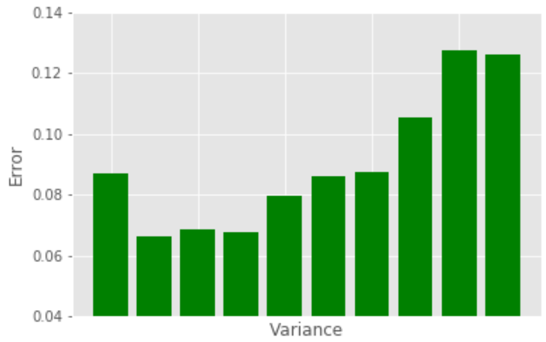

Figure 3: Variance in the predictions of 6 teacher models trained with different random seed against predictions error on the test partition of Ro-En MLQE dataset.

for Et-En, where our largest in-domain set has 25K sentences. We observe that using in-domain data appears to be much more effective than sampling larger amounts of generic data. Noise filtering gives some improvement in the results but its effect appears to be marginal compared to the effect of training with in-domain data.

Figure 3 provides an illustration of the relation between the variance in the predictions of multiple teacher models and prediction error for Ro-En language pair on the in-domain data. We group the sentences in the test partitions of MLQE dataset in 10 bins according to the variance between the predictions of the different teacher models in the ensemble. We then calculate the average prediction error in each bin, where the error is the absolute difference between model predictions and human judgements. As shown in Figure 3, higher variance in the predictions indeed corresponds to larger prediction error.

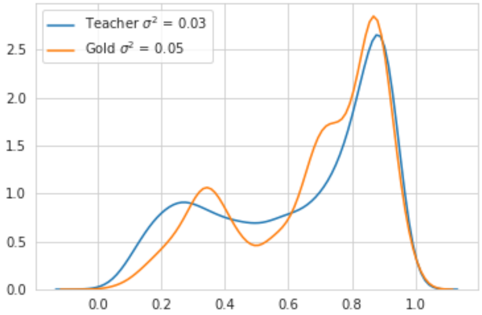

Figure 4: Distribution of teacher scores (blue) and gold labels (orange) on the training partitions of Et-En MLQE dataset.

Interestingly, from Table 3 we see that **training with distilled data brings benefits even without data augmentation** for some of the language pairs. The correlation for Et-En improves from 0.37 to 0.45 by training on teacher predictions (BiRNN$_{\text{STUDENT}}$) instead of gold labels (BiRNN) on the same MLQE dataset. To gain an intuition for this improvement, Figure 4 shows the distribution of teacher predictions and human scores on the train partition of MLQE dataset. We hypothesize that teacher predictions having a smoother distribution with reduced variance makes learning easier. As shown in Appendix B, we observe this trend for all language pairs in the dataset.

5 Conclusions

In this paper, we showed that knowledge distillation, through a teacher-student approach that directly distills QE predictions, can be effective in building a light-weight QE model with similar performance to a SoTA architecture trained on distilled yet large pre-trained representations. We also introduced a noise filtering approach that leverages the uncertainty of an ensemble of teacher models to determine which training instances should be discarded when training the student models, which can be beneficial especially for data augmentation from out-of-domain sources. This results in QE models 4x smaller in disk space with 8x fewer parameters, and 3x faster in inference speed.

Acknowledgements

This work was supported by funding from the Bergamot project (EU H2020 Grant No. 825303).

References

Guobin Chen, Wongun Choi, Xiang Yu, Tony X. Han, and Manmohan Chandraker. 2017. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 742–751.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR.

Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. 2020. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 33–41.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Julia Ive, Frédéric Blain, and Lucia Specia. 2018. DeepQuest: a framework for neural-based quality estimation. In *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics: Technical Papers*, Santa Fe, new Mexico.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 4163–4174. Association for Computational Linguistics.

Fábio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. 2019. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics–System Demonstrations*, pages 117–122, Florence, Italy. Association for Computational Linguistics.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2020. Transquest: Translation quality estimation with cross-lingual transformers. In *Proceedings of the 28th International Conference on Computational Linguistics*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Lucia Specia, Frédéric Blain, Marina Fomicheva, Erick Fonseca, Vishrav Chaudhary, Francisco Guzmán, and André F. T. Martins. 2020. Findings of the WMT 2020 shared task on quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4322–4331. Association for Computational Linguistics.

Makoto Takamoto, Yusuke Morishita, and Hitoshi Imaoka. 2020. An efficient method of training small models for regression problems with knowledge distillation. In *3rd IEEE Conference on Multimedia Information Processing and Retrieval, MIPR 2020, Shenzhen, China, August 6-8, 2020*, pages 67–72. IEEE.

Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H Chi, and Sagar Jain. 2020. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *CoRR*, abs/1903.12136.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

A Teacher and student models

Teacher model For the teacher model, we use the transformer-based MonoTransQuest (Ranasinghe et al., 2020) architecture of the TransQuest framework with XLMR-large (Conneau et al., 2020b) as the underlying pre-trained representation model. XLMR is Transformer (Vaswani et al., 2017) based masked language model (with 24 Transformer blocks and the vocabulary size of 250K) trained on one hundred languages using approximately two terabytes of CommonCrawl data. MonoTransQuest takes as input the concatenation of the original sentence and its translation separated by the the special $[SEP]$ token. The learned representation of the special $[CLS]$ token is considered as the joint representation of the original and translated sentence. The joint representation is then fed to the final $softmax$ layer to predict the quality score of the translation. For distillation we use the models that were made available for download by the authors.[9] For training the additional teacher models for data filtering we follow the training settings in Ranasinghe et al. (2020): we used a batch size of 8, Adam optimiser with learning rate 2e-5, and a linear learning rate warm-up over 10% of the training data. During the training process, the parameters of XLM-R model, as well as the parameters of the subsequent layers, are updated. All the models were trained for 3 epochs.

Student model For the student model, we rely on the BiRNN QE architecture proposed by Ive et al. (2018). Our implementation of this architecture is available for download.[10] The light-weight architecture (15 layers) of this model is as follows: both source and target sentences are independently encoded by a dedicated embedding layer followed by a bi-directional Recurrent Neural Network (RNN). The two resulting sentence representations are then concatenated as a weighted sum of their word vectors, generated by an attention mechanism. The resulting representation is then passed through an output dense layer with sigmoid activation to generate the quality estimates. We use the BiRNN model in its default configuration: both source and target embeddings are of size 300, each encoder has a hidden size of 50. The vocabulary size is limited to the 30k most common words. The model is trained with early stopping with a patience of 5.

B Output Distribution for Teacher Models

Figure 5 shows the distribution of teacher scores (blue) and gold labels (orange) on the training partitions for the language pairs in MLQE dataset.

---

[9] `https://tharindudr.github.io/TransQuest/pretrained/#available-models`

[10] `https://github.com/sheffieldnlp/deepQuest-py`

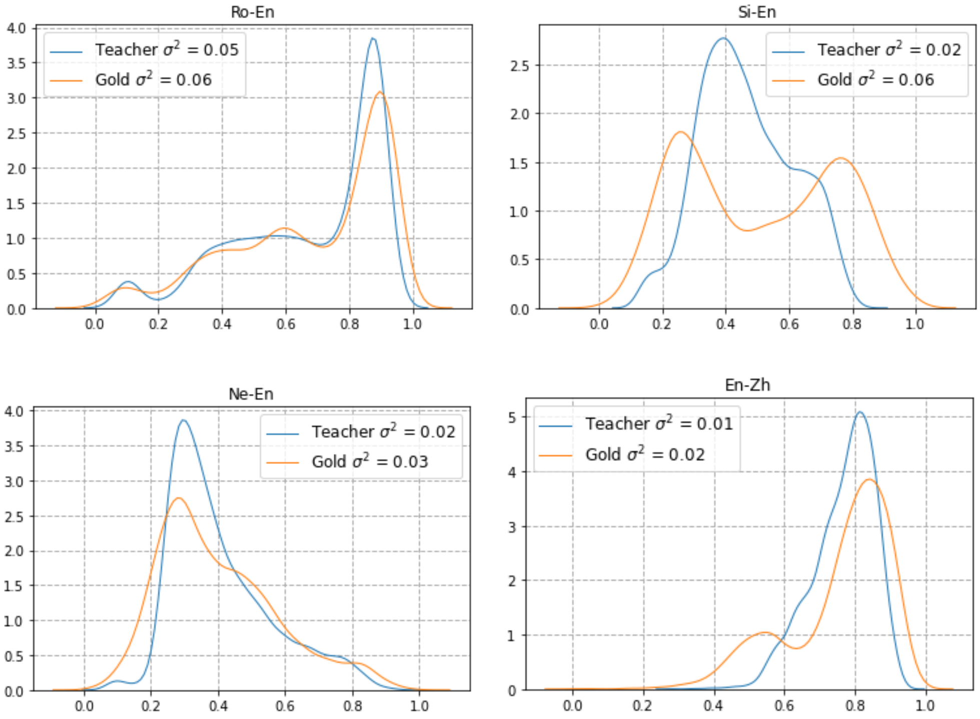

Figure 5: Distribution of teacher scores (blue) and gold labels (orange) on the training partitions for different language pairs in the MLQE dataset.

Cross-document Coreference Resolution over Predicted Mentions

Arie Cattan[1] **Alon Eirew**[1,2] **Gabriel Stanovsky**[3] **Mandar Joshi**[4] **Ido Dagan**[1]

[1]Computer Science Department, Bar Ilan University
[2]Intel Labs, Israel [3]The Hebrew University of Jerusalem
[4]Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
`arie.cattan@gmail.com` `alon.eirew@intel.com`
`gabis@cse.huji.ac.il` `mandar90@cs.washington.edu`
`dagan@cs.biu.ac.il`

Abstract

Coreference resolution has been mostly investigated within a single document scope, showing impressive progress in recent years based on end-to-end models. However, the more challenging task of cross-document (CD) coreference resolution remained relatively under-explored, with the few recent models applied only to gold mentions. Here, we introduce the first end-to-end model for CD coreference resolution from raw text, which extends the prominent model for within-document coreference to the CD setting. Our model achieves competitive results for event and entity coreference resolution on gold mentions. More importantly, we set first baseline results, on the standard ECB+ dataset, for CD coreference resolution over predicted mentions. Further, our model is simpler and more efficient than recent CD coreference resolution systems, while not using any external resources.[1]

1 Introduction

Cross-document (CD) coreference resolution consists of identifying textual mentions across multiple documents that refer to the same concept. For example, consider the following sentences from the ECB+ dataset (Cybulska and Vossen, 2014), where colors represent coreference clusters (for brevity, we omit some clusters):

1. ***Thieves*** ***pulled off*** *a two million euro jewellery heist in central Paris on Monday after* ***smashing*** ***their*** *car through the store's front window.*

2. ***Four men*** ***drove*** *a 4x4 through the front window of the store on Rue de Castiglione, before* ***making off*** *with the jewellery and watches.*

Despite its importance for downstream tasks, CD coreference resolution has been lagging behind the impressive strides made in the scope of a single document (Lee et al., 2017; Joshi et al., 2019, 2020; Wu et al., 2020). Further, state-of-the-art models exhibit several shortcomings, such as operating on gold mentions or relying on external resources such as SRL or a paraphrase dataset (Shwartz et al., 2017), preventing them from being applied on realistic settings.

To address these limitations, we develop the first end-to-end CD coreference model building upon a prominent within-document (WD) coreference model (Lee et al., 2017) which we extend with recent advances in transformer-based encoders. We address the inherently non-linear nature of the CD setting by combining the WD coreference model with agglomerative clustering that was shown useful in CD models. Our model achieves competitive results on ECB+ over gold mentions and sets baseline results over predicted mentions. Our model is also simpler and substantially more efficient than existing CD coreference systems. Taken together, our work seeks to bridge the gap between WD and CD coreference, driving further research of the latter in realistic settings.

2 Background

Cross-document coreference Previous works on CD coreference resolution learn a pairwise scorer between mentions and use a clustering approach to form the coreference clusters (Cybulska and Vossen, 2015; Yang et al., 2015; Choubey and Huang, 2017; Kenyon-Dean et al., 2018; Bugert et al., 2020). Barhom et al. (2019) proposed to jointly learn entity and event coreference resolution, leveraging predicate-argument structures. Their model forms the coreference clusters incrementally, while alternating between event and entity coreference. Based on this work, Meged et al. (2020) improved results on event coreference by leverag-

---

[1] `https://github.com/ariecattan/coref`

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5100–5107
August 1–6, 2021.

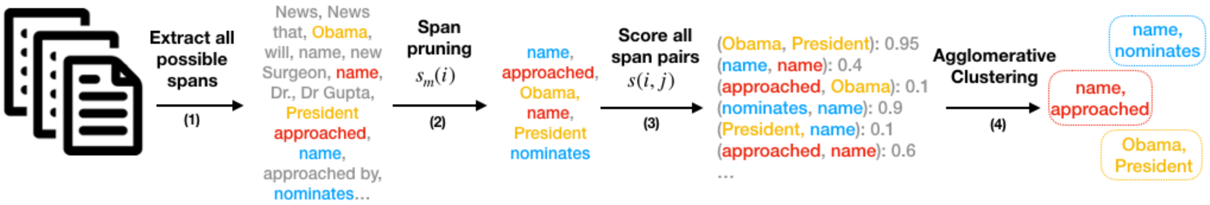

Figure 1: A high-level diagram of our model for cross-document coreference resolution. (1) Extract and score all possible spans, (2) keep top spans according to $s_m(i)$, (3) score all pairs $s(i,j)$, and (4) cluster spans using agglomerative clustering.

ing a paraphrase resource (Chirps; Shwartz et al., 2017) as distant supervision. Parallel to our work, recent approaches propose to fine-tune BERT on the pairwise coreference scorer (Zeng et al., 2020), where the state-of-the-art on ECB+ is achieved using a cross-document language model (CDLM) on pairs of full documents (Caciularu et al., 2021). Instead of applying BERT for all mentions pairs which is quadratically costly, our work separately encodes each (predicted) mention.

All above models suffer from several drawbacks. First, they use only gold mentions and treat entities and events separately.[2] Second, pairwise scores are recomputed after each merging step, which is resource and time consuming. Finally, they rely on additional resources, such as semantic role labeling, a within-document coreference resolver, and a paraphrase resource, which limits the applicability of these models in new domains and languages. In contrast, we use no such external resources.

Within-document coreference The *e2e-coref* WD coreference model (Lee et al., 2017) learns for each span i a distribution over its antecedents.

Considering all possible spans as potential mentions, the scoring function $s(i,j)$ between span i and j, where j appears before i, has three components: the two mention scores $s_m(\cdot)$ of spans i and j, and a pairwise antecedent score $s_a(i,j)$ for span j being an antecedent of span i.

Each span is represented with the concatenation of four vectors: the output representations of the span boundary (first and last) tokens ($x_{\text{FIRST(i)}}, x_{\text{LAST(i)}}$), an attention-weighted sum of token representations $\hat{x}_i$, and a feature vector $\phi(i)$. These span representations (g_i) are first fed into a mention scorer $s_m(\cdot)$ to filter the λT (where T is the number of tokens) spans with the highest scores. Then, the model learns for each of these spans to optimize the marginal log-likelihood of its correct antecedents. The full description of the model is described below:

$$\begin{aligned} g_i &= [x_{\text{FIRST(i)}}, x_{\text{LAST(i)}}, \hat{x}_i, \phi(i)] \\ s_m(i) &= \text{FFNN}_m(g_i) \\ s_a(i,j) &= \text{FFNN}_a([g_i, g_j, g_i \circ g_j]) \\ s(i,j) &= s_m(i) + s_m(j) + s_a(i,j) \end{aligned}$$

3 Model

The overall structure of our model is shown in Figure 1. The major obstacle in applying the *e2e-coref* model directly in the CD setting is its reliance on *textual ordering* – it forms coreference chains by linking each mention to an antecedent span appearing before it in the document. This linear clustering method cannot be used in the multiple-document setting since there is no inherent ordering between the documents. Additionally, ECB+ (the main benchmark for CD coreference resolution) is relatively small compared to OntoNotes (Pradhan et al., 2012), making it hard to jointly optimize mention detection and coreference decision. These challenges have implications in all stages of model development, as elaborated below.

Pre-training To address the small scale of the dataset, we pre-train the mention scorer $s_m(\cdot)$ on the gold mention spans, as ECB+ includes singleton annotation. This enables generating good candidate spans from the first epoch, and as we show in Section 4.3, it substantially improves performance.

Training Instead of comparing a mention only to its previous spans in the text, our pairwise scorer $s_a(i,j)$ compares a mention to all other spans across all the documents. The positive instances for training consist of all the pairs of highest scoring

---

[2] Few works (Yang et al., 2015; Choubey and Huang, 2017) do use predicted mentions, by considering the intersection of predicted and gold mentions for evaluation, and thus not penalizing models for false positive mention identification. Moreover, they used a different version of ECB+ with known annotation errors, as noted in Barhom et al. (2019).

mention spans that belong to the same coreference cluster, while the negative examples are sampled (20x the number of positive pairs) from all other pairs. The overall score is then optimized using the binary cross-entropy loss as follows:

$$L = -\frac{1}{|N|}\sum_{x,z \in N} y \cdot log(s(x,z))$$

where N corresponds to the set of mention-pairs (x, z), and $y \in \{0, 1\}$ to a pair label. Full implementation details are described in Appendix A.1. Notice that the mention scorer $s_m(\cdot)$ is further trained in order to generate better candidates at each training step. When training and evaluating the model in experiments over gold mentions, we ignore the span mention scores, $s_m(\cdot)$, and the gold mention representations are directly fed into the pairwise scorer $s_a(i, j)$.

Inference At inference time, we score all spans; prune spans with lowest scores; score the pairs; and finally form the coreference clusters using an agglomerative clustering (using average-linking method) over these pairwise scores, following common practices in CD coreference resolution (Yang et al., 2015; Choubey and Huang, 2017; Kenyon-Dean et al., 2018; Barhom et al., 2019). Since the affinity scores $s(i, j)$ are also computed for mention pairs in different documents, the agglomerative clustering can effectively find cross-document coreference clusters.

4 Experiments

4.1 Experimental setup

Following most recent work, we conduct our experiments ECB+ (Cybulska and Vossen, 2014), which is the largest dataset that includes both WD and CD coreference annotation (see Appendix A.2). We use the document clustering of Barhom et al. (2019) for pre-processing and apply our coreference model separately on each predicted document cluster.

Following Barhom et al. (2019), we present the model's performance on both event and entity coreference resolution. In addition, inspired by Lee et al. (2012), we train our model to perform event and entity coreference jointly, which we term "ALL". This represents a useful scenario when we are interested in finding all the coreference links in a set of documents, without having to distinguish event and entity mentions. Addressing CD coreference with ALL is challenging because (1) the search space is larger than when treating separately event and entity coreference and (2) models need to make subtle distinctions between event and entity mentions that are lexically similar but do not corefer. For example, the entity *voters* do not corefer with the event *voted*.

We apply RoBERTa$_{LARGE}$ (Liu et al., 2019) to encode the documents. Long documents are split into non-overlapping segments of up to 512 wordpiece tokens and are encoded independently (Joshi et al., 2019). Due to memory constraints, we freeze output representations from RoBERTa instead of fine-tuning all parameters. For all experiments, we use a single GeForce GTX 1080 Ti 12GB GPU. The training takes 2.5 hours for the most expensive setting (ALL on predicted mentions), while inference over the test set takes 11 minutes.

4.2 Results

Table 1 presents the combined within- and cross-document results of our model, in comparison to previous work on ECB+. We report the results using the standard evaluation metrics MUC, B^3, CEAF, and the average F1 of these metrics, called CoNLL F1 (main evaluation).

When evaluated on gold mentions, our model achieves competitive results for event (81 F1) and entity (73.1) coreference. In addition, we set baseline results where the model does not distinguish between event and entity mentions at inference time (denoted as the ALL setting). The overall performance on ECB+ obtained using two separate models for event and entity is negligibly higher (+0.6 F1) than our single ALL model.

Our model is the first to enable end-to-end CD coreference on *raw* text (predicted mentions). As expected, the performance is lower than that using gold mentions (e.g 26.6 F1 drop in event coreference), indicating the large room for improvement over predicted mentions. It should be noted that beyond mention detection errors, two additional factors contribute to the performance drop when moving to predicted mentions. First, while WD coreference systems typically disregard singletons (mentions appearing only once) when evaluating on *raw* text, CD coreference models do consider singletons when evaluating on *gold* mentions on ECB+. We observe that this difference affects the evaluation, explaining about 10% absolute points out of the aforementioned drop of 26.6. The effect of singletons on coreference evaluation is further

		MUC			B^3			$CEAFe$			CoNLL
		R	P	F_1	R	P	F_1	R	P	F_1	F_1
Event	Barhom et al. (2019)	77.6	84.5	80.9	76.1	85.1	80.3	81.0	73.8	77.3	79.5
	Meged et al. (2020)	78.8	84.7	81.6	75.9	85.9	80.6	81.1	74.8	77.8	80.0
	Our model – Gold	85.1	81.9	83.5	82.1	82.7	82.4	75.2	78.9	77.0	*81.0*
	Zeng et al. (2020)*	85.6	89.3	87.5	77.6	89.7	83.2	84.5	80.1	82.3	84.6
	Caciularu et al. (2021)*	87.1	89.2	88.1	84.9	87.9	86.4	83.3	81.2	82.2	**85.6**
	Our model – Predicted	66.6	65.3	65.9	56.4	50.0	53.0	47.8	41.3	44.3	***54.4***
Entity	Barhom et al. (2019)	78.6	80.9	79.7	65.5	76.4	70.5	65.4	61.3	63.3	71.2
	Our model – Gold	85.7	81.7	83.6	70.7	74.8	72.7	59.3	67.4	63.1	*73.1*
	Caciularu et al. (2021)*	88.1	91.8	89.9	82.5	81.7	82.1	81.2	72.9	76.8	**82.9**
	Our model – Predicted	43.5	53.1	47.9	25.0	38.9	30.4	31.4	26.5	28.8	***35.7***
ALL	Our model – Gold	84.2	81.6	82.9	76.8	77.5	77.1	68.4	72.4	70.3	***76.7***
	Our model – Predicted	49.7	58.5	53.7	33.2	46.5	38.7	40.4	35.2	37.6	***43.4***

Table 1: Combined within- and cross-document results on the ECB+ test set, for event, entity and the unified task, that we term ALL. Our results (in italics) are close to state-of-the-art (in bold) for event and entity over gold mentions, while they set a new benchmark result over predicted mentions and for the ALL setting. The run-time complexity in (Zeng et al., 2020; Caciularu et al., 2021) is substantially more expensive because they apply BERT and CDLM for every mention-pair with their corresponding context (sentence and full document).

	Gold		Predicted	
	WD	CD	WD	CD
Event	**86.6**	81.0	**59.6**	54.4
Entity	**81.2**	73.1	**39.7**	35.7
ALL	**83.9**	76.7	**46.3**	43.4

Table 2: Results (CoNLL F1) of our model, on within-document (WD) vs. cross-document (CD), using gold and predicted mentions. For all settings, results on WD are higher, indicating the need in addressing typical challenges of CD coreference resolution.

	Gold	Δ	**Predicted**	Δ
Event	76.0	**–5.0**	48.2	**–6.2**
Entity	70.9	–2.2	34.4	–1.3
ALL	74.1	–2.6	41.4	–2.0

Table 3: CoNLL F1 results of our model without document clustering , using gold and predicted mentions.

explored in (Cattan et al., 2021). Second, entities are annotated in ECB+ only if they participate in event, making participant detection an additional challenge. This explains the more important performance drop in entity and ALL.

Table 2 presents the CoNLL F1 results of within- and cross-document coreference resolution for both gold and predicted mentions on ECB+. For all settings, results are higher in within-document coreference resolution, showing the need in addressing typical challenges of CD coreference resolution.

Table 3 shows the results of our model without document clustering. Here, the performance drop and error reduction are substantially larger for event coreference (-6.2/12%) than entity coreference (-1.3/2%) and ALL (-2/3.5%). This difference is probably due to the structure of ECB+ which poses a lexical ambiguity challenge for events, while the document clustering step reconstructs almost perfectly the original subtopics, as shown in (Barhom et al., 2019).

Further, the higher results on event coreference do not mean that the task is inherently easier than entity coreference. In fact, when ignoring singletons in the evaluation, as done on OntoNotes, the performance of event coreference is lower than entity coreference (62.1 versus 65.3 CoNLL F1) (Cattan et al., 2021). This happens because event singletons are more common compared to entity singletons (30% vs. 17%), as shown in Appendix A.2.

Finally, our model is more efficient in both training and inference since the documents are encoded using just one pass of RoBERTa, and the pairwise scores are computed only once using a simple MLP. For comparison, previous models compute pairwise scores at each iteration (Barhom et al., 2019; Meged et al., 2020), or apply a BERT-model to every mention pairs with their sentence (Zeng et al., 2020) or full document (Caciularu et al., 2021).[3]

---

[3] For a rough estimation, our model runs for 2 minutes while Barhom et al. (2019)'s model runs for 37 minutes on similar hardware.

	F1	Δ
Our model	58.1	
− pre-train of mention scorer	54.9	−3.2
− dynamic pruning	54.1	−4.0
− negative sampling	56.7	−1.4

Table 4: Ablation results (CoNLL F1) of our model on the development set of ECB+ event coreference.

4.3 Ablations

To show the importance of each component of our model, we ablate several parts and compute F1 scores on the development set of the ECB+ event dataset. The results are presented in Table 4 using predicted mentions without document clustering.

Skipping the *pre-training* of the mention scorer results in a 3.2 F1 points drop in performance. Indeed, the relatively small training data in the ECB+ dataset (see Appendix A.2) might be not sufficient when using only end-to-end optimization, and pre-training of the mention scorer helps generate good candidate spans from the first epoch.

To analyze the effect of the dynamic pruning, we froze the mention scorer during the pairwise training, and kept the same candidate spans along the training. The significant performance drop (4 F1) reveals that the mention scorer inherently incorporates coreference signal.

Finally, using all negative pairs for training leads to a performance drop of 1.4 points and significantly increases the training time.

4.4 Qualitative Analysis

We sampled topics from the development set and manually analyzed the errors of the ALL configuration. The most common errors were due to an over-reliance on lexical similarity. For example, the event “*Maurice Cheeks was* ***fired***” was wrongly predicted to be coreferent with a similar, but different event, “*the Sixers* ***fired*** *Jim O’Brien*”, probably because of related context, as both coached the Philadelphia 76ers. On the other hand, the model sometimes struggles to merge mentions that are lexically different but semantically similar (e.g “*Jim O’Brien* ***was shown the door***”, “*Philadelphia* ***fire*** *coach Jim O’Brien*”). The model also seems to struggle with temporal reasoning, in part due to missing information. For example, news articles from different days have different relative reference to time, while the publication date of the articles is not always available. As a result, the model missed linking “*Today*” in one document to “*Saturday*” in another document.

5 Conclusion and Discussion

We developed the first end-to-end baseline for CD coreference resolution over predicted mentions. Our simple and efficient model achieve competitive results over gold mentions on both event and entity coreference, while setting baseline results for future models over predicted mentions.

Nonetheless, we note a few limitations of our model that could be addressed in future work. First, following most recent work on cross-document coreference resolution (§2), our model requires $\mathcal{O}(n^2)$ pairwise comparisons to form the coreference cluster. While our model is substantially more efficient than previous work (§4.2), applying it on a large-scale dataset would involve a scalability challenge. Future work may address the scalability issue by using recent approaches for hierarchical clustering on massive datasets (Monath et al., 2019, 2021). Another appealing approach consists of splitting the corpus into subsets of documents, constructing initial coreference clusters (in parallel) on the subsets, then merging meta-clusters from the different sets. We note though that it is currently impossible to test such solutions for more extensive scalability, pointing to a need in collecting larger-scale datasets for cross-document coreference. Second, to improve overall performance over predicted mentions, future work may incorporate, explicitly or implicitly, semantic role labeling signals in order to identify event participants for entity prediction, as well as for better event structure matching. Further, dedicated components may be developed for mention detection and coreference linking, which may be jointly optimized.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. The work described herein was supported in part by grants from Intel Labs, Facebook, the Israel Science Foundation grant 1951/17, the Israeli Ministry of Science and Technology, the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), and from the Allen Institute for AI.

Ethical Considerations

Dataset As mentioned in the paper, we use the ECB+ dataset, available at `http://www.newsreader-project.eu/results/data/the-ecb-corpus/`.

Model Our cross-document coreference model does not contain any intentional biasing or ethical issues. As mentioned in the paper (§4.1), we conduct our experiments on a single 12GB GPU, and both training and inference times are relatively low.

References

Shany Barhom, Vered Shwartz, Alon Eirew, Michael Bugert, Nils Reimers, and Ido Dagan. 2019. Revisiting joint modeling of cross-document entity and event coreference resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4179–4189, Florence, Italy. Association for Computational Linguistics.

Cosmin Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422, Uppsala, Sweden. Association for Computational Linguistics.

Michael Bugert, N. Reimers, and Iryna Gurevych. 2020. Cross-document event coreference resolution beyond corpus-tailored systems. *ArXiv*, abs/2011.12249.

Avi Caciularu, Arman Cohan, Iz Beltagy, Matthew E. Peters, Arie Cattan, and Ido Dagan. 2021. Cross-document language modeling. *ArXiv*, abs/2101.00406.

Arie Cattan, Alon Eirew, Gabriel Stanovsky, Mandar Joshi, and Ido Dagan. 2021. Realistic evaluation principles for cross-document coreference resolution. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

Prafulla Kumar Choubey and Ruihong Huang. 2017. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2133, Copenhagen, Denmark. Association for Computational Linguistics.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4545–4552, Reykjavik, Iceland. European Language Resources Association (ELRA).

Agata Cybulska and Piek Vossen. 2015. Translating granularity of event slots into features for event coreference resolution. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 1–10, Denver, Colorado. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Kian Kenyon-Dean, Jackie Chi Kit Cheung, and Doina Precup. 2018. Resolving event coreference with supervised representation learning and clustering-oriented regularization. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 1–10, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500, Jeju Island, Korea. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yehudit Meged, Avi Caciularu, Vered Shwartz, and Ido Dagan. 2020. Paraphrasing vs coreferring: Two sides of the same coin. In *Findings of the Association for Computational Linguistics: EMNLP 2020*,

pages 4897–4907, Online. Association for Computational Linguistics.

Nicholas Monath, Kumar Avinava Dubey, Guru Guruganesh, M. Zaheer, Amr Ahmed, A. McCallum, Gokhan Mergen, Marc Najork, Mert Terzihan, B. Tjanaka, Yuan Wang, and Yuchen Wu. 2021. Scalable Bottom-up Hierarchical Clustering. *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Nicholas Monath, A. Kobren, A. Krishnamurthy, Michael R. Glass, and A. McCallum. 2019. Scalable Hierarchical Clustering with Tree Grafting. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Vered Shwartz, Gabriel Stanovsky, and Ido Dagan. 2017. Acquiring predicate paraphrases from news tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 155–160, Vancouver, Canada. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent Bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:517–528.

Yutao Zeng, Xiaolong Jin, Saiping Guan, Jiafeng Guo, and Xueqi Cheng. 2020. Event coreference resolution with their paraphrases and argument-aware embeddings. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3084–3094, Barcelona, Spain (Online). International Committee on Computational Linguistics.

A Appendix

A.1 Implementation Details

Our model includes 14M parameters and is implemented in PyTorch (Paszke et al., 2019), using HuggingFace's library (Wolf et al., 2020) and the Adam optimizer (Kingma and Ba, 2014). The layers of the models are initialized with Xavier Glorot method (Glorot and Bengio, 2010). We manually tuned the standard hyperparameters, presented in Table 5 on the event coreference task and keep them unchanged for entity and ALL settings. Table 6 shows specific parameters, such as the maximum span width, the pruning coefficient λ and the stop criterion τ for the agglomerative clustering, that we tuned separately for each setting to maximize the CoNLL F1 score on its corresponding development set.

Hyperparameter	Value
Batch size	32
Dropout	0.3
Learning rate	0.0001
Optimizer	Adam
Hidden layer	1024

Table 5: Shared hyperparameters across the different models.

	Max span width	λ	τ
Event	10	0.25	0.75
Entity	15	0.35	0.75
ALL	15	0.4	0.75

Table 6: Specific hyperparameters for each mention type; λ is the pruning coefficient and τ is the threshold for the agglomerative clustering.

A.2 Dataset

ECB+[4] is an extended version of the EventCorefBank (ECB) (Bejan and Harabagiu, 2010) and EECB (Lee et al., 2012), whose statistics are shown in Table 7. The dataset is composed of 43 *topics*, where each topic corresponds to a famous news event (e.g Someone checked into rehab). In order to introduce some complexity and to limit the use of lexical features, each topic is constituted by a collection of texts describing two different event instances of the same event type, called *subtopic*. For example, the first topic corresponding to the event *"Someone checked into rehab"* is composed of event mention of the event *"Tara Reid checked into rehab"* and *"Lindsay Lohan checked into rehab"* which are obviously annotated into different coreference cluster. Documents in ECB+ are in English. Since ECB+ is an *event-centric* dataset, entities are annotated only if they participate in events. In this dataset, event and entity coreference clusters are denoted separately.

	Train	**Validation**	**Test**
# Topics	25	8	10
# Documents	594	196	206
# Mentions	3808/4758	1245/1476	1780/2055
# Singletons	1116/814	280/205	632/412
# Clusters	1527/1286	409/330	805/608

Table 7: ECB+ statistics. The slash numbers for # Mentions, # Singletons and # Clusters represent event/entity statistics. As recommended by the authors in the release note, we follow the split of Cybulska and Vossen (2015) that use a curated subset of the dataset.

---

[4] `http://www.newsreader-project.eu/results/data/the-ecb-corpus/`

Controllable Abstractive Dialogue Summarization with Sketch Supervision

Chien-Sheng Wu*[1], Linqing Liu*[2], Wenhao Liu[1], Pontus Stenetorp[2], Caiming Xiong[1]
[1] Salesforce Research [2] University College London
`{linqing.liu,p.stenetorp}@cs.ucl.ac.uk, {wu.jason,wenhao.liu,cxiong}@salesforce.com`

Abstract

In this paper, we aim to improve abstractive dialogue summarization quality and, at the same time, enable granularity control. Our model has two primary components and stages: 1) a two-stage generation strategy that generates a preliminary *summary sketch* serving as the basis for the final summary. This summary sketch provides a weakly supervised signal in the form of pseudo-labeled interrogative pronoun categories and key phrases extracted using a constituency parser. 2) A simple strategy to control the granularity of the final summary, in that our model can automatically determine or control the number of generated summary sentences for a given dialogue by predicting and highlighting different text spans from the source text. Our model achieves state-of-the-art performance on the largest dialogue summarization corpus SAMSum, with as high as 50.79 in ROUGE-L score. In addition, we conduct a case study and show competitive human evaluation results and controllability to human-annotated summaries.

1 Introduction

Text summarization aims to produce an abridged version of the input text by distilling its most critical information. In particular, abstractive – as opposed to extractive – summarization requires generative models with a high level of semantic understanding, as the output words do not necessarily appear in the source text. While it is more challenging, it gives more flexibility to a summary compared to extractive summarization models (Zhang et al., 2018). Significant research efforts have been focused on summarization of single-speaker documents such as text documents (Liao et al., 2018), News (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017) or scientific publications (Qazvinian and Radev, 2008; Nikolov et al., 2018). However, dialogue summarization has not received much attention despite the prevalence of dialogues (text messages, email, social media, etc.) and the vast application potential of dialogue summarization systems.

Since dialogue language is inherently different from written text, it poses a unique set of challenges (Zechner, 2001): 1) *Distributed information across multiple speakers.* The most important information is usually scattered across several conversation turns from different speakers, while in articles it mostly presents in titles or the first few sentences. 2) *Boundary detection.* In each turn pauses do not always match linguistic sensible segments; it is difficult to identify various critical information across turns due to surrounding non-content noise and disfluency. 3) *Modeling interactions between speakers.* The speaker interaction plays an important role as it would imply the current dialog state and the status of the next speaker. If we directly apply neural abstract summarization models which mostly encode the whole input only as a source sequence, the flow of the dialogue would be overlooked (Pan et al., 2018). Previous methods (Goo and Chen, 2018; Liu et al., 2019) rely on explicit annotations to capture the logic of the dialogue, however, such annotations are not always available in datasets and additional labeling is cumbersome.

To solve these challenges, we propose CODS, a COntrollable abstractive Dialogue Summarization model equipped with sketch generation. We first automatically create a summary sketch that contains user intent information and essential key phrases that may appear in summary. It identifies the interaction between speakers and salient information in each turn. This summary sketch is prefixed to the human-annotated summary while fine-tuning a generator, which provides weak supervision as the final summary is conditioned on the

---

*Equal contribution. Work mainly done when Linqing Liu was an intern at Salesforce Research.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5108–5122
August 1–6, 2021. ©2021 Association for Computational Linguistics

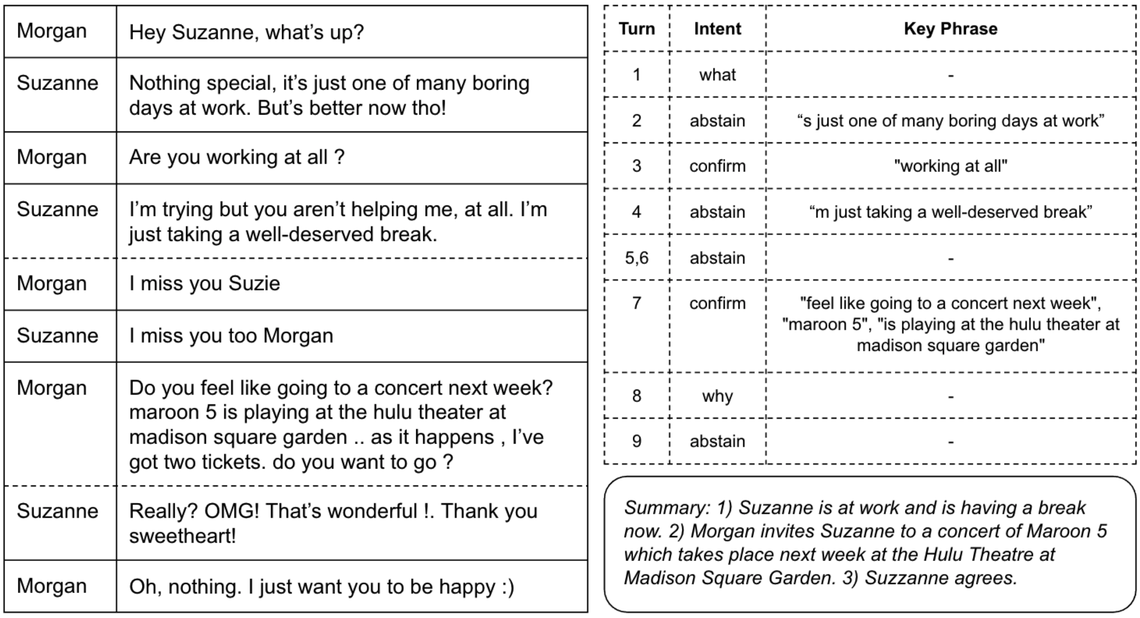

Morgan	Hey Suzanne, what's up?
Suzanne	Nothing special, it's just one of many boring days at work. But's better now tho!
Morgan	Are you working at all ?
Suzanne	I'm trying but you aren't helping me, at all. I'm just taking a well-deserved break.
Morgan	I miss you Suzie
Suzanne	I miss you too Morgan
Morgan	Do you feel like going to a concert next week? maroon 5 is playing at the hulu theater at madison square garden .. as it happens , I've got two tickets. do you want to go ?
Suzanne	Really? OMG! That's wonderful !. Thank you sweetheart!
Morgan	Oh, nothing. I just want you to be happy :)

Turn	**Intent**	**Key Phrase**
1	what	-
2	abstain	"s just one of many boring days at work"
3	confirm	"working at all"
4	abstain	"m just taking a well-deserved break"
5,6	abstain	-
7	confirm	"feel like going to a concert next week", "maroon 5", "is playing at the hulu theater at madison square garden"
8	why	-
9	abstain	-

Summary: 1) Suzanne is at work and is having a break now. 2) Morgan invites Suzanne to a concert of Maroon 5 which takes place next week at the Hulu Theatre at Madison Square Garden. 3) Suzzanne agrees.

Figure 1: An input and output example. Given the dialogue, we first construct a summary sketch with intent and key phrase information for each turn, and then split the dialogue into several segments (marked with dashed lines on the left hand side) for model controllability and interpretability.

generated summary sketch. In addition, we propose a length-controllable generation method specifically for dialogue summarization. Desired lengths of summaries strongly depend on the amount of information contained in the source dialogue and granularity of information the user wants to understand (Kikuchi et al., 2016). We first segment the dialogue into different segments by matching each summary sentence linearly to its corresponding dialogue context. Then we train our model to generate only one sentence for each dialogue segment. This strategy makes use of the distributed information of the dialogue and make the generated summaries more trackable.

We base our model on BART-xsum (Lewis et al., 2019), which is first pre-trained with unsupervised denoising objectives, and further fine-tuned on the News summarization corpus XSUM (Narayan et al., 2018). We evaluate our approach on SAMSum (Gliwa et al., 2019), the largest dialogue summarization dataset. Experimental results show that CODS achieves state-of-the-art dialogue summarization performance on several automatic metrics. The main contributions of this work[1] are: 1) We propose a two-stage strategy that uses artificial summary sketch as weak supervision, 2) we introduce a text-span based conditional generation approach to control the granularity of generated dialogue summaries without human-written summaries at different detail levels, and 3) we conduct comprehensive case study and human evaluation to show that CODS can achieve consistent and informative summary, especially for controllable summary, where existing models either cannot do it or do it poorly.

2 Methodology

Our model is based on pre-trained generative language models (Section 2.1). Given an input dialogue history, our model first generates a summary sketch that serves as additional weakly supervised signal for the final summary (Section 2.2). Then it predicts the text span cutoffs over the entire dialogue and generates summaries accordingly (Section 2.3). We define the conversational history input as $D = \{X_1, X_2, \dots, X_N\}$, where each X_i has a sequence of words, N is the total numbers of dialogue turns, and the input may contain more than two speakers. We intend to generate M-sentence dialogue summary $Y = \{Y_1, \dots, Y_M\}$ that is suppose to be briefer than the overall dialogue history.

2.1 Generative Pre-trained Language Models

As a first, our model needs transform a conversational history input into a dialogue summary. Re-

---

[1] Our code is released at `https://github.com/salesforce/ConvSumm`

cently, self-supervised pretrained language models have been employed as encoders and decoders since they (Radford et al., 2019; Yang et al., 2019; Dong et al., 2019) have achieved remarkable success across many NLP tasks. For general text summarization, this has also been the case with models such as BART (Lewis et al., 2019) and PEGASUS (Zhang et al., 2019a). However, there are no results reported for self-supervised pretrained language models applied to dialogue summarisation, and people have argued that there is an intrinsic difference of linguistic patterns between human conversations and written text (Wolf et al., 2019b; Wu et al., 2020a; Wu and Xiong, 2020). We would like to answer the question which generative language model is the best base model for dialogue summarization tasks.

2.2 Sketch Construction

Conversational data, unlike news or scientific publications, includes lots of non-factual sentences such as chit-chats and greetings. Removing these least critical information in the dialogues could potentially help the model better focus on the main content. Based on this hypothesis, we combine a syntax-driven sentence compression method (Xu and Durrett, 2019) with neural content selection.

Another potentially useful attribute for the conversational data is each dialogue turn inherently encodes user intent. However, unlike task-oriented dialogue systems, which have explicit annotated intents (e.g., book flight and check account), dialogue summarization data rarely have such labels. Thus we use a few heuristics with Snorkel (Ratner et al., 2019) to programmatically label each turn with a predefined interrogative pronoun category. The generated intents and the compressed dialogues together constitutes the summary sketch as weakly-supervised signals.

To the best of our knowledge, in general, there is no non-task-oriented established label set. Thus we draw upon the FIVE Ws principle, which often mentioned in journalism and research investigation, in that a passage can only be considered as complete if it answers these questions starting with such interrogative words (Hart). We adapt this principle to the dialogue scenario and identify a set of interrogative pronouns to support diverse enough user intents of all utterances, serving as the dialogue's logic. For example, in Figure 1, Morgan asked Suzanne "Do you feel like going to a concert next week?" One can expect that Suzanne will confirm her willingness in the next utterance. We define such dialogue intent categories including why, what, where, confirm, and abstain. More information for each category is shown in the Appendix (A.1).

To compress and remove noisy sub-sentences in the dialog, we first use a trained constituency parser (Kitaev and Klein, 2018) to parse each utterance. Then we compare the parsed phrases with the ground-truth summary to find their longest common sub-sequence (lcs), we set a threshold to filter and remove non-meaningful words (e.g., stop words) in lcs. Note that there are circumstances where the whole utterance is noisy and removable. Overall, we construct a summary sketch by concatenating utterance index, user intent label, and compressed utterance within the entire dialogue history into a string, ending with a special token, "TL;DR". Take Figure 1 as an example, the summary sketch is "1 what 2 abstain 's just one of ... square garden 8 why 9 abstain TL;DR". We train our model first to generate this summary sketch and then generate the final summary in an autoregressive way. We use TL;DR token to distinguish sketch and final summary during inference time.

2.3 Controllability

Due to the success of controllable language modeling (Keskar et al., 2019), the ability to control text summarization in the News domain has gradually been attracting attention (Fan et al., 2018; Liu et al., 2018) The high-level intuition for our solution is that if we can control a generative model only to generate one sentence as output for a partially-highlighted input, we can control the number of output sentences by choosing how to highlight the input. We highlight each dialogue split using the special token $< hl >$. For example, in Figure 1, we generate the first summary sentence for the first segment from turn one to four, and the second and third from turn five to seven and turn eight to nine, respectively (separated by the dashed lines). This way, we can not only gain the summary controllability but also make the generation more interpretable.

The next challenge is, during training, we have to find a mapping between each sentence in a reference summary to its corresponding dialogue split. In other words, how do we know where to insert the highlighting tokens? We do so by training a dialogue-turn-level binary classifier (detailed below) that predicts whether each turn is a cutting point (i.e., dialogue segmentation). Our observation is that sentences within a reference summary usually have a strong temporal dependency, that is, people summarize the dialogue almost linearly. We use a simple approach to find the cutting points: the highest similarity score between conversations and each summary sentence. The cutting point

$$t_m = \arg\max_t \text{SIM}(X_{c_m:t}, Y_m), \qquad (1)$$

where SIM could be any similarity functions (we use ROUGE-1), and c_m is the accumulated turn index ($c_1 = 1$ and $c_m = t_{m-1}$) that indicates which part of a dialogue has been covered. Note that for a summary with M sentences, we only need to decide $M - 1$ cutting points. With the pseudo labels (t_m) provided by this heuristic, we formulate the dialogue segmentation problem into a binary classification problem. Specifically, we train a classifier C, which takes dialogue history as input and predicts whether each dialogue turn is a cutting point. We prefix each dialogue turn with a separation token as input to the classifier.

$$\begin{aligned} H &= C([x_{sep}, X_1, x_{sep}, X_2, \ldots]) \in \mathbb{R}^{N \times d_{emb}}, \\ \hat{P} &= \text{Sigmoid}(W_1(H)) \in \mathbb{R}^N. \end{aligned} \qquad (2)$$

The classifier output H is the representations of those separation tokens, and each of them is a d_{emb} dimension vector. $W_1 \in \mathbb{R}^{d_{emb} \times 1}$ is a trainable linear mapping. The $\hat{P}$ is the predicted segment probability that is trained with binary cross-entropy loss. We use a BERT-base model (Devlin et al., 2018) as classifier and the i-th cutting point is triggered if $\hat{P}_i > 0.5$. This prediction means that our model can automatically determine how many sentences should be generated in the final summary. If no cutting point is triggered, we generate a one-sentence summary. If one cutting point is triggered, we will have a two-sentence summary, and so forth.

Finally, we can control the number of output summary sentences by controlling the dialogue split. Specifically, we first decide the expected number of output sentences (e.g., K), and then we choose the top $K-1$ indexes with highest probabilities in segmentation probability $\hat{P}$. We use these $K - 1$ indexes as cutting points. We can also generate one-sentence summary by clipping the whole dialogue with one pair of highlighting tokens at the beginning and the end of a dialogue (we call this setting as CODS-1).

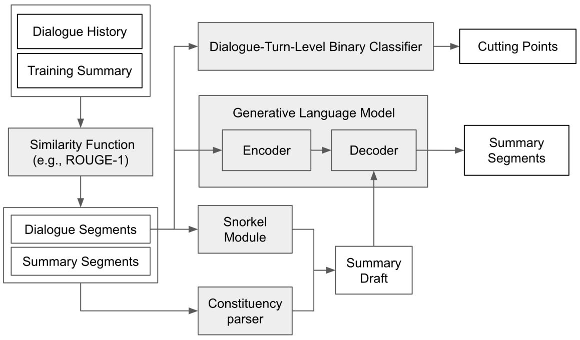

(a)

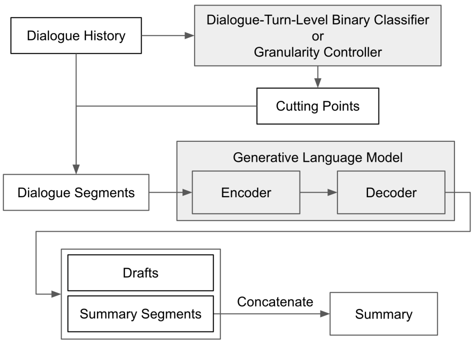

(b)

Figure 2: (a) Training and (b) inference block diagrams of CODS. Grey boxes are trainable functions.

2.4 Overall Generation

The overall training and inference block diagrams are shown in Figure 2. CODS follows a standard encoder-decoder framework. During training, we use dialogue segmentation to add highlighting tokens for each summary sentence. We take the highlighted dialogue history as input and train our model to generate its corresponding summary sketch and summary sentence. For example in Figure 1, the first summary sentence, we input the whole dialogue with added highlighting tokens both at the beginning of the first turn and at the end of the fourth turn, and generate output that contains the corresponding summary sketch "1 what 2 abstain ... well-deserved break" and the first summary sentence "Suzanne is at work and is having a break now." The entire model is trained using cross-entropy loss for the generated tokens. During inference, we first use the trained binary classifier to predict cutting points. Then, we use the predicted segmentation to add highlighting tokens into a dialogue. Finally, after generating multiple summary sentences separately, we concatenate them to be the final summary.

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
Longest-3*	32.46	10.27	29.92
Pointer Generator (See et al., 2017)*	37.27	14.42	34.36
Fast Abs RL (Chen and Bansal, 2018)*	41.03	16.93	39.05
Transformer (Vaswani et al., 2017)*	42.37	18.44	39.27
DynamicConv (Wu et al., 2019b)*	41.07	17.11	37.27
DynamicConv + GPT-2 emb*	45.41	20.65	41.45
D-HGN (Feng et al., 2020)	42.03	18.07	39.56
TGDGA (Zhao et al., 2020)	43.11	19.15	40.49
DialoGPT (Zhang et al., 2019d)	39.77	16.58	38.42
UniLM (Dong et al., 2019)	47.85	24.23	46.67
PEGASUS (Zhang et al., 2019a)	50.50	27.23	49.32
BART-xsum (Lewis et al., 2019)	51.74	26.46	48.72
BART-xsum + Sketch (Ours)	51.79	26.85	49.15
BART-xsum + Ctrl (Ours)	**52.84**	27.35	50.29
CODS (Ours)	52.65	**27.84**	**50.79**

Table 1: Dialogue summarization ROUGE evaluation on the SAMSum test set. Results with * are obtained from Gliwa et al., 2019. CODS achieves the highest ROUGE score. *BART-xsum + Sketch* and *BART-xsum + Ctrl* are ablated models individually removing controllability and sketch generation component from CODS.

	ROUGE_WE	BERTScore	MoverScore	BLEU	CIDEr	SMS
PEGASUS	0.3562	0.5335	0.3233	17.33	1.741	0.1608
BART-xsum	0.3606	0.5387	0.3391	17.55	1.701	0.1401
CODS	**0.3759**	**0.5458**	**0.3539**	**19.58**	**1.981**	**0.1689**

Table 2: Dialogue summarization evaluation on the SAMSum test set with additional recently introduced metrics that have been applied to both text generation and summarization.

3 Experiments

3.1 Dataset

We perform experiments on the recently released SAMSum dataset (Gliwa et al., 2019) [2], which is the most comprehensive resource for abstractive dialogue summarization tasks. It contains 16K natural messenger-like dialogues created by linguists fluent in English with manually annotated summaries. This dataset is more challenging than the previous corpus (McCowan et al., 2005) in the following aspects: 1) Unlike previous datasets consisting of only hundreds of dialogue-summary pairs, it has larger data size (16369 samples); 2) 75% of the conversations are between two interlocutors, the rest are between three or more people; 3) the conversations cover diverse real-life topics, and the summaries are annotated with information about the speakers. We preprocess the data by the following steps: 1) concatenate adjacent utterances of the same speaker into one utterance; 2) clean the dialogue text by removing hashtags, URLs and Emojis; 3) label each utterance with its corresponding interrogative pronoun category with a weak supervision approach (Ratner et al., 2019); 4) parse each utterance with a constituency parser and find the longest common sub-sequence between the phrases and summary to be the key phrases.

3.2 Evaluation Metrics and Baselines

We use the standard ROUGE metric (Lin, 2004) as automatic evaluation metrics, including ROUGE-1, ROUGE-2, and ROUGE-L. Following previous work (Gliwa et al., 2019), we use py-ROUGE[3] library with stemming. We compare our model with baselines reported in Gliwa et al., 2019: Longest-3 is a commonly-used extractive summarization baseline which takes the top three longest sentences as summary. The pointer generator and Fast abs are RNN-based methods with copy-attention mechanism or policy gradient. The Transformer is a random-initialized self-attention architecture with multi-head attention. The DynamicConv is a

---

[2] The conversations in SAMSum may contain offensive words, please use the dataset carefully.

[3] `pypi.org/project/pyROUGE/`

lightweight convolutional model that can perform competitively to self-attention. All of these models are not pre-trained.

Besides, we investigate four pre-trained generative language models to see which works the best for the dialogue summarization task. DialoGPT is a GPT model pre-trained on open-domain Reddit data. UniLM is pre-trained using three types of language modeling tasks: unidirectional, bidirectional, and sequence-to-sequence prediction on English Wikipedia and BookCorpus. PEGASUS masks important sentences from input and is trained to generate the missing parts, similar to an extractive summary approach. BART is trained by corrupting text with an arbitrary noising function and learning to reconstruct the original text. We use default parameters listed in the respective open-source repositories to fine-tune on the dialogue summarization task. We show the training details in the Appendix.

3.3 Results

In Table 1 of ROUGE results, we find that the methods that are pre-trained or with pre-trained embeddings perform better than those that are not. For instance, DynamicConv achieves a 3 – 4% improvement by adding GPT-2 embeddings. This further confirms the impact of language model pre-training on downstream tasks. Among the pre-trained generative language models examined, PEGASUS and BART are the two top performance models with ROUGE-1 higher than 50. DialoGPT, the model pre-trained on conversational data, does not achieve satisfactory results, implying that Reddit data has limited knowledge to be transferred to dialogue summarization tasks. CODS achieves the highest ROUGE score compared with other models, notably 50.79% ROUGE-L.

To understand the individual contribution of each component in our model, we also conduct an ablation study by removing summary sketch generation (BART+Ctrl) or controllability (BART+Sketch). In both cases we observe a performance drop, except a slight improvement on ROUGE-1 for BART+Ctrl. This suggests that the sketching step helps generate a more fluent summary even with lower unigram matching. Furthermore, recognizing the limitation of ROUGE scores in their ability to fully capture the resemblance between the generated summary and the reference, in Table 2, we follow (Fabbri et al., 2020) to compare model performances with additional metrics, including ROUGE-Word Embedding (Ng and Abrecht, 2015), BERTScore (Zhang et al., 2019b), MoverScore (Zhao et al., 2019), Sentence Mover's Similarity (SMS) (Clark et al., 2019), BLEU (Papineni et al., 2002), and CIDEr (Vedantam et al., 2015). As shown in Table 2, CODS consistently outperforms PEGASUS and BART. More information about these evaluation metrics are shown in the Appendix.

	Length Ratio	Consistent	Informative
Longest-1	0.27	**0.70**	0.23
BART-xsum-1	**0.16**	0.50	0.16
CODS-1	0.19	0.50	**0.49**
BART-xsum	0.26	0.65	0.51
CODS	**0.24**	**0.69**	**0.53**
Gold	0.27	0.74	0.55

Table 3: Human evaluation results on test set for both controllable summary and standard summary.

3.4 Analysis

3.4.1 Human Evaluation by Crowdsourcing

We leverage human judgement to evaluate the generated summaries via crowdsourcing, especially for granularity-controlled generation, since we do not have human-written reference summaries of various lengths (number of sentences). We ask workers to rate the summaries in two aspects on a scale from -1 (worst) to 1 (best): factual consistency and informativeness. *Factual consistency* acts as a precision measure, assessing whether the information provided in summary contains factual errors which are against the source dialogue; *Informativeness* is a recall-oriented measure, examining whether critical information in a dialogue is mentioned in summary. We also show the length ratio between a summary and a dialogue, where a lower ratio means a higher compression rate. For the crowdsourcing evaluation, we randomly select 6% dialogues from the test set, each of which is annotated by three workers. More details about human evaluation process are in the Appendix [4].

To show the proposed controllable generation's strengthens and quality, we provide two additional baselines, Longest-1 and BART-1. The longest-1 method is an extractive baseline that outputs the longest dialogue turn as the final summary. The BART-1 is a strong abstractive baseline where we train a BART-based summarization model with the

---

[4] The prediction file on the test set is provided in the supplementary file.

<table>
<tr><td colspan="2">Kelly: I still haven't received the rent money. Did you check with your bank?
John: Yes. I definitely sent it last week.
Kelly: But I still don't have it. Can you please check that you sent it to the right account.
John: Ok. Give me 5 min.
Kelly: OK
John: I checked and the money did go out of my account last week.
Kelly: What account number did you send it to?
John: 44-1278
Kelly: No wonder! My account number is 44-1279. You sent it to someone else's account.
John: ...! I'm really sorry!
Kelly: I still need the rent money though.
John: I'm really sorry I'll have to go to the bank tomorrow and ask if they can re-send it to the right account.
Kelly: Thanks !</td></tr>
<tr><td>Longest-1</td><td>John said I'm really sorry I'll have to go to the bank tomorrow and ask if they can re-send it to the right account.</td></tr>
<tr><td>BART-1</td><td>Kelly still hasn't received the rent money from John.</td></tr>
<tr><td>CODS-1</td><td>John sent the rent money to the wrong account and will have to ask the bank to re-send it to the correct one tomorrow.</td></tr>
<tr><td>BART</td><td>Kelly still hasn't received the rent money. John sent it to the wrong account number 44-1278. John will go to the bank tomorrow and ask if they can re-send the money to the right account.</td></tr>
<tr><td>CODS</td><td>Sketch: 1 #confirm haven't received the rent money check with your bank 2 none 3 #confirm check that you sent it to the right account 4 none 5 none 6 #abstain the money did go out of my account last week 7 #abstain did you send it to 8 none 9 #what sent it to someone else's account 10 none 11 #abstain need the rent money though 12 #abstain 'm really sorry i'll have to go to the bank tomorrow and ask if they can re-send it to the right account 13 none
Summary: John sent the rent money to the wrong account last week. John will go to the bank tomorrow and ask if he can re-send the money to the correct account.</td></tr>
<tr><td>Gold</td><td>Kelly hasn't received the rent money, because John sent it to the wrong bank account. He will go to the bank to tackle the issue.</td></tr>
</table>

Table 4: A test set example with generated summaries.

	Reference Summary	CODS Summary
Associate names with actions	Lilly will be late. Gabriel will order pasta with salmon and basil for her.	Lilly will be late for the meeting with Gabriel. Gabriel will order something for Lilly.
	Ann doesn't know what she should give to her dad as a birthday gift. He's turning 50. Fiona tries to help her and suggests a paintball match.	It's Ann's dad's 50th birthday. He's turning 50. Ann and Fiona are planning a surprise birthday party for her dad.
Extract information after the discussion	Paul will buy red roses following Cindy's advice.	Paul wants to buy red roses.
Decide important information	Rachel's aunt had an accident and she's in hospital now. She's only bruised. The perpetrator of the accident is going to pay for the rehabilitation.	Rachel is at the hospital with her aunt, who had an accident. She's bruised but fine. She will give her a hug.
	Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.	Amanda can't find Betty's number. Amanda suggests to text him.

Table 5: Case analyses by manually examining CODS generated summaries.

number of summary sentences in the training set as its start-of-sentence token during decoding. Similar to the approach from Liu et al., 2018, we can use different start-of-sentence tokens to control the BART output.

In general, it is preferable to have a factually consistent and informative summary that is succinct (low length ratio, high compression rate) at the same time. As shown in the first row of Table 3, CODS-1 achieves the highest informative score among all generated one-sentence summaries, indicating the strength of the proposed controllable method in producing succinct yet informative dialogue summaries. The Longest-1 method has a higher consistent score because its summary is directly copied from the original dialogue, preventing any factual mistakes. The second row of Table 3 shows that CODS, when automatically determining the granularity of the summary, produces summaries that are more succinct (lower length ratio), more factually consistent, and more informative, compared to the BART model.

3.4.2 Case Study

CODS outperforms the baseline models in both ROUGE scores and human evaluation metrics. We now further inspect its textual quality. In Table 4, we show an example from the SAMSum test set with summaries generated by different models. In this example, CODS and CODS-1 can both produce a near-perfect summary even compared to the human-written reference summary. On the other hand, the summary generated by BART includes overly detailed information (e.g., bank account). We show some more examples in the Appendix and all the predictions (including CODS-1 and CODS-2) in the supplementary file.

We also manually examine 100 summaries generated from CODS against the reference summaries in the test set. Specifically, we analyze each of the three following problematic cases, where summarization models frequently make mistakes, reported by Gliwa et al., 2019, and provide sample summaries in Table 5. 1) *Associating names with actions*: CODS performs well in dealing with speakers' names. It accurately associates "her dad" with "Ann's dad," also "Fiona tries to help her" with "Ann and Fiona." 2) *Extract information about the arrangement after discussion*: Even speakers hesitate about the flower's color to be yellow, pink or red in the middle of the discussion, CODS still correctly determines the right color after several turns. 3) *Decide important information in dialogues*: CODS fails to capture some of the important facts (marked as red) mentioned in reference summary. We conjecture the reason could be that 1) some of the important facts are located in the same part of the highlighted turns, and 2) those information is missed by the key phrase extraction. Simultaneously, we force the model to generate only the most important one under the constraint of controllability. The improvement of CODS on the first two summarization difficulties can be partially attributed to the clear logic in the sketch when input to the model.

4 Related Work

Neural Text Summarization There are two main paradigms for text summarization: extractive and abstractive. Inspired by the success of applying seq2seq models on neural machine translation, Rush et al., 2015 and Nallapati et al., 2016 introduce the neural seq2seq model on abstractive text summarization, with an attention-based encoder and a neural language model decoder. To solve the problem of out-of-vocabulary words and to capture salient information in source documents, See et al., 2017 propose a pointer-generator network that copy words from source to target. Many subsequent works (Gehrmann et al., 2018; Paulus et al., 2018) further demonstrate its effectiveness with reinforcement learning. Recently, Liu and Lapata, 2019 apply BERT on text summarization and propose a general framework for both extractive and abstractive models. Zhang et al., 2019c pre-train hierarchical document encoder for extractive summarization. Lewis et al., 2019 introduces BART, a denoising autoencoder for pretraining sequence-to-sequence models. BART significantly outperforms the best previous work in terms of ROUGE metrics.

Dialogue Summarization Regarding to the datasets in dialogue summarization, initial abstractive dialogue summarization work (Oya et al., 2014; Mehdad et al., 2014; Banerjee et al., 2015) are conducted on the AMI meeting corpus (McCowan et al., 2005), with only 141 summaries. Goo and Chen, 2018 propose to use the topic descriptions (high-level goals of meetings) in AMI as reference summaries and use dialogue acts as training signals. Pan et al., 2018 build the Dial2Desc dataset by reversing a visual dialogue task, aligning image dialogues with the image caption as a summary. Liu et al., 2019 collect their dataset from the logs in the DiDi customer service center. It is restricted to task-oriented scenario, where one speaker is the user and the other is the customer agent, with limited topics and it is also connected to the goal of dialogue state tracking task (Wu et al., 2019a, 2020b). Recently, Gliwa et al., 2019 introduce the SAMSum corpus, with 16k chat dialogues with manually annotated summaries. It is the first comprehensive abstractive dialogue summarization dataset spanning over various lengths and topics. Chen and Yang, 2020 propose a multi-view sequence-to-sequence model by extracting different views of structures from conversations. Both their method and ours leverage rich conversation structure information. Evaluating on SAMSum, our model CODS outperform theirs by 3 points in terms of ROUGE scores, indicating our utilized dialogue features are more effective.

Length-controllable Generation The most prevalent method for length control generation is using a special length embedding. Kikuchi et al., 2016 first propose length control for abstractive summarization by using length embedding as an additional input for the LSTM. Fan et al., 2018 train embeddings that correspond to each different output length and prepend that length marker at the beginning of the decoder. Liu et al., 2018 incorporates the length embedding into initial state of a CNN-based decoder. Takase and Okazaki, 2019 extends the positional encoding in

Transformer model by considering the remaining length explicitly at each decoding step. Saito et al., 2020 propose to control the summary length with prototype extractor. However, the retrieve-and-rewrite process is restricted by the extraction quality, leaving its performance limited by extractive solutions' capabilities. The aforementioned works all focus on structured text summarization (e.g. news document). We are the first to propose generate length-controllable summary on dialogues by highlighting arbitrary numbers of dialogue spans.

5 Conclusion

The dialogue summarization task is challenging but with vast application potential. We propose CODS, a state-of-the-art dialogue summarization model with granularity controllability. CODS uses a weakly-labeled summary sketch for its two-stage generation, and text-span conditional generation for a controllable summary. Our model surpasses existing models on the largest dialogue summarization dataset. We show with human evaluation that our model can generate factually consistent and informative summaries. We also point out several error cases to shed light on future research direction of controllable dialogue summarization.

References

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Abstractive meeting summarization using dependency graph fusion. In *Proceedings of the 24th International Conference on World Wide Web*, pages 5–6.

Jiaao Chen and Diyi Yang. 2020. Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4106–4118.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. Summeval: Re-evaluating summarization evaluation. *arXiv preprint arXiv:2007.12626*.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54.

Xiachong Feng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2020. Incorporating commonsense knowledge into abstractive dialogue summarization via heterogeneous graph networks. *arXiv preprint arXiv:2010.10044*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.

Chih-Wen Goo and Yun-Nung Chen. 2018. Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 735–742. IEEE.

Geoff Hart. The five w's of online help systems, year = 2002, url = http://www.geoff-hart.com/articles/2002/fivew.htm, urldate = 2002.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Chunyi Liu, Peng Wang, Jiang Xu, Zang Li, and Jieping Ye. 2019. Automatic dialogue summary generation for customer service. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1957–1965.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4110–4119.

I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, W. Post, Dennis Reidsma, and P. Wellner. 2005. The ami meeting corpus. In *Proceedings of Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research*, pages 137–140. Noldus Information Technology.

Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. Abstractive summarization of spoken and written conversations based on phrasal queries. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1220–1230.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930, Lisbon, Portugal. Association for Computational Linguistics.

Nikola I Nikolov, Michael Pfeiffer, and Richard HR Hahnloser. 2018. Data-driven summarization of scientific articles. *arXiv preprint arXiv:1804.08875*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 45–53.

Haojie Pan, Junpei Zhou, Zhou Zhao, Yan Liu, Deng Cai, and Min Yang. 2018. Dial2desc: end-to-end dialogue description generation. *arXiv preprint arXiv:1811.00185*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Vahed Qazvinian and Dragomir R Radev. 2008. Scientific paper summarization using citation summary networks. *arXiv preprint arXiv:0807.1560*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, Atsushi Otsuka, Hisako Asano, Junji Tomita, Hiroyuki Shindo, and Yuji Matsumoto. 2020. Length-controllable abstractive summarization by guiding with summary prototype. *arXiv preprint arXiv:2001.07331*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3999–4004.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019a. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019b. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020a. Tod-bert: Pre-trained natural language understanding for task-oriented dialogues. *arXiv preprint arXiv:2004.06871*.

Chien-Sheng Wu, Steven C.H. Hoi, and Caiming Xiong. 2020b. Improving limited labeled dialogue state tracking with self-supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4462–4472, Online. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019a. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Chien-Sheng Wu and Caiming Xiong. 2020. Probing task-oriented dialogue representation from language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5036–5051, Online. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019b. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3283–3294.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Klaus Zechner. 2001. Automatic generation of concise summaries of spoken dialogues in unrestricted domains. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–207.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. *arXiv preprint arXiv:1808.07187*.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019c. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019d. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

Lulu Zhao, Weiran Xu, and Jun Guo. 2020. Improving abstractive dialogue summarization with graph structures and topic words. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 437–449, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.

A Appendix

A.1

We define dialogue intent categories as follows: *WHY:* asks the reason of the state mentioned in the previous turn, e.g., “why” or “why not”; *WHAT:* requests more details about the aforementioned object, the sentence usually starts with “what’s” or “what about”; *WHERE:* the location of the event; *WHEN:* the time of the event, e.g. ,“when” or “what time”; *CONFIRM:* expects the other speaker to establish the correctness of a certain case, the sentence usually starts with patterns like “are you”, “will you” or “has he”; *ABSTAIN:* the utterance does not belong to any of the previous categories. It occurs when speakers continue to state or comment without seeking for more information from the others.

A.2

- DialoGPT: A GPT model pretrained on 147M conversation-like data extracted from Reddit comments. We use the model with 117M parameters. `github.com/microsoft/DialoGPT`
- UniLM: A multi-layer Transformer network optimized for three language modeling objectives: unidirectional, bidirectional and sequence-to-sequence prediction. It is initialized with BERT$_{\text{LARGE}}$, then pre-trained using English Wikipedia and BookCorpus. Same as BERT$_{\text{LARGE}}$, it contains 340M parameters. `github.com/microsoft/unilm`
- PEGASUS: They pretrain a Transformer-based encoder-decoder models with a new self-supervised objective - gap-sentence generation - on the C4 corpus. We use the PEGASUS of 568M parameters. `github.com/google-research/pegasus`
- BART: Transformer-based encoder-decoder model trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text. We use BART$_{\text{LARGE}}$ model which contains 400M parameters. `huggingface.co/transformers/model_doc/bart.html`
- CODS: It’s based on BART$_{\text{LARGE}}$ model which contains 400M parameters.

<table>
<tr><td colspan="2">Keith: Meg, pls buy some milk and cereals, I see now we’ve run out of them .
Megan: hm, sure, I can do that .
Megan: but did you check in the drawer next to the fridge ?
Keith: nope, let me have a look .
Keith: ok, false alarm, we have cereal and milk .
Deana: glad to hear it !</td></tr>
<tr><td>Summary</td><td>Megan needn’t buy milk and cereals. They’re in the drawer next to the fridge.</td></tr>
</table>

Table 6: Example key phrases in summary sketch.

<table>
<tr><td colspan="2">Norbert: we need to hurry to catch the tour .
Wendy: ok , am buying something . be right out !
Norbert: ok . am not waiting long though . missed the last one because of you .
Wendy: just be patient for once .
Norbert: im always patient .
Wendy: at the register now .
Norbert: alright .</td></tr>
<tr><td>Summary</td><td>Wendy is shopping, but she needs to hurry up to catch the tour.</td></tr>
</table>

Table 7: Example key phrases in summary sketch.

A.3 Sketch Construction

Previous methods (Goo and Chen, 2018; Pan et al., 2018) heavily rely on explicit intent annotations in datasets. We label user intent automatically for each utterance with the Snorkel library in a weak supervision approach. For each interrogative pronoun category, we first manually identify its most frequent key words and patterns (can be found in our source code). Then we use the labeling functions in Snorkel to label all the utterances.

For the utterance compression, we do LCS on the phrases generated from the constituency parser. In the example of 1, *s just one of many boring days at work* the parsed constituent overlapping with ‘at work’ in the summary, so we keep this phrase. However, in other examples, not all overlapped words are meaningful (e.g. stop words). We thus filter the LCS results and only keep important key phrases. Then we train our model to predict these key phrase spans in each turn. We show three examples of our generated key phrases in summary sketches on evaluation set (see Table 6, 7, 8)

A.4 Training Details

We use huggingface (Wolf et al., 2019a) implementation to fine-tune a BART model. We use the large

<table>
<tr><td colspan="2">Phil: is brandon in ?
Clara: not yet .
Phil: has he called to say he’d be late ?
Clara: no , he hasn’t .
Phil: it’s not the first time , ist it ?
Clara: no , it isn’t .
Phil: when he arrives , tell him to come to me .
Clara: no , it isn’t .
Phil: please prepare a report on the absenteeism and lateness . i expect it by friday on my desk .
Clara: it will be ready .</td></tr>
<tr><td>Summary</td><td>Brandon is late again. Clara will prepare a report on the absenteeism and lateness for Phil by Friday.</td></tr>
</table>

Table 8: Example key phrases in summary sketch.

version fine-tuned on the XSUM (Narayan et al., 2018) dataset with 12 self-attention encoder and decoder layers. We truncate input dialogue to a maximal length 512 with training batch size 4. We train the model with Adam optimizer (Kingma and Ba, 2014) with 0.1 proportion for linear learning rate warmup. We early stop on validation set ROUGE-1 score, and it is trained for around 40,000 steps on one NVIDIA V100 GPU. During inference, we do beam search decoding with beam size 4.

A.5 Evaluation Metrics

Information obtains from (Fabbri et al., 2020):

- ROUGE measures the number of overlapping textual units between the generated summary and a set of reference summaries.
- ROUGE-WE extends ROUGE by taking cosine similarity of Word2Vec embeddings into account.
- BERTScore computes similarity scores by aligning generated and reference summaries on a token-level based on the output of the BERT-based model. Token alignments are computed greedily with the objective of maximizing the cosine similarity between contextualized token embeddings. We report the F1 score.
- MoverScore measures semantic distance between a summary and reference text by making use of the Word Mover's Distance operating over n-gram embeddings pooled from BERT representations.
- Sentence Mover's Similarity (SMS) extends Word Mover's Distance to view documents as a bag of sentence embeddings as well as a variation which represents documents as both a bag of sentences and a bag of words.
- BLEU is a corpus-level precision-focused metric which calculates n-gram overlap between a candidate and reference utterance and includes a brevity penalty. It is the primary evaluation metric for machine translation.
- CIDEr computes 1-4-gram co-occurrences between the candidate and reference texts, down-weighting common n-grams and calculating cosine similarity between the ngrams of the candidate and reference texts.

A.6 Human Evaluation

We use roughly 6% of the test set data in SAMSum for human evaluation and we do some filtering based on the annotation of the "gold summary". Specifically, we filter those annotations if a "gold summary" has been annotated as "-1" (the meaning of each score is shown below), implying that the annotators may not pay attention to the scoring. The final results reported in Table 3 is the mean from three different annotators.

The "gold summary" is actually not perfect and it might contain some noisy annotation, this is the reason why some workers may give 0 even if it is collected from humans. Below is the scoring instruction we sent to our workers:

- Factual Consistency (Precision): The rating measures whether the information provided in a summary is correct. Score -1 if a summary contains a serious factual error. Score 0 if a summary has some minor factual errors. Score 1 if everything in a summary is factually correct.
- Informative (Recall): The rating measures whether all the important information in a dialogue is included in a summary. Score -1 if a summary misses serious key points. Score 0 if a summary misses a few key points. Score 1 if a summary covers all key points.

<table>
<tr><td colspan="2">Paul: what color flowers should i get
Cindy: any just not yellow
Paul: ok , pink ?
Cindy: no maybe red
Paul: just tell me what color and what type ok ?
Cindy: ugh , red roses !</td></tr>
<tr><td>Gold</td><td>Paul will buy red roses following Cindy’s advice.</td></tr>
<tr><td>BART</td><td>Paul wants to get red roses. Cindy doesn’t want pink or yellow roses.</td></tr>
<tr><td>CODS</td><td>Paul wants to buy red roses.</td></tr>
</table>

Table 9: Dialogue for the ”Extract information after the discussion” sample in Table 5

<table>
<tr><td colspan="2">Phil: good evening deana ! many thanks for this nice card from you . constantine was very happy !. are these sunglasses also from you ?
Deana: i thought they belonged your cathreen !
Phil: nope . she says they aren’t hers .
Deana: mine either . look , maybe you feel like keeping them ?. i seem to have so many sunglasses .. 8
Phil: where did you find them , possible that they belong to adrian ?
Deana: in this empty place above the radio . in the very back .. if adrian wants it , no pro !. exactly !
Phil: ok , they don’t belong to any of us , and nobody else drove your car . but we can look after these sunglasses .
Deana: glad to hear it !</td></tr>
<tr><td>Longest-1</td><td>Phil said good evening deana ! many thanks for this nice card from you . constantine was very happy !. are these sunglasses also from you ?</td></tr>
<tr><td>BART-1</td><td>Phil and Deana will look after Adrian’s sunglasses.</td></tr>
<tr><td>CODS-1</td><td>Deana found Adrian’s sunglasses in the back of Phil’s car.</td></tr>
<tr><td>BART</td><td>Phil and Deana are going to look after Adrian’s sunglasses.</td></tr>
<tr><td>CODS</td><td>Phil got a card from Deana. Deana found them in the empty place above the radio. Deana has a lot of them.</td></tr>
<tr><td>Gold</td><td>Phil received a card from Deana. Constantine was happy. Phil has sunglasses, that Deana found in the back above the radio. Deana and Phil don’t know who they belong too. Phil will keep the sunglasses.</td></tr>
</table>

Table 10: Test set example for qualitative study.

<table>
<tr><td colspan="2">Celia: where do you want to go for holiday ?
Mike: i was thinking about egypt
Celia: too hot . what about croatia ?
Mike: good idea , i’ve never been there</td></tr>
<tr><td>Longest-1</td><td>Celia said where do you want to go for holiday ?</td></tr>
<tr><td>BART-1</td><td>Mike wants to go for holiday to Egypt.</td></tr>
<tr><td>CODS-1</td><td>Mike wants to go on holiday to Egypt or Croatia.</td></tr>
<tr><td>BART</td><td>Celia and Mike will go for holiday to Croatia.</td></tr>
<tr><td>CODS</td><td>Mike wants to go on holiday to Egypt. Celia thinks it’s too hot. Mike has never been to Croatia, but he likes the idea.</td></tr>
<tr><td>Gold</td><td>Mike considers going to Egypt for holiday. It’s too hot for Celia, she suggests Croatia instead. Mark likes the idea, he’s never been there.</td></tr>
</table>

Table 11: Test set example for qualitative study.

<table>
<tr><td colspan="2">Diane: how long do you have to work tonight ?
Ross: about 2 hours , why ?
Diane: i just wanted to do something maybe
Ross: i think i’ll be worn out after all hat work , baby
Diane: we can just chill at home , don’t worry. i just wanted to prepare
Ross: ok. then just to be safe let’s say it will take me 3 hours
Diane: but you just said 2 !
Ross: ... , Diane , don’t start again
Diane: what am i starting !. you’re impossible
Ross: can’t you understand that this is important to me !. my career depends on it !
Diane: well , if your career is the most important thing in the world then i wouldn’t want to disturb !</td></tr>
<tr><td>Longest-1</td><td>Diane said well , if your career is the most important thing in the world then i wouldn’t want to disturb !</td></tr>
<tr><td>BART-1</td><td>Ross has to work for 2 hours tonight.</td></tr>
<tr><td>CODS-1</td><td>Ross has to work 3 hours tonight.</td></tr>
<tr><td>BART</td><td>Ross has to work tonight for 2 hours. Ross and Diane will chill at home.</td></tr>
<tr><td>CODS</td><td>Ross has to work 3 hours tonight.</td></tr>
<tr><td>Gold</td><td>Diane is not happy with Ross prioritising work over spending time with her.</td></tr>
</table>

Table 12: Test set example for qualitative study.

Elaborative Simplification: Content Addition and Explanation Generation in Text Simplification

Neha Srikanth
Department of Computer Science
The University of Texas at Austin
nehasrik@utexas.edu

Junyi Jessy Li
Department of Linguistics
The University of Texas at Austin
jessy@austin.utexas.edu

Abstract

Much of modern-day text simplification research focuses on sentence-level simplification, transforming original, more complex sentences into simplified versions. However, *adding* content can often be useful when difficult concepts and reasoning need to be explained. In this work, we present the first data-driven study of content addition in text simplification, which we call *elaborative simplification*. We introduce a new annotated dataset of 1.3K instances of elaborative simplification in the Newsela corpus, and analyze how entities, ideas, and concepts are elaborated through the lens of contextual specificity. We establish baselines for elaboration generation using large-scale pre-trained language models, and demonstrate that considering contextual specificity during generation can improve performance. Our results illustrate the complexities of elaborative simplification, suggesting many interesting directions for future work.

1 Introduction

Text simplification aims to help audiences read and understand a piece of text through lexical, syntactic, and discourse modifications, while remaining faithful to its central idea and meaning (Siddharthan, 2014). It remains an important task, improving text accessibility for children (De Belder and Moens, 2010; Kajiwara et al., 2013), language learners (Yano et al., 1994; Petersen and Ostendorf, 2007; Pellow and Eskenazi, 2014; Paetzold, 2016), and those with language impairments (Carroll et al., 1998; Rello et al., 2013). Text simplification can also be a useful pre-processing step for other NLP tasks such as machine translation (Chen et al., 2012; Štajner and Popovic, 2016) and summarization (Vanderwende et al., 2007; Silveira and Branco, 2012).

Original Text

Results, she said, "could help the team better understand ancient Egyptian health" and, correspondingly, modern-day health. For instance, some mummies still have arteries in their mummified remains, Miller-Thomas said. And, sometimes, scientists can tell if those arteries had hardened.

Simplified Text

The scans could help the team understand about ancient Egyptians' health. For example, some mummies still have arteries. **An artery is a tube that moves blood through the body.** **The artery could show if the person had been healthy or not.**

Figure 1: Elaborative simplification with two elaborations of varying contextual specificity.

With the introduction of large, parallel corpora (Zhu et al., 2010; Woodsend and Lapata, 2011; Coster and Kauchak, 2011; Xu et al., 2015), text simplification research has rapidly advanced in recent years, especially in sentence simplification (Alva-Manchego et al., 2020). However, document simplification involves rich linguistic phenomena that cannot be easily characterized by sentence-level transformations of text, e.g., the omission and addition of content (Petersen and Ostendorf, 2007; Siddharthan, 2014).

This paper presents the first data-driven, dedicated study of *elaborative simplification*, which involves inserting elaborations in the form of definitions, explanations or clarifications to improve readability by providing readers with necessary additional context. Effective elaborations must provide background in a *contextual* manner, adding relevant information to the surrounding text.

Figure 1 shows an example. The original text snippet explains that scientists study mummy arteries to see whether they are hardened. In the corresponding simplified text, we see two elaborations inserted – one, in green, simply defines an artery, and the second, in blue, states the implication of hardened arteries. The content of both elaborations is semantically absent from the original text.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5123–5137
August 1–6, 2021.

Our goal is to provide resources and directions toward understanding and generating naturally occurring elaborations. We present an annotated dataset of 1.3K instances of elaborative simplification in the Newsela corpus (Xu et al., 2015). We automatically identify candidate elaborations from simplified documents, and have human annotators verify candidates. We find that many elaborations require multi-hop reasoning, inference, commonsense reasoning, and relevant information retrieval, making it an interesting testbed for a bevy of related tasks.

The previous example highlights two elaborations on opposite ends of the spectrum – the first requires little context, while the second is highly contextualized, drawing a conclusion from content presented in the original text. To this end, we characterize elaborations by annotating their *contextual specificity*, i.e., the extent to which the added content is specific to the current topic under discussion.

We reveal that our dataset contains a fairly balanced distribution of contextual specificity. Qualitatively, while inserting definitions may help provide background about entities, highly contextualized elaborations interpreting or clarifying content can help readers understand the larger implications or significance of ideas presented in the original text. We propose the primary task of generating elaborations given document context. We present baselines for elaboration generation mainly using GPT-2 (Radford et al., 2019), and discuss some of the challenges, especially with respect to the contextual specificity of added content.

We find that generation quality can be improved by selecting an elaboration with an appropriate predicted contextual specificity level. However, existing methods struggle to effectively incorporate input context to generate elaborations. We hope that this study will motivate advancement in elaborative simplification.

In summary, our main contributions include:

1. Introduction of elaborative simplification, a previously understudied phenomenon in text simplification;
2. A new, annotated dataset of 1.3K naturally occurring elaborations in the Newsela corpus and their contextual specificity;
3. Analysis of the challenges of elaborative simplification for pre-trained language models through performance of our baselines.

We make our annotations and code available at `https://github.com/nehasrikn/elaborative-simplification`.

2 Data and Annotation

Elaborative simplification involves the *insertion* of content to make simplified text easier to understand. We present an annotated dataset of 1.3K elaborations from the Newsela corpus (Xu et al., 2015), which contains English news articles manually simplified by professional editors. We describe the scope of our elaborative simplification study (§2.1), strategies for trusted annotators to extract elaborations (§2.2) and rate contextual specificity (§2.3), and scaling up annotation through crowdsourcing with rigorous quality control (§2.4).

2.1 What is an elaboration?

We consider a sentence an elaboration if it contains new content (e.g. statements about entities, actions, or concepts) present in the simplified document, but semantically missing from the original document. Note that while elaborations can contain multiple sentences, we define our label at the sentence level. Past simplification research has focused on operations such as substitution and deletion, but simplifying a piece of text that may contain unknown or difficult concepts could involve inserting simple explanations as well. As we highlight in §6, others have shown that audiences such as new language learners benefit from elaboration or explanation insertion (and conversely, that unfamiliar concepts negatively impact reading comprehension), though computational approaches till date have been largely limited to definition retrieval.

Scope. We intentionally choose to study *how* concepts are elaborated, posing a scenario where an author has the freedom to specify where to elaborate, and our system generates an appropriate elaboration. We do this for two main reasons: first, understanding how to elaborate can be utilized in a system where users specify what to elaborate on, in the spirit of personalized simplification (Paetzold and Specia, 2016; Bingel et al., 2018). Second, determining *when* to elaborate is arguably pragmatically more complex, in that the need for elaboration often relies on the writer's belief about their readers' background, knowledge, and reading ability, as well as their own judgments on how often to elaborate. For example, in the extreme case, inserting an elaboration after every sentence could prove

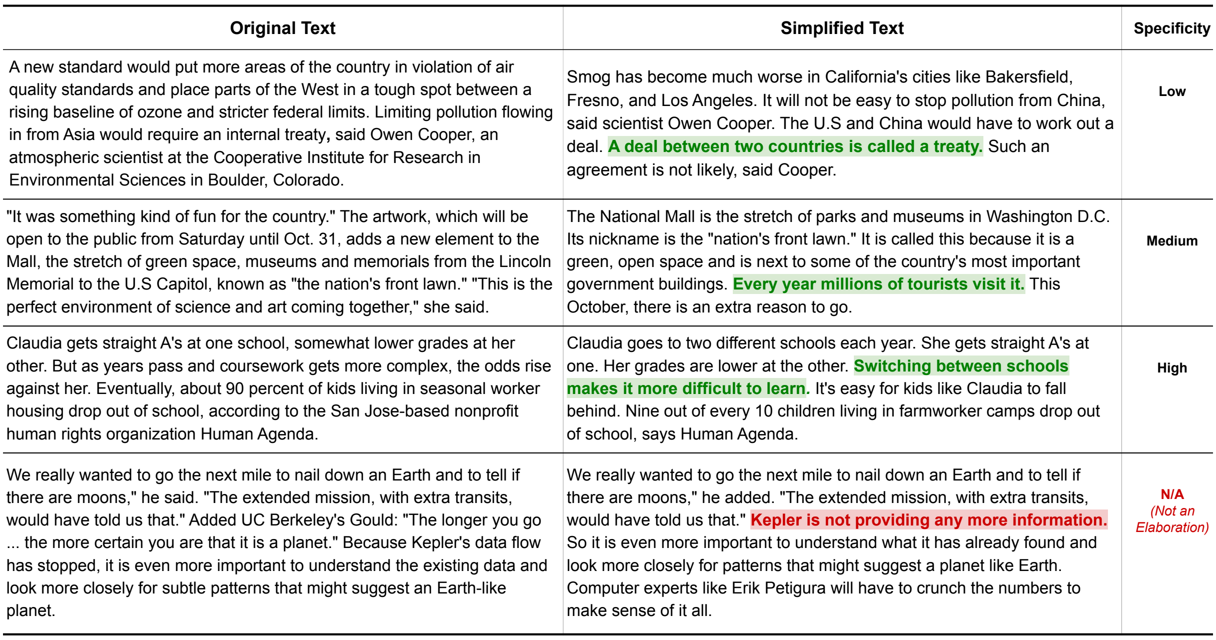

Original Text	Simplified Text	Specificity
A new standard would put more areas of the country in violation of air quality standards and place parts of the West in a tough spot between a rising baseline of ozone and stricter federal limits. Limiting pollution flowing in from Asia would require an internal treaty, said Owen Cooper, an atmospheric scientist at the Cooperative Institute for Research in Environmental Sciences in Boulder, Colorado.	Smog has become much worse in California's cities like Bakersfield, Fresno, and Los Angeles. It will not be easy to stop pollution from China, said scientist Owen Cooper. The U.S and China would have to work out a deal. **A deal between two countries is called a treaty.** Such an agreement is not likely, said Cooper.	**Low**
"It was something kind of fun for the country." The artwork, which will be open to the public from Saturday until Oct. 31, adds a new element to the Mall, the stretch of green space, museums and memorials from the Lincoln Memorial to the U.S Capitol, known as "the nation's front lawn." "This is the perfect environment of science and art coming together," she said.	The National Mall is the stretch of parks and museums in Washington D.C. Its nickname is the "nation's front lawn." It is called this because it is a green, open space and is next to some of the country's most important government buildings. **Every year millions of tourists visit it.** This October, there is an extra reason to go.	**Medium**
Claudia gets straight A's at one school, somewhat lower grades at her other. But as years pass and coursework gets more complex, the odds rise against her. Eventually, about 90 percent of kids living in seasonal worker housing drop out of school, according to the San Jose-based nonprofit human rights organization Human Agenda.	Claudia goes to two different schools each year. She gets straight A's at one. Her grades are lower at the other. **Switching between schools makes it more difficult to learn.** It's easy for kids like Claudia to fall behind. Nine out of every 10 children living in farmworker camps drop out of school, says Human Agenda.	**High**
We really wanted to go the next mile to nail down an Earth and to tell if there are moons," he said. "The extended mission, with extra transits, would have told us that." Added UC Berkeley's Gould: "The longer you go ... the more certain you are that it is a planet." Because Kepler's data flow has stopped, it is even more important to understand the existing data and look more closely for subtle patterns that might suggest an Earth-like planet.	We really wanted to go the next mile to nail down an Earth and to tell if there are moons," he added. "The extended mission, with extra transits, would have told us that." **Kepler is not providing any more information.** So it is even more important to understand what it has already found and look more closely for patterns that might suggest a planet like Earth. Computer experts like Erik Petigura will have to crunch the numbers to make sense of it all.	**N/A** *(Not an Elaboration)*

Figure 2: Example candidate elaborations. Rows 1–3 contain verified elaborations. Row 4 contains a rejected candidate. We include the original and simplified text regions, highlighting the candidate elaboration, and its corresponding level of contextual specificity in Column 3.

useful for children or readers with no background knowledge about the document content, but may be unnecessary for adults or those with sufficient knowledge.

Task. We introduce the primary task of elaboration generation: given some document context C consisting of text from the original and/or simplified documents, generate an elaboration E.

2.2 Extracting Elaborations

Detecting elaborative simplification requires crafting a way to reliably extract sentences containing new content in simplified documents. Asking humans to read and annotate every sentence in each document is prohibitively costly. To streamline this process, we first obtain candidate elaboration sentences with automatic sentence alignment, then use human annotation to extract true elaborations.

Candidate extraction. Each set of articles in the Newsela corpus consists of multiple simplified articles ranging from grades 3–12. We choose the article written for the lowest grade level as our simplified document (we leave investigating simplified documents across higher grade levels as future work). Using the approach from Zhong et al. (2020), we then align sentences from the original and simplified documents by thresholding the cosine similarity of sentence vector representations using Sent2Vec (Pagliardini et al., 2018). We then consider sentences in the simplified document that are not aligned with any sentence in the original document as *candidate elaborations*. Of the 54,892 sentences across the 1,042 simplified documents (on average, 52 sentences per document), 6,207 were extracted as candidate elaborations.

Human verification. Before crowdsourcing, we conducted a pilot study of elaboration verification with two sets of annotators: (1) Expert annotators (one graduate student, one undergraduate, both native speakers of English) who studied the data extensively; (2) 13 trusted undergraduate volunteer annotators at our university, also native English speakers. They received detailed instructions, but no face-to-face training. This allowed us to gauge task scalability and to gather feedback to design our crowdsourcing protocol. The 13 annotators each annotated a subset of 50 randomly selected documents (a total of 301 candidate elaborations) from our corpus. Each candidate elaboration was annotated by 2 to 4 annotators.

For each original-simplified document pair, we provided annotators with the entirety of both documents. We asked them to identify whether each candidate elaboration truly contained semantically new content, and to provide a rationale for their annotation. We *aggregated* the annotations for each candidate elaboration by taking the mode of all responses. The expert annotation consisted of 150 of these candidate elaborations under the same setup. Figure 2 shows some examples of verified and rejected candidate elaborations.

Agreement. Cohen's Kappa among the two expert annotators is 0.75, indicating substantial agreement (Artstein and Poesio, 2008). Cohen's Kappa between expert annotations and aggregated student annotations is also substantial, at 0.67. Krippendorff's alpha among the 13 student annotators is 0.37. As in complex NLP annotations (Nye et al., 2018), although there is subjectivity among individual annotators due to the complicated nature of the task, their aggregated judgment can be of as high quality as trained expert annotators.

2.3 Contextual Specificity

At first glance, it seemed that elaborative simplification might simply involve retrieving definitions (Paetzold and Specia, 2016) or crafting informative post modifiers (Kang et al., 2019). However, while annotating candidate elaborations, we noticed that elaborations in our corpus took a variety of forms.

To better understand content addition, we conducted an extensive study of elaborations and found that often times, clarification or analysis sentences specific to document context are inserted to aid comprehension or facilitate connections between content in the original text. Notably, elaborations vary in their *contextual specificity*, i.e., the degree to which an elaboration is specific to the context.[1] For example, while simple definitions can be inserted into several different documents mentioning the same entity (low contextual specificity), some elaborations containing clarifications, commonsense reasoning applied to document content, or explicit inference are more contextually specific, as illustrated in Figure 2.

This formulation is inspired by prior work in text specificity (Li et al., 2016; Ko et al., 2019) which is related to how a sentence "stands on its own" or sentence "decontextualization" as in Parikh et al. (2020). As we discuss in §2.4, contextually specific elaborations tend to have slightly lower sentence specificity, thus depending on the surrounding context to enhance understanding.

We ask the pair of experts from the previous pilot to annotate 116 randomly chosen verified elaborations for contextual specificity. Each expert was again given the entirety of the original and simplified documents with the highlighted elaboration, and asked to label its contextual specificity on a scale of 1–3 (low/medium/high). Their Fleiss' Kappa showed moderate agreement (Landis and Koch, 1977) with $\kappa = 0.57$. Spearman's correlation between the two annotators is 0.72. To enable collection, study, and modeling of this linguistic knowledge at scale, we gather contextual specificity ratings during crowdsourcing.

2.4 Crowdsourcing

Annotating elaboration verification and contextual specificity requires careful reading and thoughtful reasoning over text. For the pilot described in §2.2, we provided thorough instructions and example documents and annotations. While these trusted annotators delivered high quality, reliable annotations, they ultimately cannot annotate a dataset of the scale supervised systems require. To remedy this, we use Amazon Mechanical Turk to collect labels at scale, albeit with slightly more noise. Our rationale is that models can tolerate this during training, and we ensure cleaner validation and test sets through expert annotations.

Task setup. We ask workers to annotate elaboration verification and contextual specificity in a single task (HIT). For each candidate elaboration, we provide crowdworkers with the text region from the simplified document containing the elaboration, and the aligned text region from the original document. We ask crowdworkers to categorize each candidate as a true elaboration, not an elaboration, or indicate that the snippets were unrelated. If true elaboration is selected for a candidate, we asked them to rate its contextual specificity[2]. From feedback during our expert pilots, we determined that providing entire documents was often distracting, proving necessary only in rare cases where content was drastically rearranged. Instead, we display text regions of 5–7 sentences from both the simplified and original documents. The simplified text region contains the candidate elaboration and surrounding sentences, and the original text region contains sentences that are aligned with neighboring sentences

---

[1] We draw a distinction between contextual specificity and contextual relevance (as in Kang et al. (2019)).

[2] During crowdsourcing we utilized a 5-point scale, but aggregated the labels to a 3-point scale because the two scores on either end of the scale are not distinctive (i.e., are subjective).

of the elaboration in the simplified text region. We compose HITs that consist of ∼4 candidates from the same article.

Quality control. To ensure high quality annotations, we ask crowd workers to provide a rationale for each rating decision, as in §2.2. These rationales provide insight into worker interpretations of our task, allowing us to actively curate annotations to only include reliable annotations in our dataset. For example, using this method, we were able to remove annotations where crowd workers inflated specificity ratings due to coreferent entity mentions (i.e *"It is a tube that moves blood"* as opposed to *"An artery is a tube that moves blood").*

In addition, we require all crowd workers to reside in the US, UK, Canada, Australia, or New Zealand, and to have completed $\geq$ 100 HITs with an acceptance rate of 95%. Each elaboration is annotated by 5 different crowdworkers. Through active monitoring and small batches of HIT releases, we identified a set of workers that we trust and invite back to the task. Initially, we pay \$0.15 – \$0.23/HIT, and retroactively pay trusted workers at the rate of \$8/hr after work time information is obtained.

Agreement between trained and crowdsourced annotators. For both tasks, we aggregate crowdsourced labels by taking the mode of all responses[3]. Cohen's Kappa of elaboration verification between crowdworkers and experts is 0.37 (fair). To measure contextual specificity agreement between crowdworkers and experts, we use Krippendorff's alpha with an ordinal distance metric, aggregating Turker and expert responses using the mode to obtain an agreement value of α = 0.47, indicating moderate agreement (Artstein and Poesio, 2008). We attribute the disparity between inter-expert agreement and expert-crowdworker agreement to the challenge and subjectivity of this task, especially amongst untrained crowd workers. Though crowdsourcing our data does result in a slightly noisier training set, we are able to collect data for supervised learning and analysis at scale.

Dataset analysis. Using Mechanical Turk, we annotated 4,178 out of the 6,207 candidate elaborations from 1,042 documents. We obtained 1,299 verified elaborations, establishing an approximate 32% conversion rate from candidate to verified elaborations. Note that since candidate elaborations are obtained automatically, this does not accurately reflect the true elaboration rate per document, but rather a lower bound. On average, the elaborations in are corpus are 7–13 tokens long.

	Low	Medium	High	**Total**
Train	303	349	397	**1049**
Valid	71	39	24	**134**
Test	42	34	40	**116**
Total	406	423	470	**1299**

Table 1: Dataset distribution by contextual specificity.

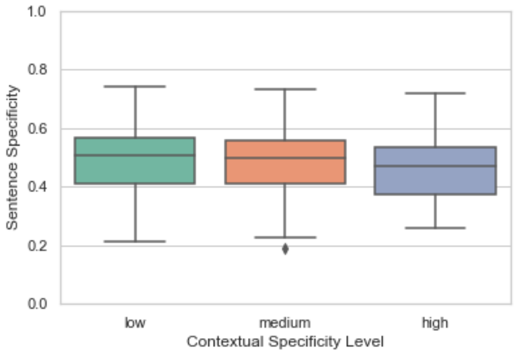

Figure 3: Sentence specificity distribution of elaborations across contextual specificity levels.

To ensure finetuning and evaluation quality, we use the expert-annotated subset of our data for the test set, and sought additional expert annotations for the validation set as well. Table 1 shows our dataset size across splits, stratified by contextual specificity. Our dataset contains a relatively uniform distribution of specificity levels, confirming our qualitative analysis that the contextual specificity of added content is diverse.

Sentence Specificity. As mentioned in §2.3, we explore the nature of sentence specificity of elaborations by running the sentence specificity predictor from Ko et al. (2019) on all standalone elaborations across all splits in our dataset. Sentence specificity predictions range on a continuous scale from 0 (very general) to 1 (highly detailed). Figure 3 shows the sentence specificity distribution across contextual specificity levels. The correlation between contextual and sentence specificity is $\tau = -0.11$, and is statistically significant. This negative correlation illustrates some of the intuition behind contextual specificity – only when highly contextualized elaborations are inserted into documents do they facilitate document understanding.

---

[3] Using the mean as an aggregation function resulted in noisier labels.

3 Elaboration Generation

We frame elaborative simplification as a natural language generation task, and describe a process mimicking editors elaborating as they compose a simplified document from the beginning (i.e. elaborations may be generated based only on the preceding simplified/original context) [4]. Elaboration generation is a challenging test for a model's ability to produce relevant and effective elaborations ranging in contextual specificity given snippets of context from documents in our corpus. We investigate the abilities of pre-trained language models to generate elaborations, establishing baselines in §3.1 and incorporating contextual specificity in §3.2. We find that selecting elaborations of appropriate levels of predicted contextual specificity can help improve elaboration generation results.

3.1 Baseline Elaboration Generation

We generate elaborations using GPT-2 (Radford et al., 2019), a large-scale pre-trained language model which has been shown to be effective in a range of generation tasks, including in recent efforts to elicit world and commonsense knowledge (Zhou et al., 2020; Shwartz et al., 2020).

Formally, we generate elaborations by conditioning on some document context, C. In this baseline setting, we generate sequences via greedy decoding. We utilize context from the original document (C_o) and from the simplified text (C_s). To understand the role that context plays in elaboration generation, we elicit elaborations from the language model by providing it one of the following: **(1)** 2 sentences prior to the gold elaboration in the simplified document (C_{2s}), **(2)** a concatenation of 2 sentences prior to the gold elaboration from the simplified document and the corresponding aligned region in the original document ($C_{2s} + C_o$), **(3)** 4 sentences prior to the gold elaboration in the simplified document (C_{4s}).

Finetuning. We finetune GPT-2 on the set of simplified documents written for the lowest grade level in the Newsela corpus, as well as on our dataset of verified elaborations excluding the test set. We found that such fine-tuning substantially improves generation quality (c.f. Appendix B.1).

3.2 Specificity-guided Generation

As discussed in §2.3, elaborations in our corpus are notably diverse in terms of their contextual specificity. Producing elaborations of appropriate contextual specificity is important, e.g., inserting an unnecessary definition instead of explaining a central concept can be ineffective or detrimental to readers' understanding. Rows 1-2 in Figure 4 show examples where the elaboration generated by the model in §3.1 does not match the level of contextual specificity of the gold elaboration, motivating our exploration of including contextual specificity and its prediction to aid elaboration generation.

Contextual specificity prediction. We build a model to classify the level of contextual specificity of an elaboration as low, medium, or high to incorporate downstream during generation. We leverage BERT (Devlin et al., 2019) for this task. Appendix A explores this auxiliary task further to understand modern NLP models' ability to capture this linguistic information.

We train the model on (E, s) pairs, where E is an elaboration, and s is its labeled contextual specificity. We feed E as input to BERT, and then feed the `[CLS]` token embedding into an output layer for classification. We freeze the BERT parameters since fine-tuning yielded unstable results. We utilize `bert-base` from the HuggingFace library (Wolf et al., 2019). After tuning on the validation set, we train for 5 epochs, using a batch size of 32 and a learning rate of 2e-3. We use the default dropout rate of 0.1 for self-attention layers, but refrain from adding dropout on our linear layer.

This contextual specificity model achieved an accuracy of 56.8 ± 1.5, a macro-averaged F1 score of 55.3 ± 1.6, a Spearman correlation of 47.5 ± 2.6, and a mean absolute error of 0.552 ± 0.01, averaged across 15 randomly initialized runs. This performance is better or on par with other models that incorporate document context in different ways (Appendix A). We find contextual specificity prediction to be a challenging task for BERT. Prediction of *expected* contextual specificity (i.e prediction from context alone, without the elaboration) was particularly difficult, and we leave building stronger models in this setting to future work.

Generation. We investigate the importance of contextual specificity in generating effective elaborations by comparing sequences generated in 3 ways:

---

[4] We explored elaboration generation as a post-processing task after document simplification (Appendix B.2). From preliminary results, we find it to be a more nuanced task which we leave for future work.

1. **Greedy:** Generate elaborations via greedy decoding. This setting was discussed in §3.1.
2. **Top-k:** Sample a sequence from the language model using top-k sampling (Fan et al., 2018), without considering contextual specificity.
3. **Contextual specificity-informed sampling**, shorthand **Contextual:** Sample sequences using top-k sampling until we have 3 elaborations of low, medium, and high contextual specificity, as predicted by the contextual specificity model, and select the sequence with predicted contextual specificity matching the gold specificity level.

In practice, one would ideally use a contextual specificity model trained *without* the elaboration itself (i.e., *Context-Only* models in Appendix A) to predict the appropriate level of contextual specificity of a generated elaboration. However, since we leave to future work to build a strong model presented with this setup, we instead utilize the gold specificity label and explore the upper bound with our generation experiments.

We use sampling-based decoding strategies to achieve contextual specificity diversity because we find that while beam-based decoding methods may result in sequences with diverse *content*, they do not necessarily result in sequences with diverse *contextual specificity*.

3.3 Experimental Settings

We use GPT-2 medium from the HuggingFace library (Wolf et al., 2019) to finetune and generate elaborations. We finetune GPT-2 on documents simplified for the lowest-grade level in the Newsela corpus for 3 epochs with a learning rate of 1e-5 and a batch size of 32. For sampled sequences, we use top-k sampling with $k = 40$, and a temperature of $t = 0.45$, tuned on validation data.

4 Generation Evaluation

As elaboration generation is a new task, we include BLEU scores for completeness and emphasize human evaluation, which provides important insight early on in the study of a new phenomenon.

4.1 Automatic Evaluation

We report BLEU (Papineni et al., 2002), a standard metric in generation tasks. Table 2 shows corpus BLEU-1 and BLEU-2 scores on our test set. As illustrated in Table 2, the best models, as reflected by BLEU, are those finetuned on the Newsela simplified corpus, with four sentences from the simplified document before the gold elaboration as context.

	Greedy		Top-k		Contextual	
Context	B-1	B-2	B-1	B-2	B-1	B-2
C_{2s}	20.8	6.77	20.4	6.12	21.4	7.26
$C_{2s} + C_o$	18.7	5.66	17.2	4.32	19.0	5.31
C_{4s}	20.8	5.54	19.7	6.06	**22.4**	**7.56**

Table 2: BLEU-1 and BLEU-2 scores for elaborations generated by GPT-2, finetuned on the Newsela simplified document corpus. Results for our best model, which we conduct human evaluation on, are in bold.

System	Greedy	Top-k	Contextual
% selected	53.2	44.9	58.0

Table 3: Percentage of annotations for which users selected elaborations generated by each model.

While BLEU captures lexical overlap between generated and gold elaborations, it is also criticized due to poor correlation with human judgments (Liu et al., 2016; Novikova et al., 2017; Chaganty et al., 2018), as it fails to capture semantic similarity or reward multiple plausible hypotheses. During manual inspection of these sequences, we find that elaborations produced after finetuning GPT-2 can be semantically plausible, coherent, and elaboration-like. Content that is pertinent and new, but that does not overlap with the content in the gold elaboration is not rewarded. In some cases, staying true to the content of the gold elaboration is likely unnecessary, as long as the contextual specificity is comparable (see row 4 in Figure 4). To that end, we also perform a human evaluation study of generated elaborations, given that the purpose of elaborations is largely to make simplified text easier to understand for readers.

4.2 Human Evaluation

We set up our human evaluation similar to Panthaplackel et al. (2020), providing a pair of expert evaluators elaborations generated by our C_{4s} model (see Table 2) in each of the three setups (greedy, top-k, contextual), and ask them to select the sequence they thought was most coherent, topical, semantically plausible, and elaboration-like. We allow selection of multiple sequences if they are equally good, and no selection if all sequences are poor. We report human evaluation results as the percentage for which evaluators chose the sequence as higher

Simplified Text	Generated Elaborations
It also has a high amount of smoking for a big city, according to a health department report. In Philadelphia, smoking is more common among the poor than people who aren't poor, said Giridhar Mallya. He works for the Philadelphia Department of Public Health. Being poor can be very hard on a person's mind. **It is not easy to relax, or to feel positive**. ***Contextual Specificity**: High*	**Greedy:** "It's not just the money," he said. **Top-k:** Mallya said the study showed that the poorest people in the city smoke a lot more than the richest people. **Contextual:** It can make them feel guilty.
Workers at the Danish zoo said they had no choice but to kill the giraffe. The reason why is complicated. They said its genes were too much like the genes of other giraffes. All plants and animals have genes. They play a big part in what animals and plants look and act like. Genes are passed down from parents. **They determine eye color, hair color, athletic ability, health and more.** ***Contextual Specificity**: Low*	**Greedy:** The scientists found that the giraffe's genes were very similar to those of other giraffes. **Top-k:** Scientists have known for years that the giraffe's genes were more like those of other giraffes. **Contextual:** The genes are passed down from generations.
Because they look like a poisonous butterfly, they are left alone by predators. Looking like another species works just fine for them. While they avoid getting eaten, the male and the masculine-colored female are still at risk. For a long time, biologists have studied how animals try to look like other animals. It teaches them a lot about natural selection. Natural selection is sometimes called "survival of the fittest." **It means that animals that can change will survive.** ***Contextual Specificity**: Low*	**Greedy:** It is a way of thinking about how animals live. **Top-k:** It means that the more an animal looks like another animal, the more likely it is to survive. **Contextual:** It's a way of thinking about how organisms adapt to change.
The newly found cat is probably part of a separate branch of the cat family, said Tseng. The modern snow leopard is most likely part of the same branch. Both are able to live high in the Tibetan mountains. Big cats present serious problems for paleontologists. **They have existed millions of years longer than people have.** ***Contextual Specificity**: High*	**Greedy:** They are slow and slow to change. **Top-k**: Paleontologists study ancient animals. **Contextual**: They are difficult to study.
Reporters and photographers watched. The big cat crept out of the cage. He looked around, and then ran off down a dirt road. The panther disappeared into the forest. **Wildlife workers were setting him free**. ***Contextual Specificity**: High*	**Greedy:** The news was not good. **Top-k:** Then, it disappeared into the darkness. **Contextual:** The news of the escape made people angry.

Figure 4: Examples of generated elaborations with the different decoding strategies described in §3. Exs. 1–3 are cases where selecting a contextually-appropriate generated elaboration was effective. Ex. 4 is a relevant, sound elaboration with no content overlap with the gold elaboration, hence not rewarded by automatic metrics. Ex. 5 is a difficult case where context is essential – the generated elaboration is not pertinent to document context.

Low	Medium	High
Cushing died in a battle in the War of 1812.	He was captured and taken to a prison.	Cushing was a hero, his supporters said.
A government shutdown is when there are no government services.	A large minority of Germans thought American lawmakers behaved badly, said a poll released Tuesday.	Many lawmakers and their supporters blame the news coverage of their actions.
Football is the national and most popular sport in the United States.	In 2010, just 1 percent of its subscribers played fantasy sports.	The league is also popular with high school and college students looking to build a fan following.

Figure 5: Example generated elaborations of varying contextual specificity.

quality. Two annotators each annotated all 116 examples in our test set, resulting in 232 evaluations total. Table 3 shows these results. We calculate human agreement via Cohen's kappa with MASI distance (Passonneau, 2006), obtaining $\kappa = 0.51$, indicating moderate agreement (Artstein and Poesio, 2008). This round of evaluation confirmed that incorporating contextual specificity is helpful, consistent with our findings with BLEU.

5 Analysis and Discussion

We observe that GPT-2, finetuned on simplified text from the Newsela corpus, is able to adopt elaborative style (i.e short sentences of 7–13 tokens with limited vocabulary), see Figure 4. We find that the model can be effective at generating simple definitions and reasoning. However, the content contained in the elaborations is often not anchored in the document itself – generated sequences seem relevant to the snippet of context provided, but less so when placed in the larger document (see row 5 of Figure 4).

Original Text. We observe that our best model involves context only from the simplified document. We attribute the drop in performance of models with C_o as a part of input largely to the crude incorporation of content from the original document, which is stylistically starkly different from simplified text, most notably in terms of length and vocabulary complexity. Since one of the main sources of relevant content during simplification is the original document, better methods to incorporate text or information from the original document is an important direction for future work.

Effectiveness of contextual specificity. Decoding with top-k sampling allowed GPT-2 to generate low, medium, and high contextualization sequences. A few examples of generated elaborations with varying contextual specificity that were conditioned on the same context are shown in Figure 5.

For most of our models, we do see an improvement when appropriately contextually specific sequences are chosen (rows 1–3 in Figure 4), suggesting the importance and need for further improvement of contextual specificity models.

While our methods take contextual specificity into account, they do not consider factuality or larger document relevance. An improved decoding scheme considering these could promote sequences that better align with larger document context.

Retrieval. Elaborations of medium to high contextual specificity often involve external knowledge not readily available from the simplified or original text. For example, generating factually correct details about a certain event or entity with little to no background on the event the document is referring to can prove challenging for pre-trained language models. To that end, generating truly effective elaborations of medium to high contextual specificity may require some type of retrieval module.

6 Related Work

Text simplification has been studied extensively (Siddharthan, 2014), especially at the sentence level. Recent progress has largely been driven by adapting monolingual translation for sentence simplification (Wubben et al., 2012; Wang et al., 2016; Xu et al., 2016; Zhang and Lapata, 2017; Dong et al., 2019; Kriz et al., 2019). This paradigm, while effective at transforming text, does not suffice when *new* content needs to be generated. A recent survey (Alva-Manchego et al., 2020) identifies explanation generation in simplification as an understudied area in dire need of new resources and methods. We tackle content addition, framed as explanation generation during simplification, and name it broadly as *elaborative simplification*.

The need for elaborative simplification is highlighted in prior hand-coded analysis (Yano et al., 1994), which showed that language learners and other audiences benefit from insertion of relevant elaborations and explanations, and that new or unfamiliar concepts negatively impact reading comprehension (Kintsch and Vipond, 1985). However, existing computational approaches are limited to the retrieval of definitions (Damay et al., 2006; Kandula et al., 2010; Eom et al., 2012; Paetzold and Specia, 2016), or constrained tasks such as post-modifier generation (Kang et al., 2019).

7 Conclusion

We presented the first data-driven study of elaborative simplification, i.e., content insertion during text simplification. We constructed a new corpus of 1.3K verified elaborations, observing a spectrum of contextual specificity and rich types of added content. We developed baselines for elaboration generation using pre-trained language models and found that considering contextual specificity could improve generation quality. We discussed some of the challenges of generating elaborations, and call for techniques to address elaborative simplification.

Acknowledgments

We thank Greg Durrett for reviewing an early draft of this paper, and Joel Tetreault and the UT Austin Computational Linguistics group for valuable feedback and discussions. This work was partially supported by NSF Grant IIS-1850153. We also acknowledge the Texas Advanced Computing Center (TACC) at UT Austin for providing the computational resources for many of the results within this paper.

References

Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. 2020. Data-driven sentence simplification: Survey and benchmark. *Computational Linguistics*, 46(1):135–187.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Joachim Bingel, Gustavo Paetzold, and Anders Søgaard. 2018. Lexi: A tool for adaptive, personalized text simplification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 245–258, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Melbourne, Australia. Association for Computational Linguistics.

Han-Bin Chen, Hen-Hsen Huang, Hsin-Hsi Chen, and Ching-Ting Tan. 2012. A simplification-translation-restoration framework for cross-domain SMT applications. In *Proceedings of COLING 2012*, pages 545–560, Mumbai, India. The COLING 2012 Organizing Committee.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA. Association for Computational Linguistics.

Jerwin Jan S Damay, Gerard Jaime D Lojico, Kimberly Amanda L Lu, D Tarantan, and E Ong. 2006. SIMTEXT: Text simplification of medical literature. In *Proceedings of the 3rd National Natural Language Processing Symposium-Building Language Tools and Resources*, pages 34–38.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Prroceedings of the SIGIR workshop on accessible search systems*, pages 19–26.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402, Florence, Italy. Association for Computational Linguistics.

Soojeong Eom, Markus Dickinson, and Rebecca Sachs. 2012. Sense-specific lexical information for reading assistance. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 316–325, Montréal, Canada. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting proper lexical paraphrase for children. In *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013)*, pages 59–73, Kaohsiung, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Sasikiran Kandula, Dorothy Curtis, and Qing Zeng-Treitler. 2010. A semantic and syntactic text simplification tool for health content. In *AMIA annual symposium proceedings*, pages 366–370.

Jun Seok Kang, Robert Logan, Zewei Chu, Yang Chen, Dheeru Dua, Kevin Gimpel, Sameer Singh, and Niranjan Balasubramanian. 2019. PoMo: Generating entity-specific post-modifiers in context. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 826–838, Minneapolis, Minnesota. Association for Computational Linguistics.

Walter Kintsch and Douglas Vipond. 1985. Reading comprehension and readability in educational practice and psychological theory. *Perspectives on learning and memory*, pages 329–365.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019. Domain agnostic real-valued specificity prediction. In *Proceedings of AAAI*, pages 6610–6617.

Reno Kriz, João Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. 2019. Complexity-weighted loss and diverse reranking for sentence simplification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3137–3147, Minneapolis, Minnesota. Association for Computational Linguistics.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Junyi Jessy Li, Bridget O'Daniel, Yi Wu, Wenli Zhao, and Ani Nenkova. 2016. Improving the annotation of sentence specificity. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3921–3927, Portorož, Slovenia. European Language Resources Association (ELRA).

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain Marshall, Ani Nenkova, and Byron Wallace. 2018. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 197–207, Melbourne, Australia. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016. Anita: An intelligent text adaptation tool. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 79–83, Osaka, Japan. The COLING 2016 Organizing Committee.

Gustavo Henrique Paetzold. 2016. *Lexical Simplification for Non-Native English Speakers*. Ph.D. thesis, University of Sheffield.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Sheena Panthaplackel, Pengyu Nie, Milos Gligoric, Junyi Jessy Li, and Raymond Mooney. 2020. Learning to update natural language comments based on code changes. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1853–1868, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Rebecca Passonneau. 2006. Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

David Pellow and Maxine Eskenazi. 2014. An open corpus of everyday documents for simplification tasks. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 84–93, Gothenburg, Sweden. Association for Computational Linguistics.

Sarah E Petersen and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Workshop on Speech and Language Technology in Education*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Luz Rello, Ricardo Baeza-Yates, Laura Dempere-Marco, and Horacio Saggion. 2013. Frequent words improve readability and short words improve understandability for people with dyslexia. In *IFIP Conference on Human-Computer Interaction*, pages 203–219.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Sara Botelho Silveira and António Branco. 2012. Enhancing multi-document summaries with sentence simplification. In *Proceedings of IJCAI*.

Sanja Štajner and Maja Popovic. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *Proceedings of AAAI*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Yasukata Yano, Michael H Long, and Steven Ross. 1994. The effects of simplified and elaborated texts on foreign language reading comprehension. *Language learning*, 44(2):189–219.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

Yang Zhong, Chao Jiang, Wei Xu, and Junyi Jessy Li. 2020. Discourse level factors for sentence deletion in text simplification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9709–9716.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9733–9740.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China. Coling 2010 Organizing Committee.

A Contextual Specificity Prediction

We further explore the auxiliary task of contextual specificity prediction introduced in §3.2, prompted by the observation of diverse elaborations in our corpus. Formally, the task involves predicting the contextual specificity s of an elaboration E as low, medium, or high, given some document context C.

A.1 Methods

As described in §3.2, we use BERT (Devlin et al., 2019) for this classification task. We do so in two settings based on surrounding text and/or the actual elaboration. Settings which include the elaboration can aid generation models by utilizing generated hypothesis elaborations and surrounding text to select sequences that are appropriately contextually specific. Settings that operate off context alone capture the *expected* level of specificity. In addition to the E-only model presented in §3.2, we explore combinations of E, C_o (original document context) and C_{4s} (4 sentences prior to the gold elaboration from the simplified document).

With elaboration. We feed the input sequence into BERT and use `[CLS]` token representation of the sequence, projecting it using a weight matrix $W \in \mathbb{R}^{dx3}$. Input sequences with the elaboration consist of `[CLS]` C `[SEP]` E, where C is either C_o or C_{4s}, or both. When both types of context are used, we learn a representation for a separation token `[CONTEXT_SEP]` to distinguish between the two, and use $C = C_o$ `[CONTEXT_SEP]` C_{4s}.

Context only. While contextual specificity clearly involves the elaboration itself, context-only models help us understand whether it is predictable from context alone, and simulate a realistic setting during simplification, when these models may be incorporated before the actual elaborative text is generated. Input to these models is crafted similarly, but excluding E from the sequence.

A.2 Experiments and Analysis

We train on (E, s) pairs, and utilize `bert-base` from the HuggingFace Transformers library. We feed the sequence representation from the `[CLS]` token embedding into an output layer for classification [5]. For each setting, we train for 5 epochs, using a batch size of 32, and a learning rate of 2e-3. We use the default dropout rate of 0.1 for self-attention layers, but refrain from adding dropout on our linear layer.

Results. We use the same four metrics to evaluate our results – two classification metrics (accuracy, macro-averaged F1), and two regression metrics (Spearman's correlation and mean absolute error), and we again report mean performance over 15 different, randomly initialized runs. Results are shown in Table 4, and suggest that this is a challenging task, even for powerful pre-trained language models. The best predictor of contextual specificity, in terms of correlation and MAE, is context in the form of 4 sentences before the elaboration combined with the elaboration itself. However, the elaboration-only model performs the best in terms of accuracy and F1.

Original Text Presence. In all settings in which the aligned snippet of text from the original document was fed in as partial or complete input to the model, we see a reduction in performance. Compared to text from the simplified document, text from the original document is stylistically distinct. Consequently, when jointly fed in as context with simplified text, the input is largely incoherent, potentially impacting the model. We leave studying more effective ways of incorporating context from the original document to future work.

Qualitative Analysis. In cases where linguistic cues explicitly indicate the level of contextual specificity, our model performs well—i.e when definitions are inserted as *"A is B"* or reasoning is inserted as *"A but B"* or *"The reason for A is B"*. However, predicting the contextual specificity of more nuanced sentences may require an improved method of modeling surrounding context. For example, when the elaboration contains a definition of a term from a different sentence using coreferent mentions, our model predicts a higher level of contextual specificity. In general, our model over-predicts highly contextualized elaborations, and under-predicts lower levels of contextual specificity. Medium contextual specificity was hardest for our models to predict accurately.

Amount of context. To understand the impact of the amount of context on performance, we vary the number of sentences ($\{2, 4, 6\}$) before the elaboration to feed into our best performing model involving context ($C_s + E$). Table 5 shows these results. We see that merely increasing the amount of con-

---

[5] We tried finetuning our contextual specificity prediction models on our elaboration dataset, but found that our dataset was too small to yield stable results.

	Context	Acc.	F1	Correlation	MAE
Context Only	$C_o + C_{4s}$	45.2 ± 3.0	43.1 ± 2.8	27.8 ± 4.9	0.729 ± 0.05
	C_{4s}	46.4 ± 2.9	44.9 ± 3.0	32.4 ± 4.4	0.679 ± 0.04
	C_o	37.9 ± 4.6	36.1 ± 5.4	20.2 ± 1.0	0.813 ± 0.07
With Elaboration	E	$\mathbf{56.8 \pm 1.5}$	$\mathbf{55.3 \pm 1.6}$	47.5 ± 2.6	0.552 ± 0.01
	$C_o + C_{4s} + E$	50.5 ± 3.8	48.3 ± 4.0	40.4 ± 5.8	0.628 ± 0.05
	$C_{4s} + E$	55.3 ± 3.3	54.0 ± 2.5	$\mathbf{50.8 \pm 4.1}$	$\mathbf{0.545 \pm 0.03}$
	$C_o + E$	43.7 ± 1.8	41.7 ± 2.0	26.7 ± 3.6	0.749 ± 0.03

Table 4: Contextual Specificity Prediction results, including accuracy, macro-averaged F1, Spearman's correlation, and Mean Absolute Error, reported across 15 runs. We bold our best results. The performance differences between (1) $C_{4s} + E$ vs E, (2) $C_o + C_{4s}$ vs C_{4s}, and (3) $C_o + C_{4s} + E$ vs. $C_{4s} + E$ are not statistically significant.

	Acc.	F1	Corr	MAE
C_{2s}	53.6 ± 1.8	51.2 ± 3.1	$\mathbf{51.7 \pm 6.8}$	0.566 ± 0.05
C_{4s}	$\mathbf{55.3 \pm 3.3}$	$\mathbf{54.0 \pm 2.5}$	50.8 ± 4.1	$\mathbf{0.545 \pm 0.03}$
C_{6s}	53.9 ± 4.0	52.0 ± 3.6	44.3 ± 5.5	0.591 ± 0.04

Table 5: Mean performance of $C_s + E$ model over 15 runs with varying amounts of pre-elaboration context.

text fed to the model does not translate to stronger results – considering overall performance, 4 sentences before the elaboration from the simplified document performed best.

B Elaboration Generation

B.1 GPT-2 Finetuning

We explore generation with GPT-2 across varying finetuning settings – (1) zero shot (no finetuning, only relying on GPT-2's pre-training), (2) finetuning on the set of simplified documents in the Newsela corpus (excluding documents from the test set), and (3) finally on our elaboration corpus. We utilize the same 3 decoding schemes described in § 3.2 across these different finetuning settings. We used a temperature of $t = 0.7$ for the zero shot setting, and $t = 0.45$ for finetuned settings. For finetuning on our elaboration corpus, we trained for 3 epochs with a batch size of 8 and a learning rate of 1e-3. We report BLEU-1 and BLEU-2 as described in § 4.1. As BLEU metrics for setting 2 are already included in Table 2, we report metrics for zero-shot generation (Table 6), and for generation after finetuning on our elaboration corpus (Table 7). Comparatively, finetuning GPT-2 on the set of simplified Newsela documents yielded the best performance, and we attribute this to there being strictly more data in that setting as opposed to our corpus of verified elaborations.

	Pre-trained GPT-2					
	Greedy		Top-k		Contextual	
Context	B-1	B-2	B-1	B-2	B-1	B-2
C_{2s}	12.48	2.71	9.82	2.04	11.93	2.66
$C_{2s} + C_o$	12.21	2.58	9.80	2.08	10.86	2.82
C_{4s}	13.46	3.35	11.78	2.43	13.80	3.89

Table 6: BLEU-1 and BLEU-2 for generation after finetuning on our elaboration corpus.

	Fine-tuned GPT-2: Elaboration Corpus					
	Greedy		Top-k		Contextual	
Context	B-1	B-2	B-1	B-2	B-1	B-2
C_{2s}	20.9	6.82	19.11	5.32	19.38	5.47
$C_{2s} + C_o$	11.89	2.78	12.72	2.77	14.2	3.05
C_{4s}	20.17	5.87	16.89	4.09	18.97	5.16

Table 7: BLEU-1 and BLEU-2 for the zero-shot generation setting.

B.2 Generation with BART

In addition to GPT-2, we experimented with BART (Lewis et al., 2020), a pre-trained sequence to sequence model. The encoder-decoder nature of BART allows us to explore elaborative simplification as a post-processing/post-editing scenario, where the model can receive context both preceding *and* following the elaboration in the simplified text.

We finetune `bart-base` available via the HuggingFace Transformers library, and feed in four different types of context (1) C_{2s}, (2) C_{4s}, (3) C_{2s+}, (4) C_{4s+}. The latter two context settings utilize two and four sentences before and after the elaboration (without the elaboration itself). In all settings, the gold elaboration was the target. We finetune for 3 epochs, with a batch size of 2, and a learning rate of 1e-4, and generate elaborations via greedy decoding. Results are shown in Table 8.

We find that BART is able to adopt elaborative

	C_{2s}	C_{2s+}	C_{4s}	C_{4s+}
B-1	18.9	21.5	20.2	20.1
B-2	5.05	6.68	6.02	6.18

Table 8: BLEU-1 and BLEU-2 for greedy generation with BART.

style, generating short sequences with limited vocabulary, however we observe that the smaller size of our corpus affected BART's ability to generate coherent, diverse elaborations. In addition, we note that framing elaborative simplification as a post-processing task is a more difficult, nuanced setting – the generated elaboration to be inserted must maintain the flow of the text and blend with the content present subsequent sentences. Elaborative simplification in this setting is another interesting, rich direction for future work.

Could you give me a hint?
Generating inference graphs for defeasible reasoning

Aman Madaan [*], **Dheeraj Rajagopal** [*], **Niket Tandon** [*] [†], **Yiming Yang, Eduard Hovy**
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[†] Allen Institute for Artificial Intelligence, Seattle, WA, USA
`{dheeraj,amadaan,yiming,hovy}@cs.cmu.edu`
`{nikett}@allenai.org`

Abstract

Defeasible reasoning is a mode of reasoning where conclusions can be overturned by taking into account new evidence. A commonly used method in cognitive science and logic literature is to handcraft argumentation supporting inference graphs. While humans find inference graphs very useful for reasoning, constructing them at scale is difficult. In this paper, we automatically generate such inference graphs through transfer learning from a related NLP task that shares the kind of reasoning that inference graphs support. Through automated metrics and human evaluation, we find that our method generates meaningful graphs for the defeasible inference task. Human accuracy on this task improves by 20% by consulting the generated graphs. Our findings open up exciting new research avenues for cases where machine reasoning can help human reasoning.[1]

1 Introduction

Defeasible inference (Rudinger et al., 2020) is a mode of reasoning in which given a premise **P** (Rob went for a hike), a hypothesis **H** (Rob saw an elephant, it was pink) may be weakened or overturned in light of new evidence i.e., an update **U** (Rob often has hallucinations). Given the non-monotonic nature of this reasoning, humans find it challenging to master this task (Morgan, 2004). This problem has been widely studied in classical AI through logic (Israel, 1980; McCarthy, 1981), and in cognitive science through argumentative models (Pollock, 1987). A prominent approach is to support defeasible inference through argumentations by constructing an *inference graph* (Pollock, 2009).

Despite their prominence (Bentahar et al., 2010), argumentative models are not scalable because an inference graph needs to be handcrafted for every example. Recently, Rudinger et al. (2020) proposed two auxiliary tasks related to defeasible inference: (i) an NLI task to predict whether an update **U** would weaken or strengthen a hypothesis **H**, and (ii) a generative task to generate an update **U** given a premise **P** and a hypothesis **H**. However, this only addresses a part of the problem because their inference is still not supported by the line of reasoning that a human typically uses to solve this task, namely mediators (e.g., hallucinations can be deceptive) and contextualizers (some elephants can have mutated gene which makes them look different) that are inherently embedded in an inference graph, limiting their utility for humans (figure 1).

In this paper, we adopt the concept of an inference graph for defeasible reasoning from cognitive science and provide a computational model to make their generation scalable. Training such a model would require a large amount of annotated inference graphs, which will be too expensive to obtain. Instead, our solution is to draw a parallel to a related reasoning task in NLP (Tandon et al., 2019), where the reasoning is supported by a graph that we find has similarities with the kind of reasoning that an inference graph supports. We train a model that can learn from the NLP task and effectively transfer it to generate inference graphs. Such transfer learning is made possible due to the powerful seq-to-seq neural language models that did not exist before.

The contributions of this paper are the answers to the following two research questions:

RQ1 Can we automate the construction of the argumentation supporting inference graphs? In §2, we show that we can effectively construct meaningful graphs using transfer learning.

RQ2 Can our generated graphs help improve hu-

---

[*] Equal Contribution

[1] A dataset of 230,000 influence graphs for each defeasible query is located at: `https://tinyurl.com/defeasiblegraphs`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5138–5147
August 1–6, 2021.

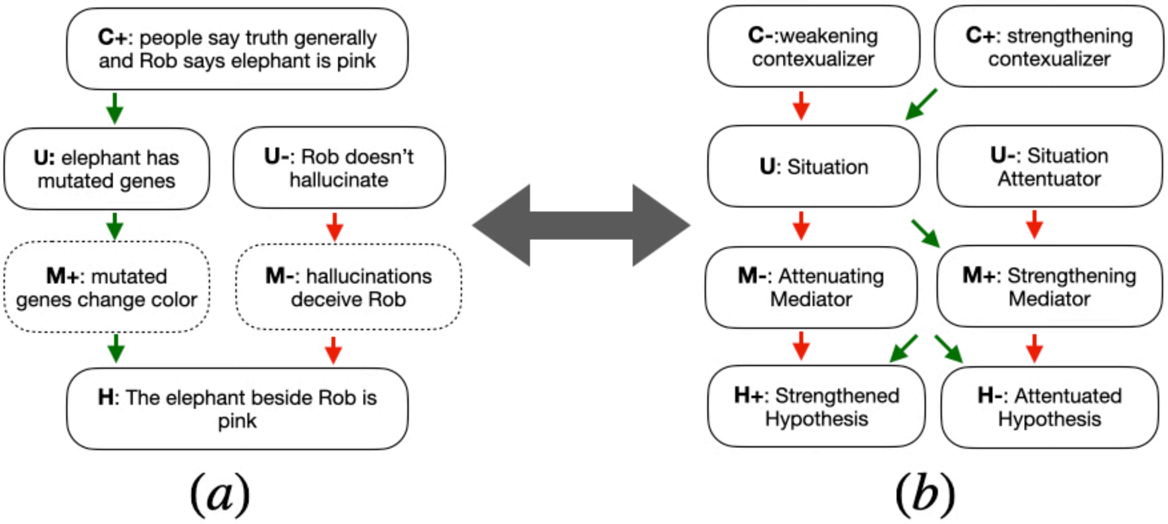

Figure 1: (**a**) An example of an Inference Graph adapted from Pollock (2009) and (**b**) Structure of an Influence Graph adapted from WIQA (Tandon et al., 2019) dataset. The adapted influence graph incorporates the contextualizers, mediators, hypotheses and situations, making them useful for defeasible reasoning.

man performance? In §3, we show that humans leverage generated graphs to improve their performance on a previously reported benchmark.

2 RQ1: Generating argumentation supporting Inference Graphs

We start by drawing parallels to a counterfactual reasoning task in NLP - the WIQA (Tandon et al., 2019) task. WIQA consists of a set of procedural passages, each accompanied by a human-curated *influence graph*. The influence graph captures the causal influences between the events in the context of the process described by the passage. We draw a connection between inference graphs (Pollock, 2009) and influence graphs (Tandon et al., 2019) by drawing parallels between their reasoning structures. In essence, each inference graph from Pollock (1987) can be instantiated via an influence graph from Tandon et al. (2019) by interpreting the nodes in both the graphs as follows (Figure 1):

i. **Contextualizers (C):** these nodes set the context around a situation and connect to the **P** in some way.

ii. **Updates (U):** these nodes are new situations that emerge which might overturn an inference.

iii. **Hypothesis (H):** Hypothesis nodes describes the outcome/conclusion of the situation.

iv. **Mediators (M):** Mediators are nodes that help bridge the knowledge gap between a situation and a hypothesis node by explaining their connection explicitly.

Figure 1 presents an example to highlight the similarities between the two graphs by labeling an example node adapted from (Pollock, 2009), and the structure of the influence graph from (Tandon et al., 2019) with the four node types that we defined above. A green edge indicates that the source node has a positive influence on the target node, and a red edge indicates a negative influence. Further, each node can either act as a *strengthener* (+) or a *weakener* (-) for the hypothesis. Consequently, these graphs can support similar type of reasoning e.g., the effect of **U** on **H** and how this can change in light of external influences (**C**) is captured by graph paths **C+** to **U** and from **U** via a mediator node (**M+/M-**) to **H**. Inspired by these similarities, we hypothesize that influence graphs can be used to supplement defeasible reasoning.

2.1 Influence Graphs Generation

To obtain an influence graph for each defeasible query, we perform a zero-shot transfer from WIQA (Tandon et al., 2019), a corpus of 2100 (passage, influence graphs) pairs.[2].

Training : We treat influence graph generation as a sequence-to-sequence mapping task. We leverage WIQA to derive parallel data $\{(\mathbf{seq}^i_{ip}, \mathbf{seq}^i_{op})\}_{i=1}^N$ for the task. Let $(\mathbf{T}_i, \mathbf{G}_i)$ be a sample in WIQA,

---

[2] Dataset details in the Appendix §E.

where $\mathbf{T}_i$ is the passage text (e.g. describing how viruses spread), and $\mathbf{G}_i$ is the corresponding influence graph (e.g., Figure 2). To create tokens of the input sequence $\mathbf{seq}^i_{ip}$, the model trains best with explicit markers:[3]

$$\mathbf{seq}^i_{ip} = \text{Premise: } \mathbf{T}_i \mid \text{Update: } \mathbf{U}_i \mid \text{less/ more: } \mathbf{H}_i \quad (1)$$

where $\mathbf{T}_i$ is the passage text (e.g. steps describing how viruses spread) and $\mathbf{U}_i$ and $\mathbf{H}_i$ are nodes of $\mathbf{G}_i$ (these are phrases as shown in Figure 2).

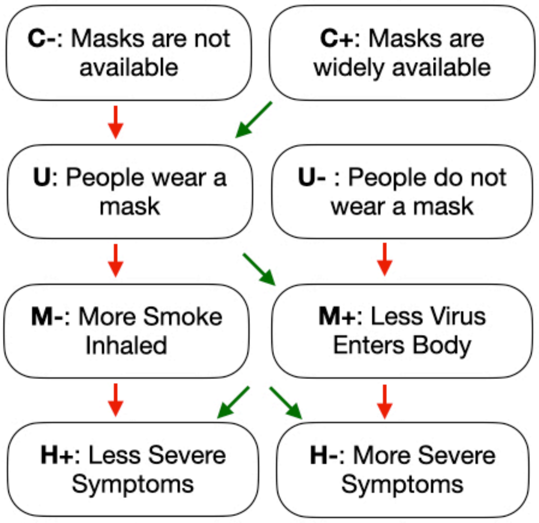

Figure 2: An example of an influence graph similar to ones in WIQA that we train on.

The output $\mathbf{seq}^i_{op}$ is set to a DOT-string representation of the corresponding influence graph $\mathbf{G}_i$, as such a representation was shown to be effective at extracting high-quality graphs (Madaan and Yang, 2021) from free-form text using language models (examples in the appendix). Thus, each passage-graph pair $(\mathbf{T}_i, \mathbf{G}_i)$ from WIQA is mapped to an input-output pair $\mathcal{D} = (\mathbf{seq}^i_{ip}, \mathbf{seq}^i_{op})$. We use this corpus to fine-tune an autoregressive language model $\mathcal{L}$ for graph generation. Essentially, the fine-tuned $\mathcal{L}$ allows us to efficiently sample an influence graph for a given input sequence $\mathbf{seq}^j_{ip}$ by drawing samples from $\mathbf{G}_j \sim P_\theta(y \mid \mathbf{seq}^j_{ip})$ using greedy sampling, where θ denotes the parameters of the language model.

Zero-shot Transfer to Defeasible Inference : We use the model $\mathcal{L}$ trained on WIQA to generate inference graphs on the defeasible inference dataset by Rudinger et al. (2020). We obtain an influence graph for each defeasible input (**P**, **H**, **U**) by converting it to an input sequence that can be fed to $\mathcal{L}$ by filling the template (1). This conversion from (**P**, **H**, **U**) to template (1) is done by setting the premise $\mathbf{P}$ as the context passage $\mathbf{T}$, the update $\mathbf{U}$ as the node $\mathbf{U}$, and the attenuated and strengthened outcomes are simulated by prefixing the hypothesis $\mathbf{H}$ with the tokens *Less* and *More* respectively. This input is then passed to the $\mathcal{L}$ to generate an influence graph.

Results on Influence Graph Generation We use T5-11B (Raffel et al., 2020) fine-tuned on $\mathcal{D}$ derived from WIQA (§2.1) as our graph generation language model ($\mathcal{L}$). All the graphs generated by our model were in valid DOT format. We use the standard generation metrics BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) to evaluate $\mathcal{L}$ on the test split of WIQA. Each node N_i in the reference graph is compared with the corresponding generated node $\hat{N}_i$ using $\texttt{BLEU}(N_i, \hat{N}_i)$ (Node-BLEU). Further, node-edge-node pairs (neighbors) (N_i, N_j) and $(\hat{N}_i, \hat{N}_j)$ are compared using Rel-BLEU $= \texttt{HM}(\texttt{BLEU}(N_i, \hat{N}_i), \texttt{BLEU}(N_j, \hat{N}_j))$ where $\texttt{HM}$ is the harmonic mean. These metrics are averaged over the graph (i.e., across the nodes and the edges), and further averaged across the corpus. We perform these experiments across two different language models: GPT-2-MEDIUM (Radford et al., 2019) and T5-11B. Finally, we calculate the overlap in the edge structures of the reference and generated graphs match as Edge-MATCH%. We report the numbers in Table 1, and include a random baseline for reference. A random baseline will correctly generate the nodes **S**, **H+**, and **H-** as they are part of the query ($\frac{3}{8}$ nodes). As neither of these nodes are connected to another, the random baseline will likely not generate any node pair correctly (Rel-BLEU $\sim$ 0). Since two unique graph structures are possible (Tandon et al., 2019), a random baseline would get Edge-match $\sim$ 50%. Table 1 shows that our T5-based model is able to generate syntactically valid (high edge-match) and semantically meaningful graphs. Additionally, we find that our generated graphs are helpful to humans on a downstream task, as described next.

3 RQ2: Do generated graphs help humans at defeasible reasoning?

Human Evaluation Rudinger et al. (2020) performed a human evaluation on 2000 defeasible queries, where given (**P**, **H**, **U**), the task was to label the nature of the effect of $\mathbf{U}$ on $\mathbf{H}$ as *Intensifies*

---

[3] An example shown in Appendix §A.

Model	Random	GPT-2-MEDIUM	T5-11B
Node-BLEU	37.5	46.05	**50.94**
Rel-BLEU	0.0	19.34	**33.01**
Edge-match%	50.0	92.86	**97.63**

Table 1: Results on automated metrics showing that our T5-11B model is able to generate very accurate graph structure and meaningful nodes that sufficiently match the reference nodes.

or *Attenuates*. Three human judges labeled each query, and the majority label was then compared with the ground-truth to ascertain the accuracy. In their setup, human judges were collectively right on 1745 samples (correct pool) and wrong on 255 samples (wrong pool). We create a challenging pool of 510 queries for the human judges by combining the 255 queries in the wrong pool with 255 queries sampled from the correct pool, giving a baseline accuracy of 50% for this eval pool. Each query in this pool is supplemented with a generated influence graph (§2).[4] We found that our generated influence graphs showed high-levels of redundancy in contextualizers and mediators, with about 46% of the generated influence graphs repeating these nodes. We found that humans find it simpler to follow positive chains of influence, so to reduce their cognitive load, we post-process each influence graph to only retain the strengthening contextualizer (Figure 1), the situation (**U**), the strengthening mediator (**M+**), and the hypothesis (**H**).

In order to establish comparable gains, we replicate the evaluation setup of Rudinger et al. (2020) by using use the same Amazon Mechanical Turk template and the instruction set, and the same pool of 230 qualified annotators that Rudinger et al. (2020) selected based on a paid qualification test, in which the workers were asked to answer SNLI queries of varying levels of difficulty. We paid slightly above $15 per hour for the tasks.

For each query, in addition to answering the defeasible question, three judges were asked to evaluate the augmented influence graphs on two aspects:

i) **Is the influence graph useful?** The judges were allowed to select from the following:

  (a) *helpful*: the graph was crucial in helping towards answering the question

  (b) *relevant but not helpful*: the graph had the right topic (relevant to the question) but did not help in answering the question.

  (c) *irrelevant or misleading*: the graph was irrelevant to the question or misled the human judge to a wrong answer.

ii) **Why is the influence graph useful?** The judges were given an option to highlight the most useful aspect of the generated influence graph. They were allowed to tag one or more of the following aspects as the most helpful: i) Extraneous node, ii) Mediating node, and iii) Structure of the graph.

We summarize the key findings below.

Finding 1: influence graphs are helpful and relevant As Table 2 shows, a large majority of the human judges found the influence graphs to be helpful or relevant. We calculate the inter-annotator agreement for this question using majority-agreement $= \frac{1}{N}\sum_{i=1}^{N} \texttt{ma}_i$ where $\texttt{ma}_i$ indicates a majority agreement for the i^{th} sample (i.e., at least 2 out of 3 judges agreed on the label for the sample). The majority-agreement (`ma`) on these labels was 0.83. The judges marked about 25% of the graphs as relevant but not helpful. The graphs in such cases were on topic but not helpful in answering the query, thereby distinguishing the cases when the graph was crucial in reaching the correct answer. Finally, we note that the graphs provided as hints could have been helpful in two ways: by helping the human annotators arrive at the answer, or by reinforcing their mental picture that helped them in making the right decision. Future research in this direction is needed to study these aspects in depth.

Helpful	47.25
Relevant but not helpful	25.09
Irrelevant or misleading	10.58
No majority agreement	17.05

Table 2: Helpfulness of the augmentations.

Finding 2: Mediators are the most helpful for defeasible queries For every sample, we asked the human judges to mark which parts of the graph was the most helpful (as shown in Figure 6 in Appendix §D.1). The judges could select more than one aspect of the graph if they found multiple useful aspects. Table 3 shows the percentage of human

---

[4] Discussion on IRB exemption in Section §B.

judges that selected the particular graph aspect as most helpful. We observe that 49.48% of the judges who found the graphs useful indicated the mediator node as the most helpful. This indicates that while there may be other events that impact $\mathbf{U}$ and $\mathbf{H}$, the mediating events are the most informative in determining the type of link between them.

Aspect	% marked useful
Mediator	49.48
Extraneous	32.03
Structure	12.81
None helpful	5.68

Table 3: Most useful aspects of an influence graph.

Finding 3: Machine generated influence graphs help humans in defeasible reasoning Table 4 shows that performance improves across all three tasks when the defeasible query is augmented with an influence graph. On our challenging set of 510 queries, the overall accuracy jumps nearly 20 points from 0.50 to 0.698. Figure 3 highlights that 113 queries that were previously given the wrong answers were marked correctly when augmented with the influence graphs.

Dataset	Human (Rudinger et al., 2020)	Human (ours)
SNLI	0.461 ± 0.11	0.553 ± 0.11
SOCIAL	0.628 ± 0.07	0.814 ± 0.06
ATOMIC	0.418 ± 0.06	0.657 ± 0.06
overall	0.500 ± 0.04	$\mathbf{0.698 \pm 0.04}$

Table 4: Human performance (accuracy) on the three tasks with and without generated influence graphs along with Wilson's score intervals for $\alpha = 95\%$. We tested the statistical significance of these results using the McNemar's test (McNemar, 1947) and found the results to be statistically highly significant ($p < 1e - 6$).

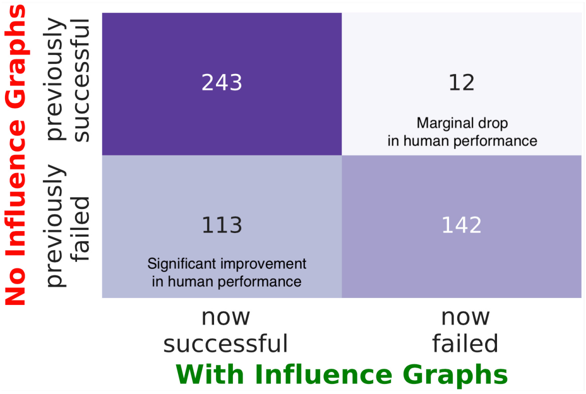

Figure 3: Human performance before and after the human judges were provided with the influence graph.

4 Discussion and Conclusion

Our work takes the idea of using inference graphs for defeasible inference and scales up its usability by automatically generating and augmenting them to a downstream defeasible task that both humans and machines are known to find difficult. We identify that the contextualizer and mediator nodes are crucial to defeasible inference, and show that our generated graphs generate these critical nodes effectively. Humans perform significantly better (20% absolute improvement) across diverse defeasible datasets and overwhelmingly attribute their success to the mediator nodes – giving insights into what helps and why. In this case study, we show that machines can fill the gaps in human knowledge when for defeasible reasoning. While we establish that humans are helped by these graphs, a further investigation on how (and if) the graphs reinforced their beliefs, and what additional information in the graphs was beneficial to their understanding is essential. Furthermore, a deeper understanding of the trade-offs (time spent in answering these questions with and without the graphs) also forms important future work.

Acknowledgments

We would like to thank Peter Clark for the thoughtful discussions, and the anonymous reviewers for valuable feedback.

This material is partly based on research sponsored in part by the Air Force Research Laboratory under agreement number FA8750-19-2-0200. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. We would like to thank Google for providing the TPU machines for conducting experiments.

References

J. Bentahar, B. Moulin, and M. Bélanger. 2010. A taxonomy of argumentation models used for knowledge representation. *Artificial Intelligence Review*, 33:211–259.

D. Israel. 1980. What's wrong with non-monotonic logic? In *AAAI*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Aman Madaan and Yiming Yang. 2021. Neural language modeling for contextualized temporal graph generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 864–881, Online. Association for Computational Linguistics.

J. McCarthy. 1981. Some philosophical problems from the standpoint of artificial intelligence. *Machine intelligence*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

C. Morgan. 2004. The nature of nonmonotonic reasoning. *Minds and Machines*, 10:321–360.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

J. Pollock. 1987. Defeasible reasoning. *Cogn. Sci.*, 11:481–518.

J. Pollock. 2009. A recursive semantics for defeasible reasoning. In *Argumentation in Artificial Intelligence*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Rachel Rudinger, Vered Shwartz, Jena D. Hwang, Chandra Bhagavatula, Maxwell Forbes, Ronan Le Bras, Noah A. Smith, and Yejin Choi. 2020. Thinking like a skeptic: Defeasible inference in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4661–4675, Online. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. Wiqa: A dataset for "what if..." reasoning over procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6078–6087.

A Sample input-output sequence for training $\mathcal{L}$

We now present a sample input-output sequence used to train out $\mathcal{L}$ for graph generation. The input-output sample ($\mathbf{seq_{ip}}, \mathbf{seq_{op}}$) is presented below. As mentioned in Section

1. As described in section 2.1, each input sequence $\mathbf{seq_{ip}}$ is formatted in a special template to be fed to the language model (Template (1)). We show an example of the same next for a sample from our training data. **`Premise:`** `Sunlight shines on plants. Cells with chlorophyll in them ... other parts of the plant.` | **`Situation`** `: more minerals are absorbed` | **`Less`** `: LESS sugar and oxygen being produced` | **`More`** `: MORE sugar and oxygen being produced`

2. Each output graph is encoded in as a DOT string. The output DOT sequence $\mathbf{seq_{op}}$ corresponding to the input shown above is: `strict digraph "C+ : less minerals in the soil [OR] less root system" -> "S : more minerals are absorbed" [label=hurts]; "C- :more minerals in the soil [OR] a better root system" -> "S : more minerals are absorbed" [label=helps]; "S : more minerals are absorbed" -> "M- : less conversion into sugars [OR] less oxygen produced" [label=hurts]; "S : more minerals are absorbed" -> "M+ : more conversion into sugars" [label=helps]; "S- : less minerals absorbed [OR] less root system" -> "M+ : more conversion into sugars" [label=hurts]; "M- : less conversion into sugars [OR] less oxygen produced" -> "H- : LESS sugar and oxygen being produced" [label=helps]; "M- : less conversion into sugars [OR] less oxygen produced" -> "H+ : MORE sugar and oxygen being produced" [label=hurts]; "M+ : more conversion into sugars" -> "H+ : MORE sugar and oxygen being produced" [label=helps]; "M+ : more conversion into sugars" -> "H- : LESS sugar and oxygen being produced" [label=hurts];`

B IRB Exemption

Our study was not an experimentation on humans (posed no identifiable risk to the human judges), did not collect any identifying information, and ensured it involved only adults. As per the IRB guidelines, this falls under the purview of human research, and we are not publishing individual workers' answers but rather the data is tallied up, much like a "benign behavioral intervention." This exempts us from IRB (category 3 of Federal Regulations for Protection of Human Research Subjects https://www.hhs.gov/ohrp/regulations-and-policy/regulations/45-cfr-46/).

C Infrastructure and hyperparameters

To train the T5-11B model, comprising of 11 billion parameters, we used v3-8 TPUs. The average time to train was 7 hours for about 10 epochs. We used the same hyperparameters as provided with the T5 checkpoint at `gs://t5-data/pretrained_models/11B`. We used maximum block size of 512 tokens, and max generation length set to 512. For decoding, we sample according to predicted distribution. We train the GPT-2 model on a Nvidia GTX 2080 Ti, and training the model takes about 30 minutes per epoch.

We use the medium (355M) variant of GPT-2 (Radford et al., 2019) with 24 layers, 1024 hidden size, 16 attention heads.

D Details of our Mechanical Turk Setup

We follow the same instructions for humans as (Rudinger et al., 2020)[5], and only additionally provided instructions for the inference graph. We used a pool of 230 annotators that were previously qualified and selected to do the defeasible inference task, thus providing a fair comparison to their setup. Eventually 12 workers out of these 230 workers

---

[5] We are grateful to the authors of (Rudinger et al., 2020) for sharing their mechanical turk setup template with us.

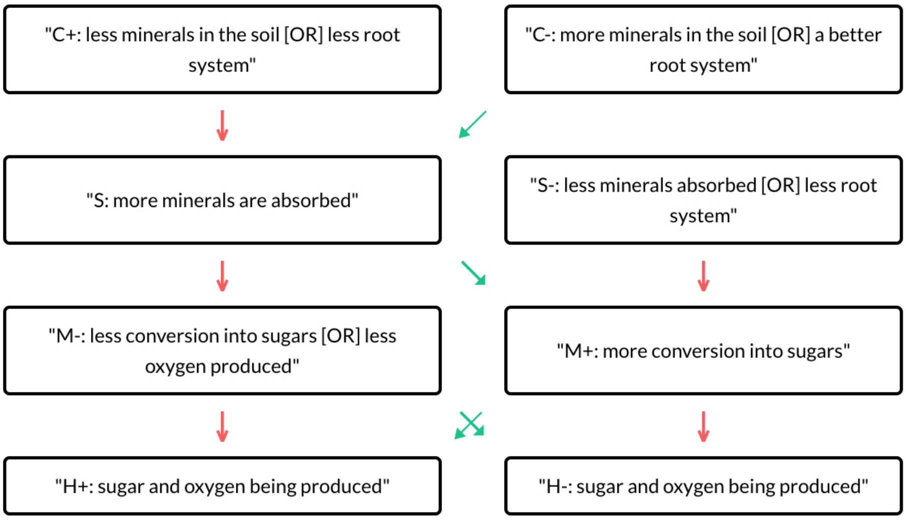

Figure 4: The influence graph corresponding to dot code shown in $\mathbf{seq_{op}}$

worked on our HITs. The graph we showed to humans was a subgraph of the inference graph, where the selected path has the relevant content from the inference graph to avoid showing redundant opposite edges. These redundant edges are useful in training a model as the model must jointly predict all the nodes, but this is redundant for humans. Figure 5 shows this subgraph.

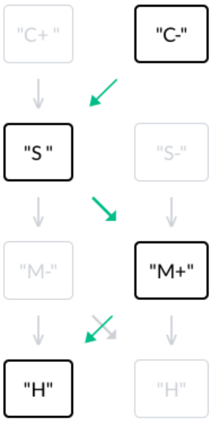

Figure 5: Part of the generated influence graph that is presented in the hit.

D.1 A sample HIT

We now show a sample HIT in Figure 6. We had two set of annotations in every HIT.

D.2 Examples that helped humans

Next, we show two examples (Figure 7, Figure 8) where humans were previously unsuccessful on this answer (in the original setup of (Rudinger et al., 2020)), and were successful now having looked at the inference graphs. The humans marked that the mediator nodes and the contextualizer nodes provide useful information.

E Dataset

Dataset	Split	# Samples	Total
WIQA	train test dev	1522 189 152	2107
ATOMIC	train test dev	35,001 4137 3839	42,977
SOCIAL	train test dev	88,675 1836 1784	92,295
SNLI	train test dev	77,015 9438 9342	95,795

Table 5: Number of samples in each dataset by split. ATOMIC, SNLI, SOCIAL are available at `https://github.com/rudinger/defeasible-nli`, WIQA is avilable at `https://allenai.org/data/wiqa`

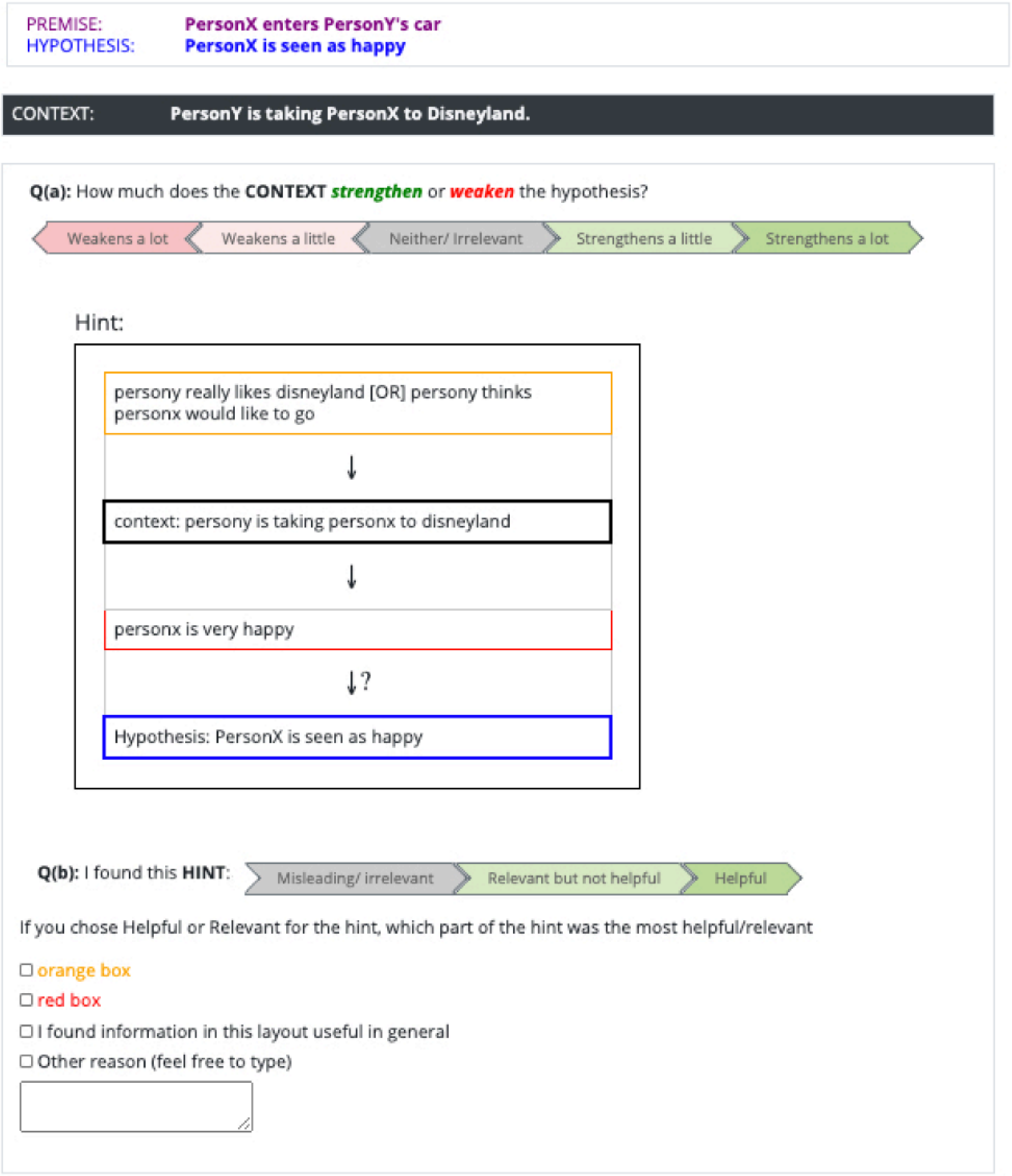

PREMISE: PersonX enters PersonY's car
HYPOTHESIS: PersonX is seen as happy

CONTEXT: PersonY is taking PersonX to Disneyland.

Q(a): How much does the **CONTEXT** ***strengthen*** or ***weaken*** the hypothesis?

Hint:

persony really likes disneyland [OR] persony thinks personx would like to go

↓

context: persony is taking personx to disneyland

↓

personx is very happy

↓?

Hypothesis: PersonX is seen as happy

Q(b): I found this **HINT**:

If you chose Helpful or Relevant for the hint, which part of the hint was the most helpful/relevant

☐ orange box
☐ red box
☐ I found information in this layout useful in general
☐ Other reason (feel free to type)

Figure 6: A sample HIT in mechanical turk.

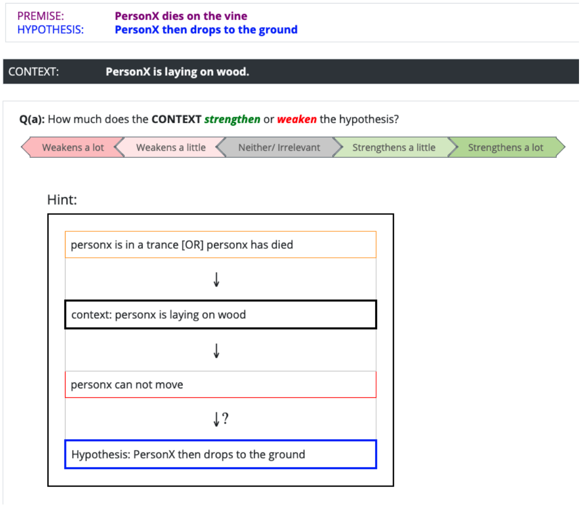
PREMISE: **PersonX dies on the vine**
HYPOTHESIS: **PersonX then drops to the ground**

CONTEXT: **PersonX is laying on wood.**

Q(a): How much does the **CONTEXT** ***strengthen*** or ***weaken*** the hypothesis?

Hint:

personx is in a trance [OR] personx has died
↓
context: personx is laying on wood
↓
personx can not move
↓?
Hypothesis: PersonX then drops to the ground

Figure 7: An example where the graph helped the human in getting the correct answer, that humans were unsuccessful on, in the past.

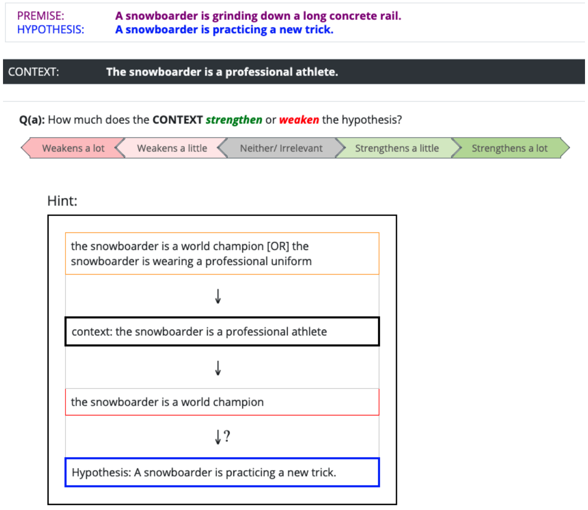
PREMISE: **A snowboarder is grinding down a long concrete rail.**
HYPOTHESIS: **A snowboarder is practicing a new trick.**

CONTEXT: **The snowboarder is a professional athlete.**

Q(a): How much does the **CONTEXT** ***strengthen*** or ***weaken*** the hypothesis?

Hint:

the snowboarder is a world champion [OR] the snowboarder is wearing a professional uniform
↓
context: the snowboarder is a professional athlete
↓
the snowboarder is a world champion
↓?
Hypothesis: A snowboarder is practicing a new trick.

Figure 8: Another example where the graph helped the human in getting the correct answer, that humans were unsuccessful on, in the past.

Characterizing Social Spambots by their Human Traits

Salvatore Giorgi
University of Pennsylvania
`sgiorgi@sas.upenn.edu`

Lyle Ungar
University of Pennsylvania
`ungar@cis.upenn.edu`

H. Andrew Schwartz
Stony Brook University
`has@cs.stonybrook.edu`

Abstract

Social spambots, an emerging class of spammers attempting to emulate people, are difficult for both human annotators and classic bot detection techniques to reliably distinguish from genuine accounts. We examine this human emulation through studying the human characteristics (personality, gender, age, emotions) exhibited by social spambots' language, hypothesizing the values for these attributes will be unhuman-like (e.g. unusually high or low). We found our hypothesis mostly disconfirmed — *individually,* social bots exhibit very human-like attributes. However, a striking pattern emerged when consider the full distributions of these estimated human attributes: social bots were extremely similar and average in their expressed personality, demographics, and emotion (in contrast with traditional bots which we found to exhibit more variance and extreme values than genuine accounts). We thus consider how well social bots can be identified only using the 17 variables of these human attributes and ended up with a new state of the art in social spambot detection (e.g. $F1 = .946$). Further, simulating the situation of not knowing the bots *a priori*, we found that even an unsupervised clustering using the same 17 attributes could yield nearly as accurate of social bot identification ($F1 = 0.925$).

1 Introduction

A *social spambot* is "a computer algorithm that automatically produces content and interacts with humans on social media, trying to emulate and possibly alter their behavior" (Ferrara et al., 2016). Previous studies have shown that standard spambot detection algorithms as well as human annotators, while quite effective at detecting standard spam (e.g. sales advertisements), fail to accurately distinguish these human emulating *social spambots* from genuine Twitter accounts (Cresci et al., 2017).

In light of the goal of social spambots to emulate human behavior, we test the assumption that social spambots should not behave differently than human accounts and that a shared set of traits *should* be common among both groups. Thus, in this study we attempt to characterize and classify social spambots, in comparison to genuine Twitter accounts and traditional spambot accounts, by real human traits and states. We estimate demographics (age and gender), personality traits (openness to experience, conscientiousness, extraversion, agreeableness, and neuroticism), sentiment (positive and negative), and emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust) using a variety of pre-trained models. First and foremost, we describe the distributions of these 17 human attributes, and then we evaluate them as features in relatively simple spambot detection classifiers (random forest) and compare to more sophisticated state of the art detection methods.

Contributions Our contributions: (1) We characterize social spambots in terms of estimated human traits: demographics, personality, sentiment, and emotion. We show that social spambots express limited gender, age, and emotional variation while being much higher in positive sentiment and neurotic language than genuine users. (2) We show that traditional bots exhibit wider variance and more extreme estimated human traits. (3) We use these 17 human traits in a social spambot detection model, achieving state of the art classification results, in addition to outperforming human annotators.

2 Related Work

Bots and bot detection methods have received increased attention due to their role in spreading information, both real and fake, on social media platforms (Shao et al., 2018; Caldarelli et al., 2020). Particular focus has been given to their role in

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 5148–5158
August 1–6, 2021.

the 2016 U.S. Presidential election (Bessi and Ferrara, 2016; Badawy et al., 2018), but bots have been discovered discussing other political events such as the Syrian war (Abokhodair et al., 2015) and Brexit (Bastos and Mercea, 2019). Additionally, the spread of misinformation from bots has been linked to multiple public health issues such as vaccines (Broniatowski et al., 2018; Yuan et al., 2019), e-cigarettes (Allem et al., 2017), and marijuana (Allem et al., 2020). There is also growing evidence that bots are spreading information about COVID-19 (Himelein-Wachowiak et al., 2021; Ferrara, 2020; Al-Rawi and Shukla, 2020). The prevalence of bots in online discourse around elections and public policy has lead some researchers to begin outlining government policy for dealing with bots (Pedrazzi and Oehmer, 2020).

Bots have had such far reaching societal impact that research has grown significantly over the last decade (Cresci, 2020), with government agencies sponsoring competitions to identify the influence of bots (Subrahmanian et al., 2016). Although much bot research was done in the context of traditional (non-social or content generating) spambots, interest in social spambots has emerged more recently mostly focused on content, online behavior, or network attributes, rather than the human-likeness of the bots (Zhang et al., 2016). For example, Kudugunta and Ferrara (2018) applied deep neural architectures to the task of classifying social spambots from a single tweet. Using an adversarial learning approach Cresci et al. (2019b) generated evolved versions of current social spambots and, subsequently, attempted to classify them as bot or human. They again show that previous methods, which have been successful in identifying traditional bots, fail to detect the evolved social bots, including their own previous state-of-the-art method. Other studies have used anomaly detection (Miller et al., 2014) as well as sentiment based methods (Dickerson et al., 2014).

Our work is aligned with a growing set of methods to to embed language processing within the social and human contexts they are applied (Lynn et al., 2019). Most similar is the work on language generation or dialog agents (i.e. chatbots). While not directly related to spambot detection, such work has attempted to produce agents with empathy (Rashkin et al., 2019), trust (Novick et al., 2018) and emotion (Zhou and Wang, 2018; Huber et al., 2018) as well as general personalizations (Li et al., 2016; Zhang et al., 2018; Mazaré et al., 2018). As researchers build machines imbued with more sophisticated human attributes, we can expect similar machines to be used for spamming purposes.

3 Data Sets

We sought to use a variety of Twitter bot corpora in order to cover multiple contexts in which bots have been used. These contexts included social and non-social spambots (i.e., content generators and fake followers). In total, we use two social spambot data sets and four traditional bot data sets, each of which is briefly described below. While the focus of the current paper is social spambots, we investigate the human traits of traditional bots to show (1) that our methods generalize across different types of bots and (2) show that social spambots are indeed more sophisticated than traditional bots.

Spambot Data We use the two social spambot data sets derived from Cresci et al. (2017). The first data set, *SSB1* (Social SpamBots #1), is identical Cresci et al's test data: 464 social spambots, who advertised products on *Amazon.com*, plus 464 genuine accounts (718,975 total tweets). As a second data set, *SSB2*, we use an additional 2,913 genuine users from Cresci et al that were not part of *SSB1*, as well as 2,913 randomly selected social spambot accounts that Cresci found to be promoting a VIP version of the *Talnts* app. Because the number of tweets from the genuine accounts was much larger than that of the social spambots, we randomly sampled the genuine tweets, so that the tweet set was evenly split (2,621,684 total tweets). Appendix A contains a selection of social bot tweets to demonstrate how realistic they seem.

Traditional spambots We apply human trait estimates to four open source bot data sets, which consist of non-social bots (i.e., traditional / content generating bots and fake followers), in order to determine if these features generalize across data sets and bot types. All data sets are available through the Bot Repository [1]. We briefly summarize the data sets here (see source papers for more details) which contain both genuine and bot accounts. Yang et al. (2020): A data set of self-identified Twitter bots and verified accounts. Cresci et al. (2019a): Twitter accounts with suspicious financial tweets promoting low-value stocks. Cresci et al. (2015):

---

[1] `https://botometer.osome.iu.edu/bot-repository/`

Fake Twitter followers created to inflate the number of followers of popular accounts. Lee et al. (2011): A collection of content polluting Twitter accounts discovered via honeypot traps.

4 Estimating Human Traits

For each user in our data sets we estimate the following traits from their tweets: age, gender, personality, sentiment, and emotion.

Age / Gender. We applied a predictive lexicon to produce real valued age and gender estimates (Sap et al., 2014). This lexicon was built over Twitter, Facebook, and blog users with labeled age and gender, and produced prediction accuracies (Pearson r) of .831 (age) and .919 (gender).

Personality. We used a language based personality model to estimate the Big Five personality traits: openness to experience, conscientiousness, extraversion, agreeableness and neuroticism (Park et al., 2015). This model was built over 1-3grams and a set of 2,000 LDA (Latent Dirichlet Allocation; Blei et al. 2003) topics and resulted in prediction accuracies (Pearson r) of .43 (openness), .37 (conscientiousness), .42 (extraversion), .35 (agreeableness) and .35 (neuroticism). Park et al. showed that these prediction accuracies are higher than correlations between self-report personality and personality ratings by friends.

Sentiment. We used the positive/negative sentiment categories from the NRC Word-Emotion Association Lexicon, a crowd sourced, word level lexicon (Emolex; Mohammad and Turney 2013).

Emotion. To estimate emotion we used the NRC Hashtag Emotion Lexicon (Mohammad and Kiritchenko, 2015), which has categories based on Plutchik's eight basic emotions: anger, anticipation, disgust, fear, joy, sadness, surprise and trust (Plutchik, 1980). This is an automatically created, word-level lexicon based on tweets with emotion hashtags, i.e. *#anger*, *#anticipation*, etc.

5 Methods

Features For both the *SSB1* and *SSB2* data sets we extract all features needed to estimate the human traits. The age, gender, sentiment, and emotion lexica all use unigrams. For personality, we extract 1-3grams and a set of 2,000 LDA topics. These topics have been used across a number of studies on age, gender, and personality (Schwartz et al., 2013; Park et al., 2016) and were built over the MyPersonality data set (Kosinski et al., 2015).

	Technique	F1	Prec	Recall	Accuracy	MCC
Past Work	Humans	.570	.647	.509	.829	.470
	BotOrNot? (2016)	.761	.635	.950	.922	.738
	Ahmed et al. (2013)	.923	.913	.935	.923	.847
	Cresci et al. (2017)	.923	**1.000**	.858	.929	.867
This Work	Age & Gender	.578	.585	.582	.581	.167
	Personality	.899	.900	.899	.899	.800
	Sentiment	.833	.850	.835	.836	.684
	Emotions	.331	.248	.500	.500	.000
	All Human Traits	**.946***	.946	**.946***	**.946***	**.892**

Table 1: Classification results. First 4 lines presented in Cresci et al. (2017), bottom 5 lines are models presented in this work. * significantly different than Cresci et al. from a bootstrapped p-value < 0.01. MCC is the Matthews correlation coefficient.

5.1 Social Spambot Classification

Here the *SSB1* and *SSB2* data sets are used to train and test our model, respectively. In order to keep modeling simple, we built a random forest classifier, utilizing extremely randomized trees (Geurts et al., 2006), as implemented in by scikit-learn (Pedregosa et al., 2011). Modeling parameters are listed in Appendix D .

Comparisons We compare our model accuracies to human annotators and current state of the art methods, which are evaluated on our test set (*SSB2*) and reported in Cresci et al. (2017).

Human Annotators: a crowd sourcing task to test whether humans are able to distinguish social spambots from genuine accounts (Cresci et al., 2017).

BotOrNot: a supervised method trained to identify social spambots (Davis et al., 2016). The model uses over 1,000 different features including tweet content, sentiment and user meta data.

Ahmed et al.: an unsupervised graph clustering of features such as URLs, hashtags, mentions, and retweets (Ahmed and Abulaish, 2013).

Cresci et al.: uses DNA-inspired techniques to model behavior of Twitter users in an unsupervised fashion (Cresci et al., 2016).

5.2 Traditional Spambot Classification

In order to access the generalizability of our features, we evaluate a similar classification task using three open source bot data sets, which we briefly describe below (see the source papers for more details). In the previous task we had dedicated train and test data sets, in order to directly compare against the results in Cresci et al. (2017). Here, we do not have a consistent baseline therefore use a 10-fold cross validation setup, and use a model built on the top 1,000 most frequent unigrams as a comparison. Again, we use an extremely randomized trees classifier; see Appendix E for full modeling and language processing details.

5.3 Unsupervised Classification

Finally, we apply an unsupervised clustering to our data in order to label each account as bot or human. We apply spectral clustering (Von Luxburg, 2007) to all 17 features (age, gender, personality, emotion, and sentiment) as well as the 5 personality dimensions and a baseline of 1,000 unigrams. The spectral clustering algorithm is implemented in scikit-learn with default settings. To simulate a more practical setting, where the human-to-bot ratio is not 50/50, we apply the clustering method to a 90/10 human-to-bot ratio. We fix the number of human accounts to the maximum number in both *SSB1* and *SSB2* (i.e., 2,913 and 464, respectively), and randomly sample bots so as to make a 90/10 human-to-bot ratio. For the 50/50 and 90/10 ratio we set the number of clusters to 2 and 10, respectively. We use 10 clusters on the unbalanced data since spectral clustering assumes equal cluster sizes. To chose the label for the bot cluster, we calculate the standard deviation of each human traits for each cluster, average the standard deviations within each cluster, and label the cluster with the minimum average standard deviation as "bots".

6 Results

Demographic, personality, sentiment, and emotion distributions are shown in Figures 1 and 2. The age estimates for the spambots (red) are tightly clustered (age mean = 28.9, standard deviation = 2.58), when compared to the genuine accounts (blue; mean = 23.8, standard deviation = 5.14). A similar pattern holds for gender – mean (standard deviation) gender is -.50 (.53) and .26 (1.17) for spambots and genuine accounts, respectively. The other traits show patterns similar to age and gender: social spambots have little variation in personality, sentiment, and emotion. Of note are neuroticism and positive sentiment distributions.

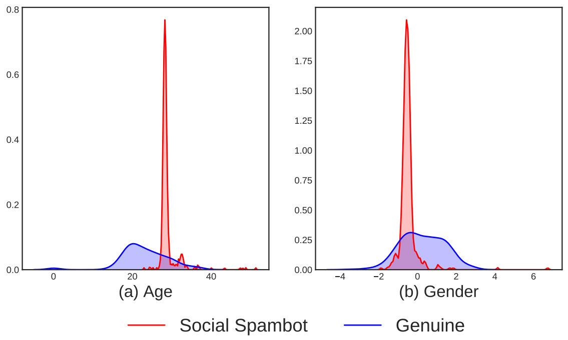

(a) Age (b) Gender

Figure 1: Age and gender distributions of genuine (blue) and social spambot (red) accounts in the *SSB2* data set.

	Number of Humans/Bots	Feats.	F1	Prec.	Recall
Yang et al. (2020)	1971 / 670	unigrams	.978	.985	.971
		human traits	.968	.979	.957
Cresci et al. (2019a)	584 / 5,022	unigrams	.941	.921	.966
		human traits	.923	.928	.918
Cresci et al. (2015)	1,075 / 297	unigrams	.921	.969	.888
		human traits	.901	.948	.870
Lee et al. (2011)	17,720 / 14,632	unigrams	.851	.854	.849
		human traits	.862	.865	.862

Table 2: Traditional bot cross validation classification results for 1,000 unigrams and 17 human traits.

Classification results are presented in Table 1. The first four lines ("Past Work") contain results evaluated in Cresci et al. 2017 and are included here for context, while the current study's results are presented in the final five lines. Personality out performs all other single feature sets (age/gender, sentiment and emotion) across all of our evaluation metrics, while emotion performs the worst. Combining all features gives the best classification performance, significantly ($p < 0.01$) above the state of the art in Cresci et al. 2016.

Next, we evaluate estimated human traits for distinguish bots in a number of additional open source bot data sets. Since existing models were largely unavailable for these, we compare to a a much larger model built on the 1,000 most frequent unigrams. Cross validation classification results are in Table 2. The 17 human traits perform almost as well as the 1,000 unigram features, outperforming unigrams in the Lee et al. (2011) data set. The unigram models use (1) features which vary across each data set (i.e., the frequency of each unigram is calculated *within* each data set) and (2) a much larger number of features (1,000 vs. 17). This strongly suggests that the human traits are both generalizable across data sets and also bot types.

Finally, Table 3 shows the results of the unsupervised clustering task. Here, we see that even without providing the means for the models to learn the ideal ranges of attributes (i.e. from training data), the social bots primarily come out in one class with the genuine accounts in the other, even when we sample for a higher ratio of genuine-to-bot accounts (90/10).

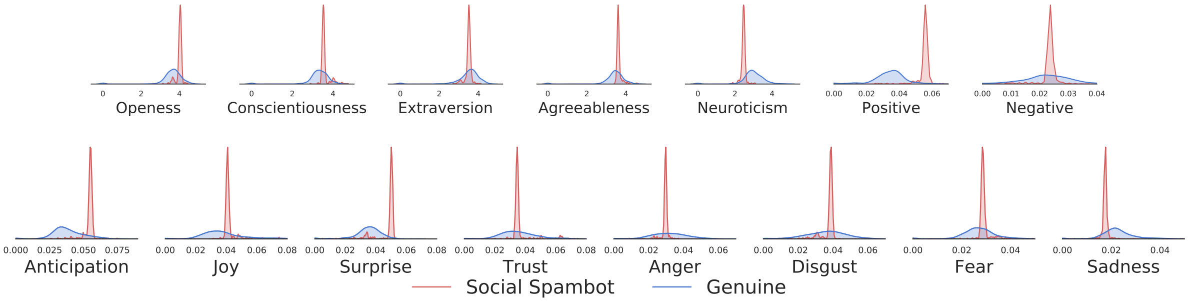

Figure 2: Personality and Emotion distributions of genuine (blue) and social spambot (red) accounts in the *SSB2* data set. Social spambots exhibited reasonable values but with little variance.

7 Ethics Statement

All bot data used in this study are publicly available. This study was reviewed by an academic institutional review board and found to be exempt, non-human subjects data. Still, ethical considerations should be raised when estimating demographics and personality from social media. These include privacy issues, biases in training data, the impact of misclassifications on downstream tasks, and outdated definitions of social constructs such as binary gender. While imperfect, we believe that these estimates are non-obtrusive and allow researchers to study average differences in estimated demographics as expressed in public language.

Further consideration was taken into account given the effectiveness of this approach for bot detection. Notably there is likely a continued arms race between bot creators and bot detectors: as natural language generation methods and bots become more advanced so too will detection methods. When published, it should be assumed that bot creators will try to reverse engineer the methods. Under open security principles, we hope the results of this study promote better detection, but caution that the effectiveness of the approach may ween over time.

		50/50 split			90/10 split		
		F1	Prec.	Recall	F1	Prec.	Recall
	MFC	.333	.248	.500	.474	.450	.500
SSB1	Unigrams	.334	.750	.500	.695	.665	.842
	Personality	**.832**	**.836**	**.832**	**.747**	**.712**	**.812**
	All Human Traits	.639	.803	.677	.472	.450	.498
SSB2	Unigrams	.336	.748	.502	.469	.450	.490
	Personality	.922	.931	.923	.949	.923	.980
	All Human Traits	**.925**	**.934**	**.926**	**.971**	**.995**	**.951**

Table 3: Unsupervised clustering classification results for both social spambot data sets using Most Frequent Class (MFC), personality, and all human traits.

8 Conclusion

In this study we showed that social spambots, while difficult for humans to detect, suffer from a lack of variation across all of our measures: age, gender, personality, sentiment, and emotion. Not only did social spambots seem to lack variation in most human traits, their mean values often aligned with the means of genuine accounts (with the exceptions of high mean positive sentiment and low neuroticism – both still with little variance).

The bots' lack of variation across any trait suggests they are "clone"-like. This could explain the inability of human annotators to properly identify them: *individual* social spambot accounts may look human (e.g., 29 years old and slightly male) but the range of humanness is limited. On the *population level*, most social spambots appear to be a clone of the same "human". This contrasts with traditional bots, whose traits were more varied and often with extreme, inhuman values (e.g., negative ages; see Appendix C), suggesting a "robot"-like interpretation of traditional bots.

The consistency of the estimated human traits was strong enough that we were able to build simple yet highly predictive classifiers using only the 17 estimates as features, outperforming human annotators and significantly more accurate than the state of the art in Cresci et al. (2017). On additional data sets spanning other types of bots (e.g., fake followers, spamming accounts, and self identifying bots), the human traits performed on par with much larger models (1,000 unigrams). Finally, simulating where bot training data is non-existant, we showed a simple unsupervised clustering based on the 17 attributes identified social bots nearly as accurately, suggesting human trait based approaches may be able to identify new bots in the wild.

References

Norah Abokhodair, Daisy Yoo, and David W McDonald. 2015. Dissecting a social botnet: Growth, content and influence in twitter. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 839–851.

Faraz Ahmed and Muhammad Abulaish. 2013. A generic statistical approach for spam detection in online social networks. *Computer Communications*, 36(10-11):1120–1129.

Ahmed Al-Rawi and Vishal Shukla. 2020. Bots as active news promoters: A digital analysis of covid-19 tweets. *Information*, 11(10):461.

Jon-Patrick Allem, Patricia Escobedo, and Likhit Dharmapuri. 2020. Cannabis surveillance with twitter data: emerging topics and social bots. *American journal of public health*, 110(3):357–362.

Jon-Patrick Allem, Emilio Ferrara, Sree Priyanka Uppu, Tess Boley Cruz, and Jennifer B Unger. 2017. E-cigarette surveillance with social media data: social bots, emerging topics, and trends. *JMIR public health and surveillance*, 3(4):e98.

Adam Badawy, Emilio Ferrara, and Kristina Lerman. 2018. Analyzing the digital traces of political manipulation: The 2016 russian interference twitter campaign. In *2018 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM)*, pages 258–265. IEEE.

Marco T Bastos and Dan Mercea. 2019. The brexit botnet and user-generated hyperpartisan news. *Social science computer review*, 37(1):38–54.

Alessandro Bessi and Emilio Ferrara. 2016. Social bots distort the 2016 us presidential election online discussion. *First Monday*, 21(11-7).

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

David A Broniatowski, Amelia M Jamison, SiHua Qi, Lulwah AlKulaib, Tao Chen, Adrian Benton, Sandra C Quinn, and Mark Dredze. 2018. Weaponized health communication: Twitter bots and russian trolls amplify the vaccine debate. *American journal of public health*, 108(10):1378–1384.

Guido Caldarelli, Rocco De Nicola, Fabio Del Vigna, Marinella Petrocchi, and Fabio Saracco. 2020. The role of bot squads in the political propaganda on twitter. *Communications Physics*, 3(1):1–15.

Stefano Cresci. 2020. A decade of social bot detection. *Communications of the ACM*, 63(10):72–83.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems*, 80:56–71.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2016. Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems*, 31(5):58–64.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2017. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 963–972. International World Wide Web Conferences Steering Committee.

Stefano Cresci, Fabrizio Lillo, Daniele Regoli, Serena Tardelli, and Maurizio Tesconi. 2019a. Cashtag piggybacking: Uncovering spam and bot activity in stock microblogs on twitter. *ACM Transactions on the Web (TWEB)*, 13(2):1–27.

Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. 2019b. Better safe than sorry: An adversarial approach to improve social bot detection. *arXiv preprint arXiv:1904.05132*.

Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 273–274. International World Wide Web Conferences Steering Committee.

John P Dickerson, Vadim Kagan, and VS Subrahmanian. 2014. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 620–627. IEEE.

Emilio Ferrara. 2020. What types of covid-19 conspiracies are populated by twitter bots? *First Monday*.

Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Communications of the ACM*, 59(7):96–104.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning*, 63(1):3–42.

McKenzie Himelein-Wachowiak, Salvatore Giorgi, Amanda Devoto, Muhammad Rahman, Lyle Ungar, H Andrew Schwartz, David H Epstein, Lorenzo Leggio, and Brenda Curtis. 2021. Bots and misinformation spread on social media: Implications for covid-19. *J Med Internet Res*, 23(5):e26933.

Bernd Huber, Daniel McDuff, Chris Brockett, Michel Galley, and Bill Dolan. 2018. Emotional dialogue generation using image-grounded language models. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 277. ACM.

Michal Kosinski, Sandra C Matz, Samuel D Gosling, Vesselin Popov, and David Stillwell. 2015. Facebook as a research tool for the social sciences: Opportunities, challenges, ethical considerations, and practical guidelines. *American Psychologist*, 70(6):543.

Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences*, 467:312–322.

Kyumin Lee, Brian Eoff, and James Caverlee. 2011. Seven months with the devils: A long-term study of content polluters on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Veronica Lynn, Salvatore Giorgi, Niranjan Balasubramanian, and H. Andrew Schwartz. 2019. Tweet classification without the tweet: An empirical examination of user versus document attributes. In *Proceedings of the Third Workshop on Natural Language Processing and Computational Social Science*, pages 18–28, Minneapolis, Minnesota. Association for Computational Linguistics.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*.

Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. 2014. Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

David Novick, Mahdokht Afravi, Adriana Camacho, Laura J Hinojos, and Aaron E Rodriguez. 2018. Inducing rapport-building behaviors in interaction with an embodied conversational agent. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, pages 345–346. ACM.

Gregory Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Michal Kosinski, David J Stillwell, Lyle H Ungar, and Martin EP Seligman. 2015. Automatic personality assessment through social media language. *Journal of personality and social psychology*, 108(6):934.

Gregory Park, David Bryce Yaden, H Andrew Schwartz, Margaret L Kern, Johannes C Eichstaedt, Michael Kosinski, David Stillwell, Lyle H Ungar, and Martin EP Seligman. 2016. Women are warmer but no less assertive than men: Gender and language on facebook. *PloS one*, 11(5):e0155885.

Stefano Pedrazzi and Franziska Oehmer. 2020. Communication rights for social bots?: Options for the governance of automated computer-generated online identities. *Journal of Information Policy*, 10:549–581.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. In *Theories of emotion*, pages 3–33. Elsevier.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381.

Maarten Sap, Greg Park, Johannes C Eichstaedt, Margaret L Kern, David J Stillwell, Michal Kosinski, Lyle H Ungar, and H Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. 2018. The spread of low-credibility content by social bots. *Nature communications*, 9(1):1–9.

Venkatramanan S Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. The darpa twitter bot challenge. *Computer*, 49(6):38–46.

Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

Chao Yang, Robert Harkreader, and Guofei Gu. 2013. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293.

Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. 2020. Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1096–1103.

Xiaoyi Yuan, Ross J Schuchard, and Andrew T Crooks. 2019. Examining emergent communities and social bots within the polarized online vaccination debate in twitter. *Social Media+ Society*, 5(3):2056305119865465.

Jinxue Zhang, Rui Zhang, Yanchao Zhang, and Guanhua Yan. 2016. The rise of social botnets: Attacks and countermeasures. *IEEE Transactions on Dependable and Secure Computing*, 15(6):1068–1082.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Xianda Zhou and William Yang Wang. 2018. MojiTalk: Generating emotional responses at scale. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1128–1137, Melbourne, Australia. Association for Computational Linguistics.

A Spambot Sample Tweets

We include a set of random tweets from each social spambot data set, in order to show that these accounts are not immediately identifiable as bots.

	Tweet
SSB1	• To your favorite song, anything you want to do.. • Best week for a long time • Seriously 90210 ended on the best note. I was in waaay too many tears. What do I do with my life now?
SSB2	• I don't read other science fiction. I don't read any at all. - Jack Vance • Better to be a geek than an idiot. • I like my money right where I can see it – hanging in my closet. - Carrie Bradshaw

Table 4: Sample of random social spambot tweets, presented here to highlight the fact that the tweet authors are not immediately identifiable as non-human.

B Social Spambots: Training Data

Here we plot the distributions of our 17 features for the social spambots and genuine users in our training data (see main paper for test data). Demographic data (age and gender) are plotted in Figure 3. We see a similar pattern to the test data, though less pronounced: social spambot distributions are considerably more peaked than the genuine users. Also, note that genuine users have a bimodal gender distribution, corresponding to the female/male splot, whereas the spambot distribution is unimodal. Figure 4 shows the distributions for personality, sentiment and emotion. Again, we see patterns similar to the test data, though less pronounced. The train spambots are also more positive than the genuine users, though the emotional stability split, seen in the test data, is not as pronounced.

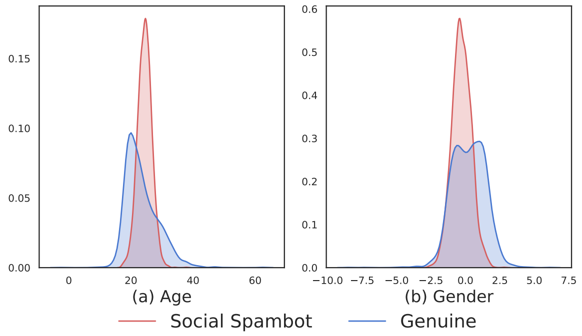

(a) Age (b) Gender

Figure 3: Demographic (age and gender) distributions of genuine and social spambot accounts in the training data.

C Human Traits: Traditional vs Social Spambots

We estimates the human traits of a set of 1,000 traditional spambots from Cresci et al. (2017), sampled from a larger set of data released in Yang et al. (2013). In Figure 5 we compare age and gender distributions to those of genuine Twitter users and social spambots. Here we see distinct distributions for each type of account, with traditional spambots having a much wider distribution. In the case of age, we see that traditional bots have non-human predictions (i.e., negative numbers), whereas the predicted age of social spambots is within a reasonable human range. The gender distribution of the traditional spambots do not show a female/male split (i.e., no distinct bimodal distribution). Thus, we have two distinct patterns: (1) traditional bots exhibit a wide range of characteristics, though often non-human and uninterpretable; and, (2) social bots exhibit a very narrow band of characteristics, though this band is within a normal human range, or at least within the range of predicted values for genuine accounts.

In Figure 6 we present the distributions for personality, sentiment, and emotion for all three classes of Twitter users: genuine accounts, traditional spambots and social spambots. First, we see three distinct distributions for each class of Twitter users across all characteristics. Similar to the age and gender plots, when comparing emotions to genuine users, the traditional bots often have a larger spread and are multi-modal. For personality and sentiment, we see a different pattern. Here the traditional bots are more spread than the social bots but still fairly limited when compared to genuine users. While the social spambots have a consistently different pattern when compared to genuine users, we see slightly more variation when comparing traditional bots to genuine users. That said, when comparing across all traits we see three fairly distinct classes of Twitter users.

D Experimental Setup for Classifying Social Bots

We used the following settings in our extremely randomized trees classifier: *bootstrap* is False, *class_weight* is None, *criterion* is gini, *max_depth* is None, *max_features* is sqrt, *max_leaf_nodes* is None, *min_impurity_split* is 1^{-7}, *min_samples_leaf* is 100, *min_samples_split* is 2, *min_weight_fraction_leaf* is 0, *n_estimators* is 1000,

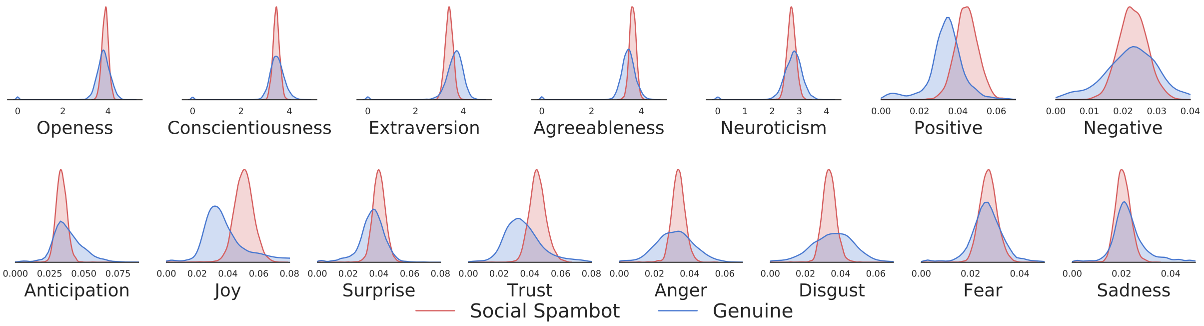

Figure 4: Personality, Sentiment and Emotion distributions of genuine accounts and social spambots in the training data.

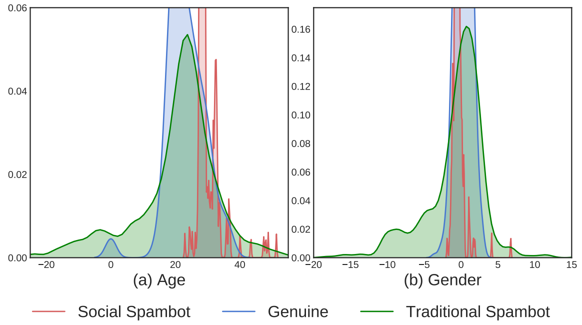

Figure 5: Demographic distributions of traditional spambots, genuine accounts and social spambots. Zoomed to highlight traditional spambots, see Figure 1 in the main paper for genuine accounts and social spambots.

n_jobs is 5, *oob_score* is False, *random_state* is None, and *warm_start* is False.

E Experimental Setup for Classifying Traditional Bots

We use our 17 human trait features in a classification task where we attempt to distinguish traditional bots from genuine human accounts. To do this we use four open source data sets: Yang et al. (2020), Cresci et al. (2019a), Cresci et al. (2015), and Lee et al. (2011). These data sets are all available at the Bot Repository[2], which is a centralized repository for open source bot data sets. Due to privacy reasons, the majority of the data sets on this repo do not contain tweet level data, which we need to estimate the human traits. Therefore, we limit our analysis to three data sets which have tweets available on the Bot Repository: Cresci et al. (2019a), Cresci et al. (2015), and Lee et al. (2011). For the fourth data set in our task, Yang et al. (2020), we use the Twitter API[3] to download available tweet histories from the *botwiki* and *verified* described in the paper (see Yang et al. (2020) for more details).

For each of these data sets we first apply an English filter, since our human trait estimators are trained on English social media data. We use use the langid Python package (Lui and Baldwin, 2012) to restrict to English only tweets. We then extract 1-3grams and a set of 2,000 LDA topics for each account, which are then used to estimate the human traits (age, gender, personality, emotion, and sentiment). See main paper for details on feature extraction. Since we are not using the original tweet set as reported in the above papers (due to English filtering or re-download the tweets from the Twitter API), we build a comparison model using the 1,000 most frequent unigrams. Finally, we restrict our analysis to accounts which have posted at least 500 words across their English tweet histories. To classify each account as human or bot we run 10-fold cross validation using stratified folds. We use an extremely randomized trees classifier as implement in scikit-learn using the following parameters using the settings listed in the social bot detection modeling. Two models are build, one for each of the following feature sets: (1) 1,000 most frequent 1-3grams and (2) 17 human traits.

Additional Acknowledgements

We wish to thank the anonymous reviewers as well as colleagues Joao Sedoc, Johannes Eichstaedt, and David Yaden for their valuable feedback on this work. Dr. Schwartz's effort was funded in part by the Defense Advanced Research Projects Agency (DARPA), via grant #W911NF-20-1-0306 to Stony Brook University. The conclusions and opinions

---

[2] `https://botometer.osome.iu.edu/bot-repository/datasets.html`

[3] `https://github.com/dlatk/TwitterMySQL`

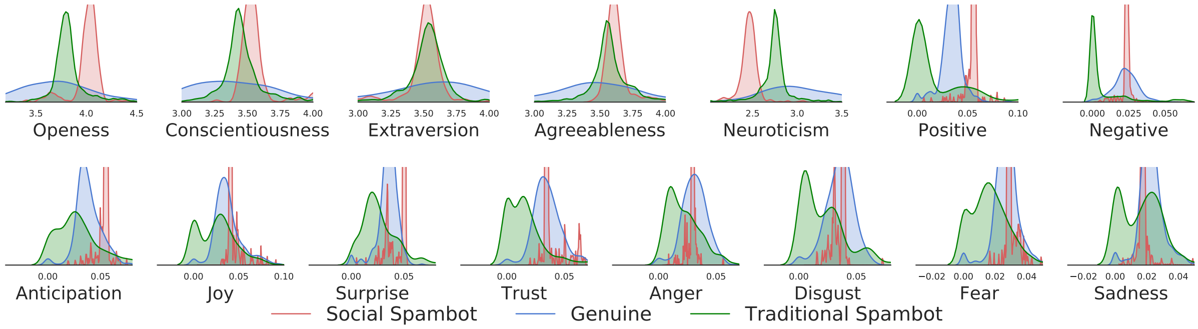

Figure 6: Personality, Sentiment and Emotion distributions of genuine accounts, traditional spambots and social spambot.

expressed are attributable only to the authors and should not be construed as those of DARPA or the U.S. Department of Defense.

Author Index